# Improving the Performance of a Wormhole Router and Wormhole Flow Control

Ming Liu

Supervisor: Zhonghai Lu
Examiner: Axel Jantsch

Master of Science Thesis
Stockholm, Sweden
IMIT/LECS-2005-80

December 8, 2005

# Abstract

As an emerging and advanced technology, Network-on-Chip (NoC) may become the alternative of the traditional bus-based System-on-Chip (SoC). In an interconnection network structure, interconnected routers are the core of the whole system and the network's performance mainly depends on their performance. There are many significant factors which determine the working mechanism of a router and its performance, such as topology, switching strategy, routing algorithm, and flow control mechanism. The research topic of this thesis, wormhole flow control, is an excellent flow control mechanism and widely used in interconnection networks.

One purpose of this thesis is to enhance the performance of previous wormhole virtual-channel router implementations. The improvement applies to different flit-admission and flit-ejection models. Synthesis results show that the speed of the previous wormhole router is increased from 76 MHz to 200 MHz, and the number of gate counts is reduced by 13.5%.

The other purpose of the project is to enhance the performance of wormhole virtual-channel router in general. Traditional wormhole routers arbitrate link usage flit-by-flit. By logically partitioning the flits of packets into groups, we propose a new flow control mechanism which arbitrate link usage group-by-group. This method makes efficient use of buffers, leading to reduced latency and improved network throughput. Its feasibility is validated not only from theoretical analysis, but also from a great deal of simulation results. In addition, synthesis results regarding speed and area show minimal implementation overhead.

# Sammanfattning

Network-on-Chip (NoC), en avancerad teknologi på framfart som kan komma att bli ett alternativ till dagens traditionella bussystemsbaserade System-on-Chip (SoC). I en kommunikations nätvärk struktur är kommunicevarde switchar kärnan av hela systemet och nätverkets prestanda beror främst på deras prestanda. Det finns många viktiga faktorer som avgör en switchar arbetsmekanism och dess prestanda; som topologi, switch strategi, routing algoritm och flödes kontroll mekanism.Detta examensarbete tar upp wormhole flödes kontroll som är en utmärkt flödes kontroll mekanism och används ofta i kommunikations nätvärk.

En anledning till detta examensarbete är att förbättra prestanda av tidigare wormhole victuell-kanal switch implementeringar. Förbättringen tillämpar olika flit-admission och flit-ejection modeller. Syntes resultaten visar att hastigheten av tidigare wormhole switch har ökat från 76MHz till 200MHz och antalet grindar har minskat med 13.5%.

Den andra anledningen är att generellt förbättra prestanda av victuell-kanal switch. Traditionella wormhole routrar tilldelas användning av länk flit-för-flit. Genom att partitionera flits från paket i grupper föreslår vi en ny flödes kontroll mekanism som tilldelar användning av länk grupp-för-grupp. Denna metod utnyttjar buffrarna effektivt vilket leder till minskad fördröjning och förbättrad genomsläpplighet av nätverket. Metoden är bekräftad både med teoretisk analys och med simulerings resultat. Utöver detta visar även syntes resultaten av hastighet och area en minimal implementerings overhead.

# Acknowledgement

This Master Thesis project has been accomplished at the Laboratory of Electronics and Computer Systems (LECS) in Department of Microelectronics and Information Technology (IMIT) of Royal Institute of Technology (KTH) in Stockholm, Sweden. Many persons have contributed to the accomplishment of this thesis, directly or indirectly.

First of all, I would like to give my great appreciation to my supervisor, Mr. Zhonghai Lu, because of his instruction and support in each subsection of the project. He is such a knowledgable and experienced scholar that I learned so much from him. Next, I want to thank my examiner, Prof. Axel Jantsch, for his opinion and help to this project. Also, thanks for the first hand documents of previous work and great help from Mr. Bei Yin. Your experience speeded up my project greatly. Finally, I want to say thanks for my parents far away in China. Your constant encourage and support gave me great courage to overcome difficulties in all aspects.

In addition, thanks for all my friends in Stockholm. I will never forget these happy days shared with you.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Background

## 1.1 Network-on-Chip

Nowadays System-on-Chip (SoC) technology has matured and been widely used in many areas. In SoC technology buses and point-to-point connections are two main means to connect some components for communicating. Point-to-point connections are not efficient because of their extremely complex structure when many components want to communicate with each other. Buses are attractive because they provide high performance interconnections while they can still be shared by several communication partners [2]. So for a long time buses have been the primary structures when a digital system is integrated on a single silicon die. While as the development of silicon technology and increasing of system complexity, many problems related to buses have appeared and need to be solved. First, it is not possible for buses to scale to a large system which includes too many components. Normally buses can efficiently connect 3-10 communication partners [2]. Too many components will degrade the efficiency of communication because they share the same bus to communicate. Second it is not easy to predict the behavior of an individual component because of the sharing of buses with others. Also from the physics of deep submicron technology, long, global wires and buses become undesirable due to their low and unpredictable performance, high power consumption and noise phenomenon [2]. So long wires in layout should be avoided while buses always mean long wires. Moreover every system has a different communication structure and it make reuse difficult [2].

Fortunately networks are good alternatives of buses which could conquer the shortcomings above and give out better performance. Figure 1.1 shows a typical structure of network. In this figure, R represents 'Resource' and S represents 'Switch'. Adjacent nodes are connected with each other by

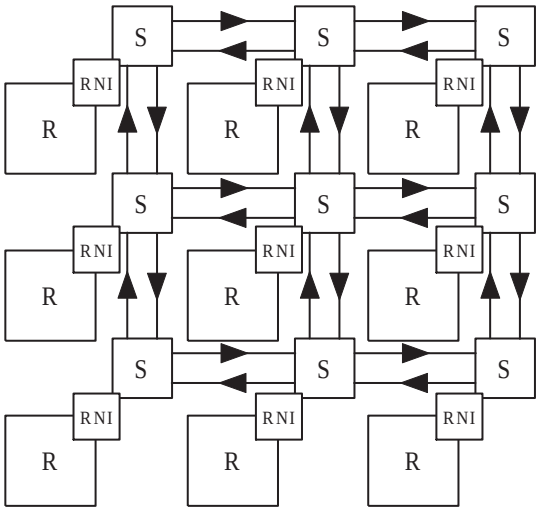


Figure 1.1: A typical 2D mesh structure of network.

switches or routers. Resources communicate with the network through an interface called 'Resource Network Interface' (RNI). In the network, resources can be any kind of IP such as processor core, DSP core, memory, FPGA or ASIC, etc.. Each resource is connected to a switch or router and transmits data with other resources through routers and channels.

It is easy for a network to include some tens or even more components because of no common buses to be shared to communicate with each other. Instead, one router is assigned to a corresponding resource to transmit and receive data. Because of this structural characteristic networks can be scaled with new resources easily. Also on the layout level we can avoid long interconnection wires which may generate many serious problems such as high power consumption and noise, etc.. There are still many advantages with the network structures built on chip and then many research groups have dedicated to this new subject in recently years.

## 1.2 Wormhole and Virtual Channel Flow Control

### 1.2.1 Wormhole Flow Control

Flow control determines how the resources of a network such as channel bandwidth and buffer capacity are allocated to packets traversing a network.

A good flow control method allocates these sources in an efficient manner so the network achieves a high fraction of its ideal bandwidth and delivers packets with low, predictable latency [3].

Wormhole flow control is a typical flit-buffered flow control which allocates channel bandwidth and buffers at a flit level. In this mechanism, packets are divided into flits which are basic units in the process of advancing from source to destination node, as shown in figure 1.2. Wormhole flow

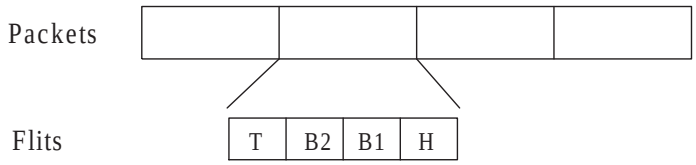


Figure 1.2: Packets are divided into flits.

control operates in the condition that channels and buffers are allocated to flits rather than packets. When the head flits arrive at a node, they must acquire three resources before they can be forwarded to the next hop along a route: a virtual channel for the packet, one flit buffer and bandwidth corresponding to one flit. Body flits then use the virtual channel acquired by the head flit and have to acquire one flit buffer and bandwidth to route. Tail flits behave like body flits, but release the virtual channel allocated to the packet. These flits of the message are transmitted through the network in a pipelined fashion. The main advantage derives from the pipelined message flow, since transmission latency is insensitive to the distance between the source and the destination [4]. Moreover, because of the buffering at flit level, buffers are well utilized and much chip area is saved when integrated. The process of a packet (4 flits) going through a typical wormhole router is shown as figure 1.3 [3]. Referring to the figure, in (a) the header arrives at the node, while the virtual channel is in the idle state (I) and the desired upper (U) output channel is busy — allocated to the lower (L) input. In (b) the header is buffered and the virtual channel is in the waiting state (W), while the first body flit arrives. In (c) the header and first body flit are buffered, while the virtual channel is still in the waiting state. In this state, which persists for two cycles, the input channel is blocked. The second body flit cannot be transmitted, since it cannot acquire a flit buffer. In (d) the output virtual channel becomes available and allocated to this packet. the state moves to active (A) and the head is transmitted to the next node. The body flits follow in (e) and (f). in (g) the tail flit is transmitted and frees the virtual channel, returning it to the idle state.

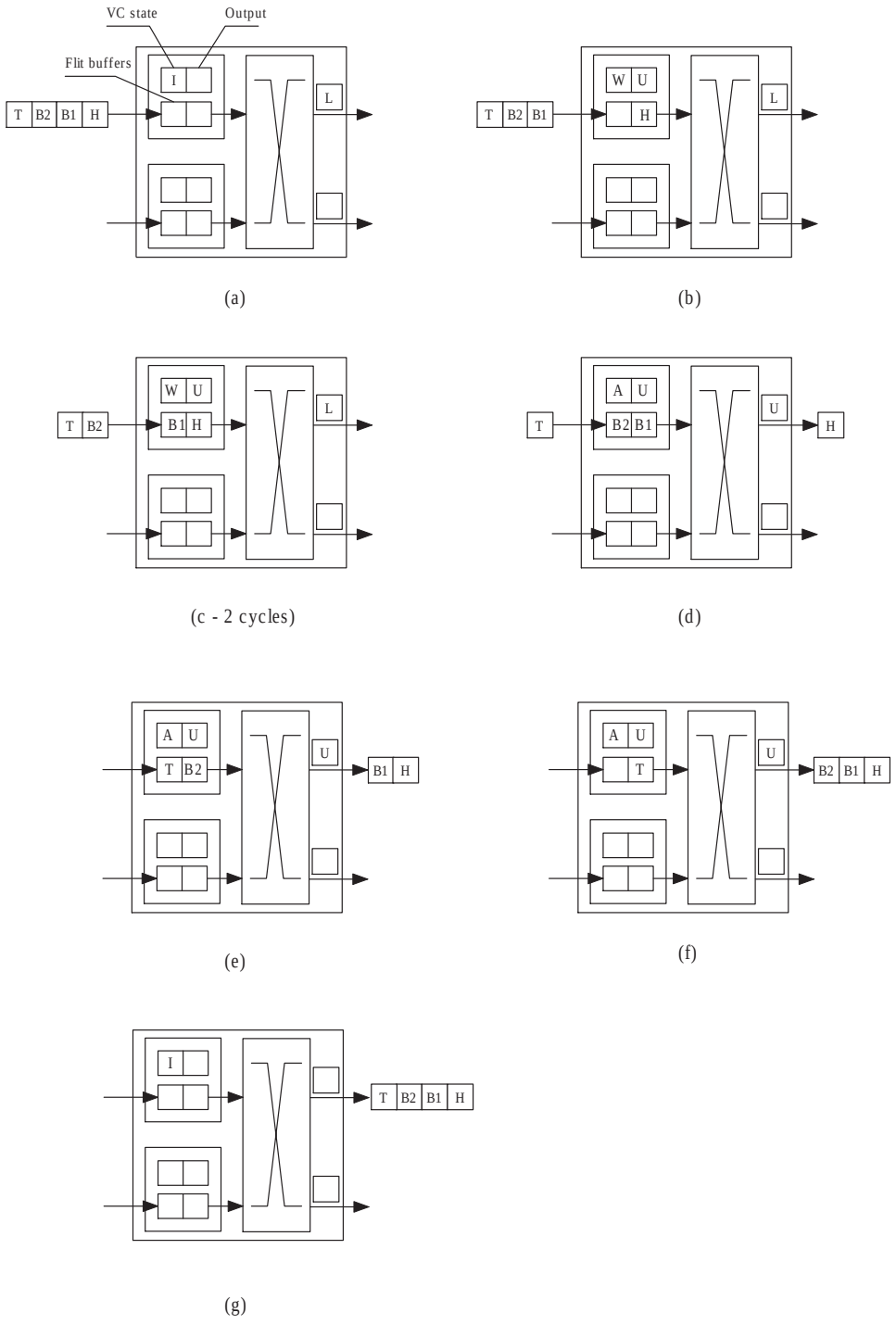


Figure 1.3: A 4-flit packet going through a wormhole router from the upper input port to the upper output port.

The distribution of flits in wormhole flow control is shown as figure 1.4.

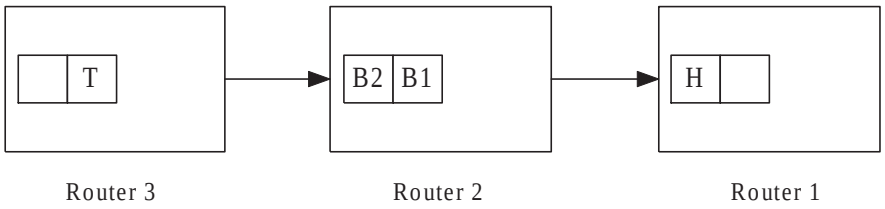


Figure 1.4: Distribution of flits in wormhole flow control.

### 1.2.2 Virtual Channel Flow Control

Virtual channel flow control, which associates several virtual channels (channel state and flit buffers) with a single physical channel, overcomes the blocking problems of wormhole flow control by allowing other packets to use the channel bandwidth that would otherwise be left idle when a packet blocks [3]. Its work mechanism is stated as below.

From the figure 1.5 (a), we can recognize that with wormhole flow control, when a packet B is blocked, channel p and q are left idle and cannot be used by A because of the unique virtual channel associated with channel p and allocated to packet B. This is a large waste of the channel bandwidth resource which is very precious in the network. To solve this problem, virtual channel flow control is introduced and multiple virtual channels are associated with a same physical channel. Referring to the figure 1.5 (b), with virtual channel flow control, packet A is able to proceed over channels p and q by using a second virtual channel associated with channel p when packet B is blocked.

Due to the advantage of using multiple virtual channels to solve the blocking problem, wormhole flow control is normally used with multiple virtual channels, combined with virtual channel flow control. A wormhole router with only one virtual channel in each physical channel is not practical and seldom used. So in this report, without special mentioning, the term of 'wormhole' should be referred as the combination of wormhole and multiple virtual channels.

## 1.3 Project Objective

Interconnected routers compose the basic structure of a network and they are the core of network. To a large degree, the performance of the network

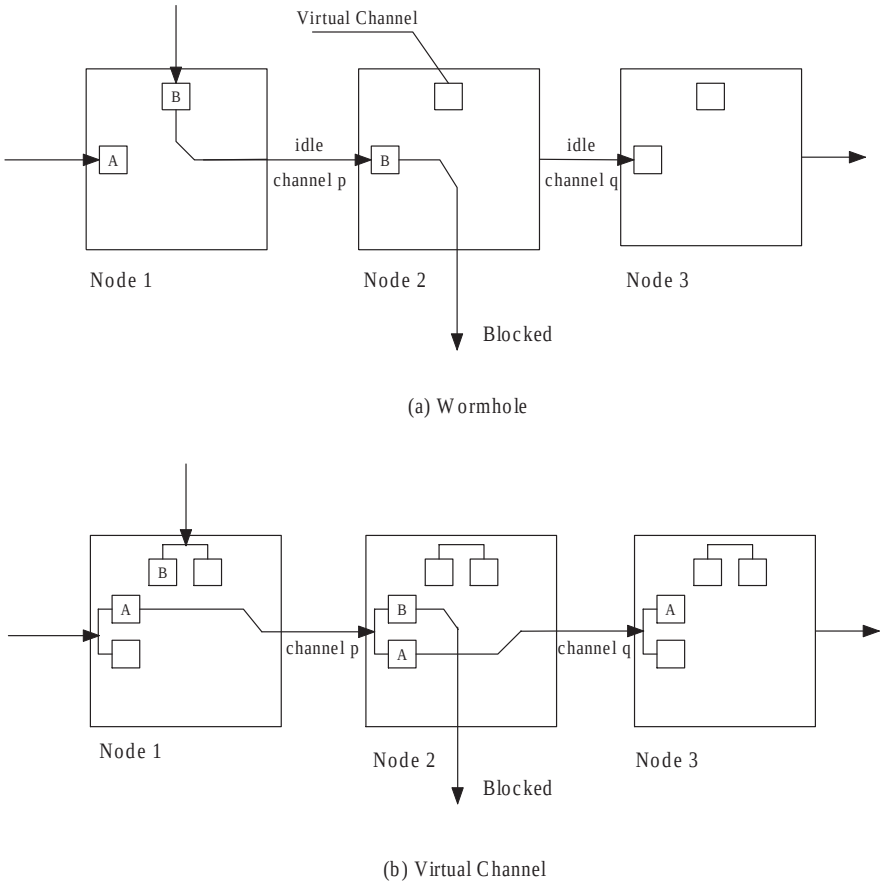


Figure 1.5: Comparison of channel bandwidth use with wormhole and virtual channel flow control.

depends on the performance of routers. And flow control mechanism is one of the most significant factors that determine the router's performance. So choosing a reasonable flow control method and designing a high performance router are key points for realizing a successful network.

Based on the previous work, a *decoupled-psink* wormhole router model should be enhanced to a practical maximum operation speed without much additional area consumption. Also, the other three models, *coupled-psink*, *decoupled-ideal*, *coupled-ideal* models should be implemented in hardware at the updated high circuit speed. Area consumption and timing reports could be recorded for comparison after synthesis.

Although an excellent mechanism and widely used in the interconnection networks, wormhole flow control still has some disadvantages which have

adverse effect on the performance of routers. In this thesis, its working mechanism should be deeply investigated and new flow control methods are expected to solve these problems. Large amount of experiments are also needed to be performed for revealing the improvement of new mechanisms and the tradeoff of hardware implementation.

## 1.4 Structure of this thesis

- Chapter 1 Background knowledge introduction. The concept of Network-on-Chip (NoC) is first introduced. Also the scheme under discussion in this thesis project, wormhole and virtual channel flow control mechanisms are explained roughly. Finally, problems to be solved are listed in the end of this chapter.

- Chapter 2 The original model of previous work is enhanced to a much higher operation speed. Synthesis results are listed to present the area and timing information when implemented in hardware. Also other three wormhole router models are implemented in hardware at the modified circuit speed.

- Chapter 3 Deep investigation and analysis to the canonical wormhole flow control reveal the problems that worsen the router's performance. Then new mechanism is introduced and validation executed not only from the theoretical analysis, but also the practical experiment. Finally hardware implementation results of the new flow control router model are presented compared with the canonical wormhole model.

- Chapter 4 Summary for the whole thesis work. Conclusions are drawn and future work is discussed in this chapter.

# Chapter 2

# Performance Improvement of a Wormhole Router

## 2.1 Four Models of Wormhole Routers

For a wormhole router, there are two admission models: decoupled admission and coupled admission [5]. Two ejection models: ideal sink and p-sink [6]. From the admission aspect, if the packet in one admission lane could be routed to any output physical channel, it is called a decoupled admission model. In contrast, if the packet could only be routed to a specific direction or output physical channel, it is called coupled admission. These two admission models are shown in figure 2.1 and 2.2.

In the packet ejection part, the model in which each virtual channel at the input channels has its own sink lane, is called ideal sink model, shown in figure 2.3. In this model, if there are $p$ physical channels and each has $v$ lanes, the total number of sink lanes needed is $p * v$. If all virtual channels sharing a same input physical channel use one lane to sink and only $p$ sink lanes are needed in total, this is the model called p-sink. Figure 2.4 shows the structure of this type. It's easy to see that p-sink model could save many buffers compared with the ideal sink model, from $p * v$ to $p$.

According to different combinations of admission and ejection methods, there exist four models of wormhole routers in total: *decoupled-psink*, *coupled-psink*, *decoupled-ideal*, and *coupled-ideal*.

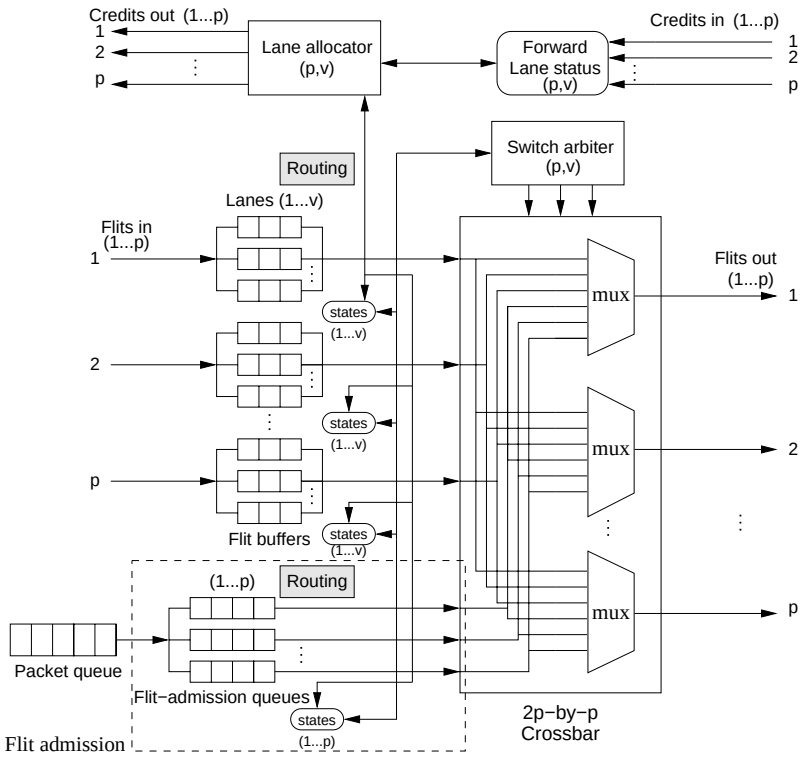


Figure 2.1: Structure of the decoupled admission model.

## 2.2 Investigation of the Previous Implementation

A canonical *decoupled-psink* wormhole router model has been implemented in VHDL by Mr. Bei Yin. For easy integration in NoC, dimension-order routing, wormhole with multiple virtual channels flow control, Round-Robin allocation and arbitration mechanisms are adopted in his design. Detailed information could be referred to in his thesis [4].

Although realized, the performance of hardware implementation of his design is not so satisfied for practical use due to a low hardware operation speed. As shown in the report [4], a *decoupled-psink* router could run at the data clock frequency of 76 MHz at most when optimized for timing by Synopsys DC (The technology is 180nm and the library is UMC18.). Also, consumed gate numbers are listed there.

Before commence on the modification of the original design, large amount of time was devoted into investigating the program code and analyzing the synthesized results of each module in the design. First, each module of the

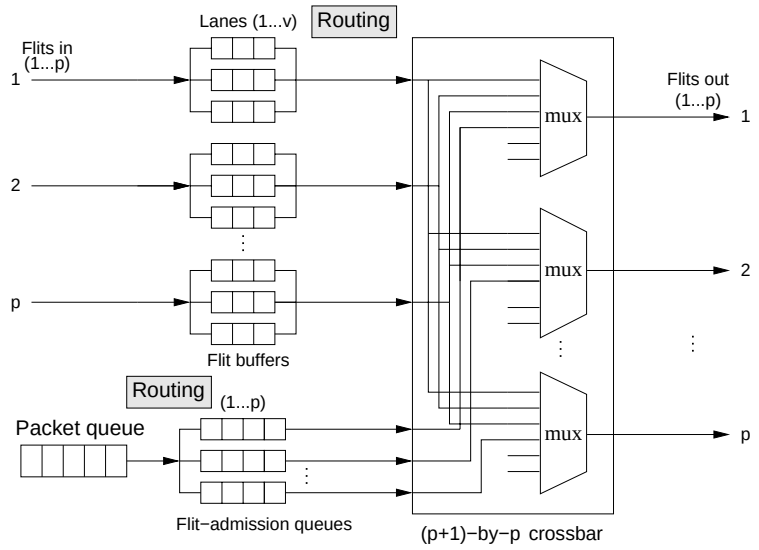


Figure 2.2: Structure of the coupled admission model.

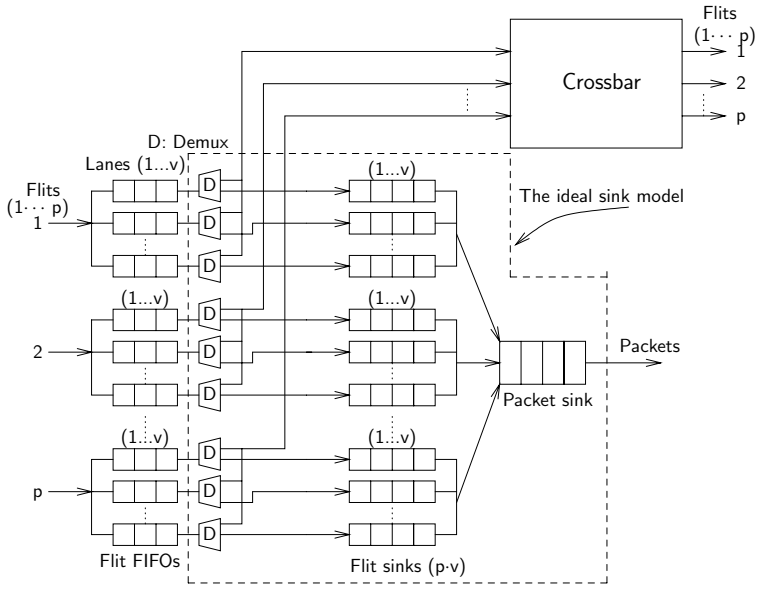


Figure 2.3: Structure of the ideal sink model.

design was synthesized and optimized for timing separately to get a rough overview of the whole design to be improved. When all results were collected, a fact was revealed that three modules contained large logic blocks between registers and generated long critical pathes. These modules are: *arbiter*,

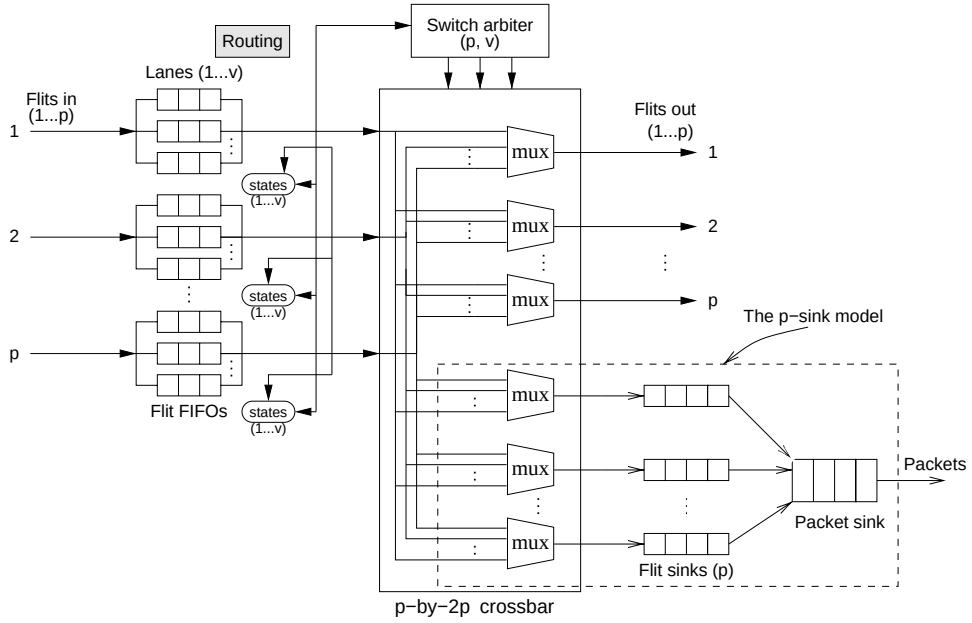


Figure 2.4: Structure of the p-sink model.

*allo_next*, and *allo_sink*. (There is a figure in Appendix A to show the code structure of the design.) And the critical pathes generated by them may become the bottleneck of the whole design, restricting the maximum operation speed. So to improve the speed of the wormhole router, large logic blocks in these three modules must be simplified or separated.

There are different mechanisms to realize the arbitration and allocation operations [3]. In the original design of Bei's, a 'perfect' mechanism is used during the process of allocating resources to requesters. (In fact, arbitration is also a process of allocating switching to requesters in virtual channels.) We call it 'perfect' because it is a maximum matching, allocating the maximum possible number of resources to requesters [3]. In theory, this is surely the most efficient allocation method that will lead to a good network performance without any waste. This method is realized in VHDL by complex nested 'loop' and 'if' structures: If there is an available resource needed by a higher priority requester, it will be allocated to this requester and a flag will be set representing that it will never be available for other requesters until released. Then for a lower priority requester's turn in the same clock cycle, the requester may also need this resource while it is not available. So only other left resources which are still available could be allocated to the lower priority requester. Thus from the highest priority requester to the lowest

one, matching operation is executed by a priority sequence within a single clock cycle and maximum allocating is done for all requesters.

However, it is just this 'perfect' mechanism and the software-like coding style that introduce long combinational logic into hardware implementation and generate critical path: For a certain priority requester, the allocation result is not only dependent on the request signals and the status signals of the resources, but also the allocation results from the higher priority requesters. If the higher priority requesters have been engaged in some resources, these resources are set unavailable. Signals representing their unavailability will take part in the allocation process of the lower priority requester as late arriving signals. Each allocation process is realized by some logic structure and the dependency of each allocated result among all requesters generates a long logic chain between registers and thus critical path, as shown in figure 2.5.

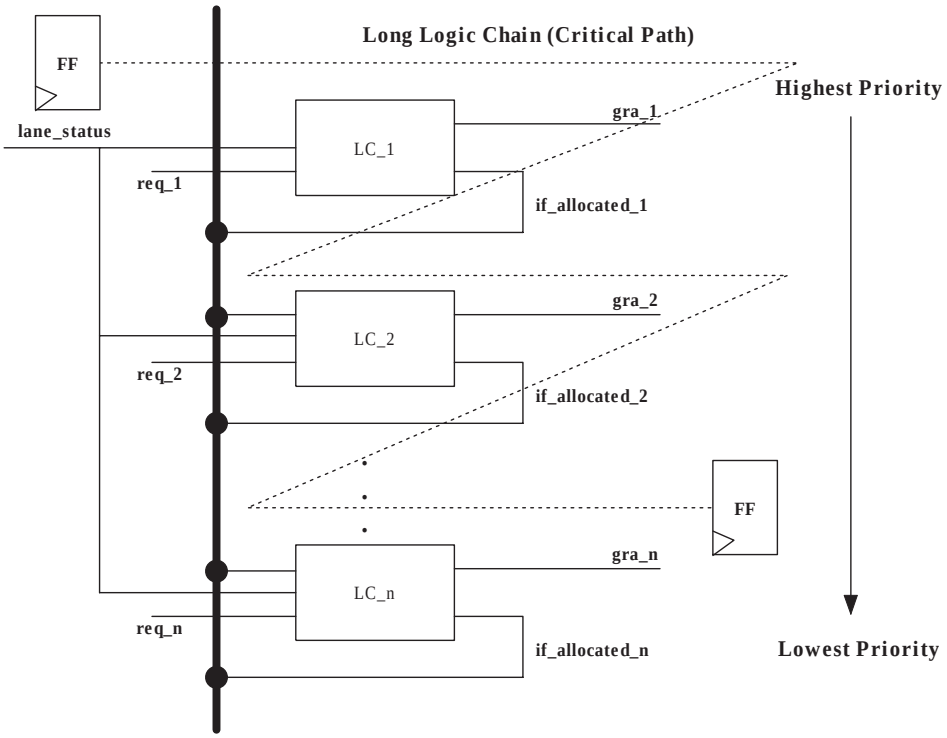


Figure 2.5: Long logic chain generated in the original design.

## 2.3 An Enhanced Implementation

To shorten the critical pathes of the three modules: *arbiter*, *allo_next*, and *allo_sink*, long logic chains must be broken into pieces or the allocation processes executed in parallel and independently for each requester. From this point, it is not easy to realize the original maximum allocation method with short logic path due to its complex algorithm.

A simple mechanism is then considered to substitute the complex one. Let's talk about the *arbiter* module first. Instead of the direct matching between the requesters in each lane and the requested link resources, two levels of arbitrations are adopted to make the allocation in parallel [7][8]. As shown in figure 2.6, requesters from the lanes sharing a same input physical channel of *crossbar* will arbitrate for the link bandwidth resource to switch. Only at most one lane could be chosen to use the link. This is the first level arbitration, shown as *arb1* in the figure. For the requesters from the admission channels, there is no *arb1* because each admission lane doesn't share a link with others. After *arb1*, the winners will be combined with their corresponding signals, *routing_result* and *sink_allo_result* to generate request signals for the second level arbitration. If they are routed to a certain output channel or sink channel, the corresponding signal will be set high to show the request. Then the second level arbitration for each output or sink channel is executed among the requesters of the winners from different *crossbar* inputs. This is a reasonable allocation mechanism because there is only one lane winner for each input channel and output or sink channel of *crossbar*. No confliction will happen.

Similarly, *allo_next* and *allo_sink* modules are also realized by two-level arbitrations for allocating the next or sink lanes to requesters. Their structures are shown in figure 2.7 and 2.8. In these allocation modules, the first arbitration is adopted to limit the requests for all the next possible lanes belonging to a same physical channel or all sink lanes to only one. The second arbitration then relate one next or sink lane to only one requester.

How about the performance or efficiency of the two-level arbitration allocation mechanism compared with the direct maximum matching method used in the original design? Will this new method worsen the system performance a lot? The simulation results could answer this question. Some parameters of the network for simulation are:

- Mesh size: 4 x 4
- Number of VCs per PC: 4
- Number of admission VCs: 4

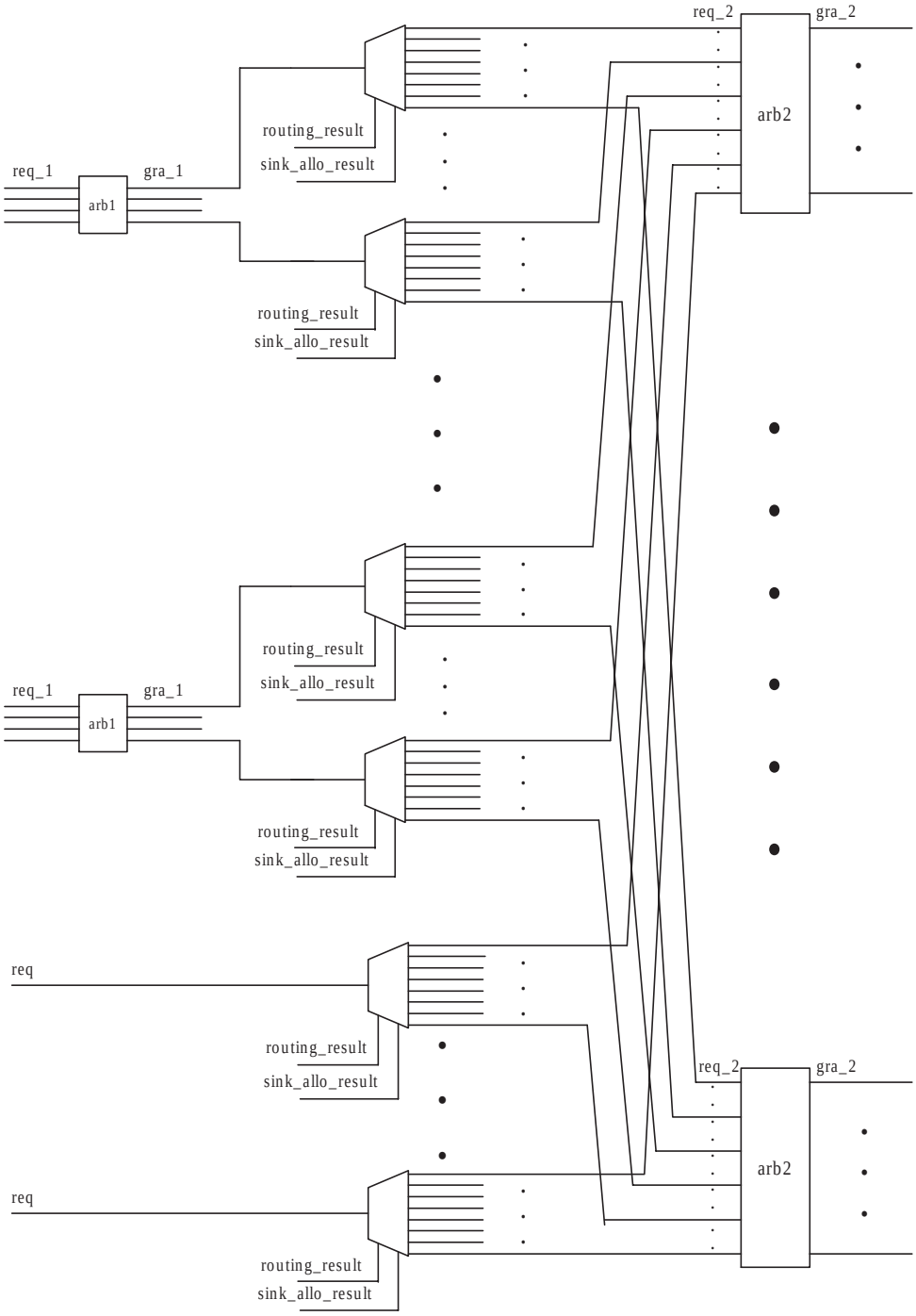


Figure 2.6: Structure of *arbiter* module.

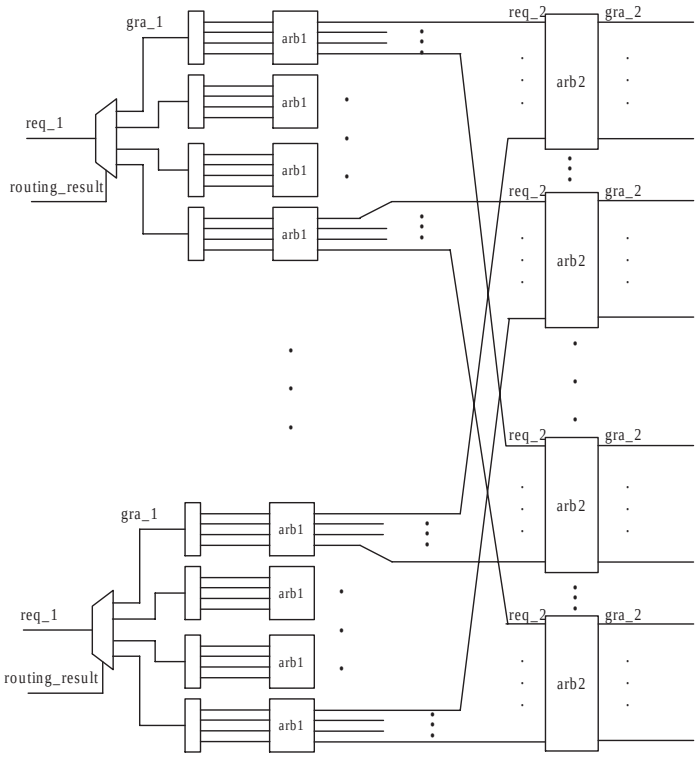


Figure 2.7: Structure of *allo_next* module.

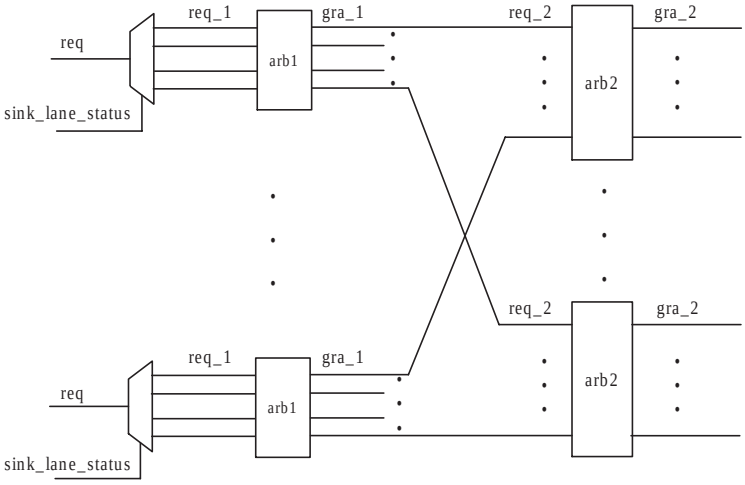


Figure 2.8: Structure of *allo_sink* module.

- Number of sink VCs: 4
- Number of flits per packet: 4
- Depth of VC buffers: 4
- Injection rate (1 packet/n cycles, step=1): 1-30

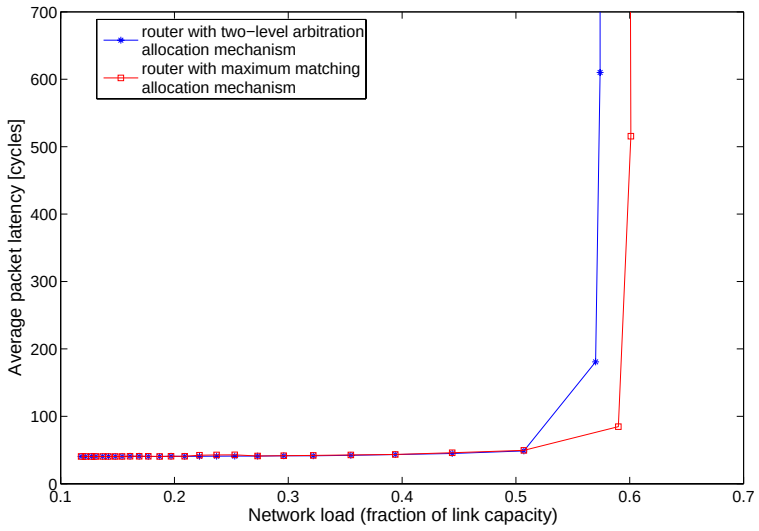


Figure 2.9: Latency comparison of the routers with different allocation mechanisms.

Figure 2.9 and 2.10 present the comparison of the latency and network delivery time characteristics between the routers with two different allocation mechanisms. Figure 2.11 shows the throughput comparison. (These parameters are all representations of the network performance. There are more detailed definition and explanation in the next chapter of this report.) Detailed and clear figures are collected and listed in table 2.1[1] for comparing. From the figures and data in table, the conclusion could be drawn that the newly adopted two-level arbitration allocation method will not exacerbate the performance of the whole system a lot. Only little reduction in the

[1]In this table, all parameter values are calculated from data collected when the network is stable. That is, without data in network warm-up and cool-down periods, these values are calculated. Also, 'average latency' and 'delivery time' refer to the typical values before saturation, or called the 'normally working area' of a network. In this report, when these parameters are listed or talked about again, the same ways are followed.

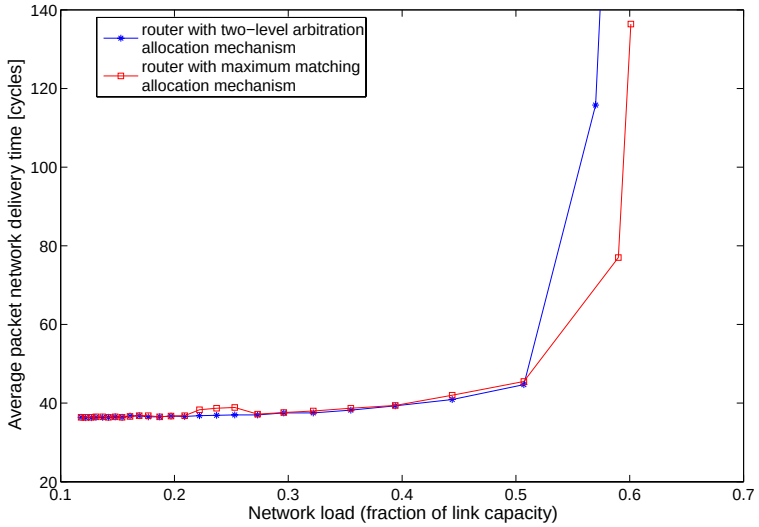


Figure 2.10: Network delivery time comparison of the routers with different allocation mechanisms.

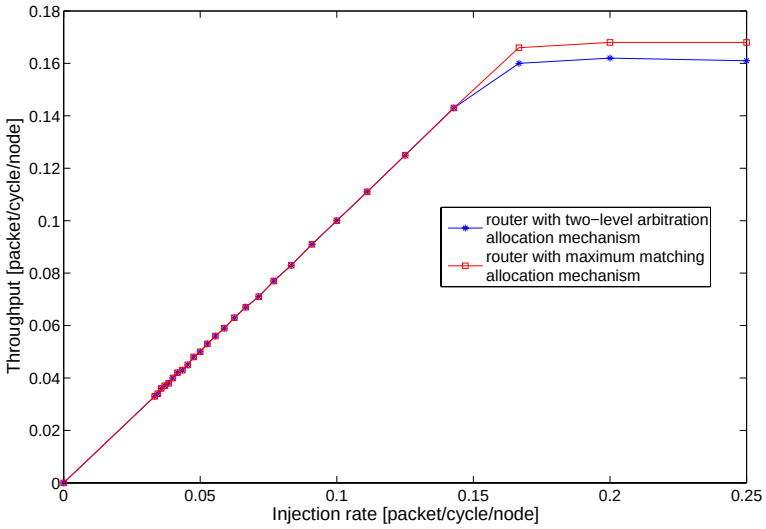


Figure 2.11: Throughput comparison of the routers with different allocation mechanisms.

maximum network load and throughput (2.5% and 3.6%). And even some improvement in average latency and network delivery time before saturation (Although it may not be improvement in fact, at least not great decrement in performance.). In summary, because of the little decrement in performance and great improvement in hardware speed, two-level arbitration allocation method is prove to be practical and efficient instead of the original one.

| | Average latency (cycles) | Average delivery time (cycles) | Max network load (fraction of link capacity) | Max throughput (packet/-cycle/node) |
|---|---|---|---|---|
| router with maximum matching allocation method | 41.2 | 37.2 | 0.601 | 0.168 |
| router with two-level arbitration allocation method | 40.9 | 36.9 | 0.586 | 0.162 |
| Performance decrement (%) | -0.73% | -0.81% | 2.5% | 3.6% |

Table 2.1: Performance comparison of routers with two different allocation mechanisms.

In addition to the structure modification to the three modules, some coding styles were also paid much attention to achieve a fast hardware implementation [10] [11]. For example, the widely used construct of nested 'if' statements has been given deep consideration. Normally an 'if' statement will result in a multiplexer (mux) when synthesized. Different arrangements of these VHDL statements will result in different circuit structures with different area and timing. So when coding the design, try to imagine the hardware structure of the VHDL statements and choose the one you want. In another word, think in hardware, not like software programming.

## 2.4 Synthesis Result Comparison

After the modification to the whole design and function validation, the modified *decoupled-psink* wormhole router was synthesized by Synopsys DC. The technology library is UMC18 (180nm) and in this technology a NAND gate consumes $12.197\mu m^2$. The results of area consumption and timing are listed in table 2.2, compared with the original design. Both designs have 4 virtual channels in a physical channel and 4 admission and sink channels. From the

figures in the table, we can see the great improvement in timing field, also even the reduction of area consumption.

| | Original design | Modified design | Improvement (%) |
|---|---|---|---|
| Crossbar | 2533 gates | 2549 gates | 0% |
| Arbiter | 13087 gates | 6212 gates | 52.5% |
| Other logic | 45242 gates | 43909 gates | 2.9% |
| Whole design[2] | 60862 gates | 52670 gates | 13.5% |
| Max HW speed (ctrl clk/data clk) | 152/76 MHz | 396/198 MHz | 160.5% |

Table 2.2: Area and timing comparison between the original *decoupled-psink* wormhole router and the modified version.

## 2.5 Implementations of the Other Three Models

The other three models: *coupled-psink*, *decoupled-ideal*, and *coupled-ideal* wormhole routers were then realized following the similar structure of the *decoupled-psink* model mentioned above. Especially, the modules of arbitration and allocation use the same two-level arbitration mechanism to ensure high operation speeds.

Not like the decoupled model, in the coupled admission model, routing must be done before a packet enters one admission lane. Because each admission lane is only associated with one output physical channel, the exact lane corresponding to the required output must be chosen to hold the packet after the routing algorithm. In the ideal sink model, each virtual channel has its own sink lane. These lanes do not need to be connected with the *crossbar* as in the p-sink model. They are demuxed with the *crossbar*'s inputs. Packets in virtual channels could either sink directly or traverse the *crossbar* to go ahead. Detailed structure is referred to in figure 2.3 shown at the beginning of this chapter.

After synthesis, the area and timing information is listed in table 2.4 for reference. As mentioned before, the FIFOs from Altera library are used and they are not included in the synthesis result. So the buffer consumptions here are estimated manually. The buffer units in the admission, input and

[2]Since the FIFOs from Altera library are used, they are not counted in the synthesis result [4]. So the area consumption of the whole design does not include the buffers in input and admission lanes.

sink VCs are counted and the results are the product of the counted number and the area consumption of a storage unit in the technology library. In the library of UMC18, the cell of DFFRSPB1 which is probably the storage unit in the lanes consumes an area of 101.639999. A NAND gate consumes an area of 12.197. The total gate consumption of buffers could be calculated by the following formulas:

$$
\begin{aligned}
total\_fifo &= input\_fifo + admi\_fifo + sink\_fifo && (2.1)\\
input\_fifo &= data\_width * depth * VC\_number * PC\_number \\
&\quad * 101.6/12.197 && (2.2)\\
admi\_fifo &= packet\_fifo + flit\_fifo \\
&= (packet\_data\_width * packet\_depth \\
&\quad * packet\_admi\_VC\_number + flit\_data\_width \\
&\quad * flit\_depth * flit\_admi\_VC\_number) \\
&\quad * 101.6/12.197 && (2.3)\\
p\_sink\_fifo &= data\_width * depth * sink\_VC\_number \\
&\quad * 101.6/12.197 && (2.4)\\
ideal\_sink\_fifo &= data\_width * depth * VC\_number * PC\_number \\
&\quad * 101.6/12.197 && (2.5)
\end{aligned}
$$

| | data_width | depth | VC_number | PC_number |
|---|---|---|---|---|
| input_fifo | 30[3] | 2 | 4 | 4 |
| packet_admi_fifo | 112 | 2 | 4 | none |
| flit_admi_fifo | 30 | 3 | 4 | none |
| sink_fifo (p-sink) | 28 | 3 | 4 | none |
| sink_fifo (ideal ejection) | 28 | 3 | 4 | 4 |

Table 2.3: Parameters for calculating the buffer consumption.

For the four router models, the buffer numbers in the input channels (input_fifo) and admission channels (admi_fifo) are the same. The difference is located in the sink buffers because the ideal ejection model has more sink lanes and then more buffers (ideal_sink_fifo) than the p-sink model (p_sink_fifo). Providing the practical parameters (listed in table 2.3) into the formulas above, we can reach the results of 7997 gates for the input channel buffers, 10462 gates for the admission channel buffers, 2799 gates for the

[3]Although the datapath is 32-bit wide in the design, there are some overhead bits which do not need to be buffered in FIFOs.

sink channel buffers in the p-sink model and 11195 gates in the ideal ejection model, the sums of which (total_fifo) are listed in the row labled 'buffer' in table 2.4.

More detailed information about the performance comparison of the four wormhole router models could be accessed in Zhonghai's papers [5] and [6].

| | *decoupled-psink* | *coupled-psink* | *decoupled-ideal* | *coupled-ideal* |
|---|---|---|---|---|
| Crossbar | 2549 gates | 1485 gates | 1280 gates | 817 gates |
| Arbiter | 6212 gates | 5691 gates | 3603 gates | 3620 gates |
| Allo_next | 12604 gates | 12604 gates | 12604 gates | 12604 gates |
| Allo_sink | 3109 gates | 3109 gates | 527 gates | 527 gates |
| Other logic | 28196 gates | 24754 gates | 48161 gates | 44390 gates |
| Buffer | 21258 gates | 21258 gates | 29654 gates | 29654 gates |
| Whole design | 73928 gates | 68901 gates | 95829 gates | 91612 gates |
| Max HW speed (ctrl clk/data clk) | 396/198 MHz | 400/200 MHz | 396/198 MHz | 400/200 MHz |

Table 2.4: Area and timing reports of the routers with different admission and sink models.

# Chapter 3

# Development of the Flit-grouped Mechanism

## 3.1 Investigation of the Wormhole Flow Control

### 3.1.1 Flit Level Link Arbitration

Wormhole flow control is a well-known switching technique and widely used in parallel computers and Network-on-Chips. It has good performance due to pipelined transmission of flits. In wormhole flow control, buffers and channels are allocated to flits rather than packets. So it has small buffer requirement since each switch stores flits instead of packets.

The most distinct character of wormhole flow control is the flit level link arbitration. In this mechanism, all flits must compete for accessing the channel bandwidth before they could advance to the next node. That is, each flit arbitrates to pass itself and the link resources are allocated at flit level. This method generates a contrast with the one at packet level, such as the packet level arbitration mechanism used in the virtual cut-through flow control.

There are many kinds of arbitration policies: fixed priority arbitration, rotating priority arbitration, random priority arbitration, Round-Robin arbitration, weighted Round-Robin arbitration, least recently used policy, etc. [3] [12]. They are all practical policies and could be used as the arbitration or allocation schemes in the wormhole flow control. While no matter which policy is chosen in the wormhole arbiter, links are allocated at flit level.

### 3.1.2 Problem Description

Latency and throughput are two most important parameters which represent the performance of a particular network. Latency is defined as the time required for a message to traverse a network, from the time head arrives at the input port to the time tail departs the output port. Throughput is the data rate in bits per second that the network accepts per input port [3]. Much work has been done and many models have been proposed for improving these parameters and the performance of the network.

Through deep investigation of the canonical wormhole model, we found some problems which have negative effect on the performance of the network. They are related to the flit level arbitration scheme. Because buffers are allocated at packet level while links at flit level, the flits of a packet are switched independently, i.e., not synchronized. As a consequence, flits belonging to the same packet may be distributed in a long range along the path from the source to the destination node. Also along this path, buffers which have been allocated to this packet by the head flit may be left idle (Body or tail flits may not win link arbitrations at the same switching cycle and keep waiting in the original positions.), resulting in buffer or link under-utilization.

As shown in figure 3.1, body flit B11 of a packet cannot win the link arbitration competing with other flits (Shown in the figure, body flit B22 of another packet wins.), while the head flit H1 is lucky enough to keep advancing and allocating buffers along the path for the whole packet. Because of B11's continuous losing in link arbitration and no advancing, buffers along the path are allocated but kept idle. Also if there is no other flits in node 2 which apply for link 2, this link will also be kept idle and wasted. The waste of precious buffer and link resources in the network will result in bad throughput and latency performance.

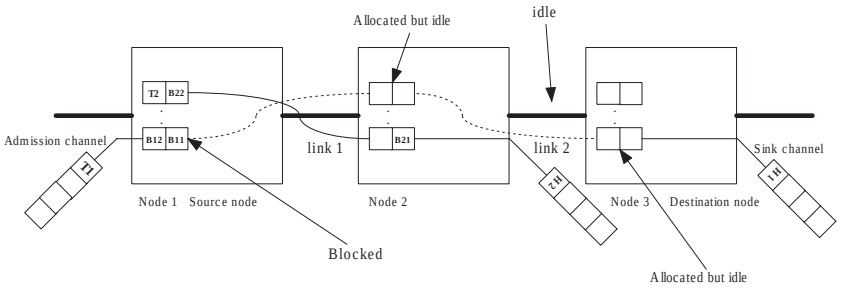


Figure 3.1: Buffer and link under-utilization scenario of canonical wormhole router.

On the other hand, the design of a wormhole router is not a single cycle design [4]. No matter what type the flit is, head, body or tail flit, it will consume multiple clock cycles for a flit to traverse a wormhole router or node. There are many FSM states which contribute to the multiple cycle consumption, such as routing, virtual channel allocation, virtual channel scheduling, and link arbitration, etc.. For a head flit, it must experience the stages of routing, virtual channel allocation and link arbitration to finally enter the next node. For a body or tail flit, the stages are virtual channel scheduling and link arbitration [4]. Except for the waiting time when a flit is blocked and cannot go ahead, the cycle or time consumption of traversing a node to the next one contributes a lot to the latency of a packet.

For the flit level switching in the canonical wormhole flow control, each flit (no matter head, body or tail flit) must experience a link arbitration stage to access the channel bandwidth. In fact this is a waste of clock cycles compared with other packet level flow controls such as virtual cut-through. (In the packet level arbitration mechanism, the arbitration stage is executed only once by the head flit and then the link will be reserved for the whole packet.) Also, in addition to the lane allocation by head flits, body and tail flits need schedule to check the status of the next lanes before they arbitrate for advancing. These complex but necessary stages in the wormhole flow control are all factors which induce multiple clock cycle consumption when traversing a node. And if these stages could be simplified and reduced, much time will be saved.

## 3.2 New Mechanism Development

Through theoretical analysis of the canonical wormhole model, we have arrived at the conclusion that link arbitration at flit level consumes much time for flits to wait in buffers and asynchronous moving of worms (flits) waste precious buffer and link resources by allocating buffers while leaving them idle. To improve the performance of a canonical wormhole router, these are two main aims which should be focused on and solved.

After much analysis and thinking, a flit-grouped mechanism was proposed to eliminate the disadvantages. The inspiration came from the life. We can imitate the process of worm advancing to a real life event: A team of children are checking in aboard to travel abroad on vacation. They must experience some procedures to sit in their seats in the airplane. Also there are many other passengers who will compete with this team to check in. This is just similar to the process of a packet routing from the source to the destination node. The airport lobby is the source and the plane is the destination. This

team is just like a packet composed of many flits (many children). Other passengers form other packets. During the process, each procedure could be treated as the link arbitration to advance. If you have done this, you can go ahead. Or you must wait in the queue. For canonical wormhole model, a child must finish each procedure by him or herself. It is very likely to distribute these children in a wide range from the lobby to the plane in such a crowded and competitive condition: Some have arrived at their seats while some are still waiting in the line of the first procedure. Then the arrived ones must wait for the coming of others to make the team complete. It may consume much time to collect all of the children in the cabin because of the lagging of someone. To solve this embarrassing case, there is a good idea. That is, combine some individuals into one group (e.g. the whole team into one group) and vote out a leader or head. The head of the group checks in each procedure for the whole group, and if permitted, the other members of the group follow the head to go without doing anything. (Of course, if it is allowed by the officers in the airport.) Through this method, it is not possible to lag someone and the team could go ahead keeping a whole part. And, if other passengers are also grouped to check in, the workload will largely shrink for the officers and it will save much time to arrange all the passengers in their seats.

From the life case described above, a new flow control method could be reached: partition the flits of a packet into groups. Each group has some members whose number is the same as the depth of buffers in virtual channels (Normally it should be ensured in practice. Of course the buffer depth may be larger than the group size. However in this condition the buffer resource will not be used efficiently and it is better to enlarge the group size, equal to the buffer depth.). In this condition, buffers are allocated at packet level while links at group level. I.e. the head flit allocates buffers for the following flits and the tail flit releases just as the canonical model of wormhole flow control does. While links are arbitrated only by the flits in the first buffer units called 'group head'. If a group head win the arbitration, the obtained link will be reserved for all other flits in this group. This method is described in detail in the following paragraph.

As shown in figure 3.2 (b), every two consecutive flits (Suppose the buffer depth is two.) are partitioned into a group and the former ones are assigned as 'head'. During delivery if there is one empty buffer unit in the allocated virtual channel of the next hop, group heads arbitrate for the link. If winning the arbitration, this group advances into the next hop flit by flit. Also H allocates buffers for the whole packet and T releases them as in the canonical model of wormhole.

It has been mentioned that if there is *one* idle buffer unit, not the number

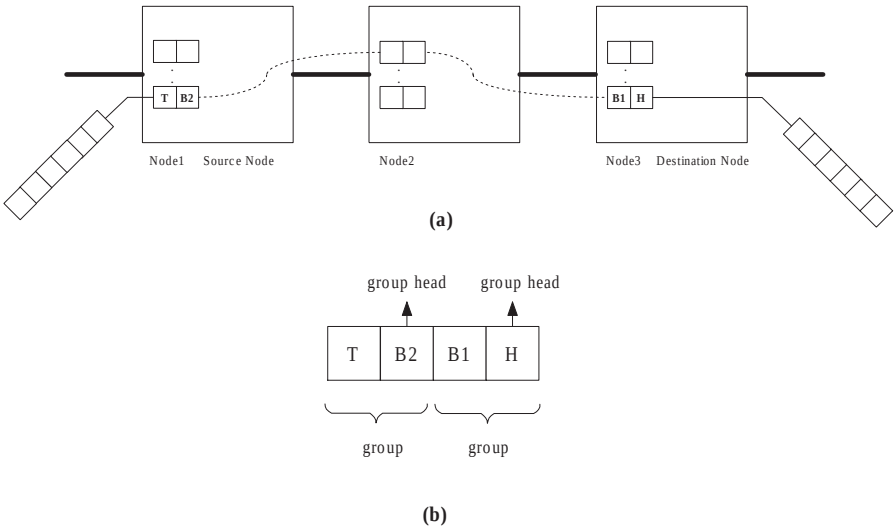


Figure 3.2: Link arbitration at group level.

of members in a group, in the allocated virtual channel of the next hop, groups could arbitrate and go ahead. Why one buffer unit is enough to advance a whole group? The reason is that in flit-grouped mechanism, flits of the same group go ahead one by one without interval and interfering from other group. So if there is one flit buffer unit idle, it means that the group currently occupying this virtual channel is also moving and will give out more buffer units for other coming groups step by step. Then one empty buffer unit is enough for a coming group to move in the virtual channel, following its front group and going together.

Of course, if the buffer resource could be provided large enough, the 4 flits of the packet could be partitioned into a single group, as shown in figure 3.3, to achieve a better performance. In fact, in the condition when a packet are partitioned into a single group, the router will operate just similar to the virtual cut-through flow control. In another word, the virtual cut-through flow control could be treated as a particular case of the flit-grouped mechanism.

Compared with the flit level switching method, there are some advantages with this new flow control mechanism. First, packets are delivered between nodes in unit of group instead of flit. This is a behavior trying to compress the widely distributed flits of a packet and flits belonging to a same packet will go ahead in a more compact fashion than in the wormhole flow control. Then the probability that some flits lag behind and distribute in a long range of nodes along the path between the source and the destination, will

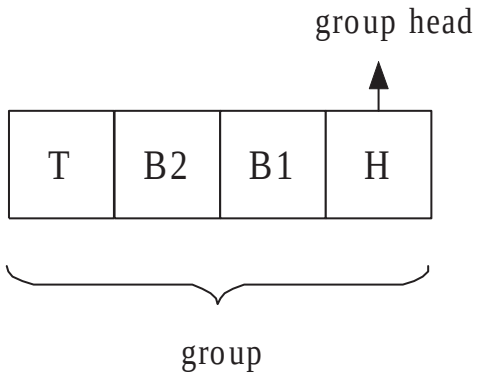


Figure 3.3: 4 flits of a packet are partitioned into one group.

be reduced greatly. In another word, the probability of generating buffer or link under-utilization will be reduced. For example, as an extreme case, if a packet is partitioned into a single group, there will never appear the case that buffers are allocated while kept idle and wasted. Second, for the flits other than group heads, the clock cycles consumed by scheduling and arbitration in the canonical wormhole model are removed (The member flits just follow the group heads to advance without scheduling and arbitration.). This shortens the packet latency and improve the performance of network.

## 3.3 Implementation of the Flit-grouped Router

The implementation of a flit-grouped router is based on the canonical wormhole design which has been realized and validated. Most of the difference between these two models is represented in the control path. As mentioned above, in the canonical wormhole router model, the head flit goes through the routing, virtual channel allocation and link arbitration stages to traverse a node. And the body or tail flit goes through the stages of virtual channel scheduling and link arbitration. All of these operations are controlled by the FSMs in the control path, shown in figure 3.4.

According to figure 3.4, a head flit will go through the states in the sequence of 'S8'-'S9'-'S1'-'S2'-'S3'-'S7'-'S8'('S0'), to traverse a node. While a body or tail flit in the sequence of 'S8'-'S9'-'S1'-'S4'-'S7'-'S8'('S0').

As a new control mechanism, a different FSM module should be realized to match the flit-grouped switching technique. Figure 3.5 shows the flow of the FSM in the flit-grouped router. In this figure, 'S0' is the initial state. When there are flits in the FIFO, the FSM sends out a read signal to the

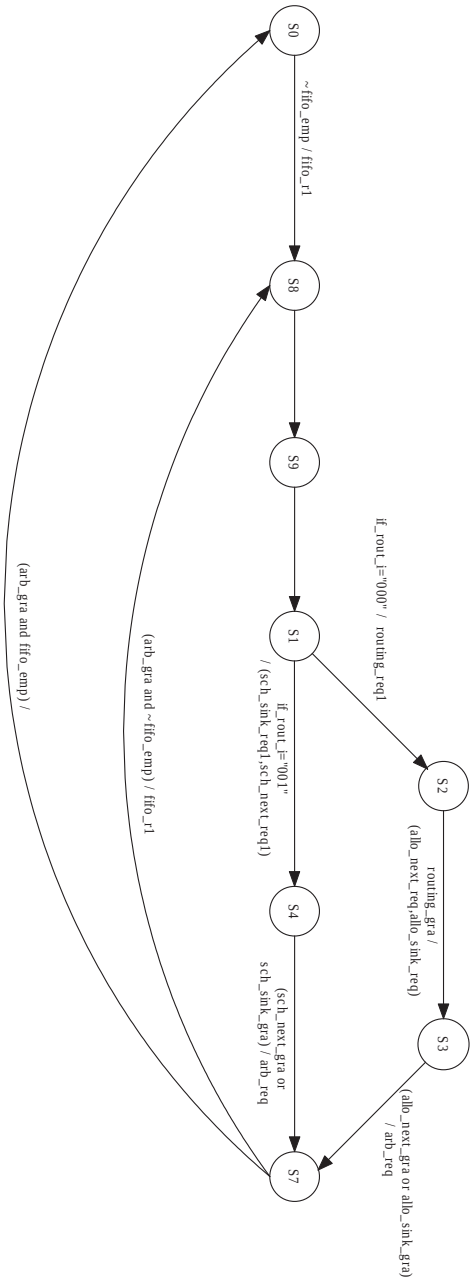


Figure 3.4: The FSM of the canonical wormhole router.

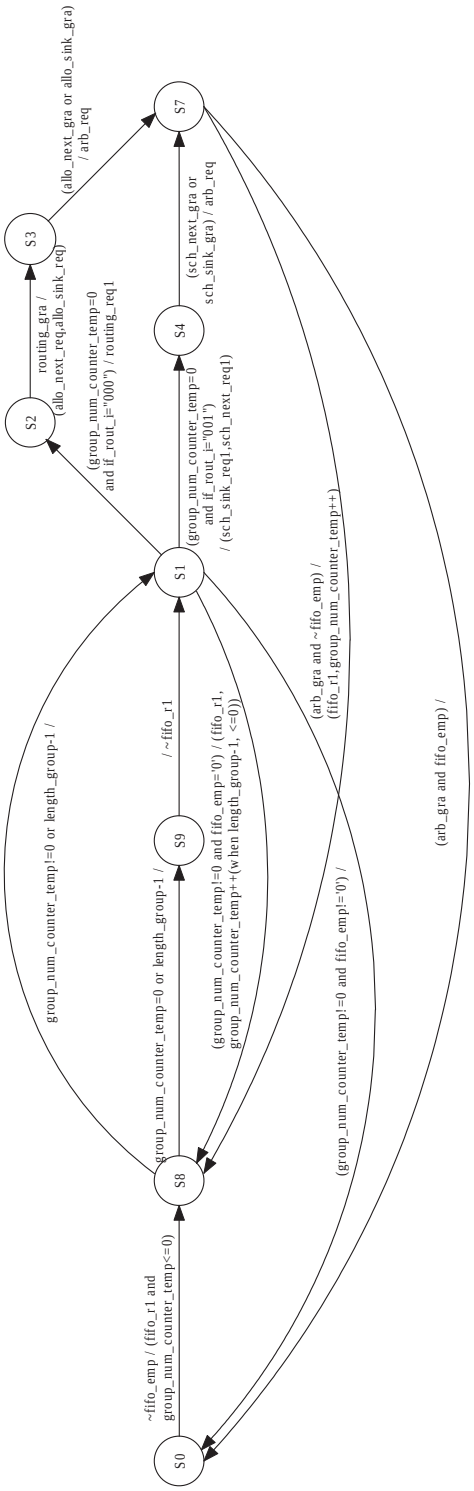


Figure 3.5: The FSM of the flit-grouped router.

FIFO and enters 'S8'. If the present flit is a group head, 'S9' will be entered then, and later 'S1'. The reason to insert the states of 'S8' and 'S9' is to generate a FIFO read signal of two control clock cycles, 'fifo_r1', before reaching the state of 'S1'. To guarantee a stable read-out flit before 'S1', a FIFO read signal of one data clock cycle which is equal to two control clock cycles should be provided. In the state of 'S1', there will be some branches to go: If the present flit is a group head ('group_num_counter_temp'=0) and it is also a head flit of the packet ('if_rout_i'="000"), the state will be transited in the sequence of 'S1'-'S2'-'S3'-'S7', finishing the operations of routing, lane allocation, and link arbitration. This path is identical with the one for a head flit in the canonical wormhole router model. If the present flit is a group head ('group_num_counter_temp'=0) while it is a body or tail flit ('if_rout_i'="001"), the state transition sequence will be 'S1'-'S4'-'S7', finishing the operations of lane scheduling and link arbitration. (Please don't suspect that a body or tail flit could also be a group head. As shown in figure 3.2 (b), flit B2 is the head of the second group.) In 'S7', if the arbitration is granted and there is no flit in the FIFO, the state will goes to 'S0'. Or if there are still flits in the FIFO, goes to 'S8'.

After the work of the group heads has been done, group members will just follow their heads to go. Demonstrated in the FSM figure, the state is transited between 'S8' and 'S1' repeatedly. In each repeat cycle, a member flit is delivered and a FIFO read signal is generated to read out the next flit. Member flits will do nothing: no lane scheduling, no link arbitration, just go! After delivering all the flits of a group, the state will be changed from 'S1' to 'S0' or 'S8', according to the status of the FIFO.

In addition to the modification on the FSM, the module of *arbiter* should also be redesigned for catching the characteristics of the flit-grouped flow control mechanism. In the canonical wormhole router, if a link request wins in the arbitration process, a grant of one data cycle will be ensured to send out a flit. By contrast in the flit-grouped model, the grant signal must be long enough to deliver all the flits belonging to a same group. Just as mentioned before, members of a group will not do anything but follow the group head to go. So in the arbitration process done by group heads, the grant signal is to allow the going of not only the group head itself, but also its members.

There are still some adjustments in other modules accordingly except for the changes in the FSM and *arbiter* listed above. After all the modifications on the related modules, a synthesizable design of flit-grouped router has been achieved for later investigation.

## 3.4 Experiments

Till now, we have obtained the VHDL designs of the canonical wormhole and the flit-grouped routers. Based on the two designs, a large amount of simulation experiment has been done to demonstrate and compare the performance of networks constituted by the two different router models.

### 3.4.1 Method of Measuring the Performance

A standard method to measure the network's performance is shown in figure 3.6 [3]. To run the network, terminal instrumentations should be at-

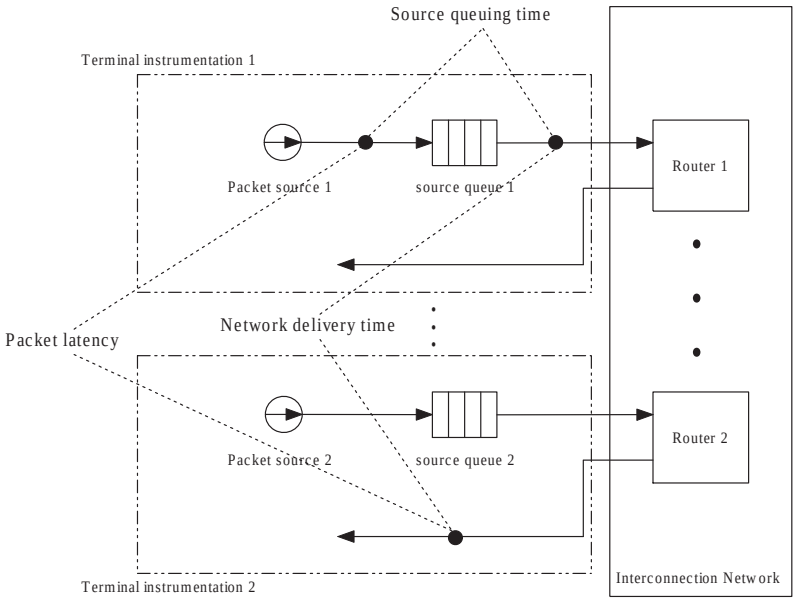


Figure 3.6: A standard method to measure the network's performance.

tached to each terminal or port of the network. In each instrumentation, a packet source generates packets according to a specific traffic pattern, packet length, and interarrival time distribution. A source queue is added between the source and the network, to buffer the packets before they enter the network. To avoid overflow, it should be large enough, or say infinite. This queue is not part of the network being simulated, but serve to isolate the traffic processes from the network itself. To measure the performance of network, three important time parameters should be recorded for analysis: packet latency, source queuing time and network delivery time, as noted in figure 3.6. It is clear that latency is the sum of the other two parameters.

In practice, a testbench file is adopted to provide packets to be transmitted and collect packets received for all nodes in the network. Also in this file, the source queues which are large enough are also included. This testbench file is the implementation of the terminal instrumentations of all nodes in fact.

In the tests, packets are periodically injected into the network. For a 4 x 4 2D mesh network, the total number of the packets to be sent is 16000, 1000 for each node. Through changing the injection rate of packets, various network load instances could be reached. Also the uniform traffic patten is adopted in the tests. Each router sends out packets to all the other nodes randomly. The destination node number is generated by a uniform random function and each node uses a different seed. Under these conditions, parameters such as latency and throughput are recorded to represent the network's performance.

Performance of the flit-grouped flow control model should be compared with the *decoupled-psink* wormhole model, because it is also based on a *decoupled-psink* model. Improvement then could be observed to present the advantages of the new flow control method.

### 3.4.2 Simulation Results

At the beginning, some useful formulas are listed for processing the collected data and calculating parameters [4]. Then curves are drawn for these parameters and results are compared.

- Network load (fraction of link capacity):

$$\frac{Sum_{active\_link}}{Number_{link} * Cycle_{simulation}}$$

- Average packet latency:

$$\frac{Sum_{latency}}{Number_{packet\_received}}$$

- Packet injection rate:

$$\frac{Number_{flit\_injected}}{Number_{node} * Cycle_{simulation}}$$

- Throughput:

$$\frac{Number_{flit\_received}}{Number_{node} * Cycle_{simulation}}$$

| | test 1 | test 2 | test 3 | test 4 | test 5 | test 6 |
|---|---|---|---|---|---|---|
| Mesh size | 4 x 4 | | | | | |
| No. of VCs per PC | 4 | | | | | |
| No. of admission VCs | 4 | | | | | |
| No. of sink VCs | 4 | | | | | |
| No. of flits per packet (packet size) | 8 | 8 | 8 | 8 | 8 | 16 |
| Depth of VC buffers | 4 | 2 | 8 | 4 | 4 | 8 |
| No. of flits per group (group size) | 4 | 2 | 8 | 4 | 4 | 8 |
| Injection rate (1 packet/n cycles, step=1) | 6-35 | 6-35 | 6-35 | 6-35 | 6-35 | 21-50 |
| Link arb. method | R-R | R-R | R-R | fixed p. | rand p. | R-R |

Table 3.1: Network parameters in all test conditions.

Also for clearly consulting, table 3.1 lists the great deal of test conditions under which results are collected and analyzed later.

**1. Results in a Typical Simulation Condition**

First the simulation was executed under a typical condition (test 1[1]) of:

- Mesh size: 4 x 4
- Number of VCs per PC: 4
- Number of admission VCs: 4
- Number of sink VCs: 4
- Number of flits per packet (packet size): 8
- Depth of VC buffers: 4
- Number of flits per group (group size): 4 (only in the flit-grouped mechanism)
- Injection rate (1 packet/n cycles, step=1): 6-35

Results of the *decoupled-psink* wormhole router and the flit-grouped one are compared as the following.

[1]Let's call the simulation under this condition 'test 1'. In this report, the data and figures marked by 'test 1' are collected and drawn in this scenario of the network. Also the same marking method will be followed to differ the data and figures under different simulation conditions.

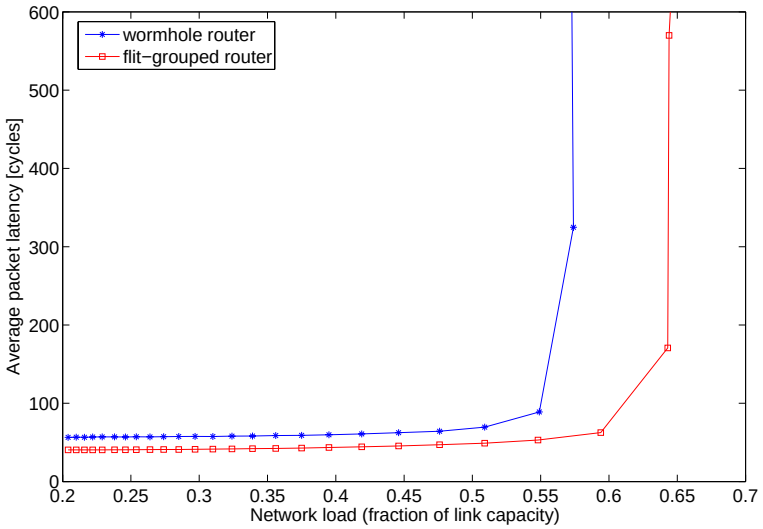


Figure 3.7: Latency comparison (test 1).

In figure 3.7, average latency values are recorded at different network loads (corresponding to different packet injection rates) and curves are drawn from them. From the figure it could be recognized that when the network load is low, the average latency is to keep stable at some value and after the network is saturated, the average latency increases very fast without increasing the network load. This is because after saturation, the packet injection rate will be larger than the absorbtion rate of the network and packets will spend much time waiting in the source queues before entering the network for delivery. Also, it is clear that the curve for the flit-grouped router is below the one for wormhole router before saturation, and it arrives at the saturation point at a much higher network load value than the wormhole router. This represents the advantages of the flit-grouped mechanism that, in the normally working area (area before saturation), the new mechanism could achieve a lower average latency than the wormhole flow control. And it could enlarge the normally working area to a higher network load and make a better use of the link resource in the network.

Especially, delivery time consumed in the network should be given more attention for analysis. The figure for the network delivery time is shown in 3.8.

As mentioned before, the network delivery time is defined as the time interval from entering the network to leaving it. It is one of the most im-

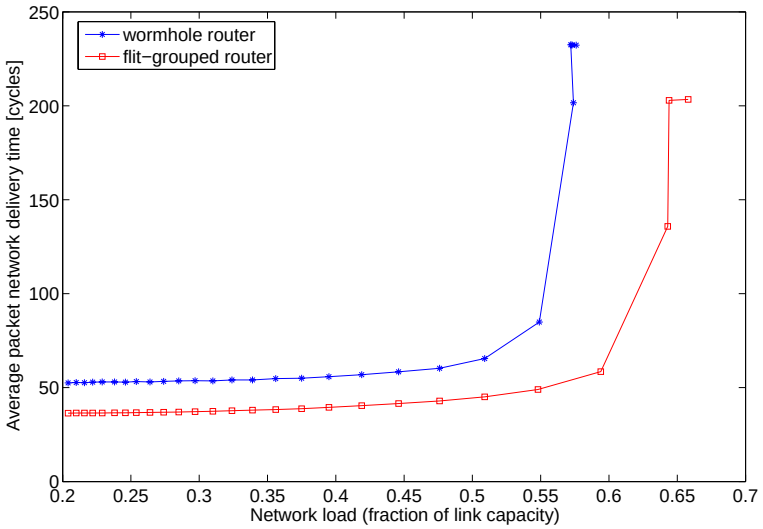


Figure 3.8: Network delivery time comparison (test 1).

portant parameters which represent the performance of a certain type of network. Similar to the figure of latency comparison, great improvement is exhibited by the reduction of time consumption before saturation and the increment in fraction of network capacity when saturated. Correspondingly, the enhancement in throughput is shown in figure 3.9.

Clearly, the improvement is represented by detailed figures in the following table 3.2.

| | Average latency (cycles) | Average delivery time (cycles) | Max network load (fraction of link capacity) | Max throughput (packet/-cycle/node) |
|---|---|---|---|---|
| *decoupled-psink* wormhole router | 57.2 | 53.2 | 0.577 | 0.0810 |
| Flit-grouped router | 40.9 | 36.9 | 0.658 | 0.0916 |
| Performance improvement (%) | 28.5% | 30.6% | 14.0% | 13.1% |

Table 3.2: Performance improvement of the flit-grouped mechanism (test 1).

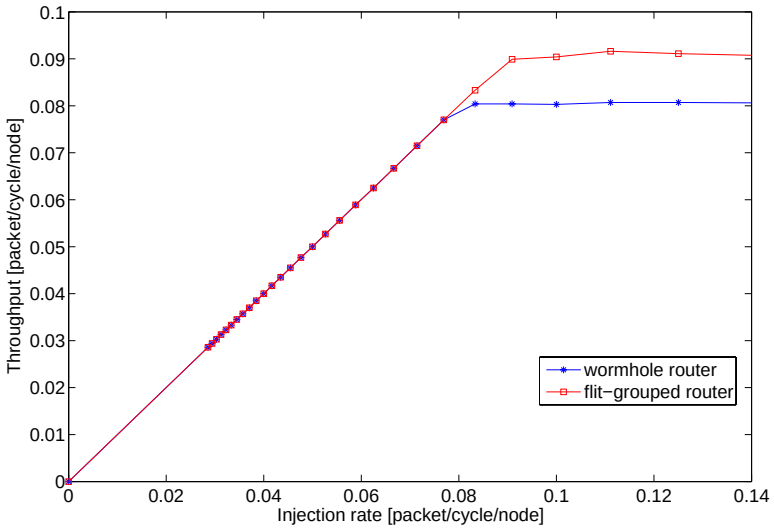


Figure 3.9: Throughput comparison (test 1).

More deeply, figure 3.10 and 3.11 show the packet distribution of the network delivery time before and after saturation. They are direct explanations of the performance improvement. From the figures, it could be observed that the new flow control method has changed the distribution of the packet network delivery time, increasing the number of packets with lower delay time and moving the distribution columns to the side of lower time consumption. And so the average value of the network delivery time is reduced a lot.

### 2. Results for Different Group Sizes

To discover the performance of the new flow control mechanism in a deeper level, more tests should be done except for the typical one above. Because the performance of the flit-grouped router has much to do with its group size, a good idea is to consider the two extreme cases of this important parameter, while keeping the packet size constant. First the worst-case test (test 2) which has a smallest group size of 2, with some different network parameters listed:

- Number of flits per packet (packet size): 8
- Depth of VC buffers: 2
- Number of flits per group (group size): 2 (only in the flit-grouped mechanism)

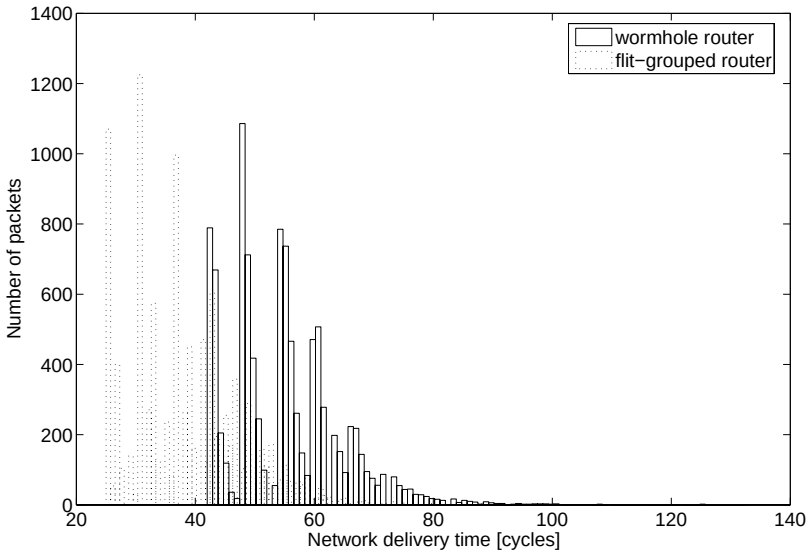


Figure 3.10: Packet distributions before saturation (test 1, packet injection rate = 1 packet/20 cycles).

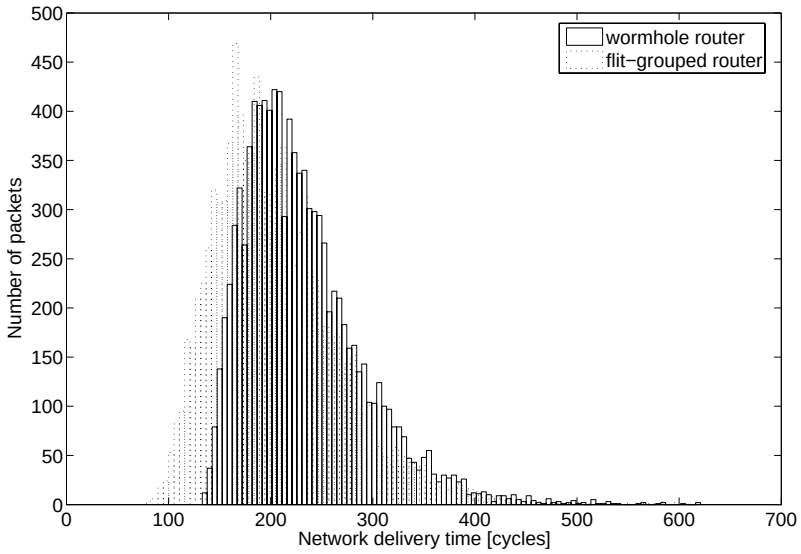


Figure 3.11: Packet distributions after saturation (test 1, packet injection rate = 1 packet/6 cycles).

Then the best-case test (test 3) for the group size of 8, which is equal to the flit number of a whole packet:

- Number of flits per packet (packet size): 8
- Depth of VC buffers: 8
- Number of flits per group (group size): 8 (only in the flit-grouped mechanism)

Comparing the flit-grouped router with the wormhole router, the corresponding simulation results are listed in figures B.1, B.2, and B.3 in Appendix B for the minimum possible group size of 2 (If the group size is 1, it returns to a wormhole model, no longer the flit-grouped router.). In the condition of the maximum possible group size of 8, the performance comparison is demonstrated by the figures B.4, B.5, and B.6.

From the detailed figures listed in table 3.3 and 3.4, advantages could be represented by the improvement of all the parameters. Moreover, adding the corresponding figures in table 3.2, it could be revealed that, the performance improvement (especially improvement in the items of the 'average latency' and 'average delivery time') increases a lot with the growth of the group size, from 2 to 4 then to 8. (It is difficult for the items of the 'max network load' and 'max throughput' to continue a high improvement in percentage when the group size is large, because the load fraction of the link capacity in the network has already been so high and almost reached its limit.) It could be explained by the fact: In the test condition of a large group size (e.g. 8), one arbitration of a group head will take away more flits. For example, when the group size is 8, 7 flits will be taken away by their head without any action. While if the group size is 2, just one. The less frequently happened arbitration actions will save much time and lead to smaller latency and delivery time in the network. As well, a larger group size could reduce the possibility of buffer or link under-utilization, moving packets in a more compact way. Then from these reasons we can conclude that, for better performance, try to partition as many flits into one group as possible.

### 3. Results for Different Arbitration Policies

Till now all of the simulation tests are based on the designs with a Round-Robin arbitration policy. According to the analysis into the flit level arbitration, the flit-grouped flow control mechanism should be adaptive in improving the performance of any kind of arbitration policy at flit level. To validate this conclusion, two other experiments have been done except for the Round-Robin method.

| | Average latency (cycles) | Average delivery time (cycles) | Max network load (fraction of link capacity) | Max throughput (packet/-cycle/node) |
|---|---|---|---|---|
| *decoupled-psink* wormhole router | 58.5 | 54.5 | 0.505 | 0.0713 |
| Flit-grouped router | 55.2 | 51.2 | 0.532 | 0.0749 |
| Performance improvement (%) | 5.6% | 6.1% | 5.3% | 5.0% |

Table 3.3: Performance improvement of the flit-grouped mechanism (test 2).

| | Average latency (cycles) | Average delivery time (cycles) | Max network load (fraction of link capacity) | Max throughput (packet/-cycle/node) |
|---|---|---|---|---|
| *decoupled-psink* wormhole router | 57.1 | 53.2 | 0.620 | 0.0863 |
| Flit-grouped router | 36.8 | 32.8 | 0.686 | 0.0953 |
| Performance improvement (%) | 35.6% | 38.3% | 10.6% | 10.4% |

Table 3.4: Performance improvement of the flit-grouped mechanism (test 3).

First, the fixed priority arbitration mechanism was chosen, in the same typical condition (test 4) of the network as the Round-Robin's stated before. That is, in addition to the same network parameters, the packet size is 8 and the group size is 4. In addition, another arbitration policy of random priority was tested with the same network parameters (test 5). Similar to the results of the Round-Robin's, related figures could be referred to in Appendix B.3 and B.4 to exhibit the great improvement of the flit-grouped flow control mechanism. Table 3.5 and 3.6 are listed to summarize the final results.

From the figures in the item of 'performance improvement' in table 3.5, 3.6 and 3.2, we can see that similar great improvement could be achieved by the flit-grouped flow control method, no matter what arbitration policy is adopted in the router design.

**4. Results for Different Packet Sizes**

| | Average latency (cycles) | Average delivery time (cycles) | Max network load (fraction of link capacity) | Max throughput (packet/-cycle/node) |
|---|---|---|---|---|
| *decoupled-psink* wormhole router | 57.3 | 53.3 | 0.580 | 0.0806 |
| Flit-grouped router | 40.8 | 36.8 | 0.670 | 0.0932 |
| Performance improvement (%) | 28.8% | 31.0% | 15.5% | 15.6% |

Table 3.5: Performance improvement of the flit-grouped mechanism (test 4).

| | Average latency (cycles) | Average delivery time (cycles) | Max network load (fraction of link capacity) | Max throughput (packet/-cycle/node) |
|---|---|---|---|---|
| *decoupled-psink* wormhole router | 57.3 | 53.3 | 0.577 | 0.0810 |
| Flit-grouped router | 40.9 | 36.9 | 0.651 | 0.0911 |
| Performance improvement (%) | 28.6% | 30.8% | 12.8% | 12.5% |

Table 3.6: Performance improvement of the flit-grouped mechanism (test 5).

Another simulation test (test 6) has been done to get an overview of the performance improvement at different packet sizes. The network condition for testing is listed as the following:

- Mesh size: 4 x 4
- Number of VCs per PC: 4
- Number of admission VCs: 4
- Number of sink VCs: 4
- Number of flits per packet (packet size): 16
- Depth of VC buffers: 8
- Number of flits per group (group size): 8 (only in the flit-grouped mechanism)

- Injection rate (1 packet/n cycles, step=1): 21-50

| | Average latency (cycles) | Average delivery time (cycles) | Max network load (fraction of link capacity) | Max throughput (packet/-cycle/node) |
|---|---|---|---|---|
| *decoupled-psink* wormhole router | 90.3 | 86.3 | 0.607 | 0.0423 |
| Flit-grouped router | 52.7 | 48.7 | 0.674 | 0.0471 |
| Performance improvement (%) | 41.6% | 43.6% | 11.0% | 11.3% |

Table 3.7: Performance improvement of the flit-grouped mechanism (test 6).

Compared with the best-case test when the packet size and group size are both 8, in the new simulation condition the packet size has been extended to 16 while keeping the same VC buffer depth and group size. The corresponding figures to show the performance comparison are collected in Appendix B.5, and the final statistical results are listed in table 3.7. From the figures in percentage of the performance improvement in table 3.7 and 3.4, it's clear that when the group size keeps fixed, a longer packet size normally means a similarly large or even larger improvement on the performance of the network.

We can evaluate the *base average packet latency* (i.e. the minimum average latency with a low injection rate and no contention) using the following formula [4]:

$$\begin{aligned} \textit{Base average packet latency} = {} & Cycle_{source\ queue} + Cycle_{head\ admission} \\ & + (\frac{2}{3} * N - 1) * Cycle_{head\ pass} + Cycle_{head\ sink} \\ & + Cycle_{body\ sink} * Length_{body} + Cycle_{tail\ sink} \end{aligned} \quad (3.1)$$

In the listed formula, the word 'admission' means that a flit goes through the source node from its admission packet queue to the input of the next node. 'Pass' means a flit goes through a node along the path from the input to the output or the input of the next node. 'Sink' means a flit traverse the destination node and is ejected from the network in the format of a packet. In our multi-cycle designs, the cycle numbers consumed by different flit types to finish the behavior of admission, pass or sink are described in table 3.8 and 3.9. Also for a packet, the number of cycles consumed in the source

queue is 4. The unit is data clock cycle (One data clock cycle consists of two control clock cycles).

| | Pass | Admission | Sink |
|---|---|---|---|
| Head flit | 6 | 7 | 7 |
| Body or tail flit | 4 | 4 | 4 |

Table 3.8: Consumed cycle numbers for the wormhole router.

| | Pass | Admission | Sink |
|---|---|---|---|
| Group head (packet head flit) | 6 | 7 | 7 |
| Group head (packet body flit) | 4 | 4 | 4 |
| Group member of body or tail flit | 1 | 1 | 1 |

Table 3.9: Consumed cycle numbers for the flit-grouped router.

With the detailed figures in table 3.8 and 3.9 and the formula listed above, the theoretical base average latency of the two models in different test conditions could be computed and shown in table 3.10. By contrast, the measured base average latency of the two models are listed in table 3.11

| | wormhole router (cycles) | flit-grouped router (cycles) | Improvement (%) |
|---|---|---|---|
| test 1 | 56 | 38 | 32.1% |
| test 2 | 56 | 44 | 21.4% |
| test 3 | 56 | 35 | 37.5% |
| test 4 | 56 | 38 | 32.1% |
| test 5 | 56 | 38 | 32.1% |
| test 6 | 88 | 46 | 47.7% |

Table 3.10: Computed base average latency comparison for two models.

It could be seen that there are some differences in the corresponding base average latency couples between the computed and the measured one. The reason is that in our test platform the sixteen nodes inject packets into the network at a same cycle. Although the injection rate is very low, there are still some contentions happened in the network. Then the measured base latency includes both the computed part in theory and the blocking time in practice. The test platform is to be improved to realize a random injection for each node and then the latency couples will be more similar than the present scenario. Also, the measured performance improvement will be much closer

| | wormhole router (cycles) | flit-grouped router (cycles) | Improvement (%) |
|---|---|---|---|
| test 1 | 56.5 | 40.4 | 28.5% |
| test 2 | 57.2 | 54 | 5.6% |
| test 3 | 56.5 | 36.5 | 35.4% |
| test 4 | 56.6 | 40.4 | 28.6% |
| test 5 | 56.6 | 40.4 | 28.6% |
| test 6 | 90.2 | 52.7 | 41.6% |

Table 3.11: Measured base average latency comparison for two models.

to the computed values (e.g. in test 1, 2 and 3, 32.1%, 21.4% and 37.5% respectively), which are much larger than our measured ones listed before (28.5%, 5.6% and 35.6%).

## 3.5 Synthesis Results

After simulation, the flit-grouped router was then synthesized for revealing its hardware character in practice. The bottom-up method [9] was still used in this task. Each module was optimized and the corresponding report information was recorded separately. The top level in the hierarchy, *router*, was then synthesized with the combination of different modules. After finishing this time-consuming work, timing and area reports are recorded for the comparison with the *decoupled-psink* wormhole router. The results are listed in table 3.12.

| | *decoupled-psink* wormhole router | Flit-grouped router | Increment (%) |
|---|---|---|---|
| Crossbar | 2549 gates | 2549 gates | 0% |
| Arbiter | 6212 gates | 10650 gates | 71.4% |
| Allo_next | 12604 gates | 13172 gates | 4.5% |
| Allo_sink | 3109 gates | 3511 gates | 12.9% |
| Other logic | 28196 gates | 26033 gates | -7.7% |
| Buffer | 21258 gates | 21258 gates | 0% |
| Whole design | 73928 gates | 77173 gates | 4.4% |
| Max HW speed (ctrl clk/data clk) | 396/198 MHz | 392/196 MHz | -1.0% |

Table 3.12: Area and timing reports.

From the figures in the table, it proves that the new flow control, flit-

grouped mechanism, could be easily implemented in hardware with little additional area consumption (4.4%) and almost the same maximum operation speed of the hardware. So it is a practical mechanism and useful in improving the performance of routers and networks.

## 3.6 Limitations

Like almost all the things in the world, flit-grouped flow control mechanism is not perfect and also has some short-comings or limitations in practical use. First, the packet size (flit number per packet) must be an integer times of the virtual channel buffer depth (also the group size). As in the tests before, 8 (flits/packet) is one time of 8 (storage units in each VC buffer) and there is only one group in a packet; 8 (flits/packet) is two times of 4 (storage units in each VC buffer) and there are two groups in a packet. And so on. In some conditions that flit number per packet is not an integer times of the VC buffer depth, additional flits with no information need be filled in for partitioning flits into groups. This is a waste of the network resources and will exacerbate the performance of network. Second, for better performance, as many flits as possible are expected to be grouped together, as shown in the simulation results before. However it is inconsistent with the viewpoint that buffer units should be saved as possible because of their large area-consumption. Then in practice, a well-balanced point should be found between performance and area consumption to achieve a practical and reasonable network. Fortunately, as the development of semiconductor technology, more and more gates could be integrated in a single chip. Thus more buffer units may appear in a virtual channel in the coming days and the advantages of flit-grouped flow control mechanism will be well performed then.

# Chapter 4

# Summary

In the thesis project, two main tasks have been executed and realized: the first one was that enhancing a hardware implementation of the *decoupled-psink* wormhole router's RTL design. Also, based on the enhanced version, the other three models, *coupled-psink*, *decoupled-ideal* and *coupled-ideal* were implemented in hardware and the corresponding area and timing information was reported. The second task was to develop a new flow control mechanism to improve the performance of the canonical wormhole router. According to this requirement, the flit-grouped flow control mechanism was introduced, with simulation and synthesis results validated its feasibility and large improvement on the network's performance.

## 4.1 Conclusion

From experimental results, conclusions are drawn as follows:

- After modifying the modules of *arbiter*, *allo_next* and *allo_sink* from a maximum allocation mechanism to a two-level arbitration mechanism, circuit operation speed has been improved from 76MHz to almost 200MHz. (This speed is referred to as the data path clock frequency of routers. The control clock speed is twice as fast as the data clock.)

- *Coupled-psink*, *decoupled-ideal* and *coupled-ideal* wormhole models were implemented in hardware based on the speed-up version of the router's RTL design. From the reported information, it could be recognized that these four models could achieve the almost same circuit operation speed, and the two ideal sink models will consume more area than p-sink models because of large buffer and control logic consumption in the sink lanes.

- A new flow control mechanism, flit-grouped mechanism is introduced after revealing the disadvantages of wormhole flow control. From the simulation results, this new method could improve many performance parameters a lot, such as the average latency, average network delivery time, maximum throughput, and maximum network load. Simulation results observe up to 43.6% network delivery time, 41.6% packet latency, 15.5% network load, and up to 15.6% network throughput. From the synthesis results, it induces little additional area consumption (4.4%) and speed penalty (1%).

## 4.2 Future Work

Future work is expected in the following directions:

- To achieve higher operation speed of the hardware circuit, pipelined structures of the complex logic block, such as *arbiter*, *allo_next* and *allo_sink*, could be considered and realized. In a pipelined model, latency and throughput parameters will be greatly enhanced for improving the network's performance.

- A flit-grouped router which is able to deal with packets of various sizes is to be modeled and implemented. For better performance, new mechanisms should be adopted to reduce the overhead when the packet size is not regular and some additional flits without information are inserted. Or consider to combine the wormhole with the flit-grouped flow control in one router: That is, try to make the flit level and the group level arbitration methods cooperate to achieve good performance. The final design should be validated to have better performance when dealing with the various size packets, compared with the pure wormhole router.

- Power investigation could be done for deeper performance comparison between the canonical wormhole router and the flit-grouped one.

# Appendix A

# Code Structure

The structure of the VHDL files is shown in figure A.1. The listed files in the dashed-line box should be compiled first before the compilation of others. The files of "altera_mf.vhd" and "altera_mf_components.vhd" are used for "fifo.vhd". They are generated from Altera Quartus II 4.1 version. The file of "DWpackages.vhd" is a library from Synopsys. The "global_parameters.vhd" file defines the global parameters of the whole design for easy configuration. The "func_pack.vhd" file is a package file which defines a function of arbitration. This file is widely used in the modules of *arbiter*, *allo_next* and *allo_sink* to construct the structure of two level arbitrations.

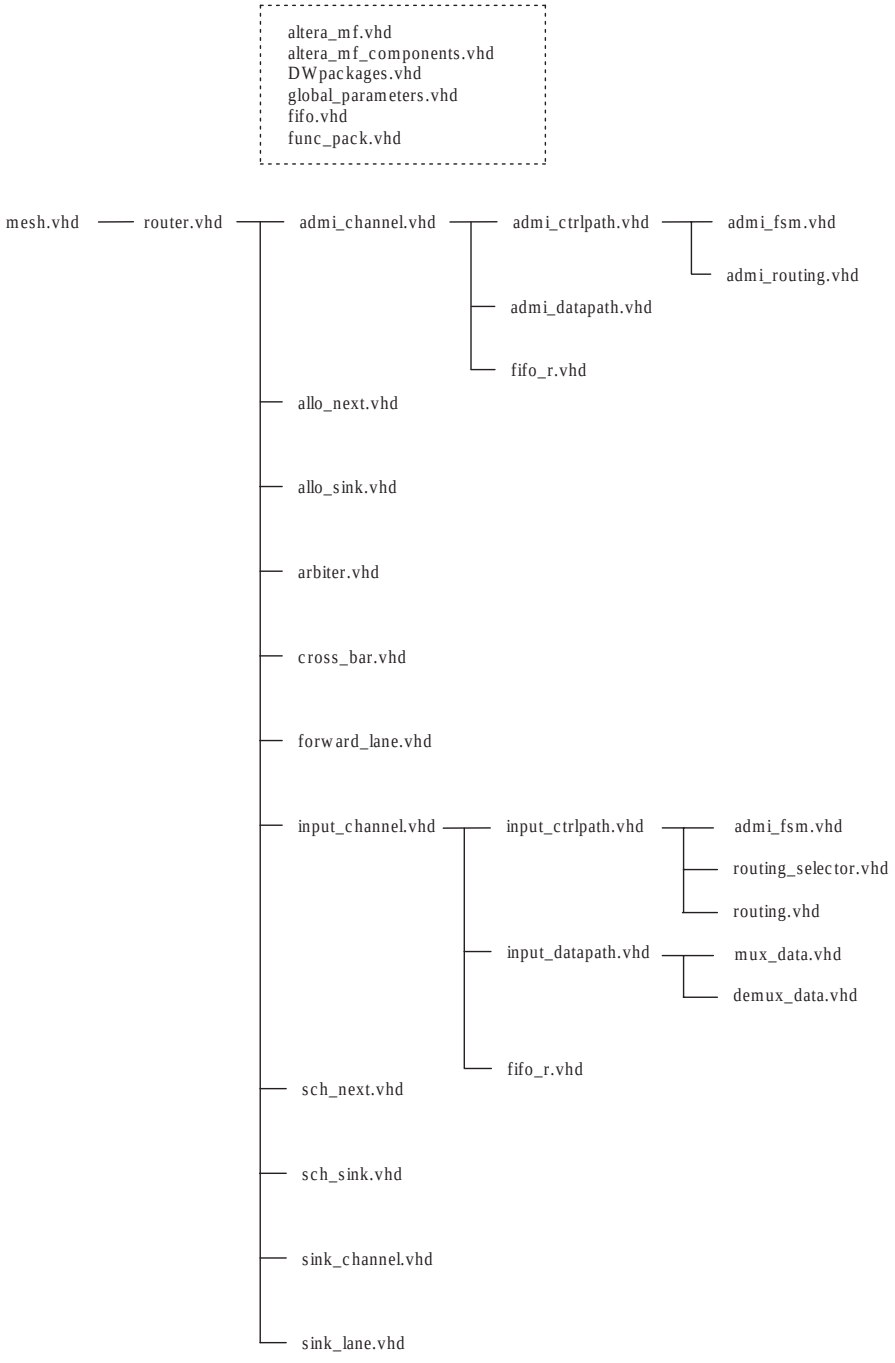


Figure A.1: The structure of VHDL files.

# Appendix B

# Performance Comparison

## B.1 Performance Comparison in the condition of 'test 2'

Performance comparison figures in the simulation condition of 'test 2' for two networks constructed by the canonical wormhole routers and the flit-grouped routers. In this test, the packet size is 8 and the group size is 2. The arbitration policy is Round-Robin for both router models.

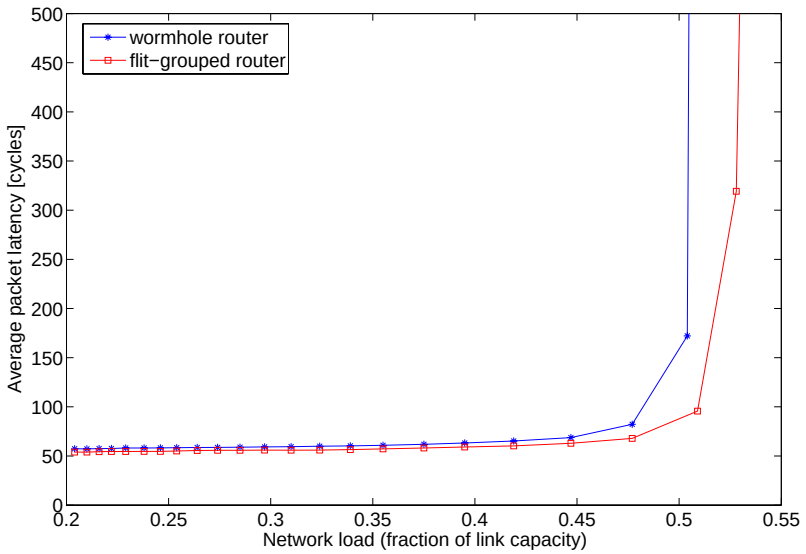


Figure B.1: Latency comparison (test 2).

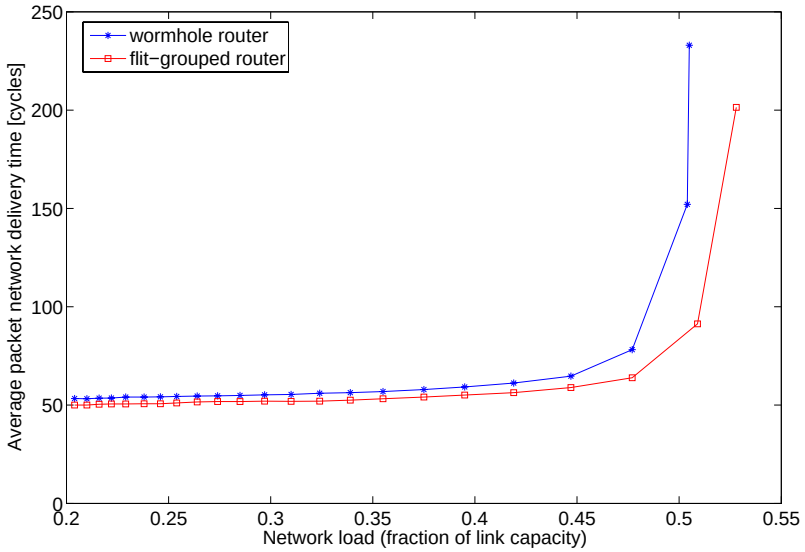


Figure B.2: Network delivery time comparison (test 2).

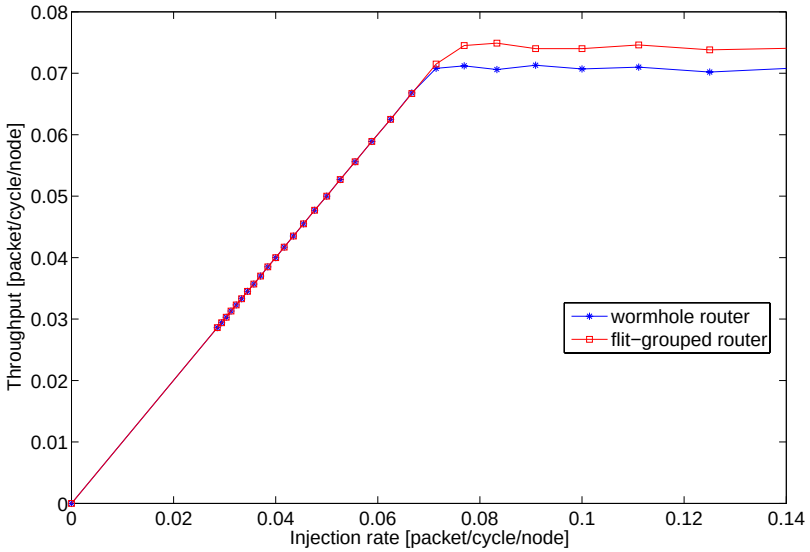


Figure B.3: Throughput comparison (test 2).

## B.2 Performance Comparison in the condition of 'test 3'

Performance comparison figures in the simulation condition of 'test 3' for two networks constructed by the canonical wormhole routers and the flit-grouped routers. In this test, the packet size is 8 and the group size is 8. The arbitration policy is Round-Robin for both router models.

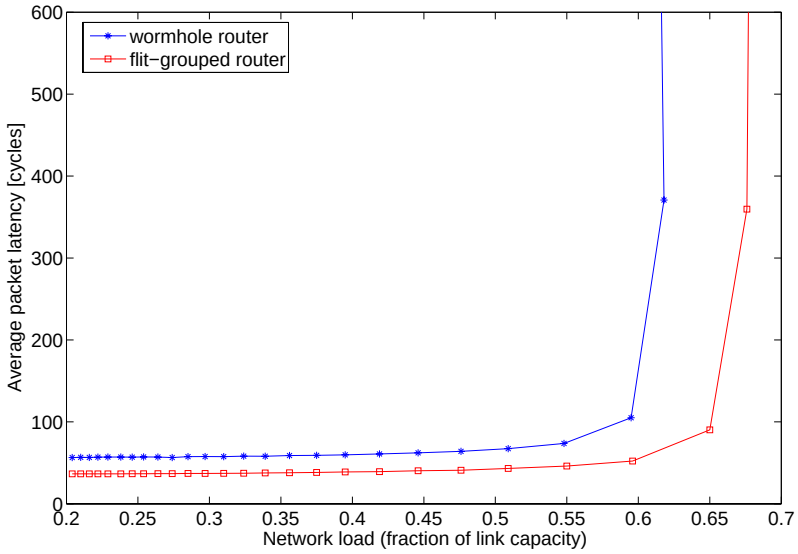


Figure B.4: Latency comparison (test 3).

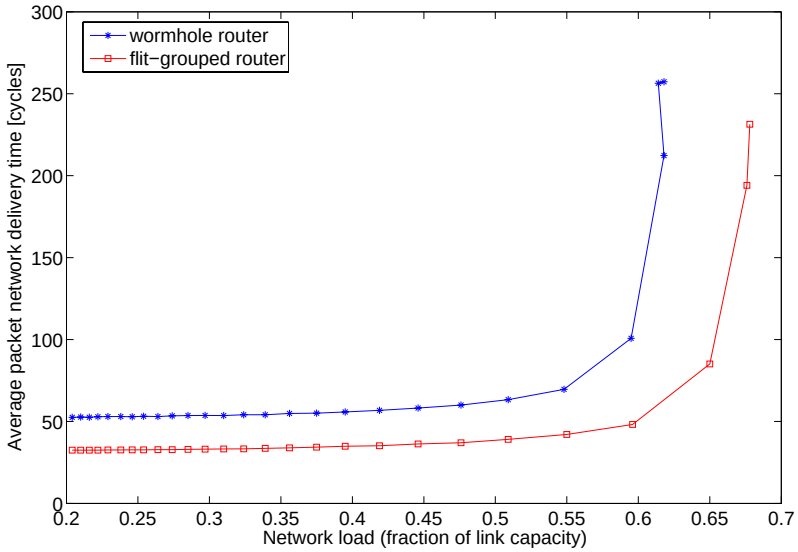


Figure B.5: Network delivery time comparison (test 3).

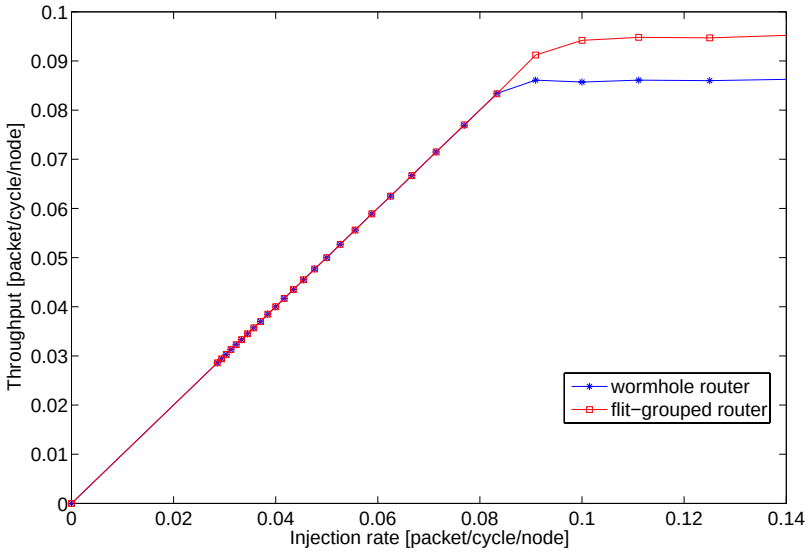


Figure B.6: Throughput comparison (test 3).

## B.3 Performance Comparison in the condition of 'test 4'

Performance comparison figures in the simulation condition of 'test 4' for two networks constructed by the canonical wormhole routers and the flit-grouped routers. In this test, the packet size is 8 and the group size is 4. The arbitration policy is the fixed priority mechanism for both router models.

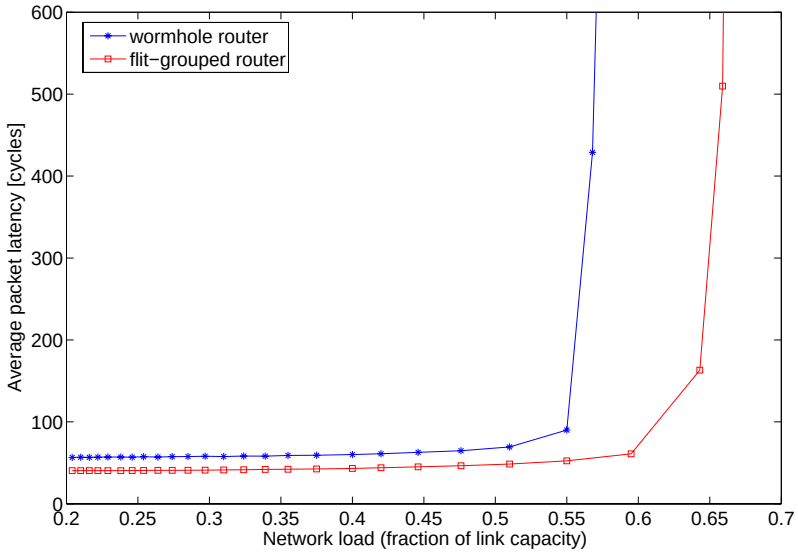


Figure B.7: Latency comparison (test 4).

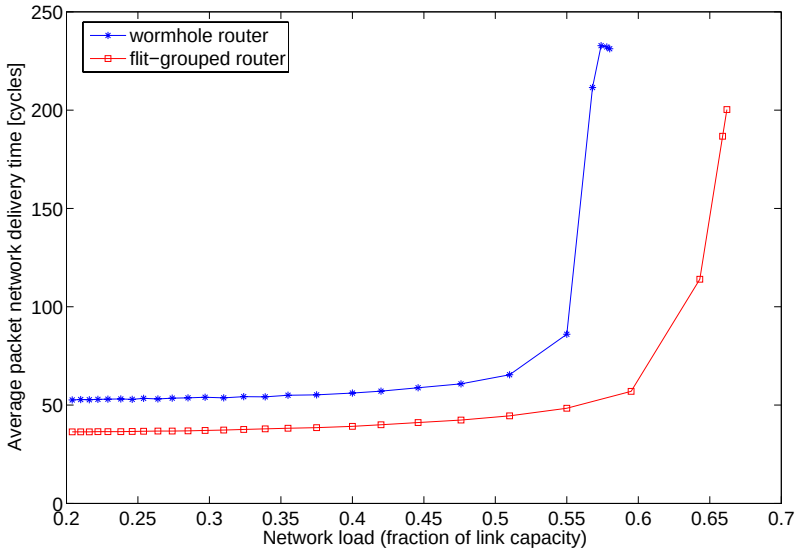


Figure B.8: Network delivery time comparison (test 4).

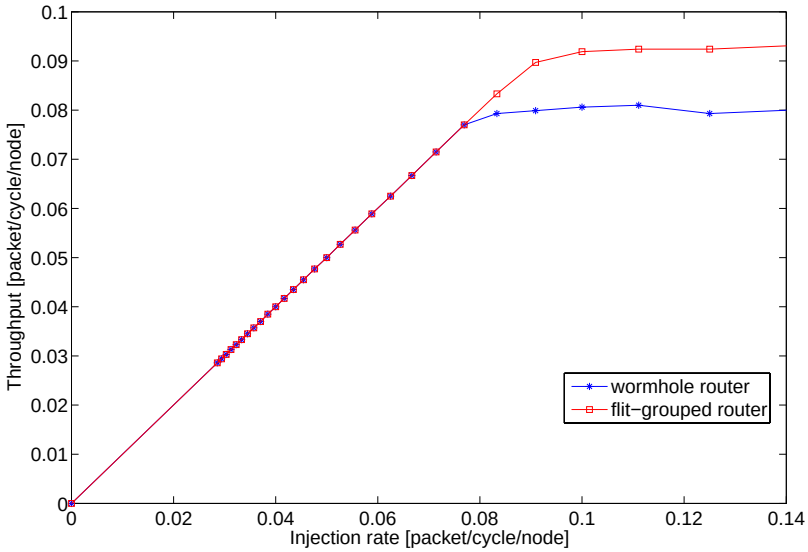


Figure B.9: Throughput comparison (test 4).

## B.4 Performance Comparison in the condition of 'test 5'

Performance comparison figures in the simulation condition of 'test 5' for two networks constructed by the canonical wormhole routers and the flit-grouped routers. In this test, the packet size is 8 and the group size is 4. The arbitration policy is the random priority mechanism for both router models.

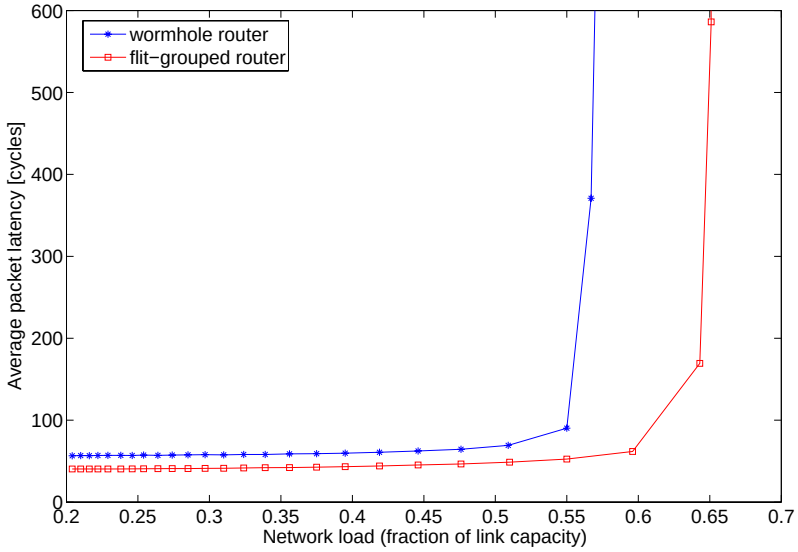


Figure B.10: Latency comparison (test 5).

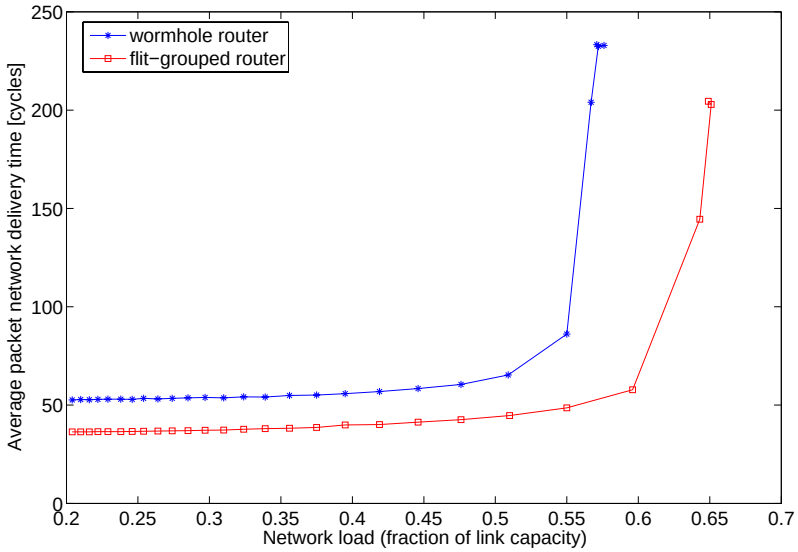


Figure B.11: Network delivery time comparison (test 5).

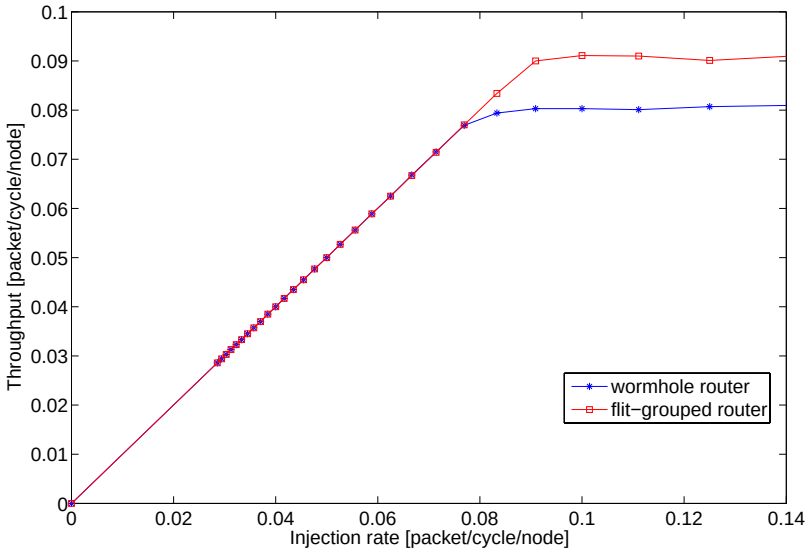


Figure B.12: Throughput comparison (test 5).

## B.5 Performance Comparison in the condition of 'test 6'

Performance comparison figures in the simulation condition of 'test 6' for two networks constructed by the canonical wormhole routers and the flit-grouped routers. In this test, the packet size is 16 and the group size is 8. The arbitration policy is Round-Robin for both router models.

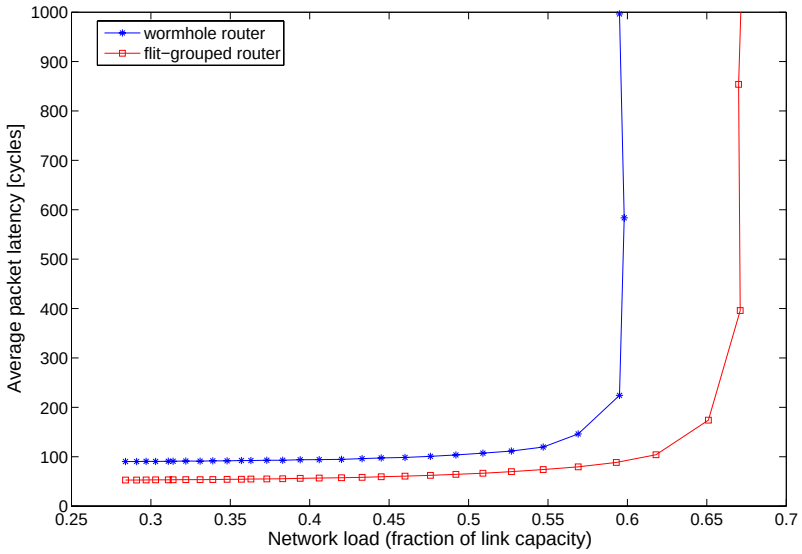


Figure B.13: Latency comparison (test 6).

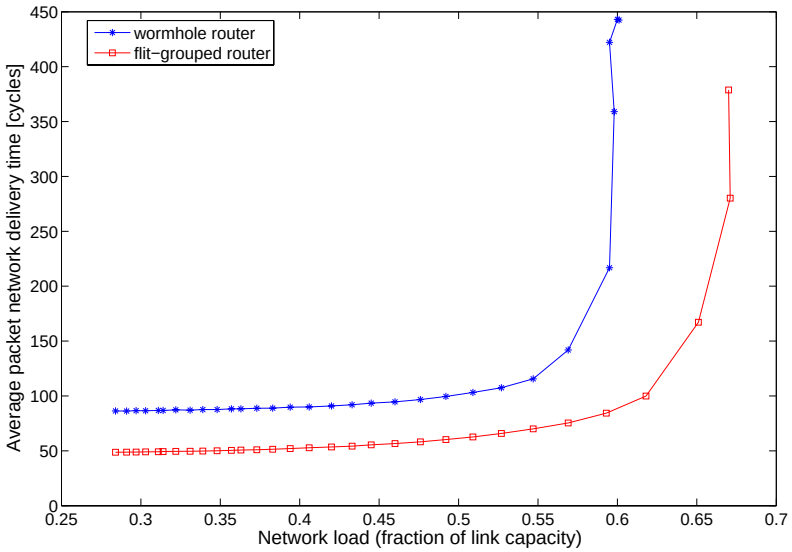


Figure B.14: Network delivery time comparison (test 6).

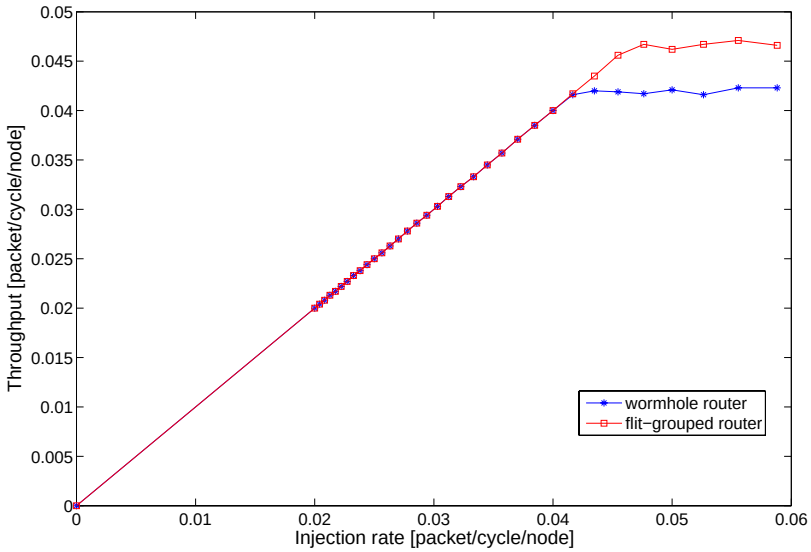


Figure B.15: Throughput comparison (test 6).

# Bibliography


[1] Zhonghai Lu, *Using wormhole Switching for Networks on Chip: Feasibility Analysis and Microarchitecture Adaptation*, Licentiate Thesis, Stockholm, Sweden 2005.

[2] Axel Jantsch and Hannu Tenhunen, *Networks on Chip*, 3-18 Kluwer Academic Publishers, 2003.

[3] W.J.Dally and B.Towles, *Principles and Practices of Interconnection Networks*, ISBN: 0-12-200751-4, Morgan Kaufman Publishers, 2004.

[4] Bei Yin, *Design and Implementation of a Wormhole Router Supporting Multicast for Networks on Chip*, Master Thesis, Stockholm, Sweden 2005, IMIT/LECS/2005-38.

[5] Zhonghai Lu and Axel Jantsch, *Flit Admission in On-chip Wormhole-switched Networks with Virtual Channels*, In Proceedings of the International Symposium on System-on-Chip 2003, November 2004.

[6] Zhonghai Lu and Axel Jantsch, *Flit Ejection in On-chip Wormhole-switched Networks with Virtual Channels*, In Proceedings of the IEEE NorChip Conference, November 2004.

[7] Li-Shiuan Peh and William J. Dally, *A Delay Model for Router Micro-architectures*, In Proceedings of the 8th Symposium on High-Performance Interconnects (Hot Interconnects), Stanford, CA, August 2000.

[8] Li-Shiuan Peh, *Flow Control and Micro-Architectural Mechanisms for Extending the Performance of Interconnection Networks*, Ph.D. Thesis, Stanford University, August 2001.

[9] Synopsys Company, *Design Compiler User Guide*, Version W-2004.12, December 2004.

[10] Synopsys Company, *Guide to HDL Coding Styles for Synthesis*, Version 2000.05, May 2000.

[11] Kevin Skahill, *Optimizing VHDL designs for programmable logic*, Cypress Semiconductor, March 1998.

[12] Dezso Sima, Terence Fountain and Peter Kacsuk, *Advanced Computer Architectures–A Design Space Approach*, ISBN: 981-4053-740, Addison Wesley Longman Limited, 2000.