AFRONOC: AN ADAPTIVE FLEXIBLE NETWORK ON CHIP ROUTER

by

Ömer Çoğal

B.S., Electronics Engineering, Istanbul Technical University, 2005

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Electrical and Electronics Engineering
Boğaziçi University

2009

ACKNOWLEDGEMENTS

I would thank to my family, Hüseyin, Cemile and İlknur, for their patience and great encouragement not only during this thesis work but also during my whole life.

I would thank to my supervisors Prof. Günhan Dündar and Assoc. Prof. Arda Yurdakul for their support and guidance during this thesis work and the lectures that I have taken during my MSc. education.

I also thank to Tevfik Nur, my colleague from TÜBİTAK, for his help about FPGA design.

This work is partially supported by The Scientific and Technological Research Council of Turkey, TÜBİTAK (Project Nr.: 104E038) and Boğaziçi University Scientific Research Projects (Project Nr.: 09A103P).

ABSTRACT

AFRONOC: AN ADAPTIVE FLEXIBLE NETWORK ON CHIP ROUTER

As the complexity of on-chip systems grows, scalability and re-configurability becomes an important issue in both system and interconnection levels for SoC systems. Flexible and configurable architectures bring the advantage of reusability of the same hardware in different regular topologies such as torus, mesh, tree and in custom irregular ones. Research in NoC design points the importance of scalability, configurability and flexibility of the routers and on chip interconnects. This thesis describes an adaptive and flexible router design for all Network on Chip topologies, which can be changed during runtime. In flexible NoCs, table updates are carried out by a central unit, which increases complexity and area of the overall system. In AFRONOC, a reduced form of the link state routing is introduced for table updates so that the overall system can set up/change the topology by itself. Hence, the proposed adaptation algorithm makes the router stand-alone, that means it can adapt to the rest of the network without help of any external or central monitoring. The proposed adaptation process initializes the routing tables in a short time when compared with the reconfiguration based methods. Design-time configurability is achieved in terms of the number of channels, the number of nodes in the network, buffer size of each channel and physical data width. As a result, the router can be considered as a solution in ad-hoc NoCs for fast prototyping, which is necessary for filling the design productivity gap in NoC design. Area occupation of an example implementation with four I/O channels, eight bit data width, four bit address width on a Virtex-II pro xcvp70 device is 750 slice, which is 2 per cent of the total area of the FPGA.

ÖZET

AFRONOC: UYARLANABİLİR VE ESNEK BİR KIRMIK ÜSTÜ AĞ YÖNLENDİRİCİSİ

Kırmık-üstü sistemlerin karmaşıklığı arttıkça ölçeklendirilebilirlik ve yeniden yapı landırılabilirlik konuları, sistem ve bağlantı düzeyinde önem kazanmaya başlamıştır. Esnek ve yapılandırılabilir mimariler simit (torus), çark (mesh) ve ağaç (tree) gibi düzenli topolojilerin yanısıra düzensiz ağ topolojilerinde de yeniden kullanılabilirlik açısından avantaj sağlamıştır. Kırmık-üstü Ağ konusunda yapılan birçok araştırma ,ölçeklendirilebilirlik, yapılandırılabilirlik ve esneklik konularının öneminini ortaya koymaktadır. Bu tez çalışmasında bir çok düzenli ve düzensiz ağ topolojisinde kullanılabilecek uyumlu ve esnek bir ağ yönlendiricisi tasarımı ele alınmıştır. Önerilen yönlendirici mimarisi, tablo-tabanlı bir alt yapıya dayanamaktadır, Yönlendirme ve uyum algoritması olarak bağlantı-durumu yönlendirmesi algoritmasının indirgenmiş ve uyarlanmış bir türevini kullanmaktadır. Çalışma zamanındaki topoloji değişimlerini destekleyecek bir altyapı ortaya koyulması hedeflenmiştir. Önerilen uyum algoritması yönlendiriciye kendi başına ağın geri kalanına uyum sağlayabilme özelliğini kazandırır. Böylece harici veya merkezi olarak ağın durumunu gözleyecek bir yapıya ihtiyaç duyulmamaktadır. Öte yandan bu uyum algoritmasıyla, yönlendiricilerin kısa sürede ilklendirilmesini sağlamaktadır. Ayrıca kanal sayısı, ağdaki düğüm sayısı, tampon büyüklüğü ve fiziksel veri yolu genişliği gibi parametrelerin değişken olarak seçilmesi sonucu tasarım zamanında yapılandırlabilirlik ve esneklik sağlayacak bir yapı elde edilmiştir. Xilinx Virtex-II pro xc2vp70 APKD üstünde gerçeklenen dört giriş çıkış kanallı, sekiz bit veri yoluna ve dört bit adres yoluna sahip örnek bir yönlendirici, bu AKPD'nın yüzde ikisi olan 750 dilimlik(slice) alanı kaplamaktadır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS/ABBREVIATIONS

$BUFDEPTH$	Input Output Buffer Depth of Router
$WADDR$	Width of Address Lines
$WDATA$	Width of Data Lines
WIC	Width of Input Channel Count
WOC	Width of Output Channel Count
ACMT	Address Channel Match Table
APKD	Alanda Programlanabilir Kapı Dizisi
AD	Address Decoder
CB	Cross Bar
CMHC	Current Minimum Hop Count
CMP	Chip Multi Processor
CLB	Configurable Logic Block
CU	Central Unit
FC	Flow Control
FIFO	First In First Out
FPGA	Field Programmable Gate Array
FSM	Finite State Machine
HDL	Hardware Description Language
ITRS	International Technology Roadmap for Semiconductors
LSI	Large Scale Integrated Circuit
MUX	Multiplexer
NoC	Network On Chip
RA	Routing Arbiter
RAM	Random Access Memory
RFC	Request For Comment

SAF	Store And Forward
SoC	System On Chip
TUB	Table Update Block
ULSI	Ultra Large Scale Integrated Circuit
VCC	Virtual Channel Controller
VCT	Virtual Cut Through
VHDL	Very High Speed Integrated Circuit Hardware Description Language
VLSI	Very Large Scale Integrated Circuit
WH	Worm Hole

1. INTRODUCTION

With the technological level shift in manufacturing processes of integrated circuits, the complexity of circuits on a single chip has grown up rapidly. For the past thirty years, the names of the circuits on a chip have been changed from LSI (Large Scale Integrated Circuits) to ULSI (Ultra Large Scale Integrated Circuits) and finally the word SoC, system-on-chip came as the hot topic in integrated circuits and systems world. According to the Moore's law declared in 1965, it is known that the number of transistors on a chip would double up in almost every two years and according to ITRS (International Technology RoadMap for Semiconductors) 2003 report [1],this grow will go on with a slight drop in period to almost every three years. So as the complexity of the systems on chip grows, the number of transistors and hence the computational speed of processing elements on a chip grows.

Mainly, the on chip systems have four different topics to handle as sub-blocks, which are computation, communication, memory and I/O modules [2]. As mentioned above, the computation power of the chips are growing exponentially, which yields to open the door for parallel processing applications and dictates the same speed up for other main blocks of a SoC. For the communication infrastructures, usually single bus based systems are used which began to be inadequate for parallel processing multi-core SoCs. The fundamental reasons for this are, the non-negligible propagation delays on long global bus wires, the switching power consumption on these relatively long wires, necessity of repeaters since the voltage levels are low and hence high probability of signal loss due to the relative increase in length of the wire. Hence, the research has been directed to find clustered, modular and parallel solutions.

So NoC concept is proposed as a new solution to the SoC interconnection infrastructures due to its following advantages over traditional bus based communication: It

provides smaller clustered wires, hence wire delays are shortened and switching power is degraded. With the pipeline fashioned transmission, the probability of data loss and errors due to the low voltage levels is decreased. Also other important issues for complex SoCs are scalability and modularity. The bus based systems are not suitable for scalable systems on chips due to their static nature. Whereas NoC solutions offers flexible, scalable and configurable infrastructure for modern SoCs.

The major elements of NoCs are routers which connects the PE (processing element) or memory blocks to the rest of the system via suitable network interfaces. The design of routers for NoCs is challenging since they have to provide a satisfactory level of data throughput while they have to occupy less area relative to the processing elements and consume relatively small amount of total power budget of the chip. Moreover, due to the nature of the NoCs and SoCs, they have to be flexible, scalable and configurable. So there are a lot of work done for design of these routing elements for the new communication infrastructure of SoCs.

In this thesis work, it is aimed to design a network on chip router, AFRONOC. The main goals for the design are to make the router flexible, configurable and adaptive to the changing network topologies. Like every design targeting the limited on chip resources, the design should consume as small power as possible and occupy small area. Under these design tasks the work done in the scope of this thesis work resulted with a flexible and configurable NoC router. Moreover, the design provides infrastructure for reconfigurable systems since it have a stand alone architecture without any need of initializing from an outer central network monitoring unit. This property would be beneficial for dynamic reconfigurable systems. In such systems, although there is a finite amount of hardware resources, these resources are used in a time sharing manner and virtually the amount of resources converged to infinity. For such systems there can be topology changes of NoC system in time or some additional processing nodes can be inserted to the system or can be vanished from the system. For such situations,

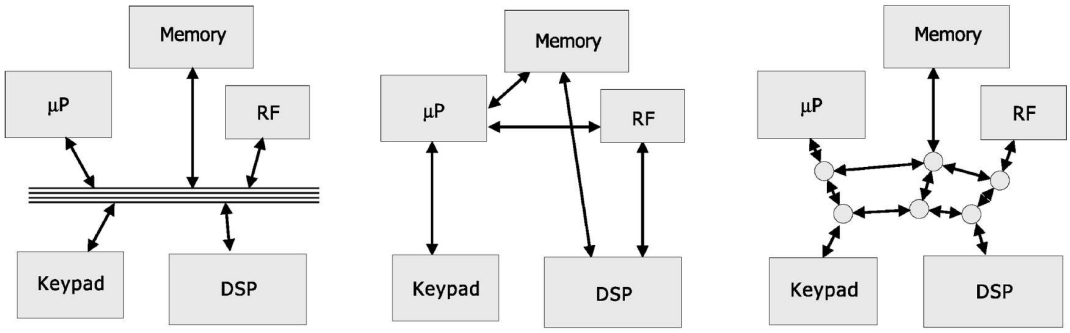
AFRONOC router provides a reasonable level of adaptability to the rest of the system.

The organization of this report is as follows: In Chapter 2, a brief information on the SoC interconnections and the network level for NoCs are described in detail and at the end of the chapter, some realizations from the literature are given. In Chapter 3, the architecture of the AFRONOC router is described in detail. After that, the proposed adaptation and routing algorithms are explained. The implementation and some experimental results are explained in Chapter 5. Next, discussions and some comparisons to the related work are summarized. And in the final chapter, the conclusions and future works for the thesis are given briefly.

2. SoC INTERCONNECT AND NoC BASICS

2.1. SoC Interconnects

The increasing number of processing elements for complex, parallel or distributed processing applications has brought the high density on-chip systems into consideration for the VLSI world. The traditional SoC structures utilize the bus based connection topologies due to their simplicity. However as these systems have grown rapidly and the production technology has scaled into sub-micron processes, the requirements for on-chip communication have changed [2], [3]. Figure 2.1, taken from [2] shows alternative methods for SoC communication.



(a) Bus Based Interconnect

(b) Point-to-Point Connection

(c) Connection with a Network

Figure 2.1. Different communication infrastructures for SoCs [2].

According to [2], since the communication tasks have turned out to be more expensive than the processing tasks for on chip systems due to the scaling of microchip fabrication, the challenging issues for on chip communication such as power consumption on relatively long wires and propagation delays of electrical signals have become cirtical issues to be considered. The main reason is that the on chip wires do not scale in the way the transistors do. This phenomenon is illustrated by the International Technology Roadmap for Semiconductors 2001 report [4], as shown in Figure 2.2.

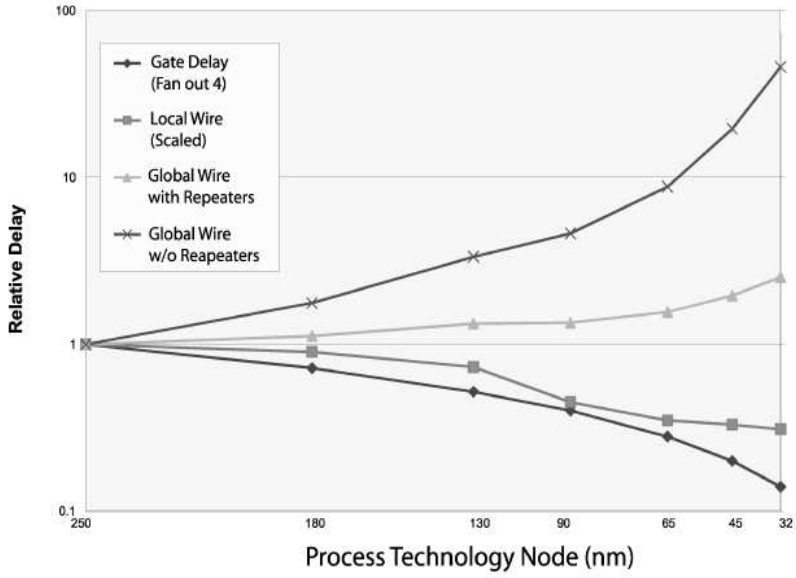


Figure 2.2. The Relative Relation Between on-chip Wires and Gate Delays for Different Near Future Technologies from ITRS 2001 report [4].

Figure 2.2 shows that as the gap between the relative delays of the driver gates and the delays of global wires increases, the wire capacitances will be dominant on the parasitic capacitances of the driver transistors as loads. The situation for local wires is different since they can scale more with the technology comparing to the long global wires. Moreover, the wire models for these sub-micron processes are nondeterministic because of fabrication uncertainties, cross-talk and noise sensitivity issues [2]. As the length of the wire grows, the probability of unreliability on the signal integrity model increases.

Another important issue is the global synchronization problem. For deep sub micron processes, the distribution of one global clock all over the large chips is a challenging issue for the designers [2], [5]. The main reasons are, the unpredictable behavior of clock skew and the power budget of clock-tree distribution methods.

The reusability of the designs for SoCs is a powerful tool in order to close the design productivity gap [2],[5]. Since scaling of microchip technologies continues, every

new technology comes with problems which are special to those technologies. So in order to catch the technology, the design times or the number of designers have to grow up quadratically unless reuse of complex building blocks of SoCs is taken as a solution. So, the scalable and reusable NoC infrastructures can be pointed as solutions for the global communication of single chip systems.

Some of the introductory works about the NoCs are proposed in [6] [7], [8]. In those proposals, the basic concepts about on chip interconnections and networks, initial trends and motivations underlying behind the subject are represented generally. For understanding basics of NoC research, an illustration is given in [2], which is shown in Figure 2.3.

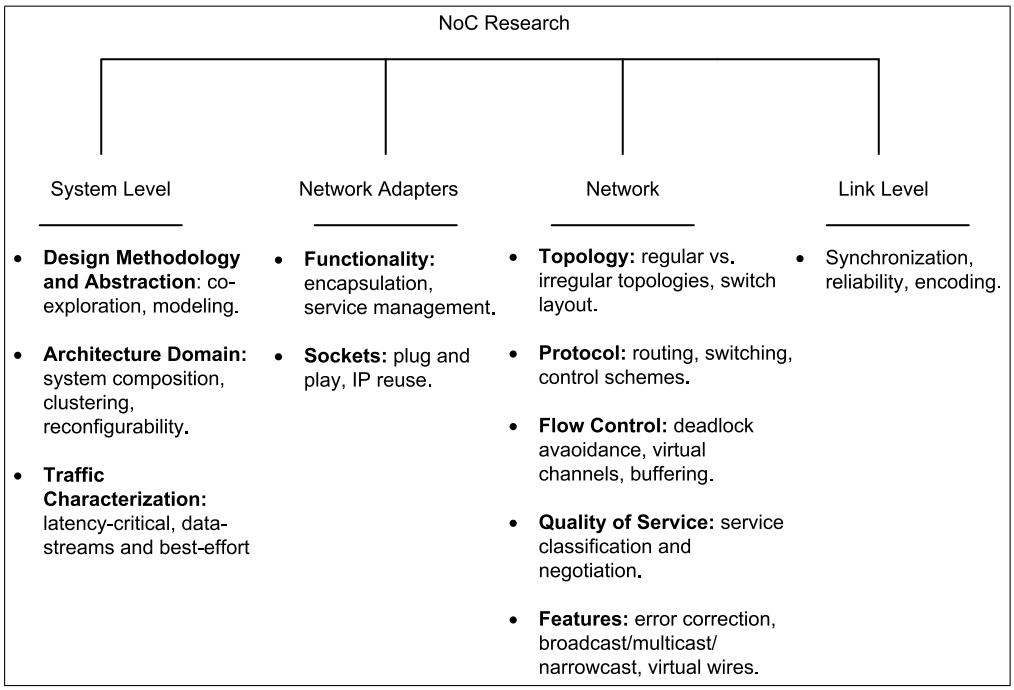


Figure 2.3. The NoC research classification [2]

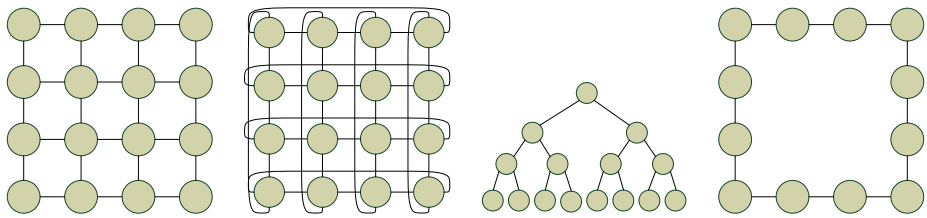
In the scope of this thesis, the most of the interest is zoom in on the router architecture, that is, the network level. So before going into explanation of the proposed router architecture, the network level NoC characteristics are explained briefly in following sections.

2.2. NoC Characteristics for Network Level

2.2.1. Topology

The topology of a NoC system defines the physical connectivity of the nodes of the network. The topology can vary according to application and communication requirements of a SoC on which the NoC infrastructure will be realized. The topologies can be classified into three main groups: Regular topologies, irregular topologies and mixed topologies. In the following subsections some well known and frequently used NoC topologies are explained.

2.2.1.1. Regular Topologies. Most of the regular NoC topologies have been derived from the general network topologies : Grid based k-ary n-cube topologies such as 2-D mesh, 2-D torus, tree based topologies such as binary tree and fat-tree topologies, ring and crossbar topologies. The illustrations of these topologies are given in Figure 2.4.



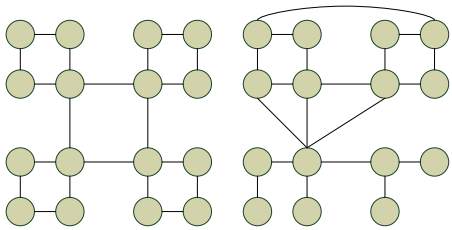
(a) 2-D Mesh Topology (b) 2-D Torus Topology (c) Tree Topology (d) Ring Topology

Figure 2.4. Some Well Known Regular NoC Topologies

There are some 3-D topologies used in literature [9],[10],[11]. In [12], the teraFLOPS chip is explained in detail. It is known as the largest implemented topology in literature to date with a 80-node 2-D mesh topology. It also supports the idea that regular topologies are more suitable for the CMP implementations.

2.2.1.2. Irregular and Mixed Topologies. Irregular topologies or custom topologies may give flexibility to a NoC system to be independent of restrictions dictated by topology such as the freedom on the choice of the routing algorithm. Also at chip layout phase, they give the freedom of placing the design modules in a desired manner. So, such a NoC router that can work even in irregular topologies would provide a level of flexibility in terms of system layer for a complete NoC design. The work on irregular topologies in literature are rare but it is believed that much more effort should focus on this issue and some examples can be found in [21] and [22].

Mixed topologies are mixture of regular topologies and can be used for specific applications according to the system requirements. In Figure 2.5, two examples are shown for irregular and mixed topologies.



(a) Mixed Topology. (b) Irregular Topology.

Figure 2.5. Two Example Illustrations for Irregular and Mixed Topologies

2.2.2. Switching

Switching policy can be defined as the way of the data path establishment between a source node and a destination node. For traditional SoC interconnects, the data is transmitted via dedicated wires with a determined and fixed width. For NoC concept, there are two known switching policies, circuit switching and packet switching.

2.2.2.1. Circuit Switching. Circuit switching is the derived form of traditional method in such a way that a virtual circuit is established between the source and the desti-

nation node before the communication starts. The data transmission is done after the establishment of the dedicated path. After the delivery of the data, the circuit is released. The long establishment period is the disadvantage of this kind of switching. However once the circuit is established, the data transfer is done in a few clock cycles which decreases the data delivery time. Examples for circuit switched NoCs can be found in [23] and [24].

2.2.2.2. Packet Switching. The packet switching technique is more utilized method for data transmission in NoCs since it is suitable for scalability. Moreover since packet switching avoids occupation of long dedicated paths, it increases the utilization of the resources on the NoC system. The disadvantage for packet switching is that it requires buffering and hence there will be a drawback by means of area when compared to the circuit switching technique. Most of the routers designed in NoC literature use packet switching method, some of them can be found in [25].

2.2.3. Routing in NoCs

Routing in NoCs is simply, determining a path for a data packet or flit that comes from an upstream router or from the source node to a downstream router or to the source node. There are several routing methods that can be found in literature [2], [3]. A classification for routing algorithms can be done by using their frequently observed aspects.

2.2.3.1. Minimal vs. Non-Minimal Routing. If a routing algorithm dictates to route the packets always on the shortest paths, it is minimal, otherwise the routing algorithm turns out to be non-minimal [2].

2.2.3.2. Table Based vs. Algorithmic Routing. In table based routing methods, the information for determining a path for a packet is kept on a look-up table. On the other

hand for algorithmic routing, the route is determined by processing the information that belongs to the packet in a routing algorithm. The information for the packet can be, for example, the channel it came from, destination and source addresses or some other algorithm-specific information. The table based methods decrease the latency since the information is always ready and usually needs a small amount of processing. The algorithm based methods can increase the latency however may be more suitable for adaptive applications.

2.2.3.3. Deterministic vs. Adaptive Routing. If a path chosen for a given destination-source pair is always the same for any instances of the network operation, the routing method is said to be deterministic. If the path varies according to some parameters (for example congestion situation of a link), then the routing algorithm is adaptive [2],[3]. The advantage of deterministic routing is its simplicity when compared to the adaptive methods, however there might be possible contentions and unnecessary wait states which might end up with increased latency for the packets. Adaptive routing increases the utilization of the bandwidth of the network with a penalty of increased complexity of the system.

2.2.3.4. Deflection Routing. Deflection routing is also known as hot-potato routing method, first tries to route the packet to a minimal path, if there are no available channels for a minimal path, then it forwards the packet to a non-minimal or randomly selected path in order to avoid the buffering and locking a specific channel. So deflection routing supports buffer-less designs which may yield to a relatively small area while increasing the probabilities of live-locks and unnecessary traveling of packets due to its nature.

2.2.4. Network Flow Control

Network flow control describes how the flow of the data traversing from one router to another router is arranged. Flow control mechanisms can be classified into two main sub groups which show parallelism to the switching technique.

2.2.4.1. Buffered Flow Control. In buffered flow control the incoming data is buffered at each router. The data here can be the whole packet or some smaller part of a packet (i.e. flit). There are three well-known buffered flow control mechanisms, which are store-and-forward, virtual-cut-trough and wormhole flow controls.

In store and forward flow (SAF) control, the whole packet is buffered in one router and then the route is determined and after that, the packet sent to the other node. Virtual-cut-trough (VCT) method is the slightly modified version of store and forward mechanism. In VCT, the packet is sent to the downstream router as soon as the header information is received and resolved, if the next router's input buffer is available. If the next router is not available for accepting a new packet, then the packet is buffered in the current router similar to the SAF flow control. A comparison for these two flow controls is given in Section 3.3.2. The last flow control mechanism is the wormhole control (WH), where the necessity for buffering the whole packet is avoided and the packet is sent to the next nodes in a flit-by-flit manner. Once the header flit is routed to the next destination, the following flits come after in a pipelined fashion. In this manner, the buffering and the waiting times of header flit are decreased. However, without the concept of virtual channels, the network can saturate and be blocked quickly. The virtual channel concept avoids deadlocks and provides a level of parallelism to the flow of the different packets on same physical channels. If there are more than one virtual channel (VC) on a physical channel and one of the virtual channels is blocked, the other ones can serve and the data flow would not be blocked.

The positioning of buffers can be an important issue for buffered type flow control. The buffers can be at the input, output or in the middle. A comparison for buffering issues for general VLSI switches is given in [26]. In [27] and [28], the problems of buffer allocation and different buffering methods are considered in detail. The general idea is that output buffering or both input-output buffering outperforms single input buffering. The reason for this result is that, if an output channel is busy at an instance and some input channel is waiting for that output channel, the input channel fills its buffer and waits for the output channel to be available. So, the input channel is blocked in this scenario, the situation is called head-of-line blocking. However if the buffering is done at the output side, the input channel may accept a new packet for routing, and if this new packet's route is a different output channel from the former one, then the flow is not blocked. A comparison for VCT and WH flow controls is given in [29], which shows that VCT flow control outperforms WH without virtual channels in many cases.

2.2.4.2. Buffer-less Flow Control. Buffer-less flow control is for circuit switching and deflection or hot-potato routing algorithms. Since all these techniques do not need buffering of packet. Usually buffer-less techniques are considered for decreasing the area overhead, but rarely used. They are the very common and frequently discussed characteristics of NoCs. For further information and different characteristics, the reader can study the literature surveys [2],[3] and [30].

2.3. NoC Routers in Literature

In this section, some router realizations from the literature are discussed [3], [30]. The architectures mentioned here are investigated in the scope of this thesis work.

An example is the DyAD smart routing chip [13]. In this architecture, the router senses the density of the network (especially local network) dynamically and switches

between the adaptive and deterministic routing methods according to the congestion status. With deterministic routing method, generally the shortest-path algorithm is applied. The main advantage of the deterministic routing is, simple logic and low message latencies. However under dense traffic, the traditional routers are not sensitive to the local densities. Hence, this results in very long delays and there may be deadlocks. In congested traffic cases, DyAD router switches into adaptive type routing algorithm and utilizes alternative routing paths and tries to solve the traffic problems. Basically, each router monitors the traffic status of the neighbor routers and decides the type of the algorithm that should be used while determining the paths of the new packets from the neighbors. Similar dynamic structures are also proposed in [14] and [15]. In [13], the authors have underlined some aspects that should be taken into account while designing routers for on-chip networks: In order to obtain low area designs, registers should be used instead of big memories. If a kind of deterministic routing is targeted, XY-routing might be suitable since it is deadlock and live-lock-free. If an adaptive routing algorithm is aimed to be chosen, minimal odd-even routing is a live-lock free algorithm and might be a good choice. In our work, we proposed a set of parameters, which makes the router configurable. These are, the link width(data and address bus width), depth of input and output FIFOs and the flit sizes(the atomic message packet on the network) could be reconfigurable and these could be parameters for such a router.

Another router algorithm is proposed in [16], which is a good example for custom structures used in NoC routers. In that work, a new structure that has no virtual channel is proposed. Important feature of this architecture is that by eliminating the virtual channels, the complexity of the router logic is decreased and this results in a decrease in setup latency. The proposed router is based on a k-ary n-mesh type network. It is partially adaptive and dead-lock free. The virtual channel concept is important for NoC routers and used in several architectures. Virtual channels at input or output terminals of the routers provide a separation of granting of buffers and granting of

channels. To do this multiple buffers are utilized for each channel of the router. In other words, if two particular paths over a router share one particular channel of that router and if one of the paths is blocked and the other one is free, the free path is used by using virtual channels. However, there are some problematic aspects for the virtual channels. Firstly, additional buffers, larger crossbar and more complex arbitration increase the complexity of the router architecture [16]. The second disadvantage is that the virtual channels come with the cost of increased setup latency. So if the virtual channel solution is chosen for avoiding the deadlock problem that would possibly occur during the communication between nodes of a NoC system, these disadvantages have to be taken into account. The methodology proposed from the authors of [16] can be taken as a solution for deadlock problem. The important factors that affect the performance of the network are the topology of the network, flow control strategy and the routing algorithm. By taking into account those factors, some general and generic router architectures have been proposed in the past. A well-known architecture is a basic work for this subject and proposed in [17]. A simplified architecture block diagram is shown in Figure 2.6. Here, the main sub-modules of the router are address decoder (AD), flow control unit (FC), crossbar(CB), routing arbiter(RA), virtual channel controller(VCC). This model is proposed for modeling latencies those could occur and effect the setup latency of the router. This router model has been considered as a rough model for our AFRONOC router.

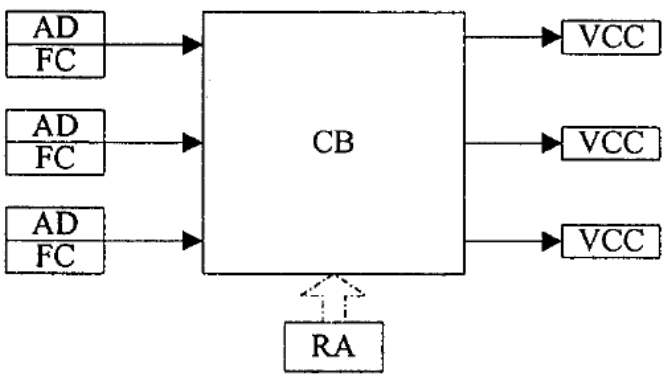


Figure 2.6. Chien's Router Model Simplified Block Diagram [16]

The work proposed in [18] describes the characteristics and feasibility of the routing topologies. In this work routability of some well known network topologies was studied. When deciding on the routability degree of a network, its logic utilization degree, logic distribution (area), maximum clock frequency, number of nets, placement and routing time is measured. As a result, they observe that some of the routing topologies as routable and some of them as not routable, which they investigated. It is seen that while examining the routability of a network topology, three main characteristics of the topology could be estimated. These are, node degree, the number of links from a node to its neighbors, link complexity, number of links that topology requires and regularity. If all nodes of the network have the same node degree, that network is said to be regular in [18].

In [19], different issues discussed about interconnections on reconfigurable devices such as FPGAs. The authors compared flexible interconnection concept and statically wired multicore systems. Important benefits of reconfiguration ability of interconnects on digital systems are briefly given with comparison to fixed networks. Then, these benefits have been shown based on some experimental results. As a result of the brief comparison of rigid interconnections and FLUX Networks [19], it is seen that the most important difference between dynamic and static systems is the changeability of the physical structure of the network. Since FLUX networks are reconfigurable, it gives the ability to the designer to switch between different network topologies such as mesh, torus, ring, etc. Since the network is rigid in static structures, the parallel system realized by that hardware cannot be parameterized. For example, if the link width is chosen 32-bit it always remains same during life time of that hardware. So parametrization of hardware can provide some advantages such as switching between design for performance or design for low power concepts. In [19], a simple example is also stated for reconfiguration of the interconnects as illustrated in Figure 2.7. Here, switching from a binary tree topology to mesh topology is stated. At the programming side of this instance, the bitstream of the desired network topology is downloaded to

the reconfigurable fabric at the "SET Network" phase and the execution of the program continues from the last instruction.

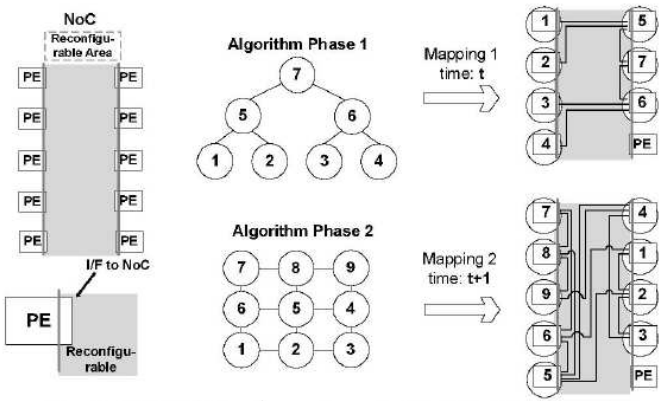


Figure 2.7. Example Illustration for Reconfiguration of Interconnect Hardware [19]

In [20], some well-known network topologies and their performance metrics are studied. First, some common concepts are discussed such as switching techniques for on chip networks. Then, an infrastructure in terms of performance parameters for on chip networks is proposed. After representing a methodology to evaluate performance of a given network, some experimental results are given. Also a switch architecture is given in scope of this work. In [20], the most important performance parameters for on-chip networks are given and formulated with the aid of a sense that is modeled from the classical network and communication theories. These main and the most important metrics are, message throughput, transport latency, energy and area. They are also well-known parameters for digital systems and communications systems. However, they are re-defined in terms of on chip network parameters in the scope of [20] .

A motivating work is represented in [22]. A topology adaptive network on chip design is proposed in this work. The router designed in this work is a kind of flexible router in terms of input and output channel number and the routing topology that router uses. The authors have tried to keep the router adaptable to different network topologies while fixing the switching technique and routing algorithm. In order to do that, virtual cut trough switching technique and a deterministic routing algorithm are

used. Flexibility to the network on chip routers and balancing the network traffic is achieved with a little area overhead.

In [21], a circuit-switched flexible router is described. The update of routing tables is partial reconfiguration based and not mentioned in detail. The flexibility of the router is in terms of number of channels, data width, address width and these parameters are configurable in synthesis time. So, the design needs the arrangement of the routing tables outside of the network and embed it to the router by partial reconfiguration.

3. AFRONOC: THE FLEXIBLE ADAPTIVE ROUTER DESIGN ARCHITECTURE

In this chapter the design goals and the architecture of the AFRONOC router is described in detail. The router has four main functional blocks: table update block, input block, central routing-switching unit and the output block. The organization of this chapter is as follows: In Section 3.1 the structural view of the router is presented. Its building blocks are explained in the remaining sections.

3.1. General Organization of Architecture

There are three main objectives at the design phase of the router. First objective is to design a flexible router. AFRONOC router is design-time condigurable in terms of number of channels, the number of nodes in the network, buffer size of each channel and physical data width as shown in Table 3.1. At the design implementation phase, the router is realized by using VHDL language. The flexibility is provided by using several *generate* statements in sub blocks of the router HDL code. The parameters for router are collected in a package file. These configuration parameters and a configuration manual for the router is given in Appendix A.

The second objective is to design an adaptive router. Here the implication made by saying "adaptive" is that, the router should recognize the network it is connected to without any external or central monitoring unit. Such a property will bring a stand-alone working principle to the router. This is achieved by the proposed adaptation algorithm which is described in detail in Section 4.2.1. On the design side, the algorithm is implemented as the table update block (TUB) which is described in Section 3.2.

The third design goal is to keep the design simple in order to keep the area of the

Table 3.1. Configurable Parameters of the Router

Parameter	Description	Bound	Effects
WIC	2^{WIC} gives the number of input channels	$2^{WIC} = 32$	Input block area , Switch area, number of hello-output channels
WOC	2^{WOC} gives the number of output channels	$2^{WOC} = 32$	Output block area, Switch area, number of hello-input channels, number of 1-bit address-channel match table RAMs
WADDR	2^{WADDR} gives the maximum number of nodes in the network	Up to HW resources and application	Address-channel match table RAM depth (area), Current minimum hop count RAM depth at hello center unit (area), hello message width, data packet flit widths
WDATA	Communication data width	Up to application	IO buffer widths (area), Switch data width (area)
BUFDEPTH	Depth of input-output buffers	Up to application	IO buffer depth (area)
HELLO_ REFRESH_ PERIOD	Period of sending new hello packet to network	To be optimized according to topology	Table update speed

each block as small as possible. Since the final work will be targeted to reconfigurable architectures such as FPGAs and large SoCs which have resource restrictions, the interconnections must occupy as small area as possible in order to leave more area to the functional processing units at system level. This point is taken into account in the design of every block of the router.

So by keeping those design goals in mind the proposed router architecture is given in Figure 3.1. In Table 3.2, the main properties of the router and the advantages and the disadvantages of the selected feature are summarized.

Table 3.2. Main Properties of the Proposed Router

Property	Description	Advantage	Disadvantage
Switching	Packet Switching	*Low Latency*	*Area Overhead*
Routing	Custom-Adaptive (States of the output channels + alternative minimal paths)	*Adaptivity*	*Area Overhead*
Network Flow Control	Buffered using "Virtual Cut Trough" with request-acknowledge control	*Simple Control Logic*	*Possible Head of line blocking*
Buffering	Both input and output buffering	*Avoid Head of line blocking, Reduce deadlock possibility*	*Area Overhead*
Topology compatibility	Both regular and irregular topologies	*Flexibility, Adaptivity*	*Area Overhead*

3.2. Table Update Block (TUB)

3.2.1. Organization of TUB

Table update block is responsible of managing the "hello" messaging which takes place on the dedicated paths. Dedicated paths are used for the adaptation task in order to achieve a parallelism between hello messaging and data messaging. By doing this, the hello messaging does not disturb the data lines while bringing a little overhead on

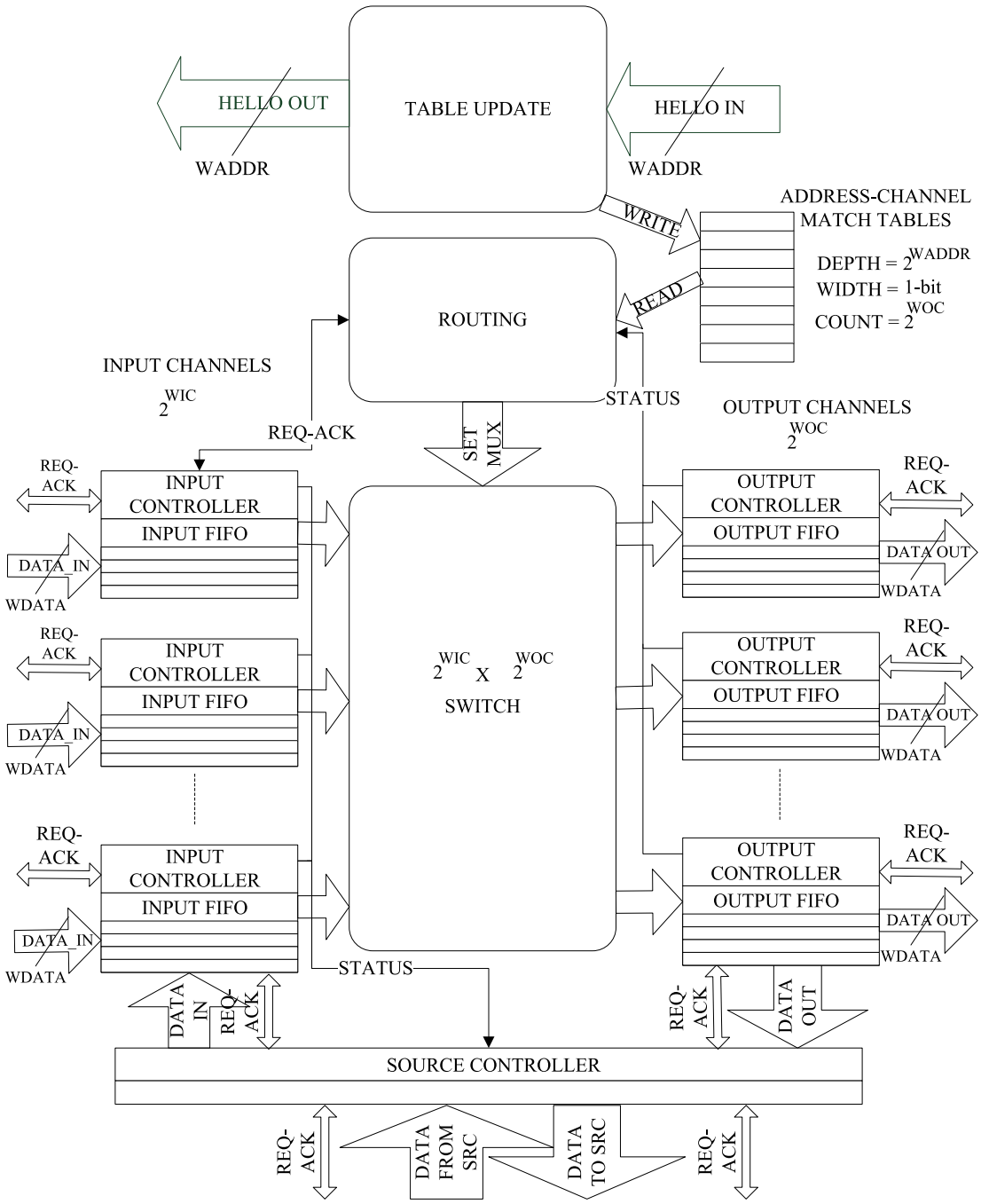


Figure 3.1. General Organization of the Router.

silicon area at system level. TUB generates and sends the hello message of the router to the neighboring routers periodically. The period is design time configurable. Another function of this block is to collect the hello messages coming from all neighbors of the router, to decide whether this information will make a change on the address-channel match table and if so, make this change and make appropriate changes on the hello message and send it to all neighbors. The decision algorithm and the hello messaging structure details are explained in Chapter 4. The block diagram of TUB is given in Figure 3.2.

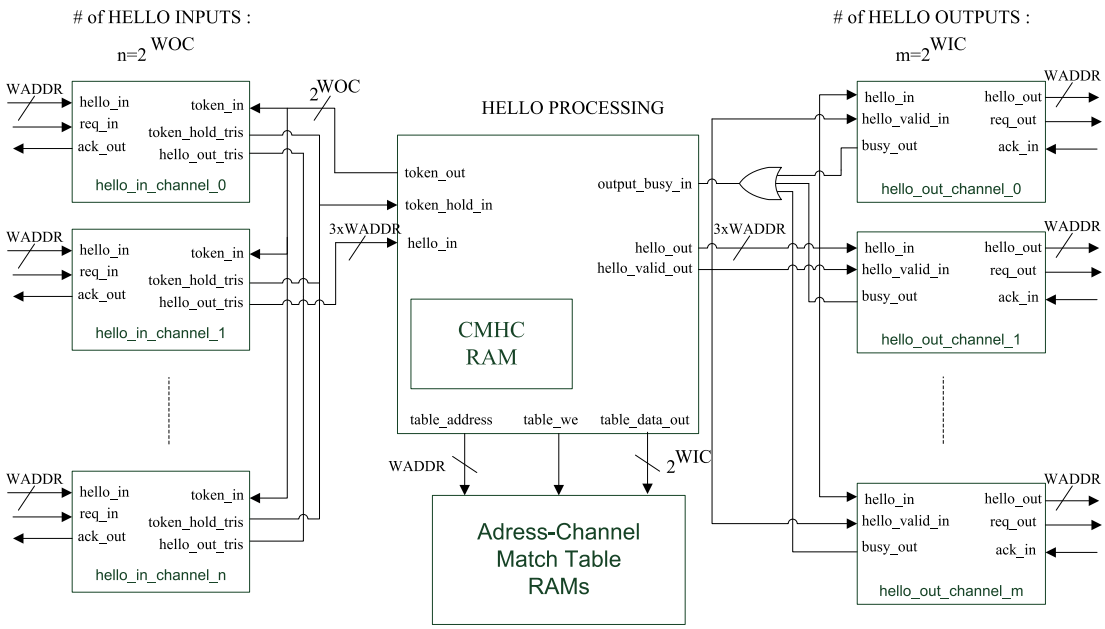


Figure 3.2. Table Update Unit Block Diagram.

The input block is composed of 2^{WOC} hello-input channels since there are 2^{WOC} output data channels of the router and since the hello messages come from the routers at the output side, this choice is reasonable. Output block is composed of 2^{WIC} hello-output channels in a similar fashion. The central block is composed of an FSM and the CMHC RAM (Current Minimum Hop Count RAM) mainly. There are two main tasks for this central block. First, it is responsible of sending the hello messages of the router itself. Processing the hello messages that come from the neighbor routers is the second main task. The details of these blocks are described in the following subsections.

3.2.2. Hello-Input Channels

The main task for a hello input channel is waiting for a hello message from the hello-output channel of the neighbor router which it is connected to. The hello-input channel is designed as a finite state machine. The state diagram of the hello-input channel is shown in Figure 3.3. The incoming hello message is captured via a shift register and registered to the central unit with the invocation of the token_in input.

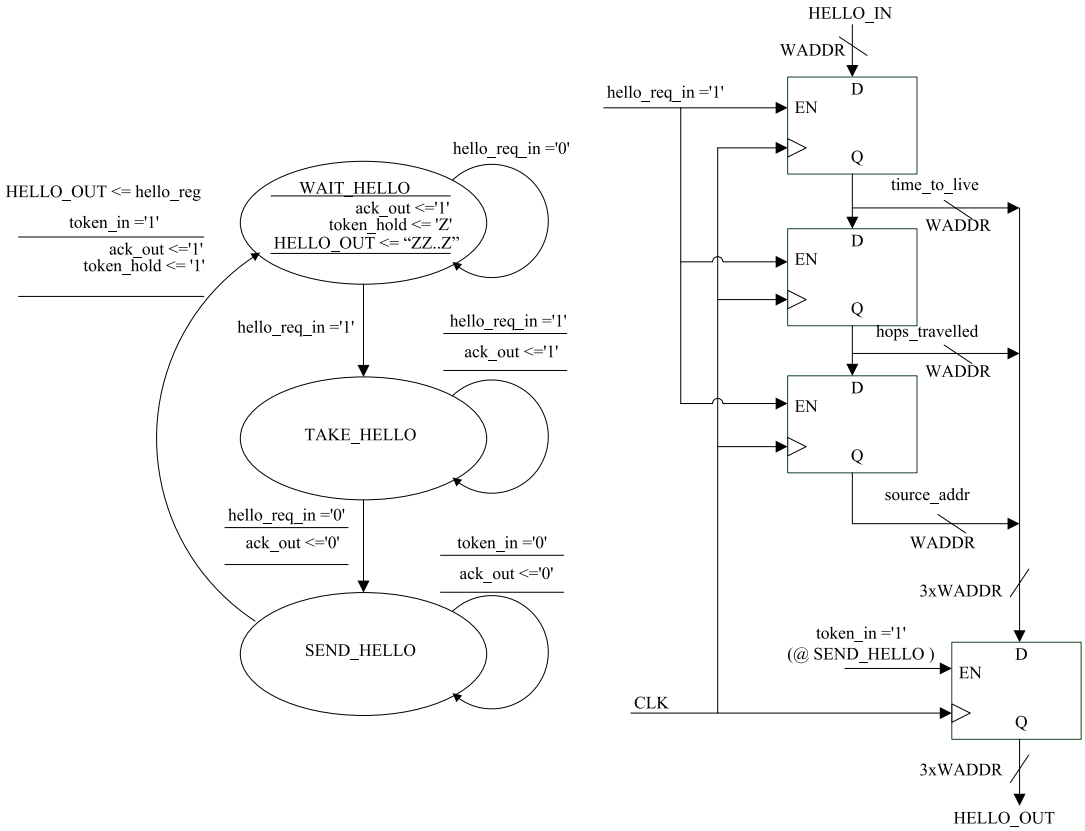


Figure 3.3. Hello Input FSM and logic.

3.2.3. Hello-Center Decision Block

Hello center unit turns a cyclic token every clock cycle and polls the *token_hold* input from the input channels. Whenever there is a new hello message, it begins to process this new hello message. It controls the *source_address* field first. If the origin of hello message is itself, then it drops the packet. If not, it checks the *hops_travelled* and

time_to_live fields for making appropriate changes on CMHC RAM and the address-channel match table RAMs. The detailed description for this processing algorithm is given in Section 4.2.1. The hello processing unit also sends the hello message of the router itself periodically. This period is determined at synthesis time by designer by changing the HELLO_REFRESH_PERIOD parameter which can be seen in Table 3.1 and in Figure A.1, in Appendix A. Choosing a relatively long period may yield long initialization times. Also choosing too short periods may cause contentions in the hello-message network which may again cause long initialization times. This period should be optimized with respect to the topology and size of the network.

3.2.4. Hello-output Channels

Each hello output channel is composed of a finite state machine and two counters. It waits for a hello packet from the central unit and informs the center unit about its situation by setting *busy_out* flag to zero. When central unit sends new hello packet it raises the *hello_valid* signal and the output units capture the new hello message. After receiving the hello message, output channel checks if the downstream hello input channel is ready for a new message and if so it sends the hello message via a shift-register. If the downstream input channel is not ready for eight clock cycles, the timeout counter generates a *timeout_expired* signal and the output channel drops the message and waits for a new one. This timeout is for avoiding the possible deadlocks and fixed. The architecture for hello-output channel is summarized in Figure 3.4.

3.3. Input Channels

3.3.1. Overview

The input block is one of the main blocks of the router. It is composed of a configurable number of separate input channels and an input source controller block.

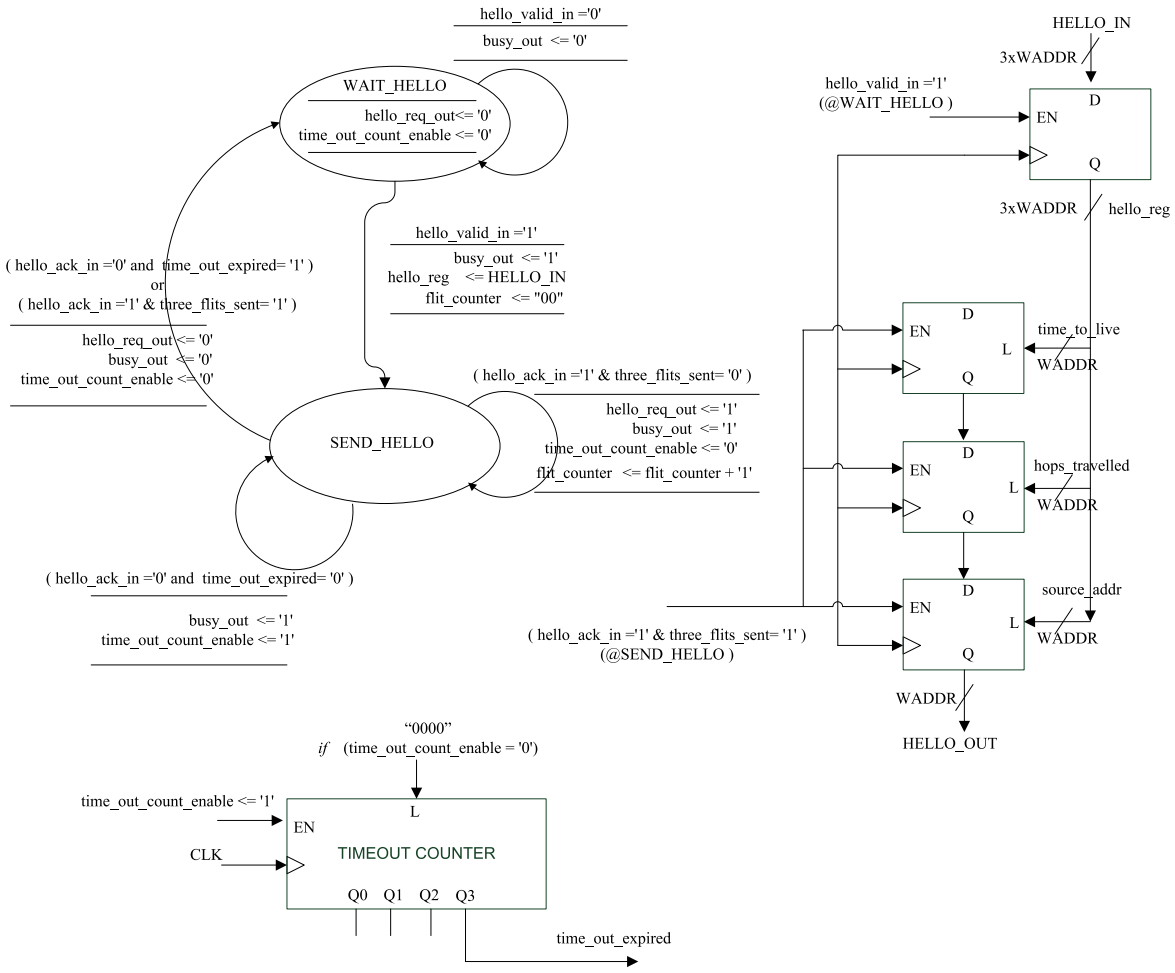


Figure 3.4. Hello Output FSM and logic.

Each input channel has a finite state machine which controls the network flow, makes route request to the routing unit and sends data to output channels via the switch. There is one FIFO buffer at each input channel for buffering the data flits. The block diagram of the Input block is shown in Figure 3.5.

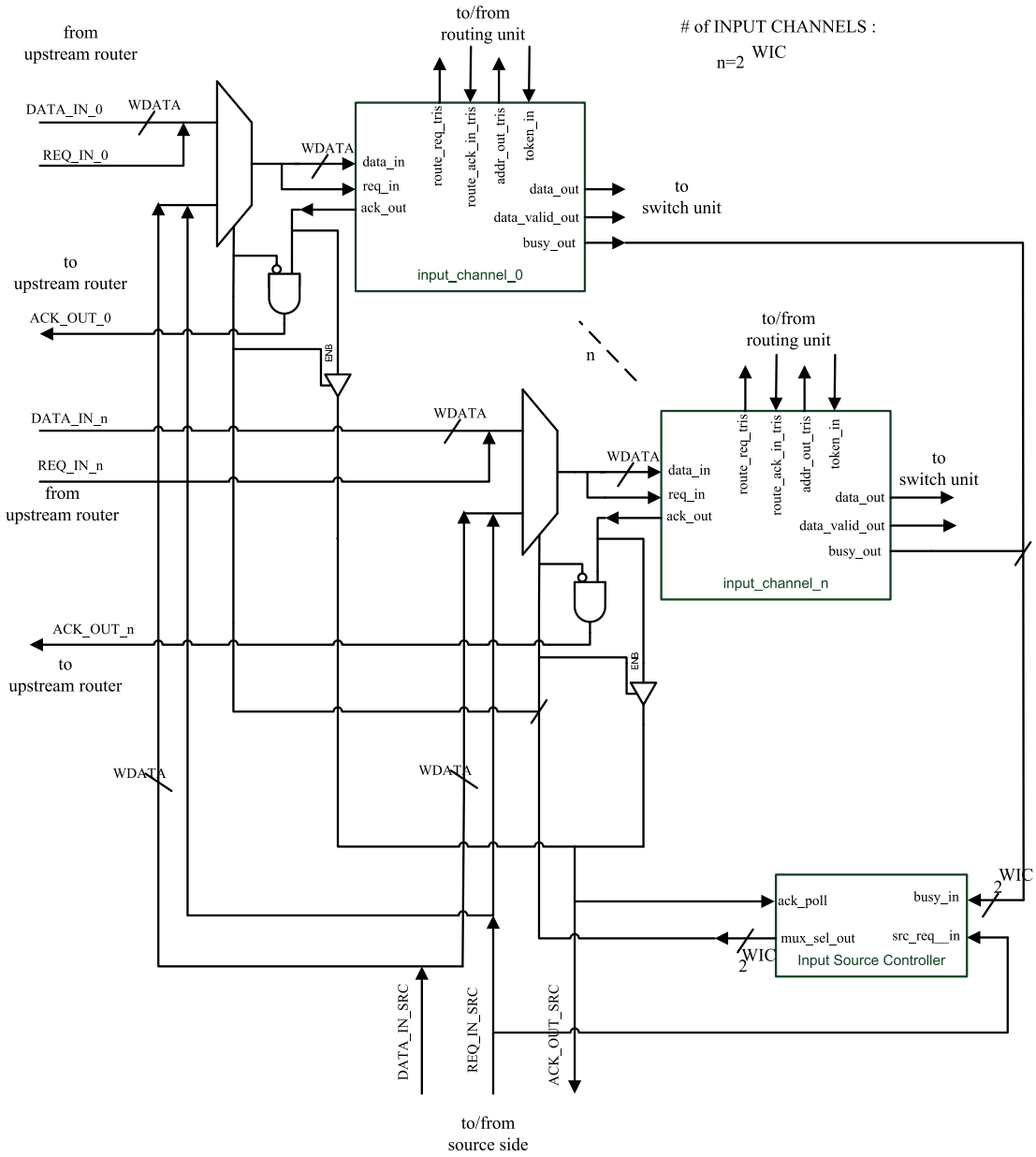


Figure 3.5. Input Channels Block Diagram.

The data, request acknowledge lines from/to source side (i.e. from/to connected processing element of the node) goes directly to the input channels via multiplexors.

By this way the buffering of the injected data done in input channels and so annihilates the need of buffering the data in a separate source buffer. This improves the utilization of input channel buffers and comes as a contribution of the proposed architecture. In the following sections the details of inner blocks and the source controller unit are described.

3.3.2. Input FSM and Buffering

Input block is composed of input channels and each input channel consists of a finite state machine and a FIFO buffer for keeping the data packets. The FSM waits for new data from upstream router/source side, buffers the packet if needed, make routing request to the central routing unit, and sends data to the output via crossbar switch. While sending the data to the output, input channels send the control signal *data_valid* to the output channel. If there is a packet in the input channel, it sets its busy flag to zero and informs the source controller about its state.

Input FSM diagram is given in Figure 3.6 for two different flow control mechanisms, SAF (Store and Forward) and VCT (Virtual Cut Through). All of the registers, inputs and outputs are not shown for simplicity. At synthesis time, the flow control can be changed to the SAF or VCT by commenting out the related FSM source codes. The difference of the two flow control mechanisms can be described shortly as follows as illustrated in Figure 3.7: In SAF, after receiving a request from the upstream router's output channel, the input channel of the current router sends an acknowledge to the upstream router if the packet buffer is empty (at time t=0 in Figure 3.7). It, then, buffers the whole packet in T_b time (buffering time), sends a routing request to the routing unit with the destination address information of the packet. After receiving the route acknowledgement after a time delay of T_r (routing decision time), sends the packet to the multiplexer of the assigned output channel in T_s time (sending time). The VCT (Virtual Cut Through) flow control is first described in [31]. In VCT flow control,

input channel of the current router sends acknowledgement to the upstream router if the packet FIFO is empty and as soon as it receives the header flit (the destination address information) in T_{bh} time (buffering time of the head flit), it sends the route request to the routing unit and as soon as it receives the route acknowledgement after a T_r time delay, it begins to send the packet flits to the output channel without waiting the buffering of the whole packet. If there is a contention on the output channels side, the VCT converges to the SAF flow control since there will be a need for waiting for output channels to be free. If the packet lengths are so long that the routing decision time is negligible comparing to the buffering time of the packet ($T_r << T_b$), the VCT is superior to the SAF, especially for contention free situations as illustrated in Figure 3.7

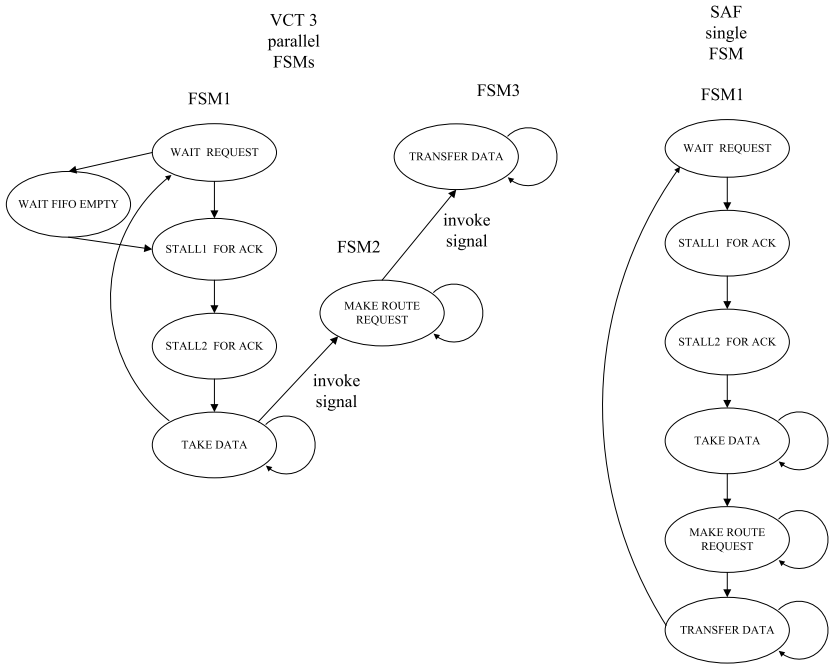


Figure 3.6. Input Channel FSM Designs for SAF and VCT.

3.3.3. Input Source Control Unit

The input source control unit mainly waits for a request from the source side and when a request occurs, it reflects this request to one of the input channels. The choice of the input channel is dependent on the states of the input channels. The source

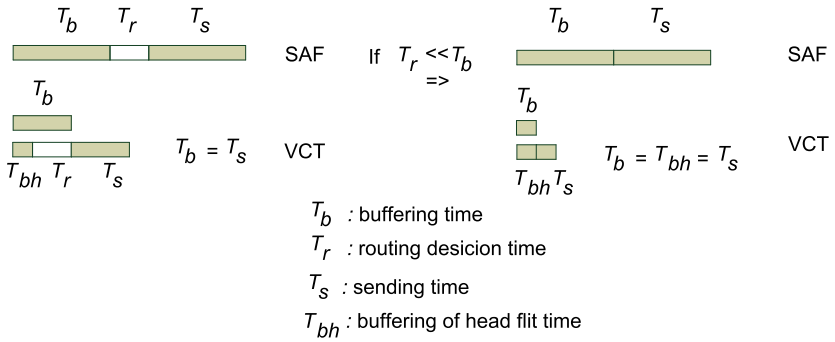


Figure 3.7. Timing comparison for VCT and SAF control flow mechanisims.

controller rotates a cyclic token and gives equal chances to the input channels so as to reduce the possibility of contentions on a specific input channel. The architecture of the input source controller is given in Figure 3.8. This architecture for connection to the processing element allows usage of the input channel buffers as source-side buffers and improves utilization. So extra buffering may not be necessary at the source controller while injecting data to network. This will yield to an increase in flit injection capability.

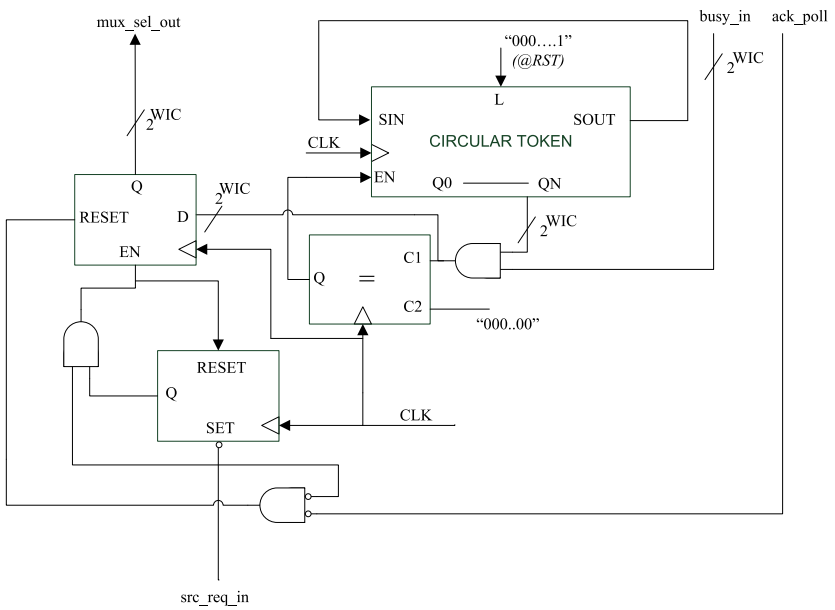


Figure 3.8. Architecture of Input Source Controller.

3.4. Central Unit: Routing and Switching

3.4.1. Overview

The central block makes a match between a given destination address and an appropriate output channel. It, then, establishes a physical route between that output channel and the input channel which makes the request. There are three main blocks in CU architecture: Routing block, flexible switch and the Address Channel Match Table (ACMT) RAM. The block diagram of the routing and switching block is shown in Figure 3.9. The inner main sub-blocks of CU is described in following sections.

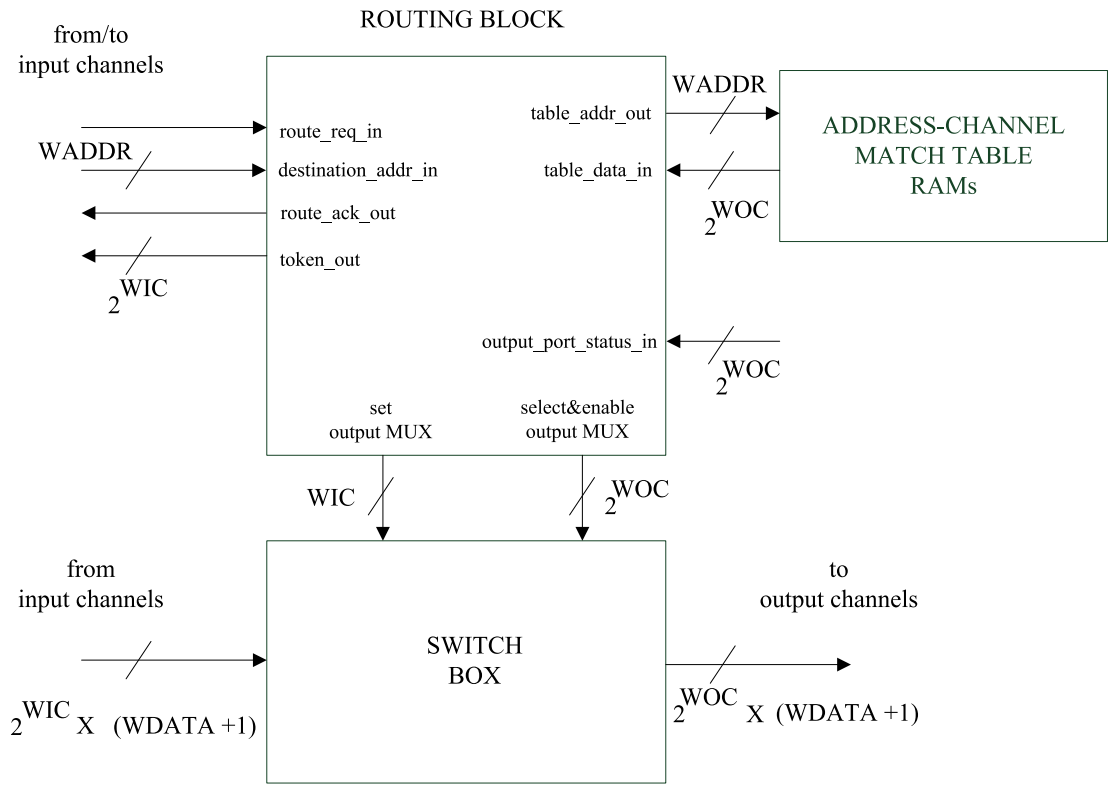


Figure 3.9. Architecture of Routing and switching unit.

3.4.2. Routing Block

The routing block takes the destination address information and "match requests" from the input channels, searches for possible routes on the address channel match

table, monitors the states of the output channels and sets the switch with the aid of the final information appropriately. The FSM for routing task and the additional logic are shown in Figure 3.10. All signals are not shown for simplicity.

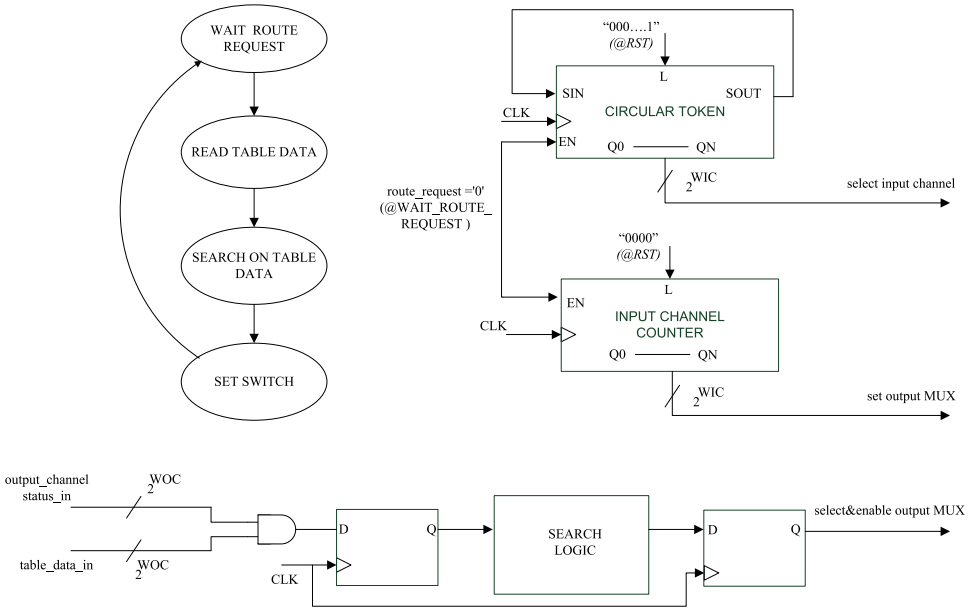


Figure 3.10. The routing FSM and additional sub-blocks of routing unit.

3.4.3. Address-Channel Match Table RAMs

Address-Channel Match Table is composed of 1-bit wide RAMs. For each output channel, there is one separate dual-port RAM and the addresses of the RAMs are network node addresses. Dual channel RAMs are used and the write and read ports are separated since table is written only by table update unit and read only by routing unit. Routing unit makes word access to RAMs such that there is only one RAM and reads all of the routing information related to a given destination address in one cycle by giving the address as index to RAM. If the value of a specific RAM field related to an output is '1', this means for the given index address, that output channel is a potential route. If the RAM field is '0', that output channel is not a suitable match for that destination address. If a hello message processing cycle ends up with a possible route, hello unit writes a '1' to the RAM field of the output port, indexed by the *source_address* of that hello message. When the result of process does not give an

acceptable route, then the RAM field remains as '0'.

3.4.4. Flexible Switch Box

The switch-box is composed of several 1-bit 2-to-1 multiplexors. In order to make the switch parametric and flexible, the following method is chosen: The data width and number of select inputs of the MUXes are parametric and chosen at synthesis time. There is one (2^{WIC})-to-1 MUX for each output channel. Data width of each MUX is "WDATA+1". WDATA width is for data traveling from input to output channel and plus 1-bit for a control signal from same input to the that output channel. For generating the switch, *generate* statements of the VHDL language is used.

3.5. Output Channels

3.5.1. Overview

The output block is composed of output channel sub-blocks and a source control unit. The number of output channels are parametric and configurable at design time. Each output channel can be directed to the downstream router or to the source side. The source controller unit monitors output channels' request lines to the source side and if there is a request from an output channel to the source node, it directs the data and control line of that output channel to the source node and does the same for the acknowledge line coming from the source side for the opposite direction. The source controller also counts each cycle for this monitoring process in order to give equal chances to each output channel. The block diagram of the output block is given in Figure 3.11 and the main sub-blocks are described in the following two sections.

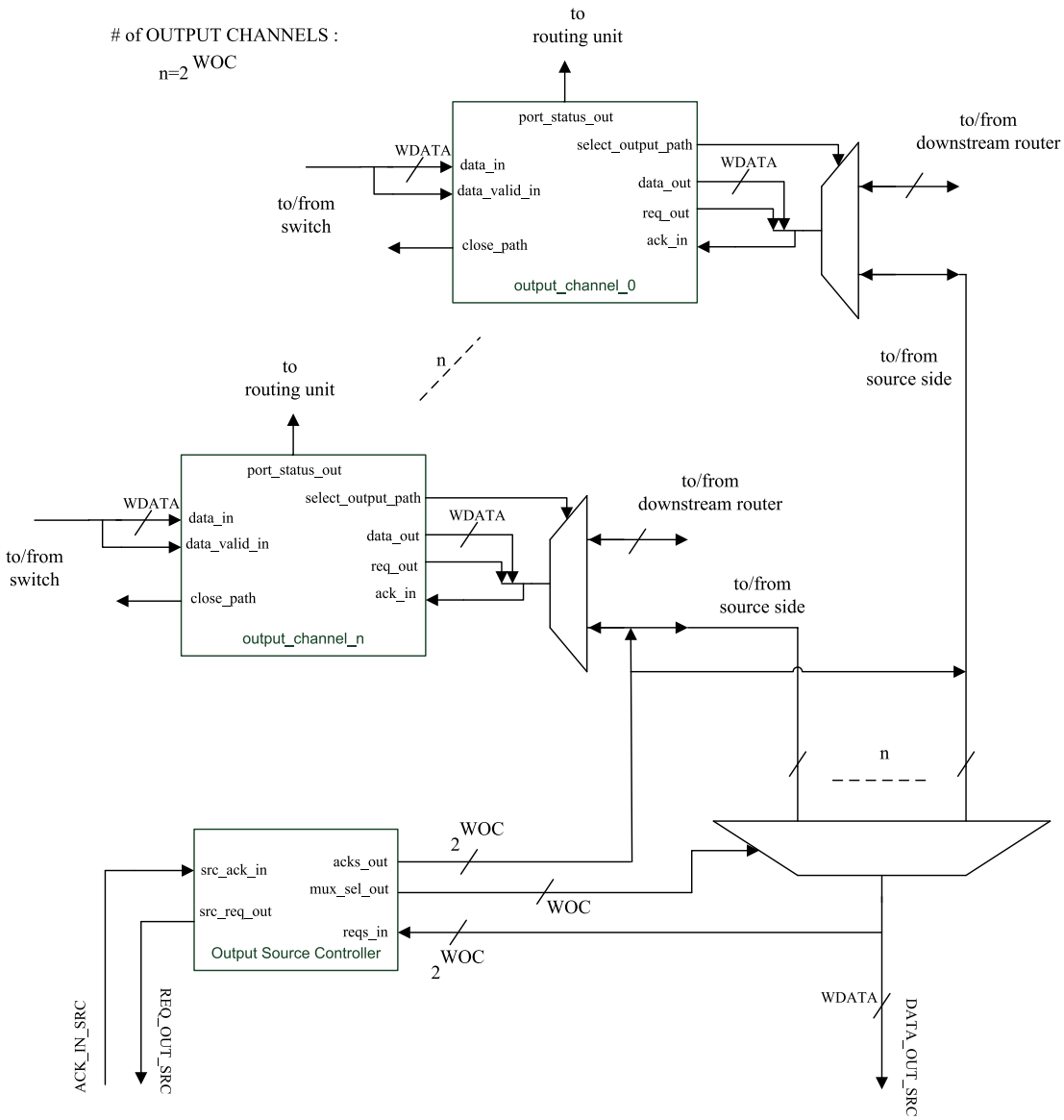


Figure 3.11. The block diagram of output block

3.5.2. Output FSM and Buffering

Each output channel has a state machine for controlling the flow at output stage and a FIFO buffer for keeping the data if the input of the downstream router is busy at that moment. After the routing unit assigns an input channel to the output via switch block, the input sends the data and a *data_valid* signal to the output. As the output channel FSM invoked with the *data_valid* signal, it begins to buffer the data and sends request to the downstream router's input channel at the same time. Receiving the acknowledge signal, it begins to send the data from the buffer to downstream router or to the source side. The decision of the output path (the consecutive router or source side) is determined at the output channel as soon as it receives the head flit which carries the destination address information of the packet. The address information of the packet is compared with the router's own address and if they are equal, the output path is set to the source side via the output de-multiplexer. If the address contained in the header flit is a different node's address, the packet forwarded to the following router.

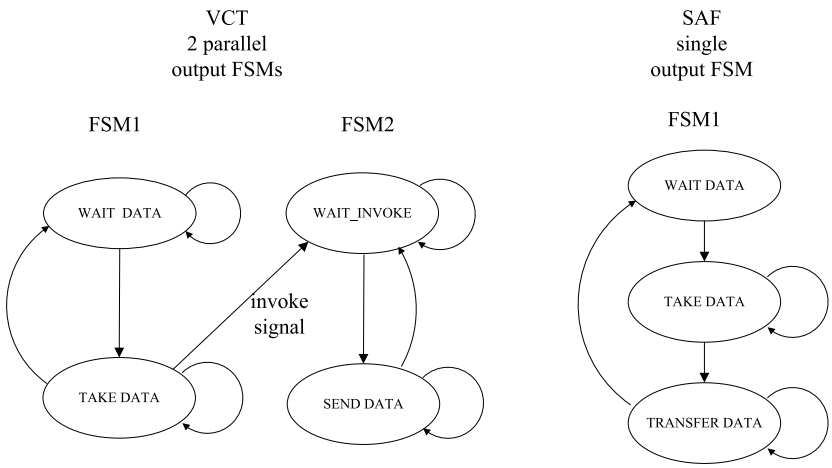


Figure 3.12. The FSM diagram of Output Channels

The output channels uses VCT or SAF flow control according to the choice at the design time by commenting out the unused method's FSM in the related source code section. Usually, VCT is used since it covers the SAF flow control. The two possible

finite state machine diagrams of each output channel is shown in Figure 3.12.

3.5.3. Output Source Control Unit

The output source controller unit monitors the requests that come from each output channel to the source side. When it recognizes an output channel is requesting data transmission to the source side, it transfers it to the source side, hires the data path to that output channel and transfers the acknowledgement of the source to that output channel.

4. ROUTING AND ADAPTATION ALGORITHM

This chapter covers the development of routing and adaptation algorithm for the AFRONOC router. First some basics about routing in general networks and well-known strategies for routing are described. Then in Section 4.1, the two possible main routing algorithms for dynamic networks are explained briefly. After the illustrative introduction, explanation of the reasons for the choices and how these algorithms covers the needs of a dynamic network on chip, the routing and adaptation algorithms are described in detail in Section 4.2.

The routing process for networks in general can be classified into two groups as dynamic routing and static routing. In static routing, the network mapping tables are generated manually before the network starts its operation and remains static. So, the nature of this type of routing is not suitable for an adaptive router, which subjects to work in dynamic NoCs.

As a consequence, dynamic routing is the point of interest for the design of the router in the scope of this thesis work. In general, it is known that, in a dynamic routing algorithm, the routing tables begin to be filled with the information as the network starts its operation. So a path discovery process flows between the members of such networks to "recognize" the environmental members and the paths between them. There are two most popular route discovery methods used for networks in global. These are link-state-routing and distance-vector routing. In the following section, these routing strategies are described briefly before explaining the development process of routing and adaptation algorithm of the router.

4.1. Possible routing Algorithms for Adaptable Routers

4.1.1. Distance Vector Routing

There are two documents about distance vector routing strategy as a standard in TCP/IP literature described in their related RFC documents [32], [33]. In distance vector routing, each router keeps a table for matching their output nodes to their neighbors and the source nodes connected to these neighbors only. So in this type of routing, the router has only the information of its neighbors and the channel to reach to that neighbor, which makes it simple to implement. After starting the operation every router sends the information of destination nodes connected to itself to its neighbors.

4.1.2. Link State Routing

The link state routing is first described in [34] and some of its improved derivatives and applications are described in detail in their related RFC documents for TCP/IP standards [35], [36]. In link state routing, each router sends the information of not only itself but also its neighbors. As a result, in each router, the information of all the network is kept. The routers send "hello" and "echo" packets in order to verify the connectivity and also estimate the distance of all other routers in the network. This process goes on periodically and so all the routers update their information about the "state" of all the "links" in the network.

These two main routing strategies are investigated from a NoC view and a simplified routing algorithm derived by the link state routing is given for NoCs in [37] where the main reason for choosing link state routing is its superiority over distance vector routing in the case of broken links. Since the link state routing is too complex and possibly will occupy much more area then distance vector implementations, the authors offered to choose a limited routing table, set it once in the beginning and do

not make periodical updates for these tables. These assumptions would be acceptable in case of ASIC implementations of NoCs as the authors of [37] declared. However, this way of simplification will take away from the dynamic nature of link state routing and may not suite for systems realized on reconfigurable systems, in which the states of nodes and even topology of connection may change during operation of the network.

So, the choice for adaptation process of the router will be a derivative of link state routing philosophy since it is suitable for dynamic network realizations and supports scalability. For the realization on reconfigurable hardware or generally in hardware, one should keep in mind that there are limited resources, so the implementation should occupy as small area as possible. The first idea for keeping area small is to make the routing tables and keeping the routing information as compact as possible. This is done by designing the address-channel match table in the way that is described in Section 3.4.3. In the following section, proposed routing and adaptation algorithms are described.

4.2. Proposed Routing and Adaptation Strategy

4.2.1. Adaptation Algorithm for AFRONOC

So we can summarize the main tasks for the adaptation algorithm designed in the inspiration of link state routing is as follows:

- Every router sends its own hello message to the neighboring routers with a given (configurable) period (HELLO_REFRESH_PERIOD)
- When receiving a hello message, router puts the message in a process and determines if this is an alternative path or not.
- After processing the hello message, router forwards it to the neighbor routers or drops the message.

The structure of a hello message used in this algorithm is given in Figure 4.1.

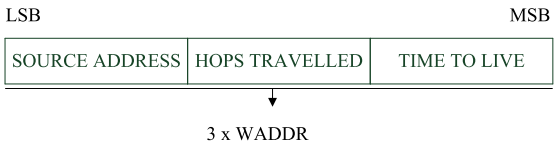


Figure 4.1. Structure of Hello Message

Here, the *source_address* field keeps the address of the router which inserts the message to the network, *time_to_live* field keeps the information of life time for a hello message and decremented in each visited router while traveling in the network. The *hops_travelled* field keeps the information of number of routers the message passed through and incremented in each router it visited. The pseudo code representation of the hello message processing, which takes place in table update block, is given in Algorithm 1 (Figure 4.2).

Here, the PORT_RAM_N[*index*] represents each of the 1-bit RAM of address-channel match table as described in Section 3.4.3. The number "N" at the end of the name indicates that the hello message is received from the router that is connected to N^{th} output channel (the same router is connected to N^{th} hello input channel also). These RAMs are filled with zero after global reset of the router (line 1). The CMHC_RAM[*index*] is abbreviation for Current Minimum Hop Count RAM and filled with ones after the reset (line 2). As the hello messages begin to flow across the router, new CMHC values are replaced with the older (and greater values) if these new ones give a shorter path than the previous ones for the related router addresses on the CMHC RAM.

Algorithm 1 Table Update Algorithm

```
PORT_RAM ← all_zeros
CMHC_RAM ← all_ones
loop
    if received_new_hello_packet = 1 then
        if source_addr ≠ my_addr then
            if hops_travelled < CMHC_RAM[source_addr] then
                CMHC_RAM[source_addr] ← hops_travelled
                PORT_RAM_N[source_addr] ← 1
            else if hops_travelled = CMHC_RAM[source_addr] then
                PORT_RAM_N[source_addr] ← 1
            end if
            if time_to_live > 0 then
                hops_travelled ← hops_travelled + 1
                time_to_live ← time_to_live − 1
                forward_hello_packet_to_all_neighbours()
            else
                drop_hello_packet()
            end if
        else if source_addr = my_addr then
            drop_hello_packet()
        end if
    end if
    if send_my_hello_counter = HELLO_REFRESH_PERIOD then
        send_my_hello_packet_to_all_neighbours()
    end if
    send_my_hello_counter ← send_my_hello_counter + 1
end loop
```

Figure 4.2. Pseudo Code of Hello Message Processing

The operation of Algoirthm 1 on the table update unit can be explained as follows: It waits for a new hello message (lines 3,4,27). When received, it looks if the origin of the hello message is itself or not(line 5). If it is so, it drops the message and looks for a new one. By this way the unnecessary insertion of the same message is prevented. Otherwise, it checks if the *hops_travelled* field of the hello message is smaller or equal to the CMHC for the router, which has sent the hello message(line 6). If smaller, it updates both the CMHC RAM and the address channel match table (lines 7,8). If the values are equal, it only updates the address channel match table related to the channel that the hello message received (lines 9,10). So it marks the channel from which the hello message received as a possible path to the router which sent that hello message. After that, it checks if the *time_to_live* value of the message has expired(line 12). If the answer is yes, it drops the hello message (lines 16,17). If not, it forwards the hello message to all neighbors after incrementing the *hops_travelled* value and decrementing the *time_to_live* value by one(lines 13-15). At the same time runs the *send_my_hello_counter* (line 26) and if the HELLO_REFRESH_PERIOD is expired, it sends its own hello to the neighbors (lines 23,24).

Here, some points have to be discussed for optimal operation of the adaptation algorithm. One point is that, the choice of the *time_to_live* value must be selected carefully according to the diameter value of the network. If it is chosen too big, there might be unnecessary traveling of the hello message which would end up with a contention on the hello messaging network. So the values in the neighborhood of the diameter value of the network will be suitable. There is another restriction about the selection of the *time_to_live* value. If it is chosen smaller than the diameter value of the network, there will be some node couples which will not be aware of each other. This problem is serious and might effect the operation of the whole system, since there may be unknown connections and addresses. So choosing the *time_to_live* value exactly the same with the diameter value of the network will fit the requirement. Another aspect is the selection of the period of inserting new hello packets to the network. If it is

chosen so small, each router will try to send its own hello message frequently to the network and this may yield contentions again. The optimization of this parameter can be done by changing the HELLO_REFRESH_PERIOD parameter. Experimentally, its default value is chosen as 32 cycles for the simulations done in the scope of this thesis.

4.2.2. Routing Algorithm for AFRONOC

The routing algorithm becomes simple with the aid of the table updating process. Since the table is filled with the possible routes for a given destination address, the only job for the routing unit is to check the table and the states of the output channels. When it finds an available channel, it sets the output switch appropriately for the input channel that sends the route request with the destination address of the data packet. The pseudo code for routing algorithm is given in the Algorithm 2 (Figure 4.3).

The pseudo code shown in Figure 4.3 is realized in the routing block as described in Section 3.4.2 and shown in Figure 3.10.The routing block waits for a *route_req_in* signal from the input channels (line 1-16). If it receives a new request(line 2), then it begins to search on the ACMT RAM (PORT_RAM_N) with the given *destination_addr* from the input channel, which sent the *route_req_in* signal (lines 3-13). It masks the route information, which comes from the ACMT RAM, with the status register, which is updated by the output channels (line 5). If the mask and search operation ends up with a available route, it sends a *route_ack_out* signal to the input channel (line 6) and sets the output switch appropriately (line 7). If it cannot find an available route for the current request (line 10), it does not change the state of the *route_ack_out* line (line 11) and continues on polling the other input channels by rotating the *rotate_input_token* (line 15).

Algorithm 2 Routing Algorithm

```
1:  loop
2:      if route_req_in = 1 then
3:          for N ← 0 to 2^WOC do
4:              if PORT_RAM_N[destination_addr] = 1 then
5:                  if status_reg_N = 1 then
6:                      route_ack_out ← 1
7:                      set_output_switch()
8:                      Break
9:                  end if
10:             else
11:                 route_ack_out ← 0
12:             end if
13:         end for
14:     end if
15:     rotate_input_token()
16: end loop
```

Figure 4.3. Pseudo Code of Routing the Data Packets

The most time consuming part is the search loop on the address channel match table (indicated by the for loop with index N in Figure 4.3)(lines 3-13). This can be done either by checking the bits of the available channels register in a cycle-by-cycle manner in multiple clock cycles or by a combinational logic in one clock cycle. The advantage of the former one is lower combinational delay which provides a higher clock frequency than the second solution for the whole routing process. In the multi-cycle solution, the number of search cycles is dependent to the number of output channels which is configurable and bounded by the number 32. So this means in the worst case, if the '1' is on the most significant bit of the searched word, (the suitable channel for that instance of routing is the 31st channel) the search process will take 32 clock

cycles which would be too long for this purpose. Moreover, the duration of the search process would be variable according to the position of the first rightmost '1' on the searched word. So, even though the single cycle solution with the combinational logic will bring a combinational latency and a bottleneck for the working frequency of the router, searching process will always take just one clock cycle, which will make the timing estimation deterministic. So the later solution is chosen for that purpose.

The hardware realizations of the routing and table updating algorithms are explained in Section 3.4 and 3.2 respectively.

5. FPGA IMPLEMENTATION and SIMULATION RESULTS

The AFRONOC router developed in VHDL, and synthesized to Xilinx FPGAs in Xilinx ISE 10.1 environment for experimental purposes. All the simulations are done using Modelsim 6.2f from Mentor Graphics. Built in features of the FPGAs are not fully utilized so that the router can also be synthesized for ASIC applications.

5.1. Implementation Results of Single Router

In this section, FPGA implementation results of the router are given. Also some design choices are described briefly. The router is mapped onto two different FPGA architectures from Xilinx. Spartan-3s5000 and Virtex-II Pro xc2vp70 are used for obtaining implementation results to compare with the related works. In Table 5.1, the place and route results for a configuration of four I/O channel, eight bit data width, four bit address width are given. In Table 5.2, the distribution of area of the sub blocks of the router are given. The I/O buffers are implemented as distributed RAMs and included in area results. The results are post place and route results and are extracted from the Xilinx ISE 10.1 place and route tool.

Table 5.1. Implementation Results for four channel Router, eight bit data width, four bit address width, I/O Buffers are eight word deep.

Parameter	Spartan-3s5000-5	Virtex-xc2vp70-7
Area(Slice)	766 (%2)	750 (%2)
Area (Equivalent Gates)	20733	30025
Operating Freq.(MHz)	128	190

Table 5.2. Distribution of the Area over the Main Sub-modules for the four channel Router on Spartan3s-5000 and Virtex-II Pro xc2vp70, eight bit data width, four bit address width, I/O Buffers are eight word deep.

Sub-module	Spartan3s-5000 (Slice/% of router area)	Virtex-xc2vp70 (Slice/% of router area)
Input Module	191/28	191/28
Output Module	189/28	190/28
Routing&Switching	115/17	114/17
Table Update Module	171/25	170/25

In Table 5.3 area and operating frequency results for different number of I/O channels and data width values are shown. The I/O buffers are not included and the results are again from Xilinx ISE 10.1 place and route tool. The router configurations are mapped onto Virtex-II Pro xc2vp70 device.

Table 5.3. Area and Operating Frequency Results for AFRONOC Router on Virtex-II Pro xc2vp70 device

I/O Channel count	Data Width	Area (Slice)	Operating Frequency (MHz)
2	8	334	230
4	8	639	190
8	8	1343	130
2	16	358	200
4	16	739	180
8	16	1585	125
2	32	422	190
4	32	923	166
8	32	2087	120

Also it is possible to set up a relation between the configuration parameters of the router and the area overhead. Below some of these relations are shown on the related figures. The results are given for Xilinx Virtex-II Pro xc2vp70. Since the CLB archtiectures of Virtex and Spartan devices are similar the area related results are not so different from each other as seen on Table 5.2.

In Figure 5.1 the relation between the allowable maximum number of nodes on the network and total area of the router is shown. We configured the router for four input and four output channels. Each I/O channel has a buffer of eight word deep and eight bit wide (the data width is eight bit). The WADDR parameter is swept from one to eight (the maximum allowable number of nodes is swept from two to 256). Here, the effect of the number of the nodes on the size of the CHMC RAM and the address-channel match table RAM can be seen. Even these RAMs are comparable with the I/O buffers in this configuration, it can be seen that the CMHC RAM and the ACMT RAM area do not make a drastic change on the total area of the router. So from the implementation view it can be said that, the compact and effective design of the tables yielded to acceptable area overhead for the router. Also the linear behavior of the change is an evidence for the scalability of AFRONOC.

In Figure 5.2, the effect of the number of I/O channels on the router area is shown. In this configuration it is assumed for a sixteen-node network, and the buffer depth is chosen as sixteen-word deep which can be also assumed as an average depth. The results are represented for three different data width eight, sixteen and 32, which would be possibly commonly used data widths for embedded applications. The number of input and output channels count is assumed as equal and swept from two to sixteen.

From Figure 5.2, it can be seen that the change of the area is in a quadratic behavior with respect to the quadratic increase in number of I/O channels. It can be said that for the channel counts between two and eight, the area overhead for the router

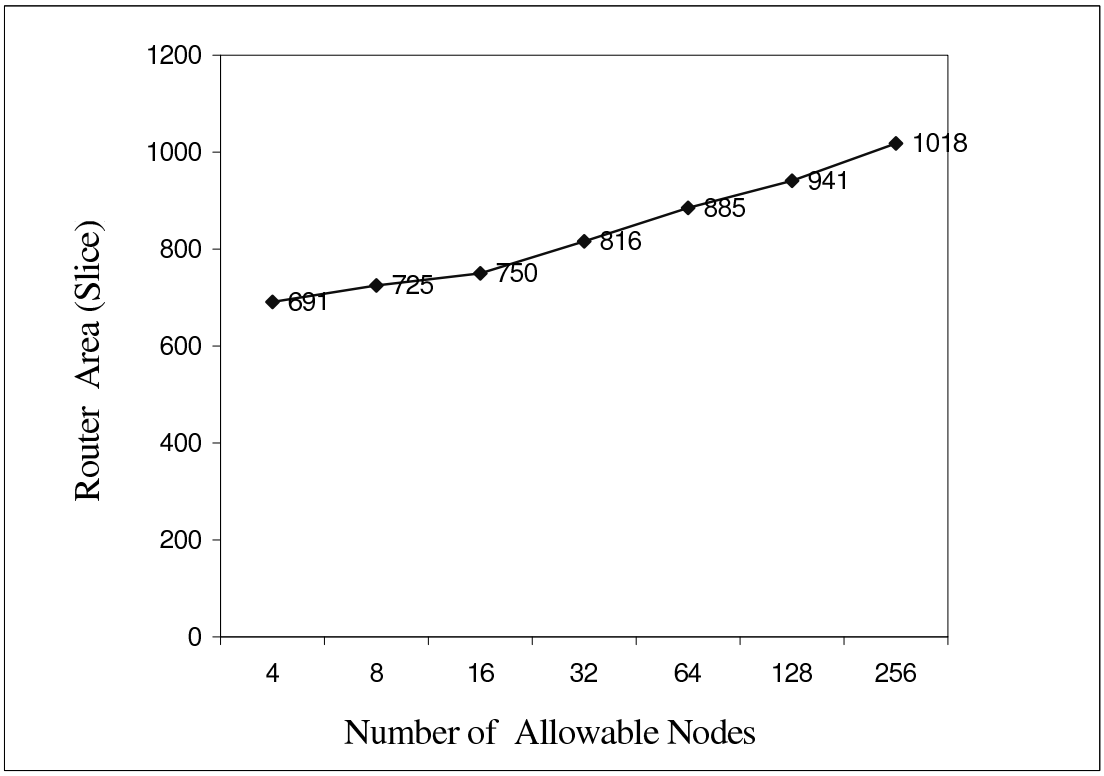


Figure 5.1. Maximum Number of Nodes vs Router Area

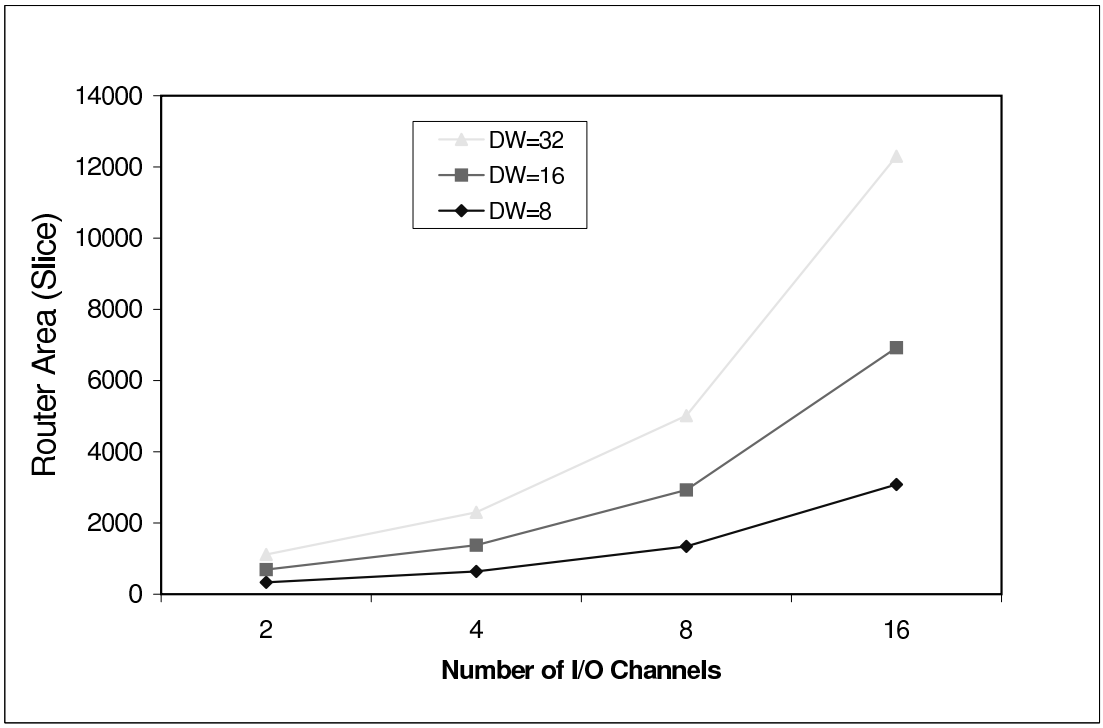


Figure 5.2. Number of I/O Channels vs Router Area

is acceptable. For the number of I/O channels over eight (which is sixteen), which would not be practical in cases such as torus, tree or mesh based or even in many cases of irregular topologies, the increase is more drastic. However since it would not find so much application area as the former I/O channel counts (between two and eight), can be negligible.

In Figure 5.3, the change of the router are versus the I/O buffer sizes can be seen. In this configuration, the maximum number of the nodes for the topology is assumed sixteen with a data width of eight. The buffer depth is swept from four to 64 which would be adequate for many applications. The results are represented for three different I/O channel count namely, two, four and eight.

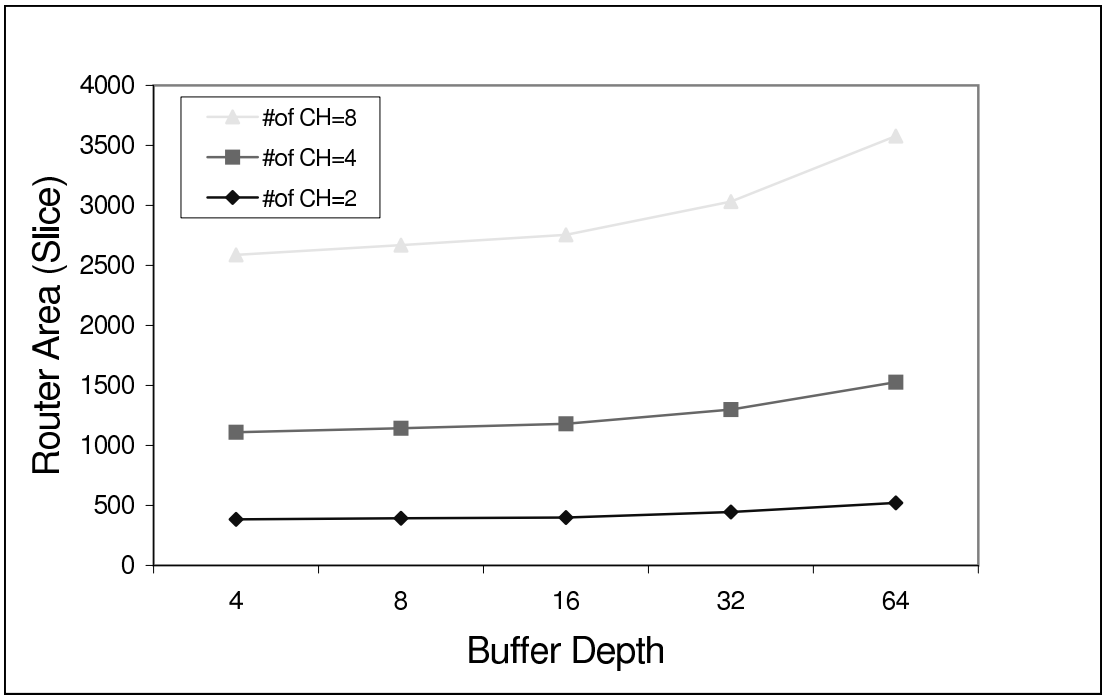


Figure 5.3. I/O Buffer Size vs Router Area

5.2. Network Generation and Implementation Results of Some Example Networks

Parameters that are necessary for generating a scalable and configurable network can be listed as follows:

- The node degree of the network: The node degree can be defined as the number of the links per node of the network.
- The diameter of the network: The greatest distance across the network from one node to another node in the network.
- And finally, the third parameter is the number of nodes.

5.2.1. Ring Network Generation

The general characteristics of the ring networks can be defined as follows: If the number of the nodes in the network is N, then the diameter of the network is given by N/2 and degree of the network is given as two. The illustration for generating parametric scalable ring networks is shown in Figure 5.4.

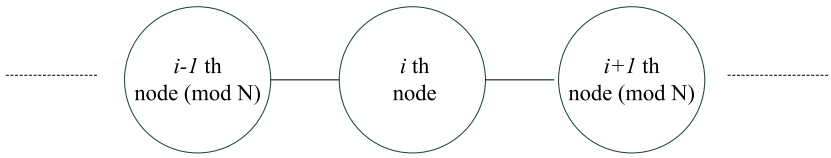


Figure 5.4. Illustration for Generating Ring Networks

5.2.2. Torus Network Generation

The torus network can be defined by number of rows and columns in the network. If the number of the rows is R and number of the columns is C, then the number of nodes in the network is RxC. The diameter of the network is max(R/2,C/2). Degree of the network is four. The illustration for generating parametric torus networks is given in Figure 5.5.

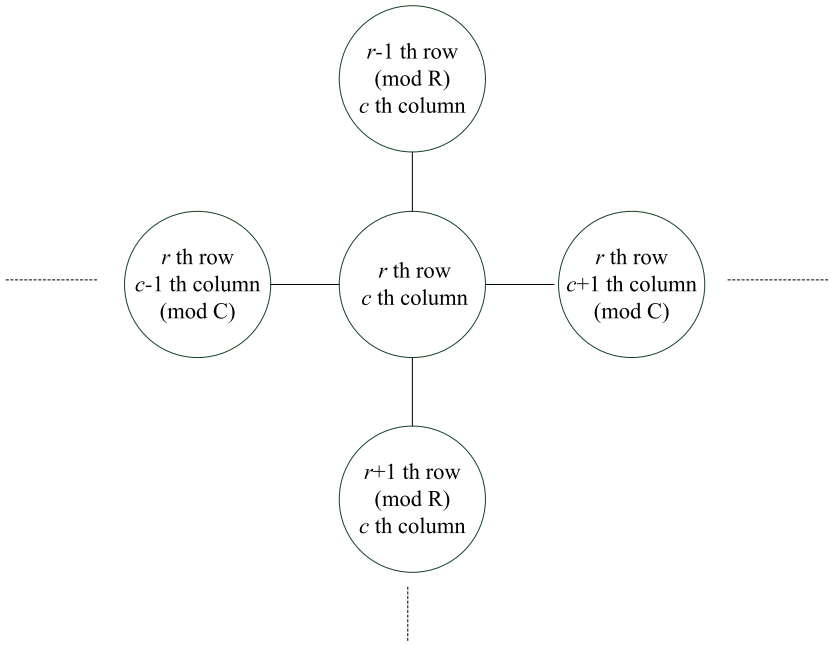


Figure 5.5. Illustration for Generating Torus Networks

5.2.3. Mesh Network Generation

The mesh network topology is similar to torus except the additional toroidal connections which extends the reachability in torus networks. Again mesh networks can be defined by number of rows and columns in the network. If the number of the rows is R and number of the columns is C, then the number of nodes in the network is RxC. The diameter of the network is (R+C-2). Degree of the network is two for the nodes at the edge of the network, three for the nodes, which are on the boundary rows and columns and four for the intermediate nodes. The illustration for generating parametric mesh networks is given in Figure 5.6.

For each example, the parameters for the network is kept in a VHDL package file and the design in VHDL is done by using the generate statements for achieving scalability.

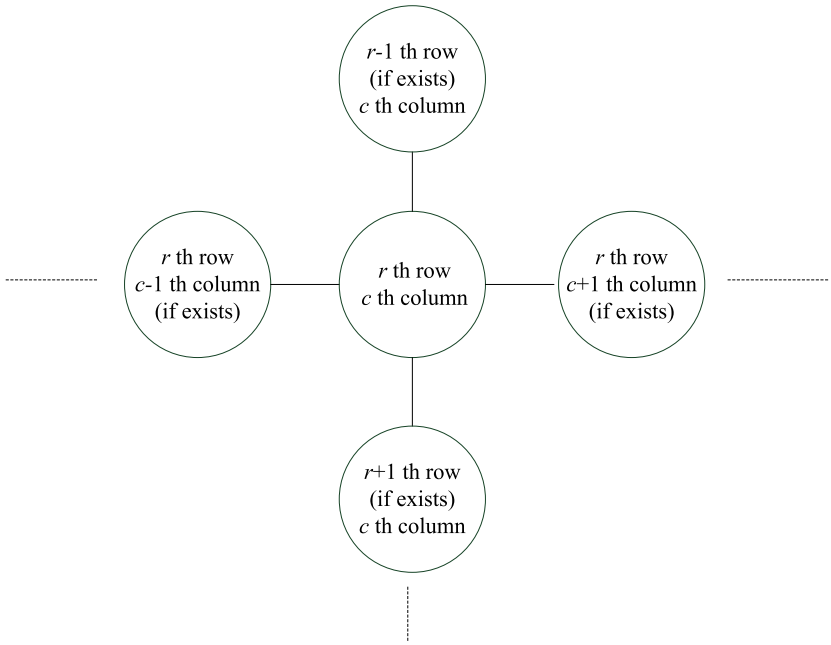


Figure 5.6. Illustration for Generating Mesh Networks

5.3. Simulation and Performance Results of an Example Network

5.3.1. Performance Parameters for NoCs

The most important evaluation parameters for NoCs are latency, throughput and area as pointed in [30], [3] and [2]. In this section some results for latency and throughput of the router for an example NoC simulation setup is described. Throughput, also equal to flit[1] injection ratio, is defined as

$$TP = \frac{(total\_message\_completed)x(message\_length)}{(number\_of\_nodes)x(total\_time)} \tag{5.1}$$

and given in flits/cycles/node in results. Latency is the total fly time of a packet or flit from its destination to source. For the simulation results latency is given in clock cycles.

[1]flit is the acronym for 'flow control digit' and is the smaller part of a data packet.

5.3.2. Simulation Setup

The simulations are done on ModelSim 6.2f suite. The test network is a 4-node grid connected network as shown in Figure 5.7. Every router in each node has 2 I/O channels, 32 word deep I/O buffers and the data width is 32-bit wide. The processing elements are represented by a simulation test-bench. There are one write and one read processes as each processing element and the packets come from the text files.

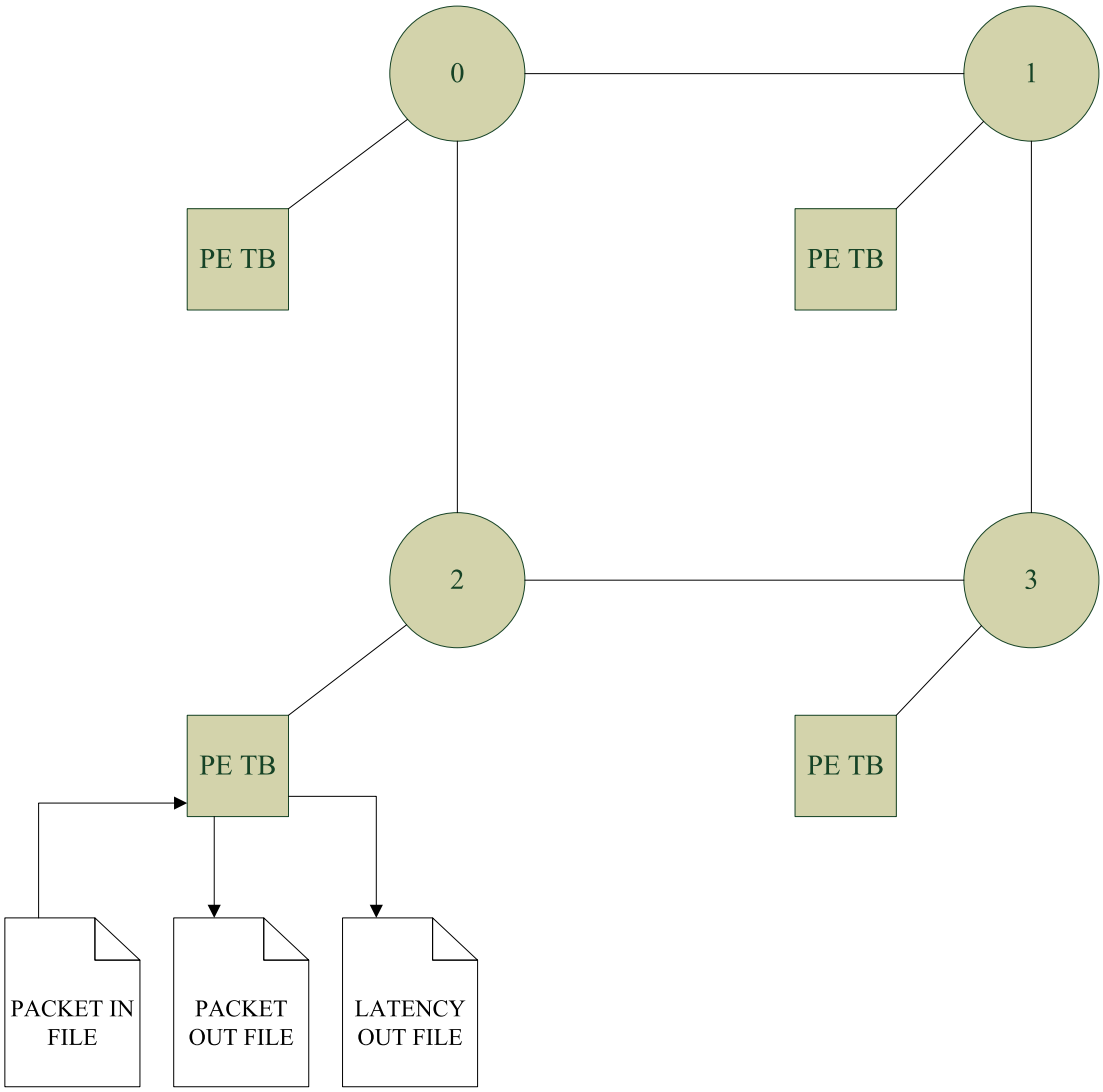


Figure 5.7. Example Simulation Setup for a Four Node NoC

Every test node behaves as pseudo processing element and inserts the packets by reading from the input file and putting some delay between each packet and marking

the insertion time of the packet in cycles and admits data from the router writes to the output file and calculates the header and full packet latency of that packet and writes to another associated file. The test packet structure is shown in Figure 5.8. Each flit in the packet is 32-bit. The header packet contains destination address for routing information and the source address possibly would be needed at the receiving node for information. The second flit carries the injection cycle of the packet and the admission cycle of the packet is written on the third flit area when receiving the packet.

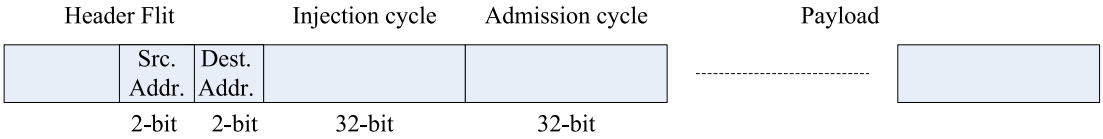


Figure 5.8. Example Packet Structure for Simulation

For traffic pattern, the bit-rotate method is chosen. In this traffic model, each node sends its packets to the another router with the address of 1-bit rotated form of its own address. For the test network, the routes are; 00->00, 01->10, 10->01, 11->11.

From the simulations on the four node test network, it is seen that minimum header latency is 10 clock cycles for the router. 31-flit long packets are used for simulation and the resulted injection ratio is 0.82 flits/cycle/node in average. For a 100MHz clock frequency this yields to 2.44 GBit/sec. The average header latency is 17 cycles and average full packet latency is 48 cycles.

A similar performance simulation test is done for 4x4 torus and 4x4 mesh networks also. Again bit rotate method is chosen as traffic load pattern. For a 100MHz clock and 32 bit data width, maximum theoretical bandwidth for each of the sixteen nodes in the network can be 400 MByte/sec [22]. According to results from experiments, the maximum bandwidth achieved for AFRONOC is 320 MByte/sec which means 80% of the maximum theoretical bandwidth. At maximum flit injection ratio, which is 0.8

flit/cycle/node, the average head flit latency is 83 clock cycles for 4x4 mesh network and 87 clock cycles for 4x4 torus network while the minimum head flit latency is 20 clock cycles for both topologies.

Also the initialization times for adaptive routers could be another criteria for evaluation. It is useful to understand the impact of the network size on the adaptation algorithm for considering the router in terms of scalability. The graph in Figure 5.9 shows the change of the initialization times with respect to the number of nodes in the network for mesh and torus topologies. The behavior of the change in Figure 5.9 is close to a linear behavior, which brings and advantage in terms of scalability.

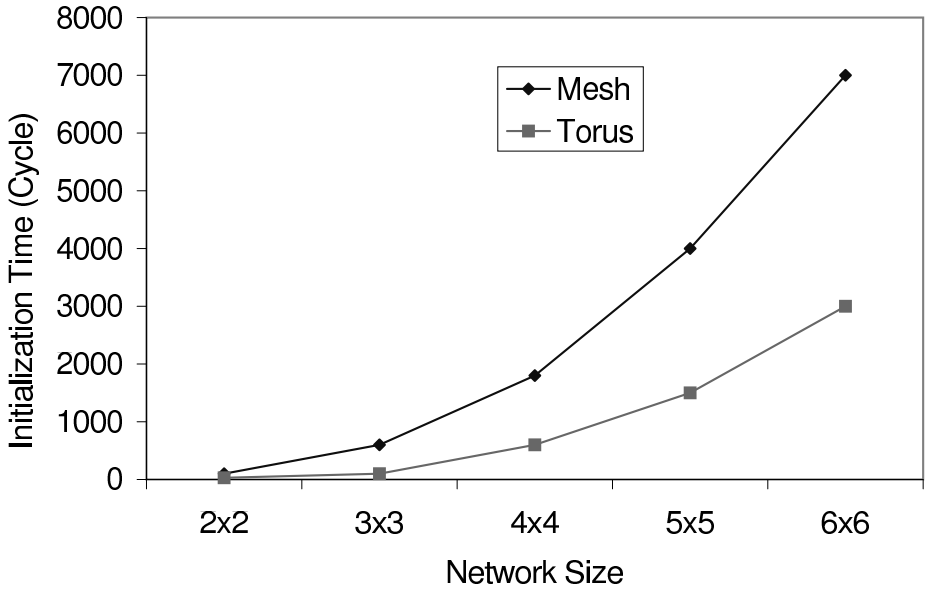


Figure 5.9. Impact of the Network Size on the Adaptation Algorithm

These initialization times are reasonable when compared to total working time of the network. For a 100MHz clock frequency, initialization time for sixteen node torus is 6 μs nearly, which can be regarded as a short time when compared to long reconfiguration times of the tables as in [22], made by an complex OS, in [21] and [38]. A direct comparison can not be made as no numeric values are specified in the related works. A waveform from Modelsim to measure initialization time of the network is

given in Figure 5.10 for four node test network.

In Figure 5.10, the CHMC and ACMT RAMs are shown. The CMHC values show the distance of the each node from the other nodes. For example for router2, $\{1-2-3-1\}$ means that the distance from router0 is 1 hop, from router1 is 2 hops and from router3 is 1 hop. The current minimum hop count value from itself remains unchanged as three and not used for routing information in routing algorithm. The ACMT RAM values are for each output channel. Again for router2 the first RAM is for output channel0 and value $\{1-1-1-0\}$ means that this channel is a suitable route for addresses 0,1 and 2 and not a good route for destination router with address 3. The time from the reset of the network and to the end of the table update process is 1665ns which yields nearly 167 clock cycles for the 100MHz clock used for the example in Figure 5.10. The initialization time values are practically shorter when compared to this example in which the whole network map is completed on the routing tables. That is, as soon as a minimum of 1 path is found for each router to the any other router in the network, the connectivity of the network is said to be established and the data routing can start after that time point. The remaining paths are alternatives and they help to shorten the latency values by providing more choice for a given routing task. For the example described above, the practical initialization time is 30 cycles which results with 300ns initialization time for 100MHz operating frequency. This practical initialization time measurement is also verified by controlling the correctness of the packet delivery after initialization of the network for any measured initialization time.

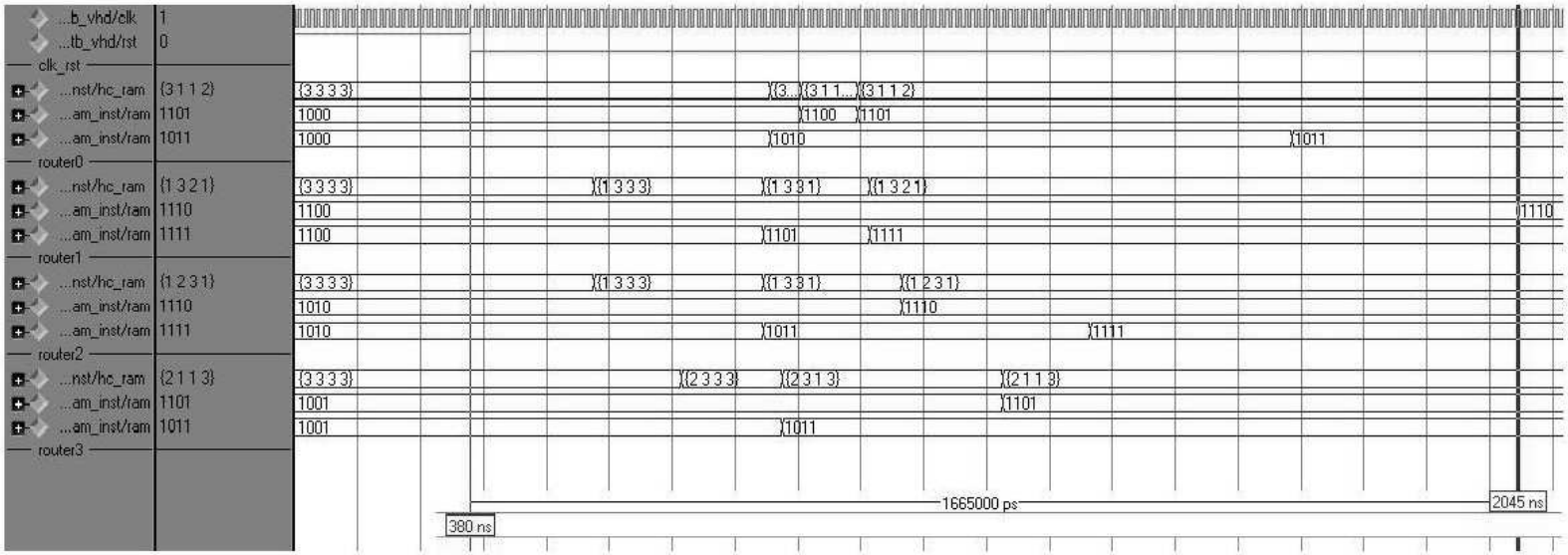


Figure 5.10. Example Waveform for Measuring Initialization Time of Networks

6. DISCUSSION AND COMPARISONS

The work on reconfigurable, adaptive and flexible router design is an emerging topic. The importance of run-time reconfigurable NoCs and related research points are mentioned in [39] where meeting the application demands in run-time is stated as the most important problem for reconfigurable NoCs, such as deciding on how to reconfigure a NoC system fully or how to configure just the routing tables according to the changing application needs in run-time. So fast and accurate adaptation of routing tables to the run-time changes becomes crucial in order to achieve correct routing tables in short time during run-time changes in the network. One of the studies done on this subject is represented in [22]. A topology adaptive router design which provides support for irregular topologies and some sample network implementations is described in [22]. The table update mechanism is achieved with the aid of a Linux based central operating system which makes the system design complex and reduces the scalability when compared to the work represented in this paper. The area occupation of a single router in [22] is comparable to AFRONOC where our distributed table update mechanism comes with nearly 25 % area overhead. The table update times is not mentioned clearly in [22], the only point mentioned in the paper is that it takes much more than a few clock cycles which is achieved in our work.

Even though flexible and adaptive NoC router design is still a hot topic, there is not so much work in literature in order to make a comparison in terms of the table update mechanisms and initialization times. To make a comparison, reconfiguration based update methods can be considered such as [40] where a self reconfiguration method is described. In order to illustrate the situation, an example can be, comparing the table update time for a 2x2 mesh topology designed in the scope of this work. By using the reconfiguration method in [40], it takes 345us for configuring the tables of the 2x2 mesh network in a Spartan3s5000 device for a 50MHz clock rate. Also there is a

need for a central logic and RAM space in order to keep the table entries and reconfigure them at same time with the method in [40]. It takes just 0.6us for initializing the tables of the same network for the same clock frequency with the method proposed for AFRONOC and there is no need for a central unit and the network topology need not be known during run-time. Reconfiguration based methods require special effort on the placement of the modules for achieving efficient reconfiguration performance. With the proposed novel stand-alone table update mechanism and parameterizable flexible architecture, AFRONOC router is suitable for fast prototyping and provides a topology-free design stage which will help to fill the design productivity gap.

7. CONCLUSIONS AND FUTURE WORK

In this thesis, a flexible and configurable router architecture is proposed for network on chip solutions, which has the important capability of adaptivity to changing network topologies. The novel adaptation and routing algorithm presented in the scope of this work is inspired from the well-known link state routing for general dynamic networks and so gives a general solution for the routing of NoCs and independency from the topology of the network. This makes the router stand-alone and independent from an outer central monitoring or controlling unit, which is offered as general solutions for such routers in literature. So with these characteristics, the designed router is suitable for both regular and irregular topologies and offers an infrastructure for reconfigurable network on chips in both run time and compile time.

As design goals, it is aimed to have as small as possible area overhead while keeping the flexibility and adaptivity of the router at a reasonable level. This is achieved by the implementation of proposed algorithms and architectures in an area-aware manner such as the design of address-channel match tables and control flow mechanisms of the router. A trade-off between the flexibility, adaptivity and area overhead is done in the choice of input-output buffering, routing and control flow mechanism. While wormhole flow control with virtual channels offers some advantages such as more immunity to deadlocks and head of line blocking problems, it comes with much more complex control schemes and hence area overhead. The avoidance from head of line blocking and deadlock problems is tried to be minimized by both input and output buffering which yields more space for keeping more messages in the network at the same time. This also brings some level of area overhead. However with the simple control flow mechanisms such as virtual cut through and store and forward, together with the implemented compact address channel match table, which keeps a compressed-like routing information, this area overhead is tried to be compensated.

By means of general network on chip evaluation parameters, the throughput rates are competitive to prior work while area occupation is in acceptable boundaries when compared to the previous similar works in literature. Latency values extracted from some simple test evaluations are also acceptable when compared to the similar works in literature.

For the future work, it is aimed to evaluate the design on some real or pseudo applications which should be realized and run on FPGAs. Moreover, these applications can be synthesized onto ASIC hardware. This will provide much more accurate results in means of area latency and throughput. Moreover it will be possible to investigate the design from the power consumption view. From the architectural view, to gain an understanding on reconfiguration and initialization times, an infrastructure that provides configuration of hard wires can be added to the router architecture for connections to the source processing nodes. By this way a comparison and study can be done on reconfiguration times of hard-wires and initialization times of tables in real applications. Another important issue is the synchronization of all nodes. For the architecture proposed, a single clock is used for simplicity however globally asynchronous locally synchronous structures also is an issue of interest in future SoC designs. So such an architecture may be adapted to present design by making each router in a separate synchronous clock domain and make the source nodes remain in their own clock domain. These issues are open for research and will be investigated as future work.

APPENDIX A: THE CONFIGURATION SETTINGS FOR THE ROUTER

The parameter configuration package file is shown in Figure A.1. The VHDL declaration for the router is shown in Figure A.2.

```
 4  package routerparams_pkg is
 5
 6     constant WIDTH_A           : natural   := 2;    -- WIDTH of node adresses in network
 7                                                     -- 2^WIDTH_A = table ram depth
 8                                                     -- "WADDR" in Table 3.1
 9
10     constant DATA_WIDTH        : natural   := 32;   -- Phsysical connection data width
11                                                     -- "WDATA" in Table 3.1
12
13     constant DEPTH_BITS        : natural   := 5;    -- Depth of input output buffers
14                                                     -- "BUFDEPTH" in Table 3.1
15
16     constant WIDTH_IN_CHN      : natural   := 1;    -- # OF input channels = 2^^WIDTH_IN_CHN-1
17                                                     -- "WIC" in Table 3.1
18
19     constant WIDTH_OUT_CHN     : natural   := 1;    -- # OF output channels = 2^^WIDTH_OUT_CHN-1
20                                                     -- "WOC" in Table 3.1
21
22     constant MYADDRESS         :  natural  := 0;    -- Adress of the router
23
24     constant HELLO_REFRESH_FREQ:  natural  := 5;    -- New hello packet insertion period.
25
26     constant TIMETOLIVE        :  natural  := 3;    -- timetolive for value for hello packets.
27
28  end routerparams_pkg;
```

Figure A.1. Configuration Parameters For the Router

```
38  use work.routerparams_pkg.all;
39  entity conf_router_top is
40     port(
41        CLK                      :  in    std_logic;
42        RST                      :  in    std_logic;
43
44        HELLO_IN                 :  in    std_logic_vector((2**WIDTH_OUT_CHN)*WIDTH_A-1 downto 0); -- from upstream
45        hello_req_in             :  in    std_logic_vector(2**WIDTH_OUT_CHN-1 downto 0);           -- router
46        hello_ack_in             :  in    std_logic_vector(2**WIDTH_IN_CHN-1 downto 0);
47
48        HELLO_OUT                :  out   std_logic_vector((2**WIDTH_IN_CHN)*WIDTH_A-1 downto 0); -- to downstream
49        hello_req_out            :  out   std_logic_vector(2**WIDTH_IN_CHN-1 downto 0);           -- router
50        hello_ack_out            :  out   std_logic_vector(2**WIDTH_OUT_CHN-1 downto 0);
51
52        request_in               :  in    std_logic_vector(2**WIDTH_IN_CHN - 1 downto 0); --from upstream router
53        IN_CHANNELS              :  in    std_logic_vector((2**WIDTH_IN_CHN)*DATA_WIDTH - 1 downto 0);
54        ack_out                  :  out   std_logic_vector(2**WIDTH_IN_CHN - 1 downto 0); -- to upstream router
55
56        ack_in                   :  in    std_logic_vector(2**WIDTH_OUT_CHN-1 downto 0);  --from downstream router
57        request_out              :  out   std_logic_vector(2**WIDTH_OUT_CHN-1 downto 0);  -- to upstream router
58        OUT_CHANNELS             :  out   std_logic_vector((2**WIDTH_OUT_CHN)*DATA_WIDTH-1 downto 0);
59
60        DATA_SRC_out             :  out   std_logic_vector(DATA_WIDTH - 1 downto 0);      -- to source
61        req_out_src              :  out   std_logic;                                      -- to source
62        ack_in_src               :  in    std_logic;                                      -- from source
63
64
65        DATA_SRC_in              :  in    std_logic_vector(DATA_WIDTH - 1 downto 0);
66        req_in_src               :  in    std_logic;
67        ack_out_src              :  out   std_logic
68  );
69
70  end conf_router_top;
```

Figure A.2. VHDL Declaration of the Router Entity

To change the related parameters, one should make the appropriate changes on the related fields.

APPENDIX B: RELATED PAPERS

In the scope of the thesis work the following paper has already been submitted for publishing some of the results:

Ö. Çoğal and A. Yurdakul, "AFRONOC: Adaptive Flexible Router Design For Ad-Hoc NoCs", DATE Conference, Dresden, Germany, Mar. 2010.

REFERENCES

1. ITRS 2003. , International technology Roadmap for Semiconductors, *International technology roadmap for semiconductors. Tech. rep.*, 2003

2. Bjerregaard, T. and S. Mahadevan, "A survey of research and practices of Network-on-chip" , *ACM Computing Surveys (CSUR)*, Vol.38, No.1, pp.1, 2006.

3. Salminen et al., "Survey of network-on-chip proposals", *White paper, OCP/IP*, March 2008.

4. ITRS 2001. , International technology Roadmap for Semiconductors, *International technology roadmap for semiconductors. Tech. rep.*, 2001

5. Jantsch, A. and H. Tenhunen, *Networks on Chip*, Kluwer Academic Publishers, Dordrecht, 2003.

6. Benini L., G. De Micheli, "Networks-on-Chips: A New SoC Paradigm", *IEEE Computer*, Vol.35, No.1, pp.70-78., Jan. 2002.

7. Hemani et al., "Network on chip: an architecture for billion transistor era",*Proceeding of the IEEE NorChip Conference*, Nov. 2000.

8. Kumar et al., "A Network on Chip Architecture and Design Methodology", *IEEE Computer Society Annual Symposium on VLSI ,ISVLSI'02*, pp.105-112, April 2002.

9. Soteriou et al., "Polaris: A system-level roadmapping toolchain for on-chip interconnection networks" *IEEE Trans. VLSI Syst.*, Vol.15, No.8, pp.855-868, Aug. 2007.

10. Pavlidis, V. F. and E. G. Friedman, "3-d topologies for networks-on-chip" *IEEE*

Trans. VLSI Syst., Vol.15, No.8, pp.1081-1090, 2007.

11. Feero, B. and P. Pande, "Performance evaluation for three-dimensional networks-on-chip", *in IEEE Computer Society ISVLSI'07*, pp.305-310, 2007.

12. Vangal et al., "An 80-tile Sub-100-W TeraFLOPS processor in 65-nm CMOS", *IEEE J. Solid-State Circuits*, Vol.43, No.1, pp.29-41, Jan. 2007.

13. Hu, J. and R. Marculescu, "DyADSmart routing for networks-on-chip", *Proceeding of DAC'04*, pp.260-263, Jun. 2004.

14. Daneshtalab, M., A. Afzali-Kusha and S. Mohammadi, "Minimizing Hot Spots in NoCs through a Dynamic Routing Algorithm based on Input and Output Selections", *International Symposium on System-on-Chip'06*, Nov. 2006.

15. Bobda, C. et al., "DyNoC: A dynamic infrastructure for communication in dynamically reconfugurable devices" *International Conference on Field Programmable Logic and Applications'05*, pp.153-158, Aug. 2005.

16. XiaoQiang, X. and J. ShiYao, "PAM-A Routing Algorithm for k-ary n-mesh", *APCC/OECC'99*, Vol.2, pp.1126-1129, 1999.

17. Liu, Z. and A.A. Chien, "Hierarchical Adaptive Routing", *Proceeding of 6th IEEE Symposium on Parallel and Distributed Processing*, pp.688-695, Oct. 1994.

18. Saldana, M. et al., "Routability of Network Topologies in FPGAs", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol.15, No.8, pp.948-951, Aug. 2007.

19. Vassiliadis, S. and I. Sourdis, "Reconfigurable Fabric Interconnects", *International Symposium on System-on-Chip'06*, Nov. 2006.

20. Pande, P. P., C. Grecu, M. Jones, A. Ivanov and R. Saleh, "Performance evaluation and design trade-offs for network-on-chip interconnect architectures", *IEEE Transactions on Computers*, Vol.54, No.8 pp.1025-1040, Aug. 2005.

21. Hilton, C. and B. Nelson, "PNoC: A flexible circuit switched NoC for FPGA based systems", *IEE Proc.Comput. Digit. Tech.*, Vol.153, No.3, pp.181-187, May 2006.

22. Bartic, T. et al., "Topology adaptive network-on-chip design and implementation", *IEE Proc. Comput. Digit. Tech.*, Vol.152, No.4, pp.467-472, Jul. 2005.

23. Chang, K. C., J.-S. Shen, and T.-F. Chen, "Evaluation and design trade-offs between circuit-switched and packet-switched NoCs for application-specific SoCs", *Proceeding of DAC'06*, pp.143-148, Jul. 2006.

24. Wolkotte P. et al., "An energy-efficient reconfigurable circuit-switched network-on-chip", *in IPDPS'05*, pp.155a, Apr. 2005.

25. Guerrier, P. and A. Greiner, "A generic architecture for on-chip packet-switched interconnections", *in DATE*, pp.250-256, Mar. 2000.

26. Tamir, Y. and Gregory L. Frazier, "High-performance multiqueue buffers for VLSI communication switches", *Proceedings of the 15th Annual International Symposium on Computer Architecture. IEEE Computer Society*, pp.343-354, 1988.

27. Hu,J., U. Y. Ogras, and R. Marculescu, "System-Level Buffer Allocation for Application-Specific Networks-on-Chip Router Design", *IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems*, Vol.25, No.12, pp.2919-2933, Dec. 2006.

28. Zimmer, H., S. Zink, T. Hollstein and M. Glesner, "Buffer-Architecture Exploration for Routers in a Hierarchical Network-on-Chip", *Proceedings of the 19th IEEE In-*

ternational Parallel and Distributed Processing Symposium, IPDPS'05, 2005.

29. Narasimhan, A., O. Kumaravelu and R. Sridhar, "An Investigation of the Impact of Network Parameters on Performance of Network-on-Chips", *48th Midwest Symposium on Circuits and Systems*, Vol.2, pp.1617-1620, Aug. 2005.

30. Salminen, E., A. Kulmala and Timo D. Hamalainen "On network-on-chip comparison",*10th Euromicro Conference on Digital System Design Architectures, Methods and Tools, DSD'07*, pp.503-510, Aug. 2007.

31. Kermani, P. and L. Kleinrock, "Virtual Cut-Through: A New Computer Communication Switching Technique" *Computer Networks*, Vol.3, pp.267-286, Sept. 1979.

32. Waitzman, D., C. Partridge and S.E. Deering, "Distance Vector Multicast Routing Protocol", *www.ietf.org/rfc/rfc1075.txt*, November 1988.

33. Perkins, C., E. Belding-Royer and S. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing", *www.ietf.org/rfc/rfc3561.txt*, July 2003.

34. McQuillan, J. M., I. Richer and E.C. Rosen, "The New Routing Algorithm for the Arpanet", *IEEE Trans. Comm.*, Vol.28, No.5, pp.711-719, May 1980.

35. Moy,J., "OSPF (Open Shorted Path First) Version 2",*www.ietf.org/rfc/rfc2328.txt*, April 1998.

36. Clausen, T. and P. Jacquet, "Optimized Link State Routing Protocol (OLSR)", *www.ietf.org/rfc/rfc3626.txt*, October 2003.

37. Ali, M., M. Welzl and S. Hellebrand, "A Dynamic Routing Mechanism for Network on chip", *23rd NORCHIP Conference*, pp.70-73, 21-22 November 2005.

38. Stensgaard, M. B. and J. Spars , "ReNoC: A Network-on-Chip Architecture

with Reconfigurable Topology", *Second ACM/IEEE International Symposium on Networks-on-Chip*, pp.55-64, 7-10 April 2008.

39. Atienza D. et al., "Network-on-chip design and synthesis outlook", *Integration, the VLSI Journal*, Vol.41, No.3, pp.340-359, 2008.

40. Bayar, S. and A. Yurdakul, "Self-reconfiguration on Spartan-III FPGAs with Compressed Partial Bitstreams via a Parallel Configuration Access Port (cpcap) core", *Proc. of Research in Microelectronics and Electronics (PRIME08)*, pp.137-140, Jun./Apr. 2008.