國立交通大學

電子工程學系　電子研究所碩士班

碩 士 論 文

自發性比例式記憶體細胞非線性網路之設計

The Design of the Autonomous Ratio Memory Cellular Nonlinear Network without Elapsed Operation for Pattern Learning and Recognition

研 究 生：周維德

指導教授：吳重雨　教授

中 華 民 國 九 十 六 年 十 二 月

自發性比例式記憶體細胞非線性網路之設計

The Design of the Autonomous Ratio Memory Cellular Nonlinear Network without Elapsed Operation for Pattern Learning and Recognition

研 究 生：周維德　　Student：Wei-Te Chou

指導教授：吳重雨　　Advisor：Chong-Yu Wu

國 立 交 通 大 學

電子工程學系電子研究所

碩 士 論 文

A Thesis

Submitted to Department of Computer and Information Science

College of Electrical Engineering and Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Electronics Engineering

December 2007

Hsinchu, Taiwan, Republic of China

中華民國九十六年十二月

自發性比例式記憶體細胞
非線性網路之設計

學生：周維德　　　　　指導教授：吳重雨 博士

國立交通大學

電子工程學系　電子研究所碩士班

摘要

在圖形辨識的領域中，聯想式記憶體是一種相當熱門的辨識方法，它能將含有雜訊的圖形恢復成完美無雜訊的圖形，而比例式記憶細胞非線性網路已被證實可以作為一種聯想式記憶的實現方法。而目前比例式記憶體非線性網路所面臨的挑戰即是如何提升其在高雜訊的環境下的辨識率。

本論文的主旨在於闡述自發性比例式記憶體細胞非線性網路（Autonomous Ratio-Memory Cellular Nonlinear Network，簡稱 ARMCNN）架構之分析與設計及其在聯想式記憶及圖像辨識上之應用。所謂的自發性是指在辨識階段，帶有雜訊的輸入訊號將在各個細胞存成初始電壓，而非一固定的輸入電壓。此外，本設計也具有免衰減操作即可得到所需比例鍵值之優點。在圖形學習階段，過去的比例式記憶細胞非線性網路（Ratio-Memory Cellular Neural Network，簡稱 RMCNN）比例鍵值產生方式是將絕對鍵值（absolute weight）與細胞鄰近四邊的絕對鍵值平均值作比較，如果大於平均值，則此鍵值將被保留，反之則忽略此鍵值。而新提出的 ARMCNN，是將細胞相鄰四邊的絕對鍵值改為只保留最大的絕對鍵值。模擬結果證明自發性比例式記憶細胞非線性網路相較於具有較高的辨識率。

論文中除了以 Matlab 和 C 語言模擬自發性比例式記憶細胞非線性網路架構（ARMCNN）及其在聯想式記憶和圖像辨識上之應用外，並實際以 TSMC 0.35um 2P4M Mixed-Signal 製程設計了一解析度為 9x9 的 ARMCNN 網路，並實現之且加以量測。本設

計的單位面積在相同製程下，縮小為前一版設計—免衰減操作之 RMCNN 的 0.28 倍大(從 4.56mm x 3.90mm 縮小到 2.24mm x 2.24mm)。

量測中所學習的三個圖形（一、二、四）皆可成功的辨識，而辨識中的一些瑕疵，也將在論文中進行探討。並從新設計電路，在 Hspice 模擬驗證新電路確實可以改善此缺陷。

The Design of the Autonomous Ratio-Memory Cellular Nonlinear Network for Pattern Learning and Recognition

Student: Wei-Te Chou　　　　　　　　Advisor: Prof. Chung-Yu Wu

Department of Electronics Engineering & Institute of Electronics

National Chiao-Tung University

ABSTRACT

The associative memory is of significant attention in the field of pattern recognition and recovery. It is proven that the cellular nonlinear network with the aid of ratio memory (RMCNN) can be used to implement as a kind of associative memory. However, there are still some imperfections that require further improvement for the existing RMCNN system. For example, the pattern recognition rate of RMCNN drops quickly as the environmental noise level raises. Moreover, the die area of the existing chip is too large (4.56mm x 3.90mm), which might suffer from the impact of process variation more seriously. Therefore, the chip area reduction and optimization are necessary.

A new type of CNN associative memory called the Autonomous Ratio-Memory Cellular Nonlinear Network (ARMCNN) is proposed and analyzed. In the proposed ARMCNN, there is no elapsed operation to perform weight enhancement as well. During recognition period, the noisy input patterns are sent into cells as initial cell state voltages, which in comparison with constantly injecting the noisy input patterns, yields a better recognition rate in simulation.

During pattern learning period, the ratio weight is original generated by comparing the four neighboring absolute weights with their mean value. The absolute weights that are bigger than the mean value will remain. However, in ARMCNN, only the strongest absolute weights will stay (might be more than one). Furthermore, the proposed ARMCNN inherits the features of RMCNN such as, feature enhancement effect and no elapsed operation (EO). The ratio weights are generated directly after pattern learned.

In this thesis, the circuit of ARMCNN w/o EO is designed and a 9x9 ARMCNN is implemented using TSMC 0.35um 2P4M mixed-signal process. The die area, as compared with the previous chip – RMCNN w/o elapsed operation, shrinks from 4.56mm x 3.90mm to 2.24mm x 2.24mm. It's only 0.28 times as large as the previous chip under the same technology process, which greatly reduces the impact of process variation. The experimental results of recognizing all three patterns are successful. However, some imperfections of pattern recovery still exist and will be discussed later in this thesis. The circuit is redesigned to correct these imperfections.

CONTENT

TABLE CAPTIONS

FIGURE CAPTIONS

CHAPTER 1
INTRODUCTION

1.1 Background of Cellular Nonlinear Network

The cellular nonlinear network (CNN) introduced by Chua and Yang [1]-[2] has been considered as one of the potential architecture in future nano-electronic systems. One of CNN's important applications is the associative memory. So far, some research works on the applications of CNN as neural associative memories for pattern learning, recognition, and association have been explored [3]-[9]. As to the hardware implementation, special learning algorithm and digital hardware implementation for CNN were proposed in [8] to solve the sensitivity problems caused by the limited precision of analog weights. Also, the CMOS chip implementation of CNN associative memory was reported in [9].

The learning circuitry can be integrated on-chip with the CNN system. There are several advantages of on-chip learning: 1) No host computer is needed to perform the learning task off-line. This makes the interface of neural system chips simpler for many practical applications; 2) The spatial-variant template weights can be on-chip learned without being loaded from outside to the CNN chips. In other words, the long loading time, complex global interconnection between cells, and analog weight storage elements to perform the loading operation for large numbers of spatial-variant template weights can be avoided; 3) The adaptability to the process variations of CNN chips can be enhanced.

To implement the associative memories, both the ratio memory (RM) [10]-[22] and the generalization of Hebb's postulate of learning [23]-[24] have been incorporated with the CNN structure to form the RMCNN with spaced-variant templates for pattern recognition. The use of ratio-memory (RM) is to enhance important ratio weights and remove less important ones through the effect of feature enhancement [10]-[22]. In the RMCNN of [10]-[13], the input signal current is applied to the neuron throughout the recognition process and the initial state of each neuron is set to zero. In this paper, the design of the Autonomous RMCNN [27] is proposed to improve the recognition rate, which makes some modification on the Hebbian learning rule. In addition, the input signal is stored as initial state of the cell and no constant input applied to the neuron throughout the recognition process.

1.2 Review of Ratio Memory Cellular Nonlinear Network

1.2.1 Ratio memory Cellular Nonlinear Network with Elapsed Operation

In the previous work, ratio memory cellular nonlinear network (RMCNN), the cell state $x_{ij}(t)$, its derivation $\dot{x}_{ij}(t)$, and the cell output $y_{ij}(t)$ for a regular cells can be expressed as [1]-[2]

$$\dot{x}_{ij}(t) = -x_{ij}(t) + \sum_{C(k,l)\in Nr(i,j)} a_{ijkl}(t) y_{kl}(t) + \sum_{C(k,l)\in Nr(i,j)} b_{ijkl}(t) u_{kl} + z_{ij} \qquad \text{Eq.(1.1)}$$

$$y_{ij}(t) = f\left(x_{ij}(t)\right) = \begin{cases} x_{ij}(t) & \text{if } -1 \le x_{ij}(t) \le +1 \\ +1 & \text{if } \quad x_{ij}(t) > +1 \\ -1 & \text{if } \quad x_{ij}(t) < -1 \end{cases} \qquad \text{Eq.(1.2)}$$

where $x_{ij}(t)$ is the state of cell(i,j), and $u_{kl}(t)$ is the input of cell(k,l) in the r-neighborhood system Nr(i, j) of the cell(i, j). In this thesis, i or k is the row number and j or l are the column number of an MxN CNN cell array. So, cell(i,j) means the ith row and jth column cell. The r-neighborhood system Nr(i, j) of the cell cell(i, j) is defined as the set of all cells including cell(i, j) and its neighboring cells, which satisfy the following property.

$$N_r(i,j) = \left\{ C(k,l) \middle| 1 \le k \le M, 1 \le l \le N, |k-i| + |l-j| \le r \right\} \quad [13] \qquad \text{Eq.(1.3)}$$

The term r is called as the radius or the number of neighboring layer. In our design, r is 1. $a_{ijkl}(t)$ is template A weight(coefficient) which correlates the cell output $y_{kl}(t)$ to the cell state

$x_{ij}(t)$. $b_{ijkl}(t)$ is the template B weight(coefficient) which correlates the cell input u_{kl} to the cell state x_{ij} and z_{ij} is the threshold or bias of cell(i,j).

The template B and the threshold z_{ij} are constant and space-invariant. The setting is

$$\mathbf{B}_{ij}(t) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad \text{Eq.(1.4)}$$

$$z_{ij}(t) = 0 \qquad \text{Eq.(1.5)}$$

That means the input of every cell influences itself only. In a r-neighborhood system Nr(i, j), the input of neighboring cell doesn't influence the central cell. The threshold z_{ij} is zero everywhere. The template A is spatial-variant and time-variant[12]-[13], and the template A_{ij} can be written as:

$$A_{ij}(0) = \begin{bmatrix} 0 & a_{ij(i-1)j}(0) & 0 \\ a_{iji(j-1)}(0) & 0 & a_{iji(j+1)}(0) \\ 0 & a_{ij(i+1)j}(0) & 0 \end{bmatrix} \qquad \text{Eq.(1.6)}$$

That means only four cell are correlated to the central cell. They are up, down, left and right side cells. In the original RMCNN with elapsed operation[13]. The weights in template A can be produced by the blow equation.

$$a_{ijkl}(0) = \frac{\sum_{p=1}^{m} \int_{T_P} u_{ij}^{p} u_{kl}^{p}\, dt}{sum1} \qquad \text{Eq.(1.7)}$$

$$kl \in (i-1)j,\ i(j-1),\ i(j+1), (i+1)j \qquad \text{Eq.(1.8)}$$

$$sum\,1 = \sum_{kl} \left| \sum_{p=1}^{m} \int_{T_P} u_{ij}^{p} u_{kl}^{p}\, dt \right| \qquad \text{Eq.(1.9)}$$

Where u_{ij}^{p} is the p-th pattern input of cell(i,j). Similarly, u_{kl}^{p} is the p-th pattern of cell(k,l). The relationship between ij and kl is shown as Eq.(1.8) that is equivalent to . The T_P is the learning time for the RMCNN to learn p-th pattern and the total learning time for the

RMCNN to learn m patterns is $T_L = \sum_{p=1}^{m} T_P$. a_{ijkl} is called as the ratio weight, and the numerator of aijkl is called as the absolute-weight.

The boundary cells don’t correlate to four cells. For example, the boundary cells at corners only correlate to two cells. Thus the boundary condition of the boundary cells can be written as

$$x_{i^* j^*}(t) = 0 \ , \quad u_{i^* j^*}(t) = 0 \qquad \text{Eq.(1.10)}$$

The i*j* means this cell is a boundary cell.

This work has advantages of longer memory retention time, and the feature enhance characteristic improves the recognition rate. However, the elapsed period changes as learning patterns change, and thus the elapsed operation let the process of pattern recognition inconvenient.

1.2.2 Ratio memory Cellular Nonlinear Network w/o Elapsed Operation

Due to the inconvenience of having elapsed operation, the RMCNN w/o elapsed operation (EO) was proposed, which yields the same pattern recognition rates and simpler circuit structure as compared with RMCNN with EO. It requires no additional elapsed period to get the feature enhanced ratio weights. Indeed, the ratio weights are generated directly after pattern learning. Moreover, RMCNN can have longer template-weight storage time or equivalently pattern recognition time which is one of the advantages of RM. Due to the feature enhancement effect of the RM, which well separates the learned weights and decreases the insignificant weights to zero, more patterns can be stored and recognized in the RMCNN as compared to the CNN associative memory without RM.

The equation used to distinguish which ratio weights increase and which ratio weights decrease can be written as [24]

$$I_{Mss}(t) = \frac{\sum_{j=1}^{n} I_{aw(j)}(t)}{n} \qquad \text{Eq (1.11)}$$

where $I_{Mss}(t)$ is the mean of absolute memory current and $I_{aw(j)}(t)$ is the jth absolute memory current. If $I_{aw(j)}(t)$ is larger than $I_{Mss}(t)$, ratio memory current increase gradually. Otherwise the ratio weights decrease. So the increasing and decreasing ratio weights are detected. After the comparing operation, the increasing weights are set an appropriate value (1,1/2,1/3 or 1/4) and the decreasing weights are set zero directly This equation is used to determine the final ratio weights directly rather than elapsed operation. The new Hebbian learning algorithm can be written as blow:

Step 1 : find the absolute weights template A $S_{ij}(p)$ after p patterns are learned

$$S_{ij}(p) \;=\; \begin{bmatrix} 0 & ss_{ij(i-1)j}(p) & 0 \\ ss_{iji(j-1)}(p) & 0 & ss_{iji(j+1)}(p) \\ 0 & ss_{ij(i+1)j}(p) & 0 \end{bmatrix}$$

$$ss_{ijkl}(p+1) = ss_{ijkl}(p) + u_{ij}^{p+1} u_{kl}^{p+1}$$

$$(k,l) \text{ can be } (i+1, j) \text{ or } (i, j+1) \text{ or } (i-1, j) \text{ or } (i, j-1)$$

Step 2 : find the absolute mean of the absolute weights in a template

$$M_{ss} = mean(\sum |ss_{ijkl}|)$$

Step 3 : generate the ratio weights

$$\begin{cases} a_{ijkl} = \dfrac{1}{PN_{Nr(i,j)}} & \text{if } ss_{ijkl} > Mss \\ a_{ijkl} = 0 & \text{if } ss_{ijkl} < Mss \end{cases}$$

Where u_{ij}^{p+1} and u_{kl}^{p+1} are the input of cell(i,j) and cell(k,l) respectively. The $PN_{Nr(i,j)}$ is number of preserved weights in $N_r(i,j)$ and r=1.

The measurement results from RMCNN w/o EO was as following. Three Chinese characters were learned: 一, 二, and 四 (they are one, two, and four, respectively). Unfortunately, the pattern '四' failed to be recognized and recovered. Further investigation into the cause of this imperfection showed that a small current influences the absolute weights on the capacitor *Cw* during the pattern transferring period, which would lead to wrong absolute weights and, thus, wrong ratio weights were generated and stored. Therefore, the newly proposed ARMCNN chip has corrected this mistake. In addition, the die area of the previous work RMCNN chip was considerable large (4.56mm x 3.90mm), which might suffer from the impact of process variation more significantly. Therefore, the chip area reduction and optimization is necessary.

1.3 Research Motivation and Thesis Organization

To improve the image pattern recognition rates of RMCNN, the autonomous Ratio-Memory Cellular Nonlinear Network (ARMCNN) [27] is proposed and analyzed. In the ARMCNN, the input currents of the noisy input patterns are used to pre-charge the capacitors of neurons (C_{ij}) to produce the initial cell state voltages at the beginning of the recognition process. After pre-charging, all the input currents are removed from the neurons. Since no B-template is used and the neuron capacitors store the initial state voltages, the proposed RMCNN is called autonomous RMCNN (ARMCNN). The ARMCNN inherits the features of RMCNN such as, feature enhancement effect and no elapsed operation (ratio weights are generated directly after pattern learned). The mathematical analysis and simulations are performed for both ARMCNN and RMCNN. It is shown that the ARMCNN has a higher recognition rate, and more number of learned and recognized patterns.

The operational principle and circuit architecture of the proposed ARMCNN are described in Chapter Two, where the prediction models of recognition rate are shown as well. Chapter Three discusses the simulation results of ARMCNN, which includes the behavior simulations using C/C++ and the transistor-level simulations using Hspice. Then in Chapter Four, layout description and measurement environmental setup are mentioned. Finally, the conclusion and future work are discussed in Chapter Five. As a demonstrative example, a resolution 9x9 ARMCNN without elapsed operation (ARMCNN w/o EO) is realized in TSMC 0.35um 2P4M Mixed-Signal technology. Both simulation and experimental results have verified the superior characteristics of the ARMCNN system.

CHAPTER 2

ARCHITECTURE AND CIRCUITRY

2.1 Operational Principle and Architecture [27]

The architecture of ARMCNN is similar to that of RMCNN [1]-[4]. The operation procedures of ARMCNN can be divided into three phases: the pattern learning phase, the ratio-weights generation phase, and the pattern recognition phase. One difference between ARMCNN and RMCNN is that in the recognition operation where the noisy pattern to be recognized and recovered is treated as the initial values of cell state voltages in ARMCNN and as the neuron input in RMCNN. The operational principle of ARMCNN is described below.

In the autonomous ratio-memory cellular nonlinear network (ARMCNN), the dynamic equations of the cell state voltage $x_{ij}(t)$ and its derivative $\dot{x}_{ij}(t)$ can be expressed as

$$\dot{x}_{ij}(t) = -x_{ij}(t) + \sum_{C(k,l)\in N_r^0(i,j)} \frac{w_{ijkl}\, y_{kl}(t)}{C_{ij}} \qquad \text{Eq.(2.1)}$$

$$y_{kl}(t) = f\left(x_{ij}(t)\right) \qquad \text{Eq.(2.2)}$$

$$-u_{ij}R_{ij} \le x_{ij}(0) \le u_{ij}R_{ij}$$

where $N_r^0(i,j)$ is the set of r-neighboring cells $N_r(i,j)$ without the cell $C(i,j)$. The r-neighborhood system $N_r(i,j)$ of the cell $C(i,j)$ is defined as the set of all cells including $C(i,j)$ and its neighboring cells, which satisfy the following property

$$N_r(i,j) = \{C(k,l)\,|\,1 \le k \le M, 1 \le l \le N, |k-i| + |l-j| \le r\}$$

The term r is an integer called the radius of the neighborhood layer and r equal to one in our design. Moreover, $x_{ij}(0)$ is the initial value of the cell state voltage and has a value of $u_{ij}R_{ij}$

(-u$_{ij}$*R*$_{ij}$*)* for a black (white) pixel and a mediate value for a grey pixel.

The coefficient $a_{ij}(t)$ represents template A weight which correlates the cell output $y_{kl}(t)$ to the cell state $x_{ij}(t)$. The template A is spatial-variant and time-variant [18]-[19], and the template $a_{ij}(t)$ can be written as

$$A_{ij}(0) = \begin{bmatrix} 0 & a_{ij(i-1)j}(0) & 0 \\ a_{iji(j-1)}(0) & 0 & a_{iji(j+1)}(0) \\ 0 & a_{ij(i+1)j}(0) & 0 \end{bmatrix} \qquad \text{Eq.(2.3)}$$

That means only four neighboring cells' outputs are correlated to the central cell. The four neighboring cells are up, down, left, and right side cells. Since the proposed ARMCNN has the neuron capacitors to store the initial state voltages, no template B weight is used to correlate the cell input u_{kl} to the cell state x_{ij} and the template B coefficient is set to zero. That is to say no constant input is injected during pattern recognition phase.

$$B_{ij}(t) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad \text{Eq.(2.4)}$$

In the learning period, assume that there are *m* patterns to be learned in the ARMCNN. The absolute weight s_{ijkl} can be determined by the modified Hebbian rule as [1]-[4]

$$s_{ijkl} = \sum_{p=1}^{m} u_{ij}^{p} u_{kl}^{p} \qquad \text{Eq.(2.5)}$$

Where u_{ij}^{p} is the pixel input at *i*th row and *j*th column of the *p*th pattern out of *m* input patterns, u_{kl}^{p} is the pixel input in the set of $N_r^0(i,j)$. After learning all input patterns, the absolute weight s_{ijkl} is to be compared with the strongest learned weight. Those weights that are equal to the strongest absolute weight will stay and transform into the ratio weight a_{ijkl} as

$$a_{ijkl} = \frac{1}{PN_{Nr(i,j)}} \qquad \text{Eq.(2.6)}$$

Where PN$_{\mathrm{Nr}}$(i,j) represents the number of preserved weights left. On the other hand, absoluted weights that are less than the strongest learned weight will be set to zero and disregard. This procedure determines and stores the final ratio weights between cells. The new Hebbian learning algorithm can be written as following:

<u>Step 1</u>: *find the absolute weights template A $s_{ij}(p)$ after p patterns are learned*

$$s_{ij}(p) = \begin{bmatrix} 0 & ss_{ij(i-1)j}(p) & 0 \\ ss_{iji(j-1)}(p) & 0 & ss_{iji(j+1)}(p) \\ 0 & ss_{ij(i+1)j}(p) & 0 \end{bmatrix} \qquad \text{Eq.(2.7)}$$

$$s_{ijkl}(p) = s_{ijkl}(p-1) + u_{ij}^{p} u_{kl}^{p} \qquad \text{Eq.(2.8)}$$

$$(k,l) \text{ can be } (i+1, j) \text{ or } (i, j+1) \text{ or } (i-1, j) \text{ or } (i, j-1)$$

<u>Step 2</u>: *determine the global maximum value of the absolute weights in system*

$$Gss = \max(\sum |ss_{ijkl}|)$$

<u>Step 3</u>: *compare the absolute weights with the global maximum weight. Those equal to the strongest absolute weights will stay. On the other hand, those less than the strongest absolute weights will be set to zero.*

<u>Step 4</u>: *transform the remaining absolute weights into the ratio weights*

$$\begin{cases} a_{ijkl} = \dfrac{1}{PN_{Nr(i,j)}} & \text{if } ss_{ijkl} = Gss \\ a_{ijkl} = 0 & \text{if } ss_{ijkl} < Gss \end{cases} \qquad \text{Eq.(2.9)}$$

Where u_{ij}^{p+1} and u_{kl}^{p+1} are the input of *cell(i,j)* and *cell(k,l)* respectively. The $PN_{Nr(i,j)}$ is number of preserved weights in $N_r(i,j)$ and $r = 1$. The boundary cells are exceptional cases since they do not correlate to the four neighboring cells: up, down, right, and left. They might correlate to only two or three neighboring cells. Therefore, the condition of the boundary cells can be expressed as

$$x_{i^* j^*}(t) = 0 \ , \quad u_{i^* j^*}(t) = 0 \qquad \text{Eq.(2.10)}$$

The notation $i^* j^*$ means this cell is a boundary cell.

Table 2.1 shows some sample template A of absolute-weights and ratio-weights. It is clear that only the strongest absolute weights are set to one after comparing with the global maximum absolute weight *Gss*, and the others are disregarded and set to zero. With the aid of this technique, the template A suppresses the unimportant weights and enhances the significant weights to get a feature enhance template. After the comparing operation, the remaining weights are set to appropriate value (*1*, *1/2*, *1/3* or *1/4*) and the others are set to zero. This equation determines the final ratio weights directly without elapsed operation.

Absolute-weights (learned)	Absolute-weights (enhancement)	Ratio weights
$z^{\#}_{44} = \begin{bmatrix} 0 & -\frac{1}{3} & 0 \\ \frac{1}{3} & 0 & \frac{1}{3} \\ 0 & \frac{1}{1} & 0 \end{bmatrix}$	$z_{44} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$	$w_{44} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$
$z^{\#}_{51} = \begin{bmatrix} 0 & \frac{1}{1} & 0 \\ 0 & 0 & \frac{1}{3} \\ 0 & \frac{1}{3} & 0 \end{bmatrix}$	$z_{51} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$w_{51} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$
$z^{\#}_{62} = \begin{bmatrix} 0 & -\frac{1}{3} & 0 \\ \frac{1}{1} & 0 & \frac{1}{1} \\ 0 & \frac{1}{3} & 0 \end{bmatrix}$	$z_{62} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$	$w_{62} = \begin{bmatrix} 0 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 0 \end{bmatrix}$

Table 2.1 some sample template A of absolute-weights and ratio-weights

In this work, a 9x9 resolution ARMCNN is implemented and measured. Fig 2.1 shows the block diagram of the ARMCNN w/o elapsed operation (EO) and the controlling relationship between every block. A 9x9 shift register is used to store image patterns. The patterns are generated by pattern generator and are inputted into the shift registers in series. Once an image pattern is stored in register completely, the pattern is inputted into ARMCNN w/o EO in parallel for pattern learning. After all patterns are learned and ratio-weights are generated, the ARMCNN w/o EO enters into recognition phase. The recognition result is readout through the output stage, which is controlled by two decoders: **Column_Decoder** and **Row_Decoder**. Since there is only one pin dedicated for output readout, the state of each cell is outputted in series.

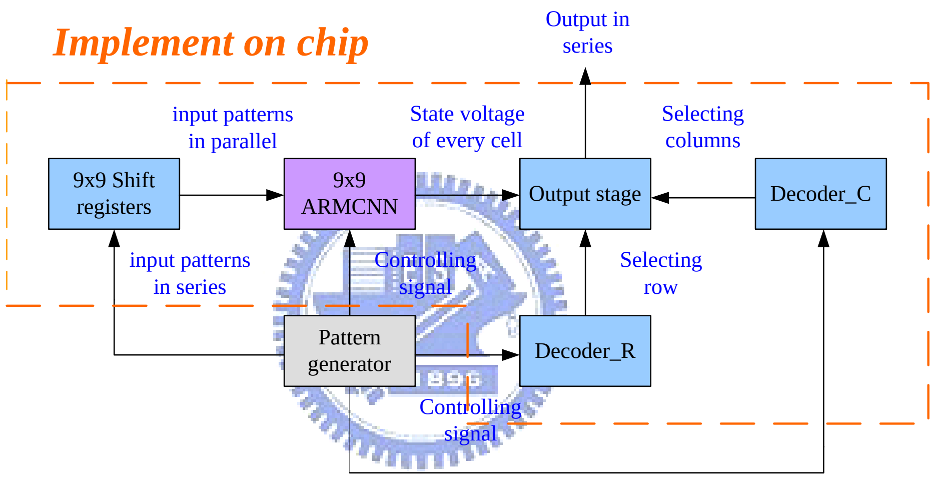


Fig 2.1 The block diagram of ARMCNN and controlling relationship between every block

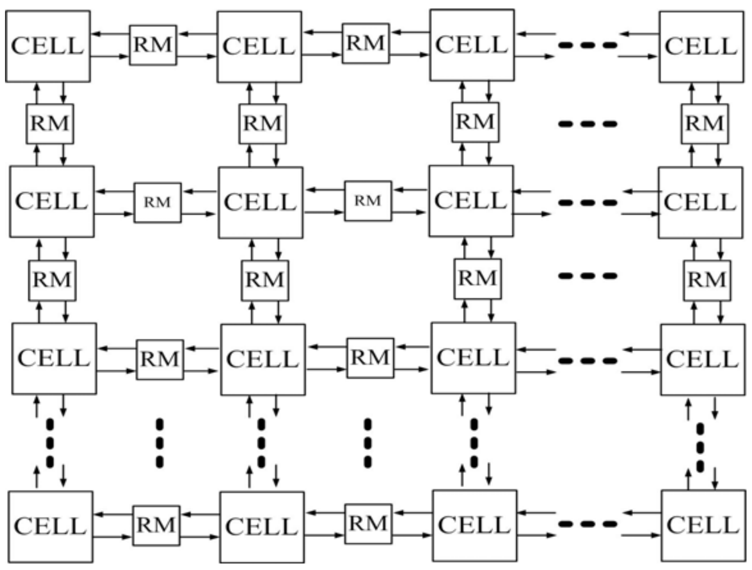


Fig 2.2 The general architecture of ARMCNN

The general architecture of ARMCNN w/o EO and connections between cells and **RM**s are shown as Fig 2.2. Each cell connects with four neighboring **RM**s (the UP, DOWN, RIGHT, and LEFT). Every **RM** stores the ratio weight between two pixels. The detailed block diagram of two neighboring cells and RM in between them is demonstrated in Fig. 2.3, where *cell(i,j)* corresponds to the *i*th row and *j*th column cell, and u_{ij}^{p} is the input voltage of *cell(i,j)* of *p*th pattern. The block **T1** and **T3** are voltage-to-current converters. The block **T2D** is also a V-I converter except that its output is in absolute current form. Moreover, **T2D** can detect the sign of voltage state Vx_{ij} and stored separately. The block **W** uses the technique of current mirror to generate the output current of the cell by ratio (*1x, 1/2x, 1/3x,* and *1/4x*). The ratio current will be determined by the result of local counter, **Counter_L,** according to how many weights are preserved. The capacitor **Cw** stores the absolute weight during learning period and the resultant voltage Vc_w then transfers into current form and the current comparator **COMP** compares Vc_w with global maximum absolute weight in current. The comparator **COMP** is a simple current comparator which decides whether the ratio weight shall be kept. The output of **Counter_L** is to control the **W** to weight the output of each cell.

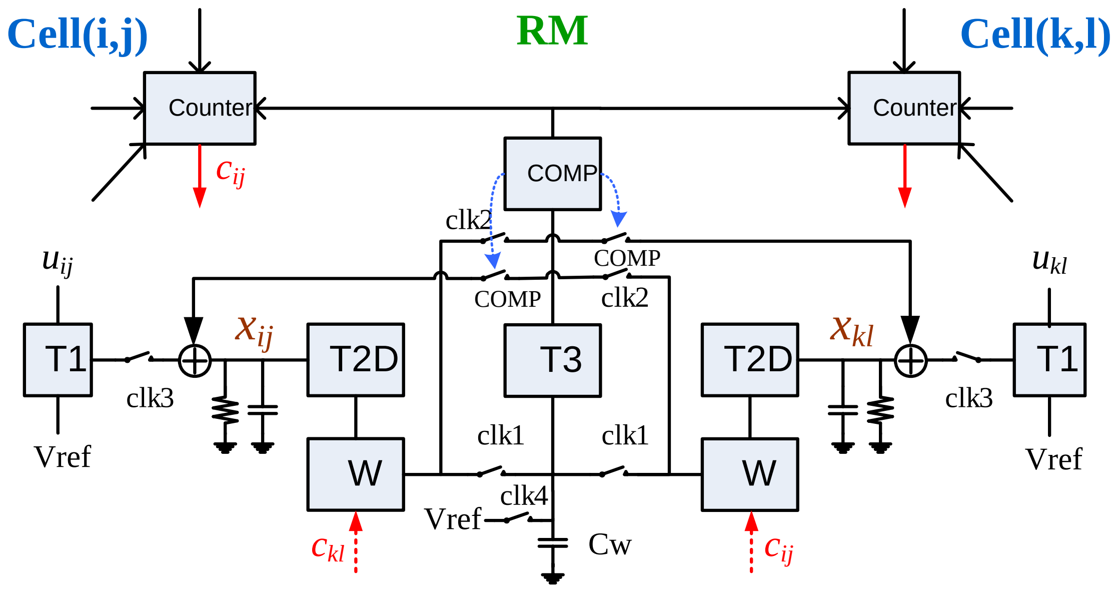


Fig 2.3 The detailed architecture of ARMCNN

The operation procedures of ARMCNN can be divided into three phases: learning, ratio-weights generation, and recognition. In the learning period, *clk1* is set to high and *clk2* is set to low. The architecture in learning phase is shown in Fig 2.4, where *cell(i,j)* input voltage of *p*th pattern u_{ij}^{p} is transferred into current Iu_{ij} and sent to node x_{ij}. Current Iu_{ij}^{p} can be expressed as

$$Iu_{ij} = \begin{cases} Iu_{sat} & when \quad u_{ij}^{p} > 1.9V \\ Gm_{T1} \times (u_{ij}^{p} - 1.5) & when \quad 1.5V < u_{ij}^{p} < 1.9V \\ 0 & when \quad u_{ij}^{p} = 1.5V \\ -Gm_{T1} \times (1.5 - u_{ij}^{p}) & when \quad 1.1V < u_{ij}^{p} < 1.5V \\ -Iu_{sat} & when \quad u_{ij}^{p} < 1.1V \end{cases} \qquad \text{Eq.(2.11)}$$

where Gm_{T1} is the transconductance of V-I Converter **T1**. The voltage level *1.5V* is defined as zero, so the current flows the opposite direction when u_{ij}^{p} is larger or smaller than *1.5V*. If u_{ij}^{p} gets larger than *1.9V* or less than *1.1V*, the output current Iu_{ij}^{p} of **T1** becomes saturated and remains at Iu_{sat}. In this work, Iu_{sat} is chosen to be the minimum required current to keep the circuit work properly and is about *6.5uA*.

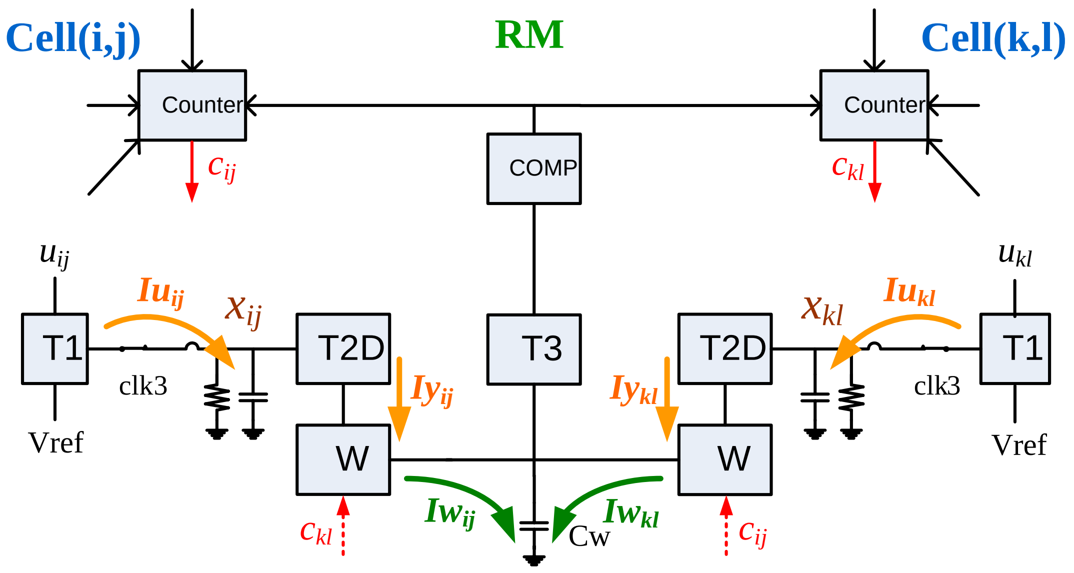


Fig 2.4 The architecture of AMCNN in learning phase

The current Iu_{sat} flows to the node x_{ij} and converts to a state voltage Vx_{ij}^{p} through the resistor *Rij* and capacitor *Cij*, which are the resistance and capacitance associated with the neuron *cell(i,j)*. Then **T2D** outputs an absolute current Iy_{ij}^{p} and a sign *sign*(Vx_{ij}^{p}) according to the stage voltage Vx_{ij}^{p}. Since the function of **T2D** is similar to **T1** and **T2D** has an absolute-value circuit, the output current Iy_{ij}^{p} and the *sign*(Iy_{ij}^{p}) can be written as

$$Iy_{ij}=\begin{cases} Iy_{sat} & when \quad u_{ij}^{p}>1.9V \\ Gm_{T2D}\times(Vx_{ij}^{p}-1.5) & when \quad 1.5V<u_{ij}^{p}<1.9V \\ 0 & when \quad u_{ij}^{p}=1.5V \\ -Gm_{T2D}\times(1.5-Vx_{ij}^{p}) & when \quad 1.1V<u_{ij}^{p}<1.5V \\ -Iy_{sat} & when \quad u_{ij}^{p}<1.1V \end{cases} \qquad \text{Eq.(2.12)}$$

$$sign(Vx_{ij})=\begin{cases} 0V & if\ Vx_{ij}^{p}<1.5V \\ 3V & if\ Vx_{ij}^{p}>1.5V \end{cases} \qquad \text{Eq.(2.13)}$$

where Gm_{T2D} is the trans-conductance of **T2D** and the current Iy_{sat} is the saturated output current of **T2D**. It is designed to be around *6.5uA* as well. The different between **T1** and **T2D** is that Iy_{ij} always flows in the same direction whether Vx_{ij} is larger or smaller than *1.5V*. Moreover, the sign of Vx_{ij} is detected by a detector in **T2D** and sent to the block **W**. According to the signs of two neighboring input voltage *Vxij* and *Vxkl*, the output current charges or discharges the capacitor *Cw*. **W** is the weighting circuit which transfers the input current Iy_{ij} into ratios: *1x, 1/2x, 1/3x, 1/4x*. In learning period, the block W is set to a default state, which multiplies Iy_{ij} by *1/4*. The reason of choosing *1/4x* as the default state is that it helps to control the length of learning time to charge or discharge the capacitor *Cw*. In addition, the capacitor *Cw* is a *1.5pF* poly-poly capacitor implemented on chip. The capacitance value of *Cw* is chosen as a compromise between weight storage time and capacitor chip area. The current Iy_{sat} is chosen as the smallest current that can have the V-I Converter operates regularly and is about *6.5uA* in this work. The smaller the current Iy_{sat} is, the better control of the value of Vw_{ijkl} on capacitor *Cw* can be. In this design, the learning time of a pattern is set to *100ns*.

After all patterns are learned, an absolute weight is generated and stored at the capacitor *Cw*. If two neighboring cells are having positive relationship, for example if they are both in black or both in white, the capacitor *Cw* between them is charged. On the other hand, if they are having negative relationship, for example they are one in black and the other in white, then the capacitor *Cw* is discharged. Thus the voltage Vw_{ijkl} stored on *Cw* can be expressed as

$$Vw_{ijkl}(p+1)=\begin{cases} Vw_{ijkl}(p)+\frac{1}{2}\frac{Iysat\times t}{Cw} & \text{when sign of } Vx_{ij} \text{ and } Vx_{kl} \text{ are the same} \\ Vw_{ijkl}(p)-\frac{1}{2}\frac{Iysat\times t}{Cw} & \text{when sign of } Vx_{ij} \text{ and } Vx_{kl} \text{ aren't the same} \end{cases} \qquad \text{Eq.(2.14)}$$

The voltage $Vw_{ijkl}(p)$ stored on the capacitor *Cw* represents the absolute weights after the *p*th pattern is learned. Since the output current of block W is set to $\frac{1}{4}Iy_{sat}$ in learning period, and there are two W blocks trying to charge or discharge the *Cw* at the same time, the voltage changing is $\frac{1}{2}\frac{Iysat\times t}{Cw}\left(=2\times\frac{1}{4}\frac{Iysat\times t}{Cw}\right)$. The time for each pattern to be learned is *100ns* in this design.

The block **T3** is also a V-I Converter, which transfers the voltage Vw_{ijkl} into the current form I_{T3} and sends this current to a simple current mode comparator **COMP**. The block **COMP** compares I_{T3} with a global maximum current I_{MAX}, which is corresponding to the largest value of absolute weight among the system. If I_{T3} is equal or larger than I_{MAX}, **COMP** outputs a logic “high” signal to the local counter **Counter_L**, which means the ratio weight between the two pixels should be preserved. Otherwise, if I_{T3} is less than I_{MAX}, **COMP** outputs a logic "low" signal to **Counter_L**, which means the relation between the two pixels is not strong enough and is of no interest.

The interconnection between **COMP**s and **Counter_L** is described in Fig 2.5. Since every cell connects with its four nearest neighbor cells, there are four **COMP**s in one cell. Every **COMP** sends out a logic signal to **Counter_L**. At the end of learning period, **Counter_L** counts how many "logic high" signals are given from the four **COMP**s. If there is (are) only one (two) "logic high", that means only one (two) ratio weight should be preserved, and so on. Then **Counter_L** controls the **W** to weight the output current of **T2D** as $1\times Iy_{ij}\left(\frac{1}{2}Iy_{ij}\right)$. Similarly, according to the total number of "logic high" signals are counted in one cell, the **Counter_L** may control the block **W** to weight the output current of **T2D** as

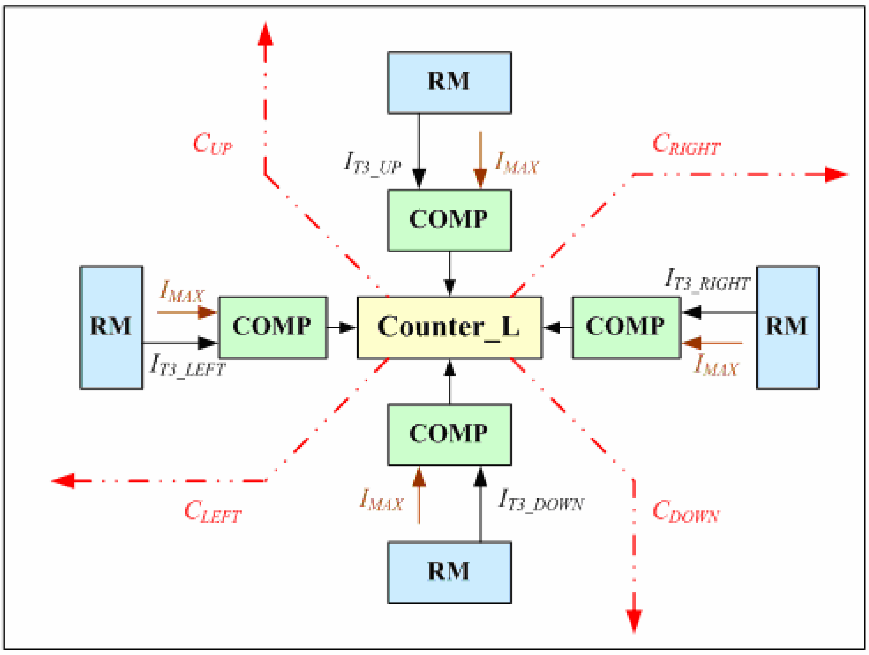


Fig 2.5 The inter-connection relationships between **COMP**s, **Counter_L** and **RM**s

$\frac{1}{3}Iy_{ij}$ or $\frac{1}{4}Iy_{ij}$. Moreover, the logic output of **COMP** controls the switch *sw_COMP* as shown in Fig 2.3. This behavior is known as no inter-relation between the two neighboring cells is generated, and thus the output contribution path of each other should be cut off. In other words, the ratio weight between two pixels is zero. For example, if the logic output of **COMP** between the two neighboring pixels is low, which means the ratio weight is zero, the switch *sw_COMP* should be turn off. Therefore, the output contribution between these two pixels is isolated in recognition period.

The ratio weight is generated as **Counter_L** counts up the total number of logic "high" from **COMP**s and controls the block **W** with appropriate weighting. After that, the operation enters into recognition phase. The architecture in recognition period is shown as Fig 2.6. As shown in Fig 2.6, *clk1* is set to low and *clk2* is set to high. The state of switches *sw_COMP* is controlled by **COMP**. A noisy image pattern is inputted to perform recognition and recovery, where u_{ij}^{noi} and u_{kl}^{noi} represents the input voltage of noisy pattern of *cell(i,j)* and *cell(k,l)* respectively. They are inputted to **T1** and transfer to currents Iu_{ij}^{noi} and Iu_{kl}^{noi}. These currents then convert to state voltages Vx_{ij}^{noi} and Vx_{kl}^{noi} through the resistor R_{ij} (R_{kl}) and capacitor C_{ij} (C_{kl}). **T2D** converts state voltages into current Iy_{ij}^{noi} and Iy_{kl}^{noi}. In accordance with the ratio weights generated previously, the output current of each cell is weighted as *1x*, *1/2x*, *1/3x*, *1/4x*, and *0x*, and contributes to its corresponding neighbor cells. For instance, if the weight is set to 1x, it means only one (two) of the four neighbor cells is correlated to this cell. Thus only one (two) neighbor cell will contribute its current output to the cell *cell(i,j)*, and so on. In the case where the two neighboring cells have no correlation to each other, the **COMP** will output a logic "low" signal to the local counter, and this signal will turn off the output contribution path between them as well.

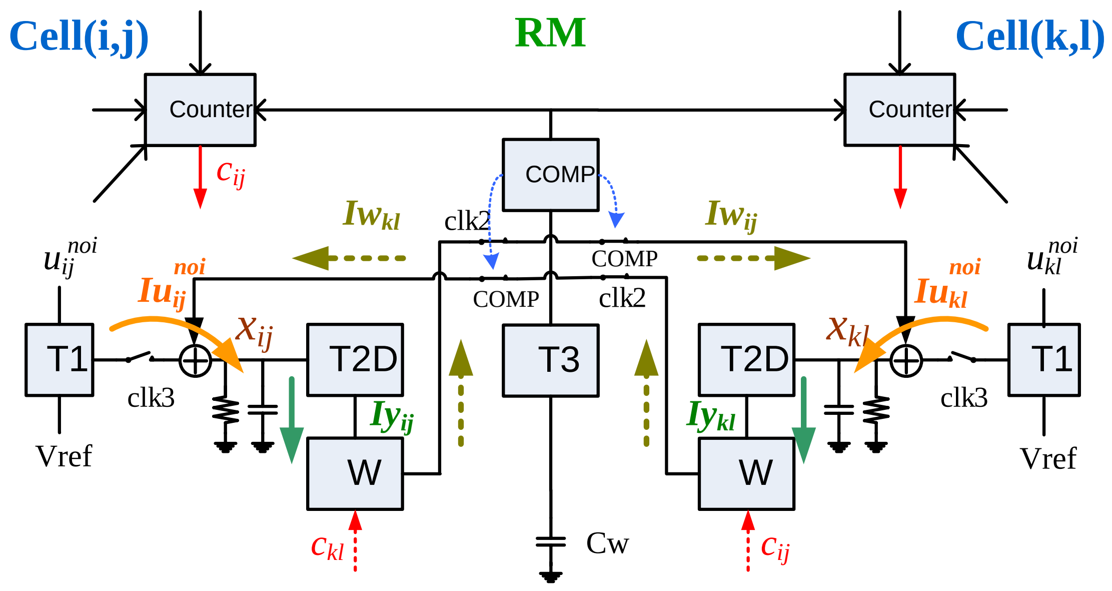


Fig 2.6 Architecture of ARMCNN in recognition period

According to KCL, the dynamic equations of the cell state voltage $Vx_{ij}(t)$ and its derivative $\dot{Vx_{ij}}(t)$ can be expressed as Eq.(2.1) and Eq.(2.2). In addition to that, the weighting of output currents can be derived from the following equation:

$$\begin{aligned} &Iw_{kl} = a_{ijkl} \times Iy_{kl} \\ &a_{ijkl} \in 1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, or\ 0 \\ &k,l \in i(j-1), i(j+1), (i-1)j, or\ (i+1)j \end{aligned} \qquad \text{Eq.(2.15)}$$

where a_{ijkl} is the template A ratio weight coefficient and generated by the block **W**. The coefficient a_{ijkl} can be *1*, *1/2*, *1/3*, *1/4* or *0*, which represents the number of preserved weights for the cell is *1*, *2*, *3*, *4* or *0*, respectively. The current Iw_{kl} is the resultant current that contributes to *cell(i,j)*. It is equal to the output current Iy_{kl} of neighbor cell times the coefiicient w_{ijkl}.

2.2 Circuit Implementation

In this work, several circuits have been employed. The voltage-to-current converter and current weighting circuit are discussed in section 2.2.1. Then a simple current mode comparator is described in section 2.2.2. Section 2.2.3 talks about some digital components such as counter, decoder, voltage detector and driver. They are necessary for some calculation purpose. At last, the shift registers, which functions as the input pattern interface, and the output stage circuits are described.

2.2.1 V-I Converter

As shown in Fig 2.3, The block **T1**, **T2D**, and **T3** are all V-to-I converters. The circuit of **T1** and state resistance R_{ij} / capacitance C_{ij} are implemented as Fig. 2.7, where the MOS dimension is written next to the MOS name. The unit of MOS dimension is in micro-meter (um). In Fig 2.7, the left side of this circuit is a differential pair structure with the cascode current mirror, and the right side of circuit is the state resistor / capacitor, formed by diode load (*MR1* and *MR2*) and MOS capacitor (*Mc*) respectively. The purpose of state resistors is to limit the operating range. The voltage *Vb1* is a constant bias voltage, which is set to *1.5V*. The reference voltage *Vref* is used to compare with input voltage and sets to *1.5V*. If the input voltage *Vin* is larger than *Vref*, the output current *Io* flows from left to right (*M4* $\rightarrow$ *M6* $\rightarrow$ *MR2*) and causes the stage voltage Vx_{ij} raise to *1.9V*. On the other hand, if the input voltage *Vin* is smaller than *Vref*, then the output current *Io* flows from right to left (*MR1* $\rightarrow$ *M2* $\rightarrow$ *M7*) and causes the stage voltage Vx_{ij} drop to *1.1V*. In other words, the state voltage Vx_{ij} is ranged from *1.1V* to *1.9V*.

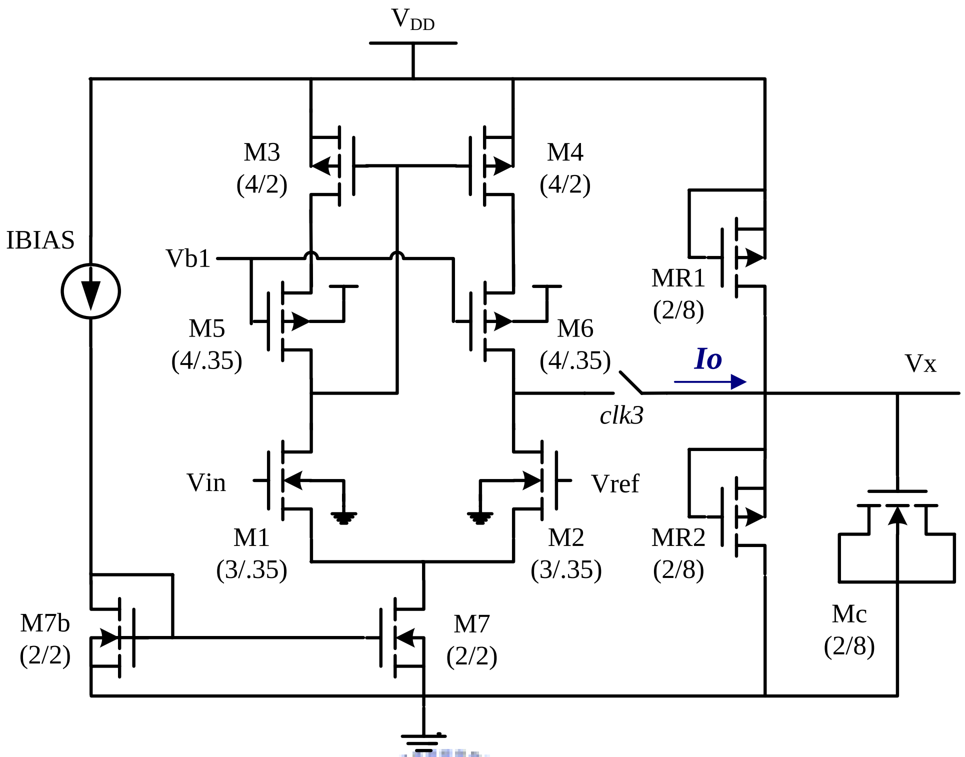


Fig 2.7 T1: Voltage to current converter

Fig 2.8 is the circuit of **T2D** block, a voltage to absolute current converter. The left side of **T2D** is a differential pair, which is the same as **T1**, and the right side of **T2D** is the absolute output current structure. The constant bias voltage *Vb2* is *1.5V*, and the constant bias voltage *Vb1* and the reference voltage *Vref* are the same as in **T1**. The operating principle of **T2D** is that when the input voltage *Vin* is larger than *Vref*, the MOS *M2* is cutoff and the current flows from MOS *M3* through *M5* and *M1* to *M7*. The cascode current mirror (*M3 ~ M6*) mirrors this current to *M4* and to the right side of **T2D** since MOS *M2* is cutoff. Note that the parasitic capacitor at the source of *M10* (the input of inverter) is charged to high. Consequently, the MOS *M11* is shorted and *M10* is cutoff. Therefore, the current *Io* flows through MOS *M12* to *M14*. The other cascode current mirror (*M12 ~ M15*) forces MOS *M8* to flow the same amount of current as MOS *M14* does. At last the absolute output current *Io,abs* is mirrored from the MOS *M8* to *M94*. Note that the switches M10 and M11 will not turn on at the same time. This can be seen from the equation that for MOS M10 to be on: Va – Vp > Vth, that is Vp < Va – Vth. But for MOS M11 to be on: Vp – Va > Vth, that is Vp > Va + Vth. Accordingly, The switches M10 and M11 will not turn on at the same time. In addition, if the differential pair provides no current flow into or out from node Vp, both switches are off.

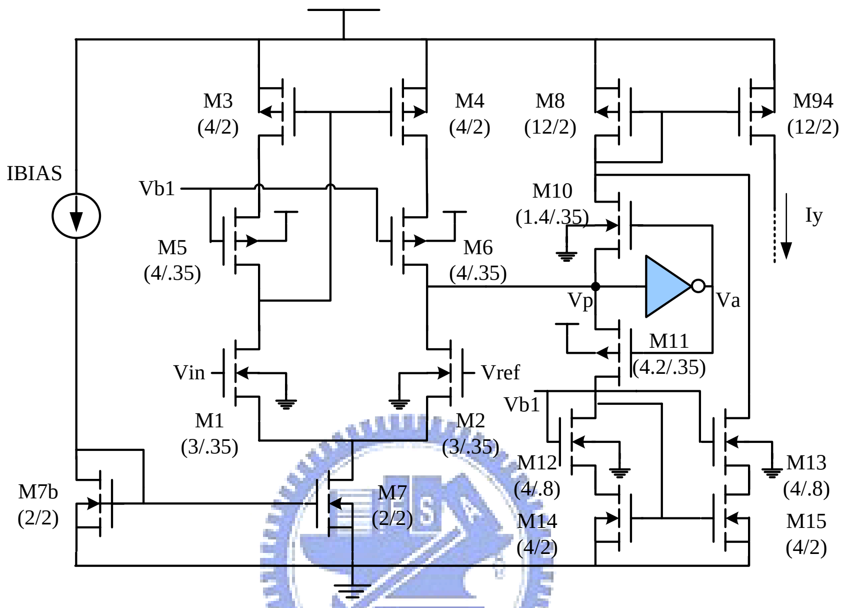


Fig 2.8 T2D: Voltage to absolute current converter

On the other hand, if the input voltage *Vin* is smaller than *Vref*, the MOS *M1* is cutoff and the cascode current mirror (*M3 ~ M6*) is off. Since *M1* is operating in cutoff region and the current source *M7* is forcing a current of *6.5uA* to flow to ground, the direction of the output current *Io* is from right to left. Moreover, the parasitic capacitor at the source of *M10* (the input of inverter) is discharged to low. As a consequent, the MOS *M10* is shorted and *M11* is cutoff. The other cascode current mirror (*M12 ~ M15*) is off as well. A current of *6.5uA* is flowing from *M8* through *M10* and *M2* to *M7*. The absolute output current *Io,abs* is also mirrored from the MOS *M8* to *M94*. Note that in the both cases, the absolute output current *Io,abs* is flowing in the same direction whether the input voltage *Vin* is larger than *Vref* or not. Therefore, the **T2D** is called a voltage to absolute current converter.

The weight circuit is shown in Fig 2.9, which is to generate the desired ratio of the output current from **T2D**. The possible current ratios are: *1x*, *1/2x*, *1/3x*, *1/4x*. In practical design, the weight circuit is directly combined with **T2D** to form the desired ratio. Note that the MOS *M94* in Fig 2.8 and the MOS *M94* in Fig 2.9 are the same. Since we wish the

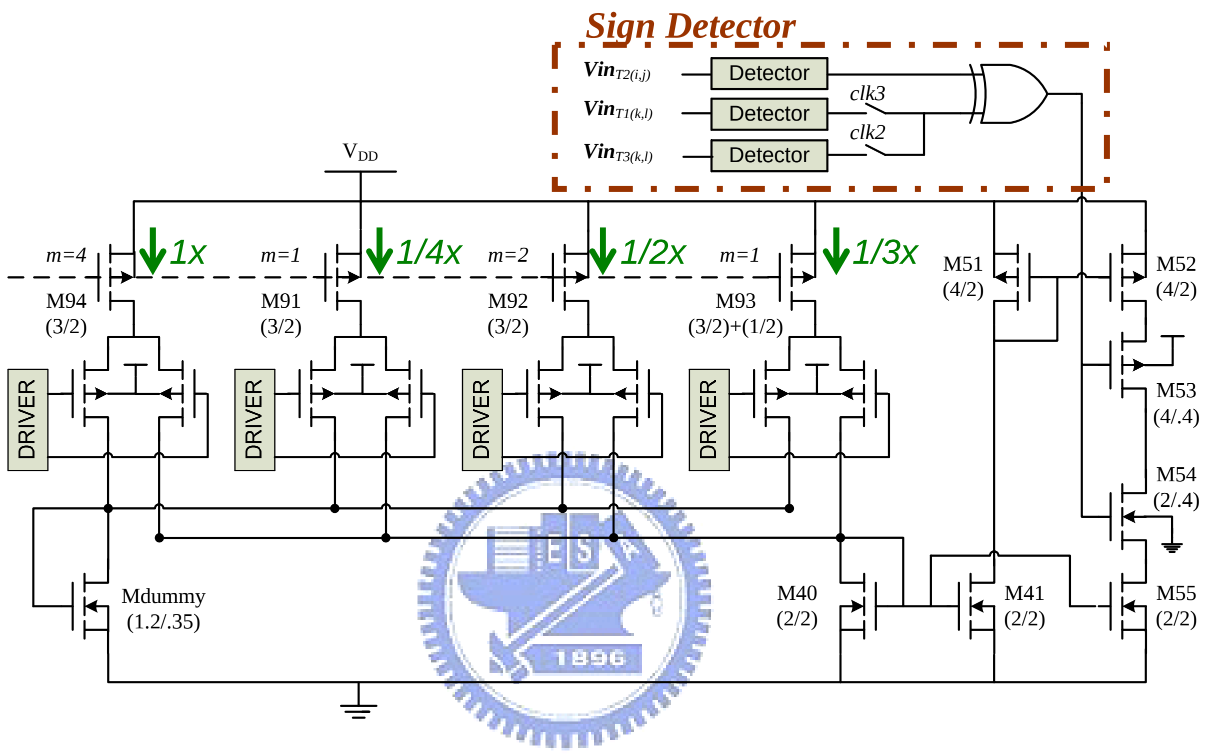


Fig 2.9 Weight: Generation of ratio current

generated current ratios to be précised, the MOS *M91*, *M92*, *M93*, and *M94* do not use minimum length so as to avoid the impact of channel length modulation. The four current paths are controlled by the circuit **DRIVER**. There is at most one path flowing to the MOS *M40* at a time and the other three paths are conducting to the ground through a dummy MOS *Mdummy*. Note that during the period of pattern transferring, all four paths are conducted to the ground through the MOS *Mdummy* to ensure no charging / discharging behavior toward the capacitor *Cw*. The use of *Mdummy* and **DRIVER** corrects the mistake made by the previous design. The upper part of weight circuit is a sign detector, which detect the sign of state voltages of the neighbor neuron and itself. If two neighbor neurons are having the same sign, then *XOR* outputs a logic low to turn on the MOS *M52* and *M53* and turn off the MOS *M54* and *M55*. This will charge the capacitor *Cw*. On the other hand, if two neighbor neurons are having different signs, *XOR* outputs a logic high to turn on the MOS *M54* and *M55* and turn off the MOS *M52* and *M53*. This will discharge the capacitor *Cw*. The detailed circuitry of **Detector** and **DRIVER** will be described in section 2.2.3.

2.2.2 Comparator (with T3)

The V-I converter **T3** is the same as **T2D** except that **T3** is followed by a current mode comparator and **T2D** is followed by a weight circuit. The schematic diagram of **T3** with a current mode comparator is shown in Fig 2.10. The output of the V-I converter **T3** is sent directly to the comparator **COMP**. The comparator we choose here is a simple current mode

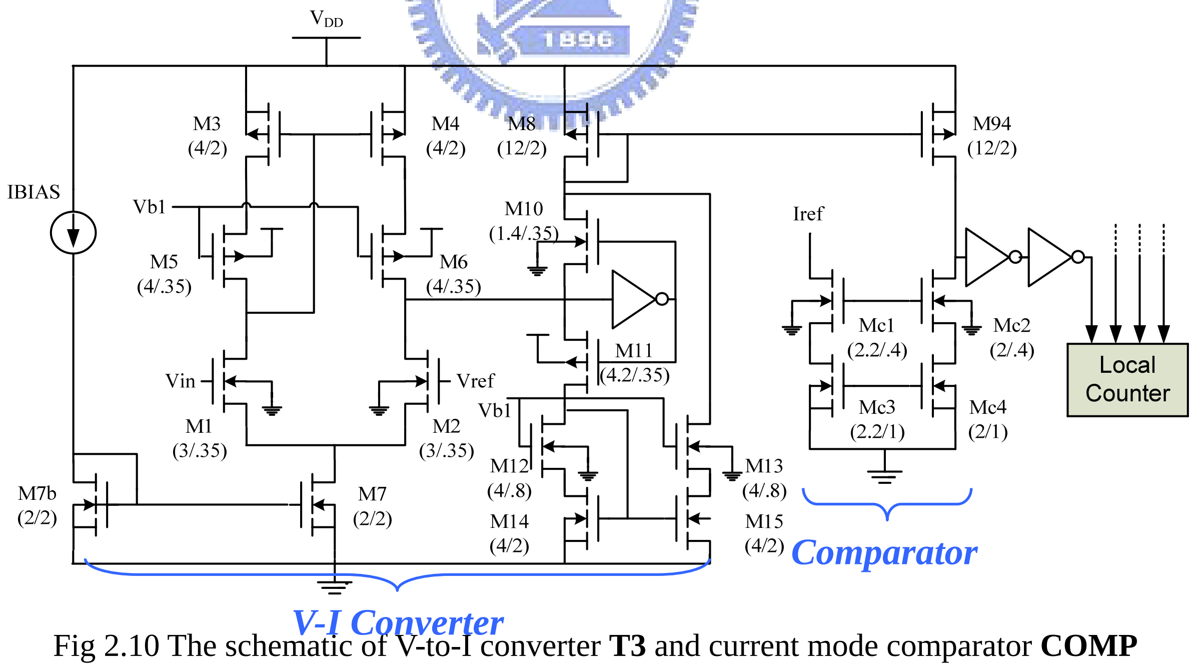


Fig 2.10 The schematic of V-to-I converter **T3** and current mode comparator **COMP**

comparator. The reason we go for simple structure is to save the area of the whole chip. Its operating principle is as following, if the output current of **T3,** Io_{T3}, is larger than or equal to the global maximum current, *Imax*, which means the absolute weight should be preserved, the logic output of **COMP** is high. On the other hand, if the output current of **T3** is smaller than the global maximum current *Imax*, which means the absolute weight is of no significance, the logic output of **COMP** is low. Since we want to achieve that the output of **COMP** is high if Io_{T3} is equal to *Imax*, the sizes of *Mn2* and *Mn4* are designed to be a little smaller than *Mn1* and *Mn2*. Doing so makes the logic output is high even if Io_{T3} equals to *Imax*.

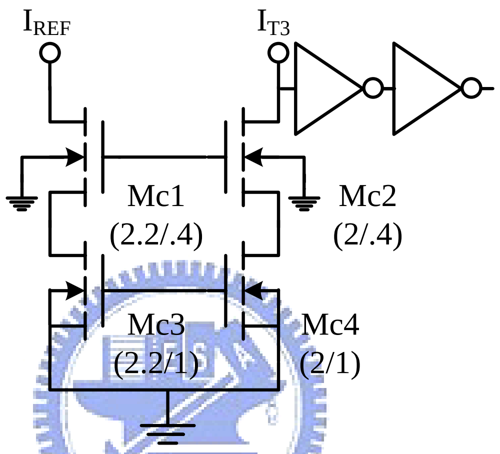


Fig 2.11 The dimension of the current mode comparator **COMP**

2.2.3 Digital Components

In the ARMCNN system, there are also some digital components being employed to achieve desired functions. For instance, the local counter **Counter_L** is to count up the total number of preserved weights in a cell. The global counter **Counter_G** is to control the switches sw1 ~ sw6, which is described later in this section. The voltage detector **Detector** is to detect the value of state voltage and output a logic signal (low or high). The last one is the weight selection circuit **Driver**. **Driver** decides which of the four current paths are conducted. The decision made is depending on the 2-bit logic output of the local counter **Counter_L**. In this section, all digital circuitries are discussed one by one in detail.

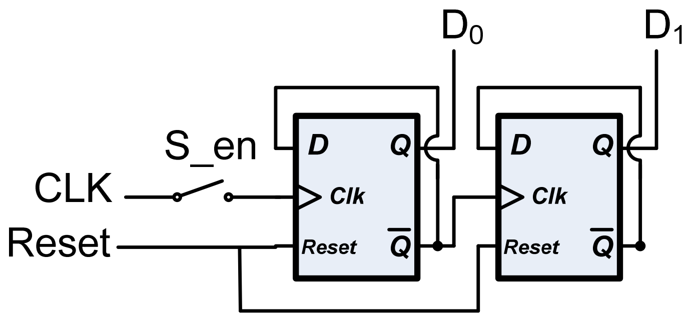


Fig 2.12 The schematic diagram of the counters in this chip

Counter

In Fig 2.13, the counters **Counter_L** and **Counter_G** are both formed by two asynchronous reset flip-flops as shown in Fig 2.12. The schematic diagram of the asynchronous flip flop is demonstrated in Fig 2.14. Instead of using digital flip-flop (DFF), using asynchronous reset flop-flops can ensure correct function under slow operating speed. In addition, it does not have the static power consumption problem as DFF does. The switch S_en enables the counting operation and it can be described in Fig 2.15, where *CLK* is clock signal and *RST* is reset signal. If the signal *RST* is set to low, *b0* and *b1* are always low. The signal *S_en* must set to high during the counting operation. If not, *b0* and *b1* do not change even if *CLK* is oscillating. Note that signal *b0* represents the logic output of the counter's LSB bit while signal *b0_bar* represents its compliment and signal *b1* represents the logic output of the counter's MSB bit while signal *b1_bar* represents its compliment.

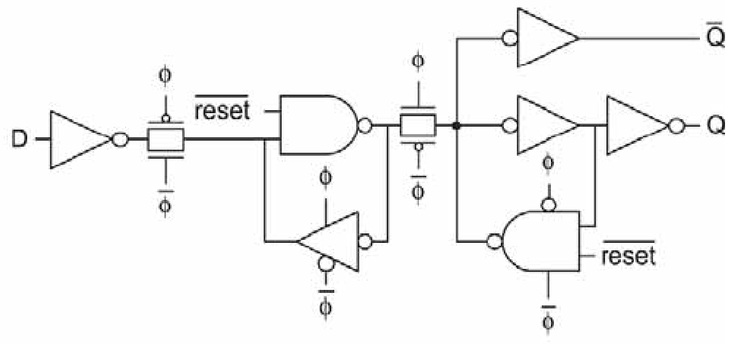
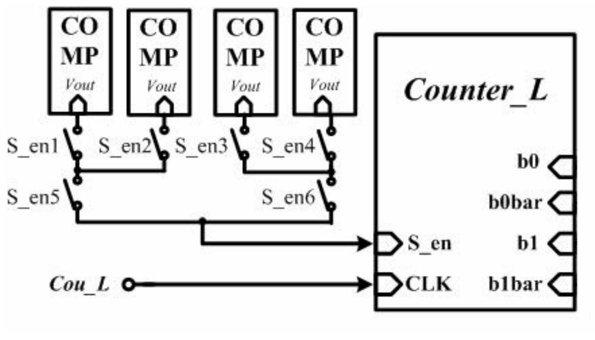


Fig 2.13 The schematic diagram of the asynchronous flip-flop



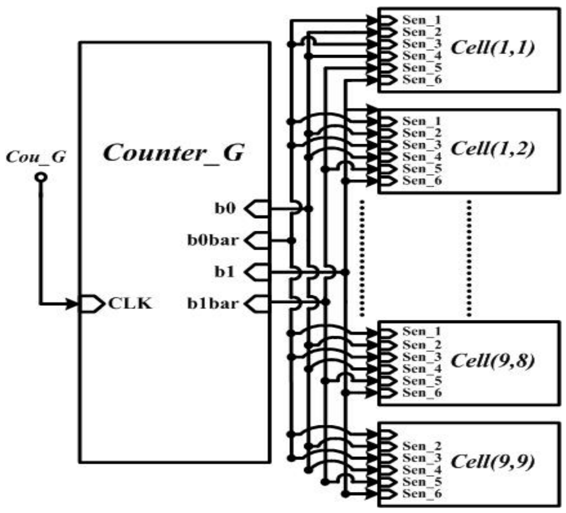


Fig 2.14 (a) Local Counter (b) Global Counter

There are six switches used in the local counters and they are controlled by the combination logic from the global counter. This can be simplified by using only four switches in the local counters. The four outputs, “b0, b0bar, b1, b1bar”, can be used to control the four switches in the local counters. This not only reduce the number of switches used, but also simplified the layout routing.

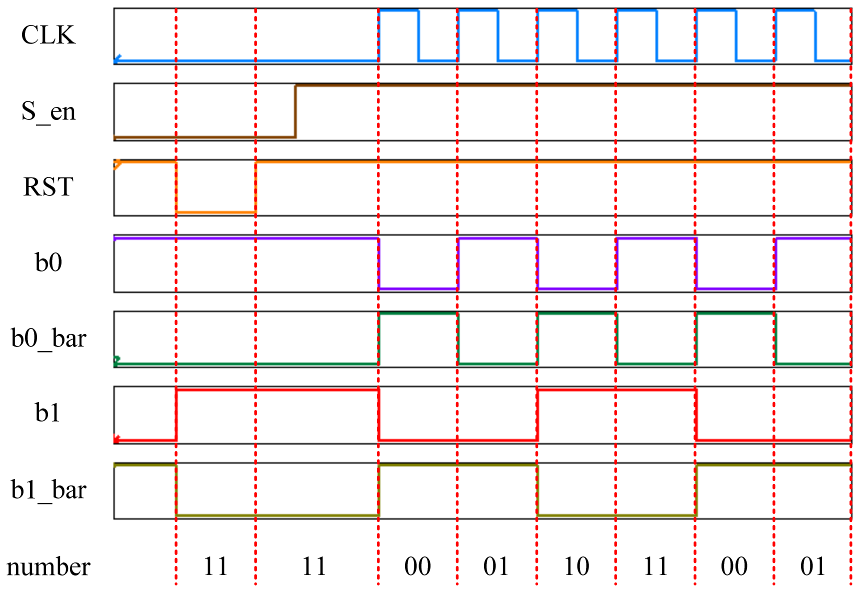


Fig 2.15 A counting example of the counter

Detector

The detector is used to detect the voltage level of each state voltage and transform to a logic signal (either high or low). As a result, the logic signal can be handled by the combinational logic. The detector is formed by a tri-state element and a simple inverter as shown in Fig 2.16. Since the input voltage to the detector is an analog signal ranging from *1V ~ 2V*, a simple inverter train structure will lead to constant current leakage because both the PMOS and the NMOS are on at the same time for the first inverter. Therefore, the use of tri-state buffer avoids the problem of the constant current leakage but at the expense of an additional controlling signal *VB1*.

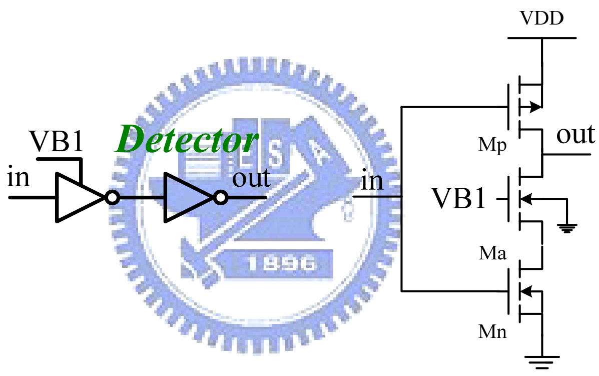


Fig 2.16 The schematic diagram of the detector and the tri-state buffer

Driver

The driver circuit is to control the current ratio paths of the weight circuit. It allows only one path to charge / discharge the capacitor *Cw* while the other three paths are conducted to the ground. In addition, during the pattern transferring period, the capacitor *Cw* should not be charged / discharged. Therefore, the driver circuit should conduct all four paths to the ground. This can be done by ORing the signals *clk1* and *clk2*. The signals *clk1* and *clk2* represent the learning period and the recognition period respectively. Consequently, ORing *clk1* and *clk2* outputs a logic signal *control* and it is true when either of them is logic high, which means the signal *control* will be high during learning and recognition only. The signal *control* will be low during the pattern transferring period. Please refer to Fig 2.17.

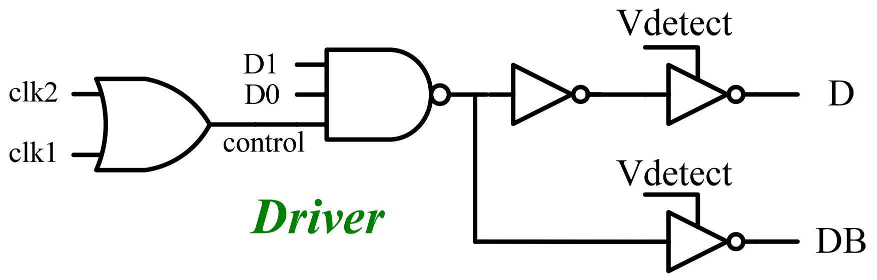


Fig 2.17 The schematic diagram of the driver circuit

A NAND gate generates a logic low only if its inputs are all low. The three inputs of the NAND gate are: *D1*, *D0*, and *control*. The signal *control* is described earlier and the signals *D1* and *D0* are the output result of the local counter **Counter_L**. The combination of the signals *D1* and *D0* and their compliments decides which of the four ratio paths are conducted to the capacitor *Cw* while the other three ratio paths are conducted to the ground. Note that the logic output *D* is to control the path to the capacitor *Cw*, and logic output *DB* is to control the path to the ground.

As shown in Fig 2.18, the default state (*D1, D0: 11*) of the weight circuit is set to Iy_{ij} multiplies by *1/4*. All states returns to default state if the signal reset is triggered. According to the output of the counter, the state is changed as following:

If counts one, the state changes to (*D1, D0: 00*), with the ratio set to Iy_{ij} by 1.

If counts two, the state changes to (*D1, D0: 01*), with the ratio set to Iy_{ij} by $\frac{1}{2}$.

If counts three, the state changes to (*D1, D0: 10*), with the ratio set to Iy_{ij} by $\frac{1}{3}$.

If counts four, state changes back (*D1, D0: 11*), with the ratio set to Iy_{ij} by $\frac{1}{4}$.

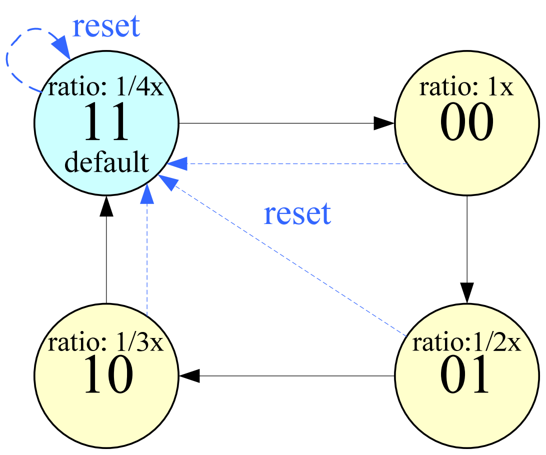


Fig 2.18 The state diagram with corresponding ratios

Decoder

Since there is only one output pin used in this work, two 4-bit decoders, **Decoder_Row** and **Decoder_Column**, are implemented to control the switches of the output stage. It allows only one of the state voltages of the 9x9 neurons to output at a time. **Decoder_Column** controls column switches SW_{C11} ~ SW_{C19} (SW_{C21} ~ SW_{C29}, SW_{C31} ~ SW_{C39} …etc.) while **Deocder_Row** controls row switches SW_{R1} ~ SW_{R9}. This structure allows every pixel to be read out one by one. The schematics for both decoders are the same. As shown in Fig 2.19, the decoder is formed by nine 4-input AND gates and four inverter-string type of buffers.

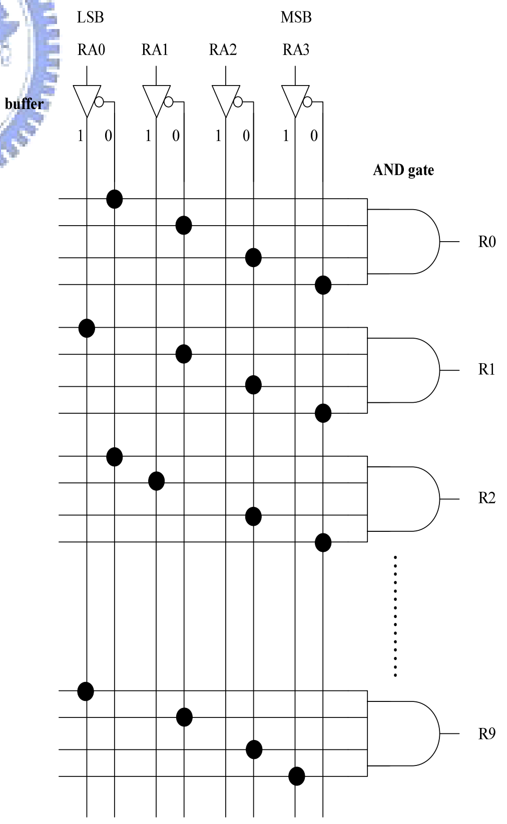


Fig 2.19 The schematic of the 4-bit decoders

2.2.4 Output Stage and Input Pattern Interface

In this work, the circuit of the output stage is used because there is only one pin available for output. Therefore, the 9x9 neurons (total 81 neurons) share the same output pin. The schematic diagram of the output stage is shown in Fig 2.20, where the nodes $x_{11} \sim x_{99}$ are the node x_{ij} in Fig 2.3. The MOS *M11* ~ *M99* perform as level shifters to drive the parasitical capacitance of the switches and metal line. The switches $SW_{C11} \sim SW_{C99}$ and $SW_{R1} \sim SW_{R9}$ are controlled by **Decoder_Column** and **Decoder_Row** respectively. The state voltage of each pixel is readout one by one. The arrow with a circle enclosed represents a current source. In the output stage, all pixels share the same current source. Since only one pixel is conducted at a time, one current source is enough for this design. Furthermore, using few current sources saves more power consumption. The unit-gain buffer is a negative feedback OP-amp and is used to drive the loading of the output pad. The circuit of the unit gain buffer is described in Fig 2.21.

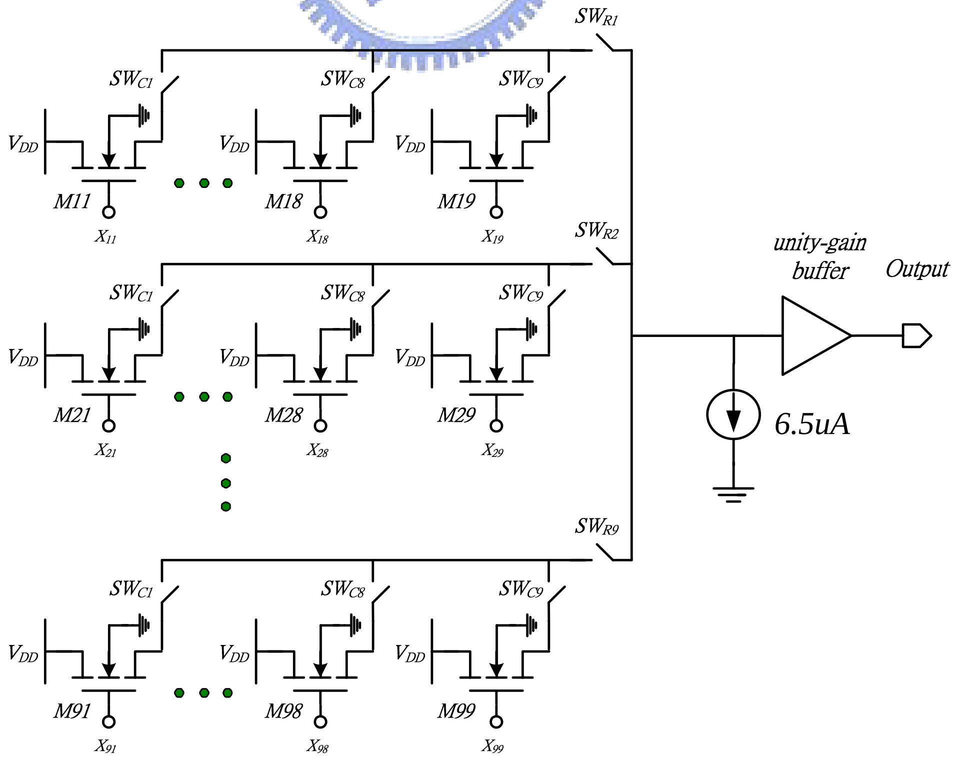


Fig 2.20 The schematic diagram of the output stage

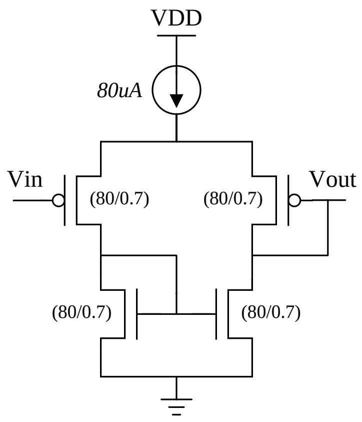


Fig 2.21 The circuit diagram of the unit gain buffer

In this work, since we have to input any arbitrary learning patterns, the shift registers are used as pattern input interface. As shown in Fig 2.22, each block represents a static flip-flop. The operation of learning period is demonstrated here. In the beginning of the learning period, the control signals *clk3* and *newp* are turned on and the node *ptn* inputs the learning patterns pixel by pixel. After the clock of flip-flops, *DFF*, oscillates nine times (because the cell array is nine), the signal *pin* turns on to input the learning pattern into each pixel. Before the signal *pin* turns on, the signal *newp* turns off to prevent the pattern changes due to a glitch. After the pattern is learned, the operation above is repeated again so as to input the next pattern. Depending on the total number of patterns to be learned, the operation above is repeated until all patterns are learned.

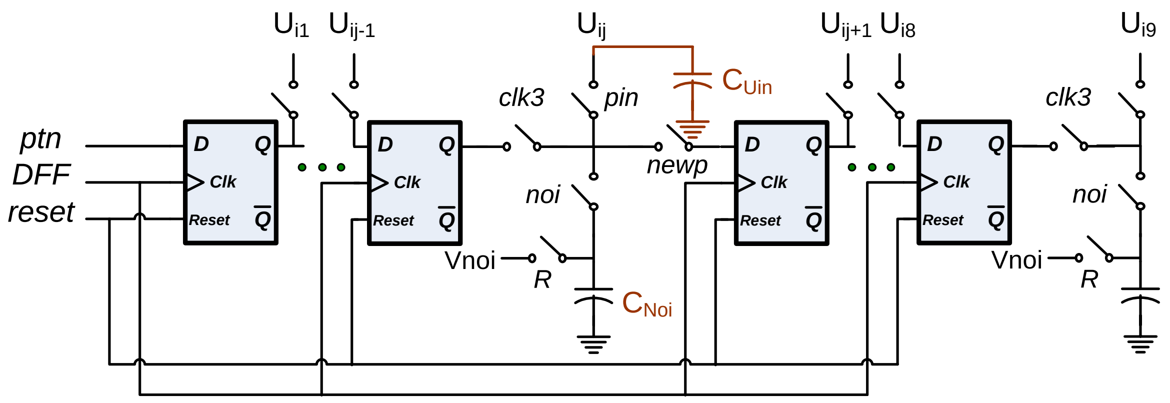


Fig 2.22 The pattern input interface formed by the shift registers

Fig 2.23 (a) is one part of Fig 2.22, which is the input stage of single pixel. Fig 2.23(b) shows how to mix the innocent learning pattern with noise in recognition period. The capacitor C_{UIN} is an embedded poly-poly capacitance with magnitude *0.45pF* and the capacitor C_{NOI} is also an embedded poly-poly capacitance with magnitude *0.1pF*. In the learning period, the capacitor C_{NOI} is pre-charged to a voltage level V_{NOI} and the control signal *noi* always turns off. In the beginning of the recognition period, the noisy pattern is inputted to perform recognition and recovery. This is done by storing the innocent learning pattern in the shift registers first, then turning off the signal *clk3* to isolate each static flip-flop. While the signal *noi* turns on, the behavior of charge sharing occurs between the capacitors C_{UIN} and C_{NOI}. As a result, the resultant voltage on the node *Uij* can be achieved by adjusting the capacitance ratio of C_{UIN} and C_{NOI}.

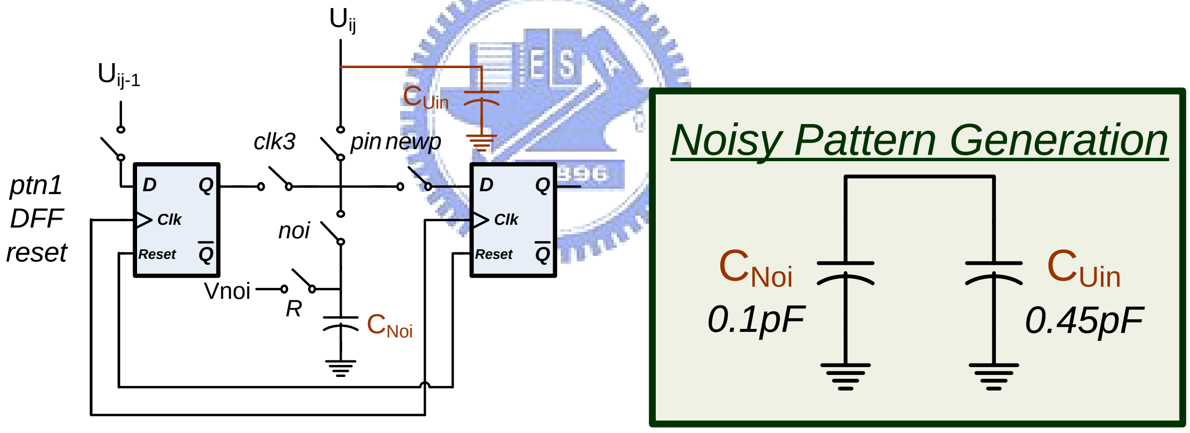


Fig 2.23 (a) The input stage of a pixel (b) The structure used to mix the pattern with noise

2.2.5 Circuit for Global Maximum Absolute Weight Determination

The preservation of the absolute weight is determined by comparing it with a global maximum absolute weight. If the weight is less than the global maximum absolute weight, the weight is set to zero. Otherwise, the weight will be preserved. As a consequent, a circuit is designed to determine the global maximum absolute weight. The global maximum absolute weight is in current form since the comparator **COMP** is a current comparator. This circuit is a replica of the cell we used in the system, except it's a simplified version which consists circuits that determined the maximum current. The circuits are: the V-I converter **T2D**, the current weighting circuit **W** and the capacitor **Cw**.

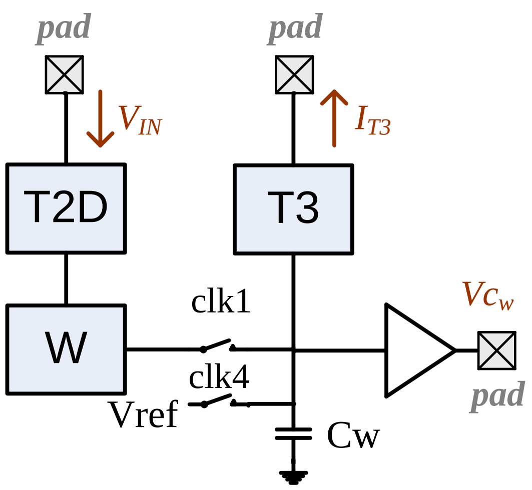


Fig 2.24 The circuit to determine the global maximum absolute weight

To generate the absolute weights, we set V_{IN} = 3V (or 0V) and turn the switch Msw on and off three times. The resultant voltage stored on capacitor Cw (Vcw) is the global maximum absolute weight and can be readout through a unity-gain buffer to pad Vcw. The voltage Vcw then converts into current I_{T3} through block T3. In measurement, an Opamp together with a resistor to form a close loop can be used to measure the current I_{T3},

$$I_{T3} = \frac{V_{T3} - Vref}{R} \qquad \text{Eq.(2.16)}$$

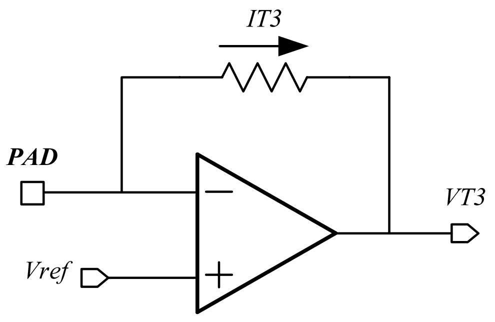


Fig 2.25 The off-chip current measuring circuit

CHAPTER 3

SIMULATION RESULTS

3.1 Behavior Simulation Results [27]

Base upon the mathematical equations of a 9x9 resolution ARMCNN system, behavior simulations are performed using the program C/C++. One hundred noisy patterns of a character are generated for a fixed standard deviation of noise level. The recognition rate (RR) of a group of *m* patterns is defined to be the number of successful recognitions divided by $100 \times m$ at a fixed standard deviation of noise level. The patterns for learning are Chinese characters "one", "two", and "four", which are shown in Fig 3.1. The recognition rates of both the traditional RMCNN and autonomous RMCNN are compared in section 3.1.1.

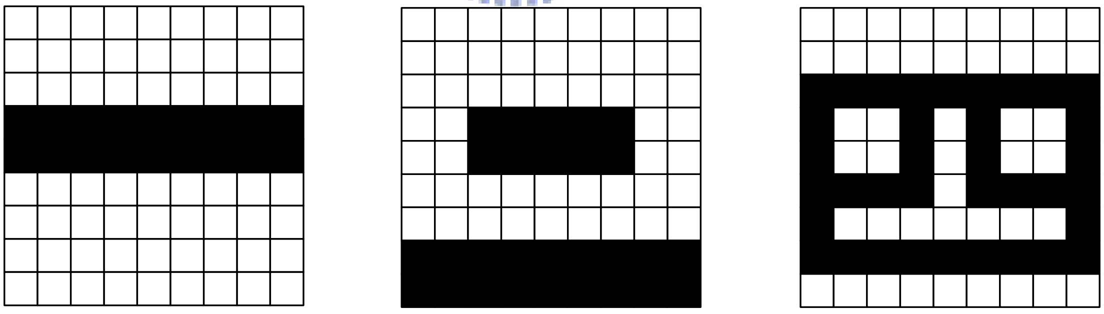


Fig 3.1 The Chinese Characters "one", "two", and "four"

The simulated recognition rates (RR) versus the standard deviation of noise level are shown in Fig 3.2. It can be seen from Fig 3.2 that the proposed ARMCNN can recognize at most four patterns. If five patterns are learned and recognized, the RR drops to zero. The RR of ARMCNN is slightly increased as the number of patterns to be recognized is decreased. In comparison with the traditional RMCNN, the RR of ARMCNN is much greater than that of RMCNN for $\sigma > 0.3$.

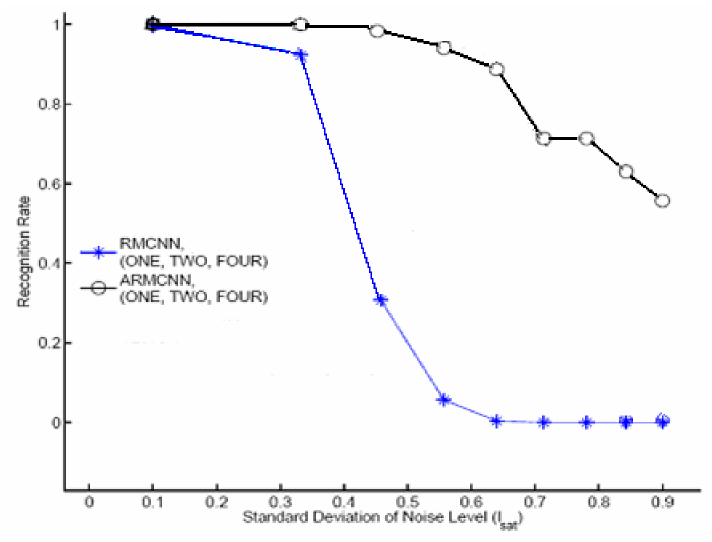


Fig 3.2 The recognition rates of recognizing Chinese characters ONE, TWO, FOUR, and at different input approaches.

3.2 Hspice Simulation Results

The simulation of V-I Converter ***T1*** and the state resistor / capacitor (*Rij* / *Cij*) is shown in Fig 3.3. The tail current is chosen as the minimum current that can have the operation function properly. In this work, the tail current is set as *5uA*. The transferring curve shows that the input voltage of ***T1*** is linear between *1.2V* ~ *1.8V*. If the input voltage range of ***T1*** exceeds this range (i.e. smaller than *1.2V* or larger than *1.8V*), the output voltage of ***T1*** is saturated (i.e. *1.2V* or *1.8V*, respectively). As a consequent, the voltage level *1.8V (1.2V)* is defined as *+1 (-1)*. The output voltage is then used as the input voltage for ***T2D***.

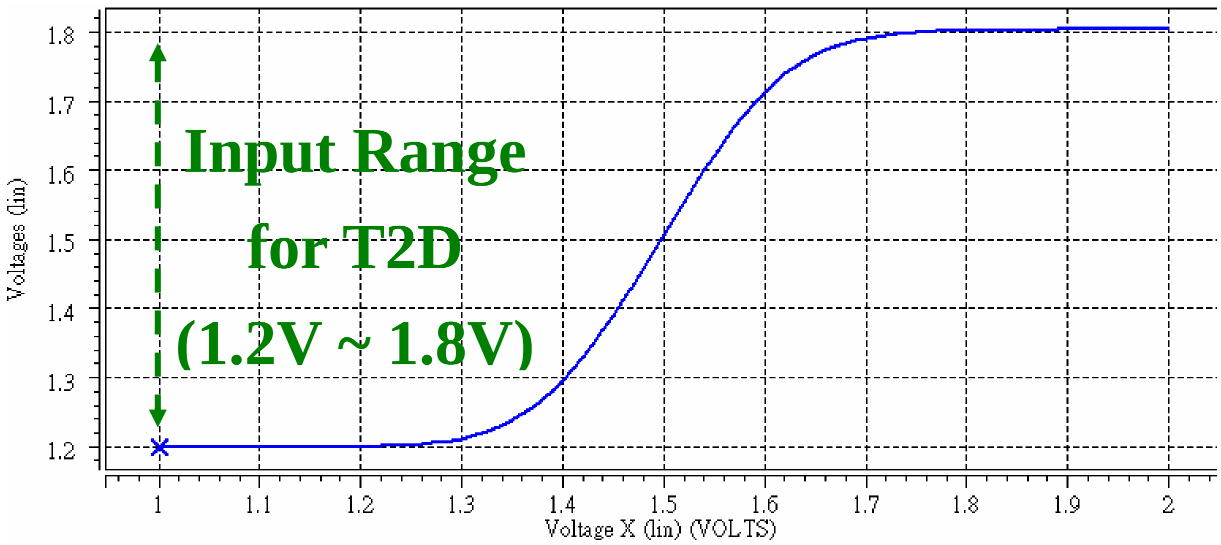


Fig 3.3 The Transferring curve of the V-I Converter ***T1*** and State Resistor / Capacitor

Fig 3.4 describes the simulation results of V-I Converter ***T2D***. Since the output of ***T2D*** is in the form of the absolute current, the flowing direction of the output current is the same whether the input voltage of ***T2D*** is larger or smaller than *1.5V*. In addition, the transferring curve is symmetric at voltage *1.5V*. Note that the output current of ***T2D*** and the output result of the current multiplied by one in weighting circuit are of a little voltage difference. This is due to the effect of different *Vds* seen by the PMOS *M8* in ***T2D*** and the PMOS *M94* in ***W*** block.

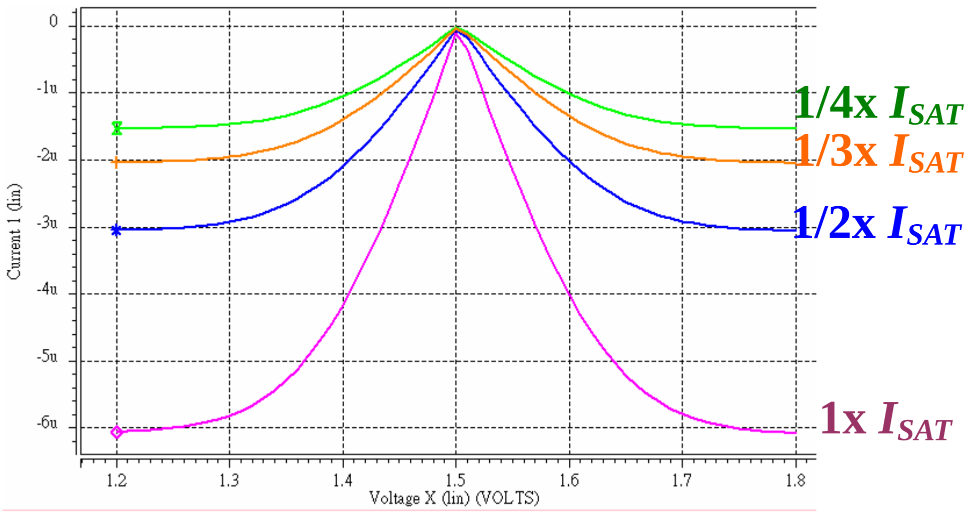


Fig 3.4 The Transferring curve of the V-I Converter ***T2D*** and the Weighting Circuit ***W***

The DC simulation results of the current comparator **COMP** is shown in Fig 3.5. The input current I_{IN} is swept and the reference current I_{REF} is kept as constant. The figure shows that as the input current gets larger than the reference current, the comparator generates a logic ‘high’ signal *(Vout2)* to the local counter. On the other hand, as the input current is less than the reference current, the comparator generates a logic ‘low’ signal to the local counter. The comparator is purposely designed such that if the absolute-weight current is equal to the reference current, the ratio weights is preserved as well. Therefore, the MOS dimension of the comparator is designed to output a logic ‘high’ signal even I_{IN} is slightly smaller than I_{REF}.

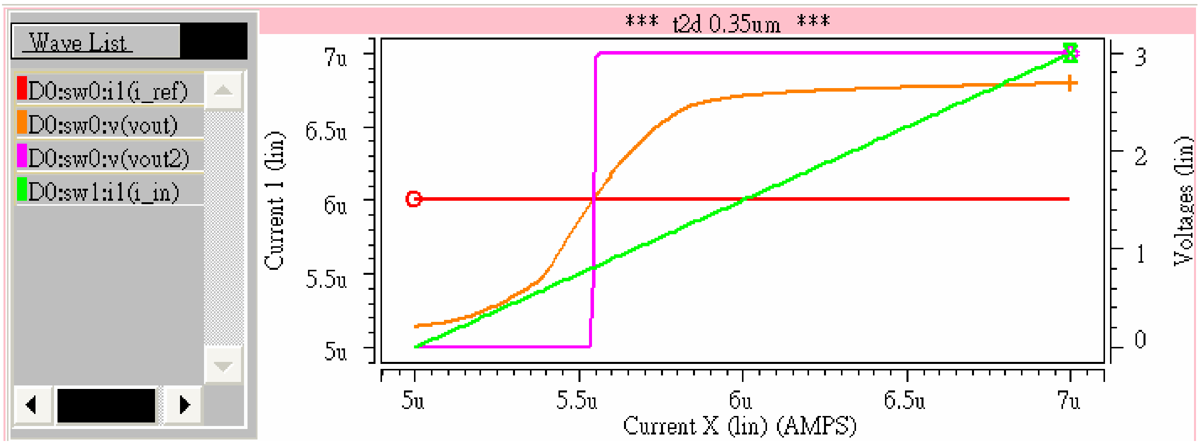


Fig 3.5 The Simulation Result of Comparator ***COMP***

The OP-Amp is employed here to function as the unit gain buffer. Fig 3.6 shows the frequency response of the OP-Amp

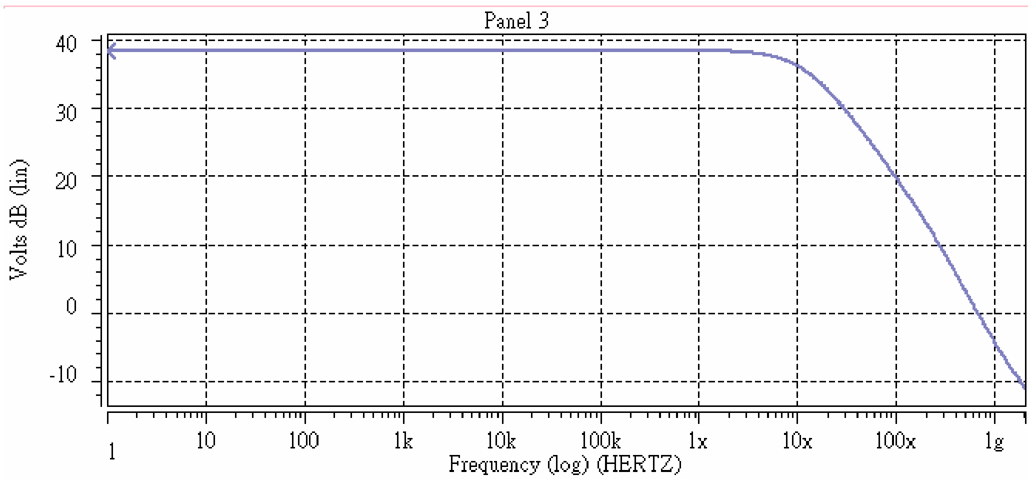


Fig 3.6 The Frequency Response of the OP-Amp that performed as unit gain buffer

Fig 3.7 below shows the absolute weight generation of three correlated patterns in the learning phase. The absolute weight generation can be conducted by the following equations

$$Isat = 6uA,\ Cw = 1.9pF,\ t = 100ns \qquad \text{Eq. (3.5)}$$

$$Vcw(p+1) = Vcw(p) + 2\times\frac{\frac{1}{4}Isat\times t}{Cw} \qquad \text{Eq. (3.6)}$$

$$Vcw(p+1) = 1.5V + 2\times\frac{1.6uA\times 100ns}{1.9pF} = 1.67V \qquad \text{Eq. (3.7)}$$

where the term *Isat* is the output current from the weighting circuit, the term *Cw* is the capacitor that stored the absolute weight. The learning period ***t*** is adjustable; therefore, we could adjust this interval to obtain the appropriate voltage level on the capacitor *Cw*. In this experiment, the learning period ***t*** is set to *100ns*. Note that in Eq.(3.6), the term $\frac{1}{4}Isat$ is set as the initial amount of current flowing out of the **W** block during the learning phase. The Eq.(3.6) describes the voltage level on the capacitor *Cw* after *(p+1)*th pattern is learned, which is the original voltage on the *Cw*, *Vcw(p)***,** and adds up the two neighboring current output from the **W** block. The resultant voltage of *Vcw(p+1)* is shown in Eq.(3.7). The absolute weight generation of three reverse correlated patterns in the learning phase is demonstrated in Fig 3.8 as well.

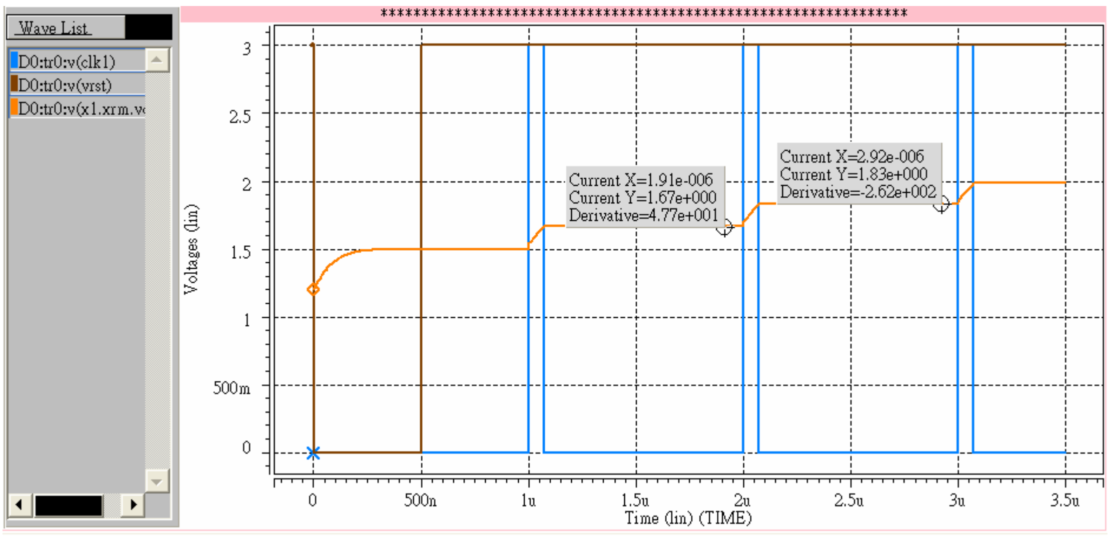


Fig 3.7 The absolute weight generation of three correlated patterns in the learning phase

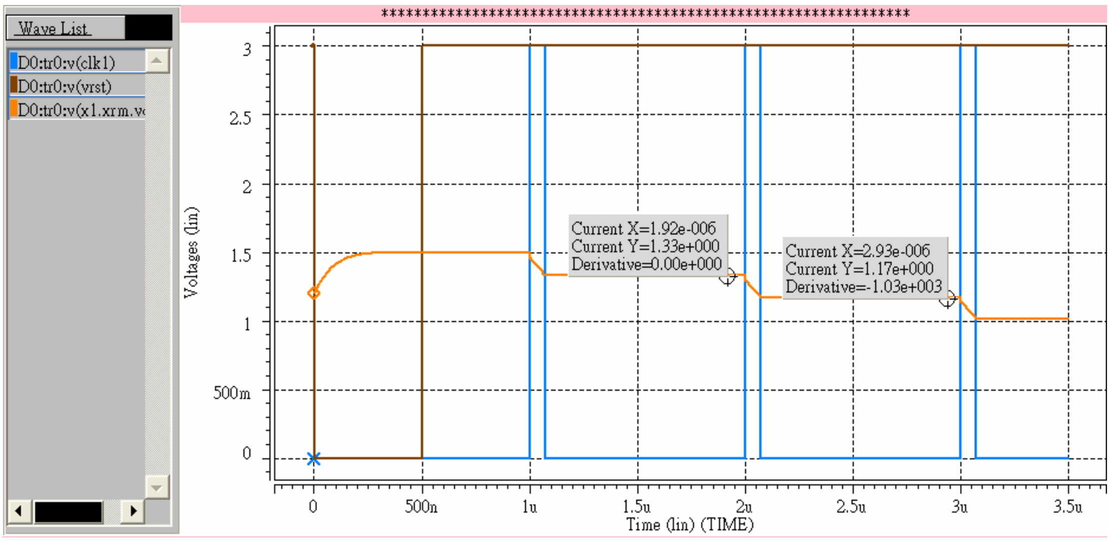


Fig 3.8 The absolute weight generation of three reverse correlated patterns in the learning phase

The post simulations of the 9x9 resolution ARMCNN system are performed using the tool UltraSim, which is dedicated for transient analysis and very suitable for this work. The pre-simulation results of the 9x9 resolution ARMCNN system are shown in Fig 3.9, Fig 3.10, Fig 3.11 with the Chinese characters “四、二、一”, respectively.

First we define the voltage level at output stage. The state voltage ranges from 1.2V to 1.8V, which has a voltage difference of 0.6V. However, due to the nonlinearity of the output stage (level-shifter), the voltage difference is only 0.54V, with 0.97V represents a black pixel and 0.43V represents a white pixel.

Considering several factors that influence the accuracy of the current ratio such as, process variation, biasing signals being disturbed by the controlling signals, layout and routing, and the designed current ratio turns out to be inaccurate and varies at most 7.3% during the post-simulation. Therefore, a tolerance must be defined to clarify whether the recognition operation fails or it's the inaccurate current ratio that leads to the imperfection. According to the factors listed above, a tolerance is defined as 10% of the output voltage

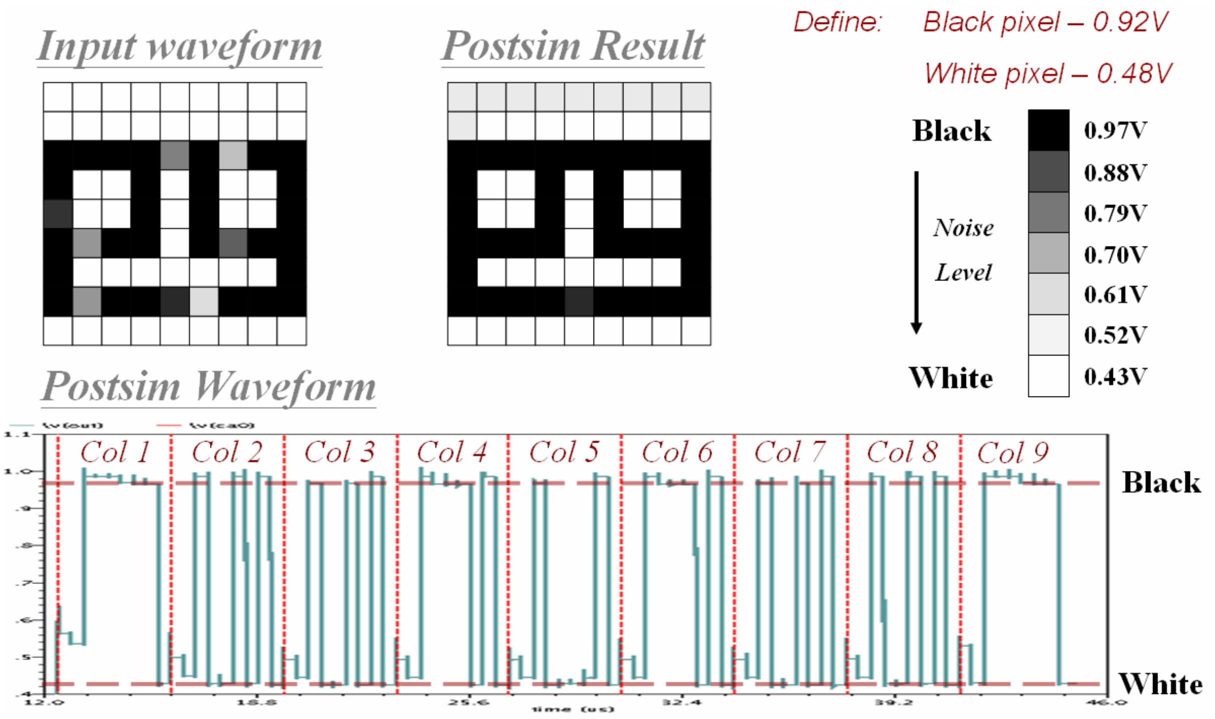


Fig 3.9 (a) Noisy pattern “四” (b) Post-sim recognition result of (a)

difference, which is 0.054V. Thus, as long as the signal is greater than 0.97V – 0.05V = 0.92V, the pixel is black. On the other hand, as long as the signal is smaller than 0.43V + 0.05V = 0.48V, the pixel is white.

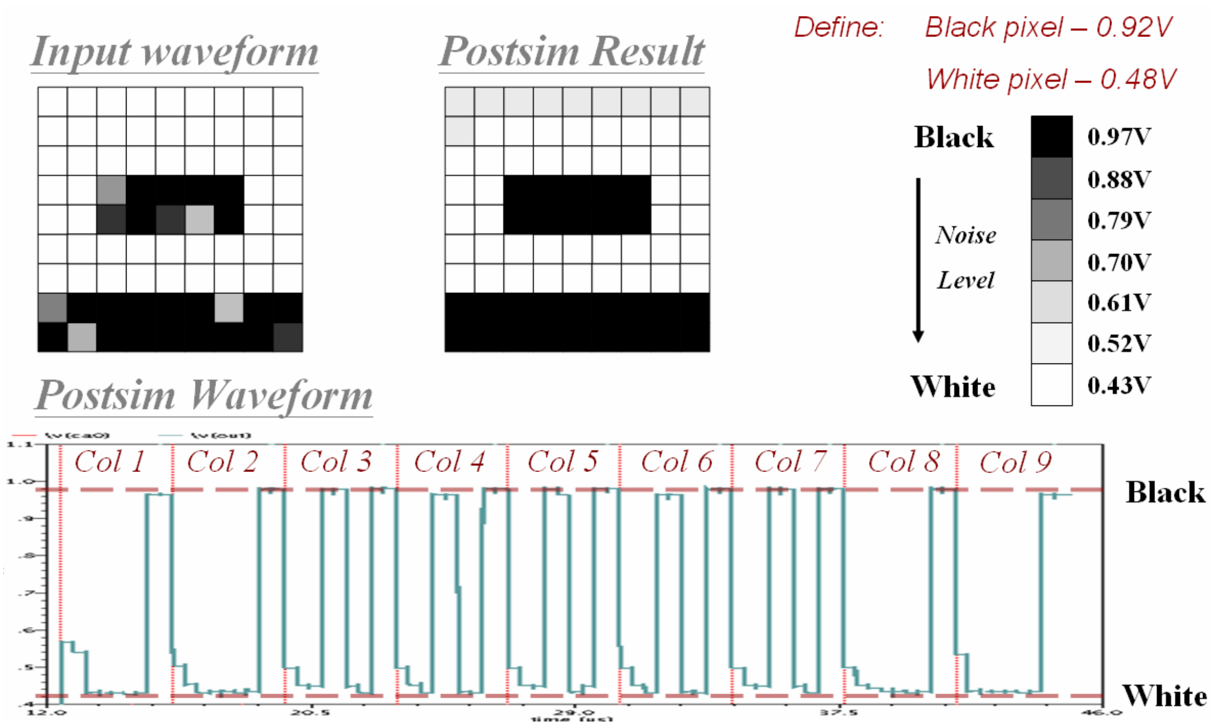


Fig 3.10 (a) Noisy pattern “二” (b) Post-sim recognition result of (a)

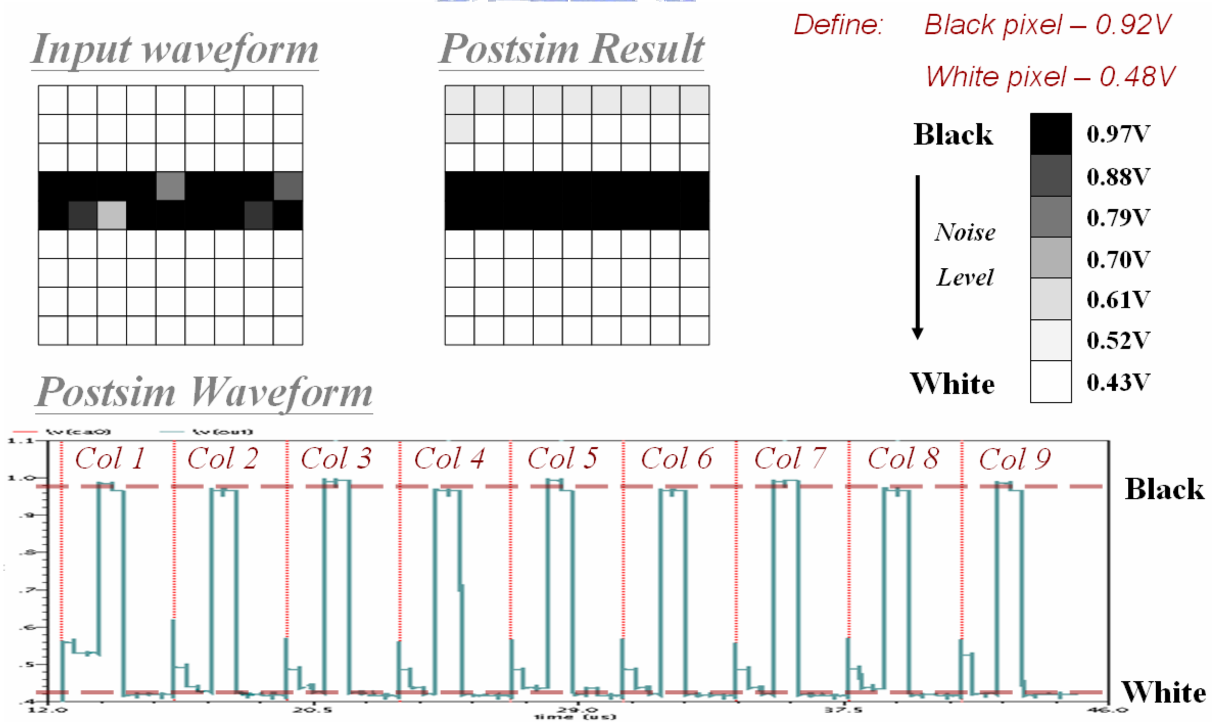


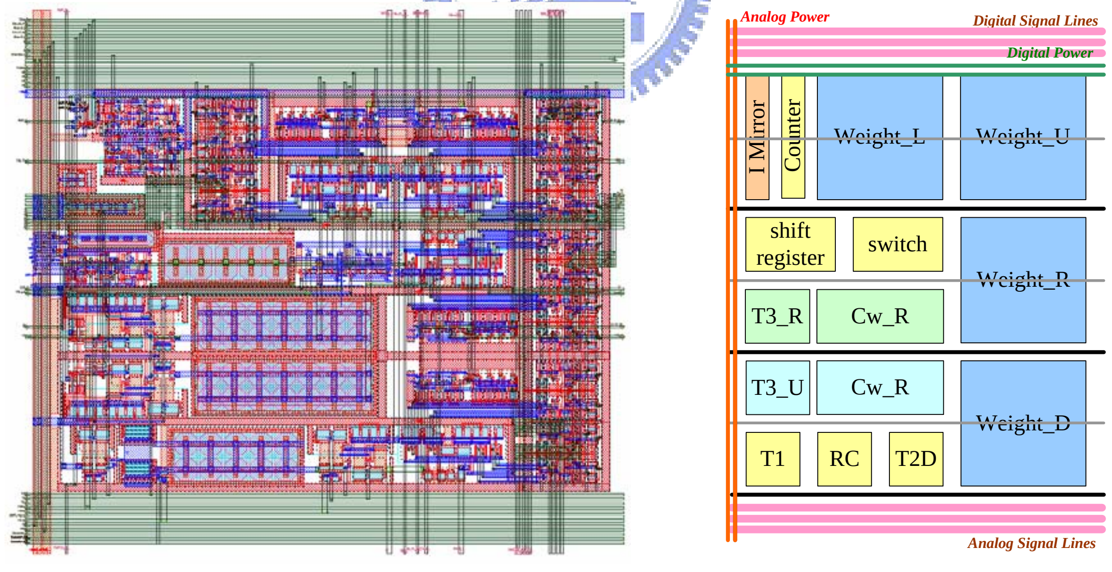
Fig 3.11 (a) Noisy pattern “一” (b) Post-sim recognition result of (a)

CHAPTER 4

EXPERIMENTAL RESULTS

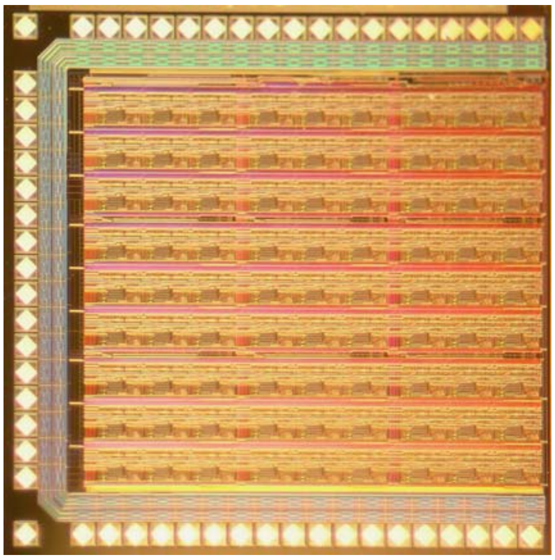
4.1 Layout Description

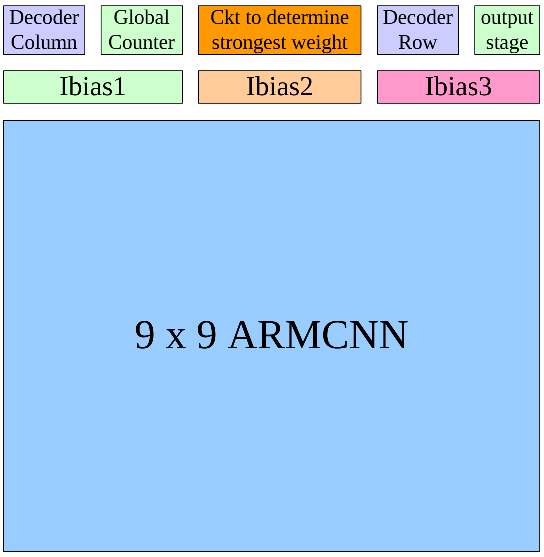
In this work, a 9x9 resolution ARMCNN is constructed. In Fig 4.1(a), the layout of one cell and two neighboring RM bocks is demonstrated, which can be better described by the symbolic illustration in Fig 4.1(b). The total area of one cell and two RM blocks is about 205um x 185um, which compares with the previous work, the tradition RMCNN, shrinks about 50% of area (400umx250um). The power consumption is as following: digital consumes about 160uA and analog consumes about 300uA at most.



(a) (b)

Fig 4.1(a) The layout of one cell and two neighboring RM blocks (b) The symbolic illustration of one cell and two neighboring RM blocks





(a) (b)

Fig 4.2(a) The die photo of this chip (b) The symbolic illustration of this chip

The total area of this whole chip is 2.24mm x 2.24mm, where the TSMC standard pads including the ESD device, pre-driver and post-driver are employed. The package diagram is shown in Fig 4.3 and the package is 68 pins LCC68. The total area included the ESD pads is only 0.28 times of the traditional RMCNN (4.56mm x 3.90mm).

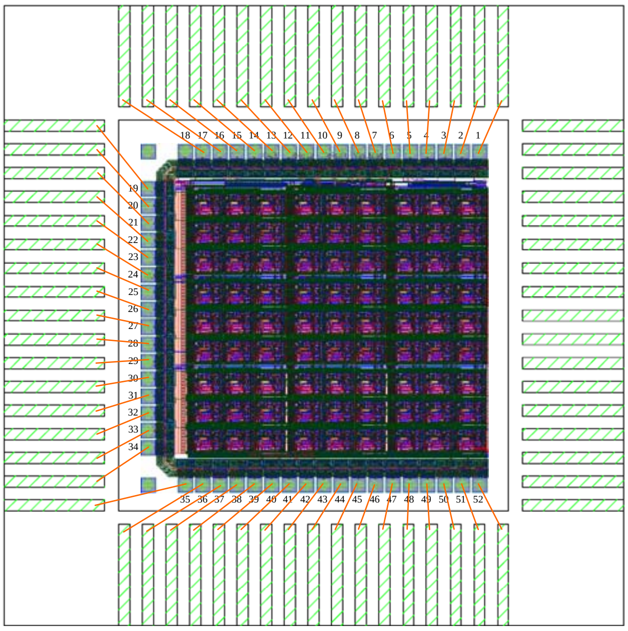


Fig 4.3 The package diagram of the ARMCNN

4.2 Experimental Environment Setup

The experimental environment is set-up as shown in Fig 4.4. The controlling signals and input pattern signals are generated by the pattern generator of HP/Agilent 16702A, in which the clock rate is set to *20MHz* and the rising / falling time of a signal is about *4.5ns*. The output waveform of each cell is read-out in series by the oscilloscope TDK 3054B. The supply voltage is 3V.

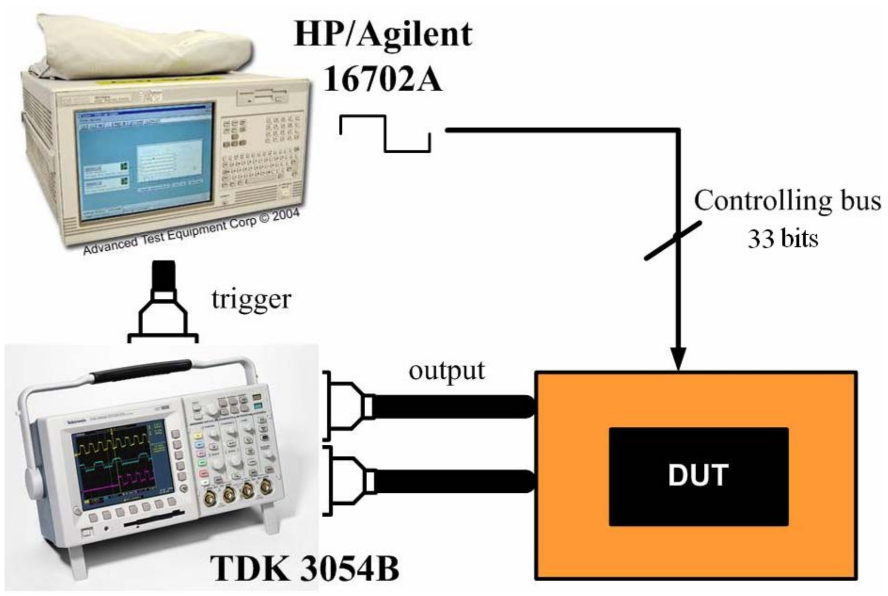


Fig 4.4 The setup of experimental environment

The timing diagram of these control signals is demonstrated in Fig 4.5, where *Clk1* determines the duration to charge / discharge the capacitor *CW* that stored the absolute weight. While *Clk2* is high, the system is operating in the pattern recognition mode. On the other hand, while *Clk3* is high, the system is operating in the pattern learning mode. *Clk4* is functioned as a reset signal, which reset the weight stored on the capacitor *CW* to *1.5V*. *DFF* triggers the static flip-flop to shift in the input pattern column by column. The signals *newp* and pin are functioned as below. When the *newp* is low, the connection between shift registers is cut off.

Thus, the data in shift registers won't be altered by the glitch from the signal *DFF*. When *newp* is high, the shift registers is able to transfer the learning patterns. Consequently, the signal *DFF* oscillates only when *newp* is high. The signal *pin* allows the pattern that stored in shift register to input into each cell. Since the operations of learning and recognition are described with figures in Chapter Two, here we are not going to repeat that again. After three patterns are learned, the ratio weights are generated in "Ratio Weight Generation" phase. The signal *Cou_L* and *Cou_G* oscillate four times to change the output of **Counter_L** and **Counter_G** from 00 → 01 → 10 → 11 sequentially, where **Counter_G**'s output controls the switch S_en1 ~ S_en6 and **Counter_L**'s output controls *Sw_a* ~ *Sw_f* to turn on one by one. The signals *noi* and *pin* together become high to generate the noisy pattern into cells. Then the circuit enters into recognition phase to recover the noisy pattern. Table 4.1 describes the function and age of all controlling signals.

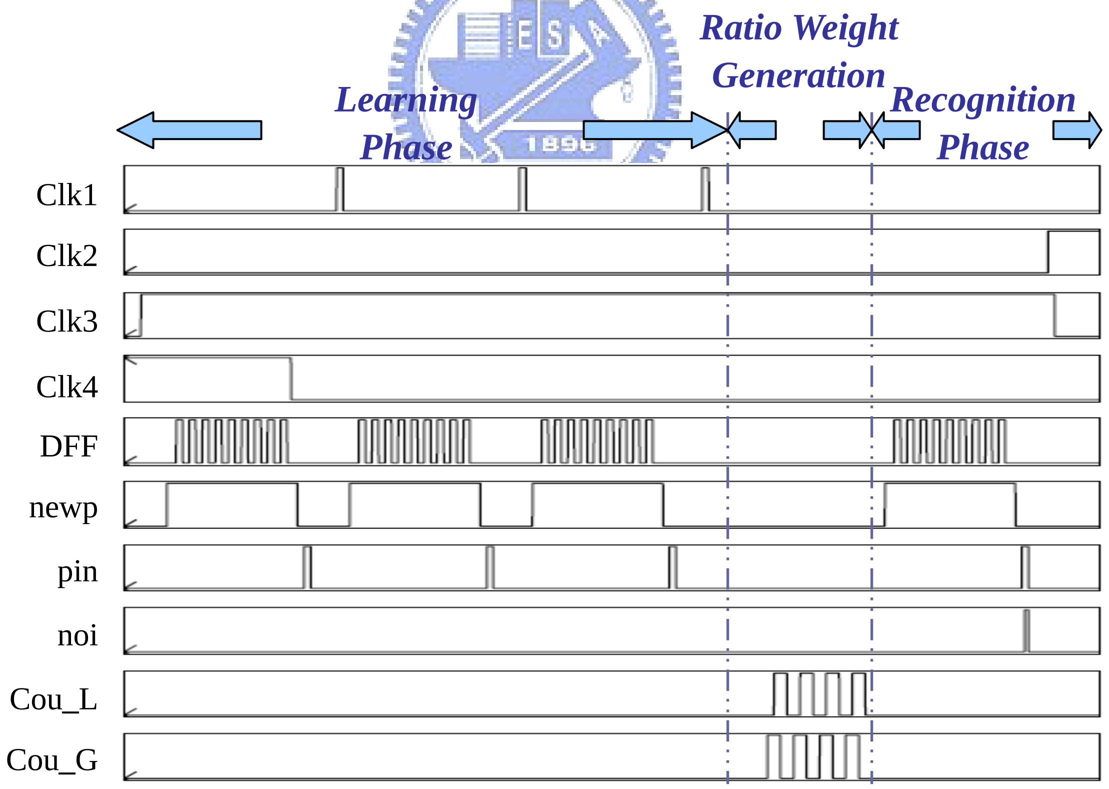


Fig 4.5 The timing diagram of the controlling signals

Control Signals	Function
Clk1	High: to charge / discharge the stored weight on CW Low: the path between cells and CW is cut off
Clk2	High: recognition mode starts Low: recognition mode stops
Clk3	High: pattern transferring starts Low: pattern transferring stops
Clk4	High: stored weights reset to 1.5V Low: do not reset
DFF	Drive the shift registers that store the learning patterns
newp	High: the shift register is able to transfer the patterns Low: the shift register is unable to transfer the patterns
pin	High: the pattern stored in shift registers inputs into cells Low: the path between shift registers and cells is cut off
noi	High: the pattern in shift registers becomes noisy Low: isolate the noise and innocent pattern in shift register
Cou_L	Drive the local counter in each cell
Cou_G	Drive the global counter

Table 4.1 The function of each control signals

4.3 Experimental Results

The output stage is described in Chapter Two. Note that there is only one pad used to read out the state voltage of all 81 pixels. Therefore, a column decoder and a row decoder are employed here to read out all 81 pixels one by one in series.

The summary comparison of this work - ARMCNN w/o EO and the previous work – RMCNN w/o EO is shown in Table 4.2. Note that both of them do not have elapsed operation.

	Previous Work	***This Work***
Technology	0.35um 2P4M Mixed-Signal Process	0.35um 2P4M Mixed-Signal Process
Resolution	9 x 9 cells	9 x 9 cells
No. of RM blocks	144 RMs	144 RMs
1 pixels	1 cell + 2 RMs	1 cell + 2 RMs
Single pixel area	400um x 250um	205um x 185um
RMCNN 9x9 array (include pad)	4560 um x 3900um	2240 x 2240
Number of Pins	81 pins	51 pins
Power supply	3V	3V
Learning power dissipation	N/A (large)	102mW
Recognition power dissipation	87mW	72mW
Quiescent power dissipation	N/A (large)	39mW
Readout time	100ns	100ns
Power Lines	N/A	Analog: 2 pairs Digital: 2 pairs Ext. I/O: 1 pairs

Table 4.2 the summary comparison of this work and the previous work

The output voltage levels of black and white are defined in section 3.2, where black pixel is 0.97V – 0.05V = 0.92V and white pixel is 0.43V + 0.05V = 0.48V.

The output of each pixel is read out column by column sequentially. The first column is read from top to bottom and then the second column and so on. Fig 4.6 shows the noisy input pattern “四” that is feed in for recognition and recovery and its presim and measurement result. Fig 4.7 and Fig 4.8 are the input patterns, the pre-simulation results, and the measurement results of the case “二” and “一”, respectively.

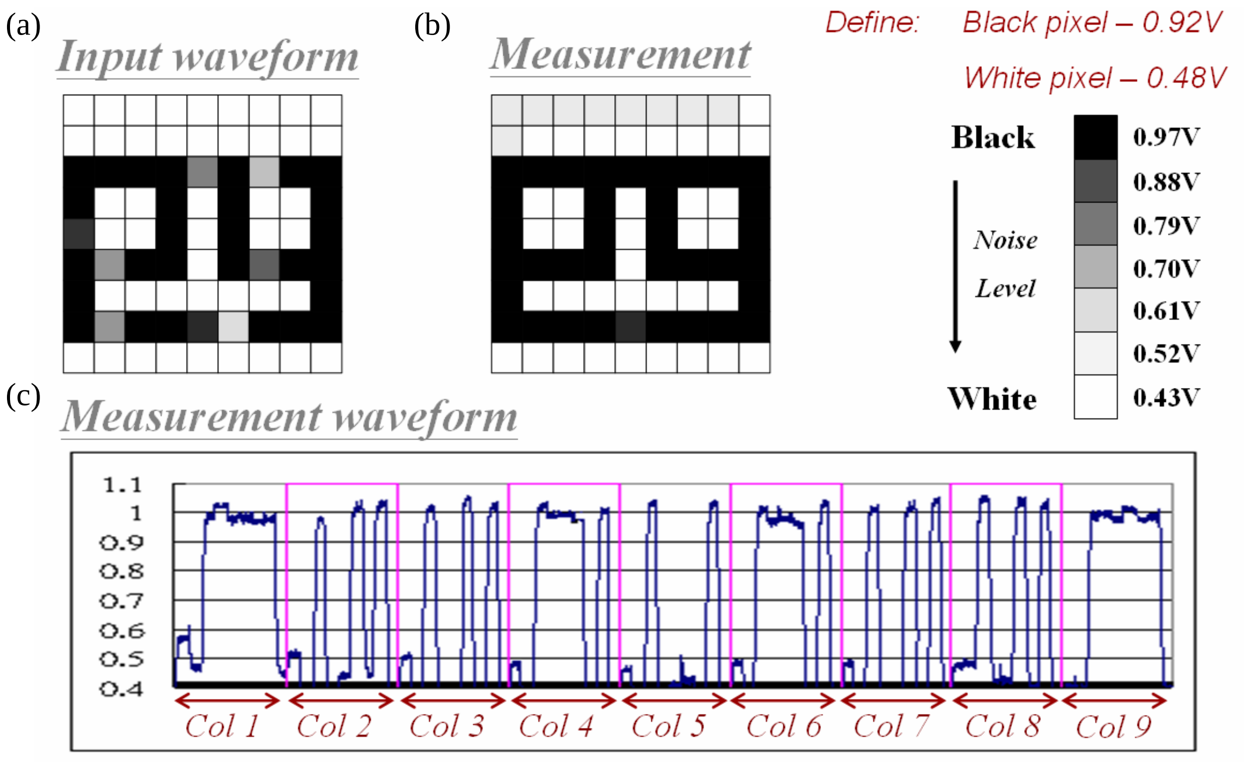


Fig 4.6 (a) noisy pattern "四" is inputted for recovery (b) (c) experimental result of recognition and recovery

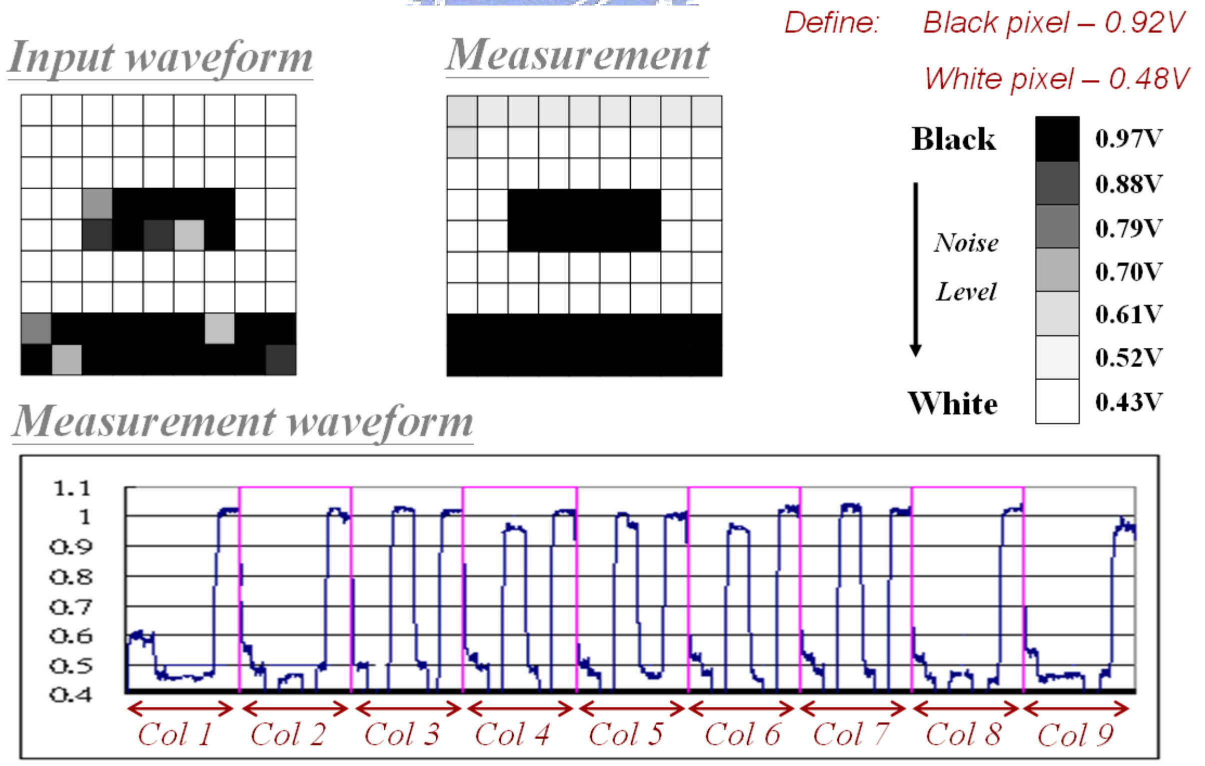


Fig 4.7 (a) noisy pattern "二" is inputted for recovery (b) (c) experimental result of recognition and recovery

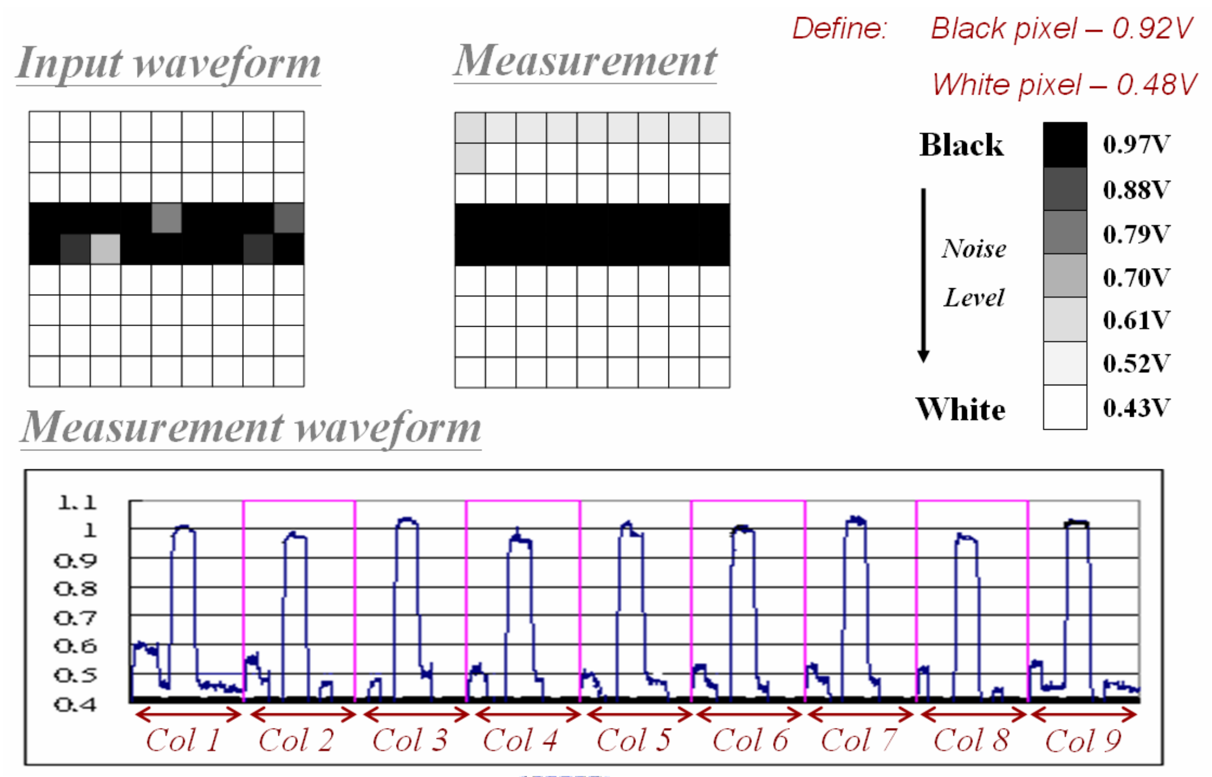


Fig 4.8 (a) noisy pattern "—" is inputted for recovery (b) (c) experimental result of recognition and recovery

4.4 Cause of the Imperfect Experiment Result

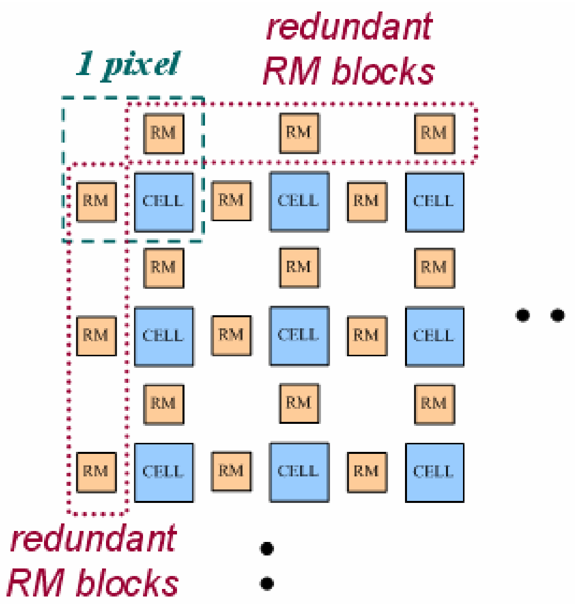


Fig 4.9 The diagram of the top left 3 x 3 cell arrays

The measurement results of all three characters show imperfection on the first row and the second row first column, which also appears in the post-simulation. Further investigation found that because in this work, a pixel consists of one cell and two RM blocks (UP and LEFT). Therefore, after 9 x 9 cell arrays are put together, there will have redundant RM blocks on the leftmost and the topmost cells, which might generated the undesired weights. Fig 4.9 shows the top left 3 x 3 cell arrays, where the number inside the circle means the cell output’s contribution factor to its neighbors and the lines between cells means two cells are having relation. If the leftmost and the topmost cells’ color never change, then unwanted weights will be generated on the redundant RM blocks. Fig 4.10 (a) is the desired ratio weights of 3 x 3 cell arrays in Fig 4.9. Fig 4.10 (b) shows how the redundant RM blocks contributes the undesired weights and lead to wrong ratio weights.

To solve this problem, redundant RM blocks must be removed to avoid undesired weights contributed from redundant RM blocks.

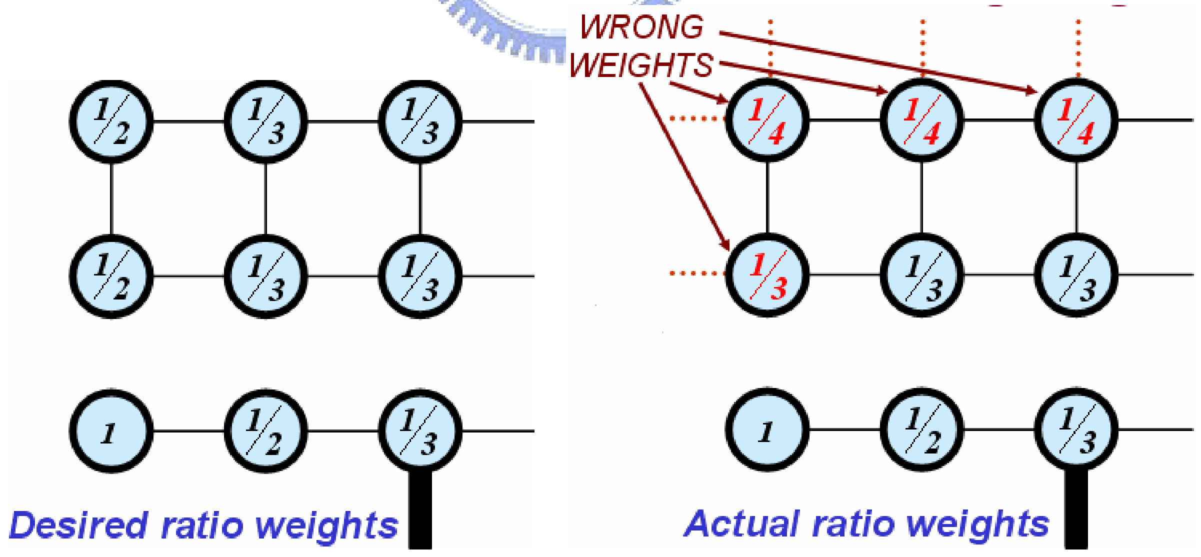


Fig 4.10 (a) desired ratio weights of Fig 4.9 (b) actual ratio weights due to redundant RM blocks

Fig 4.11 is the presim recognition result of pattern '四' for both the original design and the modified design. It is clear that in the modified version of design, the imperfection on every column's first element is fixed. Fig 4.12 and Fig 4.13 are the presim recognition result of pattern '二' and '一'.

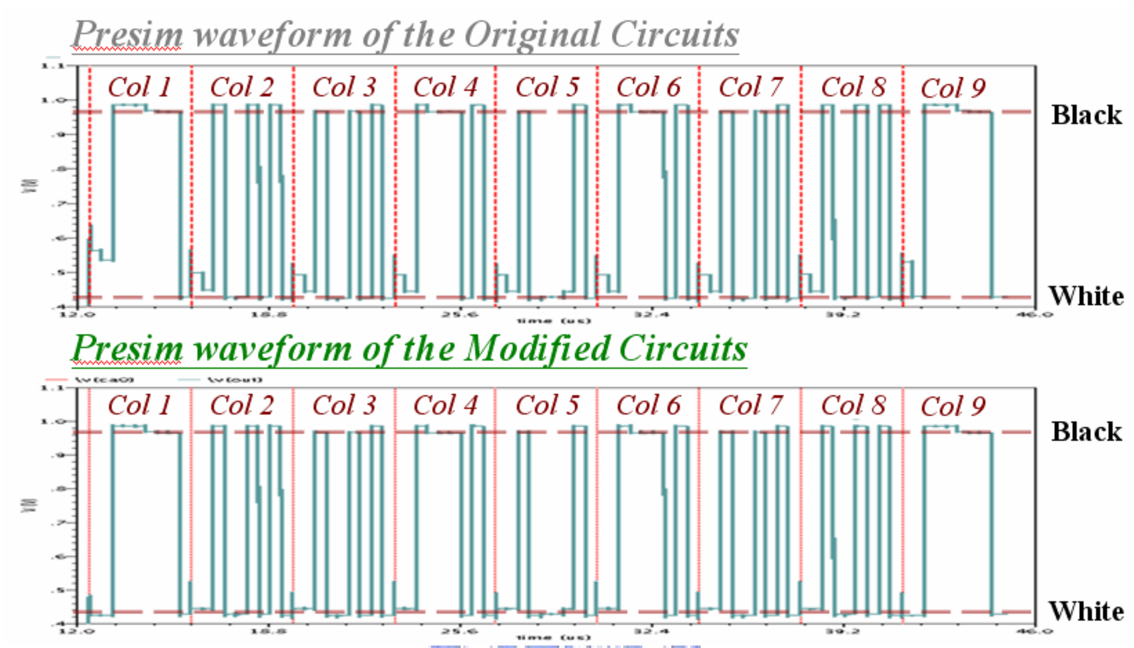


Fig 4.11 Presim results of pattern '四' for the original design and the modified design.

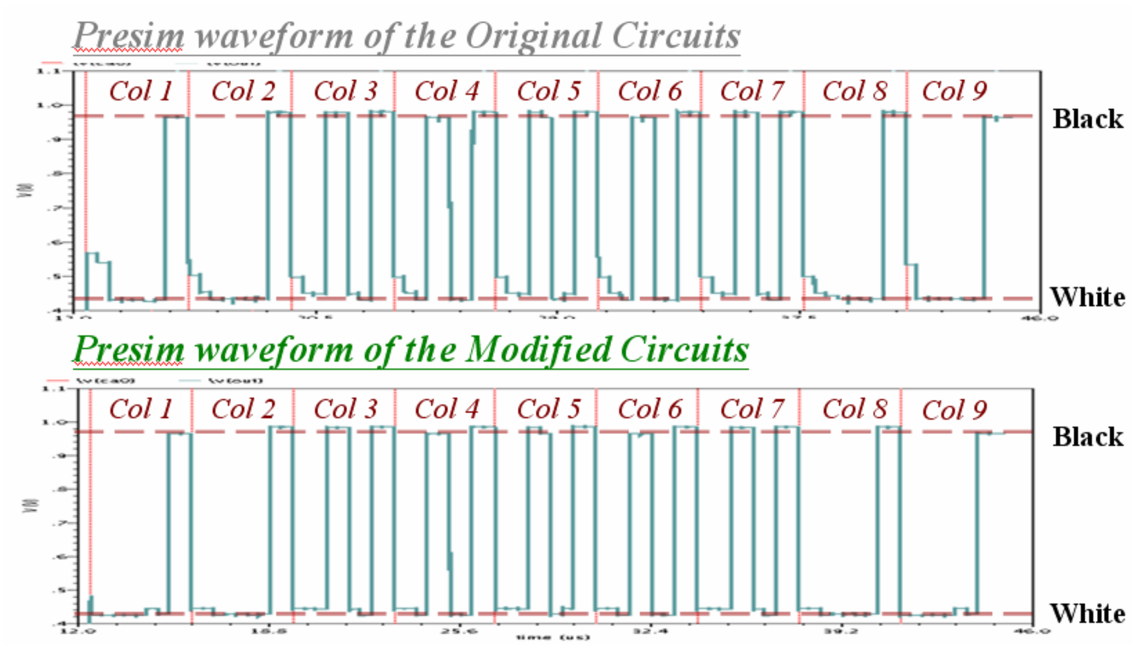


Fig 4.12 Presim results of pattern '二' for the original design and the modified design.

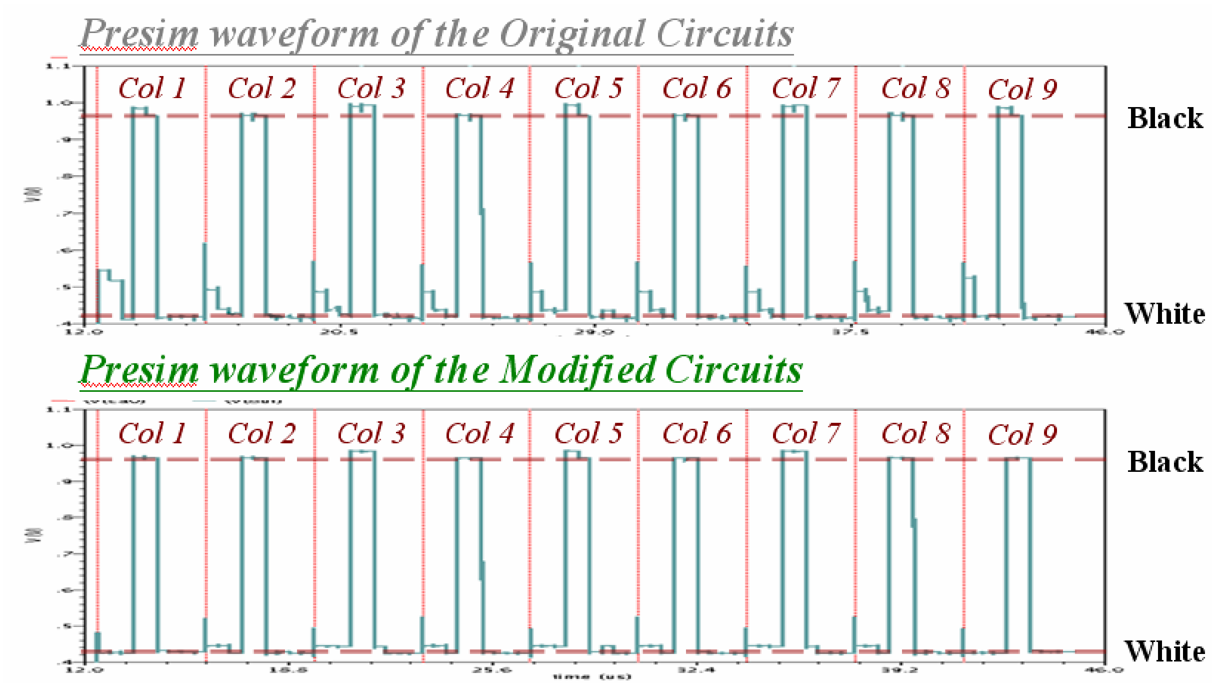


Fig 4.13 Presim results of pattern '一' for the original design and the modified design.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

A novel Autonomous RMCNN without elapsed operation is implemented. The modified Hebbian learning rule with strongest weight comparison is proposed. The new design not only inherits the advantage of the RMCNN, such as longer memory time, and image feature enhancement, but also the die area shrinks to only 0.28 times of the original design, which makes it feasible to be implemented on chip. Furthermore, all three patterns “一、二、四” can be successfully recognized and recovery.

During recognition phase, the noisy input now precharges into state capacitor rather than constantly injects to cells, which is proven to improve the recognition rate as the environmental noise raises. In addition, the proposed Hebbian learning rule ensures only the strongest weights remains instead of comparing with a mean value of abs weight in a local matrix.

The ARMCNN does not require additional elapsed phase to achieve the feature enhancement of the ratio weight. The ARMCNN uses logic operation to generate the feature enhance ratio weights directly after patterns are learned, which simplifies the complexity of the design and reduce the operation time. More importantly, it yields the same recognition rate comparing with the design with elapsed operation.

The total number of the learning patterns of ARMCNN is three. They are Chinese characters one, two and three（一、二、四）. According to the simulation results by C language, ARMCNN yields a better recognition rate when the environmental noise increases and when the total number of the learning patterns increases to four.

In addition, the design has fixed the problem that a small current charges / discharges the stored weights during pattern transferring. The dynamic flip-flop (DFF) in the previous work, RMCNN w/o EO has caused dc power consumption problem. Thus, in this work, the DFF has been replaced by the static flip-flop. Besides, the detector circuit is also modified to cutoff a huge amount of quiescent power (from 32mW to 4mW).

5.2 Future Work

The ARMCNN w/o EO in this thesis can recognize all of the three patterns. However, the redundant RM blocks cause some imperfection on the boundary cells. The modified circuit should be taped out again. Moreover, there are six switches used in the local counters and they are controlled by the combination logic from the global counter. However, the total number of switches used in the local counters can reduce to four. By doing so, we can make the layout routing easier. The controlling signals and total number of pins used are still too many (51 pins in this work), which makes the measurement and design more complicate. It is possible to reduce 9 input pins to 1 input pins and on-chip generated 3 reference voltage of (1/2 VDD). Some controlling signals can be combined together to reduce the control signals used. It can reduce 13 pins. The idea of replacing circuits for learning behavior and ratio weights generation with digital circuits might greatly reduce the die area. Because now we use many capacitors to store the weights and many analog circuits to perform learning behavior, it not only takes space, but also increase the possibility of getting errors. The total elements saved are: 144 capacitors (>1pF), 144 T3 and 144 COMP circuits, 81 T1 and T2D circuits. Accordingly, it can reduce about a half of the chip area.

REFERENCES

[1] L. O. Chua and L. Yang, "Cellular neural networks: theory," *IEEE Tran. Circuits Syst.*, vol. 35, pp.1257-1272, Oct. 1988.

[2] L. O. Chua and L. Yang, "Cellular neural networks: applications," *IEEE Tran. Circuits Syst.*, vol. 35, no. 10, pp.1273-1290, Oct. 1988.

[3] D. Liu and A. N. Michel, "Cellular neural netowrks for associative memories," *IEEE Trans. Circuits Syst. II*, vol. 40, no. 2, pp. 119-121, February 1993.

[4] A. Lukianiuk, "Capacity of cellular neural networks as associative memories," in *proc. IEEE Int. Workshop on Cellular Neural Networks and their Applications, CNNA*, June 1996, pp. 37 -40.

[5] M. Brucoli, L. Carnimeo, and G. Grassi, "An approach to the design of space-varying cellular neural networks for associative memories," in *Proc. the 37th Midwest Symposium on Circuits and Syst.*, 1994, vol. 1, pp. 549-552.

[6] H. Kawabata, M. Nanba, and Z. Zhang, "On the associative memories in cellular neural networks," in *Proc. IEEE Int. Conference on Systems, Man, and Cybernetics, Computational Cybernetics and Simulation*, 1997, vol. 1, pp. 929 - 933.

[7] P. Szolgay, I. Szatmari, and K. Laszlo, "A fast fixed point learning method to implement associative memory on CNNs," *IEEE Trans. Circuits and Syst. I*, vol. 44, no. 4, pp. 362-366, Apr. 1997.

[8] R. Perfetti and G. Costantini, "Multiplierless Digital Learning Algorithm for Cellular Neural Networks," *IEEE Trans. Circuits Syst. I*, vol. 48, no. 5, pp. 630-635, May 2001.

[9] A. Paasio, K. Halonen, and V. Porra, "CMOS implementation of associative memory using cellular neural network having adjustable template coefficients," in *Proc. IEEE Int. Symposium on Circuits and Syst., ISCAS*, 1994, vol. 6, pp. 487-490.

[10] C.-H. Cheng and C.-Y. Wu, "The design of cellular neural network with ratio memory for pattern learning and recognition," in *CNNA*, 2000, pp. 301-307.

[11] C.-Y. Wu and C.-H. Cheng, "A learnable cellular neural network structure with ratio memory for image processing," *IEEE Trans. Circuits Syst. I*, vol.49, pp. 1713-1723, Dec. 2002.

[12] C.-H. Cheng and C.-Y. Wu, "The design of ratio memory cellular neural network (RMCNN) with self-feedback template weight for pattern learning and recognition," in *CNNA*, 2002, pp. 609-615.

[13] Y. Wu and C.-Y. Wu, "The design of CMOS non-self-feedback ratio memory for cellular neural network without elapsed operation for pattern learning and recognition," in *CNNA*, 2005, pp. 282-285.

[14] J.-L. Lai and C.-Y. Wu, "A learnable self-feedback ratio-memory cellular nonlinear network (SRMCNN) with B templates for associative memory applications," in *ICECS*, 2004, pp. 183-186.

[15] C.-Y. Wu and C.-H. Cheng, "Improvement of pattern learning and recognition ability in ratio-memory cellular neural networks with non-discrete-type hebbian learning algorithm," in *ISCAS*, 2002, pp. 629-632.

[16] 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 Trans. VLSI Syst.*, vol. 12, pp. 1182-1191, Nov. 2004.

[17] C.-Y. Wu, C.-Y. Hsieh, S.-H. Chen, B. C.-Y. Hsieh, and C.-R. Chen, “Non-saturated binary image learning and recognition using the ratio memory cellular neural network (RMCNN),” in *CNNA*, 2002, pp. 624-620

[18] C.-Y. Wu and J.-F. Lan, “CMOS current-mode neural associative memory design with on-chip learning,” *IEEE Trans. Neural Networks*, vol. 1, pp. 167-181, Jan. 1996.

[19] J.-F. Lan and C.-Y. Wu, “CMOS current-mode outstar neural networks with long period analog ratio memory,” in *Proc. IEEE Int. Symp. Circuits and Systems*, vol. 3, 1995, pp. 1676-1679.

[20] ----, “Analog CMOS current-mode implementation of the feedforward neural network with on-chip learning and storage,” in *Proc. Of 1995 IEEE International Conf. on Neural Networks*, vol. 1, 1995, pp. 645-650.

[21] C.-Y. Wu and J.-F. Lan, “A new neural associative memory with learning,” in *IJCNN*, vol. 1, 1992, pp. 487-492.

[22] J.-F. Lan and C.-Y. Wu, “The multi-chip design of analog CMOS expandable modified Hamming neural network with on-chip learning and storage for pattern classification,” in *ISCAS*, vol. 1, 1997, pp. 565-568.

[23] D. O. Hebb, *The Organization of Behavior: A Neuropsychological Theory.* NY: Wiley, 1949.

[24] S. Haykin, *Neural Networks, A Comprehensive Foundation*. Macmillan College Publishing Company, Inc., 1994, pp. 290-291.

[25] L. O. Chua, “Guest Editorial,” *IEEE Trans. Circuits Syst. I*, vol. 42, pp. 557-558, Oct. 1995.

[26] J. F. Lan, C.Y. Wu, Chapter 3 of “The Designs and Implementations of the Artificial Neural Networks with Ratio Memories and Their Applications” June 1996, pp. 64-67

[27] C.-Y.Wu and S.-Y.Tsai, Autonomous ratio-memory cellular nonlinear network (ARMCNN) for pattern learning and recognition, CNNA, pp.137-141, August 2006

簡歷

姓 名：周 維 德

學 歷：

Burnaby Central Secondary School (85年 9 月 ~ 89 年 6 月)

University of British Columbia (89年 9 月 ~ 93 年 6 月)

國立交通大學電子研究所碩士班 (94年 9 月 ~ 96 年 12 月)

研究所修習課程：

類比積體電路 I、II	吳介琮教授
數位積體電路	周世傑教授
積體電路之靜電放電防護設計特論	柯明道教授
矽智產設計	黃俊達教授
計算機輔助設計特論	周景陽教授
功率積體電路設計	陳科宏教授
混合訊號式積體電路設計與實驗 I	吳介琮教授
穩健設計之品質工程	黎正中教授

永久地址： 台北市光復南路568-1號12F
Email： acer.chou@msa.hinet.net
m9311697@alab.ee.nctu.edu.tw