Aston University

Some pages of this thesis may have been removed for copyright restrictions.

If you have discovered material in AURA which is unlawful e.g. breaches copyright, (either yours or that of a third party) or any other law, including but not limited to those relating to patent, trademark, confidentiality, data protection, obscenity, defamation, libel, then please read our Takedown Policy and contact the service immediately

Title of Thesis

Novel Hardwired Distributive Tactile Sensing System For Medical Applications

by

Pg. Mohamad Iskandar Pg. Hj. Petra

A thesis
presented to the University of Aston in Birmingham
in fulfilment of the
thesis requirement for the degree of
Doctor of Philosophy

The University of Aston in Birmingham, United Kingdom, 2007

This copy of the thesis has been supplied on condition that anyone who consults it is understood to recognise that its copyright rest with its author and that quotation from the thesis and no information derived from it may be published without proper acknowledgement.

Declarations

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

(Mohamad Iskandar Petra)

Dedications

This thesis is dedicated to my beloved daughter, Dk Izarya Naziya, and my beloved wife, Pg. Norhazlin, who during the process of finishing this thesis was fighting for her life due to severe pneumonia from delivery of our child.

Acknowledgements

My greatest thanks to the al-Mighty who empowered me with the will and strength to undertake this research, and enabled me to complete this thesis. This achievement would not have been possible without the guidance and support of many different individuals and benefactors. First and foremost, I would like to give my sincere thanks to my supervisors, Professor Peter Brett and Dr David Holding, who have both played important roles by helping and guiding me throughout my study, and giving thorough revisions of this thesis. Great appreciation is also extended to the Government of his Majesty the Sultan of Brunei Darussalam and the University of Brunei Darussalam UBD, who financed me during my academic studies in the United Kingdom. My appreciation goes to my research colleague Mark Elliot, who gives important contribution to finishing this research, (particularly the swing part). My thanks also extended to those academic members of staff of the University of Aston in Birmingham in particular the technician of the fifth floor North Wing, Philips Travis who provided assistance during this research. And thanks also to all my colleagues, Mark Prince, Qasim Iqbal, Paul Slack, Yusef Bulaie and Saied Ali, all of whom helped me have a quality campus experience.

Especial thanks and love to my beloved parents, Pg. Hj. Petra and Pg. Hjh. Jamaliah, who throughout my studies relentlessly provided me with the moral support I needed to succeed (regardless of those "steaming phone bills from Brunei"). I also extend my gratitude to both my father and mother in-laws, Pg. Hj. Muhammad and Hjh. Norsiah. To my brother 'Aki', Pg. Jamra Weira, thank you for inspiring me to undertake this huge challenge. My family – *I love you all!!*

Last but not least, my grateful thanks to Dr Tan Kha Sheng, my Head of Department at UBD, who always believed in me, and to Abd Ghani of the High Commission of Brunei in London.

Abstract

This thesis described the research carried out on the development of a novel hardwired tactile sensing system tailored for the application of a next generation of surgical robotic and clinical devices, namely a steerable endoscope with tactile feedback, and a surface plate for patient posture and balance. Two case studies are examined and presented in this thesis. The first is a one-dimensional sensor for the steerable endoscope retrieving shape and 'touch' information. The second is a two-dimensional surface which interprets the three-dimensional motion of a contacting moving load. These cases demonstrate that the outcome of this research can be used to retrieve information from a distributive tactile sensing surface of a different configuration, and can interpret dynamic and static disturbances. This novel approach to sensing has the potential to discriminate contact and palpation in minimal invasive surgery (MIS) tools, and posture and balance in patients.

The distributive tactile sensing method is a method which is demonstrating advantage over conventional approaches as the high robustness and low complexity of construction. It is beneficial to many applications requiring evaluation of contact over large surface areas with few sensing points. The methods is able to output information requiring that interpreting IT tool and this has great potential in medical applications. In this work the function is produced in hardwired form.

The hardwired technology uses an embedded system based Field Programmable Gate Arrays (FPGA) as the platform to perform the sensory signal processing part in real time. High speed robust operation is an advantage from this system leading to versatile application involving dynamic real time interpretation as described in this research.

In this research the sensory signal processing uses neural networks to derive information from input pattern from the contacting surface. Three neural networks architectures namely single, multiple and cascaded were introduced in the investigation in attempt to find the optimum solution for discrimination of the contacting outputs. These architectures were modelled and implemented into the FPGA. With the recent introduction of modern digital design flows and synthesis tools that essentially take a high-level sensory processing behaviour specification for a design, fast prototyping of the neural network function can be achieved easily. This thesis outlines the challenge of the implementations and verifications of the performances.

Keywords

Distributive Tactile Sensing, Neural Network FPGA Implementation, Activation Function

Table of Contents

Chapter 7

Table of Figures

Table of Figures

Table of Figures

Table of Figures

Glossary of Terms

Abbreviation

ADC: Analogue to Digital Converter

ANN: Artificial Neural Network

ASIC: Application Specific Integrated Circuit

BP: Backpropagation

DAC: Digital to Analogue Converter

DAQ: Data Acquisition Card

FBG: Fibre Bragg Grating

FPGA: Field Programmable Gate Arrays

FFMLP: Feed-Forward Multi-Layered Perceptron

IC: Integrated Chip

IP: Intellectual Property

LUT: Look-Up Table

MIS: Minimal Invasive Surgery

PE: Processing element

RBA: Robotic-Based Approach

RBF: Radial Basis Function

SOCT: System On a Chip Technology

tanh: Hyperbolic tangent

VHDL: Very High Speed Integrated Circuit Hardware Description Language

XSG: Xilinx System Generator

Nomenclature

I_B : Actual beam length

i_B : Incremental length along beam

a_g: Acceleration due to gravitational force (9.81 ms^{-2})

EI: Flexural rigidity

E: Elastic modulus

I: Moment of inertia

v: Vertical deflection

M_B: Bending moment

t_i : Targeted output

Γ : Total number of training set

τ : Incremental number from training set

G: Gain of strain gauge amplifier

k_Ω: Resistance to voltage configuration factor

$\Delta R_\Omega / R_\Omega$: Ratio of change of resistance form deflection to the resistance of the strain gauge,

γ : Distance from the neutral axis

V_{ex}: Excitation voltage

V_{off} : Offset voltage

G_f : Gauge factor

i_p : Normalized working range of the beam

i_m : Discriminated position of normalized working range

D, W and S : Actual load, width and shape

$\overline{D}$, $\overline{W}$ and $\overline{S}$: Normalized load, width and shape

d, w and s : Discriminated load, width and shape

i : Output index of neural network

j : Hidden nodes index of neural network

k : Input index of neural network

N : Total number of inputs to neural network

Glossary of Terms

M : Total number of hidden neurons in neural network

L : Total number of outputs from neural network

x_k : Input to neural network

q_j : Input to activation function

r_j : Output from activation function

y_i : Computed output from neural network

Q_j : Normalizing factor of input layer

Y_i : Normalizing factor of output layer

$\omega_{jk}^{(1)}$: Weight of input layer

$b_j^{(1)}$: Bias of input layer

$\omega_{ij}^{(2)}$: Weight of output layer

$b_i^{(2)}$: Bias of the output layer

$W_{jk}^{(1)}$: Normalized weight of input layer

$B_j^{(1)}$: Normalized bias of input layer

$W_{ij}^{(2)}$: Normalized weight of the output layer

$B_i^{(2)}$: Normalized bias of the output layer

g: Gradients constants of Gradient scheme approximation

i^g : Vertical axis intercept of Gradient scheme approximation

k^{poly} : Arbitrary constant for Polynomial scheme approximation

q^{pade} : Maximum stability range for Padé approximation

ρ : Convergence rate

P : Shape index (refer to equation 3.17 & 3.18)

σ : Normalised range for activation function (Gradient, Polynomial and Padé scheme)

$\hat{Q}$: Global de-normalisation of Polynomial Scheme

$\vec{x}, \vec{y}, \vec{z}$: Displacement in Cartesian coordinate made by the swing

Glossary of Terms

$\vec{X}', \vec{Y}'$: Length of the plate

$\vec{x}', \vec{y}'$: Index of length of the plate

t_r : Plate thickness

υ : Poisson's ration

φ : Angle between legs

ψ : Angle between axis and the leg

I_p : Length of the pendulum leg

I'_p : Length between the load and the axis of the pendulum

Definitions of Terms

Single Neural Network: A feed forward multi-layered perceptron (FFMLP) neural network. All of the output node(s) are connected to the hidden nodes to allow the output(s) to be computed concurrently.

Multiple Neural Network: A neural network which uses multiple separate single neural networks to compute outputs(s) concurrently but separately.

Cascaded Neural Network: A neural network in which separate networks are used to determine each output individually, but in cascade, such that the inputs to the next network consisted of the sensor inputs and the output(s) computed by the previous stage network(s).

Gradient Scheme Approximation: Piecewise approximation which uses series of linear gradient to approximate Hyperbolic tangent response.

Polynomial Scheme Approximation: Piecewise approximation which uses series of polynomial equations to approximate Hyperbolic tangent response.

Fully Parallel Design: A digital neural network design which duplicates fully the explicit computational nature of the FFMLP neural network within a digital representation.

Hybrid Design: A digital neural network design that combines some elements of concurrency while 'folding' the design to allow hardware re-use such that sequences of data are passed through major subcomponents. The design is used to equip implementation of optimised neural network architecture.

Cascaded Design: A digital neural network design incorporating a hybrid-based design, used for implementing a cascaded neural network.

CHAPTER 1

Introduction

1.1 Overview and Achievements of the Research

This thesis describes a research project examining novel versatile distributive tactile sensing systems. The research described focuses on the design of an efficient automated data interpretation system that is able to output its interpretation directly to describe the nature of contact with the tactile surface. The work integrates an advanced system on a chip technology (SOCT), an artificial neural network (ANN) and distributive tactile sensing technology to produce working systems. The design of two example systems is described. The first is a one-dimensional sensor for a steerable endoscope retrieving shape and 'touch' information. The second is a two-dimensional surface which interprets the three-dimensional motion of a contacting moving load. These cases demonstrate that the outcome of this research can be used to retrieve information from distributive surface of a different configuration, can interpret dynamic and static disturbances, and that this technology has the potential to discriminate contact and palpation in minimal invasive surgery (MIS) tools, and posture and balance in patients.

The distributive tactile sensing technique used in this research relies on the coupling of sensory data through the deformation response of a contacting surface. It has distinct advantages over other types of tactile sensing, particularly in terms of the way the sensor and the sensory processing topology leads to a minimum of required sensing elements. The sensory processing relies on an interpretation tool such as an artificial neural network to interpret the measured pattern into contact information. The whole system has similarities with the biological tactile systems, in which the sensory element detecting the deformation would be receptors, and the interpretation algorithm would be the perception function of the brain (refer to Figure 1.1). It is the interpretation function that derives the output information from the contacting disturbances through previously specified descriptors. It thus relates simultaneous coupled sensors to form a description of an event. As with the biological brain, this function needs to be taught by means of a learning process. The better the interpretation tool is trained, the more accurately the system can infer the different contact conditions.

Figure 1.1: Analogy between biological and artificial tactile sensing.

Conventionally, the interpretation algorithms of neural network are run as software intelligent properties using a PC workstation processor. But this approach has only a tenuous relationship with real-time applications, as the algorithms use highly parallel

computations. In this research the algorithm has been exploited and implemented into digital hardware using the field programmable gate array (FPGA) chip type Xilinx Virtex-II 2XCV3000 of an embedded system. The embedded system is a development board from Nallatech specifically created for fast prototyping of a design to produce digital functions for specific purposes.

The embedded system employed is a one step System on a Chip technology, a technology which integrates both software and hardware Intellectual Property (IP). This system uses more than one design methodology on a single integrated chip (IC) for the purpose of defining the functionality and behaviour of the proposed system (Nekoogar & Nekoogar, 2003). The development board consists of an analogue to digital converter (ADC), a digital to analogue converter (DAC), a clock FPGA for clock management and the main FPGA to which the digital interpretation algorithm design is programmed. The implementation process still follows the routine design flow of an FPGA, but with the emergence of new design tools such as the Xilinx digital signal processing toolbox and Matlab Fixed Point Blockset (Xilinx System Generator, 2006), modelling of the topology can be done using high level design as an enhancement to the traditional way. This requires programming using a machine language, such as the very high speed integrated circuit hardware description language (VHDL), which lacks clarity during simulation. The Xilinx System Generator (XSG) tool is used to rapidly generate the high level design to the level which will be used for synthesis and implementation.

The idea of *going hardwired* is useful in many ways in this research. Compactness, high-speed operation, low-power consumption, low cost, robust operation (Barranco, Andreou, Indiveri & Shibata, 2003) and extensibility of function are all factors which have also motivated this research to explore the implementation of the devised system, facilitating the needs of tactile sensing in MIS and balance and posture measurement of patients. The implementation of the hardwired distributive approach presented in this research is flexible in many ways, and has widespread applicability beyond the functions described in the current thesis. The ease with which the system can be programmed (and re-

programmed even after deployment), combined with its high speed operation makes the system suitable for exploitation in other applications involving hardwired distributive sensing.

1.2 The Rationale of using the Hardwired System for the Applications

At this point it is useful to highlight that the hardwired distributive tactile approach to sensing in this research was able to output descriptors to sense a description of contacting loads and disturbances as set of high level descriptors. The findings of the current research demonstrate that the system can be used successfully to discriminate, a) conditions of different profiles (including force distribution) of a subject generating contacting forces (case A of Figure 1.2), and b) behaviour and posture of a quasi-steady subject making fast and continuous contact disturbance with response to its three dimensional motion (case B of Figure 1.2), with satisfactory accuracy.

Establishing high measurement accuracy was not the prominent issue during this research, as the outputs evaluated represent enough discrimination for classification. In other words, regardless of the presence of a small relative error in each evaluated output used for discriminating a particular condition, for a biological system it is the contacting force transients derived from these collective outputs that provides usable measure of state or description. With this consideration in mind, and the advantage of having minimal sensory elements, the method used is clearly appropriate for the two biomedical devices mentioned earlier, namely flexible digit with tactile feedback sensing, and human balance disorder and posture.

In a biological working environment there is a significant contrast with measurement systems applied in manufacturing environments. In the case of the latter, the structural environment and the used of materials of consistent properties and behaviour would

imply that values measured between each workpiece have relevance and can approach high level of precision. In the biological environment, tissues and patients exhibit similar but different properties and behaviours such that single valued measurements have little relevance to the output required. The output usually relates to a description from which a prognosis can be derived. As a result, in this research, the emphasis has been on interpreting multisensor data to automatically formulate a descriptive parameter of meaning.

The operational efficiency and simplicity (in terms of design, which translates into reduced cost) of the system has produced several advantages, including the potential of the flexible digit to become disposable.

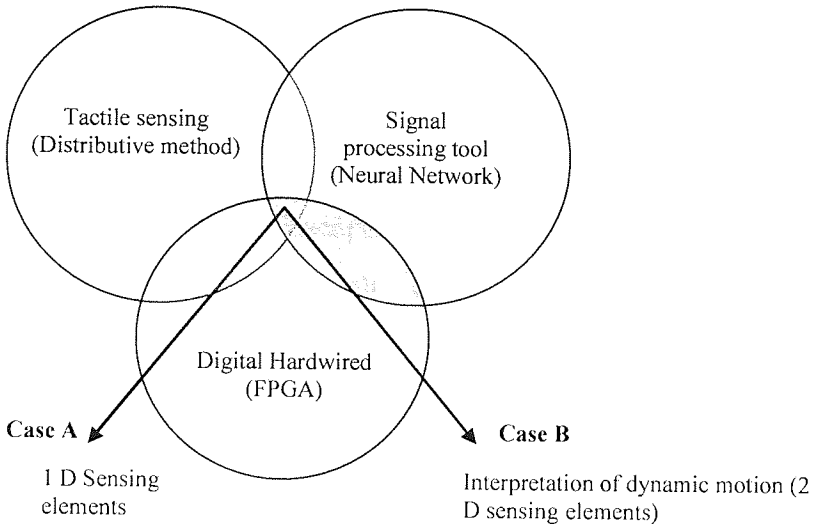


Figure 1.2: Diagram to illustrate the flow of the research.

1.2.1 Accuracies in Biological Working Environment

From the biological working environment, given the valuable notion of biological system and patient and the fact that the expression of condition is not necessarily pronounced, it is only possible to approach high accuracies in the discrimination of different categories. Therefore, given that states are not purely 'black or white', one can expect that automated

measurement systems will not approach 100% accurate. Generally, accuracies above 90% considered acceptable in clinical environment.

1.3 Statement of the Problems

1.3.1 The Endoscope in Minimal Invasive Surgery

Minimal Invasive Surgery (MIS), such as endoscopic and laparoscopic surgery, is increasing in frequency throughout the world. This is in part due to new technology, enabling surgeons to operate on previously inaccessible small scale structures. MIS has distinct advantages, such as reduction of trauma and lower risk of inflammation. In turn this means increasing the comfort level for the patient and (usually) reduced postoperative pain and fast recovery, reducing hospitalisation. However conventional endoscopes (see Figure 1.3) and laparoscopes have limitations. One issue is that feedback from contact at the operating site is poor. There is no ability to identify desirable information such as point of contact, discrimination between soft and hard tissue, and the detection of edges. This limitation has motivated the current research, aiming to introduce tactile sensing feedback into the endoscope system.

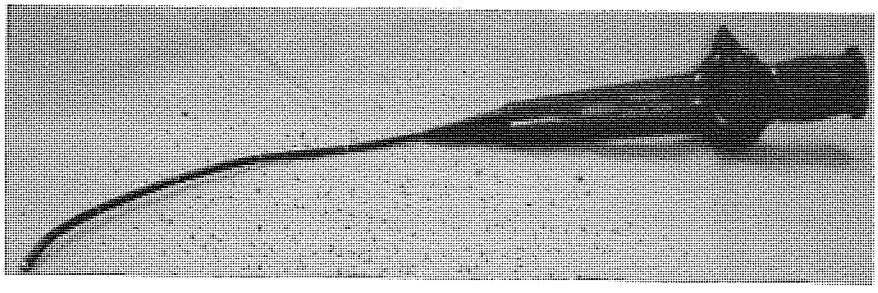


Figure 1.3: A typical steerable flexible endoscope.

1.3.2 The Need for a Tactile Endoscope

The concept of providing tactile feedback (such as touch and tool force information) for endoscopes and laparoscopes presents an ongoing challenge. In minimal access procedures a common process that would benefit from tactile information is the control of contact against the surface of the lumen whilst carrying out navigation and palpatory diagnosis. For example, during endoscope navigation a range of tactile feedback such as the sense of the point of contact of the digit, the discrimination between hard and soft contact, and the identification of edges (in conjunction with visual perception through the end of the endoscope), should allow the clinician to achieve greater control and better perception of position targets as compared with current methods. Similarly, diagnostic methods would benefit by adding in palpation (where a form of tactile perception would enable greater control of the action against the surface), to existing use of ultrasound imaging. In all such cases the key requirement relevant to the current research is the need to use the outer structure of the digit or tool as a means of providing tactile sensory feedback.

1.3.3 Previous Research and Progress Designing a Tactile Endoscope

Although previous research has attempted to design a useful tactile endoscope, much of this effort has been devoted to the improvement of the tips of the endoscopic grasper – another class of endoscope (Dargahi, Parameswaran & Payandeh, 2000; Najarian, Dargahi & Zheng, 2006). Most importantly, these methods rely on array processing techniques as the means of deriving tactile sensing feedback. Only a few previous studies have examined the possibility of a flexible tactile endoscope, which would translate information from the contact between the structural body of the digit and the surface using a distributive sensing method. The past work closest to the current research,

which formed the basis for this thesis, is that undertaken by Ma, Brett, Wright and Griffiths (2004), Tam, Brett, Holding, and Griffiths (2004) and Tam (2005).

Earlier research by Tam, Brett, Holding, and Griffiths (2004) involved the design of a flexible digit constructed from sealed flexible bellowed PVC tubes (refer to Figure 1.4), to mimic the function of a flexible tactile endoscope. The sensor elements were strain gauges that were distributed and embedded along the cantilever sensing elements. A neural network algorithm was used as the sensory processing tool, and was conducted using PC-based workstations hosting Matlab and the Matlab real-Time Workshop, Matlab XPC Target software. The workstations were equipped with interface boards, data acquisition cards DAQ, sensor signal amplifiers and associated circuits. This system provided valuable trails information, particularly in the static evaluation of the sensor networks and algorithms, and the relatively low frequency control algorithms required for the activation of the flexible digit. However, this laboratory-based system was not intended for clinical tool development. The real-time performance of the system was limited and could not take advantage of the high fidelity tactile sensing system. One means of addressing limitations is the implementation of tactile sensing algorithms in an application-specific processing system. A range of implementation technologies can be evaluated. One of these is the use of a conventional microprocessor and microcontrollers executing embedded software. Another is the implementation of reconfigurable computing designs involving either (i) a field programmable gate array (FPGA) based application-specific logic, or (ii) a system-on-chip (SOCT) solution in which the FPGA implements both the application-specific hardware portion of the design and the software portion of the design (which executes on a processor embedded in the same FPGA). The current research utilised this last approach.

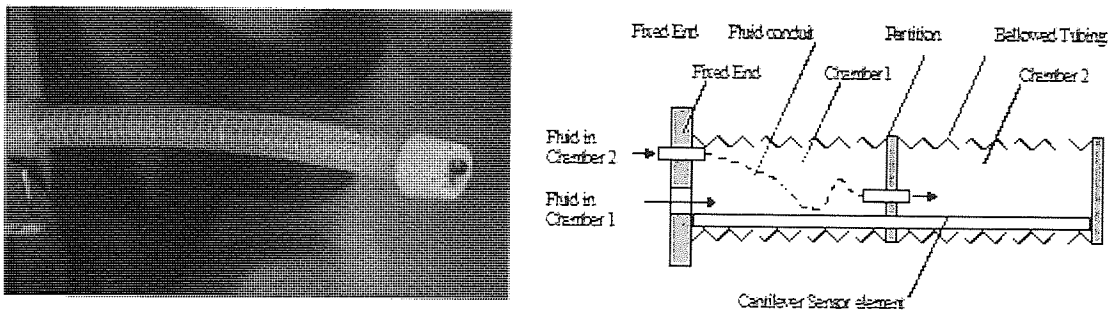


Figure 1.4: Flexible digit and details of the construction.

1.3.4 Future Benefits

The benefit that can be gained from the design and implementation of a successful tactile endoscope emerges through its ability to complement existing functionality of the conventional device. Such a system could be used to improve the power efficiency of the master-slave robotic system used in MIS (Tavakoli, Patel & Moallem, 2004), if it is to be incorporated into the larger system (see Figure 1.5). However, with the introduction of the hardwired tactile endoscope, even more prominent benefits could emerge. This high performance tool should be more robust in real-time operation than that the conventional method employed using a PC processor. In this way the new system could assist endoscopic surgery by using a more reliable skilled robotic-based approach (RBA). The term 'skilled' here refers to the ability of the RBA to sense the force on the tool, and thus compel the operation to be performed in closed loop manner. The RBA can assist a surgeon to conduct surgical operations automatically using the actuator, but the manoeuvrability of the tool should still be governed and controlled based on real-time tactile feedback information. This will facilitate haptic feedback. Another benefit from the new system is that the functions can be extended so that they perform all the signal processing parts of the operation, including the actuation controller and video streaming, in a single chip (refer to Figure 1.5). Compactness can be achieved and thus cost reduced as less space is required in the operation room. On one related example, Lawrence and Valentine (2006) have outlined a guideline for planning consideration for MIS operating

room. The average size of a traditional operating suite has been 11.22m^2. In MIS-compatible environments, a room needs to be 18.3 to 19.7m^2. The extra space is required to accommodate additional equipment such as touch screens or voice-activated systems for light, environment and sound control, and computer-assisted surgery equipment.

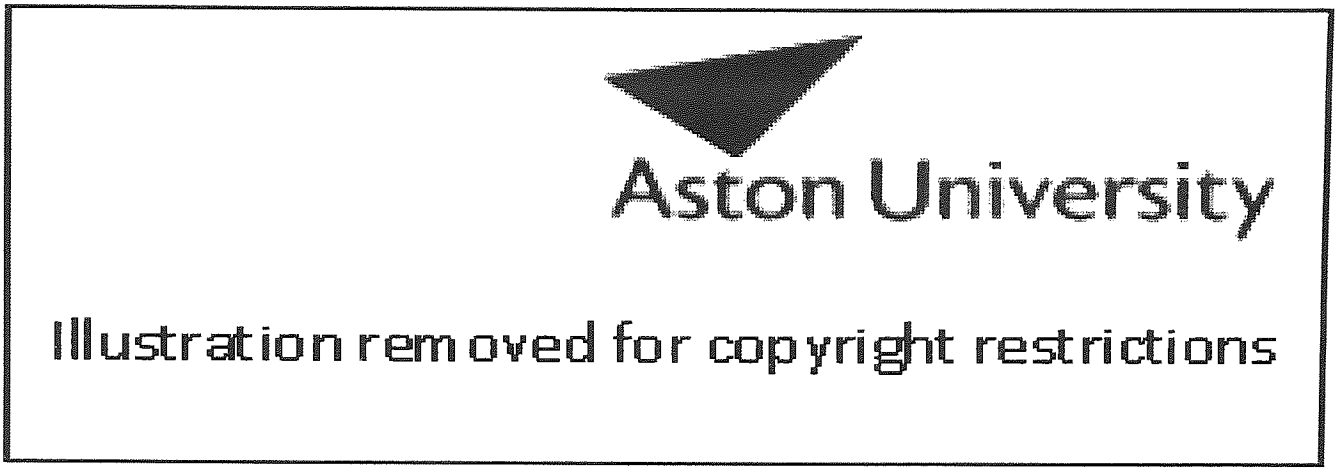


Figure 1 5: Master-slave system for an endoscopic surgery.

1.3 Dynamic Tracking using Hardwired Distributive Tactile Sensing

To demonstrate broader application of the hardwired approach, the embedded signal interpretation system was integrated into a different configuration of distributive sensor. Rather than a one-dimensional version of a cantilever beam structure, a two-dimensional plate was used as the tactile elements. The objective was to interpret the displacement motion exhibited by a standing human subject. This demonstrated that by using the same embedded subsystem, the tactile system could gather and output the different, relevant descriptions tailored to the contrasting application. This also showed that viability and versatility of the hardwired approach to the system.

One prospective application of this system in medical research is for examining patient balance and posture. Assessment of balance and posture is important in health settings as it provides clinicians with the means to perform early stage diagnosis for people, and to monitor response to therapy. Balance disorders are a serious medical issue suffered by millions of people (Hausdorff, Rios & Edelberg 2001) which can lead to injury or premature death. Individuals with chronic fatigue syndrome (CFS) and the elderly are among the populations with increased likelihood of suffering balance problems (Hausdorff, Rios & Edelberg, 2001; Fuller, 2000). Individuals' sense of balance also tends to decline with age, with an associated increased risk of falls.

Conventionally, dynamic posturography (DiFabio, Emasithi, & Paul, 1998; Johansson, Magnusson, Fransson & Karlberg, 2001) has been established as an important tool for understanding balance in a clinical setting. The dynamic posturography system is comprised of a force platform that evaluates somatosensory and visual influences on posture and equilibrium. This function can be exploited by performing a sensory test organisation (SOT) (DiFabio & Faudriat, 1996; Chaudhry, et al., 2005), which provides information about the individual's ability to integrate the visual, proprioceptive, and vestibular components of balance. During an examination the plate is normally used for measuring the horizontal shear information and to qualify the centre of gravity sway or postural stability under each of the SOT conditions. These are used to reflect the overall coordination of these components to maintain standing posture. DiFabio and Faudriat (1996) used the force platform (EquiTest, Neurocom Inc., Clackamas, OR) to obtain pediatric strategy scores (PED-SS) and strategy scores (SS) for assessing balance in children. The scores are obtained from the peak-to-peak amplitude of the horizontal shear force given out by a transducer under the centre of the platform. The patient stands on the platform facing a visual surround. The platform and visual surround moves simultaneously with the patient's body sway. Chaudhry, Findley, Quigley, Bukiet, Ji, Sims, and Maney (2004) used similar platform as DiFabio and Faudriat (1996), to measure a postural stability index score (PSI). The measurement is based on the maximum anterior-posterior (A-P) sway angles during SOT trials. Maeda, Nakamura,

Otomo, Higuchi and Motohashi (1998) and Matsuo, Narita, Senda, Hasebe and Ohtsuki (2006) both used a GS-10 Anima gravicorder for assessments and Nadapalan, Smith, Jones and Lesser (1995) used the Sway Weigh machine (Raymar).

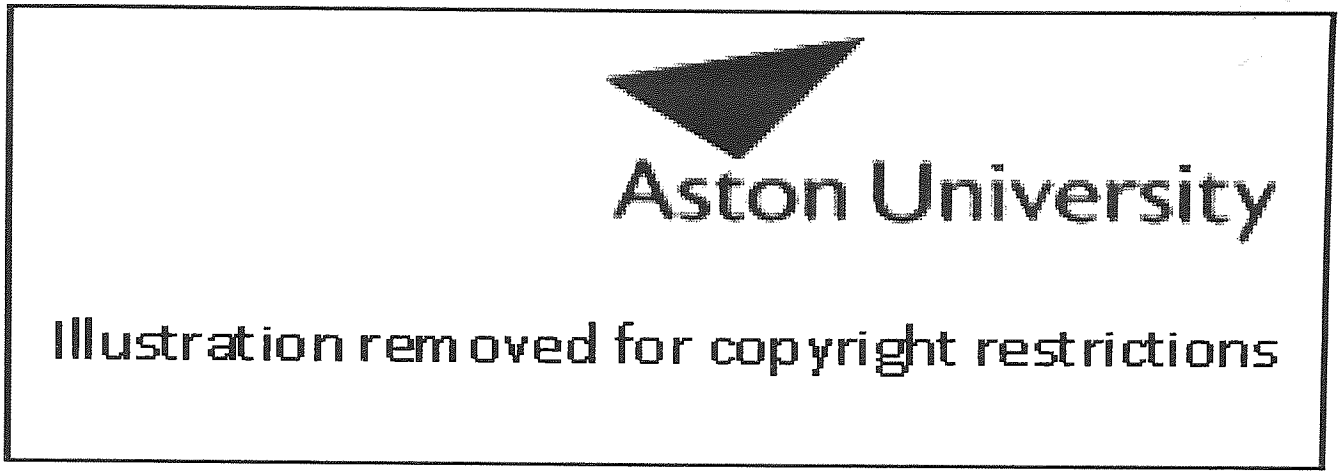


Figure 1.6: Dynamic tracking schematic diagram.

A tactile sensing system would be useful in the creation of new type of platform, such that shown in Figure 1.6. This would be similar to the force platform used for dynamic posturography, but designed to provide more useful output information with automatic interpretation, such as a direct determination as to whether a patient requires a balance intervention or whether an intervention has been effective. Reducing the number of sensor elements is also one of the advantages of such a design. The conventional force platform has five transducers – four used to measure the vertical force and one directly under the centre of the platform used to measure horizontal shear force.

1.4 Direction of the Research

The present research was largely driven by the need for a tactile endoscope, which in turn generated interest in the devising of a hardwired distributive sensing method. Previous work by Ma, Brett, Wright and Griffiths (2004), Tam, Brett, Holding, and Griffiths (2004)

and Tam (2005) have formed basis for this research, and these designs were extended b focusing on the exploitation of the signal processing system. The idea of a flexible dig was then adopted as the primary investigation. Similar to Ma, Brett, Wright and Griffitl (2004) the investigation of the structure of the digit and type of sensing elements that ca be utilised to derive tactile sensory feedback focused on modelling the cantilever sensin element of the flexible digit using a cantilever beam. Note that probably the most cruci component of this research has been the development of a specific purpose embedde signal processing function.

1.5 Principal Aims

The principal aims of this research are as follows.

1. To devise an efficient method of implementing a hardwired interpretation topolog for distributive tactile systems such as an FPGA chip.
2. To assess different arrangements of neural network architectures for the purpose c deriving the best solution to the interpretation tool for distributive sensing.
3. To demonstrate the feasibility of introducing a distributive tactile sensing system in tactile endoscope as a flexible digit, using an FPGA chip.
4. To perform real time measurements using the tactile system reflecting the tactil flexible digit. To then validate this process using outputs obtained by simulation.
5. To demonstrate the versatility of the method in higher dynamic applications such a the discrimination of three-dimensional motion from a dynamic contacting surfac This is representative of measurement of balance and posture ability of patients.

In view of these aims, the research has demonstrated the versatility of the FPGA chi interpretation system through the successful development and testing of two application of the distributive tactile sensing system. The studies also illustrated that a cascaded ANI architecture could achieve overall accuracy of better than 94%, when interpreting loadin

parameters such as load magnitude, load width and shape of an object positioned al the operating range of a one-dimensional sensing system such as for the tip of a steera endoscope. In contrast, single ANN architectures achieved overall accuracy of better t 85%. In a contrasting application involving a two-dimensional tactile system, all of ANN architectures were implemented; Cascaded, Multiple and Single. All have show be adequate when interpreting three-dimensional Cartesian components of motion c contacting distributing load with an overall accuracy greater than 93%. In both of applications the frequency between of complete sample cycles of the sensors readi was configured to be 15.625KHz. If the switching frequency is too fast exceeding optimum switching frequency of the multiplexor, the output magnitude can be impai However if too low will lead to the switching frequency becoming sluggish. Typ problem arise will be undersampling of data if dynamic measurement is considered. frequency use in this research was found to be sufficient in the applications studied. this rate the sample period is much less than the time constants associated v mechanical disturbances, and there is potential for interim system computatic functions such as data integrity checks and use of an error reduction method.

1.6 Contribution

The major contributions of this research are:

1. A new type of signal interpreter implemented in an FPGA was introduced and use assist the function of a mechanical sensing system. The integrated functions are capa of robust real time measurement.

2. An efficient approach of implementing the mathematical topology of the sig interpreter such as the neural network was introduced. The technique was used optimise the usage of the FPGA digital resources, such as the gate count and the dig multiplier. This issue is particularly important if other functions such as the chair interpretation of information and the control system are to be embedded into the sa chip.

3. Fast chip programming method were employed in this research in order to re programming time. Traditionally programming involves the use of VHDL mac language which is more complex.
4. To facilitate efficient distributive tactile sensing methods with the efficient hardw signal processing to generate a more powerful system that is capable of sensing dyn cases.
5. Because of the user-friendly techniques and tools introduced, the current rese acts as a means to bridge multi-disciplinary areas of research.
6. The research has opened a new way of research investigation on the use of hardw tactile sensing techniques specifically in medical applications.

1.7 Thesis Layout

This thesis is comprised of seven chapters, covering a background of the literatur description of the studies undertaken, presentation of the results and a conclusion.

Chapter 2 reviews different types of tactile sensing systems, analysing their advant and disadvantages, and discussing their application in the current project. This cha provides justification of the type of tactile sensing to be used throughout the research, focuses on the implementation of the sensory processing algorithm. Different way implementing the interpretation algorithm into the hardware, and other implementa issues are also reviewed. The second part of this chapter reviews the endoscope and need for tactile sensing feedback in this tool. A brief history of early endoscope de and use is presented, concluding with a discussion of the conventional endosc currently used in medical procedures. The functionality and limitation of the devic explained based on substantial references from previous theory and research.

Chapter 3 describes the experimental structure constructed to demonstrate the feasibi of introducing distributive tactile sensing system into a flexible digit. Different type

neural networks were investigated and tested. The performance and accuracy of (network is compared and measured. The proposed neural networks are used for implementation process into an FPGA as discussed in Chapter 5. Chapter 3 also pres the algorithms used for interpreting the loading condition as a continuous output. selected algorithm was then adopted for implementation.

Chapter 4 is dedicated to the study of the activation function of the neural networks the design of the solution to hardware implementation. This chapter initially discusses rationale for the selection of the type of implementation hardware used in the projec then goes on to detail the various techniques used to approximate the actual activa function. They are discussed, compared and the implementation described.

Chapter 5 is an implementation chapter. The various types of neural networks propo in Chapter 3 are implemented. The significance of the various techniques, in terms their ability to approximate the activation proposed in Chapter 4, are also discusse terms of their implementation into the neural network. All of the results are validated simulations. The findings of the real time visualisation are also presented in this chapt

Chapter 6 discusses the adoption of the hardwired distributive sensing technology interpretation of motion (surge, sway and heave displacement) from a dynamic swing this chapter the mathematical model of the surface platform and the swing are deriv The different types of neural network architectures are simulated and implemented. results are compared and discussed.

Chapter 7 presents the conclusions generated by the current work and discusses possi future directions for this research.

CHAPTER 2

Background of Studies

2.1 Introduction

This chapter reviews the theoretical and empirical research that formed the bas the current project. It provides an overview of tactile sensing systems, and exa methods of implementing such sensing capacities into the proposed designs as embedded software or hardware. This chapter also reviews the challenges of working the conventional endoscopy tools that justifies the rationale of the project, namel application of tactile sensing in the device. This chapter is composed of three sections

1. Section 2.2 reviews various forms of tactile sensing, including related previous relevant to the current research. This section also provides an introduction t concept of distributive technique for tactile sensing and discusses justifications f use.
2. Section 2.3 focuses on the signal processing aspects of distributive tactile sensin reviews different possible methods of implementing the system, including software and hardware solutions.

3. Section 2.4 explores the chosen application for this system (as described in Chapter 1), the endoscope. This section begins by providing a brief history of the endoscope, then discusses the function of the tool, and finally describes current research which is aiming to improve its functionality. These sections provide a basis for the proposed rationale of this research, namely to provide a tactile sensing element for an endoscope.

2.2 Artificial Tactile Sensing

Artificial tactile sensing can be defined as a method of measuring a spatial pressure distribution when a force is exerted perpendicular to a predetermined sensory area of a sensing medium. This basic definition is consistent with Harmon (1982), who first described tactile sensing as a continuous variable sensing of force (or simply touch) in an array. However, with the emergence of profound signal processing tools such as artificial intelligent algorithms, interpretation of a range of other contact properties (thus enhancing the information derived from contact) also becomes feasible. In other words, using an advanced signal processing tool for sensing data can then enable interpretation of that data as high level contact information (e.g., the profile of the contacting object). Nicholls and Lee (1989) highlight a range of contact parameters, such as size, shape, position, roughness, stiffness, and thermal properties, which may be inferred by such a system. Thus the detail and range of sensory processing which can be achieved was crucial to this research.

2.2.1 Types of Artificial Tactile Sensing

There are many different types of artificial tactile sensing systems (Tongpadungrod, 2002), but the most relevant to this research are the array and distributive sensing approaches. In the current context array tactile sensing can be defined as a method of sensing which uses a finite arrangement of sensing elements (usually in a complex array),

to infer contact information. This system uses sensory processing in which the topology relies explicitly on the series information of the isolated sensors. The sensory processing usually involves with reconstructing of data first before interpretation. In contrast, distributive tactile sensing can be defined as a method of sensing which relies on coupling effect of sensing elements to produce patterns of information through the deformation response of a contacting surface. This pattern of information can be directly used by the sensory processing for interpretation of state data. Thus there is advantage of discarding the data reconstruction process as applied in the array. In either of the cases the application can be in the form of a single- (linear) or a multiple- (usually two) dimensional axis sensing system. Usually the sensing elements used are integrated with the contacting surface or the substance. The operational speed of the sensing depends on the complexity of the sensor, which inherently affects the performance of signal conditioning, and the performance of the signal processor.

This section will analyse the two different types of sensing techniques and their sensory processing methods. In the context of this literature, understanding the topology of the different methods is important, as this will help identify and categorise the different types of tactile sensing methods which have been implemented.

2.2.1.1 Array Tactile Sensing

The array sensing approach is currently the most common means of constructing tactile sensing devices. In a two-dimensional surface application, illustrated in Figure 2.1, the arrangement of the sensors usually involves positioning them in arrays of A x B. In some constructions the type of sensor used (e.g., a pressure sensor), requires the sensory elements to be in direct contact with the load, but in many cases they are integrated with the contacting surface. The contacting surface is usually in the form of a substrate in which the main purpose is to protect the sensing elements from wearing out. However, regardless of the different scenarios of sensor setting, the concept of signal processing is

the same. In detail, the processing of sensor data is based on the information from isolated sensing elements in the array. Each of the sensing elements is taken into account and used for the reconstruction, thus allowing the sophisticated signal processing to reconstruct 2D and 3D images or other high level contact properties. The resolution of the array technique depends on the number of sensing elements, as they respond to contact independently. During contact only a few sensing elements at and around contact points are stimulated by contact, yet for the processing all of the data from the complete set of sensing elements has to be consistently delivered into the sensory processing unit. In most cases, due to the large number of sensing elements, a powerful multiplexor with high number of channels is employed to acquire the data. A common cost of high multiplexing is longer time required to acquire the complete sample of data signals, which hence affects the operation time of the sensory processing. Evidently faster computer processing is required to incorporate the processing of sensory data, especially when dynamic measurement is involved.

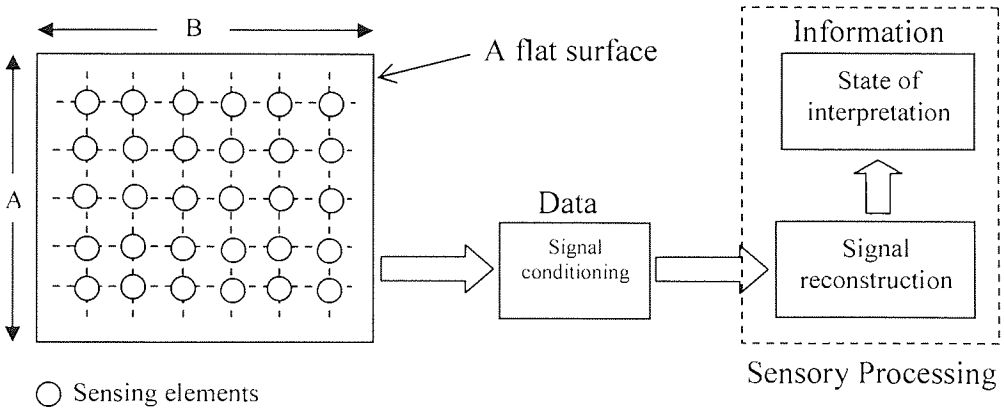


Figure 2.1: The schematic diagram to show array tactile technique.

In practice sensory processing of an array usually relies on reconstructing of signal from discrete measurement of the sensing elements to provide information of contact in a map like form. In the process, it usually uses an inverse of a closed form solution. In most cases to aid the reconstruction, an interpolation algorithm can be used (Fearing, 1990) to

recover a continuous response. Alternatives to the discrete signal reconstruction solution, such methods based on the Gaussian distribution (Stone, 1997; Stone & Brett, 1997), and the Fourier series (Provancher, 2003) can also be used. However, usually such methods are computationally extensive and result in low bandwidth operation. In some cases the information from the reconstructed pattern can then be utilised for pattern recognition and classifiers, as demonstrated by Tseng, Liou and Lee (1999) and Kim, Engel, Liu and Jones (2005). Alternatively, fuzzy logic can also be used for discrimination purposes (Li & Shida, 1997).

Much of the previous research in this field has used tactile array techniques, often addressing the static evaluation of the profile of a contacting object. Kolesar, Reston, Ford and Fitch (1992) use a monolithic silicon-integrated circuit coupled to piezoelectric polyvinylidene fluoride (PVDF) film. The integrated circuit incorporates 25 sensor electrodes arranged symmetrically in a 5 by 5 matrix. The signal conditioning and processing system is comprised of an electrical multiplexing circuit, and direct coupling of the PVDF material with a matrix of identical high-input impedance MOSFET amplifier, in order to measure the pseudosteady-state response of the sensor for an orthogonally applied load. With the aid of computer software a three-dimensional plot was implemented. In this example, the sensing elements are exposed to direct contact with the measured object.

In another prominent paper in this field, Harrison and Hillard (2000) present an application oriented system which uses pressure sensors incorporated into a plantar sensor. This is used as a platform device to measure responses from the 32 by 64 matrix of sensors. The signal processing is model based in the form of a discrete forms equation. The algorithm was then implemented in computer software such as Matlab for image reconstruction. The sensors were used to detect foot shapes. Benhadj and Dawson (1995), although having the same objective, use a different approach by designing a non-contacting tactile sensor for a pneumatic proximity-to-tactile sensing device. Essentially the system uses an array of up to a 32 by 32 matrix of pressure sensors to provide a

continuous variable output of two-dimensional sensing data. The sensing plane of the device incorporates a corresponding line array of air jets, which develop air cushion when striking a target of interest and produce back pressure as the basis for target detection and recognition. The rationale of this process is to prevent the sensor surface from making direct contact with the measured object. The advantage of such a system is that it would eliminate problems such as wear and damage that may be caused to the sensing plane. The signal processing uses computer-based processing which constructs the two-dimensional sensor information into a two-dimensional image.

Other research has been directed towards robotic applications. Fearing (1990) uses an array of 8 by 8 capacitive sensors for the construction of a cylindrical tactile sensor to detect force from touch. Here a closed form solution was used to reconstruct the sensing information. An interpolation technique aided the recovery of the location and magnitude of the centre of pressure from discrete measurements. The key feature of this sensor is that it is encapsulated in a cylindrical finger with a hemispherical tip. Work by Howe (1994) involved designing an array sensor with robot fingers for control and manipulation purposes. The most common use of such applications is for detecting changes in texture and properties of the contacting object. Hellard and Russell (2002) have been working on the design of a robotic gripper. The sensor is fabricated from urethane foam and an array of surface mounted IR emitters and detectors. The foam layer forms an optical cavity above the emitter/detector array of 7 by 7 elements of sensors. The optical devices are mounted on a circuit board below the layer of polymer foam. Compression applied to the surface of the cavity causes a change in its optical properties. The signal processing is based on the sensor model and was implemented in a microcontroller. For visualisation the processed data was plotted using a host computer.

Note that in many applications involving array techniques, the resolution of the measurement usually relies heavily on the number of sensor elements in the array, as they respond to contact independently. The recent developments in micro- and

nanotechnology fabricating miniaturised sensing elements (such as that of Engel, Chen and Liu (2003), may lead to increased resolution of array sensors.

2.2.1.2 Distributive Tactile Sensing

One common issue when implementing the array tactile sensing approach is the high numbers of sensing elements required to obtain satisfactory resolution. Although it is always possible to increase the number of sensors, a common drawback is the large number of wires needed for reading the data (Nilsson, 2000), and high signal computational effort in order to obtain satisfactory solutions. Other disadvantages of this technique are the high cost of fabrication and maintenance, and high power consumption. There is always a trade off between complexity, resolution and the cost of signal processing in the array. An alternative means of realising tactile sensing is by implementing a technique called distributive tactile sensing. This concept was first introduced by Stone and Brett (1995), and was then demonstrated by Stone and Brett (1996), Brett and Stone (1997) and Stone (1997) in the context of research on innovative medical devices. This method is an important tool for sensing for the research currently being undertaken by the Clinical Biomedical Engineering research group at Aston University.

Distributive tactile sensing emphasises the integration of a minimal number of sensing elements into a deforming contact structure. Crucially, the structure provides a non-linear coupling medium between sensors. The arrangement of the sensing elements can be arrayed (but usually in a small array), or distributed (positioned) over the prospective active region of the contacting surface prior to loading tests. However, with the introduction of an optimisation technique based on genetic algorithms (Tongpadungrod, 2002) the process will systematically search for the optimal position for the sensing elements for robust performance. In contrast to the typical arrangement of an array sensor,

the positioning of the sensing elements in distributive sensing array is as shown in Figure 2.2, such that the sensor positions look more arbitrary.

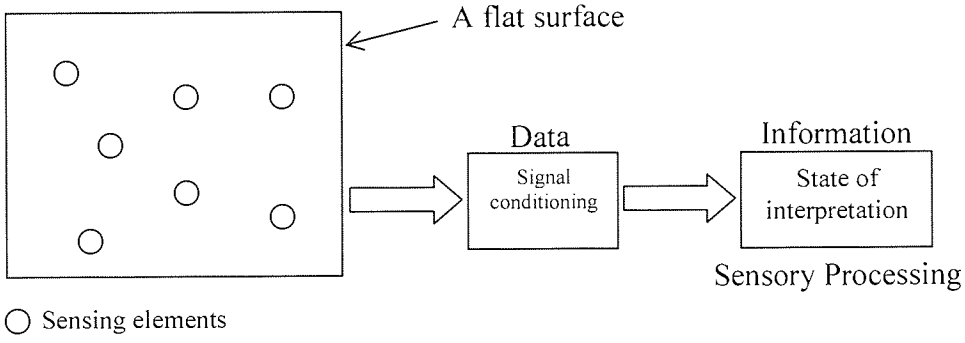


Figure 2.2: A schematic diagram depicting the distributive tactile technique.

The essence of the distributive approach is the presence of a surface or intermediate substance on which the object makes contact, and from which the sensor can detect the movement. This element is a necessity within the construction of this system. This scenario is analogous to human tactile perception, where the receptors in the dermis are enclosed by a layer of epidermis (the skin), see Figure 2.3.

Skin surface

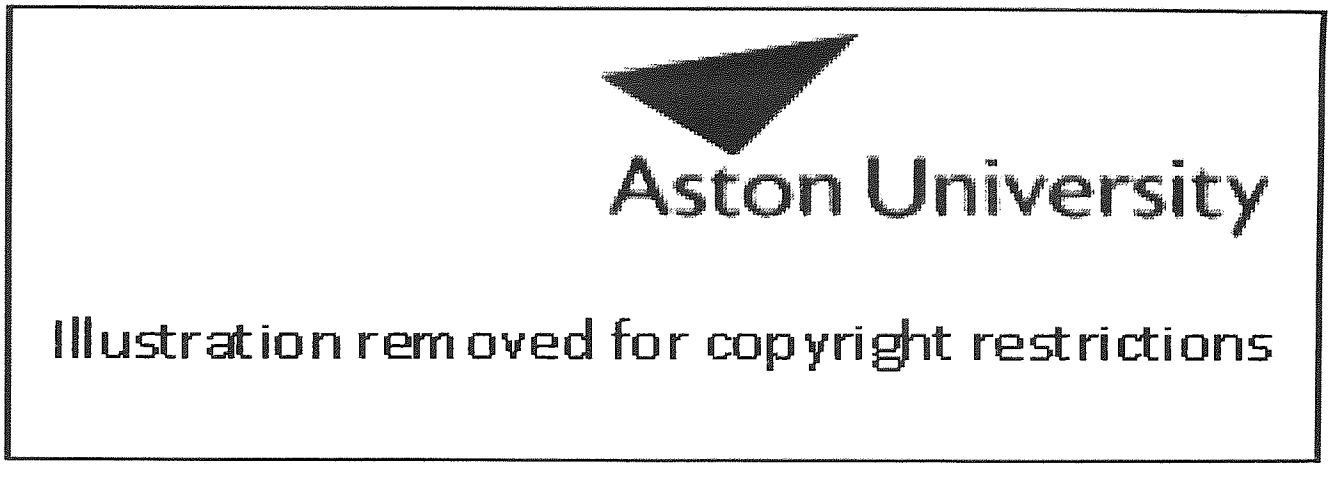


Figure 2.3: Human skin on the finger tip (figure taken from Johansson and Vallbo, (1983)).

In the distributive approach, the derivation of contact information relies on the relative information obtained from each of the individual sensor elements, in order to produce a 'signature' or 'pattern' of sensory data. The deformable surface provides a coupling effect between the sensors; thus even if a physical contact occurs in such a way that it falls in between the sensors, the sensors still can detect and produce a unique formation of sensory data. In contrast to the array technique, the resolution of the measurement does not primarily depend on the number of sensing elements, but rather on the interpretation algorithm (Brett & Li, 2000). Thus this system can have a higher spatial resolution than an array sensor of the same number of sensing elements. With this sensing method a minimal sensing element will be needed. No multiplexing of sensing data will be required for acquiring data, but even if required, employing a conventional type of multiplexor would be sufficient as only a small number of channels will emerge. Thus there is a high tendency for the data to be processed concurrently when using distributive sensing. Moreover, due to the lower complexity of the sensor, the signal processing of the distributive will not be as subtle as when using an array. With an appropriate signal processing interpreter system, a high performance tactile sensing method with high bandwidth operation can be achieved.

In the distributive approach, the signal processing takes place by using the pattern of sensor information directly, in order to interpret high level contact properties. No reconstruction of the sensing data is required. Pattern recognition tools (such as the artificial intelligent algorithm) offer the best solutions to the task of interpreting the sensing pattern into touch and contact information. With the prospect of this new distributive sensing system, applications to interpretative tasks will not only be limited to static functions, but also to dynamic disturbances.

Distributive sensing can be applied to either a one-dimensional or a two-dimensional context, as with the array system. Stone and Brett (1996) used distributive sensing for measuring gripping force distribution of a soft object for control and manipulation purposes. A thin aluminium alloy was used, as were strain gauge sensors arranged in

array, in order to measure the strain in the substrate. The essence of this work was to determine the contact location of the object.

In their 1997 paper Brett and Stone present the details of their investigation into a new method for measuring forces and tactile information. This work was conceptualised as a contribution to the sense of touch already used by surgeons, by implementing small number of sensing elements. The sensing system they describe is a one-dimensional system comprising of a beam in which one end is fixed and the other end is supported by a spring. Eight strain gauge sensors are used, arranged in single dimension array. The signal processing used is a Feed-Forward Multi Layered Perceptron (FFMLP) neural network. The method is compared to the closed form method and the Gaussian distribution, with the researchers concluding that the neural network offers the advantage of sufficiently high bandwidth, equivalent to the response of the human tactile sensing system, and with moderate accuracy. In related research, Ma and Brett (2002) conducted work on loading parameters, namely interpretation of load magnitude, load position and load width on a beam with both end simply supported. The sensors used were eight proximity sensors arranged in a single array. Signal processing and interpretation employed the neural network method similar to the previously described project, but here the emphasis was on enhancing the tool for the purpose of improving the accuracy of the measurement. For this aim the FFMLP neural network was arranged in cascade form.

In the more application oriented work, Ma, Brett, Wright and Griffiths (2004) adopted the above idea for use in a novel tactile endoscope, using distributive tactile sensing as a means of deriving contact information such as load magnitude, load position, load width and load shape. The design meets the criteria for invasive surgical environment, as ideally invasive devices should be of low complexity and low cost, with a view to producing disposable units. In the demonstration by these researchers, the system required only four strain gauge sensors, and their positions were optimised by employing the technique introduced by Tongpadungrod (2002). In more recent work, Cowie, Webb, Tam, Slack, and Brett (2006) has furthered the research by Ma, Brett, Wright and Griffiths (2004) by replacing the strain gauges sensors with Fibre Bragg Grating (FBG)

sensors. From the off-line measurements it was concluded that FBG based sensors were more appropriate than the strain gauges in this context. The only drawback yet crucial for implementing onto real time application is the incapability of the FBG approach to offer simple process for delivering on-line deflection measurement. A complicated signal processing mainly comprised of spectrum analyser device will be required to measure the changes of stress and strain. This signal output will be in frequency domain and will require a system to convert thus to adapt it onto the signal processing hardware. For this thesis, strain gauges are chosen as preferred sensing element due to their abundance, low capital cost and availability of amplification equipment.

The study of the effectiveness of interpretation algorithms made a leap forward through the work of Tongpadungrod (2002), who presented a technique using genetic algorithms to optimise the position of the sensing elements. Proximity sensors were used to detect the deflection of the substrate prior to the load placement along a one-dimensional sensing system. In the work, the researcher also used genetic algorithms as an interpretation tool. Compared to the FFMLP, the performance obtained by the interpretation based on genetic algorithm was poorer, as this method relies heavily on the agreement between the simulation of the surface behaviour and measurement.

Research on two-dimensional systems can be found in Stone (1997) and Brett and Li (2000). Stone (1997) designed a two-dimensional device in his thesis, which aimed to determine the position and magnitude of an object placed on the substrate. The surface in that system was made from a plastic material. The deformation induced by any contacting object was measured using five proximity sensors. The placement of the sensors was optimised based on analysing the response plot obtained from the surface model. All signal processing (FFMLP neural network) in the work described in Stone and Brett (1997) and Stone (1997) was run on 486 PC workstations. Similar research was performed by Brett and Li (2000), but with the objective of investigating the effectiveness of the method, by studying the insensitive area on the surface prior to an applied load. Five infrared-sensing elements were incorporated on plastic surface to

detect small deflections of the surface. The signal processing part (FFMLP neural network) was run by software on 586 PC workstations.

2.3 Hardwired Tactile Sensing

Most of the research in the literature on sensory processing incorporates sensory processing algorithms using PC-based workstations. Usually computer software was used in these studies to serially compute the interpretation algorithm. However, such algorithms are essentially running in parallel in terms of their dataflow, and thus they are well suited to parallel processing approaches. Although some work has been done on hardware implementation of such algorithms (e.g., Hellard & Russell, 2002), this approach has never been applied to neural network-based distributive tactile sensing, particularly into an FPGA. Research which has attempted to implement tactile sensory processing into an FPGA was by Nagy, Szolgay and Szolgay (2004). In their research the FPGA was used as a platform for implementing an emulated Cellular Neural Network Universal Machine (CNN-UM) – another class of artificial neural network used for solving the state equation of a micro electromechanical tactile sensor. However, this research only emphasised solving the transient of the differential equation using a realistic input parameter. Implementation of this system into a real case for measurement has never been achieved. This research did produce useful information, in that it demonstrates the reliability of the tool when handling complex (parallel) implementation, while maintaining a speed of operation superior to that of a PC. From the experiments undertaken it was deduced that using the FPGA is 12 times faster than using a Pentium IV 3GHz processor when obtaining the solution. This research realised the implementation of a hardwired neural network for sensory processing, but also maintained low processing complexity and higher bandwidth. Although there are various types of hardwired implementation of the neural network (Barranco, Andreou, Indiveri & Shibata, 2003), this research focuses on an implementation using the field programmable gate array chip (FPGA) as a platform for processing the sensory interpretation algorithm. The rationale for this selection will be presented in Chapter 4.

2.3.1 Research on the FPGA Implementation of Neural Networks

The need to implement neural networks as a means of hardwired sensory interpretation for tactile sensing is clear from the above review of the literature. However, to date few researchers have chosen to take this direction, although much work has been undertaken implementing neural networks in general, mainly focusing on, a) the application and b) investigating the optimal techniques of implementing. This work thus provided a great deal of experimental data and many techniques that were useful for the current research.

In recent years various techniques and architectures have been developed for the implementation of artificial neural network into FPGAs. Some designs implement the full algorithm directly, including the non-linear activation function, and use parallel processing to maintain speed at the cost of using extensive FPGA resources. One example of such research is work by Arroyo Leon, Ruiz Castro, and Leal Ascencio (1999), who successfully implemented an artificial neural network (FFMLP) on an FPGA chip type Altera's EPF10K20RC240-4, and accommodated parallel processing of three neurons and the non-linear activation function. Research by Galindo Hernendez, Leal Ascencio and Aguilera Galicia (1999) extended that of Arroyo Leon, Ruiz Castro, and Leal Ascencio (1999) by focusing on the application of the system and visualisation using computer software such as LabView. The system was used to estimate biomass and product concentration in the biochemical growth of a pigment. The complete design consumes 98% of the available gates of the device. However, the major disadvantage of such a design is the resource limitation of high logic density FPGAs. It can thus be understood why earlier implementations, such as that of Cox and Blanz (1992) used arrays of almost 30 FPGA Xilinx XC3000 chip to map simple multilayered perceptrons (FFMLP).

In another example of such architectures, time-division multiplexing has been introduced by two research groups, Ossoinig, Reisinger, Steger and Weiss (1996) and Izeboudjen, Farah, Titri, and Boumeridja (1999). Here a single shared multiplier per neuron is applied,

but at the expense of speed (particularly when evaluating the activation function). Using this method Izeboudjen, Farah, Titri, and Boumeridja (1999) have mapped an MLP neural network for the XOR problem with the network on a single Xilinx FPGA chip type XC4020. Ossoinig, Reisinger, Steger and Weiss (1996) have fabricated four Xilinx 4013 and a Xilinx 4005H FPGA chip on to a board. It follows that there is an ongoing need for a fast, accurate, and relatively resource efficient implementation suitable for an FPGA-based implementation. Consequently, the current research seeks to address this need by devising an architecture which is a hybrid between parallel and serial computation for a neural network. This research thus aims bring about a balance between the digital resources requirement and the expanse of operation timing.

In the current research addressed the used of MLP (also known as feed-forward multilayered perceptron neural network FFMLP). This selection was based on earlier work such as that carried out by Brett and Li (2000), Tongpadungrod (2002), Ma and Brett (2002), Brett, Wright and Griffiths (2004) and Tam (2005). The FFMLP neural network has been successfully implemented in several tactile sensing systems, tailored especially for output discrimination.

2.3.2 Implementation Issues

For all of these techniques, it is the nonlinear activation portion of the neural network which triggers the most difficulties during the implementation (Beiu, Peperstraete, Vandewalle & Lauwereins, 1994). This part has a strong influence on the accuracy and performance of the application, and may constitute the most computationally extensive aspect of the implementation. The conventional way to overcome the activation function computation problem is by using medium-granularity look up tables, such as the sigmoid look up table (Galindo Hernendez, Leal Ascencio and Aguilera Galicia, 1999; Marchesi, Orlandi, Piazza & Uncini, 1992; Ossoinig et al., 1996). However, such an approach leads to a high demand on memory by the FPGA devices to give satisfactory accuracy. Various techniques have been proposed, such as that of Tang and Kwan (1992), who use an

adaptive-slope sigmoidal activation function. Chen and Chang (1996) have introduced an adaptive generalised hyperbolic tangent function with two free parameters, the slope and the saturation level. Hikawa (2000; 2003) has also presented an activation function using a three valued function, improved by using a piecewise-linear function. However, all of these methods are unique and are designed for the specific purpose of the neural network implementation. Evidently, there is a need to devise a suitable digital activation function to suit the neural network architecture in this research.

2.4 The Endoscope

One of the focused studies of the work is the steerable flexible endoscope with tactile feedback information. This section reviewed the history and research of the endoscope.

2.4.1 History of the Endoscope

The first endoscope was invented by Philippe Bozzini in 1806 and was demonstrated in Vienna (Miller, 1986). This instrument was called a "Lichtleiter", which means light conductor. It was a simple silver tube which was illuminated by light generated by a candle. In 1853, Antoine Jean Desormeaux made an improvement on the design by replacing the light source with a turpentine and alcohol mix. The light being focused on the field of view remained no greater than the tube diameter, and he was the first surgeon to use the Lichtleiter of Philippe Bozzini on a patient.

From this point onwards there was continuous development of the endoscope, but a major improvement in the design was made after the electrical light was invented by Edison in 1891. When smaller bulbs became available this meant that internal light was available for the endoscope. Another major enhancement was made when Wolf produced the first

flexible scope in 1930, but this was based on a conventional lens system (Hopkins, 1976). In 1941 the design of the endoscope was improved again by adding manoeuvrability of the distal tip. However, it was not until 1970s that endoscopic and laparoscopic examination and surgery began. In 1984 the first laparoscopy cholecystectomy was performed. Four years afterwards the first operation involving the use of a video-laparoscopic cholecystectomy was performed (Lindberg, 2002). This was the beginning of the era of minimal invasive surgery.

2.4.2 Recent Achievements in Endoscopic Research

Work related to the development of the endoscope continues. The aim of most of this research is to obtain better feedback and improve perception. For example Yamauchi et al. (2002), has been attempting to improve the field of view (FOV). In this research the quality of visualisation was enhanced by introducing a dual-view with image shift capability. The technique uses a wide view method, which improves the FOV up to 120 degrees, and image shift controlled zoom view (for manipulating surgical tools within it), in total giving a variable zoom FOV from 40 degrees to 120 degrees. This approach also allows two views to be observed at one time. Other research by Smithwick, Seibel and Vagners (2001) have resulted in the introduction of a different class of endoscopes. The Single Fibre Scanning Endoscope (SFSE) technique is based on a combination of a resonating optical fibre and a single photo-detector. This approach produces a large field of view and high resolution images from a small flexible package.

A further challenge to the classical endoscopy operation is disorientation resulting from the limited field of vision, which can lead to surgical error and fatigue. This issue creates a tough barrier for the novice and may lead even an expert to inadvertently commit serious surgical errors. Some research has been undertaken seeking to prevent such complications and optimise training. One technique is to aid navigation is the use of a 3D model based system that shows a single perspective view of the patient and the

endoscope model (Yamashita, Yamauchi, Mochimaru, Fukui & Yokoyama, 1999). This virtual endoscope has a viewing cone with a simulated light to indicate the viewing direction and visual field in real time. The system's three clipping planes automatically follow the endoscope and help keep the surgeon aware of the actual position of the endoscope. Another improvement was the introduction of an intra body navigation system that can directly measure and visualise the three-dimensional position of the tip and the trace of an ultrasound endoscope (Tamura et al., 2002). The proposed system can identify the 3 Dimensional location and direction of the endoscope probe inserted into the body, in order to furnish endoscopic images.

Another area of interest to researchers is the development of a semi-autononomous endoscope (Menciassi, Park, Lee, Gorini, Dario & Park, 2002). This research involved the manufacture of a semi-autonomous robot which was specifically designed for colonoscopy. The project focused on two problems: the generation of an effective and reliable advancement of endoscopic tool (to going forward) use in the colon, and the possibility of steering the robot in order to overcome acute intestinal bends. A self-propelled device was introduced which generates "internal" forces and which does not require any external pushing actions. These would improve the colonoscopy procedure in terms of patient pain reduction, making advancement easier, and improving diagnosis as a whole.

2.4.3 Towards a Tactile Endoscope

The key aim of endoscopic surgical operations is that the surgeon should acquire as much information from the tool tip as possible, but with maximum safety, particularly during tissue manipulation. However, examination of the literature clearly demonstrates that, to date, the only information to guide the surgeon during the operation is in the form of visual feedback. One solution is to enhance the sensor information from the endoscope, to enable it to provide tactile feedback. Research in this direction has been undertaken by

Dargahi, Parameswaran, and Payandeh, (2000) for another class of endoscope, the endoscopic grasper. In Dargahi, Parameswaran and Payandeh (2000), the authors focused on enhancing the output information available, to provide not only the magnitude of the applied force, but also the position. The prototype sensors in this system consist of three layers. The top layer is made of micromachined silicon with a rigid tooth-like structure. The bottom layer is made of Plexiglass serving as a substrate. The middle layer is composed of Polyvinylidene Fluoride (PVDF). This work was extended by Najarian, Dargahi and Zheng (2006) to measure stiffness. Takashima, Yoshinaka, Okazaki and Ikeuchi (2005) used a different approach, designing a new tactile sensor system which measures tactile force by means of image processing. The system uses an infrared (IR) cut pattern installed in the tip of an endoscope to measure 3D information. However, the technique relies heavily on software to convert the image into a measure of the tactile force. Related research described in this chapter (and also discussed in later chapters) includes work by Brett and Stone (1997), Stone (1997), Ma, Brett, Wright and Griffiths (2004), Tam, Brett, Holding and Griffiths (2004) and Tam (2005).

2.5 Summary

This chapter reviewed two types of tactile sensing, namely the array and the distributive methods. Both techniques seek to measure and interpret contact properties on a predetermined sensory area of a sensing medium. However, there are problems with each due to the high complexity of the array, the high cost of realising the sensing system, and the inadequateness of both methods in real time application (particularly when implemented using a PC workstation). The current research sought to address these issues by exploiting the distributive tactile approach, and introducing the concept of *going hardwired* using an FPGA chip for the sensory processing unit. The adaptive hardwired information interpretation scheme should complement the distributive method, producing a novel distributive tactile sensing system. This system embraces speed of interpretation

with constructional simplicity, which is useful for another application (described later in the thesis) of tactile sensing involving dynamic interpretation.

The second part of the chapter reviewed an application for the novel distributive tactile sensing system. This section presented a literature review which clearly leads to the proposed research, and highlights the importance of creating a tactile endoscope. To address this aim previous research points to the use of the distributive type of sensing. This is mainly due to its less complicated construction, use of a smaller number of sensors, and the fact that it is simpler to fabricate. There are some designs available which have attempted to move in this direction, but they have never been tested on real time applications. The present research, which involved the development of the design, is described in the following chapters.

CHAPTER 3

Distributive Tactile Digit
(The foundation of the study)

3.1 Introduction

The research described in this chapter addressed several different objectives. The first aim was to demonstrate the principle of distributive tactile sensing. Secondly, the work described in this section investigated the performance and sensitivity of the sensing technique when applied to a fluid driven flexible digit, using the closest approximation cantilever rig as a case study. Finally, this chapter provides details of a preliminary study of the implementation of the embedded real time sensory signal processing and measurement system. The overarching purpose of this chapter is to describe the development of a sensing system able to interpret the state of the contact condition using as few sensors as possible.

In this research the first procedure was to define the optimal distributed positions for the sensing elements which are to be incorporated along the constructed cantilever rig. The method of configuration of the sensors locations obtained by Ma, Brett, Wright and Griffiths (2004) is adopted in here. This technique, which was first introduced by Tongpadungrod (2002), uses a genetic algorithm to derive the optimum position of the sensors along the beam to provide the highest discrimination of *pattern* or *signature* deduced from the types of loading conditions. The discrimination of the different loading conditions is achieved using an interpretation tool such as the neural network algorithm. Generally, upon training, the tool takes the pattern of signals from the distributed sensor elements and interprets them into information. The purpose of this procedure is to search and gather information as to the different loading conditions', different profiles of width and shape of different force, at different applied position along the total length of the beam. When using this procedure the aim is to incorporate the interpretation algorithm into a continuous sensing measurement system, to provide consistency to the distributive tactile sensing.

This research first demonstrates the use of a feed-forward multi-layered perceptron (FFMLP) neural network in order to evaluate different loading conditions along the operating length. The outcome is then compared for different classes of neural networks, radial basis function (RBF) neural networks. Influencing this judgment is the need to obtain a network model which is efficient in both the performance and computational cost. These criteria are important for embedding the signal processing system into an FPGA, and will be explained further in Chapter 5. The selected class was then used for the evaluation of a wider range of loading conditions and its performance was analysed. Finally, the class was then employed for development into a different arrangement of neural networks. The outcome of this process was then be used for developing the hardware sensing system.

3.2 The Flexible Digit: Overview

The fluid pressure driven flexible digit (discussed in Chapter 1, see Figure 1.4), was constructed by sealed flexible bellowed PVC tubes, having differential stiffness across the cross-section due to a cantilever structure attached along the length of the bellowed tubing. This structure acts both as a sensing element and forms the neutral axis at the surface of the tube. The application of pressure causes curvature of the tube at a constant radius. The prototype digit consisted of two chambers each actuated by a separate pressure supply and mounted in series. The bending moment exerted by the internal pressure is related to the area of the cross-section of the tube and its distance from the neutral axis, and therefore there is the allowance for a service conduit at the neutral axis.

3.3 The Test Rig: The Cantilever

An experimental rig was constructed using a cantilever beam to approximate the function of the cantilever sensing element of the digit. The rig consisted of a straight mild steel beam with a dimension of 228.6mm (9 inches) of length l_B, 12.7mm width and thickness of 0.4mm, with a fixed support at one end only (refer to Figure 3.1). This is similar to the flexible digit where the root and tip of the digit are analogous to the fixed and free end of the cantilever respectively. However, unlike the driveable flexible digit, the cantilever beam is only used for the measurement of *un-actuated* operations. That is, there is no other driving force acting on the beam to create deflection other than load applied for measurement. The investigation of tactile sensing with the presence of both external forces and actuation is beyond the scope of this research. However, the outcome of this investigation has direct relevance to the more general case.

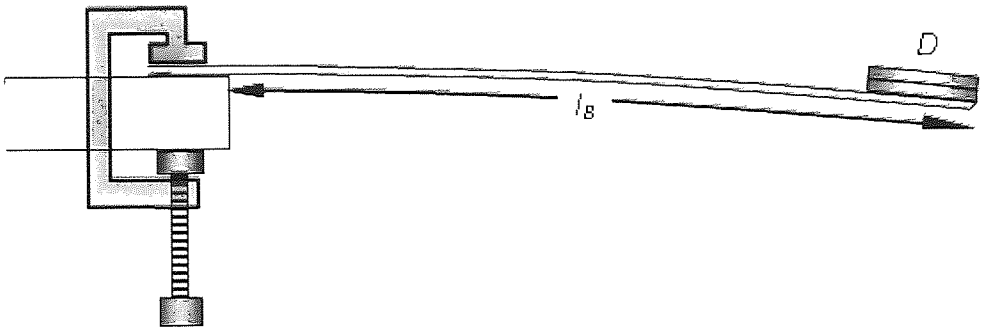


Figure 3.1: A cantilever model with length l_B and load D applied at the tip (diagram not to scale).

3.4 Simulation Model of Beam Deflection

The mathematical model of the cantilever beam is now described. The objective of the modelling is to study and predict the deflection behaviour by means of computer simulation. The model was first verified by comparing the simulation output with the real deflection of the beam. The model can thus be used to validate the sensor outputs.

3.4.1 Beam Theory

Gere (2002) has shown that the beam equation for the intermediate case (which has the most relevance to the current research) can be expressed by;

$$v = \begin{cases} -\dfrac{Da_g i_B^{\,2}}{6EI}(3a - i_B) & (0 \le i_B \le a) \\ -\dfrac{Da_g a^2}{6EI}(3i_B - a) & (a \le i_B \le I_B) \end{cases} \qquad (3.1)$$

Where v is the vertical deflection when the load is at position a along the incremental length i_B of the beam, and where D is the load weight, a_g is the acceleration due to

gravitational force, and EI is the flexural rigidity. E is the elastic modulus of the beam and I is the second moment of inertia.

For the case D applied at the tip of the beam (or $i_B = a$), (3.1) becomes;

$$v = -\frac{Da_g i_B^{\,3}}{3EI} \tag{3.2}$$

Differentiating v of (3.1) will yield the slope θ for the intermediate case;

$$\theta = v' = \begin{cases} -\dfrac{Da_g i_B}{2EI}(2a - i_B) & (0 \le i_B \le a) \\ -\dfrac{Da_g a^2}{2EI} & (a \le i_B \le I_B) \end{cases} \tag{3.3}$$

The second derivative of vertical deflection v, has the relation to the bending moment M_B, by the expression shown below;

$$\frac{\partial^2 v}{\partial i_B^{\,2}} = \frac{M_B}{EI} \tag{3.4}$$

Further differentiating (3.3) and incorporating (3.4), the bending moment can be obtained for the intermediate case;

$$M_B = v'' = \begin{cases} Da_g a\left(1 - \dfrac{i_B}{a}\right) & (0 \le i_B \le a) \\ 0 & (a \le i_B \le I_B) \end{cases} \tag{3.5}$$

To verify the mathematical models, deflections (v) obtained from measures taken during experiments were compared to the computed results of equation (3.1). In the experiment, loads of 10g, 13g, and 23g were applied at the tip of the beam and deflections were recorded at seven different positions along the length of the beam I_B. Comparative plots

in Figure 3.2 reveal that the deflection results of the plots are consistent. Thus the principal equation (3.1) is consistent with the current experiment.

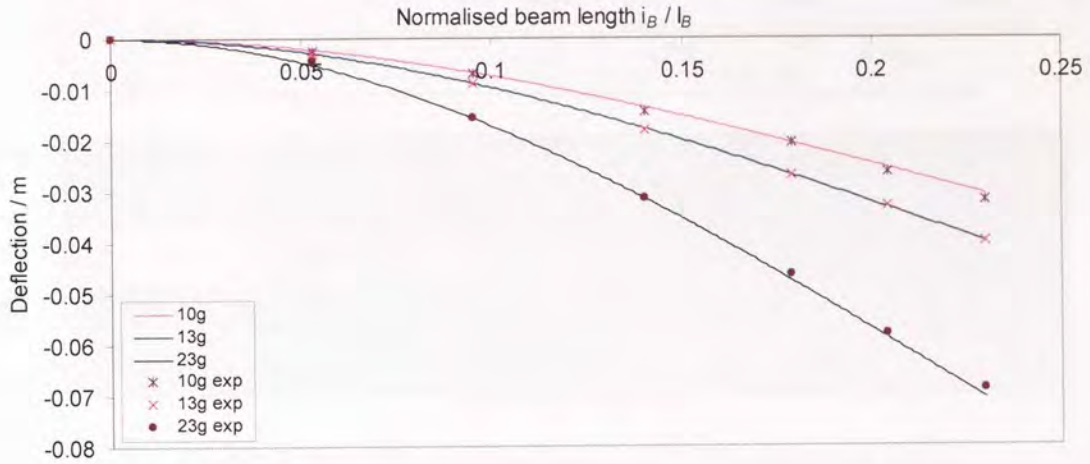


Figure 3.2: Comparison between deflections obtained by experiment (markers) and simulations (lines).

3.5 Distributive Sensor Setup

On the basis of the work carried out by Ma, Brett, Wright and Griffiths (2004), it was proposed that four sensors should be placed along the beam as shown in Figure 3.3, in order to provide the optimum peak of sensitivity for the sensing output. Various types of sensors can be used, but the most appropriate are those sensors which are resistive based sensors. This is because, in contrast to other sensors which use active potential difference as their method to measure signal, the resistive based sensors minimised the effect from electro-magnetic interference. The electro-magnetic interference can be a problem in vivo application as it interferes with the patient's body when the sensing system is exposed. By using resistive sensors this issue can be avoided.

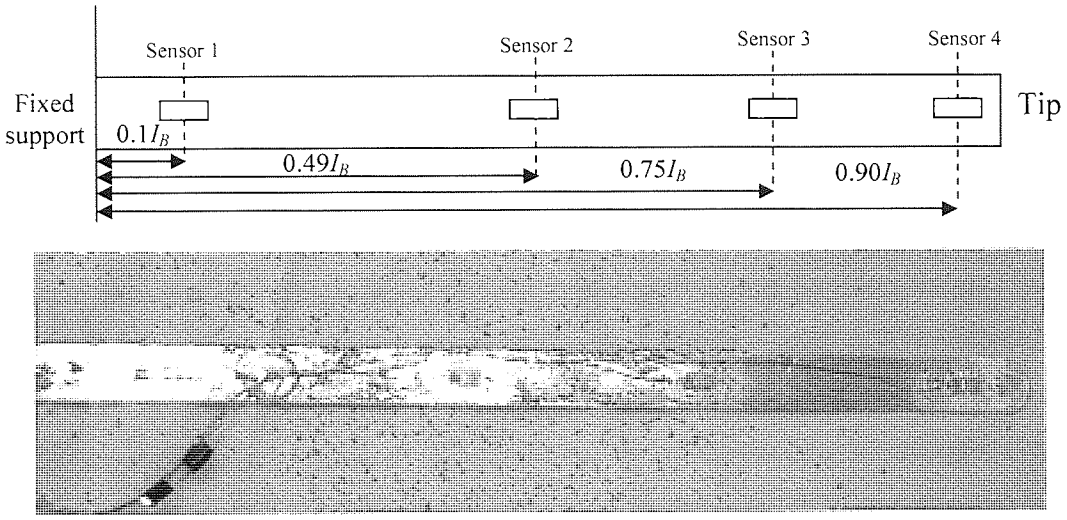


Figure 3.3: Beam with configuration of sensors locations (bottom view).

In this research the sensing elements used were strain gauge sensors (steel 5mm x 1.8mm strain gauge). These sensors were chosen for their abundance, low capital cost and availability of amplification equipment. At specific points during the measurements, they detect deflection changes by means of stress or strain changes. The stress or strain changes then alter the resistive property of the sensor's material proportionally (i.e., the resistance changes proportionally as well). As the maximum resistance change delivered from each of the 120 Ω strain gauges due from the deflection will not be higher than 0.1% under a typical maximum strain $\varepsilon = 10^{-3}$ (strain is dimensionless), the sensors were embedded back-to-back on the beam to provide a half bridge sensing configuration. There are several advantages of using such configuration. First, this configuration increases the sensitivity of the circuit, as the resistive resulting will be doubled as both gauges will be active. Secondly, error due to offset from un-steady temperature changes will be minimised by the balanced potential difference due to the temperature induced resistance changes (Fraden, 1996). The effect of temperature changes in strain measurement is always an issue in strain/stress measurement using strain gauge sensors. An unsteady temperature leads to significant error and variable measurement. This problem leads to constraints on the ability to perform the real time measurements using the system.

3.6 Sensory Data Output

The output of the bridge was in the range 0 - 10mV. To make the signals more useful, they need to be signal conditioned so that the output range is increased and the signal-to-noise ratios are improved. To do this, the voltage from every part of the bridge is amplified into the more significant voltage readings required for the process. For this procedure, a prototype circuit (as shown in Figure 3.4) comprising a Wheatstone bridge and differential amplifier were used. The amplifier is an instrumentation amplifier type AD624 (see Appendix 1 for technical specification). By introducing trimming resistors to the input resistance of the differential amplifier circuit, the gain per channel can be adjusted to give feasible ranges for the measurements. A low-pass filter with cut-off frequency of 25Hz was added to the output to remove any high frequency distortion.

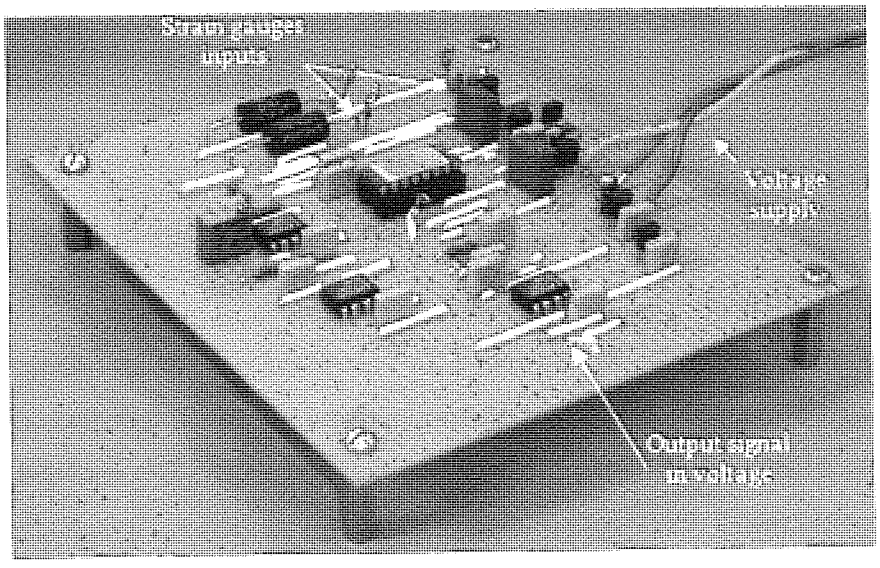


Figure 3.4: The prototype strain gauge amplifier with a low pass filter.

3.7 State of Interpretation Algorithms

A variety of transformation algorithms can be used to interpret the pattern of data output from the sensing elements in order to generate describing indices of load distribution. In this research the scheme involved the use of an artificial neural network. The technique was inspired from the bioelectrical network operation of the brain, as formed by neurons and their synapses (McCulloch and Pitts, 1943). This system has the ability to learn how to interpret data based on so-called learning stage. But the most important reason for using such a technique rather than most other methods is that there is no need to derive the inverse function of the closed form solution relating to the variables being measured (refer to Chapter 2). The output of the model is then less affected by the computational errors. A second advantage of this system is that it uses very sophisticated modelling techniques capable of modelling extremely complex functions. This is advantageous in complex situations where the contacting surface responses to an applied load cannot be accurately derived. Furthermore, since artificial neural networks have a parallel structure, real time measurement can be achieved.

3.7.1 Artificial Neural Networks (ANN): An overview

An artificial neural network is composed of a network of relatively simple processing elements (PE), such that the global behaviour of the network is determined by the layered connections between the processing elements and the parameters of those elements. A typical artificial neural network structure is depicted in Figure 3.5. There are two types of processes involved in the use of an artificial neural network. The first is the training process, also called the learning stage, where the network is trained and reconfigures itself in terms of the inputs for targeted output patterns. The second procedure is the testing process, where information from the inputs are fed into the network and processed to evaluate the approximated results. This activity is sometimes called the measuring stage.

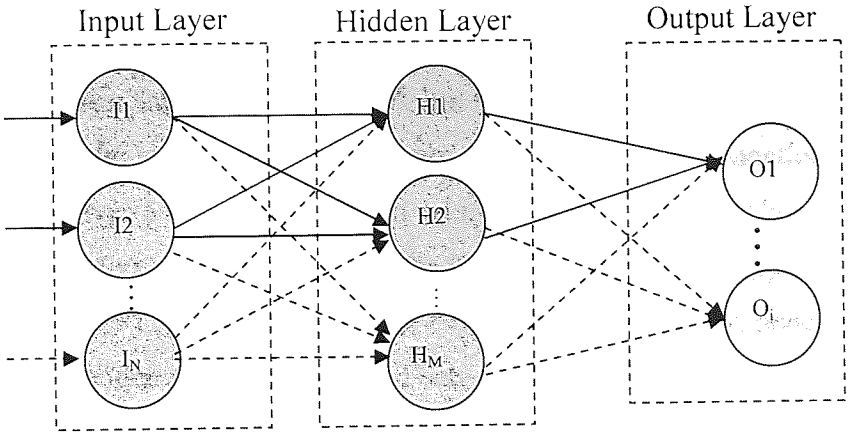


Figure 3.5: An artificial neural network processing structure.

3.7.2 Feed-Forward Multilayered Perceptron FFMLP

Although there are various types and classes of artificial neural networks available, the current research employed the most common form, namely the feed-forward multilayered perceptron neural network (also known as the FFMLP neural network). This selection was based on earlier work such as that carried out by Brett and Li (2000), Tongpadungrod (2002), Ma and Brett (2002), Brett, Wright and Griffiths (2004) and Tam (2005). The FFMLP neural network has been successfully implemented in several tactile sensing systems, tailored especially for output discrimination. In this network, each neuron in one layer has direct connections to the neurons of the subsequent layer. Thus the neurons propagate the information in one direction only, forward, from the input layer to the output layer, via the hidden layer (i.e., there are no cycles or loops in this network). It is in the hidden layers where the activation functions are applied to the receiving neurons.

The role of the activation functions in the hidden layers is to introduce nonlinearity into the network. Without nonlinearity, the hidden layers will not make nets more powerful than just plain perceptrons, or matrix computation. The reason is that a composition of

linear functions is again a linear function. It is its nonlinearity (i.e., the capability to present nonlinear functions) that makes the FFMLP neural network so powerful. Almost any nonlinear transfer function can be used, but functions such as the sigmoidal and the hyperbolic tangent are preferred and commonly employed. Transfer functions such as the sigmoid function gives a threshold of '0' and saturates at the value '1' when the inputs are - ∞ and + ∞ respectively. In contrast transfer functions such as the hyperbolic function saturates at '-1' and '+1' when the inputs are - ∞ and + ∞ respectively, hence the value of the outputs of the hidden nodes are bounded between [-1,1]. The advantage of using such functions is (because of their simple derivative criteria which importantly contribute for fast computation during training), the differentiation of the sigmoid function is $f(1-f)$, and the differentiation of the hyperbolic tangent is $(1-f^2)$, where f denotes the primary function. This criterion is thus best suited for a training scheme such as back-propagation (BP), in which the topology is based on the chain rule.

In the research, the activation function employed the hyperbolic function. This function was chosen for its better numerical conditioning. An output that produces both positive and negative values tends to yield faster convergence of training than functions that produce only positive values, such as the sigmoid function (Bishop, 2003).

3.7.3 Back-Propagation Training

A variety of learning techniques can be used to train an FFMLP neural network, in the current research the back-propagation training method was selected. Back-propagation is advantageous because the trained system generally tends to provide reasonable answers even when presented with inputs that it has never seen before (Tongpadungrod, 2002). In principle the technique is simply accomplished by computing the intrinsic error made by the net's (or interpreted output y_i) with reference to the targeted output t_i (refer to Figure 3.6), and using this to adjust the network parameters dedicated to the task. The training process is performed iteratively. A prerequisite to the success of the training of the

network is when the error converges to within a specified threshold. This process is usually established from off-line. A typical routine process follows the flow diagram shown in Figure 3.6, where τ is the sequential of the total number output set data Γ used for the training, $y(x^{\tau};\omega)$ is the output vector computed from the net, and t^{τ} is the targeted set output vector.

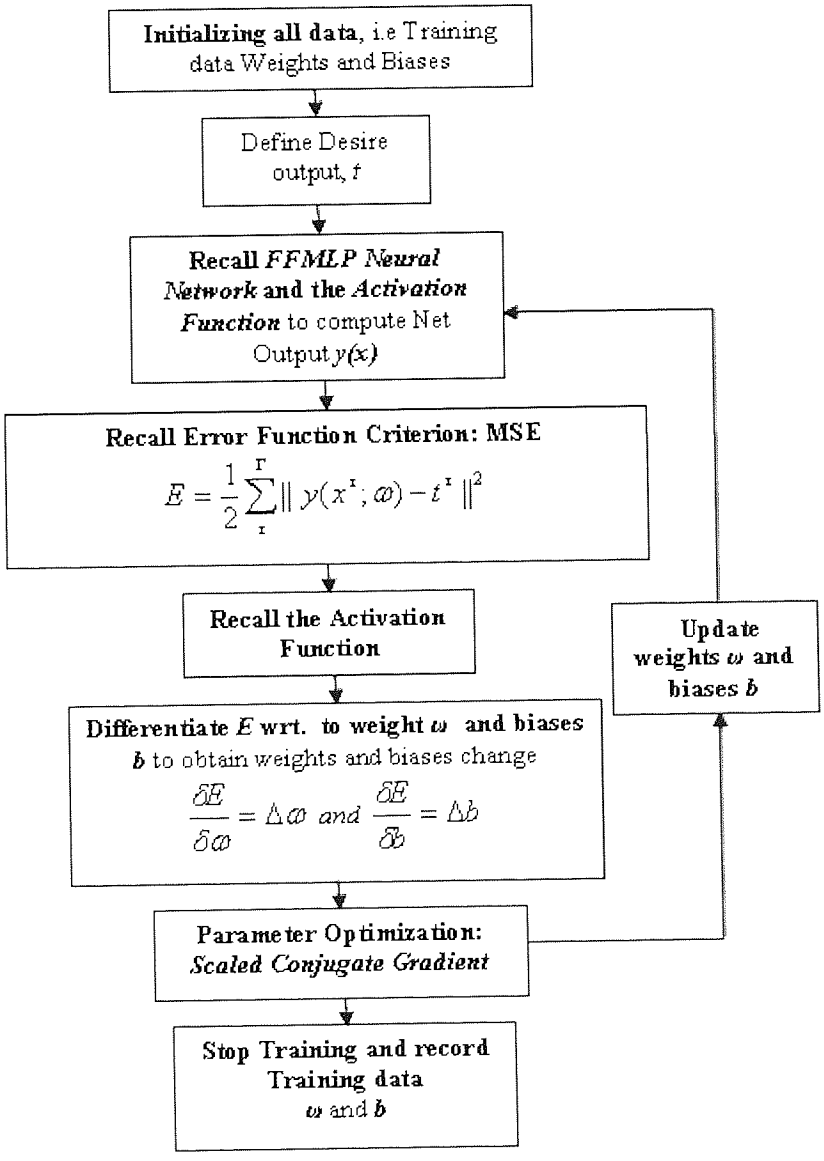


Figure 3.6: The FFMLP neural network back-propagation training flow process.

The objective during training is to minimise the error between the interpreted output and the target output by searching for the appropriate weights and biases (parameters). In the current research mean square error (equation (3.6)) was used for the back-propagation training. Equation (3.6) can also be regarded as the negative log-likelihood (Bishop, 2003).

$$E_{mean\ square\ error} = \frac{1}{2}\sum_{\tau=1}^{\Gamma} \| y(x^{\tau};\omega) - t^{\tau} \|^2 \qquad (3.6)$$

Both the error function and the activation function have to be differentiable for the delta chain topology to work. The error of the output layer is computed first and then the result is propagated towards the input layer. This is the essence of the back-propagation technique. The delta errors with respect to each layer are used for updating the weights and biases parameters using an optimiser function. The gradient decent or scale conjugate gradient can be used as optimiser functions; in this research the scale conjugate gradient was used. Although it requires more mathematical computations, this optimiser is substantially more powerful than other common optimisers such as the simple gradient decent (refer to Bishop, 2003; Nabney, 2004). The update parameters are then brought back for error re-computation. This process repeated until the decision is made to stop the training, before the network goes into over-training. Over-training is when the network starts to approximate the training input very well, but by doing so it thus inherently incorporates a large generalisation error. On the other words the network tends to memorise rather than to learn. More information on generalisation errors and means of dealing with them can be found in Section 3.8.3.

3.8 Neural Network Setup and Optimisation

This section explains the preliminary empirical research which was needed to establish distributive sensing for the discrimination of load parameters. This discussion covers how the inputs from the sensor outputs and the load parameter outputs used for training the neural network were obtained, and how sensor output was verified and validated for the experiment. This section also demonstrates the optimisation of the neural network prior to each application, and the analysis of the performance. The advantage of optimising the neural network is that it leads to the network having reduced complexity.

3.8.1 Neural Network Inputs and Outputs

One of the capabilities of neural networks is their ability to learn from experience. For a neural network such as the FFMLP this is a supervised type of training, where a set of input data and the corresponding target outputs have to be delivered into the training scheme. In this case the outputs are the load and load position derived by placing four different loads (6g, 9g, 11g and 13g) at specific points along the range of the beam. In practice, the input (deflection) was measured by the half-bridge strain gauge circuit, with the output being fed via an analogue to digital converter (ADC) to an embedded signal processing system. The processed signals representing the deflection were recorded from the digital to analogue converter (DAC) output of the embedded system. The configuration used is depicted in Figure 3.7. Further details concerning the design of the embedded signal processing system, including algorithm implementation, quantification issues, and real-time performance are covered in Chapter 5.

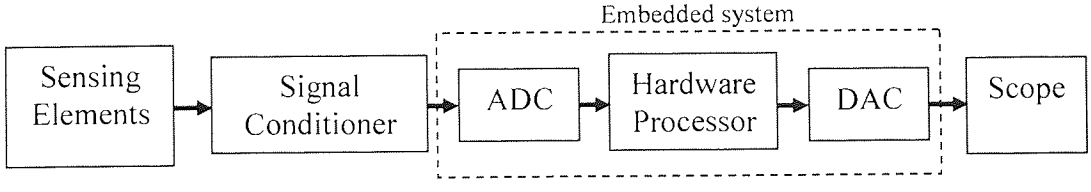


Figure 3.7: System setting for input measurement.

Usually the operating voltage range of the ADC and DAC of an embedded system is between -1V and +1V. This met the requirement that, in practice, the input and the target output variables of the neural network should be bounded within the normalised values of -1 and 1 (unity normalisation) (Bishop, 2003). No pre-processing or linear rescaling of the input variables is required with this setup. But pre-adjustment of the gain of the instrumentation amplifier is necessary to produce sensor output of not more than 0.90V from maximum deflection, and not less than -0.90V from no deflection, thus producing a common range from the four sensing elements. The ± 0.10V off-set voltage used here is to provide a safe margin from the threshold range and the ADC part of the device. Post-processing is then applied to the outputs. For the load, the maximum load was encoded into a normalised value of 0.9. Rescaling of 6g, 9g, 11g and 13g loads was done by dividing the loads by 14.44 (maximum load/0.9). This yield normalised loads $\overline{D}$ as 0.42, 0.62, 0.76 and 0.9 respectively.

3.8.2 Verification of the Sensor Inputs

In this research verification of sensor output was done by employing the bending moment, represented by equation, (3.5). The sensor outputs were obtained by placing a single load D (6g, 9g, 11g and 13g) to points along the beam I_B. Each load had a 12.7cm width dimension and was placed from the tip towards the root at every interval of 6.35mm, starting 6.35mm from the tip. The load placements covered 60% of the range from the tip. This range, which was also used as the working range has been found to be adequate to reflect the real operating range of a tactile endoscope for retrieving information during tissue manipulation and diagnosis operation (Ma, Brett, Wright and Griffiths, 2004). Sensor readings from sensor 1, sensor 2, sensor 3 and sensor 4 were recorded and the plots are displayed as Figures 3.8-3.11. For this process the beam length was normalised into unity, so that the value '1' represents the maximum length at the tip and '0' is the fixed end (root). Normalisation was done by dividing every position by total length I_B. Verification was undertaken by comparing the actual results from the

sensor outputs obtained, and the results in relation to theoretical bending moment obtained by simulation.

It follows that the relationship between the change of resistance from strain and stress, and the voltage out from the sensors, for the case with Wheatstone bridge of configuration discussed earlier can be shown by the expression below (Graham, 2006; Figliola and Beasley, 2006);

$$V_{out} = \frac{G}{k_\Omega}\frac{\Delta R_\Omega}{R_\Omega}V_{ex} + V_{off} \tag{3.7}$$

Where V_{out} is the voltage out, G is the gain of the strain gauge amplifier used, k_Ω is the resistance to voltage configuration factor (for example for the case of half bridge, ½, k_Ω is 2), $\Delta R_\Omega / R_\Omega$ is the ratio of change of resistance form deflection to the resistance of the strain gauge, V_{ex} is the excitation voltage and V_{off} is the voltage offset.

However, it was noted that the $\Delta R_\Omega / R_\Omega$ has a relation with the bending moment shown by;

$$\frac{\Delta R_\Omega}{R_\Omega} = \frac{G_f \gamma M_B}{EI} \tag{3.8}$$

Where G_f is the gauge factor, and γ is the distance from the neutral axis (in this case, the half height of the beam cross section). Incorporating (3.8) into (3.7) yields an expression relating voltage out V_{out} with the bending moment, M_B as shown below;

$$V_{out} = \frac{G}{k_\Omega}\frac{G_f \gamma M_B}{EI}V_{ex} + V_{off} \tag{3.9}$$

Knowing all the variables parameters in (3.9), the actual sensor output can be verified.

By examining Figures 3.8-3.11 it is clear that the readings obtained by the experiment and the simulation are in agreement. These demonstrate that the sensor outputs are valid.

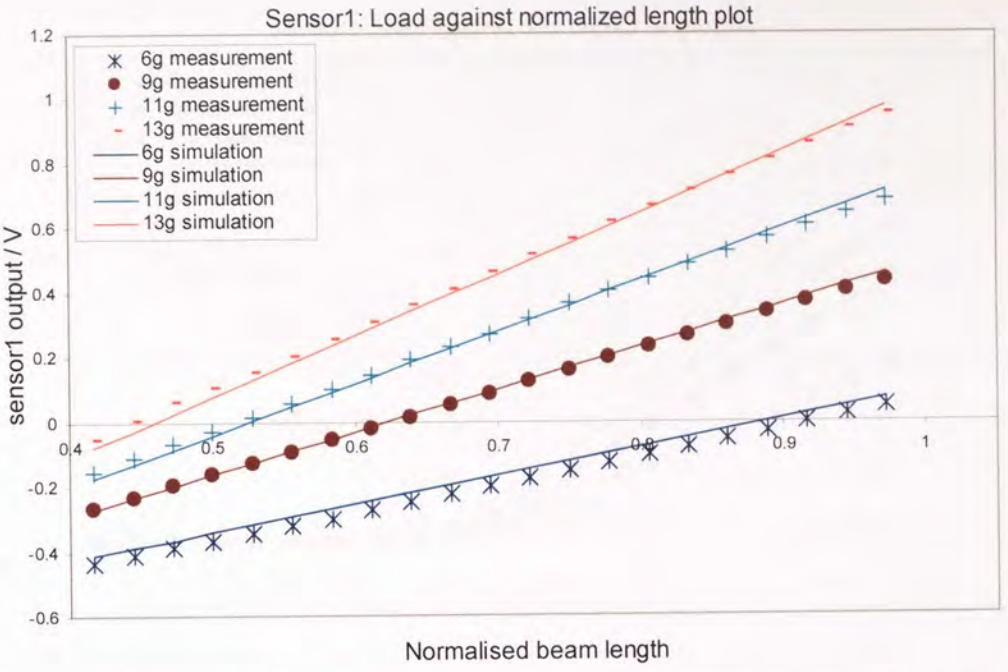


Figure 3. 8: Comparative plots of measurements obtained from sensor 1, and the simulation against normalised beam length.

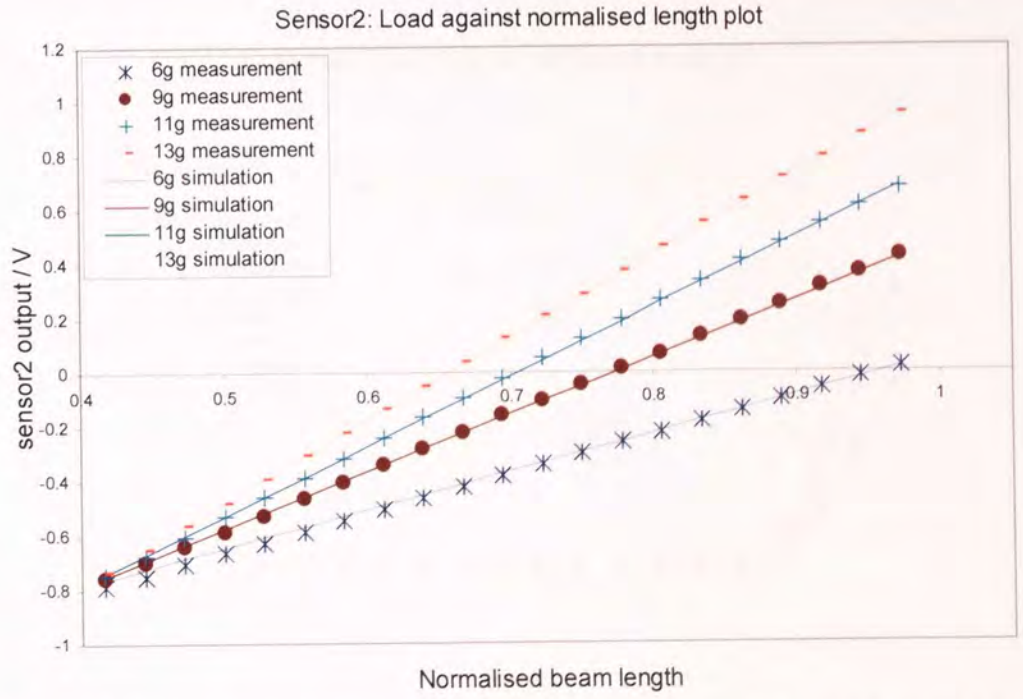


Figure 3.9: Comparative plots of measurements obtained from sensor 2, and the simulation against normalised beam length.

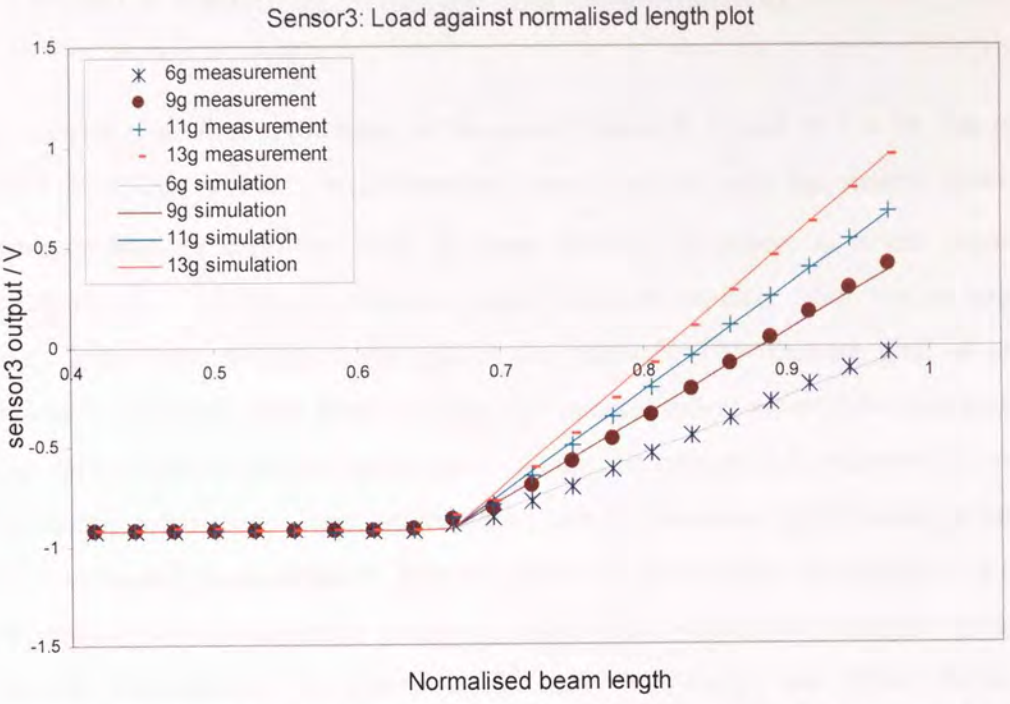


Figure 3.10: Comparative plots of measurements obtained from sensor 3, and the simulation against normalised beam length.

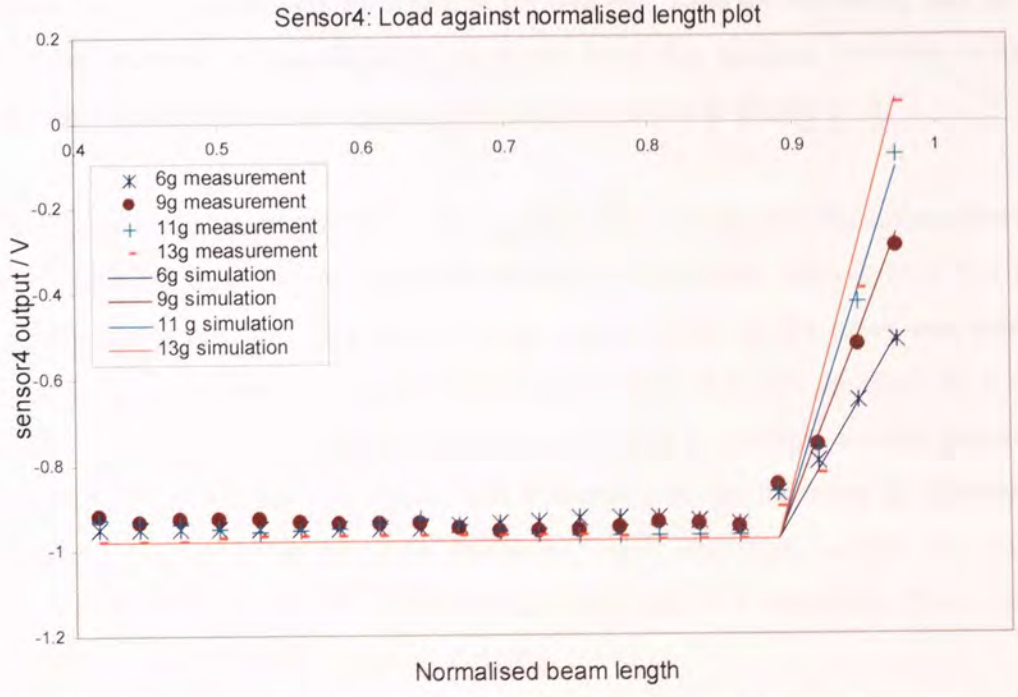


Figure 3.11: Comparative plots of measurements obtained from sensor 4 and the simulation against normalised beam length.

3.8.3 FFMLP Learning Strategy and Optimisation

To demonstrate the first application of the neural network, a total of $\Gamma = 76$ sets of data obtained from the previous measurements were used to train the neural network to discriminate loading condition load $\overline{D}$ along position i_p (where i_p is the normalised operating range). For training purpose neural network toolbox from Netlab package[1] (Nabney, 2004) was employed throughout the research. The ultimate goal of prudent training is to minimise both training error and generalisation error. However with only training data available, generalisation error can still be estimated by reserving some data for validating and testing. Then, overtraining can be prevented by choosing a network with low estimated generalisation. Another means of preventing over-fitting is to reduce the complexity of the network by removing some of the *unnecessary* weights (nodes). In practice this was achieved by optimising the number of inputs and hidden nodes to be used. A two-layered FFMLP neural network was trained off-line using the process explained in Figure 3.6. In this procedure, the total Γ training set data was first randomly permuted, 50% of which was used solely for training, 25% for validating and 25% for testing. An example of convergence of errors from the training between validation, testing and training obtained from the experiment is shown in Figure 3.12.

For the optimisation involved three cases: a two sensor case utilising information from sensors 1 and 2; a three sensor case utilising sensor information from sensors 1, 2 and 3; and a four sensor case utilising sensors 1, 2, 3 and 4. Each of the cases was simulated with a different number of hidden nodes starting from 4 to 25, in steps of 1 for 50 iterations. The maximum number of hidden nodes used in the optimisation process and through which the decision was drawn was influenced by the objective of implementing the sensing processor system into hardware, such that the amount of extensive mathematical computation that can be implemented acts as a constraint. Early stopping regularisation was applied to avoid over-fitting.

[1] Netlab package developed by I. T. Nabney is a neural network toolbox for simulation of the neural network algorithm for use in research and development. It has a library of more than 150 Matlab functions and script for the most common neural network algorithms.

The estimated validation errors of the two, three and four sensor cases with response to the training using different hidden nodes applied to the network is shown in Figure 3.13. The minimum generalisation error generated from the two, three and four sensor cases occurred at six hidden nodes with a 0.00027 error, at 20 hidden nodes with a 0.000266 error, and at five hidden nodes with a 0.000284 error respectively. With the lowest minimum of error and a small number of hidden nodes, it was therefore calculated that the two sensor case is the optimum condition for implementing the feed forward neural network.

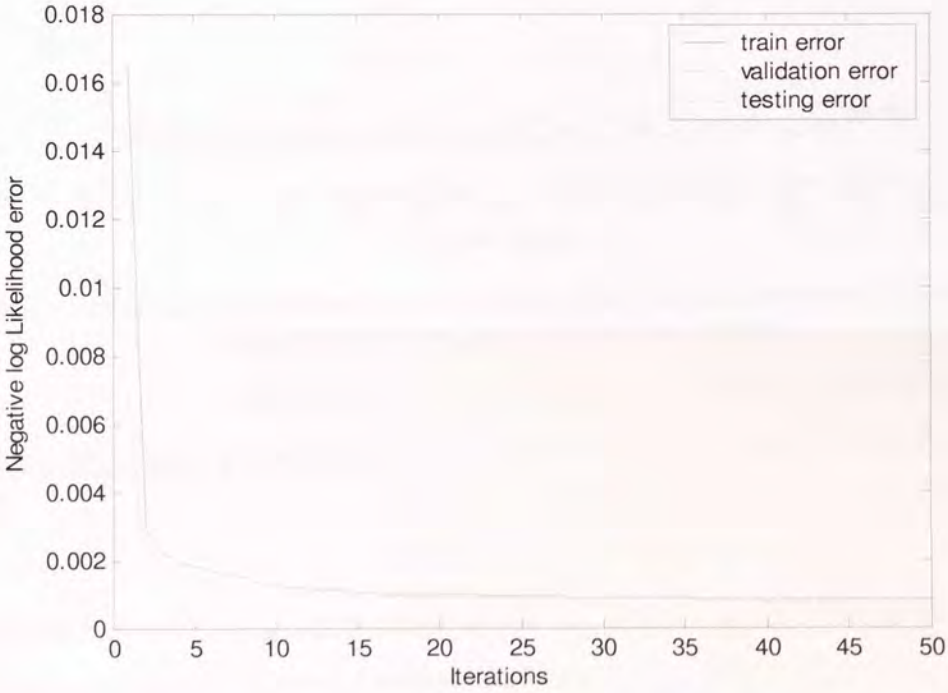


Figure 3.12: Example of different responses to show errors as a function of number of iterations to estimate load and position for training and validation data conditions (graph obtained from Matlab).

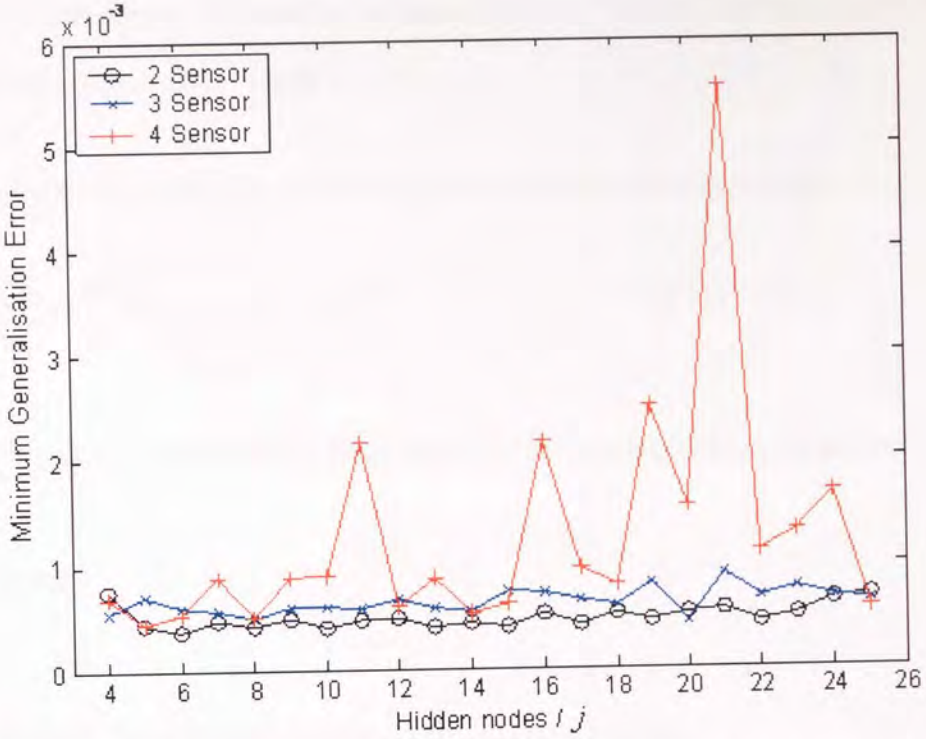


Figure 3.13: Minimum generalisation error with different hidden nodes applied for two, three and four sensor case conditions (graph obtained from Matlab).

3.9 Measuring Topology

The two-layered FFMLP neural network model explained in section 3.8 can be explicitly expressed in a function as shown in equation (3.10)

$$y_i = f(\sum_{j=1}^{M} \omega_{ij}^{(2)} r_j) \text{ where, } r_j = f(q_j) \text{ and } q_j = \sum_{k=1}^{N} \omega_{jk}^{(1)} x_k \qquad (3.10)$$

Where x_k are the input signals from the strain sensors, y_i denotes the output vectors representing load and position, q_j denotes the output vectors from the input integral, f represents the transfer function of the process, r_j denotes the output vectors of q_j after undergone the transfer function, and N and M denote the number of input units and the

number of hidden units. The weights are denoted by $\omega_{ij}^{(2)}$ and $\omega_{jk}^{(1)}$, where j is the "source" of the connection and i the "target".

The neural network defined by equation (3.10) was generalised as follows:

$$y_i = \sum_{j=1}^{M} \omega_{ij}^{(2)} f(\sum_{k=1}^{N} (\omega_{jk}^{(1)} x_k) + b_j^{(1)}) + b_i^{(2)} \qquad (3.11)$$

Where $b_j^{(1)}$ and $b_i^{(2)}$ are the offset biases and f is the activation function defined by:

$$f(z) = \tanh(z) \qquad (3.12)$$

3.10 FFMLP Neural Network Simulation

Equation (3.11) was modelled directly using Matlab and Simulink as shown in Figure 3.14. The model was implemented using two inputs and six hidden nodes, and the sets of weights and bias parameters derived from the training. This network retains the dataflow concurrency implicit in the neural network structure and is comprised of three main components: the input integral function where the inputs are multiplied by the weight $\omega_{jk}^{(1)}$ and summed with the offset biases $b_j^{(1)}$; the 'tanh' function f for the activation transfer function; and the output integral function where the activation function outputs are multiplied with the second set of weight $\omega_{ij}^{(2)}$, before they are summed with the second set of biases $b_i^{(2)}$.

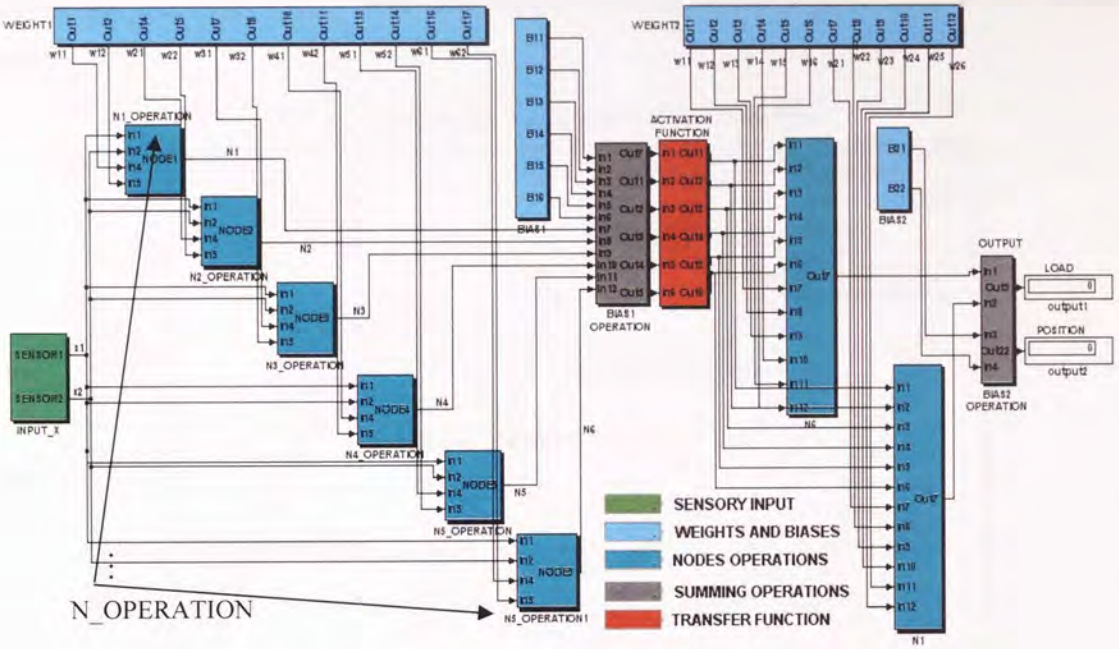


Figure 3.14: Simulink model of 2 : 6 : 2 FFMLP neural network (each of the blocks is coloured relative to their function).

The purpose of the biases is for fast and more accurate convergence during training (Bishop, 2003; Nabney, 2004). The biases add an extra input and an extra hidden neuron, and thus an extra (6,1) vector has to be added to the first layer and a (2,1) vector to the second layer. In Figure (3.14) the green block denotes the sensory input, the cyan blocks of the input integral denote the nodes operating between $\omega_{jk}^{(1)}$ and the sensory input, another set of cyan blocks of the output integral denotes the nodes operation of $\omega_{ij}^{(2)}$ and the output from the activation function. Further, red blocks denote the tanh activation function and light blue blocks denote the block storing trained parameters $\omega_{jk}^{(1)}$, $b_j^{(1)}$, $\omega_{ij}^{(2)}$ and $b_i^{(2)}$. Grey blocks denote all the summing operations.

Verification and validation of the neural network scheme used for this approach was undertaken by simulating the model using the input set τ taken sequentially from Γ. In the simulation double precision accuracy was used. Figure 3.15 reveals the comparison between the targeted and the approximated output. The relative error for the load and the

position are shown in Figure 3.18 and Figure 3.19 respectively (labelled "MLP 6 nodes error").

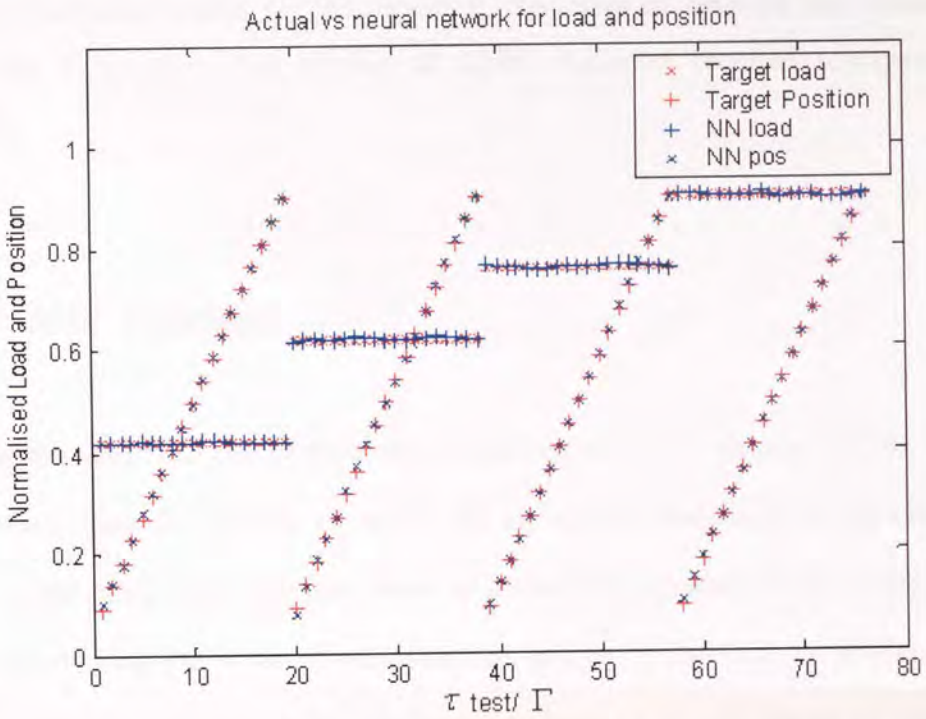


Figure 3.15: Comparison of the results obtained by actual load and position (labelled in the figure by target load and target position), and simulated load and position FFMLP (labelled in the figure by NN load and NN position) output against τ data sets for testing.

3.11 Feed-Forward Radial Basis Function RBF: An Overview

A different class of supervised feed-forward neural network, which will also be discussed in this chapter but not implemented into the embedded system, is the radial basis function neural network RBF (more detail of the type can be found in Bishop (2003) and Nabney (2004)). The RBF is the major practical alternative to the multi-layered perceptron. This network has two layers of processing. In the first layer the input is mapped onto each RBF in the hidden layer. In this process the Gaussian function is used as the basis function for this layer. One of the criteria which draw most researchers' attention to the

RBF is its two-stage training procedure, which is considerably faster than the method used to train the FFMLP. However, despite this advantage, following an investigation of the implementation issues for this research, this class of network was deemed to be inadequate in terms of the amount of digital resources required (compared to the FFMLP).

3.11.1 RBF Topology

The feed-forward RBF can be described by the expression in equation (3.13), where x_k, is the inputs from the sensors, ω_{ij} and b_i are the weights and biases of the output layer, and ϕ_j is the radial basis function which is defined by equation (3.14) where C_{jk} is the vector determining the centre of basis function, and σ_j is the width parameter. Both C_{jk} and σ_j are the weight parameter of the input layer. y_i is the output of the network. $\| X - C_{jk} \|^2$, as shown in equation (3.14), denotes the square of the Euclidean distance between two vectors.

$$y_i = \sum_{j=1}^{M} \omega_{ij} \phi_j (x_k) + b_i \qquad (3.13)$$

Where;

$$\phi_j(X) = e^{-\frac{\| X - C_{jk} \|^2}{2\sigma_j^2}} \qquad (3.14)$$

Similar to FFMLP, the purpose of training is to minimise the error between y_i and t_i by searching for the best parameter values for the weights and biases. In this case these are C_{jk} and σ_j from the first layer and ω_{ij} and b_i from the second layer.

3.11.2 RBF Simulation

The network was trained for two inputs with six hidden nodes and two outputs as for the FFMLP discussed previously, using *trainrbf* command of Netlab toolbox. Together with the acquired trained weight parameters, C_{jk}, σ_j, ω_{ij} and b_i, equation (3.13) and (3.14) were implemented directly using Simulink Matlab. The configuration of the design presented in Simulink block diagram form is shown in Figure 3.16. The network comprises three main components; the input function integral, the transfer function and the output integral. The purpose of the input function integral is to evaluate the width factors and the squared norm matrix of dimension j = 6 by computing the trained parameters C_{jk} and σ_j and the input information from the sensors. The transfer function used is the Gaussian function for activation. For the output integral the output from the activation functions are multiplied with the second set of weights ω_{ij} before they are summed with the second set of biases b_i. In Figure 3.16 the green block denotes the sensory input, the cyan blocks of the input integral function denote the nodes operation for the squared norm matrix, the cyan blocks of the output integral function denote the nodes operation of ω_{ij} and the output from the activation function. In addition, the red block denotes the Gaussian activation function, the blue blocks denote the operation of the width factors and the light blue blocks denote the block storing train parameters C_{jk}, ω_{ij} and b_i. The grey block denotes all the summing operations.

The RBF neural network model was simulated using the same data as applied to the MLP case, with associated results as shown in Figure 3.17. From the figure it was deduced that for the standard six hidden nodes, the approximation exhibited lesser accuracy than the MLP. The result shows that the position approximation for the minimal load of 6g exhibits deterioration with an error of up to 0.1. The error deteriorates further with a higher load condition, as with 13 g the error is up to 0.45.

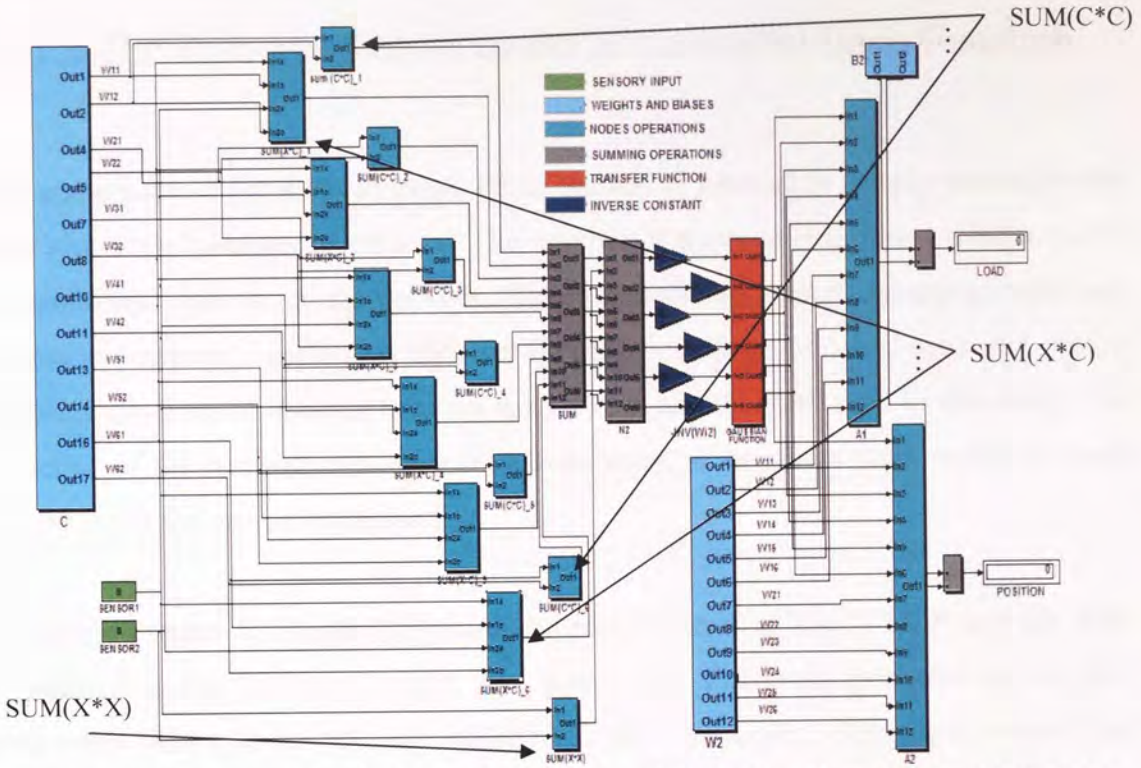


Figure 3.16: Simulink model of 2 : 6 : 2 RBF neural network (each of the blocks is coloured relative to their function).

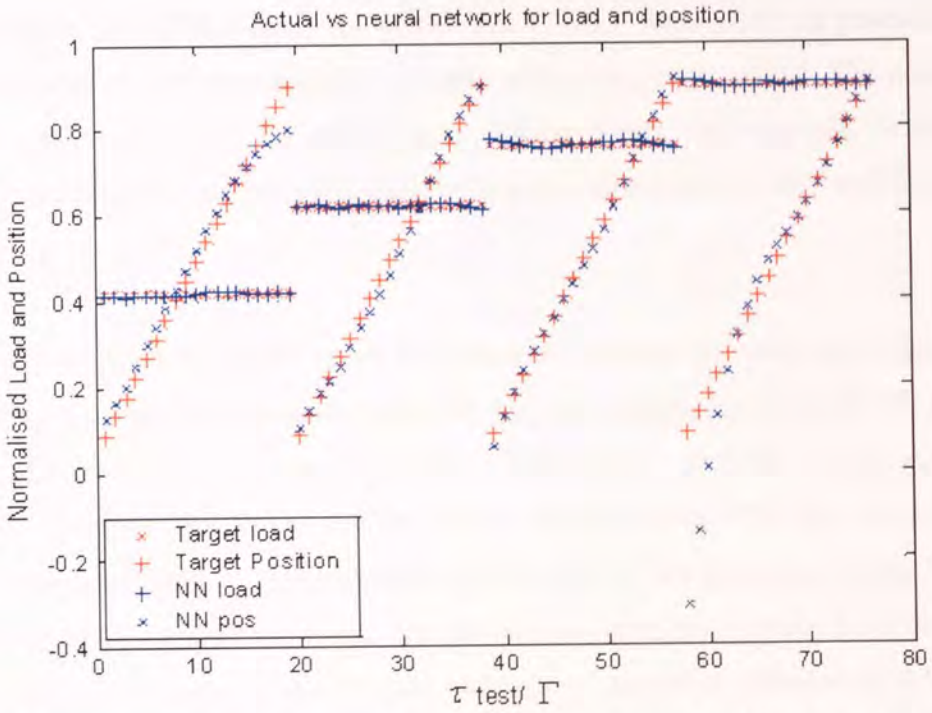


Figure 3.17: Comparison of the results obtained by actual load and position (labelled in the figure by target load and target position), and simulated load and position RBF (labelled in the figure by NN load and NN position) output against τ data sets for testing.

3.11.3 The Resource Requirements of the Radial Basis Function

The complexity of the interconnected PE of the neural network is usually defined by the total number of neurons being used. The number of neurons is in direct relation to the computational cost to run the network. The computational amount associated with each neuron is, however, significantly different for different classes of neural networks. In this section a comparison is made between the FFMLP and the RBF used in this study. The objective of the research was to draw a conclusion as to which class would be more feasible to be for the current project.

Accuracy: Figures 3.18 and 3.19 show the relative error of the FFMLP and the RBF obtained from the results of section 3.11 and 3.12.2. The result generated by the RBF with seven hidden nodes was also included for the comparison. The figures reveal that the RBF with six hidden nodes (blue line) is distinctly worse than the FFMLP with six hidden nodes (red line). In terms of load error, the RBF exhibited a maximum error of -0.015, whereas the FFMLP rate was -0.006. The FFMLP works well for position, with a maximum error of 0.01, but the RBF is worse with a maximum of 0.4. The accuracy of the RBF can be improved by adding extra hidden nodes. For example, in the seven hidden node case, for load error the maximum error improves to -0.007 and for position the error is 0.04.

Computational: With the exception of the activation function, the total multiplication and addition was counted and compared between the two classes. For the RBF, in the input function integral, each subsystem operation (cyan blocks) denoted by sum(C*C) and sum(X*C) has three multipliers and one adder. Whereas sum(X*X) has two multipliers and one adder, resulting in 38 multipliers and 13 adders. The summing blocks SUM and N2 (both in grey) have 12 additions. The inverse constant gains (blue) have the total of six multiplications. In the output integral, each nodes' operation (denoted by A1 and A2) required six multiplications and one addition. Also, two addition bias operations are required. These give a total of *56* multiplications and *29* additions. For the FFMLP, in the

input integral, each nodes' operation denoted by N_OPERATION (cyan) has two multiplications and one addition. The summing block (grey) has six additions. In the output integral each nodes' operation has six multiplications and one addition. The total number of multipliers and adders for the FFMLP is *24* and *26* respectively.

For the case of RBF having seven hidden nodes, an extra five multiplications and four additions are required by the input function integral, and an extra two multipliers for the output integral, giving a total of *65* multipliers and *33* adders.

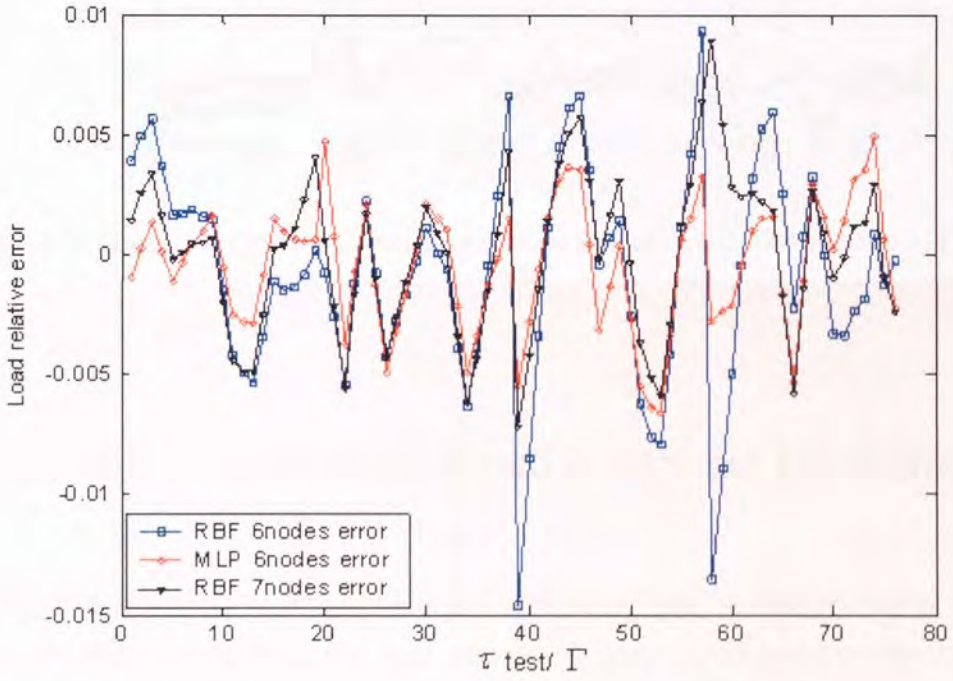


Figure 3.18: Comparative load relative error between RBF of six and seven nodes (labelled in the figure by RBF 6nodes error and RBF 7nodes error respectively) and FFMLP of six nodes (labelled in the figure by MLP 6nodes error).

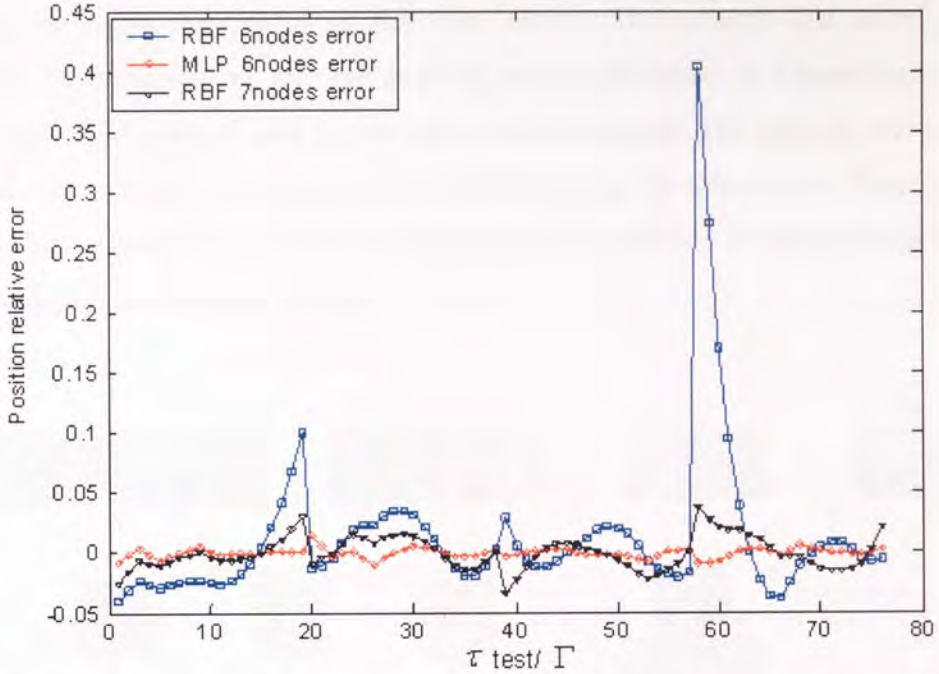


Figure 3.19: Comparative position relative error between RBF of six and seven nodes and FFMLP of six nodes.

3.12 Load, Load Position, Load Width and Load Shape

The FFMLP neural network was extended to discriminate the load width and load shape. Eight different configurations of load parameters were introduced into the training. The configurations of the loads are depicted in Figure 3.20. The loads *A*, *B*, *C*, *D*, *E*, *F*, *G* and *H* have weights of 23g, 19g, 12g, 8g, 11g, 9g, 6g and 16g respectively, and widths of 50.8 mm, 38.1 mm, 25.4 mm, 25.4 mm, 12.7 mm, 12.7 mm, 12.7 mm and 25.4 mm respectively (note that all widths are with respect to side view). For the shape, since the loads *A*, *B*, *C*, *D*, *E*, *F* and *G* have a rectangular profile, they were discriminated from H by coding them as '2'. Load H, which has the profile of a triangular shape as depicted in Figure 3.20, was coded as '1'. Using the same method explained in section 3.8.1, post-processing was applied to the load parameter output to obtain the normalised output.

This data is presented in Table 3.1. For the input, pre-adjustment to the gain was done to condition the signal acquired from the four sensors. The sensory data output prior to every trial of measurement for each loading case is presented in Appendixes 2. There were a total of 143 sets of data Γ, available for the training. The data set was randomly permuted. The network was trained and optimised using the information from four input sensors. During the training, 50% from the data set was chosen for the training stage, 25 % for validation and 25% for testing.

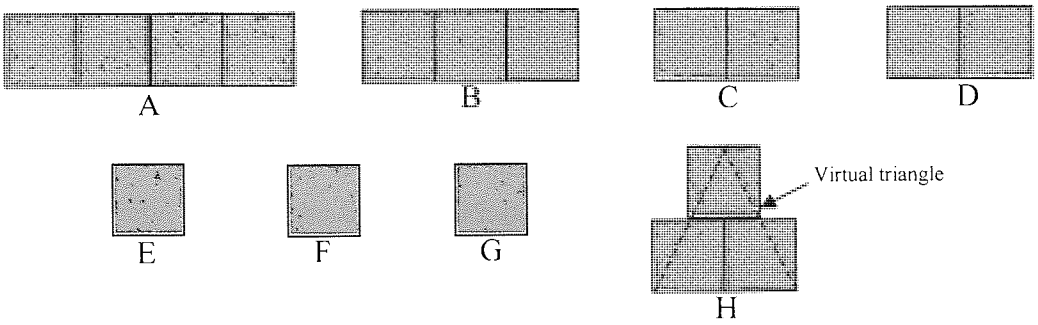


Figure 3.20: Load tests configuration.

	LOAD		Width		Shape	
	(D) /g	$(\overline{D})$ /Normalised	(W) /mm	$(\overline{W})$ /Normalised	(S) /Shape	$(\overline{S})$ /Normalised
A	23	0.9	50.8	0.9	rectangular	0.45
B	19	0.744	38.1	0.675	rectangular	0.45
C	12	0.470	25.4	0.45	rectangular	0.45
D	8	0.313	25.4	0.45	rectangular	0.45
E	11	0.431	12.7	0.225	rectangular	0.45
F	9	0.352	12.7	0.225	rectangular	0.45
G	6	0.235	12.7	0.225	rectangular	0.45
H	16	0.626	25.4	0.45	triangular	0.9

Table 3.1: Normalised loading data.

The neural network assessed up to now is termed a Single neural network. This term will be used throughout this thesis. An alternative way of realising the neural network is by using a Multiple or Cascade architecture. Both of these architectures employ the same

class of neural network, but separate the networks in order to determine each parameter individually. In the Multiple architecture all of the networks share the same number of sensor inputs (see Figure 3.21). In the Cascade type however, the input to the subsequent network(s) is the same input as to the previous network(s), but with extra input from the output of the previous network(s). This is why it is called a Cascaded neural network. Figure 3.22 shows the proposed arrangement of the Cascade network. There are four external inputs (from the four sensors) to the network. These are the sole inputs to neural network A. In the case of neural networks B, C and D, there is an additional one, two and three inputs respectively. The additional input, which is complementary to the sensory signal received by neural network B, is the discriminated load d computed by neural network A. Similarly, the two additional inputs received by neural network C are the discriminated load d and discriminated position i_m discriminated by neural networks A and B respectively. Finally, the three additional inputs received by neural network D are the discriminated load d, discriminated position i_m and discriminated width w computed by neural networks A, B and C respectively. Load was selected as the primary parameter because it has the most influence on the deformation of the beam. In practice, due to the time delay in the operation of each of the neural networks, i_m can only evaluated by neural network B when d is completely computed by neural network A, w can only evaluated by neural network C when i_m is completely computed by neural network B, and s can only evaluate when position w is completely computed by neural network C. Therefore the external inputs to the Cascaded nets B, C, D have to be delayed to prevent acceleration of the algorithm.

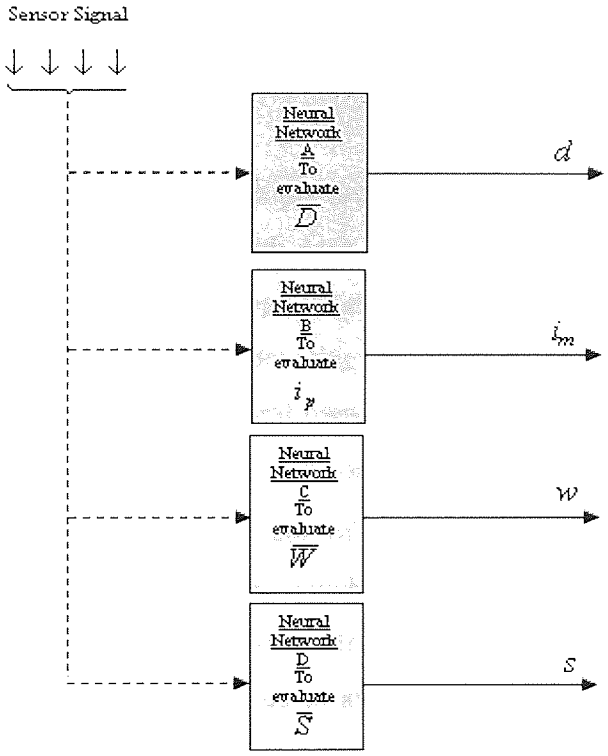


Figure 3.21: Multiple networks to determine load parameters.

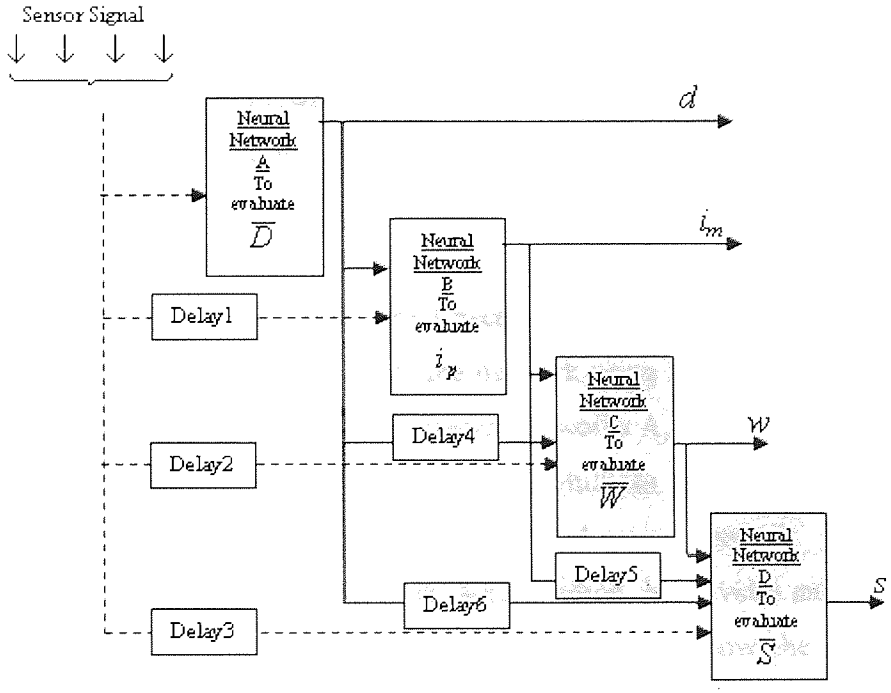


Figure 3.22: Cascade chain of neural networks to determine load parameters.

3.12.1 Training, Optimisation and Testing of a Cascade Neural Network

The training and optimisation of the Multiple architectures follows directly the method exercised for the Single network, but with each of the parallel single-output networks trained independently. The training and optimisation procedure has also been applied to the Cascade architecture, but utilising *training by-parts* procedure. Specifically, the training involved first optimising the neural network A, using solely the input from the sensory data x_k. During the optimisation, network A is repeatedly trained with varying hidden nodes, starting with 4 to 25 in steps of 1 in order to search for the optimum number for the hidden nodes (i.e., nodes which generate the minimum generalisation error throughout the training). In each of the training stages, minimum number of 50 iterations (each having 50 cycles) was used to avoid over training and over-fitting. The weights ω and biases b parameter corresponds to the optimised hidden nodes were recorded and used for the testing. Using equation (3.11) with the determined optimised number of hidden nodes and the corresponding weights ω and biases b, a single output vector of load d was obtained. The evaluated output, together with the sensor input, was then used for optimising and training neural network B using the same procedure. The process was repeated for neural networks C and D. The graphs showing the optimisation of Single, Multiple and Cascaded neural network utilising $N = 4$ inputs from the sensors are attached in the Appendixes 3.

The optimal number of hidden nodes for the Cascaded neural networks A, B, C and D were five, five, five and six respectively. For the network using the multiple architecture the optimised number of hidden nodes for the neural networks A, B, C, D were five, nine, eleven and five. The performances of the simulated Multiple and Cascaded networks were compared to the performance obtained by the 4:16:4 neural network. Figures 3.23 to 3.25 depict the comparative results for load, load position, load width and load shape discrimination between the different arrangements. The graphs show the performance defined by average percentage error μ of the load parameters against the actual position

along the beam. The average percentage error μ at any specific point along the range is defined by the sum of all percentage errors η exhibited by the load parameters at that specific point, divided by the total number of measurements being made at the point (equation (3.15)). The percentage error at that specific point is the absolute error between the actual load parameter t, and the computed load parameter output y, divided by the load actual parameter t, and multiplied by 100 (equation (3.16)). For the position and width percentage error however, adjustment to equation (3.16) was done so that the actual reference would be based on the actual length of the operating range. This was defined as the local error, which is more useful for the operator (such as the surgeon) to interpret.

$$\mu_{d,i_m,w,s}(i_p) = \frac{\sum_{1}^{\lambda} \eta_{d,i_m,w,s}(i_p)}{\lambda} \qquad (3.15)$$

Where, λ is the total number of measurements at the point:

$$\eta_{d,i_m,w,s} = \frac{|t_{\bar{D},i_p,\bar{W},\bar{S}} - y_{d,i_m,w,s}|}{t_{\bar{D},\bar{S}}} 100 \qquad (3.16)$$

The average, minimum, and maximum errors for load-magnitude, load-position and load-width are also shown in Figures 3.23 to 3.25. Figure 3.23 is the comparison between 4:16:4 (load output), the 4:5:1 Multiple network and the 4:5:1 Cascade network for the load output. Figure 3.24 is the comparison of 4:16:4 (position output), the 5:9:1 Multiple network and the 5:5:1 Cascade network for the position output. Figure 3.25 is the comparison of 4:16:4 (width output) with the 6:11:1 Multiple network and the 6:5:1 Cascade network for the width output. Finally, Figure 3.26 is the comparison of 4:16:4 (shape output) with the 7:5:1 Multiple network and the 7:6:1 Cascade network for shape output. Load magnitudes were computed to mean accuracies of better than 0.85% (4:16:4), 0.58% (Multiple) and 0.58% (Cascade); load-positions to mean accuracies of better than 2.19% (4:16:4), 1.02% (Multiple) and 0.63% (Cascade) and over the working range; load-width to mean accuracies of better than 4.29% (4:16:4), 3.54% (Multiple) and 0.63% accuracy (Cascade) and load-shape to mean accuracies of better than 9.02%

(4:16:4), 2.95% (Multiple) and 2.61% accuracy (Cascade). These results demonstrate the superior convergence and accuracy of the cascaded solution. This is primarily due to the use of load-magnitude, load-position and load-width information in this case that qualifies the contribution of individual sensors in later stages of the cascade.

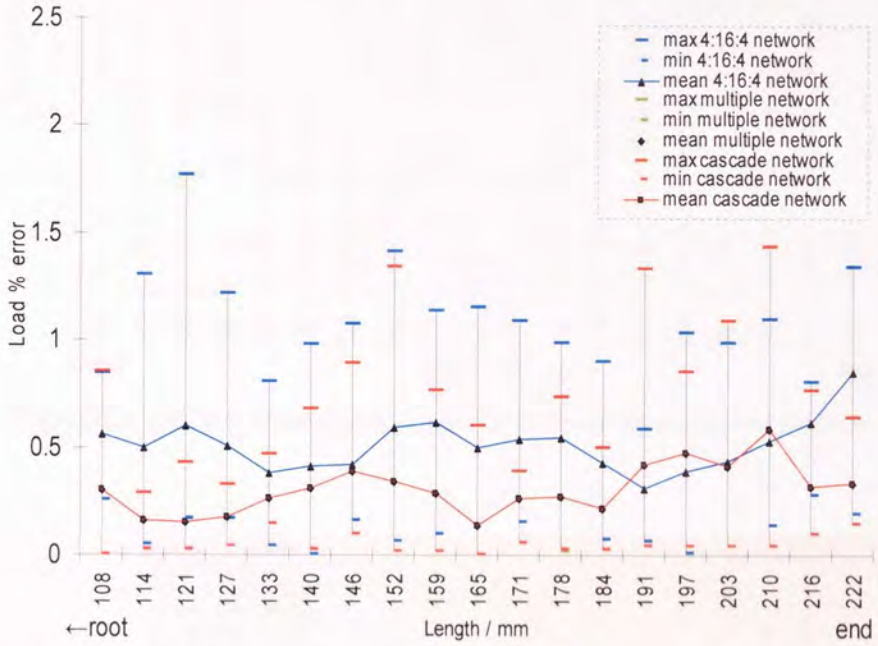


Figure 3.23: Load percentage error against the operating normalised position.

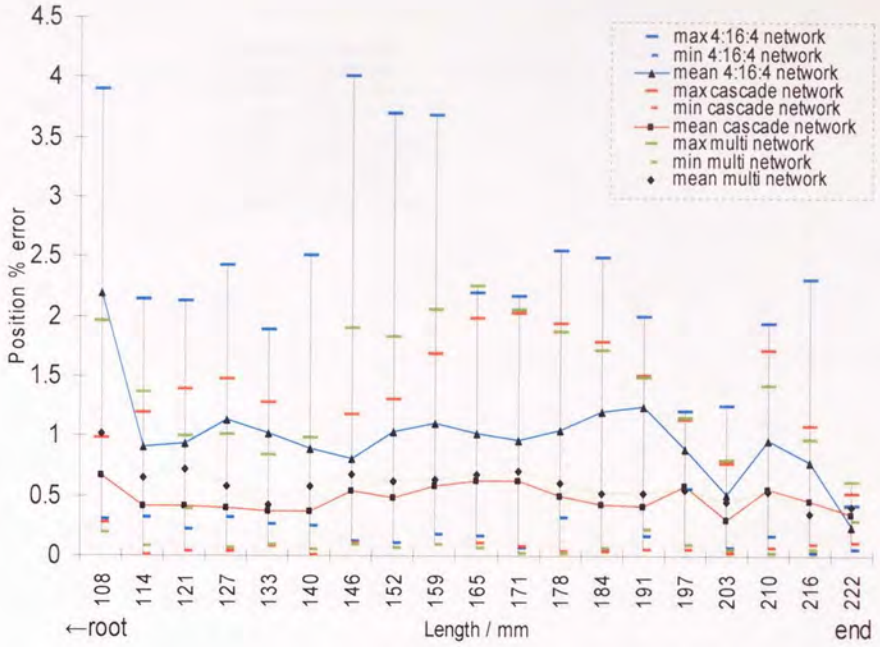


Figure 3.24: Position percentage error against the operating normalised position.

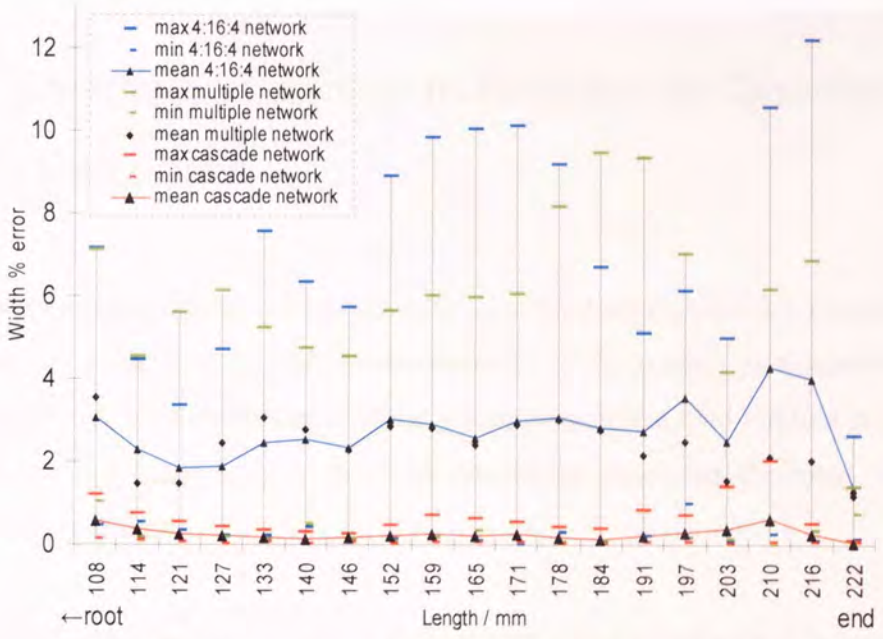


Figure 3.25: Width percentage error against the operating normalised position.

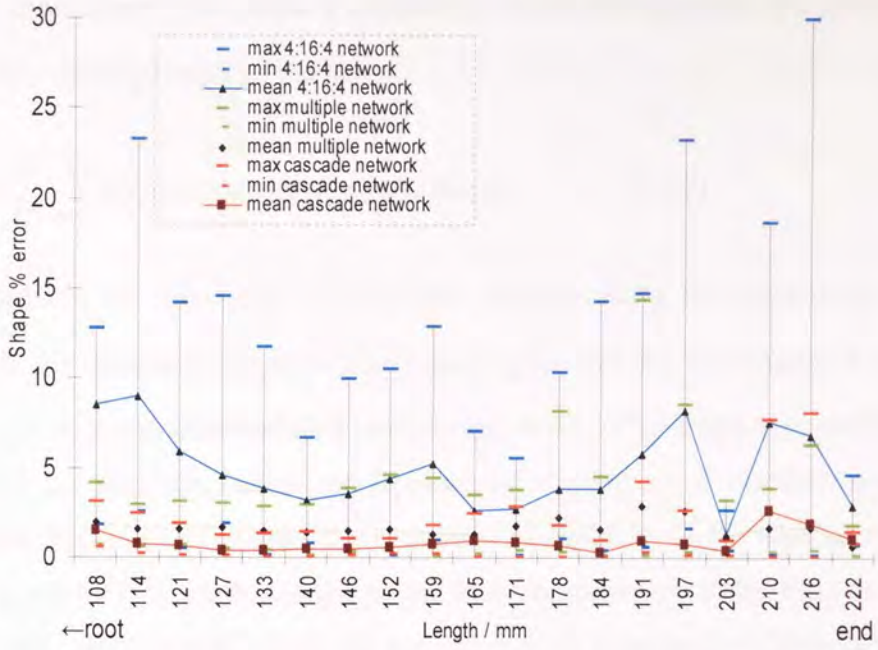


Figure 3.26: Shape percentage error against the operating normalised position.

3.13 A Continuous Function to Describe the Discriminated Loading Conditions

The distributive measurement which has been described throughout this chapter requires some means of visualising the continuous response of the output, particularly when the medium is exposed to a process and online monitoring of the load applied is needed. In this work, the approach uses a form of non-linear functions designed to aid the interpretation algorithm.

Two functions were introduced. First, a radical approach was adopted that used equation (3.17), which was obtained from the derivation of the band-gap equation using a simple harmonic approach (Lynn, Fuerst & Thomas, 1997). By having control of the rise time, maximum magnitude, fall time and the slew rate, the equation can be adapted to accommodate the evaluation of the continuous response, using variables from load-width,

load, load-position and load-shape. Consider the continuous function, $f(i_p)$ with applied load over the working length of the beam:

$$f(i_p) = 4d\sum_{m=1}^{P}\frac{1}{n\pi}\sin(m.w.\pi)\cos(m.i_m\pi)\cos(m.i_p\pi) \qquad (3.17)$$

Where i_p denotes the incremental normalised position along the normalised operating length, d is the discriminated normalised load, i_m denotes the discriminated normalised position, and w is the discriminated normalised width. The shape was defined by the sharpness of the response, which can be done by altering the P iterations used for the computation. Figure (3.27) shows the responses of equation (3.17) having condition of load: 0.5, position: 0.5, width: 0.3 and shape from sharpness of: 1000 (blue line, case A). Also load: 0.5, position: 0.5, width: 0.3 and shape from sharpness of: 10 (black line, case B). Finally load: 0.7, position: 0.5, width: 0.2 and shape from sharpness of: 1000 (red line, case C).

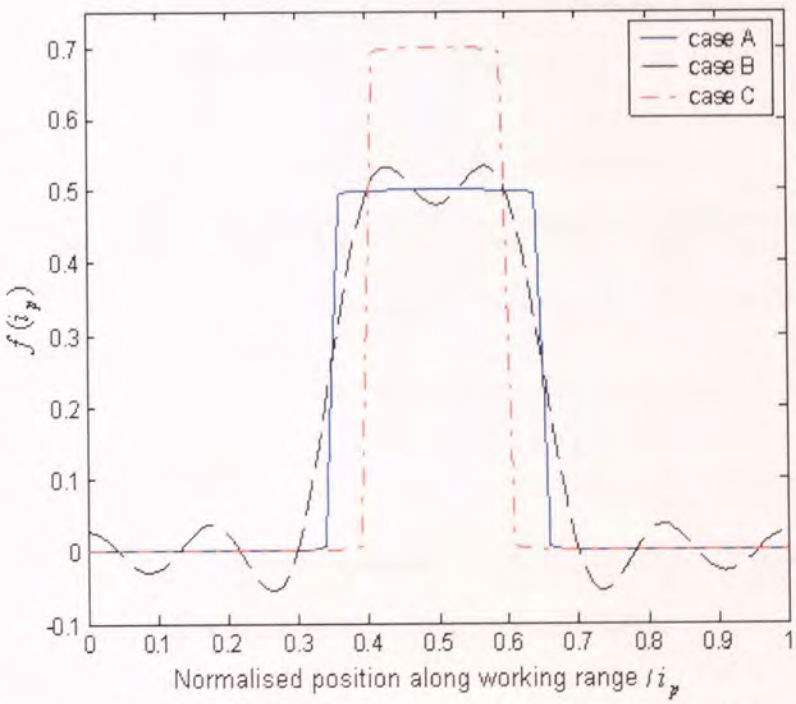


Figure 3.27: Response of equation (3.17) with various parameter conditions (graph obtained from Matlab).

Secondly, a continuous function was used that describes the continuous applied load function $f(i_p)$ over the operating length of the beam, as shown in equation (3.18).

$$f(i_p) = de^{-c^P(i_p - i_m)^P} \qquad (3.18)$$

Where the indices d, C, i_m and P relate to the amplitude, width, position and sharpness of the discriminated load distribution respectively. Figure 3.28 is the simulation output of equation (3.18) for three different conditions.

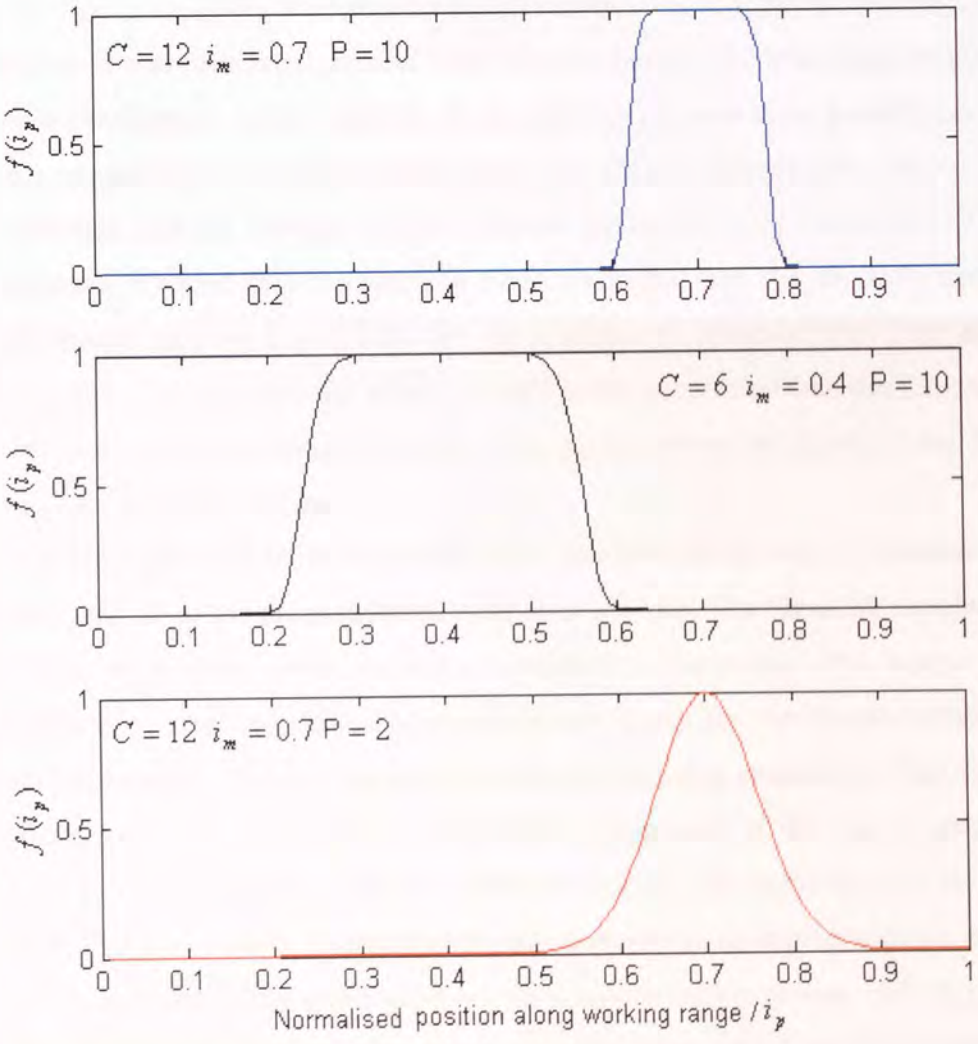


Figure 3.28: Response of equation (3.18) with various parameters conditions (graph obtained from Matlab).

3.14 Summary

This chapter has explained the development of a distributive tactile sensing system intended to be used for the flexible digit. This chapter has demonstrated that using a distributive sensing system with as few as four sensing elements is adequate for evaluating the load magnitude, load position, load width and load shape when applied on the digit (beam). This section of the thesis has explained some of the important factors which influenced the realisation of the system, such as signal conditioning of the sensor output. This is necessary to enable the interpretation tools to generate reliable information, especially if real time measurement is to be considered. The interpretation tool uses an artificial intelligence neural network. From previous research three possible architectures were proposed, namely a Single neural network used for evaluating the output parameter concurrently, and the Multiple and the Cascade neural network. The results of empirical assessments revealed that the Cascade performs better than the Multiple and then the Single neural network types, but with the expense of computational time and a high training cost. The Multiple has shown overall better accuracies than the Single, but with the expense of more complex network. Thus more favours are given to the Single and Cascade for implementation.

The implementation of these neural networks was done using Matlab Simulink computer software and all of the measurements were done off-line. The Simulink simulations were done for the models with explicit mathematical functions. The benefit of these experiments is that they provided a benchmark study for the implementation of the interpretation algorithm in a hardware distributive sensing processing. That is, from the steps described in this chapter the proposed arrangements of the neural networks were established for development into the hardware version. The importance of the activation function was also studied. Evidently, this will give rise to an implementation problem for the neural network. More information on the implementation process will be provided in Chapter 4. Information about the optimum configuration of the network (hidden nodes) and the training parameters that can be directly adopted into the specific hardware circuit is provided in Chapter 5.

CHAPTER 4

Developing an Implementation Friendly Activation Function for an Embedded System Solution

4.1 Introduction

There are various ways of implementing the neural network discussed in the previous chapter. In the current research the decision was made to implement the network as an embedded system employing an FPGA as the main programmable hardware chip. The advantages of this type of implementation are higher speed and the capacity for free stand alone operation. With the recent introduction of modern design flows and synthesis tools that essentially take a Matlab Simulink digital signal processing model as a high-level behaviour specification for a design, fast prototyping of neural network function can be achieved easily. However, this approach relies on the ability of the high level tool to capture and synthesise the mathematical expressions in the signal processing algorithm and of the FPGA to have sufficient resource to accommodate the synthesised design.

Difficulties arise due to the limited nature of the mathematics functions that most FPGAs synthesis tool can offer. Particularly difficulties arise in the case of the non-linear activation function (i.e. the transfer function which introduce nonlinearity to the neural networks, hence making the multilayered network more powerful (refer to Section 3.7.2)). This chapter describes the design of novel implementation-friendly digital activation functions which will be used to implement the neural networks into an embedded application-specific system. The primary focus of this work was the analysis of the selected activation function, the hyperbolic tangent (tanh), which is a function with an exponential argument functions that can not be synthesised directly and the mathematical representations are tedious if expressed using a conventional approximation such as the Taylor series expansion (Avci and Yildirim, 2003). This chapter then described research on devising salient techniques based on approximations of the actual characteristic and response, with the aim of generating an implementation-friendly solution to replace the 'non-implementable' explicit activation function. The conventional method is to use a Look-Up Table LUT, but that approach requires massive resources for comparatively small accuracy. Four methods were developed and implemented using a Xilinx XtremeDSP FPGA development board. These were the Gradient and the Polynomial scheme approximations, the Padé approximation and the Look-Up Table (LUT) based approximation. These methods were compared in terms of their performances, complexity of their designs, and efficiency with respect to hardware resources required.

4.2 Implementation System Overview

4.2.1 FPGA

An FPGA (Field Programmable Gate Array) is a general purpose semiconductor device that contains programmable logic components and programmable interconnections (Brown & Rose, 1996). In other words, it is a programmable chip with high density and memory elements (such as flip-flop), allowing the designer to program the duplicated

functionality of their complex combinatorial functions design, such as decodes or even polynomial mathematic functions. In general, FPGAs may be slower, have less ability to handle complex designs and draw more power than their Application-Specific Integrated Circuit (ASIC) counterparts. However, because of the important advantages they provide, such as flexibility and ability to be re-programmed (even after deployment) in the field to fix 'bugs' and lower non-recurring engineering cost[1], they were selected for this research.

Because of their quick turnaround in terms of design and production time, FPGAs have become a focus of much attention. This has led to an increase in the density and complexity of the programmable devices, and the flexibility offered by such devices prior to their application. Current achievements include the development of a fast platform FPGA on which to implement a design, with a pre-engineered high performance hardware platform to quickly verify functionality.

4.2.2 Design Methodology and Logic Synthesis

Recent developments in digital logic synthesis have greatly automated the process of devising an application-specific FPGA, when implementing a particular algorithmic function. Traditionally, such functions were developed using a hardware description and synthesis language such as VHDL (Very High Speed Integrated Circuit Hardware Description Language) that provides arguably poor support for capturing concurrency at a behavioural level. The emergence of synthesis tools based on concurrent extensions of the C programming language, such as Handel-C, facilitate the explicit capture of concurrent behaviour, such as that exhibited by neural networks (Pandya, Areibi & Moussa, 2005). However, such languages currently still have limited mathematical synthesis capability and only tenuous links to simulation models, such as Matlab Simulink. This problem has been overcome by the recent introduction of synthesis tools

[1] Also known as NRE, which refers to as the one-time cost of researching, designing, and testing a new product.

from Xilinx that take Matlab Simulink digital signal processing as a high-level behaviour specification for the design, and facilitate the direct synthesis of an appropriate digital logic circuit (using VHDL as an intermediate language, see Figure 4.2). The model can then be simulated (using bit and cycle-true simulation) using Simulink and compiled into a VHDL source using the Xilinx System Generator tool (Xilinx System Generator, 2006). The VHDL design is optimised for synthesis and implementation using Xilinx Virtex, Virtex-E, Virtex-II, or Spartan-II FPGA.

4.2.3 Xilinx VirtexII Xtreme DSP Development Board

In this research the XtremeDSP Development board from Nallatech was introduced as a targeted embedded system in an attempt to provide a complete development solution. The system was comprised of a PCI motherboard with a USB interface for standalone operation, and a daughter board (see Figure 4.1) featuring dual-channel high-performance ADCs and DACs, which use an AD6644ADC and an AD9772DAC respectively, a user programmable Virtex-II 2XCV3000 FPGA comprised of three million gates, and an internal programmable clock management from Virtex-II XC2V80 FPGA. The ADC and the DAC can operate with 65Msample/s and 165MSample/s respectively and both have a 14 bit accuracy. In practice, FPGAs can be configured through the USB with the help of FUSE software[2].

As the system is fully supported by the Xilinx Blockset and the Xilinx system generator, modelling and implementation of a system architecture can be completed relatively quickly and easily. The Xilinx Blockset is a collection of Simulink blocks that can be used to create and simulate designs with the Simulink environment, whereas the Xilinx system generator is the tool used to automatically generate the corresponding VHDL

[2] FUSE System Software originally developed by Nallatech, provides configuration, control and communications functionality between host systems and FPGA hardware. This enables developers to design complex processing systems, with seamless integration between software, hardware and FPGA applications.

code used for the synthesis and implementation into a Virtex-II 2XCV3000. Both elements support bit true and cycle true modelling of hardware, thus bridging the gap between high level system design and actual implementation in Xilinx FPGA.

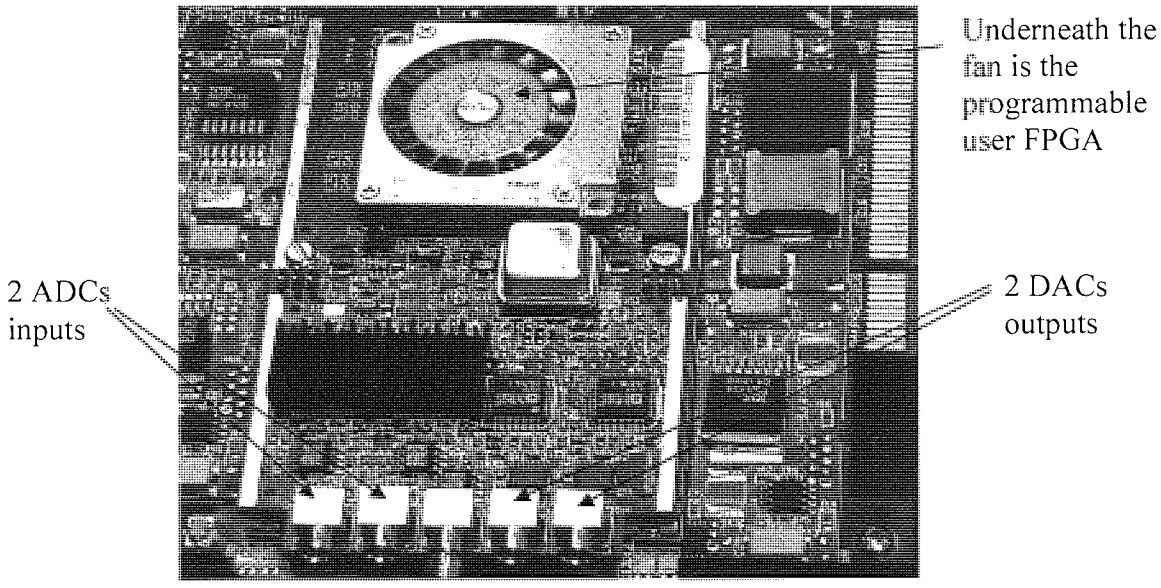


Figure 4.1: The Xtreme DSP development kit.

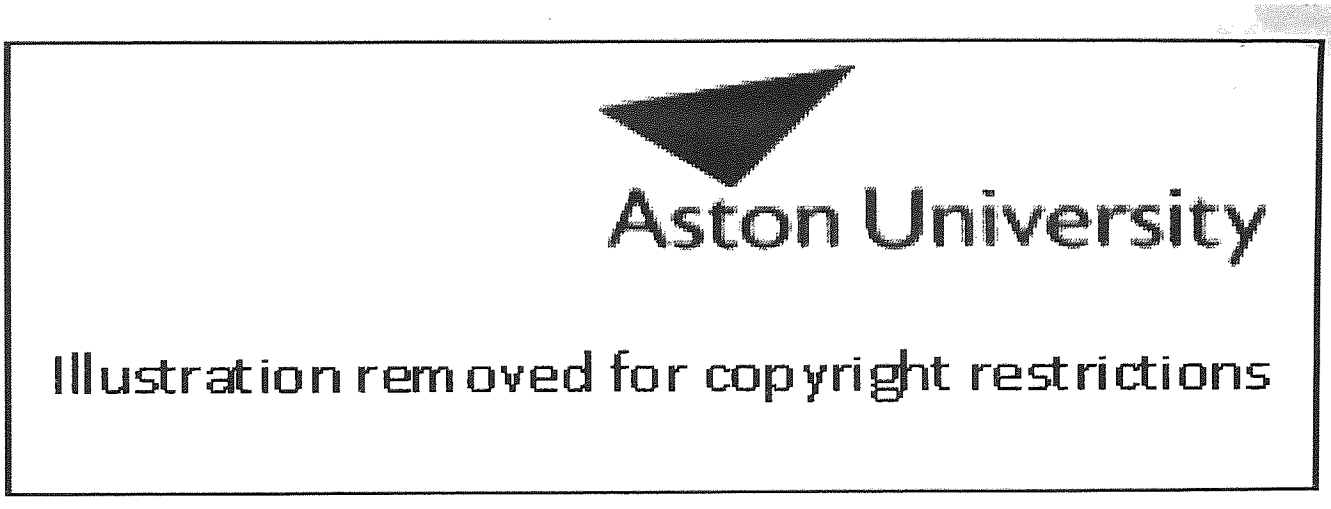


Figure 4.2: FPGA design flow.

4.3 Development of an Implementation Friendly Algorithm

In recent years, various techniques and design architectures have been developed for the implementation of neural networks into FPGAs. The aim of these projects is usually to optimise the neural network design and the usage of digital resources prior to their implementation using an FPGA. Other than the manipulation of the original linear PE processing architecture to facilitating resource re-use, attention should also be given to the function of the non-linear activation elements, as this will constitute the most computationally expensive component of the network. The traditional solution is to use a LUT utilising the EPROM of the chip, or to adopt and implement a Taylor series function, resulting in fully digital arithmetic neurons. A more radical solution is to use another function to replace the hyperbolic tangent function. Such research was undertaken by Guarnieri and Piazza (1999) in which an adaptive spline was used as the activation function. This resulted in the need to modify the training algorithm in order to adapt the feed-forward algorithm which incorporates the proposed activation function. Similar problem will also encounter if Field Programmable Analogue Array FPAA is used to generate nonlinear continuous function other than the hyperbolic tangent for the activation function. In this current research, however, a more effective approach is proposed, namely by approximating as close as possible the function of the hyperbolic tangent. The advantage of this method is that a general training tool can always be used, as this will be compatible with the feed-forward function used in this project. This section of the thesis presents the details of the pre-studies undertaken and the development of digital hyperbolic tangent activation functions.

4.3.1 The Hyperbolic Tangent

The preliminary research described in this chapter is that which was undertaken to investigate methods of implementing the hyperbolic tangent. The hyperbolic tangent is a nonlinear function which has properties that resemble those of the trigonometric function.

The function can be defined as:

$$\tanh(q) = \frac{\sinh(q)}{\cosh(q)} \qquad (4.1)$$

And can be generalised into:

$$\tanh(q) = \frac{e^{2q} - 1}{e^{2q} + 1} \qquad (4.2)$$

The graph of $\tanh(q)$ is sketched in Figure 4.3. It exhibited approximately linear behaviour within the q range -0.5 to 0.5. Beyond this range it starts to behave in a much more nonlinear way and its starts to saturate to -1 and 1 as q is increased beyond -3.5 and 3.5 respectively. But the most important characteristic of this system for the current project is that $\tanh(q)$ is an odd function, ($\tanh(-q) = -\tanh(q)$). The graph of $\tanh(q)$ shown that it is symmetric with respect to the origin. Thus the polarity of the function compel directly by the polarity of the input variable.

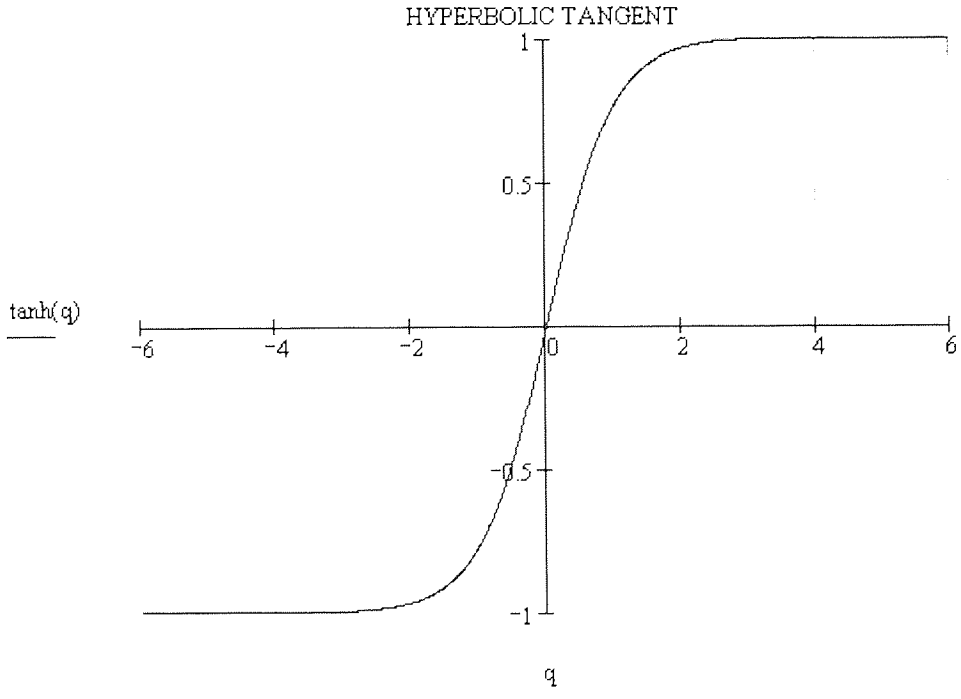


Figure 4.3: The hyperbolic tangent, $\tanh(q)$ (graph obtained using Mathcad).

The aim of these analyses was to approximate the function in the positive q plane. A range of techniques were evaluated, including piecewise method of approximation such as the Gradient and Polynomial schemes, continuous method of approximation such as the Padé approximations, and a digital hardware generating function which is based on the employment of Look-Up Tables (LUT).

4.3.2 Mathematical Function Method for Approximation

4.3.2.1 Gradient Scheme Approximation

The Gradient scheme approximates the positive q plane by a series of linear equations having gradients g and vertical-axis intercept i^g. The scheme divides the positive q range into n segments (note this term is also used to denote segments in Polynomial and Padé approaches). The total segment is approximated by a set of piecewise equations as shown in (4.3).

$$f(q) = \left|\tanh(q)\right| \cong f_{tot}(q) \begin{cases} f_1(q) = g_1 q + i_1^g \quad for \quad 0 < q \le q1 \\ f_2(q) = g_2 q + i_2^g \quad for \quad q1 < q \le q2 \\ \vdots \\ f_n(q) = g_n q + i_n^g \quad for \quad qn-1 < q \le qsat \\ f_{n+1}(q) = 1 \quad for \quad q > qsat \end{cases} \tag{4.3}$$

Where $qsat$ (which can also be denoted as qn) can be chosen where the output starts to saturate to '1'. For a better approximation, the number of segments can be increased within the region of highest nonlinearity as shown in Figure 4.4.

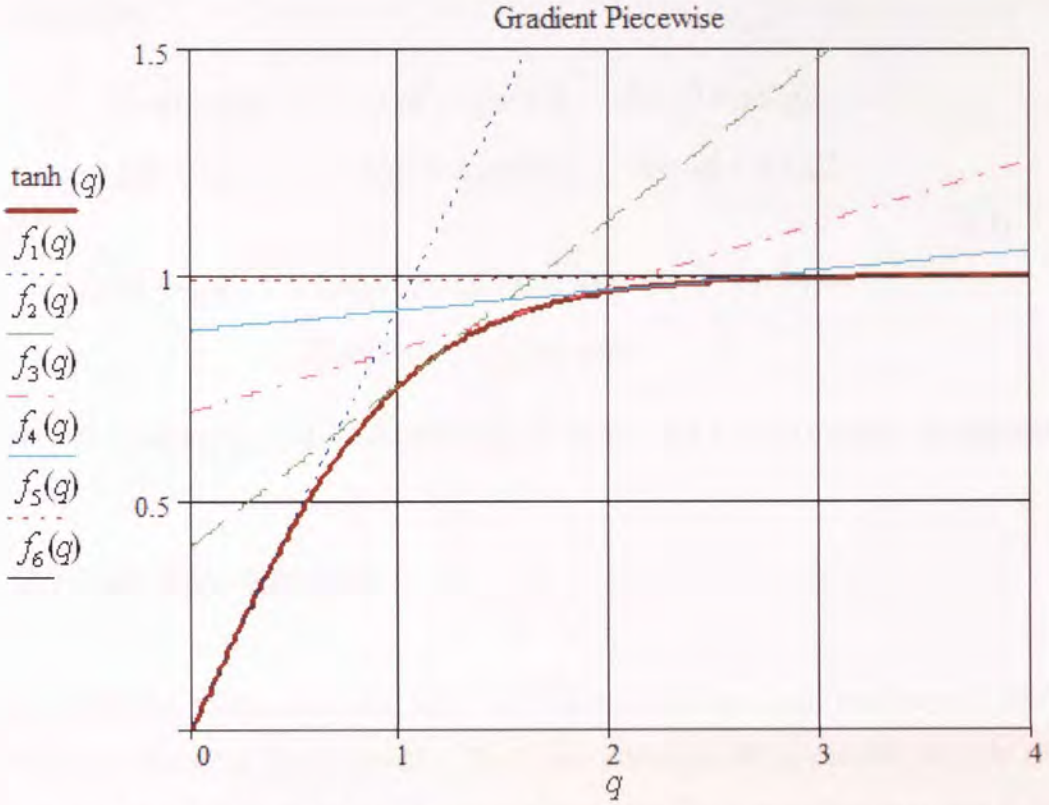


Figure 4.4: Fragments of Gradient approximation to tanh (graph obtained using Mathcad).

4.3.2.2 Polynomial Scheme Approximation

This scheme is based on the curve fitting method. It is similar to the Gradient scheme but it approximates the tanh within the selected ranges by a series of polynomial equations. By taking advantage of the odd symmetry, the accumulated piecewise series of functions as in equation (4.4) enable the scheme to approximate the targeted function locally in the region of the positive q range. One advantage of piecewise evaluation of tanh(q) for specific ranges of q is the significant lower order required to evaluate the function. This is mainly because the curve fitting experiences less nonlinear behaviour due to the splitting of ranges. Hence the approach offers less tedious arithmetic computations. This is important in digital design where extensive latency and resources are major issues. The

higher number of orders used for each of the piecewise polynomials offers a better approximation.

$$f(q) \cong \begin{cases} f_1(q) = a_{1p}q^p + \cdots + a_{12}q^2 + a_{11}q + a_{10} & for \;\; 0 < q \le q1 \\ f_2(q) = a_{2p}q^p + \cdots + a_{22}q^2 + a_{21}q + a_{20} & for \;\; q1 < q \le q2 \\ \vdots & \\ f_n(q) = a_{np}q^p + \cdots + a_{n2}q^2 + a_{n1}q + a_{n0} & for \;\; qn-1 < q \le qsat \\ f_{n+1}(q) = 1 & for \;\; q > qsat \end{cases} \tag{4.4}$$

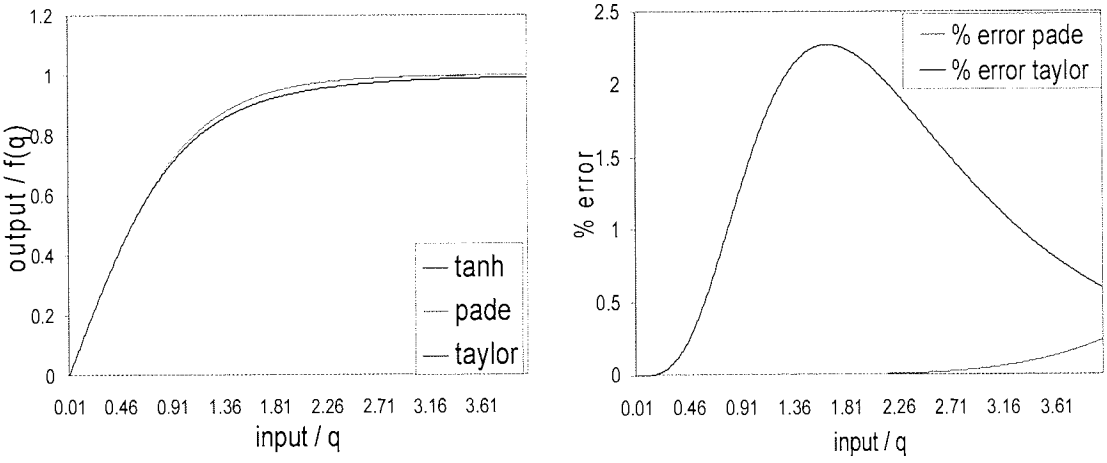
Where p is integer (e.g. 1, 2, ...,3) denoting the order, and n is the number of segment.

4.3.2.3 Padé Approximation

As an alternative to the piecewise approach, the research also explored ways of devising continuous method for approximation. Padé approximation for hyperbolic tangent (Baker and Graves, 1981) is adopted into the current research. The truncated expression is shown by equation (4.6) which uses fourth order for the numerator and the denominator [4,4] (Agnon, Madsen, & Scheffer, 1999). Equation (4.6) is simulated and compared to equation (4.2) incorporating fourth order Taylor series expansion to approximate the exponential e^{2q} (Avci and Yildirim, 2003) as shown by equation (4.5). The graphs are shown in Figure 4.5(a). The performances are also compared to a hyperbolic tangent tanh(q) and the percentage errors are shown in Figure 4.5(b). It was revealed that approximation incorporating Taylor series expansion has shown higher error of up to 2.3 %. The Padé is better than 0.25 % error. Obtaining high accuracy of approximation is crucial in this process.

$$e^{2q} = 1 + 2q + \frac{4q^2}{2!} + \frac{8q^3}{3!} + \frac{16q^4}{4!} \tag{4.5}$$

$$pade(q) = \frac{\left[1 + \frac{1}{9}q^2 + \frac{1}{945}q^4\right]}{1 + \frac{4}{9}q^2 + \frac{1}{63}q^4} q \qquad (4.6)$$



(a) (b)

Figure 4.5: (a) Comparison between hyperbolic approximation (equation 4.2) using Taylor series expansion, Padé approximation and hyperbolic tangent denoted in the figure by taylor, pade and tanh respectively. (b) Percentage error comparison between approximation (equation 4.2) using Taylor series expansion, Padé approximation denoted in the figure by % error taylor and % error pade respectively.

Second important factor, the Padé approximation offers fewer mathematical operations than Taylor's series expansion, offering a significant advantage for digital design. It was therefore decided to use the Padé approximation for continuous approximation over the Taylor. However, to overcome the problem of synthesising operations, such as division, which was awkward to implementation hardware, equation (4.6) was converted into a regression form, equation (4.7).

$$t_{m+1} = 1 + aq^2 + bq^4 - t_m(cq^2 + dq^4) \qquad (4.7)$$

Where:

$a = 1/9$, $b = 1/945$, $c = 4/9$ and $d = 1/63$.

Note that to simplify equation (4.6), the common zero q from the numerator was excluded from the regression equation (4.7), but was computed later to retrieve the final

result. Equation (4.7) was simulated using Matlab and Simulink. Analysis of the results, (Figure 4.6) shows that the regression from the Padé approximation was stable only in the absolute range $0 < |q| < |q^{pade}|$, where the maximum stability range (denoted by q^{pade}) is 1.4.

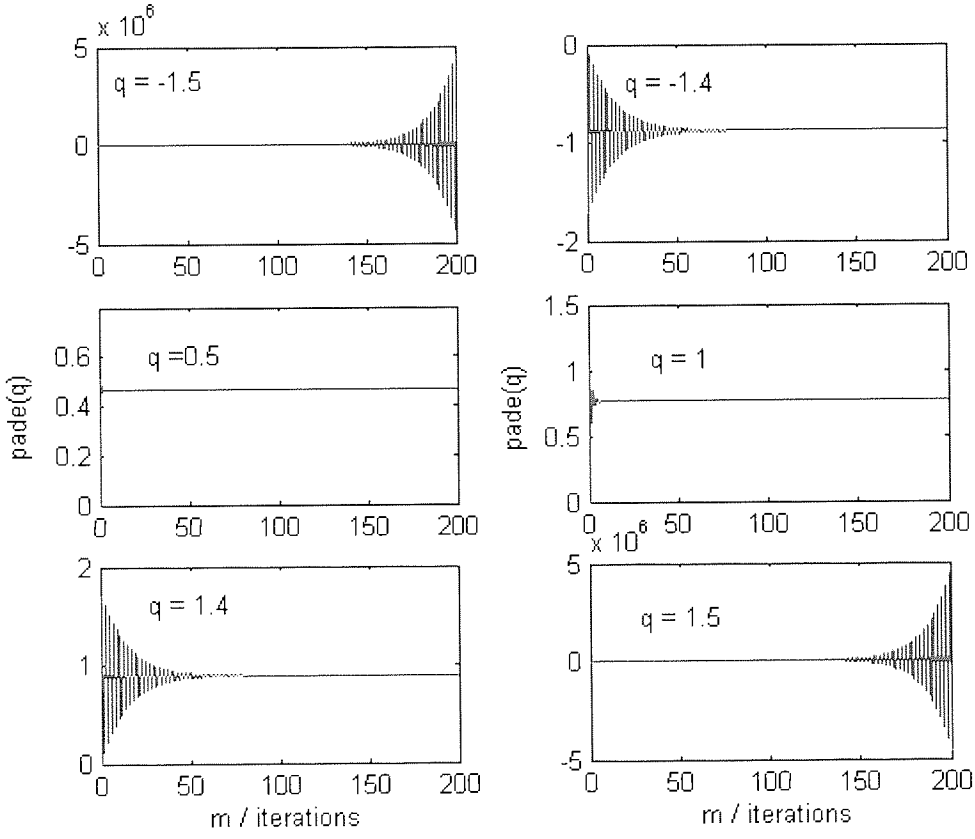


Figure 4.6: Regression Padé approximation for various inputs q

4.3.2.3.1 Solution to Instability

A. Inversing method

Investigation showed that a radical but effective method of improving the stability range of the regression Padé (4.7) is to compute the inverse of equation (4.7), as shown in equation (4.8). The outcome of the analysis can be observed in Figure (4.7), which shows the simulation of the regressive inverse Padé, equation (4.8) with different input q. It can be observed that (4.8) was stable over an increased range of $0 < |q| < 2.8$.

$$h_{m+1} = 1 + cq^2 + dq^4 - h_m(aq^2 + bq^4) \qquad (4.8)$$

It follows that Equation (4.8) is more robust and covers more range than (4.7). The only disadvantage of computing the inverse of the Padé approximation is the need to invert the result, which is again, achieved using the regression formula:

$$inv_{m+1} = 1 - inv_m(z-1) \qquad (4.9)$$

For inputs over $|2.8|$ the value of tanh starts to saturate towards '1'. Therefore the computation was simplified by introducing an 'if then else' condition into the Simulink model to set the output to '1' if the input is more than 2.8 or '-1' if the input is less than -2.8.

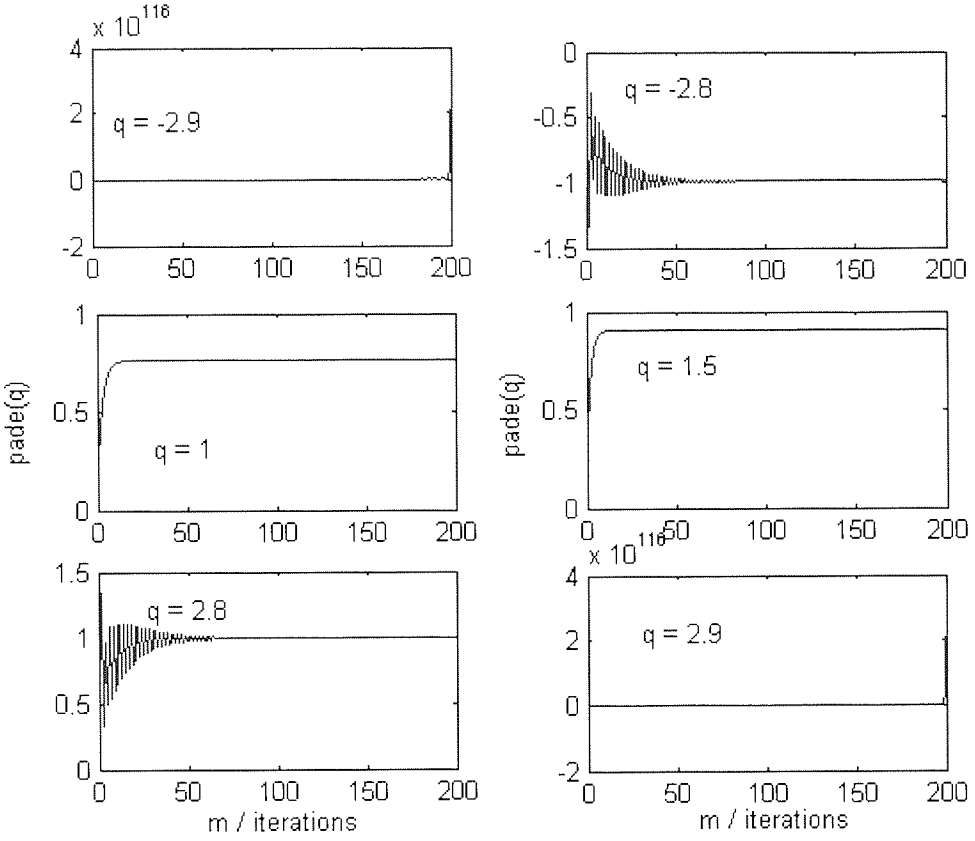


Figure 4.7: The regression inverse Padé approximation for various q inputs.

B. Rho ρ stability factor

Alternatively, an investigation in the current research also discovered that by introducing the convergence rate ρ to the denominator of the Padé equation (4.6), the stability range of the equation can be improved. Also, the numerator and the denominator can be computed separately as in equation (4.10), where only the denominator requires regression (4.11).

$$pade(q) = \frac{A(q)}{B(q)} = A(q)\frac{1}{\rho + B(q) - \rho} \qquad (4.10)$$

Where if

$$d(q) = \frac{1}{\rho + B(q) - \rho} \tag{4.11}$$

The regression to the denominator will be:

$$d_{m+1} = \frac{1}{\rho} - d_m \frac{(B(q) - \rho)}{\rho} \tag{4.12}$$

By setting the convergence rate ρ to 4, stability was improved to $0<|q|<3.3$ as shown in Figure 4.8.

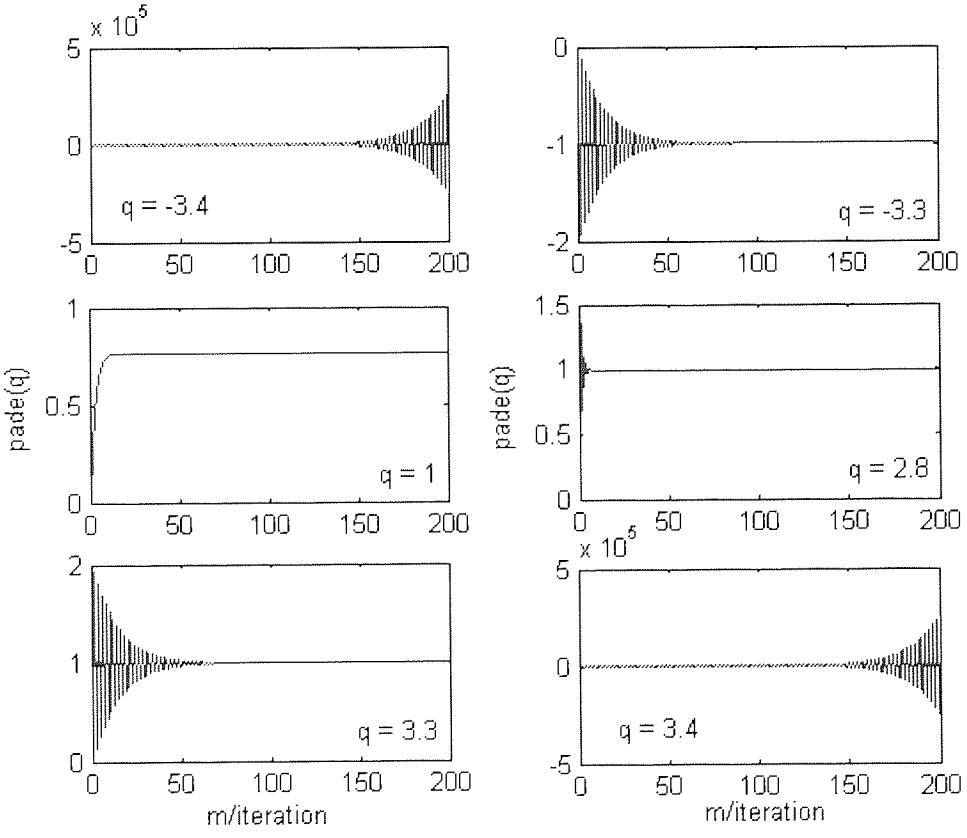


Figure 4.8: Padé Regression using $\rho = 4$.

It follows that by increasing the value of ρ the maximum stability range of the Padé approximation also increased. The only disadvantage of using the convergence rate

method is that by introducing higher values of ρ there was sluggish convergence with smaller input. This is shown in Figure 4.9 for increasing ρ of 1.0, 2.0, 3.0 and input q.

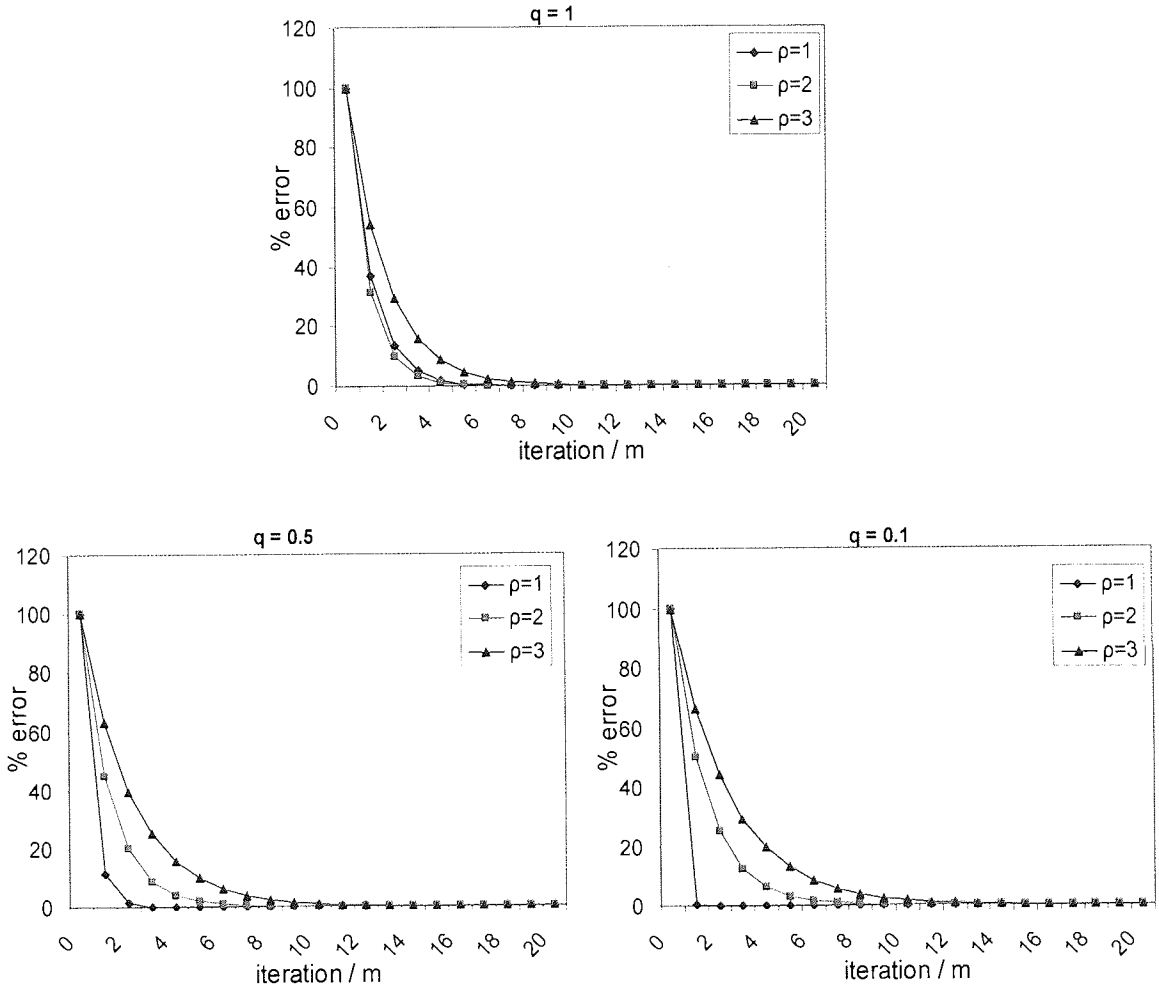


Figure 4.9: Padé responses with ρ =1.0, 2.0 and 3.0 for various sizes of for input q.

4.4 Simulation of the Activation Functions

Equations (4.3), (4.4) and (4.10) were modelled and simulated with Simulink Matlab to maintain the concurrent behaviour of the algorithms, using the Matlab real number system to maintain accuracy. The objective of the simulation is to provide benchmark studies, thus providing an intermediary concurrent implementation picture between real

number modelling (conducted here) and bit true cycle modelling (in later sections) as provided by the Xilinx Blockset and the Xilinx system generator, hence assisting the implementation of the algorithms.

Figure 4.10 shows the designed model of the Gradient scheme approximation with nine segments. The gradients g and intercepts i^g are shown in Table 4.1. The design is mainly comprised of nine subsystems f on which each performs the single piecewise computation of equation (4.3). An ABS and a Threshold Blockset is also introduced in the design to optimise the computation when handling the approximation for the case when input is negative. The ABS function was used to remove any negative sign carried with the input (making it absolute), before being further computed into each of the designs, whereas the Threshold was used to retrieve the sign of the input (such as '-1' if the input was negative and '+1' if positive). The obtained sign was then multiplied by the output from the scheme's core-process to retrieve the actual sign nature of output.

Figure 4.11 shows the designed model of the Polynomial scheme using two piecewise polynomials for the approximation in the q range of -3.5 and 3.5. The equations were derived using Matlab Polyfit toolbox. Table 4.2 tabulates the values of the coefficients and constants needed to form the equations described. Similar to the Gradient, these equations are computed by the two f subsystems of the design. Again, ABS and Threshold units were introduced to allow the Polynomial functions to be defined for positive input only, thus simplifying the design.

Figure 4.12 shows modelling of the Padé approximation algorithm, implementing the ρ method for increasing stability range. Here ρ is 4, sufficient to achieve stability approximation within the q range of $|3.3|$ (sufficiently to exhibit saturation of an output '1'). For the Padé, however, the ABS and Threshold technique is inferior but may be useful during digital implementation stage (discussed in 4.5.2.3.3). Moreover, the cost of the implementation of this technique is not significant.

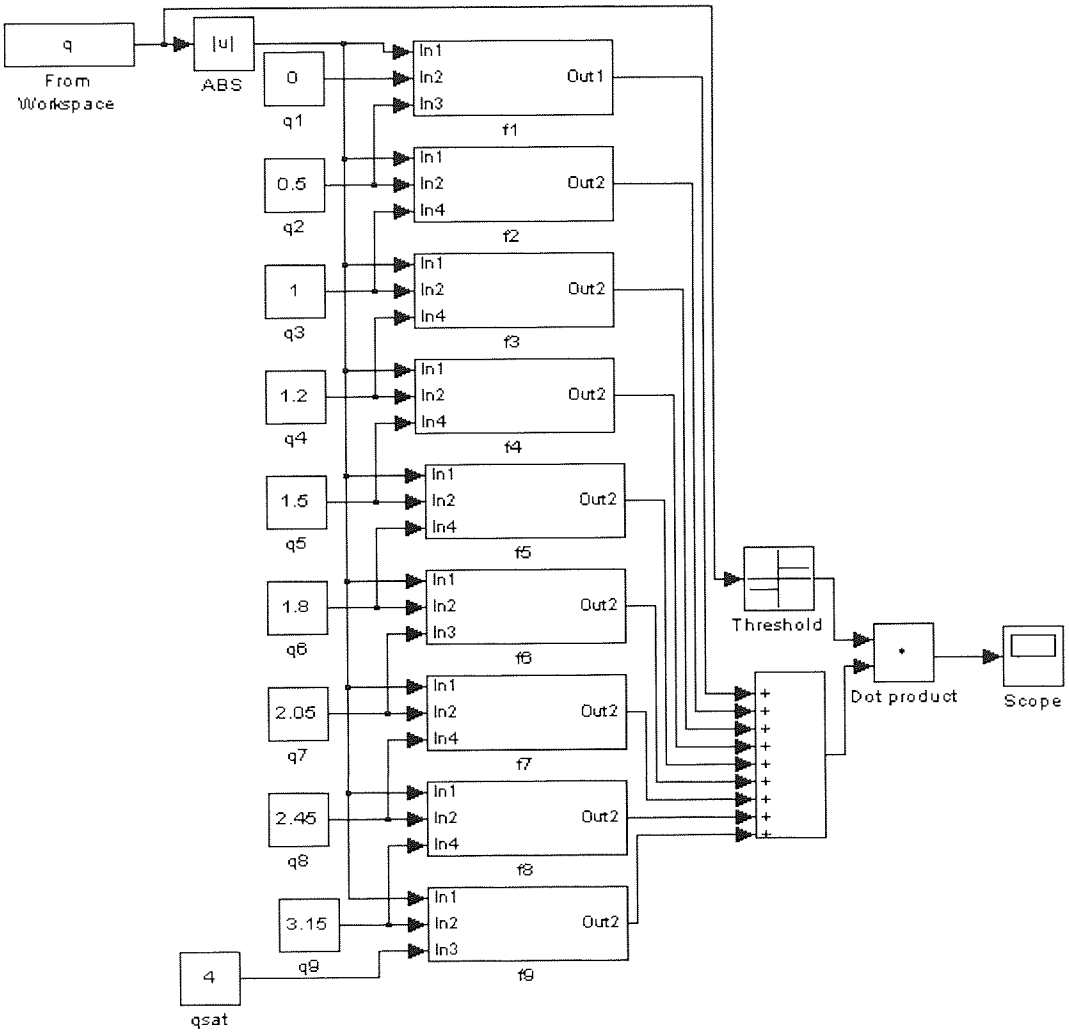


Figure 4.10: The gradient scheme design with Simulink MATLAB.

	Range / q_n	g_n	i_n^g
	0	-	-
1	0.5	0.93046	0.00767
2	1	0.59805	0.1793
3	1.2	0.35986	0.4039
4	1.5	0.23741	0.552
5	1.8	0.1381	0.7001
6	2.05	0.082	0.8001
7	2.45	0.0441	0.8783
8	3.15	0.0154	0.9498
9	4.0	0.0034	0.9862

Table 4.1: Gradient and i-intercept constants for the nine segments gradient scheme.

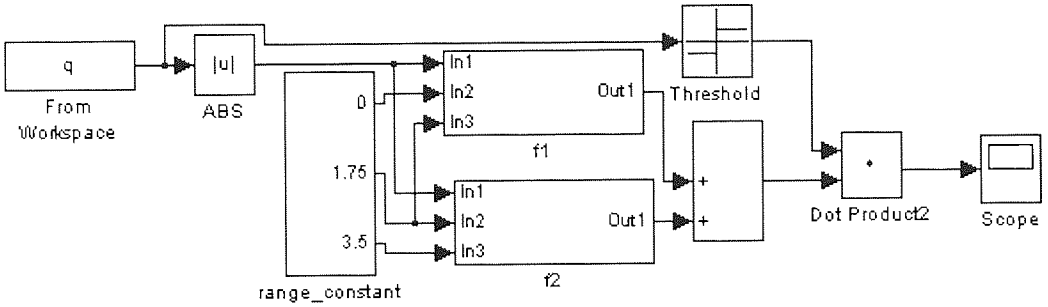


Figure 4.11: The polynomial scheme design with Simulink MATLAB.

	Range / q_n	a_{n2}	a_{n1}	a_{n0}
/ n	0	-	-	-
POLY1	1.75	-0.31247	1.07571	-0.00139
POLY2	3.50	-0.02557	0.16192	0.74118

Table 4.2: Coefficients of the 2nd order polynomial algorithm.

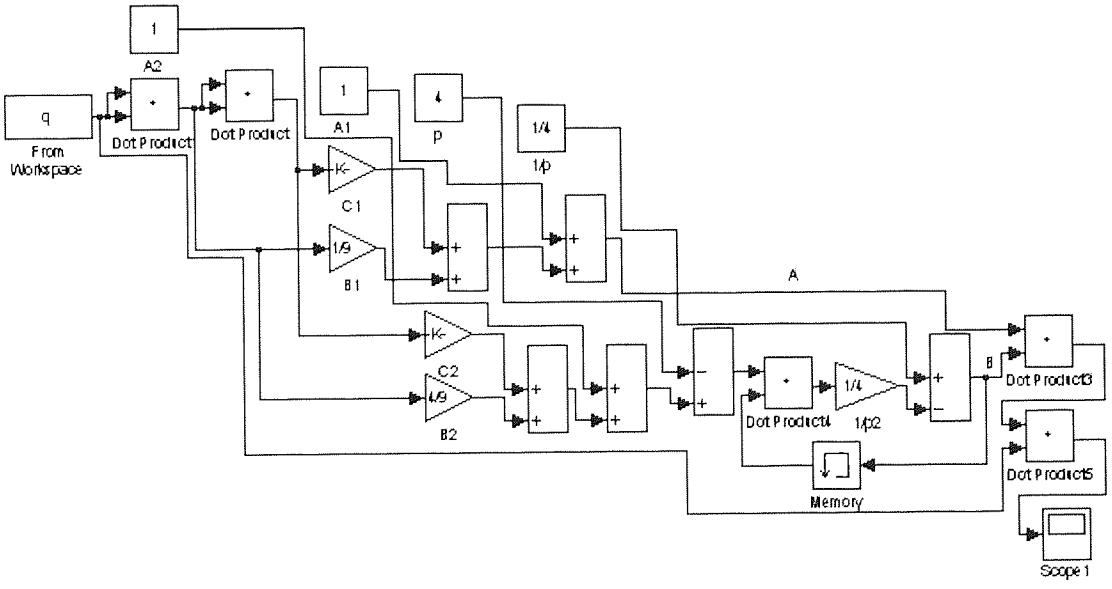


Figure 4.12: The Padé design with Simulink MATLAB.

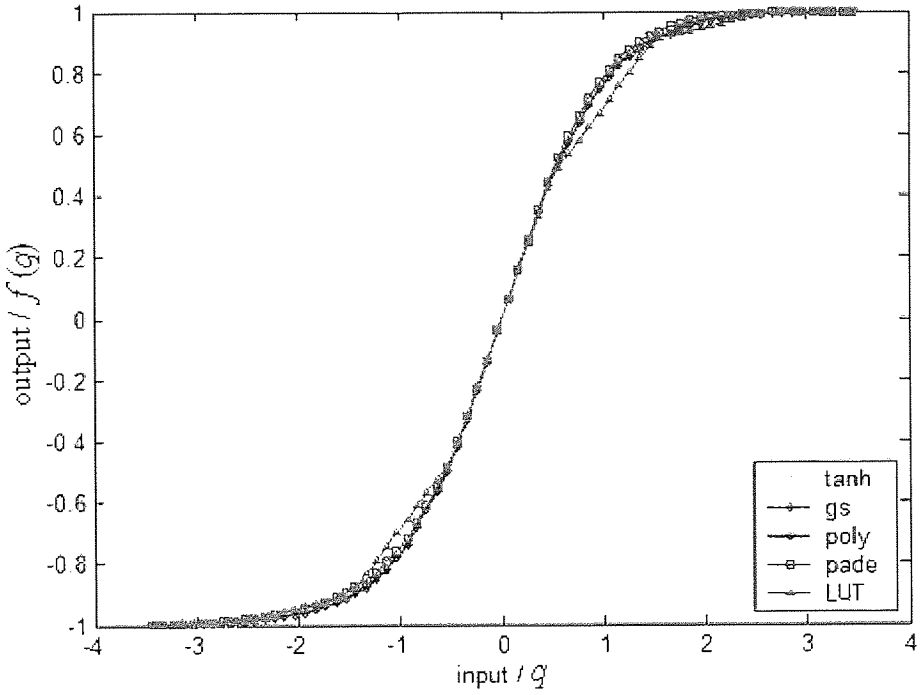


Figure 4.13: Comparison between the Gradient and Polynomial schemes, Padé approximation and LUT approach (denoted in the figure by gs, poly, pade and LUT respectively) and tanh implemented using Simulink MATLAB.

The relevant performances are shown in Figure 4.13. This figure also shows the performance of the Look-Up Table approximating the hyperbolic tangent: this was implemented using Look-Up Table Blockset of Simulink. These are compared to the actual tanh. The results demonstrate the superiority of the devised techniques (the Padé, the Gradient scheme and the Polynomial scheme) in comparison to the Look-Up Table approach. Approximating as close as possible to the real tanh is crucial in this research as the training of the neural network will be off-line, that is using explicit number system. The tanh used in the training will be the actual function of the tanh itself.

4.5 Experimental Activation Function Design

To ensure absolute consistency between the algorithms and the target FPGA implementation, the algorithms were then modelled using design and simulation tools that are compatible with hardware synthesis and modern design flows. Specifically, the designs were created using the Xilinx (Matlab/Simulink) DSP Toolbox and the Matlab fixed-point Blockset, and were simulated using Matlab Simulink running a bit true cycle simulation. The algorithms were designed to take advantage of the inherent concurrency in the algorithm's dataflow to enhance performance and minimise problems due to latency. Normalisation was applied to the linear parts of the algorithms and wordlength constraints were applied to all parts of the operations. The procedure was complied with the normalisation of the feed-forward neural network topology.

4.5.1 Normalisation of Neurons and Synapses for Digital Operation

Optimisation of the digital operations of the neural network is done by normalising all the variables (signals) in the design by adjusting the trained parameters. Inherently, normalising the trained parameters will impose the synapse mathematical operations, which involve using digital multiplication and addition, to produce outputs with numbers of not more than the normalised values of '1'. The wordlength of all of the normalised variables can be sufficiently set to be 14 bits wide with 13 bits for fractional accuracy, as in practice, though the Xilinx multiplier and adder is able to process numbers of up to 64 bits accuracies (Appendix 4), the ADC and DAC are constraint to 14 bits signed input output (1 bit for the signed and 13 bits for the fractional). Having higher than 14 bits for the multiplier and adder is unnecessary and would have only consumed higher pipelines for design.

The input and the output integral takes inputs in the range of *(-1 < x_k <1)* from the Analogue to Digital interface. The output $f(q_j)$ from the activation function will be in the range of $-1 \le f(q_j) \le 1$.

Recall the notations in equation (3.10), normalisation of the signals were done by dividing signal input x_k by the normalising factor X_k, q_j by the normalising factor Q_j, r_j by the normalising factor R_j and y_i by the normalising factor Y_i, such that shown in equation (4.13).

$$\bar{x}_k = \frac{x_k}{X_k};\ \bar{q}_j = \frac{q_j}{Q_j};\ \bar{r}_j = \frac{r_j}{R_j};\ \bar{y}_i = \frac{y_i}{Y_i} \qquad (4.13)$$

Where $\bar{x}_k$ denotes normalised x_k, $\bar{q}_j$ denotes normalised q_j, $\bar{r}_j$ denotes normalised r_j, $\bar{y}_i$ denotes normalised y_i. Substitute (4.13) into equation (3.11) to yield generalised normalised expression of the neural network;

$$\bar{y}_i = \sum_{j=1}^{M} \frac{\omega_{ij}^{(2)}}{Y_i} R_j \bar{r}_j + \frac{b_i^{(2)}}{Y_i} \qquad (4.14(a))$$

$$\text{where, } \bar{r}_j = \frac{f(\bar{q}_j Q_j)}{R_j} \qquad (4.14(b))$$

$$\text{and } \bar{q}_j = \sum_{k=1}^{N} \frac{\omega_{jk}^{(1)}}{Q_j} X_k \bar{x}_k + \frac{b_j^{(1)}}{Q_j} \qquad (4.14(c))$$

In matrix form the overall equation (4.14) can be represented into expressions as follow;

From (4.14(a)) into (4.15(a)) as shown below;

$$\begin{bmatrix} \bar{q}_1 \\ \bar{q}_2 \\ \vdots \\ \bar{q}_M \end{bmatrix} = \begin{bmatrix} \frac{\omega_{1,1}^{(1)}}{Q_1} X_1 & \frac{\omega_{1,2}^{(1)}}{Q_1} X_2 & \cdots & \frac{\omega_{1,N}^{(1)}}{Q_1} X_N \\ \frac{\omega_{2,1}^{(1)}}{Q_2} X_1 & \frac{\omega_{2,2}^{(1)}}{Q_2} X_2 & \cdots & \frac{\omega_{2,N}^{(1)}}{Q_2} X_N \\ \vdots & \vdots & \frac{\omega_{j,k}^{(2)}}{Q_j} R_j & \vdots \\ \frac{\omega_{M,1}^{(1)}}{Q_M} X_1 & \frac{\omega_{M,2}^{(1)}}{Q_M} X_2 & \cdots & \frac{\omega_{M,N}^{(1)}}{Q_M} X_N \end{bmatrix} \begin{bmatrix} \bar{x}_1 \\ \bar{x}_2 \\ \vdots \\ \bar{x}_N \end{bmatrix} + \begin{bmatrix} \frac{b_1^{(1)}}{Q_1} \\ \frac{b_2^{(1)}}{Q_2} \\ \vdots \\ \frac{b_M^{(1)}}{Q_M} \end{bmatrix} \qquad (4.15(a))$$

From (4.14(b)) into (4.15(b)) as shown below;

$$\begin{bmatrix} \bar{r}_1 \\ \bar{r}_2 \\ \vdots \\ \bar{r}_M \end{bmatrix} = \begin{bmatrix} \frac{1}{R_1} f(Q_1, \bar{q}_1) \\ \frac{1}{R_2} f(Q_2, \bar{q}_2) \\ \vdots \\ \frac{1}{R_M} f(Q_M, \bar{q}_M) \end{bmatrix} \qquad (4.15(b))$$

From (4.14(c)) into (4.15(c)) as shown below;

$$\begin{bmatrix} \bar{y}_1 \\ \bar{y}_2 \\ \vdots \\ \bar{y}_L \end{bmatrix} = \begin{bmatrix} \frac{\omega_{1,1}^{(2)}}{Y_1} R_1 & \frac{\omega_{1,2}^{(2)}}{Y_1} R_2 & \cdots & \frac{\omega_{1,M}^{(2)}}{Y_1} R_M \\ \frac{\omega_{2,1}^{(2)}}{Y_2} R_1 & \frac{\omega_{2,2}^{(2)}}{Y_2} R_2 & \cdots & \frac{\omega_{2,M}^{(2)}}{Y_2} R_M \\ \vdots & \vdots & \frac{\omega_{i,j}^{(2)}}{Y_i} R_i & \vdots \\ \frac{\omega_{L,1}^{(2)}}{Y_L} R_1 & \frac{\omega_{L,2}^{(2)}}{Y_L} R_2 & \cdots & \frac{\omega_{L,M}^{(2)}}{Y_L} R_M \end{bmatrix} \begin{bmatrix} \bar{r}_1 \\ \bar{r}_2 \\ \vdots \\ \bar{r}_M \end{bmatrix} + \begin{bmatrix} \frac{b_1^{(2)}}{Y_1} \\ \frac{b_2^{(2)}}{Y_2} \\ \vdots \\ \frac{b_L^{(2)}}{Y_L} \end{bmatrix} \qquad (4.15(c))$$

But it was noted that from the conditions of the ADC and the activation function, the signals are limited to maximum of '1' implying $X_k = R_j = 1$, and thus $\bar{x}_k = x_k$ and $\bar{r}_j = r_j$. But for Q_j and Y_j, the single vectors need to be computed using the expressions below;

$$Q_j = (\sum_{k=1}^{N} ceil\left|\omega_{j,k}^{(1)}\right|) + ceil\left|b_j^{(1)}\right| \tag{4.16}$$

$$Y_i = (\sum_{j=1}^{M} ceil\left|\omega_{i,j}^{(2)}\right|) + ceil\left|b_i^{(2)}\right| \tag{4.17}$$

Where '*ceil*' is a function used to rounds the elements to the nearest integers towards infinity (terms used to describe the function is commonly used in mathematical program such as Matlab, see Appendix 5 for Matlab program).

Thus equation (4.15(a)), (4.15(b)) and (4.15(c)) can be simplified into;

$$\bar{y}_i = \sum_{j=1}^{M} \frac{\omega_{ij}^{(2)}}{Y_i} r_j + \frac{b_i^{(2)}}{Y_i} \text{ where, } r_j = f(\bar{q}_j Q_j) \text{ and } \bar{q}_j = \sum_{k=1}^{N} \frac{\omega_{jk}^{(1)}}{Q_j} x_k + \frac{b_j^{(1)}}{Q_j} \tag{4.18}$$

And the general normalised expression of the FFMLP can then be rewritten into;

$$y_i = \sum_{j=1}^{M} (W_{ij}^{(2)} (f \sum_{k=1}^{N} (W_{jk}^{(1)} x_k + B_j^{(1)}) Q_j) + B_i^{(2)}) Y_i \tag{4.19}$$

Where

$$W_{jk}^{(1)} = \frac{\omega_{jk}^{(1)}}{Q_j}; \quad B_j^{(1)} = \frac{b_j^{(1)}}{Q_j}; \quad W_{ij}^{(2)} = \frac{\omega_{ij}^{(2)}}{Y_i}; \quad B_i^{(2)} = \frac{b_i^{(2)}}{Y_i} \tag{4.20}$$

4.5.2 Development of Digital Activation Function

4.5.2.1 Architectural Overview

All of the activation function designs consist of three major system operations, namely the ABS system, the Main activation process system and the Sign retriever system (see Figure 4.14). All can accept and process normalised signed signals, having 14 bits wide with 13 bits for fractional accuracies. The Main activation process system stores the algorithm required for each scheme and is responsible for mathematical computation of the signal. It is also where the de-normalisation of the input to the activation function is performed.

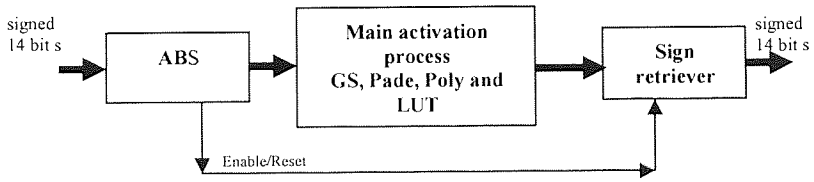


Figure 4.14: The proposed architectural flow design for the activation function.

4.5.2.2 ABS System

The ABS system is designed to extract the 'sign nature' of the input signal and convert it into its absolute magnitude. The function mimics the dual operations provided by the ABS and Threshold Blockset in the Simulink Matlab design from Section 4.4. An ABS function can be computed by multiplying the input values by its own sign obtained from the threshold function (which generates a Boolean output). In this design, however, a preferable technique was suggested, thus minimising the higher usage of the digital hardware multiplier. This was done by using a simple technique following the signed

two's compliment number operations, as will be explained in this section. Only a single Xilinx Relation Blockset, two Additions, five Inverters and two Registers were used (see Figure 4.15). Importantly no multiplication is needed.

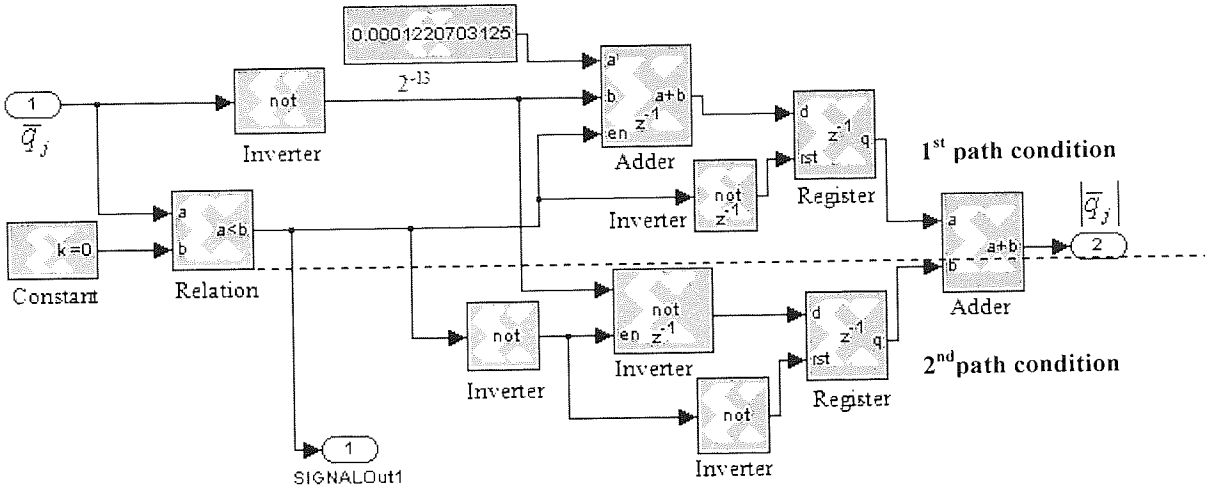


Figure 4.15: The ABS system.

Figure 4.15 shows the configuration of the ABS system design in the Xilinx Matlab environment. When the system receives a normalised signal $\overline{q}_j$, its polarity is tested by the 'less than zero' Blockset. If the input number is negative, the blockset will give a Boolean output of '1', if otherwise the output will be '0'. The Boolean output is useful in two ways. First, for enabling and resetting purposes in the following operations. Second it provides a feeding-out to the Sign retriever system as a control signal to retrieve the true sign nature of the signal after it has undergone the main activation process operation.

Technically, if $\overline{q}_j < 0$, the 14 bits signed binary number of input $\overline{q}_j$ will be inverted and then added to the 14 binary bits with constant value of 2^{-13}(e.g. .0000000000001 binary). This will result removal of negative sign if it was carried by $\overline{q}_j$. But if $\overline{q}_j \geq 0$, the operation will activate the second path of the process, whereby the inverted binary of $\overline{q}_j$ will be re-inverted. There will be no change to the output since the signal will undergo the inversion process twice. The operations of the second Inverter Blockset of the '2nd

path condition' and the Add Blockset of the '1st path condition' are controlled by the enable port which takes the conditional signal (see Figure 4.15). Registers connected in series with the two paths are used to reset the subsystem by a control from the reset port. The operation is completed by summing both outputs using the digital adder. In the experiment, Constant Blocksets and Xilinx Adder Blocksets all have an accuracy of a signed 14 bits wide with 13 bits for fractional point.

4.5.2.3 Main Activation Process System

4.5.2.3.1 The Gradient Scheme

The absolute normalised signal $|\bar{q}_j|$ from the ABS system has a relationship with its normalised factor, Q_j and the actual absolute magnitude $|q_j|$, as expressed in equation (4.21).

$$q_j = \bar{q}_j Q_j \tag{4.21}$$

Note that in the following sections the modulus signs are omitted from the equation for simplicity of explaining derivations.

A. Normalization of Gradient scheme

Normalisation of the Gradient scheme is done by substituting equation (4.21) into (4.3), to yield:

$$f(q_j) \cong \begin{cases} f_1(q_j) = g_1 \bar{q}_j Q_j + i_1^g \quad for \quad 0 < \bar{q}_j Q_j \le q1 \\ f_2(q_j) = g_2 \bar{q}_j Q_j + i_2^g \quad for \quad q1 < \bar{q}_j Q_j \le q2 \\ \vdots \\ f_n(q_j) = g_n \bar{q}_j Q_j + i_n^g \quad for \quad qn-1 < \bar{q}_j Q_j \le qsat \\ f_{n+1}(q_j) = 1 \qquad for \quad \bar{q}_j Q_j > qsat \end{cases} \tag{4.22}$$

Thus producing a new gradient G_n, where $G_n = g_n Q_j$.

$$f(q_j) \cong \begin{cases} f_1(q_j) = G_1 \bar{q}_j + i_i^g \quad for \quad 0 < \bar{q}_j \le {}^{q1}/_{Q_j} \\ f_2(q_j) = G_2 \bar{q}_j + i_2^g \quad for \quad {}^{q1}/_{Q_j} < \bar{q}_j \le {}^{q2}/_{Q_j} \\ \vdots \\ f_n(q_j) = G_n \bar{q}_j + i_n^g \quad for \quad {}^{qn-1}/_{Q_j} < \bar{q}_j \le {}^{qat}/_{Q_j} \\ f_{n+1}(q_j) = 1 \quad for \quad \bar{q}_j > {}^{qsat}/_{Q_j} \end{cases} \tag{4.23}$$

By following the proposed normalised algorithm explained by equation (4.23), no extra effort will be required to de-normalise the computed $\bar{q}_j$. De-normalisation was done automatically and internally when $\bar{q}_j$ was multiplied with the new set gradient G_n, which carried within it the information of the actual normalised factor Q_j. The advantage is that no extra multiplier is require.

B. The Main Activation Process of Gradient

The Gradient Scheme has two main functional parts: the Comparator which comprises 10 comparator sub-units and the Data Processing which has nine Gradient operation sub-units and one Enabled threshold sub-unit (responsible for executing a '1' when the signal $\bar{q}_j$ lies beyond $qsat/Q_j$). Each sub-unit of the Comparator is connected to the corresponding Gradient operation sub-units or Enabled threshold sub-unit. Each Gradient operation sub-unit is dedicated to a unique range with a unique gradient operation coefficient and constant. Table 4.3 shows the relative figures dedicated to their corresponding ranges ($0 < \bar{q}_j \le qsat/Q_j$) and the bits required for the experiment. Here the value of Q_j is 3.

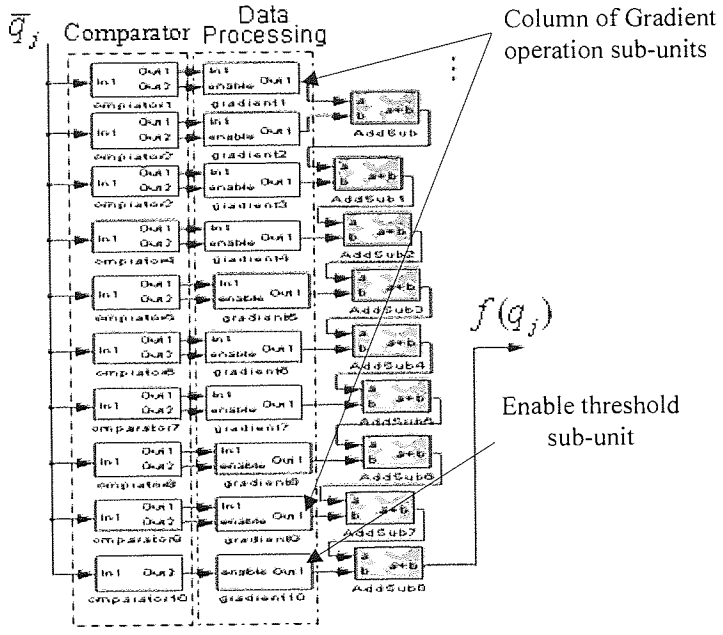


Figure 4.16: The Gradient design for the activation function used for the nine segments operation.

C. The Comparator

Each of the comparator sub-units (comparator1 to comparator9) in Figure 4.16 employed two digital Relation Blocksets for 'greater than' and 'less than and equal to' for arithmetic comparison, and a digital Logical Blockset 'AND' for logical function (see Figure 4.17). comparator10 employed one digital Relation Blockset, and a digital Logical Blockset. The constants shown are needed for specifying range shown in equation (4.23).

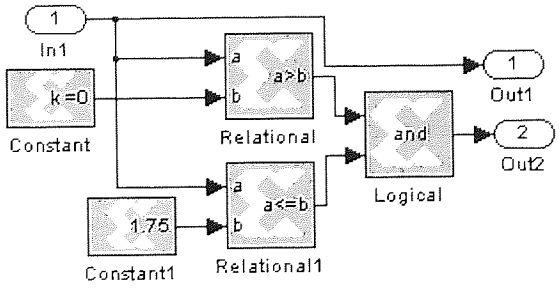


Figure 4. 17: An example of Comparator building blocks.

D. The Gradient Operation Sub-Unit

When signal $\overline{q}_j$ is received by the Main process activation system, all the comparators sub-units will test the magnitude concurrently. The outputs form a column vector of 10 elements. But from that, only one will execute a high state output of Boolean '1' signals. For $\overline{q}_j < qsat/Q_j$ the signal will be used to activate the corresponding gradient Multiplier and the Adder, hence enabling computation with the corresponding coefficient G_n and constant i_n^g (see Figure 4.18).

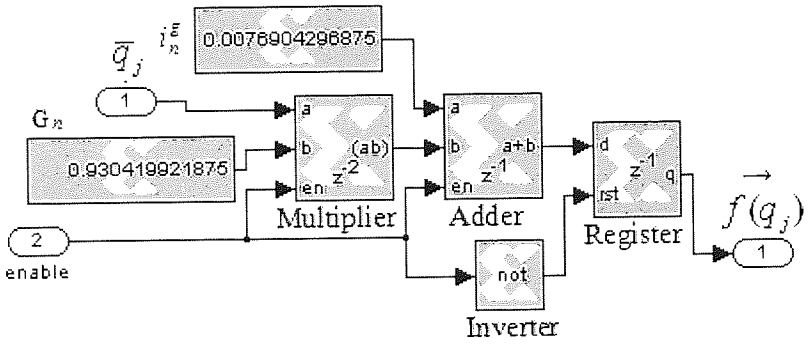


Figure 4. 18: The Gradient operation sub-unit. (Arrow in the output indicates the output still need to undergone next arithmetic process)

A register was used in each of the n^{th} rows to reset the operation. Resetting is important to make sure that no overlapping information of the previous output data and the new output occurred throughout the process.

All of the digital Multipliers and Adders Blocksets were 14 bits wide with 13 bits fractional accuracies in order to provide maximum accuracy for all the normalised values. All the multiplier and adders in each of the rows had an enable port.

	Range/ q1 /Q_j	Accuraccy/bit		G_n/g_{nQj}	Accuracy/bit		i_n^g	Accuracy/bit	
	0		Binary pt	-		Binary pt	-		Binary pt
1	0.167	14	13	2.79	14	11	0.00767	14	13
2	0.333	14	13	1.79	14	12	0.1793	14	13
3	0.4	14	13	1.08	14	12	0.4039	14	13
4	0.5	14	13	0.71	14	13	0.552	14	13
5	0.6	14	13	0.41	14	13	0.7001	14	13
6	0.633	14	12	0.25	14	13	0.8001	14	13
7	0.817	14	13	0.13	14	13	0.8783	14	13
8	1.05	14	12	0.05	14	13	0.9498	14	13
9	1.333	14	12	0.01	14	13	0.9862	14	13

Table 4.3: The constants used for the gradients and the horizontal-axis intercept for the nine segments design (with Q_j =3).

4.5.2.3.2 The Polynomial Scheme

A. Normalisation of the Algorithm

Normalisation of the polynomial equations is necessary to ensure that there are no mathematical operations (multiplication and addition in the Main Activation Process of the polynomial) involving solutions of numbers beyond the normalised values '1'. A technique was designed to normalise the equations globally and give rise to the advantage of using the very less multipliers in retrieving the actual magnitude.

Consider the polynomials as a piecewise equation;

$$f(q_j) = \begin{cases} a_{12}q_j^{\,2} + a_{11}q_j + a_{10} & for \quad 0 \le q_j < q1 \\ a_{22}q_j^{\,2} + a_{21}q_j + a_{20} & for \quad q1 \le q_j < qsat \end{cases} \tag{4.24}$$

But if $q_j = \overline{q}_j Q_j$, and substitutes into q, will yield equations as below;

$$f(q_j) = \begin{cases} a_{12}(\overline{q}_j Q_j)^2 + a_{11}(\overline{q}_j Q_j) + a_{10} & for \quad 0 \le \overline{q}_j Q_j < q1 \\ a_{22}(\overline{q}_j Q_j)^2 + a_{21}(\overline{q}_j Q_j) + a_{20} & for \quad q1 \le \overline{q}_j Q_j < qsat \end{cases} \tag{4.25}$$

Because the coefficient a_{11} is greater than '1' (as presented in Table 4.2), it was used as a normalised coefficient factor for normalising each of the coefficients present.
Dividing all coefficients in the equations by the normalised coefficient factor a_{11} and the squared value of the normalised factor Q_j and rearranging the equations gives:

$$f(q_j)=\begin{cases}\left[\frac{a_{12}}{a_{11}}(\overline{q}_j)^2+\frac{1}{Q_j}(\overline{q}_j)+\frac{a_{10}}{a_{11}Q_j^2}\right]a_{11}Q_j^2 & for \quad 0\le \overline{q}_j < {q1}/{Q_j} \\ \left[\frac{a_{22}}{a_{11}}(\overline{q}_j)^2+\frac{a_{21}}{a_{11}Q_j}(\overline{q}_j)+\frac{a_{20}}{a_{11}Q_j^2}\right]a_{11}Q_j^2 & for \quad {q1}/{Q_j} \le \overline{q}_j < {qsat}/{Q_j}\end{cases} \tag{4.26}$$

Similar to the Gradient, the range of each of the equations also changed for the normalised factor Q_j. Finally equation (4.26) generalised as:

$$\begin{aligned}&\overbrace{[A_{12}\overline{q}_j{}^2+A_{11}\overline{q}_j+A_{10}]}^{poly1\_norm}\hat{Q} \quad for \quad 0\le \overline{q}_j<\sigma_1 \\ &\underbrace{[A_{22}\overline{q}_j{}^2+A_{21}\overline{q}_j+A_{20}]}_{poly2\_norm}\hat{Q} \quad for \quad \sigma_1\le \overline{q}_j<\sigma_{sat}\end{aligned} \tag{4.27}$$

Where

$$\begin{aligned}&\sigma_1={q_1}/{Q_j},\ \sigma_{sat}={q_2}/{Q_j}, \\ &A_{12}={a_{12}}/{a_{11}},\ A_{11}={1}/{Q_j},\ A_{10}={a_{10}}/{a_{11}Q_j{}^2}, \\ &A_{22}={a_{22}}/{a_{11}},\ A_{21}={a_{21}}/{a_{11}Q_j},\ A_{20}={a_{20}}/{a_{11}Q_j{}^2},\end{aligned} \tag{4.28}$$

And the global de-normalisation factor $\hat{Q}$ of the polynomial scheme is $a_{11}Q_j{}^2$.

In the current implementation, the normalised factor Q_j is 3, thus producing new parameters for the polynomial as shown in Table 4.4.

n	Range/ qn / Q_j	Bit		A_{n2}	Bit		A_{n1}	Bit		A_{n0}	Bit		$\hat{Q}_j$	Bit	
	0		Binary pt	-		Binary Pt	-		Binary pt	-		Binary Pt			Binary pt
1	0.583	14	13	-0.2905	14	13	0.333	14	13	-0.0013	14	13	9.681	14	10
2	1.1667	14	12	-0.0238	14	13	0.0502	14	13	0.6890	14	13	9.681	14	10

Table 4.4: The normalised coefficients and normalised arbitrary constant of Polynomial scheme (with Q_j=3)

B. The Main Activation Process of Polynomial

Three comparator sub-units and two polynomial operation sub-units were used in the Comparator and Data Processing of the polynomial scheme (see Figure 4.19). Connected in row to the first comparator was a polynomial operation sub-unit design having the algorithm for *poly*1 _ *norm*, and connected to the second comparator was the sub-unit having the algorithm of *poly*2 _ *norm* (see equation (4.27)). Connected to the third comparator was the Enable threshold sub-unit.

When the Main Activation Process system receives signal $\overline{q}_j$ from the ABS system, the comparators test it and produce an output of single column with a three row vector. If $\overline{q}_j < 3.5/Q_j$, the active high from the Comparator n (where n = 1 or 2) enables the corresponding mathematical operation of the polynomial operation sub-units. The signal $\overline{q}_j$ will be squared by digitally multiplying itself with its own magnitude to produce $\overline{q}_j^{\,2}$. Concurrent to that process is the multiplication enabled control of A_{n1} with signal $\overline{q}_j$ to produce the $A_{n1}\overline{q}_j$ factor. Following this is the multiplication of the $\overline{q}_j^{\,2}$ signal with coefficient A_{n2} to produce the $A_{n2}\overline{q}_j^{\,2}$ factor, and then summing all with the constant to produce $A_{n2}\overline{q}_j^{\,2} + A_{n1}\overline{q}_j + A_{n0}$, and hence complete the activated row mathematical

operation. The enable controlled register is used on each of the rows to reset the operation. Finally, all the (n+1) rows are digitally summed to produce a single normalised output vector, before multiplying with the global de-normalised factor $\hat{Q}_j$ to retrieve the actual signal magnitude.

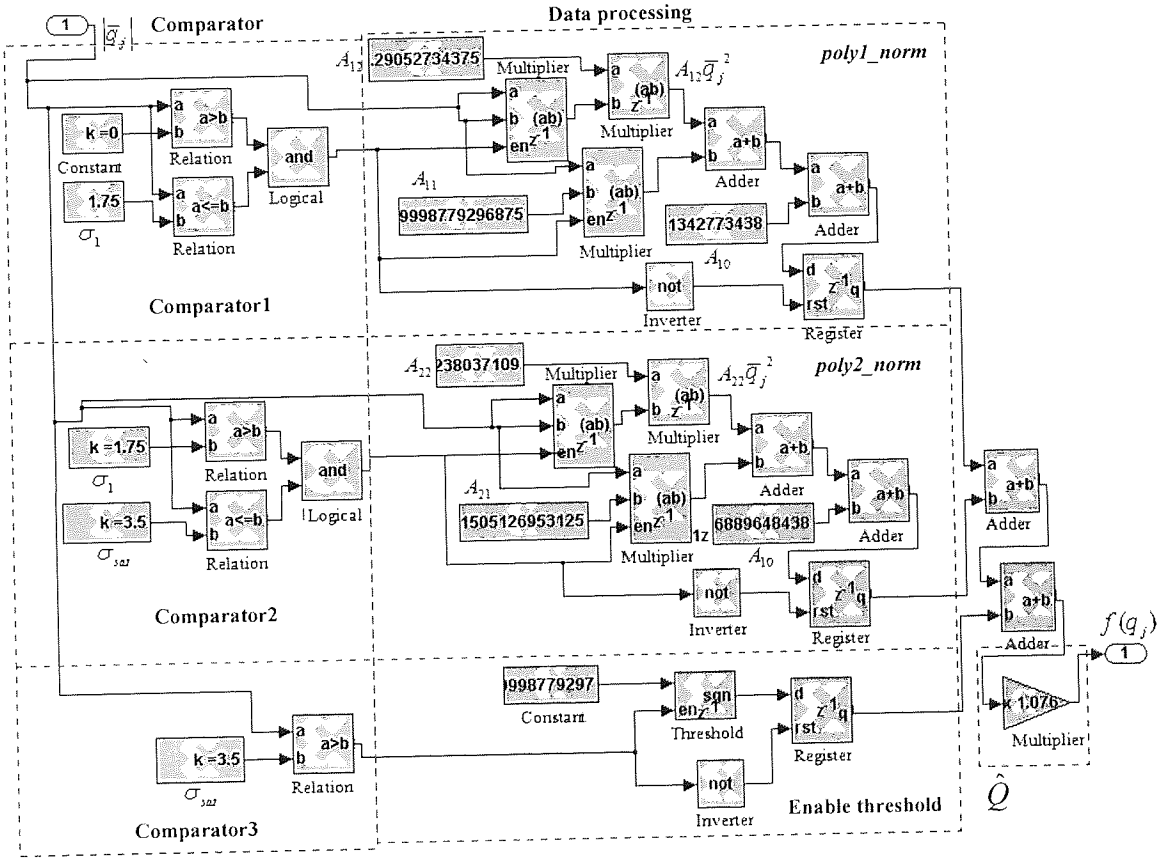


Figure 4.19: The overall view of a Polynomial process design containing the algorithm for the activation function (order two and two segments) - threshold enable controlled.

All of the constants of the parameters and coefficients of the Comparators and the Data processing of polynomial operation have accuracies as depicted in Table 4.4. All the multiplication and addition operators were implemented with 14 bits with 13 fractional bits precision. Having higher than 14 bits for the multiplier and adder is unnecessary (due to the constraint accuracy from the ADC) and would have only consumed higher pipelines for design.

4.5.2.3.3. Padé Approximation

A. Normalisation of Padé Algorithm

Other than ensuring all computation was within the '0 to 1' range, the normalisation of the Padé gives an advantage in terms of understanding what size of bits is required for each of the arithmetic computations in the loop operation involved in the iterative process. By substituting equation (4.21), into equation (4.6) incorporating ρ stability method presented in Section 4.3.2.3.1;

$$pade(\bar{q}_j Q_j) = \frac{1 + a(\bar{q}_j Q_j)^2 + b(\bar{q}_j Q_j)^4}{\rho + 1 + c(\bar{q}_j Q_j)^2 + d(\bar{q}_j Q_j)^4 - \rho}(\bar{q}_j Q_j) \tag{4.29}$$

Dividing all coefficients by Q_j^4 to yield;

$$pade(\bar{q}_j Q_j) = \frac{\frac{1}{Q_j^4} + \frac{a}{Q_j^2}\bar{q}_j^2 + b\bar{q}_j^4}{\frac{\rho}{Q_j^4} + \frac{1}{Q_j^4} + \frac{c}{Q_j^2}\bar{q}_j^2 + d\bar{q}_j^4 - \frac{\rho}{Q_j^4}}(\bar{q}_j Q_j) \tag{4.30}$$

Replacing all the coefficients by $A1$, $B1$, $C1$, $A2$, $B2$ and $C2$ and yields;

$$pade(\bar{q}_j Q_j) = \frac{\overbrace{A1 + B1\bar{q}_j^2 + C1\bar{q}_j^4}^{num}}{\underbrace{\rho' + A2 + B2\bar{q}_j^2 + C2\bar{q}_j^4 - \rho'}_{den}}(\bar{q}_j Q_j) \tag{4.31}$$

Where ρ' is the normalised convergence rate, and together with $A1$, $B1\bar{q}_j^2$, $C1\bar{q}_j^4$, $A2$ $B2\bar{q}_j^2$ and $C2\bar{q}_j^4$ are all < 1. $\bar{q}_j Q_j$ is the global de-normalised factor for the Padé.

B. Padé Main Activation Process system

The Main Activation Process system of the Padé approximation scheme has two Comparator sub-units in the range of $0 < \overline{q}_j \leq q1/Q_j$ and $\overline{q}_j > qsat/Q_j$, where *qsat* is where the maximum range of Padé to give stability (i.e. $qsat = q^{pade} = 3.3$, as depicted in Section 4.4), and two Data Processing sub-units; the Padé operation sub-unit and the enabled threshold sub-unit. The Padé operation sub-unit runs the complete algorithm of the Padé (equation (4.29)). Figure 4.20 shows a core of the building design in the Padé operation sub-unit, uses to perform the iterative operation of the *num* and *den* of equation (4.31) (excluding the $\overline{q}_j Q_j$ component).

Here $\overline{q}_j$ is first squared by multiplying itself with its own magnitude to yield $\overline{q}_j^2$, and further squared to produce $\overline{q}_j^4$. The two multipliers have an accuracy of 14 bits wide for the normalised output. The product of $\overline{q}_j^2$ and $\overline{q}_j^4$ are used for the *num*, (the numerator operation), to directly compute the $A1 + B1\overline{q}_j^{\,2} + C1\overline{q}_j^{\,4}$ and for *den*, (the denominator operation) to directly compute $A2 + B2\overline{q}_j^{\,2} + C_2\overline{q}_j^{\,4}$, which is then used to compute the regression of the Padé (see Figure 4.20). Upon completion of the regression iteration, the output is multiplied with the output from *num* thus completing the $\dfrac{A1 + B1\overline{q}_j^{\,2} + C1\overline{q}_j^{\,4}}{A2 + B2\overline{q}_j^{\,2} + C2\overline{q}_j^{\,4}}$ operations.

All the multiplications and additions use the truncate technique and wrap for the overflow. For the current implementation, the normalised factor Q_j and the stability factor ρ used for the Padé are 3 and 4 respectively. The coefficient values for the normalised coefficients *A1, B1, C1, A2 B2* and *C2* are [0.012 0.012 0.0011] and [10.012 0.049 0.0159] respectively. All were implemented using constant Blocksets of signed output 14 bits wide with 13 fractional bits.

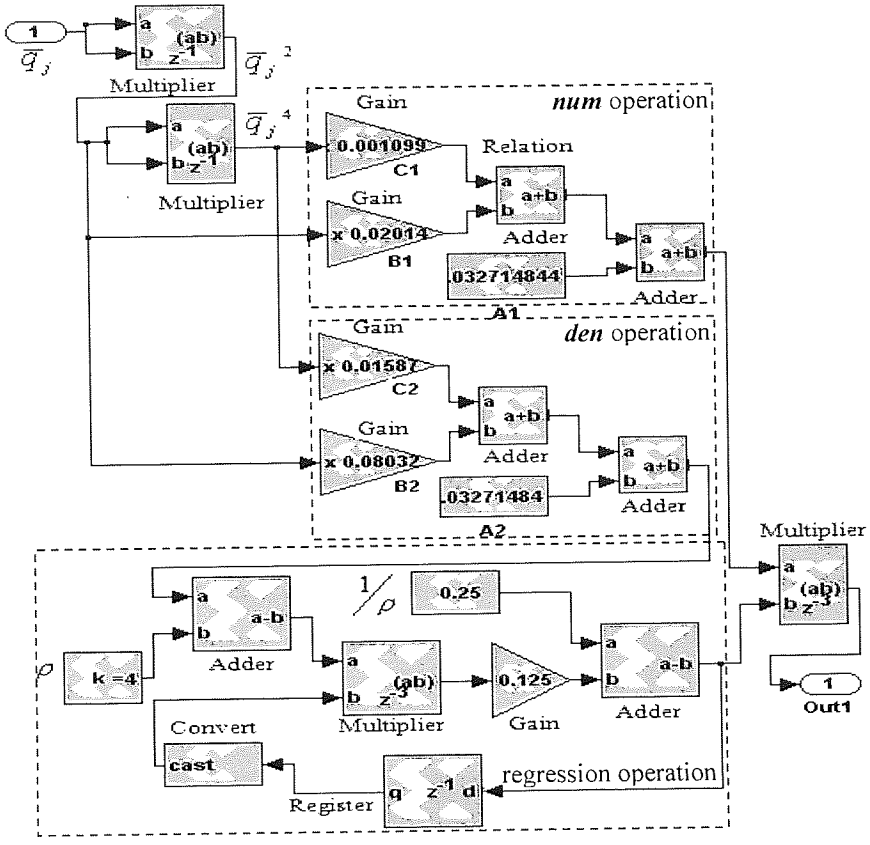


Figure 4.20: The Padé process sub-subsystem design for the activation function.

4.5.2.3.4. Look-Up Table

The look-up table design comprised 128 depth-increments and utilised the ROM in the Xilinx DSP design environment. The output from the ROM was assigned to give signed 14 with 13 binary point bits. Other than the range, the comparator section of the Look-Up Table is similar to the one used for the Padé. For the Look-Up Table, the signal is input to the LUT ROM if the input is between 0 and 3.5. Above these values it forces the design to output a '1'.

4.5.2.4. Sign Retriever Subsystem

This is the system used to retrieve the actual sign of signal $\overline{q}_j$ after computation in the main process subsystem. The procedure is the inverse of the ABS subsystem, see Figure 4.21. If the received enable signal from the ABS subsystem is high, then the Adder's output (the sum of the normalised signal with 14 binary bits and the 14 binary bits of the 2^{-13} constant) will be inverted using an enable controlled Inverter. If the enable signal is '0', the inverted signal will be re-inverted by the enable controlled inverter. There will be no change to the final sign magnitude of the signal in. To reset the two condition processes, registers are assigned in series to each of the channels. The registers have reset ports channelling the one delay latency of the inverted enable of the corresponding enable controlled inverter. The output from the two registers is digitally added to complete the whole operation.

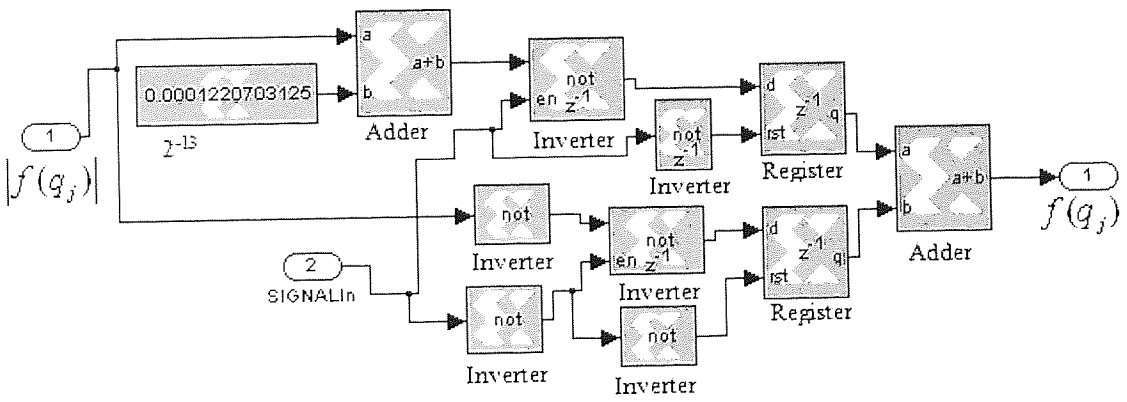


Figure 4.21: The Sign retriever subsystem.

4.6 Results

The designs were synthesised and implemented for a target Xilinx Virtex-II 2XCV3000 FPGA. The FPGA equivalent gate counts for the schemes were as follows: Gradient Scheme approximation 44k, Polynomial scheme approximation 28k, Padé approximation 33k, and Look-Up Table approach 74k.

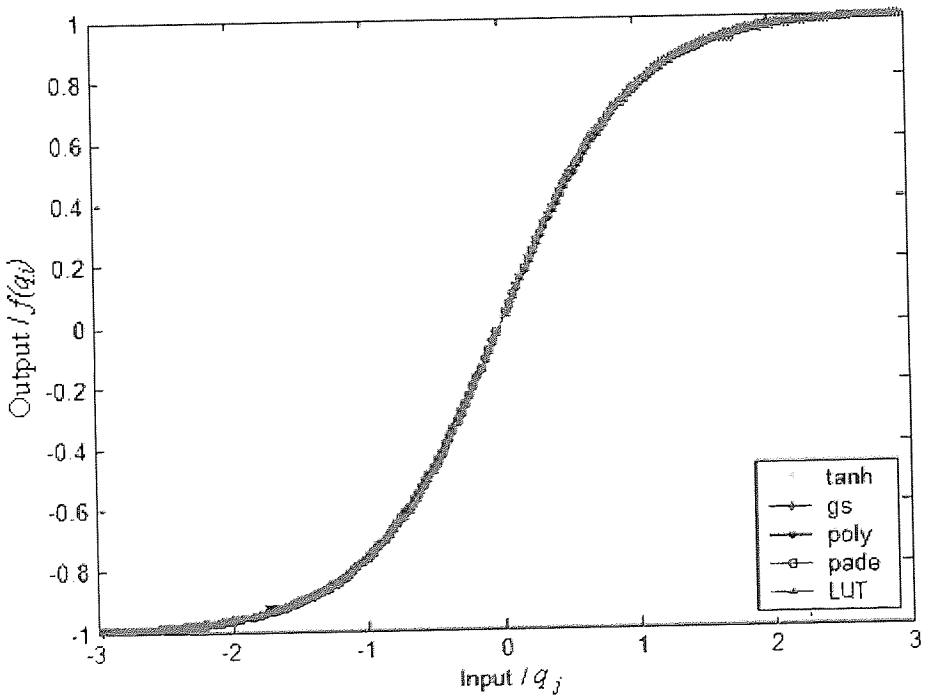


Figure 4. 22: Simulation of Xilinx DSP-based excitation function designs for Gradient and Polynomial schemes, Padé approximation and LUT approach (denoted in the figure by gs, poly, pade and LUT respectively).

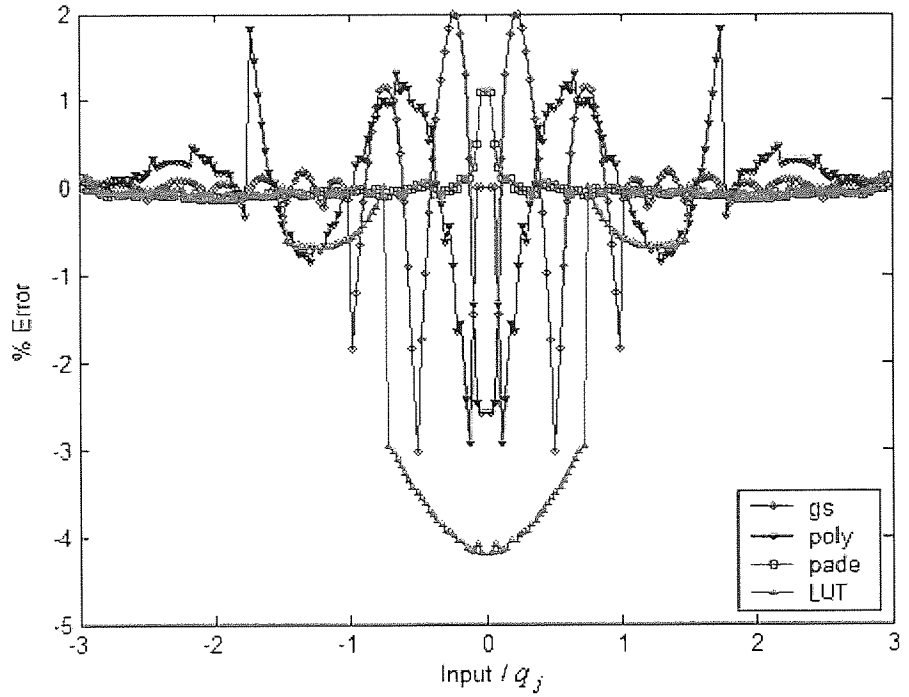


Figure 4.23: Error comparison for the Xilinx DSP-based excitation function designs for Gradient and Polynomial schemes, Padé approximation and LUT approach (denoted in the figure by gs, poly, pade and LUT respectively).

The simulation results for the Xilinx DSP-based excitation function designs for Gradient and Polynomial schemes, Padé approximation and LUT approach are shown in Figure 4.22. An error comparison for these simulated designs is shown in Figure 4.23. The performance differences between the Gradient Scheme approximation, the Polynomial Scheme approximation, the Padé approximation and the Look-Up Table were compared by evaluating the area under the absolute % error along the x-axis.

$$mean\ \%\ error = \frac{1}{6}\int_{x=-3}^{3} |error\ dx| \qquad (4.32)$$

The area of error evaluation exhibited by the Gradient Scheme, the Polynomial Scheme, the Padé approximation and the Look-Up Table is 0.40%, 0.58%, 0.09% and 1.08% respectively, demonstrating the relative superiority of the Padé, the Gradient scheme and the Polynomial scheme.

4.7 Summary

This chapter described the implementation of various activation functions as hardware circuits in an FPGA. The performances were investigated, and it was revealed that the activation functions of the Gradient and the Padé are appropriate to be employed in the neural network implementation. The Gradient and the Padé, which used 44k and 33k equivalent gates respectively, exhibited percentage errors of 0.4 and 0.09 respectively, compared to 0.58 and 1.08 for the Polynomial and Look-Up Table. High accuracy of the activation function is essential in the implementation of the neural network. This is because the training was done offline with an explicit number system. Therefore, the proposed activation function should be as close to the real activation function used in the training, such as the hyperbolic tangent. However, the drawback of a highly accurate activation function such as the Padé is the slower settling time. This is due to the iterative nature of the scheme. To date, this has led to the Gradient scheme receiving more attention in the following research, especially in terms of optimising the resources required to run the scheme. This issue will be discussed further in Chapter 5.

CHAPTER 5

Distributive Tactile Sensing Embedded Solution

5.1 Introduction

This chapter is an implementation chapter, and as such describes the implementation of different kinds of neural networks proposed for the distributive tactile sensing system in Chapter 3 (the Single and the Cascade). This chapter charts various analyses of their effectiveness. This section of the thesis discusses the attempt to design digital architectures which comply with the topologies of the networks, and describes the investigations conducted into the performance and sensitivity of the methods in real time application. Three architectures are presented and investigated. The first is the *Fully parallel design*, which implements directly the original parallel nature of the neural network algorithm. This architecture produced the best performance, but was the most resource hungry approach. It was used to incorporate the proposed activation functions devised in this study. The second architecture is called the *Hybrid design* 'folding' which

is inherent from the Fully parallel design, but employs the use of re-useable hardware techniques. It was shown that this approach is more effective than the first in that it minimises the usage of resources. The key to the success of this method is the introduction of memory functions which are used as computation optimisers. Both of these architectures are designed specifically for the Single neural network. Finally, the *Cascade design* was created to realise the functionality of neural networks which are arranged in a cascaded form. The architecture essentially implements the Hybrid design technique into a cascaded neural network. It was devised to provide both high accuracy and improved functionality, but has the expense of time delays during the complete operation.

For validation, all of the designs followed the three main processes in the general design flow, namely functional simulation, post synthesis simulation and real time analysis. Functional simulations are the simulations obtained from the explicit number system model, and have already been discussed in Chapter 3. Post synthesis is the simulation of the model represented by a bit true number system, and the real time analysis is performed using real time measurements. The second of these provides a pre-study of the performance of the output in real time, but the most important is the final real time analysis, which provides proof of the principle of the approach. This chapter also describes the design and implementation of the continuous function (as presented in Chapter 3) into a hardware circuit.

5.2 Fully Parallel Design

The most common means of implementing a neural network is to fully adopt the parallel computational function of the algorithm into a design, and program it directly into the FPGA. This approach offers clarity in terms of understanding the network's behaviour, which is advantageous during troubleshooting of the simulated design. The operation is also relatively quick. The major disadvantage of this approach is that it requires

extremely high usage of hardware resources, especially the digital multiplier (refer to Table 5.1). Mathematically, this is mainly due to the functions of the processing elements (PE) of the neural network, as the paradigm consists of matrix computation where using excessive multipliers and adders is inevitable if it is to be implemented in an authentic way. In this section, salient methods of designing and implementing the Fully parallel design are presented and discussed. The design is then used to realise the neural network discussed in Section 3.10 (with two inputs, six hidden nodes and two outputs), incorporating the proposed activation functions discussed in Chapter 4. The results of these analyses are presented and comparisons drawn.

5.2.1 Input and Output Integral

The digital design of the input and output integral of the Fully parallel design mimics the input and output integral designed with the explicit Simulink model (depicted earlier in Figure 3.14). However, adjustments were made to the output integral to accommodate the output layer part of the general normalised neural network equation (4.19). This was mainly done for the purpose of *de-normalisation*, to retrieve the actual output. Figures 5.1 and 5.2 depict the interior architecture of the input and the output integral respectively, with typical parameters. All of the digital mathematical operations involved only normalised values of not more than one, hence providing an accuracy of 14 bits wide with 13 fractional bits throughout.

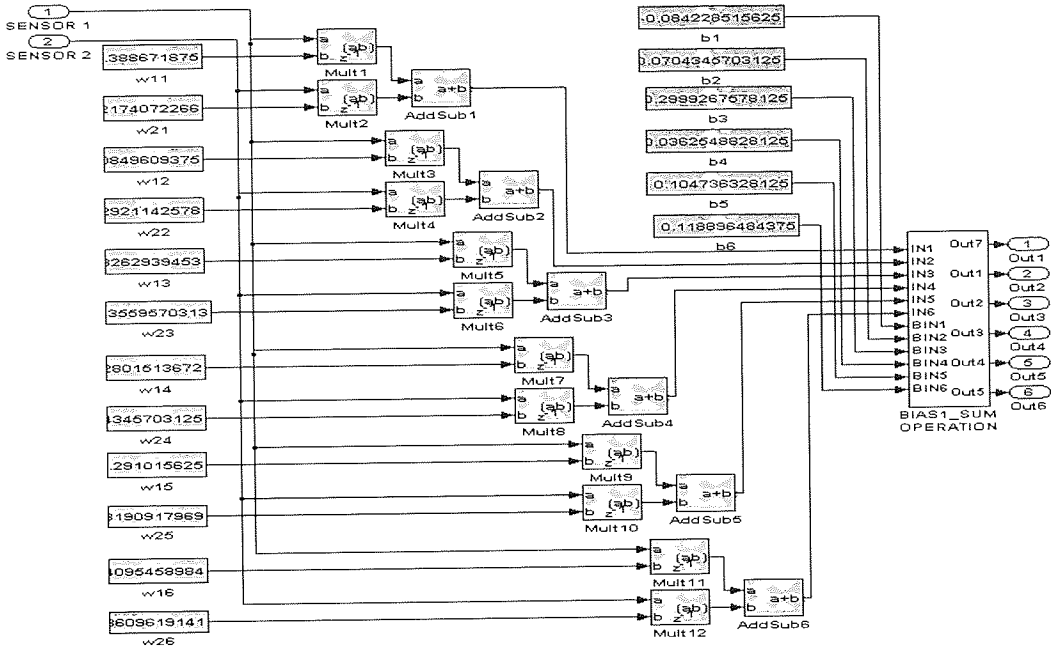


Figure 5.1: Input integral of the Fully parallel design.

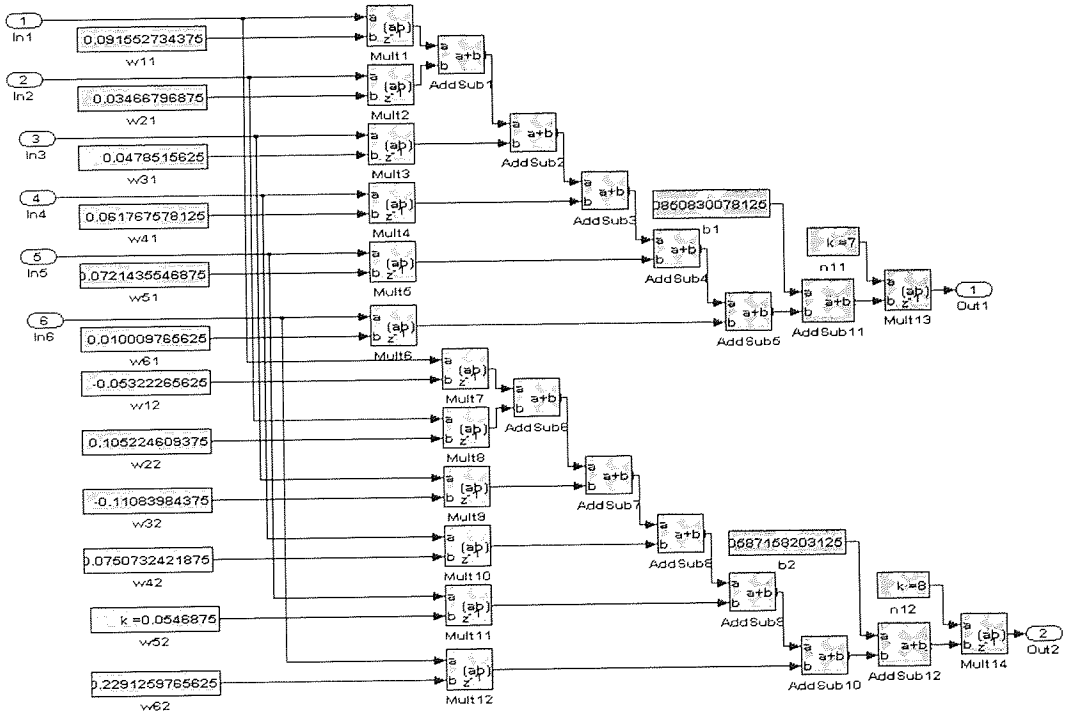


Figure 5.2: Output integral of Fully parallel design with 'de-normaliser' operations before the outputs.

5.2.2 Simulation and Implementation

The trained weights and biases were normalised using the algorithm explained in Section 4.5.1, to yield a normalised factor of Q_j and Y_i vectors. The normalised weights and biases and normalised factor Y_i were stored in the constant blocks of the design, employing the constant Xilinx Blockset – all of which were set to give accuracies 14 bits wide with 13 bits for the fraction. The normalised factor Q_j was incorporated into the various proposed activation functions design, the Gradient scheme approximation, the Polynomial scheme approximation, the Padé approximation and the Look-Up Table approach. All of the networks with different activation functions were then simulated and validated using a real time experiment. For fast prototyping of the neural network functions, the designs were generated, synthesised and implemented (following the implementation flow process discussed in Chapter 4), onto a Xilinx Virtex-II 2XCV3000 FPGA using a high-performance Nallatech XtremeDSP development board. The global frequency of the clock was generated from the on-board crystal oscillator built on the board to give a 64MHz operating frequency. The clock was also used to derive the differential clock signals for both the DAC and ADC by means of a feedback pin. Table 5.1 describes the resources required by the FPGA for each of the neural network designs.

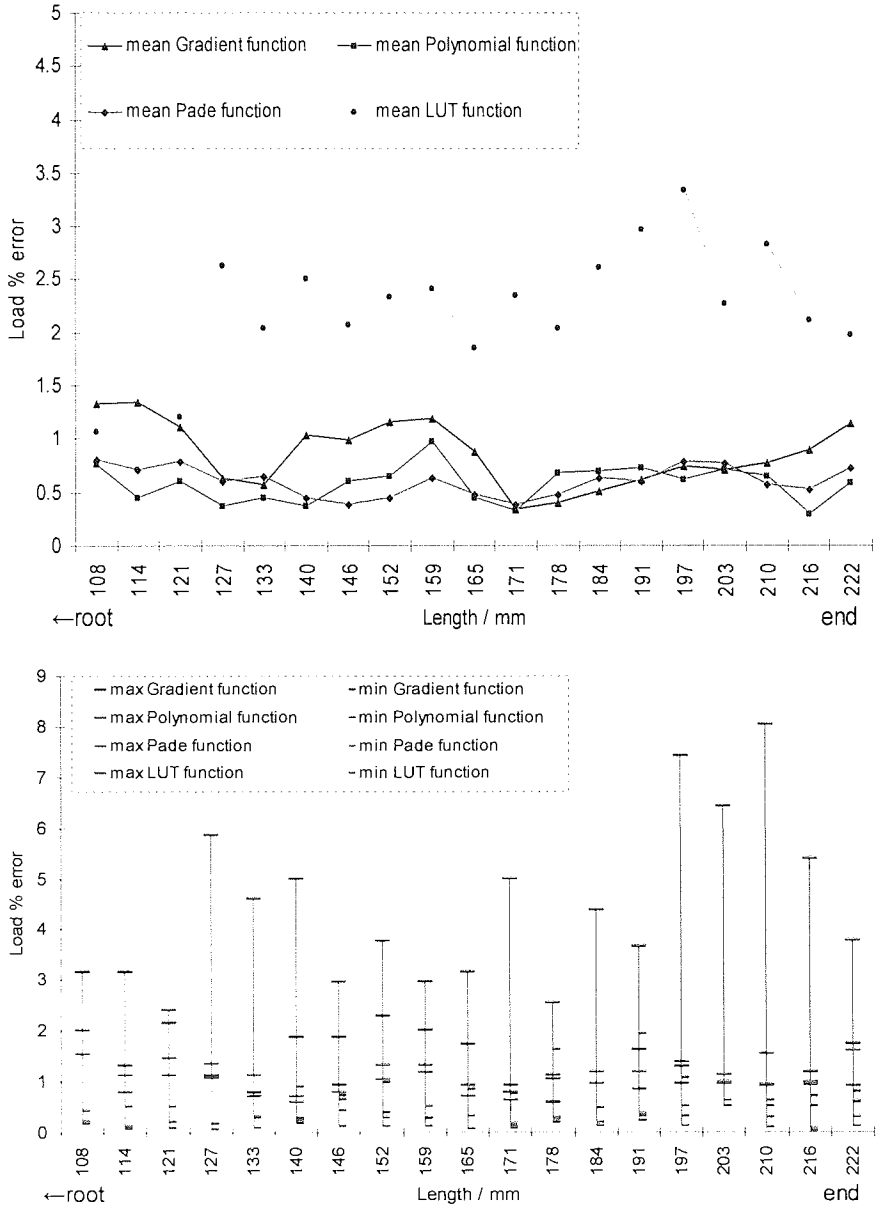
	Gradient Scheme	Polynomial scheme	Padé	Look-up Table
Occupied Slice	20%	18%	9%	18%
4 inputs LUTs	13%	14%	7%	5%
Bounded IOBs	11%	11%	11%	11%
MULT18X18s	76%	61%	60%	34%
Peak memory	176 MB	172 MB	142 MB	137 MB
Total Gate count	353K	292K	263K	494K

Table 5.1 Hardware resources required from each neural network design.

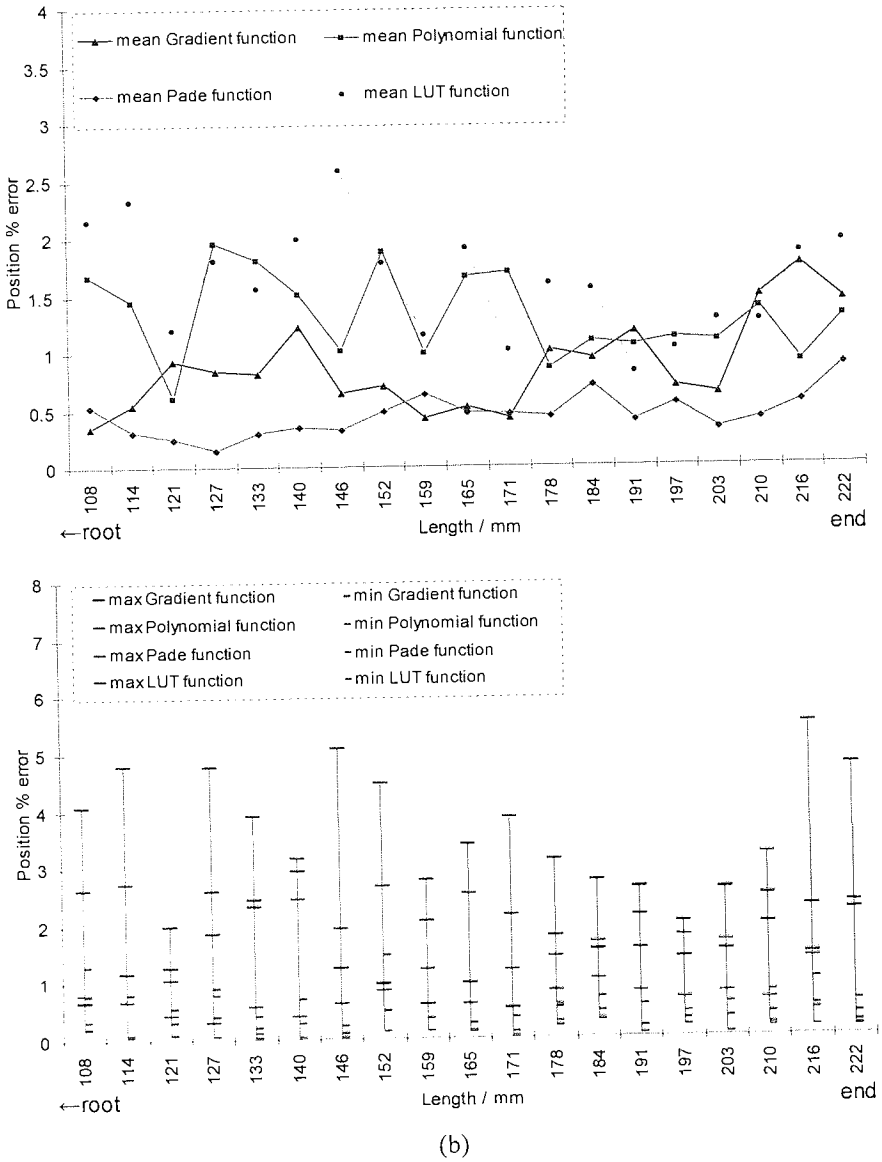
The output for load magnitude and load position interpretations of the neural network design with the Gradient scheme approximation, the Polynomial scheme approximation, the Padé approximation and the Look-Up Table (LUT) were plotted as percentage errors

against the actual length of the beam, with mean, minimum and maximum information (refer to Section 3.12.1). Figure 5.3 (a) and 5.3 (b) are comprised of plots of the results for the network outputs obtained by the bit true simulations for load magnitude and load position respectively. 5.3 (a) and 5.3 (b) reveal that the mean percentage error obtained using the Gradient scheme is better than 1.34% for load magnitude and 1.75% for load position. For the Polynomial approximation the mean error is better than 0.97% for load magnitude and 1.95% for load position. Performance is better using the Padé, which gives errors of better than 0.81% for load magnitude and 0.87% for load position. In contrast the scores were poor for the Look-Up Table, which produced errors of better than 3.34% for load magnitude and 2.58% for load position. Figure 5.4 (a) and 5.4 (b) are the plots of the validation results from the real time implementation for load magnitude and load position. Figure 5.4 (a) and 5.4 (b) reveal that load magnitude and load position errors respectively are better than 2.38% and 1.60% for the Gradient scheme, better than 1.69% and 2.05% for the Polynomial, better than 1.95% and 0.90% for the Padé approximation, and better than 3.59% and 3.28% for the Look-Up Table.

From these error measurements it is clear that the network using the Padé scheme was the most appropriate method of discriminating the loading parameters, followed by networks using the Gradient and Polynomial scheme (which have comparable performances), and finally the Look-Up Table approach. Errors produced by the network utilising the Look-Up Table are significantly higher, with the greatest error produced being 3.5% in comparison to the other schemes with maximum error rates of 2%. Improvements on these scores can be made by introducing more increments to the depth of the ROM used for the LUT, but this will require more of the FPGA's resources, especially in terms of the total gate count. However, overall all of the designs are still adequate for the flexible tactile digit application.

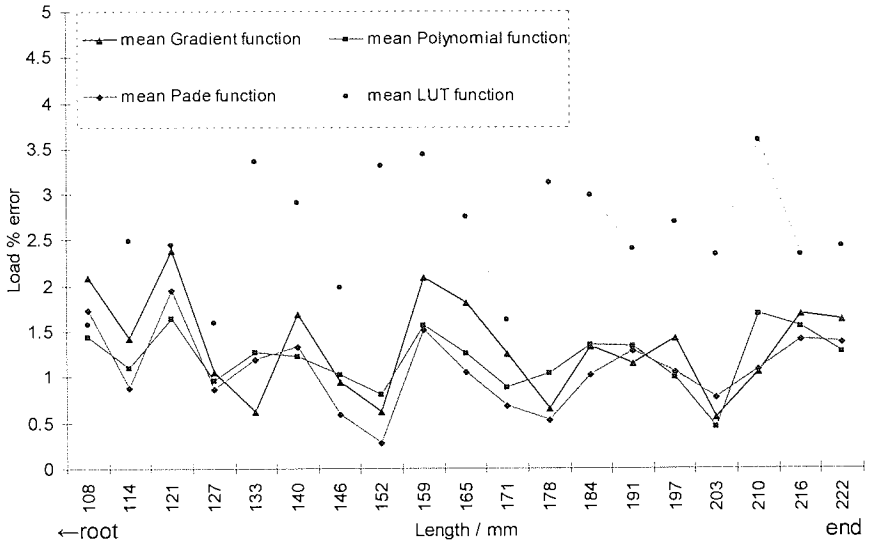

mean Gradient function
mean Polynomial function
mean Pade function
mean LUT function
Load % error
Length / mm
←root
end
max Gradient function
min Gradient function
max Polynomial function
min Polynomial function
max Pade function
min Pade function
max LUT function
min LUT function
Load % error
Length / mm
←root
end

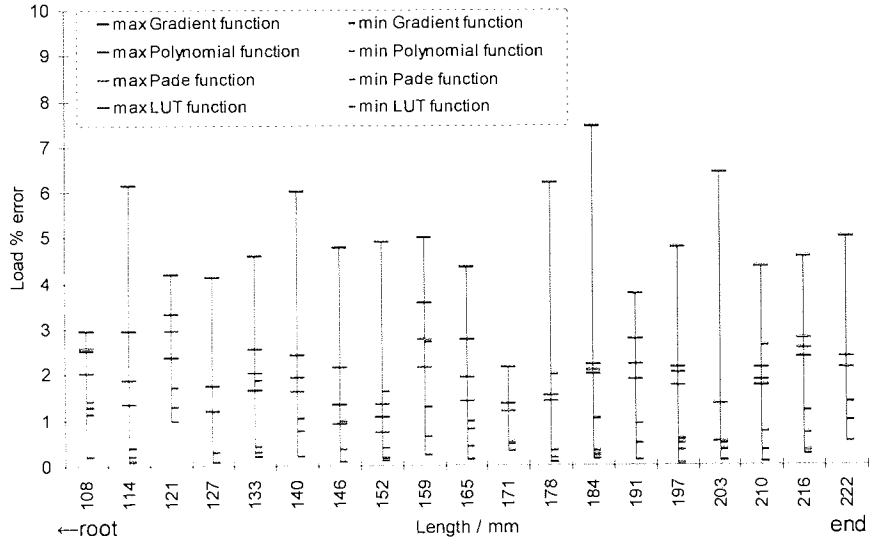
(a)



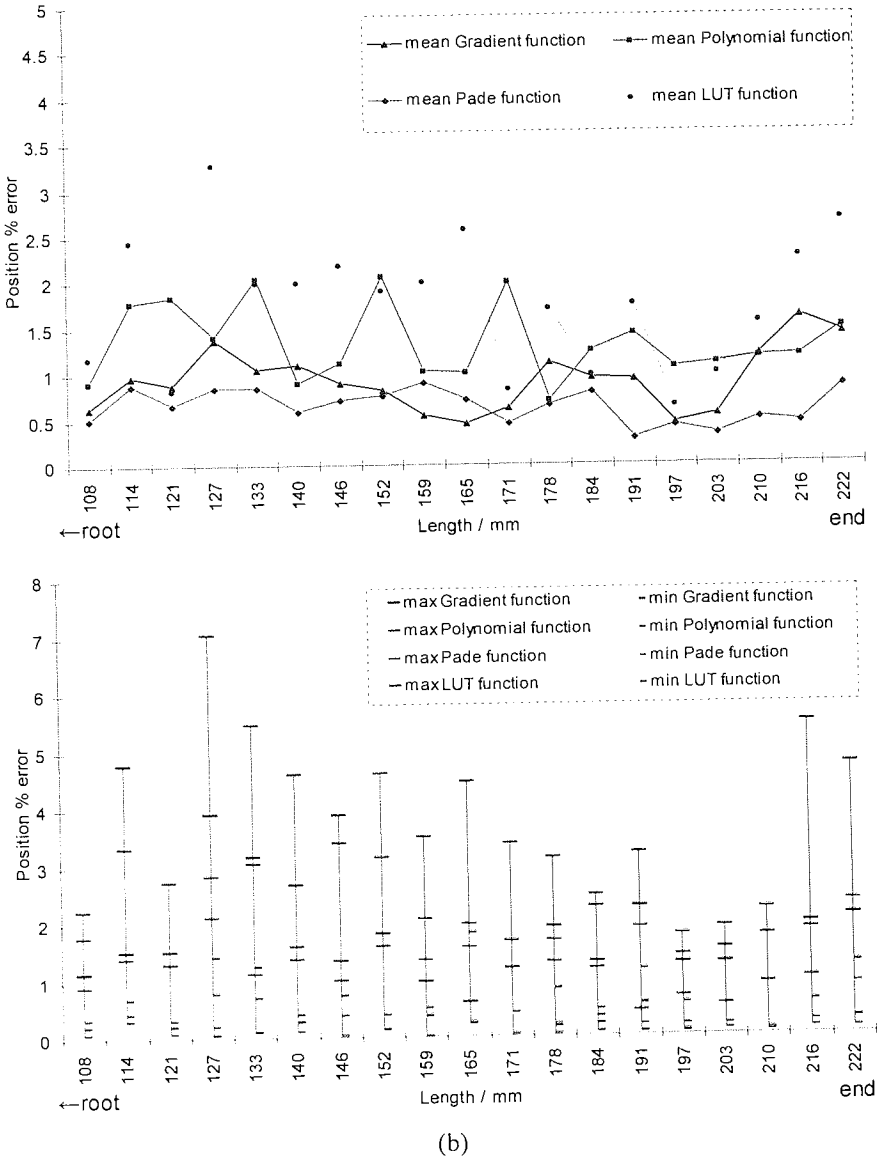
(b)

Figure 5.3: Error percentage measurement from bit true simulation for (a) load magnitude and (b) load position.


mean Gradient function
mean Polynomial function
mean Pade function
mean LUT function
Load % error
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
108
114
121
127
133
140
146
152
159
165
171
178
184
191
197
203
210
216
222
←root
Length / mm
end


max Gradient function
max Polynomial function
max Pade function
max LUT function
min Gradient function
min Polynomial function
min Pade function
min LUT function
Load % error
0
1
2
3
4
5
6
7
8
9
10
108
114
121
127
133
140
146
152
159
165
171
178
184
191
197
203
210
216
222
←root
Length / mm
end

(a)



(b)

Figure 5.4: Error percentage measurement from the real time test for (a) load magnitude and (b) load position.

5.3 Multiplexor

One characteristic of neural networks is the high inter-connectivity of their architecture. Although it is always possible to implement the whole algorithm directly into an FPGA using the available resources, usually most embedded systems could simply not accommodate the higher number of physical inputs. Even the system used here offers only two ADCs (14 bit accuracy) type AD6644, but offers a high sample period of 65MSPS. To overcome this limitation, an analogue multiplexor type maxim MAX4518CPD (see Appendix 6 for technical specification detail), as shown in Figure 5.5, was employed to multiplex up to four channels and feed the time-division sequence of analogue inputs into the embedded system's ADC Input selection can be controlled using two digital combinations of 'low' and 'low' for channel one, 'low' and 'high' for channel two, 'high' and 'low' for channel three, and 'high' and 'high for channel four. This analogue multiplexor should not be operated with switching frequency of higher than 10MHz in order to give optimum performance. Beyond this point the magnitude starts to deteriorate, causing inaccuracy to the input signal. In practice, a controller was designed in the main FPGA with user controllable sampling frequency and two user ports from the development board were used to generate a two bit stream of switching input.

The advantage of controlling the input stream from the main FPGA itself is that the multiplexor shares the same clock with the main processor, thus forcing the whole system to operate synchronically. In the FPGA, the sequence of digitised sensors signals from the ADC can be *de-multiplexed* digitally into the k inputs nodes (with each having a pre-constant characteristic) to retrieve each channel's information concurrently. The more subtle means of achieving this was chosen in this research, namely by retaining the sequential nature of the series of sensor input signals, but with the cost of using a different computational flow process of the neural network to suit the serial inputs to the network. This approach will reduce the time taken to retrieve the concurrent criteria of the network.

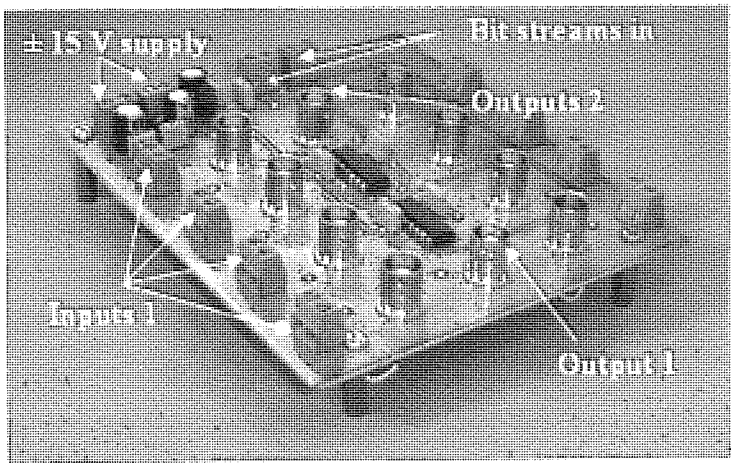


Figure 5.5: Two sets of multiplexors in parallel fabricated on a single circuit board.

5.4 Hybrid Design: Overview

Increasing the capability of the neural network to enable it to predict more output variables implies that more output neurons need to be introduced into the network. Consequently, the configuration of the hidden neurons of the system may have to be enhanced to accommodate the likelihood of a higher number of hidden nodes in order to give a better performance, by the optimisation of the learning process. Even the network described in Section 3.12 requires 16 hidden nodes with an extra two inputs from sensors to give an optimum solution for load magnitude, load position, load width and load shape discriminations (4:16:4 network configuration). However, it is inevitable that introducing more nodes means greater demand on FPGA resources. Yet, by adopting the implementation technique used for the fully parallel design (described earlier), and applying it to the 4: 16 : 4 network will result in an implementation that consumes an extensively high number of digital multipliers. This will be a problem for the case here where the available number of multiplier in the FPGA used is 96. An easy way to tackle this problem is to deploy a more powerful FPGA which has more resources, as described in Gadea, Cerda, Ballester and Mocholi (2000). The more practical way to deal with this

issue is by manipulating the algorithm itself so that the processing function can be optimised for that specific application.

There are number of ways to optimise the algorithm. The 'big hammer' approach is simply to increase the design, and try to squeeze it into the FPGA. The serial approach involves rewriting the original algorithm into a serial form of computation. In this chapter a more subtle approach is proposed, that combines some elements of concurrency while 'folding' the design to allow hardware re-use such that a sequences of data are passed through major subcomponents of the design. This process has the main objective during of maintaining a performance equal to the parallel design, while reusing available resources efficiently through a serial based technique. Specifically, the design was created to suit the serial inputs fed into it and, without postponing the operation of de-multiplexing them, follow the original concurrent neurons in the input layer. The concurrent hidden nodes operation still applies, but the design optimises the usage of the mathematical operation within each node. The system was extended to process 4 inputs, 16 hidden nodes and 4 outputs network.

5.4.1 Hybrid Design: the Rationale

From equations (5.1) and (5.2) it is obvious that a network having k input, j hidden nodes and i output would require (N x M) + (M x L) multipliers. For example, a network having 4 inputs, 16 hidden nodes and 4 outputs will require a total of 128 multipliers, which is beyond the digital multiplier resources of Virtex-II 2XCV3000 FPGA (which has only 96 available multipliers). However, as the inputs were multiplexed and fed serially into the processing system, each of the rows of weights (presented by equation (5.1)) can be generated serially as well, so that both can be multiplied serially but synchronically. The series outputs obtained from the product between the multiplexed inputs, the serial weight elements, and the corresponding bias are summed together to produce a node output of the input integral $\overline{q}$ (note that the signal is in normalised form) i.e. computation as a sequence of step but q_1, q_2 and q_M are computed in parallel. Each operation on a

single node in $\bar{q}_j$ can be accomplished using only single reusable multiplier and an adder. Note that also the 'T' outside the square bracket indicates the output is a column vector.

$$\underbrace{\begin{bmatrix} W_{1,1}^{(1)} & W_{1,2}^{(1)} & \cdots & W_{1,N}^{(1)} \\ W_{2,1}^{(1)} & W_{2,2}^{(1)} & \cdots & W_{2,N}^{(1)} \\ \vdots & \vdots & W_{i,j}^{(1)} & \vdots \\ W_{M,1}^{(1)} & W_{M,2}^{(1)} & \cdots & W_{M,N}^{(1)} \end{bmatrix}}_{W^{(1)}} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}^{serial} + \underbrace{\begin{bmatrix} B_1^{(1)} \\ B_2^{(1)} \\ \vdots \\ B_M^{(1)} \end{bmatrix}}_{B^{(1)}} = \tag{5.1}$$

$$\begin{bmatrix} \underbrace{x_1 W_{1,1}^{(1)} + x_2 W_{1,2}^{(1)} + \cdots + x_N W_{1,N}^{(1)} + B_1^{(1)}}_{\bar{q}_1} & \underbrace{x_1 W_{2,1}^{(1)} + x_2 W_{2,2}^{(1)} + \cdots + x_N W_{2,N}^{(1)} + B_2^{(1)}}_{\bar{q}_2} & \cdots & \underbrace{x_1 W_{M,1}^{(1)} + x_2 W_{M,2}^{(1)} + \cdots + x_N W_{M,N}^{(1)} + B_M^{(1)}}_{\bar{q}_M} \end{bmatrix}^T$$

For the case of equation (5.2), elements in each of the columns of weights $W^{(2)}$ (as shown by dashed line) are made serial. Each of these columns should generate elements synchronically, thus to form an ith row vector. This row vector then multiplied to the corresponding concurrent pre-constant r_j obtained from $\bar{q}_j$ after it undergoes denormalisation to produce q_j and the activation function. The multiplication operations are performed concurrently. The outputs from the operations and the corresponding bias are added together to produce each column of $\bar{y}_i$ output (note also that this is the normalised output y_i). This process is repeated for L times thus enable to sequentially generating the first, second, third and forth outputs. Each operation on a single $\bar{y}_i$ can be accomplished using only M number multiplier. These multipliers are reused for an unlimited number of L outputs. Following this proposed idea for computation, for the same configuration a network will require only 32 multipliers. Note that the 'serial' written outside the square bracket indicates that the output is a row vector.

$$\underbrace{\begin{bmatrix} W_{1,1}^{(2)} & W_{1,2}^{(2)} & \cdots & W_{1,M}^{(2)} \\ W_{2,1}^{(2)} & W_{2,2}^{(2)} & \cdots & W_{2,M}^{(2)} \\ \vdots & \vdots & W_{i,j}^{(2)} & \vdots \\ W_{L,1}^{(2)} & W_{L,2}^{(2)} & \cdots & W_{L,M}^{(2)} \end{bmatrix}}_{W^{(2)}} \cdot \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_M \end{bmatrix} + \underbrace{\begin{bmatrix} B_1^{(2)} \\ B_2^{(2)} \\ \vdots \\ B_L^{(2)} \end{bmatrix}}_{B^{(2)}} = \tag{5.2}$$

$$\begin{bmatrix} \underbrace{r_1 W_{1,1}^{(2)} + r_2 W_{1,2}^{(2)} + \cdots + r_M W_{1,M}^{(2)} + B_1^{(2)}}_{\bar{y}_1} & \underbrace{r_1 W_{2,1}^{(2)} + r_2 W_{2,2}^{(2)} + \cdots + r_M W_{2,M}^{(2)} + B_2^{(2)}}_{\bar{y}_2} & \cdots & \underbrace{r_1 W_{L,1}^{(2)} + r_2 W_{L,2}^{(2)} + \cdots + r_M W_{L,M}^{(2)} + B_L^{(2)}}_{\bar{y}_L} \end{bmatrix}^{serial}$$

Where x is the inputs to the neural network, and r is the output from the activation functions. N is the total number of inputs, M the total number of hidden nodes and L is the total number of outputs.

5.4.2 Memory Blockset: Register and ROM

One of the most important ingredients for the Hybrid design is the memories, which are used for holding and storing data and then executing them whenever they are needed. For holding temporary or changing data (i.e. variables), Xilinx Register Blocksets can be used. An initial value can be set to the block to specify the initial value in the register. For permanent storing purposes, the Xilinx ROM Blockset, a single port read-only memory is used to store the dedicated series of values (i.e. constants). These values are stored as words, and all words have the same arithmetic type, bit width and fractional point position. Each word is associated with exactly one address, and the address can be any unsigned fixed point integer from 0 to d-1, where d here denotes the ROM depth (number of words). The block has one input port for memory address and one output port for data out.

5.4.3 Modified Gradient Scheme Activation

The iterative nature of the regression formula used in the Padé scheme discussed earlier imposes a delay and the need for careful supervisory timing. To complement the Hybrid design, the activation function based on the Gradient scheme was used to create the modified Gradient activation function. It was chosen for this application because of its fast computation and the accuracy of the approximation the scheme can offer. One characteristic of this scheme is its superposition approach, namely solving problems by first breaking up the problem into selected ranges and proceeding through computation from selecting the range. This gives flexibility in re-structuring the system into the more practical digital architecture and facilitates the reusable technique, thus reducing the total FPGA resource usage effectively. Figure 5.6 shows the proposed modified design flow with the gradient scheme based on nine segments (refer to Table 4.1). The architecture employs the ABS and Sign retriever system that were discussed in Chapter 4, and two ROMs used for storing series values of the gradient G_n and the intercept i_n parameters. Ten range comparators, including the one used for addressing the input out of the range x_{sat}, were built to produce a single dimensional output vector carrying the address of the corresponding gradient G_n, and intercept i_n^g constants stored in the ROMs. These constants were then used for the gradient multiplication and the offset addition operations. Finally, the Sign retrieval operation is performed to generate the output of the activation function. With this new architecture, the activation function needs only a single multiplier to implement an unlimited number of segments. The design maintained the concurrent behaviour and the same accuracy as the original design, but effectively reduced the demand on the multipliers.

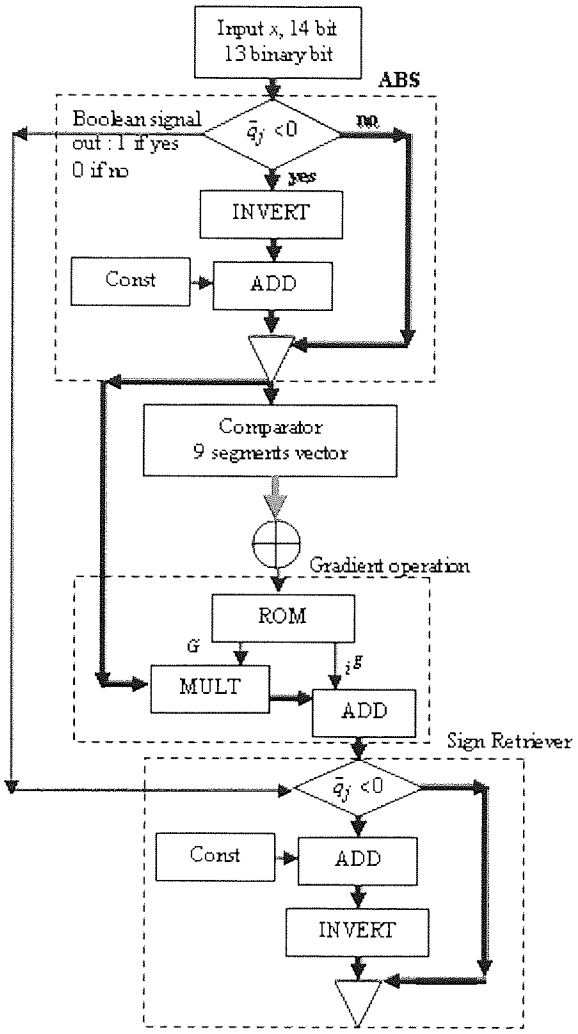


Figure 5.6: Modified Gradient scheme design flow.

5.4.4 Multiplexer Input Selector

The analogue multiplexor (see Figure 5.5) needs signals to select the sensor inputs. In practice, these signals are generated from specific-purpose building blocks designed in the FPGA. The building blocks are comprised of a counter and slice functions. The counter function is used as a primary element for controlling both the analogue multiplexor (i.e. speed of switching and channel address) and the sampling rate of the ADC. The counter also performs a frequency divider operation. The counter is used to

deliver a cyclic counted *bn* bits output (i.e. 2^0 to 2^{bn-1}), where *bn* here is an unsigned finite number. Slice Blocksets are then used to extract a two bit stream from the wordlength within the *bn* bit count value (for example the most and the second most significant bit), before sending them to the gateways (headers) out of the embedded system. More specifically, the most significant bit has a frequency of FPGA clock frequency / 2^{bn-1}, whereas the second most significant bit has FPGA clock frequency / 2^{bn-2}. However, there are some important factors in play when selecting the appropriate *bn*. First among these is the capability of the multiplexer device. Using an *bn* that is too small leads to the switching frequency become too fast. If this is beyond the optimum switching frequency of the multiplexor, the output magnitude can be impaired. However, *bn* that is too high will lead to the switching frequency becoming sluggish. Considering other applications such as the highly dynamic operation, this would lead to a problem of *undersampling*, the multiplexor being under sample. A method to prevent this second problem is the application of the Shannon sampling theorem (Bolton, 1999).

The problem of undersampling can also occur to the ADC if the value of *bn* is too small. Since the ADC of the system was clocked using the same clock frequency of the main FPGA, using an insufficiently high *bn* will mean that the ADC may acquire insufficient sampled information. In practice, the *bn* was chosen to be 12, thus with a 64MHz FPGA clock frequency, the clock is divided down by 2^{10} and 2^{11} to give 62.5KHz and 31.25KHz respectively for the two most significant counter bits. These two bits were used as multiplexor addresser, giving the system to have four times 15.625KHz between one complete channel switching. The factor was shown to be appropriate for the applications throughout the current research (including the application described in Chapter 6). Figure 5.7 shows the real time output of the two bit streams (generated from the FPGA) dedicated to the channel switching of the analogue multiplexor. The red line indicates the most significant bit, whereas the blue line is the least significant bit (second most significant bit).

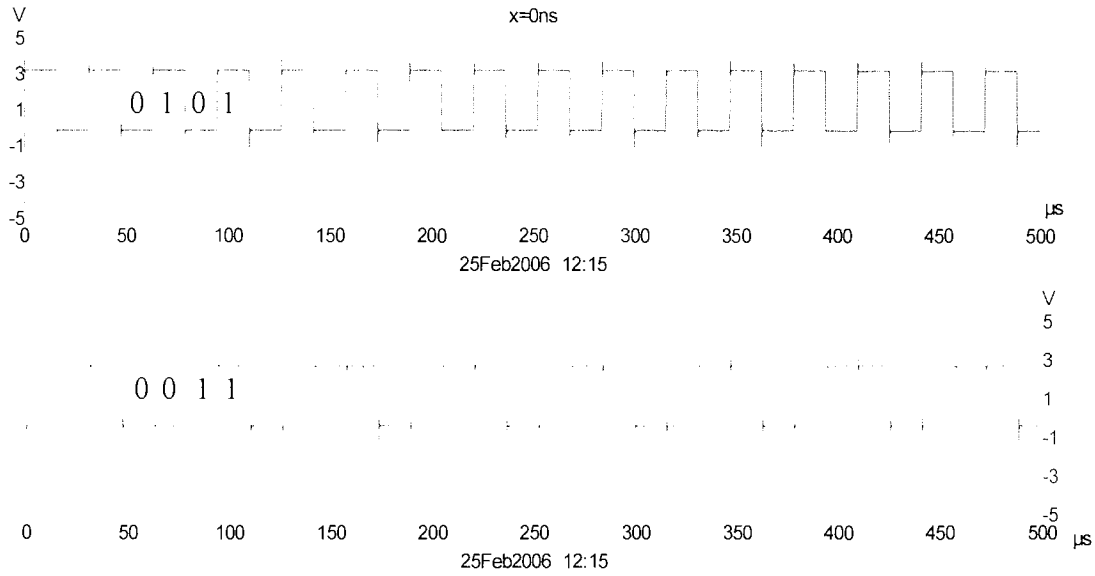


Figure 5.7: Real time measurement of the multiplexor channel addresser sgnal (by digital PICO scope).

5.4.5 Hybrid Design: The Architecture

The hybrid design, with serial node operation of the concurrent matrix computation as explained in Section 5.4.1 was implemented as an application-specific digital design. Figure 5.8 provides a simplified diagram of the Hybrid design for 4 inputs, 16 hidden nodes and 4 outputs. Single ADCs and DACs are needed to take the multiplexed four input signal from the sensors and to serially output the processed signal containing information for the four outputs. The design is still comprised of three main operations; the input integral, the hidden activation and the output integral. The optimised weights and biases adopted from Chapter 3 were normalised using the algorithm explained in Section 4.5.1, thus producing *(1* x *16)* and *(4* x *1)* vectors for Q_j and Y_i respectively.

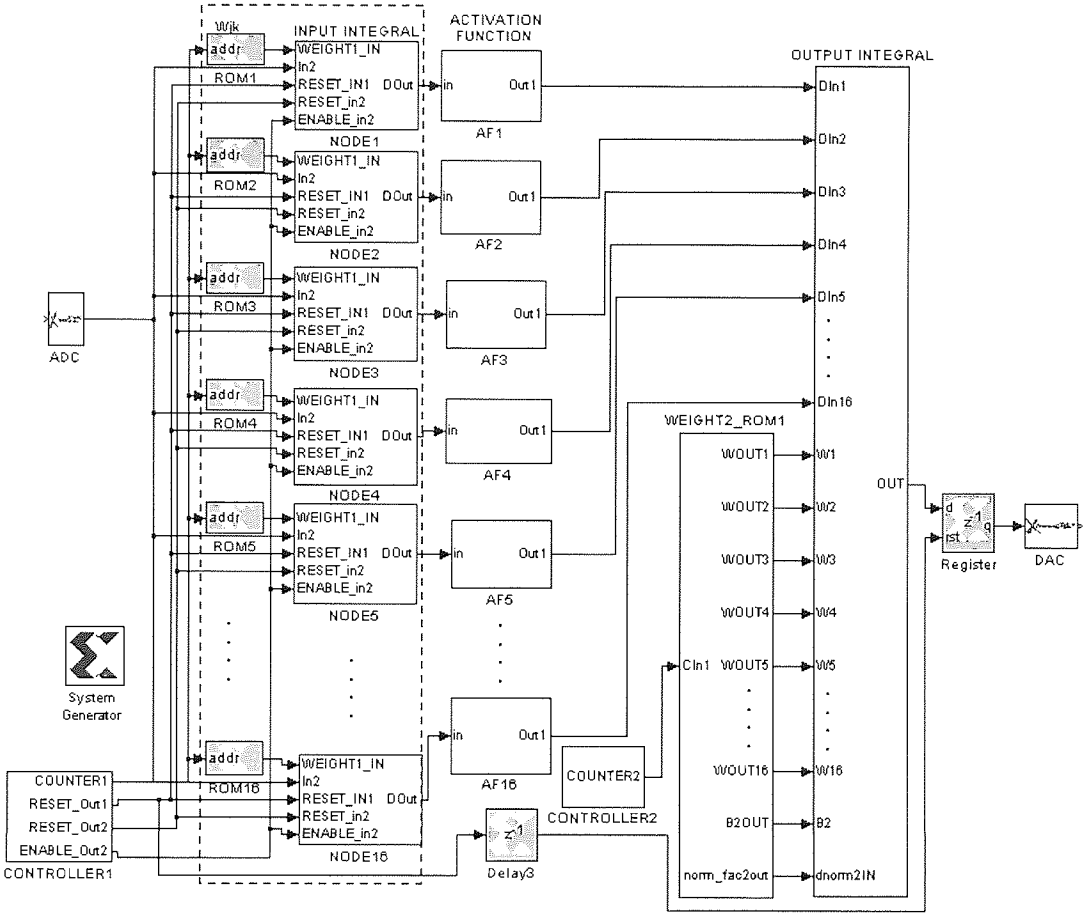


Figure 5.8: Hybrid neural network design for 4 multiplexed inputs with 16 hidden nodes and 4 outputs.

5.4.5.1 Input Integral Architecture

The input integral subsystem takes the multiplexed sensor signal x_k and channels it into *16 j* functional blocks, or nodes. Each of the nodes have a single digital multiplier, an adder, and two registers. Figure 5.9 illustrates the configuration of a single node (or neuron) of the input integral with the ROM and the shared controller, CONTROLLER 1 (see also Figure 5.8). The controller is a simple counter or sequencer, which is used to generate addresses for executing the series of weights stored in all of the ROMs, and the clock enable output which is used for resetting and enabling purposes for each of the reusable operations. A total of 16 ROMs were required in the overall input integral

subsystem to store the sets of normalised weights W_{jk}, each of which contained four signed (two's compliment) constants in series (corresponding to each of the four inputs). Each ROM was assigned to give an output accuracy 14 bits wide with 13 bits fractional accuracy.

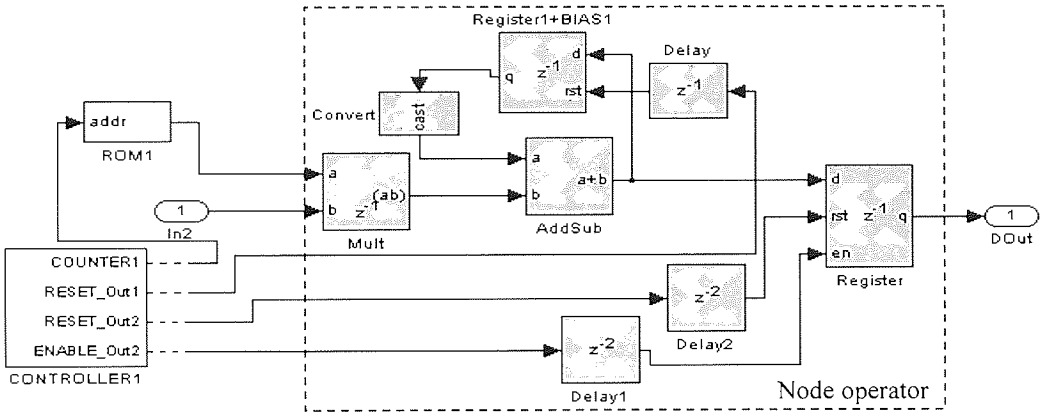


Figure 5.9: One of the re-useable designs of the input integral.

When the first sensor signal x_1 arrives, a counter which controls the execution of the weights W_{jk} sequence, produces the corresponding signal to the ROMs ordering them to output the corresponding weights W_{j1} to the node operators. Digital multiplications are performed concurrently between the first element of weight W_{j1} and the first data x_1 in all the j number of node operators. At each of the j nodes (as in Figure 5.9), the output from the multiplication is then digitally added to the bias B_j stored as an initial value in the feedback register. The output from the addition is held temporarily in the register before it is fed back to the input and added to the next multiplication output, x_2 and W_{j2}. The loop operations continue for k times to complete one cycle of the operation. The registers with reset and enable ports are used to extract and hold the last information containing the final values for $(W_{jk}x_k)+B_j$. This will create a steady output but reset to zero upon completion of the operation, to allow the next operation to present new samples. The register takes the reset and enables signals also from CONTROLLER 1. Because the parameters and variables are normalised, all of the operations are completed within an accuracy of 14 bits wide with 13 bits fractional accuracy.

5.4.5.2 Activation Function: The modified Gradient Scheme

Figure 5.10 shows the configuration of one of the activation node operators in the modified activation function subsystem. The node consists of five building functional subsystems; ABS, Addresser, Memories, GRAD_EQN and Sign retriever.

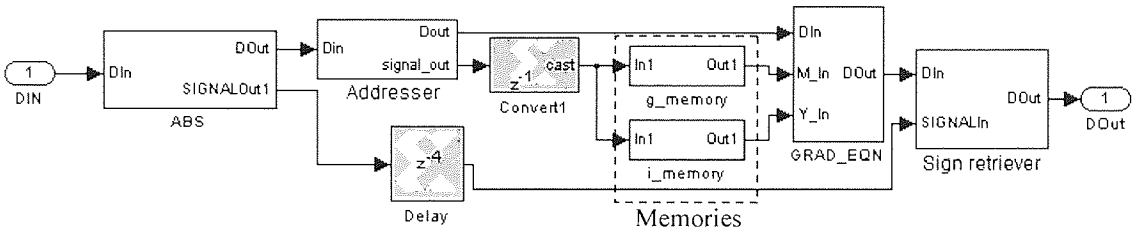


Figure 5.10: Modified Gradient scheme activation function.

The Addresser operating block takes signal from the ABS and generates the address of the gradient and offset needed for the gradient operation. It also forwards the signal to the gradient operation. The addresser consists of ten sub-units in parallel. Each contains a comparator and a locator (operated by relational and logical blocksets), and adder operator which is responsible for generating the addresser of the corresponding gradient and offset. The output can be reset via a register controlled by a Boolean output from the comparator and locator section.

The Memories consisting of g_memory and i_memory blocks are ROMs which store the sets of normalised gradient G_n , as tabulated in Table 5.2, and offset i_n^g for the operation, as tabulated in Table 4.3. Before a signal is ready to be processed in the gradient block operator, the g_memory and i_memory blocks wait for the address input to the blocks. The memory blocks output the corresponding constants when they receive an address signal.

In the gradient operating subsystem GRAD_EQN, the arrived signal is multiplied with the corresponding normalised gradient and then added to the offset.

G/g_nQ_j	Segments								
	1	2	3	4	5	6	7	8	9
nodes↓/j	0.0582	0.0374	0.0225	0.0148	0.0086	0.0051	0.0028	0.001	0.0002
1	0.1329	0.0854	0.0514	0.0339	0.0197	0.0117	0.0063	0.0022	0.0005
2	0.1551	0.0997	0.06	0.0396	0.023	0.0137	0.0073	0.0026	0.0006
3	0.1163	0.0748	0.045	0.0297	0.0173	0.0103	0.0055	0.0019	0.0004
4	0.0846	0.0544	0.0327	0.0216	0.0126	0.0075	0.004	0.0014	0.0003
5	0.1551	0.0997	0.06	0.0396	0.023	0.0137	0.0073	0.0026	0.0006
6	0.1551	0.0997	0.06	0.0396	0.023	0.0137	0.0073	0.0026	0.0006
7	0.0846	0.0544	0.0327	0.0216	0.0126	0.0075	0.004	0.0014	0.0003
8	0.093	0.0598	0.036	0.0237	0.0138	0.0082	0.0044	0.0015	0.0003
9	0.1551	0.0997	0.06	0.0396	0.023	0.0137	0.0073	0.0026	0.0006
10	0.0291	0.0187	0.0112	0.0074	0.0043	0.0026	0.0014	0.0005	0.0001
11	0.1551	0.0997	0.06	0.0396	0.023	0.0137	0.0073	0.0026	0.0006
12	0.0517	0.0332	0.02	0.0132	0.0077	0.0046	0.0024	0.0009	0.0002
13	0.0465	0.0299	0.018	0.0119	0.0069	0.0041	0.0022	0.0008	0.0002
14	0.1163	0.0748	0.045	0.0297	0.0173	0.0103	0.0055	0.0019	0.0004
15	0.1163	0.0748	0.045	0.0297	0.0173	0.0103	0.0055	0.0019	0.0004
16	0.0582	0.0374	0.0225	0.0148	0.0086	0.0051	0.0028	0.001	0.0002

Table 5.2: Normalised sets for gradients.

5.4.5.3 Output Integral Architecture

Figure 5.11 shows the simplified output integral design which performs the $(W_{ij}^{(2)}r_j + B_i^{(2)})Y_i$ operation of the neural network. 18 ROMs were needed to store the $W_{ij}^{(2)}$ weights, the biases and the normalising factors. The ROMs take the signal from CONTROLLER 2 (see also Figure 5.8), which is comprised of a counter. The ROM should contain four signed numbers for each value of i to compute $\overline{y}_i$ (see equation (5.2)). The design takes 16 j outputs concurrently from the activation subsystem and then multiplies these concurrently with the set of W_{1j} weights from the ROMs (refer to equation (5.2)). They are summed together before being added to the bias $B_1^{(2)}$ and multiplied with the normalising factor Y_1 to produce the output for load magnitude, y_1. Following this is the concurrent multiplication of the 16 outputs with the set of W_{2j} weights. They are summed together before being added to the bias $B_2^{(2)}$ and multiplied with the normalising factor Y_2 to produce the output for load position, y_2. When the

operation is completed, the steady outputs from the activation subsystem which are multiplied with the third set of weights $W_{3j}^{(2)}$ added to the second bias $B_3^{(2)}$ and multiplied with the normalising factor Y_3 to yield the output for load width, y_3. The operation is repeated for y_4, the load shape.

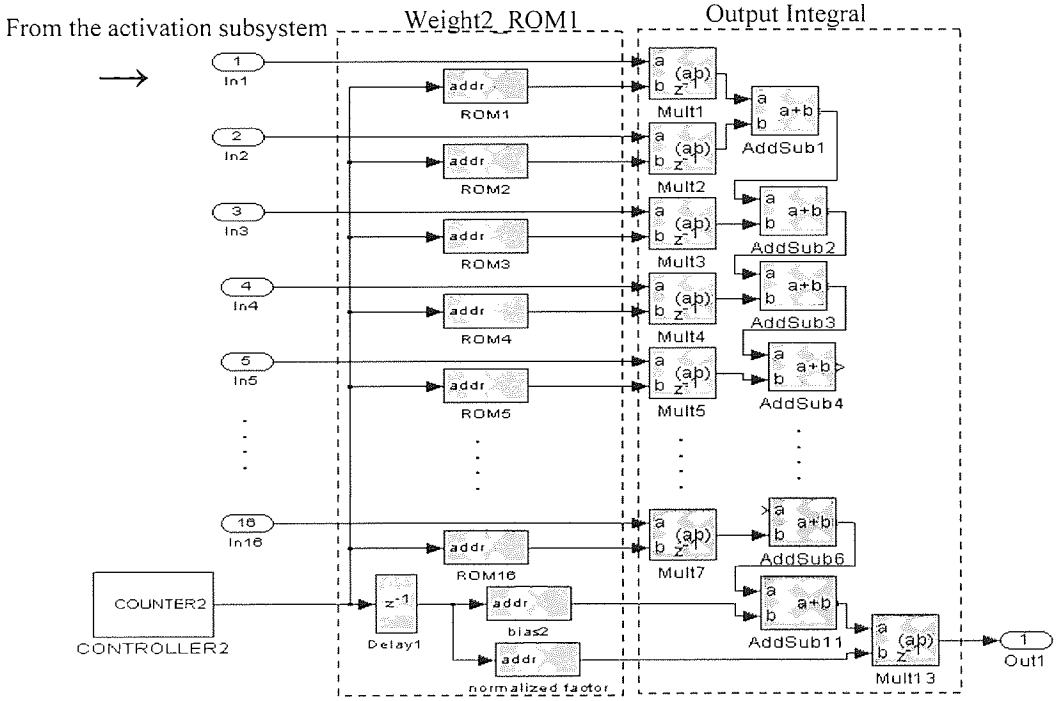


Figure 5.11: Output transfer subsystem for the 16 nodes.

5.5 The Cascade Design

To facilitate the realisation of the cascaded neural network proposed in Section 3.12, the project then introduced the Cascaded network design for the digital implementation. Figure 5.12 shows the configuration of the digital cascade approach designed with Xilinx Simulink Blocksets. The network architecture follows the topology shown in Figure 3.22 with four neural networks in cascade (NN_A, NN_B, NN_C and NN_D). These were

constructed so that they determine the load parameters individually, those of load magnitude, load position, load width and load shape respectively.

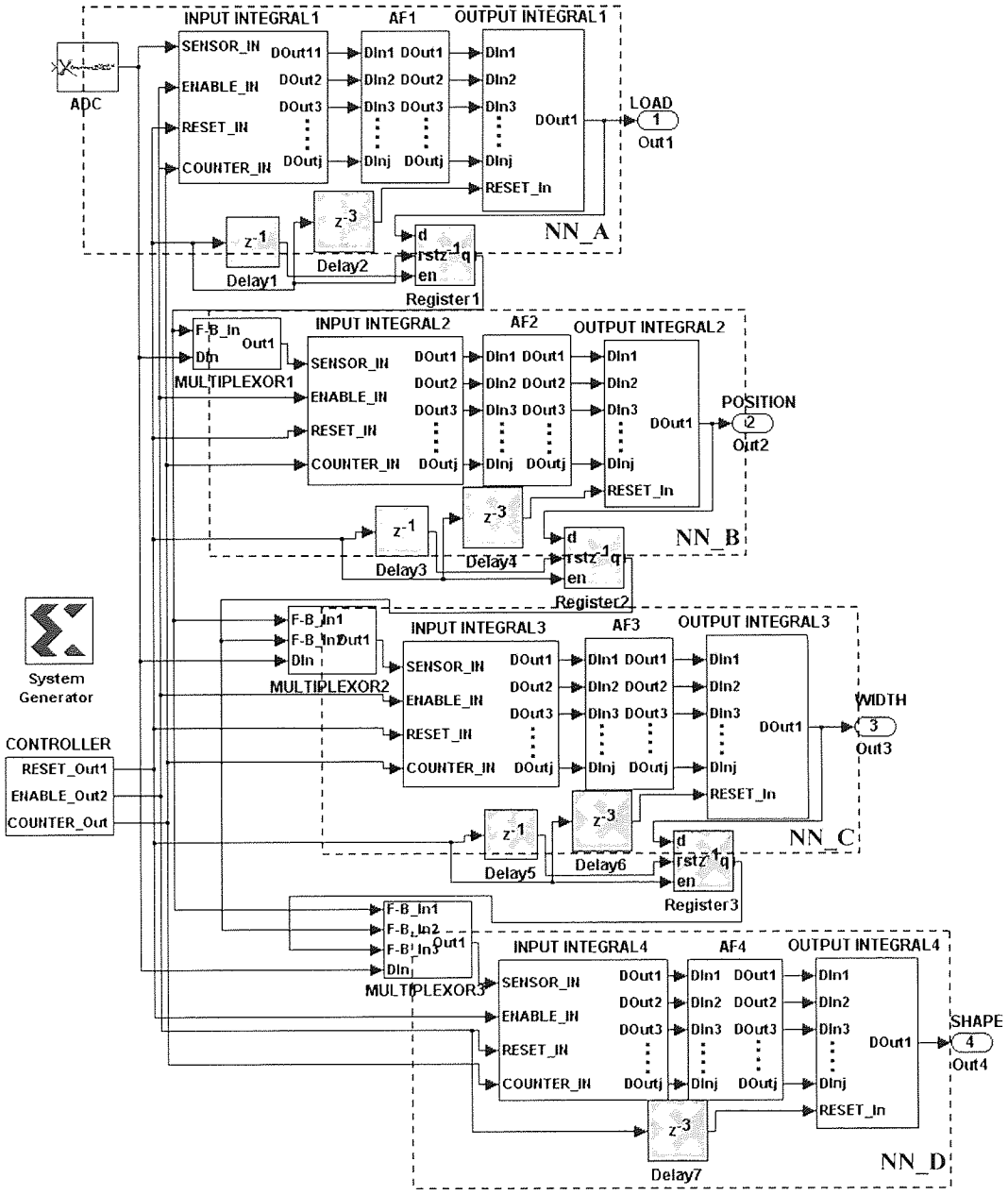


Figure 5.12: Cascade Neural network design in Xilinx Simulink MATLAB.

Each of the neural networks in the cascade directly adopts the Hybrid design, but adjusts the output integral to approximate a single output only. Therefore, because $\omega_{ij}^{(2)}$ is a single

dimensional vector, the ROM can be replaced by constant Blocksets to store the normalised weights, biases and normalised factor Y_i of the output integral.

Registers with reset and enable ports were introduced to the outputs of the first three networks; NN_A, NN_B and NN_C, to extract the discriminated outputs for load, load position and load width respectively. Generally, together with the sensor information, these extracted outputs (load position, load width and load shape) were utilised as inputs to the next network. To include the neural network outputs with the serial sensor inputs, multiplexer subsystems comprised of digital multiplexors and counters input delays were introduced (as shown in Figure 5.12). The purpose of this was to multiplex the original input sensors and the relevant retrieved neural network outputs, hence creating an input stream which carries within it the original sensor inputs and the evaluated output information in order. In Figure 5.12, these multiplexer subsystems are introduced as MULTIPLEXER 1 to multiplex sensor inputs with the retrieved discriminated load, MULTIPLEXER 2 to multiplex sensor inputs with the retrieved discriminated load and load position, and MULTIPLEXER 3 to multiplexed sensor inputs with the discriminated load, load position and load width. In addition, it was necessary to balance the delays in the networks so that, for each stage, a consistent set of sensor signal and computed neural network outputs were delivered to each stage in the cascade. To do this the signal delays shown in Figure 3.22 were built into the corresponding multiplexor rows.

However, as the sensor input stream to the cascade design should act as an information carrier, it is necessary that the stream is eligible to take additional samples. In order to do this, the previous bits stream shown in Figure 5.7 was modified to accommodate an extra three "sample inputs" containing the computed neural network outputs of load magnitude, position and width output. A clock enable with a predefined reset frequency was introduced to reset the counter described in Section 5.4.4, to produce a two bit stream with a cyclic pattern of 00 01 10 11 00 01 10 output. The new bit stream was recorded and is represented in Figure 5.13. The blue line indicates the second most significant bit and the red is the most significant bit. The response was obtained from the real time measurement of the gateways of the FPGA. The bit stream was fed into the analogue multiplexer – a typical output taken from the sensors is shown in Figure 5.14.

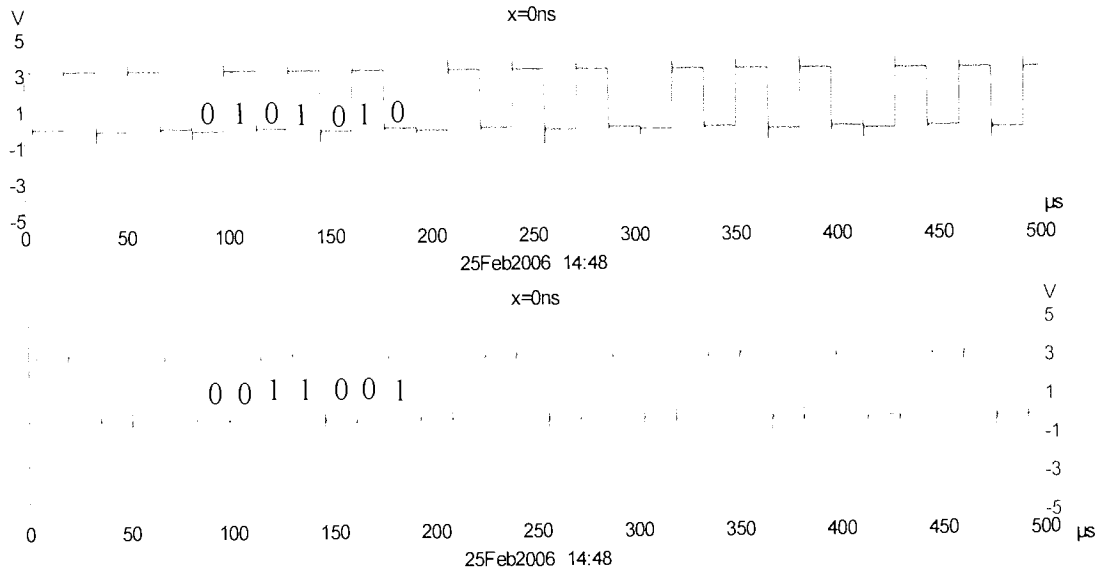


Figure 5.13: Timing response of the modified bit stream (by digital PICO scope).

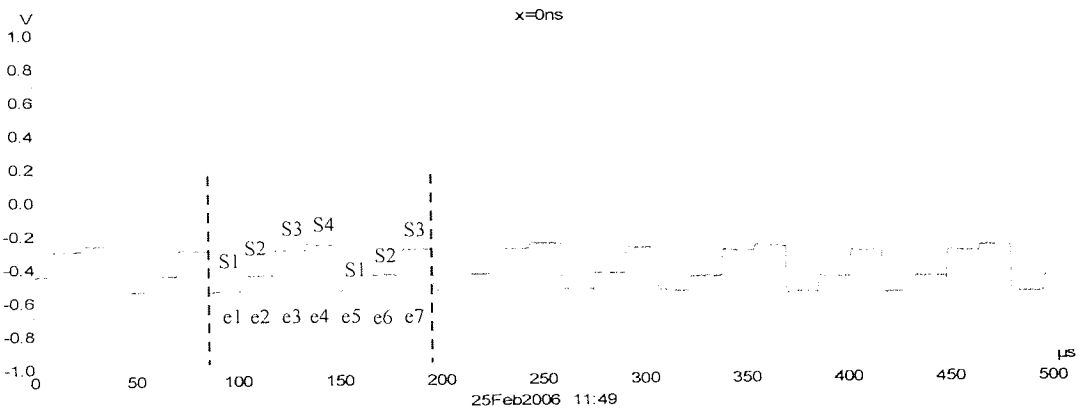


Figure 5.14: Typical multiplexed sensors from the output of the multiplexer (by digital PICO scope).

S1, S2, S3 and S4 are the channels which are selected by the 00 01 10 and 11 combinations respectively. The channels contain signals from typical experimental sensor readings of Sensor 1, Sensor 2, Sensor 3 and Sensor 4 respectively. The combinations of the bit stream forced S1, S2 and S3 to be selected twice in a complete cycle, thus creating seven elements in the cycle. Note that S1, S2 and S3 occur repeatedly in element e5, e6, and e7 of the complete cycle. These elements provide a means to include the necessary

signal into the input stream using a method of replacement. Specifically, the 'unwanted' S1, S2 and S3 samples of the elements e5, e6, and e7 were replaced by the discriminated load magnitude, load position and load width data from the neural network which were multiplexed into the stream at precise timing. For the computation of NN_B, element 5 is replaced by load magnitude data created by multiplexing the stream with the evaluated output fed from NN_A, thus producing a stream containing the cyclic input of Sensor 1, Sensor 2, Sensor 3, Sensor 4 and evaluated load. A similar procedure is performed to replace element e5 and e6 with the evaluated load and position, hence enabling NN_C to take the required six inputs (Sensor 1, Sensor 2, Sensor 3, Sensor 4, evaluated load and evaluated position data), and then compute for width W. Following this is the replacement of elements e5, e6 and e7 with the evaluated load, position and width, hence enabling NN_D to take the required seven inputs (Sensor 1, Sensor 2, Sensor 3, Sensor 4, evaluated load, evaluated position and evaluated width).

In the input integral the presence of the unwanted data elements e5, e6 and e7 for NN_A, element e6 and e7 for NN_B, and e7 for NN_D was cancelled by multiplying them with zero (ground) before summing to completion the $(W_{jk}^{(1)}x_k)+B_j^{(1)}$ operations. For these operations each ROM in the input integral of the NN_A, NN_ B and NN_ C is programmed to store an extra 3, 2 and 1 series of compensated zeros respectively.

To test the system, real sensor signals were fed into the multiplexor. Figure 5.15 illustrates the real time measurement output from the multiplexing system with a direct connection from the ADC to DAC via the FPGA of the embedded system. The readout of the measurements shows the consistency of the digital magnitude and the analogue reading (Figure 5.14).

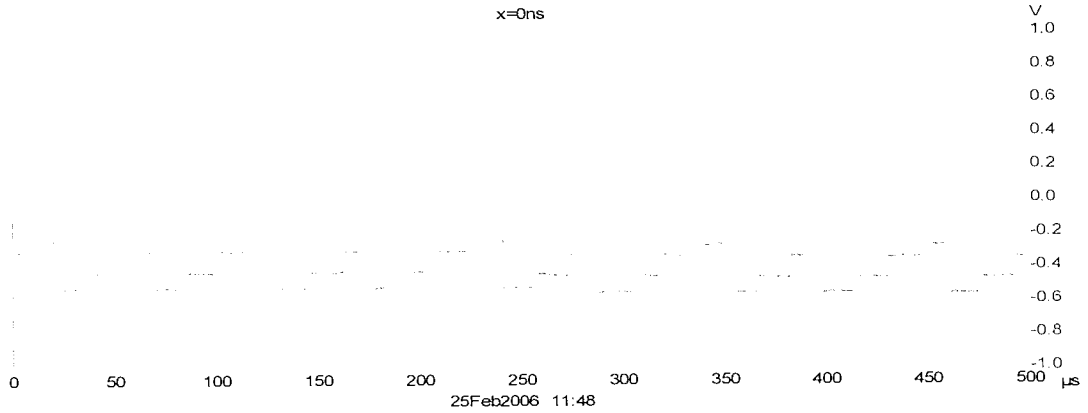


Figure 5.15: Typical multiplexed sensors from the output of the DAC of the FPGA (by digital PICO scope).

5.5.1 Network Optimisations

Two optimisation types were used in this research to obtain the optimised design configuration for the cascade network. First there is the explicit number system of optimisation (as used in Section 3.12.1), in which each network is trained and optimised using the sensor inputs and the computed outputs produced by simulating the explicit feed forward model. The configuration of the cascade network produced from the optimisation is 4:5:1, 5:5:1, 6:5:1 and 7:6:1 for NN_A, NN_B, NN_C and NN_D respectively. A second option is the bit true number system of optimisation. Here the cascade network is optimised by the procedures exercised in Section 3.12.1, but taking into account the use of the bit true feed forward model, (digital design) to simulate the outputs. The second option is more appropriate for the real case scenario, as ideally (with the exception of noise and assuming that the sensors are repeatable) the real time outputs will be the outputs obtained by the bit true simulation. However, the drawback to this approach is that it requires a high cost in terms of simulation time. The configuration of the new cascade network produced from the bit true optimisation is 4:5:1, 5:5:1, 6:4:1 and 7:4:1 for NN_A, NN_B, NN_C and NN_D respectively. Note that from here onward the first cascaded network (optimised using explicit models) will be known as Cascade

design A and the second cascade network (optimised using bit-true models) will be know as Cascade design B.

5.6 The Hybrid and the Cascade Designs: The Results

The Hybrid design and the Cascade designs A and B were simulated using bit true simulation to represent their digital implementations. To study the error performances of the simulations and real time measurements, all of the errors from the Hybrid, Cascade design A and Cascade design B were computed following equations (3.15) and (3.16). Figure 5.16 to Figure 5.19 were produced to illustrate the comparison between the Hybrid design and the Cascade design A and B.

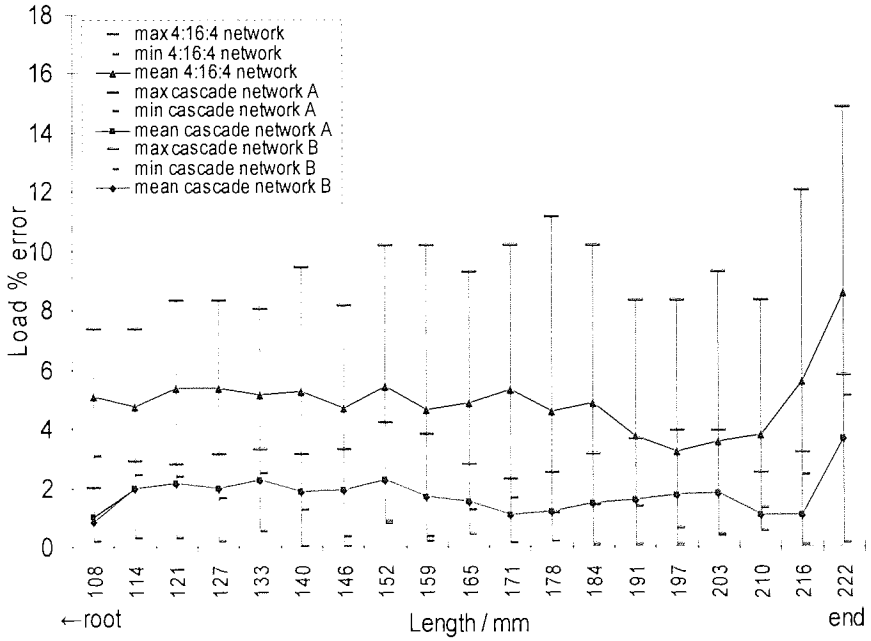


Figure 5.16: Error percentage of load magnitude from the bit true simulation of the Hybrid, Cascade A and Cascade B networks.

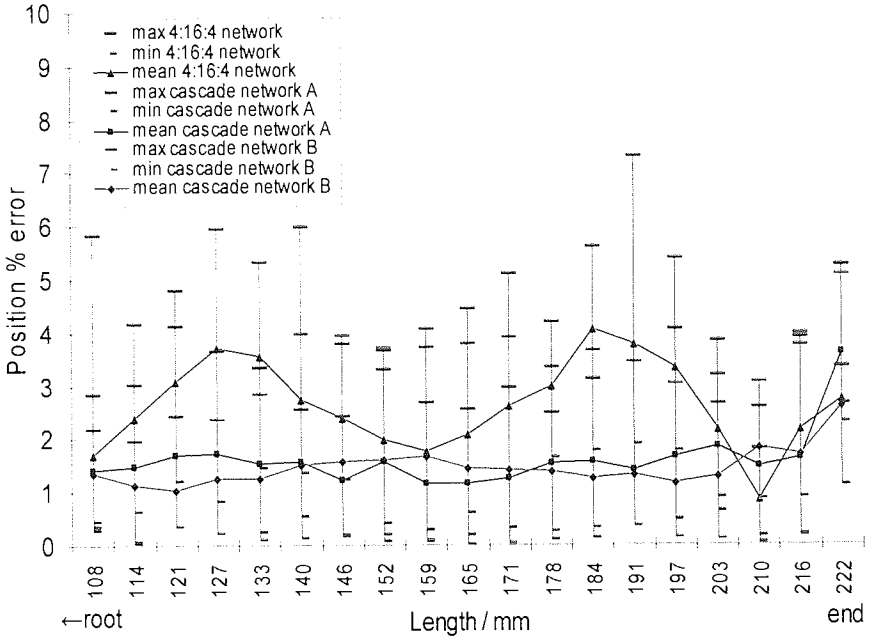


Figure 5.17: Error percentage of load position from the bit true simulation of the Hybrid, Cascade A and Cascade B networks.

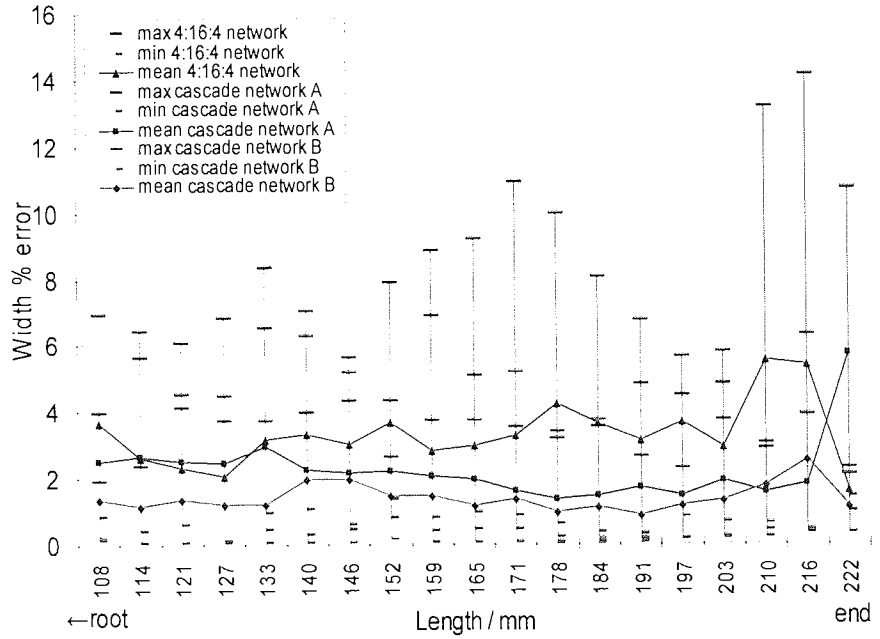


Figure 5.18: Error percentage of load width from the bit true simulation of the Hybrid, Cascade A and Cascade B networks.

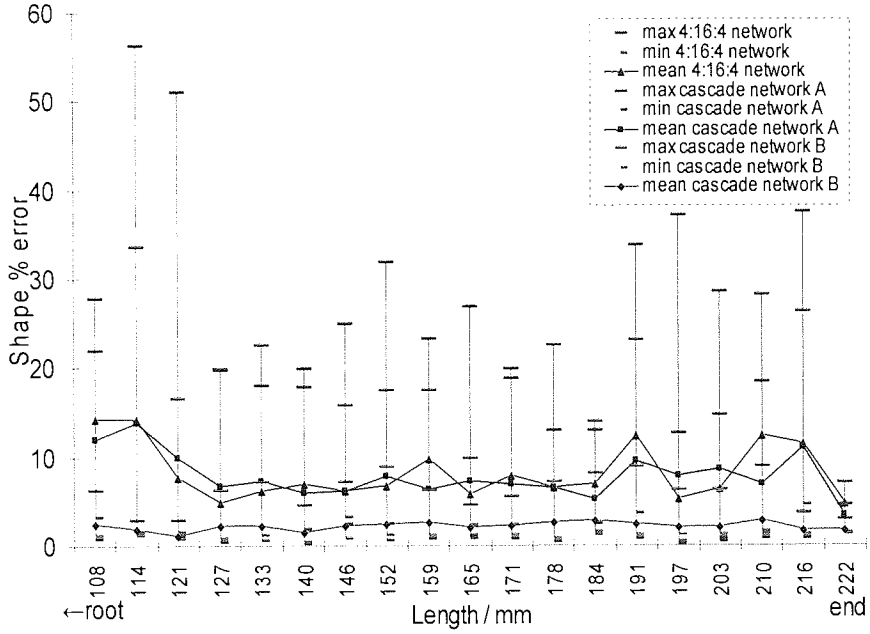


Figure 5.19: Error percentage of load shape from the bit true simulation of the Hybrid, Cascade A and Cascade B networks.

From Figure 5.16 it can be noted that the mean load percentage error for both Cascade A and B tends to be almost steady throughout the beam at about 1.5%, although a small increase at the tip is exhibited. For the Hybrid design almost steady error, at about 4%, was also shown within the range of 45% to 80% of the beam from the root, however the mean error for this design started to increase when in the range of 80% to 95%. Similarly, however, the error was at its peak at the tip of the digit.

For the position data (refer to Figure 5.17), a similar response with respect to the load mean error was exhibited by both Cascade A and B approximately 1.2%, such that the errors of both were also at their peak at the tip of the digit. However, Cascade A tends to overlap Cascade B with some degree of error difference of 0.5%. For the Hybrid design, unsteady mean error was exhibited throughout the working range. The maximum mean error scores were at their peak at 55%, 80% and at the tip. The mean error rates were better at 45%, 70% and 92% of the beam. The odd behaviour noted here is due to the effect of the sensor position, which has the most influence on the discrimination of load position.

For the width (refer to Figure 5.18), an almost steady mean error of about 1.5% was exhibited by Cascade B throughout the working range, by the Hybrid of about 3.5% between the range of 45 % to 89 %, and by Cascade A of about 2% between the range of 45% to 94%. Cascade A shows its highest mean error of rate of about 6% at the tip of digit, whereas the Hybrid demonstrates its lowest at about 1.7%. Again Cascade A tended to overlap Cascade B with some degree of error difference of about 1%. In terms of shape (refer to Figure 5.19), the Hybrid and Cascade A tended to produce higher mean error scores of about 14% towards the root of the working range. For Cascade B the mean error was steady throughout and was better than the Hybrid and Cascade A.

By evaluating these results, it is clear that the mean error rates from the Hybrid are better than 8.57% for load, 4.07% for load position, 5.54% for load width and 14.20% for load shape respectively. For Cascaded network A, the mean error rates are better than 3.67% for load, 3.63% for load position, 5.70% for load width and 13.91% for load shape. Finally, for Cascade network B, the mean errors rates are better than 3.67% for load, 2.60% for load position, 2.49% for load width and 2.9% for load shape. Overall, these results demonstrate the superiority of Cascade network B over the Hybrid design at almost any point along the working range. Comparing Cascade networks A and B, the two give the same accuracies for load at any distance along the working range (because of having equal weights and biases parameters). However, for load position and width, the error rates of Cascade network A started to increase, especially near to the tip of the digit. The error then worsens for Cascade A for the shape discrimination at any way along the range. On the basis of these findings it was concluded that optimisation using bit true simulation is a more effective and influential strategy for the digital cascade network. By considering the cost to obtain high accuracies from network complexities, more favours are put on the Single and Cascade B for implementation.

5.7 Implementation

To validate the designs, the single neural network from the Hybrid design and the cascaded neural networks from Cascade network B were synthesised, implemented and generated into a target part Virtex-II 2XCV3000, following the flow process as described in Chapter 4. The design operated at a system clock frequency of 64MHz. The digital resources required for the Hybrid and Cascaded design are tabulated in Table 5.3. Figure 5.20 to 5.23 show a comparison of the performance of the real-time Hybrid design and Cascade network B when measuring load magnitude, position, load width and load shape using signals from real sensors on the cantilever beam.

	Hybrid	Cascaded B
Occupied Slice	31%	39%
4 inputs LUTs	20%	24%
Bounded IOBs	7%	7%
MULT18X18s	51%	60%
Peak memory	218 MB	239 MB
Total Gate count	283 K	344 K

Table 5.3: Digital resources for the hybrid, and Cascaded design.

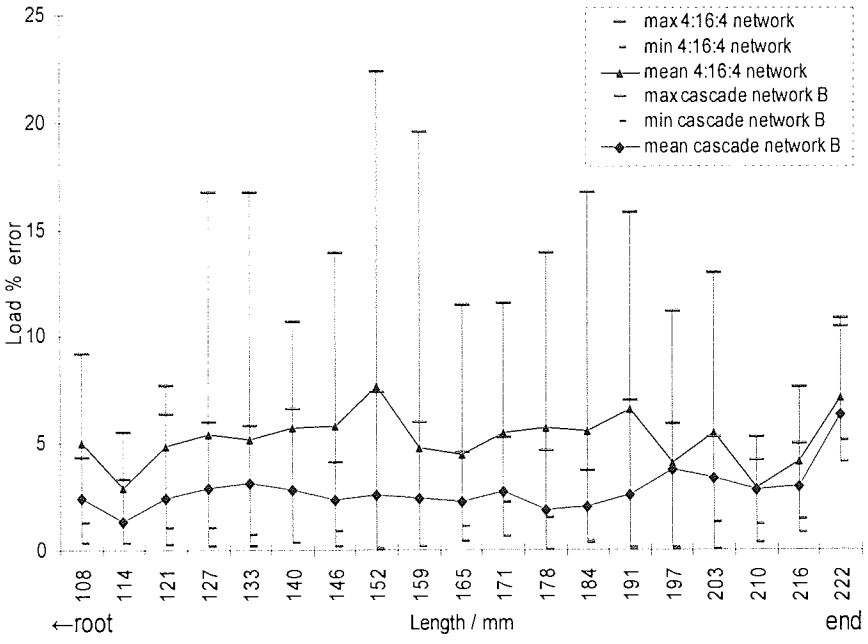


Figure 5.20: Error percentage of load magnitude from real time tests using Hybrid and Cascade network B.

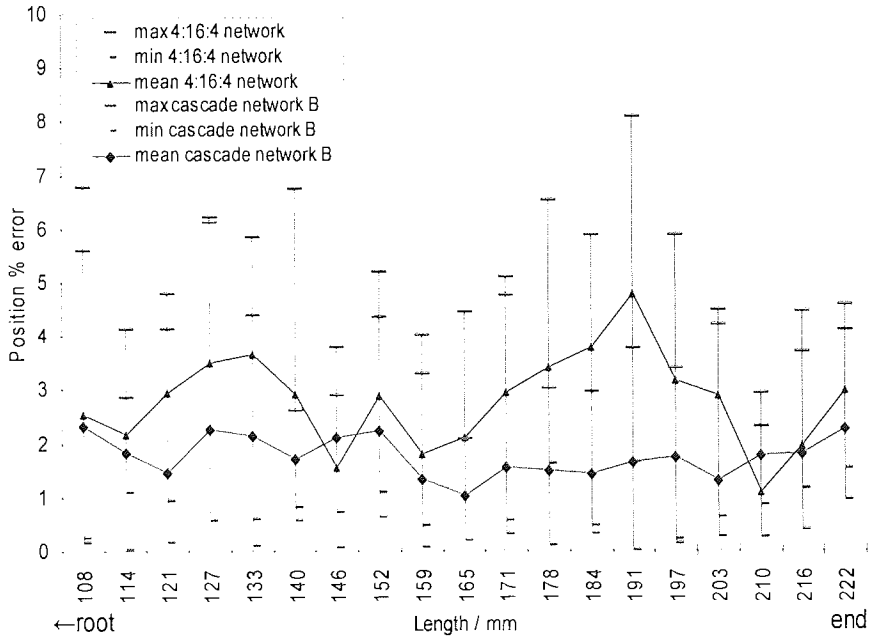


Figure 5.21: Error percentage of load position from real time tests using Hybrid and Cascade network B.

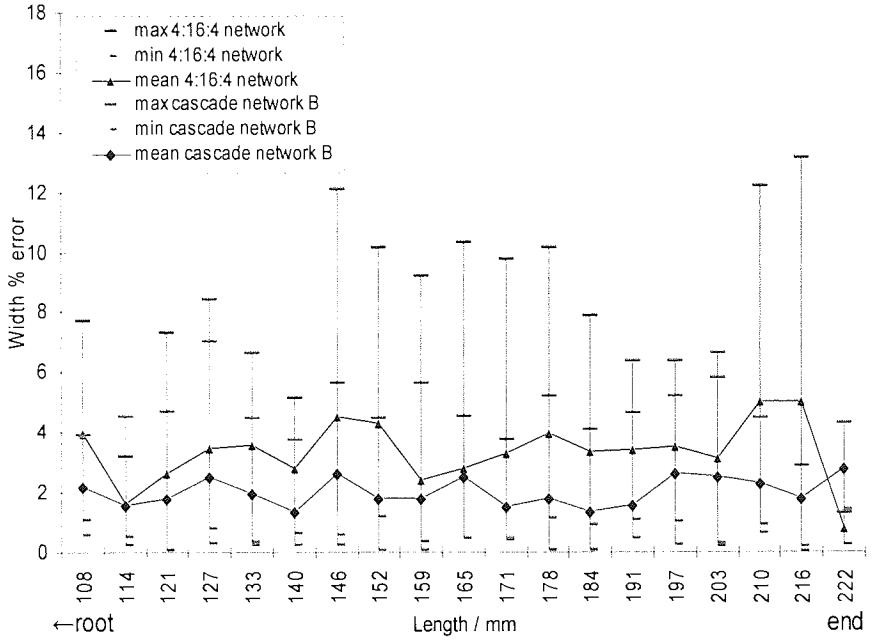


Figure 5.22: Error percentage of load width from real time tests using Hybrid and Cascade network B.

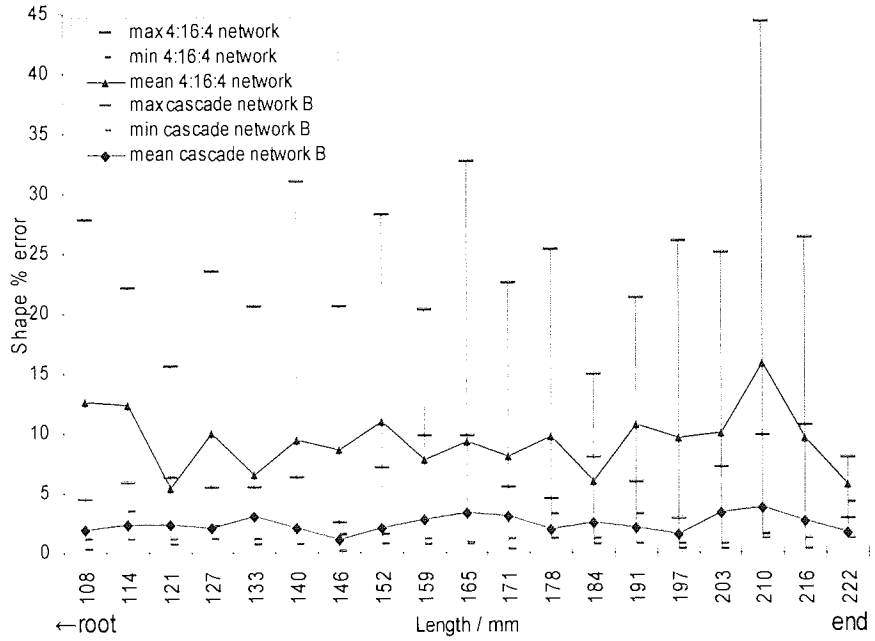


Figure 5.23: Error percentage of shape from real time tests using Hybrid and Cascade network B.

Overall, Figure 5.20 to 5.23 display similar characteristic as Figure 5.16 to 5.19 for the Hybrid and Cascade B design. Further analysis reveals that the mean error for the Hybrid design the error is better than 7.71% for load, 4.78% for load position, 5.03% for load

width and 15.83% for load shape. For the cascaded network, the mean errors are better than 6.38% for load, 2.32% for load position, 2.78% for load width and 3.77% for load shape.

The real time implementation results confirmed the selection of the cascaded technique over the Single neural network. From the overall error measurement, it is clear that both networks are more robust when discriminating load and load position. However, for the width, the networks tend to produce a higher error for smaller widths. To see the values for different widths, the local error equations (3.15) and (3.16) were used for evaluating the width error. They can be modified into a global error equation by dividing the numerator factor by the actual output (similar to load and shape error computation). Figure 5.24 was produced in this fashion, showing the accuracies of different widths for the two designs. It was noted that the Single neural network works well with widths higher than 38.1mm, whereas for the Cascade this is 25.4mm. This finding demonstrates the higher sensitivity of the Cascade.

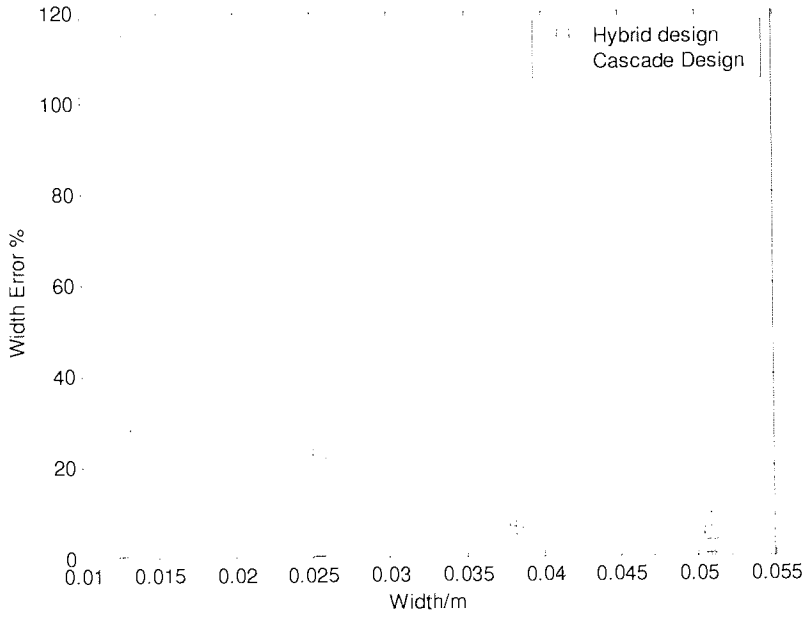


Figure 5.24: Width errors for different applied width.

5.8 Implementation of Continuous Function for Neural network Visualisation

The final research to be described in this chapter is the implementation of the continuous function explained in Chapter 3 into the targeted FPGA. This enables the outputs of the Cascaded B neural network to be visualised in real time. Two equations were presented in Chapter 3, equation (3.17) and (3.18). However, the implementation uses equation (3.18) because of its more implementation-friendly algorithm when compared to (3.17). There are no iterations and no digital divisions involved, which are advantageous for the digital design. Most importantly, there are no mathematical functions, such as trigonometric, which are inevitably digital resource hungry. The only challenge to the equation is to devise a technique to implement the mathematical power function, but with minimal use of multipliers. This was achieved by applying the 'folding' technique.

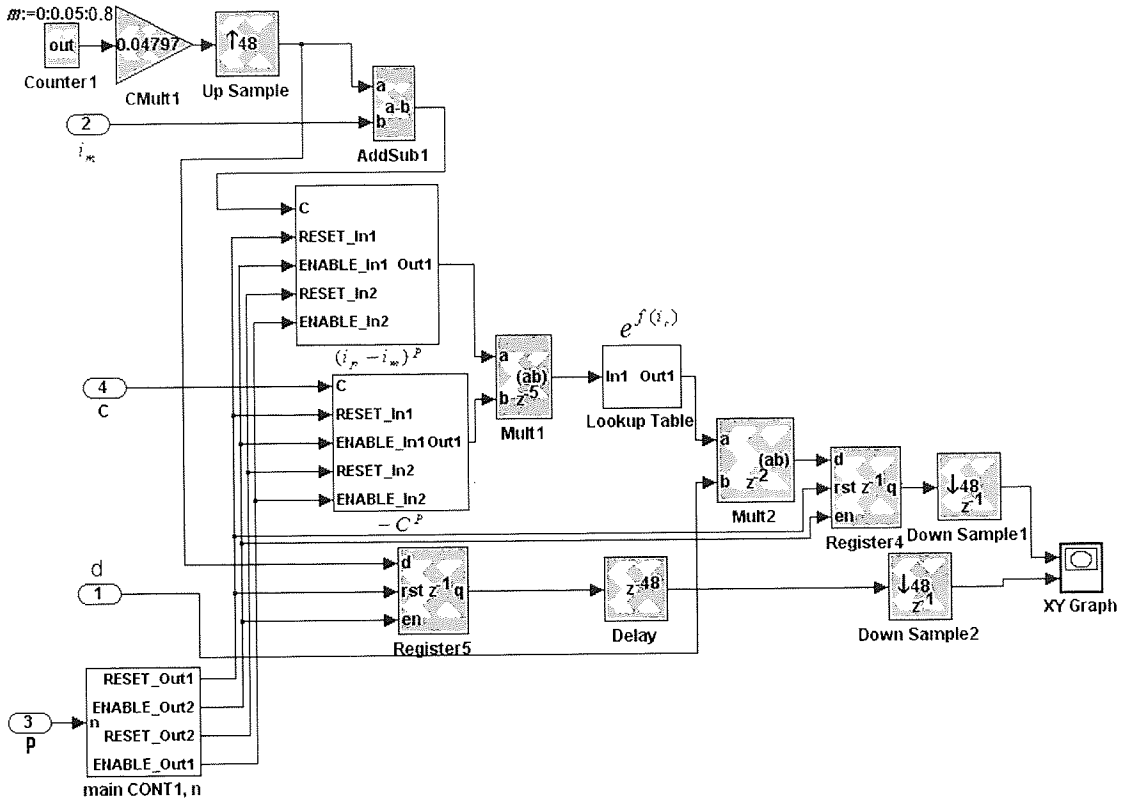
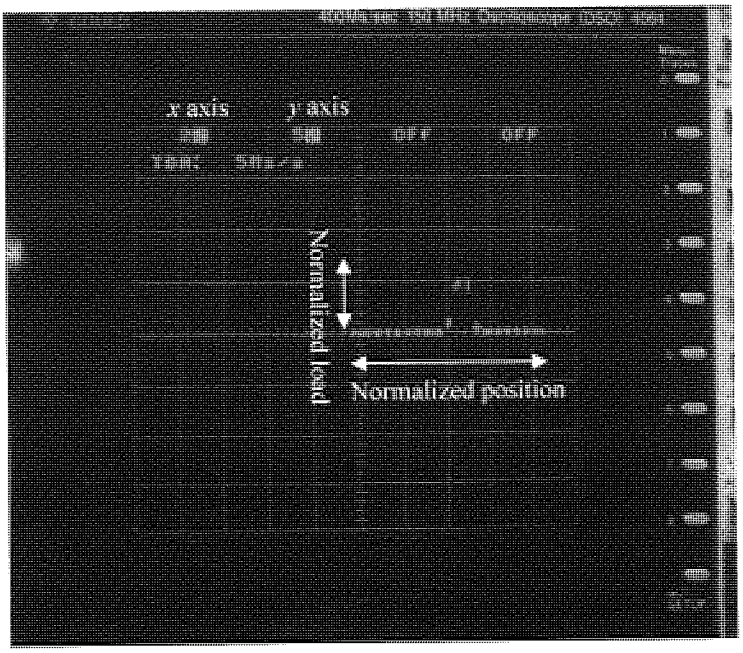


Figure 5.25: Digital design for the visualisation of the outputs.

Figure 5.25 shows the digital design used to realise equation (3.18). The design takes the discriminated load magnitude d, load position i_m, load width index C, and shape index P from the outputs of NN_A, NN_B, NN_C and NN_D of the Cascaded network respectively. A counter and a gain are used to generate a 'ramp' output of 0 to 0.8 with an increment of 0.05. This incremental output provides input variables for computing the equations. Two subsystems were designed to compute the $(i_p - i_m)^P$ and $-C^P$. As P is an integer, the power computations can be accomplished by multiplying the inputs with their outputs repeatedly for P times in sequence. Only a single multiplier is required in each of the subsystems. A register with reset and enable properties is used to extract the required output and bring it to the next process. The outputs from the two subsystems are then multiplied before being fed into a lookup table which performs the exponential function. The output is then multiplied with the load magnitude, d. Together with the incremental inputs, the two synchronised outputs can be displayed using the x-y facility of an oscilloscope. Figures 5.26 to 5.29 show typical outputs for loads C, B, A and H

(refer to Figure 3.20 and table 3.1) at the normalised position 0.5. The index P describing the rectangular shape and triangular shape is six and two respectively. Note that the probes used have x10 magnification.



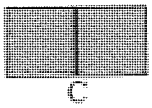
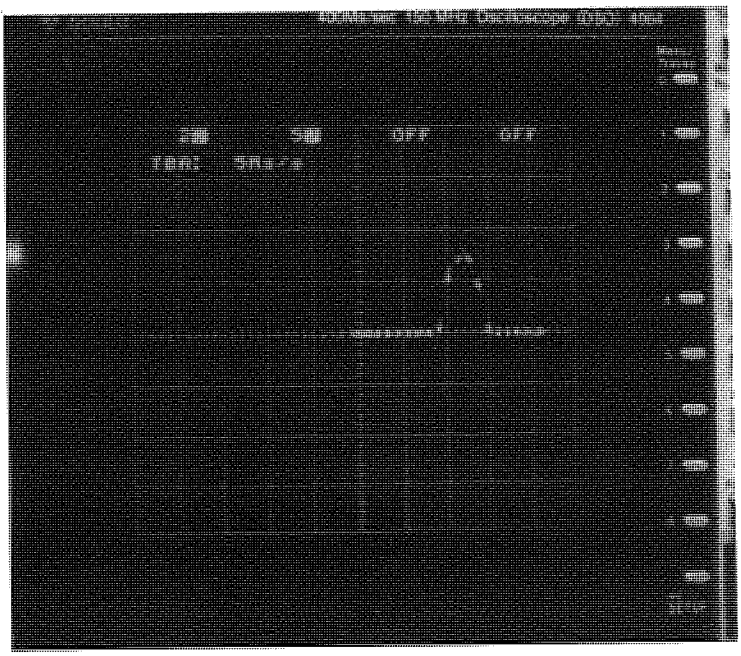


Figure 5.26: Load C with *d* of 0.431.



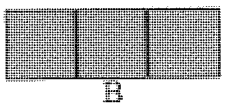
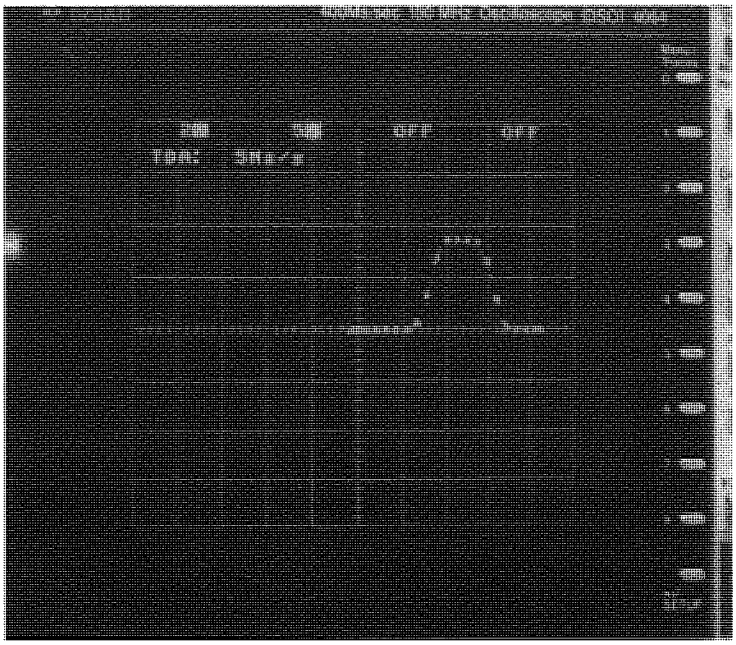


Figure 5.27: Load B with *d* of 0.744.



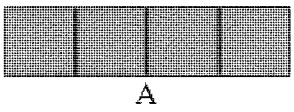
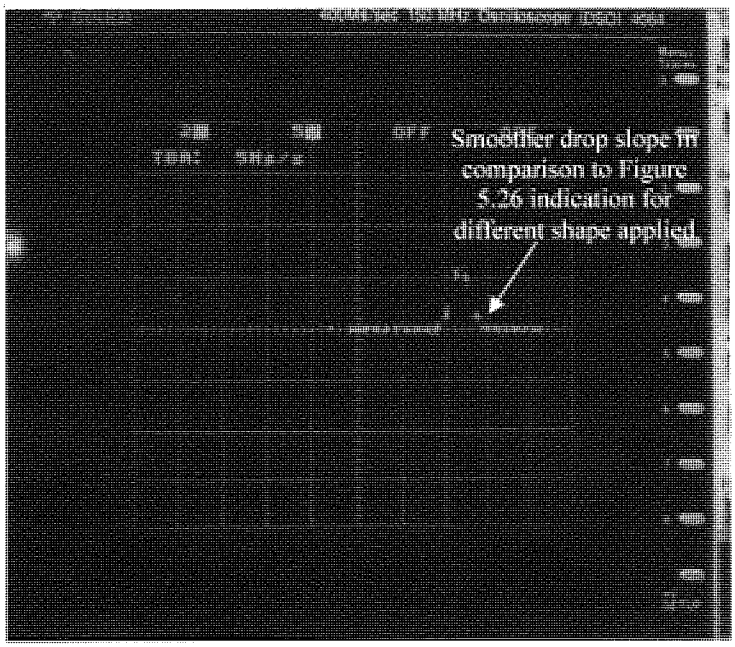


Figure 5.28: Load A with *d* of 0.9.



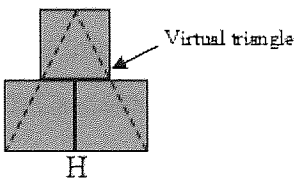


Figure 5.29: Load H with *d* of 0.626. Notice the slope of H is less sharp than C, thus discriminating the triangular shape of H and the rectangular shape of C.

5.9 Summary

This chapter has described the implementation procedures undertaken for the different kinds of neural networks discussed in Chapter 3. The first implementation was that of the Fully parallel design for a single neural network, incorporating the different proposed activation functions devised in Chapter 4. Overall, it was found that the Padé approach outperformed the Gradient scheme, the Polynomial approach and the Look-Up table. However, because of the iterative nature of the regression formula used in the Padé scheme, a faster, simpler topology such as that of the Gradient approach has influenced the decision to use it. The Hybrid (folded) design was then introduced in order to solve the problem of the high resource demand in the Fully parallel design. It was found that the Hybrid (folded) can effectively reduce the digital resource demand, such as that of the digital multiplier, by up to 75%. The modified gradient was used to complement this function. With the advantages of the Hybrid design, implementation of the network comprising cascaded neural networks can be accomplished. In this chapter two types of training of cascaded networks were implemented, known as Cascaded network A and Cascaded network B. These differed in terms of the way they were optimised. Cascaded A utilises the explicit model of the feed forward neural network algorithm during training, whereas the Cascaded B uses the bit true model. The results generated by the Hybrid, the Cascade A and the Cascaded B were compared. Through simulation and implementation (see Table 5.4) it was demonstrated that a cascade neural network with Cascaded B is the most appropriate design for producing high accuracy, followed by Cascade A and then the Hybrid design. However, the drawback of this system is the cost of the training. The neural network with Cascade B demands a longer time period to accomplish simulation in comparison to Cascade A and the neural network the with Hybrid design. The easiest design to train is the Hybrid. This time cost issue is the problem with using Cascade B, although in the context of the application which has been presented in Chapters 3 through 5, it was not a significant problem and is feasible for this technique to be employed. However, it would be an issue if the number of data sets to be simulated was higher (such as in the application in the next chapter, which uses up to more than 4000 input samples).

	Simulations			Implementations	
Load ↓	Hybrid	Cascade A	Cascade B	Hybrid	Cascade B
magnitude	8.57%	3.67%	3.67%	7.71%	6.38%
position	4.07%	3.63%	2.60%	4.78%	2.32%
width	5.54%	5.70%	2.49%	5.03%	2.78%
shape	14.20%	13.91%	2.9%	15.83%	3.77%

Table 5.4: Mean error rates for load parameters test obtained form simulations and implementations. It is important to note these performance are derived from the concurrent output of the neural networks.

To complement the neural network outputs, the continuous function explained in Chapter 3 was adopted to give real time visualisation using a scope. From the implementation it has been shown that the visualisation is satisfactory, though the P index discriminating the slope for different shapes still needs attention, due to problems implementing the power function in digital design.

CHAPTER 6

Hardwired Tactile Sensing for a Dynamic System

6.1 Introduction

The application of the embedded distributive sensory processing system which has been presented throughout the previous chapters was for the almost static case of interpretation in a one-dimensional sensing context. In this chapter the system was adopted for application into a dynamic interpretation setting using a different form of distributive sensing incorporated to suit a two-dimensional sensing system. For this process, a steel surface platform and a mechanical swing were built and used as test objects. The platform was selected such that its dynamic response was sufficient to react to the quasi-steady displacement of sinusoidal disturbances applied during experiment. The fundamental natural frequency of the plate was computed to be 53Hz (Szilard, 1974). The purpose of the swing was to create time variant disturbances prior to the motion of the attached pendulum. Three proximity sensors with a distributed arrangement were incorporated beneath the steel surface to measure deformation changes of the steel plate.

However, in contrast to the previous application, acquiring the input/output data is not straightforward. For this, a tool which can capture synchronically the motion and the corresponding sensor output– the *Vicon system* needed to be used. With the aid of computer software, the motion in terms of displacement in the translational of $\bar{x}, \bar{y}, \bar{z}$ direction in the configured reference volume exhibited by the swing can be extracted. Together with the sensor information, they were used for the training of the neural network. The aim of this chapter is to present the investigation into the sensitivity, performance and the robustness of the general purpose embedded sensory processing solutions when applied to such an application. As this approach can bring a solution to high performance need, dynamic discrimination was not the issue. The results of the experiment revealed that the system can identify the three dimensional motion of the swing in contact with the surface. This demonstrates its potential for discriminating motion in human subjects, with associated applications in sports, and posture and balance measurement (DiFabio and Faudriat, 1996).

6.2 The Rig

6.2.1 The Surface Plate

A two dimensional surface rig was designed as shown in Figure 6.1. The objective of the design was to provide a general-purpose platform that could be used for measuring the activity or motion of a subject, when making a contact disturbance on the surface. The rig was constructed using two surface steel plates (the outer and the inner plate) and a rigid steel frame. Both plates have a thickness of 2.8mm. The outer plate rests on the steel frame, which is made of hollow rectangular steel tubing of dimensions 910mm by 550mm. This is used to support the plate and raise it up to a suitable height of 60mm. This outer plate is clamped to the frame along the long sides so that the position of the plate is securely fixed. The advantage of this method is that the system is more rigid, and thus more reliable in terms of producing repeatable sensing output. The drawback is that

a sufficiently high operational load is needed to give good deflection sensitivity. The inner plate has dimensions of 400mm by 800mm, and was used to mount proximity sensors underneath. For the ease of reconfiguring the sensors, this plate is not physically attached to the rest but was simply laid on the ground below the frame. The surface rig can support the weight of a normal person, hence enabling a person of average weight to stand on and produce a deflection of a range between 1 to 2mm, an operating range suitable for the deflection sensors to detect.

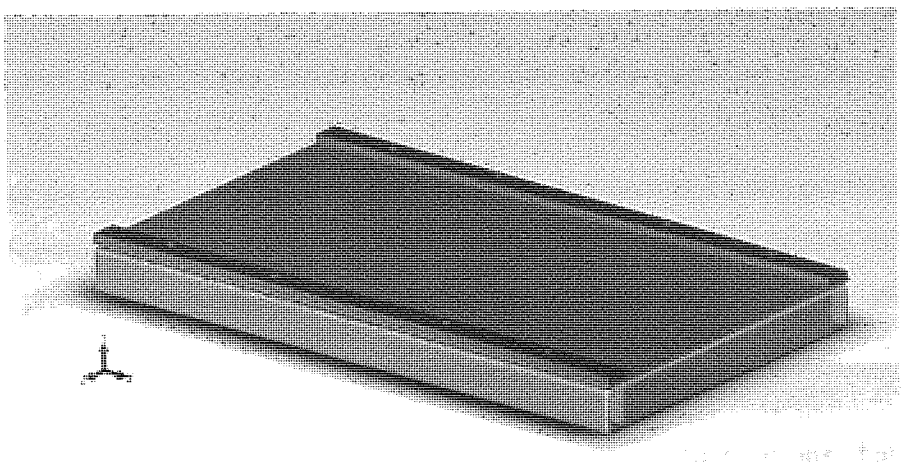


Figure 6.1: The surface plate model with two opposite edges clamped and two edges simply supported.

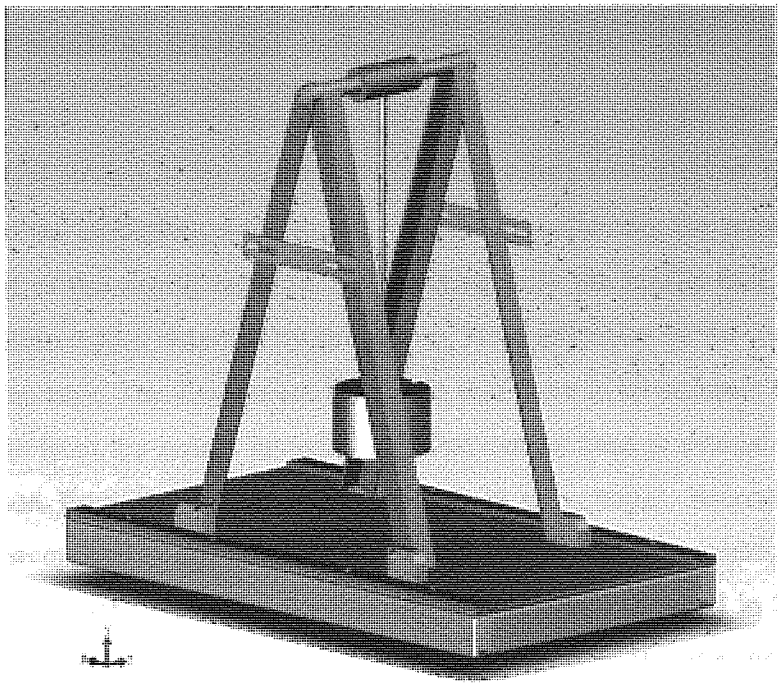
6.2.2 The swing

A swing was designed and constructed as shown in Figure 6.2. The swing is comprised of four legs, of which each pair form an 'A' shape frame. These are used as side frames. The frames are bolted together using an axial steel rod with a fixed length of 350mm (which also defined the length between the feet of the opposite frame). The axial road is used to support a one directional movement load pendulum, which has a length of 480mm from the centre to the axis of the axial. The pendulum is able to hold standard stackable metric weights of 5kg and 10kg. Each leg is 720mm in length. The angle between the legs and

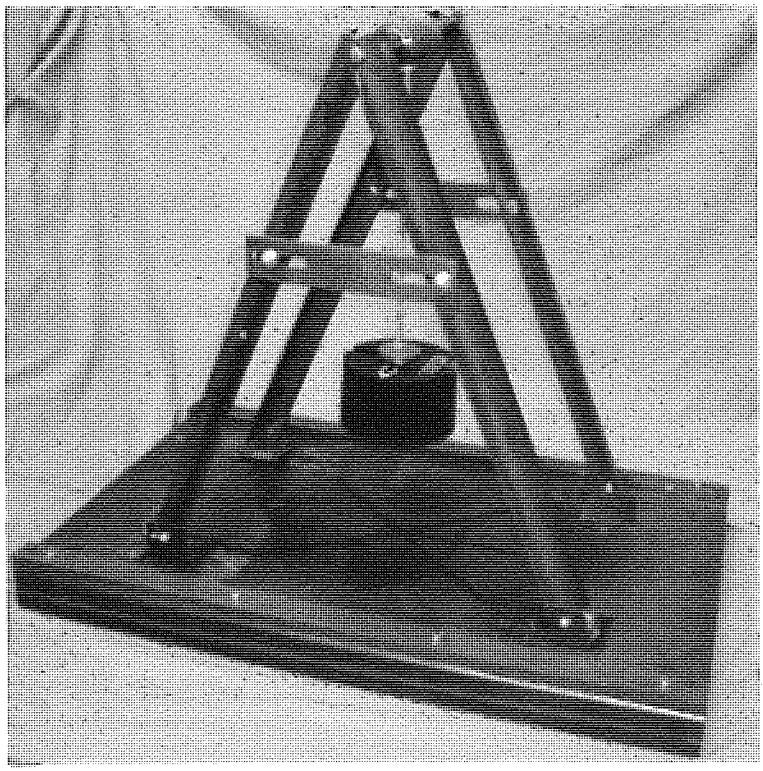
the height of the load are both adjustable to enable different configurations of the swing. The feet are free to rotate, so that if the angle is altered they always sit flush to the surface of the plate. In the experiments the width of the frames were fixed such that the distance between the frame' feet was 600mm, thus making the angle between the legs to be 49.25 degrees.

The objective of the swing is to generate translational movement of a load in three perpendicular axes, namely forward/backward, up/down, left/right. The first refers to the sway movement of the load in $\vec{y}$ direction, the second is the vertical displacement in $\vec{z}$ direction, and the last is the lateral movement due to the wobbling of the swing in $\vec{x}$ direction. In comparison to $\vec{y}$ and $\vec{z}$, $\vec{x}$ will be much smaller in magnitude. The rationale of the swing is to provide a pilot study for a system at a static position, but with dynamic criteria such as that required when measuring the sway of a human subject. This tool thus provided a means of studying the performance and sensitivity prior to the motion.

In practice, the swing has a pendulum weight of 10kg, which means that the plate deflection was relatively small. However, it was found that the sensing elements could still detect this deflection without being swamped with noise.



(a)



(a)

Figure 6.2: (a) The model and (b) the real version of the swing and the platform.

6.3 Mathematical model

The mathematical models of the swing and the plate system were derived. In this research the objective of the modelling is to provide a means for validating the sensor output used for the training of the neural network, by verifying only the profile and characteristics with the deflection output obtained by simulation. No verification on the exact magnitude response is done here. The first reason for this is that there is no accurate transformation model available from the sensor manufacturer that can be used to evaluate precisely the actual deflection from the sensor output (which is in voltage). Secondly, the signal conditioner system used in this research only produces the AC component, but filters out the DC information (the reasoning for this will be explained in detail in Section 6.4.2). Thus verification can only be done by comparing directly the profile and the characteristic, for example how the output should behave if the angle of the pendulum is at some displacement of the load, but inductively for the magnitude. With the establishment of the surface and the swing model, four conditions were made to support the validation hypothesis.

Condition 1: Providing that the operating range of the deflection is within the linear range of voltage against the distance characteristic (see Figure 6.9) of the sensor, the output from the sensor should always be directly proportional to the actual deflection.

Condition 2: Providing that the mathematical model of the surface and the experimental measurement is verified, simulated deflection made from any range of forces (load) subjective to the verification should also be true.

Condition 3: If the model of the swing is true, following *Condition 2*, the simulated deflection made by the swing foot at any position on the plate should also be true.

Condition 4: Following *Condition 3*, regardless of the exact magnitude of the simulated deflection, if *Condition 1* is true and the profile and characteristic of the simulated

surface deflection relative to the swing model is matched with the sensor signal response, the validation of the sensor output should be true.

6.3.1 The plate model

The surface plate used in the investigation can be categorised as a rectangular plate with edge conditions, specifically a rectangular plate with two opposite edges simply supported and the other two edges clamped. Obtaining the deflection model of such a plate involves derivation using the deflection for the case of simply supported plate. Equation (6.1) shows the general expression of the simply supported surface plate (Timoshenko & Woinowsky-Krieger, 1959) used to demonstrate the deflection ω_{SS} of the plate at any point in the coordinates of $\vec{X}' = \vec{x}'$ and $\vec{Y}' = \vec{y}'$ when a load D is applied at coordinate $\vec{X}' = \xi$, $\vec{Y}' = \eta$ of the two dimensional system (see Figure 6.3(a)).

$$\omega_{SS} = \frac{Da^2}{\pi^3 P}\sum_{m=1}^{\infty}[1 + \beta_m \coth\beta_m - \frac{\beta_m \vec{y}_1'}{b}\coth\frac{\beta_m \vec{y}_1'}{b} - \frac{\beta_m \eta}{b}\coth\frac{\beta_m \eta}{b}] \frac{\sinh\frac{\beta_m \eta}{b}\sinh\frac{\beta_m \vec{y}_1'}{b}\sin\frac{m\pi\xi}{a}\sin\frac{m\pi\vec{x}'}{a}}{m^3\sinh\beta_m} \qquad (6.1)$$

For which $\beta_m = \frac{m\pi b}{a}$, $\vec{y}_1' = b - \vec{y}'$ for $\vec{y}' \geq \eta$, and $\vec{y}_1' = \vec{y}'$ and replacing η by $b - \eta$ for $\vec{y}' < \eta$

$$P = \frac{Et_\tau^{\,3}}{12(1-\upsilon^2)}$$

Where P here is flexural rigidity of the plate, E is elastic modulus, t_τ is the surface thickness and υ is Poisson's ration.

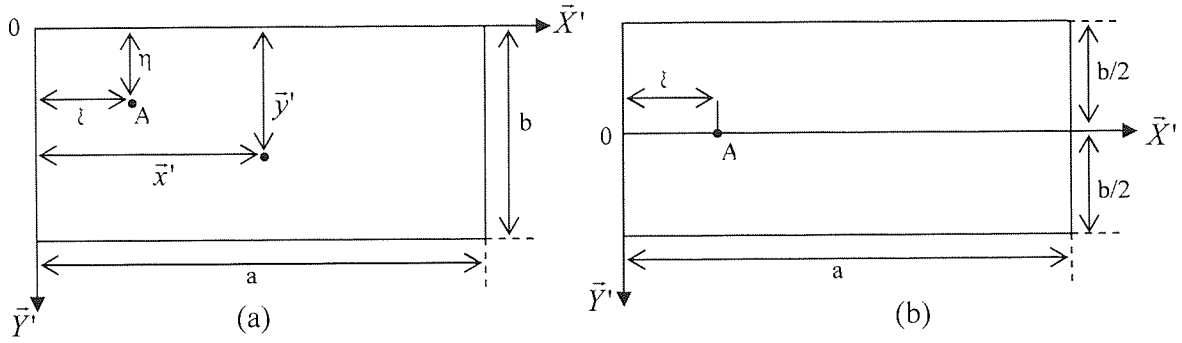


Figure 6.3: Plate parameters for simply supported edges.

For the case of load D concentrated at a point A on the axis of the symmetry of the plate (see Figure 6.3(b)), the general expression shown by equation (6.1) for the deflection of the plate can be simplified into equation (6.2) (Timoshenko & Woinowsky-Krieger, 1959) by substituting η= b/2. The condition is considered to be more relevant for this situation.

$$\omega_{SS} = \frac{Da^2}{2\pi^3 P}\sum_{m=1}^{\infty}[(1+\alpha_m \tanh\alpha_m)\sinh\frac{\alpha_m}{b}(b-2\vec{y}') - \frac{\alpha_m}{b}(b-2\vec{y}')\cosh\frac{\alpha_m}{b}(b-2\vec{y}')]\frac{\sin\frac{m\pi\xi}{a}\sin\frac{m\pi\vec{x}'}{a}}{m^3\cosh\alpha_m} \qquad (6.2)$$

Where $\alpha_m = \frac{m\pi b}{2a} = \frac{\beta_m}{2}$

The deflection of the rectangular plate representing the actual rig can be obtained by first solving the problem on the assumption that all edges are simply supported, employing equation (6.1) or (6.2). Then bending moments are applied along the tentative fixed edges (where in this case is along +b/2 and –b/2). In other words, the contribution of a deflection due to the action of the moment along the fixed edges should be removed. Thus, by considering equation (6.2), superposing the deflection of a simply supported plate and the deflection produced by moment distributed along the clamped edges, a new deflection ω_{SF} which meets the condition of the surface rig can be obtained, equation (6.3) (Timoshenko and Woinowsky-Krieger, 1959). Note that the first sum in the expression corresponds to the rig if the clamped is removed, whereas the second sum is the

deflection due to the moment along the edges. Detail of the steps explaining the derivation can be found in Timoshenko and Woinowsky-Krieger (1959) (page 180 to 191).

$$\omega_{FS}=\frac{Db^2}{2\pi^3 P}\left[\frac{a^2}{b^2}\sum_{m=1}^{\infty}\left[(1+\alpha_m\tanh\alpha_m)\sinh\frac{\alpha_m}{b}(b-2\bar{y}')-\frac{\alpha_m}{b}(b-2\bar{y}')\right.\right.$$
$$\left.\left.\cosh\frac{\alpha_m}{b}(b-2\bar{y}')\right]\frac{\sin\frac{m\pi\xi}{a}\sin\frac{m\pi\bar{x}'}{a}}{m^3\cosh\alpha_m}\right]-\frac{\pi^2}{4}\sum_{m=1}^{\infty}\frac{1}{m}\frac{\sin\frac{m\pi\xi}{a}\sin\frac{m\pi\bar{x}'}{a}\tanh^2(\alpha_m)}{\sinh\alpha_m\cosh\alpha_m+\alpha_m}\Bigg] \quad (6.3)$$

Verification of the plate model was undertaken by comparing the output deflection obtained by the computer simulation with the deflection obtained by real measurement. In the research the simulation was done using Matlab. For the real measurement a clock gauge was used to measure the vertical changes of the point $\bar{X}'=\xi=$ 320mm on the symmetry axis of the plate (with respect to Figure 6.3(b)). A range of loading magnitudes was placed at the centre of the surface plate, and the measurements were taken by recording the reading of the clock before and after the loading. The difference denotes the deflection. Figure 6.4 shows the deflection obtained by the simulation and by real measurement for a series of loads from 0kg to 10kg with increments of 1kg. By comparing the results it is clear that the real deflection measurement is in close agreement with the simulated deflection, though change of stiffness to the plate was occurred at 8kg load applied. The real measurement shows an almost steady linear relationship between the loads and the deflections. On this basis it was deduced that the model is true. Thus *Condition 2* is met.

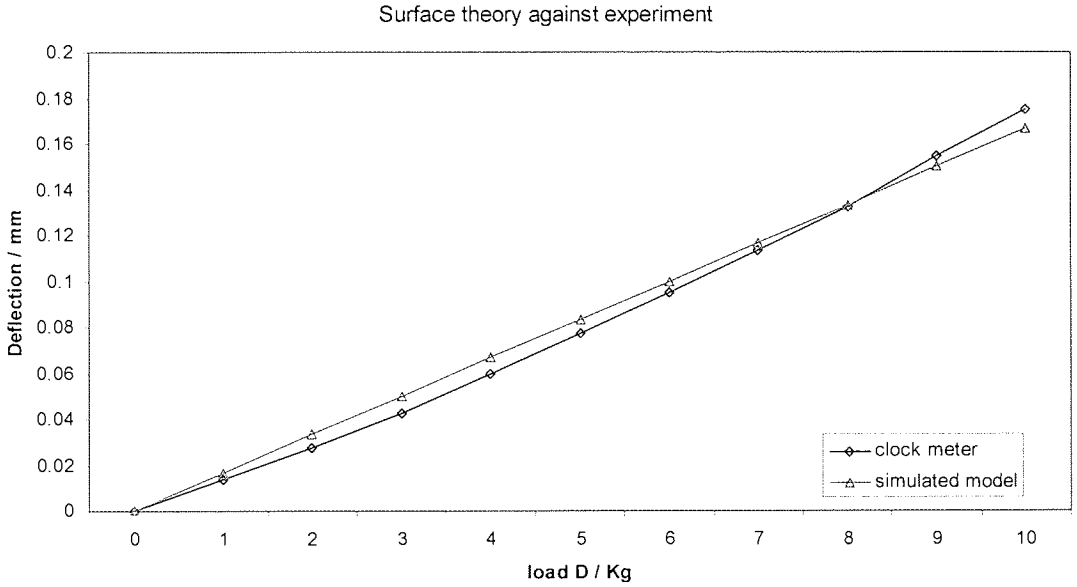


Figure 6.4: Comparative plots between the simulation and real measurement.

6.3.2 The Swing

The model of the swing is useful for verifying the serial pattern of the sensor outputs. The target of the modelling process is to evaluate the normal forces response exerted by the feet of the swing frame on the plate prior to the movement of the pendulum (see Figure 6.5). In the current research this model is derived from the mechanics of the physical pendulum (Losonc, 2003), and the final expression are presented by equation (6.4) and (6.5). (The detail explaining the derivation can be seen in the Appendix 7).

$$N_{y1} = \frac{F_y I_p \sin\psi - F_x I_p \cos\psi}{2 I_p \sin\psi} \tag{6.4}$$

$$N_{y2} = \frac{F_y I_p \sin\psi + F_x I_p \cos\psi}{2 I_p \sin\psi} \tag{6.5}$$

Where I_p is the length of leg of the pendulum, 2ψ is the angle between the leg, and F_m is tangential force. The force on the point of axis O, can be resolved into horizontal F_x and vertical F_y components.

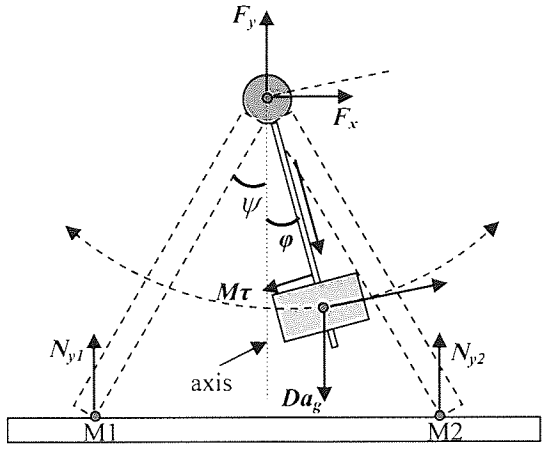


Figure 6.5: A diagram of the pendulum from the side view.

Equation (6.4) and (6.5) was simulated in Matlab, with the assumption the system is ideally symmetrical that the weight is distributed equally on each of the four feet. The simulation also taking into account damping effect and other physical weights from the swing such as the axial supports the load, and the rig. Figure (6.7) is produced to show the sub-plot of the simulation responses for N_{y1} and N_{y2}, with initial swing angle $\varphi_{initial}$ of 30 degrees. It can be observed the response obtained from normal force N_{y1} is out of phase from N_{y2} by 180 degrees.

Profile and characteristic remarks: Figure 6.7 shows the responses can be divided into two parts. Firstly is region when φ is small ($\leq\approx 10^{o}$) where the responses are represented by simple harmonic motion swing, labelled in the figure by *region A*. The second is region when φ is large ($>\approx 10^{o}$) where the responses are essentially simple harmonic motion but show signal of more complex behaviour, labelled in the figure by *region B*. It can be observed that in *region A* the system has demonstrated fully sinusoidal motion.

For *region B* the responses are essentially sinusoidal but with a 'twin peaks' effect on most of the regions of higher magnitude of forces (see circle dashed lined in the Figure 6.7).

These scenarios mainly caused by the resultant affect from horizontal force F_x which was produced from the swing (refer to Figure 6.5). When φ is small, response of forces F_x and F_y will follow the simple harmonic motion of the pendulum inherently the response of the normal forces (see equation (6.4) and (6.5)). If φ is large, non-linearity of the pendulum motion adds another harmonic to the maximum and minimum amplitude of force F_x, causing the amplitudes to saturate or even producing twin peaks if φ is sufficiently higher (see Appendix 7 for F_x when φ was varied). For F_y response still maintain sinusoidal.

The effect from F_x inherently contribute the behaviour and profile of response in *region B*. Figure 6.6 shows simulation of the forces, F_x and F_y, and N_{y2} implying measurement was taken from point M2 (refer to Figure 6.5). In Figure (6.6), force F_x was negative and this correspond to when the load swing beyond the axis (with respect to point M2), and positive when otherwise. When the load swing beyond the axis, the additional harmonic present in F_x was cancelled by F_y, thus resultant force on N_{y2} was almost sinusoidal within this duration. When the load swings in the region before the axis, F_y will not cancelled the additional harmonic present in F_x but instead will exaggerate the non-linear behaviour. Thus the 'twin peaks' effect is created.

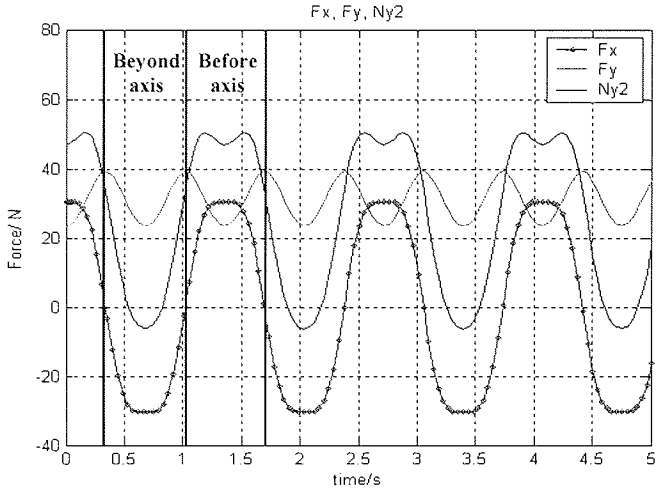


Figure 6.6: Simulations of F_x and F_y and the resultant, with angle $\varphi_{initial}$ = 30 degrees, and damping $\kappa = 0$.

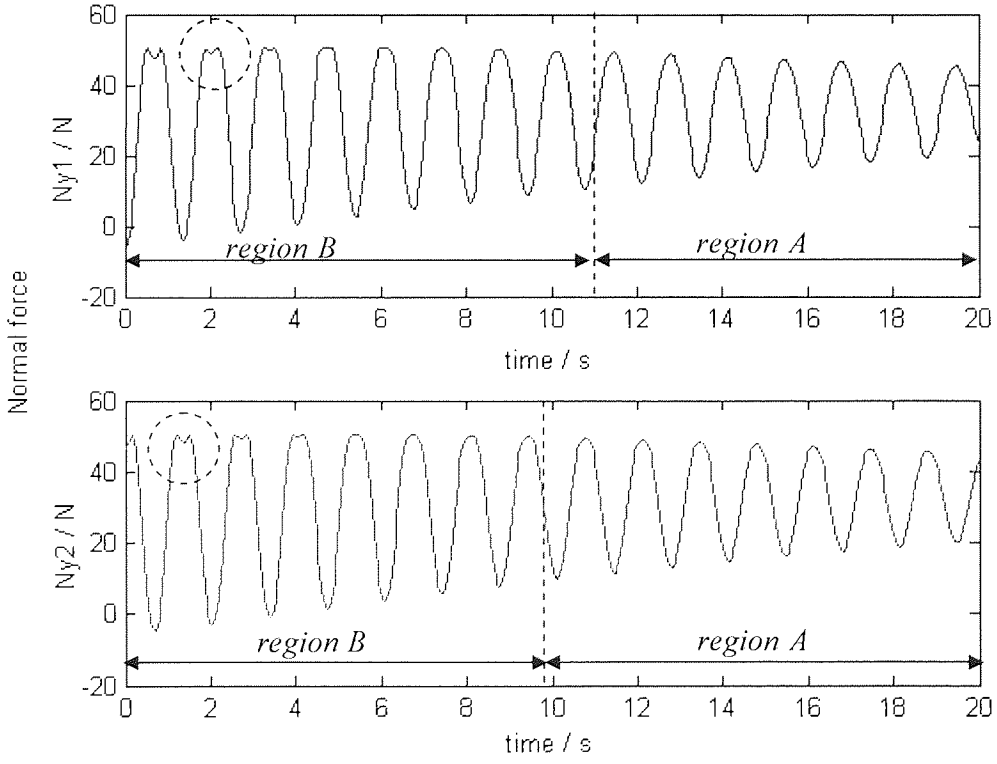


Figure 6.7: Results from simulation method using equation (6.4) and equation (6.5) with angle $\varphi_{initial}$ = 30 degrees and damping factor κ of 0.05.

6.3.3 Swing and Sensor Position

The position of the swing feet under the platform and the sensors are illustrated in Figure 6.8. Three sensors, S1, S2 and S3, were used and were positioned closely to the symmetrical axis of the plate. To get the highest coupling effect from the swing, two sensors, S1 and S3, were placed in line with the opposite feet of the frames, and one sensor, S2, placed at the centre.

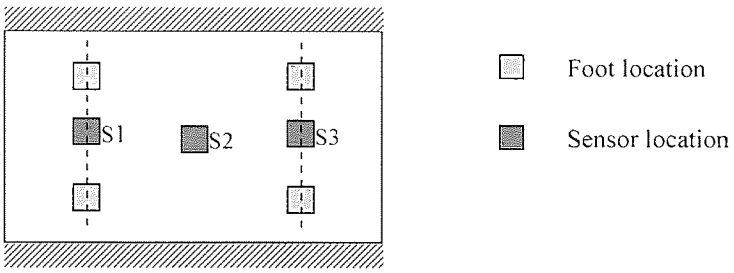


Figure 6.8: Position of the swing and the sensor on the two clamped edges surface plate (top view).

6.4 Experimental Setup

6.4.1 Sensor Setup

To measure the plate deflection, reflective sensors such as the Photo reflector SY-CR102 were used. The sensor type is a non-contact sensor. It was selected for its low cost and appropriateness for this application. The fast rise time and fall time (1KHz), make it the most suitable for dynamic measurements such that of the swing.

Each sensor has two essential mechanisms; the photo emitting diode which emits the infrared light, and the phototransistor which detects and measures the reflected light intensity. The change of the intensity changes the output voltage. Both of these

mechanisms are integrated into a compact package of dimensions 2.7mm by 3.2mm. In the operation, when there is an activity made by the swing, the plate will be deflected. The deflection changes the distance between the sensor and the plate, thus changing the intensity of the reflected light. However, a suitable height setting has to be established so that the reading will be consistent with *Condition 1*. This can be done by having an operation range bounded within a linear region as shown in Figure 6.9. In the experiment 3M Scotchlite Reflective tape was attached along the point of detection to ensure maximum reflection from the emitted light.

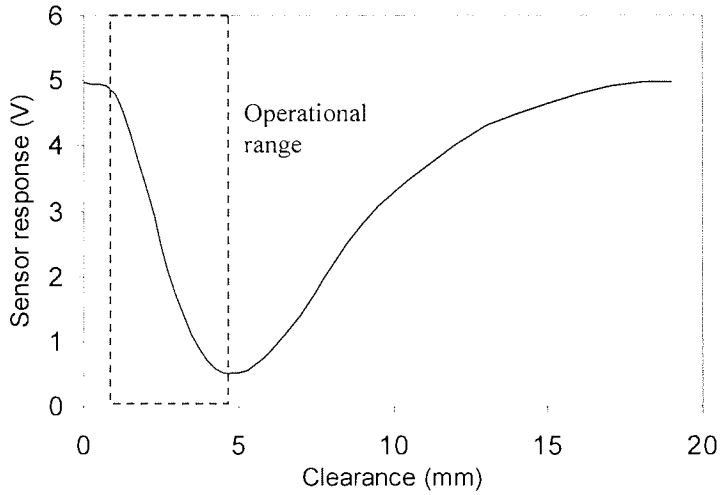


Figure 6.9: Photosensor characteristic.

6.4.2 Signal Conditioning

The objective of conditioning the signal from the sensor is to ensure that it is reliable, repeatable and compatible with the ADCs of the embedded system. With a typical 10kg swing, the output from the sensor is only able to produce a maximum excitation voltage output in the range of 10mV to 60mV from the *raw* sensor, but a very high D.C offset voltage (relative to the changes) which is in the range of 3.5V to 4.7V. The first procedure in the signal conditioning process was to remove the DC offset and amplify the

signal to a suitable range, $V_{diff} = G(V_{in} - V_{ref})$. With the measurement made from the raw sensors, a gain G of 10 for sensor S1 and S3, and 80 for S2, were used for the amplifications. As for the first attempt, a differential amplifier circuit with the selected gain configurations was designed to take the single ended inputs from the sensor and difference it with the suitable reference input voltage to give a steady zero output. This is relative to when the swing is not yet mounted on the platform. However, the technique was very fiddly to set up and was also very susceptible to drift. Because of the gain even an unsteady drifting voltage as small as 0.1V will be amplified to 1V, and worsens if G is 80. The problem can be solved by introducing a high precision reference voltage input, but as it should have 'adjustable criteria', constructing such a reference input would be intricate, especially when designing for multiple parallel sensors. An alternative solution is to remove the large differential DC voltages through active filtering. A similar technique has been used for the high-precision analogue front end of a portable ECG application (analogue article, Appendix 8). An instrumentation amplifier, INA128, with a unity gain was used to take the singled ended output from the sensor. A low pass filter circuit was connected in cascade to the output, which extracted the DC offset and fed it back to the reference port of the instrumentation amplifier circuit. The DC offset free signal could then be amplified with the gain G using a separate amplifier circuit also connected in cascade. However, to ensure the technique operated correctly, the filter circuit should have a cut-off frequency significantly smaller than the frequency of the swing, thus enabling the filter to extract the relevant DC voltage. In this operation a cut-off frequency of 0.06Hz was used, a significant cut-off frequency for filtering the offset of the 0.7Hz operation swing. Pre-calculation of the correct capacitor and resistor values was required in relation to the targeted operation frequency of the swing. A resistor with a value of 560kΩ and a capacitor of 4.7μF were found to be suitable. The schematic circuit diagram is shown in Figure 6.10. This technique was more effective than the first attempts, as the output was more robust and reliable. This method is also consistent with the essence of the current work in its aim to measure the dynamic movement. The only disadvantage of this approach is that it is not appropriate for the measurement of still deflection of the stationary load. Following the signal conditioner for amplification is the

clipping circuit, which acts as the ADCs' FPGA input voltage rail-to-rail protector. Unlike the cantilever experiment, the plate can be considered a 'heavy-duty' mechanical application which is more prone to unexpected excessive load, and thus can overshoot the output easily. Upon the removal and amplification of the DC and the clipping circuit, the inputs can then be multiplexed and fed into the ADC following the same routine described in Chapter 5.

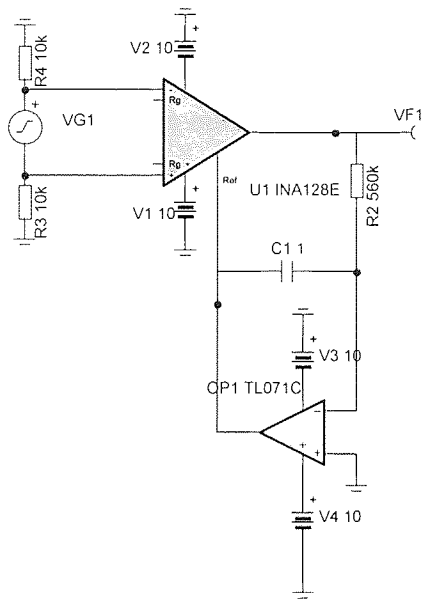


Figure 6.10: A schematic diagram of the DC remover circuit.

6.4.3 The Vicon System Setup

As detailed in the previous chapter, in order to train the neural network, sets of inputs and the corresponding outputs have to be introduced into the training process. For a static case experiment such as the cantilever beam, the input/output relation can be easily measured. But for dynamic measurement as required in this chapter, a computational aid has to be used to capture the dynamic input from the sensor as well as the corresponding physical dynamic response. Here, it was decided to use a Vicon system. Using this system the movement of an object in three-dimensional space can be captured with

theoretical accuracy of 0.5mm. The system consists of an array of twelve cameras which are positioned around the volume where the object is positioned.

Each camera has a resolution of a 1.3 megapixel CCD array and a high quality lens – a similar configuration to that of a digital camera. An array of LEDs surrounds the lens and emits light at a fixed wavelength in the near infrared (NIR) band. The subject (a person or object) is fitted with reflective markers in key areas of movement. For example, a person has the markers fitted to their joints (such as knees, elbows, etc.). The light from the LEDs is reflected off the markers onto the camera's CCD array. A filter is implemented in the camera which filters out all wavelengths of light except that of the LEDs. This means that a monochrome image is left which only contains the marker reflections.

Each camera sends the captured image back to a processing unit, which in turn sends the signal to a high powered PC. The PC uses the images (along with calibration data), to produce a calculation based on triangulation, which determines a single set of three-dimensional coordinates for each marker from the twelve two-dimensional images. The displayed markers can then be labelled and related such that a three-dimensional 'stick model' representation of the captured object is shown on the screen. The three-dimensional coordinate of any marker at any point in time can then be found. This coordinate data was used to provide a training dataset for the neural network.

The markers were positioned on the swing at key points such that a good quality three-dimensional representation could be reproduced in the motion capture software. The marker points were positioned at:

- Either end of the horizontal bar at the top of the A-frame.
- On each foot of the frame.
- The front, back, left and right sides of the lower face of the 10kg mass.
- The top face of the 10kg mass (three markers set in a triangle shape).

The resulting 3D model of the swing is shown below (Figure 6.11), along with an image of the swing in motion. The model depicts two extra markers not described above. These are virtual markers and are calculated using software scripts from the position of the real markers. Virtual marker 1 in the centre of lower surface of the 10kg mass was calculated as the centre of the four markers placed around sides of the lower face of the mass. Virtual marker 2 is on the pivot at the centre of the top horizontal bar (see Figure 6.11). The coordinates of these virtual markers are used as the reference point for the swing position in the training data for the neural network.

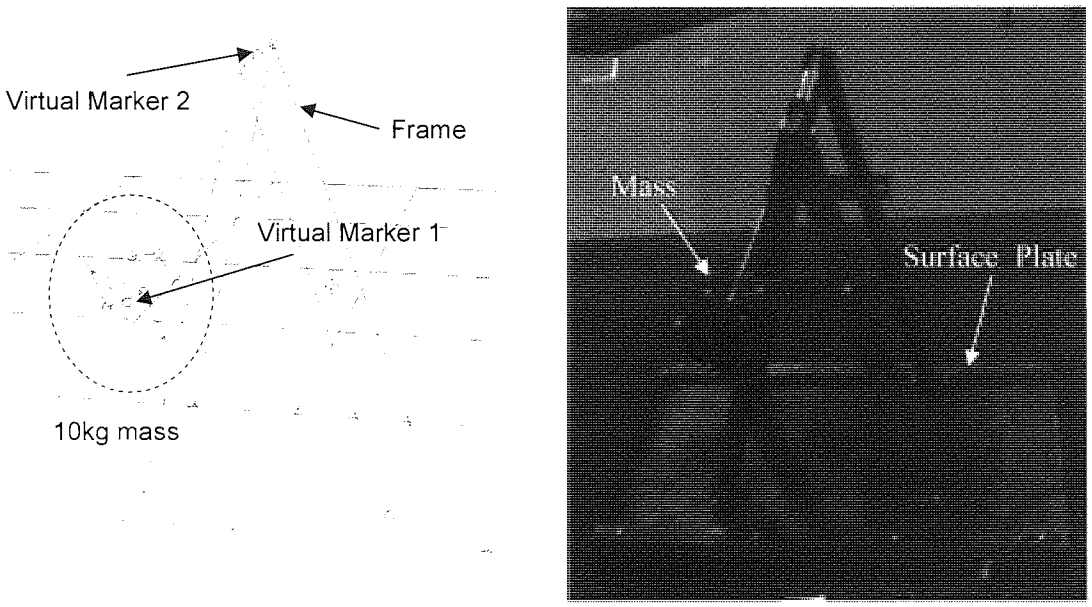


Figure 6.11: Configuration of markers on the swing.

To generate training data the sensor outputs were captured by an ADC built into the Vicon system, which is fully synchronised with the sampling rate of the cameras. The cameras and sensor data were set to sample at 200Hz for 25 seconds, thus generating 5000 samples. Each sample from the sensors had a corresponding set of three-dimensional coordinates representing the position of the swing at that point in time.

6.4.3.1 Verification of the Performance of the Vicon System

A control experiment was performed to verify the accuracy of the Vicon system. This procedure involves capturing multiple sets of readings relative to the position of the markers on the swing, and comparing the difference between them. To make the measurement more effective, the error evaluation from independent trials is imposed by considering the error measurements from different Vicon system calibrations. Two readings were captured from each calibration. A total of six separate calibrations were made to produce 12 readings. For each of the calibrations the error difference made by the first and the second readings was evaluated. The readings for different calibrations are depicted in Figure 6.12. The corresponding absolute errors with different configurations are shown in Figure 6.13. It was found that the average of the errors (green dashed line) is 0.585mm, which is in agreement with the theoretical figure of 0.5 mm. However, with this level of accuracy attention has to be given to the ability of smaller motions to be captured. This is because, for an example displacement of 5mm, theoretical accuracy of 0.5mm will generate an uncertainty of 5%, producing a significant impact on the accuracy of the measurement. Practically, one means to reduce the error is to obtain the smaller motion from the components forming the larger motion – resultant displacement of the motion. For this only a single capture is required. The components can then be derived separately.

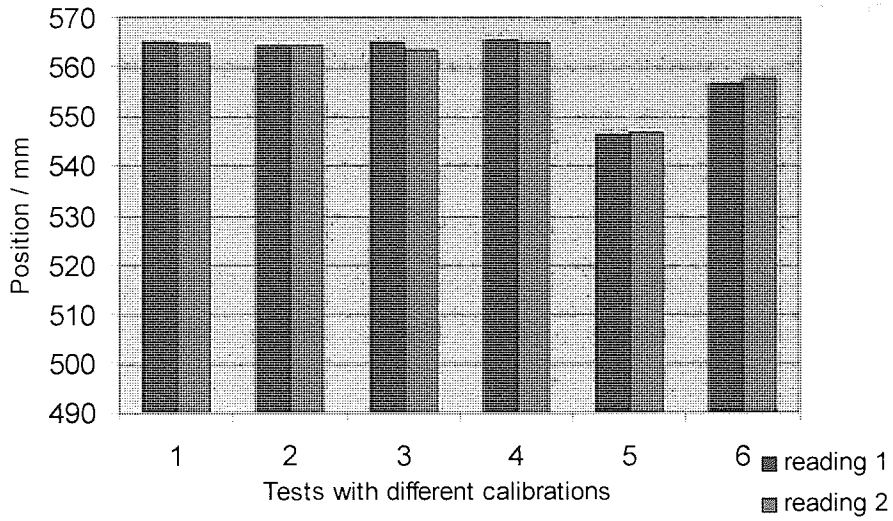


Figure 6.12: Readings for different calibrations of the Vicon system.

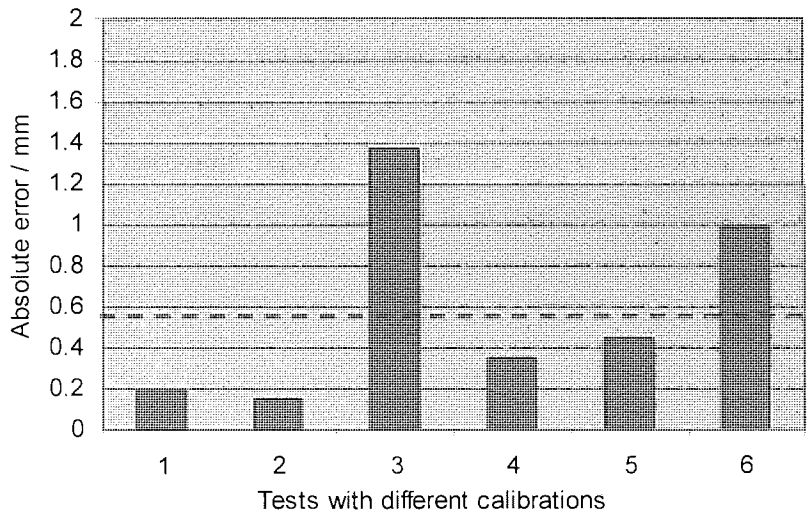


Figure 6.13: The absolute error from different calibrations.

6.4.4 Input Output Setup

Three output displacements were considered for the measurement; the displacement $\bar{y}$ of the load in the direction of the swing, the $\bar{z}$ displacement of the height achieved by the swinging load, and the $\bar{x}$ displacement from the side direction that occurs from the

wobbling movement of the swing. In this process all of the displacements are captured from the Vicon system, together with the sensor outputs. They are processed and saved. The displacement outputs use metric units for measurement (in millimetres), therefore post-processing was performed to make them appropriate for the training of the neural network. First the outputs were offset to have mean of zero, before dividing them with their corresponding maximum absolute amplitude, so that they are bounded between [-1,1]. However, for the requirement of the FPGAs' ADCs and the safe margin from the threshold range of the device, the magnitude can be bounded to [-0.7, 0.7] easily by multiplying by a factor 0.7. No pre-processing was required of the sensor outputs, as the adjustment of the gain G was already complete (Section 6.3.2). Typical real outputs from sensors 1, 2, and 3 (blue line) and the outputs (red line) from the Vicon, displacements $\bar{x}$, $\bar{y}$ and $\bar{z}$ used for training are shown in Figure 6.14. Notice that sensors 1 and 3 are out of phase by 180 degrees, consistent with equations (6.4) and (6.5). The 'twin peaks' were present throughout the early periods of the swing, implying that the swing was beyond the simple harmonic region. The profile and characteristics of the results are consistent with those obtained by simulation, thus meeting *Condition 4*.

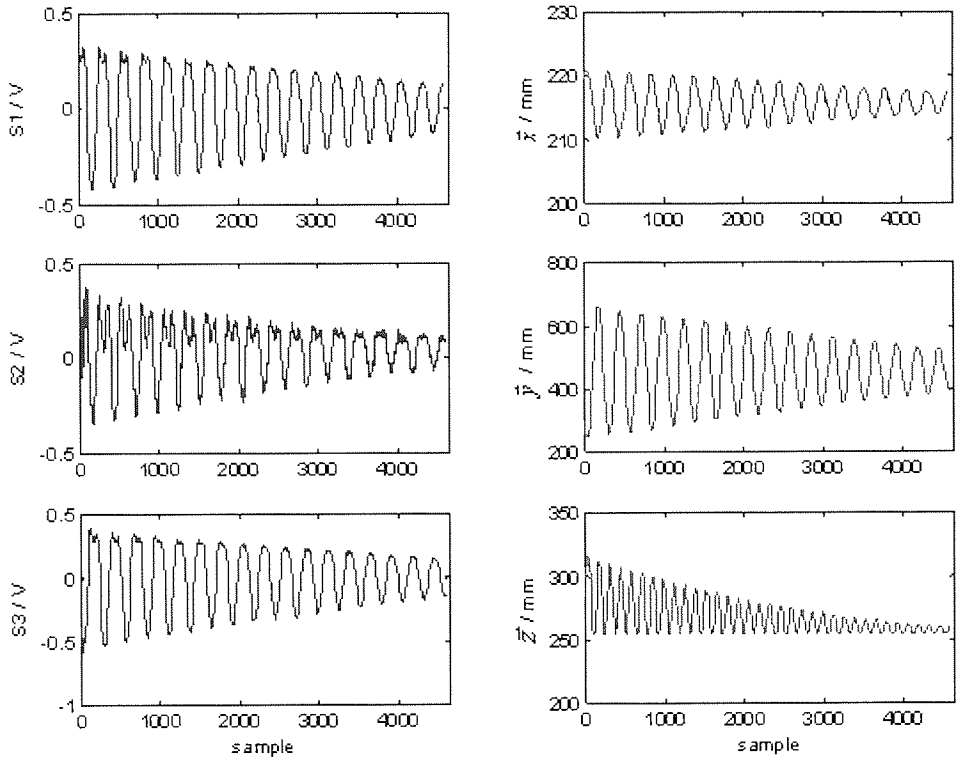


Figure 6.14: Typical sensor outputs (blue) and vicon output (red) of the $\vec{x}$, $\vec{y}$ and $\vec{z}$ displacement,

6.5 Interpretation Tool

To calculate the displacement of the outputs $\vec{x}$, $\vec{y}$ and $\vec{z}$ of the swing, the neural network algorithm (as described in the pervious chapter) was used to interpret the sensor readings. Here, the neural network, which is based on training using back-propagation, utilised an average total of 4600 samples of data to achieve satisfactory convergence. To measure the generalisation error, the data was split into three sections, one of which was used for training (50% of the data), another for validating (25%) and final section was used solely for the testing (25%). Similar to earlier training, early stopping regularisation was applied to prevent the network from over-training. Optimisation of the network configuration, such as the optimum number of hidden layers, also used a similar technique as described

earlier, for example the tactic of training the network with varying numbers of hidden nodes in order to detect the best performance.

Three different types of neural network arrangements, as introduced in Chapter 3, are adopted and used here. The Single network is a network which relies only on the output from three sensors for the evaluation of the three outputs $\vec{x}$, $\vec{y}$ and $\vec{z}$ concurrently. The Cascade network relies on the three sensor outputs, or other networks' outputs, as part of the input information as it evaluates each displacement individually. In this configuration, the three sensors were fed into a single neural network to evaluate $\vec{y}$ displacement. The evaluated $\vec{y}$, together with the sensor outputs, were then used by the next neural network to evaluate displacement $\vec{x}$. The displacements $\vec{x}$ and $\vec{y}$, and the set of outputs from the sensors are used to evaluate displacement $\vec{z}$.

Another type of arrangement added to the investigation and used for the implementation is referred to as a Multiple network. The configuration is similar to that of the Cascade, but it relies only on the sensor outputs and uses of set of three concurrent (but separate) networks to evaluate $\vec{x}$, $\vec{y}$ and $\vec{z}$ individually. The training of the Multiple is similar to the Single, but was done individually for $\vec{x}$, $\vec{y}$ and $\vec{z}$. For the training of the Cascade network the training by parts as described in Chapter 3 was adopted. However, the training by parts of the network using the suggestive approach (as discussed in Chapter 5) was not employed here. This is because optimisation was limited by the such higher number of samples for the simulation of the bit true model system. The cost of the simulations is very high. One solution is to reduce the sample time for the simulation, thus changing the sample time of the model. However, such an approach may lead to inconsistency in the actual model used in the implementation. Therefore the training follows the training procedure performed for the Cascade A design (refer to Chapter 5). Optimisation of the hidden nodes was undertaken for the three networks. It was found that the Single network was more effective with 16 hidden nodes, thus producing a configuration of 3 inputs, 16 hidden nodes, and 3 outputs. For the Multiple network, three single networks were trained and optimised. The optimum configurations were 3 : 7 : 1,

3 : 7 : 1 and 3 : 9: 1 for the outputs $\bar{x}, \bar{y}$ and $\bar{z}$ respectively. For the Cascade network, the optimum configuration was 3 : 7 : 1 for the evaluation of output $\bar{y}$, 4 : 6 : 1 for output $\bar{x}$ and 5 : 5 : 1 for output $\bar{z}$. Note that $\bar{y}$ was selected as the primary parameter because it is the most influenced by the swing. Also, based on the simulated results obtained from the Single and the Multiple networks, $\bar{y}$ is the output showing the best accuracy of evaluation, followed by the $\bar{x}$ and $\bar{z}$. As output $\bar{z}$ had the most un-linear characteristic it was placed as the last output in the cascade.

For the implementation of the networks, the Hybrid design and the Cascade design described in Chapter 5 could be used directly for the Single and Cascaded networks. For the Multiple network the Hybrid design could be adopted and duplicated into three sets of networks. All of the designs were updated for the correct number of hidden nodes. The training weights and biases were fed into the constants and ROM Blocksets. The designs were generated, synthesis and implemented on the same FPGA with a clock frequency of 64MHz. The resource requirement for the different designs is tabulated in Table 6.1.

	Single	Multiple	Cascade
Occupied Slice	33%	48%	39%
4 inputs LUTs	21%	30%	24%
Bounded IOBs	7%	14%	14%
MULT18X18s	52%	76%	60%
Peak memory	218 MB	259 MB	233 MB
Total Gate count	285 K	414 K	336 K

Table 6.1: Digital resource requirements for Single, Multiple and Cascade design

6.6 The Results

The embedded models were tested independently in real time. On each of the tests the Vicon system was calibrated to give the optimum solution during extraction of data. The output displacements $\bar{x}, \bar{y}$ and $\bar{z}$, with the sensor data for each of the Single, Multiple and Cascade designs were captured and processed. Figures 6.15 - 6.17 show the

responses obtained from the experiment (blue line), the Vicon displacement (dark-green line) and the relative errors (red line). The results suggest that the predicted response tracks the Vicon outputs satisfactorily. To measure the performance accurately requires computing the percentage error (equation (6.6)). The error was computed by dividing the mean of the absolute relative error by the mean of the total displacement, and multiplying by 100. The percentage errors with the relevant means parameters for the Single, Multiple and Cascade designs are tabulated in Table 6.2. In terms of verification of the results, percentage errors obtained from series of simulations utilising the captured sensor output of the series independent tests were also computed using FPGA simulations and Matlab simulations. The comparative results are shown in the form of bar charts in Figures 6.18 to 6.20. The FPGA simulations correspond to the feed forward simulation using the bit true number system models. The Matlab simulations used explicit number system models. The simulations are similar to those in Chapters 3 and 5, except here the actual sensor outputs captured by the Vicon system during the testing are used.

$$\%\ error = \frac{mean(abs(relative\ error)}{mean(actual\ total\ displacement)} x100 \qquad (6.6)$$

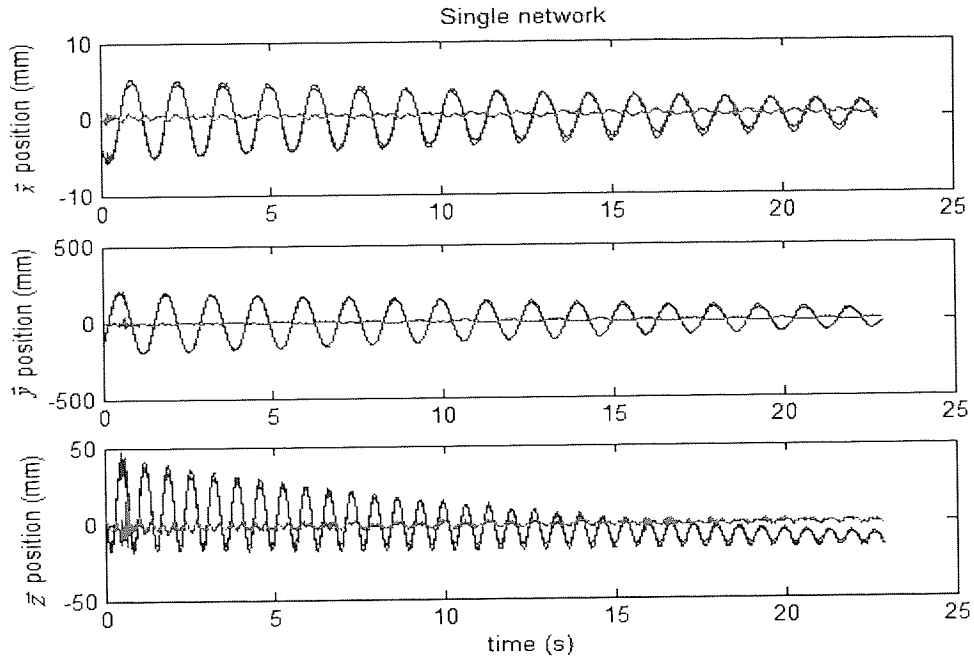


Figure 6.15: Approximated (blue line) and vicon displacement (dark-green line) and relative error (red line) of $\bar{x}$, $\bar{y}$ and $\bar{z}$ using the *Single* network.

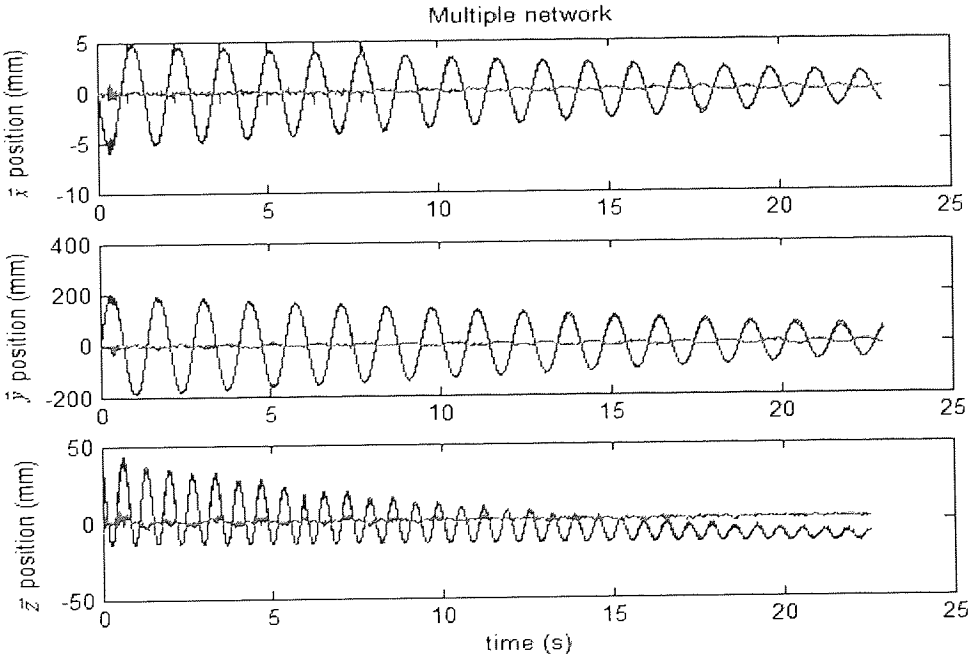


Figure 6.16: Approximated (blue line) and vicon displacement (dark-green line) and relative error (red line) of $\bar{x}$, $\bar{y}$ and $\bar{z}$ using the *Multiple* network.

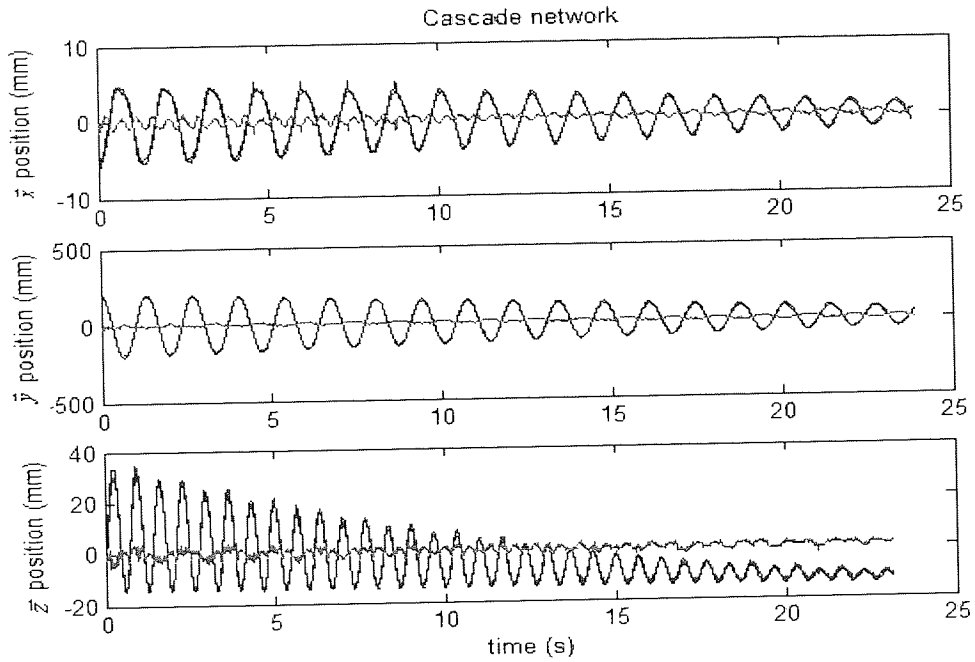


Figure 6.17: Approximated (blue line) and vicon displacement (dark-green line) and relative error (red line) of $\vec{x}, \vec{y}$ and $\vec{z}$ using the *Cascade* network.

Single	Mean absolute error/mm	Mean total displacement/mm	Percentage error
$\vec{x}$	0.3	8.9	3.3 %
$\vec{y}$	8.6	330.2	2.6 %
$\vec{z}$	1.9	26.8	7.0 %

Multiple	Mean absolute error/mm	Mean total displacement/mm	Percentage error
$\vec{x}$	0.1	8.9	1.5 %
$\vec{y}$	5.1	324.4	1.6 %
$\vec{z}$	0.9	24.4	3.5 %

Cascade	Mean absolute error/mm	Mean total displacement/mm	Percentage error
$\vec{x}$	0.3	9.0	3.2 %
$\vec{y}$	6.3	340.9	1.8 %
$\vec{z}$	0.9	21.5	4.4 %

Table 6.2: Mean absolute relative error, mean actual total displacement and the percentage error.

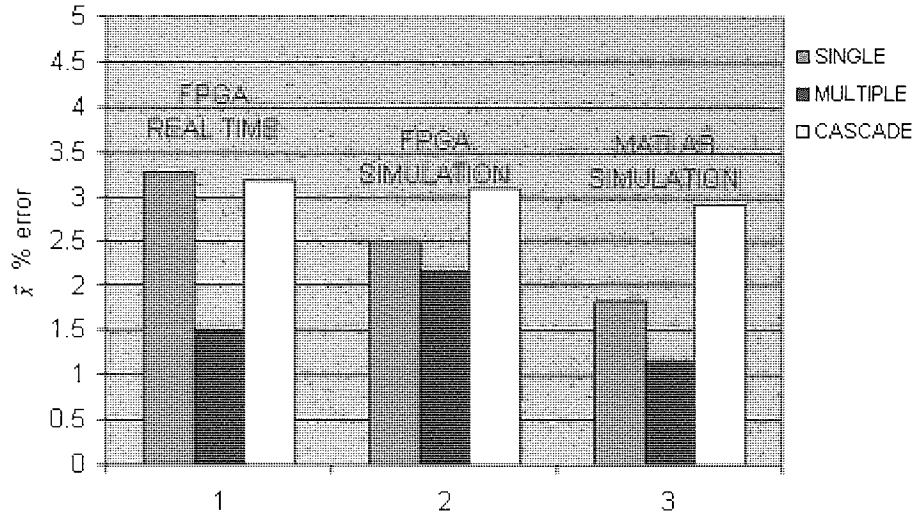


Figure 6.18: Percentage error for $\vec{x}$ from testing and simulations.

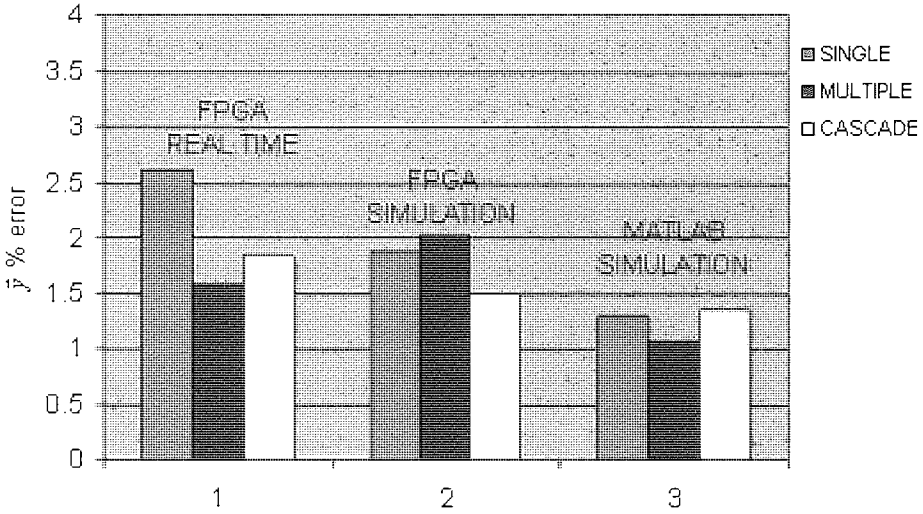


Figure 6. 19: Percentage error for $\vec{y}$ from testing and simulations.

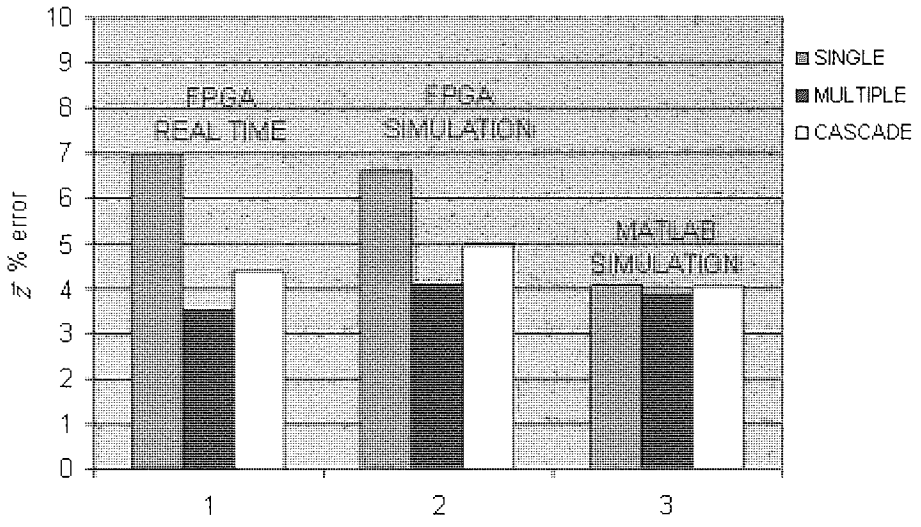


Figure 6.20: Percentage error for $\vec{z}$ from testing and simulations.

The results reveal that the performances of the three methods used for interpreting the displacements in real time are satisfactory. The overall accuracies of the methods are not lower than 97% to approximate $\vec{x}$, 97.4% for $\vec{y}$ and 93% for $\vec{z}$. For the $\vec{z}$ value, even with the presence of uncertainty from the Vicon system (as explained in Section 6.4.3.1), the interpretation is still acceptable. This is because the suggested method, taking the resultant displacement and extracting from it the individual $\vec{x}$, $\vec{y}$ and $\vec{z}$ components, was followed. By doing so, the error in each of the components can be minimised. Secondly, even if some error exists in the outputs, the high sample frequency of the salient embedded sensory interpretation system is designed to be able to reject these. This is the advantage of the technique. The generalisation and early stopping has proven to be effective here.

It is also important to note during the tests, the initial angle of the swing $\varphi_{initial}$ was relatively high (section 6.3.2) that this has lead the swing to experience simple harmonic motion but show signal of more complex behaviour mainly in the early periods. With the accuracies mentioned imply that the systems have shown their competency to predict even with non-linear case.

As all of these systems have been shown to be adequate for the application, the issue then becomes which is the most appropriate. First the cost of training can be compared. Obviously the Cascade network is the most intricate network to train, followed by the Multiple network and than the Single is the most simple. In terms of the digital resources requirement, the Multiple is the most resources hungry with gate count requirements of 414K, followed by the Cascade and the Single with 336K and 285K respectively. In terms of accuracy, the Single is the least favoured network for the dynamic interpretation in this application, followed by the Cascade and then the Multiple.

It was discovered that the Cascade had shown relatively higher error in this application in comparison to the Multiple design. This is mainly due to the cascading technique of the design. As the network is dependent on the output of the previous network, any uncertainty presents with the sensor outputs signals will inherently causing errors to all of the outputs computed in cascade. This justification is consistent since in real case scenario, measurement involving dynamic motion will be more susceptible to noise. By deduction, it is thus suggested that the Multiple is the most appropriate method for the dynamic interpretation application, but with the expanse of high digital resources required.

6.7 Summary

This chapter has demonstrated that the three architectures incorporating the designed embedded sensory sensing system are all adequate for the application of motion interpretation in real time, using a rigid and fixed steel surface as the distributive sensing medium. The three types of architectures are the Single, Multiple and Cascade networks, all of which have shown satisfactory results during their implementation, with overall accuracies greater than 93%. With such accuracies, this method has the potential to be extended for the more sophisticated discrimination of human motion, and potentially types of gait, from the disturbance produced on the contact surface.

CHAPTER 7

Conclusion and Future Research

7.1 Conclusion

Chapter 1 has broadly described some of the prospective applications of the hardwired distributive tactile sensing system in medical devices. This chapter addresses the main contributions of the current research, summarised as follows; Firstly the formulation of a distributive sensing method for application to a single dimensional system. This method benefits the development of a flexible digit with a minimal number of sensing elements, which may well lead to the production of a low cost tactile endoscope. The system having as few as four sensors was proven to produce satisfactory results. Secondly is the formulation of digital activation friendly algorithms used for the implementation of distributive tactile sensory processors, such as neural networks, into hardware. This research provides empirical data reflecting various comparisons of the proposed digital activation functions in terms of mathematical background, results and performances. Thirdly is the demonstration of different designs of neural network

architectures for hardware implementation. These architectures are also useful for those researchers interested in implementing other tactile sensing systems, such as the array method, which involves a neural network as pattern interpreter (see Figure 7.1). Fourthly is the introduction of a fast and efficient means of prototyping designs into digital hardware such as FPGAs. Following the proposed flow design allows easy reconfiguration of the target design implemented onto the device, even after deployment. And lastly is the study of a two-dimensional system for use with hardwired distributive tactile sensing to interpret real time motion demonstrating versatility of the hardwired approach as the interpreter function of distributive tactile sensor of different application and configuration.

From this list of contributions it is clear that the projects undertaken have met the original aims of this research.

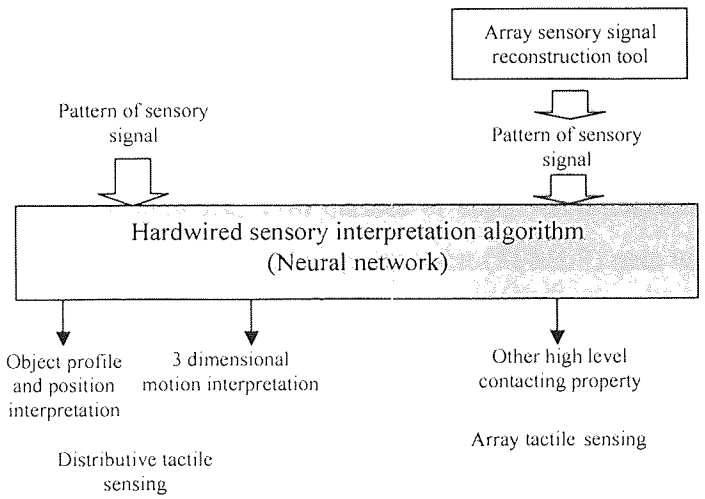


Figure 7.1: Diagram to illustrate the flow of the research and the flexibility of the technique in terms of cooperating with other techniques, such as an array with neural network based signal processing.

The studies described in this thesis have explored and discovered effective means of implementing neural networks into hardware for the purpose of developing a novel hardwired distributive tactile sensing system. Two applications of the system have been identified. The first is the use of this system in a flexible digit with tactile feedback, which has the possibility of leading to a low cost but smart tactile endoscope. The second

application is for the interpretation of sway, surge and heave dynamic information, useful in clinical applications such as balance and posture assessment of patients. The main benefit of the technique is high performance, compactness and flexibility in terms of reconfiguring abilities. The technique devised has also shown many features which make it applicable as a general purpose technique, useful for applications other than those in the biomedical field.

The essence of the sensing technology explored in this thesis is distributive tactile sensing, a type of sensing which is unique in terms of the way the sensors are constructed and the topology of the sensory processing. This research has demonstrated that only a minimal number of sensing elements is required for high accuracy. This technique thus offers the possibility of a less complication system, particularly in terms of fabricating the sensory elements, and is also more robust and has a low cost of construction.

The sensory processing involves in distributive tactile sensing is usually in the form of an interpretation algorithm, used to directly interpret the pattern of signals from the sensory elements into contact information. A clearly effective method used in this research is the use of a supervised neural network. Two classes of supervised neural networks, namely radial basis function (RBF) and multi layer perceptron (FFMLP) nets were presented, but this research has chosen to use the FFMLP type net. This decision was made based on information from previous research (refer to Chapter 2), and the empirical investigation presented in Section 3.11.3. The investigation conducted as part of this project revealed that the RBF tends to require more mathematical computation, particularly in terms of multipliers and adders, than the FFMLP, with respect to the same number of hidden nodes. With the network configuration of 2 inputs, 6 hidden nodes and 2 outputs for the prediction of load magnitude along the position, the RBF required 56 multipliers and 29 adders, whereas the FFMLP requires only 24 multipliers and 26 adders. A design which consumes a high number of mathematical operations, particularly multiplication, is a serious issue when it comes to digital implementation. Most of the devices could not cope

with a design requiring an excessive number of digital resources. In this research, the FPGA used is a Virtex-II 2XCV3000 from Xilinx and offers only 96 digital multipliers.

Also note that in Chapter 3, the accuracy of the FFMLP was investigated, including various network arrangements to infer loading condition, such as the magnitude and the profile at various applied positions. By arranging the network in cascade (Section 3.12.1) the mean errors improved to 0.58%, 0.58% and 0.85%, equivalent to than that obtained by the multiple, but better than that obtained by the single neural network arrangement for the evaluation of load. Other mean error rates for this network were 0.63%, 1.02% and 2.19% for the evaluation of position, 0.63%, 3.54% and 4.29% for the evaluation of width, and 2.61%, 2.95% and 9.02% for the evaluation of load shape. For this application, it can be deduced that the extra input(s) obtained from the output(s) of the latter network have improved the convergence of the training. This is mainly due to that these extra inputs(s) carried useful information for the proceeding network to better adapt with the problem. A similarity can be observed in a human during learning process. The more we are provided with useful information, the more easily our brain can understand.

All of the results obtained from the implementation of the distributive tactile sensing system in Chapter 3 were using computer simulations. Usually simulations are only useful for the evaluation stage, when validating the proposed technology to be used in a real system. In this case, the investigation of the system could not fully reflect the real case scenario, which is if it was tested in a real application. Inevitably, the presence of noise will significantly affect the ability of the interpretation tool to predict the result correctly and reliably. Thus this factor will reduce the applicability of the analysis generated by the simulation. As mentioned, this level of achievement is reported in previous research such as (e.g. Ma, Brett, Wright & Griffiths, 2004; Tam, 2005). However, the current research has gone further down this track, and has thus made this research innovative.

One of the primary challenges of the hardware implementation of the neural network was to duplicate as close as possible the real function of the feed-forward neural network topology into a digital representation, before its implementation into the FPGA. The idea of implementing only the forward function of the network was based on the fact that the neural networks used are of a supervised type. A supervised neural network requires supervision in order to achieve its optimum network configuration, including the parameters based on repeated training. However, although some of the work described in the literature focused on implementing training, it can still be argued that this cannot fully fulfil the aims of optimised neural network training. More importantly, even if the training of a neural network using hardware had been achieved, it would not have supported the high complexity of the *training by parts* of the cascaded neural network proposed in the current research. The complete training of the cascade is composed of a series of sub-training sessions of individual output. Important decisions related to the purpose of optimisation have to be made in between the sub-training stages.

In this research, the training and optimisation of the neural network was still run as software intelligent property using a processor such as a PC workstation. Having the training offline provides the ability to make decisions about early stopping and regularisation (as has been explained in Chapter 3) in order to prevent the network from being over-trained. The problem of over-training is a generic issue in neural network applications. Another advantage of this approach is that it allows the training to be undertaken by a different training optimiser for the purpose of searching for an effective training method. For example, if the problem of local minima is inevitable if using a conventional backpropagation training method with a gradient descent optimiser, one can employ another optimiser such that is based on scaled conjugates, fuzzy logic (Bezdek, 1992) or genetic algorithms (Fogel, 1994). Or if the training data is limited, another training algorithm such as the Bayesian technique (Bishop, 2003) can be used (see Figure 7.2). In other words this approach can facilitate the implementation of the *Hybrid neural network*, which has recently started gaining popularity (Giles, Sun and Zurada, 1998). However, in the current research training using backpropagation with a scale conjugate

optimiser was shown to be adequate, and thus was used throughout the application. Upon training, the specific purpose testing model can be calibrated easily. Training using the explicit number system will offer better accuracy than that of using the two's complement based number system training.

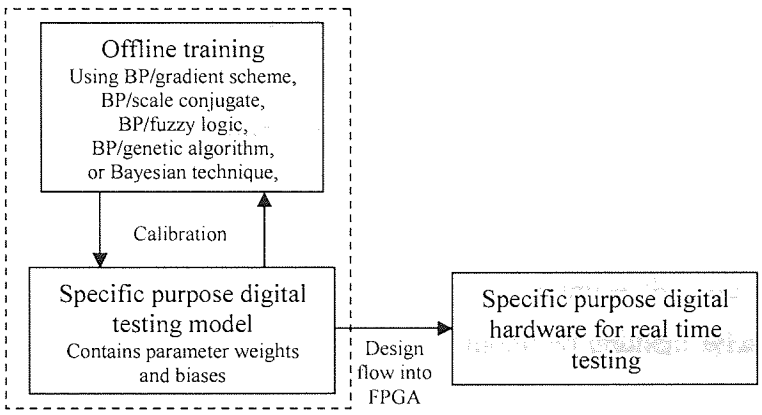


Figure 7.2: Design flow of training/testing.

The duplication of the feed-forward algorithm also includes the activation function, such as the hyperbolic tangent, which lies at the heart of the network problem. In this research four approximation algorithms were presented to replace the actual hyperbolic tangent function. These were the Gradient and Polynomial schemes, which represent piecewise mathematical approximations, the Padé approximation, which represents continuous mathematical approximation, and the Look-Up Table based technique. The models were validated using bit true simulations. The findings revealed that the Padé generated an error of 0.09% and a total gate of 33K, the Gradient produced an error of 0.4% and a total gate of 44K, and the Polynomial had an error of 0.58% and a total gate of 28K. All of these were shown to be more adequate than the conventional method, the Look-Up Table (LUT), with an error of 1.08% and a total gate of 74K. The introduction of the approximation functions was found to be an effective solution to the implementation problem.

Three architecture-friendly designs were devised for the implementation of the FFMLP neural networks, namely Fully parallel, Hybrid (folded) and Cascade (hybrid based) designs. The fully parallel architecture takes the parallel computational of the neural

network and implements it into an FPGA. This method was found to offer high performance, but was also the most resource hungry approach. This architecture was used to incorporate the various types of activation function designs, and then tested to evaluate the basic loading parameters; load magnitude and load position. From the results it was clear that all of the networks produced adequate accuracies, although the network using the LUT exhibited a slightly higher error. However, the high resource demands of the Fully parallel approach constrained the implementation of extra input, hidden and output nodes. To solve the resource problem the Hybrid (folded) model was proposed. The Hybrid (folded) is a design which mimics the concurrent behaviour of the actual network topology, but employs the re-usable technique to optimise the use of mathematical operations effectively. To facilitate the design, a modified gradient scheme is introduced as an activation function. The previous gradient scheme was selected for modification for its better *digital-compliance* algorithm in comparison to the Padé and the Polynomial. The Hybrid was adopted for use in the design of the cascaded system.

The investigation on evaluating the best interpretation tool has gone further by comparing two different optimisation types used to obtain the optimised feed forward models of the cascade network. First there is the explicit number system of optimisation (as explained in Chapter 3), in which each network is trained and optimised using the sensor inputs and the computed outputs produced by simulating the explicit feed forward model. A second option is the bit true number system of optimisation. Here the cascade network is optimised by the procedures exercised earlier for the first option, but taking into account the use of the bit true feed forward model, (digital design) to simulate the outputs. In the context of the thesis, the first is referred to as Cascade network A, and the later is Cascade network B. The Hybrid, Cascade A and Cascade B were verified using bit true simulations. From the simulations the mean error for the Hybrid are better than 8.57% for load, 4.07% for load position, 5.54% for load width and 14.20% for load shape respectively. For Cascaded network A, the mean error rates are better than 3.67% for load, 3.63% for load position, 5.70% for load width and 13.91% for load shape. Finally, for Cascade network B, the mean errors rates are better than 3.67% for load, 2.60% for load

position, 2.49% for load width and 2.9% for load shape. On the basis of these findings it was concluded that optimisation using bit true simulation is a more effective and influential strategy for the digital cascade network. However by considering the cost to obtain high accuracies from network complexities (between Single and Cascade A), more favours are put on the Single and Cascade B for implementation.

The single neural network from the Hybrid design and the cascaded neural networks from Cascade network B were successfully implemented into the Xilinx Virtex-II 2XCV3000. The results were then analysed. In the context of application to single dimensional system demonstrating for the almost static case of contact discrimination, the implementation findings described in Chapter 5 confirmed the superiority of employing the Cascade over the Hybrid (single network) in terms of accuracy. The drawbacks of the cascade model are the high cost of training and slower output operation in comparison to the single model.

In relation to the flexible digit application with tactile sensing feedback, the accuracy of the method provided by the implementations means that it is adequate for clinical evaluations. The accuracies can be verified by the work done by Ma, Brett, Wright and Griffiths (2004). In that study the accuracies achieved for the evaluation of load magnitude, load position and load shape are 7%, 6%, 6%, and 1%. In the current research the error in relation to the loading parameters read from the real-time measurement (as achieved by cascade B) are better than 6.4%, 2.3%, 2.9% and 3.8%. With these accuracies, and with the consideration that the implementations are all digitisation-based, implied the superiority of the technique devised from this research. Thus the system considered to be sufficient to facilitate operator control in MIS.

The hardwired tactile sensing approach devised in this research is not limited to a specific application. With the system allowing flexibility in terms of the ability to calibrate the design, relative simplicity in terms of programming and re-programming of the FPGA (even after deployment) and high performance, this approach is clearly feasible for other

applications, such as the interpretation of motion from contact disturbance. This approach was thus adopted for a dynamic tactile application and used for the interpretation of sway, surge and heave motion of a pendulum. This investigation revealed that the system demonstrating Single, Multiple and Cascade network configuration, had overall accuracy rates of better than 93%. The hardwired sensing approach is thus competent to handle this application, strongly implying that it can be used in upcoming research involving human subjects.

From this research it can be concluded that the novel hardwired distributive tactile sensing system is successful and will be beneficial for the two medical oriented applications described in this thesis. The hardwired signal processing system developed in this research has shown its proficiency at generating real-time interpretations with satisfactory accuracies. It can be deduced that the output from the research has met the hypothesis.

7.2 Future work

- To enhance the investigation of 1 D case by introducing a suitable means of actuation control. This will demonstrate the hardwired distributive sensing with active actuation. The study will more closely analogue to real cases such as the steerable endoscope. In the current research (as detailed in Chapter 3) an experiment was performed using a cantilever beam to mimic the sensing elements of the flexible digit system. However actuation forces such as driving force, which could largely influence the deformation of the medium, was not include in the study. Although it has been emphasised in the thesis that the empirical results from the investigation have direct relevance to the later case (with active actuation), it would be more appropriate if the investigation considered the use of the actuated flexible digit.
- To integrate functions of the actuation controller and tactile feedback to demonstrate haptic feedback in the flexible digit. Information from measurement (discrimination) and

the actuation command can be used together to assist the closed loop control of the system. If the digit presses too hard on a surface during maneuvering, excessive force from contact will be fed back and used to prevent damage.

- To adopt the Fibre Bragg Grating sensor proposed by Cowie, Webb, Tam, Slack, and Brett (2006). As described in this thesis the FPGA only takes and processes signals in the voltage domain. Since the signal from the FBG is in the frequency domain, this could be converted to voltage through the use of a frequency to voltage converter. But for the case of the FBG, this process can not be implemented directly as the signal data output from the FBG has to be processed using an algorithm to extract the relevant signal information. Therefore the idea of incorporating FBG with the hardwired distributive tactile sensing system is still receiving considerable attention.
- To apply the dynamic plate to human as the subjects such that as to discriminate walking condition.
- To continue the work on exploiting the training algorithm using the FPGA. During the period this research, the author has started designing the training tool for the FFMLP targeted to be implemented on the same FPGA. The training tool would equip simple neural network training specifically feasible for the Single neural network arrangement used in the research. The biggest challenge of the work would be devising the digital algorithm to facilitate the problem of over-fitting.

7.3 Future Five Years of the Subject

This research has opened a new direction of investigation in the use of hardwired tactile sensing, particularly in the area of medical applications. Hardwired tactile sensing will gain popularity because of its user-friendly system capable of providing high performance discriminative based measurements in real time, with high efficiency of design and low cost fabrication.

In few years time, this tool will bring important solutions that may revolutionise many industrial products involving tactile sensing.

End of Chapters

References

Agnon, Y., Madsen, P. A. and Scheffer, H. A. (1999), "A new approach to high order Boussinesq model", *Journal of Fluid Mechanics*, vol. 399, pp. 319-333, 1999.

Arroyo Leon, M. A. A. Ruiz Castro, A. Leal Ascencio, R. R. (1999), "An artificial neural network on a field programmable gate array as a virtual sensor", *Proceeding of Third International Workshop on Design of Mixed-Mode Integrated Circuits and Applications,* Puerto Vallarta, Mexico, pp.114-117, 1999.

Avci, M. and Yildirim, T. (2003), "Generation of tangent hyperbolic sigmoid function for microcontroller based digital implementation of neural networks", *Int. XII. Turkish Symposium on Artificial Intelligence and Neural Networks-TAINN Çanakkale,* Turkey, vol. E4, pp. 339-341, 2003.

Baker, G. A., Jr.; and Graves-Morris, P. (1981), "Padé Approximants Part 1: Basic Theory", *Addison-Wesley Publishing Company, Reading,* Massachusetts, 1981

Barranco, B. L., Andreou, A. G., Indiveri, G. and Shibata, T., (2003), "Guest editorial special issue on neural networks hardware implementations", *IEEE Transactions on Neural Networks*, vol. 14, no.5, pp.976-979, September 2003.

Beiu, V., Peperstraete, J. A., Vandewalle, J. and Lauwereins, R. (1994), "VLSI complexity reduction by piece-wise approximation of the sigmoid function", *Proceedings of European Symposium on Artificial Neural Networks. ESANN '94*, Brussels, Belgium, pp. 181-6, 1994.

Benhadj, R. and Dawson, B. (1995), "Air jets imaging tactile sensing device for automation applications", *Robotica*, vol.13, no.5, pp.521-529, 1995.

Bezdek, J. (1992), "Fuzzy logic and neural networks", *IEE Transactions on Neural Networks*, vol. 3, no. 5, pp. 641-642., 1992.

Bishop, C. M. (2003), "Neural Networks for pattern recognition", *Oxford University Press Inc*, New York, USA, 2003.

Bolton, W. (1999), "Mechatronics Electronic control systems in mechanical and electrical engineering", *Prentice Hall*, London, UK, 1999.

Brett, P. N. and Stone, R. S. W. (1997), "A technique for measuring contact force distribution in minimally invasive surgical procedures", *Proceedings of IMechE, Part H*, vol. 211, no. H4, 1997

Brett, P. N. and Li, Z. (2000), "A tactile sensing surface for artificial neural network based automatic recognition of the contact force position", *Proceeding of Institution of Mechanical Engineering,* vol. 214, no.1, pp.207-215, 2000.

Brown, S. and Rose, J. (1996), "Architecture of FPGAs and CPLDs: A Tutorial", *IEEE Design and Test of Computers*, vol. 13, no. 2, pp. 42-57, 1996.

Chaudhry, H., Findley, T., Quigley, K, Bukiet B, Ji, Z., Sims, T., Maney, M. (2004), "Measures of postural stability", *Journal of Rehabilitation Research and Development*, vol.41, no.5, pp. 713-720, 2004.

Chaudhry, H., Findley, T., Quigley, K., Ji, Z., Maney, M., Sims, T., Bukiet B. and Foulds, R. (2005), "Postural stability index is more valid measure of stability than equilibrium score", *Journal of Rehabilitation Research and Development*, vol.42, no. 4, pp. 547-556, 2005

Cowie, B. M., Webb, D.J., Tam, B. K. Y., Slack, P. and Brett, P. N. (2006), "Distributive Tactile Sensing using Fibre Bragg Grating Sensors for Biomedical Applications", *The First IEEE/RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics,* Italy, pp. 312- 317, February 2006.

Cox, C. E. and Blanz, W. E. (1992), "GANGLION- A fast field programmable gate array implementation of connectionist classifier", *Journal of Solid State Circuits,* vol.27, no.3, pp. 288-299, March 1992

Dargahi, J., Parameswaran, M. and Payandeh, S. (2000), "A micromachined piezoelectric tactile sensor for an endoscopic grasper : Theory, fabrication and experiments", *Journal of Microelectromechanical Systems*, vol.9, no.3, pp.329-335, 2000.

DiFabio, R. P. and Faudriat, B. A. (1996), "Responsive and realibility of a pediatric strategy score for balance", *Physiotheraphy Research International*, vol.1, no.3, pp.180-194, 1996.

DiFabio, R. P., Emasithi, A. and Paul, S. (1998), "Validity of visual stabilization conditions used with computerized dynamic platform posturography", *Acta Otolaryngol.*, vol.118, no. 4, pp. 449-454, 1998.

Engel, J., Chen, J. and Liu, C. (2003), "Development of polyimide flexible tactile sensor skin", *Journal of Micromechanics and Microengineering*, vol.13, pp. 359-366, 2003.

Fearing, R.S. (1990), "Tactile sensing mechanism", *International Journal of Robotic Research*, vol. 9, no.3, pp. 3-23, 1990.

Figliola, R. S. and Beasley, D. E. (2006), "Theory and Design for Mechanical Measurements, 4th Edition", *John Wiley & Sons*, UK, Inc, 2006.

Fogel, D. B. (1994), "An Introduction to Simulated Evolutionary Optimization", *IEEE Transaction on Neural Networks*, vol. 5, no. 1, pp. 3-14, 1994.

Fraden, J. (1996), "Handbook of modern sensors, 2nd edition", *Springer-Verlag, New York*, 1996.

Fuller, G. F. (2000), "Falls in the elderly", *American Family Physician*, vol. 61, no. 7, pp. 2159-2168, 2000.

Gadea, R., Cerda, J., Ballester, F. and Mocholi, A. (2000), "Artificial neural network implementation on a single FPGA of a pipelined on-line backpropagation", *International Symposium on System Synthesis*, pp. 225-230, 2000.

Galindo Hernendez, M., Leal Ascencio, R. R., Aguilera Galicia, C. (1999), "The study of prototype of an artificial neural network on a FPGA as a function approximator", *Third International Workshop on Design of Mixed-Mode Integrated Circuits and Applications*, Puerto Vallarta, Mexico, pp.126-129, 1999.

Gere, J. M. (2002), "Mechanics of Materials 5th SI Edition", *Nelson Thornes Ltd*, UK, 2000.

Giles, C. L., Sun, R. and Zurada, J. M. (1998), "Guest editorial neural network and hybrid intelligent models: foundations, theory and application", *IEEE Transactions on Neural Networks*, vol.9, no. 5, pp. 721-723, September 1998

Graham, L. (2006), "Strain Gages and Force measurements" http://cnx.org/content/m13779/latest/ (Last edited on August 2006)

Guarnieri, S. and Piazza, F. (1999), "Nultilayer feedforward network with adaptive spline activation function", *IEEE Transactions on Neural Networks*, vol. 10, no. 3, May 1999.

Harmon, L. D. (1982), "Automated tactile sensing", *International Journal of Robotics Research*, vol. 2, pp. 3-31, 1982.

Harrison, A. J. and Hillard, P. J. (2000), "A moment-based technique for the automatic spatial alignment of plantar pressure data", *Proceedings of the Institution of Mechanical Engineers Part H. Journal of Engineering in Medicine*, vol. 214, no. 3, pp.257-264, 2000.

Hausdorff, J. M., Rios, D.A. and Edelberg, H. K., (2001) "Gait variability and fall risk in community-living older adults: a 1-year prospective study", *Archive Physical Medicine and Rehabilitation*, vol. 82, pp.1050-1056, 2001.

Heidemann, G. and Schopfer, M. (2004), "Dynamic tactile sensing for object identification", *Proceedings of IEEE International Conference on Robotics and Automation*, ICRA '04, vol. 1, pp. 813-818, 2004

Hellard, G. and Russell, R. A. (2002), "A robust, sensitive and economical tactile sensor for a robotic manipulator", *Proceeding 2002 Australasian Conference on Robotics and Automation, Auckland,* pp. 100-104, November 2002.

Hikawa, H. (2000), "An efficient three-valued multilayer neural network with on-chip learning suitable for hardware implementation", *System and Computers in Japan,* vol. 31, no. 4, pp. 43-51, 2000.

Hikawa, H. (2003), "A digital hardware pulse-mode neuron with piecewise linear activation function", *IEEE Transactions on Neural Networks,* vol.14, no.5, pp.1028-1037, September 2003.

Hollis, R. (2004), "Berkshire encyclopedia of Human-computer interaction", *Berkshire Publishing group*, pp. 311-316, 2004.

Hopkins, H. H. (1976), "Optical principles of endoscopy. In: Berci G Endoscopy", *New York, Appleton Century Crofts,* pp. 3-63, 1976.

Howe, R. D. (1994), "Tactile sensing and control of robotic manipulation", *Advance Robotica*, vol. 8, no. 3, pp. 245-261, 1994.

Izeboudjen, N., Farah, A., Titri, S. and Boumeridja, H. (1999), "Digital implementation of artificial neural networks: from VHDL description to FPGA implementation", *Proceeding of International Conference on Artificial Neural Networks IWANN'99,* Spain, vol. 2, pp. 139-148, June 1999.

Johansson, R. S. and Vallbo, A. B. (1983), "Tactile sensory coding in the glabrous skin of the human hand", *Trends Neuroscience,* vol. 6, pp. 27-31, 1983.

Johansson, R, Magnusson, M., Fransson, P.A. and Karlberg, M. (2001), "Multi-stimulus multi-response posturography", *Mathematical Biosciences*, vol. 174, no.1, pp. 41-59, 2001.

Kim, S. H., Engel, J., Liu, C. and Jones, D.L. (2005), "Texture classification using a polymer-based MEMS tactile sensor", *Journal of Micromechanics and Microengineering*, vol. 15, pp. 912-920, 2005.

Kolesar, E. S., Reston, R. R., Ford, D.G. and Fitch, R.C. (1992), "Multiplexed piezoelectric polymer tactile sensor", *Journal of Robotic System*, pp. 37-63, 1992.

Lawrence E. and Valentine, A. (2006), "Archive issue library: Planning considerations for the minimally invasive surgical suite", http://www.healthcaredesignmagazine.com/Past_Issues.htm?ID=5024

Li, D. and Shida, K. (1997), "Fuzzy algorithm to discriminate material properties using touch sensors", *Electrical Engineering in Japan*, vol. 118, no.1, pp. 61-70, 1997.

Lindberg, F. (2002), "Carbon dioxide pneumoperitoneum-hemodynamic consequences and thromboembolic complications" *Dissertation for the degree of Doctor of Philosophy (Faculty of Medecine) in Surgery, Uppsala University*, 2002.

Losonc, Z. (2003), "The basic Properties of a Pendulum", http://www.gyogyitokezek.hu/fe/pendtutor1.htm (Last updated on 29 August 2003).

Lynn, P. A., Fuerst, W. and Thomas, B. (1997), "Introductory Digital Signal Processing with Computer Applications", *John Wiley and Sons Ltd*, 1997.

Ma, X. and Brett, P. N. (2002), "The performance of a 1-D distributive tactile sensing system for detecting the position, weight and width of a contacting load", *IEE transaction on Instrumentation and measurement*, vol. 51, no. 2, April 2002.

Ma, X., Brett, P. N., Wright, M. T., and Griffiths, M. V. (2004), "A flexible digit with tactile feedback for invasive clinical application", *Proc Inst Mech Eng [H] Engineering Medicine*, vol. 218, pp. 3, pp. 151-157, 2004.

Maeda, A., Nakamura, K., Otomo, A., Higuchi, S. and Motohashi,Y. (1998), "Body Suppot Effect on Standing Balance in the Visualy Impaired Elderly", *Archived of Physical Medicine Rehabilitation*, vol. 79, pp. 994-997, August 1998.

Marchesi, M., Orlandi, G., Piazza, F. and Uncini, A. (1992), "Fast neural networks without multipliers", *IEEE Transactions on Neural Networks*, vol.3, pp.101-120, November 1992.

Matsuo, T., Narita, A., Senda, M., Hasebe, S. and Ohtsuki, H. (2006), "Body Sway Increase Immediately after Strabismus Surgery", *Acta Medica Okayama*, vol. 60, no. 1, pp. 13-24.

McCulloch, W. S. and Pitts, W. (1943), "A logical calculus of the idea imminent in nervous activity", *Bulletin of Mathematical Biophysics*, vol. 5, pp. 115-133, 1943

Menciassi, A., Park, J. H., Lee, S., Gorini, S., Dario, P. and Park, J.O. (2002), "Robotic Solutions and Mechanisms for a Semi-Autonomous Endoscope", *IEEE/RSJ International Conference on Intelligent Robots and Systems,* pp. 1379-1384, 2002

Miller, R. A. (1986), "Endoscopic instrumentations: Evolution, physical, principle and clinical aspects", *British Medical Bulletin*, vol. 42, no.3, pp. 223-225, 1986.

Nabney, I. T. (2004), "Netlab Algorithms for pattern recognition", *Springer*, London, UK, 2004.

Nadapalan, V., Smith, C., A., Jones, A., S., and Lesser, T., H., J. (1995), "Objective measurement of the benefit of walking sticks in peripheral vestibular balance disorders, using the sway weigh and balance platform", *Journal of Larngology and Otology*, vol. 109, no. 9, pp. 836-840, 1995.

Nagy, Z., Szolgay, Z. and Szolgay, P. (2004), "Tactile sensor modelling using emulated digital CNN-UM", *Proceedings of the 8th IEEE international workshop(CNNA) on Cellular Neural Networks and their Applications*, Budapest, pp. 399-404, 2004.

Najarian, S., Dargahi, J. and Zheng, X. Z. (2006), "A novel method in measuring the stiffness of sensed objects with applications for biomedical robotic systems", *The International Journal of Medical Robotics and Computer Assisted Surgery*, vol. 2, no. 1, pp. 84-90, 2006

Nekoogar, F. and Nekoogar, F. (2003), "From ASICs to SOCs a practical approach", *Prentice Hall,* New Jersey, USA, 2003.

Nicholls, H. R. and Lee, M. H. (1989), "A survey of robot tactile sensing technology" *International Journal of Robotic Research*, vol.8, no.3, pp.3-30, 1989.

Nilsson, M. (2000), "Tactile sensors and other distributed sensors with minimal wiring complexity," *IEEE/ASME Transactions on Mechatronics*, vol. 5, no. 3, September 2000.

Ossoinig, H., Reisinger, E., Steger, C. and Weiss, R. (1996), "Design and FPGA implementation of a neural network", *Proceedings on 7th International Conference cn Signal Processing Application & Technology, Boston USA*, pp. 939-943, October 1996.

Pandya, V., Areibi, S. and Moussa, M. (2005), "A handel-C implementation of back propagation algorithm on field programmable gate array", *International Conference on Reconfigurable Computing and FPGAs,* Mexico, pp. 8, September 2005.

Provancer, W. R. (2003) "On tactile sensing and display", *PhD Thesis, Stanford University*, August 2003.

Reyneri, L. M. (1995), "A Performance analysis of pulse stream neural and fuzzy computing systems," *IEEE Transactions Circuits System*, vol. 42, pp. 642-660, October 1995.

Smithwick, Q. Y. J., Seibel, E. J. and Vagners, J. (2001), "Control Aspects of the Single Fiber Scanning Endoscope", *Proceedings of SPIE 4253*, pp. 176, 2001.

Stone, R. S. W. and Brett, P.N. (1995), "A flexible pneumatic actuator for gripping soft irregular shaped objects", *IEE Colloquium on 'Innovative Actuators for Mechatronic System'*, London, UK, pp. 13-15, October 1995

Stone, R. S. W. and Brett, P. N. (1996), "A sensing technique for the measurement of tactile forces in the gripping of dough like material", *Proceedings of the Institution of Mechanical Engineering Part B Journal of Engineering Manufacture*, vol. 210, no. 3, pp. 261-269, 1996.

Stone, R. W. S. (1997) "A distributive tactile sensing technique for soft deformable contact", *PhD Thesis, University of Bristol,* 1997.

Szilard, R. (1974), "Theory and analysis of plates, Classical and numerical methods", *Prentice Hall, INC, Englewood Cliffs, New Jersey*, pp. 415, 1974.

Takashima, K., Yoshinaka, K., Okazaki, T., and Ikeuchi, K. (2005), "An endoscopic tactile sensor for low invasive surgery", *Sensor and Actuators A 119,* pp. 372-383, 2005.

Tam, B. K. Y., Brett, P. N., Holding, D. J. and Griffiths, M. (2004), The experimental performance of a flexible digit retrieving tactile information relating to clinical applications, *Proceedings on 11th IEEE International Conference Mechatronics and Machine Vision in Practice,* Macao, pp. 234-238, November 2004.

Tam, B. K. Y. (2005), "A novel actuated digit with tactile feedback for clinical applications", *PhD Thesis, Aston University*, Birmingham, August 2005.

Tamura, S., Hirano, M., Chen, X., Sato, Y., Narumi, Y., Hori, M., Takahashi, S. and Nakamura, H. (2002), "Intrabody Three-Dimensional Position Sensor for an Ultrasound Endoscope", *IEEE Trans Biomedical Engineering,* vol. 49, no. 10, pp.1187-94, October 2002.

Tang, C. Z. and Kwan, H.K. (1992), "Convergence and generalization properties of multilayer feedfoward neural networks", *Proceeding of IEEE International Symposium Circuits System, San Diego, CA*, pp.65-68, 1992.

Tavakoli, M., Patel, R.V. and Moallem, M. (2004), "Design issue in a haptics-based master-slave system for minimal invasive surgery", *Proceedings of the 2004 IEEE, International Conference on Robotics and Actuation,* New Orlean, pp. 371-376, April 2004.

Timoshenko, S. and Woinowsky-Krieger, S. (1959), "Theory of Plates and Shells" *McGraw-Hill Book Company, USA*, 1959.

Tongpadungrod, P. (2002), "Characteristics of distributive tactile sensing systems", *PhD Thesis, University of Bristol,* December 2002.

Tseng, C. I., Liou, W. J. and Lee. H. J. (1999), "Development of an artificial skin ridge for pattern recognition", *Journal of Intelligent Material System and Structures,* vol. 8, pp. 4-11, January 1997

Xilinx System Generator (2006), Xilinx System Generator For DSP User Guide.November 2006, http://www.xilinx.com/support/sw_manuals/sysgen_ug.pdf

Yamauchi, Y., Yamashita, J., Fukuia, Y., Yokoyama, K., T.Sekiya, T., Ito, E., Kanai, M., Fukuyo, T., Hashimoto, D., Iseki, H. and Takakura, K. (2002), "A Dual-View Endoscope With Image Shift", *Proceedings of CARS-ISCAS*, pp. 1, 2002.

Yamashita, J., Yamauchi, Y., Mochimaru, M., Fukui, Y. and Yokoyama, K. (1999), "Real-Time 3-D Model-Based Navigation System for Endoscopic Paranasal Sinus Surgery", *IEEE Transactions Biomedical Engineering,* vol. 46, no. 1, pp. 107-116, January 1999.

Appendix 1

ANALOG DEVICES

Precision Instrumentation Amplifier

AD624

Aston University

Content has been removed due to copyright restrictions

Appendix 2

Sensor 1	Sensor 2	Sensor 3	Sensor 4	Load	Position	Width	Shape
-0.9300	-0.9300	-0.9300	0.8877	0.9000	0.2250	0.9000	0.4500
-0.9050	-0.8800	-0.8050	0.8877	0.9000	0.2700	0.9000	0.4500
-0.8800	-0.8300	-0.6800	0.8932	0.9000	0.3150	0.9000	0.4500
-0.8550	-0.7800	-0.5500	0.8844	0.9000	0.3600	0.9000	0.4500
-0.8300	-0.7300	-0.4200	0.9031	0.9000	0.4050	0.9000	0.4500
-0.8000	-0.6800	-0.3200	0.9042	0.9000	0.4500	0.9000	0.4500
-0.7700	-0.6300	-0.2200	0.8910	0.9000	0.4950	0.9000	0.4500
-0.7390	-0.5770	-0.1575	0.8910	0.9000	0.5400	0.9000	0.4500
-0.7080	-0.5240	-0.0950	0.8932	0.9000	0.5850	0.9000	0.4500
-0.6805	-0.4735	-0.0483	0.8965	0.9000	0.6300	0.9000	0.4500
-0.6530	-0.4230	-0.0016	0.8998	0.9000	0.6750	0.9000	0.4500
-0.6220	-0.3670	-0.0023	0.8932	0.9000	0.7200	0.9000	0.4500
-0.5910	-0.3110	-0.0030	0.9064	0.9000	0.7650	0.9000	0.4500
-0.5605	-0.2575	-0.0040	0.9152	0.9000	0.8100	0.9000	0.4500
-0.5300	-0.2040	-0.0050	0.9075	0.9000	0.8550	0.9000	0.4500
-0.8040	-0.8230	-0.8920	0.7328	0.7435	0.1800	0.6750	0.4500
-0.7805	-0.7810	-0.7870	0.7317	0.7435	0.2250	0.6750	0.4500
-0.7570	-0.7390	-0.6820	0.7405	0.7435	0.2700	0.6750	0.4500
-0.7340	-0.6980	-0.5770	0.7306	0.7435	0.3150	0.6750	0.4500
-0.7110	-0.6570	-0.4720	0.7394	0.7435	0.3600	0.6750	0.4500
-0.6875	-0.6145	-0.3675	0.7416	0.7435	0.4050	0.6750	0.4500
-0.6640	-0.5720	-0.2630	0.7174	0.7435	0.4500	0.6750	0.4500
-0.6395	-0.5280	-0.1855	0.7383	0.7435	0.4950	0.6750	0.4500
-0.6150	-0.4840	-0.1080	0.7427	0.7435	0.5400	0.6750	0.4500
-0.5900	-0.4400	-0.0635	0.7427	0.7435	0.5850	0.6750	0.4500
-0.5650	-0.3960	-0.0190	0.7350	0.7435	0.6300	0.6750	0.4500
-0.5400	-0.3510	-0.0115	0.7372	0.7435	0.6750	0.6750	0.4500
-0.5150	-0.3060	-0.0040	0.7438	0.7435	0.7200	0.6750	0.4500
-0.4895	-0.2600	-0.0045	0.7383	0.7435	0.7650	0.6750	0.4500
-0.4640	-0.2140	-0.0050	0.7372	0.7435	0.8100	0.6750	0.4500
-0.4385	-0.1695	-0.0045	0.7284	0.7435	0.8550	0.6750	0.4500
-0.4130	-0.1250	-0.0040	0.7394	0.7435	0.9000	0.6750	0.4500
-0.5410	-0.5740	-0.6750	0.4647	0.4695	0.1350	0.4500	0.4500
-0.5255	-0.5465	-0.6065	0.4669	0.4695	0.1800	0.4500	0.4500
-0.5100	-0.5190	-0.5380	0.4790	0.4695	0.2250	0.4500	0.4500
-0.4940	-0.4910	-0.4685	0.4735	0.4695	0.2700	0.4500	0.4500
-0.4780	-0.4630	-0.3990	0.4779	0.4695	0.3150	0.4500	0.4500
-0.4620	-0.4350	-0.3305	0.4801	0.4695	0.3600	0.4500	0.4500
-0.4460	-0.4070	-0.2620	0.4680	0.4695	0.4050	0.4500	0.4500
-0.4305	-0.3785	-0.1935	0.4757	0.4695	0.4500	0.4500	0.4500
-0.4150	-0.3500	-0.1250	0.4856	0.4695	0.4950	0.4500	0.4500
-0.3990	-0.3215	-0.0710	0.4779	0.4695	0.5400	0.4500	0.4500
-0.3830	-0.2930	-0.0170	0.4691	0.4695	0.5850	0.4500	0.4500
-0.3665	-0.2635	-0.0115	0.4735	0.4695	0.6300	0.4500	0.4500
-0.3500	-0.2340	-0.0060	0.4724	0.4695	0.6750	0.4500	0.4500
-0.3335	-0.2055	-0.0060	0.4680	0.4695	0.7200	0.4500	0.4500
-0.3170	-0.1770	-0.0060	0.4625	0.4695	0.7650	0.4500	0.4500
-0.3000	-0.1470	-0.0055	0.4592	0.4695	0.8100	0.4500	0.4500
-0.2830	-0.1170	-0.0050	0.4570	0.4695	0.8550	0.4500	0.4500
-0.2670	-0.0885	-0.0055	0.4680	0.4695	0.9000	0.4500	0.4500
-0.3650	-0.3880	-0.4570	0.3164	0.3130	0.1350	0.4500	0.4500
-0.3535	-0.3690	-0.4045	0.3087	0.3130	0.1800	0.4500	0.4500
-0.3420	-0.3500	-0.3520	0.3142	0.3130	0.2250	0.4500	0.4500
-0.3320	-0.3310	-0.3100	0.3208	0.3130	0.2700	0.4500	0.4500
-0.3220	-0.3120	-0.2680	0.3219	0.3130	0.3150	0.4500	0.4500
-0.3110	-0.2930	-0.2215	0.3241	0.3130	0.3600	0.4500	0.4500
-0.3000	-0.2740	-0.1750	0.3142	0.3130	0.4050	0.4500	0.4500
-0.2890	-0.2545	-0.1315	0.3120	0.3130	0.4500	0.4500	0.4500
-0.2780	-0.2350	-0.0880	0.3120	0.3130	0.4950	0.4500	0.4500
-0.2665	-0.2160	-0.0540	0.3065	0.3130	0.5400	0.4500	0.4500
-0.2550	-0.1970	-0.0200	0.3021	0.3130	0.5850	0.4500	0.4500
-0.2440	-0.1765	-0.0130	0.3131	0.3130	0.6300	0.4500	0.4500
-0.2330	-0.1560	-0.0060	0.3131	0.3130	0.6750	0.4500	0.4500
-0.2225	-0.1375	-0.0065	0.3164	0.3130	0.7200	0.4500	0.4500
-0.2120	-0.1190	-0.0070	0.3164	0.3130	0.7650	0.4500	0.4500
-0.2010	-0.0990	-0.0070	0.3175	0.3130	0.8100	0.4500	0.4500
-0.1900	-0.0790	-0.0070	0.3186	0.3130	0.8550	0.4500	0.4500
-0.1785	-0.0600	-0.0070	0.3175	0.3130	0.9000	0.4500	0.4500

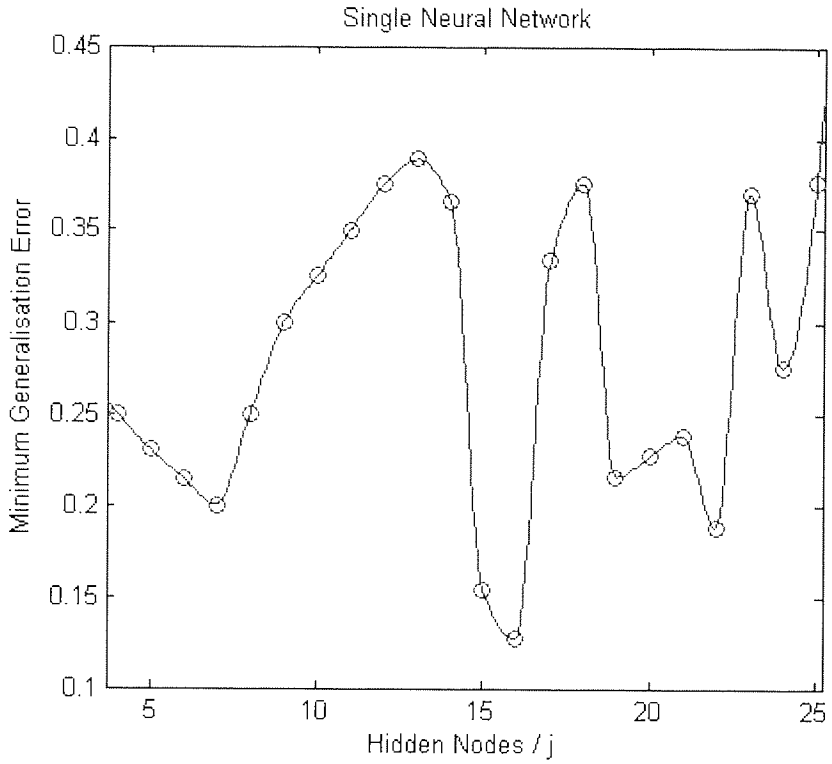
Appendix 2

-0.5140	-0.5570	-0.6900	0.4296	0.4305	0.0900	0.2250	0.4500
-0.5000	-0.5325	-0.6300	0.4318	0.4305	0.1350	0.2250	0.4500
-0.4860	-0.5080	-0.5700	0.4362	0.4305	0.1800	0.2250	0.4500
-0.4710	-0.4825	-0.5050	0.4416	0.4305	0.2250	0.2250	0.4500
-0.4560	-0.4570	-0.4400	0.4329	0.4305	0.2700	0.2250	0.4500
-0.4410	-0.4295	-0.3740	0.4395	0.4305	0.3150	0.2250	0.4500
-0.4260	-0.4020	-0.3080	0.4439	0.4305	0.3600	0.2250	0.4500
-0.4115	-0.3760	-0.2445	0.4340	0.4305	0.4050	0.2250	0.4500
-0.3970	-0.3500	-0.1810	0.4471	0.4305	0.4500	0.2250	0.4500
-0.3815	-0.3235	-0.1205	0.4384	0.4305	0.4950	0.2250	0.4500
-0.3660	-0.2970	-0.0600	0.4362	0.4305	0.5400	0.2250	0.4500
-0.3515	-0.2705	-0.0340	0.4329	0.4305	0.5850	0.2250	0.4500
-0.3370	-0.2440	-0.0080	0.4340	0.4305	0.6300	0.2250	0.4500
-0.3215	-0.2170	-0.0070	0.4285	0.4305	0.6750	0.2250	0.4500
-0.3060	-0.1900	-0.0060	0.4252	0.4305	0.7200	0.2250	0.4500
-0.2910	-0.1635	-0.0060	0.4219	0.4305	0.7650	0.2250	0.4500
-0.2760	-0.1370	-0.0060	0.4186	0.4305	0.8100	0.2250	0.4500
-0.2615	-0.1105	-0.0060	0.4285	0.4305	0.8550	0.2250	0.4500
-0.2470	-0.0840	-0.0060	0.4329	0.4305	0.9000	0.2250	0.4500
-0.4240	-0.4610	-0.5710	0.3472	0.3522	0.0900	0.2250	0.4500
-0.4120	-0.4395	-0.5190	0.3538	0.3522	0.1350	0.2250	0.4500
-0.4000	-0.4180	-0.4670	0.3593	0.3522	0.1800	0.2250	0.4500
-0.3875	-0.3960	-0.4125	0.3538	0.3522	0.2250	0.2250	0.4500
-0.3750	-0.3740	-0.3580	0.3604	0.3522	0.2700	0.2250	0.4500
-0.3630	-0.3530	-0.3065	0.3625	0.3522	0.3150	0.2250	0.4500
-0.3510	-0.3320	-0.2550	0.3691	0.3522	0.3600	0.2250	0.4500
-0.3385	-0.3105	-0.2020	0.3604	0.3522	0.4050	0.2250	0.4500
-0.3260	-0.2890	-0.1490	0.3571	0.3522	0.4500	0.2250	0.4500
-0.3140	-0.2670	-0.0985	0.3560	0.3522	0.4950	0.2250	0.4500
-0.3020	-0.2450	-0.0480	0.3549	0.3522	0.5400	0.2250	0.4500
-0.2890	-0.2225	-0.0275	0.3516	0.3522	0.5850	0.2250	0.4500
-0.2760	-0.2000	-0.0070	0.3483	0.3522	0.6300	0.2250	0.4500
-0.2635	-0.1780	-0.0065	0.3439	0.3522	0.6750	0.2250	0.4500
-0.2510	-0.1560	-0.0060	0.3527	0.3522	0.7200	0.2250	0.4500
-0.2390	-0.1345	-0.0055	0.3549	0.3522	0.7650	0.2250	0.4500
-0.2270	-0.1130	-0.0050	0.3560	0.3522	0.8100	0.2250	0.4500
-0.2145	-0.0910	-0.0060	0.3560	0.3522	0.8550	0.2250	0.4500
-0.2020	-0.0690	-0.0070	0.3549	0.3522	0.9000	0.2250	0.4500
-0.2870	-0.3120	-0.3880	0.2252	0.2348	0.0900	0.2250	0.4500
-0.2790	-0.2975	-0.3525	0.2285	0.2348	0.1350	0.2250	0.4500
-0.2710	-0.2830	-0.3170	0.2318	0.2348	0.1800	0.2250	0.4500
-0.2620	-0.2670	-0.2795	0.2362	0.2348	0.2250	0.2250	0.4500
-0.2530	-0.2510	-0.2420	0.2241	0.2348	0.2700	0.2250	0.4500
-0.2445	-0.2365	-0.2065	0.2252	0.2348	0.3150	0.2250	0.4500
-0.2360	-0.2220	-0.1710	0.2318	0.2348	0.3600	0.2250	0.4500
-0.2280	-0.2080	-0.1355	0.2417	0.2348	0.4050	0.2250	0.4500
-0.2200	-0.1940	-0.1000	0.2406	0.2348	0.4500	0.2250	0.4500
-0.2115	-0.1795	-0.0670	0.2406	0.2348	0.4950	0.2250	0.4500
-0.2030	-0.1650	-0.0340	0.2395	0.2348	0.5400	0.2250	0.4500
-0.1945	-0.1490	-0.0215	0.2439	0.2348	0.5850	0.2250	0.4500
-0.1860	-0.1330	-0.0090	0.2428	0.2348	0.6300	0.2250	0.4500
-0.1775	-0.1185	-0.0080	0.2439	0.2348	0.6750	0.2250	0.4500
-0.1690	-0.1040	-0.0070	0.2439	0.2348	0.7200	0.2250	0.4500
-0.1605	-0.0900	-0.0070	0.2417	0.2348	0.7650	0.2250	0.4500
-0.1520	-0.0760	-0.0070	0.2417	0.2348	0.8100	0.2250	0.4500
-0.1435	-0.0600	-0.0070	0.2439	0.2348	0.8550	0.2250	0.4500
-0.1350	-0.0440	-0.0070	0.2439	0.2348	0.9000	0.2250	0.4500
-0.7200	-0.7650	-0.9180	0.6251	0.6261	0.1350	0.4500	0.9000
-0.7000	-0.7280	-0.8250	0.6262	0.6261	0.1800	0.4500	0.9000
-0.6800	-0.6910	-0.7320	0.6240	0.6261	0.2250	0.4500	0.9000
-0.6600	-0.6560	-0.6420	0.6328	0.6261	0.2700	0.4500	0.9000
-0.6400	-0.6210	-0.5520	0.6119	0.6261	0.3150	0.4500	0.9000
-0.6190	-0.5845	-0.4620	0.6361	0.6261	0.3600	0.4500	0.9000
-0.5980	-0.5480	-0.3720	0.6174	0.6261	0.4050	0.4500	0.9000
-0.5765	-0.5110	-0.2825	0.5922	0.6261	0.4500	0.4500	0.9000
-0.5550	-0.4740	-0.1930	0.6054	0.6261	0.4950	0.4500	0.9000
-0.5340	-0.4365	-0.1155	0.6339	0.6261	0.5400	0.4500	0.9000
-0.5130	-0.3990	-0.0380	0.6328	0.6261	0.5850	0.4500	0.9000
-0.4910	-0.3585	-0.0235	0.6317	0.6261	0.6300	0.4500	0.9000
-0.4690	-0.3180	-0.0090	0.6273	0.6261	0.6750	0.4500	0.9000

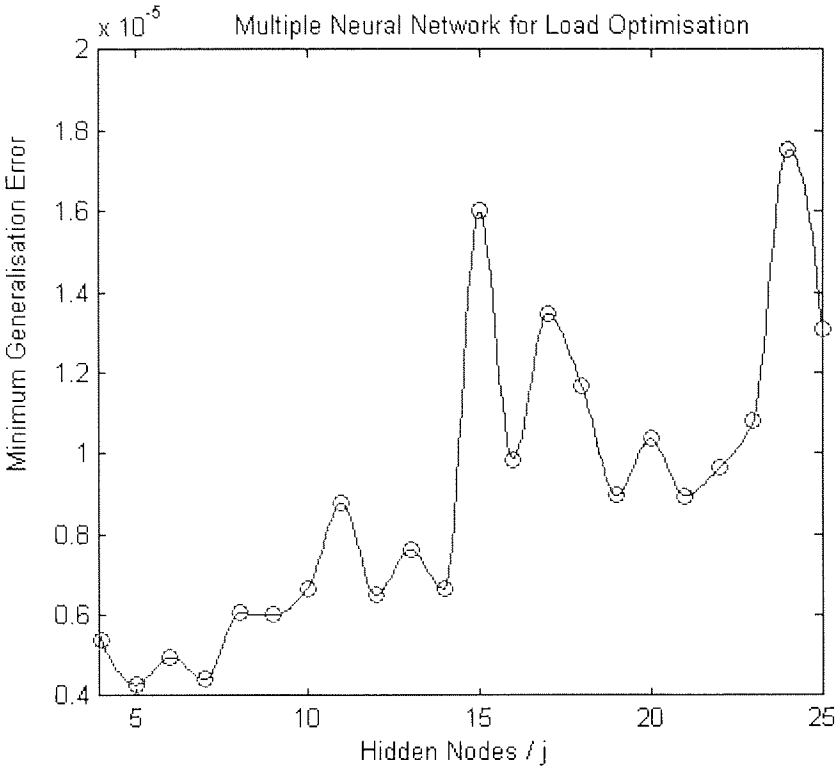
Appendix 2

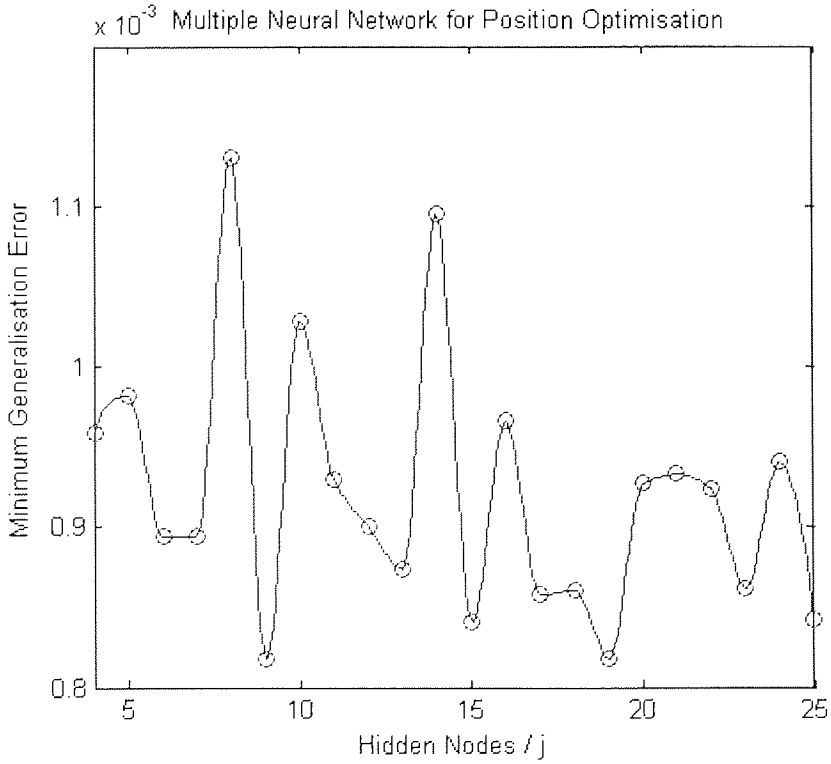
-0.4475	-0.2800	-0.0080	0.6251	0.6261	0.7200	0.4500	0.9000
-0.4260	-0.2420	-0.0070	0.6185	0.6261	0.7650	0.4500	0.9000
-0.4040	-0.2025	-0.0075	0.6141	0.6261	0.8100	0.4500	0.9000
-0.3820	-0.1630	-0.0080	0.6218	0.6261	0.8550	0.4500	0.9000
-0.3600	-0.1240	-0.0075	0.6174	0.6261	0.9000	0.4500	0.9000

Appendix 3

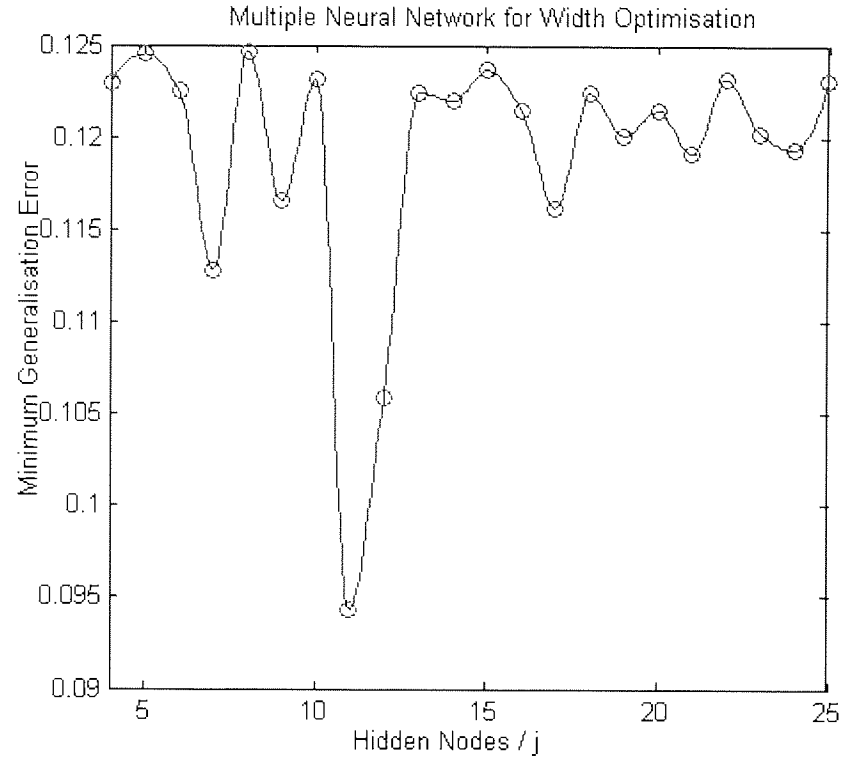

Single Neural Network
Minimum Generalisation Error
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
5
10
15
20
25
Hidden Nodes / j

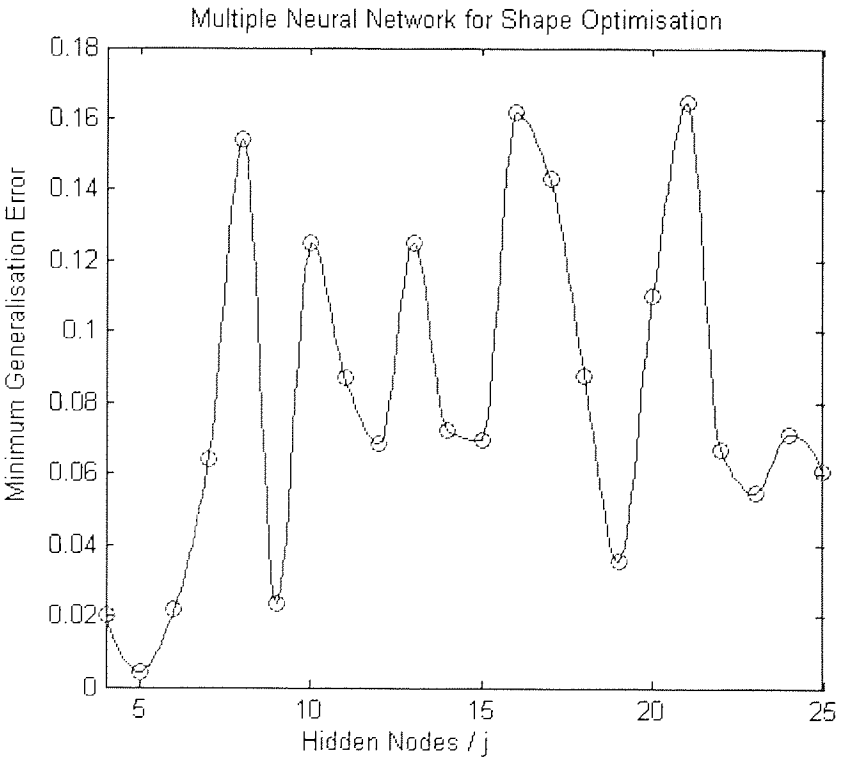
Appendix 3


x 10^-5
Multiple Neural Network for Load Optimisation
Minimum Generalisation Error
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
5
10
15
20
25
Hidden Nodes / j

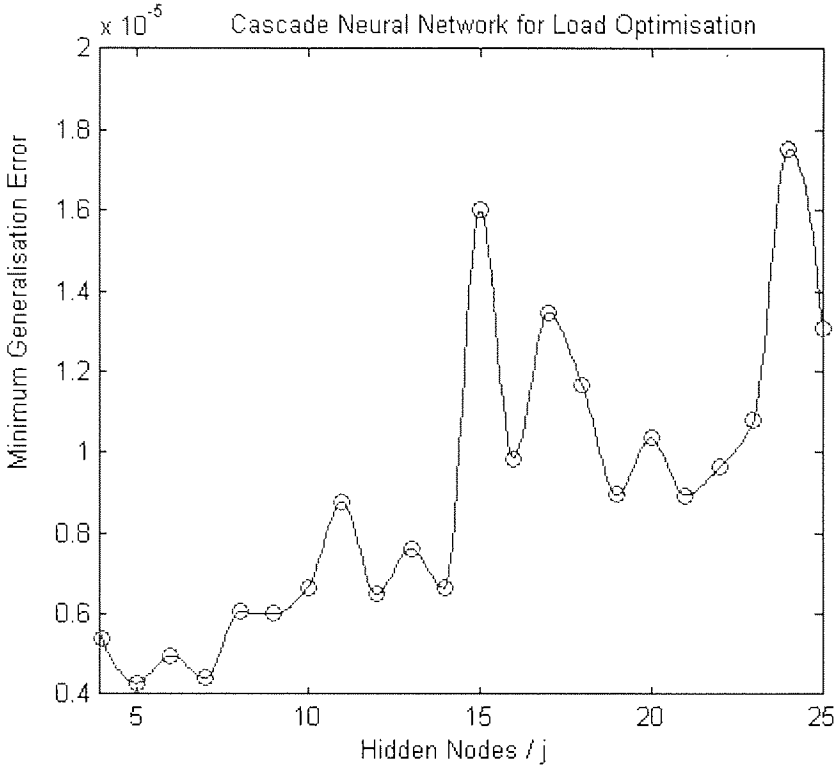

x 10^-3
Multiple Neural Network for Position Optimisation
Minimum Generalisation Error
1.1
1
0.9
0.8
5
10
15
20
25
Hidden Nodes / j

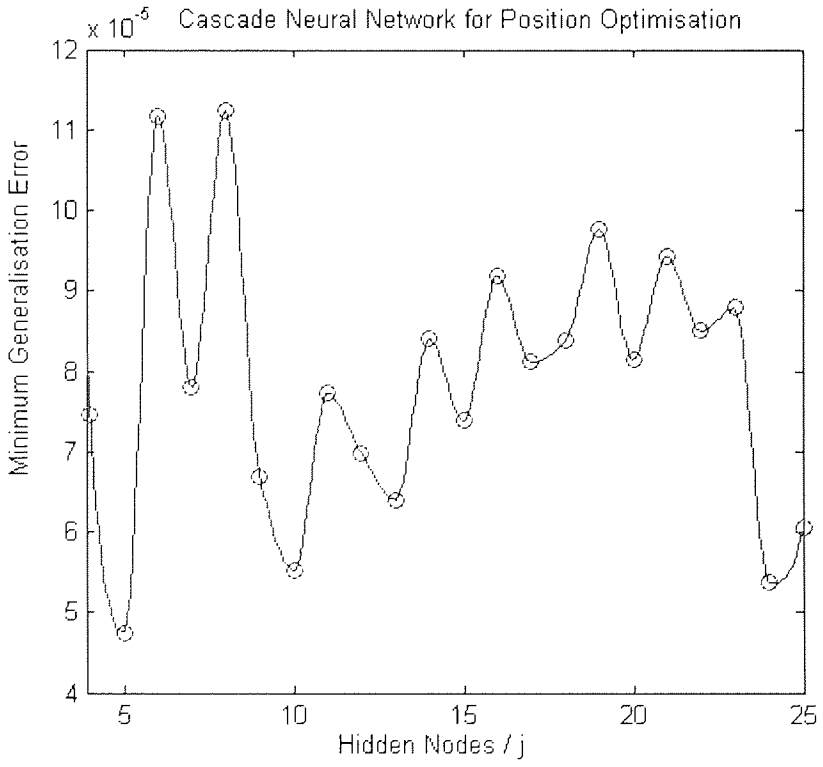
Appendix 3


Multiple Neural Network for Width Optimisation
Minimum Generalisation Error
0.125
0.12
0.115
0.11
0.105
0.1
0.095
0.09
5
10
15
20
25
Hidden Nodes / j

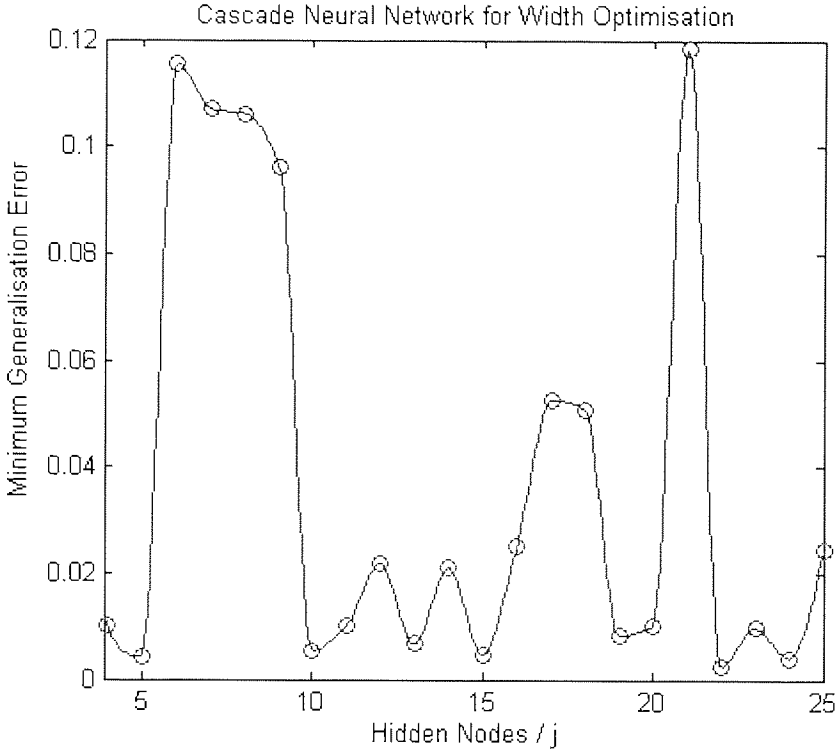

Multiple Neural Network for Shape Optimisation
Minimum Generalisation Error
0.18
0.16
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
5
10
15
20
25
Hidden Nodes / j

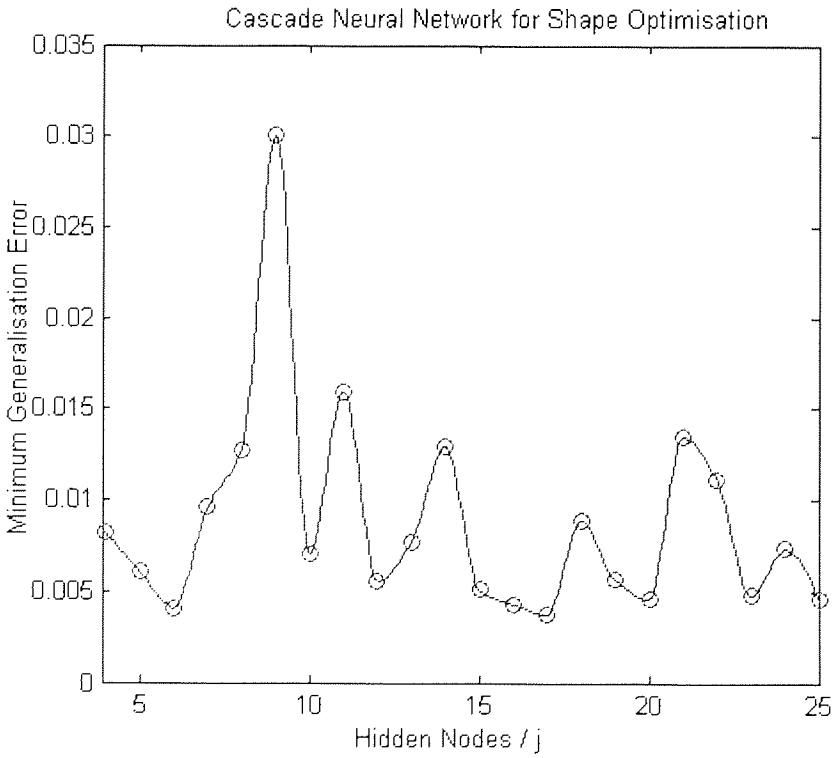
Appendix 3


x 10^-5
Cascade Neural Network for Load Optimisation
Minimum Generalisation Error
Hidden Nodes / j
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
5
10
15
20
25

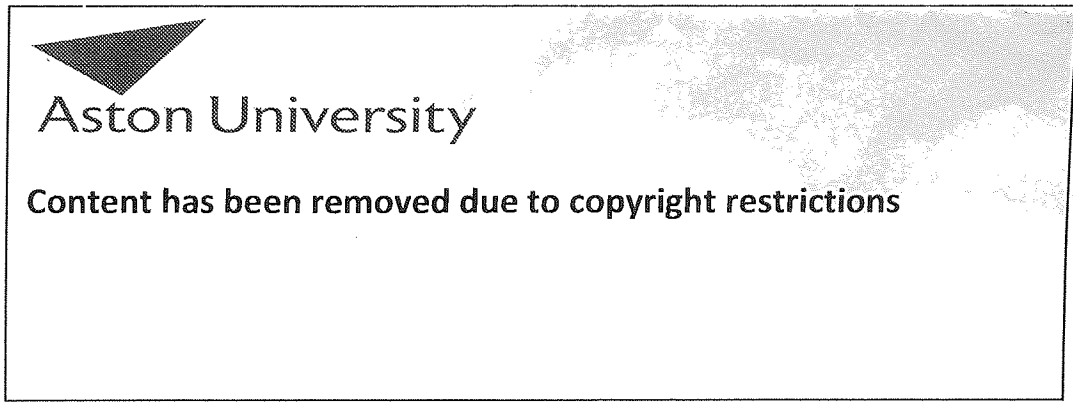

x 10^-5
Cascade Neural Network for Position Optimisation
Minimum Generalisation Error
Hidden Nodes / j
12
11
10
9
8
7
6
5
4
5
10
15
20
25

Appendix 3

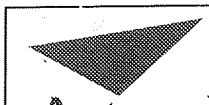

Cascade Neural Network for Width Optimisation
Minimum Generalisation Error
0.12
0.1
0.08
0.06
0.04
0.02
0
5
10
15
20
25
Hidden Nodes / j


Cascade Neural Network for Shape Optimisation
Minimum Generalisation Error
0.035
0.03
0.025
0.02
0.015
0.01
0.005
0
5
10
15
20
25
Hidden Nodes / j

Appendix 4


Aston University
Content has been removed due to copyright restrictions

Appendix 4



Content has been removed due to copyright restrictions

Appendix 5

```
clear all
load file containing vectors of the weights and bias obtained from optimization and training.
w1=demonetwork.w1;
b1=demonetwork.b1;
w2=demonetwork.w2;
b2=demonetwork.b2;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[mm m]=size(w1);
wA=[w1' b1']';
[G,H]=size(wA);
for h=1:H,
   for g=1:G,
   c(g,h)=ceil(abs(wA(g,h)));
end
end
x12=sum(c);
for i=1:m,
   a(:,i)=w1(:,i)/x12(i);
   b(i)=b1(i)/x12(i);
end

n=length(b2);
wB=[w2' b2']';
[G1,H1]=size(wB);
for h1=1:H1,
   for g1=1:G1,
   c1(g1,h1)=ceil(abs(wB(g1,h1)));
end
end
x22=sum(c1);
for j=1:n,
   x(:,j)=w2(:,j)/x22(j);
   y(j)=b2(j)/x22(j);
end

 disp('your normalized  w1: ')%% this will be the normalized of weight1, W1
 a
 disp('your normalized  b1: ') %% this will be the normalized of bias1, B1
 b
 disp('normalized factor1: ') %% this will be the normalized factor Q
 x12
 disp('your normalized  w2: ') %% this will be the normalized of weight2,W2
 x
 disp('your normalized  b2: ') %% this will be the normalized of bias2, B2
 y
 disp('normalized factor2: ') %% this will be the normalized factor Y
 x22
```

Appendix 5

19-1020; Rev 1; 5/08

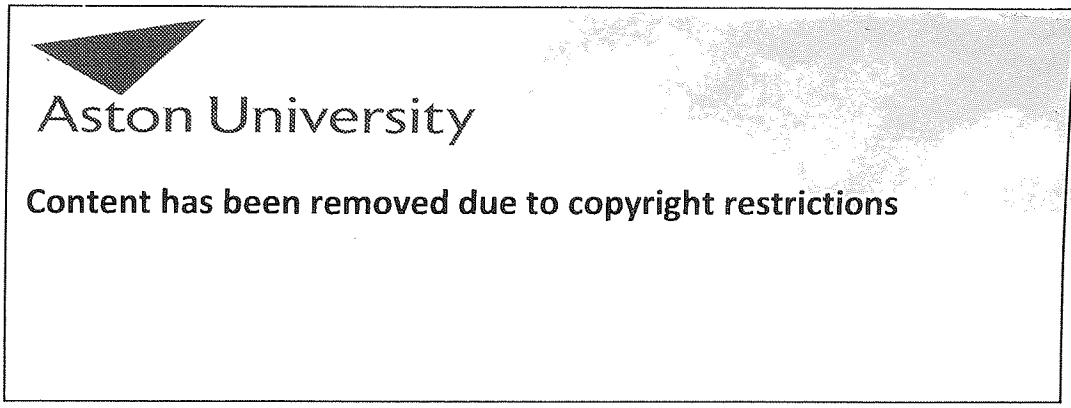
MAXIM

Precision, 4-Channel/Dual 2-Channel, Low-Voltage, CMOS Analog Multiplexers

Aston University

Content has been removed due to copyright restrictions

Appendix 5


Aston University
Content has been removed due to copyright restrictions

Appendix 7

The general equation representing the movement of the swing can be presented by the differential expression in equation (1), where φ is the angle of the swing, t is time, a_g is the gravitational acceleration, and I_p is the length of the pendulum. However, to reflect more closely the real scenario, a damping effect was added to equation (6.4) to yield equation (2), where κ is the damping factor. There are various ways to solve equations (1) and (2), but the most convenient means is to use computer simulation. The variable of interest in the preceding step is $\varphi(t)$.

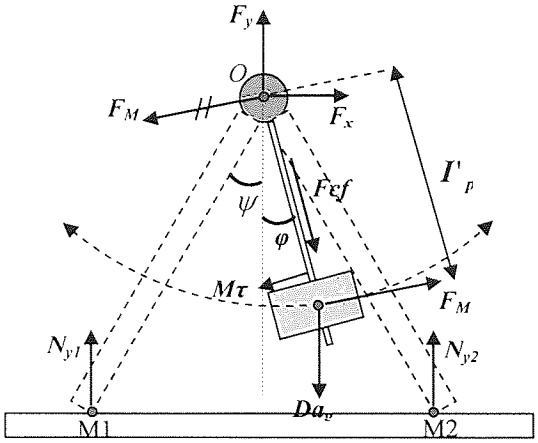
$$\frac{d^2\varphi}{dt^2} + \frac{a_g}{I_p}\sin\varphi(t) = 0 \qquad (1)$$

$$\frac{d^2\varphi}{dt^2} + \frac{\kappa}{I_p}\frac{d\varphi}{dt} + \frac{a_g}{I_p}\sin\varphi(t) = 0 \qquad (2)$$

The primary work of the derivations is to evaluate the force by the bearing 'O' of the axial rod (see Figure 1) supporting the swinging. The force on the point of axis O, can be resolved into horizontal F_x and vertical F_y components (equations (7) and (8)), by taking the effect of tangential force F_m and the tensile force F_{cp} of the rod supporting the pendulum load (equation (6) and (5)). The tensile force F_{cp} is an expression built from $\varphi_{initial}$ (the initial angle of the swing), $\varphi(t)$ (the angular time variant of from the movement of the swing as shown in equation (2), which was solved by computational numerical method), the load's moment of inertia J around axis O, equation (4), which was formed from J_s, the load's moment of inertia around its centre of gravity, equation (3)), and D is the total mass of the load.

Relative to the research work, these forces were exclusively used for resolving the total moment of the system with reference to point M1 or M2 (see figure below). This has leads to obtaining the force exerted by the feet (or the normal force N_{y1} and N_{y2}). For example, considering the swing has legs of distance I_p and the angle between them is 2ψ, the total moment of the system with reference to point M1 will be the summation of the moment produced by the $\vec{x}$ and $\vec{y}$ components of the bearing force and the moment produce by the N_{y2}. Similarly, the total moment of the system with reference to point M2 will be the summation of the moment produced by the $\vec{x}$ and $\vec{y}$ components of the bearing force and the moment produce by the N_{y1}. As the system is balance, the normal forces N_{y1} and N_{y2}, can be evaluated as expressed by (equations (12) and (13)).

Appendix 7



$$J_s = D\frac{(A^2 + B^2)}{12} \tag{3}$$

$$J = (DI'_p{}^2) + J_s \tag{4}$$

$$F_{cp} = \frac{Da_g}{J}((J + 2DI'_p{}^2)\cos\varphi(t) - (2DI'_p{}^2)\cos\varphi_{initial}(t)) \tag{5}$$

$$F_m = \frac{J_s Da_g \sin\varphi(t)}{J} \tag{6}$$

$$F_x = F_{cp}\sin\varphi(t) - F_m\sin\varphi(t) \tag{7}$$

$$F_y = F_{cp}\cos\varphi(t) + F_m\sin\varphi(t) \tag{8}$$

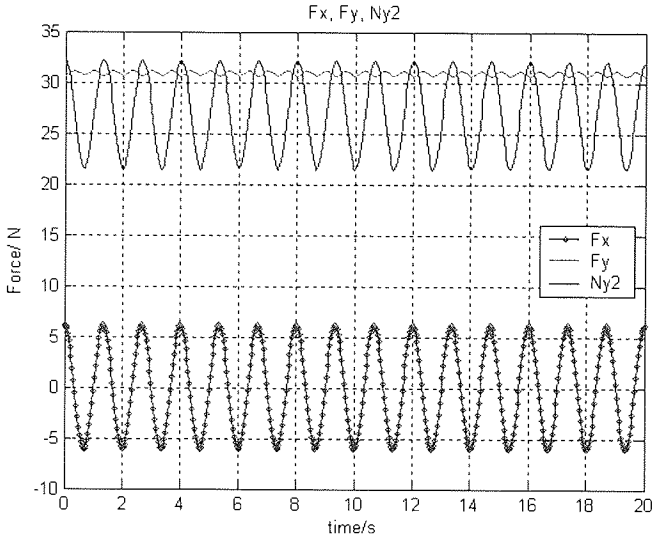
$$\sum M_1 = N_{y2}2I_p\sin\psi - F_y I_p\sin\psi - F_x I_p\cos\psi = 0 \tag{9}$$

$$\sum M_2 = -N_{y1}2I_p\sin\psi + F_y I_p\sin\psi - F_x I_p\cos\psi = 0 \tag{10}$$

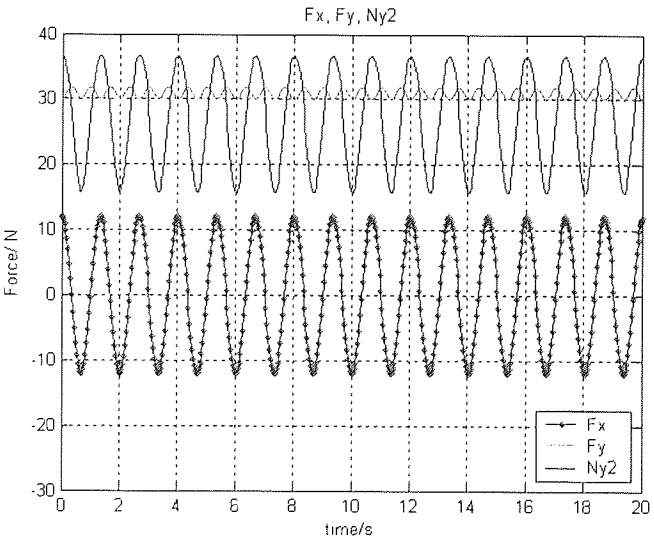
$$N_{y1} = \frac{F_y I_p\sin\psi - F_x I_p\cos\psi}{2I_p\sin\psi} \tag{11}$$

$$N_{y2} = \frac{F_y I_p\sin\psi + F_x I_p\cos\psi}{2I_p\sin\psi} \tag{12}$$

Appendix 7

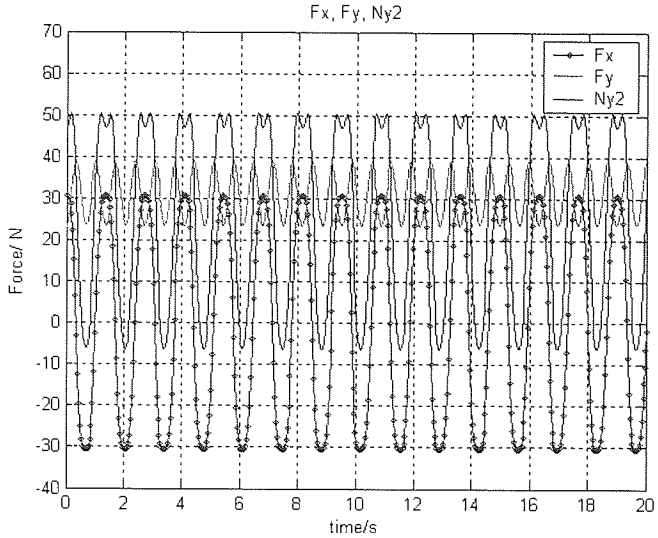


Initial swing angle $\varphi_{initial}$ = 5, damping $\kappa = 0$

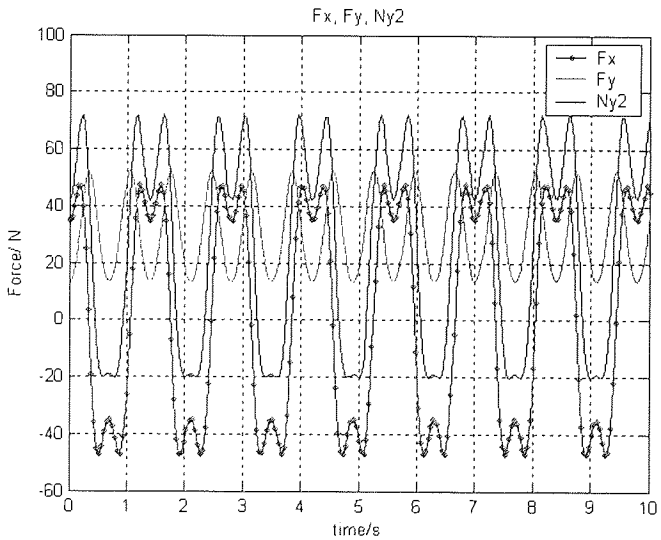


Initial swing angle $\varphi_{initial}$ = 10, damping $\kappa = 0$

Appendix 7



Initial swing angle $\varphi_{initial}$ = 30, damping κ = 0



Initial swing angle $\varphi_{initial}$ = 50, damping κ = 0.05

Appendix 8

Burr-Brown Products
from Texas Instruments

INA128
INA129

SBOS051B – OCTOBER 1995 – REVISED FEBRUARY 2005

Precision, Low Power
INSTRUMENTATION AMPLIFIERS

Aston University

Content has been removed due to copyright restrictions

Appendix 8

Aston University

Content has been removed due to copyright restrictions