RECOPILACIÓN CONCEPTUAL E INTRODUCCIÓN A LA PLATAFORMA DE APRENDIZAJE PARA REDES NETFPGA

JOSÉ CARLOS BLANCO DURÁN

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERIAS FÍSICO MECÁNICAS
ESCUELA DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES
ESPECIALIZACIÓN EN TELECOMUNICACIONES
BUCARAMANGA
2011

RECOPILACIÓN CONCEPTUAL E INTRODUCCIÓN A LA PLATAFORMA DE APRENDIZAJE PARA REDES NETFPGA

JOSÉ CARLOS BLANCO DURÁN

Monografía para optar el título de Especialista en Telecomunicaciones

Director

Msc JORGE HERNANDO RAMÓN SUÁREZ

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERIAS FÍSICO MECÁNICAS
ESCUELA DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES
ESPECIALIZACIÓN EN TELECOMUNICACIONES
BUCARAMANGA
2011

AGRADECIMIENTOS

¡A la vida! Por permitirme lograr una meta más.

A mis padres por el apoyo constante y sus sabios consejos.

Pris, gracias por la confianza y el apoyo emocional.

Ing. Jorge Ramón, gracias por creer en el tema y apoyarlo.

CONTENIDO

Pág.

LISTA DE TABLAS

Pág.

LISTA DE FIGURAS

Pág.

LISTA DE ANEXOS

Pág.

GLOSARIO

ALTERA es una de las compañías más grandes y fuertes en el desarrollo de dispositivos lógicos programables. Se le atribuye el desarrollo del primer dispositivo lógico programable (PLD) en el año de 1984.

CPLD (Complex PLD). arreglo complejo de dispositivos lógicos programables unificado. En el mercado se consigue como un circuito integrado.

DRAM (Dinamic RAM), tipo de memoria de acceso aleatorio usada como memoria primaria de un sistema de cómputo debido a la relación costo-velocidad de acceso. Se caracteriza por requerir una actualización de datos cada cierto tiempo para poder mantener la integridad de los datos.

FIREWALL dispositivo de red cuya función principal es la de gestionar la seguridad. Puede estar implementado en software o hardware y requiere de la configuración un conjunto de políticas para su funcionamiento.

GIGAE tecnología que mejora el estándar Ethernet, alcanza velocidades de 1Gbps (10 veces más que Ethernet). La definición esta publicada en los estándares IEEE 802.3ab y 802.3z.

GNOME es uno de los entornos gráficos para los sistemas operativos basados en Unix. Es de carácter libre bajo licencia GNU. Tuvo su primera aparición en el año 1999 y la última versión fue lanzada en septiembre de 2010.

HDL (hardware description language), lenguaje para el diseño de hardware.

IPCORES se refieren a bloques virtuales que tienen una funcionalidad especial; son desarrollados por terceros y pueden ser incluidos como módulos en un proyecto de descripción de hardware.

JTAG puerto de comunicaciones por medio del cual se puede hacer pruebas, programar y depurar un diseño electrónico. Está basado en el estándar IEEE 1149.1.

MAC (Media Access Control), son el conjunto de procesos que hacen que varios dispositivos puedan hacer uso del mismo medio de transmisión de una red.

MODEM (modulator demodulator), es un dispositivo que permite transmitir y recibir señales de información sobre un medio físico.

PCAP es un formato estándar para captura de paquetes.

PCIE (pci express), tecnología basada en el estándar PCI que mejora las características de ancho de banda.

PHY hace referencia a la capa física de un protocolo de red

POWERPC arquitectura de computadoras basada en RISC.

ROUTER dispositivo de red capaz de tomar decisiones para el reenvío de paquetes de datos

SCRIPT archivo de texto que contiene líneas de código de programación. Puede ser ejecutado mediante un intérprete o ser previamente compilado

SIMULACIÓN proceso de verificar la salida de un sistema sin llevarlo a la práctica. En el caso de la programación para FPGAs, este proceso se hace por medio de la aplicación de desarrollo.

SÍNTESIS es el proceso de elaboración del hardware con base en el código escrito.

SOFTCORE procesador elaborado mediante descripción de hardware que puede ser usado como modulo en el desarrollo de un nuevo proyecto de descripción de hardware.

SRAM (Static RAM), tipo de memoria de acceso aleatorio usada como memoria de acceso rápido de un sistema de cómputo. Se caracteriza por no requerir una actualización periódica para mantener la integridad de los datos guardados.

SYNOPSYS aplicación para el diseño de circuitos integrados de aplicación específica (ASIC).

SYSTEMC es una aplicación usada para hacer modelamiento de sistemas en HLD.

TELNET protocolo de red que permite tomar el control de una maquina remotamente como si se estuviera trabajando de forma local.

TRANSCEIVER dispositivo que integra funciones de transmisión y recepción de información.

VERILOG es un lenguaje de descripción de hardware con sintaxis similar a la del ANSI C.

VHDL lenguaje de descripción y modelado de hardware, usado para el diseño de sistemas y componentes electrónicos.

XILINX compañía especializada en el diseño y fabricación de circuitos lógicos programables. Fueron los desarrolladores del concepto y los creadores de las FPGAs.

RESUMEN

TÍTULO: RECOPILACIÓN CONCEPTUAL E INTRODUCCIÓN A LA PLATAFORMA DE APRENDIZAJE PARA REDES NETFPGA *

AUTOR: JOSÉ CARLOS BLANCO DURÁN **

PALABRAS CLAVE:

NETFPGA, herramienta, desarrollo, redes, enseñanza, tecnología.

DESCRIPCIÓN:

Es muy importante para la academia, en especial en el área de ingeniería, poder contrastar los conceptos teóricos que se aprenden en el aula con la práctica. A su vez, poder contar con un instrumento que brinde la posibilidad de desarrollar, simular, y optimizar trabajos, estimula en gran medida la creatividad de estudiantes e investigadores, brindando una pieza fundamental para el desarrollo de tecnología en el país.

En el área de las telecomunicaciones, hasta hace unos años no se contaba con la posibilidad de tener una herramienta que permitiera el desarrollo de tecnología desde las universidades. De esta forma, gran parte del proceso académico se basa en enseñar a usar dispositivos de red comerciales, las cuales solo permiten la interacción con la capa de aplicación del dispositivo.

NETFPGA es una plataforma para el desarrollo de sistemas de red, en la que el usuario puede iniciar con el aprendizaje de los conceptos básicos de red, afianzar sus conocimientos mediante la implementación de ejemplos o desarrollos ya elaborados y diseñar su propio sistema desde el inicio. Algunos trabajos que se pueden hacer sobre esta herramienta son: implementación y optimización de protocolos de red, análisis de rendimiento de equipos de red, diseño de equipos de red, entre otros.

El presente trabajo contiene la descripción de esta tecnología, expone los requerimientos de hardware y software para su implementación y expone la manera de usarla por medio del análisis de ejemplos.

---

* Proyecto de grado

** Facultad de Ingenierías Físico Mecánicas, Escuela de Ingenierías Eléctrica, Electrónica y Telecomunicaciones. Especialización en Telecomunicaciones. RAMÓN SUÁREZ, Jorge

SUMMARY

TITTLE: CONCEPT COLLECTION AND INTRODUCTION TO THE LEARNING PLATFORM FOR NETWORK NETFPGA *****

AUTHOR: JOSÉ CARLOS BLANCO DURÁN **

KEYWORDS:

NETFPGA, development tool, networking, education, technology.

DESCRIPTION:

It is very important for the academy, especially in the engineering field, to contrast the theoretical concepts learned in the classroom with practice. In turn, to have an instrument that provides the ability to develop, simulate, and optimize work, strongly stimulates the creativity of students and researchers, providing a cornerstone for the development of technology in the country.

In the area of telecommunications, until recently not had the chance to have a tool that allowed the development of technology from universities. Thus, much of the academic process is based on teaching using commercial network devices, which only allow interaction with the application layer of the device.

NETFPGA is a platform for the development of network systems, in which the user can start with learning the basics of network, enhance their knowledge by implementing ready-made examples or developments and design your own system from scratch. Some jobs you can do about this tool are: implementation and optimization of network protocols, performance analysis, network equipment, network equipment design, among others.

This work contains the description of this technology and discusses the hardware and software requirements for implementation and discusses how to use it through the analysis of examples.

---

* Work Degree

** Physical & Mechanical Faculty. Electrical, Electronic and Telecomunications School. RAMÓN SUÁREZ, Jorge

INTRODUCCIÓN

Durante la formación como profesionales en telecomunicaciones, se desarrollan habilidades en configuración de equipos de red como *modems, routers, firewall*, entre otros. El proceso básicamente consiste en conocer las opciones que ofrece el fabricante del equipo y establecer la configuración que mejor se acomode a la distribución de red ó la necesidad que se desea suplir. Es una práctica que se vuelve común y repetitiva en los profesionales que se dedican al montaje y configuración de redes.

Toda configuración de equipos comerciales para redes se hace a nivel de software, ya sea por medio de comandos (telnet) o alguna interfaz gráfica (web). Independientemente del modo, únicamente se permite modificar parámetros que el fabricante del equipo haya autorizado. De esta manera resulta imposible utilizar un equipo comercial para probar nuevos protocolos, configuraciones experimentales y algunos otros ensayos que tienen un valor muy significativo en la academia y la investigación, donde nacen las bases de las nuevas tecnologías.

El presente trabajo, está elaborado con el propósito de dar a conocer la plataforma NETFPGA, la cual es una tecnología que sirve como herramienta de desarrollo y experimentación de equipos y protocolos de red, esta herramienta ha sido muy aceptada a nivel mundial y es muy importante que en Colombia, especialmente las universidades, empiecen a desarrollar trabajos basados en ella.

1. PLATAFORMA DE DESARROLLO NETFPGA

1.1. HISTORIA

NETFPGA es un proyecto iniciado por un grupo de investigación de la universidad de Stanford, California, surge por la necesidad de brindar una plataforma de desarrollo que sirviera de herramienta para la enseñanza de la capa física y la capa de enlace en redes de computadoras, debido a que los estudiantes sólo ganaban destrezas por medio de la experimentación en las capas superiores. La oportunidad de desarrollo surge con la popularización de sistemas de hardware personalizados basados en FPGAs, y el reto consistió en implementarlos en el desarrollo de dispositivos de red

El proyecto inició en el 2001 y en el año de 2003 surgieron los primeros prototipos, los cuales fueron usados en un proyecto de post-grado en la universidad de Stanford. El proyecto consistió en un *rack* de tarjetas NETFPGA versión 1 en el que se podían descargar y depurar los diseños remotamente.

Luego se lanzó la versión 2 en el año 2005, la cual se popularizó por todo el mundo y es la versión más usada en la actualidad. A la fecha se han despachado alrededor de 2.000 unidades a nivel mundial.

En el 2010 se lanza la última de las versiones: NETFPGA 10G la cual conserva lo mejor de la anterior versión y agrega mucha más funcionalidad y poder de procesamiento.

1.2. VERSIONES DE LA PLATAFORMA NETFPGA

1.2.1. NETFPGA V1

Esta versión estaba constituida por 3 FPGAS ALTERA EP20K400, una de ellas actuaba como control (CFPGA) y las dos restantes eran para el usuario (UFPGA), un controlador Ethernet de 8 puertos y tres SRAM de 1MB. La única forma de comunicación con la tarjeta era por medio de la interfaz Ethernet. Los estudiantes elaboraban su diseño por medio de la herramienta Synopsys para la síntesis y simulación, y la herramienta Altera Quartus para el proceso "place and route"[1]. Al finalizar el desarrollo se cargaba el archivo de configuración a la tarjeta NETFPGA v1 por medio de una interfaz web. En este punto se podían enviar paquetes hacia la tarjeta y recibir los paquetes emitidos por ella. Por último se planteó un laboratorio en el que se interconectaba la tarjeta a la red de internet del campus universitario para implementar un analizador de tráfico.

Este sistema funciono muy bien y cumplió con el objetivo trazado al inicio del proyecto, también sirvió como base para ofrecer una capacitación llamada CS344 "*Build your Own IP Router*", en la cual se enseñaba a los estudiantes a diseñar, construir y experimentar su propio enrutador. Una vez tenían el enrutador montado, los estudiantes le agregaban alguna funcionalidad extra a sus diseños como un servidor web, servidores NAT entre otros.

Limitaciones de la versión 1.0:

Aunque la versión 1 obtuvo mucho más éxito que el que se esperaba, con el tiempo fueron notables las limitaciones que tenía:

---

[1] Proceso mediante el cual se establecen en interconectan los elementos en una FPGA

- El formato del circuito impreso requería un *rack* especial y el montaje tanto de los componentes como la instalación en el chasis se hacía de manera manual.
- Las interfaces de red eran de baja velocidad, ocho puertos de 10Mb/s.
- No poseía una CPU, así los diseños que se desarrollaban no podían ser muy sofisticados.
- La plataforma de desarrollo solamente estaba basada en sistemas Linux y Solaris.

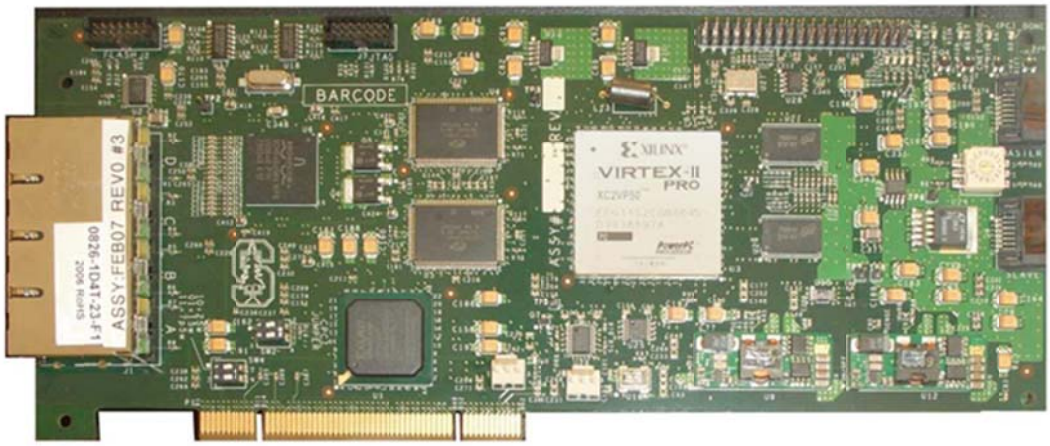
1.2.2. NETFPGA 1G

Los primeros prototipos de la versión 2 aparecieron en el año 2005, y las mejoras con respecto a su antecesor eran trascendentales.

La versión 2 trajo consigo la integración de un bus de datos PCI de 32bits y 33MHz, una FPGA Xilinx programada para gestionar la comunicación PCI, y una FPGA Xilinx Virtex II Pro para el usuario. Cuatro interfaces de red 10/100/1000 y dos interfaces seriales de alta velocidad SATA para interconexión entre tarjetas.

El diseño del circuito impreso se hizo para que fuera fácilmente instalable en un *socket* PCI estándar y solucionó mucho de los inconvenientes que se tenían en el montaje tanto de la tarjeta como del chasis. Por medio de la interfaz PCI, se brinda un canal de transferencia de datos de alta velocidad, de la misma manera provee la alimentación y *el Master Reset* de la tarjeta.

Figura 1. Tarjeta NETFPGA 1G.



Fuente: http://netfpga.org

Los entornos de desarrollo soportados por esta versión incluyen: Synopsys VCS (Linux) y ModelSim (Linux, XP), el proceso “place and route” es realizado con la herramienta Xilinx ISE la cual tiene soporte para Linux y WinXP.

Dada la popularidad, vigencia y estabilidad de este desarrollo, será la versión que se describa y profundice en este documento.

1.2.3. NETFPGA 10G

NETFPGA 10G se lanzó en el segundo semestre de año 2010, la característica principal de esta nueva versión es el cambio de la FPGA de usuario, paso de la VirtexII de la serie 1G a la Virtex5, este cambio implicó principalmente mayor número de compuertas para los desarrollos de usuario. Adicionalmente, tiene más capacidad de memoria: 27 MB de SRAM y 288MB de DRAM. Las interfaces de red son de 10Gbps, y la interfaz con el host cambio a PCI Express generación 2 de 5Gbps.

Figura 2. Tarjeta NETFPGA 10G

Fuente: http://netfpga.org

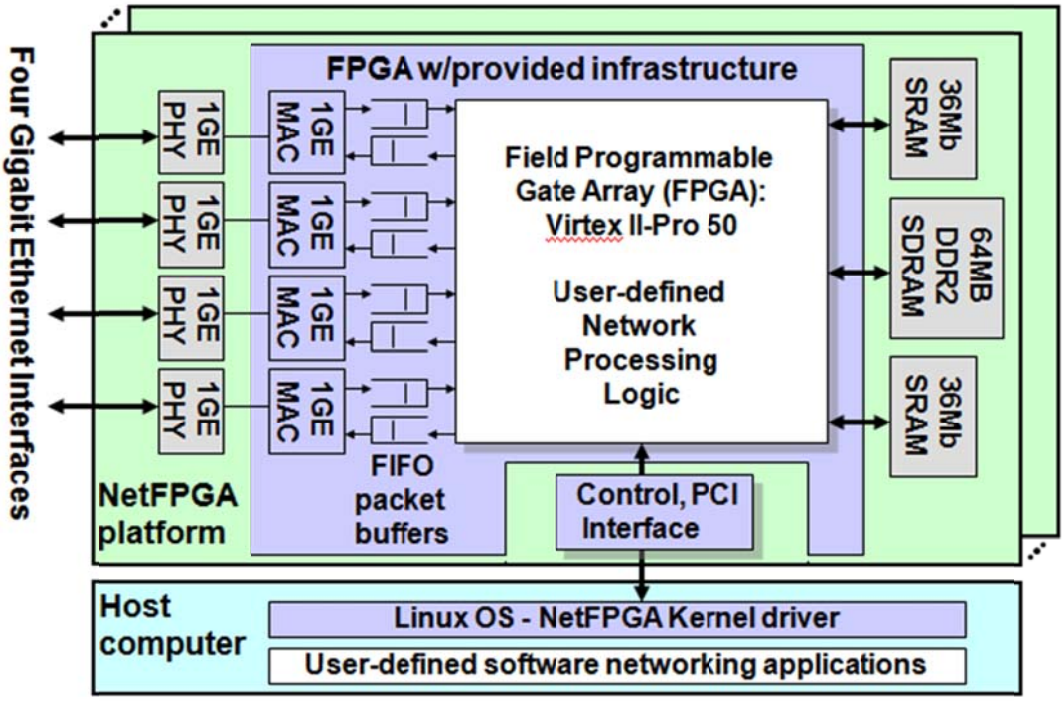
El diseño del hardware para tarjeta puede ser elaborado tanto en sistemas Linux como Windows. El soporte y los desarrollos que genera el equipo de trabajo *NETFPGA Group* están elaborados sólo bajo plataforma Linux. Hasta la fecha no se ha implementado la programación de la tarjeta a través del puerto PCI-Express, por lo que es requerido para la programación, un cable USB II de Xilinx.

1.3. COMPOSICIÓN HARDWARE DE NETFPGA 1G

La plataforma NETFPGA es de carácter abierta y puede ser usada y modificada por cualquiera que lo desee. La información de hardware (planos esquemáticos y diseño del circuito impreso), está disponible al público a través de la página web: http://netfpga.org/ .

A continuación se presenta el diagrama de bloques de la tarjeta NETFPGA 1G; luego se presenta la definición de cada una de las partes principales.

Figura 3. Diagrama de bloques Sistema NETFPGA-1G



Fuente: http://netfpga.org

1.3.1. FPGA

FPGA son las siglas de *Field Programmable Gate Array*, "Son dispositivos lógicos de propósito general programables por los usuarios, compuestos por bloques lógicos comunicados por bloques programables" [2].

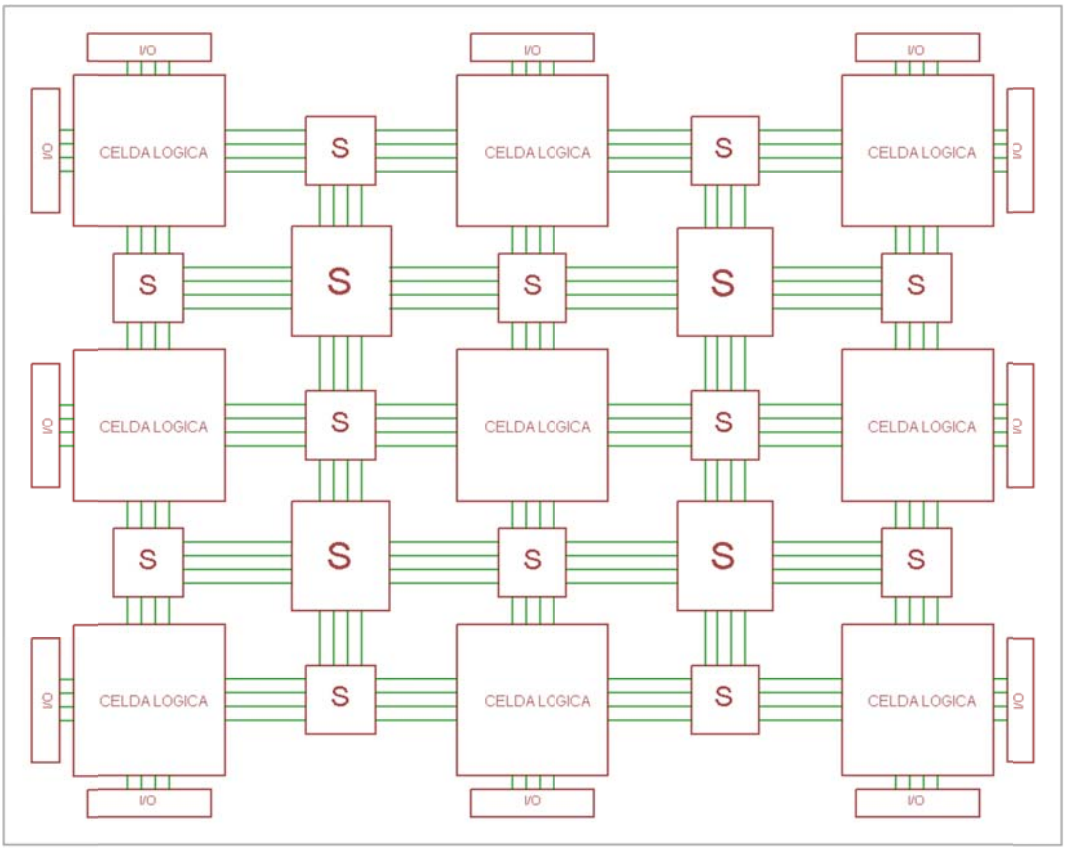
El arreglo de compuertas es bi-dimensional y la programación se hace a través de *switchs*. Ver figura 4.

---

[2] SÁNCHEZ SUÁREZ, Gabriel. Microelectrónica [online]. Universidad Francisco de Paula Santander [Cúcuta, Colombia]: [citado 28 de marzo de 2011; 15:00:00]. Disponible en Internet: http://www.ufps.edu.co/materias/uelectro/htdocs/pdf/fpga.pdf

Las primeras FPGAs fueron presentadas en el año de 1985 por XILINX, y en la actualidad empresas como ALTERA, ATMEL, CYPRESS, ACTEL tienen sus propios desarrollos y compiten por la popularización de estos. La tarjeta NETFPGA está basada en FPGAs de XILINX, las cuales además son las más trabajadas en Colombia.

Figura 4. Estructura conceptual de una FPGA.



Fuente: Autor

1.3.1.1. Características de las FPGAs

La ventaja principal de las FPGAs sobre otros tipos de tecnologías programables basadas en procesador como los microcontroladores, microprocesadores y DSPs, radica en que en éstas se pueden establecer tareas paralelas ya que la programación es a nivel de hardware y no por medio de instrucciones como en los procesadores. Esta característica es la que permite que sean empleadas en tareas complejas como la generación y procesamiento de video en tiempo real, prototipos de circuitos digitales (emulación de procesadores), controladores de bus PCI y gestión de redes de datos. La programación de estos dispositivos se inicia escribiendo el código en un lenguaje de programación. Los más populares son VHDL y VERILOG. Luego este código se valida y se transfiere al chip por medio del puerto JTAG.

Las FPGAs también se caracterizan por brindar:

- Reducción de tiempo de desarrollo.
- Integración de sistemas digitales.
- Alta capacidad de procesamiento.
- Prototipado rápido y pruebas de implementación real.
- Paralelismo en la ejecución de procesos.

1.3.1.2. Características FPGA VIRTEX II PRO 50

La VIRTEX II PRO 50 es el núcleo de la tarjeta NETFPGA 1G. Este chip tiene 53.136 celdas lógicas para el diseño del usuario, 4.176 Kbit de RAM interna, 738 Kbit de SRAM, 232 bloques multiplicadores, 2 procesadores PowerPC internos y 852 pines de entrada/salida.

La familia Virtex II PRO, permite diseños basados en IPCores[3] y diseños personalizados. Los bloques PowerPC y la interfaz multigigabit la hacen ideal en aplicaciones de alto desempeño como DSP, comunicaciones y video.

1.3.2. Memoria DDR2 y SRAM

La memoria DDR2, es una de las últimas tecnologías de memorias de acceso aleatorio (RAM). DDR significa: “Dual Data Rate Synchcronous DRAM”.

Sus características principales son:

- Anchos de banda entre 3.2 y 9.6 Gbits/seg.
- Velocidades de operación entre 400-1200MHz.
- Bajo coste de fabricación.
- Menor disipación térmica que sus antecesoras.

En la NETFPGA, la memoria DDR2 sirve como medio de almacenamiento para los desarrollos que se corren desde la capa física. La memoria es de 64MB y está configurada para almacenar 16 millones palabras de 32 bits. Esta memoria tiene un reloj independiente de 200MHz que permite transferencias de datos de 1600MB/s.

La tarjeta NETFPGA, también dispone de 2 bancos de memoria SRAM, de 36MB en configuración de 512.000 palabras de 36 bits. Esta memoria trabaja en forma síncrona con el reloj principal de 125 MHz.

---

[3] Término para definir un bloque funcional en descripción de hardware. Son componentes electrónicos sin parte física y son protegidos por derechos de propiedad intelectual.

1.3.3. Interfaz de red PHY

NETFPGA 1G cuenta con un Ethernet transceiver de 4 puertos on-chip BCM5464SR de la empresa Broadcom. Este es el medio físico con que los desarrollos que se guarden en la tarjeta puedan enviar y/o recibir paquetes. Las interfaces de red son compatibles con cableados 5, 5e y 6.

La implementación de los controladores de acceso al medio (MACs) de los cuatro puertos están implementadas como un Soft-Core en la FPGA.

1.3.4. Puerto de comunicación PCI

Una de las principales razones del éxito de la tarjeta NETFPGA 1G, ha sido la compatibilidad de interconexión con un host por medio de la interface PCI.

La interfaz PCI, es un bus de interconexión de dispositivos. PCI significa: “Peripheral Component Interconnect”, es un estándar que surgió en el año de 1991, aun en la actualidad se mantiene vigente y en constante evolución.

Las características principales de la interfaz PCI estándar usada en la tarjeta NETFPGA son:

- Ancho del bus de datos: 32 bits.
- Velocidad del bus: 33MHz.
- Velocidad de transferencia: 132MB/s.
- Soporta Plug and Play (PnP).

1.3.5. Interfaz serial de alta velocidad SATA

SATA: “Serial Advanced Technology Attachment”, es una interfaz de comunicación que surge de la necesidad de aumentar la velocidad de transferencia de unidades

de disco duro y ópticas para computadoras, ya que con el estándar vigente en la época (PATA) se tenía un límite de 100MB/s. En el 2002 se lanza SATA/150 como la alternativa, ofreciendo velocidades de hasta 150MB/s y en poco tiempo evoluciona a la versión actual, SATA/300 con velocidades de transferencia hasta de 300MB/s.

La tarjeta NETFPGA 1G posee 2 conectores SATA para la interconexión entre tarjetas NETFPGA sin usar el puerto PCI. No se pretende la interoperabilidad con otros dispositivos SATA como discos duros o unidades ópticas.

1.4. HARDWARE REQUERIDO

La tarjeta NETFPGA 1G se puede adquirir de manera independiente o en conjunto con un sistema de cómputo.

A continuación se describen los requerimientos mínimos del sistema de cómputo y las opciones completas que se encuentran actualmente en el mercado.

1.4.1. Sistema de cómputo sugerido.

Los requerimientos del sistema de cómputo que permita alojar la tarjeta e desarrollo NETFPGA 1G según las recomendaciones del fabricante y teniendo en cuenta la fecha de realización de este documento son:

- Chasis ATX o microATX.
- MotherBoard Gigabyte MA78GM-US2H mATX (Quad Core).
- Procesador AMD X4 940 3GHz.
- Memoria DDR2 mínimo de 2GB.
- Lector DVD para disco de arranque.
- Tarjeta de red adicional PCI Express x4 Intel Pro/1000 Dual Port.

- Disco duro mínimo de 250GB.
- Cables Ethernet Cat5E o Cat6.
- Monitor LCD 19”.
- Cables de alimentación para CPU y Monitor.
- Cable SATA.

Debido a que la tarjeta NETFPGA 1G es compatible con el puerto PCI, es muy probable que se pueda instalar en cualquier MotherBoard con características similares, e inclusive mejores a la descrita anteriormente. Se debe tener en cuenta que *NETFPGA Group* no se hace responsable por la compatibilidad del sistema en configuraciones de equipos diferentes a los que públicamente sugieren en el sitio web.

En la actualidad se dispone de mejores opciones de las partes que componen el sistema de cómputo. Como síntesis se enuncian las características que deben tener en cuenta para ensamblar su propio sistema de cómputo para la tarjeta NETFPGA 1G:

- Escoger la MotherBoard según la familia del procesador que se vaya a trabajar (INTEL/AMD).
- MotherBoard con soporte de procesadores de dos o cuatro núcleos.
- Procesador de dos o cuatro núcleos según la MotherBoard que se elija.
- Memoria RAM DDR2 o DDR3.
- Disponibilidad de un puerto PCI para la NETFGA.
- Disponibilidad de un puerto PCI Express para una tarjeta de red de 2 puertos Gigabit.
- Tarjeta de red PCIe de 2 puertos.
- Unidades de disco duro y DVD SATA.

1.4.2. Sistemas de cómputo pre-ensamblados.

Existen máquinas pre-ensambladas que ya han sido probadas y tiene el soporte y aval del NETFPA Group, a continuación se enuncian y se describen sus características principales:

1.4.2.1. DELL 2950

El DELL 2950 hace parte de la novena familia de servidores PowerEdge de DELL Inc. La tarjeta NETFPGA debe ir instalada en el puerto PCI-X que trae este equipo.

Figura 5. Vista posterior DELL 2950

Fuente: http://www.dell.com

Características de sistema:

- Chasis 2U.
- Soporta 2 Procesadores 2Core Intel Xeon 5000-5300.
- Velocidad del bus frontal: 667-1066MHz.
- Cache desde 2x2MB hasta 2x4MB.
- Hasta 32GB de memoria FDB.
- 2 puertos PCI-X y un PCI Express x8.
- Controlador Drive de 4 puertos SAS 5/i .
- Controlador RAID PERC 4e/DC.
- Hasta 8 bahías para disco duro de 2.5”.
- Chipset Intel 5000X

La empresa ACCENT TECHNOLOGY tiene en el mercado dos configuraciones de equipos que cumplen con los requerimientos de la Universidad de Stanford para el correcto funcionamiento de la tarjeta NETFPGA 1G. Los sistemas ya traen el sistema operativo CentOS y los paquetes necesarios para ejecutar la prueba inicial de la plataforma. Los productos ofrecidos por ACCENT son los siguientes:

1.4.1.2. ACCENT NETFPGA – Cube

Este producto viene en versión de dos núcleos y cuatro núcleos. Viene pre-instalado y pre-configurado para empezar a trabajar con la plataforma NETFPGA. Es apropiado para pruebas iniciales de escritorio.

Figura 6. NETFPGA Cube de Accent Technologies

Fuente: http://www.accenttechnologyinc.com/

Características de NETFPGA Cube Dual-Core

- Procesador AMD X2 6000+.
- Motherboard ASUS M3N78-VM.
- Sistema Operativo CentOS.
- 2GB de memoria DDR2.
- Disco Duro de 250GB.
- Unidad óptica DVD R/W.
- Tarjeta NETFPGA 1G de DIGILENT.
- Diseño del chasís en forma de cubo.
- Tarjeta de Red dual Intel Pro/1000 PT

Características de NETFPGA Cube Quad-Core

- Procesador AMD Phenom Quad Core 9650
- Motherboard ASUS M3N78-VM.
- Sistema Operativo CentOS.
- 4GB de memoria DDR2.
- Disco Duro de 500GB.
- Unidad óptica DVD R/W.
- Tarjeta NETFPGA 1G de DIGILENT.
- Diseño del chasís en forma de cubo.
- Tarjeta de Red dual Intel Pro/1000 PT

1.4.1.3. ACCENT NETFPGA - 1U

Figura 7. NETFPGA 1U de Accent Technologies

Fuente: http://www.accenttechnologyinc.com/

Este modelo viene en Chasis 1U, el cual permite que se integre a un *RACK* para interoperar con otros equipos de red como *routers, switchs o módems*. También viene en versiones de doble y cuatro núcleos.

Características de NETFPGA 1U Dual-Core

- Procesador INTEL dual core E6420
- Motherboard SUPERMICRO PDSBM-LN2+.
- Sistema Operativo CentOS.
- 2GB de memoria DDR2.
- Disco Duro de 250GB.
- Unidad óptica DVD R/W.
- Tarjeta NETFPGA 1G de DIGILENT.
- Chasis Supermicro 1U.
- Tarjeta vertical 1U

Características de NETFPGA 1U Quad-Core

- Procesador INTEL Quad Core Q9300
- Motherboard Supermicro X7SBL-LN2.
- Sistema Operativo CentOS.
- 4GB de memoria DDR2.
- Disco Duro de 500GB.
- Unidad óptica DVD R/W.
- Tarjeta NETFPGA 1G de DIGILENT.
- Chasis Supermicro 1U.
- Tarjeta vertical 1U

1.5. SOTFWARE REQUERIDO

La plataforma NETFPGA 1G es un conjunto hardware-software. Ya se ha descrito el hardware necesario para su funcionamiento. A continuación de describirá el software que se requiere para poder elaborar desarrollos en ella.

1.5.1. Sistema operativo

Es el componente software principal de la plataforma. En él se instalaran las demás aplicaciones que permitirán el diseño, comunicación y ejecución de los desarrollos.

NETFPGA Group recomienda el sistema operativo CentOS en distribuciones de 32bits (CentOS 4.4 a CentOS 5.4). Esto no es premisa que la plataforma no pueda usar un sistema operativo diferente, incluso Microsoft publicó el driver para trabajar la tarjeta NETFPGA bajo Windows. También existe una distribución de Fedora que incluye el driver y el paquete de instalación de la tarjeta NETFPGA en un LiveCD.

CentOS es un sistema operativo servidor que surgió de la re-compilación del código fuente liberado por Red Hat Enterprise Linux (RHEL). Desde su primera versión lanzada el 14 de mayo de 2004 ha empezado a ganar popularidad como sistema operativo servidor. La última versión de CentOS al momento de escribir este documento es la 5.5 liberada el 16 de mayo de 2010.

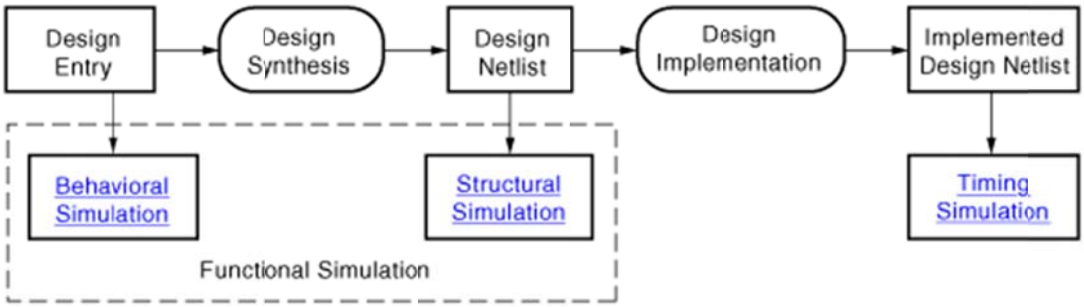
Las ventajas de este sistema operativo con respecto a otras distribuciones Linux son: amplia comunidad de apoyo que colaboran en su desarrollo conformada principalmente por administradores de red, usuarios empresariales y entusiastas los cuales están comprometidos con la causa ya que los errores que los usuarios finales detectan pueden suceder en sus propios sistemas; los parches se lanzan

rápidamente, inclusive hay empresas que se dedican a dar soporte comercial del producto.

1.5.2. Aplicaciones para el desarrollo de hardware.

El diseño de hardware para la plataforma NETFPGA requiere de herramientas que permitan el diseño, la simulación, la síntesis y la grabación de la FPGA de usuario (Virtex II Pro). Debido a que esta FPGA es de XILINX, se deberá diseñar y sintetizar con el entorno de desarrollo XILINX ISE.

Figura 8. Proceso de desarrollo de hardware para FPGA



Fuente: http://www.xilinx.com/

El proceso de simulación puede estar soportado por la aplicación ModelSim y ChipScope Pro; por último, el archivo resultante del proceso de diseño del hardware se carga en la FPGA a través del driver de la tarjeta por medio del puerto PCI. Este archivo recibe el nombre de Gateware.

1.5.2.1. Xilinx ISE

Es el entorno de desarrollo para los diseños basados en FPGAs y CPLDs de XILINX. El software ofrece la síntesis y simulación en HLD, la implementación, los ajustes y la programación del dispositivo desde la misma interfaz. Existen versiones para Windows y Linux. La versión actual es la 13.1.

El fabricante ofrece una versión básica con las funciones antes mencionadas de manera gratuita, pero existen otras distribuciones que ofrecen al usuario herramientas extras.

Tabla 1. Distribuciones de la aplicación XILINX ISE

ISE Design Suite Comparison Table	ISE WebPACK Tool (Device Limited)	Logic Edition	Embedded Edition	DSP Edition	System Edition
ISE Foundation™ Tools with ISE Simulator (ISim)	√	√	√	√	√
PlanAhead™ Design Analysis Tool	√	√	√	√	√
ChipScope™ Pro Logic Analyzer		√	√	√	√
ChipScope Pro Serial I/O Toolkit		√	√	√	√
Embedded Development Kit (EDK)			√		√
Software Development Kit (SDK)			√		√
System Generator for DSP				√	√

Fuente http://www.xilinx.com/

1.5.2.2. ModelSim DE

Modelsim DE es una aplicación elaborada por Mentor Graphics que gestiona un entorno de desarrollo que permite: editar, compilar, simular y depurar diseños de

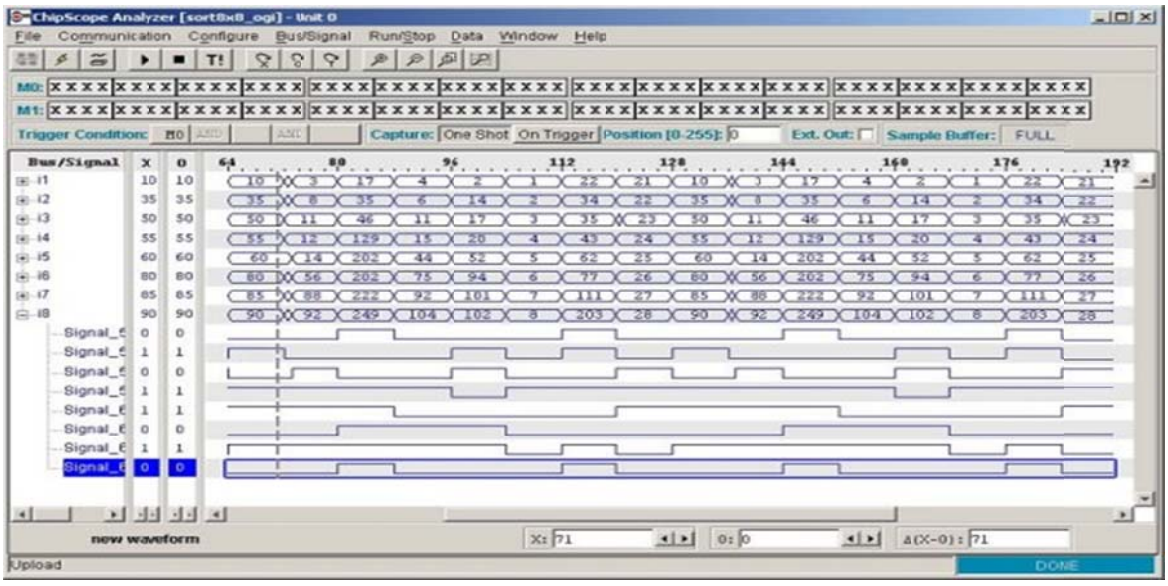
sistemas digitales descritos en VHDL, Verilog y SystemC. La versión actual es la 10.0 y es compatible con sistemas Windows y Linux.

1.5.2.3. ChipScope PRO

Es una aplicación de XILINX que integra un analizador lógico, un analizador de red y la implementación virtual de pines de entrada/salida para la depuración de proyectos basados en FPGAs. Permite monitorear cualquier señal (interna o externa) en tiempo de ejecución; las señales capturadas se muestran en una interfaz gráfica para su análisis. Esta aplicación permite también calcular la tasa de errores de bit (BER) en múltiples canales al tiempo.

Para su funcionamiento requiere sólo el puerto JTAG de la board.

Figura 9. Interfaz de la aplicación CHIPSCOPE de XILINX



Fuente: http://raintown.org/lava/

2. INSTALACIÓN Y PUESTA EN MARCHA DE LA PLATAFORMA NETFPGA 1G

Para el correcto funcionamiento de la plataforma NETFPGA 1G se requiere de un sistema de cómputo que tenga preferiblemente el sistema operativo CentOS 5.0 y las aplicaciones para desarrollo Xilinx ISE 10, ModelSim DE y ChipScope Pro.

Este documento no pretende elaborar una guía de instalación de estos programas, se enfocara principalmente en las recomendaciones de instalación de los mismos, la instalación de los drivers para la tarjeta NETFPGA y las pruebas preliminares de funcionamiento.

2.1. CONSIDERACIONES INICIALES DE INSTALACIÓN

2.1.1. Instalación del sistema operativo CentOS

La versión NETFPGA 1G trabaja sólo con sistemas operativos de 32 bits, en el momento de escribir este documento la versión actual de CentOS es la 6.

Al instalar el sistema operativo se deben tener en cuenta las siguientes recomendaciones:

- Usar GNOME como entorno de escritorio.
- No instalar la Virtualización.
- Configurar la Seguridad en: “Not enforcing”.
- Conectar la interfaz Eth0 (tarjeta de red integrada en la Motherboard) a la red externa para que tome los parámetros de DHCP, IP, Gateway y DNS.
- Aplicar las actualizaciones del sistema operativo.

Luego de que el sistema operativo este instalado, se debe entrar como administrador (root), e instalar el paquete RPMforge, para la visualización multimedia.

2.1.2. Instalación de herramientas CAD

La instalación de herramientas CAD: Xilinx ISE, ModelSim se debe hacer para poder compilar, simular y sintetizar el código fuente escrito en verilog. Las pruebas de los códigos fuentes que están disponibles en la página web http://netfpga.org/ indican la versión de las herramientas donde fueron escritas y sólo tienen soporte bajo esa versión.

Si aún no piensa trabajar con el diseño de hardware en verilog es posible omitir la instalación de estas herramientas.

La mayor parte del código escrito para el diseño de hardware para la versión 1G que está disponible en el sitio web de *NETFPGA Group* fue escrito en Xilinx ISE v10.1 SP3.

Las recomendaciones de instalación para la herramienta Xilinx ISE son:

- Instalar la versión 10.1 disponible en la página web del fabricante: http://www.xilinx.com/ise/
- Instalar el *service pack 3* de la aplicación.
- Obtener una licencia comercial o académica, en caso de ser esta última, especifique en el formulario de solicitud el uso de NETFPGA.

Las recomendaciones de instalación de la herramienta ModelSim son:

- Bajar la aplicación del sitio web del fabricante: http://www.model.com/
- Instalar la versión DE 6.6, (las versiones DE son las únicas que tienen compatibilidad en sistemas Linux).

Las recomendaciones para instalar la aplicación ChipScope PRO son:

- Si se adquirió una versión comercial del Xilinx ISE, verifique si la herramienta ChipScope está contenida en el paquete.
- Bajar la aplicación desde el sitio web del fabricante: http://www.xilinx.com/tools/cspro.htm
- Adquirir el cable USB JTAG para poder hacer el debugger.

2.2. INSTALACIÓN DE DRIVER Y CÓDIGO FUENTE

2.2.1. Descarga del paquete de instalación de la NETFPGA

A continuación se describen los pasos para la descarga del driver que servirá como puente de comunicación entre la tarjeta NETFPGA 1G y el sistema de cómputo (host).

Primero, se deben cargar los paquetes al administrador de software de CentOS (yum) desde la consola.

```
rpm -Uhv http://netfpga.org/yum/el5/RPMS/noarch/netfpga-repo-1-1_CentOS5.noarch.rpm
```

Observe que el paquete está referido directamente de la página de *NETFPGA Group*, por lo que el equipo deberá tener salida a internet.

Luego se procede a la instalación de los paquetes descargados. El comando que se debe escribir en la consola es el siguiente:

```
yum install netfpga-base
```

Al iniciar, se pedirá la instalación de otros paquetes de dependencia los cuales hay que aceptar indicándolo mediante la letra ‘y’.

Se procede con la creación del directorio de trabajo, el cual debe estar alojado en la cuenta de usuario. La creación de éste directorio se hace a través de un script.

Como recomendación antes de ejecutar el script deberá hacer un backup del directorio netfpga si ya está creado previamente ya que el comando borrara cualquier información que este en el mismo.

La línea de comandos que se debe escribir en la consola para correr el script es:

```
/usr/local/netfpga/lib/scripts/user_account_setup/user_account_setup.pl
```

Este script además de crear el directorio /root/netfpga, (donde root es el usuario), también agrega variables de entorno al archivo .bashrc, estas son:

- NF_ROOT
- NF_DESIGN_DIR
- NF_WORK_DIR
- PYTHONPATH
- PERL5LIB

Por último se debe reiniciar el equipo y se dará por terminada la descarga local del driver.

2.2.2. Instalación de módulos de memoria para simulación

Para la simulación de procesos de lectura y escritura en las memorias SRAM y DDR2 de la tarjeta NETFPGA, se deben copiar los archivos:

- ddr2_parameters.vh
- ddr2.v
- cy7c1370d.v

En el directorio:

```
netfpga/lib/verilog/core/common/src
```

2.2.3. Compilación y carga del driver

Una vez guardado, se debe compilar el driver para poder cargarlo. El proceso de compilación se hace a través del siguiente comando en consola:

```
cd ~/netfpga/  make
```

La respuesta que indica que el proceso se hizo correctamente se describe a continuación:

```
make -C C
make[1]: Entering directory `/home/gac1/temp/NF2/lib/C'
make -C kernel
make[2]: Entering directory `/home/gac1/temp/NF2/lib/C/kernel'
make -C /lib/modules/2.6.9-55.0.9.ELsmp/build
M=/home/gac1/temp/NF2/lib/C/kernel
LDDINC=/home/gac1/temp/NF2/lib/C/kernel/../include modules
make[3]: Entering directory `/usr/src/kernels/2.6.9-55.0.9.EL-smp-i686'
  Building modules, stage 2.
  MODPOST
make[3]: Leaving directory `/usr/src/kernels/2.6.9-55.0.9.EL-smp-i686'
make[2]: Leaving directory `/home/gac1/temp/NF2/lib/C/kernel'
make -C download
make[2]: Entering directory `/home/gac1/temp/NF2/lib/C/download'
make -C ../common
make[3]: Entering directory `/home/gac1/temp/NF2/lib/C/common'
```

```
make[3]: Nothing to be done for `all'.
make[3]: Leaving directory `/home/gac1/temp/NF2/lib/C/common'
make[2]: Leaving directory `/home/gac1/temp/NF2/lib/C/download'
make -C reg_access
make[2]: Entering directory `/home/gac1/temp/NF2/lib/C/reg_access'
make -C ../common
make[3]: Entering directory `/home/gac1/temp/NF2/lib/C/common'
make[3]: Nothing to be done for `all'.
make[3]: Leaving directory `/home/gac1/temp/NF2/lib/C/common'
make[2]: Leaving directory `/home/gac1/temp/NF2/lib/C/reg_access'
make -C router
make[2]: Entering directory `/home/gac1/temp/NF2/lib/C/router'
gcc -lncurses  cli.o ../common/nf2util.o ../common/util.o
../common/reg_defines.h   -o cli
gcc -lncurses  regdump.o ../common/nf2util.o ../common/reg_defines.h   -o
regdump
gcc -lncurses  show_stats.o ../common/nf2util.o ../common/util.o
../common/reg_defines.h   -o show_stats
make[2]: Leaving directory `/home/gac1/temp/NF2/lib/C/router'
make[1]: Leaving directory `/home/gac1/temp/NF2/lib/C'
make -C scripts
make[1]: Entering directory `/home/gac1/temp/NF2/lib/scripts'
make -C cpci_reprogram
make[2]: Entering directory
`/home/gac1/temp/NF2/lib/scripts/cpci_reprogram'
make[2]: Nothing to be done for `all'.
make[2]: Leaving directory `/home/gac1/temp/NF2/lib/scripts/cpci_reprogram'
make -C cpci_config_reg_access
make[2]: Entering directory
`/home/gac1/temp/NF2/lib/scripts/cpci_config_reg_access'
make[2]: Nothing to be done for `all'.
make[2]: Leaving directory
`/home/gac1/temp/NF2/lib/scripts/cpci_config_reg_access'
make[1]: Leaving directory `/home/gac1/temp/NF2/lib/scripts'
```

Si se obtiene el siguiente error:

```
make: * /lib/modules/2.6.9-42.ELsmp/build: No such file or directory. Stop.
```

Sera necesario tener las fuentes del kernel para poder compilar el driver. La versión del kernel del sistema operativo CentOS usado es la 2.6.18.164+.

Ahora, se debe instalar el driver entrando al directorio:

```
/lib/modules/`uname -r`/kernel/drivers/nf2.ko
```

Y ejecutando el comando:

```
make install
```

La respuesta que se espera es:

```
 for dir in lib bitfiles projects/scone/base projects/selftest/sw ; do \
        make -C $dir install; \
 done
 make[1]: Entering directory `/home/gac1/temp/NF2/lib'
 for dir in C scripts java/gui ; do \
        make -C $dir install; \
 done
 make[2]: Entering directory `/home/gac1/temp/NF2/lib/C'
 for dir in kernel download reg_access router ; do \
        make -C $dir install; \
 done
 make[3]: Entering directory `/home/gac1/temp/NF2/lib/C/kernel'
 make -C /lib/modules/2.6.9-55.0.9.ELsmp/build
M=/home/gac1/temp/NF2/lib/C/kernel
LDDINC=/home/gac1/temp/NF2/lib/C/kernel/../include modules
 make[4]: Entering directory `/usr/src/kernels/2.6.9-55.0.9.EL-smp-i686'
   Building modules, stage 2.
   MODPOST
 make[4]: Leaving directory `/usr/src/kernels/2.6.9-55.0.9.EL-smp-i686'
 install -m 644 nf2.ko /lib/modules/`uname -r`/kernel/drivers/nf2.ko
/sbin/depmod -a
 make[3]: Leaving directory `/home/gac1/temp/NF2/lib/C/kernel'
 make[3]: Entering directory `/home/gac1/temp/NF2/lib/C/download'
 install nf2_download /usr/local/bin
 make[3]: Leaving directory `/home/gac1/temp/NF2/lib/C/download'
 make[3]: Entering directory `/home/gac1/temp/NF2/lib/C/reg_access'
 install regread /usr/local/bin
 install regwrite /usr/local/bin
 make[3]: Leaving directory `/home/gac1/temp/NF2/lib/C/reg_access'
 make[3]: Entering directory `/home/gac1/temp/NF2/lib/C/router'
 make[3]: Nothing to be done for `install'.
 make[3]: Leaving directory `/home/gac1/temp/NF2/lib/C/router'
 make[2]: Leaving directory `/home/gac1/temp/NF2/lib/C'
 make[2]: Entering directory `/home/gac1/temp/NF2/lib/scripts'
 for dir in cpci_reprogram cpci_config_reg_access ; do \
        make -C $dir install; \
 done
 make[3]: Entering directory `/home/gac1/temp/NF2/lib/scripts/cpci_reprogram'
 install cpci_reprogram.pl /usr/local/sbin
 make[3]: Leaving directory `/home/gac1/temp/NF2/lib/scripts/cpci_reprogram'
 make[3]: Entering directory
`/home/gac1/temp/NF2/lib/scripts/cpci_config_reg_access'
 install dumpregs.sh /usr/local/sbin
 install loadregs.sh /usr/local/sbin
 make[3]: Leaving directory
`/home/gac1/temp/NF2/lib/scripts/cpci_config_reg_access'
 make[2]: Leaving directory `/home/gac1/temp/NF2/lib/scripts'
 make[2]: Entering directory `/home/gac1/temp/NF2/lib/java/gui'
 make[2]: Nothing to be done for `install'.
 make[2]: Leaving directory `/home/gac1/temp/NF2/lib/java/gui'
 make[1]: Leaving directory `/home/gac1/temp/NF2/lib'
 make[1]: Entering directory `/home/gac1/temp/NF2/bitfiles'
 for bitfile in CPCI_2.1.bit cpci_reprogrammer.bit ; do \
        install -D -m 0644 $bitfile /usr/local/netfpga/bitfiles/$bitfile ; \

 done
```

```
make[1]: Leaving directory `/home/gac1/temp/NF2/bitfiles'
 make[1]: Entering directory `/home/gac1/temp/NF2/projects/scone/base'
 make[1]: Nothing to be done for `install'.
 make[1]: Leaving directory `/home/gac1/temp/NF2/projects/scone/base'
 make[1]: Entering directory `/home/gac1/temp/NF2/projects/selftest/sw'
 make[1]: Nothing to be done for `install'.
 make[1]: Leaving directory `/home/gac1/temp/NF2/projects/selftest/sw'
```

En este punto, se debe reiniciar el equipo para que cargue el nuevo driver compilado.

En caso de que se necesite remover el driver ya sea para recompilarlo o eliminarlo definitivamente, se debe desinstalar por medio del comando rmmod.

2.3. VERIFICACIÓN DE SOFTWARE Y HARDWARE

2.3.1. Verificación de driver e interfaces de red

Cuando el sistema arranque nuevamente se debe entrar como usuario root y se debe comprobar la correcta carga del driver mediante el comando:

```
lsmod  | grep nf2
```

La cual, si todo es correcto debe entregar la siguiente respuesta:

```
nf2                    28428  0
```

Ahora, se comprobarán las interfaces de red de la tarjeta NETFPGA 1G escribiendo en consola el siguiente comando:

```
ifconfig -a | grep nf2
```

Se entregará una respuesta con la numeración de las interfaces y la dirección MAC de las mismas:

```
nf2c0     Link encap:Ethernet  HWaddr 00:4E:46:32:43:00
nf2c1     Link encap:Ethernet  HWaddr 00:4E:46:32:43:01
nf2c2     Link encap:Ethernet  HWaddr 00:4E:46:32:43:02
nf2c3     Link encap:Ethernet  HWaddr 00:4E:46:32:43:03
```

2.3.2. Reprogramación del CPCI

El CPCI es “Core” PCI de la FPGA, es la parte hardware que gestiona la interfaz PCI entre la tarjeta y el host. El intercambio de información entre el host y la tarjeta NETFPGA se hace a través del puerto PCI de ahí la importancia de cargar este controlador.

El CPCI se debe reprogramar cada vez que el host es reiniciado, esta reprogramación se hace a través del siguiente script:

```
/usr/local/sbin/cpci_reprogram.pl --all
```

La respuesta esperada es la siguiente:

```
Loading the CPCI Reprogrammer on NetFPGA 0
Loading the CPCI on NetFPGA 0
CPCI on NetFPGA 0 has been successfully reprogrammed
```

Es posible automatizar la carga del CPCI cuando el sistema se inicia, para esto se deberá agregar la siguiente línea en el archivo /etc/rc.local

```
/usr/local/netfpga/lib/scripts/cpci_reprogram/cpci_reprogram.pl --all
```

2.3.3. Diagnóstico de la plataforma

Una vez se hayan instalado todos los elementos que hacen parte del sistema, se puede elaborar una prueba del mismo por medio del auto-diagnostico que provee el fabricante.

El auto-diagnostico se compone de un componente hardware que se carga en la tarjeta NETFPGA y una aplicación software que gestiona el procedimiento. Todas las pruebas se hacen de manera paralela y se ejecutan indefinidamente hasta que el usuario lo decida.

Las operaciones de prueba son las siguientes:

- Prueba de escritura y lectura en cada posición de las memorias SRAM y DDR2.
- Envío y recepción de paquetes por las interfaces de red.
- Envío y recepción de información a través de la interfaz SATA usando las líneas Multi-Gigabit.
- Pruebas del CPCI incluyendo el modulo DMA.

Las anteriores operaciones se hacen a la máxima velocidad que soporte cada uno de los periféricos para asegurar el volumen de procesamiento del sistema.

Para llevar a cabo el auto-diagnostico se deben tener en cuenta las siguientes recomendaciones:

- Conexión de cable de retorno (loopback) para el puerto sata
- Conexión de cables de retorno (loopback) para las interfaces de red. Uno para 0-1 y otro para 2-3; los cables deben cumplir con el estándar GigaE.
- Levantar las interfaces de red por medio del comando:

```
for i in `seq 0 3`; do ifconfig nf2c$i up; done
```

- Cargar el archivo .bit en la tarjeta NETFPGA a través del comando:

```
nf_download ~/netfpga/bitfiles/selftest.bit
```

- Correr la aplicación mediante la consola con el comando:

```
~/netfpga/projects/selftest/sw/selftest
```

La respuesta que entregara el sistema si todos los procesos se ejecutan sin ningún inconveniente será:

```
Found net device: nf2c0
NetFPGA selftest 1.00 alpha
Running..... PASSED
/usr/local/netfpga/lib/scripts/cpci_reprogram/cpci_reprogram.pl --all
```

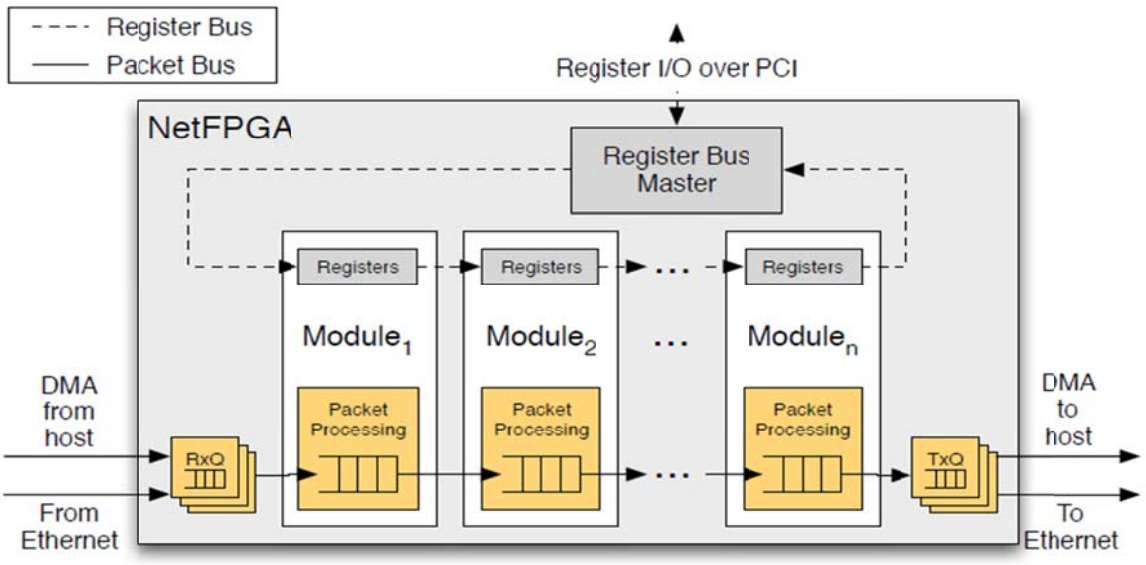
3. EXPERIMENTACIÓN DE LA PLATAFORMA NETFPGA 1G

3.1. ARQUITECTURA HARDWARE Y FLUJO DE INFORMACIÓN DE LA TARJETA NETFPGA 1G

3.1.1. Arquitectura segmentada: PIPELINE

Antes de empezar a trabajar con la plataforma NETFPGA, es muy importante que se tenga claro la arquitectura de funcionamiento del hardware. Esta arquitectura está basada en segmentos modulares interconectados por medio de dos buses independientes y con su propio ancho de banda: el bus de registros y el bus de paquetes.

Figura 10. Ilustración de la segmentación modular de la plataforma NETFPGA.



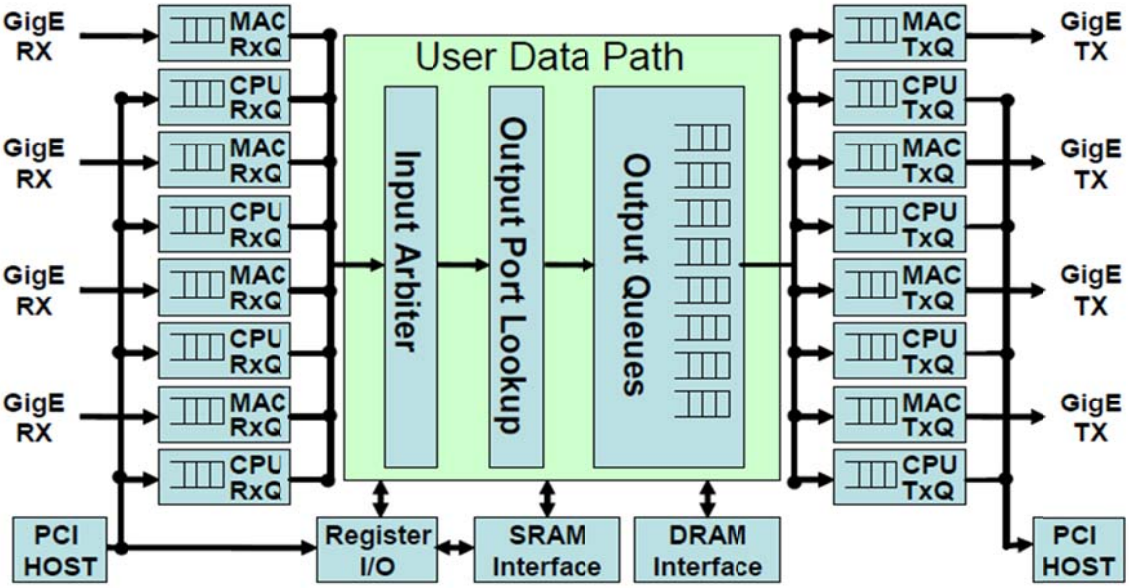
Fuente: Encouraging Reusable Network Hardware Design, Stanford University

El bus de paquetes, permite la transferencia de datos entre los módulos de procesamiento de paquetes usando una interfaz FIFO síncrona de 64 bits y un ancho de banda de 8Gbps.

El bus de registros, establece una interface entre los registros, contadores y tablas que hay en el hardware hacia las aplicaciones software, permitiendo que se consulten y/o se modifiquen tiene un ancho de banda mucho menor que el bus de paquetes.

Todas las aplicaciones hechas por el *NETFPGA Group* la estructura modular, y es fundamental que en el desarrollo de nuevas aplicaciones se esta misma estructura. A este tipo de segmentación modular se le conoce con el nombre "pipeline" cuya traducción al español es "tubería", hace analogía a que cada segmento es una tubería donde la salida alimenta otro segmento. En la siguiente figura se ilustra el diagrama de la segmentación (pipeline) de la NETFPGA:

Figura 11. Diagrama Pipeline de la NETFPGA



Fuente: http://netfpga.org/

Al elaborar un proyecto, los módulos elaborados por los usuarios se deben conectar al User Data Path. El Input Arbiter y el Output Queues son los módulos principales y están presentes en todos los diseños para la plataforma. El código fuente de estos, es proporcionado en el paquete de instalación de la tarjeta NETFPGA como código en Verilog.

El sistema de registros permite que nuevos módulos sean interconectados con el pipeline por medio de la lectura de información de estado y la configuran señales de control. De esta manera los programas (módulos software) que se ejecutan en el host pueden recibir y enviar información desde y hacia los módulos hardware.

3.1.2. Puertos de entrada y salida

Los paquetes entran y salen del “pipeline” a través de varias colas de transmisión y recepción. Estas conectan los puertos de entrada/salida al pipeline y sirven de interfaz entre los periféricos y las interfaces FIFO para que puedan acceder al “pipeline” también.

Existen tres tipos de puertos de entrada/salida en la tarjeta NETFPGA:

- Las colas de transmisión y recepción Ethernet, las cuales envían y reciben paquetes a través de los puertos GigaE.
- Las colas de transmisión y recepción CPU DMA, las cuales transfieren paquetes por medio del DMA entre la tarjeta NETFPGA y la CPU.
- Las colas de transmisión y recepción Multigigabit Seriales, que permiten la transferencia de paquetes a través de las interfaces SATA.

3.1.3. NFPs

NFP significa: NetFpga Packages, y se refiere a todos los proyectos terminados y con soporte para la plataforma. Estos proyectos están formados por bloques de hardware y software prediseñados con funciones específicas y documentadas incluido el código fuente de las mismas.

El contenido técnico de una NFP es el siguiente:

- Gateware (código en verilog y compilado .bit).
- Software del sistema (Código fuente y script para compilación).
- Software para pruebas de regresión.

3.2. MODOS DE USO DE LA PLATAFORMA NETFPGA 1G

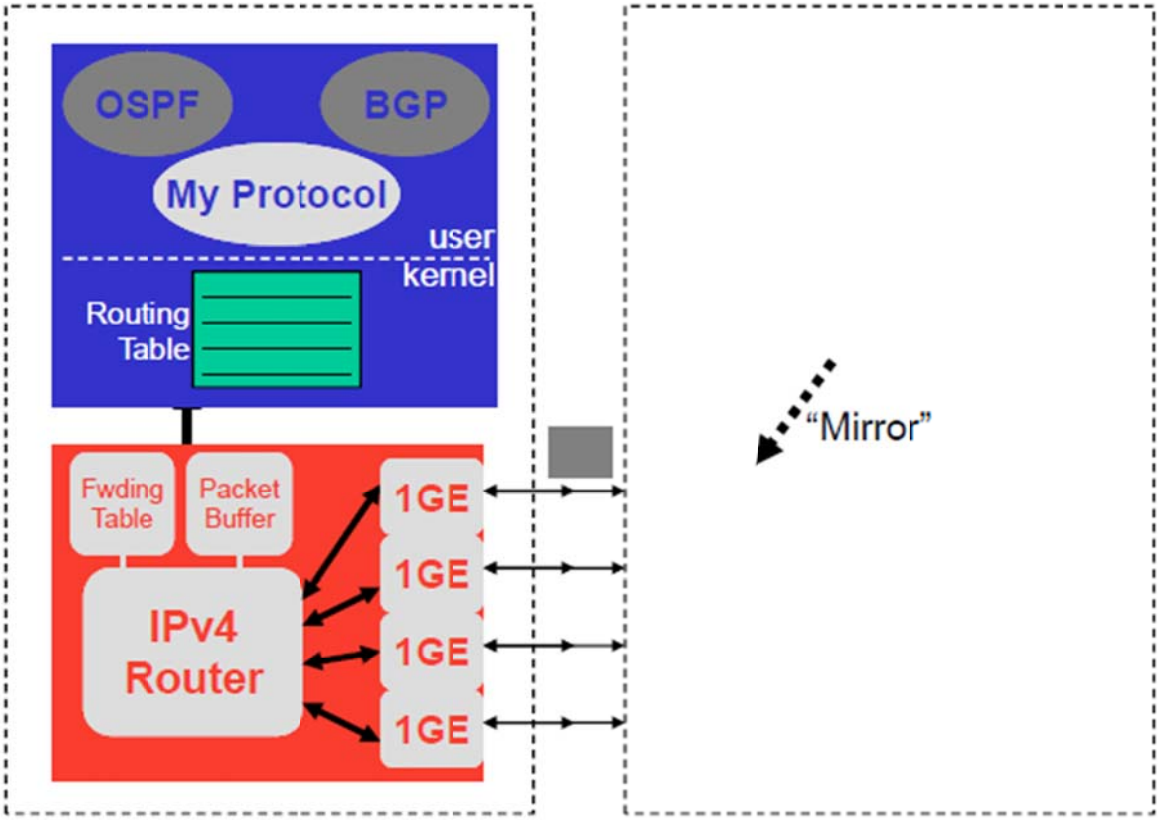
La plataforma NETFPGA 1G tiene tres modos de uso que van desde la experimentación de proyectos básicos hasta el desarrollo de complejas aplicaciones elaboradas en su totalidad por los usuarios. A continuación se describirá cada una de ellas.

3.2.1. Instalación y experimentación de diseños modulares

En este modo el usuario instala un ejemplo en la plataforma y únicamente se limita a hacer modificaciones a nivel de software.

En la figura 12, se muestra el diagrama de bloques de la aplicación *kit router*, la cual fue desarrollada por *NETFPGA Group*. Esta aplicación hace un espejo de la tabla de enrutamiento del sistema operativo y la transfiere a la tarjeta para que el procesamiento se haga de manera paralela en el hardware. El usuario podrá modificar el código fuente referente a los protocolos que se ejecutan en el host, inclusive crear uno propio y experimentar con él.

Figura 12. Uso de la plataforma NETFPGA como experimentación del *kit router*.

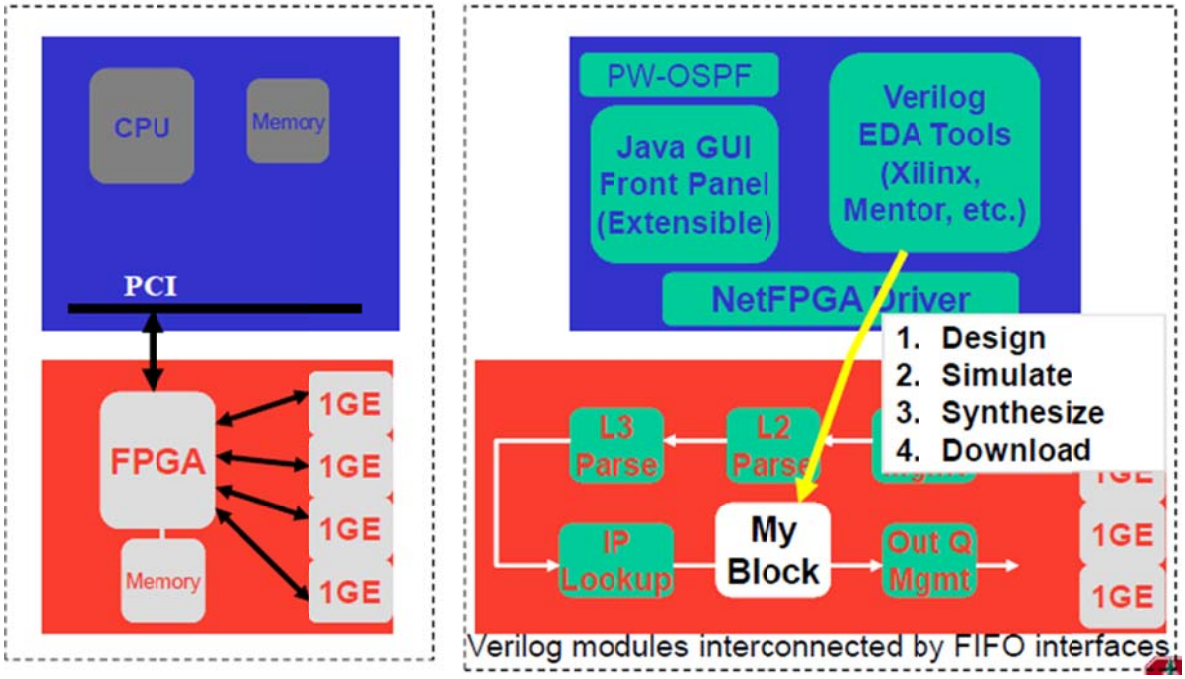


Fuente: Brno Tutorial, Universidad de Stanford

3.2.2. Instalación, modificación y experimentación de diseños modulares

Este modo, precisa cambios en el diseño del hardware y del software. Inicialmente se instala el desarrollo original y se modifica el hardware agregando módulos de otros desarrollos o elaborando módulos propios.

Figura 13. Uso de la plataforma NETFPGA como ampliación de un sistema modular.



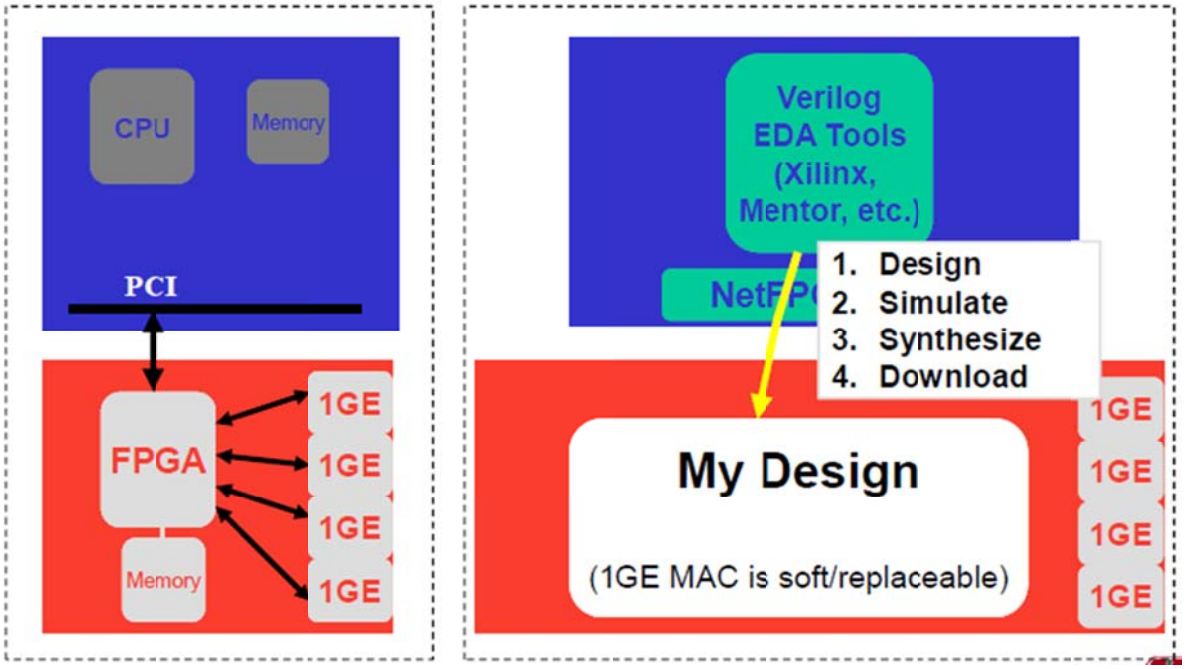
Fuente: Brno Tutorial, Universidad de Stanford

En la figura 13, se muestra la modificación del kit router y se puede detallar el proceso que se debe hacer al hacer una modificación a nivel de hardware (diseño, simulación, síntesis y programación).

3.2.3. Creación de nuevos sistemas

Este modo, requiere que el usuario diseñe todo el sistema (hardware y software) que se va a implementar. Es el nivel más alto que se puede trabajar con esta plataforma.

Figura 14. Uso de la plataforma NETFPGA para el diseño de un nuevo sistema modular



Fuente: Brno Tutorial, Universidad de Stanford

3.3. COMUNICACIÓN ENTRE HARDWARE Y SOFTWARE: DESCRIPCIÓN DE PROYECTOS.

3.3.1 NIC REFERENCE

NIC es el acrónimo de Network Interface Card y hace alusión a una interfaz de red. NIC Reference es un proyecto contenido en el paquete de soporte para la plataforma NETFPGA. Se trata de un código de ejemplo en elaborado en Perl que hace uso de las librerías del driver de la tarjeta NETFPGA para comunicarse con ella.

El código completo esta como anexo a este documento, para la ilustración de la descripción se tomará fragmentos del mismo explicando su función. También se detallaran los pasos para cargar y ejecutar la aplicación.

Como primera medida, se asume que la tarjeta NETFPGA está instalada correctamente; se ha almacenado la información de soporte en un directorio local.

El primer paso es asignar una dirección ip a las interfaces de red nf2cX, por ejemplo, el comando para configurar la dirección 192.168.200.1 en la interfaz 0 seria:

```
/sbin/ifconfig nf2c0 192.168.200.1
```

Ahora se debe cargar el archivo de configuración de hardware (reference_nic.bit) en la NETFPGA. El comando para esta operación es:

```
~/netfpga/lib/C/download/nf_download ~/netfpga/bitfiles/reference_nic.bit
```

La salida esperada es:

```
Found net device: nf2c0
Bit file built from: nf2_top_par.ncd
Part: 2vp50ff1152
Date: 2007/11/21
Time: 11: 0: 3
Error Registers: 1000000
Good, after resetting programming interface the FIFO is empty
Download completed -  2377668 bytes. (expected 2377668).
DONE went high - chip has been successfully programmed.
```

El siguiente paso consiste en compilar la aplicación, el comando es:

```
cd ~/netfpga/projects/reference_nic
make
```

Ahora es posible utilizar la herramienta software counterdump, la cual lee algunos contadores de los módulos de las interfaces de red e imprime su valor en la ventana de terminal. La aplicación se ejecuta mediante el comando:

```
./counterdump
```

La salida generada por el programa es:

```
Found net device: nf2c0
Num pkts received on port 0:           0
Num pkts dropped (rx queue 0 full):    0
Num pkts dropped (bad fcs q 0):        0
Num bytes received on port 0:          0
Num pkts sent from port 0:             0
Num bytes sent from port 0:            0

Num pkts received on port 1:           0
Num pkts dropped (rx queue 1 full):    0
Num pkts dropped (bad fcs q 1):        0
Num bytes received on port 1:          0
Num pkts sent from port 1:             0
Num bytes sent from port 1:            0

Num pkts received on port 2:           0
Num pkts dropped (rx queue 2 full):    0
Num pkts dropped (bad fcs q 2):        0
Num bytes received on port 2:          0
Num pkts sent from port 2:             0
Num bytes sent from port 2:            0

Num pkts received on port 3:           0
Num pkts dropped (rx queue 3 full):    0
Num pkts dropped (bad fcs q 3):        0
Num bytes received on port 3:          0
Num pkts sent from port 3:             0
Num bytes sent from port 3:            0
```

Como se puede observar, la totalidad de los contadores está en cero ya que se refieren al número de paquetes que gestiona cada una de las interfaces de red.

Ahora se ejecutará la aplicación send_pkts para hacer que los paquetes fluyan por las interfaces y los contadores empiecen a acumular. Al igual que la aplicación anterior, se precisa compilar antes de la ejecución. El comando para compilar esta aplicación es:

```
cd ~/netfpga/lib/C/tools/send_pkts
make
```

Una vez se reciba la confirmación de compilación, se debe invocar la aplicación para que genere 10 paquetes de 100 bytes cada uno en la interfaz 0. El comando que lo ejecuta es:

```
cd ~/netfpga/lib/C/tools/send_pkts   sudo ./send_pkts -i nf2c0 -s 10 -l 100
```

Si se ejecuta nuevamente la aplicación counterdump, se puede apreciar que los contadores ya han sido modificados según el número de paquetes y bytes que salieron por el puerto 0. La nueva salida de la aplicación counterdump es:

```
Found net device: nf2c0
Num pkts received on port 0:           0
Num pkts dropped (rx queue 0 full):    0
Num pkts dropped (bad fcs q 0):        0
Num bytes received on port 0:          0
Num pkts sent from port 0:             10
Num bytes sent from port 0:            1000

Num pkts received on port 1:           0
Num pkts dropped (rx queue 1 full):    0
Num pkts dropped (bad fcs q 1):        0
Num bytes received on port 1:          0
Num pkts sent from port 1:             0
Num bytes sent from port 1:            0

Num pkts received on port 2:           0
Num pkts dropped (rx queue 2 full):    0
Num pkts dropped (bad fcs q 2):        0
Num bytes received on port 2:          0
Num pkts sent from port 2:             0
Num bytes sent from port 2:            0

Num pkts received on port 3:           0
Num pkts dropped (rx queue 3 full):    0
Num pkts dropped (bad fcs q 3):        0
Num bytes received on port 3:          0
Num pkts sent from port 3:             0
Num bytes sent from port 3:            0
```

A continuación, la descripción del código fuente para la aplicación counterdump, para tener la referencia completa del código, remítase al anexo 1:

```
20   #include "../common/reg_defines.h"
21   #include "../common/nf2.h"
22   #include "../common/nf2util.h"
```

Primero, se incluyen las librerías, reg_defines.h contiene la definición de los registros y constantes de trabajo de la NETFPGA, nf2.h y nf2util.h incluyen macros y funciones para acceder a los registros.

```
29   static struct nf2device nf2;
```

Luego se declara una variable con la estructura de la NETFPGA, en ella estará contenida toda la información del dispositivo

```
26   #define DEFAULT_IFACE   "nf2c0"

38     nf2.device_name = DEFAULT_IFACE;
```

En la línea 26 se define nf2c0 como la interfaz por defecto, de esta forma si se quiere cambiar el puerto a monitorear, se deberá cambiar esta definición por nf2c1, nf2c2 o nf2c3. En la línea 38 se asigna la interface a la variable de control de la NETFPGA (nf2).

```
40     processArgs(argc, argv);
```

La función processArgs, permite que la interfaz por defecto sea cambiada desde la línea de comandos, de esta manera.

```
43     if (check_iface(&nf2))

47     if (openDescriptor(&nf2))
```

La función check_iface, comprueba que la interfaz exista y pueda ser levantada. La función openDescriptor abre la interface para que se pueda acceder por medio de llamados ioctl,, antes la interfz debe estar levantada y con una dirección IP asignada.

```
52     dumpCounts();

71     readReg(&nf2, TX_QUEUE_0_NUM_PKTS_SENT_REG, &val);
72     printf("Num pkts sent from port 0:             %u\n", val);
```

Las siguientes funciones, hacen la lectura de los registros (readReg) y la imprsion del resultado en la ventana de terminal. (printf). Las líneas 71 y 72 son un ejemplo de la lectura e impresión del valor del registro que contiene el número de paquetes enviados por la interfaz 0.

```
54     closeDescriptor(&nf2);
```

Por último, se cierra la interfaz mediante la función closeDescriptor.

3.3.2. PACKET GENERATOR

El proyecto packet generator fue desarrollado en la Universidad de Stanford y no hace parte del paquete básico de la plataforma NETFPGA, sin embargo fue elaborado como una NFP y brinda una guía más avanzada para el desarrollo de aplicaciones bajo la plataforma NETFPGA.

El proyecto consiste en un generador y monitor de paquetes diseñado para hacer pruebas sobre una red Ethernet. La implementación carga las tramas desde un archivo con formato PCAP y procede a transmitirlas; el retraso entre tramas asi como la velocidad de los datos pueden ser configuradas por el usuario.

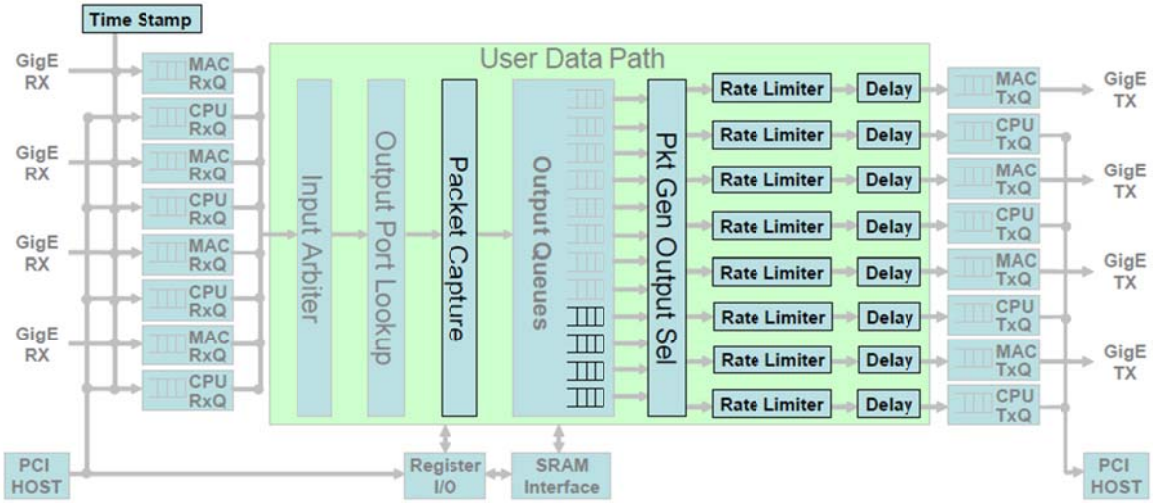
Un generador de paquetes, puede ser utilizado para reportes de estadísticas en redes, también para almacenar los paquetes que fluyen a través de un dispositivo de red específico para luego hacer un análisis de su rendimiento.

La implementación de un generador de paquetes sobre la plataforma NETFPGA ofrece una transferencia de paquetes a máxima velocidad de línea entre los cuatro puertos gigabit, mejorando el tiempo de respuesta de los generadores de paquetes implementados en software.

3.3.2.1. Arquitectura y gateware

La arquitectura del generador de paquetes está basada en el pipeline básico de la NETFPGA, como se puede apreciar en la siguiente figura:

Figura 15. Arquitectura del Generador de Paquetes



Fuente: A Packet Generator on the NetFPGA Platform, Stanford University

Se conservan los módulos principales (bloques opacos en la figura) y se agregaron cuatro módulos diferentes (bloques resaltados en la figura). A continuación una descripción de cada uno de los módulos hardware que conforman el proyecto:

Input Arbiter/Output Port Lookup

Son dos módulos básicos que hacen parte de la librería original de la NETFPGA, precisamente del desarrollo del diseño NIC Reference descrito anteriormente. El Input Arbiter selecciona una cola para atender y descarga los paquetes contenidos en ella sobre el módulo Outport Port Lookup direcciona el trafico recibido hacia su correspondiente CPU para habilitar la captura de paquetes.

Packet Capture

Es un nuevo módulo implementado en este proyecto. Tiene dos funciones principales:

- La compilación de etadisticas globales para la generación o captura de paquetes (Ejemplo: paquetes recibidos, tiempo total de captura).
- LA extracción de la fecha y hora de los paquetes cuando las funciones de generación y captura están deshabilitadas.

Output Queues

Este módulo, tiene una modificación respecto a la versión original de la librería. Se aumentó el número de colas de salida de 8 a 12, permitiendo que las cuatro nuevas colas sean usadas para almacenar la información a transmitir del PCAP cuando el generador de paquetes está habilitado.

El tamaño de las 12 colas es determinado por registros que el software configura cuando carga los archivos PCAP haciendo que estas tomen el tamaño preciso de dichos archivos, maximizando así el espacio disponible para las colas de recepción y optimizando el uso de la memoria SRAM.

Packet Generator Output Select

Las colas de salida están conectadas a este módulo, el cual determina cuáles de las doce colas deben estar conectadas a las ocho salidas en el pipeline. De esta manera habilita el diseño para manejar dos colas de transmisión por cada puerto MAC, una puede ser usada como transmisión mientras que el archivo PCAP puede estar cargándose en la otra.

Rate Limiter and Delay Module

Cada una de las líneas de salida tiene un módulo limitador de velocidad y un módulo de retardo, de esta forma se añaden las funcionalidades para cambiar la velocidad y el retardo de los paquetes que salen de cada uno de los puertos del generador de paquetes. Tanto la velocidad como el retardo se pueden configurar por medio de registros desde la aplicación software.

Registers

Permite la lectura y configuración de los registros que controlan los demás módulos desde la aplicación host.

No está de más indicar que toda la descripción del hardware está elaborada en Verilog. Para almacenarla en la NETFPGA, se deberá generar el archivo .bit correspondiente. En el directorio bitfiles se encuentra el archivo: packet_generator.bit el cual fue generado previamente al lanzamiento del proyecto y está listo para ser programado en la tarjeta NETFPGA por medio del comando:

```
nf_download packet_generator.bit
```

3.3.2.2. Módulo software

La parte software consiste en un script elaborado en Perl llamado packet_generator.pl el cual está en el directorio sw del proyecto.

El generador de paquetes puede correr en uno o en todos los puertos de la NETFPGA. Los paquetes del archivo PCAP se transfieren primero al hardware, en

caso de que estos superen el espacio de memoria disponible sólo se envía la primera parte del archivo.

Este script se ejecuta desde la consola de comandos y permite la configuración de las siguientes opciones:

-q <queue number> <pcap file> Especifica el archivo PCAP para cargar y ser enviado por una cola especifica.

-r <queue number> <rate> Especifica la velocidad en Kbps para una cola especifica.

-i <queue number> <number of iterations> Especifica el numero de iteracciones de la cola especifica.

-d <queue number> <delay between packets> Especifica el retardo en nanosegundos entre paquetes, si no se configura se asume el retardo por defecto del archivo PCAP. Si se configura con cero se deshabilita el retardo.

-c <queue number> <capture file> Especifica el archivo de captura en el modo monitor.

--pad Acorta todos los paquetes a 64 bytes máximo y permite que la NETFPGA rellene paquetes cuando los envía.

--nodrop No omite los paquetes en puertos de entrada que no se estén monitoreando.

--wait Espera por una señal antes de empezar la ejecución del modo de operación.

--ns Reporta tiempos con resolución de nanosegundos.

Al ejecutar el software, se imprime el número de paquetes que fueron cargados en las colas desde el archivo PCAP y se reporta el límite de velocidad configurado para cada cola. Cuando se habilita la captura, se imprimirá además el número de paquetes recibidos, el número de bytes, el tiempo de ejecución, y la tasa de transmisión.

3.3.2.3. Test de regresión

En todo proyecto ordenado deben existir pruebas e regresión para verificar la integridad del proyecto. El generador de paquetes tiene cuatro pruebas de regresión cada una especializada en un proceso. Estas son:

Test 1: Envío de Paquetes

Carga y envía dos archivos PCAP uno en la interfaz nf2c0 y otro en la nf2c1, luego verifica el contador de numero de paquetes enviados de cada interfaz.

Se ejecuta mediante el comando:

```
projects/packet_generator/regress/test_send_pkts $ Output
```

Test 2: Captura

Envía dos archivos PCAP, uno por la interfaz eth1 y otro por la interfaz eth2. Captura la salida usando el generador de paquetes en modo captura y verifica los paquetes obtenidos.

Se ejecuta mediante el comando:

```
projects/packet_generator/regress/test_capture $ Output
```

Test 3: Interacciones

Envía dos archivos PCAP, uno por la interfaz eth1 y otro por la interfaz eth2 con un número específico de interacciones, luego verifica el contador de paquetes enviados de cada interfaz.

Se ejecuta mediante el comando:

```
projects/packet_generator/regress/test_iterations $ Output
```

Test 4: Limitador de Velocidad

Envía dos archivos PCAP, uno por la interfaz eth1 y otro por la interfaz eth2 configurando una velocidad de 1Mbps, luego verifica la velocidad de transmision.

Se ejecuta mediante el comando:

```
projects/packet_generator/regress/test_rate_limiter $
```

3.3.2.4. Experimentación de la aplicación

La aplicación packet generator, opera de la misma manera que lo hace el programa tcpreply a nivel de sistema operativo. Los desarrolladores del proyecto hicieron dos experimentos similares, uno con el software tcpreply y otro con la aplicación packet generator en la NETFPGA y contrastaron los resultados. En esta sección se documenta el procedimiento de la experimentación y se analizan los resultados de los mismos.

Creación de archivos PCAP

Como primera medida, se genera un archivo PCAP con 43 paquetes y 25383 bytes. Este archivo servirá como fuente de información para los dos experimentos y puede ser generado con aplicaciones como Wireshark o tcpdump. La idea es

Preparación del equipo de cómputo

Los desarrolladores usaron un equipo de cómputo de alto rendimiento para el experimento con el tcpreply, las caraceristicas principales de dicho sistema son: Procesador AMD dual core @ 2.5GHz y tarjeta de red Intel dual port GigaE,

Ejecución del experimento

Se elaboró una prueba con el sistema de cómputo, la aplicación tcpreply y el archivo PCA antes generado, midiendo la velocidad promedio cuando se enviaban paquetes por un puerto y también cuando se enviaba la información por los dos puertos al tiempo. Luego se ensaya el sistema con la NETFPGA y el Packet Generator usando el mismo archivo PCAP que la prueba anterior. De igual manera, se hacen las mismas mediciones cuando se hace uso de uno y dos puertos. Cada una de las pruebas se ejecuta diez veces y se calculan los promedios de la misma como dato final.

Evaluación de resultados

Se comparan las velocidades de salida de los paquetes en ambos sistemas. El resultado es el siguiente:

Tabla 2. Resultados del experimento de generación de paquetes con tcpreply y NETFPGA

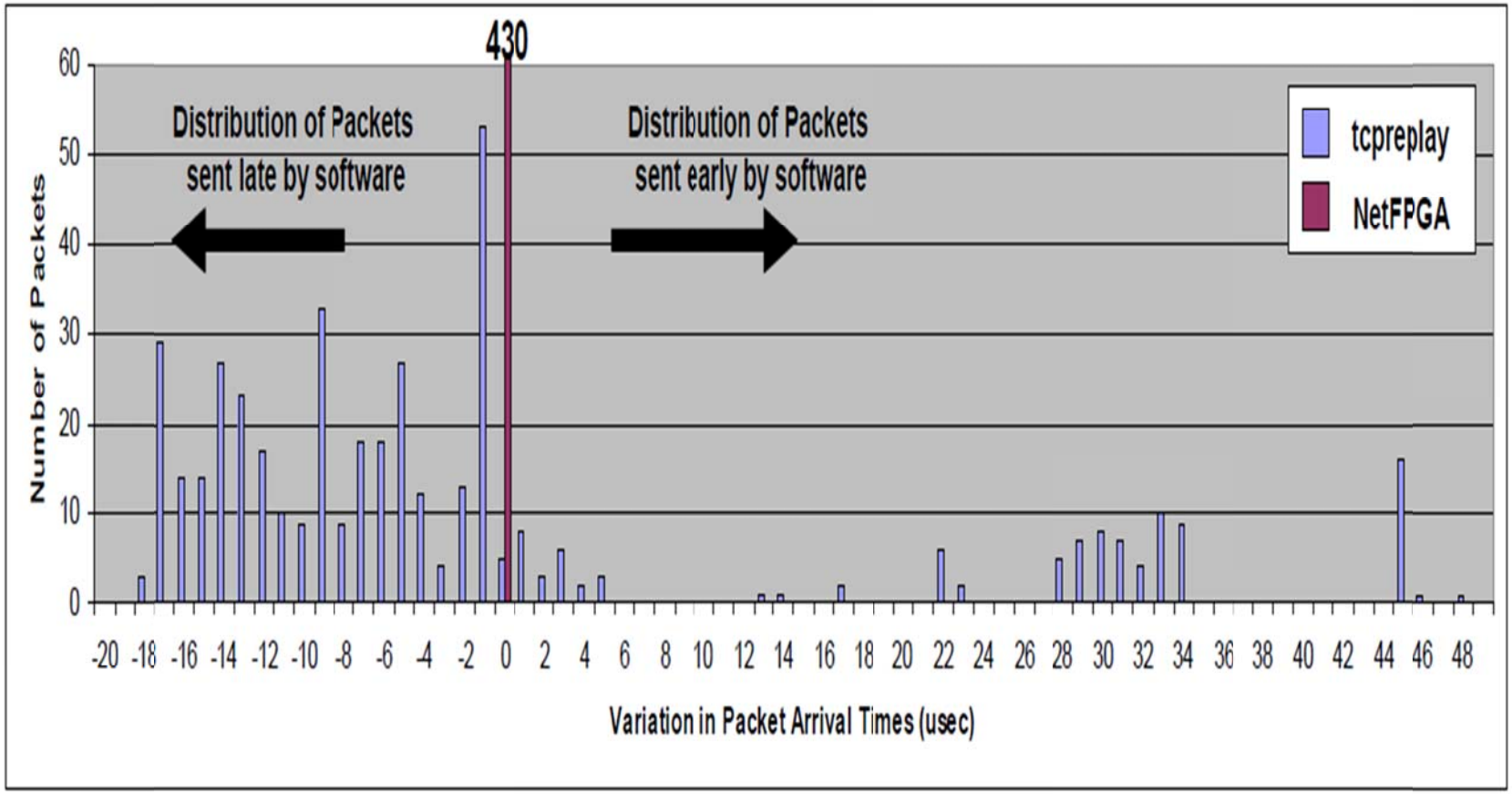
Numero de Puertos	Velocidad con tcpreply	Velocidad con NETFPGA
1	901.31 Mbps	1000 Mbps
2	896.23 Mbps	1000 Mbps

Fuente: A Packet Generator on the NetFPGA Platform, Stanford University

Esto demuestra las ventajas del procesamiento a velocidad de línea que se puede lograr con una NETFPGA, la cual mantuvo la máxima velocidad independientemente si se trabajó con uno o con los dos puertos. En el caso del tcpreply, es notable que el hecho de correr a nivel de sistema operativo incide en la velocidad de ejecución del programa

Al graficar la distribución del tiempo de llegada de los paquetes se expone otra de las ventajas de implementar el generador de paquetes sobre la NETFPGA y es la repetitividad de los retardos entre paquetes al ejecutar el experimento varias veces. Como se puede apreciar en la figura 16, los paquetes generados por la aplicación tcpreply (barras azules) llegan en su mayoría retardados entre 5 y 18 microsegundos con respecto al retardo esperado. Otros paquetes llegan unos cuatro microsegundos antes del estimado. Estas variaciones hacen que la herramienta no sea fiable. Con la NETFPGA (barra purpura) se puede comprobar que los paquetes llegan todos con el retardo configurado, sin desfase alguno.

Figura 16. Comparación de la distribución del tiempo de llegada de los paquetes entre tcpreply y NETFPGA.



Fuente: A Packet Generator on the NetFPGA Platform, Stanford University.

4. CONCLUSIONES

- La plataforma de desarrollo NETFPGA puede usarse como una herramienta de apoyo en investigaciones enfocadas al perfeccionamiento de dispositivos y protocolos en redes Ethernet, como por ejemplo la experimentación de un nuevo protocolo de enrutamiento o la implementación de algún algoritmo para la estimación de ancho de banda. Además puede ser útil también en el análisis de rendimiento en productos comerciales.

- Para lograr productos con gran valor agregado utilizando esta plataforma, se precisa de un grupo interdisciplinario que esté compuesto por profesionales del área de las telecomunicaciones y desarrolladores de hardware electrónico.

- En la elaboración de proyectos para NETFPGA, es muy importante seguir el estándar de desarrollo propuesto por el grupo de trabajo de la universidad de Stanford, el cual tiene como bases la segmentación del sistema en módulos y la implementación de pruebas de regresión para cada uno de estos. De esta manera es posible hacer uso de módulos incluidos en los proyectos de referencia del paquete de instalación e inclusive, aportar nuevos módulos al banco de proyectos de la plataforma.

- Debido a que NETFPGA es una plataforma de hardware libre que está apoyada en una plataforma de software libre, incentiva la formación de comunidades entusiastas en el tema en las que se comparten experiencias, proyectos e ideas.

BIBLIOGRAFÍA

- Glen Gibb, John W. y otros. NetFPGA – An Open Platform for Teaching How to Build Gigabit-rate Network Switches and Routers. 22p, 2008.

- MAIN AdamC. Guia de inicio. www.netfpga.org, 2011.

- STANFORD University. “Build an Internet Router”, Stanford University Computer Science. 344p, May 2007. [en línea] < http://yuba.stanford.edu/cs344/ > [Citado el 3 de marzo de 2011]

- STANFORD University. “A Packet Generator on the NetFPGA Platform”, Stanford University Computer Science. 4p, Abril de 2009. [en línea] < http://netfpga.org/documents/fccm-packet_generator-09-7.pdf-> [Citado el 31 de marzo de 2011]

- SUAREZ Gabriel. Microelectrónica. Universidad Francisco de Paula Santander. Memorias del programa de la asignatura. Bogotá 2004.

ANEXOS

ANEXO 1: CÓDIGO FUENTE COUNTERDUMP.C

GeSHi (c):

```c
/*
 ********************************************************************
 *********
  * vim:set shiftwidth=2 softtabstop=2 expandtab:
  * $Id: counterdump.c 5455 2009-05-05 18:18:16Z g9coving $
  *
  * Module:  counterdump.c
  * Project: NetFPGA NIC
  * Description: dumps the MAC Rx/Tx counters to stdout
  * Author: Jad Naous
  *
   * Change history:
   *
   */

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

#include <net/if.h>

#include "../lib/C/reg_defines_reference_nic.h"
#include "../../../lib/C/common/nf2.h"
#include "../../../lib/C/common/nf2util.h"

#define PATHLEN        80

#define DEFAULT_IFACE  "nf2c0"

/* Global vars */
static struct nf2device nf2;

/* Function declarations */
void dumpCounts();
void processArgs (int , char **);
void usage (void);

int main(int argc, char *argv[])
{
  nf2.device_name = DEFAULT_IFACE;

  processArgs(argc, argv);

```

```
42.    // Open the interface if possible
43.    if (check_iface(&nf2))
44.      {
45.        exit(1);
46.      }
47.    if (openDescriptor(&nf2))
48.      {
49.        exit(1);
50.      }
51.
52.    dumpCounts();
53.
54.    closeDescriptor(&nf2);
55.
56.    return 0;
57.  }
58.
59.  void dumpCounts()
60.  {
61.    unsigned val;
62.
63.    readReg(&nf2, MAC_GRP_0_RX_QUEUE_NUM_PKTS_STORED_REG, &val);
64.    printf("Num pkts received on port 0:           %u\n", val);
65.    readReg(&nf2, MAC_GRP_0_RX_QUEUE_NUM_PKTS_DROPPED_FULL_REG,
   &val);
66.    printf("Num pkts dropped (rx queue 0 full):    %u\n", val);
67.    readReg(&nf2, MAC_GRP_0_RX_QUEUE_NUM_PKTS_DROPPED_BAD_REG,
   &val);
68.    printf("Num pkts dropped (bad fcs q 0):        %u\n", val);
69.    readReg(&nf2, MAC_GRP_0_RX_QUEUE_NUM_BYTES_PUSHED_REG, &val);
70.    printf("Num bytes received on port 0:          %u\n", val);
71.    readReg(&nf2, MAC_GRP_0_TX_QUEUE_NUM_PKTS_SENT_REG, &val);
72.    printf("Num pkts sent from port 0:             %u\n", val);
73.    readReg(&nf2, MAC_GRP_0_TX_QUEUE_NUM_BYTES_PUSHED_REG, &val);
74.    printf("Num bytes sent from port 0:            %u\n\n", val);
75.
76.    readReg(&nf2, MAC_GRP_1_RX_QUEUE_NUM_PKTS_STORED_REG, &val);
77.    printf("Num pkts received on port 1:           %u\n", val);
78.    readReg(&nf2, MAC_GRP_1_RX_QUEUE_NUM_PKTS_DROPPED_FULL_REG,
   &val);
79.    printf("Num pkts dropped (rx queue 1 full):    %u\n", val);
80.    readReg(&nf2, MAC_GRP_1_RX_QUEUE_NUM_PKTS_DROPPED_BAD_REG,
   &val);
81.    printf("Num pkts dropped (bad fcs q 1):        %u\n", val);
82.    readReg(&nf2, MAC_GRP_1_RX_QUEUE_NUM_BYTES_PUSHED_REG, &val);
83.    printf("Num bytes received on port 1:          %u\n", val);
84.    readReg(&nf2, MAC_GRP_1_TX_QUEUE_NUM_PKTS_SENT_REG, &val);
85.    printf("Num pkts sent from port 1:             %u\n", val);
86.    readReg(&nf2, MAC_GRP_1_TX_QUEUE_NUM_BYTES_PUSHED_REG, &val);
87.    printf("Num bytes sent from port 1:            %u\n\n", val);
88.
89.    readReg(&nf2, MAC_GRP_2_RX_QUEUE_NUM_PKTS_STORED_REG, &val);
90.    printf("Num pkts received on port 2:           %u\n", val);
91.    readReg(&nf2, MAC_GRP_2_RX_QUEUE_NUM_PKTS_DROPPED_FULL_REG,
```

```
   &val);
92.    printf("Num pkts dropped (rx queue 2 full):    %u\n", val);
93.    readReg(&nf2, MAC_GRP_2_RX_QUEUE_NUM_PKTS_DROPPED_BAD_REG,
   &val);
94.    printf("Num pkts dropped (bad fcs q 2):        %u\n", val);
95.    readReg(&nf2, MAC_GRP_2_RX_QUEUE_NUM_BYTES_PUSHED_REG, &val);
96.    printf("Num bytes received on port 2:          %u\n", val);
97.    readReg(&nf2, MAC_GRP_2_TX_QUEUE_NUM_PKTS_SENT_REG, &val);
98.    printf("Num pkts sent from port 2:             %u\n", val);
99.    readReg(&nf2, MAC_GRP_2_TX_QUEUE_NUM_BYTES_PUSHED_REG, &val);
100.   printf("Num bytes sent from port 2:            %u\n\n", val);
101.
102.   readReg(&nf2, MAC_GRP_3_RX_QUEUE_NUM_PKTS_STORED_REG, &val);
103.   printf("Num pkts received on port 3:           %u\n", val);
104.   readReg(&nf2, MAC_GRP_3_RX_QUEUE_NUM_PKTS_DROPPED_FULL_REG,
   &val);
105.   printf("Num pkts dropped (rx queue 3 full):    %u\n", val);
106.   readReg(&nf2, MAC_GRP_3_RX_QUEUE_NUM_PKTS_DROPPED_BAD_REG,
   &val);
107.   printf("Num pkts dropped (bad fcs q 3):        %u\n", val);
108.   readReg(&nf2, MAC_GRP_3_RX_QUEUE_NUM_BYTES_PUSHED_REG, &val);
109.   printf("Num bytes received on port 3:          %u\n", val);
110.   readReg(&nf2, MAC_GRP_3_TX_QUEUE_NUM_PKTS_SENT_REG, &val);
111.   printf("Num pkts sent from port 3:             %u\n", val);
112.   readReg(&nf2, MAC_GRP_3_TX_QUEUE_NUM_BYTES_PUSHED_REG, &val);
113.   printf("Num bytes sent from port 3:            %u\n\n", val);
114. }
115.
116. /*
117.  *  Process the arguments.
118.  */
119. void processArgs (int argc, char **argv )
120. {
121.   char c;
122.
123.   /* don't want getopt to moan - I can do that just fine thanks!
   */
124.   opterr = 0;
125.
126.   while ((c = getopt (argc, argv, "i:h")) != -1)
127.     {
128.       switch (c)
129.         {
130.         case 'i':      /* interface name */
131.           nf2.device_name = optarg;
132.           break;
133.         case '?':
134.           if (isprint (optopt))
135.             fprintf (stderr, "Unknown option `-%c'.\n", optopt);
136.           else
137.             fprintf (stderr,
138.                      "Unknown option character `\\x%x'.\n",
139.                      optopt);
140.         case 'h':
```

```
141.         default:
142.           usage();
143.           exit(1);
144.         }
145.     }
146. }
147.
148.
149. /*
150.  *  Describe usage of this program.
151.  */
152. void usage (void)
153. {
154.   printf("Usage: ./counterdump <options> \n\n");
155.   printf("Options: -i <iface> : interface name (default
   nf2c0)\n");
156.   printf("         -h : Print this message and exit.\n");
157. }
158.
```