Application Specific Buffer Allocation for Wormhole Routing Networks-on-Chip

Wang Liwei[1], Cao Yang[1,2], Li Xiaohui[1], Zhu Xiaohu[1]
[1]School of Electronic Information
[2]State Key Laboratory of Software Engineer
Wuhan University
Wuhan, China
wangliwei@mail.whu.edu.cn, caoyang@whu.edu.cn

***Abstract*—A buffer allocation algorithm for wormhole routing networks-on-chip was proposed. When the total budget of the available buffering space is fixed, the proposed algorithm automatically assigns the buffer depth for each input channel, in different routers across the chip, according to the traffic characteristics of the target application. The simulation results show that the buffer allocation result is more reasonable and lower average packet latency can be achieved compared to the uniform buffer allocation.**

I. INTRODUCTION

As technology scales and chip integrity grows, on chip communication is playing an increasing dominant role in System-on-Chip (SoC) design. The NoC approach was proposed as a promising solution to complex intraSoC communication problems[1-3]. It consists of a grid of nodes where each node can be a SoC, an IP, a DSP, etc. Compared to traditional bus interconnection architecture, NoC is much more extensible and parallelizable.

Fig. 1 shows a 4×4 2D mesh NoC which consists of 16 nodes and a typical router architecture. There are buffers at every input channels of router which significantly affect the system performance.

Compared to a computer-network, an on-chip network is much more resource limited. In order to minimize the implementation cost, the interconnection network should be implemented with very little area overhead. In a packet switched network, the route processing logic occupies only a small portion of area(about 6.6%) in each router[2], but the input buffers take a significant one[6], consequently, their size should be carefully minimized.

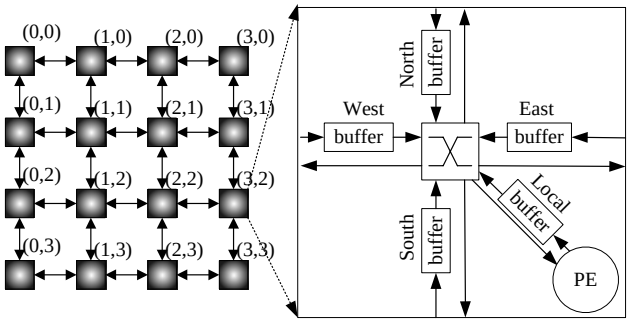

Figure 1. Block diagram of 4X4 mesh-based NoC

Wormhole routing is one of the most popular switching techniques in NoC and it's more suitable for implementing NoCs compared to store-and-forward and virtual cut-through switching[4,5]. In wormhole switching, a data packet (message) is divided into small flits for transmission and flow control. The header flit governs the route of the packet and the reaming data flits follow it in a pipeline fashion. When the header flit is blocked due to contention for output channels or due to insufficient buffer space, all other data flits wait at their current nodes forming a chain of flits that spans over multiple nodes. Wormhole switching makes the end-to-end delay insensitive to the packet destination due to the pipelining of flits, and routers require only small amount of buffer space.

II. RELATED WORK

The input buffer size significantly affects system performance, area and power consumption of NoC. Traditionally, all input channels of every router are assigned with the same amount of buffer. But because of the imbalance of the traffic pattern in most real NoCs, uniform buffer size allocation may not be the most effective way to use the silicon area. To utilize buffering resources more efficiently, one better buffer allocation way is to allocate the buffer size according to the traffic pattern of the target application, allocate more buffering resources only to the heavy loaded channels. Therefore, the buffer allocation problem of NoC can be described as: under total buffering resources constraints, calculate the buffer size of each input channel to minimize the average packet latency of NoC[6].

To solve buffer allocation problem, it is necessary to evaluate the NoC performance. Traditional work on performance evaluation uses either simulation[11] or analytical models[7,8,12,13]. The network simulation method is straight and easy to understand, but time

consuming, while using analytical model is more suitable to solve buffer allocation problem. In[13], a wormhole-based mesh network is modeled as a closed queuing network to calculate the average packet latency, but it can apply only to networks with single-flit buffers. In[7,12], two analytical models are proposed, but it can apply only to networks with infinite buffers. In[8], a analytical model for networks with finite buffers is proposed, but it's constrained to uniform traffic conditions.

In[9-10], the authors present two dynamic buffer allocation methods, their main idea are similar: all buffering resources in a router are placed together as a buffer pool, when there is a data packet enter the input channel, some amount of buffering resources are assigned to store this packet temporally; after the packet leave the router, those buffers are released to buffer pool in case for the next use. Dynamic allocation can not solve buffer allocation problem if the size of buffer pool is not big enough.

In[6], the authors describe and provide an efficient way to solve the buffer allocation problem for NoC designs which use store-and-forward or virtual cut-through switching for the first time based on the queuing theory. But in most NoC designs, the wormhole switching is more suitable. In this paper, we study the buffer allocation in wormhole routing NoCs.

III. BUFFER ALLOCATION ALGORITHM

First, a wormhole router analytical model is presented. Using this model, the system performance bottleneck among the different channels can be detected. In this paper, performance bottleneck is defined as such an input channel which owns the FIFO that has the highest probability to be "full". After finding the system performance bottleneck, the buffer allocation algorithm iteratively adds extra buffering space to the bottleneck channels until the total buffer budget is reached, which leads to the maximum improvement in performance.

A. The wormhole router analytical model

The model in this paper is based on the following assumptions, which are commonly used in similar studies[6-8,12,13].

i. Nodes generate traffic independently of each other, and according to a Poisson process.

ii. NoC uses XY routing algorithm.

iii. Packet length is fixed at M flits. Each flit takes one cycle to advance from one to the next. Buffer width equals to the bit widths of a flit.

iv. The local queue at the injection channel in the source node has infinite capacity; moreover, packets are transferred to the local PE as soon as they arrive at their destinations through the ejection channel.

The basic parameters used in this paper are summarized in Table I.

TABLE I. PARAMETER NOTATION

Param.	Description
M	The size of a data packet (M flits)
T_H	The time needed by a router to process the header flit
dir	Direction, i.e.: *North, East, South, West, Local*
$b_{x,y,dir}$	The probability of the buffer at *dir* input channel of *Node(x,y)* being full
$\rho_{x,y,dir}$	The utilization factor of *dir* input channel at *Node(x,y)*
$\lambda_{x,y,dir}$	The packet arrival rate at *dir* input channel of *Node(x,y)*
$\mu_{x,y,dir}$	The packet service rate at *dir* input channel of *Node(x,y)*
$l_{x,y,dir}$	The buffer size of *(x,y) dir* input channel at *Node(x,y)*
$a_{x,y}$	The packet injection rate of *Node(x,y)*
$d_{(x,y)(x',y')}$	The probability of a packet generated by *Node(x,y)* to be delivered to *Node(x',y')*
$T_{x,y,dir}$	The packet service time at *dir* input channel of *Node(x,y)*
$TB_{x,y,dir}$	The blocking delay at *dir* input channel of *Node(x,y)*
$\theta_{x,y,dir}$	The probability of a packet get blocked at the head of *dir* input channel of *Node(x,y)*
$\omega_{x,y,dir}$	The mean waiting time that a packet needs to wait in the event of blocking
$\lambda_{x,y,dir\to dir'}$	The packet arrival rate which is forwarded from *dir* input channel to *dir'* input channel of *Node(x,y)*

Resorting to the theory of finite queuing networks[15], every input channel buffer can be modeled as a M/M/1/K finite queue. Therefore, the probability of the buffer at *dir* input channel of *Node(x,y)* being full can be calculated using the following equations:

$$b_{x,y,dir} = \frac{1-\rho_{x,y,dir}}{1-\rho_{x,y,dir}^{\,l_{x,y,dir}+1}} \times \rho_{x,y,dir}^{\,l_{x,y,dir}} \quad (1)$$

$$\rho_{x,y,dir} = \frac{\lambda_{x,y,dir}}{\mu_{x,y,dir}} = \lambda_{x,y,dir} \times T_{x,y,dir} \quad (2)$$

The packet arrival rate at *dir* input channel of *Node(x,y)* can be calculated with the following equation:

$$\lambda_{x,y,dir} = \sum_{\forall j,k} \sum_{\forall j',k'} a_{j,k} \times d_{(j,k)(j',k')} \times \Re(j,k,j',k',x,y,dir) \quad (3)$$

In (3), the $\Re(j,k,j',k',x,y,dir)$ is the routing function, it equals 1 if the packet from *Source Node(j,k)* to *Destination Node(j',k')* uses the *dir* channel of *Node(x,y)*; it equals 0 otherwise. In this paper, we use XY deterministic routing algorithm, it's easy to calculate $\lambda_{x,y,dir}$ according to the Eq.(3).

B. Calculation of the packet service time

The packet service time at *dir* channel of *Node(x,y)* is given by

$$T_{x,y,dir} = T_H + M + TB_{x,y,dir} \quad (4)$$

In (4), $T_H + M$ represents the service time per packet in a router without contention; when there is a contention, the waiting time that a packet needs to wait in a buffer can be modeled by $TB_{x,y,dir}$. In[8], the authors present a method to calculate this blocking delay:

$$TB_{x,y,dir} = \theta_{x,y,dir} \times \omega_{x,y,dir} \quad (5)$$

In (5), $\theta_{x,y,dir}$ represents the probability that a packet may get blocked at the head of the buffer due to the contention for the same output channel; $\omega_{x,y,dir}$ represent the mean waiting time that a message needs to wait in the event of blocking.

C. Calculation of the mean waiting time

To determine the mean waiting time, each router is treated as an M/M/1 queuing system where the arrival rate is $\lambda_{x,y,dir}$, the packet service time is $T_{x,y,dir}$. Resorting to the queuing theory[15], the mean waiting time becomes

$$\omega_{x,y,dir} = \frac{\lambda_{x,y,dir}}{\mu_{x,y,dir}(\mu_{x,y,dir} - \lambda_{x,y,dir})} = \frac{\lambda_{x,y,dir} \cdot T^2_{x,y,dir}}{1 - \lambda_{x,y,dir} \cdot T_{x,y,dir}} \quad (6)$$

D. Calculation of the blocking probability

For convenience, we use θ_{dir} to represent $\theta_{x,y,dir}$. First, we have to calculate the Forwarding Probability Matrix(FPM) $\boldsymbol{F}$

$$F = \begin{bmatrix} 0 & f_{NE} & f_{NS} & f_{NW} & f_{NL} \\ f_{EN} & 0 & f_{ES} & f_{EW} & f_{EL} \\ f_{SN} & f_{SE} & 0 & f_{SW} & f_{SL} \\ f_{WN} & f_{WE} & f_{WS} & 0 & f_{WL} \\ f_{LN} & f_{LE} & f_{LS} & f_{LW} & 0 \end{bmatrix}, \text{where} \quad f_{ij} = \frac{\lambda_{i \to j}}{\lambda_i} \quad (7)$$

In (7), f_{ij} is the probability that a packet arrives at *dir i* and leaves the router through *dir j*. Every elements of the forwarding probability matrix $\boldsymbol{F}$ can be calculated using Equation (3).

Now let us calculate the blocking probability and use North direction as an example.

$$\theta_N = \sum_{\forall Dir} f_{NDir} \cdot \theta_{NDir} \quad (8)$$

In (8), f_{NDir} is the forwarding probability of $N \to Dir$, while θ_{NDir} is the probability that a packet forwarded from *N* to *Dir* may get blocked.

$$\theta_{NDir} = f_{NDir} \cdot \sum_{\forall Dir', Dir' \neq N} f_{Dir'Dir} \quad (9)$$

Combining Eq.(7), Eq.(8) and Eq.(9) together, $\theta_{x,y,dir}$ can be calculated. Combining $\theta_{x,y,dir}$, Eq.(6), Eq.(5) and Eq.(4), we can finally build a nonlinear equation about $T_{x,y,dir}$, this equation can be solved to determine $T_{x,y,dir}$. Combining $T_{x,y,dir}$, Eq.(1) and Eq.(2), we can calculate $b_{x,y,dir}$ and detect the system performance bottleneck.

E. The buffer allocation algorithm

In this paper, we propose a greedy algorithm to solve this problem based on the aforementioned analytical model. The flow of the algorithm is shown in Fig. 2.

1) Given the system parameters(such as T_H, M, size of Mesh and total buffer budget) and traffic parameters(such as $a_{x,y}$ and $d_{(x,y)(x',y')}$), the algorithm (written in C++) can calculate $\lambda_{x,y,dir}$, $\boldsymbol{F}$ and $\theta_{x,y,dir}$ respectively using Eq.(3), Eq.(7) and Eq.(8) and build a nonlinear equation about $T_{x,y,dir}$.

2) The *Matlab Math Library* is used to solve the given equation(more specifically, the `roots` utility from *Matlab* is used as the nonlinear equation solver); at the same time the algorithm also generates the initial buffer configuration which assigns the buffer in all the used channels ($\lambda_{x,y,dir} \neq 0$) to be one flit large.

3) The algorithm determine $b_{x,y,dir}$ for each input channel using Eq.(1) and select the channel with the largest $b_{x,y,dir}$ as the system performance bottleneck.

4) The buffer size of the bottleneck channel is incremented by one flit, while the system free buffer is decremented by one flit.

5) Repeat (3)~(4), until system free buffer is 0.

6) Output the final buffer allocation results.

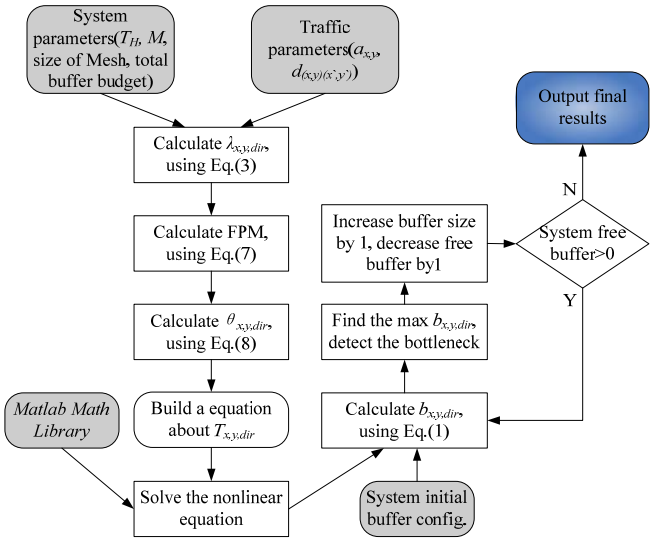

Figure 2. The buffer allocation algorithm flow

IV. EXPERIMENT RESULT

To validate the proposed algorithm, we build a wormhole routing NoC simulation platform with OMNet++[14]. OMNet++ is a public source, generic and

flexible simulation environment. It allows a fast and high-level simulation environment for NoC exploration.

Some parameters used in simulation are: $M = 16$, a data packet is divided into 16 flits; $T_H = 2$, the router needs 2 clock cycle to make routing decision for the header flit; the size of Mesh is 4, NoC is 4×4 2D Mesh; system total buffers are 240 flits, which means every used channel is assigned with 5 flits large buffer. For simplicity, every node has the same packet injection rate.

In the experiments, we applied our algorithm to applications with two typical traffic models: uniform random traffic model and hotspot traffic model. Under the uniform traffic pattern, a PE sends a packet to any other node with equal probability (this probability is 1/15 ≈ 0.0666). Under hot spot traffic pattern, one or more nodes are chosen as hot spots which receive an extra proportion of traffic (in this paper, the probability is 0.2) in addition to the regular uniform traffic. The efficiency of the algorithm is evaluated through latency-throughput curves. The average packet delay is defined as the mean amount of time from the generation of a packet until the last data flit reaches the local PE at the destination node.

Three hotspot traffic patterns used in this evaluation are hotspot1, hotspot2 and hotspot3. Traffic pattern hotspot1 have only one hot spot, while hotspot2 and hotspot3 are the hot spot traffic patterns which have two and three hot spots respectively. In each simulation experiment, a total number of about 80000 packets are delivered to their destinations. To avoid the distortions due to start-up conditions, the first 10000 packets are ignored.

In simulation results, NoCs with uniformly allocated FIFO buffers are denoted by UNOC, and systems that customized by our buffer allocation algorithm are denoted by CNOC. Their average packet latency should be compared.

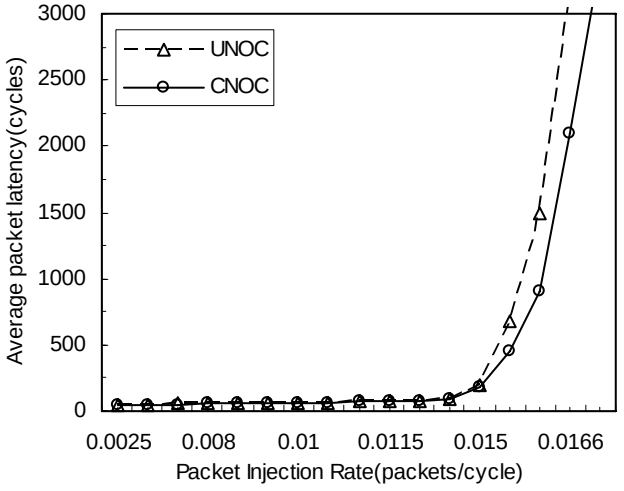

Figure 3. Performance under uniform traffic

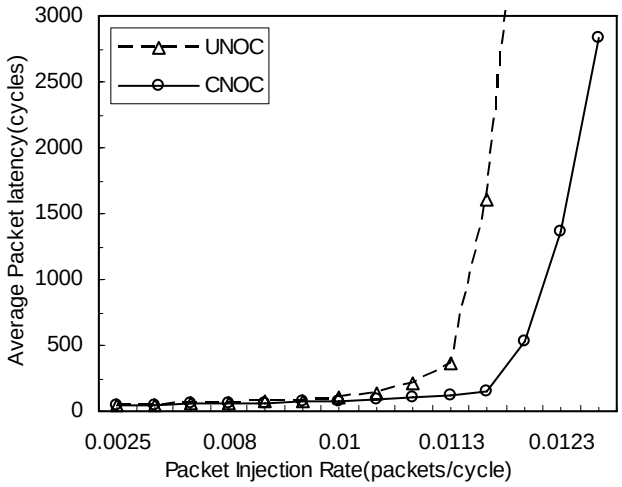

Figure 4. Performance under hotspot1 traffic

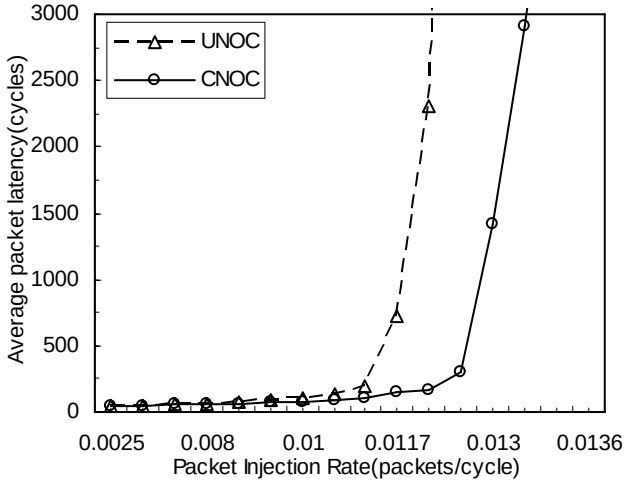

Figure 5. Performance under hotspot2 traffic

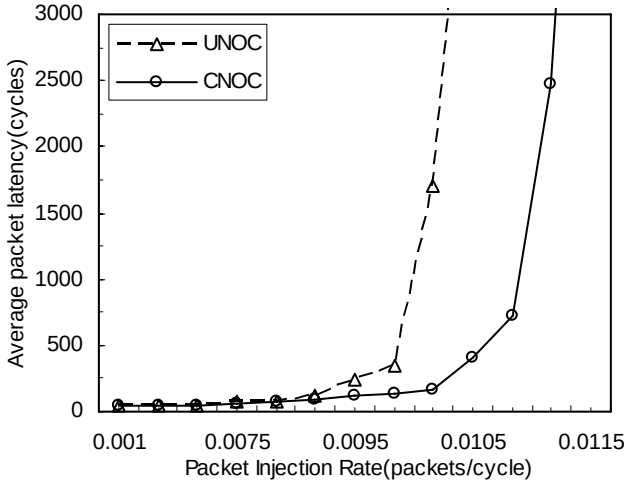

Figure 6. Performance under hotspot3 traffic

Fig. (3) shows the performance of UNOC and CNOC under the uniform traffic. The X-axis represents the packet injection rate per node, and the Y-axis represents the average packet latency of the NoC. In Fig. (3), we can see that, when packet injection rate is low(< 0.015 packets/cycle) their performance almost the same, but CNOC is better than UNOC. As the packet injection rate increases, network congestion happens in both UNOC and CNOC, and the packet latency rises dramatically, but the time that congestion happens in CNOC is later than UNOC. CNOC performs significantly better than UNOC when the packet injection rate is the same. This is because after customized by our buffer allocation algorithm, CNOC is assigned more buffering resources at performance bottleneck channels than UNOC, this customization makes the network average packet latency of CNOC is lower than UNOC.

Fig. (4) shows the performance of UNOC and CNOC under the hotspot1 traffic. Traffic pattern hotspot1 have only one hot spot, which is located at *Node(0,1)*. When packet injection rate is low(<0.015packets/cycle) their performance almost the same, but CNOC is better than UNOC. As the packet injection rate increases, congestion happens in UNOC at 0.0113 packets/cycle, while happens in CNOC at 0.012 packets/cycle. At any packet injection rate, the CNOC performs better than the UNOC.

Fig. (5) and Fig. (6) show the performance of UNOC and CNOC under the hotspot2 and hotspot3 traffic. Traffic pattern hotspot2 have two hot spots, which are located at *Node(2,1)* and *Node(0,2)*. Traffic pattern hotspot3 have three hot spots, which are located at *Node(1,1)*, *Node(2,2)* and *Node(1,3)*. The CNOC also performs better than the UNOC under these two traffic patterns, the results confirm the system behavior and conclusions discussed for uniform and hotspot1 traffic.

In all cases, CNOC performs much better than UNOC, which validate our buffer allocation algorithm. Comparing Fig. (6) with Fig. (3), it is clear that the buffer allocation is much more beneficial for hotspot3 traffic. The reason is compared to the uniform traffic pattern, the hotspot3 traffic pattern is more unbalanced, which makes the results of buffer allocation much better.

V. CONCLUSION

In order to minimize the implementation cost and maximum system performance, it's necessary to carefully allocate buffering resources to all used channel of NoC. A buffer allocation greedy algorithm for wormhole routing NoC is proposed in this paper, which can automatically assigns the buffer depth for each input channel, in different routers, according to the traffic characteristics of the target application. The simulation results show that the NoC customized by our algorithm has lower average packet latency than those whose buffering resources are uniformly allocated. Extending this research to NoCs that support adaptive routing and virtual channels will be our future work.

REFERENCES

[1] L. Benini and G. D. Micheli, "Networks on chips: a new soc paradigm", Computer, Vol.35, pp.70-78, 2002.

[2] W. J. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks" in Proc. Design Automation Conference, Las Vegas, USA, pp.684-689, June 18-22, 2001.

[3] S. Kumar, A. Jantsch, J. P. Soininen, et al, "A network on chip architecture and design methodology" in Proc. IEEE Symposium on VLSI, Pittsburgh, USA, pp.117-124, Apri. 25-26, 2002.

[4] T. Bjerregaard and S. Mahadevan, "A survey of research and pratices of network-on-chip", ACM Computing Surveys, Vol.38, pp.71-121, 2006

[5] L. M. Ni and P. K. McKinley, "A survey of wormhole routing techniques in direct networks", IEEE Tran. on Computers, Vol.26, pp.62-76, 1993.

[6] J. Hu, U. Y. Ogras and R. Marculescu, "System-level buffer allocation for application-specific networks-on-chip router design", IEEE Tran. on Computer-Aided Design Of Integrated Circuits And Systems, Vol.25, pp.2919-2933, 2006.

[7] U. Y. Ogras and R. Marculescu, "Analytical router modeling for networks-on-chip performance analysis" in Proc. Design, Automation and Test in Europe, Nice, France, pp.1-6, 2007

[8] N. Alzeidi, M. Ould-Khaoua, L. M. Mackenzie, et al, "Performance analysis of adaptively-routed wormhole-switched networks with finite buffers" in Proc. IEEE International Conference on Comunication, Glasgow, Scotland, pp.38-43, June 24-28, 2007.

[9] R. Dobkin, R. Ginosar and I. Cidon, "QNoC asynchronous router with dynamic virtual channel allocation" in Proc. First International Symposium on Networks-on-Chip, New Jersey, USA, pp.218-218, May. 7-9, 2007.

[10] Y. Tamir and G. L. Frazier, "Dynamically-allocated multi-queue buffers for VLSI communication switches", IEEE Tran on Computers, Vol.41, pp.725-737, 1992.

[11] G. M. Chiu, "The odd-even turn model for adaptive routing", IEEE Tran. on Parallel and Distributed Systems, Vol.11, pp.729-738, 2000.

[12] W. J. Dally, "Performance analysis of k-ary n-cube interconnection networks", IEEE Tran. on Computers, Vol.39, ppl775-785, 1990.

[13] V. S. Adve and M. K. Vernon, "Performance analysis of mesh interconnection networks with deterministic routing", IEEE Tran. on Parallel and Distributed Systems, Vol.5, pp.225-246, 1994.

[14] OMNet++ discrete event simulation system user manual, http://www.omnetpp.org/, 2005.

[15] F. Hillier and G. Lieberman, Introduction to operations research. NY: McGraw-Hill Companies, 2002.