Mixed-Signal VLSI Implementation of CVNS Artificial Neural Networks

Babak Zamanlooy
University of Windsor

Follow this and additional works at: https://scholar.uwindsor.ca/etd

Recommended Citation
https://scholar.uwindsor.ca/etd/5097

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) or by telephone at 519-253-3000ext. 3208.
Mixed-Signal VLSI Implementation of CVNS Artificial Neural Networks

by

Babak Zamanlooy

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
2014

© 2014 Babak Zamanlooy
Mixed-Signal VLSI Implementation of CVNS Artificial Neural Networks

by

Babak Zamanlooy

APPROVED BY:

________________________________________
A. Y. Yi, External Examiner
University of Michigan-Dearborn

________________________________________
J. Urbanic
Industrial and Manufacturing Systems Engineering

________________________________________
M. Ahmadi
Electrical and Computer Engineering

________________________________________
H. Wu
Electrical and Computer Engineering

________________________________________
M. Mirhassani, Advisor
Electrical and Computer Engineering

May 15, 2014
Declaration of Co-Authorship

I hereby declare that this dissertation incorporates material that is the result of research conducted under the supervision of my supervisor, Dr. M. Mirhassani. Results related to this research are reported in Chapters 2 through 6.

I am aware of the University of Windsor’s Senate Policy on Authorship and I certify that I have properly acknowledged the contributions of the other researchers to my dissertation, and I have obtained written permission from my co-author to include the aforementioned materials in my dissertation.

I certify that, with the above qualification, this dissertation, and the research to which it refers, is the product of my own work.

Declaration of Previous Publication

This dissertation includes three original papers that have been previously published/submitted for publication to peer reviewed journals. It also includes two papers that are in the process of submission to peer reviewed journals. These original papers are as follows:
<table>
<thead>
<tr>
<th>Dissertation Chapter</th>
<th>Publication title/full citation</th>
<th>Publication status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter 2</td>
<td>B. Zamanlooy and M. Mirhassani, “CVNS-Based sigmoid function evaluation for precise neurochips,”</td>
<td>To be submitted</td>
</tr>
<tr>
<td>Chapter 4</td>
<td>B. Zamanlooy and M. Mirhassani, “Mixed-Signal VLSI Neural Network Based on Continuous Valued Number System,”</td>
<td>To be submitted</td>
</tr>
<tr>
<td>Chapter 5</td>
<td>B. Zamanlooy and M. Mirhassani, “Area-efficient robust Madaline based on continuous valued number system,” Neurocomputing, <a href="http://dx.doi.org/10.1016/j.neucom.2014.03.029">http://dx.doi.org/10.1016/j.neucom.2014.03.029</a>, 2014</td>
<td>Accepted for publication</td>
</tr>
</tbody>
</table>

I certify that I have obtained a written permission from the copyright owner(s) to include the above published material(s) in my dissertation. 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 dissertation 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 dissertation, 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 dissertation.

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

In this work, mixed-signal implementation of Continuous Valued Number System (CVNS) neural network is proposed. The proposed network resolves the limited signal processing precision issue present in mixed-signal neural networks. This is realized by the CVNS addition, the CVNS multiplication and the CVNS sigmoid function evaluation algorithms proposed in this dissertation. The proposed algorithms provide accurate results in low-resolution environment.

In addition, an area-efficient low sensitivity CVNS Madaline is proposed. The proposed Madaline is more robust to input and weight errors when compared to the previously developed structures. Moreover, its area consumption is lower.

Furthermore, a new approximation scheme for hyperbolic tangent activation function is proposed. Using the proposed approximation scheme results in efficient implementation of digital ASIC neural networks in terms of area, delay and power consumption.
Dedication

To my family.
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 for her invaluable guidance and constant support throughout the course of this work.

In addition to my advisor, I would like to thank the rest of my dissertation committee: Dr. Majid Ahmadi and Dr. Huapeng Wu from the electrical and computer engineering department, Dr. Jill Urbanic from the industrial and manufacturing systems engineering department, and Dr. Alex (Ya Sha) Yi for their participation in my seminars, reviewing my dissertation, and their constructive comments.

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

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

Declaration of Co-Authorship / Previous Publication  iv

Abstract vii

Dedication viii

Acknowledgments ix

List of Figures xiv

List of Tables xvi

List of Abbreviations xviii

## 1 Introduction 1

1.1 References ........................................... 4

## 2 CVNS-Based Sigmoid Function Evaluation for Precise Neurochips 5

2.1 CVNS Addition in a Low Resolution Environment ...................... 7

2.2 Selection of Number of Input and Output CVNS Digits .................. 10

  2.2.1 Selection of Number of Input CVNS Digits ................... 10

  2.2.2 Selection of Number of Output CVNS Digits ................... 12

2.3 Proposed CVNS Sigmoid Function Evaluation Scheme ................... 12

2.4 VLSI Implementation of the CVNS Sigmoid Function Evaluation .......... 14

  2.4.1 Input Range Decoder .................................. 14

  2.4.2 Current Generator .................................... 16

  2.4.3 Sigmoid Approximation ............................. 17

  2.4.4 Output Assignment Block ............................. 19
5 Area-Efficient Robust Madaline Based on Continuous Valued Number System 70
5.1 Introduction ................................................. 70
5.2 Continuous Valued Number System (CVNS) ......................... 73
5.3 Noise to Signal Ratio of Previous Adalines ......................... 73
5.4 Mathematical Analysis of Madaline Structures ....................... 75
  5.4.1 Lumped Structure ........................................ 76
  5.4.2 Distributed Structure ..................................... 77
  5.4.3 CVNS-RE Structure ....................................... 78
  5.4.4 CVNS-DNN and CVNS-FDNN Structures ...................... 78
5.5 Proposed Architecture ......................................... 79
5.6 Comparison of the Proposed Madaline Structure with Previous Architectures ............................................. 83
  5.6.1 Comparison of the Proposed Madaline Structure with Previous Structures in the Linear Region of Stochastic Gain Function .................. 85
  5.6.2 Comparison of the Proposed Madaline Structure with Previous Structures in the Nonlinear Region of Stochastic Gain Function .................. 88
5.7 VLSI Implementation and Comparisons ............................. 91
5.8 Conclusion ................................................................ 103
5.9 References ....................................................... 105

6 Efficient VLSI Implementation of Neural Networks with Hyperbolic Tangent Activation Function 107
6.1 Introduction ..................................................... 107
6.2 Proposed Approximation Scheme ................................... 110
  6.2.1 Output Approximation in the Pass Region ................... 110
  6.2.2 Output Approximation in the Processing Region .......... 112
  6.2.3 Output Approximation in the Saturation Region ........... 113
6.3 Selection of Number of Input and Output Bits ....................... 113
  6.3.1 Selection of Number of Input Bits ......................... 113
  6.3.2 Selection of Number of Output Bits ....................... 114
6.4 Determining the Boundaries for Different Regions .................. 116
  6.4.1 Pass Region ............................................... 116
  6.4.2 Saturation Region .......................................... 117
  6.4.3 Processing Region ......................................... 117
6.5 Proposed Structure .............................................. 117
# List of Figures

2.1 The block diagram of Adaline ........................................... 11  
2.2 The block diagram of the proposed CVNS-based sigmoid function evaluation structure 15  
2.3 VLSI implementation of the current reference circuit used in the current generator block .................................................. 17  
2.4 VLSI implementation of mod $16 \mu A$ operation .......................... 19  
2.5 VLSI implementation of the output assignment unit ..................... 20  
2.6 The simulation results of the proposed CVNS-based sigmoid activation function ... 21  
2.7 Monte Carlo simulation result of the analog neuron developed in [27] for its maximum output ................................................. 25  
2.8 Monte Carlo simulation result of the $15 \mu A$ current ........................ 26  
3.1 VLSI implementation of mod$16 \mu A$ operation .......................... 42  
3.2 VLSI implementation of $1 \mu A \times \left\lfloor \frac{C_{m-4}}{16 \mu A} \right\rfloor$ operation ............................................. 45  
3.3 Block diagram of the VLSI implementation of $(z)_{-17}$ .................... 46  
3.4 Block diagram of the VLSI implementation of $(z)_{-13}$ .................... 47  
3.5 VLSI implementation of $(z)_{-9}, (z)_{-5}, (z)_{-1}, (z)_{3}$ ..................... 48  
3.6 Layout of the proposed CVNS synapse multiplier .......................... 49  
3.7 Post-layout simulation results of the proposed CVNS synapse multiplier ............. 50  
3.8 NSR of the CVNS Adaline using the proposed multiplication algorithm versus the previously developed multiplication algorithm .......................... 52  
4.1 The block diagram of the 2-2-1 CVNS network realizing the XOR function ........ 58  
4.2 VLSI implementation of the binary input to CVNS conversion, $b_i \times 8 \mu A \times 2^{i-m}$ and $1 \mu A \times \left(Ca_{m-\varphi} \text{ cmp } 16 \mu A \right)$ ............................................. 61  
4.3 VLSI implementation of the first layer multiplier .......................... 62
### LIST OF FIGURES

<table>
<thead>
<tr>
<th>Number</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.4</td>
<td>Block diagram of the CVNS adder</td>
<td>64</td>
</tr>
<tr>
<td>4.5</td>
<td>VLSI implementation of the output to binary converter</td>
<td>65</td>
</tr>
<tr>
<td>4.6</td>
<td>Layout of the implemented network</td>
<td>66</td>
</tr>
<tr>
<td>4.7</td>
<td>Layout of the chip sent for fabrication</td>
<td>67</td>
</tr>
<tr>
<td>4.8</td>
<td>Post-layout simulation results of the proposed CVNS network</td>
<td>67</td>
</tr>
<tr>
<td>5.1</td>
<td>(a) Ideal stochastic gain function and its approximation (b) Approximation error of the stochastic gain function</td>
<td>74</td>
</tr>
<tr>
<td>5.2</td>
<td>NSR flow diagram of an Adaline</td>
<td>75</td>
</tr>
<tr>
<td>5.3</td>
<td>Madaline general configuration</td>
<td>76</td>
</tr>
<tr>
<td>5.4</td>
<td>Block diagram of the proposed Adaline structure</td>
<td>81</td>
</tr>
<tr>
<td>5.5</td>
<td>(a) NSR flow diagram of the proposed Adaline (b) NSR flow diagram of the proposed CVNS-distributed Madaline</td>
<td>82</td>
</tr>
<tr>
<td>5.6</td>
<td>Block diagram of the proposed CVNS-distributed Madaline</td>
<td>82</td>
</tr>
<tr>
<td>5.7</td>
<td>Normalized Neuron×NSR improvement of different architectures compared to distributed structure</td>
<td>88</td>
</tr>
<tr>
<td>5.8</td>
<td>Block diagram of the 8-bit CVNS multiplier</td>
<td>93</td>
</tr>
<tr>
<td>5.9</td>
<td>VLSI implementation of the mod16µA operation</td>
<td>94</td>
</tr>
<tr>
<td>5.10</td>
<td>Simulation results of the mod16µA circuit</td>
<td>95</td>
</tr>
<tr>
<td>5.11</td>
<td>(a) VLSI implementation of the neuron (b) Output versus input of the implemented neuron</td>
<td>96</td>
</tr>
<tr>
<td>5.12</td>
<td>VLSI implementation of the the RE unit</td>
<td>97</td>
</tr>
<tr>
<td>5.13</td>
<td>Simulation results of the the RE circuit</td>
<td>99</td>
</tr>
<tr>
<td>5.14</td>
<td>Layout of the proposed Adaline</td>
<td>99</td>
</tr>
<tr>
<td>6.1</td>
<td>Different regions of hyperbolic tangent function</td>
<td>111</td>
</tr>
<tr>
<td>6.2</td>
<td>Hyperbolic tangent function derivative</td>
<td>114</td>
</tr>
<tr>
<td>6.3</td>
<td>Block diagram of the proposed structure</td>
<td>118</td>
</tr>
<tr>
<td>6.4</td>
<td>(a) approximation error (b) quantization error (c) total error (d) ideal and approximated output</td>
<td>121</td>
</tr>
<tr>
<td>6.5</td>
<td>Hardware implementation of the considered example</td>
<td>123</td>
</tr>
<tr>
<td>6.6</td>
<td>Three layer Madaline general configuration</td>
<td>127</td>
</tr>
<tr>
<td>6.7</td>
<td>Block diagram of the implemented network</td>
<td>127</td>
</tr>
<tr>
<td>6.8</td>
<td>Optical input patterns and their related class</td>
<td>128</td>
</tr>
</tbody>
</table>
List of Tables

2.1 Input range decoder truth table ........................................ 16
2.2 Current values generated by the current generator block .......... 18
2.3 Input values and their corresponding input and output CVNS digits .... 22
2.4 Comparison of different structures .................................... 22

3.1 Transistor sizes of the mod16µA circuit .......................... 43
3.2 Parameters of the structure shown in Fig. 3.5 for different output CVNS digits ... 49
3.3 Synapse multiplier input values and their corresponding output .......... 50

4.1 Required resolution of different arithmetic units .................... 59
4.2 Area, delay and power consumption of the implemented network .......... 68

5.1 NSR of the previous Adalines ........................................ 75
5.2 Total number of neurons required for different Madaline structures and their normalized value with respect to lumped structure .................. 84
5.3 NSR of different structures and their normalized value with respect to lumped structure in linear region of stochastic gain function ................. 86
5.4 Normalized neuron×NSR of different structures in linear region of stochastic gain function .................................................. 87
5.5 NSR of different structures and their normalized value with respect to lumped structure in nonlinear region of stochastic gain function ................. 90
5.6 Normalized neuron×NSR of different structures in nonlinear region of stochastic gain function .................................................. 91
5.7 Transistor sizes of the mod16µA circuit .......................... 95
5.8 Transistor sizes of the neuron circuit .............................. 96
<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.9</td>
<td>Transistor sizes of the RE circuit</td>
<td>98</td>
</tr>
<tr>
<td>5.10</td>
<td>Area consumption of different cells required for implementation of different structures</td>
<td>100</td>
</tr>
<tr>
<td>5.11</td>
<td>Total number of different cells required for implementation of Adalines used in different structures</td>
<td>101</td>
</tr>
<tr>
<td>5.12</td>
<td>Area consumption of different Adaline structures</td>
<td>102</td>
</tr>
<tr>
<td>5.13</td>
<td>Area consumption, NSR and area×NSR of different Madaline structures investigated in this case study</td>
<td>103</td>
</tr>
<tr>
<td>6.1</td>
<td>Output value for different input ranges and sub-ranges</td>
<td>120</td>
</tr>
<tr>
<td>6.2</td>
<td>Input range decoder</td>
<td>122</td>
</tr>
<tr>
<td>6.3</td>
<td>Bit-level mapping for the input range 1 &lt; x &lt; 2</td>
<td>124</td>
</tr>
<tr>
<td>6.4</td>
<td>Bit-level mapping for the input range 0.5 &lt; x &lt; 1</td>
<td>124</td>
</tr>
<tr>
<td>6.5</td>
<td>Comparison of different structures for ϵ=0.04</td>
<td>125</td>
</tr>
<tr>
<td>6.6</td>
<td>Comparison of different structures for ϵ=0.02</td>
<td>126</td>
</tr>
<tr>
<td>6.7</td>
<td>Comparison of network implementation</td>
<td>129</td>
</tr>
</tbody>
</table>
## List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adaline</td>
<td>Adaptive Linear Neuron.</td>
</tr>
<tr>
<td>ADP</td>
<td>Area $\times$ Delay $\times$ Power.</td>
</tr>
<tr>
<td>ASIC</td>
<td>Application Specific Integrated Circuit.</td>
</tr>
<tr>
<td>CMOS</td>
<td>Complementary Metal-Oxide-Semiconductor.</td>
</tr>
<tr>
<td>CVNS</td>
<td>Continuous Valued Number System.</td>
</tr>
<tr>
<td>DNN</td>
<td>Distributed Neural Network.</td>
</tr>
<tr>
<td>FDNN</td>
<td>Fully Distributed Neural Network.</td>
</tr>
<tr>
<td>FPGA</td>
<td>Field Programmable Gate Array.</td>
</tr>
<tr>
<td>IEEE</td>
<td>Institute of Electrical and Electronics Engineers.</td>
</tr>
<tr>
<td>LUT</td>
<td>LookUp Table.</td>
</tr>
<tr>
<td>Madaline</td>
<td>Multiple Adaline.</td>
</tr>
<tr>
<td>NSR</td>
<td>Noise-to-Signal-Ratio.</td>
</tr>
<tr>
<td>PLAN</td>
<td>PieceWise Linear Approximation of a Nonlinear Function.</td>
</tr>
<tr>
<td>PWL</td>
<td>PieceWise Linear.</td>
</tr>
<tr>
<td>RALUT</td>
<td>Range Addressable Lookup Table.</td>
</tr>
<tr>
<td>RE</td>
<td>Reverse Evolution.</td>
</tr>
<tr>
<td>TSMC</td>
<td>Taiwan Semiconductor Manufacturing Company.</td>
</tr>
<tr>
<td>VLSI</td>
<td>Very Large Scale Integration.</td>
</tr>
<tr>
<td>XOR</td>
<td>Exclusive OR.</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

VLSI implementation of neural networks has been exploited in various applications. Examples include pattern recognition [1], test of analog circuits [2], real-time surface discrimination [3] and smart sensing [4].

VLSI implementation methods of neural networks may be classified as analog, digital or mixed-signal. In the analog neural networks, both weight storage and processing are carried out using analog circuits. In the digital implementation of a neural network, both weight storage and processing are conducted in the digital domain.

When implemented by analog circuits, neural networks typically possess a higher energy efficiency, require less area and lower number of interconnections in comparison with their identical digital implementation. However, the resolution of analog implementations is lower and their design and test is more challenging.

The third implementation method, mixed-signal, exploits digital registers for weight storage and analog circuits for signal processing. This method profits from the ease of weight storage in digital registers while capitalizing on the advantages of the analog method.

One of the main barriers in using the mixed-signal method is the limited precision of the analog signal which is limited by the precision of the analog circuits used. The precision of the analog circuits is referred as the environment resolution in the rest of this dissertation.

To use the benefits of the analog circuits while maintaining the accuracy high, the Continuous Valued Number System (CVNS) can be exploited. The CVNS is an analog number system. In this
number system, a real number is represented by a set of analog digits [5]. The information represented by each CVNS analog digit is the same as the environment resolution. However, collectively a set of digits can increase the analog processing precision. This makes the CVNS a suitable candidate for implementing high precision analog and mixed-signal circuits [6–8].

The basic building blocks of an Adaline are adder, multiplier, and nonlinear activation function. To exploit the CVNS features for implementing high precision mixed-signal neural networks, CVNS addition, CVNS sigmoid function evaluation and CVNS multiplication algorithms are proposed in this dissertation.

The proposed CVNS addition algorithm makes the CVNS addition in a low-resolution environment feasible. Moreover, it provides the result in CVNS format as well as binary format. This feature is being exploited in development of the proposed CVNS sigmoid function evaluation.

The proposed CVNS sigmoid function evaluation is based on the piecewise linear approximation and provides a high precision output. Furthermore, the proposed sigmoid function evaluation method requires lower number of output digits for the same maximum approximation error when compared to the state of the art. This feature in combination with using current-mode mixed-signal circuits results in an optimal ASIC implementation of the sigmoid function.

A new CVNS multiplication algorithm for low-resolution environment is proposed with accurate results. In the proposed CVNS multiplication algorithm, the multiplier is in binary format while the multiplicand is in the CVNS format. This makes the proposed multiplication algorithm suitable for mixed-signal neural network implementations. Using the proposed multiplication algorithm, VLSI implementation of a 16×8 CVNS synapse multiplier is realized.

Using the proposed CVNS algorithms, a 2-2-1 mixed-signal CVNS network structure is proposed. The proposed structure realizes the two input XOR function. The CVNS features have been used to address the limited signal processing precision issue in mixed-signal networks. As a result, the proposed network meets the signal processing resolution requirements of neural networks. The implemented network is sent for fabrication through Canadian Microelectronics Corporation (CMC).

Attaining a low Noise-to-Signal Ratio (NSR) is one other major concern in VLSI implementation of neural networks. The NSR is indicator of the effect of input and weight errors on the network output. A network with lower NSR is more robust against input and weight errors.

A new area-efficient robust mixed-signal CVNS Madaline is proposed in this work. The proposed architecture stores the weights in digital registers while the processing is carried out using CVNS. Using digital registers for weight storage, eliminates the need for complex analog memory units exploited in previous CVNS neural network structures [8–10]. Moreover, the proposed network
improves upon in terms of both the NSR and the required number of neurons for a specific NSR.

In the final part of this work, efficient VLSI implementation of digital neural networks is studied. Design and test of the digital neural networks is easier. Moreover, they can provide higher resolution when compared to the analog implementations. However, implementation of nonlinear activation functions in digital networks is challenging.

An approximation method for digital implementation of hyperbolic tangent activation function is proposed in this work. The approximation is based on a mathematical analysis taking into consideration the maximum allowable approximation error as design parameter. VLSI implementation of the hyperbolic tangent activation function for maximum approximation errors of $\epsilon = 0.02$ and $\epsilon = 0.04$ is realized. Using the proposed approximation method, a 4-3-2 digital network is implemented. The implemented network is capable of recognizing six different input patterns. Post layout simulation results show that the proposed structure results in an efficient neural network VLSI implementation in terms of area, delay and power consumption.

All of the circuitries in this work are designed, simulated, and laid out in 0.18$\mu$m TSMC CMOS technology using a power supply voltage of 1.8V.

The next chapters are organized as follows. The proposed CVNS addition algorithm, CVNS sigmoid function evaluation and its VLSI implementation are provided in Chapter 2. The proposed CVNS multiplication algorithm and VLSI implementation of a 16×8 synapse multiplier are discussed in Chapter 3. The proposed 2-2-1 mixed-signal CVNS network structure and its VLSI implementation are explained in Chapter 4. The proposed area-efficient robust Madaline is discussed in Chapter 5. Efficient VLSI implementation of digital neural networks with hyperbolic tangent activation function is explained in Chapter 6. Finally, conclusions are drawn in Chapter 7.
1.1 References


Chapter 2

CVNS-Based Sigmoid Function

Evaluation for Precise Neurochips

Hardware implementation of neural networks has been used in a wide range of analog and digital signal processing applications [1–5].

Although different activation functions can be used [6], for the networks trained by the backpropagation algorithm, sigmoid and hyperbolic tangent are widely used.

The sigmoid function, \( S(x) \), is an S-shaped function which its output is in the (0,1) range, evaluated using the following equation:

\[
S(x) = \frac{1}{1 + e^{-x}}
\]

(2.1)

The exponentiation and division terms present make the hardware implementation a challenging task. Approximation methods [7–22] are used in order to overcome the problems associated with the direct realization of the sigmoid activation function.

Neurochips with on-chip learning need the sigmoid activation function to have an input range of (-8,8), with at least 8-bit output precision and with a maximum approximation error of 0.02 [23–26].

Digital neurons can provide the high resolution required in the neural network. The approximation methods used for digital sigmoid function evaluation may be classified as PieceWise Linear (PWL) approximation [7–13], piecewise nonlinear approximation [14–17], LookUp Table (LUT) [18], bit-level mapping [19] and hybrid methods [20–22].
Generally, in PWL methods, the input range is divided to different segments using linear approximation in each segment [7–13]. PWL-based hardware implementations in [7–10] require several multipliers which results in high area consumption and delay. In multiplierless structures [11–13], linear approximation coefficients are powers of two. Therefore, multipliers have been replaced with shift registers, which in turn decreases the area and delay significantly.

Piecewise nonlinear approximation [14–17] is similar to the PWL-based methods. The main difference is that a nonlinear approximation is used in each segment. This method also requires several multipliers which tend to have high area consumption and delay.

In the LUT-based method [18], the input range is divided to equal sub-ranges and the output corresponding to each different input range is stored in LUT. Generally, the amount of memory required for LUT-based method increases exponentially as the maximum approximation error decreases. Considering the low approximation error required for on-chip neurochips, this method is impractical for such an area limited application.

In [19], the bit-level mapping method is implemented using purely combinational circuits. However, its input and output resolution is less than the resolution required for neurochips with on-chip learning.

Hybrid methods [20–22] exploit a combination of previously mentioned methods to implement the hyperbolic tangent activation function, which is slightly different from the sigmoid function. The structures developed in [20] and [21] use a combination of PWL and LUT-based methods while the structure developed in [22] is based on the PWL in combination with bit-level mapping. However, as far as we know, there is no recently developed hybrid structure for sigmoid activation function implementation.

The digital neuron methods described above provide the resolution required at the cost of more area and power consumption. Analog neurons have lower area consumption [27–30] when compared to the digital neurons. However, their precision is limited. Therefore, analog neurons in general cannot meet these requirements. To use the advantages of analog circuits while keeping the accuracy high, alternative arithmetic can be employed.

The Continuous Valued Number System (CVNS) [31] is a candidate for such application. The CVNS is an analog number system with multiple analog digits which is suitable for implementing high precision analog and mixed-signal circuits [32–34]. The focus of this paper is ASIC implementation of the sigmoid function for neurochips with on-chip learning.

In this paper, a new CVNS addition algorithm is proposed. The proposed algorithm is used for the development of an efficient CVNS-based sigmoid function evaluation. The proposed function
evaluation method is based on the PWL approximation. Furthermore, the ASIC implementation of the proposed structure in TSMC 0.18µm technology is carried out. The proposed structure provides an 8-bit input and output resolution and has a low maximum approximation error. Moreover, it is area and delay efficient.

This paper is organized as follows. A new CVNS addition algorithm for low resolution environment is introduced in section 2.1. The mathematical analysis for arithmetic setup of CVNS digits is provided in section 2.2. The proposed CVNS-based sigmoid function evaluation scheme is discussed in section 2.3. The detailed proof of the proposed CVNS-based sigmoid function evaluation scheme is provided in the Appendix. The proposed CVNS-based sigmoid function evaluation structure and its VLSI implementation are explained in section 2.4. Post-layout simulation and comparison with existing structures is carried out in section 2.5. Finally, conclusions are drawn in section 2.6.

2.1 CVNS Addition in a Low Resolution Environment

The absolute value of a real number $X$ using fixed-point binary number system format can be shown as follows:

$$X = \sum_{i=-N_f}^{N_i-1} x_i \times 2^i$$  \hspace{1cm} (2.2)

where $N_i$ and $N_f$ are the number of integer and fractional digits, while $x_i$ is a binary digit.

The radix-2 CVNS digits of $X$ can be obtained as follows [31]:

$$\left\langle X \right\rangle_m = \left( X \times 2^{-m} \right) \mod 2$$ \hspace{1cm} (2.3)

where $m$ is the index of the CVNS digits and is within the $-N_f \leq m \leq N_i$ and mod 2 is a continuous modular reduction operation.

A CVNS digit $\left\langle X \right\rangle_m$ in a limited resolution environment can be truncated to the following equation [34]:

$$\left\langle X \right\rangle_m = \sum_{i=m-\varphi+1}^{m} x_i \times 2^{i-m}$$ \hspace{1cm} (2.4)

where $\varphi$ is the environment resolution.

Example: Finding the CVNS digit set of $X=1010.1111$ for a limited environment resolution of four

Using (2.4), $X$ is mapped to the CVNS digit set $\left\langle X \right\rangle = \{1.25, 0.625, 1.375, 0.875|1.875, 1.625, 1.5, 1\}$. 
The radix-2 CVNS digits can be converted back to binary digits using the following equation:

\[ x_m = \begin{cases} 
1 & \langle X \rangle_m \geq 1 \\
0 & \langle X \rangle_m < 1 
\end{cases} \quad (2.5) \]

Example: Finding the binary digits of \( \langle X \rangle = \{1.5, 1.125, 0.25, 0.625, 1.25, 0.5, 1, 0\} \)

Using (2.5), the CVNS digit set is mapped to the binary number \( x = 11001010 \).

According to (2.3), the CVNS digits \( \langle Z \rangle_m \) representing the addition of two numbers, \( X \) and \( Y \), in the CVNS format can be written in the following form:

\[ \langle Z \rangle_m = \left( (X + Y) \times 2^{-m} \right) \mod 2 \quad (2.6) \]

Using (2.2), (2.6) can be written in the following form:

\[ \langle Z \rangle_m = \left( \sum_{i=-N_f}^{N_i-1} (x_i + y_i) \times 2^{i-m} \right) \mod 2 \quad (2.7) \]

The relation in (2.7) can be written as summation of two terms in the following form:

\[ \langle Z \rangle_m = \left( \sum_{i=-N_f}^{m} (x_i + y_i) \times 2^{i-m} \right) \mod 2 + \left( \sum_{i=m+1}^{N_i-1} (x_i + y_i) \times 2^{i-m} \right) \mod 2 \quad (2.8) \]

Considering that all the terms in the second summation are even values and are greater than or equal to two, applying a mod2 operation on those terms results in zero. Therefore, (2.8) can be written in the following form:

\[ \langle Z \rangle_m = \left( \sum_{i=-N_f}^{m} (x_i + y_i) \times 2^{i-m} \right) \mod 2 \quad (2.9) \]

To satisfy the limitations of implementation environment, method of truncation [34] can be applied. The \( \langle Z \rangle_m \) can be written as multiple summation terms, where each term has maximum \( \varphi \) terms. Therefore, \( \langle Z \rangle_m \) is rewritten in the following form:

\[ \langle Z \rangle_m = \left( \sum_{i=-N_f}^{m} (x_i + y_i) \times 2^{i-m} + 2^{-\varphi} \sum_{i=m-2\varphi+1}^{m-\varphi} (x_i + y_i) \times 2^{i-(m-\varphi)} + \cdots 
+ 2^{-((n_m-1)\varphi)} \sum_{i=-N_f}^{m-n_m\varphi} (x_i + y_i) \times 2^{i-(m-n_m\varphi)} \right) \mod 2 \quad (2.10) \]

The value of \( n_m \) can be found based on the following condition:

\[ m - n_m \varphi \leq -N_f + \varphi - 1 \quad (2.11) \]
which results in the following equation:

\[
 n_m = \left\lceil \frac{m + N_f - \varphi + 1}{\varphi} \right\rceil \quad (2.12)
\]

where \([\ ]\) is the ceiling function.

One of the most fundamental properties of the CVNS is that the digits have information overlap with each other. This means that a higher index digit can be constructed partially from lower index digits. A lower index digit of the summation, \(\langle Z \rangle_{m-\varphi}\), by repeating the same process can be written as follows:

\[
\langle Z \rangle_{m-\varphi} = C_{m-\varphi} \mod 2 = \left( \sum_{i=m-2\varphi+1}^{m-\varphi} (x_i + y_i) \times 2^{i-(m-\varphi)} \right) \mod 2 \quad (2.13)
\]

Using (2.13), (2.10) can be modified as follows:

\[
\langle Z \rangle_m = \left( \sum_{i=m-\varphi+1}^{m} (x_i + y_i) \times 2^{i-m} + 2^{-\varphi} C_{m-\varphi} \right) \mod 2 \quad (2.14)
\]

To satisfy the requirement of the limited resolution environment, only the terms of \(\langle Z \rangle_{m-\varphi}\) greater than or equal to \(2^{-\varphi-1}\) are considered. Therefore, (2.14) by applying the principle of truncation method [34], can be written in the following form:

\[
\langle Z \rangle_m = \left[ \langle X \rangle_m + \langle Y \rangle_m + 2^{-\varphi} \left\lfloor \frac{C_{m-\varphi}}{2} \right\rfloor \right] \mod 2 \quad (2.15)
\]

where \([\ ]\) is the floor function and \(\langle X \rangle_m\) and \(\langle Y \rangle_m\) are based on (2.4).

Equation (2.15) shows that the CVNS addition in a low-resolution environment is feasible. This feature is being used in development of the proposed CVNS function evaluation in the next sections.

Considering the fact that the CVNS digits share information, a full CVNS digit set is not required. The remaining digits can always be obtained from the reduced digit set.

Any higher index digit can provide information of up to \(\varphi\) digits as follows:

\[
\langle Z \rangle_{m-1} = 2^\varphi \langle Z \rangle_m \mod 1 + 2^{-(\varphi-1)} \times \langle Z \rangle_{m-(\varphi+1)} \quad (2.16)
\]

Since \(\langle Z \rangle_m < 2\) and using (2.5), the term \(\langle Z \rangle_m \mod 1\) can be modified as follows:

\[
\langle Z \rangle_m \mod 1 = \langle Z \rangle_m - \lfloor \langle Z \rangle_m \rfloor \quad (2.17)
\]

where \(\lfloor \langle Z \rangle_m \rfloor = z_m\) and \(z_m\) is the m-th binary digit corresponding to the CVNS digit \(\langle Z \rangle_m\).
Thus, the CVNS addition algorithm provides the addition result in CVNS format as well as binary format. To clarify the proposed CVNS-based addition algorithm, an example is provided.

Example: Finding the addition result of two CVNS numbers \((X) = \{(X)_3 = 1.25, (X)_{-1} = 1\}\) and \((Y) = \{(Y)_3 = 0.5, (Y)_{-1} = 1.5\}\).

Addition result is obtained in CVNS format. The addition result denoted by \((Z)\), consists of the digit set \((Z) = \{(Z)_3, (Z)_2, (Z)_0, (Z)_{-1}, (Z)_{-2}, (Z)_{-3}, (Z)_{-4}\}\).

The CVNS digits \((Z)_3\) and \((Z)_{-1}\) are obtained first as previously explained, which are equal to 1.875 and 0.5. Afterwards, the remaining CVNS digits are generated. The results will be \((Z) = \{1.875, 1.625, 1.5, 1.25, 0.5, 1, 0, 0\}\). Furthermore, as it is shown, the proposed CVNS addition algorithm provides the addition result in binary format as well, which is \(Z = 1111.0100\).

The basic element of neural network is Adaline. The block diagram of an Adaline is shown in Fig. 2.1. As can be seen from Fig. 2.1, the addition results are the inputs to the nonlinear activation function. Therefore, the inputs to the CVNS-based sigmoid function are available both in binary and CVNS format. This is exploited in the proposed function evaluation method.

### 2.2 Selection of Number of Input and Output CVNS Digits

To develop a CVNS sigmoid function evaluation, the optimum number of input and output CVNS digits should be determined. In this section, a mathematical analysis is presented which addresses this need.

Since each CVNS digit has the information equal to \(\varphi\) bits, the number of required input and output CVNS digits depends on the range of representation, maximum approximation error, and the environment resolution.

#### 2.2.1 Selection of Number of Input CVNS Digits

According to (2.2), the maximum value representable by a number in binary format is \(2^{N_i}\). Therefore, to cover the input range, it is required to have:

\[
2^{N_i} \geq r
\]

where \(r\) is the input range.

In comparison, the number of fractional part bits is ascertained by the maximum approximation error. It should be noted that \(S(x_1)\) can be used as the approximation of the input range between two consecutive points \(x_1\) and \(x_2\) having an error lower than the maximum approximation error.
provided that the following equation is satisfied.

\[ S(x_2) - S(x_1) \leq \epsilon \]  

(2.19)

where \( S(x) \) is defined in (2.1) and \( \epsilon \) is the maximum allowable approximation error.

The sigmoid change between two consecutive inputs is proportional to the sigmoid derivative. The maximum derivative of sigmoid function occurs in the region close to the origin. In this region, the output of the sigmoid activation function using the Taylor series can be approximated as follows:

\[ \lim_{x \to 0} S(x) = \frac{x}{4} \]  

(2.20)

Based on (2.20), sigmoid output is approximately equal to its input divided by four in this region and therefore, (2.19) can be simplified as follows:

\[ \frac{x_2 - x_1}{4} \leq \epsilon \]  

(2.21)

The number of bits used for representing the fractional part of the input determines the difference between two consecutive points in the input and is equal to \( 2^{-N_f} \). The relation between \( N_f \) and
maximum approximation error can be obtained as follows:

\[ 2^{-N_f} \leq 4 \epsilon \]  

(2.22)

Using (2.18) and (2.22), number of input bits, \( N_{in} \), required for representation of a signed input is as follows:

\[ N_{in} = N_i + N_f + 1 = \left\lceil \frac{\ln r_i}{\ln 2} \right\rceil + \left\lceil -\frac{\ln 4\epsilon}{\ln 2} \right\rceil + 1 \]  

(2.23)

To satisfy the requirements of on-chip neurochips [23–26], the value of \( N_i \) is equal to 3 for an input range of (-8,8). The number of fractional bits, \( N_f \), for a maximum error of 0.02 is equal to 4. Therefore the total number of input bits for the CVNS activation function is 8.

2.2.2 Selection of Number of Output CVNS Digits

The output of the sigmoid function is in the range of (0,1) and can be shown in the following form:

\[ y = \sum_{k=-N_{out}}^{-1} y_k \times 2^k \]  

(2.24)

where \( N_{out} \) is the number of bits required to represent the output.

Based on the setup discussed in section 2.1, each CVNS digit contains the information corresponding to \( \varphi \) bits. Therefore, a reduced number of digits is required for the output of CVNS sigmoid function. This can be obtained as follows:

\[ N_{outc} = \frac{N_{out}}{\varphi} \]  

(2.25)

where \( N_{outc} \) is the number of output digits in CVNS format.

An 8-bit output resolution is required for neurochips with on-chip learning [23]. The simulation results of [32–34] shows that implementation of CVNS in TSMC 0.18\( \mu \)m and 90nm with \( \varphi = 4 \) is feasible. Since the same technology is used in this paper, the same value of \( \varphi \) is used. Therefore, \( N_{out} = 8 \) and with \( \varphi = 4 \), \( N_{outc} = 2 \). Accordingly, the CVNS digit set \( \{ (y)_{-1}, (y)_{-5} \} \) is used for representing the sigmoid activation function output. It may worth noting that \( (y)_{-1} \) contains the information of \( y_{-1} \) to \( y_{-4} \) while \( (y)_{-5} \) contains the information of \( y_{-5} \) to \( y_{-8} \).

2.3 Proposed CVNS Sigmoid Function Evaluation Scheme

In this section, the proposed CVNS sigmoid function evaluation is discussed. The number of input and output digits of the proposed CVNS function are determined based on the analysis performed
in section 2.2. Also, the input to the CVNS sigmoid function coming from the adder is available in both binary and CVNS format.

Since multiplierless PWL-based methods [11–13] are the most efficient structures, a similar approximation methodology is adopted in CVNS. The method developed in [13] has a lower number of input regions and a constant number of bit shifts in each region. This leads to a less complex CVNS sigmoid function. Therefore, the approximation developed in [13] is used for the CVNS sigmoid function evaluation.

The proposed CVNS-based function evaluation scheme is explained and proved in detail in the Appendix. Based on the arithmetic setup conducted in section 2.2, the CVNS digits \((\langle y \rangle)_{-1}\) and \((\langle y \rangle)_{-5}\) can represent the output of CVNS sigmoid function. The output CVNS digits in each region are summarized as shown in the following equation:

\[
(\langle y \rangle) = \begin{cases}
(\langle y \rangle)_{-11} = 1.875 & x \geq 5 \\
(\langle y \rangle)_{-51} = 1.875 \\
(\langle y \rangle)_{-12} = \frac{(\langle x \rangle)_{2 \text{ cmp } 0.75} + 1.75}{8} & 2.375 \leq x < 5 \\
(\langle y \rangle)_{-52} = (\langle x \rangle)_{0} + 1 \text{ mod } 2 \\
(\langle y \rangle)_{-13} = (\langle x \rangle)_{2} + 1.25 & 1 \leq x < 2.375 \\
(\langle y \rangle)_{-53} = (\langle x \rangle)_{-2} \\
(\langle y \rangle)_{-14} = (\langle x \rangle)_{1} + 1 & 0 \leq x < 1 \\
(\langle y \rangle)_{-54} = (\langle x \rangle)_{-3} \\
(\langle y \rangle)_{-1x} = 1.875 - (\langle y \rangle)_{-1|x|} & x < 0 \\
(\langle y \rangle)_{-5x} = 1.875 - (\langle y \rangle)_{-5|x|} & x < 0
\end{cases}
\tag{2.26}
\]

It should be noted that \((\langle y \rangle)_{ij}\) is used as an indicator of the output CVNS digits in each region, where \(i\) and \(j\) represent the CVNS digit and the region index respectively. Also, \((\langle x \rangle)_{2 \text{ cmp } 0.75}\) is a comparator operator and is evaluated as follows:

\[
(\langle x \rangle)_{2 \text{ cmp } 0.75} = \begin{cases}
1 & (\langle x \rangle)_{2} \geq 0.75 \\
0 & (\langle x \rangle)_{2} < 0.75
\end{cases}
\tag{2.27}
\]

The mathematical derivation provides the output CVNS digits \((\langle y \rangle)_{-1}\) and \((\langle y \rangle)_{-5}\) in all four different input regions as well as negative input values.

The main arithmetic operation performed in (2.26) is addition, which considering the analog nature of CVNS digits in combination with current-mode realization, results in an efficient VLSI
implementation of the proposed CVNS-based sigmoid activation function. Moreover, as the inputs to the CVNS-based sigmoid function unit are available in both CVNS and binary format, there is no need for an input to CVNS format conversion.

2.4 VLSI Implementation of the CVNS Sigmoid Function Evaluation

In this section, the proposed structure and the VLSI implementation of the CVNS-based sigmoid function is discussed. Since in the current-mode circuits addition is easily performed, VLSI implementation of the proposed CVNS function evaluation is carried out using current-mode circuits.

The block diagram of the proposed CVNS-based sigmoid function evaluation structure is shown in Fig. 2.2. It is composed of four main units including the input range decoder, current generator, sigmoid approximation and output assignment.

The input range decoder detects the input range while the current generator provides the required signals for the sigmoid approximation and output assignment units. Since there are four input regions, the sigmoid approximation unit is constituted of four sub-units. It should be noted that the sub-units are enabled by the en signal, which is activated by the input range decoder output signals, \( r_1 \) to \( r_4 \).

The output assignment unit assigns the output based on the input sign. If the input is positive, it passes its input without change. Otherwise, a subtraction is performed. Operation of each block is explained in detail next.

2.4.1 Input Range Decoder

Since there are four different input ranges, four signals \( r_1, r_2, r_3 \) and \( r_4 \) are used to decode the input range. The truth table of the input range decoder is shown in Table 2.1.

In each region, only the associated signal becomes active. These signals can be generated based on the binary inputs to the CVNS sigmoid function structure. Using the binary inputs, the \( r_1 \) to \( r_4 \) signals can be generated based on the following logic equations.

\[
r_1 = \overline{x_2} \land (x_1 \lor \overline{x_0}) \quad (2.28)
\]

\[
r_2 = \overline{r_1} \land (x_2 \lor (x_1 \land (x_0 \lor (x_{-1} \lor x_{-2} \lor (x_{-3} \lor x_{-4})))) \quad (2.29)
\]
Figure 2.2: The block diagram of the proposed CVNS-based sigmoid function evaluation structure

\[ r_3 = \overline{r_1} \wedge \overline{r_2} \wedge (x_0 \vee x_1 \vee x_2) \]  

(2.30)
Table 2.1: Input range decoder truth table

<table>
<thead>
<tr>
<th>Input range</th>
<th>r1</th>
<th>r2</th>
<th>r3</th>
<th>r4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x \geq 5$</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$2.375 \leq x &lt; 5$</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$1 \leq x &lt; 2.375$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$0 \leq x &lt; 1$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

$$r_4 = r_1 \land r_2 \land r_3$$ (2.31)

In the implemented hardware, 8\(\mu\)A is indicator of 1. Therefore, all the constant values are multiplied by 8\(\mu\)A. Thus, (2.26) can be written in the following form:

$$\begin{align*}
\left(\left(\gamma\right)\right) & = \begin{cases}
15 \mu A & r_1 = 0 \\
15 \mu A & \left(\left(\gamma\right)\right)_{-51} = 15 \mu A \\
\left(\left(\gamma\right)\right)_{-52} = (\left(\left(\gamma\right)\right)_{0} + 8 \mu A) \mod 16 \mu A & r_2 = 0 \\
\left(\left(\gamma\right)\right)_{-53} = (\left(\left(\gamma\right)\right)_{2} + 10 \mu A) & r_3 = 0 \\
\left(\left(\gamma\right)\right)_{-54} = (\left(\left(\gamma\right)\right)_{-2} + 8 \mu A) & r_4 = 0 \\
\left(\left(\gamma\right)\right)_{-55} = (\left(\left(\gamma\right)\right)_{1} - 2 \mu A) & x_3 = 1 \\
\left(\left(\gamma\right)\right)_{-56} = (\left(\left(\gamma\right)\right)_{-3} - 2 \mu A) \\
\left(\left(\gamma\right)\right)_{-57} = (\left(\left(\gamma\right)\right)_{-1} - 2 \mu A) \\
\left(\left(\gamma\right)\right)_{-58} = (\left(\left(\gamma\right)\right)_{-5} - 2 \mu A)
\end{cases}
\end{align*}$$ (2.32)

2.4.2 Current Generator

To evaluate the sigmoid function using the relations developed in (2.32), various constant current values are required.

The main building block of the current generator unit is shown in Fig. 2.3. Transistors M_1 to M_3 generate a 5 \(\mu\)A reference current. By proper sizing of the transistors an 11 \(\mu\)A current is copied to the transistors M_4 to M_{13}. The 11 \(\mu\)A current copied to the transistors M_4 and M_5 is used for
Figure 2.3: VLSI implementation of the current reference circuit used in the current generator block

generation of the current signal required for the output assignment unit. The current is copied to
the transistors M\textsubscript{6} to M\textsubscript{13} provided that the corresponding r\textsubscript{1} signal is active. Therefore, depending
on the input range, the respective 11 \( \mu \text{A} \) current will be generated. Hence, the current copied to
the transistors M\textsubscript{6} to M\textsubscript{13} is exploited to generate the constant current signals required for four
sub-units of the sigmoid approximation unit. Generation of different current values is conducted
using basic current mirrors. The current values generated by the current generator block for different
input ranges are summarized in Table 2.2. It should be noted that the 7.5 \( \mu \text{A} \) generated in all input
ranges is used by the output assignment unit.

2.4.3 Sigmoid Approximation

In this section, the input processing sub-units for different input sub-ranges are explained.

Output approximation for input range \( r_1 \)

In the input range \( r_1 \), both output CVNS digits have a constant value of 15 \( \mu \text{A} \). The 15 \( \mu \text{A} \) is
generated by the current generator block and is passed to the output provided that the \( r_1 \) signal is active.

Output Approximation for Input Range \( r_2 \)

The input processing circuit for the input range \( r_2 \) is implemented based on (2.32).
Table 2.2: Current values generated by the current generator block

<table>
<thead>
<tr>
<th>Input range</th>
<th>Generated Current</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x \geq 5 )</td>
<td>7.5 ( \mu A ), 15 ( \mu A )</td>
</tr>
<tr>
<td>2.375 ( \leq x &lt; 5 )</td>
<td>1 ( \mu A ), 7.5 ( \mu A ), 8 ( \mu A ), 14 ( \mu A ), 16 ( \mu A )</td>
</tr>
<tr>
<td>1 ( \leq x &lt; 2.375 )</td>
<td>7.5 ( \mu A ), 10 ( \mu A )</td>
</tr>
<tr>
<td>0 ( \leq x &lt; 1 )</td>
<td>7.5 ( \mu A ), 8 ( \mu A )</td>
</tr>
</tbody>
</table>

To evaluate \( (x)^2 \) cmp 6 \( \mu A \), 1 \( \mu A \) current generated by the current generator block is passed to the output of this sub-unit provided that \( (x)^2 \) is greater than 6 \( \mu A \). Since if \( x_0 \) or \( x_2 \) are one, the \( (x)^2 \) will be greater than 6 \( \mu A \), \( (x)^2 \geq 6 \mu A \) can be expressed by the \( x_2 \lor x_0 \) logic expression. This can be implemented using an OR gate. Therefore, \( (x)^2 \) cmp 6 \( \mu A \) is implemented by a transistor receiving the 1 \( \mu A \) from the current generator block and controlled by the \( x_2 \lor x_0 \) signal.

The \( (y)_{-12} \) output has two terms which their summation produces the output. Therefore, the output of \( (x)^2 \) cmp 6 \( \mu A \) evaluation is wired with the 14 \( \mu A \) current generated by the current generator block.

To generate the \( (y)_{-52} \) CVNS output digit, the 8 \( \mu A \) current generated by the current generator block is wired with the \( (x)_0 \) input and the mod 16 \( \mu A \) operation is applied. Using the mod2 operation basic properties, the mod2 block is designed as follows:

\[
((x)_0 + 8\mu A) \mod 16 \mu A = \begin{cases} 
(x)_0 + 8 \mu A & (x)_0 < 8 \mu A \\
(x)_0 - 8 \mu A & (x)_0 \geq 8 \mu A
\end{cases}
\]

The VLSI implementation of the mod 16\( \mu A \) operation is shown in Fig. 2.4. The current subtractor block is enabled by the signal mod2.en applied to the transistor M1. Considering that (\( x)_0 \geq 8 \mu A \)) is equivalent to \( x_0 = 1 \), the mod2.en signal is generated using the following logic expression:

\[
\text{mod2.en} = \overline{x}_2 \land x_0
\]

When the mod 2_en signal becomes active, the 16 \( \mu A \) current generated by the current generator block is subtracted from the current through the transistor M3 which is equal to the input current to the mod 16 \( \mu A \) block. Then, the subtracted current is delivered to the transistor M4, being copied to the transistor M5.
2. CVNS-BASED SIGMOID FUNCTION EVALUATION FOR PRECISE NEUROCHIPS

Output Approximation for Input Range $r_3$

The $\langle y \rangle_{-13}$ output CVNS digit in the input range $r_3$ is generated by wiring the 10 $\mu$A current generated by the current generator block with the input $\langle x \rangle_2$.

The $\langle y \rangle_{-53}$ is equal to the input $\langle x \rangle_{-2}$ provided that the input is in the input range $r_3$. This is implemented using a transistor as a switch controlled by the $r_3$ output of the input range decoder which turns on when the input is in the input range $r_3$.

Output Approximation for Input Range $r_4$

The $\langle y \rangle_{-13}$ output CVNS digit in the input range $r_4$ is generated by wiring the 8 $\mu$A current generated by the current generator block with the input $\langle x \rangle_1$. The $\langle y \rangle_{-53}$ is equal to the input $\langle x \rangle_{-3}$ provided that it is within the input range $r_3$. This is implemented using a transistor as a switch, and controlled by the $r_4$ output of the input range decoder.

2.4.4 Output Assignment Block

The output assignment block assigns the output based on the input sign. If the input is positive, the output of the sigmoid approximation unit is directly passed to the output; otherwise, it is subtracted from 15 $\mu$A. The output assignment unit circuit is shown in Fig. 2.5. The transistor $M_6$ acts as a
2. CVNS-BASED SIGMOID FUNCTION EVALUATION FOR PRECISE NEUROCHIPS

2.5 Post-layout Simulation and Comparisons

The proposed structure exploits the CVNS features which make the high precision analog circuits implementation feasible. Moreover, using CVNS, the proposed structure requires lower number of output digits which decreased the area consumption. Furthermore, current-mode circuits are used to realize the CVNS sigmoid function. Considering that the main arithmetic operation in the proposed CVNS-based sigmoid activation function is addition, low area and high speed addition offered by

![Diagram of VLSI implementation of the output assignment unit](image)

switch which turns on provided that the input sign is positive, passing the input to the output.

It should be noted that the input sign may be checked through $x_3$ input bit. If the input sign is negative, the $7.5 \, \mu A$ current generated by the current generator block is copied to the transistor $M_5$. Considering that the $\frac{W}{L}$ of the transistor $M_3$ is two times of the transistor $M_5$, $15 \, \mu A$ will be copied to the transistor $M_3$. The transistors $M_1$ to $M_3$ act as a current subtractor, subtracting the input from $15 \, \mu A$. The subtracted current is copied to the transistor $M_4$, generating the output.
Figure 2.6: The simulation results of the proposed CVNS-based sigmoid activation function current-mode circuits resulted in lower area consumption and higher speed.

The performance of the VLSI implementation of the proposed CVNS sigmoid function evaluation structure is verified by conducting post-layout simulations for various inputs within the input range of (-8, 8).

Fig. 2.6 shows the post-layout simulation results for four input values of 0.875, 1.875, 3.625 and 7.9375 where each are in the input ranges \( r_4, r_3, r_2 \) and \( r_1 \) respectively. These inputs are applied with intervals of 10 ns to the sigmoid function evaluation circuit. The CVNS values corresponding to these four inputs are shown in Table 2.3. Using (2.32), the expected output for each input can be calculated. The relation between \( (y)_{-1} \) and \( (y)_{-5} \) with input CVNS digits and the expected value of \( (y)_{-1} \) and \( (y)_{-5} \) for each input is summarized in Table 2.3. The post-layout simulation results shown in Fig. 2.6 are in agreement with the mathematical derivations presented in Table 2.3.

To compare the efficiency of the proposed structure with state of the art, the structures developed in [11–13], [15, 16] and [19] are coded using the Verilog hardware description language and implemented in TSMC 0.18\( \mu \)m technology. To the best of our knowledge, the multiplierless PWL-based methods developed in [12] and [13] are the best previously developed solutions to the ASIC implementation of sigmoid activation function for neurochips with on-chip learning.

It should be noted that since the VLSI implementation of the proposed structure is conducted using analog circuits, the proposed structure is laid out manually while the layout of the digital structures is generated automatically. The post-layout area, delay and power consumption results
Table 2.3: Input values and their corresponding input and output CVNS digits

<table>
<thead>
<tr>
<th>x</th>
<th>( ((x))_1 )</th>
<th>( ((x))_2 )</th>
<th>( ((x))_0 )</th>
<th>( ((x))_{-1} )</th>
<th>( ((x))_{-2} )</th>
<th>( ((x))_{-3} )</th>
<th>( ((y))_{-1} )</th>
<th>( ((y))_{-5} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.875</td>
<td>0</td>
<td>1 ( \mu A )</td>
<td>3 ( \mu A )</td>
<td>7 ( \mu A )</td>
<td>14 ( \mu A )</td>
<td>12 ( \mu A )</td>
<td>8 ( \mu A )</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>( ((x))_{1} + 8 \mu A )</td>
<td>( (x)_{-3} = 8 \mu A )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.875</td>
<td>1 ( \mu A )</td>
<td>3 ( \mu A )</td>
<td>7 ( \mu A )</td>
<td>15 ( \mu A )</td>
<td>14 ( \mu A )</td>
<td>12 ( \mu A )</td>
<td>8 ( \mu A )</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>( ((x))_{2} + 10 \mu A )</td>
<td>( (x)_{-2} = 12 \mu A )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.625</td>
<td>3 ( \mu A )</td>
<td>7 ( \mu A )</td>
<td>14 ( \mu A )</td>
<td>13 ( \mu A )</td>
<td>10 ( \mu A )</td>
<td>4 ( \mu A )</td>
<td>8 ( \mu A )</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>( ((x))_{2} ) cmp 6 ( \mu A )</td>
<td>( (x)_{0} + 8 \mu A )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>+14 ( \mu A ) = 15 ( \mu A )</td>
<td>mod 16 ( \mu A ) = 5 ( \mu A )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.9375</td>
<td>7 ( \mu A )</td>
<td>15 ( \mu A )</td>
<td>15 ( \mu A )</td>
<td>15 ( \mu A )</td>
<td>14 ( \mu A )</td>
<td>9 ( \mu A )</td>
<td>8 ( \mu A )</td>
<td>15 ( \mu A )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>15 ( \mu A )</td>
</tr>
</tbody>
</table>

Table 2.4: Comparison of different structures

<table>
<thead>
<tr>
<th>Structure</th>
<th>Input Range</th>
<th>( N_{in} )</th>
<th>( N_{out} )</th>
<th>Maximum Error</th>
<th>Area (( \mu m^2 ))</th>
<th>Delay (ns)</th>
<th>Power (( \mu W ))</th>
<th>ADP (( \mu m^2 \times ns \times \mu W ))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bharkhada</td>
<td>(-8,8)</td>
<td>17</td>
<td>16</td>
<td>0.0001</td>
<td>352 336.86</td>
<td>64.87</td>
<td>4 109.30</td>
<td>93 922 539 300.22</td>
</tr>
<tr>
<td>[16]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Zhang</td>
<td>(-4,4)</td>
<td>14</td>
<td>14</td>
<td>0.0216</td>
<td>5 416.22</td>
<td>6.66</td>
<td>866.34</td>
<td>31 250 638.31</td>
</tr>
<tr>
<td>[15]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tomniska</td>
<td>(-8,8)</td>
<td>7</td>
<td>7</td>
<td>0.0708</td>
<td>1 404.95</td>
<td>1.88</td>
<td>368.82</td>
<td>974 166.47</td>
</tr>
<tr>
<td>[19]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vassiliadis</td>
<td>(-4,4)</td>
<td>14</td>
<td>10</td>
<td>0.0240</td>
<td>1 804.32</td>
<td>2.07</td>
<td>330.30</td>
<td>1 233 651.48</td>
</tr>
<tr>
<td>[11]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Alippi</td>
<td>(-8,8)</td>
<td>12</td>
<td>8</td>
<td>0.0189</td>
<td>1 745.96</td>
<td>2.58</td>
<td>518.94</td>
<td>2 337 605.09</td>
</tr>
<tr>
<td>[12]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PLAN</td>
<td>(-8,8)</td>
<td>9</td>
<td>8</td>
<td>0.0189</td>
<td>1 347.23</td>
<td>3.71</td>
<td>268.20</td>
<td>1 340 523.49</td>
</tr>
<tr>
<td>[13]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Proposed</td>
<td>(-8,8)</td>
<td>16</td>
<td>21</td>
<td>0.0189</td>
<td>965.37</td>
<td>2.53</td>
<td>192.63</td>
<td>470 476.83</td>
</tr>
</tbody>
</table>

are summarized in Table 2.4.

Comparisons are made with state of the art where the goal is to satisfy the requirements of neural networks with on-chip learning as required by [23–26]. These include the input range, output

\(^{3}N_{out}\) for the proposed structure is indicator of the number of output CVNS digits as given by (2.25)
resolution and maximum approximation error. Since all structures have been re-implemented, area, delay, and power reports are also provided for a complete comparison.

The $N_{in}$ and $N_{out}$ in Table 2.4 are indicator of the number of input and output digits for each of the sigmoid function evaluation. For the proposed work, there are 8 binary and 8 CVNS input digits. Following the mathematical setup in section 2.2, the proposed structure requires only two output CVNS digits.

As may be noted from Table 2.4, different methods use different number of input and output digits. For example, the number of input and output bits in [19] is lower than the minimum number of bits required for on-chip neural networks [23]. All of the other structures satisfy the requirement of the minimum number of input and output digits.

The structure developed in [16] meets the requirements of on-chip neural networks in terms of input range, output resolution and maximum error. This work is based on a third order nonlinear approximation and is more appropriate for the FPGA platform. This structure has the lowest maximum error at the cost of extremely high area, delay and power consumption. The reported delays are the critical path delay of the corresponding structures.

The structure reported in [15] performs the sigmoid function approximation based on a second order nonlinear approximation. This method does not meet the input range requirement. Moreover, its area and delay is the second highest among the reported work.

The work done by Tommiska et. al [19] exploits the bit-level mapping method and has the lowest delay. However, it fails to satisfy the resolution requirement. In addition, it has the highest approximation error among all structures.

The structure reported in [11] has 2.07 ns delay. However, its input range is limited to only (-4,4) and its approximation error is slightly high.

Both works reported in [12] and [13] are based on multiplierless PWL method and provide the same maximum approximation error.

The proposed CVNS structure satisfies all the requirements of the application including maximum error, number of digits and input range. The maximum number of arithmetic operations for the proposed structure occurs for the inputs in the input range $r_2$, which includes addition, mod2 operation and the output assignment. The proposed structure has the lowest area and power consumption. The delay of the developed structure is 2.53 ns.

The power measurement is performed through post-layout simulation for all structures reported in Table 2.4. The input rate for the proposed structure and the structures developed in [11–13]
and [19] is set to a unified optimum speed that all of these structures can operate. The input period for these structures is set to 3.71 $ns$. Since the structures developed in [15] and [16] have higher delay, their input rate is set to 6.66 $ns$ and 64.87 $ns$ respectively. A uniform random bit stream including 10,000 inputs is applied to all these circuits to measure the power consumption.

The proposed addition algorithm provides inputs to the proposed structure both in CVNS and binary. Therefore, no binary to CVNS conversion is required provided that the whole network is implemented based on the CVNS arithmetic. The output of the proposed activation function is the input to the multiplier of the next layer in the network. Considering that the input to the CVNS multiplier is in the CVNS format [31], no conversion is required. Therefore, the area, delay and power consumption of the proposed structure are reported in Table 2.4 without any conversion units. The input to the CVNS conversion can be carried out using a current steering digital to analog converter while the output to binary conversion can be conducted using the current comparator developed in [36]. As an example, the area consumption of the 4-3-2 network developed in [32] is 385320 $\mu m^2$ while the area consumption of the input to CVNS and output to binary conversion circuits is 1245.22 $\mu m^2$ and 140.8 $\mu m^2$ respectively. Therefore, the area overhead of the input and output conversion compared to the area consumption of the whole network is negligible.

Considering that the area, delay and power consumption are all important in ASIC implementation, Area×Delay×Power (ADP) is defined as a performance metric. The ADP values of different structures are summarized in Table 2.4, which show that the proposed CVNS sigmoid function evaluation structure has a three times lower ADP compared to the best previously developed structure.

Moreover, comparisons are performed between the proposed structures and two analog neurons developed in [27] and [28]. Custom layout and circuit design are carried out in TSMC 0.18 $\mu m$ technology. Both structures implement the activation function with six transistors. The neuron in [27] is a resistive type distributed neuron which implements the activation function through transistor nonlinear properties. Several neurons are required for each layer of the network depending on the network size. For each unit, the area, delay and power consumption are equal to 62.32 $\mu m^2$, 0.162 $ns$ and 152.1 $\mu W$ respectively. The implementation of [28] shows an area consumption, delay and power consumption of 122.52 $\mu m^2$, 0.39 $ns$ and 205.2 $nW$ respectively. Although these neurons have area efficiency compared to the proposed CVNS sigmoid, process variation affects their performance. As an example, the Monte Carlo simulation of the neuron developed in [27] for its maximum output is conducted. The simulation result is shown in Fig. 2.7. The process variation causes error in the output which in turn makes the realization of precise sigmoid activation function using analog neurons impractical.
Since the VLSI implementation of the proposed structure is carried out using mixed-signal circuits, process variation affects the proposed structure as well. The accuracy of the proposed structure depends on the accuracy of the constant current values generated by the current generator block. Since all the constant current values generated by the current generator block are generated from the same current reference, the highest output current level of the proposed structure has the highest variation. The highest output current level of the proposed structure is equal to 15 $\mu$A. Therefore, a Monte Carlo simulation of the 15 $\mu$A output current with 10,000 runs is conducted. The Monte Carlo simulation result is shown in Fig. 2.8. Since the difference between the two consecutive output levels of the proposed CVNS sigmoid function evaluation is equal to 1 $\mu$A, all of the output current levels between 14.5 $\mu$A and 15.5 $\mu$A represent the same output. Considering that all of the output current levels between 14.5 $\mu$A and 15.5 $\mu$A represent the same output, Monte Carlo simulation results show a yield of 47.44 % for the proposed CVNS sigmoid activation function structure. Therefore, the proposed structure makes the use of analog circuits for implementation of precise sigmoid activation function required for neurochips with on-chip learning possible.

2.6 Conclusion

A new CVNS-based sigmoid activation function evaluation scheme for neurochips with on-chip learning is proposed in this paper. The proposed function evaluation scheme exploits the PWL approximation method and is based on a mathematical derivation using the CVNS features. Moreover,
based on the maximum approximation error, the number of input and output CVNS digits required for VLSI implementation of the proposed sigmoid function evaluation method is determined.

To realize the proposed CVNS-based sigmoid function evaluation scheme, a new CVNS-based structure is proposed. The proposed structure exploits the mixed-signal current-mode circuits, which efficiently implement the addition arithmetic operation. In addition, the proposed CVNS-based sigmoid function evaluation requires a lower number of output digits when compared to the state of the art. The implementation results in TSMC 0.18\(\mu m\) technology show that the proposed structure compares favorably to the state of the art.

Appendix

Proof of the Proposed CVNS Function Evaluation Scheme

The proposed CVNS function evaluation is based on the PLAN method developed in [13]. The approximation developed in [13] is as follows:

\[
y = \begin{cases} 
1 & x \geq 5 \\
0.03125 \times x + 0.84375 & 2.375 \leq x < 5 \\
0.125 \times x + 0.625 & 1 \leq x < 2.375 \\
0.25 \times x + 0.5 & 0 \leq x < 1 \\
y_x = 1 - y_{|x|} & x < 0 
\end{cases}
\]  

(2.35)
where \(x\) and \(y\) are the real numbers which represent the input and output of the sigmoid activation function.

Using (2.2) and (2.4) along with \(N_i = 3\) and \(N_f = 4\) obtained in section 2.2, the input \(x\) in terms of its corresponding CVNS digits can be written in the following form:

\[
x = 2^3 \left( \langle x \rangle_3 + \frac{\langle x \rangle_{-1}}{2^4} \right)
\] (2.36)

The output CVNS digits can be generated based on the following equation [31]:

\[
\langle y \rangle_j = (y \times 2^{-j}) \mod 2 \tag{2.37}
\]

Before going through the mathematical derivation of the CVNS-based sigmoid function evaluation, three basic properties that are going to be used several times are reviewed. These properties will be referred as basic properties one, two and three in the rest of this appendix.

1) According to (2.4), the minimum value representable by each CVNS digit is equal to \(2^{-(\varphi - 1)}\). Therefore, assuming a reliable environment resolution of four, in CVNS-based function evaluation, the values less than \(2^{-3}\) are not considered. Since an adequate number of CVNS digits are used to represent the output, this has no effect on the accuracy of the proposed function evaluation method. This can be shown in the following form:

\[
i \leq -4 \Rightarrow 2^i x_j = 0
\] (2.38)

where \(i\) and \(j\) are integer values and \(x_j\) can assume values 0 or 1.

2) A basic property of mod2 operation is that if the outcome of this continuous modular reduction operation is zero, it should be an even value, greater than one. This can be shown in the following form:

\[
i \geq 0 \Rightarrow 2^i x_j \mod 2 = 0
\] (2.39)

3) Another basic property of mod2 operation is exploited too. If the input to mod2 function is less than 2, the mod2 function passes the input to the output without change. This can be shown in the following form:

\[
x < 2 \Rightarrow x \mod 2 = x
\] (2.40)

According to (2.35), there are four different input regions. In addition, the output for negative input values can be determined using the output for absolute value of input. Therefore, using basic properties one, two and three along with the CVNS arithmetic features, mathematical derivation of determining the CVNS output digits \(\langle y \rangle_{-1}\) and \(\langle y \rangle_{-5}\) in four different input regions as well as negative input values is conducted in the following sections. It should be noted that \(\langle y \rangle_{ij}\) is used
2. CVNS-BASED SIGMOID FUNCTION EVALUATION FOR PRECISE NEUROCHIPS

as an indicator of the output CVNS digits in each region, where \( i \) and \( j \) represent the CVNS digit and region index respectively.

**CVNS Function Evaluation in the Input Range \( x \geq 5 \)**

In this region, named as \( r_1 \), \( y \) has a constant value of 1. Therefore, the output CVNS digits \( (\ldots y_{-11}) \) and \( (\ldots y_{-51}) \) are equal to each other and according to (2.4), are calculated in the following form:

\[
(\ldots y_{-11}) = \sum_{k=-4}^{-1} 2^{-k} = (\ldots y_{-51}) = \sum_{k=-8}^{-5} 2^{-k} = 1.875
\]  
(2.41)

**CVNS Function Evaluation in the Input Range \( 2.375 \leq x < 5 \)**

In this region, named as \( r_2 \), using (2.35), (2.36) and (2.37), the output CVNS digit \( (\ldots y_{-12}) \) can be written in the following form:

\[
(\ldots y_{-12}) = \frac{(\ldots x_3)}{2} + \frac{(\ldots x_{-1})}{32} + 1.6875
\]  
(2.42)

Using basic property one, (2.42) can be simplified as follows:

\[
(\ldots y_{-12}) = \frac{(\ldots x_3)}{2} + 1.6875
\]  
(2.43)

According to (2.4) and (2.43), \( (\ldots y_{-12}) \) output CVNS digit may be 1.625 or 1.875. To investigate this, the input region \( r_2 \) is divided to three sub-ranges. The first one is \( 2.375 \leq x < 3 \). In this sub-range, \( (\ldots x_3) = 0.25 \). Therefore, using basic property one and (2.42), we have \( (\ldots y_{-12}) = 1.75 \).

In the sub-range \( 3 \leq x < 4 \), we have \( (\ldots x_3) = 0.375 \). Therefore, using basic property one and (2.42), we have \( (\ldots y_{-12}) = 1.875 \). Following the same procedure and considering that in the the sub-range \( 4 \leq x < 5 \), \( (\ldots x_3) = 0.5 \), we have \( (\ldots y_{-12}) = 1.875 \). Therefore, in this region, the \( (\ldots y_{-12}) \) can be written in the following form:

\[
(\ldots y_{-12}) = \frac{(\ldots x_3 \text{ cmp } 0.375)}{8} + 1.75
\]  
(2.44)

where the cmp function is defined as follows:

\[
a \text{ cmp } b = \begin{cases} 
1 & a \geq b \\
0 & a < b
\end{cases}
\]  
(2.45)

According to (2.4) and (2.45), it can be easily proven that the condition \( (\ldots x_3 \text{ cmp } 0.375) \) is exactly the same as \( (\ldots x_2 \text{ cmp } 0.75) \). Therefore, (2.44) is simplified as follows:

\[
(\ldots y_{-12}) = \frac{(\ldots x_2 \text{ cmp } 0.75)}{8} + 1.75
\]  
(2.46)
As will be proved in the following section, \( (x)^2 \) is used to determine the \( (y)^{-13} \) output CVNS digit in the region \( 1 \leq x < 2.375 \) as well. Using the same input signal to generate the \( (y)^{-1} \) output CVNS digit in both regions may result in lower area consumption of the VLSI implementation of the proposed CVNS-based sigmoid function evaluation method.

Using (2.35), (2.36) and (2.37), the output CVNS digit \( (y)^{-52} \) in this region can be written in the following form:

\[
(y)^{-52} = \left( 8 (x)_3 + \frac{(x)^{-1}}{2} + 27 \right) \mod 2 = \left( 8x_3 + 4x_2 + 2x_1 + x_0 + \frac{x_{-1}}{2} + \frac{x_{-2}}{4} + \frac{x_{-3}}{8} + \frac{x_{-4}}{16} + 1 \right)
\]

(2.47)

Using basic properties one and two, (2.47) can be written in the following form:

\[
(y)^{-52} = (x_0 + \frac{x_{-1}}{2} + \frac{x_{-2}}{4} + \frac{x_{-3}}{8} + 1) \mod 2
\]

(2.48)

According to (2.4), (2.48) can be written in the following form:

\[
(y)^{-52} = ((x)_0 + 1) \mod 2
\]

(2.49)

**CVNS Function Evaluation in the Input Range \( 1 \leq x < 2.375 \)**

In this region, named as \( r_3 \), using (2.35), (2.36) and (2.37), the output CVNS digit \( (y)^{-13} \) can be written in the following form:

\[
(y)^{-13} = \left( 2 (x)_3 + \frac{(x)^{-1}}{8} + 1.25 \right) \mod 2
\]

(2.50)

Using basic property one, the \( \frac{(x)^{-1}}{8} \) in (2.50) is replaced with \( \lfloor \frac{(x)^{-1}}{8} \rfloor \), where \( \lfloor \rfloor \) is the floor function. Therefore, (2.50) can be written in the following form:

\[
(y)^{-13} = \left( 2 (x)_3 + \frac{\lfloor (x)^{-1} \rfloor}{8} + 1.25 \right) \mod 2
\]

(2.51)

Moreover, considering that \( \lfloor (x)^{-1} \rfloor = x_{-1} \) and using basic property two, (2.51) can be written in the following form:

\[
(y)^{-13} = \left( 2x_3 + x_2 + \frac{x_1}{2} + \frac{x_0}{4} + \frac{x_{-1}}{8} + 1.25 \right) \mod 2 = ((x)_2 + 1.25) \mod 2
\]

(2.52)

It can be easily proven that \( (x)_2 < 0.5 \) in this region. Therefore, using basic property three, (2.51) is simplified as follows:

\[
(y)^{-13} = (x)_2 + 1.25
\]

(2.53)
2. CVNS-BASED SIGMOID FUNCTION EVALUATION FOR PRECISE NEUROCHIPS

Using (2.35), (2.36) and (2.37), the $\left\langle y \right\rangle_{-53}$ output CVNS digit in this region can be written in the following form:

$$\left(\left\langle y \right\rangle_{-53} = 32\left(\left\langle x \right\rangle_{3} + 2 \left\langle x \right\rangle_{-1} + 20\right) \mod 2 \tag{2.54}$$

Using (2.4) and the basic properties two and three, (2.54) can be written in the following form:

$$\left(\left\langle y \right\rangle_{-53} = (2 \left\langle x \right\rangle_{-1}) \mod 2 = \left(2x_{-1} + x_{-2} + \frac{x_{-3}}{2} + \frac{x_{-4}}{4}\right) \mod 2 = \left\langle x \right\rangle_{-2} \tag{2.55}$$

CVNS Function Evaluation in the Input Range $0 \leq x < 1$

In this region, named as $r_{4}$, using (2.35), (2.36) and (2.37), the $\left\langle y \right\rangle_{-14}$ output CVNS digits can be written in the following form:

$$\left(\left\langle y \right\rangle_{-14} = 4 \left(\left\langle x \right\rangle_{3} + \frac{\left\langle x \right\rangle_{-1}}{4} + 1\right) \mod 2 \tag{2.56}$$

Since all input binary digits $x_{0}$ to $x_{3}$ are equal to 0 in the input range $0 \leq x < 1$, we have $\left\langle x \right\rangle_{3}=0$. Therefore, using (2.4) and considering that each CVNS digit spans over only 4 bits, (2.56) can be written in the following form:

$$\left(\left\langle y \right\rangle_{-14} = \left(\left\langle x \right\rangle_{-1} + 1\right) \mod 2 \tag{2.57}$$

According to (2.4), the $\frac{\left\langle x \right\rangle_{-1}}{4}$ can be written in the following form:

$$\frac{\left\langle x \right\rangle_{-1}}{4} = \left(\frac{x_{-1}}{4} + \frac{x_{-2}}{8} + \frac{x_{-3}}{16} + \frac{x_{-4}}{32}\right) \mod 2 \tag{2.58}$$

Considering the basic property one, it can be simplified in the following form:

$$\frac{\left\langle x \right\rangle_{-1}}{4} = \left(\frac{x_{-1}}{4} + \frac{x_{-2}}{8}\right) \mod 2 \tag{2.59}$$

Considering that since originally the range of input value $x$ is between zero and one, and using (2.4), it is clear that $x_{0}$ and $x_{1}$ are zero as well. Therefore, $\left\langle x \right\rangle_{1} = \left(\frac{x_{-1}}{4} + \frac{x_{-2}}{8}\right) \mod 2$ and $\left\langle y \right\rangle_{-14}$ output CVNS digit can be written in the following form:

$$\left(\left\langle y \right\rangle_{-14} = \left(\left\langle x \right\rangle_{1} + 1\right) \mod 2 \tag{2.60}$$

Since processing the signal $\frac{\left\langle x \right\rangle_{-1}}{4}$ requires a division, replacing it with $\left\langle x \right\rangle_{1}$ simplifies the signal processing, which in turn may result in more efficient VLSI implementation.

In addition, since $x_{0}$ and $x_{1}$ are zero, $\left\langle x \right\rangle_{1}$ is less than one. Therefore, using basic property three, (2.60) is modified as following:

$$\left(\left\langle y \right\rangle_{-14} = \left\langle x \right\rangle_{1} + 1 \tag{2.61}$$
According to (2.35), (2.36) and (2.37), the other output CVNS digit in this region, \( (y)_{-54} \), can be written in the following form:

\[
(\langle y \rangle)_{-54} = (64 (\langle x \rangle)_3 + 4 (\langle x \rangle)_{-1} + 16) \mod 2
\]  

(2.62)

Using basic properties two, three and (2.4), (2.62) can be written in the following form:

\[
(\langle y \rangle)_{-54} = (4 (\langle x \rangle)_{-1}) \mod 2 = \left(4 x_{-1} + 2x_{-2} + x_{-3} + \frac{x_{-4}}{2}\right) \mod 2 = (\langle x \rangle)_{-3}
\]  

(2.63)

**CVNS Function Evaluation for Negative Input Values**

According to (2.35), to evaluate the outputs for negative input values, the output for absolute value of input is calculated and subtracted from 1 to generate the output. Considering that the number of output bits is equal to 8, this can be shown in the following form:

\[
y_x = \sum_{k=-8}^{-1} (1 - y_k) 2^k
\]  

(2.64)

where \( y_x \) and \( y_k \) are the output for negative inputs and the binary digits representing the output corresponding the absolute value of input.

Using (2.37), the \( (\langle y \rangle)_{-1} \) output CVNS digit can be calculated as follows:

\[
(\langle y \rangle)_{-1x} = \sum_{k=-8}^{-1} ((1 - y_k) 2^{k+1}) \mod 2
\]  

(2.65)

Using basic property one, (2.65) can be written in the following form:

\[
(\langle y \rangle)_{-1x} = \sum_{k=-4}^{-1} (1 - y_k)2^{k+1}
\]  

(2.66)

which can be simplified in the following form:

\[
(\langle y \rangle)_{-1x} = 1.875 - \sum_{k=-4}^{-1} y_k 2^{k+1}
\]  

(2.67)

According to (2.4), (2.67) can be simplified as follows:

\[
(\langle y \rangle)_{-1x} = 1.875 - (\langle y \rangle)_{-1|x|}
\]  

(2.68)

Repeating the same steps for \( (\langle y \rangle)_{-5x} \), it can be obtained using the following equation:

\[
(\langle y \rangle)_{-5x} = 1.875 - (\langle y \rangle)_{-5|x|}
\]  

(2.69)

Therefore, the output CVNS digits for the negative input values can be calculated by evaluating the absolute value of input and subtracting it from 1.875.
2.7 References


REFERENCES


Chapter 3

CVNS Synapse Multiplier for Robust Neurochips with On-Chip Learning

Software implementations of neural networks may represent the inputs and weights of the network with a high precision while as a result of the limited word length accessible in hardware, the inputs and weights in hardware implementations of neural networks are represented with a finite precision [1–5].

Representing the inputs and weights with a finite precision in hardware implementations of neural networks degrades the network output response. This makes the sensitivity of neural network hardware structures to the input and weight errors an important issue [6–13]. The studies show that the Noise-to-Signal-Ratio (NSR) can be used to analyze the network output sensitivity to input and weight errors. A network with lower NSR is more robust against the input and weight errors.

The continuous Valued Number System (CVNS) is a mixed-signal number system which is previously exploited to develop neural network structures with lower sensitivity to input and weight errors [14,15].

The main arithmetic operations in a neural network include multiplication, addition and a non-linear activation function. Although a CVNS multiplication algorithm is developed in [16], its resolution is limited by the precision of the analog circuits used for its implementation. The precision of the analog circuits is referred as the environment resolution in the rest of this paper.

In [17], effect of low-resolution environment on the NSR of the CVNS Adaline developed in [15]
using the previously developed CVNS multiplication algorithm is studied. The study shows that
the NSR of the CVNS Adaline for applications requiring a multiplication resolution more than the
environment resolution increases.

According to the simulations performed in [18] and [19] for on-chip neurochips, 16-bit synap-
tic weight storage resolution is required while an 8-bit resolution for representation of the activation
function output is needed. Since in a neural network the output of the activation function is
multiplied by the weights, synapse multiplier with a resolution of 16×8 bits is required for VLSI
implementation of neurochips with on-chip learning.

The results of [20–22] reveals that the environment resolution for implementation of CVNS in
TSMC CMOS 0.18µm and 90nm is 4 bits.

The CVNS multiplication algorithm developed in [16] is based on the assumption that the en-
vIRONMENT resolution is as high as the resolution required for multiplication. Since the resolution
requirement for multipliers of neurochips with on-chip learning is higher than the environment res-
OLUTION, using the previously developed multiplication algorithm for on-chip neurochips becomes
impractical.

In this paper, a new CVNS multiplication algorithm for a low-resolution environment is proposed.
The proposed algorithm provides accurate results in the low-resolution environment. Moreover, using
the proposed multiplication algorithm, the VLSI implementation of a CVNS synapse multiplier for
on-chip neurochips is realized.

The effect of the proposed multiplication algorithm on the NSR of the CVNS Adaline developed
in [15] is studied in this paper. The study shows that the proposed multiplication algorithm provides
lower NSR. This results in more error tolerant neural network structures.

The rest of this paper is organized as follows. A new CVNS multiplication algorithm for low-
resolution environment is introduced in section 3.1. Afterwards, the VLSI Implementation of the
CVNS synapse multiplier for neurochips with on-chip learning is explained in section 3.2. Post-
layout simulation is carried out in section 3.3. Comparison of the proposed CVNS multiplication
algorithm versus previously developed algorithm is conducted in section 3.4. Finally, conclusions
are drawn in section 3.5.
3. CVNS SYNAPSE MULTIPLIER FOR ROBUST NEUROCHIPS WITH ON-CHIP LEARNING

3.1 Proposed CVNS Multiplication Algorithm in Low Resolution Environment

In this section, a new CVNS multiplication algorithm is proposed. The proposed algorithm provides accurate results in a low-resolution environment. Therefore, it can provide the multiplication resolution required by on-chip neurochips.

The absolute value of two multiplication operands \( X \) and \( Y \) using fixed-point number system with a radix of \( B \) can be shown as follows:

\[
X = \sum_{i=-N_{fx}}^{N_{ix}-1} x_i \times B^i \quad (3.1)
\]

\[
Y = \sum_{i=-N_{fy}}^{N_{iy}-1} y_i \times B^i \quad (3.2)
\]

where \( N_{ix} \) and \( N_{fx} \) are the number of integer and fractional digits of the multiplication operand \( X \) respectively. \( N_{iy} \) and \( N_{fy} \) are the number of integer and fractional digits of the multiplication operand \( Y \) respectively. \( x_i \) and \( y_i \) are the digits.

The CVNS digit \( \langle z \rangle_m \) representing the multiplication result of two operands, \( X \) and \( Y \), can be written in the following form [16]:

\[
\langle z \rangle_m = \langle X \times Y \rangle_m = \left( X \times Y \times B^{-m} \right) \mod B \quad (3.3)
\]

Replacing \( X \) and \( Y \) with their fixed-point representation using (3.1) and (3.2), (3.3) can be written in the following form:

\[
\langle z \rangle_m = \left( \sum_{i=-N_{fy}}^{N_{iy}-1} y_i \times B^i \sum_{j=-N_{fx}}^{N_{ix}-1} x_j \times B^j \times B^{-m} \right) \mod B \quad (3.4)
\]

One of the basic properties of \( \mod B \) operation is that, if it is applied on values which are a multiple of \( B \) and greater than one, the outcome of this continuous modular reduction operation will be zero. This can be shown in the following form:

\[
k \geq 0 \Rightarrow (B^k \times x_j) \mod = 0 \quad (3.5)
\]

Using this basic property, (3.4) can be modified in the following form:

\[
\langle z \rangle_m = \left( \sum_{i=-N_{fy}}^{N_{iy}-1} y_i \sum_{j=-N_{fx}}^{m-i} x_j \times B^{-m-i} \right) \mod B \quad (3.6)
\]
The CVNS digit $\langle x \rangle_m$ representing the input $X$ in an environment with resolution of $\varphi$ is as follows [15]:

$$\langle x \rangle_m = \sum_{i=m-\varphi+1}^{m} x_i \times B^{i-m}$$  \hspace{1cm} (3.7)

Equation (3.7) can be exploited to represent the $\langle z \rangle_m$ in terms of the CVNS digits $\langle x \rangle_m$ which represent the multiplication operand $X$ in the CVNS format. In order to do this, (3.6) is modified in the following form:

$$\langle z \rangle_m = \left( \sum_{i=-N_fy}^{N_iy-1} y_i \left( \sum_{j=m-i-\varphi+1}^{m-i} x_j \times B^{j-1-(m-i)} + B^{-\varphi} \sum_{j=m-i-2\varphi+1}^{m-i-\varphi} x_j \times B^{j-1-(m-i+\varphi)} \right) + \ldots + B^{-(n_i+1)\varphi} \sum_{j=-N_fy}^{m-i-\varphi} x_j \times B^{j-1-(m-i+n_i\varphi)} \right) \mod B$$  \hspace{1cm} (3.8)

Since each CVNS digit $\langle x \rangle_m$ has $\varphi$ terms, the value of $n_i$ is found based on the following condition:

$$m - i - n_i\varphi + N_f x \leq \varphi - 1$$  \hspace{1cm} (3.9)

Therefore, $n_i$ can be calculated using the following equation:

$$n_i = \left\lceil \frac{m + (N_f x + 1) - (i + \varphi)}{\varphi} \right\rceil$$  \hspace{1cm} (3.10)

in which $\lceil \rceil$ is the ceiling function.

Using (3.7) and (3.8), $\langle z \rangle_m$ is written as follows:

$$\langle z \rangle_m = \left( \sum_{i=-N_fy}^{N_iy-1} y_i \left( \langle x \rangle_{m-i} + B^{-\varphi} \langle x \rangle_{m-i+\varphi} + \ldots + B^{-(n_i+1)\varphi} \langle x \rangle_{m-i+n_i\varphi} \right) \right) \mod B$$  \hspace{1cm} (3.11)

Repeating the same procedure from (3.3) to (3.11) for $\langle z \rangle_{m-\varphi}$, $\langle z \rangle_{m-\varphi}$ can be calculated in the following form:

$$\langle z \rangle_{m-\varphi} = C_{m-\varphi} \mod B = \left( \sum_{i=-N_fy}^{N_iy-1} y_i \left( \langle x \rangle_{m-i+\varphi} + B^{-\varphi} \langle x \rangle_{m-i+2\varphi} + \ldots + B^{-(n_i+1)\varphi} \langle x \rangle_{m-i+n_i\varphi} \right) \right) \mod B$$  \hspace{1cm} (3.12)

Using (3.8) and (3.12), we can write:

$$\langle z \rangle_m = \left( \sum_{i=-N_fy}^{N_iy-1} y_i \langle x \rangle_{m-i} + B^{-\varphi} C_{m-\varphi} \right) \mod 2$$  \hspace{1cm} (3.13)
According to (3.7), the term $B^{-(\varphi - 1)}$ is the term with the lowest resolution present in the CVNS digits. Therefore, only the terms greater than $B^{-(\varphi - 1)}$ should be considered. Thus, (3.13) can be written in the following form:

$$ (\langle z \rangle)_m = \left( \sum_{i=-N_{fy}}^{N_{iu}-1} y_i (\langle x \rangle)_{m-i} + B^{-(\varphi - 1)} \left\lfloor \frac{C_{m-\varphi}}{B} \right\rfloor \right) \mod B \tag{3.14} $$

where $\lfloor \rfloor$ is the floor function.

In (3.14), the CVNS output $(\langle z \rangle)_m$ and the CVNS input $(\langle x \rangle)_{m-i}$ have a resolution of $\varphi$. Therefore, the proposed algorithm is compatible with the environment resolution and provides accurate results in the low-resolution environment. Moreover, the multiplication operands $X$ and $Y$ are represented by the analog CVNS digits $(\langle x \rangle)_{m-i}$ and the digits $y_i$ respectively. Therefore, the proposed multiplication algorithm accepts the inputs $X$ and $Y$ in the analog and digital format respectively. Thus, it may be a suitable candidate for VLSI implementation of high resolution mixed-signal synapse multipliers using low-resolution analog mixed-signal circuits.

### 3.2 VLSI Implementation of the CVNS Synapse Multiplier for Neurochips with On-Chip Learning

In this section, VLSI implementation of the CVNS synapse multiplier for on-chip neurochips based on the proposed CVNS multiplication algorithm is discussed. The VLSI implementation is realized using TSMC CMOS 0.18\textmu m technology.

A synapse multiplier with a resolution of $16 \times 8$ bits is required for neurochips with on-chip learning [18,19]. Therefore, a CVNS synapse multiplier with a resolution of 16 by 8 bits is considered for implementation.

The synaptic weight is stored on the digital registers. The output of the activation function which is the other input to the multiplier is in the CVNS format. Therefore, to use (3.14) for implementation of the synapse multiplier, weights are denoted by $y$ while the other input is denoted by the CVNS digits $(\langle x \rangle)$.

As shown in [18], variation range of (-8,8) as synaptic weight range is adequate. Therefore, 3 integer bits are required to represent the weight variation range. Moreover, since the weights are signed variables, 1 bit is required for the sign bit. The 12 remaining bits are used to represent the fractional part of the weights.

Weights are stored in the 2’s complement format. To have the multiplication result in the 2’s complement format correctly, the sign bit extension is exploited. For the 16 by 8 bit multiplication,
the sign bit is extended by 8 bits to produce the correct result. Considering the sign extension, both 
\( N_{iy} \) and \( N_{fy} \) parameters in (3.14) are equal to 12.

The output of the sigmoid activation function is always positive and less than one. Therefore, 
all of the 8 bits representing the output of the sigmoid activation function are fractional digits. This 
results in \( N_{ix} = 0 \) and \( N_{fx} = 8 \). According to (3.7), the CVNS digit set \( \{ ((x)_2, (x)_1, \ldots, (x)_{-8} \} \) can 
represent the output of the sigmoid activation function in the CVNS format. Moreover, the radix of 
CVNS is considered to be two. This provides the most efficient radix for conversion between binary 
and CVNS. According to simulations performed in [20–22], the environment resolution for CVNS is 
four bits. Therefore, assuming an environment resolution of 4 and a radix of 2, (3.14) can be written 
in the following form:

\[
((z)_m) = \left( \sum_{i=-12}^{11} y_i((x))_{m-i} + 2^{-3} \left\lfloor \frac{C_{m-4}}{16} \right\rfloor \right) \mod 2 \tag{3.15}
\]

In the equation above, due to the sign extension, \( y_4 \) to \( y_{11} \) are equal to each other and have the 
same value as \( y_3 \) which is the sign bit of the synaptic weight.

The multiplication result resolution for a 16\( \times \)8 multiplier is 24 bits. Since the variation range 
of the inputs to the multiplier are (-8,8) and (0,1), the output range of the multiplier is (-8,8). 
Therefore, 3 bits are required to represent the integer part of the multiplication result while 1 bit 
represents the sign bit. The remaining 20 bits represent the fractional part of the multiplication 
result. Since the environment resolution is four, each CVNS digit includes the information of four 
bits. Therefore, six CVNS digits can represent the 24-bits multiplication result. Thus, the CVNS 
digit set \( \{ ((z)_3, ((z))_{-1}, ((z))_{-5}, ((z))_{-9}, ((z))_{-13}, ((z))_{-17} \} \) represents the output of the CVNS 
multiplier.

Current-mode circuits are used to realize (3.15). In the VLSI implementation of (3.15), 8 \( \mu A \) is 
indicator of 1. Therefore, (3.15) can be written in the following form:

\[
((z)_m) = \left( \sum_{i=-12}^{11} y_i((x))_{m-i} + 1\mu A \times \left\lceil \frac{C_{m-4}}{16\mu A} \right\rceil \right) \mod 16 \mu A \tag{3.16}
\]

To implement (3.16), the basic building blocks required for its implementation are realized. 
Considering that the addition in the current-mode circuits is easily performed through wiring the 
nodes carrying the signals, the main building blocks required for implementation of (3.16) are mod 
16 \( \mu A \) and \( \left\lceil \frac{C_{m-4}}{16\mu A} \right\rceil \). Therefore, the VLSI implementation of these blocks is discussed next.
3.2.1 VLSI implementation of mod 16 μA

The mod 16 μA operation can be written in the following form:

\[
x \mod 16 \mu A = \begin{cases} 
  x & x < 16 \mu A \\
  x - 16 \mu A & x \geq 16 \mu A 
\end{cases}
\] (3.17)

where \( x \) is the input to mod16μA block.

The mod 16 μA operation circuit is shown in Fig. 3.1. The circuit is composed of five main sections including the input current mirror, the current comparator, the current subtractor, the inverter chain and the output current mirror. Before going through different sections of the circuit, it should be noted that (3.17) can be modified in the following form:

\[
x \mod 16 \mu A = \begin{cases} 
  \left(\frac{x}{2}\right) \times 2 & \frac{x}{2} < 8 \mu A \\
  \left(\frac{x}{2} - 8 \mu A\right) \times 2 & \frac{x}{2} \geq 8 \mu A 
\end{cases}
\] (3.18)

According to (3.18), the current comparator and subtractor sections should compare the \( \frac{x}{2} \) with 8 μA and subtract the \( \frac{x}{2} \) from 8 μA. This in turn reduces the current in these sections which results in power consumption reduction of the mod 16 μA block.

The input current mirror is a low-voltage cascode current mirror which copies the input current divided by two to the current comparator and the current subtractor sections precisely. The transistors \( M_1 \) and \( M_2 \) generate the bias voltage required for the current mirror while the transistors \( M_3 \) to \( M_8 \) form a cascode current mirror. The \( \frac{W}{L} \) of the transistors \( M_5 \) to \( M_8 \) is half of the transistors \( M_3 \) and \( M_4 \). Therefore, the input current divided by two will be copied to the current comparator and subtractor sections.

The current comparator is based on the structure developed in [23] and compares the input current with a reference current of 8 μA. The 8 μA reference current is generated by the transistors \( M_9 \) to \( M_{11} \). The generated reference current is copied to the transistors \( M_{12} \) and \( M_{13} \). Output of the comparator is connected to the inverter chain which provides a rail to rail output. The current comparator along with the inverter chain generates the CmpN and CmpP signals equal to 0 and 1.8 volt provided that the input current to the circuit is greater than 16 μA.

The two transistors \( M_{16} \) and \( M_{17} \) act as a transmission gate which turns on provided that the input current is greater than 16 μA. This allows the 8 μA reference current to flow through these transistors. Therefore, when the input is greater than 16 μA, the 8 μA reference current is subtracted from the input current divided by two and flows through the transistors \( M_{18} \) and \( M_{19} \). Otherwise, half of the input current flows through the transistors \( M_{18} \) and \( M_{19} \).
To copy the current through the transistors $M_{18}$ and $M_{19}$ to the output precisely, a current mirror with high output resistance is required. The transistors $M_{18}$ to $M_{21}$ form a Wilson current mirror.
3. CVNS SYNAPSE MULTIPLIER FOR ROBUST NEUROCIPS WITH ON-CHIP LEARNING

Table 3.1: Transistor sizes of the mod16μA circuit

<table>
<thead>
<tr>
<th>Transistor</th>
<th>(W/L) (µm/µm)</th>
<th>Transistor</th>
<th>(W/L) (µm/µm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1</td>
<td>(0.22/0.18)</td>
<td>M15</td>
<td>(2.77/0.18)</td>
</tr>
<tr>
<td>M2</td>
<td>(0.22/0.18)</td>
<td>M16</td>
<td>(1/0.5)</td>
</tr>
<tr>
<td>M3</td>
<td>(0.44/0.18)</td>
<td>M17</td>
<td>(1/0.5)</td>
</tr>
<tr>
<td>M4</td>
<td>(0.22/0.18)</td>
<td>M18</td>
<td>(2.5/0.18)</td>
</tr>
<tr>
<td>M5</td>
<td>(0.22/0.18)</td>
<td>M19</td>
<td>(2.5/0.18)</td>
</tr>
<tr>
<td>M6</td>
<td>(0.22/0.18)</td>
<td>M20</td>
<td>(5/0.18)</td>
</tr>
<tr>
<td>M7</td>
<td>(0.22/0.18)</td>
<td>M21</td>
<td>(5/0.18)</td>
</tr>
<tr>
<td>M8</td>
<td>(0.22/0.18)</td>
<td>M22</td>
<td>(0.9/0.18)</td>
</tr>
<tr>
<td>M9</td>
<td>(2.77/0.18)</td>
<td>M23</td>
<td>(0.22/0.18)</td>
</tr>
<tr>
<td>M10</td>
<td>(2.77/0.18)</td>
<td>M24</td>
<td>(0.9/0.18)</td>
</tr>
<tr>
<td>M11</td>
<td>(2.77/0.18)</td>
<td>M25</td>
<td>(0.22/0.18)</td>
</tr>
<tr>
<td>M12</td>
<td>(2.77/0.18)</td>
<td>M26</td>
<td>(0.9/0.18)</td>
</tr>
<tr>
<td>M13</td>
<td>(2.77/0.18)</td>
<td>M27</td>
<td>(0.22/0.18)</td>
</tr>
<tr>
<td>M14</td>
<td>(2.77/0.18)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The Wilson current mirror provides a high output resistance. Moreover, using this current mirror, the voltage at node S of Fig. 3.1 is VDD-(VDS_{19}+VGS_{18}). By proper sizing of the transistors M_{18} and M_{19}, this voltage provides the proper voltage to keep the transistors M_{7}, M_{8}, M_{14} and M_{15} in saturation. The \( \frac{W}{L} \) of the transistors M_{20} and M_{21} is two times of the transistors M_{18} and M_{19}. Therefore, the current through the transistors M_{18} and M_{19} is doubled and copied to the output of the mod 16 μA circuit. The transistor sizes of the mod 16 μA circuit are shown in Table 3.1.
3. CVNS SYNAPSE MULTIPLIER FOR ROBUST NEUROCHIPS WITH ON-CHIP LEARNING

3.2.2 VLSI implementation of $\left\lfloor \frac{C_{m-4}}{16 \mu A} \right\rfloor$

According to (3.12), $(z)_{m-4}$ is obtained by applying the mod $16 \mu A$ operation to $C_{m-4}$. The mod $16 \mu A$ circuit implemented in the previous section is applied to the summation of two input currents. Therefore, mod $16 \mu A$ operation is applied to each summation term of $C_{m-4}$. The CmpN output signal of the inverter chain in the mod $16 \mu A$ circuit is an indicator that the input current to this block is greater than $16 \mu A$. Therefore, the number of CmpN signals equal to 1.8 volt in the mod $16 \mu A$ circuits used to evaluate $(z)_{m-4}$ is equal to $\left\lfloor \frac{C_{m-4}}{16 \mu A} \right\rfloor$. Thus, each cmpN signal generated by the $(z)_{m-4}$ evaluation circuit should generate a $1 \mu A$ current which is used to evaluate the $(z)_m$.

The circuit used to implement $1 \mu A \times \left\lfloor \frac{C_{m-4}}{16 \mu A} \right\rfloor$ is shown in Fig. 3.2. The transistors $M_1$ to $M_3$ generate a current of $1 \mu A$. This current is copied to the transistors $M_4$ to $M_{21}$ using the cascode current mirror provided that the corresponding cmp$_i$ signal is active. The cmp$_0$ to cmp$_5$ inputs are connected to the CmpN outputs of the mod $16 \mu A$ circuits used to evaluate $(z)_{m-4}$. Therefore, this circuit implements the $1 \mu A \times \left\lfloor \frac{C_{m-4}}{16 \mu A} \right\rfloor$ required for evaluation of $(z)_m$. All of the transistors have the same $W/L$ equal to $0.3 \times 0.18 \mu m$. The implemented circuit has a maximum of 6 inputs. Therefore, based on the number of CmpN signals generated in the evaluation of $(z)_{m-4}$, the appropriate number of $1 \mu A \times \left\lfloor \frac{C_{m-4}}{16 \mu A} \right\rfloor$ block should be used to evaluate $(z)_m$.

3.2.3 VLSI implementation of the CVNS synapse multiplier

Using the basic building blocks designed in the previous sections, VLSI implementation of the CVNS multiplier is conducted. According to (3.16), the CVNS digits representing the multiplication result in the CVNS format can be shown in the following form:

$$
\begin{align*}
(z)_3 &= \left( \sum_{i=1}^{11} y_i (x)_{3-i} + 1 \mu A \times \left\lfloor \frac{C_{-1}}{16 \mu A} \right\rfloor \right) \mod 16 \mu A \\
(z)_{-1} &= \left( \sum_{i=-3}^{7} y_i (x)_{-1-i} + 1 \mu A \times \left\lfloor \frac{C_{0}}{16 \mu A} \right\rfloor \right) \mod 16 \mu A \\
(z)_{-5} &= \left( \sum_{i=-7}^{3} y_i (x)_{-5-i} + 1 \mu A \times \left\lfloor \frac{C_{-5}}{16 \mu A} \right\rfloor \right) \mod 16 \mu A \\
(z)_{-9} &= \left( \sum_{i=-11}^{-1} y_i (x)_{-9-i} + 1 \mu A \times \left\lfloor \frac{C_{-9}}{16 \mu A} \right\rfloor \right) \mod 16 \mu A \\
(z)_{-13} &= \left( \sum_{i=-12}^{-5} y_i (x)_{-13-i} + 1 \mu A \times \left\lfloor \frac{C_{-13}}{16 \mu A} \right\rfloor \right) \mod 16 \mu A \\
(z)_{-17} &= \left( \sum_{i=-12}^{-9} y_i (x)_{-17-i} \right) \mod 16 \mu A
\end{align*}
$$

(3.19)

VLSI implementation of different CVNS digits representing the multiplication result is discussed next.
VLSI implementation of $\langle z \rangle_{-17}$

Block diagram of the circuit used for implementation of $\langle z \rangle_{-17}$ is shown in Fig. 3.3. As may be noted from Fig. 3.3, the input currents $\langle x \rangle_{-5}$ to $\langle x \rangle_{-8}$ are applied to the transistors $M_1$ to $M_4$. These transistors act as switches which turn on provided that the corresponding $y_i$ input to their gate is high. This in turn implements the $y_i x_{-17-i}$ terms required to calculate $\langle z \rangle_{-17}$.

The input currents after passing the input transistors are summed at the input nodes of the mod 16 $\mu A$ blocks and mod 16 $\mu A$ operation is applied to them. Output of the mod 16 $\mu A$ blocks which receive the input currents are wired together to perform the addition. The addition result goes through another mod 16 $\mu A$ block. This generates the $\langle z \rangle_{-17}$ output CVNS digit. The three cmp$_{-170}$ to cmp$_{-172}$ signals generated by the mod 16 $\mu A$ blocks are used as an input to the $1 \mu A \times \left\lceil \frac{C_{16-4}}{16 \mu A} \right\rceil$ circuit used to evaluate $\langle z \rangle_{-13}$.

VLSI implementation of $\langle z \rangle_{-13}$

The block diagram of the circuit used for implementation of $\langle z \rangle_{-13}$ is shown in Fig. 3.4. The operation principle of the circuit is the same as the circuit used for evaluation of the $\langle z \rangle_{-17}$. As
can be seen from Fig. 3.4, the cmp\_{170} to cmp\_{172} generated by the \((z)_{-17}\) evaluation block are used as input to the \(1 \mu A \times \left\lfloor \frac{C_{-17}}{16 \mu A} \right\rfloor\) block. Since this block has 6 inputs and only three inputs are used, the remaining inputs are disabled by connecting them to the ground.

**VLSI implementation of \((z)_{-9}, (z)_{-5}, (z)_{-1}\) and \((z)_{3}\)**

All of these CVNS digits can be implemented using the same structure. The structure used for evaluation of \((z)_{-9}, (z)_{-5}, (z)_{-1}\) and \((z)_{3}\) is shown in Fig. 3.5. The structure operation is similar to the ones used for evaluation of \((z)_{-17}\) and \((z)_{-13}\).

To evaluate the CVNS digits \((z)_{-9}, (z)_{-5}, (z)_{-1}\) and \((z)_{3}\), the proper inputs to the structure shown in Fig. 3.5 is applied. The CVNS inputs and the digital weights are denoted by \((x)_i\) and \(y_j\) respectively. The outputs of the mod 16 \(\mu A\) are shown as cmp\_{160} to cmp\_{1610}. The inputs to \(1 \mu A \times \left\lfloor \frac{C_{m-4}}{16 \mu A} \right\rfloor\) blocks are shown as cmp\_{00} to cmp\_{111}. These are the CmpN outputs of mod 16 \(\mu A\) blocks used to evaluate the \((z)_{m-4}\) output CVNS digit. The output of this structure is denoted as \((z)_m\). The parameters of the structure shown in Fig. 3.5 for different output CVNS digits are summarized in Table 3.2.
Using the circuits explained in the previous sections, the proposed CVNS synapse multiplier is laid out. The layout of the proposed CVNS synapse multiplier is shown in Fig. 3.6. The layout has an area equal to 14953.67 $\mu m^2$.

### 3.3 Post-Layout Simulation

In this section, post-layout simulation of the proposed CVNS synapse multiplier is conducted. Performance of the VLSI implementation of the proposed CVNS synapse multiplier is verified by conducting post-layout simulations for various inputs.

Fig. 3.7 shows the post-layout simulation results for four different input values. These inputs...
are applied with intervals of 0.5 $\mu$s to the CVNS multiplier. The digital weights $y$ and the CVNS inputs $\langle x \rangle$ corresponding to these four inputs are shown in Table 3.3. Using (3.19), the expected output for each input can be calculated. The relation between the output CVNS digits with inputs to the CVNS multiplier is summarized in Table 3.3. The post-layout simulation results shown in Fig. 3.7 are in good agreement with the mathematical derivations presented in Table 3.3.
3.4 Comparison with Previously developed CVNS multiplication algorithm

In this section, effect of the proposed CVNS multiplication algorithm on the NSR of a CVNS Adaline is discussed. Moreover, comparison of the NSR of a CVNS Adaline using the proposed CVNS multiplication algorithm versus previously developed multiplication algorithm is carried out.

The previously developed multiplication algorithm is limited by the environment resolution. In [17], the effect of this limitation on the NSR of a CVNS Adaline is studied. The study shows that...
Table 3.3: Synapse multiplier input values and their corresponding output

<table>
<thead>
<tr>
<th>y</th>
<th>((x))</th>
<th>((z))</th>
</tr>
</thead>
<tbody>
<tr>
<td>1111 1111 1111 1111</td>
<td>{1 μA, 3 μA, 6 μA, 12 μA, 8 μA, 1 μA, 3 μA, 15 μA, 3 μA, 10 μA}</td>
<td>{15 μA, 15 μA, 15 μA, 15 μA, 3 μA, 10 μA}</td>
</tr>
<tr>
<td>0010 1010 1010 1010</td>
<td>{0 μA, 0 μA, 0 μA, 0 μA, 1 μA, 3 μA, 7 μA, 15 μA, 14 μA, 12 μA, 8 μA}</td>
<td>{0 μA, 2 μA, 7 μA, 15 μA, 15 μA, 6 μA}</td>
</tr>
<tr>
<td>0111 1111 0000 0000</td>
<td>{1 μA, 2 μA, 5 μA, 10 μA, 5 μA, 10 μA, 5 μA, 10 μA, 5 μA, 5 μA, 4 μA, 5 μA}</td>
<td>{5 μA, 4 μA, 5 μA, 5 μA, 10 μA, 0 μA}</td>
</tr>
<tr>
<td>1101 0101 0101 0101</td>
<td>{1 μA, 3 μA, 7 μA, 15 μA, 14 μA, 12 μA, 8 μA, 0 μA, 0 μA, 0 μA, 0 μA}</td>
<td>{14 μA, 7 μA, 15 μA, 15 μA, 11 μA, 0 μA}</td>
</tr>
</tbody>
</table>

Figure 3.7: Post-layout simulation results of the proposed CVNS synapse multiplier

limitation of the previously developed multiplication algorithm increases the NSR of the Adaline. This in turn degrades the output response.
The NSR of the CVNS Adaline developed in [15] in a low-resolution environment using the previously developed CVNS multiplication algorithm can be shown in the following form [17]:

\[
NSR_l = g \left( \frac{\sigma_x \sigma_w}{B^{DD} \sqrt{N}} \right) \times \left( \frac{\sigma_{\Delta x}^2}{\sigma_x^2} + \frac{\sigma_{\Delta w}^2}{\sigma_w^2} \right) \tag{3.20}
\]

where \( g \) is the stochastic gain function. The stochastic gain is a function of \( \frac{\sigma_x \sigma_w}{B^{DD} \sqrt{N}} \). \( N \) is the number of inputs, \( \sigma_x \) and \( \sigma_w \) are standard deviations of inputs and weights respectively. \( \Delta x \) and \( \Delta w \) are the input and weight errors respectively and \( DD \) is obtained in the following form:

\[
DD + 1 = \left\lceil \frac{D + 1}{\varphi} \right\rceil \tag{3.21}
\]

where \( D + 1 \) is the number of CVNS digits used.

As shown in the previous sections, the proposed CVNS multiplication algorithm eliminates the effect of low-resolution environment on the multiplication result. Therefore, the low-resolution environment has no effect on the NSR of CVNS Adaline when the proposed CVNS synapse multiplier is exploited. Thus, the NSR of a CVNS Adaline using the proposed multiplication algorithm is in the following form [17]:

\[
NSR_p = g \left( \frac{\sigma_x \sigma_w}{B^{DD} \sqrt{N}} \right) \times \left( \frac{\sigma_{\Delta x}^2}{\sigma_x^2} + \frac{\sigma_{\Delta w}^2}{\sigma_w^2} \right) \tag{3.22}
\]

The stochastic gain function, \( g(x) \), for \( x \leq 1 \), is approximately equal to 1, while for \( x > 1 \) can be estimated as follows [24]:

\[
g(x) = 0.5 + 0.53 \times x \tag{3.23}
\]

According to (3.23), the stochastic gain is a linear function of its input. Therefore, \( g(x) \propto x \) and (3.20) and (3.22) can be written in the following form:

\[
NSR_l \propto \frac{\sigma_x \sigma_w}{B^{DD} \sqrt{N}} \times \left( \frac{\sigma_{\Delta x}^2}{\sigma_x^2} + \frac{\sigma_{\Delta w}^2}{\sigma_w^2} \right) \tag{3.24}
\]

\[
NSR_p \propto \frac{\sigma_x \sigma_w}{B^{DD} \sqrt{N}} \times \left( \frac{\sigma_{\Delta x}^2}{\sigma_x^2} + \frac{\sigma_{\Delta w}^2}{\sigma_w^2} \right) \tag{3.25}
\]

Dividing (3.20) by (3.22) results in the following equation:

\[
\frac{NSR_p}{NSR_l} \propto B^{DD-D} \tag{3.26}
\]

Considering that \( D \) is always greater than \( DD \), the NSR of the CVNS Adaline using the proposed algorithm will always be lower. This results in an Adaline with lower sensitivity to input and weight errors. Therefore, the proposed CVNS multiplication algorithm can be used to design synapse multiplier for robust neurochips with on-chip learning. To illustrate this, a case study is provided.
3. CVNS SYNAPSE MULTIPLIER FOR ROBUST NEUROCHIPS WITH ON-CHIP LEARNING

Figure 3.8: NSR of the CVNS Adaline using the proposed multiplication algorithm versus the previously developed multiplication algorithm

Case Study: A CVNS Adaline with 16-bit weight storage resolution is considered. Inputs and weights of the Adaline are uniformly distributed in the range of (-8,8). The input and weight variance is equal to \( \sigma_x^2 = \sigma_w^2 = \frac{16^2}{12} \). The radix of the CVNS is considered to be two. Number of the CVNS digits used in the CVNS-DNN Adaline, \( D + 1 \), is equal to four. The environment resolution is four. Therefore, \( DD + 1 \) is equal to one.

NSR of the CVNS Adaline for number of inputs in the range of [2,10] using the previous and proposed algorithm is calculated. This is conducted using (3.20) and (3.22). The results are shown in Fig. 3.8. As may be noted from Fig. 3.8, the proposed multiplication algorithm results in an Adaline with lower NSR. Therefore, the proposed algorithm provides neural network structures which are more input and weight error tolerant.

3.5 Conclusion

A new CVNS multiplication algorithm is proposed in this paper. The proposed algorithm provides accurate results in the low-resolution environment. Moreover, the VLSI implementation of a CVNS synapse multiplier for neurochips with on-chip learning is realized. The post-layout simulations of the implemented CVNS synapse multiplier confirms its performance. The comparison of NSR of the CVNS Adaline using the proposed CVNS multiplication algorithm versus a CVNS Adaline using the previously developed multiplication algorithm is conducted. The comparison shows that...
the proposed CVNS multiplication algorithm provides a lower NSR. Therefore, the proposed CVNS multiplication algorithm provides more robust neural network structures.
3.6 References


REFERENCES


Chapter 4

Mixed-Signal VLSI Neural Network
Based on Continuous Valued Number System

ASIC implementation of neural networks has been exploited in various applications. Examples include pattern recognition [1], real-time surface discrimination [2], and, smart sensing [3].

The ASIC implementation methods of neural networks may be categorized as analog, digital or mixed-signal. In the analog neural networks, both weight storage and processing are conducted using analog circuits. When implemented by analog circuits, neural networks typically possess a higher energy efficiency, and require less area, in comparison with their identical digital implementation. However, the capacitor-based weight storage in analog designs requires refreshing and is susceptible to process and power supply variations [4].

In the digital implementation of a neural network, both weight storage and processing are carried out in the digital domain. The third implementation method, mixed-signal, utilizes digital registers for weight storage and analog circuits for signal processing. This method benefits from the ease of weight storage in digital registers while capitalizing on the advantages of analog domain such as compact addition and nonlinear neuron. Therefore, the mixed-signal method may be a suitable candidate for the ASIC implementation of neural networks.

One of the main obstacles in using the mixed-signal method is the limited precision of the analog
signal which is limited by the precision of the analog circuits used. The precision of the analog circuits is referred as the environment resolution in the rest of this paper.

To use the benefits of the analog circuits while keeping the accuracy high, the Continuous Valued Number System (CVNS) can be utilized. The information represented by each CVNS analog digit is the same as the environment resolution. However, collectively a set of digits can increase the precision of analog processing. This makes the CVNS appropriate for implementing high precision analog and mixed-signal circuits [5–7].

The results of [5–7] show that implementation of the CVNS in TSMC 0.18 µm and 90 nm with 4-bit resolution for each analog digit is viable. Since the TSMC 0.18 µm technology has been exploited in this paper, the same setting is used.

In this paper, a mixed-signal 2-2-1 CVNS neural network structure is proposed. The proposed structure realizes the XOR function. The weights of the network are stored in the digital registers with 16-bit resolution. The signal processing of the proposed structure is based on the CVNS arithmetic. Using the CVNS arithmetic, the resolution requirements of neural networks is satisfied.

The proposed network is designed and laid out in 0.18 µm technology. The performance of the network is confirmed by the post-layout simulations.

This paper is organized as follows. The proposed structure for the 2-2-1 CVNS network and its VLSI implementation are explained in section 4.1. Post-layout simulation results are discussed in section 4.2. Finally, conclusions are drawn in section 4.3.

### 4.1 VLSI Implementation of the CVNS Neural Network

In this section, the proposed structure and its VLSI implementation are discussed. The proposed structure is a 2-2-1 network. Weights are stored in the digital registers while the signal processing is based on the CVNS arithmetic.

The block diagram of the proposed structure is shown in Fig. 4.1. It is composed of four main units including the input to CVNS converter, the hidden Adaline, the output Adaline, and the output to binary converter. The proposed network realizes the two input XOR function.

The input to CVNS converter transforms the binary input to CVNS. The hidden Adaline receives the input in the CVNS format. The CVNS input is multiplied by the weights stored in the digital registers. The weights are stored with a 16-bit resolution. This makes the proposed structure compatible with the weight storage resolution required by neural networks [8]. Since the weight and input resolution are 16-bits and 1-bit respectively, a CVNS multiplier with a resolution of 16×1 is...
4. MIXED-SIGNAL VLSI NEURAL NETWORK BASED ON CONTINUOUS VALUED NUMBER SYSTEM

Figure 4.1: The block diagram of the 2-2-1 CVNS network realizing the XOR function

required for the hidden Adalines.

The bias stored in the digital registers is converted to the CVNS using the bias to CVNS converter block. Then, the outputs of the CVNS multipliers and the bias to CVNS converter block are added together by the CVNS adder. Since the resolution of the hidden layer multipliers is equal to $16 \times 1$ and the bias resolution is 16-bits, the hidden layer CVNS adder required resolution is 18-bits.

The output of the CVNS adder passes through the CVNS sigmoid function. The CVNS sigmoid function is based on the structure developed in [9]. This structure provides an output with 8-bits resolution. This in turn generates the output of the hidden Adaline.

The output Adaline is similar to the hidden Adaline. The difference is that the inputs to the CVNS multipliers in this layer are the outputs of the CVNS sigmoid functions of the hidden Adalines. Since the output resolution of the CVNS sigmoid function is 8-bits [9], CVNS multipliers with a resolution of $16 \times 8$ are required for this layer. The output Adaline CVNS adder performs the addition operation on the outputs of the two $16 \times 8$ CVNS multiplier and the bias to CVNS converter.
4. MIXED-SIGNAL VLSI NEURAL NETWORK BASED ON CONTINUOUS VALUED NUMBER SYSTEM

Table 4.1: Required resolution of different arithmetic units

<table>
<thead>
<tr>
<th>Arithmetic Unit</th>
<th>Required Resolution (bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hidden layer multiplier</td>
<td>16×1</td>
</tr>
<tr>
<td>Hidden layer adder</td>
<td>18</td>
</tr>
<tr>
<td>Output layer multiplier</td>
<td>16×8</td>
</tr>
<tr>
<td>Output layer adder</td>
<td>26</td>
</tr>
<tr>
<td>CVNS sigmoid function</td>
<td>8</td>
</tr>
</tbody>
</table>

Therefore, it requires a resolution of 26-bits. The output to binary converter transforms the output of the output Adaline to binary format.

The resolution requirements of different arithmetic units in the proposed CVNS network are summarized in Table 4.1. It is worth noting that the required resolution for various arithmetic units is higher than the environment resolution. However, the CVNS arithmetic used for the VLSI implementation makes the signal processing with the desired resolution feasible.

VLSI implementation of the proposed structure is realized using current-mode circuits. Generally, current-mode circuits provide lower power consumption, higher speed and have the ability of working with lower power supply voltages. Moreover, some arithmetic operations such as addition can be easily realized. The VLSI implementation of each block is discussed in detail next.

4.1.1 Input to CVNS Converter

The absolute value of a real number \( X \) using fixed-point binary number system format can be shown as follows:

\[
X = \sum_{i=-N_f}^{N_i-1} x_i \times 2^i
\]  

(4.1)

where \( N_i \) and \( N_f \) are the number of integer and fractional digits, while \( x_i \) is a binary digit.

A radix-2 CVNS digit \( ([X])_m \) representing \( X \) can be truncated to the following equation [7]:

\[
([X])_m = \sum_{i=m-\varphi+1}^{m} x_i \times 2^{i-m}
\]  

(4.2)

where \( \varphi \) is the environment resolution. As previously discussed, the environment resolution for the CVNS in 0.18\( \mu \)m technology is equal to four.
Using (4.2) with an environment resolution of four, the CVNS digit set \(\langle X \rangle = \{ (X)_3, (X)_2, (X)_1, (X)_0 \} = \{ x_0, x_2, x_3, x_4 \} \) can represent the one-bit input to the XOR network in CVNS format.

In the VLSI implementation of the network, 8 \(\mu\)A is indicator of 1. Therefore, the VLSI implementation of the input to CVNS converter should generate the CVNS digit set \(\langle X \rangle = \{ 1 \mu\)A, 2 \(\mu\)A, 4 \(\mu\)A, 8 \(\mu\)A \} provided that the input is one. Otherwise, all of the CVNS digits representing the input will be zero.

The circuit configuration of the binary input to CVNS conversion for each CVNS digit \((X)_i\) is shown in Fig. 4.2. The input of the circuit is equal to \(X\) while the output represents \(\langle X \rangle_i\).

In the circuit shown in Fig. 4.2, the transistor \(M_1\) acts as a switch which turns on provided that the input is one. By proper sizing of the transistors \(M_2\) and \(M_3\), the current corresponding to the CVNS digit \((X)_i\) is generated. The two transistors \(M_3\) and \(M_4\) act as a current mirror which copies the current generated by the transistors \(M_2\) and \(M_3\) to the output.

### 4.1.2 Hidden Adaline

The two hidden Adalines shown in Fig. 4.1 are composed of four main signal processing units including CVNS multiplier, bias to CVNS converter, CVNS adder and CVNS sigmoid function. Each block will be discussed subsequently.

**Multiplier**

The CVNS inputs to the hidden Adaline are multiplied by the weights stored in the digital registers. The CVNS multiplier in the hidden Adaline performs the mentioned multiplication.

Multiplication of the digital weights by the input in the CVNS format can be carried out using the following equation [10]:

\[
\langle Y \rangle_m = \left( \sum_{i=-N_{iw}}^{N_{iw}-1} w_i \langle X \rangle_{m-i} + 2^{-(\varphi-1)} \left\lfloor \frac{C_{m-\varphi}}{2} \right\rfloor \right) \mod 2 \tag{4.3}
\]

where \(N_{iw}\) and \(N_{fw}\) are the number of integer and fractional bits representing the weights respectively, \(w_i\) is the binary digits of the weight, \(\langle X \rangle_m\) and \(\langle Y \rangle_m\) are the CVNS digits representing the input and the multiplication result respectively. \(C_{m-\varphi}\) can be obtained in the following form [10]:

\[
C_{m-\varphi} = \sum_{i=-N_{fw}}^{N_{iw}-1} w_i \langle X \rangle_{m-\varphi-i} \tag{4.4}
\]

A resolution of 16-bits with a range of (-8,8) for weight and bias storage is required [8]. 4 integer bits are required to cover the (-8,8) input range. The remaining 12 bits are the fractional bits.
4. MIXED-SIGNAL VLSI NEURAL NETWORK BASED ON CONTINUOUS VALUED NUMBER SYSTEM

Figure 4.2: VLSI implementation of the binary input to CVNS conversion, $b_i \times 8 \mu A \times 2^{i-m}$ and $1 \mu A \times \left( C_{m-\varphi} \text{cmp} 16 \mu A \right)$

Since the weights and the input have a resolution of 16-bits and 1-bit respectively, the output resolution of the CVNS multiplier is 16-bits. Moreover, the same number of integer and fractional bits used to represent the weights can represent the multiplication result. Therefore, using (4.1) and (4.2), with an environment resolution of four, the CVNS digit set $\{\langle Y \rangle_3, \langle Y \rangle_{-1}, \langle Y \rangle_{-5}, \langle Y \rangle_{-9} \}$ can represent the output of the hidden Adaline CVNS multiplier. The $\langle Y \rangle_3$ CVNS digit includes the information of the integer bits while $\langle Y \rangle_{-1}$, $\langle Y \rangle_{-5}$, and, $\langle Y \rangle_{-9}$ CVNS digits represent the information of the fractional bits.

The maximum of (4.4) occurs when all the $w_i$ bits are one and the CVNS digits $\langle X \rangle_m$ are at their maximum value. For the hidden Adaline CVNS multiplier, it can be easily proven that the $C_{m-\varphi}$ will always be less than two. Therefore, (4.3) can be modified in the following form:

$$\langle Y \rangle_m = \left( \sum_{i=1}^{N_{iw} - 1} w_i \langle X \rangle_{m-i} \right) \mod 2 \quad (4.5)$$

The maximum of $\sum_{i=-N_{fw}}^{N_{iw} - 1} w_i \langle X \rangle_{m-i}$ happens when all the $w_i$ bits are one and the CVNS digits $\langle X \rangle_{m-i}$ are at their maximum value. Therefore, $\sum_{i=-N_{fw}}^{N_{iw} - 1} w_i \langle X \rangle_{m-i} < 2$. If the input to mod2 function is less than 2, it passes the input to the output without change. Thus, (4.5) can be written
4. MIXED-SIGNAL VLSI NEURAL NETWORK BASED ON CONTINUOUS VALUED NUMBER SYSTEM

Figure 4.3: VLSI implementation of the first layer multiplier

as follows:

\[
(Y)_m = \sum_{i=-N_{fw}}^{N_{fw}-1} w_i (X)_m - i
\]  

(4.6)

Considering that the input CVNS digit set \( (X) \) includes only the CVNS digits \( (X)_0 \) to \( (X)_3 \), (4.6) can be simplified in the following form:

\[
(Y)_m = \sum_{i=m-3}^{m} w_i (X)_m - i
\]  

(4.7)

The VLSI implementation of (4.7) is shown in Fig. 4.3. VLSI implementation of \( w_i (X)_m - i \) terms are realized using the transistors which receive \( (X)_i \) as their inputs and are controlled by the \( w_i \) bits. Since the current-mode circuits are exploited, the summation term is easily implemented through wiring the transistors outputs.

**Bias to CVNS Converter**

The bias is stored in the digital registers with the same resolution and variation range of the weights. Since the CVNS adder accepts its inputs in the CVNS format, the bias value stored in the digital registers should be converted to CVNS.

The 16-bit bias can be represented by four CVNS digits. Since the bias variation range is (-8,8), it is represented with 4 integer bits and 12 fractional bits. Therefore, using (4.2), the CVNS digit set \( (B)_3 = \{(B)_3, (B)_{-1}, (B)_{-5}, (B)_{-9}\} \) can represent the bias in the CVNS format. The \( (B)_3 \) includes the information of the four integer bits while \( (B)_{-1} \), \( (B)_{-5} \), and, \( (B)_{-9} \) include the information of the twelve fractional bits.

Keeping in mind that in the implemented hardware, 8 µA is indicator of one and using (4.2),
the CVNS digits representing the bias can be shown in the following form:

\[
\langle (B) \rangle_m = \sum_{i=m-3}^{m} b_i \times 8 \mu A \times 2^{i-m}
\] (4.8)

According to (4.8), the \(b_m, b_{m-1}, b_{m-2}\) and, \(b_{m-3}\) should generate 8 \(\mu\) A, 4 \(\mu\) A, 2 \(\mu\) A and, 1 \(\mu\) A respectively provided that the corresponding bias bit is one.

The VLSI implementation of the \(b_i \times 8 \mu A \times 2^{i-m}\) terms is carried out using the same circuit configuration shown in Fig. 4.2. To implement these terms, the \(b_i\) is applied to the input of the circuit while the output is \(b_i \times 8 \mu A \times 2^{i-m}\). By proper sizing of the transistors \(M_2\) and \(M_3\), each bias bit, \(b_i\), generates the respective current equal to \(8 \mu A \times 2^{i-m}\) provided that the \(b_i\) is one. The summation term present in (4.8) is implemented through wiring the output of circuits realizing the \(b_i \times 8 \mu A \times 2^{i-m}\) terms.

**CVNS Adder**

The CVNS adder is designed based on the CVNS addition algorithm developed in [9]. Addition of two CVNS operands is carried out by the addition of their corresponding CVNS digits. Addition of two CVNS digits \(\langle (Y_1) \rangle_m\) and \(\langle (Y_2) \rangle_m\) in a low-resolution environment is conducted based on the following equation [9]:

\[
\langle (Z) \rangle_m = \left[ \langle (Y_1) \rangle_m + \langle (Y_2) \rangle_m + 1 \mu A \times \left( C_{m-\varphi} \right. \right. \right. \left. \left. \text{cmp} \right. \left. \left. 16 \mu A \right) \right] \mod 16 \mu A \quad (4.9)
\]

where \(\langle (Z) \rangle_m\) is the CVNS digit representing the addition result. \(C_{m-\varphi}\) can be obtained using the following equation [9]:

\[
C_{m-\varphi} = \langle (Y_1) \rangle_{m-\varphi} + \langle (Y_2) \rangle_{m-\varphi} \quad (4.10)
\]

The \(C_{m-\varphi}\) cmp 16 \(\mu\) A can be written in the following form [9]:

\[
C_{m-\varphi} \text{ cmp } 16 \mu A = \begin{cases} 
1 & C_{m-\varphi} \geq 16 \mu A \\
0 & C_{m-\varphi} < 16 \mu A
\end{cases} \quad (4.11)
\]

The block diagram of the CVNS adder for two CVNS digits \(\langle (Y_1) \rangle_m\) and \(\langle (Y_2) \rangle_m\) is shown in Fig. 4.4. The two CVNS digits \(\langle (Y_1) \rangle_m\) and \(\langle (Y_2) \rangle_m\) along with the output of the 1 \(\mu\) A × \(C_{m-\varphi}\) cmp 16 \(\mu\) A block are wired together. Since the current-mode circuits are used, this realizes the summation term present in (4.9). Afterwards, the mod 16 \(\mu\) A operation is applied. This in turn generates the CVNS digit representing the addition result.

The VLSI implementation of 1 \(\mu\) A × \(C_{m-\varphi}\) cmp 16 \(\mu\) A is accomplished using the circuit configuration shown in Fig. 4.2. To realize the 1 \(\mu\) A × \(C_{m-\varphi}\) cmp 16 \(\mu\) A, the input is equal to
Figure 4.4: Block diagram of the CVNS adder

$Ca_{m-\varphi}$ cmp 16 $\mu$A while the output is $1 \mu A \times \left( Ca_{m-\varphi} \text{ cmp } 16 \mu A \right)$. The transistor M$_1$ acts as a switch which turns on provided that $Ca_{m-\varphi}$ is greater than 16 $\mu$A. Transistors M$_2$ and M$_3$ are sized to generate the 1 $\mu$A current. The 1 $\mu$A current is copied to the output through the current mirror constituted by the transistors M$_3$ and M$_4$. The VLSI implementation of the mod 16 $\mu$A is explained in detail in [10]. The same circuit is used here.

**CVNS Sigmoid Function**

A CVNS sigmoid function evaluation structure is developed in [9]. The same structure is used in the proposed network. The CVNS sigmoid function receives the input from the CVNS adder and provides the output in the CVNS format. The output resolution of the CVNS sigmoid function is equal to 8-bits.

### 4.1.3 Output Adaline

The output Adaline is similar to the hidden Adaline. However, the resolution of the CVNS multiplier implemented in the output Adaline is different. Since the weight storage resolution and the CVNS sigmoid function output resolution are 16-bits and 8-bits respectively, a CVNS multiplier with a resolution of $16 \times 8$ bits should be implemented. A CVNS multiplier with the required resolution of $16 \times 8$ is developed in [10]. The same multiplier is used in the output Adaline of the proposed CVNS network.
4. MIXED-SIGNAL VLSI NEURAL NETWORK BASED ON CONTINUOUS VALUED NUMBER SYSTEM

4.1.4 Output to Binary Converter

The output to binary converter transforms the CVNS output to binary format. The radix-2 CVNS digits can be converted back to the binary digits using the following equation [9]:

\[
x_m = \begin{cases} 
1 & \langle X \rangle_m \geq 8\mu A \\
0 & \langle X \rangle_m < 8\mu A 
\end{cases}
\]  \hspace{1cm} (4.12)

The CVNS sigmoid function developed in [9] provides the output in the CVNS format using two CVNS digits. Since the output is only one bit, the CVNS digit including the information of higher significant bits is converted back to the binary.

To perform the comparison, the current comparator developed in [11] is exploited. The VLSI implementation of the output to binary converter is shown in Fig. 4.5. The transistors M₁ and M₂

![Figure 4.5: VLSI implementation of the output to binary converter](image)
act as a current mirror which copies the input current to the current comparator. The transistors M
and M generate an 8 µA current. This current is copied to the transistor M through the current
mirror formed by the transistors M and M. The current comparator composed of the transistors
M and M compares the input current with the 8 µA generated by the transistors M and M. The
inverter chain converts the output of the current comparator to a rail to rail output.

4.2 Simulation Results

Using the developed circuits discussed in the previous sections, the proposed network is laid out.
The layout of the implemented network is shown in Fig. 4.6. The layout has an area equal to
113031.79 µm². The designed layout is sent for fabrication. The layout of the chip sent for fabrication
is shown in Fig. 4.7.

The developed network is trained off-line. Weights are calculated in MATLAB. Then, the calcu-
lated weights are loaded into the digital registers to conduct the simulations.

The post-layout simulation results are shown in Fig. 4.8. Two inputs with different periods
are applied to cover all possible combinations. As may be noted from Fig. 4.8, the output exactly
matches the expected output for different combinations of inputs.

The post-layout simulation results show a worst case propagation delay of 99.68 ns. To measure
the power consumption, a uniform random bit stream including 1000 inputs with an input rate of
99.68 ns is applied to the network. The simulation result shows a power consumption of 17.59 mW.
Area, delay and power consumption of the implemented network are summarized in Table 4.2.
Table 4.2: Area, delay and power consumption of the implemented network

<table>
<thead>
<tr>
<th>Area ($\mu m^2$)</th>
<th>Delay ($ns$)</th>
<th>Power consumption ($mW$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>113031.79</td>
<td>99.68</td>
<td>17.59</td>
</tr>
</tbody>
</table>

### 4.3 Conclusion

A 2-2-1 mixed-signal CVNS neural network structure is proposed in this paper. Using the CVNS arithmetic, the problem of limited resolution of analog signal processing in mixed-signal neural networks is resolved. This opens a path to design mixed-signal structures which satisfy the signal processing resolution requirements of neural networks. The proposed 2-2-1 network is designed, laid out, and post-layout simulated in 0.18$\mu m$ CMOS technology. The post-layout simulations confirm the operation of the proposed network.
4.4 References


Chapter 5

Area-Efficient Robust Madaline Based on Continuous Valued Number System

5.1 Introduction

Since the advent of artificial neural networks, software and hardware implementation methods have been used for their realization. The hardware implementation methods have been exploited in applications that require real-time and energy-efficient processing [1–5].

Neural network hardware implementation methods may be classified as analog, digital or mixed-signal. In the analog implementation, both weight storage and processing are in the analog domain. When implemented by analog circuits, neural networks typically maintain a higher energy efficiency, a lower number of interconnections, and require less area, in comparison with its equivalent digital implementation. However, the capacitor-based weight storage methods require a refresh signal and are sensitive to process and power supply variations [6]. In the digital implementation of a neural network, both weight storage and processing are done in the digital domain. The third implementation method, mixed-signal, uses digital registers for weight storage and analog circuits for signal processing. This method exploits the ease of weight storage in digital registers while capitalizing on the advantages of analog domain such as compact addition and nonlinear neuron.

The basic theory behind the neural network can be essentially described as a series of weights, that when applied to distinct inputs, provide the appropriate corresponding output. Due to the
limited word length available in hardware implementations of neural networks, inputs and weights are represented with finite precision; This degrades the output response. Therefore, the robustness to input and weight errors becomes a key issue in neural network architectures. To quantify this issue, the sensitivity of neural networks to input and weight imprecisions has been studied extensively [7–14].

The sensitivity analysis can be classified as one of two approaches: the geometrical approach or the statistical approach. The geometrical method uses a hypersphere or a hyper-rectangle model to analyze the output sensitivity, while the statistical method determines the sensitivity by calculating the Noise-to-Signal Ratio (NSR). The NSR is defined as the ratio of the variance of output error to the variance of ideal output. Therefore, a network with a lower NSR is more tolerant to input and weight errors. In this paper, the statistical approach is utilized to study the effects of input and synaptic weight perturbations on the output.

Alternative number system can be exploited to design low NSR neural network architectures. The Continuous Valued Number System (CVNS) is a mixed-signal number system in which its analog digits share information. This feature enables multiple error correction in a digit set [15] and makes this number system a candidate for implementing analog/mixed-signal neural networks with a low NSR [16–18].

A CVNS neural network is presented in [16]. This structure is analog, in which the synaptic weights are stored in the CVNS format and processing is accomplished using CVNS arithmetic. To decrease the output error, the output of the Adaline is generated based on the Reverse Evolution (RE) process [15]. This architecture is referred to as CVNS-RE in the rest of this paper.

Using the information redundancy present in the CVNS and the distributed neuron structure, two architectures named CVNS-DNN and CVNS-FDNN were developed in [17]. Similar to the CVNS-RE architecture, both of these structures store the weights in the CVNS format while the processing is carried out using the CVNS arithmetic.

The previously developed CVNS neural network architectures [16–18] store synaptic weights in analog memories which require a refresh signal and are sensitive to process and power supply variations. Moreover, the NSR improvement of the previous architectures is achieved at the cost of extra neurons, which results in area overhead and increased power consumption. Although NSR is generally a suitable measure to show the effect of input and weight errors on the network output, the efficiency of different architectures can not be measured by this quantity alone. A better indicator of the efficiency of a network model is the product of the total number of neurons in the network multiplied by the network NSR, this provides a better estimate in terms of area and power.
consumption for a specific NSR level.

To investigate the efficiency of various Madaline architectures based on the previous CVNS Adalines, mathematical analysis of NSR and neuron×NSR of these architectures is required. It is worth noting that only the NSR of CVNS Adalines has been considered previously. The mathematical analysis performed in this paper allows the efficiency evaluation of different neural network architectures.

In this paper, a new mixed-signal Adaline is proposed. The proposed architecture stores the weights in digital registers while the arithmetic is based on the CVNS number system. Using digital registers to store weights eliminates the need for a refresh signal and provides a process and power supply variation tolerant storage mechanism, eliminating the problems caused by the analog weight storage methods used in the previous CVNS structures [16–18].

In the previously developed CVNS networks, the total number of analog memories required for each Adaline is proportional to the number of CVNS digits representing the synaptic weights. However, the total number of registers in the proposed CVNS Adaline is independent of the number of CVNS digits. This in turn results in reduced weight storage elements which leads to lower area overhead and lower power consumption. Furthermore, the RE process is used to decrease the error in the CVNS digits, which improves the NSR.

Using the proposed Adaline, a Madaline named CVNS-distributed is proposed. The CVNS-distributed Madaline uses the proposed CVNS Adaline at its output layer. The NSR and the neuron×NSR of the proposed Madaline is formulated and compared with conventional lumped and distributed as well as previous CVNS structures. This analysis proves that the proposed architecture is more immune to input and weight errors while simultaneously requiring a lower number of neurons for a specific NSR, resulting in an architecture with a lower area overhead and lower power consumption. In this research, a three-layer Madaline is implemented in TSMC CMOS 0.18μm. The implementation results confirm the advancement of the proposed Madaline in terms of NSR and the area consumption required for a specific NSR.

The rest of this paper is organized as follows. In the next section, CVNS is briefly introduced. The NSR and the stochastic model proposed by Piche [10] is briefly discussed in section 5.3. Moreover, the NSR of previous Adalines is summarized in this section. The mathematical derivation of the total number of neurons and NSR of a Madaline, based on the previous structures, is provided in section 5.4, where the related equations are developed. The proposed Adaline and Madaline structure is explained in section 5.5. Comparisons with existing structures are carried out in section 5.6. The VLSI implementation of a three-layer Madaline and comparisons with previous structures
are conducted in section 5.7. Finally, conclusions are drawn in section 5.8.

5.2 Continuous Valued Number System (CVNS)

The absolute value of a real number $x$ using fixed-point number representation with a radix of $B$ can be shown as follows:

$$x = \sum_{i=-N_f}^{N_i-1} x_i \times B^i$$  \hspace{1cm} (5.1)

where $N_i$ and $N_f$ are the number of integer and fractional digits, while $x_i$ is the digit.

CVNS is a number system with continuous valued digits. CVNS analog digits representing the real number $x$ can be generated as follows [15]:

$$(x)_i = (x \times B^{-i}) \mod B$$  \hspace{1cm} (5.2)

where $((x))_i$ is the CVNS digit and $-N_f \leq i \leq N_i-1$. An ensemble of CVNS digits $((x)) = \{((x))_{N_i-1}, ..., (x)_0, (x)_{-1}, ..., (x)_{-N_f}\}$ represent the input $x$ in CVNS format.

**Example: Finding the CVNS digit set of $x = 81.92$ with a radix of 10**

According to (5.1), the number of integer digits, $N_i$, and fractional digits, $N_f$, required for representation of $x = 81.92$ is equal to two. Therefore, the CVNS digit set $((x)) = \{(x)_1, (x)_0, (x)_{-1}, (x)_{-2}\}$ can represent the $x = 81.92$ in CVNS format. To obtain the CVNS digits, (5.2) is used. This results in the CVNS digit set $((x)) = \{8.192, 1.92, 9.2, 2\}$.

As can be seen, the CVNS digits share information. The information sharing between CVNS digits allows for digit-level error detection, error correction and accurate arithmetic based on analog and mixed-signal circuits. The digit-level error correction is carried out through the Reverse Evolution (RE) process [15].

5.3 Noise to Signal Ratio of Previous Adalines

Piche [10] presented a stochastic model for the NSR of a lumped Adaline. In the general structure of lumped neural networks, inputs are multiplied by weights in the synapse, summed at the node, and passed through a nonlinear function in the neuron [19].

The stochastic model of lumped Adaline presented in [10] is based on the input NSR, $\frac{\sigma^2_x}{\sigma^2_x}$, and the weight NSR, $\frac{\sigma^2_w}{\sigma^2_w}$, and is as follows:

$$\text{NSR} = g(\sqrt{N} \sigma_x \sigma_w) \times \left( \frac{\sigma^2_x}{\sigma^2_x} + \frac{\sigma^2_w}{\sigma^2_w} \right)$$  \hspace{1cm} (5.3)
5. AREA-EFFICIENT ROBUST MADALINE BASED ON CONTINUOUS VALUED NUMBER SYSTEM

Figure 5.1: (a) Ideal stochastic gain function and its approximation (b) Approximation error of the stochastic gain function

The Adaline NSR, $\frac{\sigma^2_y}{\sigma^2_o}$, is the ratio of the variance of output error to the variance of ideal output, and depends on the stochastic gain function, $g$. This gain function is a function of $\sqrt{N}\sigma_x\sigma_w$, where $N$ is the number of inputs, and $\sigma_x$ and $\sigma_w$ are the standard deviation of inputs and weights, respectively.

Assuming $x = \sqrt{N}\sigma_x\sigma_w$, the stochastic gain function, $g(x)$, can be estimated as follows:

$$g(x) = \begin{cases} 
1 & x \leq 1 \\
0.5 + 0.53 \times x & x > 1 
\end{cases}$$ (5.4)

The stochastic gain function and its approximation are shown in Fig. 5.1a, while approximation error is shown in Fig. 5.1b, showing a maximum approximation error of 15.5%. According to (5.4), the stochastic gain for inputs greater than one is a linear function of its inputs, while for inputs less than one, has a constant value and is equal to one.

The stochastic model for NSR of an Adaline can be represented by a flow diagram as shown in Fig. 5.2 [10]. In this flow diagram, the input and weight errors, $\frac{\sigma^2_x}{\sigma^2_o}$ and $\frac{\sigma^2_w}{\sigma^2_o}$, are summed at a node, and then are multiplied by the stochastic gain, resulting in $\frac{\sigma^2_y}{\sigma^2_o}$, which represents the NSR of the Adaline.

The stochastic model developed by Piche [10] is used in [16,17,19] to analyze the NSR of the developed Adalines. The NSR analyses performed in these papers are summarized in Table 5.1.

As shown in Table 5.1, the NSR of the Adalines developed in [16,17,19] is similar to the NSR of
5. AREA-EFFICIENT ROBUST MADALINE BASED ON CONTINUOUS VALUED NUMBER SYSTEM

Figure 5.2: NSR flow diagram of an Adaline

Table 5.1: NSR of the previous Adalines

<table>
<thead>
<tr>
<th>Structure</th>
<th>NSR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lumped [10]</td>
<td>( g(\sqrt{N}\sigma_x\sigma_w) \times \left( \frac{\sigma_x^2}{\sigma_w^2} + \frac{\sigma_w^2}{\sigma_x^2} \right) )</td>
</tr>
<tr>
<td>Distributed [19]</td>
<td>( g(\frac{\sigma_x\sigma_w}{\sqrt{N}}) \times \left( \frac{\sigma_x^2}{\sigma_w^2} + \frac{\sigma_w^2}{\sigma_x^2} \right) )</td>
</tr>
<tr>
<td>CVNS-RE [16]</td>
<td>( g(\frac{\sigma_x\sigma_w}{\sqrt{N}}) \times B^{-2D} \times \left( \frac{\sigma_x^2}{\sigma_w^2} + \frac{\sigma_w^2}{\sigma_x^2} \right) )</td>
</tr>
<tr>
<td>CVNS-DNN [17]</td>
<td>( g(\frac{\sigma_x\sigma_w}{B^{D}\sqrt{N}}) \times \left( \frac{\sigma_x^2}{\sigma_w^2} + \frac{\sigma_w^2}{\sigma_x^2} \right) )</td>
</tr>
<tr>
<td>CVNS-FDNN [17]</td>
<td>( g(\frac{\sigma_x\sigma_w}{B^{D(D+1)}\sqrt{N}}) \times \left( \frac{\sigma_x^2}{\sigma_w^2} + \frac{\sigma_w^2}{\sigma_x^2} \right) )</td>
</tr>
</tbody>
</table>

the lumped Adaline [10], with the difference that they have a modified stochastic gain function. In the distributed structure [19], the stochastic gain function has been changed to \( g(\frac{\sigma_x\sigma_w}{\sqrt{N}}) \), while the stochastic gain of the CVNS-RE structure [16] is equal to the multiplication of \( g(\frac{\sigma_x\sigma_w}{\sqrt{N}}) \) by \( B^{-2D} \). It should be noted that \( B \) is the radix of the CVNS digits, while \( D + 1 \) CVNS digits are used to store the synaptic weights. Moreover, the \( B^{-2D} \) factor is indicative of the RE process [15] used in this structure. The CVNS-DNN and CVNS-FDNN structures developed in [17] have a stochastic gain function equal to \( g(\frac{\sigma_x\sigma_w}{B^{D}\sqrt{N}}) \) and \( g(\frac{\sigma_x\sigma_w}{B^{D(D+1)}\sqrt{N}}) \), respectively. A comparison of these structures in terms of NSR and neuron×NSR will be performed in section 5.6.

5.4 Mathematical Analysis of Madaline Structures

The Adaline is the basic element of a neural network in which the inputs are multiplied by weights, summed at the node, and pass through a nonlinear activation function. The Madaline is a multilayer extension of the Adaline. In a Madaline, the output of every Adaline is used as an input to all Adalines in the next layer [20]. The general configuration of a Madaline is shown in Fig. 5.3. The Madaline has \( L + 1 \) layers with \( N_0 \) inputs, and \( N_i \) Adalines in each layer \( i \).
In this section, the mathematical derivation of the total number of neurons and the NSR of several Madalines is performed. The NSR of a multilayer network is affected by the errors in the inputs to each layer, as well as errors in the synapse weights. These errors and variations are caused by various sources in the network; for example, circuit nonlinearity can create errors at the output of the neurons, which leads to errors in the input values of the next layer. However, the main source of error in synapse weights is the inevitable quantization error present in all weight storage methods. The mathematical analysis of the NSR of different architectures performed in this section, provides information regarding the sensitivity of different networks to input and weight errors. Moreover, the mathematical derivation of the total number of neurons, along with the NSR analysis of different networks conducted in this section, makes the efficiency evaluation of different structures in terms of neuron×NSR performed in section 5.6 feasible.

5.4.1 Lumped Structure

In the lumped structure, each Adaline is composed of one neuron to generate the output. Therefore, the total number of neurons for a lumped Madaline with \( L + 1 \) layers is as follows:

\[
N_{LM} = \sum_{i=1}^{L} N_i
\]
where \( N_i \) is the number of Adalines in layer \( i \), \( i = 0 \) denotes the input layer which has no neurons, \( i = 1 \) and \( i = L \) denote the first hidden layer and the output layer, respectively.

The NSR of a Madaline can be derived by cascading the NSR of Adalines of each layer. It should be noted that \( \frac{\sigma_w^2}{\sigma_x^2} \) is the weight storage quantization error while \( \frac{\sigma_x^2}{\sigma_w^2} \) is the Adaline input error. Considering that in a Madaline, all weights are stored with the same resolution, the quantization error of weight storage for all layers is the same. The inputs to every Adaline, except the Adalines in the first hidden layer, are the outputs of the previous layer Adalines. Considering that all Adalines of a Madaline are the same, and assuming the same input error for Adalines in the first hidden layer, the input error to all Adalines in a Madaline will be the same. Therefore, \( \frac{\sigma_w^2}{\sigma_x^2} \) and \( \frac{\sigma_x^2}{\sigma_w^2} \) for all layers of a Madaline are the same. Thus, the NSR of the lumped Madaline structure can be written as follows:

\[
\text{NSR}_{LM} = \left( \sum_{j=1}^{L} \prod_{k=j}^{L} g_k \right) \times \left( \frac{\sigma_x^2}{\sigma_w^2} + \frac{\sigma_w^2}{\sigma_x^2} \right) \tag{5.6}
\]

where \( g_k = g(\sqrt{N_{k-1}}\sigma_x\sigma_w) \) is the stochastic gain function of layer \( k \), in which \( N_{k-1} \) is the number of Adalines in layer \( k - 1 \) [10].

According to (5.6), the NSR of a Madaline increases as the number of inputs and network layers increases, provided that the stochastic gain function is in its linear region. The stochastic gain function of the output layer is present in all product terms of a Madaline NSR. Therefore, the output layer of a network has the greatest impact on the NSR of a Madaline.

### 5.4.2 Distributed Structure

In the distributed structure [19], each Adaline is divided into sub-neurons, where their number is equal to the number of inputs. Therefore, the total number of sub-neurons used in each layer of the distributed Madaline is equal to the number of Adalines in that layer multiplied by the number of Adalines in the previous layer. The total number of neurons in the distributed Madaline can be written in the following form:

\[
N_{DM} = \sum_{i=1}^{L} (N_i \times N_{i-1}) \tag{5.7}
\]

Using the NSR of the distributed Adaline, the NSR of a distributed Madaline can be written in the following form:

\[
\text{NSR}_{DM} = \left( \sum_{j=1}^{L} \prod_{k=j}^{L} g'_k \right) \times \left( \frac{\sigma_x^2}{\sigma_w^2} + \frac{\sigma_w^2}{\sigma_x^2} \right) \tag{5.8}
\]

where \( g'_k = g(\frac{\sigma_x\sigma_w}{\sqrt{N_{k-1}}}). \)
According to (5.8), using the distributed structure with one neuron for each input synapse results in a self-scaling property that controls the Madaline NSR, when the number of inputs to Madaline, as well as number of Adalines in different layers, increases.

### 5.4.3 CVNS-RE Structure

In the CVNS-RE structure [16], each Adaline is divided into sub-neurons, and the number of sub-neurons is equal to the number of Adaline inputs multiplied by the number of the CVNS digits used for each synaptic weight. Therefore, the total number of sub-neurons used in the CVNS-RE Madaline using \( D + 1 \) CVNS digits for each weight, for all layers is as follows:

\[
N_{CREM} = \sum_{i=1}^{L} (N_i \times N_{i-1} \times (D + 1))
\]

(5.9)

Using the NSR of the CVNS-RE Adaline, the NSR of the CVNS-RE Madaline is as follows:

\[
\text{NSR}_{CREM} = \sum_{j=1}^{L} \prod_{k=j}^{L} \left( g_k \times B^{-2D} \right) \times \left( \frac{\sigma_x^2}{\sigma_z^2} + \frac{\sigma_w^2}{\sigma_z^2} \right)
\]

(5.10)

According to (5.10), the information redundancy present in the CVNS digits along with the RE process, reduces the effect of the stochastic gain function of different layers, which consequently improves the Madaline NSR.

Since the RE method is used for each layer, and for each Adaline, the network delay is increased considerably. This limits the application of this type of network to off-chip training schemes.

### 5.4.4 CVNS-DNN and CVNS-FDNN Structures

In the CVNS-DNN Adaline, the number of sub-neurons is equal to the number of Adaline inputs. Therefore, the total number of neurons in each layer is equal to the number of Adalines in the current layer multiplied by the number of Adalines in the previous layer. The total number of sub-neurons, similar to a distributed network, is equal to:

\[
N_{CDM} = \sum_{i=1}^{L} (N_i \times N_{i-1})
\]

(5.11)

Using the NSR of the CVNS-DNN Adaline, the NSR of a CVNS-DNN Madaline can be written in the following form:

\[
\text{NSR}_{CDM} = \sum_{j=1}^{L} \prod_{k=j}^{L} g_k'' \times \left( \frac{\sigma_x^2}{\sigma_z^2} + \frac{\sigma_w^2}{\sigma_z^2} \right)
\]

(5.12)

where we have \( g_k'' = g\left( \frac{\sigma_x \sigma_w}{B \sqrt{N_{k-1}}} \right) \).
According to (5.12), the developed CVNS-DNN structure decreases the stochastic gain function of different layers, provided that the stochastic gain function is in its linear region. This results in the reduction of Madaline NSR, making the Madaline more tolerant to input and weight errors.

In the CVNS-FDNN Adaline, the number of sub-neurons is equal to the number of its inputs multiplied by the number of CVNS digits used for synapse weight storage. The total number of sub-neurons in this structure can be written in the following form:

\[ N_{CFDM} = \sum_{i=1}^{L} (N_i \times N_{i-1} \times (D + 1)) \] (5.13)

Using the NSR of the CVNS-FDNN Adaline, the NSR of the CVNS-FDNN Madaline can be written in the following form:

\[ \text{NSR}_{CFDM} = \prod_{j=1}^{L} \prod_{k=j}^{L} \sigma_{\Delta x}^{\prime\prime} \times \left( \frac{\sigma_{\Delta x}^2}{\sigma_x^2} + \frac{\sigma_{\Delta w}^2}{\sigma_w^2} \right) \] (5.14)

where we have \( \sigma_{\Delta x}^{\prime\prime} = \sigma_{\Delta x} \left( \frac{\sigma_{\Delta w}}{B^D(D+1)\sqrt{N_{k-1}}} \right) \).

According to (5.12) and (5.14), the stochastic gain function reduction of different layers in the CVNS-FDNN Madaline is a function of \( B^D(D+1) \), while in the CVNS-DNN Madaline, it is function of \( B^D \). Therefore, the CVNS-FDNN Madaline provides a lower NSR in the linear region of the stochastic gain function and is more immune to input and weight errors.

The NSR analysis of different Madalines, along with mathematical derivation of the total number of neurons, provides the platform to carry out the mathematical comparison of different Madalines in terms of NSR and neuron×NSR conducted in section 5.6.

### 5.5 Proposed Architecture

All of the previous CVNS neural networks store the synaptic weights in analog memories which require a refresh signal, hence, are sensitive to process and power supply variation. The new CVNS Madaline structure, proposed in this section, overcomes this limitation.

In the proposed CVNS Madaline structure, synaptic weights are stored in digital registers, providing reliable and low complexity storage compared to the analog memory required for previous architectures. The arithmetic and signal processing is based on the CVNS arithmetic. Efficiency of the proposed network in terms of NSR and neuron×NSR will be presented in section 5.6.

According to (5.4), the Adaline stochastic gain function is a linear function of its inputs, provided that the input to the stochastic gain function is greater than one. Therefore, an increase in the
number of inputs, input variation, and weight variation, in the linear region of the stochastic gain function, results in an increase in the stochastic gain. Considering the NSR of the distributed Adaline, shown in Table 5.1, using distributed neurons reduces the stochastic gain in the linear region, and improves the NSR of the network.

The proposed Adaline is shown in Fig. 5.4. Here, distributed sub-neurons are used, and weights are stored in digital registers. The inputs to the Adaline are in the CVNS format, represented by \( D + 1 \) CVNS digits. The number of bits representing each weight in the network is denoted by \( N_w \). Additionally, the RE process is applied to the CVNS digits representing the Adaline output.

Inputs to the network are multiplied by the weights using the CVNS multiplier [15]. The weights are stored on the registers and denoted by \( w \), while the second input to the CVNS multiplier, \( ((y))_i \), is in CVNS format. The multiplication result, \( ((z))_i \), is in CVNS format and is as follows:

\[
((z))_i = \sum_{k=0}^{i} (w_k \times ((y))_{i-k}) \mod 2 \quad (5.15)
\]

According to (5.15), each of the output digits, \( ((z))_i \), exploits the \( w_0 \) to \( w_i \) bits of the weight storage registers corresponding to that input. Therefore, different CVNS digits representing the same input to the Adaline share the same weight registers for CVNS multiplication. In previously developed CVNS structures, the number of analog memory units required for each input to the Adaline is equal to the number of CVNS digits representing the weights corresponding to that input. The reduction in the number of storage medium elements in the proposed structure results in lower area and reduced power consumption.

Since the RE process is applied to the output, the error of the output is decreased by a factor of \( B^{-D} \), decreasing the NSR by \( B^{-2D} \). Therefore, the NSR of the proposed Adaline can be written in the following form:

\[
\text{NSR}_{PA} = g(\frac{\sigma_x \sigma_w}{\sqrt{N}}) \times B^{-2D} \times \left( \frac{\sigma_x^2}{\sigma_x^2} + \frac{\sigma_w^2}{\sigma_w^2} \right) \quad (5.16)
\]

According to (5.16), the information redundancy of the CVNS digits along with the RE process improves the NSR of the proposed Adaline. The distributed nature of the sub-neurons controls the NSR of the proposed Adaline when the number of inputs increases.

The NSR flow diagram of the proposed Adaline is shown in Fig. 5.5a. Here, the summation of input and weight errors is being multiplied by \( g' \times B^{-2D} \), where \( g' \) is equal to \( g(\frac{\sigma_x \sigma_w}{\sqrt{N}}) \).

The block diagram of the proposed CVNS-distributed Madaline structure for a three-layer network with \( N \) inputs, \( N \) hidden Adalines, and one output Adaline, is shown in Fig. 5.6. The hidden layer of the neural network, shown in Fig. 5.6, is based on the distributed structure, while the output
layer uses the proposed Adaline. Using the proposed Adaline at the output layer improves the NSR, while the distributed structure used in the hidden layers maintains a low total number of neurons.
5. AREA-EFFICIENT ROBUST MADALINE BASED ON CONTINUOUS VALUED NUMBER SYSTEM

Figure 5.5: (a) NSR flow diagram of the proposed Adaline (b) NSR flow diagram of the proposed CVNS-distributed Madaline

Figure 5.6: Block diagram of the proposed CVNS-distributed Madaline
5. AREA-EFFICIENT ROBUST MADALINE BASED ON CONTINUOUS VALUED NUMBER SYSTEM

To generate the input CVNS digits for the output layer, a CVNS generator block is required. The CVNS digit generation is done based on (5.2). The number of sub-neurons in the layers using the distributed structure can be calculated using (5.7). The only difference is that, in the proposed Madaline, only $L-1$ layers use the distributed structure. The number of sub-neurons of each Adaline in the output layer is equal to the product of number of Adalines in the previous layer multiplied by the number of CVNS digits used to represent the inputs to that Adaline. Therefore, the total number of neurons in the proposed Madaline can be written in the following form:

$$N_{PM} = \left( \sum_{i=1}^{L-1} (N_i \times N_{i-1}) \right) + (N_L \times N_{L-1} \times (D_L + 1))$$  (5.17)

where $D_L + 1$ is the number of CVNS digits used to represent the inputs to the output layer.

Using the NSR flow diagram of the proposed Adaline in combination with the NSR flow diagram of the distributed structure, the NSR flow diagram of the proposed CVNS-distributed Madaline is shown in Fig. 5.5b.

Based on the NSR flow diagram of the proposed Madaline, the NSR of the proposed Madaline can be written in the following form:

$$\text{NSR}_{PM} = B^{-2D_L} \times \left( \sum_{j=1}^{L} \prod_{k=j}^{L} g_k \right) \times \left( \frac{\sigma_x^2}{\sigma_{\Delta x}^2} + \frac{\sigma_w^2}{\sigma_{\Delta w}^2} \right)$$  (5.18)

According to (5.18), since the information redundancy provided by the CVNS digits in the output layer decreases all product terms of the stochastic gain functions by a factor of $B^{-2D_L}$, the proposed structure can exploit the CVNS information redundancy more efficiently. This will be proven in the next section.

5.6 Comparison of the Proposed Madaline Structure with Previous Architectures

In this section, a comparison between the proposed Madaline structure and the previous structures in terms of NSR and neuron×NSR is performed. The comparison is based on two assumptions. First, it is assumed that the total number of CVNS digits used in the proposed and previous architectures are the same. It should be noted that all previous CVNS neural network architectures use CVNS digits in all layers, while the proposed Madaline network uses CVNS digits only in the output layer. Therefore, to have the same total number of CVNS digits in all structures the number of CVNS digits used in the output layer of the proposed network is obtained as follows:

$$D_L = (D + 1) \times L$$  (5.19)
Table 5.2: Total number of neurons required for different Madaline structures and their normalized value with respect to lumped structure

<table>
<thead>
<tr>
<th>Madaline Structure</th>
<th>Total Number of Neurons</th>
<th>Normalized Total Number of Neurons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lumped</td>
<td>$N \times L$</td>
<td>1</td>
</tr>
<tr>
<td>Distributed</td>
<td>$N^2 \times L$</td>
<td>$N$</td>
</tr>
<tr>
<td>CVNS-RE</td>
<td>$N^2 \times (D + 1) \times L$</td>
<td>$N \times (D + 1)$</td>
</tr>
<tr>
<td>CVNS-DNN</td>
<td>$N^2 \times L$</td>
<td>$N$</td>
</tr>
<tr>
<td>CVNS-FDNN</td>
<td>$N^2 \times (D + 1) \times L$</td>
<td>$N \times (D + 1)$</td>
</tr>
<tr>
<td>Proposed Madaline</td>
<td>$N^2 \times (L - 1) + N^2 \times L \times (D + 1)$</td>
<td>$N \times (1 + (D + 1) - L^{-1})$</td>
</tr>
</tbody>
</table>

where $D_L$ is the number of CVNS digits used to represent the inputs to the output layer of the proposed Madaline, and $D + 1$ is the number of CVNS digits used for weight storage in each layer of the previous CVNS architectures. It should be noted that the desired number of input CVNS digits to the output layer of the proposed Madaline can be generated using the CVNS generator blocks shown in Fig. 5.6.

Moreover, it is assumed that the number of inputs, as well as the number of Adalines in each layer of all Madalines, are equal to $N$. Based on this assumption, the total number of neurons, or sub-neurons, required for the lumped, distributed, CVNS-RE, CVNS-DNN, CVNS-FDNN and proposed CVNS-distributed structures found through (5.5), (5.7), (5.9), (5.11), (5.13) and (5.17) can be simplified as shown in Table 5.2. It should be noted that $L + 1$ is the number of Madaline layers while $D + 1$ is indicator of the number of CVNS digits used for weight storage in each layer of the previous CVNS architectures. Also, the normalized value of the total number of neurons with respect to the lumped structure are summarized in this Table. The normalized total number of neurons are used to evaluate the efficiency of different architectures in terms of neuron × NSR.

According to Table 5.2, the CVNS-DNN architecture has the lowest total number of sub-neurons among the previous CVNS architectures, while the CVNS-RE and CVNS-FDNN architectures require the same number. The proposed CVNS-distributed Madaline requires a higher number of neurons compared to the previous CVNS architectures. Robustness of a network is measured in terms of NSR while its efficiency can be evaluated in terms of neuron × NSR.
5. AREA-EFFICIENT ROBUST MADALINE BASED ON CONTINUOUS VALUED NUMBER SYSTEM

Based on (5.4), the stochastic gain has two regions: linear and nonlinear. According to the simulations performed in [22], a synaptic weight range of [-8,8] is required while according to [23] the inputs should be normalized between [0,1] or [-1,1]. Assuming uniform distribution for input and weight variables, the stochastic gain function of the lumped structure will always be in the linear region, while the stochastic gain of the other previous structures, as well as the proposed structure, depending on the number of inputs, may be in the linear or nonlinear region. Therefore, the mathematical comparison of the proposed structure with the previous structures will be performed in each region. The first region is where the input to stochastic gain function is greater than two. The stochastic gain in this region is a linear function of its inputs. The second region is where the input to the stochastic gain function is less than two, where the stochastic gain function shows a nonlinear behavior. The mathematical discussion conducted in this section proves that the proposed Madaline has a better NSR and neuron×NSR in both the linear and nonlinear regions of the stochastic gain function.

5.6.1 Comparison of the Proposed Madaline Structure with Previous Structures in the Linear Region of Stochastic Gain Function

According to (5.4), the stochastic gain function for inputs greater than two is a linear function of its input. Therefore, it can be written:

\[ g(x) \propto x \quad (5.20) \]

Using (5.20), the NSR of the lumped, distributed, CVNS-RE, CVNS-DNN, CVNS-FDNN and proposed CVNS-distributed Madaline structures found through (5.6), (5.8), (5.10), (5.12), (5.14) and (5.18) can be written in the following form:

\[
\begin{align*}
\text{NSR}_{\text{LM}} & \propto \sum_{j=1}^{L} (\sqrt{N} \sigma_x \sigma_w)^j \\
\text{NSR}_{\text{DM}} & \propto \sum_{j=1}^{L} \left( \frac{\sigma_x \sigma_w}{\sqrt{N}} \right)^j \\
\text{NSR}_{\text{CREM}} & \propto \sum_{j=1}^{L} \left( \frac{\sigma_x \sigma_w}{\sqrt{N} \times B^{2D}} \right)^j \\
\text{NSR}_{\text{CDM}} & \propto \sum_{j=1}^{L} \left( \frac{\sigma_x \sigma_w}{\sqrt{N} \times B^{D}} \right)^j
\end{align*}
\]
Table 5.3: NSR of different structures and their normalized value with respect to lumped structure in linear region of stochastic gain function

<table>
<thead>
<tr>
<th>Structure</th>
<th>NSR</th>
<th>Normalized NSR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lumped [19]</td>
<td>$\propto (\sqrt{N}\sigma_x\sigma_w)^L \times (\frac{\sigma^2_x}{\sigma_x^2} + \frac{\sigma^2_w}{\sigma_w^2})$</td>
<td>1</td>
</tr>
<tr>
<td>Distributed [19]</td>
<td>$\propto (\frac{\sigma_x\sigma_w}{\sqrt{N}})^L \times (\frac{\sigma^2_x}{\sigma_x^2} + \frac{\sigma^2_w}{\sigma_w^2})$</td>
<td>$N^{-L}$</td>
</tr>
<tr>
<td>CVNS-RE [16]</td>
<td>$\propto B^{-2DL} \times (\frac{\sigma_x\sigma_w}{\sqrt{N}})^L \times (\frac{\sigma^2_x}{\sigma_x^2} + \frac{\sigma^2_w}{\sigma_w^2})$</td>
<td>$N^{-L} \times B^{-2DL}$</td>
</tr>
<tr>
<td>CVNS-DNN [17]</td>
<td>$\propto B^{-DL} \times (\frac{\sigma_x\sigma_w}{\sqrt{N}})^L \times (\frac{\sigma^2_x}{\sigma_x^2} + \frac{\sigma^2_w}{\sigma_w^2})$</td>
<td>$N^{-L} \times B^{-DL}$</td>
</tr>
<tr>
<td>CVNS-FDNN [17]</td>
<td>$\propto (B^D \times D)^{-L} \times (\frac{\sigma_x\sigma_w}{\sqrt{N}})^L \times (\frac{\sigma^2_x}{\sigma_x^2} + \frac{\sigma^2_w}{\sigma_w^2})$</td>
<td>$N^{-L} \times (B^D \times D)^{-L}$</td>
</tr>
<tr>
<td>Proposed Madaline</td>
<td>$\propto B^{-(D+1)L-1} \times (\frac{\sigma_x\sigma_w}{\sqrt{N}})^L \times (\frac{\sigma^2_x}{\sigma_x^2} + \frac{\sigma^2_w}{\sigma_w^2})$</td>
<td>$N^{-L} \times B^{-2(D+1)L-1}$</td>
</tr>
</tbody>
</table>

\[
\text{NSR}_{\text{CFDM}} \propto \sum_{j=1}^{L} \left(\frac{\sigma_x\sigma_w}{\sqrt{N} \times B^D \times D}\right)^j \quad (5.25)
\]

\[
\text{NSR}_{\text{PM}} \propto B^{-2(D+1)L-1} \times \sum_{j=1}^{L} \left(\frac{\sigma_x\sigma_w}{\sqrt{N}}\right)^j \quad (5.26)
\]

The above equations are geometric sequences which can be written in the following general form:

\[
\sum_{j=1}^{L} \alpha^j = \frac{\alpha - \alpha^{L+1}}{1 - \alpha} \quad (5.27)
\]

The above summation term can be approximated as $\alpha^L$ provided that $\alpha^L \gg 1$. Since the input to the stochastic gain in this region is greater than two, we have $\alpha > 2$. Considering that a Madaline has at least three layers, (5.27) can be approximated as $\alpha^L$. Using this approximation, the NSR of different structures presented in (5.21) to (5.26) can be summarized as shown in Table 5.3. The normalized values of the NSR for various structures with respect to the lumped structure are shown in this Table as well. Comparing the normalized NSR of different architectures indicates that the proposed structure has the lowest NSR compared to the previous architectures. Therefore, the synapse weight errors have less effect on the network output which indicates that the proposed structure is more robust to weight errors present in the hardware implementation.

Using the normalized values of the total number of neurons and the NSR of different structures in Table 5.2 and Table 5.3, the results of the normalized neuron×NSR of all structures are provided in Table 5.4.
Table 5.4: Normalized neuron × NSR of different structures in linear region of stochastic gain function

<table>
<thead>
<tr>
<th>Structure</th>
<th>Normalized Neuron × NSR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lumped [19]</td>
<td>1</td>
</tr>
<tr>
<td>Distributed [19]</td>
<td>$N^{1-L}$</td>
</tr>
<tr>
<td>CVNS-RE [16]</td>
<td>$N^{1-L} \times (D+1) \times B^{-2DL}$</td>
</tr>
<tr>
<td>CVNS-DNN [17]</td>
<td>$N^{1-L} \times B^{-DL}$</td>
</tr>
<tr>
<td>CVNS-FDNN [17]</td>
<td>$N^{1-L} \times (D+1) \times (B^D \times D)^{-L}$</td>
</tr>
<tr>
<td>Proposed Madaline</td>
<td>$N^{1-L} \times (1 + (D+1) - L^{-1}) \times B^{-2DL} \times B^{-2(L-1)}$</td>
</tr>
</tbody>
</table>

According to Table 5.4, the CVNS-DNN architecture has a better performance than the distributed structure. To compare CVNS-FDNN and CVNS-DNN structures, the normalized neuron × NSR of the CVNS-FDNN architecture is divided by neuron × NSR of the CVNS-DNN structure resulting in the following equation:

\[(D + 1) \times D^{-L}\]  \hspace{1cm} (5.28)

Since a Madaline has at least three layers, the CVNS-FDNN structure has a better neuron × NSR. Dividing the normalized neuron × NSR of the CVNS-RE architecture by that of the CVNS-FDNN architecture, we obtain the following equation:

\[(B^{-D} \times D)^{-L} < 1\]  \hspace{1cm} (5.29)

Therefore, the CVNS-RE structure has a better neuron × NSR than that of the CVNS-FDNN. This in turn means that the CVNS-RE structure has the best neuron × NSR among all of the previously developed structures.

To compare the efficiency of the proposed architecture with CVNS-RE structure, the normalized neuron × NSR of the proposed architecture is divided by the normalized neuron × NSR of the CVNS-RE structure which results in the following equation:

\[((D + 1)^{-1} + 1 - ((D + 1) \times L)^{-1}) \times B^{-2(L-1)}\]  \hspace{1cm} (5.30)

The maximum of (5.30) happens when $B$, $D$ and $L$ are at their minimum value. A Madaline has at least three layers, while the minimum radix and number of CVNS digits is equal to two. Therefore, (5.30) is always less than one and the proposed CVNS-distributed architecture has a better
Figure 5.7: Normalized Neuron $\times$ NSR improvement of different architectures compared to distributed structure

performance compared to the CVNS-RE structure. Consequently, the proposed CVNS-distributed architecture achieves the best neuron $\times$ NSR of all structures. The proposed architecture can provide the same accuracy as previous architectures by utilizing a lower total number of sub-neurons. In other words, the proposed network obtains the same accuracy with a lower area overhead and reduced power consumption.

The normalized neuron $\times$ NSR improvement for different architectures compared to the distributed structure is shown in Fig. 5.7. All architectures have a better neuron $\times$ NSR compared to the distributed structure, while clearly, the normalized neuron $\times$ NSR of the distributed structure is better than that of the lumped structure. Therefore, all structures are more efficient than the lumped structure.

5.6.2 Comparison of the Proposed Madaline Structure with Previous Structures in the Nonlinear Region of Stochastic Gain Function

Inputs to the stochastic gain function that are less than two result in a nonlinear behavior. To compare different structures in this region, the minimum NSR and neuron $\times$ NSR of the distributed, CVNS-RE, CVNS-DNN and CVNS-FDNN structures are compared with the maximum NSR and neuron $\times$ NSR of the proposed Madaline.
As discussed previously, the stochastic gain function of the lumped structure is always in the linear region. Therefore, the NSR and neuron $\times$ NSR of the lumped structure derived in the previous section will be used in this section as well. According to (5.8), (5.10), (5.12) and (5.14), the minimum NSR of the distributed, CVNS-RE, CVNS-DNN and CVNS-FDNN structures occurs when the stochastic gain is at its minimum. The minimum value of the stochastic gain is equal to one. Therefore, the minimum NSR of these structures can be written in the following form:

$$\text{NSR}_{\text{DM}} = \text{NSR}_{\text{CDM}} = \text{NSR}_{\text{CFDM}} = L \times \left( \frac{\sigma_x^2 \Delta x}{\sigma_x^2} + \frac{\sigma_w^2 \Delta w}{\sigma_w^2} \right)$$  (5.31)

$$\text{NSR}_{\text{CREM}} = L \sum_{i=1}^{L} (B^{-2D})^i \times \left( \frac{\sigma_x^2 \Delta x}{\sigma_x^2} + \frac{\sigma_w^2 \Delta w}{\sigma_w^2} \right)$$  (5.32)

According to (5.18), the maximum NSR of the proposed structure occurs when the stochastic gain is at its maximum. Considering that the maximum stochastic gain in this region is equal to 1.56, the upper bound of the NSR of the proposed structure can be written in the following form:

$$\text{NSR}_{\text{PM}} = B^{-2((D+1)L-1)} \times \sum_{i=1}^{L} (1.56)^i \times \left( \frac{\sigma_x^2 \Delta x}{\sigma_x^2} + \frac{\sigma_w^2 \Delta w}{\sigma_w^2} \right)$$  (5.33)

The NSR of different structures presented in (5.31) to (5.33) can be summarized as shown in Table 5.5. The normalized values of the NSR for various structures, with respect to the lumped structure, are shown in this Table as well.

Dividing the NSR of the proposed Madaline by the distributed, CVNS-DNN and CVNS-FDNN structures results in the following equation:

$$\frac{L^{-1} \times B^{-2((D+1)L-1)} \times \sum_{i=1}^{L} (1.56)^i}{(B^{-2D})^i} < 1$$  (5.34)

Thus, the NSR of the proposed structure in this region is lower than those of the distributed, CVNS-DNN and CVNS-FDNN structures.

To compare the NSR of the proposed structure with that of the CVNS-RE structure, their NSR is divided by each other, which results in the following equation:

$$B^{-2(D+1)L} \times \sum_{i=1}^{L} (1.56)^i \times \left( \sum_{i=1}^{L} (B^{-2D})^i \right)^{-1}$$  (5.35)

The maximum of (5.35) occurs when $B$, $D$ and $L$ are at their minimum value. A Madaline has at least three layers, while the minimum radix and number of CVNS digits is equal to two. Therefore, (5.30) is always less than one, and the proposed structure has a better NSR compared to the CVNS-RE structure. Moreover, considering the synaptic weight range of [-8,8] and input
Table 5.5: NSR of different structures and their normalized value with respect to lumped structure in nonlinear region of stochastic gain function

<table>
<thead>
<tr>
<th>Structure</th>
<th>NSR</th>
<th>Normalized NSR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lumped [19]</td>
<td>((\sqrt{N\sigma_x\sigma_w})^L \times (\frac{\sigma^2_x}{\sigma^2_z} + \frac{\sigma^2_w}{\sigma^2_z}))</td>
<td>1</td>
</tr>
<tr>
<td>Distributed [19]</td>
<td>(L \times (\frac{\sigma^2_x}{\sigma^2_z} + \frac{\sigma^2_w}{\sigma^2_z}))</td>
<td>(N^{-\frac{1}{2}} \times L \times (\sigma_x\sigma_w)^{-L})</td>
</tr>
<tr>
<td>CVNS-RE [16]</td>
<td>(\sum_{i=1}^{L} (B^{-2D})^{i} (\frac{\sigma^2_x}{\sigma^2_z} + \frac{\sigma^2_w}{\sigma^2_z}))</td>
<td>(N^{-\frac{1}{2}} \times \sum_{i=1}^{L} (B^{-2D})^{i} \times (\sigma_x\sigma_w)^{-L})</td>
</tr>
<tr>
<td>CVNS-DNN [17]</td>
<td>(L \times (\frac{\sigma^2_x}{\sigma^2_z} + \frac{\sigma^2_w}{\sigma^2_z}))</td>
<td>(N^{-\frac{1}{2}} \times L \times (\sigma_x\sigma_w)^{-L})</td>
</tr>
<tr>
<td>CVNS-FDNN [17]</td>
<td>(L \times (\frac{\sigma^2_x}{\sigma^2_z} + \frac{\sigma^2_w}{\sigma^2_z}))</td>
<td>(N^{-\frac{1}{2}} \times L \times (\sigma_x\sigma_w)^{-L})</td>
</tr>
<tr>
<td>Proposed</td>
<td>(B^{-2((D+1)L-1)\times} \sum_{i=1}^{L} (1.56)^i \times (\frac{\sigma^2_x}{\sigma^2_z} + \frac{\sigma^2_w}{\sigma^2_z}))</td>
<td>(N^{-\frac{1}{2}} \times B^{-2((D+1)L-1)} \times \sum_{i=1}^{L} (1.56)^i \times (\sigma_x\sigma_w)^{-L})</td>
</tr>
</tbody>
</table>

range of [0,1] or [-1,1], with a uniform distribution, the normalized NSR of the proposed structure is always lower than one. Consequently, the NSR of the proposed structure is lower than all previous structures in the nonlinear region of the stochastic gain function.

Using the normalized values of the total number of neurons and the NSR of different structures in Table 5.2 and Table 5.5, the results of the normalized neuron×NSR of all structures in the nonlinear region of the stochastic gain function are provided in Table 5.6.

To compare the efficiency of the proposed structure with the distributed and CVNS-DNN structures, neuron×NSR of the proposed structure is divided by neuron×NSR of the CVNS-DNN architecture which results in the following equation:

\[
L^{-1} \times \sum_{i=1}^{L} (1.56)^i \times (1 + (D + 1) - L^{-1}) \times B^{-2((D+1)L-1)}
\]  

(5.36)

The maximum of (5.36) occurs when \(B\), \(D\) and \(L\) are at their minimum value. Therefore, (5.36) is always less than one. Consequently, the proposed structure is more efficient compared to the distributed and CVNS-DNN structures. Considering that the neuron×NSR of the CVNS-FDNN architecture is higher than that of the distributed and CVNS-DNN structures, the proposed structure is more efficient compared to the CVNS-FDNN architecture as well.

To compare the efficiency of proposed structure with CVNS-RE, the neuron×NSR of the proposed
Table 5.6: Normalized neuron×NSR of different structures in nonlinear region of stochastic gain function

<table>
<thead>
<tr>
<th>Structure</th>
<th>Normalized Neuron×NSR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lumped [19]</td>
<td>1</td>
</tr>
<tr>
<td>Distributed [19]</td>
<td>$N^{1 - \frac{L}{2}} \times L \times (\sigma_x \sigma_w)^{-L}$</td>
</tr>
<tr>
<td>CVNS-RE [16]</td>
<td>$N^{1 - \frac{L}{2}} \times (D + 1) \times \sum_{i=1}^{L} (B^{-2D})^i \times (\sigma_x \sigma_w)^{-L}$</td>
</tr>
<tr>
<td>CVNS-DNN [17]</td>
<td>$N^{1 - \frac{L}{2}} \times L \times (\sigma_x \sigma_w)^{-L}$</td>
</tr>
<tr>
<td>CVNS-FDNN [17]</td>
<td>$N^{1 - \frac{L}{2}} \times (D + 1) \times L \times (\sigma_x \sigma_w)^{-L}$</td>
</tr>
<tr>
<td>Proposed</td>
<td>$N^{1 - \frac{L}{2}} \times \sum_{i=1}^{L} (1.56)^i \times (1 + (D + 1) - L^{-1}) \times B^{-2(D+1)L^{-1}} \times (\sigma_x \sigma_w)^{-L}$</td>
</tr>
</tbody>
</table>

structure is divided by CVNS-RE NSR, which results in the following equation:

$$\sum_{i=1}^{L} (1.56)^i \times ((D + 1)^{-1} + 1 - ((D + 1) \times L)^{-1}) \times B^{-2(D+1)L^{-1}} \times (\sum_{i=1}^{L} (B^{-2D})^i)^{-1}$$ (5.37)

The maximum of (5.37) occurs when B, D and L are at their minimum. Therefore, (5.37) is always less than one. This in turn means that the proposed structure is more efficient than the CVNS-RE structure.

The maximum value of neuron×NSR of the proposed structure occurs when B, D and L are at their minimum. Considering synaptic weight range of [-8, 8] and input range of [0, 1] or [-1, 1], with a uniform distribution, neuron×NSR of the proposed structure is always less than one. Therefore, the proposed structure is more efficient compared to the lumped structure. Thus, the proposed structure is the most efficient structure compared to the previous structures in the nonlinear region of the stochastic gain function.

5.7 VLSI Implementation and Comparisons

In this section, the VLSI implementation of a three-layer Madaline with $N = 5$ Adalines in each layer using 8 bits for weight storage and inputs and weights uniformly distributed in the range of $[-1, 1]$ and $[-8, 8]$ is considered. The input and weight variance is $\sigma_x^2 = \frac{1}{3}$ and $\sigma_w^2 = \frac{8^2}{3}$, respectively. The radix of the CVNS is considered to be two, which provides the most efficient radix for conversion between binary and CVNS. The number of CVNS digits used in each layer of the previous CVNS
structures, $D$, is equal to three. Since the Madaline has three layers, $L$ is equal to two. Therefore, using (5.19), the number of CVNS digits used in the output layer of the proposed CVNS-distributed Madaline, $D_L$, is equal to six.

The neuron $\times$ NSR calculation was used in the previous sections to compare the efficiency of different structures, while the area $\times$ NSR calculation is used in this section. The neuron $\times$ NSR made the system-level efficiency evaluation of the different structures feasible, while the network area $\times$ NSR provides more information regarding the area consumption, corresponding to a specific NSR, at the circuit-level. Therefore, to measure the VLSI implementation efficiency of different Madaline structures, their area consumption and NSR were considered.

The basic building blocks of an Adaline are the adder, multiplier, neuron and the weight storage medium. The proposed and the previous structures are implemented by current-mode circuits. Addition in the current-mode circuits is easily performed through wiring the nodes carrying the signals. Thus, the addition overhead in all structures can be neglected.

The multiplier used in the lumped and distributed Adaline [19] is a Multiplying Digital to Analog Converter (MDAC), while the proposed Adaline and the previous CVNS structures require CVNS multiplier.

The lumped, distributed and the proposed Adalines exploit digital registers as the weight storage medium, while all previous CVNS structures require analog memory. The digital registers can be implemented using the TSMC 0.18$\mu$m CMOS standard cell library.

The VLSI implementation of the CVNS multiplier is realized based on the (5.15). The block diagram of an 8-bit CVNS multiplier is shown in Fig. 5.8. The CVNS multiplier is implemented using current-mode circuits. In the implemented circuit, 8 $\mu$A is indicator of 1. As can be seen in Fig. 5.8, the input currents $(x)_0$ to $(x)_7$ are applied to the transistors M1 to M8. These transistors act as switches which turn on, provided that the corresponding $w_i$ input to their gate is high. This in turn implements the $w_i(x)_7$ terms required to calculate the output $(z)$. Transistors with a $W/L$ of $(0.22 \mu m/0.18 \mu m)$ are used for implementing these switches. The input currents, after passing the input transistors, are summed at the input nodes of the mod 16 $\mu$A blocks. The output of the mod 16 $\mu$A blocks, which receive the input currents, are wired together to perform the addition. The addition result passes through two more stages of the mod 16 $\mu$A blocks. This in turn generates the multiplication result.

The main building block of the CVNS multiplier is the mod16$\mu$A block. The mod16$\mu$A opera-
The current comparator and subtractor sections should compare the input current to the mod 16μA function.

The mod 16μA operation circuit is shown in Fig. 5.9. The transistor sizes of the mod 16μA circuit are shown in Table 5.7. It should be noted that the smallest actual size of \( \frac{W}{L} \) used for VLSI implementation of the Madaline is equal to \( \frac{0.22}{0.18} \) μm.

The circuit is composed of five main sections including an input current mirror, a current comparator, a current subtractor, an inverter chain and an output current mirror.

According to (5.38), the current comparator and subtractor sections should compare the input current to the mod 16μA function.

\[
x \mod 16 \mu A = \begin{cases} 
x & x < 16 \mu A \\
x - 16 \mu A & x \geq 16 \mu A
\end{cases}
\]  

(5.38)

where \( x \) is the input to the mod 16μA function.
current $x$ with 16 $\mu$A and subtract the $x$ from 16 $\mu$A, provided that the input is greater than 16 $\mu$A. The input current mirror, which is composed of the transistors $M_1$ and $M_2$, copies the input current to the current comparator and the current subtractor sections. The current comparator is based on the structure developed in [24] and compares the input current with a reference current of 16 $\mu$A. This 16 $\mu$A is generated by the transistors $M_4$ and $M_5$ and copied to the transistor $M_6$. The output of the comparator is connected to the inverter chain which provides a rail to rail output. The current comparator, along with the inverter chain, generates the CmpN and CmpP signals equal to 0 and 1.8 volt provided that the input current to the mod 16 $\mu$A circuit is greater than 16 $\mu$A. The two transistors $M_8$ and $M_9$, which act as a transmission gate, turn on provided that the input current is greater than 16 $\mu$A. This allows the reference current generated by the transistors $M_4$ and $M_5$ be copied to the transistor $M_7$. Considering that the current through the transistor $M_3$ is the same as the input current, the 16 $\mu$A reference current will be subtracted from the input current which flows through the transistor $M_{10}$. This is only applicable if the input current is greater than 16 $\mu$A. Otherwise, the input current flows through the transistor $M_{10}$. The current through the transistor $M_{10}$ is copied to the transistor $M_{11}$, which generates the output of the mod 16 $\mu$A block.
Table 5.7: Transistor sizes of the mod16µA circuit

<table>
<thead>
<tr>
<th>Transistor</th>
<th>(W/L) (µm/µm)</th>
<th>Transistor</th>
<th>(W/L) (µm/µm)</th>
<th>Transistor</th>
<th>(W/L) (µm/µm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>M₁</td>
<td>(0.105)</td>
<td>M₇</td>
<td>(0.660.5)</td>
<td>M₁₃</td>
<td>(0.220.18)</td>
</tr>
<tr>
<td>M₂</td>
<td>(0.105)</td>
<td>M₈</td>
<td>(0.660.18)</td>
<td>M₁₄</td>
<td>(0.90.18)</td>
</tr>
<tr>
<td>M₃</td>
<td>(0.105)</td>
<td>M₉</td>
<td>(0.10.5)</td>
<td>M₁₅</td>
<td>(0.220.18)</td>
</tr>
<tr>
<td>M₄</td>
<td>(0.650.18)</td>
<td>M₁₀</td>
<td>(0.10.5)</td>
<td>M₁₆</td>
<td>(0.90.18)</td>
</tr>
<tr>
<td>M₅</td>
<td>(0.660.5)</td>
<td>M₁₁</td>
<td>(0.10.5)</td>
<td>M₁₇</td>
<td>(0.220.18)</td>
</tr>
<tr>
<td>M₆</td>
<td>(0.660.5)</td>
<td>M₁₂</td>
<td>(0.90.18)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

![Figure 5.10: Simulation results of the mod16µA circuit](image)

Simulation results of four different input values are shown in Fig. 5.10 which confirm the proper operation of the mod 16 µA circuit.

VLSI implementation of the neuron is shown in Fig. 5.11a. The implemented neuron is based on the structure developed in [25], which realizes the sigmoid activation function. The output versus input of the implemented neuron is shown in Fig. 5.11b. Furthermore, the transistor sizes of the implemented neuron are shown in Table 5.8.
The last building block of the proposed Adaline is the RE unit. This process reduces the error in the CVNS digits and can be shown in the following form [15]:

$$\langle x \rangle_{nc} = \lfloor \langle x \rangle_n - \frac{\langle x \rangle_{n-1}}{2} \rfloor + \frac{\langle x \rangle_{n-1}}{2}$$

(5.39)

where $\langle x \rangle_n$ and $\langle x \rangle_{n-1}$ are two adjacent CVNS digits and $\langle x \rangle_{nc}$ is the corrected CVNS digit. Since the radix used for the implementation is two, $\langle x \rangle_n - \frac{\langle x \rangle_{n-1}}{2}$ will always be less than two. Therefore, $\lfloor \langle x \rangle_n - \frac{\langle x \rangle_{n-1}}{2} \rfloor$ can be written in the following form:

$$\lfloor \langle x \rangle_n - \frac{\langle x \rangle_{n-1}}{2} \rfloor = \begin{cases} 0 & \langle x \rangle_n - \frac{\langle x \rangle_{n-1}}{2} < 1 \\ 1 & \langle x \rangle_n - \frac{\langle x \rangle_{n-1}}{2} \geq 1 \end{cases}$$

(5.40)

Considering that $8 \, \mu\text{A}$ is indicator of one, (5.40) can be rewritten in the following form:
Figure 5.12: VLSI implementation of the RE unit

\[
\lfloor (\langle x \rangle_n - \langle x \rangle_{n-1})/2 \rfloor = \begin{cases} 
0 & \langle x \rangle_n - \langle x \rangle_{n-1} < 8 \mu A \\
8 \mu A & \langle x \rangle_n - \langle x \rangle_{n-1} \geq 8 \mu A 
\end{cases} \tag{5.41}
\]

VLSI implementation of the RE unit is shown in Fig. 5.12. The transistor sizes of the implemented circuit are shown in Table 5.9. The implemented circuit is composed of four main sections including a current subtractor, a current comparator, an inverter chain and an output generation. The transistors M1 and M2 form a current mirror which copies the input \( \langle x \rangle_{n-1} \) to the current
### Table 5.9: Transistor sizes of the RE circuit

<table>
<thead>
<tr>
<th>Transistor</th>
<th>( \frac{W}{L} ) (( \mu \text{m} ))</th>
<th>Transistor</th>
<th>( \frac{W}{L} ) (( \mu \text{m} ))</th>
<th>Transistor</th>
<th>( \frac{W}{L} ) (( \mu \text{m} ))</th>
</tr>
</thead>
<tbody>
<tr>
<td>M(_1)</td>
<td>( \frac{1}{0.5} )</td>
<td>M(_9)</td>
<td>( \frac{1}{0.5} )</td>
<td>M(_{17})</td>
<td>( \frac{0.3}{0.5} )</td>
</tr>
<tr>
<td>M(_2)</td>
<td>( \frac{0.5}{0.5} )</td>
<td>M(_{10})</td>
<td>( \frac{1}{0.5} )</td>
<td>M(_{18})</td>
<td>( \frac{0.3}{0.5} )</td>
</tr>
<tr>
<td>M(_3)</td>
<td>( \frac{1}{0.5} )</td>
<td>M(_{11})</td>
<td>( \frac{1}{0.5} )</td>
<td>M(_{19})</td>
<td>( \frac{1}{0.5} )</td>
</tr>
<tr>
<td>M(_4)</td>
<td>( \frac{1}{0.5} )</td>
<td>M(_{12})</td>
<td>( \frac{0.6}{0.2} )</td>
<td>M(_{20})</td>
<td>( \frac{1}{0.5} )</td>
</tr>
<tr>
<td>M(_5)</td>
<td>( \frac{1}{0.5} )</td>
<td>M(_{13})</td>
<td>( \frac{0.25}{0.2} )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M(_6)</td>
<td>( \frac{1}{0.5} )</td>
<td>M(_{14})</td>
<td>( \frac{0.6}{0.2} )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M(_7)</td>
<td>( \frac{1}{0.5} )</td>
<td>M(_{15})</td>
<td>( \frac{0.25}{0.2} )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>M(_8)</td>
<td>( \frac{1}{0.5} )</td>
<td>M(_{16})</td>
<td>( \frac{1}{0.5} )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

subtractor circuit. Considering that the \( \frac{W}{L} \) of the transistor M\(_2\) is half of the transistor M\(_1\), \( \frac{(x)_{n-1}}{2} \) is copied to the current subtractor circuit. The transistors M\(_4\) and M\(_5\) constitute a current mirror which copies the input \( (x)_{n} \) to the current subtractor circuit. The transistors M\(_5\) to M\(_7\) act as a current subtractor that produces the \( (x)_{n} - \frac{(x)_{n-1}}{2} \) which is required for the calculation of (5.41). This is used as the input to the current comparator circuit formed by the transistors M\(_8\) to M\(_{11}\). This compares the input with a current reference of 8 \( \mu \text{A} \) generated by the transistors M\(_8\) and M\(_9\). The output of the current comparator is connected to the inverter chain constituted by the transistors M\(_{12}\) to M\(_{15}\). The inverter chain generates a rail to rail voltage, which indicates whether the \( (x)_{n} - \frac{(x)_{n-1}}{2} \) is greater than the 8 \( \mu \text{A} \) or not. The output of the inverter chain is connected to the CmpN input of the transistor M\(_{19}\). The transistors M\(_{16}\) to M\(_{19}\) generate an 8 \( \mu \text{A} \) current, provided that the CmpN input to the transistor M\(_{19}\) is high. This in turn implements the \[ (x)_{nc} = [(x)_{n} - \frac{(x)_{n-1}}{2}] + \frac{(x)_{n-1}}{2} \] required for implementation of the RE unit. The size of the transistor M\(_{20}\) is the same as the transistor M\(_3\). Therefore, the current \( \frac{(x)_{n-1}}{2} \) is copied to this transistor. The output of the transistors M\(_{19}\) and M\(_{20}\) are wired together which performs the addition. Hence, the \( (x)_{nc} = [(x)_{n} - \frac{(x)_{n-1}}{2}] + \frac{(x)_{n-1}}{2} \) is generated at the output of the RE circuit. The simulation results of four different inputs to the RE circuit are shown in Fig. 5.13. This confirms the proper operation of the implemented RE circuit.

Using the CVNS multiplier, the neuron, the RE and the digital register circuits, the proposed
5. AREA-EFFICIENT ROBUST MADALINE BASED ON CONTINUOUS VALUED NUMBER SYSTEM

Adaline is laid out. The layout of the proposed Adaline is shown in Fig. 5.14.

The area consumption of various cells required for different structures is summarized in Table 5.10. As can be seen from Table 5.10, the area requirement of digital register is drastically lower than that of the analog memory. This may result in area reduction of the proposed Madaline compared to all of the previous CVNS structures.
Table 5.10: Area consumption of different cells required for implementation of different structures

<table>
<thead>
<tr>
<th>Cell Type</th>
<th>Area ($\mu m^2$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>8-bit MDAC Multiplier [21]</td>
<td>529</td>
</tr>
<tr>
<td>8-bit CVNS Multiplier</td>
<td>407.73</td>
</tr>
<tr>
<td>Neuron [18]</td>
<td>62.04</td>
</tr>
<tr>
<td>Digital Register</td>
<td>53.22</td>
</tr>
<tr>
<td>Analog Memory [18]</td>
<td>713.18</td>
</tr>
</tbody>
</table>

The area consumption of different structures is calculated by the summation of the area consumption of the different cells required for the implementation of that structure. Therefore, the total number of different cells required for the previous Adalines, as well as the proposed Adaline, is determined.

The total number of multipliers required for the lumped and distributed Adalines is equal to the number of inputs to the Adaline. However, the number of multipliers required for previous CVNS Adalines, as well as the proposed Adaline, is equal to product of the number of inputs multiplied by the number of CVNS digits.

In terms of the number of required neurons, the lumped Adaline requires one neuron, while distributed and CVNS-DNN Adalines require one neuron for each input. On the other hand, the CVNS-RE, CVNS-FDNN and the proposed Adaline require one neuron corresponding to each multiplier. Thus, the total number of neurons in these Adalines is same as the number of multipliers.

Regarding the weight storage, all previous Adalines require weight storage medium as an input to each multiplier. Therefore, the number of weight storage elements is determined by the product of the total number of multipliers multiplied by the precision of the weights. In the proposed Adaline, all multipliers corresponding to different CVNS digits representing the same input, share the same weight storage element. Thus, the total number of weights in the proposed Adaline is equal to product of the number of inputs multiplied by the precision of the weights. The total number of cells required for different Adalines is summarized in Table 5.11. Here, $N_{i-1}$ and $N_w$ are the number of inputs to the Adaline and the precision of the weight storage medium, respectively, while $D + 1$ and $D_L + 1$ are the number of CVNS digits in the corresponding Adalines.
Table 5.11: Total number of different cells required for implementation of Adalines used in different structures

<table>
<thead>
<tr>
<th>Adaline Type</th>
<th>Multiplier</th>
<th>Neuron</th>
<th>Weight Storage Medium</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lumped [19]</td>
<td>MDAC</td>
<td>$N_{i-1}$</td>
<td>register</td>
</tr>
<tr>
<td>Distributed [19]</td>
<td>MDAC</td>
<td>$N_{i-1}$</td>
<td>register</td>
</tr>
<tr>
<td>CVNS-RE [16]</td>
<td>CVNS</td>
<td>$N_{i-1} \times (D + 1)$</td>
<td>analog</td>
</tr>
<tr>
<td>CVNS-DNN [17]</td>
<td>CVNS</td>
<td>$N_{i-1} \times (D + 1)$</td>
<td>analog</td>
</tr>
<tr>
<td>CVNS-FDNN [17]</td>
<td>CVNS</td>
<td>$N_{i-1} \times (D + 1)$</td>
<td>analog</td>
</tr>
<tr>
<td>Proposed Adaline</td>
<td>CVNS</td>
<td>$N_{i-1} \times (D_L + 1)$</td>
<td>register</td>
</tr>
</tbody>
</table>

As can be seen from Table 5.11, the number of weight storage elements in the proposed Adaline is independent of the number of CVNS digits. Therefore, the number of weight storage elements in the proposed Madaline will be independent of the number of CVNS digits as well. This, in combination with the low area consumption requirement of digital registers, may result in significant reduction in the area consumption of the proposed Madaline.

It should be noted that for the Madalines implemented here, $N_{i-1}$, $N_w$ and $D+1$ are equal to 5, 8 and 3, while $D_L$ is equal to 6. Therefore, the proposed Adaline requires 30 CVNS multipliers, while the previous CVNS Adalines require 15 CVNS multipliers. The lumped and distributed Adalines require 5 multipliers.

Concerning the total number of neurons, the proposed Adaline requires 30 neurons, while the CVNS-RE and CVNS-FDNN Adalines each require 15 neurons. The CVNS-DNN Adaline, similar to the distributed Adaline, requires 5 neurons, and the lumped Adaline requires 1 neuron.

In regards to the weight storage elements, the proposed Adaline, as well as the lumped and distributed Adalines, require 40 digital registers, while all of the previous CVNS Adalines require 120 Analog memory units. Therefore, the proposed Adaline requires more multipliers and neurons compared to the previous CVNS Adalines, while requiring less weight storage elements.

Using the number of different cells required for each Adaline, along with their area consumption summarized in Table 5.10, the area consumption of different Adaline structures is estimated in Table 5.12. The results summarized in this table show that the proposed Adaline area requirement is higher than the lumped and distributed structures, while it shows 82% improvement when compared to the
CVNS-DNN structure, which is the most area efficient structure among all of the previous CVNS Adalines. To estimate the area consumption of different Madalines using the information provided in Table 5.12, the total number of Adalines required for each Adaline should be determined. In the three-layer Madaline implemented here, 5 Adalines are used in both the hidden layer as well as the output layer. The previously developed structures use 10 Adalines of the same type, while the proposed CVNS-distributed structure uses 5 distributed Adalines in the hidden layer and 5 proposed CVNS distributed Adalines in the output layer. Considering the number of Adalines used in different structures, and using the area consumption of different Adalines summarized in Table 5.12, the area consumption of different Madaline structures is calculated and summarized in Table 5.13. Equations (5.6), (5.8), (5.10), (5.12), (5.14) and (5.18) are used to calculate the NSR of the various Madaline structures. The results are summarized in Table 5.13. Furthermore, the area×NSR is listed for comparison.

The results summarized in Table 5.13 show that the proposed structure area requirement is drastically lower than that of the previous CVNS structures, specifically, it demonstrates an 88% improvement when compared to the CVNS-DNN, which is the most area efficient of the previous CVNS Madalines. The reduction in the area consumption of the proposed Madaline structure is a result of the decrease in the number of weight storage elements, as well as one advantage of exploiting the digital weight storage in the proposed architecture. In terms of the NSR and area×NSR, the CVNS-RE structure is the most efficient among all of the previously developed structures. Compared to the CVNS-RE structure, the proposed architecture provides 30.14 dB improvement in terms of NSR, while in terms of area×NSR, the proposed structure shows 47.74 dB improvement. Therefore,
Table 5.13: Area consumption, NSR and area×NSR of different Madaline structures investigated in this case study

<table>
<thead>
<tr>
<th>Madaline Structure</th>
<th>Number of Adalines</th>
<th>Area (µm²)</th>
<th>NSR (dB)</th>
<th>Area×NSR (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lumped</td>
<td>10</td>
<td>48,358.4</td>
<td>-71.69</td>
<td>21.99</td>
</tr>
<tr>
<td>Distributed</td>
<td>10</td>
<td>50,840.0</td>
<td>-88.67</td>
<td>5.44</td>
</tr>
<tr>
<td>CVNS-RE</td>
<td>10</td>
<td>926,281.5</td>
<td>-118.74</td>
<td>0.59</td>
</tr>
<tr>
<td>CVNS-DNN</td>
<td>10</td>
<td>920,077.5</td>
<td>-90.31</td>
<td>28.96</td>
</tr>
<tr>
<td>CVNS-FDNN</td>
<td>10</td>
<td>926,281.5</td>
<td>-90.31</td>
<td>29.02</td>
</tr>
<tr>
<td>Proposed Madaline</td>
<td>10</td>
<td>106,529.5</td>
<td>-148.88</td>
<td>-48.33</td>
</tr>
</tbody>
</table>

the proposed structure is more tolerant to input and weight errors, and requires less area for a specific NSR compared to the previous CVNS structures.

5.8 Conclusion

A new mixed-signal CVNS Adaline structure is proposed in this paper. This structure stores the weights in digital registers while the arithmetic is based on the CVNS. In addition, the RE process is used to decrease the error in the CVNS digits, which improves the NSR. Storing the weights in digital registers provides a reliable and low complexity storage mechanism, at the same time eliminating the need for the complex analog memory units, which are sensitive to process and power supply variations, required for the implementation of the previous CVNS architectures. Moreover, the proposed structure requires a lower number of weight storage elements compared to the previous CVNS structures, this in turn results in a lower area overhead and reduced power consumption.

Combining the proposed Adaline with the distributed structure, the CVNS-distributed Madaline structure is proposed in this paper. The mathematical analysis of the NSR and neuron×NSR of the proposed Madaline structure is performed, and the comparison with the previous architectures is conducted. The results show that the proposed Madaline structure compares favorably to all previous architectures in terms of NSR and neuron×NSR. Furthermore, to have a circuit-level analysis, a three-layer Madaline is implemented. The implementation results proves that the proposed structure improves upon the previous structures in terms of the NSR and the area consumption required for
a specific NSR.

The lower NSR provided by the proposed network decreases the effect of unavoidable weight quantization error present in the hardware implementation, leading to a more robust architecture. The increased efficiency of the proposed network is an indicator of a lower total number of neurons required for a specific NSR. This in turn results in a robust neural network with a lower area overhead and reduced power consumption.
5.9 References


Chapter 6

Efficient VLSI Implementation of Neural Networks with Hyperbolic Tangent Activation Function

6.1 Introduction

Neural networks have a wide range of applications in analog and digital signal processing. Hardware implementation of neural networks has been used in applications such as pattern recognition [1], optical character recognition [2], test of analog circuits [3], real-time surface discrimination [4], smart sensing [5] and identification of heavy ions [6].

The main building blocks needed for hardware implementation of neural networks are multiplier, adder and nonlinear activation function. A lot of research has been done in digital implementation of multipliers and adders which can be readily used leaving the nonlinear activation function as the most complex building block.

To implement the neuron, various nonlinear activation functions such as threshold, sigmoid and hyperbolic tangent can be used. Hyperbolic tangent and sigmoid are mostly used because their differentiable nature makes them compatible with back propagation algorithm. Both activation functions have s-shaped curve while their output range is different. Because of the exponentiation and division terms present in sigmoid and hyperbolic tangent activation function, it is hard to realize
the hardware implementation of these functions directly.

To solve the implementation problem, approximation methods are generally applied. These methods are based on Piecewise Linear approximation (PWL), piecewise nonlinear approximation, Lookup Table (LUT), bit-level mapping and hybrid methods. Generally, in PWL approximation methods, the function is divided into segments and linear approximation is used in each segment. This method is used in [7], [8] and [9] for the hyperbolic tangent and sigmoid function implementation. Another PWL approximation method is introduced in [10] and [11]. Unlike other PWL methods, the developed method is not based on input domain segmentation and exploits lattice algebra based Centred Recursive Interpolation (CRI) algorithm.

The piecewise nonlinear approximation is similar to PWL method with the difference that nonlinear approximation is used in each segment. This method is used in [12] to approximate the sigmoid function and scheme 4 of [9] is proposed for approximating both sigmoid and hyperbolic tangent.

In the LUT based methods, input range is divided to equal sub-ranges and each sub-range is approximated by a value stored in LUT. This method is used in [13] to implement the hyperbolic tangent.

Bit-level mapping method approximates output based on a direct bit-level mapping of input. This method can be implemented using purely combinational circuits and is used in [14] to implement the sigmoid function.

Hybrid methods use a combination of the aforementioned methods. Examples include [15] and [16] which have used a combination of PWL and LUT methods for hyperbolic tangent activation function implementation.

The approximation error present in all methods affects the neural network performance. The study performed in [17] and [18] shows that the nonlinear activation function implementation with higher accuracy improves the learning and generalization capabilities of neural networks. However, implementations with higher accuracy require more silicon area and decrease the network operation speed. Therefore, having nonlinear activation function hardware structures with lower area and higher speed for a specified accuracy becomes a key issue.

In general, PWL and piecewise nonlinear approximation methods need multiplications while the LUT and bit-level mapping methods use no multipliers. An exception is the CRI based method which is a PWL approximation while it requires no multipliers. However, it requires large number of registers [10]. The choice of approximation method depends on the target implementation technology. The hardware implementation of neural networks is mostly done in FPGA or ASIC. Because the current FPGAs provide a large number of multipliers, PWL and piecewise nonlinear
approximation based methods are appropriate selection for FPGA implementation. However, due to high area requirements and delay of the multipliers in ASIC implementation, LUTs and bit-level mapping are more suitable.

The focus of this paper is on ASIC implementation of hyperbolic tangent function. Hyperbolic tangent has an output range of [-1,1] and is defined as follows:

\[
\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}
\]  

(6.1)

In the method proposed by Lin and Wang [8], PWL is used to approximate the first derivative of hyperbolic tangent. Then, the first derivative approximation is integrated to obtain the hyperbolic tangent function. Lebouf et.al [13] have proposed a new LUT-based structure for the hyperbolic tangent activation function. The proposed structure is based on Range Addressable Lookup Table (RALUT) [19]. A hybrid architecture is proposed by Namin et.al [15] which uses a simple PWL approximation in combination with RALUT. Another hybrid structure is proposed by Meher [16] which is based on a linear approximation in combination with LUT. The values stored in LUT are determined by the proposed boundary selection method.

In this paper a new hybrid architecture which is based on linear approximation in combination with bit-level mapping is proposed. The proposed architecture takes into account maximum allowable error as the design parameter.

The proposed approximation scheme divides the input range to three different regions using different strategy in each region. A mathematical analysis of the proposed approximation scheme in each region is provided.

The mathematical analysis shows that the proposed scheme requires less number of output bits for the same maximum error compared to the previous architectures. The hardware implementation of the proposed structure is realized in CMOS 0.18 µm to show the efficiency of proposed structure in terms of area, delay and product of area and delay compared to the previous architectures.

The proposed structure is used for implementing a 4-3-2 network in CMOS 0.18 µm. Post layout simulation results show that the proposed structure results in a neural network implementation with lower area, delay and power.

The rest of this paper is organized as follows. In the next section, the proposed approximation scheme is discussed. The mathematical analysis for selection of minimum number of input and output bits is provided in section 6.3. The domain boundaries of different regions are found in section 6.4. The proposed structure based on the mathematical analysis done is explained in section 6.5. Hardware implementation of the hyperbolic tangent function and comparison with existing
structures is done in section 6.4. Neural network implementation using the proposed structure is discussed in section 6.7. Finally, conclusions are drawn in section 6.8.

6.2 Proposed Approximation Scheme

In this section, mathematical analysis of approximation scheme used for hardware implementation of hyperbolic tangent function is provided. The mathematical analysis in this section and the following sections uses the basic properties of hyperbolic tangent function.

Hyperbolic tangent is an odd function.

\[
\tanh(-x) = -\tanh(x) \tag{6.2}
\]

Using this property, only the absolute value of input is processed and the input sign is directly passed to the output.

The Taylor series expansion of hyperbolic tangent is as follows:

\[
\tanh(x) = x - \frac{x^3}{3} + \frac{2x^5}{15} - \frac{17x^7}{315} + ... \tag{6.3}
\]

For small values of \(x\) the higher order terms become small and can be ignored. Therefore, the hyperbolic tangent passes the small input values to output.

\[
\lim_{x \to 0} \tanh(x) = x \tag{6.4}
\]

The output variation for large values of input is low.

\[
\frac{d\tanh(x)}{dx} = 0 \quad x \to \infty \tag{6.5}
\]

Considering the two last properties, input range is divided to three regions. Region I in which the output is approximately equal to input is named pass region while because of low variation of output in region III it is named saturation region. Region II includes the rest of input range, named processing region. Determining the boundary of each region is discussed later in section 6.4. Different regions of hyperbolic tangent function are shown in Fig. 6.1.

6.2.1 Output Approximation in the Pass Region

The input and output of hyperbolic tangent function are represented as signed-magnitude notation. Therefore, considering the first basic property of hyperbolic tangent function discussed in previous
section, input sign bit is directly passed to the output sign bit and only the absolute value of input is processed. Absolute value of input, in binary format can be represented based on the following equation:

\[
x = \sum_{k=-N_f}^{N_i-1} x_k \times 2^k = \sum_{k=0}^{N_i-1} x_k \times 2^k + \sum_{k=-N_f}^{-1} x_k \times 2^k
\]  

(6.6)

where \(N_i\) and \(N_f\) are the number of bits for integer and fractional part of the input and \(x_k\) is binary digit and can assume values 0 or 1.

In the pass region, output is approximated by passing the input to the output which means that a linear approximation is used in this region. The inputs in the pass region include the values close to the origin which are represented by fractional part of the input.

The absolute value of hyperbolic tangent function output is in the range of \([0,1]\) and can be shown as follows:

\[
\tanh(x) = \sum_{k=-N_{out}}^{-1} y_k \times 2^k = \sum_{k=-N_f}^{-1} y_k \times 2^k + \sum_{k=-N_{out}}^{-(N_f+1)} y_k \times 2^k
\]  

(6.7)

where \(N_{out}\) is the number of bits used for representation of absolute value of output and \(y_k\) can assume one of the values 0 or 1.

Using (6.6) and (6.7), \(y_k\) is obtained as follows:
6. EFFICIENT VLSI IMPLEMENTATION OF NEURAL NETWORKS WITH HYPERBOLIC TANGENT ACTIVATION FUNCTION

\[ y_k = \begin{cases} 
    x_k & -N_f \leq k \leq -1 \\
    0 & -N_{out} \leq k < -N_f 
\end{cases} \tag{6.8} \]

Based on (6.8), the fractional part of input is shifted to left by \( N_{out} - N_f \) bits and then is passed to the output.

### 6.2.2 Output Approximation in the Processing Region

Before going through the proposed approximation scheme in this region, a new parameter named \( \text{N}_{\text{one}} \) is introduced. This parameter is an indicator of position of the first occurrence of one in binary input, when scanned from left. Therefore, based on this parameter, the input range can be shown as follows:

\[ 2^{N_{\text{one}}} \leq x < 2^{N_{\text{one}}+1} \tag{6.9} \]

This input range is divided into equal sub-ranges. The number of these sub-ranges, \( N \), is based on the equation shown as follows:

\[ N = 2^{(N_{\text{one}} + N_f - i)} \quad 0 \leq i \leq N_{\text{one}} + N_f \]

Based on the value of \( N \), sub-ranges within the input range are as follows:

\[ 2^{N_{\text{one}}} \left(1 + \frac{j}{N}\right) \leq x < 2^{N_{\text{one}}} \left(1 + \frac{j + 1}{N}\right) \quad 0 \leq j < N \tag{6.10} \]

To have an approximation value close to all outputs corresponding to an input sub-range, the average value of outputs is considered as the approximation value as follows:

\[ \frac{\sum_{k=0}^{2^{N_{\text{one}}} + N_f} \tanh \left(2^{N_{\text{one}}} \left(1 + \frac{j}{N}\right) + k \times 2^{-N_f}\right)}{2^{N_{\text{one}} + N_f}} \tag{6.11} \]

The total number of sub-ranges is found based on the fact that the difference between all values inside a sub-range and the approximation value found using (6.11) should be less than maximum allowable approximation error in this region, which results in the following equation:

\[ \left| \tanh \left(2^{N_{\text{one}}} \left(1 + \frac{j}{N}\right)\right) - \frac{\sum_{k=0}^{2^{N_{\text{one}}} + N_f} \tanh \left(2^{N_{\text{one}}} \left(1 + \frac{j}{N}\right) + k \times 2^{-N_f}\right)}{2^{N_{\text{one}} + N_f}} \right| < \epsilon_a \]

\[ 0 \leq j < N \tag{6.12} \]
6. EFFICIENT VLSI IMPLEMENTATION OF NEURAL NETWORKS WITH HYPERBOLIC TANGENT ACTIVATION FUNCTION

where $\epsilon_a$ is the maximum allowable approximation error in the processing region. $\epsilon_a$ depends on the maximum allowable error described in section 6.3.

6.2.3 Output Approximation in the Saturation Region

The hyperbolic tangent function reaches its maximum value in the saturation region, while at the same time output variation in this region is low. Therefore, all output values in this region are approximated by the maximum value representable by the output bits. Using (6.7) this value is equal to $1 - 2^{-N_{out}}$.

6.3 Selection of Number of Input and Output Bits

In this section a mathematical analysis is presented, which allows for optimal finding of the number of input and output bits, required for hardware implementation of the proposed approximation scheme.

6.3.1 Selection of Number of Input Bits

The representation of absolute value of input in binary format can be shown using (6.6). The number of bits needed for integer part depends on the input range. Therefore, to cover the range it is required to have:

$$2^{N_{i}} \geq r_{i} \quad (6.13)$$

where $r_{i}$ is the input range. Using (6.13), $N_{i}$ can be written in the following form:

$$N_{i} \geq \left\lceil \frac{\ln r_{i}}{\ln 2} \right\rceil \quad (6.14)$$

in which $\lceil \cdot \rceil$ is the ceiling function which rounds its input towards the next highest integer.

In comparison the number of bits used for fractional part is determined by the maximum allowable error. It should be noted that input region between two consecutive points $x_{1}$ and $x_{2}$ can be approximated as $\tanh(x_{1})$ having an error lower than maximum allowable error provided that the following equation is satisfied.

$$\tanh(x_{2}) - \tanh(x_{1}) \leq \epsilon \quad (6.15)$$

where $\epsilon$ is the maximum allowable error.

The hyperbolic tangent change between two consecutive inputs is proportional to the hyperbolic tangent derivative shown in Fig 6.2. Therefore, the maximum change of hyperbolic tangent function
between two consecutive points occurs in the region which is close to the origin. Based on (6.4), hyperbolic tangent output is approximately equal to its input in this region and therefore (6.15) can be simplified as follows:

$$x_2 - x_1 \leq \epsilon$$

(6.16)

The difference between two consecutive points in the input is determined by the number of bits used for representing the fractional part of the input and is equal to $2^{-N_f}$. Thus, (6.16) can be written as follows:

$$2^{-N_f} \leq \epsilon$$

(6.17)

which results in the following equation:

$$N_f \geq \lceil -\frac{\ln \epsilon}{\ln 2} \rceil$$

(6.18)

using (6.14) and (6.18) it can be written:

$$N_{inp} = \lceil \frac{\ln r_i}{\ln 2} \rceil + \lceil -\frac{\ln \epsilon}{\ln 2} \rceil$$

(6.19)

where $N_{inp}$ is the minimum number of input bits required for representation of absolute value of input.

### 6.3.2 Selection of Number of Output Bits

As previously discussed, the hyperbolic tangent function is divided into three regions including pass, processing and saturation region. The number of bits required for output representation in these
three regions, assuming a maximum allowable error of $\epsilon$, depends on the properties of each region as will be discussed.

**Pass Region**

In the pass region, input is passed to the output. The pass region is where the inputs are close to the origin. These points are represented by fractional part of the input. Therefore, minimum number of bits required in this region is equal to:

$$N_{out} \geq N_f$$

(6.20)

**Processing Region**

The output error in the processing region is composed of two elements. The first one is the approximation error while the second one is the quantization error of representing the approximated output. The total error caused by these sources should be less than maximum allowable error which is shown as follows:

$$\epsilon_a + \epsilon_q \leq \epsilon$$

(6.21)

where $\epsilon_a$ is the maximum allowable approximation error and $\epsilon_q$ is the maximum quantization error of representing the approximated output.

The quantization error, $\epsilon_q$, is proportional to the number of bits used for output representation. If rounding method is used for quantization of output, maximum quantization error is going to be equal to half of Low Significant Bit (LSB) which is equal to $2^{-(N_{out}+1)}$. Therefore, the maximum allowable approximation error can be obtained as follows:

$$\epsilon_a = \epsilon - 2^{-(N_{out}+1)}$$

(6.22)

It should be noted that the maximum allowable approximation error found through (6.22) was used in (6.12) in order to find the number of sub-ranges inside the processing region. Hence the change in number of output bits changes the number of sub-ranges in this region.

**Saturation Region**

The approximated value of output in this region is equal to $1 - 2^{-N_{out}}$. This represents the maximum value of hyperbolic tangent function with an error which is less than the maximum allowable error. This results in the following equation:

$$\left| 2^{-N_{out}} \right| \leq \epsilon$$

(6.23)
which can be written in the following form:

\[ N_{\text{out}} \geq \left\lceil \frac{\ln \epsilon}{\ln 2} \right\rceil \]  \hspace{1cm} (6.24)

By comparing (6.24) and (6.18), minimum number of bits needed for output representation in the saturation region is equal to \( N_f \).

The minimum number of bits needed for output representation in the pass and saturation region is equal to \( N_f \) while there is no condition in the processing region. Therefore, the minimum number of bits needed for output representation of absolute value of hyperbolic tangent function using the proposed approximation scheme is equal to \( N_f \).

The number of output bits required in the proposed approximation scheme is lower than the number of input bits while all previously developed architectures use the same number of input and output bits. Reduction in number of output bits may result in efficient hardware implementation. This will be investigated more in the next sections.

6.4 Determining the Boundaries for Different Regions

In this section using the maximum allowable error as design parameter, boundaries of each region is determined.

6.4.1 Pass Region

Based on the Taylor series expansion of hyperbolic tangent function for small values of \( x \), higher order terms can be ignored. Therefore, the first three terms shown below are sufficient to present hyperbolic tangent function.

\[ \tanh(x) \sim x - \frac{x^3}{3} + \frac{2x^5}{15} \]  \hspace{1cm} (6.25)

Since in the pass region input is passed to the output, boundary of pass region, \( x_{pa} \), can be found using the following equation:

\[ \left| \frac{x_{pa}^3}{3} - \frac{2x_{pa}^5}{15} \right| \leq \epsilon \]  \hspace{1cm} (6.26)

The \( x_{pa} \) obtained should be rounded to the nearest lower value representable by the input bits. Therefore, the quantized value of \( x_{pa} \) can be written in the following form:

\[ x_{pa_q} = \left\lfloor \frac{x_{pa} \times 2^{N_{\text{inp}}}}{r_i} \right\rfloor \frac{2^{N_{\text{inp}}}}{r_i} \]  \hspace{1cm} (6.27)

where \( \lfloor \cdot \rfloor \) is the floor function and \( 0 \leq x \leq x_{pa_q} \) is considered as the pass region.
6. EFFICIENT VLSI IMPLEMENTATION OF NEURAL NETWORKS WITH HYPERBOLIC TANGENT ACTIVATION FUNCTION

6.4.2 Saturation Region

The starting point of the saturation region is when the difference between hyperbolic tangent function and its approximation becomes equal to maximum allowable error. Therefore, the starting point of saturation region, $x_s$, is found as follows:

$$x_s = \tanh^{-1} \left( 1 - 2^{-N_{out}} - \epsilon \right) = \frac{1}{2} \ln \left( \frac{2 - 2^{-N_{out}} - \epsilon}{2^{-N_{out}} + \epsilon} \right)$$

(6.28)

The $x_s$ obtained should be rounded to the nearest higher value representable by input bits. Therefore, the quantized value of $x_s$ can be written in the following form:

$$x_{s_q} = \left\lceil x_s \times \frac{2^{N_{inp}}}{r_3} \right\rceil \frac{2^{N_{inp}}}{r_3}$$

(6.29)

where $x \geq x_{s_q}$ is considered as the saturation region.

6.4.3 Processing Region

The region between pass and saturation region is considered as processing region which can be shown as follows:

$$x_{pa_q} < x_{pr} < x_{s_q}$$

(6.30)

where $x_{pr}$ is an input in the processing region.

6.5 Proposed Structure

Block diagram of the proposed structure is shown in Fig. 6.3. The hardware is composed of two main blocks including hyperbolic tangent approximation and output assignment.

6.5.1 Hyperbolic Tangent Approximation

This block is composed of three main blocks to approximate the hyperbolic tangent function in all three regions including saturation, processing and pass region. General arithmetic operations in each region can be described as follows:

Pass Region

In this region fractional part of input is passed to the output. Based on (6.8), a shift to left by $N_{out} - N_f$ bits before passing the input to output is required.
Figure 6.3: Block diagram of the proposed structure

**Processing Region**

For inputs in the processing region, a bit-level input mapping is required. The number of bit-level mapping blocks required is equal to the number of input ranges in this region. For each input range in the processing region, \( \log_2 N \) bits after \( N_{one} \) bit of input should be mapped to output bits using the bit-level mapping. Using \( \log_2 N \) bits after \( N_{one} \) bit covers all sub-ranges. The number of sub-ranges \( N \) is calculated using (6.12) while the output of each sub-range is found using (6.11). The bit-level mapping can be implemented using a combinational circuit.

**Saturation Region Approximation**

In this region hyperbolic tangent function is approximated by the maximum value representable by output bits, and can be realized by setting all output bits to one.
6.5.2 Output Assignment

The input range decoder detects the $N_{\text{one}}$ introduced previously which is set by input range and region of operation respectively. Depending on the input range, a multiplexer is used to obtain the appropriate output value.

To illustrate the proposed approximation scheme and structure, an example is presented. The example shows different steps of the design procedure.

Example: Design procedure for $\epsilon = 0.04$ considering an input range of (-8,8)

1) Determining the number of input and output bits: using (6.19) we have $N_{\text{inp}} = 8$. The minimum number of output bits required is equal to $N_f$. Using (6.18) we have $N_{\text{out}} = 5$.

2) Determining the boundaries of pass, processing and saturation regions: Using (6.26), (6.27), (6.28), (6.29) and (6.30) the pass, saturation and processing region boundaries are found equal to $x_{pa} = 0.5$, $x_{sa} = 1.65625$ and $0.5 < x_{pr} < 1.65625$.

3) Output assignment in pass region: In this region, the fractional part of input is shifted to left by $N_{\text{out}} - N_f$ bits and passed to the output. Therefore, no shift is required in this example and the fractional part of input is directly passed to the output.

4) Output assignment in saturation region: In saturation region, the output value is equal to $1 - 2^{-5}$ or 0.96875.

5) Output assignment in processing region: First the maximum allowable approximation error is found using (6.22) which is equal to 0.024. Then, the number of sub-ranges, $N$, is found using (6.12). Finally, sub-ranges are found using (6.10) and the appropriate value of each sub-range is assigned using (6.11).

The Table 6.1 summarizes these values for different input ranges and sub-ranges.

Also, quantization error, approximation error and total error for the considered case are shown in figures Fig. 6.4a to Fig. 6.4c. The approximated output and ideal output are shown in Fig. 6.4d.

6) Designing the proposed structure: As can be seen from Table 6.1, there are five different input ranges in the considered example. The input range is detected by the input range decoder.

The input range decoder detects the input range using $N_{\text{one}}$. Table 6.2 shows the input range decoder truth table which can be implemented using a fully combinational circuit.

The input range $0 < x < 0.5$ ($r_5$) is in the pass region. Considering that $N_{\text{out}}$ is equal to $N_f$ no shift is required and the hyperbolic tangent is approximated by passing the fractional part of input to output directly.

The input ranges $0.5 < x < 1$ and $1 < x < 2$ ($r_4$ and $r_3$) are in the processing region. Therefore, for hyperbolic tangent approximation a bit-level mapping is required in these input ranges.
6. EFFICIENT VLSI IMPLEMENTATION OF NEURAL NETWORKS WITH HYPERBOLIC TANGENT ACTIVATION FUNCTION

Table 6.1: Output value for different input ranges and sub-ranges

<table>
<thead>
<tr>
<th>Input Range</th>
<th>Input Sub-Range</th>
<th>Output Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_{i1}$</td>
<td>$x_{i2}$</td>
<td>$x_{s1}$</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>—</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>—</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>1.875</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.875</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.75</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.375</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.25</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.125</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>0.5</td>
<td>1</td>
<td>0.9375</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.875</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.8125</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.75</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.6875</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.625</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.5625</td>
</tr>
<tr>
<td>0</td>
<td>0.5</td>
<td>—</td>
</tr>
</tbody>
</table>

In the input range $1 < x < 2$, the $N_{one}$ is equal to 1 and the number of sub-ranges ($N$) is equal to 8. Therefore, a bit-level mapping on the $\log_2^8 = 3$ bits after the $N_{one}$ bit of the input is required to generate the output. Table 6.3 shows the required bit-level mapping.
For input range $0.5 < x < 1$, the $N_{\text{one}}$ is equal to 0 and the number of sub-ranges ($N$) is equal to 8. Therefore, a bit-level mapping on the $\log_2 8 = 3$ bits after the $N_{\text{one}}$ bit of the input is required to generate the output. Table 6.4 shows the required bit-level mapping.

These tables can be implemented using a purely combinational circuit.

In the input ranges $2 < x < 4$ and $4 < x < 8$ ($r_2$ and $r_1$), input is in the saturation region and hyperbolic tangent is approximated by setting all output bits to 1.

The multiplexer assigns the output value using the input range decoder. For each input range the multiplexer transfers the approximation in that range to the output.

The hardware implementation of the considered example is shown in Fig. 6.5. It should be noted
6. EFFICIENT VLSI IMPLEMENTATION OF NEURAL NETWORKS WITH HYPERBOLIC TANGENT ACTIVATION FUNCTION

Table 6.2: Input range decoder

<table>
<thead>
<tr>
<th>Input Range</th>
<th>$x_{i1}$</th>
<th>$x_{i2}$</th>
<th>$x_2$</th>
<th>$x_1$</th>
<th>$x_0$</th>
<th>$x_{-1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_1$</td>
<td>4</td>
<td>8</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>$r_2$</td>
<td>2</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>$r_3$</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>$r_4$</td>
<td>0.5</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$r_5$</td>
<td>0</td>
<td>0.5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

that the bit-level mapping needed for input ranges $r_3$ and $r_4$ is implemented using Tables 6.3 and 6.4 while the input range decoder is implemented based on Table 6.2.

6.6 Hardware Implementation of the Hyperbolic Tangent Function and Comparison with Existing Structures

The proposed structure in the previous section is implemented with maximum allowable errors of 0.02 and 0.04. The proposed structure is coded using Verilog hardware description language and synthesized by Synopsys Design Compiler using TSMC 0.18 $\mu$m library.

Meher [16] has synthesized his proposed structure using TSMC 90 nm library while the synthesis of all other previously developed architectures is done using 0.18 $\mu$m library. To have a fair comparison between all architectures, we have coded the design of Meher [16] in Verilog and synthesized using TSMC 0.18 $\mu$m library. It should be noted that signed-magnitude notation is used for input and output representation.

The comparison of different structures for $\epsilon = 0.04$ and $\epsilon = 0.02$ is summarized in Tables 6.5 and 6.6. These tables include the input range, number of input bits, number of output bits, maximum error after design and synthesis results which are area and delay. The maximum error after design is evaluated for $10^6$ points uniformly distributed in the input range [12]. Also, to have a comparison of number of cells required for different designs, the gate count measure which is the design area normalized with respect to two input NAND gate area is included. Moreover, considering that both area and delay are important in hardware design the area×delay is included in these tables too.

The method used by Lin and Wang [8] is based on PWL approximation which requires multipli-
The other three previously proposed architectures use LUT. In [13] two LUT based structures are proposed to implement the hyperbolic tangent function. In the first structure 512 and 1024 points for errors of $\epsilon = 0.04$ and $\epsilon = 0.02$ are stored in LUT. The second structure is based on RALUT. Using RALUT, number of stored points for errors of $\epsilon = 0.04$ and $\epsilon = 0.02$ is reduced to 61 and 127. The reduction in number of stored points reduces the area consumption.

In the architecture proposed by Namin et.al [15], a simple PWL approximation in combination with RALUT is used. The RALUT stores the difference between PWL approximation and the hyperbolic tangent function which results in a reduction in number of stored points compared to the RALUT used in [13]. This reduction lowers the area required while because of the subtraction present in the proposed architecture, it has more delay.

Meher [16] has proposed an optimized LUT based architecture. The proposed architecture is based on linear approximation in combination with LUT and requires 7 and 15 stored points for
Table 6.3: Bit-level mapping for the input range $1 < x < 2$

<table>
<thead>
<tr>
<th>$x_0$</th>
<th>$x_{-1}$</th>
<th>$x_{-2}$</th>
<th>$y_{-1}$</th>
<th>$y_{-2}$</th>
<th>$y_{-3}$</th>
<th>$y_{-4}$</th>
<th>$y_{-5}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 6.4: Bit-level mapping for the input range $0.5 < x < 1$

<table>
<thead>
<tr>
<th>$x_{-1}$</th>
<th>$x_{-2}$</th>
<th>$x_{-3}$</th>
<th>$y_{-1}$</th>
<th>$y_{-2}$</th>
<th>$y_{-3}$</th>
<th>$y_{-4}$</th>
<th>$y_{-5}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

errors of $\epsilon = 0.04$ and $\epsilon = 0.02$. The reduction in the number of stored points reduces the area required for hardware implementation.

Our proposed structure is based on a linear approximation in combination with bit-level mapping which removes the need to store points and can be implemented using a purely combinational circuit.
Table 6.5: Comparison of different structures for $\epsilon=0.04$

<table>
<thead>
<tr>
<th>Structure</th>
<th>Input Range</th>
<th>Input</th>
<th>Output</th>
<th>Maximum Error</th>
<th>Area ($\mu m^2$)</th>
<th>Gate Count</th>
<th>Delay (ns)</th>
<th>Area $\times$ Delay ($\mu m^2 \times$ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scheme-1 [8]</td>
<td>(-8,8)</td>
<td>24</td>
<td>24</td>
<td>0.0430</td>
<td>32069.83</td>
<td>3214</td>
<td>903</td>
<td>2.896 $\times$10^7</td>
</tr>
<tr>
<td>LUT [13]</td>
<td>(-8,8)</td>
<td>8</td>
<td>8</td>
<td>0.0365</td>
<td>9045.94</td>
<td>907</td>
<td>2.15</td>
<td>1.944 $\times$10^4</td>
</tr>
<tr>
<td>RALUT [13]</td>
<td>(-8,8)</td>
<td>8</td>
<td>8</td>
<td>0.0357</td>
<td>7090.40</td>
<td>711</td>
<td>1.85</td>
<td>1.311 $\times$10^4</td>
</tr>
<tr>
<td>Hybrid [15]</td>
<td>(-8,8)</td>
<td>8</td>
<td>8</td>
<td>0.0361</td>
<td>3646.83</td>
<td>366</td>
<td>2.31</td>
<td>8.424 $\times$10^3</td>
</tr>
<tr>
<td>Optimized LUT</td>
<td>(-8,8)</td>
<td>8</td>
<td>8</td>
<td>0.0401</td>
<td>954.67</td>
<td>96</td>
<td>2.09</td>
<td>1.995 $\times$10^3</td>
</tr>
<tr>
<td>Proposed</td>
<td>(-8,8)</td>
<td>8</td>
<td>5</td>
<td>0.0378</td>
<td>695.22</td>
<td>70</td>
<td>0.95</td>
<td>6.604 $\times$10^2</td>
</tr>
</tbody>
</table>

Also, the number of output bits required in the proposed structure is lower than all previously proposed architectures which reduces the area consumption.

On the other hand, the simple input range decoding method which only uses the position of the first occurrence of one in binary input in combination with bit-level mapping provides a high speed structure. This is confirmed by synthesis results shown in Tables 6.5 and 6.6 which show that the proposed structure in both cases compares favorably to the previously proposed structures in terms of area, delay and area $\times$ delay.

### 6.7 Neural Network Implementation Using the Proposed Structure for Hyperbolic Tangent Activation Function

General configuration of a Madaline is shown in Fig. 6.6. The Madaline has $L+1$ layers with $N_0$ inputs, and $N_i$ Adalines in each layer $i$. Multiplication and addition are the arithmetic operations required in neural network implementation.

The output of activation function should be multiplied by weights. Therefore, the multiplier size, $S_m$, in hidden layers of neural network can be written in the following form:

$$ S_m = N_w \times N_{out} \quad (6.31) $$
Table 6.6: Comparison of different structures for $\epsilon$=0.02

<table>
<thead>
<tr>
<th>Structure</th>
<th>Input Range</th>
<th>$N_{inp}$</th>
<th>$N_{out}$</th>
<th>Maximum Error</th>
<th>Area ($\mu m^2$)</th>
<th>Gate Count</th>
<th>Delay (ns)</th>
<th>Area $\times$ Delay ($\mu m^2 \times$ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scheme-1 [8]</td>
<td>(-8,8)</td>
<td>24</td>
<td>24</td>
<td>0.0220</td>
<td>83559.17</td>
<td>8374</td>
<td>1293</td>
<td>$1.080 \times 10^8$</td>
</tr>
<tr>
<td>LUT [13]</td>
<td>(-8,8)</td>
<td>9</td>
<td>9</td>
<td>0.0180</td>
<td>17864.24</td>
<td>1791</td>
<td>2.45</td>
<td>$4.377 \times 10^4$</td>
</tr>
<tr>
<td>RALUT [13]</td>
<td>(-8,8)</td>
<td>9</td>
<td>9</td>
<td>0.0178</td>
<td>11871.53</td>
<td>1190</td>
<td>2.12</td>
<td>$2.517 \times 10^4$</td>
</tr>
<tr>
<td>Hybrid [15]</td>
<td>(-8,8)</td>
<td>9</td>
<td>9</td>
<td>0.0189</td>
<td>5130.78</td>
<td>515</td>
<td>2.80</td>
<td>$1.437 \times 10^4$</td>
</tr>
<tr>
<td>Optimized LUT [16]</td>
<td>(-8,8)</td>
<td>9</td>
<td>9</td>
<td>0.0265</td>
<td>1603.32</td>
<td>161</td>
<td>2.82</td>
<td>$4.521 \times 10^3$</td>
</tr>
<tr>
<td>Proposed</td>
<td>(-8,8)</td>
<td>9</td>
<td>6</td>
<td>0.0196</td>
<td>1280.66</td>
<td>129</td>
<td>2.12</td>
<td>$2.714 \times 10^3$</td>
</tr>
</tbody>
</table>

where $N_w$ is the number of bits used for synaptic weight storage while $N_{out}$ is the number of output bits of the activation function.

The multiplication results of synapses connected to an Adaline in the next layer should be added before passing through the activation function of that layer. Considering (6.31), the number of output bits of multipliers is equal to $N_w + N_{out}$. Therefore the size of adders, $S_a$, required for addition of multiplication results of $N_i$ synapses between layer $i$ and $i+1$ can be written in the following form:

$$S_a = (N_w + N_{out}) \times N_i$$  \hspace{1cm} (6.32)

Therefore, the size of multipliers and adders in the hidden layers of neural network depend on the number of output bits of the activation function.

For a specific maximum allowable error, the proposed structure requires less number of output bits compared to the previously developed architectures. Therefore, bit width of multipliers and adders in the hidden layers of the network using proposed structure as its activation function is lower. Multipliers and adders with lower bit width have lower area, delay and power consumption. Therefore, using proposed structure results in efficient VLSI implementation of neural networks with hyperbolic tangent activation function.

To evaluate the efficiency of proposed structure, it is used to implement a 4-3-2 network for an optical template matching application. The general neural network block diagram is shown in Fig.
6. EFFICIENT VLSI IMPLEMENTATION OF NEURAL NETWORKS WITH HYPERBOLIC TANGENT ACTIVATION FUNCTION

6.7. It is capable of recognizing six different input patterns and classifying them as four different classes. The optical input patterns and their related class is shown in Fig. 6.8.

The network is coded using Verilog hardware description language and synthesized by Synopsys Design Compiler using TSMC 0.18 $\mu m$ library. The signal processing in the implemented network is based on fixed point arithmetic and the proposed structure implemented in the previous section which had maximum allowable errors of $\epsilon = 0.02$ and $\epsilon = 0.04$ is used as activation function. The network training is done off-chip and the calculated weights are stored on the registers inside the chip.

The same network is also coded and synthesized using the structure proposed by Meher [16] with maximum allowable errors of $\epsilon = 0.02$ and $\epsilon = 0.04$.

Post layout simulation results show that the implemented network using the proposed structure and the one proposed by Meher [16] for both cases of $\epsilon = 0.02$ and $\epsilon = 0.04$ performs well.

The post layout simulation results of the implemented network are summarized in Table 6.7.
6. EFFICIENT VLSI IMPLEMENTATION OF NEURAL NETWORKS WITH HYPERBOLIC TANGENT ACTIVATION FUNCTION

Figure 6.8: Optical input patterns and their related class

<table>
<thead>
<tr>
<th>Template</th>
<th>Direction</th>
<th>Equivalent Bits (Input)</th>
<th>Class (Output)</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Diagram]</td>
<td>Horizontal</td>
<td>1100</td>
<td>01</td>
</tr>
<tr>
<td>[Diagram]</td>
<td>Vertical</td>
<td>0101</td>
<td>00</td>
</tr>
<tr>
<td>[Diagram]</td>
<td>Horizontal</td>
<td>0011</td>
<td>01</td>
</tr>
<tr>
<td>[Diagram]</td>
<td>Vertical</td>
<td>1010</td>
<td>00</td>
</tr>
<tr>
<td>[Diagram]</td>
<td>Left Diagonal</td>
<td>1001</td>
<td>10</td>
</tr>
<tr>
<td>[Diagram]</td>
<td>Right Diagonal</td>
<td>0110</td>
<td>11</td>
</tr>
</tbody>
</table>

which show that the implementation of the network using proposed structure results in efficient VLSI implementation in terms of area, delay and power for both maximum allowable errors of $\epsilon = 0.02$ and $\epsilon = 0.04$.

6.8 Conclusion

A new approximation scheme for hyperbolic tangent is proposed in this paper. The proposed approximation scheme is based on a mathematical analysis considering maximum allowable error as design parameter.

Based on the proposed approximation scheme, a hybrid architecture for hardware implementation of hyperbolic tangent activation function is presented. The synthesis results show that the proposed structure compares favorably to the previously developed architectures in terms of area, delay and area $\times$ delay.

The proposed structure requires less number of output bits for the same maximum allowable error compared to the previously developed architectures. Reduction in number of activation function
output bits results in multipliers and adders with lower bit width which in turn reduces the area, power and delay in VLSI implementation of neural networks. The proposed structure is used for implementing a 4-3-2 network which is capable of recognizing six different input patterns. Post layout simulation results show that the proposed structure results in an efficient neural network VLSI implementation in terms of area, delay and power.
6.9 References


REFERENCES


Chapter 7

Conclusions and Future Work

7.1 Conclusions

CVNS addition, CVNS sigmoid function evaluation and CVNS multiplication algorithms for low-resolution environment are introduced. These algorithms make the implementation of high resolution mixed-signal neural networks in a low-resolution environment feasible.

The proposed CVNS function evaluation method exploits the PWL approximation scheme and is based on a mathematical derivation using the CVNS characteristics. A new CVNS-based structure for realization of the proposed CVNS sigmoid function evaluation scheme is proposed. The proposed structure uses the mixed-signal current-mode circuits. The implementation results show that the proposed structure compares favorably to the state of the art.

The proposed CVNS multiplication algorithm provides accurate results in low-resolution environment. Moreover, VLSI implementation of a 16×8 CVNS synapse multiplier is realized. The post-layout simulations of the implemented CVNS synapse multiplier confirms its performance.

Using the proposed CVNS algorithms, a 2-2-1 mixed-signal CVNS neural network structure is implemented. In the implemented network, weights are stored in the digital registers with 16-bit resolution. The signal processing of the network is carried out using CVNS arithmetic. The implemented network realizes the two input XOR function. Using the CVNS features, the limited analog signal processing resolution issue present in mixed-signal neural networks is resolved. This opens a path to design mixed-signal structures which meet the signal processing resolution requirements.
of neural networks. The implemented network is designed, laid out, and post-layout simulated in 0.18µm CMOS technology.

An area-efficient robust CVNS Adaline is also proposed. This structure stores the weights in digital registers while the arithmetic is based on the CVNS. In addition, the Reverse Evolution (RE) process is exploited to decrease the error in the CVNS digits, which improves the NSR. Storing the weights in digital registers eliminates the need for the complex analog memory units, required for the implementation of the previous CVNS architectures. Furthermore, the proposed structure needs less number of weight storage elements compared to the previous CVNS structures. This in turn results in a lower area overhead and reduced power consumption.

Combining the proposed Adaline with the distributed architecture, the CVNS-distributed Madaline structure is introduced. The mathematical analysis shows that the proposed Madaline structure compares favorably to all previous architectures in terms of NSR and required number of neurons for a specific NSR. In addition, to have a circuit-level analysis, a three-layer Madaline is implemented. The implementation results confirms that the the proposed structure improves upon the previous structures in terms of the NSR and the area consumption required for a specific NSR. This in turn leads to a robust neural network with a lower area overhead and reduced power consumption.

A new approximation scheme for digital implementation of hyperbolic tangent is proposed in final part of this work. The proposed approximation scheme is based on a mathematical analysis considering maximum allowable approximation error as design parameter. Based on the proposed approximation scheme, a hybrid structure for VLSI implementation of hyperbolic tangent activation function is presented. The synthesis results show that the proposed architecture compares favorably to the previously developed structures in terms of area, delay and area × delay.

The proposed structure requires lower number of output bits for the same maximum allowable approximation error compared to the previously developed architectures. Reduction in number of activation function output bits leads to multipliers and adders with lower bit width. This in turn lowers the area, power and delay. The proposed structure is used for implementing a 4-3-2 network. The implemented network is capable of recognizing six different input patterns. Post layout simulation results prove that the proposed structure results in an efficient VLSI implementation of digital neural networks in terms of area, delay and power consumption.
7. CONCLUSIONS AND FUTURE WORK

7.2 Future work

The proposed mixed-signal CVNS structure satisfies the resolution requirements of neural networks with on-chip learning. Therefore, it can be exploited to develop a network with on-chip learning capability. To realize this, additional modules are required. These modules should implement the learning algorithm on the chip. For example, for the back-propagation algorithm, the derivative of activation function is suggested to be designed similar to the structure proposed for the CVNS sigmoid activation function. In addition, multipliers and adders are required to update the weights and biases during the learning. These units can be realized using the CVNS multiplier and the CVNS adder structures proposed in this dissertation. Moreover, we suggest to apply low power design techniques to the whole system to decrease the network power consumption.
Babak Zamanlooy received the B.S. degree from the K. N. Toosi University of Technology, Tehran, Iran, and the M.S. degree from the Iran University of Science and Technology, Tehran, in 2004 and 2006, respectively, both in electrical engineering. He is currently pursuing the Ph.D. degree in electrical engineering with the University of Windsor, Windsor, ON, Canada. His current research interests include analog, digital, and mixed-signal integrated circuit design and VLSI implementation of neural networks.