Virtualizing Virtual Channels for Increased Network-on-Chip Robustness and Upgradeability

Marios Evripidou, Chrysostomos Nicopoulos
Department of ECE,
University of Cyprus
Email: evripidou.marios@ucy.ac.cy
nicopoulos@ucy.ac.cy

Vassos Soteriou
Department of EECEI,
Cyprus University of Technology
Email: vassos.soteriou@cut.ac.cy

Jongman Kim
School of ECE,
Georgia Inst. of Technology, USA
Email: jkim@gatech.edu

***Abstract*—The Network-on-Chip (NoC) router buffers are instrumental in the overall operation of Chip Multi-Processors (CMP), because they facilitate the creation of Virtual Channels (VC). Both the NoC routing algorithm and the CMP's cache coherence protocol rely on the presence of VCs within the NoC for correct functionality. In this article, we introduce a novel concept that completely decouples the number of supported VCs from the number of VC buffers physically present in the design. *Virtual Channel Renaming* enables the virtualization of existing virtual channels, in order to support an arbitrarily large number of VCs. Hence, the CMP can (a) withstand the presence of faulty VCs, and (b) accommodate routing algorithms and/or coherence protocols with disparate VC requirements. The proposed *VC Renamer* architecture incurs minimal hardware overhead to existing NoC designs and is shown to exhibit excellent performance without affecting the router's critical path.**

I. Introduction

The advent of Chip Multi-Processors (CMP) has accentuated the criticality of the on-chip communication infrastructure, which is now tasked with the mission-critical objective of maintaining swift and reliable inter-core communication. Escalating numbers of on-chip processing cores necessitate the introduction of an efficient and scalable communication backbone. Packet-based Networks-on-Chip (NoC) [1] are envisioned as the most viable solution for the many-core chips of the near future.

Among the various components comprising the on-chip network backbone, the router buffers constitute one of the fundamental cogs in the operation of the NoC. The buffering resources orchestrate the flow-control mechanism of the network and facilitate the so called Virtual Channels (VC), which enable the multiplexing of several packets onto a single physical channel. More importantly, however, VCs are obligatory constructs for the correct functionality of two elemental operations within the CMP:

(a) Network Routing Computation: The vast majority of adaptive routing algorithms developed for interconnection networks rely on the extensive use of VCs for deadlock avoidance and/or deadlock recovery. Existing adaptive routing algorithms require anywhere from 2 to 16 VCs per router input port for functional correctness [2]. While deterministic routing algorithms (e.g., XY routing) may require only one VC, adaptive algorithms offer much more flexibility, adaptability to prevailing traffic conditions, and resilience to faults.

(b) Cache Coherence: The plethora of cache coherence protocols designed for modern CMPs necessitate the use of multiple virtual networks (realized through VCs), in order to avoid protocol deadlocks. For instance, the well-known MOESI protocol requires 3 virtual networks (i.e., at least 3 VCs), while other protocols require anywhere from 2 to 8 VCs per router input port [3].

Hence, virtual channels are key enablers for the seamless functionality of both the network fabric and the cache/memory sub-system of multi-core microprocessors.

Most existing NoC designs employ statically partitioned VC resources. In other words, the router buffer space is partitioned a priori at design-time (i.e., pre-manufacturing) and the resulting NoC has a fixed number of VCs in each router port. This characteristic raises two critical concerns stemming from its innate inflexibility:

(1) System functionality in the event of a VC buffer/channel malfunction – The issue of hardware reliability is becoming increasingly relevant as technology scales deep into the nano-scale regime [4], [5]. Defects in various locations within the NoC may affect the functionality of VCs directly (e.g., buffer faults), or indirectly (e.g., arbiter faults). The NoC VCs are so deeply intertwined with the operation of the CMP, that a malfunction in any of the VC components may lead to network and/or cache coherence protocol deadlocks (i.e., whole-system inoperability).

(2) System upgradeability with new routing algorithms, and/or new cache coherence protocols, which require different numbers of VCs – A major limitation with a statically partitioned VC implementation is the inability to accommodate routing algorithms and cache coherence protocols that require a different number of VCs than the specific (and fixed) number of VCs ingrained into the CMP at design-time. The concept of upgradeability is not merely academic. The incessant emergence of newer and more demanding applications may require the introduction of newer (and/or customized) routing algorithms, in order to optimize system performance. Such capability is currently limited by the rigid requirement of strict compliance with the existing number of VCs in the NoC.

This pair of problematic facets in conventional VC implementations serves as the primary driver of the work presented in this article: Re-engineering the VC operation so as to achieve both robustness and the ability to support multiple routing algorithms and cache coherence protocols with no restrictions on the number of supported VCs. Toward this end, we hereby introduce the concept of **Virtual Channel Renaming (VC Renaming)**. The key idea is to enable the further virtualization of virtual channels. Through this process, the system would allow the mapping of any number of Virtual Virtual Channels (VVC) on top of the existing (i.e., statically partitioned at design-time) virtual channel buffers (henceforth called Physical Virtual Channels, or PVC). In essence, the technique of VC Renaming facilitates the creation of an arbitrary number of VVCs (where the Number of VVCs $\geq$ Number of PVCs), irrespective of the original fixed number of PVCs built into the NoC at manufacturing.

The proposed VC Renaming technique decouples the number of VCs (now known as VVCs) required by the routing algorithm and/or the cache coherence protocol from the actual number of physical VCs (PVCs) originally built into the system at design-time. The principle of VC Renaming is inspired by the well-known architectural technique of Register Renaming, which overcomes the restriction of a fixed number of architectural registers within the CPU's Instruction Set Architecture (ISA).

The notion of ***virtualizing virtual channels*** is an innovative idea, which, to the best of our knowledge, has not been suggested in the literature before. Our simulation results are very promising, and hardware synthesis using commercial 65 nm TSMC libraries demonstrates modest area/power overhead and no impact on the router's critical path.

The rest of the paper is organized as follows: A summary

of related work follows in Section II, while the proposed architecture is introduced in Section III. Section IV presents and analyzes various simulation results. Finally, Section V concludes the paper.

II. Related Work

There has been extensive research in architecting *reconfigurable NoCs*. However, most of this work has concentrated on the re-configuration of the router input ports, the physical links, and the network topology [6]. Pertaining to the buffers, reconfigurable NoCs mainly re-allocate buffer space to the various ports, but do not provide any flexibility in the structure of the VCs [7]. The work presented in this paper can be integrated into existing reconfigurable NoC designs, in order to provide this extra flexibility.

More relevant to our work, *dynamic NoC buffer managers* aim to break the rigidity of static VC partitioning through run-time reconfiguration. Various dynamic VC management proposals unify the buffer resources into a common pool, which is then managed centrally and allocated to various VCs based on prevailing conditions [8], [9], [10], [11], [12], [13]. Some designs can also vary the number of VCs dynamically at run-time [12], [13]. These techniques, however, require a complete redesign of the input port architecture (i.e., they are disruptive) and incur significant area/power overhead, as a result of the complexity involved in maintaining multi-ported unified buffer structures and/or larger crossbar switches [10], [12], [13]. In fact, most of these techniques were intended for off-chip communication [8], [9], [10], [11], i.e., not as severely resource-constrained as on-chip designs. Furthermore, these approaches are always-on: All packets are forced to go through the unified buffer. On the contrary, the proposed VC Renamer mechanism only requires minimal augmentations to the existing NoC input port architectures. No unified buffers are required and modifications are only made to the control logic of the conventional buffers. More importantly, the VC Renamer is disabled until it is actually needed, because the data path is not modified.

III. The VC Renamer Architecture

The VC Renamer mechanism is completely transparent to the neighboring NoC routers. The routers are only aware of the presence of a specific number of VVCs (those required by the routing algorithm and/or the coherence protocol). Note that PVC atomicity is now broken, i.e., each PVC may hold more than one VVC. Atomicity within the individual VVCs, however, is still maintained.

The critical issues of starvation and protocol-level deadlocks arise whenever two or more VVCs are multiplexed over a single PVC. If one VVC dominates the available buffer space, other VVCs may not be left with any available slots, leading to protocol-level deadlocks (i.e., the higher-level protocol expects a particular message type that never arrives, due to the above-mentioned problematic situation within a PVC). The VC Renamer architecture ensures the absence of such pathologies through (a) intelligent use of the credits mechanism, which regulates flow between the PVCs, and (b) the assumption that the number of VVCs mapped to a specific PVC cannot exceed the number of buffer slots physically present in the PVC. The credits mechanism will be explained in Section III-A.

Two different VC Renamer implementations have been developed: A **Mask-Based (MB) implementation** and a **Linked-List-Based (LLB) implementation**. Both versions offer the same functionality, but each is geared toward a different objective. The MB approach is extremely lightweight and targets fault-tolerant designs, where the VC Renamer mechanism will only be used in input ports affected by faults. The LLB approach incurs a slightly higher hardware overhead and targets system upgradeability, whereby the VC Renamer technique will be used in all routers simultaneously.

It should be noted here that *both* implementations do *not* affect the VC Allocation (VA) or Switch Arbitration (SA) pipeline stages of the router. Since only one of the VVCs mapped to a particular PVC is active in any given clock cycle, the VA and SA arbiters are completely unaffected. This attribute is of paramount importance, since the VA and SA stages usually determine the router's critical path [14]. It will be demonstrated through hardware synthesis that the VC Renamer operates within the slack of the other pipeline stages and does *not* impact the router's critical path. Moreover, as the VA and SA arbiters are unaffected, any arbiter prioritization policies can still be used.

A. A Mask-Based (MB) Implementation

In a typical NoC router input port, each PVC is realized using a k-deep FIFO buffer, where k is the maximum number of flits (a packet comprises a number of fixed-size flits) in a PVC, as shown in Figure 1. Said figure presents a high-level overview of the MB architecture. FIFO order within each PVC is maintained by *head* and *tail* pointers. These pointers are k-deep, 1-wide, 1-hot circular registers. Assuming that the head of the buffer is on the right-hand side (see Figure 1), the pointers move *from right to left*. In the MB implementation of the VC Renamer, each PVC maintains its generic head and tail pointers. Two main additions are made: (1) a *VVC-to-PVC Mapping Table*, and (2) a k-bit *Mask* for each supported VVC, which indicates the occupied positions in the respective slots of the PVC.

The *arrival* of flits is regulated by the credits mechanism. Credits are only sent to the upstream router (i.e., the potential sender) if a preliminary *arrival test* is successful. Note that the proposed VC Renamer design assumes the use of *on/off credits*, i.e., stop/go signals (for each VVC) that regulate flow based on PVC buffer availability. Credits are sent to upstream routers for each VVC in each input port. The credits signals for all VVCs mapped to a specific PVC are determined by the slot availability in said PVC; i.e., even though credits are distributed at the VVC-level, they are determined by buffer availability at the PVC-level. To ensure that the VVCs mapped to a particular PVC do not receive credits simultaneously, only credits for one VVC (per PVC) are dispatched in any given clock cycle. Without loss of generality, the credits for each VVC *mapped to a single PVC* are sent out in round-robin fashion (one in each cycle). As will be shown later on, this round-robin dispatch of credits has a minimal impact on performance. If needed, the round-robin policy can be replaced by any other policy, in order to implement different VVC prioritization schemes.

The *arrival test* algorithm – which is responsible for the credits mechanism – is shown in Algorithm 1. Step 1 of the algorithm ensures the absence of starvation and protocol-level deadlocks. Specifically, the two conditions of Step 1 guarantee that *each VVC mapped to a specific PVC will always have access to at least one buffer slot*. Remember that the number of VVCs mapped to a specific PVC cannot exceed the number of buffer slots physically present in the PVC. An "Empty" VVC is one whose Mask contains all zeros. The conditions of Step 1 are independent and can be fully parallelized in hardware. In addition to this check (Step 1), a flit is only allowed to arrive if the position pointed to by the PVC tail pointer is to the left of the left-most '1' in the corresponding VVC mask (remember, flits are assumed to fill in a VC buffer *from right to left*, as shown in Figure 1). This ensures that the flits of a particular VVC remain in order within the PVC buffer. This critical condition is checked through the *subtraction* of the VVC mask from the 1-hot PVC tail pointer (Step 2 of Algorithm 1) and observing the *sign* of the result (the *left-most bit* is assumed to be the *most significant bit*). Credits are sent to the upstream router if the PVC tail pointer is greater than the value of the VVC mask (positive subtraction result) *and* the PVC tail pointer does not point to an occupied position (Step 3). Since flit departures from the various VVCs may leave the PVC buffer fragmented, the PVC tail pointer may point to an occupied position. In such a case, no credits are sent out in the current cycle and the PVC

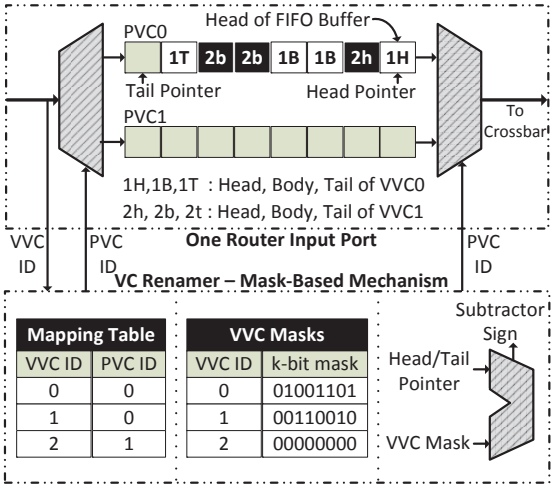


Figure 1. High-level overview of the Mask-Based (MB) implementation of the VC Renamer. Only one input port is shown for clarity. In this example, VVC0 and VVC1 are mapped to PVC0, while VVC2 is mapped to PVC1. Note that one subtractor *per PVC* is required.

tail pointer is shifted to the left. When a flit arrives at an input port, the VVC ID (contained within the flit) is used to index into the *VVC-to-PVC Mapping Table*, which uses the corresponding PVC ID to de-multiplex the incoming flit to the appropriate PVC. The existing PVC tail pointer is used to store the flit into the buffer. At the same time, the position of the PVC tail pointer denotes the respective bit position in the active VVC mask that must be set to '1'. An example of how the *arrival test* functions is illustrated in Figure 2.

Algorithm 1 Mask-Based Mechanism - Arrival Test

k = # of Empty VVCs mapped to current PVC
1: If {(PVC_Free_Slots $>$ k) OR (Current_VVC == Empty)} AND
2: (Tail Pointer $>$ VVC Mask Value) AND
3: (Tail Pointer points to Free Slot) THEN
4: => VVC_X Credits = ON

Algorithm 2 Mask-Based Mechanism - Departure Test

1: If (Downstream_Router_Credits = ON) AND
2: (Head Pointer points to Occupied Slot) AND
3: (Head Pointer $>$ VVC Mask Value) THEN
4: => Allow Flit to Depart

In a similar manner, a flit is only allowed to *depart* if the position pointed to by the PVC head pointer is the position of the right-most '1' in the corresponding VVC mask. This ensures that the flits of a particular VVC always depart the PVC buffer in the correct order. The *departure test* (as shown in Algorithm 2) is performed by doing a simple subtraction of the VVC mask from the 1-hot PVC head pointer (Step 3) and observing the *sign* of the result (the *right-most bit* is assumed to be the *most significant bit*). In the subtraction, the bit position in the VVC mask where the PVC head pointer points to is set to '0'. The flit is allowed to depart if the PVC head pointer does not point to an empty position (Step 2) *and* the PVC head pointer is greater than the value of the VVC mask (Step 3, positive subtraction result). If the PVC head pointer points to an empty position, no flit departs the buffer and the head pointer is shifted to the left. If the flit of one VVC cannot depart (e.g., no space in the downstream

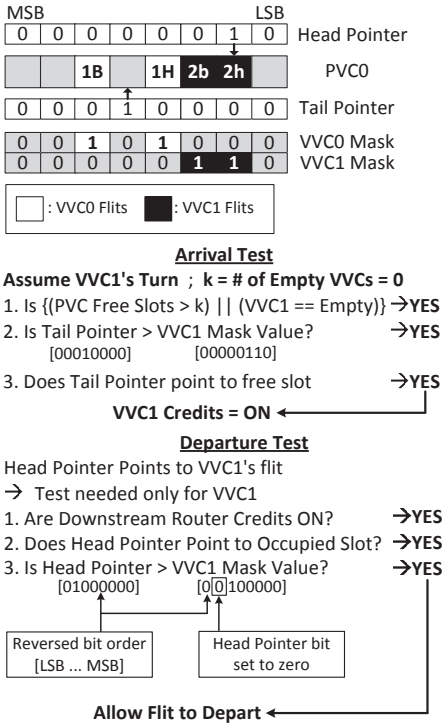


Figure 2. Step-by-step examples of the *arrival* and *departure tests* of the Mask-Based (MB) implementation of the VC Renamer. Only one PVC is shown for clarity with two VVCs (VVC0 and VVC1) mapped onto it. The various steps correspond to Algorithms 1 and 2.

router), the PVC head pointer moves to the next position in the following clock cycle, in order to allow other VVCs to proceed and avoid Head-of-Line (HoL) blocking. An example of how the *departure test* functions is illustrated in Figure 2.

The steps in both Algorithms 1 and 2 can be fully parallelized and overlapped in hardware, thus incurring minimal latency overhead (see Section IV). More importantly, they are off the router's critical path (which lies in the VA/SA stages).

B. A Linked-List-Based (LLB) Implementation

As will be demonstrated in Section IV, the mask-based approach incurs a performance penalty. Hence, the Linked-List-Based (LLB) implementation of VC Renamer targets higher performance at the expense of a slightly higher area/power overhead, as compared to the MB implementation. As in the MB approach, the modifications required to realize the LLB mechanism only affect the control logic of the existing PVC buffers. The new components comprise: (1) a *PVC Pointer List*, (2) a *Free-Slot FIFO List*, (3) a *Front-of-VC List*, (4) a *Back-of-VC List*, and (5) the same *VVC-to-PVC Mapping Table* of the MB implementation (see Section III-A). Figure 3 shows a high-level overview of the LLB architecture.

Assuming k-deep PVC buffers, the *PVC Pointer List* contains one k-deep, $\log_2 k$-bit wide vector per PVC. This vector holds the pointers to the next flit of each packet. The first flit of each VVC mapped to a specific PVC is located by accessing the *Front-of-VC List*, which is a list containing the location ($\log_2 k$ bits) of the next-departing flit of each VVC. Once the location of the next-departing flit is known, the *PVC Pointer List* points to the subsequent flits of the same packet in a linked-list manner. Similarly, the *Back-of-VC List* contains the location ($\log_2 k$ bits) of the last-stored flit of each VVC. It is used to extend the linked-list (i.e., add a new pointer) in the *PVC Pointer List* whenever a new flit of an in-flight packet arrives. The *Free-Slot FIFO List* maintains the free slots in each PVC. There is one such k-deep, $\log_2 k$-bit wide FIFO structure for each PVC in the input port. The *Free-Slot FIFO List* supplies the write locations for new incoming flits.

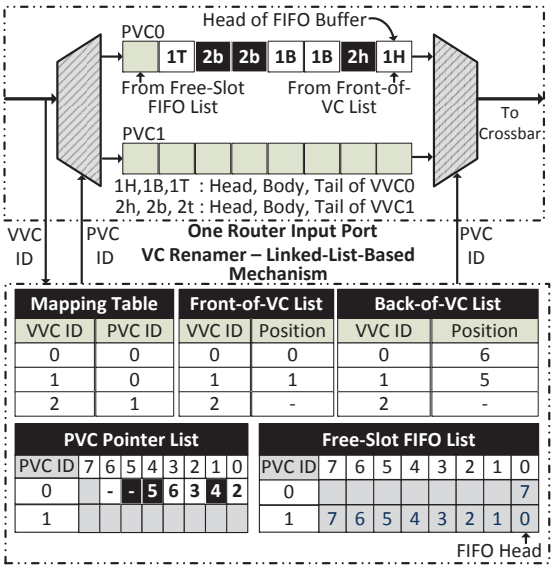


Figure 3. High-level overview of the Linked-List-Based (LLB) implementation of the VC Renamer. Only one input port is shown for clarity. In this example, VVC0 and VVC1 are mapped to PVC0, while VVC2 is mapped to PVC1.

The credits are sent to upstream routers in the same round-robin manner as in the MB approach (see Section III-A). Unlike the MB technique, the credits in the LLB implementation are regulated only by the two conditions shown in Step 1 of Algorithm 1, i.e.,

If {(PVC_Free_Slots $> k$) OR (Current_VVC == Empty)}
=> VVC_X Credits = ON,
where k = # of Empty VVCs mapped to current PVC.

As previously mentioned, this check ensures the absence of starvation and protocol-level deadlocks. In this case, the number of free slots is determined by the occupancy of the *Free-Slot FIFO List*, and the number of "Empty" VVCs is determined by the number of invalid entries in the *Front-of-VC List*.

When a new flit *arrives* at the input port (see Algorithm 3), the VVC ID within the flit is used to acquire the corresponding PVC ID from the *VVC-to-PVC Mapping Table*. The *Free-Slot FIFO List* is used to point the new flit to an available PVC slot (Steps 1 and 2). If the *Front-of-VC List* entry for the particular VVC is empty (i.e., start of new packet), it is updated with the location granted from the *Free-Slot FIFO List* (Step 3). At the same time, the corresponding *Back-of-VC List* entry is updated with the new location (Step 5). If the *Front-of-VC List* entry for the current VVC is *not* empty (i.e., the new flit is part of an existing packet), then the *Back-of-VC List* is used to index into the *PVC Pointer List* in order to extend the linked-list to the PVC location of the new flit (Step 4).

Algorithm 3 Linked-List-Based Mechanism - Flit Arrival

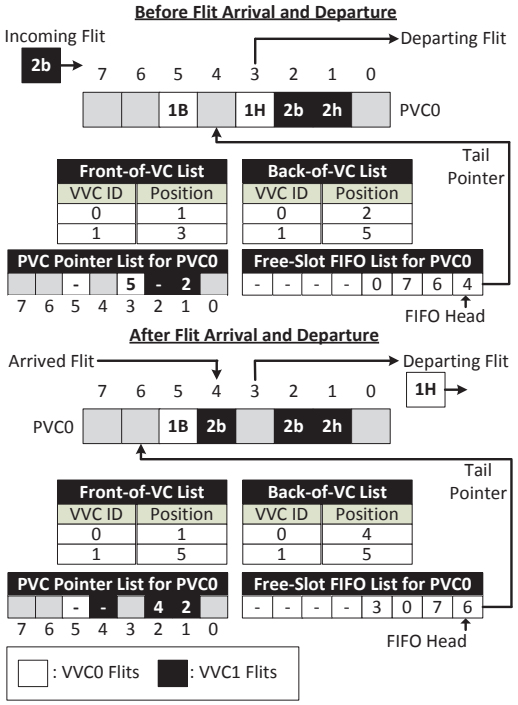
1: Get empty slot from *Free-Slot FIFO List*
2: Store flit in PVC
3: **If** (VVC = empty) update *Front-of-VC List* with flit position from *Free-Slot FIFO List*
4: **Else** use *Back-of-VC List* to index into the *PVC Pointer List* and extend the linked-list
5: Update *Back-of-VC List* with new flit position

Flit *departure* follows a similar process (see Algorithm 4). Once a flit is selected to depart, its VVC ID is used to acquire the next-departing flit location from the *Front-of-VC List* (Step 1). This value is also enqueued within the *Free-Slot FIFO List*, since the location will be vacated (Step 3). If the departing flit is a *tail* flit (i.e., the end of a packet), the corresponding entries in the *Front-of-VC List* and the *Back-of-VC List* are invalidated (Step 4, indicating an empty VVC). If the departing flit is *not* the last flit of its packet, then the *Front-of-VC List* is used to index into the *PVC Pointer List*. The indexed value within the *PVC Pointer List* points to the PVC location of the *next* flit of the same packet. This value is used to update the *Front-of-VC List*; i.e., the location of the *next* flit of the same packet now becomes the new head of the VVC (Step 5).

An example of how the *arrival* and *departure mechanisms*

Algorithm 4 Linked-List-Based Mechanism - Flit Departure

1: Acquire new Head Pointer from the *Front-of-VC List*
2: Allow flit to depart if credits are available
3: Push the head pointer value to the *Free-Slot FIFO List*
4: **If** (flit = tail flit) invalidate *Front-of-VC List* and *Back-of-VC List* entries
5: **Else** use the Head Pointer value to index into the *PVC Pointer List* and update *Front-of-VC List* with acquired value



Flit Arrival – Incoming flit belongs to VVC1
1: Acquire Tail Pointer from Free-Slot FIFO List → Position 4
2: Store Flit in Position 4
3. If (VVC != Empty) → Do Nothing
4: PVC Pointer List [Back-of-VC List[VVC1] = 2] = Position 4
5: Back-of-VC List[VVC1] = Position 4

Flit Departure – Assume VVC0's turn
1: Acquire Head Pointer from Front-of-VC List[VVC0] → Position 3
2: Remove flit from PVC Position 3
3: Push Position 3 to tail of Free-Slot FIFO List
4: If (Flit != Last flit) → Do Nothing
5: Front-of-VC List[VVC0] = PVC Pointer List [3] = Position 5

Figure 4. Step-by-step examples of the *arrival* and *departure mechanisms* of the Linked-List-Based (LLB) implementation of the VC Renamer. Only one PVC is shown for clarity with two VVCs (VVC0 and VVC1) mapped onto it. The various steps correspond to Algorithms 3 and 4.

function is illustrated in Figure 4. Again, the steps in both Algorithms 3 and 4 can be fully parallelized and overlapped in hardware, and they are off the router's critical path (see Section IV).

IV. Simulation Results

A. Simulation Platform

Both incarnations of VC Renamer were implemented within a cycle-accurate NoC simulator. The simulations assume wormhole switching, 4-stage pipelined routers, and deterministic XY routing. Each router consists of five physical ports, each with *four*, 8-deep PVCs. Every simulation runs for 1,000,000 clock cycles and each packet consists of five 32-bit flits. Our evaluation utilizes (a) synthetic Uniform Random (UR) traffic patterns in an 8×8 2D MESH network, and (b) traces from real applications running on the TRIPS [15] NoC-based multicore processor. The TRIPS processor includes a 4×10 mesh On-Chip Network (OCN) [15], which uses XY routing and 4 VCs per input port. We use traces extracted from 11 representative benchmarks of the SPEC CPU2000 Suite running on the TRIPS cycle-accurate simulator.

The *spatial distribution* of VC faults in the system is inspired by the model in [16]. A VC fault is assumed to disable one PVC of an input port. We explore two distributions of spatial VC fault placement: (1) Random (RM), where VC faults are *uniform-randomly distributed* throughout the NoC, and (2) Hotspot (HS), where VC faults are distributed only within a group of *spatially correlated* routers. In order to assess the robustness of VC Renamer, we vary the *percentage of faulty VCs in the whole NoC* from 1 to 10%. Each simulation was repeated 50 times and the results were averaged.

Finally, in order to evaluate the *hardware cost* (area and power overhead) of the proposed mechanisms, a conventional NoC router and both VC Renamer architectures were implemented in Verilog and synthesized in Synopsys Design Compiler using 65 nm commercial TSMC standard-cell libraries.

B. Analysis Of Results

We begin our evaluation with synthetic traffic patterns. We initially set the VC fault rate to 5% to see how VC Renamer fares as the traffic injection rate is varied. Each input port (in all designs) has four, 8-deep PVCs. We assume that the generic NoC design has *spare* VC buffers in *every* router input port to deal with VC faults. This, of course, amounts to an *enormous* overhead, which VC Renamer aims to eliminate by not relying on spare buffers *at all*. A fault in the MB and LLB designs is assumed to disable one of the 4 PVCs of an input port, thus *forcing two VVCs to be mapped to one of the remaining 3 PVCs*. Figure 5(a) assumes *RM* spatial fault distribution and compares the attained average network latency of VC Renamer to a generic NoC that is unaffected by the faults, because of the spare buffers (i.e., ideal scenario with immunity to faults). The MB and LLB implementations experience only 4.47% and 2.74% average drops in performance, respectively. *Throughput* decreases by only 4.96% and 0.52%, respectively. Similar trends are observed in Figure 5(b), which assumes *HS* fault distribution. The MB approach exhibits worse performance, because some cycles are *skipped* (i.e., nothing happens) during operation, as a result of the Step 2 condition check of Algorithm 1 and Step 3 of Algorithm 2.

The traffic injection rate is then set at 0.2 flits/node/cycle (in-between the zero-load and onset-of-saturation rates) and the VC fault rate is varied. Figures 5(c) and 5(d) illustrate the results assuming *RM* and *HS* spatial fault distributions, respectively. Note that the "Generic" latency in these figures is constant, since the generic design is unaffected by faults (ideal). Clearly, at low VC fault rates, the drop in performance is almost imperceptible with VC Renamer (as compared to an *ideal design unaffected by faults*). Even with 10% faulty VCs, the MB and LLB techniques experience modest 5.37% and 3.45% average drops in performance (over both spatial fault distributions), respectively.

Table I
Hardware synthesis results (65 nm): VC Renamer overhead over a generic NoC router implementation

	% Area Overhead			% Power Overhead		
# of VVCs per PVC	2	3	4	2	3	4
Mask-Based	2.09	2.87	5.36	0.23	2.33	3.71
Linked-List-Based	7.71	8.89	11.37	1.51	3.04	4.67

Figure 5(e) summarizes results of trace-driven simulations of real applications running on the TRIPS processor. A VC fault rate of 5% and *RM* spatial fault distribution are assumed. On average, the MB and LLB implementations experience 6.11% and 1.80% decreases in performance, respectively, as compared to the ideal, fault-free setting. Clearly, **both techniques are suitable for fault-tolerant designs**. If area/power overhead is an issue, the MB approach may be preferable, due to its lower overhead, as will be described shortly.

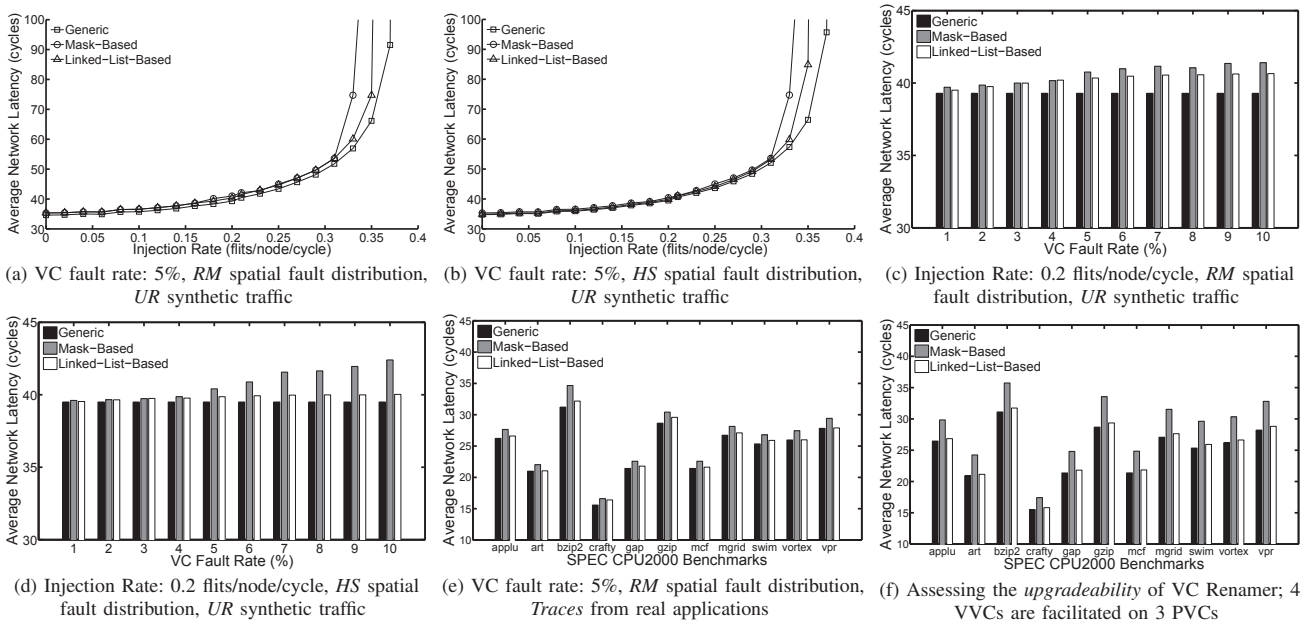
In order to assess the *upgradeability* aptitude of VC Renamer, we run the TRIPS [15] traces – which require 4 VCs/port – in a network with only 3 PVCs/port. Hence, VC Renamer is active in *all* router input ports in the *entire NoC*. Figure 5(f) shows comparison results with a generic NoC with 4 PVCs/port (ideal). For fairness, all designs have the same total number of buffer slots (e.g., equal buffer space) per input port. While the MB implementation experiences a 15.52% average drop in performance (because of excessive cycle skips attributed to the condition checks of Algorithms 1 and 2), the LLB implementation only suffers a 1.95% average decline. Clearly, **the LLB technique is more suitable for upgradeability purposes**, due to its minimal impact on performance, even when used in all routers simultaneously.

Table I shows the percentage area/power overhead of VC Renamer over a generic NoC design, as the number of mapped VVCs per PVC is varied. For example, if 2 VVCs are mapped per PVC, the number of supported VVCs is *double* that of the existing PVCs. For *doubling* the number of supported VCs (i.e., a huge flexibility boost), the MB implementation incurs minimal area and power overhead of 2.09% and 0.23%, respectively, while the LLB technique incurs area and power overhead of 7.71% and 1.51%, respectively. More importantly, our synthesis results also showed that the **critical path of the router was not affected** by either of the two VC Renamer architectures. The critical path still lies within the VC Allocation (VA) stage, which is *untouched* by VC Renamer (remember, VC Renamer simply changes the *mapping* of VVCs to PVCs; it does not interfere with the operation of the arbiters). All the new logic operates within the *slack* of the crossbar traversal and link-traversal/buffer-write stages.

As mentioned at the end of Section III-A, the VC Renamer employs a round-robin credit dispatch mechanism for the VVCs mapped to a particular PVC. This mechanism sends credits to each of the VVCs on a cycle-by-cycle basis. Figure 6 analyzes the impact of this mechanism on average network latency, as compared to an *ideal* credit mechanism that dispatches credits to *all VVCs simultaneously* and only a VVC that could make use of the credits actually uses them. It can be seen that the lightweight round-robin mechanism has negligible impact of 0.7% on performance when 2 VVCs are mapped to each PVC. For 3 VVCs/PVC, the latency increases by 2.7%, and for 4 VVCs/PVC it increases by 6.9%. Note, however, that at 2 VVCs/PVC, the number of supported VVCs already *doubles*. Hence, the credit mechanism has almost no impact on performance with this configuration.

V. Conclusion

Virtual channels are quintessential constructs in the correct operation of both the NoC routing algorithm and the CMP's cache coherence protocol. This paper introduces the notion of *VC Renaming*, which enables the further virtualization of existing VC buffers, in order to decouple the number of supported VCs in the system from the number of physically present VC buffers. The goals are (a) to enable the system



(a) VC fault rate: 5%, *RM* spatial fault distribution, *UR* synthetic traffic

(b) VC fault rate: 5%, *HS* spatial fault distribution, *UR* synthetic traffic

(c) Injection Rate: 0.2 flits/node/cycle, *RM* spatial fault distribution, *UR* synthetic traffic

(d) Injection Rate: 0.2 flits/node/cycle, *HS* spatial fault distribution, *UR* synthetic traffic

(e) VC fault rate: 5%, *RM* spatial fault distribution, *Traces* from real applications

(f) Assessing the *upgradeability* of VC Renamer; 4 VVCs are facilitated on 3 PVCs

Figure 5. Simulation results using both *synthetic traffic patterns* and *traces from real application workloads*. RM: Random, HS: Hotspot, UR: Uniform Random.

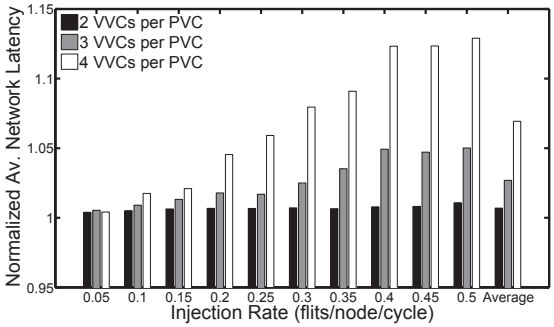


Figure 6. Comparison of average network latency achieved using the proposed round-robin (cycle-by-cycle) credit mechanism, as compared to an *ideal* mechanism that distributes credits *simultaneously* to all VVCs mapped to a particular PVC. The impact of the mechanism is assessed as the number of VVCs mapped to a PVC increases from 2 to 4. All results are *normalized to the setup with an ideal credit mechanism*.

to tolerate faulty VCs without reliance on expensive spare buffers, and (b) to accommodate routing algorithms and/or cache coherence protocols with varying VC requirements. Two different hardware implementations of the VC Renamer architecture are presented, which target different objectives (fault-tolerance vs. upgradeability). Both designs incur minimal hardware overhead and exhibit excellent performance without impacting the router's critical path. These results are very promising and demonstrate the viability of VC Renaming in future CMPs.

ACKNOWLEDGEMENT

This work falls under the Cyprus Research Promotion Foundation's Framework Programme for Research, Technological Development and Innovation 2009-10 (DESMI 2009-10), co-funded by the Republic of Cyprus and the European Regional Development Fund, and specifically under Grant ΤΠΕ/ΠΛΗΡΟ/0609(BIE)/09. The research leading to this paper is also supported by the European Commission FP7 project "Energy-conscious 3D Server-on-Chip for Green Cloud Services" (Project No:247779 "EuroCloud").

REFERENCES

[1] W.J. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," In *Proceedings of DAC*, 2001.

[2] J. Duato, S. Yalamanchili, and L.M. Ni, *"Interconnection Networks: An Engineering Approach,"* IEEE Computer Society Press, 1st Ed., 1997.

[3] N. Agarwal, L.S. Peh, and N.K. Jha, "In-Network Snoop Ordering (INSO): Snoopy coherence on unordered interconnects," In *Proceedings of HPCA*, 2009.

[4] S. Borkar, "Designing reliable systems from unreliable components: the challenges of transistor variability and degradation," In *IEEE Micro*, Nov-Dec 2005.

[5] S.R. Nassif, N. Mehta, and Y. Cao, "A resilience roadmap," In *Proceedings of DATE*, 2010.

[6] M. Modarressi, H. Sarbazi-Azad, and A. Tavakkol, "An efficient dynamically reconfigurable on-chip network architecture," In *Proceedings of DAC*, 2010.

[7] M.A.A. Faruque, T. Ebi, and J. Henkel, "Configurable links for runtime adaptive on-chip communication," In *Proceedings of DATE*, 2009.

[8] Y. Tamir and G.L. Frazier, "High-performance multi-queue buffers for VLSI communications switches," In *Proceedings of ISCA*, 1988.

[9] J. Park et al., "Design and evaluation of a DAMQ multiprocessor network with self-compacting buffer," In *Proceedings of Supercomputing*, 1994.

[10] N. Ni, M. Pirvu, and L. Bhuyan, "Circular buffered switch design with wormhole routing and virtual channels," In *Proceedings of ICCD*, 1998.

[11] Y. Choi and T. M. Pinkston, "Evaluation of queue designs for true fully adaptive routers," In *Journal of Parallel and Distributed Computing*, 2004.

[12] C.A. Nicopoulos et al., "ViChaR: A Dynamic Virtual Channel Regulator for Network-on-Chip Routers," In *Proceedings of MICRO*, 2006.

[13] M. Lai et al., "A dynamically-allocated virtual channel architecture with congestion awareness for on-chip routers," In *Proceedings of DAC*, 2008.

[14] J. Kim, "Low-cost router microarchitecture for on-chip networks," In *Proceedings of MICRO*, 2009.

[15] K. Sankaralingam et al., "Distributed Microarchitectural Protocols in the TRIPS Prototype Processor," In *Proceedings of MICRO*, 2006.

[16] A. Agarwal, D. Blaauw, and V. Zolotov, "Statistical timing analysis for intra-die process variations with spatial correlations," In *Proceedings of ICCAD*, 2003.