Reliability Aware NoC Router Architecture Using Input Channel Buffer Sharing

M. H. Neishaburi
McGill University
Montreal, QC, Canada
mh.neishabouri@mail.mcgill.ca

Zeljko Zilic
McGill University
Montreal, QC, Canada
zeljko.zilic@mcgill.ca

ABSTRACT

To address the increasing demand for reliability in on-chip networks, we proposed a novel *Reliability Aware Virtual channel* (RAVC) NoC router micro-architecture that enables both dynamic virtual channel allocations and the rational sharing among the buffers of different input channels. In particular, in the case of failure in routers, the virtual channels of routers surrounding the faulty routers can be totally recaptured and reassigned to other input ports. Moreover, our proposed RAVC router isolates the faulty router from occupying network bandwidth. Experimental result shows that proposed micro-architecture provides 7.1% and 3.1 % average latency decrease under uniform and transpose traffic pattern. Considering the existence of failures in routers of on-chip network, RAVC provides 28% and 16% decrease in the average packet latency under the uniform and transpose traffic pattern respectively.

Categories and Subject Descriptors

B.8.1 [**Performance and Reliability**]: Reliability, Testing, and Fault-Tolerance

General Terms

Reliability

Keywords

Network on Chip, System on Chip, Virtual Channel.

1. INTRODUCTION

Driven by unquenchable demand for having high bandwidth, throughput and in particular scalable platforms, System-on-Chip (SoC) designers find on-chip interconnections to be a limiting factor in terms of the performance and energy consumption[1]. Increases in the resistance of wires due to the decreases in wire cross-section cause to increase in the latency, stressing out the problem of traditional interconnect. Expectedly, a signal would require multiple clock cycles to traverse the length of a large wire in SoCs. The network-on-chip (NoC) can be used to combat the delay emanating from slow global wiring, providing scalable, low power and high-bandwidth communication infrastructure to the SoCs design. NoCs were outlined as an advance for SoCs. Since all links in the NoC can operate simultaneously on different data packets, a high level of parallelism is making it attractive for replacing previous communication architectures like dedicated point-to-point signal wires, shared buses, or segmented buses with bridges.

Since then, it was widely established that NoCs can provide enhanced throughput and scalability.

As the transistor feature size shrinks, so does the voltage, leaving less noise margin. Hence, the ability of the network to function in the presence of component failures, reliability and fault-tolerance issues become important.

The NoC topologies will need to be considered from this angle. Network topologies are the configurations of routers and the Network Interfaces (NIs) to connect to a router. The fundamental duty of NIs is the interfacing between processing elements and network infrastructure. In the domain of router architectures, the comprehensive research has been carried out to enhance the performance, power, and fault-tolerant mechanisms.

Buffers are the instrumental elements in router input and output channels. They were shown to consume about 64% of the total router leakage power [3] making them largest leakage energy consumers in NoCs. Moreover, in terms of dynamic energy consumption, buffers are dominant [3]. It was shown in [5] that far more energy consumption is expected in storing packets in buffers than transmitting them. On balance, the effective and resourceful management of buffers and hence input and output channels in NoC routers has a crucial effect in performance and efficiency of interconnection networks. In this work, we propose new NoC router architecture that enables dynamic reconfiguration of input channels in NoC routers. Our proposed architecture tries to alleviate the effect of faulty routers and congestion in an NoC. Intuitively, the crux of our design is in the effective usage of available buffers, particularly when switches fail. To the best of our knowledge, this work is the first attempt at sharing among the buffers of different input channels.

We propose a new micro-architecture for NoC routers that enables sharing among virtual channels (VCs) of input ports. Addressing the fault tolerance requirement of our architecture, our suggested switch can dynamically change the pre-assigned number of VCs to an input port. In the case of switch failure, the VCs of routers surrounding the faulty routers can be totally recaptured and reassigned to the other input ports. Isolating the faulty switch from receiving incoming packets and the eliminating related power dissipation is another advantage of our proposed reliability aware architecture.

2. RELATED WORK

The relationship between network performance and buffer resources has to be taken into account when trying to reduce buffer sizes to minimize energy consumption as well as the silicon area. In fact, flow control policies play a decisive role in the

GLSVLSI'09, May 10–12, 2009, Boston, Massachusetts, USA.
Copyright 2009 ACM 978-1-60558-522-2/09/05...$5.00

management and sizing of buffers[4]. The wormhole flow control relaxes the constraints on buffer size at each router by controlling at the granularity of flit, instead of packet. In particular, this flow control enables more efficient use of buffer space than store-and-forward and Virtual Cut-Through flow control [9].

Although in wormhole flow control buffers are allocated at the flit level, a blocked packet can impede the progress of other packets because physical paths still are allocated at the packet level. Due to distribution of single packets across the several ports, such a blocking causes substantial decreases in the performance of NoC. The application of VC flow control is instrumental to alleviating the problem of blocking in wormhole flow control.

Assigning multiple virtual paths to the same physical channel is the essence of VC flow control. Generally, each virtual path has its own associated buffer queue [10] - not only that VC routers can increase throughput by up to 40% over wormhole routers without VCs, but virtual channels also help with deadlock avoidance. However, the performance of VC flow control worsens relative to the fixed VC structures. In other words, supposing the statically-allocated VCs implementation lowers buffer utilization due to lack of flexibility in buffer size. Practically speaking, low throughput is expected at high data rates due to lack of VCs, assuming routers are configured with few deep VCs. For low data rates, on the other hand, if many shallow VCs are arranged, the packets are distributed over a large number of routers. Therefore, contentions and the increase in the latency arise as a consequence of extra interrupts in continual packet transfers.

In [4] the authors explain the aforementioned facts and show that the optimal number of VCs depend on the traffic pattern. At low data rate, increasing VC depths results in better performance. For high rates, the optimal structure depends on the distributing patterns. It is advisable to increase VCs under uniform data patterns, but to decrease VC depth under hotspot patterns, e.g., in matrix transpose. Hereafter, the dynamic buffer sizing becomes an instrumental in NoC routers. In [6] an analytical approach for assigning buffer sizes at design time have been investigated. However, their proposed technique revolves around assignment of the size and the number of VCs at the design time that is static. In fact, supposing a particular application and specific hardware mapping, they apply their method to find the optimal buffer sizes applicable to that particular application. On the other hand, within the realm of NoCs dealing with different workloads and spontaneous traffic, the runtime management and reconfiguration of buffer organization are more interesting. In fact, regardless of the traffic type in the NoC, dynamic scheme can be exploited to maximize utilization. In [7], a unified and dynamically allocated buffer structure was presented as Dynamically Allocated Multi-Queue (DAMQ) buffer; however, utilizing a fixed number of queues and hence VCs per input port is one limitation of this architecture. Another disadvantage of their approach specifically in domain of NoC is the complex control logic of the DAMQ buffer. Particularly, every read and write operation needs there cycles to complete, which might be excessive in an on-chip router. To improve drawbacks of DAMQ in terms of hardware overhead and overall complexity, DAMQ with self-compacting buffer was introduced in [7]; using registers which selectively shift some flits inside the buffer to enable all flits of one VC to occupy a continuous buffer space was the crux of this approach. The VC Regulator (ViChaR) which dynamically allocates VCs and buffer slots in real-time basically has been presented in [12]. Dynamic Allocations of VCs in this scheme is based on the traffic condition of the interconnection network. However, this dynamic VCs allocation scheme lacked efficient structure with little hardware overhead to support various packet sizes or traffic patterns. Additionally, the idea of enabling sharing among the buffers of different input channels which is our innovation in this work has not been addressed in their proposed scheme. In [11] a novel dynamic VC architecture to escape the HOL blockings is introduced. In their scheme, the low overhead link list structure is used to manage arriving and departing flits. In general, utilizing the link status and switch arbitration results, their proposed structure creates variable number of VC at the run-time to maximize the throughput. However, their proposed architecture could not perfectly utilize the unused buffers of their neighboring input channels. To the best of our knowledge, existing dynamic virtual-channel allocation schemes in the realm of NoC never address the issue of perfectly utilizing and sharing available buffers in the input channels of router.

3. PRELIMINARIES

Micro-architecture of the conventional VC NoC router is shown in Fig .1. The Routing Unit, VC allocator, Switch allocator, input channels and a Crossbar constitute basics elements of a Virtual Channel NoC router[14]. In general, NoC routers may have any number of input and output ports; however, network topology ultimately determines the number of input and output ports of each router. In most implementations, the numbers of input and output ports are five; four inputs from the four cardinal directions (North, East, South and West) and one from the Network Interface (NI) which is in charge of restructuring Local Processing Element (PE) packets to the acceptable format of NoC routers. Using the destination information in header flit, Routing Unit leads the header-flit of incoming packet to the appropriate output port. A unit of information that can be transferred across a physical channel in a single step or cycle is called flit. Basically, a NI forms the flit-sized messages before injecting them to the network. To easily distinguish the type of information carried out by flits, they are categorized into Header-Flit (H-F) that keeps the information of the source and the destination, Data-Flits (D-F) that packs the body of a message and Tail-Flit (T-F), which is the representative of end of a packet. On the other hand, a routing algorithm determines the path along which a packet is delivered to destination node. The routing algorithm can be either deterministic or adaptive. Ideally, deterministic routing algorithms always supply the same path between a given source/destination pair. On the other hand, by utilizing information about network traffic, adaptive routing algorithms try to avoid congested or faulty regions of the network[13]. In VC NoC routers, each physical channel involves the finite number of VCs. Considering North input channel of Fig. 1, we can define the basic behavior of VC NoC router. While a header-flit is arriving from the upstream, its VC identifier (VC_ID) defined with the previous router is decoded. The header-flit will be buffered in the appropriate VC according to its decoded VC-ID. Meanwhile, the state of aforementioned VC will be changed to the routing state. There are four specific sets of states corresponding to each VC, as shown in Figure 1. Every header-flit activates the specific request lines of the Global VC allocator that is directly matched to the result of the routing unit. Inspecting both the priority of input VC and the state of requested output VC global-VC-allocator gives output VC to one of the input channels. Switching step is the next state of an input VC. During this state, each input VC provisioned by its respective grant from VC-Allocator enters its request to access the output port through the cross bar.

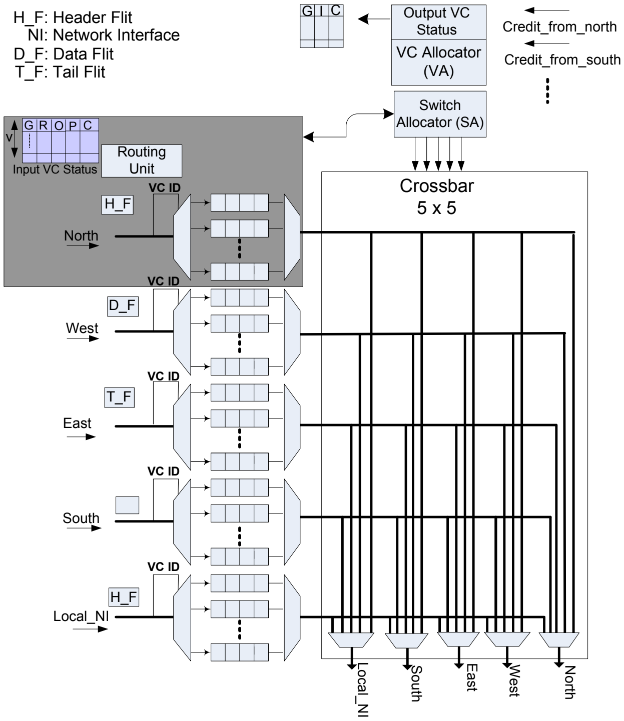


Fig. 1 Conventional VC Router Architecture

4. PROPOSED ROUTER ARCHITECTURE

The *hospitality* of an input channel is defined as the tendency of an input channel to host the arriving flit of other channels.

$$\text{Hospitality[dir]} = 1 - \left(\frac{\sum_{i=1}^{i=curVC[dir]} VC[i][dir].Capcity}{InputChannel[dir].Capacity}\right) \quad dir \in \{N, E, W, S, L\}$$

Equation 1

In Equation1, the *VC[i]][dir].Capacity* represents the number of flits in the ith VC at the specific *direction*. Hence, for each physical input channel there exists hospitality metric. Because the number of flits stored in each VC is reserved in the input VC status, hospitality of each input port can be determined using already available data in an NoC router; moreover since the input channel capacity usually is power of two, in RAVC NOC router simple shifters and adder were used for the calculation of hospitality.

The probability of VC expansion to accommodate an incoming header-flit which is supposed to depart in specific direction can be determined using the Equation 2. In general, when the value of credit in specific output VC becomes zero concerning VCs at the destination routers either no longer can accept the new flits due to congestion or have a small amount of buffers.

$$\text{if(VC[i][dir].credit} == 0) \Rightarrow \text{condition[i]} \leftarrow 1; \text{Otherwise(condition[i]} \leftarrow 0)$$

$$\text{ExpNewVC[dir]} = \frac{\sum_{i=1}^{i=curVC[dir]} Condition[i]}{curVC_{num}[dir]}$$

Equation 2

As shown in Fig. 2, when R[2,1] is going to send a packet to the R[4,4] in an ordinary and non-faulty state the flits of this packet will pass through R[2,2], R[2,3], R[2,4], R[3,4] and finally R[4,4] based on the XY routing algorithm. If R[2,3] becomes faulty, by resorting to a fault tolerant routing algorithm like XY-YX [13], R[2,2] actually will forward all the incoming flits from R[2,1] to R[3,2] or R[1,2]. As shown in Figure 2, having extra buffers in neighboring switches of R[2,3] would be instrumental in following up the extra traffic. This extra traffic is predictable as a consequence of having faulty routers in our interconnection network. In fact, R[3,3] and R[1,3] particularly in the depicted scenario due to failure in R[2,3] are supposed to received more flits compared to the non-faulty condition.

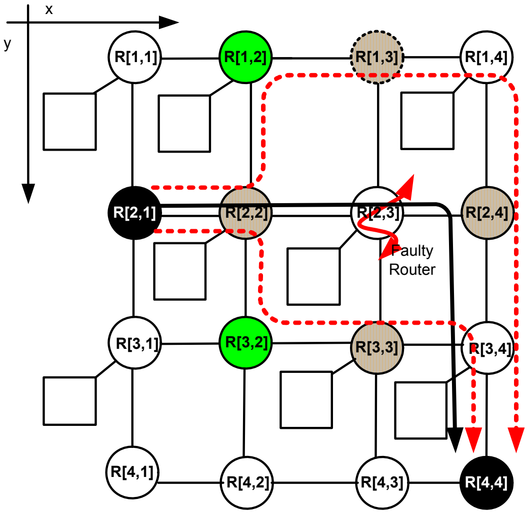
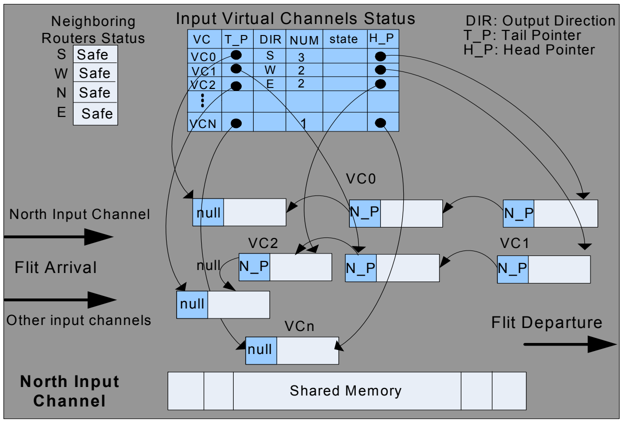


Fig. 2 Dynamic Input Ports Buffer Sharing

4.1 Proposed RAVC Router Architecture

To enable dynamic allocation of storage between different VC, link listed data structures are used to represent VCs in our RAVC router. Using the Linked list more efficient use of memory is expected. In fact, many previous studies [11][12] have shown that static VC allocation when there are large numbers of VCs leads to unbalanced loading across VCs. Thus, it is advantageous to allocate more memory to the busy channels and less to the idle channels. Fig. 3 shows our proposed input channel architecture. Moreover, we used four global registers that indicate the condition of four neighbor routers. Referring to these status registers, the input channel will be notified about the status of neighbor routers in particular whether they are safe or faulty. The arriving header-flit after passing modified RAVC router stage shown in Fig. 3 is stored into free memory location; consequently, *Dynamic VCs Allocator* will specify the particular VC identifier; hereafter, the list labeled with this identifier is nesting the arriving header-flit. On the basis of extracted VC identifier, a header-flit will be either stored into a new list or at the end of an already existed list. However, the data-flits and tail- flit actually inherits the VC-Identifier of their header-flits; in fact, they do not require to the

VC-Allocator unit to dispense a VC identifier. As shown in Fig. 3, Head-Pointer (H-P) of each VC points to the location of this flit at the shared memory of input channel. The switch allocation arbitrates among all the VCs to find the winner VC identifier. After departure of a flit pointed with the head-pointer of winner-VC, the header-pointer of that VC will be updated by the next-pointer of departed flit. In our structure, the VCs may be dynamically regulated according to the traffic conditions.



VC	T_P	DIR	NUM	state	H_P
VC0	●	S	3		●
VC1	●	W	2		●
VC2	●	E	2		●
⋮					
VCN	●		1		●

Fig. 3 Input Channel Structure

When a new flit arrives, it is inspected whether the new flit comes from the safe or faulty router; thereby, the new flit is discarded if it is emanated from the faulty router. Fig. 4 illustrates the data-path of RAVC router. Using Tail-Pointer, Head-Pointer and Next-Pointer register file, RAVC router makes one cycle read and write operation possible. In principal, group of operations shown in Fig. 5 dependent upon operation phase of RAVC router are executed on this input VC data-path. Executing these operations, winner-incoming flits will be stored in the available slot indicated by the Free-Buffer-Tracker Fig. 5 (A1), tail pointer of concerning VC is updated to point to the address of incoming flit in the shared-buffer and finally to keep chain of flits in the VC previous Tail_Pointer of this VC must be stored as a Next_Poniter of new incoming flit Fig. 5 (B1)(B2)(B3). Afterward, for the departure of each flit, there exist other three operations shown in Fig. 5 (C1)(C2) Where, eqn. (1) reads a flit from the winner VC, eqn. (2) revises the head pointer of its VC to point to the next flit, and eqn. (3) updates the slot map. Our proposed RAVC router has changed the routing stage of conventional VC router due to the effective and reliable handling of available buffers. As Fig. 5 depicts, keeping the status of neighboring routers as well as the hospitality measure of other input channel, incoming flit may be either moved to other input channels or handled in the current input channel. If all the buffers of current input channel are occupied; in other words, there is not any available input buffer for hosting a new incoming flit, this input channel instead of discarding this new incoming flit transmit it to other input channel.

As Fig. 5 shows, to avoid the increase in energy consumption, RAVC router has adopted a tri-state buffer to switch between the incoming flits of a local port or other input ports. The winner-incoming flit follows the bandwidth allocation and particularly routing in case of being header-flit. During the bandwidth allocation and routing phase respectively, *free-buffer-tracker* specifies the location of the winner-incoming flits and *routing-unit* indicates the appropriate output port of this flit. Routing decision and hospitality measurement and concerning incoming flits move carried out in parallel in that there is not any operational dependency between them.

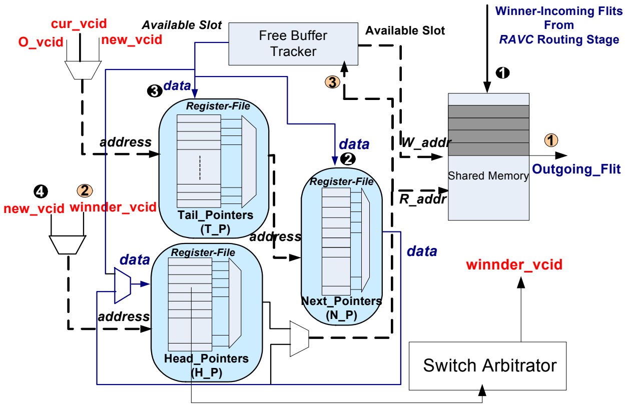


Fig. 4 RAVC Input Channel Data-Path

RAVC router supports dynamic VC allocation. During the VC allocation stage, the *VC-Availability-Tracker* and the *VC-Dispenser* are in charge of the VC determination to new header-flits. Since VC-identifier of data-flits and the tail flit has been designated previously by transmission of their header-flit also this designated VC-identifier has been stamped in these flits by their previous router, next and tail-pointer register file of regarding VC-identifier as shown in Fig. 5(B1) will be updated to the location of these flits in the shared buffer which is previously specified by the *free-buffer-tracker* (). However, *VC-Availability-Tracker* and *VC-Dispenser* specify the VC-Identifier of a header-flit in this stage. In principal, this header flit first requests access to the particular output port through our modified VC arbitrator; accruing the required grant from VC arbitrator, the header-flit sends its request to the *VC-Availability-Tracker* and the *VC-Dispenser* units. Consequently, taking into account number of current available VC as well as HOL (Head of Line blocking) condition, these units will decide whether to dispense a new VC or determine one of the available VC identifier. In case of new VC distribution, Next, Head and Tail pointer register file of newly distributed VC are updated Fig. 5(B2); however, as illustrated in Fig. 5(B3) if one of the current available VC identifiers specified to host this header-flit, the tail-pointer register file and next-pointer register file of this VC will be updated accordingly. In both cases, this router has to inform the previous one about the adopted VC-ID; thereby, previous router updates its output VC status. Eventually, during the switching stage the output port of crossbars will be specified among the input ports. Our modified *Switch Allocation (SA)* unit similar to the conventional SA unit will carry out its operation in two stages; firstly, during the first stage of SA the winner request among variable number of VCs in an input channel will be chosen; secondly, the second stage arbitrates among the winner requests from each input port for output ports. The SA unit of conventional router has been modified to be consistent by our dynamic VC planning approach. Therefore, adopting the worst-case scenario when an input channel dispensed all the allowable VCs, which is indeed VC_{max}, RAVC router apply v_{max}:1 arbiters for each input channel at the first stage. However, the second switch arbitration in RAVC router is like the conventional router. Moreover, using the output VC status and resorting to the Credit-Based flow control scheme, switching unit of RAVC router handle inter router communication. When a new header-flit arrives into the specific input channel, VC dispenser specifies whether it needs to make a new virtual channel for this flit or this flit should be placed at the tail of already existing virtual channel.

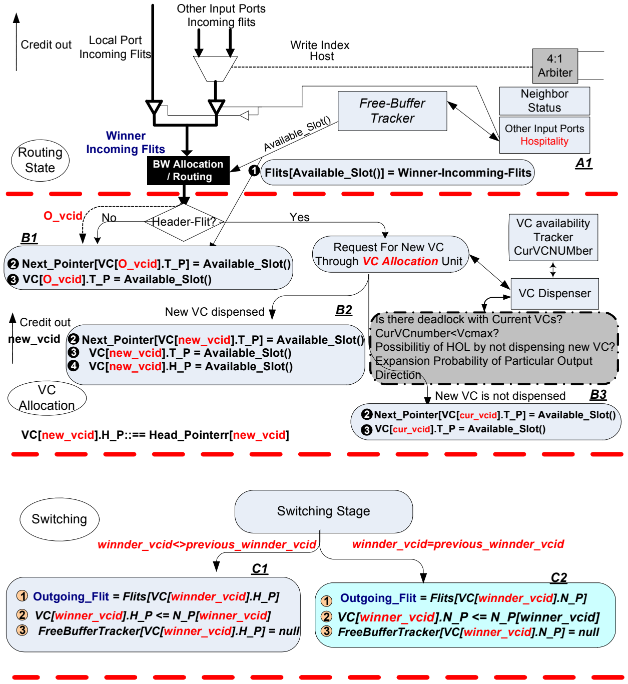

Credit out
Local Port Incoming Flits
Other Input Ports Incoming flits
Write Index Host
4:1 Arbiter
Free-Buffer Tracker
Neighbor Status
Other Input Ports Hospitality
Available_Slot()
Routing State
Winner Incoming Flits
BW Allocation / Routing
❶ Flits[Available_Slot()] = Winner-Incomming-Flits
A1
O_vcid
No
Header-Flit?
Yes
B1
❷ Next_Pointer[VC[O_vcid].T_P] = Available_Slot()
❸ VC[O_vcid].T_P = Available_Slot()
Request For New VC Through VC Allocation Unit
VC availability Tracker CurVCNUMber
VC Dispenser
New VC dispensed
B2
Credit out
new_vcid
❷ Next_Pointer[VC[new_vcid].T_P] = Available_Slot()
❸ VC[new_vcid].T_P = Available_Slot()
❹ VC[new_vcid].H_P = Available_Slot()
Is there deadlock with Current VCs?
CurVCnumber<Vcmax?
Possibilitiy of HOL by not dispensing new VC?
Expansion Probability of Particular Output Direction
New VC is not dispensed
B3
❷ Next_Pointer[VC[cur_vcid].T_P] = Available_Slot()
❸ VC[cur_vcid].T_P = Available_Slot()
VC Allocation
VC[new_vcid].H_P::== Head_Pointerr[new_vcid]
Switching
Switching Stage
winnder_vcid<>previous_winnder_vcid
winnder_vcid=previous_winnder_vcid
C1
① Outgoing_Flit = Flits[VC[winnder_vcid].H_P]
② VC[winner_vcid].H_P <= N_P[winner_vcid]
③ FreeBufferTracker[VC[winner_vcid].H_P] = null
C2
① Outgoing_Flit = Flits[VC[winnder_vcid].N_P]
② VC[winner_vcid].N_P <= N_P[winner_vcid]
③ FreeBufferTracker[VC[winner_vcid].N_P] = null

Fig. 5 Our Modified RAVC Router Stages

VC manager should consider three conditions; first, if there is a deadlock in the current network this header flit should be placed in the escape virtual channel; secondly, if placing the new header at the end of already existing virtual channels, lead to HOL blocking conditions the new virtual channel is expected to be dispensed by VC manager to accommodate the new incoming flit. Eventually, resorting to the expansion probability of particular direction, *VC dispenser* considering the destination direction of this new incoming header-flit performs its decision.

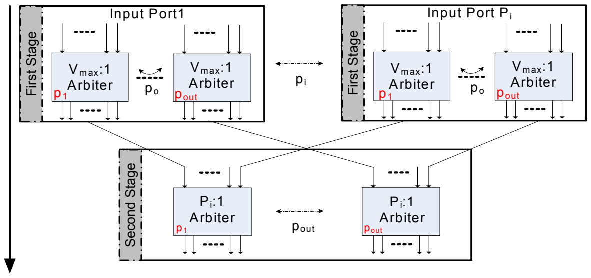


Fig. 6 Proposed Virtual Channel Allocator

The task of particular output channel allocation to header-flit of an incoming packet is given to the VC dispenser. Data-flits of a packet follow the allocated output channel of the header-flit. As Fig. 6 illustrates, supporting the dynamic VC allocation our modified VA reduces the number of virtual channel requests for a particular output port to one request in first arbitration stage. As a consequence of having anywhere between v_{min} and v_{max} VCs per input port at any given time, our proposed scheme needs larger v_{max}:1 arbiters in comparison to v:1 generic arbiter of convention arbiter in the first stage of the allocation. The proposed virtual channel uses smaller Stage 2 arbiters. As Fig. 6 illustrates, on the contrary to the conventional router which accept request for each output channel, the second arbitration stage in RAVC is account for choosing a winner for each output port among all the competing input ports. In fact, instead of having (P_{out}×V) arbiters each of which has to accept P×V request, VC allocator at the second stage in RAVC router has p_{out} arbiters.

5. EXPERIMENTAL RESULTS

To evaluate the effectiveness of our approach, we created generic and customizable RTL level description of our Reliability Aware Virtual Channel (RAVC) NoC router and conventional VC router using VHDL language. First, we find Average Latency of packets transferring under different injection rates. In the second phase of the performance evaluation, we adopted XY-YX fault tolerant routing algorithm as an alternative routing decision[13]. Moreover, we assume uniform and transpose traffic patterns. All simulations were performed in 36 nodes (6× 6)6) MESH network. In our experiments, conventional VC routers have 4 VCs each of which has 16 flits capacity. However, our proposed RAVC routers have 64 flits buffer with the default of 4 minimum VCs yet extendable to 16. In our simulation, we apply two different size packets: (1) the 8 flits, and (2)16 flits packets. To find the Average Latency under different traffic pattern, test-benches connected to each router through local injection channels as a local Processing Element (PE) generates the number of packets somewhere between 10,000 and 100,000 and fills the destination of each packet according to the traffic pattern.

Under uniform traffic pattern, average packet latency of conventional VC router and RAVC router in different packet injection rate are illustrated in Fig. 7 (A)(B) considering the packet size of 8, 16 flits respectively. Averagely, under uniform traffic pattern, our proposed RAVC supplies 7.1% improvement in Average Packet Latency compared to the conventional VC router with the same amount of input buffers. Fig. 7 (C)(D) Illustrates the average packet latency of RAVC versus conventional router under transpose traffic pattern with respect to the packet size of 8, 16 flits; under transpose traffic pattern, such a improvement in average packet latency becomes 3.5%. Simply stated, resorting to the facts that RAVC router supports dynamic VC assignment as well as input buffer sharing; additionally, RAVC dispenses more VCs under the high packet injection rate leading to the decline in the probability of HOL occurrence compare to static VC scheme of traditional router, we can construe such a decrease in the average latency and higher saturated throughput. In the second round of our simulation, we analyzed our MESH network performance in case of failures in routers. We assumed that some routers failed during the operation. As consequence of instinctive reliability awareness of our RAVC router, faulty routers cannot propagate any flits into the network and cooperate in saturation of network since input buffers of their surrounding routers are not still attainable. In our experiments, we assume that state of faulty routers becomes unsafe in their surrounding routers. Fig. 7 (E)(F) illustrate the average packet latency of RAVC router versus average packet latency of conventional router assuming specific number of router failures under uniform traffic pattern. On the basis of extracted values from our VHDL based simulation environment, in a fault prone environment, RAVC leads to 28% and 16% decrease in the average packet latency under the uniform and transpose traffic pattern, respectively.

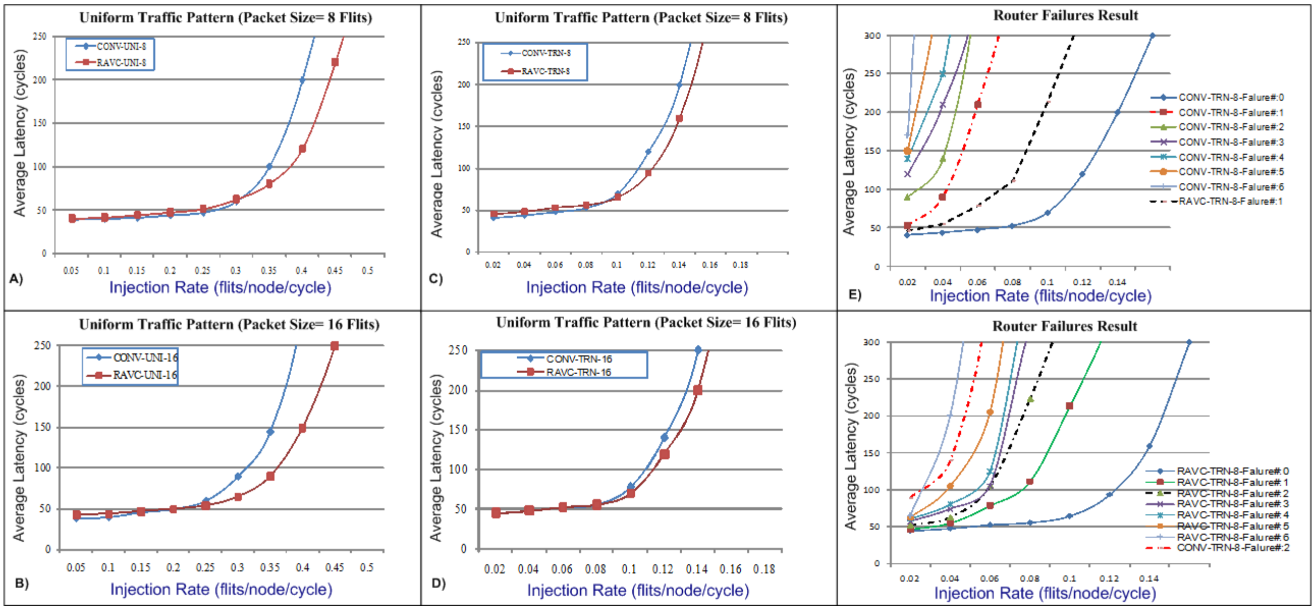


Fig. 7 Simulation Results

6. CONCLUSION

In this paper, we proposed a novel *Reliability Aware Virtual Channel (RAVC)* NoC Router architecture that enables both dynamic VC allocation and reliability aware sharing among input channels. RAVC router provides 7.1% and 3.1 % average latency decrease under uniform and transpose traffic pattern respectively; considering the probability of having failures in network, RAVC router performs more effectively than conventional VC router specially in terms of average packet latency and performance.

On the one hand, assigning virtual channel in a dynamic manner avoids the occurrence of HOL which impede the performance of on-chip network. On the other hand, avoiding faulty routers from packet injections to network reduces the traffic in network due to reliability awareness of RAVC lead to decrease in average packet latency; moreover, as a side bonus, extra buffers which become available in our RAVC routers also cooperates in such a decrease in the average latency

7. REFERENCES

[1] L. Benini and G. D. Micheli, "Networks on Chips: A New SoC Paradigm," IEEE Computer, vol. 35, pp. 70-78, 2002.

[2] S. Li, L. S. Peh, and N. K. Jha, "Dynamic voltage scaling with links for power optimization of interconnection networks," in Proceedings of the 9th International Symposium on High-Performance Computer Architecture (HPCA), pp. 91-102, 2003.

[3] W. Hangsheng, L. S. Peh, and S. Malik, "Power-driven design of router microarchitectures in on-chip networks," in Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), pp. 105-116, 2003.

[4] M. Rezazad and H. Sarbazi-azad, "The effect of virtual channel organization on the performance of interconnection networks, in Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium, 2005.

[5] T. T. Ye, L. Benini, and G. De Micheli, "Analysis of power consumption on switch fabrics in network routers," in Proceedings of the 39th Design Automation Conference (DAC), pp. 524-529, 2002.

[6] H. Jingcao and R. Marculescu, "Application-specific buffer space allocation for networks-on-chip router design," in Proceedings of the IEEE/ACM International Conference on Computer Aided Design (ICCAD), pp. 354-361, 2004.

[7] G. L. Frazier and Y. Tamir, "The design and implementation of a multiqueue buffer for VLSI communication switches," in Proceedings of the IEEE International Conference on Computer Design (ICCD), pp. 466-471, 1989.

[8] L. S. Peh and W. J. Dally, "A delay model for router microarchitectures," IEEE Micro, vol. 21, pp. 26-34, 2001.

[9] Y. M. Boura and C. R. Das, "Performance analysis of buffering schemes in wormhole routers," IEEE Transactions on Computers, vol. 46, pp. 687-694, 1997

[10] W. J. Dally, "Virtual-channel flow control," in Proceedings of the 17th Annual International Symposium on Computer Architecture (ISCA), pp. 60-68, 1990.

[11] M. Lai, Z. Wang, L. Gao, H. Lu, K. Dai, "A Dynamically-Allocated Virtual Channel Architecture with Congestion Awareness for On-Chip Routers," in Proceedings of the 46th Design Automation Conference (DAC), pp. 630-633, 2008.

[12] A. Nicopoulos, D. Park, J. Kim, N. Vijaykrishnan, S. Yousif, R. Das, "ViChaR: A Dynamic Virtual Channel Regulator for Network-on-chip Router, 39th Micro, 2006.

[13] J. Kim, C. Nicopoulos, D. Park, N. Vijaykrishnan, M. S. Yousif, and C. R. Das, "A Gracefully Degrading and Energy-Efficient Modular Router Architecture for On-Chip Networks," in Proc. of the ISCA, 2006.

[14] L. S. Peh and W. J. Dally, "A delay model and speculative architecture for pipelined routers," in Proceedings of the 7th International Symposium on High Performance Computer Architecture (HPCA), pp. 255-266, 2001.