Asynchronous Bypass Channel Routers

Tushar N. K. Jain, Paul V. Gratz, Alex Sprintson, Gwan Choi

Department of Electrical and Computer Engineering, Texas A&M University
{tnj07,pgratz,spalex,gchoi}@tamu.edu

Abstract—Network-on-Chip (NoC) designs have emerged as a replacement for traditional shared-bus designs for on-chip communications. Typically, these systems require fully balanced clock distribution trees to enable synchronous communication between all nodes on-chip, resulting in higher power consumption. One approach to reduce power consumption is to replace the balanced clock tree with a globally-asynchronous, locally-synchronous (GALS) mesochronous clocking scheme. NoCs implemented with a GALS clocking scheme, however, tend to have high latencies as packets must be synchronized at every hop between source and destination. In this paper, we propose a novel router microarchitecture for GALS NoCs which offers superior performance versus typical synchronizing router designs. Our approach features Asynchronous Bypass Channels (ABCs) at intermediate nodes thus avoiding synchronization delay. We also propose a new network topology that leverages the advantages of the bypass channel offered by our router design. Our experiments show that our design improves the performance of a conventional synchronizing design with similar resources by up to 26% at low loads and increases saturation throughput by up to 11%.

Index Terms—NoC, mesochronous clocking, GALS, asynchronous interconnect, on-chip networks.

1 Introduction

Networks-on-Chip (NoC) leverage the design techniques of macro-scale multi-hop networks for communication between various processing elements (PEs) on chip. NoCs have become a popular solution to interconnect scalability. The relationship between PE clocks is a challenge in NoC design. One approach is to have a completely synchronized architecture. Typically, synchronized architectures require fully balanced clock distribution trees which consume a significant portion of the total chip power [1]. On-chip power consumption can be reduced by replacing the balanced clock tree with a *globally-asynchronous, locally synchronous* (GALS) clocking scheme.

PEs in GALS systems are often referred to as being *mesochronous* to one-another – of the same frequency but unrelated phase. The design of mesochronous synchronizers pose several challenges. One challenge concerns latency for the synchronization required to communicate between mesochronous PEs. This latency directly impacts the communication performance and the overall system cost [2].

With low communication latency as our objective, we propose a new router microarchitecture for mesochronous NoCs which avoids synchronization delay at the intermediate nodes via an *asynchronous by-pass channel* (ABC) around these nodes. We also propose a novel 2-D grid based topology and routing algorithm to complement our router design.

The rest of the paper is organized as follows. Section 2 discusses related work. Section 3 and section 4 describe the router micro-architecture and its detailed operation respectively. Section 5 proposes an efficient topology for interconnected networks. In Section 6, we present the experimental results and compare the performance of our router against a comparable baseline router. Section 7 concludes the paper.

2 Background

A critical aspect of mesochronous system design is the avoidance of metastable state due to a violation of setup or hold time of the flip-flops and registers present in the system. In the metastable state the output of the system is unpredictable. A well established approach for mesochronous NoCs was proposed by Panades and Greiner, in the form of an optimized bi-synchronous FIFO featuring low-latency and small footprint [3]. This FIFO adds a latency of two and three clock cycles to gain robustness against metastability.

Another solution, presented by Dally and Poulton, consists of delay-line synchronizers, using a variable delay on the data lines [4]. This delay avoids switching in the meta-stability window of the receiving registers. This solution requires careful, post-manufacturing tuning to ensure metastability is avoided. Variable delay lines also may make this solution expensive and not always available in standard cell libraries.

Mangano et. al. proposed the *skew insensitive link (SKIL) solution* to address metastability in mesochronous interfaces [5]. SKIL supports arbitrarily skewed clock signals by relying on a two stage buffer structure by writing and reading flits into and from the buffer in a ping-pong fashion.

It should be noted that some of the above synchronization solutions could be used to augment our ABC approach, we plan to explore this in future work. In each of the above techniques, data must be synchronized at each intermediate node. Synchronization delay at every hop forms a major portion of the total latency and can be reduced by either lowering the hop-count or reducing the per-hop delay.

New topologies to lower the hop count have also been explored. Dally proposed express cubes which help in lowering the per-hop latency rather than the hop count [6]. An express cube is a k-ary n-cube, augmented by one or more physical links or express channels that allow nonlocal messages to bypass the intermediate nodes. Kim et al. and Grot et al. proposed topologies with higher radix routers, leading to lower hop counts but resulting in more complex router designs [7, 8].

Express virtual channels (EVCs) have also been proposed to reduce per-hop delay. EVCs virtualize the physical links in an express-cube, limiting wasted physical link bandwidth [9]. EVC-based flow control allows virtual bypassing of the packet at the intermediate nodes thus reducing the per-hop delay. This method is efficient for multi-hop packets but does not fare well for communications between neighboring routers.

The ABC router design targets both the per-hop latency and the hop count to reduce network latency. ABC routers, thus, perform well for both long-haul and short-haul packet communication. In contrast to EVCs and express cubes, ABCs are physical links present in the router offering an *asynchronous bypass* from the buffers. Since ABCs are inherent to the router, they can be dynamically assigned at every hop, unlike EVCs which can be allotted at specific nodes only. A packet takes an ABC whenever possible thereby avoiding FIFOs and synchronization delay, resulting in a lower per-hop latency.

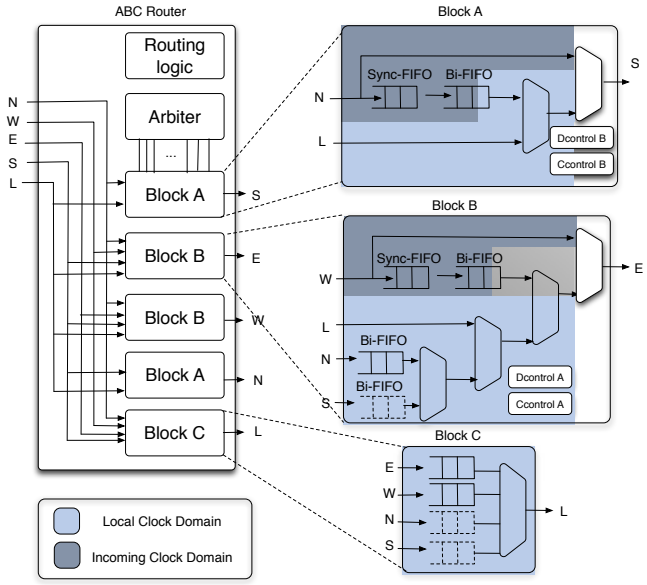


Fig. 1: ABC router microarchitecture design.

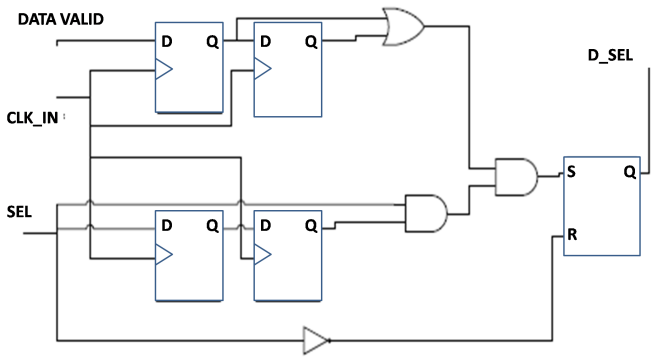


Fig. 2: Block diagram for Dcontrol

3 Router Microarchitecture

We propose an asynchronous router with five input and output ports corresponding to the four neighboring directions and the local processing element (PE). Each port's link is comprised of data bits, a clock bit, and an *on/off* flow control bit. All the data communications through the network are carried out in the form of packets, subdivided into flits, which are sent through the NoC using wormhole flow-control along with a source-synchronous clock signal. As both the data and clock signals travel same link length and router logic, they face the same delay and their skew does not increase. We employ a unicast, deterministic, source routing algorithm. In this section we describe the individual facets of our router in detail.

3.1 Packet Structure

Three most significant bits of each flit contain flit type (Header, Body or Tail) and valid. The header flit also contains routing information. At every hop, the 4th and 5th MSBs of the header flit encode the routing directions for the packet at *that particular node*. After every hop, the current routing bits are shifted away until only a flag is left which marks arrival at the destination.

3.2 Buffers

Although there are no buffers along the ABC path, we employ bi-synchronous FIFOs (bi-FIFOs)[3] to store packets in the event of congestion. In bi-FIFOs, writes are done in the incoming clock domain and reads are done in the outgoing clock domain. Fig. 1 shows the overall block diagram of the router including detail for the output unit blocks. As shown, our router contains five output unit blocks corresponding to each direction. Block B is instantiated for the W and E output directions and has additional buffers for turn paths as compared to Block A which is used for the N and S output directions. This is because the ABC router's routing algorithm disallows E or W turns to N or S outputs. Block C corresponds to the local PE connected to the router. Overall the complete router has 10 bi-FIFOs, two of which are shared between block C and the two block Bs.

ABC routers employ *on/off* flow control. In this scheme, when the number of free buffers in a direction falls below a threshold value the node transmits an *off* signal to the upstream node connected to it through the *on/off* link. Similarly, an *on* signal is transmitted when the number of free buffers exceeds the threshold. This signal needs to be synchronized with respect to the outgoing clock of the upstream node. To ensure no flit gets dropped in the process, the depth of the FIFO must be four more than the maximum packet length.

The path joining the input port with the diametrically opposite output port, for example E to W, is referred to as a straight path. It is comprised of a pair of FIFOs, a bi-FIFO and a synchronous FIFO (s-FIFO), along with the ABC. All other paths are referred to as turn paths as shown in Fig. 1. Each turn path is made up of a single bi-FIFO. The s-FIFOs are standard FIFOs which are clocked in the incoming clock domain. The ABC is a bypass around the straight path FIFOS, as shown in Fig. 1. Under certain conditions described below, the ABC can be taken by the flit to avoid getting latched into the buffers. The s-FIFOs are employed as backup to avoid dropping flits in the event congestion disallows ABC use.

3.3 Router Arbiter

As shown in Fig. 1, the output channel is shared between ABC, the straight path bi-FIFO and turn path bi-FIFOs. We have employed a fixed priority arbiter with the maximum priority to the straight path bi-FIFO followed by ABC, then the turn paths and finally the local buffer.

As shown in Fig. 1, the router arbiter is divided into two types of control modules for each output unit block, the clock control module (*Ccontrol*) and the data control module (*Dcontrol*). The Ccontrol module arbitrates the output clock while the Dcontrol module controls the output data. As the Ccontrol and Dcontrol modules receive inputs from two different clock domains, metastability must be addressed. We avoid all metastable conditions through control logic synchronization. Further detail on metastability avoidance is given in Section 4.

On arrival of a head flit at the output, the router becomes active and remains in this state until a tail flit arrives, when it becomes idle. Arbitration occurs in the idle state, except if an *off* signal is received from the downstream node while the ABC is in use in which case the chosen path is switched to the straight path bi-FIFO.

4 Router Operation

In this section we present a detailed description of the working of the router. In the following sections we describe the logic for switching the clocks and discuss about the metastability issues in our design.

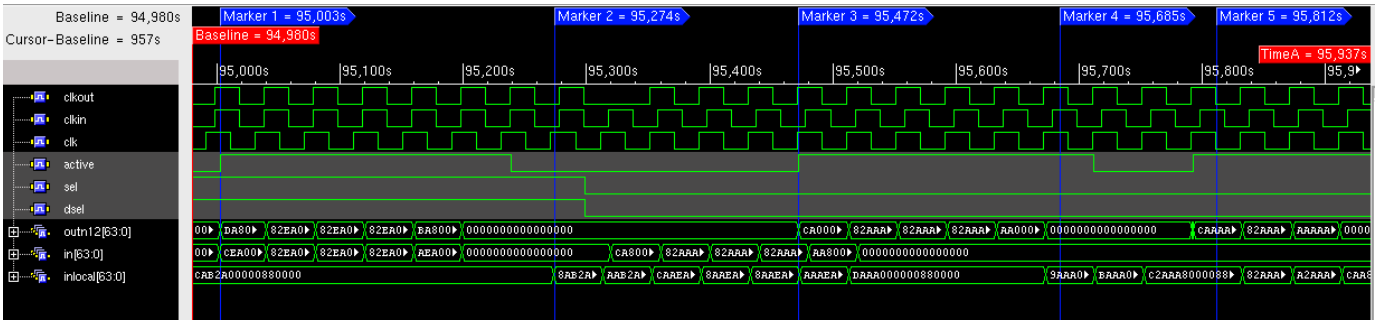


Fig. 3: Waveform at a output port

4.1 Normal Operation

On its path through the network a packet traverses three node classes, the source node, intermediate nodes and the destination node; each node operates in its own clock domain.

At the source node, the packet leaves the local buffer and propagates in the required direction along with the local clock. At each hop, the the 4th and 5th MSBs of the header flit determine whether a packet will travel straight or turn. At the intermediate nodes, if a packet travels straight, the corresponding flits try to utilize the ABC. These flits are simultaneously latched into the s-FIFO. If all the buffers corresponding to the output port are empty, these flits are transmitted to the output port through the ABC and the contents of the s-FIFO are flushed away. If the buffers are not empty, indicating presence of older packets, the contents of the s-FIFO are forwarded to the straight path bi-FIFO and the output port is fed by one of the bi-FIFOs. Thus, the s-FIFO acts as a backup in case the flit was not able to successfully utilize ABC. Alternately, if the packet needs to turn at that particular node, the flits are latched into one of the turn path bi-FIFOs.

If the outgoing flit was read out from any of the FIFOs, the local clock at that particular node is transmitted as the outgoing clock and the output port is said to be in the FIFO mode. However, if the outgoing flit came from the ABC, the incoming clock is forwarded as the outgoing clock and the output port is referred to as in the ABC mode. This ensures that the output data flits are synchronous with the output clock thus avoiding potential metastability conditions downstream.

Each output port can either be in the ABC mode or the FIFO mode at any given time. The trigger signal for transition from one mode to another is generated in the outgoing clock domain. This signal is then synchronized into the desired clock domain by standard, two-cycle synchronization. Once the trigger signal has been synchronized the transition to the new clock begin and only after a successful transition of the clock is the datapath switched, thus avoiding potential metastability conditions in the control logic.

Transition from ABC mode to FIFO mode for a given output port occurs in the following two cases: (i) If an *off* signal arrives from the downstream node; (ii) If there is no packet passing through the ABC and there is a packet in the turn-path.

In both the cases, first the output clock is switched to the local clock. The datapath is then switched to straight path bi-FIFO. If the router was idle before the switch and the bi-FIFO is empty, the datapath is then switched to the turn path. The router then waits for an *on* signal from the downstream node after which the read signal of the selected FIFO is activated. Altogether the transition from ABC mode to FIFO model takes three cycle to complete, however, the added latency is not seen by flits traversing the straight path because they are already latched into the s-FIFO.

The transition from FIFO mode to ABC mode for a given output port occurs when the FIFO mode is currently selected and there are no flits occupying any FIFO associated with this output port. This transition consists of three stages, the first two cost two cycles each while the last stage costs three cycles: (i) In the first stage, the read signals of all the bi-FIFOs are deactivated. If all the turn buffers are empty, the straight path is selected; (ii) In the second stage if the straight path bi-FIFO is empty and the *on* signal is being received, the output clock is switched to the incoming clock; (iii) In the final stage, the datapath is switched to ABC.

However, if a flit arrives on the straight-path in any of the above mentioned stages the transition is aborted and the datapath is switched back to the straight path bi-FIFO at a cost of three additional cycles.

4.2 Clock domain transitions

An important concern while designing the logic for transition between the clock domain was to avoid glitches during switching because presence of glitches can cause unpredictable behavior in the flip flops. The Ccontrol module at a particular output port controls the output clock at that port. Now, if the output mode is functioning in the FIFO mode and the data path needs to be switched from one FIFO to another no switching in the output clock is required as the output of all the bi-FIFOs are in the local clock domain of that node. Clock needs to be switched during transition from ABC mode to FIFO mode and vice versa as these two modes operate in two different clock domains which are characterized by the same frequency but different phases. While the FIFO mode is in synch with the local clock, the ABC mode is in synch with the incoming clock. As mentioned earlier the output clock will either be the local clock or the incoming clock and the selection of the output clock is based on the *select* signal generated by the following trigger inputs:

- The *on/off* signal from the downstream node.
- The active/idle state of the output port.
- The *empty* signal from all the buffers at the current node.

The active/idle state of the output port will always change in the outgoing clock domain. The output port is said to be in idle state if the last flit coming out from it was a tail flit else it is said to be in active state. The *on/off* signal should ideally arrive in the outgoing clock domain as the flits are written to the downstream bi-FIFOs in the outgoing clock domain, however because of the link delay between the nodes the on/off signal needs to be re-synchronized into the outgoing clock domain. The "*empty*" signal is received from the bi-FIFOs of the current node and is set when all the bi-FIFOs are empty. This signal is generated in the local clock domain because reads from the bi-FIFOs are done in this domain. Thus they will be synchronous with the local clock during operation in the FIFO mode else we will need to synchronize them. Now we switch from between the turn path bi-FIFOs or from straight path to turn path or vice versa only in the idle state. We can however switch from ABC to straight path bi-FIFO in case an *off* signal is received. An important point to note is that, without the arrival of an *off*

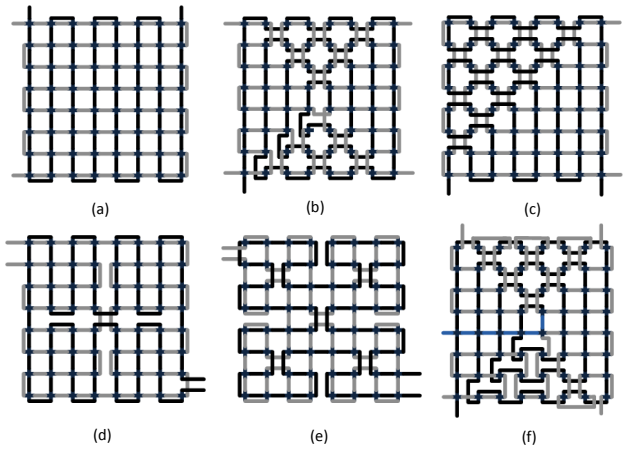


Fig. 4: ABC topologies evaluated.

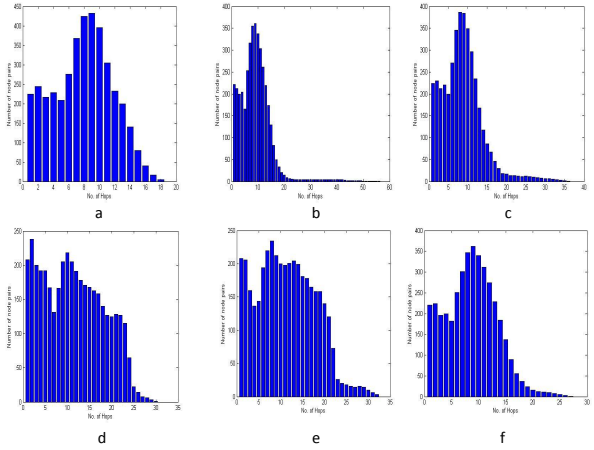


Fig. 5: ABC topologies evaluation.

signal, switching from ABC to FIFO mode can be caused only when a flit has arrived in the turn paths. Now if depending on the above mentioned three signals, a decision is made to switch the clock from the present outgoing clock to the desired outgoing clock, the following procedure is followed.

1) First, the present outgoing clock is switched *off* after the arrival of its negative edge for one complete cycle of the present outgoing clock. Thus, for one complete cycle there is *no* clock at the output port. This gives us three advantages. Firstly, in case a flit arrives at the output port through the ABC it will not be forwarded to the downstream node with a clock signal thus ensuring no repeated flits will be generated during switching. Secondly, in case any glitches are generated they will not be carried forward to the downstream node. Thirdly, because the trigger inputs are all generated in the outgoing clock domain, they will not change during the transition.
2) Next, at the negative edge of the desired outgoing clock the output clock is switched *on* again. This is done by synchronizing the trigger signal in to the desired clock domain also. Due to the synchronization delay of two cycles, the event of switching *on* the output clock follows the event of switching *off* the output clock with a delay of either one or two cycles depending on which clock had a leading phase, the present outgoing clock or the desired one.

4.3 Data path switching

The logic diagram for selecting the data path is shown in fig 2. The Dcontrol module controls the switching of the data path. It has as its inputs, the valid bit of the incoming flit and the *select* signal generated from the Ccontrol module. The Dcontrol module selects the ABC or the bi-FIFOs depending on whether DSEL is set or reset. DSEL is set in the incoming clock domain while reset in the local clock domain. This is because the select signal which is generated in the Ccontrol module is set in the incoming clock domain while reset in the incoming clock domain. The valid flit of the incoming flit is also considered in the logic to ensure that if a flit arrives during transition from FIFO mode to ABC mode, the datapaths are switched back to the FIFO mode.

4.4 Operation

The various stages of an output port of the router are shown in the fig. 3 . In fig. 3,between marker 1 and marker 2, the output port is allocated to the ABC, between marker 2 and marker 3, both the clock switching and the data path switching takes place. Between marker 3 and marker 4, the straight path bi-FIFO is selected. Between marker 4 and marker 5, switching from the straight path bi-FIFO to the turn path bi-FIFO takes place and from marker 5, the turn path bi-FIFOs are read out.

5 Topology and Routing Algorithm

The latency of a bi-FIFO is two cycles thus it takes at least two cycles for a flit to appear at the output of the FIFO after it is written. Suppose a head flit from the north must turn west and suppose that all paths are free. Then, after two cycles the flit will appear at the output of the turn-path FIFO and at the next cycle the arbiter will select the respective FIFO. Thus, there is a penalty of three cycles for a packet to make a turn. If, however, the packet travels on a straight path taking ABCs, it will only face the link delays. Thus, the router should preform best with a topology in which maximum number of nodes are connected through a straight path. As a 2-D topology makes a good fit for the 2-D planar silicon technology, we have explored several 2-D, grid-based topologies in an attempt to minimize the number of turns and thereby the latency associated with the ABC router's turn path.

We consider six different grid based topologies. These six, however, do not represent all possible topologies, we plan to investigate an optimal ABC topology further in future work. Fig. 4 depicts these six topologies on an 8x8 2-D grid layout. Each topology consists of two continuous chains connecting all nodes to maximize the number of straight path options.

To complement the router architecture we employ a modified version of standard XY dimension order routing (DOR). Our algorithm attempts to optimize packet route based upon assumptions of link and turn delays. Suppose that each link between two adjacent nodes causes a delay of x clock cycles. A turn costs 3 cycles in addition to the link delay. If a flit turns it faces an additional delay of $(3/x{+}1)$ times the link delay. Thus, if the number of hops is less than or equal to $(3/x + 1)$ hops, traveling on ABC is a better choice then making a turn. When traveling along the straight path requires more hops than this value, turning leads to less delay in terms of clock cycles. Our routing algorithm selects the path between a source destination pair with the least delay based upon this estimation.

For our simulations we have arbitrarily assumed the link and router delay to be 75% of a clock cycle, yielding a turn cost of $(3/.75 + 1) = 5$. Thus, for the given topologies, we remain on the same chain if the number of hops is less than or equal to five else we make a turn. To avoid deadlock, the chains are not

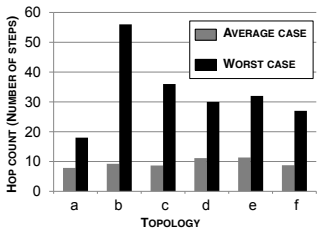
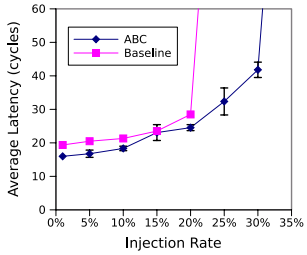
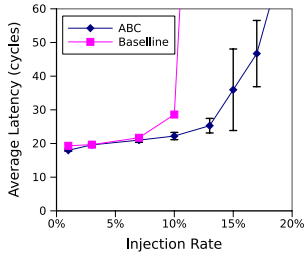


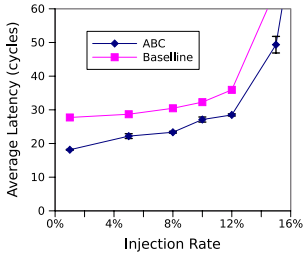
Fig. 6: Average and worst case hop count for the six topologies.



(a) uniform random



(b) transpose



(c) bit-complement

Fig. 7: Average latency vs injection rate

connected end-to-end to form a ring and we only allow turns from the black chain to grey but not vice versa.

5.1 Analysis

We have evaluated the topologies shown in Fig. 4, for minimal delay and we have arrived at an *"optimal"* topology among the above mentioned six. From the evaluation results shown in Fig. 6, topology (a) performs best with respect to both the average as well as the worst case hop count. This is because it is the only topology among the six, in which each node is connected to all its neighbors through both the chains resulting in lower cost paths in comparison to the other topologies for all source and destination pairs.

6 Experiments and Evaluation

In this section, we evaluate ABC routers experimentally to gain an understanding of their performance under different types of synthetic workloads. We also compare the performance against a baseline, synchronizing router design.

6.1 Methodology

In these experiments, we evaluate a network of ABC routers connected in the 8x8 2-D topology depicted in Fig. 4(a) and implemented using the routing algorithm described in Section 5. A fully synthesizable Verilog implementation of this network was simulated to capture all ABC router network performance results. The baseline design consists of an 8x8, 2-D mesh network with XY DOR routing. In the baseline design packets are synchronized at every hop, incurring a delay of three clock cycles per hop. The baseline router has two, eight-flit-deep, virtual channels (VCs) per port, yielding approximately the same area overhead as the ABC router.

We evaluated three types of synthetic workloads: uniform random, bit complement, and transpose. In all cases packets were injected according to a uniform random process. Five experimental runs were completed for each workload and the mean and standard deviation of the results are shown. Packet length was varied randomly between two to five flits and the simulator was run for 1000 cycles of warm-up followed by 5000 monitored packets.

6.2 Discussion

Fig. 7 compares the latency performance of ABC router with the baseline for uniform random, transpose, and bit-complement workloads. In uniform random traffic, each source is equally likely to send a packet to each destination. In transpose traffic, node (x, y) sends packets to node (y, x). In bit-complement traffic, node (x, y) exchanges packets with node $(\tilde{x}, \tilde{y})$ where, $\tilde{x}$ is the one's complement of x. Among the three traffic patterns, random traffic uniformly balances load even for topologies and routing algorithms that normally have poor load balance. The other two patterns concentrate on individual source-destination pairs, thus stress the load balance of a topology and routing algorithm [10].

For uniform random traffic, shown in Figure 7(a), ABCs offer 20% improvement in no-load latency and the traffic percentage at which saturation occurs is 10 percentage points more then the baseline design. An improvement in the saturation throughput occurs due to our topology's increase bisectional bandwidth as compared to a standard mesh network.

Transpose traffic, in Figure 7(b), shows little improvement in the no-load latency. This is because, for transpose traffic, packets must always take a turn due to the arrangement of source-destination pairing. Saturation throughput is still improved, however, as compared to the baseline design.

For bit complement traffic, shown in Figure 7(c), ABCs offer an improvement of 26% in no-load latency as well as 4 percentage point improvement in saturation throughput. In bit complement traffic, packets travel further than the other two traffics; therefore, ABC's benefits compound resulting in much better performance than baseline.

Generally, ABCs outperform baseline for all the workloads. An important feature common for all the three loads is the shape of the curve. In all the three graphs the shape of the curve for the ABC design is uneven, unlike the baseline. For example, for random traffic there is a distinct bump in latency at 8% traffic. This the point at which congestion causes the routers to switch back and forth between FIFO and ABC modes. These transitions lead to an increase in the relative rate of increase of latency until the load becomes great enough that the routers get locked in FIFO mode.

7 Conclusions and Future Work

In this paper we present the design, implementation, and evaluation of ABC routers. ABC routers aim to achieve lower latencies via bypassing synchronization in the intermediate nodes. We also evaluate several topologies specifically designed for this network. We present a detailed microarchitecture for an ABC based router, a new topology and a novel routing algorithm to complement the router design. Our experiments show that our ABC based network has significantly lower latencies compared to baseline.

This paper presents a basic framework for ABC based networks. Our future work will focus on improving the performance of the network through lower latency synchronizing techniques in the turn path. We also plan to investigate optimal topologies for ABC enhanced routers. The presented network offers the benefit of path diversity which could not be exploited by the static routing algorithm thus necessitates employing a dynamic algorithm. Furthermore, the network also needs to be evaluated for realistic workloads.

References

[1] T. Meincke, A. Hemani, S. Kumar, P. Ellervee, J. Oberg, T. Olsson, P. Nilsson, D. Lindqvist, and H. Tenhunen, "Globally

asynchronous locally synchronous architecture for large high-performance ASICs," in *The IEEE International Symposium on Circuits and Systems (ISCAS)*, vol. 2, pp. 512–515, 1999.

[2] P. Gratz, C. Kim, R. Mcdonald, S. Keckler, and D. Burger, "Implementation and Evaluation of On-Chip Network Architectures," in *International Conference on Computer Design*, pp. 477–484, 2006.

[3] I. Panades and A. Greiner, "Bi-Synchronous FIFO for Synchronous Circuit Communication Well Suited for Network-on-Chip in GALS Architectures," in *The First International Symposium on Networks-on-Chip (NOCS)*, pp. 83–94, 2007.

[4] W. Dally and J. Poulton, *Digital systems engineering*. Cambridge Univ Pr, 1998.

[5] D. Mangano, R. Locatelli, A. Scandurra, C. Pistritto, M. Coppola, L. Fanucci, F. Vitullo, and D. Zandri, "Skew insensitive physical links for network on chip," in *The 1st International Conference on Nano-Networks and Workshops (NanoNet)*, pp. 1–5, 2006.

[6] W. Dally, "Express cubes: Improving the performance of k-ary n-cube interconnection networks," *IEEE Transactions on Computers*, vol. 40, no. 9, pp. 1016–1023, 1991.

[7] J. Kim, W. Dally, B. Towles, and A. Gupta, "Microarchitecture of a high-radix router," *ACM SIGARCH Computer Architecture News*, vol. 33, no. 2, pp. 420–431, 2005.

[8] B. Grot, J. Hestness, S. Keckler, and O. Mutlu, "Express Cube Topologies for on-Chip Interconnects," in *The 15th International Symposium on High Performance Computer Architecture (HPCA)*, pp. 163–174, Feb. 2009.

[9] A. Kumar, L. Peh, P. Kundu, and N. Jha, "Express virtual channels: towards the ideal interconnection fabric," in *The 34th annual International Symposium on Computer Architecture (ISCA)*, pp. 150–161, 2007.

[10] W. Dally and B. Towles, *Principles and practices of interconnection networks*. Morgan Kaufmann, 2004.