Control System Communication Architecture for Power Electronic Building Blocks

Ivan Panchenko
Department of Computer Science and Engineering
Univ. of South Carolina
panchenk@email.sc.edu

Jason D. Bakos
Department of Computer Science and Engineering
Univ. of South Carolina
jbakos@cse.sc.edu

Herbert L. Ginn III
Department of Electrical Engineering
Univ. of South Carolina
ginnhl@cec.sc.edu

***Abstract*—Recent developments in SiC power devices have enabled the development of Power Electronic Building Blocks (PEBBs) having greater switching frequencies than Si-based devices such as IGBTs, but also require shorter time scales in their corresponding control systems. At the same time, system designers are scaling up modularized converter systems, such as the Modular Multilevel Converter, which can now comprise hundreds of PEBBs. Both of those trends present the need to evaluate architectural tradeoffs and communication requirements for hardware realizations of the Universal Controller Architecture. The control network should be designed to have minimal round-trip latency and maximal scalability. In this paper we present the results of a study to determine the most appropriate communication architecture and routing for networked PEBB control systems.**

Keywords—power electronics; FPGA; field programmable gate array; control systems; direct network; multi-hop network

I. INTRODUCTION

Present trends indicate that shipboard energy management systems will contain an increasing number of power electronic devices. In order to effectively explore the design space, system designers require an open and hierarchical power electronics system architecture with standardized modules and control interfaces. There has been progress in this area due to continued research and development of the power electronics building blocks concept [1-3]. However, there are many different control architectures for power electronics systems. In order to address the issue of custom control systems for each design, recent work [4,5] has presented the concept of the universal control architecture (UCA) in an attempt to standardize the control interfaces.

When a converter control system is partitioned, the partition interface should meet performance requirements of different control hierarchy layers, including requirements on data volume and transmission rates. Moreover, the interface should enable layer modularity such that replacement of any layer should not induce modifications in other layers. Beyond the need for modularity within individual converters, as shown in Figure 1, communication among application layer control modules forms the basis of a coordinating system control that allows for system wide energy management strategies [6]. The need for both increased switching frequency and greater modularity require the ability to evaluate architectural tradeoffs and communication requirements for hardware realizations of the Universal Controller Architecture.

The stability and performance of the system of PEBB modules is affected by the delay between when measurements are taken and when updated references are received from the controller. Since each level of the PEBB control hierarchy is connected in a local topology, transitioning packets between control levels will also contribute to the delay. In the paper we present a new FPGA-based UCA along with a proposed communication network topology. We then characterize the impact of the communication topology on latency as the system size is scaled.

II. CONTROLLER NETWORK TOPOLOGY AND ROUTING

In a ship-wide PEBB-based power distribution system, control and measurement modules are spatially distributed. While modules that form the control system for a single converter may be somewhat co-located, modules at the application level of control and above will be distributed throughout the overall system. Therefore, it is generally not feasible to connect them all directly into a single central controller. Instead, it is more practical to distribute control among the modules within converters and at layers above individual converter control, such as zonal or bus level controls. Using a multi-hop network, each control module contains a small integrated router that can both serve as a network interface and serve as an intermediate forwarding point for other messages sent among other control modules. In these types of networks, the worst-case message latency is determined by the longest possible path between two control modules. This worst-case latency serves as a constraint for the overall control system design. As such, both the physical topology of the communication network and the routing algorithm are important considerations for the system design.

Figure 2 shows a simple 1D bidirectional ring topology, where there is only one minimal-distance path between any two endpoints. The worst case round trip path delay is n, where n = the number of nodes (where a message must traverse $n/2$ rings in both directions). In this toplogy, each module requires only two bidirectional channels.

Figure 3 shows a 2D torus topology, which offers more than one possible minimum-length paths between any two endpoints that are not horizontally or vertically aligned. The 2D torus has a worst-case round trip latency of $\sqrt{n}$and requires four

This material is based upon research supported by the U.S. Office of Naval Research under award number N00014-15-1-2346.

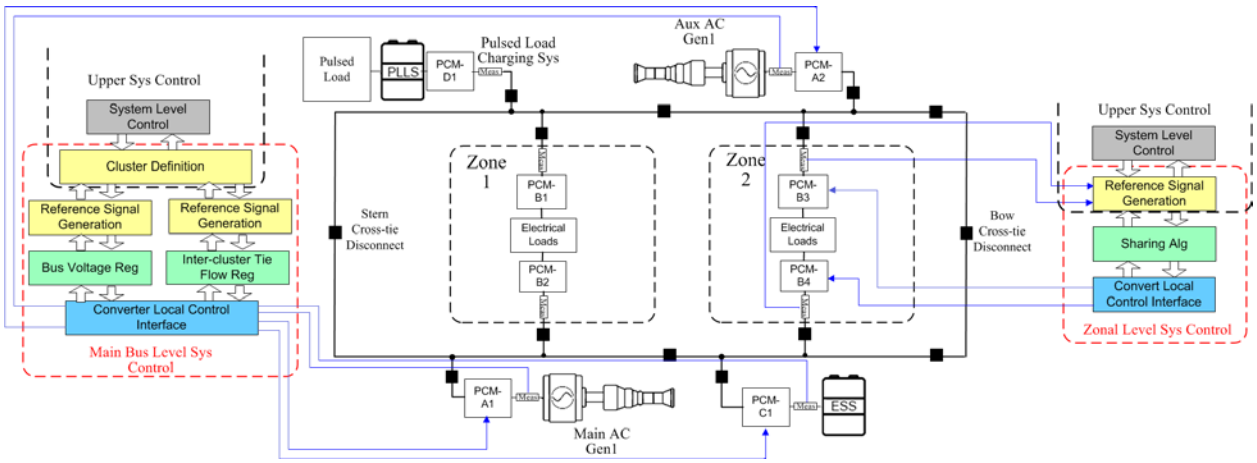

Fig. 1. Portion of a notional shipboard power system with communication among converters' control systems.

bidirectional channels per node.. Extending further, a 3D torus would require six channel per node and have a worst-case round-trip latency of $n^{1/3}$. We selected a 2D torus as the best compromise between hardware cost and performance.

A. *Multi-Hop Network Topology*

In 2D topologies, the existence of multiple minimum-length paths between most endpoint pairs requires additional considerations in order to maximize the utilization of the network's aggregate channel capacity. In a 2D torus of width w and height h, a message sent between nodes having addresses (x_1, y_1) and (x_2, y_2) has the following offsets in both dimensions:

$$\Delta x = \min\Big(\big((x_1 - x_2)\ mod\ w\big), \big((x_2 - x_1)\ mod\ w\big)\Big) \quad (1)$$

$$\Delta y = \min\Big(\big((y_1 - y_2)\ mod\ h\big), \big((y_2 - y_1)\ mod\ h\big)\Big) \quad (2)$$

The required single path routing distance is $\Delta x + \Delta y$ hops but there are

$$length_{min} = \frac{(\Delta x + \Delta y)!}{\Delta x! \Delta y!} \quad (3)$$

possible minimum-length paths.

In this paper we assume that a single node serves as an egress/ingress point to the higher-level control layer. In this way, all nodes transmit and receive measurement and control data to/from this node, and the resulting control loop imposes real-time performance constraints on the network and the on-chip routers.

In order to estimate the minimum round trip latency for various network sizes, we developed an analytical model based on the example assumed system parameters shown in Table 1. Assuming that the chosen parameter values do not exceed the maximum bandwidth of any single channel, each packet will experience a round-trip latency

$$latency_{roundtrip} = 2 \cdot \left(\frac{latency_{Aurora} + latency_{route}}{freq_{FPGA}} + \frac{size_{packet} \cdot 8}{bw_{Aurora}}\right) \cdot \left(\frac{1}{2}w + \frac{1}{2}h\right) \quad (4)$$

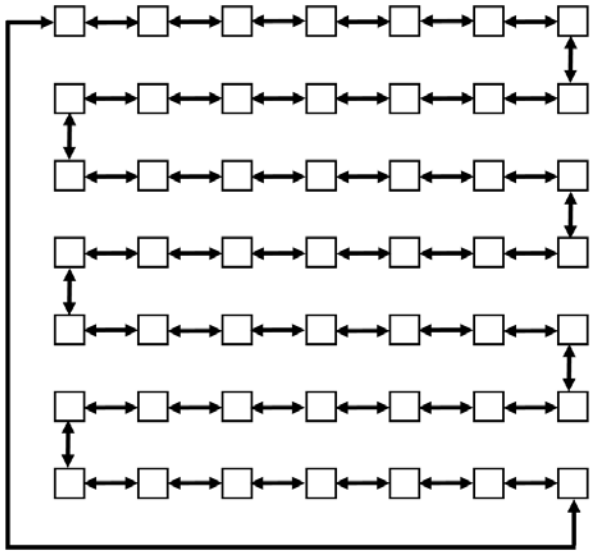

Fig. 2: Ring topology, 2 channels/node, worst case latency = n/2.

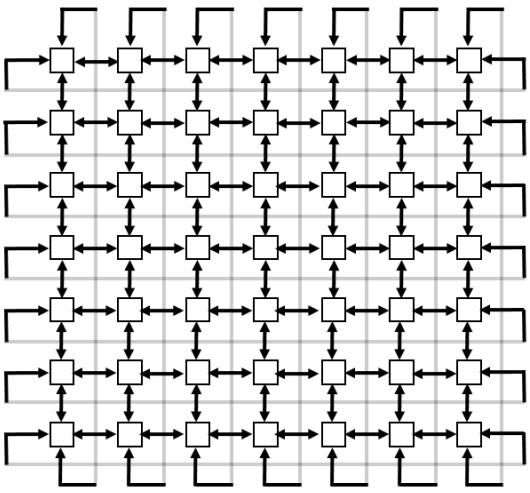

Fig. 3: Torus topology, 4 channels/node, worst case latency = √n/2.

Table 2 shows minimum round trip latencies for the parameter values shown in Table 1.

B. Routing Algorithms

As shown in Figure 4, the simplest routing scheme for multi-hop networks is X-Y (also called dimension-ordered) routing, in which the network routes packets in the X dimension until the packet reaches a node that is vertically aligned to the destination and then routes in the Y dimension [7]. X-Y routing is simple to implement and is guaranteed to follow minimal length routes.

For the traffic pattern for PEBB control networks, where all nodes periodically send one packet and receive one packet from the ingress/egress node, the north and south channels into the ingress/egress node must carry more traffic than the east and west channels. In this case, both the north and south channels will experience $\frac{w \cdot h - w}{2}$ packet traversals while the east and west channels will experience only $\frac{w}{2}$ packet traversals. The east-west channels will require a maximum channel utilization equal to

$$bw_{utilization} = \frac{size_{packet} \cdot 8 \cdot \frac{w \cdot h - w}{2}}{latency_{roundtrip}} \quad (5)$$

In order to avoid this load imbalance, the routing algorithm should equally distribute the network traffic across the channels along the minimum paths, especially around the highest congested areas around the ingress/egress node. Ideally, each of the four of the ingress/egress node's channels should experience $\frac{w \cdot h}{4}$ packet traversals. To achieve this we propose "hub routing", comprised of a set of pre-computed static routes between each node and the ingress/egress node, where each packet follows a path that keeps its location on the grid closest to the straight line between the node and the ingress/egress node.

TABLE 1: DESIGN PARAMETERS

Parameter	Variable	Expected value
Maximum latency of the Aurora links	$latency_{Aurora}$	53 clock cycles
Packet size	$size_{packet}$	100 bytes
Routing latency	$latency_{route}$	1 clock cycle
Link bandwidth	bw_{Aurora}	10 Gb/s
FPGA user clock frequency	$freq_{FPGA}$	156.25 MHz
Network size	n	100 nodes
Network order, $n = o^2$	o	10 nodes

TABLE 2: MINIMUM ROUND TRIP LATENCIES.

Network size	Round trip latency
5x5	4.3 us
10x10	8.5 us
20x20	17.0 us
30x30	25.6 us
40x40	34.1 us
50x50	42.6 us

We compute the distance between a given node at location (x_0,y_0) and a straight line ($ax + by + c = 0$) in the traditional way, i.e. $\frac{|ax_0+by_0+c|}{\sqrt{a^2+b^2}}$. Each node's integrated router can implemented hub routing through the use of a small, pre-computed static routing table comprised a $\frac{w \cdot h}{4} \times$ 3-bit memory.

Figure 5 shows an example path computed with hub-based routing, where each packet follows a path that keeps its location on the grid closest to the straight line between the node and the ingress/egress node. Thus the maximum-loaded channels will require a maximum channel utilization equal to

$$bw_{utilization} = \frac{size_{packet} \cdot 8 \cdot \frac{w \cdot h}{4}}{latency_{roundtrip}} \quad (6)$$

Table 3 compares the minimum channel bandwidth utilization for both X-Y and Hub Routing, assuming the parameters given in Table 1. X-Y routing requires more than the available 10 Gb/s bandwidth when scaling the network to 30x30, while the Hub routing supports network sizes up to 40x40.

TABLE 3: MINIMUM LINK BANDWIDTH

Network size	$bw_{utilization}$ (Gb/s): XY	$bw_{utilization}$ (Gb/s): Hub
5x5	1.9	1.2
10x10	4.2	2.3
20x20	8.9	4.7
30x30	13.6	7.0
40x40	18.3	9.4
50x50	23.0	11.7

C. Related Work

Much of the current work in routing and topologies for multi-hop networks on FPGAs focus on networks-on-chip where a single FPGA contains all the routers comprising the network. In this case the router must be as compact as possible [8,9]. These networks typically use non-minimal deflection-routing to avoid the need for buffers in the router. Deflection routing allows packets to follow non-minimal routes when the outgoing ports on the minimal path(s) are currently occupied with other traffic, as opposed to buffering in the router. Because deflection routing increases latency and timing uncertainty it is not appropriate for our application. Well-known algorithms developed for distributed computing also generally employ non-minimal routing to maximize throughput, often at the cost of latency [10]. These networks are also generally designed for dynamic traffic patterns, as opposed to the static patterns assumed for controller networks. Work that focuses on multi-FPGA systems often focus on exploration of network topologies and not specific routing algorithms, and often do not explicitly consider the overheads contributed by the on-chip processors that interact with the network [11,12].

III. EXPERIMENTAL PLATFORM

To explore the feasibility for a PEBB control network, we used an off-the-shelf KC705 FPGA board with an attached

quad-SPF+ transceiver FPGA mezzanine connector (FMC) module.

The KC705 has a relatively small Xilinx Kintex-7 FPGA with 203K logic slices and 2 MB of on chip RAM. The board can connect directly to PEBB hardware managers or other PEBB control level interfaces via a secondary FMC expansion connector. The FPGA boards are themselves interconnected via four optical channels to form a control network to form a closed loop control network among the boards. The boards also connect to a secondary, non-real time network through 1 Gb Ethernet for monitoring and control.

The design programmed onto the FPGA is structured as a system-on-chip (SoC), consisting of two Microblaze soft core microcontrollers, on-chip memories, and DMA engines connected to the four bidirectional 10 Gb/s channels using the Xilinx 66b64b Aurora link-layer protocol.

A. Platform Design Considerations

Each PEBB control module collects off-board measurements from the attached power electronics and encodes and transmits the measurements and control data over the multi-hop control network to other control nodes either within the same control hierarchy layer or across a layer boundary as dictated by the control loops in operation. Each node will later receive a corresponding control message from other nodes or layers. Since each operating control loop is deterministic, each control node must complete these tasks according to a fixed control period. In addition, the control system must also route messages on behalf of PEBB control modules on their path to or from other locations in the control network as needed.

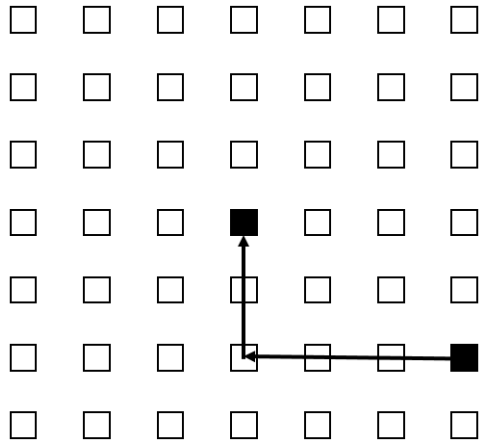

Fig. 4: X-Y Routing.

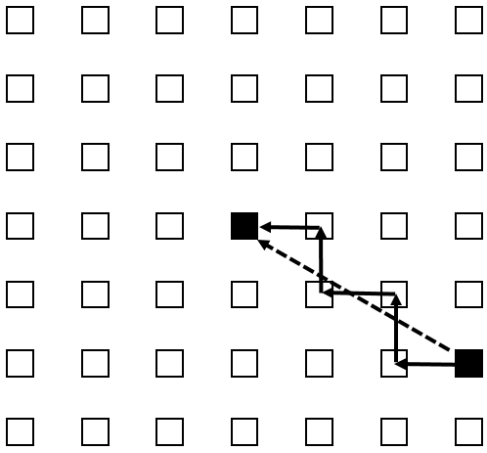

Fig. 5: Hub-based routing.

The control system is constrained by the communication latency imposed both by the network (in terms of worst-case path length) but also the on-chip overheads of processing and forwarding packets, which may be significant since we are using relatively low-speed microcontrollers. Longer worst case delays will constrain the minimum control period for a given control layer.

Likewise, as described in Section IIB, the effective channel bandwidth limits the maximum size/scale of the network, since larger networks have more overlapping routing paths requiring more channel bandwidth. Like other network technologies, the effective bandwidth is dependent on the packet size. Although Xilinx's Direct Memory Access (DMA) IP modules allow the programmer to specify an interrupt threshold that defines the number of received packets before the module triggers an interrupt, our current implementation issues an interrupt after each received packet. In this model, packets comprised of fewer bytes will require a higher interrupt rate to achieve higher utilization of the 10 Gbps channel.

Table 4 shows the required interrupt rate and the corresponding number of cycles allowed for the packet handler to utilize all channel bandwidth, assuming an interrupt threshold of one. A packet size of 32 would require 42 million interrupts per second, leaving only 2 cycles per interrupt using a 100 MHz clock, which is obviously impractical. As shown, it is only feasible with 2 KB packets and above to achieve a substantial level of channel utilization. We confirm these results experimentally later in this section. In future work we will we explore the impact of adjusting the interrupt threshold to allow multiple in-flight packets.

TABLE 4: REQUIRED PROCESSOR WORKLOAD TO ACHIEVE MAXIMUM THEORETICAL CHANNEL BANDWIDTH.

Packet size (bytes)	Interrupt rate to saturate 10 Gbps channel (interrupts per sec)	Maximum # clock cycles permitted for handler code (@100 MHz)
32	42 M	2
64	21 M	4
512	2.6 M	38
2 KB	655 K	152
4 KB	328 K	305
8 KB	163 K	610

Figure 6 shows a block diagram of the design we programmed into the FPGA. The design is logically split into two subsystems mastered by a separate Microblaze microntroller: the *controller subsystem* and the *monitor subsystem*. The two subsystems are isolated and share only one common peripheral, an on-chip BRAM that holds the controller

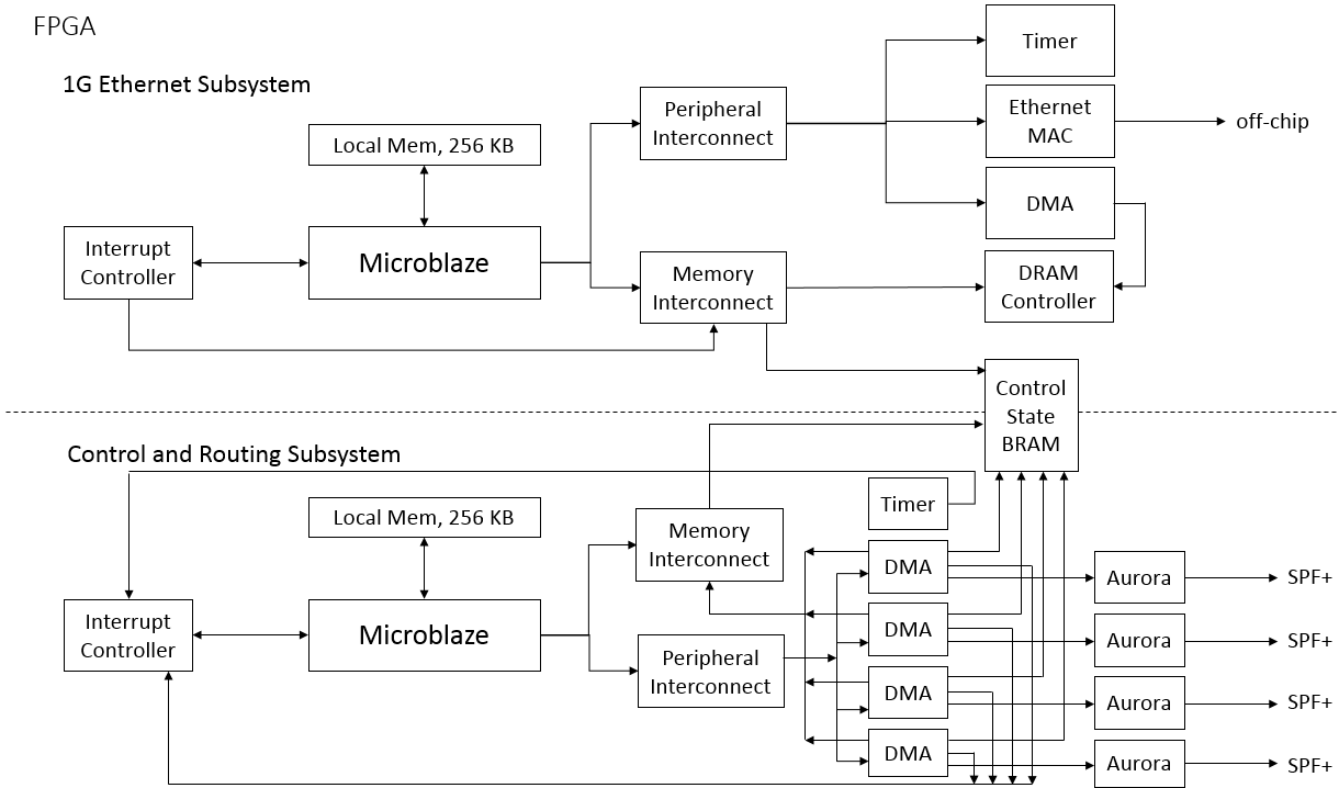

Fig. 6: Top-level Design.

state. Both processors have local on-chip memory from which they execute their respective program code, both processors have independent interrupt controllers, and both processors have independent timers (the monitor processor uses its timer for the TCP/IP stack). The TCP/IP stack stores its data on off-chip DRAM.

B. Controller Subsystem

The controller subsystem performs the control and routing tasks on behalf of the module and is optimized for latency and determinism. To minimize the amount of unpredictable delays, we took the following steps: (1) store the microcontrollers's software and data in on-chip memory, as opposed to off-chip memory, which has substantially higher latency, (2) limit the set of interrupts to only the four DMA interrupts corresponding to the four DMA modules connected to the Aurora interfaces (which only interrupt the processor when a packet arrives from any of the Aurora interfaces) and a timer interrupt (which interrupts the processor when it is time to collect measurements and transmit a message to the zone controller), and (3) place the interrupt controller in fast mode, in which the interrupt controller passes the handler address directly to the processor without any software intervention.

C. Monitoring Subsystem

We use a non-real time 1 Gb/s Ethernet interface for monitoring and control of the module. The Ethernet subsystem runs as a fully-custom hardware IP module in the FPGA logic fabric but its TCP/IP stack runs in software. The TCP/IP stack is heavyweight and imposes unpredictable loads on the microcontroller, but when running on its own microcontroller it cannot interfere with the control subsystem.

IV. Experimental Results

In this section we describe characterization results of our evaluation platform.

A. Latency

In order to evaluate the internal latency of controller subsystem, we set up an experiment with a single board having a loopback cable from channel 0 to channel 1. The software would transmit one packet every control period, and the DMA interrupt handler measured the round-trip delay. This measurement includes the latency contributions from the transmitting DMA engine, the transmitting Aurora interface, the optical transmission latency, the receiving Aurora interface, the receiving DMA engine, and the interrupt controller. These values represent the effective channel latency for one hop.

Figures 7 shows the distribution of packet latencies over 1 million packet transmissions for a 32-byte packet and a 4 KB packet. Note that the Y-axis of the histograms is plotted on a logarithmic scale. For the 32-byte packet, 18.3% of the packets experienced 1150 to 1200 cycles of latency and 81.6% of the packets experienced 1200 to 1250 cycles of latency.

On the Microblaze's 100 MHz clock, 1200 cycles equivalent to 12 us, while the transmission time of a 32 byte packet on a 10

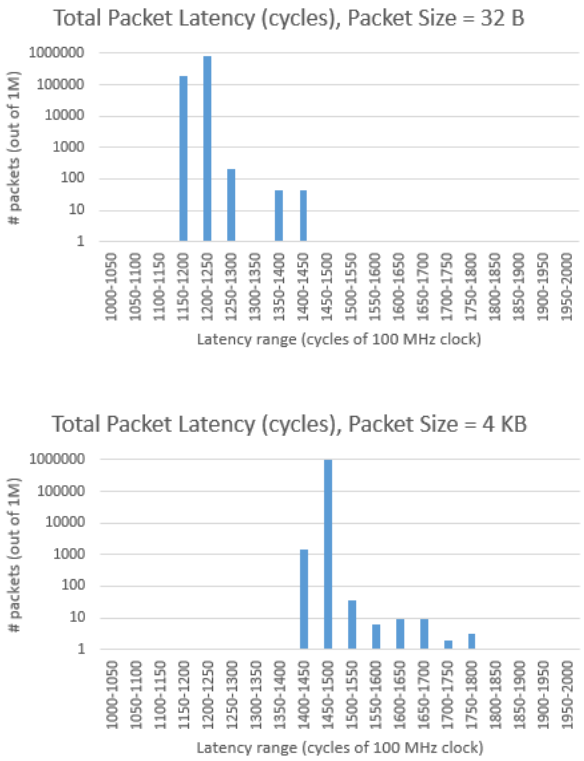

Fig. 7: Observed packet transmit latency for 32 byte packets (top) and 4 Kilobyte packets (bottom). These results include packet transmission time over the 10 Gbps link (~3 cycles for a 32 byte packet and ~328 cycles for a 4KB packet) and the platform overheads, such as those contributed by the on-chip interconnect, DMA engines, interript controller, and driver software. Note that the Y-axis is logarithmic.

Gb/s channel is 25.6 ns (note that our clock rate is less than the example parameters listed in Table 1).

For the 4 KB packet, 99.9% of the packets experienced 1250 to 1300 cycles of latency, against a 3.2 us expected transmission time. The ~100-cycle latency difference between the 32-byte and 4 KB packet size is equivalent to approximately 1 us, caused by the higher transmission time for the larger packet.

These results indicate that the packet size has little relative effect on the end-to-end transmission latency, since a 128X increase in packet size required only a 5 to 10% increase in latency. Note that because the platform overheads are 3.9X to 468X that of the channel transmission time.

B. *Bandwidth*

To evaluate the effective channel bandwidth, we added a transmit command to the DMA handler that causes the software to transmit a new packet immediately after receiving a packet. We used a 2000-cycle timer interrupt to gather statistics.

Figure 8 plots the effective bandwidth of the channel, in Megabits per second, versus the packet size. The 32-byte packet size uses 38 Mbps of the channel capacity, the 512-byte packet size uses 614 Mbps, the 4 KB-packet size uses 3.2 Gbps, and the 8 KB-packet size uses 6.5 Gbps.

These results are consistent with the extrapolated results shown in Table 4, which shows there is insufficient time to process smaller packets and allow the processor to achieve full channel utilization. Our observed bandwidth is even lower than Table 4 suggests, since the processor must also periodically call the timer interrupt handler, which calculates and records performance statistics. In this test we lose additional performance because we only allow for up to one in-flight packet. In future work we will incorporate more descriptor-based DMA and/or flow control to allow for multiple simultaneous in-flight packets to improve effective bandwidth for smaller packet sizes.

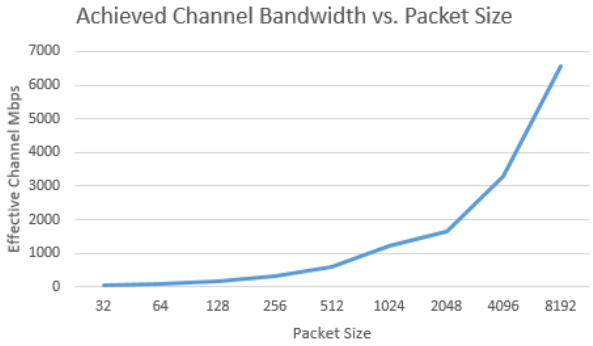

Fig. 8: Observed Aurora channel bandwidth versus packet size.

V. Conclusions and Future Work

This paper describes a general methodology for building power electronic building blocks (PEBBs) based converters and systems of converters, where individual PEBB modules are coupled with embedded controllers interconnected on a distributed multi-hop communication network. We advocate the use of FPGA-based base boards, where the FPGA serves as a substrate for embedded microcontrollers executing software that performs the control, routing, and monitoring functions.

We evaluated two routing algorithms and used an analytical performance model to evaluate the impact of load balancing on system scale. Our proposed hub-based routing algorithm is capable of balancing channel load for a static traffic pattern where all modules engage in a closed control loop with a single ingress/egress point to other control layers.

Our proposed FPGA design is decomposed into two mostly isolated subsystems. One of these systems is designed for real-time control and control network routing and the other for non-real time instrumentation and monitoring. We characterized the network performance of the 10 Gbps communication infrastructure, and showed that larger packets, or possibly higher interrupt thresholds, are needed to achieve high channel utilization., Moreover, the interrupt handling capabilities of the softcore microcontrollers adds significant latency overhead, possibly necessitating hardware acceleration for packet forwarding.

In future work we plan to develop hardware-based routers to lessen the impact of processor overhead on packet latency, and enable scatter-gather DMA mode to allow for multiple-inflight

packets between interrupts in order to lessen the impact of processing overhead.

ACKNOWLEDGMENT

This work was supported by the Office of Naval Research under contract N00014-15-1-2346.

REFERENCES

[1] T. Ericsen, “Power Electronics Building Blocks – a Systematic Approach to Power Electronics,” Proceeding of the 2000 IEEE Power Engineering Society Summer Meeting, Volume: 2, 16-20, July 2000, pp. 1216-1218.

[2] F. Wang, S. Rosado, D. Boroyevich, “Open Modular Power Electronics Building Blocks for Utility Power System Controller Applications,” Proceedings of IEEE 34th Power Electronics Specialist Conference, Vol. 4, 15-19, June 2003, pp. 1792-1797.

[3] T. Ericsen, “SiC-PEBB based zonal distribution system architectue,” Ericsen Innovations, May 2013.

[4] IEEE Std. 1676-2010, “Guide for Control Architecture for High Power Electronics (1 MW and Greater) Used in Electric Power Transmission and Distribution Systems”, 11 Feb. 2011.

[5] Hingorani, N., Ginn, H, Sullivan, J., "Control/protection architecture for power electronic converters", Proceedings of Electric Ship Technologies Symposium (ESTS), 2011 IEEE , pp. 472 – 477.

[6] M.R. Hossain and H.L. Ginn III, “Real-Time Distriubed Coordination of Power Electronic Converters in a DC Shipboard Distribution System,” IEEE Trans. On Energy Conversion, Vol. 32, No. 2, June 2017, pp. 770-778.

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

[8] N. Kapre and J. Gray, "Hoplite: Building austere overlay NoCs for FPGAs," Proc. 25th International Conference on Field Programmable Logic and Applications (FPL).

[9] Nachiket Kapre, "Implementing FPGA Overlay NoCs Using the Xilinx UltraScale Memory Cascades," Proc. 25th IEEE International Symposium on Field-Programmable Custom Computing Machines, 2017.

[10] Arjun Singh, William J. Dally, Amit K. Gupta, Brian Towles, "GOAL: a load-balanced adaptive routing algorithm for torus networks," Proceedings of the 30th annual international symposium on Computer architecture 2003.

[11] Trevor Bunker, Steven Swanson, "Latency-Optimized Networks for Clustering FPGAs," Proc. 21st Annual International IEEE Symposium on Field-Programmable Custom Computing Machines, 2013.

[12] Andrew G. Schmidt, William V. Kritikos, Rahul R. Sharma, Ron Sass, "AIREN: A Novel Integration of On-Chip and Off-Chip FPGA Networks," Proc. 17th IEEE Symposium on Field Programmable Custom Computing Machines, 2009.