Hybrid Hard NoCs for Efficient FPGA Communication

Tianqi Liu, Naveen Kumar Dumpala, and Russell Tessier

Department of Electrical and Computer Engineering
University of Massachusetts, Amherst, MA, USA

***Abstract*—Recent research has shown that the integration of a custom-silicon network-on-chip (NoC) into an FPGA fabric can significantly help on-chip communication bandwidth. Although not appropriate for all communication, FPGA hard NoCs provide a scalable infrastructure that will increase in importance as FPGA sizes increase. To date, most FPGA hard NoC implementation has focused on packet-switched routing which requires dynamic run-time decision making to transfer data from source to destination. In this paper we explore expanding hard NoC routers to support both packet-switched and prescheduled time-multiplexed communication. By limiting the use of energy-hungry routing buffers, time-multiplexed routing allows for throughput-predictable data transport with reduced energy consumption versus packet-switching for a variety of traffic patterns. The area overhead required to convert a packet-switched router to a hybrid packet-switched/time-multiplexed version is minimal (about 9%). In this research, we show significant NoC energy improvement (about 33%) while maintaining or improving packet latency values versus packet switching for select data traffic patterns.**

I. Introduction

As the size of FPGAs and their use in high-bandwidth applications increase, there is a growing need for high-speed on-chip data transfer. NoC hardware created from custom silicon (hard NoCs) [1] has been shown to support high-throughput applications. The fast switching flexibility of these resources allow for wide data transfer at clock rates of greater than 1 GHz, which are well beyond what can be achieved with circuits implemented in a traditional FPGA fabric. The small area consumed by hard NoCs (between 1-2% of overall area) relative to the remainder of the FPGA fabric makes them particularly attractive for a variety of applications with unpredictable data transfer patterns (e.g. Internet switch).

Most recent work on hard FPGA NoCs has focused on packet-switched NoCs where routing decisions are made at the grain size of a packet. Source-to-destination packet routing is controlled by a packet header that contains routing information and the routing algorithm used by each router. These run-time dynamic decisions call for an allocation of routing resources as needed. If traffic is unpredictable, buffering is provided in the network router to allow fair access to contested routing resources. Extensive experimentation has shown that this configuration allows for high throughput and, in restricted cases, predictable packet latency. Given the wide array of FPGA uses for packet-switched networks and their low overhead, hard NoCs are strong candidates for future FPGA integration.

In many cases, on-chip data communication between portions of FPGA designs is predictable at design compile time. These data streams often have known bandwidth rates and communication patterns that exist throughout design execution. In these cases, hard NoC communication resources can be preallocated at *compile time* to guarantee both bandwidth and low latency for a sizable number of data streams. The remaining bandwidth can be used for less-predictable packet-switched routing. Our time-division multiplexed (TDM) addition to hard NoC routers not only provides an effective mechanism for dealing with continuous data streams, but it also can save significant dynamic energy and support multicast routing. Since routing patterns are predictable, repetitive accesses to energy consuming data buffers in NoC routers are eliminated, reducing overall dynamic energy consumption.

In this paper, we provide a design for a *hybrid* hard NoC that is customized for use in FPGAs. This NoC supports packet-switched-only routing, TDM-only routing, or a simultaneous combination of the two routing approaches. A routing algorithm is used to preallocate persistent stream bandwidth to the NoCs during design compilation. Communication is broken into a series of time slots that can be used in each router for either packet-switched or time-multiplexed communication. Although hybrid routers have been developed in the context of multi-core NoCs (primarily to isolate high-bandwidth memory channels) [2], little work has been done on adapting the approach to FPGA NoCs that require fewer hardware resources and exhibit a number of important restrictions. In this work, we show that the amount of additional hardware required to build a hybrid NoC router is about 9% versus a previous packet-switched-only version without affecting router performance.

II. Background

A. Related Work

As packet-switched NoC router architecture has matured over the past decade, the basic elements of a packet-switched router have become standardized [3]. These routers consider the transfer of packets of information from a source core to a destination core as a series of flow-control digits (flits). When a flit arrives at a router input port it is buffered and subsequently forwarded to an output port based on a previously-determined allocation mechanism. A packet may be assigned to one of several input virtual channels (VCs). Typical components of a router include input buffers to store

978-1-5090-5602-6/16/$31.00

flits, a VC allocator to determine the output virtual channel for an incoming flit, a switch allocator that identifies which input buffers connect to specific outputs on a clock cycle, and the switch (typically a crossbar) that makes the physical connections. Each connection is determined by a packet header flit that indicates the packet destination. This information, in concert with a router's routing policy determines router input-to-output connections. A typical routing policy is dimension-ordered routing (DoR) in which packets are routed horizontally first and then vertically (or vice versa). If one VC input buffer becomes blocked due to congestion, flits in the other VC(s) may continue forward.

One approach to developing an FPGA NoC is to fashion one from soft FPGA logic and memory blocks [4] [5]. While this approach is flexible and allows customization on a per-application basis, soft NoC routers are up to 23× larger in terms of area and 6× slower than their hard NoC counterparts [6]. This overhead has motivated the study of hard FPGA NoCs [1] [7] [8]. In particular, Abdelfattah *et al.* [1] [8] provide a complete view of the embedding of a hard packet-switched router in an FPGA, soft IP core interfacing, limitations, and overheads. However, these works do not consider the use of prescheduled time-multiplexed routing using router components or data multicasting.

Although hybrid hard NoCs for FPGAs have not previously been explored, other research projects have evaluated aspects of the problem in different contexts. A hard NoC version of a strictly time-multiplexed network appeared in Francis and Moore [9]. In Kapre [5], an FPGA soft NoC was evaluated that can be used for either time-multiplexed or packet-switched routing, although not at the same time. The hybrid router does not support multi-fanout routes. Finally, an effort to combine a standard packet-switched router with time-multiplexing circuitry for a multi-core microprocessor was described in Yin *et al.* [2]. This work differs from our implementation in several important aspects. First, FPGA NoC routers are much simpler than their fixed-function multi-core counterparts, motivating more aggressive NoC optimization. Second, FPGA circuits tend to have persistent time-multiplexed communication patterns rather than the time-varying patterns of multiprocessor communication.

B. Characteristics of FPGA Hard NoCs

For this work, we enhance an existing, well-documented hard FPGA NoC router [1] [6] to support hybrid traffic. Simple FPGA hard NoCs typically incorporate a minimum of virtual channels per port (e.g. 2) and output buffering is limited to a pipeline register per virtual channel. Switch and output virtual channel allocation logic is generally simple enough to be completed in a single clock cycle. This simplicity limits latency through the router to two clock cycles of approximately 1 GHz in 45 nm technology.

Since, unlike microprocessors, soft FPGA IP cores typically cannot reorder packets, the virtual channel (e.g. 0 or 1) used by all flits in a packet must remain the same for each router along the source-destination path [1]. This consistency ensures that all flits for a packet remain in order throughout its journey. The ordering restriction also implies that packets must be routed using predictable routing patterns, such as DoR, so that the path taken by all flits in a packet is the same and deterministic. Finally, FPGA NoCs are best suited to transfer streams of data long distance on the FPGA die (e.g. one-third of the device or greater) due to the overhead required to inject data into the NoC and extract it. Communication between neighboring soft FPGA IP cores can be easily handled using the segmented wiring and flip flops found in the FPGA fabric.

C. Limitations of Packet-Switched FPGA-based Hard NoCs

Previous work [10] has shown that packet-switched FPGA hard NoCs are very effective for bursty traffic, especially if data sources and destinations vary over time in an unpredictable fashion (e.g. a network switch implementation). However, these NoCs have limitations which can impede performance and increase dynamic power consumption for certain traffic patterns.

Flit congestion for some traffic patterns. The FPGA NoC need for deterministic packet transport paths (as opposed to adaptive routing used by many microprocessor NoCs) can lead to significant flit congestion since the deterministic patterns supported by the NoC may overlap. For example, Bitar *et al.* [10] showed that FPGA hard NoCs using DoR generally perform poorly for communication patterns in which communicating cores are on the chip boundaries. Although several solutions were proposed [10], they either did not ensure in-order packet arrival or require significant additional router hardware (e.g. routing tables with source/destination information within each router).

Broadcasting and multicasting. The implementation of broadcasting and multicasting packets from a single source to multiple destinations has not been addressed for hard FPGA NoCs. Previous approaches for packet-switched many-core NoCs require a special lookup table within each NoC router to support multicasting [11]. Additionally, the approach is limited to DoR routing. In our implementation for TDM multicasting, no additional hardware in the router is needed beyond the modest extensions needed for TDM.

Router buffer energy. Since congestion can be unpredictable, packet-switched NoCs include buffering to help store intermediate flits. Reading and writing this buffer requires significant dynamic energy. Since time-multiplexing alleviates congestion at compile time, buffering is not needed and buffer accesses can be avoided. After introducing our architecture, we describe how our implementation can address each of these concerns within the context of on-chip communication that can be predicted at compile-time.

D. Characteristics of Time-Multiplexed Networks

In addition to the benefits noted in the previous section, the use of FPGA NoC resources in a prescheduled TDM fashion places restrictions on FPGA soft cores and inter-core communication. As a result, we argue that a hybrid router which

supports both packet-switched and TDM routing is suitable. Characteristics of TDM networks include the following.

Precompiled bandwidth allocation. Time-multiplexed communication works best when the source-destination pairs and required bandwidth of inter-core communication are known at compile time and do not change substantially during design execution. TDM pipelines data at the maximum clock frequency of the network so no in-network buffering is needed.

Source and destination core data rates. TDM data transfer limits the source-core NoC data insertion rate to be, at most, the rate specified at compile time. Likewise, destination cores must consume or buffer data outside the NoC at the predetermined source data generation rate or transmitted data will be lost. If a source inserts data at a slower than expected rate, some NoC bandwidth could be wasted. In our hybrid NoC implementation, router time slots that have been allocated for time-multiplexed communication but do not have time-multiplexed data available can be used to transfer packet-switched data instead.

Context memory. A limitation of time-multiplexed communication is the need for a context memory that specifies the router configuration on a cycle-by-cycle basis [12]. When hybrid or TDM-only routing is used, the context memory consumes dynamic power whether a data transfer takes place or not.

III. Hybrid NoC Implementation

Our implementation of a hard hybrid NoC for FPGAs augments a two pipeline-stage packet-switched router and soft IP core interface that have been optimized for embedding within an FPGA [1] with a minimum of additional resources to support time-multiplexed routing. For any router or interface, if time-multiplexed routing is not needed, power is not applied to the isolated TDM resources and their effect on packet-switched routing is negligible. All time-multiplexed communication is scheduled at compile-time. Once configured, additional time-multiplexed bandwidth cannot be allocated at run-time.

A. Hybrid Router Operation

An overview of our router architecture appears in Figure 1. The context memory provides a cyclical schedule for time-multiplexed routing. The context memory contains one entry per time slot. The source for each TDM destination in a time slot is encoded in the memory. The encoding allows a single source flit to fan out to multiple destination ports in the same cycle. Input and output ports that are not used by TDM in a slot can be used for packet-switched routing. This sharing can occur if a slot is not allocated to TDM at all or if it is allocated to TDM but a valid TDM flit does not appear at the appropriate input port on the previous clock cycle. For example, in the figure for the first time slot, the north (N) and south (S) destination ports are reserved for TDM while the remaining ports can service packet-switched traffic. A counter is used to cycle through the slots in a repetitive fashion. For time-multiplexed routing, flits at the input ports bypass the input buffers (*VC1*, *VC2*) and are stored in the *bypass registers*

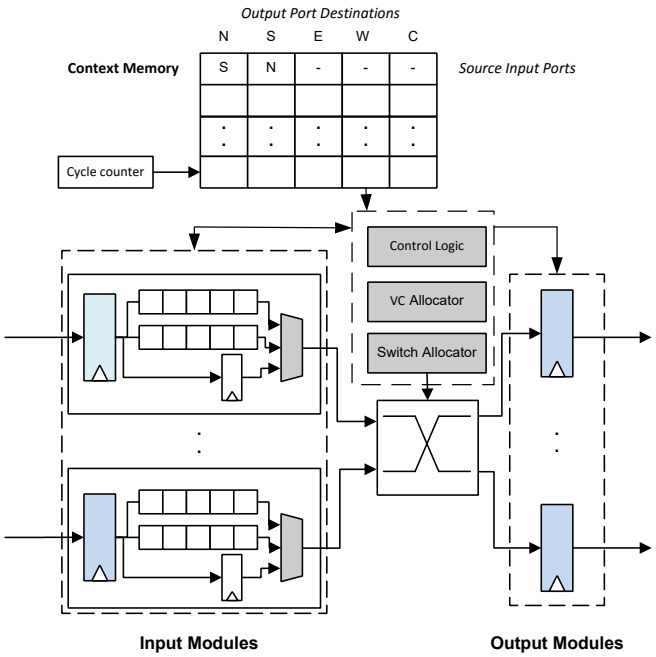

Fig. 1. Hybrid hard FPGA router

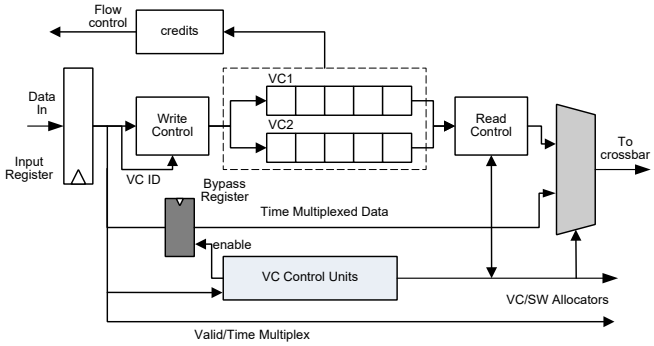

Fig. 2. Input port of a hybrid router

(Figure 2). A multiplexer is used to select crossbar input data from either the input buffers or the bypass register.

The operation of the router for packet switching involves the use of per-packet VC assignment and per-flit crossbar arbitration. When the header of a new packet arrives at an input port it is assigned to its designated VC. If a crossbar input or output is not used by TDM during a communication slot, switch allocation is performed to determine which input VC connects to which unused output port. TDM communication always takes priority over packet-switched communication. Our packet-switched switch and output port allocation techniques follow previous work [6].

Our pipelined hybrid router uses two pipeline stages and lookahead routing. In the following discussion, the processing of a single input port to output port path is described, although processing for all five input ports takes place simultaneously. At the start of the first pipeline-stage cycle, a data flit may be

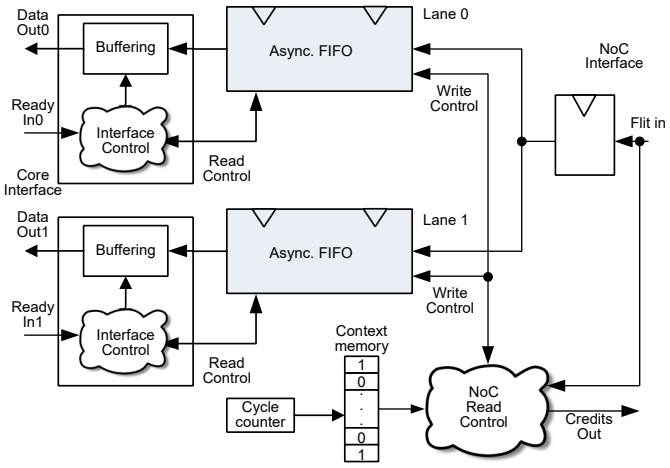

Fig. 3. Read core port interface for the hybrid NoC

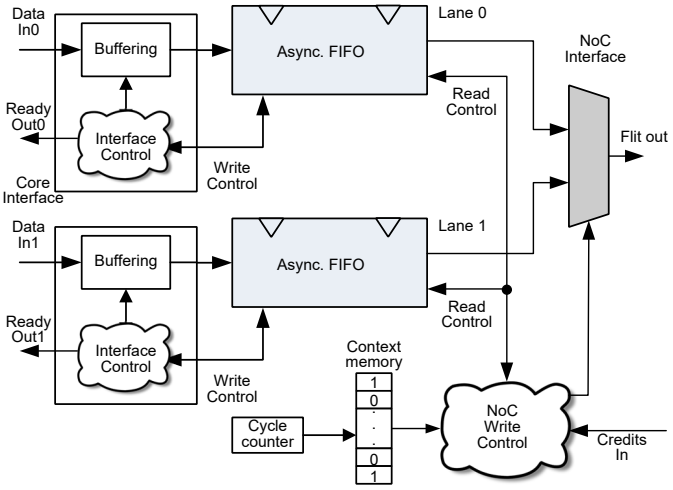

Fig. 4. Write core port interface for the hybrid NoC

located at the input register of each input port. During the first router cycle, the current entry in the context memory is fetched to determine which TDM input/output connections for the flits will be made in the next cycle. Round-robin switch allocation is simultaneously performed for packet-switched connections. Packet-switched flits are written into their respective VC buffers and TDM flits are written to bypass registers. If an expected TDM data value is invalid (as indicated by the flit's valid bit), the results of packet-switched crossbar allocation are used for the input-to-output port connection for the unused TDM output port on the next cycle. Otherwise, the packet-switched assignment is masked to allow for a TDM transfer to the designated port on the subsequent cycle. Each flit contains a bit which designates it as either a TDM or packet-switched flit. During the second pipeline cycle, data sweeps across the crossbar into the appropriate output register(s). Packet-switched flits are sourced from the input buffers while TDM flits come from the bypass registers.

B. Hybrid Core Ports

The use of hybrid routing requires adaptation of both the router and ports to FPGA soft IP cores (termed FabricPorts in [1]). This adaptation is complicated by several factors:

- The NoC is assumed to run at a different clock frequency than the cores. As a result, the ports must provide both synchronization and buffering.
- Since TDM data must be consumed within a fixed period, cores which transmit (receive) both TDM and packet-switched data require multiple write (read) *lanes* to (from) the adjacent NoC router. For example, if both TDM and packet-switched data are read from the NoC, separate lanes (one for each data type) per *port* are needed so that TDM data is readily available to the core on schedule.
- Control logic is needed in both read and write ports to coordinate flit transfer to the appropriate read or write lane depending on the flit type (TDM/packet-switched) and TDM time slot.

For our hybrid architecture, we augment packet-switched core ports to support TDM transfer while considering the issues noted above. In this data read example, the top lane in Figure 3 is used for packet-switched data and the bottom lane is used for TDM traffic. Either lane could be used for either type of traffic. If only one type is needed, the unused lane can be deactivated. Each lane provides a FIFO for inter-clock-domain synchronization. Buffering is provided at each interface to collect multiple flits prior to transfer to the FPGA core. A key aspect to the port is the context memory. Similar to the router context memory, a per-cycle indication of flit destination (e.g. 0 = packet-switched, 1 = TDM) is provided in the memory. For reading, on predetermined cycles valid flits are forwarded from the NoC to the TDM lane. On other cycles, valid flits are forwarded to the packet-switched lane. The size of the port context memory in number of entries is the same as the context memory for the routers. For the TDM lane, a core must read data from the lane at the same rate as it is read from the NoC to prevent data overwrite since backpressure control signals are not used. A *Ready in* signal is provided to the lane by the core indicating that valid data has been consumed from the NoC.

Similar to the hybrid read core port, one lane each is allocated for TDM and packet-switched traffic in the write port (Figure 4). The core writes a wide data value to the write buffer which is then forwarded to the FIFO synchronizer one flit at a time. The context memory is used to indicate which lane should forward a flit on a specific schedule clock cycle. TDM flits are given priority over packet-switched flits during designated cycles in the context memory.

C. Routing Algorithm

To preschedule TDM communication during design compilation, a routing algorithm has been developed which selects both source to destination router paths and context memory schedule time slots. A key aspect in the success of TDM routing is distributing TDM slots throughout the routing fabric.

1 The number of slots in the context memory is i_{max}. Specific schedule time slots range from 0 to $i_{max} - 1$. The delay through a router is d time slots;
2 **for** s_{start} = *0,* $s_{start} < i_{max}$, s_{start}++ **do**
3 Set start time slot $s = s_{start}$;
4 Empty queue, Q;
5 Clear visited routers in all time slots;
6 **if** *time slot* s *available in router,* r_{source}*, attached to source node* **then**
7 Push r_{source} and s into Q and mark as visited;
8 **end**
9 **else**
10 Continue;
11 **end**
12 **while** *destination* D *not reached and available feasible path* **do**
13 Pop router r and its time slot from top of Q;
14 If $r = D$, record cost C, and then end loop;
15 Identify neighbors of r on the shortest path to D;
16 Push router neighbors and time slot $(s + d)$ mod i_{max} onto Q in order of cost if not visited and available in time slot $(s + d)$ mod i_{max};
17 Mark router neighbors as visited;
18 **end**
19 **end**
20 Select successful path with lowest cost C;

Algorithm 1: Per-flit routing algorithm

If a context memory in a router becomes overfilled with TDM slots, packet-switched bandwidth through the router may become limited, creating a packet-switched bottleneck. Thus, our TDM routing attempts to locate shortest paths while balancing TDM slot usage in the routers.

Our TDM routing algorithm considers *streams* of data from a source core to a destination core that has a specific bandwidth. To achieve the required bandwidth, one or more flits must be transferred per context memory cycle (e.g. one complete pass through all context memory slots). This schedule then repeats to guarantee the required bandwidth for the stream. In the following, we consider the scheduled transfer of multiple flits for the stream. Each flit may traverse *different* routes, although since each follows a shortest path from source to destination and there is no NoC buffering, stream data is assured to arrive in order.

Before routing, streams are ordered by number of destinations, bandwidth requirements, and distance, in that order. For each flit in a stream in Algorithm 1, all possible shortest paths and starting time slots at the source core port are considered. Once a flit is injected into the NoC, consecutive time slots must be used for the route as the flit progresses towards the destination since no NoC buffering is allowed. The routing algorithm uses a queue to consider the next routers to add to the path. The cost C of using the router is inversely determined by the number of TDM slots that are already in use in the context memory schedule. Once the routes for all flits in one stream are determined, the next stream is routed.

1 Empty Q;
2 Select the next-closest destination, D_i, to the previously-routed path;
3 Identify all routers in the previously-routed path along a shortest path to D_i. For each router, identify its used TDM time slot for the route. Order routers by their closeness to D_i;
4 Push router and associated time slot in previously routed path which is closest to D_i into Q;
5 Perform steps starting at line 12 in Algorithm 1;
6 If route is unsuccessful, retry steps 4 and 5 with next-closest router to D_i;
7 Select overall lowest cost route;

Algorithm 2: Extension for per-flit multicast routing

Topology	64-node, 8 × 8 2D mesh
Technology	45 nm at 1.1V, 1.0 GHz
Routing	X-Y routing
Channel width	128 bits (16 bytes)
Packet size	4 flits
Context memory	8 entries
Virtual channels	2/port
Buffer size per VC	10 flits in depth

TABLE I
ROUTER PARAMETERS

Algorithm 1 for single destination routes is extended to route multicast data in Algorithm 2. Routing is first performed to the destination which is closest. Multicast routing progresses for each destination D_i after the first destination is reached.

IV. EXPERIMENTAL APPROACH

A. Hybrid NoC and Core Port Physical Modeling

The parameters used for all experiments (unless otherwise noted) are shown in Table I. A detailed evaluation environment was used to determine accurate area and energy consumption values for the hybrid router and core ports. A widely-used open-source packet-switched router [3] was adapted for our experimentation. The register-transfer level (RTL) code for this open-source router was modified to include our TDM additions. The design was synthesized in 45 nm technology using the NanGate Open Cell library[1] and the Synopsys Design Compiler vE-2010.12-SP5-2. The gate level netlist of the hybrid router circuitry was exercised to generate realistic circuit toggle rates. To determine per-component power, flits were repetitively inserted into all router input ports and forwarded to the output ports for 10,000 clock cycles. Timing simulation was performed using Modelsim to generate a toggle VCD file. This file was then input into the Synopsys Power Compiler to assess dynamic power consumption of the router components. The dynamic power evaluation was performed for both TDM and packet-switched routing operation.

[1]http://www.nangate.com/?page_id=22

	Packet-switched	Hybrid
Input buffers	91,944	92,275
Input buf. control	8,344	11,710
Allocator/arbitrator	4,516	4,445
Crossbar	2,912	2,977
Output module	7,061	7,318
TDM circuit	-	6,328
Total	114,777	125,053

TABLE II
ROUTER AREA (μm^2) AT 45 NM FOR PACKET-SWITCHED-ONLY AND HYBRID FPGA HARD NOC ROUTERS

	Packet-switched	TDM
Single input buffer	9.10	-
Bypass register	-	2.37
Single input module ctrl.	2.76	0.60
Single input reg.	1.60	1.60
Allocator/arbitrator	0.40	0.23
Crossbar	0.45	0.46
Inactive input regs	4.80	6.12
TDM circuitry	1.60	1.60
Single output reg.	1.74	1.96
Total	22.45	14.94

TABLE III
AVERAGE ROUTER POWER (MW) USING THE HYBRID ROUTER AT 1 GHZ FOR A SINGLE INPUT PORT TO SINGLE OUTPUT PORT DATA STREAM

Inter-router dynamic interconnect power per wire was determined by multiplying the 45 nm wire capacitance by V_{DD}^2 = $(1.1V)^2$ and scaling it by the activity factor of wires determined during simulation. To assess the power consumption of write and read core ports, transmit (TX) and receive (RX) ports were written in RTL Verilog, synthesized, and evaluated for dynamic energy using an approach which is similar to the one described above for routers.

B. Network Simulation Tools

To evaluate the performance of the hybrid FPGA NoC for a variety of traffic patterns, Booksim 2.0 [13] was used to assess packet latency and NoC dynamic power. The cycle-accurate simulator has built-in utilities to measure packet and flit latency, among other metrics. The simulator was modified to allow for hybrid TDM/packet-switched routing. Each router component and the inter-router wires were assigned dynamic energy consumption values determined via the post-synthesis simulation described in Section IV-A. These values were scaled by flit-level toggle rates determined during the Booksim simulations. The network was warmed up with enough packets to reach a stable state prior to results recording.

V. RESULTS

In our first experiment, we evaluate the area and power overhead of our TDM additions to an FPGA hard packet-switched router. The area in μm^2 of router components is shown in Table II. In this implementation, the TDM circuitry, including context memory and associated control logic, increases the router footprint by about 9%. Most of this area is consumed by the bypass registers and expanded multiplexers needed for the five input ports (about 4,500 μm^2 out of the 6,328 μm^2 TDM overhead). The 8×20 context memory and associated control requires 1,225 μm^2 of area. The area values per lane for the receive and transmit core ports are 18,151 μm^2 and 15,405 μm^2 for RX and TX, respectively without TDM circuitry. The 8×2 slot tables and associated control circuitry to allow for multiple RX and TX lanes are quite small (about 284 μm^2).

The dynamic power reduction of using the hybrid router exclusively in TDM versus packet-switched (PS) mode is substantial. Table III illustrates the dynamic power in the router consumed for each case if only one input and one output port are used. In this case, the other four input buffers are clock gated. Input buffer read-write access consumes nearly half of the router power. If all five input and output ports are used simultaneously, the disparity becomes greater (78.6 mW for PS versus 36.4 mW for TDM). A total of 45.5 of the 78.6 mW for PS is consumed in the input buffers. In hybrid mode, the TDM circuitry (primarily consisting of context memory accesses) continually consumes power even if packet-switched transfers are performed. This overhead could be removed if the hybrid router is used strictly in packet-switched mode. Note that the input registers continue to consume dynamic power even if valid data is not transferred on a cycle since the valid bit must be checked to verify data status. The per-lane dynamic power consumption of the RX and TX core ports are 16.8 mW and 11.5 mW, respectively. Added TDM control circuitry in the core port consumes about 0.3 mW.

A. Assessment of Hybrid Routing

In a series of experiments using our modified version of Booksim, three inter-core communication patterns for packet-switched and hybrid-switched routing were considered [13]. The patterns include 1) transpose (messages from (x, y) are sent to (y, x)), 2) bit-reverse (messages from the node labelled x are sent to the node whose label is *bit-reversed*(x), and 3) tornado (messages from (x, y) are sent to $(x + \frac{k}{2} - 1 \bmod k,\ y + \frac{k}{2} - 1 \bmod k)$ where k is the dimension of x and y. Latency values are directly reported by Booksim. Dynamic power was determined by Booksim by scaling the router, core port, and wiring dynamic energy determined via gate-level circuit simulation by activity.

In an initial Booksim experiment, we consider latency and dynamic power under different packet insertion rates[2] using the hybrid router if all traffic is forwarded using either packet-switched or TDM routing. Figure 5 shows that for transpose and bit-reverse traffic patterns, TDM reduces latency. These patterns stress the DoR strategy used by packet switching, while TDM can use a broad range of routes determined at compile time to connect sources and destinations. The tornado pattern is less favorable to TDM since communication is evenly distributed across links and is well-suited to DoR. Figure 6 shows that as data throughput increases, the disparity between packet-switched and TDM NoC+core port power also increases. The dynamic power savings of TDM is primarily

[2]For comparative purposes we consider a TDM "packet" to be four flits, the same size as a packet-switched packet.

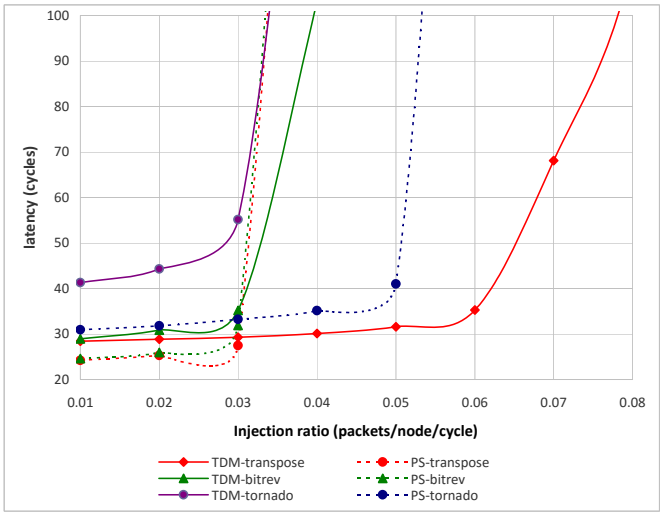

Fig. 5. Average packet (four-flit) latency for transpose, bit-reverse, and tornado routing patterns using only packet-switched or TDM routing.

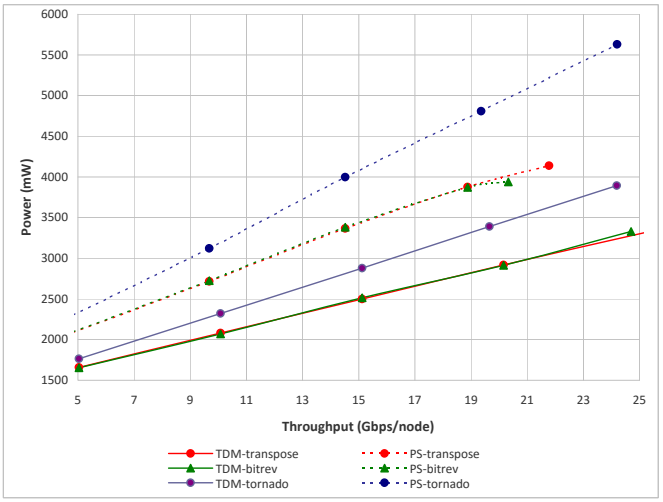

Fig. 6. Average NoC and core port dynamic power for transpose, bit-reverse, and tornado routing patterns using only packet-switched or TDM routing.

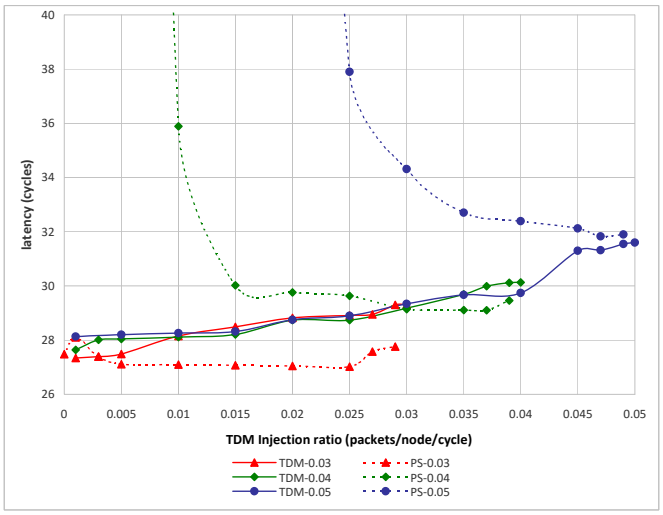

Fig. 7. Average NoC latency for different amounts of TDM traffic for transpose under fixed overall packet insertion ratios of 0.03, 0.04, and 0.05 packets/node/cycle. Note that PS traffic decreases as TDM traffic increases.

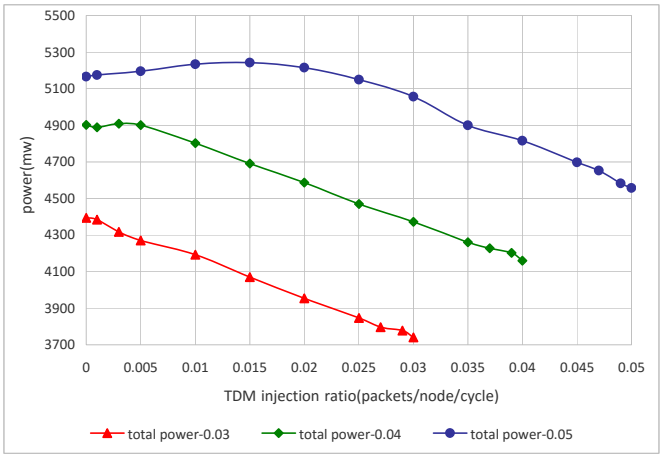

Fig. 8. Average NoC power for different amounts of TDM traffic for transpose under fixed overall packet insertion ratios of 0.03, 0.04, and 0.05.

due to the elimination of input buffer accesses in the router. It should be noted that for these experiments only one lane per RX and TX core port was needed since all traffic is either packet-switched or TDM. The second lane was clock gated.

In a second set of experiments, packet-switched and TDM routing were combined in the same simulation for the transpose pattern under total packet insertion rates of 0.03, 0.04, and 0.05 packets/node/cycle. Figure 7 shows PS and TDM latency as the total insertion rate stays constant but the TDM insertion rate increases from 0 to 0.05. In most cases, the TDM latency is less than PS, except for the 0.03 total insertion rate. Here, PS latency is lower since additional latency is present in the TDM TX core port. A flit may be ready to be sent immediately but must wait several cycles for its TDM slot to appear in the context memory. In contrast, a PS flit could be sent immediately. For higher PS insertion rates, increased PS congestion has a greater effect than the TDM TX delay. Figure 8 shows that the overall NoC+core port dynamic power decreases as the fraction of TDM traffic increases.

B. Comparisons to Packet-Switched Multicast

As mentioned in Section II-C, TDM routing inherently supports multicasting since a row in the context memory can indicate the same input port for multiple output ports. Although multicasting can be integrated in a packet-switched router, an additional CAM which indicates the output ports for each input is needed [11]. To assess this overhead, we used CACTI 4.1[3] to estimate the size and dynamic power consumption of a 8×9 CAM. These values were determined to be 625 μm^2 and 2.7 mW, respectively[4]. In contrast, TDM

[3]http://www.hpl.hp.com/research/cacti/

[4]The values for the whole packet switched router from Tables II and III are 114,777 μm^2 and 22.45 mW, respectively.

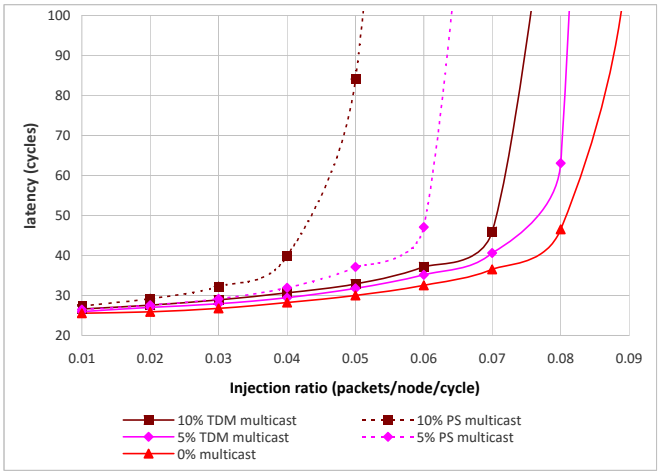

Fig. 9. Average packet latency for increasing amounts of four-destination multicast traffic. Packet-switched multicast requires the use of a separate packet for each destination

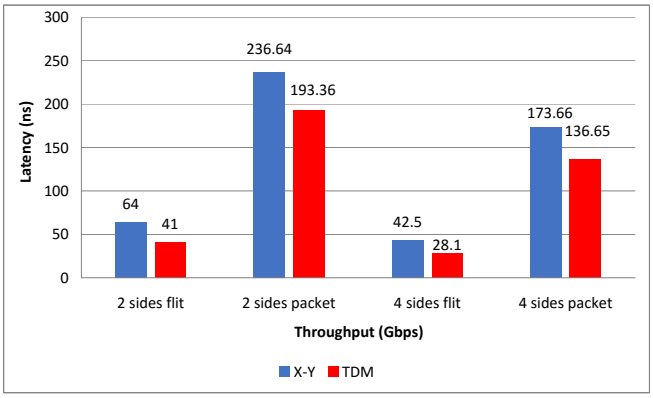

Fig. 10. TDM-only comparison versus packet-switching for average flit and packet latency. Single-source, single-destination traffic patterns [10] were used.

requires no additional multicast circuitry.

Figure 9 illustrates the benefits of TDM multicasting for various levels of multicast traffic. Multicast traffic for PS is transmitted as multiple copies of individual source-destination traffic. In the experiment, multicast packets are sent to four different destinations at least three nodes away from the source. All other traffic is packet-switched in a uniform random distribution pattern. TDM clearly leads to reduced average packet latency as the amount of multicasting increases.

C. Comparison to Single-Source, Single-Destination Routing

In a final experiment, we compare our hybrid router against a packet-switched router for single-source, single-destination routing patterns [10]. For this experiment, we use 64 routers in an 8×8 grid with 64-bit flits and a 10 Gbps overall data insertion rate. Data injection is distributed evenly across data sources. Sources and destinations are located either around the perimeter of the FPGA (4-sided) or along two parallel sides (2-sided). Permutation traffic (e.g. the top, left node sends data to the bottom, right node) is used to stress the network. Our results for flit and packet latency versus packet-switched only routing (Y-X) are shown in Figure 10. The use of TDM to avoid DoR congestion allows for a latency reduction. It should be noted that the router application in [10] can support variable packet sizes and an uneven packet insertion rate across sources, aspects that were not considered in this experiment. Traffic with these time-varying characteristics are best supported with a packet-switched framework.

VI. Conclusions and Future Work

In this paper, a hard FPGA NoC that can support both TDM and packet-switched routing has been described. Experiments show that TDM routing can reduce NoC dynamic energy consumption for traffic patterns that traditionally perform poorly with packet-switched dimension-ordered routing. We demonstrate that TDM can also natively support multicasting. In the future, we will consider FPGA core placement techniques that can assist both packet-switched and TDM routing. By better placing cores, DoR bottlenecks can be better avoided.

Acknowledgments

This research was supported by a grant from Xilinx Corporation. The authors thank Mohamed Abdelfattah for providing useful insights for this work.

References

[1] M. Abdelfattah, A. Bitar, and V. Betz, "Take the highway: Design for embedded NoCs on FPGAs," in *Proc. FPGA*, Feb. 2015, pp. 98–107.

[2] J. Yin, P. Zhou, S. S. Sapatnekar, , and A. Zhai, "Energy-efficient time-division multiplexed hybrid-switched NoC for heterogeneous multicore systems," in *Proc. IPDPS*, May 2014, pp. 293–303.

[3] D. Becker, "Efficient microarchitecture for network-on-chip routers," Ph.D. dissertation, Dept. of Elect. Eng., Stanford Univ., Aug. 2012.

[4] M. Papamichael and J. C. Hoe, "CONNECT: re-examining conventional wisdom for designing NoCs in the context of FPGAs," in *Proc. FPGA*, Feb. 2012, pp. 37–46.

[5] N. Kapre, "Marathon: Statically-scheduled conflict-free routing on FPGA overlay NoCs," in *Proc. FCCM*, 2016, pp. 156–163.

[6] M. Abdelfattah and V. Betz, "Networks-on-chip for FPGAs: Hard, soft, or mixed?" *ACM TRETS*, vol. 7, no. 3, Aug. 2014.

[7] R. Gindin, I. Cidon, and I. Keidar, "NoC-based FPGA: Architecture and routing," in *Proceedings of First International Symposium on Networks-on-Chip*, 2007, pp. 253–264.

[8] M. Abdelfattah and V. Betz, "Power analysis of embedded NoCs on FPGAs and comparison to custom buses," *IEEE TVLSI*, vol. 24, no. 1, pp. 165–177, Jan. 2016.

[9] R. Francis and S. Moore, "Exploring hard and soft networks-on-chip for FPGAs," in *Proc. FPT*, Dec. 2008, pp. 261–264.

[10] A. Bitar, J. Cassidy, N. E. Jerger, and V. Betz, "Efficient and programmable Ethernet switching with a NoC-enhanced FPGA," in *Proc. ANCS*, Oct. 2014, pp. 89–100.

[11] N. Jerger, L.-S. Peh, and M. Lipasti, "Virtual circuit tree multicasting: A case for on-chip hardware multicast support," in *Proc. ISCA*, Jun. 2008, pp. 229–240.

[12] A. Laffely, J. Liang, P. Jain, W. Burleson, and R. Tessier, "Adaptive systems on a chip (aSoC) for low power signal processing," in *Proc. Asilomar Conf. on Signals, Systems, and Comp.*, 2001, pp. 1217–1221.

[13] N. Jiang, J. Balfour, D. Becker, W. Dally, G. Michelogiannakis, and J. Kim, "A detailed and flexible cycle-accurate network-on-chip simulator," in *Proceedings of the IEEE Symposium on Performance Analysis of Systems and Software*, Apr. 2013, pp. 86–96.