VOL. 15, NO. 2, JANUARY 2020 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

NETFPGA: STATUS, USES, DEVELOPMENTS, CHALLENGES, AND EVALUATION

Dixon Salcedo Morillo[1], César Guerrero Santander[2] and Albeiro Cortés Cabezas[3]
[1]Department of Computer Science and Electronics, Coast University, Barranquilla, Colombia
[2]Engineering and Organization Research Center, Autonomous University of Bucaramanga, Bucaramanga, Colombia
[3]Department of Electronic Engineering, Surcolombiana University, Neiva, Colombia
E-Mail: dsalcedo2@cuc.edu.co

ABSTRACT

The constant growth of the Internet, driven by the demand for timely access to data center networks has meant that the technological platforms necessary to achieve this purpose are outside the current budgets. In this order to make and validate relevant, timely and relevant contributions, it is necessary that a wider community, access to evaluation, experimentation and demonstration environments with specifications that can be compared with existing networking solutions. This article introduces the NetFPGA, which is a platform to develop network hardware for reconfigurable and rapid prototyping. It introduces the application areas in high-performance networks, advantages for traffic analysis, packet flow, hardware acceleration, power consumption and parallel processing in real time. Likewise, it presents the advantages of the platform for research, education, innovation, and future trends of this platform. Finally, we present a performance evaluation of the tool called OSNT (Open-Source Network Tester) and shows that OSNT has 95% accuracy of timestamp with resolution of 10ns for the generation of TCP traffic, and 90% efficiency capturing packets at 10Gbps of full line-rate.

Keywords: NetFPGA-SUME; OSNT; open-source hardware; open-source software; traffics analysis.

INTRODUCTION

Embedded system development platforms, to create specialized hardware, have been evolving since the 80s. Since of first system based on microcontrollers to the most advanced called Systems on a Chip (SoC). Currently the development of these systems focuses on reducing the development time of complex systems, as well as material costs and energy consumption (Dagher, 2019), (Reddy, 2018) and (K.M. Gayathri, 2018). In this field, FPGAs (Field Programmable Gate Arrays) have a differentiating adds value; they can be reprogrammed using their interconnected logic blocks, which favorably impacts the hardware development cycle, allowing rapid design, modeling, debugging and optimization of any type of custom hardware required (Henkel, Wolf, & Chakradhar, 2004), (Forconesi, Sutter, Lopez-Buedo, Vergara, & Aracil, 2014). The Hardware Descriptor Language approach is used to describe and design, being Verilog and VHDL. The current hardware is development under control of two platforms trends, Xilinx® and Altera®.

Likewise, among the best-known network hardware development platforms are SoNIC (Software-defined Network Interface Card) (Lee, Wang, & Weatherspoon, 2013), which allows through software, access and control in real time to the physical layer of the network adapter; achieving rates of up to 10Gbps. Another platform is called Labkit (MIT, 2007), developed by the Massachusetts Institute of Technology (MIT), which allows the development of complex, high-performance projects, including applications for audio and video processing, among others.

Stanford University, Stanford, CA, developed the platform, known as NetFPGA (NetFPGA.org, 2017), from which its reference designs such as *reference_router*, *reference_nic*, *reference_switch*, and *accept_test*. Important applications have also been developed, including Packet generator (Covington, Gibb, Lockwood, & McKeown, 2009), OpenFlow (McKeown, y otros, 2008), SCONE (NetFPGA, 2013), Blueswitch (Han, y otros, 2015). The three platforms described above are of the open-source type, which allows the reuse of code.

Table-1 describes the most important characteristics of configurable hardware development platforms for computer networks, which allow verifying that the NetFPGA platform is the most active, due to the number of high-impact projects developed and enabled by research centers and educational institutions. These institutions maintain the NetFPGA in constant growth, validity and leadership in the area of open source hardware design.

Table-1. Platforms for development of configurable network hardware.

Platforms	Ref. designs	Active Projects	High-speed ports	Associated Institutions	Released Versions
SoNIC	2	2	4	1	1
Labkit	1	4	4	1	1
NetFPGA	6	32	4	+150	4

Additionally, a review of the literature to publications on NetFPGA allowed to find that since the beginning of the platform (Watson, McKeown, & Casado., 2006), (Lockwood, y otros, 2007), it is evolution and new technological developments, most have been produced by institutions or universities in North America and Europe; evidencing little development by Latin American institutions.

On the other hand, an ideal development platform requires being scalable, flexible, and useful for a wide range of applications in general-purpose or specific devices. For example, a network device can be used in two ways, like a network element or final host adapter. Also, open source hardware has reached maturity, although in process adoption and used on a large scale and active community that stimulates the constant growth of a library of repositories that includes reference designs, hardware designs and software (Zilberman N. , Audzevich, Kalogeridou, Manihatty-Bojan, Zhang, & Moore, 2015).

Regarding the precise generation and monitoring of traffic in high-speed computer networks and real-time applications, the NetFPGA-SUME provides a high-speed platform, which is useful for novel data-center interconnection architectures, block host construction and 100 Gbps switches, for basic networking research, and as a platform for exploring completely new protocol architectures and interconnection equipment, beyond the current restrictions of PCIe devices.

This study presents two main aspects. First, includes a review of the NetFPGA platform and its applications. Second, shows the evaluation of the performance of the OSNT tool, with respect to the precision to generate and timestamping of data packets. The remainder of this paper is structured as follows. In section II the authors briefly summarize the NetFPGA projects, its components, architecture, and characteristics of the cards that compose it. Then, in section III, an introduction to the developments of NetFPGA and its most representative applications is made. After, in section IV, the paper presents the impact of using the NetFPGA platform on education, research, and innovation. In addition, in section IV, the authors present the impact of the use of the NetFPGA platform on education, research, and innovation. In Section V you can find a discussion about the use of NetFPGA in new trends in the area of computer networks and related. Then, in Section VI you can see the evaluation of the performance of the precision and efficiency of the OSNT tool. Finally, the study draws the main conclusions and discusses future work in Section VII.

NETFPGA PLATFORM

Initially, it is important to point out that several projects at the end precede the NetFPGA platform, in the revised literature two stands out. The first, called Click (Kohler, Morris, Chen, Jannotti, & Kaashoek, 2000), is a software architecture to build flexible and configurable routers. The second, called XORP (eXtensible Open Router Platform) (Handley, Hodson, & Kohler, 2003), is a stable platform for research that allows building, improving and strengthening a router, prioritizing the configuration of parameters that frequently present conflicts. Additionally, the NetFPGA project created at ends of 2001 by Stanford University, as an effort of researchers for the teaching of computer networks, under the open-source philosophy, and the first version was used in 2003, to developing a class project at the graduate level at Stanford University (Watson, McKeown, & Casado., 2006).

NetFPGA platforms components

The NetFPGA is a hardware expansion device. In other words, a multipurpose card, that at level of computer networks allows to develop in little time and to low cost, functional prototypes of devices of interconnection networks, that can operate between 1.0 and 10 Gbps of speed. Similarly, NetFPGA is an open hardware and software platform, because it integrates four different elements, which appear below (Blott, Ellithorpe, McKeown, Vissers, & Zeng, 2010):

a) The open hardware represented on the programmable card, with its own processor and four 10Gbps high-speed network ports each,

b) Development tools and reference designs, which facilitate the creation of new devices, taking these standard designs as a starting point,

c) Developed projects, whose code is available for reuse, and

d) An academic research network, present in more than 150 institutions around the world, which is actively developing projects and documenting the platform.

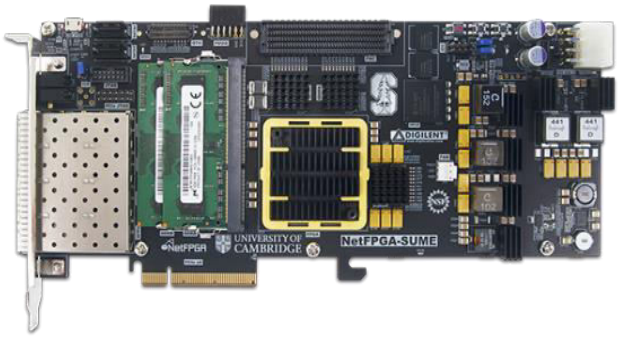

Figure-1. NetFPGA-SUME Card.

According to the versions of NetFPGA released to date are in chronological order, NetFPGA-1G (Discontinued), NetFPGA-1G-CML, NetFPGA-10G and NetFPGA-SUME (NetFPGA.org, 2017), (Zilberman., 2014); of which the last three are actively supported. The NetFPGA-1G-CML, which replaced the original NetFPGA-1G card, was designed for low-bandwidth applications (up to 1.0 Gbps), and specially adapted for network security applications (Moran Edgar, 2017). Then

introduced in 2010, the NetFPGA-10G, which has multiple 10 Gbps interfaces, and supports a large number of open source community projects (e.g., OSNT (Antichi, y otros, OSNT: Open Source Network Tester, 2014) and BlueSwitch. Finally, NetFPGA-SUME released in 2014, and it has enabled four I/O ports to operate up to 100 Gbps as a computer network device, and to function as a stand-alone computing unit or for testing and measurement. Each card has very important features. The NetFPGA-SUME card, see Figure-1, is a PCIe-(x8-Gen3) card, which incorporates a Xilinx Virtex-7 690T FPGA, with I/O capabilities to operate at rates between 10 Gbps and 100 Gbps. Additionally, we can use this card as NIC, multi-port switch, firewall, test/measurement environment, among other applications. Table 2 expands the features of the current platform versions.

NetFPGA architecture

The NetFPGA structured in blocks, you can see it in Figure-2. It has two different components. The first one refers to the host that has the hardware and software resources of the PC that contains the card, including the network software. The second is the card itself, which behaves as a hardware accelerator that integrates an FPGA, handling network ports of 1/10Gbps.

In relation to the above, and placing the block diagram in a real context. For example, in a router, the host-side components are a general-purpose computer that executes high-demand processing tasks; this represents the Control-Plane. On the other hand, there is the NetFPGA, which is high-speed hardware for transmitting packets through the management of network ports by an FPGA, and corresponds to the Data-Plane (Cao, Zheng, Sun, & Jin, 2015).

On the other hand, when performing a more detailed analysis of the internal structure of the NetFPGA, and we find that it has a modular design. Consequently, each project designed for NetFPGA inherits this same structure. Therefore, it is important to know the behavior of the basic data flow (Data_path) between the modules. Finally, the entire module structure is called Reference Pipeline, and you can see it in Figure-3 (Covington, Gibb, Naous, Lockwood, & McKeown, 2009).

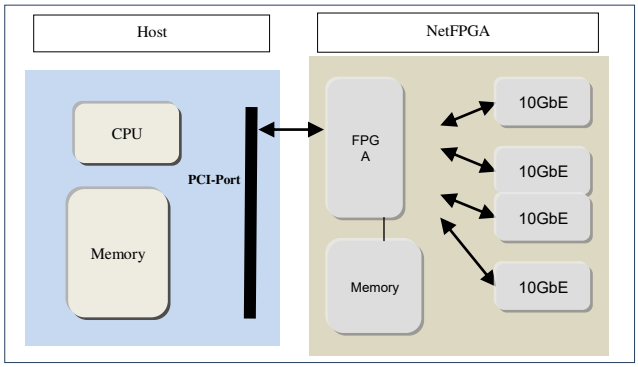

Figure-2. NetFPGA card block diagram.

The Pipeline connected to the host PCIe bus at one end of the NetFPGA; and has several Rx queues that receive packets from the I/O ports (Ethernet or PCI). These ports connected to the User_Data_Path, which contains the processing steps. Similarly, within the User_Data_Path, there is the Input_Arbiter, which is the first module that a data packet passes through when it arrives at the NetFPGA. Likewise, the Input_Arbiter decides which *Rx* queue it serves, and then takes the packet from that queue and delivers it to the next module in the pipeline. That module is the Output_Port_Lookup, which decides which port the data packets are sent through. Then, the data packet is delivered to the next module called Output_Queues, which stores the packet in the queues that correspond to the output port until the *Tx* queue accepts the data packet for transmission. Finally, each of the modules described above has a set of registers that provide information about the status, access, and adjustment control signals of the NetFPGA (Antichi, Giordano, Miller, & Moore, 2012).

Table-2. Features of NetFPGA cards.

Features	NetFPGA 10G	NetFPGA 1G CML	NetFPGA-SUME
CPU & FPGA	-Virtex-5 - 240K celdas lógicas y 11,664 Kbit de BRAM.	-Kintex-7 - 326,080 celdas lógicas y 16,020 Kbit de BRAM.	-Virtex-7 690T - 693,120 celdas lógicas y 52,920 Kbit BRAM.
RAM	4 x36 RLDRAM-II, 400MHz, de 288MB.	DDR3 x8, 800MHz, de 512MB.	DDR3-SoDIMM, 933MHz, de 8 GB (escalable a 32GB).
Net Ports	4 x 10Gbps.	4 x 1Gbps.	4 x 10Gbps. SFP+
PCI Ports	PCIe x8, de 4 líneas de 5Gbps.	PCIe x4, de 8 líneas de 2.5 Gbps.	- PCIe de 3ª. Gen. 8 líneas de 8Gbps.
Ext. Storage	2 flash Cards, de 256Mb	1 SD-Flash de 1Gb.	2, Micro-SD slot, 512Mb hasta 1Gb

Regarding I/O interfaces, the NetFPGA-SUME platform has a new industry standard called AMBA-AXI architecture. Similarly, Xilinx uses AXI4 as the transmission protocol for the Data-Plane interface, while

Control-Plane uses AXI-Lite (Cao, Zheng, Sun, & Jin, 2015).

DEVELOPMENTS ON NETFPGA

The growth of the NetFPGA community has equally driven the quantity, quality, and relevance of developments in two tracks. On the one hand, the reference designs, which serve as a basis for new developments. On the other hand, developments those are completely new, and address solutions unpublished or based on existing products, making them tools accessible to all users. Therefore, the following is a summary of the most important developments of the platform.

Traffic generation and analysis

NetFPGA traffic generators stand out for being more accurate than those executed from the application layer are. Packet Generator (Antichi, Giordano, Miller, & Moore, 2012), can play a PCAP file and capture packets at 1Gbps, allowing you to change the speed of queues, the delay between packets and the number of iterations to which PCAP files are passed. On the other hand, the Precise Traffic Generator (PTG) (Salmon, Ghobadi, Ganjali, Labrecque, & Steffan, 2009) sends host generated packets at highly accurate transmission times between extremes, and designed to integrate with existing software traffic generators and network emulators. Another tool is OSNT (Zilberman N. , Audzevich, Kalogeridou, Manihatty-Bojan, Zhang, & Moore, 2015) that can generate and receive packets on the 10 GbE (Giga bit Ethernet) interfaces, and incorporates time stamps to each outbound packet, allowing end-to-end packet delay and packet loss to be calculated. Finally, the Automatic Test Packet Generation (ATPG) (Zeng, Kazemian, Varghese, & McKeown, 2012)can evaluate protocol rules for complex networks using a small number of test packets. For example, sending 4000 test packets, 10 times per second, consumes less than 1% of the evaluated link capacity.

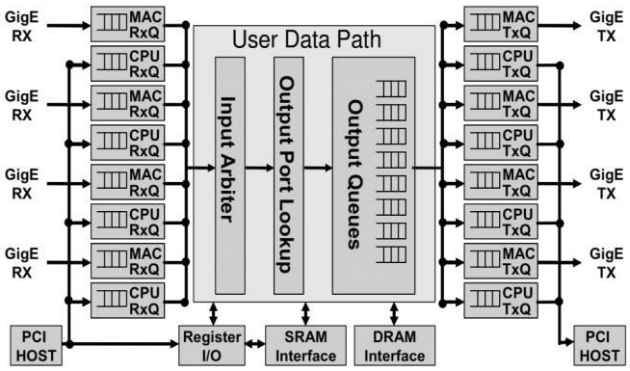

Figure-3. NetFPGA standard reference Pipeline.

In relation to traffic analyzers, in (Keinänen, Jokela, & Slavov, 2009) zFilter is a tool that eliminates the forwarding of the traditional IP address of a data packet and replaces it with Bloom's filtering technique for making packet forwarding decisions. As a result, ZFilter is faster than the IP protocol, because it reaches a forwarding delay between packets of up to 3µs. On the other hand, Dynamic Packet-filtering (Engelmann, Lukaseder, Erb, Heijden, & Kargl, 2014), uses the method of extracting signatures in header fields, this is done by processing packets in parallel in NetFPGA 10G, and reaches transmission speeds of 9.5 Gbps. We found that HyPaFilter tool (Fiessler, Hager, Scheuermann, & Moore, 2016) is a packet-classification hybrid system, based on circuits adapted to an FPGA, therefore, when evaluating HyPaFilter shows a performance 30 times higher than that executed in software. Finally, other tools that have been developed parallel to the traffic analysis on the NetFPGA-10G, are the Network Intrusion Detection Systems (NIDS), one of them is called SNORT Accelerating (Al-Dalky, Salah, Otrok, & Al-Qutayri, 2014), and other protection of Denial of Distributed Services (DDoS) (Pham-Quoc, Tran-Thanh, Tuan, & Thinh, 2016).

Routing and switching

Initially we found that one of the most significant contributions made to the networking area by the NetFPGA platform, has been for the treatment of packet flow; in this aspect, there are developments, among which the following stand out.

OpenFlow is a communications protocol that gives access to the packet-forwarding plane of a switch or router over the network, and based on an Ethernet switch, with an internal flow table and a standardized interface to add and remove packet flow entries. On the other hand, we find *Blueswitch* that supports a transaction configuration mechanism, and provides a stable packet configuration; that is, all packets that pass through the Data_path will find the old configuration or the new one, and never an inconsistent mix of the two (Han, y otros, 2015). Similarly, we find the *Quagga Routing Suite* tool (Jakma & Lamparter, 2014), which provides implementations of various routing protocols distributed through multiple communication processes through a technique called Inter-Process Communication (IPC). This technique is a functional prototype of a high-speed switch, which performs a transmission on the network communication model via socket, supported on the high-speed switching and transmission offered by NetFPGA-SUME (Su, You, Wang, & Hou, 2016).

Hardware acceleration and power consumption

In relation to the high performance in packet processing offered by the NetFPGA architecture, researchers have taken advantage of these advantages to increase the performance of the software they use. For example, RISC Based (Han, Zilberman, Zeeb, Fiessler, & Moore, 2016)shows that when a network application requires computationally intensive, it increases the number of cores in order to achieve better computing performance, and when an application requires computationally intensive network data, the single-core implementation provides resources between (70% - 80%). Another development, known as PacketShader (Han, Jang, Park, & Moon, 2010), implements IPv4 and IPv6 forwarding, with OpenFlow, switching, and IPsec to demonstrate flexibility

and performance; where evaluation results show that the Graphical Processing Unit (GPU) performs better in CPU implementation.

Regarding the power consumption of the network hardware, there are works that present an adequate power management policy, take into account the user restrictions regarding Quality of Service (QoS), and change the clock frequency of the NetFPGA according to the input bit rate to decrease the power consumed by approximately 12% (Lombardo, Panarello, Reforgiato, & Schembra., 2012). Another work that uses the NetFPGA, proposes an adaptive frequency control mechanism based on traffic to reduce power consumption. This mechanism calculates the volume of traffic in real time so that the network device can adjust its operating frequency. Consequently, the results show that the router that incorporates this mechanism can reduce power consumption by more than 20% (Zhou, Li, Liu, & Wang, 2014).

Finally, another aspect that favors the growth of this area and the increase in the performance of developments over the NetFPGA, are the studies on technologies and techniques to reduce latency in storage and forwarding of data packets, among which are related works in (Ruiz, Ramos, Sutter, Vergara, Lopez-Buedo, & Aracil, 2016), (Nakamura, Hayashi, & Matsutani, 2016), and (Jeyakumar, Alizadeh, Geng, Kim, & Maziéres., 2014).

NETFPGA APPLICATIONS

The NetFPGA platform is useful in other contexts such as education, research, and innovation. Below is a summary of project progress in these areas.

Education

Initially, it is important to note that the NetFPGA project minimizes the lack of open (non-proprietary) tools for teaching computer network systems at the undergraduate and graduate levels. In addition, professors from Stanford University observed that students only gained practical experience in creating networks from layer three and above (routing protocols, transport protocols, etc.). Likewise, students' access to the Physical and Link layer was limited to theoretical classes, and they were unable to build network systems. These and other reasons generated a need to build a platform that students can use to design, model functional network devices in real environments (NICs, switches, routers) (Watson, McKeown, & Casado., 2006).

The creation of an "Open-source hardware" platform is very different from open-source software projects, because of this; they created a new process to provide new designs. Therefore, in order for a contributed design to be included in the NetFPGA repository, it must be completely specified through a set of tests that must pass successfully. Currently the active repository is located at github.com/NetFPGA.

Similarly, the platform has grown with the creation of courses such as CS344 by Stanford University, which allowed the development of reference designs and component libraries of the NetFPGA (Gibb, Lockwood, Naous, Hartke, & McKeown, 2008). We can also find books such as "High Speed NetFPGA Router" (Khalid, High Speed NetFPGA Router: A Step by Step Guide on Developing a High Speed Router on NetFPGA Board, 2012), which explains in detail how to implement routing algorithms such as IPv4 and OSPF over NetFPGA, and covers basic routing concepts to understand reference_router.

In addition, as a strategy for introducing, updating and perfecting knowledge about the NetFPGA platform, universities such as Stanford and now Cambridge have developed and positioned since 2007 events that are an inherent part of the community, such as Seminars, Camps, Developer's Workshops and Tutorials (McKeown, Lockwood, Naous, Gibb, & Covington, 2007), (Zilberman., 2014), and (NetFPGA.org, 2017).

On the other hand, when in the processes of formation and the number of students is high. NetFPGA cards may not be sufficient for all students to access the development of practices at the same time, making the learning process difficult. Consequently, this limitation allowed the development of the Open Network Laboratory (ONL) tool (Wiseman, Turner, DeHart, Parwatikar, Wong, & Zar, 2009), which is a network testbed accessible from the Internet, and allows the use of numerous heterogeneous network resources for research and education activities, with 20 routers, 100 PCs and 6 NetFPGAs. Another remote laboratory tool is FPGA e-Lab (Hashemian & Riddley, 2007), but focuses on teaching Digital Design using FPGAs, which is a basic knowledge to use the maximum performance to the NetFPGA. In addition, e-Lab is composed of hardware and a *Laboratory Protocol* that gives access to students.

Finally, it is important to note that the NetFPGA platform has a high degree of learning difficulty for those who do not have basic training in knowledge of digital circuit design or ends. One answer to this is EMU (Galea, y otros, 2016), which uses a compiler called Kiwi (Singh & Greaves, 2008), and programs FPGAs using .NET code. Furthermore, Emu provides an implementation for network functionalities written in C#.

Research and innovation

Being flexible and allowing rapid prototyping makes the NetFPGA platform the ideal tool for students and researchers of any level, because it offers multiple solutions to them. Therefore, below we introduce the most important projects. For example, (McKeown, y otros, 2008)shows that NetFPGA provides reference projects or other inputs, which provide a complete implementation and an executable application. For example (Han, y otros, 2015), an SDN researcher interested in the control plane and lacking any hardware knowledge can use the BlueSwitch or OpenFlow project. Additionally, the NetFPGA community developed a tool called Mininet (Lantz, Heller, & McKeown, 2010), which is a system for rapidly prototyping large networks, over the limited resources of a single computer; this is because it uses the lightweight approach to virtualization features at the OS

level, including processes (Zilberman N. , Audzevich, Kalogeridou, Manihatty-Bojan, Zhang, & Moore, 2015).

Regarding the obstacles faced by researchers to develop NetFPGA projects that include innovation, we can relate the cost and development time, the skills required for hardware development. Consequently, several solutions were developed. For example, (Forconesi, Sutter, Lopez-Buedo, Vergara, & Aracil, 2014) compares the two approaches to hardware development called the Hardware Description Language (HDL), and the new, High-Level Language (HLL). Also, this work perform a practical implementation of packet processing, which allows to demonstrate that using HLL reduces the coding time, maintaining high speed, low latency and accuracy of timestamping of packets in the hardware architecture, which ratifies that using coding in C++ with Vivado-HLS, can smooth the gap between software development and hardware for network applications.

In connection with the above, tools such as Program-hosted Directability (PhD) (Sultana, y otros, 2017) were developed, which helps researchers to quickly develop prototypes of network hardware because it interprets Direction commands at runtime, allowing debugging, monitoring, and profiling that are normally set at compile time for hardware development. Indeed, the tool offers significant flexibility with low impact on hardware utilization and optimal performance.

On the other hand, researchers in the area of computer networks that focus on monitoring traffic and evaluating network protocols, the NetFPGA offers several tools for experimentation, because its flexibility allows you to modify and customize existing designs (Yaser M. Abid, 2018), (Wira Firdaus Yaakob, 2018). Of these tools, we present two that are the most representative. The first one is a module developed in hardware to monitor high-speed networks (Antichi, Miller, & Giordano, An open-source hardware module for high-speed network monitoring on netfpga, 2010), which allows filtering customized data packets. The second is a traffic generator for 10 Gbps Ethernet links, configurable from software (Groléat, y otros, 2013), and executed on platforms such as INVEA-TECH or NetFPGA-10G. Additionally, it can transmit packets as small as the Ethernet protocol allows, and is capable of delivering up to 20 Gbps, whether each transmission block is linked to the traffic generator, thus configuring two data streams of 10Gbps each.

Another important aspect that benefits from the features of the NetFPGA is innovation. Likewise, we consider that innovation is not present in every research process. Similarly, the chances of producing an impact are very low. Therefore, this low probability of success is the cause of the low availability of equipment, protocols and traffic in real scenarios (McKeown, y otros, 2008).

As a result, there have been two major developments for the NetFPGA platform. The first, known as OpenFlow, offers a way for researchers to run experimental protocols on heterogeneous switches uniformly across computer networks. Second, called OSNT, it is available to the research community through the NetFPGA project. Therefore, anyone who owns a NetFPGA card can use and manage it without additional hardware expense. Finally, OSNT is a project that allows modifications and customizations to solve specific needs. Finally, the use of the NetFPGA facilitates the possibilities of improvement and innovation oriented to high-speed interconnection devices, specifically switches. For example, in (Su, You, Wang, & Hou, 2016) tested the data rate performance of the reference_switch over the NetFPGA-SUME. Additionally, can switch and retransmit a 1.09 GB video file in 14 seconds and run it successfully at an actual transmission rate of 631 Mbps; this generates a number of opportunities to create novel solutions based on open hardware and software.

CHALLENGES OF NETFPGA vs. NETWORKING TRENDS

The constant growth of the demand for Internet services, leads to grow at the same rate in resources, transmission bandwidth, CPU processing power, interconnections to Datacenter. Consequently, this has stimulated the need for high-speed network solutions, for research and the real world, in areas such as Web load balancing, DDos, and for IDS at 100 Gbps or higher, with minimum packets length, and test and capture challenges (Cao, Zheng, Sun, & Jin, 2015). Accordingly, NetFPGA-SUME provides a high-speed platform for novel data center interconnect architectures, host block construction, and 100Gbps switch, for basic research, and as a platform for exploring completely new host architectures beyond current PCIe constraints.

Another challenge, which involves the use of the NetFPGA platform, is the difficulty of programming the hardware in the FPGA. In (Rothman & Chang, 2012), we present the tool P4FPGA, which reduces the barrier to start working with the powerful tool. Similarly, P4FPGA provides a P4 to FPGA compiler that supports multiple architectures, generating code that can run on FPGAs from manufacturers such as Xilinx or Altera. Likewise, we find two technologies that are oriented to follow in the future of networking. One such technology is Network Function Virtualization (NFV) (Joshi & Benson, 2016), which makes use of basic hardware resources as the basic platform for performing specialized network functions, as opposed to specialized network hardware devices. In this area, NetFPGA has great potential because, by integrating all its potential, with the flexibility of NFV based on General Purpose Processors (GPP). FPGA-based NFV can be useful for improving the performance of hardware resources on network devices (Kachris, Sirakoulis, & Soudris., 2014). On the other hand, we found a technology called Software Defined Networks (SDN), which has become the complexity of the network forwarding network elements (switches/routers) to a centralized controller; being OpenFlow its main development on NetFPGA. In order to, this has allowed a great space for new ideas and challenges to appear in front of the development needs in networking; for example, Internet of Things technology integration (Azeem Mohammad Abdul, 2016), (Wira Firdaus Yaakob, 2018), (A. Murali, 2016).

NETFPGA EVALUATION: AN OSNT CASE STUDY

As described in the previous sections, the NetFPGA platform has many application areas, including high-speed computer networks. This section focus on evaluating one of the most important tools for testing high-speed network protocols developed on this platform, called OSNT.

OSNT architecture

The OSNT architecture is a response to the limitations of previous solutions: proprietary/closed-source solutions, high costs, lack of flexibility and the omission of important features such as time stamping and precise packet transmission.

OSNT NetFPGA-SUME (OSNT-SUME) can generate and capture packets of any size; additionally it can manipulate the transmission rate on all four card ports simultaneously. On the other hand, the OSNT-SUME implementation provides methods to scale and coordinate multiple systems for generating and capturing traffic, and configured with a timestamping resolution of 6.4ns.

OSNT has two tools that constitute themselves as the most important of its system. The first one, called *Traffic Generator* (OSNT-Generator): capable of generating and receiving packets in four 10GbE interfaces; by incorporating time stamps in each sent packet, it allows calculating information about delay and loss in an end-to-end network. The second, *Traffic* Monitor (OSNT-Monitor): capable of capturing packets that arrive through four 10 GbE network interfaces, which are transferred to the host software for analysis and further processing.

Network testbed and experiments

To perform the OSNT evaluation, we install a NetFPGA-SUME on each host. Similarly, on each of the hosts (called A and B) the OSNT-SUME tool was configured. Likewise, on host A, the Traffic Generator is executed, and on host B, the Traffic Monitor. Therefore, these tools are directly interconnected using *nf0* (host A) and *nf1* (host B) by optical means. Finally, both hosts have Linux CentOS 7 installed as operating system, see Figure-4a and 4b.

Figure-4a. Network testbed.

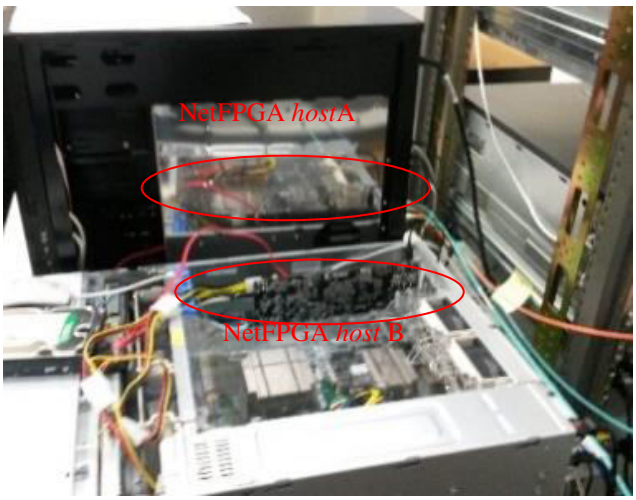

Figure-4b. Hosts A and B Interconnected.

On the other hand, we set up experiments to evaluate the measurement of timestamping accuracy and OSNT packet capture efficiency. Therefore, we created two large transmission rate scenarios of 1.0 Gps and 10Gps. Therefore, for each scenario, three delay values of 6.4ns (minimum defined OSNT), 10ns and 100ns were evaluated. Similarly, the number of packets sent per burst in each experiment was 10, 100, 1000, 10000, and 1000000 packets, with sizes of 64B and 1500B. Additionally, for each scenario with speeds of 1.0 GbE and 10 GbE, we performed 10 tests per burst, per delay and per packet size; that is, (2 scenarios x 10 repetitions x 5 bursts x 2 delays x 2 packet sizes), allowing us to perform a total of 400 tests.

After executing the experiments, the traces captured from the bursts of packets received by OSNT-Monitor were stored in a pcap files for later analysis.

Results

Table-3. Lost packets at 1.0 GbE rate.

Packets size (Bytes)	Burst packets	Delay Lost packets (%)		
		6,4ns	10ns	100ns
64	10	0%	0%	0%
64	100	0%	0%	0%
64	1000	75%	0%	0%
64	10000	96%	0%	2%
64	100000	99%	3%	7%
1500	10	0%	0%	0%
1500	100	60%	0%	5%
1500	1000	92%	0%	0%
1500	10000	80%	0%	2%
1500	100000	80%	3%	6%

Therefore, Tables 3 and 4 show the average percentage of packets lost for transmission speeds of 1GbE and 10GbE, the packet loss is high reaching up to 99% with delays of 6.4ns, which indicates that the packets are received by the adapter, but not captured by the kernel, i.e., they never get received by the operating system protocol stack. However, when the delay is 10ns and 100ns the packet loss is minimal, because it is minimal, if compared to the size of the burst sent.

On the other hand, in relation to the transmission speed of 10 GbE, the performance is very similar to that of 1.0 GbE, except that at 100ns of delay there is a loss of packets that reaches 6%, representing 6000 packets of the 10000 sent in the burst.

Regarding the timestamping accuracy with the different packet delays evaluated, we found different results that represent the average relative error for bursts between 10 and 100000 packets. Table 5 shows that for 1 GbE and 10 GbE of speed, the highest relative error found is 5%, for delays of 6.4ns and 10ns.

Finally, Table-6 shows that in general OSNT, executed on the NetFPGA platform, is very efficient and accurate, when it comes to evaluating network protocols at high speeds (optical networks), which range from 10 to 100 GbE. Although it is, correct to say that OSNT, due to the scalability characteristics of the NetFPGA platform, can support tests in networks with speeds up to 400 GbE.

Table-4. Lost packets at 10 GbE rate.

Packets size (Bytes)	Burst packets	Delay Lost packets (%)		
		6,4ns	10ns	100ns
64	10	0%	0%	0%
64	100	0%	0%	0%
64	1000	66%	0%	0%
64	10000	87%	0%	7%
64	100000	88%	4%	8%
1500	10	0%	0%	0%
1500	100	0%	0%	0%
1500	1000	68%	0%	0%
1500	10000	85%	0%	0%
1500	100000	89%	3%	0%

Table-5. Timestamping relative error at 1.0 GbE.

Packets size (Bytes)	Average Relative Delay Error (%)					
	6,4ns	Error	10ns	Error	100ns	Error
64	6,48	1%	9,48	5%	102	2%
1500	6,34	1%	8,73	3%	103,4	3%

Table-6. Timestamping relative error at 10 GbE.

Packets size (Bytes)	Average Relative Delay Error (%)					
	6,4ns	Error	6,4ns	Error	6,4ns	Error
64	6,4	0%	9,7	3%	102,2	2%
1500	6,7	5%	10,2	2%	100,6	1%

CONCLUSIONS

Initially, the current and future use of the tool with respect to new networking technologies was analyzed. Likewise, the measure that the use of FPGAs is gradually being used in high-performance systems, for large storage systems, in energy-efficient systems. Similarly, programmable SoC's are combined with highly efficient algorithms, to support flexible services, ranging from the distribution of stored content to the analysis of large volumes of data; and for services in sensory networks in the area of the Internet of Things (IoT). Finally, in relation to the above features, we consider that the NetFPGA will become an important platform for high-speed next-generation network research, playing a role in the development of interconnection hardware in the networks of the future.

Finally, OSNT's evaluation showed that it is a tool with high precision to mark the time of sending data packets at high transmission speeds. Additionally, at high transmission speeds OSNT's packet capture tool has low packet loss, and offers many advantages for research and development of new solutions in the area of network traffic analysis and monitoring, which are useful for the management of current and future telecommunications infrastructures.

ACKNOWLEDGEMENTS

Author Dixon Salcedo thanks COLCIENCIAS and the Universidad de la Costa for the financial resources to carry out this research. He also thanks the Autonomous University of Bucaramanga, which provided the laboratories for the development of the research. Finally, to Pontificial Bolivarian University where I studied my doctorate in engineering.

REFERENCIAS

A. Murali K. H. 2016. Integrating FPGAs with Trigger Circuitry Core System Insertions for Observability in Debugging Process. Journal of Engineering and Applied Sciences. 2643-2650.

Al-Dalky R., Salah K., Otrok H. & Al-Qutayri M. 2014. Accelerating snort NIDS using NetFPGA-based Bloom filter. 2014 International Wireless Communications and Mobile Computing Conference (IWCMC). pp. 869-874.

Antichi G., Giordano S., Miller D. & Moore A. 2012. Enabling Open-source High Speed Network Monitoring on NetFPGA. Network Operations and Management Symposium (NOMS), 2012 IEEE. pp. 1029-1035.

Antichi G., Miller D. & Giordano S. 2010. An open-source hardware module for high-speed network monitoring on netfpga. European NetFPGA Developers Workshop.

Azeem Mohammad Abdul, B. M. 2016. IOT Based Home Automation Using FPGA. Journal of Engineering and Applied Sciences. 1931-1937.

Blott M., Ellithorpe J., McKeown N., Vissers K. & Zeng, H. 2010. FPGA Research Design Platform Fuels Network Advances. Xilinx Xcell Journal, Fourth Quarter. 27.

Cao J., Zheng X., Sun L. & Jin J. 2015. The Development Status and Trend of NetFPGA. Network and Information Systems for Computers (ICNISC), 2015 International Conference on. pp. 101-105.

Covington A., Gibb G., Lockwood J. & McKeown N. 2009. A Packet Generator on the NetFPGA Platform. En K. L. Pocek, & D. A. Buell (Ed.), FCCM (págs. 235-238). IEEE Computer Society.

Covington A., Gibb G., Naous J., Lockwood J. & McKeown N. 2009. Encouraging Reusable Network Hardware Design. Microelectronic Systems Education, 2009. MSE '09. IEEE International Conference on. pp. 29-32.

Dagher K. E. 2019. Real-Time Adaptive Intelligent FPGA-based Back-Stepping Control Law Design for a Nonlinear Magnetic Ball Levitation System. Journal of Engineering and Applied Sciences. 6912-6929.

Engelmann F., Lukaseder T., Erb, B., Heijden R. & Kargl F. 2014. Dynamic packet-filtering in high-speed networks using NetFPGAs. Future Generation Communication Technology (FGCT), 2014 Third International Conference on. pp. 55-59.

Fiessler A., Hager S., Scheuermann B. & Moore A. W. 2016. HyPaFilter: A Versatile Hybrid FPGA Packet Filter. Proceedings of the 2016 Symposium on Architectures for Networking and Communications Systems (pp. 25-36). New York, NY, USA: ACM.

Forconesi M., Sutter G., Lopez-Buedo S., Vergara J. E. & Aracil J. 2014. Bridging the gap between hardware and software open source network developments. IEEE Network. 28, 13-19.

Galea S., Sultana N., Bressana P., Greaves D., Soulé R., Moore A. W., y otros. 2016. Emu: Rapid FPGA Prototyping of Network Services in C.

Gibb G., Lockwood J. W., Naous J., Hartke P. & McKeown N. 2008. NetFPGA - An Open Platform for Teaching How to Build Gigabit-Rate Network Switches and Routers. IEEE Transactions on Education. 51, 364-369.

Groléat Tristan, Arzel Matthieu, Vaton Sandrine, y otros. 2013. Flexible, extensible, open-source and affordable FPGA-based traffic generator. Proceedings of the first edition workshop on High performance and programmable networking, pp. 23-30.

Han J. H., Mundkur P., Rotsos C., Antichi G., Dave N. H., Moore A. W., y otros. 2015. Blueswitch: Enabling Provably Consistent Configuration of Network Switches. Proceedings of the Eleventh ANCS-15, (pp. 17-27). Washington: IEEE Computer Society.

Han J. H., Zilberman N., Zeeb B. A., Fiessler A. & Moore A. W. 2016. Prototyping RISC Based, Reconfigurable Networking Applications in Open Source. CoRR, abs/1612.05547.

Han S., Jang K., Park K. & Moon S. 2010. PacketShader: A GPU-accelerated Software Router. SIGCOMM Comput. Commun. Rev. 40, 195-206.

Handley M., Hodson O. & Kohler E. 2003. XORP: An Open Platform for Network Research. SIGCOMM Comput. Commun. Rev. 33, 53-57.

Hashemian R. & Riddley J. 2007. FPGA e-Lab, a technique to remote access a laboratory to design and test. Microelectronic Systems Education, 2007. MSE'07. IEEE International Conference on. pp. 139-140.

Henkel J., Wolf W. & Chakradhar S. 2004. On-chip networks: a scalable, communication-centric embedded system design paradigm. 17th International Conference on VLSI Design. Proceedings. pp. 845-851.

Jakma P. & Lamparter D. 2014. Introduction to the quagga routing suite. IEEE Network. 28, 42-48.

Jeyakumar V., Alizadeh M., Geng Y., Kim C. & Maziéres D. 2014. Millions of Little Minions: Using Packets for Low Latency Network Programming and Visibility. CoRR, abs/1405.7143, 16.

Joshi K. & Benson T. 2016. Network Function Virtualization. IEEE Internet Computing. 20, 7-9.

K.M. Gayathri S. B. 2018. Hardware Implementation and Testing of PAPR Reduction Using Order Bit Selector and Trellis Structure. Journal of Engineering and Applied Sciences. 5027-5036.

Kachris C., Sirakoulis G. & Soudris D. 2014. Network Function Virtualization based on FPGAs: A Framework for all-Programmable network Devices. CoRR, abs/1406.0309, 5.

Keinänen J., Jokela P. & Slavov K. 2009. Implementing zFilter based forwarding node on a NetFPGA. Proc. of NetFPGA Developers Workshop.

Khalid A. 2012. High Speed NetFPGA Router: A Step by Step Guide on Developing a High Speed Router on NetFPGA Board. Lap Lambert Academic Publishing GmbH KG.

Kohler E., Morris R., Chen B., Jannotti J. & Kaashoek M. F. 2000. The Click Modular Router. ACM Trans. Comput. Syst. 18, 263-297.

Lantz B., Heller B. & McKeown N. 2010. A Network in a Laptop: Rapid Prototyping for Software-defined Networks. Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks (pp. 19: 1-19:6). New York, NY, USA: ACM.

Lee K., Wang H. & Weatherspoon H. 2013. SoNIC: Precise Realtime Software Access and Control of Wired Networks. NSDI. pp. 213-225.

Lockwood J., McKeown N., Watson G., Gibb G., Hartke P., Naous J., y otros. 2007. NetFPGA--An Open Platform for Gigabit-Rate Network Switching and Routing. Microelectronic Systems Education, 2007. MSE '07. IEEE International Conference on. pp. 160-161.

Lombardo A., Panarello C., Reforgiato D. & Schembra G. 2012. Power Control and Management in the NetFPGA Gigabit Router. Future Network Mobile Summit (FutureNetw). pp. 1-8.

Maral Faghani M. B. 2015. Integration of Sigma-Delta ADC with Sinc Filter on FPGA. Journal of Engineering and Applied Sciences. 16-21.

McKeown N., Anderson T., Balakrishnan H., Parulkar G., Peterson L., Rexford J., y otros. 2008. OpenFlow: Enabling Innovation in Campus Networks. SIGCOMM Comput. Commun. Rev. 38, 69-74.

McKeown N., Lockwood J. W., Naous J., Gibb G. & Covington A. 2007. Hands-on with the NetFPGA to build a Gigabit-rate Router. High-Performance Interconnects, 2007. HOTI 2007. 15th Annual IEEE Symposium on. pp. 7-10.

MITIntroduction to Digital Systems: Labkit Documentation.

Moran Edgar S. P. 2017. Hardware Firewall Bypass (HWFWBYPASS) Attack on pfSense. Journal of Engineering and Applied Sciences. 7154-7158.

Nakamura K., Hayashi A. & Matsutani H. 2016. An FPGA-Based Low-Latency Network Processing for Spark Streaming. Proceedings of the Workshop on Real-Time and Stream Analytics in Big Data (IEEE BigData 2016 Workshop).

NetFPGA 1 de 2013. SCONE (Software Component Of NetFPGA). Obtenido de https://github.com/NetFPGA/netfpga/wiki/SCONEWalkthrough.

NetFPGA.org2017NetFPGA Platform.

OSNT: Open Source Network Tester2014IEEE Network 286-12.

Pham-Quoc C., Tran-Thanh B., Tuan N. Q. & Thinh T. N. 2016. A DDOS PROTECTION SYSTEM WITH MULTIPLE DEFENSE MECHANISMS USING RECONFIGURABLE HARDWARE. International Journal of Computer Engineering and Applications. X, 75-85.

Reddy S. B. 2018. Review on FPGA Implementation of 3D Distributed Arithmetic based DWT Architecture for Image Processing Applications. Journal of Engineering and Applied Sciences. 9177-9183.

Rothman J. & Chang C. 2012. BEE technology overview. 2012 International Conference on Embedded Computer Systems (SAMOS). pp. 277-277.

Ruiz M., Ramos J., Sutter G., Vergara J. E., Lopez-Buedo S. & Aracil, J. 2016. Accurate and affordable packet-train testing systems for multi-gigabit-per-second networks. IEEE Communications Magazine. 54, 80-87.

Salmon G., Ghobadi M., Ganjali Y., Labrecque M. & Steffan J. 2009. NetFPGA-Based Precise Traffic Generation. in Proc. of NetFPGA Developers Workshop09.

Singh S. & Greaves D. J. 2008. Kiwi: Synthesis of FPGA circuits from parallel programs. Field-Programmable Custom Computing Machines, 2008. FCCM'08. 16th International Symposium on. pp. 3-12.

Su T., You L., Wang Q. & Hou C. 2016. The high speed switching experiment based on NetFPGA SUME. Computer Science & Education (ICCSE), 2016 11th International Conference on. pp. 652-657.

Sultana N., Galea S., Greaves D., Wojcik M., Zilberman N., Clegg R., y otros. 2017. Extending programs with debug-related features, with application to hardware development. arXiv preprint arXiv:1705.09902.

Watson G., McKeown N. & Casado. M. 2006. NetFPGA: A Tool for Network Research and Education. 2nd workshop on Architectural Research using FPGA Platforms (WARFP). 3.

Wira Firdaus Yaakob, H. A. 2018. A Comparative Study of Smart Card Design with Memory Ciphering System on Arm-Based FPGA. Journal of Engineering and Applied Sciences. 2638-2646.

Wiseman C., Turner J., DeHart J., Parwatikar J., Wong K. & Zar D. 2009. Using the netfpga in the open network laboratory. Proceedings of the 1st NetFPGA Developers Workshop.

Yaser M. Abid, A. H. 2018. Supervised Feed Forward Neural Networks for Smart Chessboard Based on FPGA. Journal of Engineering and Applied Sciences. 4093-4098.

Zeng H., Kazemian P., Varghese G. & McKeown N. N. 2012. Automatic Test Packet Generation. Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies (pp. 241-252). New York, NY, USA: ACM.

Zhou L., Li L., Liu X. & Wang X. 2014. A low power consumption frequency adaptation mechanism based on the traffic and implementation on NetFPGA. International Journal of Future Generation Communication and Networking. 7, 141-154.

Zilberman N., Audzevich Y., Kalogeridou G., Manihatty-Bojan N., Zhang J. & Moore A. 2015. NetFPGA: Rapid Prototyping of Networking Devices in Open Source. ACM SIGCOMM Computer Communication Review. 45, pp. 363-364.

Zilberman N. 2014. The Flexible Open-Source Networking Platform. Tech. rep., University of Cambridge.