Customizing Virtual Networks with Partial FPGA Reconfiguration

Dong Yin* , Deepak Unnikrishnan, Yong Liao, Lixin Gao and Russell Tessier
Dept. of Electrical and Computer Engineering
University of Massachusetts
Amherst, MA 01003, USA
{lgao,tessier}@ecs.umass.edu

ABSTRACT

Recent FPGA-based implementations of network virtualization represent a significant step forward in network performance and scalability. Although these systems have been shown to provide orders of magnitude higher performance than solutions using software-based routers, straightforward reconfiguration of hardware-based virtual networks over time is a challenge. In this paper, we present the implementation of a reconfigurable network virtualization substrate that combines several partially-reconfigurable hardware virtual routers with software virtual routers. The update of hardware-based virtual networks in our system is supported via real-time partial FPGA reconfiguration. Hardware virtual networks can be dynamically reconfigured in a fraction of a second without affecting other virtual networks operating in the same FPGA. A heuristic has been developed to allocate virtual networks with diverse bandwidth requirements and network characteristics on this heterogeneous virtualization substrate. Experimental results show that the reconfigurable virtual routers can forward packets at line rate. Partial reconfiguration allows for 20x faster hardware reconfiguration than a previous approach which migrated hardware virtual networks to software.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design; C.2.6 [**Computer-Communication Networks**]: Internetworking

General Terms

Design, Experimentation, Performance

Keywords

Network virtualization, Partial reconfiguration, FPGA

*Dong Yin is a Ph.D student at Northwestern Polytechnical University, Xi'an, Shanxi, China. This work was performed while he was a visiting student at the University of Massachusetts, Amherst.

1. INTRODUCTION

The growth and success of the Internet has driven researchers to consider the architecture of next generation networks. Often, network applications call for strikingly diverse performance requirements from the underlying network infrastructure in terms of security, predictability, and throughput. Although the construction of numerous physically-separate networks could satisfy the requirements of different applications, the use of multiple *virtual networks* implemented on a shared physical substrate [3, 15] is generally considered a preferable choice. In these systems, network resources, such as routers, are multiplexed across several virtual networks to limit hardware implementation overhead.

Flexibility, isolation and performance are key architectural requirements for network virtualization platforms [4, 15]. A number of software-based network virtualization platforms implemented on microprocessor systems have been proposed [3, 4, 6, 11]. Although these software virtual routers offer substantial flexibility, they generally suffer from limited packet forwarding performance. For example, OS-level virtualization techniques such as OpenVZ [1] can only offer up to 300 Mbps [3] with 64 byte packets. Higher performance network virtualization platforms use special-purpose hardware such as network processors [18] and FPGAs [2, 19] to implement multiple virtual routers. Although hardware-based packet forwarding systems [2, 18] offer superior packet forwarding performance, the need to shut down hardware during virtual router customizations compromises traffic isolation between shared virtual networks. For example, virtual router modification in a recent FPGA-based network virtualization implementation [19] requires the entire FPGA to be shut down for 12 seconds. Although other shared hardware virtual networks can be switched to software virtual routers, their throughput drops by an order of magnitude during the reconfiguration period.

In this paper, we present a heterogeneous, FPGA-based network virtualization platform that features partial FPGA reconfiguration. The system consists of hardware virtual routers with dedicated FPGA resources that can be independently and dynamically configured at run time without affecting other operational virtual networks. Since the hardware resources of an FPGA can only host a limited number of high-speed virtual networks, the system supports the inclusion of additional virtual networks using software in a Linux-based workstation. A heuristic allocation algorithm has been implemented to dynamically assign virtual networks to hardware and software virtual routers. The algo-

rithm runs in real time on a host workstation and processes virtual network service requests as they arrive. Our network virtualization platform has been successfully implemented using a Virtex II Pro FPGA on a NetFPGA [14] board attached to a standard Linux-based workstation. Experimental results show that a system with two virtual networks implemented in hardware can forward packets at line rate (1 Gbps) and reconfiguration can be performed in a fraction of a second. Additional implementations using a Virtex 5 FPGA show that the system can scale to support up to 20 hardware virtual networks. Finally, we demonstrate that virtual network migration between hardware and software can be used as an effective technique to satisfy the dynamic bandwidth requirements of virtual networks.

The remainder of the paper is organized as follows. Section 2 discusses related work on network virtualization and partial FPGA reconfiguration for networking applications. Section 3 presents the details of the dynamically reconfigurable network virtualization platform. The experimental methodology used to evaluate the system is described in Section 4 and experimental results are discussed in Section 5. Section 6 concludes this paper and offers directions for future work.

2. RELATED WORK

2.1 Network Virtualization Platforms

Several hardware-based network virtualization platforms have been implemented [14, 18, 19] which demonstrate up to two orders of magnitude faster packet forwarding speed than software-based platforms.

The Supercharging PlanetLab Platform (SPP) [18] uses Intel IXP network processors to implement multiple concurrent virtual networks. Each overlay (virtual) network hosted in the SPP platform has dedicated ternary content addressable memory (TCAM) resources to store forwarding tables. Despite the high performance, the configurability of SPP is limited to customizing forwarding table entries. Recently, FPGAs have been used in a variety of network virtualization projects. A network virtualization system [2] based on the NetFPGA platform was shown to concurrently support up to eight identical virtual routers. Each virtual router in this system can forward packets at line rate. However, customization of virtual routers is also limited to forwarding table entries.

A more recent implementation [19] supports up to four concurrent virtual routers in an FPGA. Each virtual network has dedicated FPGA logic resources to implement diverse virtual network functions. Four IP virtual routers with different forwarding characteristics have been implemented in the Virtex II Pro 50 FPGA. Additional virtual networks, if required, are implemented in software in an adjacent workstation. When a virtual network implemented in hardware needs to be customized, all hardware virtual routers are migrated to software and the FPGA is completely reprogrammed with a new bitstream. Although other virtual networks in the FPGA substrate can continue transmitting packets via software virtual routers during reconfiguration, the performance of software virtual routers constrains the forwarding speed to be up to two orders of magnitude worse than their hardware counterparts. More details of the comparison between this migration approach and our partial reconfiguration approach are provided in Section 5.

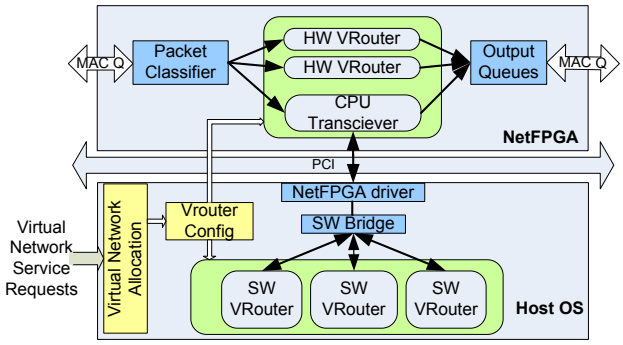

Figure 1: System design overview

2.2 FPGA Dynamic Reconfiguration for Networking Applications

FPGAs have been an important part of several networking projects, some of which use dynamic reconfiguration. The Field Programmable Port Extender (FPX) system [12] uses a partially-reconfigurable Xilinx FPGA to implement a high-speed switch. The FPX system allows packet processing functions to be implemented as reconfigurable modules. Simplified reconfiguration interfaces in the form of standardized APIs are used to adapt the modules [16]. Partial bitstreams are generated and downloaded into the target FPGA by sending specialized control packets from remote administration points. Custom tools, such as PARBIT [7], have been developed to simplify the generation and management of partial bitstreams. A reconfigurable accelerator for packet processing functions in network processors [13] allows customization of common networking tasks such as tree lookup and pattern matching through partial reconfiguration. The feasibility of this approach has been demonstrated using a network intrusion detection application. A dynamically-reconfigurable network processor [8] allows specific parts of a network processor to be reconfigured to meet the specific workload characteristics. The approach was validated using IP forwarding, encryption and media processing flows on Virtex II and Virtex 4 devices. Although steps in a similar direction, these approaches are not directly applicable for multiple virtual routers used by virtual networks.

3. SYSTEM DESIGN

3.1 System Overview

A high-level overview of our heterogeneous network virtualization platform is illustrated in Figure 1. The system is composed of both software and hardware virtual routers. The NetFPGA base router has been extended to create multiple partially-reconfigurable hardware virtual routers while the host software on a workstation is virtualized to execute software virtual routers. All the virtual routers can be configured based on the virtual network service requests from users.

The PCI bus provides an interface between hardware and software virtual routers. All virtual network control planes are implemented in the host software. Packets destined to both software and hardware virtual routers arrive at the interfaces of the NetFPGA card (MAC Q). A packet classifier implemented in the FPGA assigns incoming packets to

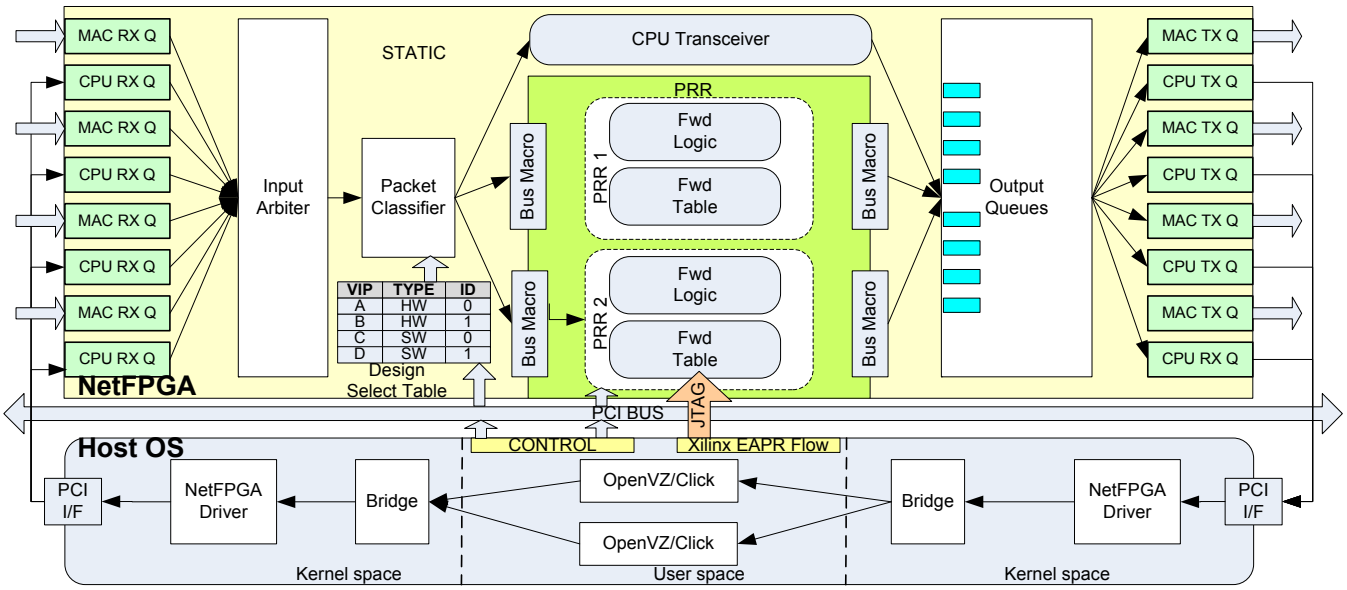

Figure 2: Detailed system implementation of a partially-reconfigurable network virtualization platform on a NetFPGA board and workstation

either hardware or software virtual routers. All packets assigned to software virtual routers are transmitted to the host PC over a PCI interface. A software bridge distributes these packets to software virtual routers. The processed packets are then placed back in NetFPGA output TX queues via the PCI interface for further transmission. Packets processed in hardware virtual routers are also transmitted via the TX output queues. Allocation requests for new virtual networks and removal and bandwidth reallocation requests for existing virtual networks are received by the workstation. An allocation algorithm is then executed by the workstation to determine the assignment of virtual routers to hardware and software based on resource availability.

3.2 System Implementation

The detailed architecture of the system is shown Figure 2. Hardware virtual routers in the system are implemented on a Xilinx Virtex II Pro device which is partially reconfigurable. The Virtex II Pro device is interfaced to four 1 Gbps Ethernet interfaces and SDRAMs on the NetFPGA board. The board is connected to a PC via the PCI interface. Software virtual routers are implemented using container virtualization on the host workstation.

3.2.1 Partially-reconfigurable FPGA System

To support virtual router isolation and facilitate partial reconfiguration, the FPGA is divided into static and partially-reconfigurable regions (PRR). This approach contrasts with previous approaches to FPGA-based network virtualization [2, 19] that do not isolate hardware virtual routers in specific FPGA regions. The static region holds the modules that are shared across multiple virtual routers. These modules include the input arbiter, packet classifier and the output queues. The MAC RX/TX queues interface to the physical MAC and the input arbiter, while the CPU RX/TX queues interface to the host workstation via the PCI bus and the input arbiter. The static region also holds a CPU transceiver module to facilitate the implementation of additional virtual routers in the host software.

Isolated features of virtual routers are implemented in partially-reconfigurable regions. Specific functions in these regions include header verification, checksum verification, IP lookup, ARP lookup and time to live (TTL) updates. These functions are grouped into the Fwd Logic block in Figure 2. A forwarding table for each reconfigurable virtual router is stored in block RAMs (BRAMs). The tables can be updated via the PCI bus by control planes running in host software. The PR regions can be configured by downloading partial bitstreams over a JTAG interface. Specific details of partial bitstream generation are described in Section 4. The packet interface between static and partially-reconfigurable regions consists of FPGA bus macros.

3.2.2 Software-Based System

OpenVZ user space instances, called containers, are used to implement software virtual routers on the host workstation. OpenVZ is a lightweight virtualization approach used in several network virtualization systems [1, 11, 19] and it is included in major Linux distributions. OpenVZ virtualizes a physical server at the operating system level. The kernel manages the allocation of resources such as CPU cycles and memory among all the containers. Each container performs like a standalone server where Click modular routers [9] are executed.

For this system, all packets are received by the NetFPGA card. The destination virtual IP address is used to associate packets with hardware or software virtual routers. A programmable CAM table (Design Select Table in Figure 2) stores the virtual IP to virtual router mappings. Packets associated with a hardware virtual router are sent to the corresponding PRR via bus macros. Processed packets are placed into the output queues for further transmission. Packets associated with software virtual routers are sent to the CPU transceiver module. The CPU transceiver modifies

the layer-2 address in the packet header before placing the packet in one of the available CPU DMA queues (CPU TX Q) interfaced to the PCI bus. The CPU DMA queues are exposed to the host software as virtual Ethernet interfaces. A software bridge forwards packets arriving at virtual Ethernet interfaces to OpenVZ containers running Click modular routers. Processed packets are placed back into the CPU DMA queues from where they are subsequently forwarded to NetFPGA output MAC queues (MAC TX Q) for further transmission.

3.2.3 *Dynamic Virtual Network Allocation*

Our system allows a virtual network operator to migrate a virtual network between hardware and software virtual routers by modifying entries in the Design Select Table and reconfiguring the routers. Virtual network service providers can exploit the heterogeneity in the network virtualization substrate to improve the operational efficiency. For example, virtual networks with time-varying bandwidth requirements may be dynamically migrated between software or hardware resources, as needed. Although attractive, heterogeneity in the virtualization platform also introduces new challenges. The number of software virtual routers is limited by the aggregate bandwidth of the host virtualization technique. The number of virtual routers in the hardware substrate depends on the density of the FPGA device, the logic/memory resource requirements of virtual routers and the placement of partial reconfiguration regions.

Virtual network service requests represent three scenarios: a new virtual network is added to the system, a virtual network is removed from the system, or the bandwidth of an existing virtual network is modified. To support changes, an allocation algorithm which supports the following system updates has been implemented:

Virtual network removal: If a removal request is made, the hardware or software virtual network is removed. All other virtual networks are left in place. A hardware-based virtual router can be removed by programming a blank partial bitstream into the selected reconfiguration region. A software-based virtual router can be removed by destroying the OpenVZ container.

Virtual network addition: If sufficient bandwidth is available, a new software virtual network is created upon request. If not, the network is allocated in hardware. If neither allocation is feasible, the request is rejected.

Virtual network bandwidth adjustment: A request for a bandwidth reduction is applied to the affected virtual network in the system. Other networks are unaffected. If the bandwidth of an existing virtual network is increased, the allocation of all virtual networks in hardware and software is rebalanced. In some cases networks are migrated from software to hardware and vice versa. A greedy approach is currently used to rebalance the virtual networks. For example, if needed, the lowest bandwidth hardware virtual network is migrated from hardware to software or the highest bandwidth software virtual network is migrated from software to hardware to make room in the target resource.

We evaluate the benefit of rebalancing for virtual network bandwidths which vary in time in Section 5.

4. EXPERIMENTAL APPROACH

This section describes the methodology used to generate virtual router partial bitstreams required for experimentation. This description is followed by a discussion of the experimental setup used for performance evaluation.

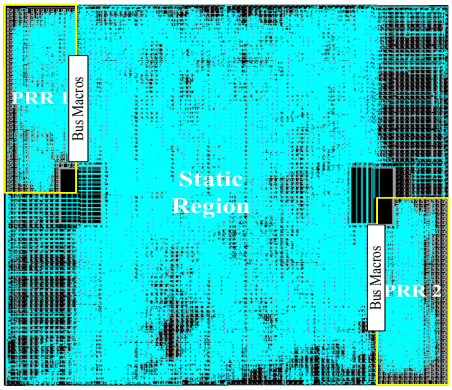

Figure 3: Layout of static and partially reconfigurable regions for network virtualization on a Virtex II Pro

4.1 Partial Bitstream Generation

Partial FPGA reconfiguration requires a priori generation of partial bitstreams for all virtual routers. For our design, virtual routers with column-based FPGA resources are generated in advance of system execution via synthesis and placement constraints and stored in a library. Virtual routers are swapped into the FPGA at run time as needed. In our implementation, slice-based, synchronous bus macros with 8-bit data widths are used as interfaces between the reconfigurable virtual routers and the static logic. All the nets between the static and reconfigurable regions with the exception of global and clock signals are connected through bus macros. The clock to the partially-reconfigurable region is fed from global clock buffers in the static region. The early-access partial reconfiguration (EAPR) [20] design methodology from Xilinx is used to create partial bitstreams. The EAPR methodology requires the designer to follow the following series of steps for generating partial bitstreams.

The static and dynamically-reconfigurable portions are described using distinct sets of Verilog files. A top-level file is created which describes both static and partially-reconfigurable regions and bus macros used for inter-region interfacing. Each portion is synthesized to logic blocks and memory components under timing constraints. Resource counts are evaluated to ensure dynamically-reconfigurable portions are appropriately sized to fit in FPGA columns. Constrained placement is performed for the two design portions using the Xilinx ISE Constraints Editor. The FPGA regions for the static and partially-reconfigurable sections are manually identified using the PlanAhead Layout Editor. The partially-reconfigurable sections can be used for any of the synthesized dynamically-reconfigurable planes. Following placement, timing analysis using timing constraints is performed with the ISE Timing Analyzer. Finally, the static and partially-reconfiguration designs are assembled and the respective bitstreams are generated.

Figure 3 illustrates the layout of a Virtex II Pro device with one reconfigurable virtual router located on each side of the static region. In the Virtex II Pro device, an entire column in a partially reconfigurable region must be reprogrammed at once using a partial bitstream [20]. Multiple

Configuration	Description	Slices/LUTs	BRAMs
I	Dest based IP routing	1443/1861	8
II	Flow based routing	1864/2348	8

Table 1: Experimental configurations

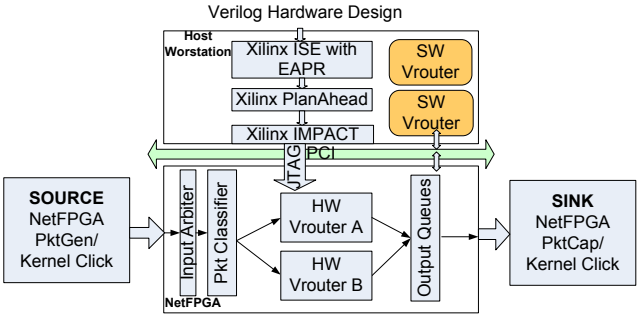

Figure 4: The experimental testbed. A separate workstation/NetFPGA card is used to generate packets and measure packet throughput

reconfiguration regions cannot be placed within the same column. The operation of the device continues unaffected while one or more columns are reconfigured.

Bitstreams generated using the EAPR flow are downloaded using the Xilinx Impact tool via the JTAG interface running at 12 MHz. Given the small size of the target Virtex II device, a maximum of two virtual routers can be implemented in the FPGA. Each virtual router can be dynamically assigned a configuration from Table 1 through partial reconfiguration. The first configuration (Configuration I) follows forwarding based on destination IP addresses. The second configuration (Configuration II) forwards packets using flow information. In this case, packets are forwarded by performing prefix lookups based on source and destination addresses in the packet header. Both configurations fit within a single FPGA column.

4.2 Testbed Setting

The source-router-sink topology shown in Figure 4 is used to measure the performance of the system. Network traffic is generated and captured with the NetFPGA packet generator tool [5] located on a separate workstation. The hardware-based packet generator can accurately generate and capture traffic at line rate (1 Gbps). The hardware-based packet generator only reports the average throughput during experiments. To measure the instantaneous changes in throughput during reconfiguration, we use a kernel Click based UDP packet generator. This packet generator can only achieve 850 Mbps throughput. Xilinx XPower (XPE) is used to estimate the power consumption of the system.

4.3 Comparison with Previous Implementation

To justify the benefits of our new system, the performance of the system is directly compared with results from an earlier system that includes both FPGA-based and software-based virtual routers [19]. This earlier system, which does not support partial FPGA reconfiguration, requires the following steps to reconfigure a virtual router in a FPGA.

- All virtual routers implemented in the FPGA are moved to software-based virtual routers in an adjacent workstation. All traffic is rerouted from the NetFPGA to a NIC located on the PCI bus.
- The FPGA is shut down and fully reconfigured. The FPGA performs no packet forwarding during this time.
- Following reconfiguration, the migrated software-based virtual routers are terminated and forwarding of these packets is once again performed by the FPGA-based hardware. All packets are once again received at the NetFPGA.

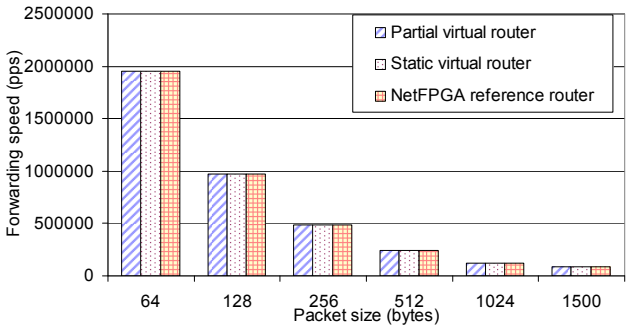

Figure 5: Throughput comparison of partially-reconfigurable, statically-reconfigurable, and reference routers

Extensive additional details of the specific steps are described in [19]. An obvious limitation of this approach is the delay associated with migrating virtual routers and reconfiguring the entire FPGA. A direct comparison between this approach and the new approach is quantified in Section 5. Subsequently, this approach will be referred to as the *statically-reconfigurable* approach.

4.4 Virtex 5 Implementation

Although no in-system experiments were performed, the virtual router architecture shown in Figure 2 was also implemented on a Virtex 5 (VLX330T) device. Virtex 5 offers enhanced placement flexibility by allowing reconfiguration regions of arbitrary rectangular shapes to be placed within the same column. This placement flexibility combined with the availability of additional logic resources allows designers to implement up to 20 virtual routers in partially reconfigurable regions. Total resource usage of the system including both static and partially reconfigurable regions is approximately 68% of the entire Virtex 5 device. Each partially reconfigurable region is isolated in a rectangular shape which can be configured with a partial bitstream.

5. EXPERIMENTAL RESULTS

The key performance parameters used in the evaluation of the system are the observed throughput of the virtual routers, traffic isolation between the virtual networks and the overhead of reconfiguration.

5.1 Single Virtual Router Throughput

In an initial experiment, the baseline performance of a partially reconfigurable virtual router is compared against the performance of one virtual router using the statically-reconfigurable approach, described in Section 4.3, and the

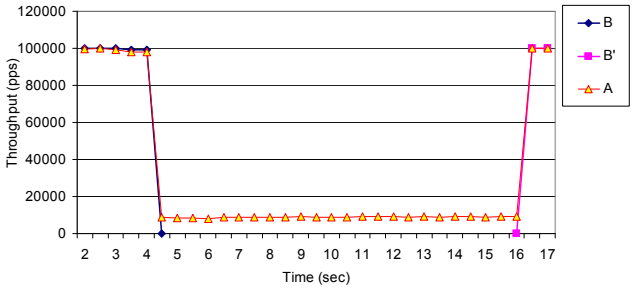

Figure 6: Instantaneous forwarding performance for two virtual networks on a Virtex II using static reconfiguration

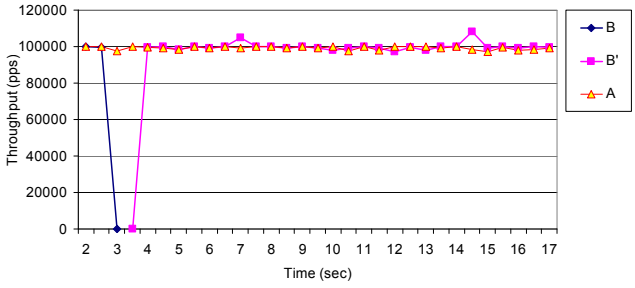

Figure 7: Instantaneous forwarding performance for two virtual networks on a Virtex II using partial reconfiguration

NetFPGA reference router [17]. The NetFPGA packet generator tool [5] is used to generate and capture packets at line rate (1 Gbps). All three designs operate at 62.5 MHz. Figure 5 compares the throughput at the receiver for different packet sizes in all three cases. The performance of the partially-reconfigurable virtual router matches the performance of the reference router and the previous statically-reconfigurable virtual router for all packet sizes. Although not shown in Figure 5, experiments with two partially reconfigurable virtual routers show that the combined aggregate throughput of the virtual networks for 64 byte packets is 1,953,125 packets per second (1 Gbps).

5.2 Instantaneous Throughput

In the next experiment, the impact of reconfiguration on forwarding performance of shared hardware virtual routers is evaluated. Consider a scenario where two virtual routers A and B with identical configurations (Configuration I in Table 1) are implemented in a FPGA. At t=3s, virtual router B is replaced by virtual router $B^{'}$ which implements Configuration II. Figure 6 shows the instantaneous throughput of each of the three virtual routers sampled every 0.5 seconds if a static reconfiguration approach is used. At the start of reconfiguration at t=4.5s, B's throughput drops to 0, while A's throughput drops by more than an order of magnitude since it has been migrated to software. Virtual router $B^{'}$ starts forwarding packets 12 seconds later when the FPGA has completed full reconfiguration. Figure 7 shows the instantaneous throughput for the partially-reconfigurable case. Although B's throughput drops to 0 at the start of partial reconfiguration, A's throughput shows no change. After partial reconfiguration completes, full throughput of virtual network $B^{'}$ is restored.

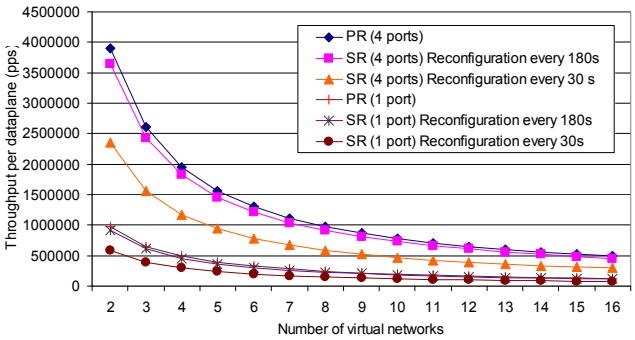

Figure 8: Average throughput for varying reconfiguration frequencies for partially and statically reconfigurable cases

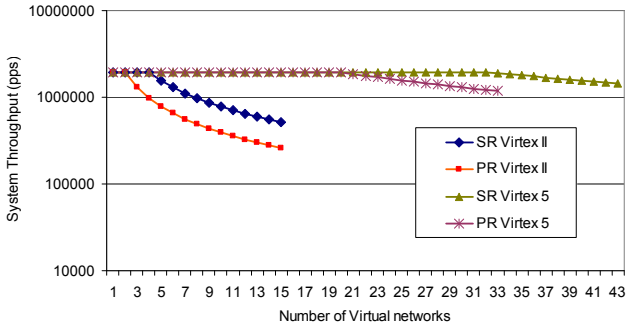

Figure 9: Average throughput of heterogeneous substrate for an increasing number of virtual networks

5.3 Average Throughput

Figure 8 indicates the benefit of using partial reconfiguration of virtual routers versus the static reconfiguration approach for the Virtex 5 device for cases when *all* virtual networks are located in FPGA hardware. In this experiment, it is assumed that virtual networks in the FPGA either remain static or must be configured either every 30 seconds or 180 seconds. Two cases are considered; either one or four 1 Gbps ports on the NetFPGA card are used for an overall potential throughput of 1 Gbps or 4 Gbps. The graph shows the per-virtual network throughput as the number of FPGA-based virtual routers increases. The throughput of the partially-reconfigurable (PR) virtual routers is unaffected since all routers except the one being configured remain active during reconfiguration. However, for the static reconfiguration (SR) cases, an FPGA shutdown for 12 seconds [19] causes increased throughput loss as the number of virtual routers and network ports increases.

The frequency of reconfiguration plays an important part in the benefit of partial reconfiguration. If virtual router reconfiguration never occurs or occurs infrequently, the statically reconfigurable approach can achieve higher throughput. For example, Figure 9 shows the average throughput of the heterogeneous system which includes both hardware and software virtual routers if reconfiguration is never per-

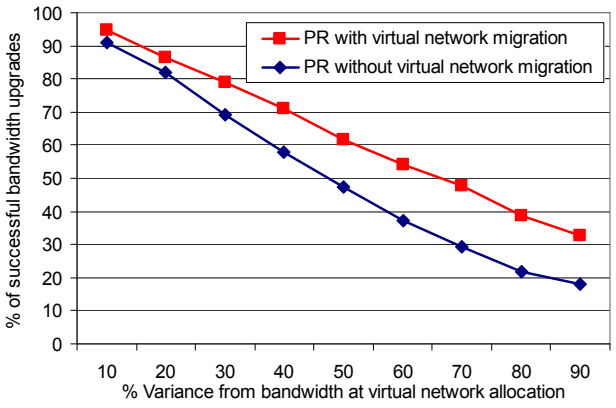

Figure 10: Usefulness of virtual network migration to satisfy the real-time bandwidth characteristics of virtual networks

formed. The use of rigid placement regions for the partially reconfigurable virtual routers limits the number of virtual networks versus the statically-reconfigurable case. For example, a total of 2 partially reconfigurable virtual routers can be placed in a Virtex II while 4 statically reconfigurable virtual routers can be supported. For the Virtex 5, the virtual router count is 20 and 32, respectively. Since fewer high-speed virtual routers can be implemented in hardware, the overall throughput of the dynamically reconfigurable system drops off a bit earlier. However, since periodic virtual network reconfiguration is expected for future systems, the results shown in Figure 8 represent a more realistic scenario.

The run time overhead of partial reconfiguration depends on the size of the partial bitstream and the frequency of the JTAG interface. Experimental results indicate that a 680 KB bitstream can be reconfigured over a 12 MHz JTAG interface in 0.6 seconds. This number is in contrast to the 12 seconds required for full (static) reconfiguration of the same FPGA through the PCI interface, including bitstream download time.

5.4 Dynamic Virtual Network Allocation

The effects of virtual network allocation described in Section 3.2.3 were quantitatively evaluated using 1000 virtual networks whose bandwidths are distributed according to a sample bandwidth distribution measured from PlanetLab nodes [10]. Software-based virtual routers, implemented as OpenVZ containers, offer an aggregate bandwidth of 100 Mbps. FPGA-based virtual routers offer up to 1 Gbps aggregate throughput. It is assumed that virtual networks addition and removal requests arrive according to a Poisson distribution with a mean arrival period of 2 hours. The mean lifetime of a virtual network is a Poisson distribution with a mean of 64 hours. Additionally, it is assumed that the bandwidth of each active virtual network changes every hour by an amount which ranges from 0% up to a maximum variance. The change in bandwidth for each specific network is uniformly distributed up to the maximum percentage variance. A high variance value indicates large fluctuations in real-time bandwidth requirements (both increases and decreases). If a bandwidth variation increase cannot be met, the current bandwidth is maintained.

Region	Dynamic Power (mW)
PR-Region1	173.38
PR-Region2	165.68
Static-Region	593.56

Table 2: Dynamic power consumption in a Virtex II device

Figure 10 shows the percentage of successful bandwidth revisions for different variance values for cases when virtual network migration is performed and when it is not performed. A larger number of bandwidth revisions are granted when virtual networks have small fluctuations from their initial bandwidth assignments. Reallocation and virtual network migration are not needed in most of these cases. However, when virtual networks show large fluctuations from their current bandwidth assignments, reallocation and virtual network migration play important roles in satisfying 10-15% more bandwidth revision requests. Virtual network additions and removals were included in generating these results.

5.5 Power Consumption

Table 2 shows the dynamic power consumption of the Virtex II system running at 62.5 MHz with two IP routing data planes. The dynamic power consumption of the virtual routers is dependent on their internal structure. Total static power consumption of the Virtex II device is 158.75 mW. When a virtual router is unused, the corresponding reconfigurable region can be shut down by downloading a blank configuration bitstream, saving approximately 16% of total device power consumption.

6. CONCLUSION AND FUTURE WORK

We have described a heterogeneous network virtualization platform that integrates several software virtual routers with partially-reconfigurable hardware virtual routers. Experimental results with a Virtex II Pro based FPGA board indicate that hardware virtual networks can forward packets at line rate and can be reconfigured within a fraction of a second. This result represents a 20x improvement over a previous approach which required complete FPGA reconfiguration. In the future, we plan to investigate the use of partial reconfiguration for network virtualization in larger Virtex 6 devices. The use of intra-FPGA ICAP interfaces for faster reconfiguration is also an interesting possibility. Our future work will also focus on developing user-friendly interfaces to generate and download partial bitstreams for network operators with little hardware knowledge. More robust virtual network allocation algorithms will also be explored.

7. ACKNOWLEDGEMENTS

We thank the anonymous reviewers for their valuable comments. The FPGA compilation software and interface cores were generously donated by Xilinx Corporation. This work was partially supported by NSF grants CNS-0626618 and CNS-0831940. Dong Yin was supported by China State Scholarship fund CSC-2008629080.

8. REFERENCES

[1] OpenVZ project page. `http://www.openvz.org/`.

[2] M. Anwer and N. Feamster. Building a fast, virtualized data plane with programmable hardware. In *Proceedings of the ACM SIGCOMM Workshop on Virtualized Infastructure Systems and Architectures*, pages 1–8, Aug. 2009.

[3] A. Bavier, N. Feamster, M. Huang, L. Peterson, and J. Rexford. In VINI veritas: realistic and controlled network experimentation. In *Proceedings of the ACM International Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 3–14, Sept. 2006.

[4] S. Bhatia, M. Motiwala, W. Muhlbauer, Y. Mundada, V. Valancius, A. Bavier, N. Feamster, L. Peterson, and J. Rexford. Trellis: A platform for building flexible, fast virtual networks on commodity hardware. In *Proceedings of the ACM Conference on Emerging Network Experiment and Technology*, pages 72–77, Dec. 2008.

[5] G. A. Covington, G. Gibb, J. W. Lockwood, and N. McKeown. A packet generator on the NetFPGA platform. In *Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines*, pages 235–238, Apr. 2009.

[6] N. Egi, A. Greenhalgh, M. Handley, M. Hoerdt, F. Huici, and L. Mathy. Towards high performance virtual routers on commodity hardware. In *Proceedings of the ACM International Conference on Emerging Networking EXperiments and Technologies*, pages 20–31, Dec. 2008.

[7] E. L. Horta, J. W. Lockwood, and S. Kerfuji. Using PARBIT to implement partial run-time reconfigurable systems. In *Proceedings of the International Conference on Field Programmable Logic and Applications*, pages 182–191, Jan. 2002.

[8] C. Kachris and S. Vassiliadis. Analysis of a reconfigurable network processor. In *Proceedings of the IEEE Reconfigurable Arcitectures Workshop*, pages 1–8, Apr. 2006.

[9] E. Kohler. *The Click modular router*. PhD thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, 2000.

[10] S. J. Lee, P. Sharma, S. Banerjee, S. Basu, and R. Fonseca. Measuring bandwidth between PlanetLab nodes. In *Proceedings of the Passive and Active Network Measurement Conference*, pages 292–305, Mar. 2005.

[11] Y. Liao, D. Yin, and L. Gao. PdP: Parallelizing data plane in virtual network substrate. In *Proceedings of the ACM SIGCOMM Workshop on Virtualized Infastructure Systems and Architectures*, pages 9–18, Aug. 2009.

[12] J. W. Lockwood, N. Naufel, J. S. Turner, and D. E. Taylor. Reprogrammable network packet processing on the field programmable port extender (FPX). In *Proceedings of the ACM International Symposium on Field Programmable Gate Arrays*, pages 87–93, Feb. 2001.

[13] G. Memik, S. O. Memik, and W. H. Mangione-Smith. Design and analysis of a layer seven network processor accelerator using reconfigurable logic. In *Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines*, pages 131–140, Apr. 2002.

[14] J. Naous, G. Gibb, S. Bolouki, and N. McKeown. NetFPGA: Reusable router architecture for experimental research. In *Proceedings of the ACM Workshop on Programmable Routers for Extensible Services of Tomorrow*, pages 1–7, Aug. 2008.

[15] L. Peterson, S. Shenker, and J. Turner. Overcoming the internet impasse through virtualization. In *Proceedings of the Third Workshop on Hot Topics in Networking*, pages 34–41, Nov. 2004.

[16] T. S. Sproull, J. W. Lockwood, and D. E. Taylor. Control and configuration software for a reconfigurable networking hardware platform. In *Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines*, pages 45–54, Apr. 2002.

[17] Stanford University. *NetFPGA User's Guide*, Mar. 2008. http://www.netfpga.org.

[18] J. Turner, P. Crowleyand, J. DeHart, A. Freestone, B. Heller, F. Kuhns, S. Kumar, J. Lockwood, J. Lu, M. Wilson, C. Wiseman, and D. Zar. Supercharging PlanetLab: a high performance, multi-application, overlay network platform. In *Proceedings of the ACM International Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 85–96, Aug. 2007.

[19] D. Unnikrishnan, R. Vadlamani, Y. Liao, A. Dwaraki, J. Crenne, L. Gao, and R. Tessier. Scalable network virtualization using FPGAs. In *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, pages 219–228, Feb. 2010.

[20] Xilinx Corporation. *Early Access Partial Reconfiguration User Guide*, Sept. 2008. User Guide UG208.