FPGA Optimized Packet-Switched NoC using Split and Merge Primitives

Yutian Huan[1], André DeHon[2]

Electrical and Systems Engineering, University of Pennsylvania
200 S. 33rd Street, Philadelphia, PA, USA 19104
[1] blithe.huan.penn@gmail.com, [2] andre@ieee.org

***Abstract*—Due to their different cost structures, the architecture of switches for an FPGA packet-switched Network-on-a-Chip (NoC) should differ from their ASIC counterparts. The CONNECT network recently demonstrated several ways in which packet-switched FPGA NoCs should differ from ASIC NoCs. However, they also concluded that pipelining was not appropriate for the FPGA switches. We show that the Split-Merge switch architecture is more amenable to pipelining on FPGAs, achieving 300MHz operation—up to three times the frequency and throughput of the CONNECT switches—with only 13–37% more area. Furthermore, we show that the Split-Merge switches are at least as efficient at routing traffic as the CONNECT switches, meaning the 2–3× frequency translates directly into two to three times the application performance.**

I. Introduction

Many applications on today's large-scale, platform FPGAs demand high bandwidth, dynamic communication (*e.g.* multiprocessors [1], CoRAM [2], sparse graph processing [3], dynamic reconfigurable accelerators [4]). Natively, today's FPGAs provide high dedicated bandwidth with configured interconnect, but only modest dynamically shared bandwidth with hardwired buses [5]. As a result, it is increasingly useful to configure a scalable, high-bandwidth, dynamically shared interconnect, such as a *Packet-Switched* (PS) *Network-on-a-Chip* (NoC), as an overlay network on top of the FPGA configured interconnect and logic.

While PS NoCs are well developed for implementation on ASICs [6], [7], the different cost structure of FPGAs mean that NoCs optimized for ASICs may not be the best solutions for FPGA NoCs. In particular, the larger delays associated with configurable interconnect coupled with the high and predetermined ratio of registers to logic suggests that FPGA NoCs should be pipelined more heavily than ASIC NoCs. A number of recent efforts have shown how to build PS FPGA NoCs [8], [4], [9], but much work remains to develop a systematic understanding of PS NoC design for FPGAs. Two FPGA NoC designs that have begun to depart from the ASIC NoC designs are CONNECT [10] and Split-Merge-based PS NoC [11].

CONNECT [10] started with a standard Virtual Channel (VC) ASIC NoC [6], [12] and observed the difference in wire availability, buffer cost, and pipelining should drive FPGA NoCs to use wider channels, fewer buffers, and less pipelined designs. It is, however, unclear whether the low pipelining conclusion is driven more by the the FPGA architecture or by the ASIC-style NoC architecture from which CONNECT is derived. The paper [10] performed systematic comparisons between ASIC NoC designs and their FPGA NoCs optimizations, showing quantitatively the benefits of their FPGA-optimized NoC designs.

The Split-Merge PS NoC [11] is a more radical departure from ASIC NoCs. Rather than starting with an ASIC NoC design, the Split-Merge PS NoC started with the design of highly pipelined primitives: split units, merge units, and queues. These primitives could be individually tuned for high frequency operation, then composed with additional pipelining as necessary to maintain high throughput. However, other than showing their ability to achieve higher bandwidth, previous work on Split-Merge PS NoCs did not perform direct comparisons to other FPGA or ASIC PS NoCs.

In this paper, we contribute to the systematic understanding of FPGA PS NoCs by comparing CONNECT and Split-Merge PS NoCs, both designed in Bluespec [13] and mapped to a Xilinx Virtex 6 Platform FPGA. Notably, we show that the Split-Merge design can be pipelined to achieve two or three times the frequency of the CONNECT design without sacrificing latency, significant area, or any net performance under congestion. As a result, the Split-Merge PS NoC achieves three times the performance of the CONNECT design on application workloads in roughly the same footprint.

Our novel contributions include:

- Split-Merge PS NoC Design in Bluespec that supports operation up to 300MHz on a Virtex 6 (Secs. III and IV)
- Quantitative comparisons of resource requirements and timing of CONNECT and Split-Merge PS NoC (Sec. IV)
- Quantitative characterization of both CONNECT and Split-Merge PS NoC on application traffic (Sec. V)

We start by reviewing PS NoC background including both CONNECT and Split-Merge PS NoCs in the next section.

II. Background

A. NoC Routers

This section discusses network topology, flow control methods and routing algorithms that characterize a typical NoC.

Topology Topology represents the way switches and processing elements (PE) interconnect with each other in the

978-1-4673-2845-6/12/$31.00 © 2012 IEEE

network. In this paper, we focus on the 2D mesh since it is the most commonly used and characterized topology and our primary goal is to compare switches rather than topology.

Routing Algorithms The role of a routing algorithm is to choose the proper path for each packet to achieve high network throughput. Due to the limited on-chip logic resources and more plentiful wiring, it is less beneficial to spend logic for the sophisticated routing algorithms used for off-chip routing. Deterministic routing, such as Dimension Ordered Routing (DOR) [14] is widely adopted for NoCs because of its simple routing logic. DOR on a mesh routes the packet along the X dimension first then the Y dimension. The West-Side First (WSF) routing algorithm allows the router choice to avoid local congestion while remaining deadlock free [14].

Fully adaptive routing algorithms often use Virtual Channels (VCs) to allow more flexibility to avoid local congestion while remaining deadlock free [15]. VCs share a physical wire between multiple logical channels, each with its own buffers. Since the VCs block independently, this can reduce the impact of Head-of-Line (HoL) blocking, prevent low priority traffic from blocking higher priority traffic, and create acyclic channel graphs while still allowing rich adaptive routing [16], [17]. VC routing algorithms tradeoff more complex routing logic and higher buffering requirements to achieve better utilization of physical wires. Since wires are pre-allocated and richly populated in modern FPGAs, this tradeoff may be less attractive than in ASIC designs.

Flow Control Most packet-switched NoCs adopt wormhole routing which breaks packets into smaller segments called flits. Both the sender and receiver switch must follow a flow control protocol. VC switches usually adopt credit-based [6] flow control. For non-VC switches, such as the split-merge routers, the valid/backpressure [18] flow control is simpler and also provides guaranteed flit transmission. CONNECT uses valid/backpressure flow control in its simpler peek scheme.

B. How FPGAs differ from ASICs

While FPGAs are generally slower and larger than ASICs [19], they are not uniformly larger or slower. Resources are pre-allocated among logic, registers, interconnect, and memory in the FPGA, meaning area is not interchangeable as in the ASIC. Modern FPGAs have a mix of coarse-grained, specialized resources (*e.g.* clocks, embedded memories, carry chains) and fine-grained logic, with the specialized resources offering lower latencies and more compact areas than the fine-grained logic. As a result, FPGAs and ASICs have significantly different cost structures, and it may be possible to significantly improve performance and resource utilization by customizing the architecture to the FPGA [10].

Delay and Pipelining The large area of programmable switches in FPGAs mean longer interconnect paths where route delays often dominate the critical path. This discourages large and complicated switches, making high-frequency, simple routing logic more attractive than complex designs running at lower frequencies. Furthermore, FPGAs pre-allocate flip-flops at the logic gate level. As a result, FPGAs usually have abundant flip-flop resources that make increased pipelining an effective technique to improve overall system performance. Since flip-flops are not pre-allocated in ASICs, they effectively have a larger area cost when inserted into ASIC designs.

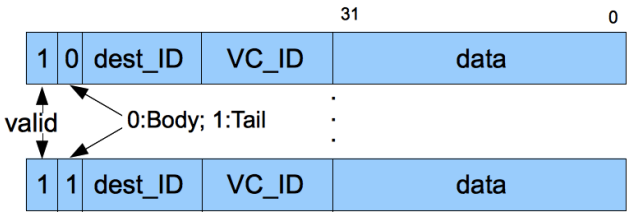


Fig. 1: Packet Format of CONNECT Network

C. CONNECT

The CONNECT router is a simple one pipeline-stage, VC-based router targeting the FPGA platform [10]. The design philosophy is to achieve minimal resource utilization while maintaining a moderate operation frequency. CONNECT uses distributed RAMs to implement look-up table based routing [6]; the default routing algorithm is DOR. CONNECT supports multiple VCs for the purpose of segregating packets, avoiding deadlock, and reducing HoL blocking effects. CONNECT also supports a Virtual Link (VL) option that guarantees continuous transmission of packets at receiving end-points. The implementation of VLs slightly increases router complexity but greatly simplifies the PE-router interface. CONNECT attaches header data to every flit, as shown in Fig. 1, to exploit the abundance of wires on the FPGA to simplify routing. The CONNECT routers support both credit-based flow control and a simpler "peek" (valid/backpressure) flow control. The valid/backpressure approach reduces the resource requirements and has the same performance under the situations when the round-trip communication delay is low.

The CONNECT designers concluded that FPGA NoCs should be pipelined less than ASIC NoCs, arguing primarily in terms of the longer wire delays and ignoring the abundance of registers in FPGAs to support pipelining. Consequently, CONNECT employs a single stage Mesh switch, noting: "*...it becomes impossible to further subdivide into balanced finer pipeline stages due to the quantization effects of the underlying realization structures and the difficulty in controlling physical details like logic placement, wiring routing, and driver sizing*" [10]. We contend that the problem is not the underlying FPGA resources, but rather the ASIC-style NoC architecture that they started with that impedes productive pipelining. Consequently, it is necessary to redesign the NoC switch architecture more significantly to better accommodate FPGA designs.

D. Split-Merge PS NoC

Unlike CONNECT routers that resemble a traditional, VC-based router and have only one pipeline stage, the split-merge design took a different approach. The typical split-merge router architecture is shown in Fig. 2. The router is constructed using primitives called splits and merges [11] (Sec. III-A). Constructing routers under different topologies is

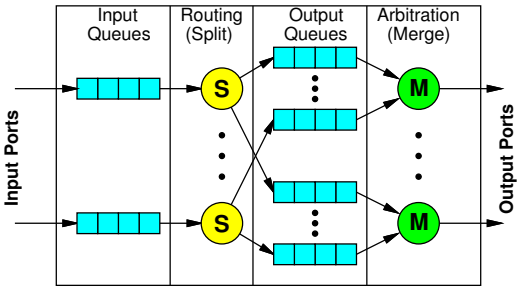


Fig. 2: Typical Split-Merge Switch Architecture

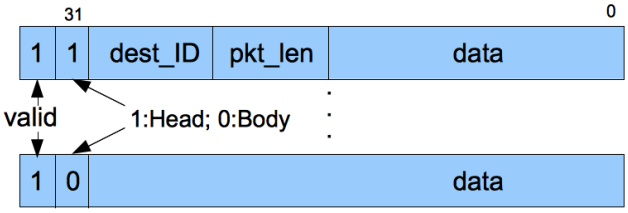


Fig. 3: Packet Format of Split-Merge Network

simply a matter of connecting these two primitives in different patterns. A big advantage of this design paradigm is that the decentralized implementation of routing and arbitrating functions greatly reduced the logic complexity for each primitive. By inserting FIFO queues at each input port of both primitives as buffers, the router can be easily pipelined at the level of the primitives.

Packet Format Split merge also uses wormhole routing of flits. Each packet contains one head flit and several body flits. The head flit distinguishes itself from body flits by setting its highest bit to one. It contains extra bits for routing information and the number of flits for the packet (pkt_len). This number is used to handle flexible packet length.

Flow Control A simple but effective switch-to-switch valid/backpressure flow control method is used to control the transmission of flits. This flow control resembles the "peek" flow control for CONNECT and uses one bit (back pressure) signal to indicate whether there is enough space for each downstream buffer. Since there may be one or more cycles of round-trip delay (depending on channel pipeline level) between adjacent primitives, the downstream buffer must produce the backpressure signal early enough so that there are still enough buffer space to hold all following flits due to pipeline delays in the backpressure status transmission.

III. Split and Merge Switching Primitives

This section describes our implementation of split-merge primitives and mesh switches using split-merge primitives.

A. Primitive Operation

Primitives are shown in Fig. 5. Buffers are implemented as FIFO queues using shift registers (SRLs) to take advantage of Xilinx FPGA's built-in SRL resources [20]. The split primitive looks at flit headers and routes input packets by sending them to the appropriate output port. The merge primitive interleaves

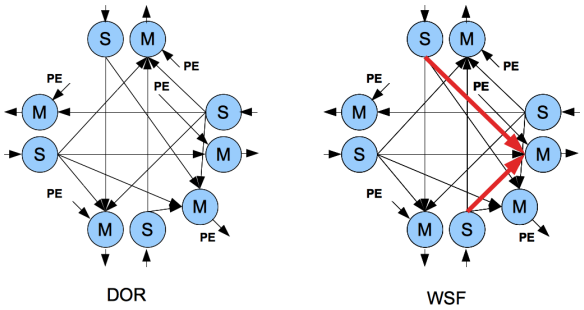


Fig. 4: Structure Diagram of DOR and WSF Mesh Switches

messages from different input ports to the same output port while maintaining packet completeness and trying to reach 100% throughput. Split and merge are each pipelined blocks. We explore giving them each one or two pipe stages to operate.

B. Constructing Switches from Split and Merge Primitives

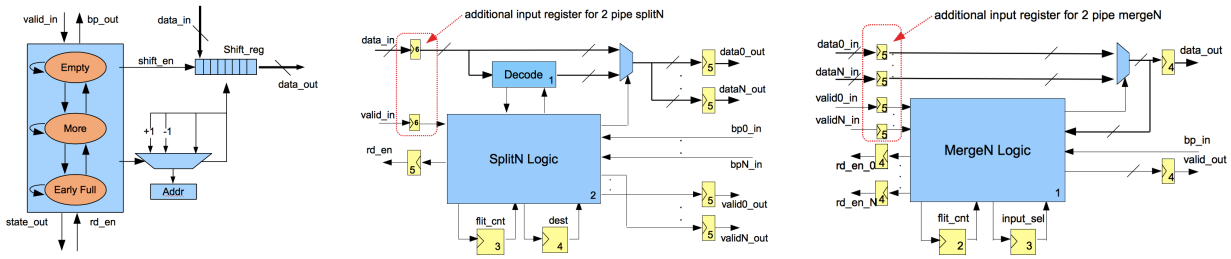
In a 2D mesh topology, each switch in the network is connected with a local PE and has direct connection with its neighbors on X and Y directions. This roughly makes the mesh switch a 5×5 crossbar. Fig. 4 (left) shows the structure diagram of the DOR switch. Splits and merges inside a single switch do not need to be fully connected because certain connections are never used by DOR routing algorithm. The structure of the WSF mesh switch is shown in Fig. 4 (right). As is analyzed in [14], a mesh network is free from deadlock if certain turns are prohibited to prevent any potential circular channel dependency. Compared with DOR switches, the WSF switches has two more internal interconnections between split and merge primitives that allow packets coming from Y directions to turn east. This gives splits at certain directions the flexibility to make route decisions for packets adaptively. All paths through the DOR and WSF mesh switches traverse one split and one merge primitive. As a result, it takes two or four cycles to traverse each mesh switch.

IV. Implementation and Comparison

We implement our split-merge network on a Xilinx Virtex 6 FPGA (XC240T-1) to demonstrate the effectiveness of our design. We generate detailed synthesis result for split-merge primitives and compare resource utilization and timing result with CONNECT switches after mapping and Place-and-Route (PAR) procedures. We obtain CONNECT networks from the public web distribution [21].

A. Primitives

We compile our Bluespec primitive modules into Verilog and use Xilinx ISE 13.2 XST tools to obtain the synthesis result for both area and timing results. All the synthesized switches have 32-bit data channel width with a buffer depth of 16. As can be seen from Table I, buffers are synthesized into SRLs that occupy SLICEM resources [22]. Merge logic is larger than split logic or buffers because the arbitration among various input channels requires more input information. All stand-alone primitives are able to achieve an operation



(a) FIFO Queue Buffer (b) N-output-Split (c) N-input Merge

Fig. 5: Split-Merge Primitive Schematics

TABLE I: Synthesis Results for w=32 Split-Merge Primitives

	Regs	Logic (6-LUT)	Memory (SRL16)	Max Freq. (MHz)
Buffer (16)	10	19	32	581
Split2	36	41	0	726
Split3	38	45	0	577
Split4	39	50	0	683
Merge2	39	94	0	351
Merge3	42	153	0	370
Merge4	43	221	0	354

frequency of over 350MHz. When composed using only a single pipeline stage per split or merge, wiring delays greatly reduced the overall performance. With a second pipeline register per stage, we are mostly able to place these routing delays into a separate pipeline stage (Table II).

B. Mesh Switch Comparison

To make a head-to-head comparison between the split-merge switch and the CONNECT switch, we built both types of switches for the mesh topology which has five input/output ports, N, S, E, W, and local PE. We use the peek flow control for the CONNECT switch because it matches the back pressure flow control, occupies the least area, and achieves the highest operating frequency. The VL is enabled to guarantee a simpler PE to switch interfacing and buffering mechanism that matches the functionality of the split-merge design. We implemented the CONNECT switches with 2 and 4 VCs under the buffer depth of 16 and data width of 32 bits. We also implemented the split-merge switches applied with DOR and WSF routing algorithms. Since the CONNECT switch has separate flow control wires to transmit routing information in addition to the data paths (See Fig. 1), packets of the same number of flits and payload data width in the CONNECT network use more physical wires than the split-merge network. Since the split-merge network puts the routing information in the head flit of a packet (See Fig. 3), it may end up needing more flits to transmit a fixed data payload. To provide a more direct comparison taking into consideration this additional packet overhead, we also characterize split-merge switches with 42-bit channels to model the extra 10b of routing information that CONNECT is adding to its flits. This is an overestimate for split-merge switches; only in the case where the payload is a single flit of exactly 32 bits do

TABLE II: Map & Post-PAR Report for Split-Merge and CONNECT Switches on XC6VLX240T-1

		Area			Timing		
		Regs	Logic (LUTs)	Mem.	Constrain (ns)	Cycle (ns)	Freq. (MHz)
CONNECT 1 clock	2VCs; 32bit	635	1396	166	9.0	9.6	104
	4VCs; 32bit	1265	1926	288	10.0	10.9	92
split-merge 1 pipe (2 clock)	DOR; 32bit	541	1449	336	4.5	4.5	220
	DOR; 42bit	641	1686	462	4.5	4.6	219
	WSF; 32bit	579	1839	400	4.6	4.6	217
	WSF; 42bit	679	2139	550	4.6	4.6	216
split-merge 2 pipe (4 clock)	DOR; 32bit	1262	1157	336	3.3	3.3	303
	DOR; 42bit	1572	1302	462	5.0	5.0	201
	WSF; 32bit	1454	1491	400	3.3	3.4	298
	WSF; 42bit	1804	1666	550	4.7	4.7	213

split-merge switches need this full 42-bit width to be a direct match with CONNECT. Under typical situations, the most comparable split-merge network will lie between the 32-bit and 42-bit case. We show the extremes to bracket the range.

C. Placed Mesh Switch

Table II shows the post-PAR static timing report for both types of switches targeting Xilinx Virtex-6 LX240T speed-grade -1 FPGAs. Overall, for placed mesh switches, the split-merge switches with one pipeline stage per split or merge are able to achieve more than twice the speed of the CONNECT switch; with two pipeline stages, they are able to achieve three times the speed. The DOR split-merge switches are between 13–37% larger than the 2 VC CONNECT switches.

V. Performance Comparison

Since the Split-Merge PS NoC is a more radical departure from traditional ASIC NoC designs, it is necessary to understand how its architectural changes impact its ability to deal with traffic congestion in the mesh network. Consequently, in this section, we perform head-to-head routing comparisons between the CONNECT and Split-Merge PS NoCs.

A. Experimental Setup

In this section, we present the cycle-accurate simulation methodology for both the split-merge network and the VC-based CONNECT network.

Dummy PE We designed dummy PEs with the same interface as normal PEs to inject traffic into the network and

monitor network performance. Each dummy PE includes a memory pre-loaded with a complete set of messages. During the simulation, PEs read every entry from the memory, translate it into a packet, and inject corresponding flits into an infinite buffer connected to the network. This trace-based traffic simulation method has the flexibility of providing traffic of any kind because all messages are generated completely offline. We assign enough space to each PE memory to hold tens of thousands of entries, which is capable of driving the simulator for over 100,000 cycles. This provides a long enough period for the warm-up and guarantees a stabilized operation environment for evaluation.

Since the CONNECT network available from [21] is in Verilog, we first construct our test bench and dummy PE in Bluespec and then compile the Bluespec module into Verilog. We use the same control method to inject packets for both networks and tune the simulation parameters of CONNECT switches to achieve good performance under similar area constraints as the split-merge network (*e.g.* number of VCs (v = 2 and 4) and buffer depth of 16 for traffics with 8 flits/pkt). The RTL-level simulation is performed under the Xilinx iSim 13.1 simulation environment.

Application Traffic In addition to synthetic, random traffic, we use traffic workloads from a graph algorithm as real-world benchmarks to test network performance. The application traffic is the complete set of communication messages between nodes during Bellman-Ford shortest path computations mapped onto finite number of NoC PEs. The original computation is based on a Barrier Synchronized Parallel model that divides computation into separate steps [3]. Since the overall run time for the whole computation depends on each step period, the maximum number of cycles to route a single step is an important metric for performance evaluation. We use MLpart [23] and load-balanced PE assignment.

B. Results

In this section, we focus on 8×8 mesh networks composed by both the split-merge switches and the CONNECT switches with data width of 32 bits and buffer depth of 16. We configure both DOR and WSF split-merge switches. To show the effects of number of VCs on the network performance, we also simulate the CONNECT network with 2VCs and 4VCs. The routing algorithm for the CONNECT switch is DOR.

Fig. 6 shows the delay-load curve for the split-merge and the CONNECT networks under uniform random traffic. At low traffic injection rate, both 2VCs and 4VCs CONNECT network have smaller average cycle delay compared with the split-merge network. When the network is under light load, the average delay is determined by the number of switches that flits traverse after being injected into the network before reaching their destinations. The CONNECT network has the advantage of using one pipeline-staged router design which halves the required cycles compared with the single pipeline stage split-merge routers. However, since the single pipeline stage split-merge routers run at half the cycle time, the end-to-end latency is roughly the same. At higher traffic load, however, both the WSF and DOR split-merge networks outperform the CONNECT networks by having higher saturation injection rate. This suggests that the split-merge network is better at handling congestion under the same bisection bandwidth. This may be because the splits and merges have rich buffering inside each single switch that reduces HoL blocking effects at input buffers. We also see that the split-merge network with DOR routing out-performs the WSF routing under uniform random traffic, suggesting it is preferable to use the more compact DOR switch.

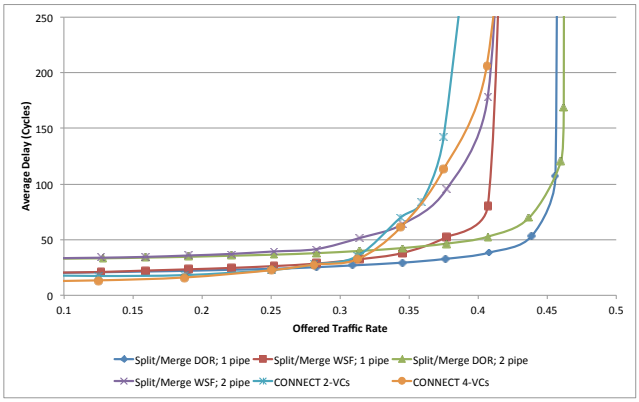


Fig. 6: Cycle Comparison between CONNECT and Split-Merge PS NoC on Uniform Random Traffic on 8×8 Mesh with 8 Flit Packets

Fig. 7 shows the network performance under the Bellman Ford traffic benchmarks. The height of each column represents the number of cycles required to route all messages in a single Barrier-synchronized traffic step. Although the split-merge network has two or three times the pipeline stages at each switch, the total number of cycles to route a single step is almost the same as the CONNECT networks. Fig. 8 shows the actual time elapsed (in nanoseconds) when both types of networks run at their maximum frequency (303MHz and 219MHz for split-merge networks and 104MHz for CONNECT). By taking advantage of the simpler logic and higher frequency, the split-merge networks routes the Bellman Ford traffic in half or one-third the time of the VC-based CONNECT network. This shows that the frequency advantage that the split-merge network gets from pipelining translates directly into higher application performance.

VI. Conclusions

We have reevaluated the split-merge paradigm for packet-switched NoC design on FPGA platforms and made head-to-head comparison with the VC-based CONNECT network. We succeed in building two- and four-staged, low latency switches for both the DOR and WSF routing algorithms. The implementation on a Xilinx Virtex 6 FPGA demonstrates the high performance of our network, showing that the entire system is capable of running at a maximum frequencies up to 300MHz, three times the frequency and throughput of the CONNECT network with only slightly more area. Evaluating the network performance of split-merge and CONNECT

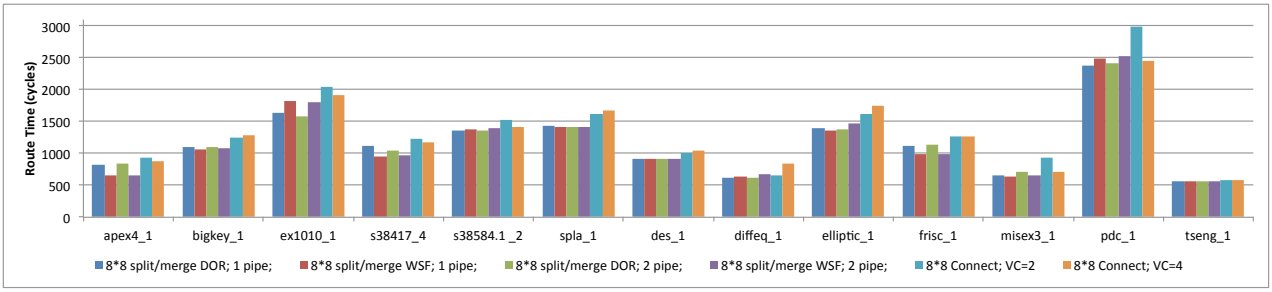


Fig. 7: Cycle Comparison between CONNECT and Split-Merge PS NoC on Bellman Ford Application Traffic on 8×8 Mesh

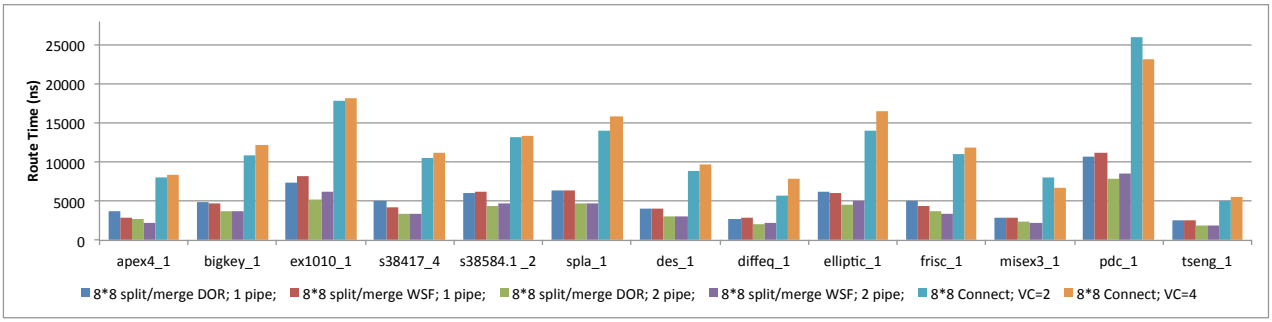


Fig. 8: Absolute Performance Comparison between CONNECT and Split-Merge PS NoC on Bellman Ford Application Traffic on 8×8 Mesh

networks under a common set of benchmarks on an 8×8 mesh, we see that our split-merge network routes the same amount of traffic three times as fast as the CONNECT mesh network. A source-level distribution of our designs is available http://ic.ese.upenn.edu/distributions/split_merge_fpt2012.

REFERENCES

[1] J. Wawrzynek, D. Patterson, M. Oskin, S.-L. Lu, C. Kozyrakis, J. C. Hoe, D. Chiou, and K. Asanović, "RAMP: Research accelerator for multiple processors," *IEEE Micro*, vol. 27, no. 2, pp. 46–57, 2007.

[2] E. S. Chung, J. C. Hoe, and K. Mai, "CoRAM: An in-fabric memory architecture for FPGA-based computing," in *FPGA*, 2011, pp. 97–106.

[3] M. deLorimier, N. Kapre, N. Mehta, and A. DeHon, "Spatial hardware implementation for sparse graph algorithms in GraphStep," *ACM TAAS*, vol. 6, no. 3, pp. 17:1–17:20, September 2011.

[4] T. Marescaux, V. Nollet, J.-Y. Mignolet, A. B. W. Moffat, P. Avasare, P. Coene, D. Verkest, S. Vernalde, and R. Lauwereins, "Run-time support for heterogeneous multitasking on reconfigurable socs," *INTEGRATION, The VLSI Journal*, vol. 38, no. 1, pp. 107–130, 2004.

[5] IBM, *CoreConnect Specification*, 1999. [Online]. Available: http://www3.ibm.com/chips/products/coreconnect/

[6] W. J. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. Morgan Kauffman, 2004.

[7] J. Duato, S. Yalamanchili, and L. Ni, *Interconnection Networks: An Enginering Approach*. Elsevier, 2003.

[8] Nallatech, "Simplifying communication across dsp networks," Programmable World, 2003, <http://www.mactivity.com/xilinx/pw2003/workshops/presos/wsa3_nallatech.pdf>.

[9] B. Sethuraman, P. Bhattacharya, J. Khan, and R. Vemuri, "LiPaR: A lightweight parallel router for FPGA based networks on chip," in *GLSVLSI*, 2005.

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

[11] N. Kapre, N. Mehta, M. deLorimier, R. Rubin, H. Barnor, M. J. Wilson, M. Wrighton, and A. DeHon, "Packet-switched vs. time-multiplexed FPGA overlay networks," in *FCCM*. IEEE, 2006, pp. 205–213.

[12] D. Becker, "Open source network-on-a-chip router RTL," https://nocs.stanford.edu/cgi-bin/trac.cgi/wiki/Resources/Router, April 2012, Stanford Concurrent VLSI Architecture Group.

[13] Bluespec, Inc., "Bluespec SystemVerilog." [Online]. Available: http://www.bluespec.com

[14] C. J. Glass and L. M. Ni, "The turn model for adaptive routing," in *ISCA*, 1992, pp. 278–287.

[15] T. Schonwald, J. Zimmermannand, O. Bringmann, and W. Rosenstiel, "Fully adaptive fault-tolerant routing algorithm for network-on-chip architectures," in *Proceedings of the 10th Euromicro Conference on Digital System Design Architectures, Methods and Tools*, 2007, pp. 527–534. [Online]. Available: http://dx.doi.org/10.1109/DSD.2007.62

[16] W. J. Dally, "Virtual-channel flow control," *IEEE Trans. Parallel Distrib. Syst.*, vol. 3, no. 2, pp. 194–205, 1992.

[17] R. Mullins, A. West, and S. Moore, "Low-latency virtual-channel routers for on-chip networks," in *ISCA*, 2004.

[18] A. DeHon, J. Adams, M. deLorimier, N. Kapre, Y. Matsuda, H. Naeimi, M. Vanier, and M. Wrighton, "Design Patterns for Reconfigurable Computing," in *FCCM*, April 2004, pp. 13–23.

[19] I. Kuon and J. Rose, "Measuring the gap between FPGAs and ASICs," *IEEE Trans. Computer-Aided Design*, vol. 26, no. 2, pp. 203–215, February 2007.

[20] E. Caspi, "Design Automation for Streaming Systems," Ph.D. dissertation, EECS Department, University of California, Berkeley, Dec. 16, 2005. [Online]. Available: http://www.cs.berkeley.edu/~eylon/phd/

[21] M. K. Papamichael and J. C. Hoe, "CONNECT: CONfigurable NEtwork Creation Tool," <http://users.ece.cmu.edu/~mpapamic/connect/>, 2012.

[22] *Virtex-6 FPGA Configurable Logic Block Users Guide*, Xilinx, Inc., 2100 Logic Drive, San Jose, CA 95124, February 2012, uG 364 <http://www.xilinx.com/support/documentation/user_guides/ug364.pdf>.

[23] A. Caldwell, A. Kahng, and I. Markov, "Improved Algorithms for Hypergraph Bipartitioning," in *Proceedings of the Asia and South Pacific Design Automation Conference*, January 2000, pp. 661–666.