

國立交通大學

電子工程學系電子研究所

博士論文

大鄰近層細胞非線性網路與比例式記憶細胞非線性

網路之設計與分析



THE DESIGN AND ANALYSIS OF
LARGE-NEIGHBORHOOD CELLULAR
NONLINEAR NETWORKS AND RATIO-MEMORY
CELLULAR NONLINEAR NETWORKS

研究生：陳勝豪

指導教授：吳重雨 博士

中華民國九十八年六月

大鄰近層細胞非線性網路與比例式記憶細胞
非線性網路之設計與分析

THE DESIGN AND ANALYSIS OF
LARGE-NEIGHBORHOOD CELLULAR
NONLINEAR NETWORK AND RATIO-MEMORY
CELLULAR NONLINEAR NETWORK

研究生：陳勝豪 Student: Sheng-Hao Chen

指導教授：吳重雨 博士 Advisor: Dr. Chung-Yu Wu

國立交通大學

電子工程系電子研究所

博士論文

A Dissertation

Submitted to

Department of Electronics Engineering and Institute of Electronics

College of Electrical and Computer Engineering

National Chiao-Tung University

In Partial Fulfillment of the Requirements

For the Degree of

Doctor of Philosophy

In

Electronics Engineering

June 2009

Hsinchu, Taiwan, Republic of China

中華民國九十八年六月

大鄰近層細胞非線性網路與比例式記憶細胞非線性網路之設計與分析

研究生：陳勝豪

指導教授：吳重雨 博士

國立交通大學電子工程系電子研究所

摘 要

此論文研究針對於類神經網路(細胞非線性網路)的研究與應用，細胞非線性網路模仿神經聯結方式運算，可視為一類比式計算機處理單元陣列，適合運用在影像處理，雖然目前數位式計算機處理單元可以達到數個GHz的處理速度，但在影像處理方面，若以各個圖元分別作運算，仍需要大量的處理時間，因此若以細胞非線性網路陣列平行運算，可達到高速運算的結果，並針對神經網路之特性與其優缺點，以類比電路實現為主軸，分別實現以下兩個部分：

1. 設計分析一可程式化之大鄰近層細胞非線性網路通用機器核心部分。
2. 設計分析一可學習之免衰減比例式記憶細胞非線性網路與一反覆學習比例式記憶細胞非線性網路。

目前細胞非線性網路通用機器僅能處理 3×3 的範本，即僅有鄰接的各圖元間有係數的關聯，而大鄰近層細胞非線性網路的主要構想，在於若可將關聯推性廣至更遠之細胞上，可增加細胞非線性網路的功能性；此外，亦有其他團隊針對將大鄰近層細胞非線性網路的範本，分解成數個 3×3 的範本來達到相同的功能，因此若能設計一大鄰近層細胞非線性網路，一步完成大鄰近層細胞非線性網路的功能，可節省所需之處理時間與消耗功率；因為大鄰近層細胞非線性網路為一大型陣列，電路設計方面主要考慮其功率消耗與面積大小，並以傳導式連結的電路架構，使其可實現大鄰近層細胞非線性網路的功能，論文中許多大鄰近層細胞非線性網

路的範本，皆可在模擬中實現，而所設計之大鄰近層細胞非線性網路陣列大小為 20×20 ，晶片大小為 $1543 \mu\text{m} \times 1248 \mu\text{m}$ ，功率消耗在待機時僅 0.7 mW ，一般操作下為 18 mW ，操作頻率為 20 MHz ，並在實現中驗證可實現人的錯覺範本。

可學習之比例式記憶細胞非線性網路目的在於學習各種樣本，並將含有雜訊的樣本復原，原理是將兩個圖元間的關係，紀錄在比例式記憶體的電容中，並利用其漏電的缺點強化圖元間的關係，並將各個圖元周圍的範本常態化(normalized)，因此稱之為比例式記憶，藉此可提高其辨識率；然而，由於各個圖元間的差異，若以相同的放電時間強化圖元間關係，可能會造成此關係被破壞或是強化不足，因此各個關係改以與圖元周圍的關係平均來決定其值的去留，以此方式可節省除法器之運用並簡化比例式記憶細胞非線性網路的複雜度。另外，從機率統計方面亦可推論出臨界值範本的必要性，即為其臨界值範本(Threshold)，由此提出以遞迴學習的方式，統計出雜訊與辨識後的臨界值，藉此可更加增加其辨識率。

本論文之主要貢獻為，建立一完整大鄰近層細胞非線性網路之架構，並以簡單之電路實現，因此可達到小面積、低功率，經實驗量測可用於二元(binary)的影像運算；另外不需放電之可學習之比例式記憶細胞非線性網路方式，亦簡化了電路的複雜度，使其容易實現。亦討論了可學習比例式記憶細胞非線性網路之機率統計模型，並依據推論結果，運用臨界值範本的學習，更增進其辨識率。

THE DESIGN AND ANALYSIS OF LARGE- NEIGHBORHOOD CELLULAR NONLINEAR NETWORK AND RATIO- MEMORY CELLULAR NON-LINEAR NETWORK

Student: Sheng-Hao Chen Advisor: Dr. Chung-Yu Wu

Institute of Electronics Engineering
National Chiao-Tung University

Abstract

This dissertation focuses on the studies and applications of the cellular neural/nonlinear networks (CNN). CNN is an analog CPU array which can imitate the operations of neural connections which is suitable for image processing. Although the speed of the recent digital CPUs can reach higher than several GHz, when the digital CPU is applied on the image processing, it takes a lot of time to achieve the processing separately. Hence, the advantage of parallel processing of CNN array is required to achieve high speed processing. According to the properties of CNN, two major topics are realized by using analog circuit design.

- I. The design and analysis of a CMOS low-power, large-neighborhood CNN with propagating connections
- II. The design and analysis of a ratio memory CNN

Recently, cellular nonlinear network universal machine (CNUM) can only achieve the 3×3 templates of nearest connecting correlations. The main concept of large-neighborhood cellular nonlinear network (LNCNN) is to extend the connecting correlations and to increase the capability of CNN. Moreover, some studies have

decomposed the LNCNN templates into several 3×3 templates to realize the same functions. However, this may take more cost to achieve one LNCNN function. Hence, it is necessary to design a LNCNN for the templates larger than 3×3 .

Because LNCNN is a very large scale array, the power consumption and chip area are considered first. With the propagating connections, the functions of LNCNN are realized by the designed 20×20 LNCNN array and the chip size is $1543 \mu\text{m} \times 1248 \mu\text{m}$. The power consumption is 0.7 mW on standby and 18 mW in operation with a system clock frequency of 20 MHz.

The purpose of the learnable ratio memory cellular nonlinear networks is to learn the every kind of patterns and recover the learned noisy patterns. The concept is to store the correlations of two neighboring cells on the capacitor in the ratio memories and use the intrinsic leakage to enhance the common characteristics. Moreover, the templates are normalized by the correlation with neighboring cells to increase the recognition rate and thus, it is called ratio memory. However, due to the difference of any two cells, if the same elapsed time for leakage is applied to enhance the characteristics, it may cause only the self-feedback term to remain or the enhancement of common characteristics to be smaller. Hence, the templates are decided by the correlation and the mean of the four correlations around one cell. This can make the design much easier and the divider can be abandoned. Besides, by the deviation of the statistics and probability, there exists a dc term except for the templates. It is found that the threshold template is required and learned by recursive learning to gather the information of the noisy patterns to increase the recognition rate.

The main contribution of this dissertation is that the complete architecture of large-neighborhood CNN has been established and realized by a simple circuit design. Hence, a small-size, low-power LNCNN chip has been fabricated and measured. According to the experimental result, the LNCNN chip can be applied on the binary image processing. Moreover, the statistic and probability models of the learnable ratio memory CNN has also been derived and, according to the results, the learning of the threshold templates are used to increase the recognition rate. Furthermore, the learnable

ratio memory CNN without elapsed time has also been proposed to simplify the complexity of the circuits for realization.



致 謝

首先感謝我的指導教授吳重雨教授，學術上循序漸進地細心指導，適時給予建議，使我能順利的完成學業，而教授在生活上也時時刻刻關心，了解學生的想法，因此不僅在研究方面，學習到老師嚴謹的態度，亦於待人處世上獲益良多；另外也要感謝和藹可親的師母曾昭玲女士，謝謝她常常給予我關懷與打氣加油。

感謝實驗室的學長姐、同學以及學弟妹的協助，使我在這六年來的博士班生涯更加的豐富，也使得我的研究更加順遂。感謝鄭秋宏、黃冠勳、廖以義、施育全、賴瑞麟、周忠昫、林俐如、虞繼堯、王文傑及蘇烜毅學長姐的經驗與教導，不論是知識的啟發或是處事的方法上，皆幫助了我許多；也感謝楊文嘉、黃祖德、陳煒明、雄霆等同期的同學，一同討論、聊天、發洩，共同做研究、傾洩攻讀博士班的壓力；另外也感謝學弟妹們，幫忙處理事務以及計畫研究；感謝實驗室助理卓慧貞小姐(蛤~~~)，以及中心助理何淳伶及小小黃瑋屏，在行政事務上的協助；還有好朋友小邱跟小邱姐姐的鼓勵，常常一起去大潤發瞎拼發洩，我會好好照顧妳們送的漂亮包包；有了學長姐、同學、學弟妹以及好朋友的幫助鼓勵下，我的論文才得以順利完成，因此在此也祝福諸位學業上能順利畢業，事業上能一帆風順。

最後我要致上最深的的感謝，給我的父親陳振源先生與我的母親葉柳青女士，由於您們的從小的教養以及支持鼓勵，讓我得以最大的力量，來完成博士班的學業，感謝您們無怨無悔的付出，並且在我心情不好時承受了我的不滿，在此對您們表示深深的歉意，我愛您們，我的爸媽。

陳勝豪
誌於 風城交大
九十八年夏

CONTENTS

ABSTRATE (CHINESE)	i
ABSTRATE (ENGLISH).....	iii
ACKNOWLEDGEMENTS.....	vi
CONTENTS	vii
TABLE CAPTIONS.....	ix
FIGURE CAPTIONS	x
CHAPTER 1 INTRODUCTION.....	1
1.1 BACKGROUND OF ARTIFICIAL NONLINEAR NETWORKS	1
1.2 RESEARCHES ON CNNs AND THEIR APPLICATIONS.....	11
1.3 REVIEW OF LNCNNs AND RMCNNs	14
1.4 RESEARCH MOTIVATION AND ORGANIZATION OF THIS DISSERTATION	17
CHAPTER 2 THE DESIGN AND ANALYSIS OF A CMOS LARGE-NEIGHBORHOOD CNN WITH PROPAGATING CONNECTIONS	21
2.1 INTRODUCTIONS	21
2.2 ARCHITECTURE AND MODELS	23
2.3 CIRCUIT IMPLEMENTATION AND SIMULATION RESULTS	30
2.4 EXPERIMENTAL RESULTS.....	49
2.5 SUMMARY	54
CHAPTER 3 THE DESIGN AND ANALYSIS OF A CMOS RATIO-MEMORY CNN WITHOUT ELAPSED TIME	56
3.1 INTRODUCTION	56
3.2 ARCHITECTURE AND MODELS	58
3.3 CIRCUIT IMPLEMENTATION WITH SIMULATION RESULTS	65
3.4 EXPERIMENTAL RESULTS.....	79
3.5 SUMMARY	86

CHAPTER 4	THE ANALYSIS OF THE RECURSIVE LEARNING RMCNN.....	87
4.1	INTRODUCTION	87
4.2	MATHEMATICAL	88
4.3	SIMULATION RESULTS.....	93
4.4	SUMMARY AND FUTURE WORK	95
CHAPTER 5	CONCLUSIONS AND FUTURE WORK.....	98
5.1	CONCLUSIONS.....	98
5.2	FUTURE WORK.....	100
REFERENCES	102



TABLE CAPTIONS

Table 2.1	DERIVED EQUATIONS OF TEMPLATE COEFFICIENTS AND GAINS OF SYNAPSES	29
Table 2.2	COMPARISON OF DEVICE NUMBERS AND INTERCONNECTION LINES	40
Table 2.3	COMPARISON OF LNCNN WITH CNNUC3 [132]-[133] AND ACE16K [131].....	53
Table 2.4	COMPARISON OF THE PROPOSED LNCNN AND LNCNN WITH SYMMETRIC TEMPLATES.....	54
Table 3.1	THE COMPARISONS OF TEMPLATES A IN CELL(4, 5), CELL(5, 3), CELL(8, 5), AND CELL(7, 5) BETWEEN RMCNN WITH AND WITHOUT ELAPSED TIME.....	74
Table 3.2	THE DESCRIPTION OF EACH CONTROL SIGNAL	80
Table 3.3	THE COMPARISON OF THE ABSOLUTE WEIGHTS A44 WITH MATLAB AND HSPICE IN DIFFERENT CONDITIONS.....	84
Table 3.4	COMPARISON BETWEEN RMCNN AND RMCNN REQUIRING NO ELAPSED TIME	85
Table 4.1	THE REQUIRED ITERATIONS TO FIT THE CONSTANS WHERE 7 PATTERNS ARE LEARNED	95
Table 4.2	THE REQUIRED ITERATIONS TO FIT THE CONSTANS $\delta = 0.03$ WHERE 6 AND 8 PATTERNS ARE LEARNED.....	96

FIGURE CAPTIONS

Fig. 1.1	The simplest computational element or node which forms a weighted sum of N inputs and passes the result through the nonlinearity.	2
Fig. 1.2	The three common types of nonlinearity of (a) hard limiters, (b) threshold logic elements, and (c) sigmoidal nonlinearities.	2
Fig. 1.3	The biological model of a neuron cell which contains the cell body (nucleus) and the I/O terminals of dendrites and axon terminals.	4
Fig. 1.4	The simplest architecture of an adaptive linear element where its weights are determined by the normalized least mean square training law by a preset desired output.	6
Fig. 1.5	A perceptron with a sigmoidal activation function. The threshold value w_0 are initialized to small non-zero values.	7
Fig. 1.6	A simple three-layer network which contains input, hidden, and output layers.	8
Fig. 1.7	Two-dimensional array of Kohonen's self-organizing feature maps.	9
Fig. 1.8	The RC circuit model of a CNN cell.	10
Fig. 1.9	The core architecture of CNN with templates A and B	11
Fig. 2.1	The architecture of a LNCNN kernel unit.	24
Fig. 2.2	The structure of the BODY in Fig. 2.1.	25
Fig. 2.3	The large-neighborhood template generated by a LNCNN with propagating connections.	28
Fig. 2.4	The circuit diagram of the Neuron and PZ in Fig. 2.2.	31
Fig. 2.5	The transfer characteristic of a neuron with different external bias currents I_{bias}	32
Fig. 2.6	The circuit diagrams of (a) the synapses PL2, PR2, PD2, and PU2; (b) the synapses PL1, PR1, PD1, and PU1; (c) the synapses PRU, PRD, PLU, PLD, and PS; (d) the Sign Controller.	35

Fig. 2.7	The HSPICE simulated Inouta vs. Inina diagram of the N-type synapse in Fig. 2.6(a) with 16 different values for Vbiasn.	36
Fig. 2.8	The range of (a) the N-type current gains and (b) the P-type current gain of the synapses with an input current range from 300 nA to 500 nA.....	37
Fig. 2.9	The circuit diagram of the PSW.	39
Fig. 2.10	The circuit diagram of the analog memory.	41
Fig. 2.11	The architecture of the 20x20 LNCNN system.	42
Fig. 2.12	The timing diagram of the controlled signals in LNCNN.....	43
Fig. 2.13	The 5 x 5 templates B, A and Z for Muller-Lyer illusion [40].....	44
Fig. 2.14	The extracted values of the diamond-shaped template from a HSPICE post-layout simulation.	45
Fig. 2.15	(a) The input patterns of Muller-Lyer illusion. (b) The resultant output pattern of Muller-Lyer illusion from the HSPICE simulation result.....	45
Fig. 2.16	The template B of 5x5 and diamond-shaped templates of (a) diffusion [138] and (b) de-blurring [40].	47
Fig. 2.17	The input patterns and simulation results of (a) diffusion and (b) de-blurring.	48
Fig. 2.18	The input and output patterns of erosion with 3 x 3 neighborhood templates [40] in two iterations and with diamond-shaped templates in one iteration.	49
Fig. 2.19	A photograph of the fabricated 20 x 20 LNCNN chip.....	50
Fig. 2.20	The experimental resultant output pattern of Muller-Lyer illusion.....	51
Fig. 2.21	The experimental results of Pixel B with the signal <i>Operation_Start</i>	52
Fig. 3.1	The general architecture of the RMCNN.	62
Fig. 3.2	(a) The input stage and neuron, (b) RM, and (c) comparator and counter in the kernel unit of RMCNN.....	63
Fig. 3.3	(a) The circuits of the blocks VTI1 and Neuron. (b) The transfer characteristic of the block VTI1.	66
Fig. 3.4	(a) The circuit of the blocks VTI2 (with ME) and VTI3 (without ME) (b) The transfer characteristic of the block VTI2 and VTI3.	68

Fig. 3.5	The circuit of the block W.....	69
Fig. 3.6	The block diagram of the sign controller where the detector is composed of two cascaded inverters.	69
Fig. 3.7	The circuit of the block COMP.	69
Fig. 3.8	The four absolute currents from VTI3 are averaged and compared with the mean current.	71
Fig. 3.9	Input patterns in the learning period.....	72
Fig. 3.11	The recognition rates by using proposed RMCNN and by being directly amplified.....	73
Fig. 3.12	The comparison of the recognition rates by using proposed RMCNN with self-feedback, without self-feedback of 50% tolerance and by being directly amplified.....	74
Fig. 3.13	The comparison of recognition rates with 3×3 neighborhood templates and large neighborhood diamond templates of $r^2=3$	75
Fig. 3.14	The ratio weights of RMCNN with different elapsed time.	76
Fig. 3.15	The recognition rates of (a) 3×3 neighborhood and large neighborhood templates by repeating the operation of the proposed algorithm (marked with 'modified') where 7 patterns are learned.	78
Fig. 3.16	The modified circuits of block COMP that can realize the repeated proposed algorithm.....	78
Fig. 3.17	The architecture of a 9×9 RMCNN without elapsed time chip.	79
Fig. 3.18	The timing diagram of control signals.....	81
Fig. 3.19	The photograph of the RMCNN without elapsed time chip.....	81
Fig. 3.20	The uniform noisy patterns for measurement.....	82
Fig. 3.21	Experimental results of recognized patterns in the recognition period after a set of patterns with noise level 0.25 are recognized.....	82
Fig. 3.22	Experimental output waveform of the third recognized pattern.....	83
Fig. 3.23	The modified circuit of the block W.	85
Fig. 4.1	The probability of the input uC1.	89

Fig. 4.2	The probability density of the input u_{C2}	90
Fig. 4.3	The probability density of the state x_{C1} after recognition.....	90
Fig. 4.4	The error rates produced by the shadow part when the output of the pixel C1 should be (a) 1 and (b) -1.....	92
Fig. 4.5	The procedure of the recursive learning algorithm.	92
Fig. 4.6	The recursive learning of $THR(i,j)$ in n th iteration.....	92
Fig. 4.7	The recognition rates of RMCNN requiring no elapsed time without and with recursive learning of constrains 0.01, 0.03, and 0.05 where 7 patterns are learned.....	94
Fig. 4.9	The recognition rates where 6 patterns and 8 patterns are learned with and without recursive learning.	96



CHAPTER 1

INTRODUCTION

1.1 BACKGROUND OF ARTIFICIAL NONLINEAR NETWORKS

Brain, one of the world's best computers, makes human devoted to investigating it to expose the source of powerful functions. With the analog neuron models, the artificial neural networks (ANNs) proposed by Hopfield [1]-[5] and Chua *et al.* [6]-[8] have firstly been implemented in circuitry [10]. Since then, ANNs have attracted strong interest of researchers to explore their scientific and engineering applications. The models of ANNs [9], [5], [11] which are based on the understanding of biological nervous systems, attempt to achieve good performance by the dense interconnection of simple computational elements. Computational elements or nodes are connected via weights that are typically adapted during the operation such as Hopfield net [1]-[10], [11], Hamming net [11]-[15], *et al.* The simplest node sums N weighted inputs and passes the result through the nonlinear function $f(\bullet)$ as shown in Fig. 1.1 [11], [16]. In Fig. 1.1, the output y can be illustrated as

$$y = f\left(\sum_{i=0}^{N-1} w_i x_i - \theta\right) \tag{1.1}.$$

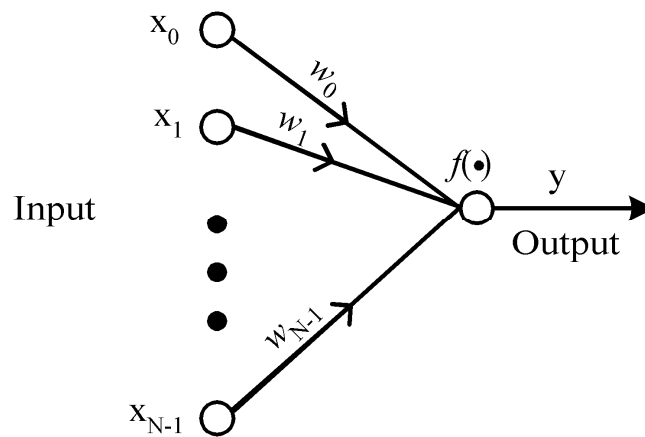


Fig. 1.1 The simplest computational element or node which forms a weighted sum of N inputs and passes the result through the nonlinearity.

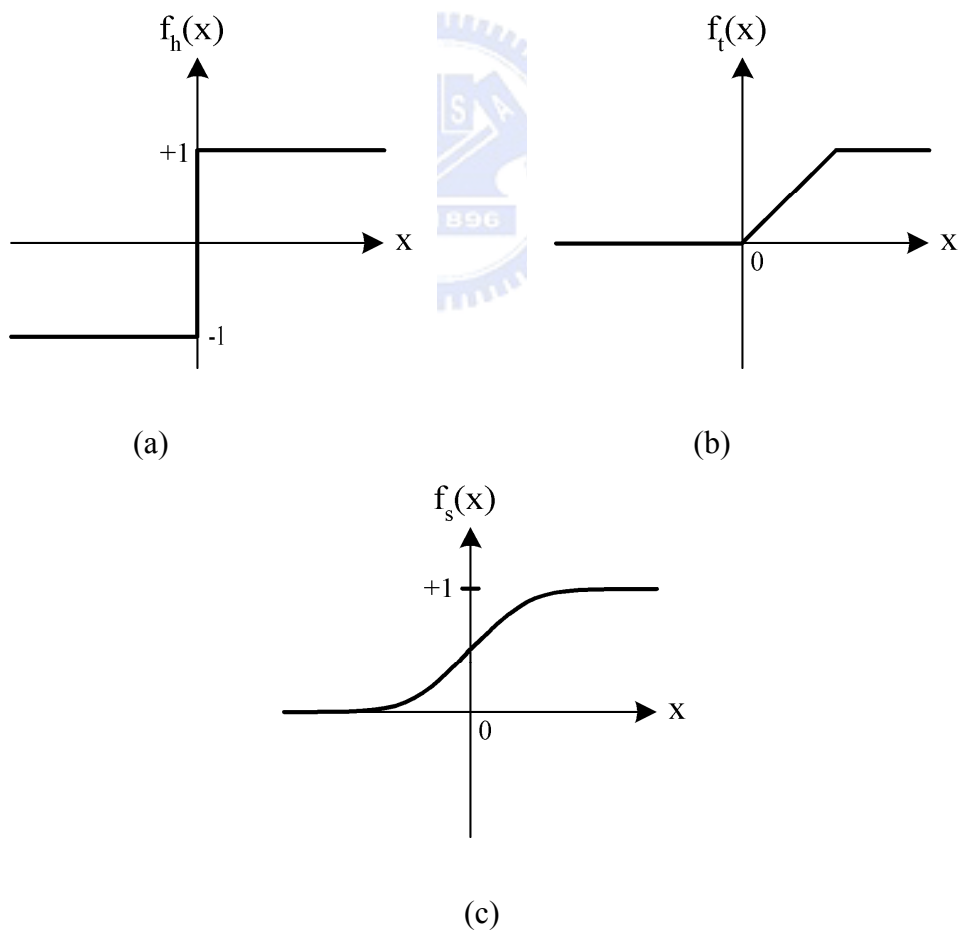


Fig. 1.2 The three common types of nonlinearity of (a) hard limiters, (b) threshold logic elements, and (c) sigmoidal nonlinearities.

where x_i is the i^{th} input, w_i is the i^{th} weight factor, and θ is the internal threshold. The node is characterized by an internal threshold or offset θ and by the type of the nonlinearities. Fig. 1.2(a)-(c) illustrate three common types of nonlinearity: hard limiters (threshold functions), piecewise linear functions, and sigmoidal nonlinearities. The common characteristic of these three nonlinearities is that the output y is saturated at both ends. More complex nodes may include temporal integration or other types of time dependencies and more complex mathematical operations than summation.

For comparison, silicon devices have an intrinsic speed about 100,000 times faster than that of natural neurobiological devices. However, in solving problems like face recognition, the neurobiological system is more effective by a factor of 10^8 [17]. In the biological model of a neuron cell as shown in Fig. 1.3, the neuron contains cell body (nucleus) and the synapses, which are the I/O terminals of the neuron and can be classified as dendrites and axon terminals by their essential functions, are illustrated. Dendrites can receive excitation or inhibition signals from other neurons or external environment. Axon terminals can pass the excitation or inhibition signals to next neurons. Through different functions, different intensities of the excitation or inhibition signals can be transferred to next neurons. The second neuron next to the first one receives the signals from the axon terminals of the first neuron and other neurons, makes a decision by the sigmoidal nonlinearity, and sends another excitation or inhibition signals to next neurons through axon terminals. By using the similar this architecture that a brain-style computational device is richly connected to one another, an artificial neural or nonlinear network is constructed. The function it computes is determined by the pattern of connections. Based on the models of ANNs, many new topologies and algorithms are developed.

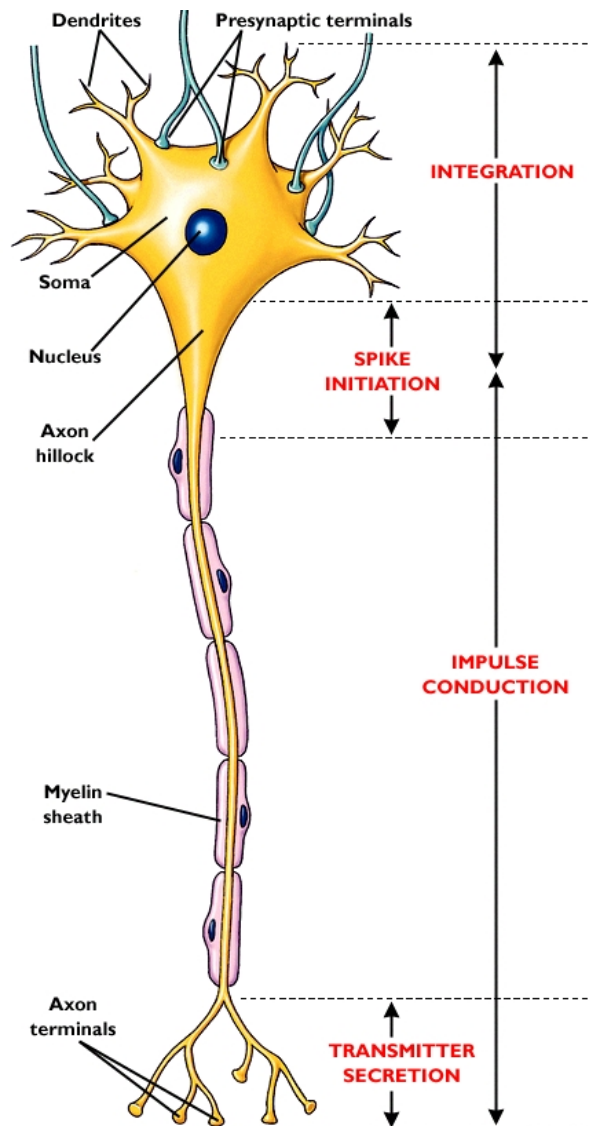


Fig. 1.3 The biological model of a neuron cell which contains the cell body (nucleus) and the I/O terminals of dendrites and axon terminals.

Work on the models of ANNs has a long history. Development of detailed mathematical models has begun about 60 years ago in the work of McCulloch and Pitts [18], Hebb [19], Rosenblatt [20], Widrow [21], *et al.* In 1980s, the work by Hopfield [1]-[10], Rumelhart and McClelland [22], Sejnowski [23], Feldman [9], Grossberg [24]-[25], *et al.* has led to a new resurgence of the field. There seems to be five reasons for the rebirth. First, the faster and faster computer makes it possible to simulate and experiment with much larger and more

interesting networks than that in 1950s and 1960s. Second, it is believed that the faster computers must be in parallel computation. However, it is generally easier to build parallel computers than to find algorithms that are efficient. Third, the empirical tools of neuroscience are expanding and more and more knowledge about how the neuron functions is learned. Besides, it is hoped that the theoretical tools developed in the study of neural network computational systems will allow for the modeling of the real neural networks. Fourth, theoretically, Hopfield provides the mathematical foundation for understanding the dynamics of the recurrent networks. The mathematical model has been extended and applied by Hinton and Seinowski [26], Cohen and Grossberg [27], Smolensky [28] and a number of scientists to provide more mathematical models and solve important problems such as optimization. Fifth, with the extension of Rosenblatt, Widrow, and Hoff's work dealing with learning in a complex, multi-layer network [20]-[21], this provided a technique, known as the back-propagation learning algorithm [29], is developed that multilayer perceptron-like devices can be reliably trained.

The interest in ANNs comes from the networks' ability to mimic human brain as well as its ability to learn and respond. Adaptation or learning is a major focus of ANN research that provides a degree of robustness to the ANN model. An adaptive linear element is a single neuron of McCulloch-Pitts type, where its weights are determined by the normalized least mean square (LMS) training law. The LMS learning algorithm was originally proposed by Widrow and Hoff [21]. This learning rule is also referred to as the delta rule. It is a well-established supervised training method that has been used over a wide range of diverse applications [30]-[33]. The simplest architecture of an adaptive linear element is shown in Fig. 1.4. In the simplest adaptive linear element, the neuron with a linear activation function is used. The weights are adjusted by the LMS error of comparing the output with the desired output.

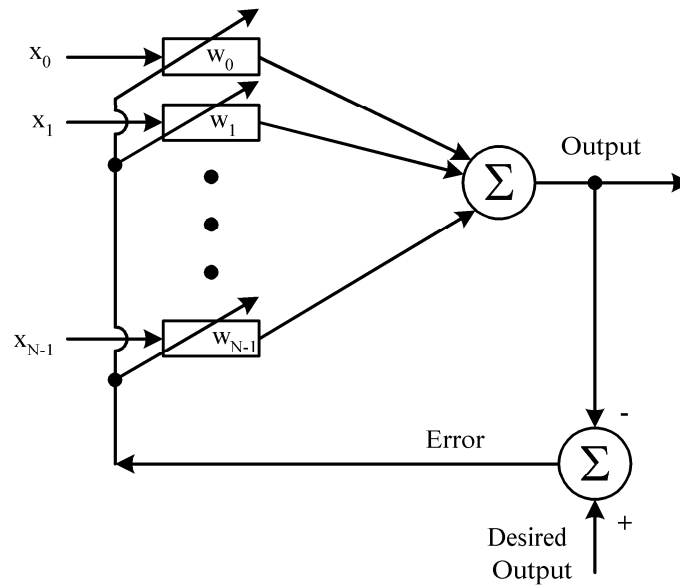


Fig. 1.4 The simplest architecture of an adaptive linear element where its weights are determined by the normalized least mean square training law by a preset desired output.

Once the weights are properly adjusted, the response of the trained unit can be tested by applying various inputs which are not in the training set. If the network produces consistent responses to a high degree with the test inputs, it is said that the network can generalize. Therefore, the process of training and generalization are two important attributes of the network. Similar to the adaptive linear element, the original idea of the perceptron has been developed by Rosenblatt in the late 1950s along with a convergence procedure to adjust the weights [20]. The original perceptron convergence procedure is developed by Minsky and Papert [34] as shown in Fig. 1.5. The perceptron [20] by Rosenblatt is based on the McCulloch-Pitts model of the neuron with the hard limitation activation function where the inputs are binary and no bias is included. The perceptron of Minsky and Papert is similar to that by Rosenblatt except for the addition of an activation function and the non-zero value of the threshold w_0 [34].

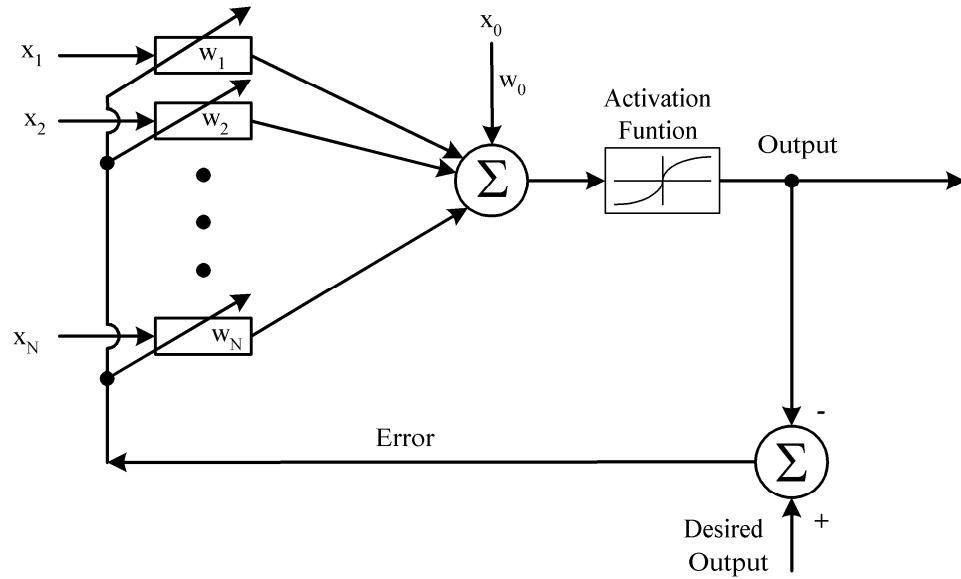


Fig. 1.5 A perceptron with a sigmoidal activation function. The threshold value w_0 are initialized to small non-zero values.

The perceptron convergence procedure and its variants are limited to simple one-layer networks involving only input and output units. It maps similar input patterns to similar output patterns. The similarity of patterns in the system is determined by their overlap which is decided outside the learning system by whatever produces the patterns. Therefore, the constraint of the system leads to an inability to learn certain mappings from input to output. In a multilayer network, the information coming to the input units is re-coded into an internal representation and the outputs are generated by the internal representation rather than by the original pattern. Multi-layer perceptrons are feed-forward nets with one or more layers of nodes between the input and output nodes called hidden layer. A simple two layer perceptron with one layer of hidden units is shown in Fig. 1.6. Each node is a perceptron with hard limiting nonlinearity. The hidden layer can be increased as the tasks are more complex. A

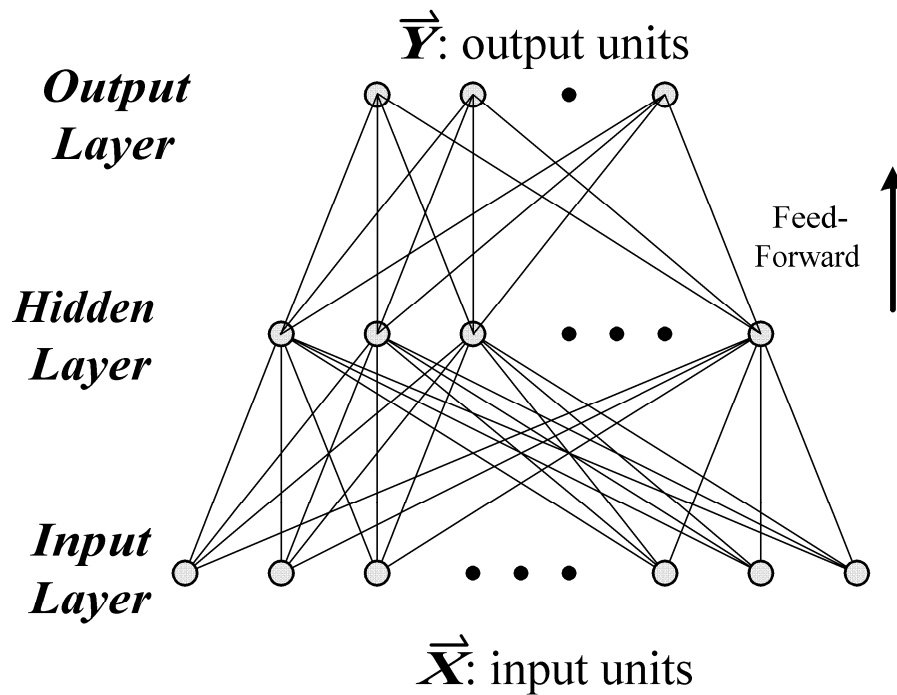


Fig. 1.6 A simple three-layer network which contains input, hidden, and output layers.

single-layer perceptron can form half-plane decision regions whereas a two-layer perceptron can form any, possibly unbounded, convex region in the space spanned by the inputs. Moreover, a three-layer perceptron can form arbitrarily complex decision regions and can separate the meshed classes. Hence, no more than three layers are required in perceptron-like feed-forward nets. Similar behavior is exhibited by multi-layer perceptrons with multiple output nodes when sigmoidal nonlinearities are used and the decision rule is to select the class corresponding to the output node with largest output. The behavior of these nets is more complex because decision regions are typically bounded by smooth curves instead of by straight line segments and analysis is thus more difficult. As a result, these nets can be trained with the new back-propagation training algorithm [29]. The back-propagation algorithm uses a gradient search technique to minimize a cost function equal to the mean square difference between the desired and the actual net outputs. The network is trained by initially selecting

small random weights and internal thresholds and then presenting all training data repeatedly. Weights are adjusted after every trial using side information specifying the correct class until weights converge and the cost function is reduced to an acceptable value.

One important organizing principle of sensory pathways in the brain is that the placement of neurons is orderly and often reflects some physical characteristics of the external stimulus being sensed [35]. Kohonen presents the algorithm which produces the self-organizing feature maps similar to those that occur in the brain [36] as shown in Fig. 1.7. Output nodes are extensively interconnected with many local connections. The algorithm that form feature maps requires a neighborhood to be defined around each node and the neighborhood slowly decreases in size with time. With the algorithm, a speech recognizer as a vector quantizer is proposed [37].

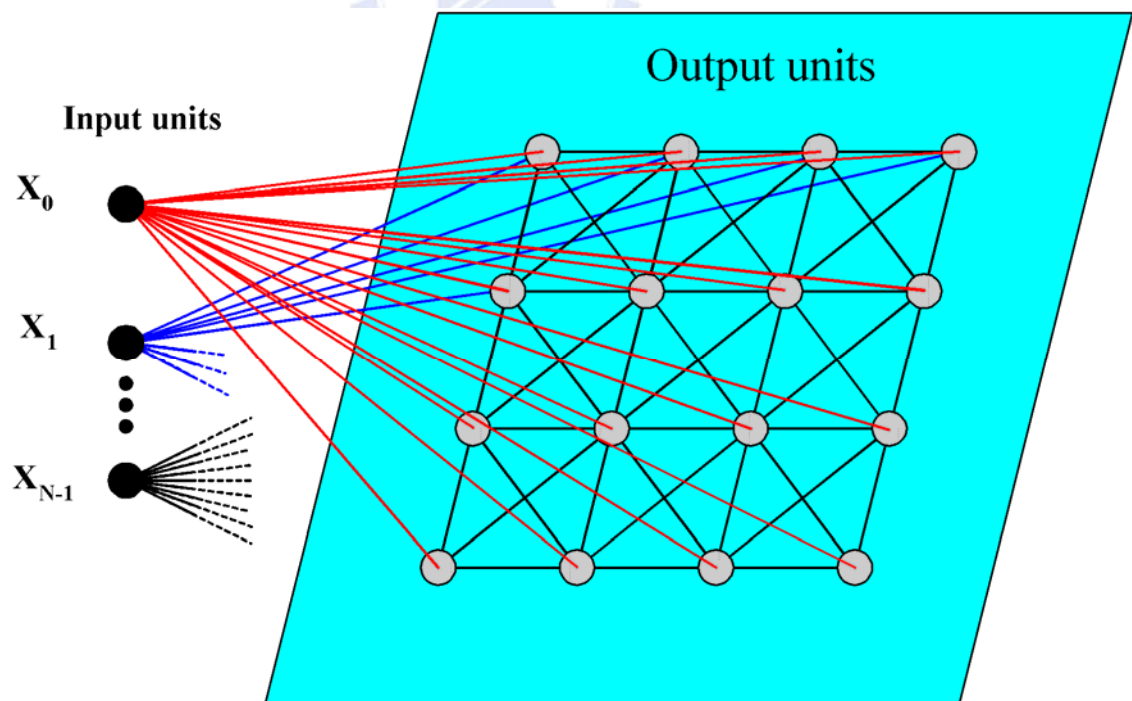


Fig. 1.7 Two-dimensional array of Kohonen's self-organizing feature maps.

Similar to Kohonen's two dimension array of self-organizing feature maps, the cellular neural/nonlinear network (CNN) has first been presented as a preferred implementation of locally connected neural networks [6]-[7]. Unlike the former learning models, CNN involves a large-scale nonlinear analogic architecture for real time processing. In 1993, a further architecture of CNN universal machine is presented [38]-[39] and many researches are verified by the cellular nonlinear network universal machine (CNNUM) [38]-[46]. CNN consist of arrays of elementary processing units (cells) and each one is connected to a set of adjacent cells. This local connection property makes CNN physical design easy, especially for the translational invariant CNNs. Chua and Yang's CNN cell circuit model [6]-[7], [40], where the neuron is model by a resistor R shunt with a capacitor C, is shown in Fig. 1.8 and can be presented by the equation

$$C \frac{dV_{x_{ij}}}{dt} = -\frac{V_{x_{ij}}(t)}{R} + I_Z + \sum_{\substack{kl \in S_{ij} \\ kl \neq ij}} [Ga_{kl} V_{y_{kl}}(t) + Gb_{kl} V_{u_{kl}}] \quad (1.2)$$

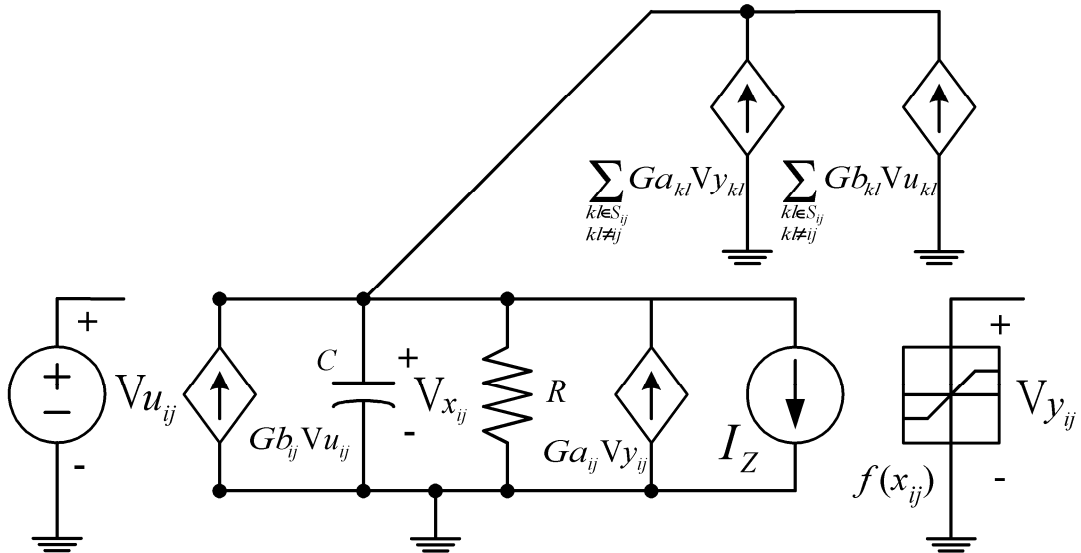


Fig. 1.8 The RC circuit model of a CNN cell.

where $V_{x_{ij}}$ is the state voltage, $V_{y_{ij}}$ is the output voltage, and I_z is the threshold current of neuron cell (i, j) . $G_{a_{kl}}$ and $G_{b_{kl}}$ are the transconductance set that can multiply the state voltage and the output voltage, and are called templates **A** and **B**, respectively. As a result, all the currents are summed and introduce a voltage drop, state voltage, on the neuron of a resistor R and a capacitor C . With the core architecture as shown in Fig. 1.9 [38]- [40] demonstrating such a large-scale array of CNN and the further architecture with logic operational units and memories of CNUM, many algorithms and applications have been investigated and proposed [38]-[46].

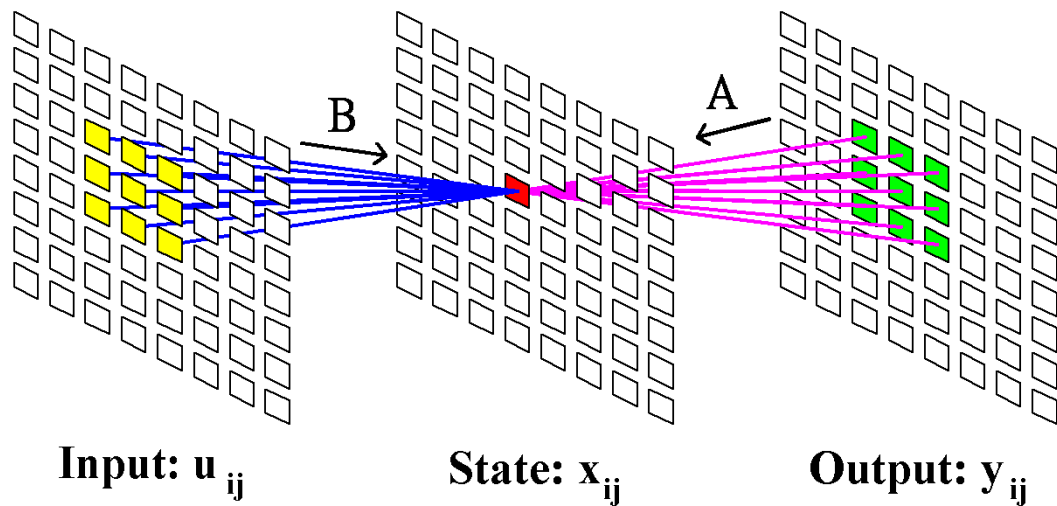


Fig. 1.9 The core architecture of CNN with templates **A** and **B**.

1.2 RESEARCHES ON CNNs AND THEIR APPLICATIONS

The cellular nonlinear/neural network (CNN) which was proposed by Chua and Yang in 1988 [6]-[7], [40] involves a large-scale nonlinear analogic architecture for real-time signal processing. Similar to the composition of the cellular automata, it is comprised of a massive aggregation of regularly spaced circuit clones, called cells, which communicate with each

other directly and locally. In a basic CNN, each cell is connected to its nearest layer of neighboring cells. Such a CNN, called a 3×3 neighborhood CNN, is the most popular CNN structure. Their local connectivity makes CNNs easy to be implemented in a VLSI design and there is great tolerance to errors depends on templates. Some research results and their applications are listed as following.

A. Autowaves, Chaotic, and oscillatory elements

The studies of dynamic phenomena in arrays composed of autowaves, chaotic, and oscillatory elements are very important for understanding natural phenomena in biology, chemistry, physics, etc [47]-[51]. Pattern formation and various types of autowaves, such as excitability waves, concentration waves, and so on, are discussed [47]-[48], [52]-[58]. CNNs are usually used as the approximations of the various types of nonlinear partial differential equations [52]-[55]. Chaos engineering has also been steadily studied in Japan and many applications are developed such as controlling power for the thawing function of microwave ovens [56]. Moreover, it can be applied to associative memory networks that have been intensively studied in the field of artificial neurocomputing [59]-[61] and some applied the chaotic structure in solving combinatorial optimization.

B. Recognition

Neural networks have been used in a number of applications due to their ability to learn and generalize. One application of the learning ability is to recognize different patterns such as characters and sounds [41]-[42], [43], [62]-[70]. Dual cellular neural network architecture can extract the global features of the handwriting and makes the decision [62]-[63]. Character template learning operates with separated characters on a basis of the character patterns or applies segmentation and recognition of text line image simultaneously via dynamic programming [64]. Ratio-memory CNN (RMCNN) can learn correlations between cells and

the features of images are stored in the ratio memories [65]-[67]. As a result, it can recognize the noisy images with templates generated by ratio-memories. For the human immune systems, sounds also can be recognized and detected [41]-[42]. The basic idea is to make a system search video images for objects that are not supposed to be there and trigger an alarm message when it occurs [43], [68]-[70].

C. Classification and Segmentation

Classification and segment are also the mainly functions of neural networks and sometimes go along with recognition or detection [71]-[72]. Classification and segment can be applied on the blind source separation [73], motion estimation for MPEG-4 encoder [74]-[78], bubble-debris classification [71]-[72], DNA microarrays analysis [79]-[81], image descreening [82]-[83], object-oriented segmentation [38]-[45], [84] etc. Genetic algorithm is attempted to minimize the objective function or the cost function and use the independent properties of initial conditions and the domain of applications combined with the implicit parallelism [82]-[83]. For the algorithm, three kinds of different CNN templates (average, inverse and time-interpolated templates) can be trained by GA [85], while ICA mixture models are conditional independence model and unsupervised classification [73].

D. Image Processing

CNN has shown a vast computing power, especially for image processing [6]-[8], [39]. Early CNN implementation were designed to perform one specific function in image processing such as edge detection, connected component detection, or hole filling. Recently, the ability to change or program the template values [86]-[90] has made image processing easily to be studied and verified. Filtering is one of the interested areas for image processing [91]-[96]. Besides, some studies focus on color image or gray level image processing by using the state of neuron and multilayer structure [97]-[98] and are applied on medical image

processing, image restoration, and weather forecasting. Many other tasks can also be resolved such as halftoning of digital images [99]-[100], image compression [101]-[102], skeletonization etc.

There still many applications of CNN such as optimization [103]-[104], control systems [105]-[109] etc. Furthermore, some has applied the fuzzy set theory into CNN architecture [83], [110]-[113]. Fuzzy cellular nonlinear networks (FCNN) can be used as an interface between the human expert and the classical CNN [114]. Meanwhile, there are some researches studying the discrete-time CNN (DTCNN). DTCNN contains two categories: an analog-array architecture and a digital-pipeline architecture. Both continuous-time CNN (CTCNN) and DTCNN have powerful ability of parallel image processing. The growth of CNUM and DTCNN processor has made the studies on applications of CNN more easily.

1.3 REVIEW OF LNCNNs AND RMCNNs

A. LNCNNs

The cellular neural network proposed by Chua and Yang [6]-[8], [40], involves a large-scale nonlinear analogic architecture for real-time signal processing. In 1992, a programmable CNN universal machine (CNUM) is proposed by Chua and Roska [115]. Many tasks can be resolved by CNUM [38]-[45] and even now, many applications are studied with CNUM. However, in many CNN applications such as image halftoning [99] and subcortical visual pathway [40], [116], the large-neighborhood templates are required. Although the large-neighborhood template can be decomposed into 3×3 templates [117]-[118], it needs more efforts and more iterations to deal with a task and, hence, more energy is consumed. Hence, in 2001, a large neighborhood CNN with a compact neuron-bipolar junction transistor (vBJT) is proposed by C. Y. Wu and W. C. Yen [119]. A

device called lambda bipolar transistor [120] is applied to be a neuron called neuron-lambda BJT ($v\lambda$ BJT), where the bipolar junction transistor is replaced by v BJT. In the Wu and Yen's LNCNN, one NMOS device is used to be a synaptic gain controller and makes the whole chip smaller. Meanwhile, $v\lambda$ BJT is also used by C. Y. Wu and C. W. Hsiao [121] to implement a LNCNN. In both LNCNNs, the structure is similar but the circuit implementation methods are different and they can realize the templates with $r > 1$.

In Wu and Yen's [119] LNCNN, there is only one single path to link cells and transfer the signals one by one. Although single path can make the connections simple and implemented easily, it also means that the two synaptic gain blocks for bridging cells attach the input of one block to the output of the other. The loop gain of these two gain blocks makes complicated the mapping between the gains of the synaptic blocks and the coefficients of the templates. Because the degree of freedom is less than the coefficients of the LN templates, the coefficients of second layer can not be determined arbitrarily under the constraint of propagating connections. Hence, it cannot realize the LN templates arbitrarily due to the architecture. However, in Wu and Hsiao's LNCNN [121], the path is separated into bi-direction but templates **A** and **B** in LNCNNs are separated and designed in the circuit. This takes a large area to realize them separately. Moreover, because BJTs are used to generate LN templates, the gain of the used BJTs is hard to be predicted and it still causes the coefficients of a template to be asymmetric. Furthermore, in both design, $v\lambda$ BJT are used to realize the neuron with a self-feedback, but the self-feedback term is not a fixed value and cannot be adapted arbitrarily.

B. RMCNNs

The previous researches on the learning neural networks with associative memory have been studied since 1995 [65]-[66] and still keep on going [67], [122]-[125]. The learning

algorithm is based on Grossberg mathematical model called the outstar to realize the ratio of the learned weights. The outstar as a classical conditioning learner can learn the related things and be refreshed by reminding and memorize the relative strengths of the input pattern but not the absolute values. The associative memory is also called ratio memory which is first proposed by J. F. Lan and C. Y. Wu in 1995 [66] and implemented with an analog neural net with on-chip learning.

In 2000, the ratio memory has been applied on cellular neural network called RMCNN which is proposed by C. Y. Wu and C. H. Cheng. The ratio memory is incorporated with the modified Hebbian learning and the ratio memory generates the absolute weights and transforms them into template A to perform the image recognition. The ratio memory stores the correlations of neighboring cells and the information of the correlations is enhanced on a capacitor with a small leakage current. Hence, due to such a small leakage, a long storage time can be achieved. By utilizing the leakage of the capacitor, an elapsed time is also applied to extract or enhance the features with large correlations to recognize the noisy patterns. Although the small leakage during an elapsed time can enhance the feature, the uncertain leakage currents in cells may make the enhancement different from that with the ideal leakage current. Moreover, a long elapsed time may destroy the correlation on capacitors.

An RMCNN chip where the learning circuitry is integrated on-chip makes the learning task operate alone without other external aids. Moreover, the learning algorithm would generate numerous space-variant templates. If the learning process were performed off-line, it must take a long loading time for each cell. In 2002, the modified Hebbian learning algorithm in RMCNN is re-modified. A self-feedback term is introduced to make the output of each cell be stable at a saturated point and the RMCNN with a self-feedback term is called

self-feedback RMCNN (SRMCNN). The feature enhancement effect of the ratio memory remains during the operation of SRMCNN.

1.4 RESEARCH MOTIVATION AND ORGANIZATION OF THIS DISSERTATION

It is believed that the large-neighborhood templates have more powerful functions and higher efficiency even in discrete time CNN (DTCNN) [117]. Although the large-neighborhood template can be decomposed into 3×3 templates, it takes more energy and time and most of decomposition methods are implemented in DTCNN but not in CTCNN. However, the connections of LNCNN are very complicated. Hence, several researches on LNCNN have been developed. In [119], a single path along one row or one column is constructed for simplification. The bi-directional signals pass through the single path. This makes it unable to generate arbitrary templates and also makes the mapping between the gain and the coefficients complicated. In [121], the paths are separated but the gain block is designed by using BJTs. The bi-directional inputs in the gain block pass through different numbers of BJTs due to the constraint of BJTs. Hence, it is hard to get a precise gain in the design. In both design of [119] and [121], $v\lambda$ BJT is used to realize the activation function with a self-feedback but the value of feedback cannot be determined.

Based upon the above description, the aim of this dissertation is to explore a new indirectly connective LNCNN. In the designed LNCNN, the degree of freedom should be higher than the coefficients of the LN templates so that the coefficients of second layer can be determined arbitrarily under the constraint of propagating connections. Furthermore, the proposed LNCNN chip, where the non-recurrent terms generated by templates \mathbf{B} and \mathbf{Z} are stored [126],

is designed to decrease the synaptic path. The bi-directional characteristic of the propagating connections is kept and is separated into two connectional nodes to prevent the closed loops. Meanwhile, more synaptic blocks are added for all possible templates with the constraint of propagating connections. An experimental chip has been designed and fabricated using 0.18- μm CMOS technology. The LNCNN chip with the array size of 20×20 can realize the function of the diamond-shaped large-neighborhood templates. The total chip area is $1543 \mu\text{m} \times 1248 \mu\text{m}$ and the area of a single cell is $33.58 \mu\text{m} \times 43.15 \mu\text{m}$. The power is 0.7 mW on standby and 18 mW in operation with a 1.8 V supply voltage and a system clock frequency of 20 MHz. With the LNCNN chip, the LN function of human illusion is realized successfully.

With regard to RMCNN [65]-[67], [122]-[125], the correlations are stored by a capacitor and leaks in an elapsed time by an intrinsic leakage current. The leakage current makes the smaller correlations disappear and enhances the large correlations. If the elapsed time is too short, the performance of the enhancement cannot be obvious. However, long elapsed time would make the correlations become 0 and cause the ratio weights generated by the correlations to be meaningless. The templates are generated according to the correlations between cells by using the modified Hebbian learning. However, how the ratio weights take effect in the recognition period has not discussed. By analyzing the effect, it can be helpful to the improvement of the recognition rates.

Hence, another aim of this dissertation is to design an RMCNN without elapsed time. In the design, the method using elapsed time for generating the templates is replaced by that using the comparator to approximate the result of original method. With this new method, the ratio memory, which is realized by a divider, can be implemented by the comparator easily. An RMCNN chip not requiring elapsed time has been designed and fabricated using TSMC 0.35- μm 2P4M mixed-signal technology. 3 Patterns are learned and recognized with the

proposed architecture and the results are analyzed and discussed. The total chip area is $4560 \mu\text{m} \times 3900 \mu\text{m}$ and the area of a single cell is $400 \mu\text{m} \times 250 \mu\text{m}$. The total power consumption is 87 mW in operation with a supply voltage of 3 V with a system clock frequency of 10 HMz.

Moreover, the mathematical analysis by using Gaussian noise is also discussed in this dissertation. It is found that the decision of the output does not locate at the optimum point according to the statistic results. The results indicate the requiring of the threshold. The proposed recursive learning [145] RMCNN can gather the information of the error probability and increasing the recognition rates and number of learned patterns. With recursive learning, the number of the learned patterns by RMCNN requiring no elapsed time is raised from 6 to 8. Hence, recursive learning indeed can raise the recognition rates.

This dissertation contains five chapters, which include introductions, the design and analysis of a CMOS large-neighborhood CNN with propagating connections, the design and analysis of a CMOS ratio-memory CNN without elapsed time, the analysis of the recursive learning RMCNN.

The rest of this dissertation is organized into 4 chapters. In chapter 2, the analysis and design of large neighborhood CNN are indicated. In chapter 3, RMCNN requiring no elapsed time is proposed and designed. In Chapter 4, the correlation between the templates of RMCNN requiring or requiring no elapsed time and the noise is discussed. Finally the conclusion is given in chapter 5. More details are illustrated as following.

In Chapter 2, the large-neighborhood CNN has been analyzed and designed. The propagating connections are used to realize the diamond templates. With the diamond templates, the Matlab simulations are also made to verify the large-neighborhood functions and the results are compared with those of 5×5 templates. Otherwise, the low power and simple design can make LNCNN suitable for large-scale array. The LNCNN chip has been

fabricated with 0.18- μm 1P6M technology. The large neighborhood function of human illusion is measured and it proves that the LNCNN chip can be applied on the binary image processing.

In Chapter 3, RMCNN requiring no elapsed time is analyzed. In the original operation of RMCNN, the long elapsed time is required. However, with a long elapsed time, some of the correlations will be destroyed and the feature enhancement of the ratio weight, hence, cannot take effect. As a result, RMCNN requiring no elapsed time has been proposed to avoid this situation and, as well, the multiplier-divider is not required anymore and replaced with a comparator and a counter. Therefore, the design of the RMCNN requiring no elapsed time chip can be simpler. By using 0.35- μm 2P4M, the RMCNN requiring no elapsed time has been fabricated and the measurement results are discussed. Moreover, large-neighborhood RMCNN requiring no elapsed time is also simulated and the modified RMCNN requiring no elapsed time is proposed.

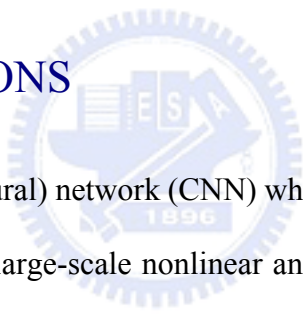
In Chapter 4, the input of each pixel with a Gaussian noise is discussed when an assumed RMCNN template is considered. According to the analysis of the output probability density, the decision of the output is not located at an optimum point. Hence, the recursive learning RMCNN is proposed to gather the error probability density of the pixel. With the error probability density, the threshold values are decided to lower the error rate. To verify the effect of the recursive learning, RMCNNs with or without recursive learning are simulated and compared in this chapter.

Finally, the conclusion of this dissertation is summarized in Chapter 5. The future work about the further implementation of CNNs and their applications is also addressed in this chapter.

CHAPTER 2

THE DESIGN AND ANALYSIS OF A CMOS LARGE-NEIGHBORHOOD CNN WITH PROPAGATING CONNECTIONS

2.1 INTRODUCTIONS



The cellular nonlinear (neural) network (CNN) which was proposed by Chua and Yang in 1988 [6]-[7], [40] involves a large-scale nonlinear analogic architecture for real-time signal processing. Similar to the composition of the cellular automata [127]-[128], it is composed of a massive aggregation of regularly spaced circuit clones, called cells, which communicate with each other directly and locally. In a basic CNN, each cell is connected to its nearest layer of neighboring cells. Such a CNN, called a 3×3 neighborhood CNN, is the most popular CNN structure. Their local connectivity makes CNNs easy to be implemented in a VLSI design. So far, many 3×3 neighborhood CNN VLSI chips have demonstrated their capabilities in realizing real-time signal and parallel processing functions [39], [119], [126], [129]-[135].

The CNN universal machine [38], [39] is a programmable CNN, which can perform several complicated functions. Recently, research on the CNNUM has been conducted and successfully implemented. Current CNNUMs are based on the 3×3 neighborhood CNN

structures [126], [129]-[133] and 3×3 neighborhood templates. Some applications [136]-[137] are verified by using the CNUM. However, 3×3 neighborhood CNNs with the nearest neighborhood are restricted in their ability to solve complex problems efficiently. Although a large-neighborhood template can be transformed into several 3×3 neighborhood templates [118], [138], the multiple operating steps with 3×3 neighborhood templates require more time and more power.

It is more efficient to construct a large-neighborhood CNN (LNCNN), which can perform functions using large-neighborhood templates. In an LNCNN, each cell is connected to more than one layer of the neighboring cells. Generally, an LNCNN is difficult to be implemented in a VLSI design through direct wire connections among the 3×3 neighborhood CNN cells. Recently, however, a design for a LNCNN has been proposed and implemented by using a new device called the neuron BJT (vBJT) [119]-[121]. Based on the vBJT, an LNCNN with symmetric templates has been designed [119]-[121]. The LNCNN with asymmetric templates has also been proposed with some limitations in realizing large-neighborhood templates [119].

In this work, a new improved low-power CMOS compact LNCNN architecture with propagating synaptic connections [139]-[140] is proposed and analyzed. In the proposed kernel unit, only one layer of the neighboring cells is connected, but it can realize large-neighborhood diamond-shaped templates in the first two neighboring layers. Thus, complicated wire connections to farther cells can be avoided. The propagating synaptic connections can be used not only in horizontal and vertical directions, but in diagonal directions. As a result, the circular symmetric templates can be realized. Moreover, the circuitry can be shared between templates A and B in the proposed architecture. This results in a simpler architecture and smaller chip area. To realize the proposed architecture, the

low-power neuron and synapses have been designed using CMOS current-mode circuits without static current paths. In addition, an experimental chip has been designed and fabricated using 0.18- μm CMOS technology. The LNCNN chip with the array size of 20×20 can realize the function of the diamond-shaped large-neighborhood templates. The LNCNN functions of diffusion, de-blurring, and Muller-Lyer illusion has been verified successfully. Meanwhile, the functions of erosion and dilation are expanded with the diamond-shaped LN templates. The total chip area is $1543 \mu\text{m} \times 1248 \mu\text{m}$ and the area of a single cell is $33.58 \mu\text{m} \times 43.15 \mu\text{m}$. The power is 0.7 mW on standby and 18 mW in operation with a 1.8 V supply voltage and a system clock frequency of 20 MHz. As a result, the proposed kernel unit has a very simple structure, small dc power dissipation, and small chip area, which can be applied to the CMOS implementation of an LNCNNUM with a huge kernel array size. Also, with the hardware of the proposed LNCNN structure, many new the functions or new templates of LNCNN can be explored.

In Section 2.2, the LNCNN model, the global architecture of the kernel unit of the LNCNNUM and the components of each regular cell are described. In Section 2.3, the CMOS circuits of the neuron, synapses, PSW, and analog memory in the proposed LNCNN are described and HSPICE simulation results are presented to verify the circuit functions. The overall chip architecture in the design is also illustrated. In Section 2.4, the measurement results are shown and discussed. Finally, a concluding section is provided.

2.2 ARCHITECTURE AND MODELS

For a standard CNN, the state equation is written as [6]-[7], [40]

$$\dot{x}_{ij} = -x_{ij} + Z_{ij} + \sum_{C_{kl} \in S_{ij}} A_{kl} y_{kl} + \sum_{C_{kl} \in S_{ij}} B_{kl} u_{kl}$$

(2.1)

where x_{ij} , y_{ij} , and u_{ij} are the state, output, and input of the neuron cell C_{ij} in a CNN array, respectively; the coefficient Z_{ij} , called the template Z , is the threshold of the neuron cell C_{ij} ; and, A_{kl} and B_{kl} are the coefficients, called templates A and B , respectively, which are multiplied with output y_{kl} and input u_{kl} of the cell C_{kl} , respectively in the sphere of influence (S_{ij}) of the neuron cell C_{ij} . The two sets of products are accumulated over all the cells C_{kl} in the sphere of influence (S_{ij}) of the neuron cell C_{ij} . Where there are non-zero coefficients for templates A and B at the neighboring cells $C_{(i\pm r)(j\pm r)}$, r is an integer called neighborhood of radius. If r is greater than 1, it is called a large-neighborhood CNN.

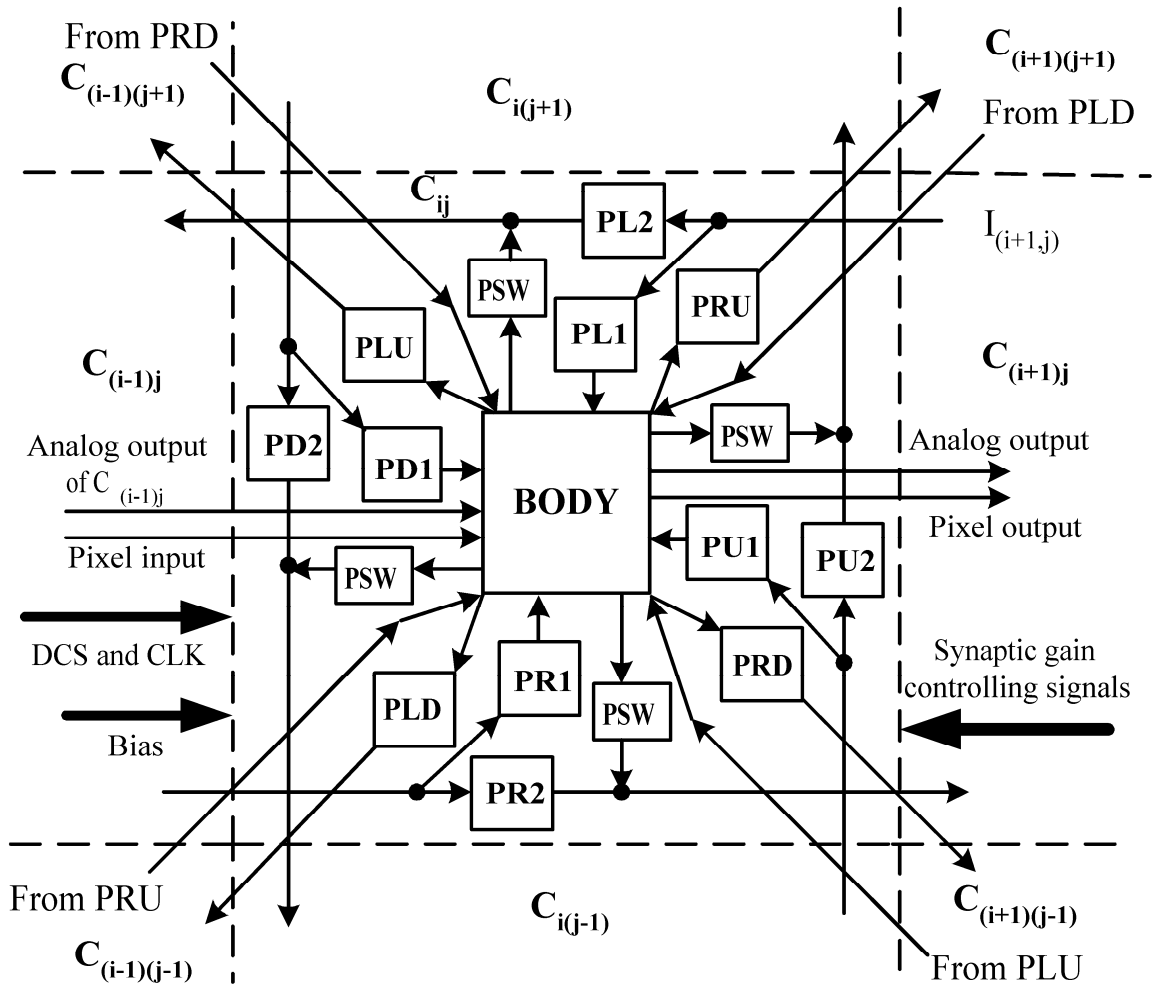


Fig. 2.1 The architecture of a LNCNN kernel unit.

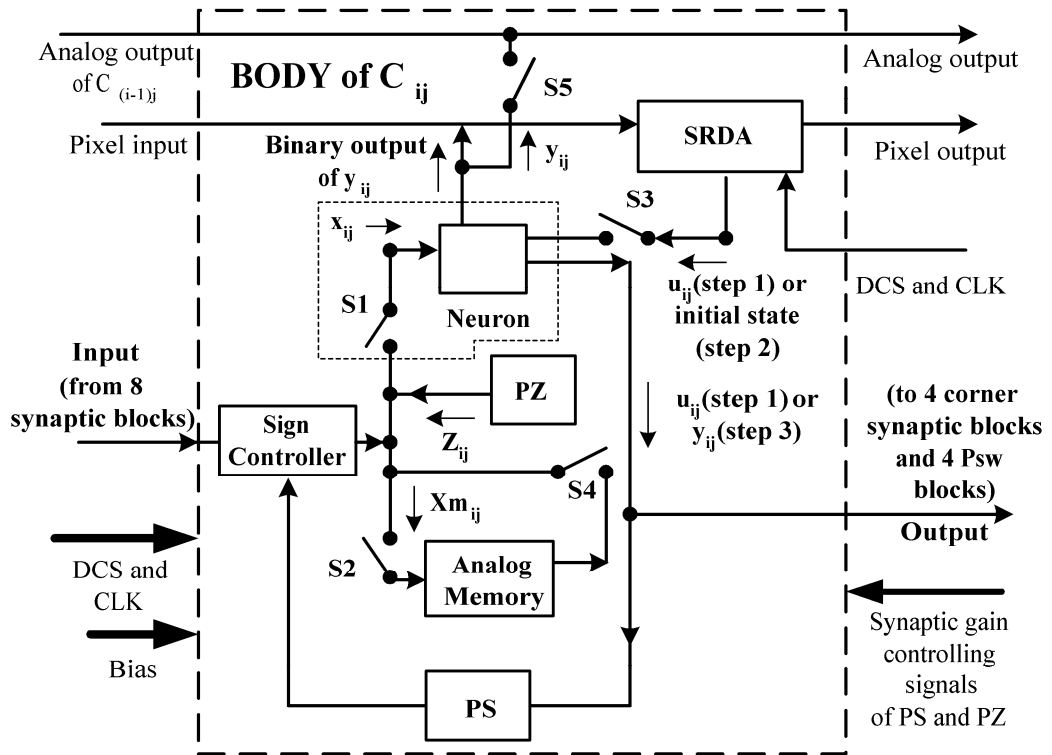


Fig. 2.2 The structure of the BODY in Fig. 2.1.

The architecture of the proposed LNCNN kernel unit is shown in Fig. 2.1 where the region surrounded by the broken line represents one neural cell C_{ij} defined by the coordinate. In C_{ij} of Fig. 2.1, the BODY shown in Fig. 2.2 consists of the neuron, analog memory, synapses, and control circuits. The PU1, PD1, PL1, PR1, PRU, PRD, PLU, PLD, PU2, PD2, PL2, and PR2 are all synapses, which can multiply input signals and result in different gains which are controlled by the synaptic gain controlling signals. As a result, these synapses can be combined to realize the coefficients of templates A and B , except the center coefficients A_{ij} and B_{ij} . Among these synapses, PU2, PD2, PR2 and PL2 can propagate signals to the cells farther than the neighboring cells. For example, the signal $I_{(i+1), j}$ from $C_{(i+1)j}$ can pass through PL2, be multiplied by the gain of the PL2, and then reach $C_{(i-1)j}$. These connections used to realize large-neighborhood templates are called the propagating connections. PLU, PLD, PRU, PRD,

PL1, PD1, PR1, and PU1 are used to connect the neighboring cells directly. These connections among the nearest neighboring cells are called direct connections. PSW is a current switch and the gain of PSW is 1. The polarities of the signals sent out of the BODY in upward, downward, leftward, and rightward directions are determined by the four PSWs. The output current of the PSW is combined with that sent from the synapse of the propagating connections in the former cell. Eventually, the resultant output is sent into the synapse of the next cell.

The DCS and CLK in Fig. 2.1 are digital controlling signals and clock signal, respectively, to control logic circuits and switches in the kernel unit. The Pixel input signal of one cell is connected to the Pixel output signal of the former cell. For example, the Pixel input of C_{ij} comes from the Pixel output of $C_{(i-1)j}$. This signal transfers the input pattern to each cell and the output pattern to the output pads in series. The arrows between the cells are connected to the relative positions of each cell. For example, the arrow line from the PRU of C_{ij} is connected to the BODY of $C_{(i+1)(j+1)}$ and similarly, the arrow line from $C_{(i+1)(j+1)}$ into the BODY of C_{ij} comes from the PLD of $C_{(i+1)(j+1)}$.

In the structure of the BODY shown in Fig. 2.2, the switches S1-S4 are controlled by the signals of DCS and CLK, and the switch S5 is controlled by a 5-bit decoder. The SRDA contains one shift register, digital controlling logic, and a 1-bit D/A converter (DAC) inside. The use of shift register makes chip implementation realizable. It is impracticable to implement a large capacitor to store the analog signal in each cell during the overall operational period. Because shift registers can be refreshed by sending a set of data into the chip, there is no additional signal to reset shift registers. The Pixel input of C_{ij} can be transferred to the next cell by the SRDA. The SRDA provides the binary input signal u_{ij} or the initial state value $x_{ij}(t=0)$ of each cell during the operation. After the operation, the SRDA can store the binary output of y_{ij} from the neuron and the analog output y_{ij} can be read out by turning on the switch S5.

In Fig. 2.2, the Neuron is a neuron with a standard piecewise linear ramp function:

$$y_{ij} = f(x_{ij}) = \frac{1}{2}|x_{ij} + 1| - \frac{1}{2}|x_{ij} - 1|. \quad (2.2)$$

The input of the BODY comes from the summation of the eight synaptic outputs as drawn in Fig. 2.1 and the output of the BODY is duplicated eight times and sent to the four PSWs and four corner synapses PRU, PRD, PLU, and PLD. The PZ generates the coefficient Z_{ij} where the PS is the synapse that generates the center coefficients A_{ij} and B_{ij} of templates \mathbf{A} and \mathbf{B} , respectively. The Analog Memory is used to store following equation:

$$Xm_{ij} = Z_{ij} + \sum_{C_{kl} \in S_{ij}} B_{kl} u_{kl}. \quad (2.3)$$

Before the Neuron, there is a Sign Controller which is used to adjust the polarities of the signals from the nine synapses.

In the first step of the operation period, only the signal of Xm_{ij} in (2.3) is calculated, sampled and stored by the Analog Memory. In addition, the digital code of the input u_{ij} is sent from the Pixel input to the shift register in the SRDA and stored. Switches S2 and S3 are closed and S1 and S4 are left open. At this time, all the synapses are set to certain gains to generate the template \mathbf{B} and the PZ is set to generate Z_{ij} . The piecewise linear ramp function of the neuron is turned off. The input signal u_{ij} from the SRDA passes through the Neuron. At this moment, the output of the neuron is the same with the input signal u_{ij} from the SRDA, multiplied with the template \mathbf{B} and combined with Z_{ij} to form Xm_{ij} , which is instilled into the Analog Memory. After the switch S2 is opened, the Xm_{ij} is stored in the analog memory.

In the second step, the digital code of the initial state $x_{ij}(t = 0)$ of the desired function is sent from the Pixel input to the shift register in the SRDA and stored. S1 and S2 are open and S3 and

S4 are closed. X_{mij} is read out and the neuron is set to the initial state $x_{ij}(t = 0)$ provided by the SRDA. Meanwhile, the gains of all the synapses are set to generate the template A . In the third step of the operation period, the S1 switch is turned on and the S3 switch is turned off. A feedback loop is constructed and then the calculation of (1) is started. After the operation is completed, the readout period commences. The output y_{ij} is converted to binary form and the binary output is sent to and stored at the shift register in the SRDA. As the input pattern of the next operation is sent into the LNCNN, the output pattern of the former operation can be read out from the Pixel output of the last cell.

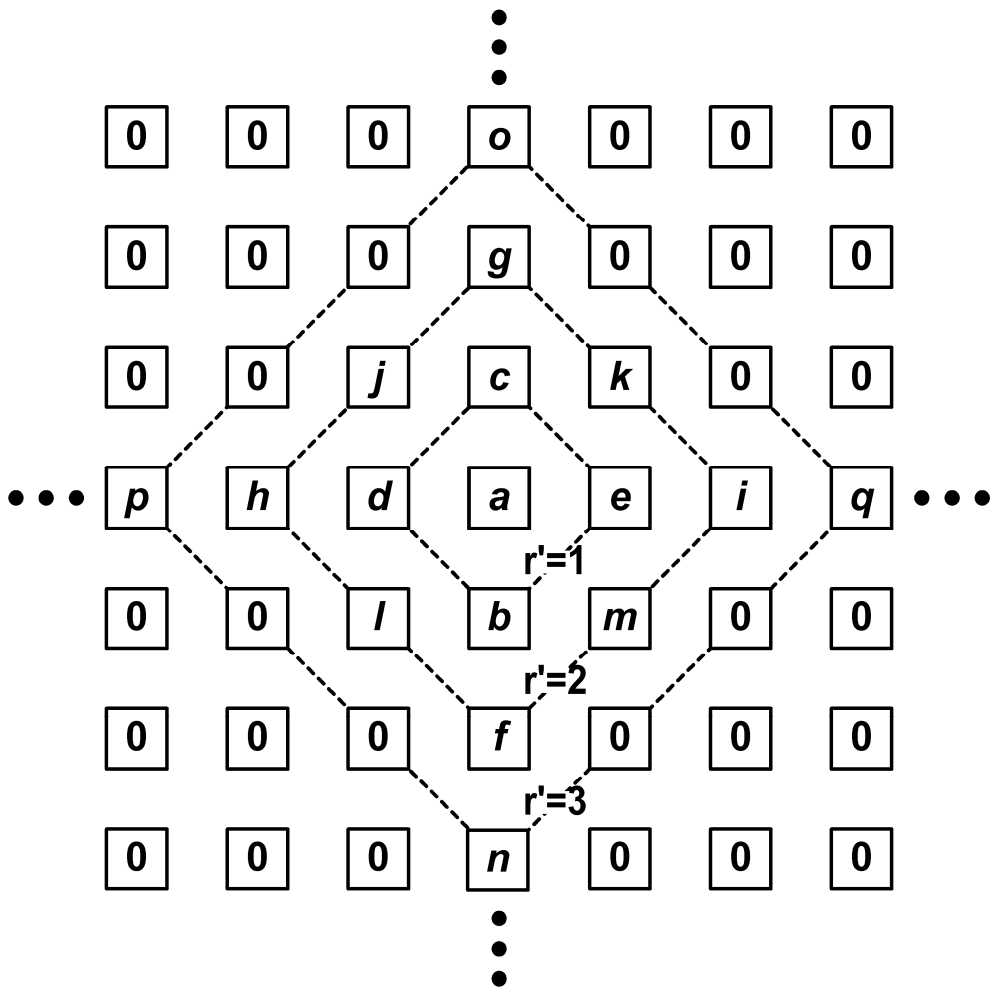


Fig. 2.3 The large-neighborhood template generated by a LNCNN with propagating connections.

Fig. 2.3 shows a large-neighborhood template where symbols from the letters a to q represent the template coefficients and the coefficients from a to m can be defined by the proposed LNCNN. The neighborhood of radius r' is redefined as shown in Fig. 2.3. Here, the sphere of influence S_{ij} of a large neighborhood is not considered as a 5×5 matrix, but is defined as a diamond-shaped matrix in Fig. 2.3 with neighborhood of radius $r'=2$. Each coefficient can be derived from the gains of the synapses in Fig. 2.1 and the PS in Fig. 2.2. The derived equations are listed in Table 2.1 where the template coefficients in Fig. 2.3 are expressed by the gains of the synapses and the gain of each synapse is expressed by the template coefficients. Thus, the architecture in Fig. 2.1 and Fig. 2.2 can be used to generate the large-neighborhood

Table 2.1 DERIVED EQUATIONS OF TEMPLATE COEFFICIENTS AND GAINS OF SYNAPSES

Connection Type	Template Coefficients Constructed by the Gains of Synapses	Gain of Each Synapse by Template Coefficients
Direct Connection	$a = PS,$ $b = PU1$ $c = PD1,$ $d = PR1$ $e = PL1,$ $j = PRD$ $k = PLD,$ $l = PRU$ $m = PLU$	$PS = a,$ $PU1 = b$ $PD1 = c,$ $PR1 = d$ $PL1 = e,$ $PRD = j$ $PLD = k,$ $PRU = l$ $PLU = m$
Propagating Connection	$f = PU1 \times PU2$ $g = PD1 \times PD2$ $h = PR1 \times PR2$ $i = PL1 \times PL2$ $n = PR1 \times PR2^2$ $o = PU1 \times PU2^2$ $p = PL1 \times PL2^2$ $q = PD1 \times PD2^2$	$PR2 = \frac{h}{d} < 1$ $PL2 = \frac{i}{e} < 1$ $PU2 = \frac{f}{b} < 1$ $PD2 = \frac{g}{c} < 1$

templates with $r' = 2$ shown in Fig. 2.3.

According to Table 2.1, the gains of the synapses PD2, PU2, PL2 and PR2 of propagating connections should be less than 1 for each. If the synaptic gain of a propagating connection is larger than or equal to 1, then the signal coming from the cells along one direction would diverge. The gains of these synapses of propagating connections can be determined from the template coefficients f , g , h , and i as listed in Table 2.1. Because of the propagating connections, if the template coefficients f , g , h , and i are not equal to zero, the coefficients o , q , n and p would not equal zero also, respectively. However, if the template coefficients n , o , q , and p are to be set zero, the template values f , g , h and i would be small enough when compared with the template values b , c , d and e , respectively.

The four corner coefficients j , k , l , and m are determined directly by the synapses PRD, PLD, PRU, and PLU, respectively, of direct connections. Similarly, the coefficient a can be generated directly by the PS in Fig. 2.2.

2.3 CIRCUIT IMPLEMENTATION AND SIMULATION

RESULTS

It has already been established that the current-mode signals can be easily combined. In addition, current-mode circuits are faster and consume less power than voltage-mode circuits. However, when the current signals need to be duplicated, more devices are required to mirror the currents. In the design, the currents in fewer paths need to be duplicated. Therefore, the proposed LNCNN has been implemented by using current-mode circuits. In all the current-mode circuit realizations, the signals represented in Fig. 2.1 and Fig. 2.2 transferred

inside the kernel unit are all in current mode except the DCS, CLK, synaptic gain controlling signals, and the digital logic circuits signals.

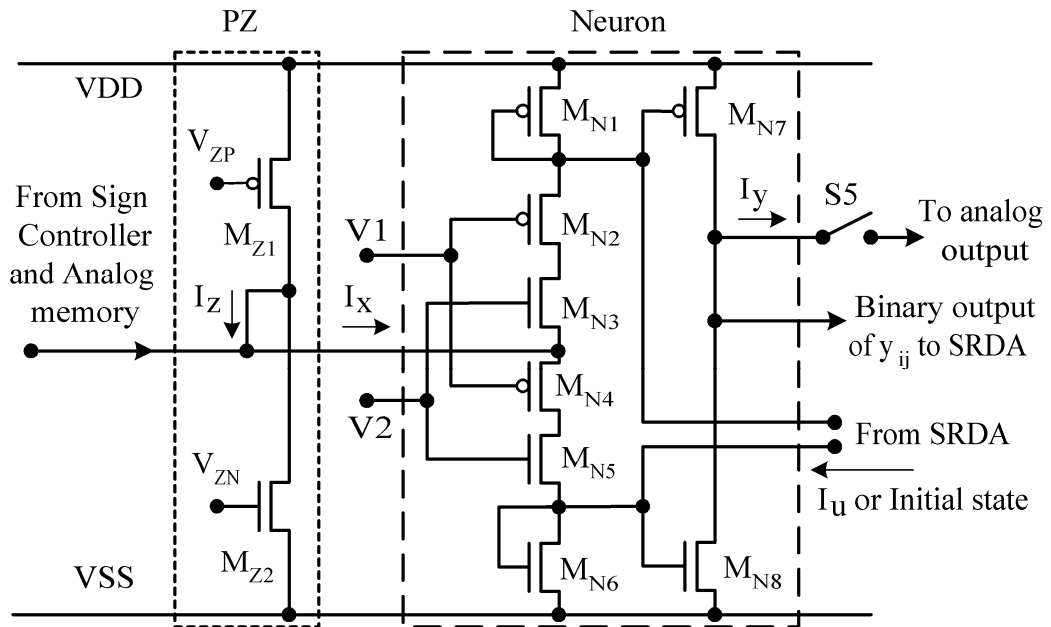


Fig. 2.4 The circuit diagram of the Neuron and PZ in Fig. 2.2.

A. Neurons and PZ

Fig. 2.4 depicts the circuit of the PZ and the Neuron inside the BODY as indicated by dotted lines in Fig. 2.2. The PZ is implemented by the devices M_{Z1} and M_{Z2} . The gate bias voltages V_{ZP} and V_{ZN} directly control the current through M_{Z1} and M_{Z2} , respectively, to generate the threshold current I_Z . The circuitry of M_{N1} - M_{N6} is the neuron core with the piecewise linear ramp function. The gate bias voltages $V1$ and $V2$ are used not only to maintain the static current of the neuron zero with the devices M_{N4} and M_{N3} , respectively, but they are also used to limit the currents through M_{N1} with M_{N2} and M_{N6} with M_{N5} , respectively. Furthermore, M_{N3} and M_{N4} also act as the switch S1 in Fig. 2.2. The gate bias voltages $V1$ and $V2$ are controlled by the external bias current I_{bias} . The transfer characteristic of the neuron is simulated as shown in Fig. 2.5. The low and high limit currents of the piecewise linear ramp function range from 351.8

nA to 487.8 nA and from 389.5 nA to 534.3 nA, respectively, when the external bias current I_{bias} is in the range from 250 nA to 360 nA and the supply voltage is 1.8 V. When the neuron is on standby or there is no input current, the leakage current is less than 1nA. In the first and second steps of the operation period, S1 is turned off; that is, M_{N2} - M_{N5} are turned off. In this way, the neuron core acts as two current mirrors. As the input current I_u , shown in Fig. 2.4, is provided by the SRDA in the first step, the current I_{Xm} is calculated and in the second step the initial value $I_x(t=0)$ is also introduced by the SRDA. Moreover, M_{N7} and M_{N8} are used to send the binary outputs to the SRDA or to send the transient currents to the analog outputs through S5.

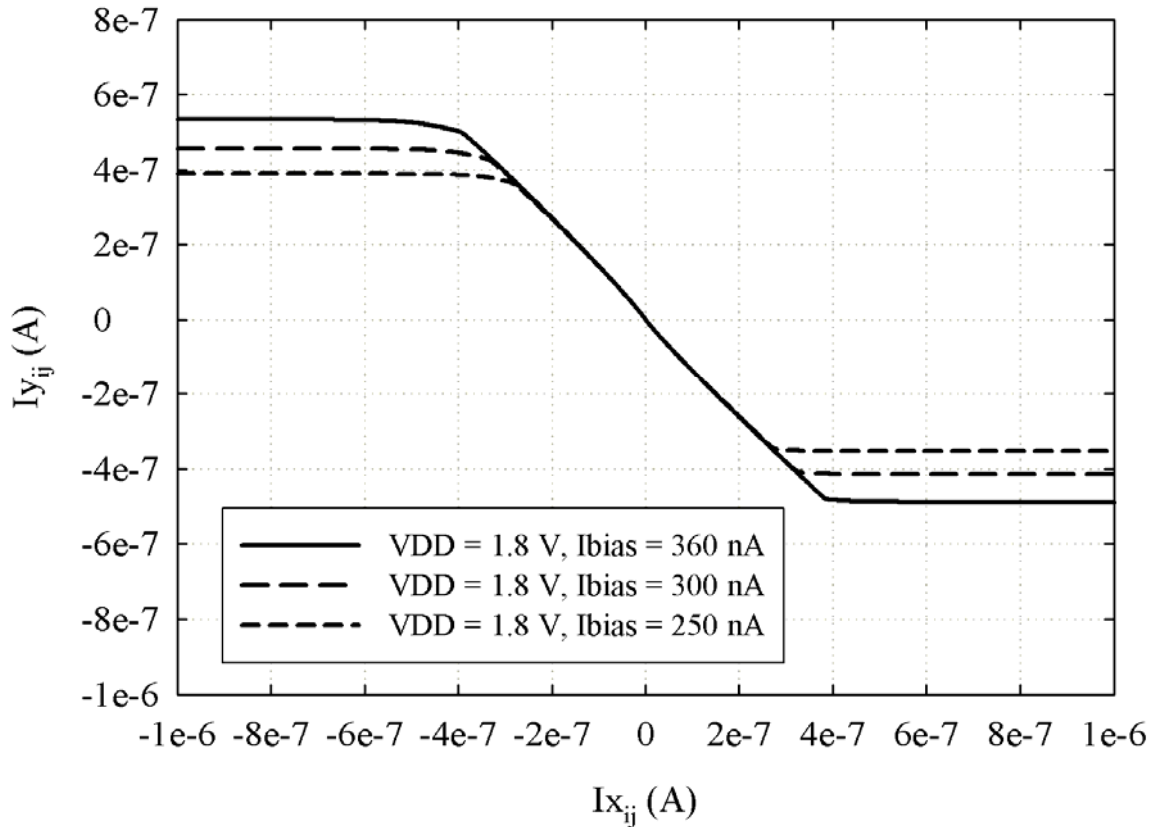
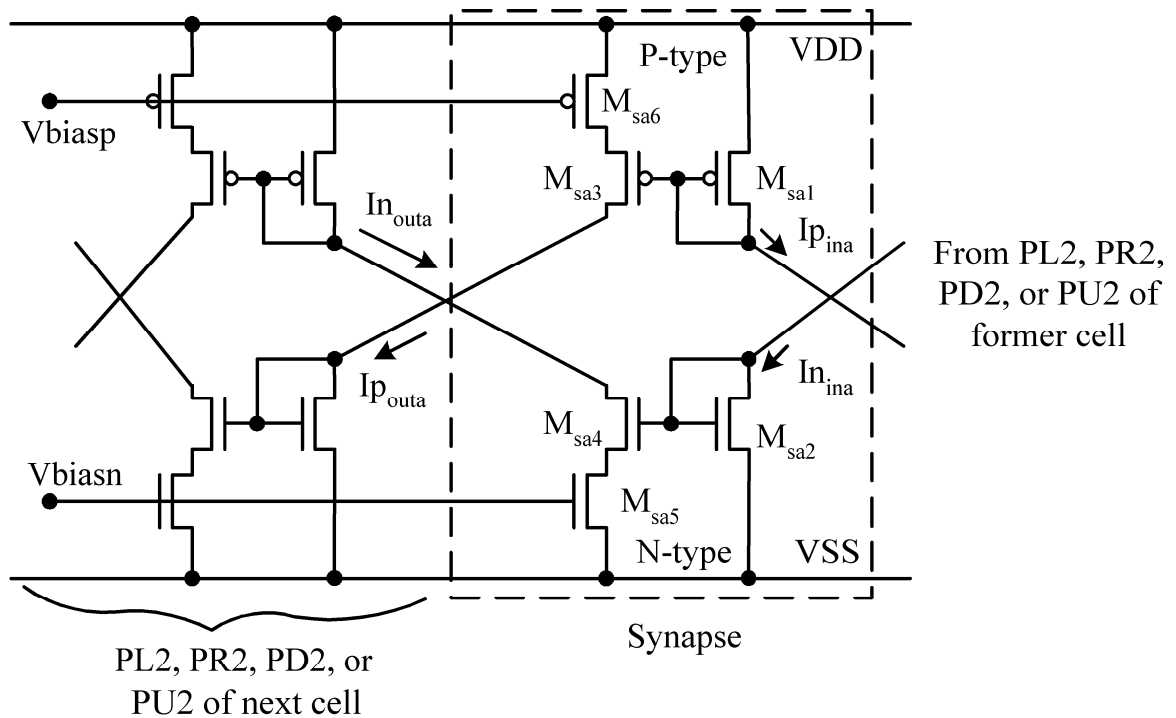


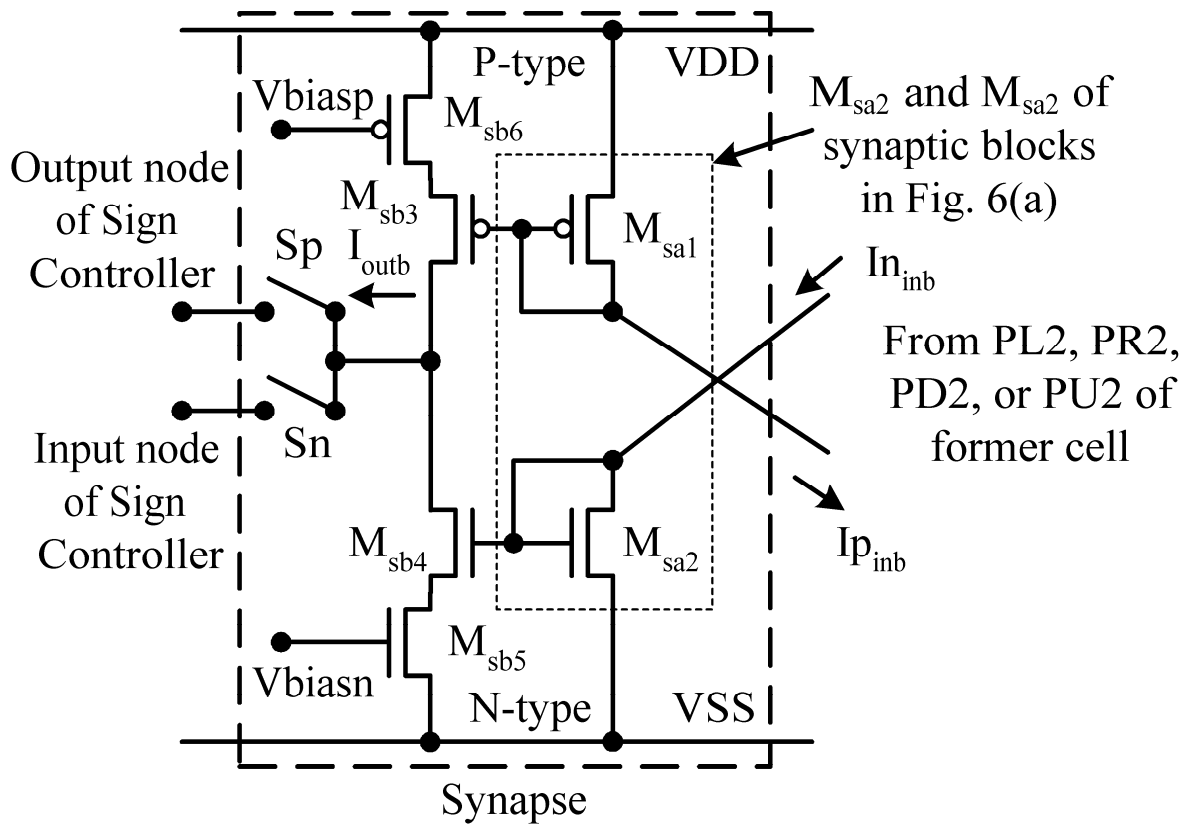
Fig. 2.5 The transfer characteristic of a neuron with different external bias currents I_{bias} .

B. Synapses and Sign Controller

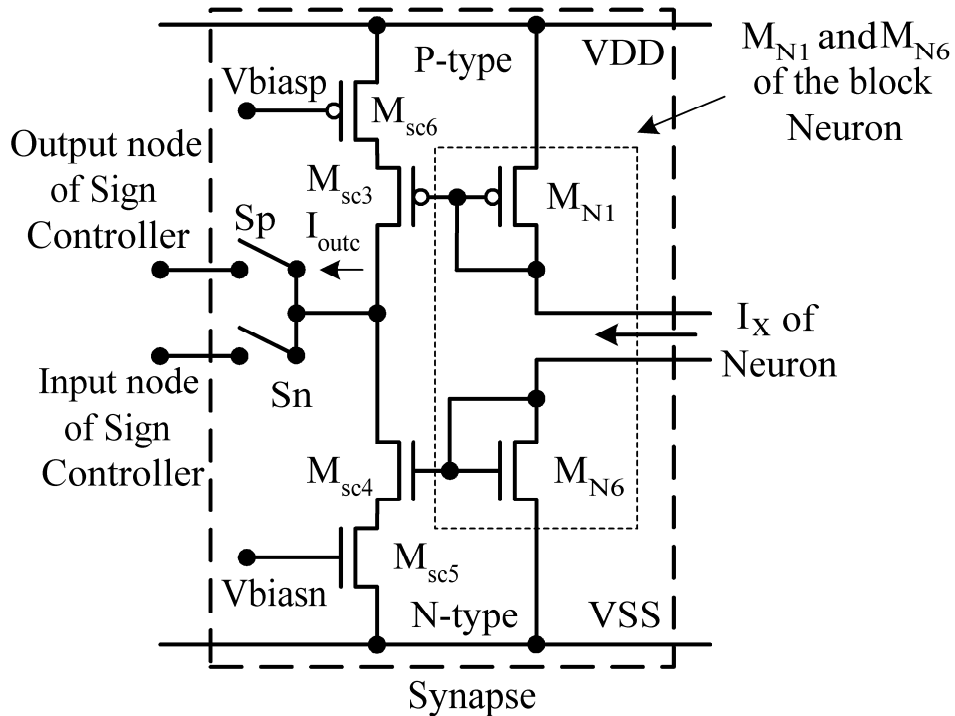
The circuit diagrams of the synapses are shown in Fig. 2.6(a)-(c) and are indicated by broken lines, whereas the circuit diagram of the Sign Controller is demonstrated by broken lines in Fig. 2.6(d). The circuit of Fig. 2.6(a) is used to realize the synapses PL2, PR2, PD2, and PR2 of propagating connections. There are two paths, N-type and P-type, in one synapse to deal with the bi-directional current inputs. If a LNCNN is on standby or there are no input currents, the synapses consume no power. The device pairs M_{sa1}/M_{sa3} and M_{sa2}/M_{sa4} can be seen as two sets of current mirrors and the maximum gains are determined by the ratios of M_{sa1}/M_{sa3} and M_{sa2}/M_{sa4} . M_{sa6} and M_{sa5} with gate bias voltages V_{biasp} and V_{biasn} are operated in the linear region to control the current mirror gains of M_{sa1}/M_{sa3} and M_{sa2}/M_{sa4} , respectively. All the gate bias voltages V_{biasp} and V_{biasn} of synapses combined with the gate bias voltages V_{ZP} and V_{ZN} of the PZ form the synaptic gain controlling signals as shown in Fig. 2.1. Furthermore, the gate bias voltages V_{biasp} and V_{biasn} are generated by using an on-chip 4-bit DAC. There are 16 different values for V_{biasp} and V_{biasn} . A HSPICE simulated I_{nouta} vs. I_{ina} diagram of



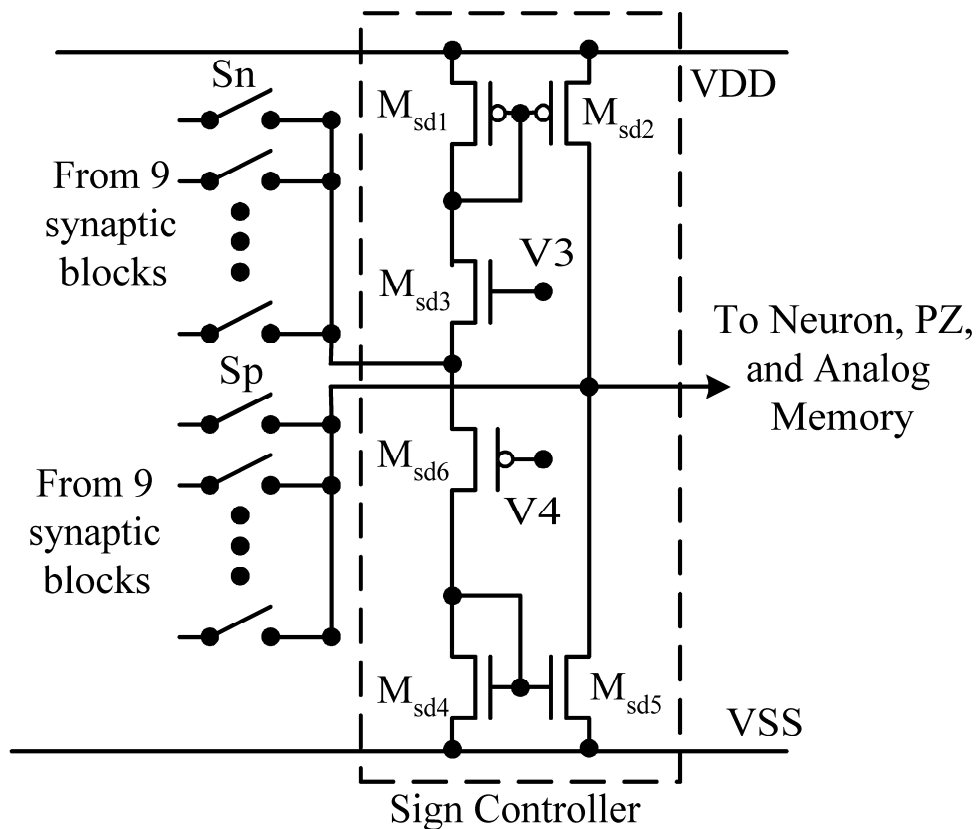
(a)



(b)



(c)



(d)

Fig. 2.6 The circuit diagrams of (a) the synapses PL2, PR2, PD2, and PU2; (b) the synapses PL1, PR1, PD1, and PU1; (c) the synapses PRU, PRD, PLU, PLD, and PS; (d) the Sign Controller.

the N-type synapse with differing gate bias voltages V_{biasn} ranging from 34.4 mV to 737 mV is shown in Fig. 2.7. The corresponding N-type and P-type current gains of the input current ranging from 300 nA to 500 nA are illustrated in Fig. 2.8, where M_{sa1} - M_{sa4} are operated in the subthreshold region with a supply voltage of 1.8V. The N-type synaptic gains with different V_{biasn} values ranges from 0 to 1.54 in the input current range from 300 nA to 500 nA while the P-type synaptic gains with different V_{biasp} values ranges from 0 to 1.42. The N-type synaptic gain has an average variation of $\pm 6.38\%$ and the P-type synaptic gain has that of $\pm 7.72\%$, as indicated by short bars over the input current range from 300 nA to 500 nA. It can be seen that the synapses can generate the desired templates with a tolerable level of error by setting the codes for the V_{biasn} and V_{biasp} voltages with proper values.

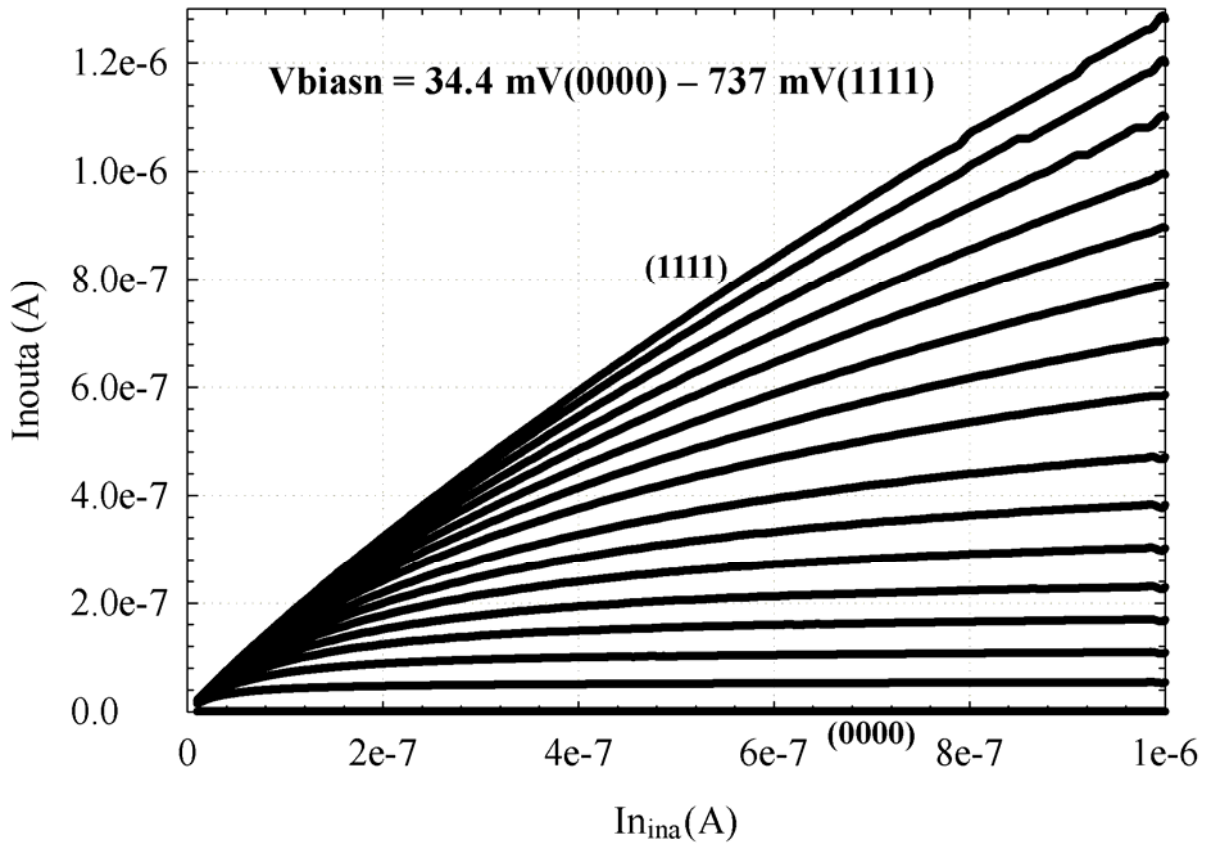
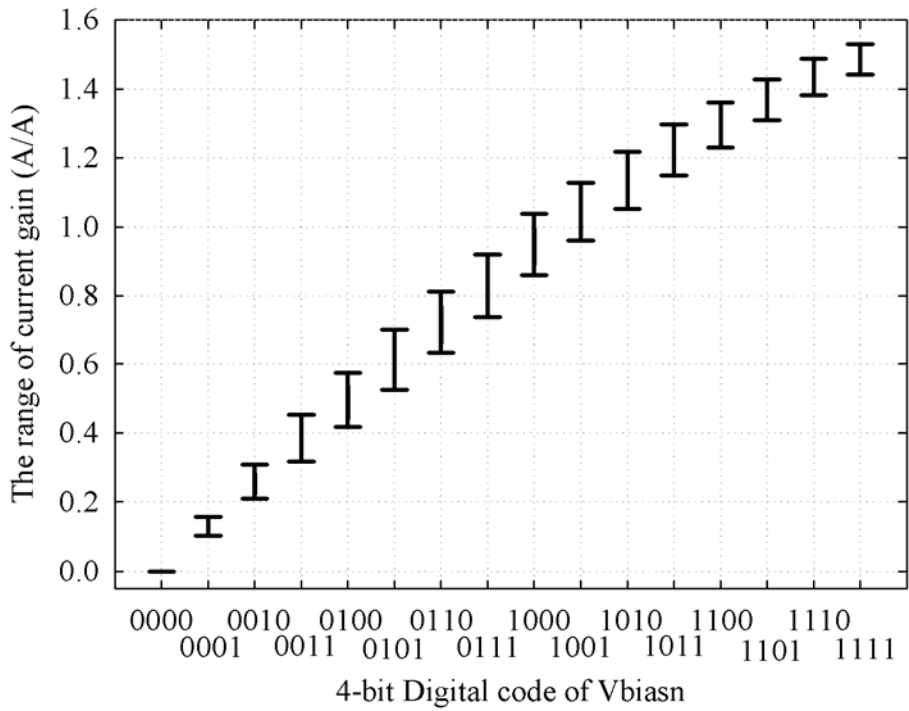
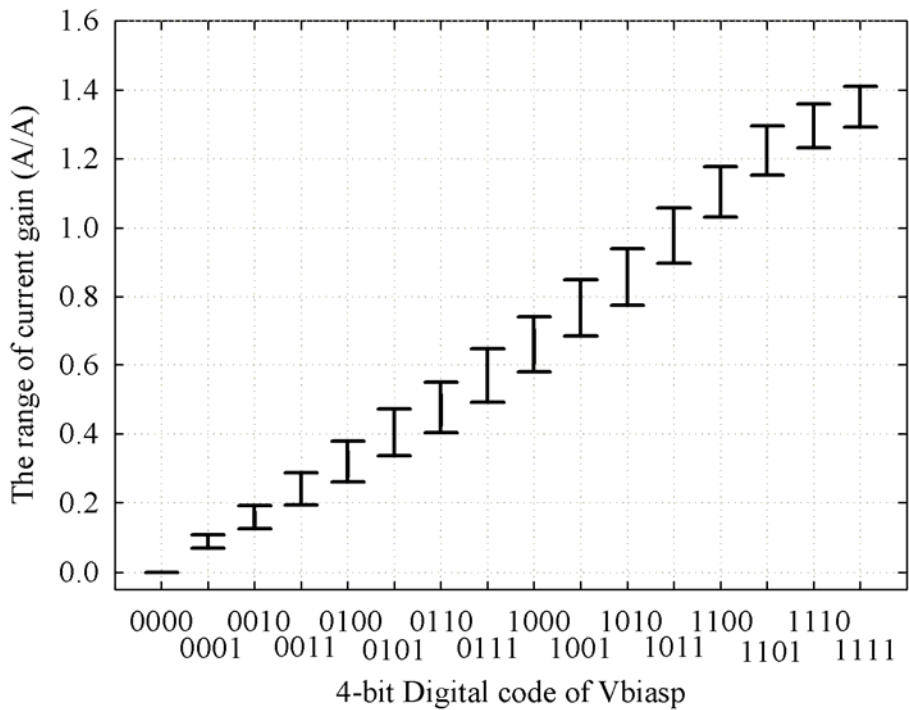


Fig. 2.7 The HSPICE simulated $I_{n_{outa}}$ vs. $I_{n_{ina}}$ diagram of the N-type synapse in Fig. 2.6(a) with 16 different values for V_{biasn} .

The circuits of the synapses of direct connections are shown in Fig. 2.6(b) and Fig. 2.6(c) and it can be seen that the circuits and operations are similar to those of the synapses of propagating connections. The circuit in Fig. 2.6(b) realizes the synapses PL1, PR1, PD1, and PU1 while that in Fig. 2.6(c) realizes PLU, PLD, PRU, PRD, and PS. The P-type and N-type synaptic gains of one synapse of direct connections can be set to different values to perform more functions. The synapses shown in Fig. 2.6(b) share the two master devices M_{sa1}/M_{sa2} with the synapses of the propagating connections while those shown in Fig. 2.6(c) share M_{N1}/M_{N6} with the Neuron. The output currents of Fig. 2.6(b) and Fig. 2.6(c) are sent to the Sign Controller using the switches S_n and S_p to decide the polarities of the signals. The maximum gains of the synapses PLU, PLD, PRU, and PRD are set to 2 and those of PL1, PR1, PU1, and



(a)



(b)

Fig. 2.8 The range of (a) the N-type current gains and (b) the P-type current gain of the synapses with an input current range from 300 nA to 500 nA.

PD1 are set to 4 whereas the gain of the synapse PS is set to 8. Through this design, this LNCNN can generate the templates as indicated in Fig. 2.3 where the center coefficient a is smaller than 8 and the coefficients $b, c, d,$ and e are smaller than 4, while the coefficients $j, k, l,$ and m are smaller than 2. The circuitry of the Sign Controller is shown in Fig. 2.6(d) where the switches S_n and S_p of the 9 synapses used to adjust the polarity of the signals from the synapses are also drawn. The devices M_{sd3} and M_{sd6} with gate bias voltages $V3$ and $V4$, respectively, maintain the static current from M_{sd1} to M_{sd4} at zero level. M_{sd1}/M_{sd2} and M_{sd4}/M_{sd5} are the current mirrors used to invert the direction of the current flow. If the polarity of the input signal from synapses is negative, the S_p is turned off and the input signal enters the neuron or analog memory through the switch S_n and Sign Controller. However, in the same situation if the input signal is positive, the S_n is turned off and the signal enters the neuron through the S_p switch.

C. PSWs

Each of the synapses contains one pair of switches S_n and S_p to control the signal polarities except the synapses of propagating connections. Hence, to confirm the output signals sent out of the BODY and those sent out of the synapses of propagating connections have the same polarities, the PSW has been added to achieve this purpose.

Fig. 2.9 depicts the circuit diagram of the PSW. The output currents of the neuron are mirrored through M_{sw1} and M_{sw4} to generate the gate voltages on M_{sw2} and M_{sw3} , respectively. The current through M_{sw5} , where the gate is connected to the gate of M_{N1} (M_{sw2}), is opposite to the current through M_{sw6} , whose gate is connected to the gate of M_{sw3} (M_{N6}). The polarity of the output current in the PSW is selected using the switches S_{sw1} - S_{sw4} . For a positive (negative) output of the PSW, the switches S_{sw1} and S_{sw4} (S_{sw2} and S_{sw3}) are closed and, at the same time, the switches S_{sw2} and S_{sw3} (S_{sw1} and S_{sw4}) are opened. There are four PSWs containing the

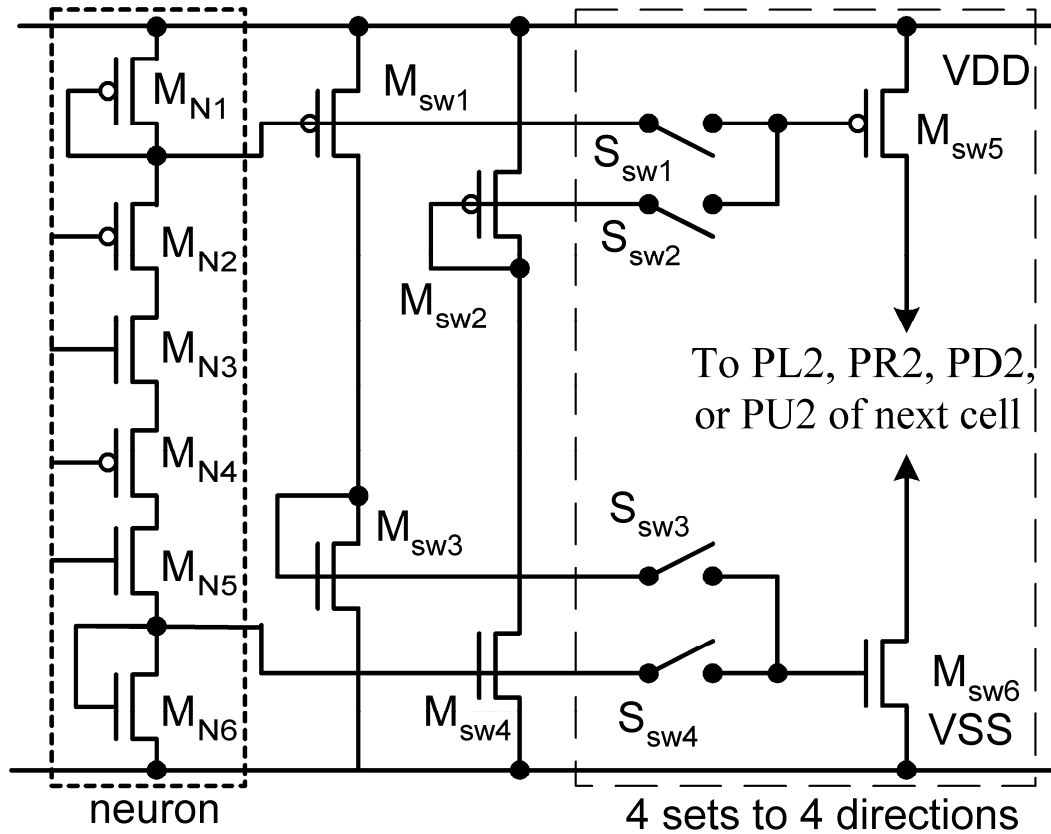


Fig. 2.9 The circuit diagram of the PSW.

switches S_{sw1} - S_{sw4} and M_{sw5} - M_{sw6} as drawn in Fig. 2.1 and these four PSWs share the circuits of M_{sw1} - M_{sw4} .

A comparison of the device numbers and interconnection lines of the kernel unit between the proposed structure and the LNCNN with direct connection using the circuit structure in [126] is given in Table 2.2. As can be seen from Table 2.2, the LNCNN with direct connections needs 12 connections, including 4 connections to the farther neighboring cells. In the proposed structure, more devices are required; however, as each cell only has 8 connections to the nearest eight neighboring cells, this facilitates the IC implementation.

D. Analog Memory

Fig. 2.10 depicts the circuit diagram of the analog memory where M_{M1} and M_{M9} are used to generate the gate voltages of M_{M6} and M_{M14} , respectively, from the input current I_{Xm} . The gate

Table 2.2 COMPARISON OF DEVICE NUMBERS AND INTERCONNECTION LINES

		This Work with Propagating Connection	LN-CNN ($r' = 2$) with Direct Connection Using the Circuit Structure in [9]
Device Number	Synapse of Direct Connections	8.67	8
	Synapse of Propagating Connections	13	-
	Neuron Core	8	10
	Neuron Cell	130	104
Interconnection Lines cross One Cell		8	12

voltages are stored at the node A (B) by turning off M_{M11} (M_{M3}) with the signal V_{sample} (with the complementary signal of V_{sample}). After sampling, the signal V_{enable} rises to high when the current is accessed. M_{M4} and M_{M12} are used to compensate for the charge injections and the clock feedthrough from M_{M3} and M_{M11} , respectively. The device size of M_{M4} and M_{M12} is half of M_{M3} and M_{M11} . M_{M5} and M_{M13} are used to increase the gate-source capacitance C_{gs} of M_{M6} and M_{M14} , respectively, in order to suppress the sampling error. The current mirror M_{M7}/M_{M8} (M_{M15}/M_{M16}) is used to isolate the storage node A (B) from the output node of analog memory so that the stored voltage is not affected by the voltage change at the output node. As the analog memory is read out, the signal V_{enable} (the complementary signal of V_{enable}) turns on M_{M17} (M_{M18}) and, at the same time, it also turns on the compensational function of M_{M4} (M_{M12}). Furthermore, the devices M_{M2} and M_{M10} with gate bias voltages V_6 and V_5 , respectively, maintain the static current from M_{M9} to M_{M1} at zero level and also, act as the switch S_2 , as can be seen in Fig. 2.2.

E. Overall Chip Architecture

Fig. 2.11 shows the architecture of the whole system where the size of the kernel unit array

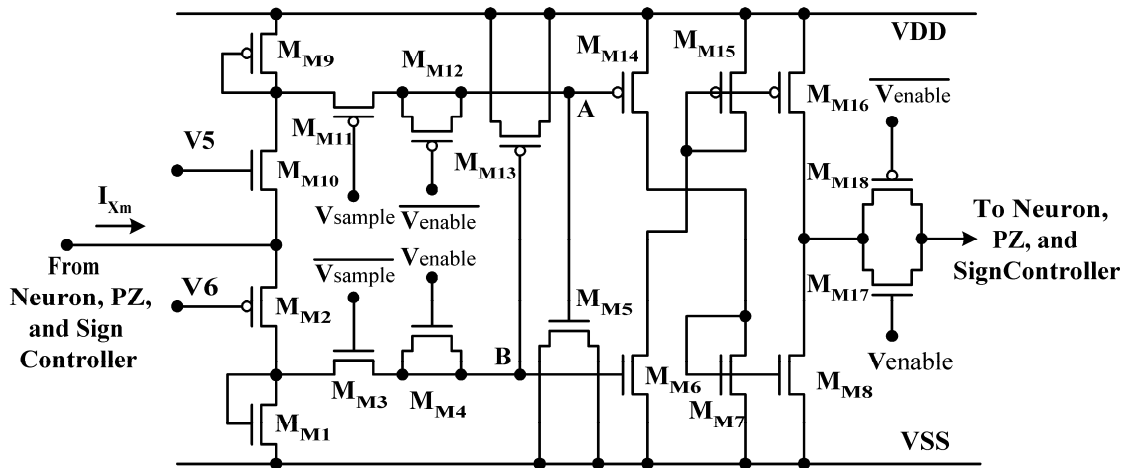


Fig. 2.10 The circuit diagram of the analog memory.

is 20×20 . There are 5×54 shift registers to store the digital codes of synaptic gain controlling signals. There is no additional signal to reset the shift registers. Shift registers can be refreshed by input signals. The digital codes of each synaptic gain controlling signal are stored in a 4-bit shift register for absolute value and a 1-bit shift register for polarity. However, one synapse requires two synaptic gain controlling signals and the signals have different values when templates **A** and **B** are generated. Hence, there are 5×52 shift registers required for templates **A** and **B**. For the synaptic gain controlling signals of template **Z**, a 6-bit register is required for the absolute value of template **Z** and a 1-bit shift register is used for its polarity. Thus, 5×2 1-bit shift registers are required for template **Z**. The signal from the Digital Controlling Circuit determines whether Generation Circuit for templates **A**, **B** and **Z**, which has 28 DACs, generates synaptic gain controlling signals for either template **A** or for templates **B** and **Z**. The external bias current I_{bias} generates the bias currents and voltages required in the system, especially the bias voltages V1-V6 inside the Neuron, Sign Controller, and Analog Memory as shown in Fig. 2.1. The signals *Input_Enable* and *Weight_Enable* with external clock signal Ext_CLK are used to determine whether the external input signals are input and initial patterns

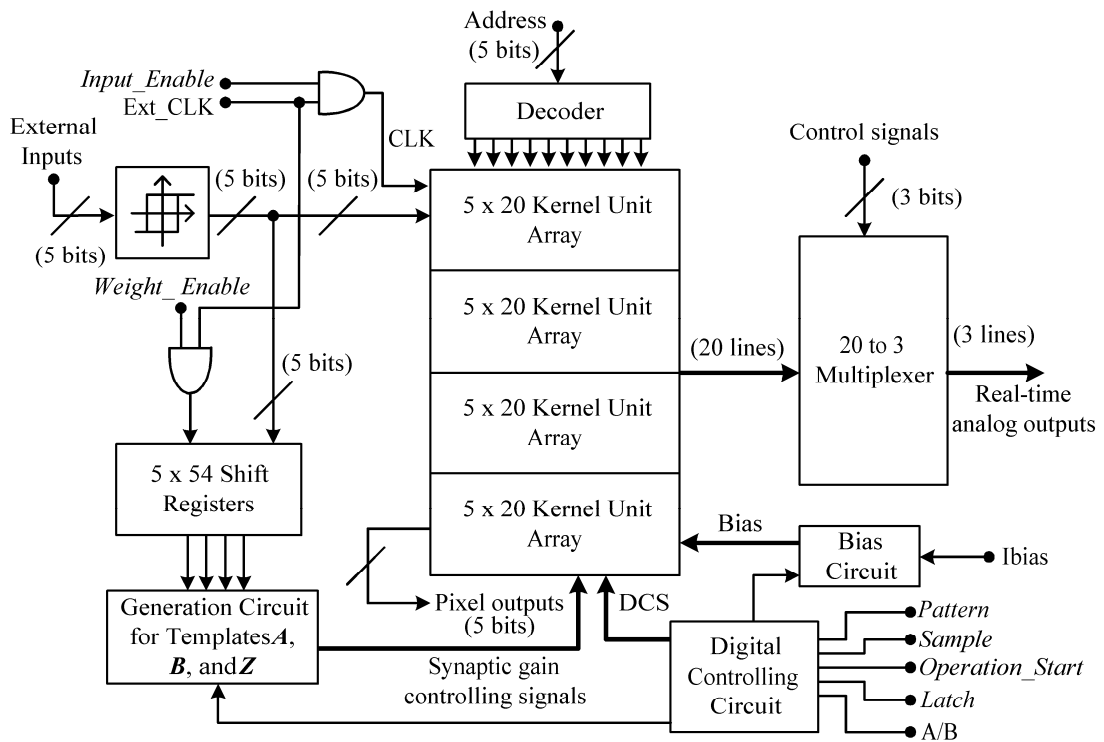


Fig. 2.11 The architecture of the 20x20 LNCNN system.

or the digital codes of synaptic gain controlling signals, respectively. In the array, 5-bit binary signals in one clock cycle are sent into the LNCNN and read out from the Pixel outputs. From 20 neuron analog output signals of one column selected by a 5-bit decoder, 3 real-time neuron analog output signals can be read out using a 20-to-3 multiplexer.

The timing diagram is shown in Fig. 2.12. In the first step, both input pattern and digital codes of the templates *A*, *B*, and *Z* are ready for operation, and the signal *A/B* is set to High first to cause the template generate circuit to generate synaptic gain controlling signals of the templates *B* and *Z*. Meanwhile, the function of the neuron in the kernel unit is turned off. The signal *Pattern* goes to High in order to inject the input pattern into all the neurons. The result of the first step is sent into the analog memory and stored after the signal *Sample* is enabled and then the signal *Pattern* returns to Low.

In the second step, the pattern in the shift register of the SRDA is replaced by the initial

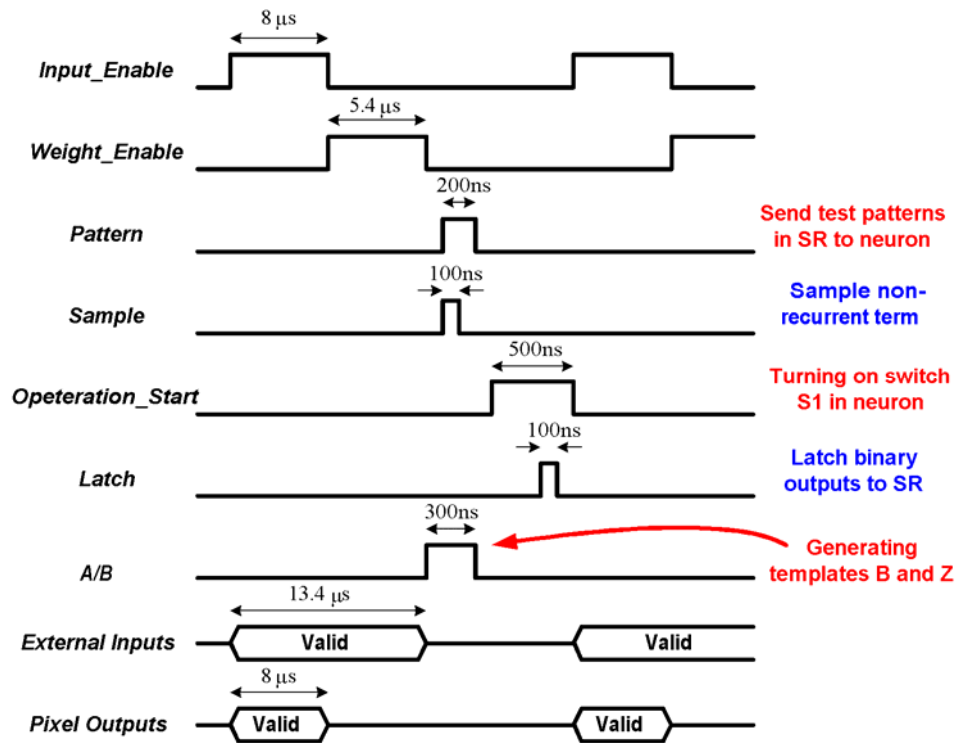


Fig. 2.12 The timing diagram of the controlled signals in LNCNN

pattern of the desired function. The initial pattern in the shift register is then sent to the neurons as the initial values by enabling the signal *Pattern* again. Meanwhile, the signal *A/B* is set to Low so that the template *A* is generated by the synapses as the template generation circuit generates the synaptic gain controlling signals of the templates *A*. In the third step, the signal *Operation_Start* is enabled and the signal *Pattern* is disabled to turn off the initial values. The function of the neurons is turned on to start the overall calculation of template *A* with the signals read out from the analog memories. After the outputs are stable, the binary output pattern can be stored in the SRDA as the signals *Latch* and *Input_Enable* are set to High. When the next input pattern comes in, all the digital signals are disabled except the signal *Input_Enable* and the output pattern can be read out from the 5-bit Pixel outputs.

F. Hspice Simulation Results

The proposed LNCNN circuit was designed using CMOS 0.18- μm technology. The HSPICE post-layout simulation was performed with a 20×20 kernel cell array to verify the circuit functions. The function of Muller-Lyer illusion with the 5×5 large-neighborhood template [40], as shown in Fig. 2.13 was adopted. According to the original 5×5 template, the predicted signs of each diamond-shaped template are set in Fig. 2.14. Only the center coefficients A_{ij} and B_{ij} are positive and the others are negative in Fig. 2.13, so it is reasonable that only the center coefficients in the diamond-shaped template are set to positive. The input pattern of Muller-Lyer illusion is shown in Fig. 2.15(a). After the HSPICE simulation, the resultant output pattern is shown in Fig. 2.15(b), where the upper line with outward arrows becomes shorter than the lower line with inward arrows after illusion. The function cannot be realized by a 3×3 neighborhood template. The coefficients of the diamond-shaped, large-neighborhood template in Fig. 2.3 were extracted from the post-layout simulation results directly and are shown in Fig. 2.14, which has the same signs as those in Fig. 2.13.

The simulated standby power consumption is about 1.148 mW where a 1.8 V supply voltage and a system clock frequency of 20 MHz are used. The external bias current is 360 nA. The kernel unit array only consumes 1 μW , which accounts for about 0.087% of overall

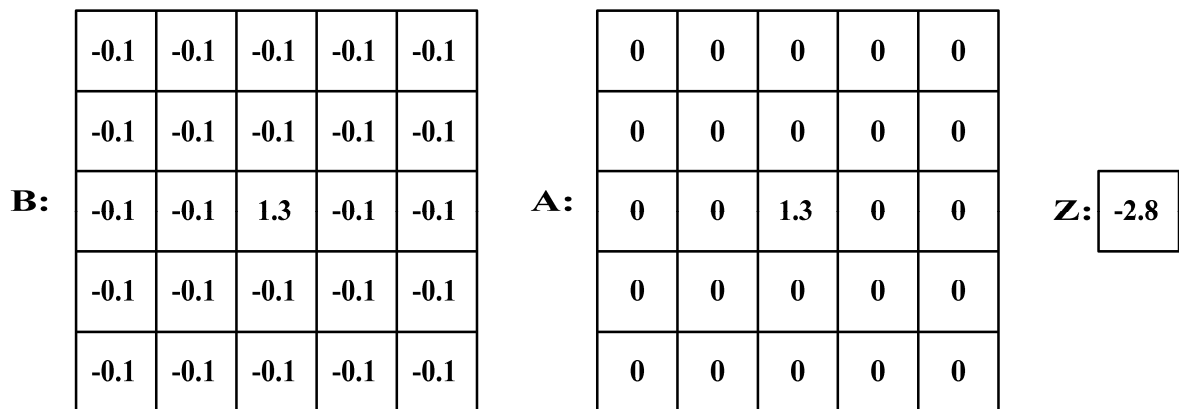


Fig. 2.13 The 5×5 templates B, A and Z for Muller-Lyer illusion [40].

	0	0	0	-0.13	0	0	0
	0	0	0	-0.26	0	0	0
	0	0	-1	-0.5	-1	0	0
B:	-0.13	-0.26	-0.5	+2.73	-0.5	-0.25	-0.13
	0	0	-1	-0.5	-1	0	0
	0	0	0	-0.25	0	0	0
	0	0	0	-0.13	0	0	0

	0	0	0	0	0	0	0
	0	0	0	0	0	0	0
	0	0	0	0	0	0	0
A:	0	0	0	+2.96	0	0	0
	0	0	0	0	0	0	0
	0	0	0	0	0	0	0
	0	0	0	0	0	0	0

Z:	-7.9
-----------	------

Fig. 2.14 The extracted values of the diamond-shaped template from a HSPICE post-layout simulation.

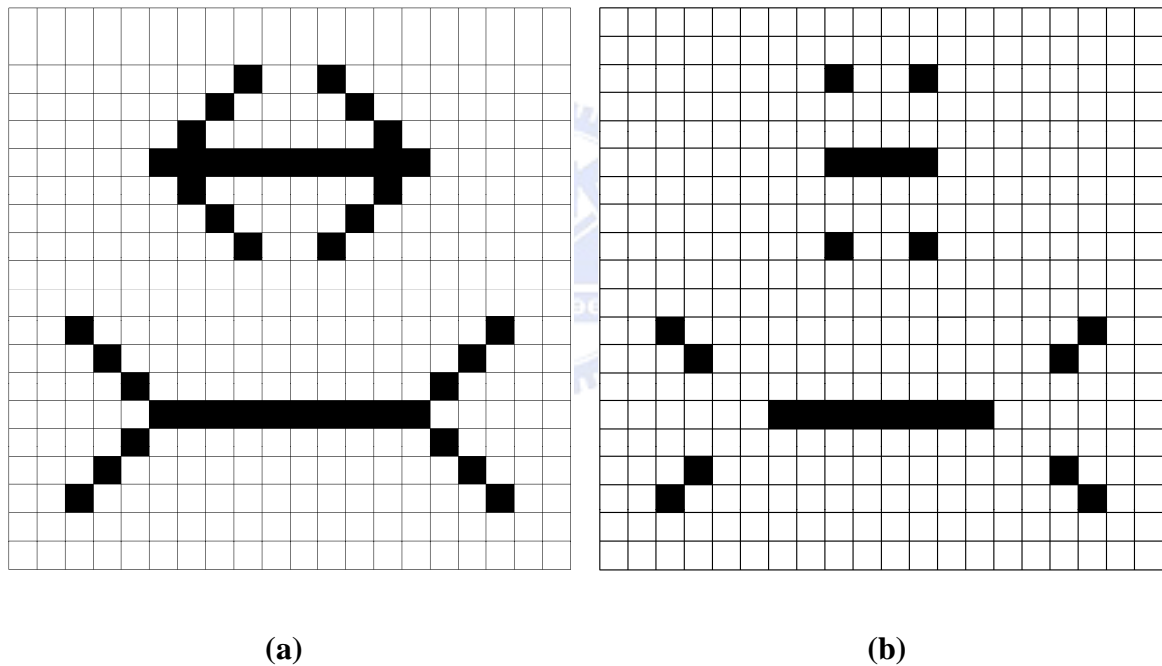


Fig. 2.15 (a) The input patterns of Muller-Lyer illusion. (b) The resultant output pattern of Muller-Lyer illusion from the HSPICE simulation result.

standby power consumption. As the array is extended to 128×128 , the standby power consumption is about 7.35 mW and is dominated by the peripheral circuits.

G. Software Simulations Results-CNN Visual Mouse Platform [141]

The published large-neighborhood templates are limited. Among the four published LNCNN templates [40], [118], [138]. Only one template [118] cannot be implemented by using

the proposed structure since it violates the constraint. Other LN templates for diffusion, de-blurring, and Muller-Lyer illusion have been successfully verified. The 5×5 templates B of diffusion [138] and de-blurring [40] are approximated by the proposed diamond-shaped templates as shown in Fig. 2.16(a) and Fig. 2.16(b). The coefficients of templates A and Z are 0 for diffusion. For de-blurring, the center coefficient of template A of 5×5 template is 10 and that of diamond-shaped template is 7. Both templates Z are 0. The input pattern and simulation results of diffusion and de-blurring are shown in Fig. 2.17(a) and Fig. 2.17(b), respectively. It is shown that the diamond-shaped template can realized the function of 5×5 templates correctly. The diamond-shaped LN templates also can realize some operations of binary images in one step which can be realized by the 3×3 neighborhood templates in two steps. The erosion and dilation function with 3×3 neighborhood templates can contract and expand the edges of images by one pixel, respectively. However, the diamond-shaped LN templates can reinforce the functions to contract or expand the edges by two pixels. Fig. 2.18 demonstrates the function of erosion where the boundary cells are set to be white (-1). For dilation, it can be realized by the same templates of erosion by making template Z positive. Besides, these two functions with the diamond-shaped LN templates cannot be achieved with 3×3 neighborhood templates in one step. Two iterations with 3×3 neighborhood templates are required to realize the same functions. Thus, it takes more time and energy.

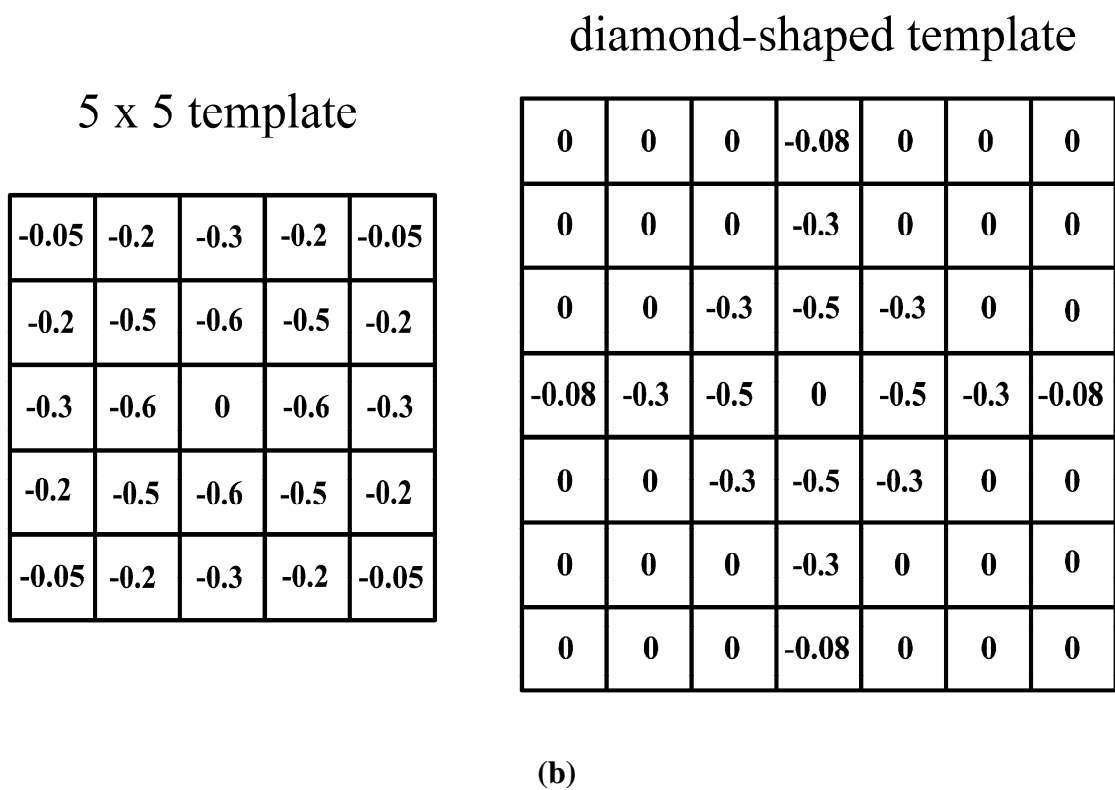
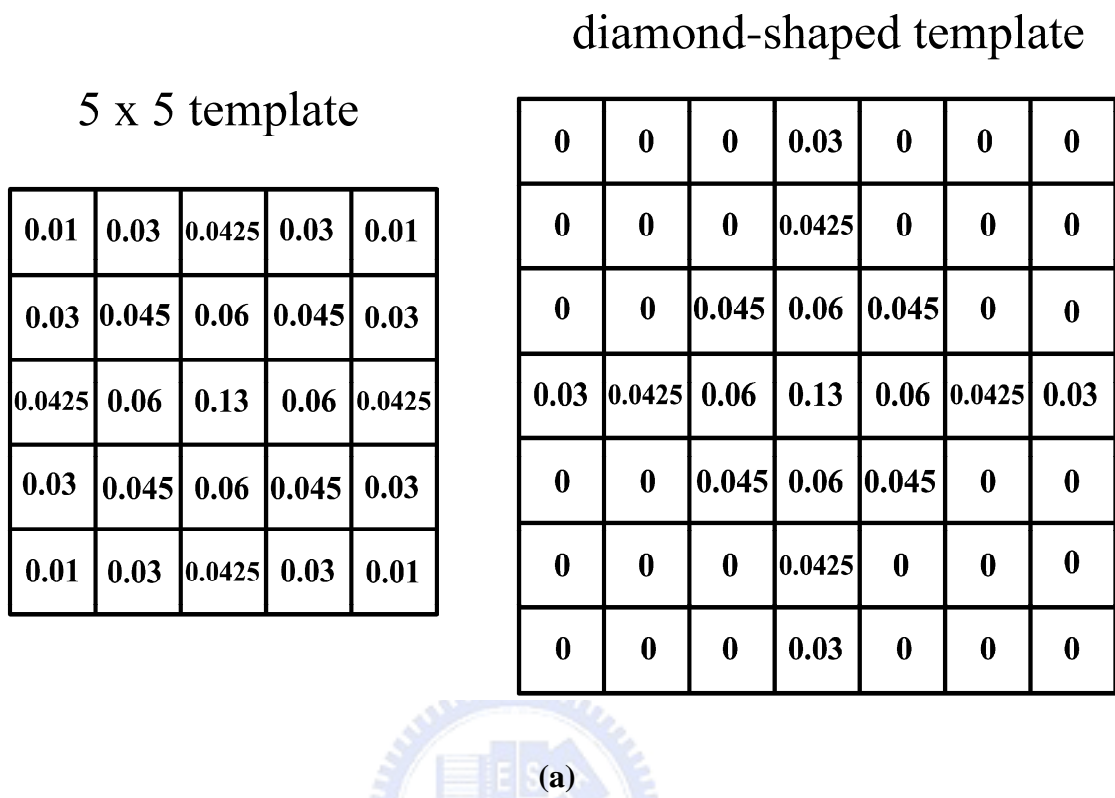


Fig. 2.16 The template B of 5x5 and diamond-shaped templates of (a) diffusion [138] and (b) de-blurring [40].

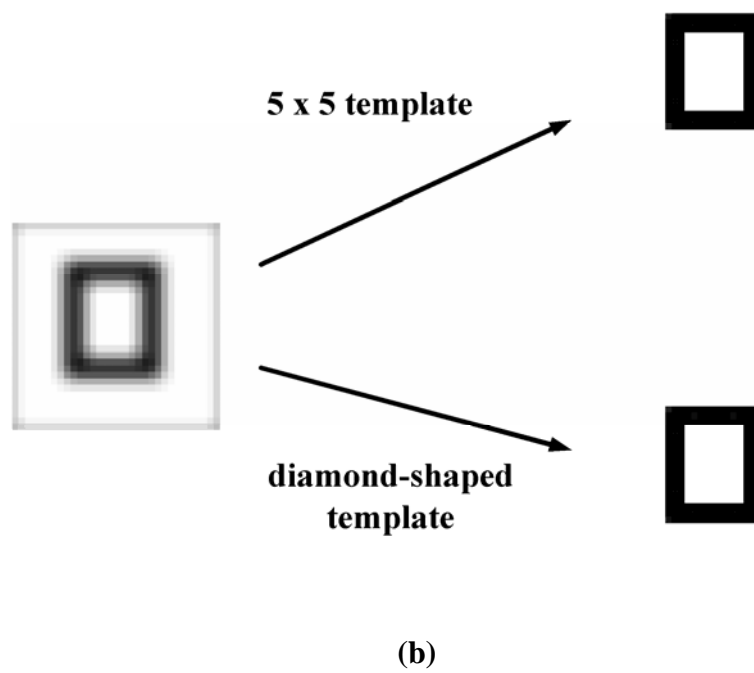
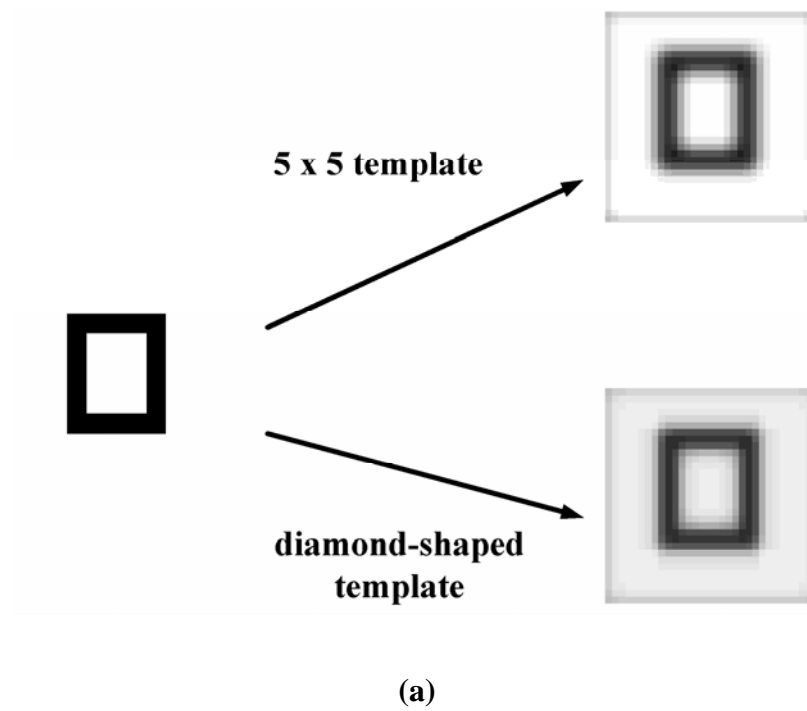
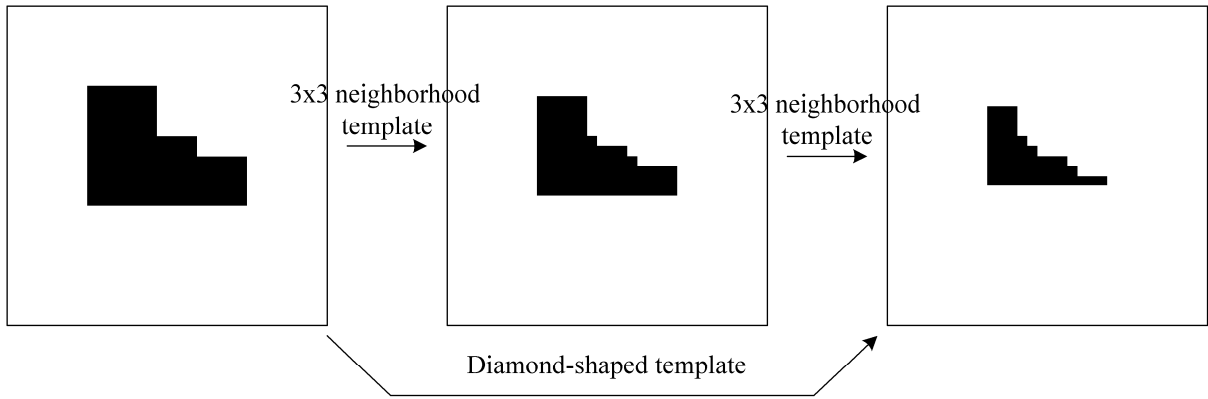


Fig. 2.17 The input patterns and simulation results of (a) diffusion and (b) de-blurring.



Functions	Erosion																																																																																																				
	B Template	A Template	Z																																																																																																		
3 x 3 Neighborhood Template	<table border="1"> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> </table>	0	1	0	1	1	1	0	1	0	<table border="1"> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>2</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>0</td></tr> </table>	0	0	0	0	2	0	0	0	0	-4.5																																																																																
0	1	0																																																																																																			
1	1	1																																																																																																			
0	1	0																																																																																																			
0	0	0																																																																																																			
0	2	0																																																																																																			
0	0	0																																																																																																			
Diamond Template	<table border="1"> <tr><td>0</td><td>0</td><td>0</td><td>0.13</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>0.5</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>1</td><td>2</td><td>1</td><td>0</td><td>0</td></tr> <tr><td>0.13</td><td>0.5</td><td>2</td><td>1</td><td>2</td><td>0.5</td><td>0.13</td></tr> <tr><td>0</td><td>0</td><td>1</td><td>2</td><td>1</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>0.5</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>0.13</td><td>0</td><td>0</td><td>0</td></tr> </table>	0	0	0	0.13	0	0	0	0	0	0	0.5	0	0	0	0	0	1	2	1	0	0	0.13	0.5	2	1	2	0.5	0.13	0	0	1	2	1	0	0	0	0	0	0.5	0	0	0	0	0	0	0.13	0	0	0	<table border="1"> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>2</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr> </table>	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-15.5
0	0	0	0.13	0	0	0																																																																																															
0	0	0	0.5	0	0	0																																																																																															
0	0	1	2	1	0	0																																																																																															
0.13	0.5	2	1	2	0.5	0.13																																																																																															
0	0	1	2	1	0	0																																																																																															
0	0	0	0.5	0	0	0																																																																																															
0	0	0	0.13	0	0	0																																																																																															
0	0	0	0	0	0	0																																																																																															
0	0	0	0	0	0	0																																																																																															
0	0	0	0	0	0	0																																																																																															
0	0	0	2	0	0	0																																																																																															
0	0	0	0	0	0	0																																																																																															
0	0	0	0	0	0	0																																																																																															
0	0	0	0	0	0	0																																																																																															

Fig. 2.18 The input and output patterns of erosion with 3×3 neighborhood templates [40] in two iterations and with diamond-shaped templates in one iteration.

2.4 EXPERIMENTAL RESULTS

An experimental LNCNN chip has been fabricated using $0.18 \mu\text{m}$ CMOS technology. The whole chip area is $1543 \mu\text{m} \times 1248 \mu\text{m}$ where the unit cell is $33.58 \mu\text{m} \times 43.15 \mu\text{m}$. Fig. 2.19 shows the photograph of the fabricated LNCNN chip.

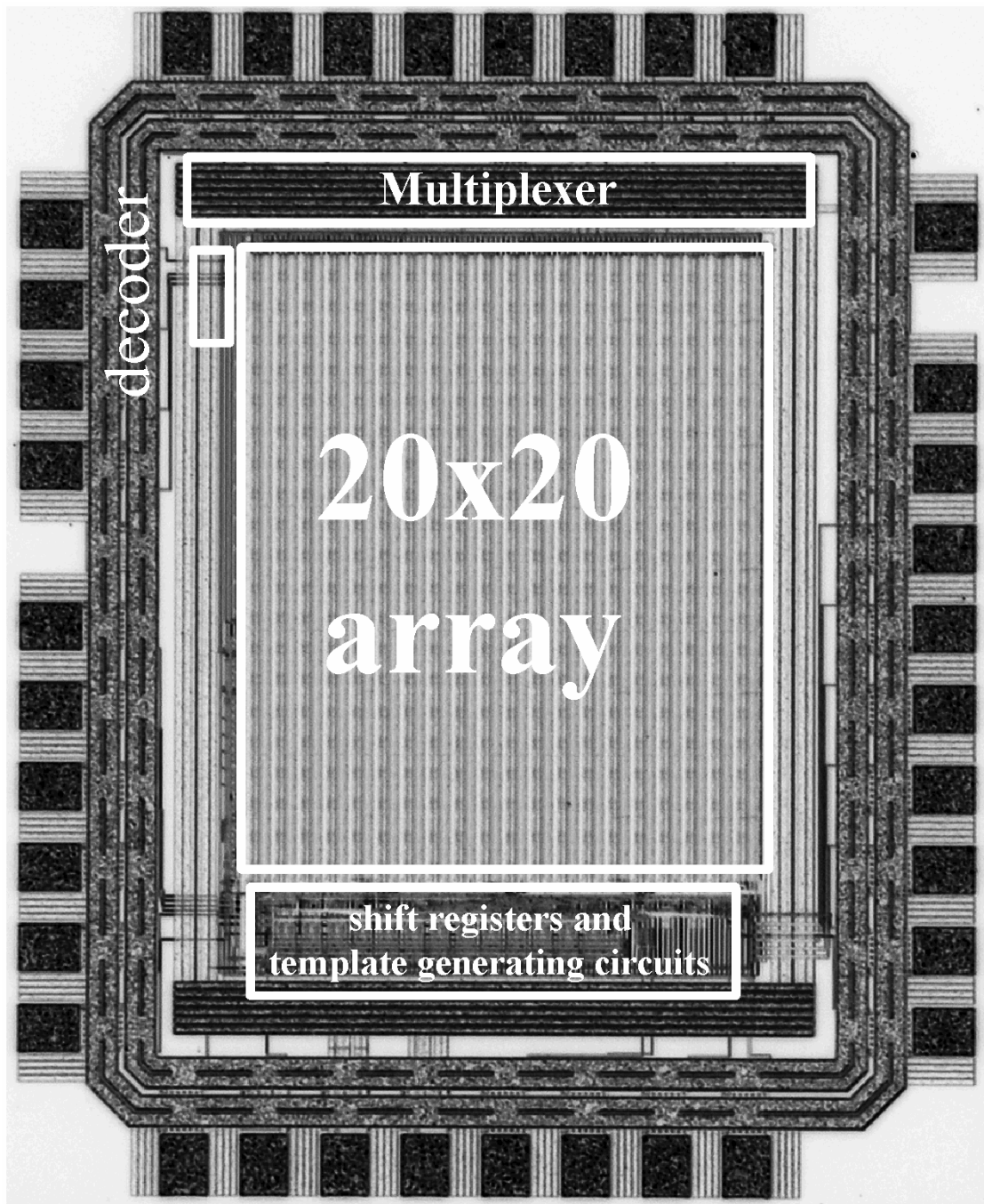


Fig. 2.19 A photograph of the fabricated 20×20 LNCNN chip.

The input image pattern in Fig. 2.15 (a) was used to verify the illusion function of the fabricated LNCNN. The digital codes of the synaptic gain were adjusted to achieve the suitable value. The binary output pattern was read out from the 5-bit pixel outputs as indicated in Fig. 2.11. The analog current-mode transients can be read out from the three real-time analog

outputs in Fig. 2.11 using the transimpedance amplifiers outside of the chip. When the analog output current is 0, the output voltage of the transimpedance amplifier is 0.9 V. Since most pixels in the input pattern shown in Fig. 2.15(a) are in white and all the white pixels remain in white after processing, the N-type synaptic gain of the PS is set to a larger value than P-type synaptic gain in the measurement. In this way, the problems of variation in the process can be overcome.

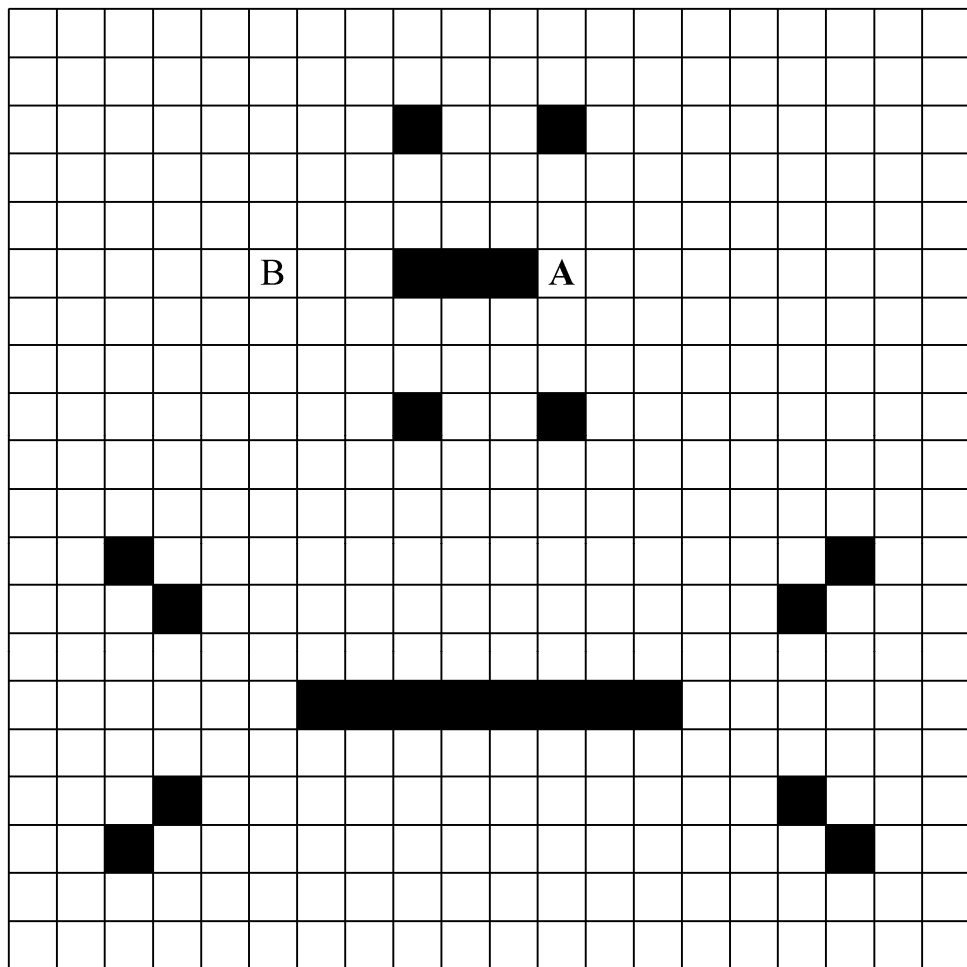


Fig. 2.20 The experimental resultant output pattern of Muller-Lyer illusion.

The measured binary output pattern is shown in Fig. 2.20. The experimental result is the same with the post-layout simulation result except the Pixel A which is black in the simulation results of Fig. 2.15(b). The reason for the error is that the bias current of the Pixel A is too small

due to process variation. Thus, the self-feedback of the Pixel A cannot keep Pixel A in the black state.

The measured analog output voltage of Pixel B through the transimpedance amplifier is shown in Fig. 2.21. The step signal is the signal *Operation_Start*, as illustrated in Fig. 2.11. As the signal *Operation_Start* rises, the analog output remains nearly at 0 V within about 1 μ s. Then it starts to rise and reaches 0.9 V at about 2 μ s. Finally, it takes 3 μ s to achieve the overall operation from black state to white. The measured transient response time is 3 μ s. From the result of post simulation, the transient response operation time is less than 0.1 μ s without the transimpedance amplifier. The difference is due to the large loading effect of the transimpedance amplifier.

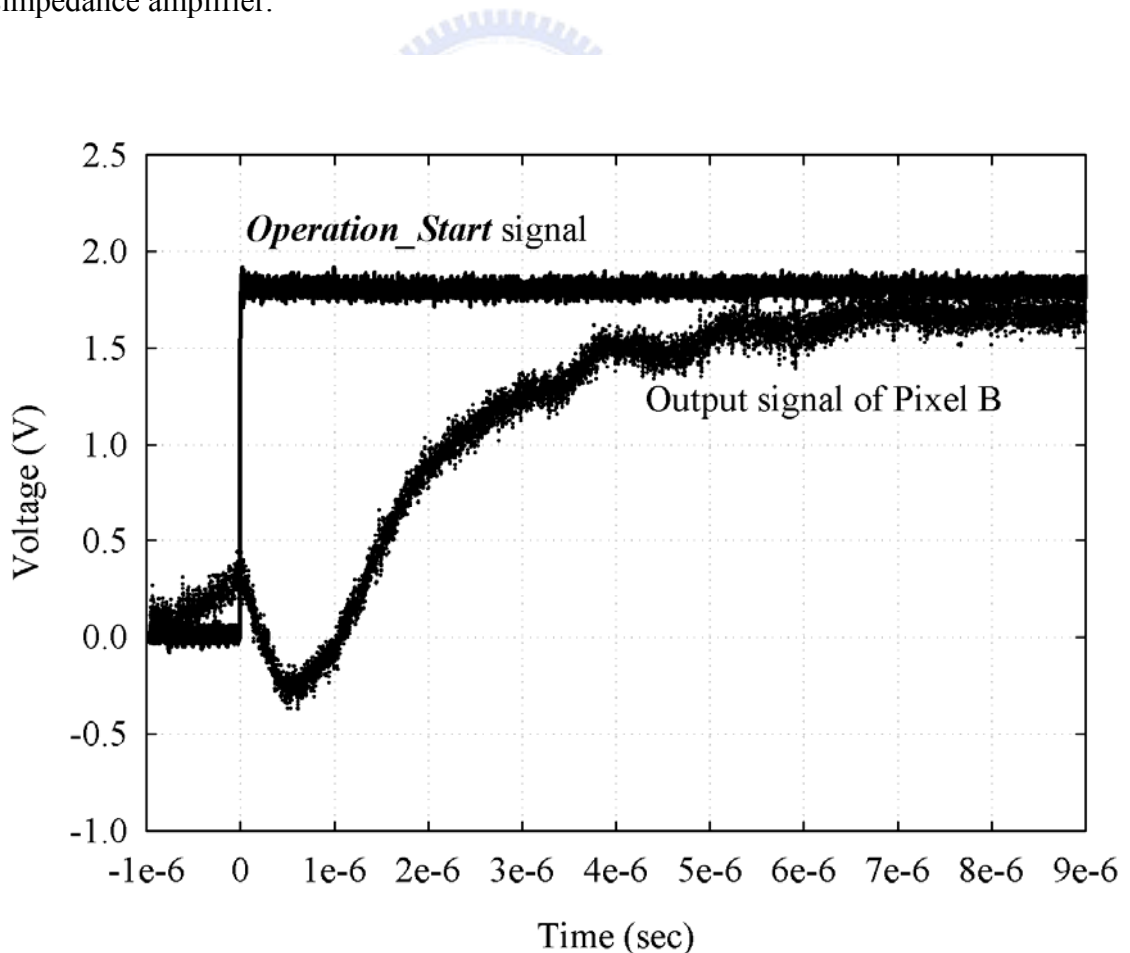


Fig. 2.21 The experimental results of Pixel B with the signal *Operation_Start*.

In the experimental result, the overall power consumption was about 0.7 mW on standby and 18 mW during the operation in the third step with a system clock frequency of 20 MHz. The comparisons of power dissipation and energy consumption per cell in the proposed LNCNN with those in CNNUC3 [132]-[133] and ACE16K [131] are listed in Table 2.3. As may be seen in Table 2.3, the cell in the LNCNN has lower power dissipation and energy consumption. The comparison between the LNCNN with symmetric templates and the proposed LNCNN is also made. The quiescent power dissipation can be much lower but the single pixel area of LNCNN with symmetric templates is much smaller. Because LNCNN with symmetric templates only can realize symmetric and positive templates, these drawbacks, therefore, save much area but cannot realize arbitrary templates. Furthermore, the tolerance to errors is based on the used templates. The diamond templates of illusion where the input patterns are combined with Gaussian noise of standard deviation 0.02 can be realized successfully by using CNN Visual Mouse Platform.

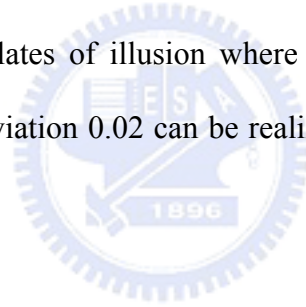


Table 2.3 COMPARISON OF LNCNN WITH CNNUC3 [132]-[133] AND ACE16K [131]

	This Work with Propagating Connection	1986 [10]	1987 [11]
Technology @ Supply	0.18 μm 6M-1P @ 1.8V	0.5 μm 3M-1P @ 3.3V	0.35 μm 5M-1P @ 3.3V
Power Dissipation (per cell)	45 μW	250 μW	180 μW
Processing Speed	20 MHz	10 MHz	30 MHz
Energy Consumption (per cell, per volt)	1.25 pJ/V	7.58 pJ/V	1.82 pJ/V
Cell Size	33.58 x 43.15 μm^2	102.2 x 120 μm^2	73.3 x 75.7 μm^2

Table 2.4 COMPARISON OF THE PROPOSED LNCNN AND LNCNN WITH SYMMETRIC TEMPLATES

	LNCNN with symmetric templates[65]-[66]	LNCNN with propagating connections
Technology	0.6 mm 1P3M N-well CMOS Technology	0.18 mm 1P6M Mixed-Signal Process
Array Size	32 x 32	20 x 20
Single Pixel Area	22 mm x 25 mm	33.58 mm x 43.15 mm
Power Supply	3 V	1.8 V
Quiescent Power Dissipation	60 m W	0.7 mW
Power Dissipation	65 mW	18 mW
Readout Time (of one pixel)	1 μ s	50 ns
Current gain of BJTs	17.5 A/A	Not Required
Dynamic range of state X_{ij}	1.3 ~ 2.1 V	- 0.5 ~ 0.5 μ A
State transition time	0.8 μ s	0.1 μ s

2.5 SUMMARY

In this chapter, a new architecture of LNCNN has been proposed. In the proposed LNCNN, the propagating connections are utilized to generate diamond-shaped large-neighborhood templates. In such a connected network, each neuron cell only needs to contact the neighboring cells without the need for farther interconnections. Therefore, such network architecture is suitable for VLSI implementation. Moreover, by separating the synapses into N-type and P-type parts without static currents, the static power dissipation can be reduced to a minimum level. Moreover, during such an operation, the synapses of direct connections with different N-type and P-type synaptic gains can also offer more functions. The connections can also be implemented both in horizontal and vertical directions and in diagonal directions to realize the circular symmetric templates. Furthermore, the LNCNN functions of diffusion, de-blurring,

and Muller-Lyer illusion has also been verified successfully. With the proposed LNCNN structure using propagating connections, many new applications and new LNCNN templates can be explored.

A CMOS large-neighborhood CNN chip with a 20×20 kernel unit array has been fabricated in 0.18- μm CMOS technology. From the experimental results of this study, it can be seen that the 5×5 template of Muller-Lyer illusion is reconstructed into a diamond-shaped LN template and the function has been successfully realized using the LNCNN and with a chip power consumption of 0.7 mW on standby and the 18 mW in operation with a system clock frequency of 20 MHz. The kernel unit of LNCNN can also perform input level of input patterns. However, due to the used shift registers for image storage, only the binary patterns can be operated with the LNCNN chip. Hence, there is great tolerance to errors due to binary signal operation and accurate circuits based on current mirrors structure are not required.

Further research on the universal machine (UM) for LNCNN needs to be conducted for various applications to be realized.

CHAPTER 3

THE DESIGN AND ANALYSIS OF A CMOS RATIO-MEMORY CNN WITHOUT ELAPSED TIME

3.1 INTRODUCTION

The cellular nonlinear (neural) network (CNN) which was proposed by Chua and Yang in 1988 [6]-[7], [40] involves a large-scale nonlinear analogic architecture for real-time signal processing. Similar in composition to the cellular automata [127]-[128], it is comprised of a massive aggregation of regularly spaced circuit clones, called cells, which communicate with each other directly and locally. With local connectivity, CNN is quite suitable for very large-scale integration (VLSI) implementation. The associated real-time and parallel-operating properties also make it popular in image processing. To date, many CNN VLSI chips have demonstrated their capabilities in realizing real-time signal and parallel processing functions [39], [126], [129]-[130]. In these chips, the templates, which can control the communications between cells, are programmable and the regular and local functions can be designed and applied on the entire CNN array. However, for the recognition of images, the programmable and space-invariant properties of CNN chips cannot realize the on-line learning directly

because the templates are space-variant due to the different local characteristics of images. To address this limitation, some algorithms that collect global image characteristics are proposed [64], [142]-[143] to learn the images.

To realize an on-line learning CNN with local-computing advantage, a learning algorithm called ratio-memory CNN (RMCNN) is proposed [65]-[67]. The ratio memory of the Grossberg outstar structure [24]-[25], [144]-[145] has been used in both feedforward and feedback neural network ICs for image processing. With the proposed RMCNN, no host computer is needed to perform the learning task off-line. It can also evaluate the correlations between cells and store these correlations on the capacitors. As a result, it no longer requires template-weight storage time or equivalent pattern recognition time which is one of the advantages of RM. The charge stored on the capacitors leaks out due to the junctions from the source and drain of CMOS to the substrate. The RMCNN utilizes this leakage effect and takes the ratio of the stored values to enhance the common characteristics of the learned patterns and to raise the recognition rate. Therefore, a very long elapsed time about 850 seconds is required after the learning period to make the weights of small correlations smaller or to approach zero by the leakage in order to enhance large correlations [67]. However, the learned characteristics in different local positions of the learned patterns are distinct and the learned values have significant differences. In the proposed RMCNN, if the elapsed time is too long in its duration, the most learned characteristics will be destroyed. However, if the elapsed time is too short in its duration, the characteristics will not be enhanced. Furthermore, when the RMCNN is utilized to learn and recognize the image patterns, the stored values keep on leaking during the recognition time and this may, with time, alter the ratio weights of the RMCNN. Finally, as the weights of cells are generated by ratio memories, a precise multiplier-divider is required.

In this work, RMCNN architecture without elapsed time [125] is proposed and analyzed to prevent the leakage effect and to simplify the circuitry. With the new algorithm, the feature enhanced ratio weights can be generated immediately after the learning period without the requirement of elapsed time and, therefore, the circuit to generate ratio weights could be very simple and remove the need for multiplier-dividers in each ratio memory. An RMCNN chip not requiring elapsed time has been designed and fabricated using TSMC 0.35- μm 2P4M mixed-signal technology. Patterns are learned and recognized with the proposed architecture and the results are analyzed and discussed. The total chip area is $4560 \mu\text{m} \times 3900 \mu\text{m}$ and the area of a single cell is $400 \mu\text{m} \times 250 \mu\text{m}$. The total power consumption is 87 mW in operation with a supply voltage of 3 V and a system clock frequency of 10 MHz.

In Section II, the models and architecture of the RN-CNN not requiring elapsed time are described. In Section III, the CMOS circuits of each block are illustrated and the HSPICE simulation results are presented to verify the functions of the blocks. In Section IV, the measurements obtained are presented and discussed. Finally, a concluding section is provided.

3.2 ARCHITECTURE AND MODELS

A. Model of RMCNN Requiring No Elapsed Time

For a standard CNN, the state equation is written as [6]-[7], [40]

$$\dot{x}_{ij} = -x_{ij} + Z_{ij} + \sum_{C_{kl} \in S_{ij}} A_{kl} y_{kl} + \sum_{C_{kl} \in S_{ij}} B_{kl} u_{kl} \quad (3.1)$$

where x_{ij} , y_{ij} , and u_{ij} are the state, output, and input of the neuron cell C_{ij} in a CNN array, respectively; the coefficient Z_{ij} , called the template \mathbf{Z} , is the threshold of the neuron cell C_{ij} ; and, A_{kl} and B_{kl} are the coefficients, called templates \mathbf{A} and \mathbf{B} , respectively, which are

multiplied with output y_{kl} and input u_{kl} of the cell C_{kl} , respectively, in the sphere of influence (S_{ij}) of the neuron cell C_{ij} . The two sets of products are accumulated over all the cells C_{kl} in S_{ij} of the neuron cell C_{ij} . However, the state equation of RMCNN can be expressed as

$$\dot{x}_{ij} = -x_{ij} + u_{ij} + \sum_{C_{kl} \in S_{ij}} A_{ijkl}(T_E) y_{kl} \quad (3.2)$$

where S_{ij} and y_{ij} are defined in an $M \times N$ array as

$$S_{ij} = \{C_{ij} | 1 \leq k \leq M, 1 \leq l \leq N, 1 \leq i \leq M, 1 \leq j \leq N, |k - i| + |l - j| \leq 1\} \quad (3.3)$$

and

$$y_{ij} = \frac{1}{2} |x_{ij} + 1| - \frac{1}{2} |x_{ij} - 1|. \quad (3.4)$$

Template $A_{ijkl}(T_E)$ is a space-variant template and is a function of elapsed period when an elapsed time is applied. Therefore, the template A_{ij} can be written as

$$A_{ij}(T_E) = \begin{bmatrix} 0 & a_{ij(i-1)j}(T_E) & 0 \\ a_{iji(j-1)}(T_E) & 0 & a_{iji(j+1)}(T_E) \\ 0 & a_{ij(i+1)j}(T_E) & 0 \end{bmatrix} \quad (3.5)$$

where $a_{ijkl}(T_E)$ is the template coefficient of the cell C_{ij} to stimulate the cell C_{kl} and is generated by using the equation

$$a_{ijkl}(T_E) = \frac{\sum_{p=1}^m \int_{T_p} u_{ij}^p \cdot u_{kl}^p dt - L_{kl}(T_E)}{\sum_{kl} \left| \sum_{p=1}^m \int_{T_p} u_{ij}^p \cdot u_{kl}^p dt - \sum_{kl} L_{lk}(T_E) \right|}, \quad kl \in \{(i-1)j, i(j-1), i(j+1), (i+1)j\}. \quad (3.6)$$

In (3.6), u_{ij}^p and u_{kl}^p are the inputs of the pixels in the learned p^{th} pattern when m patterns are learned in the learning period. In the learning period for the cell C_{ij} as in (6), its input signal is multiplied with the inputs of its four nearest cells and then these values are integrated with a learning time T_p of each pattern, respectively, to generate one set of the correlations, called correlated weights. The template coefficient is generated by the ratio of one correlated weight and summation of the four absolute correlated weights. $L_{kl}(T_E)$ is the leakage in an elapsed time T_E . The leakage depends on the correlation between two cells and process parameters. When a very long period of elapsed time is applied, the remnants in the four ratio memories around one cell may be 0. However, when a short period of elapsed time is applied, the enhancement is limited. As a response, a new template generating method is proposed. First, the mean M_{ij} of the learned absolute correlated weights is generated as

$$M_{ij} = \text{avg}_{kl} \left(\sum_{p=1}^m \int_{T_p} u_{ij}^p \cdot u_{kl}^p dt \right), \quad kl \in \{(i-1)j, i(j-1), i(j+1), (i+1)j\}. \quad (3.7)$$

where m is the number of learned patterns, T_p is the learning time of one pattern, and u_{ij}^p and u_{kl}^p are the inputs of cell(i, j) and cell(k, l), respectively, in the learned p^{th} pattern. The ratio weight a'_{ijkl} is then generated as following:

$$a'_{ijkl} = \begin{cases} 0 & , \text{ if } \left| \sum_{p=1}^m \int_{T_p} u_{ij}^p \cdot u_{kl}^p dt \right| < M_{ij} \\ 1/\text{PN}_{ij} & , \text{ if } \left| \sum_{p=1}^m \int_{T_p} u_{ij}^p \cdot u_{kl}^p dt \right| \geq M_{ij} \end{cases}, \quad kl \in \{(i-1)j, i(j-1), i(j+1), (i+1)j\} \quad (3.8)$$

,where PN_{ij} means the number of the absolute correlated weights which is larger than M_{ij} . As shown in (3.8), the template coefficient is generated by counting the number of the absolute

correlated weights which is larger than the mean M_{ij} . As a result the template value is set to $1/PN_{ij}$ when its absolute correlated weight is larger than the mean. This retains the overall summation of absolute template coefficients a'_{ijkl} of template A at 1 to avoid any divergence in recognition.

To demonstrate why the coefficient which is larger than the mean is retained, a simple model of the absolute ratio weight can be constructed as following:

$$\frac{P-L}{(P-L)+(Q-L)+(R-L)+(S-L)} > \frac{P}{P+Q+R+S}, P > L, Q > L, R > L, S > L \quad (3.9)$$

where $P, Q, R,$ and S represent the four absolute correlated weights generated in the learning period, and L represents the average leakage in the elapsed period and, after that period, the absolute ratio weight should be enlarged if the coefficient is retained. The condition to make (3.9) valid can be derived as following:

$$\frac{1}{k} \cdot \frac{P-L}{M-L} > \frac{1}{k} \cdot \frac{P}{M}, P > M, P > L, Q > L, R > L, S > L \quad (3.10)$$

where M is the mean of $P, Q, R,$ and S and the coefficient k is 4. When P is larger than M , the absolute ratio weight could be enlarged after a period of elapsed time. Hence, the coefficient is retained by comparing it with the mean value. However, as the average leakage L is larger than the correlation P , it is unreasonable to get a negative ratio weight value. The absolute ratio weight should be larger than zero. As one absolute correlated weight leaks to zero, mean M is evaluated using the residuary absolute correlated weights. In this situation, k is reduced to 3 because only three absolute correlated weights are averaged. Hence, if one of the correlated weights leaks to 0, k should be reduced by 1. In the proposed algorithm, when P is larger

(smaller) than or equal to M , the absolute ratio weight is chosen to be $1/k$ (0). This makes the sum of the absolute ratio weights equal to 1 around one cell.

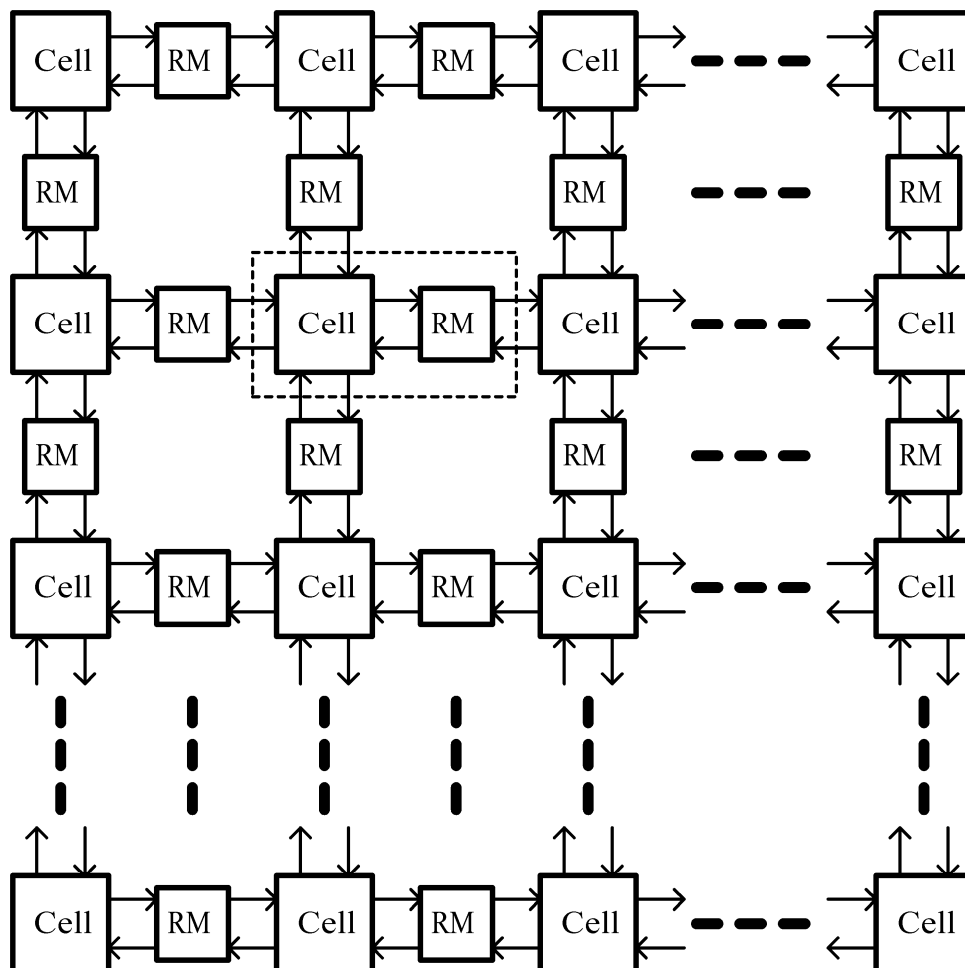


Fig. 3.1 The general architecture of the RMCNN.

B. Architecture of RMCNN Not Requiring Elapsed Time

The general architecture of RMCNN is shown in Fig. 3.1. The RM block is located at any two of the nearest cells to evaluate and store the correlated weights. In each cell, the circuitry is required to average the absolute correlated weights from the four peripheral RM blocks and to compare the correlated weights with the mean value. Meanwhile, a counter is also required to count the PN in (3.8) around the cell. With reference to the architecture, a 9×9 RMCNN chip has been designed. The structure of the kernel unit of the RMCNN not requiring elapsed time,

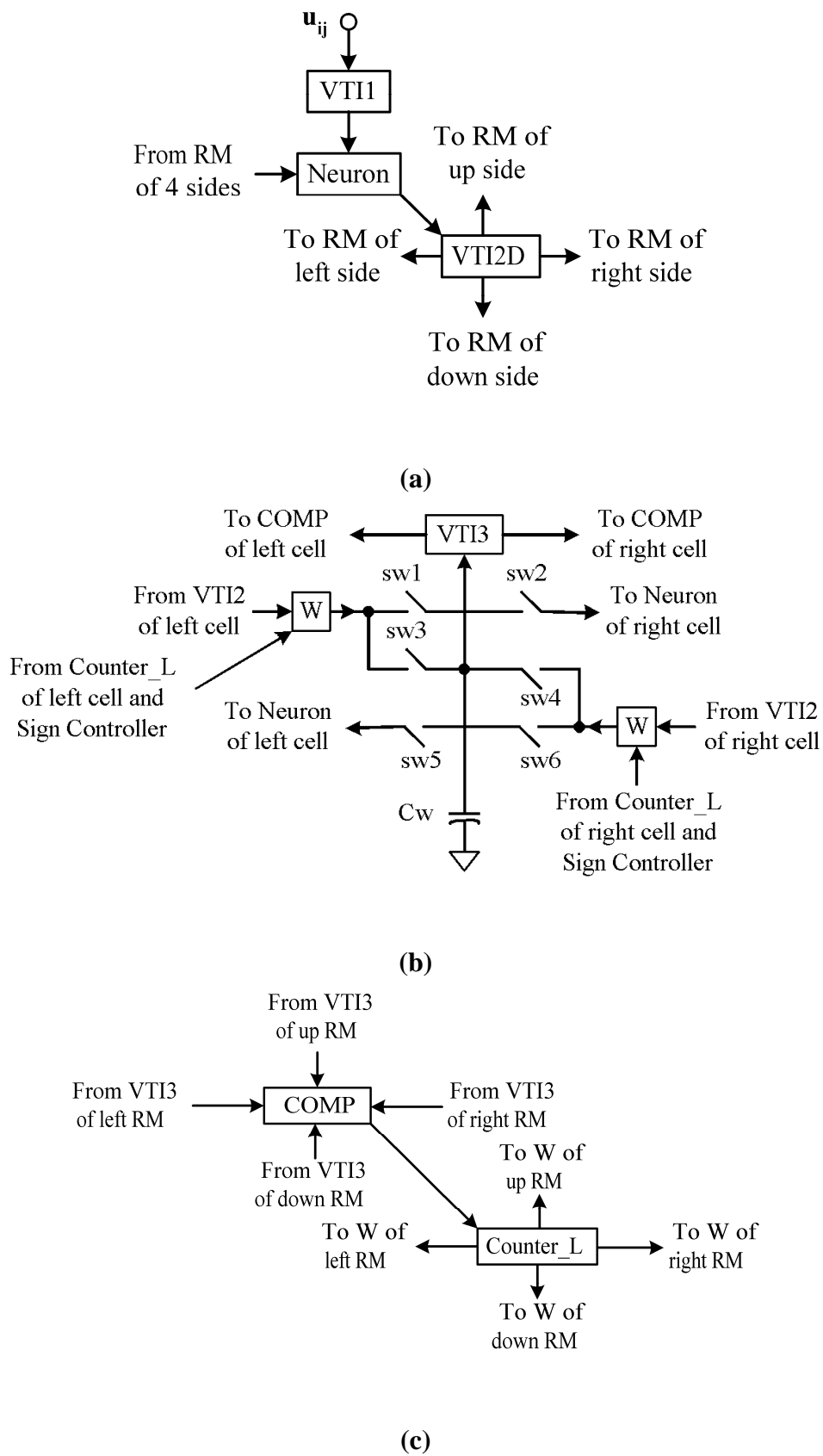


Fig. 3.2 (a) The input stage and neuron, (b) RM, and (c) comparator and counter in the kernel unit of RMCNN.

which is demonstrated in Fig. 3.1 Fig. 3.1 by broken lines, is separated into three parts in Fig. 3.2(a)-Fig. 3.2(c). There are four RMs: up, down, and on the right, and left sides around one cell and two neighboring cells around one RM. In Fig. 3.2(a), block Neuron is a neuron composed of a resistor and capacitor and block VT11 is a voltage-to-current converter with a sign detector to convert the input voltage into current. Block VT12 is a voltage-to-absolute-current converter with a detector. It detects the sign of the current with an absolute current output. In Fig. 3.2(b), block W is the synaptic gain block to multiply the absolute input current from VT12 with a chosen weight of 1/4, 1/3, 1/2, or 1 and the output sign is controlled by a sign controller. The weight is controlled by block Counter_L in Fig. 3.2(c). Block COMP is a comparator that can compare four absolute currents from RMs with the average of these four currents. Block Counter_L counts how many currents are larger than the average current. Block Counter_L can also generate the signals to control the weights of blocks W by the comparing and counting results.

In the learning period, only switches sw1, sw2, sw4, and sw5 in Fig. 3.2(b) are open. When p^{th} pattern is learned, binary input u_{ij}^P of cell(i, j) is sent into block VT11 and the output current is sent to block Neuron to generate the state voltage of cell(i, j). The positive or negative state of cell(i, j) is detected by block VT12 and the absolute current is extracted. The sign and the absolute current of cell(i, j) are both sent to block W. In the learning period, the weight of block W is set to 1/4. If the states of cell(i, j) and its neighboring cells are the same (different), it is decided to charge (discharge) capacitor Cw with the absolute current multiplied by 1/4. The learning time of one pattern can also be adjusted to prevent the voltage saturation of the capacitor.

After all the patterns are learned, block VT13 converts the voltage stored on capacitor Cw into two absolute currents for the nearest two comparators. At the same time, the correlative

signs are also been stored. There are four absolute currents from the neighboring RMs in one cell. The comparator generates a mean current of the four absolute currents and compares the four currents with the mean current. The comparisons are counted by block Counter_L to decide the ratio weights of block W. When the N (4-N) currents in neighboring RMs are larger (smaller) than the mean current, the weights of blocks W are set to 1/N (0) where N could be 1, 2, 3, or 4. The ratio weights are set at 1/4 for each block W only if the four currents are equal.

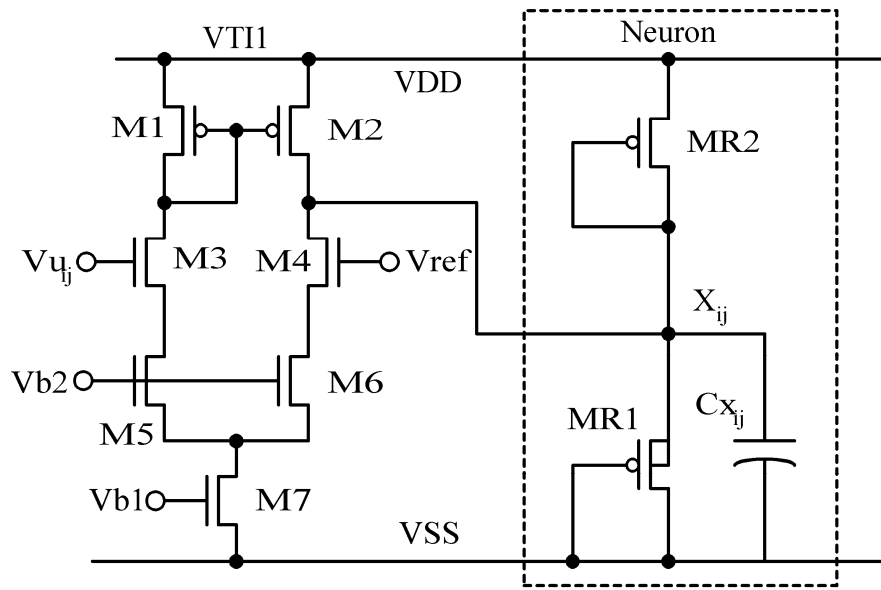
In the recognition period, the switches sw1, sw2, sw5, and sw6 in Fig. 3.2 (b) are closed. The gray level input u_{ij} in Fig. 3.2(a) of the noisy pattern is sent into block Neuron and the operation of recognition starts.

3.3 CIRCUIT IMPLEMENTATION WITH SIMULATION

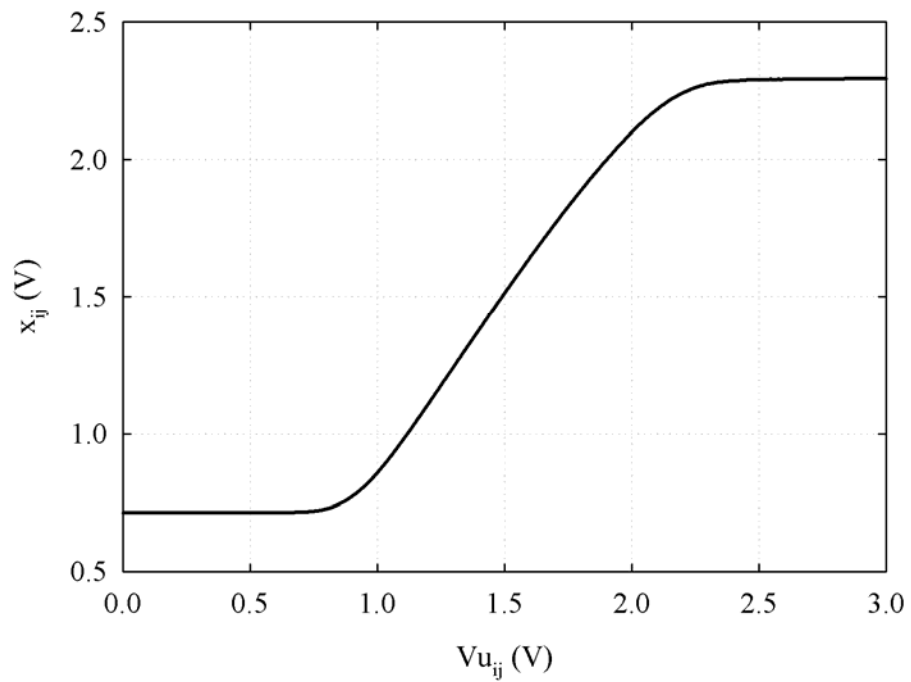
RESULTS

A. Circuit Implementation with Simulation Results

Blocks VT11 and Neuron are shown in Fig. 3.3(a). Block VT11 is constructed using a simple differential amplifier. M5-M6 are used to degenerate the transconductance of the amplifier and to enlarge the linear operating range. V_{ref} is set to 1.5 V and V_{b2} at 2.5 V. V_{b1} is controlled by a mirror with a current of 5.5 μ A. Block Neuron is simply composed of a resistance and a capacitor. The resistance is constructed using MR1 and MR2 and the capacitor is realized by the parasitic capacitance at node X_{ij} . MR1 and MR2 are realized by PMOS because the substrate of MR1 can be connected to source of MR1 to prevent body effect. Hence, the state voltage X_{ij} can be set to 1.5 V (1/2 V_{DD}) initially. The transfer characteristic of VT11 is shown in Fig. 3.3(b). The transfer curve is linear as the input voltage $V_{u_{ij}}$ is between 0.9 V and 2.1 V.



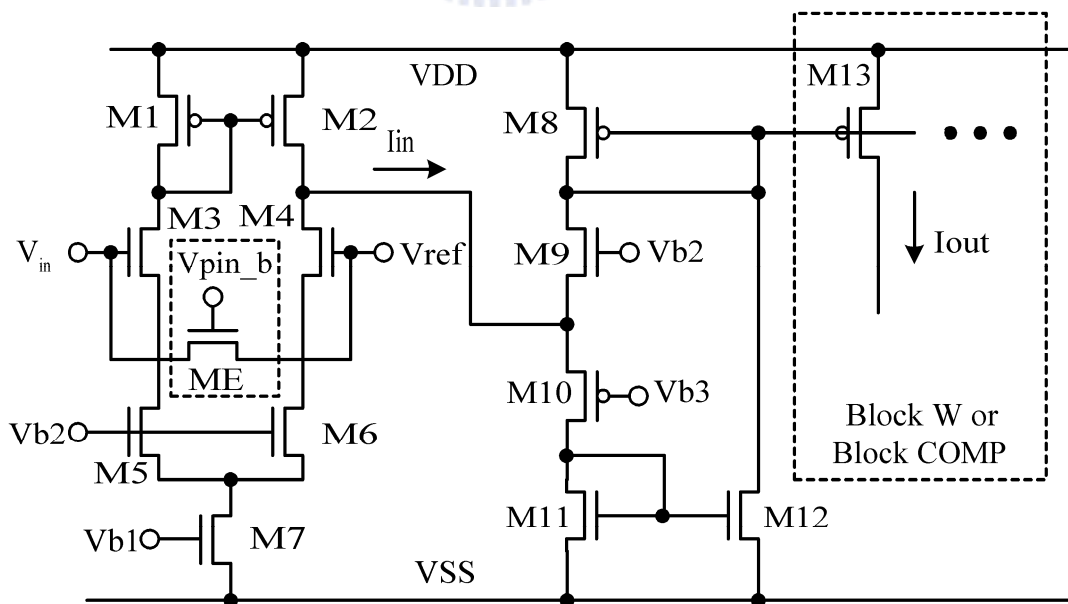
(a)



(b)

Fig. 3.3 (a) The circuits of the blocks VT11 and Neuron. (b) The transfer characteristic of the block VT11.

Fig. 3.4(a) depicts blocks VTI2 and VTI3. The circuit represented in broken lines belongs to the next stage, that is, block W or block COMP. Block VTI2 is similar to block VTI3 except the device ME. VTI2 contains ME and V_{pin_b} is set to low, when the patterns are learned and during the recognition period, in order to turn on the function of block VTI2. V_{b1} is biased by a mirror with a current of $5.5 \mu\text{A}$ and V_{b2} is set to 2.5 V . V_{b3} and V_{ref} are each set to 1.5 V . The combination of M9 with V_2 and M10 with V_3 can stop the static current. The difference ($V_2 - V_3$) is smaller than the summation of the threshold voltages of M9 and M10. Hence, M9 and M10 are turned off when there is no input current I_{in} . VTI2 and VTI3 each contain a differential amplifier and an absolute current converter. The differential amplifier generates positive and negative currents based on V_{in} . The positive (negative) current sent to the absolute current converter turns on the device M10 (M9) and M9 (M10) is still turned off. The positive current is inverted twice with two current mirrors, M11/M12 and M8/M13. The negative current is



(a)

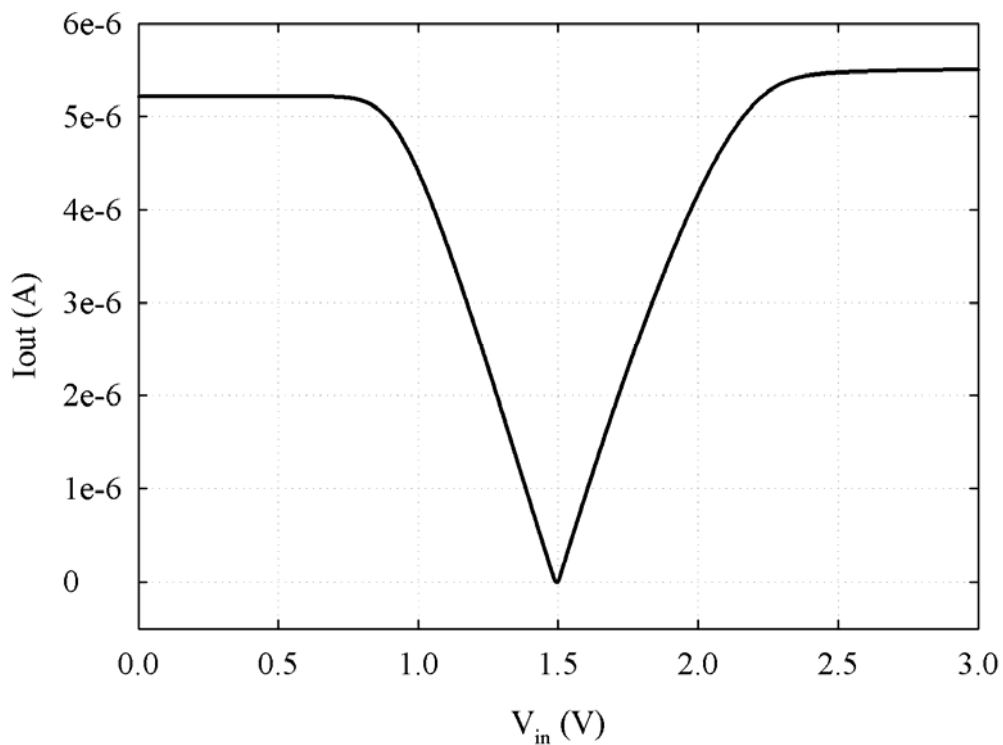


Fig. 3.4 (a) The circuit of the blocks VTI2 (with ME) and VTI3 (without ME) (b) The transfer characteristic of the block VTI2 and VTI3.

inverted once with the current mirror M8/M13. As a result, the output current I_{out} is the absolute current of I_{in} .

Fig. 3.5 shows the circuit of block W. Switches S_{wa} - S_{wf} are controlled by block Counter_L to multiply the current with a gain of 1/4, 1/3, 1/2, or 1. Based on the signs of the patterns and learned correlated weights in different periods, signal $Sign_Con$ is set to a proper digital code to decide the sign of block W, as shown in Fig. 3.6. The detector is constructed using two cascaded inverters and amplifies the input signals to achieve a digital level. During the learning period, only switch S_{wd1} is closed. Signal $Sign_Con$ is determined by input voltage V_{in} in

block VTI2 of cell(i,j) and input voltage V_u of the nearest neighboring cell(k,l) through an exclusive gate.

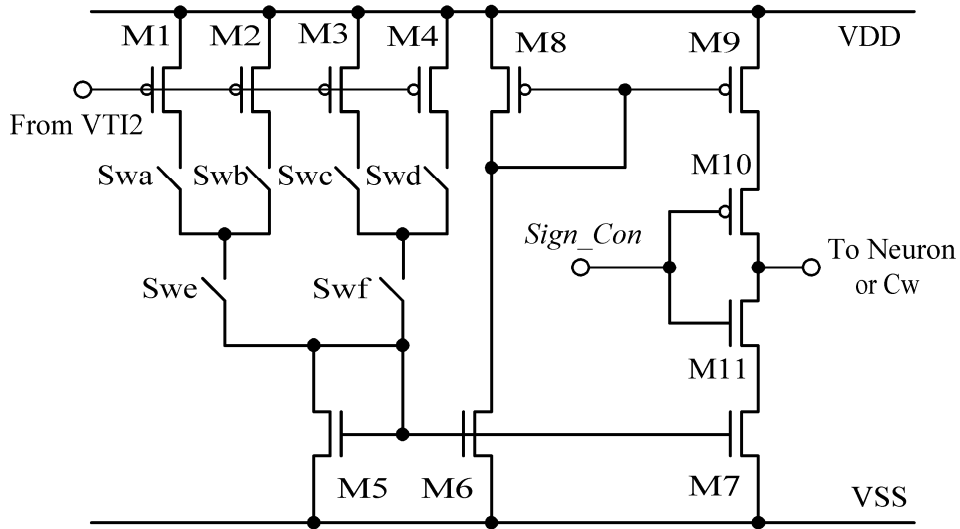


Fig. 3.5 The circuit of the block W.

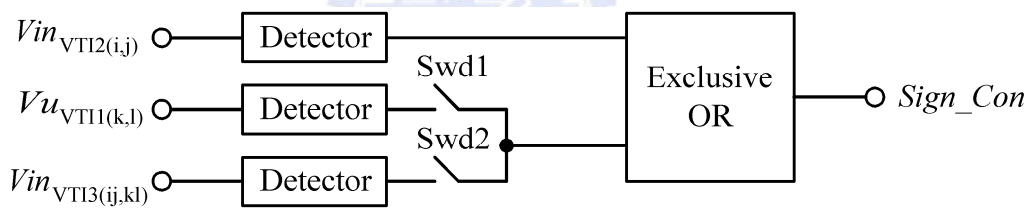


Fig. 3.6 The block diagram of the sign controller where the detector is composed of two cascaded inverters.

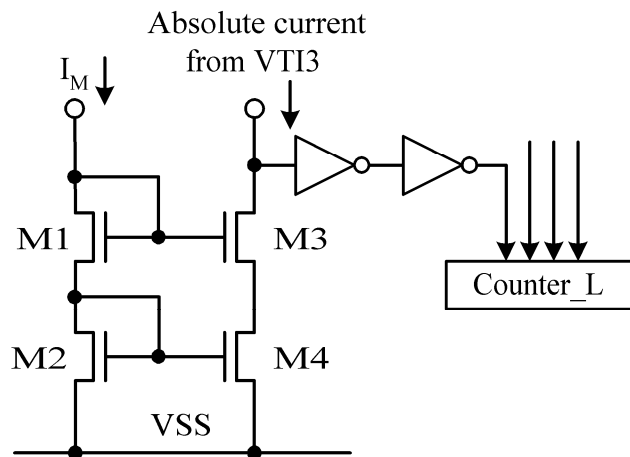


Fig. 3.7 The circuit of the block COMP.

During the recognition period, only switch Swd2 is closed. Signal *Sign_Con* is decided by input voltage V_{in} in block VTI2 of cell(i,j) and input voltage of block VTI3, which is the correlation between cell(i,j) and cell(k,l) stored on the capacitor C_w in Fig. 3.2 (b).

In block COMP in Fig. 3.7, the up, down, left, and right currents from blocks VTI3 are gathered and averaged by the current mirror. The four directional currents from blocks VTI3 are compared with the averaged current I_M and the comparing results are converted into digital signals using two cascaded inverters. The four digital signals are counted by block Counter_L, which is composed of two cascaded D-flip-flops and whose function it is to count the four digital signals one at a time.

B. Operational Steps

The operation is separated into three periods: the learning, weight generating, and recognition periods. In the learning period, blocks VTI1, Neuron, VIT2D, and W are active. Input voltage $V_{u_{ij}}$ of the learned pattern is transferred into the current signal and the current signal is applied on block Neuron to produce an output state voltage $V_{x_{ij}}$. With state voltage $V_{x_{ij}}$, block VTI2 generates the absolute current and this is multiplied by 1/4 through controlling the switches Swa-Swf. The polarity of the output current of block W is controlled by signal *Sign_Con* which is generated by the sign controller in Fig. 3.6 where only switch Swd1 is closed. With the output current of block W, capacitor C_w is charged or discharged in an interval T_p . After all the patterns are learned, the weight generating period starts.

In the weight generating period, the switches in Fig. 3.2(b) are all open. The voltage on C_w is applied to VTI3 to generate two absolute currents while the sign is also detected, simultaneously. With four absolute RM currents in four directions, mean current I_M is generated and compared with the four absolute currents as shown in Fig. 3.8. The four comparators outputs are counted by block Counter_L, which sends the control signal to four

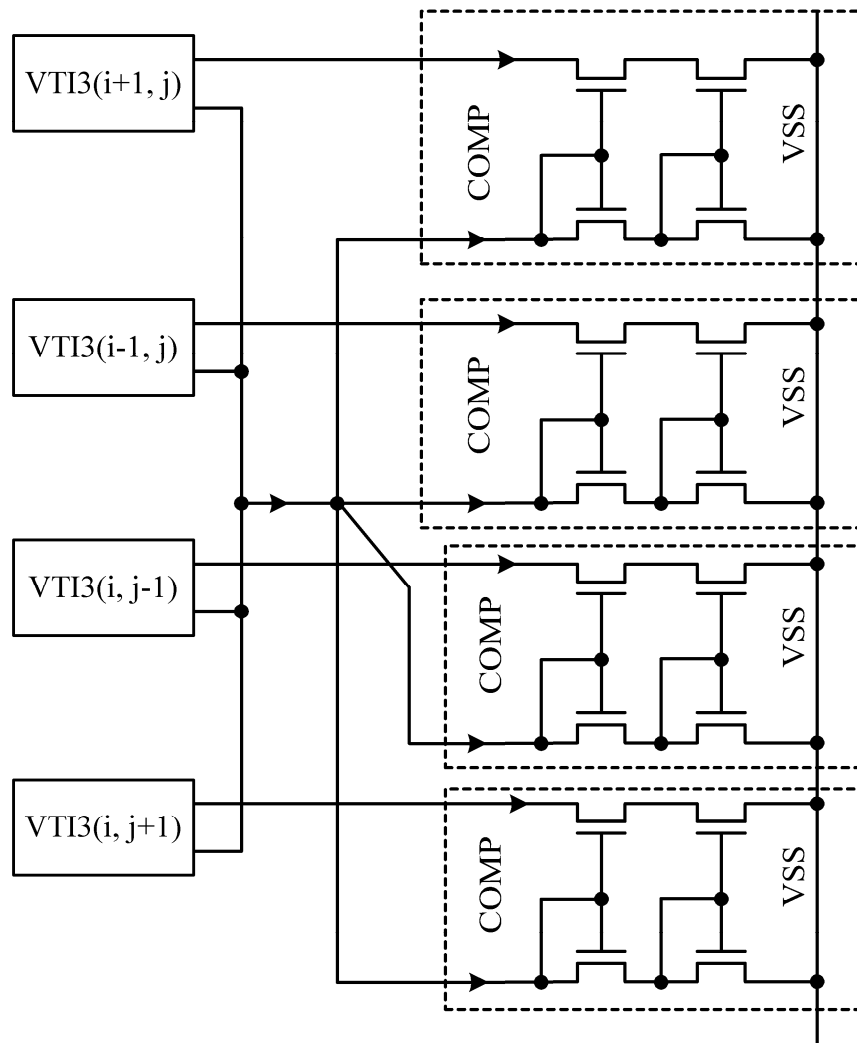


Fig. 3.8 The four absolute currents from VT13 are averaged and compared with the mean current.

blocks W to generate corresponding weights. As a result, the RMCNN is ready for recognition. In the third period, the noisy patterns are sent into the RMCNN. Switches $sw1$, $sw2$, $sw5$, and $sw6$ in Fig. 3.2(b) are closed and so is $Swd2$ in Fig. 3.6. At the same time, device ME in Fig. 3.4 (a) is turned off to commence the operation of recognition. A set of patterns in Fig. 3.9 are learned with Matlab by the proposed algorithm of a 9×9 RMCNN. The Gaussian noise patterns with different standard deviation are recognized as shown in Fig. 3.10 where the standard deviation is 0.3. The resultant recognition rate is shown in Fig. 3.11 which is

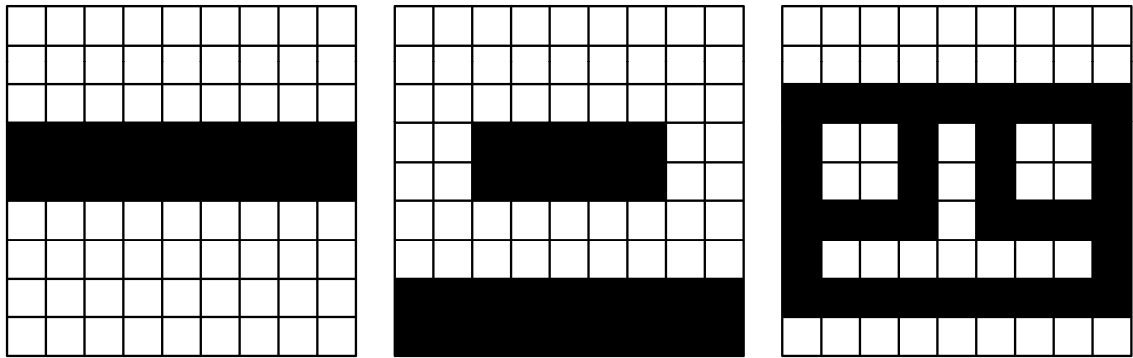


Fig. 3.9 Input patterns in the learning period.

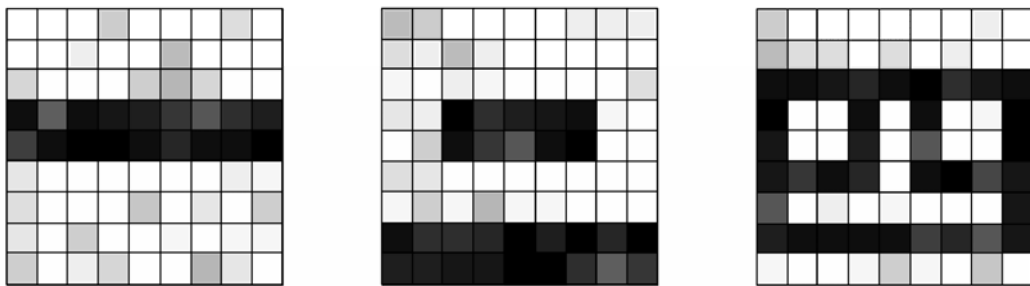


Fig. 3.10 Gaussian noise patterns with standard deviation of 0.3 to be recognized.

compared by directly amplifying the noisy patterns with an inverter where Gaussian noise is applied with a standard deviation normalized to the binary state of 1 and -1. As can be seen from Fig. 3.11, when the tolerance level is 50 %, the recognition rate is better than that of direct amplification. However, when the tolerance is sterner, the recognition rate is reduced. Due to the fact that template **A** is a non-self-feedback template, the recognized output patterns can not be pulled to a saturated state and, hence, the recognition rate is degraded. The template values of cell(4, 5), cell(5, 3), cell(8, 5), and cell(7, 5) are listed in Table 3.1 and are compared with RMCNN with an elapsed time of 800 sec. As can be seen in Table 3.1, the templates are almost the same except for some negligible coefficients.

The recognition rate is shown in Fig. 3.12 where the self-feedback RMCNN without

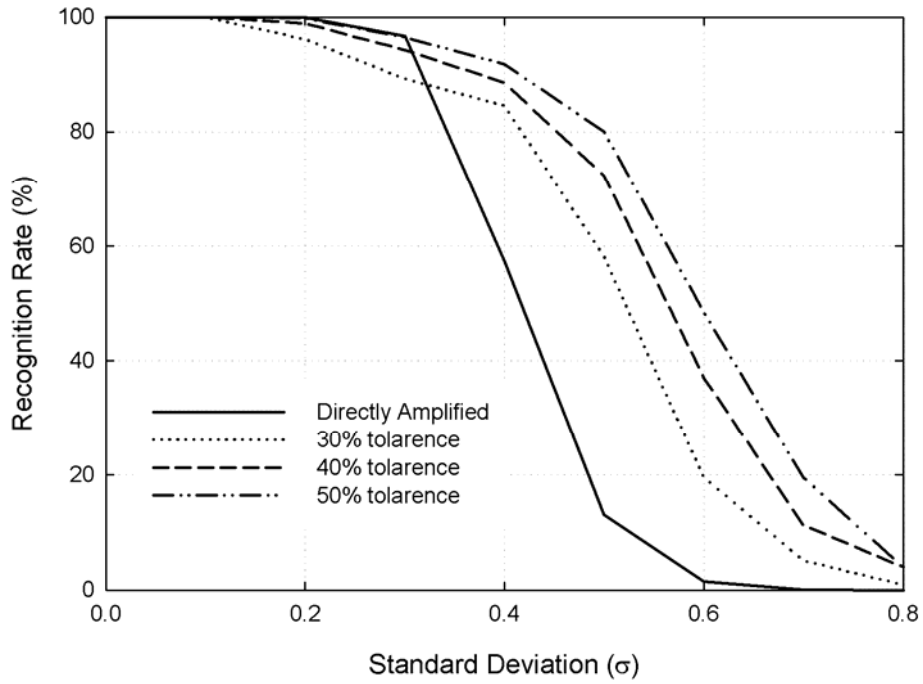


Fig. 3.11 The recognition rates by using proposed RMCNN and by being directly amplified.

elapsed time is simulated and the result is compared with that of direct amplification. Since the closed loop in the self-feedback RMCNN saturates the output, the recognition rate can be raised and is better than that without self-feedback and that of direct amplification. In this paper, only the test chip of the non-self-feedback RMCNN without elapsed time is designed and measured to verify the proposed RMCNN algorithm not requiring elapsed time. A self-feedback RMCNN not requiring elapsed time can be designed similarly.

C. Simulation Results with Large-Neighborhood Templates

The large-neighborhood diamond templates are also been applied to the proposed RMCNN requiring no elapsed time. The RMCNN with large-neighborhood diamond templates are simulated with Matlab. The tested image patterns of 18×18 array are learned and the large-neighborhood templates are generated by using the proposed method. Fig. 3.13 shows the

Table 3.1 THE COMPARISONS OF TEMPLATES A IN CELL(4, 5), CELL(5, 3), CELL(8, 5), AND CELL(7, 5) BETWEEN RMCNN WITH AND WITHOUT ELAPSED TIME

Templates A	With Elapsed Time	Without Elapsed Time
9 x 9 RMCNN r = 1	$A_{4,5}(800s) = \begin{bmatrix} 0 & -0.5 & 0 \\ 0 & 0 & 0 \\ 0 & 0.5 & 0 \end{bmatrix}$	$A_{4,5}(800s) = \begin{bmatrix} 0 & -0.5 & 0 \\ 0 & 0 & 0 \\ 0 & 0.5 & 0 \end{bmatrix}$
	$A_{5,3}(800s) = \begin{bmatrix} 0 & 0.944 & 0 \\ 0.018 & 0 & 0.018 \\ 0 & -0.018 & 0 \end{bmatrix}$	$A_{5,3}(800s) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$
	$A_{8,5}(800s) = \begin{bmatrix} 0 & 0 & 0 \\ 0.5 & 0 & 0.5 \\ 0 & 0 & 0 \end{bmatrix}$	$A_{8,5}(800s) = \begin{bmatrix} 0 & 0 & 0 \\ 0.5 & 0 & 0.5 \\ 0 & 0 & 0 \end{bmatrix}$
	$A_{7,5}(800s) = \begin{bmatrix} 0 & 0.33 & 0 \\ 0.33 & 0 & 0.33 \\ 0 & 0 & 0 \end{bmatrix}$	$A_{7,5}(800s) = \begin{bmatrix} 0 & 0.33 & 0 \\ 0.33 & 0 & 0.33 \\ 0 & 0 & 0 \end{bmatrix}$

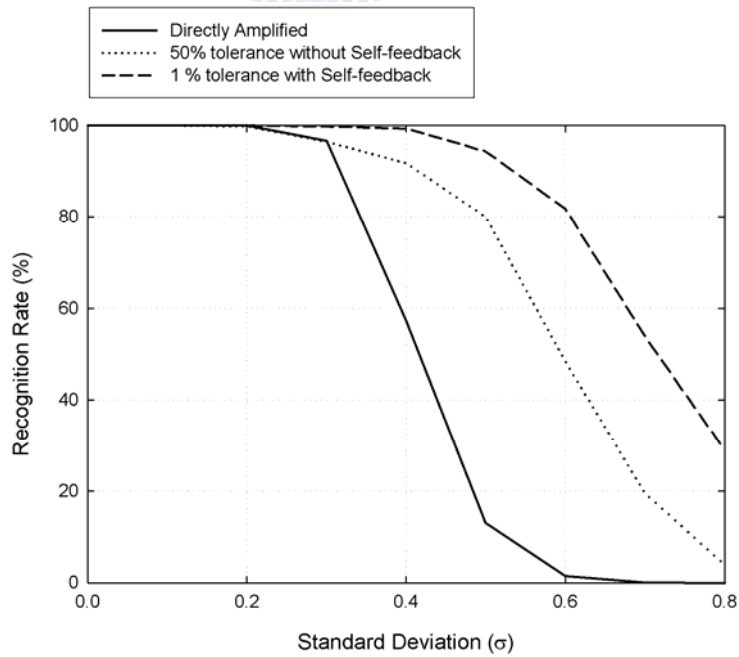


Fig. 3.12 The comparison of the recognition rates by using proposed RMCNN with self-feedback, without self-feedback of 50% tolerance and by being directly amplified.

comparison of simulated recognition rates when the neighborhood of radius r' are 1 and 3. RMCNN requiring no elapsed time with 3×3 neighborhood templates can learn 6 patterns. It can be seen that the recognition rate with large neighborhood is worse than that with 3×3 neighborhood templates when 7 patterns are learned and recognized. The reason can be explained by Fig. 3.14 that the smaller correlations are retained when a smaller elapsed time is applied. That is, when the large neighborhood diamond templates are used in the proposed algorithm, many smaller correlations are retained and these ratio weights of small correlations raise the error rates. Hence, the proposed algorithm should be modified especially when the large neighborhood diamond templates are used.

In order to generate a template as the former RMCNN with a longer elapsed time, the procedure of the proposed algorithm is modified. By using the modified method, the comparison of the correlations and their mean is repeated. The procedure is stopped when the

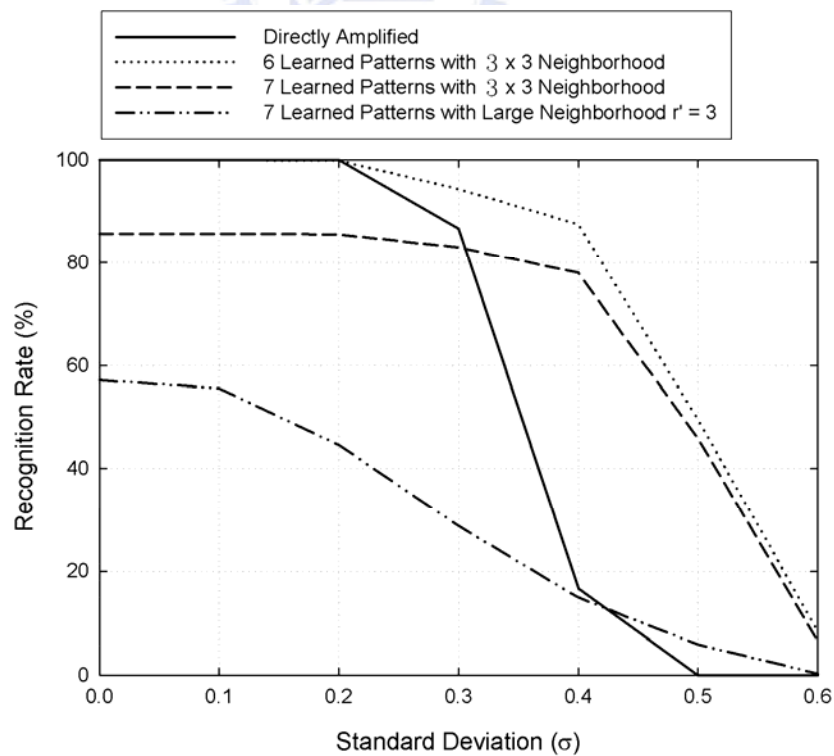


Fig. 3.13 The comparison of recognition rates with 3×3 neighborhood templates and large neighborhood diamond templates of $r'=3$.

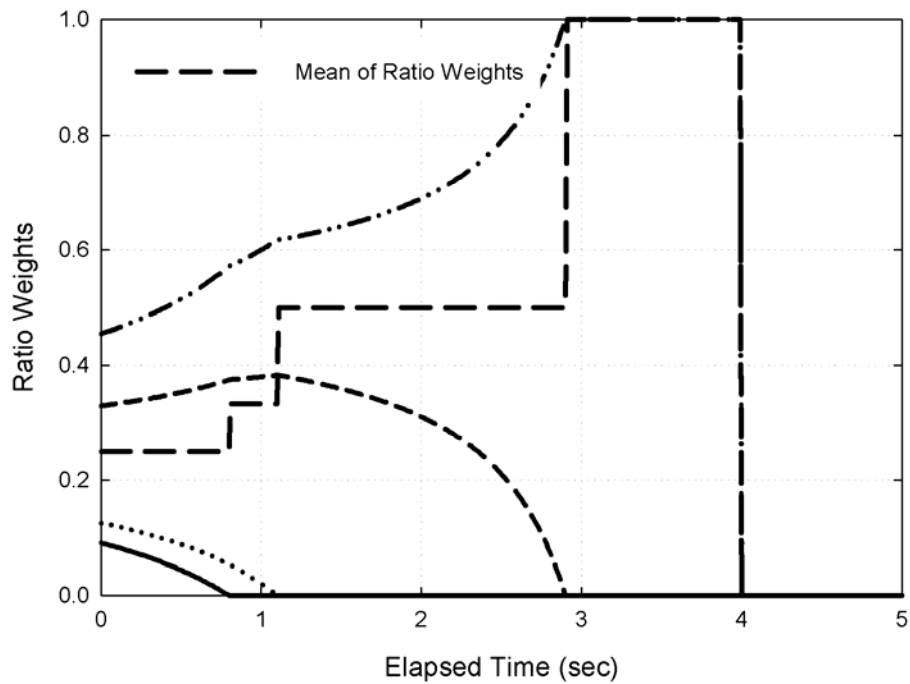
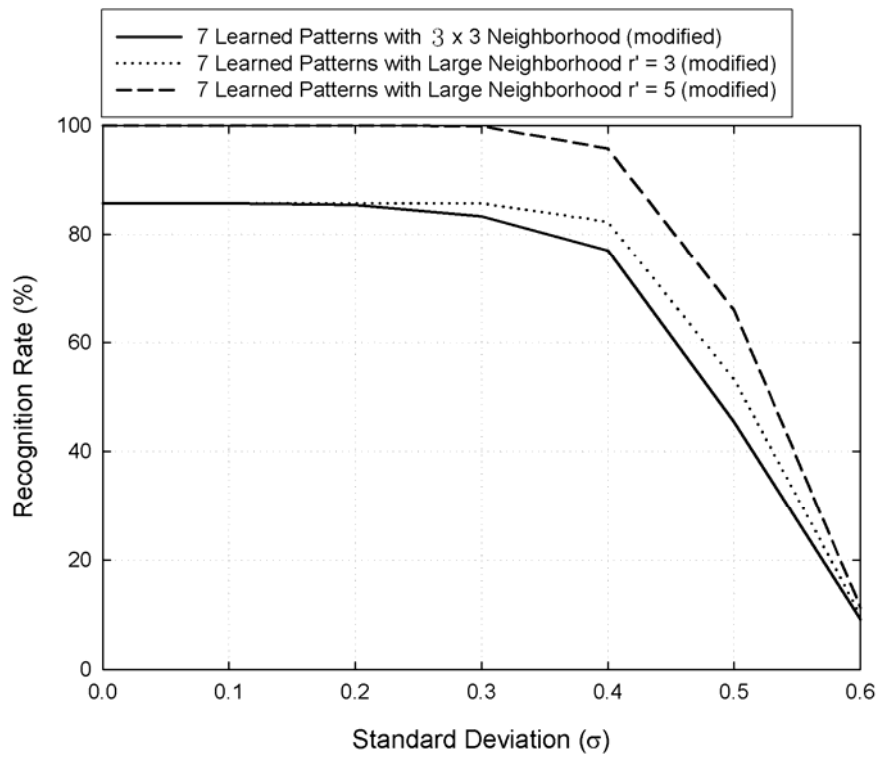
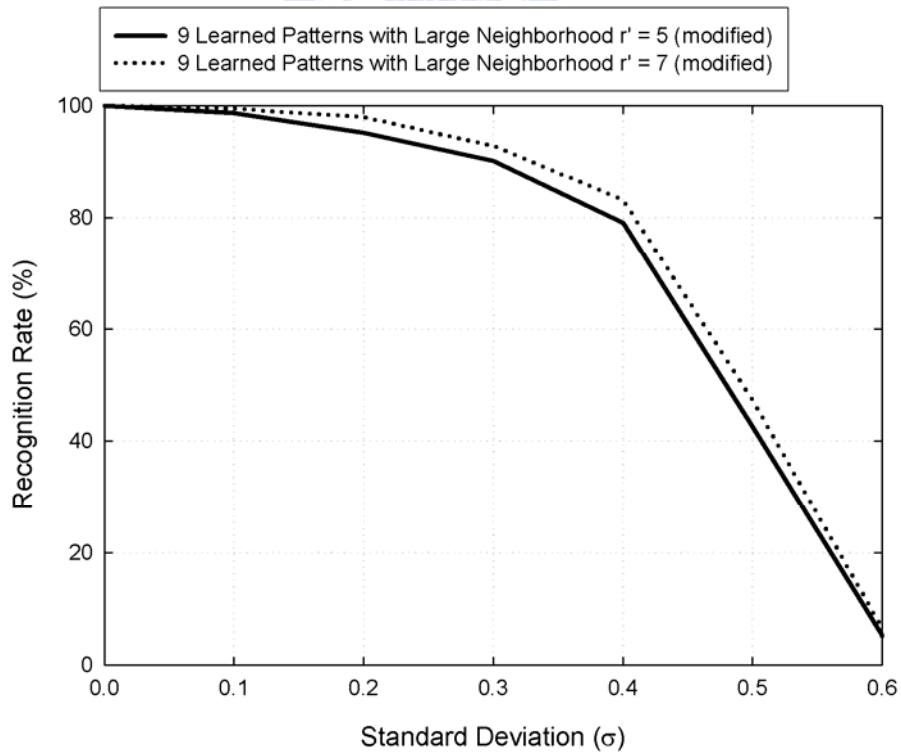


Fig. 3.14 The ratio weights of RMCNN with different elapsed time.

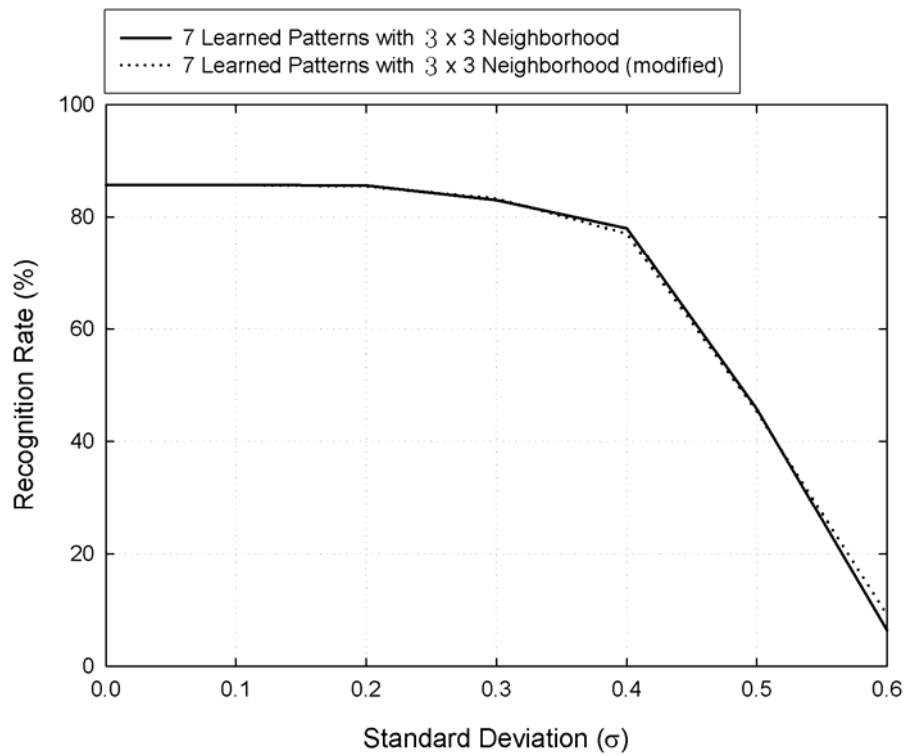
redundant correlations are not smaller than their mean any more. As a result, the ratio weights of smaller correlations can be depressed and the recognition rates are raised as shown in Fig. 3.15(a). As can be seen, the recognition rate of $r' = 5$ is much better than that of $r' = 3$. The reason is that the large-neighborhood of $r' = 5$ can gather more information between cells and hence, it is highly possible to retain larger correlations. By using the repeated proposed algorithm, the number of learned patterns is increased and it also shows that the use of large neighborhood templates can increase the number of the learned patterns than that of single neighborhood templates. In Fig. 3.15(b), it also shows larger neighborhood gives higher recognition rate. However, as shown in Fig. 3.15(c), when the single neighborhood template is used, the repeated algorithm gives no effects on the recognition rates. Hence, it can be inferred that the repeat of the algorithm is only suitable for the large-neighborhood RMCNN requiring



(a)



(b)



(c)

Fig. 3.15 The recognition rates of (a) 3×3 neighborhood and large neighborhood templates by repeating the operation of the proposed algorithm (marked with ‘modified’) where 7 patterns are learned. (b) large neighborhood templates $r' = 5$ and $r' = 7$ where 9 patterns are learned. (c) 3×3 neighborhood by the operation of the proposed algorithm and repeating the operation of the proposed algorithm where 7 patterns are learned.

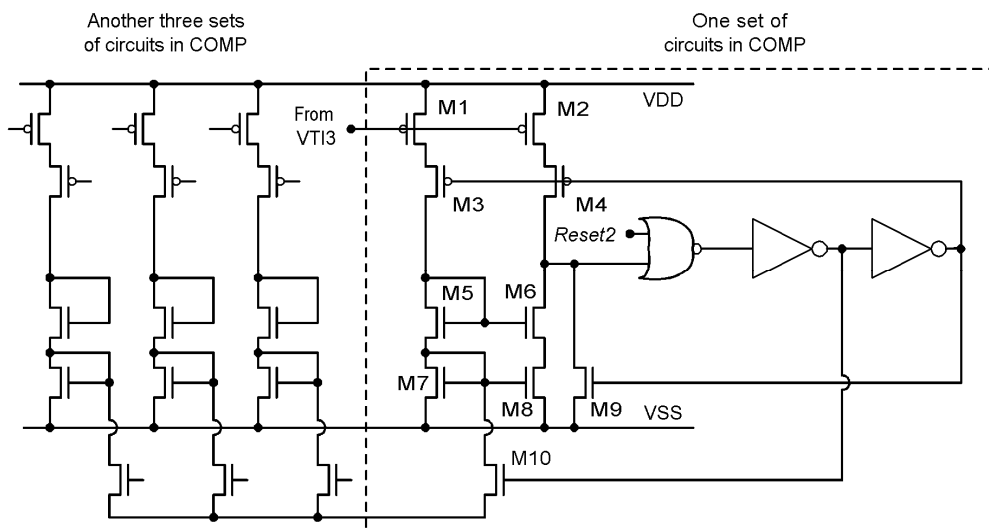


Fig. 3.16 The modified circuits of block COMP that can realize the repeated proposed algorithm.

no elapsed time is implemented. The modified circuits for repeating the operation of the proposed algorithm is also depicted in Fig. 3.16.

3.4 EXPERIMENTAL RESULTS

The architecture of a 9×9 non-self-feedback RMCNN not requiring elapsed time has been designed as shown in Fig. 3.17. The input patterns for learning and recognition are sent serially into 9×9 shift registers. The decoder can select the cells in the proposed RMCNN not requiring elapsed time to be read out in series. The controlling signals are listed in Table 3.2 with a controlling timing diagram shown in Fig. 3.18. The learning and recognition periods are controlled by signals *clk1* and *clk2*, respectively. Signal *Reset* is used to reset the charge on the capacitor *Cw*. Signal *newp* enables the shift registers and then, signal *DFF* can trigger the D-flip-flops in the shift registers to transfer the pixels of the input pattern in series. Signal *pin* generates the patterns which are sent into the neural network. Signals *Con_L* and *Con_G* trigger the local and global counters. The local counter counts the number of the currents which are larger than the mean current in the cell. The global counter generates the signals to control which comparative results in the cell should be counted by the local counter. Signal *noi* can

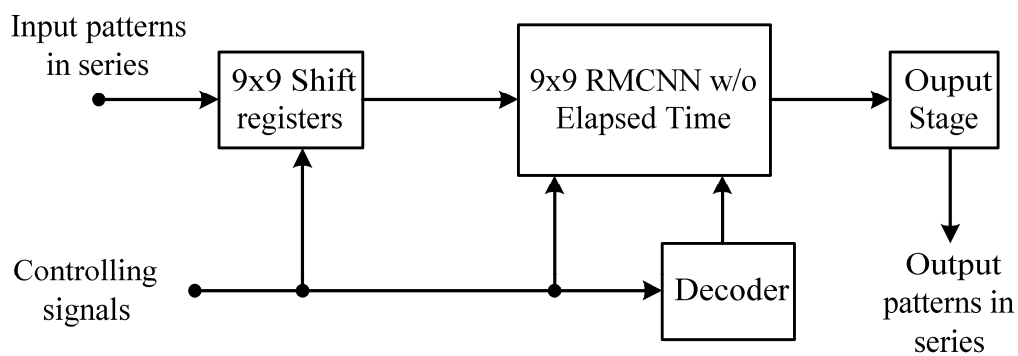


Fig. 3.17 The architecture of a 9×9 RMCNN without elapsed time chip.

Table 3.2 THE DESCRIPTION OF EACH CONTROL SIGNAL

Control signal	Description
<i>clk1</i>	High : Learning period starts Low : Learning period stops
<i>Reset</i>	High : Reset the capacitor in the block RM Low : Normal Operation
<i>DFF</i>	Trigger the D-flip-flop in shift register (negative triggered) used to stored the patterns.
<i>newp</i>	High : Enable the transfer of the shift register Low : Disable the transfer of the shift register
<i>pin</i>	High : The pattern in shift register is sent to neural cells
<i>Cou_L</i>	Trigger the block Counter_L in each cell
<i>Cou_G</i>	Trigger the global counter for controlling the blocks COMP and Counter_L
<i>clk2</i>	High : Recognition period starts Low : Recognition period stops
<i>noi</i>	High : Make the pattern in shift register noisy

introduce the noise into the input patterns. With such architecture, an RMCNN chip not requiring elapsed time has been designed and fabricated using TSMC 0.35- μm 2P4M mixed-signal technology. Fig. 3.19 shows the photograph of the fabricated chip of an RMCNN not requiring elapsed time.

During the learning period, the Chinese characters in Fig. 3.9 are learned. Because the noise cannot be programmed individually, only uniform noise can be added into the correct patterns. After the ratio weights are generated, these Chinese characters are sent again and combined with a controllable uniform noise from 0 to 0.5 as shown in Fig. 3.20 where the noise level is set

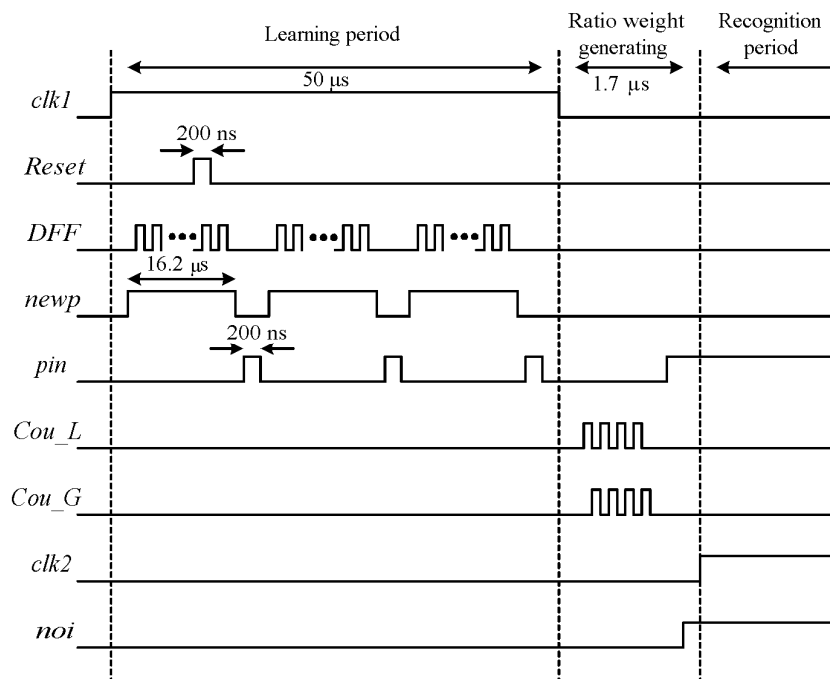


Fig. 3.18 The timing diagram of control signals.

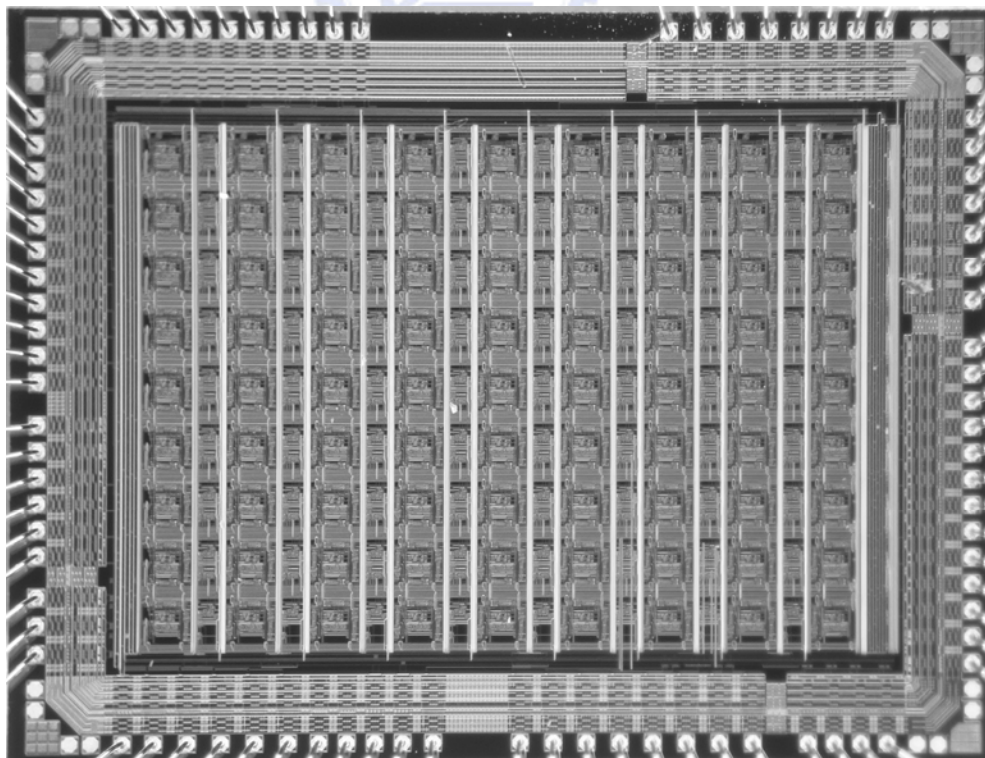


Fig. 3.19 The photograph of the RMCNN without elapsed time chip.



Fig. 3.20 The uniform noisy patterns for measurement.

0.5. Because the noise cannot be programmed by pixels, the noise in each pattern is uniform. For the first two Chinese characters, the correct patterns could be recognized. However, the last Chinese character is recognized unsuccessfully as shown in Fig. 3.21 where the uniform noise level is 0.25, and the output waveform is shown in Fig. 3.22. Channel 1 is the trigger signal which is tied to low during the readout period. Channel 2 is LSB of the decoder and channel 3 is the output waveform of the third pattern in Fig. 3.21. The output swing is between 0.2 V and 1.8 V and the output voltage is segmented into 256 gray levels. The gray level of 0.2 V is white and that of 1.8 V is black. There are four stable pixels at the gray level as the third pattern is recognized. To discuss the reason for this, the absolute weights of the post simulation at cell(4,4), which is recognized unsuccessfully in the third pattern, are listed in Table 3.3 where

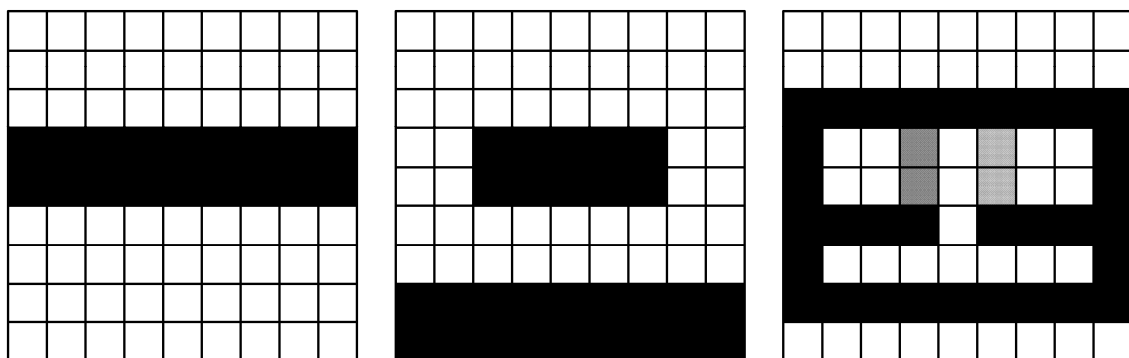


Fig. 3.21 Experimental results of recognized patterns in the recognition period after a set of patterns with noise level 0.25 are recognized.

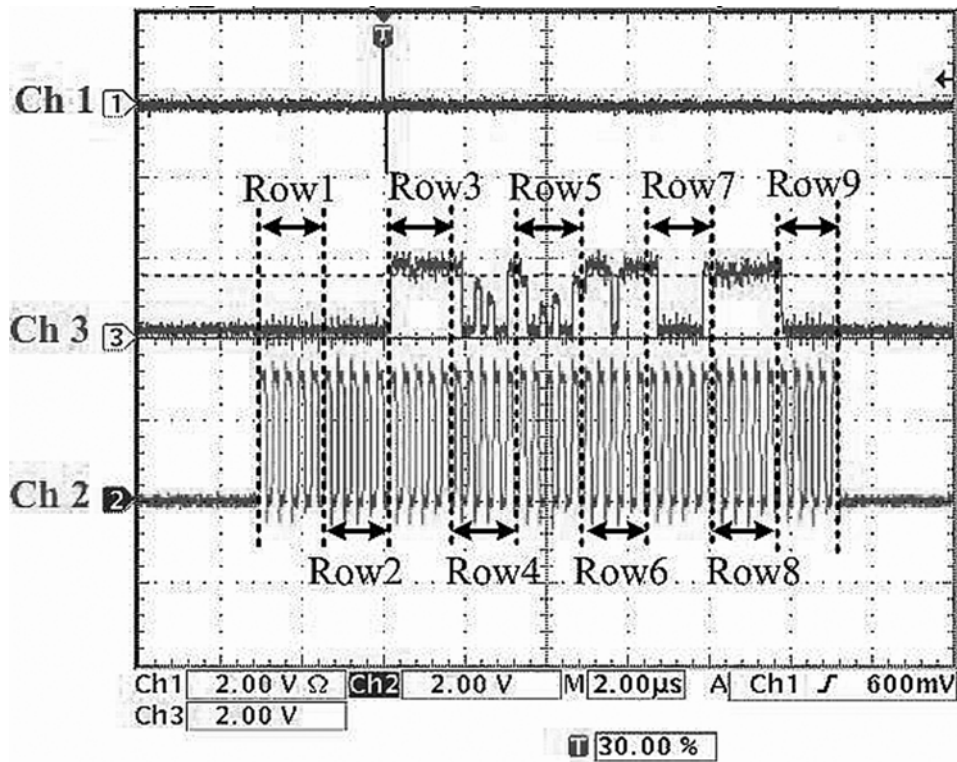


Fig. 3.22 Experimental output waveform of the third recognized pattern.

the simulation with Matlab and two post simulations with HSPICE in different conditions are compared. As can be seen, the ratio weights with Matlab and HSPICE (TT) are the same and the noisy patterns can be recognized correctly. However, with HSPICE (FS), an incorrect ratio weight is generated and leads to an unsuccessful result. Hence, even the third pattern with a smaller noise is recognized, the resultant pattern is still incorrect because the incorrect correlations are retained. When three patterns are learned, block W charges or discharges the capacitor C_w according to the input pixel of two neighboring cells. During this time, the device ME in Fig. 3.4 (a) of block VT12 is turned off. However, when a new pattern is sent into the chip after the former one is learned, device ME is turned on and the output current should be 0. However, there is still a small output current due to the asymmetric structure and the mismatch. Meanwhile, the input of the sign detector is connected to V_{ref} because device ME is turned on

Table 3.3 THE COMPARISON OF THE ABSOLUTE WEIGHTS A_{44} WITH MATLAB AND HSPICE IN DIFFERENT CONDITIONS

Simulator (Xcondition)	Absolute weight of cell(4,4)	Mean	ratio weight of cell(4,4)
Matlab	$A_{44} = \begin{bmatrix} 0 & 0.33 & 0 \\ 0.33 & 0 & 0.33 \\ 0 & 1 & 0 \end{bmatrix}$	0.5	$A_{44} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$
HSPICE (TT)	$A_{44} = \begin{bmatrix} 0 & 0.15 & 0 \\ 0.28 & 0 & 0.28 \\ 0 & 0.78 & 0 \end{bmatrix}$	0.3725	$A_{44} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$
HSPICE (FS)	$A_{44} = \begin{bmatrix} 0 & 0.116 & 0 \\ 0.41 & 0 & 0.41 \\ 0 & 0.70 & 0 \end{bmatrix}$	0.409	$A_{44} = \begin{bmatrix} 0 & 0 & 0 \\ 0.33 & 0 & 0.33 \\ 0 & 0.33 & 0 \end{bmatrix}$

and this makes it impossible to predict signal $Sign_Con$. As a result, the capacitor C_w is charged or discharged unpredictably by the small current when the learned patterns are transmitted to the 9×9 shift registers. Hence, the ideal absolute weights cannot be achieved.

To overcome the small output current from block VTI2, a new path can be inserted into block W as shown in Fig. 3.23. Only one of switches S_{tra} and S_{learn} is turned on and the other is turned off. As the learned patterns are transmitted to the shift register, switch S_{tra} is turned on. Hence, capacitor C_w would not be charged or discharged by the small current from block VTI2. Switch S_{learn} is turned on when the pattern in the shift register is sent to the neuron and can be learned or recognized correctly. Dummy load M_{dummy} is the same with M5. This can cause the current source M1-M4 to have a similar load and retain the current stable during switching.

The comparison between RMCNN [67] and RMCNN requiring no elapsed time is list in Table 3.4. The total chip area is $4560 \mu\text{m} \times 3900 \mu\text{m}$ and the area of a single cell is $400 \mu\text{m} \times$

250 μm . The total power consumption is 87 mW in operation with a supply voltage of 3 V and a system clock frequency of 10 MHz.

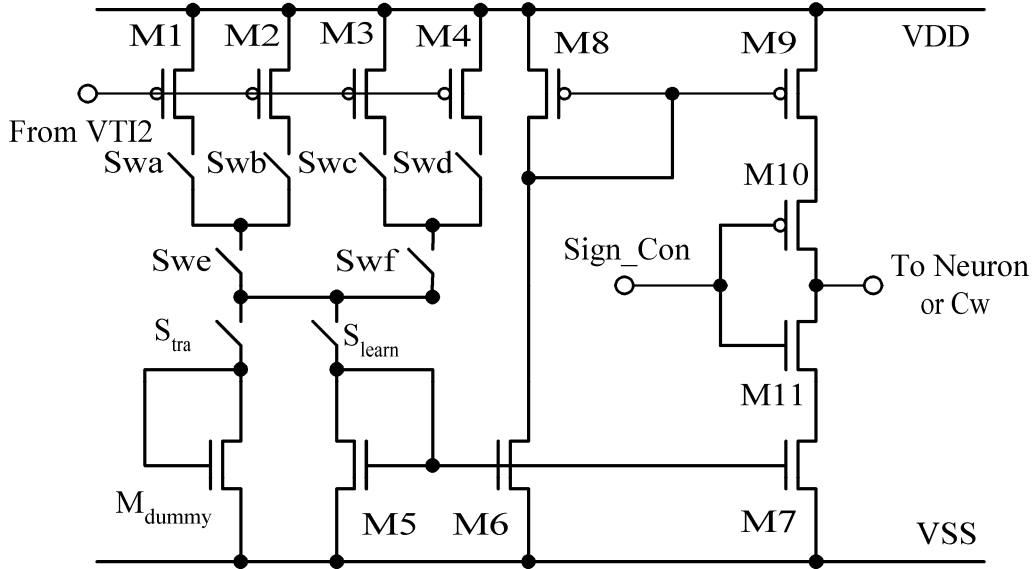


Fig. 3.23 The modified circuit of the block W.

TABLE 3.4 COMPARISON BETWEEN RMCNN AND RMCNN REQUIRING NO ELAPSED TIME

	RMCNN [67]	RMCNN requiring no elapsed time
Technology	0.35 μm 1P4M Mixed-Signal Process	0.35 μm 2P4M Mixed-Signal Process
Array Size	9 x 9	9 x 9
No. of RMs	144	144
Area of Single Pixel	350 μm x 350 μm	400 μm x 250 μm
Power Supply	3 V	3 V
Power Dissipation	120 mW	87 mW
Readout Time (of one pixel)	1 ms	80 ns
Weight Generating Time (Elapsed Time)	850 sec	1.7 μs
System Clock Frequency	N/A	10 MHz
Dynamic Range of State X_{ij} ($V_{X_{ij}} - V_{\text{ref}}$)	-0.8 ~ 0.8 V	-0.6 ~ 0.6 V

3.5 SUMMARY

In this chapter, a new algorithm of a RNCNN not requiring elapsed time has been proposed. In the proposed RMCNN, a new ratio weight generating method is also proposed. The use of this method avoids a long period of elapsed time when the ratio weights are generated. By using RMCNN requiring no elapsed time, 6 patterns can be learned and recognized. In this chapter, the large-neighborhood RMCNNs of $r' = 3$, $r' = 5$, and $r' = 7$ are also simulated. The results suggest that the proposed algorithm to compare the correlations and their mean should be repeated when the large-neighborhood templates are used. RMCNN requiring no elapsed time is modified to be suitable for large-neighborhood application. It also suggests that RMCNN with larger neighborhood templates can increase the recognition rates or the number of learned patterns. However, the efficiency to increase r' become lower when r' is large.

An experimental chip of 9x9 RMCNN not requiring elapsed time has been implemented and fabricated using TSMC 0.35- μm CMOS 2P4M technology. The weight generating time is reduced to 1.7 ms while the elapsed time required by RMCNN is more than 800 seconds.

Further applications of the proposed RMCNN not requiring elapsed time will be developed in the future.

CHAPTER 4

THE ANALYSIS OF THE RECURSIVE LEARNING RMCNN

4.1 INTRODUCTION

By using the architecture of cellular nonlinear (neural) network (CNN) which was proposed by Chua and Yang in 1988 [6]-[7], [40], the concept of RMCNN was first brought up by C. Y. Wu and J. F. Lan in 1995 [65]-[67]. RMCNN works by a set of learned space-variant templates according to the correlation of each learned patterns. With the ratio memory and a long period of retrieving time (elapsed time), the common characteristic can be enhanced. Moreover, in the past study, the algorithm of RMCNN without elapsed time is proposed and also discussed in Chap. 3 where the templates are generated by comparing the correlations with the mean of those four correlations around any one cell instead of being generated with a long elapsed time.

However, in the past study, the discussion on how the generated templates affect the recognition rate is not mentioned. Hence, in this work, the Gaussian noise probability density is considered and a simple situation of one pixel with a generated template is discussed. In the situation, the asymmetric probability of the recognized pixel is considered and it causes the asymmetric probability density of the output. Thus, the result shows the necessity of the templates Z to depress the error rate. To gather the information of the threshold, the recursive

learning technique [145] is applied. The recursive learning in [145] is used for the background and foreground modeling as following:

$$\theta_t(\bullet) = (1 - \beta_t) \cdot \theta_{t-1}(\bullet) + \alpha_t \cdot H_{\Delta}[x_t; \theta_{t-1}(\bullet)] \quad (4.1)$$

where $\theta_t(\bullet)$ is the probability density function of each pixel at time t and updated by the local kernel $H_{\Delta}[x_t; \theta_{t-1}(\bullet)]$, which is the learned target, and α_t and β_t are the learning rate and forgetting rate. The learning rate α_t is usually equal to $1/t$ where the forgetting rate β_t is equal to $1 - G \cdot \alpha_t$ and G is a coefficient smaller than 1. By using the recursive learning technique, the probability density function of error function can be obtained and recovered by the template Z . As a result, it can correct the asymmetric black and white probabilities of learned patterns and also can be demonstrated by the results of simulations. By using the proposed method, the simulations with different algorithm are made and compared.

In section 4.2, the mathematical analysis of one template generated by an RMCNN is studied and the operating procedure of the recursive learning RMCNN is illustrated. In section 4.3, the simulation results are compared and discussed. Finally, the conclusion is given.

4.2 MATHEMATICAL ANALYSIS

The generated templates in any type of RMCNN are diamond templates and can be written as

$$A_{ijkl} = \begin{bmatrix} 0 & a_{ij(i-1)j} & 0 \\ a_{iji(j-1)} & 0 & a_{iji(j+1)} \\ 0 & a_{ij(i+1)j} & 0 \end{bmatrix} \quad (4.2)$$

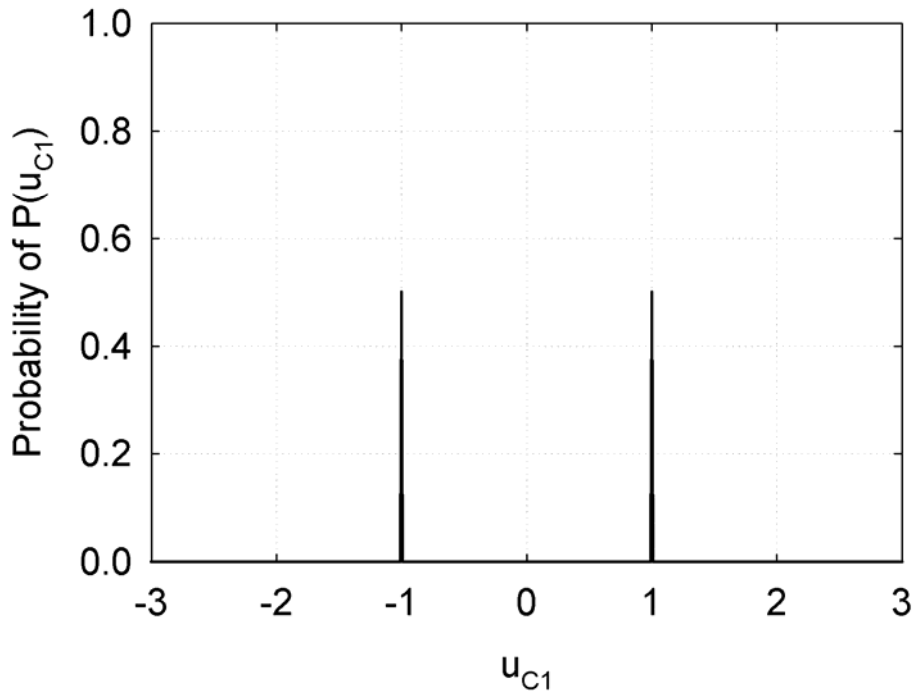


Fig. 4.1 The probability of the input u_{C1} .

except the self-feedback term. Each coefficient in RMCNN is realized by the equation (3.6) and in RMCNN requiring no elapsed time is by the equations (3.7)-(3.8). However, for simplification, only one coefficient is considered and other coefficients are 0. Because the summation of the all absolute coefficients is 1, the existent coefficient may be ± 1 . Here, we choose the coefficient to 1 for example, and assume that the pixel C1 always has a correct input and has a probability of 0.5 for black and white colors as shown in Fig. 4.1. Meanwhile, it is assume that the pixel C2 has a noisy input with Gaussian noise and with asymmetric probabilities of 0.4 and 0.6 for white and black colors, respectively, as demonstrated in Fig. 4.2. After the recognition, the state x_{C1} can be shown in Fig. 4.3. As can be seen, the asymmetric probabilities of black and white colors make the probability density of the state x_{C1} . If RMCNN requiring no elapsed time with a tolerance of 50% is taken into account, the error rate of the

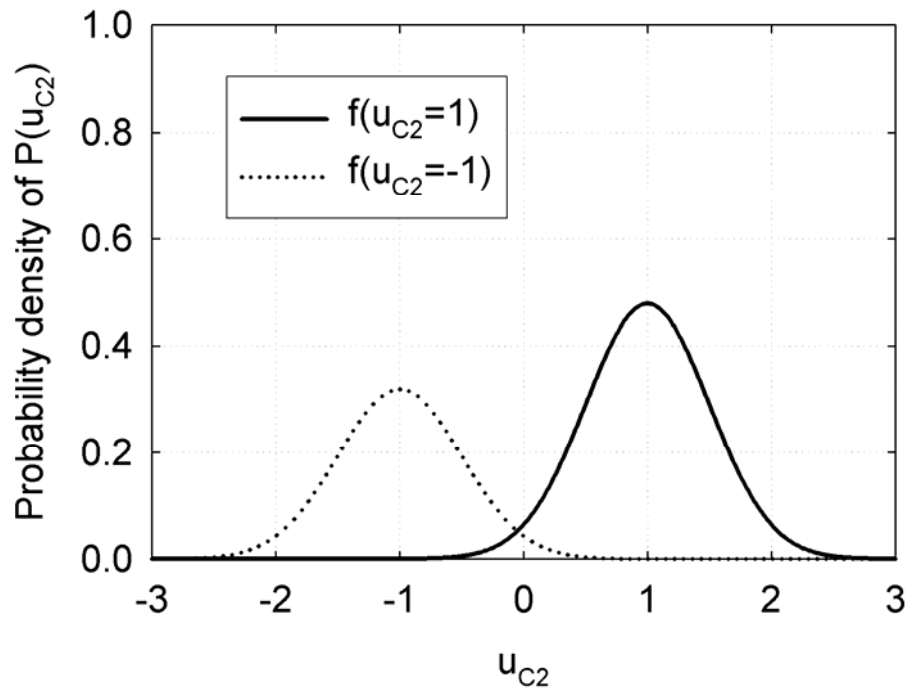


Fig. 4.2 The probability density of the input u_{C2} .

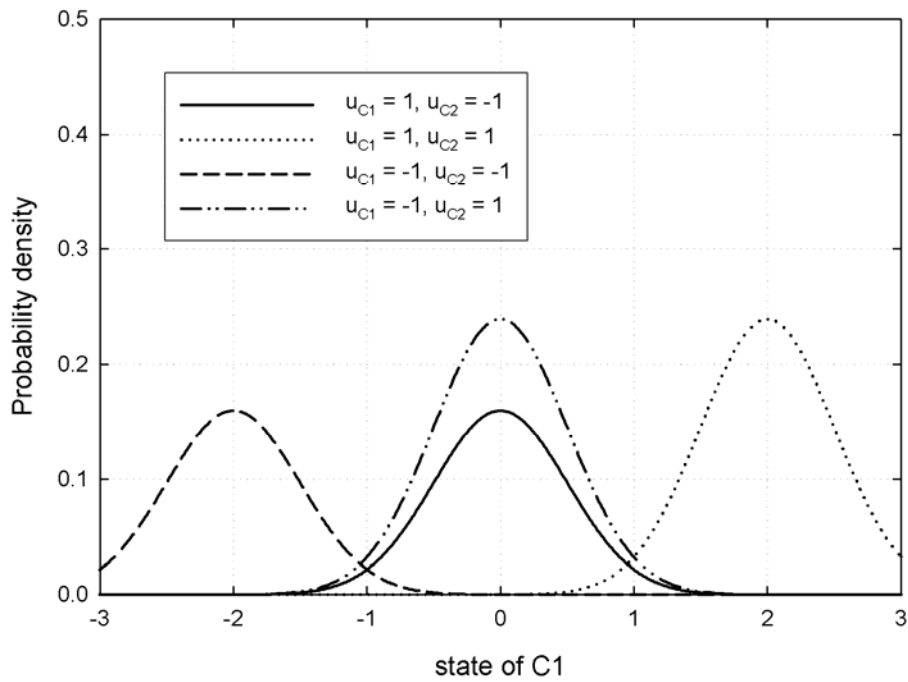
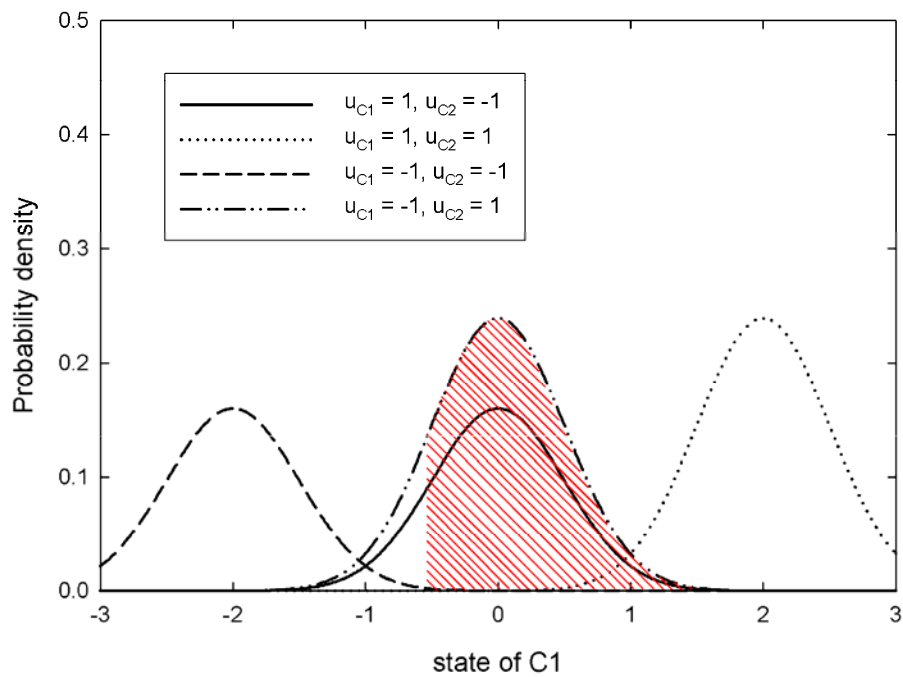


Fig. 4.3 The probability density of the state x_{C1} after recognition.

error can be depressed. With Matlab simulation, if there is a threshold value of -1, the error rate pixel C1 can be calculated by the integration of the shadow in Fig. 4.4(a) and (b). However, it can be observed that if the graph is shifted, the can be reduced from 0.4201 to 0.2935.

It can be proved that the error rate can be improved by the threshold term but it is dependent on the probability of the input signals. Hence, a recursive learning technique is applied to learn the error rate. The procedure is shown in Fig. 4.5 where the recursive learning is behind the generating of ratio weights because the recursive learning algorithm is used to learn the error rate after recognition in 5 iterations. After 5 iterations, the deviation of $THR(i,j,k)$ is calculated. If the deviation is smaller than the constrain δ , the recursive learning stops. Based on the recursive learning in (4.1), a recursive learning of error rate probability is constructed as shown in Fig. 4.6 where $THR(i,j)$ where is equal to $THR(i,j,k)$ after k iterations is the threshold value of



(a)

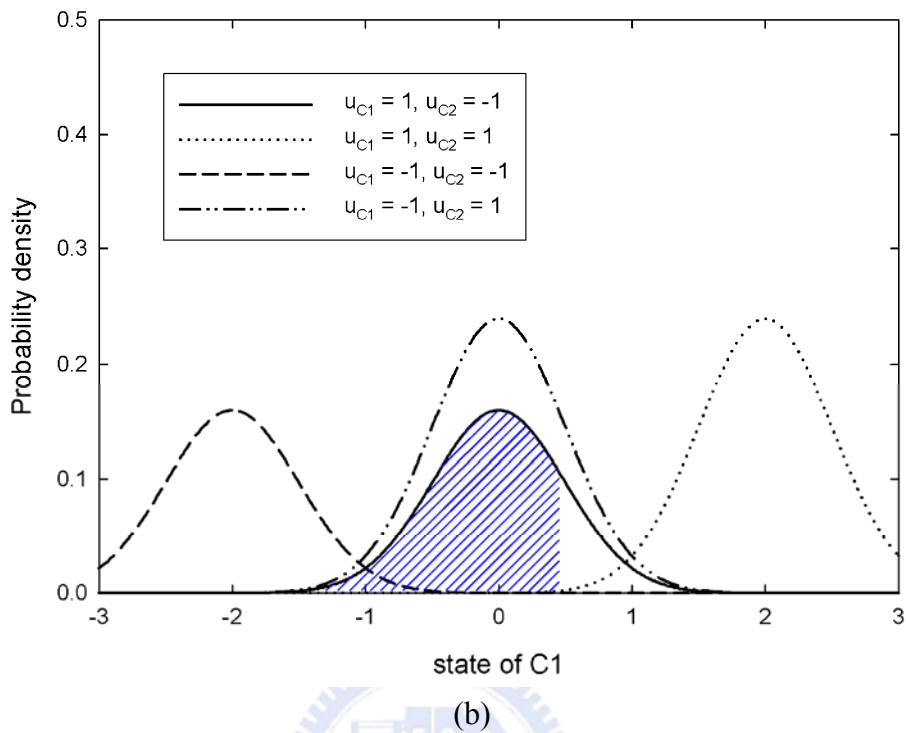


Fig. 4.4 The error rates produced by the shadow part when the output of the pixel C1 should be (a) 1 and (b) -1.

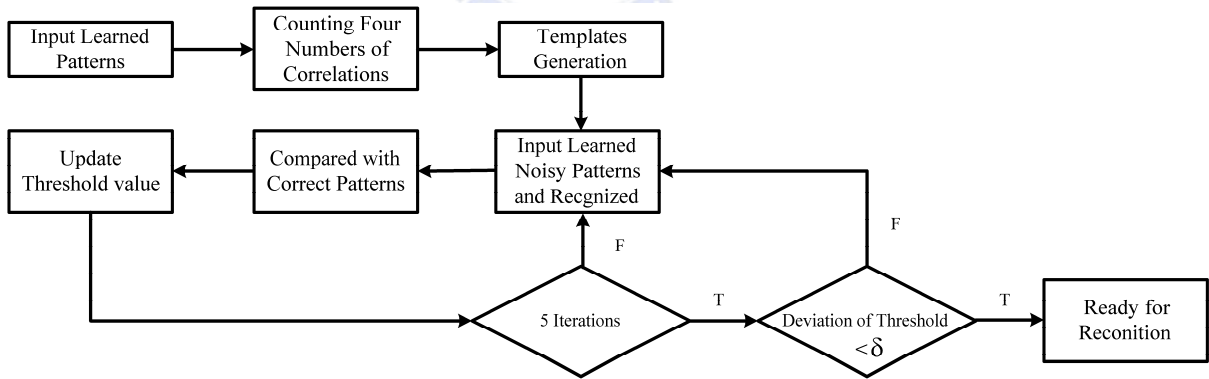


Fig. 4.5 The procedure of the recursive learning algorithm.

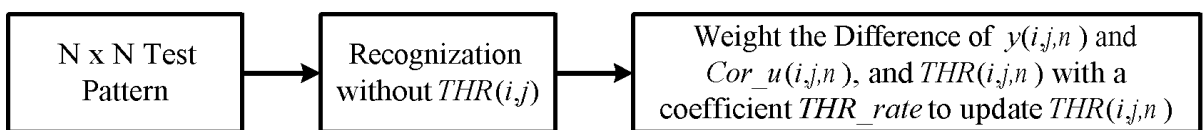


Fig. 4.6 The recursive learning of $THR(i,j)$ in n^{th} iteration.

cell(i,j), THR_rate is the learning rate, $Cor_u(i,j,n)$ and $y(i,j,n)$ are the correct pattern and recognized output, respectively, of cell(i,j) in the n^{th} iterations. In the recursive learning period, the difference of $Cor_u(i,j,n)$ and $y(i,j,n)$ is calculated in each iteration and update the error probability density by $[Cor_u(i,j,n) - y(i,j,n)]$. Based on the equation (4.1), the recursive learning of the templates can be written as:

$$THR(i, j, n) = [Cor_u(i, j, n) - y(i, j, n)] \times \frac{1}{THR_rate} + THR(i, j, n-1) \times \left(1 - \frac{1}{THR_rate}\right). \quad (4.2)$$

where the learning rate α_t is THR_rate , the forgetting rate is $1 - G \cdot \alpha_t$ and the coefficient G is chosen to be 1. Because even the chosen G is a coefficient smaller than 1, the equation can be normalized by a factor. With equation (4.2), the average of $[Cor_u(i,j,n) - y(i,j,n)]$ in k iterations since the equation (4.2) can be derived as:

$$THR(i, j, k) = \frac{\sum_{n=1}^k Cor_u(i, j, n) - y(i, j, n)}{k}. \quad (4.3)$$

Hence, the recursive learning is the learning of the average distance that the output $y(i,j,n)$ is away from the correct pattern $Cor_u(i,j,n)$.

4.3 SIMULATION RESULTS

The simulation is made by using Matlab simulator. As shown in Fig. 4.7, 7 patterns are learned by using RMCNN requiring no elapsed time without and with recursive learning. As can be seen in Fig. 4.7, the recursive learning can raise the recognition rate and improve the learning ability of RMCNN requiring no elapsed time. The simulations are made with recursive learning of different constrains. It can be found that the recognition rate can be raised after the

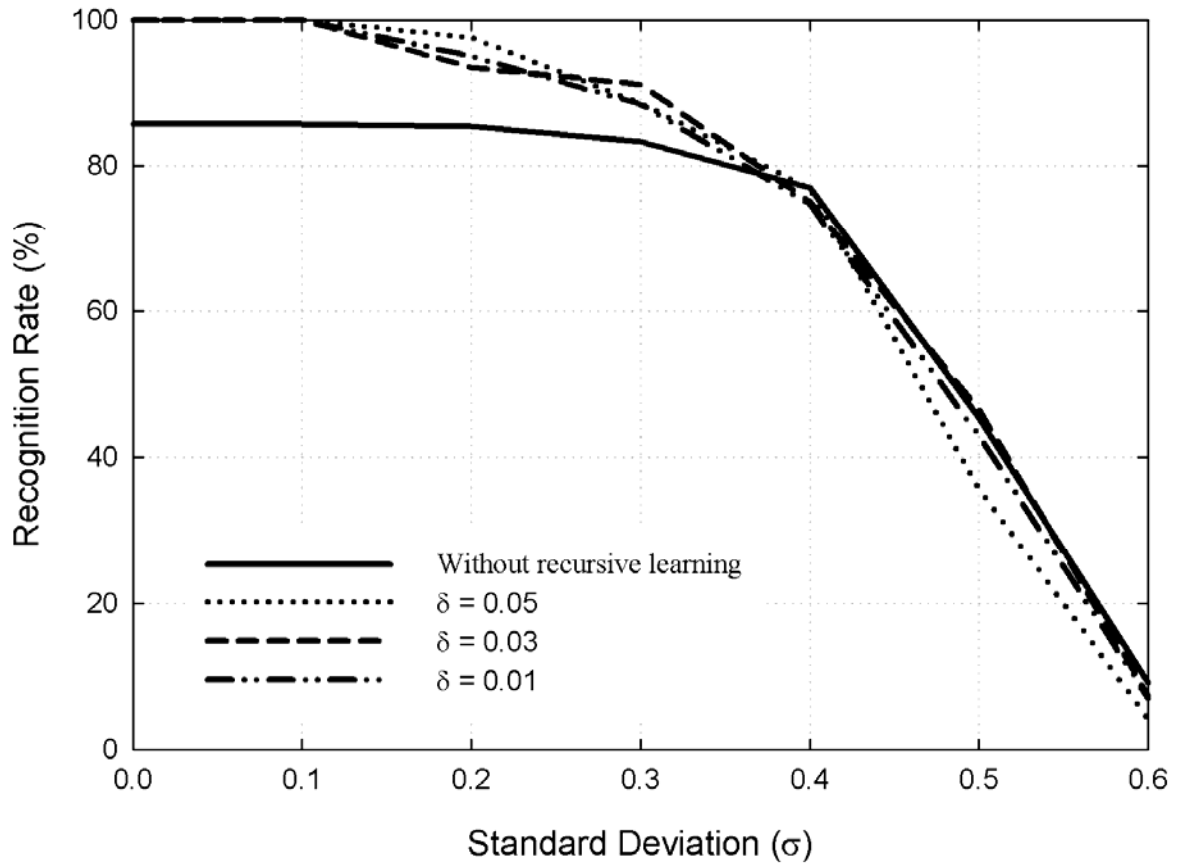


Fig. 4.7 The recognition rates of RMCNN requiring no elapsed time without and with recursive learning of constrains 0.01, 0.03, and 0.05 where 7 patterns are learned

recursive learning of different constrains 0.01, 0.03, and 0.05. The recognition rates of constrains 0.01 and 0.03 are almost the same when 7 patterns are learned but the recognition rate of constrain 0.05 is a little lower. The required iterations are listed in Table 4.1. The required iterations of constrain 0.01 is more than those of constrain 0.03 while the required iterations of constrain 0.5 is a little fewer. Hence, constrain $\delta = 0.03$ is chosen. It also can be seen that when the standard deviation of the noise is large, the required iterations also become more. The learned templates \mathbf{Z} are also shown in Fig. 4.8.

The recognition rates where 6 patterns and 8 patterns are learned with or without recursive learning are shown in Fig. 4.9. The constrain δ is set to 0.03 and the required iterations are listed

Table 4.1 THE REQUIRED ITERATIONS TO FIT THE CONSTRAINS WHERE 7 PATTERNS ARE LEARNED

Gaussian Noise (σ)	0.01	0.03	0.05
0	10	10	10
0.1	10	10	10
0.2	20	15	10
0.3	20	15	10
0.4	25	15	10
0.5	30	15	15
0.6	60	20	15

-0.14	-0.03	-0.04	-0.04	-0.01	-0.04	-0.09	-0.06	-0.15	-0.05	-0.02	-0.06	-0.07	-0.05	-0.03	-0.01
-0.08	-0.12	-0.15	-0.05	-0.01	-0.08	-0.01	-0.07	-0.63	-0.01	-0.22	-0.04	-0.04	-0.19	-0.11	-0.05
-0.26	-0.12	-0.01	-0.15	-0.13	-0.03	-0.05	-0.01	-0.03	-0.02	-0.03	-0.34	-0.07	-0.16	-0.04	-0.04
-0.15	-0.13	-0.09	-0.41	-0.44	-0.06	0.03	0.04	0.00	0.04	0.03	-0.08	-0.05	-0.01	-0.10	-0.09
-0.06	-0.03	-0.01	-0.03	-0.01	-0.09	0.01	0.04	0.06	0.36	0.00	-0.16	0.03	-0.04	0.00	-0.08
-0.05	-0.08	-0.01	0.00	0.04	0.00	-0.03	0.92	0.29	0.11	1.06	0.00	-0.05	-0.03	0.06	-0.12
-0.16	0.01	-0.03	-0.05	-0.03	-0.06	-0.11	0.00	0.00	0.02	0.03	-0.03	0.00	0.03	0.02	-0.02
-0.03	-0.02	-0.04	0.03	0.07	-0.06	-0.10	0.05	0.01	0.07	0.05	0.02	0.02	0.02	-0.04	-0.03
-0.04	-0.02	-0.03	0.00	-0.03	0.07	-0.16	0.07	0.09	0.01	0.02	-0.03	0.09	-0.01	0.00	-0.03
-0.05	-0.01	-0.04	-0.03	-0.10	-0.04	-0.48	0.11	0.04	0.08	0.03	-0.28	0.01	-0.02	-0.01	-0.02
-0.02	-0.03	0.01	-0.01	-0.25	-0.18	-0.15	-0.03	0.01	0.04	-0.02	-0.33	-0.06	0.00	-0.02	-0.01
-0.05	-0.02	-0.01	-0.04	-0.02	-0.04	-0.11	-0.03	0.05	0.01	0.01	-0.39	-0.02	-0.15	-0.06	-0.02
-0.04	-0.02	0.00	0.01	-0.04	-0.05	-0.06	0.00	0.10	0.04	0.01	-0.04	-0.07	0.36	-0.10	-0.01
-0.06	0.12	0.02	0.11	0.03	0.13	0.30	-0.09	0.04	0.03	0.06	0.00	0.03	0.01	0.21	-0.01
-0.09	0.09	0.05	0.00	-0.11	0.09	0.05	0.04	0.07	0.10	0.03	-0.02	0.03	0.05	0.06	0.00
-0.02	0.01	-0.03	0.02	0.13	0.14	0.01	0.01	0.08	0.04	0.18	0.03	0.02	0.07	-0.01	0.02
-0.04	-0.03	-0.08	-0.04	-0.02	-0.04	-0.02	-0.05	0.21	0.23	-0.07	-0.10	-0.01	-0.20	-0.06	-0.02
-0.12	-0.05	-0.04	-0.06	-0.04	-0.04	-0.04	-0.04	-0.05	-0.11	-0.03	-0.06	-0.01	-0.05	-0.04	-0.14

Fig. 4.8 The learned threshold Z where constrain δ is 0.03 and the standard deviation of Gaussian noise is 0.6.

in Table 4.2. The recognition rates where 6 patterns are learned with and without recursive learning are almost the same. However, when 8 patterns are learned, the recognition rate can be raised with recursive learning. As a result, RMCNN requiring no elapsed time can learned 6 patterns while it can learned 8 patterns by using recursive learning with a constrain of $\delta = 0.03$.

4.4 SUMMARY AND FUTURE WORK

In this chapter, with the concept of RMCNN, the effect of Gaussian noise has been discussed. According to the analyzing results, the threshold is required to decrease the error rate

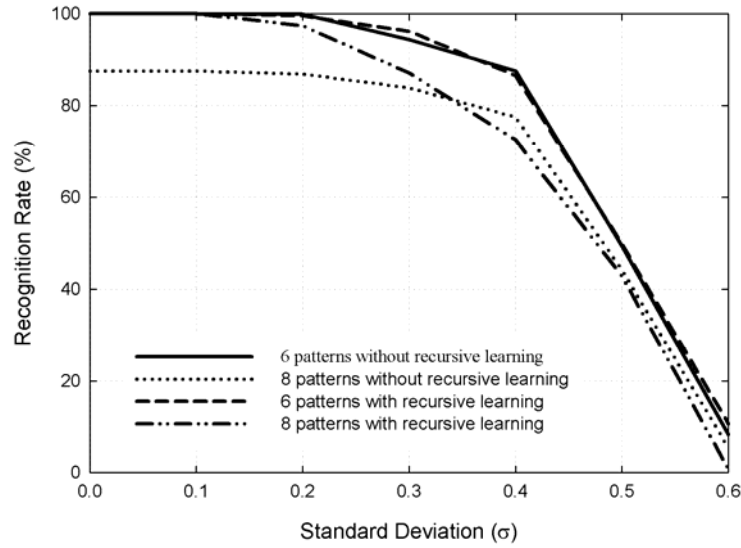


Fig. 4.9 The recognition rates where 6 patterns and 8 patterns are learned with and without recursive learning.

Table 4.2 THE REQUIRED ITERATIONS TO FIT THE CONSTRAINS $\delta = 0.03$ WHERE 6 AND 8 PATTERNS ARE LEARNED

Gaussian Noise (s)	6 patterns	8 patterns
0	5	10
0.1	10	10
0.2	10	10
0.3	10	10
0.4	10	10
0.5	15	15

because of the asymmetric probability densities of the inputs. Hence, a recursive learning technique is proposed to minimize the error rate due to this factor. With the recursive learning technique, the probability density of errors is gathered during k iterations. The deviation of templates Z is calculated per 5 iterations. As the deviation is smaller than the constrain δ , the recursive learning stops. With the proposed recursive learning, the comparison of different

applied algorithm is made by using Matlab simulations and the recognition rate and learned patterns indeed can be improved with constrain $\delta = 0.03$.

However, in this chapter only the mathematic analysis is discussed and the statistic simulations are made. Hence, further research on the circuit design of the recursive learning RMCNN needs to be conducted.



CHAPTER 5

CONCLUSIONS AND FUTURE WORK

5.1 CONCLUSIONS

In this dissertation, an LNCNN and new types of RMCNN have been proposed. In 3×3 neighborhood CNNs, their local connectivity is easy to be implemented in a VLSI design. However, 3×3 neighborhood CNNs limit the realizable functions because they only generate 3×3 templates. When a large neighborhood function is realized, 3×3 neighborhood CNNs cannot realize the function directly. Some of 5×5 templates can be decomposed into several 3×3 neighborhood templates. It takes more operation time and more power consumption to carry out these 3×3 neighborhood templates in a task. By using the proposed LNCNN, the 5×5 templates can be approximated with the diamond templates and several functions like diffusion, de-blurring, and Muller-Lyer illusion has been verified with Matlab simulation. In the kernel unit of the proposed LNCNN, only the neighboring cells are connected to each other. The propagating connections are used to deliver the stimulus from one cell to further cells expect for the neighboring cells. Thus, the proposed LNCNN can realize large-neighborhood diamond-shaped templates. By using the propagating connections, complicated wire connections to farther cells can be avoided. In the proposed LNCNN, the analog memory is also used to store the non-recurrent term produced by templates \mathbf{B} and \mathbf{Z} . As well, the simple

current-mode circuits of the synapses and neuron cells are implemented by using current mirrors. The circuits based on current mirror structure makes the implementation simple and accurate current mirrors are not required in this design. Each cell can be implemented more compactly and the diamond templates with constraints can be realized easily. An LNCNN chip of 20×20 array has been fabricated. By using the LNCNN chip, the Muller-Lyer, which is the function of 5×5 templates, has been successfully verified. The LN function has been successfully verified by using the LNCNN chip with a power consumption of 0.7 mW on standby and 18 mW in operation with a system clock frequency of 20 MHz.

In this dissertation, RMCNN without elapsed time has also been proposed. The space-variant templates are learned and generated according to different local characteristics. Same with RMCNN of former researches, RMCNN requiring no elapsed time stores the correlations between cells and their neighboring cells. The elapsed time is taken off to avoid the long weight generating time and to remove the effects of uncertain leakage. The device multiplier-divider is also replaced with a comparator and a counter and this simplifies the design. The correlations are compared with their mean, and the correlations, which are larger than the mean, are counted. As a result, all the local correlations are compared with local means. The local characteristics in different positions of the learned patterns can be enhanced due to the local property. To verify the proposed algorithm, an RMCNN without elapsed time chip of 9×9 array is designed, and the uniform noisy patterns have also been tested and discussed. With the modified circuit, the RMCNN without elapsed time chip can recognize the patterns successfully. The total chip area is $4560 \mu\text{m} \times 3900 \mu\text{m}$ and the area of a single cell is $400 \mu\text{m} \times 250 \mu\text{m}$. The total power consumption is 87 mW in operation with a supply voltage of 3 V and a system clock frequency of 10 MHz.

Finally, a recursive learning RMCNN is proposed. The statistic and probabilistic model is not concerned before when the image is recognized. Hence, in this dissertation, a Gaussian noise model is concerned and discussed when an assumed template is give. According to the analysis, the decision is not located at an optimum point. Therefore, the recursive learning of the threshold values is applied to RMCNN for further improvement. By using the recursive learning, the error probability density of $[Cor_u(i,j,n) - y(i,j,n)]$ is gathered. When the threshold is applied with the mean of the term $[Cor_u(i,j,n) - y(i,j,n)]$, the decision points can be located at an optimum points. As a result, the recognition rate and the number of learned patterns can be increased.

5.2 FUTURE WORK

In this dissertation, an LNCNN chip has been fabricated and verified successfully. However, the applications of LNCNN are few because there are few studies on LNCNN due to the lack of LNCNN hardwares. Hence, with the proposed LNCNN structure and hardware, many researches on LNCNN templates and phenomenon can be studied and verified. Furthermore, because the simple circuits are used in the proposed LNCNN chip for small area and power consumption, the linearity of the templates is not the first priority of our consideration. Hence, the linearity of the circuits can be further modified to get a more precise control on the templates. Meanwhile, the goal of the LNCNN chip proposed in this dissertation is to realize the core of the LNCNNUM. In the next phase, it is anxious to achieve an LNCNNUM chip for many applications of LNCNN. Moreover, the applications of the diamond templates and how to transfer the 5×5 templates into diamond templates are also interesting researches. The tolerance of the diamond templates will be analyzed to generate a more robust template.

Furthermore, an RMCNN without elapsed time is also presented. In the structure of RMCNN, the correlations are stored on the analog memories, that is, the capacitors. Although the analog design is an intuitional method, it is also possible to operate the RMCNN in digitalized mode or mixed-mode structure. Under analog mode, the operation is easier and faster. However, under digital mode, it is more precise and more economic in power consumption. Hence, how to design a most proper structure is the main target in the next generation. Moreover, the learning of the large-neighborhood templates can also be applied on RMCNN. The effects of the large-neighborhood templates could be analyzed and how to implement the space-variant templates on RMCNN chip is a challenging topic.

As to the recursive learning RMCNN, the templates \mathbf{Z} are learned recursively. With the simulations, it is proved that the recognition rates can be improved as an RMCNN structure is used. However, per 5 iterations, the deviation of the learned templates \mathbf{Z} is calculated and the recursive learning stops when the deviation is smaller than the constrain δ . The mathematical model and derivation will be further studied in the future. Based on the proposed algorithm, a recursive learning RMCNN chip will also be designed and implemented in 0.18 μm or better CMOS technology. Further research on the efficiency of the learning templates \mathbf{Z} will be concerned and integrated.

Finally, the integration of RMCNN and LNCNN can make the whole chip powerful. RMCNN is applied on learning where LNCNN is used for controlling and computing. As a machine with RMCNN and LNCNN contains memories, controllable instructions, and learnable abilities, it may achieve an artificial intelligence system with a proper design and controlling codes.

REFERENCES

- [1] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities" in *Proc. Natl. Acad. Sci. U.S.A.*, vol. 79, pp. 2554-2558, 1982.
- [2] J. J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons," in *Proc. Natl. Acad. Sci. U.S.A.*, vol. 81, pp. 3088-3092, 1984.
- [3] J. J. Hopfield and D. W. Tank, "'Neural' computation of decisions optimization problems," *Biological Cybern.*, vol. 52, pp. 141-152, 1985.
- [4] J. J. Hopfield and D. W. Tank, "'Computing with Neural Circuits: A Model," *Science*, vol. 233, pp. 625-633, August 1986.
- [5] J. J. Hopfield and D. W. Tank, "'Collective computation with continuous variables," in *Disordered Systems and Biological Organization*, Springer-Verlag, 1986.
- [6] L. Chua and L. Yang, "Cellular neural network: Theory," *IEEE Trans. Circuits Syst.*, vol. 35, pp. 1257-1272, Oct. 1988.
- [7] L. Chua and L. Yang, "Cellular neural network: Applications," *IEEE Trans. Circuits Syst.*, vol. 35, pp. 1273-1290, Oct. 1988.
- [8] L. Chua and T. Roska, "The CNN paradigm," *IEEE Trans. Circuits Syst. I*, vol. 40, pp. 147-156, Mar. 1993.
- [9] J. A. Feldman and D. H. Ballard, "Connectionist Models and Their Properties," *Cognitive Science*, vol. 6, pp. 205-254, 1982.
- [10] D. W. Tank and J. J. Hopfield, "Simple neural optimization networks: An A/D converter, signal decision circuits, and a linear programming circuit," *IEEE Trans. Circuits Syst.*, vol. 33, pp. 533-544, May 1986.
- [11] R. P. Lippman, "An Introduction to Computing with Neural Nets," *IEEE ASSP Magazine*, pp. 4-22, 1987.
- [12] I. Aleksander, "Microcircuit learning nets: Hamming-distance behaviour," *Electronics Letters*, vol. 6, pp. 134-136, Mar. 1970.

- [13] A. Dembo, "On the capacity of associative memories with linear threshold functions," *IEEE Transactions on information Theory*, vol 35, pp. 709-720, Jul. 1989.
- [14] P. Houselander and J. T. Taylor, "Improving the Hamming binary associative memory," *Electronics Letters*, vol. 26, pp. 705-707, May 1990.
- [15] D. L. P. Aitken, J. M. Bishop, R. J. Mitchell, and S. E. Pepper, "Pattern separation in digital learning net," *Electronics Letters*, vol. 25, pp. 685-686, May 1989.
- [16] D. E. Rumelhart, B. Widrow, and M. A. Lehr, "The basic ideas in neural networks," *Communications of the ACM*, vol. 37, pp. 87-92, Mar. 1994.
- [17] J. J. Hopfield, "Artificial neural networks," *IEEE Circuits and Devices Magazine*, vol. 4, pp. 3-10, Sept. 1988.
- [18] W. S. McCullock and W. Pitts, "A Logical Calculus of the Ideas Imminent in Nervous Activity," *Bulletin of Mathematical Biophysics*, vol. 5, pp. 115-133, 1943.
- [19] D. O. Hebb, *The Organization of Behavior: A Neuropsychological Theory*, John Wiley & Sons, New York, 1949.
- [20] F. Rosenblatt, *Principles of Neurodynamics*, New York, Spartan Books, 1959.
- [21] B. Widrow and M. E. Hoff, "Adaptive Switching Circuits," *1960 IRE WESCON Conv. Record*, Part 4, pp. 96-104, Aug. 1960.
- [22] D. E. Rumelhart and J. L. McClelland, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, MIT Press, 1986.
- [23] T. Sejnowski and C. R. Rosenberg, "NETtalk: A Parallel Network That Learns to Read Aloud," *Johns Hopkins Univ. Technical Report JHU/EECS-86/01*, 1986.
- [24] S. Grossberg, *The Adaptive Brain I: Cognition, Learning, Reinforcement, and Rhythm*, Elsevier/North-Holland, Amsterdam, 1986.
- [25] S. Grossberg, *The Adaptive Brain II: Vision, Speech, Language, and Motor Control*, Elsevier/North-Holland, Amsterdam, 1986.
- [26] G. E. Hinton and T. J. Sejnowski, "Learning and relearning in Boltzmann machines", in *Parallel Distributed Processing: Exploration in the Microstructure of Cognition*, vol. 1: Foundations, D. E. Rumerhart, J. L. McClelland, and the PDP research group, pp. 282-317, MIT Press, Cambridge, MA., 1986.
- [27] M. A. Cohen and S. Grossberg, "Absolute stability of global pattern formation and parallel memory storage by competitive neural networks," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 13, pp. 815-826, 1983.

- [28] P. Smolensky, "Information Processing in Dynamical Systems: Foundations of Harmony Theory," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, D.E. Rumelhart and J.L. McClelland, eds., vol. 1, 1986.
- [29] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Internal Representations by Error Propagation," *Parallel and Distributed Processing: Exploration in the Microstructure of Cognition*, Vol. 1, D. Rumelhart and J. McClelland (Eds.), MIT Press, Cambridge, Massachusetts, 1986, pp. 318-362.
- [30] L. V. Fausett, *Fundamentals of Neural Networks: Architectures, Algorithms And Applications*, Prentice-Hall, Englewood Cliffs, New Jersey, 1994.
- [31] S. Haykin, *Neural Networks: A Comprehensive Foundation*, Prentice Hall, Upper Saddle River, New Jersey, 1999.
- [32] F. Ham and I. Kostanic, *Principles of Neurocomputing for Science and Engineering* McGraw Hill, Nw York, 2001.
- [33] R. P. Lippmann, "An Introduction to Computing with Neural Network," *IEEE Acoustic, Speech, and Signal Processing Magazine*, vol. 61, pp. 4-22, Apr. 1987.
- [34] M. Minsky and S. Papert, *Perceptrons: An introduction to computational geometry*, MIT Press, Cambridge, MA, 1988
- [35] E. R. Kandel and J. H. Schwartz, *Principles of Neural Science*, Elsevier, New York, 1985.
- [36] T. Kohonen, *Self-Organization and Associative Memory*, Springer-Verlag, Berlin, 1984.
- [37] T. Kohonen, K. Masisara, and T. Saramaki, "Phonotopic Maps – Insightful Representation of Phonological Features for Speech Representation," In *Proceedings IEEE 7th Inter. Conf. on Pattern Recognition*, Montreal, Canada, Jul. 1984, pp. 182-185.
- [38] A. Stoffels, T. Roska, and L. O. Chua, "An object-oriented approach to video coding via the CNN Universal Machine," in *Proceeding of 1996 Fourth IEEE Cellular Neural Networks and their Applications, CNNA-96*, vol. 43, Nov. 1996, pp. 948-952.
- [39] T. Roska and L. O. Chua, "The CNN universal machine: An analogic array computer," *IEEE Trans. Circuits Syst.*, vol. 40, pp. 163-172, Mar. 1993.
- [40] L. O. Chua, *CNN: A paradigm for complexity*, World Scientific Series on Nonlinear Science, vol. 31, 1998.
- [41] J. Stolte, G. Cserey, "Artificial immune systems based sound event detection with CNN-UM," in *Proceedings of the 2005 European Conference on Circuit Theory and Design, 2005*, vol. 3, Aug. 2005, pp. 11-14.

- [42] G. Cserey and T. Roska, "Artificial immune systems based novelty detection with CNN-UM," *IEEE Symposium on Foundations of Computational Intelligence, 2007. FOCI 2007.*, Apr. 2007, pp. 156-161.
- [43] G. Cserey, A. Falus, W. Porod, and T. Roska, "An Artificial Immune System for Visual Applications with CNN-UM", in *Proc. Of ISCAS 2004*, Vancouver, 2004.
- [44] A. Stoffels, T. Roska, and L. O. Chua, "Object-oriented image analysis for very-low-bitrate video-coding systems using the CNN Universal Machine," *Int. J. Circuit Theory Appl.*, vol. 25, pp. 235-258, 1997.
- [45] T. Sziranyi, K. Laszlo, L. Czuni, and F. Ziliani, "Object-oriented motion segmentation for video compression in the CNN-UM," *J. VLSI Signal Process.*, vol. 23, pp. 479-496, 1999.
- [46] L. O. Chua and T. Roska, *Cellular neural networks and visual computing: Foundations and applications*, Cambridge University Press, 2005.
- [47] L. Pivka, "Autowaves and spatio-temporal chaos in CNNs. I. A tutorial," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 42, pp. 638-649, Oct. 1995.
- [48] L. Pivka, "Autowaves and spatio-temporal chaos in CNNs. II. A tutorial," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 42, pp. 650-664, Oct. 1995.
- [49] Z. Yang, M. Yamauchi, Y. Nishio, and A. Ushida, "Relations between spatio-temporal phenomena and eigenvalues in mutually coupled CNNs," in *Proceedings of the 2003 International Symposium on Circuits and Systems, 2003. ISCAS '03*, vol. 3, May 2003, pp. III-578-III581.
- [50] A. M. Turing, "The chemical basis of morphogenesis," *Phil. Trans. Roy. Soc. Lond.*, pp. 37-72, 1952.
- [51] J. D. Murray, *Mathematical Biology*. Springer-Verlag, Berlin, 1989.
- [52] L. Goras and L. O. Chua, "Turing patterns in CNNs. I. Once over lightly," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 42, pp. 602-611, Oct. 1995.
- [53] L. Goras and L. O. Chua, "Turing patterns in CNNs. II. Equations and behaviors," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 42, pp. 612-626, Oct. 1995.

- [54] L. Goras and L. O. Chua, "Turing patterns in CNNs. III. Computer simulation results," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 42, pp. 627-637, Oct. 1995.
- [55] G. Manganaro, P. Arena, and L. Fortuna, *Cellular Neural Networks: Chaos, Complexity and VLSI Processing*. Springer-Verlag, New York, 1999.
- [56] K. Aihara, "Chaos engineering and its application to parallel distributed processing with chaotic neural networks," in *Proceeding of the IEEE*, vol. 90, May 2002, pp. 919-930.
- [57] H. Lu, Y. He, and Z. He, "A chaos-generator: analyses of complex dynamics of a cell equation in delayed cellular neural networks," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 45, pp. 178-181, Feb. 1998.
- [58] L. Nemes and T. Roska, "A CNN model of oscillation and chaos in ant colonies: a case study," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 42, pp. 741-745, Oct. 1995.
- [59] J. A. Anderson, "A simple neural network generating interactive memory," *Math. Biosci.*, vol. 14, pp. 197-220, 1972.
- [60] T. Kohonen, "Correlation matrix memories," *IEEE Trans. Comput.*, vol. C-21, pp.353-359, Apr. 1972.
- [61] K. Nakano, "Associatron-A model of associative memory," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-2, pp. 381-388, 1972.
- [62] T. Sziranyi, T. Roska, and Szolgay, P., "A cellular neural network embedded in a dual computing structure (CNND) and its use for character recognition," in *Proceedings of 1990 IEEE International Workshop on Cellular Neural Networks and their Applications*, Dec. 1990, pp. 92-99.
- [63] T. Sziranyi and J. Csicsvari, "High-speed character recognition using a dual cellular neural network architecture (CNND)," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 40, pp. 223-231, Mar. 1993.
- [64] E. David, P. Ungureanu, L. Goras, "On the Feature Extraction Performances of CNN Gabor-Type Filters in Texture Recognition Applications," *10th International Workshop on Cellular Neural Networks and Their Applications, 2006. CNNA '06*, Aug. 2006, pp. 1-6.
- [65] C. Y. Wu and J. F. Lan, "CMOS current-mode neural associative memory design with on-chip learning," *IEEE Trans. Neural Networks*, vol. 7, no. 1, pp. 167-181, 1996.
- [66] J. F. Lan and C. Y. Wu, "CMOS current-mode outstar neural networks with long-period analog ratio memory," in *Proc. IEEE Int. Symposium on Circuits and Systems, ISCAS*, 1995, vol. 3, pp. 1676-1679.

- [67] C. Y. Wu and C. H. Cheng, "A learnable cellular neural network structure with ratio memory for image processing", *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on*, vol. 49, issue 12, pp. 1713-1723, Dec. 2002.
- [68] G. Cserey, A. Falus, and T. Roska, "Immune Response Inspired CNN Algorithms for Many-Target Detection", in *Proc. Of ECCTD '03*, Krakow, 2003.
- [69] G. Cserey, A. Falus, W. Porod, and T. Roska, "Feature Extraction CNN Algorithms for Artificial Immune Systems", in *Proc. Of ISCAS 2004*, Vancouver, 2004.
- [70] G. Cserey, W. Porod, and T. Roska, "An Artificial Immune System based Visual Analysis Model and its Real-Time Terrain Surveillance Application", *ICARIS 2004*, Springer-Verlag, 2004, pp. 250-262.
- [71] A. Schultz, C. Rekeczky, I. Szatmari, T. Roska, and L. O. Chua, "Spatio-temporal CNN algorithm for object segmentation and object recognition," in *Proceedings of 1998 Fifth IEEE International Workshop on Cellular Neural Networks and Their Applications*, Apr. 1998, pp. 347-352.
- [72] I. Szatmari, A. Schultz, C. Rekeczky, T. Kozek, T. Roska, and L. O. Chua, "Morphology and autowave metric on CNN applied to bubble-debris classification," *IEEE Transactions on Neural Networks*, vol. 11, pp. 1385-1393, Nov. 2000.
- [73] T. W. Lee, M. S. Lewicki, and T. J. Sejnowski, "ICA mixture models for unsupervised classification of non-Gaussian classes and automatic context switching in blind signal separation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 1078-1089, Oct. 2000.
- [74] L. Koskinen, A. Paasio, A. Kananen, and K. Halonen, "MPEG-4 encoder architecture for a shape segmentation CNN chip," in *Proceedings of Workshop and Exhibition on MPEG-4. 2001*, Jun. 2001, pp. 41-44.
- [75] L. Koskinen, A. Paasio, M. Laiho, and K. Halonen, "Effect of CNN shape segmentation on MPEG-4 shape bit-rate," *IEEE International Symposium on Circuits and Systems, 2002. ISCAS 2002*, vol. 4, May 2002, pp. 552-555.
- [76] L. Koskinen, M. Laiho, A. Paasio, and K. Halonen, "MPEG-4 based modifications for a CNN segmentation chip" in *Proceedings of the 2002 7th IEEE International Workshop on Cellular Neural Networks and Their Applications, 2002. (CNNA 2002)*, Jul. 2002, pp. 71-71.
- [77] L. Koskinen, A. Paasio, K. Halonen, "CNN shape segmentation advantages in MPEG-4 simple profile encoding," in *Proceedings of Seventh International Symposium on Signal Processing and Its Applications*, vol. 2, Jul. 2003, pp. 117-120.

- [78] L. Koskinen, A. Paasio, and K. A. I. Halonen, "Motion estimation computational complexity reduction with CNN shape segmentation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, pp. 771-777, Jun. 2005.
- [79] P. Arena, L. Fortuna, and L. Occhipinti, "DNA chip image processing via cellular neural networks," *The 2001 IEEE International Symposium on Circuits and Systems, 2001. ISCAS 2001*, vol. 3, May 2001, pp. 345-348.
- [80] P. Arena, L. Fortuna, and L. Occhipinti, "A CNN algorithm for real time analysis of DNA microarrays," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 49, pp. 335-340, Mar. 2002.
- [81] P. Arena, M. Bucolo, L. Fortuna, and L. Occhipinti, "Cellular neural networks for real-time DNA microarray analysis," *IEEE Engineering in Medicine and Biology Magazine*, vol.21, pp. 17-25, Mar. 2002.
- [82] Y. W. Shou and C. T. Lin, "Image descreening by GA-CNN-based texture classification," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 51, pp. 2287-2299, Nov. 2004.
- [83] C. T. Lin, C. L. Chang, and J. F. Chung, "New horizon for CNN: with fuzzy paradigms for multimedia," *IEEE Circuits and Systems Magazine*, vol. 5, pp. 20-35, 2005.
- [84] P. Arena, A. Basile, M. Bucolo, and L. Fortuna, "An object-oriented segmentation on analog CNN chip," *IEEE Trans. Circuits Syst. I: Fundam. Theory Appl.*, vol. 50, pp. 837-846, 2003.
- [85] T. Sziranyi and M. Csapodi, "Texture classification and segmentation by cellular neural networks using genetic algorithms," *Comp. Vis. Image Understand.*, vol. 71, pp. 255-270, Sep. 1998.
- [86] G. F. Betta, S. Graffi, Z. M. Kovacs, and G. Masetti, "CMOS implementation of an analogically programmable cellular neural network," *IEEE Trans. Circuits Syst., Pt. II*, vol. 40, pp. 206-215, Mar. 1993.
- [87] P. Kinget and M. Steyaert, "A programmable analog cellular neural network CMOS chip for high speed image processing," *IEEE J. Solid-State Circuits*, vol. 30, pp. 235-243, Mar. 1995.
- [88] M. Anguita, F. J. Pelayo, A. Prieto, and J. Ortega, "Analog CMOS implementation of a discrete time CNN with programmable cloning templates," *IEEE Trans. Circuits Syst., Pt. II*, vol. 40, pp.215-218, Mar. 1993.

- [89] A. Rodriguez-Vazquez, R. Dominguez-Castro, and S. Espejo, "Design of CNN universal chips: Trends and obstacles," in *Proc. CNNA '94*, 1994, pp. 59-60.
- [90] J. M. Cruz and L. O. Chua, "A fast, complex and efficient test implementation of the CNN universal machine," in *Proc. IEEE CNNA '94*, 1994, pp. 61-66.
- [91] B. E. Shi, T. Roska, and L. O. Chua, "Design of linear cellular neural networks for motion sensitive filtering," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 40, pp. 320-331, May, 1993.
- [92] B. E. Shi and L. O. Chua, "Resistive grid image filtering: input/output analysis via the CNN framework," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 39, pp. 531-548, Jul. 1992.
- [93] B. E. Shi, "A one-dimensional CMOS focal plane array for Gabor-type image filtering," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 46, pp. 323-327, Feb. 1999.
- [94] J. Kowalski, "0.8 μm CMOS implementation of weighted-order statistic image filter based on cellular neural network architecture," *IEEE Transactions on Neural Networks*, vol. 14, pp. 1366-1374, Sept. 2003.
- [95] B. E. Shi, "Gabor-type filtering in space and time with cellular neural networks," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 45, pp. 121-132, Feb. 1998.
- [96] K. Slot, J. Kowalski, A. Napieralski, and T. Kacprzak, "Analogue median/average image filter based on cellular neural network paradigm," *Electronics Letters*, vol. 35, pp. 1619-1620, Sept. 1999.
- [97] L. Wang; J. P. De Gyvez, and E. Sanchez-Sinencio, "Time multiplexed color image processing based on a CNN with cell-state outputs," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 6, pp. 314-322, Jun. 1998.
- [98] C. C. Lee and J. P. de Gyvez, "Color image processing in a cellular neural-network environment," *IEEE Transactions on Neural Networks*, vol. 7, pp. 1086-1098, Sept. 1996.
- [99] K. R. Crouse, T. Roska, and L. O. Chua, "Image halftoning with cellular neural networks," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 40, pp. 267-283, Apr. 1993.
- [100] P. R. Bakic, N. S. Vujovic, D. P. Brzakovic, P. D. Kostic, and B. D. Reljin, "CNN paradigm based multilevel halftoning of digital images," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 44, pp. 50-53, Jan. 1997.

- [101] P. L. Venetianer, F. Werblin, T. Roska, and L. O. Chua, "Analogic CNN algorithms for some image compression and restoration tasks," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 42, pp. 278-284, May 1995.
- [102] P. L. Venetianter and T. Roska, "Image compression by cellular neural networks," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 45, pp. 205-215, Mar. 1998.
- [103] T. Kwok and K. A. Smith, "A unified framework for chaotic neural-network approaches to combinatorial optimization," *IEEE Transactions on Neural Networks*, vol. 10, pp. 978-981, Jul. 1999.
- [104] R. Fantacci, M. Forti, M. Marini, and L. Pancani, "Cellular neural network approach to a class of communication problems," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 46, pp. 1457-1467, Dec. 1999.
- [105] A. Zanela and S. Taraglio, "cellular neural network stereo vision system for autonomous robot navigation," in *Proceedings of the 2000 6th IEEE International Workshop on Cellular Neural Networks and Their Applications, 2000, (CNNA 2000)*, May 2000, pp. 117-122.
- [106] A. Loncar, R. Kunz, and R. Tetzlaff, "SCNN 2000. I. Basic structure and features of the simulation system for cellular neural networks," in *Proceedings of the 2000 6th IEEE International Workshop on Cellular Neural Networks and Their Applications, 2000, (CNNA 2000)*, May 2000, pp. 123-128.
- [107] M. Takahashi, T. Narukawa, and K. Yoshida, "Intelligent transfer and stabilization control to unstable equilibrium point of double inverted pendulum," *SICE 2003 Annual Conference*, vol. 2, Aug. 2003, pp. 1451-1456.
- [108] P. Arena and L. Fortuna, "Analog cellular locomotion control of hexapod robots," *IEEE Control Systems Magazine*, vol. 22, pp. 21-36, Dec. 2002.
- [109] L. F. C. Jeanmeure and W. B. J. Zimmerman, "A CNN video based control system for a coal froth flotation," in *Proceedings of 1998 Fifth IEEE International Workshop on Cellular Neural Networks and Their Applications*, pp. 192-197, Apr. 1998.
- [110] T. Yang and L. B. Yang, "Application of fuzzy cellular neural networks to Euclidean distance transformation," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 44, pp. 242-246, Mar. 1997.

- [111] C. T. Lin, C. L. Chang, and W. C. Cheng, "A recurrent fuzzy cellular neural network system with automatic structure and template learning," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 51, pp. 1024-1035, May 2004.
- [112] C. L. Chang, K. W. Fan, I. F. Chung, and C. T. Lin, "A Recurrent Fuzzy Coupled Cellular Neural Network System With Automatic Structure and Template Learning," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 53, pp. 602-606, Aug. 2006.
- [113] A. Faro, D. Giordano, and C. Spampinato, "Evaluation of the Traffic Parameters in a Metropolitan Area by Fusing Visual Perceptions and CNN Processing of Webcam Images," *IEEE Transactions on Neural Networks*, vol. 19, pp. 1108-1129, Jun. 2008.
- [114] T. Yang, L. B. Yang, C. W. Wu, and L. O. Chua, "Fuzzy cellular neural networks: applications," in *Proceedings of 1996 Fourth IEEE International Workshop on Cellular Neural Networks and their Applications, 1996. CNNA-96*, Jun. 1996, pp. 225-230.
- [115] L. O. Chua and T. Roska, "The CNN universal machine. I. The architecture," in *Proceedings of Second International Workshop on Cellular Neural Networks and their Applications, 1992. CNNA-92*, Oct. 1992, pp. 1-10.
- [116] T. Roska, J. Hamori, E. Labos, K. Lotz, L. Orzo, J. Takacs, P. L. Venetianer, Z. Vidnyanszky, A. Zarandy, "The use of CNN models in the subcortical visual pathway," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 40, pp. 182-195, Mar. 1993.
- [117] M. H. ter Brugge, J. H. Stevens, J. A. G. Nijhuis, and L. Spaanenburger, "Efficient DTCNN implementations for large-neighborhood functions," in *Proceedings of 1998 Fifth IEEE International Workshop on Cellular Neural Networks and Their Applications*, Apr. 1998, pp. 88-93.
- [118] K. Slot, "Large-neighborhood templates implementation in discrete-time CNN Universal Machine with a nearest-neighbor connection pattern," in *Proceedings of the Third IEEE International Workshop on Cellular Neural Networks and their Applications, 1994. CNNA-94.*, Dec. 1994, pp. 213-218.
- [119] C. Y. Wu and W. C. Yen, "A new compact neuron-bipolar junction transistor (vBJT), cellular neural network (CNN) structure with programmable large neighborhood symmetric templates for image processing," *IEEE Transaction on Circuits and Systems I: Fundamental Theory and Applications*, vol. 48, No. 1, pp. 12-27, Jan. 2001.
- [120] C. Y. Wu and C. Y. Wu, "An analysis and the fabrication technology of the lambda bipolar transistor," *IEEE Transactions on Electron Devices*, vol. 27, pp. 414-419, Feb. 1980.

- [121] C. W. Hsiao, A New CMOS Large-Neighborhood Cellular-Neural-Network(CNN) Cell Structure for Large-Neighborhood CNN Universal Machine (CNUM), NCTU, Hsin-Chu, 2001.
- [122] C. Y. Wu, C. Y. Hsieh, S. H. Chen, C. Y. Hsieh, and C. R. Chen, "Non-saturated binary image learning and recognition using the ratio-memory cellular neural network (RMCNN)," in *Proceedings of the 2002 7th IEEE International Workshop on Cellular Neural Networks and Their Applications, 2002. (CNNA 2002)*, Jul. 2002, pp. 624-629.
- [123] J. L. Lai and C. Y. Wu, "Architectural Design and Analysis of Learnable Self-Feedback Ratio-Memory Cellular Nonlinear Network (SRMCNN) for Nanoelectronic Systems," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, pp. 1182-1191, Nov. 2004.
- [124] C. Y. Wu and S. Y. Tsai, "Autonomous Ratio-Memory Cellular Nonlinear Network (ARMCNN) for Pattern Learning and Recognition," in *Proceeding of the 10th IEEE International Workshop on Cellular Neural Network and Their Applications, CNNA 2006*, pp. 137-141, Aug. 2006.
- [125] C. Y. Wu and Y. Wu, "The Design of CMOS Non-Self-Feedback Ratio Memory Cellular Nonlinear Network without Elapsed Operation for Pattern Learning and Recognition" in *Proceeding of the 2005 IEEE Cellular Neural Networks and their Application, CNNA 2005*, Hsin-chu, Taiwan, May 28-30 , 2005, pp. 282-285.
- [126] R. Dominguez-Castro, S. Espejo, A. Rodriguez-Vazquez, R. A. Carmona, P. Foldesy, A. Zarandy, P. Szolgay, T. Sziranyi, and T. Roska, "A 0.8- μm CMOS two-dimensional programmable mixed-signal focal-plane array processor with on-chip binary imaging and instructions storage," *IEEE Journal of Solid-State Circuits*, vol. 32, pp. 1013-1026, Jul. 1997.
- [127] D. Farmer, T. Toffoli, and S. Wolfman, "Cellular Automata," in *Proc. Interdisciplinary Workshop*. New York: North-Holland Physics Pub. 1984
- [128] T. Toffoli, *Cellular Automata Machines: A New Environment for Modeling*. Cambridge: MIT Press, Series in Scientific Computation, 1987.
- [129] T. Roska, "Analogic algorithms running on the CNN universal machine," in *Proc. Of the 3rd IEEE Int. Workshop on Cellular Neural Networks Appl. (CNNA-94)*, Rome, Italy, Dec. 1994, pp. 3-8.

- [130] S. Espejo, R. Carmona, R. Domínguez-Castro, and A. Rodríguez-Vázquez, “A CNN universal chip in CMOS technology,” *International Journal of Circuit Theory and Applications*, vol. 24, pp.93-109, Mar. 1996.
- [131] A. Rodríguez-Vázquez, G. Linan-Cembrano, L. Carranza, E. Roca-Moreno, R. Carmona-Galan, F. Jimenez-Garrido, R. Dominguez-Castro, and S. E. Meana, “ACE16k: the third generation of mixed-signal SIMD-CNN ACE chips toward VSoCs,” *IEEE Tran. Circuits Syst.*, vol. 51, Issue 5, pp. 851 – 863, May 2004.
- [132] G. Linan, S. Espejo, R. Dominguez-Castro, E. Roca, and A. Rodriguez-Vazquen, “CNNUC3: a mixed-signal 64×64 CNN universal chip,” in *Proceedings of Conference on the Seventh International Microelectronics for Neural, Fuzzy and Bio-Inspired Systems, 1999. (MicroNeuro '99)*, 7-9 April 1999, pp. 61 – 68.
- [133] G. Linan, S. Espejo, R. Dominguez-Castro, and A. Rodriguez-Vazquen, “The CNNUC3: an analog I/O 64x64 CNN universal machine chip prototype with 7-bit analog accuracy,” in *Proc. Of the 6rd IEEE Int. Workshop on Cellular Neural Networks Appl. (CNNA 2000)*, 23-25 May 2000, pp. 201 – 206.
- [134] C. Y. Wu and W. C. Yen, “The neuron-bipolar junction transistor (v-BJT)-a new device structure for VLSI neural network implementation,” *IEEE International Conference on Electronics, Circuits and Systems*, vol.3, Sep. 1998, pp.277-270.
- [135] W. C. Yen and C. Y. Wu, “The design of neuron-bipolar junction transistor (vBJT) cellular neural network (CNN) structure with large neighborhood templates,” in *Proc. Of the 6th IEEE Int. Workshop on Cellular Networks and Their Applications*, Catania, Italy, May 2000, pp. 195-200.
- [136] G. Timar and C. Rekeczky, “A real-time multitarget tracking system with robust multichannel CNN-UM algorithms”, *IEEE Tran. Circuits Syst.*, vol. 52, issue 7, pp. 1358-1371, July 2005.
- [137] Chin-Teng Lin, Chao-Hui Huang, Shi-An Chen, “CNN-Based Hybrid-Order Texture Segregation as Early Vision Processing and Its Implementation on CNN-UM”, *IEEE Tran. Circuits Syst.*, vol. 54, issue 10, pp. 2277-2287, Oct. 2007.
- [138] N. A. Fernandez, D. L. Valarino, V. M. Brea, and D. Cabello, “On the emulation of large-neighborhood templates with binary CNN-based architectures,” *9th International Workshop on Cellular Neural Networks and Their Applications 2005*, 28-30 May 2005, pp. 274 – 277.

- [139] C.H. Cheng, S.H. Chen, L.J. Lin, K.H. Huang, and C.Y. Wu, "A new structure of large-neighborhood cellular nonlinear network (LNCNN)," in *Proc. Of IEEE Int. Joint Conference on Neural Networks*, 2003, pp. 1497-1501.
- [140] S.H. Chen and C.Y. Wu, "A low power design on diffusive interconnection large-neighborhood cellular nonlinear network for giga-scale system application," in *Proc. of the 11th IEEE Int. Conference on Electronics, Circuits and Systems*, Dec. 2004, pp.179-182.
- [141] VisMouse-CNN Visual Mouse Platform for Windows, Reference Manual, Analogical and Neural Computing Laboratory, Computer and Automation Institute (MTA SzTAKI) of the Hungarian Academy of Sciences, Budapest, Hungary, 1998.
- [142] K. Nakamura, K. Arimura, and T. Yoshikawa, "Recognition of object orientation and shape by a rotation spreading associative neural network", in *Proceedings of IJCNN '01. International Joint Conference on Neural Networks, 2001*, vol. 1, 15-19 July 2001, pp. 565-570.
- [143] M. Namba and Z. Zhang, "Cellular Neural Network for Associative Memory and Its Application to Braille Image Recognition", *International Joint Conference on Neural Networks, 2006. IJCNN '06*, 16-21 July 2006, pp. 2409-2414.
- [144] S. Grossberg, "Nonlinear difference-differential equations in prediction and learning theory," in *Proc. Natl. Acad. Sci. USA*, vol. 58, pp. 1329-1334, 1967.
- [145] J. A. Feldman and D. H. Ballard, "Connectionist models and their properties," *Cognitive Science*, vol. 6, pp. 205-254, 1982.

Publication List

(A) JOURNAL PAPERS

- [1] Chung-Yu Wu and Sheng-Hao Chen, “The Design and Analysis of a CMOS Low-Power, Large-Neighborhood CNN with Propagating Connections,” *IEEE Transactions on Circuits and Systems I*, vol. 56, issue 2, pp. 440-452, Feb. 2009.
- [2] Chung-Yu Wu, Sheng-Hao Chen, and Yu Wu, “The Design and Analysis of a CMOS Ratio-Memory Cellular Nonlinear Network (RMCNN) without Elapsed Time,” submitted to *IEEE Transactions on Circuits and Systems I*.

(B) CONFERENCE PAPERS

- [1] Sheng-Hao Chen and Chung-Yu Wu, “A low power design on diffusive interconnection large-neighborhood cellular nonlinear network for giga-scale system application,” in *Proceedings of the 2004 11th IEEE International Conference on Electronics, Circuits and Systems, 2004. ICECS 2004*, Dec. 2004, pp. 179-182.
- [2] Chiu-Hung Cheng, Sheng-Hao Chen, Li-Ju Lin, Kuan-Hsun Huang, and Chung-Yu Wu, “A new structure of large-neighborhood cellular nonlinear network (LN-CNN),” in *Proceedings of the International Joint Conference on Neural Networks, 2003.*, Jul. 2003, pp. 1497-1501.
- [3] Chung-Yu Wu, Chieh-Yu Hsieh, Sheng-Hao Chen, Brian Che-Yuan Hsieh, and Cheng-Ruei Chen, “Non-saturated binary image learning and recognition using the ratio-memory cellular neural network (RMCNN),” in *Proceedings of the 2002 7th IEEE International Workshop on Cellular Neural Networks and Their Applications, 2002. (CNNA 2002)*, Jul. 2002, pp. 624-629.

簡 歷

姓 名：陳勝豪

性 別：男

出生年月日：民國69年07月28日

出 生 地：台灣省彰化縣

地 址：彰化縣和美鎮竹營里孝昌路19巷15號

學 歷： 國立交通大學電子工程學系
(87年9月－91年1月)

國立交通大學電子研究所碩士班
(91年2月－92年7月)

國立交通大學電子研究所博士班
(92年9月－98年7月)

