A Packet Generator on the NetFPGA Platform

G. Adam Covington, Glen Gibb, John W. Lockwood, Nick McKeown

Stanford University
California, USA
{gcoving, grg, jwlockwd, nickm}@stanford.edu
http://netfpga.org

***Abstract*— A packet generator and network traffic capture system has been implemented on the NetFPGA. The NetFPGA is an open networking platform accelerator that enables rapid development of hardware-accelerated packet processing applications. The packet generator application allows Internet packets to be transmitted at line rate on up to four Gigabit Ethernet ports simultaneously. Data transmitted is specified in a standard PCAP file, transferred to local memory on the NetFPGA card, then sent on the Gigabit links using a precise data rate, inter-packet delay, and number of iterations specified by the user. The hardware circuit also simultaneously operates as a packet capture system, allowing traffic to be captured from up to all four of the Gigabit Ethernet ports. Timestamps are recorded and traffic can be transferred back to the host and stored using the same PCAP format. The project has been implemented as a fully open-source project and serves as an exemplar project on how to build and distribute NetFPGA applications. All of the code (Verilog hardware, system software, verification scripts, makefiles, and support tools) can be freely downloaded from the NetFPGA.org website. Benchmarks comparing this hardware-accelerated application to the fastest available PC with a PCIe NIC shows that the FPGA-based hardware-accelerator far exceeds the performance possible using TCP-reply software.**

I. Introduction

The NetFPGA platform allows for the rapid prototype and development of multi-Gigabit/second line rate networking applications. The open-source NetFPGA distribution consists of gateware, hardware and software. Source code and scripts are provided to build reference designs, enhance a design, or create new applications using libraries that are provided. The base distribution of the NetFPGA currently contains four reference projects – reference router, reference Network Interface Card (NIC), and hardware accelerated Linux router. In addition, there are several user-contributed projects available such as the netflow probe, OpenFlow switch, and the Packet Generator. As with all NetFPGA projects, the Packet Generator has a standard directory structure. By using this well defined directory structure, packaging scripts automate the creation of the Packet Generator packages [2].

The Packet Generator is a real-time application that is difficult to implement in software on a PC. Software packages running on PCs can't guarantee when packets are transmitted and don't allow full line rate testing. Proprietary products from companies such as Ixia are expensive and not available to the open-source community. The NetFPGA platform implements an open-source Packet Generator and packet capture system that operates at full Gigabit Ethernet line rates.

II. NetFPGA Platform

A. NetFPGA Infrastructure

The NetFPGA is a network hardware accelerator that augments the function of a standard computer. The plug-in card provides four ports of Gigabit Ethernet and includes local Static RAM (SRAM) and Dynamic RAM (DRAM) for local processing. The NetFPGA attaches to the Peripheral Communication Interconnect (PCI) bus. The FPGA directly handles all data-path switching, routing, and processing of Ethernet and Internet packets, leaving software to handle only control-path functions [5].

The combination of the NetFPGA and a PC are used to implement wire-speed Internet routers, precise network measurement systems, and hardware-accelerated network processing systems. The NetFPGA can be used in a desktop PC and in rack-mounted servers. In the classroom or teaching lab, the NetFPGA is usually installed inside a desktop PC so students can access the hardware [7] [3].

III. Gateware and Software

One of appealing aspects of the NetFPGA Platform is the availability of the open-source Verilog gateware, and the accompanying software. The packages that are released on the website, www.netfpga.org, not only contain the source code for projects, but also an environment that allows researchers the ability to easily create, simulate, and verify in hardware the applications and features they create. The gateware itself is designed in a modular fashion to allow users to create and connect modules in new configurations. Most designs invoke use of the reference pipeline.

The NetFPGA release has three major components. The first is the kernel module that is used to communicate with the NetFPGA hardware. It allows the bitfiles for the FPGA to be loaded through the PCI bus of a Linux based PC. In addition, it allows programs to communicate to the NetFPGA through a register interface implemented using shared memory. The reference systems use the DMA to send and receive packets from the card. The register system is used to read statistics counters and to write control data from software running on the host to the hardware.

The second part of the NetFPGA release are the common utilities used to communicate with the card. These utilities include but are not limited to a bitfile download utility and

register read and write programs. The third part of the release is the reference pipeline, described below.

IV. Reference Pipeline

The reference pipeline, as shown in Figure 1, is comprised of eight receive queues, eight transmit queues, and the user data path. The receive and transmit queues are divided into two types: MAC and CPU. The MAC queues are assigned to one of the four interfaces on the NetFPGA, and there is one CPU queue associated with each of the MAC queues.

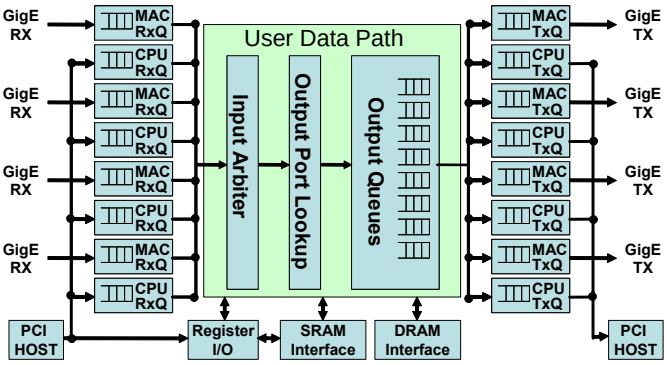

Fig. 1. NetFPGA Reference Pipeline

Users add and connect their modules to the User Data Path. The Input Arbiter and the Output Queues modules are the main modules that are present in almost all NetFPGA designs. The source code for these modules are provided in the NetFPGA Verilog library. The Input Arbiter services the eight input queues in a round robin fashion to feed a wide (64-bit) packet pipeline.

The register system allows modules to be inserted into the pipeline with minimal effort. The register interface allows software programs running on the host system to send data to and receive data from the hardware modules. Registers and counters are assigned names that are common to the hardware design and C or Perl software that runs on the host PC.

V. Related Work

The NetFPGA Packet Generator is not the first system built to generate packets. Hardware systems from Ixia [4] stochastically generate network traffic. Ixia systems allow the users to create and save synthetic traces to be rerun in the future. These systems can be useful however they do not allow the replay of previously captured traffic from live networks, such as PCAP files.

Software programs such as tcpreplay [6] and TCPivo [1] allow the replay of saved network traffic. These software programs, called trace-driven packet generators, run on off-the-shelf computers. These systems have trouble replaying traffic at line rate with consistency. When running tcpreplay multiple times it becomes apparent that the time between packets varies due to factors such as the CPU load on the system, disc I/O, and variation in the time servicing interrupts. The inter-packet jitter doesn't allow experiments to be performed without variations in the traffic. TCPivo attempts to minimize these problems by employing a low latency kernel, and implementing network trace prefetching along with low latency counters.

On the NetFPGA, PCAP data is loaded into the shared SRAM directly attached to the FPGA. The PCAP data is then played out of the four Gigabit ethernet ports at line rate. In addition, the NetFPGA Packet Generator enforces the inter-packet delays and/or rate limits the four ports individually. This allows experiments to be run with extremely predictable and repeatable results. Software is used to load the parameters into hardware. Once the parameters are loaded and the replay is enabled the hardware controls the streaming of traffic. The NetFPGA Packet Generator is not impacted by kernel preemption or the latency of accesses across the PCI bus.

VI. Packet Generator Architecture

The architecture of the Packet Generator utilizes the reference pipeline as a foundation. The main features of the Packet Generator are inserted into the User Data Path. Only minor modifications were required outside of the User Data Path. As shown at the top-left of Figure 2, a timestamp module [1] was added before the MAC fifos. This allows incoming packets to be timestamped as they are received by the hardware. This timestamp is later used during the creation of a PCAP file enabling much better precision than timestamping performed as the kernel received each packet from the hardware.

A. Input Arbiter/Output Port Lookup

Inside the User Data Path, two unmodified modules were used from the original NetFPGA library: the input arbiter and the output port lookup from the reference NIC design. The reference NIC output port lookup directs traffic received on the MAC ports to the corresponding CPU queues to enable the packet capture feature. This allows all incoming packets to be forwarded to the software host using the CPU DMA queues.

B. Packet Capture

The Packet Capture module performs the roles of (1) compiling aggregate statistics for generation/capture runs (such as number of packets received and total capture time) and (2) stripping timestamps from the packets when the generation/-capture function is disabled. When the Packet Generator is disabled, the circuit operates as a normal quad-port NIC card.

C. Output Queues

The design of the Output Queues module is based on the design of the SRAM output queues from the NetFPGA library; however, there is a slight modification. The SRAM used for the output queues is divided into 12 queues instead of the original 8 queues that the reference module implements. This allows the four new queues that are used to store the PCAP data to transmit data when the Packet Generator is enabled.

[1]The timestamp module is based upon the timestamp module developed as part of a time synchronization project.

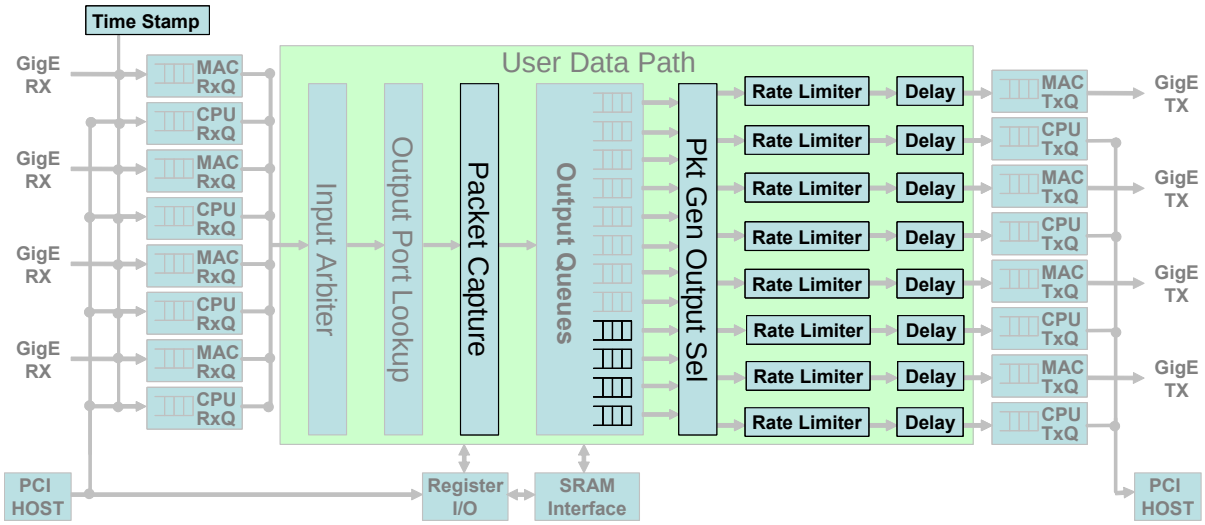

Fig. 2. Packet Generator Pipeline (new components highlighted)

The size of the 12 queues is determined by registers that the packet generator software sets as it loads PCAP files. This allows the queues to be independently sized to ensure that transmit queues are sized to match the PCAP file sizes and thus maximize the space available for receive queues.

D. Packet Generator Output Select

The Output Queues are connected to the Packet Generator Output Select module. This module determines which of the 12 output queues should be connected to the eight output queues in the reference pipeline. In essence, this module multiplexes the output queues to the eight reference output queues. This enables the design to have two transmit queues per MAC port—one which can be used for transmission while a PCAP file is being loaded into the other.

E. Rate Limiter and Delay Module

Each of the eight reference output queues have both a rate limiter, and a delay module connected to them. This allows the Packet Generator to add a delay or rate limit to each of the eight output queues individually. The delay and the rate are both set by registers that are written by the packet generator software running on the host.

F. Registers

Each of the hardware modules can be controlled through registers via software. The key registers used in the Packet Generator include: delay, rate limit, and number of iterations. Each of the rate limit and delay modules can be enabled or disabled by individual enable registers. The Packet Generator contains a global enable register that starts and stops the sending of packets.

VII. Packet Generator Software

The Packet Generator software consists of a Perl script called packet_generator.pl. This file is found in the software directory of the Packet Generator package.

The Packet Generator can be run on one or all of the four ports of the NetFPGA. To specify which PCAP file is loaded into a queue, the -q option is used. The Packet Generator loads full packets into the hardware. If the PCAP file is larger than the memory available, then only the first part of the PCAP file is loaded.

There are four additional options available on a per port basis. The Packet Generator allows the user to specify the transmission rate of each queue, the delay between packets, the number of iterations to replay, and whether or not packets being received by the NetFPGA MAC ports should be captured. If capture is enabled, the packets are stored in SRAM then transferred to the host PC and written to a file using standard PCAP format. This allows easy comparison of the incoming packets to the packets being sent by the Packet Generator, which is useful for verification of other modules connected to the NetFPGA.

Software on the host prints out the number of packets that were loaded into the queues from the PCAP file and it reports the rate limit setting of each queue. When capture is enabled, the number of packets received, number of bytes, run time, and transmission rate are all reported.

VIII. Experimentation

The NetFPGA Packet Generator operates similar to that of the software program, tcpreplay. We ran two experiments with the tcpreplay and the Packet Generator. We configured a computer system that contained a AMD dual core processor running at 2.5 GHz and an Intel dual port e1000 PCI-express x4 NIC. We then used a PCAP file that contained 43 packets and 25383 bytes. Using tcpreplay we tested the average rate that the system could achieve when playing this file on one port and also on two ports simultaneously. The Packet Generator was then used to play the same PCAP file with both one port and two ports simultaneously. Each experimental setup (i.e. one port, and two ports) were run ten times and the

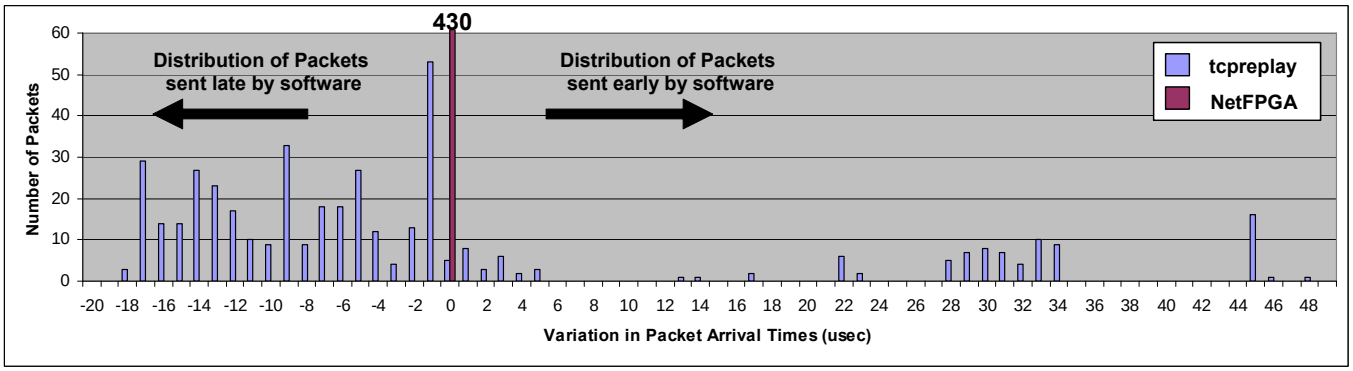

Fig. 3. Measuring the Variation of Packet Arrival Time of tcpreplay

averages are shown in the table below (Table I). Because the NetFPGA hardware runs at line rate, it can play traffic at full line rate.

Number of Ports	TCPreplay Rate	NetFPGA Rate
One	901.31 Mbps	1000 Mbps
Two	896.23 Mbps	1000 Mbps

TABLE I
AVERAGE LINE RATE FOR TCPREPLAY AND THE PACKET GENERATOR USING ONE AND TWO PORTS

Being able to reliably repeat an experiment is a key feature of the NetFPGA packet generator. When using software programs running on a PC, it is difficult to ensure that inter-packet delays are always the same between experiments. Figure 3 shows the packet arrival time distribution of tcpreplay. When tcpreplay is run in software, most packets obtain a 5-18 microsecond delay from when there were expected to arrive. Few packets are within plus or minus four microseconds of the expected arrival time. These variations in packet arrival time limits the precision experimenters can achieve. Using the NetFPGA, we can ensure that the delay between packets and the rate of sending is the same every time an experiment is run.

IX. Device Utilization

The Packet Generator uses 83% of the available slices on the Xilinx Virtex II Pro 50 FPGA. The largest use of the slices are from the Packet Capture Selector and the replication of the rate limiter and delay modules on each of the eight output ports. Sixty percent of the block RAMs available are used. The main use of block RAMs occurs in the FIFOs used between the modules and the main input and output queues of the system.

X. Conclusion

The NetFPGA Packet Generator uses the reference pipeline to replay PCAP and capture packets at Gigabit/second line rate. The NetFPGA implementation can reliably replay time sensitive traffic while giving users the ability to change the rate of the queues, the delay between packets, and the number of iterations that the PCAP files are cycled through. This application can be used to provide reliable test data to test multi-Gigabit/second networks and network appliances.

Resources	XC2VP50 Utilization	Utilization Percentage
Slices	19674 out of 23616	83%
4-input LUTS	30049 out of 47232	63%
Flip Flops	22570 out of 47232	47%
Block RAMs	140 out of 232	60%
External IOBs	356 out of 692	51%

TABLE II
DEVICE UTILIZATION FOR THE PACKET GENERATOR

References

[1] W. chang Feng, A. Goel, A. Bezzaz, W. chi Feng, and J. Walpole. Tcpivo: A high-performance packet replay engine. In *Proceedings of the ACM SIGCOMM workshop on Models, methods*, pages 57–64, 2003.

[2] G. A. Covington, G. Gibb, J. Naous, J. Lockwood, and N. McKeown. Methodology to contribute netfpga modules. In *International Conference on Microelectronic Systems Education (submitted to)*, 2009.

[3] G. Gibb, J. W. Lockwood, J. Naous, P. Hartke, and N. McKeown. Netfpga: An open platform for teaching how to build gigabit-rate network switches and routers. In *IEEE Transactions on Education*, August 2008.

[4] Ixia. Ixia website. http://www.ixiacom.com/.

[5] J. W. Lockwood, N. McKeown, G. Watson, G. Gibb, P. Hartke, J. Naous, R. Raghuraman, and J. Luo. Netfpga - an open platform for gigabit-rate network switching and routing. In *International Conference on Microelectronic Systems Education*, 2007.

[6] tcpreplay developers. tcpreplay website. http://tcpreplay.synfin.net/trac/wiki/tcpreplay.

[7] G. Watson, N. MxKeown, and M. Casado. Netfpga - a tool for network research and education. In *2nd Workshop on Architecture Research using FPGA Platforms (WARFP)*, Febuary 2006.