Elastic-Buffer Flow Control for On-Chip Networks

George Michelogiannakis, James Balfour and William J. Dally
Computer Systems Laboratory, Stanford University, Stanford, CA 94305
{mihelog,jbalfour,dally}@cva.stanford.edu

Abstract

This paper presents elastic buffers (EBs), an efficient flow-control scheme that uses the storage already present in pipelined channels in place of explicit input virtual-channel buffers (VCBs). With this approach, the channels themselves act as distributed FIFO buffers. Without VCBs, and hence virtual channels (VCs), deadlock prevention is achieved by duplicating physical channels. We develop a channel occupancy detector to apply universal globally adaptive load-balancing (UGAL) routing to load balance traffic in networks using EBs. Using EBs results in up to 8% (12% for low-swing channels) improvement in peak throughput per unit power compared to a VC flow-control network. These gains allow for a wider network datapath to be used to offset the removal of VCBs and increase throughput for a fixed power budget. EB networks have identical zero-load latency to VC networks operating under the same frequency. The microarchitecture of an EB router is considerably simpler than a VC router because allocators and credits are not required. For 5×5 mesh routers, this results in an 18% improvement in the cycle time.

1 Introduction

Semiconductor technology scaling allows more processing and storage elements to be integrated on the same die. Networks-on-chip (NoCs) provide a scalable communication infrastructure [5, 7]. With wires available in abundance on chip, channel bandwidth is plentiful, rendering buffers and flip-flops (FFs) expensive in comparison [5]. Thus, reducing buffering requirements and implementing buffers more efficiently increases network efficiency.

This paper presents elastic buffers (EBs), an efficient flow-control scheme that uses the storage already present in pipelined channels instead of input virtual-channel buffers (VCBs). Removing VCBs reduces the area and power consumed by routers, but prevents the use of virtual-channel flow-control for performance and deadlock avoidance. Duplicate physical channels can be used in the same way as virtual channels (VCs) to prevent deadlocks and to define traffic classes. Dividing a network into sub-networks provides duplicate physical channels and increases performance and power efficiency [1] in virtual-channel networks (VCNs), which we find to hold for elastic-buffered networks (EBNs).

We use universal globally-adaptive load-balanced (UGAL) routing [18] in a flattened butterfly (FBFly) [11] network to balance network load. We develop a channel congestion sensing mechanism for EBNs that is used in place of the credit count that is typically used to sense congestion in VCNs.

For a FBFly with UGAL routing, performance and power efficiency are almost equal for EBNs and VCNs with low-swing channels, with a 3% loss for EBNs with full-swing channels. EBN gains in 2D mesh networks reach 8%, increasing to 12% with low-swing channels. EBN gains depend on the fraction of cost represented by VCBs. Since the EBN datapath must be widened to compensate for the removal of VCBs, EBNs tend to occupy more area. For a fixed area budget, datapath widths are similar, and VCNs are more area efficient. Zero-load latency is equal for VCNs and EBNs operating at the same clock frequency.

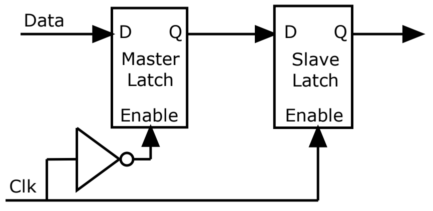
EB routers are simpler than VC routers because they use arbiters rather than allocators, and do not use credit-based flow-control. Consequently, the critical paths in an EB router are shorter than a comparable VC router. An EB router can operate at a faster clock frequency. For 5×5 mesh routers, EB routers have an 18% reduced cycle time.

The rest of this paper is organized as follows: Section 2 discusses the basic building blocks of EBNs as well as deadlock avoidance, topology choices and adaptive routing. Section 3 presents performance, area and power evaluations. Section 4 discusses EBNs and trade-offs. Section 5 outlines router implementation results. Finally, sections 6 and 7 conclude and discuss related work.

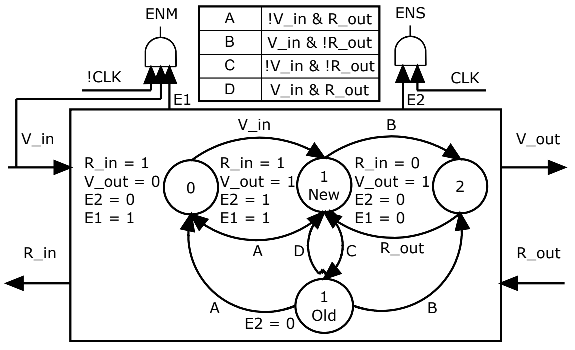
2 Elastic Buffer Networks

In this section we present EB channels and routers, the fundamental EBN building blocks. We then discuss channel

To appear in the 15th International Symposium on High-Performance Computer Architecture (HPCA-15).

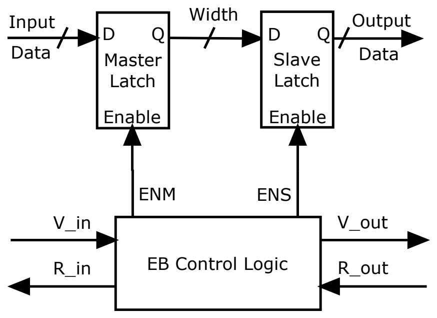


(a) A DFF implemented with a master and a slave D latch.

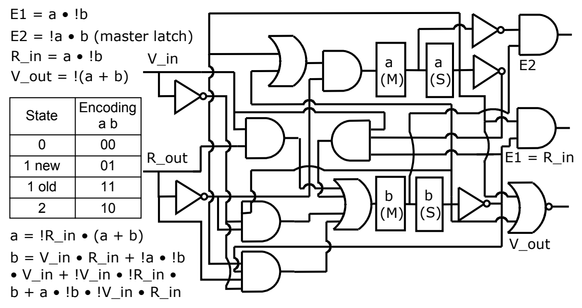


A	!V_in & R_out
B	V_in & !R_out
C	!V_in & !R_out
D	V_in & R_out

(c) Expanded view of the EB control logic as a synchronous FSM for two-slot EBs.



(b) An EB is a FF whose latches are controlled by the EB control logic and used for buffering.



E1 = a • !b
E2 = !a • b (master latch)
R_in = a • !b
V_out = !(a + b)

State	Encoding a b
0	00
1 new	01
1 old	11
2	10

a = !R_in • (a + b)
b = V_in • R_in + !a • !b • V_in + !V_in • !R_in • b + a • !b • !V_in • R_in

(d) Logic diagram of the EB control FSM with 2 FFs and 10 gates.

Figure 1. EB Channels.

duplication, deadlock avoidance, and adaptive routing.

2.1 Elastic Buffer Channels

Figure 1(a) shows a D flip-flop (DFF) [19] that is implemented using master and slave latches. By adding control logic to drive the latch enable pins independently, each latch can be used as an independent storage location. Thus, the FF becomes an EB, a FIFO with two storage locations. This is illustrated in Figure 1(b).

EB channels use many such EBs to form a distributed FIFO. FIFO storage can be increased by adding latches to EBs or by using repeater cells for storage [15]. EBs use a ready-valid handshake to advance a flit (flow-control digit). An upstream *ready* (R) signal indicates that the downstream EB has at least one empty storage location and can store an additional flit. A downstream *valid* (V) signal indicates that the flit currently being driven is valid. A flit advances when both the ready and valid signals between two EBs are asserted at the rising clock edge. This timing convention requires at least two storage slots per clock cycle delay to avoid creating unnecessary pipeline bubbles.

The finite state machine (FSM) implementing the EB control logic is illustrated in Figure 1(c), and the corresponding logic is shown in Figure 1(d). The FSM outputs that control the latch enables are qualified by the clock in the same manner as FFs. A flit is latched in the master latch at the end of the cycle and advances to the slave at the beginning of the next cycle. The initial state, state *0*, encodes that there are no valid flits stored. E2 can be 0 in state *0* to save power or 1 to simplify FSM logic. Once a flit arrives (*V_in*=1), the FSM enters the state *1 new*, because the arriving flit is stored at the master latch at the beginning of the cycle and thus needs to be moved to the slave latch by asserting E2. If the flit does not depart and no new flit arrives, the FSM enters the state *1 old*, because the flit already resides in the slave latch and it must not be overwritten. In state *2*, there are two stored flits which makes the FSM deassert both latch enables to prevent overwriting. The master latch enable can be further qualified by the incoming valid to avoid clocking the master latch when there are no incoming flits. The total FSM cost is 2 FFs and 10 logic gates. Two more AND gates are needed to qualify the FSM latch enable outputs by the clock. Since flow-control is applied at a flit granularity, control logic is amortized over the width of the channel. The channel datapath is unaffected.

All components use the same clock. To avoid penalizing its frequency, we hide the FSM setup time and clock-to-q latency. Since the EB master latch is enabled only when the clock is low, the first half of the clock cycle is long enough for the single AND gate to drive E1. However, E2 needs to be stable during the first half of the clock cycle. Therefore, we split the FFs holding state bits *a* and *b* into their master and slave latches. The outputs of the master latches are used to generate E2, allowing E2 to stabilize before the end of the previous clock cycle.

To cover the overhead of the ready and valid ports, the ready and valid wires are optimized for delay by promoting them to higher metal layers with lower RC delays, by increasing wire width and spacing, or by engineering the wires for delay using more aggressive repeaters. The ready-valid overhead consists of an 1 fanout-of-four (FO4) latch data-to-q delay, 3 FO4 logic delay (for the three levels of gates in the FSM next-state and output functions), and 3 FO4 to fanout the latch enables to the channel EBs, for a total of 7 FO4. In our implementation technology, explained in section 3.1, if we assume 1mm EB spacing, the 7 FO4 overhead can be hidden by targeting a ready and valid wire delay of 360ps/mm instead of 500ps/mm. The minimum propagation delay for wires engineered solely for delay is 150ps/mm in our technology.

2.2 Router Architecture

Unlike input-queued VC routers [16], EB routers do not use VCs and thus do not have VCBs and VC allocators. Furthermore, since inputs may request only one output, EB routers use output arbiters instead of switch allocators. Each input has an EB to buffer flits and ensure the first stage has a full clock cycle in which to operate. Credits are not used to track occupancy at the downstream router, and thus there are no credit channels. The ready-valid handshake is used to determine when output EBs are ready to receive flits and to detect valid flits arriving at the inputs. Arbitration is performed on a per-packet basis to prevent the interleaving deadlock described in section 2.6.

A block diagram of an EB router is shown in Figure 2. Only one input and one output are illustrated in detail. The router consists of two pipeline stages. Routing computation (RC) and switch arbitration (SA) occur in the first stage, and switch traversal (ST) in the second. Routing decisions are stored in state registers when head flits arrive at the RC block and are used to forward body flits. After routing, each head flit requests its output from an arbiter in the SA block. After a head flit wins arbitration, the output is held for the duration of the packet to prevent interleaving of flits from multiple packets. The SA block forwards the *ready* signal from the output EB back to the input EB so that flits only advance when there is room for them in the output buffer. When the input EB receives a *ready* signal, the flit advances to the intermediate pipeline register. During the next cycle, the flit traverses the switch and is stored into the output EB. An extra storage slot is required in the output EBs to cover the pipeline latency because arbitration is performed one

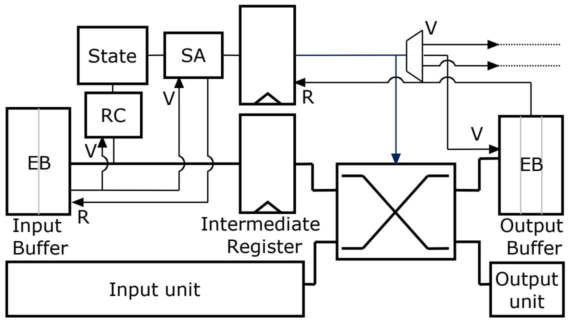


Figure 2. EB Router microarchitecture.

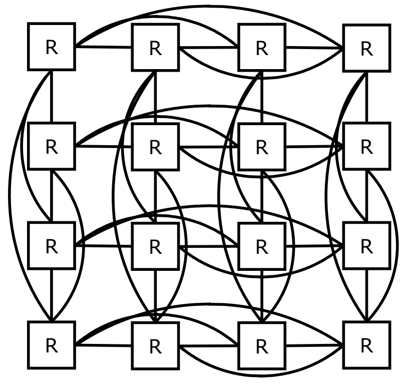


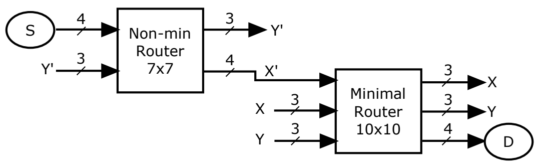
Figure 3. A 2D 4×4 FBFly.

cycle in advance of switch traversal (and without credits). Each output EB asserts its *ready* signal when it has two or more free slots or when it has one free slot and *ready* was low on the previous cycle—so there is no flit currently in transit.

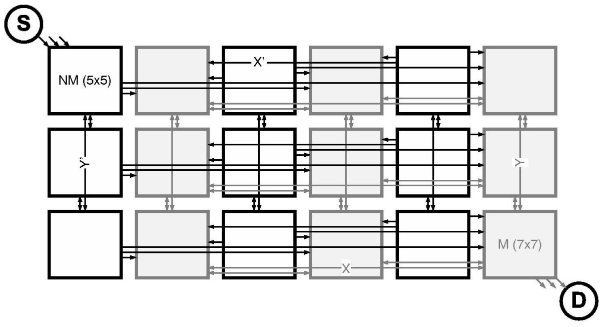
We also consider an EB router with EBs at the crosspoints of the switch. The ready-valid handshake facilitates movement to and from the crosspoints. This design improves performance if entire packets can be buffered at the crosspoints. Such packet-sized buffers, however, are prohibitively expensive for realistic packet sizes. Also, since a FF is 5.3 times larger than an inverter of twice the driving strength in our technology library, adding an EB at every crosspoint makes the crossbar area dominated by the crosspoints even for a small number of incoming and outgoing wires. Apart from the increased area, a pair of ready and valid wires to and from each crosspoint EB is required, causing an increased power consumption. This effect becomes worse when trying to design a low-swing crossbar. Therefore, we do not consider a buffered crossbar further.

2.3 Topologies

This work focuses on the 2D mesh and 2D FBFly [11] topologies. The 2D mesh is a common topology choice because it is simple. The 2D FBFly is similar to the 2D mesh, except that every router connects to every other router in the same row and column instead of only its neighbors, as illustrated in Figure 3. Topologies with non-uniform channel



(a) The two routers, one for each sub-network. Illustrated for a 4×4 UGAL FBFly.



(b) A 3×3 UGAL FBFly. Non-minimal router (NM) connect to minimal routers (M – shaded) via X' channels.

Figure 4. The UGAL FBFly. Two traffic classes are defined by two sub-networks.

lengths, such as the FBFly, require VC routers with different amounts of buffering at each input to account for the different credit round-trip delays, which increases the design complexity, or with large enough buffers to accommodate the longest channels. EBNs avoid this because they do not use credits.

Using UGAL routing (section 2.5) in an EBN requires separate traffic classes to handle minimal and non-minimal traffic to prevent cyclic dependencies (section 2.6). As discussed in section 2.7, the most efficient way to implement these traffic classes is to provide separate sub-networks for minimal and non-minimal traffic. A single FBFly router is replaced by a router for each of the two sub-networks, as illustrated in Figure 4(a). The X' and Y' channels handle non-minimal packets while the X and Y channels handle minimal traffic. The Y' channels connect the non-minimal routers to other non-minimal routers in the same column. The X' channels connect the non-minimal routers to the minimal routers in the same row. After a packet takes an X' hop it is done with non-minimal routing and proceeds to its destination on the minimal sub-network. One of the X' channels connects each non-minimal router to the minimal router with the same coordinates. This handles packets that do not require a non-minimal hop in X. Traffic sources inject traffic to the non-minimal sub-network. Destinations eject traffic from the minimal sub-network. A 3×3 UGAL FBFly is shown in Figure 4(b).

2.4 Sensing Channel Congestion

The UGAL [18] routing algorithm we use for the FBFly topology requires a means of estimating channel congestion. We evaluate five different congestion metrics for EBNs that can be used instead of the credit counts used by VCNs: blocked cycles, blocked ratio, output occupancy, channel occupancy, and channel delay.

Blocked Cycles is a running average of the number of cycles an output is blocked. Once an output is blocked, a counter keeps track of the number of clock cycles until it unblocks. At that point it updates the running average. The new value for running averages is calculated as $newvalue = 0.3 \times oldvalue + 0.7 \times newsample$.

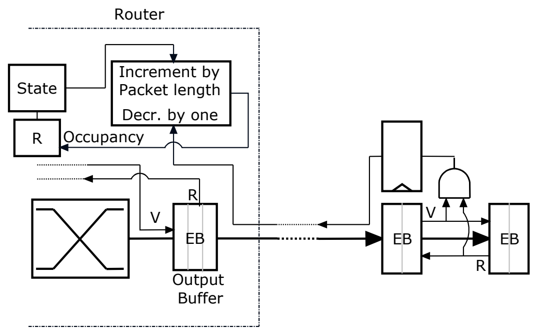


Figure 5. Measuring output occupancy.

Blocked Ratio is the ratio of the number of cycles an output has been blocked, divided by its unblocked cycles. We calculate this for the 20 most recent clock cycles.

Output Occupancy is the number of flits currently committed to an observation region of the channel. When the head flit of a packet is routed, the occupancy counter for its output is incremented by the packet length in flits. Output counters are decremented by one for each flit leaving the observation region, the output channel segment for which we keep track of flits in transit.

We detect flits leaving the observation region using an AND gate whose inputs are the ready and valid signals between the last EB of the region of observation and the next EB. A circuit to measure output occupancy for a single output is shown in Figure 5. The time required to propagate the AND gate output back to the router, determined by the length of the observation region, affects the speed of congestion sensing. As discussed in section 2.1, that wire can be optimized for delay to accommodate larger regions. In our study, the observation region is the same length as the shortest network channel. Propagation delay is half of that channel's delay, rounded up.

Channel Occupancy is similar to output occupancy, except that an output's counter is incremented when a flit ar-

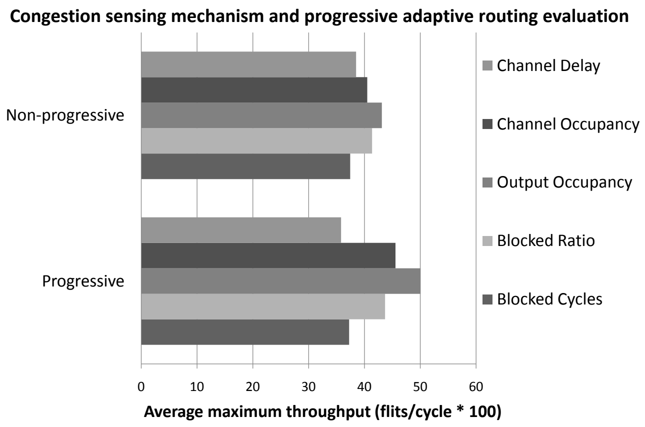


Figure 6. EBN Throughput comparison.

rives at the output EB instead of when its packet is routed to that output.

Channel Delay measures the average number of cycles needed by flits to leave the region of observation. A FIFO at each output stores timestamps of flits entering the output EBs. When a flit is detected to leave the region, the timestamp at the head of this FIFO is removed and used to determine the delay and update the running average.

Figure 6 compares the performance of these five congestion sensing mechanisms. The setup used in the simulation is explained in section 3.1. The metric of choice is output occupancy. Occupancy metrics are also the simplest. Metrics using running averages fail to adapt quickly to the current network status. Output occupancy is preferable to channel occupancy because it is important to account for flits that have been routed and waiting for an output as well as flits that have advanced to it. Otherwise, a router input is not aware of other inputs' choices. Thus, an output channel which is never blocked appears as not loaded even though many flits may be blocked waiting for that output because all inputs are making the same choice.

2.5 Adaptive Routing

After flits are injected into the non-minimal network, UGAL chooses a random intermediate destination, I. The packet is routed to I (non-minimally to the final destination, D) if the load of the output port on the route to I, multiplied by the non-minimal hop count to D, is larger than the load of the output port on the route to D multiplied by the hop count of the minimal route to D. That decision is never revisited.

Non-minimal packets are routed in Y'X' dimension order, to I in the non-minimal network and then in YX dimension order to D in the minimal network. Each dimension of each sub-network is traversed with a single hop. Packets that are routed minimally proceed directly to D in X'Y dimension order. The X' hop places them into the minimal network. If a non-minimal route has been chosen but there is no X' hop, the X hop becomes an X' hop since flits would have reached I in the non-minimal network. Therefore, routing becomes Y'X'Y. The self-channel is used when a traversal to the minimal network is required but there is no X' hop. Thus, an extra hop is created. Routing Y'X'YX results in better column load balancing than routing Y'X'XY, since in the minimal network flits traverse the randomly chosen column of I. Routing Y'X'XY makes flits use the column of D, which, depending on the traffic pattern, can result in load imbalance.

Although UGAL attempts to balance the load, routers have congestion information only for their outputs and are unaware of more distant congestion. To deal with distant congestion, we apply progressive adaptive routing (PAR)[1] which revisits the choice of I. For every non-minimal hop towards I, another intermediate destination is chosen randomly that would result in an output in the same axis as the output towards I, to preserve dimension ordering. UGAL is applied to choose the best option among I and the new intermediate destination. That option then becomes I. If I is not reached with that hop, routing proceeds in the next dimension. Thus, there is still only up to one hop in every dimension, which ensures forward progress. When the most recently chosen intermediate destination I is reached, routing proceeds to the minimal network. In addition, when taking an X' channel to the minimal network, another X' channel is randomly chosen and UGAL is applied to keep the best option. Routing in the minimal network is deterministic. PAR is not applied in the minimal network since the final destination is constant. An evaluation of PAR is included in Figure 6. Disabling PAR reduces the maximum throughput by 14% for output occupancy.

Because non-minimal routers have no X' channels to other non-minimal routers, they have a smaller radix. Therefore, they have shorter critical paths and more cycle time to make the complex adaptive routing decisions. However, having two separate networks means that all adaptive routing decisions are made in the non-minimal network, to preserve dimension-ordering in the minimal. These decisions cannot consider the load of the minimal network because the minimal network routers the flit would traverse are distant. Moreover, VCNs share channels and thus their loads. Therefore, packets are routed non-minimally when another output of the router making the decision is less loaded. This is not feasible with separate networks because load imbalance in the minimal network cannot be readily resolved by the routers attached to minimal network channels, to preserve dimension-ordering. PAR is another cause for load imbalance in the minimal network, because it makes the intermediate destination choice not truly random. Otherwise, traffic from the intermediate nodes to the minimal network would be uniform random which would result in

[1] A slightly different form of progressive adaptive routing was first proposed by Steve Scott of Cray for routing in a Dragonfly Network.

balanced load, on average.

Furthermore, traffic from sources and destinations connected to same-coordinate routers now need to change to the minimal network, using network channels to make the traversal. This makes them contend with other network traffic, which is not true without sub-networks. This issue can be solved by increasing network cost by adding more channels, such as ejection ports in the non-minimal network.

2.6 Deadlocks

EBNs use duplicate physical channels in the same manner as VCs to define disjoint traffic classes and prevent deadlocks. Moreover, another type of deadlock, interleaving deadlock, appears in EBNs.

Protocol (request-reply) deadlocks [9] are solved by guaranteeing that replies are always able to reach destinations, bypassing blocked requests to the same destinations. Destinations may be waiting for those replies before they can serve more requests. More complex protocols may require more traffic classes. Cyclic dependencies [4] are formed by a series of packets each depending on one another in a cycle in order to progress. They are broken by enforcing dimension-ordered routing (DOR) in topologies without physical channel cycles, such as our chosen 2D mesh and 2D FBFly. Otherwise, enough disjoint traffic classes must be formed such that no cyclic dependency is possible within and across classes.

EBNs suffer from another type of deadlock: interleaving deadlock. Due to the FIFO nature of EB channels, flits at the head must be sent to their outputs to let the next flit coming to the same input be processed. If packets are interleaved, blocked packets also block tails of other packets that are interleaved with them. Therefore, a cyclic dependency can be formed between the interleaved packets. For instance, a new head flit may arrive at an input port with all its routing computation registers occupied. Thus, that head flit depends on tail flits from the packets occupying those registers and interleaving with the new packet, to have routing computation registers released. However, those tail flits also depend on that head flit in order to advance.

This deadlock can be easily avoided by preventing packets from being interleaved. This guarantees that a head flit is followed by all the remaining flits of the same packet before another packet's flits arrive. Therefore, only one routing computation status register is required for each input port. Without VCs, disabling packet interleaving does not degrade network performance assuming that sources transmit flits of the same packet contiguously. To the contrary, since packets are considered to have been delivered once their tails arrive, this may decrease average packet latency. Interleaving may result in tails arriving late, behind flits from a number of other packets. Disabling interleaving also makes mantaining packet identifiers in flits unnecessary.

2.7 Duplicating Physical Channels

Duplicating physical channels allows EBNs to differentiate between traffic classes and provide isolation. We consider three ways of duplicating channels and conclude that using duplicate sub-networks is the preferred choice.

The first option is duplicating physical channels between routers but multiplexing them into the same router port. To maintain non-interleaving, multiplexers and demultiplexers must select on a per-packet basis. Moreover, routers need duplicate output EBs to prevent flits of different classes from interacting. This option yields a small performance gain since router ports remain the same, but there is a disproportional increased overhead due to having an increased channel clock and output EB area and power costs.

The second option is duplicating physical channels and router ports. We find this to yield an excessive overhead due to the crossbar cost increasing quadratically with the number of router ports.

For the same reason, dividing into sub-networks—the third option—proves to be an efficient choice, as already shown for VCNs [1]. It doubles its available bisection bandwidth and therefore its cost. However, when narrowing channels down to meet the same power or performance (maximum throughput) budget as the single network, crossbar cost decreases quadratically. This results in a more performance and power efficient overall network, enabling us to maintain a higher bisection bandwidth and performance with the same power budget.

3 Evaluation

In this section we present evaluation results for EBNs. We explain our methodology in section 3.1. We then present results in section 3.2.

3.1 Methodology

Evaluation is performed with a modified version of booksim [6]. We compare EBNs and VCNs using a 2D mesh with DOR, and the 2D FBFly using UGAL routing as described in section 2.3. For a fair comparison, both EBNs and VCNs have separate request and reply sub-networks. The 2D mesh has 16 terminals with 16 routers arranged in a 4×4 grid. Injection and ejection channels have 1 cycle of latency, and mesh network channels have 2 cycles of latency.

For the FBFly, we consider a 64-terminal network with 16 routers arranged in a 4×4 grid and four terminals per router. Short, medium and long channels have 2, 4 and 6 cycles of latency, respectively. Self-channels have 1 cycle of latency. The EB request and reply sub-networks each have 32 routers, 16 7×7 routers for the non-minimal sub-network and 16 10×10 routers for the minimal sub-network. Progressive adaptive routing, as described in section 2.5, is applied to all FBFly networks.

Sources generate fixed-size 512 bit packets according to their injection rate and enqueue them to the network interface buffer of the proper sub-network. Non-empty network interface buffers inject one flit per cycle to their sub-network, and flits are ejected from each sub-network at a rate of one flit per cycle.

The set of traffic patterns [6] used for the evaluation is uniform random, random permutations, shuffle, bit comparison, tornado, and neighbor traffic for the 2D mesh. For the FBFly we also include transpose and an adversarial traffic pattern. The adversarial traffic pattern aims to load specific network channels by making all sources connected to one router send to destinations connected to one other router. Transpose illustrates the effect of traffic destined to the same-coordinate router now contenting with other network traffic to switch to the minimal network, as explained in section 2.5. Results are averaged over the set of traffic patterns for each sample point. The maximum throughput is the average of the maximum throughput of each traffic pattern. The consumed power is the average of the power consumptions at the maximum throughput of each traffic pattern.

Area and power models are based on those described in [1]. We model channel wires as being routed above other logic and report only the area of the repeaters and FFs. Router area is estimated using detailed floorplans, and input buffers are implemented as SRAM. Device and interconnect parameters are for a 65nm general-purpose CMOS technology. We use a clock frequency of 2GHz, about 20 FO4 inverter delays. Critical devices in the channels and router datapaths, such as the repeaters used to drive large wire capacitances, are sized to ensure circuits will operate at the clock frequency. The power model includes all of the major devices in the channels and routers, and includes leakage currents. The flip-flops in the channels are clock-gated locally. Aggressive low-swing channel designs can achieve up to a 10x traversal power per bit reduction compared to full-swing [10]. As a conservative estimate, our low-swing channel model has 30% of the full-swing repeated wire traversal power, and double the channel area.

VCNs use a two-stage router design. The first stage consists of input buffering, routing, and VC and switch allocation. The second stage is switch traversal. We do not assume input buffer bypassing. Our study focuses at the saturation points at which this has minimal effect. We set the number of VCs and buffer slots to maximize the VCN performance and power efficiency. For each VCN datapath width used in the simulations, we sweep the number of VCs and buffer slots to maximize the ratio of the average maximum throughput over the consumed power per unit of throughput (power divided by throughput). We only consider buffer depths that cover the credit round trip latency to avoid penalizing latency. For 64-bit wide channels, 4 VCs is the optimal choice for both DOR and UGAL on the FBFly, of 10 slots each for full-swing and 8 slots for low-swing channels. For the mesh, the optimal choice is 4 VCs of 9 slots each for full-swing and 8 for low-swing channels.

3.2 Results

Figure 7 shows Pareto optimal curves for the FBFly and the 2D mesh for full and low-swing channels. The curves were generated by sweeping datapath width from 28 to 192 bits in numbers divisible by the packet size. Points represent an optimal design point, associating power consumption and maximum throughput, or area and maximum throughput. Thus, they illustrate the characteristics of a network with a certain area, power or performance budget.

Figure 8 presents throughput-latency curves for uniform traffic and 64-bit channel widths. The VCN was configured for low-swing channels.

Figure 9 shows an area and power breakdown, by network component, for the full-swing model. Figure 10 presents the same power breakdown for the low-swing model. All results are for a single DOR FBFly network with a 64-bit datapath under uniform random traffic, making EBN and VCN flits traverse the same paths. This provides a fair comparison given the different congestion metrics for UGAL routing. Figure 11 presents a power breakdown for the 2D mesh with low-swing wires, under a 2% packet injection rate for uniform random traffic. VCB power is 25% of the overall, compared to 21.5% for the DOR FBFly. Channel traversal refers to the power to traverse a segment with repeaters. For EBNs, input buffer read power is the traversal power for the intermediate router register shown in Figure 2. Low-swing networks have the same area breakdown except for the channel area which is doubled.

Table 1 summarizes the percentage gains for each of the two topologies. For each comparison, EBNs and VCNs have the metric shown in the first column equalized by adjusting the EBN datapath width. The percentage gains compare VCNs and EBNs against the two other aspects. That percentage is calculated by calculating the distance at each sampling point between the Pareto curve that associates each aspect under comparison with the normalized aspect, dividing by the value of the aspect under comparison, and averaging among all sampling points. A positive percentage means that EBNs provide gains for that comparison, and thus the percentage was calculated against the aspect under comparison value of VCNs. Likewise, negative percentages indicate gains for VCNs.

This section assumes a constant clock frequency, thus ignoring EBN gains presented in section 5.

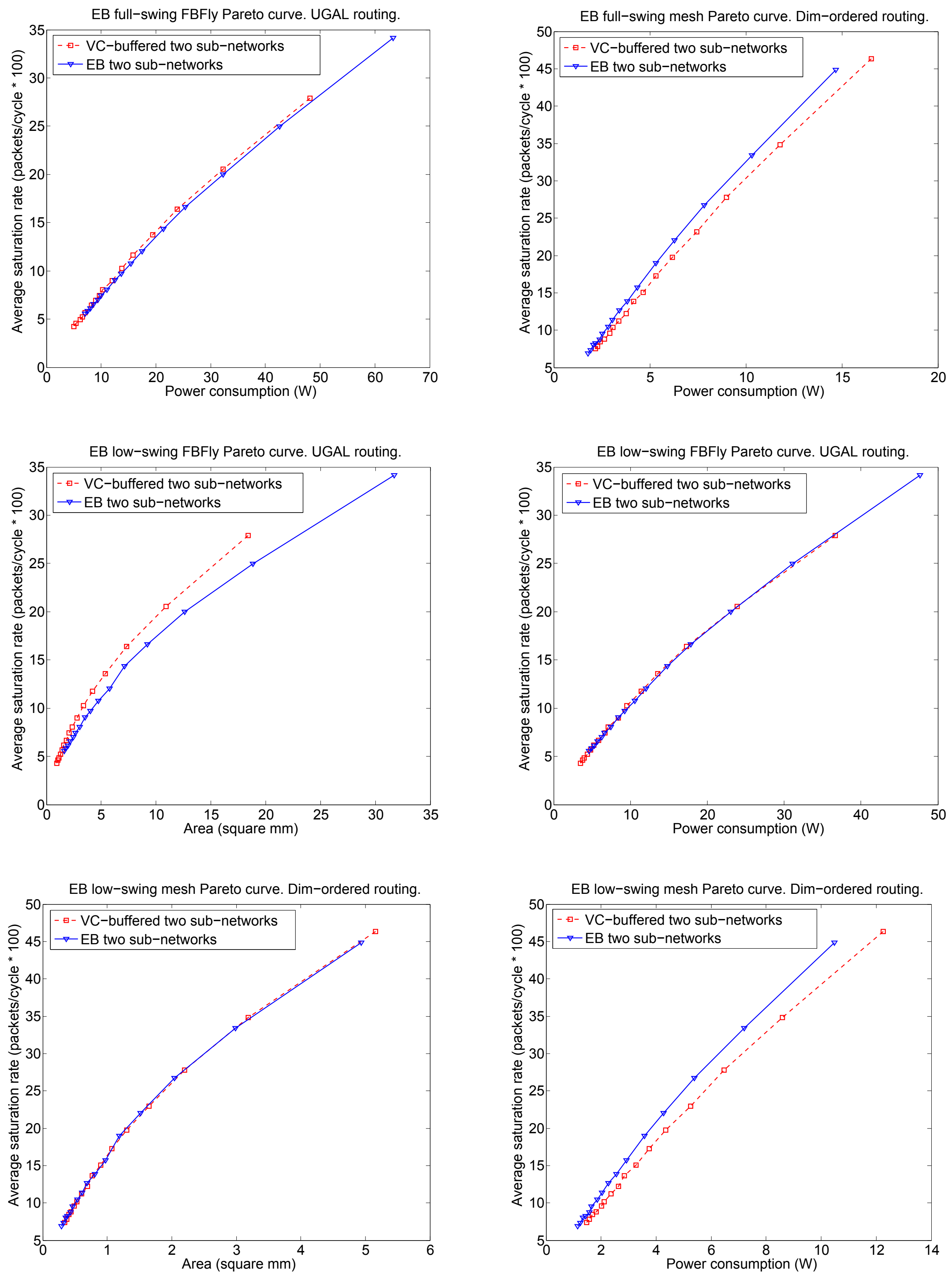


Figure 7. Paretto optimal curves for full and low-swing channel networks.

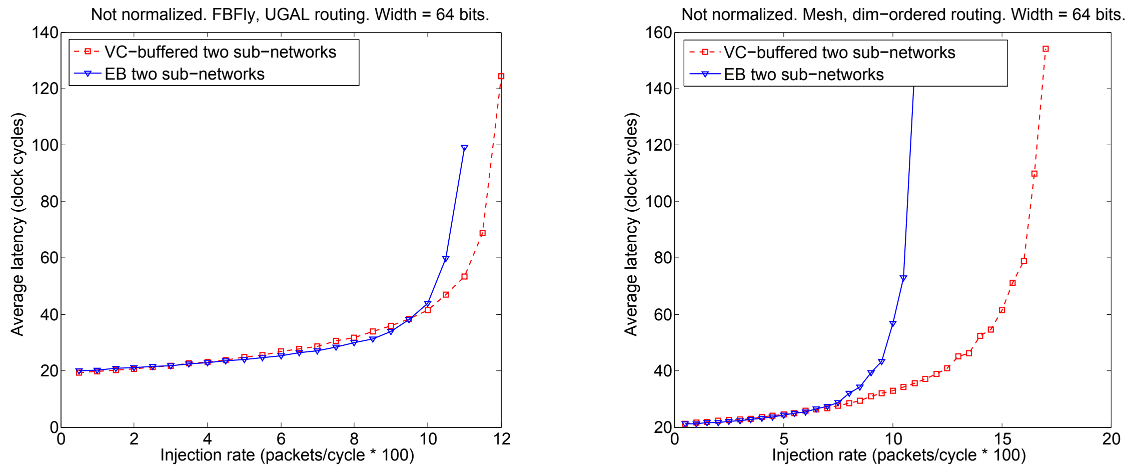


Figure 8. Throughput-latency curves for uniform random traffic.

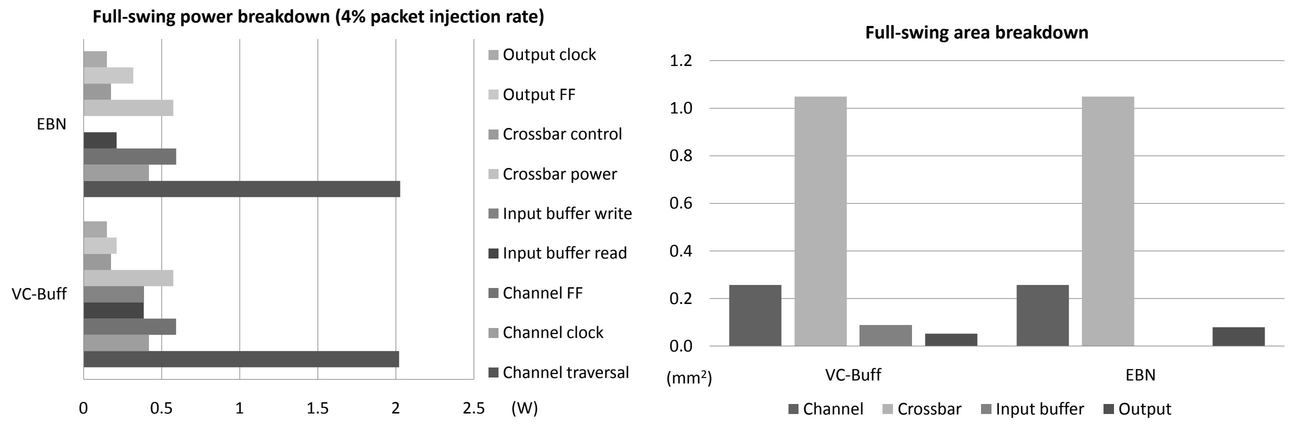


Figure 9. Cost breakdowns for a DOR FBFly with 64-bit full-swing channels under uniform traffic.

4 Discussion

As the results show, EBNs yield larger gains in a 2D mesh. In a mesh, each flit takes many router hops, incurring buffering costs at each router. Thus, a flit in the mesh network consumes more energy to reach its final destination than a flit in the FBFly, and VCB energy is a more significant portion. Therefore, 2D mesh EBNs exhibit greater power savings because more energy is consumed in the VCBs. The additional power consumed in the intermediate EBN router pipeline register is much less than the input VCB. These results demonstrate that topology, specifically as it affects average hop counts, affects the gains that are realized by EBNs.

The dominant portion of the overall power for the full-swing model is the power to traverse the repeated wires in the channels. Low-swing channels reduce the overall power, making VCB power a more significant portion of the overall network power. Therefore, EBNs yield a greater power improvement from removing VCBs in a network that uses low-swing channels.

Our analysis assumes efficient VCBs implemented with SRAM cells. Latch or FF array implementations are less efficient, increasing VCB overhead and making EBNs more attractive. Doubling channel area due to differential wiring has a small impact because the majority of the area is occupied by the router crossbar.

Table 1. EBN Percentage gains.

Norm	DOR Mesh			UGAL FBFly		
Comp:	Area	Perf	Power	Area	Perf	Power
Full-swing						
Area	-	1%	7%	-	-10%	10%
Perf	2%	-	8%	-20%	-	-3%
Power	-11%	8%	-	-16%	-2%	-
Low-swing						
Area	-	2%	10%	-	-11%	15%
Perf	2%	-	12%	-23%	-	0%
Power	-15%	10%	-	-24%	0%	-

EBNs consume less power compared to VCNs of the same datapath width. To normalize EBNs and VCNs for a fixed power budget, we increase the EBN datapath width. In that case, EBNs can support a higher maximum throughput. Alternatively, we can normalize the two networks for

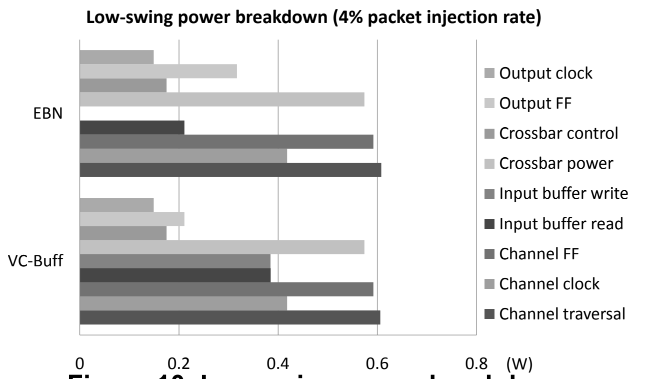


Figure 10. Low-swing power breakdown.

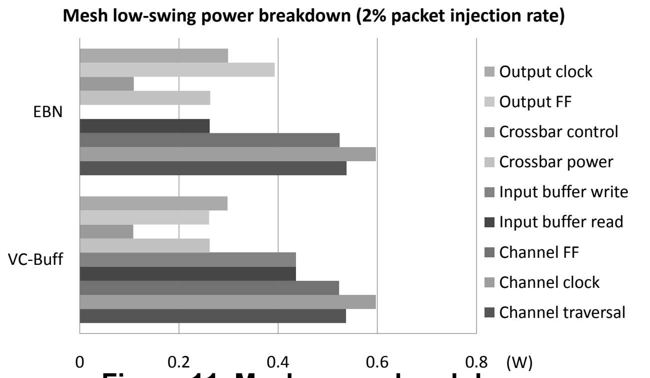


Figure 11. Mesh power breakdown.

the same maximum throughput by choosing another EBN datapath width. In that case, EBNs can consume less power. Therefore, EBNs are more performance and power efficient compared to VCNs. SRAM VCBs occupy a small amount of area. Thus, area normalization makes datapath widths equal or almost equal, resulting in similar networks as bisection bandwidth normalization.

Due to the FIFO nature of EB channels, packets in the same sub-network share the same resources. Thus, a blocked flit blocks all flits behind it, making EBNs more sensitive to contention. As a solution, more duplicate sub-networks can be provided. Consequently, quality of service (QoS) guarantees are only possible across sub-networks.

Therefore, the effect of having VCBs in networks compared to EBNs of the same datapath width is higher link utilization and maximum throughput with a higher maximum power consumption, but with a favorable $\frac{performance}{power}$ ratio. Designs that need the highest performance possible with a fixed area budget should use VCNs. This is because the SRAM VCBs occupy little area, so removing them yields minimal area savings. However, increasing the datapath width increases the crossbar area quadratically.

As explained in section 2.7, dividing the network into sub-networks improves performance and power efficiency. However, this is only true up to a certain number of sub-networks. After that, each sub-network's control overhead dominates and serialization latency becomes significant. Moreover, duplicating networks increases source and destination network interface radix. Thus, for each network configuration there is an optimal number of sub-networks for performance and power efficiency. VCNs are a favorable choice for complex protocols or adaptive routing algorithms requiring more sub-networks than that number.

The amount of buffering in EB channels scales directly with channel length, affecting EBN performance. Topologies with double the channel length have an increased performance of 8-10% in the UGAL FBFly, but on the other hand almost double the channel power for a total power increase of approximately 60%. Topologies with half the channel length show a similar trend. VCN performance is affected by the different size of VCBs, due to the different round-trip time. The 2D mesh shows a similar trend. This shows that the dominant factor affecting maximum throughput in EBNs is the contention in the bufferless routers. However, designers might still find it beneficial to add storage slots in channels, as explained in section 2.1, depending on their topology, layout and router radix.

EB channels have the same zero-load latency when measured in clock cycles, since the ready-valid protocol described in section 2.1 does not add any latency. Moreover, the EB router described in section 2.2 consists of two stages, as the assumed VC router model. We also assume that VCB depth is large enough to cover the credit processing and propagation delay. Zero-load latency in the EB UGAL FBFly is increased by 3.3% by average, due to the extra hop very few flits have to take, as explained in section 2.5. As the injection rate increases, latency increases in a similar manner between EBNs and VCNs. However, EBNs saturate earlier for equal-width datapaths due to their sensitivity to contention. Datapath width affects serialization latency and thus zero-load latency. In practice, as discussed in section 5, EBN routers are more likely to require fewer cycles to traverse, or will operate at a higher clock frequency, due to their simplified design.

EBNs provide a higher throughput for the same power budget, or consume less power for the same throughput. However, designs with strict area budgets which prioritize performance will find VCNs to be the preferable choice. Designs that need other features from the network should investigate how to implement those in EBNs and compare according to their priorities. Circuit techniques such as low-swing channels that reduce the power consumption of components outside the VCBs so that VCBs consume a greater fraction of the overall power will favor the use of EBNs. For fairness, this study uses SRAM-based VCBs, the most efficient design. Using VCBs based on latches or FFs would significantly increase area and power, and hence EBN gains.

5 Router Implementation

An EBN router is much simpler than a corresponding VCN router. Table 2 shows a comparison of area, delay,

Table 2. Router implementation comparison.

Aspect	VC router	EB router
# ports	703	683
# nets	16202	5581
# gates	60010	13917
# cells	15943	4143
Area (μm^2)	63515	14730
Cycle time (ns)	3.3	2.7
Dyn. power (mW)	2.59	0.12

and power for 5×5 mesh routers implemented in a 45nm low-power CMOS technology under worst-case conditions. The results were obtained by synthesizing the design using Synopsys Design Compiler and placing and routing the synthesized design using Cadence Silicon Encounter. Power was measured from simulations using uniform random traffic and 24% flit injection rate in each router input, to avoid saturating either router. Realistic router input and output timing constraints, loads and driving strengths were used. The VC router had 2 VCs of 8 buffer slots each. Routing pre-computation [8] was used. Due to library constraints, input buffers were implemented from FF arrays.

Results show a 77% decrease in occupied area, an 18% decrease in cycle time and a 95% decrease in dynamic power. The reduced cycle time enables the network to be clocked at a higher frequency, thus achieving a higher throughput per absolute time, or a lower zero-load latency if the pipeline stages in the VC router are increased.

The cycle time of the EB router is constrained by the output EB occupancy counters and read/write logic. The output EB was implemented as a FIFO due to the high complexity of a three-slot FSM EB. We expect that optimizing the output EB will further decrease the EB cycle time.

The VC router cycle time is constrained by the VC and switch allocators. Increasing the number of VCs increases the complexity of the first stage.

The 95% decrease in dynamic power is due to using FFs to implement the VCBs, instead of SRAM cells.

6 Related Work

An alternative EB channel implementation is described in [15]. Unlike our EB channels, those channels are unpipelined and use repeater cells for storage rather than latches. Using repeater cells for storage adds transistors affecting the channel datapath.

A hybrid VCB–EB scheme was explored in [13]. However, due to the FIFO nature of channels, the interleaving deadlock, presented in section 2.6, can still occur. Preventing it requires transmitting flits without a credit (which may be stored in the EB channel) only if they are non-head, significantly limiting gains from using EB channels. For the same reason, there is no isolation between flits of different VCs. Blocked flits in EB channels block flits from other VCs, even though there may be available buffer space in the router. We have found such hybrid schemes to be beneficial only for very small VCBs. However, such small VCBs represent an inefficient design choice since they can be made larger for a small power cost, but with a significant performance gain — assuming SRAM VCB implementations.

Compressionless routing also does not use VCs [12]. It relies on feedback from the network sent back to network interfaces. Time division circuit-switching flow-control has also been proposed [14]. Connection-based routing can also be used for deadlock-free adaptive routing [20] by allowing intermediate nodes to tear down and modify end-to-end virtual circuits. A hybrid packet-connected circuit has also been proposed, requiring the routers to have buffer space only for request packets [21]. Bufferless networks can avoid dropping packets by emitting packets in a non-ideal direction, also called deflective routing [17].

The vast majority of NoC implementations focus on packet switched networks with a per-input VCB scheme [3]. Such networks enable easy deadlock avoidance, optimized wire utilization, improved performance and QoS [2]. These characteristics along with a reasonable design complexity make VCs the current dominant NoC flow-control scheme and VCNs the comparison metric for this work.

7 Conclusion

This work presented EBNs, an efficient NoC flow-control scheme. EBNs make use of already-existent pipeline FFs in channels for storage, using channels as distributed FIFOs. The power gains from removing the VCBs can be spent on widening the EBN datapath, essentially trading VCBs for increased bandwidth, for networks with a fixed power budget. Duplicate physical channels are used instead of VCs to prevent cyclic dependency and protocol deadlocks. Dividing the network into sub-networks increases performance and power efficiency when normalizing for a fixed performance or power budget.

We compare a number of congestion sensing mechanisms for EBNs, and show that the *output occupancy* mechanism provides the best performance. Using this congestion sensing mechanism, minimal and non-minimal sub-networks, and progressive adaptive routing, we apply UGAL routing to FBFly EBNs.

For UGAL FBFly EBNs, performance and power efficiency is almost equal in the low-swing model, with a 3% loss for EBNs in the full-swing case, compared to VCNs. EBN gains in 2D mesh networks reach 8% for the full-swing channel model, and 12% for the low-swing. On the other hand, due to having a wider datapath, EBNs occupy more area when normalized for power or performance. Zero-load latency is equal between EBNs and VCNs of the same clock frequency.

EB routers are considerably simpler due to the removal of allocators, credits and other overhead, such as VC and packet IDs. This results in an 18% reduced cycle time compared to VC routers for 5×5 mesh routers. This allows EBNs to be clocked at a higher frequency, or reduces the zero-load latency if the VC router pipeline stages are increased to meet that frequency.

For many power-constrained on-chip networks, EBNs substantially increase network efficiency while at the same time simplifying the router design.

Acknowledgments

This work was supported in part by the National Science Foundation under Grant CCF-0702341, in part by the National Security Agency under Contract H98230-08-C-0272, in part by the Semiconductor Research Corporation under Grant 2007-HJ-1591, in part by the Defense Advanced Research Projects Agency under contract FA8650-07-C-7726-P00001 — monitored by the Air Force, in part by the Robert Bosch Stanford Graduate Fellowship, and in part by the Cadence Design Systems Stanford Graduate Fellowship.

References

[1] James Balfour and William J. Dally. Design tradeoffs for tiled CMP on-chip networks. In *ICS '06: Proceedings of the 20th annual International Conference on Supercomputing*, pages 187–198, 2006.

[2] T. Bjerregaard and S. Mahadevan. A survey of research and practices of network-on-chip. *ACM Computing Surveys*, 38(1):1, 2006.

[3] William J. Dally. Virtual-channel flow control. *IEEE Transactions on Parallel and Distributed Systems*, 3(2):194–205, 1992.

[4] William J. Dally and Hiromichi Aoki. Deadlock-free adaptive routing in multicomputer networks using virtual channels. *IEEE Transanctions on Parallel and Distributed Systems*, 4(4):466–475, 1993.

[5] William J. Dally and Brian Towles. Route packets, not wires: On-chip interconnection networks. In *DAC '01: Proceedings of the 38th Conference on Design Automation*, pages 684–689, 2001.

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

[7] Giovanni de Micheli and Luca Benini. Networks on chip: A new paradigm for systems on chip design. In *DATE '02: Proceedings of the conference on Design, Automation and Test in Europe*, page 418, 2002.

[8] Mike Galles. Spider: A high-speed network interconnect. *IEEE Micro*, 17(1):34–39, 1997.

[9] Andreas Hansson, Kees Goossens, and Andrei Rădulescu. Avoiding message-dependent deadlock in network-based systems on chip. *VLSI Design*, May 2007.

[10] Ron Ho, Ken Mai, and Mark Horowitz. Efficient on-chip global interconnects. In *Symposium on VLSI Circuits*, pages 271–274, 2003.

[11] John Kim, William J. Dally, and Dennis Abts. Flattened butterfly: a cost-efficient topology for high-radix networks. In *ISCA '07: Proceedings of the 34th annual International Symposium on Computer Architecture*, pages 126–137, 2007.

[12] Jong H. Kim, Ziqiang Liu, and Andrew A. Chien. Compressionless routing: a framework for adaptive and fault-tolerant routing. *SIGARCH Computer Architecture News*, 22(2):289–300, 1994.

[13] Avinash Kodi, Ashwini Sarathy, and Ahmed Louri. Design of adaptive communication channel buffers for low-power area-efficient network-on-chip architecture. In *ANCS '07: Proceedings of the 3rd ACM/IEEE Symposium on Architecture for Networking and Communications Systems*, pages 47–56, 2007.

[14] Jian Liu, Li-Rong Zheng, and H. Tenhunen. A guaranteed-throughput switch for network-on-chip. In *Proceedings of International Symposium on System-on-Chip*, pages 31–34, 2003.

[15] Masayuki Mizuno, , William J. Dally, and Hideaki Onishi. Elastic interconnects: repeater-inserted long wiring capable of compressing and decompressing data. In *ISSCC '01: Proceedings of IEEE International Solid-State Circuits Conference*, pages 346–347, 464, 2001.

[16] Robert Mullins, Andrew West, and Simon Moore. Low-latency virtual-channel routers for on-chip networks. In *ISCA '04: Proceedings of the 31st annual International Symposium on Computer Architecture*, page 188, 2004.

[17] Erland Nilsson, Mikael Millberg, Johnny Oberg, and Axel Jantsch. Load distribution with the proximity congestion awareness in a network on chip. In *DATE '03: Proceedings of the conference on Design, Automation and Test in Europe*, pages 1126–1127, 2003.

[18] Arjun Singh. *Load-Balanced Routing in Interconnection Networks*. PhD in electrical engineering, Stanford University, 2005.

[19] Vladimir Stojanovic and Vojin G. Oklobdzija. Comparative analysis of master-slave latches and flip-flops for high-performance and low-power systems. *IEEE Journal of Solid-State Circuits*, 34(4):536–548, Apr 1999.

[20] Yoshio Turner and Yuval Tamir. Deadlock-free connection-based adaptive routing with dynamic virtual circuits. *Journal of Parallel and Distributed Computing*, 67(1):13–32, 2007.

[21] Daniel Wiklund and Dake Liu. SoCBUS: Switched network-on-chip for hard real time embedded systems. In *IPDPS '03: Proceedings of the 17th International Symposium on Parallel and Distributed Processing*, page 78.1, 2003.