THE P-SINK MODEL WORMHOLE SWITCHING: A NEED FOR NOC

Pankaj Rathi[#1]
[#1]*PhD scholar,*
[#1]*Sangam University ,*
Bhilwara, (Raj), India
[#1]**rathi.panks@gmail.com**

Dr Archana Agrawal[#2]
[#2]*Associate professor,*
[#2]*Sangam University,*
Bhilwara, (Raj), India
[#2]**archana.agrawal@sangamuniversity.ac.in**

Dr. Rajesh Kumar Bathija[#3]
[#3]*Principal*
[#3]*3Shree Digamber Institute of Technology*
[#3]*3Dausa, (Raj), India*
[#3]**sukosh2019@gmail.com**

ABSTRACT

Network-on-Chip is being proposed as a scalable, reusable, global communication platform for future SoC applications. Buses (a single bus, segmented buses or a hierarchy of buses) do not scale well with the system size in bandwidth and clocking frequency. However, a bus is very efficient in broadcasting. Network on Chip (NoC) is a new prototype to intercommunicate modules on the same dice that offer the alternate of conventional buses by routers and interface cards. A router is used when two or more nodes need to be communicated. The greatest challenge in Network-on-Chip is the design of high performance router. The main intent in the design of a Network-on-Chip is to minimize its area and power consumption without affecting the performance that depends on the target technology, ASIC or FPGA. The router is the main representative that affects the performance and functionality of the system as it is liable for delivering the information efficiently and reliably. Network-on-Chip (NoC) is proposed as a systematic approach to address future System-on-Chip (SoC) design difficulties.

In the wormhole switching method, messages are divided into a number of flow control digits commonly called as flits. Every flit may bring a data word. The main advantage of the wormhole switching is that the buffer size can be set as small as possible to reduce the buffering area cost. Because of the aforementioned advantages, namely, better performance, smaller buffering requirement and greater throughput, wormhole switching with lanes is proposed for NoCs. Nevertheless, using wormhole switching for on-chip networks has to minimize the switch design complexity. Since a network on chip is an interconnect scheme using shared wires instead of dedicated wires to pass signals, its cost should be reasonable. Another reason can be seen from the power perspective. Although the communication bandwidth is achievable for future complex SoC integration, the energy consumption will probably be the bottleneck that has to trade off the performance. Therefore, it is crucial to reduce the switch design complexity to decrease the number of gates and switching capacitance in order to shrink energy.

INTRODUCTION

As illustrated in Figure 1, the delivery of packets passes through three stages: flitization, network delivery, and assembly. The flitization is performed at a source node

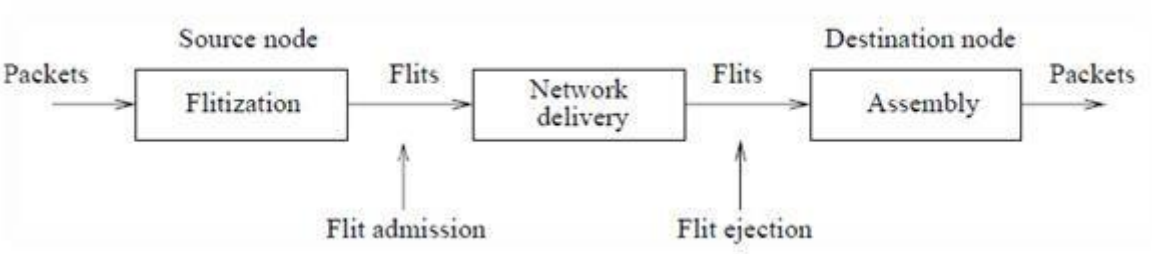

Figure 1: Flit admission and flit ejection

and the assembly, which de-capsulate flits into packets, is conducted by a destination node. Since flits are both the workload of switches and the source of network contentions for shared Virtual Channels (VCs) and Physical channels (PCs) (links), the admission speed, the ejection speed and the balance between admission and ejection are important. To achieve good network utilization and system throughput, flits should be admitted as fast as possible. However, the more flits admitted in the network, the higher contentions may occur, leading to performance degradation. Similarly, a slower ejection of flits will make the ejection become performance bottleneck. An ideal ejection, which ejects flits immediately once they reach their destinations, may over-design the switch.

Background

Performance of NoC as defined in the previous chapter depends on various factors such as network topology, routing strategy and switching technique. Hence, in this chapter, we give a brief theoretical background on

these issues. Firstly, we introduce NoC design considerations. Subsequently, most popular topologies employed in NoCs are reviewed. Then, switching techniques used in NoCs are presented. In the next section we describe widely used routing algorithms and review their performance characteristics briefly. Finally, we present brief information about traffic models for NoCs.

Switching Techniques

Switching techniques define the way and time of connections between input and output ports inside a switch. There exist various switching techniques, but the most popular ones are Circuit Switching, Packet Switching, Virtual Cut-Through Switching and Wormhole Switching.

Circuit Switching

In circuit switching the physical path from source to destination is reserved for the entire duration of data transmission. This is realized by injecting the header flit into the network. The header flit contains the destination address and some additional control information. It moves towards the destination through intermediate routers reserving physical links it has passed. By the time it reaches the destination, the complete path has been reserved and the acknowledgement is sent back to the source. The reserved path may then be released by the destination or by the last bits of message itself.

Packet Switching

In Table 1 a brief summary of switching techniques is given:

Table 1: Summary of switching techniques

Switching technique	Communication entity	Path reservation	Buffer size	Resource utilization
Circuit switching	Flit	Yes	Small	Poor
SAF switching	Packet	No	Large	Good
VCT switching	Packet	No	Large	Good
Wormhole switching	Flit	Yes	Small	Moderate

Routing Algorithms

A routing algorithm determines a path for a packet to reach its destination. It must be decided within each intermediate router, which output channels must be selected for the incoming messages. There are various types of routing algorithms differentiated according to their key characteristics. In accordance with the place where the routing decision is made they may be grouped as centralized, source, and distributed routing algorithms. If an algorithm is centralized the path is chosen by a centralized controller, if it is source routed then the route is determined by the source router prior to sending a packet, in distributed algorithms the path is chosen in a distributed manner at the intermediate routers. According to the way how they choose a path routing algorithms are broadly classified as deterministic and adaptive algorithms. Deterministic algorithms do not take into account network conditions when they take a decision that is why they always supply the same path from source to destination. But, it is not the case for adaptive ones in which network load, traffic conditions, information about available output channels are always taken into consideration.

Every algorithm has different impact on the network. Routing algorithms use a variety of metrics that affect the calculation of the optimal path for a message. Many properties of the interconnection network depend on the routing algorithm used because the complexity of an individual router has a significant impact on the complexity of the entire network. For example, if the routing algorithm is too complicated it will require extra hardware to realize the routing logic, moreover it may take much more time to make decision about the direction where the message should be sent to. It will in turn lead to increase of packet latency. Deadlock, livelock and starvation freedom are also among those properties. This property shows the ability to guarantee that packets will not block or wander across the network forever or permanently stop and never reach its destination.

Deterministic Routing Algorithms

Deterministic routing algorithms always generate the same single routing path for a given pair of source and destination, usually choosing the shortest one. Only the addresses of current and destination nodes are used to compute the path. As messages with the same source and destination always use the same network path, they cannot use alternative paths to avoid blocked channels.

As it was said in the beginning of this section if source routing is used, the path is computed at the source node without considering any information about traffic. Otherwise, if distributed routing is used, routers make unique decisions at the intermediate nodes, based on current and destination node addresses. In both cases channel status is not considered while computing the output channel to be used. Deterministic algorithms should be progressive and profitable, which means that the header should move forward reserving a new channel at each routing operation, under condition that the supplied channel always brings the packet closer to the destination. Thus deterministic routing algorithms use greedy algorithms, always choosing the shortest path.

The most popular deterministic algorithm is known as dimension-order routing. It is based on the idea that some topologies can be decomposed into several orthogonal dimensions, i.e. hyper cubes, meshes and tori. The distance between two nodes in these topologies is computed as the sum of the offsets in all dimensions. The algorithm reduces one of these offsets in each routing step. The offset of the current dimension must be equal to zero before the algorithm considers the offset of the next dimension.

Dimension-order routing is usually used for meshes and hypercubes. In 2D mesh it is called XY- or YX-routing depending on the dimension in which a packet travels first. The algorithm is deadlock-free for n-dimensional hypercubes and meshes, as their channel dependency graph (CDG) is acyclic. CDG is a directed graph where channels are represented by vertices and edges are pairs of channels connected by a routing function. However, the CDG for some topologies has cycles. In order to remove cycles, physical channels may be split into virtual channels.

Most commercially available parallel machines usually use distributed deterministic routing as it is simple and fast. But distributed deterministic routing assumes that the traffic is uniform. In case of non-uniform traffic the performance of distributed deterministic routing in terms of latency and throughput is very poor.

Adaptive Routing Algorithms

Adaptive routing algorithms do not restrict a message to a single path when traveling from the source to the destination [13]. While making a decision the current network conditions are considered. This makes the routing more flexible and reduces unnecessary waiting time delays, providing better fault tolerance.

Adaptive routing algorithms contain two functions: routing and selection. Routing function gives a set of output channels based on the current node and destination node, Selection function selects the most appropriate output channel from that set [23]. The selection function always supplies with a free channel. Thus, messages can follow alternative ways instead of waiting for a busy channel. Non greedy algorithms, which can supply channels that send packets away from its destination, are also allowed.

Adaptive algorithms also allow backtracking technique, which enables the header to backtrack, releasing previously reserved channels, thus systematically searching for appropriate path. For deterministic algorithms backtracking technique is useless as the message will go by the same path again. Adaptive routing algorithms increase routing flexibility but the hardware becomes more complex and slower.

Minimal Adaptive Routing

Minimal routing algorithms always use the shortest path in order to reach destination. While being adaptive they restrict the routing direction in some level. One of the examples for minimal adaptive routing algorithms is double Y-channel routing algorithm. This algorithm divides the network into several sub-networks. The packet is sent via particular sub-network according to the location of destination. The network is usually divided into +X sub-network and -X subnetwork. Here Y dimension has a pair of channels and X dimension has unidirectional channel. If the location of the destination node is bigger than the location of source node, in other words if dx > sx, than the packet is sent through +X sub-network. Otherwise -X subnetwork is used. If dx = sx then either sub-network can be used.

This situation can be shown in the figure shown below where shortest path to reach the destination can be achieved using a routing algorithm called minimal routing algorithm.

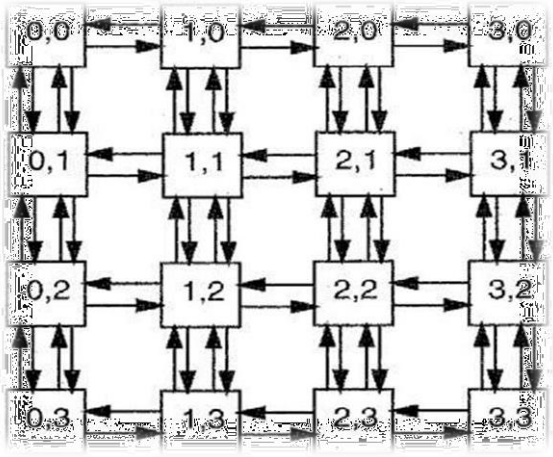

Figure 2: Double Y-channel 2D mesh

The double Y-channel algorithm is minimal and fully adaptive, which means that the packet is delivered through any of the shortest paths. In order to avoid deadlock, channels should be ordered in appropriate manner. The packet that is sent to the destination should path the channel in descending order, in other words, the channel label that was passed must be bigger than the channel label that is going to be passed. This algorithm is impractical when n is large ("n" is the number of dimensions), due to the additional channel requirement.

Non-minimal Adaptive Routing

Unlike the minimal adaptive routing algorithm, where a packet searches only for a shortest path, non-minimal adaptive routing allows the packet to take a longer path if there is no available shortest path.

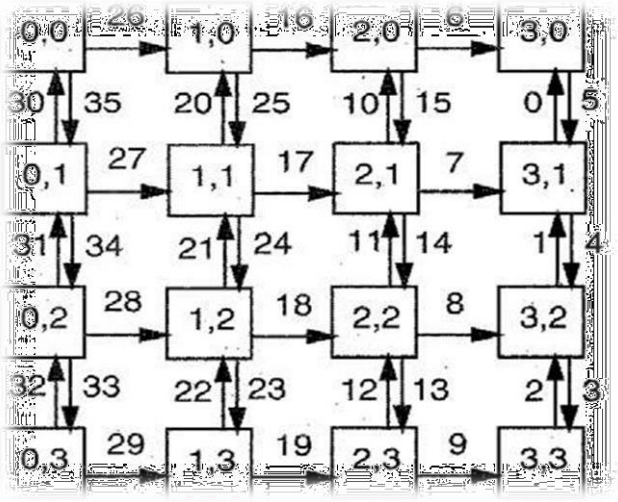

Figure 3: +X sub-network and labeling

This technique is fully adaptive and can be achieved by few more additional channels. Two non-minimal routing algorithms, one of them is static dimension reversal routing algorithm, where every two adjacent nodes are connected by r pairs of channels. Here, the network is divided into r sub networks and the ith sub-network consists of all the ith pair channels. Each packet header stores additional value c which is initially set to zero. The packet can move in any direction in its own sub-network, but when the packet moves from high dimensional channel to low dimensional channel the c field is increased by one. If c reaches the value r − 1 then the packet must switch into the deterministic dimension ordered routing algorithm. The packets can be routed also in reverse dimension order. Parameter restricts the number of times it can happen.

Another algorithm is called dynamic dimension reversal routing algorithm. Here, the channels are divided into two classes: adaptive and deterministic. At first, packets are sent through adaptive channels, moving in any direction. But, when a packet reaches a node, where all output channels are busy by packets with values of c that is smaller or equal to its own, it must switch to the deterministic channels. After entering the deterministic channels a packet cannot return to adaptive channels. The algorithm is deadlock-free.

Turn Model

This algorithm was proposed by Glass and Ni for n-dimensional meshes. It suggests a systematic approach to the development of adaptive routing without additional channels. The algorithm supports both minimal and non-minimal adaptive routing and is partially adaptive.

The packet, while traveling through the network, switches from one dimension to another, in order to reach the destination. Switching from one dimension to another is called turn. If the packet changes its direction without moving to another dimension then this is 180-degree turn. Usually physical channels are split into virtual channels and the packet may move from one virtual channel to another. If it moves from one virtual channel to another and is still in the same dimension and direction then this is a 0-degree turn. Turns can form a cycle. It is known that, deadlock occurs when the packet routes contain turns that form a cycle. The algorithm is based on the idea that, if there is no cyclic dependency between channels then deadlock cannot occur. So, the concept of the algorithm is to prohibit the smallest number of turns such that cycles are prevented. There are several steps that can be useful in developing maximal adaptive routing algorithms:

- Classify channels according to the direction in which they route packets.
- Determine all possible turns that can occur between one direction and another. 0° and 180° turns are not considered.
- Identify the cycles that can be formed.
- Prohibit one turn in each cycle.
- 0° and 180° turns cannot be prohibited as they are needed for non-minimal routing algorithms or if there are multiple channels in the same direction.

One of the examples for this algorithm is west first routing algorithm, where it first routes a packet west, if necessary, and then adaptively south, east and north.

Figure 3.9 shows the turns that are prohibited. Thus, the packet can make only six turns and two turns to the west are not allowed. Therefore, to travel west, a packet must begin in that direction. For minimal routing, the algorithm is fully adaptive if the destination is on the right side of the source, otherwise, it is deterministic. For non-minimal routing the algorithm is adaptive in either case.

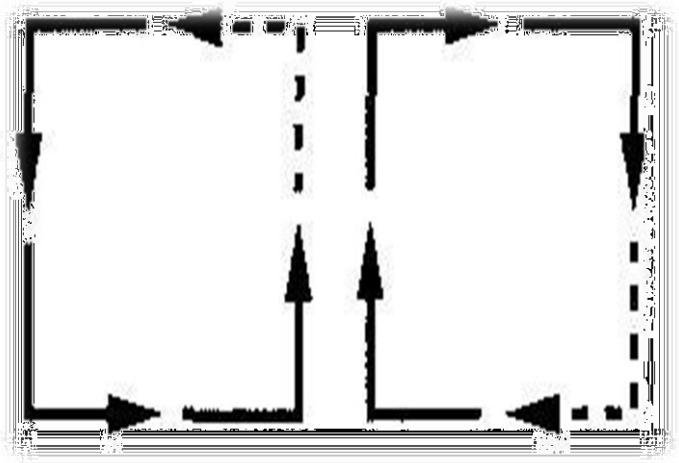

Figure 4: Six turns (solid arrows) allowed in west-first routing

Randomized Routing Algorithms

DOR algorithm is considered to be one of the most popular deterministic routing algorithms due to its simplicity for implementation and good performance according to average packet delay and throughput metrics. Valiant and Brebner [32] proposed one of the best known randomized algorithms named Valiant. This algorithm has two phases. In both of the phases it uses dimension-order routing. In the first phase a random node is selected, and a packet is sent there. In the second phase, it routes the packet from that random node to its destination. Valiant is non-minimal and tries to avoid congestion in the network. Packets with the same source and destination are unlikely to take the same path, as they will have different intermediate nodes which are selected randomly. Several sets of buffers are needed in order to avoid deadlocks. In a mesh topology, it requires two sets of independent buffers per communication link. In Valiant routing algorithm livelocks cannot occur, as the packet reaches its destination eventually, but it may take longer path than actually required. The path taken is a drawback of this algorithm, because its length increases according to the topology being used. For example, in a mesh, the packet‟s path may be doubled. This means that the Valiant routing results in longer routing times. Hence, the demand on network bandwidth is also doubled. Another routing algorithm which uses

randomization is Randomized, Oblivious, Multi- phase, Minimal (ROMM) algorithm. It inherits minimality of DOR, and randomization of Valiant. Minimality assures that the path taken by a packet will be minimal. Randomization is used to assure that packets with same source and destination will not take the same path. These properties are combined to avoid congestion. This algorithm is oblivious, thus, making it easy to implement in contrast to adaptive algorithms which are more complex As ROMM algorithm uses only minimal paths, it is constrained in randomization unlike Valiant which uses full randomization. ROMM selects random nodes within the range of a minimal path a message is required to follow in order to reach the destination. It routes a packet in p-phases. A p − phase ROMM algorithm has p − 1 randomly selected nodes Z1, Z2, . . . , Zp−1 between source and destination, such that, those Zi,,s must be on minimal path from source to destination.

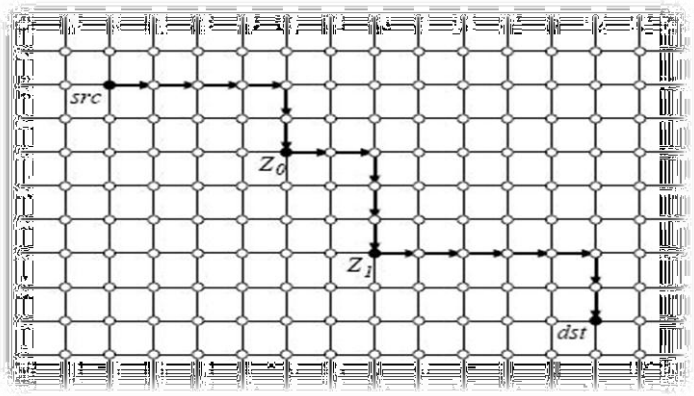

Figure 5: A possible path from src to dst using 3-phase

ROMM on a 2D mesh Figure 3.10 illustrates a possible path from source to destination in a 3-phase ROMM. Here, source node determines the path that would be taken by message using dimension-order routing and chooses a random node that lies within that path, in our case Z0. Then message is routed from src to Z0 using dimension-order routing. Now, Z0 becomes a source node and chooses another random node that lies along the minimal path from Z0 to dst in this case Z1, and send a message using dimension order routing. Z1 in its turn routes a message to dst again using dimension-order routing.

If p phases are used then ROMM algorithm requires p − virtual channels for wormhole routed mesh network, in order to avoid deadlocks. The algorithm becomes very costly as p increases, as it increases memory usage adding virtual channels and complicates routing with additional logics required to manage virtual channels.

Comparison of Routing Algorithms

In this section we introduce general comparison of routing algorithms mentioned in previous sections as it is indicated in the literature. Deterministic algorithms in comparison to adaptive ones achieve higher throughput behavior under uniform traffic pattern in 2D mesh with the same number of VCs and have lower latencies at higher throughputs. However, adaptive algorithms outperform them when used in 2D torus. Deterministic algorithms suffer from channel underutilization while adaptive algorithms distribute the traffic more uniformly across the network and do not have such a shortcoming. Under non-uniform traffic pattern adaptive algorithms perform much better than deterministic ones in terms of throughput and latency. It is due to their ability to provide several routes between the same pair of source and destination.

The Wormhole Switch Architecture

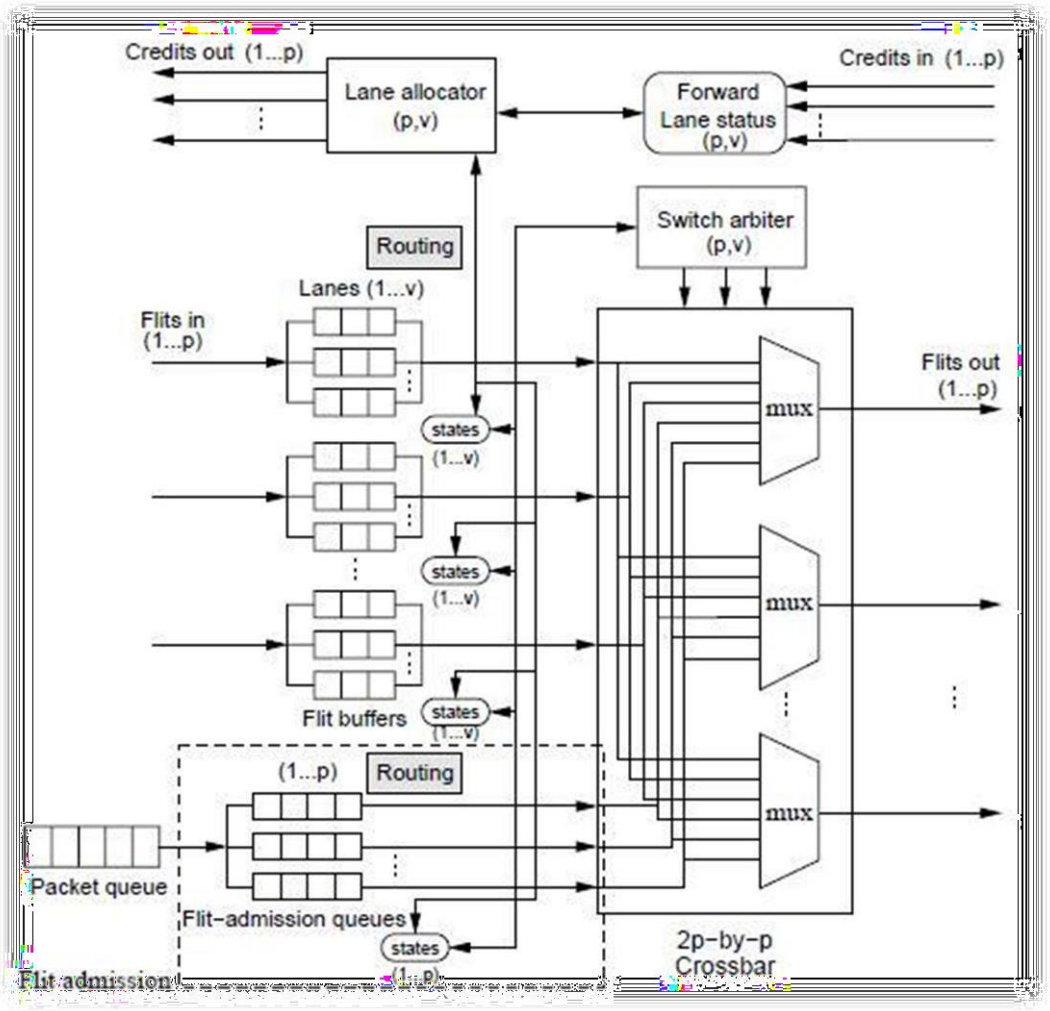

Figure 6: A canonical wormhole lane switch (no ejection)

Figure 6 illustrates canonical wormhole switch architecture with virtual channels at inputs. It has “p” physical channels (PCs) and “v” lanes per PC. It employs credit-based link-level flow control to coordinate packet delivery between switches.

A packet passes the switch through four states: routing, lane allocation, flit scheduling, and switch arbitration. In the routing state, the routing logic determines the routing path the packet advances. In the state of lane allocation, the lane allocator associates the lane the packet occupies with an available lane in the next switch on its routing path, i.e., to make a lane-to-lane association. If the lane-to-lane association succeeds, the packet enters into the scheduling state. If there is a buffer available in the associated downstream lane, the lane enters into the switch arbitration. The first level of arbitration is performed on the lanes sharing the same physical channel. The second level of arbitration is for the crossbar traversal. If the lane wins the two levels of arbitration, the flit situated at the head of the lane is switched out. Otherwise, the lane returns back to the scheduling state. The lane to-lane association is released after the tail flit is switched out. Credits are passed between adjacent switches in order to keep track of the status of lanes.

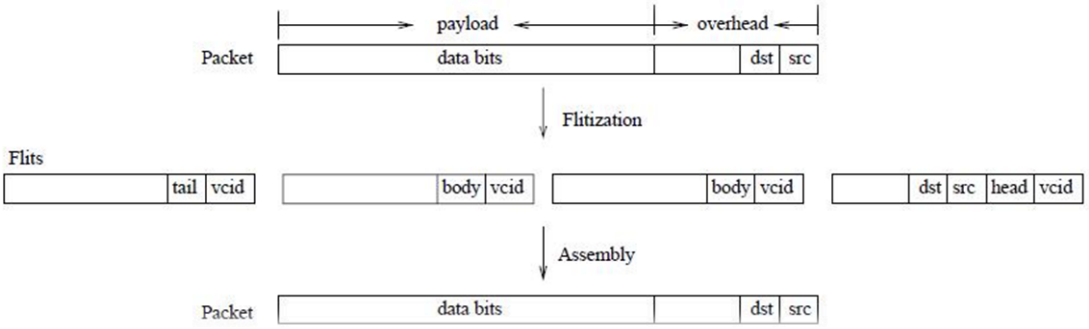

Figure 7 Flitization and assembly

With wormhole switching, a packet is decomposed into a head flit, zero or more body flit(s), and a tail flit. A single packet flit is also possible. In Figure 4.3, a packet is encapsulated into four flits, where vcid is the identity number of a virtual channel or lane. A flit differs from a packet in that (1) a flit has a smaller size; (2) only the head flit carries the routing information such as source/destination address, packet size, priority etc. As a consequence, the routing and lane allocation can only be performed with the head flit of a packet. Once a lane-to-lane association is established by the head flit of the packet, the rest of flits of the packet inherit this association. After the tail flit leaves, the lane-to-lane association is torn down. This is to say, a lane is allocated at the packet level, i.e., packet-by-packet while a link is scheduled at the flit level, i.e., flit-by-flit since the flit scheduling as well as the switch arbitration is performed on per flit basis. As the head flit advances, lanes are associated like a chain along the routing path of the packet, the rest of flits are pipelined along the chain path. Carrying routing information only in the head flit of a packet leaves more space for payload. However, flits belonging to different packets cannot be interleaved in associated lane(s) since flits other than head flits do not contain routing information. To guarantee this, a lane-to-lane association must be one-to-one, i.e., unique at a time. An upstream lane cannot associate with two or more downstream lanes simultaneously (one-to-many association). A downstream lane cannot be allocated to two or more upstream lanes at the same time (many-to-one association). Both one-to-many and many-to-one associations must be forbidden. Figure 4.4 illustrates lane-to-lane associations. The one-to-many association leads to that the flits from lane 2 in switch 2 are delivered to lane 2 and 3 in switch 3. The many-to-one association results in that lane 3 in switch 2 will receive flits from lane 1 and 2 from switch 1. Obviously the one-to-many association and many-to-one association result in that the integrity of a worm (the flit sequence of a packet) is destroyed. It becomes impossible either to route the flits of a packet or assemble the flits into a packet. In a word, only one-to-one association is permissible.
Figure 4.4: Lane-to-lane associations Flit Admission

The decoupled admission

We assume that a switch receives packets injected via a packet FIFO. A packet is first flitized into flits that are then stored in flit FIFOs, called admission queues, before being admitted into the network. There are various ways of organizing the packet queue and the admission queues. In Figure 4.5(a), flit-admission queues are organized as a FIFO. In Figure 4.5(b) and 4.5(c), they are arranged as p parallel FIFO queues (p is the number of PCs). Figure 4.5(a) and 4.5(b) allow at maximum one flit to be admitted to the network at a time while Figure 4.5(c) allows up to p flits to be admitted simultaneously. We adopt the organization of flit- admission queues in Figure 4.5(c) for our further discussions since it allows potentially higher performance.

Figure 8: Organization of packet and flit-admission queues

Figure 8 also illustrates the organization of p flit-admission queues in the switch architecture. Initially, packets are stored in the packet queue. When a flit admission queue is available, a packet is split into flits which are then put into the admission queue. A flit- admission queue transits states similarly to a lane to inject flits into the network via the crossbar. The routing is performed after flitization. By this scheme, each flit-admission queue is connected to p multiplexers. Flits from a flit-admission queue can be switched to anyone of the p output PCs. To implement this scheme, the crossbar must be fully connected, resulting in a port size of2pp. Since the flit-admission queues are decoupled from the output PCs, we call this admission scheme decoupled admission

The coupled admission

We propose a coupling scheme for the flit-admission that can sharply decrease the crossbar complexity, as

sketched in Figure 6. Just like the decoupled admission, it uses p admission queues, but one queue is bound to one and only one multiplexer for a particular output PC. Due to this coupling, flits from a flit-admission queue are dedicated to the output PC. Consequently, an admission queue only needs to be connected to one multiplexer instead of p multiplexers. The size of the crossbar is sharply decreased from 2pp to (p+1) p, as shown in Figure 6. The number of control signals per multiplexer is reduced from d log (2p) e to d log (p + 1) e for any p> 1.

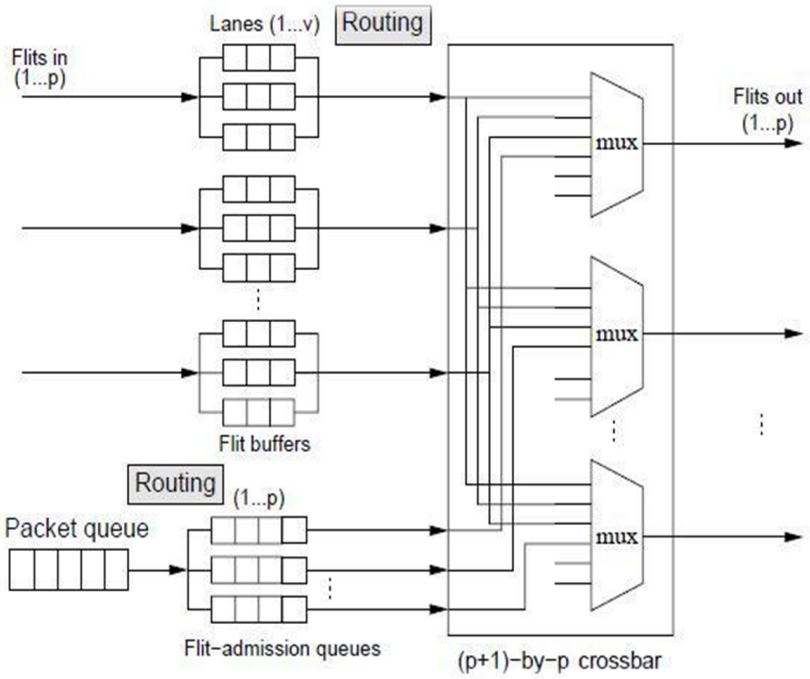

Figure 4.6: The coupled admission sharing a (p+1)-by-p crossbar

In order to support the coupling scheme, the routing must be performed before flitization instead. By a routing algorithm, the output physical channel, which a packet requests can be determined. Hence, the corresponding admission queue is identified. One drawback due to the coupling is that the head-of-line blocking may be worse if the packet injection rate is higher. Specifically, if the head packet in the packet queue is blocked due to the bounded number and size of the admission queues, the packets behind the head packet are all unconditionally blocked during the head packet‟s blocking time. By the decoupled admission, the head-of-line blocking occurs when the four flit-admission queues are fully occupied. With the coupled admission, this blocking occurs when the flit-admission queue the present packet is aimed to full.

Flit Ejection

The ideal sink model

An ideal sink model is typically assumed for a wormhole switch [28]. With an ideal ejection, flits reaching their destinations are ejected from the network immediately, emptying the lane buffers they occupy. An ideal flit-ejection model is drawn in Figure 9. A flit sink is a FIFO receiving the ejected flits. Each lane is connected to a sink and the crossbar via a de- multiplexer.

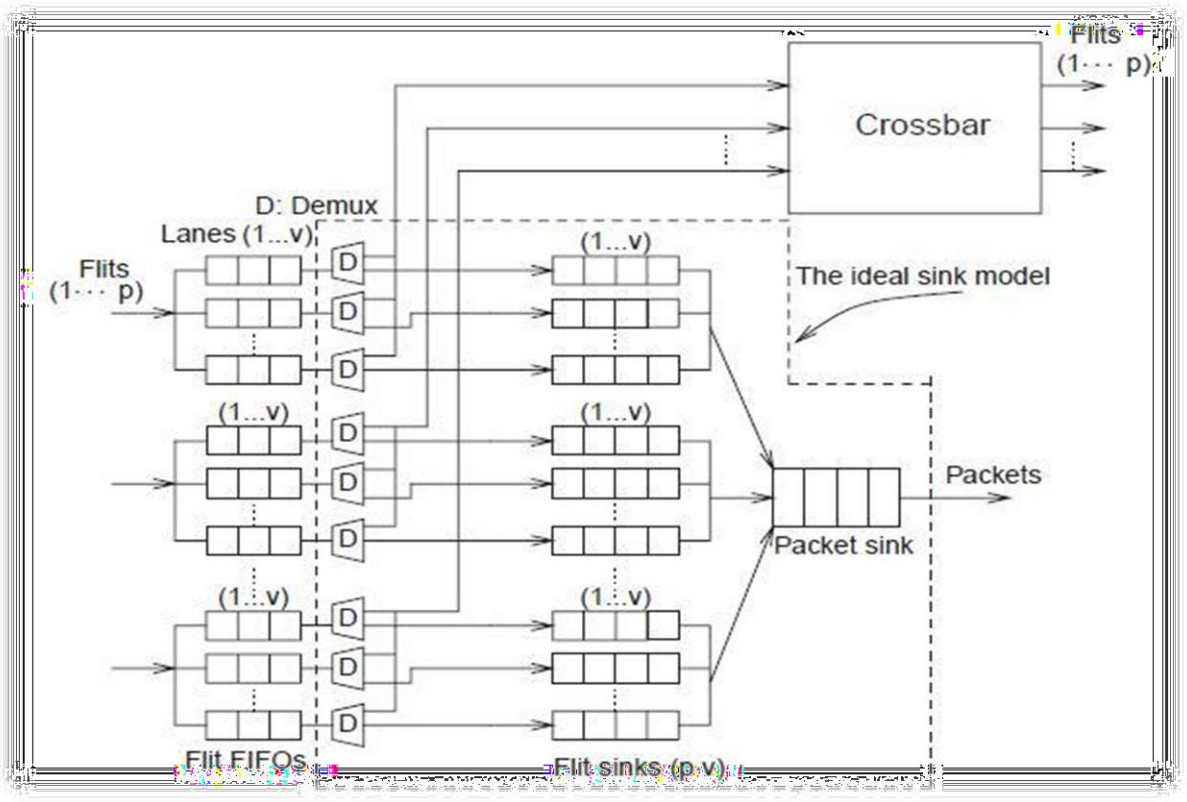

Figure 9: The ideal sink model

To incorporate ejection, the lane state is extended with a reception state in addition to the four states. If the routing determines that the head flit of a packet reaches its destination, the lane enters the reception state immediately by establishing a lane-to-sink association. Since flits from different packets cannot interleave in a sink queue, there must be p v sink queues, each of them corresponding to a lane, in order to realize an immediate transition to the reception state. Assuming that one sink takes the flits of one packet, the depth of a sink is the maximum number of flits of a packet. After the lane transits to the reception state, the head flit bypasses the crossbar and enters its sink. The subsequent flits of the packet are ejected into the sink immediately upon arriving at the switch. When the tail flit is ejected, the lane-to-sink association is freed. This model is beneficial in both time and space. Although a head flit may be blocked by flits situated in front of it in the same lane, a non-head flit reaching its destination neither waits to be ejected (time) nor occupies a flit buffer (space) once the lane is in the reception state. Moreover, it does not interfere with flits buffered in other lanes from advancing to next switches downstream (because of the use of the demultiplexers, one PC always allows one flit from a lane to be switched via the crossbar without interference with sinking flits from other lanes.). Upon receiving all the flits of a packet, the packet is composed and delivered into the packet sink. If the packet sink is not empty, the switch outputs one packet per cycle from the packet sink in a FIFO manner.

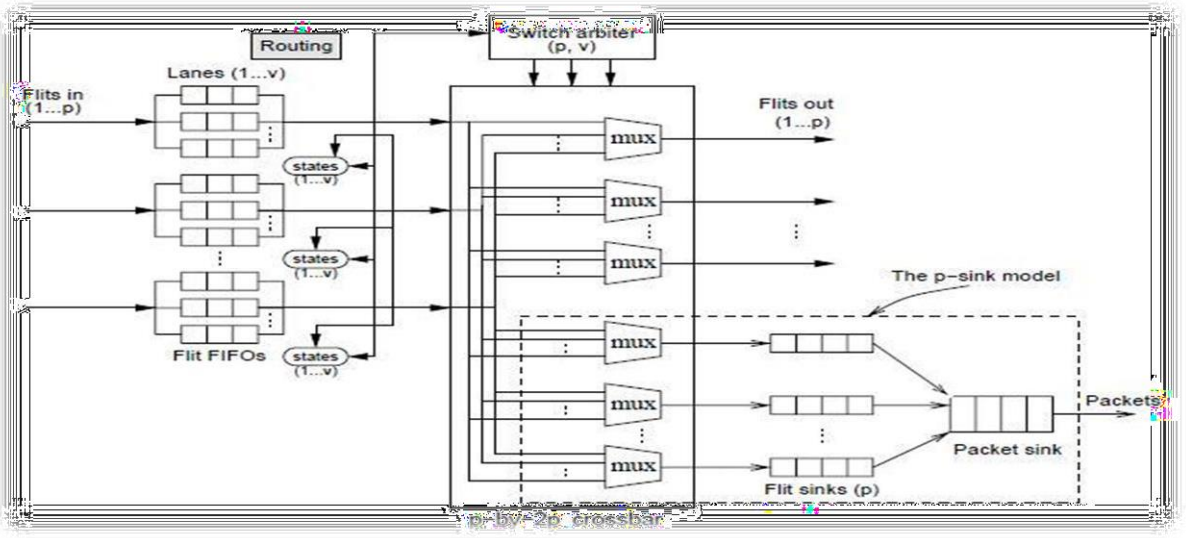

Figure 10: The p-sink model

The p-sink model

Since the maximum number of flits to be ejected per switch per cycle is p, we can use p sink queues instead of p v sink queues to eject flits to avoid over-design. Moreover, in order to have a more structured design, we could connect the p sink queues to the crossbar, as illustrated in the dashed box of figure 10.

To enable ejecting flits by the p-sink model, we now extend the four lane states with two new states: an arriving and a reception state. If a head flit reaches its destination, the lane the flit occupies transits from the routing to the arriving state. Then it will try to associate with an available sink, i.e., to establish a lane-to-sink association. If the association is successful, the lane enters the reception state. Subsequently the other flits of the packet follow this association exactly like flits advancing in the network. Upon the tail flit entering the sink, the association is torn down. If the lane-to-sink association fails; the head flit is locked in place holding the lane buffer. To speed up flit ejection, the contentions for the crossbar input channels and crossbar traversal are arbitrated on priority. A lane in a reception state has a higher priority than a lane in a state for forwarding flits. The drawback in this sink model is the increase of blocking time when flits reach their destinations. First, the lane-to-sink association may fail since all sink queues might be in use. Second, only one lane per PC can win arbitration to an input channel of the crossbar due to sharing the input channel for both advancing flits and ejecting flits. In case of more than one lane of a PC are in the reception state, only one lane can use the channel.

Table 2: Cost of the sink models

pX1Mux	4X2Demux	Flit	Sink
The ideal model	–	p.v	p.v
The p-sink model	P	–	P

To implement this p-sink model, the crossbar must double its capacity from p-by-p (p p 1 multiplexers) o p-by-2p (2p p 1 multiplexers), as illustrated in Figure 10. The number of control ports of the crossbar is doubled proportionally. Table 2 summarizes the number of each component to implement the sink models. As can be seen, the ideal sink model requires p v flit sinks while the p-sink model uses only p flit sinks. With the p-sink model, the sink number becomes independent of v, implying that the buffering cost for flit sinks is only 1=v as much as the ideal ejection model.

Conclusion and future scope

A complete structure of wormhole switching is explained. A new p sink model is proposed for wormhole routing for flit

ejection. The model is better thasn the ideal model. The future scope can be that we can implement this model into FPGA.

REFERENCES

[1] Luca Benini and Giovanni De Micheli, "Networks on Chips: A New SoC Paradigm, Computer"(1), pp. 70-78, *IEEE Computer Society, Los Alamitos, CA, USA,* 2002

[2] Sumant Sathe, Daniel Wiklund and Dake Liu, "Design of a Switching node (router) for On-chip Networks", *Linkoping university, Swedon*, pp. 75-78, *IEEE,* , 2003

[3] Wen-Fong Wang and Zhi-Chun Jao, "Design of a Partially Buffered Crossbar Router for mesh-based Network-On-chips", *International Conference on High Performance computing and communications*, pp. 772-777, *IEEE,* 2011

[4] Ling Wang, Jianye Hao and Feixuan Wang, "Bus-based and NoC performance emulation and comparison", *Sixth International Conference on Information Technology: New Generations*, pp. 855-858 *IEEE*, 2009

[5] Yahia SALAH, Mohamed ATRI and Rached TOURKI, "Design of a 2D Mesh-Torus router for Network-on-Chip", *International Symposium on Signal Processing and Information technology,* pp.626-631, *IEEE,* 2007

[6] Geng Luo-Feng, Du Gao-Ming, Gao Ming-Lun, Hou Ning and Song Yu-Kun, "Design and Performance Evaluation of a 2D-Mesh Network-on-Chip Prototype using FPGA", *Work supported in grant by NSFC grant 60576034,* pp. 1264-1267 *IEEE,* 2008

[7] Andrew A. Chien, "A cost and Speed Model for k-ary n-cube Wormhole routers", *In Proceedings of Hot Interconnects'93,* 1993, *Palo Alto, California*

[8] Jingcao Hu and Radu Marculescu, "Application-Specific Buffer Space Allocation for Networks-on-Chip Router Design", ICCAD "04: *Proceedings of the 2004 IEEE/ACM International conference on Computer-aided design*, pp. 354-361, 2004

[9] Dalu Yi, Lixin Yu, Haidong Fei and Xingyou Wang, "SpaceWire Standard and Improved Wormhole Router Design", pp. 1-8, *IEEE* 2012

[10] Chunjing Mao, Yong Guan and Jie Zhang, "on-Board Spacewire Router for Space Mission", Asia-Pacific conference on Information Processing, *IEEE,* pp. 525-528, 2009

[11] Srinivasan Murali, Paolo Meloni, Federico Angiolini, David Atienza, Salvatore Carta, Luca Benini, Giovanni De Micheli and Luigi Raffo., "Designing Application-Specific Networks on Chips with Floorplan Information", ICCAD"06: *Proceedings of the 2006 IEEE/ACM International Conference on Computer-aided design,* pp. 355-362, 2006

[12] Kees Goossens, John Dielissen, and Andrei Radulescu, "The ethereal Network on Chip:

Concepts, Architectures, and Implementations", *IEEE Design and Test of Computers*, pp. 414-421, 2005

[13] Masakazu Taniguichi, Hiroki Matsutani and Nobuyuki Yamasaki, "Design and Implementation of On-chip Adaptive Router with Predictor for Regional Congestion", *17th IEEE International Conference on Embedded and Real-time Computing Systems and Applications,* pp. 22-27, 2011

[14] Daniel U. Becker and William J. Dally, "Allocator Implementations for Network-on- Chip Routers", *Proceedings of the 2009 ACM/IEEE Conference on High Performance Computing, Networking, Storage and Analysis*, 2009

[15] William J Dally and Charles L Seitz,, "The Torus Routing Chip, Journal of Parallel and Distributed Computing (4)", pp. 187-196, 1986, *IEEE*

[16] P. Guerrier and A. Greiner. "A generic architecture for on-chip packet-switched interconnections, Design, Automation and Test" in *Europe Conference and Exhibition*, March 2000, pp. 250-256

[17] S. Kumar and et al., A. Hemani, "A network on chip architecture and design methodology", *Proceedings of IEEE Computer Society Annual Symposium on VLSI, 2002 April 25- 26*, pp. 105-112, 2002

[18] W. J. Dally and C. L. Seitz, "Deadlock-Free Message Routing in Multiprocessor Interconnection Networks", *IEEE Transactions on Computers. May 1987, Vols. C-36, 5*, pp. 547-553

[19] J. Duato, S. Yalamanchili, L. Ni, "Interconnection Networks: An Engineering Approach s.l.: Morgan Kauffman", 2002.

[20] Ananth Grama, "Introduction to Parallel Computing", Addison-Wesley, Harlow, ISBN 0- 201-64865-2, 2003

[21] R. I. Greenberg, G. Lee, "An improved analytical model for wormhole routed networks with application to butterfly fat-trees", *Proceedings of the 1997 International Conference on Parallel Processing*, 1997, pp. 44-48

[22] W. J. Dally, "Virtual-channel flow control", *IEEE Transactions on Parallel and Distributed Systems*, pp. 194-204, March 1992.

[23] Terrence Mak, Peter Y.K. Cheung, Kai-Pui Lam and Wayne Luk, "Adaptive Routing in Network-on-Chips using a Dynamic-Programming network", *IEEE Transactions on Industrial Electronics, Vol. 58, No. 8,* August 2011, *IEEE*, pp. 3701-3716

[24] GE Fen and WU Ning, "Genetic Algorithm based Mapping and Routing Approach for Network on chip Architectures", *Chinese Journal of Electronics, Vol.19, No.1*, and January, 2010

[25] Wan-Ting Su, Jih-Sheng Shen and Pao-Ann Hsiung, "Network-on-Chip Router Design with Buffer-Stealing", pp. 160-164, *IEEE*, 2011

[26] Arun Janarthanam, Karen A. Tomoko, "MoCSYS: A Multi-Clock Hybrid two-Layer Router

Architecture and Integrated Topology Synthesis Framework for System-Level Design of FPGA based On-chip Networks", *21st International Conference on VLSI Design*, pp. 397-402, *IEEE*, 2010

[27] Son Truong Nguyen and Shigeru Oyanagi, "The Design On-the-fly Virtual Channel Allocation for Low Cost High Performance On-chip Routers", pp. 88-94, *IEEE. 2010 First International Conference on Networking and Computing*

[28] Luca Sterpone, Davide Sabena and Matteo Sonza Reorda, "A New Fault Injection Approach for Testing Network-on-Chips", *20thEuromicro International Conferennce on Parallel, Distributed and Network-Based Processing*, pp. 530-535, *IEEE*, 2012

[29] Pooria M.Yaghini, Ashkan Eghbal, S.A. Asghari and H.Pedram, "Power Comparison of an Asynchronous and synchronous Network on Chip Router", pp.242-246, *IEEE*, 2009

[30] Haytham Elmilgi, Ahmed A. Morgan, M.Watheq El-Kharashi and Fayez Gebali, "Performance Analysis of Networks-on-Chip routers", pp. 232-236, *IEEE*, 2007

[31] L. G. Valiant, G. J. Brebner, "Universal schemes for parallel communication", *Proceedings of the 13th annual ACM symposium on Theory of computing*, 1981, pp. 263 - 277