VIX: Virtual Input Crossbar for Efficient Switch Allocation

Supriya Rao, Supreet Jeloka, Reetuparna Das, David Blaauw, Ronald Dreslinski, and Trevor Mudge

Department of Computer Science and Engineering, University of Michigan
{supriyar, sjeloka, reetudas, blaauw, rdreslin, tnm}@umich.edu

ABSTRACT

Separable allocators in on-chip routers perform switch allocation in two stages that often make uncoordinated decisions resulting in sub-optimal switch allocation. We propose Virtual Input Crossbars (VIX), where more than one virtual channel (VC) of an input port is connected to the crossbar. VIX improves switch allocation by allowing more than one input VC of an input port to transmit flits in the same cycle. Also, more input VCs can participate in the output arbitration, reducing the chances of uncoordinated decisions. VIX improves network throughput by more than 15% for the topologies studied without affecting the router critical path.

Categories and Subject Descriptors

C.1.2 [**PROCESSOR ARCHITECTURES**]: Multiprocessors—*Interconnection Architectures*

General Terms

Design,Algorithms

Keywords

network-on-chip, throughput, switch-allocation

1. INTRODUCTION

Processors with 10s of cores are already available commercially. A packet-based on-chip network with a regular topology is an attractive solution for connecting these large number of cores and the associated on-chip structures, as they are scalable with predictable electrical properties [7]. An efficient on-chip network design could have a significant impact on a many-core processor's performance and power consumption.

Routers in an on-chip network play a central role in determining its efficiency. A router's crossbar (switch) connects a set of input ports to its output ports. A key functionality of a router is to decide every cycle which of its output ports gets allocated to which of its input ports. This functionality is performed by the switch allocation unit of the router. Thus, the switch allocator primarily determines the utilization of a router's output links and therefore has a significant impact on overall network performance.

Optimal switch allocation for an on-chip network without virtual channels is easier to realize: each output port has an *output arbiter* which chooses from a set of input ports that are requesting for it. However, optimal switch allocation for an on-chip network with virtual channels (VCs) is considered to be a challenging problem, and there has been significant research to address it [14, 1, 21, 4, 15, 5].

The challenge in networks with VCs is that each input port contains one or more input virtual channels that could each request for a different output port. However, in conventional router designs, since there is only one input to the crossbar per input port, only one virtual channel in an input port can transmit a packet in a cycle. To select an input VC for each input port, conventional router designs employ an additional set of *input arbiters*.

The above two phase switch allocator is referred to as the input-first separable allocator. In the first phase, the input arbiters choose one virtual channel for each input port. The winners compete in the second phase, where the output arbiters choose one input port for each output port. This complexity-effective design is widely assumed in NoCs as it can meet the tight timing constraints of an on-chip router.

However, a separable allocator suffers from two problems that lead to sub-optimal switch allocation. We consider an optimal switch allocator to be one that guarantees that an output port of a router is utilized in a cycle as long as there is at least one input virtual channel requesting for it.

The first problem is the sub-optimal matching problem. The input arbiters, while selecting an input VC for each input port, fail to coordinate between each other. As a result, often two input ports may end up choosing input VCs that request the same output port, when that output port can serve only one input in a cycle. Had one of those two input ports selected a VC with a different output port request, both input ports could have been profitably allocated to different output ports.

Past work on improving switch allocation has focused on solving the above sub-optimal matching problem. They employ iterative algorithms to allocate as many output ports as possible in a cycle [14, 21]. Unfortunately, these iterative solutions come at the cost of increased delay and area, and may not be able to operate within the tight timing constraints of the router.

The second problem that leads to sub-optimal switch allocation is the input port constraint that dictates that only one VC in an input port can transmit flits in a cycle. As a result, when the only requestors for two output ports are from one input port, it is guaranteed that one of those outport ports would be left idle. Thus, even if we are to implement an optimal matching solution [8], we could still have sub-optimal switch allocation, where one or more outports ports are left idle in spite of some input VCs requesting them.

In this paper, we make a key observation that the root cause of the above two main switch allocation problems in NoC routers lies in unnecessarily restricting VCs of an input port from access to the crossbar switch. Instead of over-burdening the already complex switch allocation stage to improve its matching efficiency [14, 21], we propose the Virtual Input Crossbar (VIX) that connects to more than one VC per input port. While this increases the number of inputs to the crossbar, it is feasible without increasing the router's cycle time, as the crossbar is not in a router's critical path.

VIX addresses both problems in switch allocation we discussed earlier. As VIX connects to more than one VC per input port, it directly addresses the input port constraint problem. VIX also improves the matching efficiency. The probability that an output request from an input VC gets killed in the first phase of separable allocation reduces as we expose more input VCs to the second phase. In one extreme, if we connect all the input VCs of an input port to the VIX, we can not only achieve optimal matching but also guarantee optimal switch allocation. In practice, we limit VIX to two virtual inputs per input port in order to bound the complexity of crossbar within a router's timing constraints.

We study VIX and the different switch allocation techniques through detailed cycle accurate performance simulations, circuit-level delay analysis of routers and network power models. We model a 64 node network-on-chip. We study statistical traffic, and also 35 different benchmarks with multiprogrammed workloads. We find that VIX improves the effectiveness of separable allocators significantly, making them comparable or better than more expensive and slower allocators.

Our evaluations show that VIX based allocation improves network throughput by 16% for a mesh topology. VIX improves network throughput by 15% when compared to wavefront switch allocation, while having 39% lower delay. We also study the applicability of VIX to higher radix topologies. VIX improves network throughput by

DAC 2014, June 1-5, 2014, San Francisco, California, USA.

ACM 978-1-4503-2730-5/14/06 ...$15.00.

15% for concentrated mesh topology and by 17% for flattened butterfly topology. Higher performance from VIX can be used to reduce network buffers. VIX can reduce the network buffers by 33% while still improving network throughput by 10%. For application workloads, VIX improves system performance by 5% on average.

2. VIRTUAL INPUT CROSSBARS

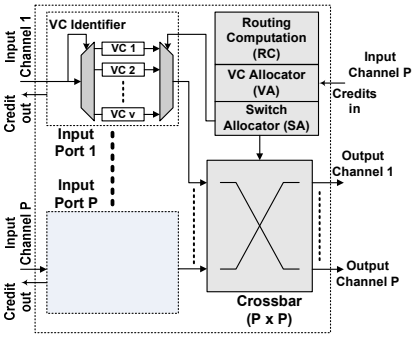


Figure 1: Router Architecture.

2.1 Description

The routers are responsible for communicating packets between the source and destination nodes connected by the on-chip network. Each packet consists of several smaller units of flow control called flits. Every flit travels hop by hop from one router to another unitl it reaches its destination. A generic router architecture is shown in Figure 1. This router has P input ports and P output ports. The Route Computation unit, RC determines the output port for each packet. The Virtual channel Allocation (VA) stage decides the winner amongst all the packets requesting access to the same output VC in the downstream router. The Switch Allocation (SA) unit decides the winning flit for each input port and each output port that can proceed to crossbar traversal. Typically crossbars in routers have as many inputs as the number of input ports. A multiplexer is used to connect one virtual channel from v virtual channels in an input port to the corresponding input in the crossbar. We can potentially increase the inputs to the crossbar to provide more virtual inputs. For instance, the crossbar can have $2P$ inputs. Thus each input port has 2 virtual inputs corresponding to it. The virtual channels are partitioned into two virtual input sets. A multiplexer can be used to connect one virtual channel from $v/2$ virtual channels to the corresponding virtual input of the crossbar as shown in Figure 2.

2.2 Switch Allocation with VIX

Our goal is to provide high matching efficiency with complexity comparable to separable allocators. VIX provide opportunities to expose more conflict free inputs during switch allocation and hence improve matching efficiency of separable allocators. The virtual inputs provide two advantages. First, now more than one virtual channel can transmit flits from one input port to different output ports. Figure 4 illustrates this advantage for a 5-port mesh router with 4 virtual channels. There are two packets in the $West$ port of the router in $VC0$ and $VC2$. The packet in $VC0$ is requesting for $Local$ port and the packet in $VC2$ is requesting for $East$ port. In absence of virtual inputs only $East$ output port is allocated and one flit is transferred. With virtual inputs, its possible to transfer flits from both $VC0$ and $VC2$ in the same cycle. Thus both $East$ and $Local$ output ports are allocated and two flits are transferred.

The second advantage of virtual inputs is better matching efficiency because more input requests are exposed for output arbitration. This reduces the probability that different input arbiters pick the same output. Figure 5 illustrates this advantage. In absence of virtual inputs (Figure 5 (a)) both $West$ and $South$ input port arbiters pick $East$ output port. $East$ output port can pick only one, which is $West$, resulting in only one flit transfer. In presence of virtual inputs (Figure 5 (b)), the input arbiters for each virtual input in $South$ pick different outputs, $North$ and $East$. This leads to a better matching, resulting in three flit transfers.

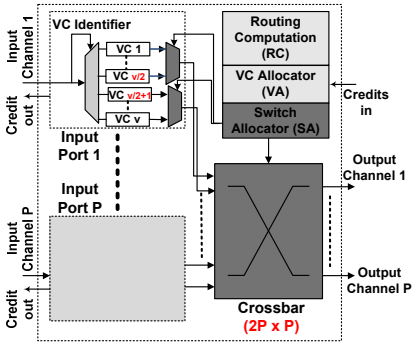


Figure 2: Router Architecture with VIX

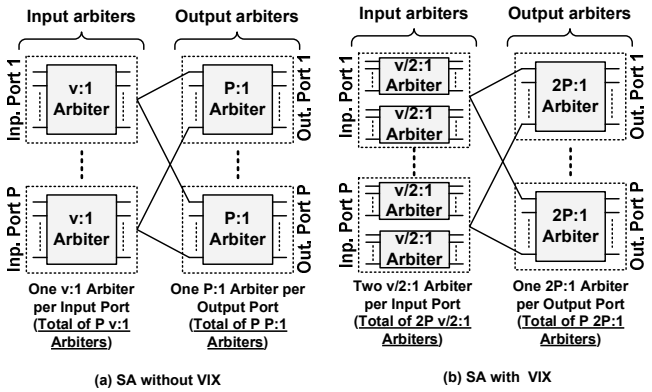


Figure 3: Switch Allocation

2.3 Virtual Channel Assignment with VIX

The Virtual Channel Allocation stage decides the winner among all the packets requesting access to the same output VC at the downstream router. Before Virtual Channel Allocation every input VC is assigned an output VC. Typically the output VC with maximum number of free flit-buffers is assigned. VIX can be optimized by intelligent assignment of output virtual channels to input virtual channels. This is because virtual channels are now divided among virtual inputs to the crossbar creating *sub-groups* of virtual channels, each sub-group connected to a specific virtual input to the crossbar. For example, for the router shown in Figure 2, there are two sub-groups of virtual channels at each port, and each sub-group has $v/2$ virtual channels. Packets can be assigned to different sub-group of output virtual channels (and hence different virtual inputs in the downstream router) based on the direction of the output port at the downstream router. By using this dimension information, there will be fewer output port conflicts because requests to different output ports will likely be generated from virtual channels belonging to different virtual inputs. The input requests are also load balanced among the sub-groups of virtual channels to make sure that the different virtual inputs connected to the crossbar always have requests. This exposes majority of the input requests at the crossbar and can help in better switch allocation. Load balancing the input requests in combination with dimension information in the VIX architecture will help improve performance in adversarial traffic patterns.

2.4 Implementation

Supporting VIX requires modest changes to the router architecture. The datapath of the router needs to support extra connections between input buffers and crossbar. The extra connections can be realized by increasing the number of output multiplexors which route flits from virtual channels to the crossbars. The extra connections for a router with *two* virtual inputs per input physical port, are shown in Figure 2.

The switch allocation unit also needs modification to support VIX. Figure 3 (a) shows the organization of switch allocation unit with input-first separable allocation for baseline router without VIX, with v virtual channels, and P physical input and output ports. The input arbitration is done in parallel by P input arbiters each of size $v : 1$. The output arbitration is done in parallel by P output arbiters, each of size $P : 1$. Figure 3 (b) shows the organization of switch allocation

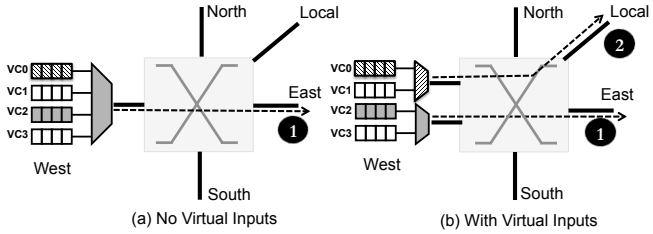


Figure 4: Virtual input crossbars allow more than one flit to be transferred from each input port per cycle.

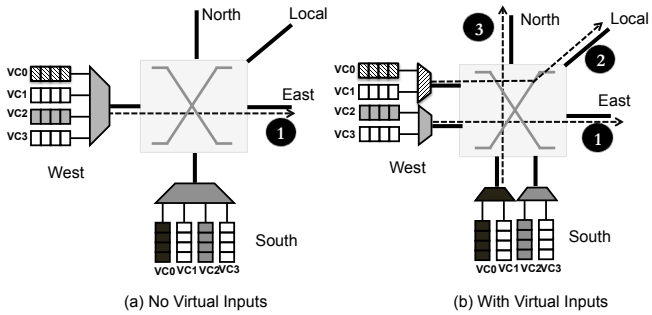
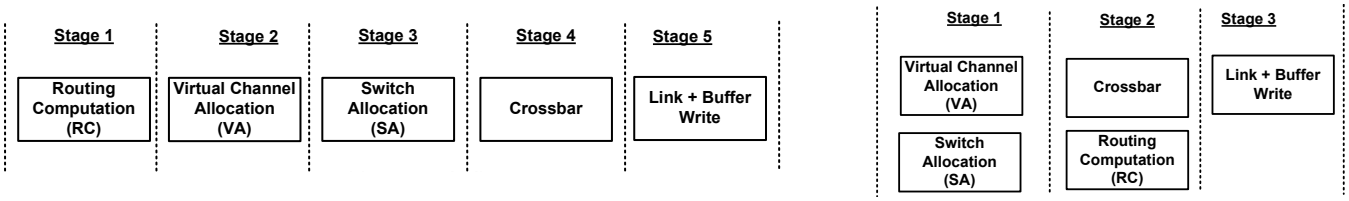


Figure 5: Virtual input crossbars expose more conflict free input requests for output arbitration.



(a) 5-stage pipeline (b) 3-stage pipeline

Figure 6: Router pipeline stages

unit with VIX. To realize two virtual inputs per physical input port, the number of input arbiters are doubled. However each input arbiter is now a $v/2:1$ arbiter and hence its complexity is reduced by half. The number of output arbiter remains same as baseline, but the complexity of each output arbiter is now doubled to $2P:1$.

The number of inputs of the crossbar are doubled to support a VIX with two virtual inputs per input physical port. The size of crossbar is increased from a $P \times P$ crossbar to a $2P \times P$ crossbar as shown in Figure 2. This can potentially effect the cycle time (and frequency) of the router. In order to examine the impact of using larger crossbars on router's cycle time, we synthesized the various components of router's pipeline. The different stages of a router's pipeline are shown in Figure 6. The Figure 6 (a) illustrates the stages of a 5-stage pipeline and Figure 6 (b) illustrates the stages of an optimize 3-stage pipeline with speculative switch allocation [19] and look ahead routing [9]. Several studies have shown that Virtual channel Allocation (VA) or Switch Allocation (SA) is on the critical path of the router [19, 17, 18, 10, 4]. Thus we examine the delays of VA stage, SA stage, and crossbars to study the impact of using larger crossbars for supporting VIX.

We synthesize the VA and SA stage using open-source NoC router RTL code [4, 3]. Synthesis was done with Synopsis Design Compiler using a commercial 45nm SOI technology library (1.0V, 25°C and topographical mode). Since crossbars are wire dominated structures, we use SPICE modeling for more accurate estimation of crossbar delay. We model 128 bit matrix crossbars whose inputs are driven by a flip-flop followed by a driving buffer. Each output gets latched into a register. Tri-state buffers connect the input wires to the output wires, with a one-hot enable signal. The enable signal is assumed to be driven from the SA pipeline stage. Metal 3 and Metal 4 are used to model the crossbar wires. Appropriate length wire models are used, assuming 2x spacing between wires, to avoid coupling. The driving buffer and tri-state gates are optimally sized to reduce delay.

Design	Radix	Xbar size	VA Delay	SA Delay	Xbar Delay
Mesh	5	5 x 5	300 ps	280 ps	167 ps
Mesh with VIX	5	10 x 5	300 ps	290 ps	205 ps
CMesh	8	8 x 8	340 ps	315 ps	205 ps
CMesh with VIX	8	16 x 8	340 ps	330 ps	289 ps
FBfly	10	10 x 10	360 ps	340 ps	238 ps
FBfly with VIX	10	20 x 10	360 ps	345 ps	359 ps

Table 1: Router pipeline stage delays.

The delays for analyzed components of the router are shown in Table 1. To study the wider applicability of VIX, we synthesized the components for routers of three different topologies. The topologies considered are mesh [10, 22], concentrated mesh [2] and flattened butterfly [11]. Each topology requires a router with different radix[1], radix 5 for mesh, radix 8 for concentrated mesh (CMesh), and radix 10 for flattened butterfly (FBfly). The different topologies also require crossbars of different sizes as noted in Table 1. We observe that crossbar stage is not in the critical path of router's pipeline. Our analysis demonstrates that there is sufficient slack in crossbar stage to support larger crossbars required for VIX architecture. For a mesh router, the delay of crossbar stage increases by 22%, while still remaining within 70% of the router's cycle time. The slack in crossbar stage reduces for higher radix flattened butterfly topology, however the allocation stage delays also increase proportionally for higher radix. Supporting VIX for flattened butterfly increases the delay of crossbar stage by 50%. However the crossbar delay continues to be lower than VA delay, thus VIX can be implemented for flattened butterfly without increasing the cycle time of router. But this also indicates that VIX architecture may not scale to very high radices unless innovative high-radix switch architectures are utilized [20].

In summary, we conclude that we can implement VIX architecture without degrading the frequency of baseline router architecture for the three topologies considered.

3. METHODOLOGY

We use a cycle-accurate network-on-chip simulator for our analysis. All routers use the three stage pipeline shown in Figure 6 (b). We model deterministic dimension order routing algorithm, finite input buffering, wormhole switching, and virtual-channel flow control. For each router, we assume a buffering of 6 virtual channels per port and a buffer depth of 5 flits per virtual channel. The datapath width of the router is constant across all topologies and is equal to 128 bits.

We evaluate the different network configurations with uniform random statistical traffic and application benchmarks. For applications, we use a trace-driven, cycle-accurate manycore simulator with the above network model integrated with core, cache and memory controller models. Table 2 provides the configuration details of the components simulated.

We use a set of multiprogrammed application workloads comprising scientific, commercial, and desktop applications. We study 35 benchmarks, including SPEC CPU2006 benchmarks, and four commercial workloads traces (`sap`, `tpcw`,`sjbb`, `sjas`). The details of how each multiprogrammed workload mix is derived from the different benchmarks are discussed in Section 4.7.

[1]Radix is equal to number of physical input/output ports of a router.

For network energy, each component of router such as, links, buffers and switch is modelled through SPICE. Our models include energy spent due to clocking and leakage energy. The activity factor of links, buffers and switches were collected from cycle-accurate simulations and integrated with component models to determine the overall network energy consumption.

Cores	64 cores , 2-way out-of-order, 2 GHz frequency
L1 Caches	32 KB per-core, private, 4-way set associative, 64B blocks, 2-cycle latency, split I/D caches, 32 MSHRs
L2 Caches	64 banks, 256KB per bank, shared, 16-way set associative, 64B block size, 6-cycle latency, 32 MSHRs
Main Memory	8 on-chip memory controllers, 4 DDR channels each @16GB/s, up to 16 outstanding requests per core, 80ns access latency

Table 2: Processor configuration

4. EVALUATION

4.1 Network Configurations Studied

We evaluate the network performance for four different switch allocation schemes - **Separable input-first (IF)**, **Wavefront (WF)**, **Augmented Path (AP)** and **VIX**. In the IF scheme, the input arbiters choose one request per input port. Next, the output arbiters choose one input request for each output port. This can lead to uncoordinated decisions between the two arbiters. The inefficiency of separable allocators can be minimized by more complex iterative allocators [14] or wavefront (**WF**) [21]. The WF allocator works on the principle of iteratively allocating all conflict free input-output pairs along the diagonals. By exposing more conflict free inputs to the outputs, WF allocation achieves better allocation efficiency than separable allocators. Better allocation comes at a cost of increased complexity and higher cycle time [4, 15]. The WF allocator has 39% higher cycle time than a separable allocator as shown in Table 3. However, the WF allocator does not necessarily achieve maximum or ideal matching. Maximum matching can achieved by the **AP** algorithm [8]. The complexity of **AP** algorithms limit their applicability to NoC routers [4]. In the experiments conducted, all VIX configurations have two virtual inputs per input port. All our experiments are performed for 64 node networks. For each of these configurations, we study the average packet latency and average network throughput for each of the four switch allocation techniques. To study the allocation techniques independent of their implementation limitations, we assume equal cycle time for all switch allocation techniques. The packet latency is reported in $cycles$ and network throughput in $packets/cycle/node$ or $flits/cycle$. The routers have a 128 bit datapath width, and support 6 virtual channels per input port. We evaluate the different network configurations with uniform random statistical traffic with a packet size of 512 bits (i.e. 4 flits).

	Separable	Wavefront	Augmented Path
Delay	280 ps	390 ps	Infeasible

Table 3: Delay of different switch allocation schemes.

4.2 Switch Allocation Efficiency

In this section, we study the switch allocation efficiency for a *single* router. This study allows us to analyze the different switch allocation techniques in isolation, without second order effects of network topology and interaction of a router with other routers. Packets are injected at maximum injection rate into each port of the router. Figure 7 illustrates the throughput achieved by the router for different radices and different allocation techniques. A single router of Radix-5 can achieve at best 5 flits/cycle and Radix-10 can achieve at best 10 flits/cycle. We observe that the trends across different radix routers are similar. The AP algorithm can provide above 30% higher throughput than separable IF for all radix configurations, while VIX provides above 25% throughput improvement over IF for all radices evaluated. Both AP and VIX achieve efficiency very close to ideal switch allocation. Ideal allocation is possible by allowing 6 virtual inputs per input port. Interestingly, some of these trends are reversed for network level performance, as we shall see in the next section.

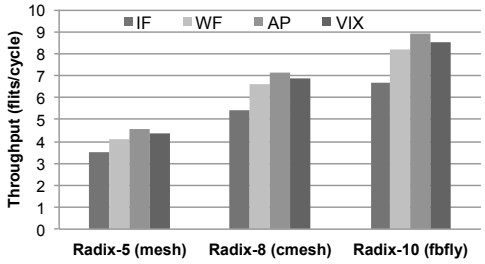


Figure 7: Switch allocation efficiency for single router.

4.3 Network-Level Performance

In this section, we evaluate the network performance for a mesh configuration using a 64 node network.

Figure 8 shows the average packet latency and average network throughput for mesh topology. We observe that VIX only provides benefits at high injection rates. Since, at low network load all the allocation schemes have nearly identical performance due to fewer output port conflicts. At higher injection rates, we observe that VIX improves network throughput by 16.2% and average packet latency by 36% over the IF allocation method. VIX also outperforms the AP scheme. VIX improves network throughput by 15.9% over the AP allocation.

The AP scheme improves the network throughput by a mere 0.3% over IF for mesh topology. This is because the AP algorithm follows a greedy approach, making optimal decisions locally while making sub-optimal decisions at the network level, leading to high levels of unfairness in the network. The extent of unfairness in the network is proportional to the average number of hops in the network. To illustrate this fact, we evaluated the maximum/minimum throughput values of all the nodes in the network. Ideally, the maximum/minimum throughput should be close to 1, as all network nodes are injecting at equal injection rate. Figure 9 shows that this ratio is 6.4 for Mesh using an AP algorithm, while it improves to 1.99 by using VIX.

Overall at the network level, VIX achieves higher network throughput and also has significantly lower average packet latency at high load. In addition, VIX also achieves maximum fairness at the network level when compared to the other allocation schemes studied.

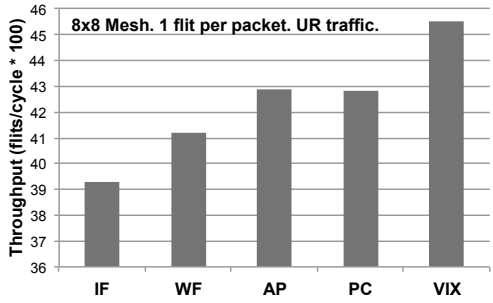


Figure 10: Network throughput achieved by Packet Chaining, VIX and other allocation techniques.

4.4 Comparison to Packet Chaining

Recent work proposes the concept of *Packet Chaining* [15] for improving the efficiency of separable allocators. The key idea is to inherit the allocation decisions made by the separable allocator in previous cycles and then decide the new allocations for current cycle. Thus, packets which want to communicate between same output-input pairs are chained. Packet chaining provides higher benefits for short packets or single-flit packets. By using past history and spreading the allocation over multiple cycles, Packet Chaining (PC) achieves some of the properties of iterative allocators.

In our view, PC works by elimination. By preserving the allocations from previous cycles, it eliminates many requests from the request matrix, thereby reducing the chances of uncoordinated decisions between the input and output arbiters. Input ports which preserve connections

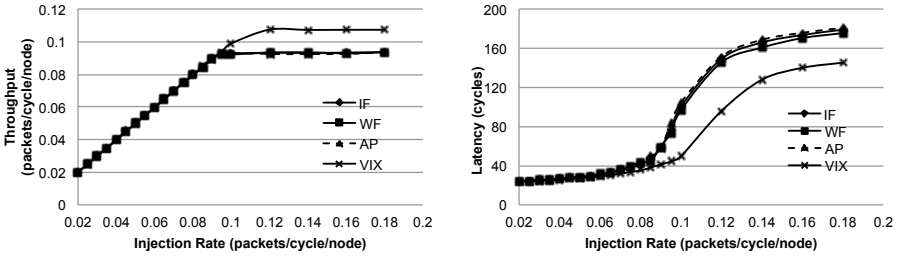


Figure 8: Network throughput (a) and packet latency (b) for a mesh topology.

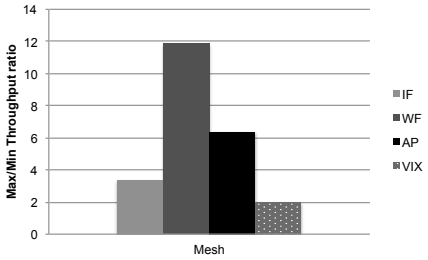


Figure 9: Fairness for a mesh topology.

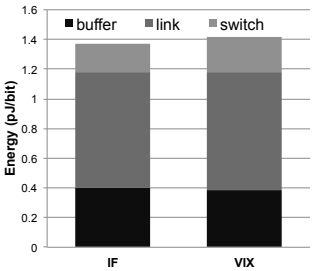


Figure 11: Network energy per bit.

from the previous cycle do not participate in input arbitration. Thus the chances that many input arbiters pick the same output port is reduced. On the other hand, VIX works by exposing more non-conflicting requests to output arbiters, and thereby reducing the chances of uncoordinated decisions between input and output arbiters. Both allocation methods solve the problem with baseline separable allocators by taking very different approaches. In addition, VIX has the advantage of allowing more than one virtual channel per input port to transmit flits in a given cycle.

To compare the two techniques quantitatively, we replicate the router pipeline used by Michelogiannakis et al. for Packet Chaining [15] and simulate the $SameInput, anyVC$ scheme. We evaluate a 8x8 mesh with uniform random traffic, and single-flit packets. Figure 10 shows the network throughput achieved by the different allocation methods at maximum injection rate. We observe that PC improves network throughout by 9%, whereas VIX improves network throughput by 16%. We conclude that for separable allocators, an approach which exposes more non-conflicting input requests to output arbiters provides better allocation than an approach which eliminates input requests and preserves prior connections.

4.5 Energy Consumption

Figure 11 shows the energy per bit for the mesh topology at an injection rate of 0.1 packets/cycle/node. In VIX, the switch energy increases due to an increase in the crossbar size. We observe that the total network energy per bit increses by 4% for a Mesh network due to a larger crossbar size. Note, VIX improves both network-level throughput and application-level throughput significantly. Tasks will complete execution earlier, leaving the network and processor's other components idle for longer periods. Hence, leakage energy of network and processor's others components can be saved. Also, energy dissipated by clocking can be saved during the idle periods.

4.6 Increasing Virtual Inputs

In this section we study the impact of increasing virtual inputs. Figure 12 shows network throughput for three configurations. First, baseline which has no virtual inputs (no VIX). Second, VIX with two virtual inputs per input port (1:2 VIX) and lastly, ideal VIX configuration which has equal number of virtual inputs and virtual channels, per input port. We study the impact of VIX for routers with both 4 VCs and 6 VCs per input port. The realistic 1:2 VIX provides significant throughput improvement for both 4 VCs (21% on average) and 6 VCs (16% on average). For routers with 4 VCs, we find that VIX with two virtual inputs (1:2 VIX) achieves nearly equal performance as ideal VIX for all topologies. Thus, two virtual inputs suffice for routers with 4 VCs per input port. For 6 VCs, two virtual inputs is close to ideal VIX for mesh and concentrated mesh topologies. The switch allocator has more opportunities for improving throughput in flattened butterfly because of its higher radix. Thus the gap between 1:2 VIX and ideal VIX is more pronounced for flattened butterfly with 6 VCs.

We also observe that 1:2 VIX with 4 VCs achieves more than 10% throughput improvement over a 6VC configuration without VIX. Thus VIX can be leveraged to reduce number of buffers, and save router area/power. VIX can be used to reduce network buffers by 33% (from 6 VCs to 4 VCs) while still improving the network throughput by 10%.

4.7 Application-Level Performance

In this section, we analyze the effect of different allocation schemes with real application workloads. We evaluate the multi-programmed workloads Mix1 - Mix8 for our experimental analysis. Each workload consists of 6 unique applications. These applications are chosen at random from a suite of 35 benchmarks. Table 4 lists the multi-programmed workloads used for the study and the speedup observed over the baseline scheme. We observe an average increase of 5% and a maximum of 7% in system performance using VIX over the base (IF) allocation scheme. VIX also shows an increase of upto 3.2% in performance over the AP algorithm.

5. RELATED WORK

Several interesting switch allocation schemes have been proposed for off-chip networks [1, 16]. We focus on switch allocation techniques which have been proposed for on-chip networks. We have already compared our approach extensively, both qualitatively and quantitatively, to existing switch allocation techniques such as separable allocators, wavefront allocators [21], almost ideal matching based on augmented path algorithms [8] and packet chaining [15].

Kumar et al. [13] proposed SPAROFLO switch allocation to improve efficiency of separable allocators. In SPAROFLO, the number of input requests presented to output arbiters is varied dynamically with network load. Similar to VIX, more than one request per input port is selected for output arbitration at low and medium load. Since there are no virtual inputs at the crossbar, only one request per input port can be granted. Thus, conflicts must be detected after output arbitration. This is done by assigning a priority to the virtual channels during input arbitration. These conflicts limit the efficiency of SPAROFLO when compared to VIX. In addition to varying dynamically the number of requests per input port, SPAROFLO prioritizes older requests during output arbitration. Priority is also given to flits of same packet, so that all flits of a packet stay together. These prioritization optimizations can be easily integrated with VIX.

Becker et al. [4] study the design space of virtual channel allocators and switch allocators for NoCs. They also propose an optimization for speculative switch allocation. In speculative switch allocation, the winners from the non-speculative requests must be prioritized over speculative requests. By pessimistically masking the speculative requests, their design removes the prioritization circuit from the critical path of switch allocation. This optimization is orthogonal to our proposed techniques and is applicable to routers with VIX.

Recently Chang et al. [5] proposed TS-router. By predicting future requests to arrive at a router, TS-router maximizes the matching of separable allocators across cycles. While this is an beneficial approach, it does not solve the problem uncoordinated decisions between the input and output arbitration stages. We believe both TS-router and

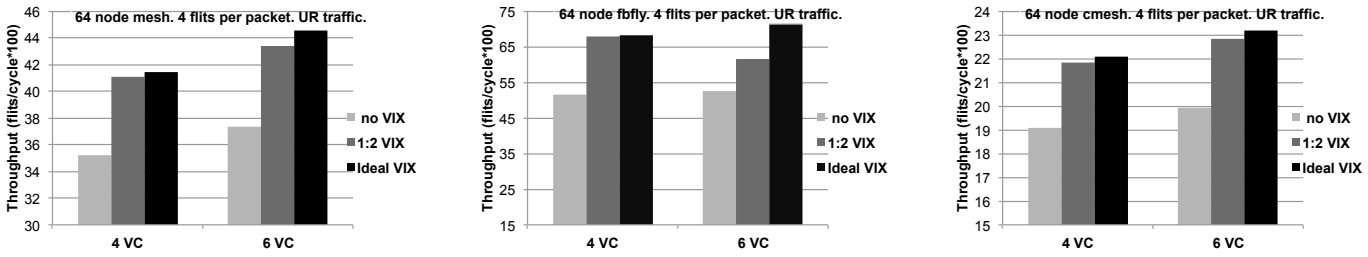


Figure 12: Impact of increasing the number of Virtual Inputs for (a) Mesh, (b) Flattened butterfly, and (c) Concentrated mesh.

Mix							avg. MPKI	Speedup
Mix1	milc (11)	applu (11)	astar (10)	sjeng (11)	tonto (11)	hmmer (10)	15.0	1.03
Mix2	sjas (11)	gcc (11)	sjbb (11)	gromacs (11)	sjeng (10)	xalan (10)	21.3	1.03
Mix3	milc (11)	libquantum (10)	astar (11)	barnes (11)	tpcw (11)	povray (10)	33.3	1.04
Mix4	astar (11)	swim (11)	leslie (10)	omnet (10)	sjas (11)	art (11)	38.4	1.05
Mix5	applu (11)	lbm (11)	Gems (11)	barnes (10)	xalan (11)	leslie (10)	42.5	1.05
Mix6	mcf (11)	ocean (10)	gromacs (10)	lbm (11)	deal (11)	sap (11)	52.2	1.05
Mix7	mcf (10)	namd (11)	hmmer (11)	tpcw (11)	omnet (10)	swim (11)	58.4	1.06
Mix8	Gems (10)	sjbb (11)	sjas (11)	mcf (10)	xalan (11)	sap (10)	66.9	1.07

Table 4: Benchmarks used to construct eight multiprogrammed workloads for our 64-core processor. Numbers in parenthesis indicate the number of application instances that we used to construct a workload. The average Misses-Per-Kilo-Instruction (MPKI) per core for a benchmark is calculated as the sum of its L1-MPKI and L2-MPKI. The last column indicates the speedup of VIX over Baseline

VIX can work together to improve efficiency of separable allocators.

Prior work has proposed input speedup - a mechanism by which output port utilization can be improved by increasing the number of input ports to the crossbar [6]. The mechanism that we propose provides a strong case for this architecture for NoCs. We expose requests coming from different VCs to the crossbar by increasing the number of input ports. VIX can also benefit from different VC allocation schemes to ensure better utilization of input virtual channels. Our studies showed that VIX can achieve a throughput higher than complex allocation schemes in most topologies studied. Moreover, the synthesis and SPICE simulations show that this can be achieved with minimal impact on delay and energy efficiency.

Kim et al. proposed a modular router architecture (RoCo) in [12] to achieve low latency and energy efficient networks. They propose decoupled parallel arbiters and use smaller crossbars for input and output port connections to improve the allocation efficiency for mesh topology with dimension-order routing. The RoCo architecture has two sets of smaller 2×2 crossbars which are used depending on the direction of flit traversal, i.e East-West or North-South. It tries to minimize the conflicts in allocation by dividing requests based on dimension. VIX and RoCo have the same goal, but VIX exposes more requests to the crossbar by increasing its size and having virtual inputs. VIX can also assign input requests to virtual channels based on the dimension of travel, thus reducing output port conflicts.

6. CONCLUSION

Switch allocation matches output links to waiting input requests. Switch allocation directly impacts network throughput. An ideal switch allocator avoids idle cycles on output links for which requests exist. For on-chip networks, separable allocators provide best cycle time vs efficiency trade-off. However, uncoordinated decisions between input arbitration and output arbitration stages of separable allocators can lead to poor allocation efficiency. We make the observation that providing virtual inputs to the crossbar, which connect more than one virtual channel per input port to the crossbar, can improve the efficiency of separable allocators significantly. We show that VIX is feasible for on-chip routers without increasing routers cycle time.

Our evaluations show that VIX based allocation improves network throughput by 16% for a mesh topology. We also study the applicability of VIX to higher radix topologies. VIX improves network throughput by 15% for concentrated mesh topology and by 17% for flattened butterfly topology. Higher performance from VIX can be used to reduce network buffers by 33% while still improving the network throughput by 10%.

7. REFERENCES

[1] T. E. Anderson, S. S. Owicki, J. B. Saxe, and C. P. Thacker. High-speed switch scheduling for local-area networks. *ACM Transactions on Computer Systems (TOCS)*, 1993.

[2] J. D. Balfour and W. J. Dally. Design tradeoffs for tiled cmp on-chip networks. In *ICS*, 2006.

[3] D. U. Becker. Efficient microarchitecture for network-on-chip routers. *PhD thesis, Stanford University*, 2012.

[4] D. U. Becker and W. J. Dally. Allocator implementations for network-on-chip routers. In *SC*, 2009.

[5] Y. Chang, Y. S.-C. Huang, M. Poremba, V. Narayanan, Y. Xie, and C.-T. Kin. Ts-router: On maximizing the quality-of-allocation in the on-chip network. In *HPCA-19*, 2013.

[6] W. Dally and B. Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[7] W. J. Dally and B. Towles. Route packets, not wires: On-chip inteconnection networks. In *DAC-38*, 2001.

[8] L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 1956.

[9] M. Galles. Scalable pipelined interconnect for distributed endpoint routing: the sgi spider chip. In *Symposium on High Performance Interconnects (Hot Interconnects)*, 1996.

[10] J. Howard, S. Dighe, Y. Hoskote, S. R. Vangal, D. Finan, G. Ruhl, D. Jenkins, H. Wilson, N. Borkar, G. Schrom, F. Pailet, S. Jain, T. Jacob, S. Yada, S. Marella, P. Salihundam, V. Erraguntla, M. Konow, M. Riepen, G. Droege, J. Lindemann, M. Gries, T. Apel, K. Henriss, T. Lund-Larsen, S. Steibl, S. Borkar, V. De, R. F. V. der Wijngaart, and T. G. Mattson. A 48-core ia-32 message-passing processor with dvfs in 45nm cmos. In *ISSCC*, 2010.

[11] J. Kim, J. Balfour, and W. Dally. Flattened butterfly topology for on-chip networks. *MICRO-40*, 2007.

[12] J. Kim, C. Nicopoulos, D. Park, V. Narayanan, M. S. Yousif, and C. R. Das. A gracefully degrading and energy-efficient modular router architecture for on-chip networks. In *Proceedings of the 33rd Annual International Symposium on Computer Architecture*, ISCA '06, pages 4–15, Washington, DC, USA, 2006. IEEE Computer Society.

[13] A. Kumary, P. Kundu, A. Singh, L.-S. Peh, and N. Jha. A 4.6 tbits/s 3.6 ghz single-cycle noc router with a novel switch allocator in 65nm cmos. In *ICCD-25*, 2007.

[14] N. McKeown. The islip scheduling algorithm for input-queued switches. *Networking, IEEE/ACM Transactions on*, 1999.

[15] G. Michelogiannakis, N. Jiang, D. U. Becker, and W. J. Dally. Packet chaining: efficient single-cycle allocation for on-chip networks. In *MICRO-44*, 2011.

[16] S. S. Mukherjee, F. Silla, P. Bannon, J. Emer, S. Lang, and D. Webb. A comparative study of arbitration algorithms for the alpha 21364 pipelined router. *ACM SIGARCH Computer Architecture News*, 2002.

[17] R. Mullins, A. West, and S. Moore. Low-latency virtual-channel routers for on-chip networks. In *ISCA '04: Proceedings of the 31st Annual International Symposium on Computer Architecture*, 2004.

[18] C. A. Nicopoulos, D. Park, J. Kim, N. Vijaykrishnan, M. S. Yousif, and C. R. Das. Vichar: A dynamic virtual channel regulator for network-on-chip routers. In *MICRO-39*, 2006.

[19] L.-S. Peh and W. J. Dally. A delay model and speculative architecture for pipelined routers. In *HPCA-7*, 2001.

[20] S. Satpathy, K. Sewell, T. Manville, Y.-P. Chen, R. G. Dreslinski, D. Sylvester, T. N. Mudge, and D. Blaauw. A 4.5tb/s 3.4tb/s/w 64x64 switch fabric with self-updating least recently granted priority and quality of service arbitration in 45nm cmos. In *ISSCC*, 2012.

[21] Y. Tamir and H.-C. Chi. Symmetric crossbar arbiters for vlsi communication switches. *Parallel and Distributed Systems, IEEE Transactions on*, 1993.

[22] D. Wentzlaff, P. Griffin, H. Hoffmann, L. Bao, B. Edwards, C. Ramey, M. Mattina, C.-C. Miao, J. F. B. III, and A. Agarwal. On-chip interconnection architecture of the tile processor. *IEEE Micro*, 2007.