Towards Current-Mode Analog Implementation of Deep Neural Network Functions

by

Shihao Wang

Submitted in partial fulfillment of the requirements
for the degree of Master of Applied Science

at

Dalhousie University
Halifax, Nova Scotia
November 2021

Table of Contents

List of Tables

List of Figures

Abstract

This thesis proposes a CMOS circuit design that implements various Deep Neural Network functions. DNN has been used in different applications such as image identification and speech recognition. However, the existing techniques suffer from the high-power consumption of digital implementation. The proposed design presents a low power consumption analog implementation of various DNN functions in the subthreshold region. In this thesis, the implemented circuit blocks include the binary weight multiplier layer, Rectified Linear Unit, and approximate Softmax layer. The proposed designs were implemented in an accurate and efficient method.

This design is implemented using 180nm CMOS technology with a 1.5V power supply. Furthermore, the impact of the proposed design on accuracy was simulated using the MNIST dataset. Using a four-layer Convolutional Neural Network with an 8 bits resolution, the design achieved an accuracy of 99.02% with 68.21uW power consumption, which is 35.65% lower than the existing analog DNN design.

List of Abbreviations Used

CMOS	Complementary Metal Oxide Semiconductor
DAC	Digital to Analog Converter
ReLU	Rectified Linear Unit
DNN	Deep Neural Network
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
CM	Current Mirror
PCB	Printed Circuit Board
PWL	Piecewise Linear Function

Acknowledgment

I would like to give my warmest thanks to everyone who assisted me during my graduate studies.

First and foremost, I would like to express my sincere thanks to my supervisor, Dr. Kamal El-Sankary, who made this research work possible. His professional advice carried me through all the stages of my project. Moreover, he encourages me when I face difficulties during my study. I would also like to thank Dr. Karama AL-Tamimi and Dr. Issam Hammad for being my co-supervisor and providing me with such a wonderful project.

Also, I would like to express my sincere thanks to Dr. Gu and Dr. Phillips for being my committee members, for letting my defense be an enjoyable moment, and for your brilliant suggestions.

Finally, I would like to thank my family for their assistance during my life; without their love, I would not be able to achieve my goals. Best regards to all my dear friends for their continuous support throughout my life.

CHAPTER 1 INTRODUCTION

1.1 Background

Neural networks are widely used in many fields, such as engineering, science, and business. Deep neural network (DNN), which is a special kind of neural network, could be implemented using interconnected artificial neurons to mimic the process of decision-making in the human brain [1]-[2]. Therefore, it is one of the promising techniques in different fields, such as image identification, speech recognition, etc. [3]-[6]. In this sense, a neural network works similarly to the natural human brain, and it contains multiple layers to analyze a series of data. A simple neural network consists of three layers: input layer, hidden layer, and output layer. Each layer of the neural network contains multiple nodes, seen as "neurons" in DNN. A neuron node is a mathematical function that collects and classifies input data information [7]. These nodes are interconnected, and the interconnections of each node to other nodes are called weight and bias, which means that when input is transmitted to the next layer, the weight and bias are applied. Figure 1 shows the structure of the superficial three layers neural network. They are transforming the sum of weight and input based on the specific activation functions. The activation function performs non-linear operations to mimic the complex real-world property. The ability to learn and model non-linear relationships is the key advantage of the neural network [8]. Currently, digital design is used as a mainstream method to implement DNNs on-chip; however, most digital implementations of DNN have the shortcoming of relatively high-power consumption [11] – [15].

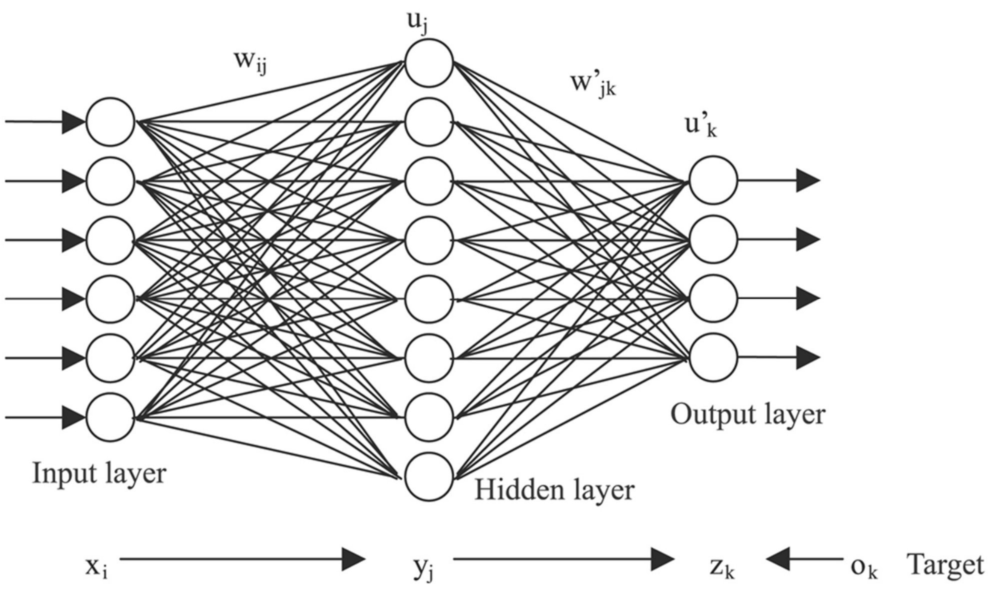


Figure 1. The basic structure of the neural network [9]

In Contrast to digital implementation, current-mode subthreshold analog circuit implementation of DNNs has been gaining attention for its advantages in terms of design simplicity, parallel processing for a significant number of signals, and low power consumption [14][16]. DNN design consists of several functional blocks. Implementing each function using the analog design is necessary to build the entire network in subthreshold current mode. The proposed design includes a weight multiplier, ReLU (Rectified Linear Unit) activation function, and Softmax activation function.

1.2 Contributions

According to the discussion above, the thesis proposes a current-mode DNN design that can satisfy low power consumption and is highly efficient. The proposed circuit aims to satisfy each functional layer's performance, low power consumption, and against PVT variations at the circuit level. The design achieved a high accuracy at the system level when tested in an entire DNN network [29]. The contribution of this thesis can be summarized as follow:

1. A tunable binary weight multiplier by using current mirror gain is proposed. The binary form of weight factor can be separated as MSB bits and LSB bits. Thus, weight multiplier results include negative parts and positive parts.
2. A ReLU function layer is proposed using a piecewise current-mode linear circuit through the curve shifting method.
3. An approximate Softmax layer is proposed through the current-mode divider circuit under the subthreshold region to work at low power.

1.3 Organization

The organization of this thesis can be summarized as follow:

In Chapter 2, the architecture background will be discussed, including definition, regular architecture, and applications of each functional layer.

Chapter 3 focused on the literature review of various implementation methods for each functional layer and discussed their advantages and disadvantages.

Chapter 4 introduced about proposed design based on the mathematical relationship and layout design of the proposed DNN. Four parts will separate the proposed analog circuit design: tunable weight multiplier, PWL ReLU activation function, and approximate Softmax function.

In Chapter 5, the discussion will concentrate on the implementation on both circuit level and system level, and testing results of the proposed DNN design. Also, this chapter will include a comparison with state-of-the-art.

Chapter 6 presented the future work and summed up the whole design.

CHAPTER 2 ARCHITECTURE BACKGROUND

This chapter will discuss the architecture background, including definition, regular architecture, and applications of each functional layer.

2.1 Definition and application

An artificial neural network is a web that includes a series of interconnected artificial nodes, and its theory is based on the inspiration of the human brain. In this case, many scientists concluded that the brain's learning process is through changes of a large amount of synapse [1]. Thus, the weight factor can be referred to as synapse in DNNs. As the critical computing element in DNN, weight exists in the connections between presynaptic neurons and postsynaptic neurons. Expect the weight factor. The bias factor is the other learnable parameter inside the neural network. During the training process of DNNs, weights and biases are adjusted in the short term to respond to an immediate learning stimulus. In general, initial weight and bias factors are random numbers in the specific range before the learning process. During the learning process, learnable parameters are adjusted toward the desired values simultaneously. Thus, weight factors influence the value of the output of the weight multiplier. On the other hand, bias values compensate for the difference between the testing and ideal values. In the proposed design, the role of the bias value can be replaced by a custom compensating function; therefore, corresponding to the structure of the neural network, which is the fully connected network. The basic mathematical equation of the weight multiplier layer can be expressed as (1):

$$y = \sum_{i=1}^{N}(X_i \times Weight) + bias \tag{1}$$

where Xi is the value of the input element, combined with equation 1, the proposed weight multiplier aims to calculate the sum of weight by using a CMOS analog circuit. The schematic diagram of the weight multiplier layer can be seen in Figure 2.

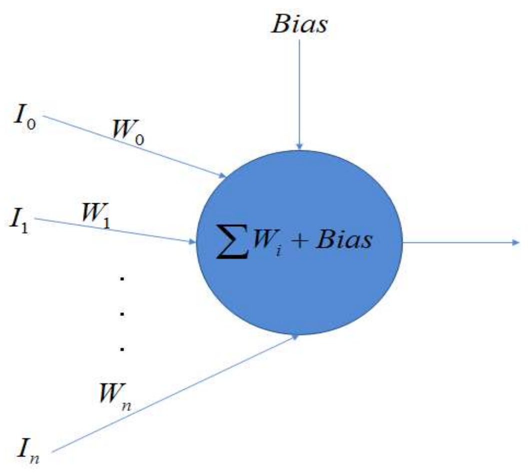


Figure 2. The basic structure of the weight multiplier

DNNs include various activation function layers to generate the non-linear output, which finds the optimal performance of the loss function during the testing process. The activation function layer aims to transform the summed weighted input signal to define a specific output signal that helps the network learn complex patterns in the data. Therefore, the output signal of DNNs will become the real-world information with the activation function layer, such as image, video, and sound.

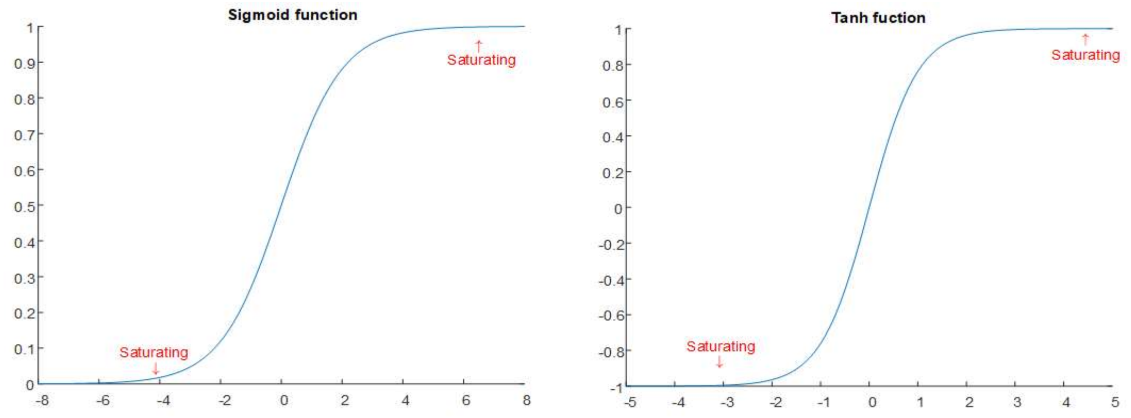
Three essential activation functions are widely used in all deep learning architectures as the hidden layer, shown in Table 1. Compared with these three functions, the ReLU (Rectified Linear Unit) function is the most outstanding among these basic functions. There are two main reasons why the ReLU function can rapidly become the default activation function in various neural networks. Firstly, the ReLu function has a simple mathematical relationship, which means implementing the ReLU function can save much power by using less complex operation blocks. Thus, ReLU function can be widely used in the low power design of DNNs. Secondly, ReLU function can solve the vanishing gradient problem. Gradient plays a vital role in the neural network design, which can measure variable changes and then impact the update of the weight value. The equation related to gradient and weight can be expressed as (2):

Table 1. Equations of activation function

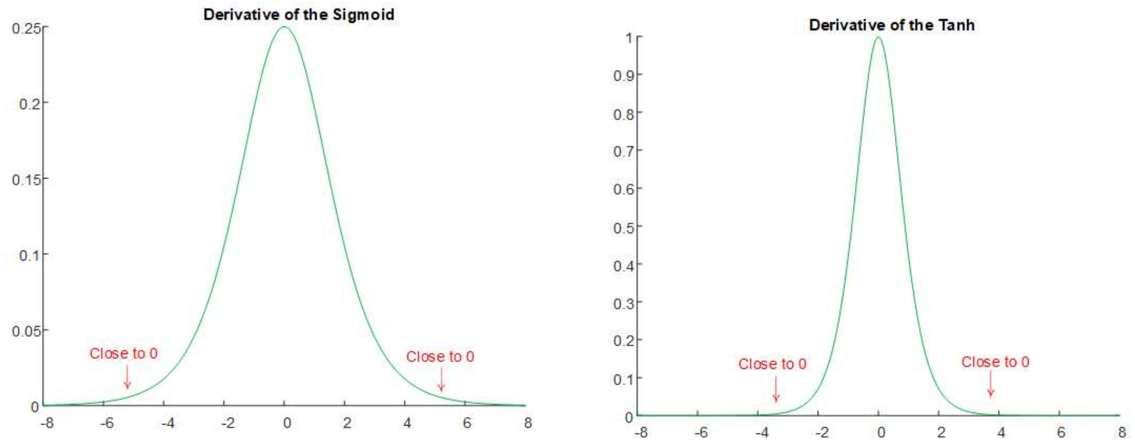
Function	Equation
Sigmoid	$f(x) = \frac{1}{1+e^{-x}}$
Tanh	$f(x) = \frac{2}{1+e^{-2x}} - 1$
ReLU	$f(x) = Max(0, x)$

$$W_{x =} W_x - \propto \left(\frac{\partial Error}{\partial W_x}\right) \tag{2}$$

where W_x is the initial weight value, $\propto$ is the learning rate of the neural network. The vanishing problem will occur when the derivative term of equation 2 is too small, which means that there are no significant changes during the testing process. Both sigmoid and tanh functions have saturating regions. Therefore, the derivative function of these two activation functions will move through a region with zero gradients, called the “vanishing region”. The mathematical performance and the relative derivate function can be seen in Figure 3.



(a)



(b)

Figure 3 (a) Sigmoid function and Tanh function (b) derivative function of Sigmoid and Tanh function

The ReLU function can solve the vanishing gradient problem based on its specific mathematical performance and related derivative function compared with these two activation functions. For the mathematical relationship of ReLU function, if the ReLu function's input signal is positive, the output will be a linear function. Otherwise, the output signal will remain zero. Thus, ReLU function does not have the saturated region (vanishing region). Also, the gradients function of ReLU only has two conditions: zero condition and relatively high condition, as illustrated in Figure 4.

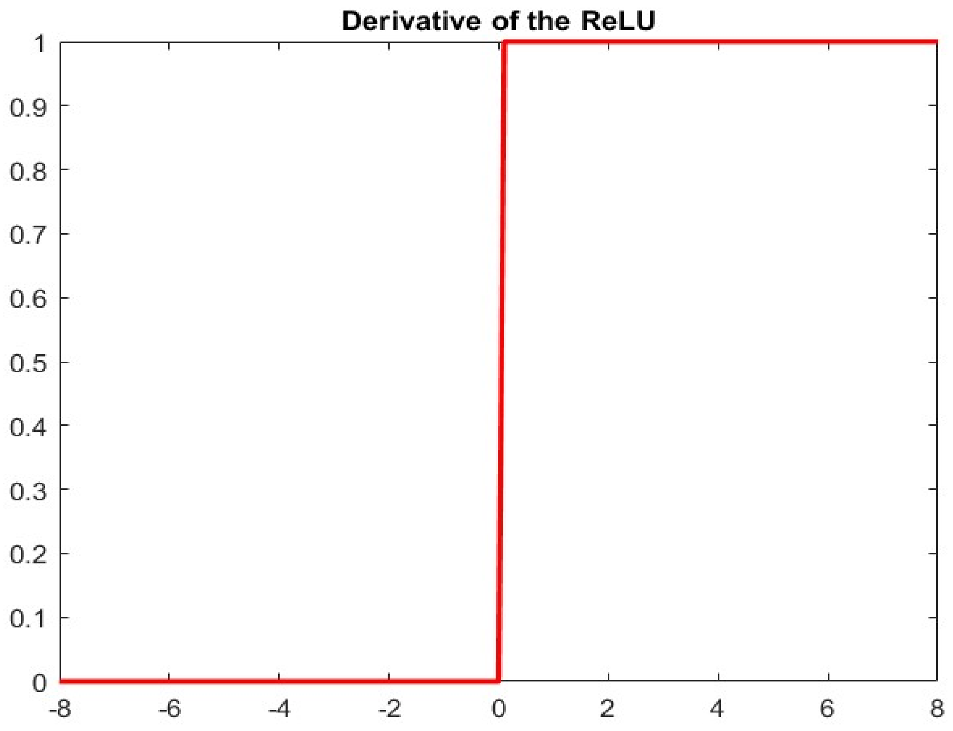


Figure 4. Derivative of ReLU function

By observing Fig.4, there is no vanishing problem of the ReLU activation function, and the gradient is either 1 or 0 during the weight updating process. Therefore, as discussed above, ReLU activation function is the better function during the activating process of the DNNs. Meanwhile, one layer's outputs can be seen as the input signal as the next layer. Thus, according to the architecture, the DNNs, ReLU layer is usually used in the hidden layer and activates the nodes after the weight multiplier layer.

The purpose of the output layer in DNNs is to perform the result through the whole network design. Therefore, the result can reflect the performance of the proposed DNN design. Thus, A suitable output layer of the DNN needs to go through the entire design and find the best performance neuron during the iterative process; therefore, the output layer needs to satisfy two conditions: normalizing the multi-classification model and the result can be interpreted as a probability. Thus, Softmax functional layer can be selected as the output layer of DNNs. The calculated output probabilities of Softmax are in the range between 0 to 1, and the sum of all the probabilities is equal to 1. Additionally, if the input element has a high value, this input element will contain a higher output probability after applying the Softmax function, which follows the principle of statistics. Also, the mathematical equation of the Softmax function has the summation term of the exponential element, which means that the Softmax function will traversal each input element.

2.2 Architecture

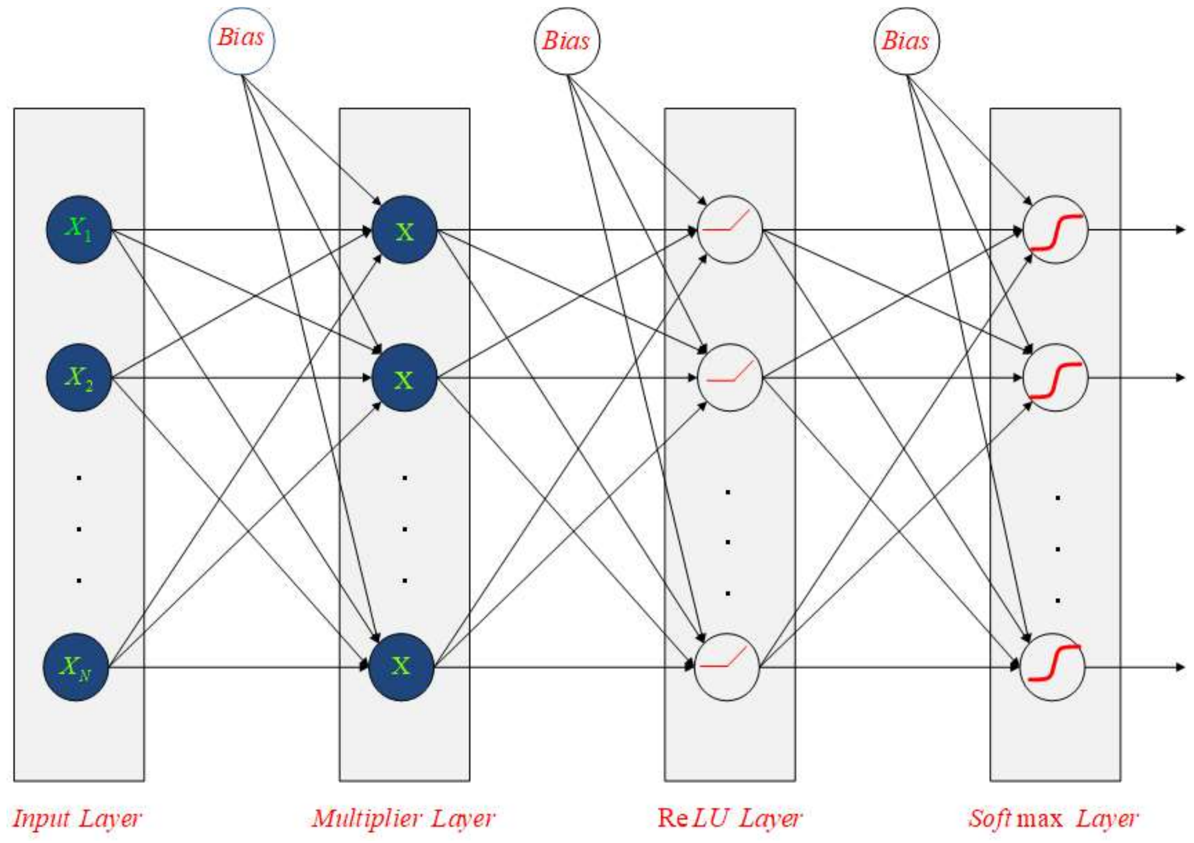


Figure 5. The architecture of DNN design

Fig.5 shows the schematic of the architecture of the DNN design. The scheme has four distinct layers mapped in series. The output of one layer can be seen as the input signal as the next layer. The weight multiplier layer provides the basic multiplication operation for the input elements and weight components. ReLU activation function can distinguish either positive signal or negative signal and then transfer to the output layer. Finally, the Softmax layer presents the probability of each element and defines which one contains the highest probability during the testing process of the DNN. The following chapter will evaluate the different implementation methods of each functional layer and then discuss the advantage of the proposed design.

CHAPTER 3 LITERATURE REVIEW

Chapter 3 will focus on the literature review of various implementation methods for each functional layer and discuss their advantages and disadvantages. Also, the challenges of each functional layer are introduced.

3.1 Weight Multiplier Function

As the essential computation element of DNN, multiplication is performed between the input layer and the corresponding weight elements in neural networks. Therefore, the results after the multiplier will directly affect the whole system's precision and power consumption. The analog implementation method has the advantages of low delay and low power consumption [16][17].

The most common implementation method of the multiplier is the four-quadrant current-mode analog multiplier. The design principle is keeping CMOS transistors operated in the subthreshold region. Based on the mathematical relationship, the aim is to use the MOSFET subthreshold region property to build the V-I antilog cell, which is related to the logarithm and exponent function, to create the log-antilog multiplier [14]. The output result equation can be expressed as:

$$I_{out} = K * I_1 I_2 \tag{3}$$

Where K is the independent controller, I_1, I_2 are two input current signals. Therefore, the four-quadrant current-mode multiplier can perfectly perform the multiplication process in the neural network. Figure 5 shows the post-layout results of the DC transfer characteristics. Besides, equation 3 performs as PVT independent result, which means unaffected by environmental conditions.

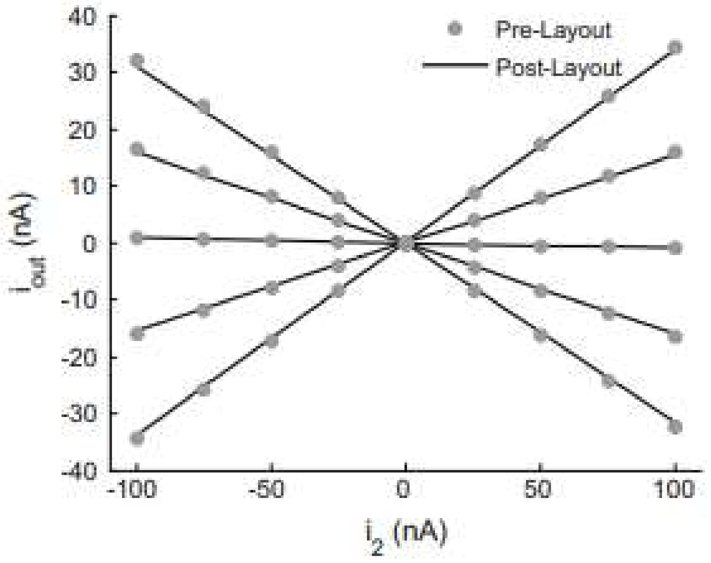


Figure 6. DC transfer characteristics of the four-quadrant multiplier [14]

The second standard multiplier implementation method is based on varying the widths of CMOS transistors to control the value of weights [4]. Compared with the four-quadrant current-mode analog multiplier, this implementation method only has one input signal, and the weight vectors can be seen as the gain factor. The weight vectors are implemented by varying the widths of the relevant CMOS transistor. The multiplying step is summing the current signal through a relevant input transistor by using a diode-connected load. Figure 7 shows the transistor schematic of implementing the published weight multiplier.

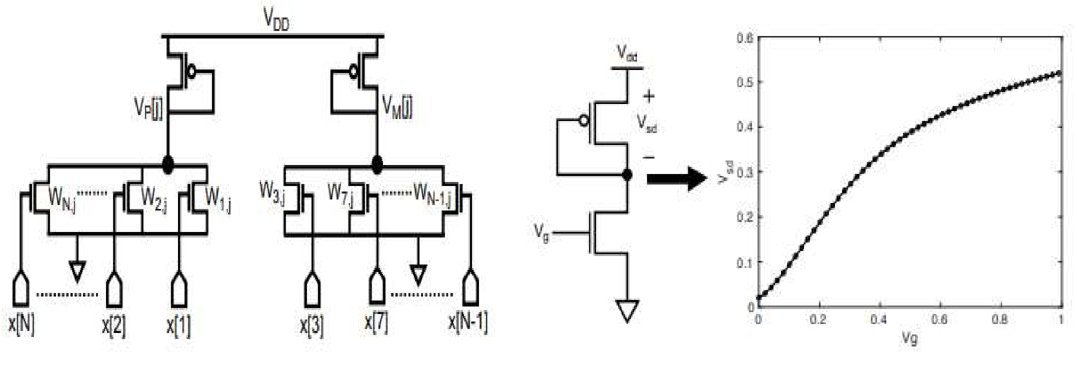


Figure 7. Transistor level of the existing weight multiplier [4]

where $X_{[i]}$ are the input vectors, $W_{[i,j]}$ are the weight vectors of the multiplier.

This building block is followed by the common-source amplifier (CS) with a PMOS diode load. The CS method has two main advantages. First, the weight vectors can be split into two halves, either positive or negative. Second, the CS implementation method can provide the non-linear characteristic between the drain-source voltage of the PMOS load and the input voltage vectors, which is friendly in the activation function simulation of DNNs.

Inspired by the CS implementation method, the current mirror implementation method of weight multiplier can also provide input and weight vectors separately, following the structure of Figure 1. [18] proposed an analog vector-matrix multiplier for DNNs. In addition, multiplication operation was implemented based on in-memory computing architecture, where learning weights are stored in the single-poly floating-gate cells embedded in current mirrors.

Combined with these three standard implementation methods of weight multiplier in DNNs, the tunable weight multiplier was proposed based on the current mirror theory. To simplify the design process, the proposed weight multiplier unit directly performs the weight vector as the current gain of the relevant CMOS transistor. Related to the software implementation, The proposed weight multiplier can also be performed as quantized activation, which includes binary representations of the weight factors. It can be separated as 3 MSB bits and 3 LSB bits. Inspired by the CS implementation method, the proposed weight multiplier also includes negative parts and positive parts.

3.2 ReLU Activation Function

ReLU function has a specific mathematical relationship, defined as $y = \max(0, x)$, $x \in R$. Theoretically, the ReLU function can be implemented based on its input-output characteristics, which means ReLU function can be seen as the PWL function during the implementation process. Meanwhile, the output of the ReLU function will follow by the performance of the input signal when the input signal is greater than 0. Thus, inspired by the character of the source follower, the ReLU function can be implemented by its similar curve. The approximate implementation method by using source follower is shown in Figure 8. To avoid the significant shift of the input voltage of the source follower, [19] uses the suitable value of capacitor C1 to filter out part of DC voltage.

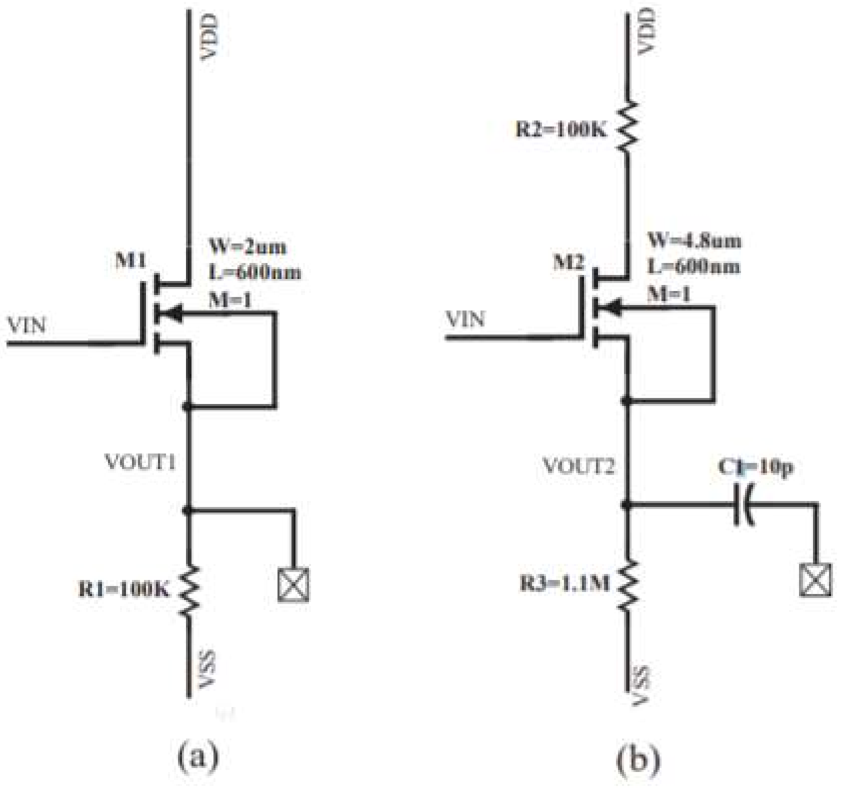


Figure 8. ReLU function based on source follower [19]

In this approach, a similar curve of the ReLU function can be implemented; however, the output curve cannot perfectly match the input signal due to the inevitable voltage loss. Also, this approach has the saturation region when the gate-source voltage reaches a specific value. The drain current becomes constant when the number of carriers in the transistor channel no longer increases.

3.3 Softmax Activation Function

In general, the implementation method of the activation function relies on its specific mathematical relationship. For example, as the output layer associated with probability, the Softmax mathematical function includes a divider and exponential units.

The analog implementation of the Softmax function is different from the ReLU function implementation method, as discussed above. Compared with the implementation of ReLU activation function, the analog implementation of the SoftMax function is quite complex. The straightforward design of SoftMax proposed by Yuan in [20] has two main drawbacks. It may cause high accuracy loss and overflow problems [20]. In this case, the approximation SoftMax function is the popular implementation method in DNNs.

The exponential unit and the normalizing ratio are the most complex and expensive calculation elements [21]. Zunino and Gastaldo presented the Taylor series approximation method to replace the exponential unit of the SoftMax function [22]. The relevant mathematical function of the Taylor series of the exponential unit is defined by:

$$f_2^{(Taylor)} = a + cx^2 \tag{4}$$

According to the Taylor series method, the current-squaring circuit with a bias source can easily implement the exponential unit. After achieving the approximate exponential unit, all of the quantities are positive after the exponential unit. Therefore, the normalizing ratio can be implemented by using the adder to sum all exponential units as a typical denominator value and using a divider circuit to compute the relevant ratio between each input element.

The other standard implemented method of approximate Softmax is mathematical transformation, which simplifies the original Softmax function. Wang *et al* implemented the mathematical transformation of the SoftMax function involving exponential and logarithmic operation [21]. Theoretically, the mathematical transformation of exponential operations is based on the multiple multiplication operations. Therefore, the relevant exponential function can be expressed as:

$$e^{y_i} = 2^{y_i * \, log_2^e} \tag{5}$$

Where log element can be simplified by using adder and subtracter, expressed as “ADD-ADD-SUB” operation. The proposed design of the exponential unit is shown in Figure 9.

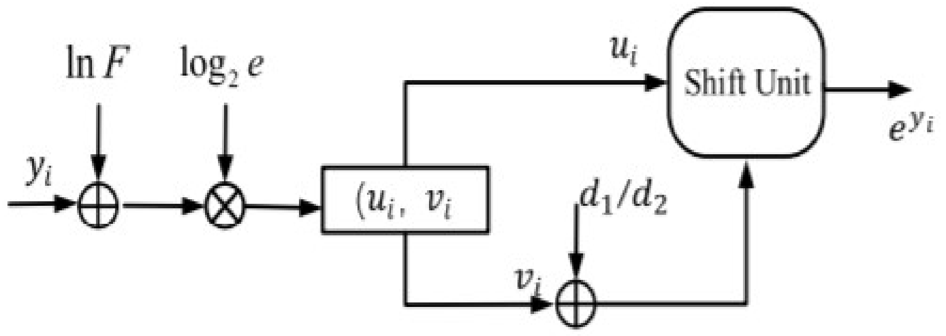


Figure 9. Softmax function based on mathematical transformation [21]

These two designs are suitable for the approximate Softmax design because of the advantage of a similar mathematical relationship as the regular Softmax function. However, these two methods will consume long delay and high-power consumption based on multiple transformation operations in exponential elements. Therefore, in the proposed design, the implementation method will consider in two fields: mathematical relationship and neural network structure relationship.

CHAPTER 4 PROPOSED DESIGN

This chapter presents the proposed design. Four parts will separate the proposed analog circuit design: tunable weight multiplier, PWL ReLU activation function, and approximate Softmax function.

4.1 Tunable Weight Multiplier Function

According to the MOSFET subthreshold region property, the source-to-drain current (I_{SD}) is given by

$$I_{SD} = I_{Do}\frac{W}{L}\exp\left(\frac{V_{gs}}{nkT/q}\right) \tag{6}$$

Where I_{Do} is the leakage current of the MOSFET, $\frac{W}{L}$ is the transistor aspect ratio of the relevant transistor, Vgs is the gate-source voltage, n is the subthreshold slope factor, K is the Boltzmann constant, T is the temperature in degree Kelvin, and q is a charge of an electron.

In current-mode circuits, the weight gain can be interpreted as the scaling factor of input current. Thus, the proposed design uses the current mirror gain principle to perform weight gain. The CM primary implementation method is detailed in Figure 10,

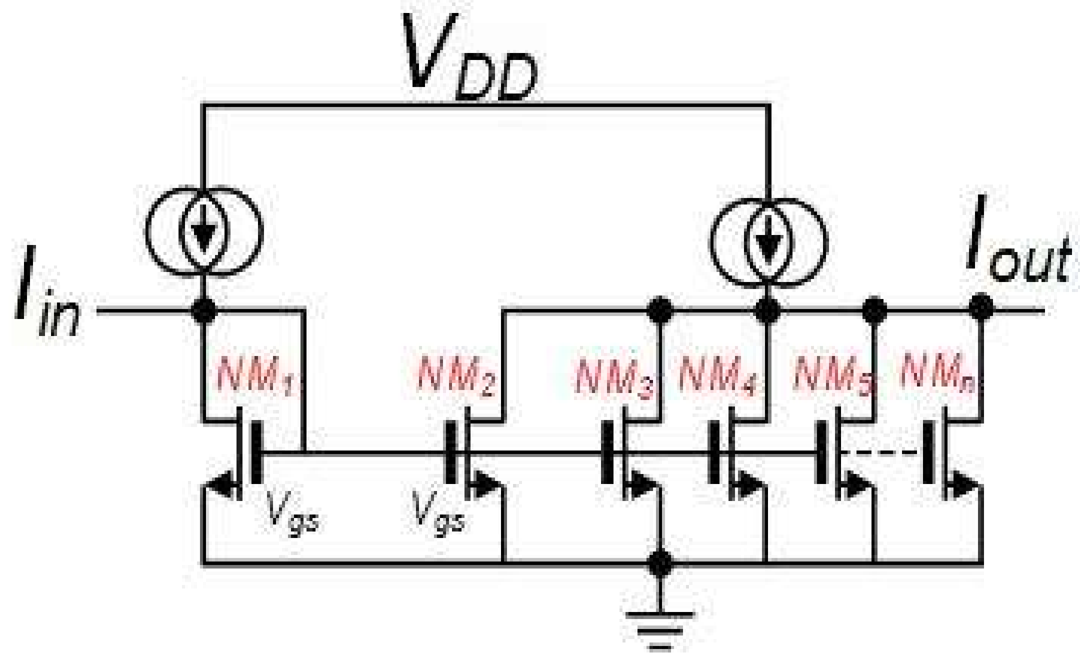


Figure 10. CM primary implementation circuit

By considering the CM function cell, the input current and output current of cell can be expressed in (7) and (8), respectively,

$$I_{SD,in} = I_{Do}\left(\frac{W}{L}\right)_1 \exp\left(\frac{V_{gs}}{nKT/q}\right) \tag{7}$$

$$I_{SD,out} = I_{Do}\left(\frac{W}{L}\right)_2 \exp\left(\frac{V_{gs}}{nKT/q}\right) + I_{Do}\left(\frac{W}{L}\right)_3 \exp\left(\frac{V_{gs}}{nKT/q}\right) + \cdots$$

$$+ I_{Do}\left(\frac{W}{L}\right)_n \exp\left(\frac{V_{gs}}{nKT/q}\right) \tag{8}$$

Where V_{gs} voltages are all the same in the CM approach, the only variable in between (7) and (8) is the term $\frac{W}{L}$. To keep the proposed design analysis straightforward, the aspect ratio of each output transistor is represented using different scaling factors of the input transistor aspect ratio $(\frac{W}{L})_1$. The equation can be expressed as demonstrated in (9).

$$(\frac{W}{L})_i = \propto_i * (\frac{W}{L})_1 \tag{9}$$

Where *i*= 2,3, . . ., n and $\propto_i$= a, b, c, . . . is a gain coefficient, based on (2) and (3), the ratio of output current and input current can be obtained as:

$$\frac{I_{SD,out}}{I_{SD,in}} = \frac{I_{Do}\left(\frac{W}{L}\right)_2 \exp\left(\frac{V_{gs}}{nKT/q}\right) + I_{Do}\left(\frac{W}{L}\right)_3 \exp\left(\frac{V_{gs}}{nKT/q}\right) + \cdots + I_{Do}\left(\frac{W}{L}\right)_n \exp\left(\frac{V_{gs}}{nKT/q}\right)}{I_{Do}\left(\frac{W}{L}\right)_1 \exp\left(\frac{V_{gs}}{nKT/q}\right)}$$

$$= \frac{(\frac{W}{L})_2 + (\frac{W}{L})_3 + \cdots + (\frac{W}{L})_n}{(\frac{W}{L})_1}$$

$$= \frac{a*(\frac{W}{L})_1 + b*(\frac{W}{L})_1 + \cdots + n*(\frac{W}{L})_1}{(\frac{W}{L})_1}$$

$$= a + b + \cdots + n \tag{10}$$

The elements in (10) are the scaling factor, we obtain $K = a + b + \cdots + n$, then I_{out} is given as (11).

$$I_{out} = I_{in} \times K \tag{11}$$

The output of the suggested weight multiplier is obtained by summing over all terms of CM blocks.

$$I_{out,WM} = \sum_{i=1}^{n} I_{out} = \sum_{i=1}^{n} (I_{in}\ x\ Weight) \tag{12}$$

The CM approach has two significant shortcomings. First, the CM approach can provide only one specific value of weight. Second, the CM approach cannot provide sufficient accuracy for the output. Therefore, the CM approach is revised with the tunable current level and tunable bulk voltage in the circuit. The proposed tunable CM output block of the weight multiplier circuit is shown in Figure 11.

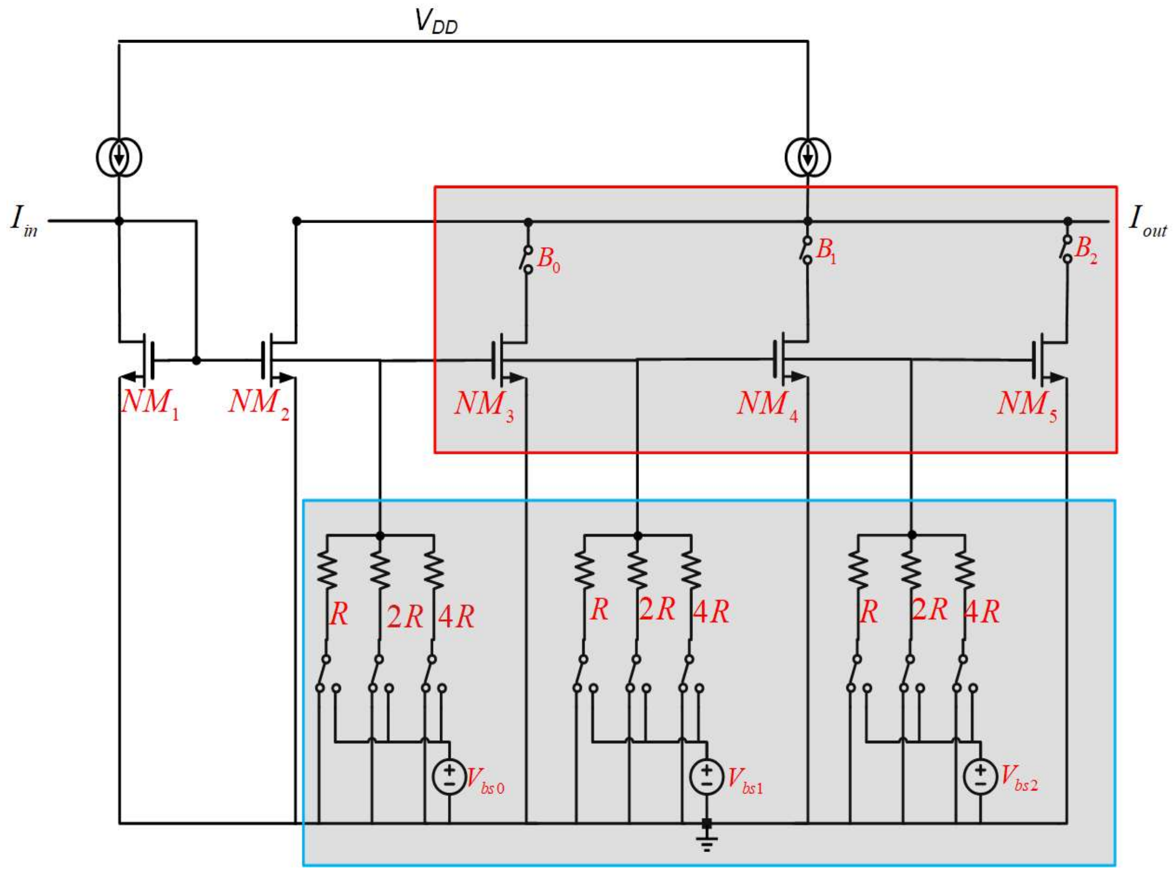


Figure 11. Tunable 6 bits CM Output Block

Tunable values are performed by using a switch to control the current level change in the circuit. Meanwhile, weight can be seen as a binary number through high and low current levels. Thus, it is accessible in coding and reduces fixed point multiplication [18]. To achieve more accurate weight gain, three simple Digital-to-Analog Converter (DAC) were added to the circuits to control the bulk voltage of each transistor. According to the MOSFET subthreshold equation related to V_{gs}, V_{th}, and V_{bs} :

$$I_{SD}\ (V_{gs}, V_{th}, V_{bs}) = I_{Do} \exp\left(\frac{(V_{gs}-V_{th})+(n-1)V_{bs}}{nkT/q}\right) \tag{13}$$

Where V_{bs} is the bulk-source voltage. Body effect should be considered when we provide voltage to the bulk terminal. The body effect equation can be expressed as:

$$V_{th} = V_{th0} + \Upsilon(\sqrt{|2\phi F + V_{sb}|} - \sqrt{|2\phi F|}) \tag{14}$$

Where V_{th0} is the threshold voltage for zero substrate bias, Υ denotes the body effect coefficient. According to (13) and (14), it is obvious to obtain the relationship between bulk voltage and I_{SD}. Therefore, the additional bulk voltage will increase I_{SD} in the proposed design. Thus, a simple DAC circuit is presented to the bulk terminal of the transistors. The value of resistors are R, 2R and 4R, which correspond to the binary weights of Bit 2 ($2^2 = 4$), Bit 1 ($2^1 = 2$), and Bit 0 ($2^0 = 1$). The aim is to correct the weight gain to two decimal places. Thus, the proposed 6 bits binary weight multiplier is composed of 3 parallel transistors and 3 DAC circuits, where 3 parallel transistors can be seen as 3 bits MSB and 3 DAC circuits can be interpreted as 3 bits LSB. Additionally, the proposed weight gain is a signed value to distinguish negative or positive weight gain. NM_5 in the positive multiplier circuit is keeping a high level. In the negative multiplier circuit, NM_5 is always a low level.

4.2 ReLU Activation Function Based on PWL Circuit

The implementation of the ReLu function is based on the relevant mathematical relationship. If the input signal of the ReLu function is positive, the output will be a linear function. Otherwise, the output signal will remain zero. It can be formulated as y = max(0,x) function.

The ReLu function can be seen as a piecewise linear function (PWL) with a breakpoint when the input current is 0 [24].

The proposed current-mode PWL function in Fig.3, without the highlighted shifting block, is a half-wave rectifier with a positive current breakpoint located at I_1. Thus, the mathematical relationship can be expressed as:

$$Iout(Iin, ReLu) = \begin{cases} 0, \ Iin < I1 \\ m(Iin, ReLu - I1), \ Iin \geq I1 \end{cases} \tag{15}$$

where Iin,ReLu is the input current of the ReLu layer, which is the output current from the previous layer (weight multiplier). m is the aspect ratio of the dimension of the output current mirror NM9/NM8.

Based on the layer-to-layer structure of the proposed design, the input current and output current of ReLU can be expressed in (16) and (17), respectively.

$$Iin, ReLu = N * Iin \tag{16}$$

$$Iout(Iin) = \begin{cases} 0, \ Iin < I1 \\ m(N * Iin - I1), \ Iin \geq I1 \end{cases} \tag{17}$$

Where N is the total multiplying factor of the weight multi-plier. A shift operation is used to adjust the breakpoint to zero. The implementation method is to left-shift the original curve to make a breakpoint located at (0,0). Therefore, according to curve translation theory, the mathematical relationship of the proposed ReLU function is revised. This relationship is given by (18):

$$Iout(Iin + S) = \begin{cases} 0, \; Iin < 0 \\ m(N(Iin + S) - I1), \; Iin \geq 0 \end{cases} \tag{18}$$

The ReLU function’s critical coordinate is (0,0), where S is the breakpoint’s shifting unit. Thus, it is obvious to obtain the value of S can be expressed by (19)

$$S = \frac{I1}{N} \tag{19}$$

We provided an additional pmos current mirror to the circuit. The providing current is called KI1. The ratio value of K can be given by

$$K = \frac{(\frac{W}{L})_{PM6}}{(\frac{W}{L})_{PM5}} \tag{20}$$

A proportional relationship between K and S is existed based on the scaling factor of the input current range αI_{in}. The formula can be expressed as

$$\frac{S}{\alpha I_{in}} = \frac{1}{K} \tag{21}$$

From (21), the exact dimension value of the additional pmos current mirror is generated; Therefore, the ReLu design is illustrated in Fig. 11.

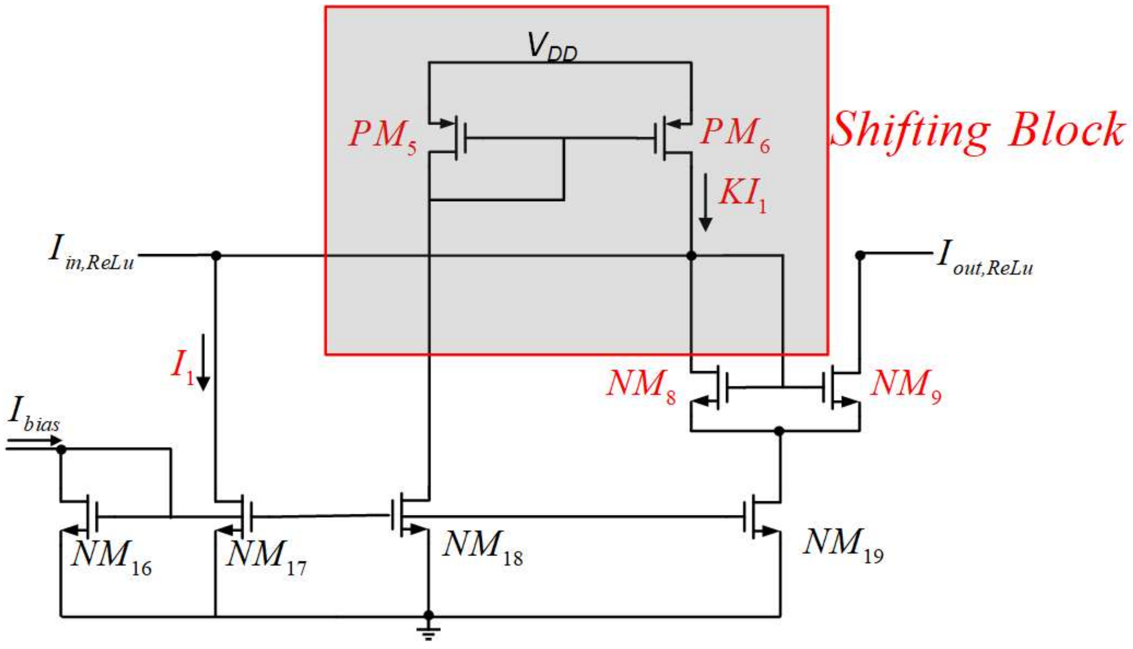


Figure 12. Proposed circuit of ReLU function

4.3 Approximate Softmax Activation Function

Based on the mathematical characteristics of the Softmax function, the implementation method includes multiple multiplications and divisions [21]. Complex mathematical operations cause high power consumption, long delays, and difficulty to design in hardware implementation. The proposed design provides an approximate method by using property analysis and mathematical analysis. Softmax function contains two specific features in DNNs: nonlinearity characteristic and limited output range. If an implementation method can simultaneously satisfy these two specific features, this function will be considered the approximate SoftMax function.

According to the mathematical characteristics of the Softmax function, the output range of Softmax is limited between 0 and 1, which can be interpreted as probabilities of DNNs. Therefore, the mathematical formula of Softmax can be expressed as:

$$f(xi) = \frac{e^{xi}}{\sum_{j=1}^{N} e^{xj}} \quad for\ i = 1,2,\dots,N \tag{22}$$

From (22), exponential unit implementation and divider unit implementation are crucial points of the approximating Softmax design.

Based on the layer-by-layer structure of the proposed design, the ReLU layer performs as a pre-layer of the Softmax layer. On the other hand, the curves of the exponential function and ReLU function have similar trends. Thus, the original exponential unit is replaced with the output of the ReLU layer, which makes the hardware implementation easy. Therefore, the formula can be simplified as follows:

$$f(x, y(i)) = \frac{y(i)=max(0,x)}{\sum_{j=1}^{N} y(j)=ma\ \ (0,x)} \quad for\ i = 1,2,\dots,N \tag{23}$$

The hardware architecture of approximating Softmax design is shown in Fig.12. It includes an adder and a divider unit. The hardware implementation method is supported by (23). The divider circuit is a multiple-input version of the process insensitive divider, presented by [25], operating under the subthreshold region to achieve the characteristic of low power consumption.

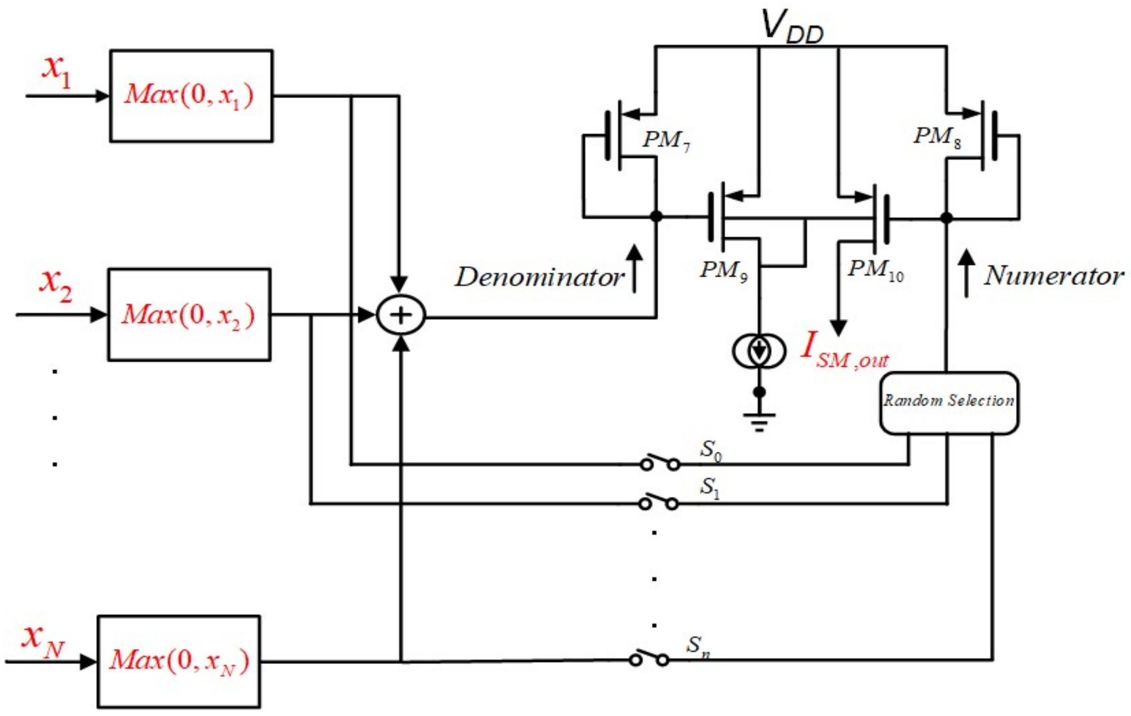


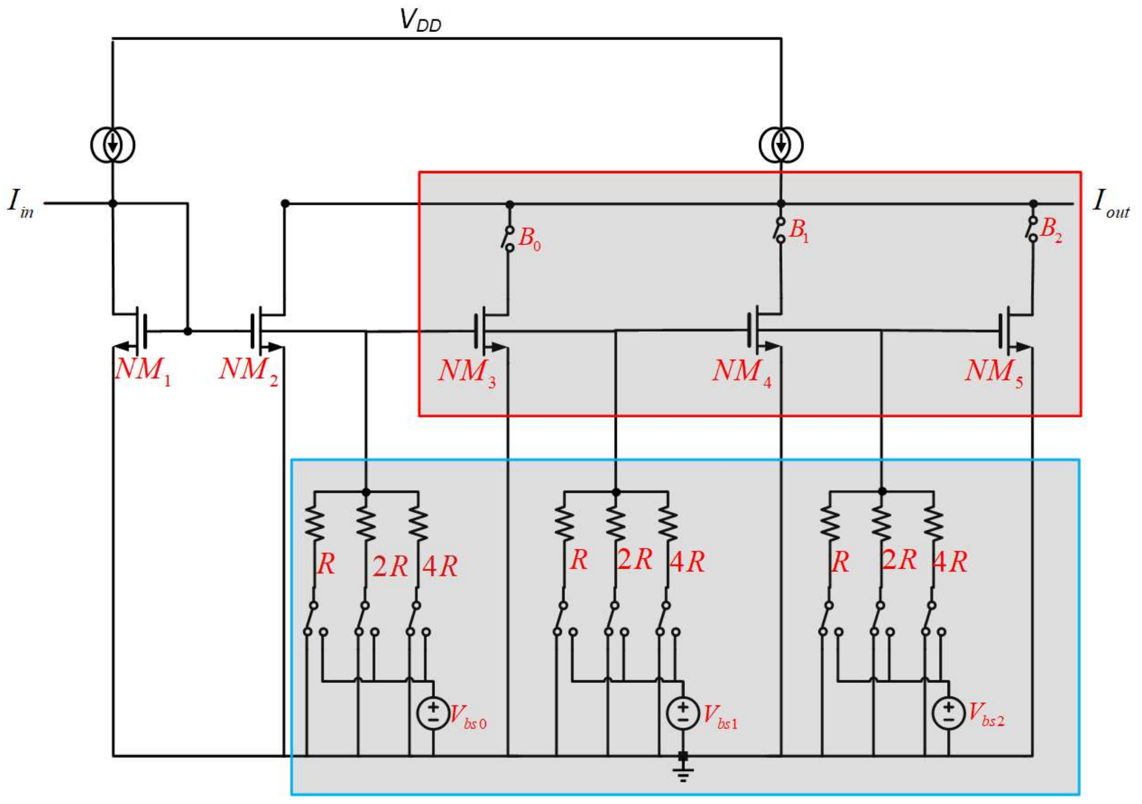
Figure 13. Architecture design of Softmax function

CHAPTER 5 CIRCUIT IMPLEMENTATION & RESULTS DISCUSSION

In Chapter 5, the discussion will concentrate on the circuit implementation and testing results. Also, this chapter will include a comparison with state-of-the-art.

5.1 Tunable Weight Multiplier Function

As discussed in the last chapters, the 6 bits tunable weight multiplier function circuit is shown in Figure 11. The overall architecture of the weight multiplier is shown in Figure 14. The appropriate transistor sizes of the CM block circuit and the adder circuits are shown in Table 2.



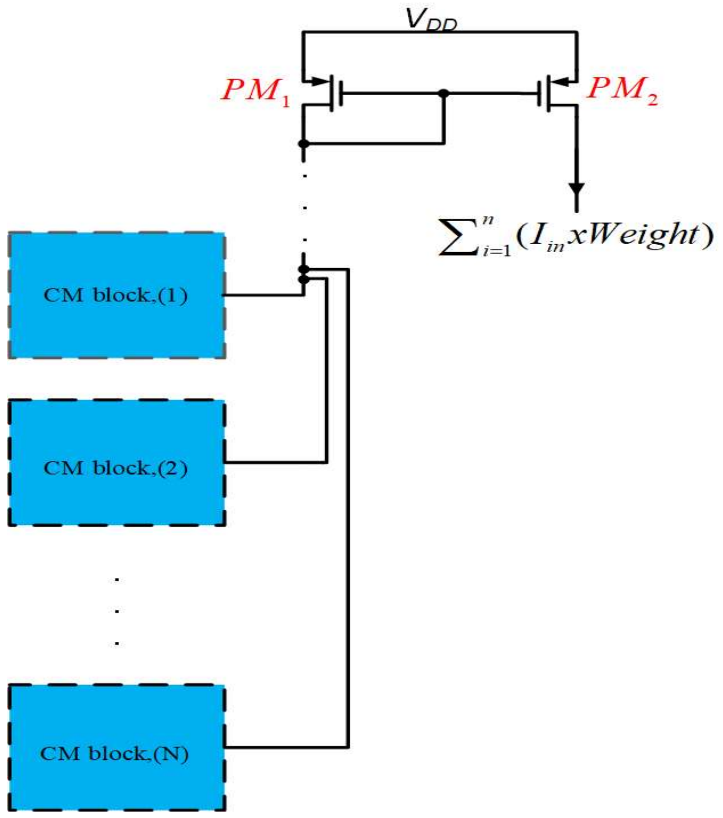


Figure 14. The overall architecture of the weight multiplier

Table 2. Transistor size of the weight multiplier function circuit

Transistor	W/L(μm)
NM1	2/1
NM2	2/1
NM3	2/1
NM4	2/1
NM5	2/1
PM1	1/1
PM2	1/1

The results of the tunable weight multiplier are shown in Figure 15, which presents the output current (in μA) as a function of the input current. Also, Figure 15 includes positive, negative weight factors and their relevant ideal case for both factors.

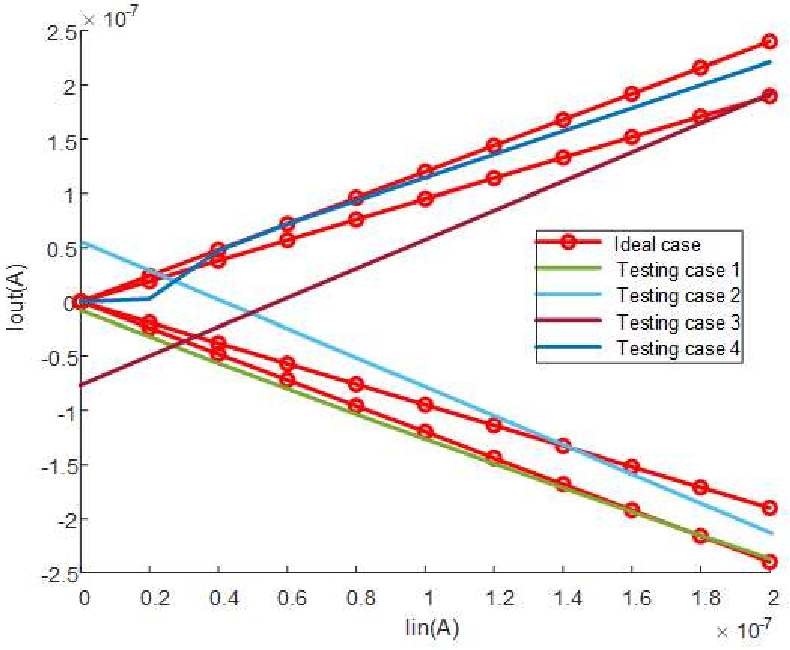


Figure 15. Results of weight multiplier

5.2 ReLU Activation Function

As discussed in the previous chapter, the ReLU function based on the PWL circuit is proposed in Figure 12. The I_{Bias} is set as 100uA. The transistors' size of the shifting block is discussed in Chapter 4. The transistor sizes of the ReLU functional circuit are presented in Table 3.

Table 3. Transistor size of the Softmax function circuit

Transistor	W/L(μm)
NM16	2/1
NM17	2/1
NM18	2/1
NM19	80/1
NM8	2/1
NM9	10/1
PM5	1/1
PM6	4.5/1

The output current results of the ReLU functional circuit are shown in Figure 16, which includes five corner analysis results (SS, SF, TT, FS, FF). As shown in the plot, if the input current is greater than 0, the output will perform as the input signal (output of the weight multiplier). Figure 17 shows the error function between the ideal case and the TT testing case.

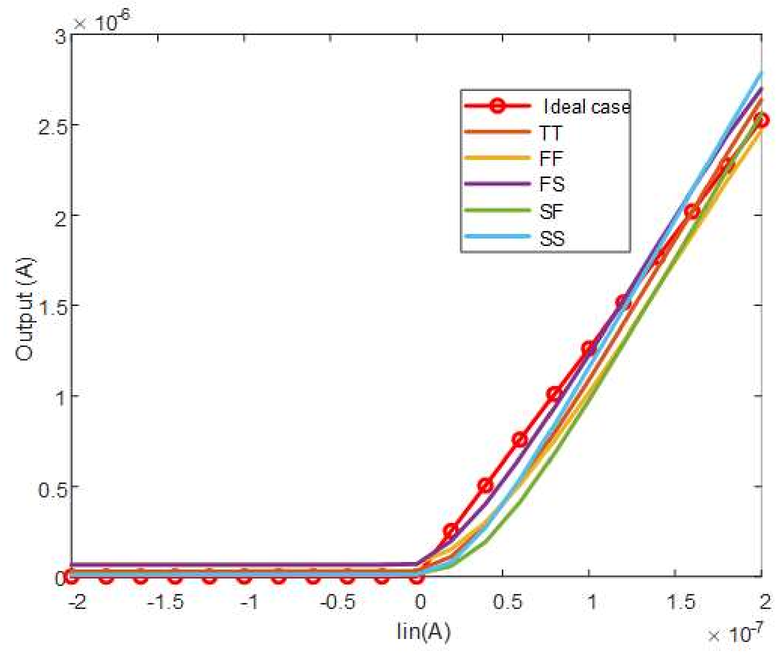


Figure 16. Corner analysis results and ideal case of ReLU function

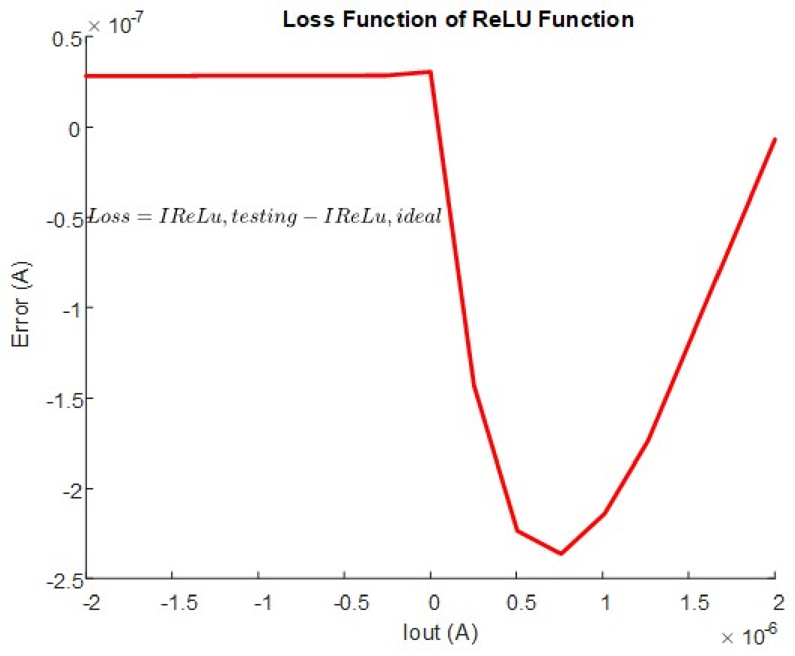


Figure 17. Error function of ReLU function

5.3 Approximate Softmax Activation Function

The proposed Softmax design is discussed in the previous chapter. The overall architecture includes the weight multiplier layer and ReLU layer, the output current from the weight multiplier layer. After rectified by the ReLU functional layer, the sum of the output current from the ReLU functional circuit will be sent to the divider circuit's denominator and control the numerator part of the Softmax function. The transistor sizes of the divider function of the Softmax functional circuit are presented in Table 4.

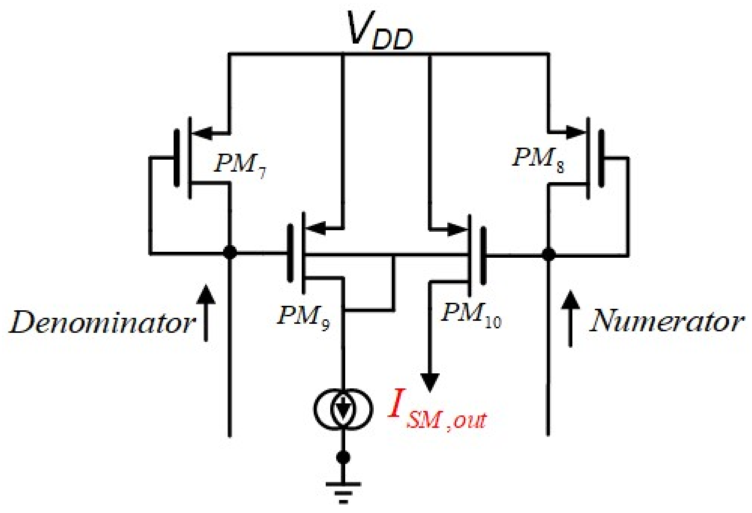


Figure 18. Divider function of ReLU function

Table 4. Transistor size of the divider function of Softmax function circuit

Transistor	W/L(μm)
PM7	28/1
PM8	28/1
PM9	10/1
PM10	20/1

The output current results of the Softmax functional circuit are shown in Figure 19, which includes five corner analysis results (SS, SF, TT, FS, FF). As shown in the plot, the output range is between -1 to 0, which can be seen as the probability. Figure 20 shows the error function between the ideal case and the TT case of the Softmax function.

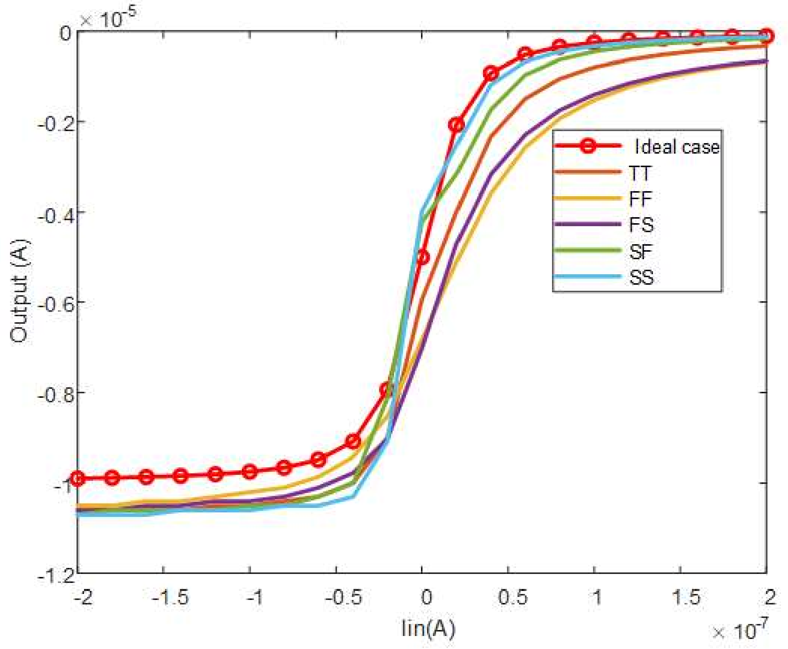


Figure 19. Corner analysis results and the ideal case of Softmax function

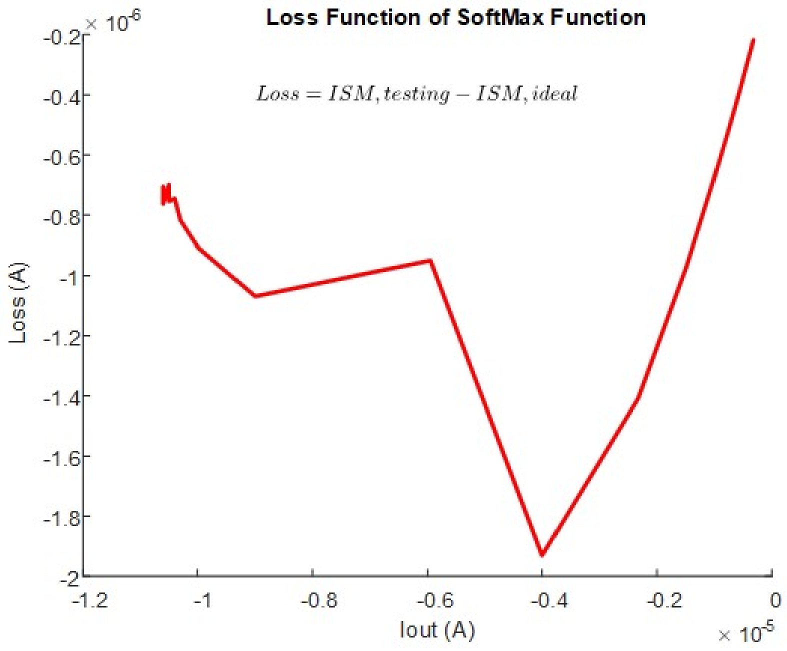


Figure 20. Error function of the Softmax function

5.4 Software structure

The testing accuracy is an essential measurement of the effectiveness of the neural network, and handwritten digit recognition is the primary testing application. Therefore, for the software architecture, a suitable algorithm can provide good performance for classification in digit recognition. [23] provides two powerful techniques to test the neural network performance: the ANN algorithm and CNN algorithm. Also, to explore the performance of digit recognition, a reliable dataset is an important stage during the testing process because of the dimensionality reduction technique. Therefore, the large dataset is suitable for this operation, which is the MNIST dataset.

The structure of the ANN algorithm is similar to the proposed DNN architecture, which includes three main layers: input layer, hidden layer, and output layer. For the software design, the loading data is implemented by the Numpy, which is the fundamental python library package. The size of the input image is equal to the number of neurons in the input layer. The number of classes in the MNIST dataset is 0 to 9, 10 classes, so the output layer includes ten neurons for 10 classes. Softmax functions calculate the output value of probabilities. Thus, the maximum value of each neuron can be easily selected.

CNN is a complex algorithm with a feature map filter for data sharing compared with the ANN algorithm. CNN also includes three layers the convolutional layer, the pooling layer, and the fully-connected layer. The convolutional layer can be seen as the preprocessing layer to apply numerous filters in a binary number. The next layer is the pooling layer, which is the feature extraction, similar to the ANN architecture's hidden layer. Because the input image has three dimensions, CNN can reduce dimensionality between layer to layer. Thus, the output from the fully-connected layer is the simple 1-D matrix which performs as the probabilities.

5.5 System level results

The data exported from the Cadence environment for weight multiplier, ReLU, and SoftMax functions have been evaluated using MATLAB. In addition, the ideal mathematical equations are included as references to demonstrate the error between ideal cases and testing cases of activation layers. These results show that the hardware implementation of activation functions performed exceptionally well. Therefore, for further improvement of the neural network's performance, the relevant error function can be used as the custom function to improve the performance analysis of the MNIST CNN. Table 5 shows the test accuracy of the different quantized activation functions.

Table 5. Test accuracy of different number of bits

Quantized activation layer	Test Accuracy
2 bits of activation layer	97.95%
4 bits of activation layer	98.63%
6 bits of activation layer	98.81%
8 bits of activation layer	99.02%

Figure 21 shows the performance results of five corner analyses (SS, SF, TT, FS, FF). The design results perform stability during the process variation. The FS corner is selected for this design, corresponding to 99.02% accuracy and 68.21uW power consumption. The presented function layer can be seen as the baseline circuit of the proposed design. [17] presented analog designs of a complete system with multiplier and ReLU functions. The proposed design has a considerable power consumption advantage compared with the published multiplier layer in [17] that consumes 496uW compared to 21.03uW in our multiplier layer design. [4] provided an activation function with a similar shape as the proposed design. In the same dataset, power consumed by the activation function is 1.4mW with a test accuracy of 90%. The activation function layer of the proposed design only consumes a power of 25.05uW. For the system performance, Table 6 shows the performance of different neural network analog designs. Compared with the data reported in [4] and [28], the proposed design has a better power consumption performance, about 35.65% lower. Meanwhile, the proposed design achieves a high programming accuracy in MNIST. Furthermore, the data reported in [26] and [27] achieves a high testing accuracy (99%) similar to the proposed design; however, state-of-the-art sacrificed much more power than the proposed design. Compared with the data reported in [30], the digital design of the neural network, the proposed design has a pretty low power consumption. Hence, the proposed design performs the excellent potential of artificial application with an optimum effective measurement (99.02%) and pretty low power consumption (68.21uW).

Table 6. Performance comparison with State-of-the-art

Parameters	[4]	[26][27]	[28]	[30]	Proposed
Tech(nm)	65	65	180	65	180
Supply Voltage(V)	0.4	1.2	1.8	1.2	1.5
Circuit Type	Analog	Analog	Analog	Digital	Analog
Power Consumption	100uW	380uW	106uW	45.1mW	68.21uW
Test Accuracy	82%	99%	94.6%	98.3%	99.02%
Dataset	MNIST	MNIST	MNIST	MNIST	MNIST

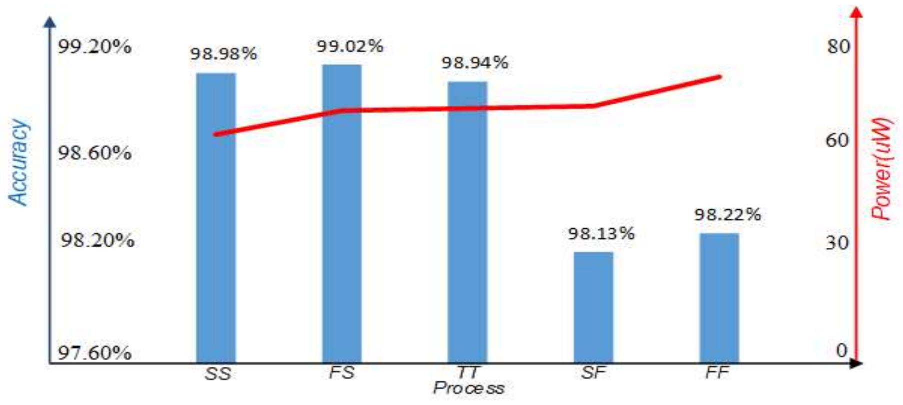


Figure 21. Test accuracy and power consumption of different corner

CHAPTER 6 FUTURE WORK & CONCLUSION

6.1 Future work

This research achieved the basic requirements of each functional layer in the neural network. To improve the proposed design performance, there are the following recommendations for future research.

1. Design different weight multiplier circuits, ReLU functional circuits, and Softmax functional circuits to improve the accuracy and power consumption between testing and ideal cases in future work.

2. Implement the proposed design in the PCB and compare it with the state-of-the-art to observe the design performance at the system level.

3. Test the proposed design in different datasets to perform multi-task learning in future work.

6.2 Conclusion

In this thesis, analog current-mode circuit implementation for various DNN functions has been proposed. Also, this thesis introduced the background information and the architecture of the DNN functions.

The proposed circuit design includes building blocks for binary weight multiplication, PWL ReLU function, and approximate Softmax function. At the system level, the MNIST dataset was used to evaluate the accuracy of the proposed design. As a result, the proposed design achieved an accuracy of 99.02% with a power consumption of 68.21uW.

The design circuit is simulated in TSMC 0.18um CMOS technology under 1.5V supply voltage. Compared with the existing application, the testing results have a great potential for artificial application.

BIBLIOGRAPHY

[1] Mishra, Manish, and Monika Srivastava. "A view of artificial neural network." *2014 International Conference on Advances in Engineering & Technology Research (ICAETR-2014)*. IEEE, 2014.

[2] Chen, Mingzhe, et al. "Artificial neural networks-based machine learning for wireless networks: A tutorial." *IEEE Communications Surveys & Tutorials* 21.4 (2019): 3039-3071.

[3] LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." *nature* 521.7553 (2015): 436-444.

[4] Jayaraj, Akshay, Imon Banerjee, and Arindam Sanyal. "Common-source amplifier based analog artificial neural network classifier." *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2019.

[5] Oh, Sechang, et al. "An acoustic signal processing chip with 142-nW voice activity detection using mixer-based sequential frequency scanning and neural network classification." *IEEE Journal of Solid-State Circuits* 54.11 (2019): 3005-3016.

[6] Rumberg, Brandon, et al. "Hibernets: Energy-efficient sensor networks using analog signal processing." *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 1.3 (2011): 321-334.

[7] Kouretas, Ioannis, and Vassilis Paliouras. "Simplified hardware implementation of the softmax activation function." *2019 8th international conference on modern circuits and systems technologies (MOCAST)*. IEEE, 2019.

[8] Mahanta, Jahnavi. "Introduction to neural networks, advantages and applications." *Towards Data Science* 13 (2017).

[9] Templeton, Graham. "Artificial neural networks are changing the world." *What are they* (2018).

[10] Ide, Hidenori, and Takio Kurita. "Improvement of learning for CNN with ReLU activation by sparse regularization." *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2017.

[11] Sawigun, C., and W. A. Serdijn. "Ultra-low-power, class-AB, CMOS four-quadrant current multiplier." *Electronics Letters* 45.10 (2009): 483-484.

[12] Tao, Xiaobing, Chao Liu, and Tao Zhao. "A four-quadrant analog multiplier under a single power supply voltage." *Analog Integrated Circuits and Signal Processing* 71.3 (2012): 525-530.

[13] Lee, Kyuho, Junyoung Park, and Hoi-Jun Yoo. "A low-power, mixed-mode neural network classifier for robust scene classification." *Journal of Semiconductor Technology and Science* 19.1 (2019): 129-136.

[14] Al-Tamimi, Karama M., and EL-Sanakary Kamal. "Body-driven log/antilog PVT compensated analog computational block." *Analog Integrated Circuits and Signal Processing* 90.3 (2017): 693-700.

[15] Mohamed, Ahmed Reda, Liang Qi, and Guoxing Wang. "A power-efficient and re-configurable analog artificial neural network classifier." *Microelectronics Journal* 111 (2021): 105022.

[16] Liu, Weihsing, Shen-Iuan Liu, and Shui-Ken Wei. "CMOS current-mode divider and its applications." *IEEE Transactions on Circuits and Systems II: Express Briefs* 52.3 (2005): 145-148.

[17] Huang, Yucong, et al. "Analog Circuit Implementation of Neurons with Multiply-Accumulate and ReLU Functions." *Proceedings of the 2020 on Great Lakes Symposium on VLSI*. 2020.

[18] Zhou, Yuteng, Shrutika Redkar, and Xinming Huang. "Deep learning binary neural network on an FPGA." *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*. IEEE, 2017.

[19] Geng, Chao, Qingji Sun, and Shigetoshi Nakatake. "An analog CMOS implementation for multi-layer perceptron with relu activation." *2020 9th International conference on modern circuits and systems technologies (MOCAST)*. IEEE, 2020.

[20] Yuan, Bo. "Efficient hardware architecture of softmax layer in deep neural network." *2016 29th IEEE International System-on-Chip Conference (SOCC)*. IEEE, 2016.

[21] Wang, Meiqi, et al. "A high-speed and low-complexity architecture for softmax function in deep learning." *2018 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS)*. IEEE, 2018.

[22] Zunino, Rodolfo, and Paolo Gastaldo. "Analog implementation of the softmax function." *2002 IEEE International Symposium on Circuits and Systems. Proceedings (Cat. No. 02CH37353)*. Vol. 2. IEEE, 2002.

[23] Beohar, Drishti, and Akhtar Rasool. "Handwritten Digit Recognition of MNIST dataset using Deep Learning state-of-the-art Artificial Neural Network (ANN) and Convolutional Neural Network (CNN)." *2021 International Conference on Emerging Smart Computing and Informatics (ESCI)*. IEEE, 2021.

[24] Bhat, M. S., S. Rekha, and H. S. Jamadagni. "Extrinsic analog synthesis using piecewise linear current-mode circuits." *19th International Conference on VLSI Design held jointly with 5th International Conference on Embedded Systems Design (VLSID'06)*. IEEE, 2006.

[25] Al-Absi, Munir A. "Low-voltage and low-power CMOS current-mode divider and 1/x circuit." *2010 International Conference on Electronic Devices, Systems and Applications*. IEEE, 2010.

[26] Sayal, Aseem, et al. "A 12.08-TOPS/W all-digital time-domain CNN engine using bi-directional memory delay lines for energy efficient edge computing." *IEEE Journal of Solid-State Circuits* 55.1 (2019): 60-75.

[27] Biswas, Avishek, and Anantha P. Chandrakasan. "Conv-RAM: An energy-efficient SRAM with embedded convolution computation for low-power CNN-based machine learning applications." *2018 IEEE International Solid-State Circuits Conference-(ISSCC)*. IEEE, 2018.

[28] Asghar, Malik Summair, Saad Arslan, and Hyungwon Kim. "A Low-Power Spiking Neural Network Chip Based on a Compact LIF Neuron and Binary Exponential Charge Injector Synapse Circuits." *Sensors* 21.13 (2021): 4462.

[29] Shihao Wang, Karama M. Al-Tamimi, Issam Hammad, and Kamal Elsankary. “Towards Current-Mode Analog Implementation of Deep Neural Network Functions.” (under review)

[30] Sim, Jaehyeong, et al. "14.6 a 1.42 tops/w deep convolutional neural network recognition processor for intelligent ioe systems." *2016 IEEE International Solid-State Circuits Conference (ISSCC)*. IEEE, 2016.