Waseda University Doctoral Dissertation

A Study on Hardware Design for High Performance Artificial Neural Network by using FPGA and NoC

DONG, Yiping

Graduate School of Information, Production and Systems
Waseda University

July 2011

Abstract

Artificial neural networks (ANNs) are widely used in various applications such as recognition, security, computer learning and so on. To meet requirements of higher performance, hardware implementations have been widely researched and developed. The popular implementation methods are FPGA, analog, digital and hybrid methods. The FPGA method is widely used due to the low cost and short design time, but at the same time it is limited by the performance compared with other hardware implementation methods. The digital method is also widely used due to high precision, good expansibility and a good design support by EDA tools, but at the same time it is limited by the higher design cost, heavy communication load and less reconfigurability. In this thesis, we formulate and address problems in these two key hardware implementation methods, namely, FPGA ANN and digital ANN.

The first problem is the performance problem limiting the development of FPGA ANN. We presented a novel architecture for the FPGA ANN, which integrates both the layer-multiplexing and pipeline architecture. This proposed architecture could achieve high performance by enhancing the efficiency of resource usage of the same FPGA board. Our proposed architecture presents an advantage in two basic respects over the traditional implementations. The first one is the hybrid of layer multiplexing and pipeline, which can optimize both the resource requirement and speed. The layer multiplexing guarantees the resource required by neural network under the constraint of an adopted FPGA chip, and the pipelining between the layers can improve the speed. The second point is the algorithm to determine the optimal hardware architecture according to the neural network parameters such as the topology, data structure and so on. Furthermore, our method just meets the resource limitation of a given FPGA, so that the FPGA board does not need to be changed for another application.

The second is a problem in digital ANN which is limited by the higher design cost, heavy communication load and less reconfigurability. Recently, Network on Chip (NoC) has attracted much attention. The packet-based network with high level parallelism architecture of NoC was used to solve complex on-chip interconnection problems for large system-on-chip (SoC). We presented a digital ANN with NoC architecture to solve the existing problems of digital ANN, such as heavy communication load and less reconfigurable. This digital ANN with NoC architecture is reconfigurable, because the weight values and activation functions can be changed as desired. We can also change the topology and routing algorithms of the NoC by sending new data to meet different kinds of ANN, so this system is easily extended. We can design this system in the style of cell-by-cell and

can easily add or remove any cell to comply with different applications. The proposed NoC system can reduce the communication load of total packet size and improve the system performance of connection-per-second (CPS). This proposed NoC mapping method can make the digital ANN more efficient.

The third is a discussion on general digital ANN with NoC architecture which is limited by the design cost when implementing large size ANN. A multiple NoC model is developed for a digital ANN, which can implement both a small size ANN and a large size one. Model-1 uses the general NoC ANN, all the layers of ANN can be implemented with it in one time, thus it can be suitable for ANN with small network size. Model-2 uses the same NoC architecture, whereas the implementation method is different. In this model, different layers of ANN will be implemented with NoC architecture one by one, so that it is appropriate for ANN with large network size. The proposed multiple NoC models can reduce communication load, increase system performance of CPS, and reduce system running time compared with the existing hardware ANN. Furthermore, this architecture is reconfigurable and reparable. It can be used to implement different applications of ANN.

As the fourth issue, routing is important in order to maximize effectiveness of NoC. The traditional routing strategy limits the communication load and performance of the NoC ANN. One of popular routing strategies is Destination-Tag (DT) method which is used in the proposed NoC ANN. The advantage is that each hop could be easily controlled and different routing algorithms could be easily realized, whereas the disadvantage is that the total destination address stored in header becomes larger and larger proportional to the network size. This drawback causes that the NoC ANN could not achieve high performance and low communication load for large size ANN. Thus, a new NoC architecture is needed to implement the ANN. The main improvement is a router model with absolute address based routing strategy instead of the former router with DT method based routing strategy. This absolute address based routing strategy could reduce the header size of the packet compared with the DT method, and it can implement different routing algorithms with a little hardware change. So that the absolute address based NoC architecture is effective in reducing communication load and increasing performance.

Finally, an on-chip neural processor is implemented to get high performance. Based on our study so far, the NoC architecture is described to structure a new type of neural processor named NoCNN and designed using 90-nm CMOS technology. The NOCNN is composed of 20 tiles in a 4x5 2-D array, and each tile includes a Process Element (PE) and a packet-switched router. It can achieve over 3.1 Giga CPS of computing speed while power dissipation is 1.1317 W at 1.2 V supply, and its chip size is 25 mm^2. It can work asynchronously in different tiles and work in parallel in the same tile to make the system computing speed of CPS higher. It is reconfigurable, because weight value, activation function and implementation information can be easily changed by sending new packets. It is also expandable, because the tiles can be easily added or removed. The packet transmission method of NoC is more efficient and smart than a general digital ANN, thus it can reduce communication load.

In conclusion, the performance problem of FPGA-based ANN is solved and it could suit for the real applications which need high performance. Furthermore, NoC architecture is proposed to instead of traditional bus-based P2P hardware ANN to solve the problems of performance, communication load and reconfigurability.

Acknowledgements

Studying towards Ph.D. at Graduate School of Information, Production and Systems, Waseda university takes three years. It is a long process filled with mixed feelings, satisfaction and disappointment, pleasure and pressure. The satisfaction comes often for the recognition of the research and work while the disappointment always for the rejections and misapprehension. The pleasure often from innovative ideas of the interesting research area while the pressure may come from the deadline of the task. And in this three years, I have learned that - I could never have done the research and writing this thesis without the support and encouragement of the following people.

First, Professor Takahiro Watanabe of Waseda Univ. is the one I thank most. I thank him far more just because of the fact that he is my supervisor. You've been my father, my friend, my confidant, my colleague, and my collaborator. You are very respectable for your personality, knowledge and creativity. And you teach me not just be a researcher, but also how to be a man. On my life road, you guide me forward. I am lucky to be one of your students.

I am also thankful for Prof. Takeshi Shima and Prof. Chen of Kanagawa Univ. who gave me some valuable comments and revised my thesis.

I am also very grateful to Prof. Shinji Kimura and Prof. Takeshi Yoshimura of Waseda Univ. who revised all the thesis and gave a lot of valuable comments. Also thanks for their constructive advise about my research at the associated seminar.

I am also thankful for Prof. Xuefeng Yan, you are my abecedarian, and you gave the first knowledge for computer and mathematics.

I am indebted to Dr. Song Chen, Dr. Zhangcai Huang and Dr. Yun Yang for their constructive and fruitful discussions of my research.

I would also like to express my thanks to my NoC research group Xiujun bai, Kiyohisa Sato, Yang Wang, Zhen Lin, Yan Li, Hua Zhang, Hui Liu, Yifei Tang, et al. I have learned much from all of you, and the discursion with you gave me a lot of help for my research.

I would also thanks my classmates in Watanabe Lab Ce Li, Zhi Li, Shuo Li, Xiukun Xu, et al., my friends in other Lab of Waseda University Chunqi Jin, Shaoyong Luan, et al., and many other students and faculty, too numerous to name. Not just I can discuss my research with you, but also I can discuss my troubles with.

The research presented in the dissertation is financed by Waseda University Global COE Program 'International Research and Education Center for Ambient SoC' sponsored by MEXT, Japan (GCOE), and also partially supported by Regional Innovation Cluster Program 2nd Stage by MEXT, and Core Research for Evolutional Science and Technology (CREST), Japan Science and Technology.

Finally I give my deepest gratitude to my family, my parents and my wife: Yangang Dong, Qinhua Wu and Lan Wang. Without your support, unending and love, I would never through this process or any of the tough times in my life. Any piece of my achievement has an invisible part of their contribution. Thank you.

Kitakyushu, Japan Yiping Dong
May 5, 2011

Table of Contents

Abstract **i**

Acknowledgements **v**

Table of Contents **vii**

List of Tables **xi**

List of Figures **xiv**

Glossary **xv**

Notations xv

Abbreviations xv

1 Introduction **1**

1.1 Historical review of ANN 1

1.2 Organization of This Thesis 3

2 Preliminaries **7**

2.1 Basic Knowledge of ANN 7

2.2 Basic Knowledge of Network on Chip (NoC) 12

2.2.1 System on Chip (SoC) Challenges 12

2.2.2 Network on Chip (NoC) Platform 13

2.2.3 Network on Chip (NoC) Architecture 15

3 A Hybrid Layer-multiplexing and Pipeline Architecture for Efficient FPGA-based Multilayer Neural Network **21**

3.1 Introduction 21

3.2 Neuron Design 23

3.2.1 Mathematical Model 23

3.2.2 Neuron Circuit Design 23

3.3 Network Design 24

3.3.1 Pipeline Design . . . 25
3.3.2 Layer Multiplexing with Partly Pipeline . . . 27
3.3.3 The Control Block Design . . . 28
3.4 Experiments . . . 29
3.5 Conclusion . . . 31

4 A New Flexible Network on Chip Architecture for Mapping Complex Feedforward Neural Network 33
4.1 Introduction . . . 33
4.2 Proposed NoC Mapping System . . . 34
4.2.1 One neuron architecture . . . 35
4.2.2 Four neurons in one PE . . . 37
4.2.3 Router design and packet architecture . . . 38
4.2.4 System design . . . 40
4.3 Examination of Proposed System . . . 42
4.4 Evaluation . . . 43
4.4.1 System reconfigurability . . . 44
4.4.2 System extensibility . . . 44
4.4.3 Reduced communication load . . . 44
4.4.4 Performance measured by connections per second (CPS) . . . 45
4.4.5 Power consumption . . . 46
4.5 Conclusions and Future Work . . . 47

5 Multiple Network-on-Chip Model for High Performance Neural Network 49
5.1 Introduction . . . 49
5.2 Architecture of Multiple NoC models Implement for FF-ANN . . . 51
5.2.1 Structure of a single neuron . . . 51
5.2.2 Four neurons aggregated in one Processing Element (PE) . . . 52
5.2.3 5-port 2-virtual channel router architecture . . . 52
5.2.4 Design for multiple NoC models . . . 53
5.3 Experiment for Support Low Power NoC Design Method . . . 59
5.4 Evaluation and Discussion . . . 61
5.4.1 Reconfigurability of the proposed system . . . 61
5.4.2 Reparability of the proposed system . . . 61
5.4.3 Communication load reduction . . . 61
5.4.4 High performance of connection-per-second (CPS) . . . 63
5.5 Conclusions . . . 65

6 High Performance Feedforward Neural Network Mapped by NoC architecture with a new Routing Strategy Implementation Method 67
6.1 Introduction . . . 67

6.2 Related Works . . . 68
6.3 Proposed iNoC-ANN System . . . 69
6.3.1 Former NoC-ANN System with DT Based Routing Strategy . . . 69
6.3.2 Proposed System . . . 70
6.4 Evaluation of the proposed iNoC-ANN . . . 78
6.4.1 Evaluation of NoC architecture with absolute address based router model . 79
6.4.2 Evaluation of proposed iNoC-ANN and other architecture of ANNs . . . 82
6.5 Conclusions . . . 84

7 A High-performance Neural Processor Based on Network-on-Chip Architecture 87
7.1 Introduction . . . 87
7.2 Related Works . . . 88
7.3 NoC Architecture for Building Neural Computing Processor . . . 89
7.3.1 PE Architecture . . . 89
7.3.2 Router Architecture and Packet Format . . . 92
7.3.3 Architecture of 4x5 NoCNN processor . . . 93
7.4 Implementation and Performance Evaluation . . . 95
7.4.1 Reconfigurability and Extensibility . . . 95
7.4.2 Communication Load Evaluation . . . 97
7.4.3 Performance Evaluation . . . 98
7.5 Conclusion . . . 102

8 Conclusions 103

Publication List 107

Bibliography 109

List of Tables

1.1 Design target of FPGA-ANN 2
1.2 Design target of NoC-ANN 3

2.1 The characteristics of analog ANN 10
2.2 The characteristics of digital ANN 10
2.3 The characteristics of hybrid ANN 11
2.4 Comparison of ASIC, FPGA and DSP method for designing hardware ANNs . . . 12

3.1 Resource and performance of a neuron with different weight precisions 30
3.2 Synthesis report for ANNs 31
3.3 Performance comparison between layer-multiplexing and our method 32

4.1 Comparison of communication load 45
4.2 Simulation environment for NIRGAM 45
4.3 NIRGAM parameters in our experiments 46
4.4 Comparison of CPS [1] 46
4.5 Comparison of CPS with existing systems and proposed system 47
4.6 Results of FPGA 48

5.1 The comparison of power dissipation between mesh topology and torus topology . 59
5.2 Comparison of communication load 63
5.3 Comparison of CPS and CPSPW 65

6.1 Header of DT method increases with NoC size 71
6.2 Header of absolute address based method increases with NoC size 71
6.3 Comparisons of FPGA resource (number of ALUTs) 79
6.4 Comparison of communication load 83
6.5 Comparison of CPS and CPSPW 84

7.1 Comparison of communication load 98
7.2 Comparison of CPS and CPSPW 101

List of Figures

1.1 Flow diagram of our thesis 4

2.1 Class of implementation types of the hardware ANN 8
2.2 Micro-network stack of NoC. 14
2.3 Conceptual realization of NoC 16
2.4 Mesh and Torus topologies 17
2.5 Fat tree topology 17
2.6 Butterfly topology 18
2.7 Phits, flits, packet and messages 19
2.8 Scope of the congestion control and the flow control 20

3.1 Neuron circuit block diagram 24
3.2 Multilayer neural network 25
3.3 Non-pipeline (a), (b), (c) vs pipeline (d) 26
3.4 Schematic of our mapping method 28
3.5 The neural network circuit 29
3.6 Control block (a) Pseudo code for the control block (b) Procedure of topology generation 30

4.1 Structure of FF-ANN 35
4.2 Structure of one neuron 36
4.3 Structure of PE 36
4.4 four-stages pipeline design of PE 37
4.5 A single packet and outputs 38
4.6 Block diagram of a router 39
4.7 NoC packet format 40
4.8 NoC architecture for mapping complex FF-ANN 41
4.9 NoC architectures for mapping 3 applications of FF-ANN 42
4.10 Comparison of total delay, energy, average delay and throughput 43
4.11 Average latency per flit and average throughput 47
4.12 Average latency per flit in pattern recognition application, Neural control application and NPCA application 48

5.1 General structure of FF-ANN 51

5.2 Single neuron architecture 52
5.3 Block diagram of a router for FF-ANN 54
5.4 Block diagram of a Switch Allocator 0 for FF-ANN 55
5.5 Block diagram of a multiple NoC models for FF-ANN 56
5.6 Flow chart of FF-ANN implemented by multiple NoC models 57
5.7 Application of FF-ANN to prediction by model-1 58
5.8 Application of FF-ANN to pattern recognition by model-2 60
5.9 Reparability of the proposed system 62
5.10 Average latency per flit of three applications 64

6.1 NoC-ANN with 4x5 2D mesh topology. 70
6.2 Absolute vs DT header size 72
6.3 Block diagram of the proposed router 72
6.4 Architecture of SA for absolute address based method 74
6.5 Comparison of latency in 4x4 NoC of different routing algorithms. (blue: the former router model)(red: the proposed router model) 80
6.6 Comparison of latency in 8x8 NoC of different routing algorithms. (blue: the former router model)(red: the proposed router model) 81

7.1 20 tiles NoCNN processor 89
7.2 Structure of a multi-layer perceptron 90
7.3 A single neuron architecture 91
7.4 Architecture of a PE 92
7.5 Block diagram of router and pipeline processing 93
7.6 Schematic diagram of the switch allocator 0 for controlling north output port 94
7.7 NoC Packet Format and its information 95
7.8 4x5 NoCNN processor for five ANN applications 96
7.9 NoCNN processor layout design 97
7.10 Average latency per flit of three applications 99
7.11 Comparison of system running time 102

Glossary

Some notations may have different meaning locally.

Notations

x_i	i^{th} input
ω_i	weight in the i^{th} connection
b	bias
N_i	bit length of output from the active function
x_k^{ij}	j^{th} input of i^{th} neuron in k^{th} layer
y_k^i	output of i^{th} neuron in k^{th} layer
N_k	total neuron number in k^{th} layer
$w_{ji}^{(k-1)k}$	weight between the j^{th} neuron in layer $(k-1)$ and the i^{th} neuron in layer k
T	latency
BW	link bandwidth in bits per cycle
R	routing delay per hop
H	number of hops from the source to the destination node
$H_k^{(s)}$	a weighted sum of the k^{th} neuron in the s^{th} layer
$o_j^{(s-1)}$	an output of the j^{th} neuron in the $(s-1)^{th}$ layer
N_{s-1}	the total number of neurons in the $(s-1)^{th}$ layer which connect with this k^{th} neuron
$f(H_k^{(s)})$	an activation function computed on the weighted sum $H_k^{(s)}$

Abbreviations

ADA	absolute destination address
ANN	Artificial Neural Networks
ASNoC	Application specific NoC
CPS	connection per second
CPSPW	connection per second per weight number
DA	Destination Address
DFN	Data from the Former Neuron
DSP	Digital Signal Processors

DT	Destination-Tag
EDA	electronic design automation
FA	Fully-Adaptive
FF-ANN	feedforward artificial neural network
FIFO	First-In First-Out
FSM	finite state machine
FT	flit type
LCB	Logic Cell Block
LM	layer multiplexing method
LUT	look up table
MAC	multiply accumulate circuit
MUX	multiplexer
NF	Negative-First
NL	North-Last
NoC	Network on Chip
NPCA	nonlinear principal component analysis
OE	Odd-Even
P2P	point-to-point
PCI	PE Control Information
PE	Process Element
PT	phit type
RAM	random access memory
RBF	Radial Basis Function
RC	Routing Computation
SA	Switch Allocator
SIMD	single instruction multiple data
SoC	System-on-Chip
ST	Switch Traversal
UN	Used Neuron
VA	virtual-channel allocators
VCC	Virtual Channel Choice
VCN	Virtual Channel Number
WF	West-First

Chapter 1

Introduction

An Artificial Neural Network (ANN) is a computational model or mathematical model which attempts to simulate the structure and functional aspects of biological nervous system [2]. A neural network consists of an interconnected group of nodes which called artificial neurons, and it processes information by computing with a connectionist approach [3]. An ANN is always an adaptive system that changes its structure according to internal or external information during the learning phase.

1.1 Historical review of ANN

ANN has been researched through four periods of extensive activity.

The first period of ANN is in the 1940s, when McCulloch and Walter Pitts began pioneering work based on networks of binary switching [4]. Although the the neuron model is far simpler than the real biological counterparts.

The second period appeared in the 1960s, when Rosenblatt and his group introduced the perceptron convergence theorem [5], and then Minsky and Papert's work showed limitations of a simple perceptron [6]. As a result, research and development of ANN were stagnant for almost 20 years.

In early 1980s, ANN's research started a new period, due to Little's research in 1974 [7], Hopfield's energy approach in 1982 [8] and the back-propagation learning algorithm for multilayer perceptrons (multilayer feedforward networks) by Werbos [9, 10, 11], and then Rumelhart et al. popularized in 1986 [12, 13].

Since the later 1980s the research has begun to focus in neural network VLSI chips, accelerator boards and multi-board neurocomputers, because these hardware ANNs' speed were higher than the software ANNs. Many neural network applications, such as recognition programs, run well

Table 1.1 Design target of FPGA-ANN

Content	Target
Size of ANN	$\leq$ 20 neurons in one layer of ANN [a]
Frequency	100 MHz
Performance of CPS	Increase 50% for ANN with 2 pipeline depth, 2 times for ANN with 3 pipeline depth 3 times for ANN with 4 pipeline depth

[a] The total resource of used FPGA is 9280 LCB, and the resource of one neuron is about 392 LCB.

on conventional von Neumann processors, some applications with high energy physics, need the high speed implementations [14]. For instance, implementation of Back Propagation (BP) neural networks has been performed on the Connection Machine (Singer, 1990) [15], Warp (Pomerleau et al., 1988) [16], MasPar (Chinn & Grasjki et al., 1990) [17], Hughes (Shams & Gaudiot, 1990) [18], GF11 (Witbrock & Zagha, 1989) [19], AAP-2 (Watanabe et al., 1989) [20], transputer based machines (Vuurpijl, 1992) [21], and the CRAY YM-P supercomputer (Leung & Setiono, 1993) [22], Hitachi WSI (Hitachi Central Research Laboratory, 1990) [23], Lneuro (Mauduit, 1992) [24]. To build larger hardware networks, multi-chip boards and even mult-board systems can provide a few thousand interconnected neurons but require at least some sequential chip and board communication. High cost made hardware ANN development slowly.

FPGA implementations of neural networks have a great develop in these years, because of its reconcilability and short design time, such as FPGA neurocomputers (Omondi et al., 2006), Arithmetic precision for implementing BP networks on FPGA (Moussa et al., 2004), FPGA Implementation of Very Large Associative Memories (Hammerstrom et al., 2006), and so on [25]. But there remains a performance problem. If the problem could be solved, the FPGA approach will make hardware ANN a bright future.

In this thesis, FPGA-based ANN are developed and applied to real applications. The work presented here aims to improve the FPGA-based ANN to make it can suit for the high performance applications. In our design, a common FPGA chip (Xilinx VirtexII XC2VP20) is used, thus the design target is described as Tables 2.1.

Traditional bus-based P2P hardware ANN could not overcome the communication problem. But NoC technology can overcome almost all the drawbacks in hardware ANN. In this thesis, NoC-based ANN is proposed and applied to real applications, so that problems of performance, communication load and reconfigurability will be solved. The design target of proposed NoC ANN

Table 1.2 Design target of NoC-ANN

Content	Target
Size of ANN	$\leq$ 72 neurons of total layers for general NoC model $\leq$ 64 neurons in one layer of ANN with multiple NoC model
Frequency	100 MHz
Communication load	Reduce 50% compared with P2P architecture
Performance of CPS	$>$ 1 G
Power consumption	$<$ 2 W
Chip area	25 mm^2

is described as Tables **??**.

1.2 Organization of This Thesis

This thesis presents two architecture to develop the hardware ANN. One architecture is based on the existing FPGA-based ANN, and integrate layer-multiplexing and pipeline architecture together. While the developings based on the existing architecture could not overcome all the drawbacks in the existing ANN. Thus, a novel NoC architecture is newly proposed to improve the hardware ANN. The NoC-based ANN shows advantages of high performance, low communication load and good reconfigurability.

This thesis consists of eight chapters and depicted in Fig. 1.1. Chapter 1 gives a background and motivation of our research.

Chapter 3 presents a novel architecture for an FPGA-based implementation of multilayer Artificial Neural Network (ANN), which integrates both the layer-multiplexing and pipeline architecture. Given a kind of FPGA to be used, the proposed method aims at enhancing the efficiency of resource usage of the FPGA and improving the forward speed at the module level, so that a larger ANN can be implemented on traditional FPGAs and also a high performance is achieved. Usually FPGA board is not changed for every applications, thus, we need not mind about the usage of it if the application can be implemented within the resource limitation. We developed a new mapping method from ANN schematic to FPGA by using this hybrid architecture, and also developed an algorithm to automatically determine the architecture by optimizing the application specific neural network topology. The experimental results show that the proposed architecture can produce a very compact circuit for multilayer ANN to meet resource limitation of a given FPGA, and higher performance is obtained compared

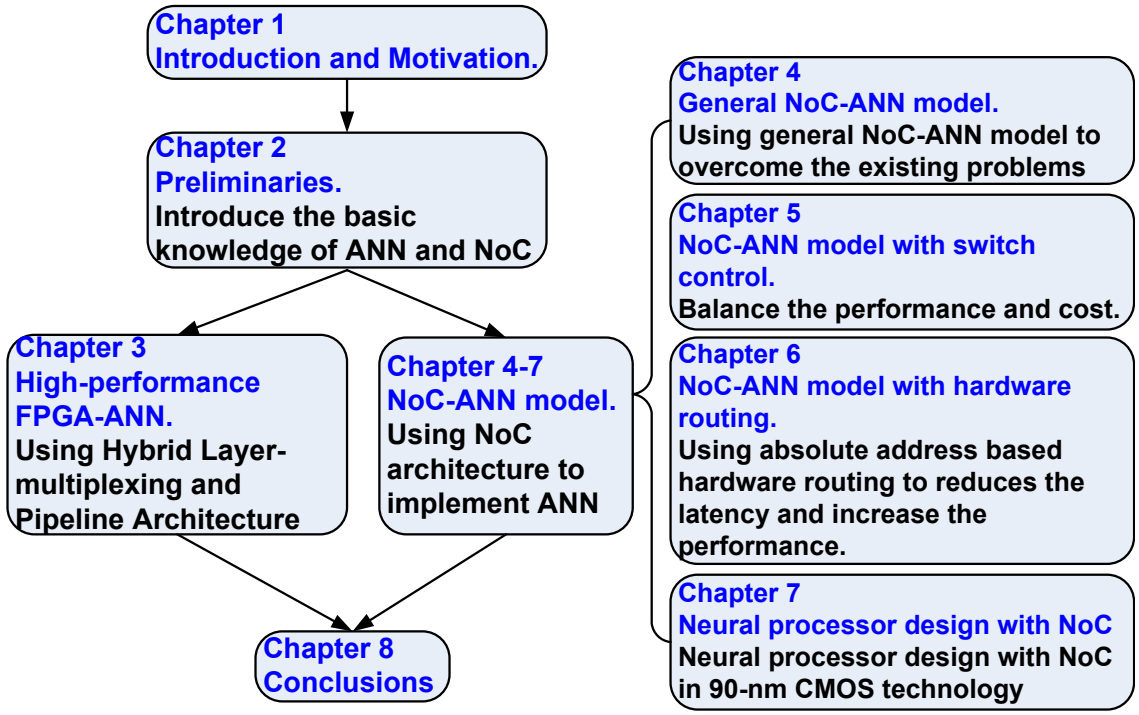


Figure 1.1 Flow diagram of our thesis

with conventional methods.

The main contributions in this chapter are as follows.

- The proposed architecture integrates both the layer-multiplexing and pipeline architecture.
- The proposed architecture can relax resource limitation problem of a given FPGA, and shows higher performance.

Chapter 4 proposes a new flexible hardware Network on Chip (NoC), a packet-based signal processing architecture, for mapping the different applications of the feedforward artificial neural network (FF-ANN). There are many problems in a traditional FF-ANN implementation. For example, application is limited, interconnection is complex and data transmission is difficult to be controlled. This NoC-based system can solve these problems, because it can be reconfigured and extended by sending the new packet. It can map the complex FF-ANN with multiple layers and multiple neurons in one layer. The system is designed to achieve low

latency, high throughout and low power. The simulation results obtained with the NIRGAM NoC simulator, NOXIM NoC simulator and Quartus II show that the proposed architecture can reduce communication load and increase connection per second (CPS) of system in real applications of FF-ANN.

The main contributions in this chapter are as follows.

- A novel architecture is proposed to overcome the existing problems for hardware ANN.
- The proposed NoC-ANN can reduce communication load and increase CPS of system in real applications of ANN.
- The proposed NoC-ANN is reconfigurable and extendable, and it can be used for different applications.

Chapter 5 proposed a multiple NoC models for ANN, which can implement both a small size ANN and a large size one. The simulation result shows that the proposed multiple NoC models can reduce communication load, increase system performance of CPS and reduce system running time compared with the existing hardware ANN. Furthermore, this architecture is reconfigurable and reparable. It can be used to implement different applications of ANN.

The main contributions are as follows.

- The NoC-ANN with switch control can implement both a small size ANN and a large size one.
- The performance and resource cost is balanced.

Chapter 6 proposed a new router model of NoC with absolute address based routing strategy instead of a router with Destination-Tag to reduce the packet size of a header. The NOXIM NoC simulator is used to evaluate the proposed router in term of average latency and max latency. The experimental results indicate that the proposed NoC architecture with this new router model is effective in reducing latency compared with the traditional one, and it brings a mapped FF-ANN higher performance and lower communication load.

The main contributions related to this NoC-ANN with absolute address based routing strategy are as follows.

- The proposed NoC architecture with absolute address based routing strategy is effective in reducing latency compared with the traditional NoC-ANN.

- Higher performance and lower communication load are achieved.

Chapter 7 describes a high-performance neural processor based on a novel Network on Chip (NoC) architecture to increase the computing speed and solve the reconfigurability and interconnection problems of a general neural system. The proposed NoC-based neural processor is composed of 20 tiles in a 4x5 2-D array, and each tile includes a Process Element (PE) and a packet-switched router. In each PE, four neurons are aggregated to achieve low communication load. The network is 2-D torus topology, and it has a 32 G/s bandwidth and asynchronous clocking system. Our proposed neural processor is designed using 90-nm CMOS technology with one poly and nine metals. It can achieve over 3.1 Giga Connection Per Second (CPS) of computing speed while power dissipation is 1.1317 W at 1.2 V supply, and its chip size is 25 mm^2. Compared with the other existing digital neural networks, the proposed processor is reconfigurable and extendable, and can achieve lower communication load, lower system running time and higher computing speed.

The main contributions related to this NoC-based neural processor are as follows

- Based on the proposed NoC-ANN model, a high-performance neural processor with NoC architecture is designed using 90-nm CMOS technology.
- The developed NoC neural processor can achieve lower communication load, lower system running time and higher computing speed.

Chapter 8 concludes this research, summarizes the thesis and discusses the further work.

Chapter 2

Preliminaries

2.1 Basic Knowledge of ANN

Hardware ANN

Why a hardware designed ANN is needed. It is because of the performance of conventional von Neuman processors. For example, the Intel Pentium series processor continues to be improved dramatically, but we need higher performance. Then we may have another question that why such a neural network algorithms need implement in special hardware. No doubt it is also because of the speed. Clark S. Lindsey [26] gave three reasons in his speech: The first, even the fastest sequential processor can't provide real-time response and learning for networks with large numbers of neurons and synapses. The second, parallel processing with multiple simple PEs can provide great speed ups. The third, when the particular task at hand does not require very fast speed, we can find a software implementation on a PC or workstation to obtain a satisfactory solution. Furthermore the specialized applications can also motivate the use of hardware ANN.

Classify of implementation types of the hardware ANN

The classification of implementation types of the hardware ANN is always a controversial task. One of the general classification was proposed by Heemskerk (1995) [27] and revised as shown in Fig. 2.1. The global hardware implementation is called neurocomputers which can be divided into standard chips or neurochips. The standard chips can be classified as sequential, accelerator boards or processor. The FPGA is one of the common types of processor. The neurochips which are constituted by ASIC can be classified as analog, digital or hybrid.

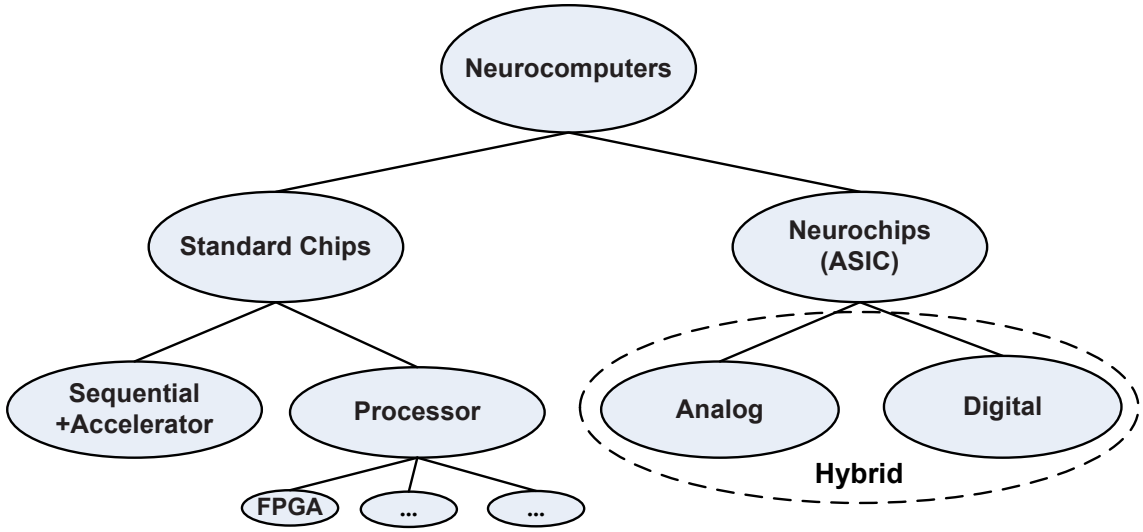


Figure 2.1 Class of implementation types of the hardware ANN

Detail information of different types of hardware ANN

The detailed information of different types of hardware ANN includes architecture, learn type, precision, number of available neurons, number of synapses and a measurement of speed or performance. The performance which we are really concerned in is affected by other information.

The architecture information usually refers to the type of network to be implemented, sometimes the additional information about the type of implementation will be also included. For example, feedforward multilayer networks, matrix operation circuits, general purpose (GP), single instruction multiple data (SIMD), floating point (FP), integer (Int), radial basis function (RBF), fully connected and recurrent (FCR), systolic array devices, slice, and so on. Different architecture of hardware ANN has different character which can be suitable for specific applications.

The learn information is described about the possibility of on-chip learning for the ANN. Some types of on-chip learning are listed as follows: Program when the algorithm can be uesd, Hopfield, Boltzmann, back propagation (BP), region of influence (ROI), restricted coulomb energy (RCE), probabilistic neural networks (PNN), K-nearest neighbour (KNN). Since the on-chip learning is always high cost, the off-chip are wildly used in these years.

The precision information indicates the format type of bits and the number of bits for input and output. High precision results high cost.

The neuron information denotes the number of neurons available in the system. In some cases the system contains PEs instead of neurons, and each PE has all functions which a neuron has.

The synapse information denotes the number of connections or the amount of memory available for storing the weights.

The performance information of the ANN system is always measured in many different ways. The Connection-Per-Second (CPS) which suggested by Lindsey and Lindblad (1994) [28] is the most common performance rating, and it is defined as the rate of multiplication and accumulation operations. However, some systems could get a high CPS according to the large number of neurons. So the value of CPS must be normalized by dividing it by the number of weights N_w, that is, the connection per second per weight number (CPSPW) suggested by Holler (1991) [29] as shown in Eq. (2.1.1). Connection update per second (CUPS) which rates the number of weight changes per second is a measure for the learning phase, if the ANN system includes on-chip learning. Another measure is connection primitives per second (CPPS), which can be calculated as Eq. (2.1.2) which was proposed by Keulan (1994) [30], where b_{in} is the number of bits used for the inputs and b_w is the number of bits used for the weights. This CPPS adds the precision into the performance measure.

$$CPSPW = CPS/N_w \ . \tag{2.1.1}$$

$$CPPS = b_{in} \times b_w \times CPS \ . \tag{2.1.2}$$

Other major distinguishing features of ANN were as follows: the number of PE's, maximum network size, whether the chips could be chained together to increase network size, accumulator size in bits, the whole system cost [26].

Comparison of existing hardware ANN

We will introduce analog, digital, DSP (Digital Signal Processors) and FPGA ANN in this subsection, because they are most commonly used.

Tables 2.1-2.3 show the characteristics of the analog ANN, digital ANN and hybrid ANN, and each table includes the features of architecture, learn, precision, the number of neurons, and synapses, speed respectively.

Many analog ANN have been realized. They are very fast, low-area and low-power, but the problems are also exist, such as low precision, difficult for designing data storage and on-chip

Table 2.1 The characteristics of analog ANN

Name	Architecture	Learn	Precision	Neurons	Synapses	Speed
Intel ETANN [31]	Feedforward, ML	No	6x6 bits	64	10280	2 GCPS
Synaptics silicon retina [26]	Neuromorphic	No	N.A.	48x48	Resistive net	N.A.

Table 2.2 The characteristics of digital ANN

Name	Architecture	Learn	Precision	Neurons	Synapses	Speed
Micro devices MD-1220 [1]	Slice, ML Feedforward	No	1x16 bits	8	8	1.9MCPS
NeuraLogix NLX-420 [1]	Slice, ML Feedforward	No	1-16 bits	16	Off chip	300CPS
Philips Lneuro-1 [24]	Slice, ML Feedforward	No	1-16 bits	16PE	64	26MCPS
Philips Lneuro-2.3 [32]	Slice, ML Feedforward	No	16-32 bits	12PE	N.A.	720MCPS
Inova N64000 [28]	SIMD GP, Int	Program	1-16 bits	64PE	256k	870MCPS
Hecht-Nielson HNC 100-NAP [33]	SIMD GP, FP	Program	32 bits	4 PE	512k	250MCPS
Hitachi WSI Wafer[34]	SIMD Hopfield	Program	9x8 bits	576	32k	138MCPS
Hitachi WSI Wafer[35]	SIMD BP	Program	9x8 bits	144	N.A.	300MCPS
Neuricam NC3001 [36]	SIMD, ML Feedforward	No	32 bits	1-32	32k	1GCPS
Neuricam NC3003 [37]	SIMD, ML Feedforward	No	32 bits	1-32	64k	750MCPS
RC Module NM6403 [38]	SIMD, ML Feedforward	Program	1-4096bits	1-64	1-64	1.2GCPS
Siemens MA-16 [39]	Systolic array Matrix ops	No	16 bits	16PE	16x16	400MCPS
Nestor/Intel NI1000 [28]	RBF	RCE, PNN Program	5 bits	1PE	256x1024	40kpat/s
IBM ZISC ZISC036 [40]	RBF	ROI	8 bits	36	64x36	250kpat/s
IBM ZISC ZISC78 [41]	RBF	KNN, L1 LSUP	N.A.	78	N.A.	1Mpat/s

Table 2.3 The characteristics of hybrid ANN

Name	Architecture	Learn	Precision	Neurons	Synapses	Speed
AT&T ANNA [42]	Feedforward ML	No	3x6 bits	16-256	4096	2.1GCPS
Bellcore CLNN-32 [43]	FCR	Bolt.	6x5 bits	32	992	100 CPS
Mesa Research Neuralclassifier [44]	Feedforward ML	No	6x5 bits	6	426	21GCPS
Ricoh RN-200 [45]	Feedforward ML	BP	N.A.	16	256	3GCPS

learning, and so on [46, 47]. Furthermore, it is expensive and not flexible, and their development work is very tricky so that designer has to be familiar with analog technology.

Many digital ANNs have also been designed. Compared to analog ANN, they provide high precision, weights storage in RAM, easy to integrate with other applications. Learning algorithms could be implemented, and digital ANNs are easily embedded into most applications. On the other hand, digital ANNs are always slow, and the implementation of activation functions is difficult compare with analog one. The digital ANN includes many sub-Class including slice architectures, SIMD and systolic array devices and RBF architectures. The Micro Devices MD1220 [1] was the first commercial digital ANN with slice architecture. The slice ANN also includes Neuralogix NLX-420 [1], Philips Lneuro 1.0 chip [24] and Philips Lneuro 2.3 chip [32]. SIMD chips include the Inova N64000 [28], the HNC 100 NAP [33], Hitachi WSI with Hopfield or BP on-chip learning [34, 35], Neuricam NC3001 TOTEM [36], Neuricam NC3003 TOTEM [37], RC module NM6403 [38] and so on. Some hardware ANNs have systolic array architecture. For example, the Siemens MA-16 is used by Beichter et al. [39], and other design groups use this architecture to design their ANN system [48, 49, 50]. The IBM ZISC036 chip [40], the Nestor Ni1000 chip [28] and Silicon recognition (also called IBM ZISC) ZISC78 chip [41] are designed by RBF architecture.

Some design groups try to obtain the advantages of analog and digital systems, and then the hybrid ANN appeared. Commonly the external inputs and outputs are digital, while most of internal processing parts are analog. The typical hybrid ANN are AT&T ANNA [42], Bellcore CLNN-32 [43], Mesa research neuroclassifier [44], Ricoh RN-200 [45], and son on.

There are also some DSP ANN design as in [51, 52, 53, 54]. These designs suffer from cost and development time.

FPGA ANN developed quickly due to easy reconcilability and short design time make, therefore

Table 2.4 Comparison of ASIC, FPGA and DSP method for designing hardware ANNs

	ASIC(analog)	ASIC(digital)	FPGA	DSP
speed	+++[1]	++[2]	+[3]	+
area	+++	++	+	- -
design cost	- -[4]	- -	++	- -
design time	- -	- -	++	+
reliability	- -	+	++	++

1 +++: Excellent; 2 ++: very good; 3 +: good; 4 - -: very bad.

it is widely used these years [55, 56, 57]. While the speed of FPGA ANN should be improved to suit some applications which require high speed, the FPGA ANN has several advantages for implementation of ANN, that is, reprogrammable FPGA ANN permit prototyping, on-chip learning is difficult and not used in FPGA ANN, FPGA ANN could be used for embedded applications, FPGA ANN may be mapped onto new improved FPGAs to get high performance [58].

As described above, different kinds of hardware ANN are roughly summarized and compared with speed, area, design cost, design time and reliability in Table 2.4.

In general, there is no perfect architecture to implement the ANN, thus, a novel architecture for ANN is required.

2.2 Basic Knowledge of Network on Chip (NoC)

2.2.1 System on Chip (SoC) Challenges

During the 1990s, a lot of researchers began to integrate more and more IP-cores and application components on one single silicon die, such as so-called System on Chip (SoC). A lot of complex applications are integrated onto single chip, not only functionally aggregated. The real products such as mobile phones, notebook-computers and personal digital assistant are becoming lower-in-power, higher-in-performance, smaller-in-size, lighter-in-weight, larger-in-capacity and cheaper-in-price. And this trend will continue indubitably. Designers expect to integrate much more complex applications and even systems onto a single chip. However, the current methodologies for SoC design are not well enough due to the big design challenges. For example global synchrony [59, 60, 61], communication architecture [62, 63], deep submicron effects [64, 65, 66, 67], interface standardization [68], Power management [69, 70], verification [71, 72], design productivity gap [73, 74] and so on. Some of these challenges are described in detail as follows:

- **Global synchrony**. Current systems are generally designed following a globally synchronous

design style. A global clock tree is distributed on the chip and logic blocks function synchronously. However, this style seems to be not suitable for the future wire interconnect. Because the technology scaling does not treat gate delay and wire delay equally. Gate delay has been getting smaller to suit for the gate length, but the reduction of wire delay is slowed down [59, 60]. Furthermore, Chip design is to be constrained by communication rather than capacity. Thus, a future chip is likely to be partly local regions synchronous and global asynchronous, such as GALS (Globally Asynchronous Locally Synchronous) [61].

- **Communication architecture**. Almost all the components use the buses or p2p (point to point) method to connect with each other. Buses are widely used because they can provide high performance interconnections while they can still be shared by several communication partners. But buses do not scale well with the system size in terms of bandwidth, clocking frequency, power and so on [62]. Disadvantages of bus are 1. Communication capability of the bus system is very limited which only one device can drive a bus segment at a time, 2. The speed of bus speed is difficult to scale up when the number of clients grows, because the intrinsic resistance and capacitance of the bus also increase, 3. The entire bus wire has to be switched on at the data transmission process, because the data transfer is broadcast. The energy cost is wasteful, 4. Buses can efficiently connect a few communication partners but they can not connect higher numbers [75]. A bus architecture will become critical in performance and has energy bottleneck in the future chip design. A novel on-chip communication architectures are required [63].

- **Deep submicron effects**. In early days of VLSI design, interconnect of chip was reliable and robust. When the scale below than 250 nm with aluminum and 180 nm with copper, interconnect started to become a dominating factor for chip performance and robustness, and when the transistor density is increased, wires are becoming slower and unreliable [66]. Long, global wires and buses become undesirable due to their low and unpredictable performance, high power consumption and noise phenomenon [64, 65].

2.2.2 Network on Chip (NoC) Platform

According to these challenges of SoC design, the Network on Chip (NoC) idea was proposed by some research groups to solve these problem in around year 2001 [76, 77, 78, 79, 80, 62]. Soon they found that it had to be addressed at all levels from the physical to the architectural to the operating system and application level. That's why a lot of NoC research groups focus on different levels

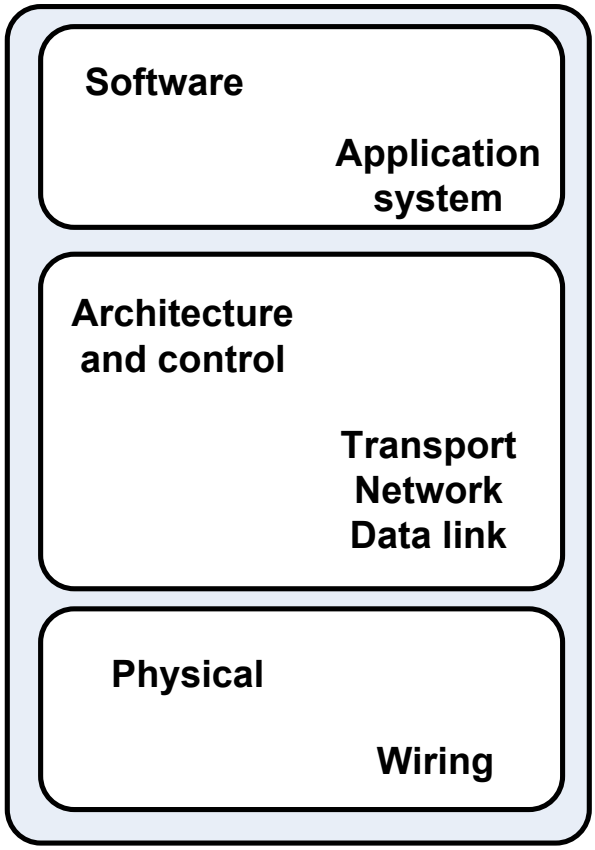


Figure 2.2 Micro-network stack of NoC.

which include software level, architecture and control level and physical level as shown in Fig. 2.2. All this together can be called an NoC platform. The features of NoC are reuse and predictability [81].

- **Reuse**. Reuse has always been the primary means to bridge the technology gap [82, 83, 84]. The reuse of processor cores has been developed during the last ten years by defining bus interfaces. With NoC platform, the general components could also be reused, such as processor cores, DSP cores, memory banks, graphics processors, FPGA blocks, and so on. The special application of NoC is reuse in communication services.

- **Predictability**. Due to its regular geometry architecture and communication network, the

communication performance becomes much more predictable. The regular and known geometry leads to the accurate analysis of electrical properties. Reuse will decrease uncertainties and risk in particular for the verification tasks. This naturally makes design and verification time more predictable.

Furthermore, NoC research not only deal with SoC design but also creates a new area [81]. The NoC method are widely used in the hardware communication infra-structure, the middleware and operating system, application programming interfaces and so on [68, 85].

All these advantages do not come for free. Essentially we must pay for loosing optimality. We know that the NoC has the feature of reusing, but the reusing components always mean that we use something more general, and then less optimal for a particular task. As refer from [81], we known that using a fixed and inflexible network topology means that all the other topologies can not be used which may be more suitable for this application at hand.

2.2.3 Network on Chip (NoC) Architecture

NoC is designed using principles that were investigated for multiprocessor computers as well as for local and wide area networks. Conceptual realization of NoC is shown in Fig. 2.3.

The elements of a network are the processing elements (PEs) and storage units, which called nodes, switches and physical links. To meet the performance specifications of a particular application, the network designers must implement the topology, routing and flow control of the network by technology constraints. We will explain the topology, switching, routing and flow control in more detail.

Topology

Network topology has been studied deeply in the context of high performance networks and parallel computers architectures [86]. We know that NoC differs from general networks because they are realized on a plane, links between routers can travel only in X or Y direction, in a limited number of planes (the number of metal layers of the IC process). As a result, many NoC have topologies that can be easily mapped to a plane, such as low-dimensional meshes, torus (shown in Fig. 2.4), crossbar, N-dimensional k-ary mesh, K-ary n-cube, express cube, D-dimensional K-ary (fat) trees (shown in Fig. 2.5), butterfly (shown in Fig. 2.6) and irregular [87, 88, 78, 89, 90, 91, 92].

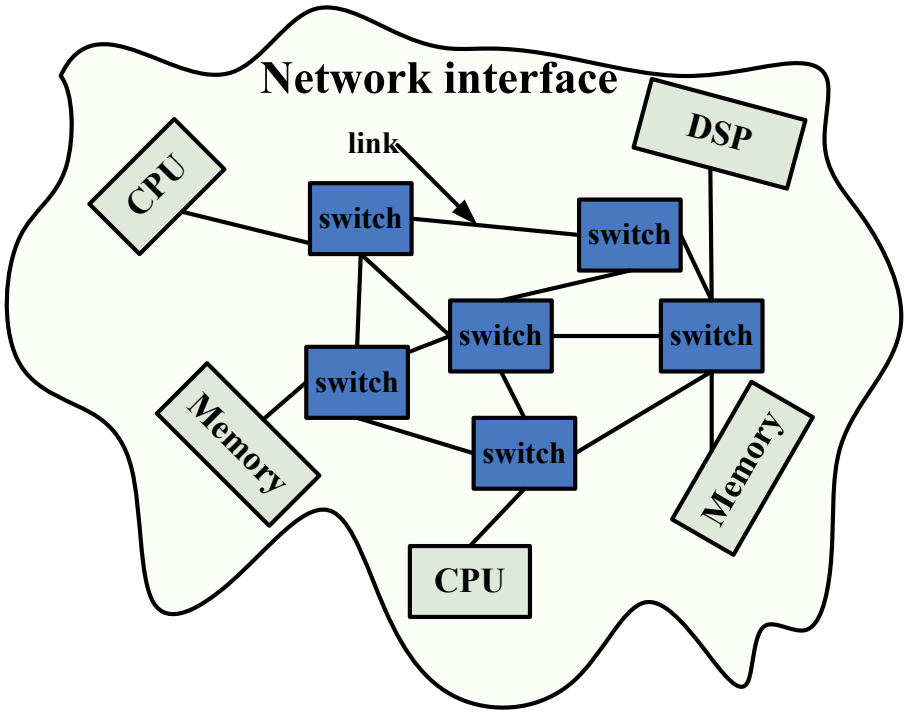


Figure 2.3 Conceptual realization of NoC

Switching

Once the topology of NoC is decided, the switching technique which include data flows control need be determined. Data is transferred on a link with a fixed width, and the bits are used to measure it. Always, the unit of data transferred in a single cycle on a link is called the phit. The unit of synchronization is flit, and the size of it is equal or larger than a phit. Several flits constitute a packet and several of packets make up message. Fig. 2.7 shows the structures of phits, flits, packet and messages. To increase the efficiency of the message packetization,the boundaries of packet need not to be aligned [93]. Different NoC designs have different phit, flit, packet and message sizes. The sizes of phit and flit always reflect different design choices, such as speed of link and router arbitration. For example, Nostrum [94] uses phits and flits of 128 bits which flits equal to phit. SPIN [95] uses phits and flits of 36 bits, and packets are unbounded in length.

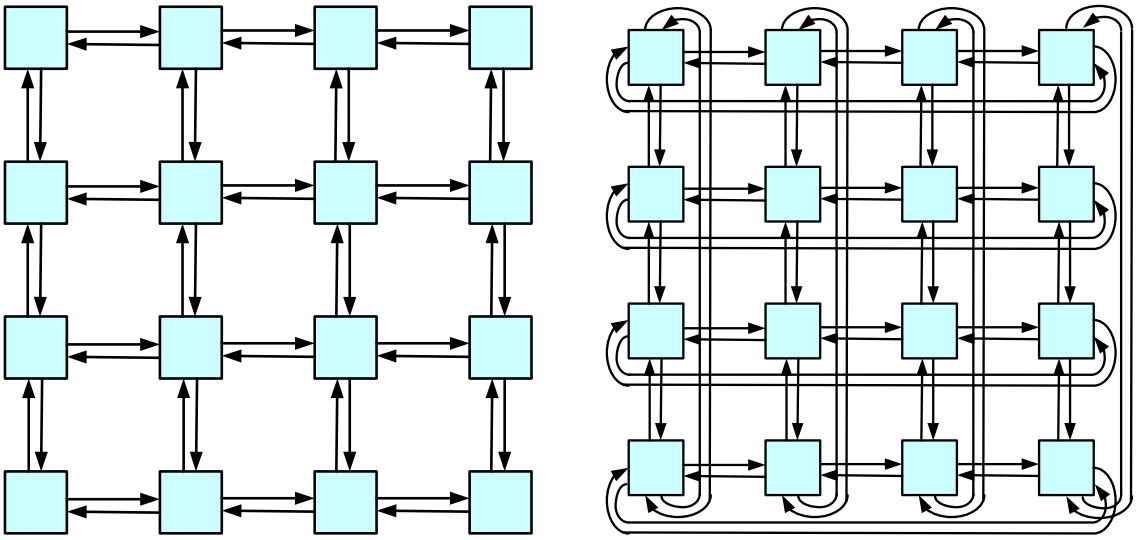


Figure 2.4 Mesh and Torus topologies

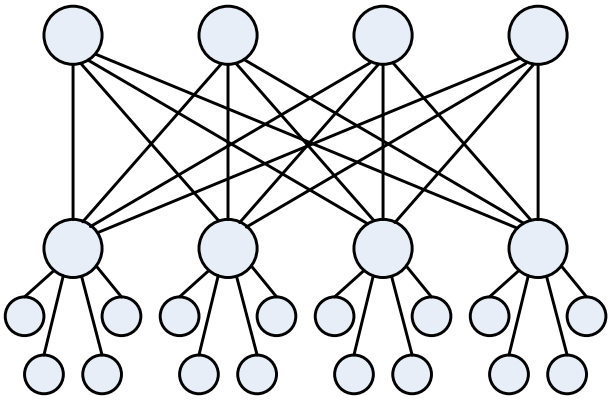


Figure 2.5 Fat tree topology

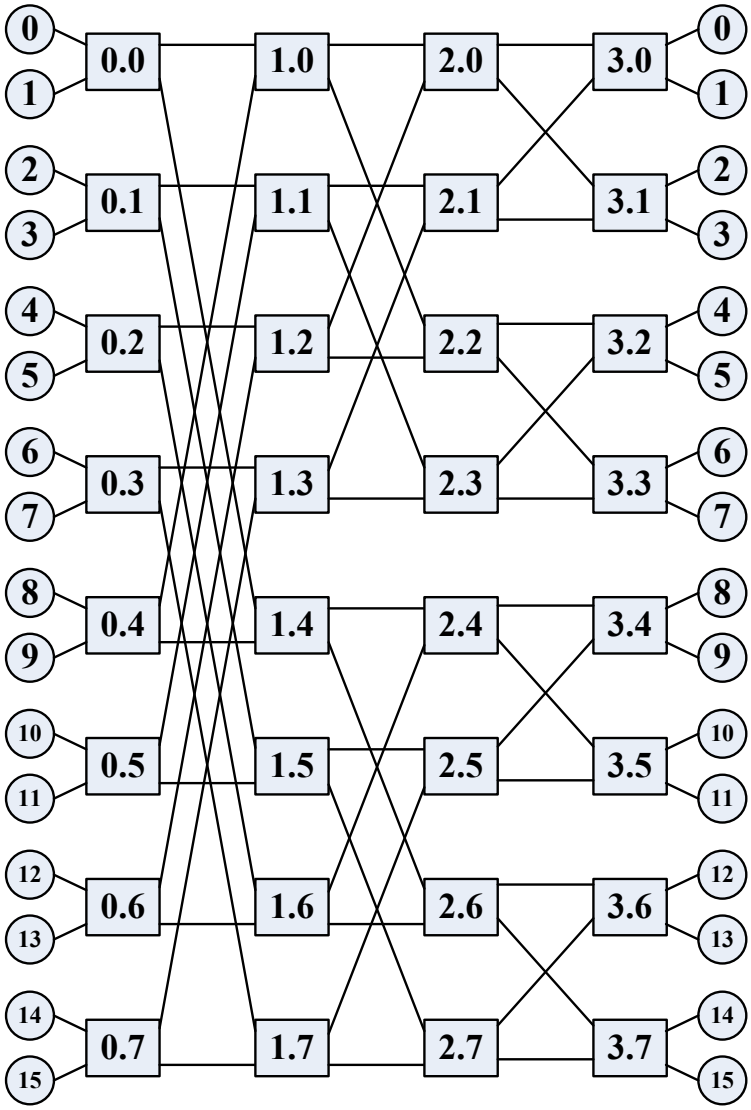


Figure 2.6 Butterfly topology

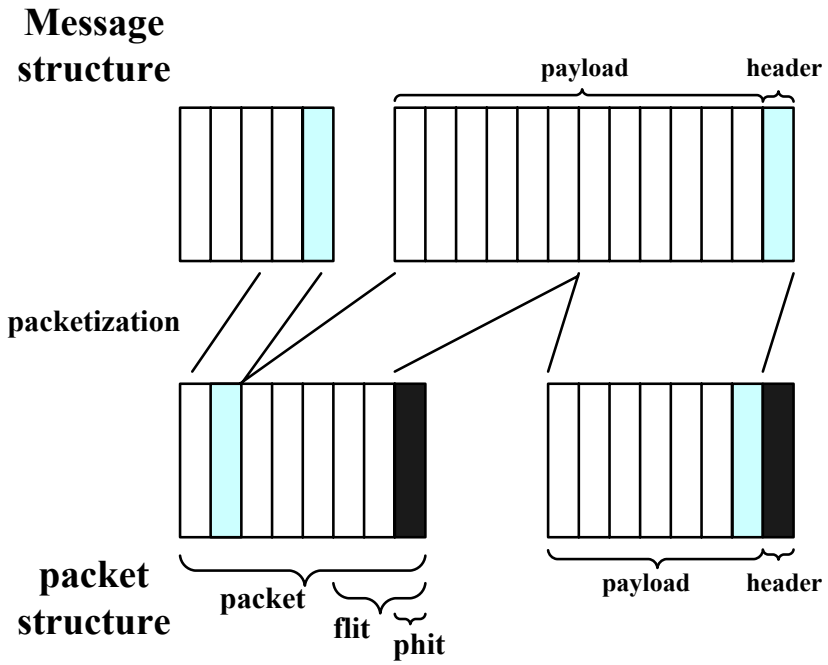


Figure 2.7 Phits, flits, packet and messages

Routing

The routing data in NoC is introduced and particularly emphasize the planar mesh topology, which is popular for NoC and will be used in this thesis. The NoC routing is responsible for correct and efficient routing of packets that are transmitted in the network from sources to destinations. The routing protocol should deal with routing decision made at every router. NoC need not follow rigid networking standards as the traditional communication or interconnection networks. A multiple routing schemes can be evaluated and compared for each NoC implementation. When we design the routing for NoC, the following potentially conflicting metrics should be balanced:

- **Power**. The power required to route packets should be minimized. It means that packets or messages may follow the minimal power path as traditional shortest distance routing [90, 96]. For example, when dynamic voltage scaling (DVS) is applied in a uncommon way, each router and link will offer a different power consumption for packet switching [97].
- **Performance**. The metrics of performance should be balance which reduces the delay or maximize the traffic utilization of the network.

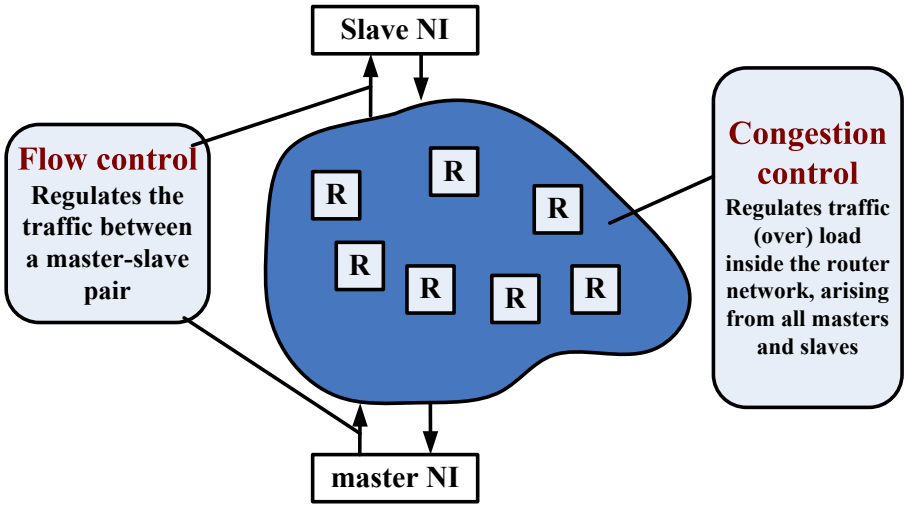


Figure 2.8 Scope of the congestion control and the flow control

- **Area and VLSI resources**. The hardware implement of routing and even the software implement of routing will consumes hardware resources like state machines, addresses tables and so on. And the routing may also uses the bandwidth if routers change information with each other.

Routing can be classified into several different categories such as static or dynamic, distributed or source, hardware or software implemented, and minimal or non-minimal. The hardware implemented routing will be described in our thesis in detail.

Congestion control and flow control

Packets are transmitted through the router network which always uses the same resources which include buffers, allocators and links at the same time. This may result contention, and contention can proliferate. At last, the contention will reduce the network performance.

Here we assume there are two communicating parties where a master and a slave are not balanced with data injection and consumption rates. As we know, a slow slave can cause a backlog of packets inside the router network.

Fig. 2.8 shows the scope of the congestion control and the flow control. We can find that, congestion control keeps the router network free of traffic jam-up. At the same time, flow control makes sure that no sender is overwhelming even one of its receivers.

Chapter 3

A Hybrid Layer-multiplexing and Pipeline Architecture for Efficient FPGA-based Multilayer Neural Network

This chapter presents a novel architecture for an FPGA-based implementation of multilayer Artificial Neural Network (ANN), which integrates both the layer-multiplexing and pipeline architecture. Given a kind of FPGA to be used, the proposed method aims at enhancing the efficiency of resource usage of the FPGA and improving the forward speed at the module level, so that a larger ANN can be implemented on traditional FPGAs and also a high performance is achieved. Usually FPGA board is not changed for every applications, thus, we need not mind about the usage of it if the application can be implemented within the resource limitation. We developed a new mapping method from ANN schematic to FPGA by using this hybrid architecture, and also developed an algorithm to automatically determine the architecture by optimizing the application specific neural network topology. The experimental results show that the proposed architecture can produce a very compact circuit for multilayer ANN to meet resource limitation of a given FPGA, and higher performance is obtained compared with conventional methods.

3.1 Introduction

Artificial neural networks (ANNs) are characterized as an adaptive, robust and parallel computing model, which has the capability to learn by using examples and to approximate any given functions [2]. It has been widely applied to the fields of signal processing, speech synthesis, pattern recognition, and so on [98, 99, 100]. Most of these applications require high-speed computation to meet the performance requirements. The traditional methods are executed by the general processors based

on Von-Neumann architecture, and it cannot meet the speed requirement when the network size becomes large. And the area and design cost are also high by using the general processor which makes the commercial application of neural network impossible [26]. So it is necessary to develop such a custom high performance, and small sized hardware architecture that can exploit the inherent parallelism of neural network models.

The general hardware platforms for implementing ANNs are DSP, ASIC, FPGA and so on [101, 102, 58]. Compared with DSP or ASIC platforms, FPGA is the most suitable for ANN implementation as it offers the parallel computation and re-configurability [103]. A lot of research groups rely their applications on an FPGA method [104, 105, 106].

The challenges of FPGA-based ANNs research fall into two categories, one is to increase the performance and another is to reduce the resource usage. Architectural efforts to improve the performance of FPGA-based ANNs are reported in [107, 108], while the resource reduction is highly required. As we know, the FPGA resource spent on one neuron is so high that it's hard to implement a whole ANN on a single FPGA chip. There are several solutions to overcome the problem of limited FPGA resource, such as the optimization of co-design [109], stochastic model [110] and multiplexing [111, 112]. All these approaches focus on reducing the resource required the neural network at the expense of processing speed. Consequently, those approach makes the FPGA-based ANN architecture not used for a lot of applications of ANN which require high performance.

By the development of semiconductor technology, available resources in a current FPGA chip become much higher, so that the resource constraints are relaxed. Therefore it is necessary to take the processing speed problem becomes critical when designing the architecture for the ANN implementation on FPGA.

The purpose of this chapter is to design a novel architecture for high speed implementation of FPGA-based ANN by enhancing the efficiency of resource usage of the same FPGA board. Our proposed architecture presents an advantage in two basic respects over the previous reported implementations. The first one is the hybrid of layer multiplexing and pipeline, which can optimize both the resource requirement and speed. The layer multiplexing guarantees the resource required by neural network under the constraint of an adopted FPGA chip, and the pipelining between the layers can improve the speed. The second point is the algorithm to determine the optimal hardware architecture according to the neural network parameters such as the topology, data structure and so on. Furthermore, our method just meet the resource limitation of a given FPGA, so that the FPGA board is not changed for another application.

The rest of this chapter is organized as follows: Sect. 3.2 describes the circuit design of the

single neuron and Sect. 3.3 presents the network design and the details of the proposed architecture. Sect. 3.4 describes experiments to validate the advantages of the proposed method. Finally, Sect. 3.5 concludes the chapter.

3.2 Neuron Design

3.2.1 Mathematical Model

A multilayer neural network is composed of one input layer, several hidden layers for computation and one output layer. Each layer consists of a set of processing elements called neurons and the main task of each neuron is processing the following function [113]:

$$y = f(x) = f(\sum_{i=1}^{n} \omega_i x_i + b) \ . \tag{3.2.1}$$

where x_i stands for the i^{th} input, and ω_i is the weight in the i^{th} connection and b is the bias. The function $f(x)$ is the nonlinear active function used in the neuron. Here we select the log-sigmoid as the active function due to its popularity [1], and it is described by the following:

$$f(x) = \frac{1}{1 + e^{-x}} \ . \tag{3.2.2}$$

Because RAM is usually used to store the weights, we did not use the online learning in our proposed architecture, so that the hardware resource is saved.

3.2.2 Neuron Circuit Design

Most of hardware implementation of ANNs focus on the forward computing work and leave the learning work to computer as an off-chip learning, so does the FPGA based ANNs. As shown in the above mathematical model and ref. [3], the computational resources required by a single neuron are a multiplication block, an accumulation block and an active function block.

Fig. 3.1 shows a block diagram of the neuron circuit, where En is an enable signal that controls the neuron's state, that is, the neuron is working or not. The selection of word length, that is, bit precision is important for the output resolution, where longer bits mean a higher resolution but also it takes a larger resource cost. And in actually ANNs design, these parameters are set according to the application in order to achieve the efficient hardware implementation. In this chapter, we consider the parameters as variables, which can be modified by users in the compilation step.

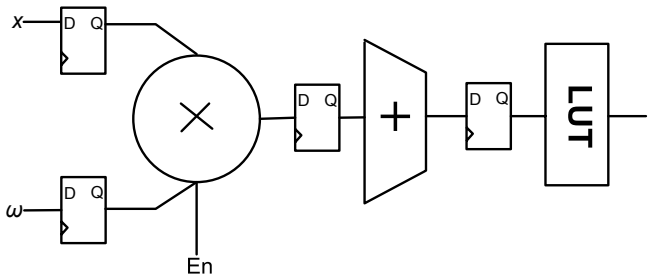


Figure 3.1 Neuron circuit block diagram

The active function is realized by using LUT(look up table) [112] according to the fact that the modern FPGA chip has a large number of built-in RAMs. As the active function is highly nonlinear, a general procedure to obtain an LUT of the minimum size for a given resolution is as follows.

As mentioned above, the bit length of output from the active function (Eq.(3.2.2)) is N_i.

1) The actual output of the active function is the value between 2^{-N_i} and $1-2^{-N_i}$ for the bit length N_i. Let x_1 and x_2 be the upper and the lower limits of the input range respectively, that is:

$$\frac{1}{1+e^{-x_1}} = 2^{-N_i} , \qquad \frac{1}{1+e^{-x_2}} = 1-2^{-N_i} . \tag{3.2.3}$$

By solving Eq.(3.2.4):

$$x_1 = -ln(2^{N_i}-1) , \qquad x_2 = +ln(2^{N_i}-1) . \tag{3.2.4}$$

2) Consider the fact that the step change in the output (Δy) is equal to 2^{-N_i}, and the corresponding minimum change in input is at the point of maximum slope, $x = 0$ in this case. So the minimum change value of input for the output change of 2^{-N_i} can be obtained from

$$\Delta x = ln(\frac{0.5+2^{-N_i}}{0.5-2^{-N_i}}) . \tag{3.2.5}$$

3) The minimum number of LUT values is given by

$$(LUT)_{min} = \frac{x_1 - x_2}{\Delta x} . \tag{3.2.6}$$

3.3 Network Design

Network connects all the neurons and all the layers together, so the data can be forwarded through the connections from the former layer to the latter layer. Fig. 3.2 gives an example of a multilayer

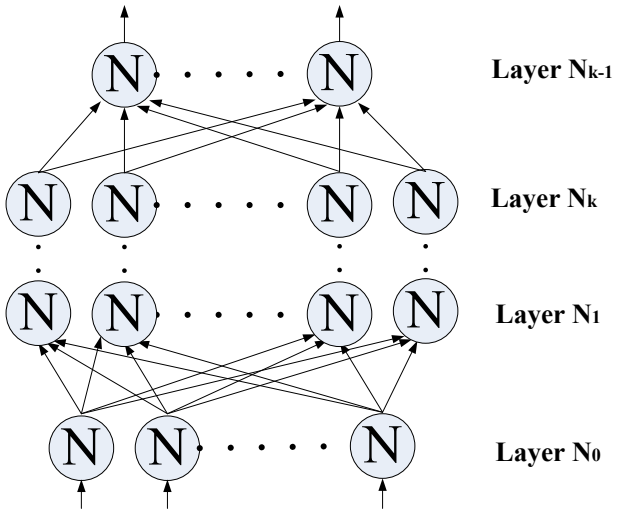


Figure 3.2 Multilayer neural network

neural network.

Assume the symbol x_k^{ij} and y_k^i stand for the input and output of neurons, where the subscript of k is the layer number and the superscript of i is the neuron number in this layer and j is the j^{th} input of this neuron. Suppose the ID for input layer is 0, then the forward process can be described as follows:

$$y_1^i = \sum_{j=1}^{N_0} w_{ji}^{01} \cdot x_j \ , \qquad i \in [1, N_1], \ \ j \in [1, N_0] \ . \tag{3.3.1}$$

$$x_k^{ij} = y_{k-1}^j \ , \ \ y_k^i = \sum_{j=1}^{N_{k-1}} w_{ji}^{(k-1)k} \cdot x_{ij} \ , \qquad k \in [2, N], \ \ i \in [1, N_k], \ \ j \in [1, N_{k-1}] \ . \tag{3.3.2}$$

where N_k is the total neuron number in layer k, and $w_{ji}^{(k-1)k}$ stands for the weight between the j^{th} neuron in layer $(k-1)$ and the i^{th} neuron in layer k, respectively.

There are two concept used in our proposed network architecture, including pipeline design and layer multiplexing design.

3.3.1 Pipeline Design

The multilayer neural network has a characteristic that the neuron in the layer depends on the neurons in the previous layer and there is no communication among neurons in the same layer. Fig. 3.3

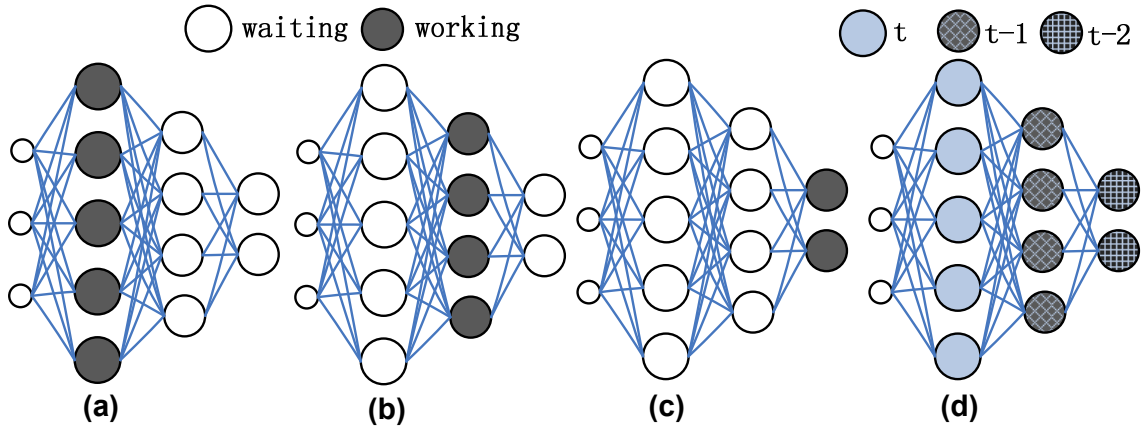


Figure 3.3 Non-pipeline (a), (b), (c) vs pipeline (d)

(a, b, c) shows an example of in the conventional non-pipeline forward phase, where only one layer is working while the other layers are just waiting for the data coming from the previous layer. In order to save the cost, we integrate the pipeline manner in the layer architecture when forwarding. The fundamental pipeline algorithm is described as follows, where t is the time factor given by a clock cycle.

$$y_1^i(t) = \sum_{j=1}^{N_0} w_{ji}^{01} \cdot x_j(t) \ , \qquad i \in [1, N_1], \ \ j \in [1, N_0] \ . \tag{3.3.3}$$

$$y_2^i(t) = \sum_{j=1}^{N_1} w_{ji}^{12} \cdot y_1^j(t-1) = \sum_{j=1}^{N_1} w_{ji}^{12} \cdot \sum_{k=1}^{N_0} w_{kj}^{01} \cdot x_k(t-1) \ . \tag{3.3.4}$$

$$y_3^i(t) = \sum_{j=1}^{N_2} w_{ji}^{23} \cdot y_2^j(t-1) = \sum_{j=1}^{N_2} w_{ji}^{23} \cdot \sum_{k=1}^{N_1} w_{kj}^{12} \sum_{m=1}^{N_0} w_{mk}^{01} \cdot x_m(t-2) \ . \tag{3.3.5}$$

As shown in Fig. 3.3(d) and Eq.(3.3.3)-(3.3.5), when the first layer is under computing for the input pattern at time t, the second layer is busy calculating the result which is transmitted from the first layer in the previous cycle $(t-1)$.

It doesn't need to wait for the end of some input pattern forwarding, but all the neurons in different layers are working simultaneously with different input pattern. By using the pipeline, the global forwarding speed would be much faster and there would not be much incidental cost or changes in the architecture, compared with a non-pipeline method. But just some modifications are needed, such as:

1) Enhance the clock rate for input memory module according to the pipeline depth.

2) Allocate a dependant memory for each neuron to store the weights.

3) Add a register to store the output of each neuron.

Unfortunately, it is hard to perform a whole neural network in pipeline manner by a popular FPGA chip due to limited FPGA resource. To solve this problem, we introduce the layer multiplexing into our pipeline method.

3.3.2 Layer Multiplexing with Partly Pipeline

Layer multiplexing was first proposed by S. Himavathi in 2007 [112], which aimed at reducing the resource requirement in a multilayer neural network. Instead of realizing a complete network, only the single largest layer with each neuron having maximum number of input is implemented. The layer plays a role of different layers with the help of a control block. The control block ensures proper functioning by assigning the enable signal, appropriate inputs and weights for each neuron.

This method presents an advantage that it can substantially save the resource so that a larger network could be implemented in a single FPGA chip. But this is achieved at the expense of the forwarding speed, because there is only one layer being working and it has to reconfigure the network before performing neural computing in the next layer.

According to the experimental synthesis report of implementing ANNs by layer multiplexing, the utilization rate of slices in an FPGA is not always so high, that is, we still have enough slice resource to do some improvement to make the forwarding speed higher. The main idea in this chapter is adding the partly pipeline manner to the layer multiplexing method.

In our proposed method, we first calculate the maximum number of neuron modules to fit an adopted FPGA chip. And then taking this value and the neural network topology into consideration, we can get an optimal solution by assigning the appropriate mapping method, pipeline depth and layer multiplexing. The optimal target is increasing the usage of chosen FPGA board to mapping more layers of ANN application to get high depth of pipeline, and then the high depth of pipeline will result high performance.

Fig. 3.4 gives an example of our method.

In this example, we suppose that the network topology is 3-4-2-3-1 and the maximum number of neuron modules in the FPGA is six. Numbers in the nodes in Fig. 3.4 mean the layer number. It performs the neural computing of adjacent two layers at a time with the pipeline manner between them. In Step1, the first two layers are under working, where the layer 1 is computing for the input pattern a^{m+1} and the layer 2 is serving for input pattern a^m, respectively. And in Step2, the 2^{nd}

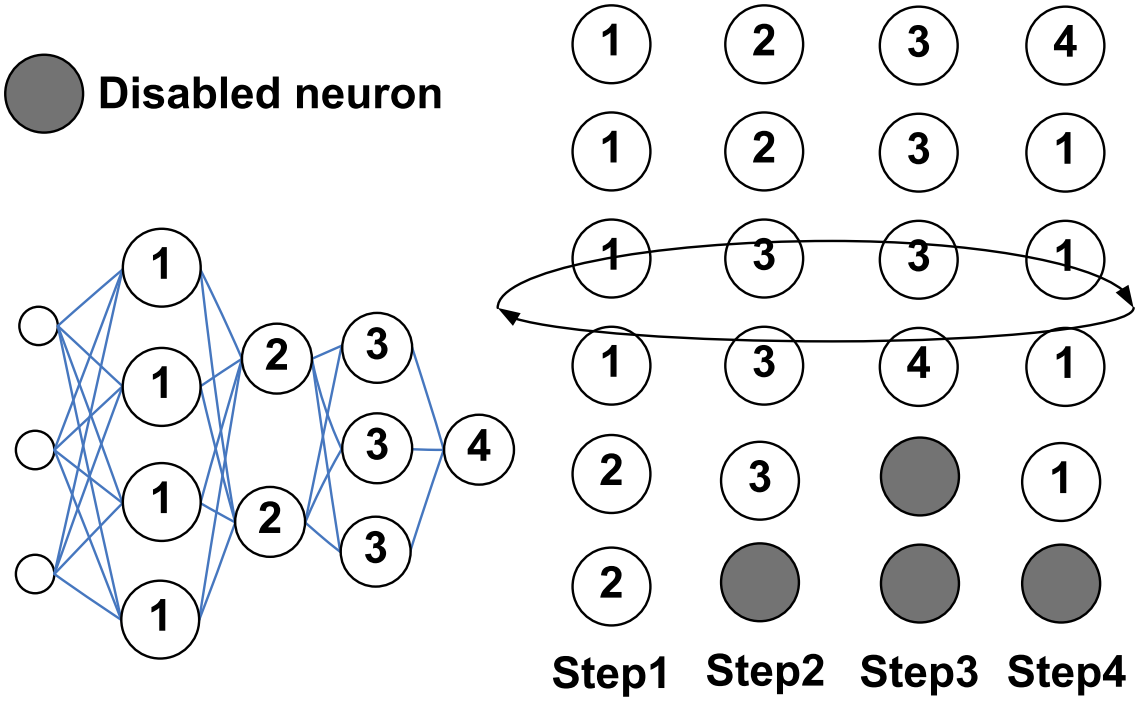


Figure 3.4 Schematic of our mapping method

and 3^{rd} layers are configured by the layer multiplexing mechanism, where the 2^{nd} layer is busy with input pattern a^{m+1} from the layer 1 in the previous step while the 3^{rd} layer is computing for input pattern am from layer 2 by layer multiplexing. And the total number of neurons in the 2^{nd} and 3^{rd} layers is five, so there are disabled neuron modules to save the power. The succeeding steps are almost the same as the first two steps, including enable the proper number of neuron modules, get the corresponding input from previous step, perform neural computing for the incoming data and then send the result to the next step by layer multiplexing. When the computation in Step4 is completed, it would turn to the Step1, forming a loop. In this example, there are always two layers mapping the neuron modules in pipeline, which means the pipeline depth is two. By assigning different FPGA chip and neural network topology, the pipeline depth may be different.

3.3.3 The Control Block Design

The operation of layer multiplexing and the partly pipeline are guaranteed by a control block, which is realized by the finite state machine (FSM). Fig. 3.5 shows the whole circuit of neural network using our proposed method. The details of the block named N_n of the n^{th} neuron is already presented in Fig. 3.1 in Sect. 3.2, so we will introduce the control block design here. The main task

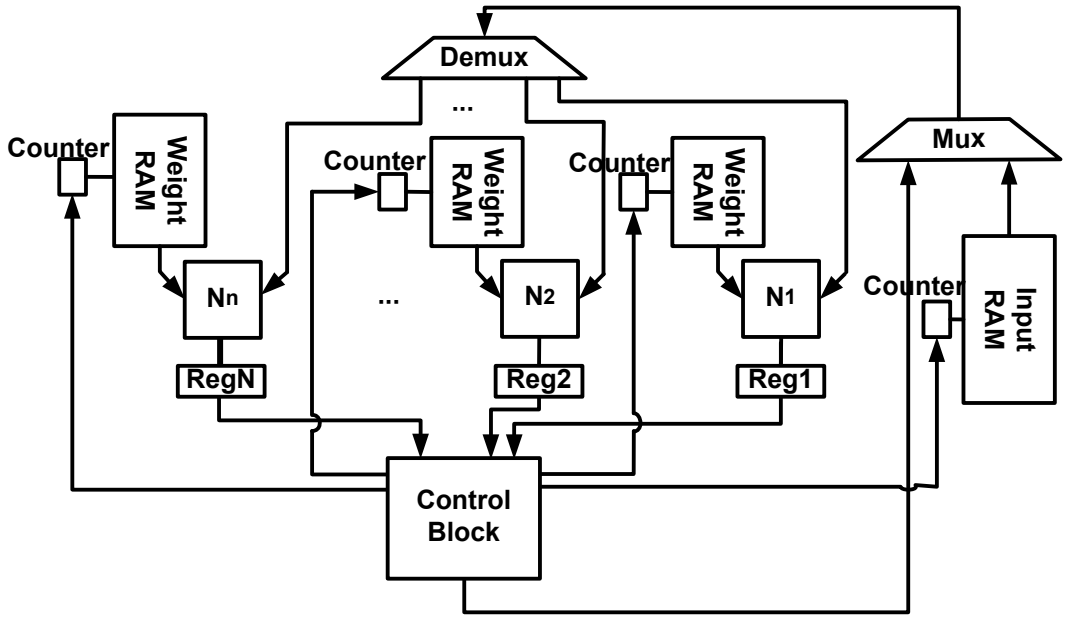


Figure 3.5 The neural network circuit

of the control block is assigning proper signals for the input and weight RAMs, the multiplexer and demultiplexer, and performing the logical data transmission. The pseudo code of the FSM is given in Fig. 3.6(a).

We also developed MATLAB program to determine the optimal architecture for a given neural network and a designated FPGA chip. Firstly, it estimates the number of slices used by a neuron module, given the data structure of the neuron. Secondly, the maximum amount of neuron modules allowed in the adopted FPGA chip can be figured out because we have integrated the information of several popular FPGA chip into the procedure. Then a searching function will be invoked to determine the optimal pipeline depth for the neural network. This procedure also provides the function that configures the parameters in the FSM. The pseudo code of the procedure is shown in Fig. 3.6(b).

3.4 Experiments

We use the Verilog HDL and Xilinx ISE 9.1i for design and synthesis. The simulation tool is performed by Modelsim XE II 7.3a. The FPGA chip we selected here is Xilinx VirtexII XC2VP20. The basic block of the VirtexII is the Logic Cell Block (LCB), which is composed of eight LUTs with four inputs, four slices, eight Flip-Flops, and so on [114].

```
FSM (reset, clk, flag1~S, en1~N,con0~N, mux, demux)
   // input signal:  reset, clk, flag1~S
   // flag1~S:  the finish signal of computation in Step s
   // en1~N: the enable signal for each neuron module
   // con0~N: the address signals for each RAM
   // mux: the selection signal for the multiplexer
  // demux: the selection signal for the demultiplexer
 always (current_state or flag1~S)
   case (current_state)
         IDLE:
               Do {…}
         Step1:
               Do { set en1~N, con0~N,mux,demux
                     next_state=Step2}
                       ……
         StepS:
               Do { set en1~N, con0~N,mux,demux
                     next_state=Step1}
        default: next_step=IDLE
   endcase
```

(a)

```
Solve(topology, chiptype, datapath)
 // the topology information includes the number
 // of layers, the number of inputs, the number of
// neurons in each layer
// the data path contains the data representation
//  of input, weight that mentioned in section II.B.
 1. begin
 2. initial pipe_depth=layer_num
 3. size=est_neuron_size( datapath)
 4. max_num=cal_max_num(chiptype,size )
 5. searching(pipe_depth, max_num, topology)
    6. { stop_condition()
    7. if (stop_condition())
    8.   searching(pipe_depth-1,max_num, topology)
    9. else
   10.   return pipe_depth
   11.  endif}
   12. config_FSM(pipe_depth, topology, data_path)
   13. end
```

(b)

Figure 3.6 Control block (a) Pseudo code for the control block (b) Procedure of topology generation

Table 3.1 Resource and performance of a neuron with different weight precisions

No. of bits	10	11	12	13	14	15	16
Slices	292	312	328	344	360	372	392
Max clk (MHz)	105.3	104.2	103.6	103.0	102.5	102.1	101.7

Table 3.1 gives the synthesis report for a single neuron module by varying the data path in weights. From the table we can see that with the increase of data precision, the maximum frequency comes down a little. And 16 bits are chosen in our simulation.

Table 3.2 shows the synthesis reports for two examples of neural network which mentioned in Sect. 3.3.2 and a pattern recognition application of ANN with topology of 5-7-3-7-5 [115]. The value of pipeline depth in the left column means the number of layers under execution at a time. Our proposed architecture is compact due to the simple module architecture and the effective control block, and also provides a flexible solution for a neural network to be implemented. Compared with other blocks, the neurons blocks take much more resource. This FPGA board used in this simulation could implement at most 20 neurons. Implementing the ANN with topology of 5-7-3-7-5 with 4 pipeline depth needs 22 neurons, so that this ANN could not be implemented with 4 pipeline depth. That is, the pipeline depth is decided by the selected FPGA board and the network topology.

Table 3.3 shows a comparison between the traditional layer multiplexing method (LM) and our

Table 3.2 Synthesis report for ANNs

Pipe Depth	Network architecture		3-4-2-3-1		5-7-3-7-5	
	Features	Resources	Utilized	%	Utilized	%
2	Slices	9280	2764	29.8	5399	58.2
	Flip Flops	18560	2148	11.6	4195	22.6
	LUTs with 4 input	18560	5128	27.6	10011	53.9
3	Slices	9280	4084	44.0	8375	90.2
	Flip Flops	18560	3400	18.3	6971	37.6
	LUTs with 4 input	18560	8036	43.3	16472	88.8
4	Slices	9280	4525	48.8	/[a]	/
	Flip Flops	18560	3819	20.6	/	/
	LUTs with 4 input	18560	9008	48.5	/	/

[a]/: not enough slice for mapping 4 pipe depth. The FPGA chip we selected here is Xilinx VirtexII XC2VP20

proposed method with different applications of ANNs [115, 112, 116, 117, 118, 119]. We have integrated the partly pipeline manner into the layer multiplexing (the number in 'proposed/' means the pipeline depth). There are several layers mapping the neuron modules by pipeline manner while LM maps only one layer at a time. The forwarding speed is measured by the interval starting from one input pattern imported to get the result at the output layer. The Connection-Per-Second (CPS) is the most common performance measurement, which is defined as the rate of multiplication and accumulation operations [26].

As a result, the forwarding speed of our proposed method is faster than that of LM with respect to the pipeline depth. Due to the pipeline manner, our method has much more neurons under working than LM, so it also shows an advantage in performance of CPS compared with LM method. The resource usage of the proposed method is increased compared with LM method, but the FPGA board need not be changed. It means that our method could provide high performance for the given FPGA board. We also observed that high pipeline depth will result higher benefit rate.

3.5 Conclusion

A general architecture for the implementation of a multilayer ANN was proposed. The circuit for each application can be easily generated by setting the parameter values to match the particular

Table 3.3 Performance comparison between layer-multiplexing and our method

Network Architecture	Implement method/ Pipeline Depth	Slices[a]	Utilization Rate (%)	Forward speed(us)	CPS (M)	Performance Benefit Ratio
3-4-2-3-1	LM	1726	18.6	1.84	15.8	3.278
	proposed / 4	4525	48.8	0.56	51.8	
5-7-3-7-5 [115]	LM	2979	32.1	2.84	39.4	2.472
	proposed / 3	8375	90.2	1.15	97.4	
8-5-5-3 [112]	LM	2142	23.1	2.28	35.1	2.373
	proposed / 3	5832	62.8	0.96	83.3	
8-5-5-5-5-3 [112]	LM	2171	23.4	3.7	35.2	4.014
	proposed / 4	8802	94.8	0.93	141.3	
4-12-1 [116]	LM	5051	54.4	1.92	31.3	2.521
	proposed / 2	5821	62.7	0.76	78.9	
5-12-8-4-1 [117]	LM	5051	54.4	3.56	53.9	1.130
	proposed / 2	8790	94.7	2.17	60.9	
4-10-1-10-4 [118]	LM	3870	41.7	3.14	31.8	1.368
	proposed / 2	6262	94.7	2.3	43.5	
4-7-13-1 [119]	LM	5464	58.9	2.88	45.8	1.747
	proposed / 2	8775	94.5	2.3	80	

[a]The FPGA chip we selected here is Xilinx VirtexII XC2VP20

network size and running the synthesis. Similarly to a particular network, the best solution of the architecture design of pipeline and layer multiplexing is calculated by MATLAB procedure, by returning the optimal pipeline depth and the control signal value in each state of the control block (FSM). We exploited the capability of a given FPGA board by assigning the proper pipeline depth, so that a higher resource utilization rate, global forwarding speed and high performance were achieved.

Our proposed architecture makes an FPGA implementation easy for a given ANN at a short time by varying the data path. It also provides the feasibility to perform a larger neural network in a popular FPGA board at a relatively higher speed by using the partly pipeline method. So it is possible to develop a neural device for commercial or industrial application by our method.

Chapter 4

A New Flexible Network on Chip Architecture for Mapping Complex Feedforward Neural Network

We propose a new flexible hardware Network on Chip (NoC), a packet-based signal processing architecture, for mapping the different applications of the feedforward artificial neural network (FF-ANN). There are many problems in a traditional FF-ANN implementation. For example, application is limited, interconnection is complex and data transmission is difficult to be controlled. This proposed NoC-based system can solve these problems, because it can be reconfigured and extended by sending the new packet. It can map the complex FF-ANN with multiple layers and multiple neurons in one layer. The system is designed to achieve low latency, high throughout and low power. The simulation results show that the proposed architecture can reduce communication load and increase connection per second (CPS) of system in real applications of FF-ANN.

4.1 Introduction

Feedforward artificial neural networks (FF-ANN) are widely used in numerous applications such as signal processing [120], anomalous detection [121], machine learning [122], system control [123], and various forecasts [124]. Software mapping methods of ANN have been researched for a long time, but they lack good performance. Hardware mapping methods using digital or analog architectures can achieve higher performance [26]. Furthermore, hardware ANNs are not always equipped with on-chip learning, because on-chip learning makes the circuit much more complex, increases power consumption and lowers performance. The analog mapping method is hard to be expansible for the neural network and has a low precision [125], whereas the digital mapping method is applied

widely due to high precision, good expansibility and plentiful electronic design automation (EDA) tools. There are many modern digital mapping architectures such as systolic array [48], single instruction multiple data (SIMD) [126], and FPGA [58]. However, they also have drawbacks such as no reconfigurability, high cost [127], low performance, heavy communication load, and limited applications. To overcome those drawbacks, a new digital mapping method is required.

In the last few years, a network on chip (NoC) appeared due to the continuous progress of the integrity of IC chip, where packet based network connections are adopted instead of point-to-point connections, and it is widely used in numerous applications afflicted by interconnection problems [128].

An NoC architecture is also discussed individually for mapping ANN to solve the existing problem. In previous paper [129], an NoC architecture was proposed for mapping the ANN with many layers and few neurons in each layer. In another paper [130], another NoC architecture was proposed for mapping a 2-hidden-layer ANN. However, their applicability is still limited. Therefore, a new NoC architecture was proposed [131]. In this chapter, this architecture is discussed for mapping the different applications of complex FF-ANN. To attain high performance and low power, our proposed NoC mapping method will also make sure of off-chip learning. Compared with other digital mapping method of ANN, the NoC mapping method makes ANN reconfigurable, extendable and flexible. The system has low communication load and high CPS performance of the system. Furthermore, it enables wide applicability at ANN. These advantages are due to high-level parallelism in NoC mapping, where wires in the links of the NoC are shared by many signals and all links in the NoC can operate simultaneously on different data packets [78]. NoC links can also reduce the complexity of wire design [68].

The rest of this chapter is organized as follows. The proposed mapping system is described in Sect. 4.2. In Sect. 4.3, experiments are explained, and the system is evaluated in Sect. 4.4. Finally, this chapter is concluded in Sect. 4.5.

4.2 Proposed NoC Mapping System

The neurons of general FF-ANN have the same architecture [132] as shown in Fig. 4.1. The purpose of hardware FF-ANN is to complete the computing work, and the off-chip learning is performed by software. The neurons of our proposed NoC FF-ANN have the same architectures and structures. Therefore, different applications of FF-ANNs can be implemented by the proposed NoC FF-ANN if every neuron executes the same task. However, if the neuron tasks are different in every

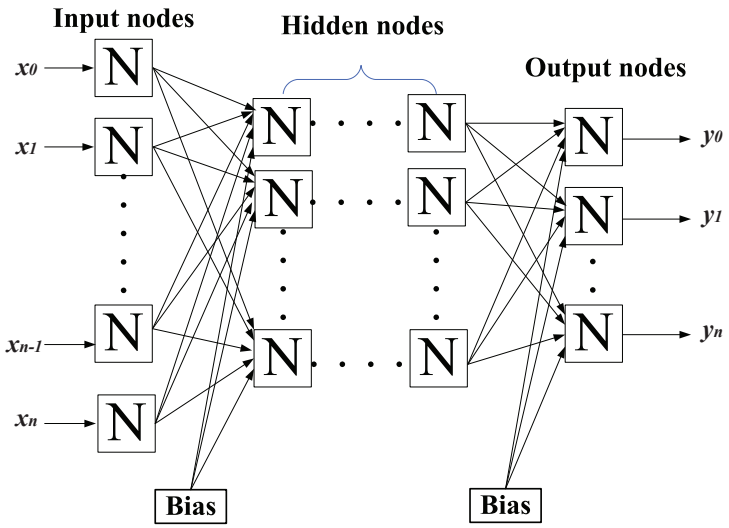


Figure 4.1 Structure of FF-ANN

layer such as in the RBF (Radial Basis Function) network, our proposed system cannot be applied. Design progresses in the following four steps: (1) design one neuron (Sect. 4.2.1); (2) aggregate four neurons in one processing element (PE) (Sect. 4.2.2); (3) design one router (Sect. 4.2.3); (4) design the system with PEs and routers (Sect. 4.2.4).

4.2.1 One neuron architecture

In FF-ANN, one neuron must transmit the computation result to every neuron in the next layer. The output y_j of neuron j is defined by [2]:

$$y_j = f(\sum_{i=0}^{n} w_{ij} x_i) \tag{4.2.1}$$

where w_{ij} is the weight value of the connection between neurons i and j, x_i is the input, n means the input number, and $f(\cdot)$ is an activation function. Well-known classical activation functions are used as follows:

$$Hyperbolic\,tangent\,sigmoid: \quad f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{4.2.2}$$

$$Logarithmic\,sigmoid: \quad f(x) = \frac{1}{1 + e^{-x}} \tag{4.2.3}$$

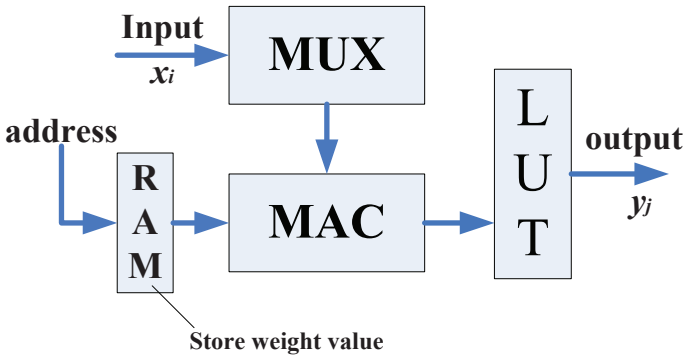


Figure 4.2 Structure of one neuron

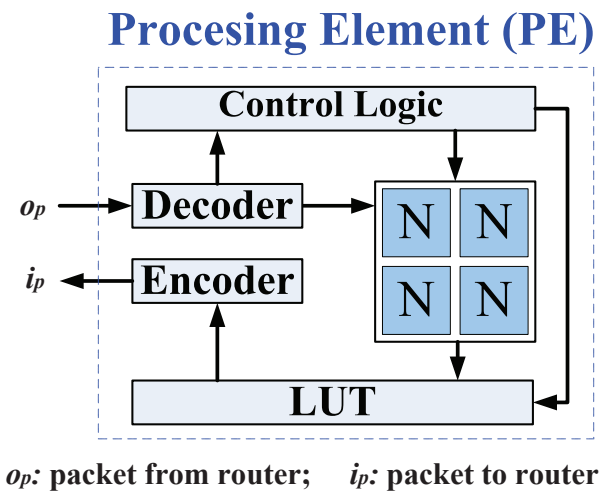


Figure 4.3 Structure of PE

Thus one neuron must contain four operations: $+, \cdot, \sum \cdot$ and $f(\cdot)$ [3]. Hence the MUX (multiplexer), MAC (multiply accumulate circuit), RAM (random access memory) and LUT (look-up table) are used to compose one neuron as shown in Fig. 4.2.

MUX is used to choose input, MAC is for multiplication and accumulation, RAM is used to store weight values, and LUT is for the expression of the activation function. We know that FF-ANN requires at least a 16-bit fixed point representation [133]. Therefore the data in our design for computation is $SIII.FFFFFFFFFFFF$, where S is a sign bit, I is an integer bit and F is a fraction bit. The 16-bit fixed point can cover the range of [-8.0, 8.0) with a quantization error of 2.44140625E-4.

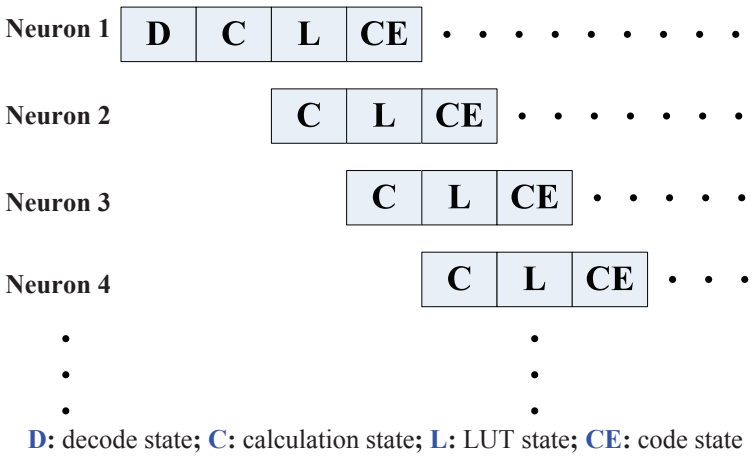


Figure 4.4 four-stages pipeline design of PE

4.2.2 Four neurons in one PE

To attain higher performance and to reduce both communication load and cost, four neurons in one PE work in parallel. The neuron number in one PE is decided by the area and speed. We knew that the area of 4 neurons is equal to 1 LUT and for the pipeline working, one neuron needs 4 cycles, and two neurons need 5 cycle, and so on. We assume the size of one neuron is 1, and there are x neurons in one PE. Implement y neurons need area of $(y/x * 4 + y)$ and time is $(4 + x - 1)$. So that the $area * time = (x + 7 + 12/x)y$. Thus, when the x equal to 3 or 4, the value of $area * time$ is best.

A PE architecture is shown in Fig. 4.3.A PE also requires a decoder, an encoder, a control logic, and LUT. PE decodes the neuron address to decide how many neurons (max is four, min is one) in this PE are used by a decoder. It requires two bits of the header packet as an indicator to distinguish the number of neurons used in PE. When one neuron is used, *"00"* is assigned to the two bits. Similarly, *"01"* is assigned when two neurons, *"10"* when three neurons, and *"11"* when four neurons are used. If no neuron is used in this PE, the packet does not transmit to this PE. The states of neurons are controlled by a control logic. The four-stage pipeline design is shown in Fig. 4.4. The states are the *decode state (D)*, *calculation state (C)*, *LUT state (L)*, and *code state (CE)*

In *D*, the whole system must be configured to satisfy a real application, that is, the number of PEs and neurons, weight values, activation function, and routing paths are decided for the application, which includes the following work: load the weight value into the RAM, load the Look-Up-Table of activation function to the ROM, and load the head packet for selected PEs. This state does not

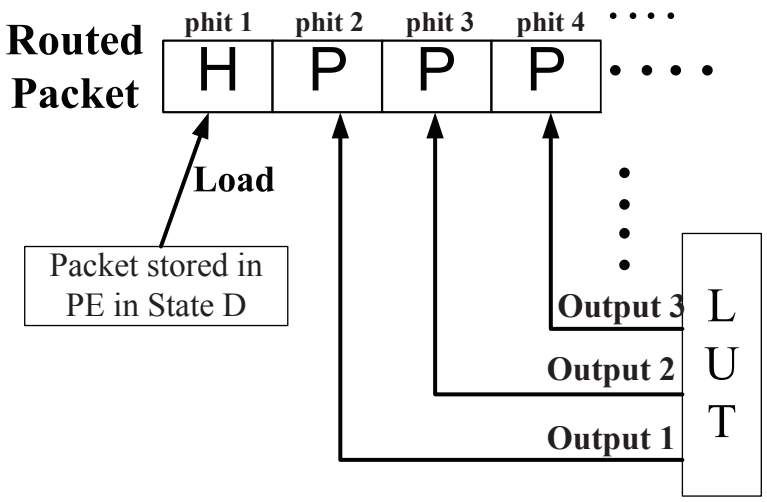


Figure 4.5 A single packet and outputs

appear anymore until the application will be changed. Then the chosen neurons begin calculations in *C*. When each neuron completes its look-up-table in *L*, the outputs of these neurons hold their output data as one single packet in *CE*. In *CE*, the packet that is loaded in *C* is used as part of the routed packet. The header phit is not changed, the payloads of this packet is used as the output of LUT, and each output corresponds to one payload, as shown in Fig. 4.5. Finally, the packet is sent to a router. The router is used to manage the transmission mechanism. Then the packet will be transmitted to all the neurons in the next layer via the router. This structure with 4 neurons in one PE can reduce the total number of transmit packets, communication load and cost, as described in Sect. 3.4.

4.2.3 Router design and packet architecture

The architecture of the proposed router for managing the transmission mechanism is shown in Fig. 4.6. This router has 5 input ports and 5 output ports that are connected to one PE and four routers. The proposed router consists of a buffer, MUX, an allocator, a shifter and a register. $i0$, $i1$, $i2$ and $i3$ in Fig. 4.6 mean input phits from four directions to the router. ii means input from PE. When phits arrive at this router, the virtual channel that has 5 First-In First-Out (FIFO) buffers is chosen, and then it is transmitted to four 5-1 MUXs to choose the output port decided by the allocator. The selected phits are then transmitted to a shifter. The shifter shifts 3 bits of header phits which are controlled by the allocator, while payload phits are not shifted.

Channels of our system transport are 18-bit-width phits of data per cycle. A 2-bit field is added

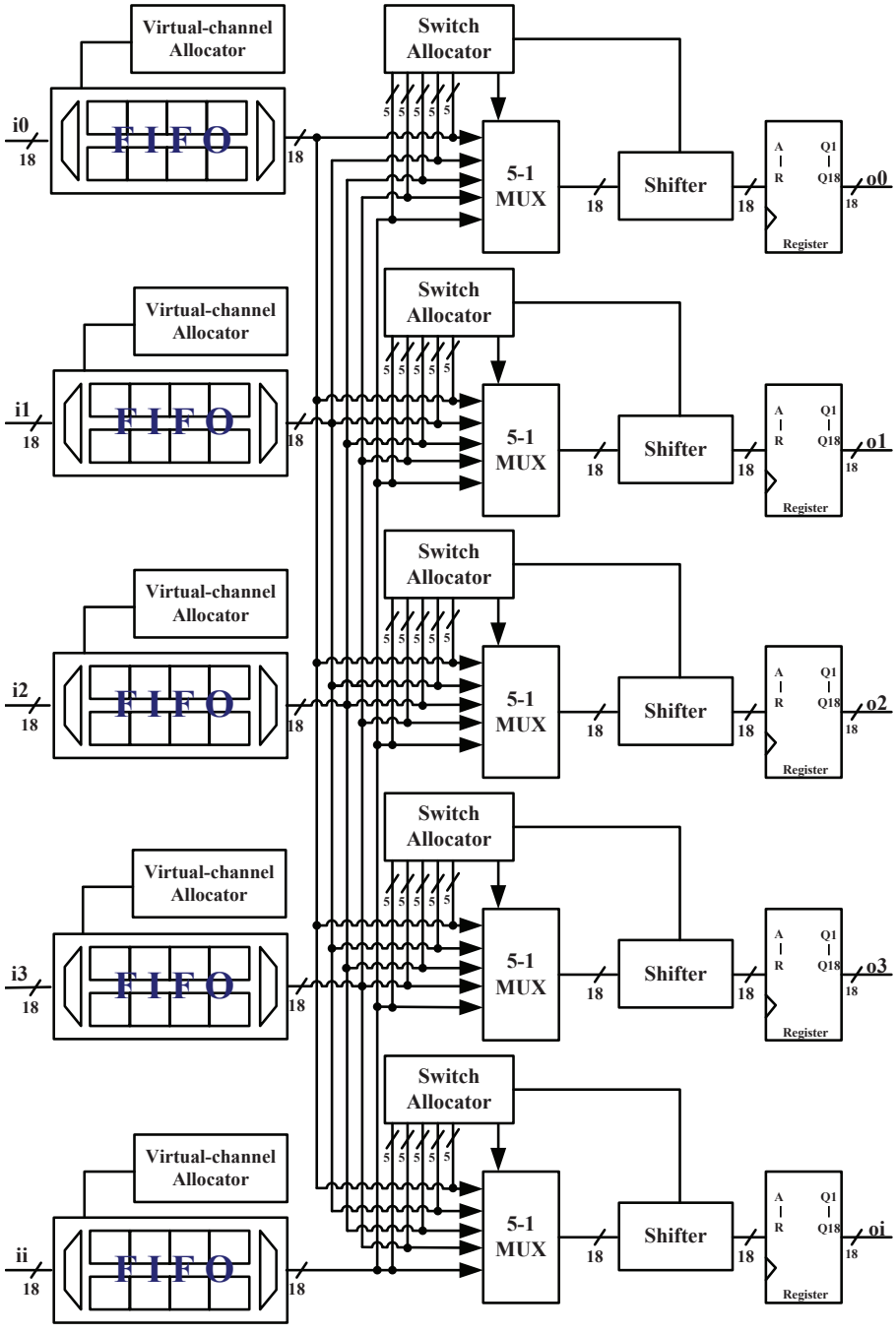


Figure 4.6 Block diagram of a router

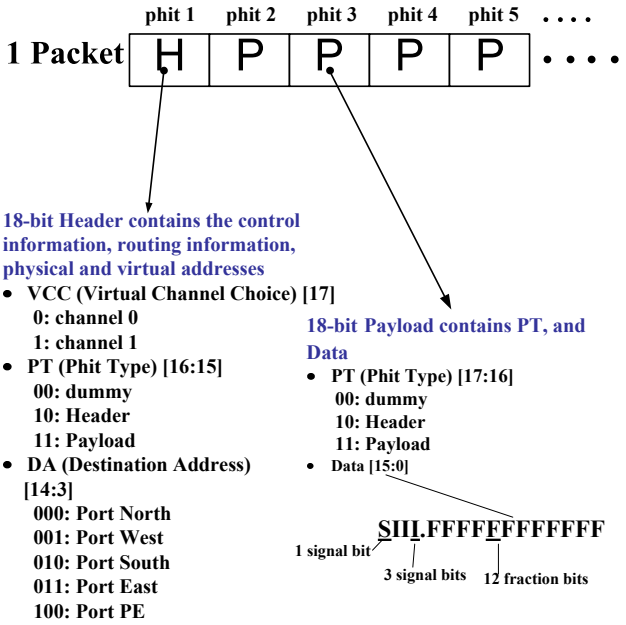


Figure 4.7 NoC packet format

to each channel to decode the type of phits (00 for dummy instruction, 10 for header and 11 for payload). The packet format is shown in Fig. 4.7. One packet contains one header and some payloads, and the payload number is decided by the neurons used in the former-layer PE. The header contains 2 bits for VCC (Virtual Channel Choice), 2 bits for PT (Phit Type), and 3 bits for each DA (Destination Address). The payload contains 2 bits for PT and 16 bits for data.

4.2.4 System design

The complex FF-ANN with 9 layers and 8 neurons in each layer will be used as an example to explain the proposed NoC mapping system shown in Fig. 4.8. This system does not include the on-chip learning circuit; it is used only for data computation and transmission.

In this figure, R is a router connected to a PE. The rectangle in the top left corner composed of two PEs is an input layer, and in this part, 8 input-layer neurons are connected with two routers. The rectangle in the center composed of two PEs is an output layer, and in this part, 8 output-layer neurons are connected with 2 routers. The remaining rectangles in the network correspond to hidden layers, and in every hidden layer, 8 hidden-layer neurons are connected with two routers.

The transmission order in this network is shown in the top right of Fig. 4.8. Data transmission from the input layer to the output layer via hidden layers is processed in clockwise order. By this

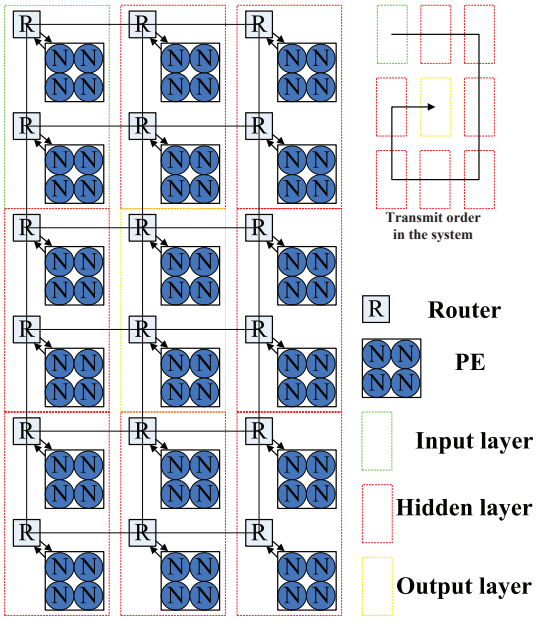


Figure 4.8 NoC architecture for mapping complex FF-ANN

order, we can reduce the number of hops in backward computation.

Two-dimensional (2-D) mesh topology is used in this chapter. It is a well-known topology. There are many routing algorithms for this topology. The XY routing algorithm is chosen in this chapter, because it suits our system well, and in the general case, the XY routing algorithm has low latency and high performance (details in Sect. 4.3.) When the system is larger and more complex (more layers and more neurons in one layer), we should extend the size of the network and use more advanced topology.

Some real applications of FF-ANN, such as pattern recognition [115], neural control [117] and nonlinear principal component analysis (NPCA) for image processing [118] are used as examples to introduce more detail to the mapping method with NoC architecture. The resultant systems are shown in Fig. 4.9.

The application of pattern recognition is mapped as shown in Fig. 4.9(a). This FF-ANN has 5-7-3-7-5 neurons in each layer. To map the input layer with 5 neurons, 4 neurons of one PE and 1 neuron of another PE are needed, and 2 routers are connected to these 2 PEs to compose the input layer. To map the first hidden layer with 7 neurons, 4 neurons of one PE and 3 neurons of another PE are needed, and 2 routers are connected to these 2 PEs to compose the first hidden layer. The second hidden layer, third hidden layer and output layer are also used in this mapping method. When each

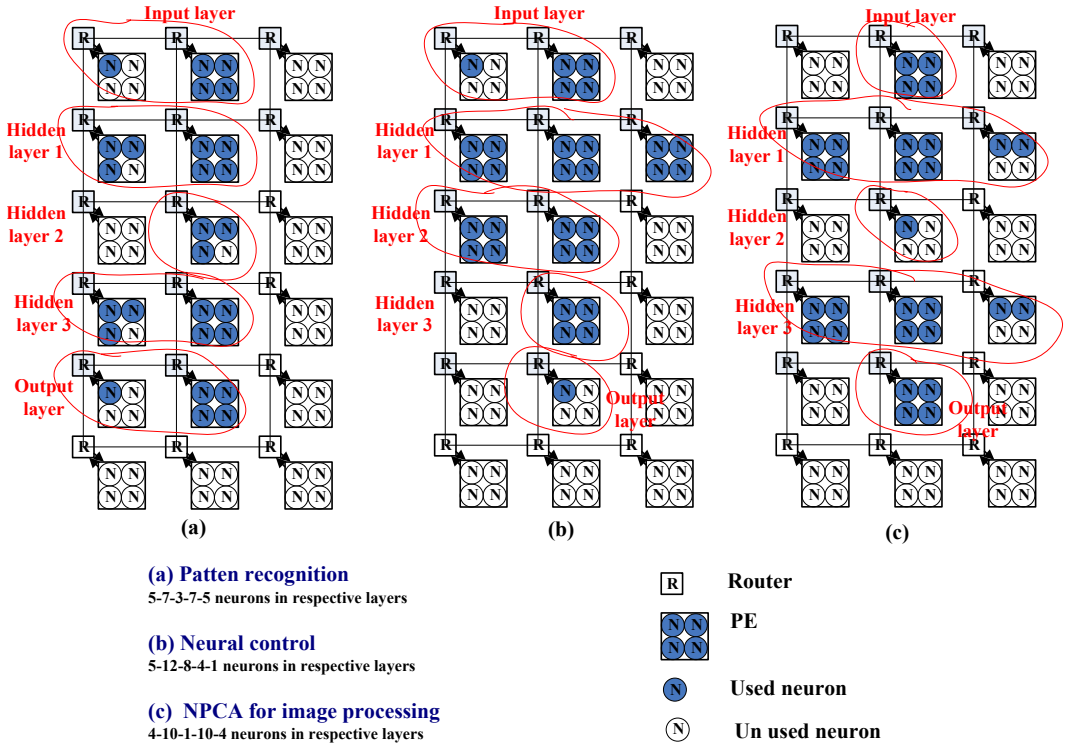


Figure 4.9 NoC architectures for mapping 3 applications of FF-ANN

PE of the input layer completes its calculation task, the packet is transmitted to two PEs of the first hidden layer via routers. The PEs of the first hidden layer, the second hidden layer and the third hidden layer must transmit the packet, similarly to the PEs of the input layer. This mapping method is flexible owing to the NoC architecture, for example, the input layer can be mapped in the first line with 2 routers. It also can be mapped in the first column with 2 routers, and can be mapped in any place of this network. In contrast, a different mapping may result in a different performance. We require mapping of the neighbor layers as close as possible, so that the communication load and system running time can be reduced.

4.3 Examination of Proposed System

When designing the NoC, a good routing algorithm results in a good performance. Modern routing algorithms such as dyad T (DT), Fully-Adaptive (FA), Negative-First (NF), North-Last (NL), Odd-Even (OE), West-First (WF), and XY are compared for suitability to the NoC mapping system design using the NOXIM NoC simulator [134]. The results of total delay, energy, average delay and

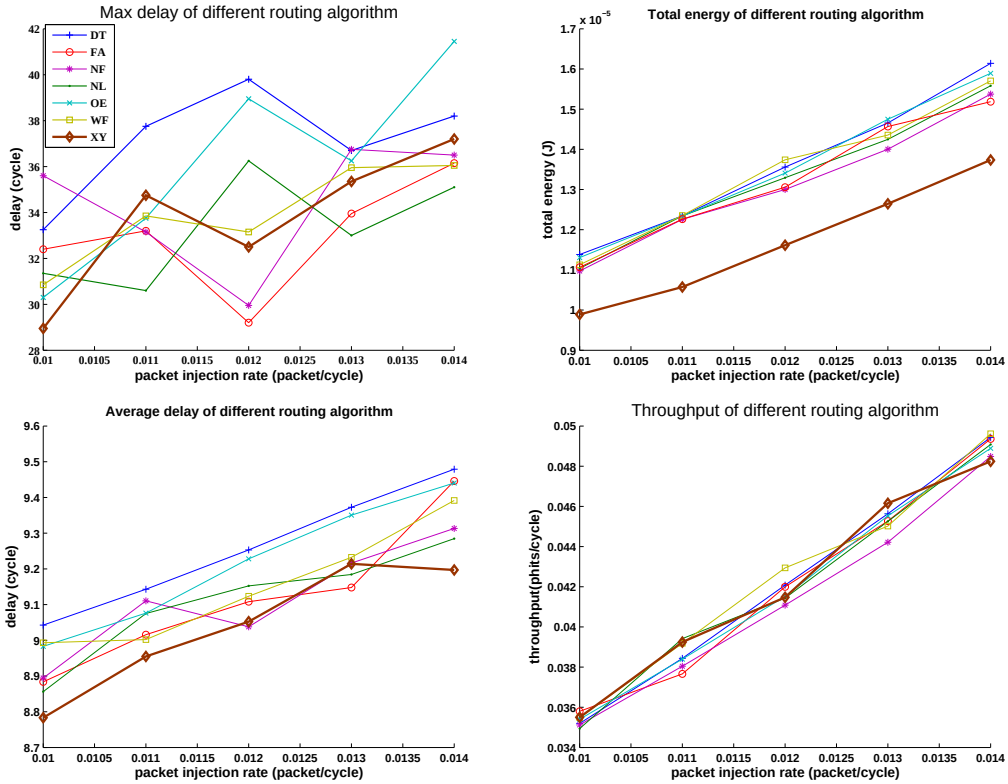


Figure 4.10 Comparison of total delay, energy, average delay and throughput

throughput are compared in Fig. 5.10.

From Fig. 5.10, the XY routing algorithm shows the lowest total energy and average delay, whereas its max delay and throughput is almost the same as those of other routing algorithms. On the basis of its low latency, high throughput and low power consumption, the XY routing algorithm is suitable for the NoC mapping system.

4.4 Evaluation

In this section, the features of the proposed NoC mapping system are introduced. Then the system is evaluated in terms of communication load, performance of CPS (connection-per-second), latency, throughput, and power consumption.

4.4.1 System reconfigurability

This system is reconfigurable, because the weight values and activation function values are stored in RAM; they can be changed as desired. The topology and routing algorithm can be also changed by sending new packets.

4.4.2 System extensibility

This system can be easily extended. Because one neuron is first designed and then extended to one layer, when we design the neural network. All neurons are the same. In the same way as that one, when we design the NoC architecture, one router is designed first and then connected to PEs to compose the whole system, where all routers are the same. Hence, every part of this system is designed in the style of cell by cell. Cells can be easily added or removed for different applications. This system can be used for not only a complex system but also a simple one. In particular, different FF-ANNs have different layer numbers, neuron numbers and algorithms. All of them can be easily changed. Thus, we simply use a different number of routers to connect with PEs, in order to implement the whole system.

4.4.3 Reduced communication load

Our system shows a great reduction in communication load. The three applications shown in Fig. 4.9 are used to introduce the difference between the traditional point-to-point (P2P) neural network and our NoC neural network. In the traditional P2P neural network system [2], one neuron must transmit n packets to all neurons in the next layer. In our NoC neural network, after a neuron finishes its computation, it executes the activation function from the LUT. The 4 neurons in one PE of the input layer assemble their output into a single packet, and communicate it to the router. Then data is transmitted from this router to other routers in the next layer. Thus the number of packets of the proposed NoC architecture is less than that of the P2P neural network system. The packet size of NoC is larger than that of the traditional P2P neural network system. The comparison between P2P and NoC for three applications are shown in Table 4.1, supposing that one neuron must transmit one packet to the next layer. From the result, the communication load of the total packet size of the proposed NoC neural network is 1.9$\sim$2.59 times less than that of the traditional P2P neural network.

Table 4.1 Comparison of communication load

Application	Architecture	Packet number	Packet size (bit)	Total (bit)
1. Pattern recognition	P2P	112	16	1792
	NoC	12	36/72/90	**882**
2. Neural control	P2P	192	16	3072
	NoC	15	36/90	**1188**
3. NPCA	P2P	100	16	1600
	NoC	12	36/54/90	**842**

Table 4.2 Simulation environment for NIRGAM

OS	Fedora 12. Linnux 2.6.31.5-i686
CPU	Intel Core i7 @3.33GHz
RAM	3G

4.4.4 Performance measured by connections per second (CPS)

The NIRGAM NoC simulator [135] is used to evaluate the latency and throughput of the proposed system. Table 4.3 shows parameters of NIRGAM used in our experiments. The average latency per flit and the average throughput are shown in Fig. 4.11. In this figure, the red bar means the amount of communication on the southward channel of the router, the blue bar means that on the northward channel, the green bar means that on the eastward channel and the yellow bar means that on the westward channel. Experimental results show that the total latency of the system is 62.00 per flit (clock cycles), the system time is 0.563us and throughput is 47.82Gbps. The simulation environment is shown in Table 4.2. The CPU time of using NIRGAM NoC simulator for three applications are 8.6s, 8.7s, 8.6s, respectively.

The proposed NoC mapping system is compared with other digital ANNs [1], which are mapped by the VLSI techniques. Table 4.4 shows a speed performance measured in terms of the value of Connection-Per-Second (CPS), which is the most common performance rating. The proposed NoC mapping system has the best CPS. And compare with a newest world's best software based ANN [136] which running with Pentium II, the CPS is just 42M.

Compared with the other bus controlled digital FF-ANN, the proposed NoC architecture can achieve high level parallelism in which the wires in the links of the NoC are shared by many signals and all links in the NoC can operate simultaneously on different data packets.

The average latency per flit of three real applications is shown in Fig. 4.12. Their CPS values are shown in Table 4.5. Compared with some available high-performance digital ANNs, the proposed

Table 4.3 NIRGAM parameters in our experiments

Parameter	Value or type
TOPOLOGY	MESH
NUM-ROWS (row number)	6
NUM-COLS (column number)	3
RT-ALGO (routing algorithm)	XY
NUM-BUFS (buffer number)	5
FLITSIZE (flit size in byte)	11.25
HEAD-PAYLOAD (head size in byte)	2.25
DATA-PAYLOAD (payload size in byte)	9
WARMUP (traffic begin)	10
SIM-NUM (simulation running)	800
TG-NUM (traffic end)	300
CLK-FREQ (clock frequency)	0.1GHZ

Table 4.4 Comparison of CPS [1]

Name	Structure	Precision	Neurons	CPS
MD-1220	FF	1-16b	8	9M
NLX-420	FF	1-16b	16	300
Lneuro-1	FF	1-16b	16PE	26M
N6400	SIMD	1-16b	64PE	870M
HNC 100	SIMD	32b	100PE	250M
MA-16	Matrix	16b	16PE	400M
MT19003	FF	12b	8	32M
WSI NAP	SIMD	9b×8b	576	138M
NoC ANN	NoC	1-16b	18PE,72	1351M

NoC architecture has higher CPS. On average, NoC CPS values are 3.68, 2.76 and 3.68 times better than those of N6400, HNC100 and MA-16, respectively.

4.4.5 Power consumption

Our NoC neural network was designed by Verilog and implemented on FPGA using Alter Quartus II to evaluate power consumption. The clock frequency was set to 0.1GHZ, similar to the NIRGAM NoC simulator. This system was implemented on StratixII EP2S60F1020C3 [137]. The results are reported in Table 4.6. Power consumption data of other existing hardware ANNs in Table 4.4 was not available.

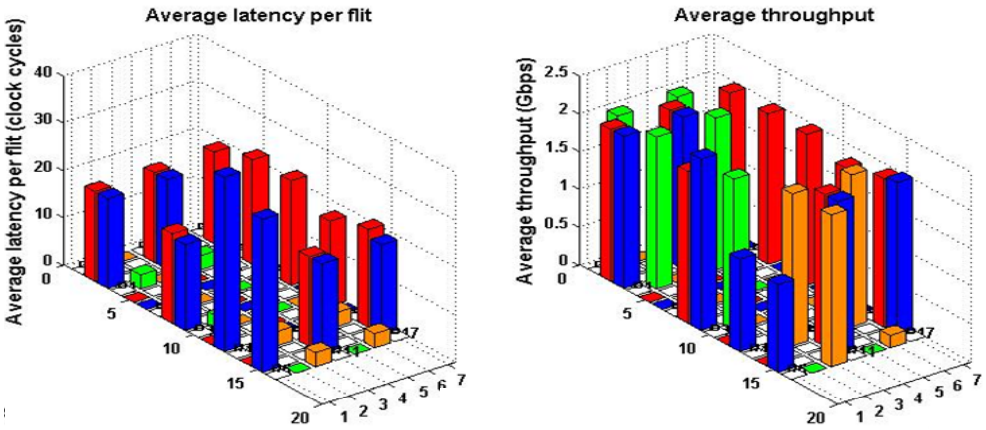


Figure 4.11 Average latency per flit and average throughput

Table 4.5 Comparison of CPS with existing systems and proposed system

	Pattern recognition	Neural control	NPCA
MD1200	/ [a]	/	/
N6400	367M	408M	394M
HNC100	67.5M	75M	72.5M
MA-16	135M	150M	145M
NoC ANN	1350M	1125M	1450M

[a] / means 'not available'

4.5 Conclusions and Future Work

A sophisticated NoC architecture with off-chip learning was proposed to satisfy various applications of the complex feedforward neural network. We designed this system aiming at low latency, high throughput and low power consumption. This system is reconfigurable, because the weight values and activation functions can be changed as desired. We can also change the topology and routing algorithms of the NoCs by sending new data to meet different kinds of feedforward neural networks, so this system is easily extended. We can design this system in the style of cell by cell and can easily add or remove any cell to comply with different applications. The proposed NoC system can reduce the communication load of total packet size and improve the system performance of CPS. This proposed NoC mapping method can make the digital ANN more efficient.

Our future works are to design an on-chip learning circuit on the NoC-based FF-ANN and to design a sleep-detection model for the NoC system to reduce power consumption.

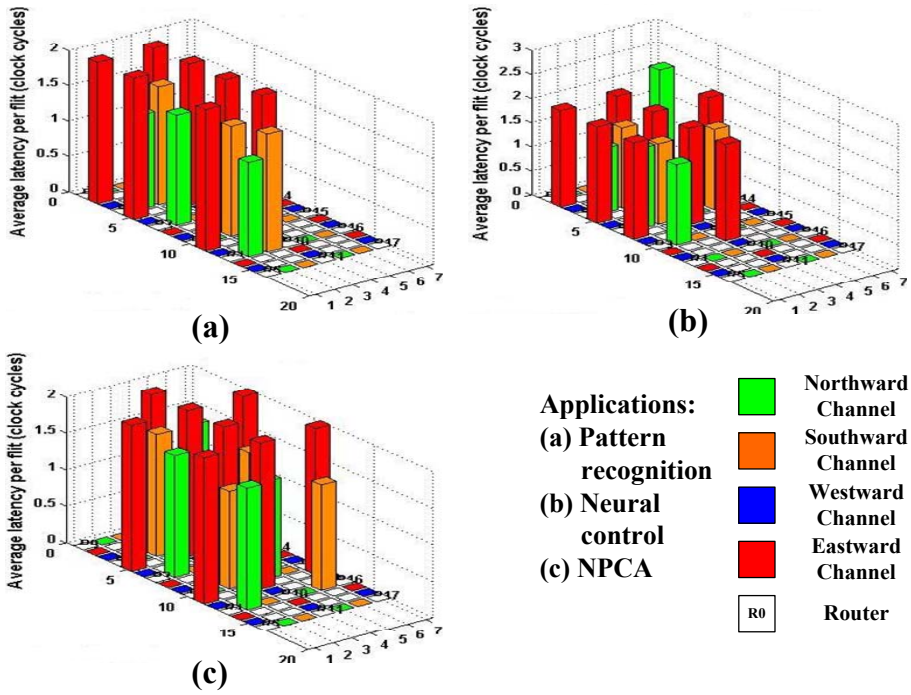


Figure 4.12 Average latency per flit in pattern recognition application, Neural control application and NPCA application

Table 4.6 Results of FPGA

Family	Stratix II
Device	EP2S60F1020C3
Frequency	100M
Register	2863
ALUT	4563
Static power dissipation	612.95mw
Dynamic power dissipation	623.70mw

Chapter 5

Multiple Network-on-Chip Model for High Performance Neural Network

We have proposed a Network on Chip (NoC) for ANN, and this architecture can reduce communication load and increase performance when an implemented ANN is small. In this chapter, a multiple NoC models are proposed for ANN, which can implement both a small size ANN and a large size one. The simulation result shows that the proposed multiple NoC models can reduce communication load, increase system performance of connection-per-second (CPS), and reduce system running time compared with the existing hardware ANN. Furthermore, this architecture is reconfigurable and reparable. It can be used to implement different applications of ANN.

5.1 Introduction

Hardware implementation methods for high performance Artificial Neural Network (ANN) have been an active field of research since 1990 [1]. Feedforward Neural Network (FF-ANN) is one of widely used ANN to solve a lot of real problems, such as pattern recognition, prediction, optimization, and so on [115, 2, 113]. The existing hardware implementation method is widely used for mapping FF-ANN. The hardware implementation method could be classified into digital one and analog one. The digital hardware implementation method was discussed deeply due to high precision, good expansibility and a good design support by EDA tools [26]. The common digital implementation architectures include systolic array architecture, slice architecture, Single Instruction Multiple Data (SIMD), and so on [50, 138, 139]. A Point-To-Point (P2P) data transmission is usually used by the digital implementation method. Whereas, some drawbacks exist as follows. (1). It is not reconfigurable, that is, only one application can be implemented by the special hardware

architecture. (2). Higher performance is required (real time control and other high performance applications are required). (3). Complex interconnection problem should be solved (communication load with P2P data transmission method is too heavy) (4). Not flexible (when one of neurons or one of link is wrong, the whole network may be useless). Thus, a new architecture with data transmission method is required to solve or relieve these problems.

The Network on Chip (NoC) architecture with a packet based data transmission method [78, 68] could be expected as a good way out. NoC is a new approach to design the communication subsystem of System-on-Chip (SoC). It is always considered by researchers to solve communication and performance problem for the P2P bus connection based MPSoC [140, 128].

In this chapter, a multiple NoC Model with 5-port 2-virtual channel wormhole-switched router is proposed for high performance Neural Network. The multiple NoC Models include two models: model-1 is based on the former work [141, 142], all the layers of ANN can be implemented with it in one time, thus it can be suitable for ANN with small network size. Model-2 will use the same NoC architecture, whereas the implementation method is different. In this model, different layers of ANN will be implemented with NoC architecture one by one, so that it is appropriate for ANN with large network size. We forecast that the following drawbacks can be overcame by the proposed system. (1). Complex interconnection problem could be relieved by replacing P2P connection method by packet based data transmission method. (2). Performance could by increased by the proposed new NoC implementation method. (3). Limited application problem could be solved by the proposed multiple NoC Models (model-1 for ANN with small network size and model-2 for ANN with large network size). Furthermore, both of two models are reconfigurable. (4). Not flexible problem could be solved due to reconfigurable NoC architecture which the placement of neurons and links are not fixed, so that the faulty one can be replaced by others. These improvements of hardware ANN are owed to high level parallelism of the NoC architecture and the packet based data transmission method [75].

The rest of this chapter is organized as follows. Section 5.2 shows the architecture of multiple NoC models and a switch control in this work. Section 5.3 shows the measurement results for supporting the low power design method of NoC. Section 5.4 shows the measurement results which compared with existing hardware FF-ANN. Section 5.5 shows the conclusion.

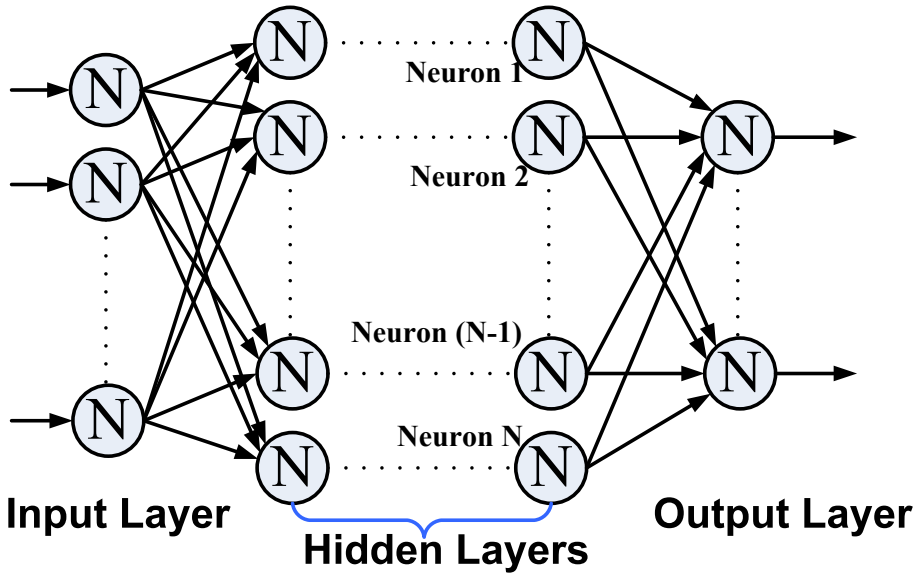


Figure 5.1 General structure of FF-ANN

5.2 Architecture of Multiple NoC models Implement for FF-ANN

Fig. 5.1 shows the general structure of FF-ANN. As shown in Fig. 5.1, neurons in different layers have the same architecture. Therefore, different applications of FF-ANNs consist of different numbers of layers and neurons. For implementing FF-ANN, the design steps of proposed multiple NoC models are as follows: 1. one neuron is designed; 2. four neurons are aggregated as one PE; 3. one router is designed; 4. proposed multiple NoC models are designed by connecting PEs and routers. The step one and the step two are almost same as our previous work [141, 142], and they will be described briefly, while the step 3 and the step 4 will be described in details.

5.2.1 Structure of a single neuron

Neuron computing need to contain four operations: addition, multiplication, multiplier-accumulator, and function [129]. Thus one single neuron consists of MUX (Multiplexer), MAC (Multiply Accumulate Circuit), RAM (Random Access Memory) and LUT (Look-up Table) as shown in Fig. 5.2.

In Fig. 5.2, inputs were chosen by MUX; multiplication and accumulation were realized by MAC; weight values were stored in RAM; activation function was expressed by LUT. At least, 16 bits fixed point representation was required by FF-ANN [58], therefore the 16 bits data for neuron computing in this work consists of one sign bit, three integer bits and twelve fraction bits. It can

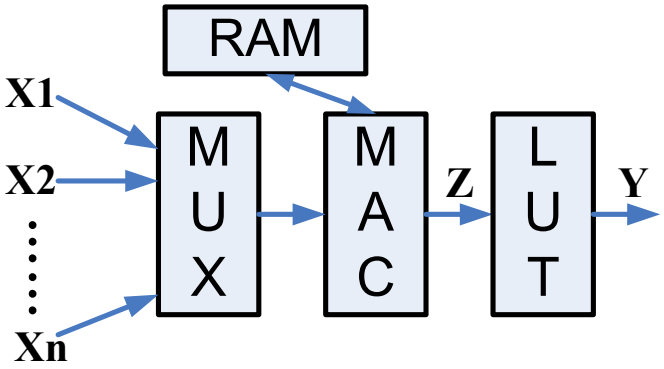


Figure 5.2 Single neuron architecture

cover the range of [-8.0, 8.0) with a quantization error of 2.44140625E-4.

5.2.2 Four neurons aggregated in one Processing Element (PE)

Each of 4 neurons in the same layer are aggregated in one Processing Element (PE) and connected to one router for reducing total transmission packet, communication load and cost. A decoder and a control logic are also required to be consisted of the PE. When the data is transmitted from a router, a decoder decodes the 4-bit neuron address for choosing the neurons which will be used. For example, ”1100” means the fourth neuron and the third neuron will be used. This design can make system flexible, and it is easy for users to choose the neuron which they want. Control logic consists of counters and flag registers for controlling the RAM using virtual address. Weights are stored in RAM. When each neuron in the same PE completes its calculation task, the outputs of them hold output data as one single packet, and then the packet is sent to a router.

5.2.3 5-port 2-virtual channel router architecture

One packet consists of two 18-bit header flits and some 18-bit payloads flits for data transmission. The first two bits of each flit are used for decoding the type of flits which ”00” for header, ”01” for payload and ”11” for null. Header contains the control information, routing information, physical and virtual addresses. Payload contains the computing result of each neuron.

For implementing the architecture of FF-ANN and managing the data transmission of the FF-ANN, a new 1GHz 5-port 2-virtual channel wormhole switched router with 18GB/s bandwidth is designed for our proposed system. For reducing the time management of data in the router, a ”multi-port chosen” design idea is proposed. The ”multi-port chosen” means multiple output ports can be

chosen at the same time by just one input header packet. The router architecture is shown in Fig. 5.3, where five input ports and five output ports are connected with routers of four directions and one PE. This proposed router consists of buffers, MUXs, allocators, shifters and registers. i0, i1, i2 and i3 mean input ports of four directions which are connected with this router. i4 means input port for PE. The virtual channel consists by 5 First-In First-Out (FIFO) buffers. When packets arrive at this router via one of five input ports, it is transmitted via the selected virtual channel, and then it is transmitted to 5-1 MUX to decide the proper output port by Switch Allocator (SA). Each SA checks three bits of input packet which will be introduced later in detail. The selected packets are then transmitted to a shifter. The shifter shifts 5 bits of header which is controlled by SA, while payload is not shifted.

The architecture of SA0 is shown in Fig. 5.4. Each SA will check three bits of input phits. The first two bits come from upper two bits (the seventeenth bit and the sixteenth bit) of input phits, and then control the shifter. It is used for deciding the type of phits. The last one bit of different SAs are different which SA0 checks the fifteenth bit, SA1 checks the fourteenth bit, SA2 checks the thirteenth bit, SA3 checks the twelfth bit, and SA4 checks the eleventh bit. In Fig. 5.4, the SA consists of three parts: decoder, arbiter and hold logic. In the decoder part, three bits of each input phit is decoded. Two bits are used for partition the phits type, and one bit for choosing output port. The hold logic part is designed to hold the selected port for the payload.

5.2.4 Design for multiple NoC models

The architecture of multiple NoC model is shown in Fig. 5.5. It consists of routers and PEs. One router is attached to one PE, and routers are connected with each other. PEs are communicated via routers. In the NoC design, topology is very important. Different topology can be suitable for different applications which make the system lower latency and higher bandwidth requirement [68]. At the same time, the power consumption may also be reduced. The torus topology has one more direct communication channel between the first router and the last router in each line. It is proper for the complex communication applications, such as FF-ANN. (The comparison will be discussed later.) According to [1], at most 64 neurons in one layer are used by common hardware ANN. Thus a 4x4 2D torus topology is proposed for our system design.

The multiple NoC models are listed as follows:

- **Model-1**: the whole FF-ANN (all the layers) will be implemented by the proposed 4x4 2D torus NoC architecture in one time.

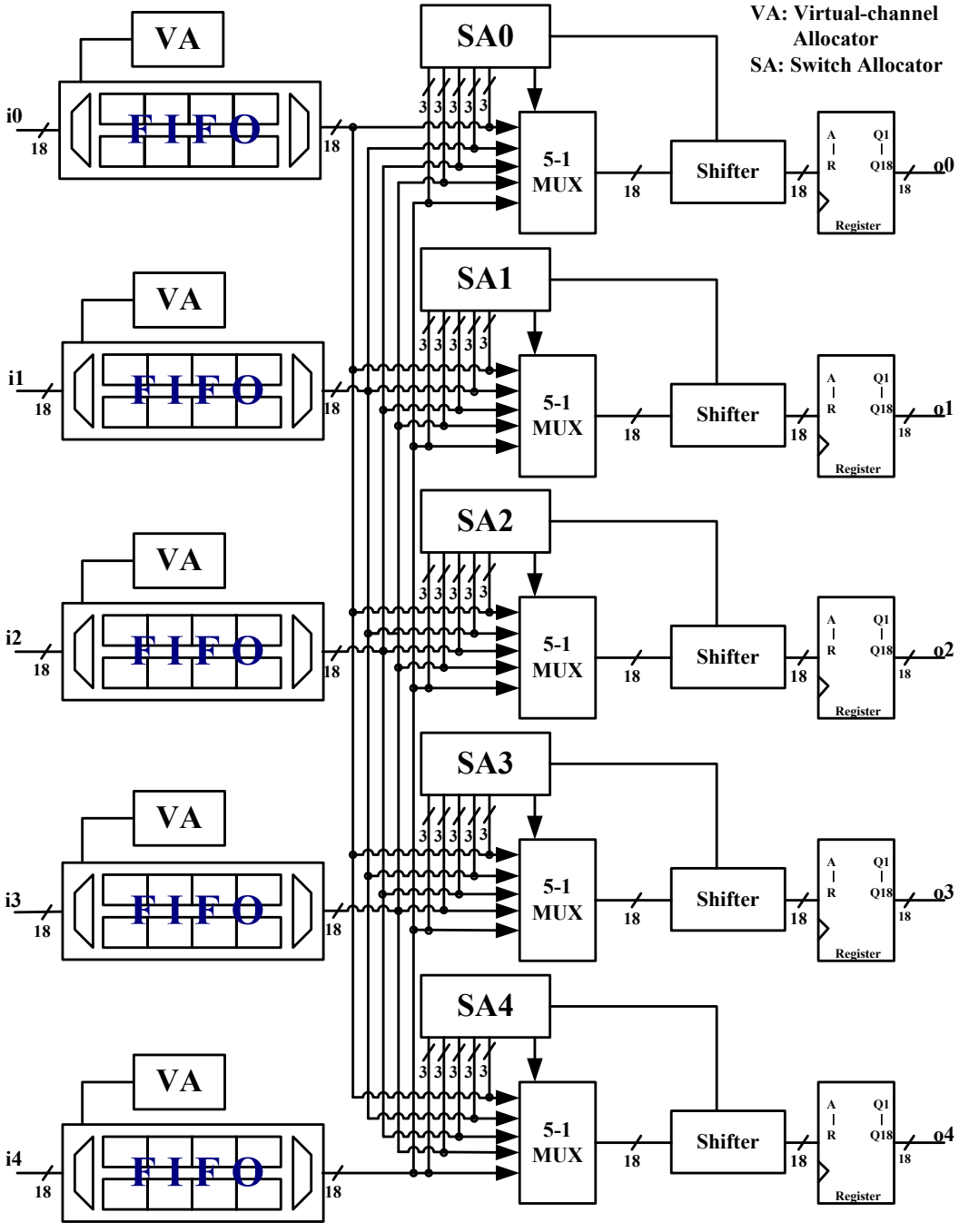


Figure 5.3 Block diagram of a router for FF-ANN

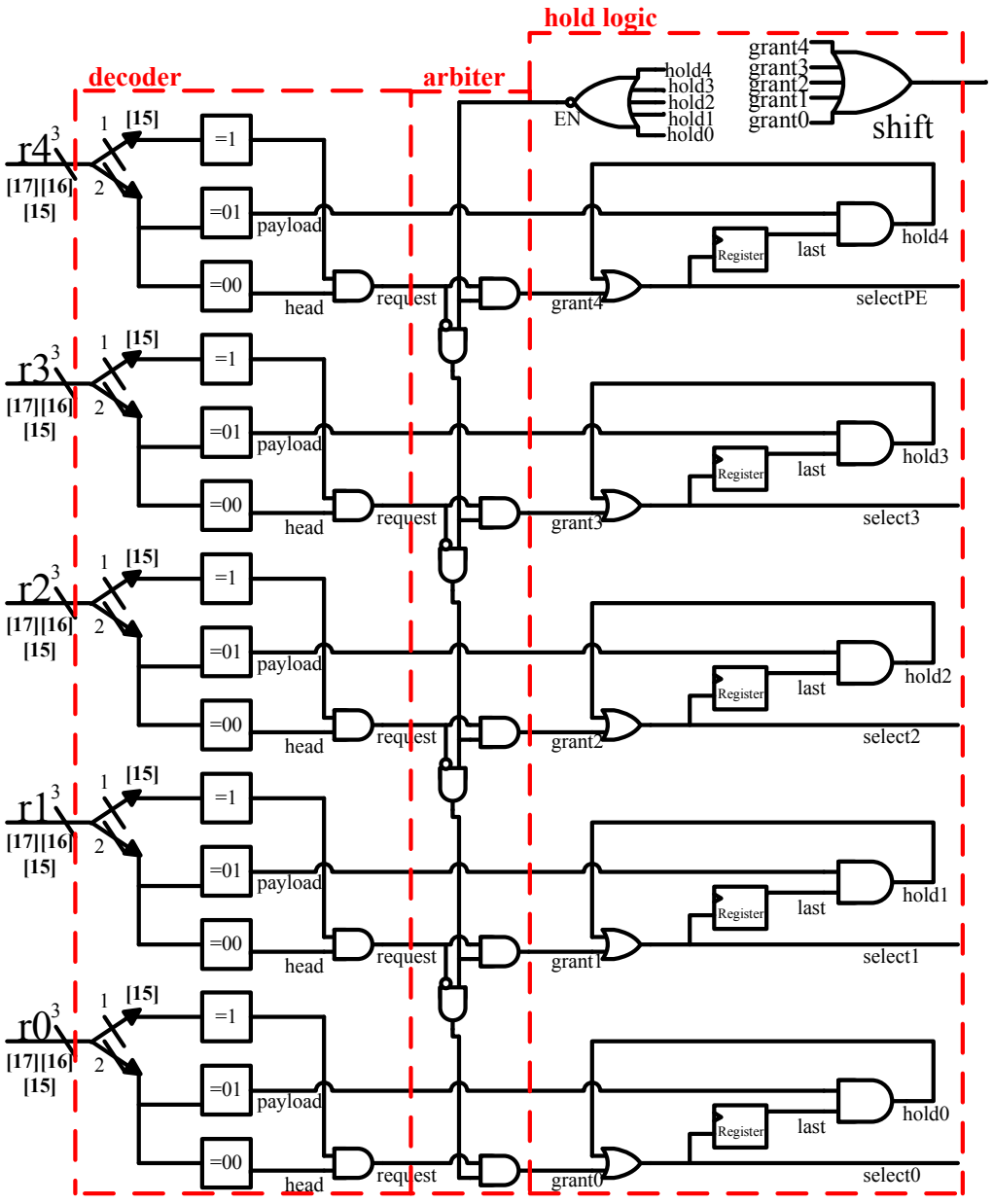


Figure 5.4 Block diagram of a Switch Allocator 0 for FF-ANN

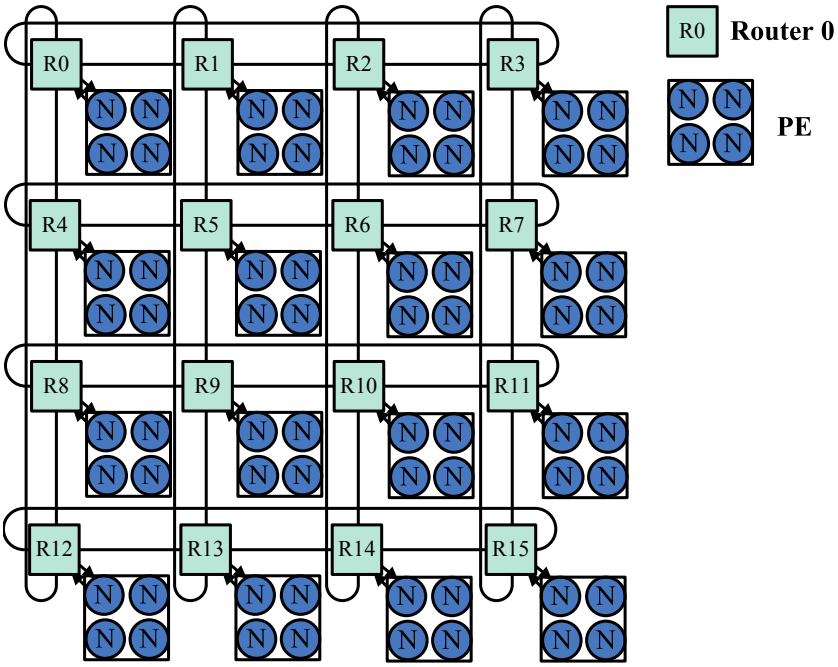


Figure 5.5 Block diagram of a multiple NoC models for FF-ANN

- **Model-2**: the FF-ANN will be implemented by the proposed 4x4 2D torus NoC architecture layer by layer.

Assumed that a_i is the total neurons in layer i, thus n layers FF-ANN can be implemented by the proposed multiple NoC models as follows:

$$\begin{cases} Error : Max\{a_1, a_2, ..., a_n\} \leq 64 \\ Model-1 : a_1 + a_2 + ... + a_n \leq 64 \\ Model-2 : a_1 + a_2 + ... + a_n > 64 \quad \cap \quad Max\{a_1, a_2, ..., a_n\} \leq 64 \end{cases} \tag{5.2.1}$$

It means that if the number of total neurons is not larger than 64 (decided by the structure in Fig. 5.5), the model-1 is used. While the total number of neurons is larger than 64, the model-2 is used. Whereas, the proposed multiple NoC model can not suit for the case which one of the layers has the number of neuron larger than 64. One general application of FF-ANN (n layers and each layer has neuron number of a_i) is implemented by Multiple NoC models as shown in the flow chart of Fig. 5.6.

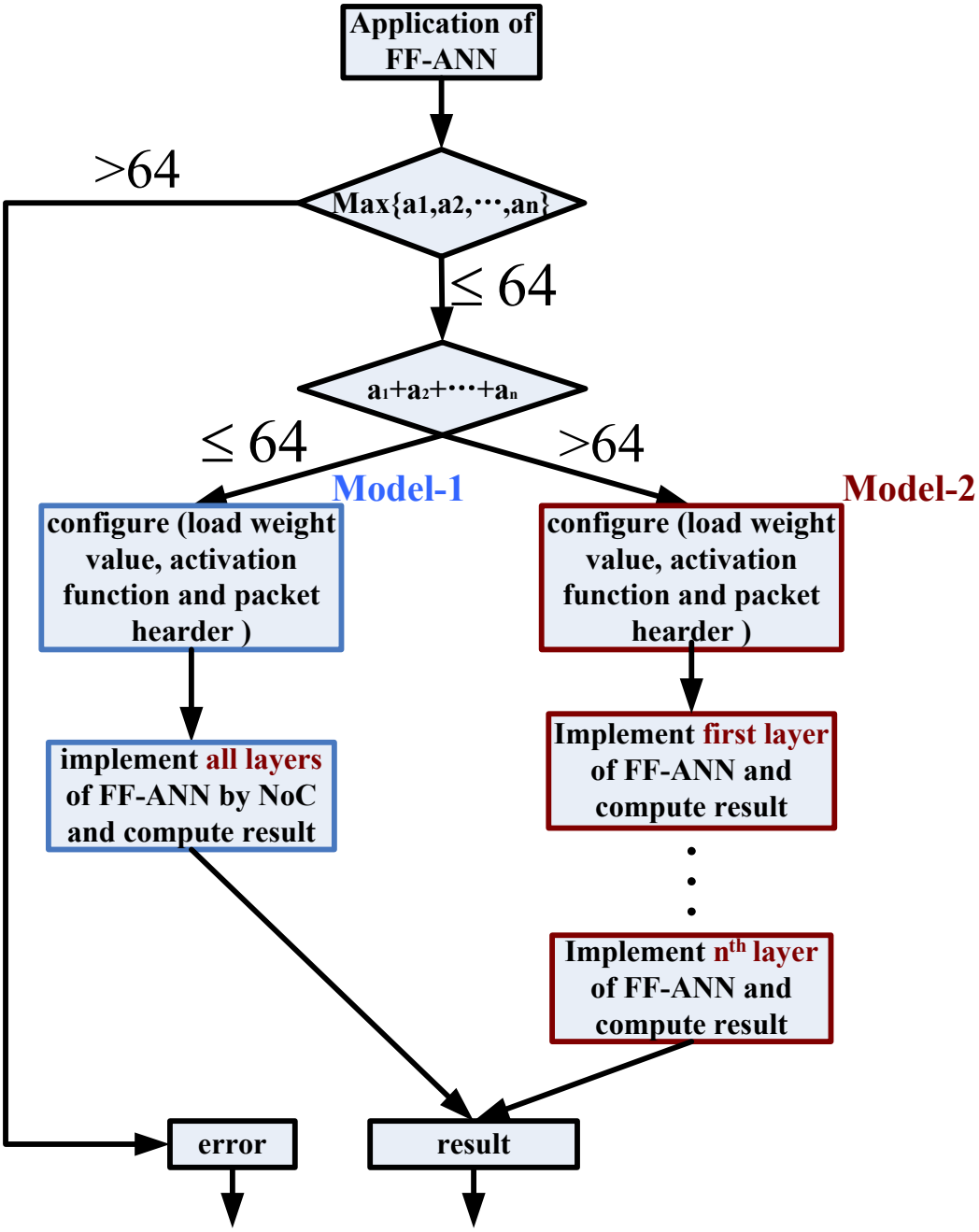


Figure 5.6 Flow chart of FF-ANN implemented by multiple NoC models

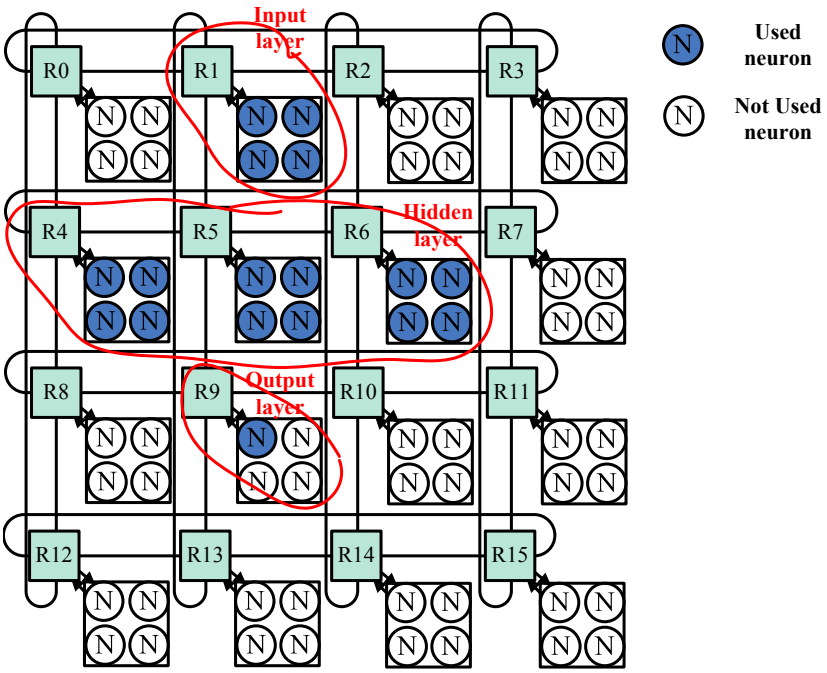


Figure 5.7 Application of FF-ANN to prediction by model-1

Two real applications, prediction and pattern recognition problems, are introduced to discuss these two models in detail. The first application is a prediction problem with the topology of 4-12-1 [143]. It has four input neurons, twelve hidden neurons and one output neuron. The number of total neurons is 17 and less than 64, so the model-1 is used. The real application is implemented by multiple NoC models as shown in Fig. 5.7. For model-1 this system is configured one time which contains the following work:

- Each neuron of each layer need to be implemented to the proposed system.
- Load weight value of each neuron to RAM.
- Load activation function to ROM for each PE.
- Load packet header for each PE to RAM.

In Fig. 5.7 the first layer of FF-ANN consists of one PE which is connected with one router; the second layer of FF-ANN consist of three PEs which is connected with three routers; the third layer of FF-ANN consists of one PE which is connected with one router, and just one neuron in this PE is used. Then, this implemented system can be used for computing.

Table 5.1 The comparison of power dissipation between mesh topology and torus topology

	Mesh (mw)	Torus (mw)
Dynamic power dissipation	1220.25	871.04
Static power dissipation	612.89	620.49
Total power dissipation	1833.14	1491.53

The second application is a pattern recognition problem with the topology of 20-50-1 [115]. It has twenty input neurons, fifty hidden neurons and one output neuron. The total number of neurons is 71 and bigger than 64, so the model-2 is used. The real application will be implemented by multiple NoC models as shown in Fig. 5.8. The whole process is as follows:

- 1. Load weight value, activation function and packet header of each layer of FF-ANN.
- 2. The first layer of FF-ANN is implemented by NoC as Fig. 5.8(a), do the computing work and store these results to RAMs of each PE.
- 3. The second layer of FF-ANN is implemented by NoC as Fig. 5.8(b), input of this layer is the output of former layer, thus read the data from the RAMs which store the results of first layer. Then do the computing work and store these results to RAMs of each PE.
- 4. The third layer of FF-ANN is implemented by NoC as Fig. 5.8(c), and read the data from the RAMs which store the results of second layer. Then do the computing work and this result is final result of FF-ANN.

5.3 Experiment for Support Low Power NoC Design Method

The proposed Multiple NoC models with mesh topology and torus topology are designed by VerilogHDL and implemented on FPGA Stratix II EP2S60F1020C3 [137] using Alter Quartus II to get the performance metrics of power consumption. The comparison between mesh topology and torus topology is shown in Table 5.1. The proposed NoC architecture with torus topology can reduce 28.6% of the dynamic power dissipation compared with the NoC with mesh topology. The static power dissipation of NoC with torus topology is a little larger than that with mesh topology, because of the additional hardware for routers. While the total power dissipation of NoC with torus topology is less than the mesh one, so that the total power dissipation is reduced while static powers are almost same.

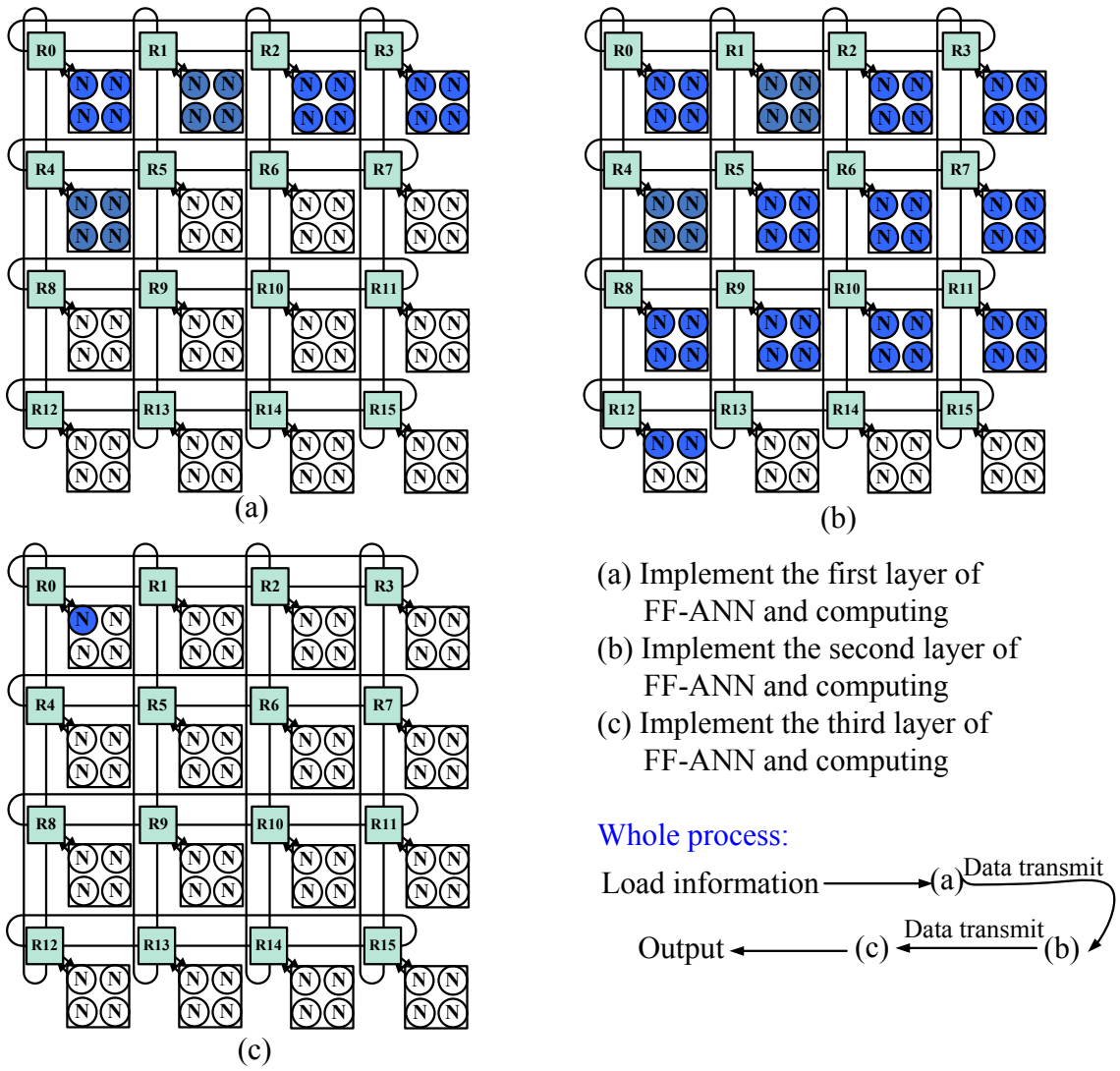


Figure 5.8 Application of FF-ANN to pattern recognition by model-2

As shown in Fig. 5.7 and Fig. 5.8, when different applications are implemented by the proposed architecture, only the work neurons are changed, while the rests are not. As a result this design will not consume the redundant power.

5.4 Evaluation and Discussion

The proposed Multiple NoC models is evaluated and discussed from the following point of view: reconfigurability, reparability, communication load, and system performance of CPS.

5.4.1 Reconfigurability of the proposed system

The proposed system is reconfigurable. Different applications of FF-ANNs just have difference in weight value of each neuron, activation function of each layer, the number of neurons and layers, and so on. These differences can be easily implemented in the proposed multiple NoC models. Because those values are stored in RAM. Furthermore, the topology and routing algorithm of NoC can be changed easily.

5.4.2 Reparability of the proposed system

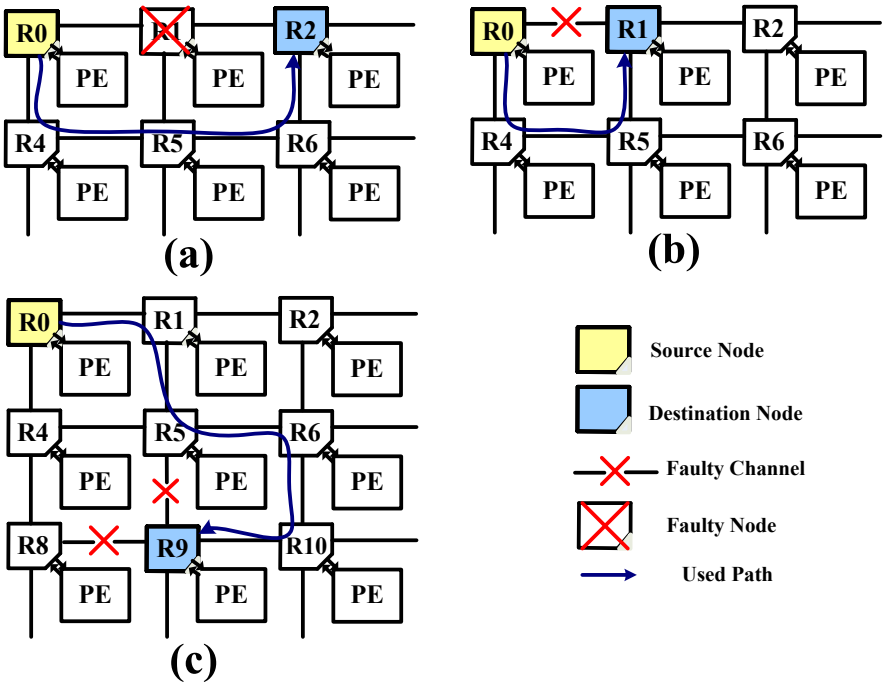
The reparability of the proposed system in this work is shown as follows:

- Neurons are reparable: each neuron has the same architecture, thus the faulty neurons can be replaced by others.
- Channels or routers are reparable: the routing paths of the data transmit from layer to layer is not fixed, and it can be changed by sending new packets. The cases are shown in Fig. 5.9.

5.4.3 Communication load reduction

The communication load of the proposed NoC data transmission architecture is reduced compared with existing Point to Point (P2P) data transmission architecture. Point to Point (P2P) architecture is always used by existing digital FF-ANNs for their data transmission, while the proposed NoC data transmission architecture is packet based.

Three applications of ANN with topology of 4-12-1, 4-5-5-1 and 20-50-1 [143, 144, 115] are simulated to compare these two types of data transmission architectures. Let's suppose that the ANN with topology of 4-12-1 is implemented by P2P data transmission architecture, and assume one packet will be transmitted to all neurons of next layer by each neuron. As a result, 60 packets



a) Assume R1 is faulty, packet can pass from R0 to R2 via R4, R5, and R6.
b) Assume channel between R0 and R1 is faulty, packet can pass from R0 to R1 via R4 and R5;
c) Assume channels between R8 and R9, R5 and R9 are faulty, packet can pass from R0 to R9 via R1, R5, R6, and R10;

Figure 5.9 Reparability of the proposed system

are transmitted and each packet size is 16 bits. Thus the total packet size is 960 bits. When the same ANN is implemented by the proposed NoC packet based data transmission architecture, it includes one PE with 4 neurons for input layer; hidden layer consists of three PEs and each with four neurons; and one PE with one neuron for output layer. Each PE will transmit one packet to the next layer's PE. The packet size of PE is decided by the number of attached neurons. When four neurons are attached to PE, the packet size is 90 bits which includes one 18 bits header and four 18 bits payloads. When three neurons are attached to PE, the packet size is 72 bits which includes one 18 bits header and three 18 bits payloads. When two neurons are attached to PE, the packet size is 54 bits which includes one 18 bits header and two 18 bits payloads. When one neuron is attached to PE, the packet size is 36 bits which includes one 18 bits header and one 18 bits payload. Thus, in total, 6 packets are transmitted and total packet size is 540 bits. Other two applications of FF-ANNs are also compared between P2P data transmission method and NoC data transmission method as

Table 5.2 Comparison of communication load

Topology	Method	Packet number	Packet size (bit)	Total size (bit)
4-12-1	P2P	60	16	960
	NoC	6	90	**540**
4-5-5-1	P2P	50	16	800
	NoC	8	36/90	**558**
20-50-1	P2P	1050	16	16800
	NoC	78	54/90	**6984**

shown in Table 5.2. Consequently the proposed NoC method can reduce the communication load of total packet size about 30.25%-58.4%.

5.4.4 High performance of connection-per-second (CPS)

Three real applications of FF-ANN [143, 144, 115] are implemented by the proposed multiple NoC models and simulated by NIRGAM NoC simulator [135]. The first one is a prediction problem [143], and the topology of it is 4-12-1. The second one is an optimization problem [144], and the topology of it is 4-5-5-1. The third one is a pattern recognition problem [115], and the topology of it is 20-50-1. The input of them is one thousand 16-bit data. We call them application-1, application-2, and application-3 respectively. The simulation result is shown in Fig. 5.10. The frequency of proposed system is 100MHz, thus the worst latency of first two applications is 20ns, and the worst latency of each stage of third application is 56ns and 56.6ns. The CPU time of using NOXIM NoC simulator for two applications are 7.7s, 20.2s, respectively.

The most common measure of performance is the Connection-Per-Second (CPS), which is defined as the rate of multiplication and accumulates operations. For the fixed type of hardware ANN, the value of CPS is different when implementing different applications. For fair comparison between the existing hardware ANN and the proposed one, we need to implement the same application and then compare the CPS of different hardware ANN. Three real applications of FF-ANN are also implemented by existing hardware ANN [1, 28] to get the CPS. Experimental results are shown in Table 5.3. In this table, the symbol ”N.A.” means this type of hardware neural network can not execute the application due to the limitation of the hardware architecture. This table shows that the proposed multiple NoC models can increase CPS about 47.1%, 25%, and 44.9% for the application-1, application-2 and application-3 respectively.

How to get a good balance between performance and cost has been researched for a long time.

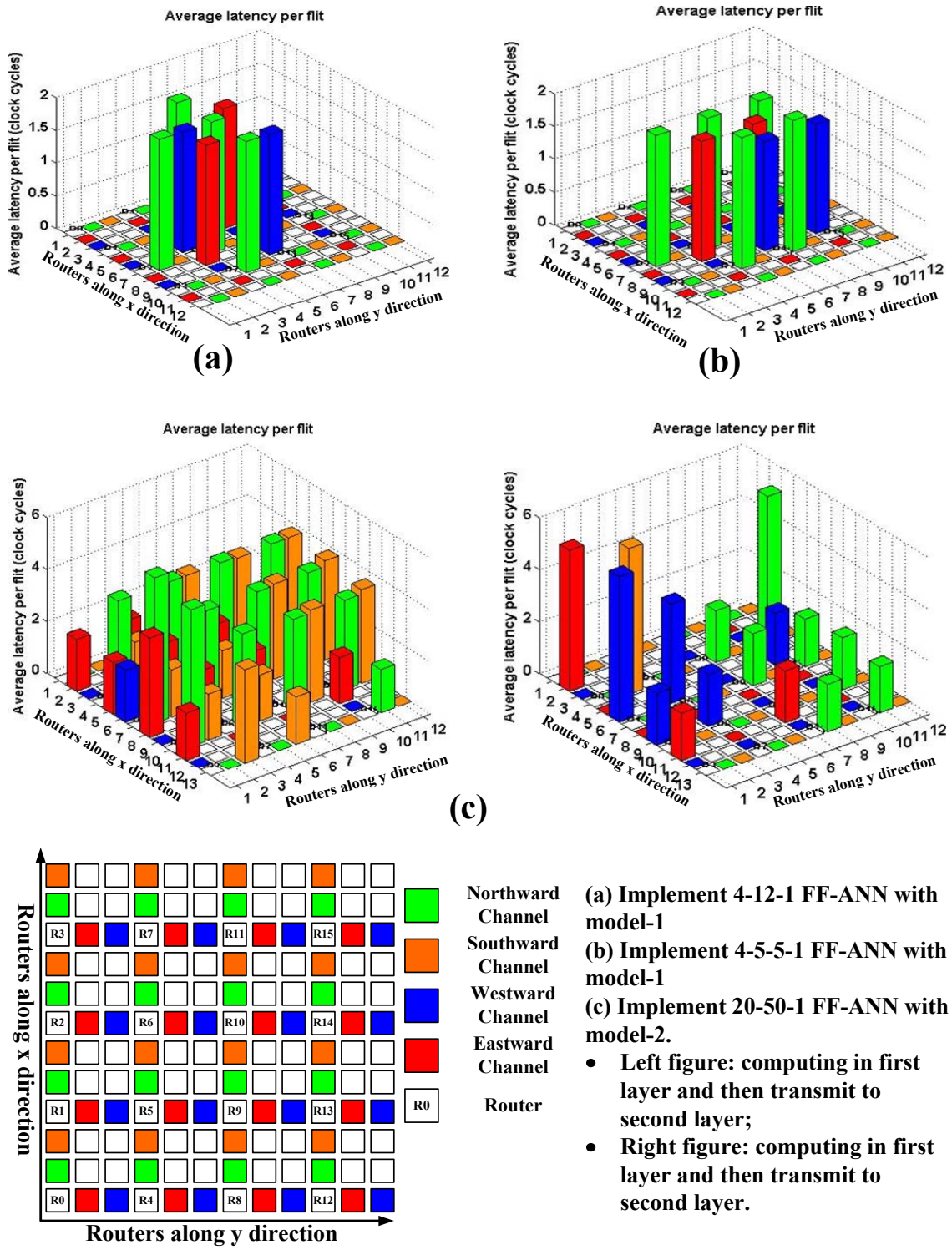


Figure 5.10 Average latency per flit of three applications

Table 5.3 Comparison of CPS and CPSPW

Name	Architecture	Learn	Precision	Neurons or PE	CPS		
					App1[a]	App2	App3
NeuraLogix NLX-420	Slice	Off-chip	1-16b	16	N.A.	300	N.A.
Philips Lneuro-2.3	Slice	Off-chip	16,32b	12PE	340M	225M	N.A.
Siemens MA-16	Systolic array	Off-chip	16b	16PE	N.A.	400M	N.A.
Hitachi WSI	SIMD	BP	9x8b	144	N.A.	300	N.A.
Inova N64000	SIMD	Program	1-16b	64PE	76.9M	50.9M	127M
Waseda Multiple NoC	**NoC**	**Off-chip**	**1-16b**	**64,16PE**	**500M**	**500M**	**184M**

[a] App1: Application1

The slice architecture and systolic array architecture show the high performance and good balance when implement the ANN with small size; SIMD architecture has a good reconfigurability, low cost and high performance when implement the ANN with large size. Model-1 of the proposed NoC architecture has a similar implement method as slice architecture, thus it is appropriate for the ANN with small size; while model-2 of the proposed NoC architecture has a similar implement method as SIMD architecture, so that, it is fit for the ANN with large size. Furthermore, the NoC architecture has a smart packet based data transmission method. These advantages make NoC architecture much more suitable for hardware ANN.

5.5 Conclusions

A multiple NoC models are designed based on low power and implemented for hardware ANN. This NoC architecture can implement both the small size ANN and the large size ANN. The proposed architecture is reconfigurable to suit for different applications of FF-ANN, by changing the weight value, activation function, and the number of neurons and layers. The hardware of it is reparable which the faulty neurons and channels can be replaced by good one. Compared to the traditional P2P data transmission method, it can reduce the communication load about 30.25%-58.4%, and it

can increase system performance of CPS about 25%-47.1% compared with the existing hardware ANN. The number of PEs in the proposed multiple NoC model is 16. But it is not fixed and could be changed, such as 9, 25, 36 and so on.

Chapter 6

High Performance Feedforward Neural Network Mapped by NoC architecture with a new Routing Strategy Implementation Method

In this chapter, Networks on Chip (NoC) architecture is proposed for mapping Feedforward Artificial Neural Network (FF-ANN). A new router model of NoC with absolute address based routing strategy is developed to replace a router with Destination-Tag to reduce the packet size of header. The NOXIM NoC simulator is used to evaluate the proposed router in term of average latency and max latency. The experimental results indicate that the proposed NoC architecture with this new router model is effective in reducing latency compared with the traditional one, and it brings a mapped FF-ANN higher performance and lower communication load.

6.1 Introduction

Artificial neural network (ANN) is a computational model or mathematical model that is inspired by the way biological nervous systems, such as the human brain. It consists of an interconnected group of neurons to computation or model complex relationships between inputs and outputs [2]. As the most common ANN, the Feedforward ANN (FF-ANN) is widely used to solve a variety of problems, such as prediction, anomalous detection, optimization, pattern recognition and so on [121][113][99].

In the last few years, a network on chip (NoC) has attracted more and more attention according

to its smart structure. It uses a general-purpose on-chip interconnection network to replace design-specific global shard buses [78]. It has been used by a lot of research groups to solve complex on-chip interconnection problems for large system-on-chip (SoC) [68][145].

In chapter 4, we proposed a Networks on Chip (NoC) architecture to implement the FF-ANN, called NoC-ANN, to achieve low communication load and high performance of Connection-Per-Second (CPS) [141]. It will be described more detail in Sect. 6.3.1.

In this chapter, the NoC architecture is improved to implement the FF-ANN, named iNoC-ANN, which has lower communication load and higher performance. The most improvement is that using a proposed router model with absolute address based routing strategy [146] to replace the former router with Destination-Tag (DT) method based routing strategy. This absolute address based routing strategy could reduce the header size of the packet for NoC compared with the DT method, so that the latency will be reduced. We evaluate the new NoC architecture, and the results indicate that the proposed method is very effective. We also evaluate the communication load and performance of iNoC-ANN system. Resultantly it could achieve lower communication load and higher performance compared with the former NoC-ANN.

This chapter is organized as follows. First, we introduce the background of the former work and motivation of this work in Sect. 6.2. In Sect. 6.3, improved NoC architecture with new router model for iNoC-ANN, and it is evaluated in Sect. 6.4. Section 6.5 concludes this chapter.

6.2 Related Works

The well-known digital implementation methods of an FF-ANN are slice architecture [48], single instruction multiple data (SIMD) [147], systolic array devices [49], and multi-processor chips [148]. The drawbacks of these implementation methods are high cost, low performance, no reconfigurability, and heavy communication load [26] [28]. For overcoming these drawbacks, we already proposed the NoC architecture to implement an FF-ANN [141]. But, lower communication load and higher performance are continuously required, and they are decided by the latency of NoC, which is affected by the packet size and router architecture [68].

One of popular routing strategies is DT method [149], which is used in many NoC applications, such as 80-Tile Intel TFLOPS NoC [150] and 18-Tile NN (Neural Network) NoC [141]. The destination addresses of DT method are used to select the output port at each router. The bit number of the destination address is decided by the combination of the port number of the router and the hop number from the source to the destination node. The advantage is that each hop could be easily

controlled and different routing algorithms could be easily realized, whereas the disadvantage is that the total destination address stored in header becomes larger and larger proportional to the network size.

Another routing strategy is an absolute address method [151]. With this method, a router decides the next move based on the absolute address of the final destination node. This method could reduce the bit number of the address which is stored in header, compared with DT method. But, the drawback is also exist: different router models have different method to realize this routing strategy and the routers should be redesigned when different routing algorithms are implemented.

We consider that a new router model must be effective if it uses absolute address and can meet different routing algorithms with a little hardware change. With this router model, the packet size could be reduced. Thus the NoC architecture could achieve lower latency and ANN based on this NoC could achieve lower communication load and higher performance. Therefore, our work focuses on reducing packet size to improve the ANN with NoC implementation method.

6.3 Proposed iNoC-ANN System

In this section, the former NoC-ANN system [141] is introduced at first, and then the design steps of the proposed iNoC-ANN system with absolute address based routing strategy is introduced.

6.3.1 Former NoC-ANN System with DT Based Routing Strategy

For the former NoC-ANN design, the following five steps are processed in this order: design one neuron, 4 neuron aggregated as one PE, router design, and finally a 4x5 NoC-ANN system is designed by a set of routers and PEs as shown in Fig. 6.1. When the input neurons finish its computing work, the computing results will then transmit to the all the neurons in first hidden layer via routers which connect with them. Also the neurons in first hidden layer will transmit their results to the neurons in second hidden layer and then transmit to the neurons in the output layer via routers.

The routing strategy is based on DT method. The transmission of the packets is controlled by the destination address part of the header. The destination addresses are used to select the output port of each router. The bit number of the destination address is decided by the port number of the router and the hop number from the source to the destination node. For example, one 5-port router needs 3 bits destination address to select its output port. If there are 8 hops from the source to the destination node, it needs 3x8=24 bits in total. The bits of addresses are stored in the header of the packet. The header grows with increasing NoC size as shown in Table 6.1. One header consists of

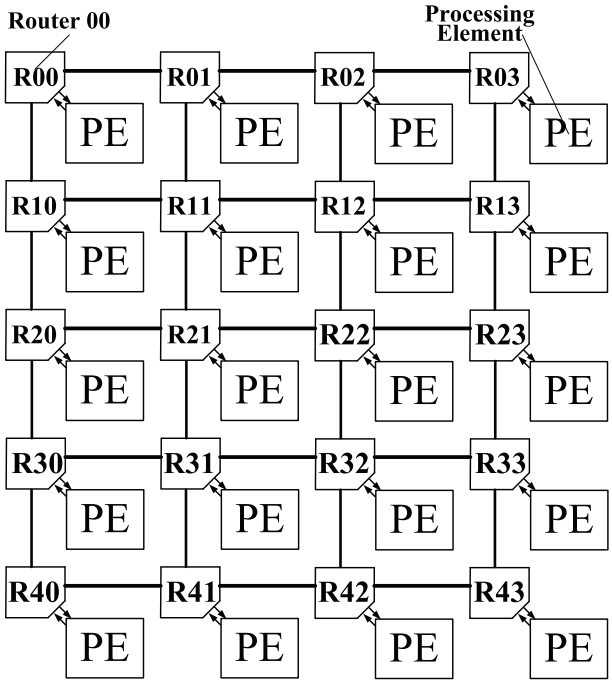


Figure 6.1 NoC-ANN with 4x5 2D mesh topology.

one bit for virtual channel, two bits for flit type and the rest for address. We use probability theory to calculate the average length of the address. We assume that each Processing Element (PE) has the same probability to sent a packet to all the PE of this NoC via routers. We calculate the average bit number of each PE in the NoC one by one, and then get the average bit number of the whole NoC.

6.3.2 Proposed System

The proposed iNoC-ANN system is designed similarly to the former five steps, except the router design step. The absolute address method is introduced at first and then applied to router.

Packet Format of Absolute Address Based Method

The absolute address based method is different in a header from DT method. The destination address of the header of absolute address based method is described by x-axis address and y-axis address of the final destination node.

It is obvious that the lengths of the absolute address and the destination tag are O(log n) and O(n), respectively, for n*n NoC. The header size of absolute address based method grows with

Table 6.1 Header of DT method increases with NoC size

NoC size nxn	Bits in the header			Tot.No.of bits
	VC[a]	FT[b]	Ave. Addr.[c]	
n=2	1	2	6	9
n=3	1	2	8.49	11.49
n=4	1	2	10.5	13.5
n=5	1	2	12.72	15.72
n=6	1	2	14.2	17.2
n=7	1	2	17.2	20.2
n=8	1	2	19.75	22.75

[a]virtual channel
[b]flit type
[c]average address

Table 6.2 Header of absolute address based method increases with NoC size

NoC size nxn	Bits in the header				Tot.No.of bits
	VC	FT	x[a]	y	
n=2	1	2	1	1	5
$3 \leq n \leq 4$	1	2	2	2	7
$5 \leq n \leq 8$	1	2	3	3	9
$9 \leq n \leq 16$	1	2	4	4	11
$17 \leq n \leq 25$	1	2	5	5	13

[a]x-axis

increasing NoC size as shown in Table 6.2 and is compared with that of DT method as shown in Fig. 6.2. So the absolute address based method is effective in reducing header size of the packet.

Router Model with Absolute Address Based Method

As described in Sect.6.2, the NoC architecture with absolute address based routing strategy could reduce the bit size of header, whereas the routers of NoC should be redesigned for implementing different routing algorithms. In this section, a new router model is introduced to implement the absolute address based routing strategy.

Block diagram of the proposed router with absolute address based method is shown in Fig. 6.3. This router has 5 input ports and 5 output ports that are connected to one process element (PE) and four neighbor routers. It consists of buffers, virtual-channel allocators (VA), switch allocators (SA), and crossbar switch. VA controls the input flits to choose one of the channels, then it is transmitted

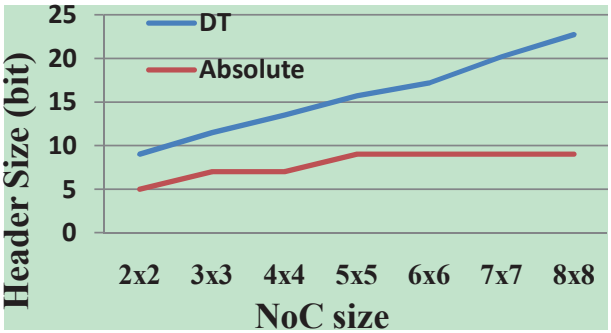


Figure 6.2 Absolute vs DT header size

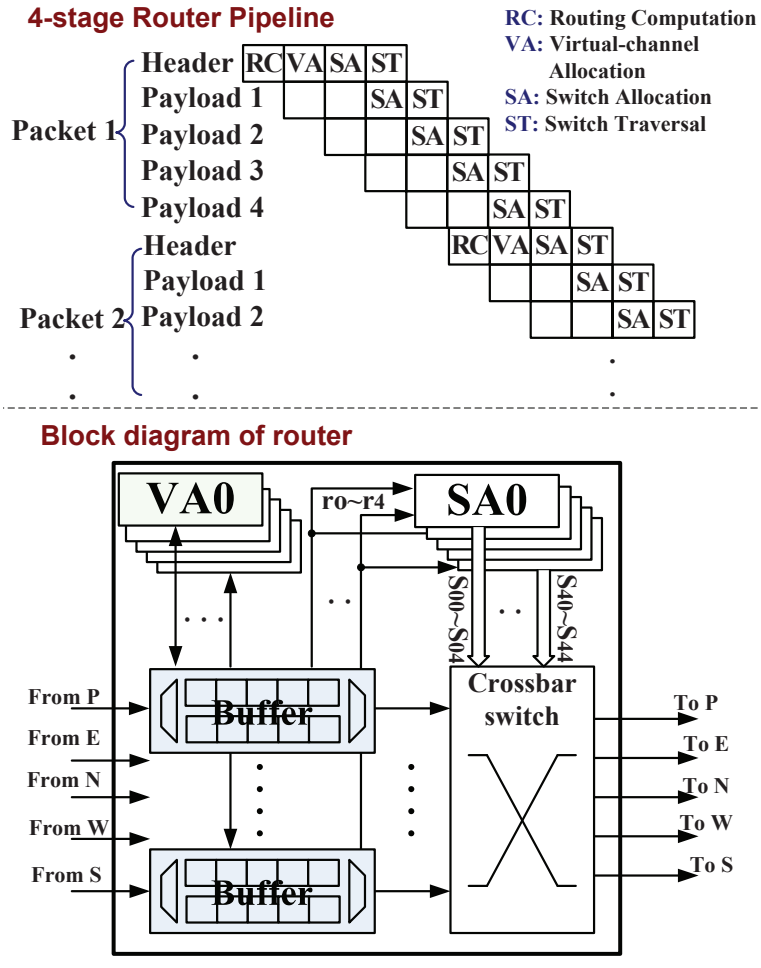


Figure 6.3 Block diagram of the proposed router

to crossbar switch. Each SA will check part of input flits and these bits are consisted by 2-bit flit type (FT) and absolute destination address (ADA). Different sizes of NoC have different size of absolute address. Under control of the SAs, input flits are selected to transmit to output ports. Switch allocator0 chooses the input flits for output port PE, and switch allocator1, 2, 3, 4 for output port East, North, West, South, respectively. Compared with the original router model in the former NoC-ANN, SAs are redesigned and the redundant shifters are removed.

Each SA selects input flits for one output port. It consists of three parts: decoder, arbiter and hold logic as shown in Fig. 6.4. Except the function part, each SA has the same architecture. In decoder part, FT and ADA of each input flit are decoded. FT is used for partition the flits type (10 for header and 11 for payload) and ADA is used for choosing output port. ADA compared with a function of each SA, where f_{ij} means a function in SA j for flit form port i. And ports 0, 1, 2, 3 and 4 denote input ports for PE, East, North, West and South, respectively. Different function of each SA will be described below in detail. The routers need not be redesigned when we implement different routing algorithms, just function blocks of SAs will be changed.

As described above, ADA need to be compared with the function blocks to select the output port. The x-axis address and the y-axis address of ADA are compared with those of a local router and also . ADA of input flit from input port i need to be compared with five functions of f_{ij} in five SAs. If this ADA is satisfied with one of functions in SA$_j$, this input flit is selected by the output port j. For example, one input flit from input port 1 has ADA, it is compared with five functions of f_{10} in SA0, f_{11} in SA1, f_{12} in SA2, f_{13} in SA3, and f_{14} in SA4. If this is satisfied with f_{12} in SA2, this flit from input port 1 is selected by output port 2. Five well-known routing algorithms are used as example to show the proposed implementing method for iNoC-ANN. And they are also used as benchmark to compare between the former DT method and the proposed method in Sect. 4.1. When we set the logical conditions for function blocks of SAs, we list all existing cases of logical conditions between current addresses and destination addresses, and then set them one by one to function blocks according to the routing algorithm. This method could avoid undefined cases and multiple-defined cases. Common pseudocode of X-Y routing algorithm is shown first, and then all the functions in different SAs are listed for hardware implementation as follows:

x_{des} and y_{des} denote x-axis and y-axis of ADA, respectively, and x_{local} and y_{local} denote x-axis and y-axis address of local router, respectively.

X-Y routing algorithm

(1) pseudocode

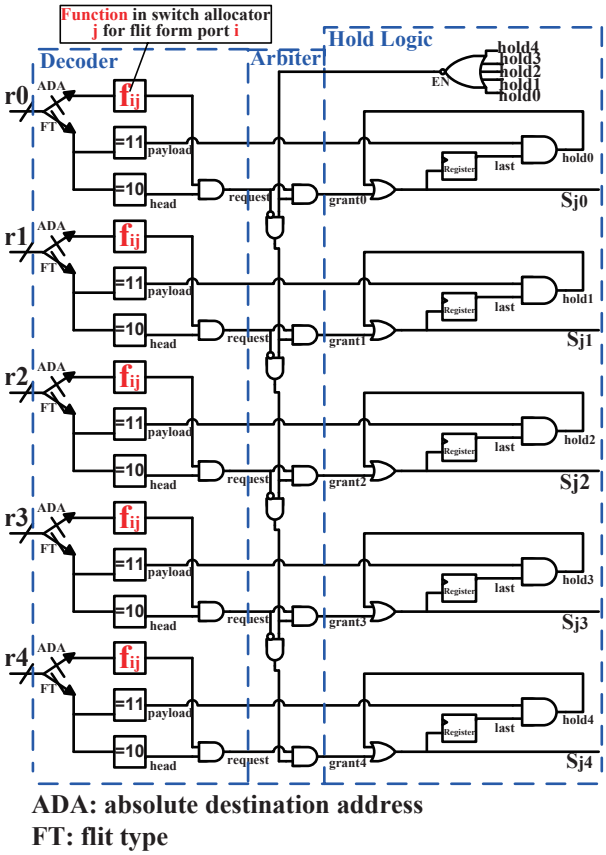


Figure 6.4 Architecture of SA for absolute address based method

if $(x_{des} > x_{local})$ {To_East;}
else if $(x_{des} < x_{local})$ {To_West;}
else if $(y_{des} > y_{local})$ {To_South;}
else if $(y_{des} < y_{local})$ {To_North;}
else {To_PE;}

(2) logical conditions in Functions of SAs

• Allocator0$\Longrightarrow$ PE (Allocator0 will select input phits for output port of PE)

f_{i0}: $(x_{des} = x_{local}) \wedge (y_{des} = y_{local})$, $i = 0, 1, ...4$;

• Allocator1$\Longrightarrow$ East

f_{i1}: $(x_{des} > x_{local})$, $i = 0, 1, ...4$;

• Allocator2$\Longrightarrow$ North

f_{i2}: $(x_{des} = x_{local}) \wedge (y_{des} < y_{local})$, $i = 0, 1, ...4$;

• Allocator3$\Longrightarrow$ West

f_{i3}: $(x_{des} < x_{local})$, $i = 0, 1, ...4$;

• Allocator4$\Longrightarrow$ South

f_{i4}: $(x_{des} = x_{local}) \wedge (y_{des} > y_{local})$, $i = 0, 1, ...4$.

For different routing algorithms, such as West-First (W-F), North-Last (N-L), Negative-First (N-F), Full-Adaptive (F-A) and so on, the implementation methods are almost the same. We just need to change the logical conditions in the functions of SAs for different routing algorithms. From ref. [86], the pseudocode of W-F, N-L, N-F and F-A routing algorithms are shown first, and then all the functions in different SAs are listed for hardware implementation as follows:

W-F routing algorithm

(1) pseudocode

if ($x_{des} <= x_{local} \,||\, y_{des} == y_{local}$)

return routing_XY;

if ($x_{des} <= x_{local}$ or $y_{des} = y_{local}$) packets will transmit ## follow XY routing.

else if ($y_{des} < y_{local}$) {To_North; To_East;}

##packets transmit to North and then to East in next hop.

else {To_South; To_East;}

##packets transmit to South and then to East in next hop.

(2) logical conditions in Functions of SAs

• Allocator0$\Longrightarrow$ PE

f_{i0}: $(x_{des} = x_{local}) \wedge (y_{des} = y_{local})$, $i = 0, 1, ...4$;

• Allocator1$\Longrightarrow$ East

f_{01}, f_{11} and f_{31} : $(x_{des} > x_{local}) \wedge (y_{des} = y_{local})$;

f_{21}: $(x_{des} > x_{local}) \wedge (y_{des} >= y_{local})$

f_{41}: $(x_{des} > x_{local}) \wedge (y_{des} <= y_{local})$.

• Allocator2$\Longrightarrow$ North

f_{02}, f_{12}, f_{22} and f_{32}: $(x_{des} >= x_{local}) \wedge (y_{des} < y_{local})$;

f_{42}: $(x_{des} = x_{local}) \wedge (y_{des} < y_{local})$;

• Allocator3$\Longrightarrow$ West

f_{i3}: $(x_{des} < x_{local})$;

• Allocator4$\Longrightarrow$ South

f_{04}, f_{14}, f_{34} and f_{44}: $(x_{des} >= x_{local}) \wedge (y_{des} > y_{local})$;
f_{24}: $(x_{des} = x_{local}) \wedge (y_{des} > y_{local})$.

N-L routing algorithm

(1) pseudocode

if ($x_{des} == x_{local}$ || $y_{des} <= y_{local}$)
return routing_XY;
if ($x_{des} = x_{local}$ or $y_{des} <= y_{local}$) packets will transmit ## follow XY routing.
else if ($x_{des} < x_{local}$) {To_South; To_West;}
##packets transmit to South and then to West in next hop.
else {To_South; To_East;}
##packets transmit to South and then to East in next hop.

(2) logical conditions in Functions of SAs

• Allocator0$\Longrightarrow$ PE
f_{i0}: $(x_{des} = x_{local}) \wedge (y_{des} = y_{local})$, $i = 0, 1, ...4$;
• Allocator1$\Longrightarrow$ East
f_{21}: $x_{des} > x_{local}$;
f_{01}, f_{11}, f_{31} and f_{41} : $(x_{des} > x_{local}) \wedge (y_{des} <= y_{local})$;
• Allocator2$\Longrightarrow$ North
f_{i2}: $(x_{des} = x_{local}) \wedge (y_{des} < y_{local})$;
• Allocator3$\Longrightarrow$ West
f_{23}: $x_{des} < x_{local}$;
f_{03}, f_{13}, f_{33} and f_{43} : $(x_{des} < x_{local}) \wedge (y_{des} <= y_{local})$;
• Allocator4$\Longrightarrow$ South
f_{04}, f_{12}, f_{34} and f_{44}: $y_{des} > y_{local}$;
f_{24}: $(x_{des} = x_{local}) \wedge (y_{des} > y_{local})$.

N-F routing algorithm

(1) pseudocode

if (($x_{des} <= x_{local}$ && $y_{des} <= y_{local}$) ||
($x_{des} >= x_{local}$ && $y_{des} >= y_{local}$))
return routing_XY;

if address is satisfied with the condition, packets will ## transmit follow XY routing.
else if ($x_{des} > x_{local}$ && $y_{des} < y_{local}$)
{To_North; To_East;}
##packets transmit to North and then to East in next hop.
else {To_South; To_West;}
##packets transmit to South and then to West in next hop.

(2) logical conditions in Functions of SAs
• Allocator0$\Longrightarrow$ PE
f_{i0}: $(x_{des} = x_{local}) \wedge (y_{des} = y_{local})$, $i = 0, 1, ...4$;
• Allocator1$\Longrightarrow$ East
f_{01}, f_{11}, f_{21} and f_{31} : $(x_{des} > x_{local}) \wedge (y_{des} >= y_{local})$;
f_{41}: $x_{des} > x_{local}$;
• Allocator2$\Longrightarrow$ North
f_{42}: $(x_{des} = x_{local}) \wedge (y_{des} < y_{local})$;
f_{02}, f_{12}, f_{22} and f_{32} : $(x_{des} >= x_{local}) \wedge (y_{des} < y_{local})$;
• Allocator3$\Longrightarrow$ West
f_{23}: $x_{des} < x_{local}$;
f_{03}, f_{13}, f_{33} and f_{43} : $(x_{des} < x_{local}) \wedge (y_{des} <= y_{local})$;
• Allocator4$\Longrightarrow$ South
f_{04}, f_{12}, f_{34} and f_{44}: $(x_{des} <= x_{local}) \wedge (y_{des} > y_{local})$;
f_{24}: $(x_{des} = x_{local}) \wedge (y_{des} > y_{local})$.

F-A routing algorithm

(1) pseudocode
if ($x_{des} == x_{local} \,||\, y_{des} == y_{local}$)
return routing_XY;
if address is satisfied with the condition, packets will ## transmit follow XY routing.
else if ($x_{des} > x_{local}$ && $y_{des} < y_{local}$)
{To_North; To_East;}
##packets transmit to North and then to East in next hop.
else if ($x_{des} > x_{local}$ && $y_{des} > y_{local}$)
{To_South; To_East;}

##packets transmit to South and then to East in next hop.
else if ($x_{des} < x_{local}$ && $y_{des} > y_{local}$)
{To_South; To_West;}
##packets transmit to South and then to West in next hop.
else {To_North; To_West;}
##packets transmit to North and then to West in next hop.

(2) logical conditions in Functions of SAs
• Allocator0$\Longrightarrow$ PE
f_{i0}: $(x_{des} = x_{local}) \wedge (y_{des} = y_{local})$, $i = 0, 1, ...4$;
• Allocator1$\Longrightarrow$ East
f_{41}: $(x_{des} > x_{local}) \wedge (y_{des} <= y_{local})$;
f_{01}, f_{11} and f_{31} : $(x_{des} > x_{local}) \wedge (y_{des} = y_{local})$;
f_{21} : $(x_{des} > x_{local}) \wedge (y_{des} >= y_{local})$;
• Allocator2$\Longrightarrow$ North
f_{02}, f_{12}, f_{22} and f_{32} : $y_{des} < y_{local}$;
f_{42}: $(x_{des} = x_{local}) \wedge (y_{des} < y_{local})$;
• Allocator3$\Longrightarrow$ West
f_{43}: $(x_{des} < x_{local}) \wedge (y_{des} <= y_{local})$;
f_{03}, f_{13} and f_{33} : $(x_{des} < x_{local}) \wedge (y_{des} = y_{local})$;
f_{23} : $(x_{des} < x_{local}) \wedge (y_{des} >= y_{local})$;
• Allocator4$\Longrightarrow$ South
f_{04}, f_{14}, f_{34} and f_{44} : $y_{des} > y_{local}$;
f_{24}: $(x_{des} = x_{local}) \wedge (y_{des} > y_{local})$.

6.4 Evaluation of the proposed iNoC-ANN

In this section, both the proposed absolute address based router model and the former DT method based router model are implemented for different routing algorithms and compared in term of hardware resource, average latency, max latency, average throughput and power consumption. And then the iNoC-ANN with proposed absolute address based router is compared with the former NoC-ANN and other existing ANN architecture in term of communication load and performance.

Table 6.3 Comparisons of FPGA resource (number of ALUTs)

Router type and network size	Conventional	Proposed		Ratios(Pro./Conv.)	
		X-Y	W-F	X-Y	W-F
3-port router	130	86	89,87,87,86[a]	0.66	0.66-0.68
4-port router	235	147	145,144,148[b]	0.63	0.61-0.63
5-port router	445	197	198	0.44	0.44
3x3 NoC	1937	1108	1101	0.57	0.57
4x4 NoC	4079	2236	2237	0.55	0.55
8x8 NoC	21858	10498	10579	0.48	0.48

[a] 3-port routers in four corners of NoC for W-F routing algorithm have different architecture according to the different function blocks

[b] 4-port routers in four edges of NoC for W-F routing algorithm have different architecture according to the different function blocks

6.4.1 Evaluation of NoC architecture with absolute address based router model

As described in Sect. 6.3, different routing algorithms could be implemented into the conventional router by sending the different packets with the DT method, where the router architecture need not be changed. With the absolute address based method, different routing algorithms could be implemented into the developed router by revising the function part of the switch allocators. The proposed routers with X-Y or W-F routing algorithm are used as examples. They are both designed by Verilog HDL language and implemented on Altera Stratix II EP2S30F672C3. The FPGA resource comparison of router architecture is shown in Table 6.3. These two methods are also compared in different sizes of NoCs as shown in Table 6.3.

From Table 6.3, the proposed router model with absolute address based method could reduce the number of ALUTs, especially when the router has more ports and the NoC size is larger.

NOXIM NoC simulator [134] is used to evaluate the latency of NoC, which is an extensible and modular simulator based on system C.

The proposed router model is evaluated by using two different-sized NoC, 4x4 and 8x8. Each size of NoC is used to run ten experiments with three different routing algorithms (X-Y, W-F and N-L routing algorithm) each for proposed router model with absolute address based method and the former router model with DT method. 10000 random packets are transmitted at random in different packet injection rate, and the buffer size is set as 5. Parameters are set for different methods. The noticeable difference between these two methods is header size which has described in 6.3.2. The payload size in this experiment is set as 1$\sim$4 flit. Higher injection rate will result in higher latency,

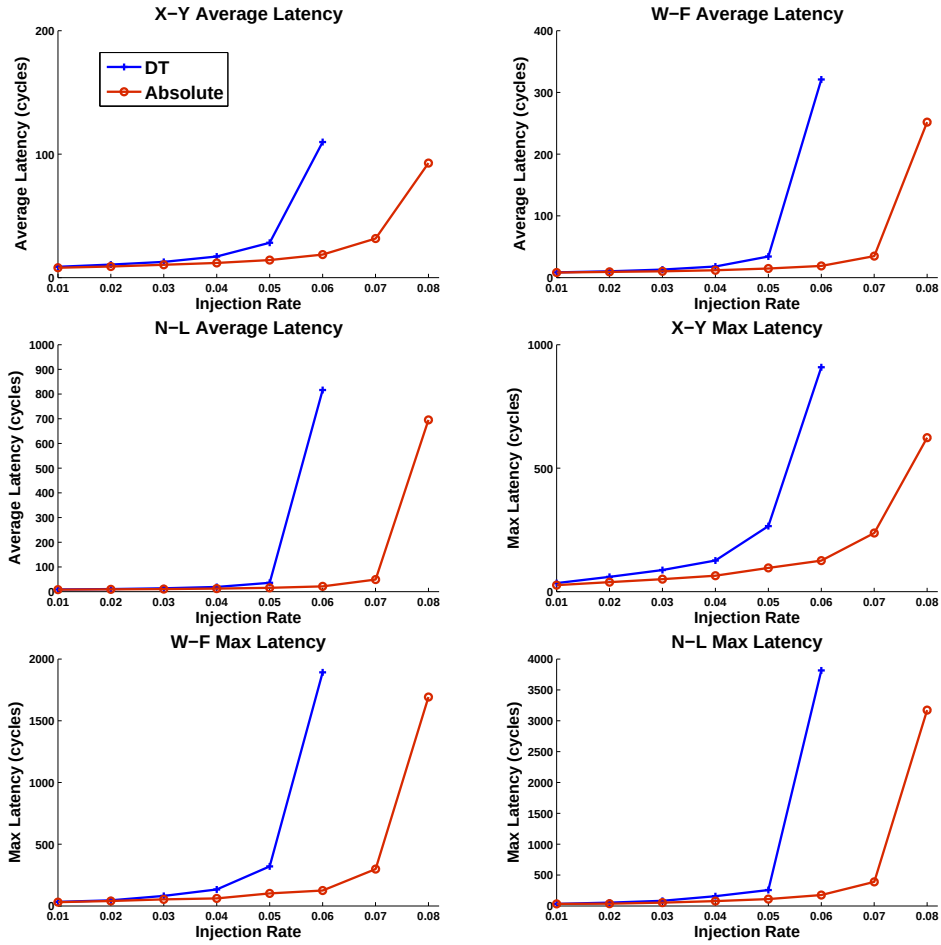


Figure 6.5 Comparison of latency in 4x4 NoC of different routing algorithms. (blue: the former router model)(red: the proposed router model)

but high latency is not be used in real applications. For example, 200-500 clock cycles are used in Ref. [152]. Thus the injection rate will set to a small value to keep the latency under 1000 cycles. The results of average latency and max latency in 4x4 NoC and 8x8 NoC are shown in Fig. 6.5 and Fig. 6.6 respectively, each of which contains twelve small figures, where the latency both methods are compared for each routing algorithm. To clarify the difference in latency and to make the comparison easy, we used an appropriate range suitable for each, that is, x-axis range is 0-0.08 for 4x4 NoC in Fig. 6.5, and 0-0.04 for 8x8 NoC in Fig. 6.6.

The performance of the routing by the DT method greatly depends on the algorithm to produce the sequence of tags, and it is controlled by the parameter of injection rate which is the speed with which a routing algorithm generates a packet to routers. If the injection rate is same, packets are

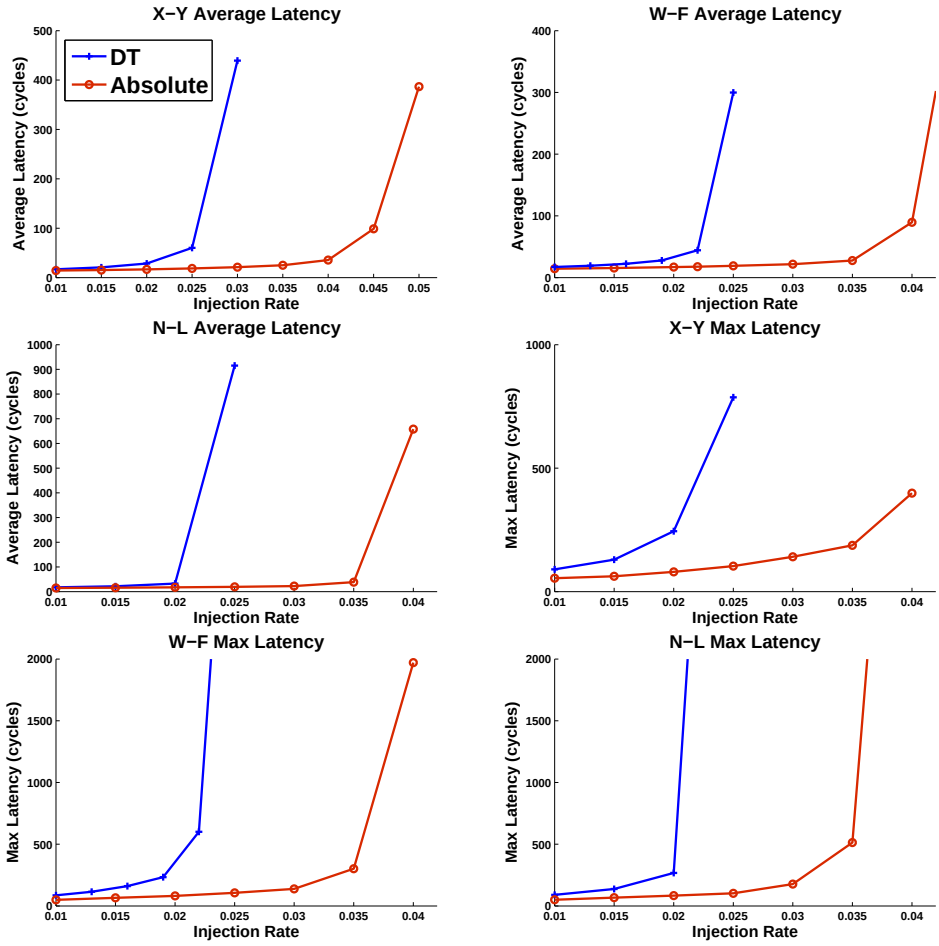


Figure 6.6 Comparison of latency in 8x8 NoC of different routing algorithms. (blue: the former router model)(red: the proposed router model)

randomly sent in the same speed, thus a sequence of tags is transmitted. Thus, these two methods are fairly compared by using three routing algorithms provided by the NOXIM NoC simulator under the same injection rate as a parameter. The results are mapped on the curves in Fig. 6.5 and Fig. 6.6.

From Fig. 6.5 and Fig. 6.6, the proposed router model could achieve low average latency and low max latency in different routing algorithms and different NoC sizes. Some special inflexion point of the injection rate is chosen to evaluate these methods in average latency consumption. The inflexion point may be a fair value to compare the proposed method and former method, where the reduction rate in this point could be thought as an average value. The reduction is smaller in the left of the inflexion point and it is larger in the right. When the injection rates of these three routing

algorithms X-Y, W-F, and N-L are 0.05 in 4x4 NoC, the average latency with the proposed router are reduced by 49.56%, 57.01% and 54.88%, respectively. When the injection rates in 8x8 NoC are 0.025, 0.022 and 0.022 for X-Y, W-F and N-L, respectively, the average latency are reduced by 69.17%, 60.03% and 61.50%, respectively.

When using DT method, the CPU time of using NOXIM NoC simulator for 4x4 NoC with XY, WF, NL routing algorithms are 20.7s, 20.8s, 21.3s, respectively. The CPU time of using NOXIM NoC simulator for 8x8 NoC with XY, WF, NL routing algorithms are 80.0s, 80.2s, 82.5s, respectively.

When using proposed method, the CPU time of using NOXIM NoC simulator for 4x4 NoC with XY, WF, NL routing algorithms are 18.2s, 18.8s, 19.0s, respectively. The CPU time of using NOXIM NoC simulator for 8x8 NoC with XY, WF, NL routing algorithms are 74.2s, 75.1s, 76.7s, respectively.

The latency T for transmitting L bits packet from the source to the destination in NoC with wormhole routers is expressed by $T = (L/BW + R) * H$, where BW is the link bandwidth in bits per cycle; R is the routing delay per hop; H is the number of hops from the source to the destination node [153] [154]. In our work, NoC architecture with the proposed router model can really reduce the packet size, and the packet size directly affects the latency. The bigger the NoC size is, the more the proposed method could reduce the packet size. Therefore, the reduction of latency becomes remarkable.

In our NoC-ANN design, the absolute address is more suitable. For example, Fig. 4.9 is NoC-ANN for implementing ANN with 5-7-3-7-5. With DT method, the neuron in left-top PE need transmit 3 hops to left hidden layer 1 (header size is 3 bits x 3 hops + 3 bits = 12 bits) and 4 hops to right hidden layer 1 (header size is 3 bits x 4 hops + 3 bits = 15 bits). With the absolute address method, the network size is 3x6, thus 2 bits for x-axis and 3 bits for y-axis. The header size is 2 bits + 3 bits + 3 bits = 8 bits. The header size with absolute address method is smaller than header size with DT method. The absolute address method is efficient then DT method.

6.4.2 Evaluation of proposed iNoC-ANN and other architecture of ANNs

In this subsection, three real applications of FF-ANNs are implemented with the proposed iNoC-ANN system to evaluate communication load and performance. These applications are pattern recognition with topology of 5-7-3-7-5 [115], neural control with topology of 5-12-8-4-1 [117] and nonlinear principal component analysis (NPCA) for image processing with topology of 4-10-1-10-4 [118].

Table 6.4 Comparison of communication load

Application	Architecture	Packet number	Packet size (bit)	Total (bit)
1. Pattern recognition	P2P	112	16	1792
	NoC	12	36/72/90	882
	iNoC[a]	12	**26/62/80**	**762**
2. Neural control	P2P	192	16	3072
	NoC	15	36/90	1188
	iNoC	15	**26/80**	**1038**
3. NPCA	P2P	100	16	1600
	NoC	12	36/54/90	842
	iNoC	12	**26/44/80**	**726**

[a] iNoC: the proposed improved NoC architecture

Reduction in communication load

For each packet, a header is indispensable and payload number is decided by active neurons. The communication load of three real applications are measured by using the proposed iNoC-ANN and compared with the former NoC-ANN architecture and general point-to-point (P2P) structure [2] as shown in Table 6.4. As far as we know, the traditional general P2P architecture does not aggregate some neurons in one PE. Without the NoC architecture and a packet switched transmission method, aggregating some neurons in one PE compulsively may reduce the communication load, but it will degrade the performance of the whole system, and hardware design will be more complicated and the communication time becomes larger. We assume each neuron in the same layer needs to transmit 1 packet/pattern to each neuron of the next layer.

The first application of ANN has 5, 7, 3, 7 and 5 neurons in each layer. With the P2P architecture, the number of packet is 112 (=5x7+7x3+3x7+7x5) and each packet has the same bit size of 16. Thus the total communication load is 1792 bits/pattern. If we use the former NoC architecture for transmission, the packet number and packet size are decided by the number of PEs and the number of neurons in each PE. Each PE of an input layer need transmit 2 packets to the next hidden layer and the packets size are 90 bits/pattern (=18 bits x (1 header + 4 payloads)) and 36 bits/pattern (=18 bits x (1 header + 1 payloads)). The other layers follow the same rule. Besides the header size of the packet, iNoC-ANN has the same data transmission method as NoC-ANN. The header size is 8 bits which consist of 1 bit for VC, 2 bits for FT, 2 bits for x-axis and 3 bits for y-axis.

Table 6.5 Comparison of CPS and CPSPW

Name	Structure	Precision	Neurons	CPS	CPSPW
MD-1220	FF	1-16b	8	9M	1.1M
NLX-420	FF	1-16b	16	300	18.75
Lneuro-1	FF	1-16b	16PE	26M	1.625M
N6400	SIMD	1-16b	64PE	870M	13.6M
HNC 100	SIMD	32b	100PE	250M	2.5M
MA-16	Matrix	16b	16PE	400M	1.56M
MT19003	FF	12b	8	32M	4M
WSI NAP	SIMD	9b×8b	576	138M	0.24M
NoC-ANN	NoC	1-16b	18PE,72	1351M	18.76M
iNoC-ANN	**iNoC**	**1-16b**	**20PE,80**	**1710M**	**21.38M**

With this proposed iNoC-ANN, communication load of total bit/pattern can be reduced by 54.6%∼66.2% compared with the existing P2P architecture, and by 12.6%∼13.8% compared with the former NoC-ANN. This reduction is mainly caused by the proposed absolute address based method which can reduce the header size of the packet.

System performance of Connection-Per-Second (CPS)

The most common performance rating is the Connection-Per-Second (CPS), which is defined as the rate of multiplication and accumulation operations. The value of CPS is normalized by the number of weights obtained from the connection per second per weight (CPSPW), which suggests the rating of performance for each solution. From the view point of the value of CPS and CPSPW, the proposed iNoC-ANN is compared with the former NoC-ANN and other existing hardware ANN [28][1] as shown in Table 6.5. The simulation CPU time for iNoC-ANN is 9.9s.

The proposed iNoC-ANN has a highest CPS and CPSPW. Compared with the former NoC-ANN, it could increase CPS and CPSPW by 26.6% and 13.97%, respectively. Compared with other existing hardware ANN, it could increase CPS and CPSPW at least 96.6% and 14.0%, respectively.

Thus, the proposed iNoC-ANN is superior in system performance to other NoC-ANN.

6.5 Conclusions

In this chapter, an NoC architecture with absolute address based routing strategy method is proposed for high performance hardware FF-ANN. An absolute address method based router model was developed. It can reduce the header size of the packet compared with the router model in the

former NoC and can implement different routing algorithms with a little hardware change. Simulation results show that it can reduce communication load about 54.6%∼66.2% compared with the existing P2P architecture and 12.6%∼13.8% compared with the former NoC architecture. It can also increase CPS by 96.6% and CPSPW by 14.0% compared with other hardware ANNs, and increase CPS 26.6% and CPSPW 13.97% compared with the former NoC-ANN. Consequently, the FF-ANN with the proposed NoC architecture is effective in reducing communication load and increasing performance.

Chapter 7

A High-performance Neural Processor Based on Network-on-Chip Architecture

This chapter describes a high-performance neural processor based on a novel Network on Chip (NoC) architecture to increase the computing speed and solve the reconfigurability and interconnection problems of general neural system. The proposed NoC-based neural processor is composed of 20 tiles in a 4x5 2-D array, and each tile includes a Process Element (PE) and a packet-switched router. In each PE, four neurons are aggregated to achieve low communication load. The network is 2-D torus topology, and it has a 32 G/s bandwidth and asynchronous clocking system. Our proposed neural processor is designed using 90-nm CMOS technology with one poly and nine metals. It can achieve over 3.1 Giga Connection Per Second (CPS) of computing speed while power dissipation is 1.1317 W at 1.2 V supply, and its chip size is 25 mm^2. Compared with the other existing digital neural networks, the proposed processor is reconfigurable and extendable, and can achieve lower communication load, lower system running time and higher computing speed.

7.1 Introduction

Artificial Neural Networks (ANN) have been proposed as one of well-known parallel computing systems to solve various problems. Especially the digital hardware ANNs are more popular due to the high precision and good expansibility [2]. However, digital architectures of ANN have major implementation issues such as higher cost due to large chip area and lower computing speed compared to analog systems [26]. Furthermore, the digital ANNs are usually not reconfigurable, that is, the digital ANN for some application can not be applied for other applications [28], and a heavy communication load is caused by its complex data transmission method.

In this chapter, the NoC architecture is described to structure a new type of neural processor named NoCNN. In the proposed NoCNN, four neurons are integrated in one PE (Processing Element) and connected with one 1-GHz 5-port wormhole-switched router to compose a tile. Twenty such tiles are integrated in a 4x5 2-D array as the whole NoCNN. Twenty tiles are enough to implement various applications and a torus topology is adopted for its simplicity. The proposed processor can overcome the above-mentioned problems of general digital ANN such as low computing speed, a lack of reconfigurability and heavy communication load.

The contribution of this chapter are as follows: (1) The neural processor by NoC architecture can work asynchronously in different tiles and work in parallel in the same tile to make the system computing speed of Connection-Per-Second (CPS) higher. (2) The proposed processor is reconfigurable, because weight value, activation function and implementation information can be easily changed by sending new packets. It is also expandable, because the tiles can be easily added or removed. Thus, it can overcome problems of various applications. (3) The packet transmission method of NoC is more efficient and intelligent than a general digital ANN, thus it can reduce communication load. We also design our proposed NoCNN using 90-nm CMOS technology to evaluate the performance.

The remainder of this chapter is organized as follows. Section 7.2 gives an overview of the related works and our design motivation. Section 7.3 describes the key building blocks of neural processor with NoC architecture. Section 7.4 presents performance evaluation and implementation results. Section 7.5 concludes by summarizing the NoCNN processor.

7.2 Related Works

The architecture of digital ANN has good expansibility, high precision and a lot of EDA tools which support the digital hardware implementation [125]. Different kinds of architectures were proposed for digital ANN, such as slice architecture [155], systolic array devices [49], single instruction multiple data (SIMD) [147], and so on. These architectures make hardware ANN develop quickly, but they have some drawbacks. For example, the performance of computing speed of the slice architecture and the reconfigurability of systolic array architecture are not good. The SIMD architecture has a little improvement in performance, whereas the complex mapping method limits it [26]. Furthermore, most of these architectures are suffer from the interconnection problem because of the global point-to-point shared buses.

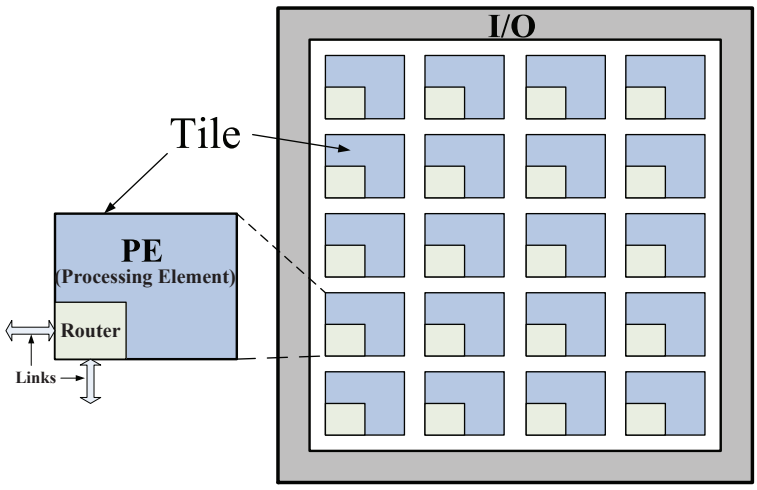


Figure 7.1 20 tiles NoCNN processor

These years, a lot of research groups contributed their effort to improve the digital ANN. However, the design method is still based on the existing architectures. A new design method with new architecture is required to overcome the problems of interconnection, computing speed, reconfigurability. NoC is such an architecture that can solve the communication problem and support high computing speed for large size SoC design [75][68], such as Intel 80-tile NoC chip [156]. Therefore, we focuses on proposing NoC architecture to build a new type of digital ANN to overcome the drawbacks in the traditional ANNs.

7.3 NoC Architecture for Building Neural Computing Processor

The proposed NoCNN processor contains 20 tiles arranged in a 4x5 2-D torus network. The overview of the architecture is shown in Fig. 7.1. In this section, the key building blocks for whole processor are described in detail. One neuron is designed and then four neurons are integrated in one PE (Processing Element), and the router is designed to connect with PE as one tile. Also the packet format of NoC is described. Finally, such 20 tiles are connected with each other as 2-D torus topology to build the NoCNN processor.

7.3.1 PE Architecture

The most common ANN is a multi-layer perceptron as shown in Fig. 7.2. It has one input layer, several hidden layers and one output layer. Neurons need to transmit the computation results to

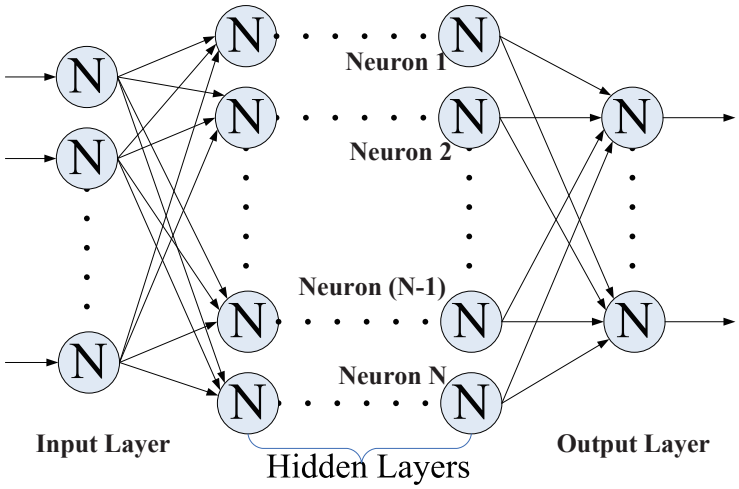


Figure 7.2 Structure of a multi-layer perceptron

every neurons in the next layer. The computation can be described by the the Eqs.(7.3.1) and (7.3.2) same as [58]

$$H_k^{(s)} = \sum_{j=1}^{N_{s-1}} w_{kj}^{(s)} o_j^{(s-1)} \ . \tag{7.3.1}$$

$$o_k^{(s)} = f(H_k^{(s)}) \ . \tag{7.3.2}$$

$H_k^{(s)}$ is a weighted sum of the k^{th} neuron in the s^{th} layer; $o_j^{(s-1)}$ is an output of the j^{th} neuron in the $(s-1)^{th}$ layer, N_{s-1} denotes the total number of neurons in the $(s-1)^{th}$ layer which connect with this k^{th} neuron. $f(H_k^{(s)})$ is an activation function computed on the weighted sum $H_k^{(s)}$. Logarithmic sigmoid function Eq.(7.3.3) and hyperbolic tangent sigmoid function Eq.(7.3.4) are typical activation functions.

$$f(x) = 1/(1+e^{-x}) \ . \tag{7.3.3}$$

$$f(x) = (e^{x} - e^{-x})/(e^{x} + e^{-x}) \ . \tag{7.3.4}$$

From these equations, we know that one hardware neuron requires a high performance and high precision multiplication block for computation and memory block for holding weight value. Besides, an adder block and an activation function block are also required [1]. Block diagram of a single neuron is shown in Fig. 7.3. RAM is used to store weight values of this neuron and controlled

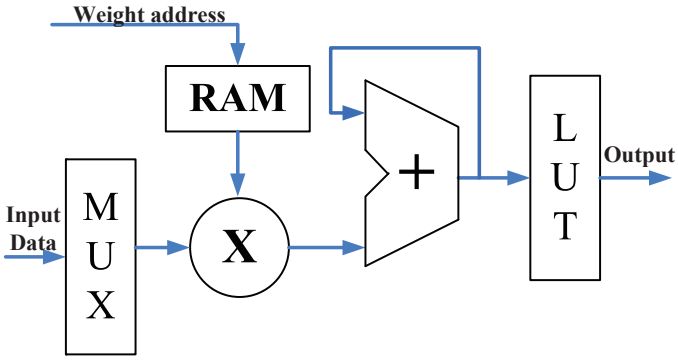


Figure 7.3 A single neuron architecture

via the weight address. The input data are selected by MUX (multiplexer) and then multiplied with weight value by the multiplier. When all the products have been added, this sum will be used for searching output via the activation function. The activation function is implemented by means of a lookup table (LUT) which is stored in RAM. 32-bit fixed-point architecture is used for our design to suit the requirement of higher precision compared with 16-bit fixed point. One sign bit, six integer bits, and twenty-five fraction bits can cover the range of [-64,64) with a quantization error of 2.98023224E-8.

In our design, four neurons in one layer are integrated in one Processing Element (PE) for reducing total transmission packet and communication load. These four neurons share one LUT to reduce design cost. The PE also requires a decoder, an encoder, a controller and a weight address generator as shown in Fig. 7.4. When the input packet arrived at the PE, a decoder decodes the neuron address to get the number of the neuron to be used and transmits the decoded input to each neuron in this PE. When the Controller receives the decoded address information from the decoder, it controls the weight address generator to generate the virtual address and transmit to each neuron. Then the generated address will control the RAM which stores the weight values. When all neurons in this PE complete their calculation and LUT task, the outputs of them are transmitted to the encoder. It holds outputs as one single packet. In this packet, each payload part comes from each neuron and header part comes from one RAM which is loaded in advance of design.

If more than four neurons are designed in one PE, it could reduce the communication load more, whereas the system performance may be low. Also we observe that the total number of neurons of a general digital ANN [1] is always every fourth such as 4, 8, 12, and so on. Therefore, 4 neurons in one PE could achieve the better trade-off between performance and communication load. We will explain the reduction of communication load in Sect. 7.4.

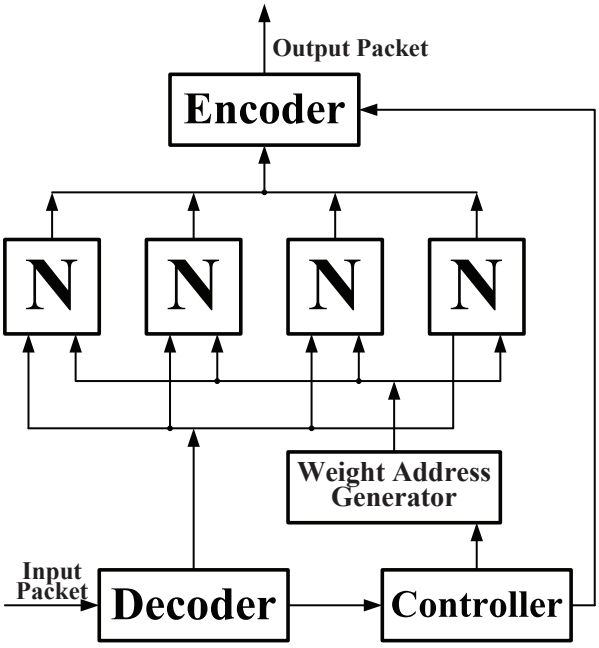


Figure 7.4 Architecture of a PE

7.3.2 Router Architecture and Packet Format

An 1GHz 5-port wormhole packet-switched router of 32 GB/s bandwidth is designed. Each port has two links for dead-lock free routing and reducing the latency. The block diagram of this 4-stage pipeline router and pipelining are shown in Fig. 7.5. Five input ports and output ports are connected with four directions of west, south, east and north and one PE. Each header flit needs to be processed through the steps of Routing Computation (RC), Virtual-channel Allocation (VA), Switch Allocation (SA) and Switch Traversal (ST). RC and VA steps will not be processed for payload flits. Assume that a header flit arrives at an input port, and it needs to be decoded and buffered according to the flit information by RC block in the first stage. The header flit then chooses a channel by VA block in the second stage. The flit travels through the selected virtual-channel and chooses a proper output port by SA block in the third stage. In this stage, each SA block will get 5-bit input from each input port and send 5-bit output signal to switch crossbar to control the output port. Flit travels through the crossbar switch in the fourth stage.

The schematic diagram of the switch allocator is shown in Fig. 7.6. It consists of three parts: decoder, arbiter and hold logic. In the decoder, 5-bit from header of the input packet is decoded, where 2-bit is used for partitioning the type of the flit, and 3-bit for output choice. The second part

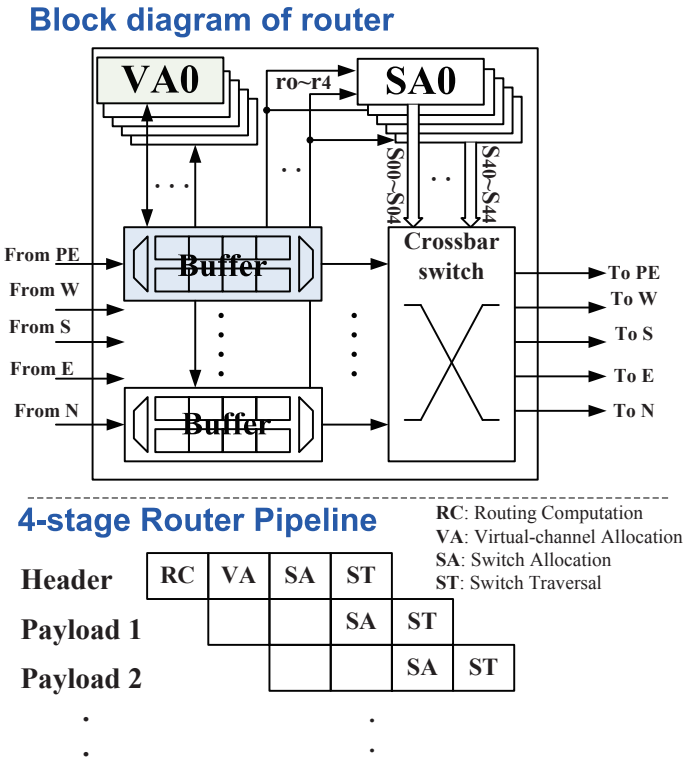


Figure 7.5 Block diagram of router and pipeline processing

is a fixed-priority arbiter. The port-s04 which connects with PE will get a high priority, because the data processing time in PE is longer than that in router transmission. The hold logic part is designed to hold the selected port for the payload.

The NoC packet format and routing protocol are shown in Fig. 7.7. Each packet contains one header and some payloads, and the number of payload depends on the number of neurons that are used in this PE connecting with this router. Thus the number of payload is always one, two, three or four. The header information contains 1 bit for VCN (Virtual Channel Number), 2 bits for FT (Flit Type), 4 bits for UN (Used Neuron), 15 bits for PCI (PE Control Information), and 3 bits for each DA (Destination Address). One packet can store four DAs. The payload contains 2 bits for FT and 32 bits for DFN (Data from the Former Neuron).

7.3.3 Architecture of 4x5 NoCNN processor

The proposed 4x5 NoCNN processor consists of the key building blocks of PE and a router arranged in a 4x5 2D torus topology. We investigated real applications of ANNs in various fields and

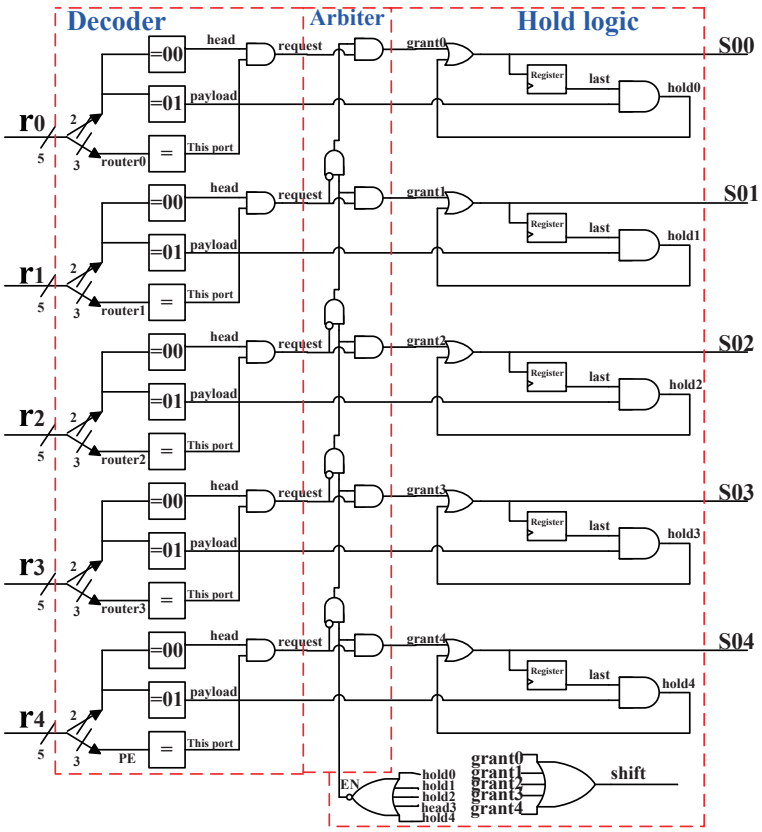


Figure 7.6 Schematic diagram of the switch allocator 0 for controlling north output port

implemented five of them [116, 143, 119, 144, 157] as shown in Fig. 7.8.

In Fig. 7.8, three colors means different states of tiles, respectively. Blue means Whole-Work tile where all neurons in the PE of this tile work, yellow means Partly-Work tile where some of neurons work, and white means Sleep tile where no neurons work. Fig. 7.8(a) shows how to implement an ANN with 3, 20, 20, 1 neurons in each layer: One PE in first column of the network is used as input layer (three neurons in this PE work, the rest one neuron does not work); five PEs in the second column of the network are used as the first hidden layer; five PEs in the third column of the network are used as the second hidden layer; one PE in the fourth column of the network is used as an output layer (one neuron in this PE work, other three neurons need not work). The other four applications use the same implementation method as shown in Fig. 7.8(b), (c), (d) and (e). The 4x5 NoCNN processor can implement different applications with at most 20-20-20-20 architecture. If more neurons are required for another application, the network topology can be easily extended by adding the tiles. And packet information of address control may be a little different. But, the

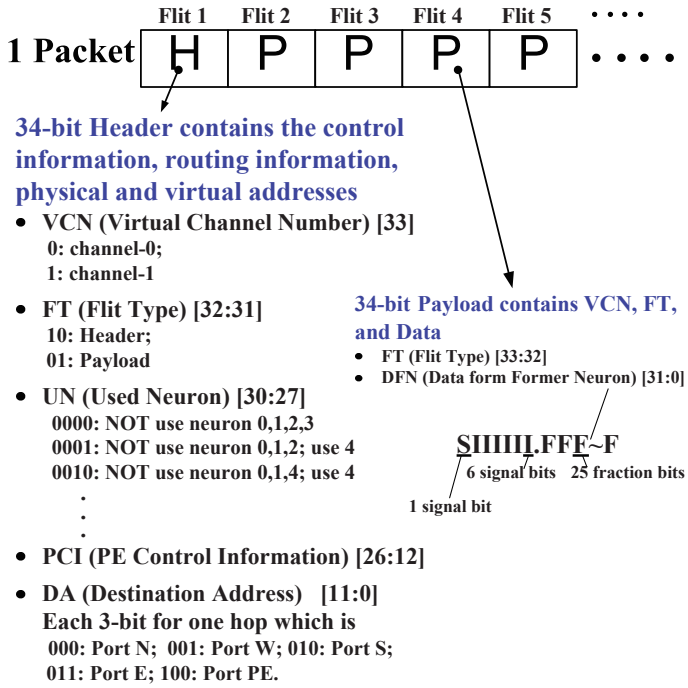


Figure 7.7 NoC Packet Format and its information

implementation method is universal.

7.4 Implementation and Performance Evaluation

The 4x5 NoCNN processor is designed using the tools of Synopsys Design Compiler and IC Compiler in 90-nm CMOS process technology with one Poly and nine metals [158]. The functional blocks of the chip and individual tile are layouted as shown in Fig. 7.9. The chip size is 25 mm^2, and it contains about 61 million 2NAND cells. Each 1.152 mm^2 tile contains about 3 million cells and dissipating 1.1317 W power at 1.2 V supply.

The features and the performance of the proposed NoCNN processor are as follows.

7.4.1 Reconfigurability and Extensibility

The proposed NoCNN processor is reconfigurable. We know that the information which stored in RAM could be easily changed according to the character of RAM. The weight value and activation function are stored in RAMs in our proposed system, thus they can be changed for different ANN applications by reloading the data into RAMs. Also, different topology of the ANN can be

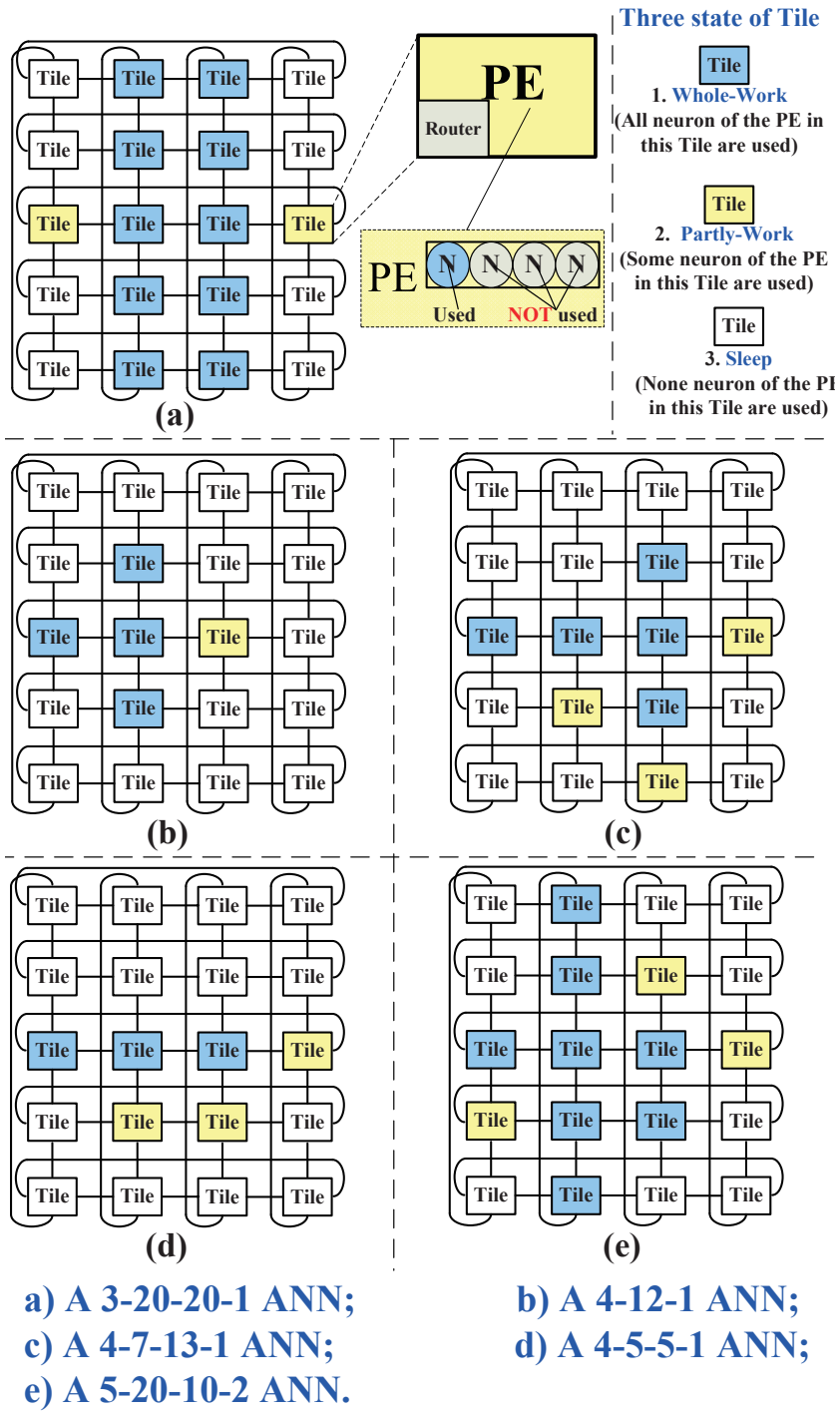
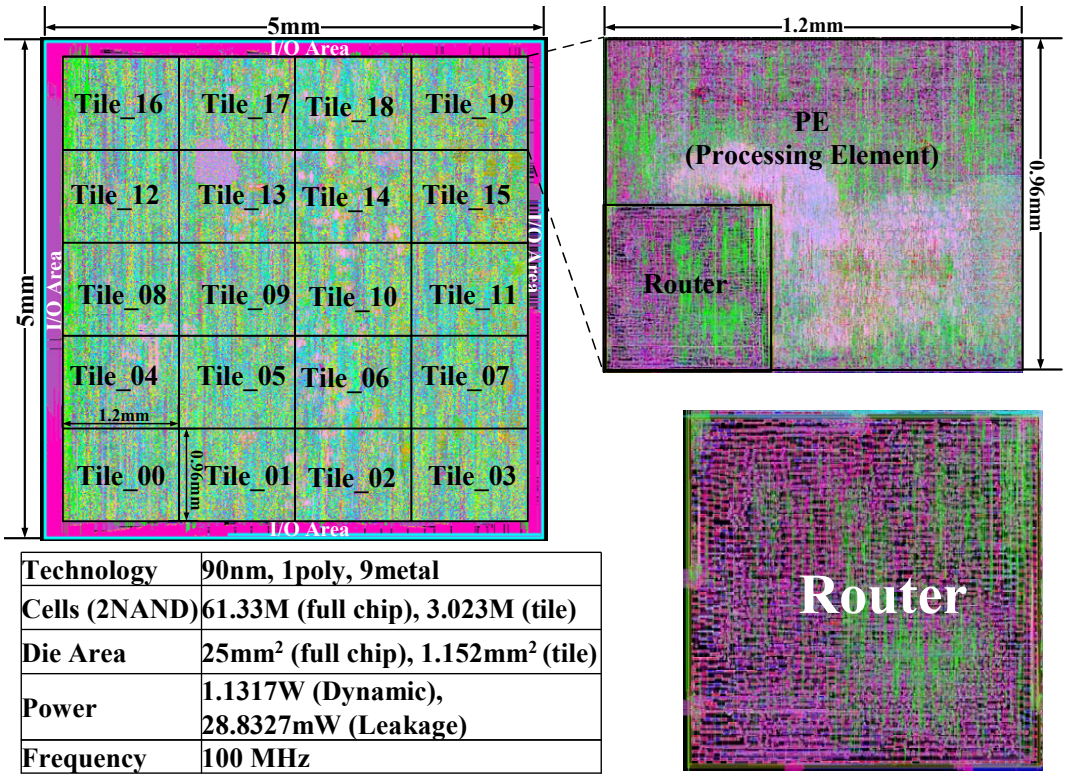


Figure 7.8 4x5 NoCNN processor for five ANN applications



Technology	90nm, 1poly, 9metal
Cells (2NAND)	61.33M (full chip), 3.023M (tile)
Die Area	25mm^2 (full chip), 1.152mm^2 (tile)
Power	1.1317W (Dynamic), 28.8327mW (Leakage)
Frequency	100 MHz

Figure 7.9 NoCNN processor layout design

implemented on the same processor which is controlled by sending new packets.

The NoCNN processor is extendable. We use a hierarchical design flow in a building block style. The key block of PE and a router are designed separately to compose the tile. This tile is added into the ASIC design library as a hardcore. In this work, 4x5 NoCNN processor is structured by such 20 hardcores and arranged as 4x5 torus topology. We could easily add or delete any tile of NoCNN to meet the size of a target ANN.

These features of reconfigurability and extensibility make NoCNN implementation possible for different applications in the same processor without any hardware change, while the other existing hardware ANNs must be changed for each application.

7.4.2 Communication Load Evaluation

Traditional digital ANNs use bus-based point to point (P2P) architecture for the data transmission. However, the proposed NoCNN processor uses NoC architecture to transmit the data by routers. We assume each neuron in the same layer needs to transmit 1 packet/pattern to each neuron of the

Table 7.1 Comparison of communication load

Topology	Type	Packet number	Packet size (bit)	Total size (bit/pattern)
3-20-20-1	P2P	480	32	15360
	NoC	35	136,170	**5780**
4-12-1	P2P	60	32	1920
	NoC	6	170	**1020**
4-7-13-1	P2P	132	32	4224
	NoC	14	68,136,170	**2108**
4-5-5-1	P2P	50	32	1600
	NoC	8	68,170	**1054**
5-20-10-2	P2P	320	32	10240
	NoC	35	68,101,170	**4182**

next layer. The communication load of five real applications (see Sect. 7.3.3) is evaluated by using the proposed NoC methodology and compared with a general P2P bus architecture, as shown in Table 7.1. For example, the first application of ANN has 3, 20, 20 and 1 neurons in each layer. With the P2P architecture, the number of packets is 480 (3x20+20x20+20x1) and each packet has the same bit size of 32. Thus the total communication load is 15360 bits/pattern. If we use the NoC architecture for transmission, the packet number and packet size are decided by the number of tiles and the number of neurons in each tile. From Fig. 7.8(a), the tile for input layer needs to transmit 5 packets to the next hidden layer and the packet size is 136 bits/pattern (34 bits x (1 header + 3 payloads)). The other layers follow the same rule. Thus the total communication load is 5780 bits/pattern (136 bits x 5 + 170 bits x 30).

As shown in Table 7.1, the communication load of total packet size per pattern can be reduced by 34.4%∼62.4% with the proposed NoC architecture. This reduction is caused by the smart packet-based data transmission mechanism.

7.4.3 Performance Evaluation

Performance is evaluated for these five applications by NIRGAM NoC simulator [135]. The latency of each application is shown in Fig. 7.10, where the red bar means the amount of communication on the southward channel of the router, the blue bar that on the northward channel, the green bar that on the eastward channel and the yellow bar that on the westward channel.

The three of the five applications are chosen for analysis and comparison between our design,

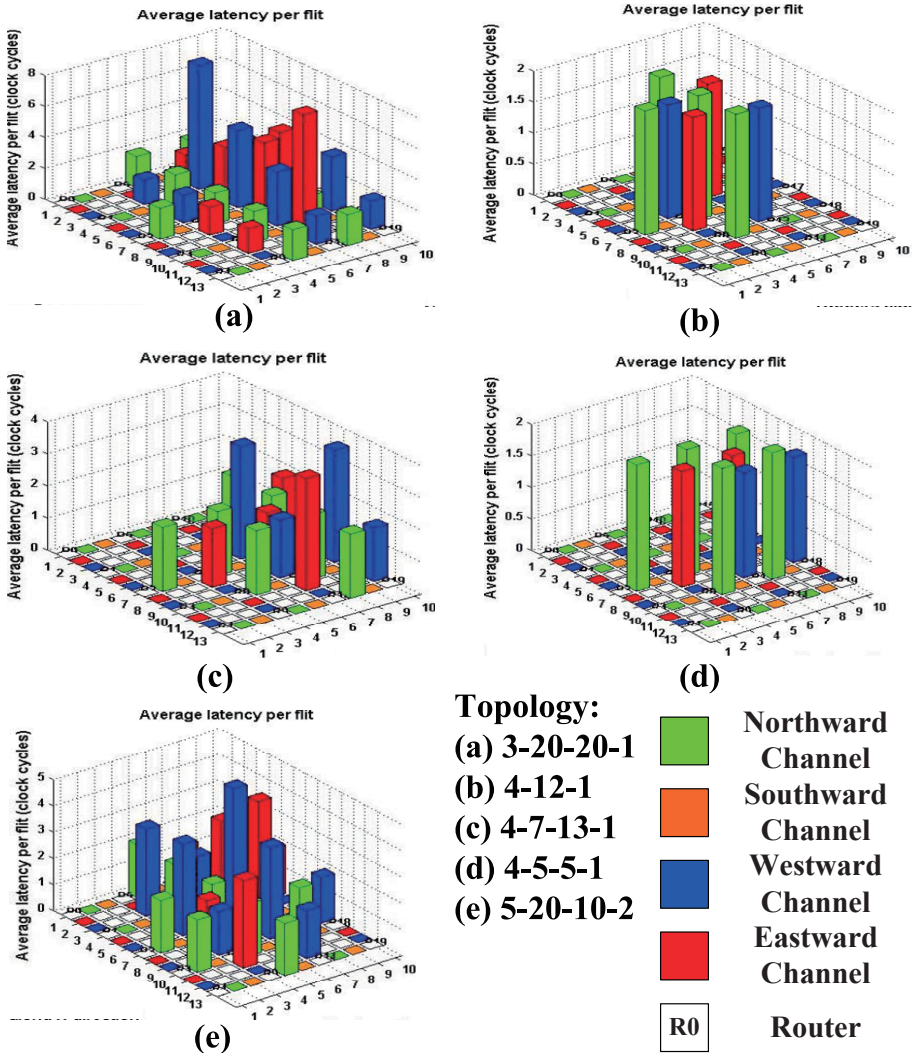


Figure 7.10 Average latency per flit of three applications

and the others. The critical path of the first application is appeared in the date transmission from the first hidden layer to the second hidden layer. It costs 37.06ns. The system running time of the second and the third applications are 7.664ns and 22.163ns, respectively. From the result, we surmise that the latency depends on the topology, especially the number of the neurons of an input and a hidden layers. The CPU time of using NIRGAM NoC simulator for three applications are 11.2s, 7.8s, 8.6s, respectively.

The performance is evaluated by the Connection-Per-Second (CPS), which is defined as the rate of multiplication and accumulation operations. However, some systems could get a high performance of computing speed according to the large number of neurons. So the value of CPS is normalized by dividing it by the number of weights, that is, the connection per second per weight number (CPSPW). CPS and CPSPW for first three applications are compared between our proposed NoCNN processor and other existing digital ANNs [28][1][107] as shown in Table 7.2. For the detailed information of the other ANNs listed here, please refer to references.

In Table 7.2, MD-1220 was one of the first implementations of slice architecture for ANN, but it could not implement three applications. WSI is a digital ANN with SIMD architecture, but it could not implement three applications for the low precision. MA-16 is a digital ANN with systolic array architecture, it could not implement three applications due to the limit number of neurons. Comparing with other digital ANNs, the proposed NoCNN processor has the highest performance of CPS and CPSPW for different applications. Compared with the existing hardware ANNs with Max CPS and CPS of application-1, application-2 and application-3, the proposed NoCNN processor can increases CPS by 158%, 268%, 63.8% and 189%, respectively. And compared with the existing hardware ANNs with CPSPW of application-1, application-2 and application-3, the proposed NoCNN processor can increases them by 266%, 61.3% and 189%, respectively. We analyze this result as follows: compared with the slice architecture (e.g. Lneuro-2.3) which used a bus control, the NoC control mechanism is more suited for the Multi-Processor SoC; the digital ANN with SIMD architecture (e.g. N64000, NC3001 and NM6403) can just implement ANN layer by layer, whereas the NoCNN processor could implement different layers' neurons at the same time.

The proposed NoCNN processor is also discussed comparing the system running time for three different applications with existing digital ANN as shown in Fig. 7.11, where system running time means average time per pattern to finish all transmission. From Fig. 7.11, the proposed NoCNN processor has the lowest system running time compared with other four digital ANNs.

Table 7.2 Comparison of CPS and CPSPW

Name (year)	Architecture	Precision	Neurons	CPS				CPSPW		
				Max[a]	App1[b]	App2	App3	App1	App2	App3
Micro devices MD-1220(94)	Slice	1x16b	8	8.9M	na[c]	na	na	na	na	na
Philips Lneuro-2.3(04)	Slice	16, 32b	12PE	720M	440M	340M	300M	0.92M	5.76M	2.27M
Inova N64000(02)	SIMD	1-16b	64PE	870M	149M	76.9M	70.31M	0.31M	1.28M	0.53M
Hitachi WSI(02)	SIMD	9x8b	144	300M	na	na	na	na	na	na
Neuricam NC3001(04)	SIMD	32b	1, 32	1G	344M	177M	195.31M	0.72M	2.95M	1.48M
RC Module NM6403(04)	SIMD	64x64b	1, 64	1.2G	206M	106M	116.97M	0.43M	1.77M	0.89M
Siemens MA-16(94)	Systolic array	16b	16PE	400M	na	na	na	na	na	na
NoC ANN 4x5 NoCNN	**NoC**	**1-32b**	**25PE**	**3.1G**	**1.62G**	**557M**	**867M**	**3.37M**	**9.29M**	**6.57M**

[a] Max: the highest CPS of the neural processor or chip which can be find in reference.
[b] App1: Application 1
[c] na: not available

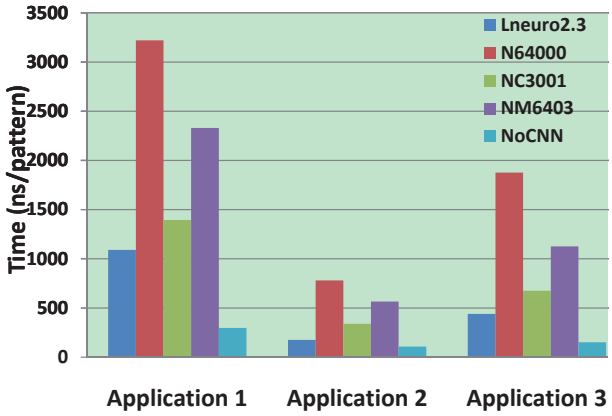


Figure 7.11 Comparison of system running time

7.5 Conclusion

This chapter presents an NoC design for a 20-tile reconfigurable and high performance NoCNN processor in a 90-nm CMOS technology. It could achieve 3.1G CPS, while power dissipation is 1.1317 W at 1.2 V supply and 25 mm^2 chip area. The simulation results indicate that the proposed NoCNN processor is effective in increasing performance of computing speed and reducing communication load. From these results, our proposed architecture is promising for higher performance and lower communication load of a digital ANN.

Chapter 8

Conclusions

Artificial Neural Networks (ANN) have been proposed as one of well-known parallel computing systems to solve various problems and researched more than half of the century. Hardware implementation of ANN was researched very actively in the 1990's, while its development remains stagnant in these years according to the design cost and interconnection problems. This thesis focuses on improving the existing FPGA-based ANN and building a new NoC architecture for ANN.

- In Chapter 1 and 2, a historical review, research motivation and preliminaries were explained.
- In Chapter 3, an FPGA-based general architecture for the implementation of a multilayer ANN was proposed. The circuit for each application can be easily generated by setting the parameter values to match the particular network size and running the synthesis. Similarly to a particular network, the best solution of the architecture design of pipeline and layer multiplexing is calculated by MATLAB procedure, by returning the optimal pipeline depth and the control signal value in each state of the control block (FSM). We exploited the capability of a given FPGA board by assigning the proper pipeline depth, so that a higher resource utilization rate, global forwarding speed and high performance were achieved. Our proposed architecture makes an FPGA implementation easy for a given ANN at a short time by varying the data path. It also provides the feasibility to perform a larger neural network in a popular FPGA board at a relatively higher speed by using the partly pipeline method. So it is possible to develop a neural device for commercial or industrial application by our method.
- In Chapter 4, a sophisticated NoC architecture with off-chip learning was proposed to satisfy various applications of the complex feedforward neural network. We designed this system aiming at low latency, high throughput and low power consumption. This system is reconfigurable, because the weight values and activation functions can be changed as desired. We can

also change the topology and routing algorithms of the NoCs by sending new data to meet different kinds of feedforward neural networks, so this system is easily extended. We can design this system in the style of cell by cell and can easily add or remove any cell to comply with different applications. The proposed NoC system can reduce the communication load of total packet size and improve the system performance of CPS. This proposed NoC mapping method can make the digital ANN more efficient.

- In Chapter 5, a multiple NoC models are designed for low power and implemented for hardware ANN. This NoC architecture can implement both the small size ANN and the large size ANN. The proposed architecture is reconfigurable to suit for different applications of FF-ANN, by changing the weight value, activation function, and the number of neurons and layers. The hardware of it is reparable which the faulty neurons and channels can be replaced by good one. Compared to the traditional P2P data transmission method, it can reduce the communication load about 30.25%-58.4%, and it can increase system performance of CPS about 25%-47.1% compared with the existing hardware ANN. The number of PE in the proposed multiple NoC model is 16, but it can be changed arbitrarily.

- In Chapter 6, an NoC architecture with absolute address based routing strategy method is proposed for high performance hardware FF-ANN. An absolute address method based router model was developed. It can reduce the header size of the packet compared with the router model in the former NoC and can implement different routing algorithms with a little hardware change. Simulation results show that it can reduce communication load about 54.6%$\sim$66.2% compared with the existing P2P architecture and 12.6%$\sim$13.8% compared with the former NoC architecture. It can also increase CPS by 96.6% and CPSPW by 14.0% compared with other hardware ANNs, and increase CPS 26.6% and CPSPW 13.97% compared with the former NoC-ANN. Consequently, the FF-ANN with the proposed NoC architecture is effective in reducing communication load and increasing performance.

- In Chapter 7, an NoC design for a 20-tile reconfigurable and high performance NoCNN processor in a 90-nm CMOS technology. It could achieve 3.1G CPS, while power dissipation is 1.1317 W at 1.2 V supply and 25 mm^2 chip area. The simulation results indicate that the proposed NoCNN processor is effective in increasing performance of computing speed and reducing communication load. From these results, our proposed architecture is promising for higher performance and lower communication load of a digital ANN.

In conclusion, our proposed hardware design for ANN could solve the performance problem of FPGA-based ANN and it could suit for the real applications which need high performance. Furthermore, NoC architecture was proposed instead of the traditional bus-based P2P hardware ANN to solve the problems of performance, communication load and reconfigurability.

Finally, we list some future work about NoC-ANN as follows:

- **Application specific NoC (ASNoC) for ANN.** Different from the regular NoC architecture, the topology of ASNoC is designed according to the application, which results in high performance and low power consumption. But the design complexity of ASNoC is much higher than the regular NoC, and the ASNoC architecture could used for neural network.
- **Other developments for NoC-ANN.** Such as new routing algorithm, a new floorplan, redesigned neuron, and so on.

In the long history, ANN ebbed and flowed. The new technology always leads its progress. Looking into the future, with the development of NoC and other new technology, the ANN will have another flow.

Publication List

Journal Paper

1. Y. Dong, C. Li, Z. Lin, and T. Watanabe, “A Hybrid Layer-multiplexing and Pipeline Architecture for Efficient FPGA-based Multilayer Neural Network,” *IEICE NOLTA*, 2011. (to be published)

2. Y. Dong, C. Li, Z. Lin, H. Zhang, and T. Watanabe, “High performance feedforward neural network mapped by noc architecture with a new routing strategy implementation method,” *Journal of Signal Processing (JSP)*, vol. 15, no. 3, pp. 113–122, Mar. 2011.

3. Y. Dong, C. Li, Z. Lin, and T. Watanabe, “Multiple network-on-chip model for high performance neural network,” *IEEK trans. Journal of Semiconductor Technology and Science (JSTS)*, vol. 10, no. 2, pp. 28–36, May 2010.

4. Y. Dong, C. Li, K. Kumai, Y. H. Li, Y. Wang, and T. Watanabe, “A new flexible network on chip architecture for mapping complex feedforward neural network,” *Journal of Signal Processing (JSP)*, vol. 13, no. 6, pp. 453–462, Nov. 2009.

Conference Paper with Review

5. C. Li, Y. Dong, and T. Watanabe, “Region based Placement Algorithm for Low-power FPGA Architecture,” in *Proc. ISLPED’11*, Aug. 2011. (accepted)

6. C. Li, Y. Dong, and T. Watanabe, “New Power-Efficient FPGA Design Combining with Region-Constrained Placement and Multiple Power Domains,” in *Proc. NEWCAS’11*, Jun. 2011. (accepted)

7. Y. Dong, C. Li, H. Liu, and T. Watanabe, “A high performance digital neural processor design by network on chip architecture,” in *Proc. VLSI-DAT’11*, Apr. 2011, pp. 243–246.

8. Z. Lin, Y. Dong, Y. Li, and T. Watanabe, “A hybrid architecture for efficient fpga-based implementation of multilayer neural network,” in *Proc. APCCAS’10*, Dec. 2010, pp. 616–619.

9. Y. Dong, Z. Lin, and T. Watanabe, “An efficient hardware routing algorithms for noc,” in *Proc. TENCON’10*, Nov. 2010, pp. 1525–1530.

10. C. Li, Y. Dong, and T. Watanabe, “A novel low power fpga architecture,” in *Proc. FIT’10*, Sep. 2010, pp. 65–68.

11. Y. Dong, Z. Lin, Y.Li, and T. Watanabe, “High performance implementation of neural networks by networks on chip with 5-port 2-virtual channels,” in *Proc. ISCAS’10*, May 2010, pp. 381–384.

12. Y. Dong, Z. Lin, and T. Watanabe, “High performance autoassociative neural network using network on chip,” in *Proc. ICISE’09*, Dec. 2009, pp. 4015–4018.

13. Y. Dong, K. Kumai, Z. Lin, Y. H. Li, and T. Watanabe, “High dependable implementation of neural networks with networks on chip architecture and a backtracking routing algorithm,” in *Proc. PrimeAsia’09*, Nov. 2009, pp. 404–407.

14. Y. Dong, Y. Wang, Z. Lin, and T. Watanabe, “High performance and low latency mapping for neural network into network on chip architecture,” in *Proc. IEEE ASICON’09*, Oct. 2009, pp. 891–894.

15. Y. Dong and T. Watanabe, “Mixed noc architecture for mapping complex feedforward neural network,” in *Proc. NCSP’09*, Mar. 2009, pp. 609–612.

16. Y. Dong and T. Watanabe, “High performance noc architecture for two hidden layers bp neural network,” in *Proc. ISOCC’08*, Nov. 2008, pp. 269–272.

Bibliography

[1] B. Muller, J. Reinhardt, and M. Strickland, Neural Networks: An Introduction (Physics of Neural Networks), Springer, 2002.

[2] D. Graupe, Principles of Artificial Neural Networks, Advanced Series in Circuits and Systems, World Scientific, 2007.

[3] T. Shima and M. Tukada, Neural Network and Neural device, Morikita publisher, 1997.

[4] W.S. McCulloch and W. Pitts, “A logical calculus of the ideas immanent in nervous activity,” Bulletin of Mathematical Biology, vol.5, no.4, pp.115–133, Dec. 1943.

[5] F. Rosenblatt, Principles of neurodynamics, Spartan Books, 1962.

[6] M. Minsky and S. Papert, Perceptrons: An Introduction to Computational Geometry, MIT, 1969.

[7] W.A. Little, “The existence of persistent states in the brain,” Bulletin of Mathematical Biology, vol.19, no.1-2, pp.101–120, Feb. 1974.

[8] J. Hopfield, “Neural networks and physical systems with emergent collective computational abilities,” Proc. National Academy of Sciences, p.2554C2558, 1982.

[9] P.J. Werbos, “Generalization of backpropagation with application to a recurrent gas market model,” Neural Networks, vol.1, pp.339–356, 1988.

[10] P.J. Werbos, “Backpropagation: Past and future,” IJCNN88, pp.343–353, 1988.

[11] P.J. Werbos, “Backpropagation through time: what it does and how to do it,” Proceedings of the IEEE, vol.78, pp.1550–1560, 1990.

[12] D.E. Rumelhart, G.E. Hinton, and R.J. Williams, “Learning representations by back-propagating errors,” Nature, vol.323, pp.533–536, Oct. 1986.

[13] D.E. Rumelhart, G.E. Hinton, and R.J. Williams, Learning internal representations by error propagation, Chapter 1, MIT Press, 1986.

[14] B. Denby, “The use of neural networks in high energy physics,” MIT, vol.5, no.4, pp.505–549, Jul. 1993.

[15] A. Singera, “Implementations of artificial neural networks on the connection machine,” Parallel Computing, vol.14, no.3, pp.305–315, Aug. 1990.

[16] D.A. Pomerleau, “Neural network simulation at warp speed: How we got 17 million connections per second,” Proc. IJCNN’88, pp.143–150, 1988.

[17] G. Chinn, K. Grasjki, C. Chen, C. Kuszmaul, and S. Tomboulian, “Systolic array implementation of neural nets on the MasPar MP-1 massively parallel processor,” Proc. IJCNN’90, pp.169–173, 1990.

[18] S. Shams and J. Gaudiot, “Massively parallel implementations of neural networks: a comperative analysis,” IEEE Trans. Computer, vol.14, no.3, pp.305–315, Aug. 1990.

[19] M. Witbrock and M. Zagha, “An implementation of back-propagation learning on gf11, a large simd parallel computer,” Parallel Computing, vol.14, pp.329–346, 1990.

[20] T. Watanabe, Y. Sugiyama, T. Kondo, and Y. Kitamura, “Neural network simulation on a massively parallel cellular array processor: Aap-2,” Proc. IJCNN’89, pp.155–161, 1989.

[21] L. Vuurpijl, “Using transputer systems for neural network simulations,” Proc. SNN Symposium on Neural Networks, pp.27–28, 1992.

[22] C. Leung and R. Setiono, “Efficient neural network training algorithm for the Cray Y-MP supercomputer,” Proc. IJCNN’93, pp.1943–1946, 1993.

[23] M. Yasunaga, N. Masuda, M. Yagyu, M. Asai, M. Yamada, and A. Masaki, “Design, fabrication and evaluation of a 5-inch wafer scale neural network LSI composed of 576 digital neurons,” Proc. IJCNN’90, pp.527–535, Jun. 1990.

[24] N. Mauduit, M. Duranton, and J. Gobert, “Lneuro 1.0: A piece of hardware lego for building neural network systems,” IEEE Trans. Neural Networks, vol.3, no.3, pp.414–422, May 1992.

[25] A.R. Omondi, J.C. Rajapakse, and M. Bajger, FPGA Implementations of Neural Networks, Springer, 2006.

[26] C.S. Lindsey, “Neural networks in hardware: Architectures, products and applications,” lecture notes of Neural Networks, Aug. 2002.

[27] J.N.H. Heemskerk, “Overview of neural hardware. neurocomputers for brain-style processing. design, implementation and application,” Draft version of PhD Thesis, 1995.

[28] C.S. Lindsey and T. Lindblad, “Review of hardware neural networks: a user’s perspective,” Plenary talk given at 3rd Workshop on Neural Networks, 1994.

[29] M. Holler, “VLSI implementation of learning and memory systems: a review,” Advances in Neural Information Processing Systems, vol.3, 1991.

[30] E.V. Keulan, S. Colak, H. Withagen, and H. Hegt, “Neural networks hardware performance criteria,” Proc. IJCNN’94, pp.1885–1888, 1994.

[31] C.S. Lindsey and B. Denby, “A study of the intel etann VLSI neural network for an electron isolation trigger,” Fermi National Accelerator Laboratory, Oct. 1992.

[32] M. Duranton, “L-neuro 2.3: a VLSI for image processing by neural networks,” Proc. Microelectronics for Neural Networks, pp.157–160, 1996.

[33] D. Hammerstrom, “A VLSI architecture for high-performance, low-cost, onchip learning,” Proc. Neural Networks, pp.537–544, 1990.

[34] M. Yasunaga, N. Masuda, M. Yagyu, M. Asai, K. Shibata, M. Ooyama, and M. Yamada, “A self-learning neural network composed of 1152 digital neurons in wafer-scale lsis,” Proc. IJCNN’91, pp.1844–1849, 1991.

[35] P. Ienne, I. f Ecublens, and G. Kuhn, “Digital systems for neural networks,” Digital Signal Processing Technology, volume CR57 of Critical Reviews Series, pp.314–345, 1995.

[36] A. Mano, “A processor approach to build an artificial neural network,” Proc. ICCA’03, pp.147–154, 2003.

[37] S. McBader, L. Clementel, A. Sartori, A. Boni, and P. Lee, “Softtotem: An fpga implementation of the totem parallel processor,” Lecture Notes in Computer Science, vol.2438, pp.63–70, 2002.

[38] P.A. Chevtchenko, D.V. Fomine, V.M. Tchernikov, and P.E. Vixne, “Using of microprocessor nm6403 for neural net emulation,” Proc. Virtual Intelligence or Dynamic Neural Networks, pp.242–252, Mar. 1999.

[39] J. Beichter, N. Bruels, E. Meister, U. Ramacher, and H. Klar, “Design of general-purpose neural signal processor,” Proc. Microelectronics for Neural Networks, pp.311–315, 1991.

[40] J.P. LeBouquin, “Ibm microelectronics zisc, zero instruction set computer,” Proc. World Congress on Neural Networks, pp.5–9, 1994.

[41] ZISC78 Datasheet, Silicon Recognition, Inc., May 2002.

[42] B.E. Boser, E. Sackinger, J. Bromley, Y. leCun, and L.D. Jackel, “Hardware requirements for neural network pattern classifiers: A case study and implementation,” IEEE Micro, vol.12, no.1, pp.32–40, 1992.

[43] J. Alspector, A. Jayakumar, and S. Luma, “Experimental evaluation of learning in a neural microsystem,” Proc. NIPS’91, pp.871–878, 1992.

[44] P. Masa, K. Hoen, and H. Wallinga, “70 input, 20 nanosecond pattern classifier,” Proc. IJCNN’94, pp.1854–1859, 1994.

[45] H. Eguchi, T. Furuta, H. Horiguchi, S. Oteki, and T. Kitaguchi, “Neural network LSI chip with on-chip learning,” Proc. IJCNN’91, pp.453–456, 1991.

[46] A. Kramer, “Array-based analog computation: principles, advantages and limitations,” Proc. MicroNeuro, pp.68–79, 1996.

[47] Y. Choi, K. Ahn, and S. Lee, “Effects of multiplier output offsets on on-chip learning for analog neuro-chips,” Neural Processing Letters, vol.4, pp.1–8, 1996.

[48] C. Lehmann, M. Viredaz, and F. Blayo, “A generic systolic array building block for neural networks with onchip learning,” IEEE Trans. Neural Network, vol.4, no.3, pp.400–407, May 1993.

[49] S. Mahapatra and R.N. Mahapatra, “Mapping of neural network models onto systolic arrays,” Journal of Parallel and Distributed Computing, vol.60, no.6, pp.677–689, Jun. 2000.

[50] Q. Wang, A. Li, Z.C. Li, and Y. Wan, “A design and implementation of reconfigurable architecture for neural networks based on systolic arrays,” Advances in Neural Networks, no.3973, pp.1328–1333, 2006.

[51] U. Muller, A. Gunzinger, and W. Guggenbuhl, “Fast neural net simulation with a dsp processor array,” IEEE Trans. Neural Networks, vol.6, no.1, pp.203–213, 1995.

[52] I. Milosavlevich, B. Flower, and M.J. PANNE, “a parallel computing engine for connectionist simulation,” Proc. MicroNeuro, pp.363–368, 1996.

[53] J. Kennedy and J. Austin, “A parallel architecture for binary neural networks,” Proc. MicroNeuro, pp.225–231, 1997.

[54] M. Schaefer, T. Schoenauer, C. Wolff, G. Hartmann, H. Klar, and U. Ruckert, “Simulation of spiking neural networks - architectures and implementations,” Neurocomputing, vol.48, no.1-4, pp.647–679, 2002.

[55] B. Girau, “Building a 2d-compatible multilayer neural network,” Proc. IJCNN’00, pp.59–64, 2000.

[56] B. Girau, “Fpna: interaction between fpga and neural computation,” Journal on Neural Systems, vol.10, no.3, pp.243–259, 2002.

[57] B. Girau, “On-chip learning of fpga-inspired neural nets,” Proc. IJCNN’01, pp.222–227, 2001.

[58] A. Omondi and J. Rajapakse, FPGA Implementations of Neural Networks, Springer, 2006.

[59] V. Agarwal, M.S. Hrishikesh, S.W. Keckler, , and D. Burger, “Clock rate versus ipc: the end of the road for conventional microarchitectures,” Proc. Annual International Symposium on Computer Architecture, pp.248–259, 2000.

[60] A. Iyer and D. Marculescu, “Power and performance evaluation of globally asynchronous locally synchronous processors,” Proc. Annual International Symposium on Computer Architecture, pp.158–168, 2002.

[61] M. Krstic, E. Grass, F.K. Guerkaynak, and P. Vivet, “Globally asynchronous, locally synchronous circuits: Overview and outlook,” IEEE Design & Test of Computers, vol.24, no.5, pp.430–441, 2007.

[62] L. Benini and G.D. Micheli, “Networks on chips: A new soc paradigm,” IEEE Computer, vol.35, no.1, pp.70–78, 2002.

[63] T. Claasen, “An industry perspective on current and future state-of-the-art in system-on-chip (soc) technology,” Proceedings of the IEEE, vol.94, no.6, pp.1121–1137, 2006.

[64] R.X. Gu and M.I. Elmasry, “Power dissipation analysis and optimization of deep submicron cmos digital circuits,” IEEE JOURNAL OF SOLID-STATE CIRCUITS, vol.31, no.5, pp.707–713, May 1996.

[65] Q.T. Huang, F. Piazza, P. Orsatti, and T. Ohguro, “The impact of scaling down to deep submicron on cmos rf circuits,” IEEE JOURNAL OF SOLID-STATE CIRCUITS, vol.33, no.7, pp.1023–1036, Jul. 1998.

[66] D. Sylvester and K. Keutzer, “A global wiring paradigm for deep submicron design,” IEEE Trans. COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, vol.19, no.2, pp.242–252, Feb. 2000.

[67] J.W. Mcpherson, “Reliability challenges for 45nm and beyond,” Proc. DAC’06, pp.176–181, Jul. 2006.

[68] L. Benini and G.D. Micheli, Networks On Chips: Technology and tools, Morgan Kaufmann, 2005.

[69] T. Mudge, “Power: A first-class architectural design constraint,” IEEE Computer, vol.34, no.4, pp.52–58, Apr. 2001.

[70] V. Raghunathan, M.B. Srivastava, and R.K. Gupta, “A survey of techniques for energy efficient on-chip communication,” Proc. DAC’03, pp.900–905, Jun. 2003.

[71] P. Rashinkar, P. Paterson, and L. Singh, System-on-a-Chip Verification - Methodology and Techniques, Kluwer Academic, 2001.

[72] W. Yang, M.K. Chung, and C.M. Kyung, “Current status and challenges of soc verification for embedded systems market,” Proc. SOC’03, pp.213–216, Sep. 2003.

[73] A. Allan, D. Edenfeld, J.W. Joyner, A.B. Kahng, M. Rodgers, , and Y. Zorian, “2001 technology roadmap for semiconductors,” IEEE Computer, vol.35, no.1, pp.42–53, 2002.

[74] T. Schattkowsky, “Uml 2.0 - overview and perspectives in soc design,” Proc. DATE’05, pp.822–833, 2005.

[75] A. Jantsch and H. Tenhunen, Networks on Chip, Kluwer Academic Publishers, 2003.

[76] A. Hemani, A. Jantsch, S. Kumar, A. Postula, J. Öberg, M. Millberg, and D. Lindqvist, “Network on chip: An architecture for billion transistor era,” Proc. IEEE NorChip Conference, Nov. 2000.

[77] P. Guerrier and A. Greiner, “A generic architecture for on-chip packetswitched interconnections,” Proc. DATE’00, pp.250–256, Mar. 2000.

[78] W.J. Dally and B. Towles, “Route packets, not wires: On-chip interconnection networks,” Proc. DAC’01, Jun. 2001.

[79] M. Sgroi, M. Sheets, A. Mihal, K. Keutzer, S. Malik, J. Rabaey, and A. Vincentelli, “Addressing the system-on-a-chip interconnect woes through communication-based design,” Proc. DAC’01, pp.667–672, Jun. 2001.

[80] E. Rijpkema, K. Goossens, and P. Wielage, “A router architecture for networks on silicon,” Proceedings of Progress 2001, 2nd Workshop on Embedded Systems, Veldhoven, the Netherlands, Oct. 2001.

[81] A. Jantsch, J.P. Soininen, M. Forsell, L.R. Zheng, S. Kumar, M. Millberg, and J. Oberg, “Networks on chip,” Workshop at the European Solid State Circuits Conference, Sept. 2001.

[82] M. Keating and P. Bricaud, Reuse Methodology Manual for System-on-Chip Designs, Kluwer Academic, 1998.

[83] T. Thomas, “Technology for ip reuse and portability,” IEEE Design & Test of Computers, vol.16, no.4, pp.6–15, Oct. 1999.

[84] H. Chang, L. Cooke, M. Hunt, G. Martin, A. Mcnelly, and L. Todd, Surviving the SOC Revolution - A Guide to Platform-Based Design, Kluwer Academic, 1999.

[85] P.G. Paulin, C. Pilkington, M. Langevin, E. Bensoudane, D. Lyonnard, O. Benny, B. Lavigueur, D. Lo, G. Beltrame, V. Gagne, and G. Nicolescu, “Parallel programming models for a multiprocessor soc platform applied to networking and multimedia,” IEEE Trans. VLSI Systems, vol.14, no.7, pp.667–680, Jul. 2006.

[86] J. Duato, S. Yalamanchili, and N. Lionel, Interconnection Networks: An Engineering Approach, Morgan Kaufmann, 2002.

[87] C. Leiserson, “Fat-trees: Universal networks for hardware-efficient supercomputing,” IEEE Trans. Computers, vol.C-34, no.10, pp.892–901, Oct. 1985.

[88] W. Dally, “Express cubes: Improving the performance of k-ary n-cube interconnection networks,” IEEE Trans. Computers, vol.40, no.9, pp.1016–1023, Sep. 1991.

[89] C. Ciordas, T. Basten, A. Rădulescu, K. Goossens, and J. van Meerbergen, “An event-based network-on-chip monitoring service,” ACM Trans. Design Automation of Electronic Systems, vol.10, no.4, pp.702–723, Oct. 2005. HLDVT’04 Special Issue on Validation of Large Systems.

[90] E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, “Qnoc: Qos architecture and design process for network on chip,” Journal of Systems Architecture, vol.50, no.2-3, pp.105–128, Feb. 2004.

[91] K. Lee, S.J. Lee, and H.J. Yoo, “A distributed crossbar switch scheduler for on-chip networks,” Proc. Custom Integrated Circuits Conference, pp.671–674, Sep. 2003.

[92] K. Lee, S.J. Lee, and H.J. Yoo, “A high-speed and lightweight on-chip crossbar switch scheduler for on-chip interconnection networks,” Proc. International Conference on European Solid-State Circuits, pp.453–456, Sep. 2003.

[93] K. Goossens, J. Dielissen, O.P. Gangwal, S.G. Pestana, A. Rbdulescu, and E. Rijpkema, “A design flow for application-specific networks on chip with guaranteed performance to accelerate soc design and verification,” Proc. DATE’05, pp.1182–1187, 2005.

[94] M. Millberg, E. Nilsson, R. Thid, and A. Jantsch, “Guaranteed bandwidth using looped containers in temporally disjoint networks within the nostrum network on chip,” Proc. DATE’04, pp.890–895, Feb. 2004.

[95] P. Guemer and A. Greiner, “A generic architecture for on-chip packet-switched interconnections,” Proc. International Conference on European Solid-State Circuits, pp.250–256, DATE’00.

[96] U.Y. Ogras, J. Hu, and R. Marculescu, “Key research problems in noc design: A holistic perspective,” Proc. Hardward/Software Codesign and System Synthesis, pp.69–74, Sep. 2005.

[97] L. Shang, L.S. Peh, and N.K. Jha, “Dynamic voltage scaling with links for power optimization of interconnection networks,” Proc. IEEE Computer Society, HPCA’03, pp.91–102, 2003.

[98] L. Jain, “Fusion of neural nets, fuzzy systems and genetic algorithms in industrial applications,” IEEE Trans. Industrial Electronics, vol.46, no.6, pp.1049–1136, 1999.

[99] D.S. Kim and J.S. Park, “Modeling network intrusion detection system using feature selection and parameters optimization,” IEICE Trans. Information and Systems, vol.E91-D, no.4, pp.1050–1057, Apr. 2008.

[100] J.F. Wang, J.C. Wang, A.N. Suen, C.H. Wu, and F.M. Li, “VLSI architecture and implementation for speech recognizer based on discriminative bayesian neural network,” IEICE Trans. Fundamentals, vol.E85-A, no.8, pp.1861–1869, Aug. 2002.

[101] G. Dede and M.H. Sazli, “Speech recognition with artificial neural networks,” Digital Signal Processing, vol.20, no.3, pp.763–768, May 2010.

[102] R. Coggins, M. Jabri, B. Flower, and S. Pickard, “A hybrid analog and digital VLSI neural network for intracardiac morphology classification,” IEEE Journal of Solid-state Circuits, vol.30, no.5, pp.542–550, May 1995.

[103] Y. Maeda and M. Wakamura, “Simultaneous perturbation learning rule for recurrent neural networks and its fpga implementation,” IEEE Trans. Neural Network, vol.16, no.6, pp.1664–1672, 2005.

[104] S.P. Johnston, G. Prasad, L. Maguire, and T.M. Mcginnity, “An fpga hardware/software co-design towards evolvable spiking neural networks for robotics application,” International Journal of Neural Systems, vol.20, no.6, pp.447–461, Dec. 2010.

[105] F.J. Lin, J.C. Hwang, P.H. Chou, and Y.C. Hung, “Fpga-based intelligent-complementary sliding-mode control for pmlsm servo-drive system,” IEEE Trans. Power Electronics, vol.25, no.10, pp.2573–2587, Oct. 2010.

[106] A. Mellit, H. Mekki, A. Messai, and H. Salhi, “Fpga-based implementation of an intelligent simulator for stand-alone photovoltaic system,” Expert Systems with Applications, vol.37, no.8, pp.6036–6051, Aug. 2010.

[107] F.M. Dias, A. Antunes, and A.M. Mota, “Artificial neural networks: a review of commercial hardware,” Engineering Applications of Artificial Intelligence, vol.17, no.8, pp.945–952, Aug. 2004.

[108] J. Zhu and P. Sutton, “Fpga implementations of neural networks - a survey of a decade of progress,” Proc. FPL’03, pp.1062–1066, Sep. 2003.

[109] L.M. Reyneri, “Implementation issues of neurofuzzy hardware: Going toward hw/sw codesign,” IEEE Trans. Neural Network, vol.14, no.1, pp.176–194, 2003.

[110] H. Li and S. D. Zhang, “A stochastic digital implementation of a neural network controller for small wind turbine systems,” IEEE Trans. Power Electronics, vol.21, no.5, pp.1502–1507, 2006.

[111] D. Ferrer and R. Gonzalez, “Neurofpga - implementing artificial neural networks on programmable logic devices,” Proc. DATE’04, pp.218–223, Feb. 2004.

[112] S. Himavathi, “Feedforward neural network implementation in fpga using layer multiplexing for effective resource utilization,” IEEE Trans. Neural Networks, vol.18, no.3, pp.880–888, 2007.

[113] I.A. Basheer and M. Hajmeer, “Artificial neural networks: Fundamentals, computing, design, and application,” Journal of Microbiological Methods, vol.43, pp.3–31, 2000.

[114] Xilinx, http://www.xilinx.com/.

[115] A.K. Jain, R.P.W. Duin, and J.C. Mao, “Statistical pattern recognition: A review,” IEEE Trans. Pattern Analysis and Machine Intelligence, vol.22, no.1, pp.4–37, Jan. 2000.

[116] A. Ouchar, R. Aksas, and H. Baudrand, “Artificial neural network for computing the resonant frequency of circular patch antennas,” Microwave and optical technology letters, vol.47, no.6, pp.564–566, Oct. 2005.

[117] G.V. Puskorius and L.A. Feldkamp, “Neurocontrol of nonlinear dynamical systems with kalman filter trained recurrent networks,” IEEE Trans. Neural Network, vol.5, no.2, pp.279–297, Mar. 1994.

[118] M.A. Kramer, “Nonlinear principal component analysis using autoassociative neural networks,” Journal of AICHE, vol.37, no.2, pp.233–243, 1991.

[119] M. Rajendra, P. Jena, and H. Raheman, “Prediction of optimized pretreatment process parameters for biodiesel production using ann and ga,” Fuel, vol.88, no.5, pp.868–875, 2009.

[120] S. Chen, C.F.N. Cowan, and P.M. Grant, “Orthogonal least squares learning algorithm for radial basis function networks,” IEEE Trans. Neural Network, vol.2, no.2, pp.302–309, NoV. 1991.

[121] I. Akitoshi, Y. Hiroshi, T. Ichi, and H. Masayasu, “Multi-layer neural network for anomalous signal detection from elf band electromagnetic wave,” Signal Processing, vol.13, no.1, pp.55–65, Jan. 2009.

[122] C. Cortes and V. Vapnik, “Support-vector networks,” Machine Learning, vol.20, no.3, pp.273–297, Sep. 1995.

[123] K.J. Hunta, D. Sbarbaroa, R. bikowskia, and P. Gawthrop, “Neural networks for control systemsła survey,” Automatica, vol.28, no.6, pp.1083–1112, Nov. 1992.

[124] M. Kolehmainen, H. Martikainen, and J. Ruuskanen, “Neural networks and periodic components used in air quality forecasting,” Atmospheric Environment, vol.35, no.5, pp.815–825, 2001.

[125] L. Gatet, H. Tap-Beteille, and F. Bony, “Comparison between analog and digital neural network implementations for range-finding applications,” IEEE Trans. Neural Network, vol.20, no.2, pp.460–470, Mar. 2009.

[126] D. Kim, H. Kim, H. Kim, G. Han, and D.J. Chung, “A simd neural network processor for image processing,” Proc. ISNN’05, May 2005.

[127] Y. Fan and M. Paindavoine, “Implementation of an rbf neural network on embedded systems: real-time face tracking and identity verification,” IEEE Trans. Neural Network, vol.14, no.5, pp.1162–1175, Sep. 2003.

[128] J. Nurmi, H. Tenhunen, J. Isoaho, and A. Jantsch, Interconnect-Centric Design for Advanced SOC and NOC, Kluwer Academic Publisher, 2004.

[129] Y. Dong and T. Watanabe, “Network on chip architecture for bp neural network,” Proc. IEEE ICCCAS’08, pp.1083–1087, May 2008.

[130] Y. Dong and T. Watanabe, “Network on chips structure for mapping two hidden layers bp-anns,” Proc. 23rd ITC-CSCC’08, pp.601–604, Jul. 2008.

[131] Y. Dong and T. Watanabe, “Mixed noc architecture for mapping complex feedforward neural network,” Proc. NCSP’09, pp.609–612, Mar. 2009.

[132] A.K. Jain, J.C. Mao, and K.M. Mohiuddin, “Artificial neural networks: a tutorial,” Computer, vol.29, no.3, pp.31–44, 1996.

[133] L.H. Jordan and E.B. Thomas, “Backpropagation simulations using limited precision calculations,” Proc. PIJCNN, pp.121–126, 1991.

[134] F. Fazzino, M. Palesi, and D. Patti, “Noxim noc simulator,” http://sourceforge.net/projects/nox.

[135] L. Jain, “Nirgam,” University of Southampton UK, http://www. nirgam.ecs.soton.ac.uk.

[136] J. Dvorak, “Brain maker,” http://www.calsci.com/BrainIndex.html.

[137] AlteraInc., Stratix Device Handbook, Jan. 2006.

[138] L. Smith, “Implementing neural models in silicon,” Handbook of Nature-Inspired and Innovative Computing Section 11, 2006.

[139] S. Vitabile, A. Gentile, G.B. Dammone, and F. Sorbello, “Mlp neural network implementation on a simd architecture,” Neural Nets, vol.2486, 2002.

[140] C. Ciordas, K. Goossens, T. Basten, A. Radulescu, and A. Boon, “Transaction monitoring in networks on chip: The on-chip run-time perspective,” Proc. IES’06, pp.1–10, Oct. 2006.

[141] Y. Dong, C. Li, K. Kumai, Y.H. Li, Y. Wang, and T. Watanabe, “A new flexible network on chip architecture for mapping complex feedforward neural network,” Journal of Signal Processing (JSP), vol.13, no.6, pp.453–462, Nov. 2009.

[142] Y. Dong, Y.H. Li, Y. Wang, and T. Watanabe, “Low power and high speed network on chip architecture for bp neural network,” Proc. ITC-CSCC’09, pp.298–301, Jul. 2009.

[143] H.M. Yao, H.B. Vuthaluru, M.O. Tade, and D. Djukanovic, “Artificial neural network-based prediction of hydrogen content of coal in power station boilers,” Fuel, vol.84, no.12–13, pp.1535–1542, Sep. 2005.

[144] X.M. Chen and D.Z. Chen, “Measuring average particle size for fluidized bed reactors by employing acoustic emission signals and neural networks,” Chemical engineering and technology, vol.31, no.1, pp.95–102, Dec. 2007.

[145] S. Murali, D. Atienza, P. Meloni, S. Carta, L. Benini, G. Micheli, and L. Raffo, “Synthesis of predictable networks-on-chip-based interconnect architectures for chip multiprocessors,” IEEE Trans. VLSI Systems, vol.15, no.8, pp.869–880, Aug. 2007.

[146] Y. Dong, H. Zhang, Z. Lin, and T. Watanabe, “A novel hardware method to implement a routing algorithm onto network on chip,” Proc. ICCCAS’10, pp.852–856, Jul. 2010.

[147] D.C. Hendry, A.A. Duncan, and N. Lightowler, “Ip core implementation of a self-organizing neural network,” IEEE Trans. neural networks, vol.14, no.5, pp.1085–1096, Sep. 2003.

[148] A. Jahnke, U. Roth, and H. Klar, “a simd/dataflow architecture for a neurocomputer for spike-processing neural networks (nespinn),” MicroNeuro’96, pp.232–237, Feb. 1996.

[149] W. Dally and B. Towles, Principles and Practices of Interconnection Networks, Morgan Kaufmann Publishers, 2004.

[150] S. Vangal and et al., “An 80-tile sub-100-w teraflops processor in 65-nm cmos,” IEEE Journal of Solid-State Circuits, vol.43, no.1, pp.29–41, 2008.

[151] K. Petersen and J. Oberg, “Toward a scalable test methodology for 2d-mesh network-on-chips,” Proc. DATE’07, pp.367–372, Apr. 2007.

[152] R. Ricardo, M. Vincent, and H. Paul, VLSI-SoC: Advanced Topics on Systems on a Chip: A Selection of Extended Versions of the Best Papers of the Fourteenth International Conference on Very Large Scale Integration of System on Chip (VLSI-SoC2007), Springer, 2007.

[153] Z. Lu, A. Jantsch, and I. Sander, “Feasibility analysis of messages for on-chip networks using wormhole routing,” Proceedings of the Asian Pacific Design Automation Conference, 2005.

[154] Z. Lu and A. Jantsch, “Flit ejection in on-chip wormhole-switched networks with virtual channels,” Proceedings of the IEEE NorChip Conference, Nov. 2004.

[155] N. Mauduit, M. Duranton, J. Gobert, and J. Sirat, “Fuzzy artmap: a neural network architecture for incremental supervised learning of analog multidimensional maps,” IEEE Trans. Neural Networks, vol.3, no.3, pp.414–422, May 1992.

[156] S. Vangal and et al., “An 80-tile 1.28tflops network-on-chip in 65nm cmos,” Proc. ISSCC’07, pp.5–7, Feb 2007.

[157] M. Chakraborty, C. Bhattacharya, and S. Dutta, “Studies on the applicability of artificial neural network (ann) in emulsion liquid membranes,” Journal of Membrane Science, vol.220, no.1-2, pp.155–164, Aug. 2003.

[158] Synopsys Inc., www.synopsys.com/Community/UniversityProgram.