

2011

# A Prototype CVNS Distributed Neural Network

Golnar Khodabandehloo  
*University of Windsor*

Follow this and additional works at: <http://scholar.uwindsor.ca/etd>

---

## Recommended Citation

Khodabandehloo, Golnar, "A Prototype CVNS Distributed Neural Network" (2011). *Electronic Theses and Dissertations*. Paper 433.

This online database contains the full-text of PhD dissertations and Masters' theses of University of Windsor students from 1954 forward. These documents are made available for personal study and research purposes only, in accordance with the Canadian Copyright Act and the Creative Commons license—CC BY-NC-ND (Attribution, Non-Commercial, No Derivative Works). Under this license, works must always be attributed to the copyright holder (original author), cannot be used for any commercial purposes, and may not be altered. Any other use would require the permission of the copyright holder. Students may inquire about withdrawing their dissertation and/or thesis from this database. For additional inquiries, please contact the repository administrator via email ([scholarship@uwindsor.ca](mailto:scholarship@uwindsor.ca)) or by telephone at 519-253-3000ext. 3208.

# **A Prototype CVNS Distributed Neural Network**

by

**Golnar Khodabandehloo**

A Dissertation

Submitted to the Faculty of Graduate Studies through  
the Department of Electrical and Computer Engineering in Partial Fulfillment  
of the Requirements for the Degree of Doctor of Philosophy at the  
University of Windsor

Windsor, Ontario, Canada  
2011

© 2011 Golnar Khodabandehloo

All Rights Reserved. No Part of this document may be reproduced, stored or otherwise retained in a retrieval system or transmitted in any form, on any medium by any means without prior written permission of the author.

A Prototype CVNS Distributed Neural Network

by

Golnar Khodabandehloo

APPROVED BY:

---

Y. Savaria

Department of Electrical Engineering, Polytechnique Montral

---

A. Jaekel

School of Computer Science, University of Windsor

---

C. Chen

Department of Electrical and Computer Engineering, University of Windsor

---

M. Khalid

Department of Electrical and Computer Engineering, University of Windsor

---

M. Ahmadi

Department of Electrical and Computer Engineering, University of Windsor

---

M. Mirhassani

Department of Electrical and Computer Engineering, University of Windsor

---

G. Reader

Department of Mechanical, Automotive and Materials Engineering, University of Windsor

March 25, 2011

---

## *Declaration of Previous Publication*

---

This thesis includes 5 original papers that have been previously published/submitted for publication in peer reviewed journals, as follows:

<b>Thesis Chapter</b>	<b>Publication title</b>	<b>Publication status</b>
Chapters 3,4, and 6	A prototype CVNS distributed neural network using synapse-neuron modules	Submitted
Chapter 5	A 16-level current-mode CVNS memory	Published
Chapter 5	CVNS-based storage and refreshing scheme for a multi-valued dynamic memory	In-press
Chapter 2	Resistive-type CVNS distributed neural networks with improved noise to signal ratio	Published
Chapter 4	Analog implementation of a novel resistive-type sigmoidal neuron	In-press

I certify that I have obtained a written permission from the copyright owner(s) to include the above published material(s) in my thesis. I certify that the above material describes work completed during my registration as graduate student at the University of Windsor.

I declare that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my

thesis. I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

---

# *Abstract*

---

Artificial neural networks are widely used in many applications such as signal processing, classification, and control. However, their practical implementation is challenging due to the number of inputs, the cost of storing the required weights, and the difficulty in realizing the activation function.

In this work, Continuous Valued Number System (CVNS) distributed neural networks are proposed which are providing the network with self-scaling property. This property aids the network to cope spontaneously with different number of inputs. The proposed CVNS DNN can change the dynamic range of the activation function spontaneously according to the number of inputs providing a proper functionality for the network.

In addition, multi-valued CVNS DRAMs are proposed to store the weights as CVNS digits. These memories can store up to 16 levels, equal to 4 bits, on each storage cell. In addition, they use error correction codes to detect and correct the error over the stored values.

A synapse-neuron module is proposed to decrease the design cost. It contains both synapse and neuron and the relevant components. In these modules, the activation function is realized through analog circuits which are far more compact compared to the digital look-up-tables while quite accurate.

Furthermore, the redundancy between CVNS digits, together with the distributed structure of the neuron make the proposal stable against process variations and reduce the noise to signal ratio.



*To my family,  
with love ...*

---

## *Acknowledgments*

---

There are several people who deserve my sincere thanks for their generous contributions to this project. I would first like to express my sincere gratitude and appreciation to Dr. Mitra Mirhassani and Dr. Majid Ahmadi, my supervisors for their invaluable guidance and constant support throughout the course of this thesis work.

In addition to my advisors, I would like to thank the rest of my thesis committee: Dr. Chunhong Chen and Dr. Mohammed Khalid from the electrical and computer engineering department, Dr. Arunita Jaekel from the school of computer science, and Dr. Yvon Savaria from the Polytechnique Montreal for their participation in my seminars, reviewing my thesis, and their constructive comments.

Also, I would like to thank Dr. Roberto Muscedere for his assistants regarding the VLSI CAD tools and facilities used during the course of the project.

I am grateful to my colleague, friend, and partner Ashkan H. Namin who has supported and believed in me.

Finally, my deepest gratitude goes to my family for their unconditional love, support, and encouragement.

# Contents

<b>Declaration of Previous Publication</b>	<b>iv</b>
<b>Abstract</b>	<b>vi</b>
<b>Dedication</b>	<b>viii</b>
<b>Acknowledgments</b>	<b>ix</b>
<b>List of Figures</b>	<b>xiii</b>
<b>List of Tables</b>	<b>xvi</b>
<b>List of Abbreviations</b>	<b>xvii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Summary of Contributions . . . . .	7
1.2 Outline of the Thesis . . . . .	7
<b>2 CVNS Distributed Neural Network</b>	<b>9</b>
2.1 Previous Structures of Resistive-Type ANNs . . . . .	9
2.1.1 Lumped Neural Network . . . . .	10
2.1.2 Conventional Distributed Neural Network . . . . .	11
2.2 CVNS Distributed Neural Network . . . . .	12

---

2.2.1	Scalability . . . . .	16
2.2.2	2-2-1 XOR . . . . .	16
2.3	CVNS Fully Distributed Neural network . . . . .	17
2.4	Stochastic Model . . . . .	18
2.5	Comparison . . . . .	23
2.6	Conclusion . . . . .	24
<b>3</b>	<b>Truncated CVNS Distributed Neural Network</b>	<b>26</b>
3.1	Truncated CVNS DNN . . . . .	27
3.2	Effect of Truncation on Performance . . . . .	31
3.3	NSR Calculation . . . . .	33
3.4	Conclusion . . . . .	35
<b>4</b>	<b>Synapse-Neuron Module</b>	<b>37</b>
4.1	CVNS DRAM . . . . .	38
4.2	Multiplier Module . . . . .	41
4.3	Interface Circuit . . . . .	42
4.4	Sigmoidal Neuron . . . . .	44
4.5	Voltage-Mode ADC . . . . .	45
4.6	Conclusion . . . . .	47
<b>5</b>	<b>CVNS Multi-Valued Dynamic Memory</b>	<b>48</b>
5.1	CVNS Modifications for Implementation . . . . .	51
5.2	Noise Margin and Error Correction Based on CVNS . . . . .	54
5.3	CVNS-Based Storage Circuitry . . . . .	58
5.4	CVNS-Based Refreshing Scheme and Circuits . . . . .	64
5.4.1	Analog to Digital Converter . . . . .	66
5.4.1.1	Current Comparators . . . . .	70
5.4.1.2	Current Mirrors . . . . .	71

---

---

5.4.2	Digital to Analog Converter . . . . .	72
5.5	Simulation Results and Comparisons . . . . .	77
5.6	Conclusion . . . . .	81
<b>6</b>	<b>A 4-3-2 CVNS DNN using Synapse-Neuron Modules</b>	<b>83</b>
6.1	Prototype CVNS DNN . . . . .	83
6.2	Comparisons . . . . .	87
6.3	Conclusion . . . . .	89
<b>7</b>	<b>Conclusions and Future Work</b>	<b>90</b>
7.1	Conclusions . . . . .	90
7.2	Future Work . . . . .	91
	<b>References</b>	<b>93</b>
	<b>VITA AUCTORIS</b>	<b>99</b>

---

# List of Figures

1.1	Block diagram of the CVNS digit generation from binary digits. . . . .	4
2.1	$(k + 1)$ -input resistive-type lumped neural network. . . . .	10
2.2	Block diagram of a $(k + 1)$ -input resistive-type DNN. . . . .	11
2.3	Proposed resistive-type CVNS DNN configuration. . . . .	13
2.4	Block diagram representation of the CVNS multiplication (equation (2.5)). .	14
2.5	Scalability of the proposed DNN for a sigmoidal activation function. . . . .	17
2.6	Block diagram of the $2 - 2 - 1$ XOR network. . . . .	18
2.7	Off-line network training by back propagation algorithm. . . . .	19
2.8	$(k + 1)$ -input resistive-type CVNS fully distributed neural network. . . . .	20
2.9	Comparison for input range of $[9, 25]$ : (a) Stochastic Gain Function for Adaline, NSR for (b) Adaline, (c) 5-layer Madaline, (d) 10-layer Madaline. . . . .	23
3.1	Final multiplication result in a high resolution environment. . . . .	30
3.2	Final multiplication result in a low resolution environment. . . . .	31
3.3	Simulation result of an XOR based on the truncated CVNS DNN. . . . .	31
3.4	NSR comparisons for Adalines and 5-layer Madalines. . . . .	35
4.1	Block diagram of the proposed synapse-neuron module. . . . .	38
4.2	Block diagram of the 13-bit multi-valued DRAM. . . . .	39
4.3	Layout of the 13-bit multi-valued DRAM. . . . .	40

---

4.4	Block diagram of the multiplier module. . . . .	41
4.5	Block diagram of the partial multiplication module. . . . .	41
4.6	Simulation result of '1, 1111, 1010, 1011' multiplied by '0111'. . . . .	42
4.7	Layout of the multiplication module. . . . .	43
4.8	Schematic of the interface circuit. . . . .	43
4.9	Schematic of the proposed resistive-type neuron to realize the sigmoid function. . . . .	44
4.10	Simulation result of the sigmoidal neuron. . . . .	45
4.11	schematic of a set of inverters used in the proposed voltage-mode ADC. . . . .	46
5.1	Block diagram of simplified CVNS digit generation using the group method. Squares show the weighted sum operation units. . . . .	52
5.2	Block diagram of the modified CVNS digit generation with $\psi = 4$ and $\nu = 1$ . . . . .	55
5.3	Overall block diagram of the proposed memory . . . . .	59
5.4	General storage scheme for a 16-bit input. In order to increase the reliability of the cells, the LSB input to each cell is repeated and is used for error correction. . . . .	59
5.5	Self-biased dynamic current mirror memory (current copier) . . . . .	60
5.6	Simulation results of different resistors in storage cell. . . . .	62
5.7	Storage cell layout . . . . .	63
5.8	Simulation result of storage cell layout for '1111' binary input. . . . .	63
5.9	Block diagram of the refreshing system for a CVNS DRAM cell. . . . .	65
5.10	Block diagram of the ADC . . . . .	66
5.11	4-bit ADC layout . . . . .	69
5.12	Schematic of a current comparator . . . . .	71
5.13	Schematic of a current source . . . . .	71
5.14	Schematic of the first current mirror . . . . .	73
5.15	Schematic of the second current mirror . . . . .	74

---

---

5.16	Overall configuration of the 4-bit DAC. . . . .	74
5.17	Schematic of a one-bit weighted current mirror . . . . .	75
5.18	DAC layout . . . . .	76
5.19	DAC layout simulation results for input of '1111' . . . . .	77
5.20	Post layout simulation result of total circuit for '1110' binary input. . . . .	78
5.21	Block diagram of the DRAM proposed by Lee et al. [1] . . . . .	80
6.1	Block diagram of the 4-3-2 CVNS DNN. . . . .	84
6.2	Training pattern set for the 4-3-2 CVNS DNN. . . . .	85
6.3	Layout of the proposed 4-3-2 CVNS DNN. . . . .	86
6.4	Simulation result of the 4-3-2 CVNS DNN to solve the classification problem of Figure 6.2. . . . .	87



# List of Tables

1.1	An example for radix-10 CVNS digits . . . . .	4
2.1	NSR of sigmoidal Adalines from <i>case study 2</i> . . . . .	21
2.2	Comparison of different sigmoidal Adalines for a $k + 1$ input . . . . .	24
3.1	Multiplication partial results for <i>case study 1</i> . . . . .	30
5.1	An example for the CVNS digits with different group lengths and digit links	54
5.2	An example for error correction when the binary input is 1011, 0101, 1110, 0011	58
5.3	Schematic storage cell and layout storage cell simulation results . . . . .	64
5.4	Values of all currents in the ADC block diagram . . . . .	67
5.5	Extraction of $x_3$ and $x_2$ from the output of current comparators. . . . .	67
5.6	Comparison made between the proposed ADCM and some published meth- ods for a 4-bit ADC . . . . .	68
5.7	Schematic ADC and layout ADC simulation results . . . . .	70
5.8	Transistor sizes for each current source in the layout . . . . .	72
5.9	Transistor sizes for each current mirror . . . . .	76
5.10	Schematic DAC and layout DAC simulation results . . . . .	77
5.11	Comparison made between the proposed DRAM circuit and available pub- lished structures . . . . .	79

---

## *List of Abbreviations*

---

Adaline	Adaptive Linear Neuron.
ADC	Analog to Digital Converter.
ANN	Artificial Neural Network.
CMOS	Complementary Metal-Oxide-Semiconductor.
CN	Common Node.
CVNS	Continuous Valued Number System.
DAC	Digital to Analog Converter.
DNN	Distributed Neural Network.
DRAM	Dynamic Random Access Memory.
ECC	Error Correction Code.
FDNN	Fully Distributed Neural Network.
IEEE	Institute of Electrical and Electronics Engineers.
LID	Least Informed Digit.
LSB	Least Significant Bit.
LUT	Look-Up-Table.
Madaline	Multiple Adaline.
MID	Most Informed Digit.
MSB	Most Significant Bit.
NSR	Noise to Signal Ratio.
QLG	Quantized Level Generator.
SCCV	Simulated Constant Current Value.
TSMC	Taiwan Semiconductor Manufacturing Company.
XOR	Exclusive OR.

---

# **Chapter 1**

---

## ***Introduction***

---

In the past decades, studies have been performed to use Artificial Neural Networks (ANNs) in different applications such as signal processing, pattern recognition, control, and many others [2–8]. One of the important features of an ANN is its scalability which means how the network responds if the number of inputs changes as different applications may need different number of inputs.

In conventional lumped ANNs, there is one neuron corresponding to all the synapses in each layer, where changing the number of inputs results in malfunctioning of the network [9]. In the case of a sigmoidal neuron, increasing the number of inputs would cause a larger saturation area for the neuron which changes the sigmoidal function to a hard limiting function. Decreasing the number of inputs, on the other hand, will result in a low-gain function. To overcome these problems, the neuron should be redesigned whenever the number of inputs changes which is not possible in hardware implementations. Therefore, the application of a lumped ANN becomes limited to applications with a certain number of inputs.

Resistive-type Distributed Neural Network (DNN) was introduced as an alternative for lumped neural networks [2, 3, 9, 10]. DNNs provide a self-scaling property for the network to cope with variable number of inputs, spontaneously. In DNNs, there are sub-neurons instead of neurons. Each sub-neuron is dedicated to one input; consequently, changing the number of inputs changes the number of sub-neurons. Therefore, DNNs can stretch or shorten the dynamic range of the activation function to keep the overall neuron characteristic in its proper functionality for different number of inputs. This causes an instant change in the dynamic range of the overall activation function according to the number of inputs. The use of sub-neurons in DNNs also provides some advantages for the network such as higher immunity to noise and process variations [9]. Noise to Signal Ratio (NSR) can be used to study the noise immunity of the ANNs [11]. Generally, a distributed configuration decreases the NSR compared to the lumped networks [3, 9, 10]. Here, NSR is used instead of signal to noise ratio to create a simple analytical expression.

In this work, a structure for feed-forward neural networks is proposed based on the Continuous Valued Number System (CVNS) as a derivative of conventional DNNs. The CVNS [12, 13] is a redundant number system which has been successfully employed in building high performance low power arithmetic units, multi-valued memories, and neural networks [14–20].

Considering inputs and outputs of the CVNS memory as binary digits, it is shown that there is an easy way for converting digits from binary to the CVNS and vice versa. The general representation of an  $(m + 1)$ -bit binary value is as follows.

$$x = \pm \sum_{i=0}^m x_i \cdot \beta^i \quad (1.1)$$

where  $x_i$  represents binary digits and  $\beta$  shows the radix;  $\beta = 2$  for a binary radix.

For example, a binary value of  $x = 101001$  is equal to  $1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 41$ .

A CVNS set is an ensemble of multiple CVNS digits,  $((x))_i$ , which are generally generated based on two methods: a series method and a parallel method [13]. The former one,

---

called *cascade Digit Generation*, starts from the first CVNS digit and generates digits one by one in series; each digit is extracted from the previously generated digits.

An other method called *Modular Reduction Digit Generation* is used in this work. It generates all the CVNS digits at the same time, in parallel, and independent to each other. Thus, this method provides a higher speed of conversion together with a lower complexity compared to the series method and is as follows.

$$((x))_{n-i} = \left( \frac{x}{M} \cdot \beta^{n-i+1} \right) \text{mod} \beta = \left( \frac{x}{M} \cdot 2^{n-i+1} \right) \text{mod} 2 \quad (1.2)$$

where  $n$  is the maximum index of the CVNS digits,  $0 \leq i \leq n$ .  $M$  is the maximum representation range, and  $\beta$  is the CVNS radix which is chosen low, equal to 2 ( $\beta = 2$ ), in order to reduce the complexity of the circuit further more. *mod* stands for the modular reduction operation on any continuous real value such that  $(a) \text{mod} \beta = a - I \times \beta$  where  $0 \leq (a) \text{mod} \beta < \beta$ .  $I$  is an integer ( $I = 0, 1, \dots, \beta - 1$ ) which in this case is either 0 or 1.

In order to simplify the conversion between binary and CVNS, the maximum range of the representation for both the binary and the CVNS systems is selected as  $2^{m+1} = 2^{n+1} = M$ . Therefore, the number of binary digits,  $m + 1$ , and the CVNS digits,  $n + 1$ , are the same.

Under these conditions, a CVNS digit is directly generated from its corresponding binary digits through a general expression by modifying expression (1.2) as follows.

$$((x))_{n-j} = \sum_{i=0}^{m-j} x_i \cdot 2^{i+j-m}, \quad j = 0, 1, \dots, n \quad (1.3)$$

By eliminating *mod2* operation in digit generation, this pre-congruent relation reduces the number of required gates for system implementation. According to the expression given by the equation (1.3), the CVNS digit generation directly from the binary digits can be realized employing simple weighted current mode mirrors. A block diagram representation of equation (1.3) is shown in Figure 1.1, where squares show the weighted sum operation units.

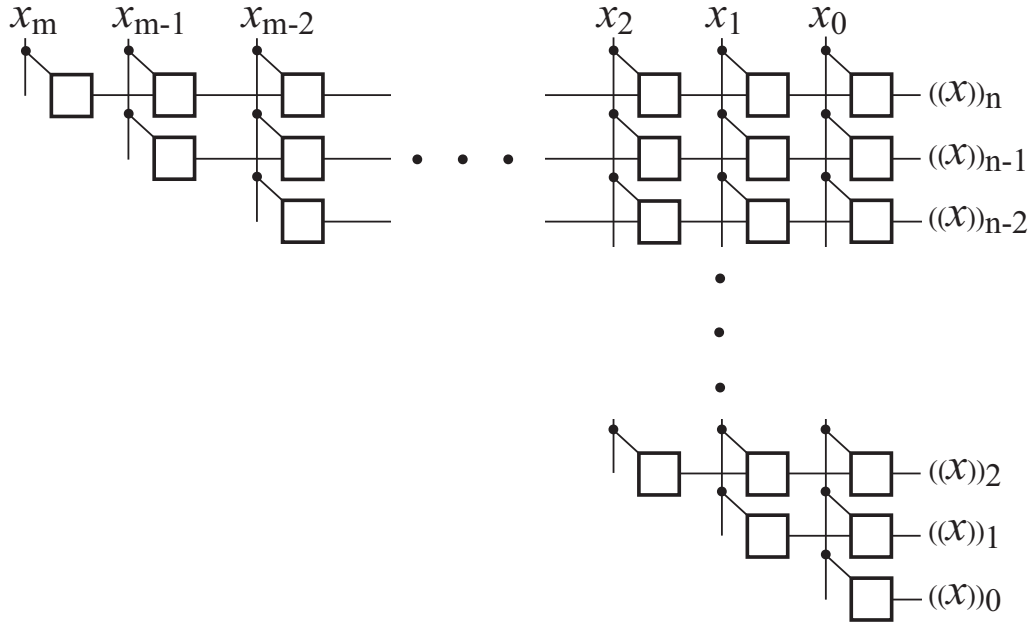


Figure 1.1: Block diagram of the CVNS digit generation from binary digits.

Table 1.1 shows the first five CVNS digits,  $((x))_i$ , of an arbitrary chosen value in radix-10 with maximum level of redundancy. Here, the original value is 89.0537412.

Table 1.1: An example for radix-10 CVNS digits

$((x))_n$	$((x))_{n-1}$	$((x))_{n-2}$	$((x))_{n-3}$	$((x))_{n-4}$
8.90537412	9.0537412	0.537412	5.37412	3.7412

In Table 1.1,  $\{8\}$  appears only in the first CVNS digit,  $\{9\}$  is repeated in only first two CVNS digits while  $\{3, 7, 4, 1, 2\}$  are repeated in first five CVNS digits. In other words, in a CVNS digit set, lower index CVNS digits can be rebuilt from higher index CVNS digits ( $((x))_{n-3} = (((x))_{n-2} \times 10) \bmod 10 = (((x))_{n-1} \times 10^2) \bmod 10 = (((x))_n \times 10^3) \bmod 10$ ). Also, lower index CVNS digits can help in recovering the higher index ones ( $((x))_n = 8 + ((x))_{n-1}/10 = 8.9 + ((x))_{n-2}/10^2$ ).

The CVNS representation also offers some degrees of freedom in selecting some representation dimensions such as the level of information overlap between the digits and the number of the required CVNS digits according to design demands. The CVNS digit generation is limited by the resolution of the environment. Therefore, truncation methods are applied to the CVNS digits to make them cope with lower resolution environments while still keeping the CVNS properties [14, 15]. One of the advantages of applying these methods is reducing the area consumption; however, it may result in lower accuracy as well.

Consequently, the CVNS digits have information overlap with each other; every digit has some level of knowledge about the digits with lower indices in the same set. This digit level redundancy is used to detect and correct errors in the digit set which makes the system more robust against the noise. The proposed CVNS DNN has all the advantages of using DNNs; furthermore, it results in a reduced NSR for the network compared to conventional DNNs. This makes the CVNS DNNs a great candidate for hardware implementations.

The CVNS can be implemented through current-mode analog circuitries which generally provide lower noise, lower voltage swing, and the ability of working with lower power supply voltages [21–23]. Nevertheless, some functions such as addition can be easily realized through current-mode circuitries.

In this work, a current-mode synapse-neuron module is designed based on the CVNS which can be used as the building block for a wide variety of applications. The synapse-neuron module decreases the design cost and increases the fan-out [3]. The weights to this module are considered to be equal to 13 bits. Truncated CVNS digits equivalent to weights are generated and stored in multi-valued DRAMs with the ability of error correction [14, 15]. These memories can store up to 16-level equivalent to 4-bit on each storage cell.

Each two adjacent CVNS digits, in the proposed DRAM, has equivalent to one binary digit overlap; this common digit is used for error correction in the system. This configuration provides a novel multi-valued DRAM with increased noise margin and reduced area. A fast Analog to Digital Converter (ADC) [24] is proposed for the refreshing circuitry. This

---

ADC has a parallel configuration and can convert two bits simultaneously. It can reduce the overall delay of the refreshing circuitry.

The proposed DRAM has to be refreshed continuously; the voltage over the storing capacitor drops after a while due to the leakage currents of the storage cell. During the refreshing cycle, the storage capacitor is charged to the correct value indicated by the error correction codes.

Except for the multi-valued memory, the proposed synapse-neuron module contains multiplication module, interface module, and sub-neuron. The multiplication module multiplies the truncated CVNS digits by the input. A voltage-mode Analog to Digital Converter (ADC) extracts the digits of the input for the sake of multiplication. The inputs are converted to a 4-bit value and are applied as multipliers to the multiplication modules.

Sign, the 13<sup>th</sup> bit of the weight, is applied to the multiplier output in the interface module releasing the final synapse output which goes to a Common Node (CN). CN is common between all the synapse-neuron modules in a layer and acts as an input/output node. In this node, currents from synapses are added up together and divided by the number of sub-neurons. Accordingly, all the sub-neurons receive an identical current. Each sub-neuron in the synapse-neuron module decides the output value based on this current.

The sigmoid function is selected as the activation function which always has a value between 0 and 1. A new analog resistive-type neuron is designed which generates the sigmoid function based on the transistor characteristics in both triode and saturation regions.

To justify the design concept, the proposed synapse-neuron modules are used to design a prototype 4-3-2 network with four digital inputs and two final digital outputs. This prototype is trained off-line based on a classification pattern. The circuitries are designed, simulated, and laid out in 0.18 $\mu m$  TSMC CMOS technology [25] using a power supply voltage of 1.8V.



## 1.1 Summary of Contributions

In this dissertation, a new family of the CVNS DNNs are introduced. The scalability and noise immunity of the proposal are studied and compared to the lumped and conventional distributed networks. Studies show that it outperforms the previous structures because of its low NSR.

The proposed CVNS DNN is implemented considering the environmental resolution limitations which is equal to 4-bit in the employed technology. Multi-valued DRAMs with error correction abilities are designed to convert and store the weights in the form of the CVNS digits. The proposed multi-valued memories are a significant progress in this field because of their lower area and power consumption as well as lower refreshing cycle time.

A 4-3-2 network is designed, laid out, and trained off-line based on a pattern set. The post-layout simulations are performed which show the proper functionality of the design.

## 1.2 Outline of the Thesis

The next chapters are organized as follows. In Chapter 2, the structures of a typical lumped ANN and a conventional DNN are shown and the proposed CVNS DNN structure is introduced. It also studies the scalability in the proposed DNN, and shows how the CVNS DNN is used to solve an XOR problem. Designing CVNS fully distributed neural network according to the CVNS DNN is discussed. A Stochastic NSR model is used, and NSR of different sigmoidal Adalines are compared together.

Chapter 3 presents the environmental limitations for implementation of the proposed CVNS DNN. Truncated methods are used to cope with these limitations. The effect of truncation on the performance and noise immunity are studied. Simulations show that the truncated CVNS DNN still has lower NSR compared to the lumped and conventional distributed neural networks.

The synapse-neuron module is presented in Chapter 4. The sub-modules of each synapse-

neuron module are the storage module, multiplication module, interface module, and sub-neuron. For each sub-module, the task and circuitries are discussed. The synapse-neuron module is designed in current-mode. There is also a voltage-mode ADC to convert the output of each layer to 4 bits that can be used as the input to the next layer.

The proposed current-mode multi-valued CVNS DRAM is introduced in detail in Chapter 5. In this chapter, the CVNS is described and the truncated CVNS is introduced in detail. Sub-modules of the CVNS DRAM and their circuitries are presented. The proposed DRAM is using a series combination of ADC and DACs for conversion, storage, and correction. A fast current-mode ADC method is proposed and compared to current-mode ADC methods in literature. A method for decreasing the refresh rate is also proposed. Error correction in this DRAM is discussed based on the CVNS features. Finally, the proposed memory is compared to current-mode memories in literature.

In chapter 6, a prototype 4-3-2 network is designed. This network is trained based on a pattern set to solve a classification problem. The weights and biases are calculated in MATLAB and used for post-layout simulations. The final layout and post-layout simulation results are also shown in this chapter. The circuitries are designed in  $0.18\mu m$  TSMC CMOS technology.

Finally, concluding remarks and future work are presented in Chapter 7.

---

## **Chapter 2**

---

### ***CVNS Distributed Neural Network***

---

In this chapter, a new family of resistive-type feed-forward distributed neural networks is proposed based on the CVNS. The proposed CVNS DNN is used for off-line training of a prototype XOR. A stochastic model is employed to formulate NSR of a sigmoidal Adaline based on the proposed CVNS DNN. This is compared to the NSR of sigmoidal Adalines based on other comparable structures. A fully distributed neural network is designed based on the proposed CVNS DNN which can provide even lower level of NSR in the network while using more neurons. The reduced sensitivity to noise property of the proposed structures makes them robust against process variations; hence, they are potential candidates for hardware applications such as industrial sensors.

#### **2.1 Previous Structures of Resistive-Type ANNs**

In resistive-type ANNs, the neuron function is realized by a nonlinear load that receives current from the synapses in a general node and generates a corresponding voltage at the

---

same node [3, 9, 10].

Although numerous designs of neural networks have been reported, in this chapter, the basic structure of resistive-type feed-forward neural networks is categorized into two main groups: lumped and distributed neural networks.

### 2.1.1 Lumped Neural Network

Lumped ANN is the original structure of neural networks in which all synapses are connected to one neuron in each layer. The block diagram of a  $(k + 1)$ -input resistive-type lumped ANN for the  $l^{th}$  layer is shown in Figure 2.1. Each input is multiplied by a synaptic weight. All of the multiplication results are added up together in a common node, and the result goes to an activation function to generate the final output.

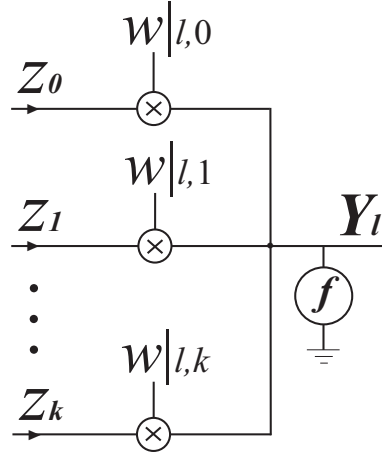


Figure 2.1:  $(k + 1)$ -input resistive-type lumped neural network.

The output of the neuron,  $Y_l$ , is generated through the following equation.

$$Y_l = f\left(\sum_{t=0}^k Z_t \times w|l, t\right) \quad (2.1)$$

where  $Z_t$  shows the  $t^{th}$  input, and  $w|l, t$  shows its corresponding synaptic weight. Activation function is represented by  $f$ .

### 2.1.2 Conventional Distributed Neural Network

Resistive-type DNNs provide one neuron for each synapse. The block diagram of a  $(k + 1)$ -input resistive-type DNN [3, 9, 10] is shown in Figure 2.2.

Here, instead of one neuron, sub-neurons are used to realize the activation function. The input and weight multiplication results are added up together. The result is divided by the number of sub-neurons so that each sub-neuron receives the same amount of current. It should be noted that in each layer, all the sub-neurons have the same activation function. The output of such a network is calculated as follows.

$$Y_l = f\left(\frac{1}{k+1} \sum_{t=0}^k Z_t \times w|l,t\right) \quad (2.2)$$

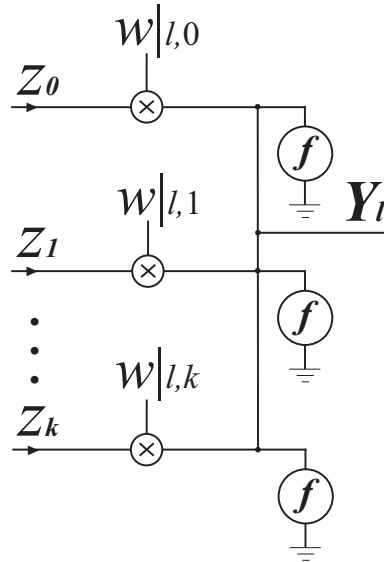


Figure 2.2: Block diagram of a  $(k + 1)$ -input resistive-type DNN.

As there is one sub-neuron corresponding to each input, the number of sub-neurons changes by the number of inputs. This self-scaling property of the DNN causes an increase or a decrease in the dynamic input range of the activation function as the number of inputs increases or decreases, respectively. The scalability feature of DNNs is used to build a new family of neural networks as is discussed next.

## 2.2 CVNS Distributed Neural Network

The proposed network is a distributed neural network in which weights are stored as CVNS digits. A family of CVNS neural networks was introduced in [20] where the output was generated from a Reverse Evolution (RE) unit using complicated sub-functions. In the proposed network, the design is modified to avoid RE and sub-functions which decreases the network complexity and makes it more suitable for practical hardware implementations.

Figure 2.3 shows the proposed  $(k + 1)$ -input resistive-type CVNS DNN configuration for the  $l^{th}$  layer.

In Figure 2.3,  $((w))_{n-j}|_{l,t}$  ( $j = 0, 1, \dots, n$  and  $t = 0, 1, \dots, k$ ) stands for the CVNS presentation of weights which are generated through the following equation.

$$((w))_{n-j}|_{l,t} = \left(\frac{w_t}{M} * B^{j+1}\right) \text{ mod } B \quad j = 0, 1, \dots, n \quad (2.3)$$

where  $w_t$  stands for the value of the weight corresponding to the  $t^{th}$  input,  $M$  is the maximum acceptable value of weights, and  $B$  is the CVNS radix.

In a CVNS digit set, digits with higher indices have information overlap with the lower index digits. For example the equivalent CVNS digits of the randomly chosen value of  $w_t = 31.89$  for  $M = 100$  and  $B = 10$  are  $((w))_3 = 3.189$ ,  $((w))_2 = 1.89$ ,  $((w))_1 = 8.9$ ,  $((w))_0 = 9$ . These CVNS digits can be shown in a CVNS digit set as  $\{3.189, 1.89, 8.9, 9\}$ .

The CVNS multiplication [13] of  $Z_t$  ( $t = 0, 1, \dots, k$ ) and  $((w))_{n-j}$  ( $j = 0, 1, \dots, n$ ) is calculated as follows.

First, the digits of the multiplier,  $(Z)_{D-d}$ , are extracted as is shown in the following equation.

$$(Z)_{D-d} = (Z_t \times B^d) \text{ mod } B - \frac{(Z_t \times B^{d+1}) \text{ mod } B}{B} \quad (2.4)$$

where  $D$  shows the number of digits of  $Z_t$  and  $d = 0, 1, \dots, D - 1$ .

In the next step, the CVNS multiplication result,  $((u))_{n-j}$ , is calculated through the

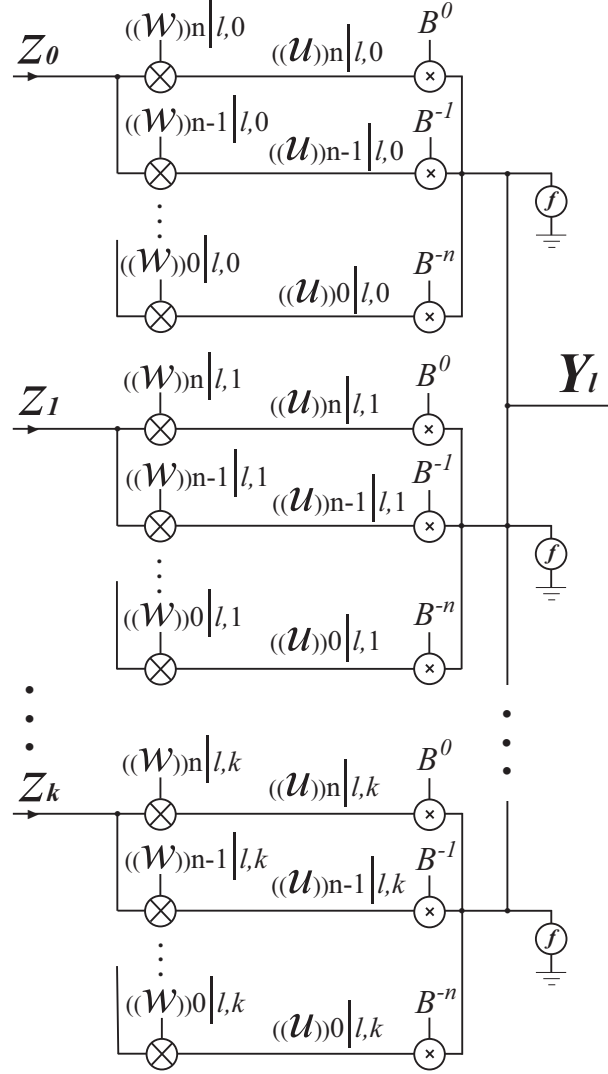


Figure 2.3: Proposed resistive-type CVNS DNN configuration.

following equation. The block diagram representation is shown in Figure 2.4.

$$\begin{aligned}
 ((u))_{n-j} &= Z \otimes ((w))_{n-j} \\
 &= \{(Z)_D \times ((w))_{n-j} \\
 &\quad + \sum_{i=1}^{D-1} (Z)_{D-i} \times ((w))_n \times B^{j-i}\} \text{mod } B
 \end{aligned} \tag{2.5}$$

*Case study 1:* In a high resolution environment, the CVNS multiplication result for a

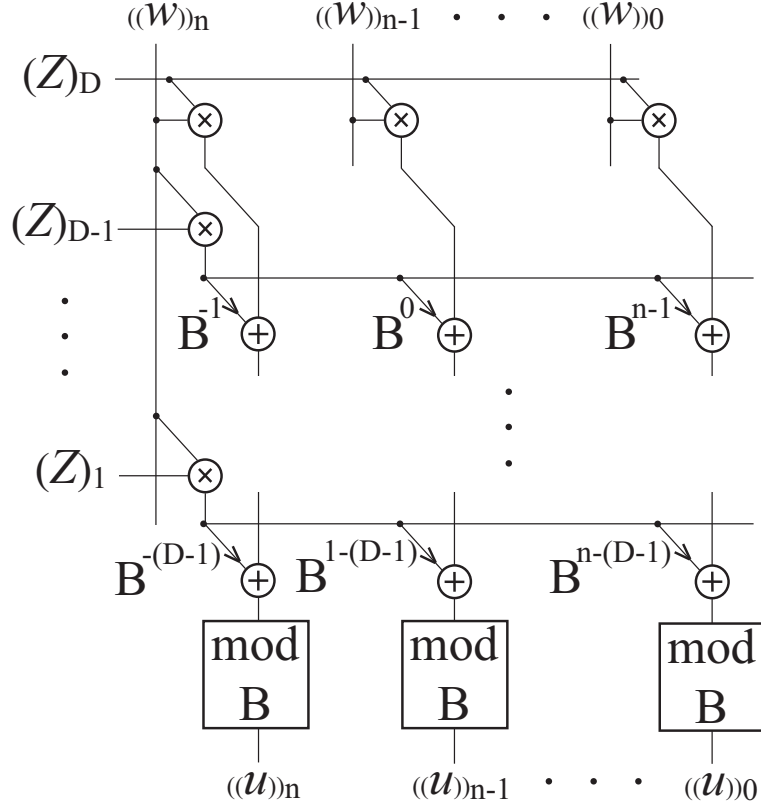


Figure 2.4: Block diagram representation of the CVNS multiplication (equation (2.5)).

multiplier of  $Z_t = 2.14$  (digits are  $[2, 1, 4]$ ) and a CVNS digit set multiplicand of  $\{3.189, 1.89, 8.9, 9\}$  with  $B = 10$  is as follows:

$$((u))_3 = \{2 \times 3.189 + 1 \times 3.189 \times 10^{-1} + 4 \times 3.189 \times 10^{-2}\} \text{mod } 10 = 6.82446$$

$$((u))_2 = \{2 \times 1.89 + 1 \times 3.189 \times 10^0 + 4 \times 3.189 \times 10^{-1}\} \text{mod } 10 = 8.2446$$

$$((u))_1 = \{2 \times 8.9 + 1 \times 3.189 \times 10^1 + 4 \times 3.189 \times 10^0\} \text{mod } 10 = 2.446$$

$$((u))_0 = \{2 \times 9.0 + 1 \times 3.189 \times 10^2 + 4 \times 3.189 \times 10^1\} \text{mod } 10 = 4.46$$

Digits in the multiplication output set,  $\{6.82446, 8.2446, 2.446, 4.46\}$ , are CVNS digits as they fit in the following equation which shows the relation between digits in a CVNS



digit set.

$$(((x))_{n-j} \times B) \bmod B = ((x))_{n-j-1} \quad j = 0, 1, \dots, n \quad (2.6)$$

As it is shown in equation (2.6), there is a redundancy between the CVNS digits. Lower index CVNS digits can be calculated from the higher index digits; however, the process can not be reversed. In other words, the higher the index of a CVNS digit is, the higher the level of its information will be. However, to add the CVNS digits together, they have to be of the same information level. To meet this requirement,  $((u))_{n-j}|_{l,t}$  is multiplied by  $B^{-j}$  ( $j = 0, 1, \dots, n$ ) in Figure 2.3.

In Figure 2.3, the total output of the CVNS DNN is calculated as  $Y_l = f(y)$  where  $y$  is generated from the CVNS multiplication outputs as follows.

$$\begin{aligned} y = & \frac{1}{k+1} \left\{ ((u))_n|_{l,0} + \frac{((u))_{n-1}|_{l,0}}{B} + \dots + \frac{((u))_0|_{l,0}}{B^n} \right. \\ & + ((u))_n|_{l,1} + \frac{((u))_{n-1}|_{l,1}}{B} + \dots + \frac{((u))_0|_{l,1}}{B^n} + \dots \\ & \left. + ((u))_n|_{l,k} + \frac{((u))_{n-1}|_{l,k}}{B} + \dots + \frac{((u))_0|_{l,k}}{B^n} \right\} \end{aligned} \quad (2.7)$$

From the CVNS features, it can be observed that CVNS addition of different CVNS digit sets results in CVNS digits as well. The following example shows the CVNS addition of four randomly chosen CVNS digit sets:  $\{2.345, 3.45, 4.5, 5\}$ ,  $\{7.891, 8.91, 9.1, 1\}$ ,  $\{3.042, 0.42, 4.2, 2\}$ , and  $\{9.157, 1.57, 5.7, 7\}$ .

$$(2.345 + 7.891 + 3.042 + 9.157) \bmod 10 = 2.435$$

$$(3.45 + 8.91 + 0.42 + 1.57) \bmod 10 = 4.35$$

$$(4.5 + 9.1 + 4.2 + 5.7) \bmod 10 = 3.5$$

$$(5 + 1 + 2 + 7) \bmod 10 = 5$$

The output digit set is a CVNS digit set according to equation (2.6).

In equation (2.7), the CVNS digits with the same index can be added up together as follows.

$$(((u))_{n-j}|_{l,0} + ((u))_{n-j}|_{l,1} + \dots + ((u))_{n-j}|_{l,k}) \bmod B = ((y))_{n-j}|_l \quad (2.8)$$


---

It should be noted that  $A = A \bmod B + I \times B$  where  $I$  shows the quotient of  $A$  divided by  $B$ . Therefore, equation (2.8) can be rewritten as the following equation.

$$((u))_{n-j|l,0} + ((u))_{n-j|l,1} + \dots + ((u))_{n-j|l,k} = ((y))_{n-j|l} + I_{n-j} \times B \quad (2.9)$$

According to equation (2.9), equation (2.7) can be represented as follows.

$$y = \frac{1}{k+1} \{ ((y))_{n|l} + \frac{((y))_{n-1|l}}{B} + \dots + \frac{((y))_{0|l}}{B^n} + Cons \} \quad (2.10)$$

where  $Cons$  is equal to  $\sum_{j=0}^n \frac{I_{n-j} \times B}{B^j}$ .

### 2.2.1 Scalability

For studying the self-scaling property of the proposed DNN, the value of the total current entering the sub-neurons,  $y \times (k + 1)$ , is considered to be between  $-60\mu A$  and  $60\mu A$  for 1-input ( $k = 0$ ). Consequently for 5-input ( $k = 4$ ), the corresponding current,  $y \times (5)$ , will be in the range of  $-300\mu A$  and  $300\mu A$ . Self-scaling property of the proposed DNN is shown in Figure 2.5 where it compares the output for networks with 1 and 5 inputs together for a sigmoidal activation function.

As is shown in Figure 2.5, the network is properly stretching the dynamic range to cope with the increased number of inputs. Note that without a proper scaling, the output of 5-input case will be equal to  $0V$  for all inputs in the range of  $[-300\mu A, -60\mu A]$  and will be  $1.8V$  for inputs in the range of  $[60\mu A, 300\mu A]$ .

### 2.2.2 2-2-1 XOR

The proposed CVNS DNN is used to build a prototype 2 – 2 – 1 XOR as is shown in Figure 2.6.

Here,  $B$  is chosen equal to 10, sigmoid function is chosen as the activation function, and  $n$  is selected equal to 5. The network is trained off-line using the back propagation algorithm [5, 26], and weights are calculated while the biases are considered to be zero.

---

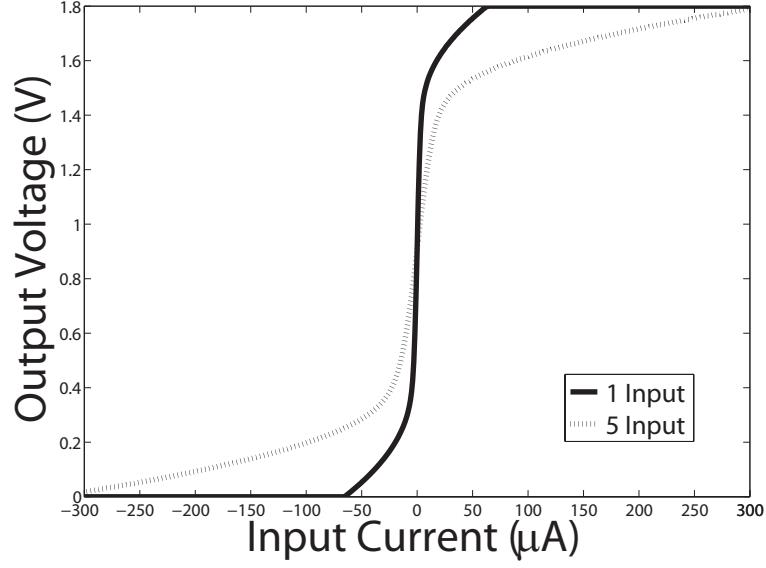


Figure 2.5: Scalability of the proposed DNN for a sigmoidal activation function.

The training result is shown in Figure 2.7. Random sets of weights are selected for initialization. The error from the ideal value is less than 0.05 (5.0%) for all input combinations.

### 2.3 CVNS Fully Distributed Neural network

Based on the proposed CVNS DNN, a Fully Distributed Neural Network (FDNN) is introduced. In this fully distributed CVNS network, the number of neurons depends not only on the number of inputs,  $k + 1$ , but also on the number of the CVNS digits in each CVNS digit set of weights,  $n + 1$ .

The block diagram of the proposed CVNS FDNN with  $(k + 1)$ -input is shown in Figure 2.8.

The output is  $Y_l = f(y)$ , where  $y$  is calculated using the following equation.

$$y = \frac{1}{(k + 1)(n + 1)} \left\{ ((y))_n|_l + \frac{((y))_{n-1}|_l}{B} + \dots + \frac{((y))_0|_l}{B^n} + Cons \right\} \quad (2.11)$$

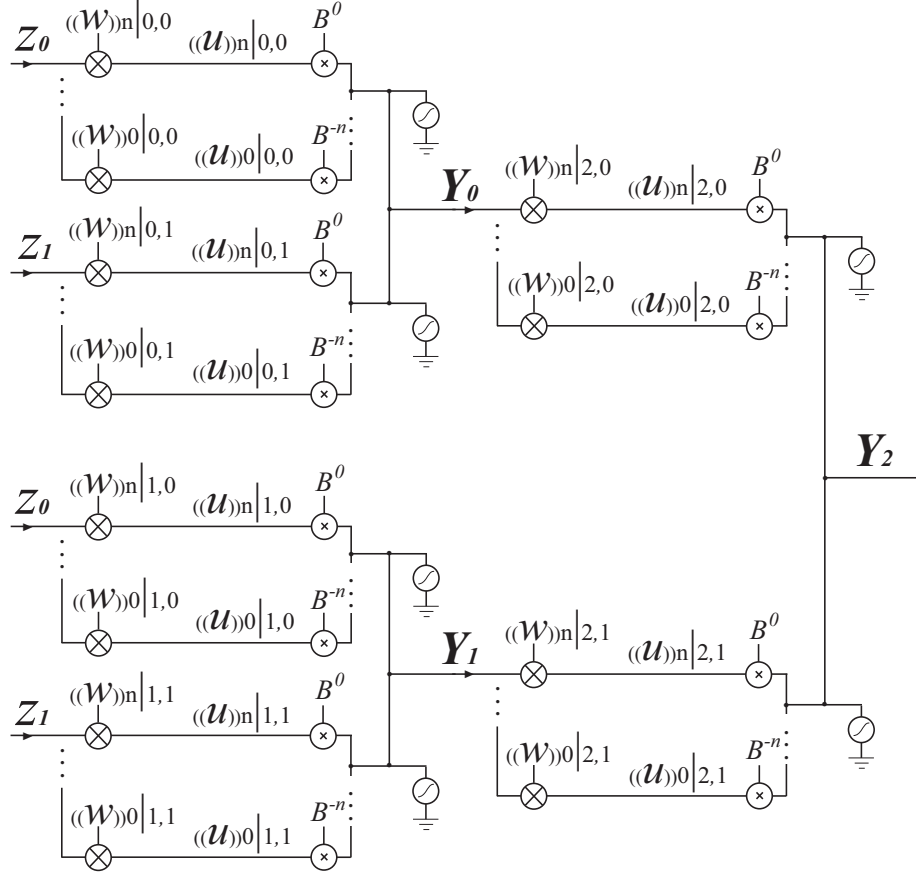


Figure 2.6: Block diagram of the 2 – 2 – 1 XOR network.

Although this network benefits from the feature of scalability, the number of neurons in this design is more than that of the CVNS DNN. Adding one input to the network will result in a  $n + 1$  increase in the number of neurons. Sensitivity of the proposed structures to the noise is studied by calculating the NSR in the next section.

## 2.4 Stochastic Model

A stochastic model based on the one presented in [11] is used in this section to find the NSR of different sigmoidal Adalines and to compare them together.

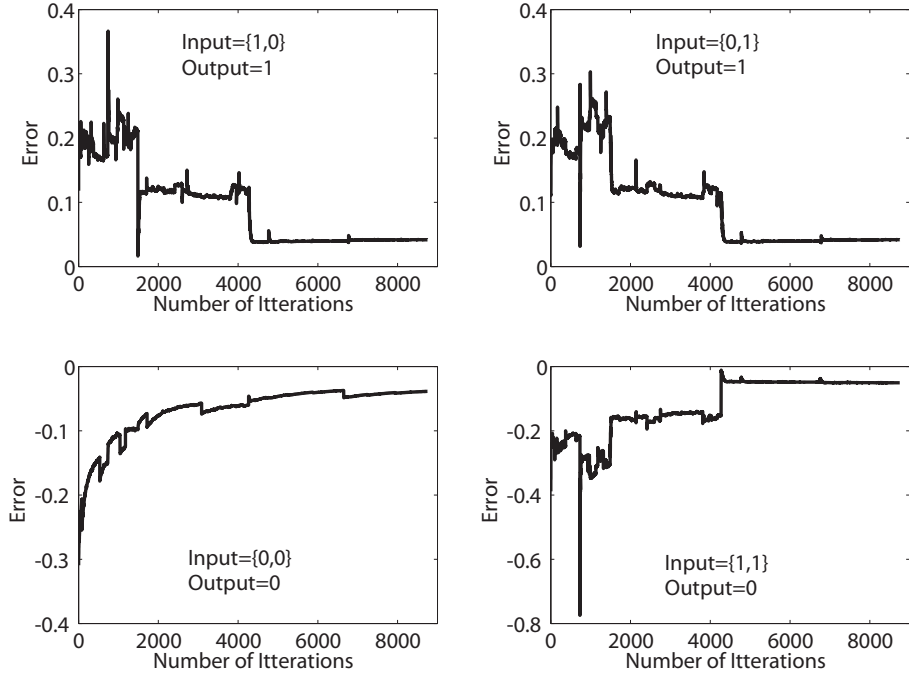


Figure 2.7: Off-line network training by back propagation algorithm.

According to this model, NSR of a  $(k+1)$ -input sigmoidal Adaline with lumped neuron is as follows [11].

$$NSR_{Lumped} = g(\sqrt{k+1}\sigma_Z\sigma_w) \times \left( \frac{\sigma_{\Delta Z}^2}{\sigma_Z^2} + \frac{\sigma_{\Delta w}^2}{\sigma_w^2} \right) \quad (2.12)$$

where  $g$  is the stochastic gain function,  $\Delta Z$  and  $\Delta w$  show input error and weight error, respectively.  $\sigma$  stands for the standard deviation, and  $\sigma^2$  is the variance.

In a DNN, the effect of each weight is divided by the number of inputs:  $\sigma_{w_{DNN}} = \frac{\sigma_w}{k+1}$ ,  $\sigma_{w_{DNN}}^2 = \frac{\sigma_w^2}{(k+1)^2}$ , and  $\Delta w_{DNN} = \frac{\Delta w}{k+1}$ . Therefore, NSR of a sigmoidal Adaline with distributed neuron is calculated as follows [9].

$$\begin{aligned} NSR_{DNN} &= g\left(\sqrt{k+1}\sigma_Z\frac{\sigma_w}{k+1}\right) \times \left( \frac{\sigma_{\Delta Z}^2}{\sigma_Z^2} + \frac{\sigma_{\Delta w}^2/(k+1)^2}{\sigma_w^2/(k+1)^2} \right) \\ &= g\left(\frac{\sigma_Z\sigma_w}{\sqrt{k+1}}\right) \times \left( \frac{\sigma_{\Delta Z}^2}{\sigma_Z^2} + \frac{\sigma_{\Delta w}^2}{\sigma_w^2} \right) \end{aligned} \quad (2.13)$$

In a CVNS network, weights are CVNS digits with digit level redundancy. An error in

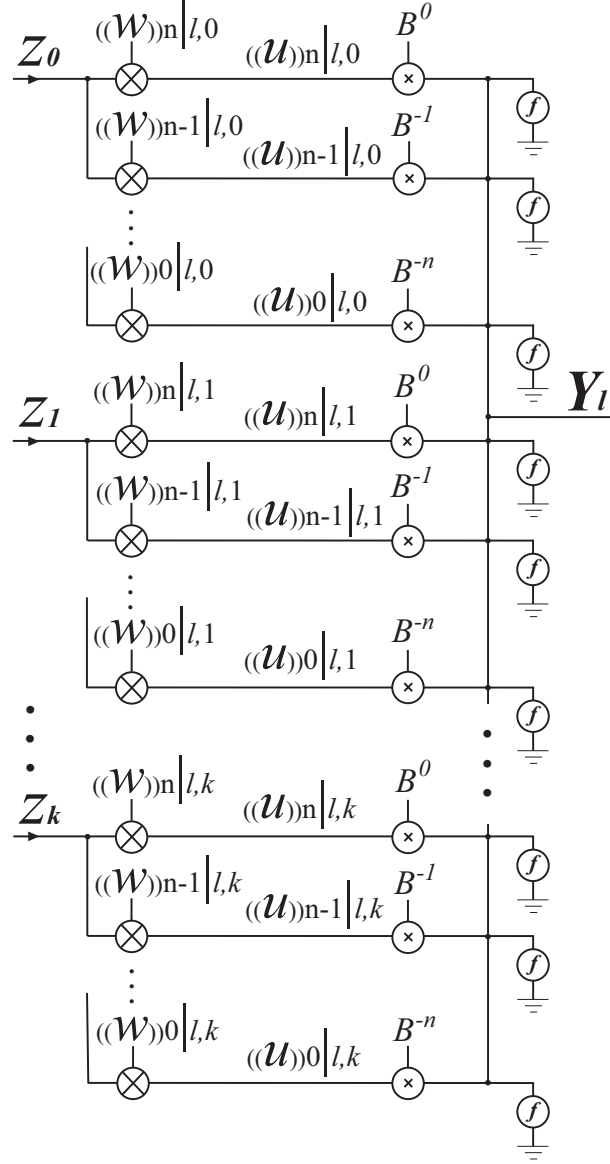


Figure 2.8:  $(k + 1)$ -input resistive-type CVNS fully distributed neural network.

a CVNS digit is corrected in the CVNS generation module except for the error in the digit with the lowest index [13]. Therefore, the only weight vulnerable to error is  $((w))_0$  which has an effect of  $((w))_n/B^n$ .  $((w))_n$  has the highest level of information in the CVNS digit set.

Accordingly, equation (2.14) is proposed to calculate NSR of a sigmoidal Adaline with a CVNS DNN structure. Note that the effect of each weight is still divided by the number of inputs because of its distributed nature.

$$NSR_{CVNS-DNN} = g\left(\frac{\sigma_Z \sigma_{((w))_n}}{B^n \sqrt{k+1}}\right) \times \left(\frac{\sigma_{\Delta Z}^2}{\sigma_Z^2} + \frac{\sigma_{\Delta((w))_n}^2}{\sigma_{((w))_n}^2}\right) \quad (2.14)$$

The NSR of a sigmoidal Adaline with a CVNS fully distributed structure is similar to the NSR of a sigmoidal Adaline with the CVNS DNN structure except for the fact that effect of each weight is also decreased by the number of CVNS digits.

$$NSR_{CVNS-FDNN} = g\left(\frac{\sigma_Z \sigma_{((w))_n}}{B^n \sqrt{k+1}(n+1)}\right) \times \left(\frac{\sigma_{\Delta Z}^2}{\sigma_Z^2} + \frac{\sigma_{\Delta((w))_n}^2}{\sigma_{((w))_n}^2}\right) \quad (2.15)$$

*Case study 2:* For a sigmoidal Adaline with  $k+1 = 9$ , considering both inputs and weights uniformly distributed in the range of  $[-10, 10]$ , input and weight variances will be  $\sigma_Z^2 = \sigma_w^2 = \sigma_{((w))_n}^2 = \frac{20^2}{12}$ . Weight error variance for a 12-bit quantization scheme is  $\sigma_{\Delta w}^2 = \sigma_{\Delta((w))_n}^2 = \frac{(20/2^{12})^2}{12}$ . Note that  $20/2^{12}$  is the equivalent space between levels. Input error variance,  $\sigma_{\Delta Z}^2$  is considered to be zero as the aim is to study the effect of noise introduced to weights on the networks.

NSR of the four considered sigmoidal Adalines is calculated as shown in Table 2.1. For the CVNS networks,  $B$  and  $n+1$  are assumed to be 2 and 3, respectively.

Table 2.1: NSR of sigmoidal Adalines from *case study 2*

Structure	NSR	Improvement
Lumped ANN	$3.21 \times 10^{-6} = -54.94dB$	—
DNN	$3.84 \times 10^{-7} = -64.16dB$	16.78%
CVNS DNN	$1.2 \times 10^{-7} = -69.2dB$	25.95%
CVNS FDNN	$6 \times 10^{-8} = -72.22dB$	31.45%

According to equations (2.12), (2.13), (2.14), and (2.15), as the number of inputs

changes, stochastic gain function will change. Stochastic gain function,  $g$ , of all four networks are compared together for an input range of  $[9, 25]$ . The comparison result is shown in Figure 2.9 where  $\sigma_Z\sigma_w = \sigma_Z\sigma_{((w))_n}$  is a constant value,  $B = 2$ , and  $n + 1 = 3$ . It should be noted that  $g(X)$  is almost equal to 1 for  $X < 2$  and can be estimated by the following equation for  $X > 2$  [9].

$$g(X) = 0.5 + 0.53 \times X \quad (2.16)$$

From Figure 2.9(a), stochastic gain function of the DNN, Proposed 1 (CVNS DNN), and Proposed 2 (CVNS FDNN) are almost half, one forth, and one twelfth of that of the lumped ANN, respectively. Stochastic gain function of the lumped neuron is greatest among all, and it increases by the number of inputs. Larger stochastic gain function will result in a larger NSR, Figure 2.9(b). Lumped neurons should be redesigned as the number of inputs increases to keep the NSR lower. The other three networks, because of their distributed neurons, can cope with an increase in the number of inputs which will also decrease the NSR gradually.

The NSR of a sigmoidal Madeline is calculated similarly to that of a sigmoidal Adaline by considering the effect of NSR of each layer on the next layer. Figure 2.9(c) and Figure 2.9(d) compare the NSR of Madalines with five and ten sigmoidal Adalines of *Case study 2*, respectively. Although the NSR of each layer is slightly bigger than the NSR of the previous layer, the proposed structures result in a smaller total NSR for the Madaline.

The number of CVNS digits is another variable in the proposed networks which can decrease the NSR. By increasing the number of CVNS digits,  $B^n$  becomes larger resulting in a smaller stochastic gain function. In the fully distributed neural network, increasing the number of CVNS digits also increases the  $n + 1$  factor, which decreases the NSR even more; however, it causes an increase in the number of sub-neurons.

According to Figure 2.9, practical implementation of the lumped neural network looks impossible as the number of layers increases. However, the distributed configurations still look promising for hardware implementations of networks with up to 10 layers.



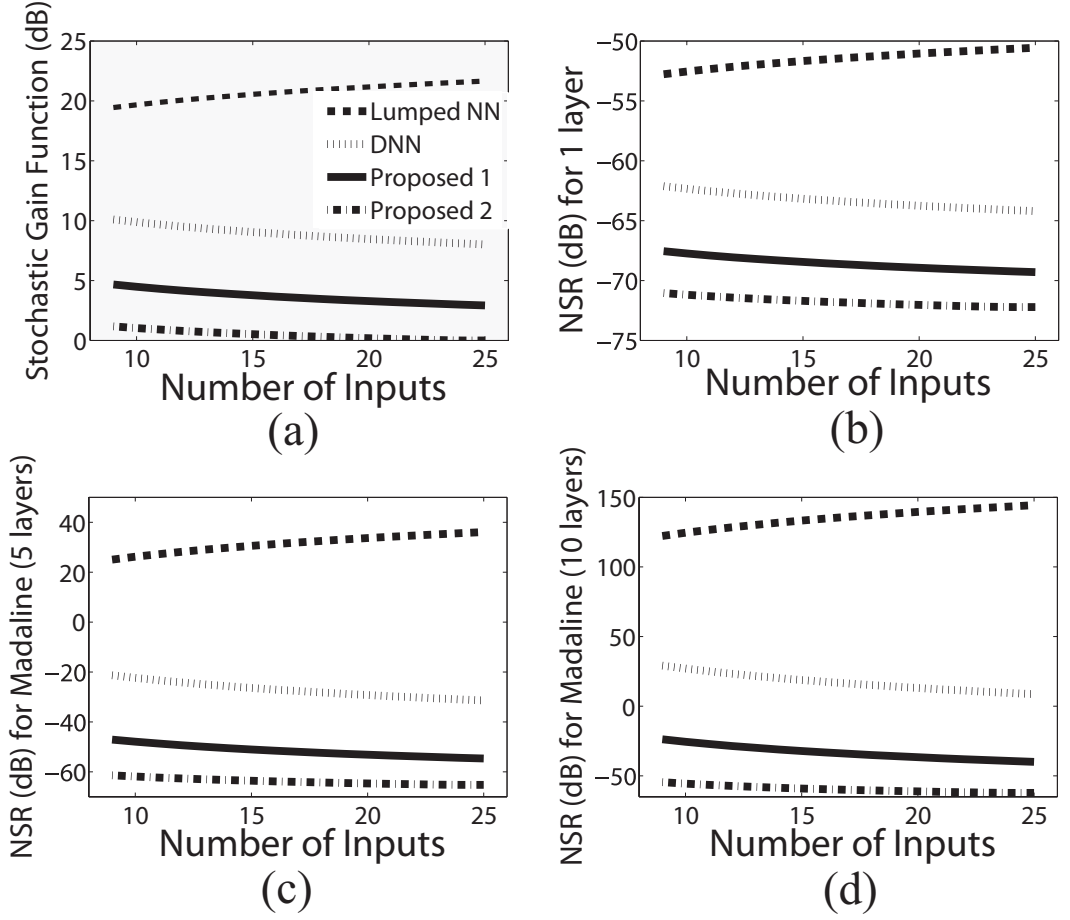


Figure 2.9: Comparison for input range of [9, 25]: (a) Stochastic Gain Function for Adaline, NSR for (b) Adaline, (c) 5-layer Madaline, (d) 10-layer Madaline.

## 2.5 Comparison

Applying equation (2.16) in the NSR calculation of lumped sigmoidal Adaline, equation (2.12), the NSR of the lumped sigmoidal Adaline can be rewritten as follows.

$$\begin{aligned}
 NSR_{Lumped} &= (0.5 + 0.53\sqrt{k+1}\sigma_Z\sigma_w) \times \left( \frac{\sigma_{\Delta Z}^2}{\sigma_Z^2} + \frac{\sigma_{\Delta w}^2}{\sigma_w^2} \right) \\
 &= (0.5 + 0.53R)D \approx RD
 \end{aligned} \tag{2.17}$$

where  $R$  and  $D$  stand for  $\sqrt{k+1}\sigma_Z\sigma_w$  and  $\frac{\sigma_{\Delta Z}^2}{\sigma_Z^2} + \frac{\sigma_{\Delta w}^2}{\sigma_w^2}$ , respectively. This approximation is correct specially for a large number of inputs.

Accordingly, the NSR of the other three sigmoidal Adalines are approximated in Table 2.2. The number of CVNS digits is equal to  $n+1$  when applicable.

Table 2.2: Comparison of different sigmoidal Adalines for a  $k+1$  input

<b>NN Structure</b>	<b>Number of Neurons</b>	<b>NSR</b>	<b>Neuron <math>\times</math> NSR</b>
<b>Lumped ANN</b>	1	$RD$	$RD$
<b>DNN</b>	$k+1$	$\frac{RD}{k+1}$	$RD$
<b>CVNS DNN</b>	$k+1$	$\frac{RD}{B^n(k+1)}$	$\frac{RD}{B^n}$
<b>CVNS FDNN</b>	$(k+1)(n+1)$	$\frac{RD}{B^n(k+1)(n+1)}$	$\frac{RD}{B^n}$

There is a tradeoff between the number of neurons and the value of NSR in distributed neural networks. To measure the overall efficiency of each network in Table 2.2, a comparison factor, Neuron  $\times$  NSR, is defined which is the result of multiplying the number of neurons by NSR for each structure. It is shown that the proposed networks outperform the other two networks due to their smaller Neuron  $\times$  NSR values.

## 2.6 Conclusion

Two novel CVNS Distributed Neural Networks are proposed. They have the self-scaling property which can potentially decrease the noise to signal ratio. The NSR of the proposed

structures are compared to those of lumped neural networks and conventional distributed neural networks. It is shown that significant improvements are attained specially as the number of inputs increases. The proposed CVNS DNN can defeat both lumped ANN and conventional DNN with its low NSR while it uses the same number of neurons as the latter one does.

---

## Chapter 3

---

# *Truncated CVNS Distributed Neural Network*

---

the CVNS has been used to build a new family of distributed neural networks [16] where it is used in a high resolution environment. The CVNS DNN was shown to provide a NSR lower than that of the conventional DNNs. However, practical implementation of the CVNS systems is limited by the resolution of the environment. For example, a CVNS digit which is extracted from 12 binary bits needs at least a reliable 12-bit resolution for hardware implementation, which is not the case in generally using technologies.

Truncation methods are applied to the CVNS digits to make them cope with lower resolution environments while still keeping the CVNS properties such as the redundancy between digits [14, 15]. One of the advantages of applying these methods is reducing the area consumption; however, it may result in lower accuracy as well.

Here, the CVNS DNN is studied according to the environmental considerations. The resolution is limited by the  $0.18\mu m$  CMOS technology characteristics. Studies are per-

formed on the accuracy and the NSR of the proposed truncated CVNS DNN for an environment with the reliable resolution of 4-bit. Studies show that the NSR is higher than the NSR of the complete CVNS DNNs, as expected; however, it is still less than that of the conventional DNNs.

### 3.1 Truncated CVNS DNN

The acceptable resolution for 0.18 $\mu$ m CMOS technology using a power supply voltage of 1.8V is considered to be 4-bit [15]. As a result, all arithmetic operations are assumed to be correct up to 4 bits, and additional values may need to be checked and corrected.

The first truncation is applied to the number of bits which are used to generate each CVNS digit. This number of bits is called group length and is selected as 4 for the sake of the practical resolution. In other words, in this work, each CVNS digit is generated from up to 4 bits.

Another flexible parameter in the CVNS digit sets is called digit link which is the number of common bits used to generate the CVNS digits. Keeping the digit link in its maximum means that the number of the CVNS digits and the binary digits are exactly the same when the radices are equal as is shown in equation (3.1); there will be 13 CVNS digits equivalent to 13 bits. This will increase the number of partial multiplications and additions and consequently the complexity of the system.

$$((w))_{n-j}|_{l,t} = \sum_{i=0}^{m-j} w_i \cdot 2^{i+j-m}, j = 0, 1, \dots, n \quad (3.1)$$

To overcome this problem, digit link is decreased to its minimum which is equal to 1. This reduces the redundancy as well as the noise immunity of the digits. However, a proper design of the CVNS generation module will suppress this problem; error correction is generally performed over digits in a CVNS generation module [13].

The following equation shows the CVNS digit generation from binary digits with group

---

length of 4 and digit link of 1 [15].

$$((w))_{n-j} = \sum_{i=m-j(4-1)-4+1}^{m-j(4-1)} w_i \cdot 2^{i+j(4-1)-m}, j = 0, 1, \dots, n \quad (3.2)$$

where  $w_i$  shows the  $i^{th}$  bit in binary representation.

There is a relation between the number of binary digits,  $m + 1$ , and the number of the CVNS digits,  $n + 1$ , as follows.

$$n + 1 = R\left(\frac{m + 1}{\text{group length} - \text{digit link}}\right) \quad (3.3)$$

where  $R$  is the classical Rounding function.

Using the CVNS digit set generated from equation (3.2) as the multiplicand, the CVNS multiplication equation is no longer applicable. This equation needs the maximum link between digits as well as the maximum group length to be adjusted.

A new method is proposed to perform the multiplication over the truncated CVNS digits. Here, the proposed multiplication method is particularized for a case that the multiplicand is a CVNS digit set with 4 CVNS digits,  $((x))_i$  ( $i = 3, \dots, 0$ ), and the multiplier has 4 bits,  $(Z)_i$  ( $i = 4, \dots, 1$ ). The radix of the CVNS system is equal to 2.

First, four partial results,  $y_i$  ( $i = 3, \dots, 0$ ), are generated.

$$\begin{aligned} y_3 &= ((w))_3 \times \sum_{i=4}^1 (Z)_i 2^{i-4} \\ y_2 &= (((w))_2 \text{mod } 1) \times \sum_{i=4}^1 (Z)_i 2^{i-4} \\ y_1 &= (((w))_1 \text{mod } 1) \times \sum_{i=4}^1 (Z)_i 2^{i-4} \\ y_0 &= (((w))_0 \text{mod } 1) \times \sum_{i=4}^1 (Z)_i 2^{i-4} \end{aligned} \quad (3.4)$$

The *mod*1 shows the modulo 1 operation and is applied as there is 1 digit link between the CVNS digits. This operation omits the effect of the first binary digit, digit link, in  $y_2$ ,

---

$y_1$ , and  $y_0$ ; therefore, it prevents the double effect of the common digit in the final result. It should be noted that finally the partial results are added up together to generate the final synapse output.

Note that the partial results are not in the CVNS form as there is no longer any overlap between them; they are just continuous values equivalent to 8 bits.

There is one total output for the synapse; thus, these partial results are scaled and added up together as follows.

$$y = ((y_0 \times 2^{-3} + y_1) \times 2^{-3} + y_2) \times 2^{-3} + y_3 \quad (3.5)$$

However, there are some practical limitations applied to equations (3.4) and (3.5) because of the resolution of the environment. These are discussed through the following case study.

*Case study 1:* The randomly chosen value for multiplicand is ‘0111, 1101, 0101, 1’, and the multiplier is chosen equal to ‘1110’. The CVNS digits for the multiplicand are extracted using equation (3.2): [0.875, 1.75, 0.625, 1.375]. The partial results are calculated through equation (3.4) as is shown in Table 3.1.

The first columns in Table 3.1 shows the partial results in a high resolution environment. The results in an environment with 4-bit resolution are shown in the second columns. They are rounded values of the previous column. Therefore, the classic Rounding function should be added for partial results extractions in equation (3.4) for hardware implementations.

The lower resolution environment also has an effect on equation (3.5). According to equation (3.5), the final output in the high resolution environment is found as shown in Figure 3.1 which is equal to ‘0110, 1101, 1010, 1101, 0’, 6.8547 in decimal.

In a 4-bit resolution environment, the classic Rounding function should be added for each scaling addition in equation (3.5). Consequently, each 7-bit result will be rounded to a 4-bit equivalent result and be scaled for the next addition.

Table 3.1: Multiplication partial results for *case study I*.

High Resolution		Low Resolution	
Partial Result	Equivalent Bits	Partial Result	Equivalent Bits
$y_3 = 1.53125$	0110,0010	$y_3 = 0.75$	0110
$y_2 = 1.3125$	0101,0100	$y_2 = 0.625$	0101
$y_1 = 1.09375$	0100,0110	$y_1 = 0.5$	0100
$y_0 = 0.65625$	0010,1010	$y_0 = 0.375$	0011

$$\begin{array}{r}
 01100010 \\
 + \quad 01010100 \\
 + \quad \quad 01000110 \\
 + \quad \quad \quad 00101010 \\
 \hline
 01101101101011010
 \end{array}$$

Figure 3.1: Final multiplication result in a high resolution environment.

Figure 3.2 shows the multiplication process in a 4-bit resolution environment. The result, ‘0111’ (7 in decimal), is the rounded value of the result in Figure 3.1.

From Figure 3.2, the effect of the lower index digits is contributed to the highest index digits. Without this, the output would be only calculated from the first CVNS digit resulting in ‘0110’ (6 in decimal) which is far from the right answer.

The proposed truncated CVNS DNN is used to build a 2 – 2 – 1 XOR with sigmoidal activation function. The simulation result is shown in Figure 3.3.





Subsequently, the probability of each partial result for effecting the final answer,  $P(y_i, y)$ , is roughly estimated as follows.

$$\begin{aligned}
 P(y_3, y) &= 1 = 100\% \\
 P(y_2, y) &= P(y_2, y_3) \times P(y_3, y) = 0.73 \times 1 = 0.73 = 73\% \\
 P(y_1, y) &= P(y_1, y_2) \times P(y_2, y) = 0.73 \times 0.73 = 0.53 = 53\% \\
 P(y_0, y) &= P(y_0, y_1) \times P(y_1, y) = 0.69 \times 0.53 = 0.37 = 37\%
 \end{aligned}$$

Each multiplicand CVNS digit is multiplied by the multiplier generating a partial result equivalent to 8-bit rounded to 4-bit. The minimum value for the 8-bit result to be considered in the rounded result is equivalent to ‘0000, 1001’. Thus, the probability of the first CVNS digit not to affect the 4-bit partial result is as follows.

$$\begin{aligned}
 \overline{P(((w))_3, y_3)} &= P(0000, Z) + P(0000, ((w))_3) \\
 &+ P(0001, Z) \times [P(0001, ((w))_3) + P(0010, ((w))_3) \\
 &\quad + P(0011, ((w))_3) + P(0100, ((w))_3)] \\
 &+ P(0010, Z) \times [P(0001, ((w))_3) + P(0010, ((w))_3)] \\
 &+ P(0011, Z) \times P(0001, ((w))_3) \\
 &+ P(0100, Z) \times P(0001, ((w))_3) \\
 &= \frac{1}{16} + \frac{1}{16} + \frac{1}{16} \times [\frac{4}{16}] + \frac{1}{16} \times [\frac{2}{16}] + \frac{1}{16} \times [\frac{1}{16}] \\
 &+ \frac{1}{16} \times [\frac{1}{16}] = 0.16 = 16\% \tag{3.6}
 \end{aligned}$$

However, this is only correct for the first CVNS digit as there is a modulo operation for the others. This operation deletes the effect of the first bit making each CVNS digit equivalent to 3 bits. Therefore, all the  $\frac{1}{16}$  in the previous equation corresponding to the CVNS digits, should be changed to  $\frac{1}{8}$  for all the CVNS digits except for the first one. In this case,  $\overline{P(((w))_i, y_i)}$  for  $i = 2, 1, 0$  would be equal to 25%.

According to equation (3.6), the probability of the first CVNS digit to affect the relevant partial result is  $P(((w))_3, y_3) = 1 - 0.14 = 0.86 = 86\%$  while this probability is equal to 75%

---

for the other CVNS digits. The probability of each CVNS digit to effect the final result is estimated through the following equations.

$$\begin{aligned}
 P(((w))_3, y) &= P(((w))_3, y_3) \times P(y_3, y) = 86\% \\
 P(((w))_2, y) &= P(((w))_2, y_2) \times P(y_2, y) = 55\% \\
 P(((w))_1, y) &= P(((w))_1, y_1) \times P(y_1, y) = 40\% \\
 P(((w))_0, y) &= P(((w))_0, y_0) \times P(y_0, y) = 28\%
 \end{aligned}$$

Consequently, the CVNS implementation in a low resolution environment provides a considerable probability to a high resolution value aiding the production of an accurate result.

### 3.3 NSR Calculation

Noise to signal ratio is a very important feature in neural networks which shows their immunity to noise and dictates the required resolution for synaptic weight implementations. Lower NSR makes the network less vulnerable to errors and violations and makes it more potential for hardware implementations.

The NSR of the proposed truncated CVNS DNN is calculated here for an Adaline. It is compared to Adalines with lumped, distributed, and complete CVNS distributed structures.

The NSR of a  $(k + 1)$ -input sigmoidal Adaline with lumped structure is calculated through a model based on the stochastic gain function,  $g$ , proposed by Piche [11].

$$NSR_{Lumped} = g(\sqrt{k + 1}\sigma_Z\sigma_w) \times \left( \frac{\sigma_{\Delta Z}^2}{\sigma_Z^2} + \frac{\sigma_{\Delta w}^2}{\sigma_w^2} \right) \quad (3.7)$$

where  $\Delta Z$  shows input error and  $\Delta w$  stands for weight error.  $\sigma$  is the standard deviation, and  $\sigma^2$  is the variance.

Sub-neurons in a DNN cause a dividing in the effect of each weight:  $\sigma_{w_{DNN}} = \frac{\sigma_w}{k+1}$ ,  $\sigma_{w_{DNN}}^2 = \frac{\sigma_w^2}{(k+1)^2}$ , and  $\Delta w_{DNN} = \frac{\Delta w}{k+1}$ . Therefore, NSR of a sigmoidal Adaline with distributed

structure is calculated as follows.

$$NSR_{DNN} = g\left(\frac{\sigma_Z \sigma_w}{\sqrt{k+1}}\right) \times \left(\frac{\sigma_{\Delta Z}^2}{\sigma_Z^2} + \frac{\sigma_{\Delta w}^2}{\sigma_w^2}\right) \quad (3.8)$$

In a CVNS digit generation module, the error in a CVNS digit is corrected using the redundancy between the digits. However, the lowest index digit is not subject to this error correction, and it is the only digit which might be corrupted with errors [13].  $((w))_0$ , as the lowest index digit, has an effect of  $((w))_n/B^n$  on the system where  $((w))_n$  is the digit with the highest level of information in the CVNS digit set. Note that  $((w))_0$  should be multiplied by  $B^n$  to be in the same order as  $((w))_n$  is.

Accordingly, the NSR of a sigmoidal Adaline with a complete CVNS DNN structure is calculated as follows.

$$NSR_{CVNS-DNN} = g\left(\frac{\sigma_Z \sigma_{((w))_n}}{B^n \sqrt{k+1}}\right) \times \left(\frac{\sigma_{\Delta Z}^2}{\sigma_Z^2} + \frac{\sigma_{\Delta((w))_n}^2}{\sigma_{((w))_n}^2}\right) \quad (3.9)$$

In the case of the proposed truncated CVNS DNN, the effect of each weight is still divided by the number of inputs due to the distribution of sub-neurons. The CVNS digits are also generated in the modules with error correction capabilities. However, the lowest index CVNS digit in a truncated digit set, has an effect of  $((w))_{nn}/B^{nn}$ . Here, the highest index is  $nn$  where  $nn+1 = R(\frac{n+1}{group\ length-digit\ link})$  which is equal to  $R(\frac{n+1}{3})$  with group length of 4 and digit link of 1. The NSR will be calculated through the following equation.

$$NSR_{Truncated} = g\left(\frac{\sigma_Z \sigma_{((w))_{nn}}}{B^{nn} \sqrt{k+1}}\right) \times \left(\frac{\sigma_{\Delta Z}^2}{\sigma_Z^2} + \frac{\sigma_{\Delta((w))_{nn}}^2}{\sigma_{((w))_{nn}}^2}\right) \quad (3.10)$$

The NSR of the proposed truncated network is compared to that of other structures through the following case study.

*Case study 2:* NSR of all configurations are compared together for an input range of  $[1, 25]$ , constant value for  $\sigma_Z \sigma_w = \sigma_Z \sigma_{((w))_n}$ ,  $B = 2$ , and  $n+1 = 13$  resulting in  $nn+1 = 4$ . The NSR is calculated for Adalines and 5-layer Madalines as is shown in Figure 3.4. In Madalines, NSR of each layer has an effect on NSR of the next layer.

---

As is shown in in Figure 3.4, the NSR in lumped neural network is increasing as the number of inputs increases, and it is always less than the NSR of other structures. The other distributed structures are facing a decrease in NSR for higher number of inputs. The truncated CVNS DNN has an NSR larger than that of the complete CVNS DNN; however, its NSR is always less than the NSR of both lumped and conventional distributed neural networks. It should be noted that the number of interconnections and the CVNS weights in the truncated network are less than those numbers in the complete CVNS network. This fact will result in less area consumption for the truncated CVNS DNN compared to the complete one. Furthermore, it is completely practical for hardware implementations in an environment with low resolution of 4-bit. However, it should be noted that as the number of layers increases, the hardware implementation of the truncated structure looks impossible because of the high value of the NSR.

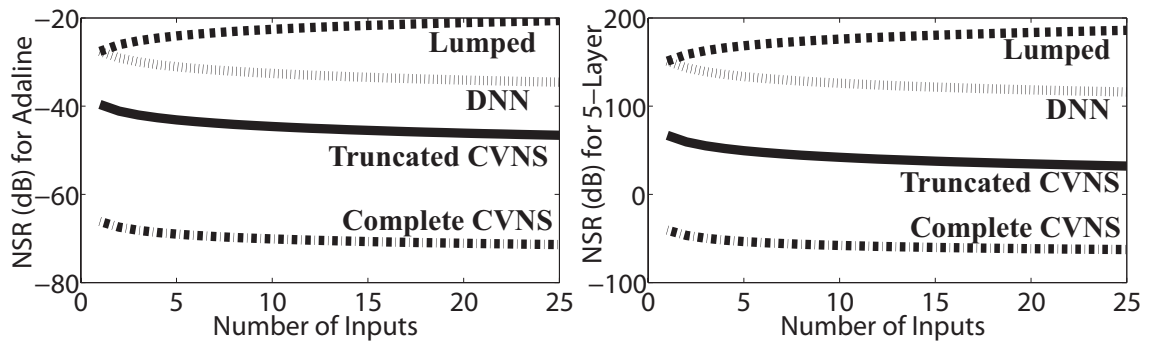


Figure 3.4: NSR comparisons for Adalines and 5-layer Madalines.

### 3.4 Conclusion

The truncated CVNS distributed neural network is proposed which is adaptable for implementation in a low resolution environment. The proposed network is simulated in  $0.18\mu m$  CMOS technology with 4-bit resolution where it shows a proper functionality. The NSR is calculated for Adaline and Madaline with the proposed truncated CVNS structures. The

NSR calculations show that the truncated CVNS network surpasses both the lumped and conventional distributed structures.

---

## Chapter 4

---

### *Synapse-Neuron Module*

---

ANNs usually have two main parts: synapses and neurons. A synapse receives the input and multiplies it by the corresponding stored weights. The result goes into the neuron where the output value is generated from the input based on the activation function. In a distributed neural network, there is one sub-neuron for each synapse. Therefore, the synapse and its corresponding neuron can be merged together resulting in a synapse-neuron module.

Using the synapse-neuron module will decrease the design cost and interconnections [2]. Once designing the synapse-neuron module, it can be used as building block of neural networks where it needs setting up the interconnections between synapse-neuron modules. The block diagram of the proposed synapse-neuron module is shown in Figure 4.1.

A 13-bit multi-valued CVNS DRAM is used to store the weights,  $w_{12}, \dots, w_0$ . The DRAM generates the CVNS digits,  $((w))_3, \dots, ((w))_0$ , from the binary weights and stores them on the storage cells. The stored CVNS digits proceed to the multiplication module.

Multiplication module performs the multiplication of the stored CVNS digits by the input. The input is converted to 4 bits,  $(Z)_4, \dots, (Z)_1$ . Consequently, the CVNS digits are

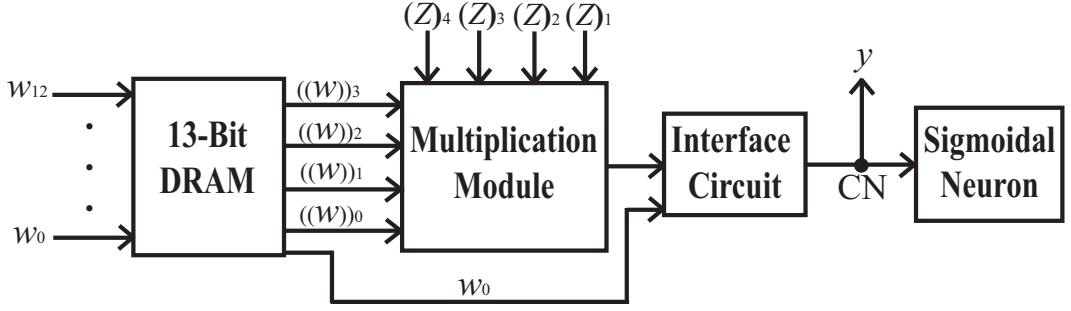


Figure 4.1: Block diagram of the proposed synapse-neuron module.

multiplied by the binary bits resulting in an analog value with 4-bit resolution.

The interface unit applies the sign,  $w_0$ , to the multiplication output and transfers the result to the common node. In this node, the currents from all the synapse-neuron modules are added up together and divided by the number of modules. A sigmoidal neuron is employed to extract the output according to the received current in the Common Node (CN).

There is also a voltage-mode ADC to convert the input to each layer to 4 bits. Note that as the input to the network is digital, there is no need of using an ADC for the inputs to the first layer.  $(Z)_4$  is the only digit which is changing from ‘0’ to ‘1’, and the other three digits are considered to be equal to ‘0’. Accordingly, The maximum value for the input happens for an input equal to  $1.8V$  which is equivalent to ‘1000’.

This synapse-neuron module is the building block in implementing distributed neural networks.

## 4.1 CVNS DRAM

The 13-bit DRAM contains four 4-bit (16-level) multi-valued DRAMs [14, 15] as is shown in Figure 4.2.

Working in current-mode, each 16-level DRAM generates a CVNS digit in the form of



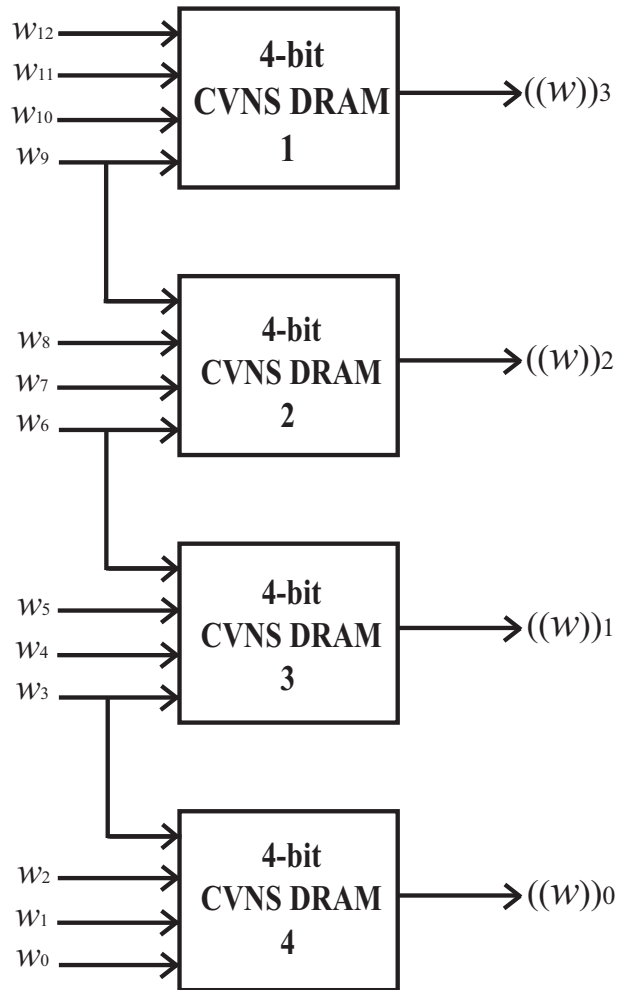


Figure 4.2: Block diagram of the 13-bit multi-valued DRAM.

current from the corresponding 4-bit input in the form of voltage.

There is a 4-bit Digital to Analog Converter (DAC) in each one of 4-bit DRAMs which is responsible for the CVNS digit generation. This DAC contains four 1-bit weighted current sources. In this work, the current sources of DAC are designed to generate currents equal to  $8\mu A$ ,  $4\mu A$ ,  $2\mu A$ , and  $1\mu A$  for the first (most significant), the second, the third, and the fourth (least significant) bits, respectively. The output currents from the current sources are summed up together, and the result generates a CVNS digit in the form of a current

changing between 0 to  $15\mu A$ .

Each CVNS digit is stored on a dynamic current mirror (current copier) [27]. Each storage cell has the ability of storing up to 4 bits equivalent to 16 levels with the capacitor size of  $20.3fF$ .

There is a link of 1-bit between the CVNS digits; the Least Significant Bit (LSB) input to each 4-bit DRAM is repeated as the Most Significant Bit (MSB) to the next 4-bit DRAM as is shown in Figure 4.2. This common bit is used as the Error Correction Code (ECC).

The value of stored CVNS digits changes with the time because of the leakage current of the storage capacitor. This changes the stored value and causes an error. In each refreshing cycle, the possible error is detected and corrected based on the redundancy between the stored CVNS digits, ECC.

Figure 4.3 shows the layout of the 13-bit multi-valued CVNS DRAM. The capacitors are metal-insulator-metal capacitors and are laid out using metal 5 and metal 6.

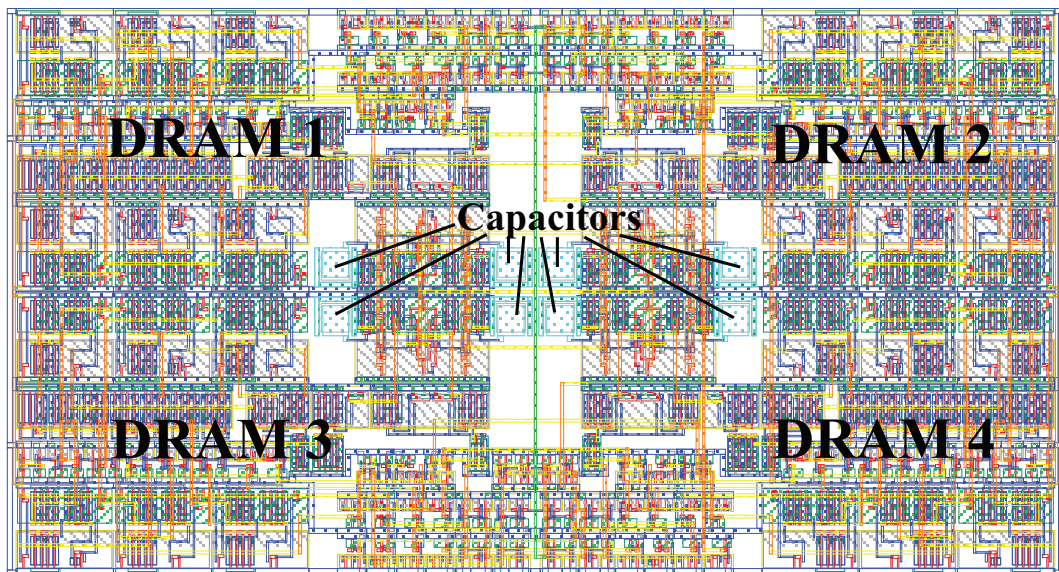


Figure 4.3: Layout of the 13-bit multi-valued DRAM.

## 4.2 Multiplier Module

The output of this module is in the form of current and is generated through the block diagram of Figure 4.4.

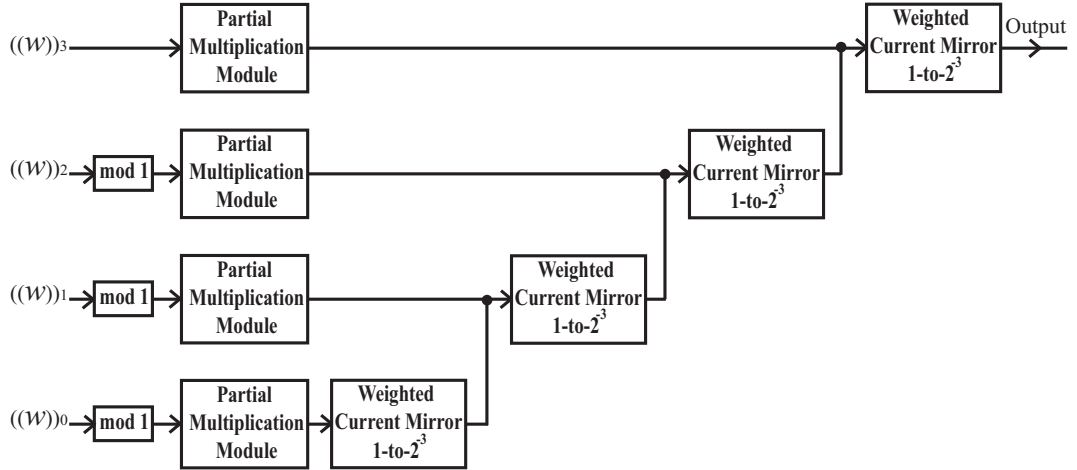


Figure 4.4: Block diagram of the multiplier module.

In Figure 4.4, *mod1* is the modulo 1 operation module. Partial multiplication modules are generating the partial results as is shown in Figure 4.5. Weighted current mirrors are used for scaling purposes.

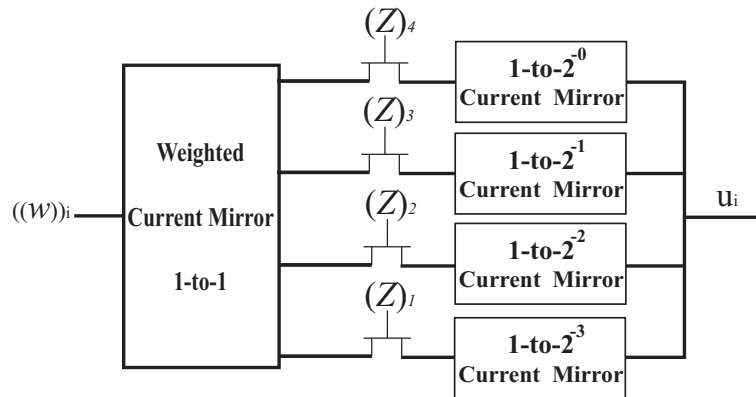


Figure 4.5: Block diagram of the partial multiplication module.

Here, the multiplier digits are applied as switches. For a digit of ‘1’, the switch is close, and the CVNS digit in the form of current is passing through it.

The simulation result for multiplication of ‘1, 1111, 1010, 1011’ by ‘0111’ in the proposed circuitries for the multiplier module is shown in Figure 4.6.

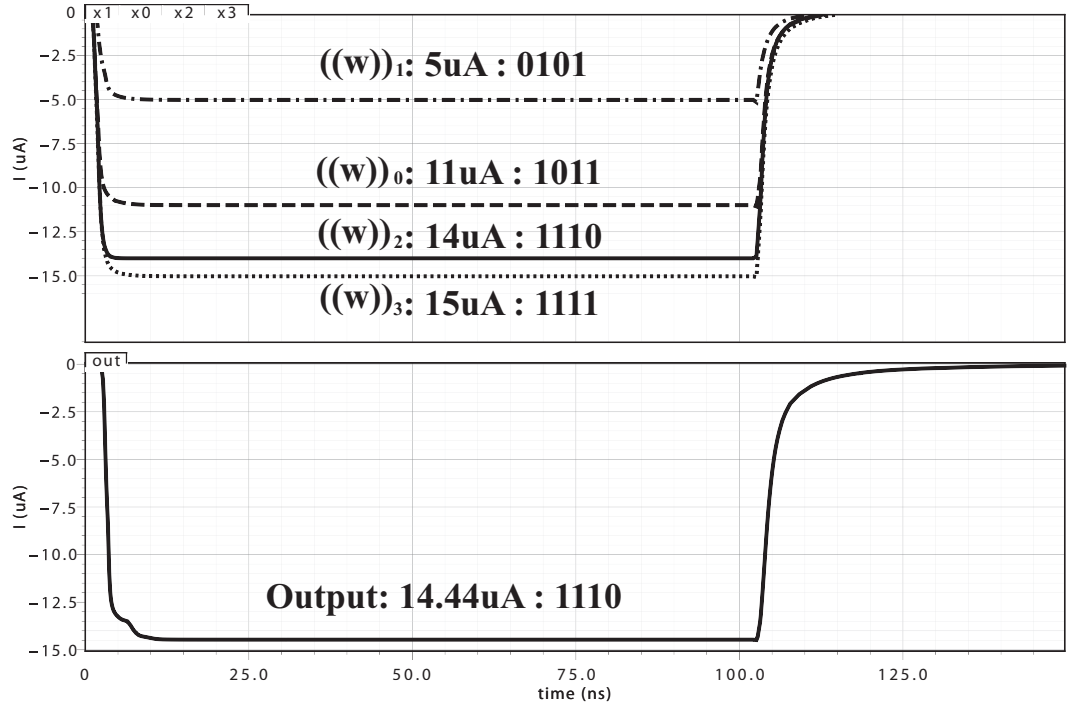


Figure 4.6: Simulation result of ‘1, 1111, 1010, 1011’ multiplied by ‘0111’.

The layout of the multiplier is shown in Figure 4.7.

### 4.3 Interface Circuit

The schematic of the interface circuit is shown in Figure 4.8. This circuit, applies the sign,  $w_0$ , to its input current and sends the result to the CN. If  $w_0$  is one, the output current is negative and flows away from the CN. For  $w_0$  equal to zero, on the other hand, the output current is positive and flows to the CN.

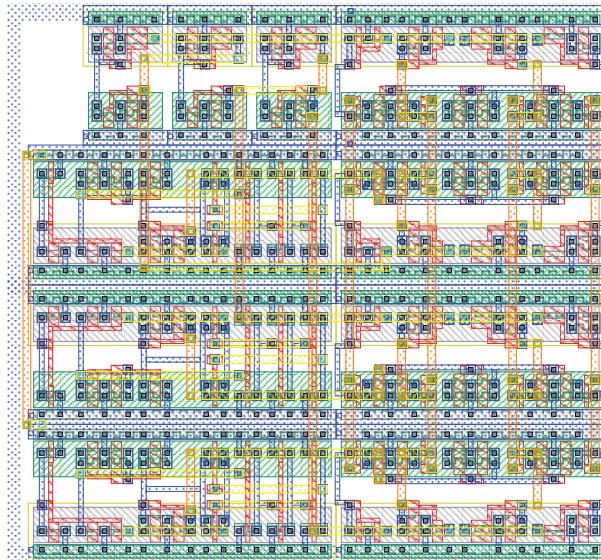


Figure 4.7: Layout of the multiplication module.

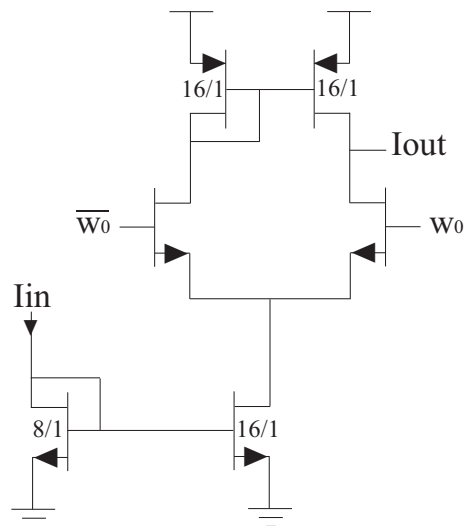


Figure 4.8: Schematic of the interface circuit.

In addition, the interface circuit scales the input current to the input range of the sigmoidal neuron.

## 4.4 Sigmoidal Neuron

The resistive-type neuron is designed using 6 transistors [28]. NMOS and PMOS characteristics in triode and saturation regions are used to realize the sigmoid function as the activation function of the network. The schematic of the neuron is shown in Figure 4.9.

Transistors M5 and M6 have their gates connected to their drains and are used for biasing. They are providing the same biasing voltage,  $V_B$ , as they are connected together. These two transistors are sized so that  $V_B = \frac{V_{dd}-V_{ss}}{2}$  which in this case is equal to  $0.9V$ .

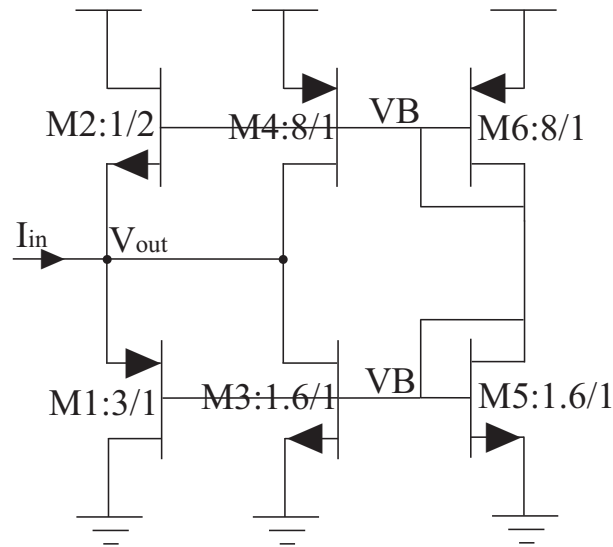


Figure 4.9: Schematic of the proposed resistive-type neuron to realize the sigmoid function.

The sigmoidal neuron is working for a range of currents between  $-60\mu A$  and  $60\mu A$  and in three operating regions: 0 to  $(V_B - V_{tn})$ ,  $(V_B - V_{tn})$  to  $(V_B + |V_{tp}|)$ , and  $(V_B + |V_{tp}|)$  to  $V_{dd}$ . The simulation result of the resistive-type sigmoidal neuron is shown in Figure 4.10.

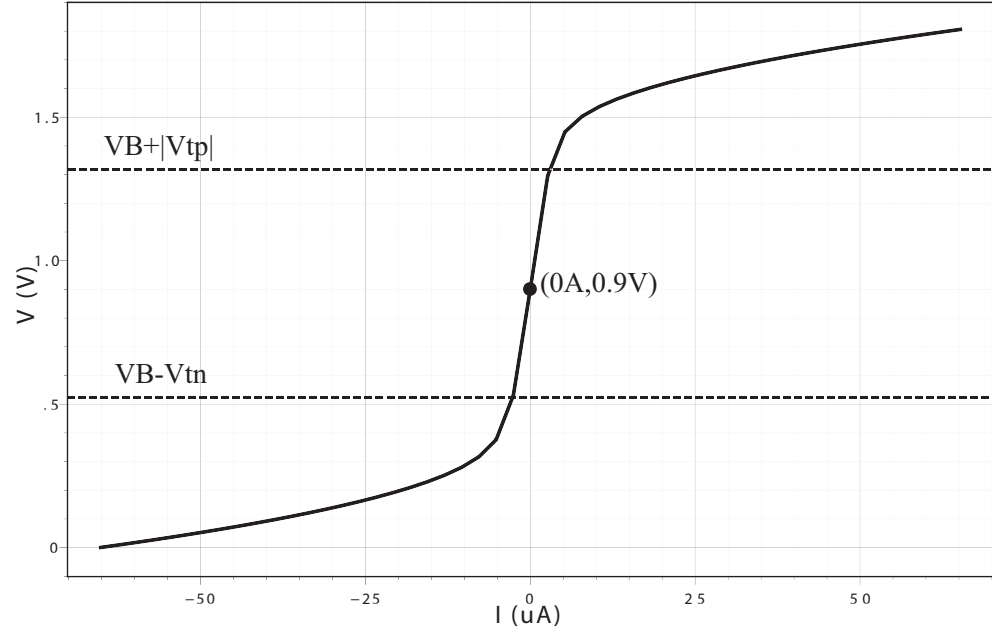


Figure 4.10: Simulation result of the sigmoidal neuron.

## 4.5 Voltage-Mode ADC

An ADC is used to convert the output of each hidden layer which is a voltage between 0 and  $1.8V$  for a 4-bit value. These four bits act as the multiplier set  $((Z)_4, (Z)_3, (Z)_2,$  and  $(Z)_1)$  for the next layer (Figure 4.1). Considering that ‘1000’ stands for the voltage of  $1.8V$ ,  $\frac{1.8-0}{8} = 0.225V$  shows the equivalent voltage spacing between levels where there are 8 levels between ‘0000’ and ‘0111’.

This voltage-mode ADC is designed based on the CMOS inverter characteristic. The input to the ADC goes to 8 set of inverters. The schematic of an inverter set is shown in Figure 4.11. The outputs,  $A_i$  ( $i = 1, \dots, 8$ ), are either ‘1’ or ‘0’ and move into an encoder which generates an equivalent 4-bit output. These 4 bits are inputs to the next layer,  $(Z)_i$  ( $i = 4, \dots, 1$ ).

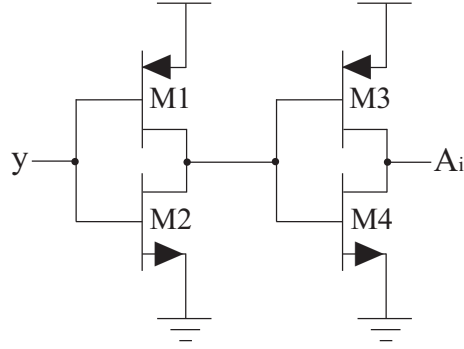


Figure 4.11: schematic of a set of inverters used in the proposed voltage-mode ADC.

Based on the CMOS inverter characteristics,  $V_{IL}$  is the lowest value of the input voltage which generates a high output voltage. The proposed design is based on changing the  $W/L$  ratio of NMOS/PMOS transistors to alter the  $V_{IL}$  of the inverters. In Figure 4.11, a higher value for  $L$  of  $M1$ , as well as a higher value for  $W$  of  $M2$ , results in a lower  $V_{IL}$ .

Each set of the 8 inverters is designed with a specific value for  $V_{IL}$  to cover a voltage level from 0 to  $1.8V$ , and the difference between each  $V_{IL}$  and the next one is equal to  $0.225V$ .

The encoder generates each bit using XOR gates as follows.

$$(Z)_4 = A_8$$

$$(Z)_3 = A_4 \oplus A_8$$

$$(Z)_2 = A_2 \oplus A_4 \oplus A_6 \oplus A_8$$

$$(Z)_1 = A_1 \oplus A_2 \oplus A_3 \oplus A_4 \oplus A_5 \oplus A_6 \oplus A_7 \oplus A_8$$

The proposed voltage-mode ADC is laid out in CMOS  $0.18\mu m$  technology with the total area of  $491.55\mu m^2$ , and its power consumption is  $7.52\mu w$ . These values are at least 1000 times smaller than those of the ADCs in literature [29–31]. Although, the working frequency in the proposed ADC is much smaller than that of the state-of-the-art, speed of the ADC is not a bottleneck in this application. Therefore, a small and low power design is preferred.



## 4.6 Conclusion

A synapse-neuron module based on the resolution of the environment, 4-bit, is proposed. In this module, CVNS multi-valued memories are used to generate the truncated CVNS digits from the weights and store them; in addition, these memories have got ECCs to correct the possible error in the CVNS digits. A multiplier multiplies the weights in the form of the CVNS digits by the input in the form of a 4-bit value. Analog sigmoidal neurons are used to realize the activation function. By properly arranging the proposed synapse-neuron modules, they can be used to build any neural network.

---

## Chapter 5

---

### *CVNS Multi-Valued Dynamic Memory*

---

One important aim in any memory circuit design is to properly store more data on a smaller area which results in a highly dense memory. In general, multi-valued DRAMs, with the ability of storing more than one bit per storage cell, have a main advantage over the conventional DRAMs, two-level DRAMs, which store each bit on one storage cell. Compared to a conventional DRAM, less area will be consumed to store information bits on a multi-valued memory as more than one bit can be stored on each capacitor; capacitors are usually the main area consumers in a memory design. Therefore, they provide higher bits-per-cell storage capacity in comparison to ordinary conventional DRAMs [1, 32–48]. Multi-valued DRAMs can be used in applications where a large number of values has to be stored on the chip such as neural network implementations [32–37].

Systematically, in a two-level system, the signal level is either  $V_{dd}$  (the highest voltage in the system) or  $V_{ss}$  (the lowest system voltage). In multi-valued systems, on the other hand, signal can also have levels between  $V_{dd}$  and  $V_{ss}$  [49, 50]. For example, in a 4-level system with  $V_{ss} = 0$ , the signal levels are  $V_{dd}$ ,  $\frac{2}{3}V_{dd}$ ,  $\frac{1}{3}V_{dd}$ , and 0.

In multi-valued DRAMs, noise margin and consequently reliability of the system is reduced. Noise margin in such systems is calculated as follows.

$$\text{Noise Margin} = (V_{dd} - V_{ss})/2(n - 1) \quad (5.1)$$

where  $n$  shows the number of signal levels or, in this case, storage values.

Thus, Error Correction Codes (ECCs) are generally used to increase noise margin and reliability of multi-valued systems [1, 38–42, 47]. Using ECCs will increase the complexity of the system.

In other words, multi-valued DRAMs may offer a denser storage scheme compared to two-level DRAMs. However, this will result in a decrease in noise margin. Noise margin compensation by using ECCs will cause an increase in the complexity of the system. The increased complexity is comparable to the complexity of a conventional DRAM specially as the number of stored bits increases.

In this chapter, error correction based on the CVNS is used. The CVNS is a fault tolerant multi-valued number system [13]. By decreasing the number of gates and interconnections in the system, the CVNS may result in lower area consumption. The CVNS has been applied as an alternative number system in developing new types of area-efficient arithmetic and signal processing units [14, 17–19, 51].

The CVNS values are represented by a set of continuous digits. The digits have information overlap with each other; every digit has some level of knowledge about the digits with lower index in the same set. This digit level redundancy is used to detect and correct errors in any digit set.

Based on the system specifications, level of redundancy can be selected. In the proposed system, the aim is to reduce the area by reducing the number of storage cells needed to store information bits; therefore, minimum level of information redundancy is chosen. This will result in savings in area while providing enough redundancy for error correction.

Any two adjacent CVNS digits, in the proposed DRAM, have equivalent to one binary digit overlap which is used for error correction in the system. This configuration provides

---

a novel multi-valued DRAM with increased noise margin and reduced area.

The CVNS digits can be easily implemented through current-mode circuits, where basic functions are easy to implement [21–23]. In addition, current-mode circuits can operate with small voltage swings and consequently lower power supply voltages. In the proposed DRAM, the storage and refreshing circuits are designed in current-mode.

The proposed DRAM has to be refreshed continuously; the voltage over the storing capacitor drops after a while due to the leakage currents of the storage cell. During the refreshing cycle, the storage capacitor is charged to the correct value indicated by the error correction codes.

A fast Analog to Digital Converter (ADC) [24] is used for the refreshing circuitry with a parallel configuration that can potentially convert two bits simultaneously. It can reduce the overall delay of the refreshing circuitry.

Dynamic memories with refreshing circuitries based on the CVNS number system were first introduced in [17] where modular ADCs were used in the refreshing circuitry as a part of error correction circuitry. An improved version was reported in [14], where it employed the ADC of [24]. Note that a faster ADC in the refreshing circuitry will result in a faster error correction after each refreshing; corrected data will be available in a shorter time.

The proposed storage and refreshing scheme for DRAM is an extended version of [14]. In the proposed work, all circuits are improved and redesigned for a minimum area consumption, and layouts are also extracted. Post layout simulations show lower delay and power consumption for the proposed circuits compared to those of [14].

In addition, it is proven mathematically how a CVNS-based scheme can result in a doubled noise margin. Nevertheless, error correction using the CVNS is formulated for the proposed design. In the storage scheme, using resistor is proposed to decrease the leakage current and increase the refreshing rate (doubled refreshing rate compared to that of [14]). It means that the proposed memory can store the correct value for a longer time and needs less number of refreshing.

The proposed storage and refreshing circuitry are designed and simulated for a dynamic current-mode multi-valued DRAM with the ability of storing up to 16-level (equivalent to 4 bits) per cell in a  $90nm$  CMOS process technology with a power supply voltage of  $1.2V$ ; the layout is extracted and simulated with parasitic elements to confirm the performance. The proposed multi-valued DRAM will be useful to extend the application of the CVNS in design and implementation of neural networks.

In section 5.1, we discuss how we have chosen optimum dimensions of the CVNS number system based on our available technology. Section 5.2 formulates the noise margin and error correction for the proposed CVNS-based design. CVNS-based storage configuration and related circuits are presented in section 5.3. Details of the refreshing circuitry are provided in 5.4. The simulation results and comparisons to similar systems are presented in 5.5 followed by the conclusion.

## 5.1 CVNS Modifications for Implementation

As can be observed from equation (1.3) and Table 1.1, the CVNS digit with the highest index contains the whole information of the original value or root value which is represented by the CVNS system. Therefore, the term Most Significant Bit (MSB) is not suitable, and Most Informed Digit (MID) is used instead.

Based on the equation (1.3), to generate more informed digits within a digit set, a high resolution environment is needed. For example, a reliable environment with 16-bit resolution is required to generate the MID digit from a 16-bit binary input value. To overcome this problem, a truncated method called group method is applied for conversion. In this method, the length of group,  $\psi$ , indicates the maximum number of binary digits which are used to generate each of the CVNS digits at a time. However, group method reduces the digit-level redundancy; each CVNS digit has information about only  $\psi$  binary digits as

follows.

$$((x))_{n-j} = \sum_{i=m-j-\psi+1}^{m-j} x_i \cdot 2^{i+j-m}, j = 0, 1, \dots, n \quad (5.2)$$

The value of the  $\psi$  is chosen based on the maximum reliable resolution of the implementation environment, and is technology dependent. For  $\psi = 4$  the block diagram of the modified conversion equation is shown in Figure 5.1.

Comparing Figure 5.1 to Figure 1.1, the number of interconnections is reduced considerably after applying the group method.

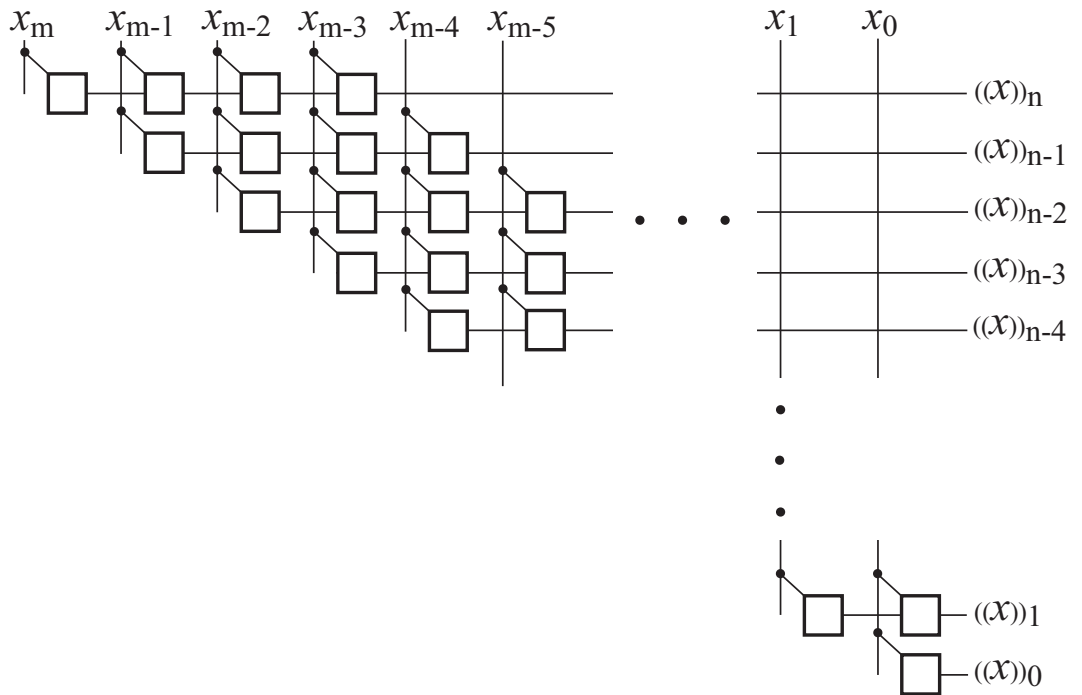


Figure 5.1: Block diagram of simplified CVNS digit generation using the group method. Squares show the weighted sum operation units.

The number of the CVNS digits in each CVNS digit set is another controllable feature which depends on the chosen radix of the CVNS representation. In general, the higher the CVNS radix is, the less the number of the required CVNS digits will be. In this design, the CVNS radix is chosen equal to two to reduce the circuit complexity, however, the same

number of binary and the CVNS digits are required which means that the CVNS digits are linked to each other with the maximum information redundancy. For example, a 16-bit binary input is represented by sixteen radix-2 CVNS digits. If the CVNS radix was chosen higher than two, the number of the required CVNS digits would have been less than that of the binary representation. The trade-off here is that the low CVNS radices have lower circuit complexity, while requiring more number of digits.

To overcome this limit while keeping the CVNS radix low, equal to 2, the link of information between the CVNS digits can be reduced. Thus, another feature is defined which represents the maximum digit link between each two neighboring CVNS digits. According to this, the equation (5.2) will be modified as follows.

$$((x))_{n-j} = \sum_{i=m-j(\psi-\nu)-\psi+1}^{m-j(\psi-\nu)} x_i \cdot 2^{i+j(\psi-\nu)-m}, \quad j = 0, 1, \dots, n \quad (5.3)$$

where  $\nu$  indicates digit links and  $\nu < \psi$ .

Reducing the number of digit links between the CVNS digits results in the less number of the required CVNS digits corresponding to a certain binary input.

From equation (5.3), each CVNS digit is generated from  $\psi$  binary digits where  $\psi - \nu - 1$  digits are not reused for generation of other CVNS digits. Therefore, the number of the CVNS digits,  $n + 1$ , becomes smaller than the number of input binary digits,  $m + 1$ , for  $\nu < (\psi - 1)$ .

The effect of different group lengths and digit links is shown in the following example.

*Example:* For the radix-10 value of 89.0537412, Table 5.1 shows the first five equivalent CVNS digits for different values of  $\psi$  and  $\nu$ . Radix-10 example is chosen as it is easier to follow.

To store up to 4 bits per storage cell in the proposed DRAM, group length,  $\psi$ , and digit link,  $\nu$ , are chosen as 4 and 1, respectively. The block diagram representation of this configuration is shown in Figure 5.2.  $\nu$  is chosen equal to one to provide the minimum possible overlap between two adjacent digits which results in less number of the required

Table 5.1: An example for the CVNS digits with different group lengths and digit links

$\psi$	$\nu$	$((x))_n$	$((x))_{n-1}$	$((x))_{n-2}$	$((x))_{n-3}$	$((x))_{n-4}$
9	8	8.90537412	9.0537412	0.537412	5.37412	3.7412
9	4	8.90537412	7.412	-	-	-
6	5	8.90537	9.05374	0.53741	5.37412	3.7412
6	3	8.90537	5.37412	4.12	-	-
6	2	8.90537	3.7412	1.2	2	-
4	2	8.905	0.537	3.741	4.12	1.2
4	1	8.905	5.374	4.12	2	-

CVNS digits.

## 5.2 Noise Margin and Error Correction Based on CVNS

In order to increase the noise margin and reliability of the proposed system, error correction is unavoidable. This section discusses the error correction based on the CVNS theory and shows its effect on the noise margin.

To discuss the noise immunity of a multi-valued memory, noise margin is introduced as a suitable quantity based on the quantized levels [1, 38, 39]. Noise margin of a multi-valued memory is defined as the maximum allowable positive or negative change in the stored value so that it still represents a correct value, neither a greater nor a smaller one. According to this definition, the multi-valued memories generally have a noise margin equal to the minimum difference between each two levels.

In the proposed DRAM, error correction is considered to suppress the errors due to the leakage current as the major source of error; therefore, noise margin is applied only for negative changes in the stored value.

In the reported CVNS-based memory, each CVNS digit is generated from 4 binary input



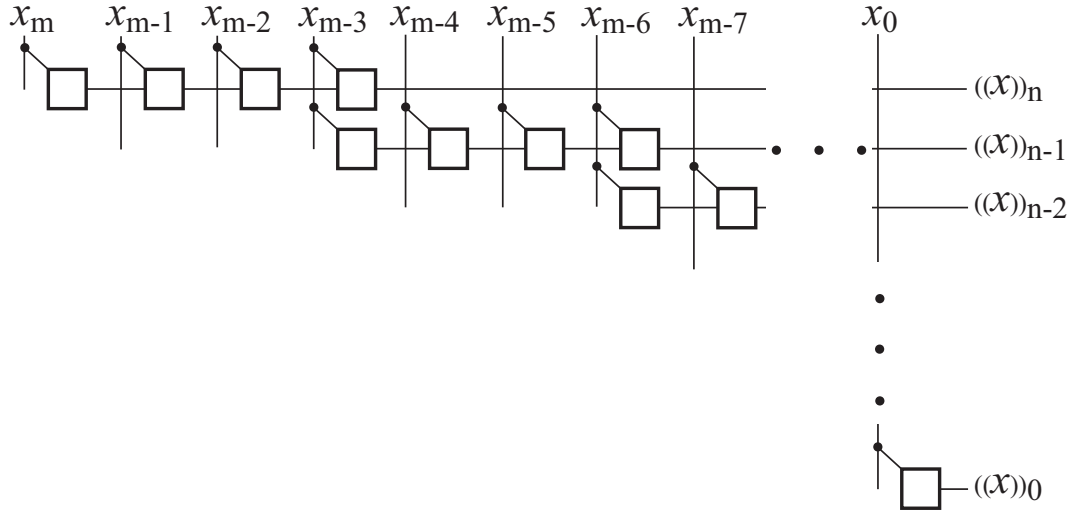


Figure 5.2: Block diagram of the modified CVNS digit generation with  $\psi = 4$  and  $\nu = 1$ .

bits and stored on one memory cell; each memory cell contains one CVNS digit. The LSB input to each cell is repeated as the MSB input to the next cell for error correction which is equivalent to one digit link between each two adjacent cells. It is shown in [1, 38, 39] that error correction using  $M$  number of digit links increases the noise margin of a multi-valued memory system by a factor of  $2^M$ . As a result, the noise margin in the proposed DRAM is increased by a factor of 2.

*Example 1:* For a 10-bit binary input of '1001100101', the first three CVNS digits with  $\psi = 4$  and  $\nu = 1$  are extracted from equation (5.3) as follows.

$$\begin{aligned}
 ((x))_n &= 1 \times 2^0 + 0 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} \\
 ((x))_{n-1} &= 1 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 0 \times 2^{-3} \\
 ((x))_{n-2} &= 0 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3}
 \end{aligned} \tag{5.4}$$

Here, the bit '1' which is multiplied by  $2^{-3}$  in  $((x))_n$  has the least significance among the other 3 bits. This '1' is repeated in  $((x))_{n-1}$  and multiplied by  $2^0$  which gives it the most significance. Therefore, if the value of the bit '1' multiplied by  $2^{-3}$  in  $((x))_n$  decreases due to the error (it is the most vulnerable bit to error in the set as it has the least significance),

its original correct value can still be observed and restored using its value in  $((x))_{n-1}$  where it has the most significance and the least vulnerability to error.

To simplify the equations, equation (5.3) is rewritten for  $\psi = 4$  and  $\nu = 1$  as follows; this equation is referred now on.

$$((x))_{n-j} = \sum_{i=m-j(3)-4+1}^{m-j(3)} x_i \cdot 2^{i+j(3)-m}, \quad j = 0, 1, \dots, n \quad (5.5)$$

To ease the calculations, two sides of equation (5.5) are multiplied by  $2^3$ . Therefore, two adjacent CVNS digits (with  $j = 0$  and 1) can be represented as follows.

$$\begin{aligned} ((x'))_n &= 2^3 \times ((x))_n = x_m 2^3 + x_{m-1} 2^2 \\ &\quad + x_{m-2} 2^1 + x_{m-3} 2^0 \\ ((x'))_{n-1} &= 2^3 \times ((x))_{n-1} = x_{m-3} 2^3 + x_{m-4} 2^2 \\ &\quad + x_{m-5} 2^1 + x_{m-6} 2^0 \end{aligned} \quad (5.6)$$

where for a binary system,  $x_m, \dots, x_{m-6}$  are always either 0 or 1.

Error in a CVNS digit,  $\varepsilon$ , can be stated through the following equation:

$$\varepsilon = | \lfloor [ \lfloor [ ((x'))_{n-j} \bmod 2^3 \rfloor \bmod 2^2 \rfloor \bmod 2^1 ] - \lfloor ((x'))_{n-j-1} \rfloor / 2^3 | \quad (5.7)$$

where the floor function over  $((x))_n$  for radix 2 is defined as  $\lfloor ((x))_n \rfloor = \lfloor ((x))_n \bmod 2^3 \rfloor$  and  $j = 0, 1, \dots, n$ .

For example, for the first CVNS digit of equation (5.6),  $\lfloor [ \lfloor [ ((x'))_n \bmod 2^3 \rfloor \bmod 2^2 \rfloor \bmod 2^1 ]$  is  $x_{m-3} 2^0$ , and  $\lfloor ((x'))_n \rfloor$  is  $x_m 2^3$ .

An ADC is employed to extract the  $\lfloor ((x'))_{n-j} \rfloor$  and  $\lfloor [ \lfloor [ ((x'))_{n-j} \bmod 2^3 \rfloor \bmod 2^2 \rfloor \bmod 2^1 ]$  of each cell in the form of binary bits which will be employed as  $x_3$  and  $x_0$ , respectively. Then, the  $x_0$  extracted from each cell is compared to the extracted  $x_3$  of the next adjacent cell through an XOR gate.

An errored CVNS digit,  $((x'))_{n-j}'$ , with error of  $\varepsilon$  can be written as follows.

$$\begin{aligned} ((x'))_{n-j}' &= x_{m-3j} 2^3 + x_{m-3j-1} 2^2 \\ &\quad + x_{m-3j-2} 2^1 + [x_{m-3j-3} - \varepsilon] 2^0 \end{aligned} \quad (5.8)$$


---

In the proposed refreshing circuitry, a constant current equal to  $x_i \times 2^0$ , where  $x_i = 1$ , is applied when required to correct the error. Therefore, the corrected CVNS digit,  $((x'))_{n-j}''$ , will be as follows.

$$((x'))_{n-j}'' = ((x'))_{n-j}' + 1 \times 2^0 \quad (5.9)$$

which represents the same value as the  $((x'))_{n-j}$  does.

Using one digit link, error can be corrected as long as equation (5.7) is smaller than  $2^1$  which is equal to two levels resulting in a doubled noise margin.

*Example2:* For the CVNS digits of *Example1*, consider an error occurred in  $((x'))_n = 2^3 \times ((x))_n$ .

$$((x'))_n' = 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 \quad (5.10)$$

The error is calculated from equation (5.7) as follows.

$$\varepsilon = |0 \times 2^0 - 1 \times 2^3/2^3| = 1 \times 2^0 \quad (5.11)$$

The errored CVNS digit will be corrected using equation (5.9) as follows.

$$\begin{aligned} ((x'))_n'' &= 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 + 1 \times 2^0 \\ &= 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 \\ &= ((x'))_n \end{aligned} \quad (5.12)$$

In the proposed radix-2 CVNS-based memory with  $\psi = 4$  and  $\nu = 1$ , the basic current of  $0.5\mu A$  is dedicated to  $2^0$ ; accordingly, for  $2^1$ ,  $2^2$ , and  $2^3$ , the corresponding currents are  $1\mu A$ ,  $2\mu A$ , and  $4\mu A$ , respectively. Here, step size is defined as the minimum difference between current levels which is equal to  $0.5\mu A$ . Table 5.2 shows an example of error correction in this system for a 16-bit digital input. Note that in this case,  $n$  and  $m$  are 4 and 15, respectively.

There is a trade-off between area and order of noise margin in the multi-valued memories. In order to increase the noise margin, more digit links should be used which will need more area for hardware implementation. According to the equation (5.3) and Table (5.1) as

---

Table 5.2: An example for error correction when the binary input is 1011, 0101, 1110, 0011

$j$	$x_{15-3j}, \dots, x_{12-3j}$	$((x'))_{4-j}$	$((x'))'_{4-j}$	Error Correction Current	$((x'))''_{4-j}$
0	1011	$5.5\mu A$	$5\mu A$	$0.5\mu A$	$5.5\mu A$
1	1010	$5\mu A$	$5\mu A$	0	$5\mu A$
2	0111	$3.5\mu A$	$3\mu A$	$0.5\mu A$	$3.5\mu A$
3	1100	$6\mu A$	$5.5\mu A$	$0.5\mu A$	$6\mu A$
4	0011	$1.5\mu A$	$1.5\mu A$	0	$1.5\mu A$

the number of digit links increases, the number of the CVNS digits,  $n + 1$ , will increase. Therefore, more cells are needed to store all equivalent CVNS digits. Considering the total area consumption as a function of number of storage cells (CVNS digits),  $n + 1$ , there is an almost linear relation between the number of digit links and area consumption as follows.

$$Total\ Area = (n + 1) \times Area\ of\ one\ Cell \quad (5.13)$$

In the proposed design, using one digit link, the area is minimized while a double noise margin is achieved.

### 5.3 CVNS-Based Storage Circuitry

The overall block diagram of the CVNS DRAM is shown in Figure 5.3. This section is dedicated to the storage cells saving the refreshing circuitry for the next section.

In the proposed 16-level DRAM, equivalent to four binary digits can be stored on each cell. By correcting errored digits due to leakage currents, the CVNS number system can provide an appropriate noise margin for storing multi-valued information.

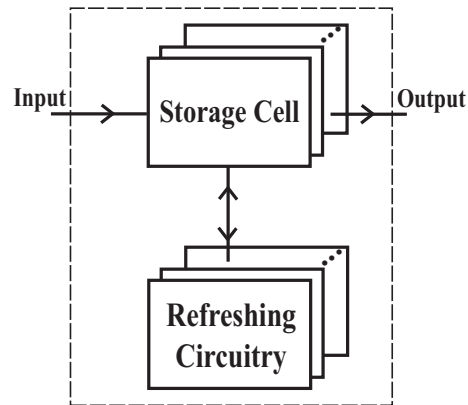


Figure 5.3: Overall block diagram of the proposed memory

Each CVNS digit is supposed to be stored on one storage cell. Each two cells has one digit in common, digit link. This overlap is used for error correction. Since each CVNS digit is generated from four binary digits, five storage cells are required to store a 16-bit digital input. Figure 5.4 shows the proposed arrangement for storing a 16-bit binary input on the 16-level CVNS memory cells.

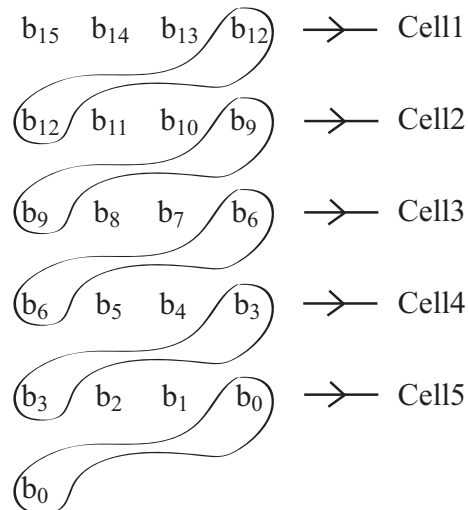


Figure 5.4: General storage scheme for a 16-bit input. In order to increase the reliability of the cells, the LSB input to each cell is repeated and is used for error correction.

Dynamic current mirrors (current copiers) [27] with the capability of storing more than one bit of information are used to implement storage cells in this memory as shown in Figure 5.5.

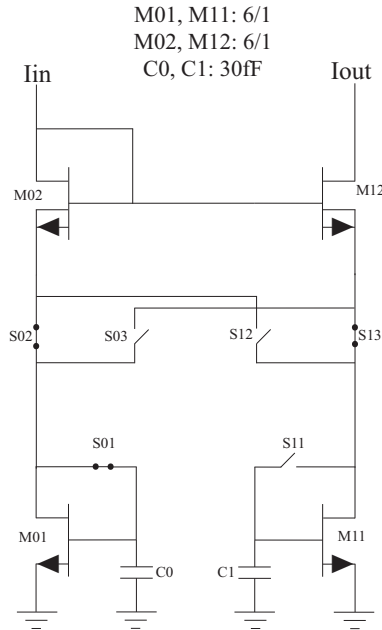


Figure 5.5: Self-biased dynamic current mirror memory (current copier)

Transistor M02 is added in series to transistor M01 to reduce the output conductance [52]. Therefore, sensitivity of the cell to the power supply variations will be decreased [39] which allows  $I_{out}$  to be equal to  $I_{in}$  independent of voltage variations. Transistor M02 should be biased; this can be performed in two ways: self-biased mirror and externally biased mirror [52]. Self-biased method (stacked memory) is used in the proposed memory cell, and transistors M11, M12, C1, and relevant switches perform the mirror part for biasing the transistor M02 in Figure 5.5.

In the self-biased memory, there are two complimentary sets of transistors and switches (base set and mirror set), and each set operates in two states: store and sink. When the base set operates in the store state, the mirror set operates in the sink state and vice versa.

The detail of each state starting from the base set is as follows. Store state happens

when  $S_{01}$  and  $S_{02}$  switches are closed and  $S_{03}$  is open. In this state, storage capacitor,  $C_0$ , starts to charge through the closed switches till the drain current of transistor  $M_{01}$  matches the input current,  $I_{in}$ , meanwhile all the mirror set switches are in the sink state.

In the sink state, the  $S_{01}$  and  $S_{02}$  become open and  $S_{03}$  closes. In this state, transistor  $M_{01}$  sinks a current equal to  $I_{in}$  from the load,  $I_{out}$ . At the same time, all the mirror set switches are in the store state.

It is important for the  $S_{i1}$  and  $S_{i2}$  ( $i = 0, 1$ ) switches to stay close long enough so that the charge on the corresponding storage capacitor,  $C_i$ , reaches the proper value. Note that the charge leakage over the capacitor limits the time of circuit operation in the sink state.

Simulations confirm that the circuit can store up to 4-bit with a capacitor as low as  $30fF$ .

To decrease the leakage current of the storage capacitor, resistance should be increased. Resistors are added in the sources of  $M_{01}$  and  $M_{11}$  as resistors are more stable than the transistors, and their resistance would not change with voltage variations. Adding the resistor, there is a drop in the output current, but the output current value shrinks slower during the time. However, the larger the value of the resistor is, the higher the layout area consumption will be.

A range of resistors between  $1K\Omega$  and  $40K\Omega$  are examined and the storage cell output current for each case is measured during  $1.5\mu s$  for the maximum circuit current (where the input is '1111'). The simulation result of the storage cell output current using different resistors is shown in Figure 5.6. In the top picture, solid line shows the initial output current while the dash-dot line shows the output current after  $1.5\mu s$ . The bottom picture shows the current reduction rate in  $1.5\mu s$ .

As can be observed from Figure 5.6, for smaller resistors, the initial output current is almost equal to the input current,  $7.5\mu A$ , while the current reduction rate is noticeably large. For larger resistors, on the other hand, current reduction rate is small, but they cause a large initial current drop. The idea of adding resistors is to reduce the current reduction

---

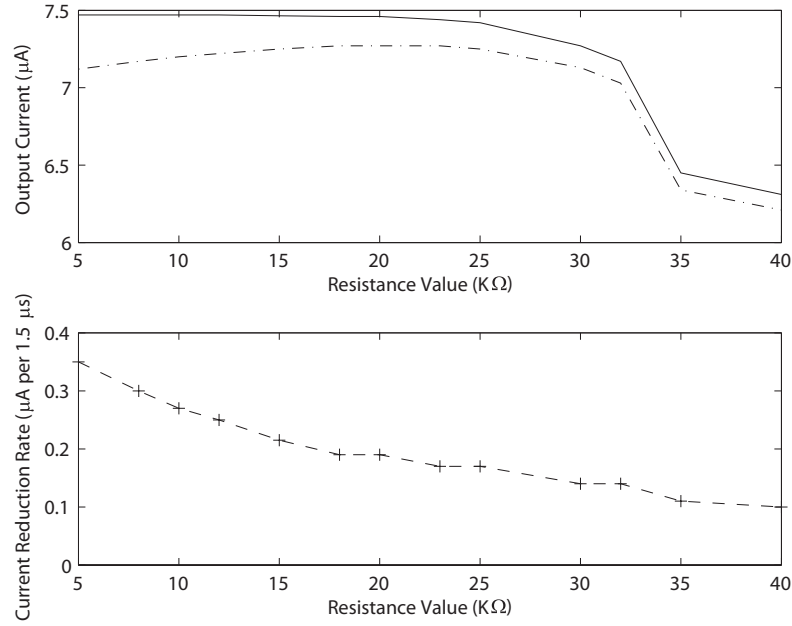


Figure 5.6: Simulation results of different resistors in storage cell.

rate while trying to keep the area consumption and current drop as low as possible.

The  $10K\Omega$  resistor is chosen as the output current drop is negligible ( $\frac{(7.5 - 7.45)}{7.5} = 0.4\%$ ) and the current reduction rate is  $0.3\mu A$  per  $1.5\mu s$ . The proposed storage cell with no resistor needs to be refreshed each  $2\mu s$  while its refreshing rate is doubled ( $4\mu s$ ) by adding the  $10K\Omega$  resistor. Greater refreshing rate means that the storage cell can keep the information for a longer time which is a desirable factor in DRAMs as it will result in less number of refreshing [38, 39, 46, 47, 53, 54].

Layout of the proposed storage cell without resistors and capacitors is shown in Figure 5.7. The overall area is  $10.81\mu m \times 6.49\mu m$ .

The  $10K\Omega$  resistor is laid out using Unsilicided  $P^+$  Poly (rpporpo). The  $30fF$  capacitor is laid out using Metal Capacitor (cmimmk).

Figure 5.8 shows the post layout simulation result of the storage cell when the binary input is '1111'. It also shows the value of switches during each period. As is shown in Fig-



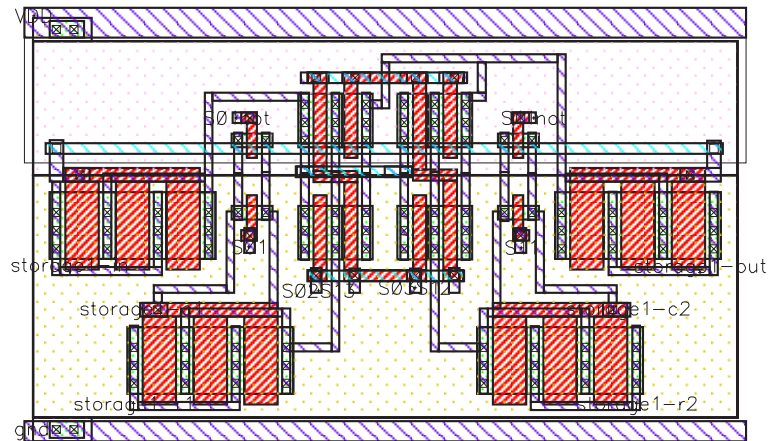


Figure 5.7: Storage cell layout

ure 5.8, the current stored on the storage cell is reduced gradually, but after each refreshing time ( $4\mu s$ ) it is restored to its original value due to the error correction in the refreshing circuitry.

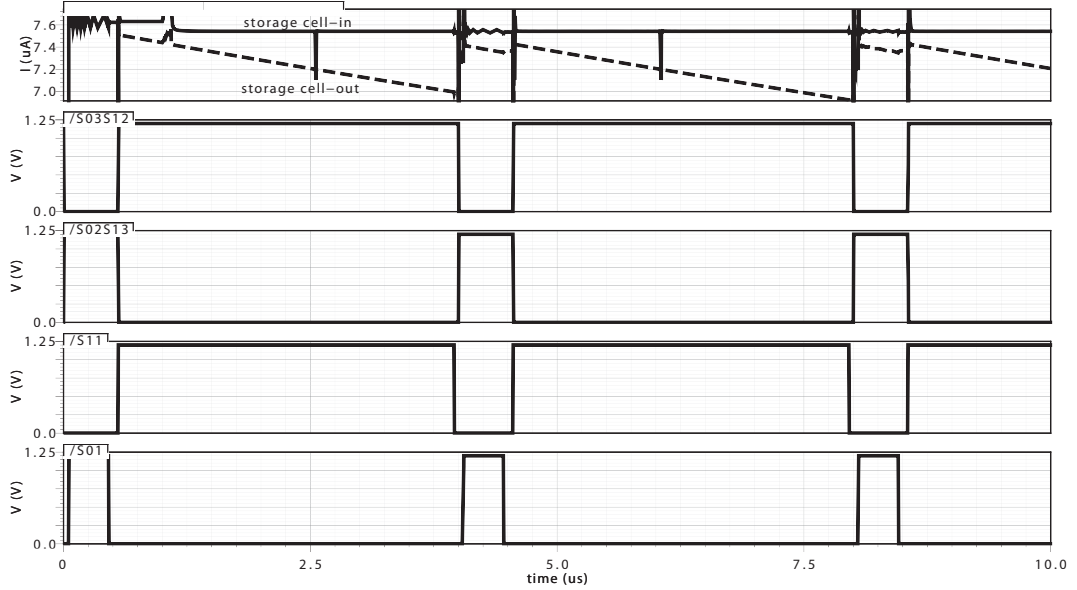


Figure 5.8: Simulation result of storage cell layout for '1111' binary input.

Table 5.3 presents the results of power consumption measurements of the schematic

storage cell and layout storage cell. Power consumption is measured for different binary inputs of '1111', '1000', and '0001'. The average power is calculated per one period.

Table 5.3: Schematic storage cell and layout storage cell simulation results

	<b>Schematic storage Cell</b>	<b>Layout Storage Cell</b>
<b>Average Power</b>	9.669 $\mu w$	9.672 $\mu w$

## 5.4 CVNS-Based Refreshing Scheme and Circuits

In multi-valued memories, storing more than one digit on one cell can reduce the area while reducing the noise margin as a drawback. Error Correction Codes have been used in the refreshing circuitry to increase the noise margin by detecting and hence correcting one or more errored values [1, 38–42, 47]. However, because of the complex decoding and addressing configurations applied in the refreshing circuitry, these systems generally have high design complexity. The CVNS number system, offering a low complexity configuration, can serve as an alternative to store more than one bit per cell while increasing the noise margin. Use of any kind of complex decoding schemes can be avoided because the CVNS-based systems can be easily implemented through low resolution current mode analog circuits as shown here.

The base of the CVNS DRAM cell is a transistor and a capacitor similar to a conventional DRAM cell. The voltage over the capacitor will be reduced after a while due to the leakage current; therefore, it needs to be refreshed to recover the capacitor voltage.

The series configuration of ADC and DAC units in the refreshing circuitry has been used in some previous published works [1, 34, 35, 38, 39, 46, 47]. The proposed refreshing circuitry is based on this general configuration and is modified to perform CVNS representation requirements as is shown in Figure 5.9.

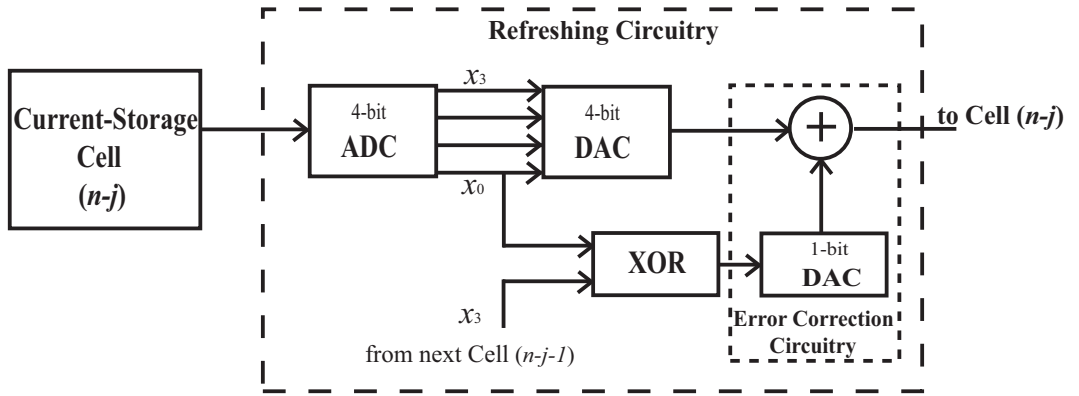


Figure 5.9: Block diagram of the refreshing system for a CVNS DRAM cell.

Here, the current quantizers and capacitors of [1,38,39] and phase detectors of [1,34,35,38,39] are avoided which results in reducing the area, power consumption, and complexity.

The general system operation is as follows. First, the value stored on a memory cell is converted to digital through a 4-bit ADC.

A 4-bit ADC [14, 24] is designed which can convert 2-bit at the same time resulting in a faster conversion compared to continuous Modular ADCs [17, 18, 21, 22, 55, 56]. In addition, employing a DAC, the designed 4-bit ADC consumes less area than ADCs with no DAC [29, 57–59]. The 4-bit ADC is made of a combination of current comparators, DAC, and encoders.

Then, the  $x_0$  of each cell is compared to the  $x_3$  of the next line. As only one bit is used for error detection, a simple XOR gate can be used for comparison. It should be noted that analog current comparators are needed when two adjacent cells have more than one digit in common. A zero in the XOR gate output shows that no error has happened and the two compared bits are the same. On the other hand, appearance of a one means that the two compared bits are different; the cell value is altered due to the leakage currents, and it needs to be refreshed.

The output of the XOR gate goes to a 1-bit DAC which acts as a current generator. When error occurs, this gate adds a constant current corresponding to  $x_i \times 2^0$  where  $x_i = 1$

to the output current of the 4-bit DAC resulting in a total refreshing current which will be restored on the memory cell.

The 4-bit DAC is a parallel combination of four simple weighted current sources. The circuit operation of ADC and DAC modules are given next.

### 5.4.1 Analog to Digital Converter

Figure 5.10 shows the block diagram of the system which is proposed to convert the CVNS analog digits into their corresponding binary values. Each 4-bit conversion is performed in two stages. In other words, there is a parallel structure to generate each 2-bit at the same time resulting in a decrease in total conversion time. Using a fast ADC in the refreshing circuitry decreases the delay time of error correction after each refreshing.

More significant bits,  $x_3$  and  $x_2$ , are produced from the analog input value (input current) in the first stage. According to the output values of the first stage, the second stage generates the other two bits,  $x_1$  and  $x_0$ . The Table 5.4 shows the value of  $I_{ref_i}$  and  $I_{const_i}$  ( $i = 1, 2, 3$ ).

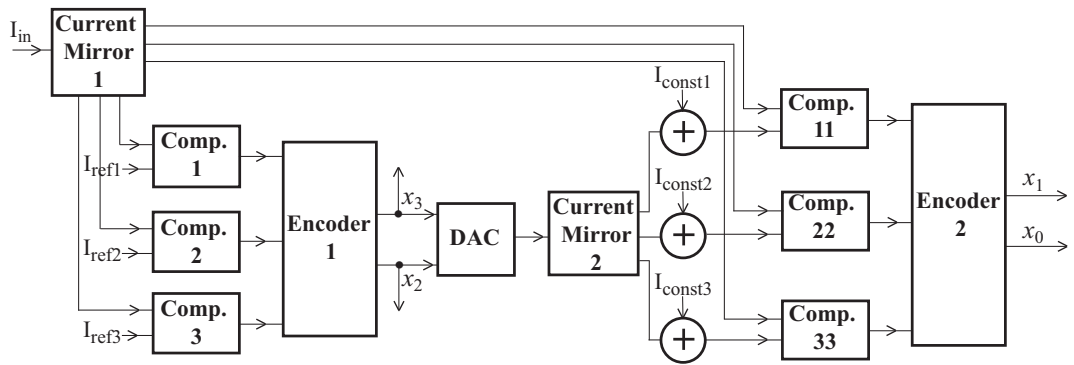


Figure 5.10: Block diagram of the ADC

The detail of 4-bit ADC operation is as follows. In the first stage, current comparators,  $Comp.i$  ( $i = 1, 2, 3$ ), compares the analog input,  $I_{in}$ , to constant reference currents,  $I_{ref_i}$ . Current comparators are simple current mirror comparators [60].

Table 5.4: Values of all currents in the ADC block diagram

$I_{ref1}$	$I_{ref2}$	$I_{ref3}$
$2\mu A$	$4\mu A$	$6\mu A$
$I_{const1}$	$I_{const2}$	$I_{const3}$
$0.5\mu A$	$1\mu A$	$1.5\mu A$

Here, the output of  $Comp.i$  is one when  $I_{in} > I_{ref_i}$  and zero otherwise. According to the three current comparator outputs, the  $Encoder1$  indicates the values of  $x_3$  and  $x_2$  (Table 5.5).

Table 5.5: Extraction of  $x_3$  and  $x_2$  from the output of current comparators.

Comp. 3	Comp. 2	Comp. 1	$x_3$	$x_2$	DAC Output
0	0	0	0	0	0
0	0	1	0	1	$2\mu A$
0	1	1	1	0	$4\mu A$
1	1	1	1	1	$6\mu A$

It is observed from Tables 5.4 and 5.5 that if the output of  $Comp.i$  ( $i = 1, 2, 3$ ) is one, the output of  $Comp.j$  would be also one as long as  $j < i$ .  $x_3$  and  $x_2$  are extracted as follows.

$$\begin{aligned}
 x_3 &= Comp. 2 \\
 x_2 &= Comp. 3 + (Comp. 1 \cdot \overline{Comp. 2})
 \end{aligned} \tag{5.14}$$

where  $\cdot$  and  $+$  shows AND and OR operations, respectively.

Due to the values of  $x_3$  and  $x_2$ , DAC generates a current as is shown in Figure 5.5. This current is added to each constant currents,  $I_{const_i}$  ( $i = 1, 2, 3$ ), resulting in a current called the *total reference current*. Current comparators of the second stage,  $Comp.ii$

Table 5.6: Comparison made between the proposed ADCM and some published methods for a 4-bit ADC

Method	Comparators		Digital Gates		Delay (ns)	Area (trans.)	Delay*Area (ns · trans.)
	series	total	series	total			
Radha [29]	1	16	4 XOR + 3 OR	7 XOR + 4 OR	$1 \times 2.5 + 4 \times 0.21 + 3 \times 0.16 = 3.79$	$16 \times 16 + 7 \times 12 + 4 \times 6 = 364$	1379
Tipsu [57]	1	16	14 XOR	18 XOR + 4 OR	$1 \times 2.5 + 14 \times 0.21 = 5.44$	$16 \times 16 + 18 \times 12 + 4 \times 6 = 496$	2698
Chuen [59]	1	16	14 XOR	18 XOR + 4 OR	$1 \times 2.5 + 14 \times 0.21 = 5.44$	$16 \times 16 + 18 \times 12 + 4 \times 6 = 496$	2698
Agarwal [56]	4	4	-	-	$4 \times 2.5 = 10$	$4 \times 22 = 88$	880
Narin [22]	4	4	-	-	$4 \times 2.5 = 10$	$4 \times 21 = 84$	840
Salama [21]	4	4	-	-	$4 \times 2.5 = 10$	$4 \times 21 = 84$	840
Proposed	2	6	4 AND/OR	4 AND/OR	$2 \times 2.5 + 4 \times 0.16 = 5.64$	$6 \times 16 + 4 \times 6 = 120$	676.8

( $i_i = 11, 22, 33$ ), compare  $I_{in}$  to each of these *total reference currents*. According to their comparison results,  $x_1$  and  $x_0$  are generated by *Encoder2*. Equations of (5.14) is changed to be used for *Encoder2* as follows.

$$\begin{aligned}
 x_1 &= \text{Comp. } 22 \\
 x_0 &= \text{Comp. } 33 + (\text{Comp. } 11 \cdot \overline{\text{Comp. } 22})
 \end{aligned} \tag{5.15}$$

The proposed ADC method is compared to some of reported ADC methods in Table 5.6.

The first three methods (first group ADCs) are those with a parallel structure which have lots of comparators together with some digital gates. The second three methods (second group ADCs), on the other hand, have a completely series configuration and generally no digital gates.

Note that for area consumption estimation all the components are important, while for delay estimation only series components are considered. It should also be noted that the delay of a comparator is much larger (more than 10 times) than the delay of digital gates,

and it uses more transistors and consequently more area than the digital gates.

Accordingly, the proposed method consumes less area (almost one forth) compared to the first group ADCs as it has less number of total components and transistors. The proposed method area consumption is slightly more than the second group ADCs.

The delay of the proposed ADC method is almost equal to the first group ADCs as they almost have the same number of series components. However, the proposed ADC method presents a delay about half of the second group ADCs delay as it has only two comparators in series.

Totally, the proposed ADC method presents the smallest delay  $\times$  area than the first and second group ADCs.

Layout of the 4-bit ADC is shown in Figure 5.11. *C.M.1* and *C.M.2* blocks refer to the current mirrors. The total area consumption is  $23.32\mu m \times 26.67\mu m$ .

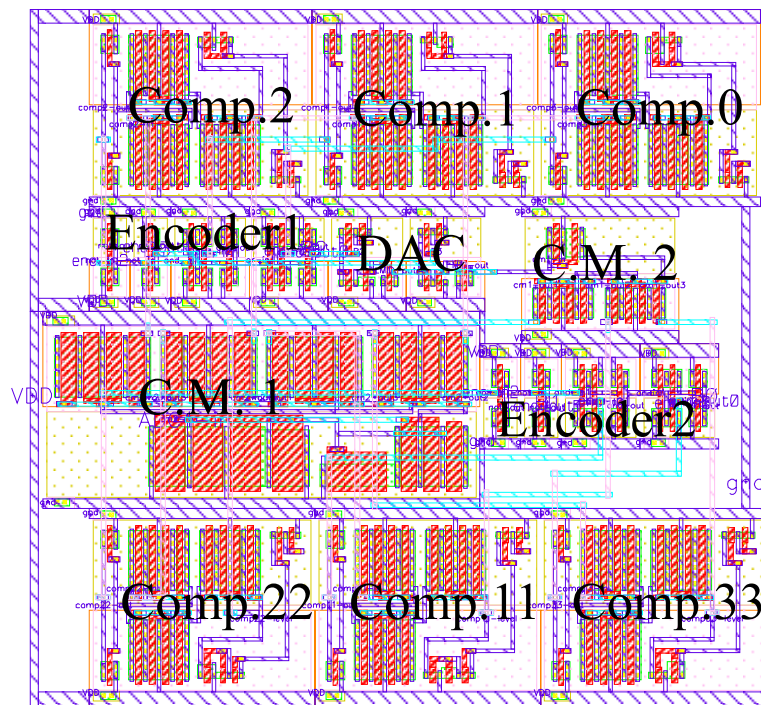


Figure 5.11: 4-bit ADC layout

Table 5.7 compares the schematic ADC and layout ADC in terms of delay and power consumptions. Capacitor loads are connected to the outputs. A pulse with period of  $200ns$  is applied to the inputs. Power consumption is measured for different input currents of  $7.5\mu A$ ,  $4\mu A$ , and  $0.5\mu A$ . The average power is calculated per one period.

Table 5.7: Schematic ADC and layout ADC simulation results

	<b>Schematic ADC</b>	<b>Layout ADC</b>
<b>Delay</b>	$12ns$	$12.65ns$
<b>Average Power</b>	$97.5\mu w$	$95.66\mu w$

The main building block of the ADC is the current comparator; current mirrors also play an important role in the design. The next subsections discuss these two blocks.

#### 5.4.1.1 Current Comparators

Current comparators are 12-transistor current mirrors with size of  $(w/L) = 4/1$ . Figure 5.8 shows the schematic of the employed current comparators.

Each current comparator needs a current source as shown in Figure 5.13 to generate a constant reference current for comparison.

Size of current source transistors for each Current Comparator (Comp.), and their Simulated Constant Current Values (SCCVs) in the ADC layout is summarized in Table 5.8. Note that transistors  $M1$  and  $M2$  have the same size.

Recalling that the step size is  $0.5\mu A$ , SCCVs have a value almost 40% of the step size ( $0.2/0.5$ ) below the desired values listed in Table 5.4. In other words, a decrease in input currents due to the system errors will not cause an error in the output as long as that decrease is within the 40% of the step size. This will make the system more accurate and can increase the reliability.



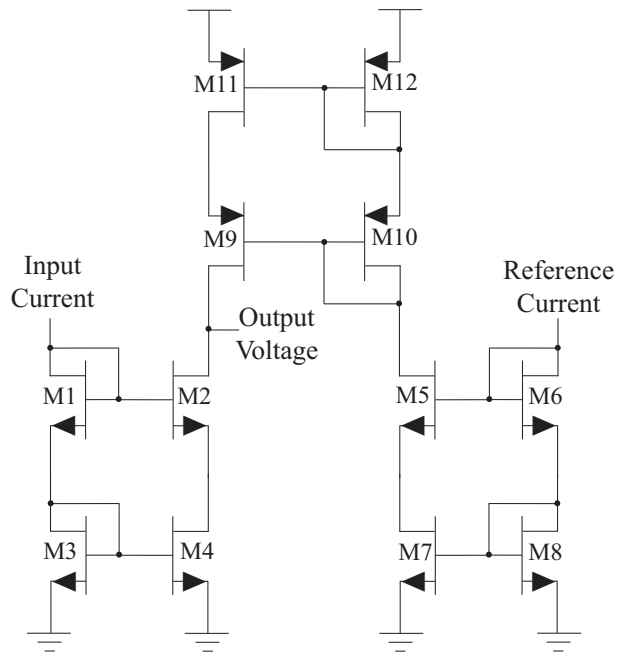


Figure 5.12: Schematic of a current comparator

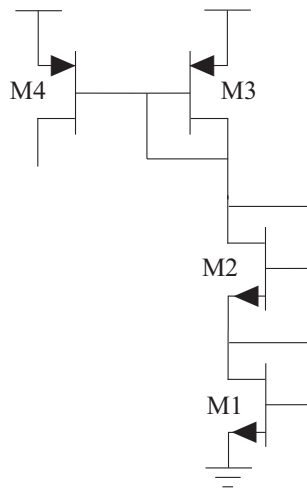


Figure 5.13: Schematic of a current source

### 5.4.1.2 Current Mirrors

Current Mirrors act as both current transferors and matching circuits; therefore, design of such units requires careful consideration of their output current in respect to the input

Table 5.8: Transistor sizes for each current source in the layout

	<i>Comp.1</i>	<i>Comp.2</i>	<i>Comp.3</i>	<i>Comp.11</i>	<i>Comp.22</i>	<i>Comp.33</i>
<i>M1</i>	1/1	1.3/1	1.3/1	1/1	1.3/1	1/1
<i>M3</i>	2.3/1	1/1	1/1	1.75/1	1.3/1	2.75/1
<i>M4</i>	1.3/1	1.75/1	2.75/1	1/1	2.25/1	1/1
<b>SCCV</b>	1.83 $\mu A$	3.77 $\mu A$	5.77 $\mu A$	0.315 $\mu A$	0.797 $\mu A$	1.305 $\mu A$

current and range of acceptable output loads. Design of current mirrors becomes even more challenging when there is a wide range of input current to deal with ( $0 - 7.5\mu A$  in this design).

There are two main current mirrors in the ADC design. The first current mirror, *C.M.1*, serves for transferring input current to all six current comparators. *C.M.1* is a combination of Wilson current mirror for NMOSs and Cascode current mirrors for PMOS branches as shown in Figure 5.14.

Since the input current is in the range of  $0 - 7.5\mu A$ , nonlinearity of transistors shows up which decreases the overall accuracy: the current mirrors output current goes lower than the input current for larger input currents. In order to solve this problem, a transistor with the size of (0.5/1) is added in parallel to the transistor *M*. It generates a small current as its gate voltage increases corresponding to larger input currents.

The second current mirror, *C.M.2*, is a Cascode current mirror as shown in Figure 5.15 and transfers output current of DAC to each of the three current comparators in the second stage. According to Figure 5.5, the DAC output is either 0, 2, 4, or 6.

## 5.4.2 Digital to Analog Converter

In the proposed multi-valued DRAM, the CVNS digits have been used. To form the CVNS digits, groups of 4 binary inputs have been used. Therefore, the maximum required reso-

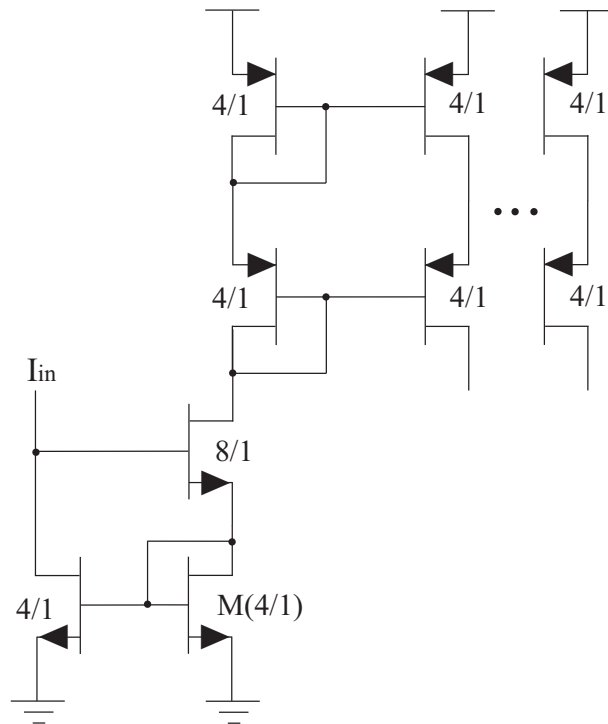


Figure 5.14: Schematic of the first current mirror

lution for digital to analog conversions is 4. In this design, digital to analog converter is made from four parallel 1-bit analog to digital converters as is shown in Figure 5.16.

Each 1-bit converter is a weighted current source (current mirror) which produces a certain current according to the order of the binary input. The schematic of a 1-bit DAC is shown in Figure 5.17.

The output of this 4-bit DAC is produced by adding output currents of all four 1-bit converters in a current summation node.

The current levels of the proposed memory are selected as  $0.5\mu A$ ,  $1\mu A$ ,  $2\mu A$ , and  $4\mu A$  for  $x_0$ ,  $x_1$ ,  $x_2$ , and  $x_3$ , respectively. Four separate weighted current mirrors are used to produce these currents.

In Figure 5.17, when the input binary bit is one, transistor  $M1$  will be in triode region as  $V_{GS} - V_t \geq V_{DS}$  while the other transistors work in saturation region. The current of

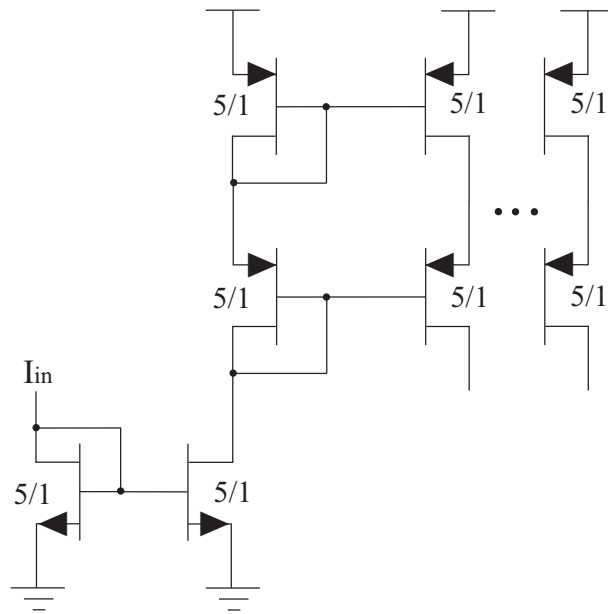


Figure 5.15: Schematic of the second current mirror

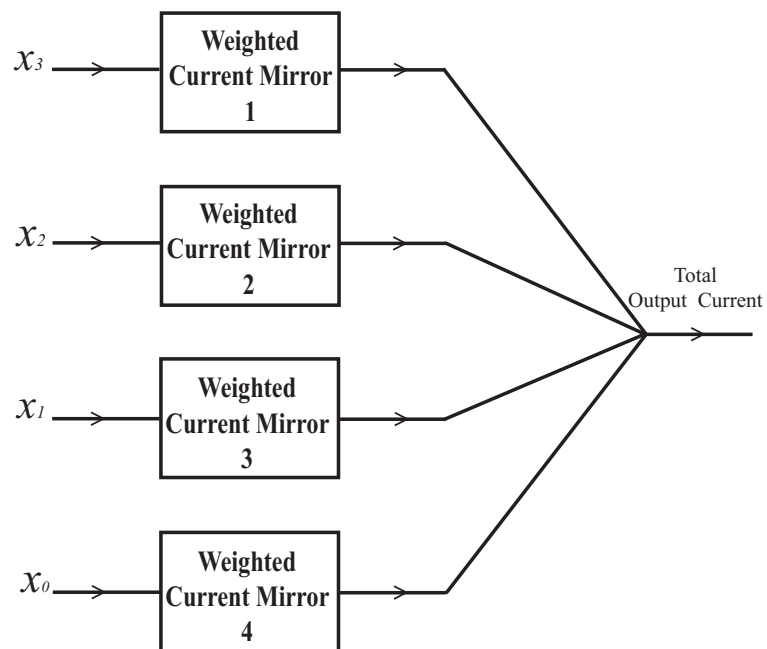


Figure 5.16: Overall configuration of the 4-bit DAC.

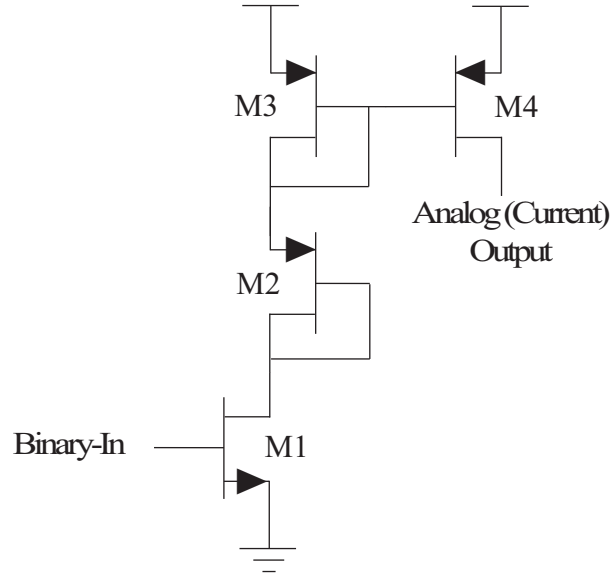


Figure 5.17: Schematic of a one-bit weighted current mirror

$M3$  and  $M4$  are calculated from the following well known equation [61].

$$I_D = \frac{1}{2} \mu_p C_{ox} \frac{W}{L} (|V_{GS}| - |V_t|)^2 (1 + \lambda |V_{DS}|) \quad (5.16)$$

According to equation (5.16), size of transistor  $M4$  can be calculated from size of  $M3$  as follows.

$$(W/L)_4 = (W/L)_3 \times \frac{I_4 (1 + \lambda |V_{DS3}|)}{I_3 (1 + \lambda |V_{DS4}|)} \quad (5.17)$$

Some issues must be considered for designing the transistor sizes. First, current of transistor  $M3$ ,  $I_3$ , must be small to decrease the power consumption. Second, the number of transistors and their sizes (both  $W$  and  $L$ ) should be minimized to make an area efficient design. Therefore, another transistor the same as  $M1$  is added in series to  $M1$  to decrease the  $V_{GS}$  of  $M1$  resulting in a smaller  $I_3$ . If  $I_3$  is much larger than the desired output current, according to equation (5.17) either  $W$  of  $M3$  or  $L$  of  $M4$  must be large.

The size of  $M1$ ,  $M2$ ,  $M3$ , and  $M4$  transistors are designed for each current mirror as is shown in Table 5.9. Table also shows the simulated output currents. The error between

simulated output currents and the ideal currents is 0%, 2%, 4%, and 0.8% for  $4\mu A$ ,  $2\mu A$ ,  $1\mu A$ , and  $0.5\mu A$ , respectively.

Table 5.9: Transistor sizes for each current mirror

$I_4$	$I_3$	$(W/L)_1$	$(W/L)_2$	$(W/L)_3$	$(W/L)_4$
$4\mu A$	$1.34\mu A$	1.3/1	1.3/1	1.3/1	3.3/1
$1.96\mu A$	$1.55\mu A$	1/1	1/1	2/1	2/1
$1.04\mu A$	$1.7\mu A$	1.3/1	1.3/1	2.6/1	1/1
$0.504\mu A$	$2\mu A$	1/1	0.66/1	4/1	0.4/1

Layout of the designed DAC is shown in Figure 5.18, and the simulated output currents for binary input of '1111' are shown in Figure 5.19.

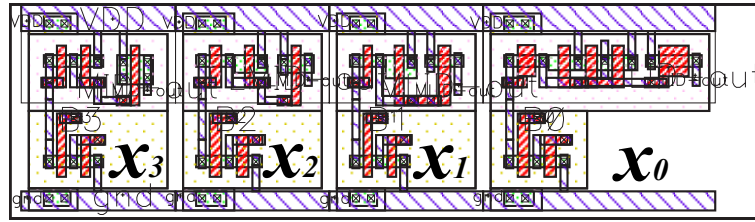


Figure 5.18: DAC layout

From Figure 5.19, the difference between post layout extracted DAC currents and ideal currents is 2.75%, 0.7%, 8.7%, and 1.06% for  $4\mu A$ ,  $2\mu A$ ,  $1\mu A$ , and  $0.5\mu A$ , respectively.

Table 5.10 compares the schematic DAC and layout DAC in terms of delay and power consumptions. NMOS loads with size of  $(W/L) = 2/1$  are connected to the outputs. A pulse with period of  $200ns$  is applied to the inputs. Power consumption is measured for different binary inputs of '1111', '1000', and '0001'. The average power is calculated per one period.

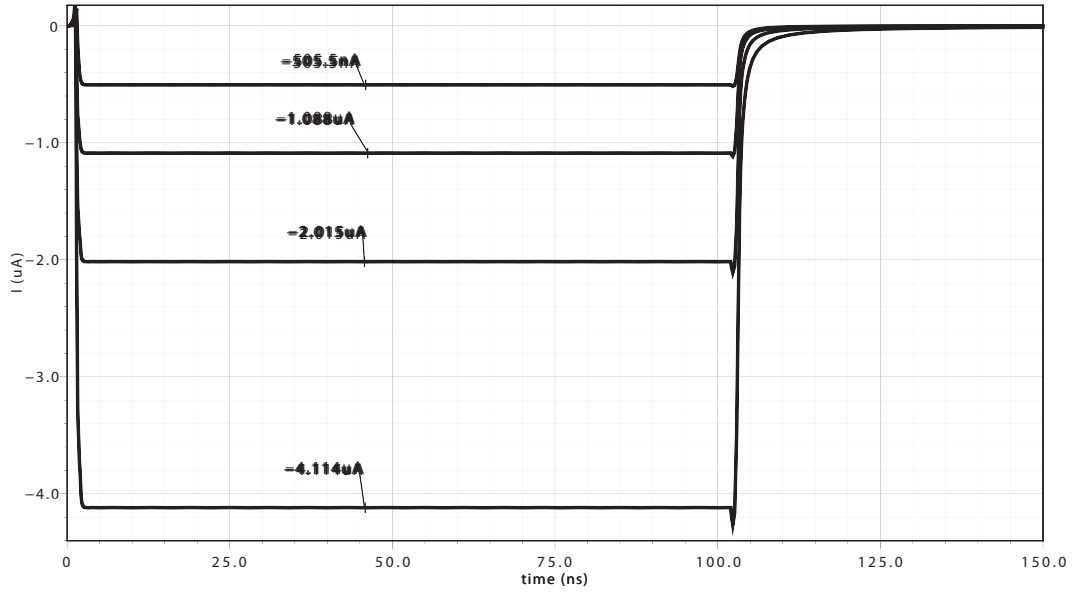


Figure 5.19: DAC layout simulation results for input of '1111'

Table 5.10: Schematic DAC and layout DAC simulation results

	<b>Schematic DAC</b>	<b>Layout DAC</b>
<b>Delay</b>	619ps	920ps
<b>Average Power</b>	4.41 $\mu\text{w}$	4.57 $\mu\text{w}$

## 5.5 Simulation Results and Comparisons

The refreshing and storage circuits of the proposed CVNS DRAM are designed and simulated in  $90\text{nm}$  CMOS technology using the power supply voltage of  $1.2\text{V}$  with the storage capacitor size of  $30\text{fF}$ . Total post layout circuit simulation for binary input of '1110' equal to  $7.0\mu\text{A}$  is shown in Figure 5.20.

The storage cell is being refreshed every  $4\mu\text{s}$ . There is a trade-off between the refreshing time and the size of storage capacitor; as the capacitor size becomes larger, the refreshing time will increase. On the contrary, the area increases as the size of capacitor

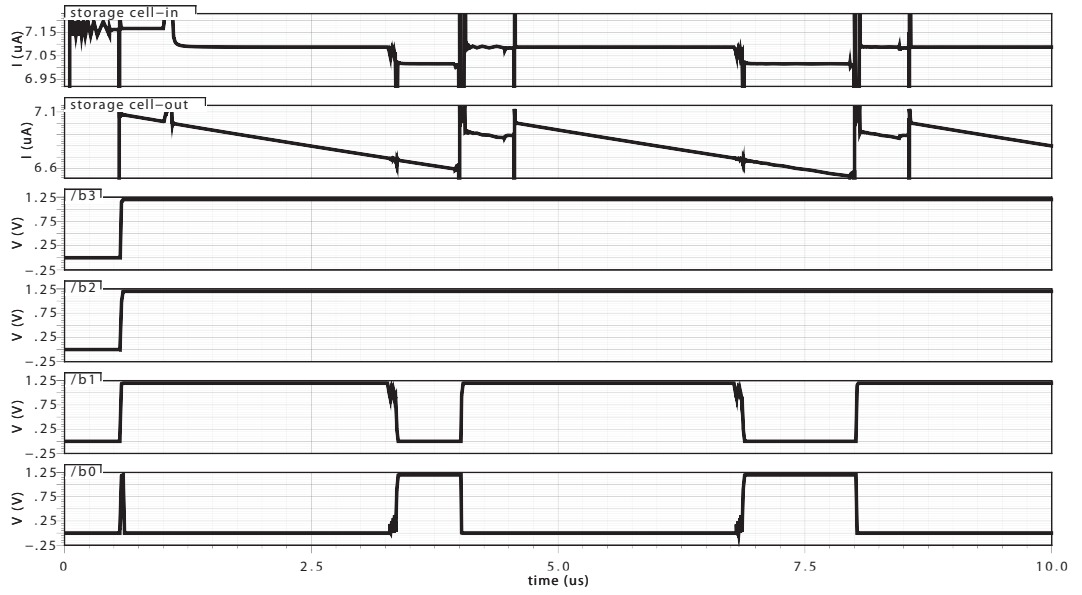


Figure 5.20: Post layout simulation result of total circuit for '11110' binary input.

grows. In this design, the size of storage capacitor is selected as small as possible to reduce the area consumption while providing a large enough refreshing period of  $4\mu s$ . This refreshing time offers the switches of storage cell enough time to complete both store and sink states properly. Moreover, the stored information will not be errored more than two levels in this time which is the system noise margin and can be corrected easily.

The proposed system specifications are compared with some of the existing designs published in literature where information was available as shown in Table 5.11.

Note that in voltage mode designs, special circuits are employed to make reference voltages as a division of power supply voltage. For example, in the 16-level design of [46], the storage levels are  $80 - 100mV$ . Therefore, the minimum acceptable power supply voltage for 16-level storage must be greater than  $16 \times 0.08 + V_{th} = 1.28V + V_{th}$  which is not suitable for CMOS technologies equal or smaller than 90nm.

The work which is more comparable to the proposed memory, due to the current-mode design and structure, is the one reported by Lee et al. [1, 38, 39]. The block diagram is



Table 5.11: Comparison made between the proposed DRAM circuit and available published structures

Architecture	Storage Level	Cap. Size	Cap. Technology	Power Supply	CMOS Technology	Extra Components	Operation Mode
Hotchet [34, 35]	16	2.5 pF	-	5 V	2 $\mu$ m	Phase Detector	Voltage
Lee [1, 38, 39]	16	-	-	3.3 V	1.2 $\mu$ m	Phase Detector Latch	Current
Birk [41]	4	25 fF	-	2.5 V	0.25 $\mu$ m	Sense Amplifier Extra capacitors	Voltage
Okuda [43]	4	60 fF	high dielectric constant material	2.5 V	0.15 $\mu$ m	Sense Amplifier Extra capacitors	Voltage
Gillingham [44]	4	30 fF	-	3.3 V	0.6 $\mu$ m	Sense amplifier Extra capacitors	Voltage
Furuyama [45]	4	40 fF	twin-tub CMOS process	5 V	-	Sense Amplifier Extra capacitors	Voltage
Horiguchi [46, 47]	16	60 fF	-	5 V	1.3 $\mu$ m	Sense Amplifier Extra capacitors	Voltage
Yunan [53]	2/4/5/8	50 fF	-	1.8 V	0.18 $\mu$ m	Sense Amplifier Extra capacitors	Voltage
Khodaba [55]	8	40 fF	-	1 V	90 nm	-	Current
Proposed Cell	16	30 fF	-	1.2 V	90 nm	-	Current

shown in Figure 5.21. The ADC function is performed using Quantized Level Generators (QLGs) and a 1-bit counter. Each QLG contains two current copiers.

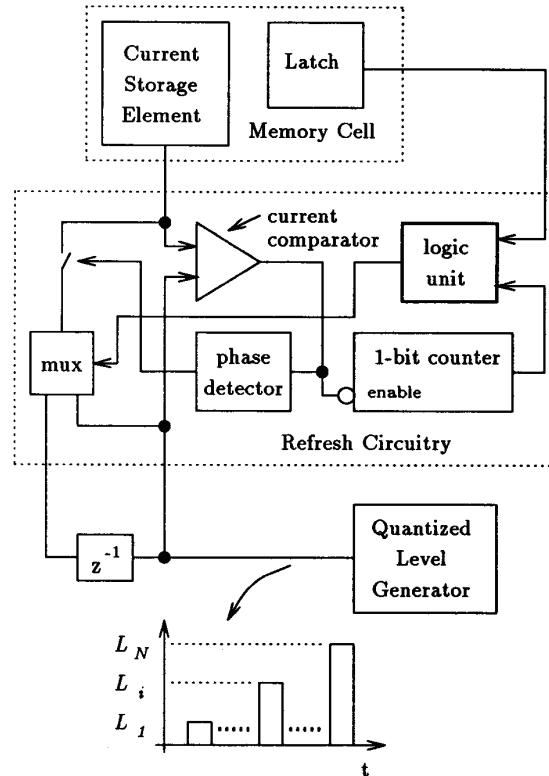


Figure 5.21: Block diagram of the DRAM proposed by Lee et al. [1]

Each current copier needs at least  $0.5\mu s$ , in  $90nm$  CMOS technology, to load the correct current. Therefore, in the DRAM reported by Lee, the refreshing time for a 16-level data takes at least  $16 \times 0.5\mu s = 8\mu s$  to generate the corrected data. Adding this time to the  $4\mu s$ , the refreshing time of the proposed DRAM, the refreshing time will be  $12\mu s$ ; the other blocks are considered to have much smaller delay times than  $0.5\mu s$ .

Simulations show that the minimum capacitor size to cope with  $12\mu s$  refreshing time in Lee's design, considering the storage cell of Fig 5.5, is  $200fF$ . Consequently, with the refreshing time of  $12\mu s$  and the capacitor size of  $200fF$ , corrected data in the Lee's DRAM

is available  $8\mu s$  after refreshing.

In the proposed DRAM, on the other hand, it takes only  $0.5\mu s$  to provide the corrected value after each refreshing (Figure 5.20). In the other words, the proposed DRAM correction speed is 16 times more than the DRAM designed by Lee while the area consumption is almost the same. Nevertheless, the proposed DRAM can provide the  $12\mu s$  refreshing time with the capacitor size of  $100fF$  which is half of the transistor size in Lee's design. It should also be noted that the DRAM designed by Lee can not operate with capacitors smaller than  $130fF$  for 16-level storage; this size of capacitor is needed to keep the stored value in the correctable range for at least  $8\mu s$ .

In comparison to other designs, following advantages are obtained in the proposed current-mode DRAM. First, the proposed memory using a CVNS-based storage scheme can store up to 16-level (equal to 4-bit) per storage cell. Second, the circuits are simplified as extra components like phase detectors and sense amplifiers in refreshing circuitry are avoided; the CVNS-based scheme is provided which is much easier for implementation while quite efficient. Moreover, repeating the LSB input to each cell as the MSB input to the next cell according to the features of the CVNS, a doubled level of noise margin is achieved compared to the systems without error correction. In addition, the small size of the storage capacitor results in a reduced area design of storage cell in particular. Finally, small power supply voltage of  $1.2V$  is used which can cause the total power consumption of the system to be decreased comparing to DRAMs with higher power supply voltages (voltage-mode DRAMs in particular).

## 5.6 Conclusion

In this chapter, design and implementation of storage and refreshing schemes for a current-mode multi-valued DRAM is proposed. The proposed system is designed according to the CVNS features and has the ability of storing 16-level equivalent to 4-bit on one storage cell.

---

Resistors are added to the storage cell to increase the refreshing time. For error correction, LSB input to each cell is stored on the next neighboring cell which also doubles the noise margin. Refreshing circuitry implementation is based on current-mode DAC and ADC modules. A fast ADC module is used that has a parallel configuration and can convert two bits at the same time. Due to the special CVNS-based design and small size of the storage capacitor, the overall area of the system is reduced.

---

## **Chapter 6**

### ***A 4-3-2 CVNS DNN using Synapse-Neuron Modules***

---

Synapse-neuron modules based on the CVNS are used to design a prototype 4-3-2 network with four digital inputs and two final digital outputs. This prototype is trained off-line based on a classification pattern. The circuitries are designed, simulated, and laid out in  $0.18\mu m$  TSMC CMOS technology using a power supply voltage of  $1.8V$ .

Comparisons show that the proposed mixed-signal structure solve the storing problem of analog ANNs while the activation function is realized through analog circuits instead of the huge Look-Up-Tables (LUTs) of digital ANNs.

#### **6.1 Prototype CVNS DNN**

The proposed synapse-neuron module is used to build a 4-3-2 network as a proof of concept design. Here, there are 4 digital inputs for the network which will lead into two final digital

outputs. The configuration is shown in Figure 6.1.

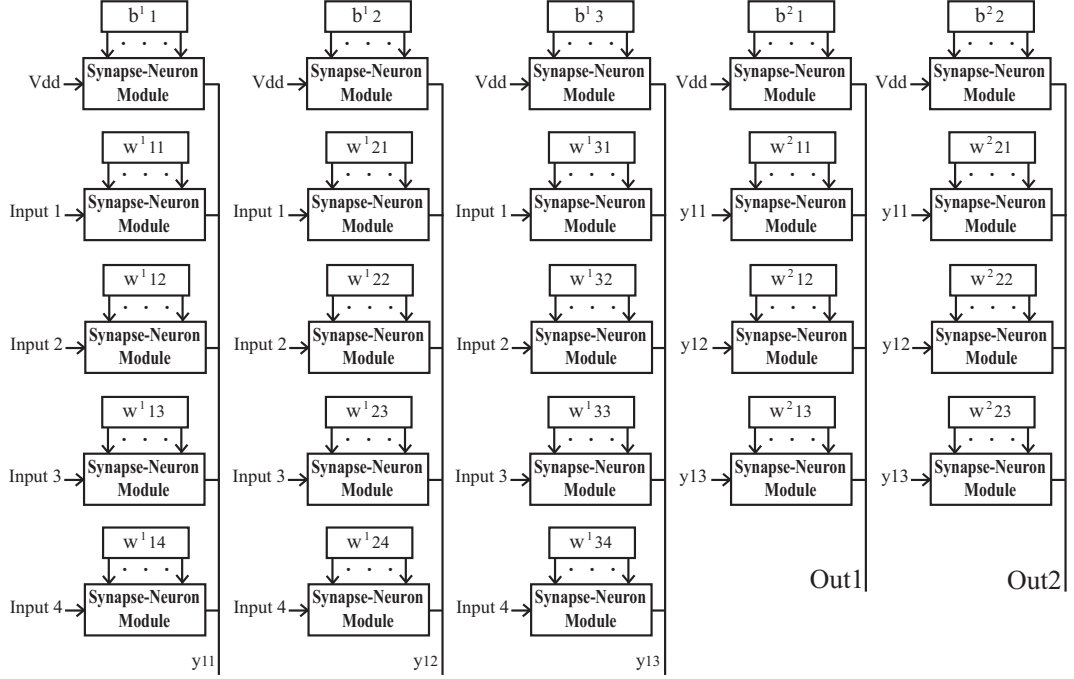


Figure 6.1: Block diagram of the 4-3-2 CVNS DNN.

In Figure 6.1, each  $Input_i$  ( $1 = 1, \dots, 4$ ) is a digital value.  $y_{11}$ ,  $y_{12}$ , and  $y_{13}$  are the outputs which are generated in the first, second, and third common nodes of the first layer. Each of them is converted to a 4-bit digit through a voltage-mode ADC and gets through the second layer.  $Out1$  and  $Out2$  are the final digital outputs produced in the first and second common nodes of the last layer.

The 4-3-2 network is trained off-line to solve a classification problem as is shown in Figure 6.2.

The weights are calculated in MATLAB and then loaded through the implemented network for simulations. The corresponding weights and biases referred to Figure 6.2 are as


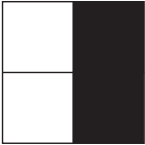
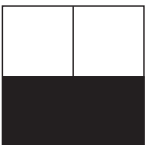



Template	Direction	Equivalent Bits (Input)	Code (Output)
	Horizontal	1100	01
	Vertical	0101	00
	Horizontal	0011	01
	Vertical	1010	00
	Right	1001	10
	Left	0110	11

Figure 6.2: Training pattern set for the 4-3-2 CVNS DNN.

follows.

$$w^1 = \begin{bmatrix} -2.6543 & 7.9883 & 11.2109 & -7.4902 \\ -10.2441 & 8.2461 & 1.1406 & -14.3066 \\ 6.4375 & 14.6484 & 1.7383 & -8.6465 \end{bmatrix}$$

$$w^2 = \begin{bmatrix} -8.7207 & 13.4648 & 1.3203 \\ 9.9570 & 10.0273 & -8.8066 \end{bmatrix}$$

$$b^1 = \begin{bmatrix} 0 \\ -8.002 \\ 4 \end{bmatrix} \quad b^2 = \begin{bmatrix} -0.2520 \\ -2.2520 \end{bmatrix}$$

The layout of the proposed 4-3-2 CVNS distributed neural network is shown in Figure 6.3. The area consumption is  $676 \times 570 \mu\text{m}^2$ .

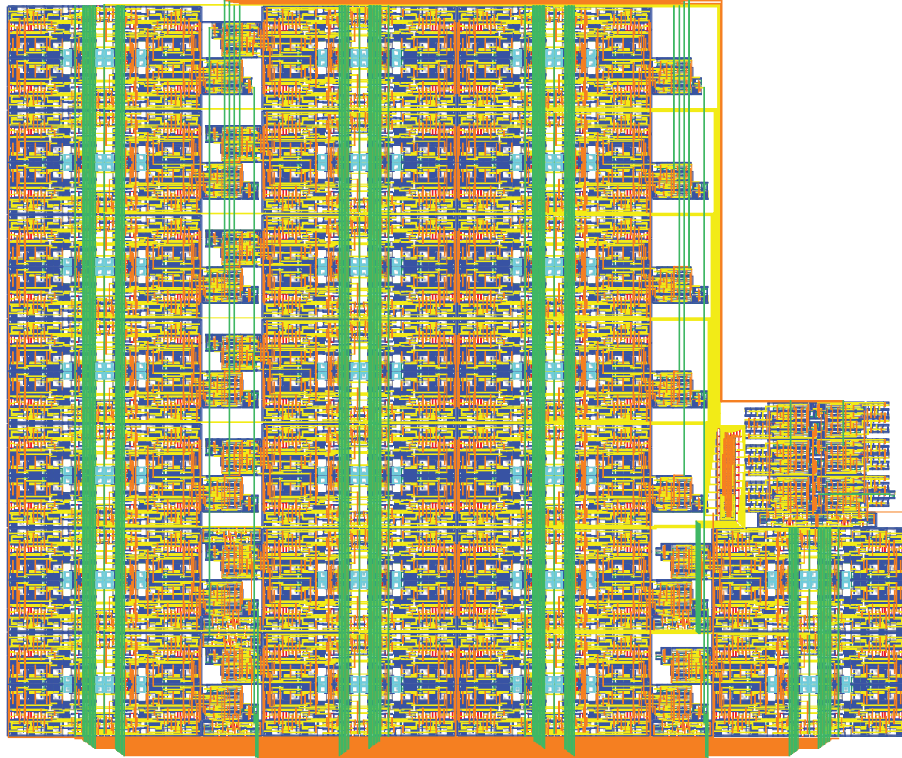


Figure 6.3: Layout of the proposed 4-3-2 CVNS DNN.



The simulation result of the trained network for different values of input is shown in Figure 6.4. Here, four square waves with different periods are applied to the inputs to cover all possible input combinations.

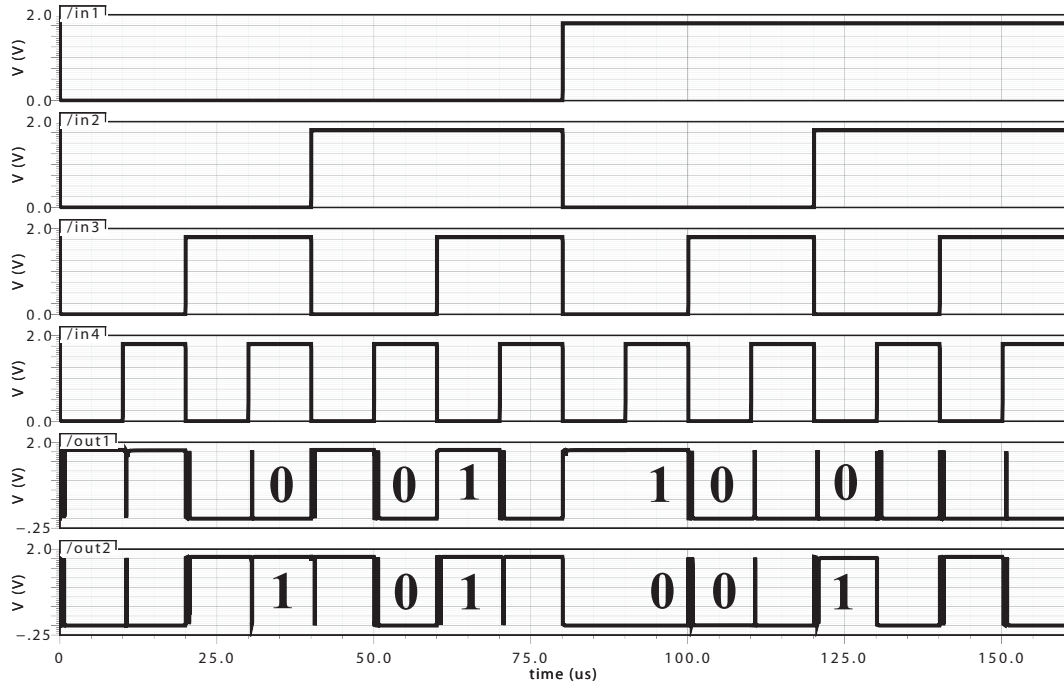


Figure 6.4: Simulation result of the 4-3-2 CVNS DNN to solve the classification problem of Figure 6.2.

As is shown in Figure 6.4, the outputs are exactly matching the expected outputs for valid combinations of inputs.

## 6.2 Comparisons

The proposed structure is a mixed-signal design where the synapse is mixed-signal and the neuron is analog.

Analog neural networks generally consumes less power compared to the digital ones; however, the reconfigurability is not achieved by analog neural networks [62, 63]. More-

over, analog networks are more sensitive to noise. The proposed network is more like analog neural networks than digital ones; therefore, the power consumption will be less than digital networks.

Nevertheless, the usage of synapse-neuron modules together with the distributed neurons make the network reconfigurable and decrease the design cost. Once designing the synapse-neuron module, it can be used in different network configurations. The inputs should be assigned, and outputs should be connected to the relevant common nodes.

Implementation of functions like summation and activation function is much more area efficient in analog networks compared to the digital ones [64, 65]. The proposed network performs all the summations in analog circuitries as well as the activation function which is realized through analog neurons. It should be noted that in digital neural networks, huge LUTs are used for realization of activation function.

Storing the weights is a major problem in analog neural networks [66]. This problem is subdued in the proposed structure by using the CVNS DRAMs where the digital values for the weights are converted to the CVNS digits and stored on multi-valued storage cells. This module also provides error correction on the stored values which will deflate the possibility of error for the stored weights.

This network also profits from the benefits of the distributed neural networks such as scalability and increased robustness [3, 9, 10]. Moreover, the CVNS redundancy increases the noise immunity furthermore.

Although the NSR is greater than the NSR of the one with complete CVNS digits [16], this is still less than the NSR of the non CVNS conventional distributed neural network [9]. Therefore, the CVNS DNN even in a low resolution environment results in better noise immunity. It should be noted that the number of sub-neurons in the truncated CVNS distributed network is equal to the number of sub-neurons in conventional DNNs.

## 6.3 Conclusion

The CVNS synapse-neuron module is used to build a 4-3-2 network which is trained by a classification pattern. The prototype is designed, simulated, and laid out in  $0.18\mu m$  CMOS technology. The proposed structure is more like an analog ANNs; however, the problem of storing the weights is solved by using the multi-valued DRAMs.

---

## **Chapter 7**

---

### ***Conclusions and Future Work***

---

#### **7.1 Conclusions**

The CVNS distributed neural network is introduced. It has all the advantages of conventional distributed neural networks such as scalability and higher noise immunity. However, the CVNS decreases the vulnerability of the proposed structure to noise.

The practical implementation of the proposed structure is limited by the resolution of the environment. Truncated methods are used to make the CVNS digits within each CVNS digit set compatible with the resolution of the environment. In case of the CMOS  $0.18\mu m$  technology, this resolution is considered equal to 4-bit. Digit link is selected as its minimum value to decrease the number of the CVNS digits. Consequently, the 13-bit weights are converted to 4 CVNS digits.

A multi-valued DRAM is also proposed based on the CVNS. This memory converts binary digits to truncated CVNS digits and stores them on the storage cells. In the proposed CVNS DRAM, a DAC is used for generating the CVNS digits. The results are then stored

on the storage cells. The redundancy between the CVNS digits, digit link, is used to detect and correct the error of leakage current over the storage cell. A fast ADC is used to convert each two bits at the same time. This ADC is used together with an XOR and a 1-bit DAC to detect and correct the error.

The multi-valued CVNS DRAM is used in the proposed synapse-neuron module to store the weights. Except for the DRAM, the synapse-neuron module contains the multiplication module, the interface module, and the neuron. The multiplication module is a mixed-signal module that multiplies the weights by the inputs. The interface module is used to apply the sign to the multiplication module output and to generate the final synapse output. Finally, analog neurons are used to realize the sigmoid function as the activation function and to generate the final output in the common node.

Using the proposed synapse-neuron module as the building block, a 4-3-2 network is designed and laid out to justify the concept. A pattern set is used to train the network off-line. Post-layout simulations are performed to test the architecture; results show the proper functionality of the network.

The proposed CVNS DNN consumes less power and area compared to digital ANNs. It also alleviates the problem of storing the weights in analog ANNs. Simulations show that the NSR of the proposed CVNS distributed network is less than the NSR of both lumped neural networks and conventional DNNs.

## 7.2 Future Work

The proposed neural network can be used in applications such as photo sensors. The proposed synapse-neuron module can be used to generate different kind of networks. However, it will need some peripherals to be designed. For example in the case of photo sensors applications, proper sensors to sense the light are inevitable. Moreover, interface circuits are necessary to match the sensors output to the input of the neural network.

In addition, the proposed structure is suggested to be used to implement an ANN with on-line training. The prototype design is working with 13-bits of weights which is the minimum number of weights which is generally necessary for on-line training. Therefore, the proposed structure has the potential of being used in a network trained on-line. In this case, additional modules and modifications are unavoidable.

The module to realize the deviation of the activation function is suggested to be designed through analog circuitries. This module is necessary for realizing the algorithms such as the back propagation algorithm used for on-line training. Nevertheless, we suggest the realization of other arithmetic modules used for on-line training through conventional analog/mixed-signal circuits. For example for the back propagation algorithm, some multiplication and addition modules are used to update the new values for both weights and biases; these modules can be realized through the circuits similar to what we have already proposed. In addition, we are suggesting to apply some methods to the whole system to synch the modules during the on-line training.

## References

- [1] E. K. F. Lee and P. G. Gulak, "Dynamic current-mode multi-valued MOS memory with error correction," in *IEEE International Symposium on Multiple-Valued Logic*, pp. 208–215, 1992.
- [2] H. Djahanshahi, *A robust hybrid VLSI neural network architecture for a smart optical sensor*. PhD thesis, University of Windsor, Windsor, Ontario, Canada, 1998.
- [3] H. Djahanshahi, M. Ahmadi, G. A. Jullien, and W. C. Miller, "A robust hybrid neural architecture for an industrial sensor application," in *IEEE International Symposium on Circuits and Systems*, pp. 41–45, November 1998.
- [4] L. Gatet, H. T. Beteille, and M. Lescure, "Analog neural network implementation for a real-time surface classification application," *IEEE Sensors Journal*, vol. 8, pp. 1413–1421, August 2008.
- [5] Y. Li, Y. Fu, S. Zhang, and H. Li, "Improved algorithm of the back propagation neural network and its application in fault diagnosis of air-cooling condenser," in *International Conference on Wavelet Analysis and Pattern Recognition*, pp. 180–184, November 2009.
- [6] S. L. Phung and A. Bouzerdoum, "A pyramidal neural network for visual pattern recognition," *IEEE Transactions on Neural Networks*, vol. 18, pp. 329–343, March 2007.
- [7] J. S. Wang and Y. P. Chen, "A fully automated recurrent neural network for unknown dynamic system identification and control," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 53, pp. 1363–1372, June 2006.
- [8] H. He, Z. Zhu, and E. Makinen, "A neural network model to minimize the connected dominating set for self-configuration of wireless sensor networks," *IEEE Transactions on Neural Networks*, vol. 20, pp. 973–982, June 2009.

- 
- [9] H. Djahanshahi, M. Ahmadi, G. A. Jullien, and W. C. Miller, "Quantization noise improvement in a hybrid distributed-neuron ANN architecture," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 48, pp. 842–846, September 2001.
- [10] H. Djahanshahi, M. Ahmadi, G. A. Jullien, and W. C. Miller, "Sensitivity study and improvements on a nonlinear resistive-type neuron circuit," *IEE Proceedings on Circuits*, vol. 147, pp. 237–242, August 2000.
- [11] S. W. Piche, "The selection of weight accuracies for madalines," *IEEE Transactions on Neural Networks*, vol. 6, p. 432445, March 1995.
- [12] A. Saed, *Continuous valued digits: a novel direction in multiple-valued arithmetic*. PhD thesis, University of Windsor, Windsor, Ontario, Canada, 1998.
- [13] A. Saed, M. Ahmadi, and G. A. Jullien, "A number system with continuous valued digits and modulo arithmetic," *IEEE Transaction on Computers*, vol. 51, pp. 1294–1304, November 2002.
- [14] G. Khodabandehloo, M. Mirhassani, and M. Ahmadi, "A 16-level current-mode CVNS memory," *IET Electronics Letters*, vol. 45, pp. 822–824, July 2009.
- [15] G. Khodabandehloo, M. Mirhassani, and M. Ahmadi, "CVNS-based storage and refreshing scheme for a multi-valued dynamic memory," *IEEE Transactions on Very Large Scale Integration Systems*, accepted in 5 pages on March 2010.
- [16] G. Khodabandehloo, M. Mirhassani, and M. Ahmadi, "Resistive-type CVNS distributed neural networks with improved noise to signal ratio," *IEEE Transactions on Circuits and Systems II*, vol. 57, no. 10, pp. 793–797, 2010.
- [17] G. Khodabandehloo, M. Mirhassani, and M. Ahmadi, "16-level current-mode multiple-valued dynamic memory with increased noise margin," in *IEEE International Symposium on Multiple-Valued Logic*, pp. 48–53, May 2009.
- [18] M. Mirhassani, M. Ahmadi, and G. A. Jullien, "Low-power mixed signal CVNS based 64-bit adder for media signal processing," *IEEE Transaction On Very Large Scale Integration Systems*, vol. 16, pp. 1141–1150, September 2008.
- [19] M. Mirhassani, M. Ahmadi, and G. A. Jullien, "Digital multiplication using continuous valued digits," in *IEEE International Symposium on Circuits and Systems*, pp. 3263–3266, May 2007.
- [20] M. Mirhassani, M. Ahmadi, and G. A. Jullien, "Robust low-sensitivity adaline neuron based on continuous valued number system," *Journal of Analog Integrated Circuits and Signal Processing*, vol. 56, pp. 223–231, September 2008.
-



- 
- [21] D. G. Nairn and C. A. T. Salama, "A current mode algorithmic analog-to-digital converter," in *IEEE International Symposium on Circuits and Systems*, pp. 2573–2576, 1988.
- [22] D. G. Nairn and C. A. T. Salama, "current mode algorithmic analog-to-digital converters," *IEEE Journal of Solid-State Circuits*, vol. 25, no. 4, pp. 997–1004, 1990.
- [23] J. Sarao and H. H. L. Kwok, "Current mode building and blocks and their application in ADC," in *IEEE Pacific Rim Conference on Communications, Computers and signal Processing*, pp. 283–286, 2001.
- [24] G. Khodabandehloo, M. Mirhassani, and M. Ahmadi, "An area-speed efficient method for current mode analog to digital converters," in *European Conference on Circuit Theory and Design*, pp. 201–204, 2009.
- [25] T. S. M. Company, "0.18 $\mu$ m TSMC CMOS technology standard cell library," September 1999.
- [26] R. Anand, K. Mehrotra, C. K. Mohan, and S. Ranka, "Efficient classification for multiclass problems using modular neural networks," *IEEE Transactions on Neural Networks*, vol. 6, pp. 117–124, January 1995.
- [27] S. J. Daubert, D. Vallancourt, and Y. P. Tsvividis, "Current copier cells," *IET Electronics Letters*, vol. 24, pp. 1560–1562, January 1998.
- [28] G. Khodabandehloo, M. Mirhassani, and M. Ahmadi, "Analog implementation of a novel resistive-type sigmoidal neuron," *IEEE Transactions on Very Large Scale Integration Systems*, accepted in 5 pages on Jan. 2011.
- [29] S. Radhakrishnan, M. Wang, and C. H. Chen, "Low-power 4-b 2.5 GSPS pipelined flash analog-to-digital converters in 0.13  $\mu$ m CMOS," in *Proceedings of the IEEE Instrumentation and Measurement Technology Conference*, pp. 287–292, 2005.
- [30] X. Zhang and M. Bayoumi, "A low power 4-bit interleaved burst sampling ADC for sub-GHz impulse UWB radio," in *IEEE International Symposium on Circuits and Systems*, pp. 1165–1168, 2007.
- [31] M. K. Agdam and A. Nabavi, "A low-power high-speed 4-bit ADC for DS-UWB communications," in *IEEE Computer Society Annual Symposium on VLSI*, pp. 506–507, 2007.
- [32] G. Cauwenberghs and A. Yariv, "Fault tolerant multilevel storage in analog VLSI," *IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing*, vol. 41, no. 12, pp. 827–829, 1994.
-

- 
- [33] R. Castello, D. D. Caviglia, M. Franciotta, and F. Montecchi, "Selfrefreshing analogue memory cell for variable synaptic weights," *IET Electronics Letters*, vol. 27, no. 20, pp. 1871–1872, 1991.
- [34] B. Hotchet, "Multivalued MOS memory for variable-synapse neural networks," *IET Electronics Letters*, vol. 25, no. 10, pp. 669–670, 1989.
- [35] B. Hotchet, V. Peiris, S. Abdo, and M. J. Declercq, "Implementation of a learning kohonen neuron based on a new multilevel storage technique," *IEEE Journal of Solid-State Circuits*, vol. 26, no. 3, pp. 262–267, 1991.
- [36] D. Matolin, J. Schreiter, S. Getzlaff, and R. Schuffny, "An analog VLSI pulsed neural network implementation for image segmentation," in *Proceedings of the international Conference on Parallel Computing in Electrical Engineering*, pp. 51–55, 2004.
- [37] T. Morie and Y. Amemiya, "An all analog expandable neural network LSI with on-chip backpropagation learning," *IEEE Transactions on Solid State Circuits*, vol. 29, pp. 1086–1093, 1994.
- [38] E. K. F. Lee and P. G. Gulak, "Current mode multivalued dynamic MOS memory with error correction," *IET Electronics Letters*, pp. 1067–1069, 1992.
- [39] E. K. F. Lee and P. G. Gulak, "Error correction technique for multi-valued MOS memory," *IET Electronics Letters*, pp. 1321–1323, 1991.
- [40] B. Polianskikh and Z. Zilic, "Design and implementation of error detection and correction circuitry for multilevel memory protection," in *IEEE International Symposium on Multiple-Valued Logic*, pp. 89–95, 2002.
- [41] G. Birk, D. G. Elliott, and B. F. Cockburn, "A comparative simulation study of four multilevel DRAMs," in *IEEE International Workshop on Memory Technology*, pp. 102–109, 1999.
- [42] T. Okuda, "Advanced circuit technology to realize post giga-bit DRAM," in *IEEE International Symposium on Multiple-Valued Logic*, pp. 2–5, 1998.
- [43] T. Okuda and T. Murotani, "A four-level storage 4-Gb DRAM," *IEEE Journal of Solid-State Circuits*, vol. 32, no. 11, pp. 1743–1747, 1997.
- [44] P. Gillingham, "A sense and restore technique for multilevel DRAM," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 43, no. 7, pp. 483–486, 1996.
-

- 
- [45] T. Furuyama, T. Ohsawa, Y. Nagahama, H. Tanaka, Y. Watanabe, T. Kimura, K. Muraoka, and K. Natori, "An experimental 2-bit/cell storage DRAM for macro cell or memory-on-logic application," in *IEEE 1988 Custom Integrated Circuits Conference*, pp. 4.4/1–4.4/4, 1988.
- [46] M. Aoki, Y. Nakagome, M. Horiguchi, S. Ikenaga, and K. Shimohigashi, "A 16-level/cell dynamic memory," *IEEE Journal of Solid-State Circuits*, vol. 22, no. 2, pp. 297–299, 1987.
- [47] M. Horiguchi, M. Aoki, Y. Nakagome, S. Ikenaga, and K. Shimohigashi, "An experimental large-capacity semiconductor file memory using 16-levels/cell storage," *IEEE Journal of Solid-State Circuits*, vol. 23, no. 1, pp. 27–33, 1988.
- [48] B. Liu, J. F. Frenzel, and R. B. Wells, "A multi-level DRAM with fast read and low power consumption," in *IEEE workshop on Microelectronics and Electron Devices*, pp. 59–62, 2005.
- [49] I. M. Thoidis, D. Soudris, J. M. Fernandez, and A. Thanailakis, "The circuit design of multiple-valued logic voltage-mode adders," in *IEEE International Symposium on Circuits and Systems*, pp. 162–165, 2001.
- [50] M. C. Mekhallalati and M. K. Ibrahim, "A new high radix maximally redundant signed digit adder," in *IEEE International Symposium on Circuits and Systems*, pp. 459–462, 1999.
- [51] M. Mirhassani, M. Ahmadi, and G. A. Jullien, "16-bit binary multiplication using high radix analog digits," in *IEEE Asilomar Conference on Signal, Systems and Computers*, pp. 332–336, 2006.
- [52] G. Wegmann and E. A. Vittoz, "Basic principles of accurate dynamic current mirrors," *IEE Processings on Circuits, Devices and Systems*, vol. 137, no. 2, pp. 95–100, 1990.
- [53] X. Yunan, B. F. Cockburn, and D. G. Elliott, "Design of a multilevel DRAM with adjustable cell capacity," in *Canadian Conference on Electrical and Computer Engineering*, pp. 295–300, 2001.
- [54] J. Kim and M. C. Papaefthymiou, "Block-based multiperiod dynamic memory design for low data-retention power," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 11, no. 6, pp. 1006–1018, 2003.
- [55] G. Khodabandehloo, M. Mirhassani, and M. Ahmadi, "Current-mode multiple-valued dynamic memory," in *IEEE International Symposium on Circuits and Systems*, pp. 3058–3061, May 2009.
-

- 
- [56] A. Agarwal, Y. B. Kim, and S. Sonkusale, "Low power current mode ADC for CMOS sensor IC," in *IEEE International Symposium on Circuits and Systems*, pp. 584–587, 2005.
- [57] V. Tipsuwanporn, A. Numsomran, W. Chuchotsakunleot, S. Chuenarom, and S. Maitreechit, "Algorithmic ADC using current mode without DAC," in *Asia-Pacific Conference on Circuits and Systems*, pp. 453–456, 2002.
- [58] B. M. Wilamowski, M. E. Sinangil, and G. Dndar, "A gray-code current mode ADC structure," in *IEEE Mediterranean Electrotechnical Conference*, pp. 35–38, 2006.
- [59] S. Chuenarom and V. Tipsuwarnpron, "Application techniques for high performance ADC," in *International Symposium on Communications and Information Technologies*, pp. 749–752, 2006.
- [60] H. Traff, "Novel approach to high speed CMOS current comparators," *IEEE Journal of Solid-State Circuits*, vol. 28, no. 3, pp. 310–312, 1992.
- [61] B. Razavi, *Design of analog CMOS integrated circuits*. McGraw Hill Higher Education, 2003.
- [62] L. Gatet, H. T. Beteille, and F. Bony, "Comparison between analog and digital neural network implementations for range-finding applications," *IEEE Transactions on Neural Networks*, vol. 20, pp. 460–470, March 2009.
- [63] V. F. Koosh and R. M. Goodman, "Analog VLSI neural network with digital perturbative learning," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 49, pp. 359–368, May 2002.
- [64] S. Satyanarayana, Y. P. Tsvividis, and H. P. Graf, "A reconfigurable VLSI neural network," *IEEE Journal of Solid-State Circuits*, vol. 27, pp. 67–81, January 1992.
- [65] S. M. Gowda, B. J. Sheu, J. Choi, C. G. Hwang, and J. S. Cable, "Design and characterization of analog VLSI neural network modules," *IEEE Journal of Solid-State Circuits*, vol. 28, pp. 301–313, March 1993.
- [66] V. F. Koosh and R. M. Goodman, "VLSI neural network with digital weights and analog multipliers," in *IEEE International Symposium on Circuits and Systems*, pp. 233–236, May 2001.
-

---

## *VITA AUCTORIS*

---

Golnar Khodabandehloo was born in Hamedan, IRAN, in 1981. She received her BSc degree in Electrical Engineering from Amirkabir University of Technology, Tehran, Iran, in 2003, and the MSc degree in Electronics from Iran University of Science and Technology, Tehran, Iran in 2005. Since September 2007, she has been pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, University of Windsor, Windsor, On, Canada. Her research interests include analog integrated circuits, hardware implementation of neural networks, and multi-valued memories for analog/mixed-signal integrated circuits.