Hardware Realisation of Quantum-inspired Fuzzy based Neural Network and its application as Signature Verification

Submitted in partial fulfilment of the
requirements for the award of the degree

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

by

Rajveer Singh

Under the guidance of

Dr. Aruna Tiwari

&

Dr. Sanjay Singh

INDIAN INSTITUTE OF TECHNOLOGY INDORE
CSIR-CENTRAL ELECTRONICS ENGINEERING RESEARCH INSTITUTE
November 2017

CANDIDATE'S DECLARATION

We hereby declare that the project entitled "Hardware Realisation of Quantum-inspired Fuzzy based Neural Network and its application as Signature Verification" submitted in partial fulfilment for the award of the degree of Bachelor of Technology in Computer Science and Engineering, carried out under the supervision of Dr. Aruna Tiwari, Associate Professor, Discipline of Computer Science and Engineering, IIT Indore and Dr. Sanjay Singh, Sr. Scientist CSIR-CEERI pilani is an authentic work.

Further, we declare that we have not submitted this work for the award of any other degree elsewhere.

Rajveer Singh
140001024
Discipline of Computer Science and Engineering
Indian Institute of Technology Indore

CERTIFICATE BY BTP GUIDE

It is certified that the declaration made by the students is correct to the best of my knowledge and belief.

Dr. Aruna Tiwari,
Associate Professor,
Discipline of Computer Science and Engineering,
IIT Indore

Contents

List of Figures

List of Tables

Abstract:

In this project, enhanced quantum inspired fuzzy based neural network algorithm is used for signature verification (E-QFNNS). Here, neural network architecture forms constructively using the quantum computing concept. The quantum computing concept is used to decide the connection weights and threshold of neurons with the help of the boundary threshold parameter. To test the performance of the proposed algorithm, the Iris and Thyroid dataset has been used. The hardware implementation is done following the basic VLSI design flow i.e. divided in four sections; Design Entry, Design Synthesis, Design Implementation and Xilinx Device Programming(specific to the Xilinx board). At start the design synthesis and design implementation is being handled by the Vivado HLS. The testing code is being generated for the algorithm and is implemented of Vivado HLS, which gave a general idea of resource utilisation, latency and. time utilisation At the later stage of project the performance is then compared to the hand coded VHDL implementation of the algorithm testing code. The next step after hardware implementation of the testing code in HLS and VHDL will be the hardware implementation of the training code into first HLS software tool and then with the analysis of its resource utilisation it will finally be coded in VHDL and then tested with the generated signature dataset. Once the hardware implementation is tested with the dataset it will finally be burned on the hardware board.

Chapter 1

Introduction

The signature has been an ancient biometric hallmark and has been used for authenticating individuals and documents. For the last decades, scientists and researchers had worked in the field of offline signature verification. Several methods and technologies have been proposed in this area some of them include elastic matching (Buryne and Forre, 1986), synthetic discriminant functions (Wilkinson et al., 1991) and grid features (Qi and Hunt, 1994). Signature verification is a technique used by banks, intelligence agencies and high-profile institutions to validate the identity of an individual. It is often used to compare signatures in bank offices and other branch capture. An image of a signature or a direct signature is fed into the signature verification software and compared to the signature image on file. Signature verification can be done by software that compares signatures and checks for authenticity. This saves time and energy and helps to prevent human error during the signature process and lowers chances of fraud in the process of authentication. The software generates a confidence score against the signature to be verified. Too low of a confidence score means the signature is most likely a forgery.

Signature verification software has now become lightweight, fast, flexible and more reliable with multiple options for storage, multiple signatures against one ID and a huge database. It can automatically search for a signature within an image or file.

Server-aided verification (SAV) has potential applicability in lightweight devices for improving signature verification, where the verifier possesses a computationally weak hardware. We observe that lightweight devices run all algorithms through hardware implementation with logic circuits. Existing SAV protocols indeed improve computational efficiency for lightweight devices, however, few of them take the hardware cost into consideration. The hardware implementation of SAV protocols could be still costly and expensive for lightweight devices. In this project we propose implementation of Quantum inspired Fuzzy based Neural Network Algorithm for offline signature verification aimed with small development cycle, lower cost and best suited for initial prototyping.

1.1 Motivation for Hardware Implementation

The main motivation to work on hardware implementation of the Quantum inspired Fuzzy Neural Network (QFNN) algorithm was a targeted effort towards the automation of signature verification/Offline signature verification in banking and trading industry. As the market is moving towards digitalisation of money and its transaction.

It'll be an effort for bridging the gap between people comfortable with the latest technology and the people still trusting the pen-paper based system. It can also have application directly in banks and also in cheque and draft deposit kiosk with some supporting applications. The need for offline hardware verification dedicated hardware was felt specially after the demonetisation of money was introduced in India, moving transaction from offline to online modes hence creating a large gap between the old and the latest technology.

1.2 Why Quantum Fuzzy Neural Network Algorithm

Researchers had presented many learning algorithms like perceptron, back propagation and multilayer perceptron for solving two classes as well as multi-class problems. It is also been studied that Fuzzy Logic algorithm is the best at computing cost and predictive accuracy as compared to other algorithms. The performance of neural network system mainly varies according to its architecture, hidden layers, the number of neurons in the hidden layer, connection weights, and threshold. It also depends on features of input dataset (Chan et al., 2009; Sarikaya et al., 2014). Here an enhanced quantum inspired fuzzy based neural network learning application is used for signaure verification. In this algorithm, the connection weights are decided using the quantum computing concept. In the algorithm paper, a further improvement has been proposed, which decides centres and fuzzification parameter using the quantum computing concept. The neural network formed in this way is trained and tested on the signature dataset, which is manually prepared under this project.

Chapter 2

Literature Review

2.1 Quantum Fuzzy Neural Network Algorithm

In this project work, an enhanced quantum inspired fuzzy based neural network learning algorithm is used which will be trained and tested on signature dataset. The proposed algorithm is divided into two major sections: constructing a quantum neural network and evaluating system performance by the signature database. Both the sections require some necessary pre-requisites, which are described briefly. The necessary preliminaries of quantum neural network and related basic of feature extraction from the signature dataset are explained in details.

This method forms a neural network architecture, which consists of three layers, input layer, one hidden layer and the output layer. The number of input nodes is equal to the number of attributes(features extracted) of signature dataset. Let $P_1 = (X_1;X_2;X_3;::::;X_c)$ denote the input samples, where c1 is the number of input samples and $X_i = (X_l;\ X_2;\ X_3;\ ::::;\ X_e)$ where e is the number of attributes in one instance of input sample. Therefore, the number of input layer node is equal to e . The number of neurons in the hidden layer is decided using the used constructive algorithm. For i^{th} hidden layer neuron, connection weights are denoted as follows:

$$W_i^{real} = (\ W_1\ ;W_2\ ;W_3\ ;\ ::::;W_e)$$

In the algorithm used, these connection weights are decided using the quantum computing concept. Its representation in terms of quantum bits is defined as follows:

$$W_i^{quant} = (Q_1;Q_2;Q_3;\ ::::::;Q_e)$$

Where, each W_i^j is denoted by Q_i^j , thus W_i^{real} can be represented as W_i^{quant}.

2.1.1 The Quantum Concept

For an appropriate selection of learning parameters, quantum computing concept is utilised (Gandhi et al., 2013). The parameters are represented in terms of a quantum bit (Q_i) . These quantum bits (Q_i) , are made up of several qubits q_{ij} (where j=1,2,....,k). Here, the k number of qubits which represent quantum bit (Q_i). A single qubit (q_{ij}) is the smallest unit of representing information.

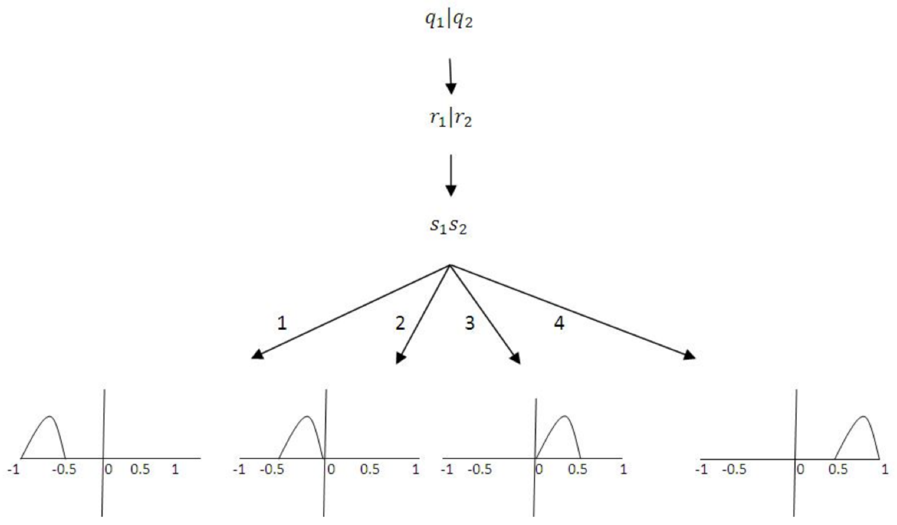


Figure 2.1: Conversion into qubits and further into subspaces

2.1.2 Conversion from quantum bits to real value

In the proposed algorithm, quantum based weights and threshold are required to convert into real coded value, so that it can be processed on classical computers. The weight matrix in terms of quantum bits W_i^{quant} , is converted into a real value weight matrix W_i^{real}. Similarly, the threshold value in terms of quantum bits, Th_i is converted into real value Th_i^{real} . This conversion process starts by taking random number matrices R , where $R_i = [r_1r_2::::r_k]$, corresponding to a quantum bit$Qi = (q_{i1j}\ q_{i2j}\ ::::::q_{ik}$). Then, further mapping is done by using binary matrix S_i where $S_i = [S_1\ S_2\ ::::S_k]$ and Gaussian random number generator with mean value (mue) and variance (sigma), which can be represented as N(mue,sigma) .The mapping between the binary value to the Gaussian number generator is done with the help of formula binary to decimal conversion. The value of matrix S_i is passed into *bin2dec*(Si) formula to select a value from a Gaussian random generator (Lu et al., 2013).

2.1.3 Fuzzy Concept

A fuzzy neural network or neuro-fuzzy system is a learning machine that finds the parameters of a fuzzy system (i.e., fuzzy sets, fuzzy rules) by exploiting approximation techniques from neural networks. Both neural networks and fuzzy systems have some things in common. They can be used for solving a problem (e.g. pattern recognition, regression or density estimation) if there does not exist any mathematical model of the given problem. They solely do have certain disadvantages and advantages which almost completely disappear by combining both concepts. Neural networks can only come into play if the problem is expressed by a sufficient amount of observed examples. These observations are used to train the black box. On the one hand no prior knowledge about the problem needs to be given. On the other hand, however, it is not straightforward to extract comprehensible rules from the neural network's structure.

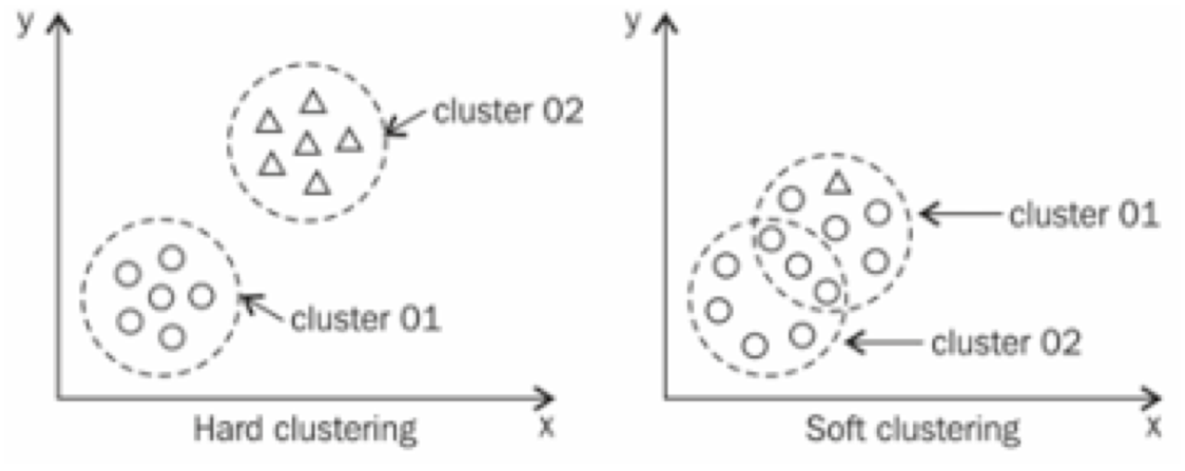


Figure 2.2: A Fuzzy Cluster

On the contrary, a fuzzy system demands linguistic rules instead of learning examples as prior knowledge. Furthermore the input and output variables have to be described linguistically. If the knowledge is incomplete, wrong or contradictory, then the fuzzy system must be tuned. Since there is not any formal approach for it, the tuning is performed in a heuristic way. This is usually very time consuming and error-prone(Scholarpedia).

2.1.5 Boundary Parameters Calculations

In the proposed algorithm, the threshold Th_i^{real} of neuron is evolved using quantum concept and boundary parameters. Here, to select threshold, min _net and max _net parameters are introduced. These parameters are initialised as infinity and -infinity respectively. Now these parameters are finding minimum and maximum value from the cartesian product of input sample of both classes and weights (W_1^{real}) . These values help to find out actual distribution of projection of input dataset with connection weights. Thus, its gives more diversity to search appropriate value of threshold and avoid local minima and maxima problem. This parameter finds out the boundary of the projection of input sample with connection weights help to select a threshold within this range. Thus, it helps to bind threshold in the defined range, which solve the problem of selecting a random threshold value and also saves the time of the learning process

2.2 Signature Processing and Feature Extraction

Before feeding the signature data for the generation of dataset and extraction of features from it, some preprocessing is being done as signatures can be done by inks of various colours with different kinds of pens or pencils. It is hence first converted into a basis black and white uniform thickness signature and then the signatures are extracted as will be explained further. Firstly, original and forged signatures of a few people are collected in hard copy, then these hard copies are scanned to get a signature in the form of images. To get the signature in the input form for the proposed quantum neural network first, the signature images are pre-processed then feature extraction is done. In this process, the signature images are first standardised so that the differences in signatures due to the variation in pen, discrepancies, and background noise are removed. The following steps are followed for the preprocessing and extracting the features from the signature image.

2.2.1 Conversion from RGB to B/W

To preprocess signature image first its colour is made uniform. Therefore, for each signature image which may have different colour is converted into a black and white image

2.2.2 Noise Removal

Once the image is converted into black and white the noise removal process start. The signature image can have noise due to two main sources: first, the background paper on which the signature is taken which may not be uniformly white. Secondly, the scanning of the signed hard copy into its soft form may lead to the addition of noise. This noise will hinder the training and testing of signatures and hence must be removed. Median Filtering is used as a remedy here. Median filtering is more effective than convolution methods.

2.2.3 Decreasing the thickness

Since the difference in the nib size of pen must not be a factor affecting the verification process, hence the thickness of pen strokes is reduced to a single pixel and the most intense pixel value is stored with their coordinated for feature extraction.

2.2.4 Feature Extraction

After the Preprocessing is being done on the input signatures, features will be extracted so as to put them up in the first layer of Neural Network such as

- Number of loops

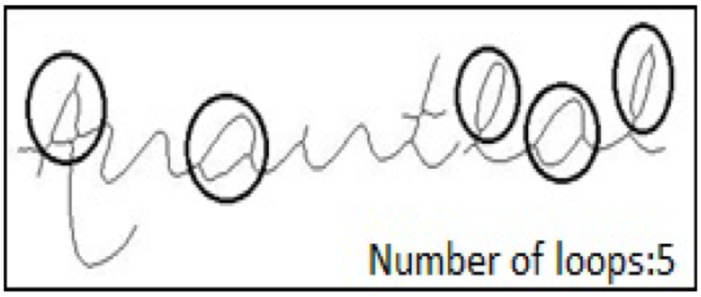


Figure2.3: Number of Loops

- Dimensions

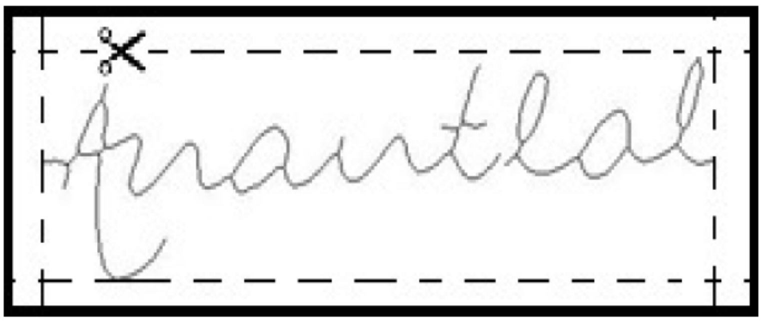


Figure 2.4: Dimensions or say Ration of dimensions

- Vertical Dense strip
- Horizontal Dense strip
- Dense Patch

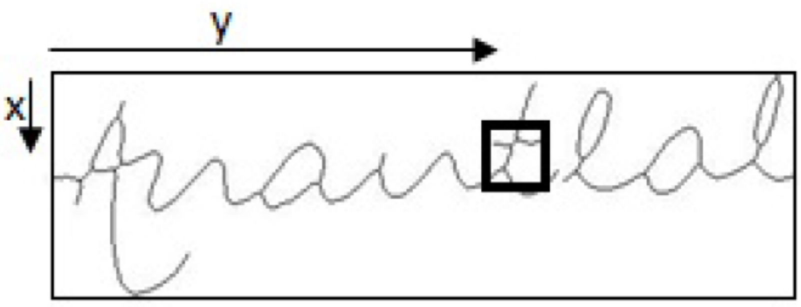


Figure 2.5: Dense Square

- Angle

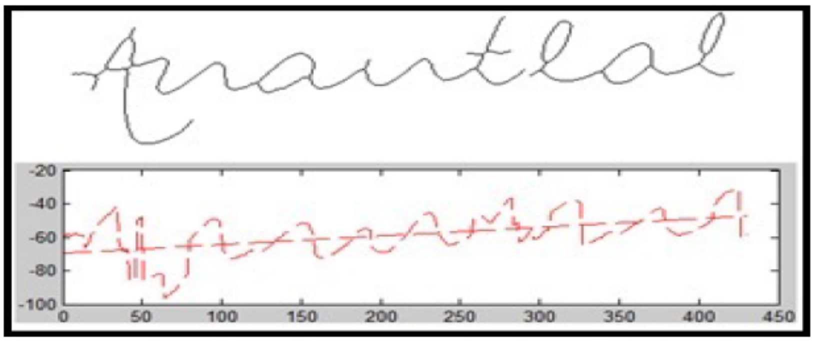


Figure 2.6: Slope of Signature

- Bounding Caps

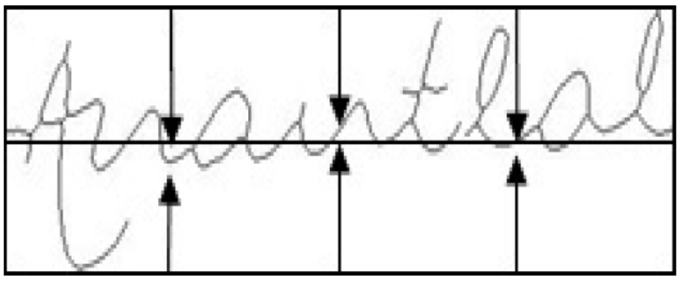


Figure 2.7: Image is divided in four uniform patches

2.3 Proposal for hardware implementation

We proposed on following the FPGA based approach for hardware implementation as it was better than the ASIC approach for testing and initial implementation as we were aiming at

- Comparable Performance.
- Small Development Cycle.
- Lower Cost.
- Possibility of performing algorithmic changes at the later stages of development.
- Best suited for initial prototyping

2.4 XST User Guide

At initial stage for the learning of hardware concept and introduction to HDL coding for coding hardware blocks, several reading of the Xylinx XST Used Guide were recommended. Xilinx® Synthesis Technology (XST) is a Xilinx application that synthesises Hardware Description Language (HDL) designs to create Xilinx-specific netlist files called NGC files. The NGC file is a netlist that contains both logical design data and constraints. The NGC file takes the place of both Electronic Data Interchange Format (EDIF) and Netlist Constraints File (NCF) files.

Designs are usually made up of combinatorial logic and macros such as flip-flops, adders, subtractors, counters, FSMs, and RAMs. The macros greatly improve performance of the synthesised designs. It is important to use coding techniques to model the macros so they are optimally processed by XST.
XST first tries to recognise (infer) as many macros as possible. These macros are then passed to the Low Level Optimisation step. In order to obtain better optimisation results, the macros are either preserved as separate blocks, or merged with surrounded logic. This filtering depends on the type and size of a macro. For example, by default, 2-to-1 multiplexers are not preserved by the optimisation engine. Synthesis constraints control the processing of inferred macros

2.4.1 XST HDL Coding Techniques

Before beginning the hardware implementation we first had to take an idea of HDL coding techniques to be utilised in FPGA implementation. The general sections in this context were:

- A general description of Macro processing
- Sample log files
- Constraints that can be used to control macro processing in XST
- VHDL and Verilog coding examples.

The chapter on HDL coding techniques included:

Signed and unsigned support in XST: When using Verilog or VHDL in XST, some macros, such as adders or counters, can be implemented for signed and unsigned values. For VHDL, depending on the operation and type of the operands, we must include additional packages in our code. To create an unsigned adder, use the arithmetic packages and types that operate on unsigned values shown in the table below:

PACKAGE	TYPE
numeric_std	unsigned
std_logic_arith	unsigned
std_logic_unsigned	std_logic_vector

Table 2.1 : Arithmetic packages and types that operate on unsigned values

To create a signed adder, use the arithmetic packages and types that operate on unsigned values shown in the table below:

PACKAGE	TYPE
numeric_std	signed
std_logic_arith	signed
std_logic_signed	std_logic_vector

Table 2.2 : arithmetic packages and types that operate on signed values

Coding techniques for different blocks :

- Register HDL Coding Technique
- Latches HDL Coding Technique
- Tristates HDL Coding Technique
- Counters HDL Coding Technique
- Accumulators HDL Coding Technique
- Shift Registers HDL Coding Technique
- Dynamic Shift Registers HDL Coding Technique
- Multiplexers HDL Coding Technique

2.4.2 XST Design Constraints

Constraints help you meet our design goals and obtain the best implementation of our circuit. Constraints control various aspects of synthesis, as well as placement and routing. Synthesis algorithms and heuristics automatically provide optimal results in most situations. If synthesis fails to initially achieve optimal results, we use available constraints to try other synthesis alternatives. VHDL attributes can be directly inserted into the VHDL code and attached to individual elements of the design to control both synthesis, and placement and routing.

The local specification of a constraint overrides its global setting. Similarly, if a constraint is set both on a node (or an instance) and on the enclosing design unit, the former takes precedence for the considered node (or instance).
Follow these general rules:

- Several constraints can be applied on signals. In this case, the constraint must be placed in the block where the signal is declared and used.

- If a constraint can be applied on an entity (VHDL), then it can also be applied on the component declaration. The ability to apply constraints on components is not explicitly stated for each individual constraint, since it is a general XST rule.

- Some third party synthesis tools allow you to apply constraints on architectures. XST allows constraints on architectures only for those third party constraints automatically supported by XST.

Generally there are many classifications and further sub classifications of XST Design constraints, but generally they are organised the the following types:

- XST general constraints
- XST HDL constraints
- XST FPGA constraints
- XST CPLD(complex programable logic device) constraints
- XST Timing constraints
- XST implementation constraints
- Third party constraints

2.4.3 XST VHDL Language Support

This chapter we learned how XST supports the VHDL hardware description language, and provides details on VHDL, supported constructs, and synthesis options. VHDL offers a broad set of constructs for compactly describing complicated logic:

- VHDL allows the description of the structure of a system — how it is decomposed into subsystems, and how those subsystems are interconnected.

- VHDL allows the specification of the function of a system using familiar programming language forms.

- VHDL allows the design of a system to be simulated before being implemented and manufactured. This feature allows you to test for correctness without the delay and expense of hardware prototyping.

- VHDL provides a mechanism for easily producing a detailed, device-dependent version of a design to be synthesised from a more abstract specification. This feature allows you to concentrate on more strategic design decisions, and reduce the overall time to market for the design.

Further in the chapter there was an extensive study of:

- VHDL IEEE support, explaining the IEEE supports, IEEE conflicts and how to resolve them and the Non-LRM Complaint constructs in VHDL.

- XST VHDL File type support explaining file read capabilities, file write capabilities for debugging processes or to write a specific constant or generic value to an external file.

- Debugging using write operation in VHDL

- VHDL Data Types

- VHDL Record Types
- VHDL Initial Values

2.5 Vivado Design Suite(UG902)

For the introduction of C-based FPGA design HLS was introduced as, the Xilinx High-Level Synthesis software Vivado HLS transforms a C specification into a Register Transfer Level (RTL) implementation that synthesises into a Xilinx Field Programmable Gate Array (FPGA).

High-Level Synthesis (HLS) bridges the software and hardware domains.

- Allows hardware designers who implement designs in an FPGA to take advantage of the productivity benefits of working at a higher level of abstraction, while creating high-performance hardware.
- Provides software developers with an easy way to accelerate the computationally intensive parts of their algorithms on a new compilation target, the FPGA provides a massively paralleled architecture with benefits in performance, cost and power over traditional processors.

The primary benefits of an HLS design methodology are improved productivity or hardware designers and improved system performance for software designers as follow:

- Develops algorithms at the C-level, which abstract you from the implementation details that consume development time.
- Verification at the C-level, which allows you to validate the functional correctness of the orders of magnitude faster than traditional hardware description languages allows.
- Controls the C synthesis process through optimisation directives allowing the creation of specific high-performance hardware implementations.
- Quickly create many different implementations from the C source code using optimisation directives which enables easy design space exploration and improves the likelihood of finding the most-optimal implementation

Using the HLS Design methodology also ensures read-able and portable C source code. You can re-target the C source into different FPGA devices as well as incorporated into newer projects.

The introduction section explains the basic concepts associated with High-Level Synthesis (HLS) and provides an overview of the usage and capabilities of Vivado HLS.

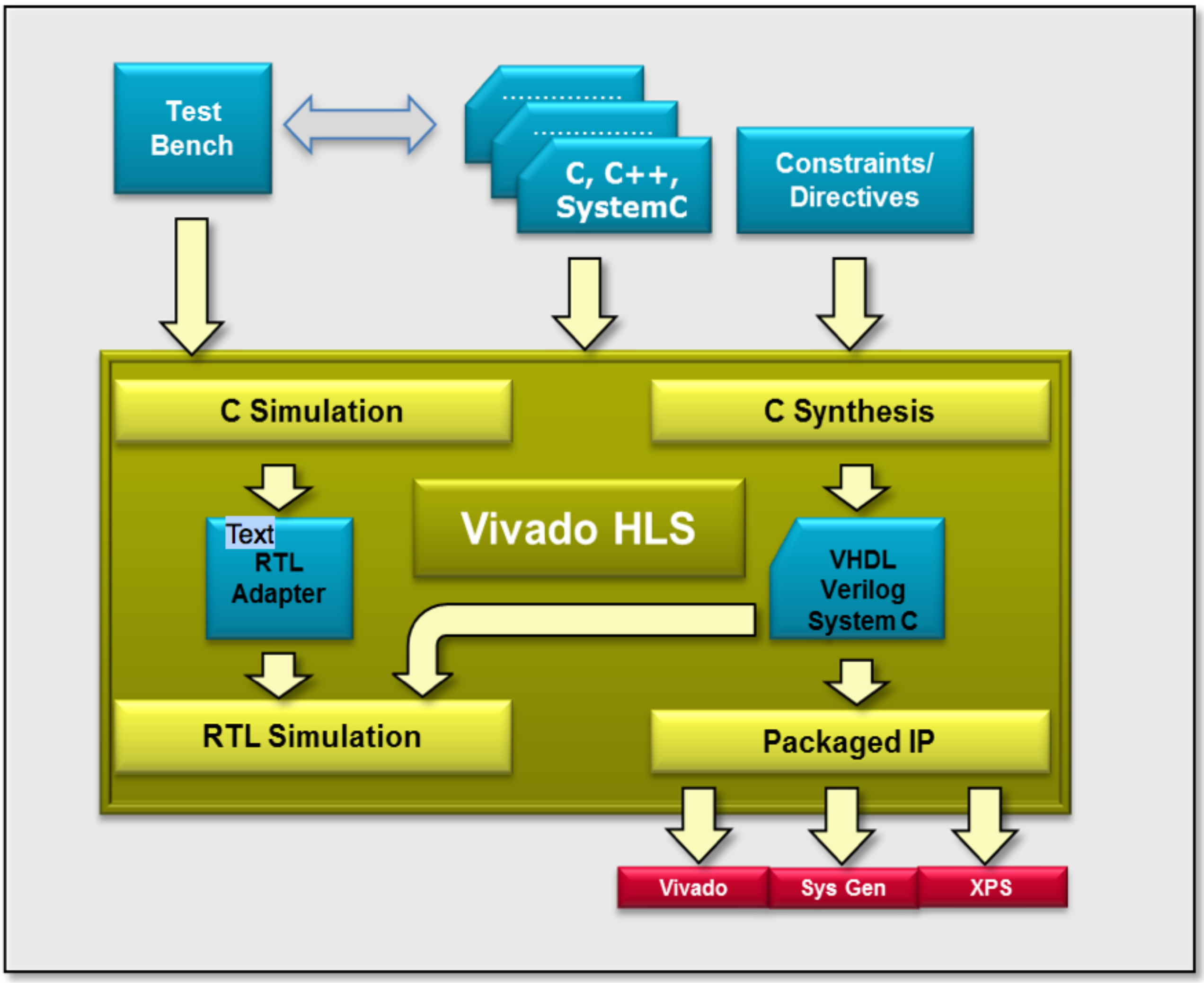


Figure 2.8 : HLS Overview

Further in this chapter the basics about Vivado HLS were explained such as how to use it(starting a new project, documentation, tutorials), HLS ultra fast design methodology as a key component in using a High-Level Synthesis design flow is to follow a good design methodology, managing interfaces which is very important to understand because in C based design, all input and output operations are performed, in zero time, through formal function arguments. In an RTL design these same input and output operations must be performed through a port in the design interface and typically operates using a specific I/O (input-output) protocol. Design optimisation outlining the various optimisations and techniques that can be employed to direct Vivado HLS to produce a micro-architecture that satisfies the desired performance, and area goals. RTL verification is a Post-synthesis verification is automated through the C/RTL co-simulation feature which re-uses the pre-synthesis C test bench to perform verification on the output RTL.. Finally exporting the RTL design, the final step in the Vivado HLS flow is to export the RTL design as a block of Intellectual Property (IP) which can be used by other tools in the Xilinx design flow.

2.6 Vivado HLS

Our respective study starts with the paper published in IEEE in 2016 naming High level synthesis using Vivado HLS for Zynq SOC : Image Processing Case Studies authored by A. Cortes et. al. The paper gives a comparative analysis between the synthesis reports of hand coded HDL and vivado HLS. They have shown some of the most common image processing algorithms : Data Binning, Sobel Filter, Step Row Filter. These algorithm have very high computational complexity. Authors tried to explain pros and cons of Vivado HLS. Author finds that time to market is less and handling of complex computational algorithm is quite easy in case of Vivado HLS. However the libraries of Vivado HLS significantly increases the necessary FPGA resources. Vivado HLS have some libraries similar to OpenCV which helps the user to develop projects even faster since software designers are familiar to these libraries. the data binning operation was implemented by using the Resize function of the hls_video_imgproc library. This function uses bi-linear interpolation to reduce the size of the input image to the size of the output image. The FPGA resources using the hls_video_imgproc library increase significantly with respect to the hand-coded solution.

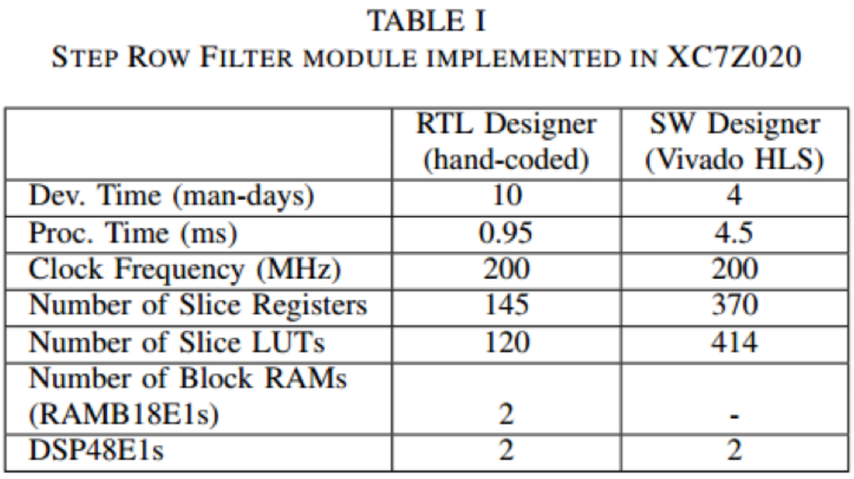


TABLE I
STEP ROW FILTER MODULE IMPLEMENTED IN XC7Z020

	RTL Designer (hand-coded)	SW Designer (Vivado HLS)
Dev. Time (man-days)	10	4
Proc. Time (ms)	0.95	4.5
Clock Frequency (MHz)	200	200
Number of Slice Registers	145	370
Number of Slice LUTs	120	414
Number of Block RAMs (RAMB18E1s)	2	-
DSP48E1s	2	2

Figure 3.3: Illustration 1: A. Cortes et. al.

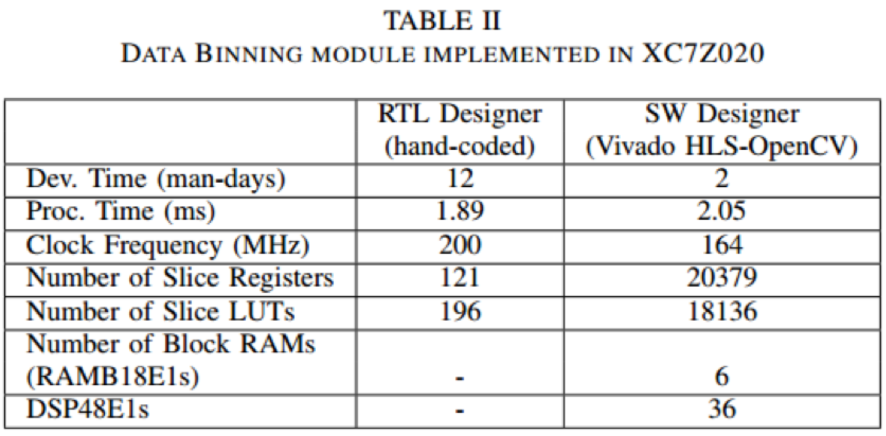


TABLE II
DATA BINNING MODULE IMPLEMENTED IN XC7Z020

	RTL Designer (hand-coded)	SW Designer (Vivado HLS-OpenCV)
Dev. Time (man-days)	12	2
Proc. Time (ms)	1.89	2.05
Clock Frequency (MHz)	200	164
Number of Slice Registers	121	20379
Number of Slice LUTs	196	18136
Number of Block RAMs (RAMB18E1s)	-	6
DSP48E1s	-	36

Figure 3.4: Illustration 2: A. Cortes et. al.

Based on these illustrations some conclusions were being made,

2.6.1 Usefulness of Vivado HLS

- Easier to implement computationally demanding algorithm.
- Higher Abstract level makes easy for normal software program to code for FPGA.
- Using a simple set of directives, multiple iterations are unrolled, each having its own loop body without the designer needing to worry about synchronisation issues as required in an HDL design.
- It is easier to pipeline the design. It reduces the complexity involved with the tracking register assignments.
- Flexibility — instantiation of multiple identical modules.

2.6.2 Vivado HLS are used by

- Companies who want to reduce time to market, want to use re-configurable logic and do not want to invest on VLSI design engineer.
- Researchers are using it for comparative analysis.
- Researchers are also using this tool to implement the algorithm and test its functionality on hardware before actually going for HDL code.
- Used for hardware software co-design problems (used at cern).

2.6.3 Problems with Vivado HLS

- Proper control is hard to find.
- Too much latency.
- Over utilisation of resources.

Chapter 3

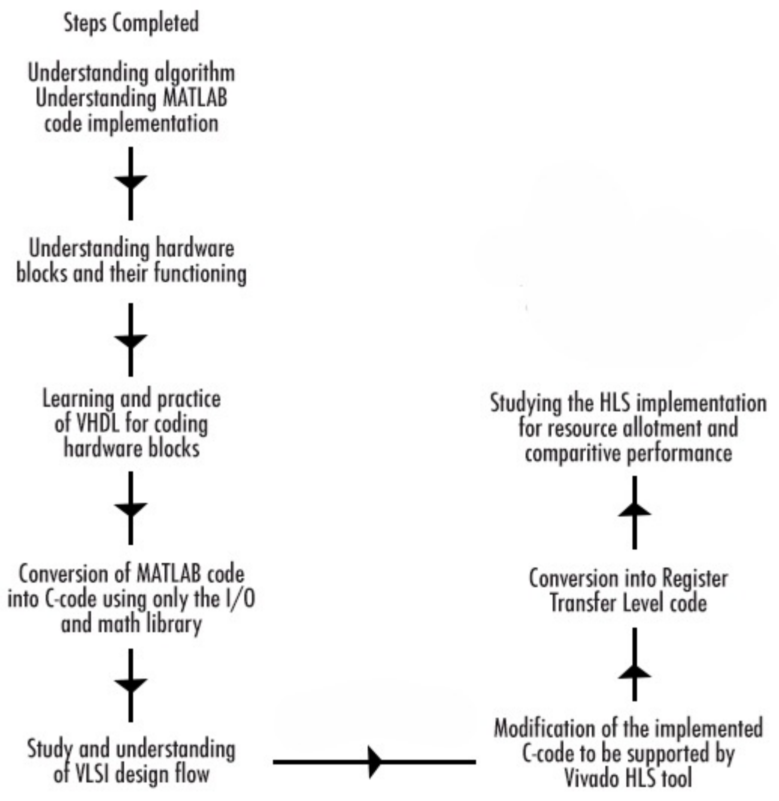
Design and Analysis

3.1 Overview

We started off the project work with the understanding of the Quantum Fuzzy Neural Network Algorithm and understanding the software implementation of the algorithm in MATLAB. For the next step we learned and practiced various hardware modules from basics to multiple blocks. Moving on to next step we were recommended to convert the presently available MATLAB implementation of the algorithm into a C-code for better understanding of all the blocks and flow of the code involved in the training, but conversion of MATLAB code into C-code was to be hardcoded line by without using any libraries except the standard input/output library and the math.h header file library. After the conversion of code we went on to understanding of VLSI hardware design flow to be followed for the hardware implementation. Design flow will be necessary so as to provide some algorithmic accelerations on the later stages of hardware implementation as well as it involves the study of optimal resource allotment. the work flow on project is also represented in the flow diagram on the next page.

3.2 VLSI Design Flow

Today, VLSI design flow is a very solid and mature process. The overall VLSI design flow and the various steps within the VLSI design flow have proven to be both practical and robust in multi-millions VLSI designs until now. Assuming your VLSI specifications are completed and approved by the different parties, it's time to start thinking about the architectural design. In VLSI system design phase, the entire chip functionality is broken down to small pieces with clear understanding about the block implementation. For example: for an encryption block, do you use a CPU or a state



Steps Completed
Understanding algorithm Understanding MATLAB code implementation
Understanding hardware blocks and their functioning
Learning and practice of VHDL for coding hardware blocks
Conversion of MATLAB code into C-code using only the I/O and math library
Study and understanding of VLSI design flow
Modification of the implemented C-code to be supported by Vivado HLS tool
Conversion into Register Transfer Level code
Studying the HLS implementation for resource allotment and comparitive performance

Figure 3.1: Flow of Project work

machine. Some other large blocks need to be divided into subsystems and the relationship between the various blocks has to be defined. In this phase the working environment is documentation. Each and every step of the VLSI design flow has a dedicated EDA tool that covers all the aspects related to the specific task perfectly. And most importantly, all the EDA tools can import and export the different file types to help making a flexible VLSI design flow that uses multiple tools. VLSI design flow can be better explained in the flow chart below:

3.2.1 Design Entry

This is the algorithmic stage of hardware implementation where frame the design of the algorithm. It includes converting the algorithm into a working VHDL(or any other hardware definition language) code. This part

was taken care in the stage of C-code generation and while modification of C-code into Vivado HLS supported code.

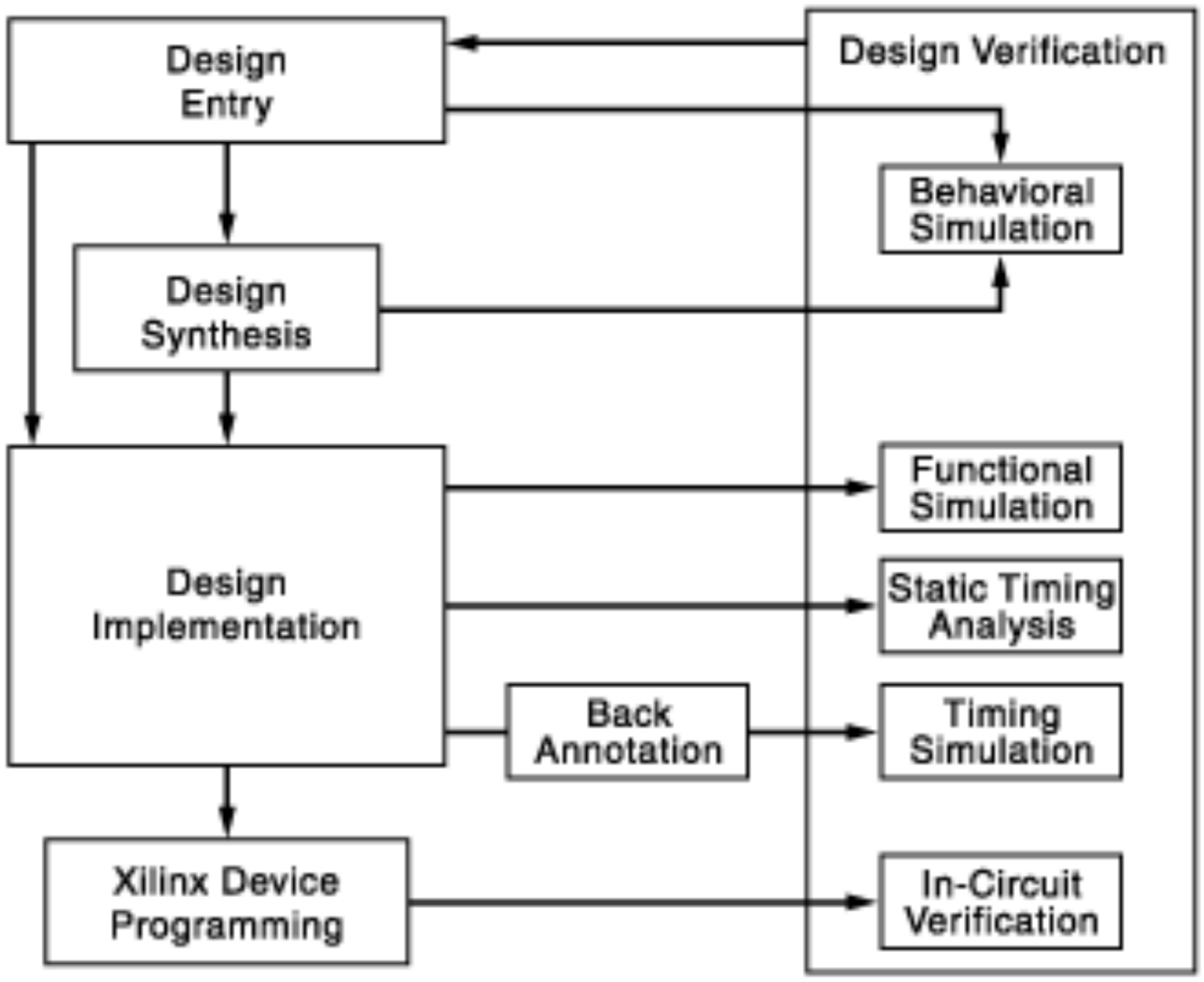


Figure 3.2: VLSI Design Flow

3.2.2 Design Synthesis

This stage essentially covers the conversion of hardware definition language code to Register Transfer Level(RTL), or say conversion of VHDL in RTL. Creation of logic elements takes place at this stage. this stage on initial level will be taken care by the HLS tool, generally this stage is done manually after the study of performance of HLS tool is being completed by an expert in VLSI design.

3.2.3 Design Implementation

This is the final placement routing stage, it includes where the logics are to be implemented, in what cell and how the connections are made so as to attain synchronisation and less latency. This stage too will be handled by the HLS tool for the initial hardware implementation.

3.3 C-code Conversion

In this project the major challenging part was coding of the algorithm into a C or C ++ code without using any libraries except the basic math.h and I/O libraries. As the Vivado HLS Math Library (hls_math.h) provides extensive support for the synthesis of the standard C (math.h) and C++ (cmath.h) libraries but not every function supported by the standard C math libraries is provided in the HLS Math Library. Some of the supported math functions are in the list below(which were used in the code) which are supported for synthesis:

- Exponential
- Logarithmic
- Trigonometric
- Modulus, Square,Sqrt, etc.

The C-code Implementation of the algorithm can be downloaded from *https://www.dropbox.com/sh/83rm0ufwy57dbed/AACBogu42ESHqZ6PhEsjmew5a?dl=0*

3.4 Testing code Generation

Next step after conversion of the C-code was generating Testing code in C. Initially, we read the IRIS data from the "iris.txt" file and split the training examples and category into two matrices 'X' and 'y' respectively.

We then create matrices 'Centres', 'Betas', 'Theta' which would store the values we calculated after training the classifier.

In the next step, we call the 'get_accuracy' function and pass the above matrices as parameters.
The 'get_accucracy' function will predict the category for each input row one by one by passing each of them to the 'evaluateRBFN' function. The predicted category is then compared to the actual answer and in case of a match, we increment the 'numRight' counter by one.

The final accuracy can be calculated by the formula: ((numRight)/(total)) * 100

The 'evaluateRBFN' function works in the following manner-
1. 'diffs' matrix is created which holds the absolute distance between each row of the 'Centres' matrix and the 'input' vector.
2. 'sqrdDists' array is calculated by squaring each element of the 'diffs' matrix and storing the sum of each row as the Ith element in the array.
3. 'phis' matrix is calculated by exp(-1 * 'Betas' * 'sqrdDists')
4. A row of ones is added to the top of the 'phis' matrix.
5. The transpose of theta is calculated and stored in the 'Theta_transpose' matrix.
6. The final scored matrix can be calculated by multiplying 'Theta_transpose' with 'phis' matrix.
7. In the 'scores' matrix, if the Ith row has the maximum values, then the predicted category is 'I'. (1-based indexing)

3.5 Testing code to Vivado HLS

Once the testing code was ready certain modifications had to be made in order for it to work on the Vivado HLS tool. All the inputs being given to the software were given as arrays and stored in the given Block RAM. First the Iris dataset matrix (150X4) was sent in as an array, following that Final_centers, Final_Thetas and Final_Betas matrices were given as input and stored in the block RAMs, then inputs were given from the test bench and data is classified one by one into their respective categories.

3.6 Analysis of obtained results with results generated from VHDL code.

The results obtained by synthesis of testing code done by Vivado HLS tool cab be compared with the results generated by the hand coded VHDL implementation of the testing code. Although the boards used in the software simulation were different but the resources used can be compared.

Accuracy obtained by VHDL implementation : 90.07%
Accuracy obtained by HLS Tool : 89.33%

Time taken by VHDL implementation : 0.11 Microseconds
Time taken by HLS Tool : 0.27 Microseconds

Total Slice LUTs utilised by VHDL implementation : 1538
Total Slice LUTs utilised by HLS Tool : 11662

Total DSP utilised by VHDL implementation : 87
Total DSP utilised by HLS Tool : 74

The above comparison between the results obtained by HLS tool and hand coded VHDL code clearly shows that for a smaller set of operation hand-coding components in VHDL is definitely a better option, but Vivado HLS tool does come in handy when the implementation becomes complicated and there is extensive use of mathematical functions. Therefore HLS tool was used as to compare the results generated by two different methods and it reduces programers involvement lower level languages which comes in handy with hardware implementation of complex algorithms.

Chapter 4

Results & Future Scope

The converted C-code was then executed and their results were compared to the existing MATLAB code, it was observed that there was a reduction is execution of time and dataset was learned at 100% accuracy.

Synthesis results observed by Vivado HLS

FPGA Device : **Virtex-7 xc7vx690tffg1761-2**
Software Tool : Vivado HLS 2016.4

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	908
FIFO	-	-	-	-
Instance	-	74	9413	10514
Memory	-	-	-	-
Multiplexer	-	-	-	230
Register	-	-	719	10
Total	0	74	10132	11662
Availible	2940	3600	866400	433200
Utilisation(%)	0	2	1	2

Table 4.1 : Resource utilisation by Vivado HLS

Time elapsed : .27 Microseconds
Time Elapsed In Software : 56 Microseconds
Accuracy obtained : 89.33
Note : The complete Report is generated and given in appendix 1.

Synthesis Results Obtained by VHDL hand coding

FPGA Device : **Zynq-7000 XC7Z020 CLG484-1 AP SoC**
Software Tool : Vivado 2016.4

Resources	Utilised	Availible	Utilisation (%)
Slice LUTs	1538	53200	2.89
Slice Registers	1259	106400	1.18
DSP	87	220	39.55

Table 4.2 : Resource utilisation by HDL Code

Maximum Clock Frequency = 222.2 MHz
Number of Clock Cycles = 24
Time Taken = 0.11 Microseconds (Speedup by 500 Times)
Time Taken In Software = 57.54 Microseconds
Accuracy on Iris Database = 90.07 %

Results of Training code

Before generating the testing code and then taking it to Vivado HLS synthesizable code profiling test using the thyroid dataset and iris dataset were done with the training code generated after conversion of code in C, to get an estimate of time utilisation of individual functions working in the code. The complete profiling test report is attached in appendix 2.

Hardware Generation for Q-FNN

With the understanding of the Q-FNN algorithm and its resource utilisation hardware profiling was done for the algorithm and the following VLSI Design was generated :

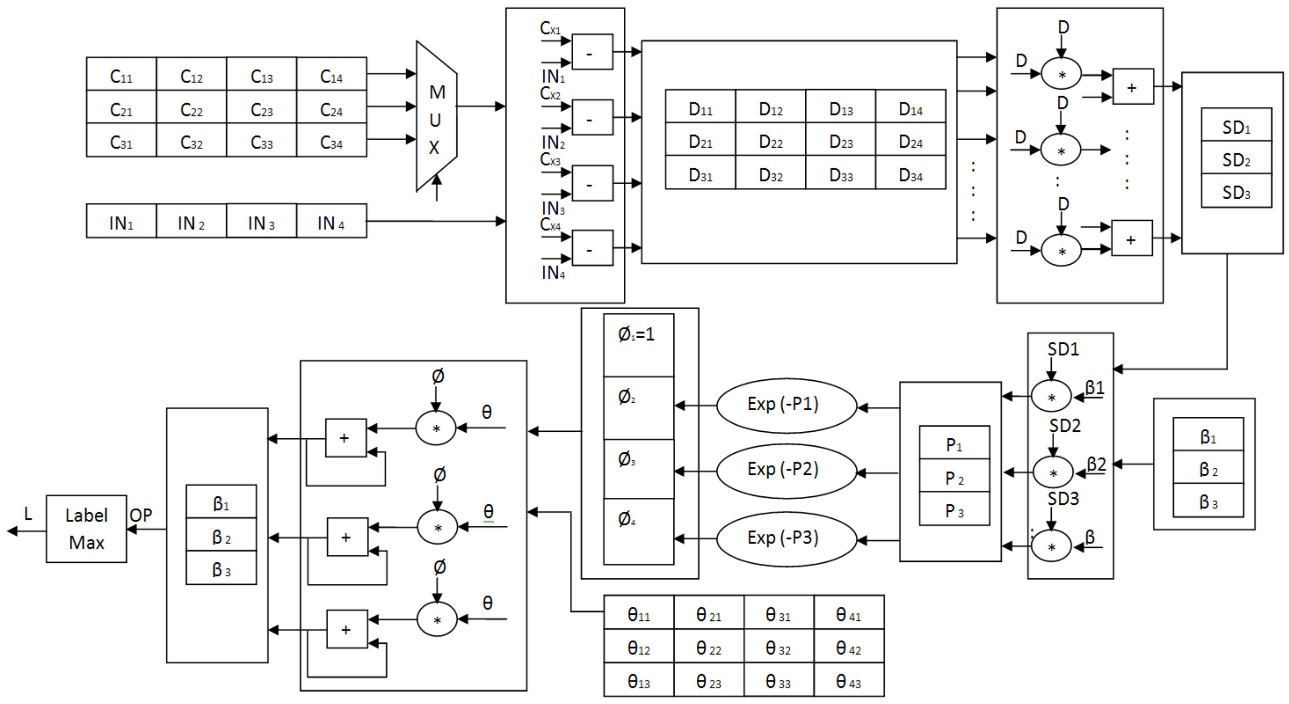


Figure 4.1 : Design and Implemented Architecture of Q-FNN

Future Scope

The works to be taken care in the remaining part of the project and future aspects of this project can be:

- FPGA implementation of complete QFNN algorithm in HDL
- FPGA implementation of complete QFNN algorithm in Vivado HLS
- Dedicated hardware for Signature verification using QFNN algorithm

Bibliography

Patel, O. P. and Tiwari, A. (2014) ‘Quantum inspired binary neural network algorithm’, IEEE International Conference on Information Technology (ICIT), Bhubaneswar, India , pp.270–274.

Buryne, P. d. and Forre, R. (1986) ‘Signature verification with elastic image matching’, IEEE International Carnahan Conference on Security Technology Gothenburg, Sweden, August

Gandhi,V., Prasad, G., Coyle, D., Behera, L., and McGinnity, T. M. (2013) ‘Quantum neural network-based eeg filtering for a brain-computer interface’, IEEE Transactions on Neural Networks and Learning Systems , Vol. 25, No. 2, pp.278–288.

Xilinx XST User Guide for basics of hardware blocks and block coding in VHLD. https://www.xilinx.com/itp/xilinx10/books/docs/xst/xst.pdf

Xilinx Vivado HLS UG902, UG871 and UG998 user guide for transformation of a C specification into a Register Transfer Level (RTL) implementation that synthesizes into a Xilinx Field Programmable Gate Array (FPGA). https://www.xilinx.com/support/documentation/sw_manuals/xilinx2014_1/ug902-vivado-high-level-synthesis.pdf

Xilinx XAPP 890 user guide for how to generate the Sobel edge detection filter in the Zynq™-7000 All Programmable SoC ZC702 Base Targeted Reference Design (TRD) using the Vivado™ High-Level Synthesis (HLS) tool.
https://www.xilinx.com/support/documentation/application_notes/xapp890-zynq-sobel-vivado-hls.pdf

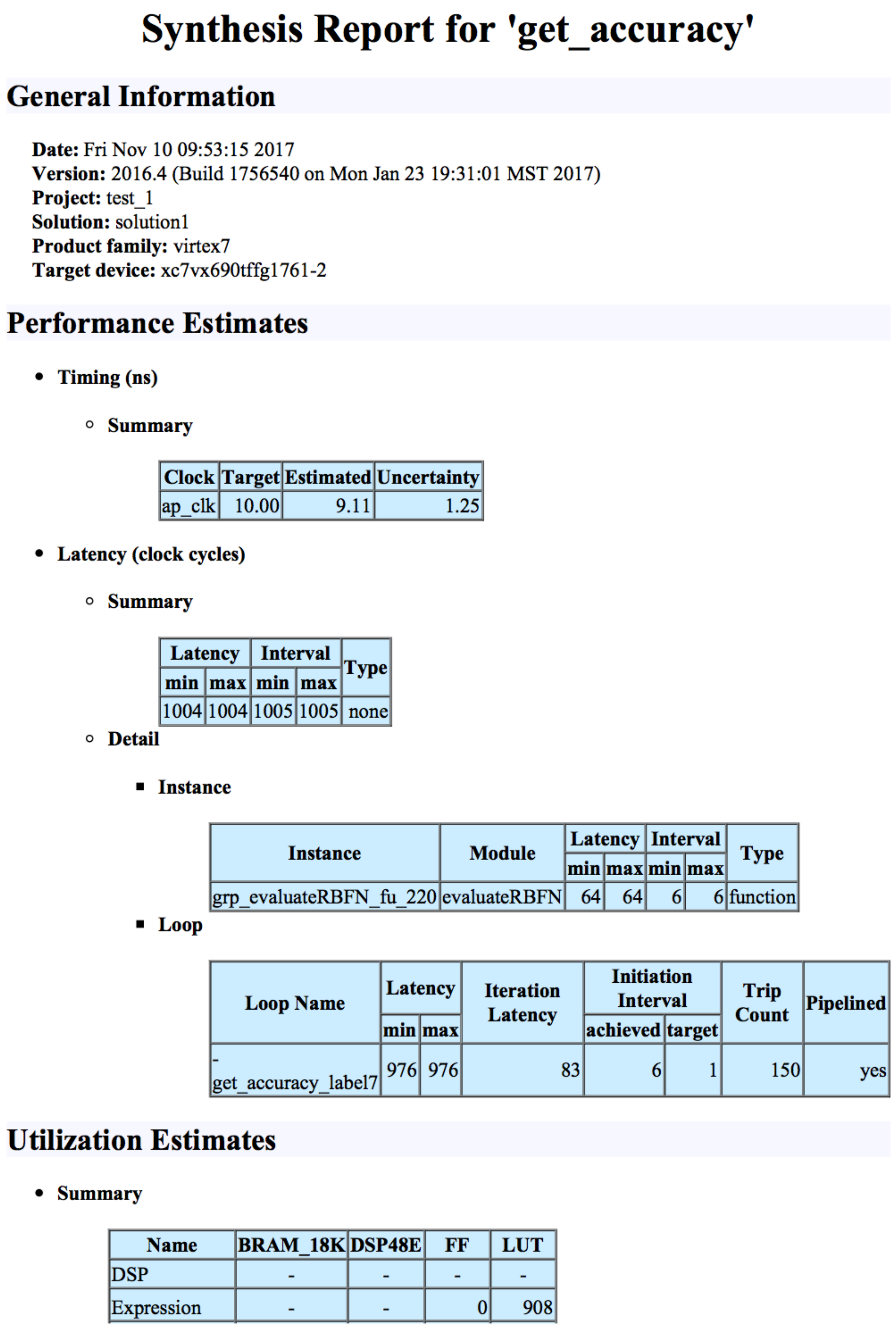
Xilinx AXI Interconnect for the information on how to connect one or more AXI memory-mapped Master devices to one or more memory-mapped Slave devices. https://www.xilinx.com/products/intellectual-property/axi_interconnect.html#overview

Wikipiedia : for basic study of Quantum and Fuzzy concepts, Physical design (electronics) for VLSI design and study of features and feature extraction for signature dataset.

anysilicon.com for instructions on VLSI design flow.

Scholorpedia for Concepts of Neural Networks and Comparisions in different algorithms.

Appendix 1



Synthesis Report for 'get_accuracy'

General Information

Date: Fri Nov 10 09:53:15 2017
Version: 2016.4 (Build 1756540 on Mon Jan 23 19:31:01 MST 2017)
Project: test_1
Solution: solution1
Product family: virtex7
Target device: xc7vx690tffg1761-2

Performance Estimates

- **Timing (ns)**
  - **Summary**

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	9.11	1.25

- **Latency (clock cycles)**
  - **Summary**

Latency		Interval		Type
min	max	min	max	
1004	1004	1005	1005	none

  - **Detail**
    - **Instance**

Instance	Module	Latency		Interval		Type
		min	max	min	max	
grp_evaluateRBFN_fu_220	evaluateRBFN	64	64	6	6	function

    - **Loop**

Loop Name	Latency		Iteration Latency	Initiation Interval		Trip Count	Pipelined
	min	max		achieved	target		
- get_accuracy_label7	976	976	83	6	1	150	yes

Utilization Estimates

- **Summary**

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	-	0	908

FIFO	-	-	-	-
Instance	-	74	9413	10514
Memory	-	-	-	-
Multiplexer	-	-	-	230
Register	-	-	719	10
Total	0	74	10132	11662
Available	2940	3600	866400	433200
Utilization (%)	0	2	1	2

- **Detail**
  - **Instance**

Instance	Module	BRAM_18K	DSP48E	FF	LUT
grp_evaluateRBFN_fu_220	evaluateRBFN	0	61	6768	6199
get_accuracy_ddivkbM_U28	get_accuracy_ddivkbM	0	0	1697	3253
get_accuracy_dmulg8j_x_U27	get_accuracy_dmulg8j	0	11	299	203
get_accuracy_fcmpjbC_U26	get_accuracy_fcmpjbC	0	0	66	72
get_accuracy_fpexfYi_x_U25	get_accuracy_fpexfYi	0	0	100	138
get_accuracy_fptreOg_x_U24	get_accuracy_fptreOg	0	0	128	94
get_accuracy_fsubbkb_x_U22	get_accuracy_fsubbkb	0	2	227	214
get_accuracy_sitoibs_U23	get_accuracy_sitoibs	0	0	128	341
Total	8	0	74	9413	10514

  - **DSP48**

N/A

  - **Memory**

N/A

  - **FIFO**

N/A

  - **Expression**

Variable Name	Operation	DSP48E	FF	LUT	Bitwidth P0	Bitwidth P1
i_1_fu_296_p2	+	0	0	8	8	1
idx_op_fu_608_p2	+	0	0	32	1	32
numRight_1_2_fu_864_p2	+	0	0	32	1	32
sh_assign_fu_693_p2	+	0	0	9	8	9
neg_fu_793_p2	-	0	0	32	1	32
p_Val2_7_i_i_fu_779_p2	-	0	0	32	1	32
tmp_4_i_i_fu_707_p2	-	0	0	8	7	8
tmp_13_fu_420_p2	and	0	0	1	1	1

tmp_20_fu_504_p2	and	0	0	1	1	1
tmp_22_fu_510_p2	and	0	0	1	1	1
tmp_29_fu_596_p2	and	0	0	1	1	1
tmp_31_fu_602_p2	and	0	0	1	1	1
tmp_38_fu_852_p2	and	0	0	1	1	1
tmp_40_fu_858_p2	and	0	0	1	1	1
abscond_fu_799_p2	icmp	0	0	11	32	1
exitcond2_fu_290_p2	icmp	0	0	3	8	8
notlhs1_fu_830_p2	icmp	0	0	3	8	2
notlhs2_fu_339_p2	icmp	0	0	3	7	2
notlhs3_fu_468_p2	icmp	0	0	3	8	2
notlhs5_fu_486_p2	icmp	0	0	3	8	2
notlhs7_fu_560_p2	icmp	0	0	3	8	2
notlhs9_fu_578_p2	icmp	0	0	3	8	2
notlhs_fu_402_p2	icmp	0	0	3	8	2
notrhs1_fu_584_p2	icmp	0	0	8	23	1
notrhs2_fu_836_p2	icmp	0	0	8	23	1
notrhs3_fu_284_p2	icmp	0	0	8	22	1
notrhs4_fu_474_p2	icmp	0	0	8	23	1
notrhs6_fu_492_p2	icmp	0	0	8	23	1
notrhs8_fu_566_p2	icmp	0	0	8	23	1
notrhs_fu_408_p2	icmp	0	0	8	23	1
tmp_7_i_i_fu_737_p2	lshr	0	0	65	24	24
tmp_11_fu_414_p2	or	0	0	1	1	1
tmp_18_fu_480_p2	or	0	0	1	1	1
tmp_19_fu_498_p2	or	0	0	1	1	1
tmp_24_fu_315_p2	or	0	0	15	10	1
tmp_27_fu_572_p2	or	0	0	1	1	1
tmp_28_fu_590_p2	or	0	0	1	1	1
tmp_2_fu_629_p2	or	0	0	1	1	1
tmp_33_fu_345_p2	or	0	0	15	10	2
tmp_36_fu_842_p2	or	0	0	1	1	1
tmp_37_fu_848_p2	or	0	0	1	1	1
tmp_41_fu_359_p2	or	0	0	15	10	2
abs_fu_805_p3	select	0	0	32	1	32
idx_1_fu_634_p3	select	0	0	32	1	32
idx_2_0_op_fu_614_p3	select	0	0	32	1	1
idx_2_1_op_cast_cast_fu_621_p3	select	0	0	2	1	2
maxscore_1_1_fu_516_p3	select	0	0	32	1	32
maxscore_1_fu_425_p3	select	0	0	32	1	32
numRight_1_1_fu_870_p3	select	0	0	32	1	32
p_Val2_3_fu_771_p3	select	0	0	32	1	32
p_Val2_5_fu_785_p3	select	0	0	32	1	32
sh_assign_1_fu_717_p3	select	0	0	9	1	9
tmp_9_i_i_fu_743_p2	shl	0	0	272	78	78

Total	53	0	0	908	439	534

- **Multiplexer**

Name	LUT	Input Size	Bits	Total Bits
X_address0	10	3	10	30
X_address1	10	3	10	30
ap_NS_fsm	60	35	1	35
ap_enable_reg_pp0_iter13	1	2	1	2
grp_fu_240_p0	32	4	32	128
grp_fu_251_opcode	5	3	5	15
grp_fu_251_p0	32	5	32	160
grp_fu_251_p1	32	5	32	160
i_phi_fu_199_p4	8	2	8	16
i_reg_195	8	2	8	16
numRight_1_reg_207	32	2	32	64
Total	230	66	171	656

- **Register**

Name	FF	LUT	Bits	Const Bits
abs_reg_1020	32	0	32	0
ap_CS_fsm	34	0	34	0
ap_enable_reg_pp0_iter0	1	0	1	0
ap_enable_reg_pp0_iter1	1	0	1	0
ap_enable_reg_pp0_iter10	1	0	1	0
ap_enable_reg_pp0_iter11	1	0	1	0
ap_enable_reg_pp0_iter12	1	0	1	0
ap_enable_reg_pp0_iter13	1	0	1	0
ap_enable_reg_pp0_iter2	1	0	1	0
ap_enable_reg_pp0_iter3	1	0	1	0
ap_enable_reg_pp0_iter4	1	0	1	0
ap_enable_reg_pp0_iter5	1	0	1	0
ap_enable_reg_pp0_iter6	1	0	1	0
ap_enable_reg_pp0_iter7	1	0	1	0
ap_enable_reg_pp0_iter8	1	0	1	0
ap_enable_reg_pp0_iter9	1	0	1	0
ap_reg_grp_evaluateRBFN_fu_220_ap_start	1	0	1	0
eps	0	0	32	32
eps_load_to_int_reg_889	0	0	32	32
exitcond2_reg_899	1	0	1	0
i_1_reg_903	8	0	8	0
i_reg_195	8	0	8	0
idx_1_reg_1000	32	0	32	0
idx_fu_118	32	0	32	0
irow_0_0_reg_939	32	0	32	0
irow_0_1_reg_944	32	0	32	0

maxscore_1_1_reg_994	32	0	32	0
notlhs2_reg_924	1	0	1	0
notrhs3_reg_894	1	0	1	0
numRight_1_reg_207	32	0	32	0
reg_266	32	0	32	0
scores_1_reg_966	32	0	32	0
scores_2_reg_973	32	0	32	0
scores_reg_959	32	0	32	0
tmp_12_reg_979	1	0	1	0
tmp_13_reg_984	1	0	1	0
tmp_15_reg_908	8	0	10	2
tmp_22_reg_989	1	0	1	0
tmp_4_reg_1041	64	0	64	0
tmp_5_reg_1046	64	0	64	0
tmp_9_reg_1025	32	0	32	0
tmp_s_reg_1036	64	0	64	0
x_assign_reg_1015	32	0	32	0
y_load_reg_1010	32	0	32	0
exitcond2_reg_899	0	1	1	0
i_reg_195	0	8	8	0
notlhs2_reg_924	0	1	1	0
Total	719	10	795	66

Interface

- **Summary**

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_hs	get_accuracy	return value
ap_rst	in	1	ap_ctrl_hs	get_accuracy	return value
ap_start	in	1	ap_ctrl_hs	get_accuracy	return value
ap_done	out	1	ap_ctrl_hs	get_accuracy	return value
ap_idle	out	1	ap_ctrl_hs	get_accuracy	return value
ap_ready	out	1	ap_ctrl_hs	get_accuracy	return value
X_address0	out	10	ap_memory	X	array
X_ce0	out	1	ap_memory	X	array
X_q0	in	32	ap_memory	X	array
X_address1	out	10	ap_memory	X	array
X_ce1	out	1	ap_memory	X	array
X_q1	in	32	ap_memory	X	array
y_address0	out	8	ap_memory	y	array
y_ce0	out	1	ap_memory	y	array
y_q0	in	32	ap_memory	y	array
Centers_address0	out	4	ap_memory	Centers	array
Centers_ce0	out	1	ap_memory	Centers	array
Centers_q0	in	32	ap_memory	Centers	array

Centers_address1	out	4	ap_memory	Centers	array
Centers_ce1	out	1	ap_memory	Centers	array
Centers_q1	in	32	ap_memory	Centers	array
Theta_address0	out	4	ap_memory	Theta	array
Theta_ce0	out	1	ap_memory	Theta	array
Theta_q0	in	32	ap_memory	Theta	array
Theta_address1	out	4	ap_memory	Theta	array
Theta_ce1	out	1	ap_memory	Theta	array
Theta_q1	in	32	ap_memory	Theta	array
Betas_address0	out	2	ap_memory	Betas	array
Betas_ce0	out	1	ap_memory	Betas	array
Betas_q0	in	32	ap_memory	Betas	array
Betas_address1	out	2	ap_memory	Betas	array
Betas_ce1	out	1	ap_memory	Betas	array
Betas_q1	in	32	ap_memory	Betas	array
accuracy	out	32	ap_vld	accuracy	pointer
accuracy_ap_vld	out	1	ap_vld	accuracy	pointer
numRight	out	32	ap_vld	numRight	pointer
numRight_ap_vld	out	1	ap_vld	numRight	pointer
total	out	32	ap_vld	total	pointer
total_ap_vld	out	1	ap_vld	total	pointer

Appendix 2(Profiling result Iris Dataset)

```
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  ms/call  ms/call  name
100.03      0.01     0.01        6     1.67     1.67  getTheta
  0.00      0.01     0.00     9583     0.00     0.00  sqr
  0.00      0.01     0.00     6831     0.00     0.00  getnew
  0.00      0.01     0.00     1554     0.00     0.00  parse
  0.00      0.01     0.00     1526     0.00     0.00  getRBFActivations
  0.00      0.01     0.00     1036     0.00     0.00  getCofactor
  0.00      0.01     0.00      793     0.00     0.00  multiply
  0.00      0.01     0.00      775     0.00     0.00  transpose
  0.00      0.01     0.00      763     0.00     0.00  evaluateRBFN
  0.00      0.01     0.00      148     0.00     0.00  randval
  0.00      0.01     0.00      104     0.00     0.00  lookup_table
  0.00      0.01     0.00       95     0.00     0.00  det
  0.00      0.01     0.00       34     0.00     0.00  readfile
  0.00      0.01     0.00       30     0.00     0.00  concat_row_wise
  0.00      0.01     0.00       24     0.00     0.00  writefile
  0.00      0.01     0.00        6     0.00     0.00  GetCenters
  0.00      0.01     0.00        6     0.00     0.00  adjoint
  0.00      0.01     0.00        6     0.00     0.00  euclidean
  0.00      0.01     0.00        6     0.00     0.00  get_accuracy
  0.00      0.01     0.00        6     0.00     0.00  get_betas
  0.00      0.01     0.00        6     0.00     0.00  get_kmat
  0.00      0.01     0.00        6     0.00     0.00  get_m
  0.00      0.01     0.00        6     0.00     0.00  get_mem_mat
  0.00      0.01     0.00        6     0.00     0.00  getrandnew
  0.00      0.01     0.00        6     0.00     0.00  initCentersUsingQ
  0.00      0.01     0.00        6     0.00     0.00  inverse
  0.00      0.01     0.00        3     0.00     0.00  unique_elements
  0.00      0.01     0.00        2     0.00     0.00  get_max
  0.00      0.01     0.00        2     0.00     0.00  init_center_quantum
  0.00      0.01     0.00        2     0.00     0.00  init_m_quantum
  0.00      0.01     0.00        2     0.00     5.00  trainRBFN_final
  0.00      0.01     0.00        1     0.00     0.00  in_dat
  0.00      0.01     0.00        1     0.00    10.00  train_final

 %         the percentage of the total running time of the
time       program used by this function.

cumulative a running sum of the number of seconds accounted
 seconds   for by this function and those listed above it.

 self      the number of seconds accounted for by this
seconds    function alone.  This is the major sort for this
           listing.

calls      the number of times this function was invoked, if
           this function is profiled, else blank.

 self      the average number of milliseconds spent in this
ms/call    function per call, if this function is profiled,
           else blank.

 total     the average number of milliseconds spent in this
ms/call    function and its descendents per call, if this
           function is profiled, else blank.
```

```
name       the name of the function.  This is the minor sort
          for this listing. The index shows the location of
              the function in the gprof listing. If the index is
              in parenthesis it shows where it would appear in
              the gprof listing if it were to be printed.
```

```
                         Call graph (explanation follows)


granularity: each sample hit covers 2 byte(s) for 99.97% of 0.01 seconds

index % time    self  children    called     name
              0.01    0.00       6/6           trainRBFN_final [2]
[1]    100.0    0.01    0.00       6         getTheta [1]
              0.00    0.00     763/1554        parse [7]
              0.00    0.00     763/1526        getRBFActivations [8]
              0.00    0.00      30/6831        getnew [6]
              0.00    0.00      30/793         multiply [10]
              0.00    0.00      12/775         transpose [11]
              0.00    0.00       6/6           inverse [29]
-----------------------------------------------
              0.00    0.01       2/2           train_final [3]
[2]    100.0    0.00    0.01       2         trainRBFN_final [2]
              0.01    0.00       6/6           getTheta [1]
              0.00    0.00     130/9583        sqr [5]
              0.00    0.00     104/104         lookup_table [14]
              0.00    0.00      42/6831        getnew [6]
              0.00    0.00      22/34          readfile [16]
              0.00    0.00      10/24          writefile [18]
              0.00    0.00       6/6           get_m [25]
              0.00    0.00       6/6           initCentersUsingQ [28]
              0.00    0.00       6/6           get_kmat [24]
              0.00    0.00       6/6           get_mem_mat [26]
              0.00    0.00       6/6           GetCenters [19]
              0.00    0.00       6/6           get_betas [23]
              0.00    0.00       6/6           get_accuracy [22]
              0.00    0.00       2/2           init_center_quantum [32]
              0.00    0.00       2/2           init_m_quantum [33]
-----------------------------------------------
              0.00    0.01       1/1           main [4]
[3]    100.0    0.00    0.01       1         train_final [3]
              0.00    0.01       2/2           trainRBFN_final [2]
              0.00    0.00      30/30          concat_row_wise [17]
              0.00    0.00      28/1554        parse [7]
              0.00    0.00      14/6831        getnew [6]
              0.00    0.00       1/1           in_dat [34]
              0.00    0.00       1/3           unique_elements [30]
-----------------------------------------------
                                             <spontaneous>
[4]    100.0    0.00    0.01                 main [4]
              0.00    0.01       1/1           train_final [3]
-----------------------------------------------
              0.00    0.00     130/9583        trainRBFN_final [2]
```

```
               0.00    0.00     136/9583        initCentersUsingQ [28]
               0.00    0.00     213/9583        get_betas [23]
               0.00    0.00    9104/9583        get_kmat [24]
[5]      0.0    0.00    0.00    9583        sqr [5]
-----------------------------------------------
               0.00    0.00       1/6831        in_dat [34]
               0.00    0.00       4/6831        init_center_quantum [32]
               0.00    0.00       6/6831        adjoint [20]
               0.00    0.00       6/6831        euclidean [21]
               0.00    0.00       6/6831        get_mem_mat [26]
               0.00    0.00       6/6831        get_m [25]
               0.00    0.00       6/6831        GetCenters [19]
               0.00    0.00       6/6831        get_accuracy [22]
               0.00    0.00      12/6831        inverse [29]
               0.00    0.00      12/6831        get_kmat [24]
               0.00    0.00      12/6831        initCentersUsingQ [28]
               0.00    0.00      12/6831        get_betas [23]
               0.00    0.00      14/6831        train_final [3]
               0.00    0.00      30/6831        concat_row_wise [17]
               0.00    0.00      30/6831        getTheta [1]
               0.00    0.00      34/6831        readfile [16]
               0.00    0.00      42/6831        trainRBFN_final [2]
               0.00    0.00     418/6831        det [15]
               0.00    0.00     775/6831        transpose [11]
               0.00    0.00     793/6831        multiply [10]
               0.00    0.00    1554/6831        parse [7]
               0.00    0.00    3052/6831        getRBFActivations [8]
[6]      0.0    0.00    0.00    6831        getnew [6]
-----------------------------------------------
               0.00    0.00      28/1554        train_final [3]
               0.00    0.00     763/1554        getTheta [1]
               0.00    0.00     763/1554        get_accuracy [22]
[7]      0.0    0.00    0.00    1554        parse [7]
               0.00    0.00    1554/6831        getnew [6]
-----------------------------------------------
               0.00    0.00     763/1526        getTheta [1]
               0.00    0.00     763/1526        evaluateRBFN [12]
[8]      0.0    0.00    0.00    1526        getRBFActivations [8]
               0.00    0.00    3052/6831        getnew [6]
-----------------------------------------------
               0.00    0.00      89/1036        adjoint [20]
               0.00    0.00     947/1036        det [15]
[9]      0.0    0.00    0.00    1036        getCofactor [9]
-----------------------------------------------
               0.00    0.00      30/793         getTheta [1]
               0.00    0.00     763/793         evaluateRBFN [12]
[10]     0.0    0.00    0.00     793        multiply [10]
               0.00    0.00     793/6831        getnew [6]
-----------------------------------------------
               0.00    0.00      12/775         getTheta [1]
               0.00    0.00     763/775         evaluateRBFN [12]
[11]     0.0    0.00    0.00     775        transpose [11]
               0.00    0.00     775/6831        getnew [6]
-----------------------------------------------
               0.00    0.00     763/763         get_accuracy [22]
[12]     0.0    0.00    0.00     763        evaluateRBFN [12]
               0.00    0.00     763/1526        getRBFActivations [8]
               0.00    0.00     763/775         transpose [11]
               0.00    0.00     763/793         multiply [10]
-----------------------------------------------
```

```
               0.00    0.00      12/148         get_m [25]
               0.00    0.00      68/148         getrandnew [27]
               0.00    0.00      68/148         initCentersUsingQ [28]
[13]     0.0    0.00    0.00     148         randval [13]
-----------------------------------------------
               0.00    0.00     104/104         trainRBFN_final [2]
[14]     0.0    0.00    0.00     104         lookup_table [14]
-----------------------------------------------
                                 947             det [15]
               0.00    0.00       6/95          inverse [29]
               0.00    0.00      89/95          adjoint [20]
[15]     0.0    0.00    0.00      95+947     det [15]
               0.00    0.00     947/1036        getCofactor [9]
               0.00    0.00     418/6831        getnew [6]
                                 947             det [15]
-----------------------------------------------
               0.00    0.00       6/34          get_m [25]
               0.00    0.00       6/34          initCentersUsingQ [28]
               0.00    0.00      22/34          trainRBFN_final [2]
[16]     0.0    0.00    0.00      34         readfile [16]
               0.00    0.00      34/6831        getnew [6]
-----------------------------------------------
               0.00    0.00      30/30          train_final [3]
[17]     0.0    0.00    0.00      30         concat_row_wise [17]
               0.00    0.00      30/6831        getnew [6]
-----------------------------------------------
               0.00    0.00       2/24          init_center_quantum [32]
               0.00    0.00       6/24          get_m [25]
               0.00    0.00       6/24          initCentersUsingQ [28]
               0.00    0.00      10/24          trainRBFN_final [2]
[18]     0.0    0.00    0.00      24         writefile [18]
-----------------------------------------------
               0.00    0.00       6/6           trainRBFN_final [2]
[19]     0.0    0.00    0.00       6         GetCenters [19]
               0.00    0.00       6/6831        getnew [6]
-----------------------------------------------
               0.00    0.00       6/6           inverse [29]
[20]     0.0    0.00    0.00       6         adjoint [20]
               0.00    0.00      89/1036        getCofactor [9]
               0.00    0.00      89/95          det [15]
               0.00    0.00       6/6831        getnew [6]
-----------------------------------------------
               0.00    0.00       6/6           get_kmat [24]
[21]     0.0    0.00    0.00       6         euclidean [21]
               0.00    0.00       6/6831        getnew [6]
-----------------------------------------------
               0.00    0.00       6/6           trainRBFN_final [2]
[22]     0.0    0.00    0.00       6         get_accuracy [22]
               0.00    0.00     763/1554        parse [7]
               0.00    0.00     763/763         evaluateRBFN [12]
               0.00    0.00       6/6831        getnew [6]
-----------------------------------------------
               0.00    0.00       6/6           trainRBFN_final [2]
[23]     0.0    0.00    0.00       6         get_betas [23]
               0.00    0.00     213/9583        sqr [5]
               0.00    0.00      12/6831        getnew [6]
-----------------------------------------------
               0.00    0.00       6/6           trainRBFN_final [2]
[24]     0.0    0.00    0.00       6         get_kmat [24]
               0.00    0.00    9104/9583        sqr [5]
```

```
                0.00    0.00      12/6831       getnew [6]
                0.00    0.00       6/6         euclidean [21]
-----------------------------------------------
                0.00    0.00       6/6         trainRBFN_final [2]
[25]    0.0    0.00    0.00       6        get_m [25]
                0.00    0.00      12/148        randval [13]
                0.00    0.00       6/34         readfile [16]
                0.00    0.00       6/6831       getnew [6]
                0.00    0.00       6/24         writefile [18]
-----------------------------------------------
                0.00    0.00       6/6         trainRBFN_final [2]
[26]    0.0    0.00    0.00       6        get_mem_mat [26]
                0.00    0.00       6/6831       getnew [6]
-----------------------------------------------
                0.00    0.00       6/6         initCentersUsingQ [28]
[27]    0.0    0.00    0.00       6        getrandnew [27]
                0.00    0.00      68/148        randval [13]
-----------------------------------------------
                0.00    0.00       6/6         trainRBFN_final [2]
[28]    0.0    0.00    0.00       6        initCentersUsingQ [28]
                0.00    0.00     136/9583        sqr [5]
                0.00    0.00      68/148        randval [13]
                0.00    0.00      12/6831       getnew [6]
                0.00    0.00       6/6         getrandnew [27]
                0.00    0.00       6/34         readfile [16]
                0.00    0.00       6/24         writefile [18]
-----------------------------------------------
                0.00    0.00       6/6         getTheta [1]
[29]    0.0    0.00    0.00       6        inverse [29]
                0.00    0.00      12/6831       getnew [6]
                0.00    0.00       6/95         det [15]
                0.00    0.00       6/6         adjoint [20]
-----------------------------------------------
                0.00    0.00       1/3         train_final [3]
                0.00    0.00       2/3         init_center_quantum [32]
[30]    0.0    0.00    0.00       3        unique_elements [30]
-----------------------------------------------
                0.00    0.00       2/2         init_center_quantum [32]
[31]    0.0    0.00    0.00       2        get_max [31]
-----------------------------------------------
                0.00    0.00       2/2         trainRBFN_final [2]
[32]    0.0    0.00    0.00       2        init_center_quantum [32]
                0.00    0.00       4/6831       getnew [6]
                0.00    0.00       2/2         get_max [31]
                0.00    0.00       2/3         unique_elements [30]
                0.00    0.00       2/24         writefile [18]
-----------------------------------------------
                0.00    0.00       2/2         trainRBFN_final [2]
[33]    0.0    0.00    0.00       2        init_m_quantum [33]
-----------------------------------------------
                0.00    0.00       1/1         train_final [3]
[34]    0.0    0.00    0.00       1        in_dat [34]
                0.00    0.00       1/6831       getnew [6]
-----------------------------------------------
```

This table describes the call tree of the program, and was sorted by the total amount of time spent in each function and its children.

Each entry in this table consists of several lines. The line with the index number at the left hand margin lists the current function.

```
The lines above it list the functions that called this function,
and the lines below it list the functions this one called.
This line lists:
    index       A unique number given to each element of the table.
                Index numbers are sorted numerically.
                The index number is printed next to every function name so
                it is easier to look up where the function is in the table.

    % time      This is the percentage of the `total' time that was spent
                in this function and its children.  Note that due to
                different viewpoints, functions excluded by options, etc,
                these numbers will NOT add up to 100%.

    self   This is the total amount of time spent in this function.

    children    This is the total amount of time propagated into this
                function by its children.

    called      This is the number of times the function was called.
                If the function called itself recursively, the number
                only includes non-recursive calls, and is followed by
                a `+' and the number of recursive calls.

    name        The name of the current function.  The index number is
                printed after it.  If the function is a member of a
                cycle, the cycle number is printed between the
                function's name and the index number.


For the function's parents, the fields have the following meanings:

    self   This is the amount of time that was propagated directly
                from the function into this parent.

    children    This is the amount of time that was propagated from
                the function's children into this parent.

    called      This is the number of times this parent called the
                function `/' the total number of times the function
                was called.  Recursive calls to the function are not
                included in the number after the `/'.

    name        This is the name of the parent.  The parent's index
                number is printed after it.  If the parent is a
                member of a cycle, the cycle number is printed between
                the name and the index number.

If the parents of the function cannot be determined, the word
`<spontaneous>' is printed in the `name' field, and all the other
fields are blank.

For the function's children, the fields have the following meanings:

    self   This is the amount of time that was propagated directly
                from the child into the function.

    children    This is the amount of time that was propagated from the
                child's children to the function.

    called      This is the number of times the function called
```

```
                    this child `/' the total number of times the child
                    was called.  Recursive calls by the child are not
                    listed in the number after the `/'.

     name           This is the name of the child.  The child's index
                    number is printed after it.  If the child is a
                    member of a cycle, the cycle number is printed
                    between the name and the index number.

 If there are any cycles (circles) in the call graph, there is an
 entry for the cycle-as-a-whole.  This entry shows who called the
 cycle (as parents) and the members of the cycle (as children.)
 The `+' recursive calls entry shows the number of function calls that
 were internal to the cycle, and the calls entry for each member shows,
 for that member, how many times it was called from other members of
 the cycle.

Copyright (C) 2012-2015 Free Software Foundation, Inc.

Copying and distribution of this file, with or without modification,
are permitted in any medium without royalty provided the copyright
notice and this notice are preserved.

Index by function name

  [19] GetCenters             [24] get_kmat               [14] lookup_table
  [20] adjoint                [25] get_m                  [10] multiply
  [17] concat_row_wise        [31] get_max                 [7] parse
  [15] det                    [26] get_mem_mat            [13] randval
  [21] euclidean               [6] getnew                 [16] readfile
  [12] evaluateRBFN           [27] getrandnew              [5] sqr
   [9] getCofactor            [34] in_dat                  [2] trainRBFN_final
   [8] getRBFActivations      [28] initCentersUsingQ       [3] train_final
   [1] getTheta               [32] init_center_quantum    [11] transpose
  [22] get_accuracy           [33] init_m_quantum         [30] unique_elements
  [23] get_betas              [29] inverse                [18] writefile
```

Appendix 3(Profiling Result Thyroid Dataset)

```
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  ms/call  ms/call  name
 50.01      0.01     0.01     2568     0.00     0.00  parse
 50.01      0.02     0.01       32     0.31     0.31  GetCenters
  0.00      0.02     0.00    54247     0.00     0.00  sqr
  0.00      0.02     0.00    11156     0.00     0.00  getnew
  0.00      0.02     0.00     2486     0.00     0.00  getRBFActivations
  0.00      0.02     0.00     1298     0.00     0.00  multiply
  0.00      0.02     0.00     1265     0.00     0.00  transpose
  0.00      0.02     0.00     1243     0.00     0.00  evaluateRBFN
  0.00      0.02     0.00     1216     0.00     0.00  getCofactor
  0.00      0.02     0.00      292     0.00     0.00  randval
  0.00      0.02     0.00      216     0.00     0.00  lookup_table
  0.00      0.02     0.00      145     0.00     0.00  det
  0.00      0.02     0.00       86     0.00     0.00  concat_row_wise
  0.00      0.02     0.00       64     0.00     0.00  readfile
  0.00      0.02     0.00       45     0.00     0.00  writefile
  0.00      0.02     0.00       32     0.00     0.00  euclidean
  0.00      0.02     0.00       32     0.00     0.00  get_kmat
  0.00      0.02     0.00       32     0.00     0.00  get_mem_mat
  0.00      0.02     0.00       11     0.00     0.00  adjoint
  0.00      0.02     0.00       11     0.00     0.44  getTheta
  0.00      0.02     0.00       11     0.00     0.44  get_accuracy
  0.00      0.02     0.00       11     0.00     0.00  get_betas
  0.00      0.02     0.00       11     0.00     0.00  get_m
  0.00      0.02     0.00       11     0.00     0.00  getrandnew
  0.00      0.02     0.00       11     0.00     0.00  initCentersUsingQ
  0.00      0.02     0.00       11     0.00     0.00  inverse
  0.00      0.02     0.00        4     0.00     0.00  unique_elements
  0.00      0.02     0.00        3     0.00     0.00  get_max
  0.00      0.02     0.00        3     0.00     0.00  init_center_quantum
  0.00      0.02     0.00        3     0.00     0.00  init_m_quantum
  0.00      0.02     0.00        3     0.00     6.56  trainRBFN_final
  0.00      0.02     0.00        1     0.00     0.00  in_dat
  0.00      0.02     0.00        1     0.00    20.01  train_final_thyroid

 %         the percentage of the total running time of the
time       program used by this function.

cumulative a running sum of the number of seconds accounted
 seconds   for by this function and those listed above it.

 self      the number of seconds accounted for by this
seconds    function alone.  This is the major sort for this
           listing.

calls      the number of times this function was invoked, if
           this function is profiled, else blank.

 self      the average number of milliseconds spent in this
ms/call    function per call, if this function is profiled,
           else blank.

 total     the average number of milliseconds spent in this
```

```
ms/call    function and its descendents per call, if this
           function is profiled, else blank.

name       the name of the function.  This is the minor sort
        for this listing. The index shows the location of
           the function in the gprof listing. If the index is
           in parenthesis it shows where it would appear in
           the gprof listing if it were to be printed.

Copyright (C) 2012-2015 Free Software Foundation, Inc.

Copying and distribution of this file, with or without modification,
are permitted in any medium without royalty provided the copyright
notice and this notice are preserved.

                     Call graph (explanation follows)


granularity: each sample hit covers 2 byte(s) for 49.99% of 0.02 seconds

index % time    self  children    called     name
                0.00    0.02       1/1           main [2]
[1]    100.0    0.00    0.02       1         train_final_thyroid [1]
                0.00    0.02       3/3           trainRBFN_final [3]
                0.00    0.00      82/2568          parse [4]
                0.00    0.00      86/86           concat_row_wise [18]
                0.00    0.00      20/11156          getnew [9]
                0.00    0.00       1/1           in_dat [34]
                0.00    0.00       1/4           unique_elements [30]
-----------------------------------------------
                                                 <spontaneous>
[2]    100.0    0.00    0.02                 main [2]
                0.00    0.02       1/1           train_final_thyroid [1]
-----------------------------------------------
                0.00    0.02       3/3           train_final_thyroid [1]
[3]     98.4    0.00    0.02       3         trainRBFN_final [3]
                0.01    0.00      32/32           GetCenters [5]
                0.00    0.00      11/11           getTheta [6]
                0.00    0.00      11/11           get_accuracy [7]
                0.00    0.00     270/54247          sqr [8]
                0.00    0.00     216/216           lookup_table [16]
                0.00    0.00      79/11156          getnew [9]
                0.00    0.00      42/64           readfile [19]
                0.00    0.00      32/32           get_kmat [22]
                0.00    0.00      32/32           get_mem_mat [23]
                0.00    0.00      20/45           writefile [20]
                0.00    0.00      11/11           get_m [26]
                0.00    0.00      11/11           initCentersUsingQ [28]
                0.00    0.00      11/11           get_betas [25]
                0.00    0.00       3/3           init_center_quantum [32]
                0.00    0.00       3/3           init_m_quantum [33]
-----------------------------------------------
                0.00    0.00      82/2568          train_final_thyroid [1]
                0.00    0.00    1243/2568          getTheta [6]
                0.00    0.00    1243/2568          get_accuracy [7]
[4]     50.0    0.01    0.00    2568         parse [4]
                0.00    0.00    2568/11156          getnew [9]
-----------------------------------------------
                0.01    0.00      32/32           trainRBFN_final [3]
[5]     50.0    0.01    0.00      32         GetCenters [5]
```

```
                0.00    0.00      32/11156       getnew [9]
-----------------------------------------------
                0.00    0.00      11/11          trainRBFN_final [3]
[6]     24.2    0.00    0.00      11         getTheta [6]
                0.00    0.00    1243/2568        parse [4]
                0.00    0.00    1243/2486        getRBFActivations [10]
                0.00    0.00      55/11156       getnew [9]
                0.00    0.00      55/1298        multiply [11]
                0.00    0.00      22/1265        transpose [12]
                0.00    0.00      11/11          inverse [29]
-----------------------------------------------
                0.00    0.00      11/11          trainRBFN_final [3]
[7]     24.2    0.00    0.00      11         get_accuracy [7]
                0.00    0.00    1243/2568        parse [4]
                0.00    0.00    1243/1243        evaluateRBFN [13]
                0.00    0.00      11/11156       getnew [9]
-----------------------------------------------
                0.00    0.00     270/54247       initCentersUsingQ [28]
                0.00    0.00     270/54247       trainRBFN_final [3]
                0.00    0.00     372/54247       get_betas [25]
                0.00    0.00   53335/54247       get_kmat [22]
[8]      0.0    0.00    0.00   54247         sqr [8]
-----------------------------------------------
                0.00    0.00       1/11156       in_dat [34]
                0.00    0.00       6/11156       init_center_quantum [32]
                0.00    0.00      11/11156       adjoint [24]
                0.00    0.00      11/11156       get_m [26]
                0.00    0.00      11/11156       get_accuracy [7]
                0.00    0.00      20/11156       train_final_thyroid [1]
                0.00    0.00      22/11156       inverse [29]
                0.00    0.00      22/11156       initCentersUsingQ [28]
                0.00    0.00      22/11156       get_betas [25]
                0.00    0.00      32/11156       euclidean [21]
                0.00    0.00      32/11156       get_mem_mat [23]
                0.00    0.00      32/11156       GetCenters [5]
                0.00    0.00      55/11156       getTheta [6]
                0.00    0.00      64/11156       readfile [19]
                0.00    0.00      64/11156       get_kmat [22]
                0.00    0.00      79/11156       trainRBFN_final [3]
                0.00    0.00      86/11156       concat_row_wise [18]
                0.00    0.00     483/11156       det [17]
                0.00    0.00    1265/11156       transpose [12]
                0.00    0.00    1298/11156       multiply [11]
                0.00    0.00    2568/11156       parse [4]
                0.00    0.00    4972/11156       getRBFActivations [10]
[9]      0.0    0.00    0.00   11156         getnew [9]
-----------------------------------------------
                0.00    0.00    1243/2486        getTheta [6]
                0.00    0.00    1243/2486        evaluateRBFN [13]
[10]     0.0    0.00    0.00    2486         getRBFActivations [10]
                0.00    0.00    4972/11156       getnew [9]
-----------------------------------------------
                0.00    0.00      55/1298        getTheta [6]
                0.00    0.00    1243/1298        evaluateRBFN [13]
[11]     0.0    0.00    0.00    1298         multiply [11]
                0.00    0.00    1298/11156       getnew [9]
-----------------------------------------------
                0.00    0.00      22/1265        getTheta [6]
                0.00    0.00    1243/1265        evaluateRBFN [13]
[12]     0.0    0.00    0.00    1265         transpose [12]
```

```
                0.00    0.00    1265/11156       getnew [9]
-----------------------------------------------
                0.00    0.00    1243/1243       get_accuracy [7]
[13]     0.0    0.00    0.00    1243        evaluateRBFN [13]
                0.00    0.00    1243/2486       getRBFActivations [10]
                0.00    0.00    1243/1265       transpose [12]
                0.00    0.00    1243/1298       multiply [11]
-----------------------------------------------
                0.00    0.00     134/1216       adjoint [24]
                0.00    0.00    1082/1216       det [17]
[14]     0.0    0.00    0.00    1216        getCofactor [14]
-----------------------------------------------
                0.00    0.00      22/292        get_m [26]
                0.00    0.00     135/292        getrandnew [27]
                0.00    0.00     135/292        initCentersUsingQ [28]
[15]     0.0    0.00    0.00     292        randval [15]
-----------------------------------------------
                0.00    0.00     216/216        trainRBFN_final [3]
[16]     0.0    0.00    0.00     216        lookup_table [16]
-----------------------------------------------
                                1082            det [17]
                0.00    0.00      11/145        inverse [29]
                0.00    0.00     134/145        adjoint [24]
[17]     0.0    0.00    0.00     145+1082    det [17]
                0.00    0.00    1082/1216       getCofactor [14]
                0.00    0.00     483/11156      getnew [9]
                                1082            det [17]
-----------------------------------------------
                0.00    0.00      86/86         train_final_thyroid [1]
[18]     0.0    0.00    0.00      86        concat_row_wise [18]
                0.00    0.00      86/11156      getnew [9]
-----------------------------------------------
                0.00    0.00      11/64         get_m [26]
                0.00    0.00      11/64         initCentersUsingQ [28]
                0.00    0.00      42/64         trainRBFN_final [3]
[19]     0.0    0.00    0.00      64        readfile [19]
                0.00    0.00      64/11156      getnew [9]
-----------------------------------------------
                0.00    0.00       3/45         init_center_quantum [32]
                0.00    0.00      11/45         get_m [26]
                0.00    0.00      11/45         initCentersUsingQ [28]
                0.00    0.00      20/45         trainRBFN_final [3]
[20]     0.0    0.00    0.00      45        writefile [20]
-----------------------------------------------
                0.00    0.00      32/32         get_kmat [22]
[21]     0.0    0.00    0.00      32        euclidean [21]
                0.00    0.00      32/11156      getnew [9]
-----------------------------------------------
                0.00    0.00      32/32         trainRBFN_final [3]
[22]     0.0    0.00    0.00      32        get_kmat [22]
                0.00    0.00   53335/54247      sqr [8]
                0.00    0.00      64/11156      getnew [9]
                0.00    0.00      32/32         euclidean [21]
-----------------------------------------------
                0.00    0.00      32/32         trainRBFN_final [3]
[23]     0.0    0.00    0.00      32        get_mem_mat [23]
                0.00    0.00      32/11156      getnew [9]
-----------------------------------------------
                0.00    0.00      11/11         inverse [29]
[24]     0.0    0.00    0.00      11        adjoint [24]
```

```
                0.00    0.00     134/1216        getCofactor [14]
                0.00    0.00     134/145         det [17]
                0.00    0.00      11/11156       getnew [9]
-----------------------------------------------
                0.00    0.00      11/11          trainRBFN_final [3]
[25]     0.0    0.00    0.00      11         get_betas [25]
                0.00    0.00     372/54247       sqr [8]
                0.00    0.00      22/11156       getnew [9]
-----------------------------------------------
                0.00    0.00      11/11          trainRBFN_final [3]
[26]     0.0    0.00    0.00      11         get_m [26]
                0.00    0.00      22/292         randval [15]
                0.00    0.00      11/64          readfile [19]
                0.00    0.00      11/11156       getnew [9]
                0.00    0.00      11/45          writefile [20]
-----------------------------------------------
                0.00    0.00      11/11          initCentersUsingQ [28]
[27]     0.0    0.00    0.00      11         getrandnew [27]
                0.00    0.00     135/292         randval [15]
-----------------------------------------------
                0.00    0.00      11/11          trainRBFN_final [3]
[28]     0.0    0.00    0.00      11         initCentersUsingQ [28]
                0.00    0.00     270/54247       sqr [8]
                0.00    0.00     135/292         randval [15]
                0.00    0.00      22/11156       getnew [9]
                0.00    0.00      11/11          getrandnew [27]
                0.00    0.00      11/64          readfile [19]
                0.00    0.00      11/45          writefile [20]
-----------------------------------------------
                0.00    0.00      11/11          getTheta [6]
[29]     0.0    0.00    0.00      11         inverse [29]
                0.00    0.00      22/11156       getnew [9]
                0.00    0.00      11/145         det [17]
                0.00    0.00      11/11          adjoint [24]
-----------------------------------------------
                0.00    0.00       1/4           train_final_thyroid [1]
                0.00    0.00       3/4           init_center_quantum [32]
[30]     0.0    0.00    0.00       4         unique_elements [30]
-----------------------------------------------
                0.00    0.00       3/3           init_center_quantum [32]
[31]     0.0    0.00    0.00       3         get_max [31]
-----------------------------------------------
                0.00    0.00       3/3           trainRBFN_final [3]
[32]     0.0    0.00    0.00       3         init_center_quantum [32]
                0.00    0.00       6/11156       getnew [9]
                0.00    0.00       3/3           get_max [31]
                0.00    0.00       3/4           unique_elements [30]
                0.00    0.00       3/45          writefile [20]
-----------------------------------------------
                0.00    0.00       3/3           trainRBFN_final [3]
[33]     0.0    0.00    0.00       3         init_m_quantum [33]
-----------------------------------------------
                0.00    0.00       1/1           train_final_thyroid [1]
[34]     0.0    0.00    0.00       1         in_dat [34]
                0.00    0.00       1/11156       getnew [9]
-----------------------------------------------
```

This table describes the call tree of the program, and was sorted by the total amount of time spent in each function and its children.

```
Each entry in this table consists of several lines.  The line with the
index number at the left hand margin lists the current function.
The lines above it list the functions that called this function,
and the lines below it list the functions this one called.
This line lists:
     index          A unique number given to each element of the table.
                    Index numbers are sorted numerically.
                    The index number is printed next to every function name so
                    it is easier to look up where the function is in the table.

     % time         This is the percentage of the `total' time that was spent
                    in this function and its children.  Note that due to
                    different viewpoints, functions excluded by options, etc,
                    these numbers will NOT add up to 100%.

     self   This is the total amount of time spent in this function.

     children       This is the total amount of time propagated into this
                    function by its children.

     called         This is the number of times the function was called.
                    If the function called itself recursively, the number
                    only includes non-recursive calls, and is followed by
                    a `+' and the number of recursive calls.

     name           The name of the current function.  The index number is
                    printed after it.  If the function is a member of a
                    cycle, the cycle number is printed between the
                    function's name and the index number.


For the function's parents, the fields have the following meanings:

     self   This is the amount of time that was propagated directly
                    from the function into this parent.

     children       This is the amount of time that was propagated from
                    the function's children into this parent.

     called         This is the number of times this parent called the
                    function `/' the total number of times the function
                    was called.  Recursive calls to the function are not
                    included in the number after the `/'.

     name           This is the name of the parent.  The parent's index
                    number is printed after it.  If the parent is a
                    member of a cycle, the cycle number is printed between
                    the name and the index number.

If the parents of the function cannot be determined, the word
`<spontaneous>' is printed in the `name' field, and all the other
fields are blank.

For the function's children, the fields have the following meanings:

     self   This is the amount of time that was propagated directly
                    from the child into the function.

     children       This is the amount of time that was propagated from the
                    child's children to the function.
```

```
    called      This is the number of times the function called
                this child `/' the total number of times the child
                was called.  Recursive calls by the child are not
                listed in the number after the `/'.

    name        This is the name of the child.  The child's index
                number is printed after it.  If the child is a
                member of a cycle, the cycle number is printed
                between the name and the index number.

 If there are any cycles (circles) in the call graph, there is an
 entry for the cycle-as-a-whole.  This entry shows who called the
 cycle (as parents) and the members of the cycle (as children.)
 The `+' recursive calls entry shows the number of function calls that
 were internal to the cycle, and the calls entry for each member shows,
 for that member, how many times it was called from other members of
 the cycle.

Copyright (C) 2012-2015 Free Software Foundation, Inc.

Copying and distribution of this file, with or without modification,
are permitted in any medium without royalty provided the copyright
notice and this notice are preserved.

Index by function name

   [5] GetCenters             [22] get_kmat              [16] lookup_table
  [24] adjoint                [26] get_m                 [11] multiply
  [18] concat_row_wise        [31] get_max                [4] parse
  [17] det                    [23] get_mem_mat           [15] randval
  [21] euclidean               [9] getnew                [19] readfile
  [13] evaluateRBFN           [27] getrandnew             [8] sqr
  [14] getCofactor            [34] in_dat                 [3] trainRBFN_final
  [10] getRBFActivations      [28] initCentersUsingQ      [1] train_final_thyroid
   [6] getTheta               [32] init_center_quantum   [12] transpose
   [7] get_accuracy           [33] init_m_quantum        [30] unique_elements
  [25] get_betas              [29] inverse               [20] writefile
```

Appendix 4

```
#include <stdio.h> // Used for IO
//#include "matrix.h" // Includes the matrix.h file
#include <math.h> // Used for abs(),sin(),cos()
#include <stdlib.h>// Used for rand() function

#define pi 3.14159265358979323846; // Value of PI
float sq2=1.41; // value of square root of 2
float eps=1e-8; // epsilon value used for comparison of two double numbers

//
//float multiply(matrix a,rs1, cs1, matrix b, rs2, cs2){// Returns the product of two matrices
//    int i, j, k;
//    float res[rs1][cs2];
//    //matrix res=getnew(a.rs,b.cs);
//    for (i = 0; i < rs1; i++)
//    {
//        for (j = 0; j < cs2; j++)
//        {
//            res[i][j] = 0; // initializing element to 0.0
//            for (k = 0; k<rs2; k++)
//                res[i][j] += a[i][k]*b[k][j]; // adding values to element( refer to how matrix multiplication
is done)
//        }
//    }
//    return res;
//}
//
//float sqr(float x){ // Functions returns square of a number
//        return x*x;
//} // 125,151,185

/*float getRBFActivations(float centers,float betas,float input){ // Function to calculate the
RBFActivations
        float diffs [3][4];
        //matrix diffs=getnew(centers.rs,centers.cs); // matrix stores the difference of elements
between center matrix and input matrix
        float sqrdDists[4]; //diffs.rs+1 = 6
        int i,j;
        getRBFActivations_label3:for(i=0;i<3;i++){
                for(j=0;j<4;j++) diffs[i][j]=centers[i][j]-input[0][j]; // note that input matrix is a row
matrix
        }
        sqrdDists[0]=0;
        getRBFActivations_label4:for(i=0;i<3;i++){
                sqrdDists[i]=0;
                for(j=0;j<4;j++) sqrdDists[i]+=diffs[i][j]*diffs[i][j]; // finding sum of squares of
elements of diff matrix
        }
        float z[3][1];
        //matrix z=getnew(betas.rs,betas.cs);
        getRBFActivations_label5:for(i=0;i<3;i++) z[i][0]=exp(-1.0*betas[i][0]*sqrdDists[i]);
        return z;
```

```
}*/

//matrix evaluateRBFN(matrix Centers,matrix betas,matrix Theta,matrix input){ // Function to
evaluate RBFN
//	matrix phis=getRBFActivations(Centers, betas, input);
//	int i,j;
//	for(j=1;j<=phis.cs;j++){ // Function of this loop is to add a row of '1's  before the first row of
the phis matrix
//		for(i=phis.rs+1;i>=2;i--){ // shifting all elements of the phis matrix down by 1 unit
//			phis.mat[i][j]=phis.mat[i-1][j];
//		}
//		phis.mat[1][j]=1; // setting the first row of the phis matrix to be 1
//	}
//	++phis.rs; // increasing row size of phis matrix because we added a row of 1s at the top
//	matrix z=multiply(transpose(Theta),phis);
//	return z;
//}

void evaluateRBFN(float Centers[3][4], float Betas[3][1], float Theta[4][3], float input[1][4], float
scores[3][1]){ // Function to evaluate RBFN
	//check here
	float phis[4][1];
	//float phis[3][1]=getRBFActivations(Centers, betas, input);
	float z[3][1];
	//till here
	float Theta_transpose[3][4];
	//int i,j;

	//float getRBFActivations(float centers,float betas,float input){ // Function to calculate the
RBFActivations
		float diffs [3][4];
		//matrix diffs=getnew(centers.rs,centers.cs); // matrix stores the difference of
elements between center matrix and input matrix
		float sqrdDists[4]; //diffs.rs+1 = 6
		//int i,j;
		for(int i=0;i<3;i++){
			for(int j=0;j<4;j++) diffs[i][j]=(Centers[i][j])-(input[0][j]); // note that input
matrix is a row matrix
		}
		sqrdDists[0]=0;
		for(int i=0;i<3;i++){
			sqrdDists[i]=0;
			for(int j=0;j<4;j++) sqrdDists[i]+=diffs[i][j]*diffs[i][j]; // finding sum of
squares of elements of diff matrix
		}
		//float z[3][1];
		//matrix z=getnew(betas.rs,betas.cs);
		for(int i=0;i<3;i++) phis[i][0]=exp(-1.0*(Betas[i][0])*sqrdDists[i]);

	evaluateRBFN_label6:for(int j=0;j<1;j++){ // Function of this loop is to add a row of '1's
before the first row of the phis matrix
		for(int i=3;i>0;i--){ // shifting all elements of the phis matrix down by 1 unit
			phis[i][j]=phis[i-1][j];
		}
		phis[0][j]=1; // setting the first row of the phis matrix to be 1
	}
	//++phis.rs; // increasing row size of phis matrix because we added a row of 1s at the top
	//Theta_transpose = transpose(Theta,3,3);
	//matrix transpose(float temp, char cs, char rs){// Returns the transpose of the matrix
	//float ans[rs][cs];
		for(int i=0;i<3;i++){
			for(int j=0;j<4;j++){
				Theta_transpose[i][j]=Theta[j][i]; // (i,j) of transpose matrix
corresponds to (j,i) of original matrix
			}
		}
	  //float multiply(matrix a,rs1, cs1, matrix b, rs2, cs2){// Returns the product of two matrices
		  //int i, j, k;
		  //float res[rs1][cs2];
		  //matrix res=getnew(a.rs,b.cs);
		  for (int i = 0; i < 3; i++)
		  {
			  for (int j = 0; j < 1; j++)
			  {
				  z[i][j] = 0; // initializing element to 0.0
				  for (int k = 0; k<4; k++)
					  z[i][j] += Theta_transpose[i][k]*phis[k][j]; // adding values
to element( refer to how matrix multiplication is done)
					scores[i][j]=z[i][j];
			  }
		  }
		  int j;
	//*scores = z;
}

void get_accuracy(float X[150][4],float y[150][1],float Centers[3][4],float Theta[4][3], float Betas[3]
[1], float *accuracy,int *numRight,int *total){ // Function to calculate Accuracy, Number of right
predictions,total predictions and indices of wrong predictions
	*numRight=0;
	int i,j;
	int numRight_1=0;
	char wrong [150][1];
	float scores[3][1];
	int l=-1;
	get_accuracy_label7:for(i=0;i<150;i++){
		float irow[1][4];
				for(j=0;j<4;j++){
					irow[0][j]=X[i][j]; }
		//printf("Iteration=%d\n",i);
		evaluateRBFN(Centers,Betas,Theta,irow,scores); //parse(X,i,i,1,X.cs) is the
rectangle with left upper corner (i,1) and lower right corner (i,X.cs)
		/*for(j=0;j<3;j++){
			printf("%f\n",scores[j][0]);
		}*/
		int idx;
		float maxscore=-1e5; // initializing maxscore to a large negative value
		for(j=0;j<3;j++){
			if(scores[j][0]>maxscore){
				maxscore=scores[j][0]; //if element is greater than maxscore, set
maxscore = element
				idx=j; // if element is greater than maxscore, we record its index
			}
```

```
                }
                //printf("maxscore=%f , idx=%d\n, y[i][0]=%f",maxscore,idx,y[i][0]);
                ++idx;
                if(abs(idx-y[i][0])<eps){
                        numRight_1+=1; // if idx is equal to y.mat[i][1] then increment number of
right predictions
                }

        }
        *numRight=numRight_1;
        *total=150; // setting total number of examples equal to total number of rows in X matrix
        (*accuracy)=(100.0*(float)(numRight_1))/ 150; // calculating accuracy of classifier
        printf("total=%d ,numRight=%d \n",*total,numRight_1);

}
```