

# Asynchronous Bypass Channels for Multi-Synchronous NoCs: A Router Microarchitecture, Topology, and Routing Algorithm

Tushar N. K. Jain, Mukund Ramakrishna, Paul V. Gratz, *Member, IEEE*,  
Alex Sprintson, *Member, IEEE*, and Gwan Choi

**Abstract**—Network-on-chip (NoC) designs have emerged as a replacement for traditional shared-bus designs for on-chip communication. As with all current very large scale integration designs, however, reducing power consumption in NoCs is a critical challenge. One approach to reduce power consumption is to dynamically scale the voltage and frequency of each network node or groups of nodes (DVFS). Another approach is to replace the balanced clock tree with a globally-asynchronous, locally-synchronous (GALS) clocking scheme. In both DVFS and GALS designs, the chip as a whole is multi-synchronous. As the NoCs interconnecting those nodes must communicate across these clock domain boundaries, they tend to have high latencies as packets must be synchronized at the intermediate nodes. In this paper, we propose a novel router microarchitecture which offers superior performance with respect to typical synchronizing router designs for multi-synchronous networks. Our approach features asynchronous bypass channels which allow flit traversal of intermediate nodes within the network without the latching or synchronization overheads of typical designs. We also propose a new network topology and routing algorithm that leverage the advantages of the bypass channel offered by our router design. We present a detailed analysis of design decisions which affect the performance of the asynchronous bypass channel network. 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 50% for a uniform random traffic.

**Index Terms**—Asynchronous interconnect, DVFS, GALS, NoC, on-chip networks.

## I. INTRODUCTION

WITH THE CONTINUED advance of Moore's law, ever increasing transistor densities are yielding ever greater numbers of processing elements (PEs) on chip, resulting in the current proliferation of chip-multiprocessor (CMP) and system-on-chip (SoC) designs [1], [2]. As the number of PEs increases, the communication between those components is

becoming ever more critical. Networks-on-chip (NoCs) have recently emerged as a scalable alternative to the bus-based and ad-hoc interconnect seen in past designs. NoCs leverage the design techniques of multi-hop interconnection networks developed for the large scale multiprocessor computers of the 1990s, to provide low-latency, scalable communication between PEs on chip.

Although Moore's law continues to provide greater transistor densities, current generation very large scale integration (VLSI) presents several challenges going forward. In particular, balancing performance against the energy and power consumption in a given design is a challenge. A popular method to achieve this balance is dynamic voltage and frequency scaling (DVFS) [3]. In CMP and SoC designs which support DVFS, the voltage and frequency of individual PE nodes, or sets of PE nodes are dynamically managed to reduce overall power and energy consumption while maintaining enough performance to meet application demands [4].

Another common approach to reduce the power and energy consumption in CMPs and SoCs is the use of a globally-asynchronous, locally synchronous (GALS) clocking scheme [5]. In traditional VLSI design, a single synchronized clock is maintained throughout the entire chip. These synchronized architectures require fully balanced clock distribution trees to ensure minimal clock skew between communicating components. Fully balanced clock distribution trees, however, consume a significant portion of the total chip power, which can be as high as 30% of the total power consumed by the chip [5]. On-chip power consumption can be reduced by replacing the balanced clock tree with a GALS clocking scheme which only guarantees minimal clock skew within the local processing element.

PEs in both DVFS and GALS systems are implemented with each PE operating in its own respective clock domain, each having a different phase and/or frequency. Systems in which each PE has its own clock domain are known as multi-synchronous systems. Communication between nodes in a multi-synchronous system typically requires synchronization into each clock domain traversed in order to avoid metastability problems caused by violations of setup or hold time of the flip-flops when latching data from a different clock domain [6]. A well-established approach to avoiding metastability involves adding multiple stages of latching on

Manuscript received July 2, 2010; revised January 20, 2011; accepted May 21, 2011. Date of current version October 19, 2011. This paper was recommended by Associate Editor L. P. Carloni.

T. N. K. Jain is with AMD, Austin, TX 78735 USA (e-mail: tnj07@tamu.edu).

M. Ramakrishna, P. V. Gratz, A. Sprintson, and G. Choi are with the Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX 77843 USA (e-mail: mukund.r@tamu.edu; pgratz@tamu.edu; spalex@tamu.edu; gchoi@tamu.edu).

Color versions of one or more of the figures in this paper are available online at <http://ieeexplore.ieee.org>.

Digital Object Identifier 10.1109/TCAD.2011.2161190

incoming signals to reduce the likelihood of a metastable event. While this approach allows reliable communication across clock domains, it comes at the cost of an added two or three cycles of latency for every clock domain traversal.

Another well-known technique to avoid metastability is the use of link-level asynchronous protocols to coordinate link traversal between clock domains [7]. While this approach does not require bi-synchronous first in, first out (FIFOs), it does require each link traversal be coordinated via a handshaking protocol prior to flit launch, at the potential cost of one or two cycles of pre-traversal latency. As the number of CMP and SoC PE nodes scales up, the number of clock domains which data must traverse between source and destination will increase, thereby increasing the effective latency of that communication. This latency has been shown to directly impact the system performance and its overall cost [8].

With low communication latency as our objective, we propose a new architecture for multi-synchronous NoCs which avoids synchronization delay at the intermediate nodes. The main contributions of this paper are as follows.

- 1) A novel router microarchitecture for multi-synchronous NoCs. This router provides a low latency, asynchronous bypass channel (ABC) through the intermediate routers, which allows us to avoid synchronization delay and latching at intermediate nodes and thus having a direct impact on the network latency.
- 2) A new class of interconnection network topologies, the double-chain NoC topology. Double-chain topologies are comprised of two disjoint but overlapping chains, each of which connects all network nodes. These topologies are well suited to both 2-D planar VLSI technology and the ABC router microarchitecture.
- 3) An optimized routing algorithm which leverages the ABC router microarchitecture and the greater path diversity found in double chain topologies to lower packet latencies.

The rest of this paper is organized as follows. Section II presents the motivation for ABCs. Section III gives a detailed description of the network. Section IV proposes an efficient topology for the interconnected networks. In Section V, we present the experimental results and compare the performance of our router against a comparable baseline router. Section VI discusses the related work and Section VII concludes this paper.

## II. MOTIVATION

The intuition behind the ABC design is that a packet's data should not be required to latch at the intermediate nodes within an NoC. At a minimum, intermediate node latching results in one cycle of overhead for synchronous designs and at least two cycles of overhead for multi-synchronous designs which require synchronization [9], [10]. If intermediate node latching and synchronization can be avoided, a significant fraction of the packet latency overhead can be removed. We ensure that when possible, the data does not traverse clock domains and rather stays in its own clock domain as it traverses the switch. To this end, our state machine waits for the incoming

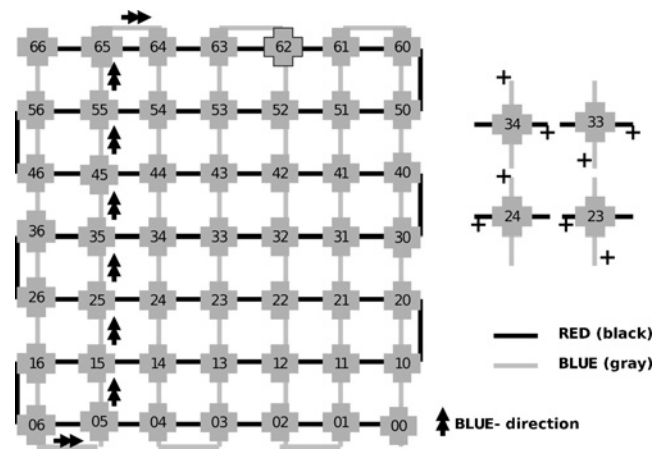


Fig. 1. ABC network topology showing red and blue chains. Also depicts path traversed between (0, 6) and (6, 4).

signals to become stable before switching between the clock domains.

In the absence of network congestion, a packet should not, in fact, require latching at intermediate nodes. At high congestion, when multiple packets are vying for the same buffer FIFO and output port resources, the packet must be latched at the intermediate nodes as it waits for the contention to clear. At low congestion levels, however there is no such constraint, as the output multiplexer may be pre-set to allow the asynchronous propagation of data through the router without latching.

As we will show, the ABC design offers an asynchronous bypass path which bypasses the buffer FIFOs as well as synchronization at the intermediate nodes. Thus, the packets can virtually fly through the network without getting latched at the intermediate nodes, eschewing both synchronization and asynchronous protocol handshake latency and reaching their destination with a delay approaching the wire delay of the connecting links.

## III. NETWORK OPERATION

On its path through the network a packet traverses three node classes, the source node, intermediate nodes, and the destination node. In a typical GALS or DVFS NoC, each of these nodes may operate in its own clock domain. The goal of the ABC router design is to reduce or remove latching and synchronization at the intermediate nodes. Furthermore, the routers should independently and dynamically determine the availability of the bypass channels without the overheads of allocation, setup, and tear-down. In this section, we discuss the design and operation of the ABC router to show how these goals were met.

### A. ABC Network Architecture

As illustrated in the packet propagation path shown in Fig. 1, we observe that packets traversing a given intermediate node router are most likely to travel straight, that is from input port on one side of the router to output port on the opposite side. In a typical 2-D Mesh network, a minimum length path

between any two nodes may be found which has at most one turn. We propose to leverage this observation by biasing our ABC routers toward the straight path over the turn path. ABC routers are pre-disposed to allow the asynchronous propagation of packets along the straight path, as long as conditions allow.

In 2-D Mesh networks, only a limited set of source-destination pairs require no turn during packet traversal, those pairs which are in the same column or row. We observe that it should be possible to gain a further benefit from the ABC router’s straight path bias if the network topology provides a greater number source-destination pairs which do not require a turn. To this end, we propose a new class of network topologies, the double-chain topology, an example of which is shown in Fig. 1. A double-chain topology is comprised of two disjoint but overlapping chains, each of which connects all network nodes. Double-chain topologies provide an advantage over 2-D Mesh networks for ABC routers by providing two paths comprised solely of “straight-path” links between all source-destination pairs. Double-chain topologies also offer higher amount of path diversity as compared to a standard 2-D Mesh. In contrast to a 2-D Mesh, where all source and destination pairs have only two deadlock-free paths between them, double-chain topologies offer four such paths.

As the traditional 2-D Mesh terminology of *North*, *South*, *East*, and *West* is less useful in a double-chain topology, we will label these chains the *red* chain and a *blue* chain, respectively (note that for the purpose of black and white printing, the red chain is shown in black, while the blue chain is shown in gray). As shown in Fig. 1, nodes within each chain are perceived to be ordered such that traversing a given chain in one direction is considered to be increasing (e.g., the *red+* direction) and traversing that chain in the opposite direction is considered to be decreasing (e.g., the *red-* direction). Jumping from one chain to other is referred to as a *turn*. To avoid deadlock due to cyclic dependences in the network, the chains themselves are ordered, and a jump is allowed only from the *blue* chain to the *red* chain and not vice versa.

**B. Router Microarchitecture**

Fig. 2 shows a high-level block diagram of the proposed ABC router. Similar to a typical 2-D Mesh NoC router, the ABC router contains five input and output ports corresponding to the four neighboring directions and the local PE. 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. The ABC router design is source-synchronous, hence each port’s link is comprised of a clock bit along with data bits and a *credit* flow control bit. Both the data and clock signals travel same link length and router logic, therefore they face nearly the same delay and their skew does not increase much for each hop traversed. Nevertheless, after place and route, if an unexpected skew is observed between the clock signal and the flits, transparent latches or buffers may be inserted to reduce it, at the cost of some added latency.

1) *Packet Structure*: ABC packets are deterministically source routed, i.e., they have a full set of routing directions encoded into the header, as shown in Fig. 3(b). Source routing

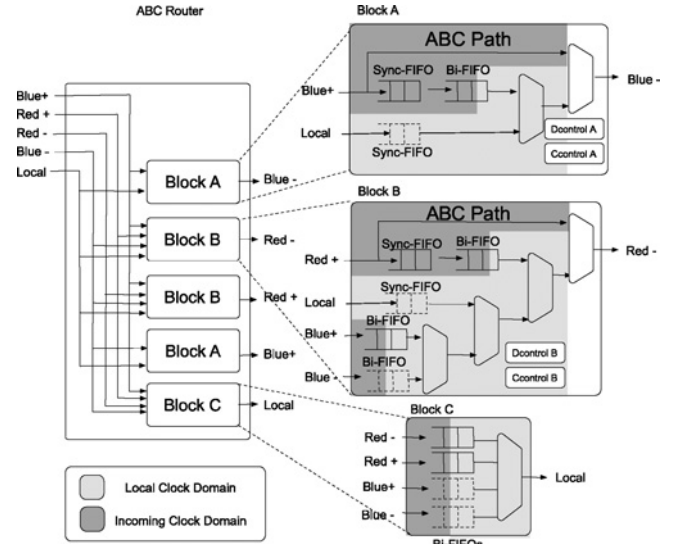


Fig. 2. ABC router microarchitecture design.

is used to ensure a minimal delay and skew induced at each hop. As the figure shows, the three most significant bits of each flit contain flit type (Header, Body, or Tail) and a valid bit. The fourth and fifth most significant bits of the header flit encode the routing directions for the packet at that particular node. After each hop, the current routing bits are shifted away until only a flag is left which marks arrival at the destination as depicted in Fig. 3. Since the static shift operation can be achieved without any logic, we are able to retrieve the next hop information at the node without impacting the skew between the flits and its clock signal.

2) *Router Datapath*: As shown in Fig. 2, the ABC router contains five output unit blocks corresponding to each direction. These output unit blocks are broken into three types. Output block “A” connects the *blue* (+ or -) and *local* inputs to the opposite *blue* chain output, while output block “B” connects all the possible inputs to the corresponding *red* output. Block “C” connects all possible inputs to the router’s local PE. Block “A” has a smaller set of inputs because it is illegal to route from a *red* input to a *blue* output as discussed in Section III-A.

At the intermediate nodes, if a packet travels straight through the router, it is desirable that the packet be forwarded without latching. To this end, output blocks “A” and “B” both contain an “ABC path” which the router is pre-disposed to use in the absence of congestion. When an incoming flit arrives, it is simultaneously latched into the synchronous FIFO (*sync-FIFO*) and propagated along the ABC path toward the output multiplexer. In the absence of congestion, indicated by a lack of packets currently waiting for the output port and the presence of favorable downstream control flow, the flit is propagated through the output port multiplexer via the ABC and the contents of the sync-FIFO are flushed away. Alternately, if the buffer FIFOs are not empty, indicating presence of older packets, the contents of the sync-FIFO are forwarded to the straight path bi-synchronous FIFOs (*bi-FIFOs*) [11] and the output port is fed by one of the bi-FIFOs. Thus, the sync-FIFO acts as a backup in case the flit was not

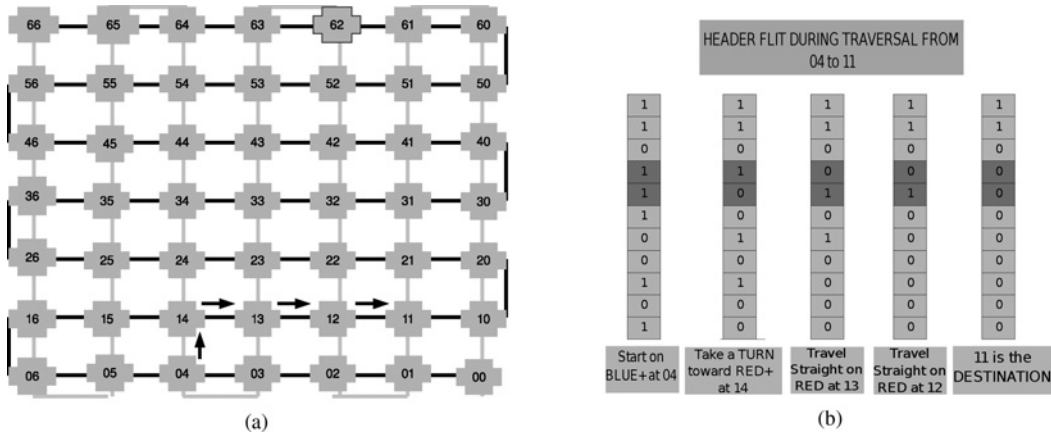


Fig. 3. Routing algorithm. (a) Path of a packet generated at (0, 4) for (1, 1). (b) Most significant bits of the header flit for the packet shown in Fig. 3(a) flowing through the network.

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.

Although there are no buffer FIFOs along the ABC path, we employ bi-synchronous FIFOs (bi-FIFOs) to store packets in the event of congestion. Bi-FIFOs allow writes and reads to be done in different clock domains, although they require an overhead of two to three clock cycles to avoid metastability. In the ABC routers, bi-FIFO writes are done in the incoming clock domain and reads are done in the local clock domain. Overall the complete router has five sync-FIFOs, one of which is interfaced between the local output port and all routing blocks, and ten bi-FIFOs, two of which are shared between the “C” block and the two “B” blocks (the shared FIFOs are shown in dotted lines in Fig. 2).

The path joining a given input port with the same color output port (e.g., *red-* to *red+*), is referred to as a “straight-path.” All other paths are referred to as turn paths as shown in Fig. 2. Each turn path is made up of a single bi-FIFO. The sync-FIFOs are standard FIFOs which are clocked in the incoming clock domain. In the current ABC router design, the ABC is a bypass only around the straight path FIFOs, as shown in Fig. 2. While it is also possible to implement ABC paths for the turns, the number of FIFOs required would increase significantly due to our use of output buffering, required to enable per-output unit clock independence. Therefore, we will restrict the discussion in this paper to the implementation of ABC channels to only the straight paths through the router and leave router designs with more ABC paths to future work.

3) *Flow Control*: ABC routers employ credit based flow control. In this scheme, the upstream node keeps a counter for the number of flits available in each relevant FIFO in the downstream node. Thus, there are counters at the output ports associated with the blue output port of the upstream node, corresponding to FIFOs at one of the blue outputs and the FIFOs at both the red outputs of the downstream node. For every flit output by the upstream node, the counter for the relevant FIFO is decremented by one. Whenever a flit leaves the downstream node, a credit signal is sent back to the upstream node from the downstream node, to increment

the associated counter. These credits are maintained between adjacent nodes just as in traditional interconnection network credit based flow-control. If a flit uses the ABC path, the downstream node immediately returns the credit to the upstream node. This credit signal is generated in the outgoing clock domain of the downstream node and suffers the link delay before reaching the upstream node and hence, needs to be synchronized into the outgoing clock domain of the upstream node before the counter may be updated to avoid metastability. This is done using the standard two flip-flop synchronizer, which takes two cycles for generating the output. The use of a source synchronous clock along with credit based flow decouples synchronization from the flow control and exempts us from using an asynchronous handshaking protocol for the credits. The upstream node may transmit flits only when the counter for the relevant FIFO is non-zero. Since the counters operate in the outgoing clock domain of the upstream node, no over-commitment of buffers, as in on-off flow control, is required.

Since the chains are ordered, a packet traveling on the blue chain is allowed to jump onto the red chain but not vice versa, the output ports corresponding to blue chain have three counters while the output ports corresponding to the red chain have only one counter to keep an account of the available credits at the downstream node.

4) *Router Arbiter*: As shown in Fig. 2, the output channel is shared between ABC, the straight path bi-FIFO, the local FIFO, and turn path bi-FIFOs. To avoid livelock and ensure routing fairness the arbiter selects the ABC path in the ABC mode, and arbitrates among the bi-FIFOs according to the Round Robin algorithm. However, on transition from ABC mode to FIFO mode, the straight path bi-FIFO is given a priority.

As shown in Fig. 2, 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 by selecting amongst the local clock and the incoming clock while the *Dcontrol* module selects the channel which provides the output data. As the *Ccontrol* and *Dcontrol* modules receive

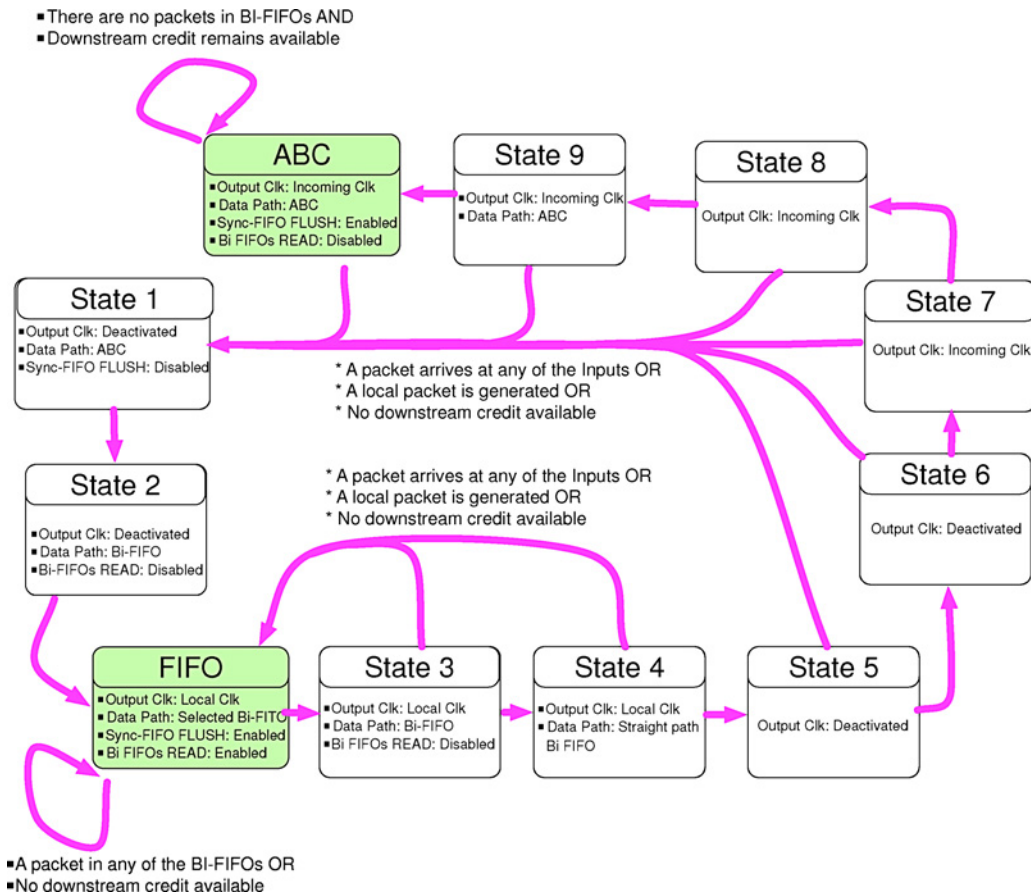


Fig. 4. Finite state machine (FSM) depicting the states of the ABC router’s output port control logic.

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 III-B5. When the router switches from the FIFO mode to the ABC mode, the output clock is changed from the local clock to the incoming clock using two-stage synchronization. Once this is set up, all subsequent flits can pass through without latching. As the arbitration modules for each port are independent of each other, the router can simultaneously transmit on both the red and blue paths. Both these transmissions will be synchronous to the output clock selected by the corresponding port’s clock arbitration module.

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.

5) *Control Logic*: The control logic in an ABC router’s output port can be represented as a Moore finite state machine (FSM). The states of this FSM are displayed in Fig. 4.

Each ABC router output port operates in one of two modes.

**ABC**: When the output port is in ABC mode flits are forwarded along with the upstream or the incoming clock. In this mode flits are not latched as they traverse the router; only combinatorial logic and wire delays are seen by data as it traverses the router.

**FIFO**: When the output port is in FIFO mode incoming packets are synchronized into the local clock domain and forwarded along with the local clock. In this mode, flits experience synchronization delays as they traverse the router.

While switching between clocks, the router ensures that the successive output clock edges are not too close to each other. Whenever the clock needs to be switched, the output clock is deactivated—held high, and then is only reactivated when more than one full new cycle has passed. The forwarding of the clock with the data flits ensures that the output data flits are synchronous with the output clock thus avoiding potential metastability conditions downstream. Note that we do synchronize eventually, but we synchronize less often than traditional GALS NoC router designs. The ABC state and the FIFO state are two stable states for this FSM. The other states are transient, as the state machine can stay in any of these states for only one cycle of the local clock. These states are used to produce a synchronous and safe transition between the two stable states. In effect, they could be called the states of control logic synchronization. The transition from one mode to another is controlled only by the trigger signal. Therefore, in the middle of a multi-flit packet, the router might switch modes, however no flits will be dropped due to the sync-FIFO’s operation as we describe below.

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. The same synchronization method is followed for all the other control signals which are not in the local clock domain. Once the trigger signal has been synchronized the transition to the new clock begins and only after a successful transition of the clock is the datapath switched, thus avoiding potential metastability conditions in the control logic.

a) *ABC mode to FIFO mode*: The transition from ABC mode to FIFO mode for a given output port occurs in the following two cases.

- i) If a local packet has been generated or there is a packet in the turn-path. If a local packet is generated at the node, it is stored into the bi-FIFOs of block C (Fig. 2). If a packet needs to turn, it is latched into the relevant turn-path bi-FIFOs. In both the cases, the packets can be served by an output port only in the FIFO mode.
- ii) If there is no available credit associated with the relevant bi-FIFO at the downstream node. For example, if the available credit at the *blue+* output port of the local node, for the bi-FIFO corresponding to the *blue-* input port at the downstream node drops down to zero, and a packet was destined to go straight, the control logic corresponding to the *blue+* port would trigger a switch from the ABC mode to the FIFO mode.

As presented in Fig. 4, in both the cases, first the output clock is deactivated and switched to the local clock (State 1). The datapath is then switched to one of the bi-FIFOs depending on which has a packet in it (State 2). The straight path bi-FIFO is always given preference just after the transition. In FIFO State the router begins transmission of the packets present in the bi-FIFO. If the router was idle before the switch and the straight-path bi-FIFO is empty, the datapath is switched to the turn path. This ensures that all the flits of the packet pass through the router before a new packet is transmitted. If more than one FIFO contains flits while in FIFO mode, round robin algorithm is used to select among them.

The router then transmits the flits read from the selected bi-FIFO while there remains available credit in the relevant bi-FIFO at the downstream node. On successful departure of flits from the bi-FIFO at the downstream node, available credits at the upstream node are restored and transmission is restarted.

Altogether the transition from ABC mode to FIFO mode takes three cycle to complete. This added latency is not seen by flits traversing the straight path because they are already latched into the sync-FIFO.

b) *FIFO mode to ABC mode*: 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.

These transitions can be divided into four stages corresponding to State 3 to State 9 in Fig. 4.

- i) In the first stage, the read signals of all the bi-FIFOs are deactivated. This makes sure that no packet is read out during the clock transition (State 3).
- ii) If all the turn buffer FIFOs are empty, the straight path is then selected as the output datapath, (State 4).

- iii) In the third stage if the straight path bi-FIFO is empty and *credits* corresponding to the relevant bi-FIFO at the downstream node are available, the output clock is switched to the incoming clock, (State 5 to State 8). It takes at least two cycles for the arrival of a packet at the input of the bi-FIFOs to translate in the reading clock domain, so to ensure that no packet was latched into the bi-FIFO while the clock transition was taking place, this stage takes four cycles to complete.
- iv) In the final stage, the datapath is switched to ABC State 9.

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.

#### IV. TOPOLOGY AND ROUTING ALGORITHM

As discussed in Section III-A, we propose a new class of topologies, the double-chain topologies, to leverage the advantages of our ABC router design. In particular, double-chain topologies take advantage of the lower latency of the straight-path over the turn-path in an ABC router. A double-chain topology is comprised of two disjoint but overlapping chains, each of which connects all network nodes. As the possible space of all possible double-chain topologies is large, in this section we discuss the selection of a double-chain topology and routing algorithm suitable for the 2-D planar silicon substrate of current NoCs.

##### A. Topology Characteristics

To limit the state space of topologies to only those that might be easily implemented in 2-D planar silicon, we began with the following restrictions.

- 1) The topology should not require more router ports than a typical 2-D Mesh would require (five ports in total). Adhering to this restriction helps ensure that the design complexity and area of the ABC router remains similar to that of a 2-D Mesh router.
- 2) Each topology must consist of only two chains, each of which must pass through every network node. Trivial analysis shows that it is not possible to connect all nodes with more than two chains without increasing the number of router ports or subdividing the network into non-overlapping chains. Subdividing the network into more than two non-overlapping chains always performs worse than connecting the non-overlapping chains together.
- 3) Each chain may only connect near neighbor nodes together in one hop. This restriction ensures that there are no “fly-over” links, links which pass through but do not connect to a given node. Fly-over links reduce the maximal wire density without adding to a node’s I/O bandwidth.
- 4) The overall maximum bisection bandwidth of the topology must not be significantly more than that of a standard mesh topology. Strictly speaking, connecting all network nodes together in two chains will always require

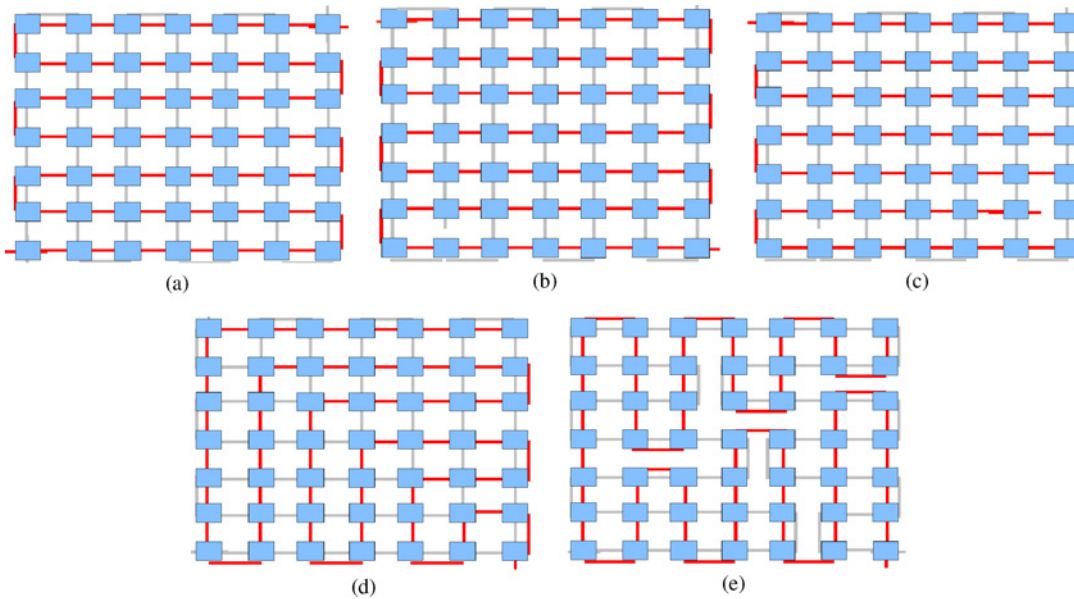


Fig. 5. 2-D topologies for ABC router network. (a) Serpentine. (b) Hook. (c) Double hook. (d) Sickle. (e) “H” shape.

at least one more bisection link than a comparable 2-D Mesh topology. We will only examine topologies which increase bisection wire density minimally and in our evaluation we will ensure packet flit counts are adjusted to reflect the slightly reduced available wire density.

Note that for the purposes of this discussion we will examine chains rather than rings, to simplify deadlock avoidance.

The intent of double-chain topologies is to increase the number of source-destination pairs which are connected by straight-paths, thereby reducing the average packet latency. In order to evaluate the relative merits of individual topologies we must understand the relative latencies of the straight-path versus the turn-path.

### B. Router Traversal Latency

As shown in Fig. 2, when not experiencing congestion, packets traversing an ABC router may take either a straight-path ABC channel or be latched into a turn-path bi-FIFO. The latency of bi-FIFO is two cycles. Thus, combined with the latency of the control logic, there is a penalty of three cycles for a packet to make a turn.

Alternately packets which traverse the router on a straight-path, utilizing the ABCs, will only face the link delays. For the purpose of this discussion, we will estimate the link delay, combined with the small amount of combinatorial logic delay in each router to be approximately 75% of a clock cycle. These numbers have been chosen to model a forward looking architecture. As we will show, the exact choice of these numbers is irrelevant as our technique will outperform a synchronizing router regardless of link latency. Nevertheless, according to International Technology Roadmap for Semiconductors [12], assuming 22 nm technology the link delay should be between 700 and 1300 ps/mm. If we assume a global clock frequency of approximately 1 GHz, this 75% corresponds to a link length of between 1.4 to 0.77 mm.

While our router microarchitecture tries to reduce skew, over larger chip sizes skew might become a major problem. To overcome this, buffers can be added at every  $n$  hops, where  $n$  can be calculated based on the longest path that a packet can take on a chip. The impact on the results to follow should be small as we estimate the additional latency added will be on the order of one cycle or less for the average packet. We leave the placement of skew removing buffers for future work.

### C. Routing Algorithm

To complement the new, double-chain topologies we use with ABC routers, we developed a modified version of the standard XY dimension order routing (DOR). Our routing algorithm leverages the observation that in any double chain network there are always three different deadlock free routes between any two nodes:

- 1) following the straight-path on the *blue* chain;
- 2) following the straight-path on the *red* chain;
- 3) following a DOR path starting on the *blue* chain and turning to the *red* chain.

Our algorithm attempts to optimize packet route based upon assumptions of link and turn delays under no load. Suppose that each link between two adjacent nodes causes a delay of  $x$  clock cycles. A turn costs three cycles in addition to the link delay. If a flit turns it faces a delay of  $(3/x + 1)$  times more than the link delay. Thus, if the number of hops is less than or equal to  $(3/x + 1)$ , traveling on ABC is a better choice than 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 from among the three legal paths between a source-destination pair based upon this estimation of latency under no-load. As the network congestion increases, we see that the turn cost has a degrading effect on performance.

Using the estimated router latencies discussed in Section IV-B yields a turn cost of  $(3/0.75 + 1) = 5$ . Thus, for the

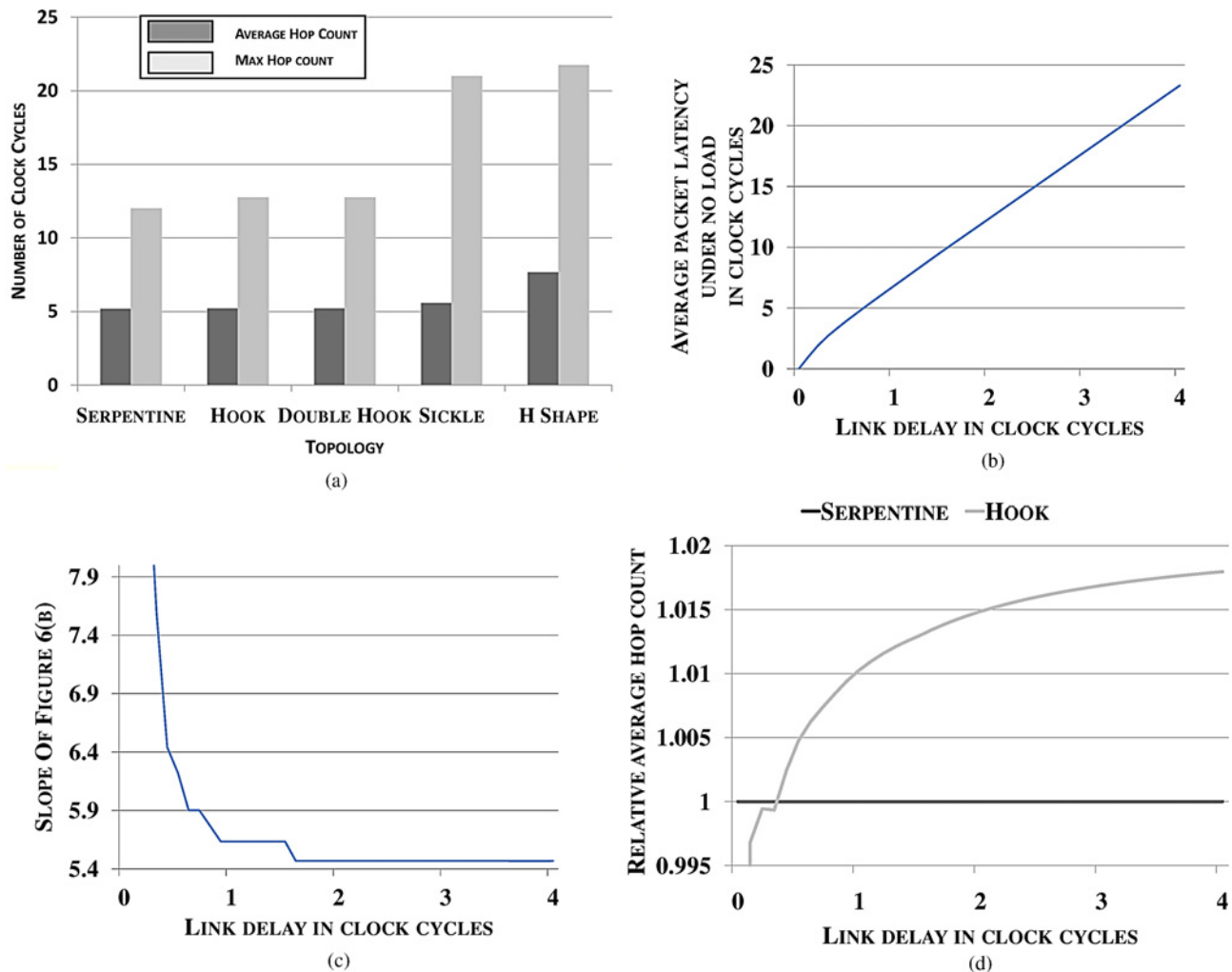


Fig. 6. Examination of the relative difference in hop count for topologies evaluated and variation of average hop count as link delay changes. (a) Average and worst case hop count for the five topologies. (b) Average no-load latency in terms of clock cycles for the Serpentine topology for different link delays and a turn cost of three cycles. (c) Slope of the curve in Fig. 6(b) for different link delays and a turn cost of three cycles. (d) Average hop-count for the Hook topology w.r.t. the Serpentine topology for different link delays and a turn cost of three cycles.

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. As can be seen from Fig. 1, a packet generated at (0, 6) will travel only through the blue chain to reach (6, 4). However, a packet generated at (0, 4) for (1, 1) will take up the blue chain and then turn and travel through the red chain, as shown in Fig. 3(a).

#### D. Analysis

In this subsection, we discuss our selection of an approximately optimal double-chain topology for a  $7 \times 7$  network. Initially, we ran an exhaustive search to find the optimum topology. We computed all the possible topologies that met the topology restrictions described in Section IV-A and evaluated them based upon average, zero-load latency for random traffic given the routing algorithm described in Section IV-C. Using this technique we were able to find the optimum topologies for  $4 \times 4$  and  $5 \times 5$  networks, however the exponential expansion of the solution set makes performing exhaustive search on larger topologies computationally prohibitive. Instead, we extrapolated the top five results obtained from the evaluation of  $4 \times 4$  and  $5 \times 5$  networks to up to a  $7 \times 7$  by network and then com-

pared the five resultant topologies to determine the best topology. As the search of both  $4 \times 4$  and  $5 \times 5$  networks produced the same five double-chain topologies, with the same order or precedence, we believe this method is likely to provide a network topology that is close to optimal for a  $7 \times 7$  network.

Fig. 5 depicts the five topologies evaluated for an  $7 \times 7$  2-D network. The evaluation results for these topologies are shown in Fig. 6(a). The figure shows both the average, as well as the worst case no-load, packet latency for every possible source-destination pair. As shown, the *Serpentine* topology performs the best with respect to both the average and the worst case latency. This is because it is the only topology among the five, in which each node is connected to all its physical neighbors through both the chains resulting in lower cost paths in comparison to the other topologies for all source and destination pairs. The *Hook* topology is also a close contender. We found, however, that as the turn-path penalty decreases with respect to the straight path, the *Serpentine* topology performs better than the *Hook* topology. Therefore, we use the *Serpentine* topology for our implementation. These five topologies do not represent all possible topologies, and we

plan to investigate an optimal ABC topology further in future work.

Fig. 6(b) presents the change in the average packet latency under no load for uniform random traffic in the *Serpentine* topology with a gradually incrementing link delay. Fig. 6(c) presents the variation in slope of the curve presented in Fig. 6(b). As is evident from Fig. 6(c), the curve for the average hop count is piece-wise linear. When the link delay is near *zero*, straight-paths are much more attractive than turn-paths even for the source and destination pairs which are placed on the opposite corners of the network. As the link delay increases, however, the relative penalty of the turn-path with respect to the straight-path decreases, and packets are routed such that they start facing turns. In this region, the slope rapidly falls, as slight increases in the link delays make more turns profitable. With further increase in the link delay, the rate of change reduces, and we find more regions of constant slopes. Finally, after a certain threshold [around 1.2 clock cycles in Fig. 6(c)], all source destination paths now require turns, and the average hop count now changes proportionally with the link delay and thus keeping the slope a constant. Therefore, with small link delay, packet latency is dominated by the turn cost latency, but with large link delays the packet latency is dominated by the link delay. In this region, most packets are routed along the same path as they would be routed in a 2-D Mesh topology except those that are produced at the nodes near the edges of the topology.

We found that the *Hook* topology performs marginally worse than the *Serpentine* topology for the 0.75 cycle link delay evaluated previously. The relative average hop count of the *Hook* topology with respect to the *Serpentine* topology for different link delays is presented in Fig. 6(d). For lower values of the link delay, when the packets are routed only on the straight path, the *Hook* topology outperforms the *Serpentine* topology. This is because, if only straight paths are taken into consideration, the nodes on the corners of the network are better connected to the other nodes, in case of *Hook* topology rather than the *Serpentine* topology. However, *Hook* loses this advantage with the increase in link delay, as turns become more attractive. Also, as the link delay increases beyond seven clock cycles, the average hop count of the *Hook* starts increasing linearly with respect to the link delay. However, the average hop count is worse than that of the *Serpentine* topology in this region.

Another important observation is that as the number of the nodes in the network increases into the hundreds of nodes, all the topologies evaluated tend to perform equivalently. This is due to the fact that in these large networks, the extra links that double-chain topologies provide at the edges of the network become less significant relative the distance from the edge of most traffic.

## V. EXPERIMENTS AND EVALUATION

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

### A. Methodology

In these experiments, we evaluate a network of ABC routers connected in the  $7 \times 7$  2-D *Serpentine* topology depicted in Fig. 5(a) and implemented using the routing algorithm described in Section IV-C. All the bi-FIFOs are eight flits deep. All FIFO depths are set to cover the round trip credit and synchronization time. To capture all ABC router network performance results, a fully synthesizable Verilog implementation of the ABC network was designed and simulated. Each flit is 128 bits wide. The baseline design consists of an  $7 \times 7$ , 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. An injection rate of  $n\%$  means that  $n\%$  of the clock ticks a data word is injected into the network on every link. 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.

To determine the performance under a realistic workload the ABC network was also evaluated under network traces taken from the SPLASH-2 suite of benchmarks [13]. The traces were obtained from a 49 node, shared memory CMP system simulator, arranged in a  $7 \times 7$  2-D Mesh topology [14]. Due to the simulator performance limitations associated with full Verilog simulation of all 49 cores, we were forced to run only a small portion of the actual trace. To have an unbiased evaluation, we chose 500 000 cycles from the middle of each trace with a warm-up period of 10 000 cycles.

Finally, the *Serpentine* topology, shown in Fig. 5(a), has one extra bisectional link than a standard 2-D Mesh topology. Assuming equal bitwidths between the ABC and baseline topologies would give the ABC network 14.28% more bisection bandwidth. To approximate a uniform wire density across all bisections between the ABC and baseline designs, we decreased the number of flits in each baseline network packet by one relative to the ABC design.

To evaluate the feasibility of our router design we synthesize the ABC router using a 45 nm ASIC design library using Synopsys DesignCompiler. Our synthesized ABC router achieves a clock frequency of 480 MHz. We expect that when scaled to 22 nm we could achieve a clock frequency of 1 GHz. The synthesized ABC router design consumes an area of 0.055 mm<sup>2</sup> in 45 nm, somewhat less than a comparable baseline 2-VC router which consumes 0.073 mm<sup>2</sup>.

### B. Synthetic Traffic

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

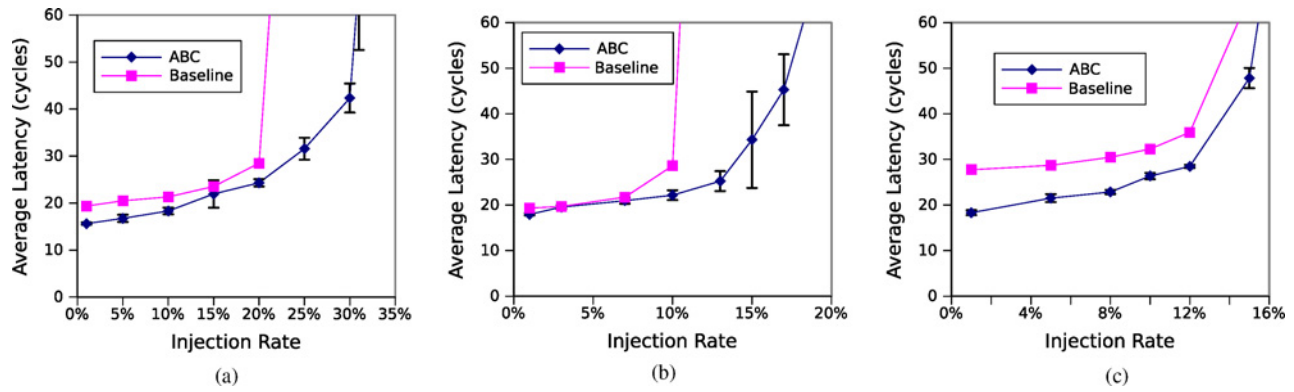


Fig. 7. Average latency versus injection rate for synthetic workloads. (a) Uniform random. (b) Transpose. (c) Bit-complement.

transpose traffic, node  $(x, y)$  sends packets to node  $(y, x)$ . In bit-complement traffic, node  $(x, y)$  exchanges packets with node  $(\bar{x}, \bar{y})$  where  $\bar{x}$  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 [15].

For uniform random traffic, shown in Fig. 7(a), ABCs offer 20% improvement in no-load latency and saturation throughput is improved by 50% over the baseline design. The improvement in the saturation throughput occurs due to our topology's increased number of bisectional links compared to a standard mesh network.

Transpose traffic, in Fig. 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, however, is greatly improved as compared to the baseline design.

For bit complement traffic, shown in Fig. 7(c), ABCs offer an improvement of 26% in no-load latency as well as 26% percent improvement in saturation throughput. In bit complement traffic, packets travel further than the other two synthetic patterns; 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 15% injection rate, as can be observed in Fig. 7(a). This is caused due to the following reason. At this injection rate, congestion causes the routers to switch back and forth between FIFO and ABC modes which lead to an increase in the relative rate of increase of latency until the load becomes large enough that the routers get locked in FIFO mode, thus causing a fall in the rate and thus a bump is formed.

### C. Realistic Traffic

Fig. 8 presents the relative performance of the ABC network with respect to the baseline design for the SPLASH-2 traces. From the figure it can be observed that ABC outperforms

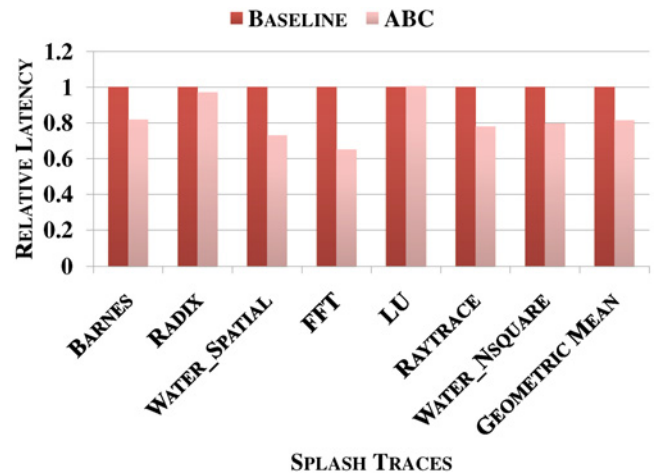


Fig. 8. Relative performance against baseline for realistic workload (SPLASH-2) [16].

the baseline design in all the traces except one, (LU), where baseline is marginally better than ABC. ABC shows an improvement of nearly 30–35% over the baseline design for FFT and Water-Spatial. The geometric mean for the relative performance is around 84.5% with respect to the baseline thus, on the average, the ABC network performed 15.5% better than the baseline design for the fraction of realistic traffic observed.

### D. Analysis

In this section, we will examine a variation from the baseline ABC router microarchitecture and analyze state machine performance.

1) *Flow Control*: As discussed in Section III-B3, the baseline ABC router microarchitecture implements a credit-based flow control. Unlike typical NoC router designs, credit-based flow-control requires credit counters be maintained for the downstream straight paths, as well as the turn paths because the ABC router effectively only latches packets at the output port. As a result, packet traversal of the upstream router must be throttled based upon the credit available in all the possible downstream paths, “and”ed together. We observe that given this constraint, it may be possible to simplify the router design somewhat via use of an *on/off* flow control methodology

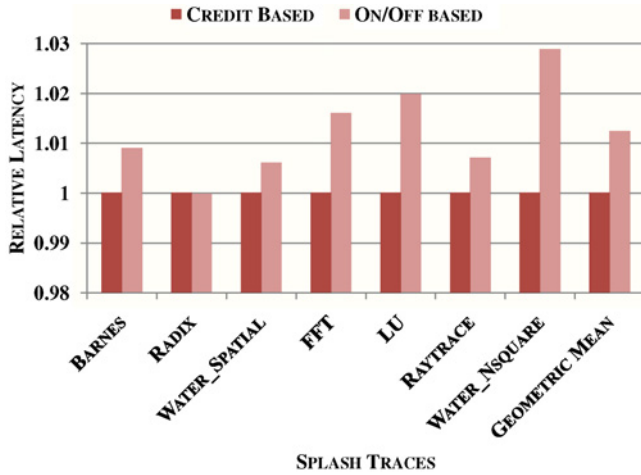


Fig. 9. Relative performance of ABC with on/off flow control against respect to ABC credit based flow control for realistic workload (SPLASH-2).

where only a single on/off signal is propagated from the downstream node to the upstream node. This signal would signify the availability of buffer space in the downstream node sufficient for any possible path of a packet from that input port, with extra buffering to cover the round trip control signaling time. Fig. 9 compares the relative performance of ABC network with on/off flow control against the baseline ABC network with credit based flow control for the SPLASH-2 traces.

To implement on/off flow control, the state machine was modified to base the FIFO mode to ABC mode transitions upon the assertion of the on signal, and to force the ABC mode to FIFO mode transition upon transmission of an off signal from the downstream node.

In case of on/off flow control, when the number of free buffer FIFOs 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 buffer FIFOs exceeds the threshold. This signal needs to be synchronized with respect to the outgoing clock of the upstream node. The on/off signal is generated in the downstream clock domain, and thus four flits of extra downstream FIFO buffers must be reserved to cover the round trip control signaling time. In the credit flow-control design, the credit signal on which state transitions are made, is updated in the outgoing clock domain; therefore, buffers do not need to be undercommitted. Thus, the effective FIFO depth available for on/off flow control based design is at least four less than the credit based flow control design. We therefore expect congestion to propagate faster for the on/off based design as compared to the credit based design, leading to lower utilization of the ABC path in case of on/off based flow control design.

Fig. 9 shows, as expected, the credit based design outperforms the on/off flow control based design for all the traces. For Water-Nsquare, the relative performance of on/off based ABC design with respect to the credit based ABC design is 3% worse, whereas on the whole the relative performance of the on/off based design is around 1% worse with respect to the credit flow control based design. These results show

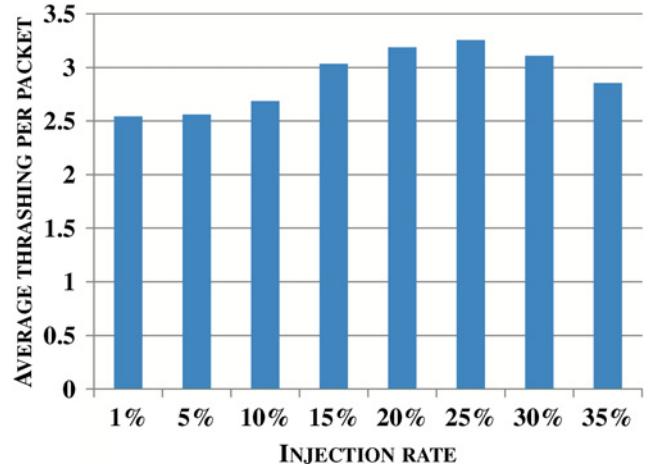


Fig. 10. Aborted state machine transitions (thrashes) per packet for different packet injection rates under uniform random traffic.

relatively minimal performance loss for the on/off flow control versus the credit flow control, making on/off a potentially attractive alternative if wire density for control signaling or logical complexity of the baseline ABC router design is too great.

2) *Aborted State Machine Transitions*: As shown in Fig. 4, the FSM traverses seven transition states during transition from the FIFO state to the ABC state and two transition states while traveling from ABC state to FIFO state. During the transition from the FIFO state to the ABC state, in case an input flit arrives before reaching state 5, the FSM reverts back to the FIFO state in one cycle. However, if the input flit arrives after state 4 and before reaching the ABC state, the FSM must traverse through state 1 and state 2 to return to the FIFO state, causing a loss of additional two cycles. Thus, a packet entering the router during the transition from the FIFO to ABC state may experience up to two cycles of additional latency beyond what it would experience if the router had remained in the FIFO state. These aborted state machine transitions, which we will label thrashes, are expensive in terms of packet latency. In this section, we will examine their frequency of occurrence.

Fig. 10 presents the number of instances per packet, when the output port of a router in the network attempted to switch to the ABC mode but was unsuccessful, under uniform random traffic in the network. In this figure, we only consider the instances when the FSM was able to reach state 5 but not the ABC state. As can be observed from Fig. 10, thrashing increases with increasing injection rate, and achieves a maximum at around 25% injection rate after which it drops down. This is because at lower injection rates there is lower congestion, and the output ports tend to remain in the ABC mode. Similarly, at very high injection rates, most of the router ports remain in the FIFO mode and do not attempt to switch often. Passing the 15% injection rate, however, the total traffic load is just enough, that when the router output port attempts to switch, a packet arrives causing it to revert back to the FIFO state and thus leading to poorer performance.

The affects of thrashing can be observed in all the three synthetic traffics (Fig. 7). In case of uniform random traffic

[Fig. 7(a)], a “kink” is visible at around 15% injection rate. This is because the relative increment in thrashing over previous injection rate is the highest for this value (Fig. 10).

One method to avoid router state machine thrashing is to adaptively determine when to transition from the FIFO mode to ABC mode based upon the instantaneous load at the router. While generally a router output port would prefer to switch to the ABC mode, the switching can be throttled based upon the traffic at that router node, thus reducing thrashing.

## VI. RELATED WORK

In this section, we describe the connections between this paper and other works which address performance improvements in asynchronous interconnection networks and new topologies to reduce hop count.

### A. Asynchronous Interconnect

ABC network is a multi-synchronous architecture where each node is clocked at the same frequency but can tolerate any amount of skew [17]. A critical aspect of multi-synchronous interconnect 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 multi-synchronous NoCs was proposed by Panades and Greiner, in the form of an optimized bi-synchronous FIFO featuring low-latency and small footprint [11]. This FIFO adds a latency of two and three clock cycles to gain robustness against metastability.

Chelcea and Nowick presented a design of a mixed clock FIFO used to interface two logic blocks operating in different clock domains. This design handles the synchronization problem by redefining the empty/full flags of the get/put interfaces, respectively, to be asserted one element early. Additionally, they also used a separate traditional full/empty signal to avoid deadlocks when only one element is present [7].

Another solution, presented by Dally and Poulton, consists of delay-line synchronizers, using a variable delay on the data lines [18]. This delay avoids switching in the metastable 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 undesirable as they are not always available in standard cell libraries.

Mangano *et al.* proposed the skew insensitive link (SKIL) solution to address metastability in multi-synchronous interfaces [19]. SKIL supports arbitrarily skewed clock signals by relying on a two-stage buffer FIFO structure by writing and reading flits into and from the buffer FIFO 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. In the

ABC design, when in ABC mode, the output port is clocked by the upstream clock provided by the incoming source-synchronous data, hence synchronization is not required to perform routing.

Another facet of GALS architecture is a completely asynchronous implementation. Asynchronous circuits exchange information using a handshake to explicitly indicate the validity and acceptance of data. This is in contrast to the multi-synchronous and synchronous design styles, which use a globally distributed or source synchronous clock signal to indicate moments of stability of the data. There are a number of alternative asynchronous design styles differing in how they indicate data validity. Some styles base the handshake upon a matched-delay path representing the slowest part of the data path, but these suffer from similar latency issues as source synchronous designs [20].

Horak and Nowick [21] presented a new asynchronous network design. They proposed new routing and arbitration primitives which have low overhead and utilize transition signaling. Also, the design is implemented nearly entirely using standard cells. Recently, they proposed a modified arbitration block where the router operation is biased toward one of the two input channels based on recent activity [22]. Examples of other NoCs based on asynchronous circuit techniques are CHAIN [23], MANGO [24], ANoC [25]. The chip area interconnect (CHAIN) network is implemented entirely using asynchronous, or clockless, circuit techniques. The network is based on narrow delay-insensitive highspeed links using one-of-four data encoding.

In order to address the design challenges of multi-application SoCs, CEA-LETI has proposed and developed ANOC [26], an asynchronous network-on-chip architecture, where NOC nodes and links are implemented using quasi-delay-insensitive asynchronous logic while the NoC functional units are implemented with independent clock domains using standard synchronous design methodologies.

While these techniques offer high performance in terms of latency and reduced idle power, they involve asynchronous or non-standard cell modules in the router design. ABC, on the other hand, can be implemented with standard cell design. Moreover, asynchronous communication requires a handshaking protocol before any data is sent. This handshaking requires multiple link traversal times to set up prior to data transmission whereas in a source synchronous design as ours; we can send the data immediately upon receipt, only accruing the latency of the link once for each hop.

### B. Low Hop Count Topologies

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 [27]. 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 [28], [29].

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 [30]. 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. Ogras *et al.* proposed a structured algorithm to insert application-specific long-range links in the network so as to reduce the network latency and improve throughput [31]–[33]. Unlike these approaches, the ABC router and topologies do not propose adding any extra router ports nor any extra wire density utilization, beyond the extra connections along the edges of the network.

In contrast to the prior work, 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. Also in contrast to EVCs and express cubes, ABCs are physical links present in the router offering an asynchronous bypass from the buffer FIFOs. Since ABCs are inherent to the router, they can be dynamically assigned at every hop, unlike EVCs which must be allocated upstream at specific nodes only. A packet takes an ABC whenever possible thereby avoiding FIFOs and synchronization delay, resulting in a lower per-hop latency.

## VII. CONCLUSION AND FUTURE WORK

In this paper, we presented the design, implementation, and evaluation of asynchronous bypass channel (ABC) routers. ABC routers aim to achieve lower latencies by eschewing synchronization and latching at the intermediate nodes between source and destination with a network. We presented a detailed microarchitecture for an ABC router.

We also presented a new class of network topologies and associated routing algorithm to complement the router design. This new class of double-chain topologies which leverage the ABC router's bias toward the straight path over the turn path. Our experiments showed that our ABC router, interconnected via a double-chain topology, produces significantly lower latencies compared to baseline. We presented a detailed analysis and evaluation of these topologies with respect to different design constraints.

This paper introduced a framework for ABC router based networks. The unique architecture of the ABC router offers many opportunities for future work. One avenue of future work we plan to pursue will focus on improving the performance of the network through lower latency synchronizing techniques in the turn path as well as the control logic.

Most of the realistic workloads depict a localized behavior such that load balance across the network is highly uneven. Adaptive techniques could be implemented to leverage the greater path diversity which double-chain topologies provide. In addition to more traditional adaptive routing techniques, the ABC router with its inherently low-latency ABC path could be used to implement a polymorphic network in which the router adapts which input–output port pairs are connected to the ABC channel based upon dynamic traffic information. Each router

can decide to place the ABC path between any combination of input and output ports depending on the dynamic load information on its input ports, thus allowing the ABC path to serve the maximum amount of traffic.

## REFERENCES

- [1] P. Pande, C. Grecu, M. Jones, A. Ivanov, and R. Saleh, "Performance evaluation and design tradeoffs for network-on-chip interconnect architectures," *IEEE Trans. Comput.*, vol. 54, no. 8, pp. 1025–1040, Aug. 2005.
- [2] A. Bassi, A. Veggetti, L. Croce, and A. Bogliolo, "Measuring the effects of process variations on circuit performance by means of digitally-controllable ring oscillators," in *Proc. IEEE Int. Conf. Microelectron. Test Structures*, Mar. 2003, pp. 214–217.
- [3] P. Macken, M. Degrauwe, M. V. Paemel, and H. Oguey, "A voltage reduction technique for digital systems," in *Proc. IEEE Int. Solid-State Circuits Conf.*, Feb. 1990, pp. 238–239.
- [4] W. Kim, M. Gupta, G. Wei, and D. Brooks, "System level analysis of fast, per-core DVFS using on-chip switching regulators," in *Proc. IEEE 14th Int. Symp. HPCA*, Feb. 2008, pp. 123–134.
- [5] 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 *Proc. IEEE ISCAS*, vol. 2, Jul. 1999, pp. 512–515.
- [6] M. J. S. Smith, *Application-Specific Integrated Circuits*, 1st ed. Boston, MA: Addison-Wesley Professional, 1997.
- [7] T. Chelcea and S. Nowick, "Robust interfaces for mixed-timing systems," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 12, no. 8, pp. 857–873, Aug. 2004.
- [8] P. Gratz, C. Kim, R. McDonald, S. Keckler, and D. Burger, "Implementation and evaluation of on-chip network architectures," in *Proc. Int. Conf. Comput. Des.*, 2006, pp. 477–484.
- [9] D. Wiklund, "Mesochronous clocking and communication in on-chip networks," in *Proc. Swedish Syst. Chip Conf.*, 2003.
- [10] L. Dennison, W. Dally, and D. Xanthopoulos, "Low-latency pleiochronous data retiming," in *Proc. 16 Anniversary Conf. Adv. Res. VLSI*, 1995, pp. 304–315.
- [11] I. Panades and A. Greiner, "Bi-synchronous FIFO for synchronous circuit communication well suited for network-on-chip in GALS architectures," in *Proc. 1st Int. Symp. NOCS*, 2007, pp. 83–94.
- [12] *International Technology Roadmap for Semiconductors* [Online]. Available: <http://public.itrs.net>
- [13] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta, "The SPLASH-2 programs: Characterization and methodological considerations," *SIGARCH Comput. Architecture News*, vol. 23, no. 2, pp. 24–36, 1995.
- [14] A. Kumar, L.-S. Peh, P. Kundu, and N. K. Jha, "Express virtual channels: Toward the ideal interconnection fabric," in *Proc. 13th ISCA*, 2007, pp. 150–161.
- [15] W. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. San Francisco, CA: Morgan Kaufmann, 2004.
- [16] S. Woo, M. Ohara, E. Torrie, J. Singh, and A. Gupta, "The SPLASH-2 programs: Characterization and methodological considerations," in *Proc. 22nd Annu. Int. Symp. Comput. Architecture*, 1995, pp. 24–36.
- [17] A. Sheibanyrad, A. Greiner, I. Miro-Panades, K. SoC, and L. de Paris, "Multisynchronous and fully asynchronous NoCs for GALS architectures (HTML)," *IEEE Des. Test Comput.*, vol. 25, no. 6, pp. 572–580, Nov.–Dec. 2008.
- [18] W. Dally and J. Poulton, *Digital Systems Engineering*. Cambridge, MA: Cambridge University Press, 1998.
- [19] D. Mangano, R. Locatelli, A. Scandurra, C. Pistrutto, M. Coppola, L. Fanucci, F. Vitullo, and D. Zandri, "Skew insensitive physical links for network on chip," in *Proc. 1st Int. Conf. NanoNet*, 2006, pp. 1–5.
- [20] T. Bjerregaard and S. Mahadevan, "A survey of research and practices of network-on-chip," *ACM Comput. Surv.*, vol. 38, no. 1, p. 1, 2006.
- [21] M. N. Horak, S. M. Nowick, M. Carlberg, and U. Vishkin, "A low-overhead asynchronous interconnection network for GALS chip multiprocessors," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 30, no. 4, pp. 494–507, Apr. 2011.
- [22] G. Gill, S. S. Attarde, G. Lacourba, and S. M. Nowick, "A low-latency adaptive asynchronous interconnection network using bi-modal router nodes," in *Proc. 5th ACM/IEEE Int. Symp. Netw. Chip*, 2011.
- [23] J. Bainbridge and S. Furber, "Chain: A delay-insensitive chip area interconnect," *IEEE Micro*, vol. 22, no. 5, pp. 16–23, Sep.–Oct. 2002.

- [24] T. Bjerregaard and J. Sparso, "A router architecture for connection-oriented service guarantees in the MANGO clockless network-on-chip," in *Proc. Conf. DATE*, vol. 2, 2005, pp. 1226–1231.
- [25] D. Rostislav, V. Vishnyakov, E. Friedman, and R. Ginosar, "An asynchronous router for multiple service levels networks on chip," in *Proc. 11th IEEE Int. Symp. Asynchronous Circuits Syst.*, Mar. 2005, pp. 44–53.
- [26] Y. Thonnart, E. Beigné, and P. Vivet, "Design and implementation of a GALS adapter for ANoC based architectures," in *Proc. 15th IEEE Symp. Asynchronous Circuits Syst.*, May 2009, pp. 13–22.
- [27] W. Dally, "Express cubes: Improving the performance of k-ary n-cube interconnection networks," *IEEE Trans. Comput.*, vol. 40, no. 9, pp. 1016–1023, Sep. 1991.
- [28] J. Kim, W. Dally, B. Towles, and A. Gupta, "Microarchitecture of a high-radix router," *ACM SIGARCH Comput. Architecture News*, vol. 33, no. 2, pp. 420–431, 2005.
- [29] B. Grot, J. Hestness, S. Keckler, and O. Mutlu, "Express cube topologies for on-chip interconnects," in *Proc. 15th Int. Symp. HPCA*, Feb. 2009, pp. 163–174.
- [30] A. Kumar, L. Peh, P. Kundu, and N. Jha, "Express virtual channels: Toward the ideal interconnection fabric," in *Proc. 34th Ann. ISCA*, 2007, pp. 150–161.
- [31] U. Ogras and R. Marculescu, "Application-specific network-on-chip architecture customization via long-range link insertion," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, 2005, p. 253.
- [32] U. Ogras, R. Marculescu, H. Lee, and N. Chang, "Communication architecture optimization: Making the shortest path shorter in regular networks-on-chip," in *Proc. Conf. Des. Autom. Test Eur.*, 2006, p. 717.
- [33] U. Ogras and R. Marculescu, "It's a small world after all: NoC performance optimization via long-range link insertion," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 14, no. 7, pp. 693–706, Jul. 2006.



**Tushar N. K. Jain** received the B.Tech. degree in electrical engineering from the Indian Institute of Technology Roorkee, Roorkee, India, in 2007, and the Masters degree from Texas A&M University, College Station, in 2010.

From June 2007 to July 2008, he was a Design Engineer with Bajaj Auto, Ltd., Pune, India. He is currently with AMD, Austin, TX. His current research interests include computer architecture, on-chip interconnection networks, application-specific integrated circuit design, and embedded systems.



**Mukund Ramakrishna** received the B.Tech. degree in electronics and communication engineering from the International Institute of Information Technology, Hyderabad, India, in 2010. He is currently working toward the Masters degree with the Department of Electrical and Computer Engineering, Texas A&M University, College Station.

From June 2010 to July 2010, he was a Visiting Research Student with the Intelligent Systems Research Center, University of Ulster, Derry, Northern Ireland, U.K. His current research interests include computer architecture, on-chip interconnection networks, application-specific integrated circuit design, and computer networking.



**Paul V. Gratz** (S'04–M'09) received the B.S. and M.S. degrees in electrical engineering from the University of Florida, Gainesville, in 1994 and 1997, respectively, and the Ph.D. degree in electrical and computer engineering from the University of Texas, Austin, in 2008.

He is currently an Assistant Professor with the Department of Electrical and Computer Engineering, Texas A&M University, College Station. From 1997 to 2002, he was a Design Engineer with Intel Corporation, Santa Clara, CA. His current research

interests include high performance computer architecture, processor memory systems, and on-chip interconnection networks.

Dr. Gratz co-authored *An Evaluation of the TRIPS Computer System* at the 2009 International Conference on Architectural Support for Programming Languages and Operating Systems, and received the Best Paper Award.



**Alex Sprintson** (S'00–M'03) received the B.Sc. (summa cum laude), M.Sc., and Ph.D. degrees in electrical engineering from the Technion Israel Institute of Technology, Haifa, Israel, in 1995, 2001, and 2003, respectively.

He is currently an Assistant Professor with the Department of Electrical and Computer Engineering, Texas A&M University, College Station. From 2003 to 2005, he was a Post-Doctoral Research Fellow with the California Institute of Technology, Pasadena. His current research interests include the

general area of communication networks with a focus on network coding, network survivability and robustness, and quality-of-service routing.

Dr. Sprintson is an Associate Editor for the IEEE COMMUNICATIONS LETTERS AND COMPUTER NETWORKS JOURNAL. His current honors include the Viterbi Post-Doctoral Fellowship and the NSF CAREER Award. He was a member of the Technical Program Committee for IEEE Infocom from 2006 to 2010.



**Gwan Choi** received the B.S., M.S., and Ph.D. degrees, all in electrical and computer engineering, from the University of Illinois at Urbana-Champaign, Urbana, in 1988, 1989, and 1994, respectively.

He is currently an Associate Professor with the Department of Electrical and Computer Engineering, Texas A&M University, College Station. He was with Cray Research, Inc., Mendota, MN, and Tandem Computers, Inc., Cupertino, CA. He has been a Visiting Scientist with the NASA Langley Research

Center, Hampton, VA. His current research interests include low-power high-performance very-large-scale integration design, system-on-chip design issues, and sensor/network design.

Dr. Choi is a member of Eta Kappa Nu. He has served as a program committee member of several international conferences. He has won a number of awards including the National Science Foundation Career Award in 1997.