RNA: REUSABLE NEURON ARCHITECTURE FOR ON-CHIP ELECTROCARDIOGRAM CLASSIFICATION AND MACHINE LEARNING

by

Yuwen Sun

Bachelor in Engineering, Zhejiang University, 2006

Submitted to the Graduate Faculty of

Swanson School of Engineering in partial fulfillment

of the requirements for the degree of

Master of Science in Electrical Engineering

University of Pittsburgh

2010

UNIVERSITY OF PITTSBURGH

SWANSON SCHOOL OF ENGINEERING

This thesis was presented

by

Yuwen Sun

It was defended on

April 5, 2010

and approved by

Allen Cheng, Assistant Professor, Department of Electrical and Computer Engineering

Steven Levitan, John A. Jurenko Professor, Department of Electrical and Computer Engineering

Mingui Sun, Professor, Department of Neurological Surgery

Thesis Advisor: Allen Cheng, Assistant Professor, Department of Electrical and Computer Engineering

RNA: REUSABLE NEURON ARCHITECTURE FOR ON-CHIP ELECTROCARDIOGRAM CLASSIFICATION AND MACHINE LEARNING

Yuwen Sun, M.S

University of Pittsburgh, 2010

Artificial neural networks (ANN) offer tremendous promise in classifying electrocardiogram (ECG) for detection and diagnosis of cardiovascular diseases. In this thesis, we propose a reusable neuron architecture (RNA) to enable an efficient and cost-effective ANN-based ECG processing by multiplexing the same physical neurons for both feed-forward and back-propagation stages. RNA further conserves the area and resources of the chip and reduces power dissipation by coalescing different layers of the neural network into a single layer. Moreover, the microarchitecture of each RNA neuron has been optimized to maximize the degree of hardware reusability by fusing multiple two-input multipliers and a multi-input adder into one two-input multiplier and one two-input adder. With RNA, we demonstrated a hardware implementation of a three-layer 51-30-12 artificial neural network using only thirty physical RNA neurons.

A quantitative design space exploration in area, power dissipation, and speed between the proposed RNA and three other implementations representative of different reusable hardware strategies is presented and discussed. An RNA ASIC was implemented using 45nm CMOS technology and verified on a Xilinx Virtex-5 FPGA board. Compared with an equivalent software implementation in C executed on a mainstream embedded microprocessor, the RNA ASIC improves both the training speed and the energy efficiency by three orders of magnitude, respectively. The real-time and functional correctness of RNA was verified using real ECG signals from the MIT-BIH arrhythmia database.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

PREFACE

I first thank my advisor, Dr. Allen Cheng. Without his relentless support, guidance, patience, and advice, I could never have done this thesis. Even more importantly, his enthusiasm in research and optimism deeply impressed and encouraged me.

I want to thank Professor Steve Levitan and Professor Mingui Sun for joining my committee and giving me precious advice on my thesis. I also thank my fellow researchers, my friends, Shimeng Huang, Zhanpeng Jin, Joseph Oresko, Jun Cheng and Heather Duschl helped implement “HeartToGo” Smartphone basic ECG diagnosis platform, Liang Liao helped implement the neuron design, Yaojun Zhang and Yanglong Lin helped to extract ECG data for RNA testing.

1.0 INTRODUCTION

1.1 DESIGN MOTIVATION

Cardiovascular diseases (CVDs), such as coronary disease, heart attack and stroke, affect the heart and blood vessels [1] and are responsible for more mortality worldwide than any other disease, including cancers. According to the World Health Organization (WHO) [2], approximately 17.5 million people died from CVDs in 2005 alone, representing 30% of all deaths in the world. CVDs are projected to continue being the leading cause of global mortality - by 2015 more than 20 million people will die from CVDs each year. Also, regarding to a report from The Wall Street Journal recently [3], the men at 40, have the risk of cardiac death 1 in 8 over the rest of their lives, while for the women, also have 1 in 24 chance of suffering sudden cardiac death, which is going to be the leading disease caused death in United States. Therefore, how to prevent the sudden death that results from the CVDs becomes a hot topic in research.

To date, physicians diagnose CVDs by studying the morphology of one's heartbeats using the electrocardiogram (ECG), which is a recording of the electrical activity of the heart via skin electrodes. A typical ECG heartbeat consists of a P wave, a QRS complex, and a T wave. The duration of a normal QRS complex is 120ms on average [4]. An abnormal CVD condition can change the period of one's ECG signal and make the QRS complex wider, narrower, or

distort it into various shapes depending on the particular disease; therefore, the duration, amplitude, and morphology of the QRS complex are useful features in diagnosing CVDs.

However, it is difficult to diagnose many arrhythmias with a standard resting ECG because it can only provide a snapshot in time of the patient's cardiovascular activity. An intermittent arrhythmia can go unnoticed, and physicians must rely on self-monitoring and symptoms reported by patients to support their final diagnosis. In some cases, ambulatory recording of ECG data, collected over extended periods of time, may be taken in an attempt to acquire data during an occurrence of an intermittent arrhythmia. In other words, a solution that can perform diagnose in real-time with portable ability is needed to meet the requirements. However, the existing portable ECG solutions, such as Holter monitors [5] are limited in that these devices only provide recording and monitoring capabilities and no real-time classification of ECGs because the classification is performed off-line.

To develop the real-time CVD detection, artificial neural network (ANN) is chosen among the algorithms and methods such as fuzzy logic [6][7], morphology features, hidden Markov model [8], because of its simpler structure and competitive high accuracy, which are introduced in related works. Artificial neural networks can automatically perform ECG classification, and also be able to be implemented on a portable device for real-time application. Many advantages of an artificial neural network solution have been demonstrated to be successful performance [9], such as (1) artificial neural network is a superior algorithm in pattern recognition problems [10][11][12]; (2) its updated weights and biases are calculated by iteratively training; (3) artificial neural network that is constructed by similar basic cell, has very simple structure for physical implementation; (4) it can easily map complex class distributions, and generalize the property of the artificial neural network produces appropriate results for the

input vectors that are not present in the training set. A great deal of papers was published during the past 20 years, which have done the research on ECG detection and classification [13][14][15][16][17].

Artificial neural network (ANN), an established biologically inspired machine learning paradigm, mimics its biological counterpart in the human brain to provide powerful learning capability in recognizing patterns in complex, non-linear signals. Prior works have demonstrated that artificial neural network can be effective in classifying abnormal ECG patterns [18]. Artificial neural network is adaptive in that the weights and biases of its interconnected neuron model can be adjusted based on the external input during its training phase. Due to individual's physiological differences, everyone's ECG signals can be different; even within the same individual, his/her ECG can vary over time. Recognizing changing patterns requires updating artificial neural network's weights and biases through re-training. Also pointed out by Dipti Itchhaporia's review paper, the absence of some kinds of sample data in training data set may causes inaccurate result [19]. Therefore, frequently updating the training database is significantly important to achieve a better result. However, most prior ECG-related artificial neural network research performed training and database offline, which would require the user to go to a central facility (e.g., a hospital) every time when the artificial neural network needs to be trained, which is very impractical in real life.

Thus, we argue that a better solution is to let the mobile ECG device have the capability to perform on-demand training on the device itself. Running artificial neural network as a software program on a mobile device such as a smartphone, is one of the potential solutions evaluated in this thesis. Nevertheless, as we will demonstrate in our implementation section, performing artificial neural network training on a smartphone can be a very time consuming task

- it is estimated to take several days to even months to complete depending on the size of the training data. In addition to the excessively prolonged training time, the extra energy and power consumption toll post other critical issues: excessively high energy consumption can drain the battery life from days to hours; the extra heat generated by excessively high power dissipation can not only make the phone uncomfortable to wear but may also cause permanent damages if peak power threshold is exceeded. To address the need for high-speed, low-power, and energy-efficient artificial neural network training and classification, we propose to implement artificial neural network as an application-specific integrated circuit (ASIC). Because artificial neural network as a mathematical algorithm is established and mature, which makes custom ASIC a feasible solution. However, the non-linear activation functions and multipliers required by both the feed-forward and back-propagation part of the artificial neural network algorithm can require significant amount of hardware resources to build, which can make the artificial neural network ASIC too big or still too power-hungry for a small mobile device.

1.2 SYSTEM OVERVIEW

In this thesis, we propose a cost-effective reusable neuron architecture (RNA) to perform electrocardiogram processing and artificial neural network machine learning, and also compare proposed RNA design with other circuit reused designs. In RNA, only one layer of neurons (the layer with the maximum number of neurons) and one unique look-up table (LUT) are required to implement on artificial neural network. RNA reduces the resource requirement by multiplexing the same layer of neurons so that it can behave like a different layer of the network at different algorithmic stage. The Global Controller is in charge of reconfiguring the network layer to

execute proper computational tasks. Each Neuron-Cell only needs one multiplier and one adder. All the computation for both feed-forward and back-propagation is achieved by adjusting the multiplexer (MUX) in front of the multiplier and adder. A single look-up table is implemented for the entire artificial neural network system to represent the transformation of the activation function. Due to the time-division strategy, the same hardware resource of RNA can be effectively multiplexed and both feed-forward classification and back-propagation training can be performed by the same Neuron-Cells. With comparing with other embedded software solution platforms, such as smartphone, several performance dimensions are measured and discussed. To the best of our knowledge, this thesis is the first one to implement the artificial neural network on ASIC aiming on application of ECG classification with machine learning.

1.3 CONTRIBUTION

The contributions of this work are in three domains: (1) We performed an extensive survey of modern architectures of hardware designs of artificial neural network, and proposed a novel architecture of artificial neural network implementation (RNA) with the resources reused, which can perform both feed-forward and back-propagation computation. (2) The proposed architecture surpasses its portable counterpart device smartphone with distinguished advantages: dramatically fast in execution time and less energy requirement. (3) By the capability of performing training on-chip, the proposed design RNA can frequently update its artificial neural network architecture’s weights and biases, and provide the possibility of using in practical life for the patients or users.

1.4 THESIS ORGANIZATION

The remainder of this thesis is organized as follows: Section 2.0 provides a background about ECG and artificial neural network (ANN). More specifically, section 2.1 focus on the introduction of electricalcardiograph (ECG) and section 2.2 introduces artificial neural network, including its structure, activation function, feed-forward algorithm and learning algorithms. In section 3.0, related works and research are discussed, including artificial neural network implementation in section 3.1 and ECG classification in section 3.2. The methodology of RNA model is illustrated in section 4.0, including model construction in section 4.1 and model verification in section 4.2. The proposed RNA design is introduced in section 5.0, including the Global Controller in section 5.1, the Neurons Group in section 5.2, the Local Controller in section 5.3, look-up table in section 5.4, the RAM Group in section 5.5, at last the Initialization-ROM and ECG-Database-ROM in section 5.6. The RNA implementation procedure, results and summary are discussed in section 6.0. As a comparison of the proposed RNA, 3 other ASIC designs are discussed in section 7.0, which are Flat design in section 7.1, Lightweight-Neuron design in section 7.2 and Layer-Reused design in section 7.3. Summary and discussion among total 4 ASIC designs are in section 7.4. The post-synthesis results and simulation are discussed in section 8.0. To compare with RNA ASIC design, an embedded software implementation and discussion are presented in section 9.0. At last, the FPGA demo implementation is demonstrated in section 10.0. Finally, the conclusion is in section 11.0 and the future works is in section 12.0.

2.0 BACKGROUND

2.1 ELECTROCARDIOGRAPHIC (ECG)

In the year of 1872, Alexander Birmick Muirhead firstly attached wires to a patient's wrist to obtain a record of the patient's heartbeat [20]. After that, Professor Augustus Waller [21] was the first one that approached the heartbeat from an electrical point-of-view systematically, and his electrocardiograph machine consisted of a Lippmann capillary electrometer fixed to a projector. The trace from the heartbeat was projected onto a photographic plate which was itself fixed to a toy train, allowing a heartbeat to be recorded in real time. In the year of 1903, Willem Einthoven [22] made a breakthrough by inventing a string galvanometer, which was much more sensitive than the devices designed prior and used in the same time. 21 years later, Einthoven [23] won the Nobel Prize in Medicine by his discovery and assigned the letters from P to T, in order to symbol various deflections and describe the electrocardiographic (ECG) features. Although the technology of testing heartbeat has been improved day after day, the same basic principles are still use today.

Lead, the electrodes that attached to patients are used to obtain the heartbeat information via skin, by transferring biological signals to electrical signals. In a typically 12-lead ECG pattern, 10 electrodes are attached to patient's body to obtain the heartbeat information [24][25], as shown in figure 1:

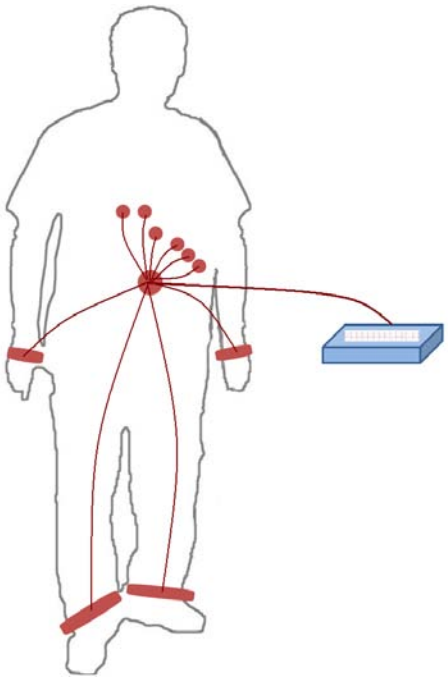


Figure 1. Body with 10 electrodes attached for ECG recording

(1) RA: On the right arm, avoiding bony prominences; (2) LA: In the same location that RA was placed, but on the left arm this time; (3) RL: On the right leg, avoiding bony prominences; (4) LL: In the same location that RL was placed, but on the left leg this time; (5) V1: In the fourth intercostal space (between ribs 4 and 5) just to the right of the sternum (breastbone); (6) V2: In the fourth intercostal space (between ribs 4 and 5) just to the left of the sternum; (7) V3: Between leads V2 and V4; (8) V4: In the fifth intercostal space (between ribs 5 and 6) in the mid-clavicular line (the imaginary line that extends down from the midpoint of the clavicle (collarbone); (9) V5: Horizontally even with V4, but in the anterior axillary line. (The anterior axillary line is the imaginary line that runs down from the point midway between the middle of the clavicle and the lateral end of the clavicle; the lateral end of the collarbone is the end closer to the arm.); (10) V6: Horizontally even with V4 and V5 in the mid-axillary line. The mid-axillary line is the imaginary line that extends down from the middle of the patient's armpit.

A typical ECG is the tracing of a normal heartbeat, which mainly consists of a P wave, a QRS complex and a T wave, as shown in figure 2.

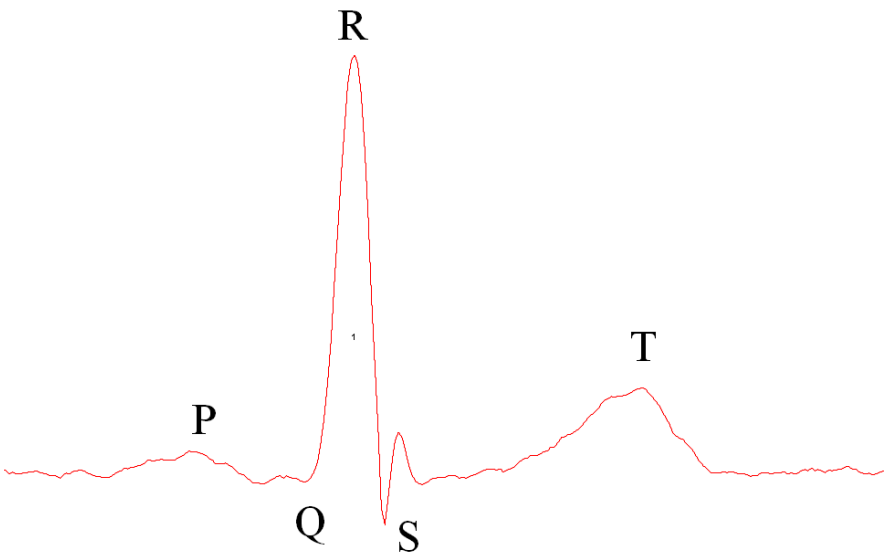


Figure 2. A typical ECG with P wave, T wave and QRS complex

The four deflections are formed by the point, P, Q, R, S and T. The P wave demonstrates that the electrical vector spreads from the right atrium to the left one, and normal duration of the P wave is around 80ms. The PR segment connects the P wave and QRS complex, which has the duration about 150-200ms. The QRS complex is a main part of structure on the ECG that shows a single heartbeat that relates to the depolarization of the right and left ventricles, and its duration is 70-110ms. Similar as PR segment, the ST segment connects the QRS complex and T wave, and has the duration of 80-120ms. The T wave represents the repolarization of ventricles, while the duration is 160ms. PR interval is the measurement from P point to R point, which usually has the duration of 120-200ms, the ST interval is the measurement from S point to T point, which has the duration of 320ms, and the QT interval is the measurement from Q point to T point, which has the duration 300-430ms.

2.2 ARITIFICIAL NEURAL NETWORK

Artificial neural network (ANN) is an emulation processing of biological neuron structure. The mathematical model was first built by a psychologist Warren S. Mcculloch and mathematician Walth H.Pitts in year 1943. It has been ignored for a very long time, but in recent years, it came back to our research focus and rapidly developed [26].

Technically, there is no precise agreed-upon definition among researchers about what exactly a neural network is. But most of them might agree that a neural network should involve a network of simple processing elements (that is called neurons). These elements should be able to show a complex global behavior which determined by the ways of connection between the processing elements and element parameters. Another acceptable definition of a neural network is an adaptive machine [27], that a neural network is a massively parallel distributed processor made up of simple processing units, which has a natural propensity for storing experiential knowledge and making it available for use, and it resembles the brain in two respects: (1) knowledge is acquired by the network from its environment through a learning process; (2) interneuron connection strengths, known as synaptic weights, are used to store the acquired knowledge. The learning algorithms are the procedure for learning process, which is used to modify artificial neural network's synaptic weights and biases to achieve a neural network that can obtain a desired classification result.

2.2.1 Artificial neural network's structure

The basic stand cell of the artificial neural network is the computation cell called neuron, which has multiple inputs and one output, as shown in figure 3.

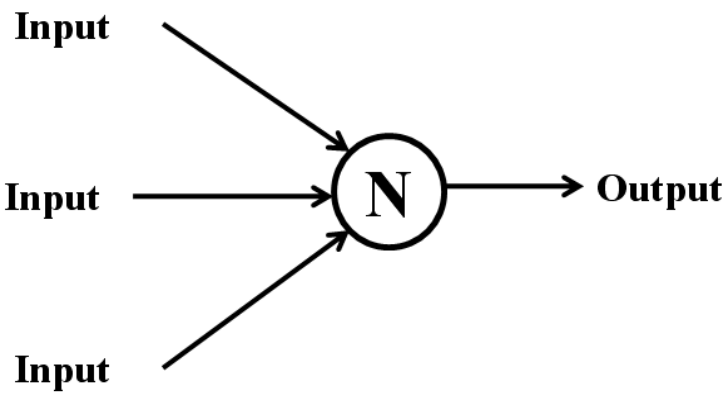


Figure 3. A neuron with 3 inputs and 1 output

Each input multiplies the different weights, and then all the products are summed together with a bias. At last, a linear or non-linear activation function is used on this summary, to get the final output of the neuron. The artificial neural network is constructed by many basic stand cells, from decades of neurons to even thousands of neurons. These neurons are divided by layers, and usually three kinds of the layers together form a completed artificial neural network, which are input-layer, hidden-layer and output-layer, as shown in figure 4.

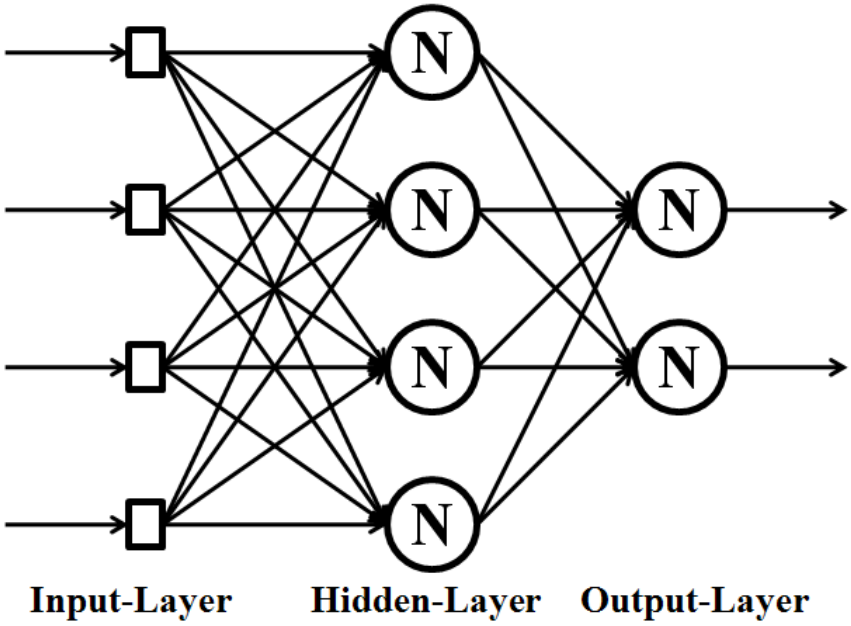


Figure 4. Artificial neural network with 3 layers

2.2.2 Activation function

The activation function is a very important part in the artificial neural network, which donates critical effects on the performance of the network. Linear activation function and non-linear activation function are the two categories of activation functions which are commonly used in artificial neural network, and non-linear activation function is used more since it can support more complicated computational models. Three basic types of the activation functions are popular widely because of their succinct structure and solid performance [28], which are threshold function, as shown in figure 5; piecewise function, as shown in figure 6; and sigmoid function, as shown in figure 7.

$$\varphi(\upsilon) = \begin{cases} 1 & \text{if } \upsilon \geq 0 \\ 0 & \text{if } \upsilon < 0 \end{cases}$$

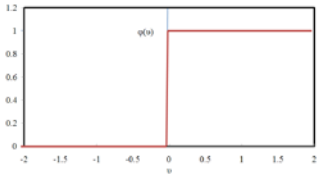


Figure 5. Threshold function: mathematics expression and waveform display

$$\varphi(\upsilon) = \begin{cases} 1, & \upsilon \geq +\frac{1}{2} \\ \upsilon + 0.5, & \pm\frac{1}{2} > \upsilon > -\frac{1}{2} \\ 0, & \upsilon \leq -\frac{1}{2} \end{cases}$$

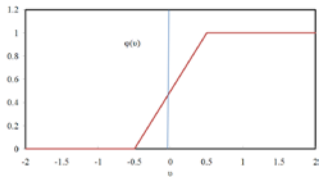


Figure 6. Piecewise function: mathematics expression and waveform display

$$\varphi(\upsilon) = \frac{1}{1 + \exp(-\alpha\upsilon)}$$

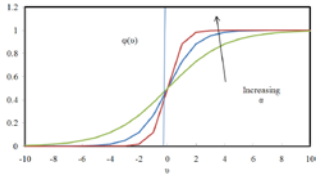


Figure 7. Sigmoid function: mathematics expression and waveform display

2.2.3 Feed-forward algorithm

The feed-forward classification is the algorithm used for calculating the output of the artificial neural network. Once the training process has finished, the feed-forward process can be performed, which is executed by feeding in the inputs from outside to the input-layer of the network. Each neuron performs the summary among the production of input and according weight, and then performs activation function. The final outputs of the neurons in this layer, become the inputs to the neurons of next layer. The process repeats layer by layer among the hidden-layers, and the final outputs of the last hidden-layer feed into output-layer as the inputs. At last, the outputs of the output-layer become the final outputs of the whole artificial neural network, as the final outputs of feed-forward classification.

2.2.4 Learning algorithm

The artificial neural network can successfully solve many kinds of difficult problems in varied areas by very well training ahead in some supervised manners, and error back-propagation algorithm is one of the popular supervised training algorithms, which is based on error correction learning rule. The back-propagation algorithm contains two passes through all the layers, which are a forward pass and a backward pass. The forward pass is similar to the feed-forward algorithm, and the backward pass is used for adjusting the weights and biases for each neuron in the network. In a multi hidden-layer artificial neural network, the signal flow is from output-layer back to input-layer, through all the hidden-layers, so that the error information of the outputs can be propagated to each layer to calculate the change of the weights and biases for each neuron in each layer.

3.0 RELATED WORKS

3.1 ARTIFICIAL NEURAL NETWORK IMPLEMENTATION

S. Himavathi et al. [29], used reconfigurable FPGA (Xilinx FPGA XCV400hq240) to implement a multilayer feed-forward artificial neural network. In this design, the neurons were reused, and a physical implemented layer behaved as different layers being configured by a controller, which reduced the logic resource usage of the chip. However, in this design only feed-forward part of the artificial neural network was implemented on the FPGA, which means that this design requires a general PC or some other platform to perform the training pass and get the weights and biases value, so that it can be implemented on FPGA for feed-forward computation. Once it's implemented, the weights cannot be changed any more, in other words, the weights cannot be adjusted as needed on-the-fly.

D. Ferrer et al. [30], implemented a multilayer perception neural network on FPGA (ARC-PCI board) with the parameterized number of both layers and neurons in each of them, to perform the feed-forward computation. The reuse technique reduced the hardware resource usage of the FPGA, and more over the parameterization made the design more flexible to diverse applications, but the same as the previous work, this implementation only had feed-forward part implemented without the training pass.

P. Domingos et al. [31], implemented the artificial neural networks system with both feed-forward and back-propagation stages on the reconfigurable hardware, and compared the performance difference between the hardware implementation solution and software implementation solution. In this work, a significant advantage in speed was shown on hardware implementation. In this design an almost complete system was implemented in hardware, but this work only focused on the implementation and comparison between the hardware and the software, no hardware resource reuse technique implemented on this design, resulting in ineffectively using the hardware resource.

J. Eldredge et al. [32], presented the RRANN architecture to perform both feed-forward and back-propagation computation algorithms on the FPGA (Xilinx XC3090). This work implemented both feed-forward and back-propagation algorithms on the FPGA, and this design also adopted the hardware resource reused technique, that the neurons were reused as the ones in different layers, however in this design, the feed-forward algorithm and back-propagation algorithm were implemented by two pieces of hardware circuit, which required reconfiguring the FPGA or a larger hardware resource to implement both two pieces of the circuits.

Different from the previous works, in this thesis we propose a novel design that takes a step further comparing to the works mentioned above. In this design, both feed-forward and back-propagation algorithms are implemented on the hardware, so that the weights and biases of the artificial neural network can be retrained itself, so periodically training to update the weights and biases is feasible and practical with this implementation. Not only implementing a complete structure of the artificial neural network, but also a further step to reuse the hardware resource is obtained in this design. Additionally, the feed-forward structure is reused for back-propagation computation.

On the other hand, many research have been done for implementing artificial neural networks on analog VLSI. M. Giulinoi et al. [33], presented a design of a configurable analog VLSI chip to perform neural network, which was implemented with 32 integrate-and-fire (IF) neurons and 2048 reconfigurable, Hebbian, plastic bistable, spike-driven stochastic synapses. The neurons in this design were spike-frequency adaptable, while the synapses were endowed with a self-regulating mechanism that was able to stop unnecessary synaptic changes.

Robert W. Newcomb [34], implemented an analog VLSI ANN circuit, which was built with Differential Voltage Controlled Current Source (DVCCS) and capacitors. The DVCCS design was used for implementing synaptic weights and activation functions, while the capacitors were used for dynamics, which took a voltage difference as input, and gave an output current as a function of that difference.

Since the digital VLSI circuits are robust to noise and disturbances, and have advantage in accuracy, in this thesis we implement an artificial neural network structure on a digital VLSI, with a technique that reusing neurons to reduce the hardware resource usage. By the proposed design, a full phase with both feed-forward and back-propagation computations are performed on RNA ASIC design.

3.2 ECG CLASSIFICATION

P.M. Rautaharju et al. [35], presented a system called NOVACODE serial ECG classification system, for clinical trials and epidemiologic studies. This system was used for visual and computer coding of serial ECGs, which was designed to alleviate some of the instability

problems by quantifying changes in critical waveform patterns on a continuous scale. A serial comparison program was used to detect major changes and adjust instability in standard MC classification at decision boundaries.

P. Bozzola et al. [36], built a system called Hybrid Neuron-Fuzzy System for ECG Classification, which was based on a hybrid Neuro-Fuzzy model. The classification power of the connectionist paradigm was coupled with the ability of the fuzzy set formalism to treat in a quantitative way natural language, enabling to build up a system capable of both accuracy and to give meaningful explanations of the diagnoses, in form of symbolic if-then rules. An accuracy range of 60% to 95% for varied kinds of abnormal heartbeats was claimed in this paper.

W.T. Cheng et al. [37], presented a project to classify the electrocardiogram by using hidden Markov models. The QRS was firstly detected by one-pole filter, and then the hidden Markov models (HMMs) were investigated to automatically classify the ECG signal. ECG data from the American Heart Association (AHA) was used for testing in this project. An average range of classification accuracy was from 56.38% to 93% for varied kinds of heartbeats.

Yu Hen et al. [38], presented several artificial neural network structures for the applications of ECG signal detection and classification. The neural network was used to detect the ECG signal firstly, and was also used to classify the ECG signal into normal and abnormal. Furthermore, an artificial neural network structure was proposed to classify the different kinds of heart arrhythmias, which had 51 neurons in the input-layer, 30 neurons in the hidden-layer, and 12 neurons in the output-layer. An accuracy range of 62% to 100% for varied kinds of abnormal heartbeats was claimed by this 51-30-12 structure. The artificial neural network with 51-30-12 structure is chosen to perform ECG training and classification in this thesis, since it has succinct structure and competitive high classification accuracy.

4.0 METHODOLOGY: RNA MODEL

4.1 ARTIFICIAL NEURAL NETWORK MODEL

In this thesis, an artificial neural network model with three layers is built, which includes 51 neurons in input-layer, 30 neurons in hidden-layer and 12 neurons in output-layer. The succinct structure and solid performance of applying this artificial neural network on ECG classification application, has been discussed and proved by the research of Yu Hen et al. [38]. The feed-forward pass for classification and back-propagation pass for training are described below.

The feed-forward pass, from input-layer to the hidden-layer is described as:

$$\text{Sum\_H (j)} = \text{Bias(j)} + \sum_{n=1}^{N} \text{In((n))} \times \text{Wt\_IH((n))(j)}$$

Eq. 4.1

$$\text{H\_out(j)} = \frac{1}{1 + e^{-\text{Sum\_H(j)}}}$$

Eq. 4.2

In(n) is the input of neuron n in the input-layer, and n is from 1 to 51, since we have 51 neurons in input-layer. Wt_IH(n)(j) is the weight from neuron n in the input-layer to the neuron j in the hidden-layer. Bias(j) is the bias of neuron j in hidden-layer, and Sum_H(j) is the summation of the products of all the inputs multiplying their corresponding weights to neuron j

and plus the bias for neuron j. Feeding Sum_H(j) into activation function, H_out(j) is obtained as the final output of neuron j in hidden-layer. Sigmoid with parameter $\alpha = 1$, is chosen for the non-linear activation function in this design.

Then, the computation from hidden-layer to output-layer in the feed-forward pass is similar to the computation from input-layer to hidden-layer. All the outputs of neurons in hidden-layer become the inputs of the neurons in output-layer.

$$\text{Sum\_O (j)} = \text{Bias(j)} + \sum_{n=1}^{N} \text{H\_out((n))} \times \text{Wt\_HO(n)(j)}$$

Eq. 4.3

$$\text{O\_out(j)} = \frac{1}{1 + e^{-\text{Sum\_O(j)}}}$$

Eq. 4.4

H_out(n) is the final output of neuron n in hidden-layer, which is the input of the neurons in output-layer at the same time. So the range of n is from 1 to 30, the number of the neurons in hidden-layer. Wt_HO(n)(j) is the weight from neuron n in hidden-layer to the neuron j in output-layer. Bias(j) is the bias of neuron j in the output-layer, and Sum_O(j) is the summation of the products of all the inputs multiplying their corresponding weights and plus the bias. Feeding Sum_O(j) into the same sigmoid activation function, O_out(j) is obtained as the final output of neuron j in the output-layer.

For the computation of back-propagation training pass, the difference between the outputs of neurons in output-layer and the target final outputs is compared, and used for calculating the delta-output value of the neuron j in output-layer:

$$\text{Del\_O (j)} = (\text{Tar(j)} - \text{O\_out(j)}) \times \text{O\_out(j)} \times (1 - \text{O\_out(j)})$$

Eq. 4.5

Tar(j) is the target final output of the neuron j in output-layer, O_out(j) is the final output of the neuron j in output-layer and Del_O(j) is delta-output value of the neuron j in output-layer.

Then the delta-output of neuron j in hidden-layer is calculated as:

$$\text{Sim\_BP(j)} = \sum_{n=1}^{N} \text{Del\_O(n)} \times \text{Wt\_HO(j)(n)}$$

Eq. 4.6

$$\text{Del\_H(j)} = \text{Sum\_BP(j)} \times \text{H\_Out(j)} \times (1 - \text{H\_Out(j)})$$

Eq. 4.7

Del_O(j) is delta-output value of the neuron j in output-layer and Wt_HO(j)(n) is the weight from neuron j in hidden-layer to the neuron n in output-layer. So the intermediate Sim_BP(j) is the summation of the delta-outputs multiplying their corresponding weights from the same neural j in hidden-layer to the 12 neurons in output-layer. By the intermediate Sim_BP(j) of each neuron j in hidden-layer and its final output (H_out(j)), the delta-output value of each neuron j in hidden-layer is calculated, as described by Del_H(j).

Based on the delta-output value of each neuron in the hidden-layer, the delta-weight for neuron i in the input-layer to neuron j in the hidden-layer is calculated:

$$\text{DelWt\_IH(i)(j)} = \alpha \times \text{Del\_H(j)} \times \text{In(i)} + \beta \times \text{DelWt\_IH(i)(j)}$$

Eq. 4.8

In(n) is the input of neuron n in the input-layer, and Del_H(j) is delta-output value of the neuron j in hidden-layer. DelWt_IH(i)(j) is the delta-weight for neuron i in input-layer to neuron j in hidden-layer, which is updated by adding the previous value multiplying a parameter β. The parameters α and β are used to control the learning rate in the training process, and are assigned α = 0.5 and β = 0.9 in this design.

DelWt_HO(m)(n), the delta-weight for neuron m in hidden-layer to neuron n in output-layer, is calculated as:

$$\text{DelWt\_HO(m)(n)} = \alpha \times \text{Del\_O(n)} \times \text{H\_out(m)} + \beta \times \text{DelWt\_HO(m)(n)}$$

Eq. 4.9

H_out(n) is the final output of neuron n in the hidden-layer, and Del_O(j) is delta-output value of the neuron j in output-layer. DelWt_HO(m)(n) is the delta-weight for neuron m in hidden-layer to neuron n in output-layer, which is updated by adding the previous value multiplying a parameter β. The parameters α and β are used to control the learning rate in the training process, and are assigned α = 0.5 and β = 0.9 in this design.

The bias for the neuron i in hidden-layer and neuron j in output-layer are also updated as:

$$\text{Bias(i)} = \alpha \times \text{Del\_H(i)} + \beta \times \text{Bias(i)}$$

Eq. 4.10

$$\text{Bias(j)} = \alpha \times \text{Del\_O(j)} + \beta \times \text{Bias(j)}$$

Eq. 4.11

Del_O(j) is delta-output value of the neuron j in output-layer, and Del_H(j) is delta-output value of the neuron j in hidden-layer. All of the biases of the neurons, those in hidden-layer and output-layer, are calculated by adding the previous bias multiplying parameter β. The same parameters α and β are set to control the learning rate and speed.

At last, the weight from neuron i in input-layer to neuron j in hidden-layer is calculated:

$$\mathrm{Wt\_IH(i)(j)} = \mathrm{Wt\_IH(i)(j)} + \mathrm{DelWt\_IH(i)(j)}$$

Eq. 4.12

Wt_IH(i)(j), the weight from neuron i in input-layer to neuron j in hidden-layer, is updated by adding corresponding delta-weight, DelWt_IH(i)(j).

The weight from neuron m in hidden-layer to neuron n in output-layer is also calculated:

$$\mathrm{Wt\_HO(m)(n)} = \mathrm{Wt\_HO(m)(n)} + \mathrm{DelWt\_HO(m)(n)}$$

Eq. 4.13

Wt_HO(m)(n), the weight from neuron m in hidden-layer to the neuron n in output-layer, are updated by adding the corresponding delta-weight, DelWt_HO(m)(n).

The training process is stop when either condition below is satisfied: 1) The epoch number of training exceeds the maximum number that is set in this design; 2) The total error energy is less than the target average squared error energy. The average squared error energy is obtained by summing E(n) over all n and then normalizing with respect to the set size N:

$$\mathrm{E_{av}(n)} = \frac{1}{N}\sum_{n=1}^{N} \mathrm{E(n)}$$

Eq. 4.14

E(n) is the instantaneous value of total error energy, obtained by summing over $e_j^2(n)$ all neurons in output-layer:

$$E(n) = \frac{1}{2} \sum_{j \in C} e_j^2(n)$$

Eq. 4.15

$e_j(n)$ is the error of the neuron j in output-layer at iteration n. More specifically, $e_j(n)$ is obtained by taking the difference between the target value $d_j(n)$ and $y_j(n)$, the output value of the neuron in output-layer:

$$e_j(n) = d_j(n) - y_j(n)$$

Eq. 4.16

This artificial neural network model for feed-forward and back-propagation passes is originated from the mathematics model of Haykin's Neural Network [28] with modification to fit the digital implementation requirements.

The completed training process is, 1) perform a feed-forward pass and a back-propagation pass for each data one by one in the database, and accumulate error energy at the same time; 2) check the accumulated error energy, and if it's less than the target error energy, break the training process since stop condition 2 is met; 3) clear the accumulated error energy to 0; 4) perform a second feed-forward pass and a back-propagation pass for each same data one by one in the database, and accumulated error energy; 5) the training process is the iteration from step 1 to step 3, and the iteration number is the training "epoch"; 6) until epoch larger than the threshold set in this design, the training stops, as the stop condition 1 is met. The completed training process is also written as pseudo-code, as shown below with the related equation number labeled behind:

```
TrainANN (trainingdata_path,diagnosis_path,train_length) {
        Assign value to input array;
        Assign value to target array;

        Initialize weights from input-layer to hidden-layer;
        Initialize weights from hidden-layer to output-layer;

        For (epoch = 0 ; epoch < training times ; epoch++) {

            Randomize order of training beats;
            For ( np = 1 ; np <= NumBeats ; np++ ) {
                Feed-forward input-layer to hidden-layer; (Eq. 4.1, 4.2)
                Feed-forward hidden-layer to output-layer; (Eq. 4.3, 4.4)

                Error energy calculate; (Eq. 4.14, 4.15, 4.16)

                Back-propagate output-layer to hidden-layer; (Eq. 4.5)
                Weights and Bias updated; (Eq. 4.9, 4.11, 4.13)
                Back-propagate hidden-layer to input-layer; (Eq. 4.6, 4.7)
                Weights and Bias updated; (Eq. 4.8, 4.10, 4.12) }

            If (error < Threshold) {
                Break; }}}
```

4.2 MODEL VERIFICATION

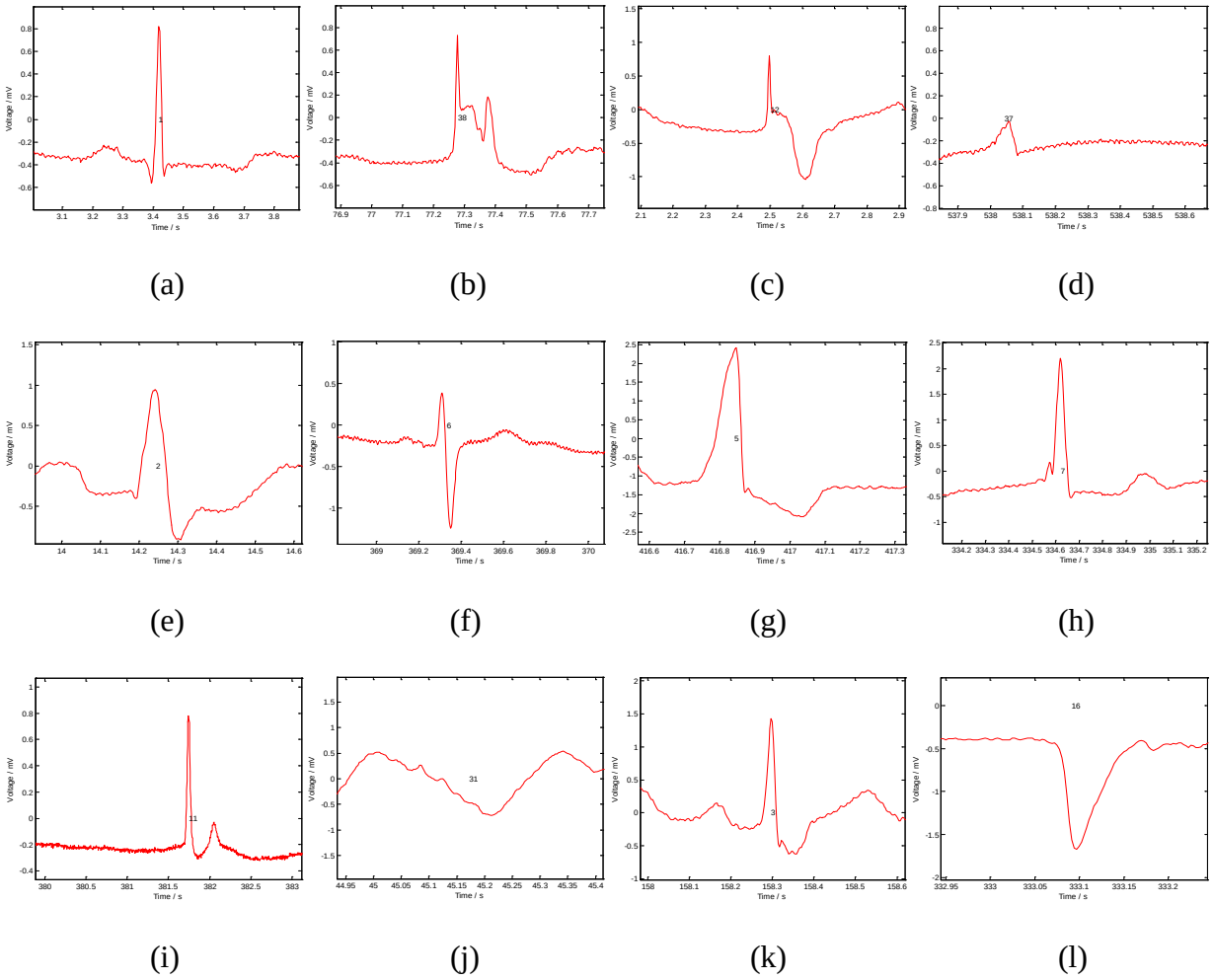


Figure 8. Segment heartbeat from MIT-BIH database: (a) Normal Heartbeat; (b) PFUS Heartbeat; (c) PACE Heartbeat; (d) NAPC Heartbeat; (e) LBBB Heartbeat; (f) FUSION Heartbeat; (g) PVC Heartbeat; (h) NPC Heartbeat; (i) NESC Heartbeat; (j) FLWAV Heartbeat; (k) RBBB Heartbeat; (l) ARFCT Heartbeat.

The artificial neural network with 51-30-12 structure was firstly built on general PC in C language. Total 413 heartbeats were selected from MIT-BIH database [39], including 12 kinds of heartbeats, which are normal heartbeats from patient No.100, PFUS heartbeats from patient No.102, PACE heartbeats from patient No.104, NAPC heartbeats from patient No.108, LBBB

heartbeats from patient No.109, FUSION heartbeats from patient No.114, PVC heartbeats from patient No.116, NPC heartbeats from patient No.124, NESC heartbeats from patient No.201, FLWAV heartbeats from patient No.207, RBBB heartbeats from patient No.212 and ARFCT heartbeats from patient No.105, as shown in figure 8.a-l. For each kind, the heartbeats were randomly divided into 2 groups as the training set and testing set. Total 206 heartbeats with all 12 kinds of heartbeats formed a database for training our artificial neural network model, as shown in table 1.

Table 1. The 12 kinds of heartbeats in training set and testing set

No.	Heartbeats Name	Number of Heartbeats Training	Number of Heartbeats Testing	Number of Heartbeats Total
1	Normal	90	110	200
2	PFUS	8	7	15
3	PACE	10	13	23
4	NAPC	7	3	10
5	LBBB	11	9	20
6	FUSION	11	7	18
7	PVC	11	8	19
8	NPC	16	12	28
9	NESC	6	7	13
10	FLWAV	14	13	27
11	RBBB	16	14	30
12	ARFCT	6	4	10
SUM		206	207	413

To verify the efficiency and performance of this 51-30-12 artificial neural network model, we changed the number of the neurons in hidden-layer from 30 to 40 and 20 respectively, and also we changed the parameter α in the sigmoid activation function from 1 to 1.25 and 0.75 respectively, to obtain the change in training process and testing process. In the training process, the error energy converges for all 5 models as epoch numbers increased, obtained and shown in figure 9. After training, the testing set with total 207 heartbeats was classified by trained artificial

neural network, and the average classification accuracy of all 5 artificial neural network models were also shown in figure 9.

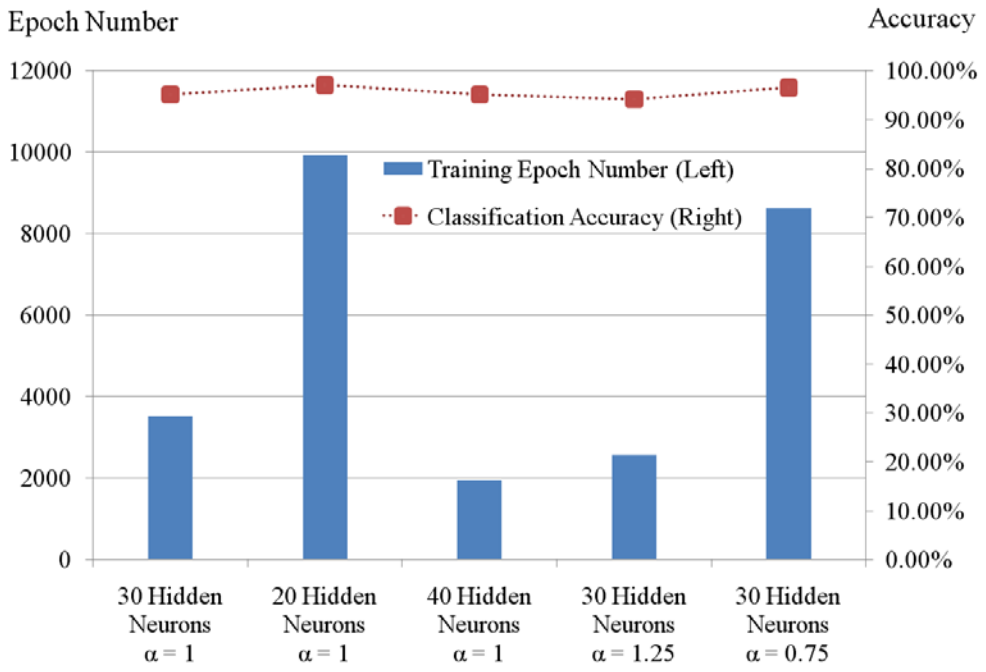


Figure 9. Training epoch number and testing accuracy of 5 models

From figure 9, as the number of hidden neurons decreased, the number of training epoch was dramatically increased (2.5x), nevertheless as the number of hidden neurons increased, the number of training epoch was decreased. For the sigmoid activation function parameter, as α increased from 1 to 1.25, the number of training epoch decreased, while as α decreased from 1 to 0.75, the number of training epoch significantly increased (more than 2x). Since the neurons are the computation cells, more neurons can handle more complicated computation, and reduce the training epoch number. However, parameter α controls the slope of the sigmoid function, and big α results in a more aggressive but few epochs training process, while small α achieves a fine but more epochs training process. However, the testing accuracies of all these 5 models were in the range from 94% to 97%, which presents high accuracy.

5.0 PROPOSED DESIGN: RNA

Based on the previous technology, this work proposes an evolved, highly efficient architecture for artificial neural network. Only one array of neurons (the layer with the maximum number of neurons) with a unique look-up table is implemented. This novel architecture executes the single layer repeatedly for each layer, and behaves like different layers of the network. The Global Controller is in charge of handling the proper computation of the layer. Each Neuron-Cell only needs one multiplier and one adder. All the computation for both feed-forward and back-propagation is achieved by adjusting the MUX ahead of the multiplier and adder. A single look-up table is physically implemented for the entire artificial neural network system to represent the transformation of the activation function. Due to the time-division strategy, the hardware resource is efficiently used. Both feed-forward classification and back-propagation training can be performed by the same Neuron-Cells.

Since the hardware structure we designed is based on reusing Neuron-Cells, only 30 Neuron-Cells are actually needed to implement on hardware, greatly reducing the logic resource consumption. Reducing the number of Neuron-Cells makes it more efficient in using the hardware resource, because each Neuron-Cell contains one multiplier, which occupies many more logic resources. The RNA system we proposed includes 1 Global Controller, 1 Local Controller, 30 Neuron-Cells as 1 neuron group, 30 Neuron-Registers as 1 RAM group, and 1 look-up table (MUX&LUT), which are all shown in figure 10. Besides, the completed system

also has an Initialization-ROM for initializing and an ECG-Database-ROM for storing the ECG data, which are not shown in figure 10.

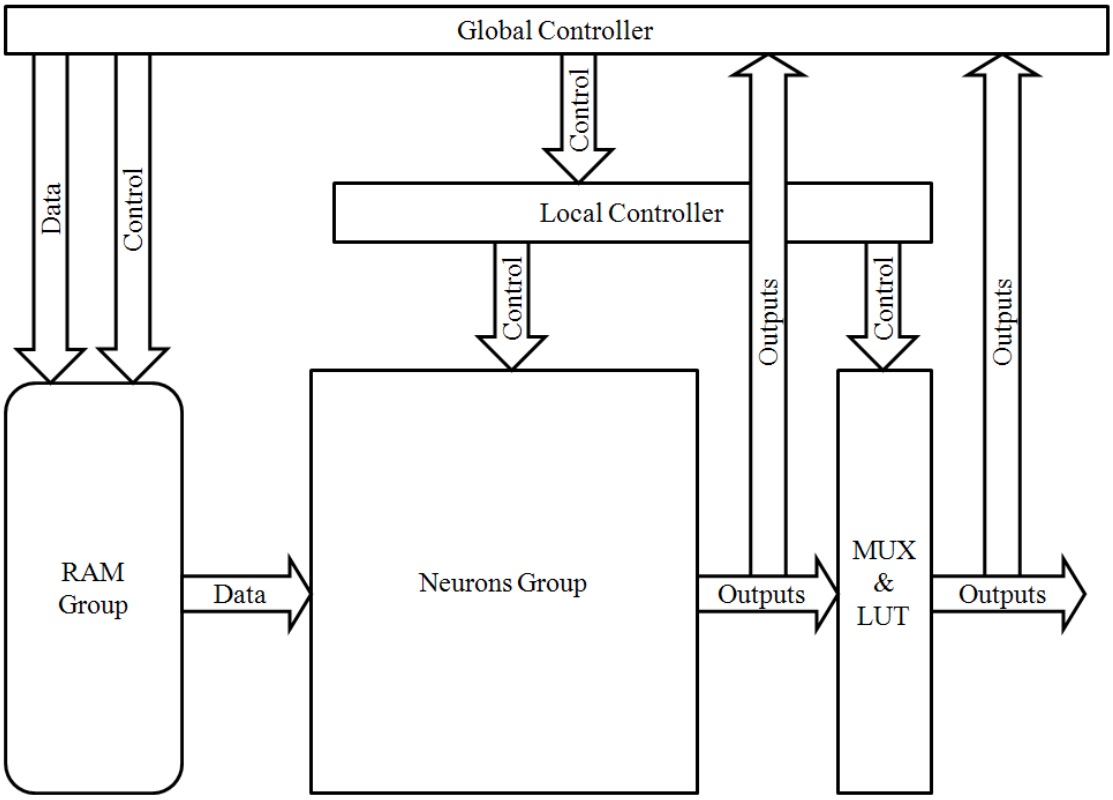


Figure 10. RNA's architecture without Initialization-ROM and ECG-Database-ROM

5.1 THE GLOBAL CONTROLLER

The Global Controller is designed to control the entire system, make all other modules work at the same pace and correct steps for both feed-forward and back-propagation computation. The Global Controller has control signals to the Local Controller, which provide calculation code, start trigger and clear trigger. The Global Controller also provides control signals to each neural-register, which are address, data and enable signals. In order to initialize the Neuron-Register, the

Global Controller also connects to the Initialization-ROM with address, data and enable signals, as well as the connection between the Global Controller and ECG-Database-ROM, including address, data and enable signals. The interface of the Global Controller is shown in figure 11.

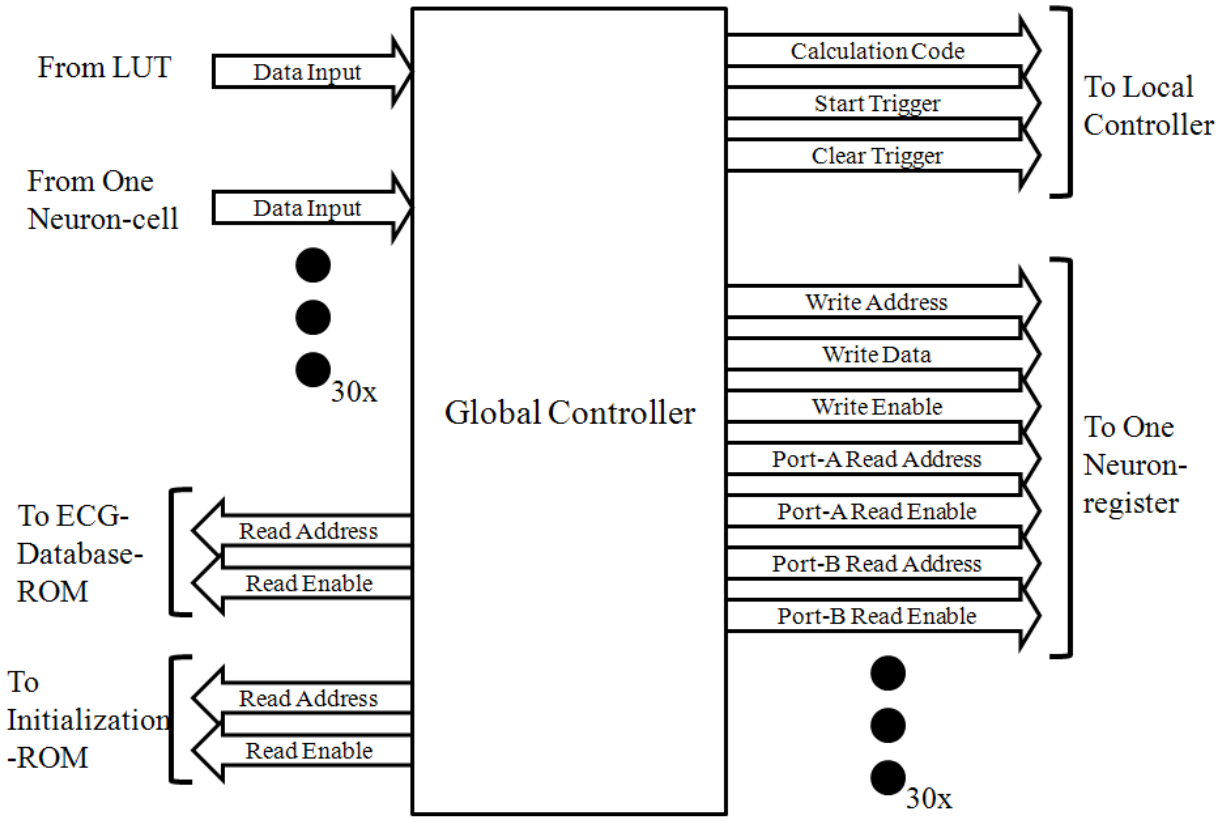


Figure 11. The Global Controller block diagram with interface declared

The Global Controller controls the steps for initializing, training and classification. For the initializing, the controller loads the initial value of all the weights and biases for both from input-layer to hidden-layer and from hidden-layer to output-layer, as well as the parameters such as α, β, 0 and 1, into the Neuron-Registers of each neuron from the Initialization-ROM. After the initializing, the back-propagation training pass for ECG heartbeat begins, and one back-propagation pass is divided into 19 steps, as shown in figure 12.

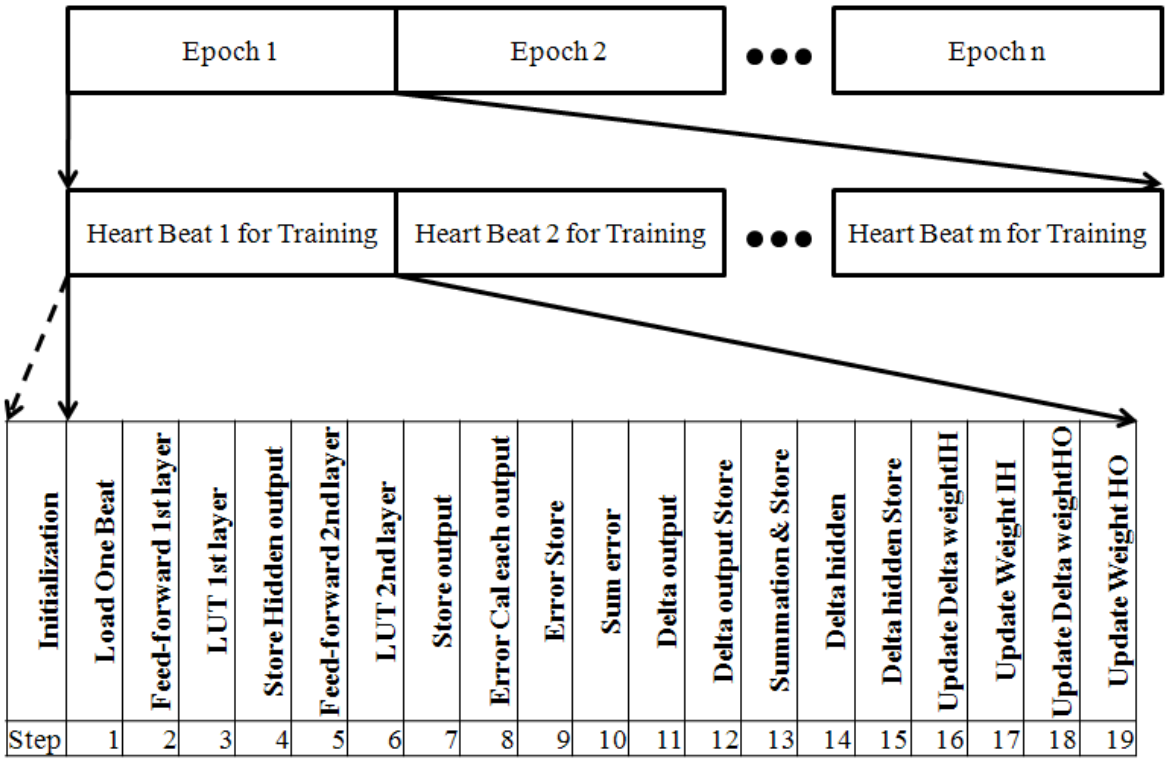


Figure 12. The training process divided by Global Controller

Step 1: the controller sends control signal to ECG-Database-ROM to load one heartbeat data into each Neuron-Register; step 2: perform the Eq. 4.1 for the input-layer to hidden-layer; step 3: perform the Eq. 4.2, since the non-linear activation function is realized by implementing look-up table; step 4: store the output of the neurons in hidden-layer into the Neuron-Registers for later usage; step 5: perform the Eq. 4.3; step 6: perform the Eq. 4.4, and the same look-up table is used as the one in step 3; step 7: store the output of the neurons in output-layer into the Neuron-Registers for later usage; step 8: perform the Eq. 4.14 and Eq. 4.15; step 9: store the error values back input Neuron-Register; step 10: perform the Eq. 4.13 to calculated the squared error energy; step 11: perform the Eq. 4.5; step 12: store the values of delta-output in Neuron-Register; step 13: perform the Eq. 4.6; step 14: perform the Eq. 4.7; step 15: store the values of

delta-hidden in Neuron-Register; step 16: perform the Eq. 4.8; step 17: perform the Eq. 4.10 and Eq. 4.12; step 18: perform the Eq. 4.9 and Eq. 4.11.

After one back-propagation pass from step 1 to step 19, the Global Controller goes back to step one, loading second ECG data for training, and the error energy in step 10 is accumulated in each epoch. Until the last ECG data in ECG-Database-ROM finishes its training, which means an epoch of training finishes, the accumulated error is compared with the target error energy. If the accumulated error energy is larger than the target error energy limits, the Global Controller enters into the next epoch, executing step 1 to step 19 for each ECG beat data in ECG-Database-ROM again. Otherwise, if the accumulated error energy is smaller than the target error limits, the training process is accomplished.

The feed-forward classification pass only contains 7 steps, which are from step 1 to step 7 in figure 10.

When the artificial neural network performs the feed-forward computation from input-layer to hidden-layer, the 30 Neuron-Cells are set as the neurons in hidden-layer, so the weights connected to different hidden Neuron-Cells can be computed in parallel. However the 51 weights connected to each Neuron-Cell in hidden-layer need to perform serially, therefore a counter is needed to perform 51 times iteration for accumulation. Furthermore, within one pass of iteration, the process is also divided into several steps, to perform the detail action. Taking step 2 in figure 12 as an example, the controller sends the read address and read-enable signal to Neuron-Register to read the input value and corresponding weight, and then sends the computation code to the Local Controller, in order to inform the Neuron-Cell perform the correct computation. Once the data is ready in Neuron-Register, the controller sends the trigger to Local Controller to start the computation. After computation, the controller clears the trigger and sends the write

address and write-enable to Neuron-Registers to store the output from Neuron-Cell. At last the write-enable signals are cleared, which is shown in figure 13.

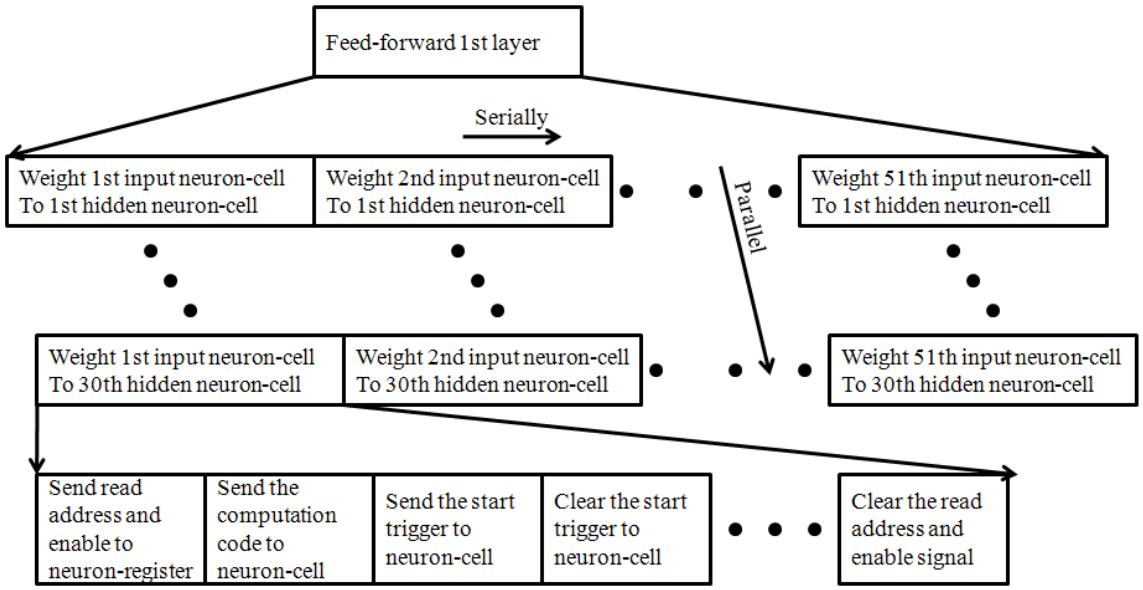


Figure 13. The parallel and series flow of RNA

5.2 THE NEURONS GROUP

The Neurons Group contains 30 Neuron-Cells, which are designed to perform both feed-forward and back-propagation computation. The Neuron-Cell is designed to contain 1 adder, 1 multiplier, several MUXs and registers to perform six kinds of computation, which are $(A-B)^2$, (A-B)*B, A-A*B, A+B, A*B, and $\sum A_i*B_i$, as shown in figure 14.

32 bits are used in the architecture of RNA design: the highest 1 bit is the sign bit (“1” as positive number and “0” as negative number), following 5 bits are integer part, and the rest 26 bits are fraction part. By testing the same model on general PC in C language, we found the inputs, outputs and intermediate value are usually bonded between -32 to 32.

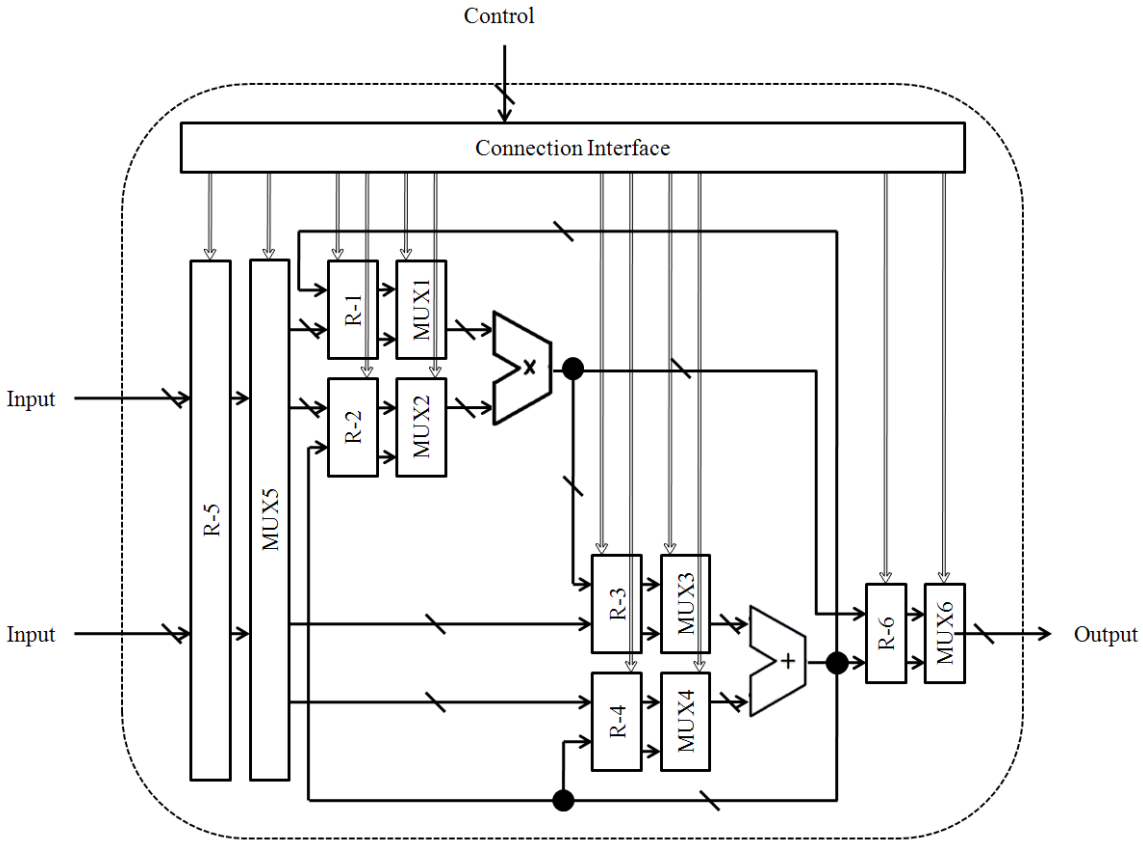


Figure 14. The architecture of one Neuron-Cell in Neuron Group

Adder: The adder used in the design is a 32 bits adder, and in order to protect the overflow, the output is set to “32” if the positive overflow occurs, and the output is set to “-32”, once the negative overflow occurs.

Registers: R-1 to R-6 are 6 registers to store the intermediate data for the adder and multiplier, which are controlled by the Local Controller. 32 bits data line is also used for the inputs and outputs of all the registers.

Multiplier: The fix point multiplier is used in the design. The inputs are firstly converted to the absolute value for multiplying, and then the output of the multiplier is converted to the real value result. The same overflow protection method as adder is used for the multiplier, that is the

output is set to “32” if the positive overflow occurs, and the output is set to “-32”, once the negative overflow occurs.

MUX: 6 MUXs are used in one Neuron-Cell to configure the linking route between adder and multiplier, which are MUX1 to MUX6 in the figure 6. The selection signals for the MUXs are generated by the Local Controller.

Connection Interface: The connection between the Local Controller and the Neuron-Cell is provided by the connection interface, which includes the selection signals to MUXs and store trigger to the registers that the Local Controller generates.

To perform both feed-forward and back-propagation computation, each Neuron-Cell needs to be configured to perform 7 different patterns of computation. For example, to perform the calculation A-A×B, the two inputs of Neuron-Cell, A and B are connected to the multiplier, and then the output of the multiplier and input A are connected to the adder, while the add/sub signal of the adder is set to ‘0’ to perform subtraction, and finally the output of the adder is the output of the whole neuron.

5.3 THE LOCAL CONTROLLER

The Local Controller is designed to receive the calculation code and start trigger and clear trigger signal from the Global Controller and decode them into the selection signals and store triggers, which are sent to MUXs and registers of the Neuron-Cells through the connection interface of each Neuron-Cell, as shown in figure 15. Local Controller also generates the selection signal to the MUX&LUT.

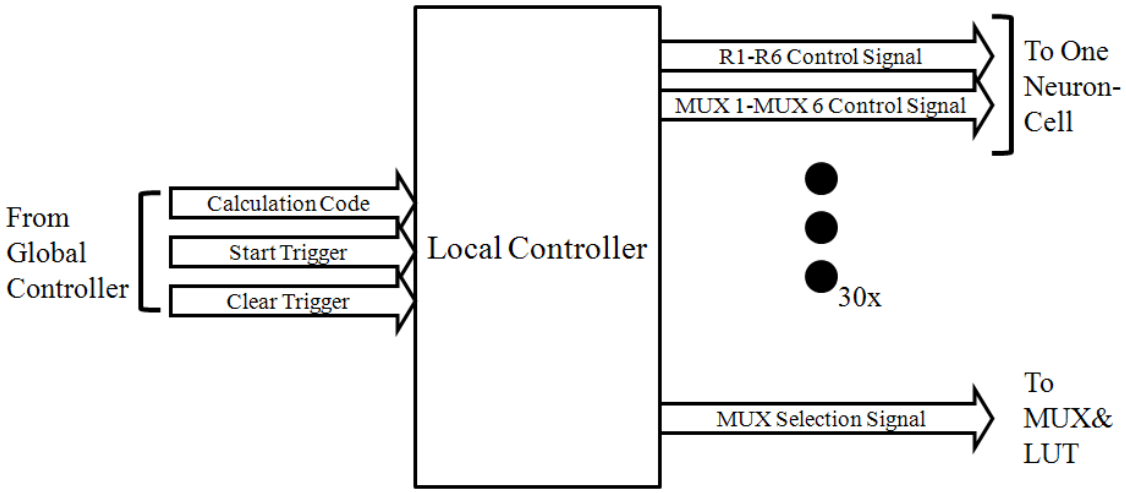


Figure 15. The Local Controller block diagram with interface declared

Calculation codes are used to perform both feed-forward pass (101 and 111) and back-propagation pass (001,010,011,100 and 110) are summarized in table 2.

Table 2. The calculation code and corresponding operations

Code	001	010	011	100	101	110	111
Operation	$(A-B)^2$	A+B	(A-B)XB	A-AXB	AXB+	AXB	LUT
Algorithm	BP	BP	BP	BP	FF	BP	FF

5.4 THE LOOK-UP TABLE

In the design, a non-linear sigmoid function is implemented to perform the activation function, as shown in Eq. 4.2 and Eq. 4.4:

$$y = \frac{1}{1 + e^{-x}}$$

In order to implement the non-linear sigmoid activation function, a division function block is needed to be implemented. However, in S. Himavathi's [29] research, it has been already proved that using a look-up table to implement the non-linear function is much better than the hardware implementation without a look-up table, but by implementing a division function block instead.

Based on their research results, the merits of implementing look-up table instead of implementing division are clearly shown in both saving logic resource and execution time. 70% reduction in resource requirement and 79% improvement in speed have been claimed in their results. Therefore, in order to save the logic resource and improve the execution speed at the same time, the look-up table is built to perform the non-linear sigmoid activation function in this thesis.

Even though designed with a look-up table, it also takes much logic resource usage if building 30 look-up tables, one for each Neuron-Cell. Therefore in RNA design, only one look-up table is implemented. When performing the calculation code "111" for searching look-up table, the Neuron-Cells search the look-up table in turn as time division, and the Local Controller generates the selection signal for the MUX before the LUT. The output of LUT is connected to the Global Controller to deliver the final result back to Global Controller. The look-up table block with interface is shown in figure 16.

From the mathematics model, an accuracy of 0.1 is enough for the input the look-up table, which is also verified by our testing program. Therefore, in order to save the logic resource and limit the chip area, only highest 10 bits of the input are validated for searching look-up table, which result in total 2048 words in look-up table, from "0000000000" to "1111111111", that from -32.9375 to 31.9375, and the step is 0.0625.

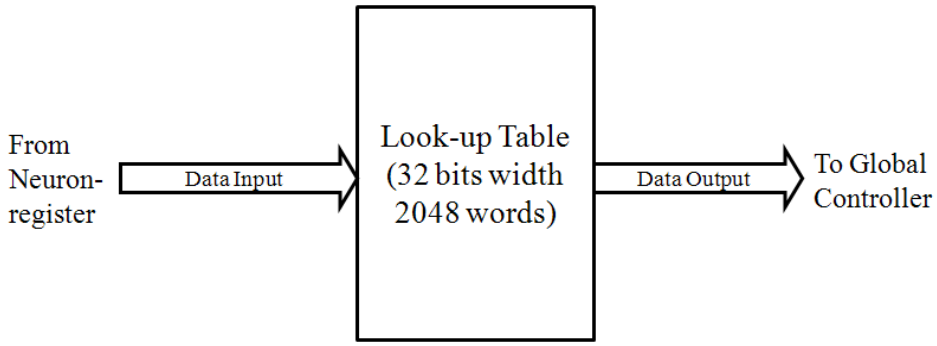
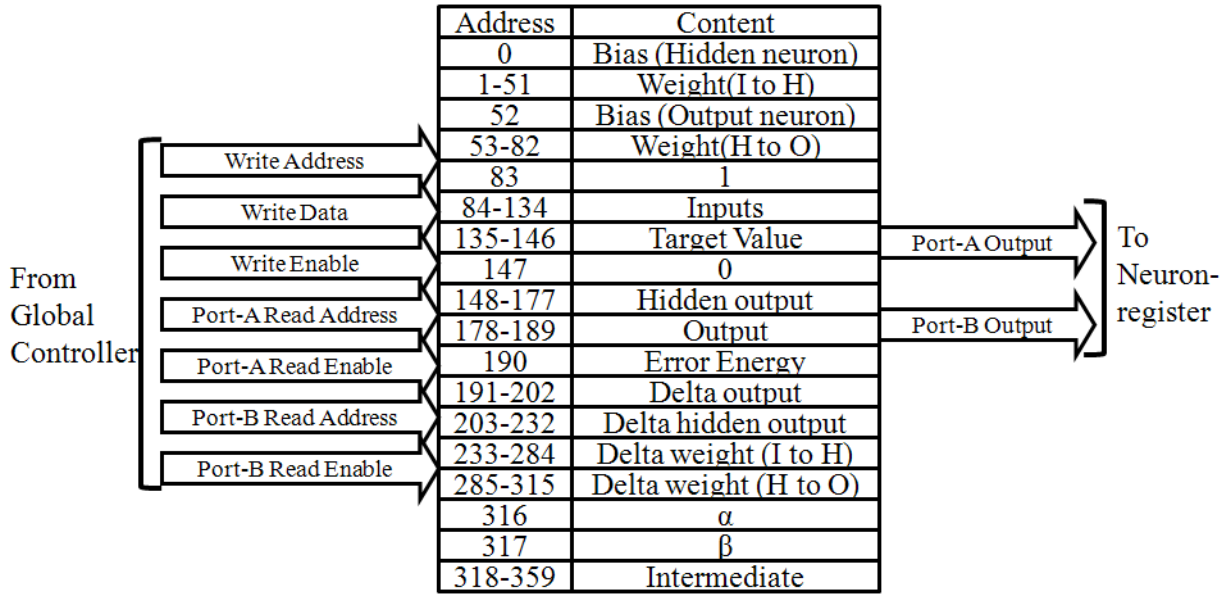


Figure 16. The Look-up Table block diagram with interface declared

5.5 THE RAM GROUP

The RAM Group in figure 10 contains 30 Neuron-Registers, and each of them is a dual-port read and one-port write RAM block, which has the size of 360 words at 32 bits wide for each word. Each Neuron-Cell has a Neuron-Register, to store the two sets of weights and biases, which are those from input-layer to hidden-layer and from hidden-layer to output-layer.



Address	Content
0	Bias (Hidden neuron)
1-51	Weight(I to H)
52	Bias (Output neuron)
53-82	Weight(H to O)
83	1
84-134	Inputs
135-146	Target Value
147	0
148-177	Hidden output
178-189	Output
190	Error Energy
191-202	Delta output
203-232	Delta hidden output
233-284	Delta weight (I to H)
285-315	Delta weight (H to O)
316	α
317	β
318-359	Intermediate

Figure 17. The Neuron-Register storage architecture with interface declared

Since each Neuron-Cell comes with a Neuron-Register, a total of 30 Neuron-Registers are implemented in this design, and each Neuron-Register connects to 1 Neuron-Cell with a dual-port reads. Besides two sets of weights and biases, the intermediate results are also stored in the Neuron-Register, and the storage structure for each Neuron-Register and interface declaration is shown in figure 17.

5.6 INITIALIZATION-ROM AND ECG-DATABASE-ROM

The Initialization-ROM is designed to store the values that are used to initialize the value of each word in the Neuron-Registers at beginning, including the initial value of the weights and biases of both input-layer to hidden-layer and hidden-layer to output-layer, as well as the parameters, such as 0, 1, α and β. Therefore, at the very beginning of the training process, the Global Controller sends address and read-enable signal to Initialization-ROM, and loads the output of Initialization-ROM into the Neuron-Registers, by sending write-enable and write address to the Neuron-Registers.

The ECG-Database-ROM is designed to store all the input ECG data of artificial neural network, which are the amplitude value of ECG heartbeats in binary format, and the target output value of each Neuron-Cell in the output-layer. The size of the ECG-Database-ROM depends on the size of the training database. As a prototype, a design with 32 bits width and 252 words is implemented for the 1st version, which contains 4 heartbeats, or 4 ECG signals. Each heartbeat has 51 data as input and 12 data as target value, so 63 words are required to store for one ECG heartbeat signal.

6.0 RNA IMPLEMENTATION

6.1 IMPLEMENTATION PROCEDURE

The basic modules are encoded by HDL code, including the Neuron-Cells, Global Controller, Local Controller, and look-up table. The Neural-Registers are implemented by the dual-port RAM from the Design Compiler library, and the Initialization-ROM and ECG-Database-ROM are implemented by the ROM generator.

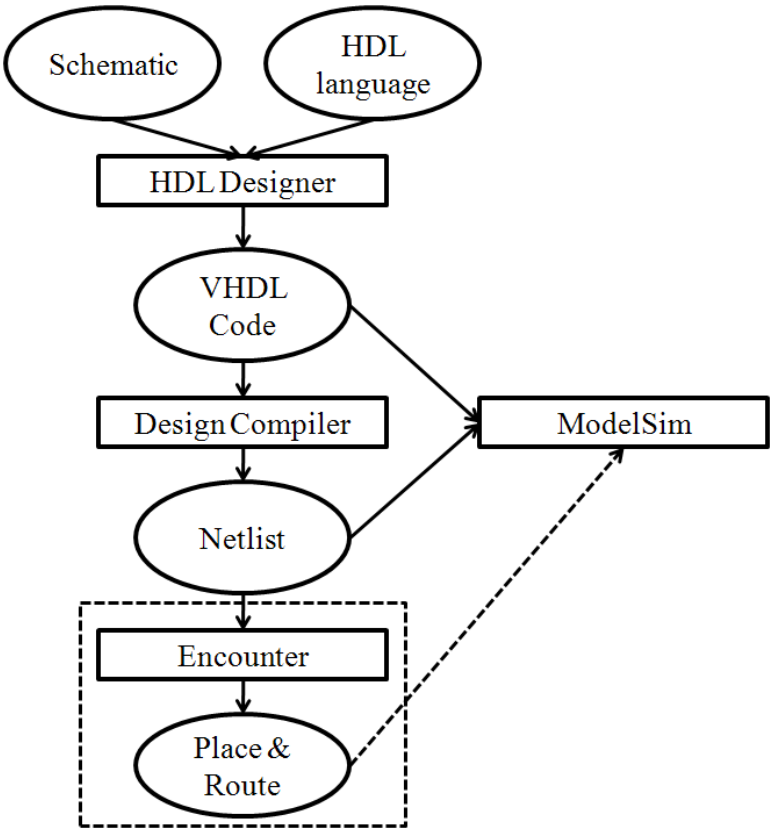


Figure 18. RNA ASIC implementation procedure

The ModelSim is used to simulate each module individually as the pre-synthesis simulation first of all. After that, each module block is synthesized by Design Compiler, and simulated individually by ModelSim, as the post-synthesis simulation. For the entire system, the HDL Designer is used to generate the HDL code for the combination of modules firstly, and then the pre-synthesis simulation is performed by ModelSim. Based on the HDL code of whole system, the synthesis is taken by Design Compiler to obtain the netlist, which is used to perform the post-synthesis simulation (also by ModelSim). The library used in this paper is based on the 45nm CMOS technology. The implementation procedure is shown in figure 18. Since this work focuses on front design, the “Place & Route” step is discussed in future work part.

6.2 IMPLEMENTATION RESULTS

After being linked to the 45nm CMOS technology library by the Design Compiler, the basic modules of the proposed design were synthesized. By running the area, power and timing reports in the synthesis, the area size, dynamic power consumption, leakage power consumption and critical path time delay information were obtained. The area size, dynamic power and leakage power consumption of each basic logic module are shown in Table 3, including Global Controller, Local Controller, look-up table and single Neuron-Cell.

Since much more advanced techniques of implementing RAM and ROM are already available besides that of the library used in this paper, only the logic parts of the RNA system were implemented together and synthesized, including Global Controller, Local Controller, 1 look-up table and 30 Neuron-Cells. The area size, dynamic power and leakage power consumption of RNA system (logic part) were also presented in Table 3.

Table 3. The synthesis results of component blocks and logic system of RNA

RNA	Area (um^2)	Dynamic Power (uW)	Leakage Power (uW)
Global Controller	100355.1	47300.0	455.5
Single Neuron-Cell	19517.2	1452.9	120.4
Local Controller	67.6	3.1	0.3
Look-up Table	7405.1	371.1	33.6
RNA: Logic Part	693343.8	91261.2	4101.4

6.3 SUMMARY AND DISCUSSION

From the synthesis result, the entire system only occupied less than 0.7 mm^2 of area size, 91.3 mW of dynamic power consumption and 4.1 mW of leakage power consumption, compatible to the goal of achieving small physical size and power consumption. Since the Local Controller was simply implemented as a group of decoders, the area size and power consumption of the Local Controller only took a small part of the entire system. The Neuron-Cell is the most resource consuming module in all the logic parts, and total 30 Neuron-Cells occupied roughly 84% of the logic parts according to Table 3. Therefore, reusing Neuron-Cells is significant by dramatically saving the resources of the entire system. The area size of the Global Controller is roughly 5.0x bigger than that of a Neuron-Cell, so the layer reusing technique is optimal, compared to adding 12 more Neuron-Cells without layer reusing, if only considering the feed-forward pass. It saves much more if considering the back-propagation part, which is discussed in comparison ASIC design. Besides, instead of implementing 30 LUT, only implementing 1 LUT also saves 31% area size of the entire logic part design.

7.0 COMPARISON ASIC DESIGNS

In order to compare the merits of the logic source saving and power consumption saving by using the resource reusing technique in the proposed design RNA, 3 other architectures of the same artificial neural network were also explored and implemented in this thesis, which were Flat design, Lightweight-Neuron design and Layer-Reused design. The Flat design, is the artificial neural network architecture without any resource reuse technique, the Lightweight-Neuron design is based on the Flat design, reuses some part of the logic resource, and the Layer-Reused design is based on Lightweight-Neuron design, reuses more logic resource.

7.1 FLAT DESIGN

The Flat design, having the biggest area size and power consumption, but the shortest time delay for the computation, is the system architecture without any layer or logic resource reused. Since no resource is reused in this design, there is no RAM needed in the system. The prior research only implemented the feed-forward algorithm of the Flat design, such as P. Masa [40]. In this thesis, the completed system, containing both feed-forward pass and back-propagation pass is implemented, as shown in figure 19. The grey area is for the back-propagation pass, and the rest part is for the feed-forward pass.

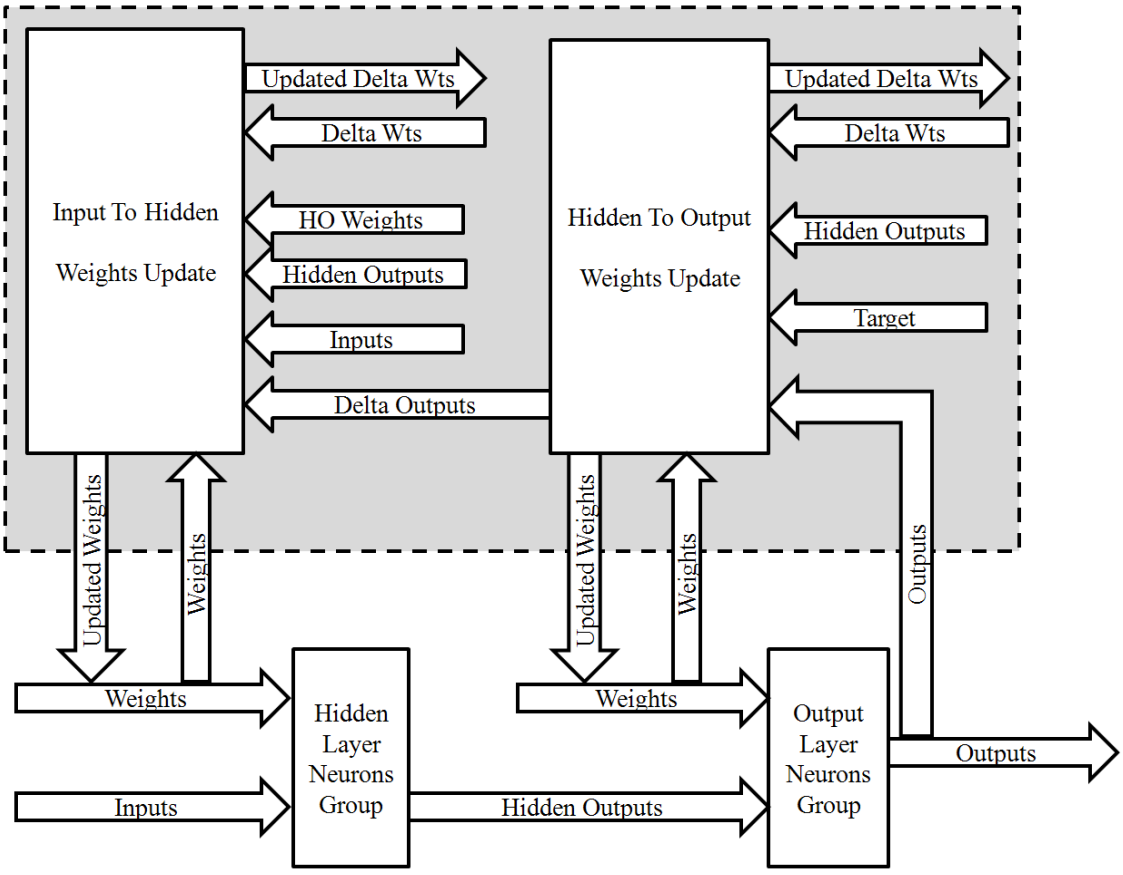


Figure 19. Flat design system block diagram

Hidden Layer Neurons Group is designed for the computation from input-layer to hidden-layer in the feed-forward pass. In the Hidden Layer Neurons Group module, 30 neurons were implemented, since we have 30 neurons in the hidden-layer as mentioned in the Methodology section. Each neuron has 51 2-input multipliers, 1 52-input adder and 1 Look-up Table. Each 2-input multiplier is design to perform the multiplication of one input among total 51 inputs and corresponding weight from that neuron in input-layer to this neuron in hidden-layer, and the 52-input adder is designed to summarize these 51 production and a bias for this neuron, at last the look-up table is designed to perform the non-linear sigmoid activation function. The neuron architecture design is shown in figure 20.

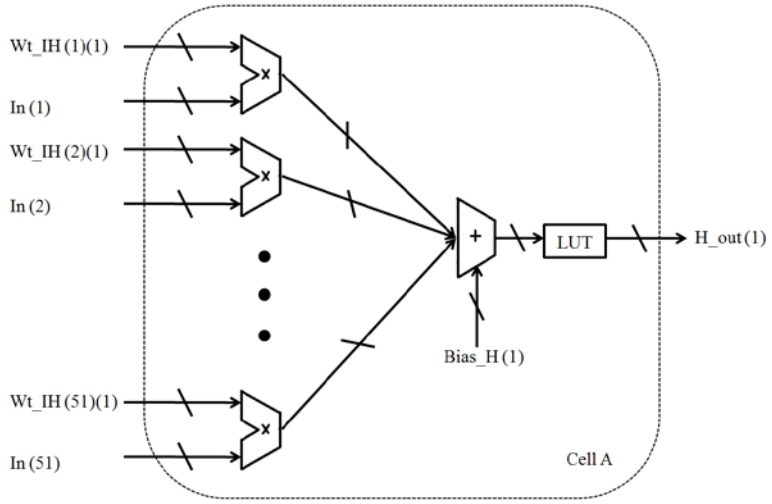


Figure 20. The neuron architecture in Hidden Layer Neurons Group

This structure performs the Eq. 4.1 and Eq. 4.2 mentioned in the methodology section.

In the Output Layer Neurons Group module, 12 neurons were implemented. Each neuron has 30 2-input multipliers, 1 31-input adder and 1 Look-up Table, since the inputs number of the output-layer is 30, same as the output number of the hidden-layer, and the adder also performs the summation of the time products and bias, which is shown in figure 21.

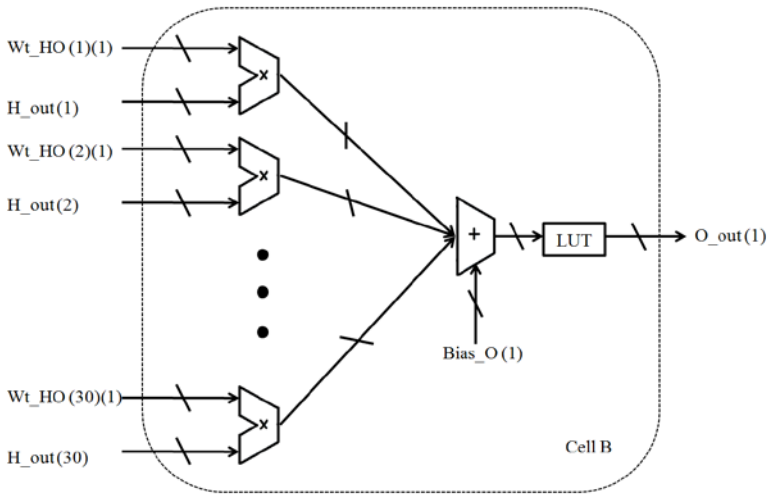


Figure 21. The neuron architecture in Output Layer Neurons Group

This structure performs the Eq. 4.3 and Eq. 4.4. The 2 inputs for each multiplier are the output from hidden-layer and the corresponding weights from that neuron to the neuron in output-layer. Total 30 inputs in the hidden-layer feed in one input of each multiplier, and the other input is fed by the weights from each neuron in hidden-layer to this neuron.

In the Hidden To Output Weights Update module, 12 basic cells are implemented. Each of them contains 1 Cell-A and 31 Cell-B, which are used to calculate the Del_O and DelWt_HO, Wt_HO and Bias, as shown in figure 22.

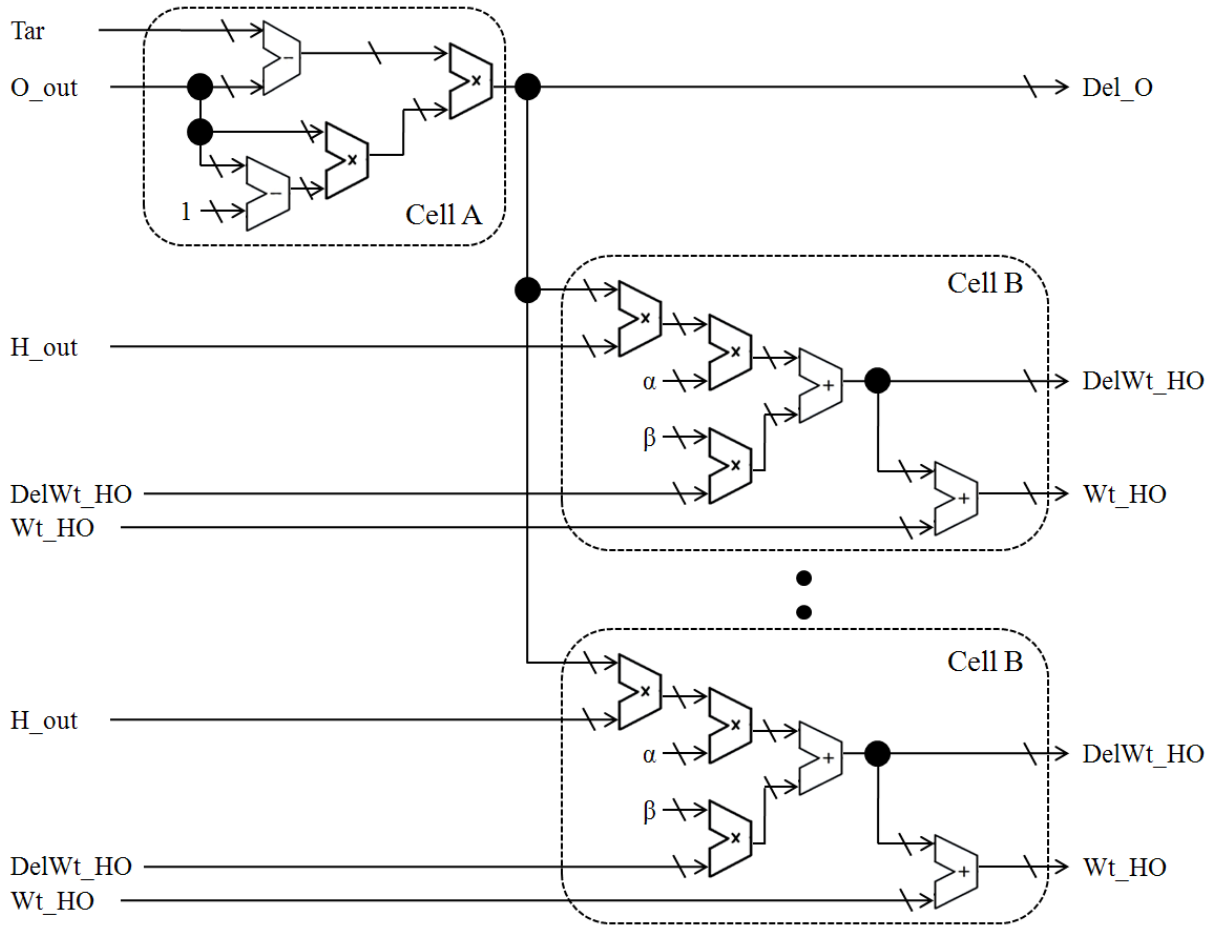


Figure 22. Basic cell for Hidden To Output Weights Update

The Cell-A is the structure of figure 22 that performs the Eq. 4.5, and each Cell-B is the structure that performs Eq. 4.9, 4.11 and 4.13. Each Cell-B calculates the weights and bias from one neuron in hidden-layer to this neuron in output-layer, so 31 Cell-B calculate the weights and bias from all the neurons in hidden-layer to this neuron in output-layer. Furthermore, 12 basic cells cover all the neurons in output-layer.

In the Input To Hidden Weights Update module, 30 basic cells are implemented, each of them contains 1 Cell-C and 52 Cell-B, which are used to calculate the Del_H and DelWt_IH, Wt_IH and Bias, as shown in figure 23.

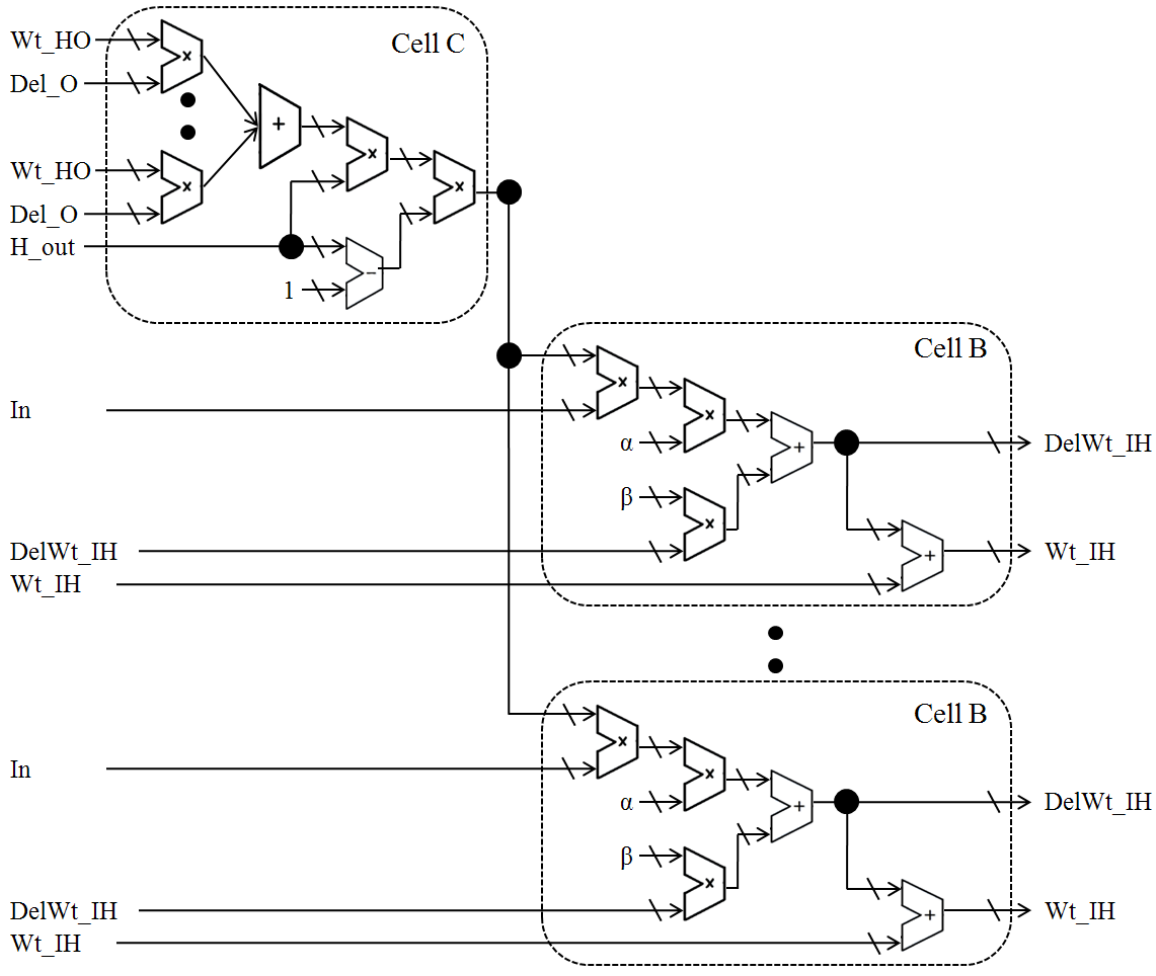


Figure 23. Basic cell for Input To Hidden Weights Update

The Cell-C is the structure that performs the Eq. 4.6 and 4.7, and each Cell-B is the structure that performs Eq. 4.8, 4.10 and 4.12. Each Cell-B calculates the weights and bias from one neuron in input-layer to this neuron in hidden-layer, so 52 Cell-B calculate the weights and bias from all the neurons in input-layer to this neuron in hidden-layer. Furthermore, 30 basic cells cover all the neurons in hidden-layer.

7.2 LIGHTWEIGHT-NEURON DESIGN

The Lightweight-Neuron design is an architecture which reused multipliers in each neuron for feed-forward computation, and also reused the Cell-B and the multipliers of Cell-C in figure 23 for the back-propagation computation. The Lightweight-Neuron design has much smaller area size and power consumption compared to the Flat design, while the time delay for the Lightweight-Neuron design is longer than the Flat design. As shown in figure 24, the grey area is for the back-propagation, and the rest part is for the feed-forward. A Global Controller is needed to control the RAM groups and steps for feed-forward and back-propagation process, which is simpler but different from the one for RNA.

In the Hidden Layer Neurons Group module of the Lightweight-Neuron design, 30 neurons are implemented, as discussed in the methodology section. Each neuron only has 1 2-input multiplier, 1 2-input adder, 1 Look-up Table, and 1 MUX, which is added to perform the accumulation, instead of implementing many multipliers. In other words, the parallel summation of the multiply products in Flat design is changed to the serial summation of the multiply products as time division in Lightweight-Neuron design. The schematic diagram of each neuron is shown in figure 25.

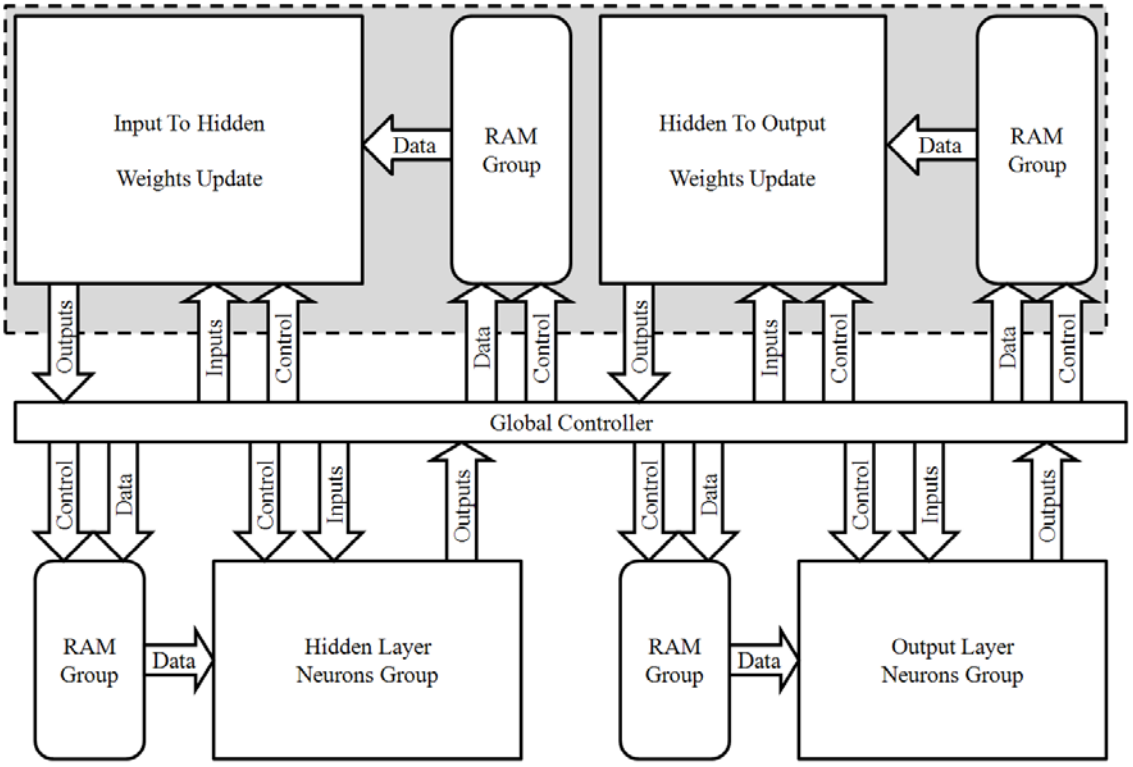


Figure 24. Lightweight-Neuron design system block diagram

This structure performs the Eq. 4.1 and Eq. 4.2. The 2 inputs for each multiplier are the input from input-layer and the corresponding weights from that input to the neuron in hidden-layer. Controlled by the Global Controller, the 2 inputs change periodically to perform the summation of total 51 inputs in the input-layer and their corresponding weights.

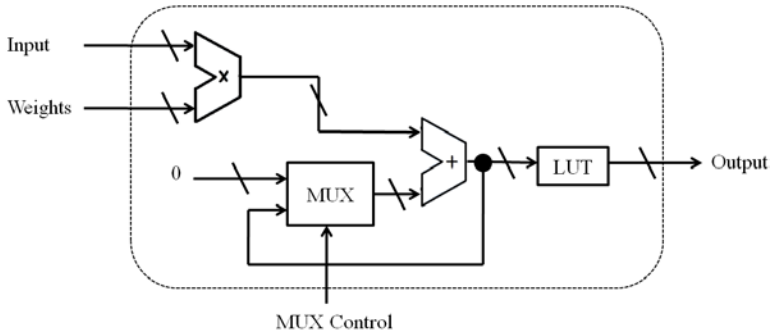


Figure 25. Basic cell in Hidden Layer Neurons Group

In the Output Layer Neurons Group module, 12 neurons were implemented. Each neuron has 1 2-input multiplier, 1 2-input adder and 1 Look-up Table, and each neuron is the same as the neuron in the Hidden Layer Neurons Group model, as shown in figure 26.

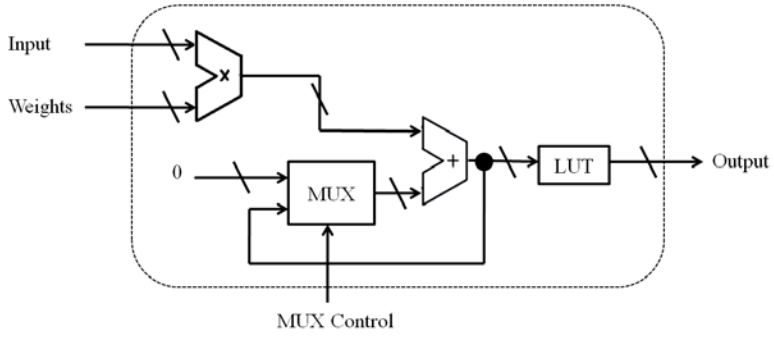


Figure 26. Basic cell in Output Layer Neurons Group

This structure performs the Eq. 4.3 and Eq. 4.4. The 2 inputs for each multiplier are the output from hidden-layer and the corresponding weights from that neuron in hidden-layer to the neuron in output-layer. Controlled by the Global Controller, the 2 inputs change periodically to perform the summation of total 30 outputs of the hidden-layer and their corresponding weights.

Each neuron has a one-port RAM to assist, 30 RAM blocks form the RAM Group for the Hidden Layer Neurons Group, and 12 RAM blocks form the RAM Group for Output Layer Neurons Group. In the RAM, the Weights and Biases are stored. When performing feed-forward, the Global Controller broadcasts the inputs, and also provides the control signal to RAM accordingly, so that the RAM can provide the weights and biases to neurons accordingly.

In the Hidden To Output Weights Update module, 12 basic are implemented. Each of them contains 1 Cell-A and 1 Cell-B, which are used to calculate the Del_O and DelWt_HO, Wt_HO and Bias, as shown in figure 27.

The Cell-A is the structure that performs the Eq. 4.5, and each Cell-B is the structure that performs Eq. 4.9, 4.11 and 4.13. Cell-B is reused as time division in this model. Each basic cell

needs 1 one-port RAM to assist, and 12 RAM blocks form the RAM Group for this model. Each RAM stores the 30 sets of DelWt_HO and Wt_HO, which comes from 30 neurons in the hidden-layer to a neuron in the output-layer. When performing back-propagation, the Global Controller will broadcast the Tar, O_out, H_out and also provide control signals to RAM accordingly, so that the RAM can provide the weights, delta-weights and bias to basic cell accordingly.

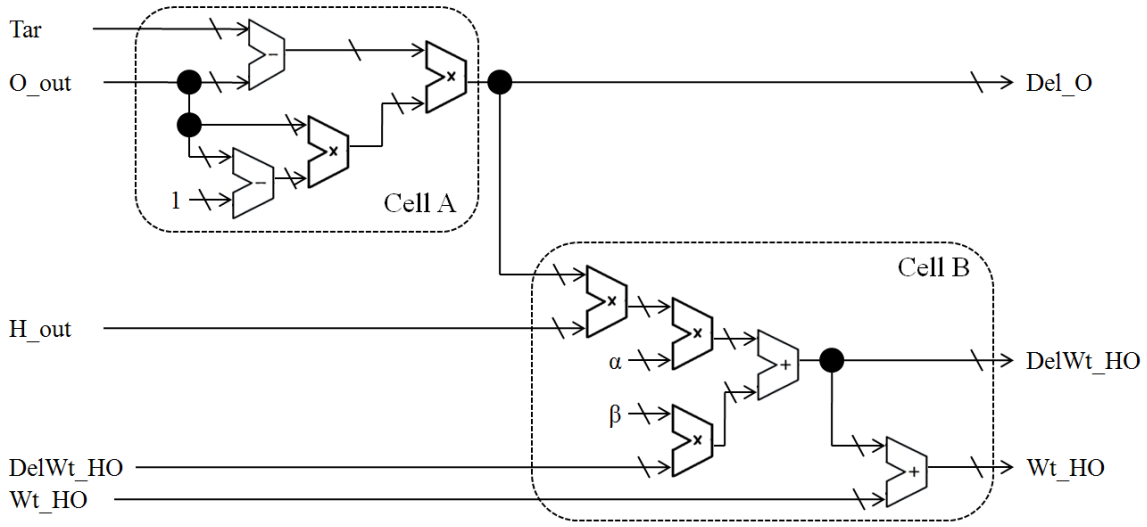


Figure 27. Basic cell in Hidden To Output Weights Update

In the Input To Hidden Weights Update module, 30 basic cells are implemented. Each of them contains 1 Cell-D and 1 Cell-B, which are used to calculate the Del_H and DelWt_IH, Wt_IH and Bias, as shown in figure 28.

The Cell-D is the structure that performs the Eq. 4.6 and 4.7, and each Cell-B is the structure performs Eq. 4.8, 4.10 and 4.12. The difference between this basic cell and Cell-D and Cell-C in Flat design, is that the multiplier is reused in Cell-D and 1 MUX is added to perform the accumulation of the multiply products, instead of implementing 29 more multipliers. Also, Cell-B is reused as time division in this model, calculating the updated weights in parallel into in series. Each basic cell needs 1 one-port RAM to assist, and 30 RAM blocks form the RAM

Group for this model. Each RAM stores the 51 sets of DelWt_IH and Wt_IH, which come from 51 neurons in the input-layer to a neuron in the hidden-layer. When performing back-propagation, the Global Controller broadcasts the H_out / In and also provide control signals to RAM accordingly, so that the RAM can provide the weights, delta-weights and bias to basic cells accordingly.

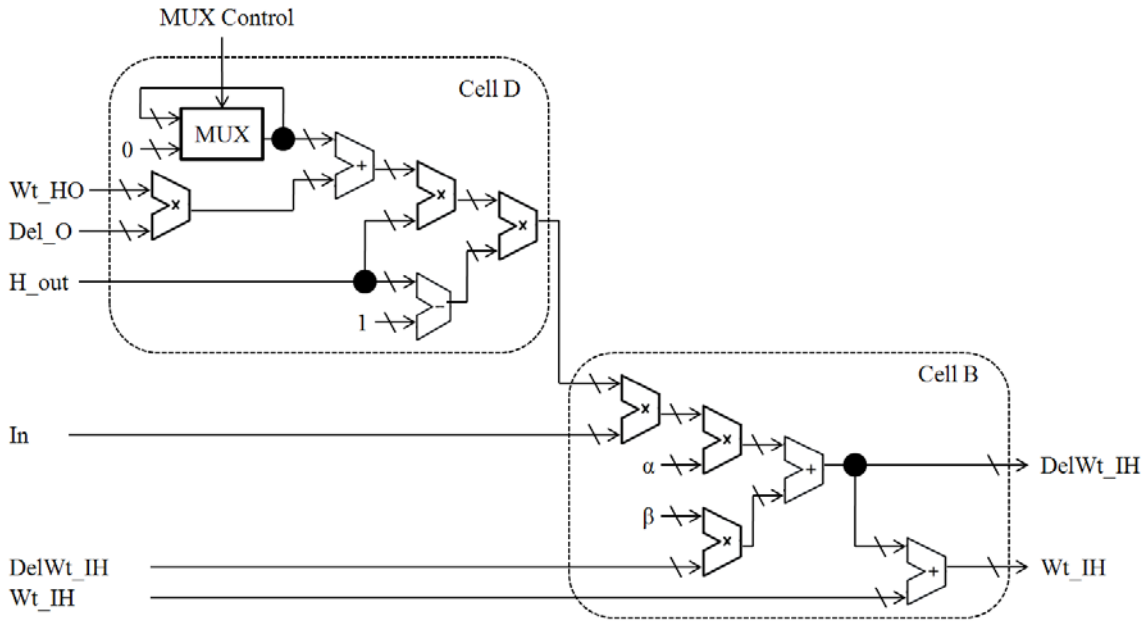


Figure 28. Basic cell in Input To Hidden Weights Update

7.3 LAYER-REUSED DESIGN

The Layer-Reused design is an architecture that reused logic resource as both hidden-layer and output-layer for feed-forward computation, and also reused the basic cell as both Input To Hidden Weights Update and Hidden To Output Weights Update for the back-propagation computation. The prior research only implemented the feed-forward algorithm of the Layer-Reused design, such as S. Himavathi [29], mentioned in the related work section, but in this

thesis, the entire system, containing feed-forward and back-propagation was implemented, as shown in figure 29.

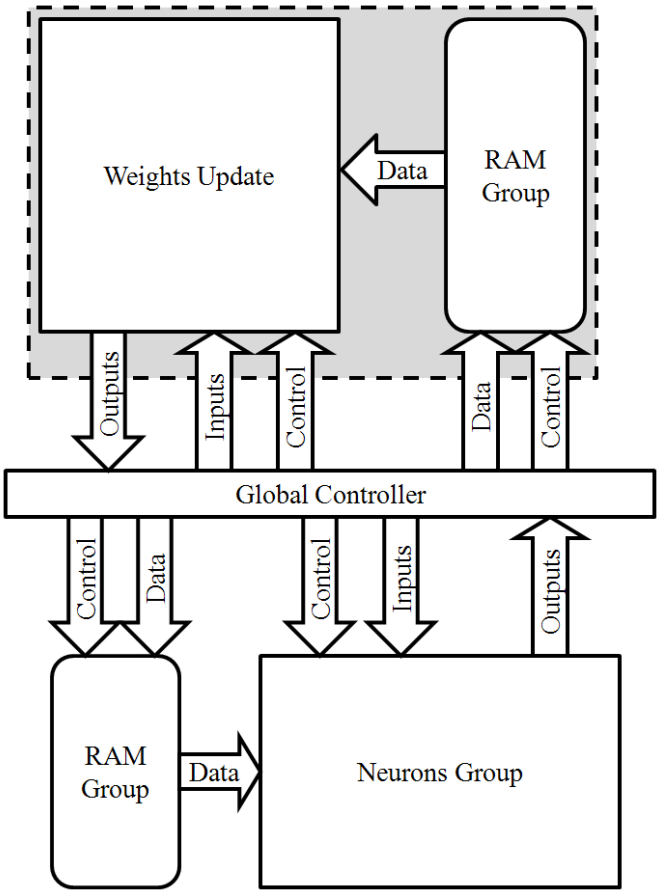


Figure 29. Layer-Reused design system block diagram

The grey area in figure 29 is for the back-propagation calculation, and rest is for the feed-forward calculation. The Layer-Reused design has smaller area size and power consumption, compared to the Lightweight-Neuron and Flat designs, while the time delay of the Layer-Reused design is longer than the Lightweight-Neuron and Flat design.

In the Neurons Group module of the Layer-Reused design, 30 neurons are implemented, as the maximum number of neurons in hidden-layer and output-layer. The same as the Lightweight-Neuron design, each neuron has 1 2-input multiplier, 1 2-input adder, 1 Look-up

Table and 1 MUX that is added to perform the accumulation, instead of implementing many multipliers. The schematic of each neuron is shown in figure 30.

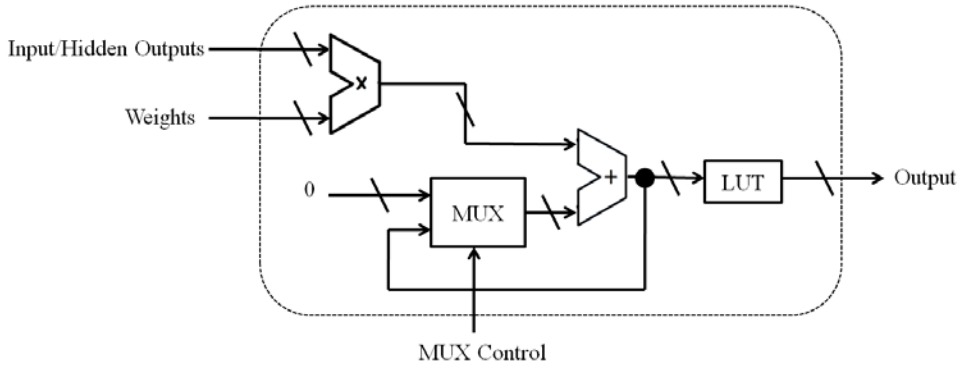


Figure 30. Basic cell in Neurons Group

Each neuron has an one-port RAM to assist, and 30 RAM blocks form the RAM Group for Neurons Group. In the RAM, both the Weights and Bias of input-layer to hidden-layer and hidden-layer to output-layer are stored. When performing feed-forward, the Global Controller broadcasts the inputs, and also provides control signal to RAM accordingly, so that the RAM can provide the weights from the neuron in input-layer to the neuron in hidden-layer and bias at first, and then the Global Controller stores the output value of the neurons in hidden-layer into the RAM, as performing the Eq. 4.1 and 4.2. After that, Global Controller broadcasts the outputs of neurons in hidden-layer that are just stored, and also provides control signal to RAM accordingly, so that the RAM can provide the weights from the neuron in hidden-layer to the neuron in output-layer and bias, and then the Global Controller stores the final output value of the neurons in output-layer into the RAM, as perform the Eq. 4.3 and 4.4.

In the Weights Update module, 30 basic cells are implemented, and each of them contains 1 Cell-A, 1 Cell-D and 1 Cell-B, which are used to calculate both Del_O, DelWt_HO, Wt_HO and Del_H, DelWt_IH, Wt_IH, as shown in figure 31.

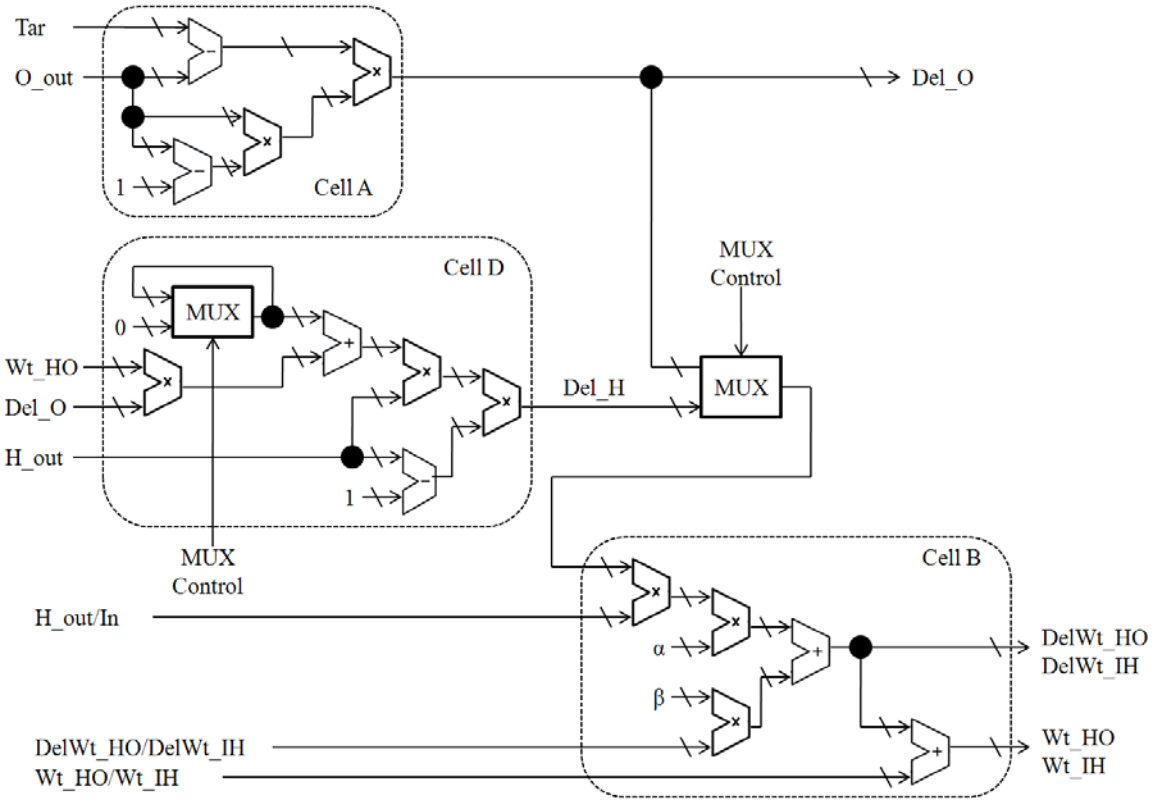


Figure 31. Basic cell in back-propagation weights update

The Cell-A is used to perform Eq. 4.5, the Cell-D performs the Eq. 4.6 and 4.7, and each Cell-B is the structure performs Eq. 4.8, 4.9, 4.10, 4.11 and 4.12. Each basic cell needs 1 one-port RAM to assist, and 30 RAM blocks form the RAM Group for this model. Each RAM stores both the 30 sets of DelWt_HO and Wt_HO, which come from 30 neurons in hidden-layer to a neuron in output-layer, and 51 sets of DelWt_IH and Wt_IH, which come from 51 neurons in input-layer to a neuron in hidden-layer. Cell-B is reused as time division in this model. When performing back-propagation, the Global Controller firstly broadcasts the Tar, O_out to Cell-A, and H_out to Cell-B, and also provides control signal to RAM accordingly, so that the RAM can provide the weights, delta-weights and bias to basic cell accordingly to calculate DelWt_HO and Wt_HO. Then, the Global Controller broadcasts the H_out to Cell-D, In to Cell-B and control

signal to RAM accordingly, so that the RAM can provide the weights, delta-weights and bias to basic cell accordingly, to calculate DelWt_IH and Wt_IH.

7.4 SUMMARY AND DISCUSSION

The implementation procedure of Flat design, Lightweight-Neuron design and Layer-Reused design is similar to the proposed RNA design, as discussed in 6.1. However, several dimensions are compared among the Flat design, Lightweight-Neuron design, Layer-Reused design and the proposed design RNA, including synthesis result of area size, dynamic power, leakage power and time delay, as well as whether a controller is needed, summarized in Table 4.

Since the Flat design, the Lightweight-Neuron design and the Layer-Reused design all have separated circuits to perform feed-forward and back-propagation computation, the synthesis results of these three designs were divided into two parts accordingly, feed-forward and back-propagation, as shown in the Table 4. For the area size, dynamic power and leakage power, the synthesis results of the proposed design were normalized as 1 respectively. The other three designs were calculated as the ratio to the proposed design accordingly. The area size, dynamic power and leakage power of the RAM were obtained from HP-CACTI [41], and also normalized to the proposed RNA. The time delay was obtained by the longest critical path in the design from the synthesis, and these time delays included both logic delay and the RAM access time.

From Table 4, we can observe a 98.7% area size savings in the proposed design, compared to the Flat design, a 99.1% dynamic power savings, and a 30 times overabundance of the time delay. The larger time delay of the proposed design is caused by the logic resource

being reused and the RAM accessing. Then, a trade-off between area size, power and processing speed is obtained.

Table 4. Synthesis results among RNA, Flat, Lightweight-Neuron and Layer-Reused design

		Flat Design	**Lightweight-Neuron Design**	**Layer-Reused Design**	**Proposed Design RNA**
Area Size (um^2)	Feed-Forward	18.26x	0.64x	0.48x	0.41x
	Back-Propagation	61.25x	2.25x	2.06x	
	RAM	0x	0.22x	0.24x	0.59x
	Summary	79.51x	3.11x	2.78x	1
Dynamic Power (uW)	Feed-Forward	25.50x	0.58x	0.43x	0.14x
	Back-Propagation	81.62x	3.91x	3.56x	
	RAM	0x	1.08x	0.85x	0.86x
	Summary	107.12x	5.57x	4.84x	1
Leakage Power (uW)	Feed-Forward	2.06x	0.07x	0.05x	0.05x
	Back-Propagation	6.46x	0.23x	0.20x	
	RAM	0x	0.47x	0.60x	0.95x
	Summary	8.52x	0.77x	0.84x	1
Time Delay (ns)	Feed-Forward	47	301	301	682
	Back-Propagation	32	616	767	1777
	Summary	79	917	1068	2459
Controller		No	Yes	Yes	Yes

Regarding to this ECG training and classification application, the back-propagation learning algorithm is used to train the artificial neural network by the database combined by authorized ECG data and personal data. Since the personal data is only a small part of the

training database, we first focus on the amount of the popular ECG database, which are used for training. One of the most popular ECG database is the MIT-BIH database [39], which contains about 40 patients recording, and each of them has around 30 minutes data recording. Usually, people have average of 80 heartbeats per minute, so total 96000 data are used for training as the authorized database. Multiplying by the epoch times (10000 times iteration), total 960000000 back-propagation passes need for the training. From the synthesis result, 2459 ns is the delay for one back-propagation pass of the proposed design RNA, which means around 39 minutes for the whole training process; 79 ns is the delay for one back-propagation pass of the Flat design, which means around 1.3 minutes for the training process. Usually, 40 minutes' training time is acceptable for user, therefore in order to achieve a resource and power consumption efficient design, the proposed design RNA is primary.

8.0 SIMULATION RESULT AND DISCUSSION

8.1 ECG SIMULATION DATA

Matlab was used to generate the normalized ECG signal waveform from the MIT-BIH database [39]. The output files have the annotation information for each heartbeat, marked in a certain location for each beat (roughly the R points of QRS complex). The amplitude was converted into a 32 bit binary number as the input to artificial neural network. Among the 32 bits, the highest bit was used as the sign bit, and the following 5 bits were used as the integer part, while the last 26 bits were decimal bits.

The sample rate of this extracted waveform is 180 points per second, so 51 points can cover 283ms duration of each heartbeat. We use the amplitude of annotation points (roughly the R points of QRS complex) fed to the 26th neuron of the input-layer. The first 25 neurons of the input-layer are fed by the continual data before the R point, while the last 25 neurons of the input-layer are fed by the data after the R point. As mentioned in the introduction, 283ms can cover the entire QRS complex and also parts of the P wave and T wave. It is also known that the majority of heart disease is classified by the abnormal shape of the QRS complex. Therefore, this method to feed the ECG signal to the ANN is enough to provide the information of the QRS complex to the artificial neural network, and also the structure we used in this project is suitable and compatible with the application of the ECG signal classification.

8.2 RESULTS AND DISCUSSION

For the post-synthesis simulation, 4 heartbeats were used as the training database, which contains 1 normal heartbeat, and 3 abnormal heartbeats (PVC, PACE and RBBB). The heartbeats all came from the patients' records in MIT-BIH database: the normal heartbeat came from patient No.100, the PVC came from patient No.119, the RBBB came from patient No.118, and the PACE came from patient No.104. As introduced in ECG data format, the amplitude and annotation were firstly extracted from MIT-BIH database by Matlab, with the sample rate of 180 points per second. 51 data points were chosen as the input of the artificial neural network, and the center points have the annotation. These 51 data points were converted into 32 bits with the data format. Then targets for training were set for these 4 heartbeats. The 63 data points, including 51 inputs and 12 targets, were stored in the ECG-Database-ROM, as one ECG signal data. And, all 4 ECG signal data were stored in the ECG-Database-ROM in the testing.

Then the post-synthesis simulation was performed by ModelSim, and the result waveform was obtained, as shown in figure 32. From the simulation result, we can focus on the signal "stop", which going to high means the current accumulated error energy is less than the target training requirement error. The other important signal "trainOK", which going to high means the overall accumulated error energy is less than the target training requirement error, in other words, the training is done. From the simulation waveform, we can observe that before 5^{th} epoch, the "stop" goes high as impulse, because the accumulated error is cleared at the end of each epoch. Till the 5^{th} epoch, the "stop" stays high for a duration, which means the accumulated some heartbeat ECG error less than target. The high duration increases as epoch increases, till the 11^{th} epoch, the accumulated error energy for all 4 heartbeats is less than the target, "trainOK" goes to high, which means training is over.

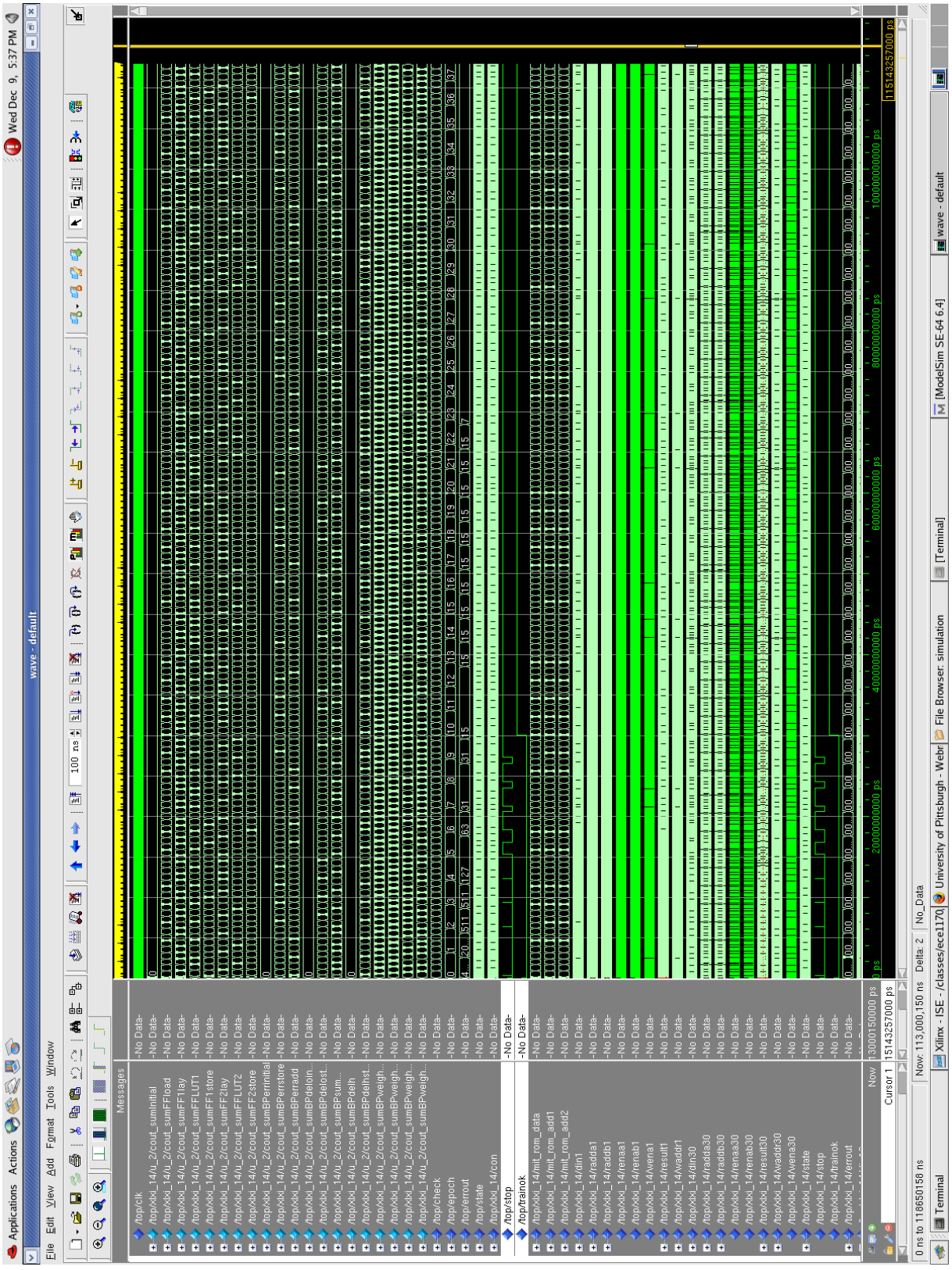


Figure 32. The post-synthesis simulation result of RNA

From the post-synthesis simulation waveform, the outputs of the Neuron-Cells in the output-layer were compared between those before training (epoch 0) and those after training (epoch 10), to observe the convergence of the output toward target value. The error energy was calculated by the end of each epoch. In this research, the threshold of the error energy was set as 0.04, which was achieved after the 10th epoch training. An artificial neural network with the same structure of RNA was implemented on general PC in C language, and all the intermediate data were truncate into the range from -32 to 32 with same resolution of RNA. The same training process with same 4 ECG heartbeats was performed with software solution on PC, to verify the RNA hardware implementation.

The training error energy is plotted in the figure 33, the similar converging trend between RNA hardware implementation and software implementation was obtained.

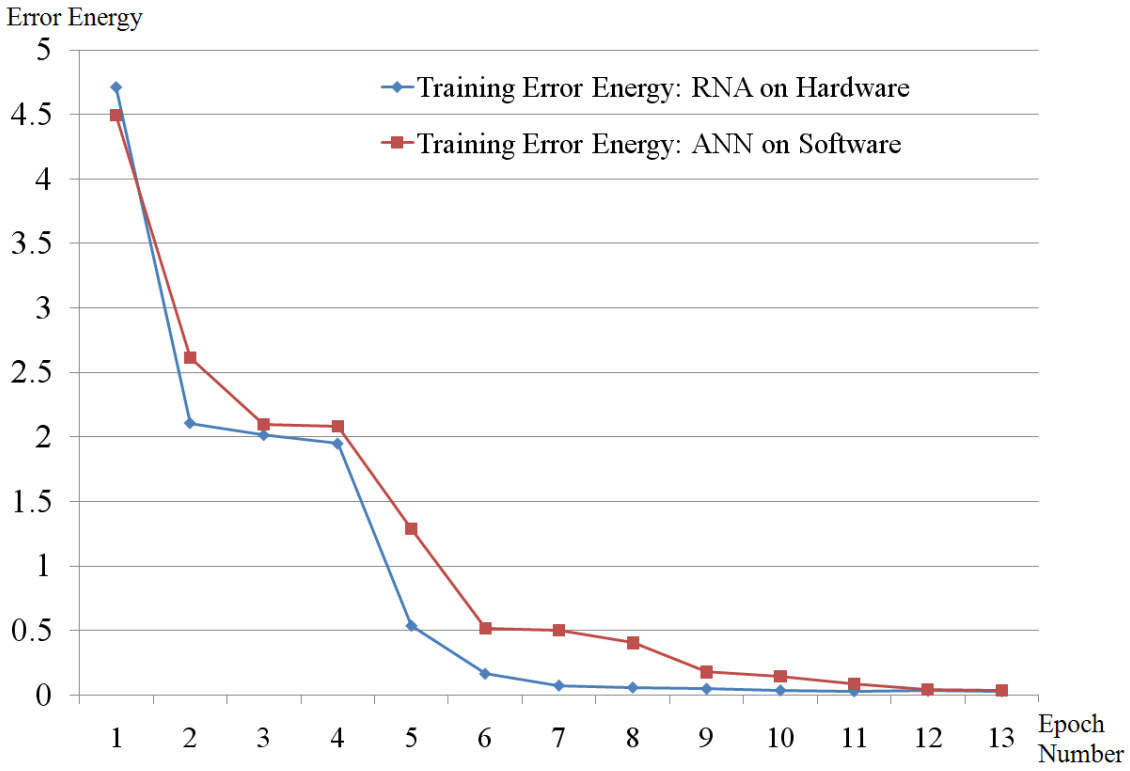


Figure 33. Training error energy converge to zero

9.0 EMBEDDED SOFTWARE IMPLEMENTATION AND DISCUSSION

9.1 EMBEDDED SOFTWARE IMPLEMENTATION

Additionally, we also implemented a software version of the ECG-classifying artificial neural network algorithm in C and compiled the C code using the Visual studio 2008 for the Windows Mobile 5.0 platform. This ANN ECG classifying software is deployed to a Windows Mobile Amoi E72 smartphone, as shown in figure 34.

Figure 34. Amoi E72 smartphone ECG classification display with sensor

Amoi E72 has a TI OMAP 1030 200MHz processor, a 128MB Nandflash, and a 64MB DDR Ram. When performing the comparison between this software baseline and ASIC implementations, both training data and diagnosis data are stored in the phone, emulating their hardware counterparts. The execution time of the program is obtained using the system timer function. The energy consumption is obtained by measuring the current and voltage supply from the battery to the smartphone when running the program.

9.2 IMPLEMENTATION RESULT AND DISCUSSION

The performance analyses include execution time and energy consumption comparisons among three scenarios: 1) Unit Classification, 2) Unit Training, and 3) Database Training. The unit training contains normalized training for only one heart beat, while the database training contains the heart beat data from the entire MIT-BIH database, which includes heart beat data of 40 patients, each of which contains 30-minute ECG recording. Unit classification includes only one heart beat feed-forward classification. We compared both scenarios between the smartphone based artificial neural network software implementation and RNA-ASIC based implementation platform. The comparison results are summarized in figure 35 and figure 36.

Our experimental results show that our cost-effective RNA ASIC implementation has significant advantages over its software counterpart running on the smartphone: approximately 5000x speed-up and 4000x reduction in energy consumption. Training the smartphone software implementation using the entire MIT-BIH database can take 2700 hours or more than 16 weeks to complete, while it takes only 40 minutes in the RNA ASIC. The three orders of magnitude speed and energy improvement of RNA-ASIC implementation over the smartphone

implementation clearly illustrate the benefits to have a cost-effective hardware to perform artificial neural network.

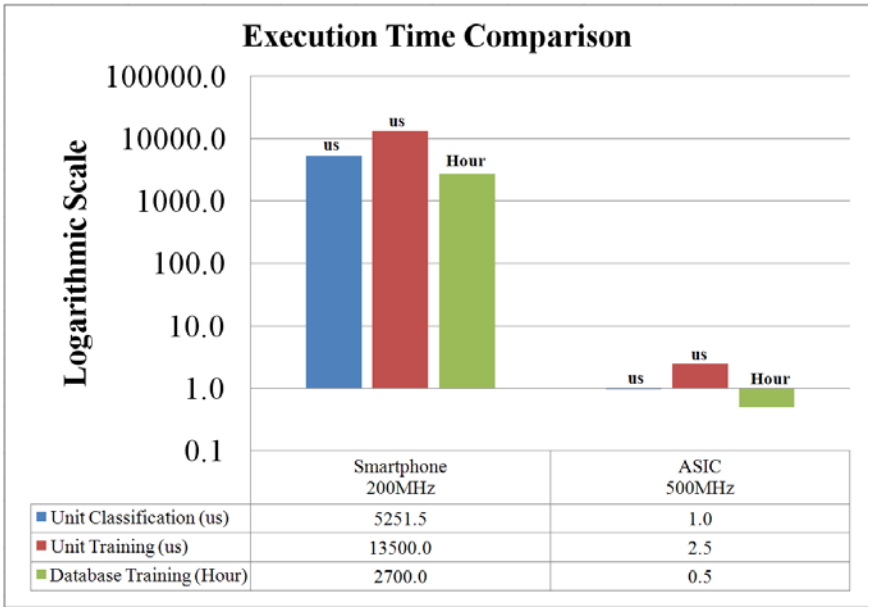


Figure 35. Execution time comparison between smartphone and RNA ASIC

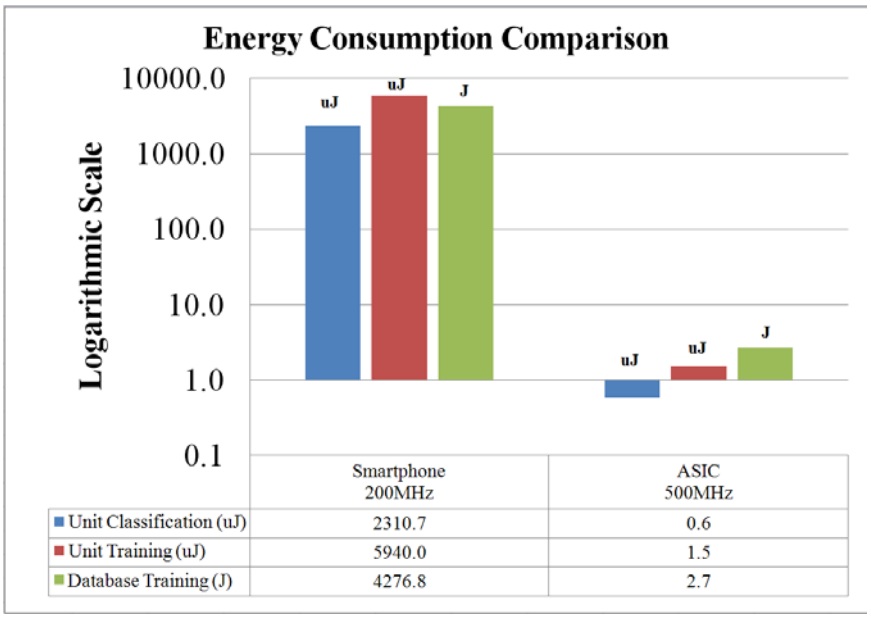


Figure 36. Energy consumption comparison between smartphone and RNA ASIC

10.0 FPGA DEMO IMPLEMENTATION

The same architecture of the RNA, but a smaller version with only 16 bits data-path, is implemented on Xilinx Vertex-5 ML505 XC5VLX110T FPGA board, in order to build the demo of this RNA artificial neural network, as shown in figure 37.

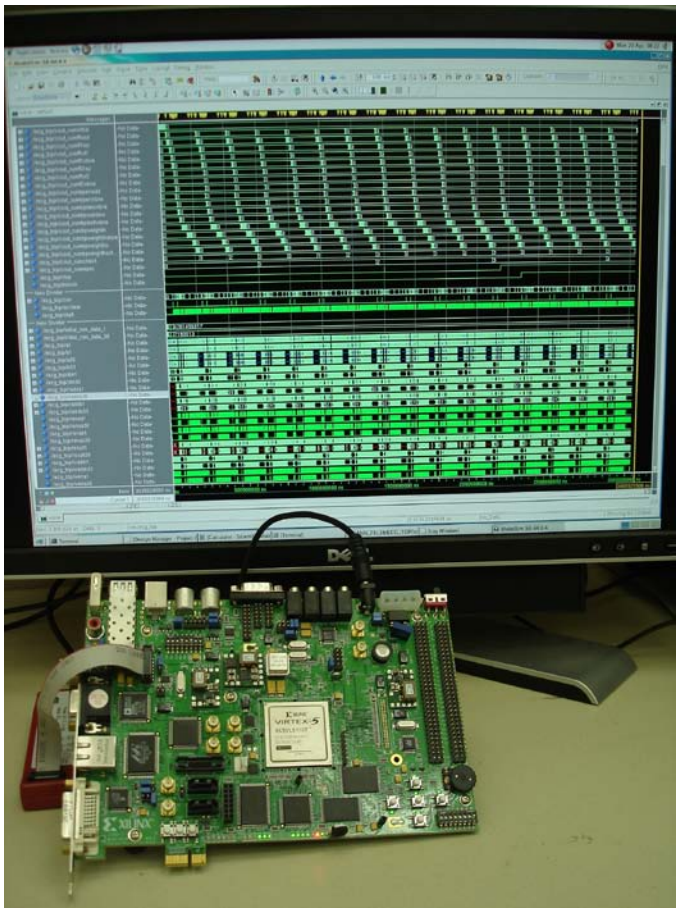


Figure 37. Xilinx ml505 FPGA V-5 board platform and simulation result display on screen

10.1 DESIGN AND IMPLEMENTATION

The same functional Global Controller is implemented on FPGA, with the 16 bits data path, shorter from the original 32 bits. The verilog code is synthesized, translated, mapped and placed & routed by the Xilinx tools ISE. The generated hardware circuit after synthesized is shown in figure 38.

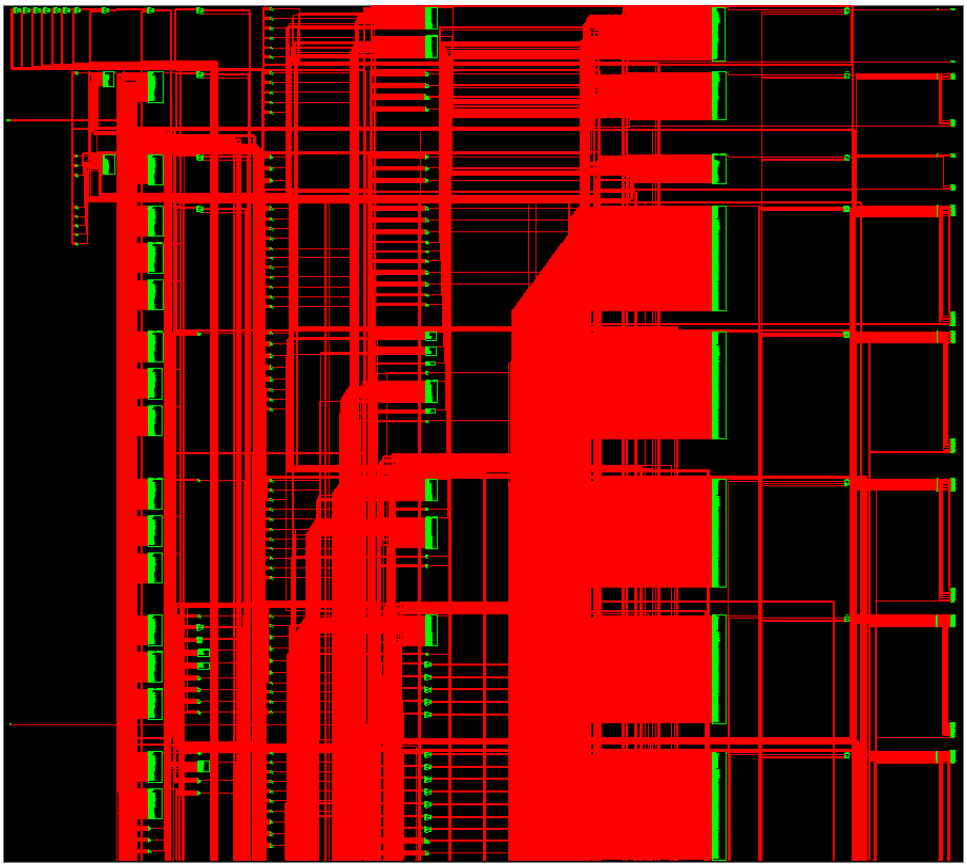


Figure 38. The Global Controller circuit implementation after synthesis

Since the Global Controller is mainly based on if-else and counter, the resource usage of the Global Controller is primary logic resource, including LUTs and registers. The synthesis report is summarized in table 5.

Table 5. The Global Controller synthesis results

Global Controller	Used	Available	Utilization
Number of Slice Registers	1897	69120	2%
Number of Slice LUTs	4370	69120	6%
Number of fully used LUT-FF pairs	1882	4385	42%
Number of bonded IOBs	2699	640	---
Number of BUFG/BUFGCTRLs	1	32	3%

The Neuron-Cells are implemented on FPGA, with the 16 bits data path. The VHDL code is synthesized, translated, mapped and placed & routed by the Xilinx tools ISE. The generated hardware circuit for a single Neuron-Cell after synthesized is shown in figure 39.

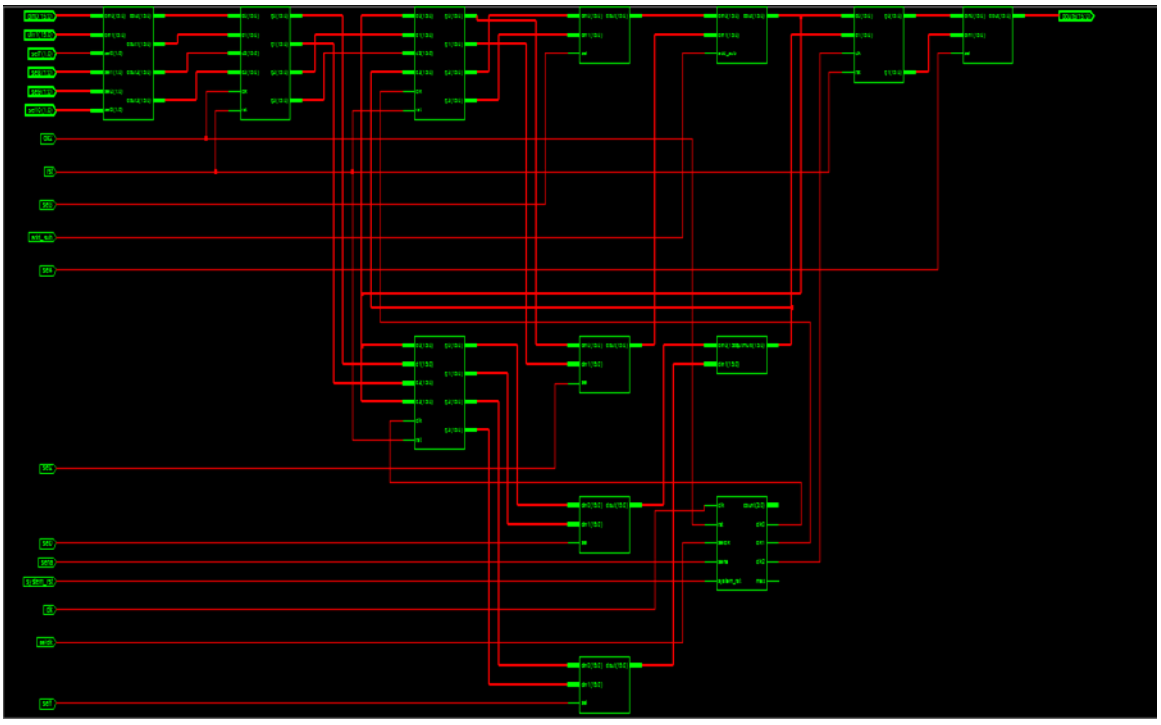


Figure 39. The Neuron-Cell circuit implementation after synthesis

Since each Neuron-Cell has a multiplier implemented, one DSP48e block is programmed to perform the multiply function, which is good for saving the logic resource and also speeds up the multiplying. The synthesis report of a single Neuron-Cell is summarized in table 6.

Table 6. The single Neuron-Cell synthesis results

Single Neuron-Cell	Used	Available	Utilization
Number of Slice Registers	215	69120	0%
Number of Slice LUTs	235	69120	0%
Number of fully used LUT-FF pairs	181	269	67%
Number of bonded IOBs	68	640	---
Number of DSP48Es	1	64	2%
Number of BUFG/BUFGCTRLs	5	32	15%

The look-up table and the Local Controller are both as same as the ones in RNA, and the only difference between them are the width of the data-path, changing from 32 bits to 16 bits. The synthesis report of the look-up table and Local Controller are combined and summarized in table 7 and table 8.

Table 7. The Local Controller synthesis results

Local Controller	Used	Available	Utilization
Number of Slice LUTs	12	69120	0%
Number of bonded IOBs	19	640	---

Table 8. The look-up table synthesis results

Look-up Table	Used	Available	Utilization
Number of Slice LUTs	134	69120	0%
Number of bonded IOBs	26	640	---

Combining all the module blocks, includes 1 Global Controller, 1 Local Controller, 30 Neuron-Cells as 1 neuron group, 30 Neuron-Registers as 1 RAM group, and 1 look-up table together to form the RNA core, the core architecture is synthesized, translated, mapped and placed & routed by the Xilinx tools ISE, and the synthesized report is summarized in table 9.

Table 9. The RNA core synthesis results

RNA Core Design	Used	Available	Utilization
Number of Slice Registers	8507	69120	12%
Number of Slice LUTs	13408	69120	19%
Number of fully used LUT-FF pairs	7381	14534	50%
Number of bonded IOBs	74	640	11%
Number of Block RAM/FIFO	30	148	20%
Number of DSP48Es	30	64	46%
Number of BUFG/BUFGCTRLs	6	32	18%

In order to make the entire system successfully running on the FPGA board, and easy to observe the results, two more modules are added on the system, besides the ECG-Database-ROM, which are clkadd and display. Clkadd is used to provide the proper system clock

frequency to FPGA user, and display is the model to convert the error energy data into other form. The entire system diagram is shown in figure 40.

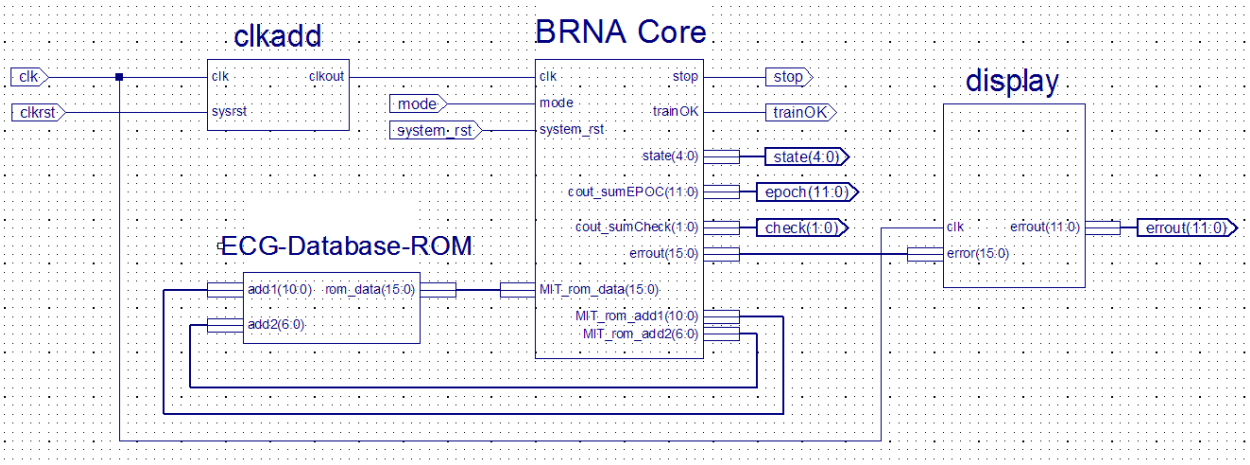


Figure 40. The schematic diagram of entire system on FPGA implementation

And the synthesis result of this entire system is summarized in table 10.

Table 10. The entire FPGA implementation synthesis results

RNA Design	Used	Available	Utilization
Number of Slice Registers	8520	69120	12%
Number of Slice LUTs	13731	69120	19%
Number of fully used LUT-FF pairs	7448	14803	50%
Number of bonded IOBs	37	640	5%
Number of Block RAM/FIFO	30	148	20%
Number of DSP48Es	30	64	46%
Number of BUFG/BUFGCTRLs	7	32	21%

10.2 DEMO DISPLAY

In order to show the outputs on demo clearly, GPIO of the Xilinx ml505 FPGA board is used to display the output signals with LEDs, including error energy, epoch number, heartbeat choosing number, state of the state machine, and the single heartbeat train satisfied signal and whole heartbeat database train finish signal, as shown in figure 41 and 42.

The LEDs are divided into 5 groups, from left to right, 12 LEDs in first group to display error energy, 2 LEDs in second group to display the training situations, 6 LEDs in third group to display epoch numbers, 2 LEDs in forth group to display heartbeat index and 5 LEDs in the last group to display the states of the state machine in global controller.

The LEDs in error energy group, from left to right mean, larger than 16, larger than 8, larger than 4, larger than 2, larger than 1, larger than 0.5, larger than 0.25, larger than 0.125, larger than 0.0625, larger than 0.03125, larger than 0.015625 and larger than 0.0078125. Therefore, at the beginning of the training, all 12 LEDs may be lighted, because the error energy is the biggest before the training, larger than 16. However, as training going on, the error energy is decreased as epoch numbers increased, and the LEDs vanishes from left to night. The single train ok is the signal that symbols that the error energy is less than target value at the right now, while the group train ok signal means the training is over. And then, the epoch signal records the total epochs RNA need to train these four data. Heartbeat index presents which heartbeat is under processing, which is alternated from 0 to 3 as the epoch numbers increased. The LEDs for state display the steps that controlled by Global Controller, which are total 19 mentioned in Global Controller section.

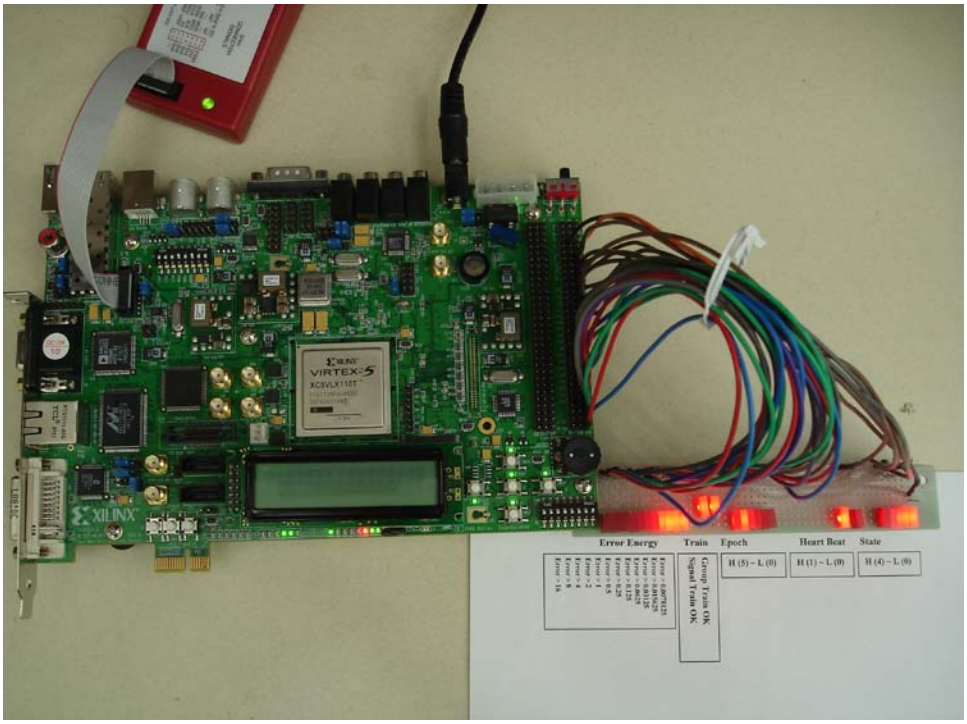


Figure 41. The FPGA RNA demo with external LED board

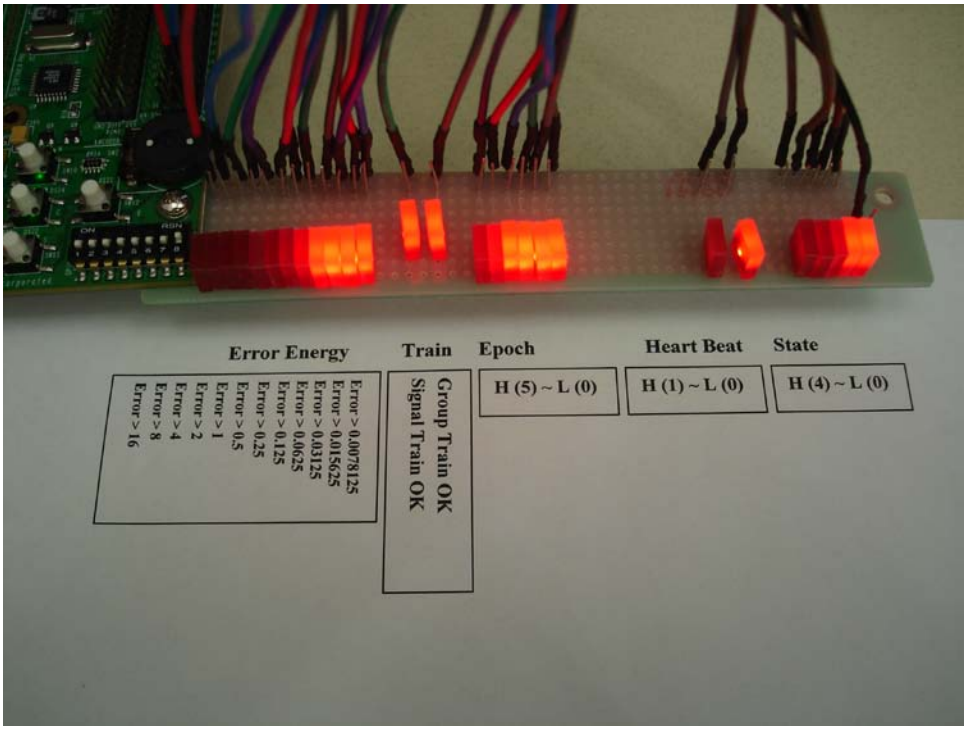


Figure 42. The explanation of LEDs on external LED board

11.0 CONCLUSION

In this thesis, we firstly proposed a design of artificial neural network for ECG signal classification, with Neuron-Cells reused as neurons in the hidden-layer and output-layer, and the design can perform both feed-forward and back-propagation computation implemented on ASIC with 45nm technology. The 51-30-12 artificial neural network model was built on general PC in C language, and total 413 ECG data with 12 kinds of heartbeats were divided into training set and testing set. With the artificial neural network trained, an average accuracy of 95.2% in classification was obtained from testing set. Furthermore, the model has been explored by increasing/decreasing the number of neurons in hidden-layer and changing the parameter in sigmoid function. The numbers of training epoch to achieve the same error energy were compared and discussed, as well as the accuracy of the testing process.

By synthesis result, the total area is 1.70 mm^2, which includes the logic parts' area of 0.7 mm^2, and RAM part's area of 1.0 mm^2. The logic part contains the Global Controller, Local Controller, 30 Neuron-Cells and 1 look-up table. The total dynamic power consumption is 662.6 mW, which includes the logic parts' 91.3 mW, and the RAM part's 571.3 mW. The total leakage power consumption is 89.6 mW, which includes logic parts' 4.1 mW, and RAM part's 85.5 mW. The time delay for a feed-forward and a back-propagation is less than 2.5 us.

4 heartbeats were selected from MIT-BIH database to perform the feed-forward and back-propagation computation, which contained 1 normal heartbeat, and 3 different kinds of

disease heartbeats. By post-synthesis simulation, the training target was achieved by the 10 epochs, and the error energy convergence was obtained, which was also verified by comparing to the result of running same artificial neural network structure on general PC in C language.

In this thesis, the other 3 architectures of artificial neural network were also implemented and tested, which are the Flat design, Lightweight-Neuron design and Layer-Reused design. By comparing the results of area size, power consumption and time delay, the proposed design is optimal for the ECG application, because of the smallest area size and least power consumption among these 4 designs, with an acceptable time delay. By comparing to the Flat design, a 98.7% area size saving, a 99.1% dynamic power savings, and a 30 times overabundance of the time delay are obtained by the proposed design RNA. One back-propagation pass of the proposed design RNA has 2.5 ns delay, which means around 39 minutes for the whole training process; 79 ns is the delay for one back-propagation pass of the Flat design, which means around 1.3 minutes for the training process.

Besides the RNA ASIC design, a C language version of our artificial neural network is also implemented in Amoi E72, the smartphone. The execution time and power consumption are measured in our experimental, which show that our cost-effective RNA ASIC implementation has significant advantages over its software counterpart: approximately 5000x speed-up and 4000x reduction in energy consumption. By this execution time speed-up, training the artificial neural network on smartphone software with the entire MIT-BIH database can take 2700 hours or more than 16 weeks to complete, while it takes only 39 minutes in the RNA ASIC, which is acceptable for users. In this thesis, a 16 bits data-path RNA design is also implemented on Xilinx ml505, a Virtex-5 FPGA board for verification and demonstration. With this demo, the training and classification are performed on board, and learning curve of the training process is observed.

12.0 FUTURE WORKS

First, the Placing and Routing will be completed, based on the netlist generated from synthesis results, and then the real chip can be fabricated and tested.

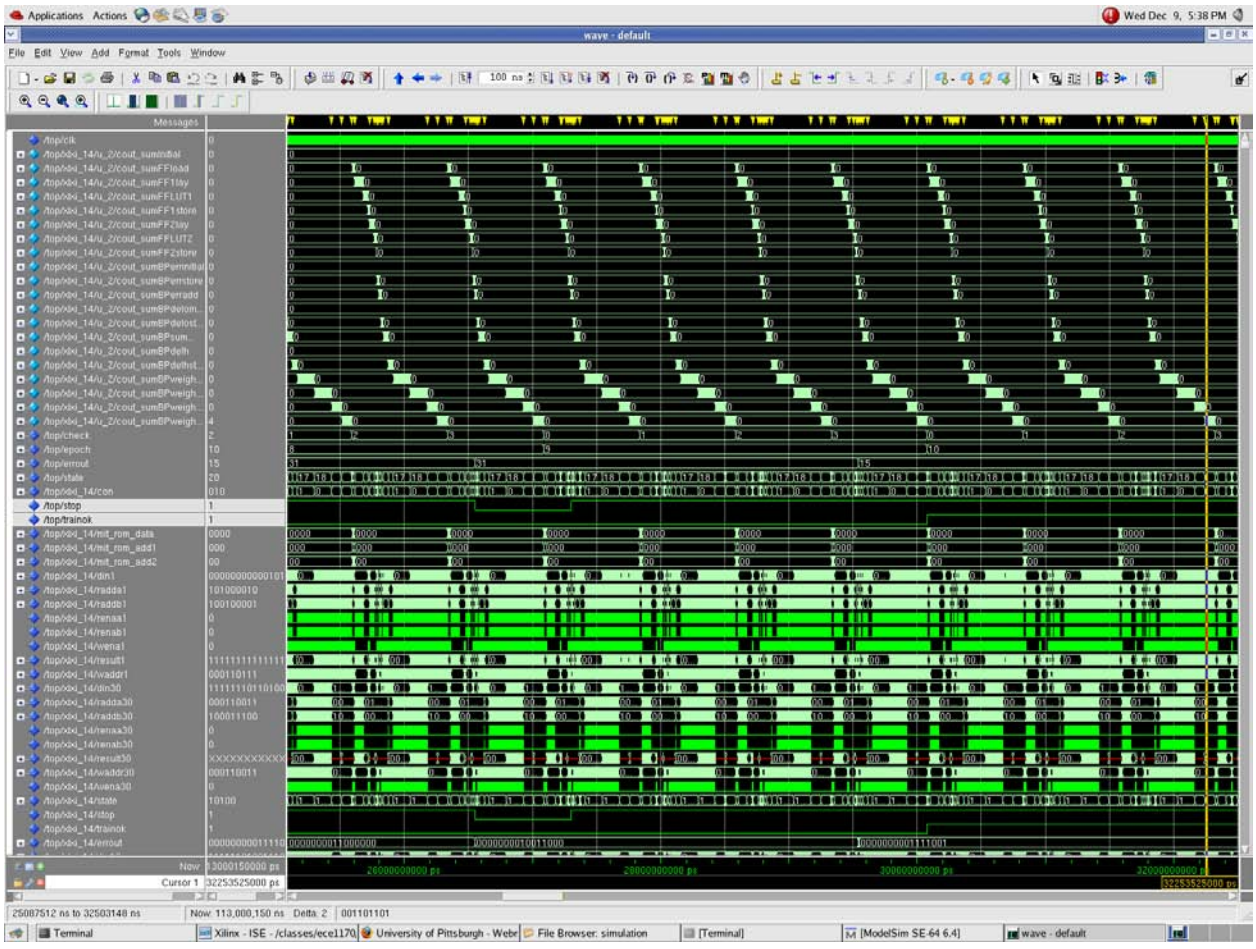
Second, the biomedical sensors will be added in the front, and the sensor will be small, comfortable for wearing and easily changeable, which also has simple and durable interface for the chip to click on.

Third, an amplifier and feature extraction parts are also needed, to make the entire system operate with the ECG signal live and tested by a real patient.

At last, a Bluetooth module will be added on the chip for easily communication between the chip and smartphone or laptop.

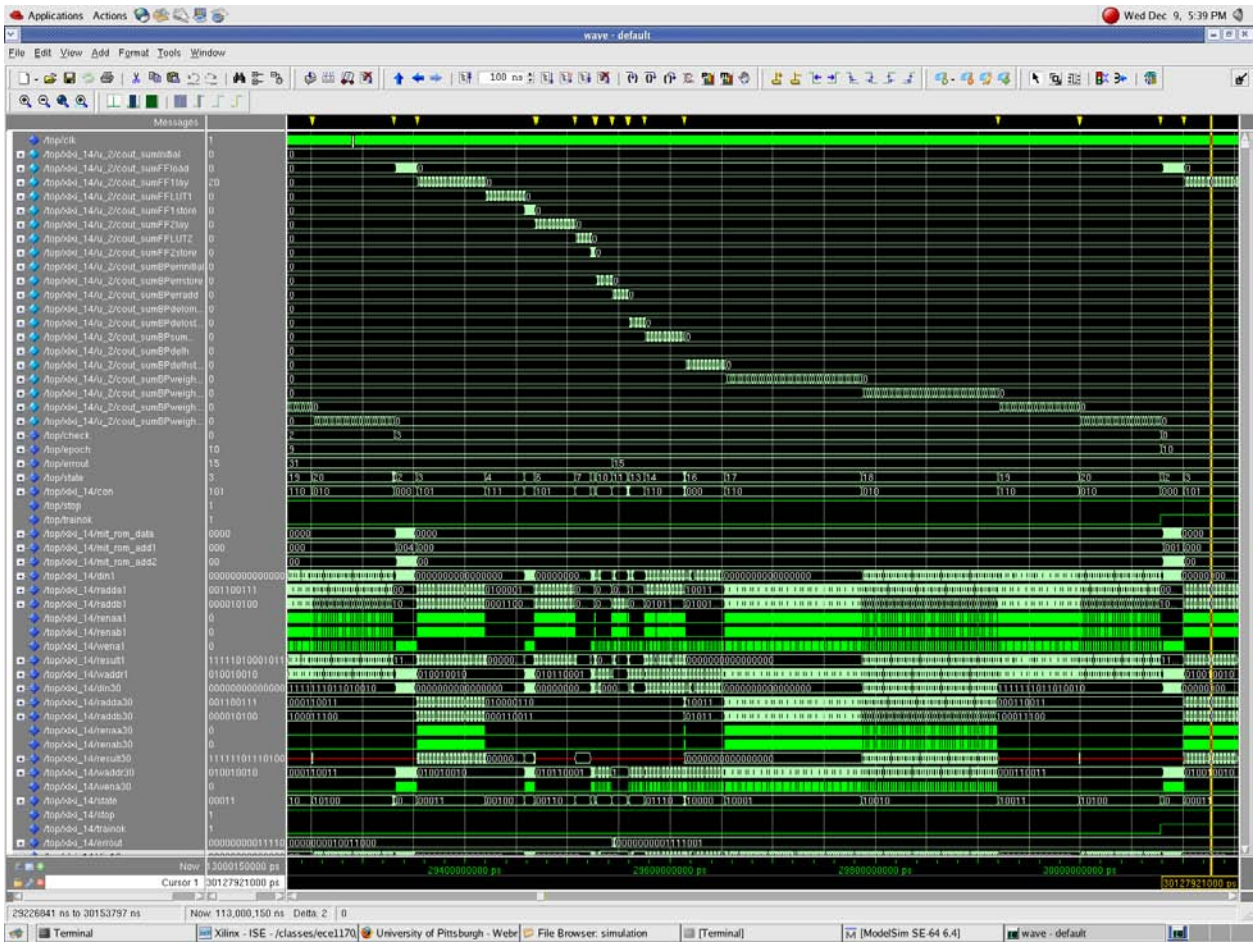
APPENDIX A

RNA POST-SYNTHESIS SIMULATION-1



APPENDIX B

RNA POST-SYNTHESIS SIMULATION-2



APPENDIX C

VERILOG SOURCE CODE OF THE GLOBAL CONTROLLER

```
module QRS_control(

        system_rst,
        clk,
        mode,

        counter,
        state,

        //from/to MIT ROM memory
        MIT_rom_data,
        MIT_rom_add1,
        MIT_rom_add2,
        MIT_rom_ren,

        //from/to Initial Rom Memory
        Initial_rom_data_1,
        Initial_rom_data_2,
        Initial_rom_data_3,
        Initial_rom_data_4,
        Initial_rom_data_5,
        Initial_rom_data_6,
        Initial_rom_data_7,
        Initial_rom_data_8,
        Initial_rom_data_9,
        Initial_rom_data_10,
        Initial_rom_data_11,
        Initial_rom_data_12,
        Initial_rom_data_13,
        Initial_rom_data_14,
        Initial_rom_data_15,
        Initial_rom_data_16,
        Initial_rom_data_17,
        Initial_rom_data_18,
        Initial_rom_data_19,
        Initial_rom_data_20,
        Initial_rom_data_21,
        Initial_rom_data_22,
        Initial_rom_data_23,
        Initial_rom_data_24,
        Initial_rom_data_25,
        Initial_rom_data_26,
        Initial_rom_data_27,
        Initial_rom_data_28,
        Initial_rom_data_29,
        Initial_rom_data_30,
```

Initial_rom_add,
Initial_rom_ren,

//from/to register and neuron

register_Radd_A_1, register_Radd_B_1, register_Ren_A_1, register_Ren_B_1, register_Wadd_1, register_Wen_1, register_Wdata_1, neuron_out_1,
register_Radd_A_2, register_Radd_B_2, register_Ren_A_2, register_Ren_B_2, register_Wadd_2, register_Wen_2, register_Wdata_2, neuron_out_2,
register_Radd_A_3, register_Radd_B_3, register_Ren_A_3, register_Ren_B_3, register_Wadd_3, register_Wen_3, register_Wdata_3, neuron_out_3,
register_Radd_A_4, register_Radd_B_4, register_Ren_A_4, register_Ren_B_4, register_Wadd_4, register_Wen_4, register_Wdata_4, neuron_out_4,
register_Radd_A_5, register_Radd_B_5, register_Ren_A_5, register_Ren_B_5, register_Wadd_5, register_Wen_5, register_Wdata_5, neuron_out_5,
register_Radd_A_6, register_Radd_B_6, register_Ren_A_6, register_Ren_B_6, register_Wadd_6, register_Wen_6, register_Wdata_6, neuron_out_6,
register_Radd_A_7, register_Radd_B_7, register_Ren_A_7, register_Ren_B_7, register_Wadd_7, register_Wen_7, register_Wdata_7, neuron_out_7,
register_Radd_A_8, register_Radd_B_8, register_Ren_A_8, register_Ren_B_8, register_Wadd_8, register_Wen_8, register_Wdata_8, neuron_out_8,
register_Radd_A_9, register_Radd_B_9, register_Ren_A_9, register_Ren_B_9, register_Wadd_9, register_Wen_9, register_Wdata_9, neuron_out_9,
register_Radd_A_10, register_Radd_B_10, register_Ren_A_10, register_Ren_B_10, register_Wadd_10, register_Wen_10, register_Wdata_10, neuron_out_10,
register_Radd_A_11, register_Radd_B_11, register_Ren_A_11, register_Ren_B_11, register_Wadd_11, register_Wen_11, register_Wdata_11, neuron_out_11,
register_Radd_A_12, register_Radd_B_12, register_Ren_A_12, register_Ren_B_12, register_Wadd_12, register_Wen_12, register_Wdata_12, neuron_out_12,
register_Radd_A_13, register_Radd_B_13, register_Ren_A_13, register_Ren_B_13, register_Wadd_13, register_Wen_13, register_Wdata_13, neuron_out_13,
register_Radd_A_14, register_Radd_B_14, register_Ren_A_14, register_Ren_B_14, register_Wadd_14, register_Wen_14, register_Wdata_14, neuron_out_14,
register_Radd_A_15, register_Radd_B_15, register_Ren_A_15, register_Ren_B_15, register_Wadd_15, register_Wen_15, register_Wdata_15, neuron_out_15,
register_Radd_A_16, register_Radd_B_16, register_Ren_A_16, register_Ren_B_16, register_Wadd_16, register_Wen_16, register_Wdata_16, neuron_out_16,
register_Radd_A_17, register_Radd_B_17, register_Ren_A_17, register_Ren_B_17, register_Wadd_17, register_Wen_17, register_Wdata_17, neuron_out_17,
register_Radd_A_18, register_Radd_B_18, register_Ren_A_18, register_Ren_B_18, register_Wadd_18, register_Wen_18, register_Wdata_18, neuron_out_18,
register_Radd_A_19, register_Radd_B_19, register_Ren_A_19, register_Ren_B_19, register_Wadd_19, register_Wen_19, register_Wdata_19, neuron_out_19,
register_Radd_A_20, register_Radd_B_20, register_Ren_A_20, register_Ren_B_20, register_Wadd_20, register_Wen_20, register_Wdata_20, neuron_out_20,
register_Radd_A_21, register_Radd_B_21, register_Ren_A_21, register_Ren_B_21, register_Wadd_21, register_Wen_21, register_Wdata_21, neuron_out_21,
register_Radd_A_22, register_Radd_B_22, register_Ren_A_22, register_Ren_B_22, register_Wadd_22, register_Wen_22, register_Wdata_22, neuron_out_22,
register_Radd_A_23, register_Radd_B_23, register_Ren_A_23, register_Ren_B_23, register_Wadd_23, register_Wen_23, register_Wdata_23, neuron_out_23,
register_Radd_A_24, register_Radd_B_24, register_Ren_A_24, register_Ren_B_24, register_Wadd_24, register_Wen_24, register_Wdata_24, neuron_out_24,
register_Radd_A_25, register_Radd_B_25, register_Ren_A_25, register_Ren_B_25, register_Wadd_25, register_Wen_25, register_Wdata_25, neuron_out_25,
register_Radd_A_26, register_Radd_B_26, register_Ren_A_26, register_Ren_B_26, register_Wadd_26, register_Wen_26, register_Wdata_26, neuron_out_26,
register_Radd_A_27, register_Radd_B_27, register_Ren_A_27, register_Ren_B_27, register_Wadd_27, register_Wen_27, register_Wdata_27, neuron_out_27,
register_Radd_A_28, register_Radd_B_28, register_Ren_A_28, register_Ren_B_28, register_Wadd_28, register_Wen_28, register_Wdata_28, neuron_out_28,
register_Radd_A_29, register_Radd_B_29, register_Ren_A_29, register_Ren_B_29, register_Wadd_29, register_Wen_29, register_Wdata_29, neuron_out_29,
register_Radd_A_30, register_Radd_B_30, register_Ren_A_30, register_Ren_B_30, register_Wadd_30, register_Wen_30, register_Wdata_30, neuron_out_30,

//to neuron specific the computation and begin signal

```
BP_code, start,acclear,LUTmux,stop,trainOK,


//for test
cout_cntInitial,
cout_cntFFload,
cout_cntFF1lay,
cout_cntFFLUT1,
cout_cntFF1store,
cout_cntFF2lay,
cout_cntFFLUT2,
cout_cntFF2store,
cout_cntBPerrinitial,
cout_cntBPerrstore,
cout_cntBPerradd,
cout_cntBPdeloinitial,
cout_cntBPdelostore,
cout_cntBPsumdow,
cout_cntBPdelh,
cout_cntBPdelhstore,
cout_cntBPweightih,
cout_cntBPweightihstore,
cout_cntBPweightho,
cout_cntBPweighthostore,
cout_cntCheck,

cout_sumInitial,
cout_sumFFload,
cout_sumFF1lay,
cout_sumFFLUT1,
cout_sumFF1store,
cout_sumFF2lay,
cout_sumFFLUT2,
cout_sumFF2store,
cout_sumBPerrinitial,
cout_sumBPerrstore,
cout_sumBPerradd,
cout_sumBPdeloinitial,
cout_sumBPdelostore,
cout_sumBPsumdow,
cout_sumBPdelh,
cout_sumBPdelhstore,
cout_sumBPweightih,
cout_sumBPweightihstore,
cout_sumBPweightho,
cout_sumBPweighthostore,

cout_sumCheck,
cout_sumEPOC,
errout
);

parameter maxcounter                                    = 22795;
parameter maxcounter2                                   = 8938;
parameter cout_sumInitialbegin          = 2;
parameter cout_sumInitialend                    = 4098;     //8*(512)
parameter cout_sumFFloadbegin                   = 4098;
parameter cout_sumFFloadend                     = 4602;     //8*(51+12)
parameter cout_sumFF1laybegin                   = 4602;
parameter cout_sumFF1layend                     = 6266;     //32*52
parameter cout_sumFFLUT1begin                           = 6266;
parameter cout_sumFFLUT1end                     = 7226;     //32*30
parameter cout_sumFF1storebegin                 = 7226;
parameter cout_sumFF1storeend                   = 7466;     //8*30
parameter cout_sumFF2laybegin                   = 7466;
parameter cout_sumFF2layend                     = 8458;     //32*31
parameter cout_sumFFLUT2begin                           = 8458;
parameter cout_sumFFLUT2end                     = 8842;     //32*12
parameter cout_sumFF2storebegin                 = 8842;
parameter cout_sumFF2storeend                   = 8938;     //8*12
```

```
parameter cout_sumBPerrinitialbegin          = 8938;
parameter cout_sumBPerrinitialend            = 8970;    //32*1
parameter cout_sumBPerrstorebegin            = 8970;
parameter cout_sumBPerrstoreend                         = 9354;    //32*12
parameter cout_sumBPerraddbegin                         = 9354;
parameter cout_sumBPerraddend                           = 9738;    //32*12
parameter cout_sumBPdeloinitialbegin= 9738;
parameter cout_sumBPdeloinitialend           = 9770;    //32*1
parameter cout_sumBPdelostorebegin           = 9770;
parameter cout_sumBPdelostoreend                        = 10154;   //32*12
parameter cout_sumBPsumdowbegin                         = 10154;
parameter cout_sumBPsumdowend                                   = 11114;   //32*30
parameter cout_sumBPdelhbegin                           = 11114;
parameter cout_sumBPdelhend                             = 11146;   //32*1
parameter cout_sumBPdelhstorebegin           = 11146;
parameter cout_sumBPdelhstoreend                        = 12106;   //32*30
parameter cout_sumBPweightihbegin                       = 12106;
parameter cout_sumBPweightihend                                 = 15434;   //64*52
parameter cout_sumBPweightihstorebegin = 15434;
parameter cout_sumBPweightihstoreend         = 18762;   //64*52
parameter cout_sumBPweighthobegin                       = 18762;
parameter cout_sumBPweighthoend                                 = 20746;   //64*31
parameter cout_sumBPweighthostorebegin   = 20746;
parameter cout_sumBPweighthostoreend         = 22730;   //64*31
parameter cout_sumCheckbegin                                    = 22730;
parameter cout_sumCheckend                                              = 22794;   //64*1

parameter bitwidth                = 16;
parameter REGaddwidth             = 9;
parameter ROMaddinweight= 0;
parameter ROMaddECG                          = 84;

parameter REGaddinweight= 0;
parameter REGaddinweight2=52;
parameter REGaddECG                          = 84; //83 for 1
parameter REGtarget               = 135;
parameter REGaddout1                         = 148; //147 for 1
parameter REGaddout2                         = 178;
parameter REGerrorcheck           = 190;
parameter REGdelo                            = 191;
parameter REGdelh                            = 203;
parameter REGdelweightih= 233;
parameter REGdelweightho= 285;
parameter REGeta                             = 316;
parameter REGalpha                = 317;
parameter REGerror                = 318;
parameter ERROR                              = 128;
parameter EPOC                                          = 1000;



input                             system_rst;
input                             clk;
input                             mode;
input[15:0]               MIT_rom_data;

output[10:0]              MIT_rom_add1; //used as 1024 complex total
output[6:0]               MIT_rom_add2; //used as7930 63 data, including 51 inputs and 12 target
output                            MIT_rom_ren;
output[2:0]               BP_code;
output                            start;
output                            acclear;
output[4:0]               LUTmux;
output                            stop;
output                            trainOK;

output[4:0]               state;
output[19:0]              counter;
```

```
//ports from and to register and neuron
input[bitwidth-1:0] neuron_out_1;
input[bitwidth-1:0] neuron_out_2;
input[bitwidth-1:0] neuron_out_3;
input[bitwidth-1:0] neuron_out_4;
input[bitwidth-1:0] neuron_out_5;
input[bitwidth-1:0] neuron_out_6;
input[bitwidth-1:0] neuron_out_7;
input[bitwidth-1:0] neuron_out_8;
input[bitwidth-1:0] neuron_out_9;
input[bitwidth-1:0] neuron_out_10;
input[bitwidth-1:0] neuron_out_11;
input[bitwidth-1:0] neuron_out_12;
input[bitwidth-1:0] neuron_out_13;
input[bitwidth-1:0] neuron_out_14;
input[bitwidth-1:0] neuron_out_15;
input[bitwidth-1:0] neuron_out_16;
input[bitwidth-1:0] neuron_out_17;
input[bitwidth-1:0] neuron_out_18;
input[bitwidth-1:0] neuron_out_19;
input[bitwidth-1:0] neuron_out_20;
input[bitwidth-1:0] neuron_out_21;
input[bitwidth-1:0] neuron_out_22;
input[bitwidth-1:0] neuron_out_23;
input[bitwidth-1:0] neuron_out_24;
input[bitwidth-1:0] neuron_out_25;
input[bitwidth-1:0] neuron_out_26;
input[bitwidth-1:0] neuron_out_27;
input[bitwidth-1:0] neuron_out_28;
input[bitwidth-1:0] neuron_out_29;
input[bitwidth-1:0] neuron_out_30;

output[REGaddwidth-1:0] register_Radd_A_1, register_Radd_B_1, register_Wadd_1;
output register_Ren_A_1, register_Ren_B_1,  register_Wen_1;
output[bitwidth-1:0] register_Wdata_1;
output[REGaddwidth-1:0] register_Radd_A_2, register_Radd_B_2, register_Wadd_2;
output register_Ren_A_2, register_Ren_B_2,  register_Wen_2;
output[bitwidth-1:0] register_Wdata_2;
output[REGaddwidth-1:0] register_Radd_A_3, register_Radd_B_3, register_Wadd_3;
output register_Ren_A_3, register_Ren_B_3,  register_Wen_3;
output[bitwidth-1:0] register_Wdata_3;
output[REGaddwidth-1:0] register_Radd_A_4, register_Radd_B_4, register_Wadd_4;
output register_Ren_A_4, register_Ren_B_4,  register_Wen_4;
output[bitwidth-1:0] register_Wdata_4;
output[REGaddwidth-1:0] register_Radd_A_5, register_Radd_B_5, register_Wadd_5;
output register_Ren_A_5, register_Ren_B_5,  register_Wen_5;
output[bitwidth-1:0] register_Wdata_5;
output[REGaddwidth-1:0] register_Radd_A_6, register_Radd_B_6, register_Wadd_6;
output register_Ren_A_6, register_Ren_B_6,  register_Wen_6;
output[bitwidth-1:0] register_Wdata_6;
output[REGaddwidth-1:0] register_Radd_A_7, register_Radd_B_7, register_Wadd_7;
output register_Ren_A_7, register_Ren_B_7,  register_Wen_7;
output[bitwidth-1:0] register_Wdata_7;
output[REGaddwidth-1:0] register_Radd_A_8, register_Radd_B_8, register_Wadd_8;
output register_Ren_A_8, register_Ren_B_8,  register_Wen_8;
output[bitwidth-1:0] register_Wdata_8;
output[REGaddwidth-1:0] register_Radd_A_9, register_Radd_B_9, register_Wadd_9;
output register_Ren_A_9, register_Ren_B_9,  register_Wen_9;
output[bitwidth-1:0] register_Wdata_9;
output[REGaddwidth-1:0] register_Radd_A_10, register_Radd_B_10, register_Wadd_10;
output register_Ren_A_10, register_Ren_B_10,  register_Wen_10;
output[bitwidth-1:0] register_Wdata_10;
output[REGaddwidth-1:0] register_Radd_A_11, register_Radd_B_11, register_Wadd_11;
output register_Ren_A_11, register_Ren_B_11,  register_Wen_11;
output[bitwidth-1:0] register_Wdata_11;
output[REGaddwidth-1:0] register_Radd_A_12, register_Radd_B_12, register_Wadd_12;
output register_Ren_A_12, register_Ren_B_12,  register_Wen_12;
output[bitwidth-1:0] register_Wdata_12;
output[REGaddwidth-1:0] register_Radd_A_13, register_Radd_B_13, register_Wadd_13;
output register_Ren_A_13, register_Ren_B_13,  register_Wen_13;
```

```
output[bitwidth-1:0] register_Wdata_13;
output[REGaddwidth-1:0] register_Radd_A_14, register_Radd_B_14, register_Wadd_14;
output register_Ren_A_14, register_Ren_B_14,  register_Wen_14;
output[bitwidth-1:0] register_Wdata_14;
output[REGaddwidth-1:0] register_Radd_A_15, register_Radd_B_15, register_Wadd_15;
output register_Ren_A_15, register_Ren_B_15,  register_Wen_15;
output[bitwidth-1:0] register_Wdata_15;
output[REGaddwidth-1:0] register_Radd_A_16, register_Radd_B_16, register_Wadd_16;
output register_Ren_A_16, register_Ren_B_16,  register_Wen_16;
output[bitwidth-1:0] register_Wdata_16;
output[REGaddwidth-1:0] register_Radd_A_17, register_Radd_B_17, register_Wadd_17;
output register_Ren_A_17, register_Ren_B_17,  register_Wen_17;
output[bitwidth-1:0] register_Wdata_17;
output[REGaddwidth-1:0] register_Radd_A_18, register_Radd_B_18, register_Wadd_18;
output register_Ren_A_18, register_Ren_B_18,  register_Wen_18;
output[bitwidth-1:0] register_Wdata_18;
output[REGaddwidth-1:0] register_Radd_A_19, register_Radd_B_19, register_Wadd_19;
output register_Ren_A_19, register_Ren_B_19,  register_Wen_19;
output[bitwidth-1:0] register_Wdata_19;
output[REGaddwidth-1:0] register_Radd_A_20, register_Radd_B_20, register_Wadd_20;
output register_Ren_A_20, register_Ren_B_20,  register_Wen_20;
output[bitwidth-1:0] register_Wdata_20;
output[REGaddwidth-1:0] register_Radd_A_21, register_Radd_B_21, register_Wadd_21;
output register_Ren_A_21, register_Ren_B_21,  register_Wen_21;
output[bitwidth-1:0] register_Wdata_21;
output[REGaddwidth-1:0] register_Radd_A_22, register_Radd_B_22, register_Wadd_22;
output register_Ren_A_22, register_Ren_B_22,  register_Wen_22;
output[bitwidth-1:0] register_Wdata_22;
output[REGaddwidth-1:0] register_Radd_A_23, register_Radd_B_23, register_Wadd_23;
output register_Ren_A_23, register_Ren_B_23,  register_Wen_23;
output[bitwidth-1:0] register_Wdata_23;
output[REGaddwidth-1:0] register_Radd_A_24, register_Radd_B_24, register_Wadd_24;
output register_Ren_A_24, register_Ren_B_24,  register_Wen_24;
output[bitwidth-1:0] register_Wdata_24;
output[REGaddwidth-1:0] register_Radd_A_25, register_Radd_B_25, register_Wadd_25;
output register_Ren_A_25, register_Ren_B_25,  register_Wen_25;
output[bitwidth-1:0] register_Wdata_25;
output[REGaddwidth-1:0] register_Radd_A_26, register_Radd_B_26, register_Wadd_26;
output register_Ren_A_26, register_Ren_B_26,  register_Wen_26;
output[bitwidth-1:0] register_Wdata_26;
output[REGaddwidth-1:0] register_Radd_A_27, register_Radd_B_27, register_Wadd_27;
output register_Ren_A_27, register_Ren_B_27,  register_Wen_27;
output[bitwidth-1:0] register_Wdata_27;
output[REGaddwidth-1:0] register_Radd_A_28, register_Radd_B_28, register_Wadd_28;
output register_Ren_A_28, register_Ren_B_28,  register_Wen_28;
output[bitwidth-1:0] register_Wdata_28;
output[REGaddwidth-1:0] register_Radd_A_29, register_Radd_B_29, register_Wadd_29;
output register_Ren_A_29, register_Ren_B_29,  register_Wen_29;
output[bitwidth-1:0] register_Wdata_29;
output[REGaddwidth-1:0] register_Radd_A_30, register_Radd_B_30, register_Wadd_30;
output register_Ren_A_30, register_Ren_B_30,  register_Wen_30;
output[bitwidth-1:0] register_Wdata_30;

output[bitwidth-1:0] errout;

input[bitwidth-1:0] Initial_rom_data_1;
input[bitwidth-1:0] Initial_rom_data_2;
input[bitwidth-1:0] Initial_rom_data_3;
input[bitwidth-1:0] Initial_rom_data_4;
input[bitwidth-1:0] Initial_rom_data_5;
input[bitwidth-1:0] Initial_rom_data_6;
input[bitwidth-1:0] Initial_rom_data_7;
input[bitwidth-1:0] Initial_rom_data_8;
input[bitwidth-1:0] Initial_rom_data_9;
input[bitwidth-1:0] Initial_rom_data_10;
input[bitwidth-1:0] Initial_rom_data_11;
input[bitwidth-1:0] Initial_rom_data_12;
input[bitwidth-1:0] Initial_rom_data_13;
input[bitwidth-1:0] Initial_rom_data_14;
input[bitwidth-1:0] Initial_rom_data_15;
```

```
input[bitwidth-1:0] Initial_rom_data_16;
input[bitwidth-1:0] Initial_rom_data_17;
input[bitwidth-1:0] Initial_rom_data_18;
input[bitwidth-1:0] Initial_rom_data_19;
input[bitwidth-1:0] Initial_rom_data_20;
input[bitwidth-1:0] Initial_rom_data_21;
input[bitwidth-1:0] Initial_rom_data_22;
input[bitwidth-1:0] Initial_rom_data_23;
input[bitwidth-1:0] Initial_rom_data_24;
input[bitwidth-1:0] Initial_rom_data_25;
input[bitwidth-1:0] Initial_rom_data_26;
input[bitwidth-1:0] Initial_rom_data_27;
input[bitwidth-1:0] Initial_rom_data_28;
input[bitwidth-1:0] Initial_rom_data_29;
input[bitwidth-1:0] Initial_rom_data_30;

output[8:0]             Initial_rom_add;
output                          Initial_rom_ren;


//for test
output[2:0]cout_cntInitial;
output[2:0] cout_cntFFload;
output[4:0]cout_cntFF1lay;
output[4:0]cout_cntFFLUT1;
output[2:0]cout_cntFF1store;
output[4:0]cout_cntFF2lay;
output[4:0] cout_cntFFLUT2;
output[2:0]cout_cntFF2store;
output[4:0]cout_cntBPerrinitial;
output[4:0] cout_cntBPerrstore;
output[4:0] cout_cntBPerradd;
output[4:0] cout_cntBPdeloinitial;
output[4:0] cout_cntBPdelostore;
output[4:0] cout_cntBPsumdow;
output[4:0] cout_cntBPdelh;
output[4:0] cout_cntBPdelhstore;
output[5:0] cout_cntBPweightih;
output[5:0] cout_cntBPweightihstore;
output[5:0] cout_cntBPweightho;
output[5:0] cout_cntBPweighthostore;
output[5:0] cout_cntCheck;

output[8:0] cout_sumInitial;
output[6:0] cout_sumFFload;
output[6:0] cout_sumFF1lay;
output[6:0]cout_sumFFLUT1;
output[6:0] cout_sumFF1store;
output[6:0] cout_sumFF2lay;
output[6:0]cout_sumFFLUT2;
output[6:0] cout_sumFF2store;
output[6:0]cout_sumBPerrinitial;
output[6:0] cout_sumBPerrstore;
output[6:0] cout_sumBPerradd;
output[6:0] cout_sumBPdeloinitial;
output[6:0] cout_sumBPdelostore;
output[6:0] cout_sumBPsumdow;
output[6:0] cout_sumBPdelh;
output[6:0] cout_sumBPdelhstore;
output[6:0] cout_sumBPweightih;
output[6:0] cout_sumBPweightihstore;
output[6:0] cout_sumBPweightho;
output[6:0] cout_sumBPweighthostore;
output[1:0]  cout_sumCheck;
output[11:0] cout_sumEPOC;

reg[4:0]                state;
reg[19:0]               counter;
reg[bitwidth-1:0] neuron_out_temp[30:0];
```

```
reg[REGaddwidth-1:0] register_Radd_A_1, register_Radd_B_1, register_Wadd_1;
reg register_Ren_A_1, register_Ren_B_1,  register_Wen_1;
reg[bitwidth-1:0] register_Wdata_1, neuron_out_temp_1;
reg[REGaddwidth-1:0] register_Radd_A_2, register_Radd_B_2, register_Wadd_2;
reg register_Ren_A_2, register_Ren_B_2,  register_Wen_2;
reg[bitwidth-1:0] register_Wdata_2, neuron_out_temp_2;
reg[REGaddwidth-1:0] register_Radd_A_3, register_Radd_B_3, register_Wadd_3;
reg register_Ren_A_3, register_Ren_B_3,  register_Wen_3;
reg[bitwidth-1:0] register_Wdata_3, neuron_out_temp_3;
reg[REGaddwidth-1:0] register_Radd_A_4, register_Radd_B_4, register_Wadd_4;
reg register_Ren_A_4, register_Ren_B_4,  register_Wen_4;
reg[bitwidth-1:0] register_Wdata_4, neuron_out_temp_4;
reg[REGaddwidth-1:0] register_Radd_A_5, register_Radd_B_5, register_Wadd_5;
reg register_Ren_A_5, register_Ren_B_5,  register_Wen_5;
reg[bitwidth-1:0] register_Wdata_5, neuron_out_temp_5;
reg[REGaddwidth-1:0] register_Radd_A_6, register_Radd_B_6, register_Wadd_6;
reg register_Ren_A_6, register_Ren_B_6,  register_Wen_6;
reg[bitwidth-1:0] register_Wdata_6, neuron_out_temp_6;
reg[REGaddwidth-1:0] register_Radd_A_7, register_Radd_B_7, register_Wadd_7;
reg register_Ren_A_7, register_Ren_B_7,  register_Wen_7;
reg[bitwidth-1:0] register_Wdata_7, neuron_out_temp_7;
reg[REGaddwidth-1:0] register_Radd_A_8, register_Radd_B_8, register_Wadd_8;
reg register_Ren_A_8, register_Ren_B_8,  register_Wen_8;
reg[bitwidth-1:0] register_Wdata_8, neuron_out_temp_8;
reg[REGaddwidth-1:0] register_Radd_A_9, register_Radd_B_9, register_Wadd_9;
reg register_Ren_A_9, register_Ren_B_9,  register_Wen_9;
reg[bitwidth-1:0] register_Wdata_9, neuron_out_temp_9;
reg[REGaddwidth-1:0] register_Radd_A_10, register_Radd_B_10, register_Wadd_10;
reg register_Ren_A_10, register_Ren_B_10,  register_Wen_10;
reg[bitwidth-1:0] register_Wdata_10, neuron_out_temp_10;
reg[REGaddwidth-1:0] register_Radd_A_11, register_Radd_B_11, register_Wadd_11;
reg register_Ren_A_11, register_Ren_B_11,  register_Wen_11;
reg[bitwidth-1:0] register_Wdata_11, neuron_out_temp_11;
reg[REGaddwidth-1:0] register_Radd_A_12, register_Radd_B_12, register_Wadd_12;
reg register_Ren_A_12, register_Ren_B_12,  register_Wen_12;
reg[bitwidth-1:0] register_Wdata_12, neuron_out_temp_12;
reg[REGaddwidth-1:0] register_Radd_A_13, register_Radd_B_13, register_Wadd_13;
reg register_Ren_A_13, register_Ren_B_13,  register_Wen_13;
reg[bitwidth-1:0] register_Wdata_13, neuron_out_temp_13;
reg[REGaddwidth-1:0] register_Radd_A_14, register_Radd_B_14, register_Wadd_14;
reg register_Ren_A_14, register_Ren_B_14,  register_Wen_14;
reg[bitwidth-1:0] register_Wdata_14, neuron_out_temp_14;
reg[REGaddwidth-1:0] register_Radd_A_15, register_Radd_B_15, register_Wadd_15;
reg register_Ren_A_15, register_Ren_B_15,  register_Wen_15;
reg[bitwidth-1:0] register_Wdata_15, neuron_out_temp_15;
reg[REGaddwidth-1:0] register_Radd_A_16, register_Radd_B_16, register_Wadd_16;
reg register_Ren_A_16, register_Ren_B_16,  register_Wen_16;
reg[bitwidth-1:0] register_Wdata_16, neuron_out_temp_16;
reg[REGaddwidth-1:0] register_Radd_A_17, register_Radd_B_17, register_Wadd_17;
reg register_Ren_A_17, register_Ren_B_17,  register_Wen_17;
reg[bitwidth-1:0] register_Wdata_17, neuron_out_temp_17;
reg[REGaddwidth-1:0] register_Radd_A_18, register_Radd_B_18, register_Wadd_18;
reg register_Ren_A_18, register_Ren_B_18,  register_Wen_18;
reg[bitwidth-1:0] register_Wdata_18, neuron_out_temp_18;
reg[REGaddwidth-1:0] register_Radd_A_19, register_Radd_B_19, register_Wadd_19;
reg register_Ren_A_19, register_Ren_B_19,  register_Wen_19;
reg[bitwidth-1:0] register_Wdata_19, neuron_out_temp_19;
reg[REGaddwidth-1:0] register_Radd_A_20, register_Radd_B_20, register_Wadd_20;
reg register_Ren_A_20, register_Ren_B_20,  register_Wen_20;
reg[bitwidth-1:0] register_Wdata_20, neuron_out_temp_20;
reg[REGaddwidth-1:0] register_Radd_A_21, register_Radd_B_21, register_Wadd_21;
reg register_Ren_A_21, register_Ren_B_21,  register_Wen_21;
reg[bitwidth-1:0] register_Wdata_21, neuron_out_temp_21;
reg[REGaddwidth-1:0] register_Radd_A_22, register_Radd_B_22, register_Wadd_22;
reg register_Ren_A_22, register_Ren_B_22,  register_Wen_22;
reg[bitwidth-1:0] register_Wdata_22, neuron_out_temp_22;
reg[REGaddwidth-1:0] register_Radd_A_23, register_Radd_B_23, register_Wadd_23;
reg register_Ren_A_23, register_Ren_B_23,  register_Wen_23;
reg[bitwidth-1:0] register_Wdata_23, neuron_out_temp_23;
reg[REGaddwidth-1:0] register_Radd_A_24, register_Radd_B_24, register_Wadd_24;
```

```
reg register_Ren_A_24, register_Ren_B_24,  register_Wen_24;
reg[bitwidth-1:0] register_Wdata_24, neuron_out_temp_24;
reg[REGaddwidth-1:0] register_Radd_A_25, register_Radd_B_25, register_Wadd_25;
reg register_Ren_A_25, register_Ren_B_25,  register_Wen_25;
reg[bitwidth-1:0] register_Wdata_25, neuron_out_temp_25;
reg[REGaddwidth-1:0] register_Radd_A_26, register_Radd_B_26, register_Wadd_26;
reg register_Ren_A_26, register_Ren_B_26,  register_Wen_26;
reg[bitwidth-1:0] register_Wdata_26, neuron_out_temp_26;
reg[REGaddwidth-1:0] register_Radd_A_27, register_Radd_B_27, register_Wadd_27;
reg register_Ren_A_27, register_Ren_B_27,  register_Wen_27;
reg[bitwidth-1:0] register_Wdata_27, neuron_out_temp_27;
reg[REGaddwidth-1:0] register_Radd_A_28, register_Radd_B_28, register_Wadd_28;
reg register_Ren_A_28, register_Ren_B_28,  register_Wen_28;
reg[bitwidth-1:0] register_Wdata_28, neuron_out_temp_28;
reg[REGaddwidth-1:0] register_Radd_A_29, register_Radd_B_29, register_Wadd_29;
reg register_Ren_A_29, register_Ren_B_29,  register_Wen_29;
reg[bitwidth-1:0] register_Wdata_29, neuron_out_temp_29;
reg[REGaddwidth-1:0] register_Radd_A_30, register_Radd_B_30, register_Wadd_30;
reg register_Ren_A_30, register_Ren_B_30,  register_Wen_30;
reg[bitwidth-1:0] register_Wdata_30, neuron_out_temp_30;

reg[8:0]     Initial_rom_add;
reg                                  Initial_rom_ren;

reg[10:0] MIT_rom_add1;
reg[6:0] MIT_rom_add2;
reg MIT_rom_ren;
reg[bitwidth-1:0] MIT_data_temp;
reg[2:0] BP_code;
reg start;
reg acclear;
reg stop;
reg[4:0]LUTmux;
reg trainOK;

//for test
reg[2:0]     cout_cntInitial;
reg[2:0] cout_cntFFload;
reg[4:0]     cout_cntFF1lay;
reg[4:0]     cout_cntFFLUT1;
reg[2:0]     cout_cntFF1store;
reg[4:0]     cout_cntFF2lay;
reg[4:0] cout_cntFFLUT2;
reg[2:0]     cout_cntFF2store;
reg[4:0]     cout_cntBPerrinitial;
reg[4:0] cout_cntBPerrstore;
reg[4:0] cout_cntBPerradd;
reg[4:0] cout_cntBPdeloinitial;
reg[4:0] cout_cntBPdelostore;
reg[4:0] cout_cntBPsumdow;
reg[4:0] cout_cntBPdelh;
reg[4:0] cout_cntBPdelhstore;
reg[5:0] cout_cntBPweightih;
reg[5:0] cout_cntBPweightihstore;
reg[5:0] cout_cntBPweightho;
reg[5:0] cout_cntBPweighthostore;
reg[5:0] cout_cntCheck;

reg[8:0] cout_sumInitial;
reg[6:0] cout_sumFFload;
reg[6:0] cout_sumFF1lay;
reg[6:0]     cout_sumFFLUT1;
reg[6:0] cout_sumFF1store;
reg[6:0] cout_sumFF2lay;
reg[6:0]     cout_sumFFLUT2;
reg[6:0] cout_sumFF2store;
reg[6:0]     cout_sumBPerrinitial;
reg[6:0] cout_sumBPerrstore;
reg[6:0] cout_sumBPerradd;
reg[6:0] cout_sumBPdeloinitial;
```

```
reg[6:0] cout_sumBPdelostore;
reg[6:0] cout_sumBPsumdow;
reg[6:0] cout_sumBPdelh;
reg[6:0] cout_sumBPdelhstore;
reg[6:0] cout_sumBPweightih;
reg[6:0] cout_sumBPweightihstore;
reg[6:0] cout_sumBPweightho;
reg[6:0] cout_sumBPweighthostore;
reg[1:0]  cout_sumCheck;
reg[11:0] cout_sumEPOC;

reg[bitwidth-1:0] errout;

always @ (posedge clk)
        begin
                if (system_rst == 1)
                        begin
                                counter <= 0;
                        end
                else if ((counter == maxcounter)||((counter == maxcounter2)&&(mode == 0)))
                        begin
                                counter <= cout_sumFFloadbegin;
                        end
                else
                        begin
                                counter <= counter + 1;
                        end
        end

always @ (posedge clk)
        begin
                if (system_rst == 1)
                        begin
                                state <= 0;
                        end
                else
                        begin
if (counter > cout_sumInitialbegin && counter < cout_sumInitialend)
        state <= 1;
else if (counter > cout_sumFFloadbegin && counter < cout_sumFFloadend)
        state <= 2;
else if (counter > cout_sumFF1laybegin && counter < cout_sumFF1layend)
        state <= 3;
else if (counter > cout_sumFFLUT1begin && counter < cout_sumFFLUT1end)
        state <= 4;
else if (counter > cout_sumFF1storebegin && counter < cout_sumFF1storeend)
        state <= 5;
else if (counter > cout_sumFF2laybegin && counter < cout_sumFF2layend)
        state <= 6;
else if (counter > cout_sumFFLUT2begin && counter < cout_sumFFLUT2end)
        state <= 7;
else if (counter > cout_sumFF2storebegin && counter < cout_sumFF2storeend)
        state <= 8;
else if (counter > cout_sumBPerrinitialbegin && counter < cout_sumBPerrinitialend)
        state <= 9;
else if (counter > cout_sumBPerrstorebegin && counter <cout_sumBPerrstoreend)
        state <= 10;
else if (counter > cout_sumBPerraddbegin && counter < cout_sumBPerraddend)
        state <= 11;
else if (counter > cout_sumBPdeloinitialbegin && counter < cout_sumBPdeloinitialend)
        state <= 12;
else if (counter > cout_sumBPdelostorebegin && counter < cout_sumBPdelostoreend)
        state <= 13;
else if (counter > cout_sumBPsumdowbegin && counter < cout_sumBPsumdowend)
        state <= 14;
else if (counter > cout_sumBPdelhbegin && counter < cout_sumBPdelhend)
        state <= 15;
else if (counter > cout_sumBPdelhstorebegin && counter < cout_sumBPdelhstoreend)
        state <= 16;
else if (counter > cout_sumBPweightihbegin && counter < cout_sumBPweightihend)
```

```
            state <= 17;
else if (counter > cout_sumBPweightihstorebegin && counter <cout_sumBPweightihstoreend)
            state <= 18;
else if (counter > cout_sumBPweighthobegin && counter < cout_sumBPweighthoend)
            state <= 19;
else if (counter > cout_sumBPweighthostorebegin && counter <cout_sumBPweighthostoreend)
            state <= 20;
else if (counter > cout_sumCheckbegin && counter <cout_sumCheckend)
            state <= 21;
else
            state <= 0;
                        end
            end

always @ (posedge clk)
            begin
                  if (system_rst == 1)
                        begin
cout_cntInitial             <= 0;
cout_cntFFload                    <= 0;
cout_cntFF1lay                    <= 0;
cout_cntFFLUT1                    <= 0;
cout_cntFF1store            <= 0;
cout_cntFF2lay                    <= 0;
cout_cntFFLUT2                    <= 0;
cout_cntFF2store            <= 0;
cout_cntBPerrinitial <= 0;
cout_cntBPerrstore   <= 0;
cout_cntBPerradd            <= 0;
cout_cntBPdeloinitial<= 0;
cout_cntBPdelostore  <= 0;
cout_cntBPsumdow     <= 0;
cout_cntBPdelh       <= 0;
cout_cntBPdelhstore  <= 0;
cout_cntBPweightih   <= 0;
cout_cntBPweightho   <= 0;
cout_cntBPweightihstore  <= 0;
cout_cntBPweighthostore  <= 0;
cout_cntCheck                           <= 0;

                        end
                  else
                        begin
                              case (state)
                                    0:          begin
cout_cntInitial           <= 0;
cout_cntFFload    <= 0;
cout_cntFF1lay    <= 0;
cout_cntFFLUT1    <= 0;
cout_cntFF1store  <= 0;
cout_cntFF2lay    <= 0;
cout_cntFFLUT2              <= 0;
cout_cntFF2store  <= 0;
cout_cntBPerrinitial <= 0;
cout_cntBPerrstore   <= 0;
cout_cntBPerradd            <= 0;
cout_cntBPdeloinitial<= 0;
cout_cntBPdelostore  <= 0;
cout_cntBPsumdow     <= 0;
cout_cntBPdelh       <= 0;
cout_cntBPdelhstore  <= 0;
cout_cntBPweightih   <= 0;
cout_cntBPweightho   <= 0;
cout_cntBPweightihstore  <= 0;
cout_cntBPweighthostore  <= 0;
cout_cntCheck                           <= 0;
                                                end
                                    1:          begin
            cout_cntInitial         <= cout_cntInitial + 1;
            cout_cntFFload    <= 0;
```

```
                cout_cntFF1lay    <= 0;
                cout_cntFFLUT1    <= 0;
                cout_cntFF1store  <= 0;
                cout_cntFF2lay    <= 0;
                cout_cntFFLUT2                  <= 0;
                cout_cntFF2store  <= 0;
                cout_cntBPerrinitial <= 0;
                cout_cntBPerrstore   <= 0;
                cout_cntBPerradd                <= 0;
                cout_cntBPdeloinitial<= 0;
                cout_cntBPdelostore  <= 0;
                cout_cntBPsumdow     <= 0;
                cout_cntBPdelh       <= 0;
                cout_cntBPdelhstore  <= 0;
                cout_cntBPweightih   <= 0;
                cout_cntBPweightho   <= 0;
                cout_cntBPweightihstore  <= 0;
                cout_cntBPweighthostore  <= 0;
                cout_cntCheck                                   <= 0;
                                                                end
                                                        2:      begin
cout_cntInitial         <= 0;
cout_cntFFload          <= cout_cntFFload + 1;
cout_cntFF1lay    <= 0;
cout_cntFFLUT1    <= 0;
cout_cntFF1store  <= 0;
cout_cntFF2lay    <= 0;
cout_cntFFLUT2                  <= 0;
cout_cntFF2store  <= 0;
cout_cntBPerrinitial <= 0;
cout_cntBPerrstore   <= 0;
cout_cntBPerradd                <= 0;
cout_cntBPdeloinitial<= 0;
cout_cntBPdelostore  <= 0;
cout_cntBPsumdow     <= 0;
cout_cntBPdelh       <= 0;
cout_cntBPdelhstore  <= 0;
cout_cntBPweightih   <= 0;
cout_cntBPweightho   <= 0;
cout_cntBPweightihstore  <= 0;
cout_cntBPweighthostore  <= 0;
cout_cntCheck                                           <= 0;
                                                                end
                                                        3:      begin
                cout_cntInitial         <= 0;
                cout_cntFFload    <= 0;
                cout_cntFF1lay          <= cout_cntFF1lay + 1;
                cout_cntFFLUT1    <= 0;
                cout_cntFF1store  <= 0;
                cout_cntFF2lay    <= 0;
                cout_cntFFLUT2                  <= 0;
                cout_cntFF2store  <= 0;
                cout_cntBPerrinitial <= 0;
                cout_cntBPerrstore   <= 0;
                cout_cntBPerradd                <= 0;
                cout_cntBPdeloinitial<= 0;
                cout_cntBPdelostore  <= 0;
                cout_cntBPsumdow     <= 0;
                cout_cntBPdelh       <= 0;
                cout_cntBPdelhstore  <= 0;
                cout_cntBPweightih   <= 0;
                cout_cntBPweightho   <= 0;
                cout_cntBPweightihstore  <= 0;
                cout_cntBPweighthostore  <= 0;
                cout_cntCheck                                   <= 0;
                                                                end
                                                        4:      begin
cout_cntInitial         <= 0;
cout_cntFFload    <= 0;
cout_cntFF1lay    <= 0;
```

```
cout_cntFFLUT1    <= cout_cntFFLUT1 + 1;
cout_cntFF1store  <= 0;
cout_cntFF2lay    <= 0;
cout_cntFFLUT2                  <= 0;
cout_cntFF2store  <= 0;
cout_cntBPerrinitial <= 0;
cout_cntBPerrstore   <= 0;
cout_cntBPerradd                <= 0;
cout_cntBPdeloinitial<= 0;
cout_cntBPdelostore  <= 0;
cout_cntBPsumdow     <= 0;
cout_cntBPdelh       <= 0;
cout_cntBPdelhstore  <= 0;
cout_cntBPweightih   <= 0;
cout_cntBPweightho   <= 0;
cout_cntBPweightihstore  <= 0;
cout_cntBPweighthostore  <= 0;
cout_cntCheck                                   <= 0;
                                                end
                                                5:              begin
cout_cntInitial             <= 0;
cout_cntFFload    <= 0;
cout_cntFF1lay    <= 0;
cout_cntFFLUT1    <= 0;
cout_cntFF1store  <= cout_cntFF1store + 1;
cout_cntFF2lay    <= 0;
cout_cntFFLUT2                  <= 0;
cout_cntFF2store  <= 0;
cout_cntBPerrinitial <= 0;
cout_cntBPerrstore   <= 0;
cout_cntBPerradd                <= 0;
cout_cntBPdeloinitial<= 0;
cout_cntBPdelostore  <= 0;
cout_cntBPsumdow     <= 0;
cout_cntBPdelh       <= 0;
cout_cntBPdelhstore  <= 0;
cout_cntBPweightih   <= 0;
cout_cntBPweightho   <= 0;
cout_cntBPweightihstore  <= 0;
cout_cntBPweighthostore  <= 0;
cout_cntCheck                                   <= 0;
                                                end
                                                6:              begin
cout_cntInitial             <= 0;
cout_cntFFload    <= 0;
cout_cntFF1lay    <= 0;
cout_cntFFLUT1    <= 0;
cout_cntFF1store  <= 0;
cout_cntFF2lay    <= cout_cntFF2lay + 1;
cout_cntFFLUT2                  <= 0;
cout_cntFF2store  <= 0;
cout_cntBPerrinitial <= 0;
cout_cntBPerrstore   <= 0;
cout_cntBPerradd                <= 0;
cout_cntBPdeloinitial<= 0;
cout_cntBPdelostore  <= 0;
cout_cntBPsumdow     <= 0;
cout_cntBPdelh       <= 0;
cout_cntBPdelhstore  <= 0;
cout_cntBPweightih   <= 0;
cout_cntBPweightho   <= 0;
cout_cntBPweightihstore  <= 0;
cout_cntBPweighthostore  <= 0;
cout_cntCheck                                   <= 0;
                                                end
                                                7:              begin
cout_cntInitial             <= 0;
cout_cntFFload    <= 0;
cout_cntFF1lay    <= 0;
cout_cntFFLUT1    <= 0;
```

```
cout_cntFF1store  <= 0;
cout_cntFF2lay    <= 0;
cout_cntFFLUT2                  <= cout_cntFFLUT2 + 1;
cout_cntFF2store  <= 0;
cout_cntBPerrinitial <= 0;
cout_cntBPerrstore   <= 0;
cout_cntBPerradd                <= 0;
cout_cntBPdeloinitial<= 0;
cout_cntBPdelostore  <= 0;
cout_cntBPsumdow     <= 0;
cout_cntBPdelh       <= 0;
cout_cntBPdelhstore  <= 0;
cout_cntBPweightih   <= 0;
cout_cntBPweightho   <= 0;
cout_cntBPweightihstore  <= 0;
cout_cntBPweighthostore  <= 0;
cout_cntCheck                                   <= 0;
                                                        end
                                                8:      begin
cout_cntInitial          <= 0;
cout_cntFFload    <= 0;
cout_cntFF1lay    <= 0;
cout_cntFFLUT1    <= 0;
cout_cntFF1store  <= 0;
cout_cntFF2lay    <= 0;
cout_cntFFLUT2                  <= 0;
cout_cntFF2store  <= cout_cntFF2store + 1;
cout_cntBPerrinitial <= 0;
cout_cntBPerrstore   <= 0;
cout_cntBPerradd                <= 0;
cout_cntBPdeloinitial<= 0;
cout_cntBPdelostore  <= 0;
cout_cntBPsumdow     <= 0;
cout_cntBPdelh       <= 0;
cout_cntBPdelhstore  <= 0;
cout_cntBPweightih   <= 0;
cout_cntBPweightho   <= 0;
cout_cntBPweightihstore  <= 0;
cout_cntBPweighthostore  <= 0;
cout_cntCheck                                   <= 0;
                                                        end
                                                9: begin
cout_cntInitial          <= 0;
cout_cntFFload    <= 0;
cout_cntFF1lay    <= 0;
cout_cntFFLUT1    <= 0;
cout_cntFF1store  <= 0;
cout_cntFF2lay    <= 0;
cout_cntFFLUT2                  <= 0;
cout_cntFF2store  <= 0;
cout_cntBPerrinitial <= cout_cntBPerrinitial + 1;
cout_cntBPerrstore   <= 0;
cout_cntBPerradd                <= 0;
cout_cntBPdeloinitial<= 0;
cout_cntBPdelostore  <= 0;
cout_cntBPsumdow     <= 0;
cout_cntBPdelh       <= 0;
cout_cntBPdelhstore  <= 0;
cout_cntBPweightih   <= 0;
cout_cntBPweightho   <= 0;
cout_cntBPweightihstore  <= 0;
cout_cntBPweighthostore  <= 0;
cout_cntCheck                                   <= 0;
                                                        end
                                                10:begin
cout_cntInitial          <= 0;
cout_cntFFload    <= 0;
cout_cntFF1lay    <= 0;
cout_cntFFLUT1    <= 0;
cout_cntFF1store  <= 0;
```

```
cout_cntFF2lay    <= 0;
cout_cntFFLUT2                  <= 0;
cout_cntFF2store  <= 0;
cout_cntBPerrinitial <= 0;
cout_cntBPerrstore   <= cout_cntBPerrstore + 1;
cout_cntBPerradd                <= 0;
cout_cntBPdeloinitial<= 0;
cout_cntBPdelostore  <= 0;
cout_cntBPsumdow     <= 0;
cout_cntBPdelh       <= 0;
cout_cntBPdelhstore  <= 0;
cout_cntBPweightih   <= 0;
cout_cntBPweightho   <= 0;
cout_cntBPweightihstore  <= 0;
cout_cntBPweighthostore  <= 0;
cout_cntCheck                                           <= 0;
                                                                end
                                                        11:begin
cout_cntInitial           <= 0;
cout_cntFFload    <= 0;
cout_cntFF1lay    <= 0;
cout_cntFFLUT1    <= 0;
cout_cntFF1store  <= 0;
cout_cntFF2lay    <= 0;
cout_cntFFLUT2                  <= 0;
cout_cntFF2store  <= 0;
cout_cntBPerrinitial <= 0;
cout_cntBPerrstore   <= 0;
cout_cntBPerradd                <= cout_cntBPerradd + 1;
cout_cntBPdeloinitial<= 0;
cout_cntBPdelostore  <= 0;
cout_cntBPsumdow     <= 0;
cout_cntBPdelh       <= 0;
cout_cntBPdelhstore  <= 0;
cout_cntBPweightih   <= 0;
cout_cntBPweightho   <= 0;
cout_cntBPweightihstore  <= 0;
cout_cntBPweighthostore  <= 0;
cout_cntCheck                                           <= 0;
                                                                end
                                                        12:begin
cout_cntInitial           <= 0;
cout_cntFFload    <= 0;
cout_cntFF1lay    <= 0;
cout_cntFFLUT1    <= 0;
cout_cntFF1store  <= 0;
cout_cntFF2lay    <= 0;
cout_cntFFLUT2                  <= 0;
cout_cntFF2store  <= 0;
cout_cntBPerrinitial <= 0;
cout_cntBPerrstore   <= 0;
cout_cntBPerradd                <= 0;
cout_cntBPdeloinitial<= cout_cntBPdeloinitial + 1;
cout_cntBPdelostore  <= 0;
cout_cntBPsumdow     <= 0;
cout_cntBPdelh       <= 0;
cout_cntBPdelhstore  <= 0;
cout_cntBPweightih   <= 0;
cout_cntBPweightho   <= 0;
cout_cntBPweightihstore  <= 0;
cout_cntBPweighthostore  <= 0;
cout_cntCheck                                           <= 0;
                                                                end
13:begin
                        cout_cntInitial           <= 0;
                        cout_cntFFload    <= 0;
                        cout_cntFF1lay    <= 0;
                        cout_cntFFLUT1    <= 0;
                        cout_cntFF1store  <= 0;
                        cout_cntFF2lay    <= 0;
```

```
                cout_cntFFLUT2                  <= 0;
                cout_cntFF2store  <= 0;
                cout_cntBPerrinitial <= 0;
                cout_cntBPerrstore   <= 0;
                cout_cntBPerradd                <= 0;
                cout_cntBPdeloinitial<= 0;
                cout_cntBPdelostore  <= cout_cntBPdelostore + 1;
                cout_cntBPsumdow     <= 0;
                cout_cntBPdelh       <= 0;
                cout_cntBPdelhstore  <= 0;
                cout_cntBPweightih   <= 0;
                cout_cntBPweightho   <= 0;
                cout_cntBPweightihstore  <= 0;
                cout_cntBPweighthostore  <= 0;
                cout_cntCheck                                   <= 0;
        end
14:begin
                cout_cntInitial          <= 0;
                cout_cntFFload    <= 0;
                cout_cntFF1lay    <= 0;
                cout_cntFFLUT1    <= 0;
                cout_cntFF1store  <= 0;
                cout_cntFF2lay    <= 0;
                cout_cntFFLUT2                  <= 0;
                cout_cntFF2store  <= 0;
                cout_cntBPerrinitial <= 0;
                cout_cntBPerrstore   <= 0;
                cout_cntBPerradd                <= 0;
                cout_cntBPdeloinitial<= 0;
                cout_cntBPdelostore  <= 0;
                cout_cntBPsumdow     <= cout_cntBPsumdow + 1;
                cout_cntBPdelh       <= 0;
                cout_cntBPdelhstore  <= 0;
                cout_cntBPweightih   <= 0;
                cout_cntBPweightho   <= 0;
                cout_cntBPweightihstore  <= 0;
                cout_cntBPweighthostore  <= 0;
                cout_cntCheck                                   <= 0;
        end
15:begin
                cout_cntInitial          <= 0;
                cout_cntFFload    <= 0;
                cout_cntFF1lay    <= 0;
                cout_cntFFLUT1    <= 0;
                cout_cntFF1store  <= 0;
                cout_cntFF2lay    <= 0;
                cout_cntFFLUT2                  <= 0;
                cout_cntFF2store  <= 0;
                cout_cntBPerrinitial <= 0;
                cout_cntBPerrstore   <= 0;
                cout_cntBPerradd                <= 0;
                cout_cntBPdeloinitial<= 0;
                cout_cntBPdelostore  <= 0;
                cout_cntBPsumdow     <= 0;
                cout_cntBPdelh       <= cout_cntBPdelh + 1;
                cout_cntBPdelhstore  <= 0;
                cout_cntBPweightih   <= 0;
                cout_cntBPweightho   <= 0;
                cout_cntBPweightihstore  <= 0;
                cout_cntBPweighthostore  <= 0;
                cout_cntCheck                                   <= 0;
        end
16:begin
                cout_cntInitial          <= 0;
                cout_cntFFload    <= 0;
                cout_cntFF1lay    <= 0;
                cout_cntFFLUT1    <= 0;
                cout_cntFF1store  <= 0;
                cout_cntFF2lay    <= 0;
                cout_cntFFLUT2                  <= 0;
```

```
                cout_cntFF2store  <= 0;
                cout_cntBPerrinitial <= 0;
                cout_cntBPerrstore   <= 0;
                cout_cntBPerradd                <= 0;
                cout_cntBPdeloinitial<= 0;
                cout_cntBPdelostore  <= 0;
                cout_cntBPsumdow     <= 0;
                cout_cntBPdelh       <= 0;
                cout_cntBPdelhstore  <= cout_cntBPdelhstore + 1;
                cout_cntBPweightih   <= 0;
                cout_cntBPweightho   <= 0;
                cout_cntBPweightihstore  <= 0;
                cout_cntBPweighthostore  <= 0;
                cout_cntCheck                                           <= 0;
        end
17:begin
                cout_cntInitial          <= 0;
                cout_cntFFload    <= 0;
                cout_cntFF1lay    <= 0;
                cout_cntFFLUT1    <= 0;
                cout_cntFF1store  <= 0;
                cout_cntFF2lay    <= 0;
                cout_cntFFLUT2                  <= 0;
                cout_cntFF2store  <= 0;
                cout_cntBPerrinitial <= 0;
                cout_cntBPerrstore   <= 0;
                cout_cntBPerradd                <= 0;
                cout_cntBPdeloinitial<= 0;
                cout_cntBPdelostore  <= 0;
                cout_cntBPsumdow     <= 0;
                cout_cntBPdelh       <= 0;
                cout_cntBPdelhstore  <= 0;
                cout_cntBPweightih   <= cout_cntBPweightih + 1;
                cout_cntBPweightho   <= 0;
                cout_cntBPweightihstore  <= 0;
                cout_cntBPweighthostore  <= 0;
                cout_cntCheck                                           <= 0;
        end
18:begin
                cout_cntInitial          <= 0;
                cout_cntFFload    <= 0;
                cout_cntFF1lay    <= 0;
                cout_cntFFLUT1    <= 0;
                cout_cntFF1store  <= 0;
                cout_cntFF2lay    <= 0;
                cout_cntFFLUT2                  <= 0;
                cout_cntFF2store  <= 0;
                cout_cntBPerrinitial <= 0;
                cout_cntBPerrstore   <= 0;
                cout_cntBPerradd                <= 0;
                cout_cntBPdeloinitial<= 0;
                cout_cntBPdelostore  <= 0;
                cout_cntBPsumdow     <= 0;
                cout_cntBPdelh       <= 0;
                cout_cntBPdelhstore  <= 0;
                cout_cntBPweightih   <= 0;
                cout_cntBPweightho   <= 0;
                cout_cntBPweightihstore  <= cout_cntBPweightihstore + 1;
                cout_cntBPweighthostore  <= 0;
                cout_cntCheck                                           <= 0;
        end
19:begin
                cout_cntInitial          <= 0;
                cout_cntFFload    <= 0;
                cout_cntFF1lay    <= 0;
                cout_cntFFLUT1    <= 0;
                cout_cntFF1store  <= 0;
                cout_cntFF2lay    <= 0;
                cout_cntFFLUT2                  <= 0;
                cout_cntFF2store  <= 0;
```

```
                cout_cntBPerrinitial <= 0;
                cout_cntBPerrstore  <= 0;
                cout_cntBPerradd                <= 0;
                cout_cntBPdeloinitial<= 0;
                cout_cntBPdelostore  <= 0;
                cout_cntBPsumdow    <= 0;
                cout_cntBPdelh      <= 0;
                cout_cntBPdelhstore  <= 0;
                cout_cntBPweightih  <= 0;
                cout_cntBPweightho  <= cout_cntBPweightho + 1;
                cout_cntBPweightihstore  <= 0;
                cout_cntBPweighthostore  <= 0;
                cout_cntCheck                                           <= 0;
        end
20:begin
                cout_cntInitial         <= 0;
                cout_cntFFload   <= 0;
                cout_cntFF1lay   <= 0;
                cout_cntFFLUT1   <= 0;
                cout_cntFF1store  <= 0;
                cout_cntFF2lay   <= 0;
                cout_cntFFLUT2                  <= 0;
                cout_cntFF2store  <= 0;
                cout_cntBPerrinitial <= 0;
                cout_cntBPerrstore  <= 0;
                cout_cntBPerradd                <= 0;
                cout_cntBPdeloinitial<= 0;
                cout_cntBPdelostore  <= 0;
                cout_cntBPsumdow    <= 0;
                cout_cntBPdelh      <= 0;
                cout_cntBPdelhstore  <= 0;
                cout_cntBPweightih  <= 0;
                cout_cntBPweightho  <= 0;
                cout_cntBPweightihstore  <= 0;
                cout_cntBPweighthostore  <= cout_cntBPweighthostore + 1;
                cout_cntCheck                                           <= 0;
        end
21:begin
                cout_cntInitial         <= 0;
                cout_cntFFload   <= 0;
                cout_cntFF1lay   <= 0;
                cout_cntFFLUT1   <= 0;
                cout_cntFF1store  <= 0;
                cout_cntFF2lay   <= 0;
                cout_cntFFLUT2                  <= 0;
                cout_cntFF2store  <= 0;
                cout_cntBPerrinitial <= 0;
                cout_cntBPerrstore  <= 0;
                cout_cntBPerradd                <= 0;
                cout_cntBPdeloinitial<= 0;
                cout_cntBPdelostore  <= 0;
                cout_cntBPsumdow    <= 0;
                cout_cntBPdelh      <= 0;
                cout_cntBPdelhstore  <= 0;
                cout_cntBPweightih  <= 0;
                cout_cntBPweightho  <= 0;
                cout_cntBPweightihstore  <= 0;
                cout_cntBPweighthostore  <= 0;
                cout_cntCheck                                           <= cout_cntCheck + 1;
        end
default:
        begin
                cout_cntInitial         <= 0;
                cout_cntFFload   <= 0;
                cout_cntFF1lay   <= 0;
                cout_cntFFLUT1   <= 0;
                cout_cntFF1store  <= 0;
                cout_cntFF2lay   <= 0;
                cout_cntFFLUT2                  <= 0;
                cout_cntFF2store  <= 0;
```

```
                cout_cntBPerrinitial <= 0;
                cout_cntBPerrstore   <= 0;
                cout_cntBPerradd                <= 0;
                cout_cntBPdeloinitial<= 0;
                cout_cntBPdelostore  <= 0;
                cout_cntBPsumdow     <= 0;
                cout_cntBPdelh       <= 0;
                cout_cntBPdelhstore  <= 0;
                cout_cntBPweightih   <= 0;
                cout_cntBPweightho   <= 0;
                cout_cntBPweightihstore  <= 0;
                cout_cntBPweighthostore  <= 0;
                cout_cntCheck                                   <= 0;
        end
                                endcase
                        end
        end

always @ (posedge clk)
        begin
if (system_rst == 1)
        begin
                cout_sumInitial                 <= 0;
                cout_sumFFload          <= 0;
                cout_sumFF1lay          <= 0;
                cout_sumFFLUT1                          <= 0;
                cout_sumFF1store                <= 0;
                cout_sumFF2lay          <= 0;
                cout_sumFFLUT2                          <= 0;
                cout_sumFF2store                <= 0;
                cout_sumBPerrinitial <= 0;
                cout_sumBPerrstore   <= 0;
                cout_sumBPerradd     <= 0;
                cout_sumBPdeloinitial<= 0;
                cout_sumBPdelostore  <= 0;
                cout_sumBPsumdow     <= 0;
                cout_sumBPdelh       <= 0;
                cout_sumBPdelhstore  <= 0;
                cout_sumBPweightih    <= 0;
                cout_sumBPweightho    <= 0;
                cout_sumBPweightihstore  <= 0;
                cout_sumBPweighthostore  <= 0;

        end
else
        begin
                if (state == 0)
                        begin
                cout_sumInitial                 <= 0;
                cout_sumFFload          <= 0;
                cout_sumFF1lay          <= 0;
                cout_sumFFLUT1                          <= 0;
                cout_sumFF1store                <= 0;
                cout_sumFF2lay          <= 0;
                cout_sumFFLUT2                          <= 0;
                cout_sumFF2store                <= 0;
                cout_sumBPerrinitial <= 0;
                cout_sumBPerrstore   <= 0;
                cout_sumBPerradd     <= 0;
                cout_sumBPdeloinitial<= 0;
                cout_sumBPdelostore  <= 0;
                cout_sumBPsumdow     <= 0;
                cout_sumBPdelh       <= 0;
                cout_sumBPdelhstore  <= 0;
                cout_sumBPweightih   <= 0;
                cout_sumBPweightho   <= 0;
                cout_sumBPweightihstore  <= 0;
                cout_sumBPweighthostore  <= 0;
        end
else if (cout_cntInitial == 7)
```

```
		begin
			cout_sumInitial			<= cout_sumInitial + 1;
			cout_sumFFload		<= 0;
			cout_sumFF1lay		<= 0;
			cout_sumFFLUT1				<= 0;
			cout_sumFF1store		<= 0;
			cout_sumFF2lay		<= 0;
			cout_sumFFLUT2				<= 0;
			cout_sumFF2store		<= 0;
			cout_sumBPerrinitial <= 0;
			cout_sumBPerrstore   <= 0;
			cout_sumBPerradd     <= 0;
			cout_sumBPdeloinitial<= 0;
			cout_sumBPdelostore  <= 0;
			cout_sumBPsumdow     <= 0;
			cout_sumBPdelh       <= 0;
			cout_sumBPdelhstore  <= 0;
			cout_sumBPweightih   <= 0;
			cout_sumBPweightho   <= 0;
			cout_sumBPweightihstore  <= 0;
			cout_sumBPweighthostore  <= 0;
		end
	else if (cout_cntFFload == 7)
		begin
			cout_sumInitial			<= 0;
			cout_sumFFload		<= cout_sumFFload + 1;
			cout_sumFF1lay		<= 0;
			cout_sumFFLUT1				<= 0;
			cout_sumFF1store		<= 0;
			cout_sumFF2lay		<= 0;
			cout_sumFFLUT2				<= 0;
			cout_sumFF2store		<= 0;
			cout_sumBPerrinitial <= 0;
			cout_sumBPerrstore   <= 0;
			cout_sumBPerradd     <= 0;
			cout_sumBPdeloinitial<= 0;
			cout_sumBPdelostore  <= 0;
			cout_sumBPsumdow     <= 0;
			cout_sumBPdelh       <= 0;
			cout_sumBPdelhstore  <= 0;
			cout_sumBPweightih   <= 0;
			cout_sumBPweightho   <= 0;
			cout_sumBPweightihstore  <= 0;
			cout_sumBPweighthostore  <= 0;
		end
	else if (cout_cntFF1lay == 31)
		begin
			cout_sumInitial			<= 0;
			cout_sumFFload		<= 0;
			cout_sumFF1lay		<= cout_sumFF1lay + 1;
			cout_sumFFLUT1				<= 0;
			cout_sumFF1store		<= 0;
			cout_sumFF2lay		<= 0;
			cout_sumFFLUT2				<= 0;
			cout_sumFF2store		<= 0;
			cout_sumBPerrinitial <= 0;
			cout_sumBPerrstore   <= 0;
			cout_sumBPerradd     <= 0;
			cout_sumBPdeloinitial<= 0;
			cout_sumBPdelostore  <= 0;
			cout_sumBPsumdow     <= 0;
			cout_sumBPdelh       <= 0;
			cout_sumBPdelhstore  <= 0;
			cout_sumBPweightih   <= 0;
			cout_sumBPweightho   <= 0;
			cout_sumBPweightihstore  <= 0;
			cout_sumBPweighthostore  <= 0;
		end
	else if (cout_cntFFLUT1 == 31)
		begin
```

```
                cout_sumInitial                 <= 0;
                cout_sumFFload      <= 0;
                cout_sumFF1lay      <= 0;
                cout_sumFFLUT1                  <= cout_sumFFLUT1 + 1;
                cout_sumFF1store                <= 0;
                cout_sumFF2lay      <= 0;
                cout_sumFFLUT2                  <= 0;
                cout_sumFF2store                <= 0;
                cout_sumBPerrinitial <= 0;
                cout_sumBPerrstore   <= 0;
                cout_sumBPerradd    <= 0;
                cout_sumBPdeloinitial<= 0;
                cout_sumBPdelostore  <= 0;
                cout_sumBPsumdow    <= 0;
                cout_sumBPdelh      <= 0;
                cout_sumBPdelhstore  <= 0;
                cout_sumBPweightih   <= 0;
                cout_sumBPweightho   <= 0;
                cout_sumBPweightihstore  <= 0;
                cout_sumBPweighthostore  <= 0;
        end
else if (cout_cntFF1store == 7)
        begin
                cout_sumInitial                 <= 0;
                cout_sumFFload      <= 0;
                cout_sumFF1lay      <= 0;
                cout_sumFFLUT1                  <= 0;
                cout_sumFF1store                <= cout_sumFF1store + 1;
                cout_sumFF2lay      <= 0;
                cout_sumFFLUT2                  <= 0;
                cout_sumFF2store                <= 0;
                cout_sumBPerrinitial <= 0;
                cout_sumBPerrstore   <= 0;
                cout_sumBPerradd    <= 0;
                cout_sumBPdeloinitial<= 0;
                cout_sumBPdelostore  <= 0;
                cout_sumBPsumdow    <= 0;
                cout_sumBPdelh      <= 0;
                cout_sumBPdelhstore  <= 0;
                cout_sumBPweightih   <= 0;
                cout_sumBPweightho   <= 0;
                cout_sumBPweightihstore  <= 0;
                cout_sumBPweighthostore  <= 0;
        end
else if (cout_cntFF2lay == 31)
        begin
                cout_sumInitial                 <= 0;
                cout_sumFFload      <= 0;
                cout_sumFF1lay      <= 0;
                cout_sumFFLUT1                  <= 0;
                cout_sumFF1store                <= 0;
                cout_sumFF2lay      <= cout_sumFF2lay + 1;
                cout_sumFFLUT2                  <= 0;
                cout_sumFF2store                <= 0;
                cout_sumBPerrinitial <= 0;
                cout_sumBPerrstore   <= 0;
                cout_sumBPerradd    <= 0;
                cout_sumBPdeloinitial<= 0;
                cout_sumBPdelostore  <= 0;
                cout_sumBPsumdow    <= 0;
                cout_sumBPdelh      <= 0;
                cout_sumBPdelhstore  <= 0;
                cout_sumBPweightih   <= 0;
                cout_sumBPweightho   <= 0;
                cout_sumBPweightihstore  <= 0;
                cout_sumBPweighthostore  <= 0;
        end
else if (cout_cntFFLUT2 == 31)
        begin
                cout_sumInitial                 <= 0;
```

```
                cout_sumFFload      <= 0;
                cout_sumFF1lay      <= 0;
                cout_sumFFLUT1                  <= 0;
                cout_sumFF1store            <= 0;
                cout_sumFF2lay      <= 0;
                cout_sumFFLUT2                  <= cout_sumFFLUT2 + 1;
                cout_sumFF2store            <= 0;
                cout_sumBPerrinitial <= 0;
                cout_sumBPerrstore  <= 0;
                cout_sumBPerradd    <= 0;
                cout_sumBPdeloinitial<= 0;
                cout_sumBPdelostore  <= 0;
                cout_sumBPsumdow    <= 0;
                cout_sumBPdelh      <= 0;
                cout_sumBPdelhstore  <= 0;
                cout_sumBPweightih  <= 0;
                cout_sumBPweightho  <= 0;
                cout_sumBPweightihstore  <= 0;
                cout_sumBPweighthostore  <= 0;
        end
else if (cout_cntFF2store == 7)
        begin
                cout_sumInitial             <= 0;
                cout_sumFFload      <= 0;
                cout_sumFF1lay      <= 0;
                cout_sumFFLUT1                  <= 0;
                cout_sumFF1store            <= 0;
                cout_sumFF2lay      <= 0;
                cout_sumFFLUT2                  <= 0;
                cout_sumFF2store            <= cout_sumFF2store + 1;
                cout_sumBPerrinitial <= 0;
                cout_sumBPerrstore  <= 0;
                cout_sumBPerradd    <= 0;
                cout_sumBPdeloinitial<= 0;
                cout_sumBPdelostore  <= 0;
                cout_sumBPsumdow    <= 0;
                cout_sumBPdelh      <= 0;
                cout_sumBPdelhstore  <= 0;
                cout_sumBPweightih  <= 0;
                cout_sumBPweightho  <= 0;
                cout_sumBPweightihstore  <= 0;
                cout_sumBPweighthostore  <= 0;
        end
else if (cout_cntBPerrinitial == 31)
        begin
                cout_sumInitial             <= 0;
                cout_sumFFload      <= 0;
                cout_sumFF1lay      <= 0;
                cout_sumFFLUT1                  <= 0;
                cout_sumFF1store            <= 0;
                cout_sumFF2lay      <= 0;
                cout_sumFFLUT2                  <= 0;
                cout_sumFF2store            <= 0;
                cout_sumBPerrinitial <= cout_sumBPerrinitial + 1;
                cout_sumBPerrstore  <= 0;
                cout_sumBPerradd    <= 0;
                cout_sumBPdeloinitial<= 0;
                cout_sumBPdelostore  <= 0;
                cout_sumBPsumdow    <= 0;
                cout_sumBPdelh      <= 0;
                cout_sumBPdelhstore  <= 0;
                cout_sumBPweightih  <= 0;
                cout_sumBPweightho  <= 0;
                cout_sumBPweightihstore  <= 0;
                cout_sumBPweighthostore  <= 0;
        end
else if (cout_cntBPerrstore == 31)
        begin
                cout_sumInitial             <= 0;
                cout_sumFFload      <= 0;
```

```
                cout_sumFF1lay      <= 0;
                cout_sumFFLUT1                  <= 0;
                cout_sumFF1store            <= 0;
                cout_sumFF2lay      <= 0;
                cout_sumFFLUT2                  <= 0;
                cout_sumFF2store            <= 0;
                cout_sumBPerrinitial <= 0;
                cout_sumBPerrstore   <= cout_sumBPerrstore + 1;
                cout_sumBPerradd    <= 0;
                cout_sumBPdeloinitial<= 0;
                cout_sumBPdelostore  <= 0;
                cout_sumBPsumdow    <= 0;
                cout_sumBPdelh      <= 0;
                cout_sumBPdelhstore  <= 0;
                cout_sumBPweightih   <= 0;
                cout_sumBPweightho   <= 0;
                cout_sumBPweightihstore  <= 0;
                cout_sumBPweighthostore  <= 0;
        end


else if (cout_cntBPerradd == 31)
        begin
                cout_sumInitial             <= 0;
                cout_sumFFload      <= 0;
                cout_sumFF1lay      <= 0;
                cout_sumFFLUT1                  <= 0;
                cout_sumFF1store            <= 0;
                cout_sumFF2lay      <= 0;
                cout_sumFFLUT2                  <= 0;
                cout_sumFF2store            <= 0;
                cout_sumBPerrinitial <= 0;
                cout_sumBPerrstore   <= 0;
                cout_sumBPerradd    <= cout_sumBPerradd + 1;
                cout_sumBPdeloinitial<= 0;
                cout_sumBPdelostore  <= 0;
                cout_sumBPsumdow    <= 0;
                cout_sumBPdelh      <= 0;
                cout_sumBPdelhstore  <= 0;
                cout_sumBPweightih   <= 0;
                cout_sumBPweightho   <= 0;
                cout_sumBPweightihstore  <= 0;
                cout_sumBPweighthostore  <= 0;
        end
else if (cout_cntBPdeloinitial == 31)
        begin
                cout_sumInitial             <= 0;
                cout_sumFFload      <= 0;
                cout_sumFF1lay      <= 0;
                cout_sumFFLUT1                  <= 0;
                cout_sumFF1store            <= 0;
                cout_sumFF2lay      <= 0;
                cout_sumFFLUT2                  <= 0;
                cout_sumFF2store            <= 0;
                cout_sumBPerrinitial <= 0;
                cout_sumBPerrstore   <= 0;
                cout_sumBPerradd    <= 0;
                cout_sumBPdeloinitial<= cout_sumBPdeloinitial + 1;
                cout_sumBPdelostore  <= 0;
                cout_sumBPsumdow    <= 0;
                cout_sumBPdelh      <= 0;
                cout_sumBPdelhstore  <= 0;
                cout_sumBPweightih   <= 0;
                cout_sumBPweightho   <= 0;
                cout_sumBPweightihstore  <= 0;
                cout_sumBPweighthostore  <= 0;
        end
else if (cout_cntBPdelostore == 31)
        begin
                cout_sumInitial             <= 0;
```

```
                cout_sumFFload      <= 0;
                cout_sumFF1lay      <= 0;
                cout_sumFFLUT1                  <= 0;
                cout_sumFF1store            <= 0;
                cout_sumFF2lay      <= 0;
                cout_sumFFLUT2                  <= 0;
                cout_sumFF2store            <= 0;
                cout_sumBPerrinitial <= 0;
                cout_sumBPerrstore  <= 0;
                cout_sumBPerradd    <= 0;
                cout_sumBPdeloinitial<= 0;
                cout_sumBPdelostore  <= cout_sumBPdelostore + 1;
                cout_sumBPsumdow    <= 0;
                cout_sumBPdelh      <= 0;
                cout_sumBPdelhstore  <= 0;
                cout_sumBPweightih  <= 0;
                cout_sumBPweightho  <= 0;
                cout_sumBPweightihstore  <= 0;
                cout_sumBPweighthostore  <= 0;
        end
else if (cout_cntBPsumdow == 31)
        begin
                cout_sumInitial             <= 0;
                cout_sumFFload      <= 0;
                cout_sumFF1lay      <= 0;
                cout_sumFFLUT1                  <= 0;
                cout_sumFF1store            <= 0;
                cout_sumFF2lay      <= 0;
                cout_sumFFLUT2                  <= 0;
                cout_sumFF2store            <= 0;
                cout_sumBPerrinitial <= 0;
                cout_sumBPerrstore  <= 0;
                cout_sumBPerradd    <= 0;
                cout_sumBPdeloinitial<= 0;
                cout_sumBPdelostore  <= 0;
                cout_sumBPsumdow    <= cout_sumBPsumdow + 1;
                cout_sumBPdelh      <= 0;
                cout_sumBPdelhstore  <= 0;
                cout_sumBPweightih  <= 0;
                cout_sumBPweightho  <= 0;
                cout_sumBPweightihstore  <= 0;
                cout_sumBPweighthostore  <= 0;
        end
else if (cout_cntBPdelh == 31)
        begin
                cout_sumInitial             <= 0;
                cout_sumFFload      <= 0;
                cout_sumFF1lay      <= 0;
                cout_sumFFLUT1                  <= 0;
                cout_sumFF1store            <= 0;
                cout_sumFF2lay      <= 0;
                cout_sumFFLUT2                  <= 0;
                cout_sumFF2store            <= 0;
                cout_sumBPerrinitial <= 0;
                cout_sumBPerrstore  <= 0;
                cout_sumBPerradd    <= 0;
                cout_sumBPdeloinitial<= 0;
                cout_sumBPdelostore  <= 0;
                cout_sumBPsumdow    <= 0;
                cout_sumBPdelh      <= cout_sumBPdelh + 1;
                cout_sumBPdelhstore  <= 0;
                cout_sumBPweightih  <= 0;
                cout_sumBPweightho  <= 0;
                cout_sumBPweightihstore  <= 0;
                cout_sumBPweighthostore  <= 0;
        end
else if (cout_cntBPdelhstore == 31)
        begin
                cout_sumInitial             <= 0;
                cout_sumFFload      <= 0;
```

```
                cout_sumFF1lay      <= 0;
                cout_sumFFLUT1                  <= 0;
                cout_sumFF1store            <= 0;
                cout_sumFF2lay      <= 0;
                cout_sumFFLUT2                  <= 0;
                cout_sumFF2store            <= 0;
                cout_sumBPerrinitial <= 0;
                cout_sumBPerrstore   <= 0;
                cout_sumBPerradd    <= 0;
                cout_sumBPdeloinitial<= 0;
                cout_sumBPdelostore  <= 0;
                cout_sumBPsumdow    <= 0;
                cout_sumBPdelh      <= 0;
                cout_sumBPdelhstore  <= cout_sumBPdelhstore + 1;
                cout_sumBPweightih   <= 0;
                cout_sumBPweightho   <= 0;
                cout_sumBPweightihstore  <= 0;
                cout_sumBPweighthostore  <= 0;
        end
else if (cout_cntBPweightih == 63)
        begin
                cout_sumInitial             <= 0;
                cout_sumFFload      <= 0;
                cout_sumFF1lay      <= 0;
                cout_sumFFLUT1                  <= 0;
                cout_sumFF1store            <= 0;
                cout_sumFF2lay      <= 0;
                cout_sumFFLUT2                  <= 0;
                cout_sumFF2store            <= 0;
                cout_sumBPerrinitial <= 0;
                cout_sumBPerrstore   <= 0;
                cout_sumBPerradd    <= 0;
                cout_sumBPdeloinitial<= 0;
                cout_sumBPdelostore  <= 0;
                cout_sumBPsumdow    <= 0;
                cout_sumBPdelh      <= 0;
                cout_sumBPdelhstore  <= 0;
                cout_sumBPweightih   <= cout_sumBPweightih + 1;
                cout_sumBPweightho   <= 0;
                cout_sumBPweightihstore  <= 0;
                cout_sumBPweighthostore  <= 0;
        end
else if (cout_cntBPweightihstore == 63)
        begin
                cout_sumInitial             <= 0;
                cout_sumFFload      <= 0;
                cout_sumFF1lay      <= 0;
                cout_sumFFLUT1                  <= 0;
                cout_sumFF1store            <= 0;
                cout_sumFF2lay      <= 0;
                cout_sumFFLUT2                  <= 0;
                cout_sumFF2store            <= 0;
                cout_sumBPerrinitial <= 0;
                cout_sumBPerrstore   <= 0;
                cout_sumBPerradd    <= 0;
                cout_sumBPdeloinitial<= 0;
                cout_sumBPdelostore  <= 0;
                cout_sumBPsumdow    <= 0;
                cout_sumBPdelh      <= 0;
                cout_sumBPdelhstore  <= 0;
                cout_sumBPweightih   <= 0;
                cout_sumBPweightho   <= 0;
                cout_sumBPweightihstore  <= cout_sumBPweightihstore + 1;
                cout_sumBPweighthostore  <= 0;
        end
else if (cout_cntBPweightho == 63)
        begin
                cout_sumInitial             <= 0;
                cout_sumFFload      <= 0;
                cout_sumFF1lay      <= 0;
```

```
                cout_sumFFLUT1                          <= 0;
                cout_sumFF1store                <= 0;
                cout_sumFF2lay          <= 0;
                cout_sumFFLUT2                          <= 0;
                cout_sumFF2store                <= 0;
                cout_sumBPerrinitial <= 0;
                cout_sumBPerrstore   <= 0;
                cout_sumBPerradd     <= 0;
                cout_sumBPdeloinitial<= 0;
                cout_sumBPdelostore  <= 0;
                cout_sumBPsumdow     <= 0;
                cout_sumBPdelh       <= 0;
                cout_sumBPdelhstore  <= 0;
                cout_sumBPweightih   <= 0;
                cout_sumBPweightho   <= cout_sumBPweightho + 1;
                cout_sumBPweightihstore  <= 0;
                cout_sumBPweighthostore  <= 0;
        end
else if (cout_cntBPweighthostore == 63)
        begin
                cout_sumInitial                 <= 0;
                cout_sumFFload          <= 0;
                cout_sumFF1lay          <= 0;
                cout_sumFFLUT1                          <= 0;
                cout_sumFF1store                <= 0;
                cout_sumFF2lay          <= 0;
                cout_sumFFLUT2                          <= 0;
                cout_sumFF2store                <= 0;
                cout_sumBPerrinitial <= 0;
                cout_sumBPerrstore   <= 0;
                cout_sumBPerradd     <= 0;
                cout_sumBPdeloinitial<= 0;
                cout_sumBPdelostore  <= 0;
                cout_sumBPsumdow     <= 0;
                cout_sumBPdelh       <= 0;
                cout_sumBPdelhstore  <= 0;
                cout_sumBPweightih   <= 0;
                cout_sumBPweightho   <= 0;
                cout_sumBPweightihstore  <= 0;
                cout_sumBPweighthostore  <= cout_sumBPweighthostore + 1;
        end

else
        begin
                cout_sumInitial                 <= cout_sumInitial;
                cout_sumFFload          <= cout_sumFFload;
                cout_sumFF1lay          <= cout_sumFF1lay;
                cout_sumFFLUT1                          <= cout_sumFFLUT1;
                cout_sumFF1store                <= cout_sumFF1store;
                cout_sumFF2lay          <= cout_sumFF2lay;
                cout_sumFFLUT2                          <= cout_sumFFLUT2;
                cout_sumFF2store                <= cout_sumFF2store;
                cout_sumBPerrinitial <= cout_sumBPerrinitial;
                cout_sumBPerrstore   <= cout_sumBPerrstore;
                cout_sumBPerradd     <= cout_sumBPerradd;
                cout_sumBPdeloinitial<= cout_sumBPdeloinitial;
                cout_sumBPdelostore  <= cout_sumBPdelostore;
                cout_sumBPsumdow     <= cout_sumBPsumdow;
                cout_sumBPdelh       <= cout_sumBPdelh;
                cout_sumBPdelhstore  <= cout_sumBPdelhstore;
                cout_sumBPweightih   <= cout_sumBPweightih;
                cout_sumBPweightho   <= cout_sumBPweightho;
                cout_sumBPweightihstore  <= cout_sumBPweightihstore;
                cout_sumBPweighthostore  <= cout_sumBPweighthostore;
        end
        end
        end
always @ (posedge clk)
        begin
if (system_rst == 1)
```

```
		begin
			LUTmux <= 0;
		end
else if (state == 4)
		begin
			LUTmux <= cout_sumFFLUT1 + 1;
		end
else if (state == 7)
		begin
			LUTmux <= cout_sumFFLUT2 + 1;
		end

		end
always @ (posedge clk)
		begin
			if (system_rst == 1)
				begin
		register_Ren_A_1 <= 0;
		register_Ren_B_1 <= 0;
		register_Wen_1   <= 0;
		register_Ren_A_2 <= 0;
		register_Ren_B_2 <= 0;
		register_Wen_2   <= 0;
		register_Ren_A_3 <= 0;
		register_Ren_B_3 <= 0;
		register_Wen_3   <= 0;
		register_Ren_A_4 <= 0;
		register_Ren_B_4 <= 0;
		register_Wen_4   <= 0;
		register_Ren_A_5 <= 0;
		register_Ren_B_5 <= 0;
		register_Wen_5   <= 0;
		register_Ren_A_6 <= 0;
		register_Ren_B_6 <= 0;
		register_Wen_6   <= 0;
		register_Ren_A_7 <= 0;
		register_Ren_B_7 <= 0;
		register_Wen_7   <= 0;
		register_Ren_A_8 <= 0;
		register_Ren_B_8 <= 0;
		register_Wen_8   <= 0;
		register_Ren_A_9 <= 0;
		register_Ren_B_9 <= 0;
		register_Wen_9   <= 0;
		register_Ren_A_10 <= 0;
		register_Ren_B_10 <= 0;
		register_Wen_10   <= 0;
		register_Ren_A_11 <= 0;
		register_Ren_B_11 <= 0;
		register_Wen_11   <= 0;
		register_Ren_A_12 <= 0;
		register_Ren_B_12 <= 0;
		register_Wen_12   <= 0;
		register_Ren_A_13 <= 0;
		register_Ren_B_13 <= 0;
		register_Wen_13   <= 0;
		register_Ren_A_14 <= 0;
		register_Ren_B_14 <= 0;
		register_Wen_14   <= 0;
		register_Ren_A_15 <= 0;
		register_Ren_B_15 <= 0;
		register_Wen_15   <= 0;
		register_Ren_A_16 <= 0;
		register_Ren_B_16 <= 0;
		register_Wen_16   <= 0;
		register_Ren_A_17 <= 0;
		register_Ren_B_17 <= 0;
		register_Wen_17   <= 0;
		register_Ren_A_18 <= 0;
		register_Ren_B_18 <= 0;
```

```
register_Wen_18   <= 0;
register_Ren_A_19 <= 0;
register_Ren_B_19 <= 0;
register_Wen_19   <= 0;
register_Ren_A_20 <= 0;
register_Ren_B_20 <= 0;
register_Wen_20   <= 0;
register_Ren_A_21 <= 0;
register_Ren_B_21 <= 0;
register_Wen_21   <= 0;
register_Ren_A_22 <= 0;
register_Ren_B_22 <= 0;
register_Wen_22   <= 0;
register_Ren_A_23 <= 0;
register_Ren_B_23 <= 0;
register_Wen_23   <= 0;
register_Ren_A_24 <= 0;
register_Ren_B_24 <= 0;
register_Wen_24   <= 0;
register_Ren_A_25 <= 0;
register_Ren_B_25 <= 0;
register_Wen_25   <= 0;
register_Ren_A_26 <= 0;
register_Ren_B_26 <= 0;
register_Wen_26   <= 0;
register_Ren_A_27 <= 0;
register_Ren_B_27 <= 0;
register_Wen_27   <= 0;
register_Ren_A_28 <= 0;
register_Ren_B_28 <= 0;
register_Wen_28   <= 0;
register_Ren_A_29 <= 0;
register_Ren_B_29 <= 0;
register_Wen_29   <= 0;
register_Ren_A_30 <= 0;
register_Ren_B_30 <= 0;
register_Wen_30   <= 0;

        MIT_rom_add1                    <= 0;
        MIT_rom_add2                    <= 0;
        MIT_rom_ren         <= 0;
        MIT_data_temp        <= 0;
        BP_code             <= 0;
        start               <= 0;
acclear                                 <= 0;
stop                                                <= 0;
cout_sumCheck                           <= 0;
cout_sumEPOC                            <= 0;
trainOK                                 <= 0;

errout <= 16'b0111111111111111;
end
        else
                begin
                        case (state)
                                0:      begin
register_Ren_A_1 <= 0;
register_Ren_B_1 <= 0;
register_Wen_1   <= 0;
register_Ren_A_2 <= 0;
register_Ren_B_2 <= 0;
register_Wen_2   <= 0;
register_Ren_A_3 <= 0;
register_Ren_B_3 <= 0;
register_Wen_3   <= 0;
register_Ren_A_4 <= 0;
register_Ren_B_4 <= 0;
register_Wen_4   <= 0;
register_Ren_A_5 <= 0;
register_Ren_B_5 <= 0;
```

```
register_Wen_5   <= 0;
register_Ren_A_6 <= 0;
register_Ren_B_6 <= 0;
register_Wen_6   <= 0;
register_Ren_A_7 <= 0;
register_Ren_B_7 <= 0;
register_Wen_7   <= 0;
register_Ren_A_8 <= 0;
register_Ren_B_8 <= 0;
register_Wen_8   <= 0;
register_Ren_A_9 <= 0;
register_Ren_B_9 <= 0;
register_Wen_9   <= 0;
register_Ren_A_10 <= 0;
register_Ren_B_10 <= 0;
register_Wen_10   <= 0;
register_Ren_A_11 <= 0;
register_Ren_B_11 <= 0;
register_Wen_11   <= 0;
register_Ren_A_12 <= 0;
register_Ren_B_12 <= 0;
register_Wen_12   <= 0;
register_Ren_A_13 <= 0;
register_Ren_B_13 <= 0;
register_Wen_13   <= 0;
register_Ren_A_14 <= 0;
register_Ren_B_14 <= 0;
register_Wen_14   <= 0;
register_Ren_A_15 <= 0;
register_Ren_B_15 <= 0;
register_Wen_15   <= 0;
register_Ren_A_16 <= 0;
register_Ren_B_16 <= 0;
register_Wen_16   <= 0;
register_Ren_A_17 <= 0;
register_Ren_B_17 <= 0;
register_Wen_17   <= 0;
register_Ren_A_18 <= 0;
register_Ren_B_18 <= 0;
register_Wen_18   <= 0;
register_Ren_A_19 <= 0;
register_Ren_B_19 <= 0;
register_Wen_19   <= 0;
register_Ren_A_20 <= 0;
register_Ren_B_20 <= 0;
register_Wen_20   <= 0;
register_Ren_A_21 <= 0;
register_Ren_B_21 <= 0;
register_Wen_21   <= 0;
register_Ren_A_22 <= 0;
register_Ren_B_22 <= 0;
register_Wen_22   <= 0;
register_Ren_A_23 <= 0;
register_Ren_B_23 <= 0;
register_Wen_23   <= 0;
register_Ren_A_24 <= 0;
register_Ren_B_24 <= 0;
register_Wen_24   <= 0;
register_Ren_A_25 <= 0;
register_Ren_B_25 <= 0;
register_Wen_25   <= 0;
register_Ren_A_26 <= 0;
register_Ren_B_26 <= 0;
register_Wen_26   <= 0;
register_Ren_A_27 <= 0;
register_Ren_B_27 <= 0;
register_Wen_27   <= 0;
register_Ren_A_28 <= 0;
register_Ren_B_28 <= 0;
register_Wen_28   <= 0;
```

```
                register_Ren_A_29 <= 0;
                register_Ren_B_29 <= 0;
                register_Wen_29   <= 0;
                register_Ren_A_30 <= 0;
                register_Ren_B_30 <= 0;
                register_Wen_30   <= 0;

                MIT_rom_add1                                    <= 0;
                MIT_rom_add2                                    <= 0;
                MIT_rom_ren             <= 0;
                MIT_data_temp           <= 0;
                Initial_rom_add                         <= 0;
                Initial_rom_ren                         <= 0;
                BP_code                 <= 0;
                start                   <= 0;
        end
        1: begin
                if (cout_cntInitial == 1)
                        begin
                                Initial_rom_add <= cout_sumInitial;
                                Initial_rom_ren <= 1;
                        end
                else if (cout_cntInitial == 4)
                        begin
                                register_Wadd_1  <=  cout_sumInitial;  register_Wen_1  <=  1;  register_Wdata_1  <=
Initial_rom_data_1;
                                register_Wadd_2  <=  cout_sumInitial;  register_Wen_2  <=  1;  register_Wdata_2  <=
Initial_rom_data_2;
                                register_Wadd_3  <=  cout_sumInitial;  register_Wen_3  <=  1;  register_Wdata_3  <=
Initial_rom_data_3;
                                register_Wadd_4  <=  cout_sumInitial;  register_Wen_4  <=  1;  register_Wdata_4  <=
Initial_rom_data_4;
                                register_Wadd_5  <=  cout_sumInitial;  register_Wen_5  <=  1;  register_Wdata_5  <=
Initial_rom_data_5;
                                register_Wadd_6  <=  cout_sumInitial;  register_Wen_6  <=  1;  register_Wdata_6  <=
Initial_rom_data_6;
                                register_Wadd_7  <=  cout_sumInitial;  register_Wen_7  <=  1;  register_Wdata_7  <=
Initial_rom_data_7;
                                register_Wadd_8  <=  cout_sumInitial;  register_Wen_8  <=  1;  register_Wdata_8  <=
Initial_rom_data_8;
                                register_Wadd_9  <=  cout_sumInitial;  register_Wen_9  <=  1;  register_Wdata_9  <=
Initial_rom_data_9;
                                register_Wadd_10 <=  cout_sumInitial; register_Wen_10 <=  1; register_Wdata_10 <=
Initial_rom_data_10;
                                register_Wadd_11 <=  cout_sumInitial; register_Wen_11 <=  1; register_Wdata_11 <=
Initial_rom_data_11;
                                register_Wadd_12 <=  cout_sumInitial; register_Wen_12 <=  1; register_Wdata_12 <=
Initial_rom_data_12;
                                register_Wadd_13 <=  cout_sumInitial; register_Wen_13 <=  1; register_Wdata_13 <=
Initial_rom_data_13;
                                register_Wadd_14 <=  cout_sumInitial; register_Wen_14 <=  1; register_Wdata_14 <=
Initial_rom_data_14;
                                register_Wadd_15 <=  cout_sumInitial; register_Wen_15 <=  1; register_Wdata_15 <=
Initial_rom_data_15;
                                register_Wadd_16 <=  cout_sumInitial; register_Wen_16 <=  1; register_Wdata_16 <=
Initial_rom_data_16;
                                register_Wadd_17 <=  cout_sumInitial; register_Wen_17 <=  1; register_Wdata_17 <=
Initial_rom_data_17;
                                register_Wadd_18 <=  cout_sumInitial; register_Wen_18 <=  1; register_Wdata_18 <=
Initial_rom_data_18;
                                register_Wadd_19 <=  cout_sumInitial; register_Wen_19 <=  1; register_Wdata_19 <=
Initial_rom_data_19;
                                register_Wadd_20 <=  cout_sumInitial; register_Wen_20 <=  1; register_Wdata_20 <=
Initial_rom_data_20;
                                register_Wadd_21 <=  cout_sumInitial; register_Wen_21 <=  1; register_Wdata_21 <=
Initial_rom_data_21;
                                register_Wadd_22 <=  cout_sumInitial; register_Wen_22 <=  1; register_Wdata_22 <=
Initial_rom_data_22;
                                register_Wadd_23 <=  cout_sumInitial; register_Wen_23 <=  1; register_Wdata_23 <=
Initial_rom_data_23;
```

```
                                        register_Wadd_24 <=  cout_sumInitial; register_Wen_24 <= 1; register_Wdata_24 <=
Initial_rom_data_24;
                                        register_Wadd_25 <=  cout_sumInitial; register_Wen_25 <= 1; register_Wdata_25 <=
Initial_rom_data_25;
                                        register_Wadd_26 <=  cout_sumInitial; register_Wen_26 <= 1; register_Wdata_26 <=
Initial_rom_data_26;
                                        register_Wadd_27 <=  cout_sumInitial; register_Wen_27 <= 1; register_Wdata_27 <=
Initial_rom_data_27;
                                        register_Wadd_28 <=  cout_sumInitial; register_Wen_28 <= 1; register_Wdata_28 <=
Initial_rom_data_28;
                                        register_Wadd_29 <=  cout_sumInitial; register_Wen_29 <= 1; register_Wdata_29 <=
Initial_rom_data_29;
                                        register_Wadd_30 <=  cout_sumInitial; register_Wen_30 <= 1; register_Wdata_30 <=
Initial_rom_data_30;
                                end
                        else if (cout_cntInitial == 6)
                                begin
                                        register_Wen_1 <= 0;
                                        register_Wen_2 <= 0;
                                        register_Wen_3 <= 0;
                                        register_Wen_4 <= 0;
                                        register_Wen_5 <= 0;
                                        register_Wen_6 <= 0;
                                        register_Wen_7 <= 0;
                                        register_Wen_8 <= 0;
                                        register_Wen_9 <= 0;
                                        register_Wen_10 <= 0;
                                        register_Wen_11 <= 0;
                                        register_Wen_12 <= 0;
                                        register_Wen_13 <= 0;
                                        register_Wen_14 <= 0;
                                        register_Wen_15 <= 0;
                                        register_Wen_16 <= 0;
                                        register_Wen_17 <= 0;
                                        register_Wen_18 <= 0;
                                        register_Wen_19 <= 0;
                                        register_Wen_20 <= 0;
                                        register_Wen_21 <= 0;
                                        register_Wen_22 <= 0;
                                        register_Wen_23 <= 0;
                                        register_Wen_24 <= 0;
                                        register_Wen_25 <= 0;
                                        register_Wen_26 <= 0;
                                        register_Wen_27 <= 0;
                                        register_Wen_28 <= 0;
                                        register_Wen_29 <= 0;
                                        register_Wen_30 <= 0;
                                end
                        else if (cout_cntInitial == 7)
                                begin
                                        Initial_rom_ren         <= 0;
                                end
                end
        2:      begin
                        if (cout_cntFFload == 1)
                                begin
                                        MIT_rom_add1        <= cout_sumCheck + 1;
                                        MIT_rom_add2        <= cout_sumFFload;
                                        MIT_rom_ren         <= 1;
                                end
                        else if (cout_cntFFload == 4)
                                begin
                                        MIT_data_temp       <= MIT_rom_data;
                                        MIT_rom_ren         <= 0;
                                end
                        else if (cout_cntFFload == 5)
                                begin
                                        register_Wadd_1  <=  REGaddECG  +  cout_sumFFload;  register_Wen_1  <=  1;
register_Wdata_1 <= MIT_data_temp;
```

```
                register_Wadd_2 <= REGaddECG + cout_sumFFload; register_Wen_2 <= 1;
register_Wdata_2 <= MIT_data_temp;
                register_Wadd_3 <= REGaddECG + cout_sumFFload; register_Wen_3 <= 1;
register_Wdata_3 <= MIT_data_temp;
                register_Wadd_4 <= REGaddECG + cout_sumFFload; register_Wen_4 <= 1;
register_Wdata_4 <= MIT_data_temp;
                register_Wadd_5 <= REGaddECG + cout_sumFFload; register_Wen_5 <= 1;
register_Wdata_5 <= MIT_data_temp;
                register_Wadd_6 <= REGaddECG + cout_sumFFload; register_Wen_6 <= 1;
register_Wdata_6 <= MIT_data_temp;
                register_Wadd_7 <= REGaddECG + cout_sumFFload; register_Wen_7 <= 1;
register_Wdata_7 <= MIT_data_temp;
                register_Wadd_8 <= REGaddECG + cout_sumFFload; register_Wen_8 <= 1;
register_Wdata_8 <= MIT_data_temp;
                register_Wadd_9 <= REGaddECG + cout_sumFFload; register_Wen_9 <= 1;
register_Wdata_9 <= MIT_data_temp;
                register_Wadd_10 <= REGaddECG + cout_sumFFload; register_Wen_10 <= 1;
register_Wdata_10 <= MIT_data_temp;
                register_Wadd_11 <= REGaddECG + cout_sumFFload; register_Wen_11 <= 1;
register_Wdata_11 <= MIT_data_temp;
                register_Wadd_12 <= REGaddECG + cout_sumFFload; register_Wen_12 <= 1;
register_Wdata_12 <= MIT_data_temp;
                register_Wadd_13 <= REGaddECG + cout_sumFFload; register_Wen_13 <= 1;
register_Wdata_13 <= MIT_data_temp;
                register_Wadd_14 <= REGaddECG + cout_sumFFload; register_Wen_14 <= 1;
register_Wdata_14 <= MIT_data_temp;
                register_Wadd_15 <= REGaddECG + cout_sumFFload; register_Wen_15 <= 1;
register_Wdata_15 <= MIT_data_temp;
                register_Wadd_16 <= REGaddECG + cout_sumFFload; register_Wen_16 <= 1;
register_Wdata_16 <= MIT_data_temp;
                register_Wadd_17 <= REGaddECG + cout_sumFFload; register_Wen_17 <= 1;
register_Wdata_17 <= MIT_data_temp;
                register_Wadd_18 <= REGaddECG + cout_sumFFload; register_Wen_18 <= 1;
register_Wdata_18 <= MIT_data_temp;
                register_Wadd_19 <= REGaddECG + cout_sumFFload; register_Wen_19 <= 1;
register_Wdata_19 <= MIT_data_temp;
                register_Wadd_20 <= REGaddECG + cout_sumFFload; register_Wen_20 <= 1;
register_Wdata_20 <= MIT_data_temp;
                register_Wadd_21 <= REGaddECG + cout_sumFFload; register_Wen_21 <= 1;
register_Wdata_21 <= MIT_data_temp;
                register_Wadd_22 <= REGaddECG + cout_sumFFload; register_Wen_22 <= 1;
register_Wdata_22 <= MIT_data_temp;
                register_Wadd_23 <= REGaddECG + cout_sumFFload; register_Wen_23 <= 1;
register_Wdata_23 <= MIT_data_temp;
                register_Wadd_24 <= REGaddECG + cout_sumFFload; register_Wen_24 <= 1;
register_Wdata_24 <= MIT_data_temp;
                register_Wadd_25 <= REGaddECG + cout_sumFFload; register_Wen_25 <= 1;
register_Wdata_25 <= MIT_data_temp;
                register_Wadd_26 <= REGaddECG + cout_sumFFload; register_Wen_26 <= 1;
register_Wdata_26 <= MIT_data_temp;
                register_Wadd_27 <= REGaddECG + cout_sumFFload; register_Wen_27 <= 1;
register_Wdata_27 <= MIT_data_temp;
                register_Wadd_28 <= REGaddECG + cout_sumFFload; register_Wen_28 <= 1;
register_Wdata_28 <= MIT_data_temp;
                register_Wadd_29 <= REGaddECG + cout_sumFFload; register_Wen_29 <= 1;
register_Wdata_29 <= MIT_data_temp;
                register_Wadd_30 <= REGaddECG + cout_sumFFload; register_Wen_30 <= 1;
register_Wdata_30 <= MIT_data_temp;
            end
        else if (cout_cntFFload == 6)
            begin
                register_Wen_1 <= 0;
                register_Wen_2 <= 0;
                register_Wen_3 <= 0;
                register_Wen_4 <= 0;
                register_Wen_5 <= 0;
                register_Wen_6 <= 0;
                register_Wen_7 <= 0;
                register_Wen_8 <= 0;
                register_Wen_9 <= 0;
```

```
                                        register_Wen_10 <= 0;
                                        register_Wen_11 <= 0;
                                        register_Wen_12 <= 0;
                                        register_Wen_13 <= 0;
                                        register_Wen_14 <= 0;
                                        register_Wen_15 <= 0;
                                        register_Wen_16 <= 0;
                                        register_Wen_17 <= 0;
                                        register_Wen_18 <= 0;
                                        register_Wen_19 <= 0;
                                        register_Wen_20 <= 0;
                                        register_Wen_21 <= 0;
                                        register_Wen_22 <= 0;
                                        register_Wen_23 <= 0;
                                        register_Wen_24 <= 0;
                                        register_Wen_25 <= 0;
                                        register_Wen_26 <= 0;
                                        register_Wen_27 <= 0;
                                        register_Wen_28 <= 0;
                                        register_Wen_29 <= 0;
                                        register_Wen_30 <= 0;
                                    end
                            end
                    3: begin
                                    if (cout_cntFF1lay == 1)
                                        begin
                                            register_Radd_A_1 <= REGaddECG + cout_sumFF1lay-1; register_Radd_B_1 <= REGaddinweight + cout_sumFF1lay; register_Ren_A_1 <= 1; register_Ren_B_1 <= 1;
                                            register_Radd_A_2 <= REGaddECG + cout_sumFF1lay-1; register_Radd_B_2 <= REGaddinweight + cout_sumFF1lay; register_Ren_A_2 <= 1; register_Ren_B_2 <= 1;
                                            register_Radd_A_3 <= REGaddECG + cout_sumFF1lay-1; register_Radd_B_3 <= REGaddinweight + cout_sumFF1lay; register_Ren_A_3 <= 1; register_Ren_B_3 <= 1;
                                            register_Radd_A_4 <= REGaddECG + cout_sumFF1lay-1; register_Radd_B_4 <= REGaddinweight + cout_sumFF1lay; register_Ren_A_4 <= 1; register_Ren_B_4 <= 1;
                                            register_Radd_A_5 <= REGaddECG + cout_sumFF1lay-1; register_Radd_B_5 <= REGaddinweight + cout_sumFF1lay; register_Ren_A_5 <= 1; register_Ren_B_5 <= 1;
                                            register_Radd_A_6 <= REGaddECG + cout_sumFF1lay-1; register_Radd_B_6 <= REGaddinweight + cout_sumFF1lay; register_Ren_A_6 <= 1; register_Ren_B_6 <= 1;
                                            register_Radd_A_7 <= REGaddECG + cout_sumFF1lay-1; register_Radd_B_7 <= REGaddinweight + cout_sumFF1lay; register_Ren_A_7 <= 1; register_Ren_B_7 <= 1;
                                            register_Radd_A_8 <= REGaddECG + cout_sumFF1lay-1; register_Radd_B_8 <= REGaddinweight + cout_sumFF1lay; register_Ren_A_8 <= 1; register_Ren_B_8 <= 1;
                                            register_Radd_A_9 <= REGaddECG + cout_sumFF1lay-1; register_Radd_B_9 <= REGaddinweight + cout_sumFF1lay; register_Ren_A_9 <= 1; register_Ren_B_9 <= 1;
                                            register_Radd_A_10 <= REGaddECG + cout_sumFF1lay-1; register_Radd_B_10 <= REGaddinweight + cout_sumFF1lay; register_Ren_A_10 <= 1; register_Ren_B_10 <= 1;
                                            register_Radd_A_11 <= REGaddECG + cout_sumFF1lay-1; register_Radd_B_11 <= REGaddinweight + cout_sumFF1lay; register_Ren_A_11 <= 1; register_Ren_B_11 <= 1;
                                            register_Radd_A_12 <= REGaddECG + cout_sumFF1lay-1; register_Radd_B_12 <= REGaddinweight + cout_sumFF1lay; register_Ren_A_12 <= 1; register_Ren_B_12 <= 1;
                                            register_Radd_A_13 <= REGaddECG + cout_sumFF1lay-1; register_Radd_B_13 <= REGaddinweight + cout_sumFF1lay; register_Ren_A_13 <= 1; register_Ren_B_13 <= 1;
                                            register_Radd_A_14 <= REGaddECG + cout_sumFF1lay-1; register_Radd_B_14 <= REGaddinweight + cout_sumFF1lay; register_Ren_A_14 <= 1; register_Ren_B_14 <= 1;
                                            register_Radd_A_15 <= REGaddECG + cout_sumFF1lay-1; register_Radd_B_15 <= REGaddinweight + cout_sumFF1lay; register_Ren_A_15 <= 1; register_Ren_B_15 <= 1;
                                            register_Radd_A_16 <= REGaddECG + cout_sumFF1lay-1; register_Radd_B_16 <= REGaddinweight + cout_sumFF1lay; register_Ren_A_16 <= 1; register_Ren_B_16 <= 1;
                                            register_Radd_A_17 <= REGaddECG + cout_sumFF1lay-1; register_Radd_B_17 <= REGaddinweight + cout_sumFF1lay; register_Ren_A_17 <= 1; register_Ren_B_17 <= 1;
                                            register_Radd_A_18 <= REGaddECG + cout_sumFF1lay-1; register_Radd_B_18 <= REGaddinweight + cout_sumFF1lay; register_Ren_A_18 <= 1; register_Ren_B_18 <= 1;
                                            register_Radd_A_19 <= REGaddECG + cout_sumFF1lay-1; register_Radd_B_19 <= REGaddinweight + cout_sumFF1lay; register_Ren_A_19 <= 1; register_Ren_B_19 <= 1;
                                            register_Radd_A_20 <= REGaddECG + cout_sumFF1lay-1; register_Radd_B_20 <= REGaddinweight + cout_sumFF1lay; register_Ren_A_20 <= 1; register_Ren_B_20 <= 1;
                                            register_Radd_A_21 <= REGaddECG + cout_sumFF1lay-1; register_Radd_B_21 <= REGaddinweight + cout_sumFF1lay; register_Ren_A_21 <= 1; register_Ren_B_21 <= 1;
                                            register_Radd_A_22 <= REGaddECG + cout_sumFF1lay-1; register_Radd_B_22 <= REGaddinweight + cout_sumFF1lay; register_Ren_A_22 <= 1; register_Ren_B_22 <= 1;
```

```
                                        register_Radd_A_23 <= REGaddECG + cout_sumFF1lay-1; register_Radd_B_23 <=
REGaddinweight + cout_sumFF1lay; register_Ren_A_23 <= 1; register_Ren_B_23 <= 1;
                                        register_Radd_A_24 <= REGaddECG + cout_sumFF1lay-1; register_Radd_B_24 <=
REGaddinweight + cout_sumFF1lay; register_Ren_A_24 <= 1; register_Ren_B_24 <= 1;
                                        register_Radd_A_25 <= REGaddECG + cout_sumFF1lay-1; register_Radd_B_25 <=
REGaddinweight + cout_sumFF1lay; register_Ren_A_25 <= 1; register_Ren_B_25 <= 1;
                                        register_Radd_A_26 <= REGaddECG + cout_sumFF1lay-1; register_Radd_B_26 <=
REGaddinweight + cout_sumFF1lay; register_Ren_A_26 <= 1; register_Ren_B_26 <= 1;
                                        register_Radd_A_27 <= REGaddECG + cout_sumFF1lay-1; register_Radd_B_27 <=
REGaddinweight + cout_sumFF1lay; register_Ren_A_27 <= 1; register_Ren_B_27 <= 1;
                                        register_Radd_A_28 <= REGaddECG + cout_sumFF1lay-1; register_Radd_B_28 <=
REGaddinweight + cout_sumFF1lay; register_Ren_A_28 <= 1; register_Ren_B_28 <= 1;
                                        register_Radd_A_29 <= REGaddECG + cout_sumFF1lay-1; register_Radd_B_29 <=
REGaddinweight + cout_sumFF1lay; register_Ren_A_29 <= 1; register_Ren_B_29 <= 1;
                                        register_Radd_A_30 <= REGaddECG + cout_sumFF1lay-1; register_Radd_B_30 <=
REGaddinweight + cout_sumFF1lay; register_Ren_A_30 <= 1; register_Ren_B_30 <= 1;

                                        BP_code <= 5;
                        end
                else if (cout_cntFF1lay == 5)
                        begin
                                if (cout_sumFF1lay == 0)
                                        begin
                                                acclear <= 1;
                                        end
                        end
                else if (cout_cntFF1lay == 7)
                        begin
                                if (cout_sumFF1lay == 0)
                                        begin
                                                acclear <= 0;
                                        end
                        end
                else if (cout_cntFF1lay == 9)
                        begin
                                start           <= 1;
                        end
                else if (cout_cntFF1lay == 10)
                        begin
                                start                   <= 0;
                        end
                else if (cout_cntFF1lay == 20)
                        begin
                                register_Ren_A_1 <= 0; register_Ren_B_1 <= 0;
                                register_Ren_A_2 <= 0; register_Ren_B_2 <= 0;
                                register_Ren_A_3 <= 0; register_Ren_B_3 <= 0;
                                register_Ren_A_4 <= 0; register_Ren_B_4 <= 0;
                                register_Ren_A_5 <= 0; register_Ren_B_5 <= 0;
                                register_Ren_A_6 <= 0; register_Ren_B_6 <= 0;
                                register_Ren_A_7 <= 0; register_Ren_B_7 <= 0;
                                register_Ren_A_8 <= 0; register_Ren_B_8 <= 0;
                                register_Ren_A_9 <= 0; register_Ren_B_9 <= 0;
                                register_Ren_A_10 <= 0; register_Ren_B_10 <= 0;
                                register_Ren_A_11 <= 0; register_Ren_B_11 <= 0;
                                register_Ren_A_12 <= 0; register_Ren_B_12 <= 0;
                                register_Ren_A_13 <= 0; register_Ren_B_13 <= 0;
                                register_Ren_A_14 <= 0; register_Ren_B_14 <= 0;
                                register_Ren_A_15 <= 0; register_Ren_B_15 <= 0;
                                register_Ren_A_16 <= 0; register_Ren_B_16 <= 0;
                                register_Ren_A_17 <= 0; register_Ren_B_17 <= 0;
                                register_Ren_A_18 <= 0; register_Ren_B_18 <= 0;
                                register_Ren_A_19 <= 0; register_Ren_B_19 <= 0;
                                register_Ren_A_20 <= 0; register_Ren_B_20 <= 0;
                                register_Ren_A_21 <= 0; register_Ren_B_21 <= 0;
                                register_Ren_A_22 <= 0; register_Ren_B_22 <= 0;
                                register_Ren_A_23 <= 0; register_Ren_B_23 <= 0;
                                register_Ren_A_24 <= 0; register_Ren_B_24 <= 0;
                                register_Ren_A_25 <= 0; register_Ren_B_25 <= 0;
                                register_Ren_A_26 <= 0; register_Ren_B_26 <= 0;
                                register_Ren_A_27 <= 0; register_Ren_B_27 <= 0;
```

```
                                        register_Ren_A_28 <= 0; register_Ren_B_28 <= 0;
                                        register_Ren_A_29 <= 0; register_Ren_B_29 <= 0;
                                        register_Ren_A_30 <= 0; register_Ren_B_30 <= 0;
                                end
                end
        4:      begin
                        if (cout_cntFFLUT1 == 1)
                                begin
                                        BP_code <= 7;
                                end
                        else if (cout_cntFFLUT1 == 24)
                                begin
                                        start       <= 1;
                                end
                        else if (cout_cntFFLUT1 == 25)
                                begin
                                        start <= 0;
                                end
                        else if (cout_cntFFLUT1 == 30)
                                begin
                                        neuron_out_temp[1] <= neuron_out_1;
                                        neuron_out_temp[2] <= neuron_out_2;
                                        neuron_out_temp[3] <= neuron_out_3;
                                        neuron_out_temp[4] <= neuron_out_4;
                                        neuron_out_temp[5] <= neuron_out_5;
                                        neuron_out_temp[6] <= neuron_out_6;
                                        neuron_out_temp[7] <= neuron_out_7;
                                        neuron_out_temp[8] <= neuron_out_8;
                                        neuron_out_temp[9] <= neuron_out_9;
                                        neuron_out_temp[10] <= neuron_out_10;
                                        neuron_out_temp[11] <= neuron_out_11;
                                        neuron_out_temp[12] <= neuron_out_12;
                                        neuron_out_temp[13] <= neuron_out_13;
                                        neuron_out_temp[14] <= neuron_out_14;
                                        neuron_out_temp[15] <= neuron_out_15;
                                        neuron_out_temp[16] <= neuron_out_16;
                                        neuron_out_temp[17] <= neuron_out_17;
                                        neuron_out_temp[18] <= neuron_out_18;
                                        neuron_out_temp[19] <= neuron_out_19;
                                        neuron_out_temp[20] <= neuron_out_20;
                                        neuron_out_temp[21] <= neuron_out_21;
                                        neuron_out_temp[22] <= neuron_out_22;
                                        neuron_out_temp[23] <= neuron_out_23;
                                        neuron_out_temp[24] <= neuron_out_24;
                                        neuron_out_temp[25] <= neuron_out_25;
                                        neuron_out_temp[26] <= neuron_out_26;
                                        neuron_out_temp[27] <= neuron_out_27;
                                        neuron_out_temp[28] <= neuron_out_28;
                                        neuron_out_temp[29] <= neuron_out_29;
                                        neuron_out_temp[30] <= neuron_out_30;
                                end
                end
        5: begin
                        if (cout_cntFF1store == 3)
                                begin
                                        register_Wadd_1 <= REGaddout1 + cout_sumFF1store; register_Wen_1 <= 1;
register_Wdata_1 <= neuron_out_temp[cout_sumFF1store+1];
                                        register_Wadd_2 <= REGaddout1 + cout_sumFF1store; register_Wen_2 <= 1;
register_Wdata_2 <= neuron_out_temp[cout_sumFF1store+1];
                                        register_Wadd_3 <= REGaddout1 + cout_sumFF1store; register_Wen_3 <= 1;
register_Wdata_3 <= neuron_out_temp[cout_sumFF1store+1];
                                        register_Wadd_4 <= REGaddout1 + cout_sumFF1store; register_Wen_4 <= 1;
register_Wdata_4 <= neuron_out_temp[cout_sumFF1store+1];
                                        register_Wadd_5 <= REGaddout1 + cout_sumFF1store; register_Wen_5 <= 1;
register_Wdata_5 <= neuron_out_temp[cout_sumFF1store+1];
                                        register_Wadd_6 <= REGaddout1 + cout_sumFF1store; register_Wen_6 <= 1;
register_Wdata_6 <= neuron_out_temp[cout_sumFF1store+1];
                                        register_Wadd_7 <= REGaddout1 + cout_sumFF1store; register_Wen_7 <= 1;
register_Wdata_7 <= neuron_out_temp[cout_sumFF1store+1];
```

```
                register_Wadd_8 <= REGaddout1 + cout_sumFF1store; register_Wen_8 <= 1;
register_Wdata_8 <= neuron_out_temp[cout_sumFF1store+1];
                register_Wadd_9 <= REGaddout1 + cout_sumFF1store; register_Wen_9 <= 1;
register_Wdata_9 <= neuron_out_temp[cout_sumFF1store+1];
                register_Wadd_10 <= REGaddout1 + cout_sumFF1store; register_Wen_10 <= 1;
register_Wdata_10 <= neuron_out_temp[cout_sumFF1store+1];
                register_Wadd_11 <= REGaddout1 + cout_sumFF1store; register_Wen_11 <= 1;
register_Wdata_11 <= neuron_out_temp[cout_sumFF1store+1];
                register_Wadd_12 <= REGaddout1 + cout_sumFF1store; register_Wen_12 <= 1;
register_Wdata_12 <= neuron_out_temp[cout_sumFF1store+1];
                register_Wadd_13 <= REGaddout1 + cout_sumFF1store; register_Wen_13 <= 1;
register_Wdata_13 <= neuron_out_temp[cout_sumFF1store+1];
                register_Wadd_14 <= REGaddout1 + cout_sumFF1store; register_Wen_14 <= 1;
register_Wdata_14 <= neuron_out_temp[cout_sumFF1store+1];
                register_Wadd_15 <= REGaddout1 + cout_sumFF1store; register_Wen_15 <= 1;
register_Wdata_15 <= neuron_out_temp[cout_sumFF1store+1];
                register_Wadd_16 <= REGaddout1 + cout_sumFF1store; register_Wen_16 <= 1;
register_Wdata_16 <= neuron_out_temp[cout_sumFF1store+1];
                register_Wadd_17 <= REGaddout1 + cout_sumFF1store; register_Wen_17 <= 1;
register_Wdata_17 <= neuron_out_temp[cout_sumFF1store+1];
                register_Wadd_18 <= REGaddout1 + cout_sumFF1store; register_Wen_18 <= 1;
register_Wdata_18 <= neuron_out_temp[cout_sumFF1store+1];
                register_Wadd_19 <= REGaddout1 + cout_sumFF1store; register_Wen_19 <= 1;
register_Wdata_19 <= neuron_out_temp[cout_sumFF1store+1];
                register_Wadd_20 <= REGaddout1 + cout_sumFF1store; register_Wen_20 <= 1;
register_Wdata_20 <= neuron_out_temp[cout_sumFF1store+1];
                register_Wadd_21 <= REGaddout1 + cout_sumFF1store; register_Wen_21 <= 1;
register_Wdata_21 <= neuron_out_temp[cout_sumFF1store+1];
                register_Wadd_22 <= REGaddout1 + cout_sumFF1store; register_Wen_22 <= 1;
register_Wdata_22 <= neuron_out_temp[cout_sumFF1store+1];
                register_Wadd_23 <= REGaddout1 + cout_sumFF1store; register_Wen_23 <= 1;
register_Wdata_23 <= neuron_out_temp[cout_sumFF1store+1];
                register_Wadd_24 <= REGaddout1 + cout_sumFF1store; register_Wen_24 <= 1;
register_Wdata_24 <= neuron_out_temp[cout_sumFF1store+1];
                register_Wadd_25 <= REGaddout1 + cout_sumFF1store; register_Wen_25 <= 1;
register_Wdata_25 <= neuron_out_temp[cout_sumFF1store+1];
                register_Wadd_26 <= REGaddout1 + cout_sumFF1store; register_Wen_26 <= 1;
register_Wdata_26 <= neuron_out_temp[cout_sumFF1store+1];
                register_Wadd_27 <= REGaddout1 + cout_sumFF1store; register_Wen_27 <= 1;
register_Wdata_27 <= neuron_out_temp[cout_sumFF1store+1];
                register_Wadd_28 <= REGaddout1 + cout_sumFF1store; register_Wen_28 <= 1;
register_Wdata_28 <= neuron_out_temp[cout_sumFF1store+1];
                register_Wadd_29 <= REGaddout1 + cout_sumFF1store; register_Wen_29 <= 1;
register_Wdata_29 <= neuron_out_temp[cout_sumFF1store+1];
                register_Wadd_30 <= REGaddout1 + cout_sumFF1store; register_Wen_30 <= 1;
register_Wdata_30 <= neuron_out_temp[cout_sumFF1store+1];
            end
        else if (cout_cntFF1store == 7)
            begin
                register_Wen_1 <= 0;
                register_Wen_2 <= 0;
                register_Wen_3 <= 0;
                register_Wen_4 <= 0;
                register_Wen_5 <= 0;
                register_Wen_6 <= 0;
                register_Wen_7 <= 0;
                register_Wen_8 <= 0;
                register_Wen_9 <= 0;
                register_Wen_10 <= 0;
                register_Wen_11 <= 0;
                register_Wen_12 <= 0;
                register_Wen_13 <= 0;
                register_Wen_14 <= 0;
                register_Wen_15 <= 0;
                register_Wen_16 <= 0;
                register_Wen_17 <= 0;
                register_Wen_18 <= 0;
                register_Wen_19 <= 0;
                register_Wen_20 <= 0;
                register_Wen_21 <= 0;
```

```
                                        register_Wen_22 <= 0;
                                        register_Wen_23 <= 0;
                                        register_Wen_24 <= 0;
                                        register_Wen_25 <= 0;
                                        register_Wen_26 <= 0;
                                        register_Wen_27 <= 0;
                                        register_Wen_28 <= 0;
                                        register_Wen_29 <= 0;
                                        register_Wen_30 <= 0;
                                end
                end
        6:      begin
                        if (cout_cntFF2lay == 1)
                                begin
                                        register_Radd_A_1 <= REGaddout1 + cout_sumFF2lay-1; register_Radd_B_1 <=
REGaddinweight2 + cout_sumFF2lay; register_Ren_A_1 <= 1; register_Ren_B_1 <= 1;
                                        register_Radd_A_2 <= REGaddout1 + cout_sumFF2lay-1; register_Radd_B_2 <=
REGaddinweight2 + cout_sumFF2lay; register_Ren_A_2 <= 1; register_Ren_B_2 <= 1;
                                        register_Radd_A_3 <= REGaddout1 + cout_sumFF2lay-1; register_Radd_B_3 <=
REGaddinweight2 + cout_sumFF2lay; register_Ren_A_3 <= 1; register_Ren_B_3 <= 1;
                                        register_Radd_A_4 <= REGaddout1 + cout_sumFF2lay-1; register_Radd_B_4 <=
REGaddinweight2 + cout_sumFF2lay; register_Ren_A_4 <= 1; register_Ren_B_4 <= 1;
                                        register_Radd_A_5 <= REGaddout1 + cout_sumFF2lay-1; register_Radd_B_5 <=
REGaddinweight2 + cout_sumFF2lay; register_Ren_A_5 <= 1; register_Ren_B_5 <= 1;
                                        register_Radd_A_6 <= REGaddout1 + cout_sumFF2lay-1; register_Radd_B_6 <=
REGaddinweight2 + cout_sumFF2lay; register_Ren_A_6 <= 1; register_Ren_B_6 <= 1;
                                        register_Radd_A_7 <= REGaddout1 + cout_sumFF2lay-1; register_Radd_B_7 <=
REGaddinweight2 + cout_sumFF2lay; register_Ren_A_7 <= 1; register_Ren_B_7 <= 1;
                                        register_Radd_A_8 <= REGaddout1 + cout_sumFF2lay-1; register_Radd_B_8 <=
REGaddinweight2 + cout_sumFF2lay; register_Ren_A_8 <= 1; register_Ren_B_8 <= 1;
                                        register_Radd_A_9 <= REGaddout1 + cout_sumFF2lay-1; register_Radd_B_9 <=
REGaddinweight2 + cout_sumFF2lay; register_Ren_A_9 <= 1; register_Ren_B_9 <= 1;
                                        register_Radd_A_10 <= REGaddout1 + cout_sumFF2lay-1; register_Radd_B_10 <=
REGaddinweight2 + cout_sumFF2lay; register_Ren_A_10 <= 1; register_Ren_B_10 <= 1;
                                        register_Radd_A_11 <= REGaddout1 + cout_sumFF2lay-1; register_Radd_B_11 <=
REGaddinweight2 + cout_sumFF2lay; register_Ren_A_11 <= 1; register_Ren_B_11 <= 1;
                                        register_Radd_A_12 <= REGaddout1 + cout_sumFF2lay-1; register_Radd_B_12 <=
REGaddinweight2 + cout_sumFF2lay; register_Ren_A_12 <= 1; register_Ren_B_12 <= 1;

                                        BP_code <= 5;
                                end
                        else if (cout_cntFF2lay == 5)
                                begin
                                        if (cout_sumFF2lay == 0)
                                                begin
                                                        acclear <= 1;
                                                end
                                end
                        else if (cout_cntFF2lay == 7)
                                begin
                                        if (cout_sumFF2lay == 0)
                                                begin
                                                        acclear <= 0;
                                                end

                                end
                        else if (cout_cntFF2lay == 9)
                                begin
                                        start   <= 1;
                                end
                        else if (cout_cntFF2lay == 10)
                                begin
                                        start           <= 0;
                                end
                        else if (cout_cntFF2lay == 20)
                                begin
                                        register_Ren_A_1 <= 0; register_Ren_B_1 <= 0;
                                        register_Ren_A_2 <= 0; register_Ren_B_2 <= 0;
                                        register_Ren_A_3 <= 0; register_Ren_B_3 <= 0;
                                        register_Ren_A_4 <= 0; register_Ren_B_4 <= 0;
```

```
                                        register_Ren_A_5 <= 0; register_Ren_B_5 <= 0;
                                        register_Ren_A_6 <= 0; register_Ren_B_6 <= 0;
                                        register_Ren_A_7 <= 0; register_Ren_B_7 <= 0;
                                        register_Ren_A_8 <= 0; register_Ren_B_8 <= 0;
                                        register_Ren_A_9 <= 0; register_Ren_B_9 <= 0;
                                        register_Ren_A_10 <= 0; register_Ren_B_10 <= 0;
                                        register_Ren_A_11 <= 0; register_Ren_B_11 <= 0;
                                        register_Ren_A_12 <= 0; register_Ren_B_12 <= 0;
                                end
                end
        7: begin
                        if (cout_cntFFLUT2 == 1)
                                begin
                                        BP_code <= 7;
                                end
                        else if (cout_cntFFLUT2 == 24)
                                begin
                                        start          <= 1;
                                end
                        else if (cout_cntFFLUT2 == 25)
                                begin
                                        start <= 0;
                                end
                        else if (cout_cntFFLUT2 == 30)
                                begin
                                        neuron_out_temp[1] <= neuron_out_1;
                                        neuron_out_temp[2] <= neuron_out_2;
                                        neuron_out_temp[3] <= neuron_out_3;
                                        neuron_out_temp[4] <= neuron_out_4;
                                        neuron_out_temp[5] <= neuron_out_5;
                                        neuron_out_temp[6] <= neuron_out_6;
                                        neuron_out_temp[7] <= neuron_out_7;
                                        neuron_out_temp[8] <= neuron_out_8;
                                        neuron_out_temp[9] <= neuron_out_9;
                                        neuron_out_temp[10] <= neuron_out_10;
                                        neuron_out_temp[11] <= neuron_out_11;
                                        neuron_out_temp[12] <= neuron_out_12;
                                end
                end
        8:      begin
                        if (cout_cntFF2store == 2)
                                begin
                                        register_Wadd_1  <=  REGaddout2  +  cout_sumFF2store;  register_Wen_1  <=  1;
register_Wdata_1 <= neuron_out_temp[cout_sumFF2store+1];
                                        register_Wadd_2  <=  REGaddout2  +  cout_sumFF2store;  register_Wen_2  <=  1;
register_Wdata_2 <= neuron_out_temp[cout_sumFF2store+1];
                                        register_Wadd_3  <=  REGaddout2  +  cout_sumFF2store;  register_Wen_3  <=  1;
register_Wdata_3 <= neuron_out_temp[cout_sumFF2store+1];
                                        register_Wadd_4  <=  REGaddout2  +  cout_sumFF2store;  register_Wen_4  <=  1;
register_Wdata_4 <= neuron_out_temp[cout_sumFF2store+1];
                                        register_Wadd_5  <=  REGaddout2  +  cout_sumFF2store;  register_Wen_5  <=  1;
register_Wdata_5 <= neuron_out_temp[cout_sumFF2store+1];
                                        register_Wadd_6  <=  REGaddout2  +  cout_sumFF2store;  register_Wen_6  <=  1;
register_Wdata_6 <= neuron_out_temp[cout_sumFF2store+1];
                                        register_Wadd_7  <=  REGaddout2  +  cout_sumFF2store;  register_Wen_7  <=  1;
register_Wdata_7 <= neuron_out_temp[cout_sumFF2store+1];
                                        register_Wadd_8  <=  REGaddout2  +  cout_sumFF2store;  register_Wen_8  <=  1;
register_Wdata_8 <= neuron_out_temp[cout_sumFF2store+1];
                                        register_Wadd_9  <=  REGaddout2  +  cout_sumFF2store;  register_Wen_9  <=  1;
register_Wdata_9 <= neuron_out_temp[cout_sumFF2store+1];
                                        register_Wadd_10  <=  REGaddout2  +  cout_sumFF2store;  register_Wen_10  <=  1;
register_Wdata_10 <= neuron_out_temp[cout_sumFF2store+1];
                                        register_Wadd_11  <=  REGaddout2  +  cout_sumFF2store;  register_Wen_11  <=  1;
register_Wdata_11 <= neuron_out_temp[cout_sumFF2store+1];
                                        register_Wadd_12  <=  REGaddout2  +  cout_sumFF2store;  register_Wen_12  <=  1;
register_Wdata_12 <= neuron_out_temp[cout_sumFF2store+1];
                                        register_Wadd_13  <=  REGaddout2  +  cout_sumFF2store;  register_Wen_13  <=  1;
register_Wdata_13 <= neuron_out_temp[cout_sumFF2store+1];
                                        register_Wadd_14  <=  REGaddout2  +  cout_sumFF2store;  register_Wen_14  <=  1;
register_Wdata_14 <= neuron_out_temp[cout_sumFF2store+1];
```

```
                                        register_Wadd_15 <= REGaddout2 + cout_sumFF2store; register_Wen_15 <= 1;
register_Wdata_15 <= neuron_out_temp[cout_sumFF2store+1];
                                        register_Wadd_16 <= REGaddout2 + cout_sumFF2store; register_Wen_16 <= 1;
register_Wdata_16 <= neuron_out_temp[cout_sumFF2store+1];
                                        register_Wadd_17 <= REGaddout2 + cout_sumFF2store; register_Wen_17 <= 1;
register_Wdata_17 <= neuron_out_temp[cout_sumFF2store+1];
                                        register_Wadd_18 <= REGaddout2 + cout_sumFF2store; register_Wen_18 <= 1;
register_Wdata_18 <= neuron_out_temp[cout_sumFF2store+1];
                                        register_Wadd_19 <= REGaddout2 + cout_sumFF2store; register_Wen_19 <= 1;
register_Wdata_19 <= neuron_out_temp[cout_sumFF2store+1];
                                        register_Wadd_20 <= REGaddout2 + cout_sumFF2store; register_Wen_20 <= 1;
register_Wdata_20 <= neuron_out_temp[cout_sumFF2store+1];
                                        register_Wadd_21 <= REGaddout2 + cout_sumFF2store; register_Wen_21 <= 1;
register_Wdata_21 <= neuron_out_temp[cout_sumFF2store+1];
                                        register_Wadd_22 <= REGaddout2 + cout_sumFF2store; register_Wen_22 <= 1;
register_Wdata_22 <= neuron_out_temp[cout_sumFF2store+1];
                                        register_Wadd_23 <= REGaddout2 + cout_sumFF2store; register_Wen_23 <= 1;
register_Wdata_23 <= neuron_out_temp[cout_sumFF2store+1];
                                        register_Wadd_24 <= REGaddout2 + cout_sumFF2store; register_Wen_24 <= 1;
register_Wdata_24 <= neuron_out_temp[cout_sumFF2store+1];
                                        register_Wadd_25 <= REGaddout2 + cout_sumFF2store; register_Wen_25 <= 1;
register_Wdata_25 <= neuron_out_temp[cout_sumFF2store+1];
                                        register_Wadd_26 <= REGaddout2 + cout_sumFF2store; register_Wen_26 <= 1;
register_Wdata_26 <= neuron_out_temp[cout_sumFF2store+1];
                                        register_Wadd_27 <= REGaddout2 + cout_sumFF2store; register_Wen_27 <= 1;
register_Wdata_27 <= neuron_out_temp[cout_sumFF2store+1];
                                        register_Wadd_28 <= REGaddout2 + cout_sumFF2store; register_Wen_28 <= 1;
register_Wdata_28 <= neuron_out_temp[cout_sumFF2store+1];
                                        register_Wadd_29 <= REGaddout2 + cout_sumFF2store; register_Wen_29 <= 1;
register_Wdata_29 <= neuron_out_temp[cout_sumFF2store+1];
                                        register_Wadd_30 <= REGaddout2 + cout_sumFF2store; register_Wen_30 <= 1;
register_Wdata_30 <= neuron_out_temp[cout_sumFF2store+1];
                                    end
                                else if (cout_cntFF2store == 7)
                                    begin
                                        register_Wen_1 <= 0;
                                        register_Wen_2 <= 0;
                                        register_Wen_3 <= 0;
                                        register_Wen_4 <= 0;
                                        register_Wen_5 <= 0;
                                        register_Wen_6 <= 0;
                                        register_Wen_7 <= 0;
                                        register_Wen_8 <= 0;
                                        register_Wen_9 <= 0;
                                        register_Wen_10 <= 0;
                                        register_Wen_11 <= 0;
                                        register_Wen_12 <= 0;
                                        register_Wen_13 <= 0;
                                        register_Wen_14 <= 0;
                                        register_Wen_15 <= 0;
                                        register_Wen_16 <= 0;
                                        register_Wen_17 <= 0;
                                        register_Wen_18 <= 0;
                                        register_Wen_19 <= 0;
                                        register_Wen_20 <= 0;
                                        register_Wen_21 <= 0;
                                        register_Wen_22 <= 0;
                                        register_Wen_23 <= 0;
                                        register_Wen_24 <= 0;
                                        register_Wen_25 <= 0;
                                        register_Wen_26 <= 0;
                                        register_Wen_27 <= 0;
                                        register_Wen_28 <= 0;
                                        register_Wen_29 <= 0;
                                        register_Wen_30 <= 0;
                                    end
                        end
                9:      begin
                                if (cout_cntBPerrinitial == 1)
                                    begin
```

```
                                        register_Radd_A_1 <= REGtarget + 0; register_Radd_B_1 <= REGaddout2 + 0;
register_Ren_A_1 <= 1; register_Ren_B_1 <= 1;
                                        register_Radd_A_2 <= REGtarget + 1; register_Radd_B_2 <= REGaddout2 + 1;
register_Ren_A_2 <= 1; register_Ren_B_2 <= 1;
                                        register_Radd_A_3 <= REGtarget + 2; register_Radd_B_3 <= REGaddout2 + 2;
register_Ren_A_3 <= 1; register_Ren_B_3 <= 1;
                                        register_Radd_A_4 <= REGtarget + 3; register_Radd_B_4 <= REGaddout2 + 3;
register_Ren_A_4 <= 1; register_Ren_B_4 <= 1;
                                        register_Radd_A_5 <= REGtarget + 4; register_Radd_B_5 <= REGaddout2 + 4;
register_Ren_A_5 <= 1; register_Ren_B_5 <= 1;
                                        register_Radd_A_6 <= REGtarget + 5; register_Radd_B_6 <= REGaddout2 + 5;
register_Ren_A_6 <= 1; register_Ren_B_6 <= 1;
                                        register_Radd_A_7 <= REGtarget + 6; register_Radd_B_7 <= REGaddout2 + 6;
register_Ren_A_7 <= 1; register_Ren_B_7 <= 1;
                                        register_Radd_A_8 <= REGtarget + 7; register_Radd_B_8 <= REGaddout2 + 7;
register_Ren_A_8 <= 1; register_Ren_B_8 <= 1;
                                        register_Radd_A_9 <= REGtarget + 8; register_Radd_B_9 <= REGaddout2 + 8;
register_Ren_A_9 <= 1; register_Ren_B_9 <= 1;
                                        register_Radd_A_10 <= REGtarget + 9; register_Radd_B_10 <= REGaddout2 + 9;
register_Ren_A_10 <= 1; register_Ren_B_10 <= 1;
                                        register_Radd_A_11 <= REGtarget + 10; register_Radd_B_11 <= REGaddout2 + 10;
register_Ren_A_11 <= 1; register_Ren_B_11 <= 1;
                                        register_Radd_A_12 <= REGtarget + 11; register_Radd_B_12 <= REGaddout2 + 11;
register_Ren_A_12 <= 1; register_Ren_B_12 <= 1;

                                        BP_code <= 1;
                                end
                        else if (cout_cntBPerrinitial == 4)
                                begin
                                        start <= 1;
                                end
                        else if (cout_cntBPerrinitial == 5)
                                begin
                                        start <= 0;
                                end
                        else if (cout_cntBPerrinitial == 13)
                                begin
                                        register_Ren_A_1 <= 0; register_Ren_B_1 <= 0;
                                        register_Ren_A_2 <= 0; register_Ren_B_2 <= 0;
                                        register_Ren_A_3 <= 0; register_Ren_B_3 <= 0;
                                        register_Ren_A_4 <= 0; register_Ren_B_4 <= 0;
                                        register_Ren_A_5 <= 0; register_Ren_B_5 <= 0;
                                        register_Ren_A_6 <= 0; register_Ren_B_6 <= 0;
                                        register_Ren_A_7 <= 0; register_Ren_B_7 <= 0;
                                        register_Ren_A_8 <= 0; register_Ren_B_8 <= 0;
                                        register_Ren_A_9 <= 0; register_Ren_B_9 <= 0;
                                        register_Ren_A_10 <= 0; register_Ren_B_10 <= 0;
                                        register_Ren_A_11 <= 0; register_Ren_B_11 <= 0;
                                        register_Ren_A_12 <= 0; register_Ren_B_12 <= 0;
                                end
                        else if (cout_cntBPerrinitial == 14)
                                begin
                                        neuron_out_temp[1] <= neuron_out_1;
                                        neuron_out_temp[2] <= neuron_out_2;
                                        neuron_out_temp[3] <= neuron_out_3;
                                        neuron_out_temp[4] <= neuron_out_4;
                                        neuron_out_temp[5] <= neuron_out_5;
                                        neuron_out_temp[6] <= neuron_out_6;
                                        neuron_out_temp[7] <= neuron_out_7;
                                        neuron_out_temp[8] <= neuron_out_8;
                                        neuron_out_temp[9] <= neuron_out_9;
                                        neuron_out_temp[10] <= neuron_out_10;
                                        neuron_out_temp[11] <= neuron_out_11;
                                        neuron_out_temp[12] <= neuron_out_12;
                                end
                end
        10:begin
                        if (cout_cntBPerrstore == 1)
                                begin
```

```
                                        register_Wadd_1 <= REGerror + cout_sumBPerrstore; register_Wen_1 <= 1;
register_Wdata_1 <= neuron_out_temp[cout_sumBPerrstore+1];
                                        register_Wadd_2 <= REGerror + cout_sumBPerrstore; register_Wen_2 <= 1;
register_Wdata_2 <= neuron_out_temp[cout_sumBPerrstore+1];
                                        register_Wadd_3 <= REGerror + cout_sumBPerrstore; register_Wen_3 <= 1;
register_Wdata_3 <= neuron_out_temp[cout_sumBPerrstore+1];
                                        register_Wadd_4 <= REGerror + cout_sumBPerrstore; register_Wen_4 <= 1;
register_Wdata_4 <= neuron_out_temp[cout_sumBPerrstore+1];
                                        register_Wadd_5 <= REGerror + cout_sumBPerrstore; register_Wen_5 <= 1;
register_Wdata_5 <= neuron_out_temp[cout_sumBPerrstore+1];
                                        register_Wadd_6 <= REGerror + cout_sumBPerrstore; register_Wen_6 <= 1;
register_Wdata_6 <= neuron_out_temp[cout_sumBPerrstore+1];
                                        register_Wadd_7 <= REGerror + cout_sumBPerrstore; register_Wen_7 <= 1;
register_Wdata_7 <= neuron_out_temp[cout_sumBPerrstore+1];
                                        register_Wadd_8 <= REGerror + cout_sumBPerrstore; register_Wen_8 <= 1;
register_Wdata_8 <= neuron_out_temp[cout_sumBPerrstore+1];
                                        register_Wadd_9 <= REGerror + cout_sumBPerrstore; register_Wen_9 <= 1;
register_Wdata_9 <= neuron_out_temp[cout_sumBPerrstore+1];
                                        register_Wadd_10 <= REGerror + cout_sumBPerrstore; register_Wen_10 <= 1;
register_Wdata_10 <= neuron_out_temp[cout_sumBPerrstore+1];
                                        register_Wadd_11 <= REGerror + cout_sumBPerrstore; register_Wen_11 <= 1;
register_Wdata_11 <= neuron_out_temp[cout_sumBPerrstore+1];
                                        register_Wadd_12 <= REGerror + cout_sumBPerrstore; register_Wen_12 <= 1;
register_Wdata_12 <= neuron_out_temp[cout_sumBPerrstore+1];
                                        register_Wadd_13 <= REGerror + cout_sumBPerrstore; register_Wen_13 <= 1;
register_Wdata_13 <= neuron_out_temp[cout_sumBPerrstore+1];
                                        register_Wadd_14 <= REGerror + cout_sumBPerrstore; register_Wen_14 <= 1;
register_Wdata_14 <= neuron_out_temp[cout_sumBPerrstore+1];
                                        register_Wadd_15 <= REGerror + cout_sumBPerrstore; register_Wen_15 <= 1;
register_Wdata_15 <= neuron_out_temp[cout_sumBPerrstore+1];
                                        register_Wadd_16 <= REGerror + cout_sumBPerrstore; register_Wen_16 <= 1;
register_Wdata_16 <= neuron_out_temp[cout_sumBPerrstore+1];
                                        register_Wadd_17 <= REGerror + cout_sumBPerrstore; register_Wen_17 <= 1;
register_Wdata_17 <= neuron_out_temp[cout_sumBPerrstore+1];
                                        register_Wadd_18 <= REGerror + cout_sumBPerrstore; register_Wen_18 <= 1;
register_Wdata_18 <= neuron_out_temp[cout_sumBPerrstore+1];
                                        register_Wadd_19 <= REGerror + cout_sumBPerrstore; register_Wen_19 <= 1;
register_Wdata_19 <= neuron_out_temp[cout_sumBPerrstore+1];
                                        register_Wadd_20 <= REGerror + cout_sumBPerrstore; register_Wen_20 <= 1;
register_Wdata_20 <= neuron_out_temp[cout_sumBPerrstore+1];
                                        register_Wadd_21 <= REGerror + cout_sumBPerrstore; register_Wen_21 <= 1;
register_Wdata_21 <= neuron_out_temp[cout_sumBPerrstore+1];
                                        register_Wadd_22 <= REGerror + cout_sumBPerrstore; register_Wen_22 <= 1;
register_Wdata_22 <= neuron_out_temp[cout_sumBPerrstore+1];
                                        register_Wadd_23 <= REGerror + cout_sumBPerrstore; register_Wen_23 <= 1;
register_Wdata_23 <= neuron_out_temp[cout_sumBPerrstore+1];
                                        register_Wadd_24 <= REGerror + cout_sumBPerrstore; register_Wen_24 <= 1;
register_Wdata_24 <= neuron_out_temp[cout_sumBPerrstore+1];
                                        register_Wadd_25 <= REGerror + cout_sumBPerrstore; register_Wen_25 <= 1;
register_Wdata_25 <= neuron_out_temp[cout_sumBPerrstore+1];
                                        register_Wadd_26 <= REGerror + cout_sumBPerrstore; register_Wen_26 <= 1;
register_Wdata_26 <= neuron_out_temp[cout_sumBPerrstore+1];
                                        register_Wadd_27 <= REGerror + cout_sumBPerrstore; register_Wen_27 <= 1;
register_Wdata_27 <= neuron_out_temp[cout_sumBPerrstore+1];
                                        register_Wadd_28 <= REGerror + cout_sumBPerrstore; register_Wen_28 <= 1;
register_Wdata_28 <= neuron_out_temp[cout_sumBPerrstore+1];
                                        register_Wadd_29 <= REGerror + cout_sumBPerrstore; register_Wen_29 <= 1;
register_Wdata_29 <= neuron_out_temp[cout_sumBPerrstore+1];
                                        register_Wadd_30 <= REGerror + cout_sumBPerrstore; register_Wen_30 <= 1;
register_Wdata_30 <= neuron_out_temp[cout_sumBPerrstore+1];
                                    end
                                else if (cout_cntBPerrstore == 5)
                                    begin
                                        register_Wen_1 <= 0;
                                        register_Wen_2 <= 0;
                                        register_Wen_3 <= 0;
                                        register_Wen_4 <= 0;
                                        register_Wen_5 <= 0;
                                        register_Wen_6 <= 0;
                                        register_Wen_7 <= 0;
```

```
                                        register_Wen_8 <= 0;
                                        register_Wen_9 <= 0;
                                        register_Wen_10 <= 0;
                                        register_Wen_11 <= 0;
                                        register_Wen_12 <= 0;
                                        register_Wen_13 <= 0;
                                        register_Wen_14 <= 0;
                                        register_Wen_15 <= 0;
                                        register_Wen_16 <= 0;
                                        register_Wen_17 <= 0;
                                        register_Wen_18 <= 0;
                                        register_Wen_19 <= 0;
                                        register_Wen_20 <= 0;
                                        register_Wen_21 <= 0;
                                        register_Wen_22 <= 0;
                                        register_Wen_23 <= 0;
                                        register_Wen_24 <= 0;
                                        register_Wen_25 <= 0;
                                        register_Wen_26 <= 0;
                                        register_Wen_27 <= 0;
                                        register_Wen_28 <= 0;
                                        register_Wen_29 <= 0;
                                        register_Wen_30 <= 0;
                                end
                end
        11:begin
                        if (cout_cntBPerradd == 1)
                                begin
                                        register_Radd_A_1 <= REGerrorcheck; register_Radd_B_1 <= REGerror +
cout_sumBPerradd; register_Ren_A_1 <= 1; register_Ren_B_1 <= 1;
                                        BP_code <= 2;
                                end
                        else if (cout_cntBPerradd == 4)
                                begin
                                        start       <= 1;
                                end
                        else if (cout_cntBPerradd == 5)
                                begin
                                        start <= 0;
                                end
                        else if (cout_cntBPerradd == 13)
                                begin
                                        register_Ren_A_1 <= 0;
                                        register_Ren_B_1 <= 0;
                                end
                        else if (cout_cntBPerradd == 14)
                                begin
                                        neuron_out_temp[1] <= neuron_out_1;
                                end
                        else if (cout_cntBPerradd == 15)
                                begin
                                        register_Wadd_1 <= REGerrorcheck; register_Wen_1 <= 1; register_Wdata_1 <=
neuron_out_temp[1];
                                        if (cout_sumCheck == 3)
                                                begin
                                                        errout <= neuron_out_temp[1];
                                                end
                                        else
                                                begin
                                                        errout <= errout;
                                                end
                                        if (neuron_out_temp[1] < ERROR)
                                                begin
                                                        stop <= 1;
                                                end
                                        else
                                                begin
                                                        stop <= 0;
                                                end
                                end
```

```
                        else if (cout_cntBPerradd == 16)
                                begin
                                        register_Wen_1 <= 0;
                                end
                end
        12:begin
                        if (cout_cntBPdeloinitial == 1)
                                begin
                                        register_Radd_A_1 <= REGtarget + 0; register_Radd_B_1 <= REGaddout2 + 0;
register_Ren_A_1 <= 1; register_Ren_B_1 <= 1;
                                        register_Radd_A_2 <= REGtarget + 1; register_Radd_B_2 <= REGaddout2 + 1;
register_Ren_A_2 <= 1; register_Ren_B_2 <= 1;
                                        register_Radd_A_3 <= REGtarget + 2; register_Radd_B_3 <= REGaddout2 + 2;
register_Ren_A_3 <= 1; register_Ren_B_3 <= 1;
                                        register_Radd_A_4 <= REGtarget + 3; register_Radd_B_4 <= REGaddout2 + 3;
register_Ren_A_4 <= 1; register_Ren_B_4 <= 1;
                                        register_Radd_A_5 <= REGtarget + 4; register_Radd_B_5 <= REGaddout2 + 4;
register_Ren_A_5 <= 1; register_Ren_B_5 <= 1;
                                        register_Radd_A_6 <= REGtarget + 5; register_Radd_B_6 <= REGaddout2 + 5;
register_Ren_A_6 <= 1; register_Ren_B_6 <= 1;
                                        register_Radd_A_7 <= REGtarget + 6; register_Radd_B_7 <= REGaddout2 + 6;
register_Ren_A_7 <= 1; register_Ren_B_7 <= 1;
                                        register_Radd_A_8 <= REGtarget + 7; register_Radd_B_8 <= REGaddout2 + 7;
register_Ren_A_8 <= 1; register_Ren_B_8 <= 1;
                                        register_Radd_A_9 <= REGtarget + 8; register_Radd_B_9 <= REGaddout2 + 8;
register_Ren_A_9 <= 1; register_Ren_B_9 <= 1;
                                        register_Radd_A_10 <= REGtarget + 9; register_Radd_B_10 <= REGaddout2 + 9;
register_Ren_A_10 <= 1; register_Ren_B_10 <= 1;
                                        register_Radd_A_11 <= REGtarget + 10; register_Radd_B_11 <= REGaddout2 + 10;
register_Ren_A_11 <= 1; register_Ren_B_11 <= 1;
                                        register_Radd_A_12 <= REGtarget + 11; register_Radd_B_12 <= REGaddout2 + 11;
register_Ren_A_12 <= 1; register_Ren_B_12 <= 1;

                                        BP_code <= 3;
                                end
                        else if (cout_cntBPdeloinitial == 4)
                                begin
                                        start <= 1;
                                end
                        else if (cout_cntBPdeloinitial == 5)
                                begin
                                        start <= 0;
                                end
                        else if (cout_cntBPdeloinitial == 13)
                                begin
                                        register_Ren_A_1 <= 0; register_Ren_B_1 <= 0;
                                        register_Ren_A_2 <= 0; register_Ren_B_2 <= 0;
                                        register_Ren_A_3 <= 0; register_Ren_B_3 <= 0;
                                        register_Ren_A_4 <= 0; register_Ren_B_4 <= 0;
                                        register_Ren_A_5 <= 0; register_Ren_B_5 <= 0;
                                        register_Ren_A_6 <= 0; register_Ren_B_6 <= 0;
                                        register_Ren_A_7 <= 0; register_Ren_B_7 <= 0;
                                        register_Ren_A_8 <= 0; register_Ren_B_8 <= 0;
                                        register_Ren_A_9 <= 0; register_Ren_B_9 <= 0;
                                        register_Ren_A_10 <= 0; register_Ren_B_10 <= 0;
                                        register_Ren_A_11 <= 0; register_Ren_B_11 <= 0;
                                        register_Ren_A_12 <= 0; register_Ren_B_12 <= 0;
                                end
                        else if (cout_cntBPdeloinitial == 14)
                                begin
                                        neuron_out_temp[1] <= neuron_out_1;
                                        neuron_out_temp[2] <= neuron_out_2;
                                        neuron_out_temp[3] <= neuron_out_3;
                                        neuron_out_temp[4] <= neuron_out_4;
                                        neuron_out_temp[5] <= neuron_out_5;
                                        neuron_out_temp[6] <= neuron_out_6;
                                        neuron_out_temp[7] <= neuron_out_7;
                                        neuron_out_temp[8] <= neuron_out_8;
                                        neuron_out_temp[9] <= neuron_out_9;
                                        neuron_out_temp[10] <= neuron_out_10;
```

```
                        neuron_out_temp[11] <= neuron_out_11;
                        neuron_out_temp[12] <= neuron_out_12;
                    end
                else if (cout_cntBPdeloinitial == 15)
                    begin
                        register_Wadd_1 <= REGerror; register_Wen_1 <= 1; register_Wdata_1 <=
neuron_out_temp[1];
                        register_Wadd_2 <= REGerror; register_Wen_2 <= 1; register_Wdata_2 <=
neuron_out_temp[2];
                        register_Wadd_3 <= REGerror; register_Wen_3 <= 1; register_Wdata_3 <=
neuron_out_temp[3];
                        register_Wadd_4 <= REGerror; register_Wen_4 <= 1; register_Wdata_4 <=
neuron_out_temp[4];
                        register_Wadd_5 <= REGerror; register_Wen_5 <= 1; register_Wdata_5 <=
neuron_out_temp[5];
                        register_Wadd_6 <= REGerror; register_Wen_6 <= 1; register_Wdata_6 <=
neuron_out_temp[6];
                        register_Wadd_7 <= REGerror; register_Wen_7 <= 1; register_Wdata_7 <=
neuron_out_temp[7];
                        register_Wadd_8 <= REGerror; register_Wen_8 <= 1; register_Wdata_8 <=
neuron_out_temp[8];
                        register_Wadd_9 <= REGerror; register_Wen_9 <= 1; register_Wdata_9 <=
neuron_out_temp[9];
                        register_Wadd_10 <= REGerror; register_Wen_10 <= 1; register_Wdata_10 <=
neuron_out_temp[10];
                        register_Wadd_11 <= REGerror; register_Wen_11 <= 1; register_Wdata_11 <=
neuron_out_temp[11];
                        register_Wadd_12 <= REGerror; register_Wen_12 <= 1; register_Wdata_12 <=
neuron_out_temp[12];
                    end
                else if (cout_cntBPdeloinitial == 17)
                    begin
                        register_Wen_1 <= 0;
                        register_Wen_2 <= 0;
                        register_Wen_3 <= 0;
                        register_Wen_4 <= 0;
                        register_Wen_5 <= 0;
                        register_Wen_6 <= 0;
                        register_Wen_7 <= 0;
                        register_Wen_8 <= 0;
                        register_Wen_9 <= 0;
                        register_Wen_10 <= 0;
                        register_Wen_11 <= 0;
                        register_Wen_12 <= 0;
                    end
                else if (cout_cntBPdeloinitial == 18)
                    begin
                        register_Radd_A_1 <= REGerror; register_Radd_B_1 <= REGaddout2 + 0;
register_Ren_A_1 <= 1; register_Ren_B_1 <= 1;
                        register_Radd_A_2 <= REGerror; register_Radd_B_2 <= REGaddout2 + 1;
register_Ren_A_2 <= 1; register_Ren_B_2 <= 1;
                        register_Radd_A_3 <= REGerror; register_Radd_B_3 <= REGaddout2 + 2;
register_Ren_A_3 <= 1; register_Ren_B_3 <= 1;
                        register_Radd_A_4 <= REGerror; register_Radd_B_4 <= REGaddout2 + 3;
register_Ren_A_4 <= 1; register_Ren_B_4 <= 1;
                        register_Radd_A_5 <= REGerror; register_Radd_B_5 <= REGaddout2 + 4;
register_Ren_A_5 <= 1; register_Ren_B_5 <= 1;
                        register_Radd_A_6 <= REGerror; register_Radd_B_6 <= REGaddout2 + 5;
register_Ren_A_6 <= 1; register_Ren_B_6 <= 1;
                        register_Radd_A_7 <= REGerror; register_Radd_B_7 <= REGaddout2 + 6;
register_Ren_A_7 <= 1; register_Ren_B_7 <= 1;
                        register_Radd_A_8 <= REGerror; register_Radd_B_8 <= REGaddout2 + 7;
register_Ren_A_8 <= 1; register_Ren_B_8 <= 1;
                        register_Radd_A_9 <= REGerror; register_Radd_B_9 <= REGaddout2 + 8;
register_Ren_A_9 <= 1; register_Ren_B_9 <= 1;
                        register_Radd_A_10 <= REGerror; register_Radd_B_10 <= REGaddout2 + 9;
register_Ren_A_10 <= 1; register_Ren_B_10 <= 1;
                        register_Radd_A_11 <= REGerror; register_Radd_B_11 <= REGaddout2 + 10;
register_Ren_A_11 <= 1; register_Ren_B_11 <= 1;
```

```
                                        register_Radd_A_12 <= REGerror; register_Radd_B_12 <= REGaddout2 + 11;
register_Ren_A_12 <= 1; register_Ren_B_12 <= 1;

                                        BP_code <= 4;
                                end
                        else if (cout_cntBPdeloinitial == 21)
                                begin
                                        start <= 1;
                                end
                        else if (cout_cntBPdeloinitial == 22)
                                begin
                                        start <= 0;
                                end
                        else if (cout_cntBPdeloinitial == 30)
                                begin
                                        register_Ren_A_1 <= 0; register_Ren_B_1 <= 0;
                                        register_Ren_A_2 <= 0; register_Ren_B_2 <= 0;
                                        register_Ren_A_3 <= 0; register_Ren_B_3 <= 0;
                                        register_Ren_A_4 <= 0; register_Ren_B_4 <= 0;
                                        register_Ren_A_5 <= 0; register_Ren_B_5 <= 0;
                                        register_Ren_A_6 <= 0; register_Ren_B_6 <= 0;
                                        register_Ren_A_7 <= 0; register_Ren_B_7 <= 0;
                                        register_Ren_A_8 <= 0; register_Ren_B_8 <= 0;
                                        register_Ren_A_9 <= 0; register_Ren_B_9 <= 0;
                                        register_Ren_A_10 <= 0; register_Ren_B_10 <= 0;
                                        register_Ren_A_11 <= 0; register_Ren_B_11 <= 0;
                                        register_Ren_A_12 <= 0; register_Ren_B_12 <= 0;
                                end
                end
        13:begin
                        if (cout_cntBPdelostore == 1)
                                begin
                                        neuron_out_temp[1] <= neuron_out_1;
                                        neuron_out_temp[2] <= neuron_out_2;
                                        neuron_out_temp[3] <= neuron_out_3;
                                        neuron_out_temp[4] <= neuron_out_4;
                                        neuron_out_temp[5] <= neuron_out_5;
                                        neuron_out_temp[6] <= neuron_out_6;
                                        neuron_out_temp[7] <= neuron_out_7;
                                        neuron_out_temp[8] <= neuron_out_8;
                                        neuron_out_temp[9] <= neuron_out_9;
                                        neuron_out_temp[10] <= neuron_out_10;
                                        neuron_out_temp[11] <= neuron_out_11;
                                        neuron_out_temp[12] <= neuron_out_12;
                                end
                        else if (cout_cntBPdelostore == 2)
                                begin
                                        register_Wadd_1 <= REGdelo + cout_sumBPdelostore; register_Wen_1 <= 1;
register_Wdata_1 <= neuron_out_temp[cout_sumBPdelostore+1];
                                        register_Wadd_2 <= REGdelo + cout_sumBPdelostore; register_Wen_2 <= 1;
register_Wdata_2 <= neuron_out_temp[cout_sumBPdelostore+1];
                                        register_Wadd_3 <= REGdelo + cout_sumBPdelostore; register_Wen_3 <= 1;
register_Wdata_3 <= neuron_out_temp[cout_sumBPdelostore+1];
                                        register_Wadd_4 <= REGdelo + cout_sumBPdelostore; register_Wen_4 <= 1;
register_Wdata_4 <= neuron_out_temp[cout_sumBPdelostore+1];
                                        register_Wadd_5 <= REGdelo + cout_sumBPdelostore; register_Wen_5 <= 1;
register_Wdata_5 <= neuron_out_temp[cout_sumBPdelostore+1];
                                        register_Wadd_6 <= REGdelo + cout_sumBPdelostore; register_Wen_6 <= 1;
register_Wdata_6 <= neuron_out_temp[cout_sumBPdelostore+1];
                                        register_Wadd_7 <= REGdelo + cout_sumBPdelostore; register_Wen_7 <= 1;
register_Wdata_7 <= neuron_out_temp[cout_sumBPdelostore+1];
                                        register_Wadd_8 <= REGdelo + cout_sumBPdelostore; register_Wen_8 <= 1;
register_Wdata_8 <= neuron_out_temp[cout_sumBPdelostore+1];
                                        register_Wadd_9 <= REGdelo + cout_sumBPdelostore; register_Wen_9 <= 1;
register_Wdata_9 <= neuron_out_temp[cout_sumBPdelostore+1];
                                        register_Wadd_10 <= REGdelo + cout_sumBPdelostore; register_Wen_10 <= 1;
register_Wdata_10 <= neuron_out_temp[cout_sumBPdelostore+1];
                                        register_Wadd_11 <= REGdelo + cout_sumBPdelostore; register_Wen_11 <= 1;
register_Wdata_11 <= neuron_out_temp[cout_sumBPdelostore+1];
```

```
                                        register_Wadd_12 <= REGdelo + cout_sumBPdelostore; register_Wen_12 <= 1;
register_Wdata_12 <= neuron_out_temp[cout_sumBPdelostore+1];
                                        register_Wadd_13 <= REGdelo + cout_sumBPdelostore; register_Wen_13 <= 1;
register_Wdata_13 <= neuron_out_temp[cout_sumBPdelostore+1];
                                        register_Wadd_14 <= REGdelo + cout_sumBPdelostore; register_Wen_14 <= 1;
register_Wdata_14 <= neuron_out_temp[cout_sumBPdelostore+1];
                                        register_Wadd_15 <= REGdelo + cout_sumBPdelostore; register_Wen_15 <= 1;
register_Wdata_15 <= neuron_out_temp[cout_sumBPdelostore+1];
                                        register_Wadd_16 <= REGdelo + cout_sumBPdelostore; register_Wen_16 <= 1;
register_Wdata_16 <= neuron_out_temp[cout_sumBPdelostore+1];
                                        register_Wadd_17 <= REGdelo + cout_sumBPdelostore; register_Wen_17 <= 1;
register_Wdata_17 <= neuron_out_temp[cout_sumBPdelostore+1];
                                        register_Wadd_18 <= REGdelo + cout_sumBPdelostore; register_Wen_18 <= 1;
register_Wdata_18 <= neuron_out_temp[cout_sumBPdelostore+1];
                                        register_Wadd_19 <= REGdelo + cout_sumBPdelostore; register_Wen_19 <= 1;
register_Wdata_19 <= neuron_out_temp[cout_sumBPdelostore+1];
                                        register_Wadd_20 <= REGdelo + cout_sumBPdelostore; register_Wen_20 <= 1;
register_Wdata_20 <= neuron_out_temp[cout_sumBPdelostore+1];
                                        register_Wadd_21 <= REGdelo + cout_sumBPdelostore; register_Wen_21 <= 1;
register_Wdata_21 <= neuron_out_temp[cout_sumBPdelostore+1];
                                        register_Wadd_22 <= REGdelo + cout_sumBPdelostore; register_Wen_22 <= 1;
register_Wdata_22 <= neuron_out_temp[cout_sumBPdelostore+1];
                                        register_Wadd_23 <= REGdelo + cout_sumBPdelostore; register_Wen_23 <= 1;
register_Wdata_23 <= neuron_out_temp[cout_sumBPdelostore+1];
                                        register_Wadd_24 <= REGdelo + cout_sumBPdelostore; register_Wen_24 <= 1;
register_Wdata_24 <= neuron_out_temp[cout_sumBPdelostore+1];
                                        register_Wadd_25 <= REGdelo + cout_sumBPdelostore; register_Wen_25 <= 1;
register_Wdata_25 <= neuron_out_temp[cout_sumBPdelostore+1];
                                        register_Wadd_26 <= REGdelo + cout_sumBPdelostore; register_Wen_26 <= 1;
register_Wdata_26 <= neuron_out_temp[cout_sumBPdelostore+1];
                                        register_Wadd_27 <= REGdelo + cout_sumBPdelostore; register_Wen_27 <= 1;
register_Wdata_27 <= neuron_out_temp[cout_sumBPdelostore+1];
                                        register_Wadd_28 <= REGdelo + cout_sumBPdelostore; register_Wen_28 <= 1;
register_Wdata_28 <= neuron_out_temp[cout_sumBPdelostore+1];
                                        register_Wadd_29 <= REGdelo + cout_sumBPdelostore; register_Wen_29 <= 1;
register_Wdata_29 <= neuron_out_temp[cout_sumBPdelostore+1];
                                        register_Wadd_30 <= REGdelo + cout_sumBPdelostore; register_Wen_30 <= 1;
register_Wdata_30 <= neuron_out_temp[cout_sumBPdelostore+1];
                                    end
                                else if (cout_cntBPdelostore == 4)
                                    begin
                                        register_Wen_1 <= 0;
                                        register_Wen_2 <= 0;
                                        register_Wen_3 <= 0;
                                        register_Wen_4 <= 0;
                                        register_Wen_5 <= 0;
                                        register_Wen_6 <= 0;
                                        register_Wen_7 <= 0;
                                        register_Wen_8 <= 0;
                                        register_Wen_9 <= 0;
                                        register_Wen_10 <= 0;
                                        register_Wen_11 <= 0;
                                        register_Wen_12 <= 0;
                                        register_Wen_13 <= 0;
                                        register_Wen_14 <= 0;
                                        register_Wen_15 <= 0;
                                        register_Wen_16 <= 0;
                                        register_Wen_17 <= 0;
                                        register_Wen_18 <= 0;
                                        register_Wen_19 <= 0;
                                        register_Wen_20 <= 0;
                                        register_Wen_21 <= 0;
                                        register_Wen_22 <= 0;
                                        register_Wen_23 <= 0;
                                        register_Wen_24 <= 0;
                                        register_Wen_25 <= 0;
                                        register_Wen_26 <= 0;
                                        register_Wen_27 <= 0;
                                        register_Wen_28 <= 0;
                                        register_Wen_29 <= 0;
```

```
                                        register_Wen_30 <= 0;
                                end
                        end
                14:begin
                                if (cout_cntBPsumdow == 1)
                                        begin
                                                register_Radd_A_1 <= REGaddinweight2 + cout_sumBPsumdow+1; register_Radd_B_1
<= REGdelo + 0; register_Ren_A_1 <= 1; register_Ren_B_1 <= 1;
                                                register_Radd_A_2 <= REGaddinweight2 + cout_sumBPsumdow+1; register_Radd_B_2
<= REGdelo + 1; register_Ren_A_2 <= 1; register_Ren_B_2 <= 1;
                                                register_Radd_A_3 <= REGaddinweight2 + cout_sumBPsumdow+1; register_Radd_B_3
<= REGdelo + 2; register_Ren_A_3 <= 1; register_Ren_B_3 <= 1;
                                                register_Radd_A_4 <= REGaddinweight2 + cout_sumBPsumdow+1; register_Radd_B_4
<= REGdelo + 3; register_Ren_A_4 <= 1; register_Ren_B_4 <= 1;
                                                register_Radd_A_5 <= REGaddinweight2 + cout_sumBPsumdow+1; register_Radd_B_5
<= REGdelo + 4; register_Ren_A_5 <= 1; register_Ren_B_5 <= 1;
                                                register_Radd_A_6 <= REGaddinweight2 + cout_sumBPsumdow+1; register_Radd_B_6
<= REGdelo + 5; register_Ren_A_6 <= 1; register_Ren_B_6 <= 1;
                                                register_Radd_A_7 <= REGaddinweight2 + cout_sumBPsumdow+1; register_Radd_B_7
<= REGdelo + 6; register_Ren_A_7 <= 1; register_Ren_B_7 <= 1;
                                                register_Radd_A_8 <= REGaddinweight2 + cout_sumBPsumdow+1; register_Radd_B_8
<= REGdelo + 7; register_Ren_A_8 <= 1; register_Ren_B_8 <= 1;
                                                register_Radd_A_9 <= REGaddinweight2 + cout_sumBPsumdow+1; register_Radd_B_9
<= REGdelo + 8; register_Ren_A_9 <= 1; register_Ren_B_9 <= 1;
                                                register_Radd_A_10      <=      REGaddinweight2      +      cout_sumBPsumdow+1;
register_Radd_B_10 <= REGdelo + 9; register_Ren_A_10 <= 1; register_Ren_B_10 <= 1;
                                                register_Radd_A_11      <=      REGaddinweight2      +      cout_sumBPsumdow+1;
register_Radd_B_11 <= REGdelo + 10; register_Ren_A_11 <= 1; register_Ren_B_11 <= 1;
                                                register_Radd_A_12      <=      REGaddinweight2      +      cout_sumBPsumdow+1;
register_Radd_B_12 <= REGdelo + 11; register_Ren_A_12 <= 1; register_Ren_B_12 <= 1;

                                                BP_code <= 6;
                                        end
                                else if (cout_cntBPsumdow == 4)
                                        begin
                                                start          <= 1;
                                        end
                                else if (cout_cntBPsumdow == 5)
                                        begin
                                                start <= 0;
                                        end
                                else if (cout_cntBPsumdow == 13)
                                        begin
                                                register_Ren_A_1 <= 0; register_Ren_B_1 <= 0;
                                                register_Ren_A_2 <= 0; register_Ren_B_2 <= 0;
                                                register_Ren_A_3 <= 0; register_Ren_B_3 <= 0;
                                                register_Ren_A_4 <= 0; register_Ren_B_4 <= 0;
                                                register_Ren_A_5 <= 0; register_Ren_B_5 <= 0;
                                                register_Ren_A_6 <= 0; register_Ren_B_6 <= 0;
                                                register_Ren_A_7 <= 0; register_Ren_B_7 <= 0;
                                                register_Ren_A_8 <= 0; register_Ren_B_8 <= 0;
                                                register_Ren_A_9 <= 0; register_Ren_B_9 <= 0;
                                                register_Ren_A_10 <= 0; register_Ren_B_10 <= 0;
                                                register_Ren_A_11 <= 0; register_Ren_B_11 <= 0;
                                                register_Ren_A_12 <= 0; register_Ren_B_12 <= 0;
                                        end
                                else if (cout_cntBPsumdow == 14)
                                        begin
                                                neuron_out_temp[1] <= neuron_out_1;
                                                neuron_out_temp[2] <= neuron_out_2;
                                                neuron_out_temp[3] <= neuron_out_3;
                                                neuron_out_temp[4] <= neuron_out_4;
                                                neuron_out_temp[5] <= neuron_out_5;
                                                neuron_out_temp[6] <= neuron_out_6;
                                                neuron_out_temp[7] <= neuron_out_7;
                                                neuron_out_temp[8] <= neuron_out_8;
                                                neuron_out_temp[9] <= neuron_out_9;
                                                neuron_out_temp[10] <= neuron_out_10;
                                                neuron_out_temp[11] <= neuron_out_11;
                                                neuron_out_temp[12] <= neuron_out_12;
```

```
                end
            else if (cout_cntBPsumdow == 15)
                begin
                    neuron_out_temp[0] <= neuron_out_temp[1]+neuron_out_temp[2]+neuron_out_temp[3]+neuron_out_temp[4]+neuron_out_temp[5]+neuron_out_temp[6]+neuron_out_temp[7]+neuron_out_temp[8]+neuron_out_temp[9]+neuron_out_temp[10]+neuron_out_temp[11]+neuron_out_temp[12];
                end
            else if (cout_cntBPsumdow == 25)
                begin
                    register_Wadd_1 <= REGerror + cout_sumBPsumdow; register_Wen_1 <= 1; register_Wdata_1 <= neuron_out_temp[0];
                    register_Wadd_2 <= REGerror + cout_sumBPsumdow; register_Wen_2 <= 1; register_Wdata_2 <= neuron_out_temp[0];
                    register_Wadd_3 <= REGerror + cout_sumBPsumdow; register_Wen_3 <= 1; register_Wdata_3 <= neuron_out_temp[0];
                    register_Wadd_4 <= REGerror + cout_sumBPsumdow; register_Wen_4 <= 1; register_Wdata_4 <= neuron_out_temp[0];
                    register_Wadd_5 <= REGerror + cout_sumBPsumdow; register_Wen_5 <= 1; register_Wdata_5 <= neuron_out_temp[0];
                    register_Wadd_6 <= REGerror + cout_sumBPsumdow; register_Wen_6 <= 1; register_Wdata_6 <= neuron_out_temp[0];
                    register_Wadd_7 <= REGerror + cout_sumBPsumdow; register_Wen_7 <= 1; register_Wdata_7 <= neuron_out_temp[0];
                    register_Wadd_8 <= REGerror + cout_sumBPsumdow; register_Wen_8 <= 1; register_Wdata_8 <= neuron_out_temp[0];
                    register_Wadd_9 <= REGerror + cout_sumBPsumdow; register_Wen_9 <= 1; register_Wdata_9 <= neuron_out_temp[0];
                    register_Wadd_10 <= REGerror + cout_sumBPsumdow; register_Wen_10 <= 1; register_Wdata_10 <= neuron_out_temp[0];
                    register_Wadd_11 <= REGerror + cout_sumBPsumdow; register_Wen_11 <= 1; register_Wdata_11 <= neuron_out_temp[0];
                    register_Wadd_12 <= REGerror + cout_sumBPsumdow; register_Wen_12 <= 1; register_Wdata_12 <= neuron_out_temp[0];
                    register_Wadd_13 <= REGerror + cout_sumBPsumdow; register_Wen_13 <= 1; register_Wdata_13 <= neuron_out_temp[0];
                    register_Wadd_14 <= REGerror + cout_sumBPsumdow; register_Wen_14 <= 1; register_Wdata_14 <= neuron_out_temp[0];
                    register_Wadd_15 <= REGerror + cout_sumBPsumdow; register_Wen_15 <= 1; register_Wdata_15 <= neuron_out_temp[0];
                    register_Wadd_16 <= REGerror + cout_sumBPsumdow; register_Wen_16 <= 1; register_Wdata_16 <= neuron_out_temp[0];
                    register_Wadd_17 <= REGerror + cout_sumBPsumdow; register_Wen_17 <= 1; register_Wdata_17 <= neuron_out_temp[0];
                    register_Wadd_18 <= REGerror + cout_sumBPsumdow; register_Wen_18 <= 1; register_Wdata_18 <= neuron_out_temp[0];
                    register_Wadd_19 <= REGerror + cout_sumBPsumdow; register_Wen_19 <= 1; register_Wdata_19 <= neuron_out_temp[0];
                    register_Wadd_20 <= REGerror + cout_sumBPsumdow; register_Wen_20 <= 1; register_Wdata_20 <= neuron_out_temp[0];
                    register_Wadd_21 <= REGerror + cout_sumBPsumdow; register_Wen_21 <= 1; register_Wdata_21 <= neuron_out_temp[0];
                    register_Wadd_22 <= REGerror + cout_sumBPsumdow; register_Wen_22 <= 1; register_Wdata_22 <= neuron_out_temp[0];
                    register_Wadd_23 <= REGerror + cout_sumBPsumdow; register_Wen_23 <= 1; register_Wdata_23 <= neuron_out_temp[0];
                    register_Wadd_24 <= REGerror + cout_sumBPsumdow; register_Wen_24 <= 1; register_Wdata_24 <= neuron_out_temp[0];
                    register_Wadd_25 <= REGerror + cout_sumBPsumdow; register_Wen_25 <= 1; register_Wdata_25 <= neuron_out_temp[0];
                    register_Wadd_26 <= REGerror + cout_sumBPsumdow; register_Wen_26 <= 1; register_Wdata_26 <= neuron_out_temp[0];
                    register_Wadd_27 <= REGerror + cout_sumBPsumdow; register_Wen_27 <= 1; register_Wdata_27 <= neuron_out_temp[0];
                    register_Wadd_28 <= REGerror + cout_sumBPsumdow; register_Wen_28 <= 1; register_Wdata_28 <= neuron_out_temp[0];
                    register_Wadd_29 <= REGerror + cout_sumBPsumdow; register_Wen_29 <= 1; register_Wdata_29 <= neuron_out_temp[0];
                    register_Wadd_30 <= REGerror + cout_sumBPsumdow; register_Wen_30 <= 1; register_Wdata_30 <= neuron_out_temp[0];
                end
```

```
                else if (cout_cntBPsumdow == 27)
                        begin
                                register_Wen_1 <= 0;
                                register_Wen_2 <= 0;
                                register_Wen_3 <= 0;
                                register_Wen_4 <= 0;
                                register_Wen_5 <= 0;
                                register_Wen_6 <= 0;
                                register_Wen_7 <= 0;
                                register_Wen_8 <= 0;
                                register_Wen_9 <= 0;
                                register_Wen_10 <= 0;
                                register_Wen_11 <= 0;
                                register_Wen_12 <= 0;
                                register_Wen_13 <= 0;
                                register_Wen_14 <= 0;
                                register_Wen_15 <= 0;
                                register_Wen_16 <= 0;
                                register_Wen_17 <= 0;
                                register_Wen_18 <= 0;
                                register_Wen_19 <= 0;
                                register_Wen_20 <= 0;
                                register_Wen_21 <= 0;
                                register_Wen_22 <= 0;
                                register_Wen_23 <= 0;
                                register_Wen_24 <= 0;
                                register_Wen_25 <= 0;
                                register_Wen_26 <= 0;
                                register_Wen_27 <= 0;
                                register_Wen_28 <= 0;
                                register_Wen_29 <= 0;
                                register_Wen_30 <= 0;
                        end
                end
        15:begin
                        if (cout_cntBPdelh == 1)
                                begin
                                        register_Radd_A_1 <= REGerror + 0; register_Radd_B_1 <= REGaddout1 + 0;
register_Ren_A_1 <= 1; register_Ren_B_1 <= 1;
                                        register_Radd_A_2 <= REGerror + 1; register_Radd_B_2 <= REGaddout1 + 1;
register_Ren_A_2 <= 1; register_Ren_B_2 <= 1;
                                        register_Radd_A_3 <= REGerror + 2; register_Radd_B_3 <= REGaddout1 + 2;
register_Ren_A_3 <= 1; register_Ren_B_3 <= 1;
                                        register_Radd_A_4 <= REGerror + 3; register_Radd_B_4 <= REGaddout1 + 3;
register_Ren_A_4 <= 1; register_Ren_B_4 <= 1;
                                        register_Radd_A_5 <= REGerror + 4; register_Radd_B_5 <= REGaddout1 + 4;
register_Ren_A_5 <= 1; register_Ren_B_5 <= 1;
                                        register_Radd_A_6 <= REGerror + 5; register_Radd_B_6 <= REGaddout1 + 5;
register_Ren_A_6 <= 1; register_Ren_B_6 <= 1;
                                        register_Radd_A_7 <= REGerror + 6; register_Radd_B_7 <= REGaddout1 + 6;
register_Ren_A_7 <= 1; register_Ren_B_7 <= 1;
                                        register_Radd_A_8 <= REGerror + 7; register_Radd_B_8 <= REGaddout1 + 7;
register_Ren_A_8 <= 1; register_Ren_B_8 <= 1;
                                        register_Radd_A_9 <= REGerror + 8; register_Radd_B_9 <= REGaddout1 + 8;
register_Ren_A_9 <= 1; register_Ren_B_9 <= 1;
                                        register_Radd_A_10 <= REGerror + 9; register_Radd_B_10 <= REGaddout1 + 9;
register_Ren_A_10 <= 1; register_Ren_B_10 <= 1;
                                        register_Radd_A_11 <= REGerror + 10; register_Radd_B_11 <= REGaddout1 + 10;
register_Ren_A_11 <= 1; register_Ren_B_11 <= 1;
                                        register_Radd_A_12 <= REGerror + 11; register_Radd_B_12 <= REGaddout1 + 11;
register_Ren_A_12 <= 1; register_Ren_B_12 <= 1;
                                        register_Radd_A_13 <= REGerror + 12; register_Radd_B_13 <= REGaddout1 + 12;
register_Ren_A_13 <= 1; register_Ren_B_13 <= 1;
                                        register_Radd_A_14 <= REGerror + 13; register_Radd_B_14 <= REGaddout1 + 13;
register_Ren_A_14 <= 1; register_Ren_B_14 <= 1;
                                        register_Radd_A_15 <= REGerror + 14; register_Radd_B_15 <= REGaddout1 + 14;
register_Ren_A_15 <= 1; register_Ren_B_15 <= 1;
                                        register_Radd_A_16 <= REGerror + 15; register_Radd_B_16 <= REGaddout1 + 15;
register_Ren_A_16 <= 1; register_Ren_B_16 <= 1;
```

```
                register_Radd_A_17 <= REGerror + 16; register_Radd_B_17 <= REGaddout1 + 16;
register_Ren_A_17 <= 1; register_Ren_B_17 <= 1;
                register_Radd_A_18 <= REGerror + 17; register_Radd_B_18 <= REGaddout1 + 17;
register_Ren_A_18 <= 1; register_Ren_B_18 <= 1;
                register_Radd_A_19 <= REGerror + 18; register_Radd_B_19 <= REGaddout1 + 18;
register_Ren_A_19 <= 1; register_Ren_B_19 <= 1;
                register_Radd_A_20 <= REGerror + 19; register_Radd_B_20 <= REGaddout1 + 19;
register_Ren_A_20 <= 1; register_Ren_B_20 <= 1;
                register_Radd_A_21 <= REGerror + 20; register_Radd_B_21 <= REGaddout1 + 20;
register_Ren_A_21 <= 1; register_Ren_B_21 <= 1;
                register_Radd_A_22 <= REGerror + 21; register_Radd_B_22 <= REGaddout1 + 21;
register_Ren_A_22 <= 1; register_Ren_B_22 <= 1;
                register_Radd_A_23 <= REGerror + 22; register_Radd_B_23 <= REGaddout1 + 22;
register_Ren_A_23 <= 1; register_Ren_B_23 <= 1;
                register_Radd_A_24 <= REGerror + 23; register_Radd_B_24 <= REGaddout1 + 23;
register_Ren_A_24 <= 1; register_Ren_B_24 <= 1;
                register_Radd_A_25 <= REGerror + 24; register_Radd_B_25 <= REGaddout1 + 24;
register_Ren_A_25 <= 1; register_Ren_B_25 <= 1;
                register_Radd_A_26 <= REGerror + 25; register_Radd_B_26 <= REGaddout1 + 25;
register_Ren_A_26 <= 1; register_Ren_B_26 <= 1;
                register_Radd_A_27 <= REGerror + 26; register_Radd_B_27 <= REGaddout1 + 26;
register_Ren_A_27 <= 1; register_Ren_B_27 <= 1;
                register_Radd_A_28 <= REGerror + 27; register_Radd_B_28 <= REGaddout1 + 27;
register_Ren_A_28 <= 1; register_Ren_B_28 <= 1;
                register_Radd_A_29 <= REGerror + 28; register_Radd_B_29 <= REGaddout1 + 28;
register_Ren_A_29 <= 1; register_Ren_B_29 <= 1;
                register_Radd_A_30 <= REGerror + 29; register_Radd_B_30 <= REGaddout1 + 29;
register_Ren_A_30 <= 1; register_Ren_B_30 <= 1;

                BP_code <= 6;
        end
    else if (cout_cntBPdelh == 4)
        begin
                start <= 1;
        end
    else if (cout_cntBPdelh == 5)
        begin
                start <= 0;
        end
    else if (cout_cntBPdelh == 13)
        begin
                register_Ren_A_1 <= 0; register_Ren_B_1 <= 0;
                register_Ren_A_2 <= 0; register_Ren_B_2 <= 0;
                register_Ren_A_3 <= 0; register_Ren_B_3 <= 0;
                register_Ren_A_4 <= 0; register_Ren_B_4 <= 0;
                register_Ren_A_5 <= 0; register_Ren_B_5 <= 0;
                register_Ren_A_6 <= 0; register_Ren_B_6 <= 0;
                register_Ren_A_7 <= 0; register_Ren_B_7 <= 0;
                register_Ren_A_8 <= 0; register_Ren_B_8 <= 0;
                register_Ren_A_9 <= 0; register_Ren_B_9 <= 0;
                register_Ren_A_10 <= 0; register_Ren_B_10 <= 0;
                register_Ren_A_11 <= 0; register_Ren_B_11 <= 0;
                register_Ren_A_12 <= 0; register_Ren_B_12 <= 0;
                register_Ren_A_13 <= 0; register_Ren_B_13 <= 0;
                register_Ren_A_14 <= 0; register_Ren_B_14 <= 0;
                register_Ren_A_15 <= 0; register_Ren_B_15 <= 0;
                register_Ren_A_16 <= 0; register_Ren_B_16 <= 0;
                register_Ren_A_17 <= 0; register_Ren_B_17 <= 0;
                register_Ren_A_18 <= 0; register_Ren_B_18 <= 0;
                register_Ren_A_19 <= 0; register_Ren_B_19 <= 0;
                register_Ren_A_20 <= 0; register_Ren_B_20 <= 0;
                register_Ren_A_21 <= 0; register_Ren_B_21 <= 0;
                register_Ren_A_22 <= 0; register_Ren_B_22 <= 0;
                register_Ren_A_23 <= 0; register_Ren_B_23 <= 0;
                register_Ren_A_24 <= 0; register_Ren_B_24 <= 0;
                register_Ren_A_25 <= 0; register_Ren_B_25 <= 0;
                register_Ren_A_26 <= 0; register_Ren_B_26 <= 0;
                register_Ren_A_27 <= 0; register_Ren_B_27 <= 0;
                register_Ren_A_28 <= 0; register_Ren_B_28 <= 0;
                register_Ren_A_29 <= 0; register_Ren_B_29 <= 0;
```

```
                register_Ren_A_30 <= 0; register_Ren_B_30 <= 0;
        end
    else if (cout_cntBPdelh == 14)
        begin
                neuron_out_temp[1] <= neuron_out_1;
                neuron_out_temp[2] <= neuron_out_2;
                neuron_out_temp[3] <= neuron_out_3;
                neuron_out_temp[4] <= neuron_out_4;
                neuron_out_temp[5] <= neuron_out_5;
                neuron_out_temp[6] <= neuron_out_6;
                neuron_out_temp[7] <= neuron_out_7;
                neuron_out_temp[8] <= neuron_out_8;
                neuron_out_temp[9] <= neuron_out_9;
                neuron_out_temp[10] <= neuron_out_10;
                neuron_out_temp[11] <= neuron_out_11;
                neuron_out_temp[12] <= neuron_out_12;
                neuron_out_temp[13] <= neuron_out_13;
                neuron_out_temp[14] <= neuron_out_14;
                neuron_out_temp[15] <= neuron_out_15;
                neuron_out_temp[16] <= neuron_out_16;
                neuron_out_temp[17] <= neuron_out_17;
                neuron_out_temp[18] <= neuron_out_18;
                neuron_out_temp[19] <= neuron_out_19;
                neuron_out_temp[20] <= neuron_out_20;
                neuron_out_temp[21] <= neuron_out_21;
                neuron_out_temp[22] <= neuron_out_22;
                neuron_out_temp[23] <= neuron_out_23;
                neuron_out_temp[24] <= neuron_out_24;
                neuron_out_temp[25] <= neuron_out_25;
                neuron_out_temp[26] <= neuron_out_26;
                neuron_out_temp[27] <= neuron_out_27;
                neuron_out_temp[28] <= neuron_out_28;
                neuron_out_temp[29] <= neuron_out_29;
                neuron_out_temp[30] <= neuron_out_30;
        end
    else if (cout_cntBPdelh == 15)
        begin
                register_Wadd_1 <= REGerror; register_Wen_1 <= 1; register_Wdata_1 <=
neuron_out_temp[1];
                register_Wadd_2 <= REGerror; register_Wen_2 <= 1; register_Wdata_2 <=
neuron_out_temp[2];
                register_Wadd_3 <= REGerror; register_Wen_3 <= 1; register_Wdata_3 <=
neuron_out_temp[3];
                register_Wadd_4 <= REGerror; register_Wen_4 <= 1; register_Wdata_4 <=
neuron_out_temp[4];
                register_Wadd_5 <= REGerror; register_Wen_5 <= 1; register_Wdata_5 <=
neuron_out_temp[5];
                register_Wadd_6 <= REGerror; register_Wen_6 <= 1; register_Wdata_6 <=
neuron_out_temp[6];
                register_Wadd_7 <= REGerror; register_Wen_7 <= 1; register_Wdata_7 <=
neuron_out_temp[7];
                register_Wadd_8 <= REGerror; register_Wen_8 <= 1; register_Wdata_8 <=
neuron_out_temp[8];
                register_Wadd_9 <= REGerror; register_Wen_9 <= 1; register_Wdata_9 <=
neuron_out_temp[9];
                register_Wadd_10 <= REGerror; register_Wen_10 <= 1; register_Wdata_10 <=
neuron_out_temp[10];
                register_Wadd_11 <= REGerror; register_Wen_11 <= 1; register_Wdata_11 <=
neuron_out_temp[11];
                register_Wadd_12 <= REGerror; register_Wen_12 <= 1; register_Wdata_12 <=
neuron_out_temp[12];
                register_Wadd_13 <= REGerror; register_Wen_13 <= 1; register_Wdata_13 <=
neuron_out_temp[13];
                register_Wadd_14 <= REGerror; register_Wen_14 <= 1; register_Wdata_14 <=
neuron_out_temp[14];
                register_Wadd_15 <= REGerror; register_Wen_15 <= 1; register_Wdata_15 <=
neuron_out_temp[15];
                register_Wadd_16 <= REGerror; register_Wen_16 <= 1; register_Wdata_16 <=
neuron_out_temp[16];
```

```
                register_Wadd_17 <= REGerror; register_Wen_17 <= 1; register_Wdata_17 <=
neuron_out_temp[17];
                register_Wadd_18 <= REGerror; register_Wen_18 <= 1; register_Wdata_18 <=
neuron_out_temp[18];
                register_Wadd_19 <= REGerror; register_Wen_19 <= 1; register_Wdata_19 <=
neuron_out_temp[19];
                register_Wadd_20 <= REGerror; register_Wen_20 <= 1; register_Wdata_20 <=
neuron_out_temp[20];
                register_Wadd_21 <= REGerror; register_Wen_21 <= 1; register_Wdata_21 <=
neuron_out_temp[21];
                register_Wadd_22 <= REGerror; register_Wen_22 <= 1; register_Wdata_22 <=
neuron_out_temp[22];
                register_Wadd_23 <= REGerror; register_Wen_23 <= 1; register_Wdata_23 <=
neuron_out_temp[23];
                register_Wadd_24 <= REGerror; register_Wen_24 <= 1; register_Wdata_24 <=
neuron_out_temp[24];
                register_Wadd_25 <= REGerror; register_Wen_25 <= 1; register_Wdata_25 <=
neuron_out_temp[25];
                register_Wadd_26 <= REGerror; register_Wen_26 <= 1; register_Wdata_26 <=
neuron_out_temp[26];
                register_Wadd_27 <= REGerror; register_Wen_27 <= 1; register_Wdata_27 <=
neuron_out_temp[27];
                register_Wadd_28 <= REGerror; register_Wen_28 <= 1; register_Wdata_28 <=
neuron_out_temp[28];
                register_Wadd_29 <= REGerror; register_Wen_29 <= 1; register_Wdata_29 <=
neuron_out_temp[29];
                register_Wadd_30 <= REGerror; register_Wen_30 <= 1; register_Wdata_30 <=
neuron_out_temp[30];
            end
        else if (cout_cntBPdelh == 17)
            begin
                register_Wen_1 <= 0;
                register_Wen_2 <= 0;
                register_Wen_3 <= 0;
                register_Wen_4 <= 0;
                register_Wen_5 <= 0;
                register_Wen_6 <= 0;
                register_Wen_7 <= 0;
                register_Wen_8 <= 0;
                register_Wen_9 <= 0;
                register_Wen_10 <= 0;
                register_Wen_11 <= 0;
                register_Wen_12 <= 0;
                register_Wen_13 <= 0;
                register_Wen_14 <= 0;
                register_Wen_15 <= 0;
                register_Wen_16 <= 0;
                register_Wen_17 <= 0;
                register_Wen_18 <= 0;
                register_Wen_19 <= 0;
                register_Wen_20 <= 0;
                register_Wen_21 <= 0;
                register_Wen_22 <= 0;
                register_Wen_23 <= 0;
                register_Wen_24 <= 0;
                register_Wen_25 <= 0;
                register_Wen_26 <= 0;
                register_Wen_27 <= 0;
                register_Wen_28 <= 0;
                register_Wen_29 <= 0;
                register_Wen_30 <= 0;
            end
        else if (cout_cntBPdelh == 18)
            begin
                register_Radd_A_1 <= REGerror; register_Radd_B_1 <= REGaddout1 + 0;
register_Ren_A_1 <= 1; register_Ren_B_1 <= 1;
                register_Radd_A_2 <= REGerror; register_Radd_B_2 <= REGaddout1 + 1;
register_Ren_A_2 <= 1; register_Ren_B_2 <= 1;
                register_Radd_A_3 <= REGerror; register_Radd_B_3 <= REGaddout1 + 2;
register_Ren_A_3 <= 1; register_Ren_B_3 <= 1;
```

```
                                        register_Radd_A_4 <= REGerror; register_Radd_B_4 <= REGaddout1 + 3;
register_Ren_A_4 <= 1; register_Ren_B_4 <= 1;
                                        register_Radd_A_5 <= REGerror; register_Radd_B_5 <= REGaddout1 + 4;
register_Ren_A_5 <= 1; register_Ren_B_5 <= 1;
                                        register_Radd_A_6 <= REGerror; register_Radd_B_6 <= REGaddout1 + 5;
register_Ren_A_6 <= 1; register_Ren_B_6 <= 1;
                                        register_Radd_A_7 <= REGerror; register_Radd_B_7 <= REGaddout1 + 6;
register_Ren_A_7 <= 1; register_Ren_B_7 <= 1;
                                        register_Radd_A_8 <= REGerror; register_Radd_B_8 <= REGaddout1 + 7;
register_Ren_A_8 <= 1; register_Ren_B_8 <= 1;
                                        register_Radd_A_9 <= REGerror; register_Radd_B_9 <= REGaddout1 + 8;
register_Ren_A_9 <= 1; register_Ren_B_9 <= 1;
                                        register_Radd_A_10 <= REGerror; register_Radd_B_10 <= REGaddout1 + 9;
register_Ren_A_10 <= 1; register_Ren_B_10 <= 1;
                                        register_Radd_A_11 <= REGerror; register_Radd_B_11 <= REGaddout1 + 10;
register_Ren_A_11 <= 1; register_Ren_B_11 <= 1;
                                        register_Radd_A_12 <= REGerror; register_Radd_B_12 <= REGaddout1 + 11;
register_Ren_A_12 <= 1; register_Ren_B_12 <= 1;
                                        register_Radd_A_13 <= REGerror; register_Radd_B_13 <= REGaddout1 + 12;
register_Ren_A_13 <= 1; register_Ren_B_13 <= 1;
                                        register_Radd_A_14 <= REGerror; register_Radd_B_14 <= REGaddout1 + 13;
register_Ren_A_14 <= 1; register_Ren_B_14 <= 1;
                                        register_Radd_A_15 <= REGerror; register_Radd_B_15 <= REGaddout1 + 14;
register_Ren_A_15 <= 1; register_Ren_B_15 <= 1;
                                        register_Radd_A_16 <= REGerror; register_Radd_B_16 <= REGaddout1 + 15;
register_Ren_A_16 <= 1; register_Ren_B_16 <= 1;
                                        register_Radd_A_17 <= REGerror; register_Radd_B_17 <= REGaddout1 + 16;
register_Ren_A_17 <= 1; register_Ren_B_17 <= 1;
                                        register_Radd_A_18 <= REGerror; register_Radd_B_18 <= REGaddout1 + 17;
register_Ren_A_18 <= 1; register_Ren_B_18 <= 1;
                                        register_Radd_A_19 <= REGerror; register_Radd_B_19 <= REGaddout1 + 18;
register_Ren_A_19 <= 1; register_Ren_B_19 <= 1;
                                        register_Radd_A_20 <= REGerror; register_Radd_B_20 <= REGaddout1 + 19;
register_Ren_A_20 <= 1; register_Ren_B_20 <= 1;
                                        register_Radd_A_21 <= REGerror; register_Radd_B_21 <= REGaddout1 + 20;
register_Ren_A_21 <= 1; register_Ren_B_21 <= 1;
                                        register_Radd_A_22 <= REGerror; register_Radd_B_22 <= REGaddout1 + 21;
register_Ren_A_22 <= 1; register_Ren_B_22 <= 1;
                                        register_Radd_A_23 <= REGerror; register_Radd_B_23 <= REGaddout1 + 22;
register_Ren_A_23 <= 1; register_Ren_B_23 <= 1;
                                        register_Radd_A_24 <= REGerror; register_Radd_B_24 <= REGaddout1 + 23;
register_Ren_A_24 <= 1; register_Ren_B_24 <= 1;
                                        register_Radd_A_25 <= REGerror; register_Radd_B_25 <= REGaddout1 + 24;
register_Ren_A_25 <= 1; register_Ren_B_25 <= 1;
                                        register_Radd_A_26 <= REGerror; register_Radd_B_26 <= REGaddout1 + 25;
register_Ren_A_26 <= 1; register_Ren_B_26 <= 1;
                                        register_Radd_A_27 <= REGerror; register_Radd_B_27 <= REGaddout1 + 26;
register_Ren_A_27 <= 1; register_Ren_B_27 <= 1;
                                        register_Radd_A_28 <= REGerror; register_Radd_B_28 <= REGaddout1 + 27;
register_Ren_A_28 <= 1; register_Ren_B_28 <= 1;
                                        register_Radd_A_29 <= REGerror; register_Radd_B_29 <= REGaddout1 + 28;
register_Ren_A_29 <= 1; register_Ren_B_29 <= 1;
                                        register_Radd_A_30 <= REGerror; register_Radd_B_30 <= REGaddout1 + 29;
register_Ren_A_30 <= 1; register_Ren_B_30 <= 1;

                                        BP_code <= 4;
                                end
                        else if (cout_cntBPdelh == 21)
                                begin
                                        start <= 1;
                                end
                        else if (cout_cntBPdelh == 22)
                                begin
                                        start <= 0;
                                end
                        else if (cout_cntBPdelh == 30)
                                begin
                                        register_Ren_A_1 <= 0; register_Ren_B_1 <= 0;
                                        register_Ren_A_2 <= 0; register_Ren_B_2 <= 0;
                                        register_Ren_A_3 <= 0; register_Ren_B_3 <= 0;
```

```
                                        register_Ren_A_4 <= 0; register_Ren_B_4 <= 0;
                                        register_Ren_A_5 <= 0; register_Ren_B_5 <= 0;
                                        register_Ren_A_6 <= 0; register_Ren_B_6 <= 0;
                                        register_Ren_A_7 <= 0; register_Ren_B_7 <= 0;
                                        register_Ren_A_8 <= 0; register_Ren_B_8 <= 0;
                                        register_Ren_A_9 <= 0; register_Ren_B_9 <= 0;
                                        register_Ren_A_10 <= 0; register_Ren_B_10 <= 0;
                                        register_Ren_A_11 <= 0; register_Ren_B_11 <= 0;
                                        register_Ren_A_12 <= 0; register_Ren_B_12 <= 0;
                                        register_Ren_A_13 <= 0; register_Ren_B_13 <= 0;
                                        register_Ren_A_14 <= 0; register_Ren_B_14 <= 0;
                                        register_Ren_A_15 <= 0; register_Ren_B_15 <= 0;
                                        register_Ren_A_16 <= 0; register_Ren_B_16 <= 0;
                                        register_Ren_A_17 <= 0; register_Ren_B_17 <= 0;
                                        register_Ren_A_18 <= 0; register_Ren_B_18 <= 0;
                                        register_Ren_A_19 <= 0; register_Ren_B_19 <= 0;
                                        register_Ren_A_20 <= 0; register_Ren_B_20 <= 0;
                                        register_Ren_A_21 <= 0; register_Ren_B_21 <= 0;
                                        register_Ren_A_22 <= 0; register_Ren_B_22 <= 0;
                                        register_Ren_A_23 <= 0; register_Ren_B_23 <= 0;
                                        register_Ren_A_24 <= 0; register_Ren_B_24 <= 0;
                                        register_Ren_A_25 <= 0; register_Ren_B_25 <= 0;
                                        register_Ren_A_26 <= 0; register_Ren_B_26 <= 0;
                                        register_Ren_A_27 <= 0; register_Ren_B_27 <= 0;
                                        register_Ren_A_28 <= 0; register_Ren_B_28 <= 0;
                                        register_Ren_A_29 <= 0; register_Ren_B_29 <= 0;
                                        register_Ren_A_30 <= 0; register_Ren_B_30 <= 0;
                                end
                        end
                16:begin
                                if (cout_cntBPdelhstore == 1)
                                        begin
                                                neuron_out_temp[1] <= neuron_out_1;
                                                neuron_out_temp[2] <= neuron_out_2;
                                                neuron_out_temp[3] <= neuron_out_3;
                                                neuron_out_temp[4] <= neuron_out_4;
                                                neuron_out_temp[5] <= neuron_out_5;
                                                neuron_out_temp[6] <= neuron_out_6;
                                                neuron_out_temp[7] <= neuron_out_7;
                                                neuron_out_temp[8] <= neuron_out_8;
                                                neuron_out_temp[9] <= neuron_out_9;
                                                neuron_out_temp[10] <= neuron_out_10;
                                                neuron_out_temp[11] <= neuron_out_11;
                                                neuron_out_temp[12] <= neuron_out_12;
                                                neuron_out_temp[13] <= neuron_out_13;
                                                neuron_out_temp[14] <= neuron_out_14;
                                                neuron_out_temp[15] <= neuron_out_15;
                                                neuron_out_temp[16] <= neuron_out_16;
                                                neuron_out_temp[17] <= neuron_out_17;
                                                neuron_out_temp[18] <= neuron_out_18;
                                                neuron_out_temp[19] <= neuron_out_19;
                                                neuron_out_temp[20] <= neuron_out_20;
                                                neuron_out_temp[21] <= neuron_out_21;
                                                neuron_out_temp[22] <= neuron_out_22;
                                                neuron_out_temp[23] <= neuron_out_23;
                                                neuron_out_temp[24] <= neuron_out_24;
                                                neuron_out_temp[25] <= neuron_out_25;
                                                neuron_out_temp[26] <= neuron_out_26;
                                                neuron_out_temp[27] <= neuron_out_27;
                                                neuron_out_temp[28] <= neuron_out_28;
                                                neuron_out_temp[29] <= neuron_out_29;
                                                neuron_out_temp[30] <= neuron_out_30;
                                        end
                                else if (cout_cntBPdelhstore == 2)
                                        begin
                                                register_Wadd_1 <= REGdelh + cout_sumBPdelhstore; register_Wen_1 <= 1;
register_Wdata_1 <= neuron_out_temp[cout_sumBPdelhstore+1];
                                                register_Wadd_2 <= REGdelh + cout_sumBPdelhstore; register_Wen_2 <= 1;
register_Wdata_2 <= neuron_out_temp[cout_sumBPdelhstore+1];
```

```
                register_Wadd_3 <= REGdelh + cout_sumBPdelhstore; register_Wen_3 <= 1;
register_Wdata_3 <= neuron_out_temp[cout_sumBPdelhstore+1];
                register_Wadd_4 <= REGdelh + cout_sumBPdelhstore; register_Wen_4 <= 1;
register_Wdata_4 <= neuron_out_temp[cout_sumBPdelhstore+1];
                register_Wadd_5 <= REGdelh + cout_sumBPdelhstore; register_Wen_5 <= 1;
register_Wdata_5 <= neuron_out_temp[cout_sumBPdelhstore+1];
                register_Wadd_6 <= REGdelh + cout_sumBPdelhstore; register_Wen_6 <= 1;
register_Wdata_6 <= neuron_out_temp[cout_sumBPdelhstore+1];
                register_Wadd_7 <= REGdelh + cout_sumBPdelhstore; register_Wen_7 <= 1;
register_Wdata_7 <= neuron_out_temp[cout_sumBPdelhstore+1];
                register_Wadd_8 <= REGdelh + cout_sumBPdelhstore; register_Wen_8 <= 1;
register_Wdata_8 <= neuron_out_temp[cout_sumBPdelhstore+1];
                register_Wadd_9 <= REGdelh + cout_sumBPdelhstore; register_Wen_9 <= 1;
register_Wdata_9 <= neuron_out_temp[cout_sumBPdelhstore+1];
                register_Wadd_10 <= REGdelh + cout_sumBPdelhstore; register_Wen_10 <= 1;
register_Wdata_10 <= neuron_out_temp[cout_sumBPdelhstore+1];
                register_Wadd_11 <= REGdelh + cout_sumBPdelhstore; register_Wen_11 <= 1;
register_Wdata_11 <= neuron_out_temp[cout_sumBPdelhstore+1];
                register_Wadd_12 <= REGdelh + cout_sumBPdelhstore; register_Wen_12 <= 1;
register_Wdata_12 <= neuron_out_temp[cout_sumBPdelhstore+1];
                register_Wadd_13 <= REGdelh + cout_sumBPdelhstore; register_Wen_13 <= 1;
register_Wdata_13 <= neuron_out_temp[cout_sumBPdelhstore+1];
                register_Wadd_14 <= REGdelh + cout_sumBPdelhstore; register_Wen_14 <= 1;
register_Wdata_14 <= neuron_out_temp[cout_sumBPdelhstore+1];
                register_Wadd_15 <= REGdelh + cout_sumBPdelhstore; register_Wen_15 <= 1;
register_Wdata_15 <= neuron_out_temp[cout_sumBPdelhstore+1];
                register_Wadd_16 <= REGdelh + cout_sumBPdelhstore; register_Wen_16 <= 1;
register_Wdata_16 <= neuron_out_temp[cout_sumBPdelhstore+1];
                register_Wadd_17 <= REGdelh + cout_sumBPdelhstore; register_Wen_17 <= 1;
register_Wdata_17 <= neuron_out_temp[cout_sumBPdelhstore+1];
                register_Wadd_18 <= REGdelh + cout_sumBPdelhstore; register_Wen_18 <= 1;
register_Wdata_18 <= neuron_out_temp[cout_sumBPdelhstore+1];
                register_Wadd_19 <= REGdelh + cout_sumBPdelhstore; register_Wen_19 <= 1;
register_Wdata_19 <= neuron_out_temp[cout_sumBPdelhstore+1];
                register_Wadd_20 <= REGdelh + cout_sumBPdelhstore; register_Wen_20 <= 1;
register_Wdata_20 <= neuron_out_temp[cout_sumBPdelhstore+1];
                register_Wadd_21 <= REGdelh + cout_sumBPdelhstore; register_Wen_21 <= 1;
register_Wdata_21 <= neuron_out_temp[cout_sumBPdelhstore+1];
                register_Wadd_22 <= REGdelh + cout_sumBPdelhstore; register_Wen_22 <= 1;
register_Wdata_22 <= neuron_out_temp[cout_sumBPdelhstore+1];
                register_Wadd_23 <= REGdelh + cout_sumBPdelhstore; register_Wen_23 <= 1;
register_Wdata_23 <= neuron_out_temp[cout_sumBPdelhstore+1];
                register_Wadd_24 <= REGdelh + cout_sumBPdelhstore; register_Wen_24 <= 1;
register_Wdata_24 <= neuron_out_temp[cout_sumBPdelhstore+1];
                register_Wadd_25 <= REGdelh + cout_sumBPdelhstore; register_Wen_25 <= 1;
register_Wdata_25 <= neuron_out_temp[cout_sumBPdelhstore+1];
                register_Wadd_26 <= REGdelh + cout_sumBPdelhstore; register_Wen_26 <= 1;
register_Wdata_26 <= neuron_out_temp[cout_sumBPdelhstore+1];
                register_Wadd_27 <= REGdelh + cout_sumBPdelhstore; register_Wen_27 <= 1;
register_Wdata_27 <= neuron_out_temp[cout_sumBPdelhstore+1];
                register_Wadd_28 <= REGdelh + cout_sumBPdelhstore; register_Wen_28 <= 1;
register_Wdata_28 <= neuron_out_temp[cout_sumBPdelhstore+1];
                register_Wadd_29 <= REGdelh + cout_sumBPdelhstore; register_Wen_29 <= 1;
register_Wdata_29 <= neuron_out_temp[cout_sumBPdelhstore+1];
                register_Wadd_30 <= REGdelh + cout_sumBPdelhstore; register_Wen_30 <= 1;
register_Wdata_30 <= neuron_out_temp[cout_sumBPdelhstore+1];
            end
        else if (cout_cntBPdelhstore == 4)
            begin
                register_Wen_1 <= 0;
                register_Wen_2 <= 0;
                register_Wen_3 <= 0;
                register_Wen_4 <= 0;
                register_Wen_5 <= 0;
                register_Wen_6 <= 0;
                register_Wen_7 <= 0;
                register_Wen_8 <= 0;
                register_Wen_9 <= 0;
                register_Wen_10 <= 0;
                register_Wen_11 <= 0;
```

```
                                        register_Wen_12 <= 0;
                                        register_Wen_13 <= 0;
                                        register_Wen_14 <= 0;
                                        register_Wen_15 <= 0;
                                        register_Wen_16 <= 0;
                                        register_Wen_17 <= 0;
                                        register_Wen_18 <= 0;
                                        register_Wen_19 <= 0;
                                        register_Wen_20 <= 0;
                                        register_Wen_21 <= 0;
                                        register_Wen_22 <= 0;
                                        register_Wen_23 <= 0;
                                        register_Wen_24 <= 0;
                                        register_Wen_25 <= 0;
                                        register_Wen_26 <= 0;
                                        register_Wen_27 <= 0;
                                        register_Wen_28 <= 0;
                                        register_Wen_29 <= 0;
                                        register_Wen_30 <= 0;
                                    end
                                end
                            17:begin
                                    if (cout_cntBPweightih == 1)
                                        begin
                                            register_Radd_A_1 <= REGeta; register_Radd_B_1 <= REGdelh + 0; register_Ren_A_1
<= 1; register_Ren_B_1 <= 1;
                                            register_Radd_A_2 <= REGeta; register_Radd_B_2 <= REGdelh + 1; register_Ren_A_2
<= 1; register_Ren_B_2 <= 1;
                                            register_Radd_A_3 <= REGeta; register_Radd_B_3 <= REGdelh + 2; register_Ren_A_3
<= 1; register_Ren_B_3 <= 1;
                                            register_Radd_A_4 <= REGeta; register_Radd_B_4 <= REGdelh + 3; register_Ren_A_4
<= 1; register_Ren_B_4 <= 1;
                                            register_Radd_A_5 <= REGeta; register_Radd_B_5 <= REGdelh + 4; register_Ren_A_5
<= 1; register_Ren_B_5 <= 1;
                                            register_Radd_A_6 <= REGeta; register_Radd_B_6 <= REGdelh + 5; register_Ren_A_6
<= 1; register_Ren_B_6 <= 1;
                                            register_Radd_A_7 <= REGeta; register_Radd_B_7 <= REGdelh + 6; register_Ren_A_7
<= 1; register_Ren_B_7 <= 1;
                                            register_Radd_A_8 <= REGeta; register_Radd_B_8 <= REGdelh + 7; register_Ren_A_8
<= 1; register_Ren_B_8 <= 1;
                                            register_Radd_A_9 <= REGeta; register_Radd_B_9 <= REGdelh + 8; register_Ren_A_9
<= 1; register_Ren_B_9 <= 1;
                                            register_Radd_A_10 <= REGeta; register_Radd_B_10 <= REGdelh + 9;
register_Ren_A_10 <= 1; register_Ren_B_10 <= 1;
                                            register_Radd_A_11 <= REGeta; register_Radd_B_11 <= REGdelh + 10;
register_Ren_A_11 <= 1; register_Ren_B_11 <= 1;
                                            register_Radd_A_12 <= REGeta; register_Radd_B_12 <= REGdelh + 11;
register_Ren_A_12 <= 1; register_Ren_B_12 <= 1;
                                            register_Radd_A_13 <= REGeta; register_Radd_B_13 <= REGdelh + 12;
register_Ren_A_13 <= 1; register_Ren_B_13 <= 1;
                                            register_Radd_A_14 <= REGeta; register_Radd_B_14 <= REGdelh + 13;
register_Ren_A_14 <= 1; register_Ren_B_14 <= 1;
                                            register_Radd_A_15 <= REGeta; register_Radd_B_15 <= REGdelh + 14;
register_Ren_A_15 <= 1; register_Ren_B_15 <= 1;
                                            register_Radd_A_16 <= REGeta; register_Radd_B_16 <= REGdelh + 15;
register_Ren_A_16 <= 1; register_Ren_B_16 <= 1;
                                            register_Radd_A_17 <= REGeta; register_Radd_B_17 <= REGdelh + 16;
register_Ren_A_17 <= 1; register_Ren_B_17 <= 1;
                                            register_Radd_A_18 <= REGeta; register_Radd_B_18 <= REGdelh + 17;
register_Ren_A_18 <= 1; register_Ren_B_18 <= 1;
                                            register_Radd_A_19 <= REGeta; register_Radd_B_19 <= REGdelh + 18;
register_Ren_A_19 <= 1; register_Ren_B_19 <= 1;
                                            register_Radd_A_20 <= REGeta; register_Radd_B_20 <= REGdelh + 19;
register_Ren_A_20 <= 1; register_Ren_B_20 <= 1;
                                            register_Radd_A_21 <= REGeta; register_Radd_B_21 <= REGdelh + 20;
register_Ren_A_21 <= 1; register_Ren_B_21 <= 1;
                                            register_Radd_A_22 <= REGeta; register_Radd_B_22 <= REGdelh + 21;
register_Ren_A_22 <= 1; register_Ren_B_22 <= 1;
                                            register_Radd_A_23 <= REGeta; register_Radd_B_23 <= REGdelh + 22;
register_Ren_A_23 <= 1; register_Ren_B_23 <= 1;
```

```
                register_Radd_A_24 <= REGeta; register_Radd_B_24 <= REGdelh + 23;
register_Ren_A_24 <= 1; register_Ren_B_24 <= 1;
                register_Radd_A_25 <= REGeta; register_Radd_B_25 <= REGdelh + 24;
register_Ren_A_25 <= 1; register_Ren_B_25 <= 1;
                register_Radd_A_26 <= REGeta; register_Radd_B_26 <= REGdelh + 25;
register_Ren_A_26 <= 1; register_Ren_B_26 <= 1;
                register_Radd_A_27 <= REGeta; register_Radd_B_27 <= REGdelh + 26;
register_Ren_A_27 <= 1; register_Ren_B_27 <= 1;
                register_Radd_A_28 <= REGeta; register_Radd_B_28 <= REGdelh + 27;
register_Ren_A_28 <= 1; register_Ren_B_28 <= 1;
                register_Radd_A_29 <= REGeta; register_Radd_B_29 <= REGdelh + 28;
register_Ren_A_29 <= 1; register_Ren_B_29 <= 1;
                register_Radd_A_30 <= REGeta; register_Radd_B_30 <= REGdelh + 29;
register_Ren_A_30 <= 1; register_Ren_B_30 <= 1;

                        BP_code <= 6;
                end
        else if (cout_cntBPweightih == 4)
                begin
                        start       <= 1;
                end
        else if (cout_cntBPweightih == 5)
                begin
                        start <= 0;
                end
        else if (cout_cntBPweightih == 13)
                begin
                        register_Ren_A_1 <= 0; register_Ren_B_1 <= 0;
                        register_Ren_A_2 <= 0; register_Ren_B_2 <= 0;
                        register_Ren_A_3 <= 0; register_Ren_B_3 <= 0;
                        register_Ren_A_4 <= 0; register_Ren_B_4 <= 0;
                        register_Ren_A_5 <= 0; register_Ren_B_5 <= 0;
                        register_Ren_A_6 <= 0; register_Ren_B_6 <= 0;
                        register_Ren_A_7 <= 0; register_Ren_B_7 <= 0;
                        register_Ren_A_8 <= 0; register_Ren_B_8 <= 0;
                        register_Ren_A_9 <= 0; register_Ren_B_9 <= 0;
                        register_Ren_A_10 <= 0; register_Ren_B_10 <= 0;
                        register_Ren_A_11 <= 0; register_Ren_B_11 <= 0;
                        register_Ren_A_12 <= 0; register_Ren_B_12 <= 0;
                        register_Ren_A_13 <= 0; register_Ren_B_13 <= 0;
                        register_Ren_A_14 <= 0; register_Ren_B_14 <= 0;
                        register_Ren_A_15 <= 0; register_Ren_B_15 <= 0;
                        register_Ren_A_16 <= 0; register_Ren_B_16 <= 0;
                        register_Ren_A_17 <= 0; register_Ren_B_17 <= 0;
                        register_Ren_A_18 <= 0; register_Ren_B_18 <= 0;
                        register_Ren_A_19 <= 0; register_Ren_B_19 <= 0;
                        register_Ren_A_20 <= 0; register_Ren_B_20 <= 0;
                        register_Ren_A_21 <= 0; register_Ren_B_21 <= 0;
                        register_Ren_A_22 <= 0; register_Ren_B_22 <= 0;
                        register_Ren_A_23 <= 0; register_Ren_B_23 <= 0;
                        register_Ren_A_24 <= 0; register_Ren_B_24 <= 0;
                        register_Ren_A_25 <= 0; register_Ren_B_25 <= 0;
                        register_Ren_A_26 <= 0; register_Ren_B_26 <= 0;
                        register_Ren_A_27 <= 0; register_Ren_B_27 <= 0;
                        register_Ren_A_28 <= 0; register_Ren_B_28 <= 0;
                        register_Ren_A_29 <= 0; register_Ren_B_29 <= 0;
                        register_Ren_A_30 <= 0; register_Ren_B_30 <= 0;
                end
        else if (cout_cntBPweightih == 14)
                begin
                        neuron_out_temp[1] <= neuron_out_1;
                        neuron_out_temp[2] <= neuron_out_2;
                        neuron_out_temp[3] <= neuron_out_3;
                        neuron_out_temp[4] <= neuron_out_4;
                        neuron_out_temp[5] <= neuron_out_5;
                        neuron_out_temp[6] <= neuron_out_6;
                        neuron_out_temp[7] <= neuron_out_7;
                        neuron_out_temp[8] <= neuron_out_8;
                        neuron_out_temp[9] <= neuron_out_9;
                        neuron_out_temp[10] <= neuron_out_10;
```

```
                neuron_out_temp[11] <= neuron_out_11;
                neuron_out_temp[12] <= neuron_out_12;
                neuron_out_temp[13] <= neuron_out_13;
                neuron_out_temp[14] <= neuron_out_14;
                neuron_out_temp[15] <= neuron_out_15;
                neuron_out_temp[16] <= neuron_out_16;
                neuron_out_temp[17] <= neuron_out_17;
                neuron_out_temp[18] <= neuron_out_18;
                neuron_out_temp[19] <= neuron_out_19;
                neuron_out_temp[20] <= neuron_out_20;
                neuron_out_temp[21] <= neuron_out_21;
                neuron_out_temp[22] <= neuron_out_22;
                neuron_out_temp[23] <= neuron_out_23;
                neuron_out_temp[24] <= neuron_out_24;
                neuron_out_temp[25] <= neuron_out_25;
                neuron_out_temp[26] <= neuron_out_26;
                neuron_out_temp[27] <= neuron_out_27;
                neuron_out_temp[28] <= neuron_out_28;
                neuron_out_temp[29] <= neuron_out_29;
                neuron_out_temp[30] <= neuron_out_30;
            end
        else if (cout_cntBPweightih == 15)
            begin
                register_Wadd_1 <= REGerror + cout_sumBPweightih; register_Wen_1 <= 1;
register_Wdata_1 <= neuron_out_temp[1];
                register_Wadd_2 <= REGerror + cout_sumBPweightih; register_Wen_2 <= 1;
register_Wdata_2 <= neuron_out_temp[2];
                register_Wadd_3 <= REGerror + cout_sumBPweightih; register_Wen_3 <= 1;
register_Wdata_3 <= neuron_out_temp[3];
                register_Wadd_4 <= REGerror + cout_sumBPweightih; register_Wen_4 <= 1;
register_Wdata_4 <= neuron_out_temp[4];
                register_Wadd_5 <= REGerror + cout_sumBPweightih; register_Wen_5 <= 1;
register_Wdata_5 <= neuron_out_temp[5];
                register_Wadd_6 <= REGerror + cout_sumBPweightih; register_Wen_6 <= 1;
register_Wdata_6 <= neuron_out_temp[6];
                register_Wadd_7 <= REGerror + cout_sumBPweightih; register_Wen_7 <= 1;
register_Wdata_7 <= neuron_out_temp[7];
                register_Wadd_8 <= REGerror + cout_sumBPweightih; register_Wen_8 <= 1;
register_Wdata_8 <= neuron_out_temp[8];
                register_Wadd_9 <= REGerror + cout_sumBPweightih; register_Wen_9 <= 1;
register_Wdata_9 <= neuron_out_temp[9];
                register_Wadd_10 <= REGerror + cout_sumBPweightih; register_Wen_10 <= 1;
register_Wdata_10 <= neuron_out_temp[10];
                register_Wadd_11 <= REGerror + cout_sumBPweightih; register_Wen_11 <= 1;
register_Wdata_11 <= neuron_out_temp[11];
                register_Wadd_12 <= REGerror + cout_sumBPweightih; register_Wen_12 <= 1;
register_Wdata_12 <= neuron_out_temp[12];
                register_Wadd_13 <= REGerror + cout_sumBPweightih; register_Wen_13 <= 1;
register_Wdata_13 <= neuron_out_temp[13];
                register_Wadd_14 <= REGerror + cout_sumBPweightih; register_Wen_14 <= 1;
register_Wdata_14 <= neuron_out_temp[14];
                register_Wadd_15 <= REGerror + cout_sumBPweightih; register_Wen_15 <= 1;
register_Wdata_15 <= neuron_out_temp[15];
                register_Wadd_16 <= REGerror + cout_sumBPweightih; register_Wen_16 <= 1;
register_Wdata_16 <= neuron_out_temp[16];
                register_Wadd_17 <= REGerror + cout_sumBPweightih; register_Wen_17 <= 1;
register_Wdata_17 <= neuron_out_temp[17];
                register_Wadd_18 <= REGerror + cout_sumBPweightih; register_Wen_18 <= 1;
register_Wdata_18 <= neuron_out_temp[18];
                register_Wadd_19 <= REGerror + cout_sumBPweightih; register_Wen_19 <= 1;
register_Wdata_19 <= neuron_out_temp[19];
                register_Wadd_20 <= REGerror + cout_sumBPweightih; register_Wen_20 <= 1;
register_Wdata_20 <= neuron_out_temp[20];
                register_Wadd_21 <= REGerror + cout_sumBPweightih; register_Wen_21 <= 1;
register_Wdata_21 <= neuron_out_temp[21];
                register_Wadd_22 <= REGerror + cout_sumBPweightih; register_Wen_22 <= 1;
register_Wdata_22 <= neuron_out_temp[22];
                register_Wadd_23 <= REGerror + cout_sumBPweightih; register_Wen_23 <= 1;
register_Wdata_23 <= neuron_out_temp[23];
```

```
                                        register_Wadd_24 <= REGerror + cout_sumBPweightih; register_Wen_24 <= 1;
register_Wdata_24 <= neuron_out_temp[24];
                                        register_Wadd_25 <= REGerror + cout_sumBPweightih; register_Wen_25 <= 1;
register_Wdata_25 <= neuron_out_temp[25];
                                        register_Wadd_26 <= REGerror + cout_sumBPweightih; register_Wen_26 <= 1;
register_Wdata_26 <= neuron_out_temp[26];
                                        register_Wadd_27 <= REGerror + cout_sumBPweightih; register_Wen_27 <= 1;
register_Wdata_27 <= neuron_out_temp[27];
                                        register_Wadd_28 <= REGerror + cout_sumBPweightih; register_Wen_28 <= 1;
register_Wdata_28 <= neuron_out_temp[28];
                                        register_Wadd_29 <= REGerror + cout_sumBPweightih; register_Wen_29 <= 1;
register_Wdata_29 <= neuron_out_temp[29];
                                        register_Wadd_30 <= REGerror + cout_sumBPweightih; register_Wen_30 <= 1;
register_Wdata_30 <= neuron_out_temp[30];
                                end
                        else if (cout_cntBPweightih == 17)
                                begin
                                        register_Wen_1 <= 0;
                                        register_Wen_2 <= 0;
                                        register_Wen_3 <= 0;
                                        register_Wen_4 <= 0;
                                        register_Wen_5 <= 0;
                                        register_Wen_6 <= 0;
                                        register_Wen_7 <= 0;
                                        register_Wen_8 <= 0;
                                        register_Wen_9 <= 0;
                                        register_Wen_10 <= 0;
                                        register_Wen_11 <= 0;
                                        register_Wen_12 <= 0;
                                        register_Wen_13 <= 0;
                                        register_Wen_14 <= 0;
                                        register_Wen_15 <= 0;
                                        register_Wen_16 <= 0;
                                        register_Wen_17 <= 0;
                                        register_Wen_18 <= 0;
                                        register_Wen_19 <= 0;
                                        register_Wen_20 <= 0;
                                        register_Wen_21 <= 0;
                                        register_Wen_22 <= 0;
                                        register_Wen_23 <= 0;
                                        register_Wen_24 <= 0;
                                        register_Wen_25 <= 0;
                                        register_Wen_26 <= 0;
                                        register_Wen_27 <= 0;
                                        register_Wen_28 <= 0;
                                        register_Wen_29 <= 0;
                                        register_Wen_30 <= 0;
                                end
                        else if (cout_cntBPweightih == 18)
                                begin
                                        register_Radd_A_1 <= REGalpha; register_Radd_B_1 <= REGdelweightih +
cout_sumBPweightih; register_Ren_A_1 <= 1; register_Ren_B_1 <= 1;
                                        register_Radd_A_2 <= REGalpha; register_Radd_B_2 <= REGdelweightih +
cout_sumBPweightih; register_Ren_A_2 <= 1; register_Ren_B_2 <= 1;
                                        register_Radd_A_3 <= REGalpha; register_Radd_B_3 <= REGdelweightih +
cout_sumBPweightih; register_Ren_A_3 <= 1; register_Ren_B_3 <= 1;
                                        register_Radd_A_4 <= REGalpha; register_Radd_B_4 <= REGdelweightih +
cout_sumBPweightih; register_Ren_A_4 <= 1; register_Ren_B_4 <= 1;
                                        register_Radd_A_5 <= REGalpha; register_Radd_B_5 <= REGdelweightih +
cout_sumBPweightih; register_Ren_A_5 <= 1; register_Ren_B_5 <= 1;
                                        register_Radd_A_6 <= REGalpha; register_Radd_B_6 <= REGdelweightih +
cout_sumBPweightih; register_Ren_A_6 <= 1; register_Ren_B_6 <= 1;
                                        register_Radd_A_7 <= REGalpha; register_Radd_B_7 <= REGdelweightih +
cout_sumBPweightih; register_Ren_A_7 <= 1; register_Ren_B_7 <= 1;
                                        register_Radd_A_8 <= REGalpha; register_Radd_B_8 <= REGdelweightih +
cout_sumBPweightih; register_Ren_A_8 <= 1; register_Ren_B_8 <= 1;
                                        register_Radd_A_9 <= REGalpha; register_Radd_B_9 <= REGdelweightih +
cout_sumBPweightih; register_Ren_A_9 <= 1; register_Ren_B_9 <= 1;
                                        register_Radd_A_10 <= REGalpha; register_Radd_B_10 <= REGdelweightih +
cout_sumBPweightih; register_Ren_A_10 <= 1; register_Ren_B_10 <= 1;
```

```
                                        register_Radd_A_11  <=  REGalpha;  register_Radd_B_11  <=  REGdelweightih  +
cout_sumBPweightih; register_Ren_A_11 <= 1; register_Ren_B_11 <= 1;
                                        register_Radd_A_12  <=  REGalpha;  register_Radd_B_12  <=  REGdelweightih  +
cout_sumBPweightih; register_Ren_A_12 <= 1; register_Ren_B_12 <= 1;
                                        register_Radd_A_13  <=  REGalpha;  register_Radd_B_13  <=  REGdelweightih  +
cout_sumBPweightih; register_Ren_A_13 <= 1; register_Ren_B_13 <= 1;
                                        register_Radd_A_14  <=  REGalpha;  register_Radd_B_14  <=  REGdelweightih  +
cout_sumBPweightih; register_Ren_A_14 <= 1; register_Ren_B_14 <= 1;
                                        register_Radd_A_15  <=  REGalpha;  register_Radd_B_15  <=  REGdelweightih  +
cout_sumBPweightih; register_Ren_A_15 <= 1; register_Ren_B_15 <= 1;
                                        register_Radd_A_16  <=  REGalpha;  register_Radd_B_16  <=  REGdelweightih  +
cout_sumBPweightih; register_Ren_A_16 <= 1; register_Ren_B_16 <= 1;
                                        register_Radd_A_17  <=  REGalpha;  register_Radd_B_17  <=  REGdelweightih  +
cout_sumBPweightih; register_Ren_A_17 <= 1; register_Ren_B_17 <= 1;
                                        register_Radd_A_18  <=  REGalpha;  register_Radd_B_18  <=  REGdelweightih  +
cout_sumBPweightih; register_Ren_A_18 <= 1; register_Ren_B_18 <= 1;
                                        register_Radd_A_19  <=  REGalpha;  register_Radd_B_19  <=  REGdelweightih  +
cout_sumBPweightih; register_Ren_A_19 <= 1; register_Ren_B_19 <= 1;
                                        register_Radd_A_20  <=  REGalpha;  register_Radd_B_20  <=  REGdelweightih  +
cout_sumBPweightih; register_Ren_A_20 <= 1; register_Ren_B_20 <= 1;
                                        register_Radd_A_21  <=  REGalpha;  register_Radd_B_21  <=  REGdelweightih  +
cout_sumBPweightih; register_Ren_A_21 <= 1; register_Ren_B_21 <= 1;
                                        register_Radd_A_22  <=  REGalpha;  register_Radd_B_22  <=  REGdelweightih  +
cout_sumBPweightih; register_Ren_A_22 <= 1; register_Ren_B_22 <= 1;
                                        register_Radd_A_23  <=  REGalpha;  register_Radd_B_23  <=  REGdelweightih  +
cout_sumBPweightih; register_Ren_A_23 <= 1; register_Ren_B_23 <= 1;
                                        register_Radd_A_24  <=  REGalpha;  register_Radd_B_24  <=  REGdelweightih  +
cout_sumBPweightih; register_Ren_A_24 <= 1; register_Ren_B_24 <= 1;
                                        register_Radd_A_25  <=  REGalpha;  register_Radd_B_25  <=  REGdelweightih  +
cout_sumBPweightih; register_Ren_A_25 <= 1; register_Ren_B_25 <= 1;
                                        register_Radd_A_26  <=  REGalpha;  register_Radd_B_26  <=  REGdelweightih  +
cout_sumBPweightih; register_Ren_A_26 <= 1; register_Ren_B_26 <= 1;
                                        register_Radd_A_27  <=  REGalpha;  register_Radd_B_27  <=  REGdelweightih  +
cout_sumBPweightih; register_Ren_A_27 <= 1; register_Ren_B_27 <= 1;
                                        register_Radd_A_28  <=  REGalpha;  register_Radd_B_28  <=  REGdelweightih  +
cout_sumBPweightih; register_Ren_A_28 <= 1; register_Ren_B_28 <= 1;
                                        register_Radd_A_29  <=  REGalpha;  register_Radd_B_29  <=  REGdelweightih  +
cout_sumBPweightih; register_Ren_A_29 <= 1; register_Ren_B_29 <= 1;
                                        register_Radd_A_30  <=  REGalpha;  register_Radd_B_30  <=  REGdelweightih  +
cout_sumBPweightih; register_Ren_A_30 <= 1; register_Ren_B_30 <= 1;

                                                BP_code <= 6;
                                        end
                                else if (cout_cntBPweightih == 21)
                                        begin
                                                start           <= 1;
                                        end
                                else if (cout_cntBPweightih == 22)
                                        begin
                                                start <= 0;
                                        end
                                else if (cout_cntBPweightih == 30)
                                        begin
                                                register_Ren_A_1 <= 0; register_Ren_B_1 <= 0;
                                                register_Ren_A_2 <= 0; register_Ren_B_2 <= 0;
                                                register_Ren_A_3 <= 0; register_Ren_B_3 <= 0;
                                                register_Ren_A_4 <= 0; register_Ren_B_4 <= 0;
                                                register_Ren_A_5 <= 0; register_Ren_B_5 <= 0;
                                                register_Ren_A_6 <= 0; register_Ren_B_6 <= 0;
                                                register_Ren_A_7 <= 0; register_Ren_B_7 <= 0;
                                                register_Ren_A_8 <= 0; register_Ren_B_8 <= 0;
                                                register_Ren_A_9 <= 0; register_Ren_B_9 <= 0;
                                                register_Ren_A_10 <= 0; register_Ren_B_10 <= 0;
                                                register_Ren_A_11 <= 0; register_Ren_B_11 <= 0;
                                                register_Ren_A_12 <= 0; register_Ren_B_12 <= 0;
                                                register_Ren_A_13 <= 0; register_Ren_B_13 <= 0;
                                                register_Ren_A_14 <= 0; register_Ren_B_14 <= 0;
                                                register_Ren_A_15 <= 0; register_Ren_B_15 <= 0;
                                                register_Ren_A_16 <= 0; register_Ren_B_16 <= 0;
                                                register_Ren_A_17 <= 0; register_Ren_B_17 <= 0;
```

```
                register_Ren_A_18 <= 0; register_Ren_B_18 <= 0;
                register_Ren_A_19 <= 0; register_Ren_B_19 <= 0;
                register_Ren_A_20 <= 0; register_Ren_B_20 <= 0;
                register_Ren_A_21 <= 0; register_Ren_B_21 <= 0;
                register_Ren_A_22 <= 0; register_Ren_B_22 <= 0;
                register_Ren_A_23 <= 0; register_Ren_B_23 <= 0;
                register_Ren_A_24 <= 0; register_Ren_B_24 <= 0;
                register_Ren_A_25 <= 0; register_Ren_B_25 <= 0;
                register_Ren_A_26 <= 0; register_Ren_B_26 <= 0;
                register_Ren_A_27 <= 0; register_Ren_B_27 <= 0;
                register_Ren_A_28 <= 0; register_Ren_B_28 <= 0;
                register_Ren_A_29 <= 0; register_Ren_B_29 <= 0;
                register_Ren_A_30 <= 0; register_Ren_B_30 <= 0;
            end
        else if (cout_cntBPweightih == 31)
            begin
                neuron_out_temp[1] <= neuron_out_1;
                neuron_out_temp[2] <= neuron_out_2;
                neuron_out_temp[3] <= neuron_out_3;
                neuron_out_temp[4] <= neuron_out_4;
                neuron_out_temp[5] <= neuron_out_5;
                neuron_out_temp[6] <= neuron_out_6;
                neuron_out_temp[7] <= neuron_out_7;
                neuron_out_temp[8] <= neuron_out_8;
                neuron_out_temp[9] <= neuron_out_9;
                neuron_out_temp[10] <= neuron_out_10;
                neuron_out_temp[11] <= neuron_out_11;
                neuron_out_temp[12] <= neuron_out_12;
                neuron_out_temp[13] <= neuron_out_13;
                neuron_out_temp[14] <= neuron_out_14;
                neuron_out_temp[15] <= neuron_out_15;
                neuron_out_temp[16] <= neuron_out_16;
                neuron_out_temp[17] <= neuron_out_17;
                neuron_out_temp[18] <= neuron_out_18;
                neuron_out_temp[19] <= neuron_out_19;
                neuron_out_temp[20] <= neuron_out_20;
                neuron_out_temp[21] <= neuron_out_21;
                neuron_out_temp[22] <= neuron_out_22;
                neuron_out_temp[23] <= neuron_out_23;
                neuron_out_temp[24] <= neuron_out_24;
                neuron_out_temp[25] <= neuron_out_25;
                neuron_out_temp[26] <= neuron_out_26;
                neuron_out_temp[27] <= neuron_out_27;
                neuron_out_temp[28] <= neuron_out_28;
                neuron_out_temp[29] <= neuron_out_29;
                neuron_out_temp[30] <= neuron_out_30;
            end
        else if (cout_cntBPweightih == 32)
            begin
                register_Wadd_1 <= REGdelweightih + cout_sumBPweightih; register_Wen_1 <= 1;
register_Wdata_1 <= neuron_out_temp[1];
                register_Wadd_2 <= REGdelweightih + cout_sumBPweightih; register_Wen_2 <= 1;
register_Wdata_2 <= neuron_out_temp[2];
                register_Wadd_3 <= REGdelweightih + cout_sumBPweightih; register_Wen_3 <= 1;
register_Wdata_3 <= neuron_out_temp[3];
                register_Wadd_4 <= REGdelweightih + cout_sumBPweightih; register_Wen_4 <= 1;
register_Wdata_4 <= neuron_out_temp[4];
                register_Wadd_5 <= REGdelweightih + cout_sumBPweightih; register_Wen_5 <= 1;
register_Wdata_5 <= neuron_out_temp[5];
                register_Wadd_6 <= REGdelweightih + cout_sumBPweightih; register_Wen_6 <= 1;
register_Wdata_6 <= neuron_out_temp[6];
                register_Wadd_7 <= REGdelweightih + cout_sumBPweightih; register_Wen_7 <= 1;
register_Wdata_7 <= neuron_out_temp[7];
                register_Wadd_8 <= REGdelweightih + cout_sumBPweightih; register_Wen_8 <= 1;
register_Wdata_8 <= neuron_out_temp[8];
                register_Wadd_9 <= REGdelweightih + cout_sumBPweightih; register_Wen_9 <= 1;
register_Wdata_9 <= neuron_out_temp[9];
                register_Wadd_10 <= REGdelweightih + cout_sumBPweightih; register_Wen_10 <= 1;
register_Wdata_10 <= neuron_out_temp[10];
```

```
                register_Wadd_11 <= REGdelweightih + cout_sumBPweightih; register_Wen_11 <= 1;
register_Wdata_11 <= neuron_out_temp[11];
                register_Wadd_12 <= REGdelweightih + cout_sumBPweightih; register_Wen_12 <= 1;
register_Wdata_12 <= neuron_out_temp[12];
                register_Wadd_13 <= REGdelweightih + cout_sumBPweightih; register_Wen_13 <= 1;
register_Wdata_13 <= neuron_out_temp[13];
                register_Wadd_14 <= REGdelweightih + cout_sumBPweightih; register_Wen_14 <= 1;
register_Wdata_14 <= neuron_out_temp[14];
                register_Wadd_15 <= REGdelweightih + cout_sumBPweightih; register_Wen_15 <= 1;
register_Wdata_15 <= neuron_out_temp[15];
                register_Wadd_16 <= REGdelweightih + cout_sumBPweightih; register_Wen_16 <= 1;
register_Wdata_16 <= neuron_out_temp[16];
                register_Wadd_17 <= REGdelweightih + cout_sumBPweightih; register_Wen_17 <= 1;
register_Wdata_17 <= neuron_out_temp[17];
                register_Wadd_18 <= REGdelweightih + cout_sumBPweightih; register_Wen_18 <= 1;
register_Wdata_18 <= neuron_out_temp[18];
                register_Wadd_19 <= REGdelweightih + cout_sumBPweightih; register_Wen_19 <= 1;
register_Wdata_19 <= neuron_out_temp[19];
                register_Wadd_20 <= REGdelweightih + cout_sumBPweightih; register_Wen_20 <= 1;
register_Wdata_20 <= neuron_out_temp[20];
                register_Wadd_21 <= REGdelweightih + cout_sumBPweightih; register_Wen_21 <= 1;
register_Wdata_21 <= neuron_out_temp[21];
                register_Wadd_22 <= REGdelweightih + cout_sumBPweightih; register_Wen_22 <= 1;
register_Wdata_22 <= neuron_out_temp[22];
                register_Wadd_23 <= REGdelweightih + cout_sumBPweightih; register_Wen_23 <= 1;
register_Wdata_23 <= neuron_out_temp[23];
                register_Wadd_24 <= REGdelweightih + cout_sumBPweightih; register_Wen_24 <= 1;
register_Wdata_24 <= neuron_out_temp[24];
                register_Wadd_25 <= REGdelweightih + cout_sumBPweightih; register_Wen_25 <= 1;
register_Wdata_25 <= neuron_out_temp[25];
                register_Wadd_26 <= REGdelweightih + cout_sumBPweightih; register_Wen_26 <= 1;
register_Wdata_26 <= neuron_out_temp[26];
                register_Wadd_27 <= REGdelweightih + cout_sumBPweightih; register_Wen_27 <= 1;
register_Wdata_27 <= neuron_out_temp[27];
                register_Wadd_28 <= REGdelweightih + cout_sumBPweightih; register_Wen_28 <= 1;
register_Wdata_28 <= neuron_out_temp[28];
                register_Wadd_29 <= REGdelweightih + cout_sumBPweightih; register_Wen_29 <= 1;
register_Wdata_29 <= neuron_out_temp[29];
                register_Wadd_30 <= REGdelweightih + cout_sumBPweightih; register_Wen_30 <= 1;
register_Wdata_30 <= neuron_out_temp[30];
                end
        else if (cout_cntBPweightih == 34)
                begin
                register_Wen_1 <= 0;
                register_Wen_2 <= 0;
                register_Wen_3 <= 0;
                register_Wen_4 <= 0;
                register_Wen_5 <= 0;
                register_Wen_6 <= 0;
                register_Wen_7 <= 0;
                register_Wen_8 <= 0;
                register_Wen_9 <= 0;
                register_Wen_10 <= 0;
                register_Wen_11 <= 0;
                register_Wen_12 <= 0;
                register_Wen_13 <= 0;
                register_Wen_14 <= 0;
                register_Wen_15 <= 0;
                register_Wen_16 <= 0;
                register_Wen_17 <= 0;
                register_Wen_18 <= 0;
                register_Wen_19 <= 0;
                register_Wen_20 <= 0;
                register_Wen_21 <= 0;
                register_Wen_22 <= 0;
                register_Wen_23 <= 0;
                register_Wen_24 <= 0;
                register_Wen_25 <= 0;
                register_Wen_26 <= 0;
                register_Wen_27 <= 0;
```

```
                register_Wen_28 <= 0;
                register_Wen_29 <= 0;
                register_Wen_30 <= 0;
            end
        else if (cout_cntBPweightih == 35)
            begin
                register_Radd_A_1 <= REGerror + cout_sumBPweightih; register_Radd_B_1 <= REGaddECG + cout_sumBPweightih-1; register_Ren_A_1 <= 1; register_Ren_B_1 <= 1;
                register_Radd_A_2 <= REGerror + cout_sumBPweightih; register_Radd_B_2 <= REGaddECG + cout_sumBPweightih-1; register_Ren_A_2 <= 1; register_Ren_B_2 <= 1;
                register_Radd_A_3 <= REGerror + cout_sumBPweightih; register_Radd_B_3 <= REGaddECG + cout_sumBPweightih-1; register_Ren_A_3 <= 1; register_Ren_B_3 <= 1;
                register_Radd_A_4 <= REGerror + cout_sumBPweightih; register_Radd_B_4 <= REGaddECG + cout_sumBPweightih-1; register_Ren_A_4 <= 1; register_Ren_B_4 <= 1;
                register_Radd_A_5 <= REGerror + cout_sumBPweightih; register_Radd_B_5 <= REGaddECG + cout_sumBPweightih-1; register_Ren_A_5 <= 1; register_Ren_B_5 <= 1;
                register_Radd_A_6 <= REGerror + cout_sumBPweightih; register_Radd_B_6 <= REGaddECG + cout_sumBPweightih-1; register_Ren_A_6 <= 1; register_Ren_B_6 <= 1;
                register_Radd_A_7 <= REGerror + cout_sumBPweightih; register_Radd_B_7 <= REGaddECG + cout_sumBPweightih-1; register_Ren_A_7 <= 1; register_Ren_B_7 <= 1;
                register_Radd_A_8 <= REGerror + cout_sumBPweightih; register_Radd_B_8 <= REGaddECG + cout_sumBPweightih-1; register_Ren_A_8 <= 1; register_Ren_B_8 <= 1;
                register_Radd_A_9 <= REGerror + cout_sumBPweightih; register_Radd_B_9 <= REGaddECG + cout_sumBPweightih-1; register_Ren_A_9 <= 1; register_Ren_B_9 <= 1;
                register_Radd_A_10 <= REGerror + cout_sumBPweightih; register_Radd_B_10 <= REGaddECG + cout_sumBPweightih-1; register_Ren_A_10 <= 1; register_Ren_B_10 <= 1;
                register_Radd_A_11 <= REGerror + cout_sumBPweightih; register_Radd_B_11 <= REGaddECG + cout_sumBPweightih-1; register_Ren_A_11 <= 1; register_Ren_B_11 <= 1;
                register_Radd_A_12 <= REGerror + cout_sumBPweightih; register_Radd_B_12 <= REGaddECG + cout_sumBPweightih-1; register_Ren_A_12 <= 1; register_Ren_B_12 <= 1;
                register_Radd_A_13 <= REGerror + cout_sumBPweightih; register_Radd_B_13 <= REGaddECG + cout_sumBPweightih-1; register_Ren_A_13 <= 1; register_Ren_B_13 <= 1;
                register_Radd_A_14 <= REGerror + cout_sumBPweightih; register_Radd_B_14 <= REGaddECG + cout_sumBPweightih-1; register_Ren_A_14 <= 1; register_Ren_B_14 <= 1;
                register_Radd_A_15 <= REGerror + cout_sumBPweightih; register_Radd_B_15 <= REGaddECG + cout_sumBPweightih-1; register_Ren_A_15 <= 1; register_Ren_B_15 <= 1;
                register_Radd_A_16 <= REGerror + cout_sumBPweightih; register_Radd_B_16 <= REGaddECG + cout_sumBPweightih-1; register_Ren_A_16 <= 1; register_Ren_B_16 <= 1;
                register_Radd_A_17 <= REGerror + cout_sumBPweightih; register_Radd_B_17 <= REGaddECG + cout_sumBPweightih-1; register_Ren_A_17 <= 1; register_Ren_B_17 <= 1;
                register_Radd_A_18 <= REGerror + cout_sumBPweightih; register_Radd_B_18 <= REGaddECG + cout_sumBPweightih-1; register_Ren_A_18 <= 1; register_Ren_B_18 <= 1;
                register_Radd_A_19 <= REGerror + cout_sumBPweightih; register_Radd_B_19 <= REGaddECG + cout_sumBPweightih-1; register_Ren_A_19 <= 1; register_Ren_B_19 <= 1;
                register_Radd_A_20 <= REGerror + cout_sumBPweightih; register_Radd_B_20 <= REGaddECG + cout_sumBPweightih-1; register_Ren_A_20 <= 1; register_Ren_B_20 <= 1;
                register_Radd_A_21 <= REGerror + cout_sumBPweightih; register_Radd_B_21 <= REGaddECG + cout_sumBPweightih-1; register_Ren_A_21 <= 1; register_Ren_B_21 <= 1;
                register_Radd_A_22 <= REGerror + cout_sumBPweightih; register_Radd_B_22 <= REGaddECG + cout_sumBPweightih-1; register_Ren_A_22 <= 1; register_Ren_B_22 <= 1;
                register_Radd_A_23 <= REGerror + cout_sumBPweightih; register_Radd_B_23 <= REGaddECG + cout_sumBPweightih-1; register_Ren_A_23 <= 1; register_Ren_B_23 <= 1;
                register_Radd_A_24 <= REGerror + cout_sumBPweightih; register_Radd_B_24 <= REGaddECG + cout_sumBPweightih-1; register_Ren_A_24 <= 1; register_Ren_B_24 <= 1;
                register_Radd_A_25 <= REGerror + cout_sumBPweightih; register_Radd_B_25 <= REGaddECG + cout_sumBPweightih-1; register_Ren_A_25 <= 1; register_Ren_B_25 <= 1;
                register_Radd_A_26 <= REGerror + cout_sumBPweightih; register_Radd_B_26 <= REGaddECG + cout_sumBPweightih-1; register_Ren_A_26 <= 1; register_Ren_B_26 <= 1;
                register_Radd_A_27 <= REGerror + cout_sumBPweightih; register_Radd_B_27 <= REGaddECG + cout_sumBPweightih-1; register_Ren_A_27 <= 1; register_Ren_B_27 <= 1;
                register_Radd_A_28 <= REGerror + cout_sumBPweightih; register_Radd_B_28 <= REGaddECG + cout_sumBPweightih-1; register_Ren_A_28 <= 1; register_Ren_B_28 <= 1;
                register_Radd_A_29 <= REGerror + cout_sumBPweightih; register_Radd_B_29 <= REGaddECG + cout_sumBPweightih-1; register_Ren_A_29 <= 1; register_Ren_B_29 <= 1;
                register_Radd_A_30 <= REGerror + cout_sumBPweightih; register_Radd_B_30 <= REGaddECG + cout_sumBPweightih-1; register_Ren_A_30 <= 1; register_Ren_B_30 <= 1;

                BP_code <= 6;
            end
        else if (cout_cntBPweightih == 38)
```

```
		begin
			start		<= 1;
		end
	else if (cout_cntBPweightih == 39)
		begin
			start <= 0;
		end
	else if (cout_cntBPweightih == 47)
		begin
			register_Ren_A_1 <= 0; register_Ren_B_1 <= 0;
			register_Ren_A_2 <= 0; register_Ren_B_2 <= 0;
			register_Ren_A_3 <= 0; register_Ren_B_3 <= 0;
			register_Ren_A_4 <= 0; register_Ren_B_4 <= 0;
			register_Ren_A_5 <= 0; register_Ren_B_5 <= 0;
			register_Ren_A_6 <= 0; register_Ren_B_6 <= 0;
			register_Ren_A_7 <= 0; register_Ren_B_7 <= 0;
			register_Ren_A_8 <= 0; register_Ren_B_8 <= 0;
			register_Ren_A_9 <= 0; register_Ren_B_9 <= 0;
			register_Ren_A_10 <= 0; register_Ren_B_10 <= 0;
			register_Ren_A_11 <= 0; register_Ren_B_11 <= 0;
			register_Ren_A_12 <= 0; register_Ren_B_12 <= 0;
			register_Ren_A_13 <= 0; register_Ren_B_13 <= 0;
			register_Ren_A_14 <= 0; register_Ren_B_14 <= 0;
			register_Ren_A_15 <= 0; register_Ren_B_15 <= 0;
			register_Ren_A_16 <= 0; register_Ren_B_16 <= 0;
			register_Ren_A_17 <= 0; register_Ren_B_17 <= 0;
			register_Ren_A_18 <= 0; register_Ren_B_18 <= 0;
			register_Ren_A_19 <= 0; register_Ren_B_19 <= 0;
			register_Ren_A_20 <= 0; register_Ren_B_20 <= 0;
			register_Ren_A_21 <= 0; register_Ren_B_21 <= 0;
			register_Ren_A_22 <= 0; register_Ren_B_22 <= 0;
			register_Ren_A_23 <= 0; register_Ren_B_23 <= 0;
			register_Ren_A_24 <= 0; register_Ren_B_24 <= 0;
			register_Ren_A_25 <= 0; register_Ren_B_25 <= 0;
			register_Ren_A_26 <= 0; register_Ren_B_26 <= 0;
			register_Ren_A_27 <= 0; register_Ren_B_27 <= 0;
			register_Ren_A_28 <= 0; register_Ren_B_28 <= 0;
			register_Ren_A_29 <= 0; register_Ren_B_29 <= 0;
			register_Ren_A_30 <= 0; register_Ren_B_30 <= 0;
		end
	else if (cout_cntBPweightih == 48)
		begin
			neuron_out_temp[1] <= neuron_out_1;
			neuron_out_temp[2] <= neuron_out_2;
			neuron_out_temp[3] <= neuron_out_3;
			neuron_out_temp[4] <= neuron_out_4;
			neuron_out_temp[5] <= neuron_out_5;
			neuron_out_temp[6] <= neuron_out_6;
			neuron_out_temp[7] <= neuron_out_7;
			neuron_out_temp[8] <= neuron_out_8;
			neuron_out_temp[9] <= neuron_out_9;
			neuron_out_temp[10] <= neuron_out_10;
			neuron_out_temp[11] <= neuron_out_11;
			neuron_out_temp[12] <= neuron_out_12;
			neuron_out_temp[13] <= neuron_out_13;
			neuron_out_temp[14] <= neuron_out_14;
			neuron_out_temp[15] <= neuron_out_15;
			neuron_out_temp[16] <= neuron_out_16;
			neuron_out_temp[17] <= neuron_out_17;
			neuron_out_temp[18] <= neuron_out_18;
			neuron_out_temp[19] <= neuron_out_19;
			neuron_out_temp[20] <= neuron_out_20;
			neuron_out_temp[21] <= neuron_out_21;
			neuron_out_temp[22] <= neuron_out_22;
			neuron_out_temp[23] <= neuron_out_23;
			neuron_out_temp[24] <= neuron_out_24;
			neuron_out_temp[25] <= neuron_out_25;
			neuron_out_temp[26] <= neuron_out_26;
			neuron_out_temp[27] <= neuron_out_27;
			neuron_out_temp[28] <= neuron_out_28;
```

```
                neuron_out_temp[29] <= neuron_out_29;
                neuron_out_temp[30] <= neuron_out_30;
            end
        else if (cout_cntBPweightih == 49)
            begin
                register_Wadd_1 <= REGerror + cout_sumBPweightih; register_Wen_1 <= 1;
register_Wdata_1 <= neuron_out_temp[1];
                register_Wadd_2 <= REGerror + cout_sumBPweightih; register_Wen_2 <= 1;
register_Wdata_2 <= neuron_out_temp[2];
                register_Wadd_3 <= REGerror + cout_sumBPweightih; register_Wen_3 <= 1;
register_Wdata_3 <= neuron_out_temp[3];
                register_Wadd_4 <= REGerror + cout_sumBPweightih; register_Wen_4 <= 1;
register_Wdata_4 <= neuron_out_temp[4];
                register_Wadd_5 <= REGerror + cout_sumBPweightih; register_Wen_5 <= 1;
register_Wdata_5 <= neuron_out_temp[5];
                register_Wadd_6 <= REGerror + cout_sumBPweightih; register_Wen_6 <= 1;
register_Wdata_6 <= neuron_out_temp[6];
                register_Wadd_7 <= REGerror + cout_sumBPweightih; register_Wen_7 <= 1;
register_Wdata_7 <= neuron_out_temp[7];
                register_Wadd_8 <= REGerror + cout_sumBPweightih; register_Wen_8 <= 1;
register_Wdata_8 <= neuron_out_temp[8];
                register_Wadd_9 <= REGerror + cout_sumBPweightih; register_Wen_9 <= 1;
register_Wdata_9 <= neuron_out_temp[9];
                register_Wadd_10 <= REGerror + cout_sumBPweightih; register_Wen_10 <= 1;
register_Wdata_10 <= neuron_out_temp[10];
                register_Wadd_11 <= REGerror + cout_sumBPweightih; register_Wen_11 <= 1;
register_Wdata_11 <= neuron_out_temp[11];
                register_Wadd_12 <= REGerror + cout_sumBPweightih; register_Wen_12 <= 1;
register_Wdata_12 <= neuron_out_temp[12];
                register_Wadd_13 <= REGerror + cout_sumBPweightih; register_Wen_13 <= 1;
register_Wdata_13 <= neuron_out_temp[13];
                register_Wadd_14 <= REGerror + cout_sumBPweightih; register_Wen_14 <= 1;
register_Wdata_14 <= neuron_out_temp[14];
                register_Wadd_15 <= REGerror + cout_sumBPweightih; register_Wen_15 <= 1;
register_Wdata_15 <= neuron_out_temp[15];
                register_Wadd_16 <= REGerror + cout_sumBPweightih; register_Wen_16 <= 1;
register_Wdata_16 <= neuron_out_temp[16];
                register_Wadd_17 <= REGerror + cout_sumBPweightih; register_Wen_17 <= 1;
register_Wdata_17 <= neuron_out_temp[17];
                register_Wadd_18 <= REGerror + cout_sumBPweightih; register_Wen_18 <= 1;
register_Wdata_18 <= neuron_out_temp[18];
                register_Wadd_19 <= REGerror + cout_sumBPweightih; register_Wen_19 <= 1;
register_Wdata_19 <= neuron_out_temp[19];
                register_Wadd_20 <= REGerror + cout_sumBPweightih; register_Wen_20 <= 1;
register_Wdata_20 <= neuron_out_temp[20];
                register_Wadd_21 <= REGerror + cout_sumBPweightih; register_Wen_21 <= 1;
register_Wdata_21 <= neuron_out_temp[21];
                register_Wadd_22 <= REGerror + cout_sumBPweightih; register_Wen_22 <= 1;
register_Wdata_22 <= neuron_out_temp[22];
                register_Wadd_23 <= REGerror + cout_sumBPweightih; register_Wen_23 <= 1;
register_Wdata_23 <= neuron_out_temp[23];
                register_Wadd_24 <= REGerror + cout_sumBPweightih; register_Wen_24 <= 1;
register_Wdata_24 <= neuron_out_temp[24];
                register_Wadd_25 <= REGerror + cout_sumBPweightih; register_Wen_25 <= 1;
register_Wdata_25 <= neuron_out_temp[25];
                register_Wadd_26 <= REGerror + cout_sumBPweightih; register_Wen_26 <= 1;
register_Wdata_26 <= neuron_out_temp[26];
                register_Wadd_27 <= REGerror + cout_sumBPweightih; register_Wen_27 <= 1;
register_Wdata_27 <= neuron_out_temp[27];
                register_Wadd_28 <= REGerror + cout_sumBPweightih; register_Wen_28 <= 1;
register_Wdata_28 <= neuron_out_temp[28];
                register_Wadd_29 <= REGerror + cout_sumBPweightih; register_Wen_29 <= 1;
register_Wdata_29 <= neuron_out_temp[29];
                register_Wadd_30 <= REGerror + cout_sumBPweightih; register_Wen_30 <= 1;
register_Wdata_30 <= neuron_out_temp[30];
            end
        else if (cout_cntBPweightih == 51)
            begin
                register_Wen_1 <= 0;
                register_Wen_2 <= 0;
```

```
                                        register_Wen_3 <= 0;
                                        register_Wen_4 <= 0;
                                        register_Wen_5 <= 0;
                                        register_Wen_6 <= 0;
                                        register_Wen_7 <= 0;
                                        register_Wen_8 <= 0;
                                        register_Wen_9 <= 0;
                                        register_Wen_10 <= 0;
                                        register_Wen_11 <= 0;
                                        register_Wen_12 <= 0;
                                        register_Wen_13 <= 0;
                                        register_Wen_14 <= 0;
                                        register_Wen_15 <= 0;
                                        register_Wen_16 <= 0;
                                        register_Wen_17 <= 0;
                                        register_Wen_18 <= 0;
                                        register_Wen_19 <= 0;
                                        register_Wen_20 <= 0;
                                        register_Wen_21 <= 0;
                                        register_Wen_22 <= 0;
                                        register_Wen_23 <= 0;
                                        register_Wen_24 <= 0;
                                        register_Wen_25 <= 0;
                                        register_Wen_26 <= 0;
                                        register_Wen_27 <= 0;
                                        register_Wen_28 <= 0;
                                        register_Wen_29 <= 0;
                                        register_Wen_30 <= 0;
                                    end
                            end
                    18:begin
                                if (cout_cntBPweightihstore == 1)
                                    begin
                                        register_Radd_A_1 <= REGerror + cout_sumBPweightihstore; register_Radd_B_1 <= REGdelweightih + cout_sumBPweightihstore; register_Ren_A_1 <= 1; register_Ren_B_1 <= 1;
                                        register_Radd_A_2 <= REGerror + cout_sumBPweightihstore; register_Radd_B_2 <= REGdelweightih + cout_sumBPweightihstore; register_Ren_A_2 <= 1; register_Ren_B_2 <= 1;
                                        register_Radd_A_3 <= REGerror + cout_sumBPweightihstore; register_Radd_B_3 <= REGdelweightih + cout_sumBPweightihstore; register_Ren_A_3 <= 1; register_Ren_B_3 <= 1;
                                        register_Radd_A_4 <= REGerror + cout_sumBPweightihstore; register_Radd_B_4 <= REGdelweightih + cout_sumBPweightihstore; register_Ren_A_4 <= 1; register_Ren_B_4 <= 1;
                                        register_Radd_A_5 <= REGerror + cout_sumBPweightihstore; register_Radd_B_5 <= REGdelweightih + cout_sumBPweightihstore; register_Ren_A_5 <= 1; register_Ren_B_5 <= 1;
                                        register_Radd_A_6 <= REGerror + cout_sumBPweightihstore; register_Radd_B_6 <= REGdelweightih + cout_sumBPweightihstore; register_Ren_A_6 <= 1; register_Ren_B_6 <= 1;
                                        register_Radd_A_7 <= REGerror + cout_sumBPweightihstore; register_Radd_B_7 <= REGdelweightih + cout_sumBPweightihstore; register_Ren_A_7 <= 1; register_Ren_B_7 <= 1;
                                        register_Radd_A_8 <= REGerror + cout_sumBPweightihstore; register_Radd_B_8 <= REGdelweightih + cout_sumBPweightihstore; register_Ren_A_8 <= 1; register_Ren_B_8 <= 1;
                                        register_Radd_A_9 <= REGerror + cout_sumBPweightihstore; register_Radd_B_9 <= REGdelweightih + cout_sumBPweightihstore; register_Ren_A_9 <= 1; register_Ren_B_9 <= 1;
                                        register_Radd_A_10 <= REGerror + cout_sumBPweightihstore; register_Radd_B_10 <= REGdelweightih + cout_sumBPweightihstore; register_Ren_A_10 <= 1; register_Ren_B_10 <= 1;
                                        register_Radd_A_11 <= REGerror + cout_sumBPweightihstore; register_Radd_B_11 <= REGdelweightih + cout_sumBPweightihstore; register_Ren_A_11 <= 1; register_Ren_B_11 <= 1;
                                        register_Radd_A_12 <= REGerror + cout_sumBPweightihstore; register_Radd_B_12 <= REGdelweightih + cout_sumBPweightihstore; register_Ren_A_12 <= 1; register_Ren_B_12 <= 1;
                                        register_Radd_A_13 <= REGerror + cout_sumBPweightihstore; register_Radd_B_13 <= REGdelweightih + cout_sumBPweightihstore; register_Ren_A_13 <= 1; register_Ren_B_13 <= 1;
                                        register_Radd_A_14 <= REGerror + cout_sumBPweightihstore; register_Radd_B_14 <= REGdelweightih + cout_sumBPweightihstore; register_Ren_A_14 <= 1; register_Ren_B_14 <= 1;
                                        register_Radd_A_15 <= REGerror + cout_sumBPweightihstore; register_Radd_B_15 <= REGdelweightih + cout_sumBPweightihstore; register_Ren_A_15 <= 1; register_Ren_B_15 <= 1;
                                        register_Radd_A_16 <= REGerror + cout_sumBPweightihstore; register_Radd_B_16 <= REGdelweightih + cout_sumBPweightihstore; register_Ren_A_16 <= 1; register_Ren_B_16 <= 1;
                                        register_Radd_A_17 <= REGerror + cout_sumBPweightihstore; register_Radd_B_17 <= REGdelweightih + cout_sumBPweightihstore; register_Ren_A_17 <= 1; register_Ren_B_17 <= 1;
                                        register_Radd_A_18 <= REGerror + cout_sumBPweightihstore; register_Radd_B_18 <= REGdelweightih + cout_sumBPweightihstore; register_Ren_A_18 <= 1; register_Ren_B_18 <= 1;
```

```
                                        register_Radd_A_19 <= REGerror + cout_sumBPweightihstore; register_Radd_B_19 <=
REGdelweightih + cout_sumBPweightihstore; register_Ren_A_19 <= 1; register_Ren_B_19 <= 1;
                                        register_Radd_A_20 <= REGerror + cout_sumBPweightihstore; register_Radd_B_20 <=
REGdelweightih + cout_sumBPweightihstore; register_Ren_A_20 <= 1; register_Ren_B_20 <= 1;
                                        register_Radd_A_21 <= REGerror + cout_sumBPweightihstore; register_Radd_B_21 <=
REGdelweightih + cout_sumBPweightihstore; register_Ren_A_21 <= 1; register_Ren_B_21 <= 1;
                                        register_Radd_A_22 <= REGerror + cout_sumBPweightihstore; register_Radd_B_22 <=
REGdelweightih + cout_sumBPweightihstore; register_Ren_A_22 <= 1; register_Ren_B_22 <= 1;
                                        register_Radd_A_23 <= REGerror + cout_sumBPweightihstore; register_Radd_B_23 <=
REGdelweightih + cout_sumBPweightihstore; register_Ren_A_23 <= 1; register_Ren_B_23 <= 1;
                                        register_Radd_A_24 <= REGerror + cout_sumBPweightihstore; register_Radd_B_24 <=
REGdelweightih + cout_sumBPweightihstore; register_Ren_A_24 <= 1; register_Ren_B_24 <= 1;
                                        register_Radd_A_25 <= REGerror + cout_sumBPweightihstore; register_Radd_B_25 <=
REGdelweightih + cout_sumBPweightihstore; register_Ren_A_25 <= 1; register_Ren_B_25 <= 1;
                                        register_Radd_A_26 <= REGerror + cout_sumBPweightihstore; register_Radd_B_26 <=
REGdelweightih + cout_sumBPweightihstore; register_Ren_A_26 <= 1; register_Ren_B_26 <= 1;
                                        register_Radd_A_27 <= REGerror + cout_sumBPweightihstore; register_Radd_B_27 <=
REGdelweightih + cout_sumBPweightihstore; register_Ren_A_27 <= 1; register_Ren_B_27 <= 1;
                                        register_Radd_A_28 <= REGerror + cout_sumBPweightihstore; register_Radd_B_28 <=
REGdelweightih + cout_sumBPweightihstore; register_Ren_A_28 <= 1; register_Ren_B_28 <= 1;
                                        register_Radd_A_29 <= REGerror + cout_sumBPweightihstore; register_Radd_B_29 <=
REGdelweightih + cout_sumBPweightihstore; register_Ren_A_29 <= 1; register_Ren_B_29 <= 1;
                                        register_Radd_A_30 <= REGerror + cout_sumBPweightihstore; register_Radd_B_30 <=
REGdelweightih + cout_sumBPweightihstore; register_Ren_A_30 <= 1; register_Ren_B_30 <= 1;

                                        BP_code <= 2;
                        end
                else if (cout_cntBPweightihstore == 4)
                        begin
                                        start           <= 1;
                        end
                else if (cout_cntBPweightihstore == 5)
                        begin
                                        start <= 0;
                        end
                else if (cout_cntBPweightihstore == 13)
                        begin
                                        register_Ren_A_1 <= 0; register_Ren_B_1 <= 0;
                                        register_Ren_A_2 <= 0; register_Ren_B_2 <= 0;
                                        register_Ren_A_3 <= 0; register_Ren_B_3 <= 0;
                                        register_Ren_A_4 <= 0; register_Ren_B_4 <= 0;
                                        register_Ren_A_5 <= 0; register_Ren_B_5 <= 0;
                                        register_Ren_A_6 <= 0; register_Ren_B_6 <= 0;
                                        register_Ren_A_7 <= 0; register_Ren_B_7 <= 0;
                                        register_Ren_A_8 <= 0; register_Ren_B_8 <= 0;
                                        register_Ren_A_9 <= 0; register_Ren_B_9 <= 0;
                                        register_Ren_A_10 <= 0; register_Ren_B_10 <= 0;
                                        register_Ren_A_11 <= 0; register_Ren_B_11 <= 0;
                                        register_Ren_A_12 <= 0; register_Ren_B_12 <= 0;
                                        register_Ren_A_13 <= 0; register_Ren_B_13 <= 0;
                                        register_Ren_A_14 <= 0; register_Ren_B_14 <= 0;
                                        register_Ren_A_15 <= 0; register_Ren_B_15 <= 0;
                                        register_Ren_A_16 <= 0; register_Ren_B_16 <= 0;
                                        register_Ren_A_17 <= 0; register_Ren_B_17 <= 0;
                                        register_Ren_A_18 <= 0; register_Ren_B_18 <= 0;
                                        register_Ren_A_19 <= 0; register_Ren_B_19 <= 0;
                                        register_Ren_A_20 <= 0; register_Ren_B_20 <= 0;
                                        register_Ren_A_21 <= 0; register_Ren_B_21 <= 0;
                                        register_Ren_A_22 <= 0; register_Ren_B_22 <= 0;
                                        register_Ren_A_23 <= 0; register_Ren_B_23 <= 0;
                                        register_Ren_A_24 <= 0; register_Ren_B_24 <= 0;
                                        register_Ren_A_25 <= 0; register_Ren_B_25 <= 0;
                                        register_Ren_A_26 <= 0; register_Ren_B_26 <= 0;
                                        register_Ren_A_27 <= 0; register_Ren_B_27 <= 0;
                                        register_Ren_A_28 <= 0; register_Ren_B_28 <= 0;
                                        register_Ren_A_29 <= 0; register_Ren_B_29 <= 0;
                                        register_Ren_A_30 <= 0; register_Ren_B_30 <= 0;
                        end
                else if (cout_cntBPweightihstore == 14)
                        begin
```

```
                neuron_out_temp[1] <= neuron_out_1;
                neuron_out_temp[2] <= neuron_out_2;
                neuron_out_temp[3] <= neuron_out_3;
                neuron_out_temp[4] <= neuron_out_4;
                neuron_out_temp[5] <= neuron_out_5;
                neuron_out_temp[6] <= neuron_out_6;
                neuron_out_temp[7] <= neuron_out_7;
                neuron_out_temp[8] <= neuron_out_8;
                neuron_out_temp[9] <= neuron_out_9;
                neuron_out_temp[10] <= neuron_out_10;
                neuron_out_temp[11] <= neuron_out_11;
                neuron_out_temp[12] <= neuron_out_12;
                neuron_out_temp[13] <= neuron_out_13;
                neuron_out_temp[14] <= neuron_out_14;
                neuron_out_temp[15] <= neuron_out_15;
                neuron_out_temp[16] <= neuron_out_16;
                neuron_out_temp[17] <= neuron_out_17;
                neuron_out_temp[18] <= neuron_out_18;
                neuron_out_temp[19] <= neuron_out_19;
                neuron_out_temp[20] <= neuron_out_20;
                neuron_out_temp[21] <= neuron_out_21;
                neuron_out_temp[22] <= neuron_out_22;
                neuron_out_temp[23] <= neuron_out_23;
                neuron_out_temp[24] <= neuron_out_24;
                neuron_out_temp[25] <= neuron_out_25;
                neuron_out_temp[26] <= neuron_out_26;
                neuron_out_temp[27] <= neuron_out_27;
                neuron_out_temp[28] <= neuron_out_28;
                neuron_out_temp[29] <= neuron_out_29;
                neuron_out_temp[30] <= neuron_out_30;
            end
        else if (cout_cntBPweightihstore == 15)
            begin
                register_Wadd_1 <= REGdelweightih + cout_sumBPweightihstore; register_Wen_1 <=
1; register_Wdata_1 <= neuron_out_temp[1];
                register_Wadd_2 <= REGdelweightih + cout_sumBPweightihstore; register_Wen_2 <=
1; register_Wdata_2 <= neuron_out_temp[2];
                register_Wadd_3 <= REGdelweightih + cout_sumBPweightihstore; register_Wen_3 <=
1; register_Wdata_3 <= neuron_out_temp[3];
                register_Wadd_4 <= REGdelweightih + cout_sumBPweightihstore; register_Wen_4 <=
1; register_Wdata_4 <= neuron_out_temp[4];
                register_Wadd_5 <= REGdelweightih + cout_sumBPweightihstore; register_Wen_5 <=
1; register_Wdata_5 <= neuron_out_temp[5];
                register_Wadd_6 <= REGdelweightih + cout_sumBPweightihstore; register_Wen_6 <=
1; register_Wdata_6 <= neuron_out_temp[6];
                register_Wadd_7 <= REGdelweightih + cout_sumBPweightihstore; register_Wen_7 <=
1; register_Wdata_7 <= neuron_out_temp[7];
                register_Wadd_8 <= REGdelweightih + cout_sumBPweightihstore; register_Wen_8 <=
1; register_Wdata_8 <= neuron_out_temp[8];
                register_Wadd_9 <= REGdelweightih + cout_sumBPweightihstore; register_Wen_9 <=
1; register_Wdata_9 <= neuron_out_temp[9];
                register_Wadd_10 <= REGdelweightih + cout_sumBPweightihstore; register_Wen_10
<= 1; register_Wdata_10 <= neuron_out_temp[10];
                register_Wadd_11 <= REGdelweightih + cout_sumBPweightihstore; register_Wen_11
<= 1; register_Wdata_11 <= neuron_out_temp[11];
                register_Wadd_12 <= REGdelweightih + cout_sumBPweightihstore; register_Wen_12
<= 1; register_Wdata_12 <= neuron_out_temp[12];
                register_Wadd_13 <= REGdelweightih + cout_sumBPweightihstore; register_Wen_13
<= 1; register_Wdata_13 <= neuron_out_temp[13];
                register_Wadd_14 <= REGdelweightih + cout_sumBPweightihstore; register_Wen_14
<= 1; register_Wdata_14 <= neuron_out_temp[14];
                register_Wadd_15 <= REGdelweightih + cout_sumBPweightihstore; register_Wen_15
<= 1; register_Wdata_15 <= neuron_out_temp[15];
                register_Wadd_16 <= REGdelweightih + cout_sumBPweightihstore; register_Wen_16
<= 1; register_Wdata_16 <= neuron_out_temp[16];
                register_Wadd_17 <= REGdelweightih + cout_sumBPweightihstore; register_Wen_17
<= 1; register_Wdata_17 <= neuron_out_temp[17];
                register_Wadd_18 <= REGdelweightih + cout_sumBPweightihstore; register_Wen_18
<= 1; register_Wdata_18 <= neuron_out_temp[18];
```

```
                register_Wadd_19 <= REGdelweightih + cout_sumBPweightihstore; register_Wen_19
<= 1; register_Wdata_19 <= neuron_out_temp[19];
                register_Wadd_20 <= REGdelweightih + cout_sumBPweightihstore; register_Wen_20
<= 1; register_Wdata_20 <= neuron_out_temp[20];
                register_Wadd_21 <= REGdelweightih + cout_sumBPweightihstore; register_Wen_21
<= 1; register_Wdata_21 <= neuron_out_temp[21];
                register_Wadd_22 <= REGdelweightih + cout_sumBPweightihstore; register_Wen_22
<= 1; register_Wdata_22 <= neuron_out_temp[22];
                register_Wadd_23 <= REGdelweightih + cout_sumBPweightihstore; register_Wen_23
<= 1; register_Wdata_23 <= neuron_out_temp[23];
                register_Wadd_24 <= REGdelweightih + cout_sumBPweightihstore; register_Wen_24
<= 1; register_Wdata_24 <= neuron_out_temp[24];
                register_Wadd_25 <= REGdelweightih + cout_sumBPweightihstore; register_Wen_25
<= 1; register_Wdata_25 <= neuron_out_temp[25];
                register_Wadd_26 <= REGdelweightih + cout_sumBPweightihstore; register_Wen_26
<= 1; register_Wdata_26 <= neuron_out_temp[26];
                register_Wadd_27 <= REGdelweightih + cout_sumBPweightihstore; register_Wen_27
<= 1; register_Wdata_27 <= neuron_out_temp[27];
                register_Wadd_28 <= REGdelweightih + cout_sumBPweightihstore; register_Wen_28
<= 1; register_Wdata_28 <= neuron_out_temp[28];
                register_Wadd_29 <= REGdelweightih + cout_sumBPweightihstore; register_Wen_29
<= 1; register_Wdata_29 <= neuron_out_temp[29];
                register_Wadd_30 <= REGdelweightih + cout_sumBPweightihstore; register_Wen_30
<= 1; register_Wdata_30 <= neuron_out_temp[30];
            end
        else if (cout_cntBPweightihstore == 17)
            begin
                register_Wen_1 <= 0;
                register_Wen_2 <= 0;
                register_Wen_3 <= 0;
                register_Wen_4 <= 0;
                register_Wen_5 <= 0;
                register_Wen_6 <= 0;
                register_Wen_7 <= 0;
                register_Wen_8 <= 0;
                register_Wen_9 <= 0;
                register_Wen_10 <= 0;
                register_Wen_11 <= 0;
                register_Wen_12 <= 0;
                register_Wen_13 <= 0;
                register_Wen_14 <= 0;
                register_Wen_15 <= 0;
                register_Wen_16 <= 0;
                register_Wen_17 <= 0;
                register_Wen_18 <= 0;
                register_Wen_19 <= 0;
                register_Wen_20 <= 0;
                register_Wen_21 <= 0;
                register_Wen_22 <= 0;
                register_Wen_23 <= 0;
                register_Wen_24 <= 0;
                register_Wen_25 <= 0;
                register_Wen_26 <= 0;
                register_Wen_27 <= 0;
                register_Wen_28 <= 0;
                register_Wen_29 <= 0;
                register_Wen_30 <= 0;
            end
        else if (cout_cntBPweightihstore == 18)
            begin
                register_Radd_A_1 <= REGaddinweight + cout_sumBPweightihstore;
register_Radd_B_1 <= REGdelweightih + cout_sumBPweightihstore; register_Ren_A_1 <= 1; register_Ren_B_1 <= 1;
                register_Radd_A_2 <= REGaddinweight + cout_sumBPweightihstore;
register_Radd_B_2 <= REGdelweightih + cout_sumBPweightihstore; register_Ren_A_2 <= 1; register_Ren_B_2 <= 1;
                register_Radd_A_3 <= REGaddinweight + cout_sumBPweightihstore;
register_Radd_B_3 <= REGdelweightih + cout_sumBPweightihstore; register_Ren_A_3 <= 1; register_Ren_B_3 <= 1;
                register_Radd_A_4 <= REGaddinweight + cout_sumBPweightihstore;
register_Radd_B_4 <= REGdelweightih + cout_sumBPweightihstore; register_Ren_A_4 <= 1; register_Ren_B_4 <= 1;
                register_Radd_A_5 <= REGaddinweight + cout_sumBPweightihstore;
register_Radd_B_5 <= REGdelweightih + cout_sumBPweightihstore; register_Ren_A_5 <= 1; register_Ren_B_5 <= 1;
```

```
                                        register_Radd_A_6       <=      REGaddinweight      +       cout_sumBPweightihstore;
register_Radd_B_6 <= REGdelweightih + cout_sumBPweightihstore; register_Ren_A_6 <= 1; register_Ren_B_6 <= 1;
                                        register_Radd_A_7       <=      REGaddinweight      +       cout_sumBPweightihstore;
register_Radd_B_7 <= REGdelweightih + cout_sumBPweightihstore; register_Ren_A_7 <= 1; register_Ren_B_7 <= 1;
                                        register_Radd_A_8       <=      REGaddinweight      +       cout_sumBPweightihstore;
register_Radd_B_8 <= REGdelweightih + cout_sumBPweightihstore; register_Ren_A_8 <= 1; register_Ren_B_8 <= 1;
                                        register_Radd_A_9       <=      REGaddinweight      +       cout_sumBPweightihstore;
register_Radd_B_9 <= REGdelweightih + cout_sumBPweightihstore; register_Ren_A_9 <= 1; register_Ren_B_9 <= 1;
                                        register_Radd_A_10      <=      REGaddinweight      +       cout_sumBPweightihstore;
register_Radd_B_10 <= REGdelweightih + cout_sumBPweightihstore; register_Ren_A_10 <= 1; register_Ren_B_10 <= 1;
                                        register_Radd_A_11      <=      REGaddinweight      +       cout_sumBPweightihstore;
register_Radd_B_11 <= REGdelweightih + cout_sumBPweightihstore; register_Ren_A_11 <= 1; register_Ren_B_11 <= 1;
                                        register_Radd_A_12      <=      REGaddinweight      +       cout_sumBPweightihstore;
register_Radd_B_12 <= REGdelweightih + cout_sumBPweightihstore; register_Ren_A_12 <= 1; register_Ren_B_12 <= 1;
                                        register_Radd_A_13      <=      REGaddinweight      +       cout_sumBPweightihstore;
register_Radd_B_13 <= REGdelweightih + cout_sumBPweightihstore; register_Ren_A_13 <= 1; register_Ren_B_13 <= 1;
                                        register_Radd_A_14      <=      REGaddinweight      +       cout_sumBPweightihstore;
register_Radd_B_14 <= REGdelweightih + cout_sumBPweightihstore; register_Ren_A_14 <= 1; register_Ren_B_14 <= 1;
                                        register_Radd_A_15      <=      REGaddinweight      +       cout_sumBPweightihstore;
register_Radd_B_15 <= REGdelweightih + cout_sumBPweightihstore; register_Ren_A_15 <= 1; register_Ren_B_15 <= 1;
                                        register_Radd_A_16      <=      REGaddinweight      +       cout_sumBPweightihstore;
register_Radd_B_16 <= REGdelweightih + cout_sumBPweightihstore; register_Ren_A_16 <= 1; register_Ren_B_16 <= 1;
                                        register_Radd_A_17      <=      REGaddinweight      +       cout_sumBPweightihstore;
register_Radd_B_17 <= REGdelweightih + cout_sumBPweightihstore; register_Ren_A_17 <= 1; register_Ren_B_17 <= 1;
                                        register_Radd_A_18      <=      REGaddinweight      +       cout_sumBPweightihstore;
register_Radd_B_18 <= REGdelweightih + cout_sumBPweightihstore; register_Ren_A_18 <= 1; register_Ren_B_18 <= 1;
                                        register_Radd_A_19      <=      REGaddinweight      +       cout_sumBPweightihstore;
register_Radd_B_19 <= REGdelweightih + cout_sumBPweightihstore; register_Ren_A_19 <= 1; register_Ren_B_19 <= 1;
                                        register_Radd_A_20      <=      REGaddinweight      +       cout_sumBPweightihstore;
register_Radd_B_20 <= REGdelweightih + cout_sumBPweightihstore; register_Ren_A_20 <= 1; register_Ren_B_20 <= 1;
                                        register_Radd_A_21      <=      REGaddinweight      +       cout_sumBPweightihstore;
register_Radd_B_21 <= REGdelweightih + cout_sumBPweightihstore; register_Ren_A_21 <= 1; register_Ren_B_21 <= 1;
                                        register_Radd_A_22      <=      REGaddinweight      +       cout_sumBPweightihstore;
register_Radd_B_22 <= REGdelweightih + cout_sumBPweightihstore; register_Ren_A_22 <= 1; register_Ren_B_22 <= 1;
                                        register_Radd_A_23      <=      REGaddinweight      +       cout_sumBPweightihstore;
register_Radd_B_23 <= REGdelweightih + cout_sumBPweightihstore; register_Ren_A_23 <= 1; register_Ren_B_23 <= 1;
                                        register_Radd_A_24      <=      REGaddinweight      +       cout_sumBPweightihstore;
register_Radd_B_24 <= REGdelweightih + cout_sumBPweightihstore; register_Ren_A_24 <= 1; register_Ren_B_24 <= 1;
                                        register_Radd_A_25      <=      REGaddinweight      +       cout_sumBPweightihstore;
register_Radd_B_25 <= REGdelweightih + cout_sumBPweightihstore; register_Ren_A_25 <= 1; register_Ren_B_25 <= 1;
                                        register_Radd_A_26      <=      REGaddinweight      +       cout_sumBPweightihstore;
register_Radd_B_26 <= REGdelweightih + cout_sumBPweightihstore; register_Ren_A_26 <= 1; register_Ren_B_26 <= 1;
                                        register_Radd_A_27      <=      REGaddinweight      +       cout_sumBPweightihstore;
register_Radd_B_27 <= REGdelweightih + cout_sumBPweightihstore; register_Ren_A_27 <= 1; register_Ren_B_27 <= 1;
                                        register_Radd_A_28      <=      REGaddinweight      +       cout_sumBPweightihstore;
register_Radd_B_28 <= REGdelweightih + cout_sumBPweightihstore; register_Ren_A_28 <= 1; register_Ren_B_28 <= 1;
                                        register_Radd_A_29      <=      REGaddinweight      +       cout_sumBPweightihstore;
register_Radd_B_29 <= REGdelweightih + cout_sumBPweightihstore; register_Ren_A_29 <= 1; register_Ren_B_29 <= 1;
                                        register_Radd_A_30      <=      REGaddinweight      +       cout_sumBPweightihstore;
register_Radd_B_30 <= REGdelweightih + cout_sumBPweightihstore; register_Ren_A_30 <= 1; register_Ren_B_30 <= 1;

                                                BP_code <= 2;
                                        end
                                else if (cout_cntBPweightihstore == 21)
                                        begin
                                                start           <= 1;
                                        end
                                else if (cout_cntBPweightihstore == 22)
                                        begin
                                                start <= 0;
                                        end
                                else if (cout_cntBPweightihstore == 30)
                                        begin
                                                register_Ren_A_1 <= 0; register_Ren_B_1 <= 0;
                                                register_Ren_A_2 <= 0; register_Ren_B_2 <= 0;
                                                register_Ren_A_3 <= 0; register_Ren_B_3 <= 0;
                                                register_Ren_A_4 <= 0; register_Ren_B_4 <= 0;
                                                register_Ren_A_5 <= 0; register_Ren_B_5 <= 0;
                                                register_Ren_A_6 <= 0; register_Ren_B_6 <= 0;
                                                register_Ren_A_7 <= 0; register_Ren_B_7 <= 0;
```

```
                register_Ren_A_8 <= 0; register_Ren_B_8 <= 0;
                register_Ren_A_9 <= 0; register_Ren_B_9 <= 0;
                register_Ren_A_10 <= 0; register_Ren_B_10 <= 0;
                register_Ren_A_11 <= 0; register_Ren_B_11 <= 0;
                register_Ren_A_12 <= 0; register_Ren_B_12 <= 0;
                register_Ren_A_13 <= 0; register_Ren_B_13 <= 0;
                register_Ren_A_14 <= 0; register_Ren_B_14 <= 0;
                register_Ren_A_15 <= 0; register_Ren_B_15 <= 0;
                register_Ren_A_16 <= 0; register_Ren_B_16 <= 0;
                register_Ren_A_17 <= 0; register_Ren_B_17 <= 0;
                register_Ren_A_18 <= 0; register_Ren_B_18 <= 0;
                register_Ren_A_19 <= 0; register_Ren_B_19 <= 0;
                register_Ren_A_20 <= 0; register_Ren_B_20 <= 0;
                register_Ren_A_21 <= 0; register_Ren_B_21 <= 0;
                register_Ren_A_22 <= 0; register_Ren_B_22 <= 0;
                register_Ren_A_23 <= 0; register_Ren_B_23 <= 0;
                register_Ren_A_24 <= 0; register_Ren_B_24 <= 0;
                register_Ren_A_25 <= 0; register_Ren_B_25 <= 0;
                register_Ren_A_26 <= 0; register_Ren_B_26 <= 0;
                register_Ren_A_27 <= 0; register_Ren_B_27 <= 0;
                register_Ren_A_28 <= 0; register_Ren_B_28 <= 0;
                register_Ren_A_29 <= 0; register_Ren_B_29 <= 0;
                register_Ren_A_30 <= 0; register_Ren_B_30 <= 0;
            end
        else if (cout_cntBPweightihstore == 31)
            begin
                neuron_out_temp[1] <= neuron_out_1;
                neuron_out_temp[2] <= neuron_out_2;
                neuron_out_temp[3] <= neuron_out_3;
                neuron_out_temp[4] <= neuron_out_4;
                neuron_out_temp[5] <= neuron_out_5;
                neuron_out_temp[6] <= neuron_out_6;
                neuron_out_temp[7] <= neuron_out_7;
                neuron_out_temp[8] <= neuron_out_8;
                neuron_out_temp[9] <= neuron_out_9;
                neuron_out_temp[10] <= neuron_out_10;
                neuron_out_temp[11] <= neuron_out_11;
                neuron_out_temp[12] <= neuron_out_12;
                neuron_out_temp[13] <= neuron_out_13;
                neuron_out_temp[14] <= neuron_out_14;
                neuron_out_temp[15] <= neuron_out_15;
                neuron_out_temp[16] <= neuron_out_16;
                neuron_out_temp[17] <= neuron_out_17;
                neuron_out_temp[18] <= neuron_out_18;
                neuron_out_temp[19] <= neuron_out_19;
                neuron_out_temp[20] <= neuron_out_20;
                neuron_out_temp[21] <= neuron_out_21;
                neuron_out_temp[22] <= neuron_out_22;
                neuron_out_temp[23] <= neuron_out_23;
                neuron_out_temp[24] <= neuron_out_24;
                neuron_out_temp[25] <= neuron_out_25;
                neuron_out_temp[26] <= neuron_out_26;
                neuron_out_temp[27] <= neuron_out_27;
                neuron_out_temp[28] <= neuron_out_28;
                neuron_out_temp[29] <= neuron_out_29;
                neuron_out_temp[30] <= neuron_out_30;
            end
        else if (cout_cntBPweightihstore == 32)
            begin
                register_Wadd_1 <= REGaddinweight + cout_sumBPweightihstore; register_Wen_1 <=
1; register_Wdata_1 <= neuron_out_temp[1];
                register_Wadd_2 <= REGaddinweight + cout_sumBPweightihstore; register_Wen_2 <=
1; register_Wdata_2 <= neuron_out_temp[2];
                register_Wadd_3 <= REGaddinweight + cout_sumBPweightihstore; register_Wen_3 <=
1; register_Wdata_3 <= neuron_out_temp[3];
                register_Wadd_4 <= REGaddinweight + cout_sumBPweightihstore; register_Wen_4 <=
1; register_Wdata_4 <= neuron_out_temp[4];
                register_Wadd_5 <= REGaddinweight + cout_sumBPweightihstore; register_Wen_5 <=
1; register_Wdata_5 <= neuron_out_temp[5];
```

```
                                                register_Wadd_6 <= REGaddinweight + cout_sumBPweightihstore; register_Wen_6 <=
1; register_Wdata_6 <= neuron_out_temp[6];
                                                register_Wadd_7 <= REGaddinweight + cout_sumBPweightihstore; register_Wen_7 <=
1; register_Wdata_7 <= neuron_out_temp[7];
                                                register_Wadd_8 <= REGaddinweight + cout_sumBPweightihstore; register_Wen_8 <=
1; register_Wdata_8 <= neuron_out_temp[8];
                                                register_Wadd_9 <= REGaddinweight + cout_sumBPweightihstore; register_Wen_9 <=
1; register_Wdata_9 <= neuron_out_temp[9];
                                                register_Wadd_10 <= REGaddinweight + cout_sumBPweightihstore; register_Wen_10
<= 1; register_Wdata_10 <= neuron_out_temp[10];
                                                register_Wadd_11 <= REGaddinweight + cout_sumBPweightihstore; register_Wen_11
<= 1; register_Wdata_11 <= neuron_out_temp[11];
                                                register_Wadd_12 <= REGaddinweight + cout_sumBPweightihstore; register_Wen_12
<= 1; register_Wdata_12 <= neuron_out_temp[12];
                                                register_Wadd_13 <= REGaddinweight + cout_sumBPweightihstore; register_Wen_13
<= 1; register_Wdata_13 <= neuron_out_temp[13];
                                                register_Wadd_14 <= REGaddinweight + cout_sumBPweightihstore; register_Wen_14
<= 1; register_Wdata_14 <= neuron_out_temp[14];
                                                register_Wadd_15 <= REGaddinweight + cout_sumBPweightihstore; register_Wen_15
<= 1; register_Wdata_15 <= neuron_out_temp[15];
                                                register_Wadd_16 <= REGaddinweight + cout_sumBPweightihstore; register_Wen_16
<= 1; register_Wdata_16 <= neuron_out_temp[16];
                                                register_Wadd_17 <= REGaddinweight + cout_sumBPweightihstore; register_Wen_17
<= 1; register_Wdata_17 <= neuron_out_temp[17];
                                                register_Wadd_18 <= REGaddinweight + cout_sumBPweightihstore; register_Wen_18
<= 1; register_Wdata_18 <= neuron_out_temp[18];
                                                register_Wadd_19 <= REGaddinweight + cout_sumBPweightihstore; register_Wen_19
<= 1; register_Wdata_19 <= neuron_out_temp[19];
                                                register_Wadd_20 <= REGaddinweight + cout_sumBPweightihstore; register_Wen_20
<= 1; register_Wdata_20 <= neuron_out_temp[20];
                                                register_Wadd_21 <= REGaddinweight + cout_sumBPweightihstore; register_Wen_21
<= 1; register_Wdata_21 <= neuron_out_temp[21];
                                                register_Wadd_22 <= REGaddinweight + cout_sumBPweightihstore; register_Wen_22
<= 1; register_Wdata_22 <= neuron_out_temp[22];
                                                register_Wadd_23 <= REGaddinweight + cout_sumBPweightihstore; register_Wen_23
<= 1; register_Wdata_23 <= neuron_out_temp[23];
                                                register_Wadd_24 <= REGaddinweight + cout_sumBPweightihstore; register_Wen_24
<= 1; register_Wdata_24 <= neuron_out_temp[24];
                                                register_Wadd_25 <= REGaddinweight + cout_sumBPweightihstore; register_Wen_25
<= 1; register_Wdata_25 <= neuron_out_temp[25];
                                                register_Wadd_26 <= REGaddinweight + cout_sumBPweightihstore; register_Wen_26
<= 1; register_Wdata_26 <= neuron_out_temp[26];
                                                register_Wadd_27 <= REGaddinweight + cout_sumBPweightihstore; register_Wen_27
<= 1; register_Wdata_27 <= neuron_out_temp[27];
                                                register_Wadd_28 <= REGaddinweight + cout_sumBPweightihstore; register_Wen_28
<= 1; register_Wdata_28 <= neuron_out_temp[28];
                                                register_Wadd_29 <= REGaddinweight + cout_sumBPweightihstore; register_Wen_29
<= 1; register_Wdata_29 <= neuron_out_temp[29];
                                                register_Wadd_30 <= REGaddinweight + cout_sumBPweightihstore; register_Wen_30
<= 1; register_Wdata_30 <= neuron_out_temp[30];
                                            end
                                        else if (cout_cntBPweightihstore == 34)
                                            begin
                                                register_Wen_1 <= 0;
                                                register_Wen_2 <= 0;
                                                register_Wen_3 <= 0;
                                                register_Wen_4 <= 0;
                                                register_Wen_5 <= 0;
                                                register_Wen_6 <= 0;
                                                register_Wen_7 <= 0;
                                                register_Wen_8 <= 0;
                                                register_Wen_9 <= 0;
                                                register_Wen_10 <= 0;
                                                register_Wen_11 <= 0;
                                                register_Wen_12 <= 0;
                                                register_Wen_13 <= 0;
                                                register_Wen_14 <= 0;
                                                register_Wen_15 <= 0;
                                                register_Wen_16 <= 0;
                                                register_Wen_17 <= 0;
```

```
                                        register_Wen_18 <= 0;
                                        register_Wen_19 <= 0;
                                        register_Wen_20 <= 0;
                                        register_Wen_21 <= 0;
                                        register_Wen_22 <= 0;
                                        register_Wen_23 <= 0;
                                        register_Wen_24 <= 0;
                                        register_Wen_25 <= 0;
                                        register_Wen_26 <= 0;
                                        register_Wen_27 <= 0;
                                        register_Wen_28 <= 0;
                                        register_Wen_29 <= 0;
                                        register_Wen_30 <= 0;
                                end
                end
        19:begin
                        if (cout_cntBPweightho == 1)
                                begin
                                        register_Radd_A_1 <= REGeta; register_Radd_B_1 <= REGdelo + 0; register_Ren_A_1
<= 1; register_Ren_B_1 <= 1;
                                        register_Radd_A_2 <= REGeta; register_Radd_B_2 <= REGdelo + 1; register_Ren_A_2
<= 1; register_Ren_B_2 <= 1;
                                        register_Radd_A_3 <= REGeta; register_Radd_B_3 <= REGdelo + 2; register_Ren_A_3
<= 1; register_Ren_B_3 <= 1;
                                        register_Radd_A_4 <= REGeta; register_Radd_B_4 <= REGdelo + 3; register_Ren_A_4
<= 1; register_Ren_B_4 <= 1;
                                        register_Radd_A_5 <= REGeta; register_Radd_B_5 <= REGdelo + 4; register_Ren_A_5
<= 1; register_Ren_B_5 <= 1;
                                        register_Radd_A_6 <= REGeta; register_Radd_B_6 <= REGdelo + 5; register_Ren_A_6
<= 1; register_Ren_B_6 <= 1;
                                        register_Radd_A_7 <= REGeta; register_Radd_B_7 <= REGdelo + 6; register_Ren_A_7
<= 1; register_Ren_B_7 <= 1;
                                        register_Radd_A_8 <= REGeta; register_Radd_B_8 <= REGdelo + 7; register_Ren_A_8
<= 1; register_Ren_B_8 <= 1;
                                        register_Radd_A_9 <= REGeta; register_Radd_B_9 <= REGdelo + 8; register_Ren_A_9
<= 1; register_Ren_B_9 <= 1;
                                        register_Radd_A_10 <= REGeta; register_Radd_B_10 <= REGdelo + 9;
register_Ren_A_10 <= 1; register_Ren_B_10 <= 1;
                                        register_Radd_A_11 <= REGeta; register_Radd_B_11 <= REGdelo + 10;
register_Ren_A_11 <= 1; register_Ren_B_11 <= 1;
                                        register_Radd_A_12 <= REGeta; register_Radd_B_12 <= REGdelo + 11;
register_Ren_A_12 <= 1; register_Ren_B_12 <= 1;

                                        BP_code <= 6;
                                end
                        else if (cout_cntBPweightho == 4)
                                begin
                                        start <= 1;
                                end
                        else if (cout_cntBPweightho == 5)
                                begin
                                        start <= 0;
                                end
                        else if (cout_cntBPweightho == 13)
                                begin
                                        register_Ren_A_1 <= 0; register_Ren_B_1 <= 0;
                                        register_Ren_A_2 <= 0; register_Ren_B_2 <= 0;
                                        register_Ren_A_3 <= 0; register_Ren_B_3 <= 0;
                                        register_Ren_A_4 <= 0; register_Ren_B_4 <= 0;
                                        register_Ren_A_5 <= 0; register_Ren_B_5 <= 0;
                                        register_Ren_A_6 <= 0; register_Ren_B_6 <= 0;
                                        register_Ren_A_7 <= 0; register_Ren_B_7 <= 0;
                                        register_Ren_A_8 <= 0; register_Ren_B_8 <= 0;
                                        register_Ren_A_9 <= 0; register_Ren_B_9 <= 0;
                                        register_Ren_A_10 <= 0; register_Ren_B_10 <= 0;
                                        register_Ren_A_11 <= 0; register_Ren_B_11 <= 0;
                                        register_Ren_A_12 <= 0; register_Ren_B_12 <= 0;
                                end
                        else if (cout_cntBPweightho == 14)
                                begin
```

```
                                        neuron_out_temp[1] <= neuron_out_1;
                                        neuron_out_temp[2] <= neuron_out_2;
                                        neuron_out_temp[3] <= neuron_out_3;
                                        neuron_out_temp[4] <= neuron_out_4;
                                        neuron_out_temp[5] <= neuron_out_5;
                                        neuron_out_temp[6] <= neuron_out_6;
                                        neuron_out_temp[7] <= neuron_out_7;
                                        neuron_out_temp[8] <= neuron_out_8;
                                        neuron_out_temp[9] <= neuron_out_9;
                                        neuron_out_temp[10] <= neuron_out_10;
                                        neuron_out_temp[11] <= neuron_out_11;
                                        neuron_out_temp[12] <= neuron_out_12;
                                end
                        else if (cout_cntBPweightho == 15)
                                begin
                                        register_Wadd_1 <= REGerror + cout_sumBPweightho; register_Wen_1 <= 1;
register_Wdata_1 <= neuron_out_temp[1];
                                        register_Wadd_2 <= REGerror + cout_sumBPweightho; register_Wen_2 <= 1;
register_Wdata_2 <= neuron_out_temp[2];
                                        register_Wadd_3 <= REGerror + cout_sumBPweightho; register_Wen_3 <= 1;
register_Wdata_3 <= neuron_out_temp[3];
                                        register_Wadd_4 <= REGerror + cout_sumBPweightho; register_Wen_4 <= 1;
register_Wdata_4 <= neuron_out_temp[4];
                                        register_Wadd_5 <= REGerror + cout_sumBPweightho; register_Wen_5 <= 1;
register_Wdata_5 <= neuron_out_temp[5];
                                        register_Wadd_6 <= REGerror + cout_sumBPweightho; register_Wen_6 <= 1;
register_Wdata_6 <= neuron_out_temp[6];
                                        register_Wadd_7 <= REGerror + cout_sumBPweightho; register_Wen_7 <= 1;
register_Wdata_7 <= neuron_out_temp[7];
                                        register_Wadd_8 <= REGerror + cout_sumBPweightho; register_Wen_8 <= 1;
register_Wdata_8 <= neuron_out_temp[8];
                                        register_Wadd_9 <= REGerror + cout_sumBPweightho; register_Wen_9 <= 1;
register_Wdata_9 <= neuron_out_temp[9];
                                        register_Wadd_10 <= REGerror + cout_sumBPweightho; register_Wen_10 <= 1;
register_Wdata_10 <= neuron_out_temp[10];
                                        register_Wadd_11 <= REGerror + cout_sumBPweightho; register_Wen_11 <= 1;
register_Wdata_11 <= neuron_out_temp[11];
                                        register_Wadd_12 <= REGerror + cout_sumBPweightho; register_Wen_12 <= 1;
register_Wdata_12 <= neuron_out_temp[12];
                                end
                        else if (cout_cntBPweightho == 17)
                                begin
                                        register_Wen_1 <= 0;
                                        register_Wen_2 <= 0;
                                        register_Wen_3 <= 0;
                                        register_Wen_4 <= 0;
                                        register_Wen_5 <= 0;
                                        register_Wen_6 <= 0;
                                        register_Wen_7 <= 0;
                                        register_Wen_8 <= 0;
                                        register_Wen_9 <= 0;
                                        register_Wen_10 <= 0;
                                        register_Wen_11 <= 0;
                                        register_Wen_12 <= 0;
                                end
                        else if (cout_cntBPweightho == 18)
                                begin
                                        register_Radd_A_1 <= REGalpha; register_Radd_B_1 <= REGdelweightho +
cout_sumBPweightho; register_Ren_A_1 <= 1; register_Ren_B_1 <= 1;
                                        register_Radd_A_2 <= REGalpha; register_Radd_B_2 <= REGdelweightho +
cout_sumBPweightho; register_Ren_A_2 <= 1; register_Ren_B_2 <= 1;
                                        register_Radd_A_3 <= REGalpha; register_Radd_B_3 <= REGdelweightho +
cout_sumBPweightho; register_Ren_A_3 <= 1; register_Ren_B_3 <= 1;
                                        register_Radd_A_4 <= REGalpha; register_Radd_B_4 <= REGdelweightho +
cout_sumBPweightho; register_Ren_A_4 <= 1; register_Ren_B_4 <= 1;
                                        register_Radd_A_5 <= REGalpha; register_Radd_B_5 <= REGdelweightho +
cout_sumBPweightho; register_Ren_A_5 <= 1; register_Ren_B_5 <= 1;
                                        register_Radd_A_6 <= REGalpha; register_Radd_B_6 <= REGdelweightho +
cout_sumBPweightho; register_Ren_A_6 <= 1; register_Ren_B_6 <= 1;
```

```
                                        register_Radd_A_7 <= REGalpha; register_Radd_B_7 <= REGdelweightho +
cout_sumBPweightho; register_Ren_A_7 <= 1; register_Ren_B_7 <= 1;
                                        register_Radd_A_8 <= REGalpha; register_Radd_B_8 <= REGdelweightho +
cout_sumBPweightho; register_Ren_A_8 <= 1; register_Ren_B_8 <= 1;
                                        register_Radd_A_9 <= REGalpha; register_Radd_B_9 <= REGdelweightho +
cout_sumBPweightho; register_Ren_A_9 <= 1; register_Ren_B_9 <= 1;
                                        register_Radd_A_10 <= REGalpha; register_Radd_B_10 <= REGdelweightho +
cout_sumBPweightho; register_Ren_A_10 <= 1; register_Ren_B_10 <= 1;
                                        register_Radd_A_11 <= REGalpha; register_Radd_B_11 <= REGdelweightho +
cout_sumBPweightho; register_Ren_A_11 <= 1; register_Ren_B_11 <= 1;
                                        register_Radd_A_12 <= REGalpha; register_Radd_B_12 <= REGdelweightho +
cout_sumBPweightho; register_Ren_A_12 <= 1; register_Ren_B_12 <= 1;

                                        BP_code <= 6;
                                end
                        else if (cout_cntBPweightho == 21)
                                begin
                                        start          <= 1;
                                end
                        else if (cout_cntBPweightho == 22)
                                begin
                                        start <= 0;
                                end
                        else if (cout_cntBPweightho == 30)
                                begin
                                        register_Ren_A_1 <= 0; register_Ren_B_1 <= 0;
                                        register_Ren_A_2 <= 0; register_Ren_B_2 <= 0;
                                        register_Ren_A_3 <= 0; register_Ren_B_3 <= 0;
                                        register_Ren_A_4 <= 0; register_Ren_B_4 <= 0;
                                        register_Ren_A_5 <= 0; register_Ren_B_5 <= 0;
                                        register_Ren_A_6 <= 0; register_Ren_B_6 <= 0;
                                        register_Ren_A_7 <= 0; register_Ren_B_7 <= 0;
                                        register_Ren_A_8 <= 0; register_Ren_B_8 <= 0;
                                        register_Ren_A_9 <= 0; register_Ren_B_9 <= 0;
                                        register_Ren_A_10 <= 0; register_Ren_B_10 <= 0;
                                        register_Ren_A_11 <= 0; register_Ren_B_11 <= 0;
                                        register_Ren_A_12 <= 0; register_Ren_B_12 <= 0;
                                end
                        else if (cout_cntBPweightho == 31)
                                begin
                                        neuron_out_temp[1] <= neuron_out_1;
                                        neuron_out_temp[2] <= neuron_out_2;
                                        neuron_out_temp[3] <= neuron_out_3;
                                        neuron_out_temp[4] <= neuron_out_4;
                                        neuron_out_temp[5] <= neuron_out_5;
                                        neuron_out_temp[6] <= neuron_out_6;
                                        neuron_out_temp[7] <= neuron_out_7;
                                        neuron_out_temp[8] <= neuron_out_8;
                                        neuron_out_temp[9] <= neuron_out_9;
                                        neuron_out_temp[10] <= neuron_out_10;
                                        neuron_out_temp[11] <= neuron_out_11;
                                        neuron_out_temp[12] <= neuron_out_12;
                                end
                        else if (cout_cntBPweightho == 32)
                                begin
                                        register_Wadd_1 <= REGdelweightho + cout_sumBPweightho; register_Wen_1 <= 1;
register_Wdata_1 <= neuron_out_temp[1];
                                        register_Wadd_2 <= REGdelweightho + cout_sumBPweightho; register_Wen_2 <= 1;
register_Wdata_2 <= neuron_out_temp[2];
                                        register_Wadd_3 <= REGdelweightho + cout_sumBPweightho; register_Wen_3 <= 1;
register_Wdata_3 <= neuron_out_temp[3];
                                        register_Wadd_4 <= REGdelweightho + cout_sumBPweightho; register_Wen_4 <= 1;
register_Wdata_4 <= neuron_out_temp[4];
                                        register_Wadd_5 <= REGdelweightho + cout_sumBPweightho; register_Wen_5 <= 1;
register_Wdata_5 <= neuron_out_temp[5];
                                        register_Wadd_6 <= REGdelweightho + cout_sumBPweightho; register_Wen_6 <= 1;
register_Wdata_6 <= neuron_out_temp[6];
                                        register_Wadd_7 <= REGdelweightho + cout_sumBPweightho; register_Wen_7 <= 1;
register_Wdata_7 <= neuron_out_temp[7];
```

```
                                        register_Wadd_8 <= REGdelweightho + cout_sumBPweightho; register_Wen_8 <= 1;
register_Wdata_8 <= neuron_out_temp[8];
                                        register_Wadd_9 <= REGdelweightho + cout_sumBPweightho; register_Wen_9 <= 1;
register_Wdata_9 <= neuron_out_temp[9];
                                        register_Wadd_10 <= REGdelweightho + cout_sumBPweightho; register_Wen_10 <= 1;
register_Wdata_10 <= neuron_out_temp[10];
                                        register_Wadd_11 <= REGdelweightho + cout_sumBPweightho; register_Wen_11 <= 1;
register_Wdata_11 <= neuron_out_temp[11];
                                        register_Wadd_12 <= REGdelweightho + cout_sumBPweightho; register_Wen_12 <= 1;
register_Wdata_12 <= neuron_out_temp[12];
                                end
                        else if (cout_cntBPweightho == 34)
                                begin
                                        register_Wen_1 <= 0;
                                        register_Wen_2 <= 0;
                                        register_Wen_3 <= 0;
                                        register_Wen_4 <= 0;
                                        register_Wen_5 <= 0;
                                        register_Wen_6 <= 0;
                                        register_Wen_7 <= 0;
                                        register_Wen_8 <= 0;
                                        register_Wen_9 <= 0;
                                        register_Wen_10 <= 0;
                                        register_Wen_11 <= 0;
                                        register_Wen_12 <= 0;
                                end
                        else if (cout_cntBPweightho == 35)
                                begin


                                        register_Radd_A_1 <= REGerror + cout_sumBPweightho; register_Radd_B_1 <=
REGaddout1 + cout_sumBPweightho-1; register_Ren_A_1 <= 1; register_Ren_B_1 <= 1;
                                        register_Radd_A_2 <= REGerror + cout_sumBPweightho; register_Radd_B_2 <=
REGaddout1 + cout_sumBPweightho-1; register_Ren_A_2 <= 1; register_Ren_B_2 <= 1;
                                        register_Radd_A_3 <= REGerror + cout_sumBPweightho; register_Radd_B_3 <=
REGaddout1 + cout_sumBPweightho-1; register_Ren_A_3 <= 1; register_Ren_B_3 <= 1;
                                        register_Radd_A_4 <= REGerror + cout_sumBPweightho; register_Radd_B_4 <=
REGaddout1 + cout_sumBPweightho-1; register_Ren_A_4 <= 1; register_Ren_B_4 <= 1;
                                        register_Radd_A_5 <= REGerror + cout_sumBPweightho; register_Radd_B_5 <=
REGaddout1 + cout_sumBPweightho-1; register_Ren_A_5 <= 1; register_Ren_B_5 <= 1;
                                        register_Radd_A_6 <= REGerror + cout_sumBPweightho; register_Radd_B_6 <=
REGaddout1 + cout_sumBPweightho-1; register_Ren_A_6 <= 1; register_Ren_B_6 <= 1;
                                        register_Radd_A_7 <= REGerror + cout_sumBPweightho; register_Radd_B_7 <=
REGaddout1 + cout_sumBPweightho-1; register_Ren_A_7 <= 1; register_Ren_B_7 <= 1;
                                        register_Radd_A_8 <= REGerror + cout_sumBPweightho; register_Radd_B_8 <=
REGaddout1 + cout_sumBPweightho-1; register_Ren_A_8 <= 1; register_Ren_B_8 <= 1;
                                        register_Radd_A_9 <= REGerror + cout_sumBPweightho; register_Radd_B_9 <=
REGaddout1 + cout_sumBPweightho-1; register_Ren_A_9 <= 1; register_Ren_B_9 <= 1;
                                        register_Radd_A_10 <= REGerror + cout_sumBPweightho; register_Radd_B_10 <=
REGaddout1 + cout_sumBPweightho-1; register_Ren_A_10 <= 1; register_Ren_B_10 <= 1;
                                        register_Radd_A_11 <= REGerror + cout_sumBPweightho; register_Radd_B_11 <=
REGaddout1 + cout_sumBPweightho-1; register_Ren_A_11 <= 1; register_Ren_B_11 <= 1;
                                        register_Radd_A_12 <= REGerror + cout_sumBPweightho; register_Radd_B_12 <=
REGaddout1 + cout_sumBPweightho-1; register_Ren_A_12 <= 1; register_Ren_B_12 <= 1;

                                        BP_code <= 6;
                                end
                        else if (cout_cntBPweightho == 38)
                                begin
                                        start           <= 1;
                                end
                        else if (cout_cntBPweightho == 39)
                                begin
                                        start <= 0;
                                end
                        else if (cout_cntBPweightho == 47)
                                begin
                                        register_Ren_A_1 <= 0; register_Ren_B_1 <= 0;
                                        register_Ren_A_2 <= 0; register_Ren_B_2 <= 0;
                                        register_Ren_A_3 <= 0; register_Ren_B_3 <= 0;
```

```
                                        register_Ren_A_4 <= 0; register_Ren_B_4 <= 0;
                                        register_Ren_A_5 <= 0; register_Ren_B_5 <= 0;
                                        register_Ren_A_6 <= 0; register_Ren_B_6 <= 0;
                                        register_Ren_A_7 <= 0; register_Ren_B_7 <= 0;
                                        register_Ren_A_8 <= 0; register_Ren_B_8 <= 0;
                                        register_Ren_A_9 <= 0; register_Ren_B_9 <= 0;
                                        register_Ren_A_10 <= 0; register_Ren_B_10 <= 0;
                                        register_Ren_A_11 <= 0; register_Ren_B_11 <= 0;
                                        register_Ren_A_12 <= 0; register_Ren_B_12 <= 0;
                                end
                        else if (cout_cntBPweightho == 48)
                                begin
                                        neuron_out_temp[1] <= neuron_out_1;
                                        neuron_out_temp[2] <= neuron_out_2;
                                        neuron_out_temp[3] <= neuron_out_3;
                                        neuron_out_temp[4] <= neuron_out_4;
                                        neuron_out_temp[5] <= neuron_out_5;
                                        neuron_out_temp[6] <= neuron_out_6;
                                        neuron_out_temp[7] <= neuron_out_7;
                                        neuron_out_temp[8] <= neuron_out_8;
                                        neuron_out_temp[9] <= neuron_out_9;
                                        neuron_out_temp[10] <= neuron_out_10;
                                        neuron_out_temp[11] <= neuron_out_11;
                                        neuron_out_temp[12] <= neuron_out_12;
                                end
                        else if (cout_cntBPweightho == 49)
                                begin
                                        register_Wadd_1 <= REGerror + cout_sumBPweightho; register_Wen_1 <= 1;
register_Wdata_1 <= neuron_out_temp[1];
                                        register_Wadd_2 <= REGerror + cout_sumBPweightho; register_Wen_2 <= 1;
register_Wdata_2 <= neuron_out_temp[2];
                                        register_Wadd_3 <= REGerror + cout_sumBPweightho; register_Wen_3 <= 1;
register_Wdata_3 <= neuron_out_temp[3];
                                        register_Wadd_4 <= REGerror + cout_sumBPweightho; register_Wen_4 <= 1;
register_Wdata_4 <= neuron_out_temp[4];
                                        register_Wadd_5 <= REGerror + cout_sumBPweightho; register_Wen_5 <= 1;
register_Wdata_5 <= neuron_out_temp[5];
                                        register_Wadd_6 <= REGerror + cout_sumBPweightho; register_Wen_6 <= 1;
register_Wdata_6 <= neuron_out_temp[6];
                                        register_Wadd_7 <= REGerror + cout_sumBPweightho; register_Wen_7 <= 1;
register_Wdata_7 <= neuron_out_temp[7];
                                        register_Wadd_8 <= REGerror + cout_sumBPweightho; register_Wen_8 <= 1;
register_Wdata_8 <= neuron_out_temp[8];
                                        register_Wadd_9 <= REGerror + cout_sumBPweightho; register_Wen_9 <= 1;
register_Wdata_9 <= neuron_out_temp[9];
                                        register_Wadd_10 <= REGerror + cout_sumBPweightho; register_Wen_10 <= 1;
register_Wdata_10 <= neuron_out_temp[10];
                                        register_Wadd_11 <= REGerror + cout_sumBPweightho; register_Wen_11 <= 1;
register_Wdata_11 <= neuron_out_temp[11];
                                        register_Wadd_12 <= REGerror + cout_sumBPweightho; register_Wen_12 <= 1;
register_Wdata_12 <= neuron_out_temp[12];
                                end
                        else if (cout_cntBPweightho == 51)
                                begin
                                        register_Wen_1 <= 0;
                                        register_Wen_2 <= 0;
                                        register_Wen_3 <= 0;
                                        register_Wen_4 <= 0;
                                        register_Wen_5 <= 0;
                                        register_Wen_6 <= 0;
                                        register_Wen_7 <= 0;
                                        register_Wen_8 <= 0;
                                        register_Wen_9 <= 0;
                                        register_Wen_10 <= 0;
                                        register_Wen_11 <= 0;
                                        register_Wen_12 <= 0;
                                end
                end
        20:begin
                        if (cout_cntBPweighthostore == 1)
```

```
                begin
                        register_Radd_A_1 <= REGerror + cout_sumBPweighthostore; register_Radd_B_1 <=
REGdelweightho + cout_sumBPweighthostore; register_Ren_A_1 <= 1; register_Ren_B_1 <= 1;
                        register_Radd_A_2 <= REGerror + cout_sumBPweighthostore; register_Radd_B_2 <=
REGdelweightho + cout_sumBPweighthostore; register_Ren_A_2 <= 1; register_Ren_B_2 <= 1;
                        register_Radd_A_3 <= REGerror + cout_sumBPweighthostore; register_Radd_B_3 <=
REGdelweightho + cout_sumBPweighthostore; register_Ren_A_3 <= 1; register_Ren_B_3 <= 1;
                        register_Radd_A_4 <= REGerror + cout_sumBPweighthostore; register_Radd_B_4 <=
REGdelweightho + cout_sumBPweighthostore; register_Ren_A_4 <= 1; register_Ren_B_4 <= 1;
                        register_Radd_A_5 <= REGerror + cout_sumBPweighthostore; register_Radd_B_5 <=
REGdelweightho + cout_sumBPweighthostore; register_Ren_A_5 <= 1; register_Ren_B_5 <= 1;
                        register_Radd_A_6 <= REGerror + cout_sumBPweighthostore; register_Radd_B_6 <=
REGdelweightho + cout_sumBPweighthostore; register_Ren_A_6 <= 1; register_Ren_B_6 <= 1;
                        register_Radd_A_7 <= REGerror + cout_sumBPweighthostore; register_Radd_B_7 <=
REGdelweightho + cout_sumBPweighthostore; register_Ren_A_7 <= 1; register_Ren_B_7 <= 1;
                        register_Radd_A_8 <= REGerror + cout_sumBPweighthostore; register_Radd_B_8 <=
REGdelweightho + cout_sumBPweighthostore; register_Ren_A_8 <= 1; register_Ren_B_8 <= 1;
                        register_Radd_A_9 <= REGerror + cout_sumBPweighthostore; register_Radd_B_9 <=
REGdelweightho + cout_sumBPweighthostore; register_Ren_A_9 <= 1; register_Ren_B_9 <= 1;
                        register_Radd_A_10 <= REGerror + cout_sumBPweighthostore; register_Radd_B_10 <=
REGdelweightho + cout_sumBPweighthostore; register_Ren_A_10 <= 1; register_Ren_B_10 <= 1;
                        register_Radd_A_11 <= REGerror + cout_sumBPweighthostore; register_Radd_B_11 <=
REGdelweightho + cout_sumBPweighthostore; register_Ren_A_11 <= 1; register_Ren_B_11 <= 1;
                        register_Radd_A_12 <= REGerror + cout_sumBPweighthostore; register_Radd_B_12 <=
REGdelweightho + cout_sumBPweighthostore; register_Ren_A_12 <= 1; register_Ren_B_12 <= 1;

                        BP_code <= 2;
                end
        else if (cout_cntBPweighthostore == 4)
                begin
                        start           <= 1;
                end
        else if (cout_cntBPweighthostore == 5)
                begin
                        start <= 0;
                end
        else if (cout_cntBPweighthostore == 13)
                begin
                        register_Ren_A_1 <= 0; register_Ren_B_1 <= 0;
                        register_Ren_A_2 <= 0; register_Ren_B_2 <= 0;
                        register_Ren_A_3 <= 0; register_Ren_B_3 <= 0;
                        register_Ren_A_4 <= 0; register_Ren_B_4 <= 0;
                        register_Ren_A_5 <= 0; register_Ren_B_5 <= 0;
                        register_Ren_A_6 <= 0; register_Ren_B_6 <= 0;
                        register_Ren_A_7 <= 0; register_Ren_B_7 <= 0;
                        register_Ren_A_8 <= 0; register_Ren_B_8 <= 0;
                        register_Ren_A_9 <= 0; register_Ren_B_9 <= 0;
                        register_Ren_A_10 <= 0; register_Ren_B_10 <= 0;
                        register_Ren_A_11 <= 0; register_Ren_B_11 <= 0;
                        register_Ren_A_12 <= 0; register_Ren_B_12 <= 0;
                end
        else if (cout_cntBPweighthostore == 14)
                begin
                        neuron_out_temp[1] <= neuron_out_1;
                        neuron_out_temp[2] <= neuron_out_2;
                        neuron_out_temp[3] <= neuron_out_3;
                        neuron_out_temp[4] <= neuron_out_4;
                        neuron_out_temp[5] <= neuron_out_5;
                        neuron_out_temp[6] <= neuron_out_6;
                        neuron_out_temp[7] <= neuron_out_7;
                        neuron_out_temp[8] <= neuron_out_8;
                        neuron_out_temp[9] <= neuron_out_9;
                        neuron_out_temp[10] <= neuron_out_10;
                        neuron_out_temp[11] <= neuron_out_11;
                        neuron_out_temp[12] <= neuron_out_12;
                end
        else if (cout_cntBPweighthostore == 15)
                begin
                        register_Wadd_1 <= REGdelweightho + cout_sumBPweighthostore; register_Wen_1 <=
1; register_Wdata_1 <= neuron_out_temp[1];
```

```
                                        register_Wadd_2 <= REGdelweightho + cout_sumBPweighthostore; register_Wen_2 <=
1; register_Wdata_2 <= neuron_out_temp[2];
                                        register_Wadd_3 <= REGdelweightho + cout_sumBPweighthostore; register_Wen_3 <=
1; register_Wdata_3 <= neuron_out_temp[3];
                                        register_Wadd_4 <= REGdelweightho + cout_sumBPweighthostore; register_Wen_4 <=
1; register_Wdata_4 <= neuron_out_temp[4];
                                        register_Wadd_5 <= REGdelweightho + cout_sumBPweighthostore; register_Wen_5 <=
1; register_Wdata_5 <= neuron_out_temp[5];
                                        register_Wadd_6 <= REGdelweightho + cout_sumBPweighthostore; register_Wen_6 <=
1; register_Wdata_6 <= neuron_out_temp[6];
                                        register_Wadd_7 <= REGdelweightho + cout_sumBPweighthostore; register_Wen_7 <=
1; register_Wdata_7 <= neuron_out_temp[7];
                                        register_Wadd_8 <= REGdelweightho + cout_sumBPweighthostore; register_Wen_8 <=
1; register_Wdata_8 <= neuron_out_temp[8];
                                        register_Wadd_9 <= REGdelweightho + cout_sumBPweighthostore; register_Wen_9 <=
1; register_Wdata_9 <= neuron_out_temp[9];
                                        register_Wadd_10 <= REGdelweightho + cout_sumBPweighthostore; register_Wen_10
<= 1; register_Wdata_10 <= neuron_out_temp[10];
                                        register_Wadd_11 <= REGdelweightho + cout_sumBPweighthostore; register_Wen_11
<= 1; register_Wdata_11 <= neuron_out_temp[11];
                                        register_Wadd_12 <= REGdelweightho + cout_sumBPweighthostore; register_Wen_12
<= 1; register_Wdata_12 <= neuron_out_temp[12];
                                end
                        else if (cout_cntBPweighthostore == 17)
                                begin
                                        register_Wen_1 <= 0;
                                        register_Wen_2 <= 0;
                                        register_Wen_3 <= 0;
                                        register_Wen_4 <= 0;
                                        register_Wen_5 <= 0;
                                        register_Wen_6 <= 0;
                                        register_Wen_7 <= 0;
                                        register_Wen_8 <= 0;
                                        register_Wen_9 <= 0;
                                        register_Wen_10 <= 0;
                                        register_Wen_11 <= 0;
                                        register_Wen_12 <= 0;
                                end
                        else if (cout_cntBPweighthostore == 18)
                                begin
                                        register_Radd_A_1       <=      REGaddinweight2      +      cout_sumBPweighthostore;
register_Radd_B_1 <= REGdelweightho + cout_sumBPweighthostore; register_Ren_A_1 <= 1; register_Ren_B_1 <= 1;
                                        register_Radd_A_2       <=      REGaddinweight2      +      cout_sumBPweighthostore;
register_Radd_B_2 <= REGdelweightho + cout_sumBPweighthostore; register_Ren_A_2 <= 1; register_Ren_B_2 <= 1;
                                        register_Radd_A_3       <=      REGaddinweight2      +      cout_sumBPweighthostore;
register_Radd_B_3 <= REGdelweightho + cout_sumBPweighthostore; register_Ren_A_3 <= 1; register_Ren_B_3 <= 1;
                                        register_Radd_A_4       <=      REGaddinweight2      +      cout_sumBPweighthostore;
register_Radd_B_4 <= REGdelweightho + cout_sumBPweighthostore; register_Ren_A_4 <= 1; register_Ren_B_4 <= 1;
                                        register_Radd_A_5       <=      REGaddinweight2      +      cout_sumBPweighthostore;
register_Radd_B_5 <= REGdelweightho + cout_sumBPweighthostore; register_Ren_A_5 <= 1; register_Ren_B_5 <= 1;
                                        register_Radd_A_6       <=      REGaddinweight2      +      cout_sumBPweighthostore;
register_Radd_B_6 <= REGdelweightho + cout_sumBPweighthostore; register_Ren_A_6 <= 1; register_Ren_B_6 <= 1;
                                        register_Radd_A_7       <=      REGaddinweight2      +      cout_sumBPweighthostore;
register_Radd_B_7 <= REGdelweightho + cout_sumBPweighthostore; register_Ren_A_7 <= 1; register_Ren_B_7 <= 1;
                                        register_Radd_A_8       <=      REGaddinweight2      +      cout_sumBPweighthostore;
register_Radd_B_8 <= REGdelweightho + cout_sumBPweighthostore; register_Ren_A_8 <= 1; register_Ren_B_8 <= 1;
                                        register_Radd_A_9       <=      REGaddinweight2      +      cout_sumBPweighthostore;
register_Radd_B_9 <= REGdelweightho + cout_sumBPweighthostore; register_Ren_A_9 <= 1; register_Ren_B_9 <= 1;
                                        register_Radd_A_10      <=      REGaddinweight2      +      cout_sumBPweighthostore;
register_Radd_B_10 <= REGdelweightho + cout_sumBPweighthostore; register_Ren_A_10 <= 1; register_Ren_B_10 <= 1;
                                        register_Radd_A_11      <=      REGaddinweight2      +      cout_sumBPweighthostore;
register_Radd_B_11 <= REGdelweightho + cout_sumBPweighthostore; register_Ren_A_11 <= 1; register_Ren_B_11 <= 1;
                                        register_Radd_A_12      <=      REGaddinweight2      +      cout_sumBPweighthostore;
register_Radd_B_12 <= REGdelweightho + cout_sumBPweighthostore; register_Ren_A_12 <= 1; register_Ren_B_12 <= 1;

                                        BP_code <= 2;
                                end
                        else if (cout_cntBPweighthostore == 21)
                                begin
                                        start           <= 1;
```

```
                end
        else if (cout_cntBPweighthostore == 22)
                begin
                        start <= 0;
                end
        else if (cout_cntBPweighthostore == 30)
                begin
                        register_Ren_A_1 <= 0; register_Ren_B_1 <= 0;
                        register_Ren_A_2 <= 0; register_Ren_B_2 <= 0;
                        register_Ren_A_3 <= 0; register_Ren_B_3 <= 0;
                        register_Ren_A_4 <= 0; register_Ren_B_4 <= 0;
                        register_Ren_A_5 <= 0; register_Ren_B_5 <= 0;
                        register_Ren_A_6 <= 0; register_Ren_B_6 <= 0;
                        register_Ren_A_7 <= 0; register_Ren_B_7 <= 0;
                        register_Ren_A_8 <= 0; register_Ren_B_8 <= 0;
                        register_Ren_A_9 <= 0; register_Ren_B_9 <= 0;
                        register_Ren_A_10 <= 0; register_Ren_B_10 <= 0;
                        register_Ren_A_11 <= 0; register_Ren_B_11 <= 0;
                        register_Ren_A_12 <= 0; register_Ren_B_12 <= 0;
                end
        else if (cout_cntBPweighthostore == 31)
                begin
                        neuron_out_temp[1] <= neuron_out_1;
                        neuron_out_temp[2] <= neuron_out_2;
                        neuron_out_temp[3] <= neuron_out_3;
                        neuron_out_temp[4] <= neuron_out_4;
                        neuron_out_temp[5] <= neuron_out_5;
                        neuron_out_temp[6] <= neuron_out_6;
                        neuron_out_temp[7] <= neuron_out_7;
                        neuron_out_temp[8] <= neuron_out_8;
                        neuron_out_temp[9] <= neuron_out_9;
                        neuron_out_temp[10] <= neuron_out_10;
                        neuron_out_temp[11] <= neuron_out_11;
                        neuron_out_temp[12] <= neuron_out_12;
                end
        else if (cout_cntBPweighthostore == 32)
                begin
                        register_Wadd_1 <= REGaddinweight2 + cout_sumBPweighthostore; register_Wen_1
<= 1; register_Wdata_1 <= neuron_out_temp[1];
                        register_Wadd_2 <= REGaddinweight2 + cout_sumBPweighthostore; register_Wen_2
<= 1; register_Wdata_2 <= neuron_out_temp[2];
                        register_Wadd_3 <= REGaddinweight2 + cout_sumBPweighthostore; register_Wen_3
<= 1; register_Wdata_3 <= neuron_out_temp[3];
                        register_Wadd_4 <= REGaddinweight2 + cout_sumBPweighthostore; register_Wen_4
<= 1; register_Wdata_4 <= neuron_out_temp[4];
                        register_Wadd_5 <= REGaddinweight2 + cout_sumBPweighthostore; register_Wen_5
<= 1; register_Wdata_5 <= neuron_out_temp[5];
                        register_Wadd_6 <= REGaddinweight2 + cout_sumBPweighthostore; register_Wen_6
<= 1; register_Wdata_6 <= neuron_out_temp[6];
                        register_Wadd_7 <= REGaddinweight2 + cout_sumBPweighthostore; register_Wen_7
<= 1; register_Wdata_7 <= neuron_out_temp[7];
                        register_Wadd_8 <= REGaddinweight2 + cout_sumBPweighthostore; register_Wen_8
<= 1; register_Wdata_8 <= neuron_out_temp[8];
                        register_Wadd_9 <= REGaddinweight2 + cout_sumBPweighthostore; register_Wen_9
<= 1; register_Wdata_9 <= neuron_out_temp[9];
                        register_Wadd_10 <= REGaddinweight2 + cout_sumBPweighthostore; register_Wen_10
<= 1; register_Wdata_10 <= neuron_out_temp[10];
                        register_Wadd_11 <= REGaddinweight2 + cout_sumBPweighthostore; register_Wen_11
<= 1; register_Wdata_11 <= neuron_out_temp[11];
                        register_Wadd_12 <= REGaddinweight2 + cout_sumBPweighthostore; register_Wen_12
<= 1; register_Wdata_12 <= neuron_out_temp[12];
                end
        else if (cout_cntBPweighthostore == 34)
                begin
                        register_Wen_1 <= 0;
                        register_Wen_2 <= 0;
                        register_Wen_3 <= 0;
                        register_Wen_4 <= 0;
                        register_Wen_5 <= 0;
                        register_Wen_6 <= 0;
```

```
                register_Wen_7 <= 0;
                register_Wen_8 <= 0;
                register_Wen_9 <= 0;
                register_Wen_10 <= 0;
                register_Wen_11 <= 0;
                register_Wen_12 <= 0;
            end
        end
    21:begin
            if (cout_cntCheck == 1)
                begin
                    if ((cout_sumEPOC == EPOC) || ((stop == 1)&&(cout_sumCheck == 3)))
                        begin
                            trainOK <= 1;
                        end
                end
            else if (cout_cntCheck == 3)
                begin
                    if (cout_sumCheck == 3)
                        begin
                            register_Wadd_1 <= REGerrorcheck; register_Wen_1 <= 1;
register_Wdata_1 <= 0;
                        end
                end
            else if (cout_cntCheck == 6)
                begin
                    if (cout_sumCheck == 3)
                        begin
                            register_Wen_1 <= 0;
                        end
                end
            else if (cout_cntCheck == 10)
                begin
                    if (cout_sumCheck == 3)
                        begin
                            cout_sumEPOC <= cout_sumEPOC + 1;
                        end
                    else
                        begin
                            cout_sumEPOC <= cout_sumEPOC;
                        end
                end
            else if (cout_cntCheck == 20)
                begin
                    cout_sumCheck <= cout_sumCheck + 1;
                end
        end
    /*              22:begin
                        end
                    23:begin
                        end
                    24:begin
                        end
                    25:begin
                        end
                    26:begin
                        end

                    default
                        begin
                        end*/
                endcase
            end
        end

    endmodule
```

BIBLIOGRAPHY

[1] A. Maton, "Human Biology and Health," Englewood Cliffs, New Jersey: Prentice Hall, 1993.

[2] World Health Organization (WHO). "Cardiovascular diseases", in Feb, 2007, website available at http://www.who.int/mediacentre/factsheets/fs317/en/

[3] The Wall Street Journal, Health section, in Nov 16th, 2009, website available on: http://online.wsj.com/article/SB125833149978449651.html

[4] Michael O. Sweeney el at., "Adverse Effect of Ventricular Pacing on Heart Failure and Atrial Fibrillation Among Patients With Normal Baseline QRS Duration in a Clinical Trial of Pacemaker Therapy for Sinus Node Dysfunction," *Journal of American Heart Association*, vol.107, pp.2932-2937, Jun. 2003.

[5] Hilbel, Thomas; Thomas M Helms, Gerd Mikus, Hugo A Katus, Christian Zugck (01/10/2008). "Telemetry in the clinical setting". Herzschrittmachertherapie & Elektrophysiologie 19 (3): 146–64. ISSN 0938-7412. Retrieved 2009-Aug-04, available at https://www.researchgate.net/publication/23423927_Telemetry_in_the_clinical_setting

[6] Anuradha, B. and Reddy, V.C.Veera. "Cardiac arrhythmia classification using fuzzy classifiers," *Journal of Theoretical and Applied Information Techn.*, vol. 4, no. 4, pp. 353-359, 2008.

[7] M. Kundu, M. Nasipuri, and D.K. Basu. "A Knowledge-Based Approach to ECG Interpretation Using Fuzzy Logic," *IEEE Trans. on Syst., Man and Cybern.* Part B, vol. 28, no. 2, pp. 237-243, Apr. 1998.

[8] W. T. Cheng and K. L. Chan, "Classification of electrocardiogram using hidden markov models," 20th Annual International Conference of the IEEE Engineering in Medicine and Biology society, Vol. 20, No 1, 1998.

[9] Baum, E. B. What size net gives valid generalization? Neural Computation Vol. 1, pp. 151-160, 1989.

[10] G. Cottrell, P. Munro and D. Zipser, "Learning internal representations of gray scale images: an example of extensional programming," Proc. 9th Annual Conference of the Cognitive Science Society, Seattle, Washington, Jul, 1987.

[11] M. Bianchina, P. Frasconi and M. Gori, “Learning in multilayered networks used as auto-associators,” *IEEE Transaction on Neural Networks*, vol. 6, No. 2, Mar, 1995.

[12] H. Bourland and M. Gori, “Auto-association by multilayer perceptrons and singular value decomposition,” *Journal of Biological Cybernetics*, vol. 59, No. 4-5, Sep, 1988.

[13] R. Watrous and G. Towell, “A patient-adaptive neural network ECG patient monitoring algorithm,” *Computers in Cardiology*, Vienna, Austria, pp. 10-13, Sep, 1995.

[14] Guangchen Liu and Mei Song, “Application of a BP neural network based on principal component analysis in ECG diagnosis of the right ventricular hypertrophy,” 2nd International Conference on Information and Computing Science, 2009.

[15] M. H. Song, J. Lee, H. D. Park and K. J. Lee, “Classification of heartbeats based on linear discriminant analysis and artificial neural network,” 27th Annual Conference of IEEE on Engineering in Medicine and Biology, Sep, 2005.

[16] Alireza Behrad and Karim Facz, “New method for QRS-wave recognition in ECG using MART neural network,” Seventh Australian and New Zealand Intelligent Information Systems Conference, Nov, 2001.

[17] Yutaka Fukuoka, Miho Fukuhara and Akimasa Ishida, “Stress assessment based on ECG using neural networks,” 18th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Amsterdam, 1996.

[18] G.Bortolan et al., “Diagnostic ECG Classification Based on Neural Networks,” Journal of Electrocardiology, Vol. 26, pp. 75-79, 1994.

[19] Dipti Itchhaporia, Peter B. Snow, Robert J. Almassy and William J. Oetgen, “Artificial neural networks: Current status in cardiovascular medicine,” *Journal of the American College of* Cardiology, vol. 28, pp. 515-521, Aug. 1996.

[20] Ronald M. Birse, Rev. Patricia and E. Knowlden, “Oxford Dictionary of National Biography,” 2004.

[21] A. D. Waller, "A demonstration on man of electromotive changes accompanying the heart's beat". Journal of Physiol., London vol. 8, pp. 229–34

[22] Moises Rivera-Ruiz, Christian Cajavilca and Joseph Varon, “Einthoven’s string galvanometer: The first electrocardiograph,” Texas Heart Institute Journal, vol. 35, pp.174-178, 2008.

[23] J. Cooper, "Electrocardiography 100 years ago. Origins, pioneers, and contributors". N Engl J Med 315 (7): 461–4.1986.

[24] Website available at http://library.med.utah.edu/kw/ecg/ecg_outline/Lesson1/lead_dia.html

[25] Welch Allyn, Advancing Frontline Care, Introduction online website available at http://www.welchallyn.com/documents/Cardiopulmonary/Electrocardiographs/PC-Based%20Exercise%20Stress%20ECG/poster_110807_pcexerecg.pdf

[26] Kevin Gurney and Kevin N. Burney, "An Introduction to Neural Networks," 1997.

[27] I. Aleksander and H. Morton, "An Introduction to Neural Computing," Chapman and Hall, 1990.

[28] S. Haykin, "Neural Networks: A Comprehensive Foundation," Prentice Hall Press, 1998.

[29] S. Himavathi el at., "Feedforward Neural Network Implementation in FPGA Using Layer Multiplexing for Effective Resource Utilization," Neural Networks, vol.18, pp.880-888, May. 2007.

[30] D. Ferrer el at., "NeuroFPGA -- Implementing Artificial Neural Networks on Programmable Logic Devices," Design, Automation and Test in Europe, vol. 3, pp. 30218, 2004.

[31] P. Domingos el at., "An Efficient and Scalable Architecture for Neural Networks with backpropagation Learning," Field Programmable Logic and Applications, pp. 89-94, Aug. 2005.

[32] J. Eldredge el at., "Density Enhancement of a Neural Network Using FPGA and Run-Time Reconfiguration," IEEE Workshop on FPGAs for Custom Computing Machines, pp.180-188, Apr. 1994.

[33] M. Giulioni el at., "A configurable analog VLSI neural network with spiking neurons and self-regulating plastic synapses which classifies overlapping patterns," available at: http://books.nips.cc/papers/files/nips20/NIPS2007_0707.pdf

[34] R.W. Newcomb, "Analog VLSI Neural Networks," in Handbook of Brain Theory and Neural Networks, Bradford Books, MIT Press, 1995.

[35] P.M. Rautaharju el at., "NOVACODE serial ECG classification system for clinical trials and epidemiologic studies," *Division of Cardiology*, vol.24, pp.179-187, 1992.

[36] P. Bozzola el at., "A Hybrid Neuron-Fuzzy System for ECG Classification of Myocardial Infarction," *Computers in Cardiology*, pp.241-244, Sep. 1996.

[37] W.T. Cheng and K.L. Chan, "Classification of electro-cardiogram using hidden Markov models," Engineering in Medicine and Biology Society, vol.1, pp.143-146, Oct. 1998.

[38] Yu Hen el at., "Applications of Artificial Neural Networks for ECG Signal Detection and Classification," Journal of Electro-cardiology, vol.26, pp.66-73, 1993.

[39] Website available at Harvard-MIT Division of Health Sciences and Technology, available at http://www.physionet.org

[40] P. Masa el at., “High Speed VLSI Neural Network for High-Energy Physics,” Microelectronics for Neural Networks and Fuzzy Systems, vol.26-28, pp.422-428, Sep. 1994.

[41] Website available at http: http://www.hpl.hp.com/research/cacti/