

Performance Analysis of a High-speed Dynamically Reconfigurable LAN

Saad AlKasabi and Salim Hariri
Electrical and Computer Engineering
Syracuse University
Syracuse, NY 13244
alkasabi@cat.syr.edu

Abstract

In this paper, we present the design of a dynamically reconfigurable switch that will be used to build a high-speed multi-link ring local area network. Using FPGAs technology, switch reconfigurability can be exploited to implement different interconnection topologies and support different application requirements. We present two approaches to analyze the multi-link ring network performance. In the first approach, we develop an analytical model that uses the M/M/n and M/D/n queuing systems to study the virtual channels access delay. In this approach, the virtual channel occupancy probabilities are found using an infinite state Markov model. In the second approach, we introduce an analytical model based on a finite state Markov model developed for analyzing networks with virtual channels flow control. Simulation, using the OPNET tool, indicates that the second approach is more accurate in analyzing the behavior of the packet transfer time. Furthermore our performance analysis shows that the use of *wormhole* routing and virtual channel flow control improves the system throughput and decreases the packet transfer time.

1 Introduction

Over the last decade, there have been significant advancements in computer communications and processing systems that have increased the processing and transmission rates. The use of fiber optics technology offers data transmission rates that are several order of magnitudes higher than the current rates. The developments of programmable hardware have open the door to the implementation of reconfigurable devices that are more efficient and have greater flexibility than the ones with fixed topologies. In this paper, we present the design of an FPGA-based reconfigurable switch and show how it can be used to build a multi-link ring network that can embed several important topologies (e.g. Hypercube and Mesh).

Parallel algorithms run more efficiently on parallel

computers when their communication requirements map naturally to the communication structure of the interconnection network. There is no general interconnection topology suitable for all applications in a parallel/distributed computing system. Attaining more versatility in such an environment suggests the incorporation of different topologies requested by various applications. The design of reconfigurable switches will provide the means to connect the resources of an interconnection network using the best topology for solving a given problem.

Embedding different topologies and reconfigurability of interconnection networks have been addressed by many researchers recently [1, 2, 3, 4, 5]. The Hnet (High-performance Network Evaluation Testbed) system is constructed to support any topology of degree six or less. The Hnet system can be viewed as a tool to better understand the complex interactions between parallel applications and the underlying interconnection network of parallel systems [1]. In this system, a processor capable of executing parallel programs is connected to each of the 64 user configurable switches. However, due to the complexity of the switch nodes, the cost of such a design is too expensive to be adopted for many LAN implementations. Other approaches were presented in [3, 4]. The limitations of these approaches are the complexity in the physical interconnection and the complication involved in routing.

Reconfigurable networks are becoming attractive due to the recent advances in programmable hardware (e.g. Field Programmable Gate Arrays and Field Programmable Interconnect Components) and high-speed communication links. In this paper, we propose a design of a reconfigurable switch for high-speed networks. Our approach to design the switch is aimed at achieving simplicity and parallel processing of messages. The simple design reduces the overhead of routing messages as well as its processing time. To support parallel routing of messages, a distributed control strategy is used. If the node is the source of a message, the switch controller

determines the routing scheme and the number of hops the message needs to reach its destination. Whenever an incoming message passes through the switch, a destination check, to determine whether or not its hop count equals zero, is performed without interrupting other processing activities. If the hop count is found to be zero, the message is transferred to the node. Otherwise, the message is passed to the next node after its hop count is decremented by one.

In ring networks the time it takes a data unit to reach its destination (*latency*) is not only a function of the propagation delay on the links, but also of the store and forward delay at the intermediate nodes [6]. Our design uses *wormhole routing* [5] and *virtual channel flow control* [7] to reduce the message latency and increase the communication bandwidth. Besides avoiding buffering the message at intermediate nodes, wormhole routing and its flow control reduce the message latency caused by long distances in the network [8]. Virtual channels improve the efficiency of the resource allocation scheme and can be exploited to improve network scalability [7]. We also use multiple links (channels) between adjacent nodes to allow for concurrent packet transmission and to overcome the problem of having the network throughput severely restricted by the link data rate [9].

The organization of this paper is as follows: In Section 2 we describe the design of the proposed reconfigurable switch and its mode of operation. In Section 3, we discuss how to use the switch to build a multi-link ring network. Performance analysis of the multi-link ring network will be presented in Section 4. Finally, the conclusion and future work are presented in Section 5.

2 Reconfigurable Switch Block Diagram

In any computer interconnection network, the switch is a critical element in reducing the packet latency. A low latency and high bandwidth can be attained by building a dedicated hardware switch that has the ability to route packets without the involvement of the node's general-purpose processor [10].

As depicted in Figure 1, the main components of the switch include a Switch Controller (SC), a Transfer unit, a Receive unit, a memory unit (RAM), a Processing Element (PE), and a Router (see [11] for the detailed description of the switch components). We use two independent unidirectional links with data flowing in opposite directions for each bidirectional link connecting adjacent nodes. Having two independent unidirectional links to support bidirectional communication allows data to flow in opposite directions simultaneously and breaks any cyclic dependency that might occur in message routing. Therefore, deadlock caused by such dependency is prevented. Also, this approach is more reliable since a link failure affects only one direction. The main func-

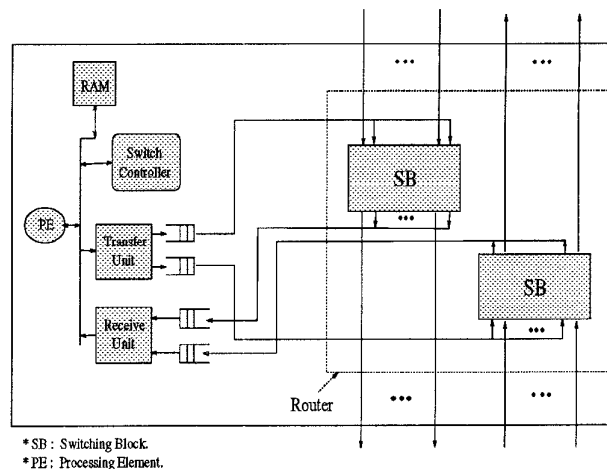


Figure 1: Switch Block Diagram

tions performed by the **Router** unit can be outlined as follows:

- Transfer messages from the network to the node and vice versa.
- Pass messages destined to other nodes without interrupting other switch components.
- Resolve any conflict between incoming messages in their route to their destinations.
- Guarantee a fault free flit level flow control between adjacent nodes.

The switch reconfiguration can be attained by reconfiguring the router unit to the new switching requirements (see [11] for more details). This process takes a short time (order of microseconds using Xilinx FPGAs [12]); it is the time required to load the new configuration program.

3 Multi-link Ring Network

In this section we discuss how the proposed switch is used to build a multi-link ring network (see Figure 2). In this environment, when a topology request is received at the application layer, it is passed to the virtual topology layer where the request is processed. It is granted if the topology is supported by the network. In this case, the check for circuit/packet switching communication is performed and the message transfer starts accordingly.

The multi-link ring network consists of N nodes (where N is a multiple of four), and multiple logical bidirectional links connecting adjacent nodes. The nodes are grouped into $l = N/4$ building blocks of four nodes each such that the number of hops between neighbor

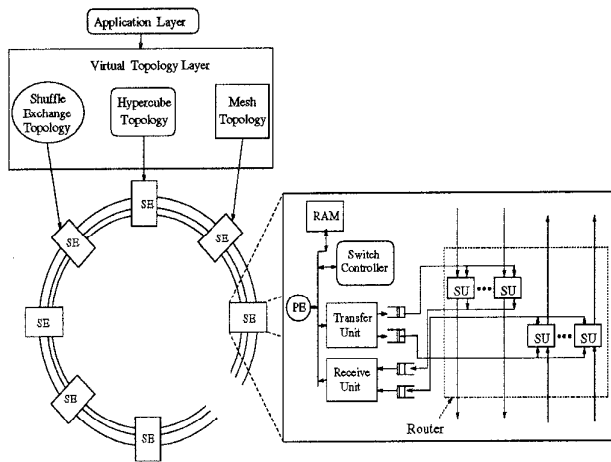


Figure 2: General View of the Multi-link Ring Network

nodes that belong to the same block is l [11]. Each node has a SE that supports communication between all processing elements (PEs) attached to it. The SBs are constructed from a group of **Switching Units** (SUs) that perform parallel message routing. Each SU (detailed description of the SU architecture is given in [11]) routes a flit from an input link i to an output link i ; the links construct multiple unidirectional rings. Every link passes its data to a node through the SE which in its turn decides if that data is to be received or passed to a subsequent node.

We use *virtual-channel* based flow control to effectively utilize the physical links bandwidth and its memory resources [7], and prevent deadlock. Implementing each channel in a wormhole-routed network with a separate set of physical wires is expensive in terms of the number of links needed to construct such network. Furthermore the channel utilization is not high in most applications. One way to address this problem is to multiplex several virtual channels on a single physical link [5]. The overhead incurred is a small amount of additional control logic.

For *packet switching* communication, the implementation of *wormhole* routing [13] is adopted to reduce the communication latency of this model. In such routing strategy, messages are split into fixed small size flits (this format is similar to the ATM fixed size packets or "cells"). Each flit has a control information that contains its type (*header*, *tail*, or *intermediate*), its hop count, and the virtual channel on which it is sent. The header flit sets the path to the destination node by reserving a free virtual channel in every stage in its route to the destination. If in any stage that flit is blocked because all channels are occupied, all subsequent flits are also blocked at the nodes where they reside and prevented

from advancing. The hop count is initialized with the number of hops needed to reach the destination node and decremented by one whenever the flit passes through an intermediate node.

4 Performance Analysis

In this section, we develop analytical performance models for the proposed multi-link ring design with wormhole routing and virtual channel flow control. We present two approaches to analyze the system performance. In the first, we use the M/M/n and M/D/n queuing systems to study the virtual channels access delay; the virtual channel occupancy probabilities are found using an infinite state Markove model. The second approach is based on a previous work developed for analyzing networks with virtual channels flow control and where finite state Markove model is used to analyze such occupancy probabilities. We analyze the packet transfer time (latency) and the node throughput for different number of virtual channels, network sizes, packet sizes, and routing schemes. The following is a brief description of the notations used in our analysis :

λ_n (λ_c): Node (Channel) packet arrival rate.

l : Number of building blocks.

v_c : Number of virtual channels per link.

p_d^i : Probability of delay in the virtual channel request queue at stage i .

b_j^i : Probability that all j occupied virtual channels are blocked at stage i .

p_j^i : Probability that j virtual channels are occupied at stage i .

p : Probability of passing a message to next node in the ring ($p = 1 - q$).

pkt_i ($flit_i$): Packet (Flit) length in bits.

C_L : Physical link capacity (bits/sec).

H_{avg} : Average number of hops.

t_{sch} : Virtual channel scheduling time.

t_{hpc} : Packet processing time.

t_s^i : Service time at stage i .

t_j^i : Effective service time at stage i when j virtual channels are occupied.

W_i : Average waiting time spent acquiring virtual channels between stage i and the destination.

T : Packet transfer time.

In wormhole routing, the packet transfer time, T is determined as the time spent by the header flit to reach the destination node plus the time required to transfer the rest of the packet. To simplify the analysis, we assume that the packet traverses an average number of nodes (stages), H_{avg} , to reach its destination¹. We model the node using the queuing model shown in Figure 3 and use the following assumptions:

1. Packet destinations are uniformly distributed.
2. Packets are generated by a Poisson process with rate λ_n . Therefore, packets arrival rate in each direction is $\lambda_n/2$.
3. All packets are of length pkt_l .
4. Physical channel blocking probabilities are independent.
5. A packet that arrives at its destination is consumed without waiting, i.e. $t_s^0 = \frac{1}{\mu} = (pkt_l/w)$, where w is the channel capacity.

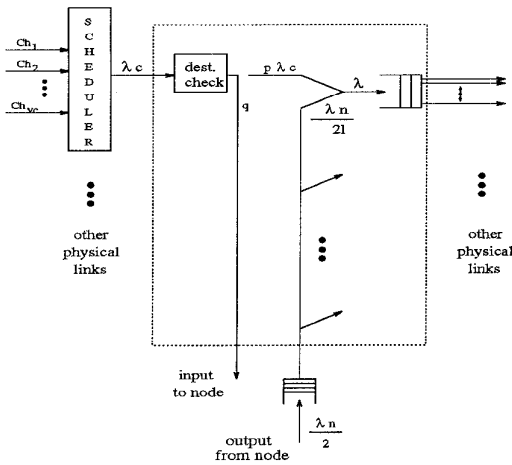


Figure 3: Node Model

As shown in Figure 3, the arrival rate at the input of any link, λ , is the summation of the transient traffic rate sent to other nodes through that link, $p\lambda_c$, plus the traffic rate generated by that node and routed through the same link, $\lambda_n/2l$. Hence, λ is given by the following equation:

$$\lambda = \frac{\lambda_n}{2l} + p\lambda_c \quad (1)$$

We fix the routing strategy and assume that there are l links between adjacent nodes. The links, as well as the four-node rings, are numbered from 0 to $l-1$. Without

¹ H_{avg} is rounded to the nearest integer.

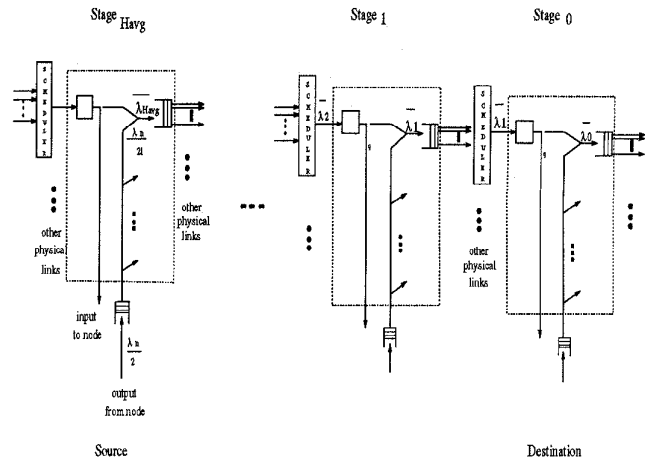


Figure 4: Message Queuing Model

loss of generality, we focus our analysis on calculating the average delay of packets sent through one direction of one ring, k . The same approach can be used to find the delay of packets sent through other rings. The routing strategy we use is to send messages from any node in the network to a node in ring j through link j only. Hence, the arrival rate at link k in stage i , $\bar{\lambda}_i$, can be found using the following formula:

$$\bar{\lambda}_i = \frac{\lambda_n}{4l-1} ((3l-1) - 2(i-1)) \quad 0 < i \leq H_{avg} \quad (2)$$

The first term, $\frac{\lambda_n}{4l-1}(3l-1)$, represents the arrival rate from all nodes that use link k in routing messages to their correspondent destinations in ring k ($\frac{\lambda_n}{4l-1}$ is the packet rate sent by a node to each of the remaining $N-1$ nodes). The second term, $\frac{\lambda_n}{4l-1} 2(i-1)$, is the share of the nodes in subsequent stages in that rate; it is subtracted because such nodes do not compete for virtual channels sharing link k at stage i .

The main obstacle in analyzing the performance of the proposed design is the analysis of the virtual channel access delay at each node in the packet (flits) path. We started by modeling the average delay a packet incurs until it reaches its destination by the multiple queues in Tandem as shown in Figure 4. Stage H_{avg} represents the source node, while stage 0 denotes the destination node. Every hop takes the packet from stag i to stage $i-1$. To study the impact of the service time distribution on the packet latency, we used M/M/n and M/D/n queuing systems to analyze the virtual channels queuing delay in each of the subsequent stages.

4.1 Infinite State Markove Model

4.1.1 M/M/n Queuing model

If we assume circuit switching network and M/M/n model with infinite queue and $n = v_c$ (in this section, v_c represents the *average* number of virtual channels sharing the same physical link), the probability that a header flit will be delayed, at any stage i , until it gets access to a virtual channel can be computed as [14]:

$$p_d^i = \frac{(\rho_i v_c)^{v_c} p_0^i}{(1 - \rho_i) v_c!} \quad (3)$$

Where p_0^i is the probability that there were no header flits in the queuing system at stage i and it is given by

$$p_0^i = \left(\sum_{j=0}^{v_c-1} \frac{(\rho_i v_c)^j}{j!} + \frac{1}{1 - \rho_i} \frac{(v_c \rho_i)^{v_c}}{v_c!} \right)^{-1} \quad (4)$$

where $\rho_i = \frac{\lambda_i}{\mu_i v_c}$, and $\mu_i = 1/t_s^i$ is the service rate seen by the packet at stage i . The service time of a header flit at stage i is the summation of the time it takes to service that flit through the queue in the subsequent stage, $i-1$ (this includes waiting plus service time), the hop count processing time, virtual channel scheduling time (under the assumption that we have a processor of 25 MIPs per second and that the hop count processing and the virtual channels scheduling takes 5 instructions each, t_{hcp} and t_{sch} are approximated to 200 ns), and the flit transfer time through that channel ($t_{flt} = flt_l/w$). Hence, we can write

$$t_s^i = t_s^{i-1} + E(W)_{i-1} + t_{hcp} + t_{sch} + t_{flt} \quad 0 < i \leq H_{avg} \quad (5)$$

The first two terms represent the delay a header flit incurs in the subsequent stage to reach the destination node, while the sum of the other terms represents the delay it takes to reach that stage. If all virtual channels are being occupied at stage i , the header flit waits (blocked) in the queue until one of the packets reserving the virtual channels passes that stage and releases its channel. Then, the blocked flit reserves that channel for t_s^i unit times². $E(W)_i$ is the average waiting time the header flit spends at stage i to access a free virtual channel and is given by

$$E(W)_i = \begin{cases} t_s^i & \frac{p_d^i}{v_c(1-\rho_i)} & : & 0 < i \leq H_{avg} \\ 0 & & : & i = 0 \end{cases} \quad (6)$$

Every packet travels from a source node at stage H_{avg} to its destination at stage 0. If we assume that the virtual channels buffers are of size one (flit), the service

²At stage 1, the header flit waiting time, $E(W)_1$, is found assuming a queue with service time t_s^0 . However, the flit service time which will be substituted in Eq. (5) is just t_{flt} .

time seen by the header flits at the source node virtual channels queue, \bar{t}_s , is

$$\bar{t}_s = E(W)_{H_{avg}} + t_s^{H_{avg}} + \left(\frac{pkt_l}{flt_l} - 1 \right) t_{flt} \quad (7)$$

It is the delay the header flit takes to reach the destination plus the time required to receive the rest of the flits. Finally, we consider the time delay at the source node segmentation queues. If we assume that there are only two queues; each stores flits belong to packets sent in the same direction (left or right), then the packet transfer time, T , can be found by:

$$T = \frac{\bar{t}_s}{(1 - \rho)} \left(1 - \frac{\rho}{2} \right) \quad (8)$$

This is the queuing delay of a M/D/1 queuing system [14] with average service time \bar{t}_s and utilization $\rho = \frac{\lambda}{2} \bar{t}_s$.

4.1.2 M/D/n Queuing model

Here, we assume M/D/n instead of the M/M/n queuing model for each of the virtual channel queues of the subsequent stages. The analysis of a M/D/n type queuing system characterized by random arrivals, fixed service time, and n servers is quite complex. However, such analysis can be simplified if we use the same assumptions presented in [15] to find the packet delay in such a queuing system. Considering that the time in service at stage i , t_q^i , is the sum of the waiting and service times, we can write

$$t_q^i = t_q^{i-1} + t_{hcp} + t_{sch} + t_{flt} \quad (9)$$

where,

$$t_q^i = t_q^{i-1} \left(1 + \frac{Pd_0}{v_c(1 - \rho_i)} \right) \quad (10)$$

and Pd_0 is the probability that the delay at stage $i > 0$. The first term of Eq. (9) represents the queuing delay at the subsequent stage while the other terms represent the processing, scheduling, and transmission delay the flit incurs to reach that stage, respectively. It is interesting to note that the values of Pd_0 for the M/M/n system ($\bar{P}d_0$) is only about 6% higher than those of the M/D/n system [15]. Hence,

$$Pd_0 = (1 - 0.6) \bar{P}d_0 \quad (11)$$

where

$$\bar{P}d_0 = \frac{p_d^i v_c}{v_c(1 - \rho_i(1 - p_d^i))} \quad (12)$$

Similar to Eq. 7, the service time seen by the header flit at the source node virtual channels queue, \bar{t}_s , is

$$\bar{t}_s = t_q^{H_{avg}} + \left(\frac{pkt_l}{flt_l} - 1 \right) t_{flt} \quad (13)$$

The rest of the analysis is exactly similar to that of the M/M/n system described in the previous section. We just substitute the service time seen by the source node, predicted by Eq. (13), in Eq. (8). Notice that the main difference between the M/M/n and M/D/n systems is the service time obtained by Eq. (5) and Eq. (9), respectively.

The multi-stage queuing system shown in Figure 4 represents an open queuing model with H_{avg} queues in tandem [16]. As will be explained in Section 4.3, the discrepancy between the packet latency obtained by simulation and any of the above models is large, especially for high loads. This is because the assumption of having Poisson arrivals at each of the subsequent queues in that system is not accurate. Thus, we use a second approach, finite state Markove model, to analyze the virtual channel occupancy probabilities. Our experimental results show that the second approach is more accurate in analyzing the system behavior than the previous one.

4.2 Finite State Markove Model

Our analysis in this section is based on that developed for the k -ary n -fly networks with virtual channels flow control [7]. However, we assume a different interconnection topology and variable packet arrival rates at every stage. Also, our analysis takes into consideration the flit scheduling and processing delays.

As stated in our assumptions, the delay at the destination node is just t_0 ; it is always ready to accept a packet. The delay in the internal nodes is increased because a header flit may be blocked if no virtual channel at a subsequent stage is available. The average waiting time such a flit takes to acquire a virtual channel at stage i , t_a^i , is given by:

$$t_a^i = \frac{t_{vc}^i}{2} p_{v_c}^i \quad (14)$$

It is the product of the *average* service time seen by the header flit at stage i when all virtual channels are occupied (in this model, v_c is the *exact* number of virtual channels per link) and the probability that this condition has occurred. Therefore, the total time a header flit spends waiting to acquire virtual channels between stage i and the destination node is

$$W_i = \sum_{j=0}^i t_a^j \quad (15)$$

The approximate probability of having an individual header flit blocked at stage i , given that there are j occupied virtual channels is

$$W_{i-1}/jt_0 \quad (16)$$

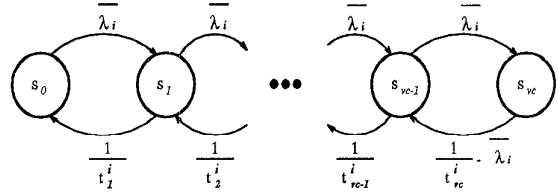


Figure 5: Markove model used to calculate the virtual channels occupancy probabilities

This is the ratio of the time an individual channel is blocked in the subsequent stages to the time required to service that channel with no idling (when there are j occupied virtual channels), jt_0 ³. Under the assumption that the blocking probabilities of the j virtual channels are independent, the probability that all occupied virtual channels are blocked at stage i when there are j of them is

$$b_j^i = \left(\frac{W_{i-1}}{jt_0} \right)^j \quad (17)$$

The next step is to find the effective service time seen by the header flit at stage i when there are j virtual channels occupied, t_j^i . It is the time required to service the packet at stage i without idling plus the delay time it incurs when all occupied virtual channels at that stage are blocked. Hence, t_j^i is given by

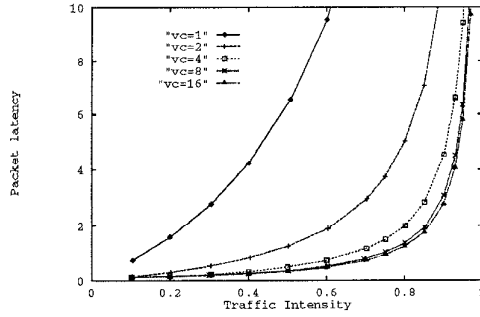
$$t_j^i = t_0 (1 + b_j^i) \quad (18)$$

Now, we write the virtual channel occupancy probabilities at stage i , p_j^i , used in Eq. (14). Figure 5 illustrates the Markov model used to calculate such probabilities. It is a finite state model [17] with $v_c + 1$ states, where s_j corresponds to having j virtual channels being occupied. At any node in the subsequent stages, a transition from state s_j to s_{j+1} takes place with rate $\bar{\lambda}_i$, where i is the stage number. The transition rate to state s_{j-1} is $1/t_j^i$. Solving the state equations of such model [7] gives

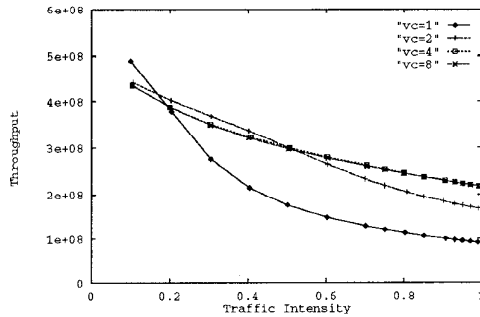
$$q_j^i = \begin{cases} 1 & : j = 0 \\ q_{j-1}^i \bar{\lambda}_i t_j^i & : 0 < j < v_c \\ q_{v_c-1}^i \frac{\bar{\lambda}_i}{\frac{1}{t_{v_c}^i} - \bar{\lambda}_i} & : j = v_c \end{cases} \quad (19)$$

$$p_j^i = \begin{cases} \frac{1}{\sum_{m=0}^{v_c} q_{i,m}^i} & : j = 0 \\ \frac{q_{j-1}^i \bar{\lambda}_i t_j^i}{\sum_{m=0}^{v_c} q_{i,m}^i} & : 0 < j < v_c \\ \frac{q_{v_c-1}^i \bar{\lambda}_i}{\sum_{m=0}^{v_c} q_{i,m}^i} & : j = v_c \end{cases} \quad (20)$$

³Note that this is not an exact estimate since it does not consider the processing, scheduling, and transmission delays incurred at subsequent stages. Also it does not consider the increase in the effective service time due to blockage [7].



(a) $N = 256$ nodes, $C_L = 624$ M bits/s, and $pkt_l = 4Kbytes = 100flt_l$



(b) $N = 32$ nodes, $C_L = 624$ Mbits/s, and $pkt_l = 2Kbytes = 50flt_l$

Figure 6: Predicted Latency (a) and Throughput (b) for different numbers of virtual channels

If there are vc_{avg} virtual channels being occupied at any stage in the network, a flit takes $vc_{avg} \frac{flt_l}{C_L}$ time to traverse one channel. This is due to the delay caused by virtual channels multiplexing. At the source node, vc_{avg} is given by [7]:

$$vc_{avg} = \frac{\sum_{j=1}^{v_c} j^2 p_j^{H_{avg}-1}}{\sum_{j=1}^{v_c} j p_j^{H_{avg}-1}} \quad (21)$$

Finally, the service time seen by the packet at the source node, \bar{t}_s , is calculated by

$$\bar{t}_s = \frac{pkt_l}{C_L} vc_{avg} + \frac{\lambda_{H_{avg}}^-(t_0 + t_a^{H_{avg}-1})^2}{2(1 - \lambda_{H_{avg}}^-(t_0 + x_a^{H_{avg}-1}))} + T_{fld}(H_{avg}-1) \quad (22)$$

As shown in Eq. (22), \bar{t}_s has three components:

- Virtual channels multiplexing delay.
- Waiting delay at the source node virtual channel queue. It is modeled with a M/D/1 queue with $\lambda = \lambda_{H_{avg}}^-$ and $\mu = t_0 + x_{H_{avg}-1}$

- The time a one flit takes traversing the $H_{avg}-1$ hops of the network, where $T_{fld} = (flt_l/C_i) + t_{hpc} + t_{sch}$, when there is no contention.

The last step is to consider the queuing delay in the source node segmentation queue that contains flits sent in the same direction. Similar to our previous analysis, the packet latency, T , can be found using Eq. (8), where \bar{t}_s is found by Eq. (22).

Figure 6 (a) shows the latency predicted by Eq. (8) versus the packet arrival rate for different numbers of virtual channels when \bar{t}_s is computed using Eq. (22). Note that for low number of virtual channels, for instance $v_c = 1$, the packet latency increases more rapidly. As the number of virtual channels, v_c , increases, a lower latency can be obtained. However, the improvement in the packet latency becomes flattened when the number of virtual channels per link becomes larger than 8.

If we define the throughput as the maximum traffic that can be offered by a node in one direction, λ_{max} , then we can write

$$\lambda_{max} = \frac{1}{\bar{t}_s} \quad (23)$$

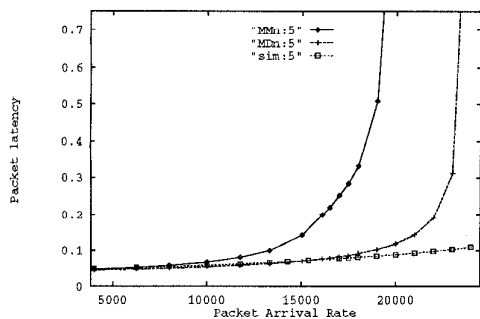
Figure 6 (b) depicts the throughput, predicted by Eq. (23), for different numbers of virtual channels. For very low loads, lesser number of virtual channels gives better throughput. However, as the load increases, the throughput decreases more rapidly for smaller number of virtual channels. Higher number of virtual channels gives better throughput for higher loads. It is clear from Figures 6 (a,b) that having four to eight virtual channels per physical link is adequate.

4.3 Experimental Results

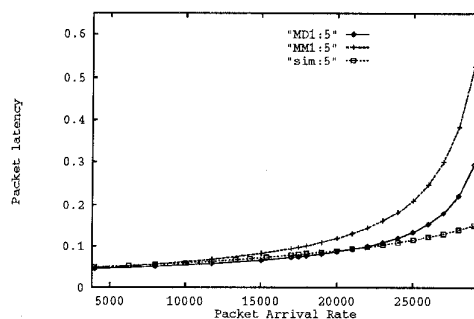
To validate our analysis and study the effect of the parameters used in that analysis on the performance, we have simulated the multi-link ring network using the OPNET⁴ simulation tool [18]. We have used the OPNET tool because of its support to hierarchical structures. With such property, many of the events that take place in reality can be simulated more accurately.

We have simulated a number of multi-link ring networks and different routing strategies. Each simulation was run until saturation in the average delay occurred. The network model consists of N communication nodes connected by multiple unidirectional links (in both directions) of 624 Mbits/s each. The link capacity is divided among the busy virtual channels sharing that link. Each node has a packet generator (P-G), a segmentation queue (S-Q), $2l$ transmitters (T_i s, $0 \leq i \leq 2l-1$), $2l$ receivers (R_i s), and $2l$ routers which are also queue modules ($R-P_i$ s).

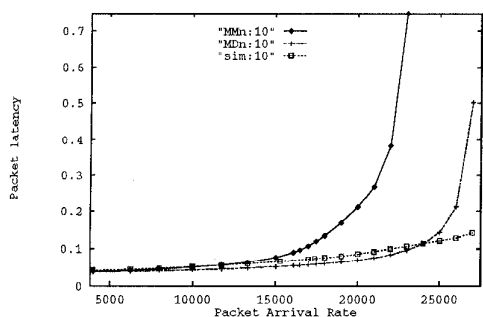
⁴OPNET is a trademark of MIL 3, Inc.



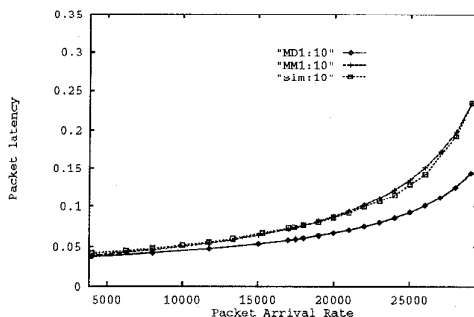
(a) $pkt_l = 5flt_l$



(a) $pkt_l = 5flt_l$



(b) $pkt_l = 10flt_l$



(b) $pkt_l = 10flt_l$

Figure 7: Comparison of predicted (using M/D/n & M/M/n) and measured latency (in ms), T , when $N = 16$ nodes.

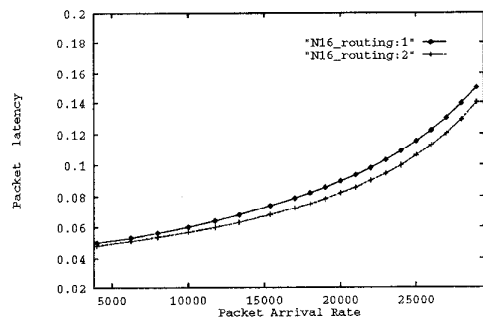
Figure 8: Comparison of predicted and measured latency, T , when the source segmentation queue is an M/D/1 or M/M/1, $N = 16$ nodes.

Figure 7 shows the expected packet latency, predicted by Eq. (8), versus the packet arrival rate. The figure depicts the analytical results predicted by the M/M/n and M/D/n queuing models discussed above, and simulation. Note that the virtual channel occupancy probabilities of these queuing models were found based on an infinite state Markov model. In Figure 7 (a), we assumed $N = 16$, $pkt_l = 2 \text{ kbytes} = 5flt_l$ (the packet is divided to 5 flits), and $v_c = 4$. Figure 7 (b) shows the packet latency when $pkt_l = 10flt_l$. As depicted in the figures, dividing the packet to smaller flits produces results closer to those measured by simulation. Also, the results predicted using the M/D/n queue model are closer to simulation results than those of the M/M/n queues. Both figures show that for very low loads, the results predicted by the analysis and simulation are almost identical. However, as the load increases the discrepancy between them also increases. This is due to the assumption that packets follow a Poisson arrival mechanism at each of the successive queues which is not accurate for high loads.

For a system with multiple M/D/n or M/M/n queues in tandem and a fixed packet length, such assumption

is not quite valid. This is because the service time at the subsequent queues of such system are not independent. Moreover, it has been shown by simulation that when packet lengths and inter arrival times are correlated and under heavy traffic conditions, the average delay per packet is smaller than it is in the ideal situation when there is no such correlation [19]. This argument explains the difference between simulation and the analytical model that uses the above queuing systems at high loads.

Figure 8 shows the expected packet latency, predicted by Eq. (8), versus packet arrival rates when \bar{t}_s is computed using the finite state Markov model. The figure depicts the analytical results predicted by assuming M/M/1 or M/D/1 queuing model for the segmentation queues at the source node, and simulation. In Figure 8 (a), we assumed $N = 16$, $pkt_l = 2 \text{ kbytes} = 5flt_l$, and $v_c = 4$. For low loads, all three results are almost identical. However, as the load increases, the difference between the M/M/1 and M/D/1 gets larger. As depicted in the figure, the measured results obtained by simulation are closer to the M/M/1 model when the load is low. As the load increases, the simulation results get



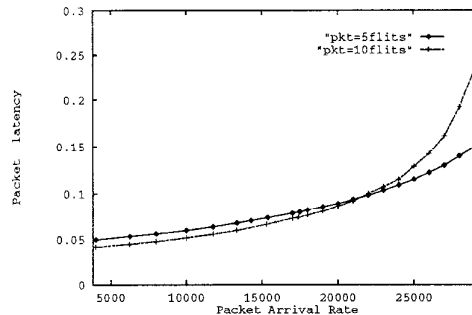
$N = 16$ and $pkt_l = 5fl_t$

Figure 9: Comparison between two measured routing strategies

closer to those of the M/D/1 model. For high loads, simulation results are smaller than both models and the discrepancy gets larger as the load increases. This is due to using Eq. (16) in finding the probability of having an individual header flit blocked at stage i . In that equation, we assumed that the total time required to service j occupied virtual channels at stage i with no idling is jt_0 . This slightly underestimates the virtual channel delay (since it overestimates the blocking probability) because the processing delay, scheduling delay, transmission delay, and the increase in the effective service time due to blockage are not considered.

Figure 8 (b) shows the packet latency with the same parameters assumed in Figure 8 (a), except that $pkt_l = 10fl_t$. It is clear that reducing the flit size (increasing the number of flits per packet) overcomes the problem of overestimating the blocking probability occurred in Figure 8 (a). For low loads, the M/D/1 model results also agree with that of the M/M/1 and simulation. As the load increases, the discrepancy between them gets larger.

Figure 9 shows the impact of the routing scheme on the packet latency. We have simulated two routing strategies for a multi-link ring with $N = 16$ and $pkt_l = 2Kbytes = 5fl_t$. In the first strategy, packets that are originated from a node that belongs to building block i (four-node ring i) and sent to a destination node in ring j use ring j only. In the second strategy, packets with source nodes belong to ring i , and destination nodes belong to ring j use ring i only. Note that there are two directions that packets can use (right or left), depending on the distance between the source and the destination nodes. This figure shows that the second strategy is better because the contention on each physical link is less. Only packets originated from nodes that belong to the same building block can share the same link. Consequently, a more channel service rate can be



(b) $N = 16$, $v_c = 4$, and $pkt_l = 2Kbytes$

Figure 10: Measured packet latency for different flit sizes.

achieved. In the first strategy, more packets can share the same link. Since we assume that there is only one segmentation queue to store all flits sent in the same direction, more flits will be waiting for service in that queue and more delay is incurred.

Figure 10 shows the measured packet latency for different flit sizes ($N = 16$, $v_c = 4$, and $pkt_l = 2Kbytes$). For low loads, a smaller flit size produces lesser latency. As the load increases, a bigger flit size gives lesser latency. Higher load means more flits sharing the same channel. Smaller flit size means that flits are more spread between source and destination nodes. Therefore, more delay is incurred. A similar argument can be applied for blocked header flits; blocked flits will block lesser number of virtual channels when the number of flits per packet is less.

5 Conclusion:

In this paper, we have presented a new design for a dynamically reconfigurable multi-link ring network to meet communication requirements of high-performance parallel and distributed systems. Our approach to design the switch is aimed at achieving simplicity and parallel processing of messages.

We have discussed two approaches to analyze the system performance. In the first approach, we have developed an analytical model using the M/M/n and M/D/n queuing models. While in the second approach, we have introduced an analytical model based on a finite state Markov model developed for networks with virtual channel flow control. Simulation, using the OPNET tool, has been implemented to check the analytical results. The simulation results indicate that the second approach is more accurate in analyzing the behavior of the packet transfer time in the proposed network. This is because the assumption that packets follow a Poisson arrival mechanism at each of the M/D/n or M/M/n

queues in sequence in the first approach is unrealistic, especially at high loads.

The analysis also indicates that the use of wormhole routing and virtual channel flow control is a significant factor in increasing the network performance. For low traffic intensity, a small number of virtual channels is adequate. However, for high traffic intensity, a larger number (up to 8) of virtual channels is more appropriate. With the reconfigurability property the proposed design can support the best choice for the number of virtual channels, and therefore a better performance can be achieved. In addition to the implementation of the multi-link ring network, we are currently studying implementing other topologies, such as Mesh and 2-d Torus.

References

- [1] D. Smitley, F. Hady, and D. Burns. "Hnet: A High-performance network evaluation testbed". *Proceedings of the 1992 International Conference on Parallel Processing*, I:276-279, 1992.
- [2] O. Landsverk, J. Griepsl, J. Mathisen, J. Solheim, and L. Utne. "RENNs- a Reconfigurable computer system for simulating artificial neural network algorithms". *Proceedings of the ISMM International Conference on Parallel and Distributed Computing and Systems*, pages 251-256, Oct 1992.
- [3] K. Aly and P. Dowd. "Parallel computer reconfigurability through optical interconnects". *Proceedings of the 1992 International Conference on Parallel Processing*, I:105-108, 1992.
- [4] R. Johnson and J. R. Goodman. "Synthesizing general topologies from rings". *Proceedings of the 1992 International Conference on Parallel Processing*, I:86-95, 1992.
- [5] L. M. Ni and P. K. McKinley. "A Survey of wormhole routing techniques in direct networks". *IEEE Computer*, pages 62-76, Feb 1993.
- [6] Reuven Cohen. "Session swapping: A new approach for optimal bandwidth sharing of ring circuit switched channel". *IEEE/ACM Transactions on Networking*, 2(3):263-268, June 1994.
- [7] W. J. Dally. "Virtual-channel flow control". *IEEE Trans. Parallel and distributed Systems*, 3(2):194-205, Mar 1992.
- [8] Xiaodong Zhang. "System effects of interprocessor communication latency in multicomputers". *IEEE Micro*, April 1991.
- [9] Terence D. Todd. "The Token grid network". *IEEE/ACM Transactions on Networking*, 2(3):279-287, June 1994.
- [10] I. Cidon, I. Gopal, and A. Segall. "Connection establishment in high-speed networks". *IEEE/ACM Transactions on Networking*, 1(4):469-481, Aug. 1993.
- [11] Saad Alkasabi and Salim Hariri. "A Dynamically reconfigurable switch for high-speed networks". In *Proceedings of the International Phoenix Conf. on Computers and Communications*, pages 508-515, March 1995.
- [12] Inc. Xilinx. *The Programmable Logic Data Book*. Xilinx, Inc., 2100 Logic Drive, San Jose, California 95124, 1994.
- [13] D. H. Linder and C. Harden. "An Adaptive and fault tolerant wormhole routing strategy for k-ary n-cubes". *IEEE Transactions on computers*, 40(1):2-12, Jan 1991.
- [14] Mischa Schwartz. *Telecommunication Networks Protocols, Modeling and Analysis*. Addison-Wesley Publishing Company, Inc., 1988.
- [15] Roshan L. Sharma. *Network Topology Optimization*. Van Nostrand Reinhold, 115 Fifth Ave., New York, Ny, 10003, 1990.
- [16] H. G. Perros and T. Altioik. *Queuing Networks with Blocking*. Elsevier Science Publishing Company, Inc., New York, 1989.
- [17] Kishor S. Trivedi. *Probability & Statistics with Reliability, Queuing, and Computer Science Applications*. Prentic Hall Inc., Englewood Cliffs, New Jersey, 1982.
- [18] K. Marshall, R. Garb, S. Baraniuk, A. Cohen, and S. Oppenheimer. *OPNET: Tool Operations, User Interface, and Simulation Kernel manuals*. MIL 3, Inc., 3400 International Drive NW, Washington, DC 20008, 1989-1993.
- [19] D. Bertsekas and R. Gallager. *Data Networks*. Prentic Hall, Englewood Cliffs, New Jersey, 1992.