TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY
Julkaisu 742 • Publication 742

Xin Wang

Designing Globally-Asynchronous Locally-Synchronous On-Chip Communication Networks

Tampere 2008

Tampereen teknillinen yliopisto. Julkaisu 742
Tampere University of Technology. Publication 742

Xin Wang

Designing Globally-Asynchronous Locally-Synchronous On-Chip Communication Networks

Thesis for the degree of Doctor of Technology to be presented with due permission for public examination and criticism in Tietotalo Building, Auditorium TB104 at Tampere University of Technology, on the 3rd of June 2008, at 12 noon.

Tampereen teknillinen yliopisto - Tampere University of Technology
Tampere 2008

ISBN 978-952-15-1987-1 (printed)
ISBN 978-952-15-2005-1 (PDF)
ISSN 1459-2045

ABSTRACT

This thesis addresses two aspects of designing on-chip communication networks. One is about applying Globally-Asynchronous Locally-Synchronous (GALS) communication scheme into Network-on-Chip (NoC). Another is of designing and realizing different types of on-chip communication structures in the frame of GALS scheme.

The work of applying GALS scheme into on-chip networks presented in this thesis includes the strategy of realizing GALS scheme in a NoC, synchronization method in a GALS NoC, and asynchronous circuit design. GALS scheme is applied in the NoC designs presented in this thesis by applying synchronous style in the communications between network nodes and their attached function hosts while applying asynchronous style in the communications among network nodes. The asynchronous circuits developed for realizing the GALS on-chip networks include an asynchronous First-In First-Out (FIFO) design, control pipeline structures, C-element structure, and an arbiter design.

Three different types of on-chip networks are designed and presented in this thesis, which include a direct network, a Code-Division Multiple-Access (CDMA) network, and a crossbar network. The direct on-chip network presented in this thesis is a bidirectional ring network which gives an example of realizing GALS scheme in Proteo NoC architecture. The ring network realization consists of six nodes and requires an area of 177K equivalent gates when it is realized with a 0.18μm standard-cell library of Application Specific Integrated Circuits (ASIC). Although the ring network has a scalable network structure, its data transfer latency can vary largely depending on the data destination and routing process. This drawback increases the difficulty for the ring network to provide constant quality of communication service.

Therefore, a network structure which applies CDMA technique is developed and presented in this thesis in order to provide non-blocking data transfers among network nodes so that data transfer latencies have small variances. The CDMA NoC achieves this feature by applying orthogonal codes to build non-blocking data transfer channels among network nodes. The six-node realization of CDMA NoC presented in this thesis has an area of 272K equivalent gates when it is realized with a 0.18μm standard-cell library and the data path width is 32 bits. The compensation of the larger area cost is that the asynchronous data transfer latency in the six-node CDMA NoC is equivalent to the best-case latency in the ring network. When the data path width is 32 bits, the realized CDMA network can transfer a 96-bit payload packet between network nodes within 49*ns* through a four-phase handshake protocol if there is no congestion of destination, which is equivalent to 11.76Gbits/s throughput of the network.

Crossbar is a well-known structure which can also supply the feature of non-blocking data transfers. Therefore, a six-node crossbar network is developed in this work as a reference to evaluate the CDMA network. In comparison with the six-node crossbar network, the CDMA network realization has 39.4% larger logic gate area cost when the data path width is 8 bits, whereas, the number of data wires in the CDMA network is 80.1% less than the number in the crossbar network if there are 31 network nodes.

Besides ASIC realizations, a four-node GALS bidirectional ring network is realized on an Field-Programmable Gate Array (FPGA) device as an example of prototyping a synchronous-asynchronous mixed NoC design on a Look-Up-Table (LUT) based FPGA device. The realization consumes 41.7K LUTs on an Altera StratixII FPGA device.

PREFACE

The work presented in this thesis has been carried out in the Department of Computer Systems at Tampere University of Technology (TUT) during the years 2003-2007.

I would like to deeply thank my supervisor Professor Jari Nurmi for his kind encouragement, patient guidance, and financial support throughout this research work. I would also like to express my deep gratitude to Dr. Tapani Ahonen and Dr. David Sigüenza-Tortosa for their numerously inspiring suggestions and warmly help during the past four years. I also want to express many thanks to my other colleagues, Mikko Alho, Sanna Määttä, Bin Hong, Yang Qu, Claudio Brunelli, Fabio Garzia, Markus Moisio, Srinivasan Sudharsan, Pauli Perälä, Raimo Mäkelä, Ethiopia Nigussie, for their kindness, friendliness, encouragement, and help during these years. I would also like to thank Professor Hannu Heusala from University of Oulu in Finland and Professor Axel Jantsch from Royal Institute of Technology in Sweden for reviewing this thesis and giving valuable comments to improve it.

I would also like to thank Juha Pirttimäki, Ari Nuuttila, and Timo Rintakoski for their patient and warm help to handle all kinds of computer and software problems or requests from me during these years. My sincere gratitude is also expressed to the institute secretaries and co-ordinators, Irmeli Lehto, Johanna Reponen, Ulla Siltaloppi, and Elina Orava, for their warm help about many administrative and document affairs during my stay in TUT. Of course, there are many other friends whose names are not listed at here are also very important to make my living in Tampere smoothly and happily, I would also like to express my thankfulness to them.

This research work was financially supported by the Department of Computer Systems of Tampere University of Technology, which is gratefully acknowledged.

Finally, I would like to express my sincere love and deep thankfulness to my wife – Xi Guo, my parents – YanMing Zhang and FuLu Wang, my brother – Qun Wang, and other relatives who constantly supported and encouraged me during my living in Finland. Without their love and support, I can not imagine how I could carry out this research work.

Tampere, April 2008

Xin Wang

TABLE OF CONTENTS

Abstract i

Preface iii

Table of Contents v

List of Publications ix

List of Figures xi

List of Tables xiii

List of Abbreviations xv

Part I: Argumentation 1

1. Introduction 3

1.1 Research Background 3

1.2 Objective and Scope of Research 4

1.3 Thesis Outline 5

2. Network-on-Chip Overview 7

2.1 NoC Design Issues 8

2.2 Examples of Existing NoC Designs 13

3. Applying GALS Scheme into On-Chip Networks 17

3.1 Multi-Clock Challenge and GALS Scheme 17

3.2 The Synchronization in GALS NoC 18

3.3 The Asynchronous Design for GALS NoC 20

3.3.1 Introduction of Asynchronous Design 21

3.3.2 The Asynchronous Designs Applied in the GALS NoCs 22

4. A Direct On-Chip Network 25
4.1 Network Topology 25
4.2 Network Structure 26
4.3 Network Node Design 26
4.4 Front-End Synthesis and Simulation Results 28
5. A CDMA On-Chip Network 31
5.1 Introduction of CDMA Technique 31
5.2 Motivation of Applying CDMA Technique in NoC 33
5.3 Applying CDMA Technique into On-Chip Networks 34
5.4 CDMA NoC Structure 36
5.4.1 Network Node 37
5.4.2 Network Arbiter 38
5.4.3 CDMA Transmitter 39
5.5 Front-End Synthesis and Simulation Results 40
5.6 SystemC Modeling and Performance Estimation 42
5.6.1 Modeling the CDMA NoC with SystemC 43
5.6.2 Performance Estimations 43
6. A Crossbar On-Chip Network 49
6.1 Introduction of Crossbar Structure 49
6.2 Network Structure and Network Node Design 50
6.3 Front-End Synthesis and Simulation Results 51
7. Comparisons 55
7.1 Data Transfer Principles 55
7.2 Network and Node Structures 56
7.3 Performance 57
7.3.1 Area Cost of Logic Gates 57
7.3.2 Number of Data Connection Wires 58
7.3.3 Area Cost of Interconnect Wires 60

7.3.4 Data Transfer Latency 62
7.3.5 Dynamic Power Consumption 64
7.4 Performance Comparisons with Other NoC Designs 65
8. Realizing a GALS NoC on an FPGA Device 69
8.1 Two Key Components for Realizing Asynchronous Designs on an FPGA Device 69
8.1.1 C-element Structure 70
8.1.2 Arbiter Structure 70
8.2 Realizing a Four-Node GALS Ring NoC 72
9. Conclusions 75
9.1 Summary of Publications 75
9.2 The Main Results 77
9.3 Future Research Directions 80
Bibliography 81

Part II: Publications 91

LIST OF PUBLICATIONS

This is a compilation style thesis which bases on the following nine publications. The publications are enclosed in Part II of this thesis and are referred as [P1], [P2] ..., [P9].

[P1] X. Wang, T.Ahonen, and J. Nurmi, “A Synthesizable RTL Design of Asynchronous FIFO”, in *Proceedings of the 2004 International Symposium on System-on-Chip*, (SOC 2004), pages 123-128, Tampere, Finland, November 2004.

[P2] X. Wang, D. Sigüenza-Tortosa, T. Ahonen, and J. Nurmi,“Asynchronous Network Node Design for Network-on-Chip”, in *Proceedings of the 2005 International Symposium on Signal, Circuits, and System*, (ISSCS 2005), Volume 1, pages 55-58, Iasi, Romania, July 2005.

[P3] X. Wang, and J. Nurmi,“An On-Chip CDMA Communication Network”, in *Proceedings of the 2005 International Symposium on System-on-Chip*, (SOC 2005), pages 155-160, Tampere, Finland, November 2005.

[P4] X. Wang, T. Ahonen, and J. Nurmi,“Prototyping A Globally Asynchronous Locally Synchronous Network-on-Chip On A Conventional FPGA Device Using Synchronous Design Tools”, in *Proceedings of the 2006 International Conference on Field Programmable Logic and Applications*, (FPL 2006), pages 657-662, Madrid, Spain, August 2006.

[P5] X. Wang, and J. Nurmi, “A RTL Asynchronous FIFO Design Using Modified Micropipeline”, in *Proceedings of the 10th Biennial Baltic Electronics Conference*, (BEC 2006), pages 95-98, Tallinn, Estonia, October 2006.

[P6] X. Wang, and J. Nurmi, “Comparison of a Ring On-Chip Network and a Code-Division Multiple-Access On-Chip Network”, in *VLSI Design, Special Issue on Networks-on-Chip*, Volume 2007, Article ID 18372, 14 pages, Hindawi Publishing Corporation, April 2007.

[P7] X. Wang, and J. Nurmi, “Comparing Two Non-Blocking Concurrent Data Switching Schemes for Network-on-Chip”, in *Proceedings of the 2007 International Conference on Computer as a tool*, (EUROCON 2007), pages 2587-2592, Warsaw, Poland, September 2007.

[P8] X. Wang, T. Ahonen, and J. Nurmi, “Applying CDMA Technique to Network-on-Chip”, in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Volume 15, Number 10, pages 1091-1100, October 2007.

[P9] X. Wang, and J. Nurmi, “Modeling A Code-Division Multiple-Access Network-on-Chip Using SystemC”, in *Proceedings of the 25th Norchip Conference*, (NORCHIP 2007), Aalborg, Denmark, November 2007.

LIST OF FIGURES

1 ISO Open Systems Interconnection Reference Model. 8

2 Network Topology Examples. 10

3 Packet-Buffer Flow Control Methods. 11

4 Flit-Buffer Flow Control Methods. 13

5 A Method of Applying GALS Scheme in a NoC Design. 18

6 Double-Latching Synchronization Scheme. 19

7 The Control Logic of Micropipeline. 22

8 Control Pipeline of Control-Centric Blocks. 24

9 Block Control Pipeline. 24

10 Ringlet Topology Example. 25

11 Six-Node Bidirectional Ring Network. 26

12 Network Node Structure of the Bidirectional Ring Network. 27

13 ATL Portions of the Bidirectional Ring Network. 30

14 The Principle of FHSS Technique. 32

15 The Principle of DSSS Technique. 33

16 The Principle of Digital CDMA NoC. 34

17 Digital CDMA Data Encoding Scheme. 35

18 Digital CDMA Data Decoding Scheme. 35

19 Six-Node CDMA On-Chip Network Structure. 37

20 The Block Diagram of the Network Node for CDMA NoC. 37

21 Bit-Synchronous Transfer Scheme. 39

22 ATL Portions of the CDMA NoC. 42

23 Channels and Interfaces in the CDMA Network Node. 44
24 Channels and Interfaces in the CDMA NoC. 44
25 Open-Loop Simulation Environment. 45
26 ATL Estimations with Different Channel Widths. 46
27 ATL Estimations with Different Number of Network Nodes. 46
28 ATL Estimations with Hot-Spot Traffic Pattern. 47
29 An Example of Crossbar Structure. 49
30 A Four-Node Crossbar Switch Structure. 50
31 Crossbar Network Structure. 51
32 Logic Gate Area Costs of Network Nodes. 58
33 Total Logic Gate Area Costs of the Three Networks. 58
34 Placement of the NoC Designs. 61
35 ATL Values of Tx 1-data-cell Packet. 63
36 ATL Values of Tx 2-data-cell Packet. 63
37 ATL Values of Tx 3-data-cell Packet. 63
38 Dynamic Power Consumption Comparison. 64
39 C-element Structures. 70
40 The Arbiter Structure for FPGA Realization. 71

LIST OF TABLES

1 Truth Table of C-Element. 23

2 Area Cost of the Bidirectional Ring Network. 28

3 STL Values of the Bidirectional Ring Network. 29

4 ATL Values of the Bidirectional Ring Network. 29

5 Area Cost of the Six-Node CDMA Network. 41

6 ATL Values of the CDMA NoC. 42

7 Area Cost of the Six-Node Crossbar Network. 52

8 ATL Values of the Crossbar Network. 52

9 Number of Data Connection Wires. 60

10 Equivalent Number of Intermediate Nodes in the Ring NoC. 64

11 Theoretical Throughput of the Six-Node CDMA Network. 65

12 ALUTs Utilization of ‘Network Node’ Blocks. 73

LIST OF ABBREVIATIONS

2-D	2-Dimensional
AHB	Advanced High-performance Bus
ALUT	Adaptive LUT
AMBA	Advanced Microcontroller Bus Architecture
APB	Advanced Peripheral Bus
ASB	Advanced System Bus
ASIC	Application Specific Integrated Circuits
ATL	Asynchronous Transfer Latency
A-T Protocol	Arbiter-based T Protocol
BE	Best-Effort
BVCI	Basic VCI
CDMA	Code-Division Multiple-Access
CRC	Cyclic Redundant Code
DCR	Device Control Register
D-FF	D-Flip-Flop
DI	Delay-Insensitive
DS-CDMA	Direct-Sequence CDMA
DSM	Deep Sub-Micron
DSSS	Direct-Sequence Spread-Spectrum
EMI	Electromagnetic Interference

FF	Flip-Flop
FHSS	Frequency-Hopping Spread-Spectrum
FIFO	First-In First-Out
FPGA	Field-Programmable Gate Array
GALS	Globally-Asynchronous Locally-Synchronous
GS	Guaranteed Services
HDL	Hardware Description Language
IC	Integrated Circuit
ITRS	International Technology Roadmap for Semiconductors
LAB	Logic Array Block
LUT	Look-Up-Table
MTBF	Mean Time Between Failure
NoC	Network-on-Chip
NRE	Non-Recurring Engineering
OCP	Open Core Protocol
OPB	On-chip Peripheral Bus
OSI	Open Systems Interconnection
PBL	Packet Bypass Latency
PCB	Printed Circuit Board
PCC	Packet Connected Circuit
PCI	Peripheral Component Interconnect
PLB	Processor Local Bus
PLL	Packet Loading Latency
PSL	Packet Storing Latency
PTL	Packet Transfer Latency
QDI	Quasi-Delay-Independent

QoS	Quality of Service
RTL	Register-Transfer Level
SI	Speed-Independent
SoC	System-on-Chip
SS	Spread Spectrum
STL	Synchronous Transfer Latency
TLM	Transaction-Level Modeling
T Protocol	Transmitter-based Protocol
T-R Protocol	Transmitter-Receiver-based Protocol
TUT	Tampere University of Technology
VCI	Virtual Component Interface
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit
VME	VersaModule Eurocard
XOR	eXclusive-OR

Part I: Argumentation

1. INTRODUCTION

Communications play a fundamental and crucial role for the development of human society in every aspect because better communications facilitate better understanding and cooperation between individuals, which in turn facilitate the achievements and development in society. As the society is continuously growing and developing, the need for cooperation and development expands to global level. Therefore, communication plays more and more important role of this globalization process, and the need for effective communications in all kinds of ways becomes higher and higher.

As a researcher in electronics field, the author believes that the same truth also applies to on-chip systems, which means that the quality of communication in an on-chip system prominently affects system performance. As the complexity of an on-chip system keeps growing, the communication among functional hosts in the system becomes a non-trivial issue to deal with. Therefore, with the interest in on-chip communication, the author started his research work on this topic in Tampere University of Technology (TUT) in October of 2003.

1.1 Research Background

Currently, silicon chips which contain thousands of million of transistors with 45nm feature size are already available on market, e.g. Intel Penryn processor. According to the report [42] from International Technology Roadmap for Semiconductors (ITRS) in 2007, a single semiconductor chip will contain multi-billion transistors with feature sizes around 22nm and clock frequencies around 35GHz by the year of 2016. This growing manufacture capacity and the highly demanding applications continuously drive the complexity of a System-on-Chip (SoC) to a higher degree in terms of number of system components and functionalities. For example, Cell Broadband Engine Architecture (CBEA) [33] jointly developed by IBM, Sony, and Toshiba, also referred as Cell processor, contains altogether 9 processing units in one chip. Furthermore, Tilera, a MIT spin-off company, released a 64-core processor called TILE64 [89] in 2007. As the number of system components becomes larger, current widely applied bus structures for data transfers in an on-chip system, e.g. CoreConnect [39], expose several disadvantages as addressed in [34]. Two main disadvantages are bus arbitration bottleneck and bandwidth limitation. The arbitration bottleneck means that the arbitration delay will grow if the number of bus hosts increases. The bandwidth limitation refers to the fact that the data transfer bandwidth of a bus structure is shared by all hosts attached to it in a time division manner. Hence, more hosts incur a lower share of bandwidth for each one.

Another challenge that an on-chip system faces is the heterogeneous characteristics of system components. The components in a SoC may include processors for computation tasks, functional blocks for accelerating certain tasks, and the modules for communicating with the peripherals of system. The different functions among different system components naturally cause them to work in different clock rates for optimal performance. Hence, coordination and communications among those components become challenging tasks. At the same time, the issues of wire delay, on-chip noise, process variance, and power consumption in the realm of Deep Sub-Micron (DSM) technologies also become challenging for chip design. Altogether, these challenges have brought more and more concerns on the on-chip communication issue of a SoC design.

In order to overcome the disadvantages of bus structures, the concept of Network-on-Chip (NoC) has been proposed as a solution at the beginning of 2000s, e.g. [6, 19, 34]. The idea of NoC is to separate the concerns of communication from computation by building on-chip communication structure with concepts adopted from computer networks. Each component of a SoC is viewed as a node of the on-chip communication network. System components communicate with each other through the on-chip network. For the challenges of multiple clock domains and DSM technology effect, Globally-Asynchronous Locally-Synchronous (GALS) scheme has been proposed as a solution. In 1984, the concept of GALS scheme was firstly introduced in [15] to handle metastability problem. In 1999, several chip designs [62,69] which apply GALS scheme were published. The idea of a GALS system is to partition a system into separate clock domains which run at different clock rates, and the separated domains communicate with each other in an asynchronous manner.

When the author joined the research group at TUT in 2003, a NoC architecture named Proteo [85] was under development in the group. At that time, a few NoC designs have been published, including Æthereal NoC [23], NOSTRUM NoC [54], and SPIN NoC [34]. There was no published paper dedicated to GALS NoC. Therefore, the author started this research work with realizing GALS scheme in a Proteo NoC instance.

1.2 Objective and Scope of Research

The goal of this work is to design a GALS on-chip network in the frame of Proteo NoC architecture while experimenting with different NoC structures. The work focuses on the following topics.

(1) Developing a network node structure and building a bidirectional-ring network to find a

way of realizing GALS scheme in Proteo NoC.

(2) Designing asynchronous circuits which include asynchronous FIFO design and asynchronous control logic for realizing the GALS NoC designs.

(3) Developing a GALS on-chip network which applies CDMA technique.

(4) Developing a modeling and performance estimation method for a GALS NoC design using SystemC.

(5) Developing a crossbar on-chip data switch structure for comparison purpose.

(6) Examining the characteristics of the developed CDMA NoC by comparing it with other NoC structures.

(7) Realizing a synchronous-asynchronous mixed GALS NoC design on a LUT-based FPGA device.

1.3 Thesis Outline

This thesis consists of two parts: Argumentation (Part I) and Publications (Part II). Part II includes reprints of nine international conference and journal publications on which this thesis bases. Part I starts with this chapter, Chapter 1, to introduce background and objective of this work. Chapter 2 gives an overview of on-chip networks including the background of NoC, the related design issues, and some examples of existing NoC designs. The topics addressed in Chapter 3 include the method of applying GALS scheme in on-chip networks and the related issues including synchronization and asynchronous designs for realizing a GALS NoC. Chapter 4 presents the work of designing and realizing a GALS ring NoC as a direct network instance of Proteo NoC architecture. Chapter 5 presents the work of designing and realizing an on-chip network which applies CDMA technique. Chapter 6 presents a crossbar network developed for comparison purpose. The comparisons between the CDMA network and other types of NoC designs including the ring and crossbar networks developed in this work are presented in Chapter 7. Chapter 8 presents the work of realizing a GALS NoC design on an FPGA device. Finally, the conclusions of this thesis including summary of the publications in Part II and the main results of the research work are presented in Chapter 9.

2. NETWORK-ON-CHIP OVERVIEW

The appearance of Integrated Circuit (IC) in 1959 was a milestone of the development of electronics industry. It created a productive way to manufacture large scale electronic circuits on a semiconductor device. As stated by Gordon Moore in 1965, “the complexity for minimum component costs has increased at a rate of roughly a factor of two per year” [64]. This statement is known as the original formulation of Moore’s law and often quoted as “the number of transistors that can be placed on an IC is increasing exponentially, doubling approximately every two years.” The Moore’s Law is still valid nowadays and believed to be valid until reaching the size of atoms.

Therefore, driven by the growing manufacture capacity and the growing requirement of applications, the complexity of an on-chip system is continuously growing in terms of number of transistors and functionalities. For example, Intel’s ‘Core 2 Duo’ processor fabricated with 65nm technology process contains 291 million transistors [40]. When the on-chip system becomes complicated, the system design methodology called orthogonalization of concerns [49] can be applied to deal with the complexity. As addressed in Chapter 1, the communication issue is very crucial for an on-chip system to perform its tasks efficiently. Therefore, in the context of SoC design, one way of applying the methodology of concerns orthogonalization is to separate the concerns of communication from computation to enable more efficient exploration of optimal solutions on each subject.

On-chip bus structure was firstly applied to handle on-chip communications for a SoC design in 1990s. The idea of on-chip bus is derived from the bus schemes, such as VersaModule Eurocard (VME) bus [78] and Peripheral Component Interconnect (PCI) bus [77], which are designed for connecting discrete devices on a Printed Circuit Board (PCB). The examples of on-chip bus structures include CoreConnect [39] and Advanced Microcontroller Bus Architecture (AMBA) [3]. CoreConnect is a complete and versatile bus specification which defines three types of buses: Processor Local Bus (PLB), On-chip Peripheral Bus (OPB) and Device Control Register Bus (DCR). AMBA, which is similar to CoreConnect, also specifies three kinds of buses: Advanced High-performance Bus (AHB), Advanced System Bus (ASB) and Advanced Peripheral Bus (APB). These bus structures supply many advanced features, such as split transactions and line transfers, for on-chip systems which contain a few processors.

However, as addressed in [34], bus structures have several disadvantages by the comparison of on-chip networks. The main disadvantages, bus arbitration bottleneck and bandwidth limitation as mentioned in section 1.1 of Chapter 1, are caused by the centralized and time-division manner of sharing a communication channel among all the hosts of a bus. The trend of future on-chip systems is that a large number of processing units will be integrated into one system, as the example shown by TILE64 [89]. Therefore, if a bus structure is applied in the future on-chip systems which contain a large number of components, it will suffer from the problems of arbitration delay, bandwidth limitation, and poor scalability. Hence, developing a dedicated on-chip network is the most promising solution for future on-chip communication. The issues of designing an on-chip network and the existing NoC designs will be introduced in the following two sections of this chapter.

2.1 NoC Design Issues

The concept of on-chip networks is derived from the well-established inter-computer networks. Therefore, taking a look at the design issues of designing computer networks is helpful for tackling design problems of NoC because they have a lot of similarities despite of different characteristics and application environments. The Open Systems Interconnection (OSI) reference model [41] is a layered description which has been used for building computer networks. Thus, the NoC design issues can be addressed according to the OSI reference model illustrated in Fig.1.

Seven layers are defined in the OSI model and illustrated in Fig.1. The seven layers include application layer, presentation layer, session layer, transport layer, network layer, data link layer, and physical layer. Each layer provides certain services to facilitate the communication

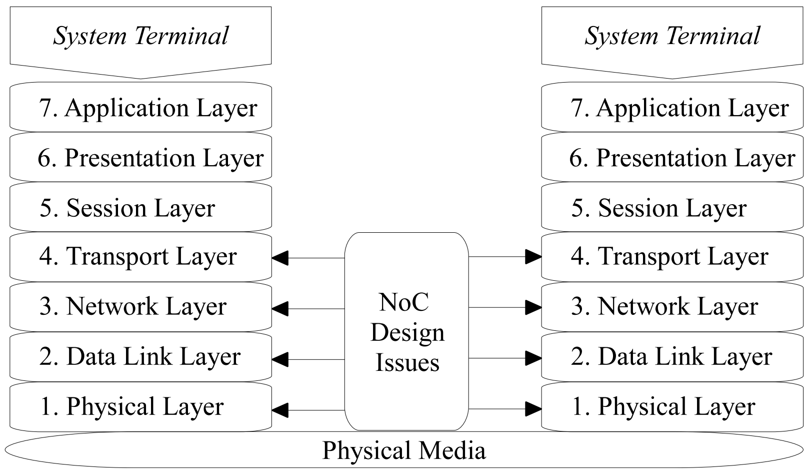


Fig. 1. *ISO Open Systems Interconnection Reference Model.*

processes in the network. The issues and challenges of designing on-chip networks will be addressed together with describing the functions of each layer in the following paragraphs of this section.

(1) Physical Layer. This layer defines all the electrical and physical specifications to activate, maintain, and de-activate physical connections for data transfers. Normally, a NoC design is implemented on a silicon chip in which the characteristics of the physical connection medium are determined by the manufacturing technology. As the manufacture technology scales down to DSM domain, the on-chip physical links face the challenges of large wire delay, large power consumption, crosstalk noise, etc. Therefore, the NoC design efforts in this layer mainly concentrate on conquering the above mentioned challenges in physical level. For example, the work presented in [27] gives a wire segmentation repeater structure to reduce wire delays. The work in [71] presents a booster structure to drive long wires instead of repeaters to achieve better performance in terms of area, power, and placement sensitivity. The work presented in [98] applies low-swing signaling techniques to reduce the power consumption of link wires. A physical link design for a NoC application is presented in [59]. The design applies mesochronous approach to realize a clock skew insensitive physical link.

(2) Data Link Layer. This layer is responsible for setting up reliable data transfers over physical links. The NoC design issues in this layer can include error detection and correction, access arbitration of physical media, and the methods of utilizing physical links. For instance, the Cyclic Redundant Code (CRC) scheme is applied in Xpipe NoC [7] to detect the possible transition errors. Another design issue related with this layer is the multi-clock-domain communication issue. In a large on-chip system, different functional hosts may work in different clock domains in order to achieve optimal performance; hence data transfer crossing clock domains is a design challenge. A data link design is presented in [58] to deal with this issue using mesochronous links. Another approach is to apply GALS scheme [15, 69] into on-chip networks. It means that the global links and the local links in a large on-chip system apply different communication methods to solve the multiple clock domain problem and increase data transfer reliability. This topic will be discussed further in Chapter 3.

(3) Network Layer. The network layer provides the means of data transfers through a network connection between a source and a destination. It should make the transport layer independent on the data routing and relay considerations. For a NoC design, the main issues to be handled in this layer include network topology and data routing.

Network topology concerns the layout and connectivity of the nodes and channels in a network. According to the functions of network nodes in a network topology, networks can be classified into direct and indirect networks. In a direct network, each node is both a terminal and a switch node. An example topology of direct networks is the 2-Dimensional (2-D) mesh topology illustrated in Fig.2(a). In a mesh topology, each node is used as a terminal node connecting with a functional host and as a router node switching data to their destinations.

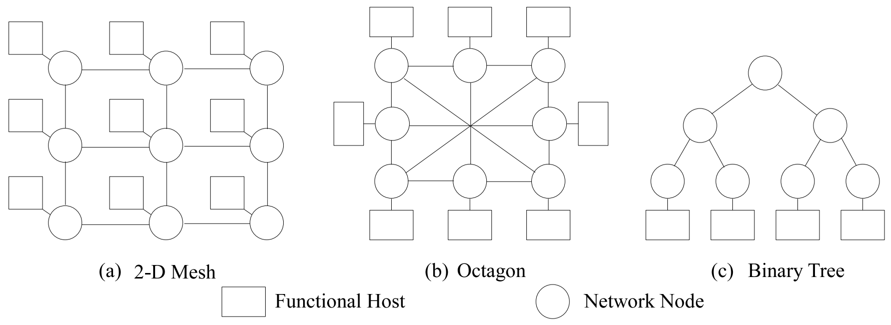


Fig. 2. *Network Topology Examples.*

Many NoC designs apply mesh topology since its simple structure and the ease of placement. Another topology of direct networks which has been applied in NoC designs is the octagon topology illustrated in Fig.2(b). It is an eight-node ring network with extra links between each pair of opposite nodes in the ring structure. In an indirect network, each node works either as a terminal or a switch. It cannot carry out both functions. An example of this category is the tree-based topology illustrated in Fig.2(c) with a case of binary tree. As illustrated in the figure, the nodes at the bottom level work as terminals to connect with functional hosts, while the nodes in higher levels function only as data switching nodes. Although other topologies exist for interconnection networks, only a few examples which have been applied in NoC designs were presented in this paragraph to give a glimpse on NoC topology choices. The NoC examples which apply these topologies will be presented in section 2.2.

Besides network topology, the routing method is another issue that needs to be considered in the network layer of a NoC design. After a network topology is set, a routing method is used to decide the path that data will be transferred from the source node to the destination node. According to different aspects, routing methods summarized in [72] can be classified in three ways as presented in the following three paragraphs.

Depending on where the routing decision is made, we can have source routing and distributed routing. By source routing method, the entire path of data transfers is determined by the source node before data transfers. By distributed routing, each router node decides the next node where the received data will be sent.

Depending on the information on which the routing decision bases, routing methods can be classified into deterministic routing and adaptive routing. Deterministic routing means that the data transfer path is determined only according to the source and destination addresses. Whereas, with adaptive routing method, the path is decided not only by the source and destination information, but also by the dynamic network conditions, such as traffic congestion information in the network.

Depending on the length of the decided path, minimal routing and non-minimal routing meth-

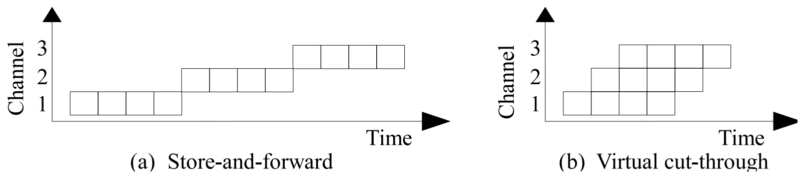


Fig. 3. *Packet-Buffer Flow Control Methods.*

ods can be differentiated. If a selected path is one of the shortest paths between the source and the destination, this method is called minimal routing. Otherwise, it is called non-minimal routing method.

A routing method applied in NoC designs can be a mixture of different routing categories. For example, the X-Y routing method is both deterministic and minimal. In X-Y routing, the data are transferred along the rows first, then are moved along the columns toward the destination in a 2-D mesh network. Because adaptive routing involves dynamic arbitration mechanisms which incur complex node implementation, deterministic routing is normally applied in NoC designs.

(4) Transport Layer. Transport layer protocols establish and maintain end-to-end connections between transport level entities. The concerned design issues in this layer include flow control and Quality of Service (QoS) management.

Flow control is the mechanism that determines the allocation of resources for data as they progress along their routes. According to the way of utilizing the channels between network nodes, two different approaches, circuit-switching and packet-switching [20], can be applied. In a circuit-switched network, a dedicated path from source to destination is set up before data transport and reserved until the transport is complete. In a packet-switched network, the data are transferred in form of packets. There are no channels set up for a data packet. All packets travel to their destinations by sharing the existing channels among nodes and following their paths determined by a routing method. The main disadvantage of circuit-switched networks is the lower efficiency of channel usage than the packet-switched network, which is caused by setting up dedicated paths for data transport. Therefore, packet-switching method is popular in NoC designs.

Normally, an on-chip network needs to include buffers to facilitate data transfers. According to the granularity at which buffers and channels are allocated and the way of forwarding data along their routes, flow control methods can be classified into packet-buffer flow control and flit-buffer flow control [20]. A flit is the minimum unit in a packet that can be recognized by a flow control method.

Two basic packet-buffer flow control methods are store-and-forward and virtual cut-through [20]. The principle of store-and-forward method is illustrated in Fig.3(a) with an example of transferring a four-flit packet. With store-and-forward method, a packet will not be forwarded to the next node along its path until all flits of the packet are received by the current intermediate node. Therefore, the disadvantage of this method is the high packet transfer latency caused by inefficient usage of channels. Hence, virtual cut-through method is proposed to solve this problem by immediately forwarding the received packet flit to the next node if the buffer and channel resources are available for the whole packet, without waiting for the entire packet to be received. Its principle is illustrated in Fig.3(b) without contentions. By transferring packets as soon as possible, virtual cut-through method reduces the serialization latency of store-and-forward method. However, there are two main shortcomings of virtual cut-through, or of any other packet-based flow control methods. One of them is the inefficient usage of buffers caused by allocating buffers in units of packets. This is very important when there are multiple buffer sets to reduce blocking or providing deadlock avoidance in a NoC. Another shortcoming is that the contention latency is increased by allocating channels in units of packets. The blocked packet needs to wait for the whole packet in transmission passing through the channel before it can acquire that channel. These shortcomings can be overcome by allocating resources in units of flits rather than packets.

A popular flit-buffer flow control method is wormhole method [20] which operates like virtual cut-through, but with resources allocated to flits rather than packets. It means that a flit only needs to acquire one flit buffer and one flit channel bandwidth before it can travel to the next node, which relieves the requirement of resources in comparison with virtual cut-through method. Whereas, with wormhole method, a packet in transfer occupies multiple channels when its flits are traversing along the channels one by one. This will cause a problem if the current packet is blocked during transfer. As illustrated in Fig.4(a), if the flit of packet A is blocked at the intermediate node 3 because of congestions, all the channels occupied by this packet will exclude packet B from using them. In this situation, virtual-channel method [20] is proposed to solve this blocking problem by associating multiple buffers to one physical channel. By using the buffers, multiple virtual channels can be set up on a single physical channel. As illustrated in Fig.4(b), with the virtual channels, the flits of packet B can be transferred to node 3 even when the flits of packet A are blocked at node 3.

Generally, the flit-buffer flow control methods are preferred in NoC designs because of its efficient usage of buffers and channel bandwidth. The application examples in NoC designs will be presented later in section 2.2.

Another issue concerned in the transport layer of a NoC design is QoS. It refers to the service qualification that is provided by the network to its users. In [43], the QoS in the context of NoC designs is classified into two basic classes, Best-Effort (BE) services and Guaranteed Services (GS). In BE services, the network makes no strong guarantee about the delay or loss, while the GS scheme can guarantee a certain level of performance as long as the injected traffic complies with a set of restrictions. Both types of QoS have been applied in NoC

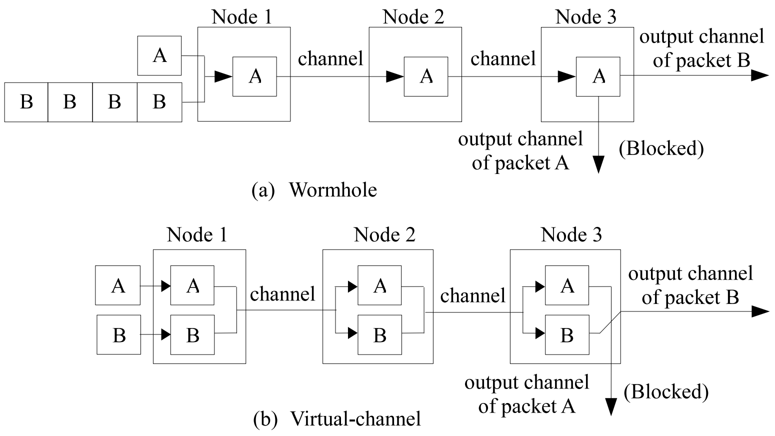


Fig. 4. *Flit-Buffer Flow Control Methods.*

designs. Because GS service demands more resource reservation and complex control logic than the BE service does, it is more expensive to support GS service in a NoC design.

(5) Session/Presentation/Application Layer. These three layers handle the communication processes of an interconnection network in high levels. Session layer mainly focuses on the connections between hosts. Presentation layer concerns the data representation and security issues. Application layer supplies services to user-defined application processes using interconnection networks. Generally, the services and functions of these three layers will be implemented by processors or software. Therefore, a NoC design normally does not need to directly handle the issues related to these layers.

The OSI model is only a reference for designing an interconnect network. Therefore, it only gives a guideline for designing an on-chip network rather than a regulation. From the above discussions about the OSI model and NoC design issues, we can see that the NoC design issues are generally within the three or four lowest layers in the OSI model and the boundaries between the design issues according to the layer definitions are not very strict. The presented design issues in this subsection do not mean a complete list of all possible design issues of on-chip networks, or rather, they are some typical NoC design issues addressed according to the OSI model.

2.2 Examples of Existing NoC Designs

A lot of research work about NoC structures has been carried out with different application requirements and backgrounds. There is no standard way to classify or summarize them. In this section, some examples are introduced to present the diversity of the existing NoC designs.

1. Different Topologies

As presented in section 2.1, 2-D mesh is the most widely applied topology since its simple structure and tidiness for placement. The SoCBUS presented in [93], HERMES NoC presented in [66], and the NoC design presented in [19] are the examples of 2-D mesh network. Based on 2-D mesh, another topology called 2-D torus can be formed by connecting each row and column of nodes in a 2-D mesh network into a ring. An interconnection network presented in [60] is the example of 2-D torus network. The torus network consists of four nodes and it is implemented on an FPGA device. Another type of topology quite different from the mesh and torus is an octagon topology illustrated in Fig.2(b), the NoC presented in [45] applies this topology. In the octagon NoC, the channels between every node are bidirectional links.

Besides the topologies of direct networks, indirect network topology is also applied in NoC designs. For example, SPIN [34] is a NoC design which applies a fat-tree topology consisted of two levels of routers, four routers in each level. Each router in the first level connects with four functional hosts. Each channel is comprised of two one-way 32-bit data paths. The fat-tree topology network is further explored by a NoC design called XGFT [46]. The XGFT NoC applies an extended generalized fat-tree topology to achieve better scalability and performance in comparison with a fat-tree network.

2. Different Data Switching Methods

The PROPHID architecture [55] is an early developed NoC which applies circuit-switching scheme. PROPHID uses a three-stage switch structure which consists of time-division switch and space-division switch to carry out data transfers in a multiprocessor system. Because the circuit-switching scheme has the disadvantage of non-scalability and insufficient parallelism for future on-chip systems, packet-switching scheme is most widely applied in current NoC designs, such as the mentioned HERMES network, SPIN network, and XGFT network.

However, there exists switching methods which combine the characteristics of both circuit switching and packet switching in NoC designs. For example, the SoCBUS applies a Packet Connected Circuit (PCC) [93] method which hybrids circuit switching with packet switching to transfer data in the network. It uses packet switching to set up the connection between network nodes and lock the setup as a circuit for data transmission. The Æthereal NoC [23] developed by Philips research laboratories applies a pipelined time-division multiplexed circuit switching scheme in a packet-switched network in order to acquire contention-free routing.

3. Different Routing Methods

For the sake of simplicity, deterministic routing methods are applied in the most of NoC designs. For example, the X-Y routing method introduced in section 2.1 has been applied in the 2-D mesh HERMES NoC, SoCIN NoC [97], and the network presented in [60]. In

SoCBUS, each node makes the routing decision based on the destination address and the static knowledge of the general direction to each destination. In Octagon NoC, a deterministic minimal routing method is realized by choosing the output direction at each node according to predefined rules.

In [22], three partially adaptive routing methods, west-first, north-last, and negative-first, are proposed for 2-D mesh networks. The common idea of those methods is that a deterministic route is followed when certain limits are obeyed, otherwise, the routing decision made by a node can be adaptive according to traffic conditions. For example, with the west-first routing, a node always tries to transfer packets firstly to the west direction of the source node whenever it is possible, otherwise, routing direction is adaptive to the traffic condition. The comparison between the three partially adaptive methods and X-Y method is also presented in [22]. The conclusion is that X-Y routing appears as the better choice in most situations in HERMES NoC.

4. Different Flow Control Methods

Wormhole and virtual-channel methods are two most frequently used flow control methods in NoC designs. Because the wormhole method requires less buffers and simpler control, it is easier to be applied in NoC designs. The widely accepted Æthereal NoC applies wormhole method in its best-effort router. HERMES, SoCIN, and SPIN also apply wormhole method. For virtual-channel method, the NoC design presented in [19] is an example. It applies 10K bits storage for virtual channels at each input controller. Virtual channels are also applied in a router design presented in [28] to support different QoS.

5. Different QoS Strategies

As addressed in section 2.1, GS and BE are two types of QoS applied in NoC designs. GS provides predictability of data transfers, while BE service has higher resource utilization. NOSTRUM [54] is an example of NoC design which provides GS. It uses looped containers implemented by virtual circuits to support GS in a mesh network. While in Æthereal NoC, both GS and BE services are provided by using a combined GS-BE router structure. The router includes two parts; one part applies pipelined circuit switching to implement its guaranteed service, while the other part applies input-queued wormhole flow control to provide best-effort service. Another type of combination of GS and BE services is presented in [28]. The design provides differentiated QoS between GS and BE by allowing higher priority data streams to overtake those of lower priority in virtual channels.

6. Different Implementation Strategies

As the design requirements for a NoC may vary largely depending on the applications, there is no a universal design which suits for all applications. Therefore, some NoC designs, e.g. SoCIN and Xpipes, realize the network components in a soft format which can be customized for a specific application. With the support of specialized design tools, e.g. XpipesCompiler, many design parameters, such as topology, network interfaces, and switch structures, can be customized to meet the requirements of a specific application during the design stage. Of course, the changeable design parameters in this type of NoC design can not be arbitrary. For instance, the topologies supported by SoCIN NoC only include mesh and torus. However, these choice limitations are reasonable since it is impossible to predict and meet all possible application requirements in one design.

7. Different Communication Synchronization Strategies

As addressed in section 2.1, GALS communication scheme is introduced in NoC to deal with the issue of multiple clock domain data transfer. Thus, asynchronous circuit design is applied in some NoC designs to implement GALS scheme. A NoC design called CHAIN [4] is such an example of an asynchronous NoC. It applies self-timed logic to build pipelines, multiplexing structures, and steering latches to transfer data with handshake protocols. Another GALS NoC example is MANGO presented in [8] which applies OCP compliant network adapter block to connect the functional blocks with its asynchronous communication network. It also provides both guaranteed and best-effort services by utilizing virtual channels [11]. Nexus NoC presented in [57] is an example of GALS NoC different from router-based NoC. It applies a 16-port asynchronous crossbar structure to build an on-chip data switch.

8. The Proteo NoC Architecture

Finally, the Proteo NoC architecture [85] developed in our department is introduced. Its name, Proteo, is taken from ancient Greek mythology to express the idea of flexibility of the proposed NoC architecture. Conceptually, as stated in [85], the Proteo NoC architecture consists of a library of hardware components, a set of CAD tools and a methodology of usage. The idea of Proteo NoC is to make a NoC instance be easily and quickly built for a specific application by applying specialized design tools and optimization methodology on a heterogeneous hardware IP library. A Proteo NoC instance is a packet-switched network in which topologies can be customized according to applications. Currently, deterministic routing and virtual cut-through flow control method are applied in a Proteo NoC instance. Proteo NoC supports Virtual Component Interface (VCI) [91] and Open Core Protocol (OCP) [75] interface standards for connecting each functional host to its corresponding network node.

3. APPLYING GALS SCHEME INTO ON-CHIP NETWORKS

As mentioned in Chapter 2, the number of processing or functional components in an on-chip system becomes larger and larger. Currently, a 64-core on-chip system, TILE64 [89], has already been produced. It is believed that a future on-chip system will consist of sea-of-processors in one chip [80]. Besides the growing number of system components, the functionalities of system components also become largely different from each other. It means that an on-chip system may include different processors for different computation tasks, varied hardware accelerators for varied functions, and various interface controllers for various peripheral devices. Therefore, these heterogeneous system components have different optimal working clock frequencies according to the tasks that they are handling. When integrating all the heterogeneous components into an on-chip system, coordinating different clock domains is a challenge.

3.1 Multi-Clock Challenge and GALS Scheme

From the viewpoint of a chip design, as addressed in [74], for large high-speed globally synchronous ASICs, designing the clock distribution net becomes a troublesome task because of the problems caused by clock skew, by the growing die sizes and shrinking clock periods. At the same time, the power consumption is increasing tremendously because the working clock frequency driven by demanding applications is getting higher in the scale of giga-hertz.

Therefore, one solution of the challenges mentioned above is to enable different processing or functional system components to work at their own clock rates. Thus, the following challenge that a SoC designer needs to handle is how to integrate the clock independent components into one system. In this situation, GALS scheme is proposed to solve the system integration challenge. The GALS scheme is firstly introduced in [15] to prevent metastability by stretching local clocks. The basic idea of applying GALS scheme into on-chip systems is to partition the system into several independently clocked domains that communicate with each other in an asynchronous fashion. The GALS scheme is the basis of the NoC structures designed and realized in this work. The method and challenges of designing a GALS on-chip network will be presented in the following two sections.

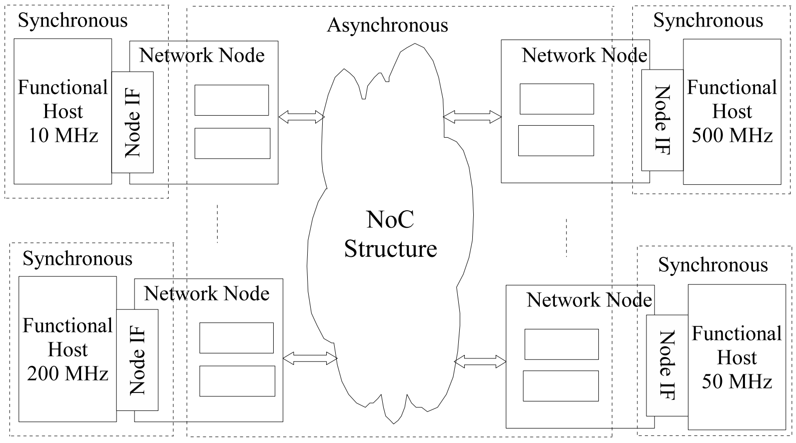


Fig. 5. *A Method of Applying GALS Scheme in a NoC Design.*

3.2 The Synchronization in GALS NoC

In an on-chip system, communication tasks among system components are performed by the on-chip network. Thus, the issue of realizing GALS scheme in an on-chip system equals to realize GALS scheme in the on-chip network. The method of applying GALS scheme in the NoC structures developed in this work are illustrated in Fig.5. From the figure, we can see that each network node contains an interface block which works at the same clock rate as the system functional block attached to it, while the blocks for global communication among network nodes apply asynchronous scheme.

Therefore, the data synchronization between synchronous and asynchronous domains is the main challenge of designing a GALS NoC. The term, synchronous domain, used in this thesis refers to the group of design blocks which work under the dictation of clock signals in a SoC, while, the term of asynchronous domain refers to the group of blocks which work in a self-timed manner without any clock signals.

Many synchronization schemes or structures for data transfers among independent clock domains in a GALS system have been presented. One category of solutions is to avoid synchronization failure by adjusting the clock signal of the local synchronous module or by generating a controllable clock signal in the synchronization interface. For example, the work presented in [35] develops a stoppable clock structure to build a deterministic wrapper. The work in [69, 99] presents stretchable clock schemes to avoid synchronization failure in the interface between synchronous and asynchronous domains. A pausible clock scheme is firstly presented in [96] to manage the data transfers between independent clock domains without synchronization failure. The work presented in [63, 68] further develops the pausible clock scheme. The work in [12] presents an asynchronous wrapper which combines the

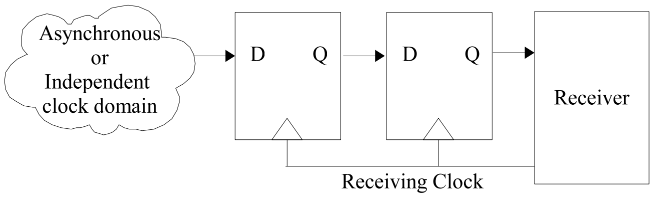


Fig. 6. *Double-Latching Synchronization Scheme.*

stretchable and pausible schemes together. This wrapper can avoid synchronization failures caused by metastability in circuits. One common feature of those presented synchronization schemes is that they all involve specialized clock generation or control circuits which need to be implemented in circuit level. Thus, if a GALS NoC design applies one of those synchronization schemes, the whole design can not be realized in Register-Transfer Level (RTL) by using Hardware Description Language (HDL), which in turn makes the NoC design less implementation flexible and portable. Therefore, this type of synchronization scheme is not applied in the NoC designs presented in this thesis.

Another type of solutions of data synchronization in a GALS system is to synchronize the signals from asynchronous domain with the local clock in an arbitrary timing relationship and limit synchronization failures within an acceptable level. The most widely applied scheme in this category is the double-latching scheme as illustrated in Fig.6. It consists of two serially connected D-Flip-Flop (D-FF) components to latch the input signals with the reference clock of the receiver. It is possible that the first D-FF enters into metastable state if input signal transitions violate the setup or hold timing requirement. In this situation, the second D-FF gives a whole clock cycle for the first D-FF to resolve the metastability before latching its output. However, in the double-latching scheme, there still exists the failure possibility if the first latch can not get rid of metastability state before the second flip-flop samples its output. Therefore, Mean Time Between Failure (MTBF) is introduced to measure the safety of a synchronizer. MTBF gives indication about how often a synchronization failure occurs. The performance analysis of double-latching synchronizers and the equation of calculating MTBF of a synchronizer are presented in [24, 53]. As addressed in [24] and presented in (1), the MTBF equation consists of the time (t) allowed for synchronization, the settling time (τ) of Flip-Flop (FF), the sampling clock frequency (f_s), the frequency (f_d) of data edges which generates a metastability, and a parameter (T_w) related to the metastability window of the FF.

$$MTBF = \frac{e^{t/\tau}}{T_w \cdot f_s \cdot f_d} \tag{1}$$

Besides the double-latching scheme, many other synchronization schemes or structures have also been proposed. For instance, a pipeline synchronization structure is proposed in [82] to achieve high communication bandwidth while keeping the failure possibility arbitrarily low.

Publication [26] presents a speculative synchronizer structure in transistor level to reduce synchronization latency. Another transistor level synchronizer design is presented in [44] to achieve high performance in a low voltage application. In [52], a parallel synchronizer scheme which bases on the double-latching scheme is introduced to reduce synchronization latency. All those presented synchronizer structures require the design to be implemented in gate or transistor level. In order to make the entire design of a GALS NoC suit the commonly used synchronous design flow, both the synchronous and asynchronous designs need to be modeled by using the commonly used HDL. Therefore, the double-latching scheme is selected to be used in the GALS NoC designs in this thesis.

In the GALS NoC designs presented in this thesis, double-latching scheme is used for synchronizing the handshake control signals for data transfers rather than the data signals themselves. For example, when transferring data from asynchronous domain to synchronous domain, the asynchronous logic will assert a request signal after the data to be transferred are ready. Then the asserted request signal will be synchronized with the receiving clock domain through a double-latching structure as illustrated in Fig.6. Whereas, the acknowledge signal that the synchronous domain sends back to the asynchronous domain can be received directly. When data are transferred from synchronous domain to asynchronous domain, the double-latching scheme is only needed for the synchronous logic to receive an acknowledge signal from the asynchronous domain during a four-phase handshake process. The safety of applying double-latching scheme has been analyzed in [31], where it is stated that the MTBF of most SoC designs is safe far more than enough by simply setting the resolving time window to one clock cycle. Among the published NoC designs, MANGO NoC is an example which applies the double-latching scheme to synchronize the synchronous and asynchronous domains. In [9], the designer of MANGO NoC claims that the estimated MTBF of the implemented double-latching synchronizer is longer than 8000 years. Therefore, the simple and safe enough double-latching scheme is a reasonable choice for a GALS NoC design.

3.3 The Asynchronous Design for GALS NoC

In order to realize a GALS NoC design, both synchronous and asynchronous designs are needed. Synchronous design methodology and techniques have been well established and applied. Many standard design tools and design flows are developed for synchronous designs. Whereas, asynchronous design has not been widely applied after it was born in 1950s. The asynchronous designs of the GALS NoCs developed in this work will be presented in this section.

3.3.1 Introduction of Asynchronous Design

As stated in [70], asynchronous design methods can date back to 1950s and to two people in particular: D.A. Huffman and D.E. Muller. Huffman developed an asynchronous design methodology known as fundamental-mode circuits [38] in which the delay in all circuit elements and wires is assumed to be known, or at least bounded. The methodology developed by Muller is Speed-Independent (SI) circuits [67] in which gate delays are assumed to be unbounded while the wire delays are negligible.

Almost all the other types of asynchronous design methods can find their roots in those two fundamental methodologies. For example, Delay-Insensitive (DI) circuit model extends the assumption of SI circuits by assuming that both gate and wire delays in circuits are unbounded. The burst-mode design methodology [73] assumes that only the specified input bursts which can make circuits leave the current state can occur in a given circuit state, and the fundamental-mode assumption is applied between transitions among different input bursts. Ivan Sutherland developed a micropipeline structure [87] as an asynchronous alternative of synchronous elastic pipelines. A micropipeline structure consists of a bounded-delay data path controlled by delay-insensitive control logic.

After the birth of asynchronous design in 1950s, it has not been as widely adopted as synchronous designs except several academic projects during the first several decades, such as the ILLIAC II computer [13] developed at University of Illinois in 1960s, the first operational data-flow computer [21] developed at the University of Utah in 1970s, and the first fully asynchronous microprocessor [61] developed at California Institute of Technology in 1980s. As the development of IC design in recent decades, synchronous designs face the hard challenges of clock distribution, power consumption, and design complexity. Therefore, as an alternative to synchronous design, asynchronous design gains more applications than before. For instance, Philips developed asynchronous pager chips [79] in 1998 and a contactless smart-card chip [48] in 2000. A series of asynchronous microprocessors called Amulet [29, 30, 94] have been developed in University of Manchester from 1994 to 2000. In 2005, products based on an asynchronous NoC design were released by a company called Silistix [37]. One common motivation of those asynchronous design applications is to utilize the advantages of asynchronous design. Several main advantages of asynchronous design are briefly introduced in the following five paragraphs as the end of this short introduction of asynchronous design.

(1) Low power consumption. Because asynchronous circuits do not need any clock signals, the power spent on clock switching in a synchronous chip is avoided. Additionally, the signal transitions in asynchronous circuits will automatically stop when there is no driven event. Therefore, asynchronous designs can achieve lower power consumption.

(2) No clock distribution and clock skew. This advantage is obvious since the lack of clock

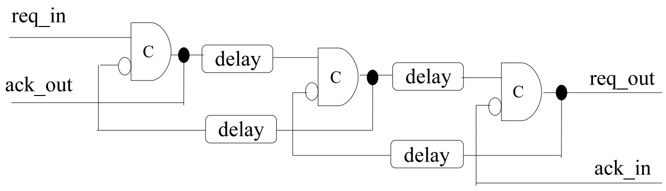


Fig. 7. *The Control Logic of Micropipeline.*

signal in asynchronous circuits. Thus, the difficulties of clock distribution and clock skew faced by synchronous designs are removed from asynchronous designs.

(3) Average-case performance. In a synchronous design, the operating speed is limited by the worst-case, called critical path, in the circuits. However, in asynchronous circuits, the operating speed is determined by actual local latencies in the circuits rather than the global worst-case latency. In most of cases, the average-case of latencies are smaller than the worst-case latency, hence, asynchronous designs can achieve better operating speed performance.

(4) Less Electromagnetic Interference (EMI) radiation. In a synchronous design, flip-flop transitions follow a certain clock frequency so that the energy spent on signal transitions concentrates within the very narrow bands around the clock frequency. Thus, the synchronized signal switching activities will produce substantial electrical noise. Whereas, the switching activities in an asynchronous circuit are correlated loosely because there is no universal timing pace, hence, they produce a more distributed noise spectrum and a lower peak noise value.

(5) Robust and adaptive. A synchronous circuit is sensitive to the delay variations caused by the variations of clock signal, supply voltage, and operating temperature related with the manufacture process and application surrounding. Whereas, because the loose timing requirement, asynchronous circuits can operate correctly under large variations caused by different manufacture processes and application environment.

3.3.2 The Asynchronous Designs Applied in the GALS NoCs

Although some asynchronous design tools and methods have been proposed, such as Balsa [81] and Tangram [90], there is no widely adopted or standard one. The asynchronous designs applied in the GALS NoC structures in this thesis base on the delay-insensitive control logic of micropipeline. The structure of micropipeline control logic is illustrated in Fig.7.

The logic components marked with ‘C’ in the figure represent the basic component of asynchronous circuits called C-element. The truth table of a C-element is listed in Table 1. From Fig.7, we can see that the principle of micropipeline control logic is to use the output from the next stage to enable or disable the output of the current stage. The components marked

Table 1. *Truth Table of C-Element.*

Input1	***Input2***	***Output***
0	0	0
0	1	No Change
1	0	No Change
1	1	1

with ‘delay’ in Fig.7 illustrate the logic and wire delays along the paths. The asynchronous design applied in this work bases on the micropipeline control logic and will be presented in the following paragraphs.

As illustrated in Fig.5, the GALS NoC designs in this thesis apply asynchronous design in a part of the network node and the interconnection structures between network nodes. The asynchronous design in the GALS NoCs can be divided into two parts which include data path and control logic. The data path is composed of the data registers which store or deliver the data items through a four-phase dual-rail handshake protocol under the control of the micropipeline-based control logic. Hence, the main design task is to design the control logic.

Two pipeline structures were developed as the control logic of the asynchronous design. One type of control pipeline is used for control-centric blocks, such as the control block in a network node which coordinates packet receiving and sending tasks. Another type of control pipeline, called block control pipeline, is used in data-path centric blocks, such as packet receiver and packet sender blocks in a network node. A hybrid control pipeline structure which combines the two pipelines mentioned above is applied as the control logic of the asynchronous FIFOs used in the packet buffer blocks of the GALS NoCs. Therefore, the following two paragraphs will present the two main control pipeline structures by analyzing their basic portions.

Two stages of the control pipeline used in control-centric blocks are illustrated in Fig.8. In the figure, we can see that the control pipeline uses micropipeline control logic as the backbone and applies a few AND gates as the delay components, hence, it is still delay-insensitive. The state information of the pipeline is passed through each stage in the pipeline by a four-phase handshake protocol. If we take the ‘Stage 1’ illustrated in Fig.8 as an example, when both the ‘req_from_stage0’ and ‘stag1_enable’ signals are ‘1’, the output of ‘C1’ will be set to logic ‘1’ which indicates that the current active state of the pipeline is in the ‘Stage 1’. Then the output of ‘C1’ can be used as a request signal to trigger the control logic in the corresponding function blocks for a certain communication process.

The structure of block control pipeline is illustrated in Fig.9. The main task of this type of control logic is to generate four-phase request or acknowledge signals for data transfers. Each stage of the control pipeline is composed of two C-elements as illustrated in Fig.9. The ‘C1’ is used to record the rising edge of a request or acknowledge signal, while the ‘C2’ is used

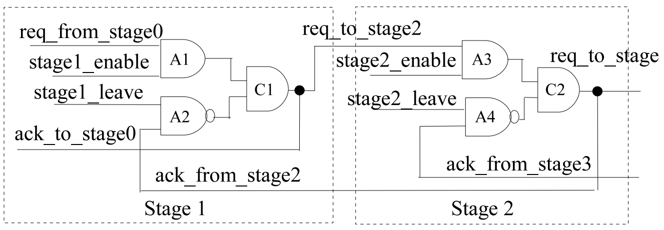


Fig. 8. *Control Pipeline of Control-Centric Blocks.*

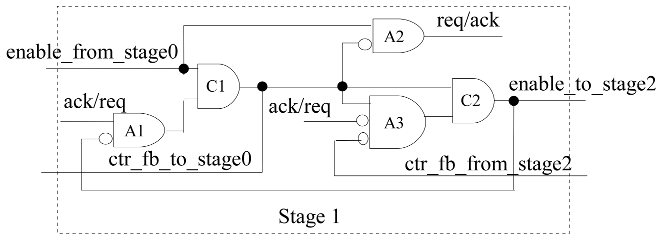


Fig. 9. *Block Control Pipeline.*

to record the falling edge of a request or acknowledge signal. Therefore, each stage of the block control pipeline will pass the enable signal to the next stage only after the four-phase handshake process on the current stage has been completed. Although the presented block control pipeline structure can only meet Quasi-Delay-Independent (QDI) model because the input ‘ack/req’ signal is branched to ‘A1’ and ‘A3’, the timing requirement for distributing the ‘ack/req’ input signal along the isochronic wire forks is quite loose since the logic delays in ‘A1’ and ‘C1’ are usually much larger than the logic delay of the inverter at the input of ‘A3’.

By using the presented control pipelines, the asynchronous design for the GALS NoCs proposed in this thesis are realized in RTL by using VHSIC Hardware Description Language (VHDL) together with the synchronous design. Thus, the GALS NoC designs are compatible with the commonly used design tools and flow for synchronous circuits. This feature facilitates the portability and flexibility of the NoC designs.

4. A DIRECT ON-CHIP NETWORK

As mentioned in section 1.1, the work presented in this thesis started from implementing GALS scheme in Proteo NoC architecture. This chapter presents the work of designing a GALS direct on-chip network as an instance of Proteo NoC architecture.

4.1 Network Topology

As stated in [85], Proteo NoC architecture consists of an interconnection IP library and software tools which automate the process of designing and configuring a network instance. Therefore, the topology of Proteo NoC architecture is open for exploration during the design stage. An optimal topology can be generated and utilized in a network instance of Proteo NoC for a specific application.

The author of Proteo NoC suggests applying ring-like topologies in an instance of Proteo NoC. As addressed in [85], this kind of topologies can simplify the analysis and make routing algorithms straightforward, while attaining a reasonable flexibility. After the further development of the automation tools and interconnect IP library of Proteo NoC architecture, the topologies presented in Chapter 2, such as 2-D mesh, torus, and octagon, can also be included.

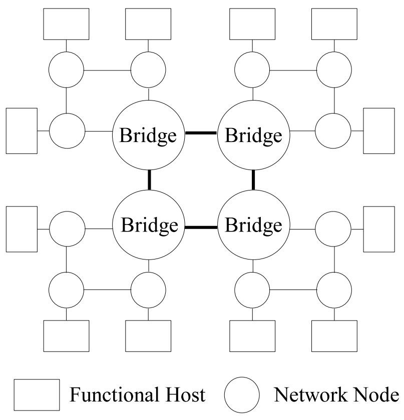


Fig. 10. *Ringlet Topology Example.*

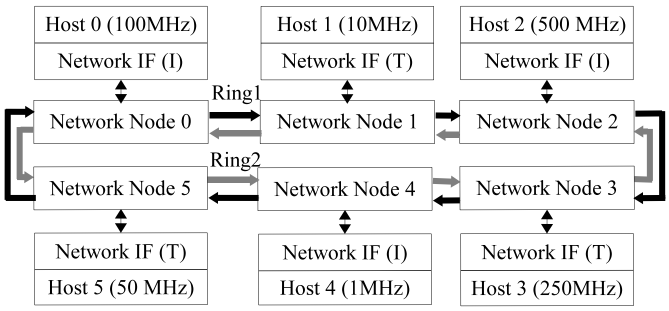


Fig. 11. *Six-Node Bidirectional Ring Network.*

Currently, studying on ring-like topologies can already give a clue of the characteristics of Proteo NoC when a direct network structure is applied. Therefore, the GALS NoC design presented in this chapter applies a bidirectional ring topology. More complex topologies can be built by using the ring structure and bridge blocks as described in [85] and illustrated in Fig.10.

4.2 Network Structure

The bidirectional ring NoC developed in this work is illustrated in Fig.11. It is a six-node network in which each functional host block is connected into the network through individual network node block. The interface between a functional host and a network node can apply one of the two widely accepted interface standards, VCI [91] or OCP [75], to adopt heterogeneous functional IP blocks in the design library. In this work, Basic VCI (BVCI) is applied because it is a good compromise between advanced features and the simplicity of realization. In the bidirectional ring network, each network node has two output and two input connections with its two neighbour nodes. From Fig.11, we can see that each functional host works at its own clock rate which may be very different from the clock rates of the other functional hosts. This setup highlights the necessity of applying GALS scheme in an on-chip network since all functional hosts work at different clock rates. In the context of the illustrated network, applying GALS scheme means that each local functional host can work with its own frequency while the communications between network nodes are performed in an asynchronous manner.

4.3 Network Node Design

A ring network belongs to direct network category in which each network node is both an interface block of a functional host in the system and a router node of switching data in the

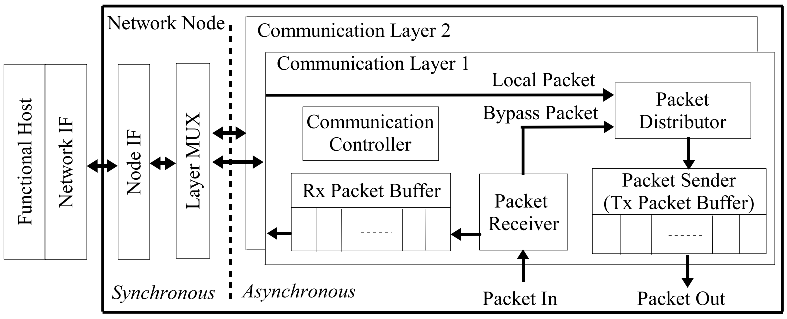


Fig. 12. *Network Node Structure of the Bidirectional Ring Network.*

network. A network node structure developed for the bidirectional ring network is illustrated in Fig.12. It contains two 'Communication Layer' blocks to handle the two output and two input links with its neighbour nodes. Although it is designed for the bidirectional ring network, it can be easily adapted for other types of topologies by adding more 'Communication Layer' blocks. For example, by using four 'Communication Layer' blocks, the illustrated network node structure can be used to set up a mesh network. The functions of each block in the network node are described in the following paragraphs of this section.

1. 'Node IF' Block

This block is the interface block which applies BVCI interface standard to communicate with the functional host via the 'Network IF' block of the functional host. The 'Node IF' block acts as an counterpart of the functional host. It means that if the host is the initiator type according to the BVCI standard, the 'Node IF' should be the target type and vice versa. The main tasks of the 'Node IF' include receiving data from the functional host and assembling the data into packet format. And then, it will deliver the packets to 'Communication Layer' blocks via 'Layer MUX' block. In the process of receiving data, this block takes care of extracting data from the received data packets and delivers the data to the functional host according to the BVCI interface standard.

2. 'Layer MUX' Block

This block is a multiplexer which sets up data channels between 'Node IF' and the two 'Communication Layer' blocks. 'Layer MUX' can set up one data sending channel and one data receiving channel for the 'Node IF' block simultaneously. It means that the two 'Communication Layer' blocks can communicate with 'Node IF' at the same time only if one is used for data sending while the other one is used for data receiving. Otherwise, only one 'Communication Layer' block can communicate with 'Node IF' through 'Layer MUX'. If one 'Communication Layer' block is communicating with 'Node IF' both for data sending and receiving simultaneously, then the other one can not be connected to 'Node IF' at this moment.

3. ‘Communication Layer’ Block

The function of this block is to perform the asynchronous communications with other network nodes through a handshake protocol. As illustrated in Fig.12, two ‘Communication Layer’ blocks are used for the bidirectional ring topology. However, more ‘Communication Layer’ blocks can be included to support other types of topologies.

There are five sub-blocks in a ‘Communication Layer’ block to carry out the data packet sending or receiving tasks. The ‘Packet Receiver’ sub-block is used to receive data packets from the other network node connected to the current node. If the destination of the received packet is the current node, the packet is called ‘incoming packet’, and it will be stored in ‘Rx Packet Buffer’. Otherwise, the received packet is called ‘bypass packet’, and it will be dispatched into ‘Packet Sender’ block via ‘Packet Distributor’ for further transfers. The data packets which come from the functional host via ‘Node IF’ will also be sent to ‘Packet Sender’ block for asynchronous transfer. The ‘Communication Controller’ sub-block is the controller which takes care of the necessary arbitrations and control tasks of data sending and receiving processes.

4.4 Front-End Synthesis and Simulation Results

The presented network node design and the six-node bidirectional ring network illustrated in Fig.11 are modeled in RTL by using VHDL and synthesized with a 0.18μm standard-cell library. The ‘functional host’ blocks and their ‘Network IF’ blocks are not realized with any real IP blocks, instead, they are simulated by adding stimulus signals on each ‘Network Node’ block according to the BVCI standard. Two important aspects of this front-end realization results, logic gate area cost and data transfer latency, are presented in this section.

Table 2. *Area Cost of the Bidirectional Ring Network.*

Block Name		***Area (K equivalent gates)***
Node IF	**Target**	1.145
	Initiator	2.835
Layer MUX		0.374
Communication Controller		0.665
Packet Distributor		0.577
Packet Sender (include Tx Packet Buffer)		3.893
Packet Receiver		0.643
Rx Packet Buffer		3.422
Total area of the six-node bidirectional ring network		***177.007***

Table 3. *STL Values of the Bidirectional Ring Network.*

Node Type	*Latency of sending data to 'Network Node'*	*Latency of receiving data from 'Network Node'*
BVCI Initiator	8 local clock cycles + 2.5 ns	8 local clock cycles + 3.2 ns
BVCI Target	4 local clock cycles + 2.5 ns	4 local clock cycles + 3.1 ns

Table 4. *ATL Values of the Bidirectional Ring Network.*

Packet Length	*PLL (ns)*	*PTL (ns)*	*PBL (ns)*	*PSL (ns)*
1 data cell	11.7	13.4	10.7	3.3
2 data cells	15.2	18.7	14.2	3.3
3 data cells	18.6	24.0	17.6	3.3

1. Logic Gate Area Cost

The logic gate area cost of the network node and the six-node network are listed in Table 2 with the number of equivalent gates. From the figures in Table 2, we can see that the Tx/Rx buffer takes the largest portion in the area cost. Although this area report is only taking logic gates into account, these front-end synthesis results are indicative enough for estimating the complexity of the design and carrying out comparisons to alternative implementations. An estimation of the area cost of wires will be presented in section 7.3.3.

2. Data Transfer Latency

Data transfer latency is an important performance indicator of the bidirectional ring network. The latency values of transferring data packets in the six-node network are measured during the gate-level simulations. Because the GALS scheme is applied in the ring network, the data transfer latency of the network can be divided into two parts, Synchronous Transfer Latency (STL) and Asynchronous Transfer Latency (ATL).

The STL refers to the data transfer latency between a functional host and the network node attached to it. The value of STL depends on the local clock and the type of interface. The measured STL values during gate-level simulations are listed in Table 3. The constant values in Table 3 are caused by the handshakes in the asynchronous domain. They are independent of the local clock rate but belong to the synchronous transfer processes. Therefore they are counted as a part of STL.

The ATL refers to the data transfer latency of transferring data packets from one network node to the neighbour node by using a four-phase handshake protocol. The ATL value of the bidirectional ring network consists of four portions: Packet Loading Latency (PLL), Packet Transfer Latency (PTL), Packet Bypass Latency (PBL), and Packet Storing Latency (PSL). The concept of these four latency portions is illustrated in Fig.13 with an example that 'Network Node 0' sends one packet to 'Network Node 2' via 'Network Node 1'. The black arrows

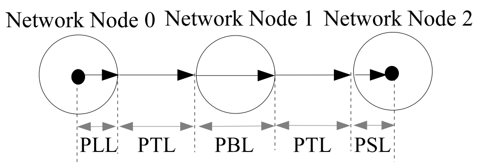


Fig. 13. *ATL Portions of the Bidirectional Ring Network.*

in Fig.13 represent the packet transfer direction. The different portions of the ATL are marked by grey arrows in Fig.13 and explained in the following paragraphs.

(1) Packet Load Latency (PLL). It is the time used to load one packet from a functional host into 'Tx Packet Buffer' block.

(2) Packet Transfer Latency (PTL). This latency refers to the time used to transfer one data packet from the 'Packet Sender' of a network node to the 'Packet Receiver' of its adjacent node using a four-phase handshake protocol.

(3) Packet Bypass Latency (PBL). After a network node receives a packet from another node, it will check the destination address of the received packet. If it is a 'bypass packet', it will be delivered into 'Tx Packet Buffer'. The time spent on this process is called PBL.

(4) Packet Storing Latency (PSL). It is the time spent on storing one 'incoming packet' into 'Rx Packet Buffer'.

The formula of calculating the ATL of transferring one packet in the ring network is given in (2). It represents the situation in which the packet traverses several network nodes before reaching its destination. N refers to the number of intermediate nodes between the source node and destination node of a packet. If a packet is transferred between two adjacent network nodes, then N is 0. The measured values of each ATL portion are listed in Table 4. The values are measured in a non-congestion situation which means that no conflicts between 'bypass packet' transfer and the 'local packet' transfer are included in the simulation. The listed latency values only include the logic gate delay of the circuits, no wire delay is considered. More accurate latency values could be obtained by including the wire delay after layout.

$$ATL = PLL + PTL \times (N+1) + PBL \times N + PSL \tag{2}$$

By using the listed latency values and (2), we can estimate the latency range of the six-node ring network in a non-congestion situation. From Fig.11, we can see that the worst case would be a 3-data-cell packet transfer from functional host 4 (1 MHz) to host 1 (10 MHz), and the estimation value is 8534.7*ns*. The best case would be a 1-data-cell packet transfer from the host 2 (500 MHz) to host 3 (250 MHz), and the estimation value is 66*ns*.

5. A CDMA ON-CHIP NETWORK

In 1948, Claude Shannon, a Bell Labs research mathematician, published his landmark paper titled ‘A Mathematical Theory of Communication’ [83] on information theory. Claude Shannon observed that “the fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point” [83], and gave the Shannon’s communication capacity equation as presented in (3). In (3), the parameter B_ω refers to the bandwidth of the communication channel. S is the signal power, and N is the noise power.

$$C = B_\omega \cdot log_2[1 + \frac{S}{N}] \tag{3}$$

The equation reveals that the communication channel capacity relates with the channel bandwidth and signal-to-noise ratio. Therefore, the communication channel capacity can be increased by two means, increasing the signal-to-noise ratio, and/or spreading the channel bandwidth. The work presented in this chapter concentrates on the application of channel bandwidth spreading method in a NoC design.

5.1 Introduction of CDMA Technique

The Spread Spectrum (SS) technique is based on the idea of increasing channel capacity by spreading the channel bandwidth. CDMA technique [92] is one of SS techniques that originate from military secure communication. There are two types of mainly used CDMA techniques, one is called Frequency-Hopping Spread-Spectrum (FHSS), and another is called Direct-Sequence Spread-Spectrum (DSSS). The principle of FHSS technique is illustrated in Fig.14. In a FHSS system, each user’s narrow-band signal hops among discrete frequencies with a certain sequence, and the receiver follows the hopping frequencies with the sequence. It is currently used mainly in military communication.

The principle of the DSSS technique is illustrated in Fig.15 with a two-user system. It applies a set of orthogonal codes to spread the narrow-band signals from different users to wide-band signals, and then transmits the wide-band signals in the same communication media with the same frequency concurrently. At the receiving end, the original signals can be recovered by utilizing the auto-correlation characteristic of orthogonal codes. Therefore, by introducing orthogonal code domain, different signals can share the time and frequency simultaneously in the transmission media. The illustrated system is also called Direct-Sequence CDMA (DS-CDMA) and it is widely used in civil communication systems. The main advantages of CDMA technique are listed in the following six paragraphs:

(1)Data transmission overlapping both in time and frequency domains. This feature increases usage efficiency of communication media.

(2)Anti-jam capability. This is because the user-unique orthogonal code for data encoding is known only by the intended receiver.

(3)Anti-interference capability. This feature is obtained by the fact that the signal dispreading can reject strong undesired noise, even stronger than the desired one.

(4)Low probability of intercept. Because the transmitted signal is masked in the background noise by spreading the signal uniformly in the frequency domain, it is very difficult to detect the signal among the noise in the communication channel.

(5)Anti-multipath capability. By using path-delay differences, a CDMA receiver can use the multipath signals for receiving data.

(6)Multiple access capability. This feature is supported by applying the orthogonal code of the sender in multiple receivers.

These advantages apply for the traditional use of DS-CDMA technique, for the purpose of on-chip data transfers, advantages (1) and (6) are of interest.

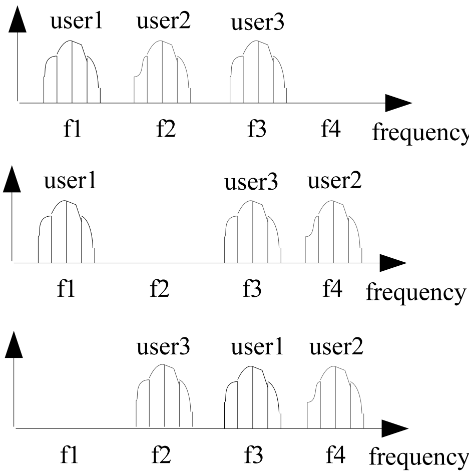


Fig. 14. *The Principle of FHSS Technique.*

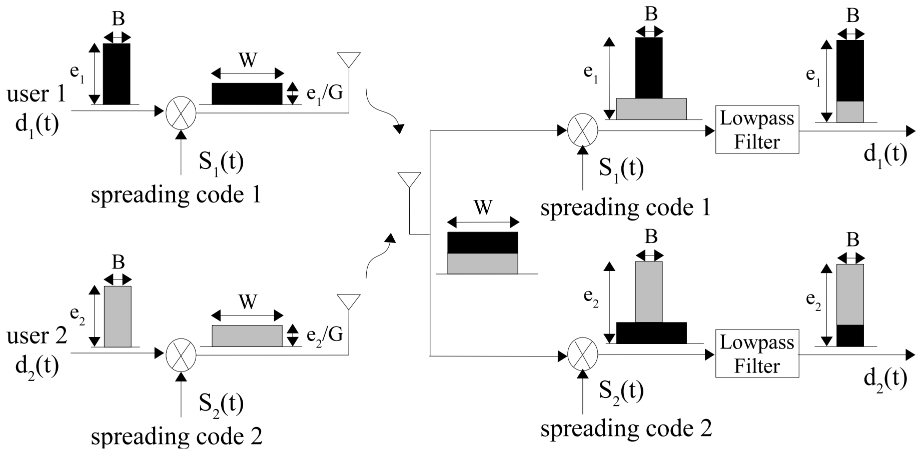


Fig. 15. *The Principle of DSSS Technique.*

5.2 *Motivation of Applying CDMA Technique in NoC*

In a direct on-chip network, the packet transfer latency caused by routing data packets via multiple intermediate network nodes is unavoidable except that the destination of a packet is a neighbour node. Therefore, the packet transfer latency in a direct connection NoC may vary largely when data packets are transferred to different destinations or to the same destination via different routes. The variant data transfer latencies in a direct network have negative effect for the on-chip network to provide a stable and guaranteed data transfer service.

One way of solving this problem is to optimize the routing technique, such as X-Y routing [18] and planar-adaptive routing [16], etc., so that the transfer latency caused by packet routing is minimized or reduced. The research work presented in this thesis examines another way of solving the transfer latency variation problem by applying the CDMA technique into on-chip communications. As introduced in section 5.1, the CDMA technique can transfer data concurrently both in time and frequency domains by separating different data streams in the orthogonal code domain. Therefore, if the data packets from different network nodes are encoded with a set of orthogonal codes, they can be transferred in a shared communication media concurrently without interfering each other. Namely, data packets in a CDMA network can be transferred to their different destinations with 'one-hop' transfer. Hence, applying CDMA technique can be a promising way of realizing constant data transfer latency in an on-chip network.

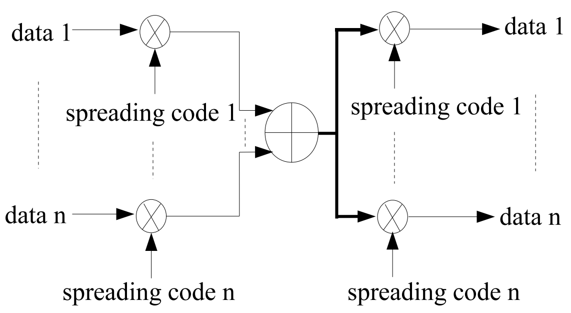


Fig. 16. *The Principle of Digital CDMA NoC.*

5.3 Applying CDMA Technique into On-Chip Networks

The CDMA technique applied in this work is the DS-CDMA technique introduced in section 5.1. The principle of DS-CDMA is illustrated in Fig.15 and the name of DS-CDMA technique will be referred as CDMA technique in the rest of this thesis for the sake of simplicity. The idea of applying the CDMA technique into on-chip communication has already been presented in several works [47, 84, 88, 95] before this work. They all apply analog circuits to implement the CDMA technique, for example, the encoded data are represented by the continuous voltage or capacitance value of the circuits. Thus, the data transfers through the analog circuits are challenged by the coupling noise, clock skew, and the variations of capacitance and resistance caused by manufacture processes [88]. Therefore, a way of applying the CDMA technique in an on-chip network by using digital circuit design is developed in this work. The basic principle of the digital CDMA on-chip network is illustrated in Fig.16. It follows the principle of CDMA systems illustrated in Fig.15. The difference is that the radio transmission part is replaced by adding the encoded signals together for transferring through on-chip wires. Several important design issues of realizing the CDMA technique in an on-chip network will be addressed in the following.

1. Digital CDMA Encoding/Decoding Schemes

In order to avoid the challenges faced by the analog circuit implementation, digital encoding and decoding schemes were developed for the CDMA NoC and illustrated in Fig.17 and Fig.18 respectively. In the encoding scheme illustrated in Fig.17, each data bit from different senders will be spread into S bits by multiplying it with a unique S-bit spreading code. The actual encoding operations are realized by XOR logic gates. Each bit of the S-bit encoded data generated by XOR operations is called a data chip. Then the data chips from different senders are arithmetically added together according to their positions in the S-bit sequences. In other words, all the first data chips from different senders are added together, and all the second data chips from different senders are added together, and so on. Therefore, after the add operations, we will get S sum values of S-bit encoded data. Each sum value is repre-

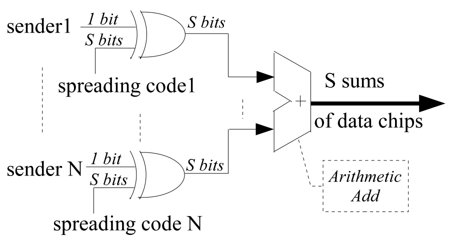


Fig. 17. *Digital CDMA Data Encoding Scheme.*

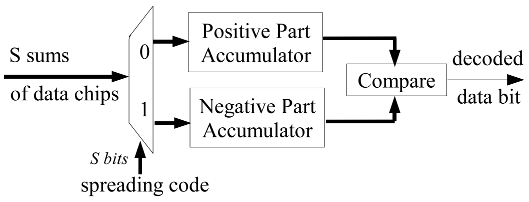


Fig. 18. *Digital CDMA Data Decoding Scheme.*

sented by a (log_2N)-bit number, where N is the number of senders in the network. Finally, as proposed in [5], binary equivalents of the S sum values are transferred to the receiving end one by one.

The digital decoding scheme for applying the CDMA technique into an on-chip network is illustrated in Fig.18. The decoding scheme accumulates the received sum values into two separate parts, a positive part and a negative part, according to the bit value of the spreading code used for decoding. For example, the received first sum value will be put into the positive part accumulator if the first bit of the spreading code for decoding is ‘0’, otherwise, it will be put into the negative part accumulator. The same selection and accumulation operations are also carried out on the other received sum values. If the spreading codes have orthogonal and balance properties, either the positive part or negative part is larger than the other after accumulating all sum values according to the bit values of the spreading code. Hence, the original data bit can be decoded by comparing the values of the two accumulators. Namely, if the positive accumulation value is larger than the negative accumulation value, the original data bit is ‘1’; otherwise, the original data bit is ‘0’.

2. Spreading Code Selection

As addressed previously, the presented encoding and decoding schemes require the spreading codes having both the orthogonal and balance properties. The orthogonal property in the context of the CDMA NoC means that the normalized auto-correlation of the spreading codes is 1, while the cross-correlation of the spreading codes is 0. The balance property means that the number of bit ‘1’ and bit ‘0’ in a spreading code should be equal. Several types of spreading codes have been proposed for CDMA communication, such as Walsh code, M-sequence, Gold sequence, and Kasami sequence, etc [25]. However, only Walsh code [25] has the required orthogonal and balance properties. Therefore, the Walsh code family is selected as the spreading code library for the proposed CDMA NoC. In an S-bit (S>0, S mod 4 = 0) length Walsh code set, there are S-1 sequences which have both the orthogonal and balance properties. Hence, the proposed CDMA NoC can have at most S-1 network nodes, in other words, S has to be at least N+1 for a N-node network.

3. Spreading Code Protocol

In a CDMA network, if multiple users apply the same spreading code to encode their data to be transferred at the same time, the encoded data will interfere with each other because of the loss of orthogonal property among the spreading codes. This situation is called spreading code conflict, which should be avoided. Spreading code protocol is a policy used to decide how to assign and use the spreading codes in a CDMA network in order to eliminate or reduce the possible spreading code conflicts. Several spreading code protocols have been proposed for CDMA packet radio network [56, 86]. Among these proposed spreading code protocols, only Transmitter Based Protocol (T protocol) [86] and Transmitter-Receiver-Based Protocol (T-R protocol) [56] are conflict-free if the users in the network send data to each other randomly. In the T protocol, each user is allocated a unique spreading code to encode and transfer data to others. However, the main drawback is that a receiver can not choose the proper spreading code for decoding because it can not know who is sending data to it. In the T-R protocol, two unique spreading codes are assigned to each user in the network, and then a user will generate a new spreading code from the assigned two spreading codes for its data encoding. Hence, it suffers from using a large amount of spreading codes and complicated decoding scheme.

In order to solve this problem, an Arbiter-Based T protocol (A-T protocol) is developed for the CDMA on-chip network. In a CDMA network which applies A-T protocol, each user is assigned with a unique spreading code for data transfer. When a user wants to send data to another user, he will send the destination information of the data packet to the arbiter before starting data transmission. Then the arbiter will inform the requested receiver to prepare the corresponding spreading code for data decoding according to the sender. After the arbiter has got the acknowledge signal from the receiver, it will send an acknowledge signal back to the sender to grant its data transmission. If there are several users who want to send data to the same receiver, the arbiter will grant only one sender at a time. Therefore, by applying the A-T protocol, spreading code conflicts in the CDMA NoC can be eliminated.

5.4 CDMA NoC Structure

Based on the schemes and protocols presented in section 5.3, a small CDMA on-chip network was developed for study and functional verification. Its network structure is illustrated in

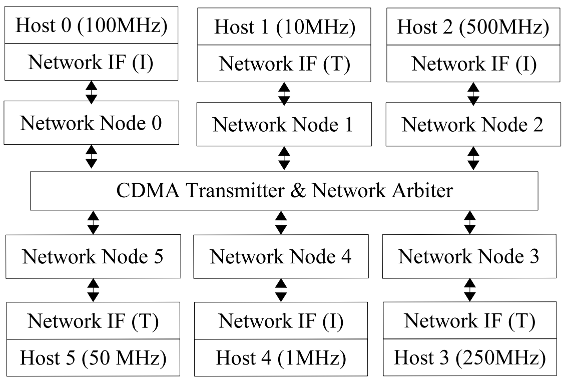


Fig. 19. *Six-Node CDMA On-Chip Network Structure.*

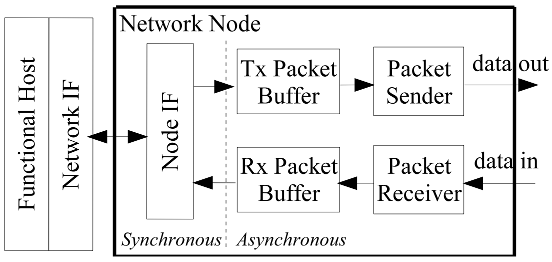


Fig. 20. *The Block Diagram of the Network Node for CDMA NoC.*

Fig.19. By comparing it with the bidirectional ring NoC structure illustrated in Fig.11, the CDMA NoC has two unique blocks named as ‘CDMA Transmitter’ and ‘Network Arbiter’ besides the common ‘network node’ blocks. The function of each block in the CDMA NoC will be introduced in the following three subsections.

5.4.1 Network Node

The block diagram of the ‘Network Node’ design of the CDMA NoC is illustrated in Fig.20. The arrows in Fig.20 represent the direction of data packet flow. The ‘Network IF’ block in the network node which belongs to the functional host is an interface block for connecting a functional host with a ‘Network Node’ through VCI or OCP interface standard. The function of each sub-block in the ‘Network Node’ will be described in the following paragraphs.

(1) Node IF. This block is used to receive data from the ‘Network IF’ block of a functional host through the applied interface standard. The main task is to assemble the received data into packet format and send the packet to ‘Tx Packet Buffer,’ or disassemble the received packet from ‘Rx Packet Buffer’ and send the extracted data to the functional host.

(2) Tx/Rx Packet Buffer. These two blocks are buffers for sending or receiving data packets.

‘Tx Packet Buffer’ is used to store the data packets from ‘Node IF’ block, and then deliver the packets to ‘Packet Sender.’ The ‘Rx Packet Buffer’ stores the received packets and delivers them from ‘Packet Receiver’ to ‘Node IF.’

(3) Packet Sender. This block will fetch a data packet from ‘Tx Packet Buffer’ block by an asynchronous handshake protocol when it finds that the buffer is not empty. Then the destination information of the fetched packet will be extracted and sent to ‘Network Arbiter.’ After ‘Packet Sender’ gets the grant signal from the arbiter, it will start to send the data packet to ‘CDMA Transmitter’ block.

(4) Packet Receiver. This block will wait for the sender information from ‘Network Arbiter’ to select the proper spreading code for decoding. After the spreading code for decoding is ready, the receiver will send an acknowledge signal back to ‘Network Arbiter’ and wait to receive and decode the data from ‘CDMA Transmitter,’ and then send the decoded data to ‘Rx Packet Buffer’ in packet format.

5.4.2 Network Arbiter

As addressed in section 5.3, an A-T spreading code protocol is applied in the CDMA on-chip network to avoid spreading code conflicts. By applying the A-T protocol, every node needs to get the grant from the ‘Network Arbiter’ before it can start to send data packets to the network. Thus, the ‘Network Arbiter’ block is the core component for implementing the A-T protocol in the CDMA NoC. Its main functions are described in the following paragraph.

After receiving a data sending request from a network node, the ‘Network Arbiter’ needs to inform the requested receiver node to prepare the proper spreading code for decoding. When the ‘Network Arbiter’ gets the acknowledge signal from the requested receiver node, it will send a grant signal back to the sender node to enable the data transfer process. If there are multiple network nodes requesting to send data to the same destination node simultaneously or at different time, the ‘Network Arbiter’ will apply ‘round-robin’ scheme or the ‘first-come first-served’ principle, respectively, to guarantee that there is only one sender sending data to one specific receiver at a time. However, if the destination nodes requested by the sender nodes are different, these requests from different senders will be handled in parallel without blocking each other. The ‘Network Arbiter’ in the CDMA NoC is different from the arbiter used in a conventional bus. The reason is that the ‘Network Arbiter’ in the CDMA NoC is only used to set up spreading codes for receiving and it handles the requests concurrently in time domain. In contrary, a conventional bus arbiter is used to allocate the usage of the common communication media among the users in the time-division manner.

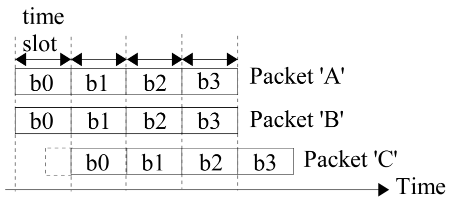


Fig. 21. *Bit-Synchronous Transfer Scheme.*

5.4.3 CDMA Transmitter

The ‘CDMA Transmitter’ block is the core component to perform the data encoding and transfer operations in the CDMA NoC. As introduced in section 5.4.2, a network node will start to send data packets to the ‘CDMA Transmitter’ after it gets the grant signal from the ‘Network Arbiter.’ When the ‘CDMA Transmitter’ receives the data to be transferred, it will encode the data bits with the unique spreading code of the sender node.

Although each network node may send data to the ‘CDMA Transmitter’ block independently and randomly, the ‘CDMA Transmitter’ block applies a bit-synchronous transfer scheme to coordinate the asynchronous data transfer processes. The basic principle of this bit-synchronous transfer scheme is that the data from different nodes will be encoded and transmitted synchronously in terms of data bits rather than any clock signals. Fig.21 illustrates the principle of bit-synchronous transfer scheme by an example in which network nodes ‘A’ and ‘B’ send data packets to ‘CDMA Transmitter’ simultaneously and node ‘C’ sends a data packet later than node ‘A’ and ‘B’. In this situation, the data packet from node ‘A’ will be encoded and transmitted together with the packet from node ‘B’ synchronously in terms of each data bit. As the data packet from node ‘C’ arrives at a later time point, the transmitter will handle the data bit from ‘Packet C’ together with the data bits from packet ‘A’ and ‘B’ at the next start point of the time slot for bit encoding and transmitting processes. The dot-line frame at the head of the ‘Packet C’ in Fig.21 illustrates the waiting duration when the ‘Packet C’ arrives in the middle of the time slot for handling the previous data bit. The time slot for handling a data bit is formed by a four-phase handshake process.

The main advantage of the presented bit-synchronous scheme is that it avoids the interferences caused by the phase offsets among the orthogonal spreading codes when the data bits from different nodes are encoded and transmitted asynchronously with each other. Because the nodes in the network can request data transfer randomly and independently of each other, ‘CDMA Transmitter’ applies the ‘first come, first served’ mechanism to ensure that the data encoding and transmission are performed as soon as there is a data transfer request.

5.5 Front-End Synthesis and Simulation Results

The CDMA NoC structure illustrated in Fig.19 is modeled in RTL using VHDL for functional verification and performance estimation. The principle of data encoding and decoding schemes illustrated in Fig.17 and Fig.18 use an example of processing and delivering one data chip of encoded data from the sender to the receiver at one time. Since one original data bit will be spread into S bits after encoding, the degree of data transfer parallelism between the 'CDMA Transmitter' and 'Network Node' blocks affects the data transfer latency of the CDMA NoC significantly. Namely, increasing the number of data bits encoded and delivered via 'CDMA Transmitter' at one time can reduce the data transfer latency of the CDMA NoC and vice versa. However, increasing the data processing and delivering parallelism will incur larger area cost. Thus, in order to figure out the trade-off between the data processing parallelism and the area cost, the 'Packet Sender,' 'CDMA Transmitter,' and 'Packet Receiver' blocks are realized with four different data path widths. According to the number of data bits transferred from the 'Packet Sender' in a sender node to the 'Packet Receiver' in the receiver node through 'CDMA Transmitter,' the four data path widths which have been applied are named as 1-, 8-, 16-, and 32-bit schemes.

Except the different data path widths, the CDMA NoC illustrated in Fig.19 has the similar network configurations as the bidirectional ring NoC illustrated in Fig.11. The six network nodes work at different clock rates. The network nodes communicate with each other through 'CDMA Transmitter' and 'Network Arbiter' blocks. The spreading codes used in the network are six 8-bit Walsh codes. The 'functional host' blocks and their 'Network IF' blocks are not realized with any real IP blocks, instead, they are simulated by adding stimulus signals on each 'Network Node' block according to the BVCI standard. A 0.18μm standard-cell library is used in the synthesis. Two aspects of the realization results, logic gate area cost and data transfer latency, are presented in the following paragraphs.

1. Logic Gate Area Cost

The logic gate area cost of each block in the CDMA NoC and the total of the six-node network under different data path widths are listed in Table 5. Although these figures only include the area cost of logic gates, they are indicative enough to estimate the design complexity and compare of alternative implementations. An estimation of the area cost of wires will be presented in section 7.3.3. From Table 5, we can see that when the data path width is increased from 1 to 32 bits in the CDMA NoC, the area cost of the network becomes 2.4 times larger because more logic are used to perform parallel data encoding and decoding. To be noticed in Table 5 is that the area cost of the 32-bit version of 'Packet Sender' block is smaller than the costs under other data path widths. The reason is that the data width of the output of 'Tx Packet Buffer' block is 32 bits, thus the 'Packet Sender' block needs control logic to adjust the fetched packet cells to be sent out according to the applied data path width

Table 5. *Area Cost of the Six-Node CDMA Network.*

Block Name		***Area Cost (K equivalent gates)***			
		1-bit	**8-bit**	**16-bit**	**32-bit**
Node IF	**Target**	1.600			
	Initiator	3.882			
Tx/Rx Packet Buffer		6.101			
Packet Sender		1.505	1.509	1.513	0.962
Packet Receiver		1.977	7.331	13.718	26.488
CDMA Transmitter		0.879	3.970	7.730	15.161
Network Arbiter		0.936			
Area cost of the 6-node CDMA network		***113.145***	***148.369***	***191.037***	***272.806***

if it is smaller than 32 bits. However, when the path width is increased to 32 bits, the output data width adjusting logic is not needed in the ‘Packet Sender’ block.

2. Data Transfer Latency

The data transfer latency of the six-node CDMA NoC consists of two parts, the synchronous transfer latency (STL) and asynchronous transfer latency (ATL), which have the same names as the latency components of the ring network presented in Chapter 4.

Because the same control logic of ‘Node IF’ block design is applied in the two networks, the STL values of the CDMA NoC are the same with the values listed in Table 3. The constant values in Table 3 are caused by the handshakes in the asynchronous domain. They are independent of the local clock rate but belong to the synchronous transfer processes. Therefore, they are included in the STL part.

The ATL of the CDMA NoC has the same meaning with the ATL of the ring NoC presented in Chapter 4, however, the ATL of the CDMA NoC has less portions because there are no bypass packets going through a network node due to the ‘one-hop’ data transfer scheme. The concept of the ATL portions of the CDMA NoC is illustrated in Fig.22 with an example where ‘Network Node 0’ sends one data packet to ‘Network Node 2.’ The black arrows in Fig.22 represent the direction of packet transfer. The three portions of the ATL, PLL, PTL, and PSL, are marked by grey arrows in Fig.22 and will be briefly introduced in the following paragraphs.

(1) Packet Load Latency (PLL). This is the time used by the ‘Packet Sender’ block in a network node to fetch a data packet from ‘Tx Packet Buffer’ and prepare to send the packet out via ‘CDMA Transmitter.’

(2) Packet Transfer Latency (PTL). This latency refers to the time used to transfer one data packet from the ‘Packet Sender’ of the sender node to the ‘Packet Receiver’ of the receiver node via the ‘CDMA Transmitter’ and ‘Network Arbiter’ blocks using a handshake protocol.

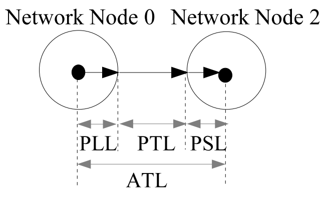


Fig. 22. *ATL Portions of the CDMA NoC.*

(3) Packet Storing Latency (PSL). The receiver node needs to spend a certain amount of time to store the received data packet into 'Rx Packet Buffer' after it receives a data packet. This time duration is measured as PSL.

The measured values of ATL portions in the CDMA NoC under different data path configurations are listed in Table 6. The ATL value of the CDMA NoC can be calculated by directly adding the three portions under the same configuration together. By using the latency values listed in Table 3 and 6, we can estimate the latency range of the six-node CDMA network with 8-bit channel width in a non-congestion situation. From Fig.19, we can see that the worst case would be a 3-data-cell packet transfer from functional host 4 (1 MHz) to host 1 (10 MHz), and the estimation value is 8547.7*ns*. The best case would be a 1-data-cell packet transfer from the host 2 (500 MHz) to host 3 (250 MHz), and the estimation value is 94.7*ns*. If the data width is increased to 16 bits, the best-case and worst-case estimation values will change to 8488.6*ns* and 75.0*ns*, respectively.

5.6 SystemC Modeling and Performance Estimation

As the complexity of an on-chip system is continuously growing, a system designer needs to estimate the performance of the on-chip system in an early design stage. Therefore, as an important part of an on-chip system, the on-chip communication network needs to be evaluated in terms of performance and cost in a fast and flexible way to facilitate the design exploration of the system architecture. Section 5.5 presented a design exploration on different data path widths of the CDMA network by using RTL realizations. Although the design exploration

Table 6. *ATL Values of the CDMA NoC.*

ATL Portion		***1 data cell***	***2 data cells***	***3 data cells***
PLL (ns)		5.7	5.7	5.7
PTL (ns)	**1-bit**	384.6	768.9	1153.7
	8-bit	45.9	88.4	130.9
	16-bit	26.2	49.0	71.8
	32-bit	14.7	26.0	37.8
PSL (ns)		5.5	5.5	5.5

using RTL realizations has the advantage of accuracy, it is time consuming and not efficient due to the tedious process of synthesis and simulation. Therefore, a fast and flexible method of evaluating the CDMA network is developed in this work by using SystemC modeling technique. This section will present the SystemC model and the performance estimation method of the CDMA network.

5.6.1 Modeling the CDMA NoC with SystemC

SystemC [76] is a C++ class library which has been developed to meet the requirement for system modeling. Since a SystemC model is described by a programming language, the abstraction level of the model can be very flexible and the simulation can run at a faster speed than a RTL model if the SystemC model is built in a higher level. Therefore, a transaction-level SystemC model of the CDMA NoC is developed to facilitate design exploration.

Transaction-Level Modeling (TLM) [32] is a modeling style which bases on the features of channels and interfaces of SystemC 2.1. With TLM style, the communication processes are modeled by calling the interface functions of a channel without knowing the implementation details of the interface functions in the channel. Therefore, by separating the definition from the implementation of the interface functions, the system model only needs to concern the transactions among modules and data flows in the system without the details of the communication method. The SystemC model of the CDMA NoC is built according to the block diagrams illustrated in Fig.19 and Fig.20 in order to keep the uniform hierarchy between different levels of abstractions. Each block in the CDMA NoC is modeled as a channel. The channels and relationships of interface function calls within a 'Network Node' are illustrated in Fig.23. Fig.24 presents the interface functions and calling relationships among 'Network Node,' 'Network Arbiter,' and 'CDMA Transmitter.' In Fig.23 and Fig.24, each grey square at the boundary of a channel represents an interface of that channel, and each grey circle at the boundary of a channel represents an instantiated interface port of other channels. The arrows in the figures point from the instantiated interface port to its original channel interfaces. For example, as illustrated in Fig.24, the 'CDMA Transmitter' block communicates with a 'Network Node' block by instantiating an interface, called 'tx_if,' of the 'Network Node,' then calling the functions in the 'tx_if' interface to get data from the 'Network Node' block.

5.6.2 Performance Estimations

Although a transaction-level SystemC model is good for fast functionality modeling and simulation, in comparison with the RTL realization, it has the shortcoming of getting accurate performance estimation results, such as data transfer latency. Therefore, a performance estimation method which combines the merits from both the SystemC model and the RTL

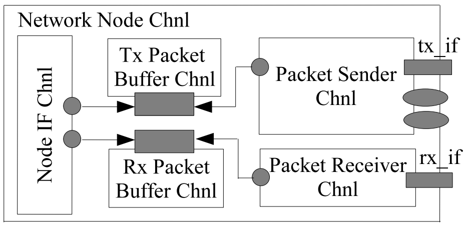


Fig. 23. *Channels and Interfaces in the CDMA Network Node.*

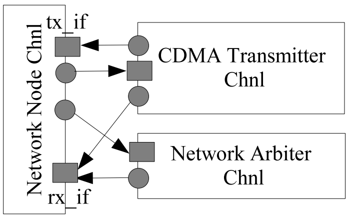


Fig. 24. *Channels and Interfaces in the CDMA NoC.*

realization is developed. It bases on the SystemC model of the CDMA NoC. The method can be summarized as the following four steps.

Step1: Model each block of the CDMA NoC as a channel using SystemC and build the CDMA NoC model in transaction-level according to the block hierarchy.

Step2: Realize each block of the CDMA NoC in RTL and do the synthesis and gate-level simulation using the target technology library.

Step3: Record the latency information of the handshake processes among the blocks from the gate-level simulation, and then back annotate the latency values to the corresponding channels in the SystemC model of the CDMA NoC.

Step4: Estimate the performance of the CDMA NoC under different configurations by simulating the timed SystemC model.

Based on the presented estimation method, the performance of the CDMA NoC under three different types of configurations, CDMA channel widths, number of network nodes, and traffic patterns, are estimated through the simulations in an open-loop environment. The setup of the simulation environment is illustrated in Fig.25. In order to concentrate on simulating the global asynchronous network other than the local synchronous communications in the CDMA NoC, the ‘Network Node’ block is revised as illustrated in Fig.25. The ‘Node IF’ is replaced by a ‘Packet Source’ block which generates data packets according to a specific traffic pattern. The size of ‘Tx Packet Buffer’ block is set to be large enough for storing all packets generated by the packet source during the simulation in order to make the simulation to be an open-loop simulation, which ensures that the traffic produced by the source is not

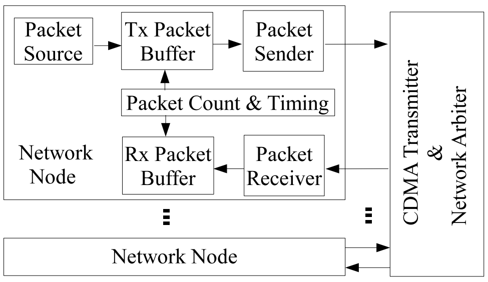


Fig. 25. *Open-Loop Simulation Environment.*

influenced by the status of the network. The ‘Rx Packet Buffer’ is used to store the received packets. The ‘Packets Count & Timing’ process is added on the ‘Tx/Rx Packet Buffer’ blocks for counting and recording the packet transfer information during simulations. The simulation results are presented in the following paragraphs.

1. Different CDMA channel widths

A preliminary data transfer latency estimation of a six-node CDMA NoC with different channel widths has been presented in section 5.5. However, that estimation does not include packet transfer congestions in the simulations. With the SystemC model and the presented performance estimation method, an overall data transfer latency estimation of the six-node CDMA NoC is performed in transaction level. The traffic pattern used in the simulations is independent and uniform traffic. It means that the same amount of packets is independently generated at each network node, and the destinations of the generated packets in each node are uniformly distributed to all the other network nodes. Each node sent 5000 packets to the network, and the average number of data cells of the packets is two. Fig.26 gives the average ATL of delivering a 32-bit data cell with different channel widths in the CDMA NoC. From the values illustrated in Fig.26, we can see that the packet arbitration contentions and CDMA transfer contentions increase the latency severely in comparison with the ATL values listed in Table 6.

2. Different number of network nodes

Another configuration of the CDMA NoC which has been explored by using the SystemC model is the number of network nodes. The traffic pattern and other simulation configurations are the same with the ones used in the simulations of different channel widths except that each network node sent 500 packets to every other node, thus increasing the traffic proportionally to the network size. The channel width used in the simulations is 8 bits. The average ATL values under different numbers of network nodes are illustrated in Fig.27. From the results, we can see that the transfer latency increases when the number of network nodes increases since the probability of contentions increases.

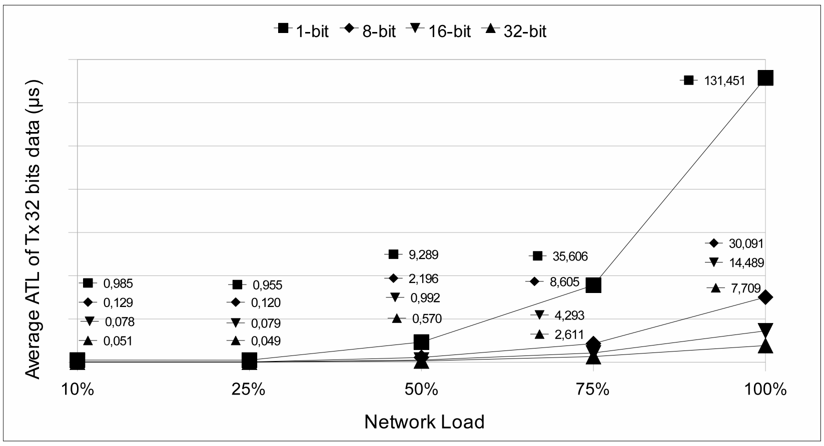


Fig. 26. *ATL Estimations with Different Channel Widths.*

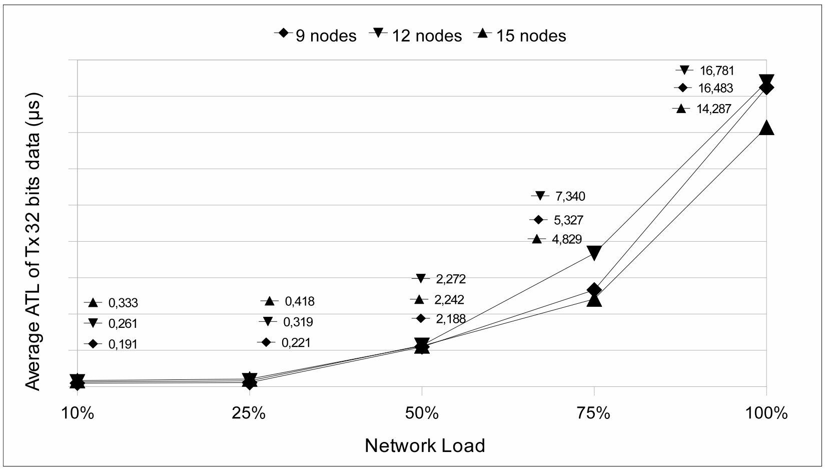


Fig. 27. *ATL Estimations with Different Number of Network Nodes.*

3. Different traffic patterns

The performance estimations presented in the two previous paragraphs are all under the uniform traffic pattern. In a real application, a hot-spot traffic pattern is more likely to appear. Hence, the performance of the CDMA NoC with different numbers of network nodes is estimated under a hot-spot traffic pattern. In the simulations, Node 1 is selected as the 'hot' node and the 'hot' degree is 0.25, which means that 25% of the generated packets in each node are transferred to Node 1. The other generated packets are still uniformly distributed to all the other nodes besides Node 1. The channel width used in the simulations is still 8 bits.

The average ATL values of transferring a 32-bit data cell with hot-spot traffic is illustrated in Fig.28. In comparison with the latency values illustrated in Fig.27, the CDMA NoC has

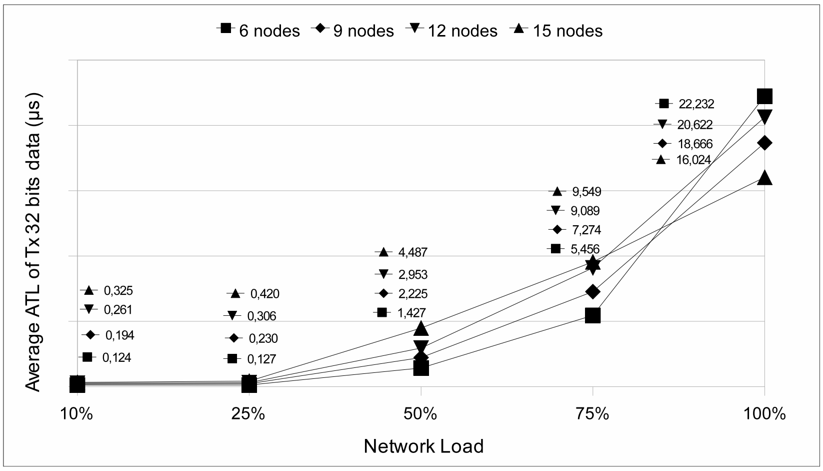


Fig. 28. *ATL Estimations with Hot-Spot Traffic Pattern.*

similar latency values under hot-spot traffic when the network load is smaller than 50%. Under the heavier network loads, the transfer latencies become larger. It means that the CDMA NoC is not sensitive to the balance of network load when the network load is light.

The presented SystemC model and performance estimation method of the CDMA NoC are very important for applying it in an on-chip system design because they present a way to make a fast estimation of the NoC in the early design stage.

6. A CROSSBAR ON-CHIP NETWORK

By applying the CDMA technique, the CDMA NoC can provide non-blocking concurrent data transfer service for an on-chip system. However, the CDMA NoC is not the only structure that can provide such kind of communication service. The crossbar structure is another type of non-blocking concurrent data transfer structure. Therefore, in order to examine the characteristics of the CDMA NoC, a network which applies crossbar structure to switch data among network nodes was developed for the comparison purpose. This chapter presents the structure and front-end realization results of the crossbar network developed in this work.

6.1 Introduction of Crossbar Structure

Crossbar is a well-known and widely accepted structure for composing a circuit-switched network. A four-port crossbar switch structure is illustrated in Fig.29 in order to explain the data transfer principle of crossbar. As illustrated in the figure, an input port in a crossbar structure can be connected to any output ports by optionally closing the switches between input and output lines. For correct operations, one output can be connected to at most one input. Therefore, we can see that the data transfers in the crossbar switch are non-blocking because a dedicated data channel can be set up from each input to its selected output without any conflicts with other channels if the selected outputs are different.

In order to apply the crossbar structure in an on-chip environment, a method of realizing the cross points needs to be settled. An on-chip crossbar network for SoC designs has already been presented in [57], the cross points in the network were implemented in circuit level with

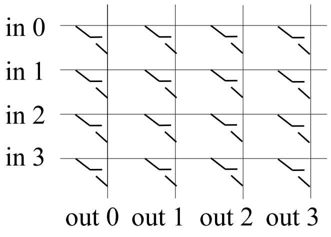


Fig. 29. *An Example of Crossbar Structure.*

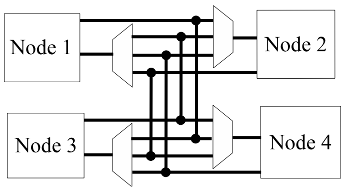


Fig. 30. *A Four-Node Crossbar Switch Structure.*

transistors. Therefore, it does not suit the purpose of designing a crossbar network in RTL with the same network configuration as the CDMA NoC for comparisons. In this design, the crossbar structure is realized by using multiplexers. A four-node crossbar switch structure realized by multiplexers is illustrated in Fig.30 as an example. From the figure, we can see that the crossbar structure is realized by using four multiplexers to set up data channels among network nodes. Each multiplexer is called a channel multiplexer to emphasize its function of setting up data channels in the network. The number of channel multiplexers is equal to the number of network nodes. By applying this structure, the crossbar network can be modeled in the same abstract level as the CDMA network does by using HDL.

6.2 Network Structure and Network Node Design

The crossbar network structure developed in this work is illustrated in Fig.31. In order to compare the crossbar network with the CDMA network, the configuration of the crossbar network is kept as same as possible with the CDMA network illustrated in Fig.19. The crossbar network is also a six-node network which applies GALS scheme. The configurations of the functional host are same with the CDMA network. The only difference is that the ‘CDMA Transmitter’ and ‘Network Arbiter’ blocks in the CDMA network are replaced by a ‘Crossbar Switch’ block composed of channel multiplexers. Each channel multiplexer contains arbitration logic to control the selection of the output. When multiple data transfer requests from different network nodes come to a channel multiplexer simultaneously, the multiplexer will record the requests and serve one request at a time. For the requests that come at different time, a channel multiplexer will serve the requests by the principle of ‘first come, first served.’ After setting up the data channel, a channel multiplexer will send a grant signal back to the sender node to enable data transfer processes.

The network node used in the crossbar network has the same structure as the network node of the CDMA NoC illustrated in Fig.20. The only difference is the functions of the ‘Packet Sender’ and ‘Packet Receiver’ blocks in the asynchronous domain of a node. In the crossbar network, after fetching a packet from the buffer, the ‘Packet Sender’ will assert a request signal to the channel multiplexer attached to the receiver node. Then the ‘Packet Sender’ will wait for the grant signal from the requested channel multiplexer before it starts to send a data

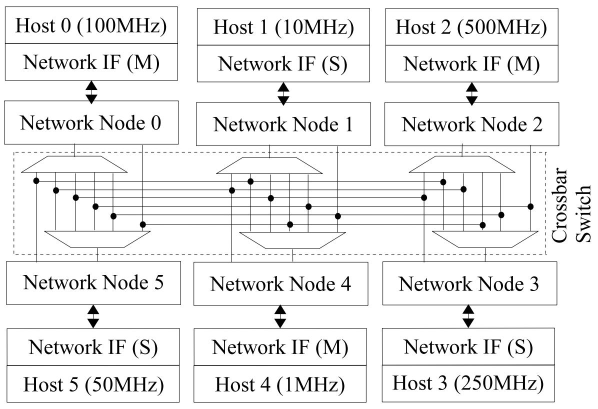


Fig. 31. *Crossbar Network Structure.*

packet to the receiver node through the multiplexer. After a data packet transfer is completed, the 'Packet Sender' needs to clear the request signal in order to release the requested channel multiplexer for serving requests from other nodes. The 'Packet Receiver' block in the crossbar network will wait for the request from the multiplexer block attached with it. When a request comes, the 'Packet Receiver' block will receive the data packet and then deliver it to 'Rx Packet Buffer' block.

6.3 Front-End Synthesis and Simulation Results

In order to compare with the CDMA network, the six-node crossbar network is also realized with four different data path widths, 1-bit, 8-bit, 16-bit, and 32-bit, as the CDMA network realization presented in section 5.5. The 'functional host' blocks and their 'Network IF' blocks are not realized with any real IP blocks, instead, they are simulated by adding stimulus signals on each 'Network Node' block according to the BVCI standard. The same 0.18μm standard-cell library is used in the synthesis. Two aspects of the front-end realization results, logic gate area cost and data transfer latency, are presented in the following paragraphs.

1. Logic Gate Area Cost

The logic gate area costs of each basic component and the whole crossbar network with different data path widths are listed in Table 7. An estimation of the area cost of wires will be presented in section 7.3.3. From the table, we can see that the gate area costs of 'Packet Sender' and 'Packet Receiver' blocks increase slightly when the data path width increases. When the data path width is 1 bit, the 'Packet Receiver' block has a larger area caused by the logic for arranging the received serialized data bits into packet format. The reason of the large area increase of the 'Channel Multiplexer' blocks is that the logic for setting up data channels is multiplied when the data path width increases from 1-bit to 32-bit.

Table 7. *Area Cost of the Six-Node Crossbar Network.*

Block Name		***Area Cost (K equivalent gates)***			
		1-bit	**8-bit**	**16-bit**	**32-bit**
Node IF	**Target**	1.600			
	Initiator	3.882			
Tx/Rx Packet Buffer		6.101			
Packet Sender		1.514	1.518	1.523	1.538
Packet Receiver		0.995	0.906	0.921	0.934
Channel Multiplexer		0.236	0.325	0.437	0.660
Area cost of the 6-node crossbar network		***106.393***	***106.415***	***107.206***	***108.698***

Table 8. *ATL Values of the Crossbar Network.*

ATL Portion		***1 data cell***	***2 data cells***	***3 data cells***
PLL (ns)		3.0	3.0	3.0
PTL (ns)	**1-bit**	112.5	211.8	354.4
	8-bit	17.3	32.7	47.5
	16-bit	10.6	19.4	27.8
	32-bit	7.5	12.7	17.9
PSL (ns)		4.7	4.7	4.7

2. Data Transfer Latency

The data transfer latency of the realized six-node crossbar network has the same components, STL and ATL, as the CDMA network has. The STL of the crossbar network has the same values as the STL of the CDMA network presented in Table 3 because the same node structure and the same design of the 'Node IF' and 'Tx/Rx Packet Buffer' blocks are applied in the crossbar network realization.

The ATL of the crossbar network has the same portions of the ATL of the CDMA network. The measured latency values during the gate-level simulation of the crossbar network are listed in Table 8. These values are measured in a contention-free situation during the simulation. The PTL values listed in the table decrease roughly linearly as the data path width increases from 1 bit to 16 bits. When the data path width is increased from 16 bits to 32 bits, the improvement of PTL is not large if the size of a packet is small, such as 1-data-cell. The reason is that the overhead of setting up data channels takes a larger portion of the PTL value when a packet is small. Hence, the overall PTL value can not be reduced linearly because of the overhead caused by the data channel setting up. From the table, we can also see that the overall ATL values of the crossbar network are not very large due to its direct and non-blocking data transfer scheme.

By using the latency values listed in Table 3 and Table 8, we can estimate the latency range of the six-node crossbar network with 8-bit channel width in a non-congestion situation. From

Fig.31, we can see that the worst case would be a 3-data-cell packet transfer from functional host 4 (1 MHz) to host 1 (10 MHz), and the estimation value is 8460.8*ns*. The best case would be a 1-data-cell packet transfer from the host 2 (500 MHz) to host 3 (250 MHz), and the estimation value is 62.6*ns*. If the data width is increased to 16 bits, the best-case and worst-case estimation values will change to 8441.1*ns* and 55.9*ns*, respectively.

7. COMPARISONS

By realizing the ‘one-hop’ non-blocking transfer scheme, the CDMA NoC can reduce the variations of data transfer latencies caused by data routing in a direct connection network. However, the price of this feature is the area and performance overhead incurred by introducing data encoding and decoding operations. Therefore, in order to examine the pros and cons of the CDMA NoC thoroughly, this chapter presents the comparisons between the CDMA network and the two on-chip networks developed in this work, the bidirectional ring network and the crossbar network, in the first three sections. In the last section of this chapter, performance comparisons between the CDMA NoC and other NoC designs found in publications are presented.

7.1 Data Transfer Principles

The data transfer principle is the main difference between the CDMA NoC and the two other networks. The ring network presented in Chapter 4 and the crossbar network presented in Chapter 6 both apply a direct and plain data transfer principle which means that the data are transferred in their original form, while the CDMA NoC introduces data encoding and decoding operations for data transfers.

The main advantage of applying CDMA technique is the feature of data transfer concurrency realized by using data encoding and decoding operations. Although the data transfers in the ring NoC can also be concurrent if they take place in different channels among the network nodes, the data transfer parallelism in the ring network is largely limited by the possible traffic congestions in a channel. The congestions are unavoidable because a channel between two network nodes is shared by all the packets which need to pass this channel in a time-division manner. This character exists in all the networks which apply direct connections, such as 2-D mesh, torus, and fat-tree. However, in the CDMA and crossbar networks, this type of congestions are avoided because of the feature of non-blocking data transfers. The difference between the CDMA and crossbar networks is that the CDMA network applies orthogonal codes to set up independent data transfer channels, whereas a crossbar structure uses direct circuit connections.

Another advantage of applying the CDMA technique is that it can easily realize multicast data transfer by requesting multiple receiver nodes to use the same spreading code for receiving. In the ring network, the multicast transfer can be realized only by sending multiple copies of a data packet to its multiple destinations, unless extra logic is added in each network node to copy the multicast or broadcast packet both to the functional host and to the output link to the next node. In the crossbar network, the multicast transfer can be done by sending multiple copies of packets to several channel multiplexers. All of these operations either increase the traffic load in the network, or complicate the network implementation.

The main drawback of applying the CDMA transfer principle is that the data transfer efficiency obtained by concurrent data transfers is compromised by the latency introduced by the data spreading scheme. As presented in section 5.3, one data bit will be extended to S bits for CDMA data transfers. As the number of nodes in the NoC increasing, the spreading code width will increase. Then the transfer latency caused by data spreading will also increase.

7.2 Network and Node Structures

By comparing the network structures of the ring NoC illustrated in Fig.11, the CDMA NoC illustrated in Fig.19, the crossbar NoC illustrated in Fig.31, the similarity among the three network structures is the way of applying GALS scheme in the networks. All networks apply asynchronous scheme in the communications among network nodes and synchronous scheme in the local communications between a network node and the functional host attached to it.

The main difference observed is that the ring network applies distributed communication scheme while the other two networks apply centralized scheme. The distributed scheme means that the data traffic load in the ring network distributes to all the links among network nodes. The advantage of this distributed scheme is the scalability, whereas the disadvantage is that the data transfer latency between two network nodes can be largely different because the data may be delivered through different routes.

In the CDMA network and the crossbar network, all network nodes communicate each other through a central block. In the CDMA network, the central block is composed of ‘CDMA Transmitter’ and ‘Network Arbiter’ blocks, while, in the crossbar network, the ‘Crossbar Switch’ block is the central one. This centralized scheme is different from conventional bus structures since it provides parallel data transfers both in time and space domains by either applying CDMA technique or setting up parallel channels, whereas a bus structure supplies data transfer service among users in a time-division manner. The advantage of the centralized scheme is the ‘one-hop’ concurrent data transfer ability obtained by setting up parallel channels either in code domain or multiple physical links.

Due to the centralized scheme, the network node designs of the CDMA and crossbar network illustrated in Fig.20 have simpler structure than the design of the ring network illustrated in

Fig.12. In the ring network, every network node needs to take care of receiving and forwarding the bypass packets. Hence, ‘Communication Controller’ and ‘Packet Distributor’ blocks are included in the network node to handle the packet routing processes. However, these two blocks are removed in the network nodes of the CDMA and crossbar networks because all data packets are delivered to their destination nodes directly without routing. One drawback of the node design in the ring network is that more ‘Communication Layer’ blocks are needed in each network node in order to set up more links with other nodes if the data transfer parallelism needs to be increased or the topology needs to be changed, whereas, the node structure in the CDMA and crossbar network does not need to be changed in those situations.

7.3 Performance

The performance comparison of the three networks bases on the three six-node networks illustrated in Fig.11, Fig.19, and Fig.31. Area cost, data transfer latency, number of data wires, and dynamic power consumption are compared in the following subsections.

7.3.1 Area Cost of Logic Gates

For comparison purpose, the logic gate area costs of the node designs of the three networks are illustrated in Fig.32. The portions of each sub-blocks in each node design are also illustrated in the figure. The data presented in the figure base on the values listed in Table 2, Table 5, and Table 7 under the 32-bit category. From the figure we can see that the CDMA network node has the largest gate area cost when the data path width is set to 32 bits in all the three networks. This is due to the large logic for parallel decoding in the ‘Packet Receiver’ blocks of the CDMA network. In order to get an overall view of the gate area costs of all the three networks, Fig.33 illustrates the total gate area costs of the networks with different data path widths. From the figure we can see that the crossbar network takes the smallest area costs with all the situations. The reason is that the crossbar network has the simplest structure since it does not include either the data encoding and decoding operations in the CDMA network or the data routing operations in the ring network. However, if the area cost of wires is included, the area cost of the crossbar network would increase a lot because it requires a large number of connection wires. This will be discussed later in section 7.3.2 and 7.3.3. When the data path width increases in the crossbar network, only the ‘Packet Receiver’ and ‘Packet Sender’ blocks need to be adjusted a little to suit the different path widths. Hence, its overall gate area cost changes slightly with different data path widths. The same situation would also happen in the ring network if it was realized with different data path widths. The small changes in the ‘Packet Receiver’ and ‘Packet Sender’ blocks to be compatible with different data path widths would not affect the overall network area in the ring network. The gate area cost

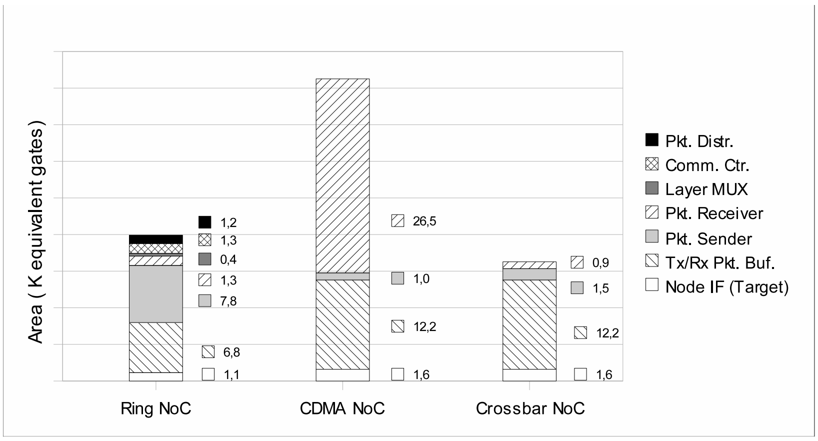


Fig. 32. *Logic Gate Area Costs of Network Nodes.*

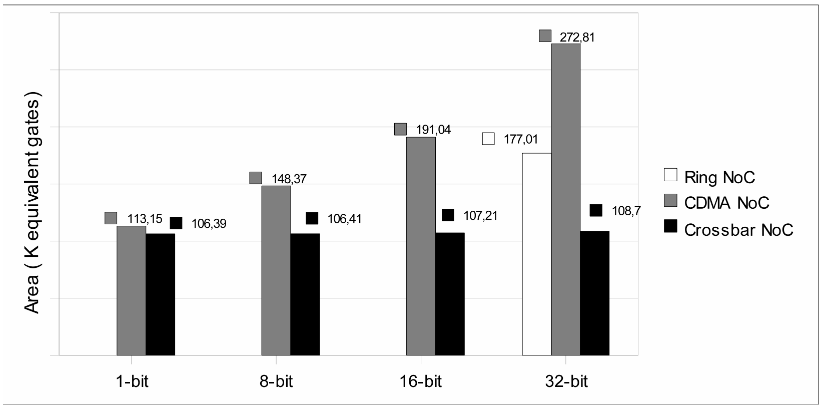


Fig. 33. *Total Logic Gate Area Costs of the Three Networks.*

of the CDMA network increases almost linearly as the data path width increases. This is because more logic components are required to perform parallel data encoding and decoding processes in the CDMA network. With 16- and 32-bit data path widths, the CDMA network loses its area cost advantage in comparison with the ring network. Therefore, in terms of logic gate area cost, the 8-bit version of the CDMA network could be an optimal alternative to replace the ring network.

7.3.2 Number of Data Connection Wires

Unlike the distributed structure applied in the ring network, the centralized structure applied in the CDMA and crossbar networks needs a large amount of data connections wires to set up parallel data channels among network nodes. Therefore, this subsection presents a comparison about this issue between the CDMA and crossbar networks.

In comparison with the CDMA network, the crossbar network has smaller area cost by setting up parallel physical connections among nodes. However, these parallel connections cause a large overhead of the required number of data connection wires. The number of data connection wires in a crossbar network refers to the number of data wires between 'Network Node' blocks and channel multiplexer blocks. In the CDMA network, this number refers to the number of data wires between 'Network Node' blocks and 'CDMA Transmitter' block.

The equation of calculating the number of data wires in the crossbar network is given in (4). In (4), parameter 'n' refers to the number of network nodes, and parameter 'w' refers to the data path width. The first term of (4) represents the data wires for connecting the data output port of each node to all the other nodes via channel multiplexers. The second term of (4) refers to the data wires between the data output port of a channel multiplexer and its attached network node.

The equation of calculating the number of data wires in the CDMA network is given in (5). In (5), the meaning of parameters 'n' and 'w' is the same with the parameters in (4). The parameter 's' refers to the bit length of spreading codes. The first term in (5) represents the data wires for connecting data output port of each network node with the input port of 'CDMA Transmitter' block. The number of data wires from the data output port of 'CDMA Transmitter' is represented by the second term in (5). In the CDMA network, each data bit to be transferred will be extended into s bits by the s-bit spreading code. Each bit of the s-bit encoded data is called a data chip. The sum value for n data chips from n network nodes can be represented by log_2n bits. Therefore, the 'CDMA Transmitter' needs to use $s \cdot log_2$n bits to represent all the sum values of s-bit encoded data. Hence, in order to transfer w data bits at one time, we need $w \cdot s \cdot log_2$n data wires as the output of 'CDMA Transmitter' block.

Table 9 lists the number of data wires in the crossbar and CDMA networks under different data path widths. We can see that the crossbar network needs a huge amount of data wires in order to obtain the feature of concurrent data transfer as the CDMA network does. This is a major obstacle to apply the crossbar structure in an on-chip system because the number of network nodes in a future SoC will be very large. Therefore, the CDMA network has the advantage of utilizing less data wires to achieve the feature of concurrency in comparison with the crossbar network.

$$N_{crossbar\_noc} = n \cdot (n-1) \cdot w + n \cdot w = w \cdot n^2 \tag{4}$$

$$N_{CDMA\_noc} = n \cdot w + w \cdot s \cdot log_2 n \tag{5}$$

Table 9. *Number of Data Connection Wires.*

NoC Type		***Number of Data Wires***		
		n=6, s=8	***n=15, s=16***	***n=31, s=32***
Crossbar NoC	**w = 1**	36	225	961
	w = 8	288	1,800	7,688
	w = 16	576	3,600	15,376
	w = 32	1,152	7,200	30,752
CDMA NoC	**w = 1**	30	79	191
	w = 8	240	632	1,528
	w = 16	480	1,264	3,056
	w = 32	960	2,528	6,112

7.3.3 Area Cost of Interconnect Wires

As presented in section 7.3.2, both the CDMA and crossbar networks need a large number of interconnect wires to build the non-blocking data transfer channels among network nodes. The main portion of the interconnect wires among network nodes are the data wires. Therefore, taking the area cost of data wires into account would be helpful for getting more accurate views of the presented NoC designs. The data-wire area estimations of the presented six-node CDMA, crossbar, and the bidirectional ring networks with 32-bit data path width are presented in the following paragraphs.

According to the logic gate area cost of the six-node CDMA network listed in Table 5 and the average gate density, 85K gates/mm^2, of the 0.18μm technology library, we can get that the gate area of a network node of the CDMA network with 32-bit data path width is 0.5mm^2, which is approximately equivalent to a 0.72mm x 0.72mm square area. Similarly, we can get that the gate area of ‘CDMA Transmitter’ block is 0.18mm^2 which is equivalent to a 0.1mm x 1.8mm rectangular area. The gate area of ‘Network Arbiter’ block is 0.011mm^2 which is equivalent to a 0.1mm x 0.11mm rectangle. Therefore, if the six-node CDMA network is placed as the pattern illustrated in Fig.34 (a), we can get an approximate 1.6mm x 2.4mm core area of the design including some overhead and spacing. Because all the network nodes need to be connected to the central located ‘CDMA Transmitter’ and ‘Network Arbiter’ blocks, we can assume the average wire length is half of the core dimension, which is (1.6mm + 2.4mm) / 2 = 2mm. For those global interconnect wires among blocks, the upper metal layers, metal 5 or 6, which have a minimum width and spacing of 0.64μm in the 0.18μm library should be used for the sake of better conductance. Therefore, the equation of estimating the wire area cost is given in (6). In (6), N_{wire} refers to the number data wires in a network. The $L_{average}$ is the average length of the wires. W_{wire} and W_{space} are the minimum width and spacing of the interconnect wires defined by the technology library. Hence, through the equation given in (6) and the number of data wires listed in Table 9, we can get the approximate interconnect wire area cost of the six-node CDMA network with 32-bit data path width is 960 x 2mm x

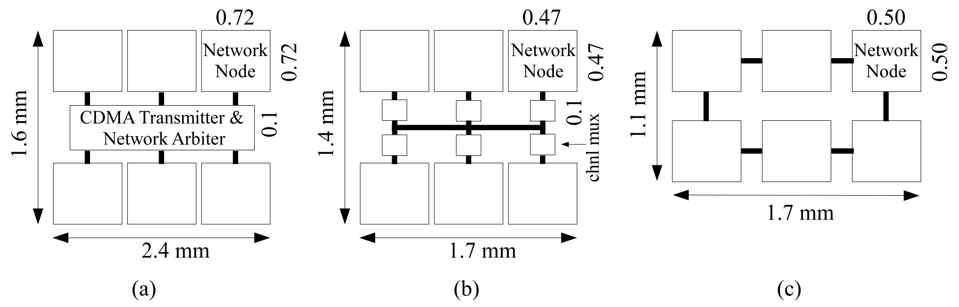


Fig. 34. *Placement of the NoC Designs.*

(0.64μm + 0.64μm) $\approx$ 2.46mm^2 which is 76.6% of the logic gate area of the CDMA network.

For the six-node crossbar network with 32-bit data path width, we can get that the logic gate area of a network node is 0.22mm^2 according to the value listed in Table 7. This area is equivalent to a 0.47mm x 0.47mm square area. Each channel multiplexer block occupies 0.01mm^2 which is equivalent to a 0.1mm x 0.1mm square. Hence, if the six-node crossbar network is placed as the pattern illustrated in Fig.34 (b), we can get an approximate 1.4mm x 1.7mm core area of the design. Similar to the estimation for the CDMA network, we also assume the average wire length is half of the core dimension, which is (1.4mm + 1.7mm) /2 $\approx$ 1.6mm. Therefore, according to the number of data wires listed in Table 9 and (6), we can get the approximate wire area cost of the crossbar network is 1152 x 1.6mm x (0.64μm + 0.64μm) $\approx$ 2.36mm^2 which is 184.4% of the logic gate area of the crossbar network.

$$Wire\ Area = N_{wire} \times L_{average} \times (W_{wire} + W_{space}) \tag{6}$$

In a similar way, we can also get the corresponding area figures of the six-node bidirectional ring network presented in Chapter 4. According to the values listed in Table 2, the logic gate area of a network node in the ring network is 0.25mm^2 which is equivalent to a 0.5mm x 0.5mm square. If the six-node ring network is placed as the pattern illustrated in Fig.34 (c), we can get an approximate 1.1mm x 1.7mm core area of the design. We also take half of the core dimension as the the average wire length, which is (1.1mm + 1.7mm) / 2 = 1.4mm. Because the data connection between two network nodes is bidirectional and the data path width of each direction is 32 bits, the number of data wires in the six-node ring network is (32 x 2) x 6 = 384. Hence, according to (6), we can get the approximate interconnect wire area cost of the ring network is 384 x 1.4mm x (0.64μm + 0.64μm) $\approx$ 0.69mm^2 which is 33.2% of its logic gate area.

Through the presented estimations, we can see that the ring network has the smallest interconnect wire area cost since its smallest core area and number of data wires. In comparison with the crossbar network, the advantage of less number of data wires gained by the CDMA network is degraded in terms of wire area cost because of its larger core area. However, one fact to be noticed is that the difference of number of data wires between the six-node CDMA and crossbar network is very small. Therefore, as the number of network nodes grows, the difference of the number of wires will greatly increase as presented in Table 9. For example, when there are 15 nodes in a network and the data path width is 32 bits, the crossbar network needs 7,200 data wires while the CDMA network only needs 2,528 data wires. With the presented estimation, we can already see that the wire area of the six-node crossbar network is almost 2 times larger than its logic gate area. Hence, as the number of nodes grows, the overhead of wire area cost of the crossbar network will tremendously increase.

7.3.4 Data Transfer Latency

The data transfer latency in the three networks consists of two parts, STL and ATL, as presented in chapters 4, 5 and 6. Because STL values mainly depend on the local clock rates of a functional host, the comparison presented in this subsection mainly concerns the ATL values of the networks.

Because the CDMA and crossbar networks both apply ‘one-hop’ concurrent data transfer scheme, the ATL of these two networks consists of same portions, PLL, PTL, and PSL. The values of ATL in the two networks can be obtained by directly adding the three portions together. However, the ATL of the ring network has different portions and it is a variable depending on the packet traffic route. The ATL portion called PBL of the ring network does not exist in the ATL of the other two networks because the data packets in the CDMA and crossbar networks are transferred directly from the source node to the destination node.

Based on the values listed in Table 4, Table 6, and Table 8, Fig.35, Fig.36, and Fig.37, are plotted to illustrate the ATL of the three networks with different data path widths and packet lengths. The ATL values of the ring network illustrated in the figures are measured in the best case which means that packets are transferred between two adjacent nodes in the ring network. Thus, PBL values of the ring network are zero.

From the figures, we can see that ATL values of the CDMA network are tremendously larger than the values of the crossbar network when the data path width is 1 bit. The difference is getting smaller quickly when the data path width is increased. For example, the ATL value of transferring one-data-cell packet in the crossbar network is around 70% less than the value of the CDMA network when the data path width is 1 bit, whereas this figure is reduced to 41% when the data path width is increased to 32 bits. The large latency in the CDMA network is mainly caused by the data encoding and decoding operations.

In comparison to the ATL values of the ring network realized with 32-bit data path width, the ATL values of the CDMA network are quite close. As illustrated in Fig.35, the ATL value of the CDMA network is even smaller than the best case ATL value of the ring network when

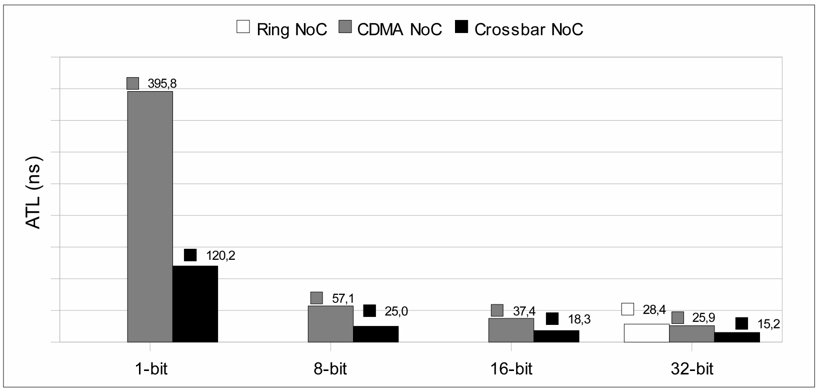


Fig. 35. *ATL Values of Tx 1-data-cell Packet.*

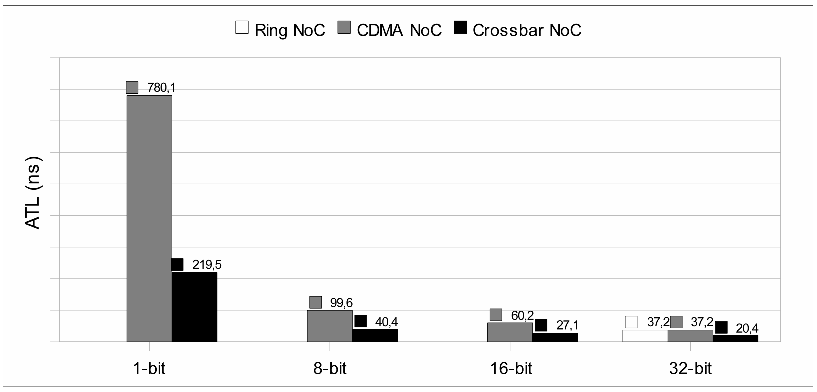


Fig. 36. *ATL Values of Tx 2-data-cell Packet.*

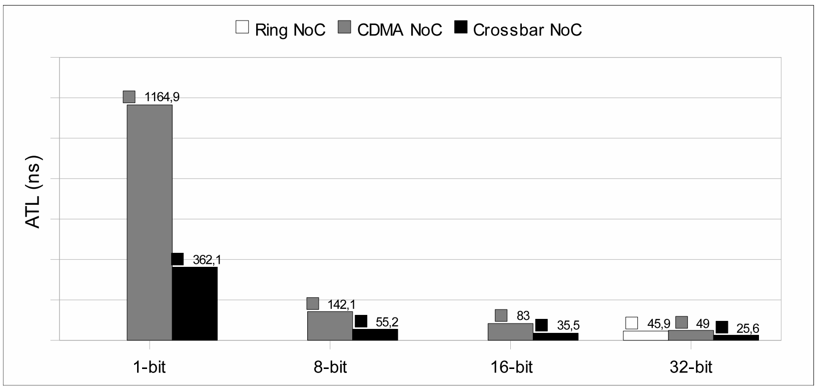


Fig. 37. *ATL Values of Tx 3-data-cell Packet.*

Table 10. *Equivalent Number of Intermediate Nodes in the Ring NoC.*

Packet Length	***1-bit***	***8-bit***	***16-bit***	***32-bit***
1 data cell	N = 15.2	N = 1.2	N = 0.4	N = -0.1
2 data cells	N = 22.6	N = 1.9	N = 0.7	N = 0.0
3 data cells	N = 26.9	N = 2.3	N = 0.9	N = 0.1

the transferred packet has one data cell. In order to compare the data transfer latencies of the ring and CDMA networks clearly, Table 10 lists the equivalent number of intermediate network nodes which would be gone through by a data packet in the ring NoC when the same packet is transferred in the CDMA network under different data path widths. From Table 10, we can see that when the data path width is larger than 8 bits, the ATL value of the CDMA network is already very close to the best-case value of the ring network. Therefore, we can see that the latency caused by the data encoding and decoding scheme in the CDMA network is compensated by its 'one-hop' data transfer capability in comparison with the ring network.

7.3.5 Dynamic Power Consumption

Dynamic power consumption values of the three networks are also estimated during the gate level simulations using the same test stimulus. The measured consumption values are illustrated in Fig.38. From the figure, we can see that the 1-bit CDMA network should not be applied due to the largest power consumption in comparison with other realizations.

The reason of the large power consumption is that it needs much more switching activities than the others because of the over-serialized data transfers. As illustrated in Fig.38, when the data path width is over 8 bits, the power consumption values of the three networks are very close to each other, which means that a similar amount of switching activities happened

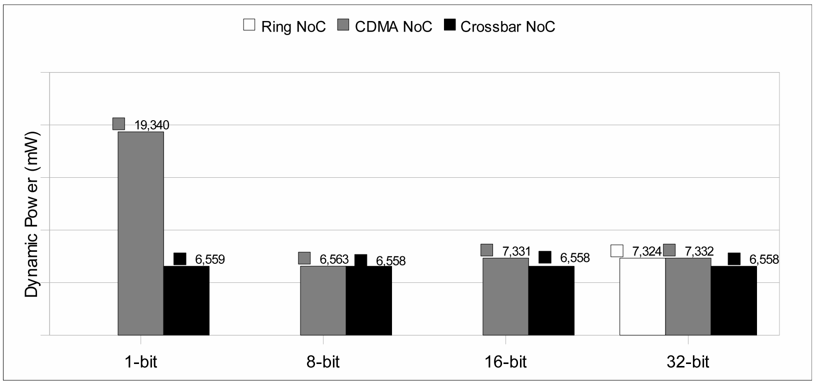


Fig. 38. *Dynamic Power Consumption Comparison.*

Table 11. *Theoretical Throughput of the Six-Node CDMA Network.*

Packet Length	***Throughput under Different Data Path Widths (Gbits/s)***			
	1-bit	**8-bit**	**16-bit**	**32-bit**
1 data cell	0.48	3.36	5.13	7.41
2 data cells	0.49	3.86	6.38	10.32
3 data cells	0.49	4.05	6.94	11.76

in all the networks to perform the same data transfers.

Through the presented comparison work, we can see that the CDMA network can not compete with the other two networks in the aspect of area cost because of its data encoding and decoding scheme. However, when the data path width is equal or larger than 8 bits, the CDMA network has better asynchronous transfer latency performance than the ring network. Although the crossbar network shows smaller logic gate area cost and asynchronous transfer latency, the CDMA network can still be a good alternative of the crossbar structure because it largely reduces the requirement of data connection wires.

7.4 Performance Comparisons with Other NoC Designs

The NoC designs found in literature have different structures and implementations, and their performance estimations are reported with different forms. Therefore, it is difficult to make fair and thorough comparisons with the presented CDMA NoC realization under the same conditions. In this section, quantitative comparisons between the CDMA NoC realization and several other types of NoC designs are presented in order to evaluate the presented CDMA NoC structure in a wider scope.

In order to facilitate comparisons, theoretical throughput values of the six-node CDMA network are listed in Table 11. The values are calculated according to the ATL values listed in Table 6 and the equation given in (7). From (7), we can see that the calculated throughput values exclude the synchronous transfer latency between a network node and its attached functional host. The reason is that these latencies depend on the local clock rate of the host instead of the on-chip network. Hence, the throughput values listed in Table 11 give one type of performance estimations of the GALS CDMA NoC. In this case, the number of nodes used in calculations is 6. The number of data bits depends on the number of data cells in a packet. The size of a data cell is fixed at 32 bits.

$$Throughput = \frac{(Number\ of\ Data\ Bits) \times (Number\ of\ Network\ Nodes)}{ATL\ value} \tag{7}$$

1. Comparing with a Synchronous NoC Design

In the category of synchronous NoC designs, Æthereal NoC developed by Philips research laboratories is a well-known and frequently quoted NoC design. It consists of routers which can provide both guaranteed and best-effort services. According to the performance reported in [23], an Æthereal NoC router which has six bidirectional ports was realized with a 0.13μm technology library. The data path width is 32 bits according to the width of the input queues. This router has an area of 0.175mm^2 after layout and a bandwidth of 16Gbits/s at 500 MHz. This figure is larger than the throughput value of the CDMA NoC with 32-bit data path as listed in Table 11. In this comparison, the factor of different technology libraries for realizations should be noticed.

2. Comparing with a GALS NoC Design

MANGO NoC is a good example of a NoC design which applies both synchronous and asynchronous designs to realize GALS scheme while using routing schemes to share the data links among users in the network. Similar with the Æthereal NoC, a router of MANGO NoC also provides both guaranteed and best-effort services by using virtual channels. A virtual channel design of MANGO router realized with a 0.18μm technology library is reported in [10]. The design has 16-bit data path width and 8 virtual channels. 537Mflits/s throughput of the virtual channel is achieved in typical timing cases. This throughput value equals to 8.59Gbits/s if a flit consists of 16 bits. According to the figures listed in Table 11, this value is close to the 10.32 Gbits/s throughput of the CDMA network when the data path is 32 bits and a packet containing two 32-bit data cells.

3. Comparing with an Asynchronous NoC Design

A NoC design, named CHAIN, which applies pure asynchronous design has been presented in [4]. The CHAIN network applies routers, arbiters, multiplexers, and pipeline channels to set up data links between senders and receivers. By implementing with a 0.18μm technology library, the CHAIN network can achieve a throughput of 1Gbits/s per data link when the data path width in a link is 1 bit. Wider data paths can be built by using parallel data links, although it will incur extra latencies in route set-up and end-to-end transfers as reported in [4]. By comparing of the throughput values of the CDMA network with 1-bit data path width, the CHAIN network can achieve around two times higher throughput.

4. Comparing with Other CDMA Schemes

As addressed in section 5.3 of Chapter 5, the presented GALS CDMA NoC design is not the only one which applies CDMA technique into on-chip communications. A CDMA bus structure which applies analog design to implement CDMA transfers is presented in [88]. The data are encoded and decoded by modulating the voltage signal in the bus. According to the simulation results reported in [88], its throughput is only 70Mbits/s even it can support 60 simultaneous transmissions in the bus. Another analog CDMA bus which has a better performance is presented in [84]. The presented realization which can contain 16 hosts is realized with a 0.35μm technology library and has a throughput of 2.5Gbits/s when the data path width is 15 bits. This throughput figure is smaller than the throughput values of the CDMA NoC even with 8-bit data path width.

Besides analog CDMA buses, another NoC design which also applies CDMA technique is presented in [50, 51]. However, it applies fully synchronous scheme in the network, which means that the CDMA coding and transfer processes are realized with synchronous design instead of the asynchronous design used by the CDMA NoC presented in this thesis. Therefore, it does not address the multi-clock-domain issue in a SoC design. The synchronous CDMA NoC applies a receiver-based spreading code protocol which is not conflict-free as addressed in Chapter 5. The decoding scheme applied in the synchronous CDMA NoC involves subtraction and division operations which make it more complicated than the scheme used in the GALS CDMA NoC presented in this thesis. The synchronous CDMA NoC has been realized with a 0.18μm structured ASIC library which is different from the standard-cell library used in this work. According to the realization results presented in [51] and the explanation presented in [50], the seven-node synchronous CDMA NoC realization running at maximum 94.2 MHz has a throughput of 5.28Gbits/s when the data path width is 32 bits. This figure is less than the throughput values of the GALS CDMA NoC with 32-bit data path as listed in Table 11.

Through the presented comparisons, we can see that the GALS CDMA NoC developed in this work has no apparent advantages of throughput values by comparing with the NoC designs which share data links among network nodes in a time-division manner with all kinds of routing methods. The reason of the lower throughput of the CDMA network is that the data encoding and decoding operations required by CDMA technique incur large latency overhead in data transfer processes. However, this throughput penalty does not obliterate the merit of the non-blocking and ‘one-hop’ data transfer abilities of the CDMA NoC as presented in Chapter 5. By comparing with other on-chip communication structures which also apply CDMA technique, the CDMA NoC presented in this thesis has a better throughput performance by applying the presented GALS scheme and simplified data encoding and decoding schemes.

8. REALIZING A GALS NOC ON AN FPGA DEVICE

The Non-Recurring Engineering (NRE) cost of chip design and manufacture continuously rises as the size of transistors shrinks down. At the 90 nm process node, the total development cost of a single standard-cell chip can be in the range of $20-30 million [17]. Also, the trial and error process of getting an on-chip system design works is also a time consuming process. Therefore, Field-Programmable Gate Array (FPGA) prototyping is widely accepted as a fast and cheap way of verifying an IC design before manufacturing or even using the FPGA in the final product for low production volumes.

FPGA device is a chip containing programmable logic blocks and programmable interconnects. The logic blocks in an FPGA chip can be programmed to perform basic logic operations such as AND, and OR. Therefore, digital circuit designs implemented on an FPGA device can be easily modified in a fast and cheap way. Hence, prototyping a SoC design on an FPGA device is a good way of carrying out design exploration and functional verification before turning to ASIC implementation. Currently, the FPGA devices available on the market are oriented only for realizing synchronous designs by using Look-Up-Table (LUT) structures [14]. Hence, a major challenge in realizing a GALS NoC design on a LUT-based FPGA device is how to realize asynchronous designs. This chapter presents a solution to this challenge and the work of prototyping a bidirectional ring GALS NoC on a LUT-based FPGA device.

8.1 Two Key Components for Realizing Asynchronous Designs on an FPGA Device

As presented in section 3.3.2, the control pipelines of the asynchronous circuits used in the bidirectional ring NoC mainly consist of C-elements. Therefore, realizing the C-element in a LUT-based FPGA is the prerequisite for realizing the asynchronous design. Besides the C-element, an asynchronous arbiter is another important component used in the asynchronous design to allocate the shared resource to only one user at a time. For example, ‘Communication Controller’ block needs an arbiter to decide that either the ‘local packet’ or ‘bypass packet’ will be put into the ‘Tx Packet Buffer’ first if they come to the ‘Packet Distributor’ simultaneously. Thus, the structures of the C-element and arbiter developed for FPGA

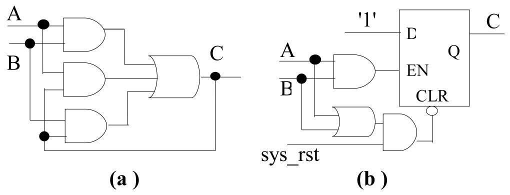


Fig. 39. *C-element Structures.*

realization will be presented in the following subsections.

8.1.1 C-element Structure

In order to map C-element on a LUT-based FPGA, an equivalent two-input C-element structure illustrated in Fig.39(a) has been presented in [36]. It has been proved to be logic hazard-free under the single-bit input change assumption and certain two-input change patterns. The drawback of this structure is that it needs a netlist-format description as a component library in the design flow to ensure that the feedback path is mapped on a LUT correctly. In order to avoid the explicit feedback path, another two-input C-element structure is developed in this work and illustrated in Fig.39(b). It bases on a D-latch which uses ‘A AND B’ as the enable (‘EN’) signal and ‘A OR B’ as the reset signal (‘CLR’). The data input port (‘D’) of the D-latch is tied to logic ‘1’ constantly. The idea of using latch to map a C-element in LUT has already been presented in [36] where a RS-latch is suggested. Whereas, the C-element structure based on D-latch in Fig.39(b) is safer than the suggested RS-latch structure because it avoids data switching at the data input port ‘D’.

8.1.2 Arbiter Structure

Cross-coupled NAND gates are normally used as the simplest arbiter structure in an ASIC implementation. For implementing an arbiter on an FPGA device, the built-in Flip-Flop is suggested to be used in order to minimize metastability effects [65]. Therefore, an arbiter structure which bases on the built-in Flip-Flop of an FPGA device and applies the cross-coupled NAND structure is developed for realizing the asynchronous design of the GALS NoC. The developed arbiter structure is illustrated in Fig.40. The arbiter is a two-input fixed-priority arbiter and can be divided into three stages.

The first stage consists of two cross-coupled AND gates, ‘A1’ and ‘A2’, with inverted inputs. The gate ‘A0’ is used to disable the input ‘r2’ when a conflict between ‘r1’ and ‘r2’ is detected at the output of C-element ‘C3’. If the two input requests ‘r1’ and ‘r2’ appear simultaneously or very close to each other, the LUT implementation of ‘Stage1’ will enter into an oscillation

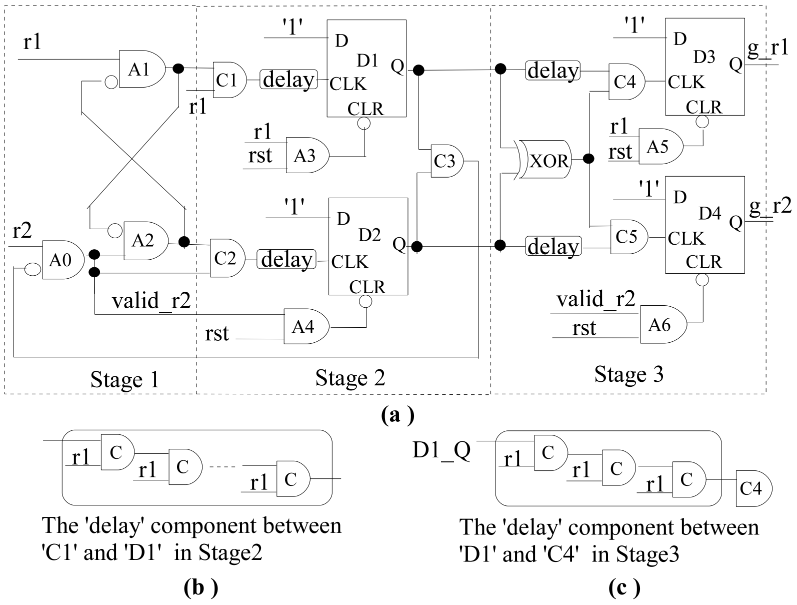


Fig. 40. *The Arbiter Structure for FPGA Realization.*

state instead of the metastability state as the ASIC implementation. In this situation, the second stage of the arbiter is used to filter out the possible oscillating outputs from ‘Stage1’.

The ‘Stage2’ illustrated in Fig.40(a) bases on two built-in D-FF registers of an FPGA device. The C-elements, ‘C1’ and ‘C2’, are used to convert the oscillation outputs from ‘Stage1’ into a single 0→1 signal transition which is used as the clock signal to trigger the registers ‘D1’ and ‘D2’ respectively. After passing through ‘Stage2’ of the arbiter, the oscillation outputs from ‘Stage1’ may trigger the outputs of both ‘D1’ and ‘D2’ into logic ‘1’. In this case, the ‘C3’ will detect this conflict and disable the ‘r2’ request by the feedback path from the output of ‘C3’ to the input of ‘A0’. The ‘delay’ components in ‘Stage2’ which consist of a C-element chain as illustrated in Fig.40(b) are used to ensure that the rising edge from the outputs of ‘C1’ and ‘C2’ will arrive after the ‘CLR’ signals from ‘A3’ and ‘A4’.

The actual arbitration process takes place in the ‘Stage 3’ where another two built-in D-FF registers are used. When a request conflict is detected at the outputs of ‘D1’ and ‘D2’, the ‘XOR’ logic in ‘Stage3’ will close the arbiter output by disabling ‘C4’ and ‘C5’. The arbitration outputs will be enabled only after the output of ‘D2’ is cleared by the feedback from ‘C3’. Therefore, request ‘r1’ has a higher priority in the presented arbiter. The ‘delay’ components in ‘Stage3’ are used to filter out the possible glitches from ‘XOR’ when the output signals of ‘D1’ and ‘D2’ did not reach the inputs of the ‘XOR’ gate simultaneously in a request-conflict situation.

8.2 Realizing a Four-Node GALS Ring NoC

By applying the presented C-element and arbiter structures, the barriers of realizing the asynchronous designs on a LUT-based FPGA device are removed. Therefore, both the asynchronous and synchronous designs of the GALS bidirectional ring NoC can be fed into the same design flow for realizing synchronous designs. The design tool and the FPGA device used in this work are QuartusII and Altera StratixII respectively. The realizing method used in this work is summarized as the following four steps.

Step1: Describe both synchronous and asynchronous designs in a hierarchical manner by using VHDL.

Namely, the C-element structure illustrated in Fig.39(b) is modeled using VHDL as a component. Then, any other blocks, such as the arbiter or the control pipelines, use the VHDL model of the C-element as component instances in their own VHDL description. In the same manner, the control pipelines, arbiter, and C-element are used by a higher level block as component instances in their VHDL descriptions.

Step2: Define a design partition for each component.

In order to prevent the synthesis tool to mix all the combinational logic from different components together, each instance of the C-element and arbiter in the design is set as a design partition by using QuartusII. The higher level components or blocks are also to be set as separate partitions according to the design hierarchy. During the synthesis process, each partition of the design will be synthesized separately from each other by QuartusII. Therefore, the presented C-element and arbiter structures will be generated correctly.

Step3: Set a LogicLock region for all delay sensitive arbiter and control blocks.

In order to meet the QDI timing requirements of the presented arbiter and block control pipeline structures, LogicLock technique [1] provided by QuartusII is applied during the placement and routing process. A LogicLock region is set to each arbiter and block control logic pipeline in the design so that the components in the arbiter and the control pipeline will be automatically placed into one Logic Array Block (LAB) [2] or the adjacent LABs by QuartusII. Thus, the fast intra-connects inside a LAB and inter-connects [2] between adjacent LABs can meet the loose timing requirements of the arbiter and the block control pipeline as addressed in section 3.3.2.

Step4: Run synthesis, placement, and routing steps without additional constraint files.

After the design partitions and LogicLock regions are set by using QuartusII, both the synchronous and asynchronous designs of the GALS NoC can be realized on an FPGA device without any other constraints.

By using the presented realization method, a four-node GALS bidirectional ring NoC is re-

Table 12. *ALUTs Utilization of 'Network Node' Blocks.*

Block Name		***Utilized ALUTs***
Node IF (BVCI Slave Type)		146
Node IF (BVCI Master Type)		399
Layer MUX		142
Communication Layer (CL block)	**Communication Controller**	238
	Packet Distributor	451
	Packet Sender +Tx Packet Buffer	2,202
	Packet Receiver	233
	Rx Packet Buffer	1,878
Total (with 2 CL blocks)	**BVCI Slave Type Network Node**	10,292
	BVCI Master Type Network Node	10,545

alized on a StratixII EP2S60 device. The whole network utilizes 41,674 Adaptive LUTs (ALUTs) [2] which is 86.2% of the ALUTs on the device. The area costs of 'Network Node' block and their sub-blocks in terms of utilized ALUTs are listed in Table 12. The reason of the large area cost is that each LogicLock region for a basic component or a delay sensitive block, such as C-element and arbiter, exclusively occupies a square area even the block can not fully utilize the LUT resources in that area. Therefore, when all LogicLock regions for basic components and delay sensitive blocks in the design pile together, they occupy a large area on the FPGA device.

Through this prototyping practice of a GALS NoC design, we can see that it is possible to realize a synchronous-asynchronous mixed design on an FPGA device aimed for synchronous designs. However, extra care of placing the asynchronous components is needed and the utilization of the resources on an FPGA device is not efficient due to the compensation caused by meeting the timing requirements of asynchronous designs. Therefore, an FPGA device which includes the basic and frequently used asynchronous components, such as C-element, would be very helpful for the FPGA prototyping work of a GALS NoC design.

9. CONCLUSIONS

This chapter concludes this thesis by summarizing the author's publications included in Part II, and then describing the main results of the presented work. Finally, directions of future research are proposed.

9.1 Summary of Publications

Publication [P1]: *A Synthesizable RTL Design of Asynchronous FIFO.* This publication presents the work of developing an asynchronous FIFO. This FIFO is modeled in RTL using VHDL and suits the commonly used synchronous design tools and flow. The motivation of this work is to develop an asynchronous FIFO as a reusable RTL IP block for the following work of designing GALS NoC. The overall structure and an ad hoc control logic of the asynchronous FIFO are presented. The presented asynchronous FIFO is synthesized using a synchronous design tool and it passes the functional verification in gate-level.

Publication [P2]: *Asynchronous Network Node Design for Network-on-Chip.* In order to apply GALS scheme in Proteo NoC architecture, a network node which includes both synchronous and asynchronous designs is developed and presented in this publication. The synchronous design is applied in the interface block to carry out locally synchronous communications with the attached functional host, while the asynchronous design is used in the blocks for performing globally asynchronous data transfers. The presented network node is realized as a synthesizable IP block in RTL using VHDL. A six-node bidirectional ring on-chip network composed of the presented network node design is built for the simulation purpose. The way of applying GALS scheme in Proteo NoC architecture is firstly presented in this publication and adopted by the other NoC structures developed later on.

Publication [P3]: *An On-Chip CDMA Communication Network.* This is the first publication of the author to present the idea of applying CDMA technique into an on-chip network. The issues and related methods of applying CDMA technique in a NoC design, including data encoding and decoding, spreading code selection, and spreading code protocol, are presented in this publication. The structure of the blocks for building the CDMA NoC is also presented. The GALS scheme is applied in the CDMA network by utilizing both synchronous and asynchronous designs. A six-node CDMA NoC is built and synthesized. The preliminary performance estimations of the area cost and data transfer latency of the six-node network are presented.

Publication [P4]: *Prototyping A Globally Asynchronous Locally Synchronous Network-on-Chip on a Conventional FPGA Device Using Synchronous Design Tools.* This publication presents the work of prototyping a four-node GALS bidirectional ring NoC design on a LUT-based FPGA device. The issues and related solutions of realizing a synchronous-asynchronous mixed NoC design on an FPGA device aimed for synchronous designs are presented. The structures of key components, C-element, arbiter, and the structures of two control pipelines are presented. The presented structures suit for the realizations on a LUT-based FPGA device. A method of prototyping a GALS NoC design on an Altera FPGA device using synchronous design tools is also presented in this publication.

Publication [P5]: *A RTL Asynchronous FIFO Design Using Modified Micropipeline.* This is the second publication of the author to present a general purpose asynchronous FIFO design. In this publication, the preliminary FIFO design presented in [P1] is improved largely by replacing the ad hoc control logic design with the newly developed control pipelines. The presented control pipelines are based on the control logic of Micropipeline and they are more robust than the control logic presented in [P1] in terms of delay sensitivity. An arbiter structure and a C-element structure which suit for RTL modeling are also presented. A synchronous FIFO design is also developed as a reference to evaluate the performance of the presented asynchronous FIFO. The area cost and power consumption of the synchronous and asynchronous FIFO designs are estimated and compared according to the gate-level realizations. The data transfer latency values extracted from gate-level simulations of the asynchronous FIFO are also presented.

Publication [P6]: *Comparison of a Ring On-Chip Network and a Code-Division Multiple-Access On-Chip Network.* The main purpose of this work is to thoroughly compare the GALS ring NoC design presented in [P2] with the GALS CDMA NoC design presented in [P3] in order to examine their different characteristics. The aspects of the two NoC designs examined and compared in this publication include network structures, data transfer principles, network node structures, and their asynchronous designs. Based on the gate-level realizations of two six-node networks, the performance of the two networks, including area costs and data transfer latencies, are also compared. At the end of this publication, a preliminary work of SystemC modeling is briefly introduced.

Publication [P7]: *Comparing Two Non-Blocking Concurrent Data Switching Schemes for Network-on-Chip.* Although the CDMA NoC presented in [P3] achieves the feature of non-blocking data transfers by using CDMA technique, it incurs a large overhead caused by introducing data encoding and decoding operations. Another type of data switching scheme called crossbar has also the feature of non-blocking data transfers. Therefore, the overhead of applying CDMA technique in an on-chip network is examined by comparing a CDMA network with a crossbar network in the same network environment. The characteristics of the CDMA and the crossbar networks are further examined by comparing the two networks under different data path widths. Based on the synthesis results, area costs, power consumption, data transfer latencies, and the numbers of data wires of the two networks are also compared.

Publication [P8]: *Applying CDMA Technique to Network-on-Chip.* This publication thoroughly presents the work of developing the CDMA NoC. The issues of applying CDMA technique in an on-chip network and the design of realizing a GALS CDMA NoC are addressed with details and examples. The realizations of a six-node GALS CDMA network with different data path widths are presented and compared with the six-node GALS bidirectional ring NoC presented in [P6]. By the comparisons, the area cost and power consumption overhead caused by applying CDMA technique in an on-chip network are further examined. The effect of different data path widths on data transfer latency performance in the CDMA NoC is also examined. In comparison to the bidirectional ring NoC, the optimal configuration of the CDMA NoC is clarified.

Publication [P9]: *Modeling A Code-Division Multiple-Access Network-on-Chip Using SystemC.* This publication presents a SystemC modeling and simulation work based on the preliminary SystemC modeling work introduced in [P6]. A SystemC model of the presented CDMA NoC design is developed in order to facilitate the design exploration of the CDMA NoC with different configurations in a fast and flexible way. The presented SystemC model uses transaction level modeling approach to model the asynchronous handshake processes of data transfers in the CDMA NoC. By utilizing the RTL realizations of the CDMA NoC presented in [P8], a performance estimation method which bases on timing back-annotation is presented to estimate the CDMA NoC performance under different configurations. Finally, the performance estimation results of the CDMA NoC with different channel widths, different number of network nodes, and different traffic patterns are presented.

9.2 The Main Results

The work presented in this thesis concentrates on designing and realizing GALS on-chip networks. The main results are summarized in the following six aspects.

1. Developed asynchronous designs for GALS NoCs

An asynchronous FIFO structure was developed. Although the asynchronous FIFO design is meant to be used in the GALS NoC designs, it is a general purpose asynchronous FIFO IP module for any other applications. In comparison to a synchronous FIFO reference, the asynchronous FIFO can save 48.5% logic gate area cost and 45.8% dynamic power consumption. Besides the asynchronous FIFO design, two control pipelines were developed as the control logic for the asynchronous designs of the GALS NoCs. All the developed asynchronous designs in this work suit to be modeled in RTL by using HDL. This feature facilitates the asynchronous designs to be realized together with the synchronous designs in a GALS NoC by using the commonly used synchronous design tools and design flow.

2. Applying GALS scheme into Proteo NoC architecture

A network node which applies both synchronous and asynchronous designs was developed as an IP module to realize the GALS scheme in the Proteo NoC. A six-node bidirectional ring network was set up for evaluating the network node design. The buffers for sending and receiving data packets occupy around 33% of the network node area in the bidirectional ring network. According to a standard-cell realization of the six-node network, the network node can deliver a 64-bit packet to an adjacent node in 28.4*ns* through a four-phase dual-rail handshake protocol.

3. Developed a GALS CDMA NoC structure

An on-chip GALS NoC which applies CDMA technique was developed. The main benefit of applying CDMA technique in on-chip communications is the feature of non-blocking and ‘one-hop’ data transfers. This feature is very useful for providing small-variance data transfer latency in an on-chip network. A six-node GALS CDMA network was built and realized in gate-level in order to examine its performance. By comparing the six-node CDMA network with a six-node ring network, the CDMA network has 54.1% larger logic gate area cost when they both apply 32-bit data path width. The larger area cost in the CDMA network is mainly caused by the parallel data encoding and decoding logic. However, the reward for this area overhead is that the asynchronous data transfer latency in the 32-bit CDMA network equals to the best case latency in the ring network. By comparing the performance of the CDMA network under different data path widths, the CDMA network has a good balance between area cost and data transfer latency when its data path width is 8 bits or 16 bits. If an application is sensitive to area cost and power consumption, the CDMA network with 8-bit data path is a good option to replace the bidirectional ring network. If small data transfer latency is required in the application, the 16-bit CDMA network is a better choice. In comparison with other NoC designs which share data links among network nodes in a time-division manner, the CDMA NoC has no apparent advantages of throughput because of the latency overhead incurred by the data encoding and decoding processes. In comparison with analog CDMA buses and a synchronous CDMA NoC design, the CDMA NoC presented in this work can achieve better throughput performance by using the GALS scheme and simplified data encoding and decoding schemes. Through an estimation of layout with a 0.18μm technology, a six-node CDMA network with 32-bit data path width has a 3.21mm^2 logic gate area and a 2.46mm^2 data wire area, and its possible core area on a wafer would be around 1.6*mm* x 2.4*mm*.

4. Developed a GALS Crossbar Network

In order to examine the overhead of the data encoding and decoding operations in the CDMA network, another six-node non-blocking network which uses a crossbar switch was developed for comparisons. The crossbar network applies multiplexer structure to realize the crossbar switch in the network. By comparing the crossbar network with the CDMA network, the data encoding and decoding logic incurs 39.4% larger logic gate area cost in the CDMA network when the data path width in both networks is set to 8 bits. In comparison to the crossbar network, the advantage of the CDMA network is that the number of data connection wires is much smaller. For example, when the number of nodes is 31, the crossbar network requires 30,752 connection wires to achieve concurrent transfers if the data path is 32 bits, whereas, this number is reduced to 6,112 in the CDMA network.

5. Realized a GALS NoC design on an FPGA device

A four-node GALS bidirectional ring network was realized on a LUT-based FPGA device. This prototype work exhibits a way of realizing a synchronous-asynchronous mixed design on an FPGA device aimed for synchronous designs. A C-element and an arbiter structure which suit for LUT-based FPGA devices were developed. A method of realizing asynchronous designs on an Altera FPGA device was presented. The realized four-node bidirectional ring network takes 41,674 ALUTs which is 86.2% of the ALUTs on a StratixII EP2S60 FPGA device. The drawback of the presented realization method is that lots of unused LUT resources in each LogicLock region can not be utilized in order to meet the delay requirements of each asynchronous block. This situation can be improved in the future if an FPGA device containing the basic components of asynchronous designs, such as the C-element, will become available.

6. SystemC modeling for design exploration and performance estimation

A SystemC model of the CDMA network was built in transaction level for design exploration and performance estimation in an early design stage. Based on back-annotating the delay information of a synthesized CDMA network, a method was developed to estimate asynchronous data transfer latencies under different configurations by using the SystemC model. Different data path widths, different number of network nodes, and different traffic patterns of the CDMA network were experimented by using the SystemC model. The simulation results show that the asynchronous data transfer latency in the CDMA network increases as the number of network nodes increases. The transfer latency also increases when the traffic load in the CDMA network increases. The developed SystemC model and the performance estimation method are important for integrating the CDMA NoC into a SoC design exploration process in transaction level.

The GALS NoC designs developed in this work present a few preliminary solutions to address the on-chip communication issue of on-chip systems. When conventional bus structures can not meet the on-chip communication requirements pushed up by the growing on-chip systems, the GALS NoC structures proposed in this work can be considered as possible alternative choices depending on the application requirements. Although the presented GALS NoC designs are currently realized with a 0.18μm standard-cell technology library, they can be easily adapted to other technologies because the developed GALS NoC designs are modeled in RTL using VHDL.

9.3 Future Research Directions

Since several disadvantages of the CDMA NoC have been discovered during this work, one direction of future research can focus on the following aspects.

(1) Develop a distributed data encoding and decoding scheme in order to make the CDMA NoC structure more scalable.

(2) Find more efficient data encoding or decoding method to reduce the data transfer latency and minimize area cost and power consumption.

(3) Prototyping a larger CDMA NoC design on an FPGA device to evaluate its performance.

(4) Comparison of the CDMA NoC to more examples of other NoC topologies.

Another direction of future work is to improve the presented asynchronous designs with other methodologies so that the performance can be improved in terms of delay, area cost, and power consumption.

BIBLIOGRAPHY

[1] *Quartus II Version 5.1 Handbook*, Altera, December 2005, Volume 2, pages 1009-1049.

[2] *Stratix II Device Handbook*, Altera, December 2005, Volume1, pages 23-47.

[3] *AMBA Specification Rev2.0*, ARM, http://www.arm.com/, 1999.

[4] J. Bainbridge and S. Furber, "Chain: a delay-insensitive chip area interconnect," *IEEE Micro*, Volume 22, Issue 5, pages 16–23, September-October 2002.

[5] R. H. Bell, Jr., K. Y. Chang, L. John, and E. E. Swartzlander, Jr., "CDMA as a multiprocessor interconnect strategy," in *Conference Record of the Thirty-Fifth Asilomar Conference on Signals, Systems and Computers*, Volume 2, November 2001, pages 1246–1250.

[6] L. Benini and G. D. Micheli, "Networks on chip:a new paradigm for systems on chip design," in *Proceedings of the Conference and Exhibition on Design, Automation and Test in Europe 2002*, March 2002, pages 418–419.

[7] D. Bertozzi and L. Benini, "Xpipes: a network-on-chip architecture for gigascale systems-on-chip," *IEEE Circuits and Systems Magazine*, Volume 4, Issue 2, pages 18–31, Second Quarter 2004.

[8] T. Bjerregaard, "The MANGO clockless network-on-chip: Concepts and implementation," Ph.D. dissertation, Technical University of Denmark, 2005.

[9] T. Bjerregaard, S. Mahadevan, R. G. Olsen, and J. Sparso, "An OCP compliant network adapter for GALS-based SoC design using the MANGO network-on-chip," in *Proceedings of 2005 International Symposium on System-on-Chip*, November 2005, pages 171–174.

[10] T. Bjerregaard and J. Sparso, "Virtual channel designs for guaranteeing bandwidth in asynchronous network-on-chip," in *Proceedings of Norchip Conference 2004*, November 2004, pages 269–272.

[11] ——, "A router architecture for connection-oriented service guarantees in the MANGO clockless network-on-chip," in *Proceedings of the Conference and Exhibition on Design, Automation and Test in Europe 2005*, Volume 2, March 2005, pages 1226–1231.

[12] D. S. Bormann and P. Y. K. Cheung, “Asynchronous wrapper for heterogeneous systems,” in *Proceedings of the 1997 International Conference on Computer Design: VLSI in Computers and Processors*, October 1997, pages 307–314.

[13] H. C. Brearley, “ILLIAC II: A short description and annotated bibliography,” *IEEE Transactions on Computers*, Volume EC-14, Issue 3, pages 399–403, June 1965.

[14] S. Brown and J. Rose, “Architecture of FPGAs and CPLDs: A tutorial,” *IEEE Design and Test of Computers*, Volume 13, No. 2, pages 42–57, Summer 1996.

[15] D. Chapiro, “Globally-asynchronous locally-synchronous systems,” Ph.D. dissertation, Stanford University, 1984, report No. STANCS- 84-1026.

[16] A. A. Chien and J. H. Kim, “Planar-adaptive routing: Low-cost adaptive networks for multiprocessors,” in *Proceedings of the* 19^{th} *Annual International Symposium on Computer Architecture*, May 1992, pages 268–277.

[17] E. Clarke, “FPGAs and structured ASICs: Low-risk SoC for the masses,” in *Design and Reuse Industry Articles*, August 2005. URL: http://www.us.design-reuse.com/articles/article12360.html

[18] W. Dally and C. Seitz, “Deadlock-free message routing in multiprocessor interconnection networks,” *IEEE Transactions on Computers*, Volume C-36, Issue 5, pages 547–553, May 1987.

[19] W. J. Dally and B. Towles, “Route packets, not wires: on-chip interconnection networks,” in *Proceedings of the* 38^{th} *Conference on Design Automation*, June 2001, pages 684–689.

[20] ——, *Principles and Practices of Interconnection Networks*. Elsevier, Inc., December 2003, pages 18, 228, and 234–239.

[21] A. L. Davis, “The architecture and system method of ddm-1: A recursively structured data driven machine,” in *Proceedings of* 5^{th} *IEEE/ACM Annual International Symposium on Computer Architecture*, April 1978, pages 210–215.

[22] A. V. de Mello, L. C. OST, F. G. Moraes, and N. L. V. Calazans, “Evaluation of routing algorithms on mesh based NoCs,” Faculdade de Informatica PUCRS, Brazil, Technical Report No.040, May 2004.

[23] J. Dielissen, A. Radulescu, K. Goossens, and E. Rijpkema, “Concepts and implementation of the Philips network-on-chip,” in *Proceedings of IP/SOC Conference for IP-based SoC Design 2003*, November 2003.

[24] C. Dike and E. Burton, “Miller and noise effects in a synchronizing flip-flop,” *IEEE Journal of Solid-State Circuits*, Volume 34, Issue 6, pages 849–855, June 1999.

[25] E. H. Dinan and B. Jabbari, "Spreading codes for direct sequence CDMA and wideband CDMA cellular networks," *IEEE Communications Magazine*, Volume 36, Issue 9, pages 48–54, September 1998.

[26] D.J.Kinniment and A.V.Yakovlev, "Low latency synchronization through speculation," in *Proceedings of* 14th *International Workshop on Integrated Circuit and System Design: Power and Timing Modeling, Optimization and Simulation*, September 2004.

[27] I. Dobbelaere, M. Horowitz, and A. E. Gamal, "Regenerative feedback repeaters for programmable interconnections," *IEEE Journal of Solid-State Circuits*, Volume 30, Issue 11, pages 1246–1253, November 1995.

[28] T. Felicijan and S. B. Furber, "An asynchronous on-chip network router with quality-of-service (QoS) support," in *Proceedings of IEEE International SOC Conference*, September 2004, pages 274–277.

[29] S. B. Furber, J. D. Garside, P. Riocreux, S. Temple, P. Day, J. Liu, and N. C. Paver, "AMULET2e: an asynchronous embedded controller," *in Proceedings of the IEEE*, Volume 87, Issue 2, pages 243–256, February 1999.

[30] J. D. Garside, W. J. Bainbridge, A. Bardsley, D. M. Clark, D. A. Edwards, S. B. Furber, J. Liu, D. W. Lloyd, S. Mohammadi, J. S. Pepper, O. Petlin, S. Temple, and J. V. Woods, "AMULET3i - an asynchronous system-on-chip," in *Proceedings of the 2000 International Symposium on Advanced Research in Asynchronous Circuits and Systems*. IEEE Computer Society Press, April 2000, pages 162–175.

[31] R. Ginosar, "Fourteen ways to fool your synchronizer," in *Proceedings of the Ninth International Symposium on Asynchronous Circuits and Systems*, May 2003, pages 89–96.

[32] T. Grötker, S. Liao, G. Martin, and S. Swan, *System Design with SystemC*. Kluwer Academic Publishers, 2002, pages 131–153.

[33] M. Gschwind, H. P. Hofstee, B. Flachs, M. Hopkins, Y. Watanabe, and T. Yamazaki, "Synergistic processing in cell's multicore architecture," *IEEE Micro*, Volume 26, Issue 2, pages 10–24, March-April 2006.

[34] P. Guerrier and A. Greiner, "A generic architecture for on-chip packet-switched interconnections," in *Proceedings of the Conference and Exhibition on Design, Automation and Test in Europe 2000*, March 2000, pages 250–256.

[35] M. W. Heath, W. P. Burleson, and I. G. Harris, "Synchro-tokens: eliminating nondeterminism to enable chip-level test of globally-asynchronous SoC's," in *Proceedings of the Conference and Exhibition on Design, Automation and Test in Europe 2004,* Volume 1, February 2004, pages 410–415.

[36] Q. T. Ho, J. B. Rigaud, L. Fesquet, M. Renaudin, and R. Rolland, “Implementing asynchronous circuits on LUT based FPGAs,” in *Proceedings of* 12th *International Conference on Field-Programmable Logic and Applications*, September 2002, pages 36–46.

[37] *Silistix*, http://www.silistix.com, 2005.

[38] D. A. Huffman, “The synthesis of sequential switching circuits,” *Journal of the Franklin Institute*, Volume 257, No. 3 and 4, March-April 1954.

[39] *CoreConnect Bus Architecture*, IBM, http://www.chips.ibm.com/products/coreconnect, 1999.

[40] *Intel High-Performance Consumer Desktop Microprocessor Timeline*, Intel, July 2006. URL: http://www.intel.com/pressroom/kits/core2duo/pdf/microprocessor_timeline.pdf

[41] *Open Systems Interconnection - Basic Reference Model*, ISO/IEC 7498-1, 1994.

[42] *International Technology Roadmap for Semiconductors 2007 Edition-Executive Summary*, ITRS, 2007, http://www.itrs.net/.

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

[44] J.Zhou, D.J.Kinniment, G. Russell, and A. Yakovlev, “A robust synchronizer circuit,” in *Proceedings of the 2006 Computer Society Annual Symposium on VLSI*, March 2006, pages 442–443.

[45] F. Karim, A. Nguyen, and S. Dey, “An interconnect architecture for networking systems on chips,” *IEEE Micro*, Volume 22, Issue 5, pages 36–45, September-October 2002.

[46] H. Kariniemi and J. Nurmi, “Reusable XGFT interconnect IP for network-on-chip implementations,” in *Proceedings of the International Symposium on System-on-Chip 2004*, November 2004, pages 95–102.

[47] T. B. Keat, R. Yoshimura, T. Matsuoka, and K. Taniguchi, “A novel dynamically programmable arithmetic array using code division multiple access bus,” in *Proceedings of the* 8th *International Conference on Electronics, Circuits and Systems*, Volume 2, September 2001, pages 913–916.

[48] J. Kessels, T. Kramer, G. den Besten, A. Peeters, and V. Timm, “Applying asynchronous circuits in contactless smart cards,” in *Proceedings of Sixth International Symposium on Advanced Research in Asynchronous Circuits and Systems*, April 2000, pages 36–44.

[49] K. Keutzer, A. R. Newton, J. M. Rabaey, and A. Sangiovanni-Vincentelli, “System-level design: orthogonalization of concerns and platform-based design,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Volume 19, Issue 12, pages 1523–1543, December 2000.

[50] D. Kim, M. Kim, and G. E. Sobelman, “CDMA-based network-on-chip architecture,” in *Proceedings of the 2004 IEEE Asia-Pacific Conference on Circuits and Systems*, Volume 1, December 2004, pages 137–140.

[51] ——, “Design of a high-performance scalable CDMA router for on-chip switched networks,” in *Proceedings of the International SoC Design Conference 2005*, October 2005, pages 32–35.

[52] S. Kim, J. Lee, and K. Kim, “A parallel flop synchronizer for bridging asynchronous clock domains,” in *Proceedings of 2004 Asia-Pacific Conference on Advanced System Integrated Circuits*, August 2004, pages 184–187.

[53] D. Kinniment, A. Bystrov, and A.Yakovlev, “Synchronization circuit performance,” *IEEE Journal of Solid-State Circuits*, Volume 37, No. 2, pages 202–209, February 2002.

[54] S. Kumar, A. Jantsch, J. P. Soininen, M. Forsell, M. Millberg, J. Oberg, A. K. Tiensyrj, and A. Hemani, “A network-on-chip architecture and design methodology,” in *Proceedings of the IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2002, pages 117–124.

[55] J. A. J. Leijten, J. L. van Meerbergen, A. H. Timmer, and J. A. G. Jess, “Stream communication between real-time tasks in a high-performance multiprocessor,” in *Proceedings of the Conference and Exhibition on Design, Automation and Test in Europe 1998*, March 1998, pages 125–131.

[56] D. D. Lin and T. J. Lim, “Subspace-based active user identification for a collision-free slotted ad hoc network,” *IEEE Transactions on Communications*, Volume 52, Issue 4, pages 612–621, April 2004.

[57] A. Lines, “Nexus: an asynchronous crossbar interconnect for synchronous system-on-chip designs,” in *Proceedings of the* 11th *IEEE Symposium on High Performance Interconnects*, August 2003, pages 2–9.

[58] D. Mangano, G. Falconeri, C. Pistritto, and A. Scandurra, “Effective full-duplex mesochronous link architecture for network-on-chip data-link layer,” in *Proceedings of* 10th *Euromicro Conference on Digital System Design Architectures, Methods and Tools*, August 2007, pages 519–526.

[59] D. Mangano, R. Locatelli, A. Scandurra, C. Pistritto, M. Coppola, L. Fanucci, F. Vitullo, and D. Zandri, “Skew insensitive physical links for network on chip,” in *Proceedings of* 1st *International Conference on Nano-Networks and Workshops 2006*, September 2006, pages 1–5.

[60] T. Marescaux, A. Bartic, D. Verkest, S. Vernalde, and R. Lauwereins, “Interconnection networks enable fine-grain dynamic multi-tasking on FPGAs,” in *Proceedings of* 12th *International Conference on Field-Programmable Logic and Applications*, September 2002, pages 795–805.

[61] A. J. Martin, S. M. Burns, T. K. Lee, D. Borkovic, and P. J. Hazewindus, “The design of an asynchronous microprocessor,” *ACM SIGARCH Computer Architecture News*, Volume 17, Issue 4, pages 99–110, June 1989.

[62] T. Meincke, A. Hemani, S. Kumar, P. Ellervee, J. Oberg, T. Olsson, P. Nilsson, D. Lindqvist, and H. Tenhunen, “Globally asynchronous locally synchronous architecture for large high-performance ASICs,” in *Proceedings of the 1999 IEEE International Symposium on Circuits and Systems*, Volume 2, May 1999, pages 512–515.

[63] J. Mekie, S. Chakraborty, and D. K. Sharma, “Evaluation of pausible clocking for interfacing high speed IP cores in GALS framework,” in *Proceedings of the* 17th *International Conference on VLSI Design*, January 2004, pages 559–564.

[64] G. Moore, “Cramming more components onto integrated circuits,” *Electronics*, pages 114–117, April 1965.

[65] S. W. Moore and P. Robinson, “Rapid prototyping of self-timed circuits,” in *Proceedings of International Conference on Computer Design 1998*, October 1998, pages 360–365.

[66] F. Moraes, N. Calazans, A. Mello, L. Möller, and L. Ost, “HERMES: An infrastructure for low area overhead packet-switching networks on chip,” *Integration, the VLSI Journal*, Volume 38, Issue 1, pages 69–93, October 2004.

[67] D. E. Muller and W. S. Bartky, “A theory of asynchronous circuits,” in *Proceedings of International Symposium on the Theory of Switching*. Harvard University Press, April 1959, pages 204–243.

[68] J. Muttersbach, T. Villiger, and W. Fichtner, “Practical design of globally-asynchronous locally-synchronous systems,” in *Proceedings of Sixth International Symposium on Advanced Research in Asynchronous Circuits and Systems*, April 2000, pages 52–59.

[69] J. Muttersbach, T. Villiger, H. Kaeslin, N. Felber, and W. Fichtner, “Globally-asynchronous locally-synchronous architectures to simplify the design of on-chip systems,” in *Proceedings of the* 12th *IEEE International ASIC/SOC Conference*, September 1999, pages 317–321.

[70] C. J. Myers, *Asynchronous Circuit Design*. John Wiley & Sons, Inc., 2001.

[71] A. Nalamalpu, S. Srinivasan, and W. P. Burleson, “Boosters for driving long onchip interconnects - design issues, interconnect synthesis, and comparison with repeaters,”

IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Volume 21, Issue 1, pages 50–62, January 2002.

[72] L. M. Ni and P. K. McKinley, “A survey of wormhole routing techniques in direct networks,” *Computer*, Volume 26, Issue 2, pages 62–76, February 1993.

[73] S. M. Nowick and D. L. Dill, “Automatic synthesis of locally-clocked asynchronous state machines,” in *Proceedings of the 1991 IEEE/ACM International Conference on Computer-Aided Design*, November 1991, pages 318–321.

[74] T. Olsson, M. Torkelsson, P. Nilsson, A. Hemani, and T. Meincke, “A digitally controlled on-chip clock multiplier for globally asynchronous locally synchronous systems,” in *Proceedings of the IEEE* 42^{nd} *Midwest Symposium on Circuits and Systems*, Volume 1, August 1999, pages 84–87.

[75] *Open Core Protocol Specification 2.0*, The Open Core Protocol International Partnership, http://ww.ocpip.org, November 2003.

[76] *IEEE Standard SystemC Language Reference Manual*, Open SystemC Initiative (OSCI), 2005, http://www.systemc.org.

[77] *Conventional PCI 3.0*, Peripheral Component Interconnect Special Interest Group (PCI-SIG), http://www.pcisig.com/, April 2004.

[78] W. D. Peterson, *VMEbus Handbook*, 4th Edition, VITA, 1997.

[79] *PCA5007 handshake-technology pager IC data sheet*, Philips Semiconductors, 1999.

[80] C. Rowen, *Engineering the Complex SOC: Fast, Flexible Design with Configurable Processors*. Prentice Hall Professional Technical Reference, June 2004, pages 436-437.

[81] *The Balsa Asynchronous Synthesis System*, School of Computer Science, The University of Manchester, 2005. URL: http://intranet.cs.man.ac.uk/apt/projects/tools/balsa/

[82] J. N. Seizovic, “Pipeline synchronization,” in *Proceedings of the International Symposium on Advanced Research in Asynchronous Circuits and Systems 1994*, November 1994, pages 87–96.

[83] C. E. Shannon, “A mathematical theory of communication,” *Bell System Technical Journal*, Volume 27, pages 379–423 and 623–656, July and October 1948.

[84] S. Shimizu, T. Matsuoka, and K. Taniguchi, “Parallel bus systems using code-division multiple access technique,” in *Proceedings of the 2003 International Symposium on Circuits and Systems*, Volume 2, May 2003, pages 240–243.

[85] D. Sigüenza-Tortosa, T. Ahonen, and J. Nurmi, “Issues in the development of a practical NoC: the Proteo concept,” *Integration, the VLSI journal*, Volume 38, Issue 1, pages 95–105, September 2004.

[86] E. S. Sousa and J. A. Silvester, “Spreading code protocols for distributed spread-spectrum packet radio networks,” *IEEE Transactions on Communications*, Volume 36, Issue 3, pages 272–281, March 1988.

[87] I. E. Sutherland, “Micropipelines,” *Communications of the ACM*, Volume 32, No. 6, pages 720–738, June 1989.

[88] M. Takahashi, T. B. Keat, H. Iwamura, T. Matsuoka, and K. Taniguchi, “A study of robustness and coupling-noise immunity on simultaneous data transfer CDMA bus interface,” in *Proceedings of 2002 International Symposium on Circuits and Systems*, Volume 4, May 2002, pages 611–614.

[89] *The* $TILE64^{TM}$ *Processor*, Tilera, http://www.tile64.com/, 2007.

[90] K. van Berkel, J. Kessels, M. Roncken, R. Saeijs, and F. Schalij, “The VLSI-programming language tangram and its translation into handshake circuits,” in *Proceedings of the 1991 European Conference on Design Automation*, February 1991, pages 384–389.

[91] *Virtual component interface standard*, The Virtual Socket Interface Alliance, http://www.vsi.org, April 2001.

[92] A. J. Viterbi, *CDMA: Principles of Spread Spectrum Communications*. Addison-Wesley Publishing Company, 1995.

[93] D. Wiklund and D. Liu, “SoCBUS: Switched network on chip for hard real time systems,” in *Proceedings of the International Parallel and Distributed Processing Symposium*, April 2003, page 8 pp.

[94] J. V. Woods, P. Day, S. B. Furber, J. D. Garside, N. C. Paver, and S. Temple, “AMULET1: An asynchronous ARM microprocessor,” *IEEE transactions on Computers*, Volume 46, No. 4, pages 385–398, April 1997.

[95] R. Yoshimura, T. B. Keat, T. Ogawa, S. Hatanaka, T. Matsuoka, and K. Taniguchi, “DS-CDMA wired bus with simple interconnection topology for parallel processing system LSIs,” in *Digest of Technical Papers of 2000 IEEE International Solid-State Circuits Conference*, February 2000, pages 370–371.

[96] K. Y. Yun and R. P. Donohue, “Pausible clocking: A first step toward heterogeneous systems,” in *Proceedings of the 1996 International Conference on Computer Design: VLSI in Computers and Processors*, October 1996, pages 118–123.

[97] C. A. Zeferino and A. A. Susin, “SoCIN: A parametric and scalable network-on-chip,” in *Proceedings of the IEEE Integrated Circuits and Systems Design Symposium*, September 2003, pages 169–174.

[98] H. Zhang, V. George, and J. M. Rabaey, “Low-swing on chip signaling techniques: Effectiveness and robustness,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Volume 8, Issue 3, pages 264–272, August 1999.

[99] S. Zhuang, W. Li, J. Carlsson, K. Palmkvist, and L. Wanhammar, “Asynchronous data communication with low power for GALS systems,” in *Proceedings of the* 9th *International Conference on Electronics, Circuits and Systems*, Volume 2, September 2002, pages 753–756.

Part II: Publications

PUBLICATION 1

X. Wang, T. Ahonen, and J. Nurmi, “A Synthesizable RTL Design of Asynchronous FIFO”, in Proceedings of the 2004 International Symposium on System-on-Chip, (SOC 2004), pages 123-128, Tampere, Finland, November 2004.

A Synthesizable RTL Design of Asynchronous FIFO

Xin Wang, Tapani Ahonen, Jari Nurmi
Institute of Digital and Computer Systems, Tampere University of Technology
P.O.Box 553, FIN-33101, Tampere,Finland
E-mail: {xin.wang, tapani.ahonen, jari.nurmi}@tut.fi

Abstract

An asynchronous FIFO which avoids data movement in a micropipeline FIFO is presented and it has been implemented as a gate-level netlist. The presented asynchronous FIFO model is constructed by commonly used Hardware-Description Language and synthesized using the conventional EDA tools and methods for synchronous design. The purpose of this work is to construct a reusable asynchronous FIFO design which suits the commonly used synchronous design tools and flow.

1. Introduction

Globally-Asynchronous Locally-Synchronous (GALS) [1] is a promising paradigm to solve the problem of clock skew and delay in the deep submicron System-on-Chip (SoC) design. In the GALS architecture, the blocks in different clock domains communicate with each other using asynchronous connections. Asynchronous FIFO is an important component for the efficient data transfer in asynchronous communication. Therefore, the asynchronous FIFO design is necessary for implementing the GALS structure in a SoC design.

Asynchronous FIFO using micropipeline is presented in [2]. The main characteristic of a micropipeline FIFO is that the data will flow through all data cells in the FIFO before reaching the output port. Therefore, the latency (the delay from the input of a data item to its presence at the output [4]) caused by data movement is inevitable. In [3], an asynchronous FIFO using counter as control logic is presented, data movement is avoided, but complexity is high. [4, 5] present similar asynchronous FIFO structures using token passing (a sender/receiver can transmit/receive data to/from FIFO only when it has a token) and a common data bus for data in and out. Using these structures, the data can be pushed into or popped from the asynchronous FIFO without data movement inside the FIFO. Therefore, the latency caused by data movement in a micropipeline FIFO is eliminated and less power is consumed. The asynchronous FIFO presented in this paper also bases on the token passing presented in [4, 5], but the difference is that the presented asynchronous FIFO model is suitable for HDL modeling in Register-Transfer Level (RTL) and implementation using the conventional synchronous design tools and flow. Current asynchronous design tools require a significant re-education of designers, and their capabilities are limited compared to commonly used synchronous tools [6]. Therefore, if commonly used synchronous design methods could be used for asynchronous design, the benefit is that it would facilitate the integration of synchronous and asynchronous parts of a design.

The paper structure is as follows. In section 2, the structure of asynchronous FIFO is presented and the control logic is reviewed in detail. Section 3 describes the RTL model of the basic elements used in this synthesizable RTL design of the asynchronous FIFO. The simulation results are presented in section 4. The conclusion is drawn in section 5.

2. The Structure of the Asynchronous FIFO

2.1 Top Structure

There are two blocks in the asynchronous FIFO illustrated in Figure 1 – 'Control Logic' Block and 'Data Bank' Block. This asynchronous FIFO works with four-phase bundled-data handshake protocol [7]. The process of pushing data into the asynchronous FIFO is explained as following steps:

- Step1: The sender sets the request signal ('push_req' signal in Figure1) after the data to be sent are ready ('Data_In[D-1:0]' in Figure1).
- Step2: The FIFO will set acknowledge signal (push_ack signal in Figure1) after successfully obtaining the incoming data.
- Step3: Then the sender responds to push_ack signal by resetting push_req signal.
- Step4: The FIFO resets the push_ack signal after push_req has been reset.

The process of popping data from the asynchronous FIFO is equal to pushing process except that the data is supplied by the FIFO and obtained by the receiver.

The 'control logic' block contains control cells for every data-cell in 'Data Bank' Block. The 'Control Signals' illustrated in Figure 1 are used to control the actions of pushing and popping data from the data bank.

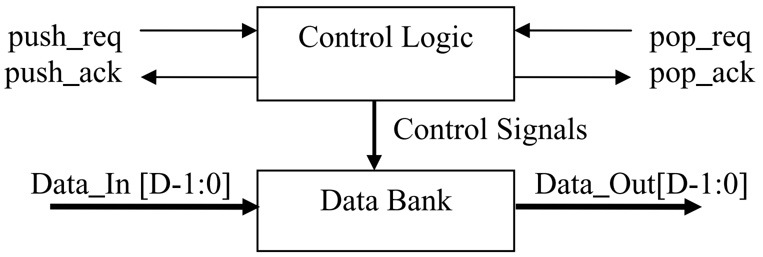


Figure 1. Block Diagram of the Asynchronous FIFO

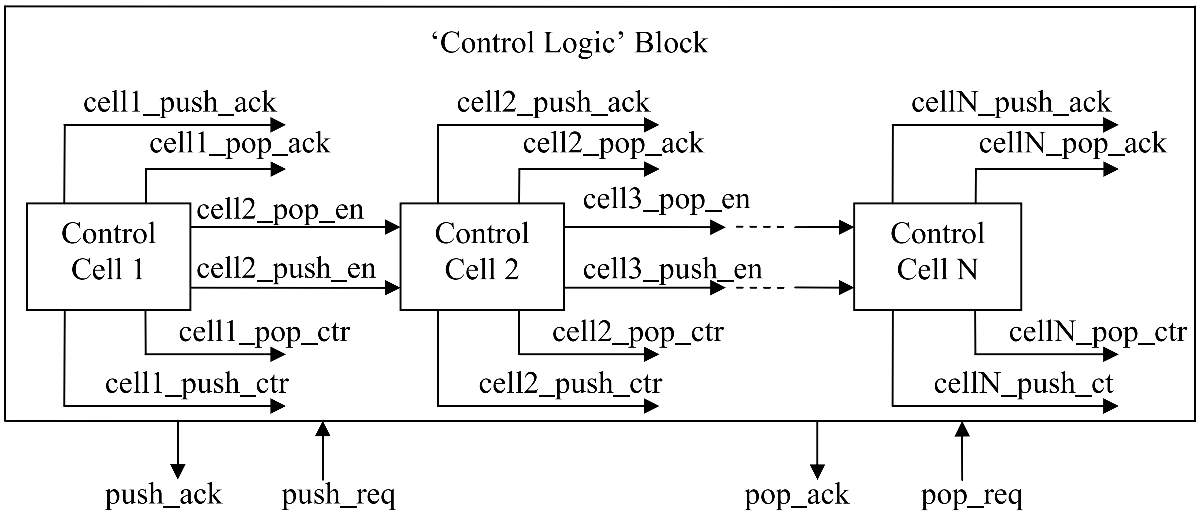


Figure 2. Block Diagram of the 'Control Logic' Block

2.2 The Structure of the 'Control Logic' Block

The 'Control Logic' block consists of N control cells illustrated in Figure 2. The number N is the depth of FIFO referring to the maximum number of data items that can be stored into the FIFO. Each control cell is used to control one data cell in the 'Data Bank' Block. In Figure 2, the push/pop_req input signals are fed into every control cell by default. The basic operating principle of this 'Control Logic' Block is that every control cell responds to push and pop request signals only when it has a token (permission) for the corresponding operation. The token for responding push operation (push token) is granted to 'Control Cell 1' after system reset. Then, the push token will be transferred to the next control cell after the current owner of push token completed its operations about pushing data. When the push token reached the last control cell, 'control cell N', it will return to 'control cell 1'. The operating process of pop token, token for responding pop request, is equal to the process of push token except that the initial pop token will be granted to 'control cell 1' only when the first data cell in 'Data Bank' Block contains a valid data item. With this token passing principle, the data can be pushed into or popped from the asynchronous FIFO eliminating data movement.

The token transfer is implemented by the 'celli_pop/push_en' (i=1, 2, 3, … , N) signals illustrated in Figure 2. For example, when the 'cell2_pop_en' signal is set, it means that the pop token is delivered to 'control cell 2'. After 'control cell 2' has responded to the current pop request, the 'cell2_pop_en' signal will be reset and the 'cell3_pop_en' signal will be set in order to transfer the pop token to 'control cell 3'.

The 'push/pop_ack' output signals in Figure 2 are composed by the 'push/pop_ack' signals from all control cells, which is reasonable because only one control cell will assert its push/pop acknowledge signal for the current push/pop request. The 'cellN_push/pop_ctr' signals in Figure 2 represent the control signals fed into the 'Data Bank' Block.

The detailed structure of 'control cell 1' and other central control cells, which include from 'control cell 2' to 'control cell N-1', are illustrated together in Figure 3, because they have almost the same structure. The dashed line in Figure 3 means those connections are only valid for 'control cell 1'. For example, the 'celli_pop_en' signal will be replaced by 'pop_req' signal when Figure 3 refers to 'control cell 1'. In Figure 3, the letter 'i' used in the signal names refers to the signal index in different control cells. For example, 'celli_push_en' refers to the 'cell2_push_en' signal in 'control cell 2'.

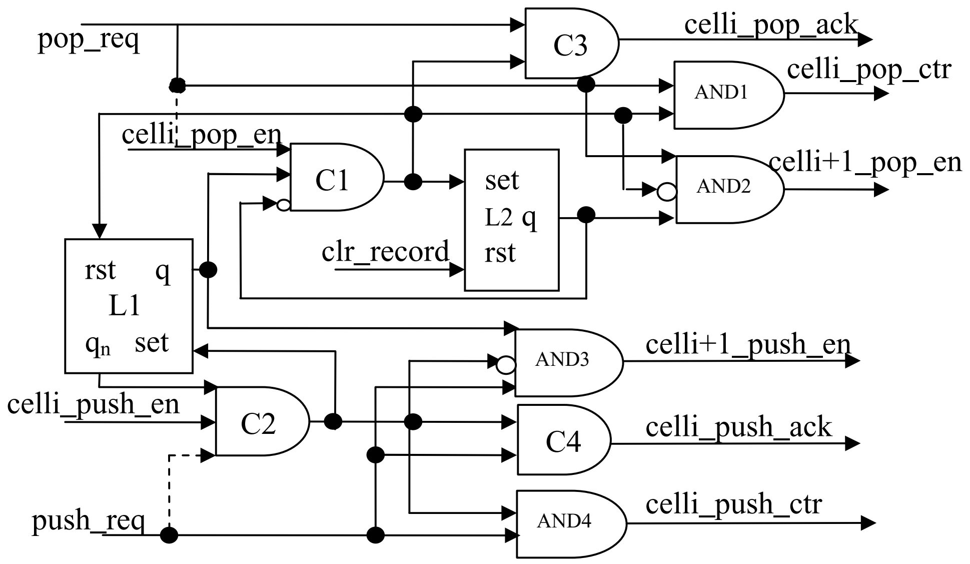


Figure 3. First and Central Control Cell Block Diagram

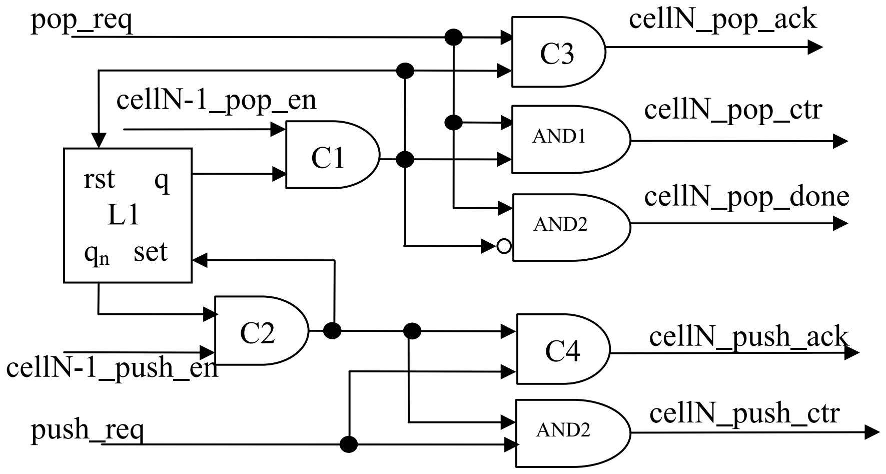


Figure 4. 'Control Cell N' Block Diagram

The little circles at the input port of C1, AND2 and AND3 element represent inverters. In Figure 4, the detailed structure of last control cell – 'control cell N' is depicted.

Three types of logic elements (Latch, Muller C-element and AND logic gate) are used in the control cells. Except the well understood AND logic, it is necessary to describe the model of Latch and Muller C-element used here before introducing the operating principle of an individual control cell.

(1) The truth table of the latch model used here is given in Table 1. When the value at input port 'rst' is set, the output value at port 'q' will be reset. When the value at input port 'set' is set, the output value at port 'q' will be set. The value at 'rst' and 'set' ports should not be set at the same time. The value at output port 'q' will be kept unchanged when the value at 'rst' and 'set' ports is '0'. The value at port 'q_n' is always the inverse value at port 'q'.

Table 1. Truth Table of Latch Model

rst	set	q	q_n
0	0	no change	no change
1	0	0	1
0	1	1	0
1	1	unstable	unstable

(2) Two types of Muller C-element [7] are used in the control cells – two-input and three-input C-elements. The operating principles of these C-elements are the same. The truth table of two-input C-element is given in Table 2 as an example. The output value of C-element would be set/reset only when all of its input values are set/reset.

If the input signals have different values from each other, the output value of C-element should not change.

Table 2. Truth Table of two-input C-element

input a	input b	output q
0	0	0
1	0	no change
0	1	no change
1	1	1

After all logic elements used in the control cells are introduced, the control process of 'control logic' Block can be examined through the 'control cell 1' as the example in the following steps. An assumption taken here is that the transition delay (the time duration in which the output signal becomes stable after the newly coming input signal became stable) of the latch is larger than the transition delay of the inverter. This assumption is reasonable in most cases.

- Step1: After system reset, all output values of logic elements in Figure 3 will be reset except that the 'celli_push_en' signal will be set.
- Step2: When the 'push_req' signal is set, all inputs of C2 will be set. Therefore, the output value of C2 will also be set after a certain delay.
- Step3: Then the output value of C2 will cause the value at 'q' port of L1 be set to '1'. When the value at 'q' port of L1 is set, it means the first data cell in the 'Data Bank' Block contains a valid data item. Otherwise, there is no valid data item in the first data cell.
- Step4: In this step, the 'celli_push_en' will be reset to indicate that the first control cell already responded to the first push_req signal and the push token will be transferred to the next control cell.
- Step5: The 'celli+1_push_en' signal is reset after the output value of C2 is set in step2. This '0' value of 'celli+1_push_en' signal means that the current process of push request is not complete. Thus, the push token should not be transferred to the next control cell at this moment. At the same time, 'celli_push_ack' and 'cell_i_push_ctr' signals which are used to generate acknowledge signal and enable latching the incoming data respectively will be set since all inputs of C4 and AND4 are '1'.
- Step6: After the sender has received the acknowledge signal for the current pushing request, it will reset 'push_req' signal.
- Step7: The reset of 'push_req' signal will cause all inputs of C2 become '0'. Consequently, the output value of C2 will be reset.
- Step8: As the output value of C2 is reset, the inputs of AND3 will be set. Thus, the 'celli+1_push_en' signal will be set when the next pushing request signal comes. It means that the push token is transferred to the next control cell by the asserted 'celli+1_push_en' signal.
- Step9: Because the output of C2 and 'push_req' signal became '0', the output of C4 -- 'celli_push_ack' signal, will be reset. The resetting of acknowledge signal indicates that a four-phase handshake process is completed.

The control process for a pop-request is similar to that of a push-request. The L2 in Figure 3 is used to record the position where the pop token locates by keeping its output always '1' after the output value of C1 is set. That means the control cells which has been granted with pop token will be marked by the '1' value at the output of L2. The pop token will be transferred to the next control cell which has no mark value at its L2 latch. When the pop token returns back to the first control cell, the L2 will be reset by the asserted 'clr_record' signal.

The structure of 'control cell N' illustrated in Figure 4 has some differences by comparing with other control cells depicted in Figure 3. The 'cellN_pop_done' signal is used as one condition to generate 'clr_record' signal. Latch L2 and 'cellN_push/pop_en' signals are not needed in the last control cell because the pop token will be transferred back to 'control cell 1' by resetting the mark value at the output of L2 in other control cells. Except these mentioned differences, the control process is same with the process of the first and the central control cells.

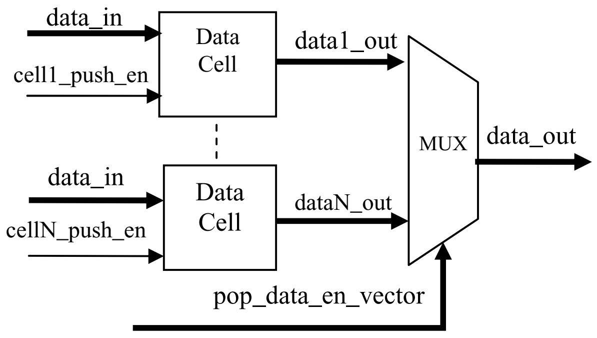


Figure 5. Data Bank Block Diagram

2.3The Structure of the Data Bank

The structure of the 'Data Bank' Block illustrated in Figured 5 is composed by a series of latch-array named as 'Data Cell' and a multiplexer. The main function of the 'Data Bank' Block is to latch the incoming data or output the requested data by the control signals from the 'Control Logic' Block.

The 'celli_push_en' (i=1, 2, … , N) signals coming from 'Control Logic' Block are used as the enable signals for latching the incoming data. The latches used in 'Data Cell' latch-array are same with the latches used in the 'Control Logic' Block. The number of latches in 'Data Cell' latch-array depends on the data width of the asynchronous FIFO. The 'pop_data_en_vector' in Figure 5 is the combination of 'celli_pop_en' signals from the control cells depicted in Figures 3 and 4. It is used as the control signal of selecting the requested data item in different data cells.

3. The RTL Modeling

Three elements, AND logic gate, Muller C-element and latch, are used in this asynchronous FIFO. Unlike the AND logic gate, the Muller C-element and the latch model used in this design are not basic elements in the conventional synchronous design. Therefore, in order to make this asynchronous FIFO to be constructed by commonly used HDL description and synchronous design flow, it is necessary to construct the RTL models using basic elements in digital circuits for Muller C-element and the latch model described in section 2.2. D-Latch, inverter, OR logic gate and AND logic gate are used as the basic elements in this RTL design.

3.1 The RTL Model of C-element

Figure 6 illustrates a two-input C-element model suitable for synthesizable HDL modeling. The three-input C-element model can be obtained by adding the third input on AND1 gate and OR1 gate respectively.

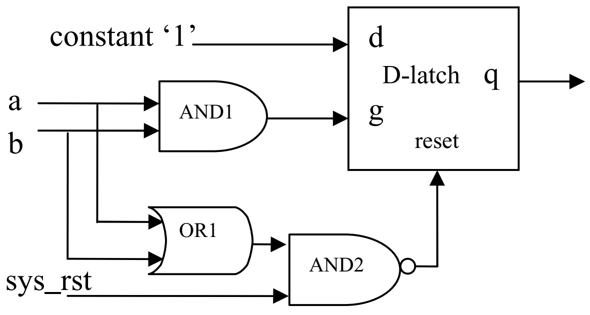


Figure 6. The two-input C-element model

The latch used in the two-input C-element model is a D-Latch with an asynchronous reset port. The value at 'q' output port of D-Latch will follow the value at 'd' port when the value at 'g' port is '1' (gate is open). If the value at 'g' port becomes '0' (gate is closed), the value at 'q' port will be kept unchanged. The 'reset' port is used to reset the value at 'q' port, that is, the value at 'q' port will be reset whenever the value at 'reset' port is '1'. The operating process of the two-input C-element model can be observed through Figure 6 directly. During the system reset (sys_rst = '0'), the output value of this C-element at 'q' port will be reset to '0'. If one input signal 'a' or 'b' is '1' and another is '0', the gate of D-Latch will be closed. Therefore, the reset value at 'q' port will be kept unchanged. When the values at 'a' and 'b' are both '1', the gate of D-Latch will open. Then the constant value '1' at 'd' port will be captured by the output port 'q' of D-Latch, thus, the output value of this C-element will be '1'. The output value of C-element will be back to '0' again only when both input signals 'a' and 'b' are reset to '0'.

3.2 The RTL Model of Latch

The D-Latch described in section 3.1 can also be used to construct the latch model described in section 2.2. The latch model used in this design has two input ports 'rst' and 'set' and one output port 'q' ('qn' is the inverse of 'q'). The RTL model of the latch can be obtained by feeding the 'set' input directly into the 'g' port of D-Latch and the 'rst' port into the 'reset' port of D-Latch. The 'd' port of D-Latch is connected with constant '1' as illustrated in Figure 6.

With the RTL model depicted in sections 3.1 and 3.2, the synthesizable model of the asynchronous FIFO can be constructed with the commonly used hardware description languages such as VHDL and Verilog. Then, the synthesizable HDL descriptions will suit the synchronous design tools naturally in the following steps of the conventional synchronous design flow.

4. Simulation Results

The asynchronous FIFO has been implemented using VHDL. A 0.35µm technology library was used for synthesis. The latency of the asynchronous FIFO is measured as the time between the positive edge of 'push/pop_req' signal and the negative edge of the acknowledge signal. This latency is independent on the depth of the asynchronous FIFO, because the push/pop request is performed into/from the respective data cell. Table 3 summarizes the timing characteristics obtained by gate-level simulation when data width of asynchronous FIFO is four bits and FIFO depth is four.

All the values presented in Table 3 are the average values among the measured values of the four different data cells in 'Data Bank' Block. The terms presented in Table 3 are explained as following.

- 'Data Valid Delay' is measured as the time between the rising edge of 'pop_req' signal and the valid data appearing on the output of the FIFO.
- 'Ack Rise Delay' is measured as the time between the rising edge of 'push/pop_req' signal and the rising edge of 'push/pop_ack' signal.

Table 3 Timing Characteristics of the Asynchronous FIFO

	DataValid Delay (ns)	Ack Rise Delay (ns)	Req Hold Time (ns)	Ack Fall Delay (ns)	Handshake Cycle (ns)
Push Request	-	3.78	0.50	3.34	7.62
Pop Request	2.95	3.51	0.01	3.41	6.93

- 'Req Hold Time' is measured as the time between the rising edge of 'push/pop_ack' signal and the falling edge of 'push/pop_req' signal.
- 'Ack Fall Delay' is measured as the time between the falling edge of 'push/pop_req' signal and the falling edge of 'push/pop_ack' signal.
- 'Handshake Cycle' is measured as the time between the rising edge of 'push/pop_req' signal and the falling edge of 'push/pop_ack' signal.

The timing for pushing and popping data from different data cells is different. The reason is the different signal paths and the different types of gates incorporated. The delay in the asynchronous FIFO mainly depends on the technology used. According to the timing of 'Handshake Cycle', the throughput of the asynchronous FIFO is about 100 million data items per second when 0.35μm technology is used.

5. Conclusions

An asynchronous FIFO structure suitable for HDL modeling and the conventional synchronous design tools and flow is presented and the gate-level simulation results are discussed. An approach of constructing the asynchronous circuits using HDL descriptions is presented. The approach is that constructing the RTL model of the circuits with Muller C-element and latches firstly, then, modeling the latches and Muller C-elements using applicable basic element, such as D-Latch, which can be described by HDL code. The presented RTL model of Muller C-element and latches can be reused for other asynchronous design if the RTL models of those elements are needed.

The drawback of the presented asynchronous FIFO design is that the RTL model of Muller C-element is not as efficient as the model constructed in circuit-level. If the circuit-level model of Muller C-element is available as a library component, this disadvantage can be conquered.

References

[1] Muttersbach, J.; Villiger, T.; Kaeslin, H.; Felber, N.; Fichtner, W.; "Globally-asynchronous locally-synchronous architectures to simplify the design of on-chip systems"; *ASIC/SOC Conference, 1999. Proceedings*. Twelfth Annual IEEE International , 15-18 Sept. 1999; Pages:317 – 321.

[2] I.E. Sutherland;"Micropipelines"; *Communications of the ACM*, vol. 32, no.6, pp. 720-738, June 1989.

[3] A.V. Yakovlev, A.M. Koelmans, L. Lavagno, "High-Level Modeling and Design of Asynchronous Interface Logic", *IEEE Design and Test of Computers*, Spring 1995.

[4] Chelcea, T.; Nowick, S.M.; "Low-latency asynchronous FIFO's using token rings"; *Advanced Research in Asynchronous Circuits and Systems*, *2000. (ASYNC 2000) Proceedings. Sixth International Symposium*, 2-6 April 2000 Pages: 210 – 220

[5] K.K. Yi, "The Design of a Self–Timed Low Power FIFO Using a Word–Slice Structure", *M.Phil Thesis, Univ. of Manchester*, September 1998.

[6] Kondratyev, A.; Lwin, K.; "Design of asynchronous circuits by synchronous CAD tools", *Design Automation Conference, 2002. Proceedings. 39th* , 10-14 June 2002 Pages:411 – 414

[7] Jens Sparso, Steve Furber; "Principles of Asynchronous Circuit Design-Asystem Perspective"; Page 9~11, 15; Kluwer Academic Publishers, 2001.

PUBLICATION 2

X. Wang, D. Sigüenza-Tortosa, T. Ahonen, and J. Nurmi, “Asynchronous Network Node Design for Network-on-Chip”, in Proceedings of the 2005 International Symposium on Signal, Circuits, and System, (ISSCS 2005), Volume 1, pages 55-58, Iasi, Romania, July 2005.

Asynchronous Network Node Design for Network-on-Chip

Xin Wang, David Sigüenza-Tortosa, Tapani Ahonen, Jari Nurmi
Institute of Digital and Computer Systems, Tampere University of Technology, Tampere, Finland
{xin.wang, david.siguenza-tortosa, tapani.ahonen, jari.nurmi}@tut.fi

***Abstract*— A network node for Proteo Network-on-Chip (NoC) has been developed in order to support Globally-Asynchronous Locally-Synchronous (GALS) communication in an on-chip system. The network node presented in this paper was implemented as a synthesizable Intellectual Property (IP) block in Register-Transfer Level (RTL) using VHDL. The proposed design applies both asynchronous and synchronous circuits to make the globally asynchronous data transfer rate between network nodes independent of local clocks.**

I. INTRODUCTION

A System-on-Chip (SoC) combines multiple functional IP blocks together to implement complex applications. As the number of components becomes larger, designing a SoC requires the introduction of networking concepts to manage the complexity of the interconnection. Network-on-Chip (NoC) addresses the issue of constructing an efficient and flexible on-chip communication infrastructure in the framework of Deep Sub-Micron (DSM) technology.

Proteo [1] is a packet-switched NoC developed at Tampere University of Technology. A Proteo NoC instance basically consists of network nodes and asynchronous links. Each functional block in the system is connected to a corresponding network node through an interface which supports VCI [2] and OCP [3] standards. A synchronous network node IP [4] [5] has been implemented for constructing Proteo NoCs. Nodes could be clocked using the same clock signal as the local functional block uses or introducing an independent network clock. Both schemes present difficulties in synchronization. In this situation, a Globally-Asynchronous Locally-Synchronous (GALS) scheme [6] was proposed as a solution. For a NoC, GALS means that data transfers between each functional block and its attached node are synchronous, whereas data transfers between network nodes are asynchronous. The network node presented in this paper uses asynchronous circuits to perform global data transfers and synchronous circuits to deal with local data transfers.

This paper is organized as follows: in Section II, the structure of the network node is presented. The implementation issues will be addressed in Section III. In Section IV, the synthesis and simulation results are presented. Finally, conclusions are drawn in Section V.

II. NETWORK NODE STRUCTURE

A. Block Description

The network node structure is illustrated in Fig.1. The two blocks outside of the dash-dot frame represent the functional IP block ('Functional Host') which is connected to the network node through its VCI or OCP standard interface ('Standard Network IF') block. The arrows in Fig.1 illustrate the data flow. The function of the blocks in the network node will be described in the following paragraphs.

I) 'Node IF'. This block is the interface block which complies with the VCI or OCP standards. It acts as the counterpart of the block named 'Standard Network IF' which belongs to the functional host. If 'Standard Network IF' is of the master type, 'Node IF' should be of the slave type and vice versa (details can be found in [2] [3]). The functionality of this block consists of communicating with the functional host and, through 'Layer MUX', the 'Communication Layer' blocks, and assembling or extracting data into or from the predefined Proto packet format.

II) 'Layer MUX'. This block behaves as a multiplexer connecting 'Node IF' with a set of 'Communication Layer' blocks. 'Layer MUX' can connect 'Node IF' with two different 'Communication Layer' blocks at the same time if only one of them is used to send packets and the other one is used to receive packets. However, the 'Communication Layer' block for sending packets and the 'Communication Layer' block for receiving packets can be the same block at any given time.

III) 'Communication Layer'. The function of this block is to perform the globally asynchronous communication with other network nodes through a handshake protocol. In Fig. 1, two 'Communication Layer' blocks labeled with 1 and 2 respectively are presented, but the number of 'Communication Layer' blocks can be more than two. Each 'Communication Layer' block is used to connect the network node into a certain network topology. The function of the sub-blocks that constitute 'Communication Layer' will be explained by describing the communication process of sending and receiving data packets in the following paragraphs:

a) Sending a locally generated packet. After 'Node IF' obtained the data to be sent from the functional host and assembled it into a packet, called 'local packet', 'Node IF' will send a request signal to the selected 'Communication Layer'. Then the 'Communication Controller' sub-block will check whether the FIFO array in the 'Packet Sender', called 'Tx Packet Buffer', is full or not. If the buffer is full, 'local packet' will be held by 'Node IF' until there is a room available. If the buffer is not full, 'Communication Controller' will enable the 'Packet Distributor' to push 'local packet' into 'Tx Packet Buffer'. The data packets in 'Tx Packet Buffer' will be sent to the adjacent network node by 'Packet Sender' using a handshake protocol under the control of 'Communication Controller'.

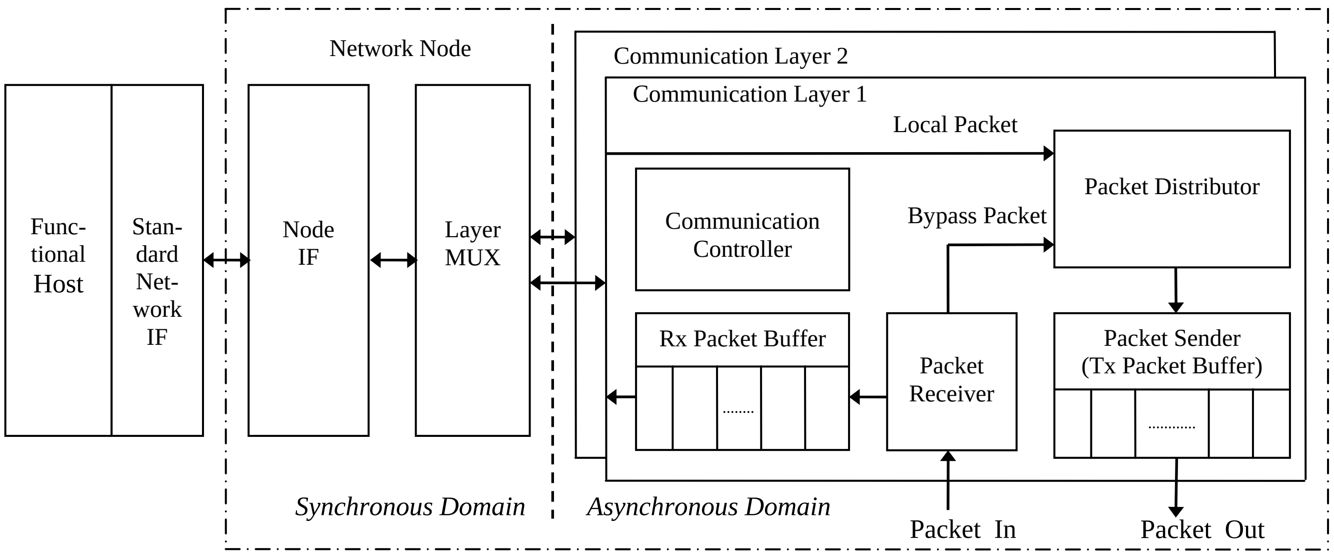


Figure 1. Network Node Block Diagram and Data Flow

b) Receiving a packet. If 'Communication Controller' received the packet-transfer request signal from the network node connected to the input of the node, it will enable 'Packet Receiver' to obtain the data packet. 'Packet Receiver' will check the destination address information, and if it is the current network node, the 'incoming packet' will be delivered to the FIFO array named 'Rx Packet Buffer' after getting the grant signal from the 'Communication Controller'. If 'Packet Receiver' found that the destination of the received packet is not the current network node, the packet is a 'bypass packet', and will be pushed into 'Tx Packet Buffer' through 'Packet Distributor' under the control of 'Communication Controller'. If either buffer is full when needed, 'Packet Receiver' will hold the packet until there is room available.

B. Packet Transfer Arbitration

Different packet transfer processes may use the same sub-block in 'Communication Layer' during sending or receiving packets. For example, 'Packet Distributor' is a shared resource for moving 'local packet' and 'bypass packet' into 'Tx Packet Buffer'. Therefore, an arbitration mechanism is needed to coordinate the packet transfer processes in the 'Communication Layer' block. The basic principle of arbitration is 'First Come, First Served'. In case of conflict, the resolution mechanism is explained below:

I) One conflict may occur if the requests of pushing 'local packet' and pushing 'bypass packet' into 'Tx Packet Buffer' are presented to 'Communication Controller' simultaneously. In this situation, 'bypass packet' will be sent into 'Tx Packet Buffer' first. Assigning a higher priority to 'bypass packet' can reduce the communication load of network.

II) Another conflict may occur in 'Packet Sender' sub-block if it receives the packet pushing request from 'Packet Distributor' and transmission handshake request from the network node connected to the output of the node simultaneously. In this case, 'Packet Sender' will send the packet in 'Tx Packet Buffer' to the other network node first in order to make room in the buffer for new packets. This strategy decreases the probability of 'Tx Packet Buffer' being full. Also other conflict handling schemes [7] [8] have been devised and will be implemented.

III. Implementation of The Network Node

In order to implement the GALS scheme in Proteo NoC, the presented network node applies both synchronous and asynchronous circuits, delimited by the dash line in Fig.1.

The reason of applying synchronous circuits is that both most of the functional hosts available and the network interface standards used in Proteo NoC are synchronous. Therefore, the network node should work in a synchronous manner and at the same clock rate when communicating with its functional host in order to be naturally compatible with it. The blocks in this domain are implemented in RTL with VHDL.

The blocks in the asynchronous domain interact with each other and perform the globally asynchronous data transfer using a handshake protocol. A four-phase bundled data protocol is used in this implementation. In order to make the asynchronous design compatible with the commonly used synchronous design tools and flow, the asynchronous circuits are constructed in RTL with VHDL. The approach is to construct the asynchronous circuits using C-elements, latches and combinational logic gates, and then describe the RTL structure in VHDL. An example of using this approach to design asynchronous circuits can be found in [9].

The challenge of combining the synchronous and asynchronous circuits together is how to avoid synchronization failure caused by setup or hold-time violation during signal transfer from asynchronous domain to synchronous domain. Two principal solutions of this problem have been developed. One solution is to stretch or pause the clock signal [10, 11, 12]. The other solution is to reduce the probability of synchronization failure by using multiple receiving flip-flops. The first category of solutions introduces some special components implemented in

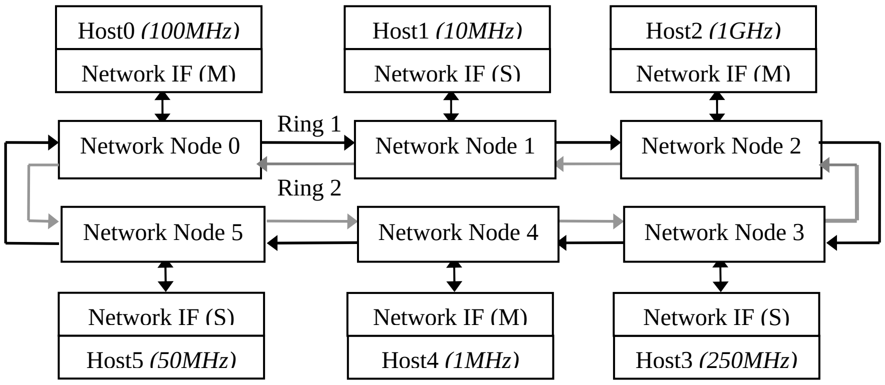


Figure 2. Six-Node Bidirectional Ring Network

transistor level, such as stretchable or pausable clock generator [10, 11] and delay lines [12], therefore, it does not meet the requirement of implementing the presented network node in RTL. The synchronization method used in the presented network node is a double-latching scheme. Hence, synchronization failure may occur during the double-latching process. However, in this network node, the synchronization failures only can delay the data transfer other than ruin it, because the double-latching scheme is only used for sampling the handshake signals from asynchronous domain other than the data. For example, if '0→1' transition of a request signal from asynchronous domain failed to be sampled by the receiving flip-flop, the block in synchronous domain would not latch the data until it got the request signal correctly after the receiving flip-flop recovered from the synchronization failure in following cycles.

Therefore, by designing both the synchronous and asynchronous circuits using VHDL, the network node is implemented as a synthesizable IP block which suits the commonly used synchronous EDA design tools.

IV. Synthesis and Simulation

A. *Synthesis Results*

The network node has been synthesized using a 0.18μm standard cell technology library. The data depth and data width of FIFO used in 'Tx/Rx Packet Buffer' is set to 4 and 32 bits respectively. The area cost (without wire area) for each block of the network node is listed in Table I. The FIFOs in 'Tx/Rx Packet Buffer' take around 58% area of the network node.

TABLE I. Area Cost of Network Node

Blocks of Network Node	**Area (μm^2)**	**Percentage of total area**
Node IF (BVCI Slave Type)	13430.8	9.7 %
Layer MUX	18346.0	13.3 %
Communication Controller	7823.4	5.7 %
Packet Distributor	6783.0	4.9 %
Packet Sender (include Tx Packet Buffer)	44740.6	32.3 %
Packet Receiver	6955.0	5.0 %
Rx Packet Buffer	40255.5	29.1 %
Total	138334.3	100 %

B. *Simulation Network Set-up*

Fig.2 illustrates the network constructed with the presented network node for simulation. All the nodes are connected together in a bidirectional ring topology which consists of one clockwise ring, marked as 'Ring 1', and one anti-clockwise ring, marked as 'Ring 2'. The number of data cells (handshake units) in each packet varies between two and four. The width of each data cell is 32 bits. In this simulation, three hosts are acting as masters and the other three as slaves, as denoted by the labels 'M' and 'S' in the 'Network IF' blocks. The different hosts work at different clock frequencies as illustrated in Fig.2. Any master can send a request to any slave. Request and response packets travel through the shortest path in the network according to a simple deterministic hop-by-hop routing mechanism. For example, requests sent from Host 0 to Host 3 are delivered through nodes 1 and 2 ('Ring 1'). The interface standard modeled in this simulation is the Basic VCI (BVCI).

The data transactions performed in the simulation are listed in Table II. Each data transaction consists of one request packet from a Master Node to a Slave Node and one corresponding response packet from the Slave Node to the Master Node.

C. *Simulation Results*

The simulation is performed in gate level. The data transfer latency between functional host and network node is measured as the locally synchronous transfer latency, which depends on the local clock and the type of interface. The measured values of the

TABLE II. Data Transactions Specification

Master Node	**Slave Node**	**Number of Transactions**	**Packet Length**	
			Request Packet	**Response Packet**
Node 0	Node 1	1	4	2
	Node 3	1	3	3
	Node 5	3	3, 4, 3	3, 2, 3
Node 2	Node 1	4	3, 4, 3, 4	3, 2, 3, 2
	Node 3	1	3	3
	Node 5	1	3	3
Node 4	Node 1	1	4	2
	Node 3	3	4, 3, 4	2, 3, 2
	Node 5	1	4	2

TABLE III. SYNCHRONOUS TRANSFER LATENCY

Interface Type	Latency of sending data to 'Network Node'	Latency of receiving data from 'Network Node'
BVCI Master	8 local clock cycles	13 local clock cycles + 2.6 ns
BVCI Slave	4 local clock cycles	9 local clock cycles + 2.6 ns

TABLE IV. ASYNCHRONOUS TRANSFER LATENCY PARAMETERS

Packet Length	PLL (ns)	PTL (ns)	PBL (ns)	PSL (ns)
2 data cells	11.7	9.7	10.7	3.3
3 data cells	15.2	13.1	14.2	3.3
4 data cells	18.6	16.5	17.6	3.3

synchronous transfer latency are listed in Table III. The constant value of 2.6 ns in Table III is caused by the latency of popping data from 'Rx Packet Buffer' in the asynchronous domain, and it is independent on the local clock rate but belongs to the process of receiving data from 'Network Node'.

The latency for globally asynchronous transfers consists of four parameters: Packet Loading Latency (PLL), Packet Transfer Latency (PTL), Packet Bypass Latency (PBL), and Packet Storing Latency (PSL). These latency parameters are measured in a non-congested situation, which means that the packet transfer conflicts discussed in Section II are not included in the simulation. The concept of the four latency parameters is illustrated in Fig.3 with an example: 'Network Node 0' sends one packet to 'Network Node 2' via 'Network Node 1'. The black arrows in Fig.3 represent the packet transfer direction. The portions of the transfer used to measure the different parameters of latency are marked by gray arrows in Fig.3 and explained as below:

I) Packet Load Latency (PLL): It is the time used to load one 'local packet' into 'Tx Packet Buffer'.

II) Packet Transfer Latency (PTL): This latency refers to the time used to transfer one data packet from the 'Packet Sender' of a network node to the 'Packet Receiver' of an adjacent node using a four-phase handshake protocol.

III) Packet Bypass Latency (PBL): After a network node receives a packet from another node, it will check its destination address. If it is not targeted to the current node, the 'bypass packet' is transferred into 'Tx Packet Buffer'. The time spent on these operations is called PBL.

IV) Packet Storing Latency (PSL): It is the time it takes to store one 'incoming packet' into 'Rx Packet Buffer'.

The formula of Asynchronous Transfer Latency (ATL) of one packet is given in equation (1). It represents the situation in which the packet traverses several network nodes before reaching its destination. N refers to the number of intermediate nodes between the source node and destination node of a packet. If a packet is transferred between two adjacent network nodes, then N is 0.

$$ATL = PLL + PTL \times (N+1) + PBL \times N + PSL \quad (1)$$

The values of asynchronous transfer latency parameters measured in the 0.18μm technology node are listed in Table IV. The listed latency values only include the logic gate delay of the circuits, no wire delay is considered. More accurate latency values could be obtained by adding the wire delay after layout.

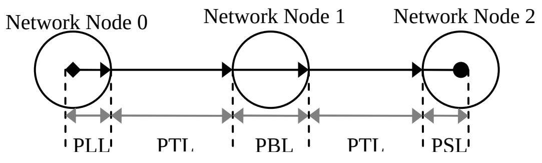


Figure 3. Asynchronous Transfer Latency Parameters

In Table IV, we can see that PLL, PTL, and PBL increase as the packet length increases. This is because the data cells in a packet are sent in serial way, so that more data cells need more time for transferring. The reason that PSL is not affected by the packet length is that the data cells of 'incoming packet' are stored in parallel in 'Rx Packet Buffer'.

V. CONCLUSIONS

A network node for Proteo NoC which can support a GALS communication scheme in on-chip systems was presented. It uses asynchronous circuits to perform global data transfers between network nodes, and synchronous circuits to deal with the local data transfers. Both the asynchronous and synchronous circuits of this network node were implemented using VHDL to suit the conventional synchronous design tools. A six-node bidirectional ring network was constructed and synthesized for simulation at gate level. Figures for the asynchronous transfer latency between network nodes are given. The simulation reveals that the latency of the globally asynchronous transfers of data packets is independent on the local clock rates at each functional host.

REFERENCES

[1] D. Sigüenza-Tortosa and J. Nurmi, "Issues in the Development of a Practical NoC: the Proteo Concept", *Integration, the VLSI journal*, volume 38, issue 1, 2004.

[2] VSI Alliance. *Virtual Component Interface Standard v 2*, April 2001

[3] OCP-IP Association. *Open Core Protocol Specification*, 2001.

[4] I. Saastamoinen, M. Alho, J. Pirttimäki and J. Nurmi, "Proteo Interconnect IPs for Networks-on-Chip", *Proceedings of IP Based SoC Design 2002*, Grenoble, France, October , 2002.

[5] M. Alho and J. Nurmi, "Implementation of Interface Router IP for Proteo Network-on-Chip", *Proceedings of DDECS'03*, Poland, April, 2003.

[6] D. M. Chapiro, "Globally-Asynchronous Locally-Synchronous Systems", *PhD thesis*, Stanford University, Oct. 1984

[7] D. Sigüenza-Tortosa and J. Nurmi, "Packet Scheduling In Proteo Network-on-Chip", *Proceedings of IASTED PDCN 2004*, Innsbruck, Austria, February 2004.

[8] D. Sigüenza-Tortosa and J. Nurmi, "Packet Scheduling Configuration in Proteo Network-on-Chip", *Proceedings of IEE CSNDSP*, Newcastle-upon-Tyne, UK, July 2004.

[9] X. Wang, T. Ahonen, and J. Nurmi, "A Synthesizable RTL Design of Asynchronous FIFO", *Proceedings of 2004 International Symposium on System-on-Chip*, Tampere, Finland, November, 2004.

[10] K. Y. Yun and R. P. Donohue, "Pausible clocking: A first step toward heterogeneous systems", *Proceedings of International Conf. Computer Design (ICCD)*, Austin, USA, Oct. 1996.

[11] J. Muttersbach, T. Villiger, H. Kaeslin, N. Felber, and W. Fichtner, "Globally-asynchronous locally-synchronous architectures to simplify the design of on-chip systems", *Proceedings of the 12th Annual IEEE International ASIC/SOC Conference*, Washington, DC, USA, Sep. 1999.

[12] S. Moore, G. Taylor, R. Mullins, and P. Robinson, "Point to point GALS interconnect", *Proceedings of Eighth International Symposium on Asynchronous Circuits and Systems*, Manchester, UK, April 2002.

PUBLICATION 3

X. Wang, and J. Nurmi, “An On-Chip CDMA Communication Network”, in Proceedings of the 2005 International Symposium on System-on-Chip, (SOC 2005), pages 155-160, Tampere, Finland, November 2005.

An On-Chip CDMA Communication Network

Xin Wang, Jari Nurmi
Institute of Digital and Computer Systems, Tampere University of Technology, Tampere, Finland
{xin.wang, jari.nurmi}@tut.fi

Abstract

An on-chip packet-switched communication network which applies Code-Division Multiple Access (CDMA) technique has been developed and implemented in Register-Transfer Level (RTL) using VHDL. In order to support Globally-Asynchronous Locally-Synchronous (GALS) communication scheme, the proposed CDMA on-chip network combines both synchronous and asynchronous circuits together. In a packet-switched Network-on-Chip (NoC) which applies point-to-point connection scheme, the data transfer latency varies largely if the packets are transferred to different destinations or to a same destination through different routes in the network. The proposed CDMA NoC can make the data transfer latency become a constant value by multiplexing the data transfers in code domain instead of in time domain. Therefore, the data transfer latency can be guaranteed in the proposed CDMA network by avoiding communication media sharing in time domain.

1. Introduction

As more and more components are integrated into an on-chip system, the communication issue in the system becomes complicated. Network-on-Chip is proposed to solve the on-chip communication issue by separating the concerns of communication from computation and constructing an on-chip communication network to connect the system components together. The NoC structures which have been proposed can be sorted into two categories, circuit-switched and packet-switched network. PROPHID architecture [1] is an example of circuit-switched network which connects the terminals in the network by allocating them a set of time or space slices on the communication links. In packet-switched category, SPIN [2] and Proteo NoC [3] are the examples. SPIN network applies fat-tree topology and router blocks to transfer data packets from source node to destination node. In Proteo NoC, all functional Intellectual Property (IP) blocks in a system are connected through network nodes and hubs. The network topology and connection in Proteo NoC can be customized and optimized for a specific application. The circuit-switched network will face the problem of scalability and parallelism if it is applied in a future on-chip system which contains hundreds of functional IP blocks. The packet-switched network can overcome the shortcomings of circuit-switched network, however, if it applies point-to-point connection as in [2] and [3], the packet transfer latency will be uncertain when data packets are transferred to different destinations or to a same destination via different routes in the network.

CDMA as one of spread-spectrum techniques [4] has been widely used in wireless communication systems because it has great bandwidth efficiency and multiple access capabilities. CDMA applies orthogonal codes to encode the information before transmission in a communication media, hence, it permits multiple users to use the communication media parallel in time domain by separating the different data streams in code domain. The CDMA NoC proposed in this paper uses the multiple-access feature of CDMA technique to transfer the data packets from the source nodes to their destination nodes directly and in parallel. Namely, the variance of data transfer latency caused by sharing the communication media in time domain in a point-to-point connection network can be eliminated. Therefore, the data transfer latency in the proposed CDMA NoC can be guaranteed.

In the second section of this paper, the considerations about applying CDMA technique into an on-chip network will be discussed. In Section 3, the structure of the proposed CDMA NoC will be presented. The synthesis and simulation results of the CDMA NoC will be addressed in Section 4. Finally, the conclusions are drawn in Section 5.

2. Applying CDMA Technique in NoC

The principle of CDMA system is to use orthogonal spreading codes to encode the original data. Then the encoded data from different data senders are added together for transmission without messing each other because of the orthogonality of spreading codes. The orthogonality means that the normalized auto-correlation of spreading codes is 1, while the cross-correlation of spreading codes is 0. Therefore, at the receiving end the data can be decoded from the received signals by multiplying the received signals with the corresponding spreading code. The following paragraphs will discuss the considerations which have been made for the proposed CDMA network in this paper.

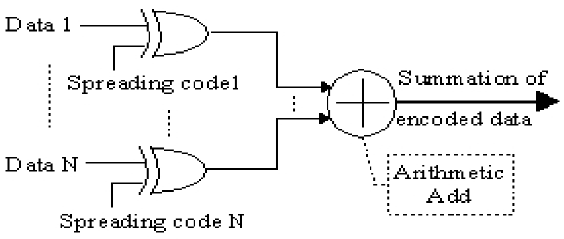
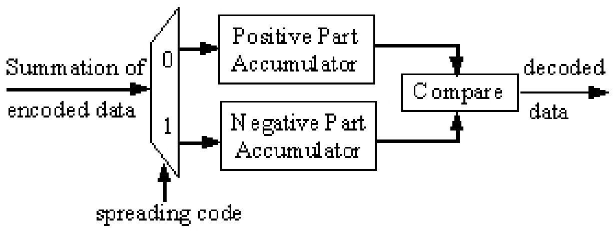


Figure 1. Digital CDMA Encoding Scheme

2.1. Digital Encoding and Decoding Scheme

Several on-chip bus schemes which apply CDMA technique have been proposed in [5] [6] [7] [8]. Those schemes are implemented by analog circuits, namely, the encoded data are represented by the continuous voltage or capacitance value of the circuits. Therefore, the data transfers on the analog bus are challenged by the coupling noise, clock skew, and variations of capacitance and resistance caused by circuit implementation [8]. In order to avoid the challenges faced by the analog circuit implementation, digital circuit implementation is preferred in the proposed CDMA NoC. The encoding scheme suitable for digital circuit implementation is illustrated in Fig.1. The principle, as proposed in [9], is to transfer the binary equivalent of the summation value of the encoded data to the receiving end, which is instead of using the encoded data value to modulate the voltage or capacitance parameters in the analog CDMA buses. This encoding scheme can reduce the number of wires for data transfer. For example, in Fig. 1, if we use one data transfer wire for each data source, the number of wires for data transfer will be N, whereas, if the binary summation value of the data is transferred, we only need $\log_2 N$ wires.

A new decoding scheme suitable for the proposed CDMA NoC is illustrated in Fig.2. The new decoding scheme simplifies the decoding scheme presented in [9] by accumulating the received summation values into two separated parts, positive part and negative part, according to the spreading code used for decoding. As illustrated in Fig.2, the received summation values will be accumulated into the positive part when the current chip of spreading code for decoding is 0, otherwise, they will be accumulated into the negative part. The principle can be explained as following. If the original data to be encoded are 1, after the XOR logic in the encoding scheme, they can only contribute non-zero value to the summation of encoded data when a chip of spreading code is 0. Similarly, the 0-value original data to be encoded can only contribute to the summation of encoded data when a chip of spreading code is 1. Therefore, after accumulating the summation values according to the bit values of spreading code, either the positive part or negative part is larger than the other if the spreading code has orthogonal and balance property. Hence, the original data can be decoded by comparing the two accumulation parts.



Summation of encoded data
0
1
spreading code
Positive Part Accumulator
Negative Part Accumulator
Compare
decoded data

Figure 2. Digital CDMA Decoding Scheme

2.2. Spreading Code Selection

As discussed in Section 2.1, the proposed decoding scheme requires the spreading codes used in the CDMA NoC should have the orthogonal and balance properties. The orthogonal property has been explained in the first paragraph of Section 2. The balance property in this paper means that the number of bit ‘1’ and bit ‘0’ in a spreading code should be equal. Many spreading codes have been proposed for CDMA communication, such as Walsh code, M-sequence, Gold sequence, and Kasami sequence etc [10]. However, only Walsh code has both the orthogonal and balance properties. Therefore, the proposed CDMA NoC uses Walsh code as the spreading code in the network. In a L-bit (L>0, L mod 4=0) length Walsh code set, there are L-1 sequences which have both orthogonal and balance properties. Hence, the proposed CDMA NoC can connect L-1 functional hosts at most.

2.3. Spreading Code Protocol

Spreading code protocol is a policy used to decide how to assign and use the spreading codes in the CDMA network in order to eliminate or reduce the possible conflicts during the communication processes. Several spreading code protocols have been proposed for CDMA packet radio network [11] [12] and will be shortly introduced as below.

1) Common Code Protocol (C protocol): All users in the network use a same spreading code to encode their data packets to be transferred.

2) Receiver Based Protocol (R protocol): Each user in the network will be assigned a unique spreading code which will be used by other users to send data to it.

3) Transmitter Based Protocol (T protocol): The unique spreading code allocated to each user will be used by the user itself to transfer data to others.

4) Common-Transmitter-Based Protocol(C-T protocol): The destination address of a data packet is encoded using C protocol, whereas, the data portion of a packet is encoded using T protocol.

5) Receiver-Transmitter-Based Protocol (R-T protocol): It is same as the C-T protocol except that the destination address of a data packet is encoded using R protocol.

6) Transmitter-Receiver-Based Protocol (T-R protocol): Two unique spreading codes will be assigned to each user in the network, and then a user will generate a new

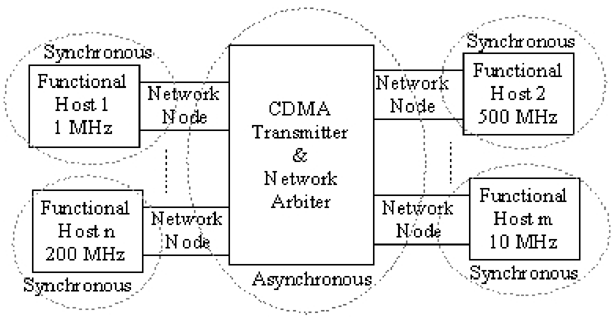


Figure 3. The Proposed CDMA NoC Structure

spreading code from the assigned two unique codes for encoding its data packets.

Among these spreading code protocols introduced above, only T protocol and T-R protocol are collision-free if the users in the network send data to each other randomly. Because the T-R protocol has the drawback of using a large amount of spreading codes and complicated decoding scheme, T protocol is preferred in the proposed CDMA NoC. However, if T protocol is applied in the network, the receiver can not choose the proper spreading code for decoding because it can not know who is sending data to it. In order to solve this problem, an Arbiter-Based T protocol (A-T protocol) is proposed for the CDMA NoC. In a CDMA network which applies A-T protocol, each user is allocated with a unique spreading code for data transfer. When a user wants to send data to the other user, it will send the destination information of data to the arbiter before starting data transfer. Then the arbiter will inform the receiver to prepare the corresponding spreading code of the sender for data decoding. After the sender receives the acknowledge signal from the arbiter, it will start the data transfer by using its unique spreading code. If there are more than one user who want to send data to a same receiver, the arbiter will grant only one sender to send data at a time. Therefore, transfer conflicts in the proposed CDMA NoC which uses A-T protocol can be avoided.

3. The Proposed CDMA NoC Structure

The proposed CDMA NoC is a packet-switched network which consists of 'Network Node', 'CDMA transmitter', and 'Network Arbiter' blocks as illustrated in Fig.3. The functional IP blocks (functional hosts) in the system are connected with the on-chip CDMA network through individual network node blocks. The CDMA communications in the network are performed by 'CDMA Transmitter' and 'Network Arbiter' blocks. VCI or OCP standard [13] is applied as the interface standard between a functional host and a network node. As the different functional hosts may work with different clock frequencies, coordinating the communications among different clock domains would be a problem when integrating all functional hosts into one network.

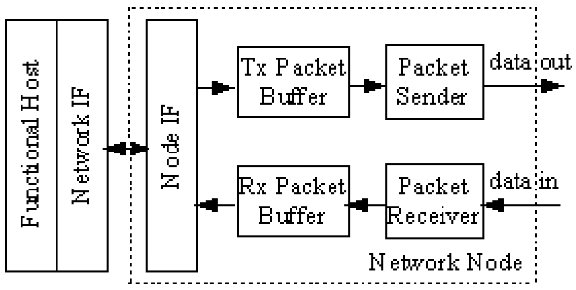


Figure 4. Block Diagram of Network Node

Globally-Asynchronous Locally-Synchronous (GALS) scheme [14] was proposed as a solution for this problem. The meaning of GALS scheme in the proposed NoC is that the communications between each functional host and its network node use local clock frequency, while the communications between network nodes through the CDMA network are asynchronous. In order to support GALS scheme, both synchronous and asynchronous circuits are used in the proposed design. The three components of the proposed CDMA NoC will be presented in Section 3.1 and 3.2 with more details.

3.1. Network Node

The block diagram of 'Network Node' is illustrated in Fig.4 where the arrows represent the flows of data packets. Because the interface standard, VCI or OCP, and the functional host both work in synchronous manner, the 'Node IF' sub-block applies synchronous design, whereas the other sub-blocks in 'Network Node' use asynchronous design to implement GALS scheme in the network. The function of the sub-blocks in 'Network Node' will be described in the following paragraphs.

1) 'Node IF': This sub-block is the interface block which applies the same interface standard, VCI or OCP standard, as the 'Network IF' interface block of functional host. 'Node IF' sub-block is used to assemble the data from functional host into packet format and send the packet to 'Tx Packet Buffer', or deassemble the received packet from 'Rx Packet Buffer' and send the extracted data to the functional host.

2) 'Tx/Rx Packet Buffer': These two sub-blocks are buffers which consist of the asynchronous FIFO proposed in [15]. 'Tx Packet Buffer' is used to store the data packets from 'Node IF' sub-block, and then deliver the packets to 'Packet Sender' sub-block, while the 'Rx Packet Buffer' delivers the packets from 'Packet Receiver' to 'Node IF'.

3) 'Packet Sender': If 'Tx Packet Buffer' is not empty, 'Packet Sender' will fetch the data packet from the buffer by asynchronous handshake protocol. Then it will extract the destination information from the received packet and send the destination address to 'Network Arbiter'. After 'Packet Sender' gets the grant signal from arbiter, it will start to send data packet to 'CDMA Transmitter'.

4) 'Packet Receiver': After system reset, this sub-block will wait the sender information from 'Network Arbiter' to select the spreading code for decoding. After the spreading code for decoding is set, the receiver will start to receive and decode the data from 'CDMA Transmitter', and then send the decoded data to 'Rx Packet Buffer' in packet format.

Because the 'Network Node' block in the CDMA network need not handle any bypass packets, the 'Network Node' block presented in this paper has less complexity by comparing with the 'Network Node' block presented in [16] for a point-to-point connection network. This is another benefit of applying CDMA technique into an on-chip network.

3.2. CDMA Transmitter and Network Arbiter

The 'CDMA Transmitter' block takes care of receiving data packets from network nodes and encoding the data to be transferred with the corresponding unique spreading code of the sender node. Although this block is implemented by asynchronous circuits, it applies synchronous CDMA transfer scheme which means that the data from different nodes will be encoded and transmitted synchronously. The synchronous CDMA scheme can avoid the interferences caused by the phase offsets among the orthogonal spreading codes if the data from different nodes are encoded and transmitted asynchronously with each other. The data encoding and transfer processes for different network nodes are performed in parallel and independently in 'CDMA Transmitter'. However, because the nodes in the proposed CDMA network can request data transfer randomly, 'CDMA Transmitter' applies 'first come, first served' mechanism to ensure the data encoding and transfer are performed in synchronous manner. For example, if network node 'A' and 'B' assert data transfer requests to 'CDMA Transmitter' simultaneously and node 'C' asserts the request later than 'A' and 'B', the transmitter will encode and transfer the data from node 'A' and 'B' at the same time, and then will start to do the encoding and transferring for node 'C' after the data transfer for 'A' and 'B' are completed. The synchronization of encoding and transferring in 'CDMA Transmitter' are controlled by the four-phase handshake control signals in the asynchronous circuits.

'Network Arbiter' block is the core component to implement the A-T spreading code protocol proposed in Section 2.3. Every sender node can not send data packet to 'CDMA Transmitter' until it gets the grant signal from 'Network Arbiter'. Namely, 'Network Arbiter' takes charge of setting up data transfer channel between sender node and receiver node. In case that there are more than one sender node requesting to send data to the same receiver node, the arbiter will also apply 'first come, first served' principle to guarantee that there is only one sender sending data to one specific receiver at a time. The reason of this limitation is that the 'Packet Receiver' of 'Network Node' block can receive and decode data from only one sender at a time. However, if different sender nodes request to send data to different receiver nodes, these requests will be handled independently and in parallel. Therefore, the arbitration scheme applied in 'Network Arbiter' is distributed arbitration other than the centralized arbitration applied in the conventional bus.

4. Synthesis and Simulation

In order to support GALS scheme in the proposed CDMA NoC, only 'Node IF' sub-block in network node uses synchronous circuits, the other components use asynchronous circuits. Both the synchronous and asynchronous designs in the proposed CDMA NoC are implemented in RTL using VHDL in order to suit the conventional synchronous design tools and flow. The method of designing asynchronous circuits in RTL using VHDL was presented in [15]. The basic principle is to construct the asynchronous circuits with C-element, latches, and combinational logic gates, and then describe the RTL structure using VHDL.

4.1. Synthesis Results

The components of the proposed CDMA NoC have been synthesized using a 0.18μm standard cell library. The data width and buffer depth in 'Network Node' are set to 32 bits and 4 respectively. The area costs (without wire area) of components of the CDMA NoC are listed in Table 1. From Table 1 we can see that the 'Tx/Rx Packet Buffer' takes a large portion in the total design costs. Therefore, it needs to compromise between buffering ability and area cost.

4.2. Simulation Network Setup

A network which applies the proposed CDMA NoC structure is built for simulation purpose. The simulation network illustrated in Fig.5 contains six network nodes. Six functional hosts work in different clock domains as presented in Fig.5. Three hosts act as masters and the other three act as slaves, as denoted by the labels 'M' and

Table 1. Area Cost of CDMA NoC Components

Block Name		Area (μm^2)
Network Node	Node IF	18825.2
	Tx/Rx Packet Buffer	71778.3
	Packet Sender	17707.0
	Packet Receiver	23253.0
CDMA Transmitter		10338.3
Network Arbiter		17686.5

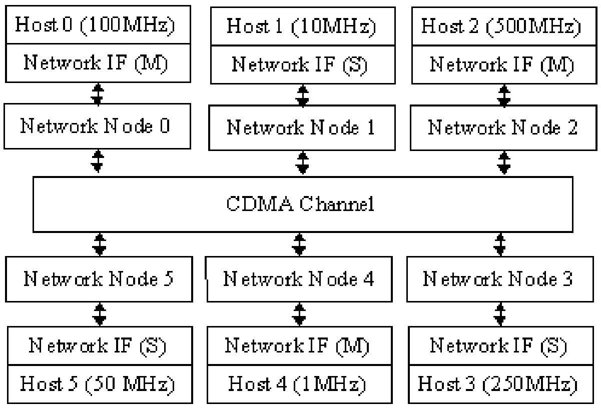


Figure 5. Six-Node CDMA On-Chip Network

'S' in the 'Network IF' blocks. The network nodes are connected to each other through a 'CDMA Channel' composed by 'CDMA Transmitter' and 'Network Arbiter' blocks. The spreading codes used in the network are six 8-bit Walsh codes. The interface standard applied in the network is Basic VCI (BVCI) [13] standard. The basic data transfer unit in the simulation network is a data packet composed of data cells. The number of data cells in a packet varies from two to four, while the width of each data cell is fixed at 32 bits. The data transactions performed in the simulation are listed in Table 2. Each data transaction consists of one request packet from a master host to a slave host and one corresponding response packet from the slave host to the master host.

4.3. Simulation Results

The simulation was performed in gate-level. Because GALS scheme is applied in the network, the data transfer latency in the simulation network is separated into two parts, Synchronous Transfer Latency (STL) and Asynchronous Transfer Latency (ATL). The STL refers to the data transfer latency between a functional host and the network node attached to it. STL depends on the local clock and the type of interface. The measured values of STL are listed in Table 3. The constant values in Table 3 are caused by the handshakes in the asynchronous domain, and they are independent on the local clock rate but belong to the synchronous transfer processes.

The ATL refers to the data transfer latency of transferring data packets from one network node to the other node through the CDMA channel by asynchronous circuits. The ATL consists of three parameters: Packet Loading Latency (PLL), Packet Transfer Latency (PTL), and Packet Storing Latency (PSL). The concept of those ATL parameters is illustrated in Fig.6 with an example where 'Network Node 0' sends one data packet to 'Network Node 1'. The black arrows in Fig.6 represent the packet transfer direction. The portions of the transfer used to measure the different parameters of latency are marked by grey arrows in Fig.6 and explained as below:

Table 2. Data Transaction Specification

Master Node	Slave Node	Number of Transactions	Packet Length	
			Request Packet	Response Packet
Node 0	Node 1	2	4, 3	2, 3
	Node 3	2	3, 4	3, 2
	Node 5	3	3, 4, 3	3, 2, 3
Node 2	Node 1	4	3, 4, 3, 4	3, 2, 3, 2
	Node 3	1	3	3
	Node 5	2	4, 3	2, 3
Node 4	Node 1	2	3, 4	3, 2
	Node 3	3	4, 3, 4	2, 3, 2
	Node 5	1	4	2

Table 3. Synchronous Transfer Latency

Interface Type	Latency of sending data to 'Network Node'	Latency of receiving data from 'Network Node'
BVCI Master	8 local clock cycles + 2.5 ns	8 local clock cycles + 3.2 ns
BVCI Slave	4 local clock cycles + 2.5 ns	4 local clock cycles + 3.1 ns

Table 4. Asynchronous Transfer Latency Parameters

Packet Length	PLL (ns)	PTL (ns)	PSL (ns)
2 data cells	5.7	384.6	5.5
3 data cells	5.7	768.9	5.5
4 data cells	5.7	1153.7	5.5

1) Packet Load Latency (PLL): This is the time used by 'Packet Sender' sub-block in network node to fetch one data packet from 'Tx Packet Buffer' and prepare to send the data packet to 'CDMA Transmitter'.

2) Packet Transfer Latency (PTL): This latency refers to the time used to transfer one data packet from the 'Packet Sender' of sending node to the 'Packet Receiver' of receiving node through the CDMA channel using a four-phase handshake protocol.

3) Packet Storing Latency (PSL): After the receiving node receives a data packet, it will spend a certain time to store the received data packet into 'Rx Packet Buffer'. This time duration is measured as PSL.

The measured values of ATL parameters are listed in Table 4. The listed latency values only include the logic gate delay of the circuits, no wire delay is considered. More accurate latency values could be obtained by adding the wire delay after layout. In Table 4, we can see that PTL increases as the packet length increases. This is because the data cells in a packet are sent in a serial manner between 'CDMA Transmitter' and 'Packet Receiver', so that more data cells need more transmission time. The reason that PLL and PSL are not affected by the packet length is that the data cells in a packet are loaded or stored in a parallel manner. By comparing with point-to-point connection on-chip network presented in [16], the ATL value in the CDMA network is larger although it is a constant value. For example, according to values in Table 4, if a 2-data-cell packet is transferred in the

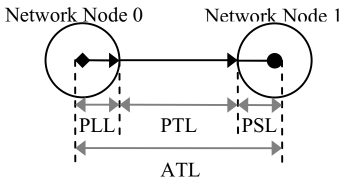


Figure 6. Asynchronous Transfer Latency Parameters

proposed CDMA network, the ATL will equal to the ATL of transfer a same size packet through 19 intermediate nodes in the network presented in [16]. The reason of the large ATL value in the CDMA network is that each bit of the data to be transferred is spreaded into L bits, where L is the length of the spreading code, then the encoded data are transferred serially. However, the data bits are transferred directly in the point-to-point network in parallel. Therefore, the ATL value of CDMA network can be reduced by transferring the encoded bits in parallel.

5. Conclusions

An on-chip communication network which applies CDMA technique and supports GALS communication scheme was presented. The proposed CDMA network uses asynchronous circuits to perform the global data transfers between network nodes, and synchronous circuits to deal with the local data transfers between a functional host and the network node attached to it. Both the asynchronous and synchronous circuits of the proposed network are implemented in RTL using VHDL to suit the synchronous design tools naturally. A six-node CDMA network was constructed and synthesized for simulation purpose. The simulation reveals that the data can be correctly transferred in parallel in time domain by using CDMA technique in an on-chip communication network implemented totally by digital circuits. By comparing with a point-to-point connection network which shares the communication media in time domain, the proposed CDMA NoC has a constant data transfer latency value by sharing the communication media in code domain.

References

[1] J. A. J. Leijten, J. L. van Meerbergen, A. H. Timmer, and J. A. G. Jess; “PROPHID: A Data-Driven Multi-Processor Architecture for High- Performance DSP”; *Proceedings of the 1997 European Design & Test Conference*, Mar. 1997.

[2] P. Guerrier, and A. Greiner; “A Generic Architecture for On-Chip Packet-Switched Interconnections”; *Proceedings of the Design, Automation and Test in Europe Conference 2000*, Mar. 2000.

[3] D. Sigüenza-Tortosa, T. Ahonen, and J. Nurmi; “Issues in the Development of a Practical NoC: the Proteo Concept”; *Integration, the VLSI jounal*, Volume 38, Issue 1; 2004.

[4] M. K. Simon, J. K. Omura, R. A. Scholtz, and B. K. Levitt; “Spread Spectrum Communicalions”; MD: *Computer Science*, 3 vols. Rockville, 1984.

[5] R. Yoshimura, T. B. Keat, T. Ogawa, S. Hatanaka, T. Matsuoka, and K. Taniguchi; “DS-CDMA wired bus with simple interconnection topology for parallel processing system LSIs”; *Digest of Technical Papers of IEEE International Solid-State Circuits Conference, 2000*; Feb. 2000.

[6] T. B. Keat, R. Yoshimura, T. Matsuoka, and K. Taniguchi; “A novel dynamically programmable arithmetic array using code division multiple access bus”; *Proceedings of the 8th IEEE International Conference on Electronics, Circuits and Systems, 2001,* Volume 2; Sept. 2001.

[7] S. Shimizu, T. Matsuoka, and K. Taniguchi; “Parallel bus systems using code-division multiple access technique”; *Proceedings of the 2003 International Symposium on Circuits and Systems, 2003,* Volume 2; May 2003.

[8] M. Takahashi, T. B. Keat, H. Iwamura, T. Matsuoka, and K. Taniguchi; “A study of robustness and coupling-noise immunity on simultaneous data transfer CDMA bus interface”; *Proceedings of IEEE International Symposium on Circuits and Systems, 2002,* Volume 4; May 2002.

[9] R. H. Bell, Jr., K. Y. Chang, L. John, and E. E. Swartzlander, Jr.; “CDMA as a multiprocessor interconnect strategy”; *Conference Record of the Thirty-Fifth Asilomar Conference on Signals, Systems and Computers, 2001,* Volume 2; Nov. 2001.

[10] E. H. Dinan, and B. Jabbari; “Spreading codes for direct sequence CDMA and wideband CDMA cellular networks”; *IEEE Communications Magazine,* Volume 36, Issue 9; Sep. 1998.

[11] E. S. Sousa, and J. A. Silvester; “Spreading code protocols for distributed spread-spectrum packet radio networks”; *IEEE Transactions on Communications,* Volume 36, Issue 3; Mar. 1988.

[12] D. D. Lin, and T. J. Lim; “Subspace-based active user identification for a collision-free slotted ad hoc network”; *IEEE Transactions on Communications*, Volume 52, Issue 4; Apr. 2004.

[13] VSI Alliance; *Virtual Component Interface Standard v 2*; April 2001. OCP-IP Association; *Open Core Protocol Specification*; 2001.

[14] D. M. Chapiro; “Globally-Asynchronous Locally-Synchronous Systems”;*PhD thesis*, Stanford University; Oct. 1984.

[15] X. Wang, T. Ahonen, and J. Nurmi; “A Synthesizable RTL Design of Asynchronous FIFO”; *Proceedings of 2004 International Symposium on System-on-Chip*; Finland, Nov. 2004.

[16] X. Wang, D. Sigüenza-Tortosa, T. Ahonen, and J. Nurmi; “Asynchronous Network Node Design for Network-on-Chip”; *Proceedings of 2005 International Symposium on Signal, Circuits, and System*; Romania, Jul. 2005.

PUBLICATION 4

X. Wang, T. Ahonen, and J. Nurmi, “Prototyping A Globally Asynchronous Locally Synchronous Network-on-Chip On A Conventional FPGA Device Using Synchronous Design Tools”, in Proceedings of the 2006 International Conference on Field Programmable Logic and Applications, (FPL 2006), pages 657-662, Madrid, Spain, August 2006.

PROTOTYPING A GLOBALLY ASYNCHRONOUS LOCALLY SYNCHRONOUS NETWORK-ON-CHIP ON A CONVENTIONAL FPGA DEVICE USING SYNCHRONOUS DESIGN TOOLS

Xin Wang, Tapani Ahonen, Jari Nurmi

Institute of Digital and Computer Systems
Tampere University of Technology
Korkeakoulunkatu 1, Tampere, Finland
email: {xin.wang, tapani.ahonen, jari.nurmi}@tut.fi

ABSTRACT

An FPGA prototype of a four-node globally-asynchronous locally-synchronous network-on-chip is described. The network for global communication operates asynchronously at the link level and synchronously within a node. Two C-element control pipelines constitute the control logic for the asynchronous part. C-element and asynchronous arbiter realizations on FPGA using standard synchronous design tools are presented.

1. INTRODUCTION

As the complexity of System-on-Chip (SoC) is increasing, the communications among the large number of components in a SoC become more and more complicated. Network-on-Chip (NoC) has been proposed to separate the concern of communications with the concern of computations in a SoC, namely, NoC only handle the communications in an on-chip system. In a complex SoC, it is very common that different functional components work with different clock frequencies. Therefore, a NoC scheme should have the ability to handle the multiple-clock-domain communication problem. Globally-Asynchronous Locally-Synchronous (GALS) scheme [1] has been proposed as a solution of this problem, and many NoC architectures [2-4] based on GALS scheme have been presented. However, only ASIC implementations of those proposed GALS NoC have been reported. As the cost of ASIC implementation is getting higher, FPGA prototyping supplies a fast and cheap way of verifying a NoC design.

The FPGA devices available on the market have been oriented only for prototyping synchronous circuits. This type of FPGA device will be referred as conventional FPGA in this paper. Hence, a main challenge of prototyping a GALS NoC design on a conventional FPGA is how to prototype asynchronous circuits. There are two approaches to conquer this challenge. The first approach is to develop a specific FPGA structure for prototyping asynchronous circuits. Some specific FPGA structures [5-7] for asynchronous circuits have been presented. However, those asynchronous FPGA structures exclude synchronous circuits out of the considerations, and they are not available on the market. Therefore, we come to the second approach which is prototyping asynchronous circuits on a conventional FPGA. Few presented works [8-10] belong to this approach. In [8], a GALS system which applies stoppable clock and port-controller to connect different clock domains is presented, hence, it concerns more about prototyping particular asynchronous components rather than asynchronous NoC. The works in [9, 10] presented a methodology of prototyping a GALS SoC on a conventional FPGA device. However, the presented methodology involves a special synthesis tool for asynchronous circuits and a netlist-format component library in the design flow. Therefore, in order to suit the synchronous design tools and flow of conventional FPGA more smoothly, both the synchronous and asynchronous circuits in the proposed GALS NoC have been designed using VHDL. A C-element structure and an asynchronous arbiter which suit for asynchronous circuits' prototyping are presented. The prototyping work in this paper presents a way of prototyping a synchronous-asynchronous mixed design onto a conventional FPGA by the commonly used synchronous design tools and flow.

The paper is organized as follows. Section 2 presents the GALS NoC structure to be prototyped in this work. The asynchronous design in a network node and the control logics for the asynchronous circuits are addressed in Section 3. In Section 4, the C-element structure and the arbiter for prototyping the NoC and the prototyping method are presented. The conclusion is drawn in Section 5.

2. THE GALS NOC

The prototyped GALS NoC is an asynchronous version of the Proteo NoC presented in [11]. Proteo NoC [3] is a packet-switched on-chip communication network developed for performing the complex communication tasks in a SoC. In Proteo NoC, each functional Intellectual Property (IP) block (Functional Host) is connected to the on-chip network through a corresponding network node. Then the network is built by connecting the network nodes

1-4244-0 312-X/06/$20.00 ©2006 IEEE.

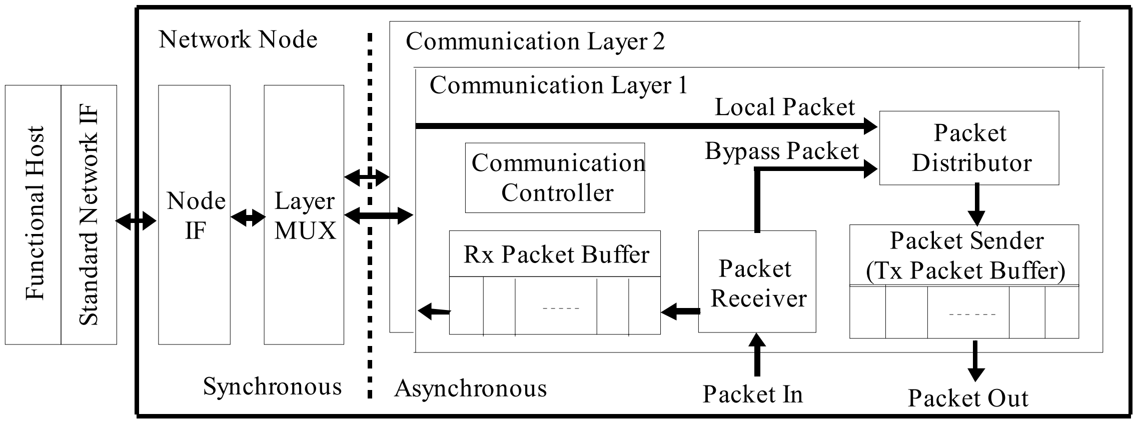


Fig. 1. Network Node Block Diagram

together with any topologies which suit the application requirements. A Proteo network node [11] which can support GALS scheme is illustrated in Fig.1. The network node consists of 'Node IF', 'Layer MUX', and 'Communication Layer' blocks. The two blocks outside of the network node illustrate how a 'Functional Host' block is connected with a network node through a standard network interface ('Standard Network IF') block. In Proteo NoC, the applied interface standards include VCI [12] and OCP [13]. As illustrated in Fig.1, GALS scheme in the on-chip network is supported by applying both synchronous and asynchronous designs in each network node. The synchronous blocks, 'Node IF' and 'Layer MUX', are used to communicate with locally synchronous 'Functional Host' in the system, while the asynchronous blocks are used to perform asynchronous communications among the network nodes. The bold arrows in Fig.1 demonstrate the data packet flow in a network node. The functions of each block in a network node are shortly presented in the following three paragraphs.

1) 'Node IF'. This block is responsible for assembling the data from 'Functional Host' into data packets or transferring the data packets from network node to 'Functional Host' block according to the interface standard applied in 'Standard Network IF' block.

2) 'Layer MUX'. As the name of this block indicated, this block behaves as a multiplexer used to connect 'Node IF' block with a certain 'Communication Layer' block during the data transfers between network node and 'Functional Host' block.

3) 'Communication Layer'. The function of this block is to perform the globally asynchronous communication with other network nodes through a handshake protocol. As illustrated in Fig.1, there can be more than one 'Communication Layer' block in a network node. With multiple 'Communication Layer' blocks, a network node can connect with multiple other network nodes in Proteo NoC. There are four sub-blocks in the 'Communication Layer' block. The 'Packet Receiver' sub-block is used to receive data packets from another network node. If the destination of the received packet is the current network node, the packet is called 'incoming packet', and it will be stored in 'Rx Packet Buffer'. Otherwise, the received packet is called 'bypass packet', and it will be dispatched into 'Packet Distributor/Sender' for further transferring. The 'Communication Controller' sub-block in Fig.1 represents the controller which takes charge of the necessary arbitrations and communication controls.

A four-node bidirectional ring network which consists of the network node illustrated in Fig.1 is used for the prototyping work in this paper. In the four-node network, two nodes are BVCI slave [12] type and two nodes are BVCI master [12] type. No any 'Functional Host' blocks are included in the prototyping work. Therefore, only a GALS communication network, not a GALS system, is considered in this work.

3. ASYNCHRONOUS NETWORK NODE DESIGN

The presented GALS on-chip network in Section 2 is built by connecting the multiple network nodes illustrated in Fig.1 together. Therefore, the major challenge of the GALS NoC design is the asynchronous design in each network node. The control logic structures and data path used in the asynchronous blocks of a network node will be addressed in this section.

The asynchronous design in the network node illustrated in Fig.1 can be divided into two parts which are data path and control logic part. The data path represented by the black arrows in Fig.1 applies four-phase dual-rail protocol in order to transfer data in a delay-insensitive manner. The control logics used in a network node include the Finite State Machine (FSM) in the 'Communication Controller' block and the block controls in 'Packet Receiver', 'Packet Distributor', and 'Packet Sender' blocks illustrated in Fig.1. The FSM in the 'Communication Controller' block takes care of the processes of receiving, sending or storing data packets in a network node by triggering the block control

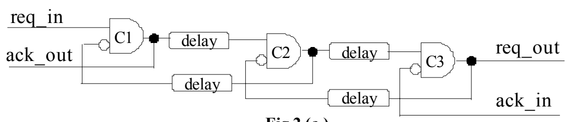


Fig.2 (a)

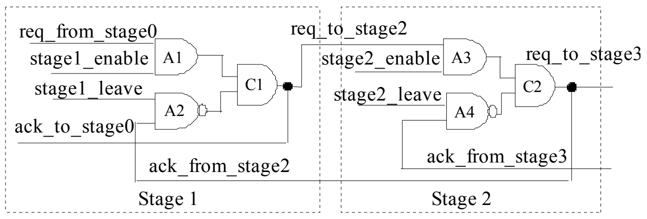


Fig.2 (b)

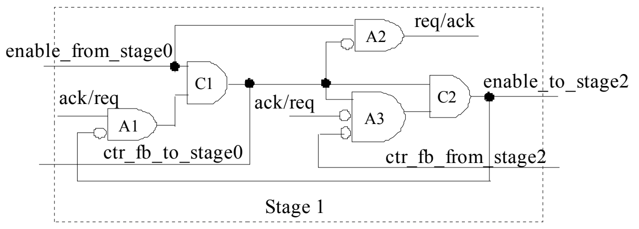


Fig.2 (c)

Fig. 2. Four-phase C-element Control Pipelines

logics in the corresponding function blocks with a four-phase handshake protocol. The block control logics take charge of moving the data packets in or out of the individual blocks through four-phase dual-rail protocol. For example, the control logic in 'Packet Sender' is used to control the process of storing the incoming packets into the 'Tx Buffer' or sending out the packets to the other network node via dual-rail protocol. In order to ensure the control logics operate correctly, the delay-insensitive (DI) or quasi-delay-independent (QDI) model is preferred for the control logic structure. A delay-insensitive control logic for micropipeline has been presented in [14] and illustrated in Fig.2(a). The principle of micropipeline control logic is to use the output from the next stage to enable or disable the output of current stage. Two C-element control pipelines based on the micropipeline control logic are designed as the control logics in the network node.

Two stages of the C-element control pipeline used in 'Communication Controller' block to build the FSM are illustrated in Fig.2(b). Each stage of the pipeline represents a state element of the FSM. In Fig.2(b), we can see that the FSM uses micropipeline control logic as the backbone and applies few AND gates as the delay components illustrated in Fig.2(a), hence, it is delay-insensitive. The state information of the FSM is passed through each stage in the pipeline by a four-phase handshake protocol. If we take the 'Stage 1' illustrated in Fig.2(b) as an example, when both the 'req_from_stage0' and 'stag1_enable' signals are '1', the output of 'C1' will be set to logic '1' which indicates that the current state of the FSM is in the 'Stage 1'. Then the output of 'C1' can be used as a request signal to trigger the control logics in the corresponding function blocks for a certain communication process.

The C-element control pipeline structure illustrated in Fig.2(c) is used in the 'Packet Receiver', 'Packet Distributor', and 'Packet Sender' blocks as the block control logic to generate four-phase request or acknowledge signals for data transfers. Each stage of the control pipeline is composed by two C-elements as illustrated in Fig.2(c). The 'C1' is used to record the rising edge of a request or acknowledge signal, while the 'C2' is used to record the falling edge of a request or acknowledge signal. Therefore, each stage of the block control pipeline will pass the enable signal to the next stage only after the four-phase handshake process on the current stage has done. Although the presented block control pipeline structure can only meet QDI model because the input 'ack/req' signal is branched to 'A1' and 'A3', the timing requirement for distributing the 'ack/req' input signal along the isochronic wire forks is quite loose since the logic delays in 'A1' and 'C1' are usually much larger than the logic delay of the inverter at the input of 'A3'.

Besides the control logics, asynchronous arbiters are also needed in the network node to allocate the shared resource to only one user at a time. For example, 'Communication Controller' block needs an arbiter to decide that either the 'local packet' or 'bypass packet' will be transferred by the 'Packet Sender' first if they come to the 'Packet Distributor' simultaneously. The arbiter structure used in this work will be discussed with more details in Section 4.2.

Another issue about the asynchronous design of the network node is how to transfer the signals from asynchronous domain to synchronous domain safely. In this design, a normal double-latching scheme [15] is applied to sample the signals from asynchronous domain by the 'Layer Switch' block illustrated in Fig.1.

Finally, by applying the presented pipeline control logics, arbiters, and other data path logics, the asynchronous design in the network node has been done and combined with the synchronous design through the double-latching scheme. Thus, the presented four-node GALS NoC design is ready for the prototyping work.

4. PROTOTYPING THE GALS NOC ON A CONVENTIONAL FPGA

Look-Up-Table (LUT) based FPGA device has been widely used for prototyping synchronous designs because of its flexible and reprogrammable features. However, if a LUT based conventional FPGA device is used to prototype asynchronous circuits, several limitations which have been addressed in [5, 16] are summarized as follows.

1) LUT cannot guarantee hazard-free because the routing delays and input change patterns are unpredictable.

2) The timing requirements of asynchronous circuits are difficult to guarantee in a conventional FPGA.

3) The commonly used arbitration structures in

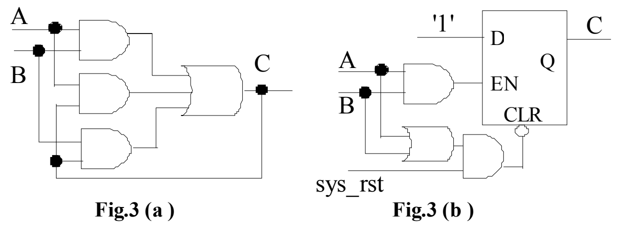


Fig. 3. C-element Structure for FPGA

asynchronous circuits are usually designed in transistor level. Therefore, they are not supported by LUT based FPGA.

As discussed in Section 1, the LUT based conventional FPGA is the only choice for prototyping the presented GALS NoC although some limitations have been addressed. In order to circumvent the addressed limitations, the C-element and arbiter structures which suit for this purpose will be presented in Section 4.1 and Section 4.2. The method of prototyping the presented four-node GALS NoC will be addressed in Section 4.3.

4.1. A C-element Structure for FPGAs

The Reed-Müller C-element is a basic component for asynchronous circuits, which is normally implemented in transistor level. In order to map asynchronous circuits on a conventional FPGA, an equivalent two-input C-element structure illustrated in Fig.3(a) has been presented in [9, 10]. It has been proved to be logic hazard-free under the single-bit input change assumption and certain two-input change patterns. However, it needs a netlist-format description as a component library in the design flow to ensure the feedback path illustrated in Fig.3(a) is mapped on a LUT correctly. In order to avoid the explicit feedback path, another two-input C-element structure is proposed and illustrated in Fig.3(b). The C-element illustrated in Fig.3(b) bases on a D-latch which uses ‘A AND B’ as the enable (‘EN’) signal and ‘A OR B’ as the reset signal (‘CLR’). The data input port (‘D’) of the D-latch is attached to logic ‘1’ constantly. Because the explicit feedback path is avoided in the D-latch-based C-element structure, the special netlist format component library in the design flow can be removed. Therefore, the C-element structure illustrated in Fig.3(b) suits the design flow of conventional FPGA better. The idea of using latch to map a C-element in LUT has already been presented in [9] where a RS-latch is suggested. Whereas, the C-element structure based on D-latch in Fig.3(b) is more safe than the suggested RS-latch structure because there is no data switching at the data input port ‘D’.

In Fig.3(b), we can see that the C-element structure is hazard-free under one-input change assumption by applying the ‘AND’ gate and ‘OR’ gate at the ‘EN’ port and ‘CLR’ port respectively. For certain two-input switch patterns,

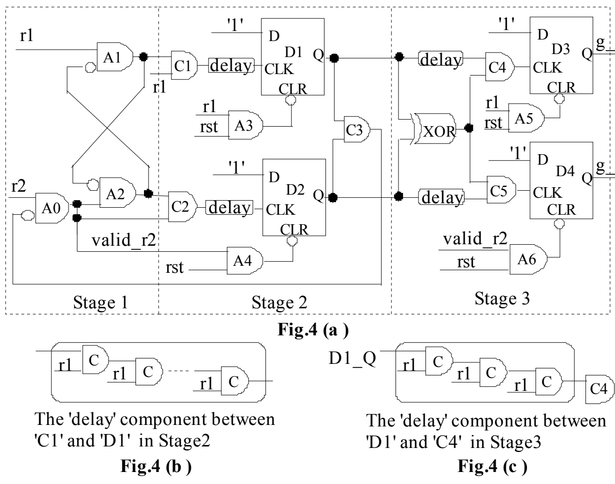


Fig. 4. Asynchronous Arbiter Structure

00→11 and 11→00, the structure in Fig.3(b) is also hazard-free. Whereas, the input switch patterns, 01→10 and 10→01, are not allowed because they may produce a logic error which depends on the wire delay. Because all the C-elements in the GALS NoC design are used to follow a request or acknowledge signal of four-phase handshake protocol, as the control logics illustrated in Fig.2, there are no 01→10 or 10→01 input switch patterns for the C-elements in the design. Thus, the proposed C-element structure is safely to be used in this prototyping work.

4.2. An Asynchronous Arbiter Realization on FPGA

Asynchronous arbiter is a component used to allocate a shared resource among multiple users to only one user at a time. For ASIC implementation, cross-coupled NAND gates are used as the simplest arbiter structure. For implementing arbiter on a conventional FPGA, the built-in Flip-Flop is suggested to use in order to minimize the metastability effects [17]. An asynchronous arbiter structure which uses the built-in Flip-Flop of conventional FPGA has been presented in [17]. It uses a clock signal which comes from the outside of the arbiter block to drive the built-in Flip-Flop. Hence, it needs an extra clock generation circuit presented in [17].

In order to take advantage of the simplicity of the cross-coupled NAND gates arbiter structure and avoid using extra clock generation circuit, a two-input fixed-priority arbiter structure is designed and illustrated in Fig.4(a). It applies a revised cross-coupled NAND gates structure and the built-in Flip-Flop of a conventional FPGA. The arbiter illustrated in Fig.4(a) can be divided into three stages.

The first stage consists of two cross-coupled AND gates, ‘A1’ and ‘A2’, with inverted inputs. The ‘A0’ gate is used to disable the input ‘r2’ when a conflict between ‘r1’ and ‘r2’ is detected at the output of C-element ‘C3’. When the combinational logics of ‘Stage1’ illustrated in Fig.4(a) are modeled using VHDL and synthesized by a tool for the conventional FPGA, they will be implemented by LUTs instead of the AND gates. Therefore, the feedback delays in the LUTs for 'Stage1' are much larger than the ASIC implementation which uses standard cells. Hence, when two input requests 'r1' and 'r2' appear simultaneously or very close to each other, the LUT implementation of 'Stage1' will enter into an oscillation state instead of the metastability state as the ASIC implementation. In this situation, the second stage of the arbiter is used to filter out the possible oscillation outputs from 'Stage1'.

The 'Stage2' illustrated in Fig.4(a) bases on two built-in D Flip-Flop (D-FF) registers of a conventional FPGA. The C-elements, 'C1' and 'C2', are used to convert the oscillation outputs from 'Stage1' into a single $0 \rightarrow 1$ signal transition which is used as the clock signal to trigger the registers 'D1' and 'D2' respectively. The 'A3' and 'A4' gates are used to generate reset signals for 'D1' and 'D2'. After passing through 'Stage2' of the arbiter, the oscillation outputs from 'Stage1' may trigger the outputs of both 'D1' and 'D2' into logic '1'. In this case, the 'C3' will detect this confliction and disable the 'r2' request by the feedback path from the output of 'C3' to the input of 'A0'. The 'delay' components in 'Stage2' as illustrated in Fig.4(a) are used to ensure that the rising edge from the outputs of 'C1' and 'C2' will arrive after the 'CLR' signals from 'A3' and 'A4'. An exemplar structure of the 'delay' components is presented in Fig.4(b) where the number of C-element depends on the timing character of the built-in D-FF.

The actual arbitration process is taken place in the 'Stage 3' where another two built-in D-FF registers are used. When a request conflict is detected at the outputs of 'D1' and 'D2', the 'XOR' logic in 'Stage3' will close the arbiter output by disabling 'C4' and 'C5'. The arbitration outputs will be enabled only after the output of 'D2' is cleared by the feedback from 'C3'. Therefore, request 'r1' has a higher priority in the presented arbiter. The 'delay' components in 'Stage3' are used to filter out the possible glitch from 'XOR' when the output signals of 'D1' and 'D2' did not reach the inputs of the 'XOR' gate simultaneously in a request-conflict situation. One exemplar structure of the 'delay' components is illustrated in Fig.4(c).

Although the presented arbiter works under QDI model since the isochronic wire forks for distributing 'r1' and 'valid_r2' signals are needed, the timing requirement for those isochronic forks is loosened a lot by the logic delays and 'delay' components in the design. In the presented arbiter, the data input ports 'D' of the built-in D-FF registers are attached to logic '1' constantly in order to enhance the operation stability of the D-FF register. The presented asynchronous arbiter consumes 45 Adaptive LUTs (ALUTs) [19] on a StratixII device when six C-elements are used in its 'delay' components illustrated in Fig.4(b) and Fig.4(c).

4.3. Prototyping The Four-Node GALS NoC

By applying the presented C-element and arbiter structure, the asynchronous design of the network node discussed in Section 3 can be modeled using VHDL in a hierarchy manner. Namely, the C-element structure presented in Fig.3(b) is modeled using VHDL as a component. Then, any other logics, such as the arbiter or the control logics, use the C-elements as component instances in their own VHDL files. In a same manner, the control logic, arbiter, and C-element are used by a higher level asynchronous block as component instances in their VHDL descriptions. The synchronous blocks in the network node are also designed with VHDL. Therefore, both the synchronous and asynchronous designs in the presented GALS NoC apply a same design input format which suits the design tools for conventional FPGA. The design tool and the conventional FPGA used in this work are QuartusII and Altera StratixII respectively.

In order to prevent the synthesis tool to mix all the combinational logics from different components together, each instance of the C-element and arbiter in the design is set as a design partition. The higher level components or blocks are also be set as a unique partition according to the design hierarchy. During the synthesis process, each partition of the design will be synthesized separately from each other by QuartusII. Therefore, the proposed C-element and arbiter structures will be generated correctly.

After synthesis the entire design with the hierarchical partition method, the next issue is to meet the QDI timing requirements of the proposed arbiter and block control pipeline structures during the placement and routing process. LogicLock technique [18] is applied for this purpose. A LogicLock region is set to each arbiter and block control logic pipeline in the design so that the components in a same arbiter or control pipeline will be automatically placed into one Logic Array Block (LAB) [19] or the adjacent LABs of StratixII device by QuartusII. Thus, the fast intra-connects inside a LAB and inter-connects between adjacent LABs [19] can meet the loose timing requirements of the arbiter and the block control pipeline addressed respectively in Section 4.2 and Section 3.

Finally, the prototyping method with Quartus II can be summarized as follows.

1) Step1: Describe both synchronous and asynchronous parts in the hierarchical manner using VHDL.

2) Step2: Define a design partition for each component.

3) Step3: Set a LogicLock region for all delay sensitive arbiter and control blocks.

4) Step4: Run synthesis, placement, and routing steps without additional constraint files.

By using the proposed prototyping method, the four-node GALS NoC presented in Section 2 are realized on a StratixII device. The four-phase dual-rail protocol is used in the asynchronous data transfers between network nodes. The whole network utilizes 41,674 ALUTs on the StratixII

Table 1. ALUTs Utilization of 'Network Node' Blocks

Sub-blocks		Utilized ALUTs
Node IF (BVCI Slave Type)		146
Node IF (BVCI Master Type)		399
Layer MUX		142
Communication Layer (CL block)	Communication Controller	238
	Packet Distributor	451
	Packet Sender +Tx Packet Buffer	2,202
	Packet Receiver	233
	Rx Packet Buffer	1,878
Total (with 2 CL blocks)	BVCI Slave Type Network Node	10,292
	BVCI Master Type Network Node	10,545

device. The utilized ALUTs by the two different types of 'Network Node' blocks and their sub-blocks are listed in Table 1.

The gate-level simulation reveals that the prototyped four-node GALS network can deliver a four-cell packet between two adjacent network nodes with 632.78 ns by a four-phase dual-rail handshake protocol. Each data cell in a packet has 32 data bits. Thus, the asynchronous transfer speed of the prototyped GALS network in a StratixII device is equivalent to 202.24 kb/s in theory.

5. CONCLUSION

Conventional FPGA devices and design tools are oriented towards synchronous design. The presented work applied the synchronous design tools of Quartus II to realize a synchronous-asynchronous mixed design on a Stratix II device.

We described a four-node on-chip network prototype that applies both synchronous and asynchronous designs. Two C-element control pipelines were designed for the control of the asynchronous parts. A C-element structure and an asynchronous arbiter suited for realization on a conventional FPGA were presented.

6. REFERENCES

[1] D. M. Chapiro, "Globally-Asynchronous Locally-Synchronous Systems", *PhD thesis*, Stanford University, Oct. 1984.

[2] P. Zipf, H. Hinkelmann, A. Ashraf, and M. Glesner, "Networks-on-chip: A switch architecture and signal synchronization for GALS system-on-chips", *Proceedings of the 17th symposium on Integrated circuits and system design*, Sep. 2004.

[3] D. Sigüenza-Tortosa, T. Ahonen, and J. Nurmi, "Issues in the Development of a Practical NoC: the Proteo Concept", *Integration, the VLSI jounal*, volume 38, issue 1, 2004.

[4] A. Lines,"Nexus: an asynchronous crossbar interconnect for synchronous system-on-chip designs",; *Proceedings of 11th Symposium on High Performance Interconnects*, pp.2–9, Aug. 2003.

[5] S. Hauck, S. Burns, G. Borriello, and C. Ebeling, "An FPGA for implementing asynchronous circuits", *IEEE Design & Test of Computers*, vol. 11, issue 3, pp. 60, Fall 1994.

[6] C. Traver, R. B. Reese, and M. A. Thornton, "Cell designs for self-timed FPGAs", *Proceedings. 14th Annual IEEE International ASIC/SOC Conference*, pp.175-179, Sep. 2001.

[7] N. Huot, H. Dubreuil, L. Fesquet, and M. Renaudin, "FPGA architecture for multi-style asynchronous logic", *Proceedings of Design, Automation and Test in Europe 2005*, vol. 1, pp. 32-33, 2005.

[8] M. Najibi, K. Saleh, M. Naderi, H. Pedram, and M. Sedighi, "Prototyping globally asynchronous locally synchronous circuits on commercial synchronous FPGAs", *Proceedings of 16th IEEE International Workshop on Rapid System Prototyping*, pp.63-69, Jun. 2005.

[9] Q. T. Ho, J. B. Rigaud, L. Fesquet, M. Renaudin, and R. Rolland, "Implementing Asynchronous Circuits on LUT Based FPGAs", *Proceedings of 12th International Conference on Field-Programmable Logic and Applications*, Sep. 2002.

[10] J. Quartana, S. Renane, A. Baixas, L. Fesquet, and M. Renaudin, "Gals systems prototyping using multiclock fpgas and asynchronous network-on-chips", *Proceedings of 2005 International Conference on Field Programmable Logic and Applications*, pp. 299-304, Aug. 2005.

[11] X. Wang, D. Sigüenza-Tortosa, T. Ahonen, and J. Nurmi; "Asynchronous Network Node Design for Network-on-Chip", *Proceedings of 2005 International Symposium on Signal, Circuits, and System*; Jul. 2005.

[12] VSI Alliance. *Virtual Component Interface Standard version 2*, April 2001. http://www.vsi.org

[13] OCP-IP Association. *Open Core Protocol Specification*, 2001. http://www.ocpip.org.

[14] I. E. Sutherland, "Micropipelines", *Communications of the ACM*, vol. 32, no.6, pp. 720-738, Jun. 1989.

[15] J. U. Horstmann, H. W. Eichel, and R. L. Coates, **"**Metastability behavior of CMOS ASIC flip-flops in theory and test", *IEEE Journal of Solid-State Circuits*, vol. 24, issue1, pp.146 – 157, Feb. 1989.

[16] R. Payne, "Asynchronous FPGA architectures", *IEE Proceedings of Computers and Digital Techniques*, vol. 143, issue 5, pp.282-286, Sep. 1996.

[17] S. W. Moore, and P. Robinson, "Rapid Prototyping of Self-Timed Circuits", *International Conference on Computer Design*, pp 360--365, Oct. 1998.

[18] Altera, "Quartus II Version 5.1 Handbook", vol.2, pp.1009-1049, Dec. 2005.

[19] Altera, "Stratix II Device Handbook", vol.1, pp.23-47, Dec. 2005.

PUBLICATION 5

X. Wang, and J. Nurmi, “A RTL Asynchronous FIFO Design Using Modified Micropipeline”, in Proceedings of the 10th Biennial Baltic Electronics Conference, (BEC 2006), pages 95-98, Tallinn, Estonia, October 2006.

A RTL Asynchronous FIFO Design Using Modified Micropipeline

Xin Wang, Jari Nurmi

Institute of Digital and Computer Systems, Tampere University of Technology
P.O.Box 553, FIN-33101, Tampere, Finland
E-mail: {xin.wang, jari.nurmi}@tut.fi

ABSTRACT: An asynchronous FIFO which applies four-phase handshake protocol to read or write data has been designed in Register-Transfer Level (RTL) using VHDL. The asynchronous FIFO in this paper avoids data movement in a flow-through FIFO by applying token passing scheme in its control pipelines and multiplexer in its data register bank. Two control pipelines which base on micropipeline structure are proposed and used as the control logic for the asynchronous FIFO. An asynchronous arbiter and C-element RTL structures used in the proposed asynchronous FIFO are also presented.

1 Introduction

Asynchronous FIFO is an important component for building the packet buffers in a packet-switched on-chip network which applies Globally-Asynchronous Locally-Synchronous (GALS) [1] scheme. A network node structure for building GALS on-chip network has been presented in [2]. The network node applies asynchronous handshake protocol for globally data packet transfers. Therefore, it needs efficient asynchronous FIFOs to buffer the input or output data packets of the network node. Although the asynchronous FIFO presented in this paper is designed under an on-chip network application background, it is a general purpose asynchronous FIFO which can be used in other applications.

The asynchronous FIFO designs which have been presented can be roughly classified into two categories in terms of data movement. The first category includes the flow-through FIFOs [3, 4] which base on micropipeline structure [3]. Those FIFOs make the data flow through all data cells in the FIFO before reaching the output port. Thus, the latency is poor although the throughput can be high. The asynchronous FIFOs in another category use counter control logic [5], or token passing and common data bus structure [6, 7] to avoid the data movement inside the FIFOs. Therefore, the latency caused by data movement in a flow-through FIFO is eliminated. However, the compensation is the high complexity of the control logic. The asynchronous FIFO presented in this paper belongs to the second category in which the data would not flow through the data cells, while the control complexity is lowered by using a modified micropipeline structure to pass around the read or write token in the FIFO. Another advantage of the presented FIFO design is that it is designed in RTL by using commonly adopted hardware description language VHDL. Therefore, it can be easily integrated with other synchronous RTL designs to build a large synchronous-asynchronous mixed system, such as a GALS on-chip network, and it also suits for the commonly used synchronous design flow and tools.

The following sections of this paper are organized as follows. In Section 2, the asynchronous FIFO structure and the modified micropipeline control logic are presented. Section 3 describes the RTL model of C-element and the arbiter used in this asynchronous FIFO. The synthesis and simulation results are presented in Section 4. Finally, the conclusions are drawn in Section 5.

2 The Asynchronous FIFO Structure

The proposed asynchronous FIFO works with four-phase handshake protocol. By changing the interface descriptions, it can support bundled-data and dual-rail protocols. For the simplicity, an asynchronous FIFO which uses four-phase bundled-data protocol is illustrated in Fig.1. There are two main functional blocks in the FIFO, 'Control Logic' block and 'Data Bank' block, as illustrated in Fig.1. The 'control logic' block consists of the control pipelines derived from micropipeline structure. Each cell in the control pipeline is used to control read or write operation of a data cell in 'Data Bank' Block. The details of 'Control Logic' block and 'Data Bank' block will be presented in Section 2.1 and Section 2.2 respectively.

2.1 The 'Control Logic' Block

A micropipeline control logic [3] is illustrated in Fig.2 (a). The principle of micropipeline control logic is to use

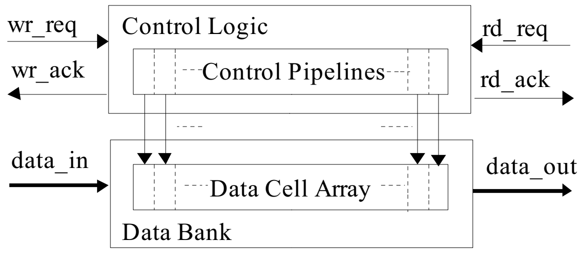


Fig.1 Asynchronous FIFO Block Diagram

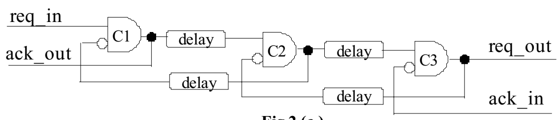


Fig.2 (a)

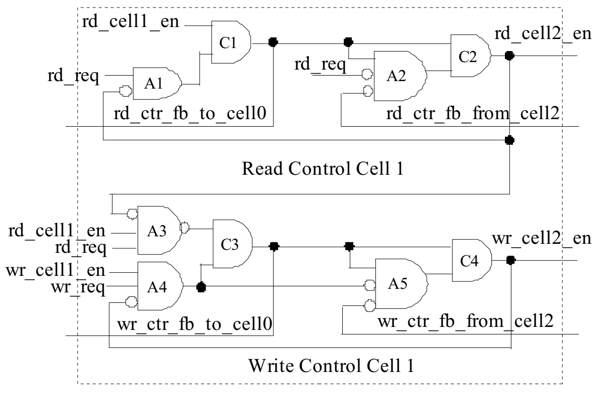


Fig.2 (b)

Fig.2 Asynchronous FIFO Control Pipelines

the output from the next stage to enable or disable the output of current stage, therefore, the request event at the input port will be delivered to the output port in a pipelined manner. The 'Control Logic' block in the presented FIFO design consists of two control pipelines which base on the micropipeline control logic. One pipeline is dedicated to read control while another is for write control. Each pipeline is composed of a series of control cells. Fig.2 (b) illustrates two single cells from the two control pipelines respectively and the signaling between the two control cells.

By comparing the two structures in Fig.2 (a) and (b), we can see that the modified control pipelines in Fig.2 (b) use micropipeline control logic as the backbone and apply few AND gates as the delays illustrated in Fig.2 (a). Each cell of the modified control pipelines consists of two C-elements. The first C-element, 'C1' in read control cell and 'C3' in write control cell, is used to record the rising edge of request signal. The second C-element, 'C2' and 'C4' in Fig.2 (b), is used to record the falling edge of request signal. Therefore, each cell of the control pipeline illustrated in Fig.2 (b) will pass the enable signal as a token to the next cell only after the four-phase handshake process on the current cell has done. The token signals in the current control cells, such as 'rd_cell1_en' and 'wr_cell1_en' signals in Fig.2 (b), are used to enable the corresponding data cell to respond to the current read or write operation. After the completion of current data operation, the token will be delivered to the next control cell. When the read or write token reaches the last cell of the control pipeline, it will be passed back to the first control cell if the corresponding data cell is ready for read or write operation.

The feedback signal from 'C2' to 'A3' in Fig.2 (b) is used to synchronize the status information between read and write control cells. For example, in Fig.2 (b), if the output of 'C2' is '1', it means that the data in the corresponding data cell has been read. Therefore the output of 'A3' is set to '1' which indicates that the empty data cell is enabled for future write operation. The other two input signals of 'A3', 'rd_cell1_en' and 'rd_req', are used to reset the write control cell after the corresponding data cell has been read, so that the write control cell will have room for receiving the write token in the next round.

The control pipelines illustrated in Fig.2 (b) require that the read and write requests should not appear at the same time. The arbiter used to solve this read/write conflict problem in the presented FIFO will be discussed in Section 3.1.

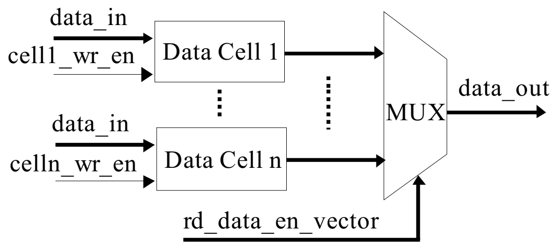


Fig.3 Data Bank Block Diagram

2.2 The 'Data Bank' Block

The structure of the 'Data Bank' Block illustrated in Fig.3 is composed of a multiplexer and multiple 'Data Cell' blocks. Each 'Data Cell' block consists of a set of latches. The number of latches in a 'Data Cell' depends on the data width of the asynchronous FIFO. The main function of the 'Data Bank' Block is to latch the incoming data or output the requested data by the control signals from the 'Control Logic' Block. For example, the 'cell1_wr_en' signal comes from the first write control cell is used to enable write operation on the 'Data Cell 1' block in Fig.3. The 'rd_data_en_vector' in Fig.3 is the combination of read enable signals from the read control pipeline. It is used to select the corresponding data cell for the current read operation.

3 RTL Structures of Asynchronous Arbiter and C-Element

3.1 RTL Structure of Asynchronous Arbiter

As mentioned in Section 2.1, the proposed control pipelines of the asynchronous FIFO can handle only one request at a time, therefore, an arbiter is needed to grant either read or write request to take effect at one time.

For ASIC implementation, cross-coupled NAND gates illustrated in Fig.4 (b) are used as the simplest arbiter structure. However, if two requests appear simultaneously, the NAND structure will enter into metastability state. Four-input NOR gates illustrated in Fig.4 (b) are suggested in [8] to filter the metastability because they have high threshold voltage. However, this scheme needs to be designed in gate level with certain standard-cell library. If it is described in RTL using VHDL, the four inputs of the NOR gates will be simplified by the synthesis tool because they are from a same signal node.

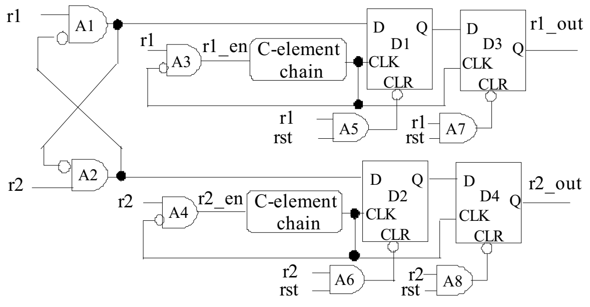


Fig.4 (a)

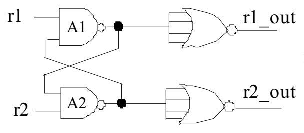


Fig.4 (b)

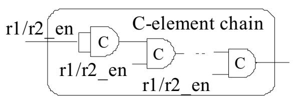


Fig.4 (c)

Fig.4 Asynchronous Arbiter Structure

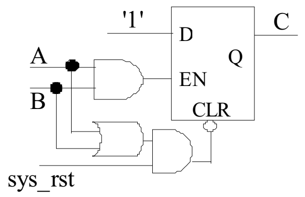


Fig.5 C-element Structure

Therefore, it does not suit our purpose of modeling the asynchronous FIFO in RTL using VHDL. Hence, an arbiter structure illustrated in Fig.4 (a) is proposed for the presented asynchronous FIFO design. The proposed arbiter applies the cross-coupled AND gates derived from the cross-coupled NAND structure as the basic structure and uses double-latching scheme to filter out metastability. The clock signals for sampling the outputs from the cross-coupled AND gates are generated by a delay line composed of a chain of C-elements. The C-element chain illustrated in Fig.4 (c) uses the input requests as the trigger signal to generate a clock signal at a certain frequency until the qualified request is sampled successfully. The sampling clock cycle depends on the gate and wire delays in the C-element chain and gives the cross-coupled AND gates a certain amount of time to recover from metastability before sampling. An advantage of using the C-element chain to generate sampling clock is that the sampling frequency can be easily adjusted according to the target technology library by changing the number of C-elements. The mean time between failures (MTBF) of double-latching scheme can be guaranteed to be long enough by using low sampling frequency [9].

3.2 RTL Structure of C-Element

The Müller C-element is a basic component widely used in the presented asynchronous FIFO. It is normally implemented in transistor level. In order to model the presented FIFO totally in RTL using VHDL, a RTL two-input C-element structure is proposed in Fig.5. The proposed C-element bases on a D-latch which uses

Table 1. Area and Power Consumptions

FIFO Types	*Area (μm^2)*	*Dynamic Power (mW)*
Synchronous FIFO	21032.96 (100%)	3.84 (100%)
Asynchronous FIFO	10821.63 (51.5%)	2.08 (54.2%)

Table 2 Timing Characteristics of the Asynchronous FIFO

	Ack Rise Latency (ns)	*Req Hold Time (ns)*	*Ack Fall Latency (ns)*	*Handshake Cycle (ns)*
Write Request	4.2	0.1	0.8	5.1
Read Request	4.5	0.1	0.9	5.5

'A AND B' as the enable ('EN') signal and 'A OR B' as the reset signal ('CLR'). The data input port ('D') of the D-latch is attached to logic '1' constantly. The idea of using a latch to build a C-element has already been presented in [10] where a RS-latch is suggested to use. Whereas, the D-latch C-element structure in Fig.5 is safer than the suggested RS-latch C-element structure because it avoids data switching at the data input port 'D'.

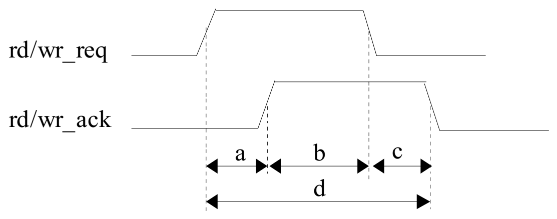
4 Synthesis and Simulation Results

4.1 Synthesis Result

After all the main components, C-element, control pipelines, and the arbiter, have been built in RTL, the asynchronous FIFO design presented in Section 2 are constructed in RTL with VHDL. Thus, the presented asynchronous FIFO design suits for the synchronous design flow and tools naturally and can be easily integrated with other synchronous designs.

For the purpose of comparison, a reference synchronous FIFO which has very similar structure has been designed also in RTL using VHDL. The reference synchronous FIFO has the same top level structure with the asynchronous FIFO illustrated in Fig.1. The differences are that the synchronous FIFO applies clocked four-phase handshake protocol to read or write data, and a Finite State Machine (FSM) and a few counters are used as the control logic to record the location of current read or write token.

Both the presented asynchronous FIFO and the reference synchronous FIFO have been synthesized by a same synthesis tool and 0.18 µm technology library. The data width and FIFO depth for both synchronous and asynchronous FIFO designs are set to 32 bits and four cells respectively. In the asynchronous FIFO design, five C-elements are used in the C-element chain of the arbiter to generate the sampling clock signal. The area and dynamic power consumption after synthesis are listed in Table1. The dynamic power consumptions listed in Table1 are the estimated value reported by the synthesis tool without any switching activity back-annotation.



a: Ack Rise Latency c: Ack Fall Latency

b: Req Hold Time d: Handshake Cycle

Fig.6 Asynchronous FIFO Latency Parameters

In Table1, we can see that the presented asynchronous FIFO design has smaller dynamic power consumption and area even though it has a large overhead of the C-element chain in its arbiter.

4.2 Simulation Result

A gate-level simulation of the synthesized asynchronous FIFO has been performed by an event-driven simulation tool. The latency of the asynchronous FIFO is measured as the time between the rising edge of 'rd/wr_req' signal and the falling edge of the acknowledge signal. This latency is independent on the depth of the asynchronous FIFO because the data in the presented asynchronous FIFO are not moved in the FIFO after stored. The meanings of measured timing parameters of a four-phase handshake process during the simulation are illustrated in Fig.6 and the values are listed in Table2. To be noticed is that the 'Req Hold Time' is decided by the environment other than the FIFO itself. A 0.1 ns delay is used in this simulation.

According to the timing of 'Handshake Cycle' listed in Table2, the presented asynchronous FIFO can perform 94.3 million 32-bit data read-after-write operations per second, which is equivalent to 3.01 Gb/s throughput in theory.

5 Conclusions

A RTL asynchronous FIFO design which suits the conventional synchronous design flow and tools has been presented. The presented asynchronous FIFO is mainly composed of control logic and data bank blocks. The presented FIFO in this paper avoids data movement in a flow-through FIFO by applying token passing scheme in its control pipelines and applying multiplexer in its data register bank. Two control pipelines which base on micropipeline structure have been proposed and used in this design. The RTL structures of asynchronous arbiter and C-element which suit for VHDL modeling have been presented. By comparing with a reference synchronous FIFO design, the proposed asynchronous FIFO has smaller area and dynamic power consumption. When a 0.18μm technology library is used for implementation, the throughput of the presented asynchronous FIFO can reach 3.01 Gb/s in theory.

References

[1] D. M. Chapiro, "Globally-Asynchronous Locally-Synchronous Systems", PhD thesis, Stanford University, October 1984.

[2] X. Wang, D. Sigüenza-Tortosa, T. Ahonen, and J. Nurmi, "Asynchronous Network Node Design for Network-on-Chip", Proceedings of 2005 International Symposium on Signal, Circuits, and System, July 2005.

[3] I.E. Sutherland, "Micropipelines", Communications of the ACM, vol. 32, no.6, pp. 720-738, June 1989.

[4] E. Brunvand, "Low latency self-timed flow-through FIFOs", Proceedings of Sixteenth Conference on Advanced Research in VLSI, pp.76 – 90, March 1995.

[5] A.V. Yakovlev, A.M. Koelmans, and L. Lavagno, "High-Level Modeling and Design of Asynchronous Interface Logic", IEEE Design and Test of Computers, Spring 1995.

[6] T. Chelcea, and S.M. Nowick, "Low-latency asynchronous FIFO's using token rings", Proceedings of Sixth International Symposium on Advanced Research in Asynchronous Circuits and Systems, pp. 210 – 220, April 2000.

[7] K.K. Yi, "The Design of a Self–Timed Low Power FIFO Using a Word–Slice Structure", M.Phil Thesis, Univ. of Manchester, September 1998.

[8] J. Kessels and P. Marston, "Designing asynchronous standby circuits for a low-power pager", Proceedings of International Symposium on Advanced Research in Asynchronous Circuits and Systems, pp. 268–278, April 1997.

[9] J.U. Horstmann, H.W. Eichel, and R.L. Coates, "Metastability behavior of CMOS ASIC flip-flops in theory and test"; IEEE Journal of Solid-State Circuits, Volume: 24 , Issue: 1 , pp.146 – 157, February 1989.

[10] Q. T. Ho, J. B. Rigaud, L. Fesquet, M. Renaudin, and R. Rolland, "Implementing Asynchronous Circuits on LUT Based FPGAs", Proceedings of 12th International Conference on Field-Programmable Logic and Applications, September 2002.

PUBLICATION 6

X. Wang, and J. Nurmi, “Comparison of a Ring On-Chip Network and a Code-Division Multiple-Access On-Chip Network”, VLSI Design, Special Issue on Networks-on-Chip, Volume 2007, Article ID 18372, 14 pages, Hindawi Publishing Corporation, April 2007.

Hindawi Publishing Corporation
VLSI Design
Volume 2007, Article ID 18372, 14 pages
doi:10.1155/2007/18372

Research Article

Comparison of a Ring On-Chip Network and a Code-Division Multiple-Access On-Chip Network

Xin Wang and Jari Nurmi

Institute of Digital and Computer Systems, Tampere University of Technology, 33101 Tampere, Finland

Received 5 October 2006; Accepted 6 February 2007

Recommended by Shashi Kumar

Two network-on-chip (NoC) designs are examined and compared in this paper. One design applies a bidirectional ring connection scheme, while the other design applies a code-division multiple-access (CDMA) connection scheme. Both of the designs apply globally asynchronous locally synchronous (GALS) scheme in order to deal with the issue of transferring data in a multiple-clock-domain environment of an on-chip system. The two NoC designs are compared with each other by their network structures, data transfer principles, network node structures, and their asynchronous designs. Both the synchronous and the asynchronous designs of the two on-chip networks are realized using a hardware-description language (HDL) in order to make the entire designs suit the commonly used synchronous design tools and flow. The performance estimation and comparison of the two NoC designs which are based on the HDL realizations are addressed. By comparing the two NoC designs, the advantages and disadvantages of applying direct connection and CDMA connection schemes in an on-chip communication network are discussed.

1. INTRODUCTION

As the technology feature size of integrated circuit fabrication is continuously shrinking down in the deep submicron regime, the number of components which can be integrated into an on-chip system is getting larger and larger. Therefore, the communications among the large number of components in a system-on-chip (SoC) are challenging tasks to deal with. Network-on-chip addresses this communication issue in an on-chip system by separating the concerns of communication from the concerns of computation. It means that the communication issue in an SoC is abstracted and handled by an on-chip communication network which hides the detailed information about how the communications are performed. Therefore, a system designer can pay more attention on the functions of system components and system integration by treating the NoC as a component of an on-chip system.

The NoC structures which have been proposed can be roughly classified into two categories, circuit-switched network and packet-switched network, in terms of the way of using the communication media. PROPHID architecture [1] is an example of a circuit-switched network which connects the terminals in the network by allocating them a set of time or space slices on the communication links. Examples in the packet-switched category are SPIN [2] and Proteo NoC [3]. SPIN network applies fat-tree topology and router blocks to transfer data packets from source node to destination node. In Proteo NoC, the components in the system are connected through network nodes and hubs. The network topology and connections in Proteo NoC can be customized and optimized for a specific application. Since an on-chip system can contain hundreds of functional intellectual property (IP) blocks in the near future, the circuit-switched network will face the problem of scalability and parallelism in that situation. Therefore, a packet-switched scheme is a better choice for future NoC designs because its structure is scalable and its data transfers are performed in parallel by sharing the communication media among multiple network nodes in a time-division manner.

As the number of IP blocks in an SoC is increasing, it is natural that different functional blocks work with different clock frequencies in an SoC. Hence, data transfer among multiple clock domains is another issue that needs to be handled by an on-chip network. A globally asynchronous locally synchronous (GALS) scheme [4] has been proposed to solve this problem of SoC designs. For an NoC, GALS means that data transfers between each functional IP block and its attached network node are synchronous, whereas data transfers between network nodes are asynchronous.

From the analysis addressed in the previous two paragraphs, we can see that the GALS packet-switched scheme is a promising direction to explore the NoC designs for future on-chip systems. A common and natural way of composing a GALS packet-switched on-chip network is to connect two network nodes with a direct link. This way of connecting network nodes will be referred to as point-to-point (PTP) connection in this paper. Different patterns of the connection links in the network form different network topologies. For example, a mesh topology is applied in the NoC design presented in [5]. In a PTP connection NoC, the routing scheme and router architecture, such as the router presented in Æthereal NoC [6], are very important factors for supplying guaranteed services because the packet transfer latency may vary largely when data packets are transferred to different destinations or to the same destination via different routes in the network.

In order to eliminate the variance of the data transfer latency and complexity incurred by routing in a PTP connection NoC, a connection scheme which applies code-division multiple-access technique has been briefly introduced in [7].

By separating the different data from different users in the code domain, the data transfer latency in the CDMA NoC is stabilized by enabling multiple users to use the communication media parallel in time domain.

In order to examine the advantages and disadvantages of both the PTP connection scheme and the CDMA connection scheme, a bidirectional ring NoC design and a CDMA NoC design developed in our institute will be addressed and compared in terms of the network structure, data transfer principle, network node design, asynchronous design, and performance. For the sake of abbreviation, the bidirectional ring NoC design will be referred to as the PTP NoC in this paper.

The following sections of this paper will be arranged as follows. The network structures of the PTP NoC design and the CDMA NoC design will be presented and compared in Section 2. In Section 3, the data transfer principles of the two NoC designs will be studied and compared. Then the different network node structures in the two NoC designs will be addressed and compared in Section 4. Section 5 will present the asynchronous designs applied in both the PTP NoC and the CDMA NoC designs. In Section 6, two simulation networks of the two NoC designs built up for performance estimation will be presented. Then the performance comparison of the two NoC designs will be addressed upon the simulation results. Finally, the conclusion will be drawn in Section 7.

2. THE NETWORK STRUCTURES

2.1. The network structures of the two NoC designs

The PTP NoC design examined in this paper is based on a network node design [8] proposed for implementing GALS scheme in Proteo NoC. The network structure of the PTP NoC is illustrated in Figure 1. From Figure 1, we can see that

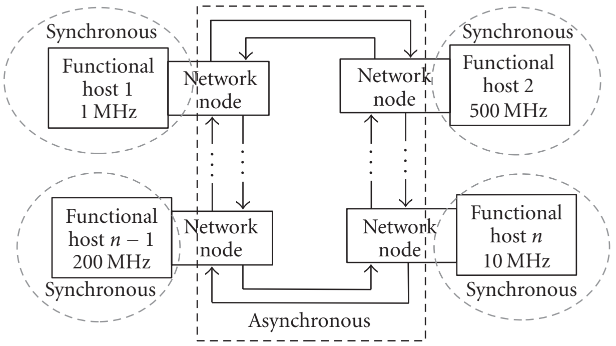


FIGURE 1: The bidirectional ring NoC structure.

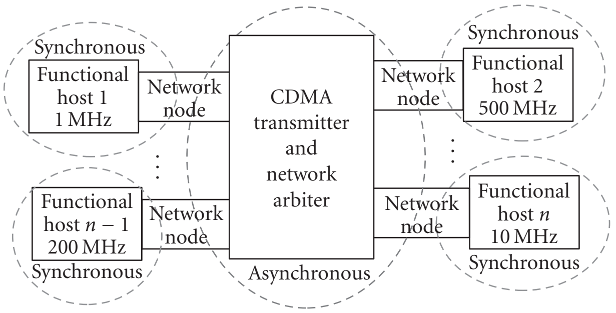


FIGURE 2: The CDMA NoC structure.

the communication between a "functional host" (functional IP block) and its network node is synchronous, while the data transfers among network nodes are preformed in asynchronous manner. For a large network, it may be necessary to break the entire network into sections and use some bridge nodes or hub nodes as addressed in [3] to connect the network sections together, whereas these bridges or hubs can be seen as one type of network nodes. Therefore, as illustrated in Figure 1, the PTP NoC can be composed simply by connecting the network nodes together with direct links.

The network structure of the CDMA NoC design introduced in [7] is illustrated in Figure 2. In the CDMA NoC, the GALS scheme is applied in the same way as in the PTP NoC; however, the network nodes in the CDMA NoC are no longer connected to each other with direct links. A "CDMA transmitter" and a "network arbiter" blocks are introduced in the CDMA NoC. All the network nodes need to communicate with the "CDMA transmitter" and "network arbiter" blocks directly. The functionality of the "CDMA transmitter" and "network arbiter" blocks will be addressed thoroughly in Section 3. With a direct comparison, we can see that the structure of the CDMA NoC is more complex than the PTP NoC since extra blocks are introduced in the CDMA NoC.

2.2. Distributed traffic versus centralized traffic

After a direct comparison of the two network structures in terms of simplicity, we can take a further analysis of the effects of the two different network structures on the features of the networks. In the PTP NoC as illustrated in Figure 1, the data traffic load is distributed into the links among the network nodes. This distributed data traffic scheme has the merits of flexibility and scalability, whereas the main disadvantage of the PTP connection is that the data transfer latency between two network nodes can be largely different because the data may be transferred through different routes or because of data traffic congestions in the network. Therefore, the main concern of designing a PTP NoC is to find out the optimal topology and use all kinds of routing and flow-control methods to guarantee a high throughput and low transfer latency.

In the CDMA NoC illustrated in Figure 2, a centralized data transfer scheme is applied since all network nodes communicate with the "CDMA transmitter" and the "network arbiter" blocks directly. This centralized data transfer scheme is different from the conventional bus structures since it can supply parallel data transfers both in time and space domains by applying CDMA technique, whereas a bus structure supplies data transfer service among users in a time-division manner. The advantage of the centralized scheme applied in the CDMA NoC is that the data transfer latency between network nodes is a stable value. This stable transfer latency is contributed by the feature of parallel data transfer in time domain and the universal link distance among all network nodes. With the stable data transfer latency, the communication quality in the CDMA NoC will not vary.

3. THE DATA TRANSFER PRINCIPLES

The fundamental reason of the different network structures between the PTP NoC and the CDMA NoC is the different data transfer principles applied in the two NoC designs. Thus, the data transfer principles of the two NoC designs will be addressed and compared in this section.

3.1. *Data transfer principle in the PTP NoC*

As presented in Section 2.1, the PTP NoC is built by connecting the network nodes with direct links. The reason of this simplicity of connecting the network nodes is that the data are transferred in their original form in the PTP NoC. The only operation on the data is to encapsulate them into a packet format. This operation is done by a network node after getting the data from its attached functional host block. The packet format used in the PTP NoC is illustrated in Figure 3. After the data packet is formed, the PTP NoC will transfer the data bits with their original values to their destination through the links to the other nodes. Therefore, direct links between the network nodes in the PTP NoC are enough to handle the data transfers.

3.2. *Data transfer principle in the CDMA NoC*

As indicated by the name, the CDMA NoC applies CDMA technique to perform data transfers in the NoC. The basic principle of CDMA technique is illustrated in Figure 4. At the sending end, the data from different senders are encoded using a set of orthogonal spreading codes. Then the

Destination node ID	Source node ID (optional for CDMA NoC)	(Other fields)
Data cell 1		
Data cell 2(optional)		
Data cell 3(optional)		

FIGURE 3: Packet format specification.

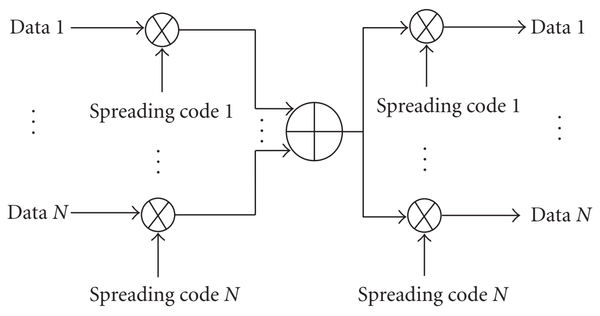


FIGURE 4: CDMA technique principle.

encoded data from different senders are added together for transmission without interfering with each other because of the orthogonal property of spreading codes. The orthogonal property means that the normalized autocorrelation of the spreading codes is 1, while the cross-correlation of the spreading codes is 0. Therefore, at the receiving end, the data can be decoded from the received sum signals by multiplying the received signals with the corresponding spreading code used for encoding. The data packet format applied in the CDMA NoC is the same format as illustrated in Figure 3. The issues related with the data encoding/decoding and transfer principles in the CDMA NoC will be addressed with details in the following subsections.

3.2.1. *Data encoding and decoding schemes*

Some CDMA encoding and decoding schemes for on-chip communication implemented by analog circuits have been proposed [9–11]. In those schemes, the encoded data are represented by the continuous voltage or capacitance value of the circuits. Therefore, the data transfers in the analog circuits are challenged by the coupling noise, clock skew, and the variations of capacitance and resistance caused by circuit implementation [11]. In order to avoid the challenges faced by the analog circuit implementation, digital encoding and decoding schemes are developed for the CDMA NoC and are illustrated in Figures 5 and 7, respectively. In the presented encoding scheme, data from different senders are fed into the encoding function bit by bit. Each data bit will be spread into S bits by multiplying it with a unique S-bit spreading code. The multiplications are performed by XOR logic gates as illustrated in Figure 5. Each bit of the S-bit encoded data generated by XOR operations is called a data chip. Then the data chips which come from different senders are added

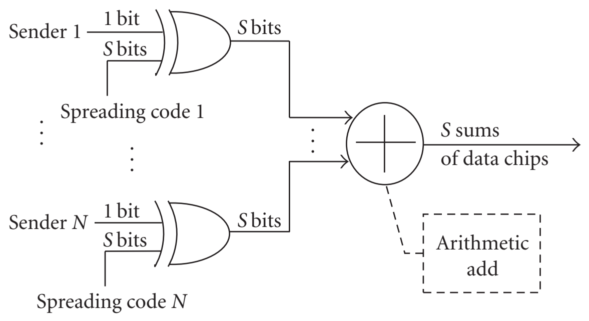


Figure 5: Digital CDMA encoding scheme.

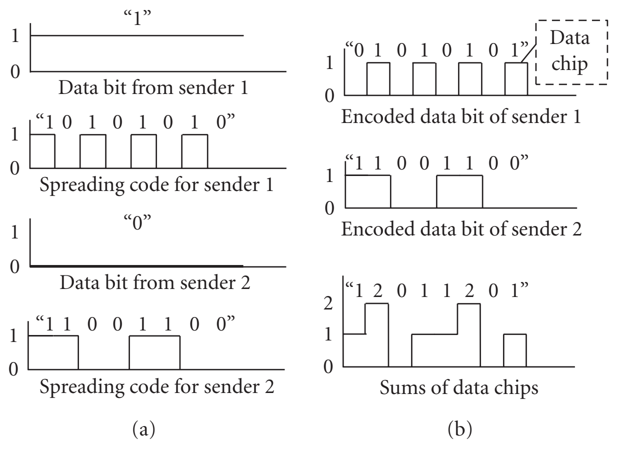


Figure 6: Data encoding example.

together arithmetically according to their positions in the S-bit sequences. Namely, all the first data chips from different senders are added together, and all the second data chips from different senders are added together, and so on. Therefore, after the add operations, we will get S sum values of S-bit encoded data. Finally, as proposed in [12], binary equivalents of the S sum values are transferred to the receiving end one by one. An example of encoding two data bits from two senders is illustrated in Figure 6 in order to explain the proposed encoding scheme more specifically. Figure 6(a) shows two original data bits from different senders and two 8-bit spreading codes. The top two figures in Figure 6(b) illustrate the results after data encoding (XOR operations) for the original data bits. The bottom figure in Figure 6(b) presents the 8 sum values after adding operations. Then the binary equivalents of each sum value will be transferred to the receiving end. In this case, two binary bits are enough to represent the three possible decimal sum values, "0," "1," and "2." Hence, for example, if a decimal sum value "2" needs to be transferred, we need to transfer two binary digits "10."

The digital decoding scheme used in the CDMA NoC is illustrated in Figure 7. The decoding scheme accumulates the received sum values into two separated parts, a positive part and a negative part, according to the bit value of the spreading code used for decoding. For instance, as illustrated in Figure 7, the received first sum value will be put into the positive accumulator if the first bit of the spreading code for decoding is "0," otherwise, it will be put into the negative accumulator.

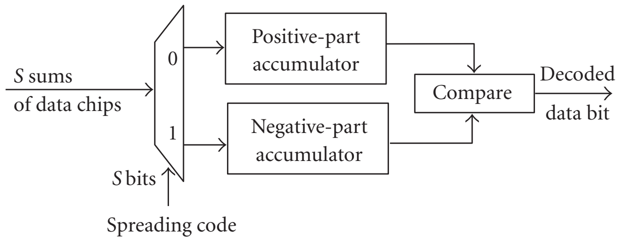


Figure 7: Digital CDMA decoding scheme.

The same selection and accumulation operations are also performed on the other received sum values. The principle of this decoding scheme can be explained as follows. If the original data bit to be transferred is "1," after the XOR logic in the encoding scheme illustrated in Figure 5, it can only contribute a nonzero value to the sums of data chips when a bit of spreading code is "0." Similarly, the 0-value original data bit can only contribute a nonzero value to the sums of data chips when a bit of spreading code is "1." Therefore, after accumulating the sum values according to the bit values of the spreading code, either the positive part or negative part is larger than the other if the spreading codes have orthogonal and balance properties. Hence, the original data bit can be decoded by comparing the values between the two accumulators. Namely, if the positive accumulation value is larger than the negative accumulation value, the original data bit is "1"; otherwise, the original data bit is "0."

3.2.2. *Spreading code selection*

As discussed in Section 3.2.1, the presented encoding/decoding scheme requires the spreading codes used in the CDMA NoC to have both the orthogonal and balance properties. The orthogonal property was explained in the first paragraph of Section 3.2. The balance property means that the number of bit "1" and the number of bit "0" in a spreading code should be equal. Because Walsh code [13] has the required orthogonal and balance properties, it is chosen as the spreading code for the CDMA NoC. In an S-bit ($S = 2^N$, integer $N > 1$) length Walsh code set, there are S-1 sequences which have both the orthogonal and balance properties. Hence, the proposed CDMA NoC can have at most S-1 nodes connecting with one "CDMA transmitter" and one "network arbiter" block as illustrated in Figure 2.

3.2.3. *Spreading code protocol*

In a CDMA network, if multiple users simultaneously use the same spreading code to encode their data packets for transmission, the data to be transferred will interfere with each other because of the loss of orthogonal property among the spreading codes. This situation is called spreading code conflict, which should be avoided. Spreading code protocol is a policy used to decide how to assign and use the spreading codes in a CDMA network in order to eliminate or reduce the possible spreading code conflicts. Several spreading code protocols have been proposed for CDMA packet radio network [14, 15]. Among these proposed spreading code protocols, only transmitter-based protocol (T protocol) and transmitter-receiver-based protocol (T-R protocol) are conflict-free if the users in the network send data to each other randomly. The principles of these two spreading code protocols will be shortly introduced in the following two paragraphs.

(1) *Transmitter-based protocol (T protocol)*: the unique spreading code allocated to each user will be used by the user himself to transfer data to others.

(2) *Transmitter-receiver-based protocol (T-R protocol)*: two unique spreading codes will be assigned to each user in the network, and then a user will generate a new spreading code from the assigned two unique codes for its data encoding.

Because the T-R protocol has the drawback of using a large amount of spreading codes and complicated decoding scheme, T protocol is preferred in the CDMA NoC. However, if T protocol is applied in the network, a receiver cannot choose the proper spreading code for decoding because it cannot know who is sending data to it. In order to solve this problem, an arbiter-based T protocol (A-T protocol) is proposed for the CDMA NoC. In a CDMA network which applies A-T protocol, each user is assigned a unique spreading code for data transfer. When a user wants to send data to another user, he will send the destination information of the data packet to the arbiter before starting data transmission. Then the arbiter will inform the requested receiver to prepare the corresponding spreading code for data decoding according to the sender. After the arbiter has got the acknowledge signal from the receiver, it will send an acknowledge signal back to the sender to grant its data transmission. If there are several users who want to send data to the same receiver, the arbiter will grant only one sender to send data at a time. Therefore, by applying the A-T protocol, spreading code conflicts in the CDMA NoC can be eliminated.

3.2.4. Parallel data transfer principle

The parallel data transfer principle of the CDMA NoC is based on the A-T spreading code protocol described in Section 3.2.3. By applying A-T spreading code protocol, every node in the CDMA NoC needs to send the destination address of the packets to the "network arbiter" as illustrated in Figure 2. After getting the grant signal from "network arbiter," the sender node will send data packets to the "CDMA transmitter" block. The data encoding operations and data transfers are performed in the "CDMA Transmitter" block. Finally, the data decoding operation will be carried out by the data receiving network node. Therefore, the data transfer process in the CDMA NoC can be clarified by describing the functions in the "network arbiter" and the "CDMA transmitter" block, respectively.

(1) *Network arbiter*. After receiving a data transfer request from a network node, "network arbiter" will inform the requested receiver node to prepare the proper spreading code for decoding and send a grant signal back to the sender node. In case that there are more than one sender nodes requesting to send data to the same receiver node simultaneously or at different times, the arbiter will apply "round-robin" arbitration scheme or the "first-come first-served" principle to guarantee that there is only one sender sending data to one specific receiver at a time. The reason for this limitation is that the "packet receiver" block in a network node can receive and decode data from only one sender at a time. However, if different sender nodes request to send data to different receiver nodes, these requests will not block each other and will be handled in parallel in the "network arbiter." The "network arbiter" in the CDMA NoC is different from the arbiter used in a conventional bus. This is because the "network arbiter" here is only used to set up spreading codes for receiving and it handles the requests in parallel in the time domain. In contrary, a conventional bus arbiter is used to allocate the usage of the common communication media among the users in the time-division manner.

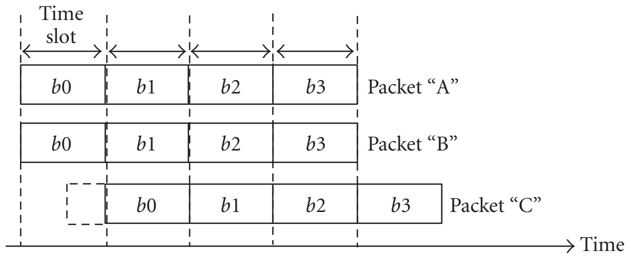


Figure 8: Bit-synchronous transfer scheme.

(2) *CDMA transmitter*. The sender node will start to send data packets to the "CDMA transmitter" after it gets the grant signal from the arbiter. Then the "CDMA transmitter" will encode the data to be transferred with the corresponding unique spreading code of the sender node. Although the "CDMA transmitter" block is implemented by asynchronous circuits, it applies the bit-synchronous transfer scheme. This means that the data from different nodes will be encoded and transmitted synchronously in terms of data bits rather than any clock signals. In Figure 8, the principle of the referred bit-synchronous transfer is illustrated by a situation in which network nodes "A" and "B" send data packets to "CDMA transmitter" simultaneously and node "C" sends a data packet later than "A" and "B." In this situation, the data packet from node "A" will be encoded and transmitted together with the data packet from node "B" synchronously in terms of each data bit. As the data packet from node "C" arrive at a later time point, the transmitter will handle the data bit from "packet C" together with the data bits from packets "A" and "B" at the next start point of the time slot for bit encoding and transmitting processes. The dotted-line frame at the head of the "packet C" in Figure 8 illustrates the waiting duration if the "packet C" arrived in the middle of the time slot for handling the previous data bit. The time slot for handling a data bit is formed by a four-phase

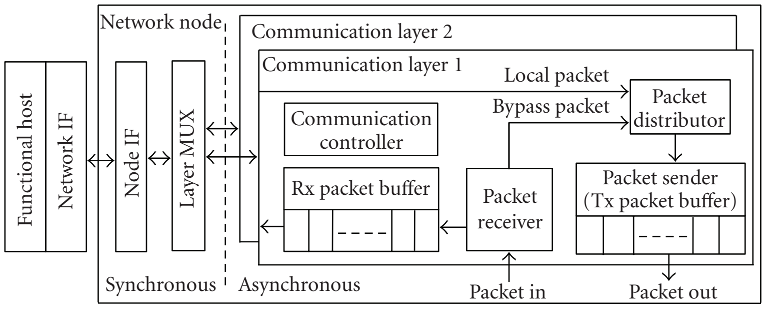


Figure 9: Network node structure of the PTP NoC.

handshake process. The bit-synchronous scheme can avoid the interferences caused by the phase offsets among the orthogonal spreading codes when the data bits from different nodes are encoded and transmitted asynchronously with each other. Because the nodes in the network can request data transfer randomly and independently of each other, "CDMA transmitter" applies the "first-come first-served" mechanism to ensure that the data encoding and transmission are performed as soon as there is a data transfer request.

3.3. Comparison of the data transfer principles

One advantage of the data transfer principle in the CDMA NoC is the feature of parallel data transfer. Although the data transfers in the PTP NoC can also be parallel if they take place in different links among the network nodes, the parallelism in the PTP NoC is largely limited by the possible traffic congestions in a link because the data are transferred through a link between two network nodes in a time-division manner. Another advantage of the CDMA data transfer principle is that no data routing is needed because of the centralized data transfer scheme. This feature can supply stable transfer latency in the CDMA NoC, which in turn facilitates the CDMA NoC to provide a guaranteed service for the on-chip system.

Another advantage of the CDMA NoC is that it can easily support multicast data transfers by requesting multiple receiver nodes to use the same spreading code for receiving. In the PTP NoC, the multicast transfer can be realized only by sending multiple copies of a data packet to its multiple destinations, unless extra logic is added in each network node to copy the multicast packet to both the functional host and the output link to the next node. This increases the traffic load in the network, or complicates the network implementation.

By comparing with the data transfer principle in the PTP NoC, one disadvantage of the CDMA data transfer principle is its complexity caused by the data encoding and decoding operations. Another drawback of the CDMA data transfer principle is that the data transfer efficiency obtained by parallel transfers in the time domain is compromised by the latency introduced by the data spreading scheme. As presented in Section 3.2.1, one data bit will be extended to S bits for the CDMA data transfers. The parameter S is the width of the spreading code applied in the CDMA NoC. As the number of nodes in the NoC increases, the width of the applied spreading code will also be increased. Then the data latency caused by the data spreading will be also increased.

4. THE NETWORK NODE STRUCTURES

"Network node" block is a common type of component needed in both of the PTP NoC and the CDMA NoC. However, different data transfer principles in the two NoC designs imply different structures in the network nodes. This section will present the network node structure in each of the two NoCs.

4.1. Network node structure in the PTP NoC

The network node structure of the PTP NoC is illustrated in Figure 9. The network node consists of "node if," "layer MUX," and "communication layer" blocks. The two blocks outside of the network node illustrate how a "functional host" block as presented in Figure 1 is connected with a network node through a network interface ("network if") block. In the PTP NoC, the applied interface standards include VCI [16] and OCP [17]. As illustrated in Figure 9, GALS scheme in the network is implemented by applying both synchronous and asynchronous designs in each network node. The synchronous blocks, "node if" and "layer MUX," are used to communicate with locally synchronous "functional host" in the system, while the asynchronous blocks are used to perform asynchronous communications among the network nodes. The arrows in Figure 9 demonstrate the data packet flow in a network node. The blocks in the network node illustrated in Figure 9 will be introduced in the following three paragraphs.

(1) *Node if.* This block is responsible for assembling the data from "functional host" into data packets or delivering the data packets from network node to "functional host" according to the interface standard applied in "network if" block.

(2) *Layer MUX.* As the name of this block indicates, this block behaves as a multiplexer used to connect "node if" block with a certain "communication layer" block during the data transfers between network node and "functional host" block.

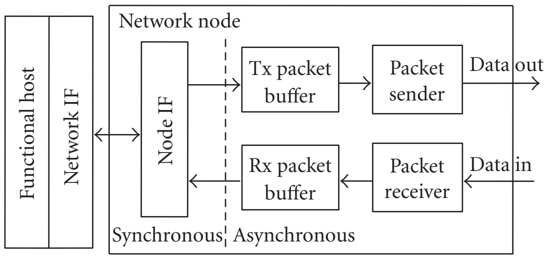


Figure 10: Network node structure of the CDMA NoC.

(3) *Communication layer*. The function of this block is to perform the globally asynchronous communication with other network nodes through a handshake protocol. As illustrated in Figure 9, the two "communication layer" blocks in a network node are used to connect with two other network nodes in the bidirectional ring NoC. More "communication layer" blocks can be used in a network node to implement other types of topology. There are five subblocks in a "communication layer" block. The "packet receiver" subblock is used to receive data packets from another network node. If the destination of the received packet is the current network node, the packet is called "incoming packet," and it will be stored in "Rx packet buffer." Otherwise, the received packet is called "bypass packet," and it will be dispatched into "packet sender" block via "packet distributor" for further transferring. The "communication controller" subblock in Figure 9 represents the controller which takes care of the necessary arbitrations and communication control.

4.2. Network node structure in the CDMA NoC

The block diagram of the network node structure of the CDMA NoC is shown in Figure 10 where the arrows represent the flows of data packets. In Figure 10, the "network if" block which belongs to the functional host is an interface block for connecting a functional host with a "network node." The GALS scheme is applied in "network node" block of the CDMA NoC by using synchronous design in the "node if" subblock and using asynchronous design in the other subblocks. The network interface standards supported in the CDMA NoC also include the VCI and OCP standards. The subblocks in the network node will be addressed in the following four paragraphs.

(1) *Node if*. This block is used to assemble the data from "functional host" into packets and send the packets to "Tx packet buffer" or disassemble the received packet from "Rx packet buffer" and send the extracted data to "functional host."

(2) *Tx/Rx packet buffer*. "Tx packet buffer" is used to store the data packets from "node if," and then deliver the packets to "packet sender." The "Rx packet buffer" stores and delivers the received packets from "packet receiver" to "node if."

(3) *Packet sender*. If "Tx packet buffer" is not empty, "packet sender" will fetch a data packet from the buffer by an asynchronous handshake protocol. Then it will extract the destination information from the fetched packet and send the destination address to "network arbiter." After "packet sender" gets the grant signal from the arbiter, it will start to send data packets to "CDMA transmitter."

(4) *Packet receiver*. After system reset, this subblock will wait for the sender information from "network arbiter" to select the proper spreading code for decoding. After the spreading code for decoding is ready, the receiver will send an acknowledge signal back to "network arbiter" and start to receive data from "CDMA transmitter," and then send the decoded data to "Rx packet buffer" in the packet format.

4.3. Comparison of the network node structures

By comparing with the presented network node in the PTP NoC, the network node in the CDMA NoC has less complexity. The main reason is that the network node of the CDMA NoC does not need to handle any bypass packets or the packet routing issues because of the centralized traffic scheme. Therefore, the "communication controller" block and the "packet distributor" block in the network node for the PTP NoC are not needed in the node for the CDMA NoC. When the data transfer parallelism needs to be increased in the PTP NoC, more "communication layer" blocks in a network node are needed in order to set up more links with other nodes, whereas the network node in the CDMA NoC does not need to change in this situation. One advantage of both of the network nodes is that they are both replicable because each network node structure in the network is the same. This advantage makes both the PTP NoC and the CDMA NoC designs modular.

5. ASYNCHRONOUS DESIGNS

As presented in Section 4, the GALS scheme is applied in the PTP NoC and in the CDMA NoC by implementing the global interconnect fabric with asynchronous designs. However, this is not the only way to implement GALS scheme in an on-chip network. For example, in "islands of synchronicity" (IoS) methodology presented in [18], the GALS scheme is implemented in SoC designs by localizing the clock in each of the functional IP blocks and connecting the isolated clock "islands," the functional IPs, with asynchronous communication links. If the IoS methodology is applied in the presented PTP NoC and the CDMA NoC, it means that all the blocks, including network nodes, "CDMA transmitter," and "network arbiter," in the designs need to be synchronous designs which work with different local clock frequencies. Then, the communications among the blocks in the NoC designs use asynchronous protocols. The advantage of applying the IoS methodology is that all the blocks in the design can be implemented by using standard synchronous design tools and flow. However, two disadvantages addressed in the following two paragraphs need to be noticed.

(1) *Synchronization cost*. The signals need to be synchronized with the local clocks when they cross different clock domains. If the IoS methodology is applied, two synchronization operations are needed when data enter into and leave from the global interconnect fabric during a data transfer process because the interconnect fabric works with its own clock rate. If the interconnect fabric is implemented by asynchronous designs, the synchronization step is not needed when data enter into the global interconnect fabric because a signal from a synchronous domain can enter into an asynchronous domain directly. Therefore, synchronization-related latency and area cost can be reduced 50% if the global interconnect fabric is implemented by asynchronous designs directly.

(2) *Area and power costs*. As presented in Section 4, "Tx/Rx packet buffer" composed by the asynchronous FIFO presented in [19] takes a large portion of the network node structure in both of the NoCs. As addressed in [19], the area and power costs of the asynchronous FIFO are 51.5% and 54.2% less than a synchronous implementation. Hence, if all the blocks related with global interconnection are implemented by synchronous designs, the area and power costs of the "Tx/Rx packet buffer" will be nearly doubled by comparing with the asynchronous implementation.

Therefore, in order to reduce the cost of synchronization, area, and power, asynchronous designs are applied to implement the blocks for global interconnections in the PTP NoC and the CDMA NoC. The asynchronous designs applied in the two NoC designs will be addressed in this section.

5.1. Asynchronous design in the PTP NoC

The basic component of the PTP NoC is the network node presented in Section 4.1. As illustrated in Figure 9, the blocks which apply asynchronous designs in the network node are the "communication controller," "packet receiver," "packet distributor/sender," and "packet Rx/Tx buffer" blocks. The four-phase dual-rail protocol is applied in the asynchronous designs in order to make the data transfers delay-insensitive. The control logic used in the asynchronous blocks of the PTP NoC will be presented in this subsection.

5.1.1. Control logic in the "communication controller"

The "communication controller" block is the main control block which takes care of data packet receiving, sending, and storing processes in the network node. Each of the mentioned packet handling processes is controlled by a control pipeline which can be seen as a finite state machine (FSM) in the "communication controller" block. The control processes of the FSMs are illustrated in Figure 11 and are explained in the following three paragraphs.

(1) *Packet receiving FSM*. As illustrated in Figure 11(a), there are six states in this FSM. The machine will move from its initial "rx_idle" state to "rx_pkt" state when "packet receiver" starts to receive a packet. After the packet receiving is completed, the FSM will move to "chk_addr" state to check the destination of the received packet. If the received packet is "incoming packet," the FSM will move to "chk_buf" state

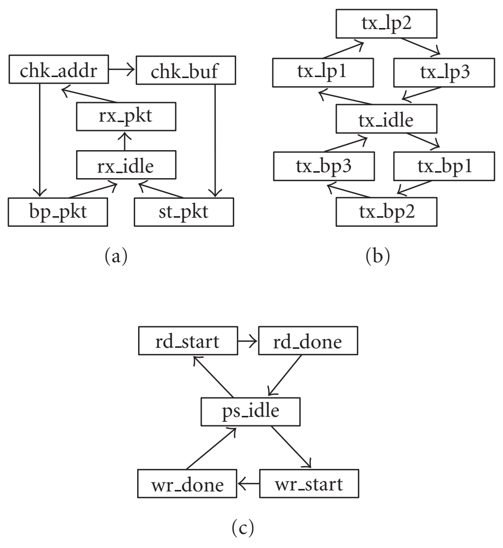


Figure 11: State transfer graphs of the FSMs.

to check the status of "Rx packet buffer." If the buffer is not full, the "incoming packet" will be stored in the buffer during "st_pkt" state, otherwise, it will be held by "packet receiver" until there is room available in the buffer. If the received packet is "bypass packet," it will be sent to the network node connected to the output port of current node in "bp_pkt" state.

(2) *Packet sending FSM*. This FSM illustrated in Figure 11(b) is responsible for sending two types of data packets into the "Tx packet buffer" in "packet sender" via "packet distributor" block. One type of packets is "local packet" which refers to the packet which comes from the "functional host" connected with the network node. Another type of packets is the "bypass packet" explained in Section 4.1. For sending "local packet," the FSM will be triggered by the signal from the "node if" block after a "local packet" is ready to be transferred. Then the FSM will go through "tx_lp1," "tx_lp2," and "tx_lp3" states for checking the status of the "Tx packet buffer," sending the packet into the buffer, and going back to "tx_idle" state, respectively. The process of sending a "bypass packet" into the transfer buffer is similar to the process of sending "local packet" except that the FSM will go through "tx_bp1," "tx_bp2," and "tx_bp3" states.

(3) *Packet storing FSM*. This FSM presented in Figure 11(c) is used to store or fetch an "incoming packet" to or from "Rx packet buffer" block. The process of storing an "incoming packet" will be triggered by packet receiving FSM during the "st_pkt" state as illustrated in Figure 11(a). The packet storing FSM will go through the "wr_start" and "wr_done" states to complete the storing task. The "rd_start" and "rd_done" states are for the process of fetching a stored "incoming packet" from "Rx packet buffer." The fetching process will be triggered by the "node if" block after getting flag signal from"communication controller" block.

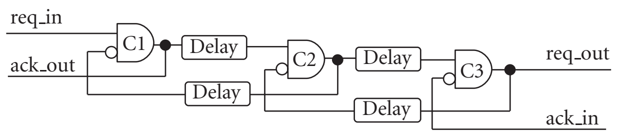


Figure 12: Micropipeline control logic.

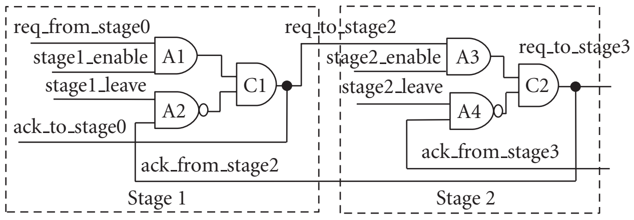


Figure 13: Control pipeline structure in the FSMs.

The presented control FSMs in the "communication controller" block are realized by applying the delay-insensitive micropipeline control logic presented in [20]. The structure of the micropipeline control logic is illustrated in Figure 12. The principle of micropipeline control logic is to use the output from the current stage to enable or disable the input of previous stage. Two stages of the control pipeline used in "communication controller" block for building the FSMs are illustrated in Figure 13. Each stage of the pipeline represents a state element of an FSM. In Figure 13, we can see that the FSM uses micropipeline control logic as the backbone and few AND gates as the delay components illustrated in Figure 12, hence it is also delay-insensitive. The state information of the FSM is passed through each stage in the pipeline by a four-phase handshake protocol. If we take the "stage 1" illustrated in Figure 13 as an example, when both the "req_from_stage0" and "stag1_enable" signals are "1," the output of "C1" will be set to logic "1" which indicates that the current state of the FSM is in "stage 1." Then the output of "C1" can be used as a request signal to trigger the control logic in the corresponding function blocks for a certain communication process.

5.1.2. *Control logic in other blocks*

Besides the FSMs in the "communication controller" block, control pipelines exist also in other asynchronous blocks used to perform the concrete task of moving the data packets in or out of the individual blocks through a four-phase dual-rail protocol. The FSMs in the "communication controller" block control the processes of receiving, sending, or storing data packets by triggering the control pipeline in the corresponding function blocks. The control pipeline structure used in the "packet receiver," "packet distributor," and "packet sender" blocks is illustrated in Figure 14. This structure is derived from the micropipeline control logic and is used to perform data transfers by interacting with the control signals coming from the "communication controller" block. For example, when the pipeline structure is used in "packet

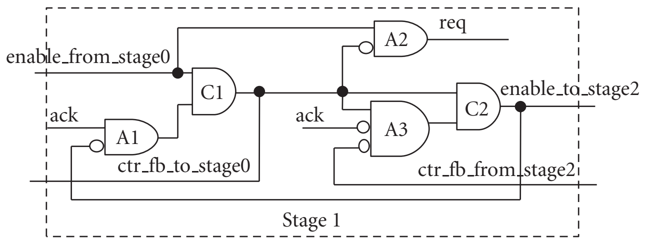


Figure 14: Block control pipeline structure.

receiver," it will receive the packet receiving enable signal from the "communication controller" block as the enable signal for the first stage of the pipeline structure. Then, a request signal will be generated by gate "A2" in the pipeline stage as illustrated in Figure 14. The generated request signal will start a handshake process to receive and store a packet. When the acknowledge signal appears at the input of "A1," it will turn the output of "C1" to "1," which will clear the request signal via "A2" and enable the "C2" to capture the falling edge of the acknowledge signal through "A3." The falling edge of the "ack" signal means that the required tasks of the current step have been done and the current handshake process is completed. Hence, when it appears at the input of "A3," the output of "C2" will be triggered to "1" to enable the next stage to take over the control process of the next operation, for example, receiving next data packet cell in the "packet receive" block.

The presented block control pipeline structure can only meet quasidelay-independent (QDI) model because the input "ack" signal is branched to "A1" and "A3," however, the timing requirement for distributing the "ack" input signal along the isochronic wire forks is quite loose since the logic delays in "A1" and "C1" are usually much larger than the logic delay of the inverter at the input of "A3."

The control pipeline structure illustrated in Figure 14 is also used in the "Tx/Rx packet buffer" blocks to control the process of accessing the asynchronous FIFOs presented in [19].

5.2. ***Asynchronous design in the CDMA NoC***

As illustrated in Figure 2 and addressed in Section 4.2, the asynchronous blocks in the CDMA NoC include the "CDMA transmitter," "network arbiter," "Tx/Rx packet buffer," and "packet receiver/sender" blocks. Since the "CDMA transmitter" and "network arbiter" blocks are data-path centric blocks, the control logic used in these blocks can be composed by a straightforward C-element pipeline as illustrated in Figure 15. Each stage in the C-element pipeline is enabled by the enable signals generated by the data completion detection circuits. The control token will be passed from one stage to the next one through each C-element in the pipeline.

The control logic used in the "Tx/Rx packet buffer" and "packet receiver/sender" blocks of the network node for the CDMA NoC is similar to the control logic illustrated in Figure 14 except that the enable conditions of the C-element are different.

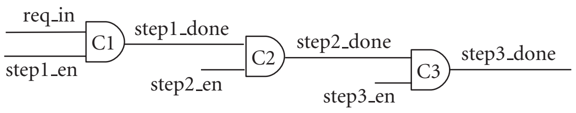


Figure 15: C-element control pipeline.

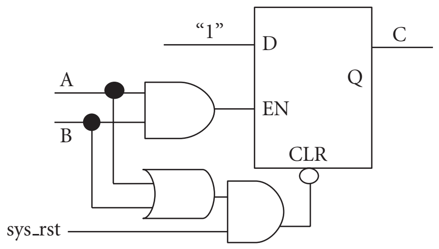


Figure 16: C-element structure.

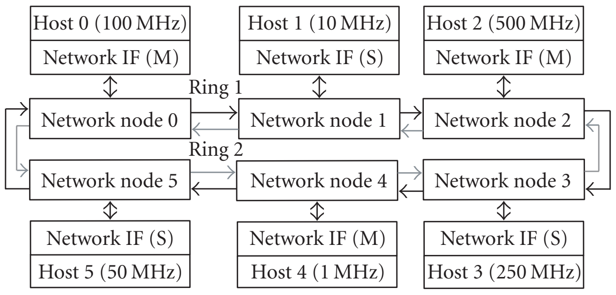


Figure 17: Six-node PTP NoC simulation network.

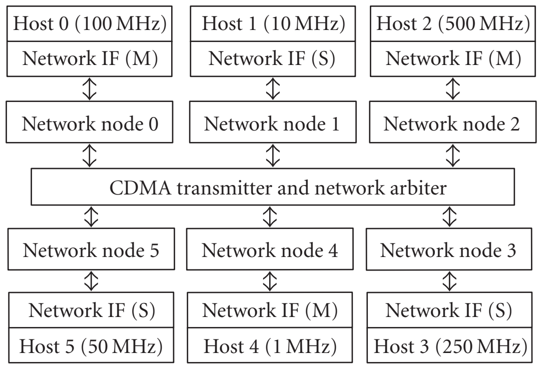


Figure 18: Six-node CDMA NoC simulation network.

5.3. *Asynchronous design implementation*

Since the synchronous designs in the presented PTP NoC and the CDMA NoC are done in register-transfer level (RTL) by using VHDL, it would be convenient for the implementation if the asynchronous designs apply the same design format. From the control logic structures described, we can see that the C-element is a basic component widely used in the asynchronous designs and a C-element is normally implemented in transistor level. Therefore, modeling the C-element in RTL is an important task for modeling the asynchronous designs using VHDL. Hence, an RTL two-input C-element structure is proposed in Figure 16. The proposed C-element structure is based on a D-flipflop (D-FF) which uses "A AND B" as the enable ("EN") signal and "A OR B" as the reset signal ("CLR"). The data input port ("D") of the D-FF is attached to logic "1" constantly. The idea of using a flipflop to build a C-element has been presented in [21] where an RS-FF is suggested to be used; whereas, the D-FF C-element structure in Figure 16 is more stable because it avoids data switching at the data input port.

The C-element structure illustrated in Figure 16 is hazard-free under one-input change assumption by applying the "AND" gate and "OR" gate at the "EN" port and "CLR" port, respectively. For certain two-input switch patterns, 00→11 and 11→00, the structure in Figure 16 is also hazard-free; whereas, the input switch patterns, 01→10 and 10→01, are not allowed because they may produce a logic error which depends on the wire delay. Because all the C-elements in the PTP and the CDMA NoC designs are used to follow a four-phase handshake protocol, there are no 01→10 or 10→01 input switch patterns for the C-elements in the designs. Thus, the proposed C-element structure can be safely used in the NoC designs.

By using the proposed RTL C-element structure, the asynchronous designs of the two NoCs are modeled in RTL using VHDL. Since the entire designs are in a uniform VHDL format, the commonly used synchronous design tools and flow can be used for implementing the NoC designs.

6. PERFORMANCE ESTIMATION

After the structures and designs of the PTP NoC and the CDMA NoC have been discussed, the performance of the two NoC designs will be addressed in this section. The two six-node simulation networks used for performance estimations and the estimation results will be presented and discussed in the following subsections.

6.1. *The simulation network setup*

In order to estimate the performance, two six-node networks have been set up for simulation purpose. The simulation network which applies the PTP NoC structure is illustrated in Figure 17, while the network which applies the CDMA NoC structure is illustrated in Figure 18. In each of the two simulation networks, six "functional host" blocks are connected into the network through six network nodes. The network nodes are connected to each other through the two different NoC structures, respectively, for the purpose of comparison. The interface standard applied in the simulation networks is BVCI standard [16]. Three hosts act as masters and the other three act as slaves, as denoted by the labels "M" and "S,"

Table 1: Area cost of the PTP NoC components.

Blocks of network node	Area (μm^2)
Node IF (BVCI slave type)	13430.8
Layer MUX	18346.0
Communication controller	7823.4
Packet distributor	6783.0
Packet sender (include Tx packet buffer)	44740.6
Packet receiver	6955.0
Rx packet buffer	40255.5
Total area of a network node (includes 2 "communication layer blocks")	**244891.8**

Table 2: Area cost of the CDMA NoC components.

Block name		Area (μm^2)
Network node	Node IF	18825.2
	Tx/Rx packet buffer	71778.3
	Packet sender	17707.0
	Packet receiver	23253.0
	Total area of a network node	**131563.5**
	CDMA transmitter	10338.3
	Network arbiter	17686.5

respectively, in the "network if" blocks. The master hosts can generate requests to any slave hosts, while the slave hosts can generate responses only for the received requests passively. The functional hosts in the two simulation networks are not implemented as any concrete designs. They are simulated by the stimulus which comes from the "network if" block to the network nodes. Hence, the configurations of the two simulation networks are the same except for the connection structures.

6.2. The area costs

By using the scheme described in Section 5.3, both the synchronous and asynchronous designs in the simulation networks are realized in RTL using VHDL. A 0.18 μm standard cell library is used for synthesizing the two simulation networks. The area costs of the two simulation networks are listed in Tables 1 and 2, respectively. As listed in Table 2, the area cost of the "network node" in the CDMA NoC is 53% smaller than the area of the "network node" block in the PTP NoC. The reason of this big difference of the area costs is that there are two "communication layer" blocks contained in each network node of the PTP NoC in order to set up bidirectional ring links. After including the area costs of "CDMA transmitter" and "network arbiter" blocks, the total area cost of the six-node CDMA NoC is 55% smaller than the area cost of the six-node PTP NoC.

6.3. The data transfer latencies

After the networks were synthesized, gate-level simulations of the two networks were performed using an event-driven

Table 3: Synchronous transfer latency.

Interface type	Latency of sending data to "network node"	Latency of receiving data from "network node"
BVCI master	8 local clock cycles + 2.5 ns	8 local clock cycles + 3.2 ns
BVCI slave	4 local clock cycles + 2.5 ns	4 local clock cycles + 3.1 ns

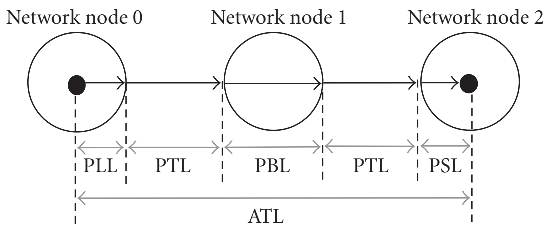


Figure 19: ATL parameters of the PTP NoC.

simulator. Because the GALS scheme is applied in the two simulation networks, the data transfer latency of the networks can be separated into two parts which include synchronous transfer latency (STL) and asynchronous transfer latency (ATL). The STL refers to the data transfer latency between a functional host and the network node attached to it. STL depends on the local clock and the type of interface. Since the same "node if" block is applied in both networks, the STL values for the two simulation networks are the same. The measured STL values are listed in Table 3. The constant values in Table 3 are caused by the handshakes in the asynchronous domain. They are independent of the local clock rate but belong to the synchronous transfer processes. Therefore they are counted as a part of STL.

The ATL refers to the data transfer latency of transferring data packets from one network node to the other node through an NoC structure using asynchronous handshake protocols. The ATL values in the PTP and CDMA simulation networks consist of different parameters which will be discussed in the following subsections.

6.3.1. ATL in the PTP NoC

The ATL in the PTP NoC consists of four parameters: packet loading latency (PLL), packet transfer latency (PTL), packet bypass latency (PBL), and packet storing latency (PSL). These latency parameters are measured in a noncongested situation which means that no conflicts between "bypass packet" transfer and the "local packet" transfer are included in the simulation. The concept of the four latency parameters is illustrated in Figure 19 with an example that "network node 0" sends one packet to "network node 2" via "network node 1." The black arrows in Figure 19 represent the packet transfer direction. The portions of the transfer used to measure the different parameters of latency are marked by grey arrows in Figure 19 and are explained in the next four

Table 4: Measured ATL values of the PTP NoC.

Packet length	PLL (ns)	PTL (ns)	PBL (ns)	PSL (ns)
2 data cells	11.7	9.7	10.7	3.3
3 data cells	15.2	13.1	14.2	3.3
4 data cells	18.6	16.5	17.6	3.3

Table 5: Measured ATL values of the CDMA NoC.

Packet length	PLL (ns)	PTL (ns)	PSL (ns)
2 data cells	5.7	384.6	5.5
3 data cells	5.7	768.9	5.5
4 data cells	5.7	1153.7	5.5

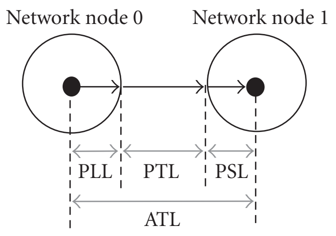


Figure 20: ATL parameters of the CDMA NoC.

paragraphs,

$$\text{ATL} = \text{PLL} + \text{PTL} \times (N+1) + \text{PBL} \times N + \text{PSL}. \quad (1)$$

(1) *Packet load latency (PLL)*. It is the time used to load one "local packet" into "Tx packet buffer."

(2) *Packet transfer latency (PTL)*. This latency refers to the time used to transfer one data packet from the "packet sender" of a network node to the "packet receiver" of an adjacent node using a handshake protocol.

(3) *Packet bypass latency (PBL)*. After a network node receives a packet from another node, it will check its destination address. If it is a "bypass packet," it will be delivered into "Tx packet buffer." The time spent on this process is called PBL.

(4) *Packet storing latency (PSL)*. It is the time spent on storing one "incoming packet" into "Rx packet buffer."

The formula of calculating the ATL of transferring one packet in the PTP NoC is given in (1). It represents the situation in which the packet traverses several network nodes before reaching its destination. N refers to the number of intermediate nodes between the source node and destination node of a packet. If a packet is transferred between two adjacent network nodes, then N is 0. The values of ATL parameters measured in the simulation are listed in Table 4. The listed latency values only include the logic gate delay of the circuits, no wire delay is considered. More accurate latency values could be obtained by including the wire delay after layout.

6.3.2. ATL in the CDMA NoC

The ATL in the CDMA NoC consists of three parameters: packet loading latency (PLL), packet transfer latency (PTL), and packet storing latency (PSL). The concept of those ATL parameters is illustrated in Figure 20 with an example where "network node 0" sends one data packet to "network node 1." The black arrows in Figure 20 represent the packet transfer direction. The portions of the transfer used to measure the different parameters of ATL are marked by grey arrows in Figure 20 and are explained in the following three paragraphs.

(1) *Packet load latency (PLL)*. This is the time used by the "packet sender" block in a "network node" to fetch one data packet from "Tx packet buffer" and prepare to send the data packet to "CDMA transmitter."

(2) *Packet transfer latency (PTL)*. This latency refers to the time used to transfer one data packet from the "packet sender" of the sender node to the "packet receiver" of the receiver node through the CDMA channel using a handshake protocol.

(3) *Packet storing latency (PSL)*. After the receiver node receives a data packet, it will spend a certain amount of time to store the received data packet into "Rx packet buffer." This time duration is measured as PSL.

The measured values of ATL parameters of the CDMA NoC are listed in Table 5. The listed latency values only include the logic gate delay of the circuits, no wire delay is considered. In Table 5, we can see that PTL increases as the packet length increases. This is because the data cells in a packet are sent in a serial manner in the CDMA NoC. Thus, more data cells need more transmission time. The PLL and PSL values are not affected by the packet length. The reason is that the data cells in a packet are loaded or stored in a parallel manner.

6.3.3. Comparing the ATL values

From the simulation results presented in Sections 6.3.1 and 6.3.2, we can see that the ATL value in the six-node CDMA NoC is a stable value for a certain data packet length, whereas the ATL value in the PTP NoC is a variable depending on the packet traffic route. The ATL parameter "PBL" of the PTP NoC does not exist in the ATL of the CDMA NoC because the data packets in the CDMA NoC are transferred directly from their source nodes to their destination nodes. The stable ATL value is an advantage of the CDMA NoC since it is very helpful for supplying guaranteed transfer latency service in the network. However, by comparing with the ATL value of the PTP NoC, the ATL value of the CDMA NoC is much larger. For example, according to values listed in Table 5, ATL of transferring a two-cell packet in the CDMA NoC is 395.8 nanoseconds. This value equals the ATL value of transferring the same size packet through 17 intermediate nodes in the PTP NoC according to (1) and Table 4. The reason for the large ATL value in the CDMA NoC is that each original data bit is extended into S bits by an S-bit spreading code during the obligatory data spreading process for CDMA transmission, and the encoded data are transferred bit by bit in the current realization of the CDMA NoC; whereas, in the

PTP NoC, the data bits are transferred cell by cell without any encoding, namely 32 original data bits are transferred at one time. Therefore, the ATL value of CDMA NoC can be reduced by transferring the encoded bits in parallel.

6.4. *SystemC modeling for further estimation*

The data transfer latency estimations made in Section 6.3 are based on two six-node simulation networks. However, in different applications, the number of nodes in an NoC can be different. Therefore, data transfer latency estimations under different numbers of network nodes of the two NoC designs would be helpful for further evaluation.

As discussed in Section 6.3.1, the data transfer latency of the PTP NoC is mainly affected by the number of intermediate network nodes which a packet passes through during the transfer. Therefore, by using the transfer latency values extracted from the six-node RTL simulation network, the ATL values of the PTP NoC with different numbers of network nodes can be estimated by using (1). For the CDMA NoC, the ATL values with different network node numbers are difficult to get from the six-node simulation network presented in Section 6.3 due to the lack of scalability in the CDMA NoC. Since the data transfer latency estimation presented in Section 6.3 is based on the RTL simulation network realized by using VHDL, any changes in the simulation network will incur a time-consuming synthesis and simulation design cycle. Therefore, a flexible and fast simulation model of the CDMA NoC is preferred for further ATL performance estimations of the CDMA NoC.

SystemC [22] is a C++ class library which can be used to model system-level designs. Since a SystemC model is totally described by a software programming language, the abstraction level of the system model can be very flexible and the simulation can run at a faster speed than an RTL model. Thus, a SystemC model of the CDMA NoC is built for the flexible and fast simulation purpose.

The SystemC model of the CDMA NoC is built in transaction level by modelling each block of the CDMA NoC as a channel [22]. The asynchronous communications among the blocks are modelled by calling each others' channel interface functions in the SystemC model. In order to estimate the ATL values of the CDMA NoC via the transaction-level SystemC model, a set of latency values listed in Table 6 is extracted from the gate-level simulation of the RTL six-node CDMA network presented in Section 6.3. By back-annotating the transfer latency values to the corresponding channels in the SystemC model, the ATL estimations of the CDMA NoC with different numbers of network nodes can be obtained through simulating the SystemC model in transaction level. The obtained ATL estimation values from the SystemC model simulations are listed in Table 7. From Table 7, we can see that the transfer latency of the CDMA NoC increases as the number of network nodes increases. The main reason of the latency increasing is that the data encoding latency in "CDMA transmitter" block is getting larger when the number of network nodes increased. Another reason is that the width of the orthogonal codes used for encoding increases as the number of nodes increased in the network. Thus, the spreading code loading latency in "packet receiver" block would be increased as a consequence. Because the back-annotated SystemC model of the CDMA NoC only uses a limited number of extracted latency values presented in Table 6, the ATL values listed in Table 7 only can give a quick glimpse on the latency situations when the number of network nodes in the CDMA NoC is changed. The accurate latency information needs to be obtained through real circuit implementations.

Table 6: Extracted transfer latency values.

Blocks	Processes	Latency
Tx/Rx packet buffer	Read	10.9 ns
	Write	11.5 ns
Packet sender	Send a 2-cell packet to "CDMA transmitter"	99.2 ns
Packet receiver	Load decoding PN code	1.2 ns
	Receive a 2-cell packet from "CDMA transmitter"	192.0 ns
Network arbiter	Arbitration	4.3 ns
CDMA transmitter	Data encoding	2.9 ns

Table 7: ATL estimation values of the CDMA NoC with different numbers of nodes.

Number of nodes	Spreading code length (bits)	ATL of sending a 2-cell packet
3	4	362.7 ns
6	8	411.8 ns
12	16	510.0 ns
24	32	706.4 ns

7. CONCLUSION

A PTP connection NoC and a CDMA connection NoC were examined and compared in this paper. Both of the presented NoC designs are packet-switched networks and support the GALS communication scheme. The two NoC designs are compared in terms of NoC structures, data transfer principles, network node designs, asynchronous designs, and the performances. The features of the two NoC structures are summarized in the following five paragraphs.

(1) *NoC structures.* The PTP NoC applies direct links among the network nodes and a distributed traffic scheme for the data communication. The CDMA NoC applies a centralized connection scheme and provides parallel data transfers in time domain.

(2) *Data transfer principles.* The PTP NoC transfers data packets in their original form among the links in the network. The CDMA NoC applies CDMA technique to share the centralized communication channel among all the network nodes both in time and space domains. The data streams from different network nodes in the CDMA NoC are separated from each other by encoding them with a set of orthogonal codes.

(3) *Network node designs.* The network node structure in the PTP NoC is more complex than the structure of the network node in the CDMA NoC. The communication control tasks in the CDMA NoC network node are less than the tasks in the PTP NoC network node. The complexity caused by packet routing processes in the network node of the PTP NoC is avoided in the network node of the CDMA NoC.

(4) *Asynchronous designs.* The asynchronous designs applied in the PTP and CDMA NoCs are similar to each other. The four-phase dual-rail protocol is applied in both NoC designs. The control logic used in the asynchronous designs of the two NoC designs is based on the micropipeline control logic. Both the synchronous and asynchronous designs in the two NoC designs are realized in RTL using VHDL.

(5) *Performance estimations.* Two simulation networks which apply the PTP NoC structure and the CDMA NoC structure, respectively, have been synthesized using a 0.18 μm standard cell library. The area cost of the CDMA simulation network is 55% smaller than the PTP simulation network. When the number of network nodes is certain, the ATL value of the CDMA NoC is a stable value for the same-size packets. However, the ATL value of the PTP NoC is smaller than the value of the CDMA NoC when no data transfer congestions are considered in the PTP NoC. One reason of the large ATL in the CDMA NoC is that the applied data spreading technique produces a large amount of encoded data bits for transmission. Another reason is that the encoded data are delivered bit by bit in the CDMA NoC, whereas the PTP NoC transfers 32 data bits at one time. Therefore, the ATL value of the CDMA NoC can be improved largely by increasing the number of data bits delivered at one time.

REFERENCES

[1] J. A. J. Leijten, J. L. van Meerbergen, A. H. Timmer, and J. A. G. Jess, "PROPHID: a data-driven multi-processor architecture for high performance DSP," in *Proceedings of European Design and Test Conference*, p. 611, Paris, France, March 1997.

[2] P. Guerrier and A. Greiner, "A generic architecture for on-chip packet-switched interconnections," in *Proceedings of the Design, Automation and Test in Europe Conference (DATE '00)*, pp. 250–256, Paris, France, March 2000.

[3] D. Sigüenza-Tortosa, T. Ahonen, and J. Nurmi, "Issues in the development of a practical NoC: the Proteo concept," *Integration, the VLSI Journal*, vol. 38, no. 1, pp. 95–105, 2004.

[4] J. Muttersbach, T. Villiger, H. Kaeslin, N. Felber, and W. Fichtner, "Globally-asynchronous locally-synchronous architectures to simplify the design of on-chip systems," in *Proceedings of the 12th Annual IEEE International ASIC/SOC Conference*, pp. 317–321, Washington, DC, USA, September 1999.

[5] C. A. Zeferino, F. G. M. E. Santo, and A. A. Susin, "ParIS: a parameterizable interconnect switch for networks-on-chip," in *Proceedings of the 17th Symposium on Integrated Cicuits and Systems Design (SBCCI '04)*, pp. 204–209, Pernambuco, Brazil, September 2004.

[6] K. Goossens, J. Dielissen, and A. Rădulescu, "Æthereal network on chip: concepts, architectures, and implementations," *IEEE Design and Test of Computers*, vol. 22, no. 5, pp. 414–421, 2005.

[7] X. Wang and J. Nurmi, "An on-chip CDMA communication network," in *Proceedings of International Symposium on System-on-Chip*, pp. 155–160, Tampere, Finland, November 2005.

[8] X. Wang, D. Sigüenza-Tortosa, T. Ahonen, and J. Nurmi, "Asynchronous network node design for Network-on-Chip," in *Proceedings of International Symposium on Signals, Circuits and Systems (ISSCS '05)*, vol. 1, pp. 55–58, Iasi, Romania, July 2005.

[9] B.-K. Tan, R. Yoshimura, T. Matsuoka, and K. Taniguchi, "A novel dynamically programmable arithmetic array using code division multiple access bus," in *Proceedings of the 8th IEEE International Conference on Electronics, Circuits and Systems (ICECS '01)*, vol. 2, pp. 913–916, Malta, September 2001.

[10] S. Shimizu, T. Matsuoka, and K. Taniguchi, "Parallel bus systems using code-division multiple access technique," in *Proceedings of IEEE International Symposium on Circuits and Systems*, vol. 2, pp. 240–243, Bangkok, Thailand, May 2003.

[11] M. Takahashi, B.-K. Tan, H. Iwamura, T. Matsuoka, and K. Taniguchi, "A study of robustness and coupling-noise immunity on simultaneous data transfer CDMA bus interface," in *Proceedings of IEEE International Symposium on Circuits and Systems*, vol. 4, pp. 611–614, Phoenix, Ariz, USA, May 2002.

[12] R. H. Bell Jr., C. Y. Kang, L. John, and E. E. Swartzlander Jr., "CDMA as a multiprocessor interconnect strategy," in *Proceedings of the 35th Asilomar Conference on Signals, Systems and Computers*, vol. 2, pp. 1246–1250, Pacific Grove, Calif, USA, November 2001.

[13] E. H. Dinan and B. Jabbari, "Spreading codes for direct sequence CDMA and wideband CDMA cellular networks," *IEEE Communications Magazine*, vol. 36, no. 9, pp. 48–54, 1998.

[14] E. S. Sousa and J. A. Silvester, "Spreading code protocols for distributed spread-spectrum packet radio networks," *IEEE Transactions on Communications*, vol. 36, no. 3, pp. 272–281, 1988.

[15] D. D. Lin and T. J. Lim, "Subspace-based active user identification for a collision-free slotted ad hoc network," *IEEE Transactions on Communications*, vol. 52, no. 4, pp. 612–621, 2004.

[16] VSI Alliance, "Virtual Component Interface Standard version 2," April 2001, http://www.vsi.org/.

[17] OCP-IP Association, "Open Core Protocol Specification," 2001, http://www.ocpip.org/.

[18] A. P. Niranjan and P. Wiscombe, "Islands of synchronicity, a design methodology for SoC design," in *Proceedings of Design, Automation and Test in Europe Conference and Exhibition (DATE '04)*, vol. 3, pp. 64–69, Paris, France, February 2004.

[19] X. Wang and J. Nurmi, "A RTL asynchronous FIFO design using modified micropipeline," in *Proceedings of the 10th Biennial Baltic Electronics Conference (BEC '06)*, Tallinn, Estonia, October 2006.

[20] I. E. Sutherland, "Micropipelines," *Communications of the ACM*, vol. 32, no. 6, pp. 720–738, 1989.

[21] Q. T. Ho, J. B. Rigaud, L. Fesquet, M. Renaudin, and R. Rolland, "Implementing asynchronous circuits on LUT based FPGAs," in *Proceedings of the 12th International Conference on Field-Programmable Logic and Applications (FPL '02)*, vol. 2438 of *Lecture Notes in Computer Science*, pp. 36–46, Montpellier, France, September 2002.

[22] *IEEE Standard SystemC Language Reference Manual*, http://www.systemc.org/.

PUBLICATION 7

X. Wang, and J. Nurmi, “Comparing Two Non-Blocking Concurrent Data Switching Schemes for Network-on-Chip”, in Proceedings of the 2007 International Conference on Computer as a tool, (EUROCON 2007), pages 2587-2592, Warsaw, Poland, September 2007.

Comparing Two Non-Blocking Concurrent Data Switching Schemes for Network-on-Chip

Xin Wang*, and Jari Nurmi*
* Tampere University of Technology, 33101 Tampere, Finland
E-mail: *{xin.wang, jari.nurmi}@tut.fi*

***Abstract*— Based on a previously developed Code-Division Multiple-Access (CDMA) Network-on-Chip (NoC) structure, this paper examines the overhead cost of data encoding and decoding operations in the CDMA data switching scheme by comparing it with another non-blocking concurrent data switching scheme, crossbar, in the same six-node on-chip network environment. Different data path configurations are explored in the realizations of the six-node network in order to further examine the characteristics of the CDMA NoC. The crossbar structure is realized by parallel multiplexers. The two different realizations of the six-node network which apply the CDMA scheme and crossbar scheme separately are realized in Register-Transfer Level (RTL) by using Hardware Description Language (HDL). Based on the RTL realizations, area and power costs, data transfer latencies, and the number of data wires of the two schemes is compared.**

***Keywords*—Code-Division Multiple-Access, Crossbar, Network-on-Chip**

I. Introduction

Driven by the complexity of System-on-Chip (SoC) applications, more and more components are integrated into an on-chip system. Thus, the communication among the large number of components is a challenging task to deal with. In this situation, Network-on-Chip is proposed to handle the communications in an on-chip system. Namely, the concerns of communication are separated from computation by applying a dedicated on-chip communication network in a SoC design. The published NoC structures can be classified into two categories, circuit-switched network and packet-switched network, according to data switching schemes. The circuit-switched network, such as PROPHID [1], connects the terminals in the network by allocating them a set of time or space slices on the communication links. The packet-switched networks, e.g. Æthereal [2] and Proteo [3], share the communication links among all the terminals in space and time domains by encapsulating data into packet format and delivering them through routers or switch nodes.

For performing the data transfers among a large amount of components in the future SoC, the packet-switched scheme is a more promising choice than circuit-switched scheme in terms of scalability and parallelism. However, routing issue is a main challenge of a packet-switched NoC because it affects the packet transfer latency in the network severely. One type of solutions is to optimize the routing scheme to equalize and improve data transfer latencies when data packets are transferred to their destinations via different routes. Another type of solutions is to avoid the routing issue by applying non-blocking concurrent data switching scheme which means that data transfers which have different destinations can be performed concurrently without blocking each other. In this paper, the costs and performance of applying two non-blocking concurrent data switching schemes in a NoC design are presented and compared.

A CDMA scheme [4] has been proposed to implement the non-blocking concurrent data switching scheme in a NoC design. CDMA technique applies a set of orthogonal codes to encode the data from different users before transmission in a shared communication media. Hence, it permits multiple users to use the communication media concurrently by separating the different data from different users in the code domain. The CDMA scheme presented in [4] uses the feature of multiple access of CDMA technique to transfer the data packets from different sources to their destinations directly and concurrently. The main overhead of the CDMA NoC structure is the complexity and data transfer latency caused by data encoding and decoding operations. Another type of non-blocking concurrent data transfer scheme is crossbar which avoids data encoding and decoding operations while keeping the feature of concurrent data transfers. Therefore, in order to examine the overhead of applying CDMA technique in a NoC, a crossbar structure realized by parallel multiplexers is presented and compared with the CDMA NoC. Furthermore, in this paper, the characteristics of the CDMA NoC are further examined by realizing the NoC with different data path configurations.

The rest of this paper is arranged as follows. Section II will compare the CDMA data switching scheme with the crossbar scheme in terms of data transfer principles and network structures. The realizations of the six-node CDMA NoC and crossbar NoC with different data path configurations will be presented in Section III. Section IV will present and compare the simulation results based on the RTL realizations of the six-node network. Finally, the conclusions will be drawn in Section V.

II. Comparing The CDMA and Crossbar Data Switching Schemes

In this section, data transfer principles of the CDMA data switching scheme and the crossbar scheme will be compared firstly. Then the NoC structures which apply the two data switching schemes will be compared.

A. Data Transfer Principles

As the name indicated, the CDMA data switching scheme applies the Code-Division Multiple-Access communication technique [5] to transfer data among multiple users concurrently. The principle of CDMA data transfer is illustrated in Fig.1. Each data stream from different users is encoded with a unique orthogonal spreading code at the sending end. Then the encoded data

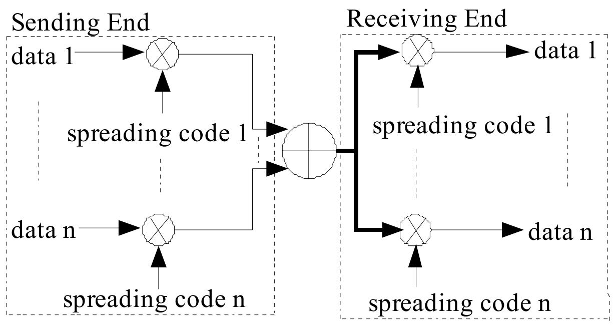


Fig. 1. CDMA Data Transfer Principle

from different users are added together for transmission in a shared communication media without interfering with each other because of the orthogonal property of the spreading code. The orthogonal property means that the normalized autocorrelation value of the spreading codes is 1, while the normalized cross-correlation value is 0. At the receiving end, each data stream from different users can be decoded from the received sum signals by using the same spreading code for encoding as illustrated in Fig.1.

Crossbar is a well-known structure for building a circuit-switched network. The data transfer principle of crossbar is illustrated in Fig.2 with a four-port switch. In the crossbar structure, an input can be connected to any outputs by optionally closing the switches between input and output lines. For correct operation, one output can be connected to at most one input.

From the introduced data transfer principles, we can see that the data transfers in the CDMA scheme and the crossbar scheme are non-blocking and concurrent because a dedicated data transfer channel can be set up from each input to its selected output without any conflicts with other channels if the selected outputs are different. Hence, the crossbar scheme is a good reference to evaluate the overhead of applying the CDMA scheme in a NoC.

B. *Applying the CDMA Scheme and Crossbar Scheme In a NoC*

In order to adapt the CDMA scheme and crossbar scheme into a NoC design, all the operations or structures in the schemes need to suit the commonly used digital circuit design. The following two paragraphs describe the methods of implementing the CDMA and crossbar schemes for NoC designs.

The key operations in the CDMA scheme are data encoding and decoding. The detailed information of implementing the encoding and decoding operations for a NoC design has been presented in [4] and briefly summarized as follows. The encoding operations are performed by XOR logic gates in the design, while the decoding operations are carried out by accumulating the received sum values into two separate parts according to the bit value of the spreading code used for decoding. Then the original data value can be restored by comparing the values in the two accumulators.

In order to realize the crossbar scheme in a NoC design, multiplexers can be used to set up the switches between input ports and output ports. An example structure of realizing crossbar among four nodes is illustrated in Fig.3. It uses four 3:1 multiplexers to build the required data switching channels. The control signal of each multiplexer is generated by the arbitration result of the input requests.

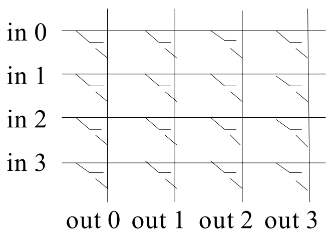


Fig. 2. Four-Port Crossbar Structure

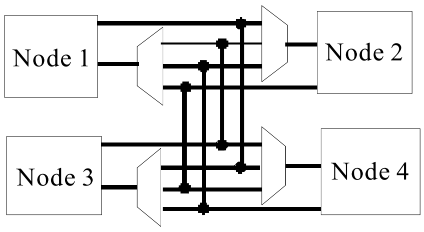


Fig. 3. Four-Node Crossbar Realized by Multiplexers

C. *NoC Structures*

The NoC structure which can apply either CDMA scheme or crossbar scheme is illustrated in Fig.4. If the CDMA scheme is applied, the NoC structure contains ‘Network Node’, ‘CDMA Transmitter’, and ‘Network Arbiter’ blocks. If the crossbar scheme is applied, we need to replace the ‘CDMA Transmitter’ and ‘Network Arbiter’ blocks with ‘Crossbar Switch’ block as illustrated in Fig.4.

‘Network Node’ block is used to connect the functional Intellectual Property (IP) blocks (‘Functional Host’) into the on-chip network. The structure and function of the ‘Network Node’ blocks for the CDMA NoC and the crossbar NoC will be presented in Section II D.

The function of ‘CDMA Transmitter’ and ‘Network Arbiter’ blocks in the CDMA NoC which have been presented in [4] will be briefly summarized in the following paragraph.

‘Network Arbiter’ takes care of informing the requested receiver node to prepare the proper spreading code for decoding, and then sending a grant signal back to the sender node. After getting the grant signal, the sender node will start to send data packets to ‘CDMA Transmitter’. In case that there are more than one sender node requesting to send data to the same receiver node, the arbiter will apply the ‘first come, first served’ principle to guarantee that only one sender is sending data to one specific receiver at a time. If different sender nodes request to send data to different receiver nodes, these requests would not interfere with each other and will be handled in parallel in the ‘Network Arbiter’. The main task of the ‘CDMA Transmitter’ block is to receive data packets from different network nodes and encode the data with the corresponding unique spreading code of the sender node. Because the nodes in the network can request data transfer randomly and independently of each other, ‘CDMA Transmitter’ also applies the ‘first come, first served’ mechanism to ensure that the data encoding and transmission are performed as soon as a data transfer request appears.

In the crossbar NoC, ‘Crossbar Switch’ block consists of multiple channel multiplexers to set up data channels between certain network nodes according to data transfer requests. The number of channel multiplexers in the ‘Crossbar Switch’ block is equal to the number of

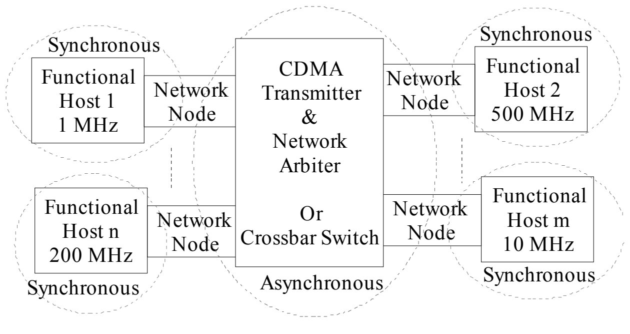


Fig. 4. The CDMA and Crossbar NoC Structure

network nodes in the NoC. Each channel multiplexer contains arbitration logic to control the selection of the output. When multiple data transfer requests from different network nodes come to a channel multiplexer simultaneously, the multiplexer will record the requests and serve one request at a time. For the requests that come at different times, a channel multiplexer will serve the requests by the principle of 'first come, first served'. After setting up the data channel, a channel multiplexer will send a grant signal back to the sender node to enable the data transfer process.

Since the crossbar NoC is exempted from data encoding and decoding operations in the CDMA NoC, it has simpler structure than the CDMA NoC. Although the CDMA and crossbar data switching schemes are non-blocking, the NoC which applies either of these two data switching schemes is not non-blocking network because of the buffers introduced in the 'Network Node' blocks.

D. *'Network Node' Structures*

The structure of 'Network Node' blocks for the CDMA NoC and the crossbar NoC are same except the different functions in 'Packet Receiver' and 'Packet Sender' sub-blocks. The block diagram of a 'Network Node' block is illustrated in Fig.5. It consists of 'Node IF', 'Tx/Rx Buffer', 'Packet Sender', and 'Packet Receiver' sub-blocks. As illustrated in Fig.5, the 'Network IF' block which belongs to the functional host is an interface block for connecting a functional host with a 'Network Node'. Because the different functional hosts may work at different clock frequencies as illustrated in Fig.4, Globally-Asynchronous Locally- Synchronous (GALS) scheme [6] scheme is applied in the network node by using synchronous design in the 'Node IF' block and using asynchronous design in the other blocks. VCI [7] or OCP [8] interface standards can be applied to transfer data between 'Network IF' and 'Node IF'. The function of the sub-blocks in a 'Network Node' will be described in the following four paragraphs.

1) 'Node IF'. This block is used to assemble the data from the functional host into packet format and send the packet to 'Tx Packet Buffer', or disassemble the received packet from 'Rx Packet Buffer' and send data to the functional host.

2) 'Tx/Rx Packet Buffer'. These two blocks are the buffers which consist of the asynchronous FIFO presented in [9]. 'Tx Packet Buffer' is used to store the data packets for transfer. The 'Rx Packet Buffer' stores and delivers the received packets from 'Packet Receiver' to 'Node IF'.

3) 'Packet Sender'. In the CDMA NoC, when 'Tx Packet Buffer' is not empty, 'Packet Sender' will fetch a data packet from the buffer. Then it will extract the

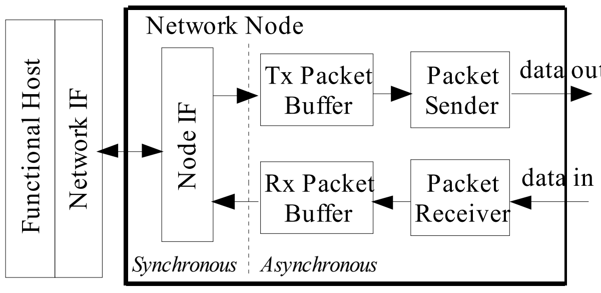


Fig. 5. The Network Node Block Diagram

destination information from the packet and send the destination address to 'Network Arbiter'. After 'Packet Sender' gets the grant signal from the arbiter, it will start to send the data packet to 'CDMA Transmitter'. In the crossbar NoC, after fetching a packet from the buffer, the 'Packet Sender' will assert a request signal to the channel multiplexer attached to the receiver node. After 'Packet Sender' gets the grant signal from the requested channel multiplexer, it will start to send the data packet to the receiver node through the multiplexer. After a data packet transfer is completed, the 'Packet Sender' needs to clear the request signal in order to release the requested channel multiplexer for serving requests from other nodes.

4) 'Packet Receiver'. In the CDMA NoC, this block will wait for the sender information from 'Network Arbiter' to select the proper spreading code for decoding. After the spreading code for decoding is ready, the receiver will start to receive and decode the data from 'CDMA Transmitter', and then send the decoded data to 'Rx Packet Buffer' in packet format. In the crossbar NoC, this block will wait for the request from the multiplexer block attached with it. When the request comes, the 'Packet Receiver' block will first receive the data packet and then deliver it to 'Rx Packet Buffer' block.

III. Realizations of a Six-Node NoC

In order to compare the performance of the CDMA and crossbar data switching schemes in a NoC surrounding, a six-node network is built for simulation purpose. This simulation network has the same configurations with the network presented in [4] except that different data path configurations of the network are explored in this work to further examine the characteristics of the CDMA NoC. The different realizations of the six-node simulation network will be addressed and compared in this section.

A. *Simulation Network Setup*

The six-node simulation network is illustrated in Fig.6. Each functional host works in different clock frequencies. The interface standard applied in the network is Basic VCI (BVCI) [7] standard. Three hosts act as initiators and the other three act as targets, as denoted by the labels 'I' and 'T' respectively in the 'Network IF' blocks. The initiator hosts can generate requests to any target hosts, while the target hosts can generate responses only for the received requests passively. The basic data unit transferred in the network is a data packet composed of one header cell and several data cells. The number of data cells in a packet varies from one to three, while the width of each packet cell is fixed at 32 bits. As illustrated in Fig.6, the interconnection structure in the network is either 'CDMA Transmitter' and 'Network Arbiter' blocks or the 'Crossbar Switch' block. For the CDMA scheme

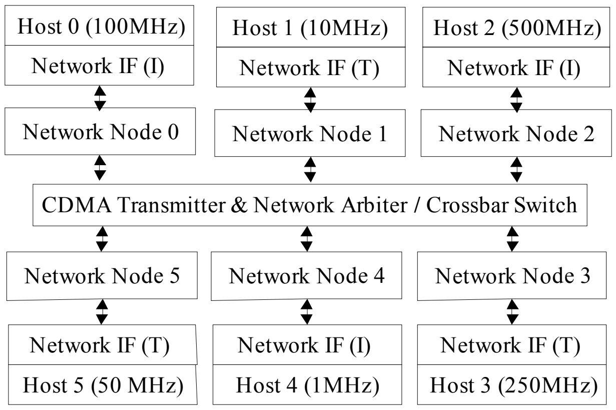


Fig. 6. Six-Node Simulation Network

realization, the spreading codes used in the network are six 8-bit Walsh codes. For the crossbar scheme realization, the 'Crossbar Switch' block is composed by six multiplexers similar with the example illustrated in Fig.3.

B. Data Path Configuration

The CDMA NoC presented in [4] delivers one data bit from the sender to the receiver at one time. Namely, the data path was set to 1-bit. In the CDMA NoC, since one original data bit will be spread into S bits after encoding, the degree of data transfer parallelism between the 'CDMA Transmitter' and 'Packet Sender/Receiver' blocks affects the data transfer latency largely. Therefore, increasing the number of data bits encoded and delivered via 'CDMA Transmitter' at one time can reduce the data transfer latency in the CDMA NoC. However, increasing the data processing and delivering parallelism will incur larger area cost. Therefore, in order to figure out the trade-off character between the data path width and the area cost, four different data path widths are explored in both the CDMA NoC and the crossbar NoC realizations. According to the number of data bit transferred from a 'Packet Sender' to a 'Packet Receiver' through 'CDMA Transmitter' in the CDMA NoC or through 'Crossbar Switch' in the crossbar NoC, the data path configurations are named as 1-bit, 8-bit, 16-bit, and 32-bit schemes.

C. Area Costs

Both the synchronous and asynchronous designs in the network illustrated in Fig.6 are realized in RTL using VHDL in order to suit the conventional synchronous design tools and flow. The same 0.18μm standard cell library used to synthesize the realizations of CDMA and crossbar NoCs. The data width and buffer depth in a 'Network Node' block are set to 32 bits and 4 respectively. The logic-gate area costs of the components in the CDMA and crossbar networks under different data path configurations are listed in Table I and Table II respectively. Since the designs of 'Node IF' and 'Tx/Rx Packet Buffer' blocks are same in the two NoC realizations, the area costs of these two blocks are same in both realizations. The big difference appears in the area cost of 'Packet Receiver' block. The 'Packet Receiver' block in the crossbar NoC is much smaller than the one in the CDMA NoC because it does not have any data decoding circuits. Because there is no data encoding in the crossbar NoC, the area cost of one single channel multiplexer is much smaller than the total area of the 'CDMA Transmitter' and 'Network Arbiter' blocks in the CDMA NoC. However, in the crossbar NoC, each node needs a channel multiplexer block. Thus, the area cost of all the multiplexers will increase linearly when the number of nodes increases in the crossbar NoC.

Table I. Area Cost of CDMA NoC Components

		Area (μm^2)			
		1-bit	***8-bit***	***16-bit***	***32-bit***
Node IF	Target	18825.2			
	Initiator	45674.5			
Tx/Rx Packet Buffer		71778.3			
Packet Sender		17707.0	17756.2	17805.3	11321.3
Packet Receiver		23253.0	86241.3	161390.6	311623.7
CDMA Transmitter		10338.3	46710.8	90935.3	178368.5
Network Arbiter		11014.1			

Table II. Area Cost of Crossbar NoC Components

		Area (μm^2)			
		1-bit	***8-bit***	***16-bit***	***32-bit***
Node IF	Target	18825.2			
	Initiator	45674.5			
Tx/Rx Packet Buffer		71778.3			
Packet Sender		17813.5	17862.6	17911.8	18092.0
Packet Receiver		11710.5	10653.7	10833.9	10989.6
Channel MUX		2773.0	3821.6	5136.4	7761.9

Table III. Total Area Cost of The Two Six-Node NoC

	Area (μm^2)			
	1-bit	***8-bit***	***16-bit***	***32-bit***
CDMA NoC	1331118.1	1745518.6	2247491.5	3209482.3
Crossbar NoC	1251684.4	1251938.3	1261252.6	1278799.9

The total area costs of the two six-node network realizations under different data path configurations are listed in Table III. We can see that the area cost of the CDMA NoC becomes 2.4 times larger when the data path width increases from 1-bit to 32-bit. Whereas, the area cost of the crossbar NoC only increases 2.2% in the same situation. Therefore, introducing CDMA technique incurs a large area overhead in the six-node simulation network.

D. Number of Data Wires

By avoiding data encoding and decoding schemes in the CDMA NoC, the crossbar NoC has smaller area cost by setting up direct connections from each input to each output. However, these direct connections cause a large overhead of the number of data wires in the crossbar NoC. This sub-section will address and compare the number of data wires in the CDMA NoC and crossbar NoC.

The number of data wires in the crossbar NoC refers to the number of data wires between 'Network Node' blocks and channel multiplexer blocks. The equation for calculating the number of data wires in the crossbar NoC is given in (1). In (1), parameter 'n' refers to the number of network nodes in the NoC, and parameter 'w' refers to the data path width. The first term of (1) represents the data wires for connecting the data output port of each node to all the other nodes via channel multiplexers in the network. The second term of (1) refers to the data wires between the data output port of a channel multiplexer and its attached network node.

Table IV. Number of Data Wires

		Number of Data Wires		
		n=6, s=8	n=15, s=16	n=31, s=32
Crossbar NoC	w = 1	36	225	961
	w = 8	288	1800	7688
	w = 16	576	3600	15376
	w = 32	1152	7200	30752
CDMA NoC	w = 1	30	79	191
	w = 8	240	632	1528
	w = 16	480	1264	3056
	w = 32	960	2528	6112

$$N_{crossbar\_noc} = n \cdot (n-1) \cdot w + n \cdot w = w \cdot n^2 \qquad (1)$$

$$N_{cdma\_noc} = n \cdot w + w \cdot s \cdot \log_2 n \qquad (2)$$

In the CDMA NoC, the number of data wires refers to the number of data wires between 'Network Node' blocks and 'CDMA Transmitter' block. The equation for calculating the number of data wires in the CDMA NoC is given in (2). In (2), the meaning of parameters 'n' and 'w' is same with the parameters in (1). The parameter 's' refers to the bit width of spreading codes used in the CDMA NoC. The first term in (2) represents the data wires for connecting data output port of each network node with data input port of 'CDMA Transmitter' block. The number of data wires from the data output port of 'CDMA Transmitter' is represented by the second term in (2). In the CDMA NoC, each data bit to be transferred will be extended into s bits by the s-bit spreading code after data encoding step. Each bit of the s-bit encoded data is called a data chip. Then each data chip from different network nodes will be added together in the 'CDMA Transmitter' block. The sum value for n data chips from n network nodes can be represented by $\log_2 n$ bits. Therefore, the 'CDMA Transmitter' need to use $s \cdot \log_2 n$ bits to represent the all the sum values of s-bit encoded data. Hence, for transferring w bits data at one time, we need $w \cdot s \cdot \log_2 n$ data wires as the output of 'CDMA Transmitter' block.

Table IV lists the numbers of data wires for the crossbar NoC and the CDMA NoC with different number of network nodes and data path widths. From Table IV, we can see that the presented crossbar NoC structure will incur a huge amount of data wires in order to obtain the feature of concurrent data transfer as the CDMA NoC does. This large number of data wires of the crossbar NoC is a strong weakness for its application in an on-chip system because the number of network nodes in a future SoC will be very large. Therefore, the CDMA NoC has the advantage of utilizing less data wires to achieve concurrent data transfer feature by comparing with the crossbar NoC.

IV. Simulation Results

The simulation is performed in gate-level after the RTL realizations of the simulation network were synthesized. The functional hosts which are not realized with any designs are simulated by adding stimulus signals on each 'Network IF' block according to the BVCI standard. The data transactions performed in the simulation are listed in Table V. Each data transaction consists of one request packet from an initiator host to a target host and one corresponding response packet from the target host to the initiator host. The following paragraphs in this section will present and compare the data transfer latency and power consumption figures of the CDMA NoC and crossbar NoC obtained from the gate-level simulation.

Table V. Data Transaction Specification

Initiator Node	Target Node	Number of Transactions	Packet Length (cell)	
			Request Packet	Response Packet
Node 0	Node 1	2	4, 3	2, 3
	Node 3	2	3, 4	3, 2
	Node 5	1	4	2
Node 2	Node 1	1	4	2
	Node 3	1	3	3
	Node 5	2	4, 3	2, 3
Node 4	Node 1	2	3, 4	3, 2
	Node 3	1	3	3
	Node 5	1	4	2

Table VI. Synchronous Transfer Latency

Interface Type	Latency of sending data to 'Network Node'	Latency of receiving data from 'Network Node'
BVCI Initiator	8 local clock cycles + 2.5 ns	8 local clock cycles +3.2 ns
BVCI Target	4 local clock cycles + 2.5 ns	4 local clock cycles + 3.1 ns

Because the GALS scheme is applied in the network, the data transfer latency in the simulation network is separated into two parts, Synchronous Transfer Latency (STL) and Asynchronous Transfer Latency (ATL). The STL refers to the data transfer latency between a functional host and the network node attached to it. STL depends on the local clock and the type of interface. The measured STL values are listed in Table VI. The constant values in Table VI are caused by the handshakes in the asynchronous domain. They are independent of the local clock rate but belong to the synchronous transfer processes. Therefore they are counted as a part of STL. Because the 'Node IF' block and 'Tx/Rx Packet Buffer' block of the crossbar NoC are same as the blocks in the CDMA NoC, the STL values are also same with the values of the CDMA NoC.

The ATL refers to the data transfer latency of transferring data packets from one network node to the other node using asynchronous four-phase handshake protocol in the simulation network. The ATL of the CDMA NoC and the crossbar NoC consists of three portions: Packet Loading Latency (PLL), Packet Transfer Latency (PTL), and Packet Storing Latency (PSL). The concept of those ATL portions is illustrated in Fig.7 with an example where 'Network Node 1' sends data packets to 'Network Node 2'. The black arrows in Fig.7 represent the packet transfer direction, whereas the three portions of the ATL are marked by grey arrows in Fig.7 and explained in the following three paragraphs.

1) Packet Load Latency (PLL). This is the time used by the 'Packet Sender' block in a 'Network Node' to fetch one data packet from 'Tx Packet Buffer' and prepare to assert a request signal to the corresponding channel multiplexer in the crossbar NoC or to the 'Network Arbiter' in the CDMA NoC.

2) Packet Transfer Latency (PTL). This latency refers to the time used to transfer one data packet from the 'Packet Sender' of the sender node to the 'Packet Receiver' of the receiver node through 'Crossbar Switch' or 'CDMA Transmitter' using a handshake protocol.

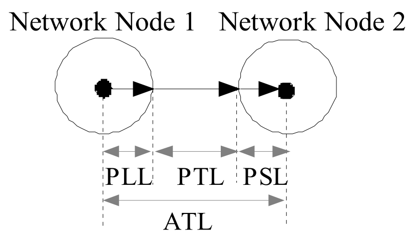


Fig. 7. Asynchronous Transfer Latency

3) Packet Storing Latency (PSL). After the receiver node receives a data packet, it will spend a certain amount of time to store the received data packet in 'Rx Packet Buffer'. This time duration is measured as PSL.

The values of ATL portions of the CDMA NoC and the crossbar NoC measured in the simulations are listed in Table VII and Table VIII respectively. From the listed figures, we can see that PTL increases as the packet length increases. This is because the data cells in a packet are sent in a serial manner. Thus, more data cells need more transmission time. The PLL and PSL values are not affected by the packet length because the data cells in a packet are loaded or stored in a parallel manner. From Table VII and Table VIII, we also can see that the PLL and PSL values are reduced in the crossbar NoC because of the less complexity in its 'Packet Sender' and 'Packet Receiver' blocks. The PTL values of the crossbar NoC are smaller than the PTL values of the CDMA NoC because the latency caused by data encoding and decoding schemes is exempted from the crossbar NoC. However, the differences are getting smaller when the data path width increases. For example, the PTL values of the crossbar NoC is around 30% of the PTL values of the CDMA NoC when the data path is 1 bit, whereas, this figure changes to 49% when the data path width is increased to 32 bits.

Power costs of the two NoC designs are also estimated during the simulations by back annotating the switching activities to the gate-level netlists. The estimated dynamic power costs of the two NoCs with different data path widths are listed in Table IX. From the table, we can see that the CDMA NoC consumes more dynamic power than the crossbar NoC due to the data encoding and decoding schemes. When the data path width is 8-bit, the CDMA NoC has similar power cost with the crossbar NoC. Therefore, 8-bit CDMA NoC is a good choice in terms of power consumption.

Table VII. ATL Portion Values of the CDMA NoC

		1 data cell	2 data cells	3 data cells
PLL (ns)		5.7	5.7	5.7
PTL (ns)	***1-bit***	384.6	768.9	1153.7
	8-bit	45.9	88.4	130.9
	16-bit	26.2	49.0	71.8
	32-bit	14.7	26.0	37.8
PSL (ns)		5.5	5.5	5.5

Table VIII. ATL Portion Values of the Crossbar NoC

		1 data cell	2 data cells	3 data cells
PLL (ns)		3.0	3.0	3.0
PTL (ns)	***1-bit***	112.5	211.8	354.4
	8-bit	17.3	32.7	47.5
	16-bit	10.6	19.4	27.8
	32-bit	7.5	12.7	17.9
PSL (ns)		4.7	4.7	4.7

Table IX. Dynamic Power Costs of the Two NoC Designs

	Dynamic Power Costs (mW)			
	1-bit	***8-bit***	***16-bit***	***32-bit***
CDMA NoC	19.340	6.563	7.331	7.332
Crossbar NoC	6.559	6.558	6.558	6.558

V. CONCLUSIONS

Two non-blocking concurrent data switching schemes, CDMA scheme and crossbar scheme, for on-chip networks are presented and compared in this paper. Both data switching schemes are examined and compared in the same six-node network environment. The crossbar structure is realized by using one multiplexer for each node in the network to set up concurrent data transfer channels. By comparing the two data switching schemes in the same network environment, the overhead of data encoding and decoding operations in the CDMA NoC are examined. Different data path configurations are also explored in the network realizations in order to further examine the characteristics of the CDMA NoC.

According to the realization and simulation results, the six-node crossbar NoC has a smaller area than the six-node CDMA NoC with four different data path widths. The asynchronous data transfer latency values in the CDMA NoC is reduced nearly 30 times when the data path width is increased from 1-bit to 32-bit. In the realized six-node simulation network, the overhead of data encoding and decoding operations in the CDMA NoC causes around 50% larger asynchronous data transfer latency by comparing with the figures in the crossbar NoC. However, the crossbar structure suffers from the drawback of requiring a huge number of data wires in the NoC when the number of network nodes is large. The CDMA NoC can reduce the number of data wires while keeping the feature of concurrent data transfers. When the data path width is set to 8-bit, the six-node CDMA NoC has similar dynamic power cost and area cost with the crossbar NoC. Hence, 8-bit CDMA NoC is a good option for applying the CDMA scheme in a NoC design.

REFERENCES

[1] J. A. J. Leijten, J. L. van Meerbergen, A. H. Timmer, and J. A. G. Jess, "PROPHID: A Data-Driven Multi-Processor Architecture for High- Performance DSP," *Proceedings of the 1997 European Design & Test Conference*, Mar. 1997.

[2] K. Goossens, J. Dielissen, and A. Radulescu, "Æthereal network on chip: concepts, architectures, and implementations", *IEEE Design & Test of Computers*, Volume 22, Issue 5, Sept.-Oct. 2005.

[3] D. Sigüenza-Tortosa, T. Ahonen, and J. Nurmi, "Issues in the Development of a Practical NoC: the Proteo Concept," *Integration, the VLSI jounal*, Volume 38, Issue 1; 2004.

[4] X. Wang, and J. Nurmi, "An On-Chip CDMA Communication Network", *Proceedings of 2005 International Symposium on System-on-Chip*, Nov. 2005.

[5] A. J. Viterbi, *CDMA: Principles of Spread Spectrum Communications*, Addison-Wesley Publishing Company, 1995.

[6] D. M. Chapiro, "Globally-Asynchronous Locally-Synchronous Systems,"*PhD thesis*, Stanford University; Oct. 1984.

[7] VSI Alliance, *Virtual Component Interface Standard v 2.* April 2001.

[8] OCP-IP Association; *Open Core Protocol Specification*; 2001.

[9] X. Wang, and J. Nurmi, "A RTL Asynchronous FIFO Design Using Modified Micropipeline", *Proceedings of 10th Biennial Baltic Electronics Conference*, Oct. 2006.

PUBLICATION 8

X. Wang, T. Ahonen, and J. Nurmi, “Applying CDMA Technique to Network-on-Chip”, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Volume 15, Number 10, pages 1091-1100, October 2007.

Applying CDMA Technique to Network-on-Chip

Xin Wang, Tapani Ahonen, and Jari Nurmi

Abstract*—The issues of applying the code-division multiple access (CDMA) technique to an on-chip packet switched communication network are discussed in this paper. A packet switched network-on-chip (NoC) that applies the CDMA technique is realized in register-transfer level (RTL) using VHDL. The realized CDMA NoC supports the globally-asynchronous locally-synchronous (GALS) communication scheme by applying both synchronous and asynchronous designs. In a packet switched NoC, which applies a point-to-point connection scheme, e.g., a ring topology NoC, data transfer latency varies largely if the packets are transferred to different destinations or to the same destination through different routes in the network. The CDMA NoC can eliminate the data transfer latency variations by sharing the data communication media among multiple users concurrently. A six-node GALS CDMA on-chip network is modeled and simulated. The characteristics of the CDMA NoC are examined by comparing them with the characteristics of an on-chip bidirectional ring topology network. The simulation results reveal that the data transfer latency in the CDMA NoC is a constant value for a certain length of packet and is equivalent to the best case data transfer latency in the bidirectional ring network when data path width is set to 32 bits.

***Index Terms*—Code-division multiple access (CDMA), integrated circuit (IC) design, network-on-chip (NoC).**

I. Introduction

As MORE and more components are integrated into an on-chip system, communication issues become complicated. Network-on-chip (NoC) is proposed to solve the on-chip communication problem by separating the concerns of communication from computation. The idea of NoC is to construct an on-chip communication network to perform data transfers among a large number of system components. The NoC structures that have been proposed can be roughly sorted into two categories, circuit switched network and packet switched network, according to their data switching modes. SoCBUS architecture [1], a mesh on-chip network, is an example of a circuit switched network that uses packet connected circuit scheme to allocate time or space slices on the switch links among the terminals in the network. Æthereal NoC [2] and Proteo NoC [3] are examples of the packet switched category. Æthereal NoC applies the combined guaranteed service and best-effort routers to transfer data packets in the network. In Proteo NoC, the components in the system are connected through network nodes and hubs. The network topology and data links in Proteo NoC can be customized and optimized for a specific application. Circuit-switched networks will face the problem of scalability and parallelism if they are applied in a future on-chip system which contains hundreds of functional intellectual property (IP) blocks. The packet switched network can overcome the shortcomings of the circuit switched network by dividing data streams into packets and routing packets to their destinations node by node. However, in a packet switched network that applies multihop point-to-point (PTP) connection scheme as in [2] and [3], the packet transfer latency will vary largely when data packets are transferred to different destinations or to the same destination via different routes in the network. Hence, the upper bound of the packet transfer latency is determined by the worst case scenario.

In order to eliminate variance of data transfer latency and complexity incurred by routing issues in a PTP connected NoC, an on-chip network which applies a code-division multiple access (CDMA) technique is introduced in this paper. As one of the spread-spectrum techniques, the CDMA technique [4] has been widely used in wireless communication systems because it has great bandwidth efficiency and multiple access capability. The CDMA technique applies a set of orthogonal codes to encode the data from different users before transmission in a shared communication media. Therefore, it permits multiple users to use the communication media concurrently by separating data from different users in the code domain. Hence, the CDMA NoC proposed in this paper can transfer data packets from different sources to their destinations directly and concurrently. Consequently, the large variance of data transfer latencies in a PTP connected NoC is eliminated in the CDMA NoC. The constant data transfer latency in the CDMA NoC is helpful for providing a guaranteed communication service for an on-chip system.

The rest of this paper is arranged as follows. In Section II, issues with applying CDMA technique into an on-chip network will be discussed. Section III presents the structure of the CDMA NoC. The realization of the basic components in the CDMA NoC is presented in Section IV. A six-node CDMA NoC is presented in Section V in order to examine characteristics of the CDMA NoC by comparing it with a PTP connected NoC. Finally, conclusions are drawn in Section VI.

II. Applying CDMA Technique to NoC

The principle of the CDMA technique is illustrated in Fig. 1. At the sending end, the data from different senders are encoded using a set of orthogonal spreading codes. The encoded data from different senders are added together for transmission without interfering with each other because of the orthogonal property of spreading codes. The orthogonal property means that the normalized autocorrelation value and the cross-correlation value of spreading codes are 1 and 0, respectively. Autocorrelation of spreading codes refers to the sum of the products of a spreading code with itself, while cross-correlation refers to the sum of the products of two different spreading

Manuscript received May 30, 2006; revised April 15, 2007.

The authors are with the Institute of Digital and Computer Systems, Tampere University of Technology, 33101 Tampere, Finland (e-mail: xin.wang@tut.fi; tapani.ahonen@tut.fi; jari.nurmi@tut.fi).

Digital Object Identifier 10.1109/TVLSI.2007.903914

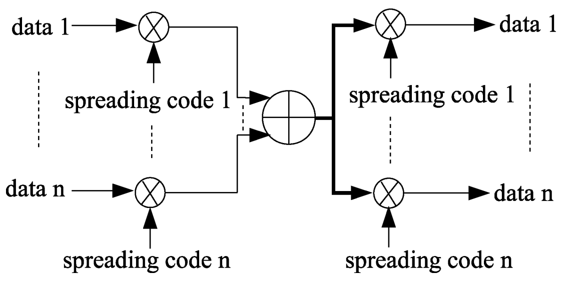


Fig. 1. CDMA technique principle.

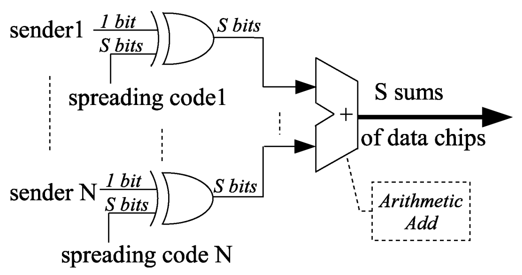


Fig. 2. Digital CDMA encoding scheme.

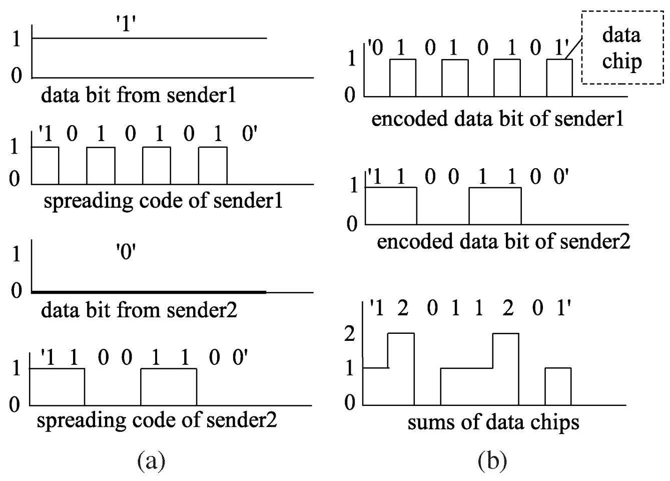


Fig. 3. Data encoding example.

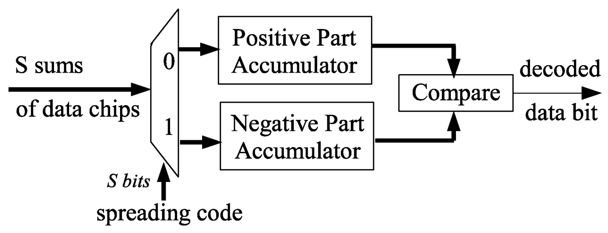


Fig. 4. Digital CDMA decoding scheme.

codes. Because of the orthogonal property, at the receiving end, the data can be decoded from the received sum signals by multiplying the received signals with the spreading code used for encoding. The following three subsections will discuss the issues related to apply the CDMA technique in an NoC.

A. Digital Encoding and Decoding Scheme

Several on-chip bus schemes that apply the CDMA technique have been presented in [5]–[8]. Those schemes are implemented by analog circuits, namely, the encoded data are represented by the continuous voltage or capacitance value of the circuits. Therefore, the data transfers in the analog bus are challenged by the coupling noise, clock skew, and the variations of capacitance and resistance caused by circuit implementation [8]. In order to avoid the challenges faced by the analog circuit implementation, digital encoding and decoding schemes developed for the CDMA NoC are illustrated in Figs. 2 and 4, respectively. In the encoding scheme illustrated in Fig. 2, data from different senders fed into the encoder bit by bit. Each data bit will be spread into S bits by XOR logic operations with a unique S-bit spreading code as illustrated in Fig. 2. Each bit of the S-bit encoded data generated by XOR operations is called a data chip. Then, the data chips which come from different senders are added together arithmetically according to their bit positions in the S-bit sequences. Namely, all the first data chips from different senders are added together and all the second data chips from different senders are added together, and so on. Therefore, after the add operations, we will get S sum values of S-bit encoded data. Finally, as proposed in [9], binary equivalents of the S sum values are transferred to the receiving end. An example of encoding two data bits from two senders is illustrated in Fig. 3 in order to illustrate the proposed encoding scheme in more detail. Fig. 3(a) illustrates two original data bits from different senders and two 8-bit spreading codes. The top two figures in Fig. 3(b) illustrate the results after data encoding (XOR operations) for the original data bits. The bottom figure in Fig. 3(b) presents the eight sum values after add operations. Then the binary equivalents of each sum value will be transferred to the receiving end. In this case, two binary bits are enough to represent the three possible different decimal sum values, "0," "1," and "2." For example, if a decimal sum value "2" needs to be transferred, we need to transfer two binary digits "10."

The digital decoding scheme applied in the CDMA NoC is depicted in Fig. 4. The decoding scheme accumulates the received sum values into two separate parts, a positive part and a negative part, according to the bit value of the spreading code used for decoding. For instance, as illustrated in Fig. 4, the received first sum value will be put into the positive accumulator if the first bit of the spreading code for decoding is "0," otherwise, it will be put into the negative accumulator. The same selection and accumulation operations are also performed on the other received sum values. The principle of this decoding scheme can be explained as follows. If the original data bit to be transferred is "1," after the XOR operations in the encoding scheme illustrated in Fig. 2, it can only contribute nonzero value to the sums of data chips when a bit of spreading code is "0." Similarly, the 0-value original data bit can only contribute nonzero value to the sums of data chips when a bit of spreading code is "1." Therefore, after accumulating the sum values according to the bit values of the spreading code, either the positive part or negative part is larger than the other if the spreading codes are orthogonal and balance. Hence, the original data bit can be decoded by comparing the values between the two accumulators. Namely, if the value of the positive accumulator is larger than the value in the negative accumulator, the original data bit is "1"; otherwise, the original data bit is "0."

B. Spreading Code Selection

As discussed in Section II-A, the proposed decoding scheme requires the spreading codes used in the CDMA NoC to have both the orthogonal and balance properties. The orthogonal property has been explained in the first paragraph of Section II. The balance property means that the number of bit “1” and bit “0” in a spreading code should be equal. Several types of spreading codes have been proposed for CDMA communication, such as Walsh code, M-sequence, Gold sequence, and Kasami sequence, etc. [10]. However, only Walsh code [10] has the required orthogonal and balance properties. Therefore, Walsh code family is chosen as the spreading code library for the CDMA NoC. In an S-bit ($S = 2^N$, integer $N > 1$) length Walsh code set, there are $S - 1$ sequences that have both the orthogonal and balance properties. Hence, the proposed CDMA NoC can have at most $S - 1$ network nodes. The length of applied Walsh code set should be kept as small as possible according to the number network nodes. The purpose is to reduce the number of data chips generated during data encoding operations as illustrated in Fig. 2. For example, if there are six nodes in the CDMA NoC, the 8-bit Walsh code set should be used instead of a longer Walsh code set.

C. Spreading Code Protocol

In a CDMA network, if multiple users use the same spreading code to encode their data packets for transmission simultaneously, the data to be transferred will interfere with each other because of the loss of orthogonal property among the spreading codes. This situation is called spreading code conflict, which should be avoided. Spreading code protocol is a policy used to decide how to assign and use the spreading codes in a CDMA network in order to eliminate or reduce the possible spreading code conflicts during the communication processes. Several spreading code protocols have been presented for CDMA packet radio network [11], [12] and will be shortly introduced in the following six paragraphs.

1) *Common Code Protocol (C protocol)*: All users in the network use the same spreading code to encode their data packets to be transferred.
2) *Receiver-Based Protocol (R protocol)*: Each user in the network is assigned a unique spreading code used by the other users who want to send data to that user.
3) *Transmitter-Based Protocol (T protocol)*: The unique spreading code allocated to each user is used by the user himself to transfer data to others.
4) *Common-Transmitter-Based Protocol (C-T protocol)*: The destination address portion of a data packet is encoded using C protocol, whereas, the data portion of a packet is encoded using T protocol.
5) *Receiver-Transmitter-Based Protocol (R-T protocol)*: It is the same as the C-T protocol except that the destination address portion of a data packet is encoded using R protocol.
6) *Transmitter-Receiver-Based Protocol (T-R protocol)*: Two unique spreading codes are assigned to each user in the network, and then a user will generate a new spreading code from the assigned two unique codes for its data encoding.

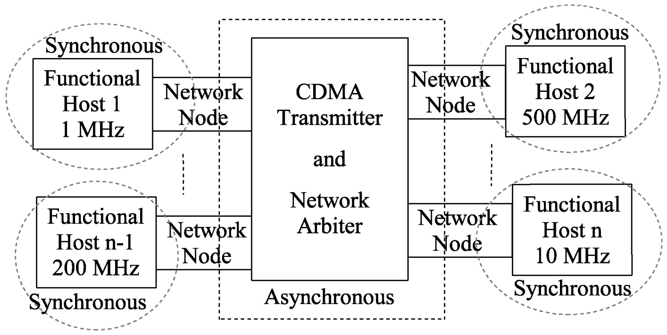


Fig. 5. Proposed CDMA NoC structure.

Among the introduced spreading code protocols, only T protocol and T-R protocol are conflict-free if the users in the network send data to each other randomly. Because the T-R protocol has the drawback of using a large amount of spreading codes and complicated decoding scheme, T protocol is preferred in the CDMA NoC. However, if T protocol is applied in the network, a receiver cannot choose the proper spreading code for decoding because it cannot know who is sending data to it. In order to solve this problem, an arbiter-based T protocol (A-T protocol) is developed for the CDMA NoC. In a CDMA NoC which applies A-T protocol, each user is assigned with a unique spreading code for data transfer. When a user wants to send data to another user, he will send the destination information of the data packet to the arbiter before starting data transmission. Then, the arbiter will inform the requested receiver to prepare the corresponding spreading code for data decoding according to the sender. After the arbiter has got the acknowledge signal from the receiver, it will send an acknowledge signal back to the sender to grant its data transmission. If there is more than one user who wants to send data to the same receiver, the arbiter will grant only one sender to send data at a time. Therefore, by applying the proposed A-T protocol, spreading code conflicts in the CDMA NoC can be eliminated.

III. CDMA NoC Structure

The proposed CDMA NoC is a packet switched network that consists of “Network Node,” “CDMA Transmitter,” and “Network Arbiter” blocks as illustrated in Fig. 5. The functional IP blocks (functional hosts) are connected to the CDMA NoC through individual “Network Node” blocks. The CDMA communications in the network are performed by “CDMA Transmitter” and “Network Arbiter” blocks. Because the different functional hosts may work at different clock frequencies as illustrated in Fig. 5, coordinating the data transfers among different clock domains would be a problem. A globally-asynchronous locally-synchronous (GALS) scheme [13] has been proposed as a solution for this problem. Applying the GALS scheme to the CDMA NoC means that the communications between each functional host and its network node use local clock frequency, while the communications between network nodes through the CDMA network are asynchronous. In order to support the GALS scheme, both synchronous and asynchronous circuits are applied in the design. The three types of components in the CDMA NoC will be presented in the following three subsections.

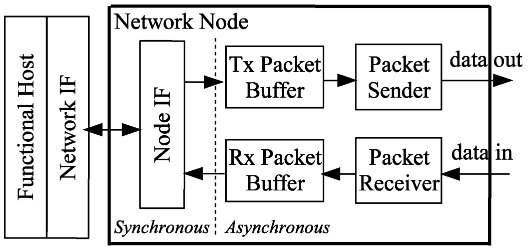


Fig. 6. Block diagram of the network node in CDMA NoC.

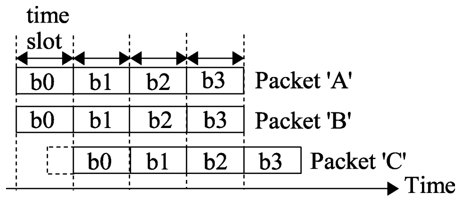


Fig. 7. Bit-synchronous transfer scheme.

A. Network Node

The block diagram of the "Network Node" in the CDMA NoC is illustrated in Fig. 6, where the arrows represent the flows of data packets. In Fig. 6, the "Network IF" block, which belongs to the functional host, is an interface block for connecting a functional host with a "Network Node" through VCI [14] or OCP interface standard [15]. GALS scheme is realized in "Network Node" block by using synchronous design in the "Node IF" subblock and using asynchronous design in the other subblocks. The function of the subblocks in a "Network Node" will be described in the following four paragraphs.

1) *Node IF*: This block is used to receive data from the "Network IF" block of a functional host through the applied VCI or OCP standard. Then it will assemble the received data into packet format and send the packet to "Tx Packet Buffer," or disassemble the received packet from "Rx Packet Buffer" and send the extracted data to the functional host.
2) *Tx/Rx Packet Buffer*: These two blocks are buffers that consist of the asynchronous first-input–first-output (FIFO) presented in [16]. "Tx Packet Buffer" is used to store the data packets from "Node IF" block, and then deliver the packets to "Packet Sender" block. The "Rx Packet Buffer" stores and delivers the received packets from "Packet Receiver" to "Node IF."
3) *Packet Sender*: If "Tx Packet Buffer" is not empty, "Packet Sender" will fetch a data packet from the buffer by an asynchronous handshake protocol. Then it will extract the destination information from the fetched packet and send the destination address to "Network Arbiter." After "Packet Sender" gets the grant signal from the arbiter, it will start to send the data packet to "CDMA Transmitter."
4) *Packet Receiver*: After system reset, this block will wait for the sender information from "Network Arbiter" to select the proper spreading code for decoding. After the spreading code for decoding is ready, the receiver will send an acknowledge signal back to "Network Arbiter" and wait to receive and decode the data from "CDMA Transmitter," and then send the decoded data to "Rx Packet Buffer" in packet format.

B. Network Arbiter

"Network Arbiter" block is the core component to implement the A-T spreading code protocol presented in Section II-C. By applying A-T spreading code protocol, every sender node cannot start to send data packets to "CDMA Transmitter" until it gets the grant signal from "Network Arbiter." "Network Arbiter" takes charge of informing the requested receiver node to prepare the proper spreading code for decoding and sending a grant signal back to the sender node. In the case that there are more than one sender nodes requesting to send data to the same receiver node simultaneously or at different times, the arbiter will apply a "round-robin" arbitration scheme or the "first-come first-served" principle, respectively, to guarantee that there is only one sender sending data to one specific receiver at a time. However, if different sender nodes request to send data to different receiver nodes, these requests would not block each other and will be handled in parallel in the "Network Arbiter." The "Network Arbiter" in the CDMA NoC is different from the arbiter used in a conventional bus. The reason is that the "Network Arbiter" here is only used to set up spreading codes for receiving and it handles the requests in parallel in the time domain. However, a conventional bus arbiter is used to allocate the usage of the common communication media among the users in the time-division manner.

C. CDMA Transmitter

The "CDMA Transmitter" block takes care of receiving data packets from network nodes and encoding the data to be transferred with the corresponding unique spreading code of the sender node. Although this block is realized using asynchronous circuits, it applies a bit-synchronous transfer scheme. It means that the data from different nodes will be encoded and transmitted synchronously in terms of data bits rather than any clock signals. In Fig. 7, the principle of the referred bit-synchronous transfer scheme is illustrated by a situation that network nodes "A" and "B" send data packets to "CDMA Transmitter" simultaneously and node "C" sends a data packet later than "A" and "B." In this situation, the data packet from node "A" will be encoded and transmitted together with the data packet from node "B" synchronously in terms of each data bit. When the data packet from node "C" arrives at a later time point, the transmitter will handle the data bit of "Packet C" together with the data bits of packet "A" and "B" at the next start point of the time slot for bit encoding and transmitting processes. The dot-line frame at the head of the "Packet C" in Fig. 7 is used to illustrate the waiting duration if the "Packet C" arrived in the middle of the time slot for handling the previous data bit. The time slot for handling a data bit is formed by a four-phase handshake process. The bit-synchronous transfer scheme can avoid the interferences caused by the phase offsets among the orthogonal spreading codes if the data bits from different nodes are encoded and transmitted asynchronously with each other. Because the nodes in the network can request data transfer randomly and independently of each other, "CDMA

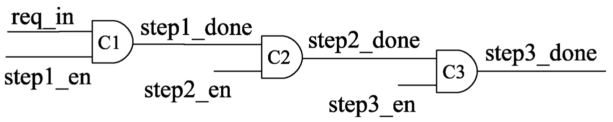


Fig. 8. C-element control pipeline.

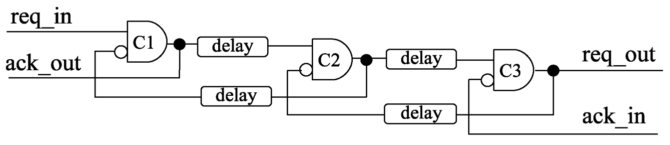


Fig. 9. Micropipeline control logic.

Transmitter" applies the "first come, first served" mechanism to ensure that the data encoding and transmission are performed as soon as there is data transfer request.

IV. Realization

Two issues related with realizing the CDMA NoC are addressed in this section. One issue is about asynchronous design realization. Another is the configuration of the data path in the CDMA NoC.

A. Asynchronous Design

As illustrated in Fig. 5 and addressed in Section III-A, the asynchronous blocks in the CDMA NoC include the "CDMA Transmitter," "Network Arbiter," "Tx/Rx Packet Buffer," and "Packet Receiver/Sender" blocks. The important part of the asynchronous design of these blocks is the control logic. Since the "CDMA Transmitter" and "Network Arbiter" blocks are data-path centric blocks, the control logic used in these blocks is composed by a straightforward C-element pipeline as illustrated in Fig. 8. Each stage in the C-element pipeline is enabled by the enable signals generated from data completion detection circuits. The control token will be passed from one stage to the next one through each C-element in the pipeline. The control logic used in the "Tx/Rx Packet Buffer" and "Packet Receiver/Sender" blocks bases on the micropipeline control logic presented in [17] and illustrated in Fig. 9. The principle of micropipeline control logic is to use the output from the current stage to enable or disable the input of previous stage. The "delay" components illustrated in Fig. 9 are realized by logic gates of generating or receiving four-phase handshake signals for control tasks in the asynchronous blocks in the CDMA NoC. An example with more details about applying micropipeline control logics to asynchronous designs can be found in [18].

In order to suit the conventional synchronous design tools and other synchronous designs in the CDMA NoC, all the asynchronous blocks of the CDMA NoC are realized in RTL using VHDL together with the synchronous blocks. The basic principle is to model the basic components, C-element, latches, and combinational logic gates, in RTL using VHDL, and then build the asynchronous circuits using these RTL component models in a hierarchical way.

TABLE I
Area Cost of CDMA NoC Components

Data Path Width / *Block Name*		**Area (K equivalent gates)**			
		1-bit	**8-bit**	**16-bit**	**32-bit**
Node IF	Target	1.600			
	Initiator	3.882			
Tx/Rx Packet Buffer		6.101			
Packet Sender		1.505	1.509	1.513	0.962
Packet Receiver		1.977 ***(100.0%)***	7.331 ***(370.9%)***	13.718 ***(694.1%)***	26.488 ***(1340%)***
CDMA Transmitter		0.879 ***(100.0%)***	3.970 ***(451.8%)***	7.730 ***(879.6%)***	15.161 ***(1725%)***
Network Arbiter		0.936			

B. Data Path Configuration

Figs. 2 and 4 illustrate the principle of data encoding and decoding schemes used in the CDMA NoC by an example of processing and delivering one data chip of encoded data from the sender to the receiver at one time. Since one original data bit will be spread into S bits after encoding, the degree of data transfer parallelism between the "CDMA Transmitter" and "Packet Sender/Receiver" blocks affects the data transfer latency in the CDMA NoC largely. Namely, increasing the number of data bit encoded and delivered via "CDMA Transmitter" at one time can reduce the data transfer latency in the CDMA NoC and vice versa. However, increasing the data processing and delivering parallelism will incur larger area cost. Hence, in order to figure the tradeoff character between the parallelism and the area cost, the "Packet Sender," "CDMA Transmitter," and "Packet Receiver" blocks have been realized with four different data path configurations. According to the number of data bit transferred from a "Packet Sender" to a "Packet Receiver" through "CDMA Transmitter," the configurations are named as 1-, 8-, 16-, and 32-bit schemes.

C. Synthesis Results

The components of the CDMA NoC are synthesized using a 0.18-μm standard cell library. The Basic VCI (BVCI) interface standard [14] is applied in the realization of "Node IF" block. The data width and buffer depth in the "Tx/Rx Packet Buffer" blocks are set to 32 bits and 4 packets, respectively. In order to facilitate the simulation work later on, six network node and 8-bit Walsh codes are applied for synthesizing the "CDMA Transmitter," "Network Arbiter," and "Packet Sender/Receiver" blocks. The area cost of the components of the CDMA NoC under different data path configurations are listed in Table I. The area cost figures in Table I are presented as the number of equivalent gates. 85 K gates/mm^2 is used to calculate the number of equivalent gates for the 0.18-μm standard cell library.

From Table I we can see that when the data path width is increased from 1 to 32 bits, the area cost of "Packet Receiver" and "CDMA Transmitter" becomes 13 and 17 times larger. The area increase is due to the duplications of the encoding and decoding logic in the "CDMA Transmitter" and "Packet Receiver" blocks for increasing the data path width. By comparing the ratio of increased data path width, the increased area cost of the components is reasonable. To be noticed in Table I is that the area cost of the 32-bit version of "Packet Sender" block is smaller

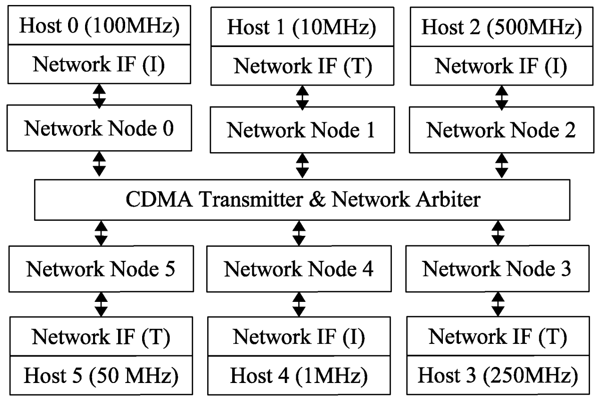


Fig. 10. Six-node CDMA NoC simulation network.

Destination Node ID	Source Node ID (optional for CDMA NoC)	other fields (optional)
Data Cell 1		
Data Cell 2 (optional)		
Data Cell 3 (optional)		

Fig. 11. Data packet format specification.

than others. The reason is that the data width of the output of "Tx Packet Buffer" block is 32 bits, thus the "Packet Sender" block need some control logic to adjust the fetched packet cells to be sent out according to the data path width if it is smaller than 32 bits. However, when the data path width is increased to 32 bits, the output data width adjusting logic is not needed in the "Packet Sender" block. The initiator type "Node IF" has larger area than the target type because it needs a buffer to store the header cell of received packets for supporting split-transaction feature in the BVCI standard.

V. Comparing With a PTP NoC

In order to examine the characteristics and performance of the CDMA NoC thoroughly, a simulation network that applies the CDMA NoC scheme is built and compared with a PTP NoC presented in [19].

A. *Simulation Network Setup*

The simulation network that applies the CDMA NoC is illustrated in Fig. 10. It contains six network nodes which work in different clock frequencies as illustrated in Fig. 10. The BVCI interface standard is applied in the network. Three hosts act as initiators and the other three act as targets, as denoted by the labels "I" and "T," respectively, in the "Network IF" blocks. The initiator hosts can generate requests to any target hosts, while the target hosts can generate responses only for the received requests passively. The network nodes are connected to each other through "CDMA Transmitter" and "Network Arbiter" blocks. The spreading codes used in the network are six 8-bit Walsh codes. The basic data unit transferred in the network is data packets composed by one header cell and several data cells as illustrated in Fig. 11. The number of data cells in a packet varies from one to three, while the width of each packet cell is fixed at 32 bits. The "functional host" blocks and their "Network IF"

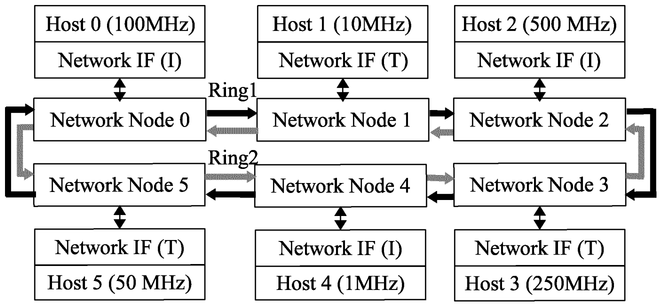


Fig. 12. Six-node PTP NoC simulation network.

blocks are not realized with any real IP blocks; they are simulated by adding stimulus signals on each "Network Node" block according to the BVCI standard. A four-phase dual-rail handshake protocol is applied in the CDMA network to transfer data between network nodes. The PTP network illustrated in Fig. 12 has the same mentioned network configurations as the CDMA network except that the network nodes in the PTP network are connected with each other through bidirectional ring topology. Therefore, the characteristics of the CDMA NoC can be examined more clearly by comparing the two networks in different aspects in the following four subsections.

B. *Comparison of Data Transfer Principles*

In the PTP connected network illustrated in Fig. 12, the data traffic load is distributed into the links among the network nodes. This distributed traffic scheme has the benefits of flexibility and scalability, whereas the main disadvantage is that the data transfer latency between two network nodes can be largely different when data are transferred to different destinations or to the same destination via different routes.

Although data transfers in the PTP network can be parallel if they take place in different links among the network nodes, concurrent data transfers over a single link is impossible in the PTP NoC because a link between two network nodes is shared in a time-division manner. Therefore, by applying CDMA technique, the main advantage of the CDMA NoC is the feature of concurrent data transfers. Hence, the data transfer latency in the CDMA NoC is a constant value which in turn helps the CDMA NoC to provide a guaranteed service for the on-chip system.

Another advantage of the CDMA NoC is that it can easily support multicast data transfers by requesting multiple receiver nodes to use the same spreading code for receiving. In the PTP NoC, the multicast transfer can be realized only by sending multiple copies of a data packet to its multiple destinations, unless extra logic is added in each network node to copy the multicast packet to both the functional host and the output link to the next node. This would increase the traffic load in the PTP network, or complicate the network implementation. One more benefit of applying the CDMA NoC is that the header cell in a packet needs not to be transferred in the network after a sending node gets the grant signal from the "Network Arbiter" since the receiving node already knew the sender information through the A-T protocol presented in Section II-C. However, in the PTP NoC, the header cell in a packet needs to be transferred in the network for packet routing.

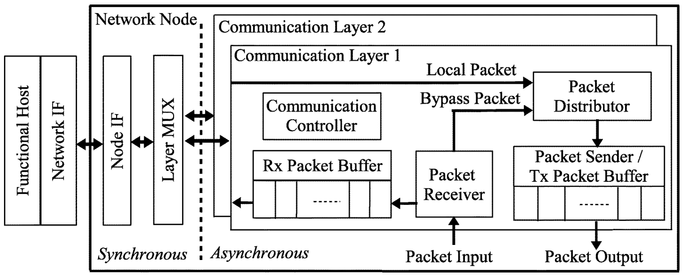


Fig. 13. Network node structure of the PTP NoC.

TABLE II
DATA TRANSACTION SPECIFICATION

Initiator Node	Target Node	Number of Transactions	Packet Length	
			Request Packet	Response Packet
Node 0	Node 1	2	4, 3	2, 3
	Node 3	2	3, 4	3, 2
	Node 5	1	4	2
Node 2	Node 1	1	4	2
	Node 3	1	3	3
	Node 5	2	4, 3	2, 3
Node 4	Node 1	2	3, 4	3, 2
	Node 3	1	3	3
	Node 5	1	4	2

C. Comparison of Network Node Structures

The network node structure of the PTP NoC presented in [19] is illustrated in Fig. 13. It contains two same "Communication Layer" blocks for supporting the bidirectional ring topology. By comparing with the network node illustrated in Fig. 6, the network node of the CDMA NoC has less complexity. The main reason is that the network node of the CDMA NoC does not need to handle any bypass packets or the packet routing issues because of its one-hop data transfer scheme. Therefore, the "Communication Controller" and "Packet Distributor" blocks illustrated in Fig. 13 are not needed in the node of the CDMA NoC. Since the CDMA NoC applies centralized traffic scheme, its network node does not need multiple "Communication Layer" blocks and "Layer MUX" block in the node of the PTP NoC illustrated in Fig. 13. When the data transfer parallelism needs to be increased in the PTP NoC, more "Communication Layer" blocks in a network node are needed in order to set up more links with other nodes, whereas the network node structure in the CDMA NoC does not need to change in this situation because of the parallel data transfer scheme.

D. Comparison of Data Transfer Latencies

The CDMA network illustrated in Fig. 10 and the PTP NoC illustrated in Fig. 12 are both synthesized using the same 0.18-μm technology library. Gate-level simulations are performed on both simulation networks. The data transactions performed during the simulations are listed in Table II. Each data transaction consists of one request packet from an initiator host to a target host and one corresponding response packet from the target host to the initiator host.

Because the GALS scheme is applied both in the CDMA network and the PTP network, the data transfer latency in the two simulation networks can be separated into two parts, synchronous transfer latency (STL) and asynchronous transfer latency (ATL). The STL refers to the data transfer latency between a functional host and the network node attached to it. STL depends on the local clock and the type of interface. The measured STL values of the CDMA network are listed in Table III. The constant values in Table III are caused by the handshakes in the asynchronous domain. They are independent of the local clock rate but belong to the synchronous transfer processes. Therefore, they are counted as a part of STL. From Table III, we can see that an initiator type of network node takes more clock cycles for local data transfers. The reason is that the initiator node needs to store or read the header cell to or from a buffer as mentioned in Section IV-C. Since the same "Node IF" block design is applied in both simulation networks, the STL of the PTP network has the same value as listed in Table III.

TABLE III
SYNCHRONOUS TRANSFER LATENCY

Node Type	Latency of sending data to 'Network Node'	Latency of receiving data from 'Network Node'
BVCI Initiator	8 local clock cycles + 2.5 ns	8 local clock cycles + 3.2 ns
BVCI Target	4 local clock cycles + 2.5 ns	4 local clock cycles + 3.1 ns

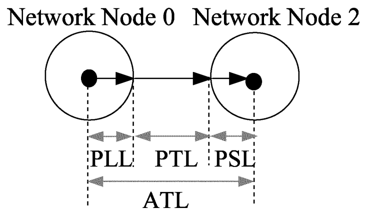


Fig. 14. ATL portions of the CDMA NoC.

The ATL refers to the data transfer latency of transferring data packets from one network node to the other node through a NoC structure using asynchronous handshake protocols. The ATL values in the PTP and CDMA networks consist of different portions which will be discussed separately in the following subsections.

1) ATL in the CDMA NoC: The ATL of the CDMA network consists of three portions: packet loading latency (PLL), packet transfer latency (PTL), and packet storing latency (PSL). The concept of those ATL portions is illustrated in Fig. 14 with an example where "Network Node 0" sends one data packet to "Network Node 2." The black arrows in Fig. 14 represent the packet transfer direction. The different portions of ATL are marked by grey arrows in Fig. 14 and explained in the following three paragraphs.

a) *PLL*: This is the time used by the "Packet Sender" block to fetch a data packet from "Tx Packet Buffer" and prepare to send the packet to "CDMA Transmitter."

b) *PTL*: This latency refers to the time used to transfer one data packet from the "Packet Sender" of the sender node to the "Packet Receiver" of the receiver node through the "CDMA Transmitter" and "Network Arbiter" blocks using a handshake protocol.

c) *PSL*: After the receiver node receives a data packet, it will spend a certain amount of time to store the received

TABLE IV
ATL PORTION VALUES OF THE CDMA NOC

		1 data cell	2 data cells	3 data cells
PLL (ns)		5.7	5.7	5.7
PTL (ns)	***1-bit***	384.6	768.9	1153.7
	8-bit	45.9	88.4	130.9
	16-bit	26.2	49.0	71.8
	32-bit	14.7	26.0	37.8
PSL (ns)		5.5	5.5	5.5

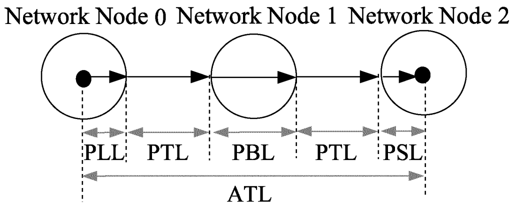


Fig. 15. ATL portions of the PTP NoC.

TABLE V
ATL PORTION VALUES OF THE PTP NOC

Packet Length	PLL (ns)	PTL (ns)	PBL (ns)	PSL (ns)
1 data cell	11.7	13.4	10.7	3.3
2 data cells	15.2	18.7	14.2	3.3
3 data cells	18.6	24.0	17.6	3.3

data packet into “Rx Packet Buffer.” This time duration is measured as PSL.

The measured values of ATL portions of the CDMA NoC under different data path configurations are listed in Table IV. The ATL value of the CDMA NoC can be calculated by directly adding the three portions under the same configuration.

2) ATL in the PTP NoC: The concept of the ATL portions of the PTP NoC is illustrated in Fig. 15 with an example that “Network Node 0” sends one packet to “Network Node 2” via “Network Node 1.” The black and grey arrows in Fig. 15 represent the same meanings as the arrows in Fig. 14. The meaning of ATL portions will be explained briefly in the following four paragraphs.

a) *PLL*: It is the time used to load one “local packet” into “Tx packet buffer” in the “Packet Sender” block as illustrated in Fig. 13.

b) *PTL*: This latency refers to the time used to transfer one data packet from the “Packet Sender”

$$\mathrm{ATL} = \mathrm{PLL} + \mathrm{PTL} \times (N+1) + \mathrm{PBL} \times N + \mathrm{PSL} \quad (1)$$

of a network node to the “Packet Receiver” of an adjacent node using a handshake protocol.

c) *PBL*: After a network node receives a packet from another node, it will check its destination address. If it is a “bypass packet,” it will be delivered into “Tx Packet Buffer.” The time spent on this process is called PBL.

d) *PSL*: It is the time spent on storing one “incoming packet” into “Rx Packet Buffer” block.

The ATL portion values of the PTP NoC are listed in Table V. The formula of calculating the ATL of transferring one packet in

TABLE VI
EQUIVALENT NUMBER OF INTERMEDIATE NODES IN THE PTP NOC

	1-bit	8-bit	16-bit	32-bit
1 data cell	N=15.2	N=1.2	N=0.4	N= - 0.1
2 data cells	N= 22.6	N=1.9	N=0.7	N=0.0
3 data cells	N= 26.9	N=2.3	N=0.9	N=0.1

the PTP NoC is given in (1). N refers to the number of intermediate nodes between the source node and destination node of a packet. If a packet is transferred between two adjacent network nodes, then N is 0.

3) Comparing the ATL Values: In Tables IV and V, we can see that PTL values of the CDMA NoC and the PTP NoC increases as the packet length increases. This is because the data cells in a packet are sent in a serial manner in the two networks. Thus, more data cells need more transmission time. Whereas, the PLL and PSL values of the CDMA NoC and the PTP NoC are nearly not affected by the packet length. The reason is that the data cells in a packet are loaded or stored in a parallel manner in both networks.

The main difference between the ATL values of the two NoCs is that the ATL value of the CDMA NoC is a constant value for a certain data packet length, whereas the ATL value in the PTP NoC is a variable depending on the packet traffic route. The ATL portion PBL of the PTP NoC does not exist in the ATL of the CDMA NoC because the data packets in the CDMA NoC are transferred directly from their source nodes to their destination nodes. The stable ATL value is an advantage of the CDMA NoC since it is very helpful for providing guaranteed service in the network.

The PTL values listed in Table IV show that the data path width configuration affects the ATL of the CDMA NoC in a linear manner. For instance, the PTL value of transferring a three-data-cell packet is reduced around 30 times when the data path width is increased from 1 to 32 bits. Since the data path width in the PTP network illustrated in Fig. 12 is realized as 32 bits, only the ATL value of the CDMA NoC with 32-bit data path width is comparable with the ATL value of the PTP NoC. However, in order to compare the data transfer latency characteristics of the two NoCs thoroughly, Table VI lists the equivalent number of intermediate network nodes which would be gone through by a data packet in the PTP NoC when the same size packet is transferred in the CDMA NoC under different data path configurations. From Table VI, we can see that when the data path widths in the CDMA NoC and the PTP NoC are both 32 bits, the ATL of delivering a two-data-cell packet in the CDMA NoC is equivalent to transferring the same packet between two adjacent network nodes in the PTP NoC, which means that the ATL of the CDMA NoC equals to the best case ATL value in the PTP NoC. When transferring a one-data-cell packet, the ATL in the CDMA NoC is even smaller than the best case ATL in the PTP NoC as denoted by the negative value of N in Table VI. The latency caused by the data encoding and decoding scheme in the CDMA NoC is compensated by its one-hop data transfer scheme. Hence, the CDMA NoC can transfer data packets with the equivalent best case ATL of the PTP NoC when the data path width is set to 32 bits.

TABLE VII
AREA AND POWER COSTS OF THE TWO NETWORKS

NoC Type	**Data Path Width**	**Six-Node Simulation Network**		
		Area (K equivalent gates)	Dynamic Power (mw)	Energy Cost of Delivering 32 bits (pJ)
CDMA NoC	*1-bit*	113.145	19.340	12.5168
	8-bit	148.369	6.563	3.7428
	16-bit	191.037	7.331	4.0868
	32-bit	272.806	7.332	4.0873
PTP NoC	*32-bit*	177.007	7.324	4.7401

E. Comparison of Area and Power Costs

The two simulation networks illustrated in Figs. 10 and 12 are synthesized using a 0.18-μm technology library. The area costs of the two simulation networks with different data path widths are listed in Table VII for comparison purpose. According to the data transactions performed in the gate level simulations and listed in Table II, the dynamic power costs during simulations and the energy costs of transferring 32 data bits in the CDMA NoC and the PTP NoC are also listed in Table VII.

From the figures in Table VII, we can see that when the data path width is increased from 1 to 32 bits in the CDMA NoC, the area cost of the CDMA network becomes 2.4 times larger because more logic are used to perform parallel data encoding and decoding. With 16- and 32-bit data path widths, the CDMA NoC loses its area cost advantage by comparing with the PTP NoC.

In terms of the dynamic power costs listed in Table VII, a 1-bit CDMA NoC should not be applied due to the much larger power consumption by comparing with the CDMA NoCs under other data path width configurations and the PTP NoC. The reason of the large power cost of the 1-bit CDMA NoC is that the 1-bit CDMA NoC needs much more switching activities than the other versions of the CDMA NoC due to the over-serialized data transfer scheme. To be noticed in Table VII is that the 16- and 32-bit CDMA NoCs have almost the same power consumptions. The reason is that the power consumption increase caused by the data path width increasing is compensated by reducing the control logic for data output adjust operations in each "Packet Sender" in the 32-bit CDMA NoC as explained in Section IV-C. By comparing the dynamic power costs and the energy costs of transferring 32 bits in Table VII, we can see that the PTP NoC has similar dynamic power cost with the 16- and 32-bit CDMA NoCs, while the energy figures are slightly larger than the figures in the two CDMA NoCs. This is because the PTP NoC takes more time to perform the data transactions listed in Table II due to its multiple hop data routing scheme. However, the CDMA NoC can perform the same data transactions with shorter time since its one-hop concurrent data transfer scheme. Therefore, the average energy spent on transferring 32 data bits in the CDMA NoC, except the 1-bit CDMA NoC, is smaller than the energy cost in the PTP NoC.

VI. CONCLUSION

An on-chip packet switched communication network that applies the CDMA technique and supports the GALS communication scheme was presented. The presented CDMA NoC uses an asynchronous scheme to perform the global data transfers between network nodes, and uses synchronous scheme to deal with the local data transfers between a functional host and the network node attached to it. A CDMA encoding and decoding scheme which suits digital-circuit implementation was presented. The main advantage of the presented CDMA NoC is that it can perform data transfer concurrently by applying CDMA technique in the network. Therefore, the large data transfer latency variance caused by the packet routing in a PTP NoC is eliminated in the CDMA NoC. The constant data transfer latency in the CDMA NoC is helpful for providing guaranteed communication services to an on-chip system. Another advantage of the CDMA NoC is that it can perform multicast data transfers easily by utilizing the multiple access feature of CDMA technique.

Both the asynchronous and synchronous circuits of the CDMA NoC with different data path widths are realized in RTL using VHDL in order to suit the conventional synchronous design flow and tools. Two six-node on-chip networks were constructed to compare the CDMA NoC with a PTP NoC. One network applies the CDMA NoC, while the other applies a bidirectional ring PTP NoC. The two networks were simulated and compared against each other. The simulation results reveal that when the data path width of the two simulation networks is set to 32 bits, the asynchronous transfer latency in the CDMA NoC is equivalent to the best case data transfer latency in the PTP NoC. The best case data transfer in the PTP NoC means that packets are transferred between two adjacent nodes. It indicates that the data transfers between any network nodes in the CDMA NoC can be performed as quickly as transferring the same data packets between two adjacent nodes in the PTP NoC.

By considering the tradeoff between transfer latency performance listed in Table VI and the costs listed in Table VII, a 16-bit CDMA NoC is a good option for replacing the PTP NoC in an on-chip system where universal data transfer latency is a desired requirement. With a 16-bit data path width, the data transfer latency of the CDMA NoC is close to the best case transfer latency in the PTP NoC while the area and dynamic power costs remain similar. If the area and power costs have higher priority, the 8-bit CDMA NoC can be applied because its area is 16.2% smaller than the PTP NoC while its energy cost of transferring 32 bits is 21.0% smaller than the cost in the PTP NoC.

REFERENCES

[1] D. Wiklund and D. Liu, "SoCBUS: Switched network on chip for hard real time systems," in *Proc. Int. Parallel Distrib. Process. Symp. (IPDPS)*, 2003, p. 8.

[2] K. Goossens, J. Dielissen, and A. Radulescu, "Æthereal network on chip: Concepts, architectures, and implementations," *IEEE Des. Test Comput.*, vol. 22, no. 5, pp. 414–421, Sep./Oct. 2005.

[3] D. Sigüenza-Tortosa, T. Ahonen, and J. Nurmi, "Issues in the development of a practical NoC: The proteo concept," *Integr., VLSI J.*, vol. 38, no. 1, pp. 95–105, 2004.

[4] A. J. Viterbi, *CDMA: Principles of Spread Spectrum Communications*. Reading, MA: Addison-Wesley, 1995.

[5] R. Yoshimura, T. B. Keat, T. Ogawa, S. Hatanaka, T. Matsuoka, and K. Taniguchi, "DS-CDMA wired bus with simple interconnection topology for parallel processing system LSIs," in *Dig. Tech. Papers IEEE Int. Solid-State Circuits Conf.*, 2000, pp. 370–371.

[6] T. B. Keat, R. Yoshimura, T. Matsuoka, and K. Taniguchi, "A novel dynamically programmable arithmetic array using code division multiple access bus," in *Proc. 8th IEEE Int. Conf. Electron., Circuits Syst.*, 2001, pp. 913–916.

[7] S. Shimizu, T. Matsuoka, and K. Taniguchi, "Parallel bus systems using code-division multiple access technique," in *Proc. Int. Symp. Circuits Syst.*, 2003, pp. 240–243.
[8] M. Takahashi, T. B. Keat, H. Iwamura, T. Matsuoka, and K. Taniguchi, "A study of robustness and coupling-noise immunity on simultaneous data transfer CDMA bus interface," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2002, pp. 611–614.
[9] R. H. Bell, Jr., K. Y. Chang, L. John, and E. E. Swartzlander, Jr., "CDMA as a multiprocessor interconnect strategy," in *Conf. Record 35th Asilomar Conf. Signals, Syst. Comput.*, 2001, pp. 1246–1250.
[10] E. H. Dinan and B. Jabbari, "Spreading codes for direct sequence CDMA and wideband CDMA cellular networks," *IEEE Commun. Mag.*, vol. 36, no. 9, pp. 48–54, Sep. 1998.
[11] E. S. Sousa and J. A. Silvester, "Spreading code protocols for distributed spread-spectrum packet radio networks," *IEEE Trans. Commun.*, vol. 36, no. 3, pp. 272–281, Mar. 1988.
[12] D. D. Lin and T. J. Lim, "Subspace-based active user identification for a collision-free slotted ad hoc network," *IEEE Trans. Commun.*, vol. 52, no. 4, pp. 612–621, Apr. 2004.
[13] D. M. Chapiro, "Globally-asynchronous locally-synchronous systems," Ph.D. dissertation, Dept. Comput. Sci., Stanford University, Stanford, CA, 1984.
[14] VSI Alliance, Wakefield, MA, "Virtual component interface standard version 2," 2001. [Online]. Available: http://www.vsi.org/
[15] OCP-International Partnership. Beaverton, OR, "Open core protocol specification," 2001. [Online]. Available: http://www.ocpip.org/
[16] X. Wang and J. Nurmi, "A RTL asynchronous FIFO design using modified micropipeline," in *Proc. 10th Biennial Baltic Electron. Conf. (BEC)*, 2006, pp. 1–4.
[17] I. E. Sutherland, "Micropipelines," *Commun. ACM*, vol. 32, no. 6, pp. 720–738, 1989.
[18] X. Wang, T. Ahonen, and J. Nurmi, "Prototyping a globally asynchronous locally synchronous network-on-chip on a conventional FPGA device using synchronous design tools," in *Proc. Int. Conf. Field Program. Logic Appl.*, 2006, pp. 1–6.
[19] X. Wang, D. Sigüenza-Tortosa, T. Ahonen, and J. Nurmi, "Asynchronous network node design for network-on-chip," in *Proc. Int. Symp. Signal, Circuits, Syst.*, 2005, pp. 55–58.

Xin Wang received the M.Sc. degree in electronic circuits and systems from Northwestern Polytechnical University, Xi'an, China in 2002.

Currently, he is a Researcher with the Institute of Digital and Computer Systems, Tampere University of Technology, Tampere, Finland. His research interests are focused on on-chip communication networks and asynchronous circuits design.

Tapani Ahonen received the M.Sc. degree in electrical engineering and the Ph.D. degree in information technology from Tampere University of Technology, Tampere, Finland.

He is a Senior Research Scientist with Tampere University of Technology. His research interests are focused on varying aspects of system-on-chip design.

Jari Nurmi is received the Ph.D. degree from Tampere University of Technology (TUT), Tampere, Finland, in 1994.

He is a Professor of digital and computer systems with TUT. He has held various research, education, and management positions at TUT and in the industry since 1987. His current research interests include system-on-chip integration, on-chip communication, embedded and application-specific processor architectures, and circuit implementations of digital communication, positioning, and DSP systems. He is leading a group of about 25 researchers at TUT. He is the author or coauthor of about 160 international papers, the editor of *Processor Design: System-on-Chip Computing for ASICs and FPGAs* (Springer, 2007), coeditor of *Interconnect-Centric Design For Advanced SoC and NoC* (Kluwer, 2004), and has supervised more than 90 M.Sc., Licentiate, and Doctoral theses.

Dr. Nurmi is currently the general chairman of the annual International Symposium on System-on-Chip (SoC) and of its predecessor SoC Seminar in Tampere since 1999 and a board member of SoC, FPL, and NORCHIP conference series. He was the head of the national TELESOC graduate school 2001–2005. He is a senior member in the IEEE Signal Processing Society, the Circuits and Systems Society, the Computer Society, the Solid-State Circuits Society, and the Communications Society. In 2004, he was a corecipient of the Nokia Educational Award, the recipient of the Tampere Congress Award in 2005, and the Academy of Finland Senior Scientist research grant for the academic year 2007–2008.

PUBLICATION 9

X. Wang, and J. Nurmi, “Modeling A Code-Division Multiple-Access Network-on-Chip Using SystemC”, in Proceedings of 25th Norchip Conference, (NORCHIP 2007), Aalborg, Denmark, November 2007.

Modeling A Code-Division Multiple-Access Network-on-Chip Using SystemC

Xin Wang, and Jari Nurmi
Tampere University of Technology, 33101 Tampere, Finland
E-mail: *{xin.wang, jari.nurmi}@tut.fi*

Abstract- **A SystemC model of A Code-Division Multiple-Access (CDMA) Network-on-Chip (NoC) is presented in this paper. The CDMA NoC modeled in this paper is a Globally-Asynchronous Locally-Synchronous (GALS) on-chip communication network which applies CDMA technique to transfer data among different network nodes concurrently. The presented SystemC model uses Transaction-Level Modeling (TLM) approach to model the asynchronous handshake processes for data transfers in the CDMA NoC. A performance-estimation method which bases on timing back-annotation is also presented for exploring the CDMA NoC performance under different configurations in a fast and efficient way. Finally, the performance estimation results of the CDMA NoC with different configurations and traffic patterns are presented.**

I. INTRODUCTION

As the number of components integrated into an on-chip system is increasing, the communications among the large number of components become more and more complicated. Network-on-Chip has been proposed as a promising solution for the complex on-chip communication issue. A NoC scheme which applies CDMA technique has been presented in [1]. As a different approach from the point-to-point (PTP) connection NoC such as SPIN [2] and Æthereal [3], the CDMA NoC uses a set of orthogonal pseudo-noise codes to separate the data streams from different network nodes in code domain, therefore, the different data streams can be transferred concurrently in time domain. Hence, the CDMA NoC can supply an invariable data transfer latency independent on the data transfer routes and network topology.

Before applying a NoC scheme into an on-chip system, the designer needs to estimate the NoC performance under different system configurations in an early design stage. Therefore, a high level system model which can run much faster and be more flexible than Register-Transfer Level (RTL) model is needed. SystemC [4], a C++ class library, has been developed to meet this requirement for system modeling. Since a SystemC model is totally described by a software programming language, the abstraction level of the system model can be very flexible and the simulation can run at a faster speed than a RTL model. Several works about modelling on-chip communication architecture in transaction level using SystemC have been published. [5] and [6] concern more about system design methodologies using SystemC. The work in [7] presented a transaction-level interface model which translates interface functions into signal level for verifying blocks designed by a Hardware-Description Language (HDL). The issue of modeling a GALS NoC structure using SystemC is presented in [8]. It uses sc_fifo primitive to model the asynchronous logics of the NoC, therefore, it encounters the problem of memory effect of sc_fifo primitive and has difficulty to model an active-input/passive-output channel [8]. The SystemC modeling work presented in this paper avoids the problems met in [8] by abstracting each asynchronous block of the CDMA NoC as a channel and using interface functions to model asynchronous communication processes. By using the proposed SystemC model, a performance-estimation method for evaluating the performance of the CDMA NoC under different configurations is also presented in this paper.

The following sections of this paper are arranged as follows. Section 2 will briefly introduce the CDMA NoC structure. Section 3 presents the SystemC model of the CDMA NoC using TLM approach. The performance-estimation of the CDMA NoC under different configurations will be presented in Section 4. Finally, the conclusions are drawn in Section 5.

II. THE GALS CDMA NOC STRUCTURE

The on-chip CDMA network used in this modeling work has been presented in [1]. It applies both GALS scheme [9] and CDMA technique [10] to deliver the data packets for an on-chip system. By applying the GALS scheme, it solves the problem of data transfers among different clock domains. By applying CDMA technique, the different data streams from different functional components in the system are separated in code domain to achieve concurrent data transfers in time domain. By comparing with a PTP connection on-chip network, such as a GALS bidirectional ring NoC [11], the CDMA NoC has the advantage of supplying a constant data transfer latency which is independent on data transfer routes and network topology. Since the details of the CDMA NoC have been presented in [1] already, the following part of this section will briefly go over the basic information of the NoC. In Fig.1, we can see the structure of the CDMA NoC in which each functional component ('functional host') is connected to the network through a 'Network Node' block. The network nodes transfer data to each others through a backbone structure which consists of 'Network Arbiter' and 'CDMA Transmitter' blocks. The dotted circles and the clock frequencies marked in Fig.1 are used to illustrate how the GALS scheme is applied in the CDMA NoC. The function of each block in the NoC will be briefly introduced by the following subsections.

A. Network Node

The block diagram of 'Network Node' is illustrated in Fig.2 where the arrows represent the flows of data packets. The 'Network IF' block in Fig.2, which belongs to the functional host, is an interface block between a functional host and a 'Network Node'. Only 'Node IF' block in a 'Network Node' applies synchronous design since the interface standard, VCI [12] or OCP [13], used in the interface is synchronous. The function of the other blocks in 'Network Node' will be briefly described in the following four paragraphs.

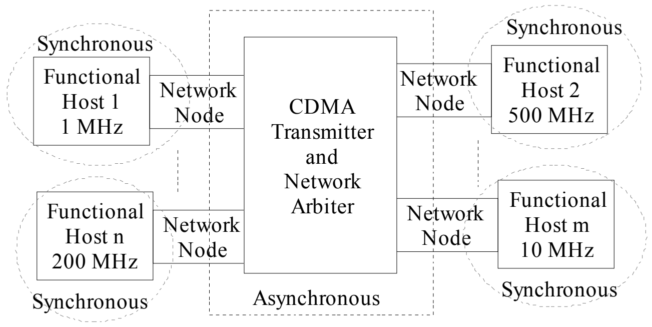


Fig. 1. The GALS CDMA NoC Structure

1) *'Node IF'*: This is an interface block which applies VCI or OCP standard to assemble the data from the functional host into packet format and send the packet to 'Tx Packet Buffer', or to disassemble the received packet from 'Rx Packet Buffer' and send the extracted data to the functional host.

2) *'Tx/Rx Packet Buffer'*: These two blocks are buffers used to store the data packets to be transferred or to store the received data packets from 'Packet Receiver'.

3) *'Packet Sender'*: If 'Tx Packet Buffer' is not empty, 'Packet Sender' will fetch a data packet from the buffer. Then, it will extract the destination information from the fetched packet and send the destination address to 'Network Arbiter'. After 'Packet Sender' gets the grant signal from the arbiter, it will start to send data packets to the destination node through 'CDMA Transmitter'.

4) *'Packet Receiver'*: This block waits the request from 'Network Arbiter' to load the proper spreading code for decoding. After it is ready, the receiver will start to receive and decode the data from 'CDMA Transmitter', and then send the decoded data to 'Rx Packet Buffer' in packet format.

B. CDMA Transmitter and Network Arbiter

Each 'Network Node' can start to send data to the 'CDMA Transmitter' block only after it received grant signal from 'Network Arbiter'. Then the 'CDMA Transmitter' will encode the data with the corresponding unique spreading code of the sender node and send the encoded data to destination nodes. 'First come, first served' mechanism is applied to ensure that the data encoding and sending are performed as soon as there is data transfer request. The data from different nodes will be encoded and delivered in the unit of channel width. For instance, if the channel width is 8 bits, 8-bit data sections from different network nodes will be handled by 'CDMA Transmitter' at a time. If a node requests to send data to 'CDMA Transmitter' before the completion of the current data encoding and transfer process, the sender node has to wait until the next round of data handling starts in 'CDMA Transmitter'. This situation is called CDMA transfer contention.

'Network Arbiter' takes charge of informing the destination nodes to prepare the proper decoding code and arbitrating the data transfer requests from different sender nodes if they are requesting the same destination node. In the case that there are more than one sender nodes requesting to send data to the same destination node simultaneously or at different times, the arbiter will apply 'round-robin' arbitration scheme or the 'first-come first-served' principle, respectively, to guarantee that there is only one sender sending data to one

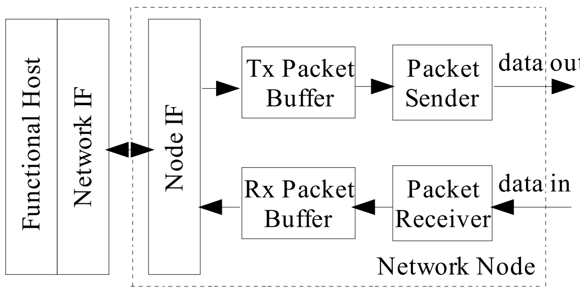


Fig. 2. Block Diagram of 'Network Node'

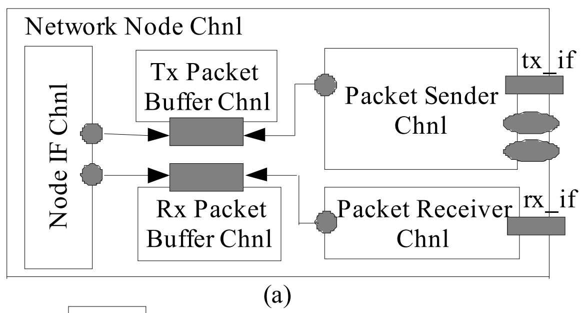
specific receiver at a time. However, if different nodes request to send data to different destination nodes, these requests would not interfere with each other and will be handled concurrently in 'Network Arbiter'.

III. SYSTEMC MODEL OF THE CDMA NOC

Transaction-Level Modeling (TLM) [14] is a modeling style which bases on the features about channels and interfaces of SystemC 2.1. In TLM, the communication transactions are modeled by calling the interface functions defined in the interfaces of channels. The interface functions are implemented in the channels. Therefore, by separating the definition from the implementation of the interface functions, the system model only needs to care about the transactions among modules and the data flow in the system without the details of the communication method. The processes of calling an interface function of a channel includes call and return steps, which is very similar to the request and acknowledge steps in an asynchronous handshake protocol. Thus, TLM method is used to model the asynchronous data transfers in the CDMA NoC by modeling the handshake processes as the interface functions of a channel. The details about the transaction-level SystemC model of the CDMA NoC will be presented in the following subsections.

A. The Channels and Interfaces in the SystemC Model of the CDMA NoC

The data packets are delivered from one block to the other in the CDMA NoC through an asynchronous handshake protocol. Therefore, the SystemC model of the CDMA NoC also follows the block hierarchy illustrated in Fig.1 and Fig.2 in order to keep the uniform hierarchy between different levels of abstractions. Each block in the CDMA NoC is modelled as a channel. The interface functions and the relationships among the channels are illustrated in Fig.3. Fig.3 (a) illustrates the channels and interface function calling relationships within a 'Network Node', while, Fig.3 (b) presents the relationships among 'Network Node', 'Network Arbiter', and 'CDMA Transmitter'. In Fig.3, each grey square at the boundaries of a channel represents an interface of that channel, and each grey circle at the boundaries of a channel represents an instantiated interface port of other channels. The arrows in Fig.3 point from the instantiated interface port to its original interfaces of channels. For example, the 'CDMA Transmitter' block communicates with a 'Network Node' block by instantiating an interface, called 'tx_if', of the 'Network Node', then calling the functions in the 'tx_if' interface to get data. The interface functions contained in



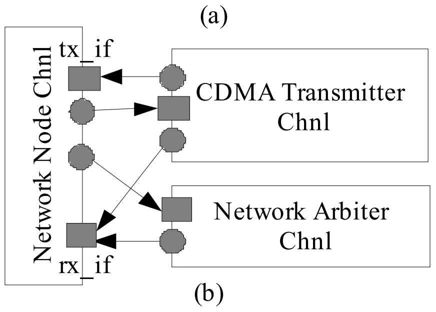


Fig. 3 Channels and Interfaces Diagram of the CDMA NoC SystemC Model

each interface in the NoC model are listed in Table I and described in the following seven paragraphs.

1) read()/write(): These two functions are used to read or write data packets from or to a 'Tx/Rx Packet Buffer'.

2) get_status(): This function is used to get the stored number of packets in a buffer.

3) cdma_tx_ack(): This function of 'CDMA Transmitter' channel is used to send acknowledge to the 'Packet Sender' channel as the end of current data transfer.

4) load_rx_pn_code(): The 'Network Arbiter' will call this function to let a receiver node load the right spreading code for decoding.

5) rx_from_noc():'CDMA Transmitter' uses this function to send data to a 'Network Node' channel.

6) cdma_tx_req(): This function is used by a 'Network Node' to send data to 'CDMA Transmitter' channel.

7) tx_arb_req(): Each 'Network Node' will call this function to request arbitration from the 'Network Arbiter'.

B. Modeling the Parallel Request/Arbitration Processes

The main challenge of modeling the CDMA NoC is to model the parallel data transfer and arbitration processes carried out in hardware blocks, 'CDMA Transmitter' or 'Network Arbiter' blocks as described in Section II-B, with SystemC software model. In hardware design, there are dedicated circuits for each 'Network Node' to handle the requests so that the handling processes are performed in parallel. However, in the software model, a design method of modelling the parallel handling processes needs to be applied since the simulation kernel of SystemC only can run one process at a time [4]. The parallel process modelling method applied in this work is illustrated in Fig.4 with an example from 'CDMA Transmitter' channel. In Fig.4, we can see that all 'Network Node' send their requests to 'CDMA Transmitter' via the generic interface function 'cdma_tx_ req()'. In order to let more requests get through the calling of 'cdma_tx_req()', a separate SC_THREAD process, cdma_tx_proc(), is used to handle the received requests after waiting for a certain amount of time. This method works

Table I. Interface Functions in Each Channel

Channels	Interface Functions
Tx/Rx Packet Buffer Chnl	read()
	write()
	get_status()
Packet Sender Chnl	cdma_tx_ack()
Packet Receiver Chnl	load_rx_pn_code()
	rx_from_noc()
CDMA Transmitter Chnl	cdma_tx_req()
Network Arbiter Chnl	tx_arb_req()

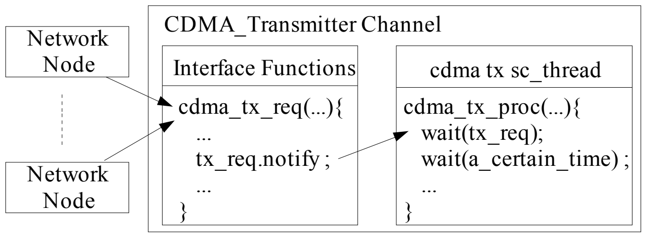


Fig. 4 Modeling Parallel Request/Arbitration Processes

since a SC_THREAD process will be suspended when it runs to a wait() function. Thus, the simulation kernel gets chance to run more request function call processes before starting the request handling process. In this way, multiple request handling processes are performed in parallel.

The same method is also applied in 'Network Arbiter' to realize the arbitration process by using a generic interface function for 'Network Node' to send requests. Simultaneously appeared requests for the same destination node will be arbitrated by the simulation kernel automatically since only one request process will be chosen to run at a time. The other failed requests will be kept until they are granted. The requests for different destination nodes will be handled one by one in the 'Network Arbiter' channel.

IV. PERFORMANCE ESTIMATION OF THE CDMA NOC

Based on the SystemC model of the CDMA NoC presented in Section III and the RTL realization presented in [15], a performance estimation method which combines the flexibility of SystemC model and the accuracy of RTL realization of the CDMA NoC is proposed as the following steps.

Step1: Model each block of the CDMA NoC as a channel and build the CDMA NoC model according to the block hierarchy.

Step2: Realize each block of the CDMA NoC in RTL and do the synthesis and gate-level simulation using the target technology library.

Step3: Record the latency information of the handshake processes among the blocks from the RTL simulation, and then back annotate the latency information to the corresponding channels in the SystemC model of the CDMA NoC.

Step4: Estimate the performance of the CDMA NoC under different configurations by simulating the timed SystemC model of the CDMA NoC.

A. Simulation Environment Setup

The setup of the simulation environment for the CDMA SystemC model is illustrated in Fig.5. In order to concentrate

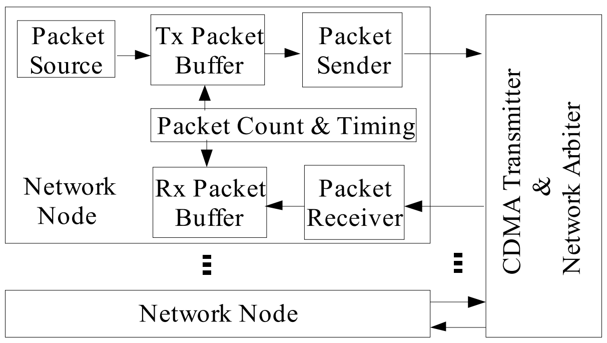


Fig. 5 Simulation Environment Setup

Table II. The Extracted Latency Values from RTL Realization

Communication Processes		Latency
Read/Write Tx Packet Buffer		10.9/11.5 ns
Store a packet into Rx Packet Buffer		5.5 ns
Tx 32 bits data from 'Packet Sender' to 'Packet Receiver' (with different channel widths)	1-bit	384.6 ns
	8-bit	45.9 ns
	16-bit	26.2 ns
	32-bit	14.7 ns
Arbitration retry after contention		2.4 ns

the simulation work on the asynchronous global network itself other than the local synchronous interface which depends on the type of applied interface standard in the CDMA NoC, the 'Network Node' block is revised as illustrated in Fig.5. The 'Node IF' is replaced by a 'Packet Source' block which generates data packets according to a specific traffic pattern. The size of 'Tx Packet Buffer' block is set to be large enough for storing all packets from the packet source during the simulation in order to make the simulation to be an open-loop simulation, which ensures that the traffic produced by the source is not influenced by the network. The 'Rx Packet Buffer' is used to store the received packets. The 'Packets Count and Timing' process is added on the 'Tx/Rx Packet Buffer' blocks for counting and recording the packet transfer information during simulations.

B. *Performance Estimation Results*

Three kinds of configuration parameters, CDMA channel widths, numbers of network node, and traffic patterns, are explored during the simulations. The latency information used in the simulations are extracted from the RTL realization presented in [15] and listed in Table II.

Firstly, the different CDMA channel widths, which refer to the number of data bits encoded and transferred at a time via 'CDMA Transmitter', are explored in the 6 nodes CDMA network. The traffic pattern used in the simulations is independent and uniform traffic which means that the same amount of packets are independently generated at each network node. The destinations of the generated packets in each node are uniformly distributed to all the other network nodes. Each node has sent 5000 packets to the network, and the average number of data cells in the packets is 2. Fig.6 gives the average Asynchronous Transfer Latency (ATL) of delivering a 32 bits data cell with different channel widths in the CDMA NoC. By comparing with the latency values

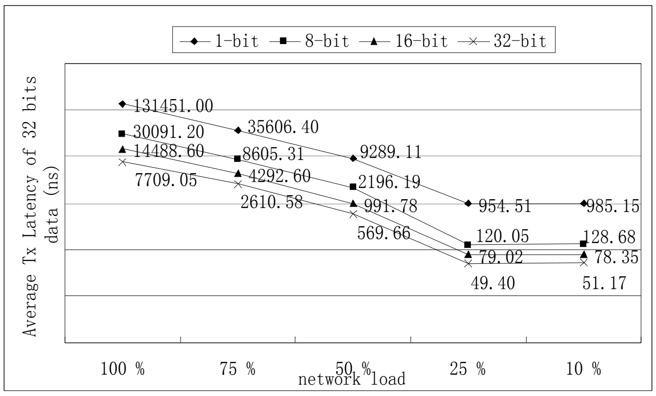


Fig. 6 Latencies with Different Channel Widths

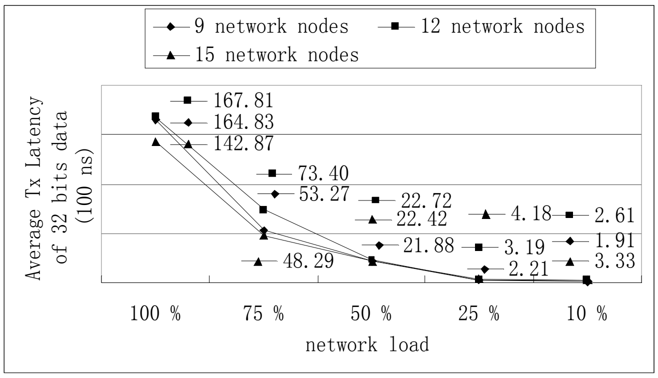


Fig. 7 Latencies with Different Numbers of Nodes

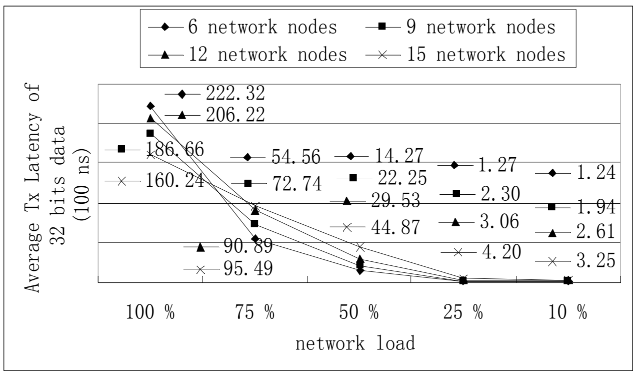


Fig. 8 Latencies of Hot-Spot Traffic

presented in [15], we can see that the packet arbitration contentions and CDMA transfer contentions increase the latency severely.

Secondly, the different numbers of network nodes are tested in the SystemC simulations since it can be done much easier than the RTL realization and simulation. The traffic pattern and other simulation configurations are same with the ones used in the simulations of different channel widths except that each network node sends 500 packets to each other nodes. The channel width used in the simulations is 8-bit. The average ATL values under different numbers of network nodes are illustrated in Fig.7. From the results, we can see that the transfer latency increases when the number of network nodes increases since the probability of contentions increases.

Thirdly, a hot-spot traffic pattern which is more likely appears in real applications is simulated in the CDMA networks with different numbers of network nodes. Node 1 is selected as the 'hot' node and the 'hot' degree is 0.25 which means that 25% of the generated packets in each node are transferred to Node 1. The other packets are still uniformly distributed to all other nodes besides Node 1. The average ATL values of transferring a 32 bits data cell with hot-spot traffic is illustrated in Fig.8. By comparing with the transfer latency values illustrated in Fig.7, the CDMA NoC has similar latency values under hot-spot traffic when the network load is smaller than 50%. For the heavier network loads, the transfer latencies become larger. It means that the CDMA NoC is not sensitive to the balance of network load when the network load is light.

V. Conclusions

A SystemC modelling work for a GALS CDMA NoC is presented in this paper. The SystemC model is built in transaction level by modelling each block of the CDMA NoC as a channel. The asynchronous handshake processes for data transfers in the NoC are modelled by the interface function calls between channels. Based on the presented SystemC model and the previously developed RTL realization of the CDMA NoC, a performance estimation method is presented. With the estimation method, the performances of the CDMA NoC under different configurations have been simulated. The different configurations which have been explored during simulations include channel width, number of network nodes, and traffic patterns. The simulation results give a fast estimation of the transfer latency of the CDMA NoC under different channel widths when contentions are included. According to the simulation results, when the number of nodes increased in the CDMA NoC, the transfer latency will increase linearly since the possibility of contention increases. Finally, a hot-spot traffic pattern is also been simulated, the results reveal that the CDMA NoC is not sensitive to the network load balance when the network load is lighter than 50%.

With the presented performance-estimation method, a system designer can evaluate the performance of the CDMA NoC under different configurations in a manner which combines the fastness of TLM SystemC model and the accuracy of RTL realization.

References

[1] X. Wang, J. Nurmi, "An On-Chip CDMA Communication Network", Proc. of 2005 International Symposium on System-on-Chip, Nov. 2005.

[2] P. Guerrier, A. Greiner, "A Generic Architecture for On-Chip Packet-Switched Interconnections," *Proceedings of the Design, Automation and Test in Europe Conference 2000*, Mar. 2000.

[3] K. Goossens, J. Dielissen, and A. Radulescu, "Æthereal network on chip: concepts, architectures, and implementations," *IEEE Design & Test of Computers*, Volume 22, Issue 5, Sep.-Oct. 2005.

[4] Open SystemC Initiative (OSCI), "IEEE Standard SystemC Language Reference Manual", http://www.systemc.org

[5] A. Wieferink, T. Kogel, R. Leupers, G. Ascheid, H. Meyr, G. Braun, A. Nohl, "A system level processor/communication co-exploration methodology for multi-processor system-on-chip platforms", Proc. of Design, Automation and Test in Europe Conference and Exhibition 2004, Vol. 2, Feb. 2004, pp. 1256 - 1261.

[6] W. Klingauf, R. Gunzel, "Rapid prototyping with SystemC and transaction level modeling", Proc. of IEEE International Conference on Field-Programmable Technology 2005, Dec. 2005, pp. 285 – 286.

[7] S. Xu, H. Pollitt-Smith, "A TLM platform for system-on-chip simulation and verification", Proc. of IEEE VLSI-TSA International Symposium on VLSI Design, Automation and Test 2005, Apr. 2005, pp. 220 – 221.

[8] E. Beigne, F. Clermidy, P. Vivet, A. Clouard, M. Renaudin, "An asynchronous NOC architecture providing low latency service and its multi-level design framework", Proc. of 11th IEEE International Symposium on Asynchronous Circuits and Systems, Mar. 2005, pp. 54 – 63

[9] D. M. Chapiro, "Globally-Asynchronous Locally-Synchronous Systems", PhD thesis, Stanford University, Oct. 1984.

[10] M. K. Simon, J. K. Omura, R. A. Scholtz, B. K. Levitt, "Spread Spectrum Communicalions", MD: Computer Science, Vol. 3, Rockville, 1984.

[11] X. Wang, and J. Nurmi, "Comparison of a Ring On-Chip Network and a Code-Division Multiple-Access On-Chip Network," *VLSI Design*, Apr. 2007.

[12] VSI Alliance, "Virtual Component Interface Standard v2", Apr. 2001.

[13] OCP-IP Association, "Open Core Protocol Specification", 2001.

[14] T. Grötker, S. Liao, G. Martin, S. Swan, "System Design with SystemC", Kluwer Academic Publishers, 2002, pp.131-153

[15] X. Wang, and J. Nurmi, "Comparing Two Non-Blocking Concurrent Data Switching Schemes for Network-on-Chip," Proc. of International Conference on 'Computer as a Tool' (Eurocon 2007), Sept. 2007.