國立交通大學

電子工程學系　電子研究所碩士班

碩　士　論　文

免衰減操作無自迴授比例式記憶體細胞非線性網路之設計

The Design of CMOS Non-Self-Feedback Ratio Memory Cellular Nonlinear Network without Elapsed Operation for Pattern Learning and Recognition

研 究 生 ： 吳　諭

指導教授 ： 吳重雨 教授

中華民國九十四年九月

免衰減操作無自迴授比例式記憶體細胞非線性網路之設計

The Design of CMOS Non-Self-Feedback Ratio Memory Cellular Nonlinear Network without Elapsed Operation for Pattern Learning and Recognition

研 究 生：吳　諭　　Student ：Yu Wu

指導教授：吳重雨 教授　　Advisor ：Prof. Chung-Yu Wu

國立交通大學

電子工程學系　電子研究所碩士班

碩士論文

A Thesis
Submitted to Department of Electronics Engineering & Institute of Electronics
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

September 2005
Hsin-Chu, Taiwan, Republic of China

中華民國 94 年九月

免衰減操作無自迴授比例式記憶細胞非線性網路之設計

學生：吳諭　　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

在圖形辨識的領域上，聯想式記憶是一個相當熱門的辨識方法，而無自迴授比例式記憶類神經網路則已被證實可以作為一種聯想式記憶的實現方法。然而，無自迴授比例式記憶類神經網路需要一段漏電操作以產生高辨識率的比例鍵值，而這段漏電操作的時間則會因所學習的圖形不同而改變，造成圖形辨識上的困擾。

本論文的主旨在於闡述免漏電操作無自迴授比例式記憶細胞非線性網路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。免漏電操作無自迴授比例式記憶類神經網路在產生高辨識率的比例鍵值時，無須使用到漏電操作，可在圖形學習完畢後，直接產生所需的比例鍵值，並達到和原本比例式記憶類神經網路相同的辨識率。

本論文中引述了免漏電操作比例式記憶類神經網路所用以直接產生比例鍵值的理論，並實際用 TSMC 0.35um 2P4M Mixed-Signal 製程設計了一個解析度為 9x9 的免漏電操作比例式記憶類神經網路，並實現之且加以量測。電路中用到架構簡單的比較器，以節省面積。並使用計數器和比較器的組合以簡單地達到免漏電產生比例鍵值的目的。此設計中，還加上了得以任意輸入所希望學習的圖形的介面，因此，此電路可以學習任何 9x9 的圖形。另外，本論文中的設計省略了原本無自迴授比例式記憶類神經網路所需要的乘除法器，使得此設計的單位面積比原本的無自迴授比例式記憶細胞非線性網路來的小。

在量測上，雖然所學習的三個圖形，有一個辨識的不順利，但此論文也對造成此結果的原因做了探討。並重新設計電路，在 Hspice 模擬上驗證新電路確實可以改善此缺陷

The Design of CMOS Non-Self-Feedback Ratio Memory Cellular Nonlinear Network without Elapsed Operation for Pattern Learning and Recognition

Student: Yu Wu Advisor: Prof. Chung-Yu Wu

Department of Electronics Engineering & Institute of Electronics

National Chiao-Tung University

ABSTRACT

The associative memory is a hot topic in domain of pattern recognition. It is proven that the non-self-feedback ratio memory nonlinear network (RMCNN) with elapsed operation can be used as a kind of associative memory. However, the RMCNN with elapsed operation needs a elapsed period to get the feature enhanced ratio weights. The elapsed period changes as learning patterns change, and thus the elapsed operation let the process of pattern recognition inconvenient.

This thesis expounds the design and usage of RMCNN without elapsed operation (RMCNN w/o EO) in the domain of pattern recognition. The RMCNN w/o EO doesn't need the elapsed period when it generates the feature enhance ratio weights. The design in this thesis can generate the feature enhance ratio weights directly after pattern learning, and it has a good recognition rate that is the same with RMCNN with elapsed operation.

This thesis quotes the theory used to generate the feature enhance ratio weights directly. In this thesis, the circuit of RMCNN w/o EO is designed and a 9x9 RMCNN w/o EO is implemented by TSMC 0.35um 2P4M mixed-signal process. A simple comparator is used to save chip area. The counters and comparators let the ratio weights without elapsed operation be generated easily. In this design, a pattern input interface that can input any patterns into the

circuit is implemented too. Thus this chip can learn any patterns. Besides, the design in this thesis didn't use the M/D in the RMCNN with elapsed operation, and the area of one cell is smaller than the RMCNN with elapsed operation.

The experimental result isn't successful completely. One of the three learning patterns isn't recognized successfully. This thesis discovers the cause of the experiment defect, and the circuit is redesigned. The new circuit operates well in the simulation result.

致謝

首先，我要謝謝我的指導教授吳重雨教授，老師本身嚴謹的研究精神，讓我在這兩年中獲得最珍貴的研究態度與方法。而且在老師的指導與充沛的研究資源下，我才得以將我的研究順利下線並以量測儀器驗證。此外，老師也提供相當優裕的研究經費使我在這兩年中不至於生活匱乏而能更努力的從事碩士論文研究，在我畢業之後的職場生活，我都會記得老師的謹慎的研究態度並以此自律要求我自己。

在我碩士兩年的研究生涯，有苦也有樂，在我煩惱或遇上抉擇時，我的父親吳宗興先生與母親陳錦綉女士總是給我最大的支持，無論我的選擇為何，而且我的父母也總是在我消沈時給我最大的鼓勵。在此致上我最高的謝意。另外，我也要謝謝我的女友陳俐君小姐，即使我正忙於課業與研究，依然在我身邊陪伴我而無一句怨言。

接著要感謝的是在我碩士兩年一直指導我的鄭秋宏學長，廖以義學長，施育全學長，王文傑學長，林俐如學姐，陳勝豪學長，虞繼堯學長，蘇烜毅學長，陳旻珓學長以及林韋霆學長。多謝他們不耐煩的與我討論並指導我。尤其感謝陳勝豪學長和林俐如學姐，他們總是在小 meeting 時，耐心的聽我報告，並給我電路上很多建議，我才得以完成整個研究。另外，很感激蔡夙勇學長幫我修訂論文中的 BUG，讓我論文健全且清楚不少，而且在碩士生涯尾端與蔡夙勇學長討論也讓我對 CNN 多懂不少。

最後要謝謝在 520 及 527 實驗室跟我一起打拼的家熒，大建樺，小鍵樺，煒明，靖驊，弼嘉，宗信，竣帆，志朋，傑忠，啟佑，岱原，進元，宗熙，建文，台祐以及待在 307 的文芩，少了他們，我的碩士生涯將黯淡不少，也將少了很多成長的機會，也會少了很多打 GAME 的同伴。

CONTENT

TABLE CAPTIONS

FIGURE CAPTIONS

CHAPTER 1

INTRODUCTION

1.1 Background of Cellular Nonlinear Network

Due to the advantageous feature of local connectivity, the cellular nonlinear network (CNN) introduced by Chua and Yang [1] is very suitable for VLSI implementation and thus enables many applications [2]-[3]. So far, some research works on the applications of CNNs as neural associative memories for pattern learning, recognition, and association have been explored [4], [5], [6]-[10]. Among them, many innovative algorithms and software simulations of CNN associated memories were reported [4], [5], [6]-[8]. As to the hardware implementation, special learning algorithm and digital hardware implementation for CNNs were proposed in [9] to solve the sensitivity problems caused by the limited precision of analog weights. Moreover, CMOS chip implementation of CNN associative memory was also reported in [10].

In realizing CNN associative memories, the learning circuitry can be integrated on-chip with CNNs. The major advantages of on-chip learning are : 1) No host computer is needed to perform the learning task off-line. This makes the interface of neural system chips simple for many practical applications; 2) The spatial-variant template weights can be on-chip learned without being loaded from outside to the CNN chips. Thus long loading time, complex cell global interconnection, 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.

The ratio memory (RM) of Grossberg outstar structure [11], [12]-[13] has been used in both feedforward and feedback neural network ICs for image processing [14]-[15]. It is found that the RM in neural network ICs has the advantages of long memory time and image feature

enhancement under constant leakage on stored weights.

In this chapter, both RM and modified Hebbian learning function [16] are implemented in the CNN structure with spatial-variant templates and constant leakage on stored template weights [17] for pattern learning, storing, and recognition. The proposed CNN with ratio memory (RM) is called the RMCNN. It has the advantages of on-chip learning as mentioned above. Since most of on-chip learning circuits can be shared with both RM and CNN core circuits, the extra chip area required for on-chip learning circuits is small. Moreover, the 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, but with spatial-variant template weights, the same constant leakage on template weights, and the same learning rule. As a demonstrative example, a 9x9 RMCNN without elapsed operation (RMCNN w/o EO) is realized in CMOS technology. Both simulation and experimental results have verified the advantageous characteristics of the RMCNN.

1.2 Algorithm of Ratio Memory Cellular Nonlinear Network

In our 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]-[3]

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

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

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 $N_r(i, j)$ 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 $N_r(i, j)$ 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.

$$N_r(i, j) = \{C(k,l) \mid 1 \le k \le M, 1 \le l \le N, |k-i| + |l-j| \le r\} \text{[18]} \qquad \text{Eq.(1.3)}$$

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 zij 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 $N_r(i, j)$, 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[18]-[19], and the template A_{ij} can be written as:

$$A_{ij}(0) = \begin{bmatrix} 0 & a_{ij(i-1)j}(0) & 0 \\ a_{iji(j-1)}(0) & 0 & a_{iji(j+1)}(0) \\ 0 & a_{ij(i+1)j}(0) & 0 \end{bmatrix} \qquad \text{Eq.(1.4)}$$

That means only four cell are correlated to the central cell. That's up, down, left and right side cells. In the original RMCNN with elapsed operation[18]. The weights in template A can be produced by the blow equation.

$$a_{ijkl}(0) = \frac{\sum_{p=1}^{m} \int_{T_p} u_{ij}^p u_{kl}^p \, dt}{sum1} \qquad \text{Eq.(1.5)}$$

$$kl \in (i-1)j,\ i(j-1),\ i(j+1),(i+1)j \qquad \text{Eq.(1.6)}$$

$$sum\ 1 = \sum_{kl}\left|\sum_{p=1}^{m}\int_{T_P} u_{ij}^{p} u_{kl}^{p}\, dt\right| \qquad \text{Eq.(1.7)}$$

Where u_{ij}^{p} is the pth pattern input of cell(i,j). Similarly, u_{kl}^{p} is the pth pattern of cell(k,l). The relationship between *ij* and *kl* is shown as Eq.(1.6) 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 a_{ijkl} 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

$$x_{i^* j^*}(t) = 0\ ,\ u_{i^* j^*}(t) = 0 \qquad \text{Eq.(1.8)}$$

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

1.3 Research Motivation and Thesis Organization

After learning period, the weight $a_{ijkl}(0)$ in Eq(1.5) are not used directly. Instead , we use the $a_{ijkl}(T)$ after elapsed period[18]-[19]. The weight $a_{ijkl}(T)$ can be written as

$$a_{ijkl}(T) = \frac{(\sum_{p=1}^{m}\int_{T_P} u_{ij}^{p} u_{kl}^{p}\ dt) - c(T)}{sum1 - \sum_{kl} c(T)} \qquad \text{Eq.(1.9)}$$

The c(T) is the amount of the absolute-weight decaying. After the elapsed process, all absolute-weights decay. Some of the absolute weights even decays to zero. But not all of the ratio weights a_{ijkl} decay, some of the ratio weights are enhanced and the others decay. After this elapsed period, the important ratio weights become larger and the trivial weights are

smaller. Table 1.1 shows some template A of absolute-weights and ratio weights.[18]-[19] Before elapsed period, the template A of ratio weights A_{44}(0s) are the learning result according to Eq.(1.5), and the ss_{44} is the numerator of Eq.(1.5). A_{44}(0s) and ss_{44}(0s) both don't have zero elements. It's obvious, after elapsed period, some of elements in ss_{44}(850s) decay to zero. Computing the corresponding ratio weights with Eq.(1.5), then we'll get the A_{44}(850). In A_{44}(850), the important ratio weight $\frac{1}{2}$ increases to 1, and the others decrease to 0. So the template A becomes a feature enhanced template. With this characteristic, the recognition rate is improved.

The original design, RMCNN with elapsed operation, needs a elapsed period to get the feature enhance ratio weights *Aij*. But the length of elapsed period must be controlled well. If the length of elapsed period is too long, all of the ratio weights decay to zero and the circuit doesn't have any recognition function. If the length of elapsed period is too short, we can't get a good feature enhanced ratio weights. Some weights that should decay to zero don't decay to zero completely.

When those learning patterns change, the best length of elapsed period changes too. Then it's necessary to tune the best length of elapsed period with software when we want to let the circuit learn different patterns. This step let the operation of this circuit not automatic enough.

We develop a new RMCNN w/o EO. This new structure generates the feature enhanced ratio weights directly after learning period. When the learning patterns change, we needn't adjust the elapsed time. The new structure can recognize noisy pattern directly after learning period. In this thesis, chapter 2 describes the architecture and the CMOS circuit implementation. Chapter 3 is about the simulation result of Hspice and Matlab. The experimental result and some layout description are in chapter 4. Finally, chapter 5 is the conclusion and future work.

Table 1.1 Template A of ratio weights and the corresponding absolute-weights

RMCNN	Ratio weights	Corresponding absolute-weights
9x9 $r = 1$	A_{44} (0 s)= $\begin{bmatrix} 0 & \frac{-1}{6} & 0 \\ \frac{1}{6} & 0 & \frac{1}{6} \\ 0 & \frac{1}{2} & 0 \end{bmatrix}$ A_{44} (850 s)= $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$	ss_{44} (0 s)= $\begin{bmatrix} 0 & -\frac{1}{3} & 0 \\ \frac{1}{3} & 0 & \frac{1}{3} \\ 0 & \frac{1}{1} & 0 \end{bmatrix}$ ss_{44} (850 s)= $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & \frac{2}{3} & 0 \end{bmatrix}$
9x9 $r = 1$	A_{51} (0 s)= $\begin{bmatrix} 0 & \frac{3}{5} & 0 \\ 0 & 0 & \frac{1}{5} \\ 0 & \frac{1}{5} & 0 \end{bmatrix}$ A_{51} (850 s)= $\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	ss_{51} (0 s)= $\begin{bmatrix} 0 & \frac{1}{1} & 0 \\ 0 & 0 & \frac{1}{3} \\ 0 & \frac{1}{3} & 0 \end{bmatrix}$ ss_{51} (850 s)= $\begin{bmatrix} 0 & \frac{2}{3} & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$
9x9 $r = 1$	A_{44} (0 s)= $\begin{bmatrix} 0 & \frac{-1}{8} & 0 \\ \frac{3}{8} & 0 & \frac{3}{8} \\ 0 & \frac{1}{8} & 0 \end{bmatrix}$ A_{62} (850 s)= $\begin{bmatrix} 0 & 0 & 0 \\ 0.5 & 0 & 0.5 \\ 0 & 0 & 0 \end{bmatrix}$	ss_{62} (0 s)= $\begin{bmatrix} 0 & -\frac{1}{3} & 0 \\ \frac{1}{1} & 0 & \frac{1}{1} \\ 0 & \frac{1}{3} & 0 \end{bmatrix}$ ss_{62} (850 s)= $\begin{bmatrix} 0 & 0 & 0 \\ \frac{2}{3} & 0 & \frac{2}{3} \\ 0 & 0 & 0 \end{bmatrix}$

CHAPTER 2

ARCHITECTURE AND CIRCUIT IMPLEMENTATION

2.1 Operational Principle and Architecture

It is known that the ratio memory (RM) can suppress the unimportant weight and enhance the significant weight to get the feature enhance characteristics.[18]-[19] Since the absolute weights are decreased with the leakage current, significant ratio weights increase whereas the unimportant ratio weights decrease. For example, two of the four weights in template A increase and the others decrease. Finally the two increasing weights increase up to 1/2. Similarly, these significant three (four) weights increase to 1/3 (1/4).

After leakage current decay the absolute weight, some ratio weights increase and some decrease. The equation used to distinguish which ratio weights increase and which ratio weights decrease can be written as[20]

$$I_{Mss}(t) = \frac{\sum_{j=1}^{n} I_{aw(j)}(t)}{n} \qquad \text{Eq (2.1)}$$

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

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

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

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

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.

A 9x9 array size RMCNN is implemented in this thesis. Fig. 2.1 shows the block diagram of the RMCNN w/o EO and the controlling relationship between every block. The 9x9 shift register is used to store learning patterns. The learning patterns is generated by pattern generator and is inputted into the shift registers in series. When a learning pattern is stored in register completely, the pattern is inputted into RMCNN w/o EO in parallel for pattern learning. After every pattern is learned, the RMCNN w/o EO enter recognition period. The recognized result is sent to the output stage that is controlled by two decoders and the output stage output the state of each cell in series. The decoder Decoder_C selects the columns, and Decoder_R selects the rows of the 9x9 array of output stage.

The general architecture of RMCNN is shown as Fig. 2.2. Fig. 2.2 shows the connections between cells and RMs. Each cell connects with four RMs (the up, left, right, down side). And

every RM supports the ratio weight between two pixels. With the power supply 3V in the circuit, 1.5V is defined as zero whereas 2.1V(0.9V) as +1(-1).

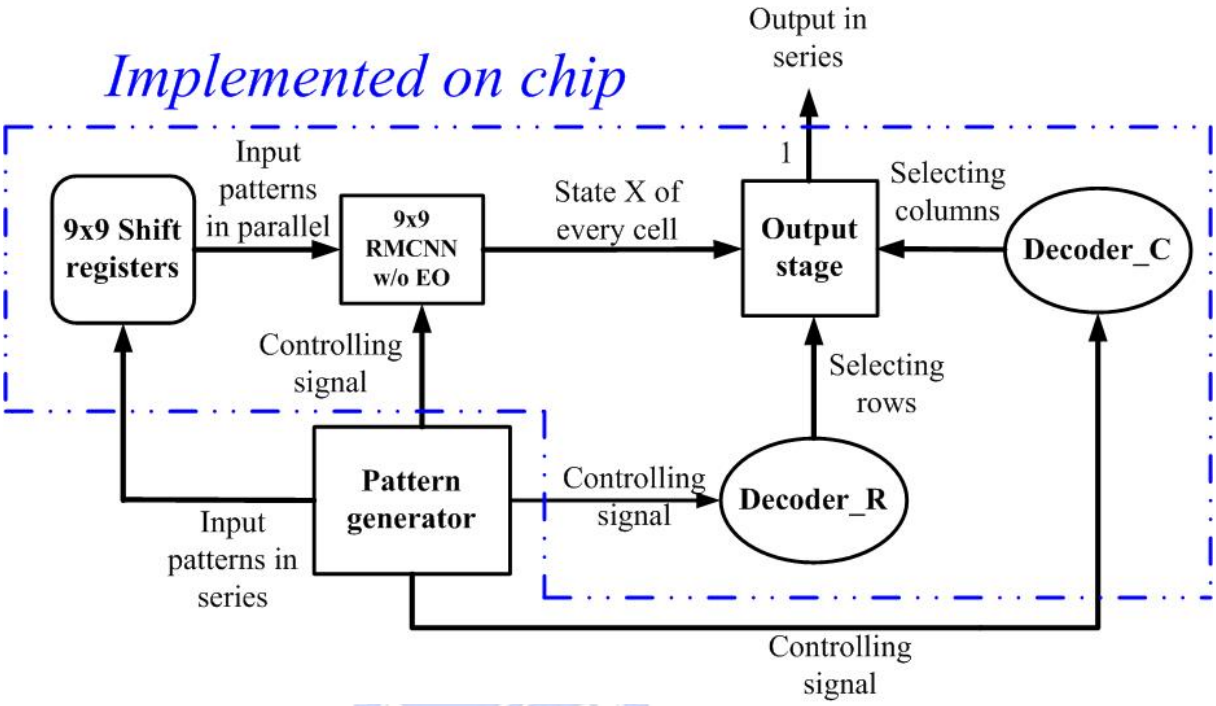

Fig. 2.1 The block diagram of RMCNN

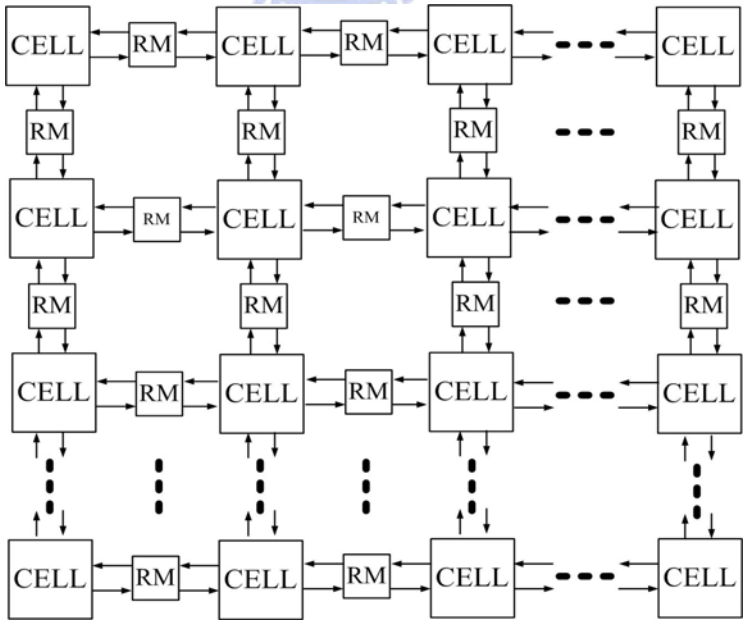

Fig. 2.2 The general architecture of RMCNN

The detailed block diagram of cell and RM is shown in Fig. 2.3. In Fig. 2.3, cell(i,j) is

the ith row and jth column cell and u_{ij}^{p} is the cell(i,j) input voltage of pth pattern. The block **T1**、**T3** in the cell (i,j) is a V-I converter to change voltage to current. . **T2D** contains a detector to detect the sign of state $X_{ij.}$ **T2D** block is also a V-I converter, and its output is absolute current. The sign of **T2D** input voltage is detected and stored separately. The block **W** uses current mirror to multiply the cell outputs by 1, 1/2, 1/3, or 1/4. One of the four weights will be chosen by **Counter_L** according to how many weight are preserved. The capacitor **Cw** stores absolute weight in learning period, and the V-I converter **T3** transfer the voltage on **Cw** to absolute current to the **COMP** block. **COMP** is a simple comparator. **COMP** block compares the mean of the four absolute memory currents with the absolute memory current, and deciding if the ratio weight should be kept. The **Counter_L** controls block **W** to weight the output of each cell. The block **T3**, capacitor **Cw**, and several switch form RM. Other blocks form CNN cell.

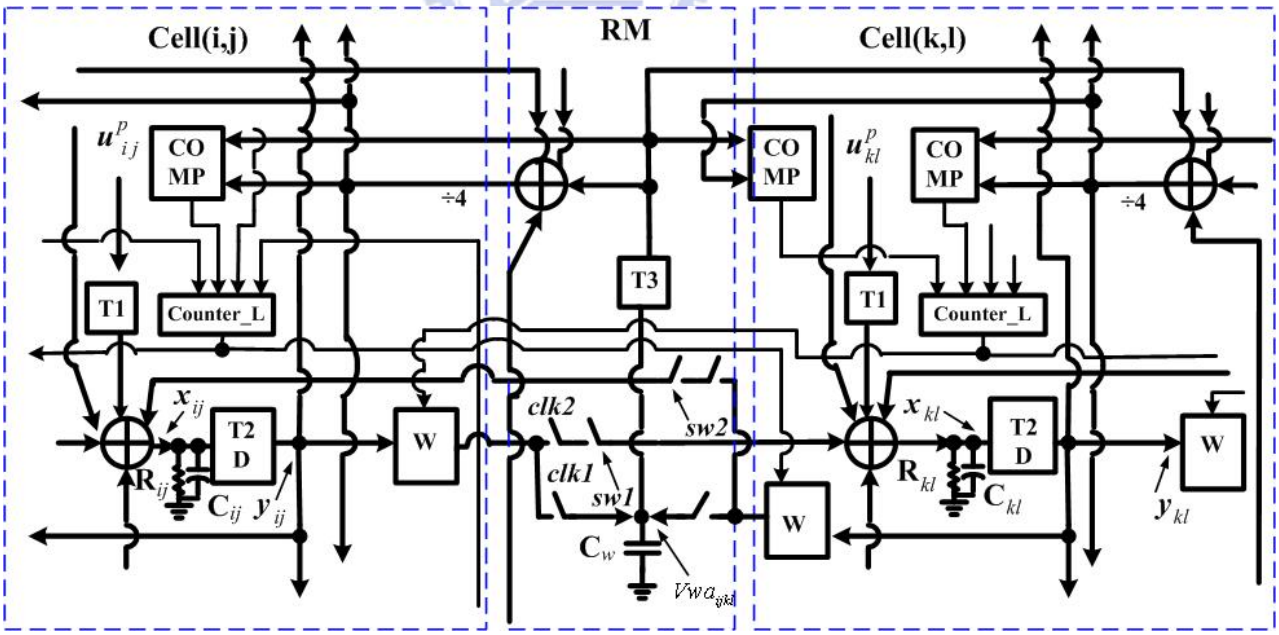

Fig. 2.3 The detail architecture of RMCNN

The Operation of this circuit is divided to two parts: learning period and recognition

period. In learning period, *clk1* is high and *clk2* is low. So the architecture in learning period is shown in Fig. 2.4. In learning period, the cell(i,j) input voltage of pth pattern u_{ij}^{p} is transferred to current Iu_{ij} by **T1** and sent to the node x_{ij}. Current Ix_{ij} can be written as

$$Iu_{ij} = \begin{cases} Iusat & when \quad u_{ij}^{p} > 2.1V \\ Gm_{T1} \times (u_{ij}^{p} - 1.5) & when \quad 1.5V < u_{ij}^{p} < 2.1V \\ 0 & when \quad u_{ij}^{p} = 1.5V \\ -Gm_{T1} \times (1.5 - u_{ij}^{p}) & when \quad 0.9V < u_{ij}^{p} < 1.5V \\ -Iusat & when \quad u_{ij}^{p} < 0.9V \end{cases} \qquad \text{Eq.(2.2)}$$

.Where Gm_{T1} is the transconductance of V-I converter **T1**. The voltage level 1.5V is defined as zero, so the current flow to opposite direction when u_{ij}^{p} is larger or smaller than 1.5V. When u_{ij}^{p} is larger than 2.1V or smaller than 0.9V, output current Iu_{ij} of **T1** becomes saturated and keeps at the current *Iusat*. *Iusat* is about 5.5uA.

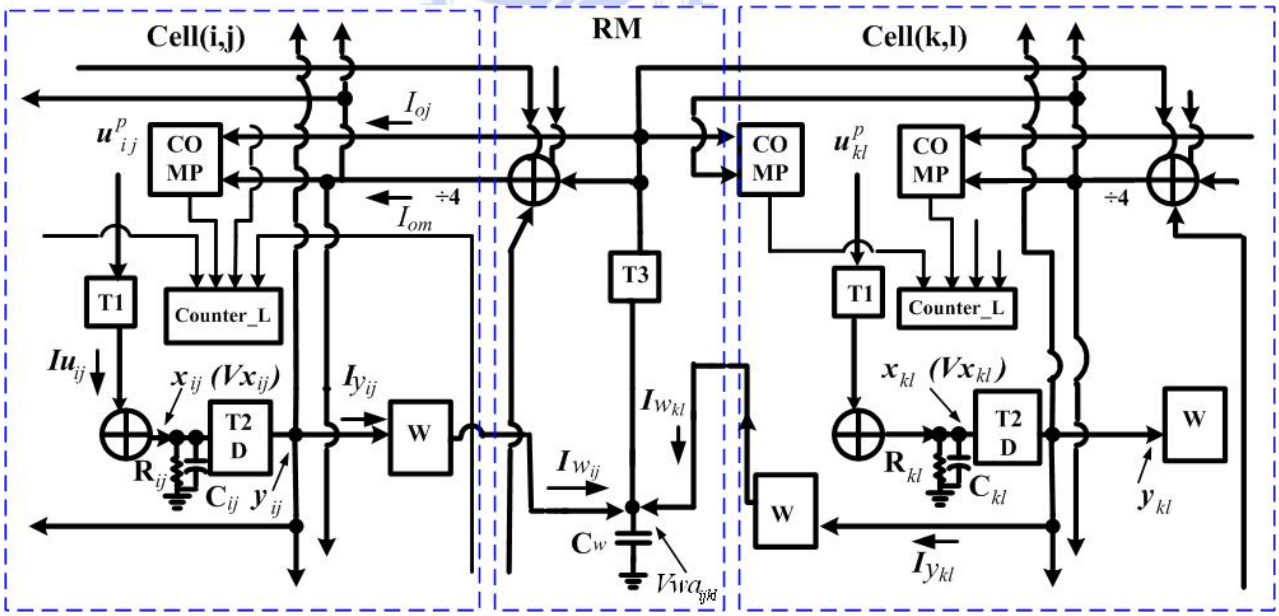

Fig. 2.4 Architecture of RMCNN in learning period

The current Iu_{ij} flows to the node x_{ij} and is converted into a voltage Vx_{ij} through the resistor R_{ij} and capacitor C_{ij}. **T2D** outputs an absolute current Iy_{ij} and a *sign*(Iy_{ij}) according to

the value of Vx_{ij}. Since the structure of **T2D** is similar to **T1** and **T2D** has a absolute-value circuit, the output current Iy_{ij} and the $sign(Iy_{ij})$ can be written as

$$Iy_{ij} = \begin{cases} Iysat & when \quad Vx_{ij} > 2.1V \\ Gm_{T2D} \times (u_{ij}^{p} - 1.5) & when \quad 1.5V < Vx_{ij} < 2.1V \\ 0 & when \quad Vx_{ij} = 1.5V \\ Gm_{T2D} \times (1.5 - u_{ij}^{p}) & when \quad 0.9V < Vx_{ij} < 1.5V \\ Iysat & when \quad Vx_{ij} < 0.9V \end{cases} \qquad \text{Eq.(2.3)}$$

$$sign(Iy_{ij}) = \begin{cases} 0V & if \quad Vx_{ij} < 1.5V \\ 3V & f \quad Vx_{ij} > 1.5V \end{cases} \qquad \text{Eq.(2.4)}$$

.Where Gm_{T2D} is the transconductance of **T2D** and the current *Iysat* is the saturated output current of **T2D**. It is about 5.5uA too. Note that Iy_{ij} always flows to the same direction whether Vx_{ij} is larger or smaller than 1.5V. The sign of Vx_{ij} is detected by a detector in **T2D** and sent to the block **W**. Current Iy_{ij} flows into the block **W**. According to the signs of input voltage Vx_{ij} and Vx_{kl}, the output current of **W** charges or discharges the capacitor Cw. The block **W** is set to a default state in learning period. The default state is multiplying Iy_{ij} by 1/4. The choice of this default state is just for circuit design convenience and we can control the length of learning time to charge or discharge the capacitor Cw. The capacitor Cw is a MOS capacitor and the capacitance value is 2p F. The capacitance value of Cw and *Iysat* is as large as RMCNN with elapsed operation [18]. To consider the leakage current effect, a constant leakage current of 0.8 fA is applied to the capacitor Cw of 2 pF so the voltage Vwa_{ijkl} is decreased. The 2 pF capacitor Cw is implemented on the chip. The value of 2 pF is chosen as a compromise between weight storage time and capacitor chip area. The capacitance value of Cw can't be chosen too small because of the leakage current consideration. Thus 2 pF is chosen. The current *Iysat* is chosen as the smallest current that can let the V-I converter operates regularly. The current *Iysat* must be small because the voltage Vwa_{ijkl} stored on Cw must be charged or discharged slowly and then the value of Vwa_{ijkl} can be controlled slightly. Thus the *Iysat* is chosen as 5.5uA and the learning time of a pattern is 100ns.

This charging or discharging Cw process is the learning behavior and generates the absolute weight at the capacitor Cw. When the inputs of neighboring cell(i,j) and cell(k,l) are white or black in a learning pattern. The capacitor Cw between these two cells is charged. Otherwise, when the inputs of these two cells are opposite color, the capacitor Cw is discharged. The voltage Vwa_{ijkl} stored on Cw can be written as

$$Vwa_{ijkl}(p+1)=\begin{cases} Vwa_{ijkl}(p)+\frac{1}{2}\frac{Iysat\times t}{Cw} & when\ sign\ of\ Vx_{ij}\ and\ Vx_{kl}\ are\ the\ same \\ Vwa_{ijkl}(p)-\frac{1}{2}\frac{Iysat\times t}{Cw} & when\ sign\ of\ Vx_{ij}\ and\ Vx_{kl}\ aren't\ the\ same \end{cases} \quad \text{Eq.(2.5)}$$

$Vwa_{ijkl}(p)$ means the voltage level after the pth pattern is learned. The output current of block **W** is $\frac{1}{4}Iysat$, and there are two **W** blocks charge or discharge a Cw at the same time. Thus after each pattern learning, the voltage changing is $\frac{1}{2}\frac{Iysat\times t}{Cw}$ ($2\times\frac{1}{4}Iysat$). The learning time of each pattern is 100ns.

After every pattern is input to circuit, capacitor Cw stores the absolute voltage weight Vwa_{ijkl}. Then **T3** converts the voltage Vwa_{ijkl} to current and sends this current to the current mode comparator **COMP**. The **COMP** compares two current: I_{oj} and I_{om} . I_{oj} is the current transferred from **T3**; I_{om} is the mean of all absolute weight current in one template A. If I_{oj} is larger than I_{om} , **COMP** gives the **Counter_L** a “logic high” that means the ratio weight between the two pixels should be preserved.

The connection between **COMP** and **Counter_L** is shown as Fig. 2.5. Since each cell just connects with the four nearest cells, there are four **COMP**s in one cell. Every **COMP** gives a logic output to **Counter_L**. At the end of learning period, **Counter_L** counts how many “logic high” are given from the four **COMP**s. If there is (are) only one (two) “logic high”, only one (two) ratio should be preserved. Then **Counter_L** controls the **W** to weight the output current of **T2D** as $1\times Iy_{ij}$ ($\frac{1}{2}\times Iy_{ij}$). Similarly, according to the output situation of

COMPs in one cell, the **Counter_L** may control the block **W** to weight output current of **T2D** as $\frac{1}{3}\times Iy_{ij}$ or $\frac{1}{4}\times Iy_{ij}$. The logic output of **COMP** in cell(i,j) (cell(k,l)) also controls the switch *sw2*(*sw1*) in Fig. 2.2. For example if the logic output of **COMP** in cell(i,j) is low (that means the ratio weight should be zero.), the switch *sw2* turns off. Then the information from cell(k,l) in recognition period is isolated. That behavior is equivalent to setting a ratio weight in a template A as zero.

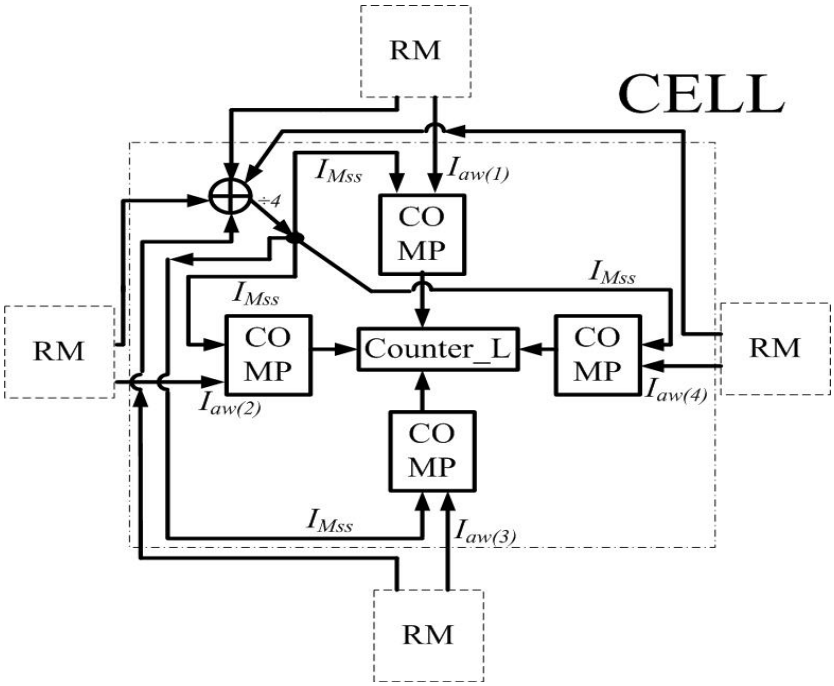

Fig. 2.5 The connection relationships of COMP, Counter_L and RM

At the ending of learning period, every **Counter_L** counts how many "logic high" are sent from **COMP**s and controls the **W** appropriately.

After learning period, the operating process enters recognition period. In this period, the input pattern is noisy pattern. The architecture in recognition period is shown as Fig.2.5. In Fig. 2.6, *clk1* is low and *clk2* is high. The states of switches *sw1* and *sw2* are controlled by **COMP**. In this period, u_{ij}^{noi} and u_{kl}^{noi} are the input voltage of noisy pattern. u_{ij}^{noi} are inputted to **T1** and transferred to current Iu_{ij}^{noi}. Iu_{ij}^{noi} and the output current Iw_{kl} from other

neighboring cell C(k,l) ($k,l \in i,j-1 \quad or \quad i,j+1 \quad or \quad i-1,j \quad or \quad i+1,j$) flow to the node x_{ij} and form the voltage Vx_{ij}. According to KCL, the $\dot{V}x_{ij}$ can be written as

$$C_{ij}\dot{Vx}_{ij}(t) = -\frac{Vx_{ij}}{R_{ij}} + \sum_{C(k,l)\in N_r^0(i,j)} Iw_{kl} + Iu_{ij}^{noi} \qquad \text{Eq.(2.6)}$$

$$Iw_{kl} = w_{kl}^{a} \times Iy_{kl} \qquad \text{Eq.(2.7)}$$

$$w_{kl}^{a} \in 1 \quad or \quad \frac{1}{2} \quad or \frac{1}{3} \quad or \quad \frac{1}{4} \qquad \text{Eq.(2.8)}$$

$$k,l \in i,j-1 \quad or \quad i,j+1 \quad or \quad i-1,j \quad or \quad i+1,j \qquad \text{Eq.(2.9)}$$

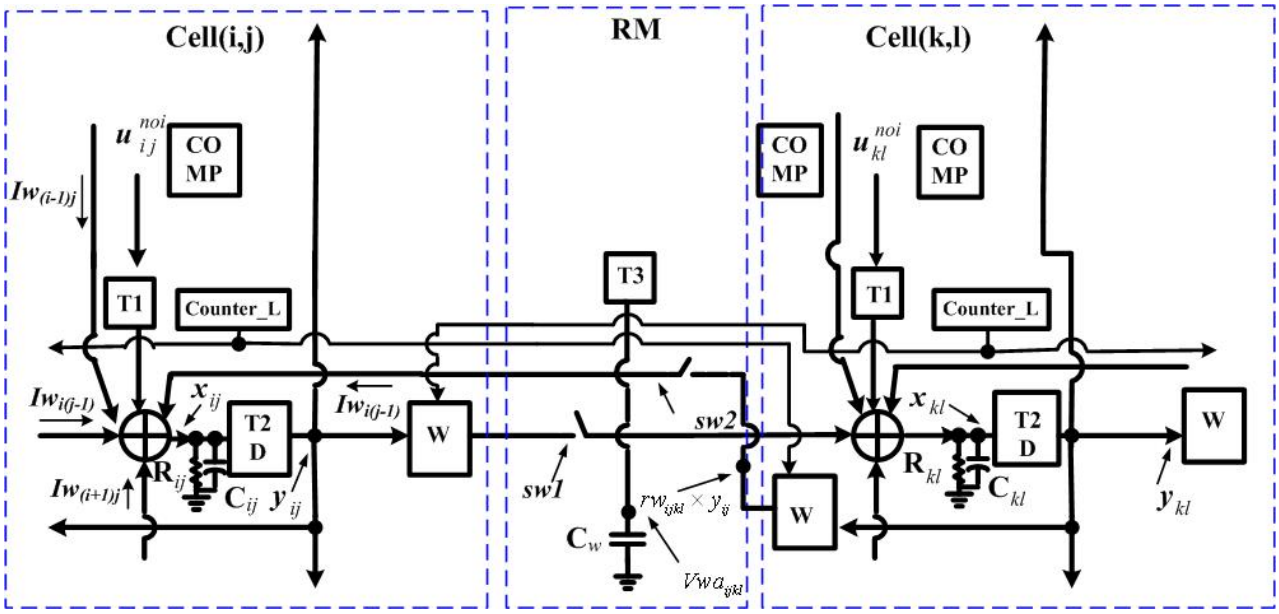

Fig. 2.6 Architecture of RMCNN in recognition period

Where w_{kl}^{a} is the template A ratio weight. It is generated by **W**. The Eq.(2.6) implement the RMCNN mathematical equation Eq.(1.1). Because of the settings of template B and threshold are Eq.(1.4) and Eq.(1.5). Thus in Eq.(2.6) there isn't the threshold and the coefficient of input Iu_{ij}^{noi} is 1.

2.2 Circuit Implementation

2.2.1 V-I Converter

The circuit of **T1** and R_{ij} is shown as Fig. 2.7. In all of the circuit implementation figure,

the MOS size is written next to the MOS number. The unit of MOS size is micro meter. In Fig. 2.7, the left side is a differential pair structure, and right side is MOS resistor. The voltage Vb1 and Vref are constant bias voltage. Vb1 is 2.5V and Vref is 1.5V. MOS M5 and M6 perform as large resistances to let the linear operating range larger. When the input voltage Vin is larger than Vref, the output current Io flows from left to right. Then the voltage Vx_{ij} rises. Similarly, when the Vin is smaller than Vref, the voltage Vx_{ij} falls.

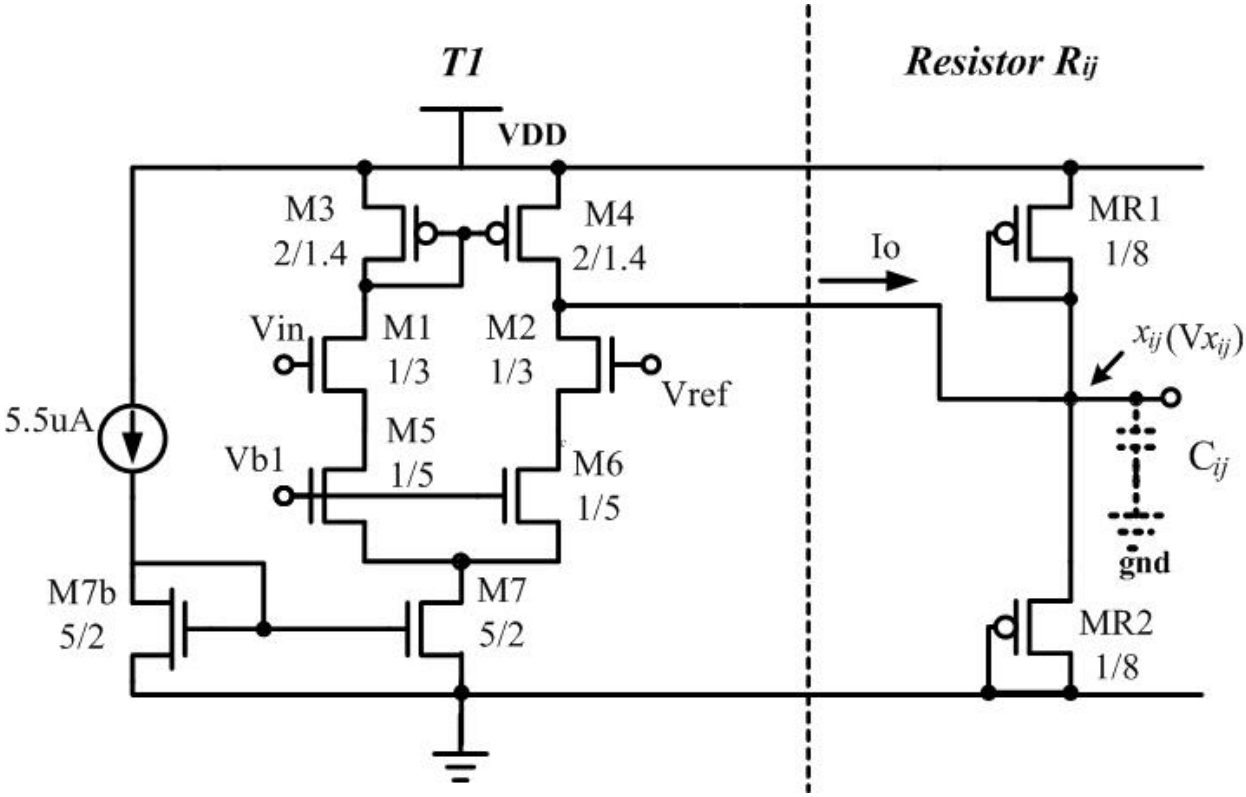

Fig. 2.7 The V-I converter T1

Fig. 2.8 is the circuit of **T2D**. **T2D** is similar to **T1**, but it has a detector and an absolute output current structure. The circuit of detector is shown as Fig. 2.9. The detector is just an inverter chain. It is used to detect the sign of **T2D** input, and the function of detector is described as Eq.(2.6). In Fig. 2.8, left side is also differential pair structure, and right side is the absolute output current structure. The constant bias voltage Vb2 is 1.5V, and the constant bias voltage Vb1 and Vref are the same with **T1**. When the input voltage Vin is larger than Vref, the current Io flows from left to right. Then the MOS M10 in Fig. 2.8 turns off, and MOS M11 turns on. The current are mirrored by current mirror M12 and M13, and flow

through the M8. Then the MOS M94 mirrors the current of M8 and output the current Ioabs. Similarly, if the voltage Vin is smaller than Vref, M10 turns on and M11 turns off. The output current Ioabs is mirrored by the current mirror M8 and M94 directly. Whether the input voltage Vin is larger than Vref or not, the flowing direction of Ioabs is always the same. So the circuit has an absolute output current. The usage of the MOS M26 will be explained in section 4.3.

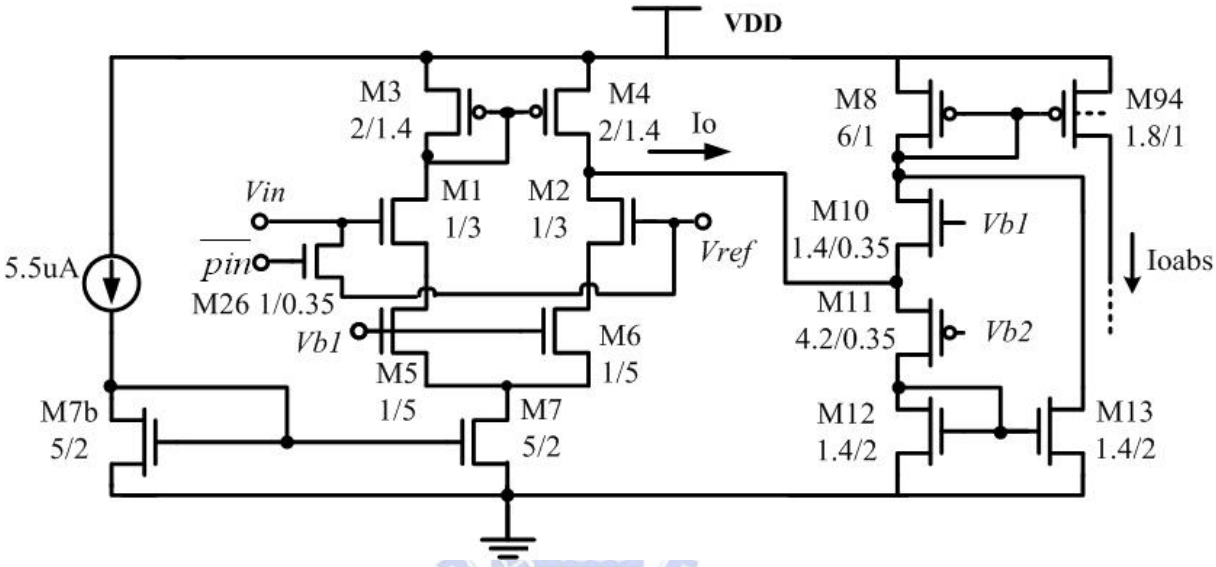

Fig. 2.8 The V-I converter in T2D

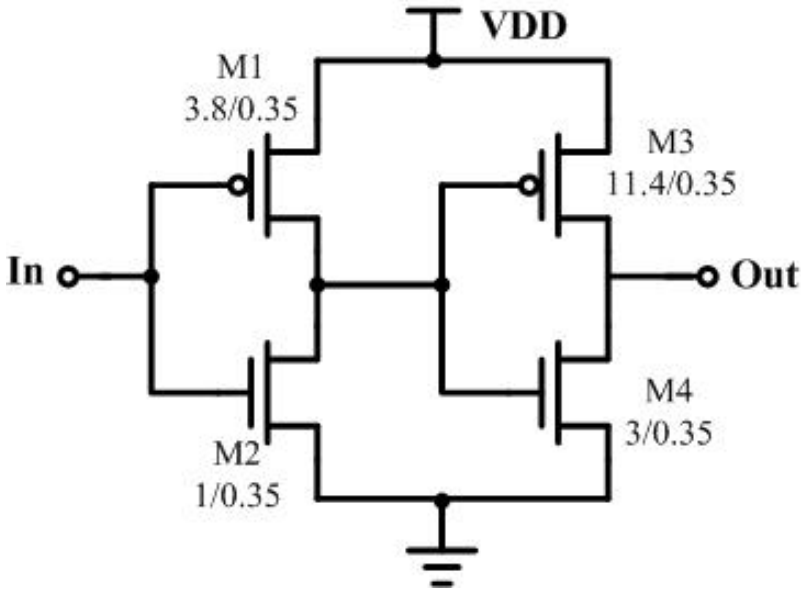

Fig. 2.9 The detector in T2D

The circuit of block **W** is Fig. 2.10. Actually, the block **W** is combined with **T2D**. In order to show the MOS size of these two circuits, the diagrams are drawn respectively. Note that the MOS M94 in Fig. 2.10 and the M94 in Fig. 2.8 are the same MOS. The complete circuit diagram of **T2D** and **W** is shown as Fig. 2.11. The function of **W** is to weight the output of **T2D**. We use current mirror to weight the output of **T2D**. In Fig. 2.10, because M94, M91, M92 and M93 are current mirror, we don't use minimum length to avoid strong channel modulation effect. In Fig. 2.11, the drain current of M94 is $\frac{1}{4} \times Ioabs$, but the size of M94 isn't really $\frac{1}{4}$ time of M8. Because even we use 1 micro meter channel length, the drain and source voltage drops Vds of M8 and M94 still influence the current accuracy. Thus the channel width of M94 is adjusted to modify the current accuracy. Similarly, the sizes of M92 and M93 are adjusted too. A better method to let the current mirror operate accurately is using MOS parallel connection. A small MOS is chosen as a unity MOS first. Then the M8 in **T2D** uses twelve unity MOSs that has parallel connection with each other and M94 uses three unity MOSs has parallel connection with each other. Similarly M91 uses twelve unity MOSs and M92 uses six unity MOSs and M93 uses four unity MOSs. This modified structure will has more accurate mirrored current.

The switches Sw_a, Sw_b, Sw_c, Sw_d, Sw_e and Sw_f are controlled by **Counter_L**. According to output of counter, only one path of these switches turns on at the same time. The XOR gate in Fig. 2.10 is used to control the flowing direction of output current. In learning period, the $Vin_{T1(k,l)}$ is inputted to the XOR gate and the Vin_{T3ijkl} is inputted to the XOR gate in recognition period.

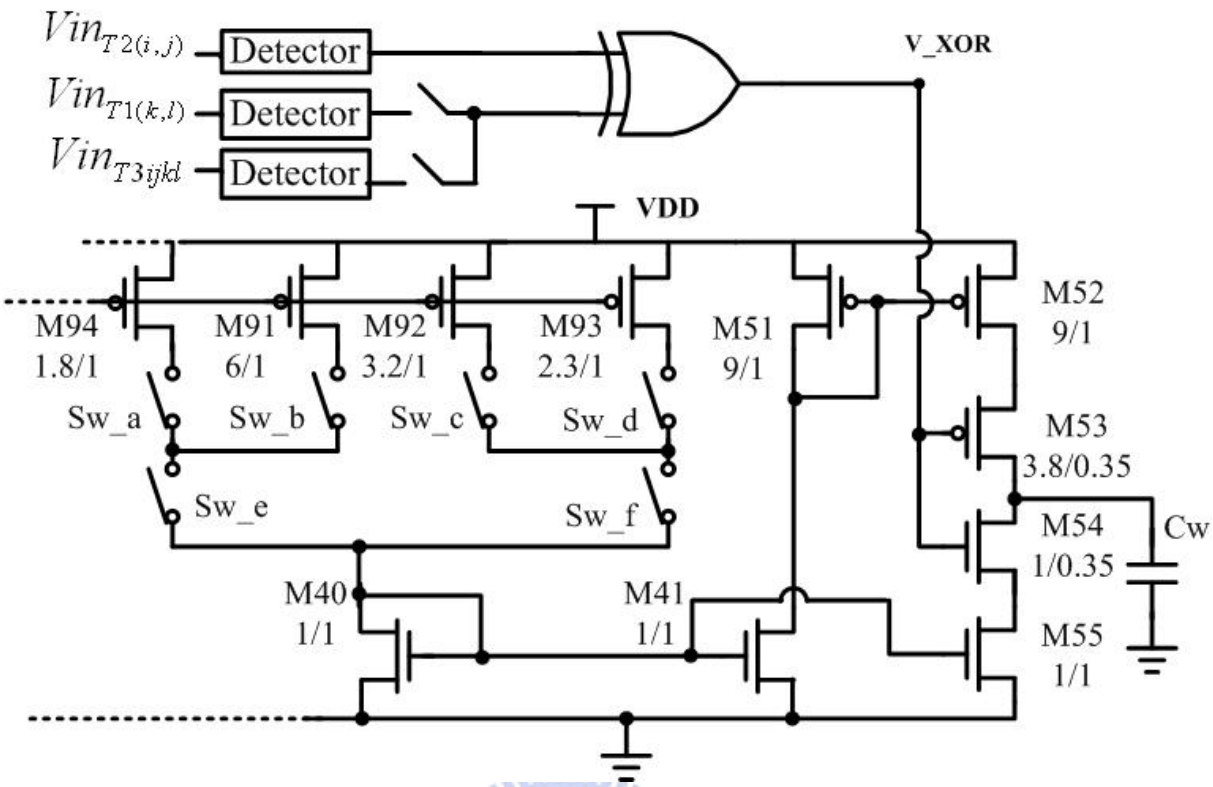

Fig. 2.10 The CMOS circuit of W

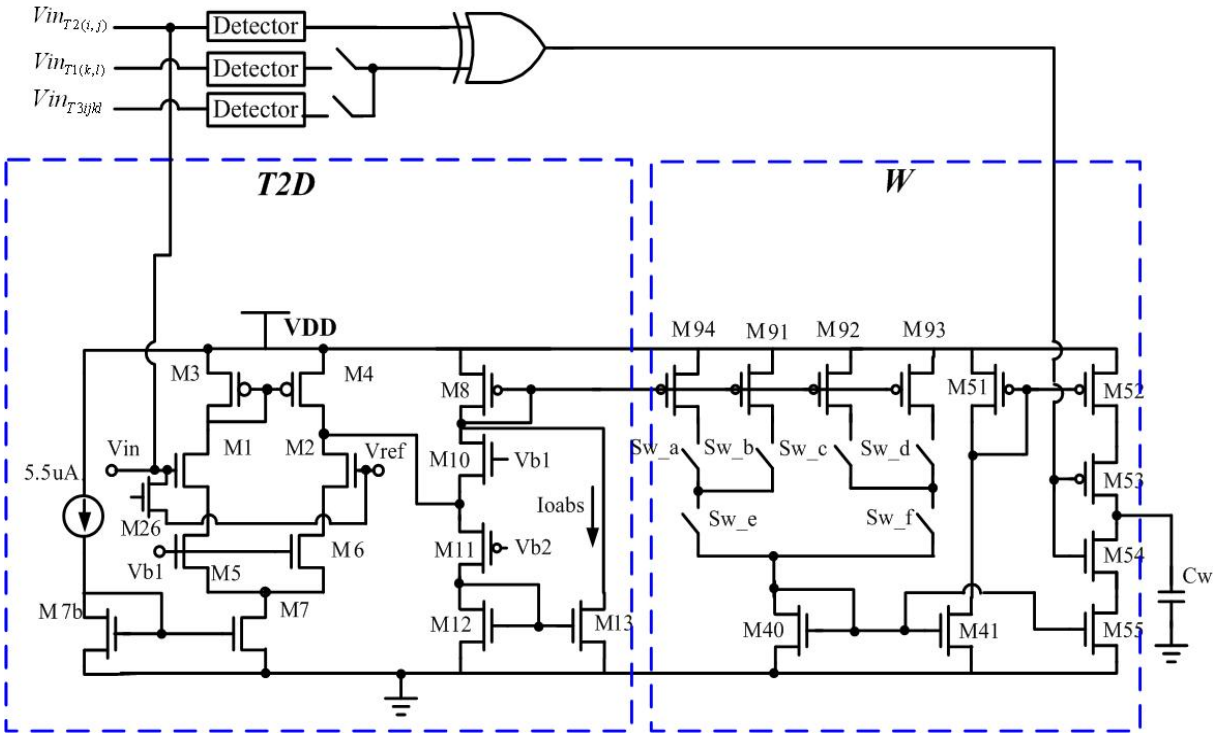

Fig. 2.11 The overview of T2D and W

The V-I converter **T3** is similar to **T2D**. The circuit is shown in Fig. 2.12. In Fig. 2.3, T3

has four outputs. Two of the four outputs are sent to **COMP**, and the others are sent for summation. Thus the circuit in Fig. 2.12 has four outputs, and the MOS sizes of the current mirrors (M9s1, M9s2, M9s3, M9s4 and M8) are the same.

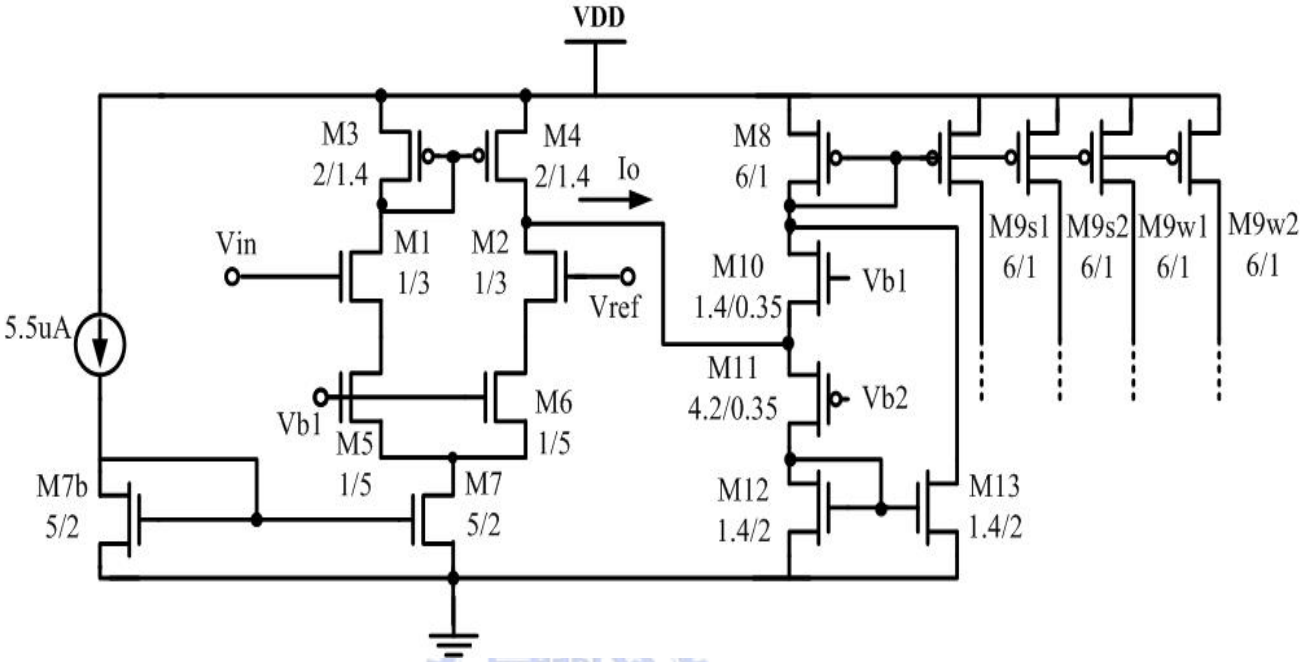

Fig. 2.12 The CMOS circuit of T3

2.2.2 Comparator

Fig. 2.13 shows the circuit of Comparator. In order to save the area of whole chip, we use a simple current mode comparator. In Fig. 2.13, if the input current I_{Mss} is larger than I_{aw}, the logic output *Vout* is low. Otherwise, *Vout* is high. The port I_{Mss} is used to receive the mean of summed currents, and the port I_{aw} receives the absolute-weight current that is transferred from T3. In the above algorithm, if the absolute-weight current equals to the mean of the summed current, the ratio weights should be preserved too. That means the logic output of comparator should be high if I_{Mss} equals to I_{aw}. Because the usages of I_{Mss} and I_{aw} are specified, the sizes of Mc3 and Mc4 are designed as little smaller than Mc1 and Mc2. The difference of the MOS size makes the logic output is high even if I_{Mss} equals to I_{aw}.

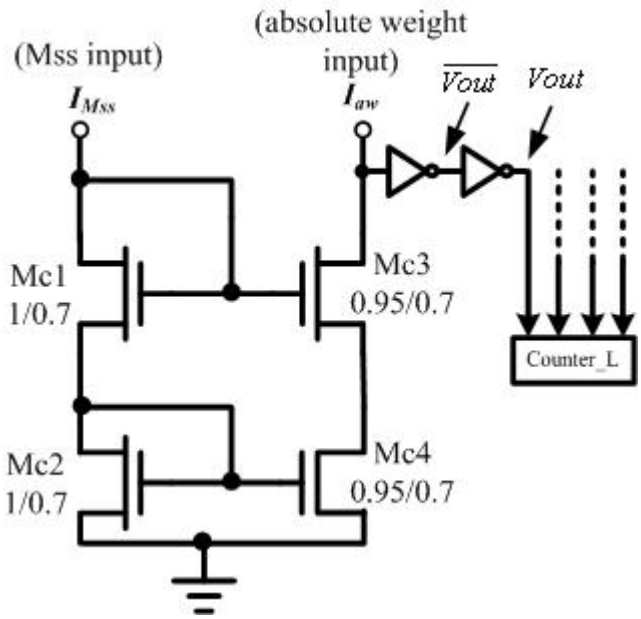

Fig. 2.13 The CMOS circuit of comparator (COMP)

In section 2.1, it is described that we need to count the mean of four absolute-weight current. That means it is necessary to divide a summed current by four. But there isn't any divider in this circuit, the dividing behavior is implemented by the wire connection of **COMP**. The detail is shown as Fig. 2.14. In Fig. 2.14, two of the T3 output ports are drawn, and the others are abridged. The four output currents of $T3_{i(j+1)}$, $T3_{i(j-1)}$, $T3_{(i+1)j}$ and $T3_{(i-1)j}$ are summed at the node N and form the current I_{sum}. Because the connection of MOS Mc1 and Mc2 in Fig. 2.13 are diode connection, they are all in saturation region. The input impedance of I_{in1} port is very large and isn't sensitive to the drain and source voltage drop Vds and the flowing current. In Fig. 2.14, node N is connected to the input of all four comparators. Because of the similar input impedance of the four comparators, the current I_{sum} flows into the four comparators averagely. Thus the currents flow into Mc11, Mc12, Mc13 and Mc14 are $\frac{1}{4}I_{sum}$ and the current $\frac{1}{4}I_{sum}$ is the mean of summed current.

Process variation is considered in the RMCNN w/o EO. If the capacitor Cw and all of the V-I converter have process variation, the **COMP** can't get the accurate current. However, if the neighboring five cell has the same process variation, the comparative magnitude of

absolute weights I_{aw} and the mean I_{Mss} doesn't change. Thus the RMCNN w/o EO has a little tolerance to process variation.

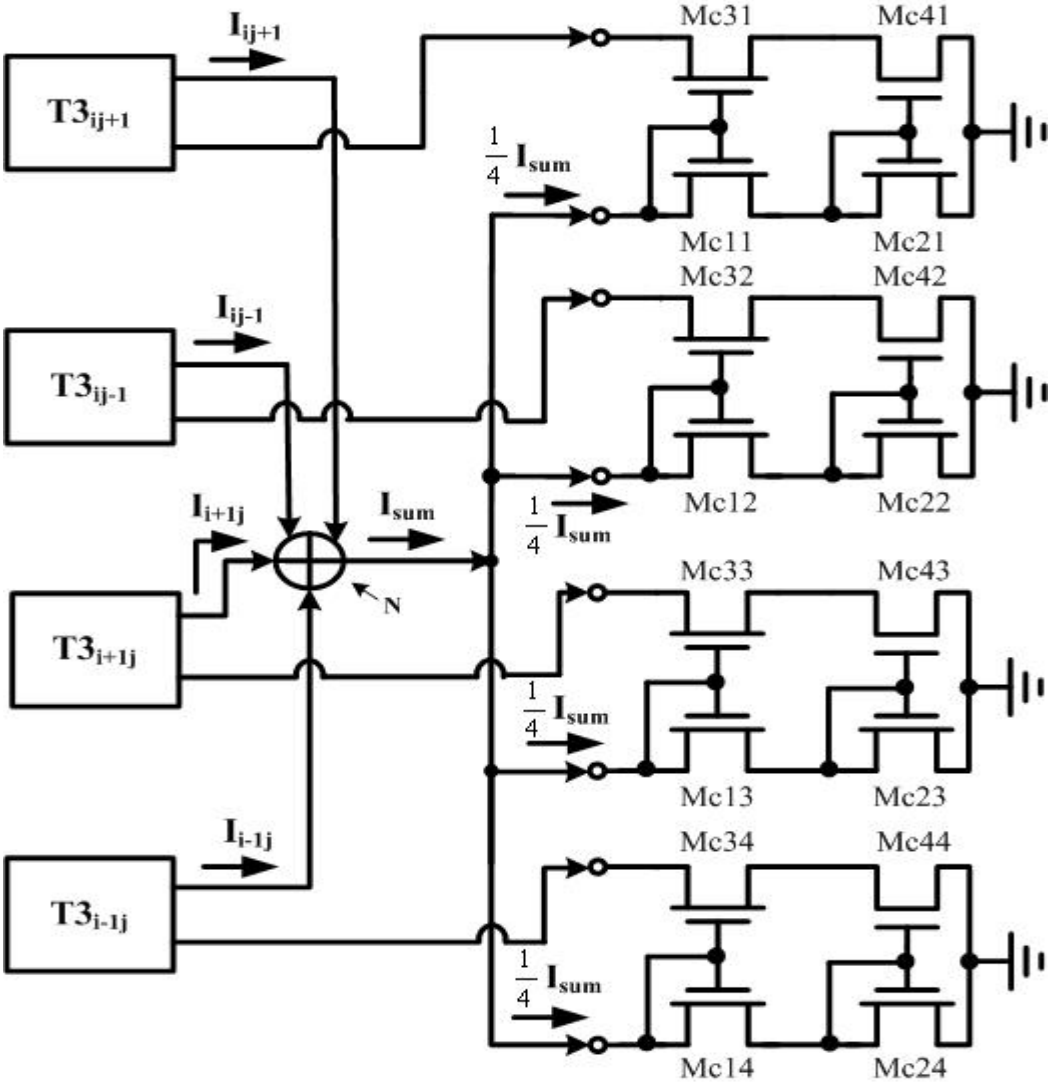

Fig. 2.14 The method that divides the summed current by 4

2.2.3 Counter and Weight Selection Structure

The counter in this architecture is formed by two D-flip-flop. The structure of counter is shown as Fig. 2.15. DFF_P is a positive edge trigger D-flip-flop, and DFF_N is a negative edge trigger D-flip-flop. The MOS M1 is used to reset the signal *Cou_L*(*Cou_G)*. Switch S_en enables the counting operation. The counting operation can be described as Fig. 2.16. Note that b0 is the output of positive edge trigger D-flip-flop, and b1 is the output of negative edge trigger D-flip-flop. If the signal R is high, *Cou_L(Cou_G)*, b0 and b1 are always low.

When the signal S_en is low, b0 and b1 don't change even if *Cou_L(Cou_G)* is oscillating.

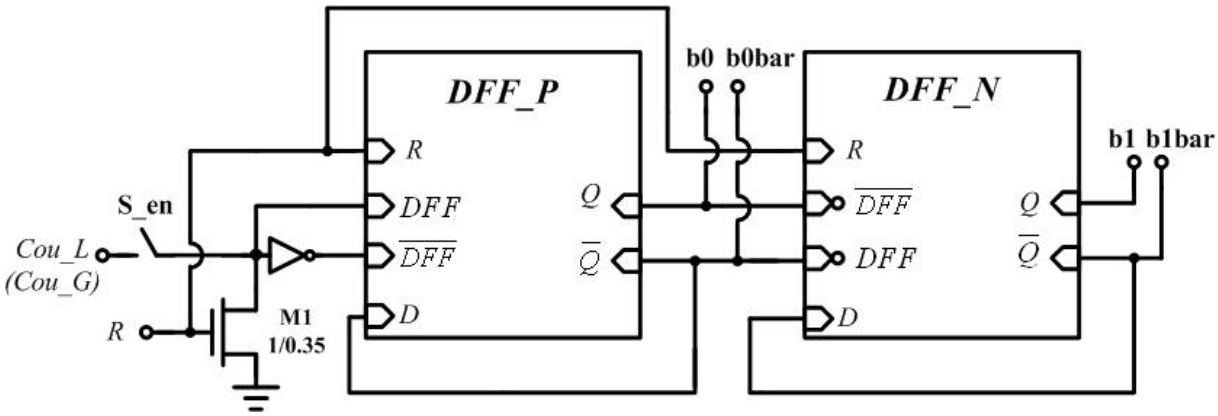

Fig. 2.15 The circuit of the counters in this chip

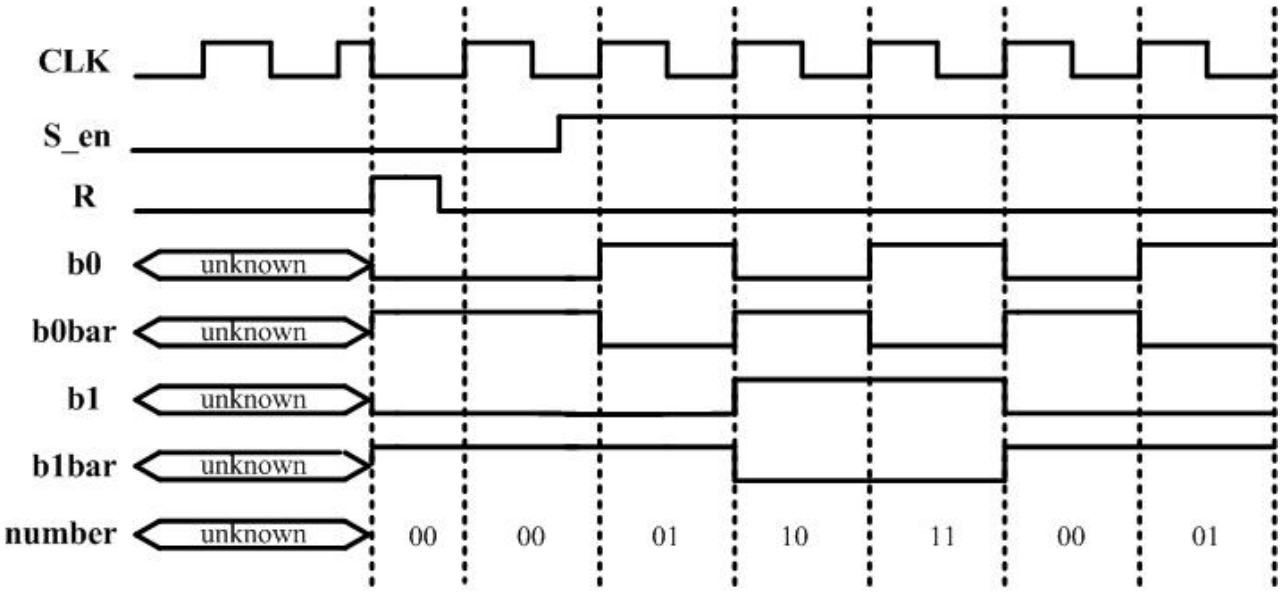

Fig. 2.16 A counting example of the counter

Dynamic D-flip-flop is used in this chip, because the transistors count is less than static D-flip-flop. The circuit of the dynamic D-flip-flop is shown in Fig. 2.17. MOS M0 and M9 are used to reset the output of D-flip-flop. Fig. 2.17 is a positive edge trigger D-flip-flop. If change the port position of *DFF* and $\overline{DFF}$, that's negative edge trigger D-flip-flop. The D-flip-flop in Fig. 2.17 has static power consumption when the port *D* and *R* is high and *DFF* is low. Thus we should use static D-flip-flop instead of the dynamic D-flip-flop to save power

consumption..

In Fig. 2.10, it is known that Sw_a~Sw_f are controlled by counter. Fig. 2.18 shows how the counter controls those switches. In Fig. 2.18, some I/O ports are abridged. In a *r*-neighborhood system $N_r(i, j)$ (r=1) of the cell cell(i, j), four comparators connect with a counter. The output of each comparator controls the switch S_en of the counter. The switches S_en1~S_en6 are controlled by another global counter **Counter_G** and only one path of S_en1~S_en6 turns on at the same time. The controlling method of S_en1~S_en6 is similar to Sw_a~Sw_f. At the ending of learning period, *Cou_L (Cou_G)* oscillates four times. The *Cou_L (Cou_G)* oscillates, the turn-on path of S_en1~Sen6 changes. If the output of **COMP** is high, the binary number output of counter adds one. That's the method used to count how many ratio weights should be preserved

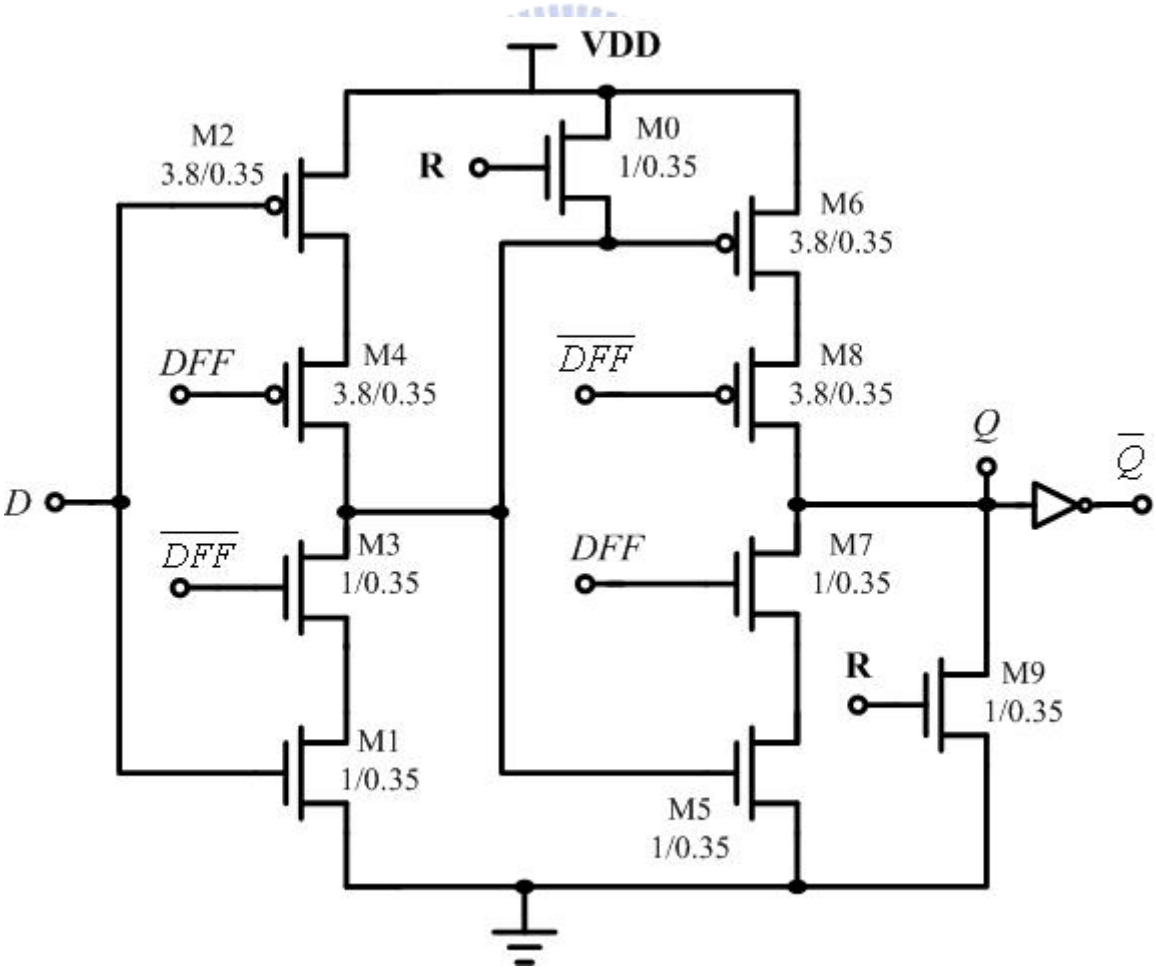

Fig. 2.17 The circuit of DFF_P

Every cell has a **Counter_L**, but there is only one **Counter_G** that is used to control switches S_en1~S_en6 in the whole chip circuit. Fig. 2.19 shows how the **Counter_G** controls the switches S_en1~S_en6 in every cell. The **Counter_G** is drived by the signal *Cou_G*, and all **Counter_L** are driven by the signal *Cou_L*.

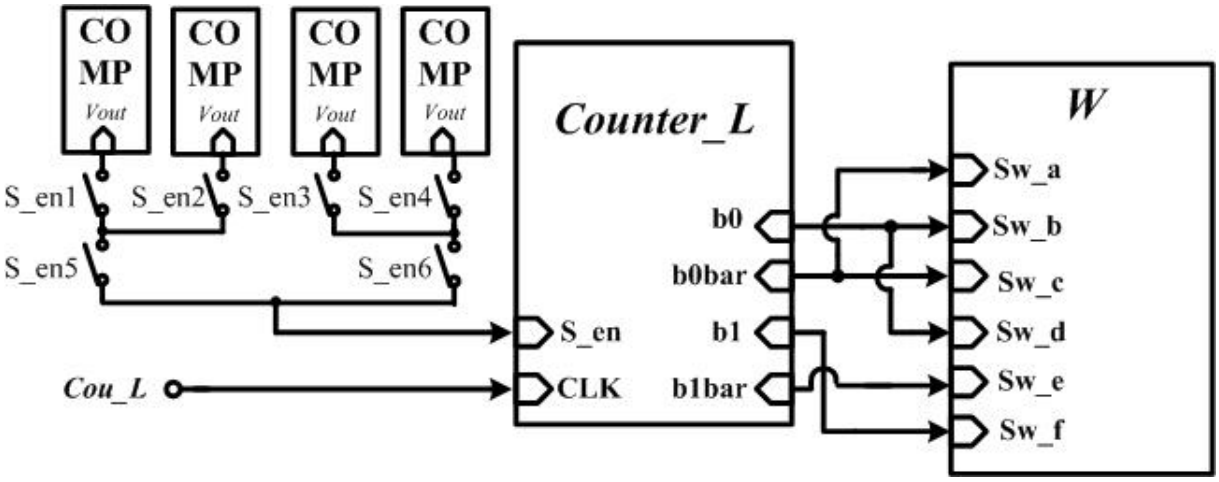

Fig. 2.18 The connection between W and Counter_L

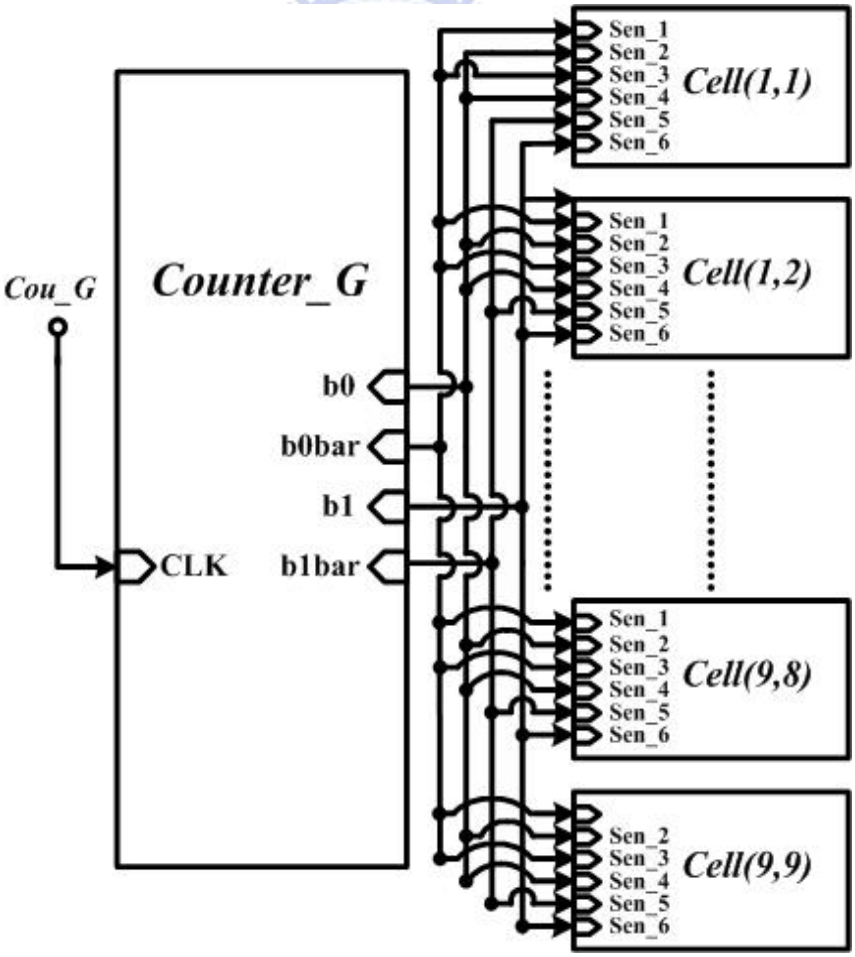

Fig. 2.19 The connection between Counter_G and every cell

2.2.4 Output stage and input pattern interface

The output stage is shown as Fig. 2.20. The nodes x_{11}~x_{99} are the node x_{ij} in Fig. 2.4. M11~M99 perform as level shifter to drive parasitical capacitance of the switches and metal line. The unit gain buffer is a negative feedback OP and it is used to drive the output pad. The circuit of unit gain buffer is shown as Fig. 2.22. Two 4-bit decoders are used to control those switches Sw_{c11}~Sw_{c99} and Sw_{r1}~Sw_{r9}. One decoder controls column switches Sw_{c11}~Sw_{c19} (Sw_{c21}~Sw_{c29}, Sw_{c31}~Sw_{c39}, …etc.), and the other controls switches Sw_{r1}~Swr9. This structure is used to read out every pixel one by one.

There are some current source can be shared in the output stage shown in Fig. 2.20. The modified output stage is shown as Fig. 2.21. In Fig. 2.21 every MOS in the same row uses one current source. This modified output stage saves much power consumption.

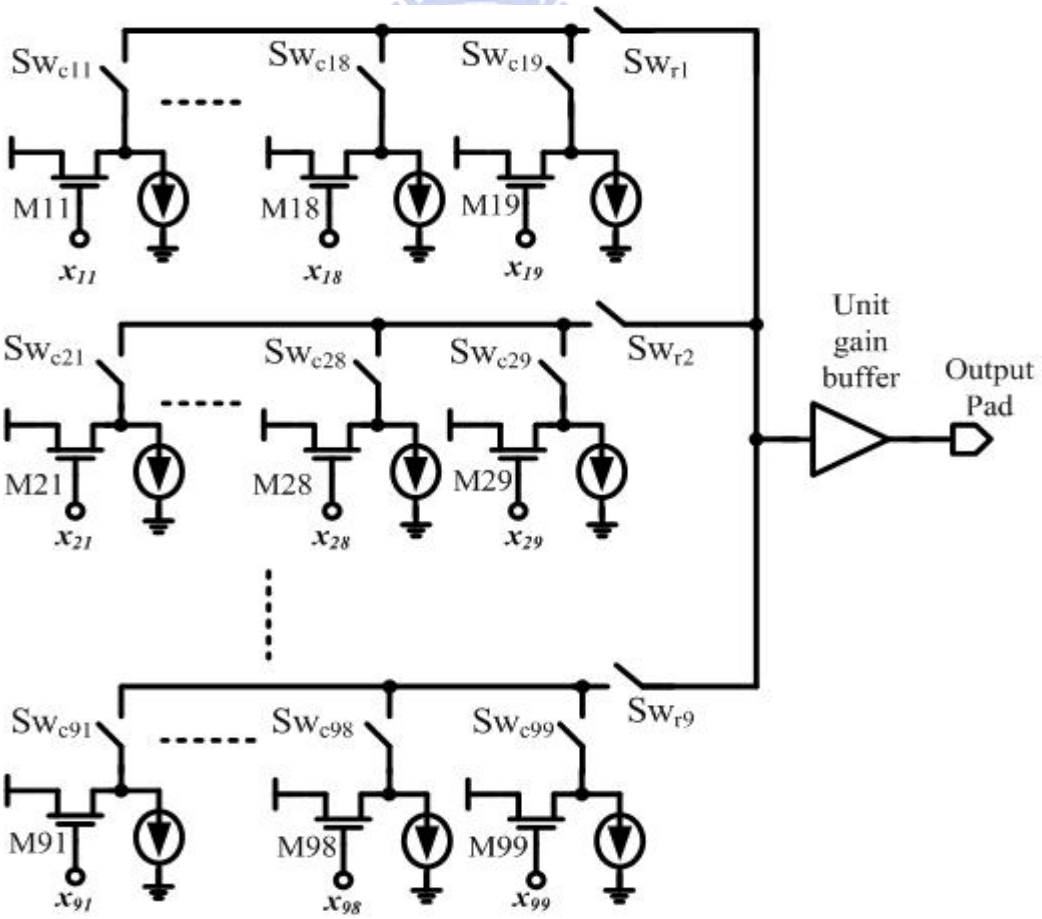

Fig. 2.20 The output stage

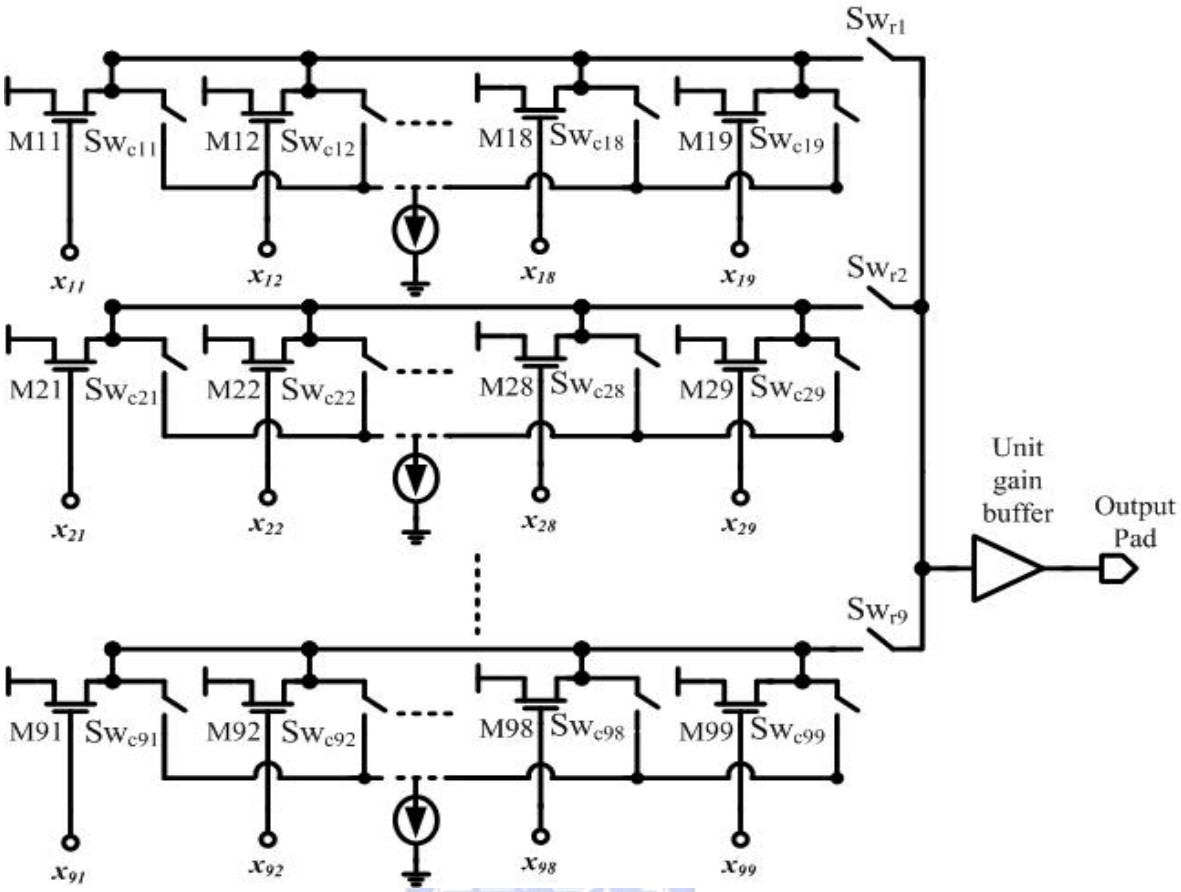

Fig. 2.21 The modified output stage

In order to input any arbitrary learning patterns, the shift registers input interface is used. Fig. 2.23 shows the input interface. *DFF_N* is negative edge trigger D type flip-flop. In the beginning of learning period, *clk1* and *newp* turn on and ptn_i inputs the learning pattern pixel by pixel. After the CLK of *DFF_N* oscillates nine times (because the cell array has 9 columns), *pin* turns on to input the learning pattern into each cell. When *pin* turns on, *newp* turns off to prevent the pattern changes as a glitch occurs on CLK of *DFF_N*. After the first pattern is learned, *clk1* and *newp* turn on again and *pin* turns off. Then shift registers transfer the stored learning pattern and the learning of the second pattern starts.

Fig. 2.24 is one part of Fig. 2.23 and it shows how to mix the noise with learning pattern in recognition period. The capacitance C_{gp} is the gate capacitance of M1 in Fig. 2.7 and other parasitical capacitance. In learning period, the capacitance C_{noi} is pre-charge to V_{noi} and *noi* always turns off. When recognition period starts, the innocent pattern is already stored in shift

register and *clk1* turns off to isolate D-flip-flop. Then *noi* turns on and charge sharing occurs between C_{gp} and C_{noi}. So the voltage on node *Nd* is a mid level voltage and the amplitude can be adjusted by changing the capacitance ratio of C_{gp} and C_{noi}.

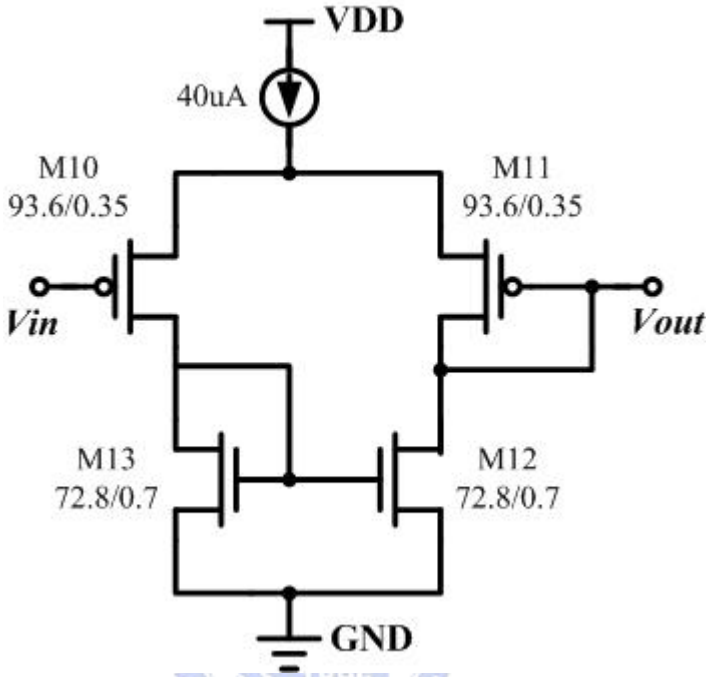

Fig. 2.22 The unit gain buffer in the output stage

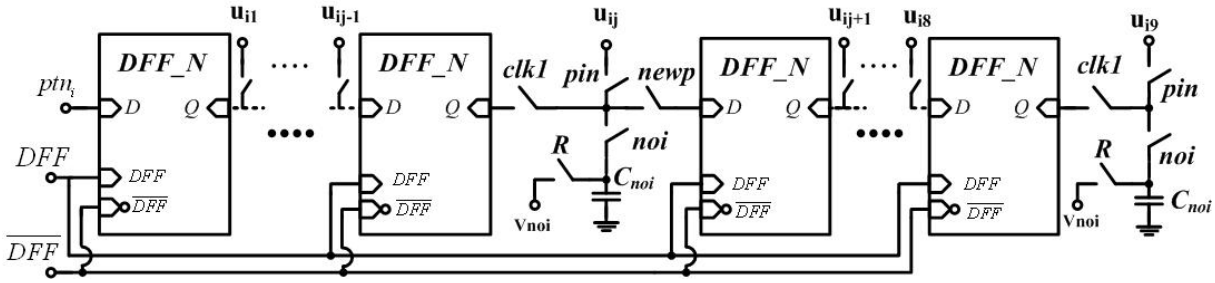

Fig. 2.23 The pattern input interface that formed by shift register

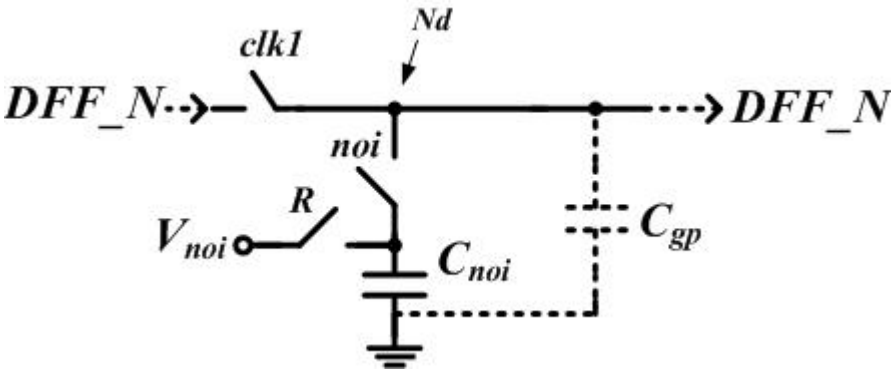

Fig. 2.24 The structure that used to mix noise with innocent pattern

CHAPTER 3

SIMULATION RESULT

3.1 Matlab Simulation Result

The MATLAB software is used to simulate the behavior of the CNN with ratio memory (RMCNN) as an associative memory. In the MATLAB simulation, 9x9 cells are used to form the RMCNN with r = 1. Thus, it can process patterns with 81 pixels. The total three learning pattern is shown as Fig. 3.1. The patterns are Chinese character “one”, “two” and “four”. Normal distribution and uniform distribution noise are both mixed with the clear pattern respectively, and the Matlab simulation result shows that the three patterns can be recovered. Fig. 3.2 shows the three patterns mixed with normal distribution noise. Fig. 3.3 shows the three patterns mixed with uniform distribution noise.

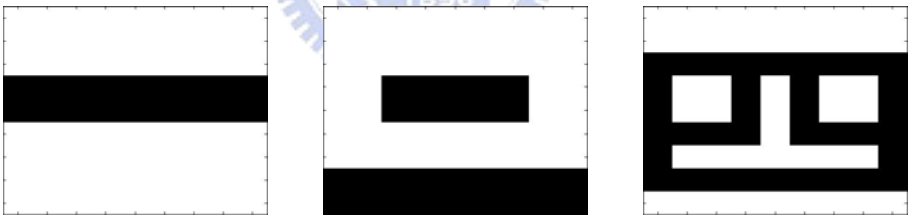

Fig. 3.1 The three clear learning patterns

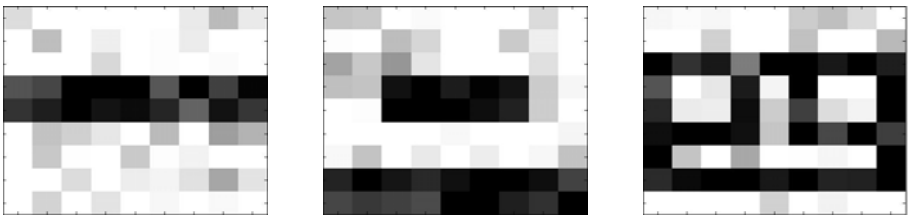

Fig. 3.2 Patterns mixed with normal distribution noise (standard deviation:0.5)

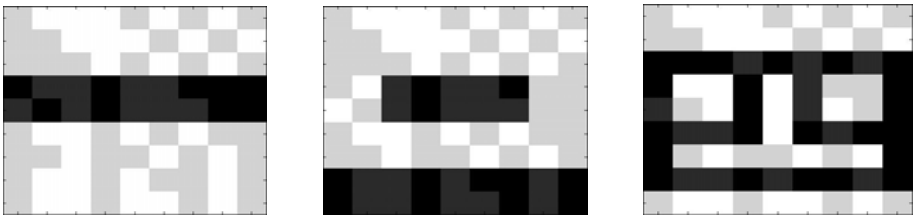

Fig. 3.3 Patterns mixed with uniform distribution noise

The design in this thesis implements a method that generates ratio weights without elapsed operation. Table 3.1 compares the ratio weights generated by elapsed operation and ratio weights generated by this design. In the RMCNN with elapsed operation design, the absolute-weights stored on capacitance are decayed by leakage current. To consider the leakage current effect, a constant leakage current of 0.8 fA is applied to the capacitor C*ss* of 2 pF. In Table 3.1, the elapsed time is 800s. Some small ratio weights generated by elapsed operation don't decay to zero completely, and some largest ratio weights generated by elapsed operation don't enhance to one. So the ratio weights aren't feature enhanced enough. If the elapsed time is longer (for example: 850s), the ratio weights generated by elapsed operation can be feature enhanced completely. But if the elapsed time is too long, the ratio weights disappear (because all of the absolute-weights decay to zero). RMCNN w/o EO doesn't have this trouble. We needn't tune the best elapsed time and the circuit can get the best feature enhanced ratio weights.

In Matlab simulation result, not all of the noisy pattern can be recognized. If the intensity of mixed noisy is very strong, RMCNN can't recognize the noisy pattern too. Two kinds of noise are simulated in this thesis: normal distribution and uniform distribution. If the standard deviation of noise is larger than 0.3, the recognition rate is lower than 90%.

The recognition rate is also simulated. Ninety random noisy patterns (thirty noisy patterns for each Chinese character) are generated by Matlab and recognized. Fig. 3.4 shows the recognition rates of three algorithms. The "CNN without RM" means that the algorithm recognizes noisy patterns directly after learning process. It doesn't have the feature enhanced ratio weights, and its recognition rate is worst. Chinese character "four" always can't be recognized. The recognition rates of "RMCNN with elapsed operation" and "RMCNN without elapsed operation" is similar. In Fig. 3.4, the elapsed time of "RMCNN with elapsed operation is 800s. So the recognition rate of "RMCNN without elapsed operation" is lightly better than "RMCNN with elapsed operation". If the elapsed time is 850s, the two recognition

rates are the same completely because they get the same ratio weights.

Table 3.1 The ratio weights generated by (1) RMCNN with elapsed operation (2) RMCNN w/o EO

Ratio Weights	With elapsed operation	Without elapsed operation
9x9 $r = 1$	A_{45} (800s)= $\begin{bmatrix} 0 & -0.49 & 0 \\ 0 & 0 & 0 \\ 0 & 0.49 & 0 \end{bmatrix}$ A_{53} (800 s)= $\begin{bmatrix} 0 & 0.944 & 0 \\ 0.018 & 0 & 0.018 \\ 0 & -0.018 & 0 \end{bmatrix}$ A_{85} (800 s)= $\begin{bmatrix} 0 & 0 & 0 \\ 0.49 & 0 & 0.49 \\ 0 & 0 & 0 \end{bmatrix}$ A_{75} (800 s)= $\begin{bmatrix} 0 & 0.311 & 0 \\ 0.311 & 0 & 0.311 \\ 0 & 0 & 0 \end{bmatrix}$	$A_{45} = \begin{bmatrix} 0 & -0.5 & 0 \\ 0 & 0 & 0 \\ 0 & 0.5 & 0 \end{bmatrix}$ $A_{54} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ $A_{85} = \begin{bmatrix} 0 & 0 & 0 \\ 0.5 & 0 & 0.5 \\ 0 & 0 & 0 \end{bmatrix}$ $A_{75} = \begin{bmatrix} 0 & 0.333 & 0 \\ 0.333 & 0 & 0.333 \\ 0 & 0 & 0 \end{bmatrix}$

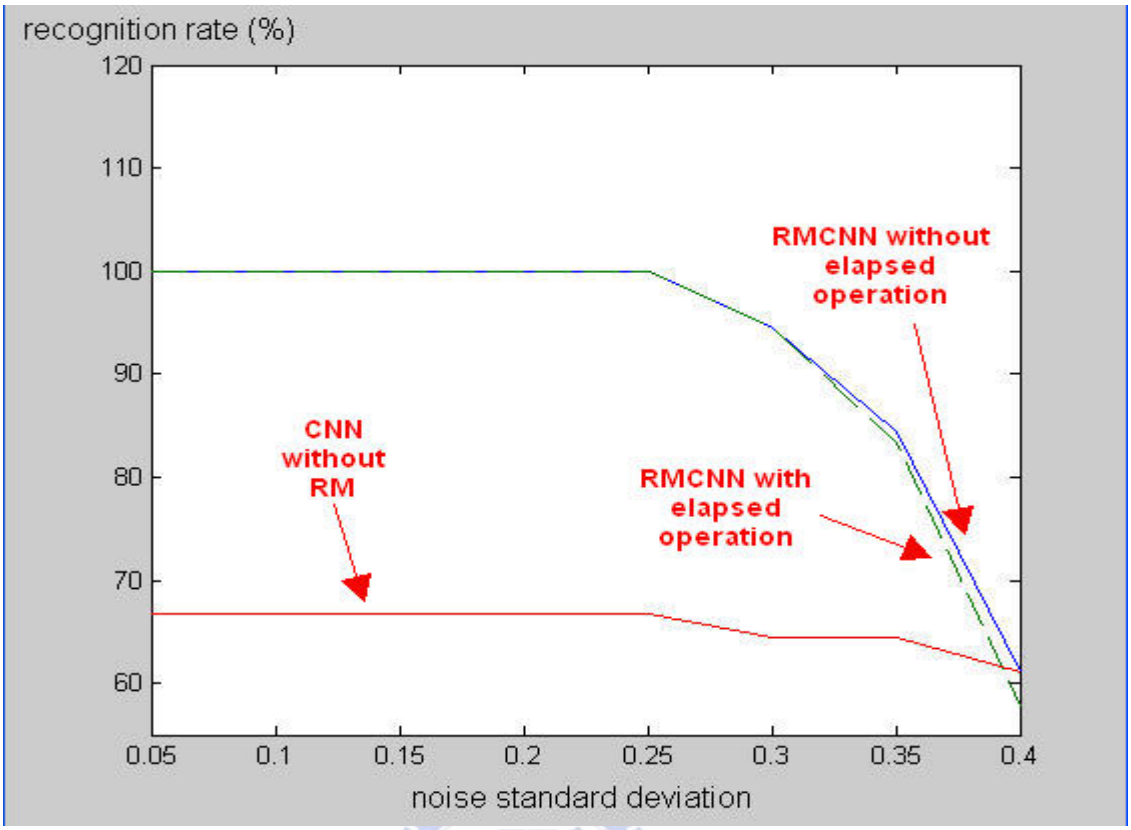

Fig. 3.4 Recognition rate of Matlab simulation (1) CNN without RM (2)RMCNN with elapsed operation (3) RMCNN w/o EO

3.2 Hspice Simulation Result

The simulation of **T1** and R_{ij} shown in Fig. 2.7 is shown as Fig. 3.5. When the input voltage is between 0.9V and 2.1V, the transfer curve in Fig. 3.5 is linear. If the input voltage of **T1** is smaller than 0.9V or larger than 2.1V, the output voltage is saturated. Thus it is described in chapter 2 that the voltage level 2.1V (0.9V) is defined as +1 (-1). Fig. 3.6 shows the simulation result of **T2D**. Because the output current of **T2D** is an absolute current, the flowing direction of the output current is the same when the input voltage of **T2D** is larger or smaller than 1.5V. The transfer curve of **T2D** is linear when the input voltage is between 0.9V and 2.1V. The simulation result of **COMP** is shown as Fig. 3.7. In Fig. 3.7, The input current I_{Mss} is swept and I_{aw} is kept as constant. Fig. 3.7 has three rows. The first row is the overall observation of .DC simulation. To observe the dead zone of the **COMP**, the second row of

Fig. 3.7 is the transfer curve which is zoomed out. In Fig. 3.7, the first and second rows are the transfer curve of *Vout* in Fig. 2.13, and the third row is the transfer curve of $\overline{Vout}$ in Fig. 2.13. Fig. 3.7 shows the dead zone of the comparator is about 10nA.

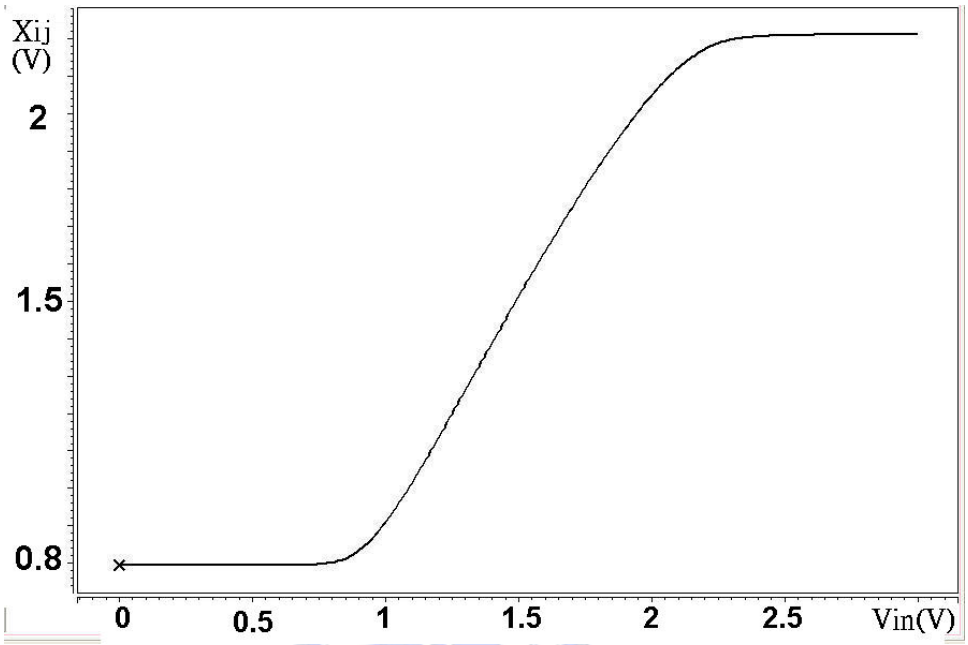

Fig. 3.5 Transferring curve of the V-I converter T1 and R_{ij}

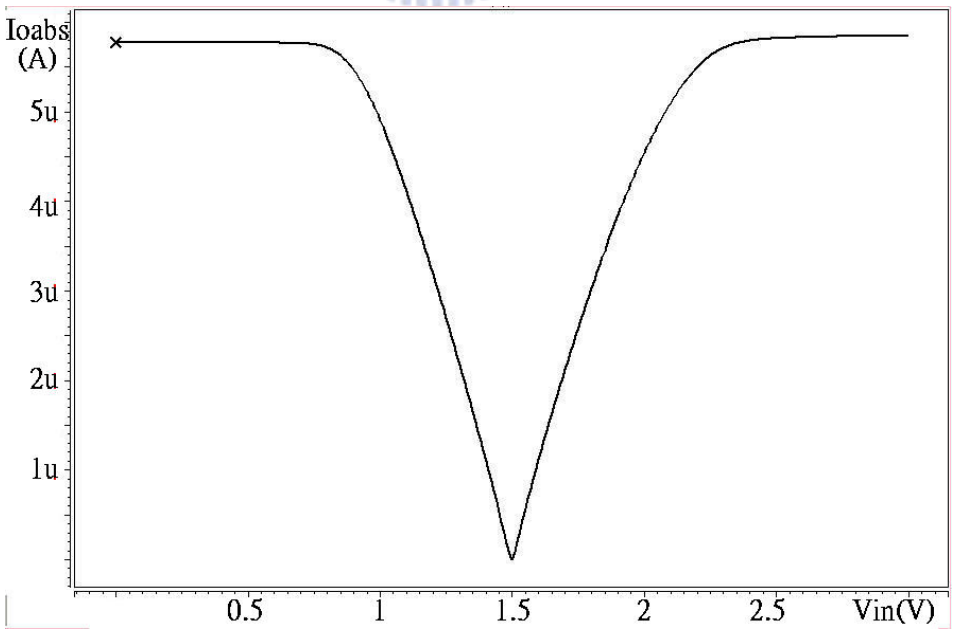

Fig. 3.6 Transferring curve of the V-I converter T2D

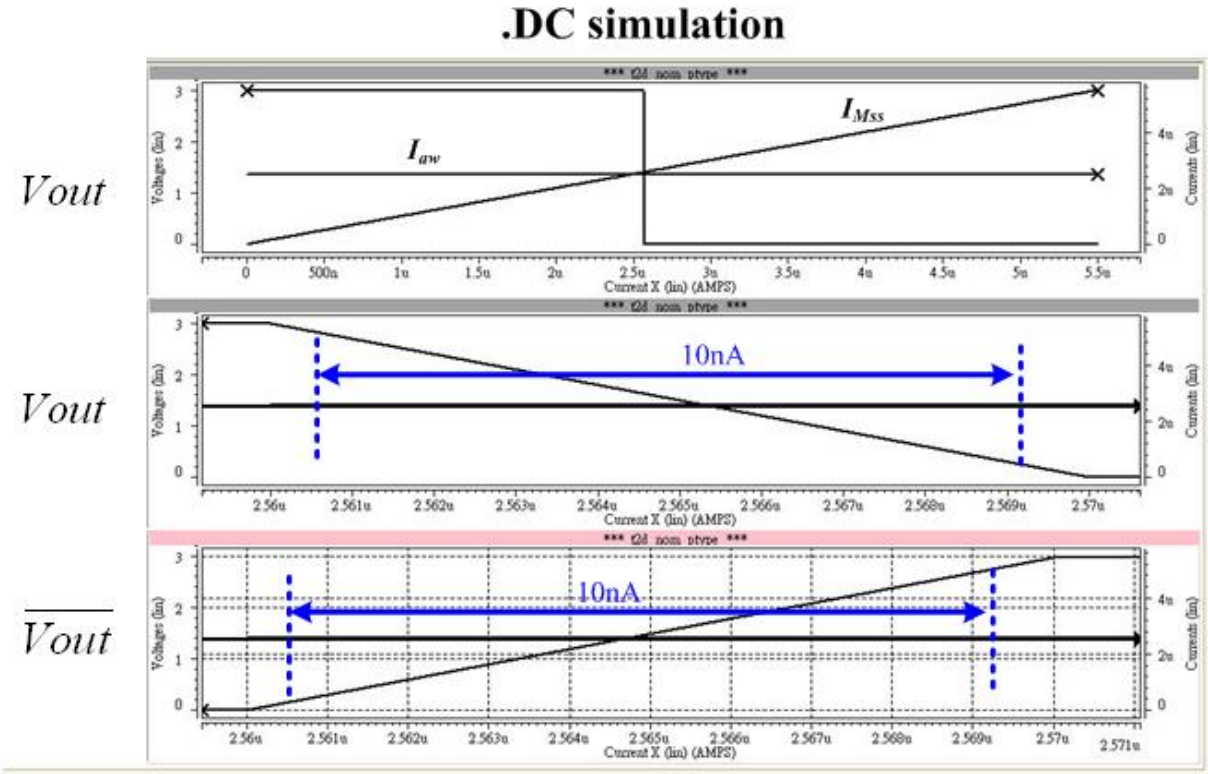

Fig. 3.7 .DC Simulation result of comparator

Fig. 3.8 and Fig. 3.9 show the simulation result of the unit gain buffer in Fig. 2.20 and Fig. 2.21. Fig. 3.8 shows the frequency response of the OP in Fig. 2.21 and Fig. 3.9 shows the difference between *Vin* and *Vout* of the unit gain buffer in Fig 2.18. Table 3.2 is the specification of the OP in Fig. 2.21.

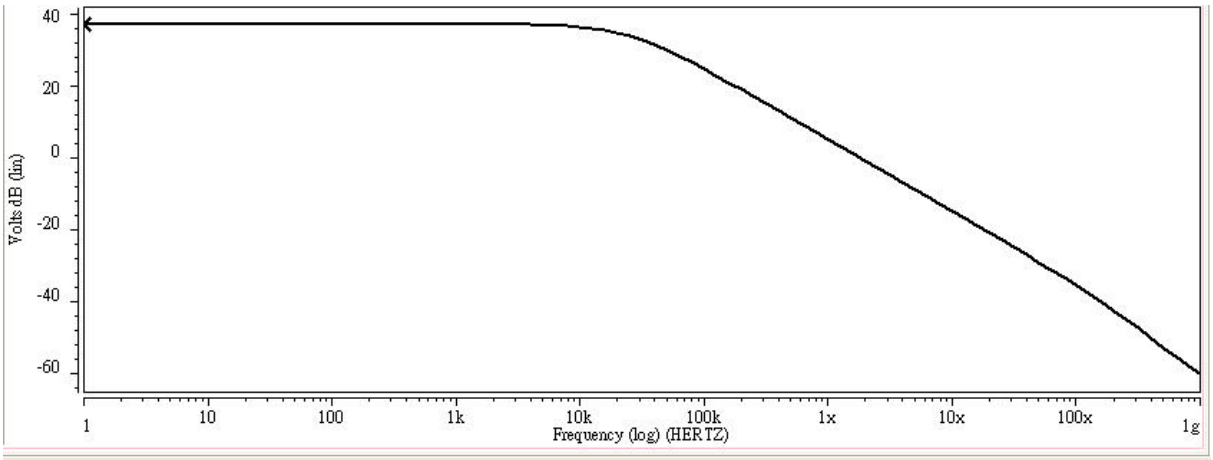

Fig. 3.8 Frequency response of the OP that performed as unit gain buffer

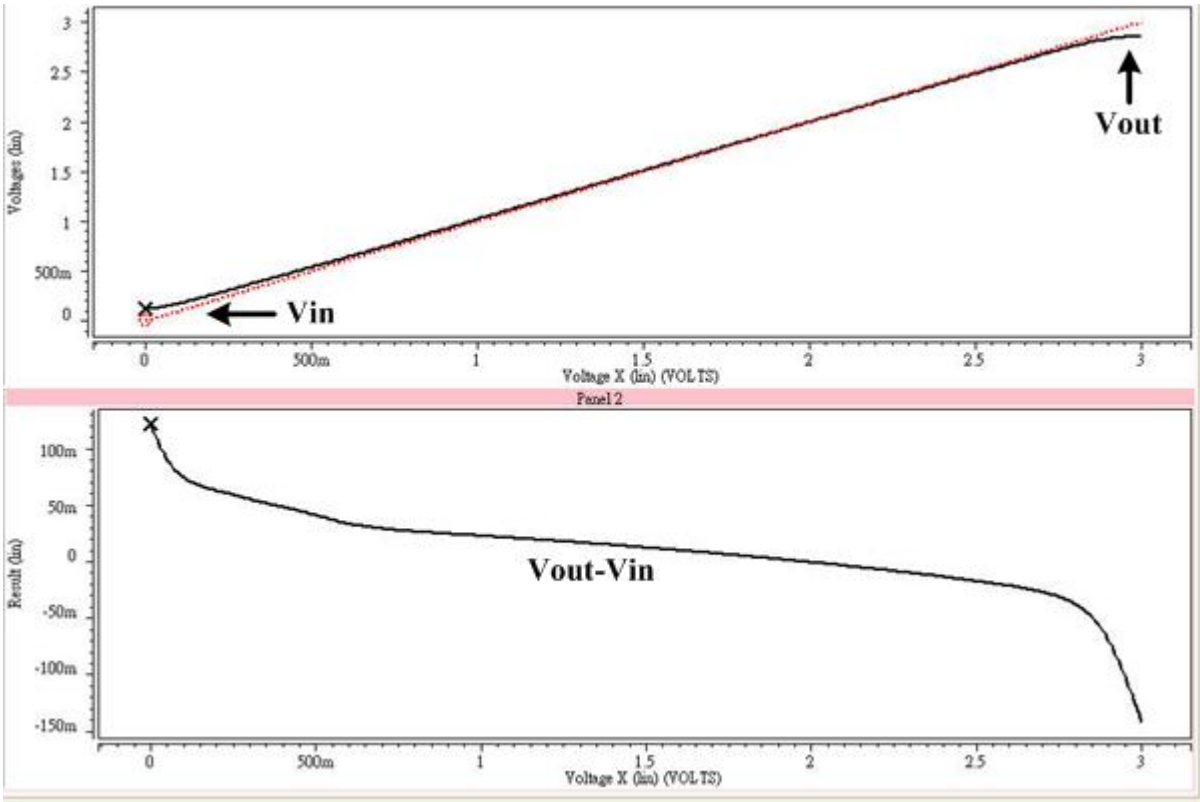

Fig. 3.9 The voltage difference between Vin and Vout of unit gain buffer

Table 3.2 Specification of the OP performed as unit gain buffer

DC gain	37.2 dB
3dB freq	24K Hz
Unit gain freq	1.8M Hz
Load capacitor	20p
Bias current	800 uA

The Whole chip recognition process is also simulated by Hspice. Because there are 81 pixels, it isn't feasible to show the learning and recognition process of all pixels. Thus several pixels are shown as examples. All of the pixels are checked and they are all recovered.

Fig. 3.10~Fig. 3.13 show the whole chip learning and recognition process of four pixels. In Fig. 3.10~Fig. 3.13, circuit learns patterns in "learning period", and the "pattern transferring" is used to transfer the learning patterns stored in shift register. The timing "counter" means the counter is counting how many ratio weights are preserved. In "noisy pattern read in", the noisy pattern that supposed to be recognized is inputted into the circuit.

After the “noisy pattern read in”, the recognition process starts.

It is described in chapter 2 that the pure black voltage level is defined as 2.1V and the pure white voltage level is defined as 0.9V. Fig. 3.10 is the operation process of the second row and the fourth column pixel *P(2,4)* and Fig. 3.11 is the operation process of *P(2,2)*. *P(2,2)* is a white pixel with noise, and *P(2,4)* is a white pixel without noise. When “noisy pattern read in” starts, the voltage level of *P(2,4)* is between 0.9V and 2.1V. Thus that’s a gray pixel. When recognition period begins, the voltage level of *P(2,4)* is pulled blow 0.9V, thus *P(2,4)* is recognized and recovered. *P(2.2)* is also pulled blow 0.9V after recognition period. Thus the *P(2,2)* is recognized too. Fig. 3.12 shows the operation process of *P(3,8)*, and Fig. 3.13 shows the operation process of *P(3,2)*. P(3,8) is a black pixel without noise, and P(3,2) is a black pixel with noise. Similarly, when “noisy pattern read in” starts, voltage level of P(3,2) is between 0.9 and 2.1V. That means P(3,2) is a gray pixel in this timing. After recognition period, this pixel is pulled over 2.1V, and that shows it is recover to a pure black pixel. Similarly, P(3,8) is pulled over 2.1V too, and it is recognized.

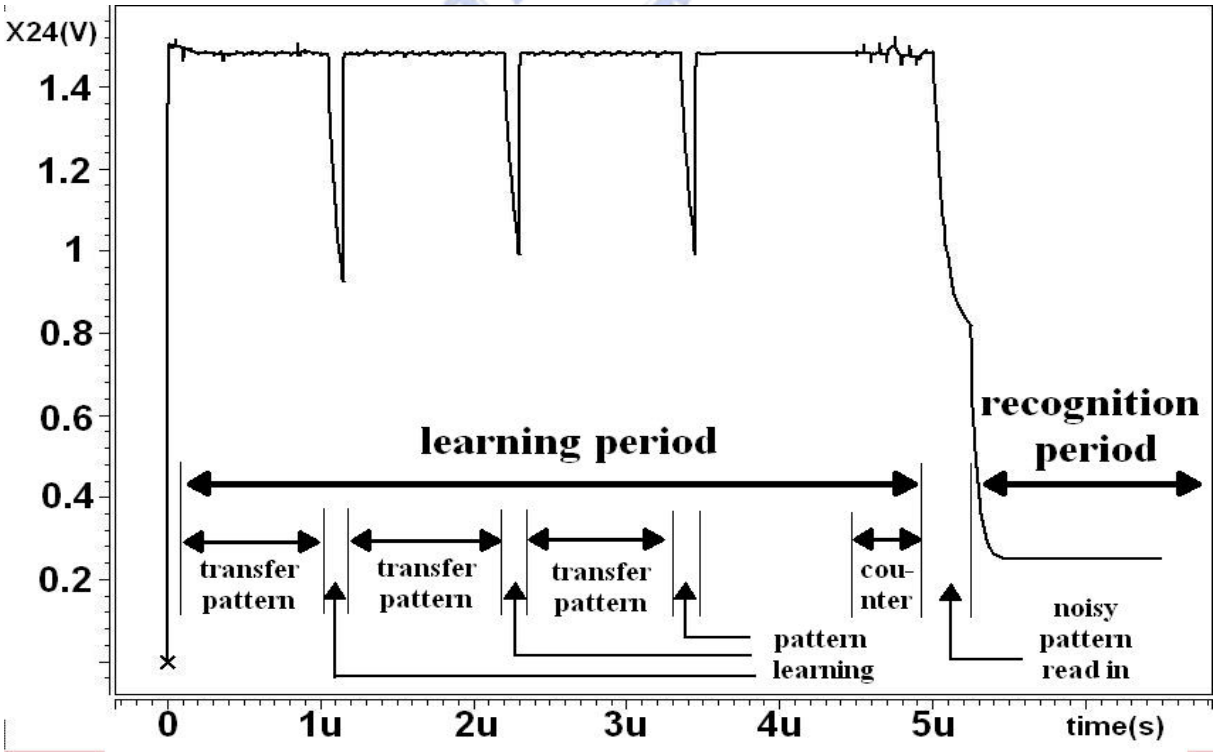

Fig. 3.10 Recognizing process of the white pixel without noise *P(2,4)* (Hspice)

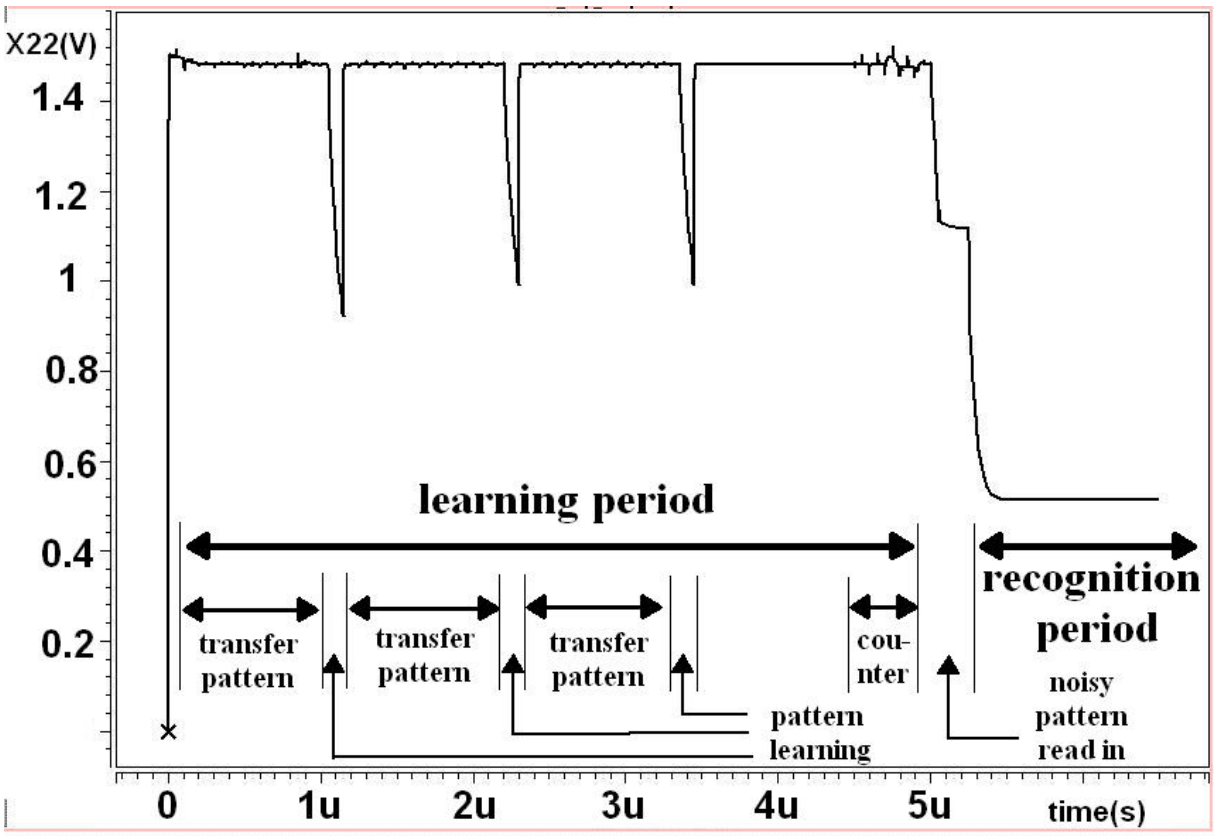

Fig. 3.11 Recognizing process of the white pixel with noise *P(2,2)* (Hspice)

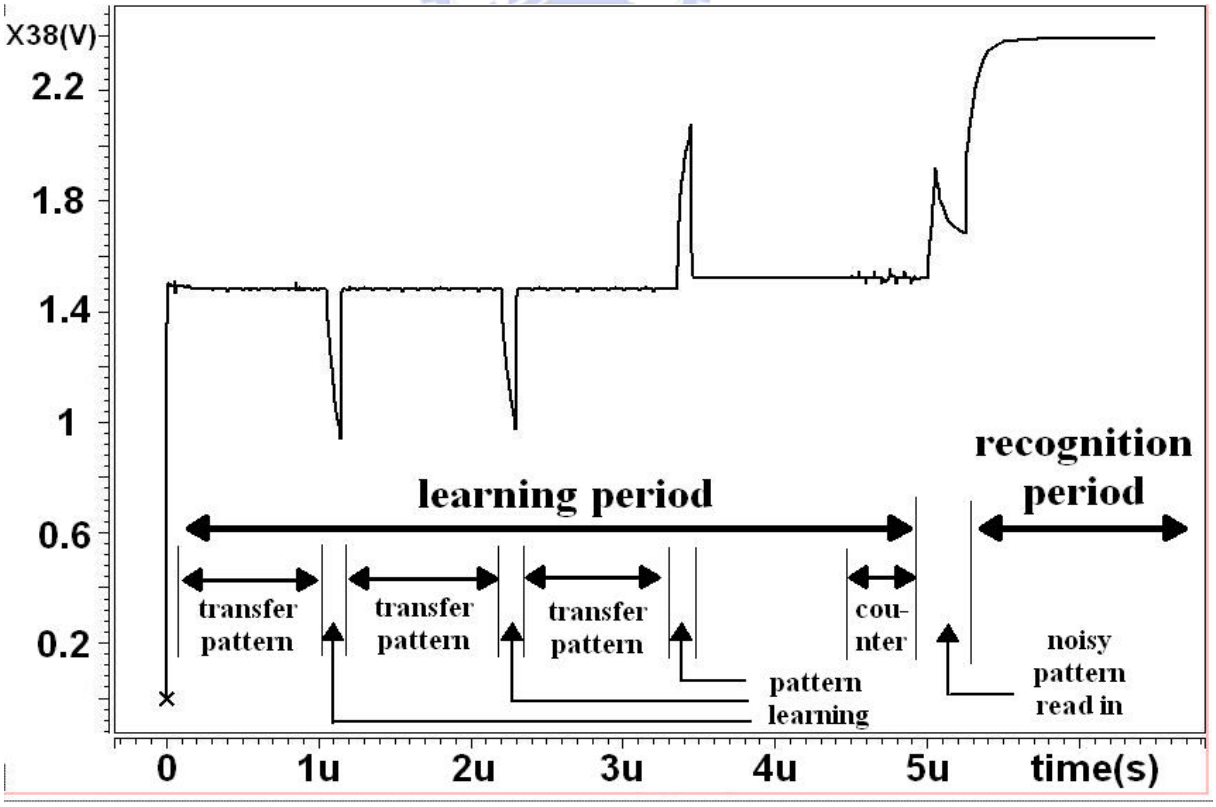

Fig. 3.12 Recognizing process of the black pixel without noise *P(3,8)* (Hspice)

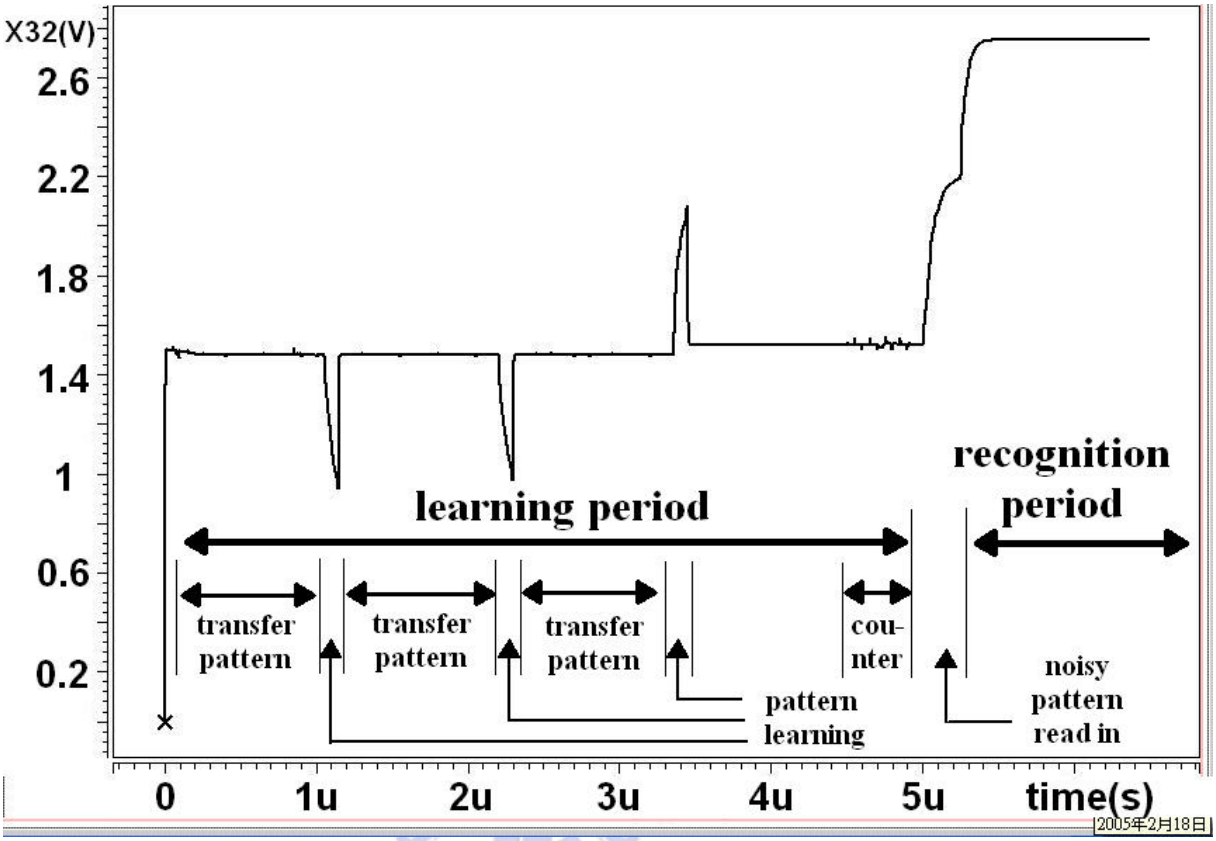

Fig. 3.13 Recognizing process of the black pixel with noise *P(3,2)* (Hspice)

CHAPTER 4

LAYOUT DESCRIPTIONS AND EXPERIMENTAL RESULTS

4.1 Layout and Experimental Environment Setup

Fig. 4.1 and Fig. 4.2 show the layout of the chip. Fig. 4.1 shows the layout of one cell and two ratio memories. The central part of Fig. 4.1 is cell, and the left side and right side of Fig. 4.1 are ratio memories. The area of one cell and two RM is 400x250 um^2. Fig. 4.2 shows the whole chip layout. In Fig. 4.2, the TSMC standard pads which include ESD device, pre-driver and post-driver are used. The die area is 4.56x3.49 mm^2. Fig. 4.3 is the package diagram, and the package is 84 pins LCC84. The die photo is shown as Fig. 4.4. Table 4.1 shows the summary of performance. That performance is compared with RMCNN with elapsed operation[18]. The RMCNN w/o EO is compared with the RMCNN with elapsed operation. The area per pixel of RMCNN w/o EO is smaller than the RMCNN with elapsed operation, but the whole chip area of RMCNN w/o EO is larger. Because the large TSMC standard pad is adapted in RMCNN w/o EO, the whole chip area is larger even if the area per pixel is smaller.

The environment of measurement is shown as Fig. 4.5. The controlling signals and some input signals are generated by the pattern generator of HP/Agilent 16702A. The clock in the pattern generator is 12.5MHz and the signal rising (falling) time is about 4.5ns. Output waveform is shown on the oscilloscope TDK 3054B. The power supply is 3V.

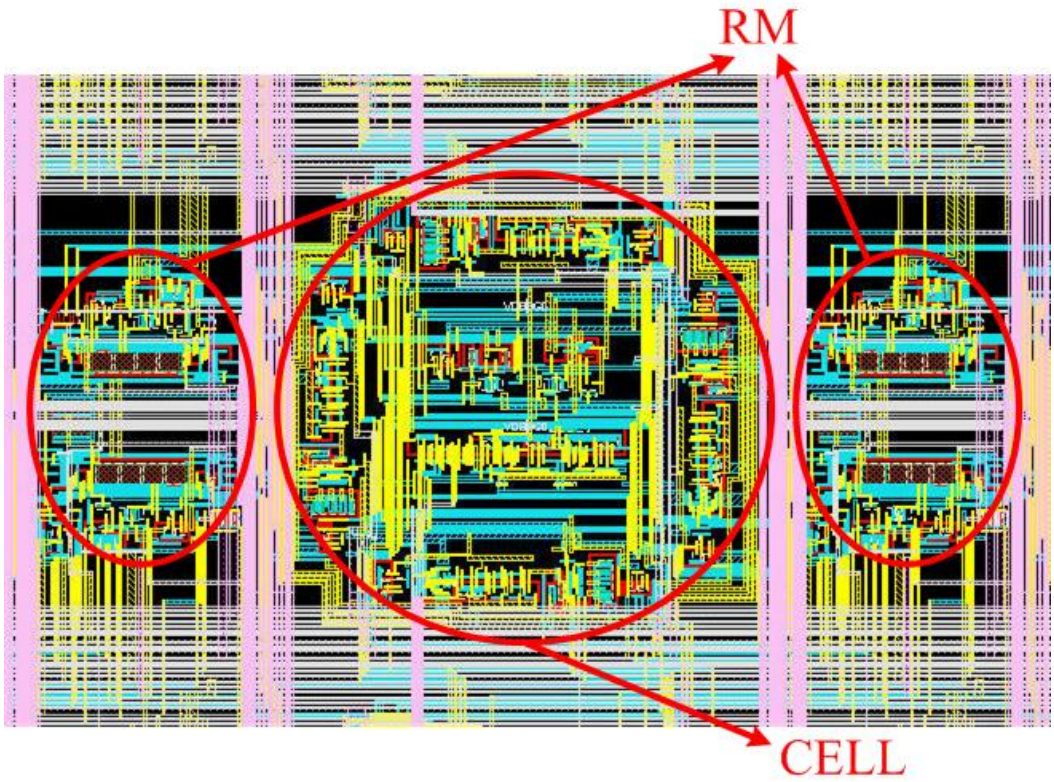

Fig. 4.1 Layout of one pixel (two RM and one cell)

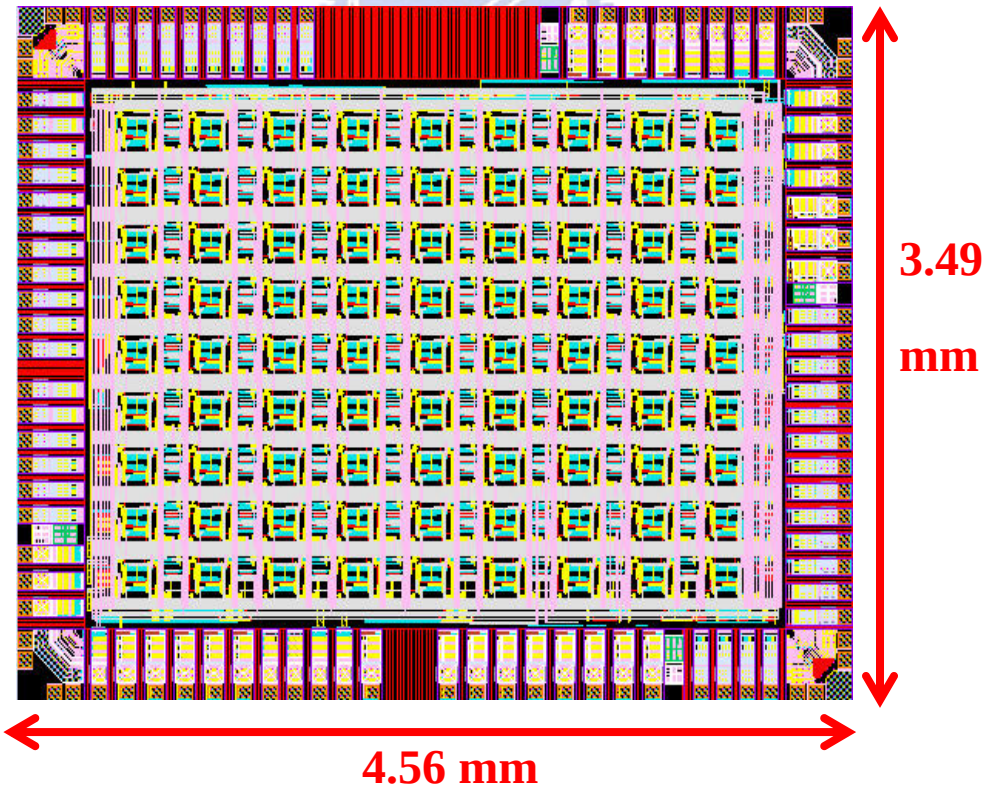

Fig. 4.2 Layout of the whole chip (pad included)

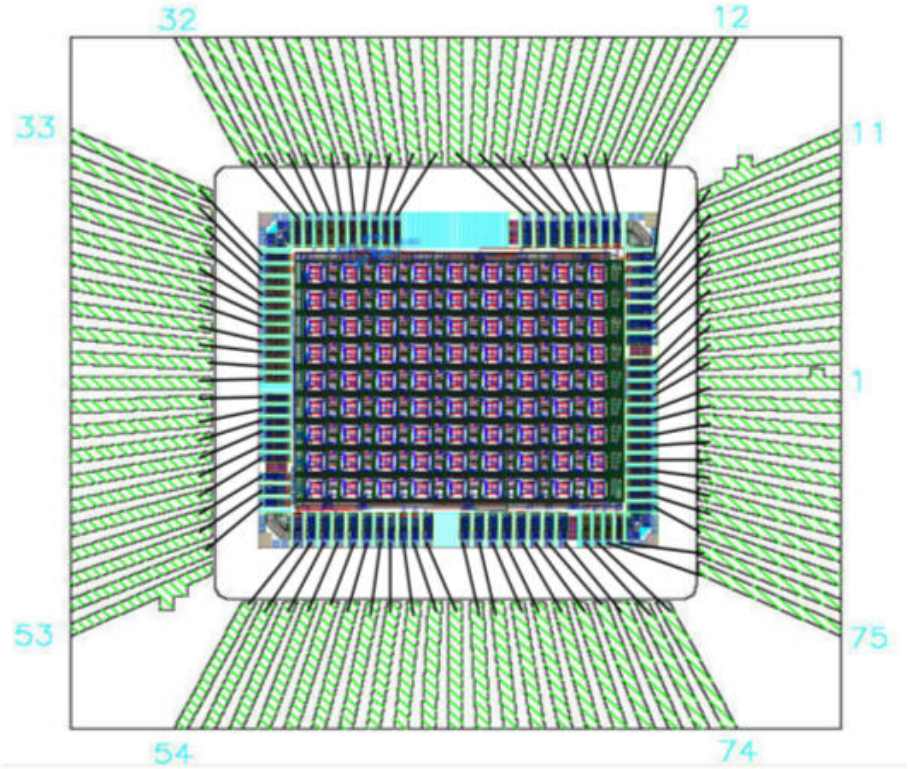

Fig. 4.3 The package diagram

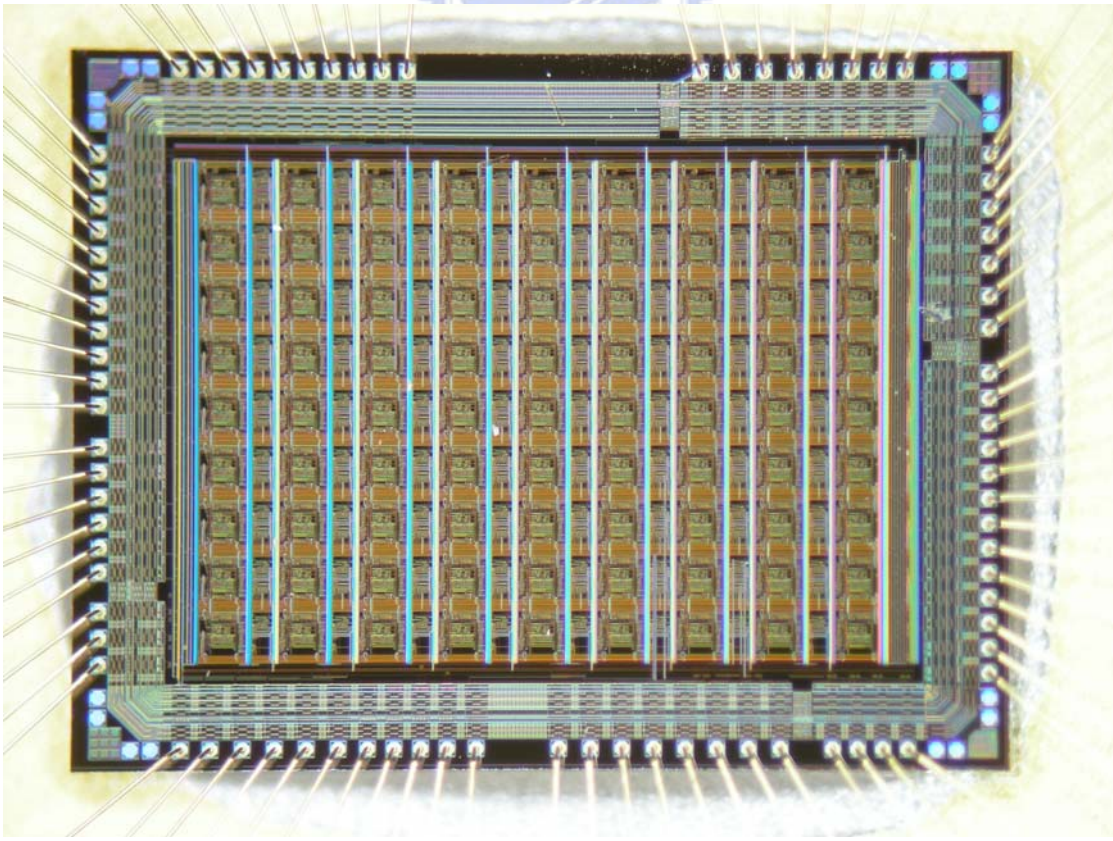

Fig. 4.4 The die photo of 9x9 RMCNN without elapsed period

Table 4.1 the summary of the RMCNN w/o EO compared with RMCNN with elapsed operation

	RMCNN with EO	RMCNN w/o EO
Technology	0.35 μm 1P4M Mixed-Signal Process	0.35um 2P4M Mixed-Signal Process
Resolution	9 x 9 Cells	9x9 Cells
No. of RM blocks	144 RMs	144 RMs
1 Pixels	1 cell + 2 RMs	1 cell + 2 RMs
Single pixel area	350 μm x 350 μm	400 um x 250um
CNN array size (include pads)	3800 μm x 3900 μm	4560 um x 3900 um
Power supply	3 V	3V
Total quiescent power dissipation	120 mW	87mW
Minimum readout time of a pixel	1 μs	100ns
Elapsed operation	Require	Not require

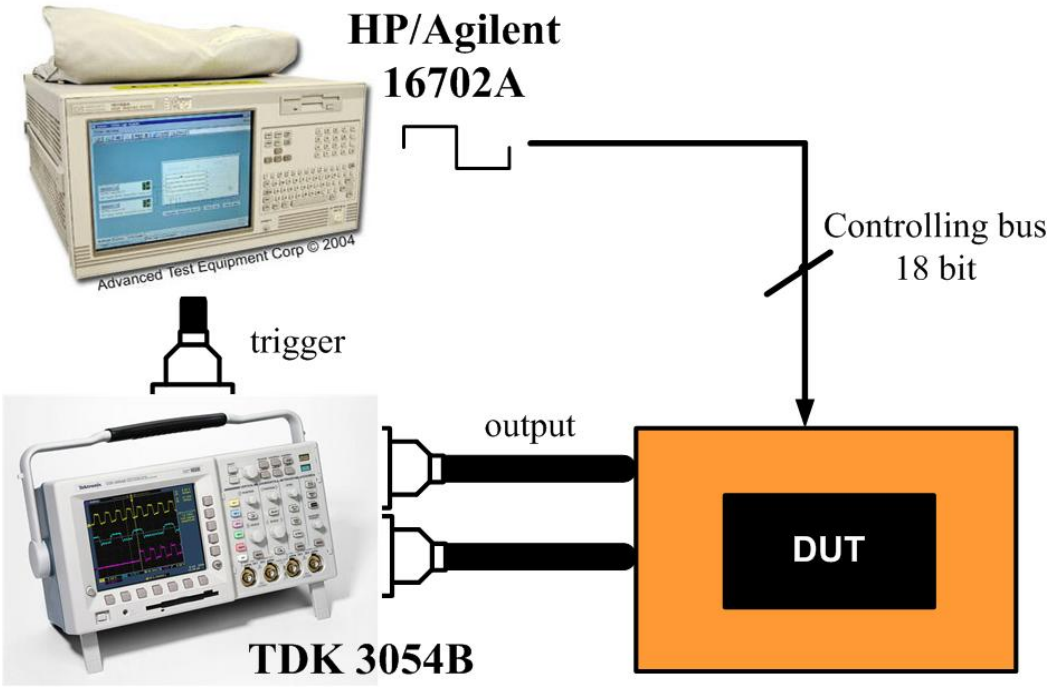

Fig. 4.5 The environment of measurement

This circuit is controlled by many controlling signals. Fig. 4.6 shows the timing relationship of these controlling signals. The circuit figures in chapter 2 explained how these controlling signals control the circuit. The signals *clk1* and *clk2* determine the architecture of the circuit. If *clk1* is high, the architecture of the circuit is learning architecture which is shown as Fig. 2.4. If *clk2* is high, the architecture of the circuit is recognition architecture which is shown as Fig. 2.6. Thus the signals *clk1* and *clk2* can't be high at the same time. Otherwise the circuit can't operate correctly.

In Fig. 4.6, the learning period is marked in the timing that *clk1* is high. Similarly, recognition period is marked in the timing that *clk2* is high. Signal *R* is used to reset the output of some sub-circuits in the circuit. The *DFF* is used to drive the negative edge trigger D-flip-flop in Fig. 2.22. The signals *newp* and *pin* appear in Fig. 2.22. When the *newp* is low, the connection between shift registers is cut off. Then the data in shift registers won't be changed by the glitch on signal *DFF*. When *newp* is high, the shift registers can transfer the learning patterns. Thus the signal *DFF* oscillates only when *newp* is high. Signal *pin* let the pattern stored in shift register input into cells. After learning period, the ratio weights are generated in the timing "Ratio weight generating". In this timing, the signals *Cou_L* and *Cou_G* which appear in Fig. 2.18 and 2.19 oscillate four times to change the output of **Counter_L** and **Counter_G** from "00" to "11" sequentially. Then the paths of Sw_a~Sw_f and S_en1~S_en6 turn on one by one and the ratio weight will be generated. After the timing "Ratio weight generating", the signals *noi* and *pin* which appear in Fig. 2.23 become high to input the noisy pattern into cells. Then the circuit starts recognition period to recover the noisy pattern. Table 4.1 shows the function and usage of the all controlling signals.

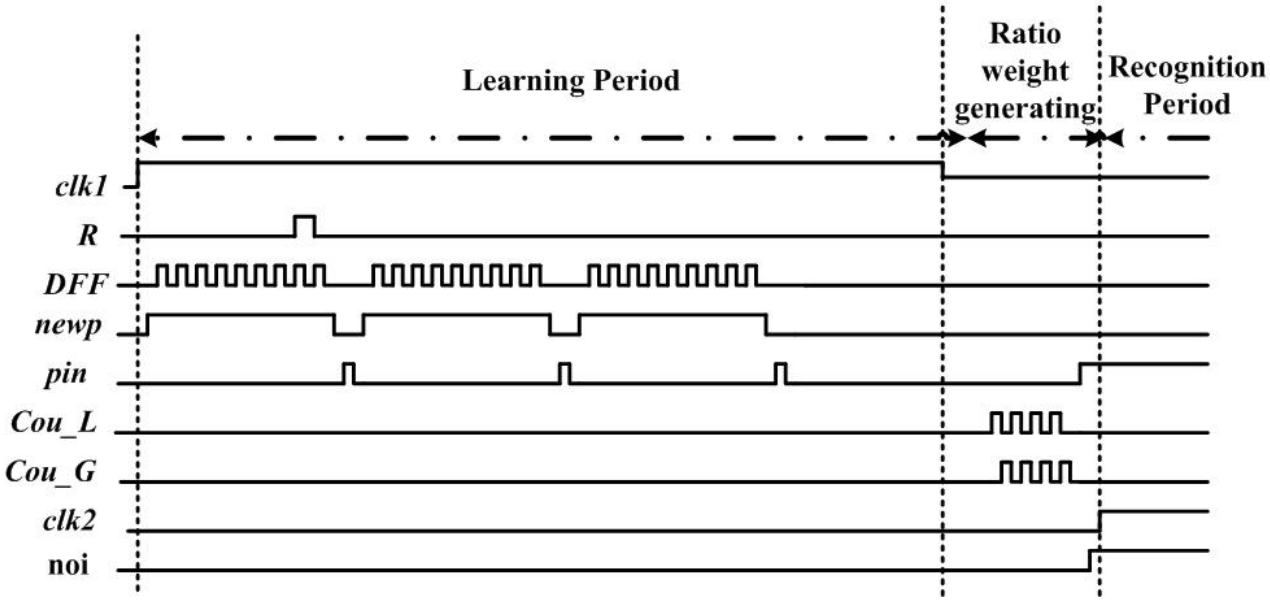

Fig. 4.6 The control-timing diagram in the measurement of the 9x9 RMCNN with r = 1.

Table 4.1 The function of every controlling signal

Control signal	Usage
clk1	High：learning period starts Low：learning period stops
R	High：reset the circuit Low：don't reset
DFF	Drive the shift registers (negative trigger D-flip-flop) used to store the learning patterns.
newp	High：the shift register can transfer the learning patterns Low：the shift register can't transfer the learning patterns
pin	High：the pattern stored in shift registers input to the cells. Low：the path between shift registers and cells is cut off
Cou_L	Drive every local counter in every cell
Cou_G	Drive the global counter
clk2	High：recognition period start Low：recognition period stop
noi	High：the pattern in shift registers becomes noisy Low：isolate the noise and innocent pattern in shift register

4.2 Experimental Result

The output stage is described in chapter 2 and Fig. 2.20. Only one pad is used to output the state of every cell. Thus the 81 pixels are read out sequentially.

Before pattern recognition, the learning function is checked first. That verification of learning function checks if the learning patterns are sent into the shift register exactly and the patterns stored in shift registers input to every cell correctly. The pattern is read out directly after the pattern is inputted into the circuit. Fig. 4.7~Fig. 4.9 is the verified result of learning function. Fig. 4.7 shows the learning pattern ”一” in the shift registers. Fig. 4.8 shows the learning pattern ”二” and Fig. 4.9 shows the learning pattern ”四” in the shift registers. In Fig. 4.7~Fig. 4.9, “Ch 2” is the output data of the chip and “Ch 3” is the LSB of the decoder which controls the switches Sw_{c11}~Sw_{c99} in Fig. 2.20. “Ch 1” is a trigger signal, and it is meaningless in this measurement. Each row is read out sequentially. The first row is read out first, and then the second row is following. Each row is marked in Fig. 4.7~Fig. 4.9. The output waveform of “Ch 2” in Fig. 4.7~Fig.4.9 can be cut off and recombined to form a new pattern that is more easily discerned. Fig. 4.10~Fig. 4.12 show these recombined output waveform. Left sides of Fig. 4.10~Fig. 4.12 is the pattern that supposed to be learned, and right side is the recombined output waveform. In Fig. 4.7~Fig. 4.12, the output of black pixel is about 1.5V, and the output of white pixel is about 0.2V.

It is obvious that all of the learning patterns are inputted exactly into the circuit, and the shift register indeed work well. But the measurement of recognition function isn’t so successful. Fig. 4.13 is the recognition result of pattern “四” without noise, and Fig. 4.14 shows the recombined output waveform of Fig. 4.13. It is obvious that some pixels in row 4 and row5 are not pulled up enough. That means these pixels are not recover to pure black of pure white color. The colors of these pixels are just gray. Though the recognition of innocent pattern “四” isn’t successful, however the recognition result of patterns “一” and “二” without noise are very successful. Fig. 4.15 and Fig. 4.16 are the measurement result of recognition of

patterns “一” and “二”.

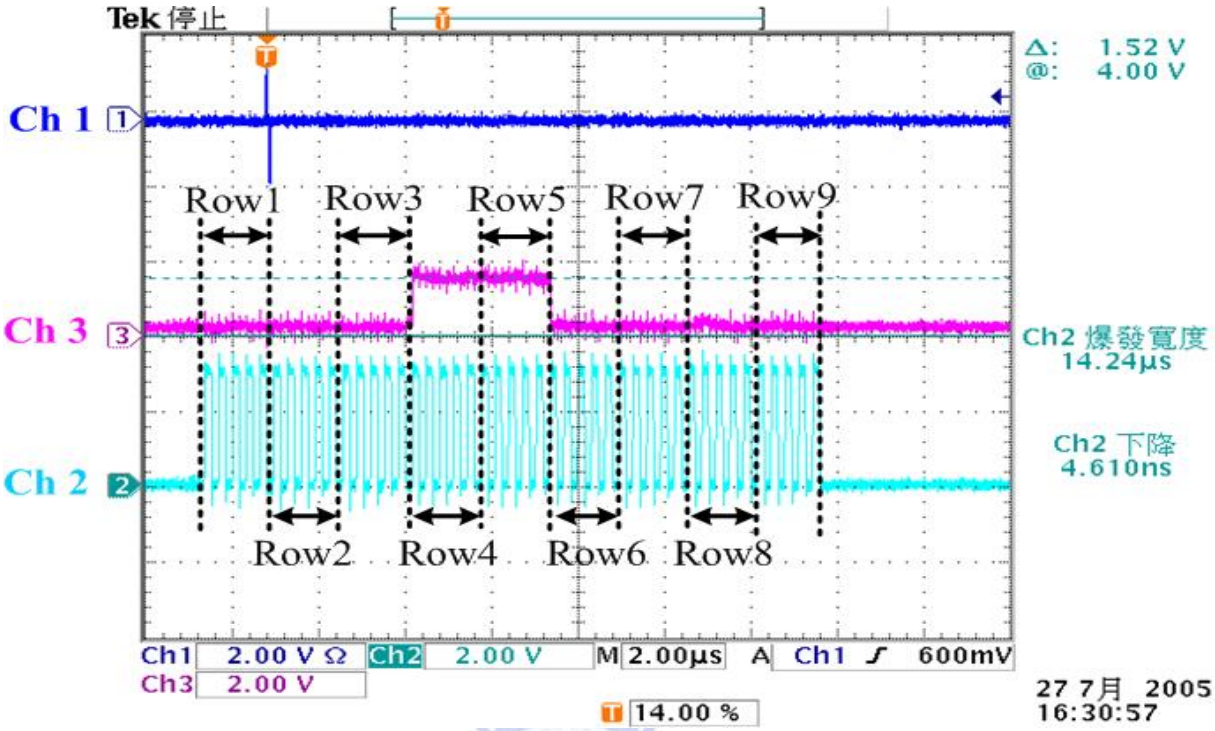

Fig. 4.7 Experimental verification of learning function (“一”)

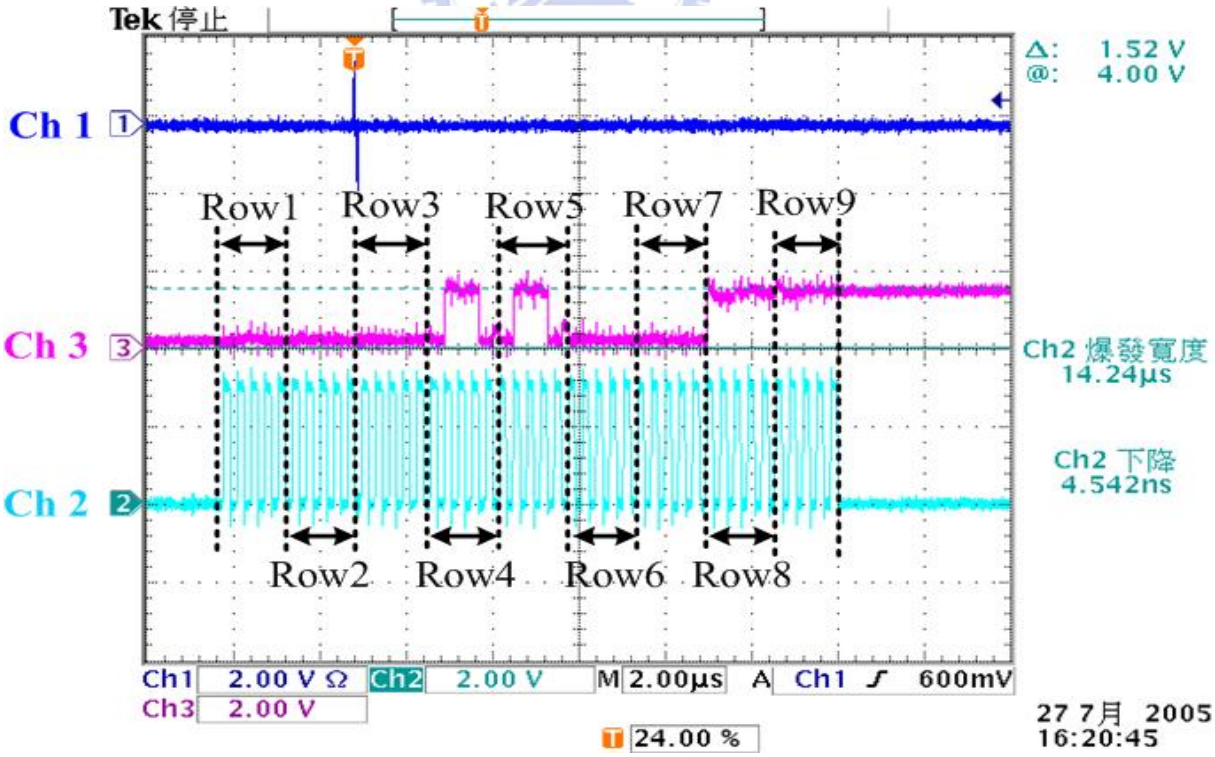

Fig. 4.8 Experimental verification of learning function (“二”)

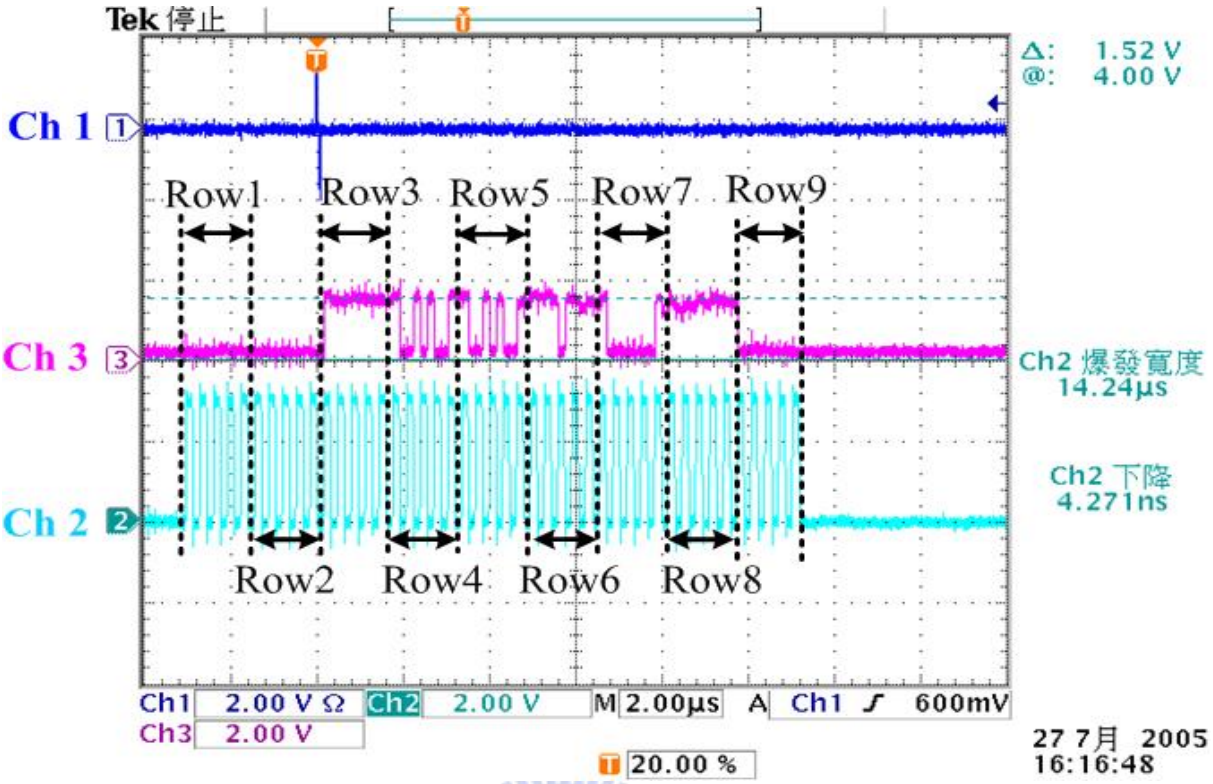

Fig. 4.9 Experimental verification of learning function ("四")

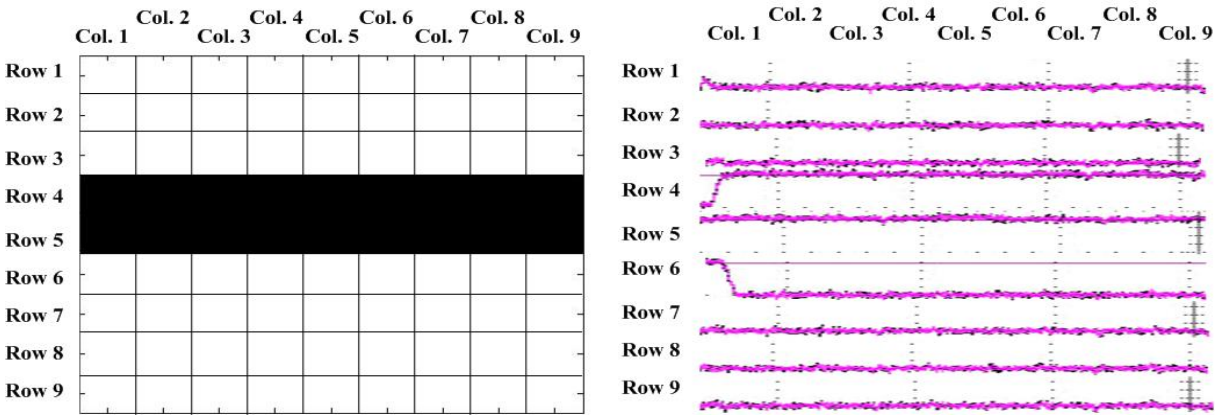

Fig. 4.10 The recombined waveform of the verification of learning function ("一")

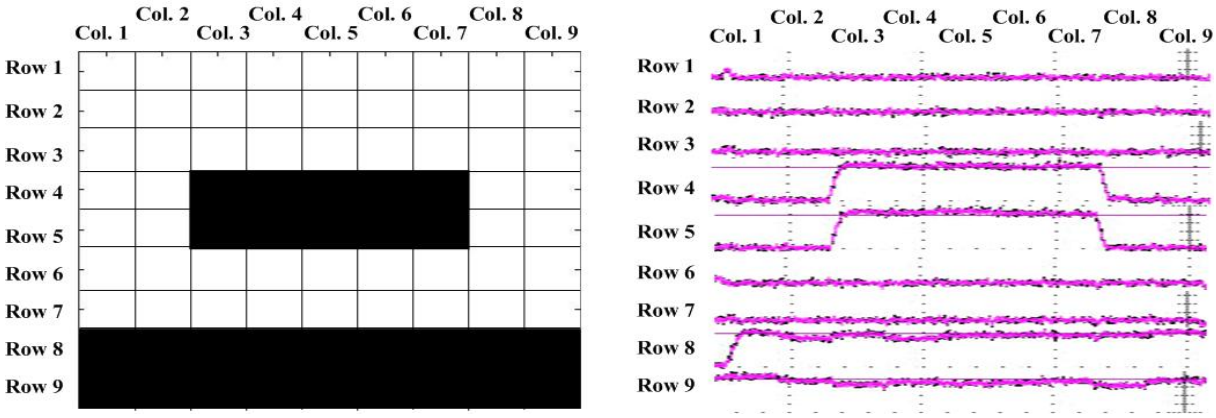

Fig. 4.11 The recombined waveform of the verification of learning function ("二")

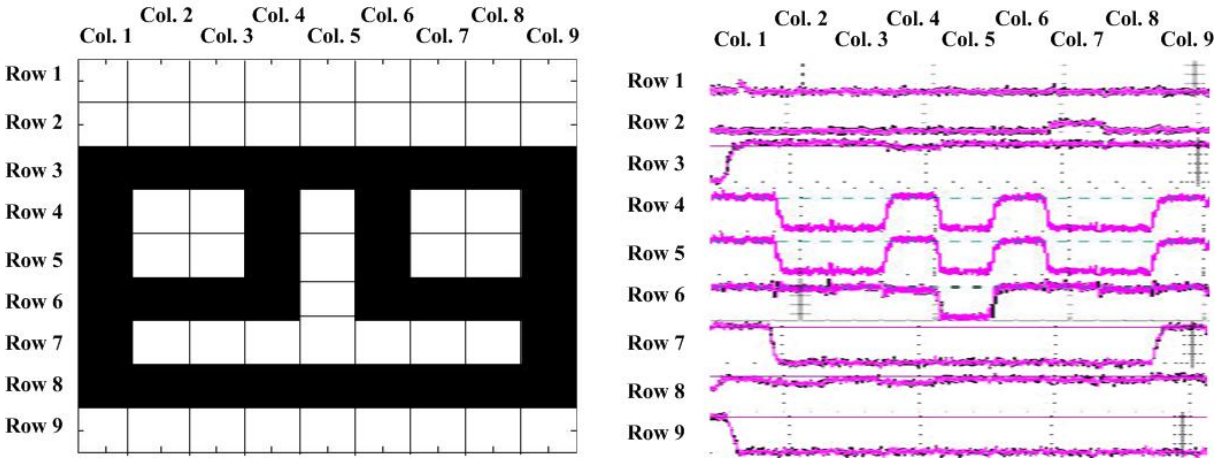

Fig. 4.12 The recombined waveform of the verification of learning function (“四”)

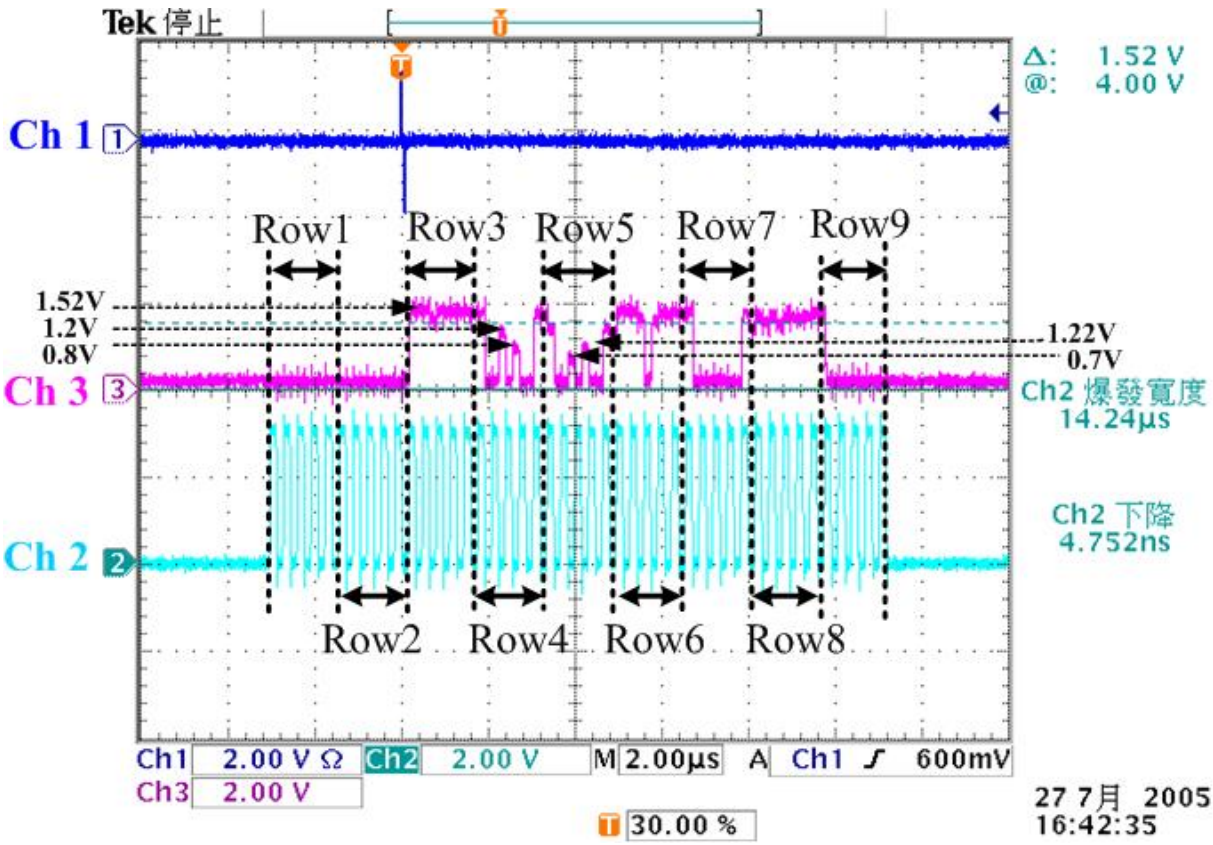

Fig. 4.13 Experimental recognizing result of the clear pattern “四”

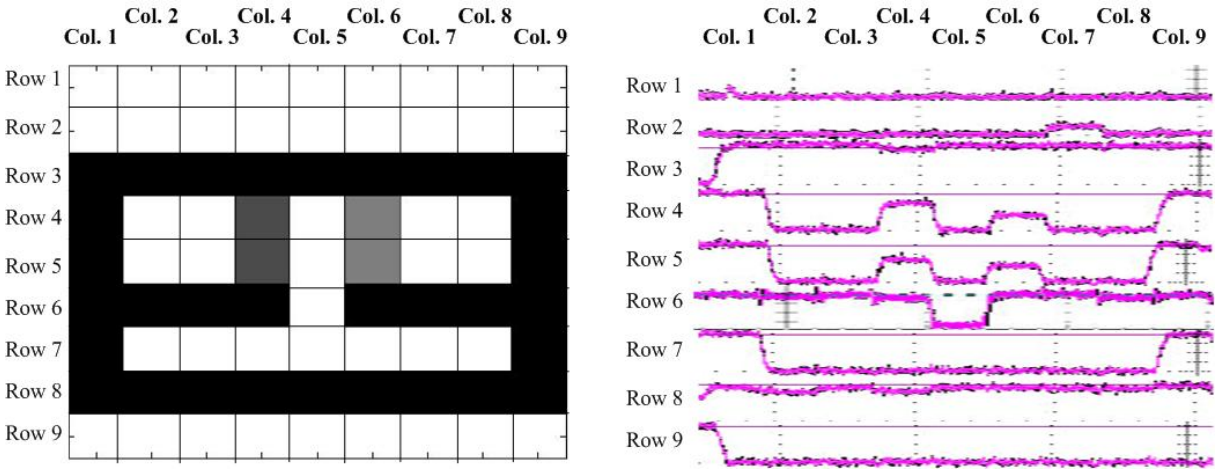

Fig. 4.14 The recombined waveform of the experimental recognizing result of the clear pattern “四”

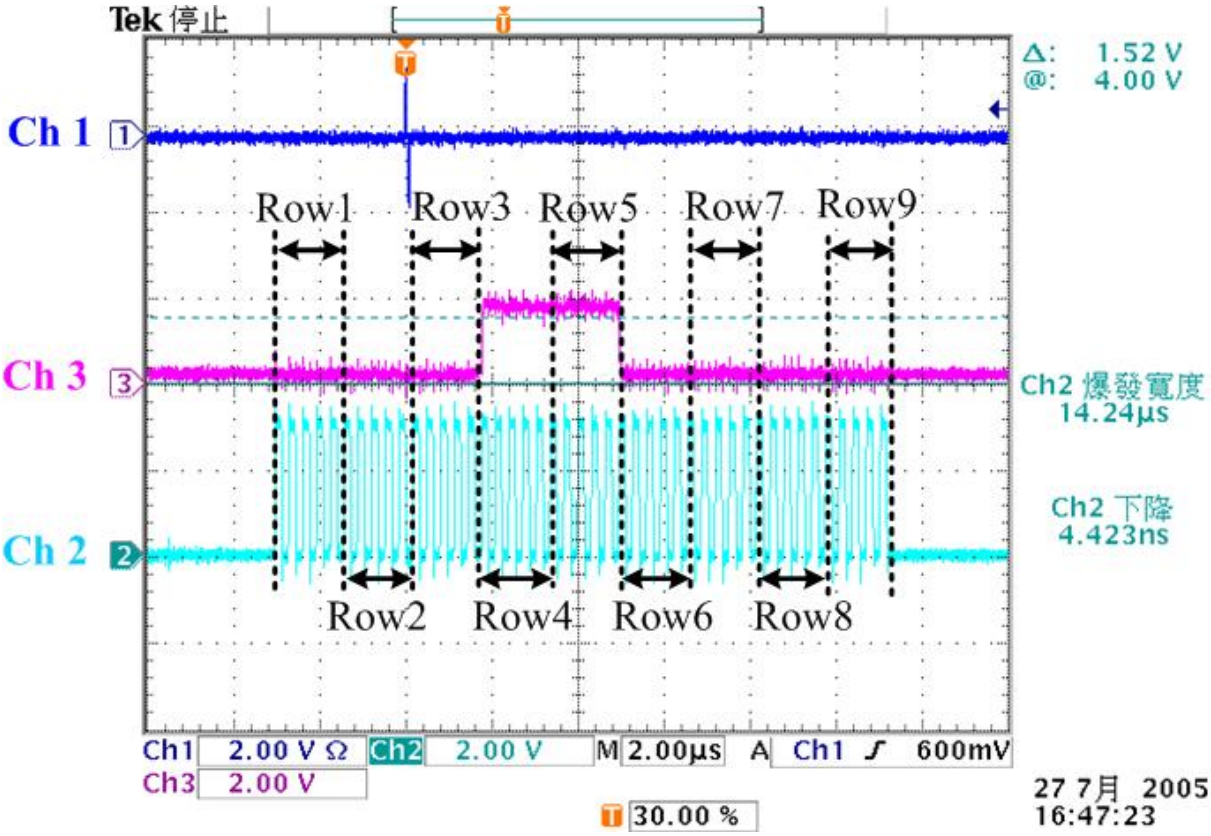

Fig. 4.15 Experimental recognizing result of the clear pattern “一”

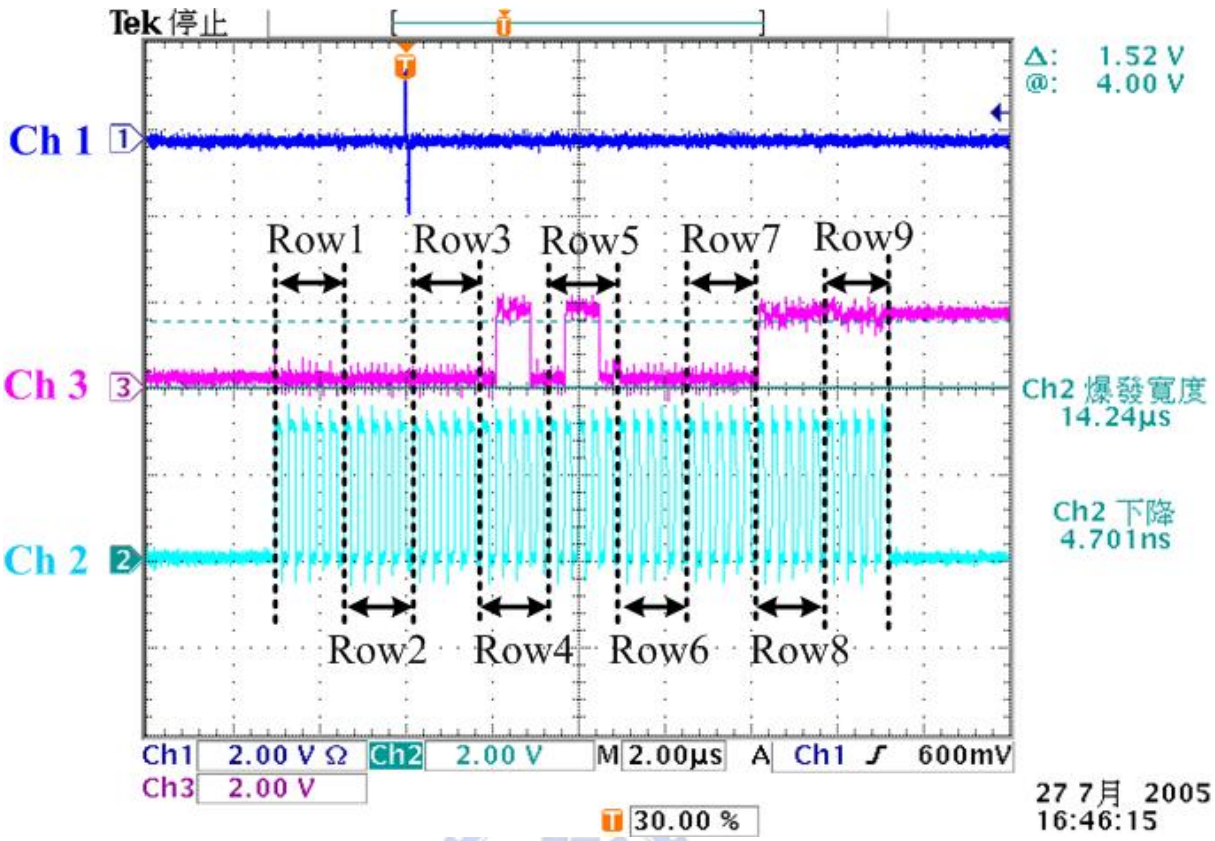

Fig. 4.16 Experimental recognizing result of the clear pattern "二"

The recognition result of noisy pattern with noise level 0.5 is shown as Fig 4.17 and Fig 4.18. Fig. 4.17 is the recognition result of pattern "一", and Fig. 4.18 is the recognition result of pattern "二". Both the two noisy pattern is unrecognized. The noisy pattern with noise level 0.5 is unrecognized in simulation result too.

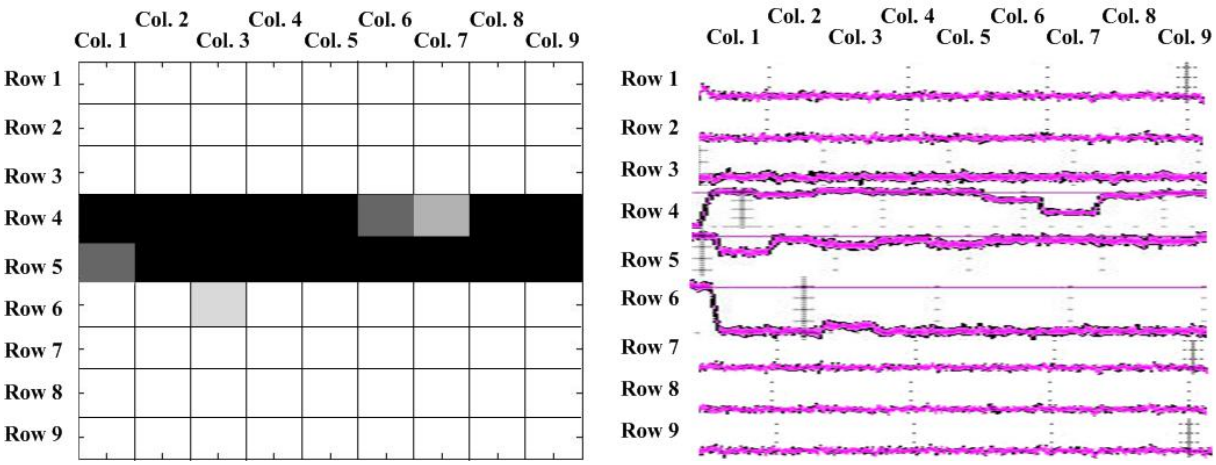

Fig. 4.17 Experimental recognizing result of the noisy pattern "一" with noise level 0.5

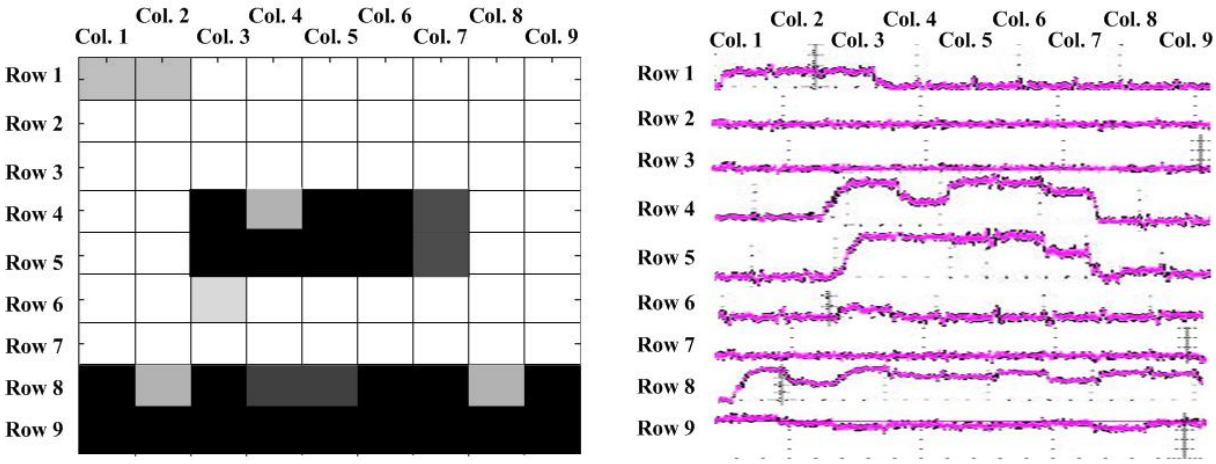

Fig. 4.18 Experimental recognizing result of the noisy pattern "二" with noise level 0.5

4.3 Cause of the Imperfect Experimental Result

The cause of the unsuccessful recognition is found in this thesis. Table 4.2 shows the absolute-weight of cell(4,4) which is recognized unsuccessfully. Three simulation conditions are in Table 4.2. The absolute-weight ss_{44}^{M} is simulated by Matlab, and that is a ideal weight. The absolute-weight ss_{44}^{TT} is simulated by Hspice in typical-typical corner condition. The absolute-weight ss_{44}^{FS} is simulated by Hspice in fast-slow corner condition. The absolute-weights ss_{44}^{TT} and ss_{44}^{FS} are strange. The absolute-weights in practical circuit is stored on the capacitor Cw in Fig. 2.4. The Hspice simulation result shows the charging and discharging currents are unbalanced. It is described in chapter 2 that the ratio weights are generated according to the absolute mean of absolute-weight. Table 4.3 shows the generated ratio weights according to the absolute-weights in Table 4.2. Because of the wrong absolute-weights ss_{44}^{TT} and ss_{44}^{FS}, the absolute means of the two absolute-weights are wrong too. Though the mean of ss_{44}^{TT} is wrong, there is still only one weight that is larger than the mean of ss_{44}^{TT}. Thus the ratio weight of ss_{44}^{TT} is the same with the ratio weight of ss_{44}^{M}, and these two ratio weights are correct. But the mean of ss_{44}^{FS} is too wrong to get a correct ratio

weights. There are three weights in ss_{44}^{FS} larger than the mean of ss_{44}^{FS}, so the generated ratio weight of ss_{44}^{FS} is completely wrong. The wrong ratio weight results the wrong recognition. The cause of the wrong absolute-weights is shown as blow.

Chapter 2 explained the learning structure and the all detailed sub-circuits. Fig. 4.19 is the learning structure. The block **W** charges or discharges the capacitor Cw according to the input of two neighboring cells, and the charging current direction is controlled by the XOR gate in Fig. 4.20. Fig. 4.20 is a part of Fig. 2.10. The two inputs of XOR gate are the signs of two neighboring cells. Fig. 4.21 shows that one of the two inputs of XOR is connected to the *Vin* of **T2**.

When a pattern is learned, the shift registers need to transfer the new pattern. The pattern transferring takes a little time, and the MOS M26 in Fig. 2.8 is turned on in this timing. The MOS M26 in Fig. 2.8 is turned on and let the current I_charge in Fig. 4.19 become very small. However, this small current still influences the absolute-weights on Cw, and Fig 4.22 shows the small current in the pattern transferring time. Note that there is small current in the pattern transferring time. Because M26 in Fig. 2.8 is turned on in the pattern transferring time, one input of the XOR gate would be Vref(1.5V). Because one input of the XOR gate is connected with 1.5V, the output of XOR is unpredictable. Thus the influence of the small current in the pattern transferring timing is out of control, and the absolute-weights are affected by the small current.

The modified circuit is shown as Fig. 4.23. A new path connected with a dummy load is inserted. The path turns on when patterns is transferring, and then the small current in the pattern transferring timing doesn’t influence the absolute-weights. Fig 4.24 is the simulation result of modified **T2D**, and it shows the modified design of **T2D** doesn’t contribute a small current to Cw. One pixel model with modified **T2D** is simulated too, and the modified design can indeed recognize the noisy pixel.

Table 4.2 The absolute weight of cell(4,4) in three simulation condition

Simulation condition	Absolute-weight of cell(4,4)
Matlab (ideal)	$ss_{44}^{M} = \begin{bmatrix} 0 & -0.33 & 0 \\ 0.33 & 0 & 0.33 \\ 0 & 1 & 0 \end{bmatrix}$
Hspice (TT)	$ss_{44}^{TT} = \begin{bmatrix} 0 & -0.15 & 0 \\ 0.28 & 0 & 0.28 \\ 0 & 0.78 & 0 \end{bmatrix}$
Hspice (FS)	$ss_{44}^{FS} = \begin{bmatrix} 0 & 0.116 & 0 \\ 0.41 & 0 & 0.41 \\ 0 & 0.7 & 0 \end{bmatrix}$

Table 4.3 The absolute mean and generated ratio weights of cell(4,4) in three simulation condition

Simulation condition	Absolute-weight of cell(4,4)	Mean	Ratio weights of cell(4,4)
Matlab	$ss_{44}^{M} = \begin{bmatrix} 0 & -0.33 & 0 \\ 0.33 & 0 & 0.33 \\ 0 & 1 & 0 \end{bmatrix}$	0.5	$A_{44} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$
Hspice (TT)	$ss_{44}^{TT} = \begin{bmatrix} 0 & -0.15 & 0 \\ 0.28 & 0 & 0.28 \\ 0 & 0.78 & 0 \end{bmatrix}$	0.3725	$A_{44} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$
Hspice (FS)	$ss_{44}^{FS} = \begin{bmatrix} 0 & 0.116 & 0 \\ 0.41 & 0 & 0.41 \\ 0 & 0.7 & 0 \end{bmatrix}$	0.409	$A_{44} = \begin{bmatrix} 0 & 0 & 0 \\ 0.33 & 0 & 0.33 \\ 0 & 0.33 & 0 \end{bmatrix}$

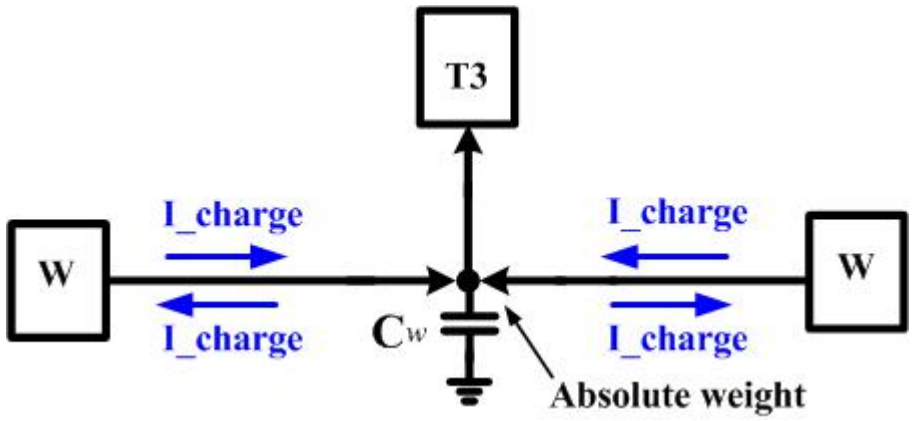

Fig. 4.19 The absolute-weights learning structure

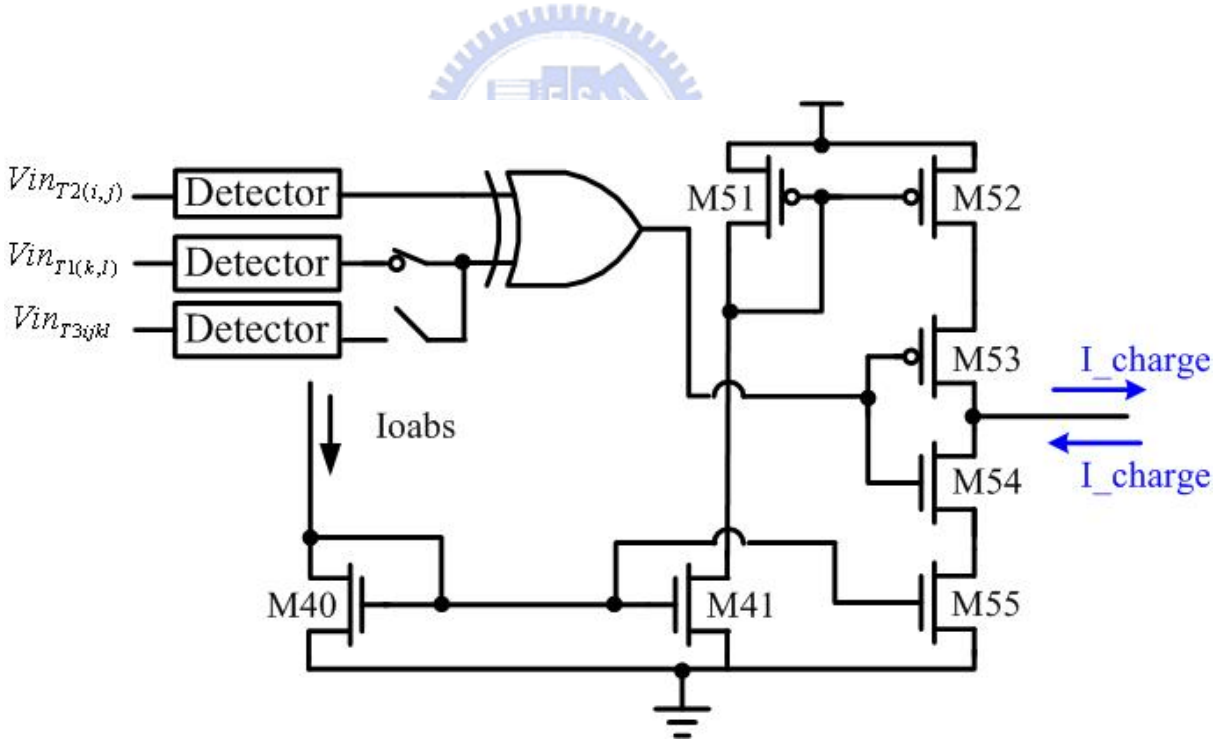

Fig. 4.20 The structure that controls flowing direction of I_charge

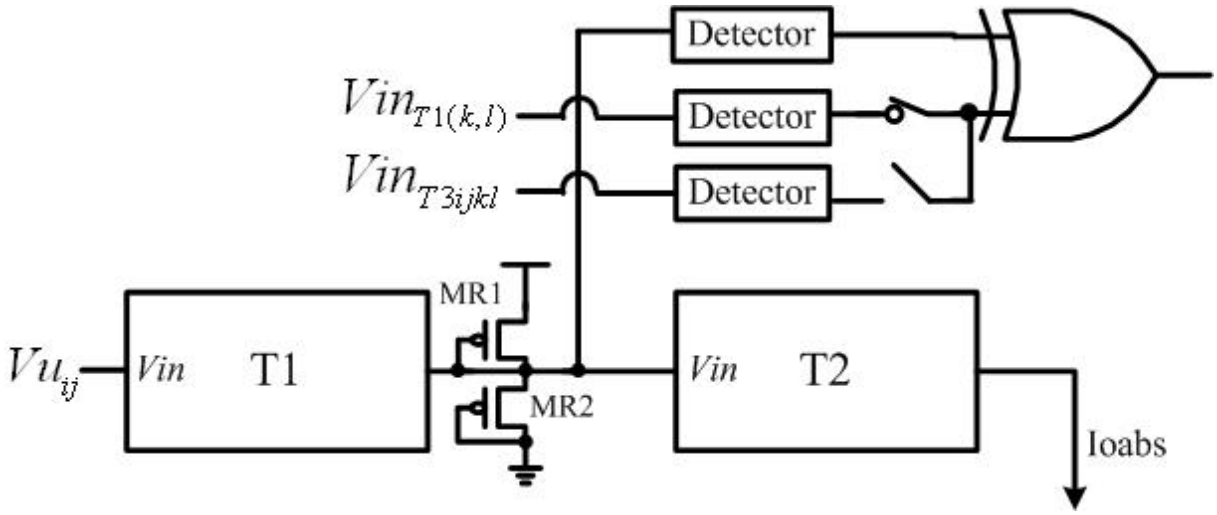

Fig. 4.21 The connection between T2 and input of XOR gate

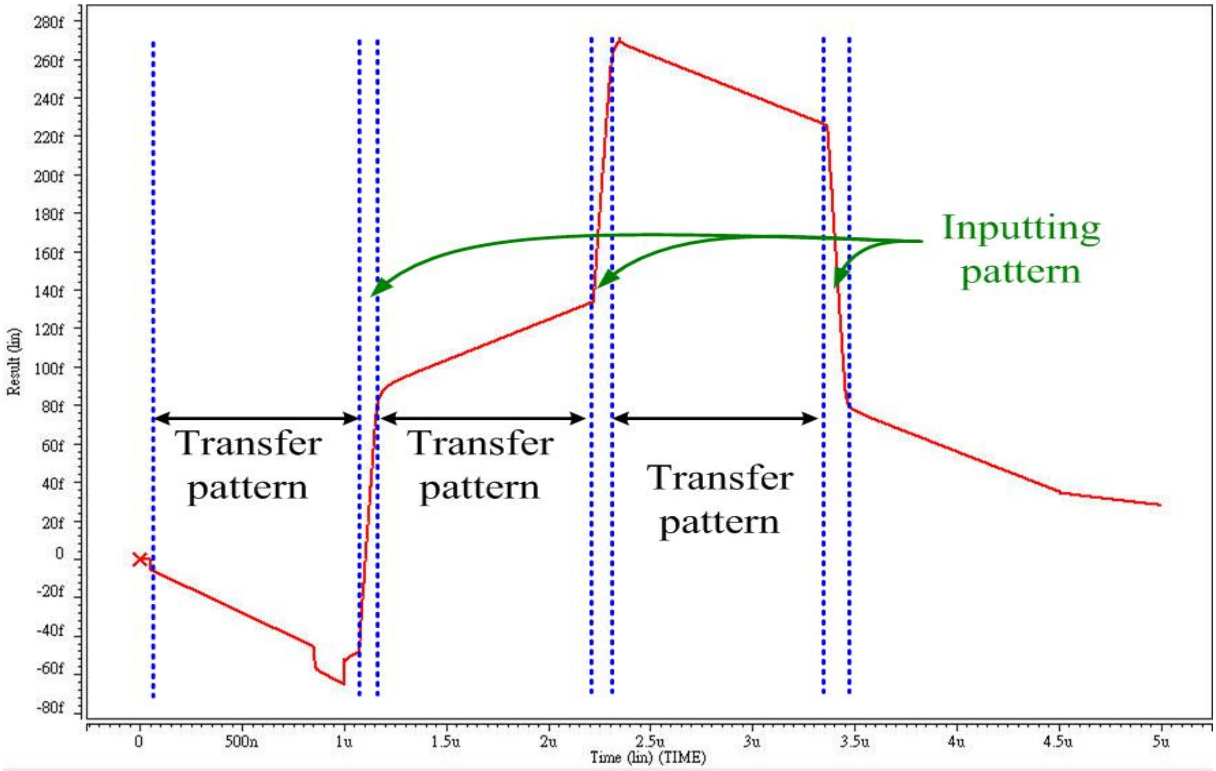

Fig. 4.22 The integration of T2D output current and time

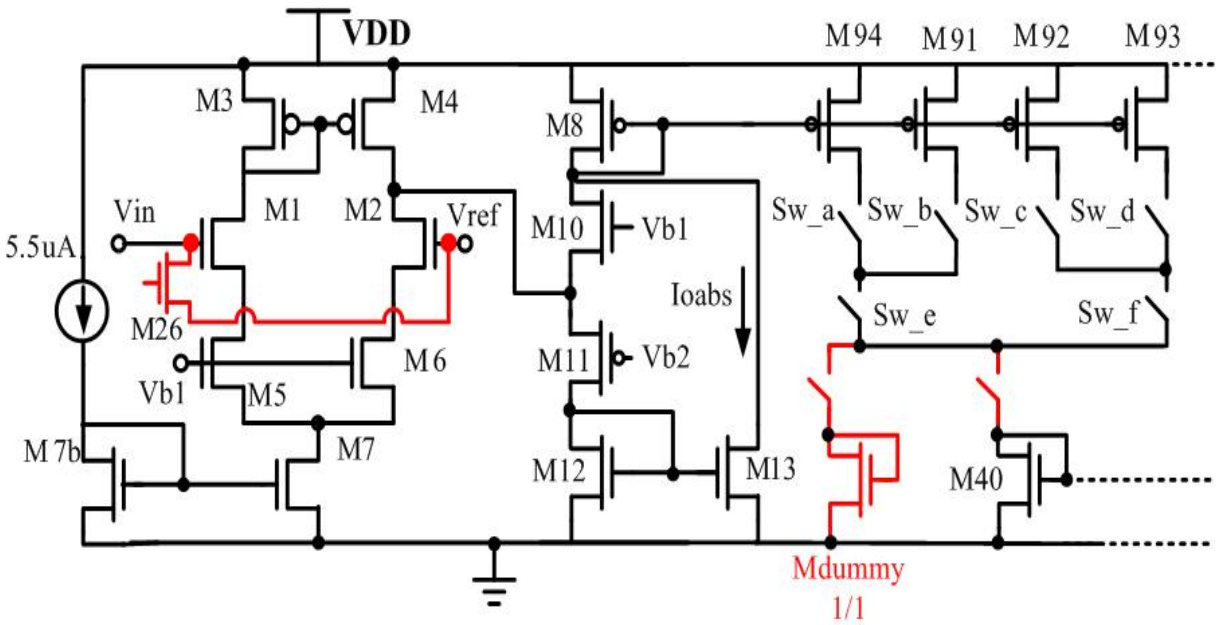

Fig. 4.23 The modified circuit

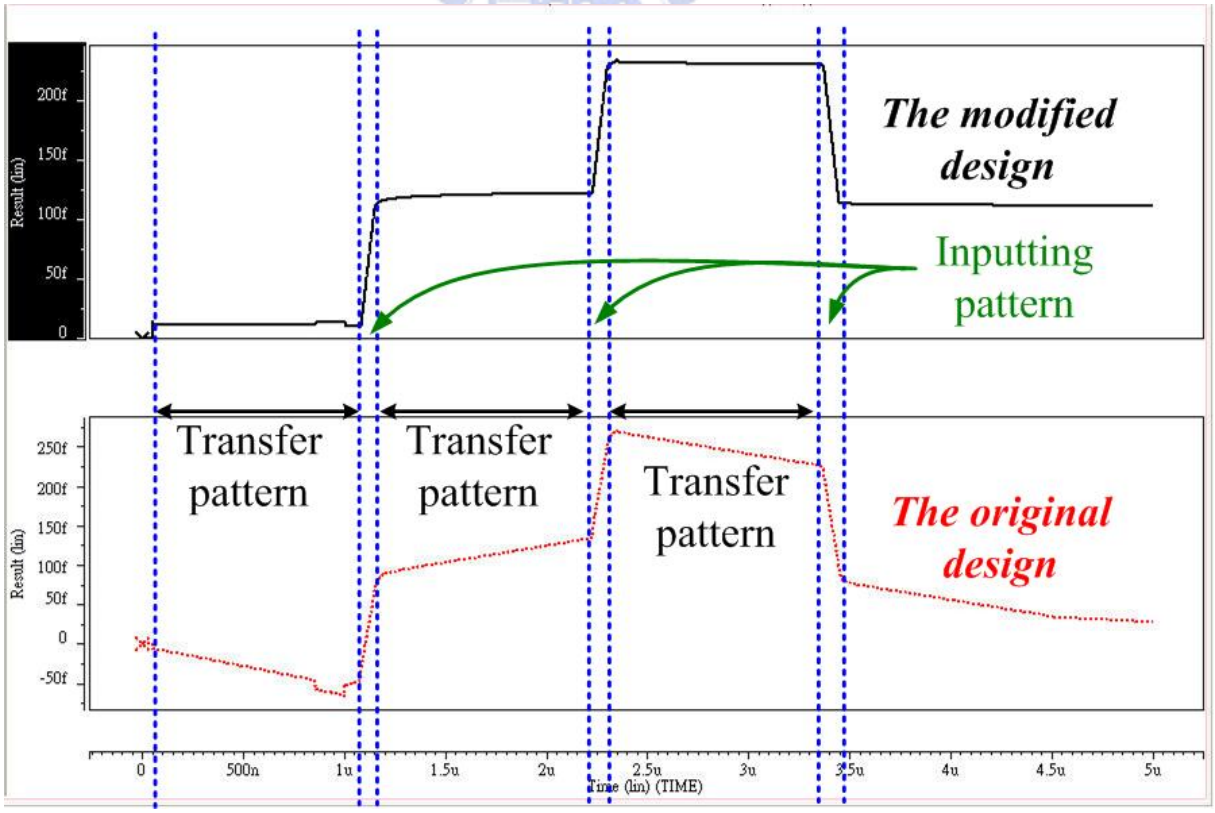

Fig. 4.24 The integration of T2D output current and time 1) the modified design 2) the original design

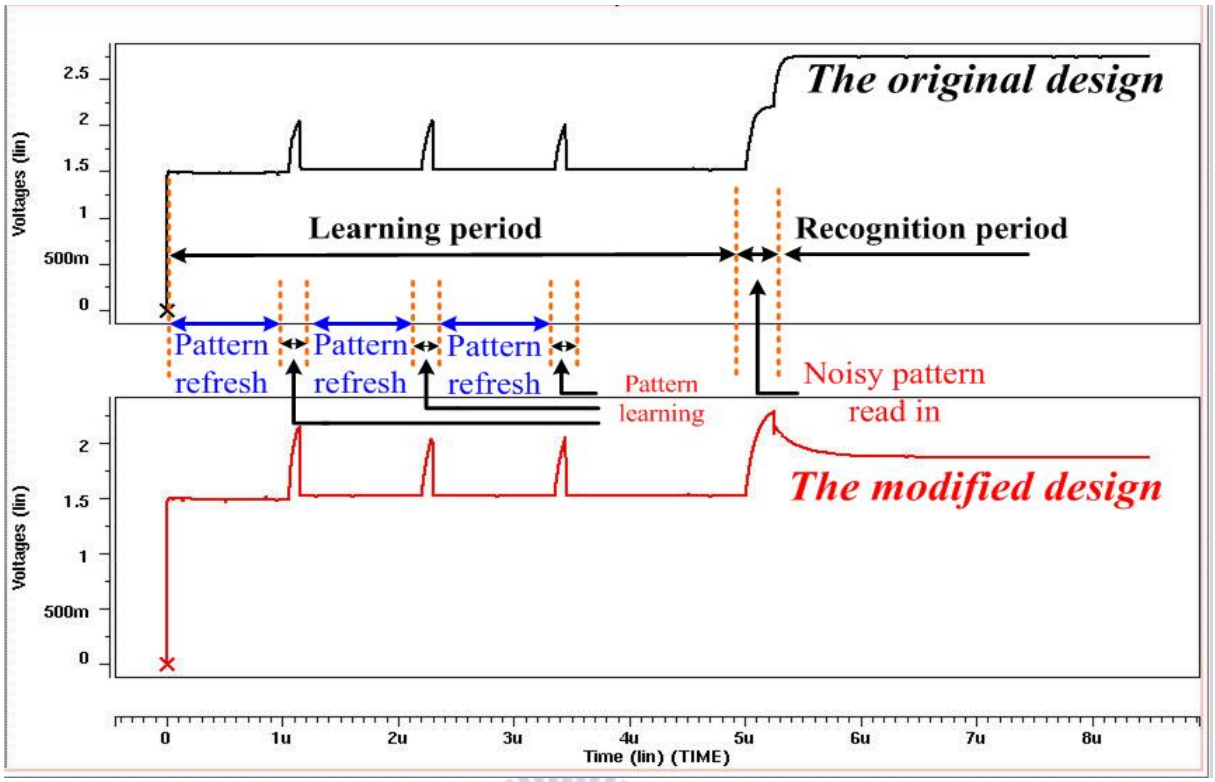

Fig. 4.25 Simulation result of one cell model 1) the original design 2) the modified design

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

A new circuit of RMCNN w/o EO is implemented. The new circuit has the same recognition rate with RMCNN with elapsed operation, but the operation of RMCNN w/o EO is simpler.

The new RMCNN w/o EO doesn't need a elapsed period to get the feature enhanced ratio weights. The RMCNN w/o EO can generate the feature enhance ratio weights directly after pattern learning, and it has a good recognition rate that is the same with RMCNN with elapsed operation. Though the operation of the RMCNN w/o EO is simpler, the circuit of RMCNN w/o EO isn't complicated. The RMCNN w/o EO doesn't need the multi-divider (M/D)[18] in the RMCNN with elapsed operation. Thus the transistors count of RMCNN w/o EO is less than the RMCNN with elapsed operation. Besides, there is a division behavior in RMCNN w/o EO, but there isn't any divider in the circuit. Thus the hardware of RMCNN w/o EO is simple.

The number of the learning patterns of RMCNN w/o EO is 3. They are Chinese characteristic one , two and four (一, 二 and 四). Maximum standard deviation of normal distribution noise is about 0.3. The number of learning patterns that RMCNN w/o EO can remember is still few. To increase the number of learning patterns that can be remembered, we should modify the learning algorithm or the recognizing algorithm continuously in the future.

In the experimental result, some vertical lines of pattern "四" are unrecognized. The recognition results of patterns "一" and "二" are successful and all lines in pattern "一" and "二" are horizontal. That doesn't mean the recognition rate of horizontal lines is better than vertical lines. The failure of recognition result dues to the ratio weights around one pixel and

the inputs of neighborhood pixels. If the ratio weights around one pixel are wrong, the recognition fails even that pixel is on horizontal line. Thus the failure of recognition will appear in other patterns like "五" if the wrong ratio weights are generated. Fig. 5.1 shows some examples that recognizing failure may happen and not all of the failure examples are vertical lines.

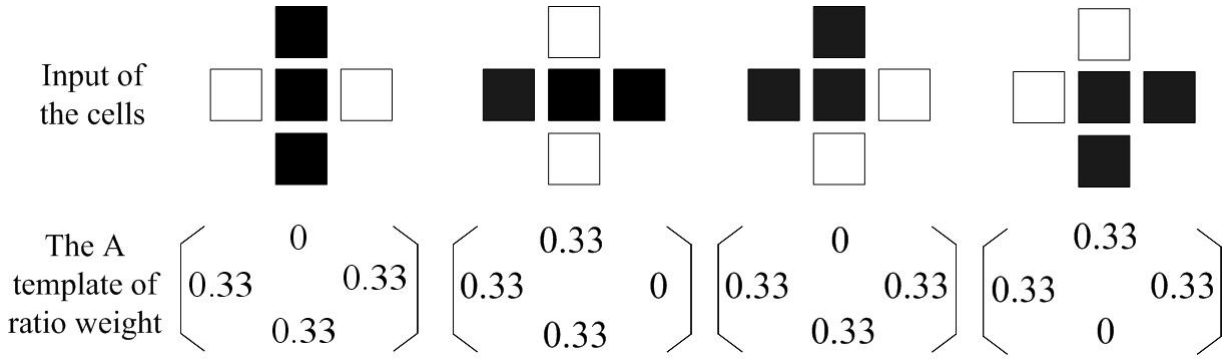

Fig. 5.1 Examples of Recognizing failure

5.2 Future Works

The RMCNN w/o EO in this thesis can't recognize all of the three patterns. The cause is found and the circuit is also redesigned in this thesis. Simulation supported that the modified design can really recognized all of the three patterns. Thus the RMCNN w/o EO should be taped out again. To reduce the chip area, the routing of RMCNN w/o EO should be modified too.

There are some modifying methods for the next chip are proposed in this thesis

1. The capacitance value should be optimized in the future. The operating speed of RMCNN w/o EO should be decided and the capacitance value of Cw and the saturated output current of **T2D** *Iysat* can be chosen according to the operating speed of RMCNN w/o EO.
2. The routing of layout can be more effective and save die area and the smaller die area has less process variation.
3. The static D-flip-flop should be used instead of dynamic D-flip-flop.
4. The modified output stage can be used to save power consumption.

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] T. Roska, "Analog events and a dual computing structure using analog and digital circuits and operators," *Discrete Event Systems: Models and Applications*, pp. 225-238, P. Varaiya and A. B. Kurzhanski (ed), Springer Verlag, New York, 1988.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] S. Grossberg, “Nonlinear difference-differential equations in prediction and learning theory,” in *Proc. Natl. Acad. Sci. USA*, vol. 58, pp. 1329-1334, 1967.

[12] J. A. Feldman and D. H. Ballard, “Connectionist models and their properties,” *Cognitive Science*, vol. 6, pp. 205-254, 1982.

[13] S. Grossberg, *The Adaptive Brain I: Cognition, Learning, Reinforcement, and Rhythm*, Elsevier/North-Holland, Amsterdam, 1986.

[14] 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.

[15] 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.

[16] B. Kosko, “Bidirectional associative memories,” *IEEE Trans. Systems, Man, and Cybernetics*, vol. 18, no. 1, pp. 49-60, Jan./Feb. 1988.

[17] L. O. Chua, M. Hasler, G. S. Moschytz, and J. Neirynck, “Autonomous cellular neural networks: a unified paradigm for pattern formation and active wave propagation,” *IEEE Trans. Circuits Syst. I*, vol. 42, pp. 559-577, Mar. 1995.

[18] Chung-Yu Wu and Chiu-Hung 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 , Dec. 2002 pp. 1713 - 1723

[19] Chung-Yu Wu and Chiu-Hung Cheng,” The Design of Cellular Neural Network with Ratio Memory for Pattern Learning and Recognition” *Cellular Neural Networks and Their Applications*, 2000, 23-25 May 2000 , pp. 301 – 307

[20] Student: J. F. Lan ; Advisor: 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

簡歷

姓 名：吳　諭

學 歷：

台北市立建國高級中學（85年9 月~88 年6 月）

國立中央大學電機工程學系（88年9 月~92 年6 月）

國立交通大學電子研究所碩士班（92年9 月~94 年9 月）

研究所修習課程：

類比積體電路 I	吳介琮教授
類比積體電路 II	吳重雨教授
數位積體電路	柯明道教授
積體電路之靜電放電防護設計特論	柯明道教授
數位通訊	桑梓賢教授
高等數位信號處理	劉志尉教授
隨機過程	王聖智教授
混合訊號式積體電路設計與實驗 I	吳介琮教授

永久地址：板橋市介壽街66之2號
Email： vivid175.ee92g@nctu.edu.tw
u88084100@cc.ncu.edu.tw
m9211657@alab.ee.nctu.edu.tw