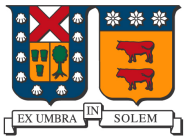

UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

Departamento de Electrónica

ELO-322 Redes de Computadores I

Proyecto Final:
Comunicación UDP-IP sobre Ethernet basado en FPGA

Integrantes:	*ROL:*
Julio Labra	201521016-0
Patricio Henríquez	201521036-5
Antonia Murillo	201403011-8

Viernes 5 de Julio de 2019

Resumen

En el modelo de capas TCP-IP existen diversos protocolos de capa de aplicación que permiten la comunicación lógica entre hosts. En particular, IPbus es un protocolo de código abierto generado por la Organización Europea para la Investigación Nuclear (CERN) que requiere, para su ejecución exitosa, al resto de las capas inferiores del modelo TCP-IP. Con el objetivo de implementar en el futuro IPbus sobre una FPGA, en este documento se expone el diseño modular de hardware, basado en el esquema TCP-IP, para la transmisión y recepción de paquetes UDP/IP entre dos tarjetas FPGA Artix 7 comunicadas a través de un enlace Ethernet.

Introducción

Este documento, se realizó en base a la idea de implementar un protocolo para comunicar dispositivos de hardware. La versatilidad de las FPGAs permite describir un protocolo con distintas particularidades que se podrían requerir, logrando además tener un control fino sobre como y que momento sucederán dichas acciones. Muchos requerimientos importantes deben ser considerados cuando se diseña la arquitectura e implementación de un sistema de control de hardware, por lo que la FPGA ofrece una gran gama de posibilidades en este aspecto.

El protocolo de capa de aplicación mencionado corresponde a uno diseñado por la Organización Europea para la Investigación Nuclear (CERN), el cual ofrece como código abierto, llamado IPbus. La idea principal de este protocolo es agregar confiabilidad a la conexión UDP, reemplazando en parte a los servicios que proporcionaría TCP, pero no con todas las exigencias que este protocolo especifica, siendo más simple y entregando un tiempo de respuesta mucho menor. Como protocolo de capa de aplicación, IPbus estará implementado sobre un conjunto de protocolos de capas inferiores del modelo TCP/IP. Por esta razón, es necesario generar en FPGA la modificación de los encabezados de un paquete de capa de aplicación transmitido/recibido los protocolos de capas inferiores, los cuales serían, en este caso, el protocolo UDP en la capa de transporte, IP en la capa de red y Ethernet sobre la capa de enlace de datos. A continuación se expone de forma general la implementación de cada uno de estos.

Implementación

En esta sección se describe el hardware y software utilizado para implementar el sistema de comunicación UDP-IP basada en el protocolo Ethernet.

Field-Programmable Gate Array (FPGA)

Una matriz de puertas programables o FPGA (del inglés *field-programmable gate array*) es un dispositivo programable que contiene bloques de lógica cuya interconexión y funcionalidad puede ser configurada en el momento, mediante un lenguaje de descripción especializado. Para realizar este proyecto se utilizan dos

tarjetas FPGA Artix 7, una para llevar a cabo la encapsulación y transmisión de información, y la otra para realizar su correspondiente recepción y desempaquetado, estando ambas conectadas a través de un cable Ethernet.

Figura 1: Tarjeta NexysVideo con FPGA Artix 7

Módulos de Transmisión y Recepción

La figura 9, incluida como anexo, muestra una vista general de los módulos y flujo de datos implementados en las FPGA de transmisión y recepción. En azul se encierran los módulos de hardware descritos "manualmente" dentro de cada FPGA. Separados por una línea fragmentada se muestra el flujo de datos y el hardware correspondiente a la FPGA de transmisión (izquierda) y recepción (derecha) y fuera de este se muestra el bloque de capa física que transmite/recibe los bits de cada tarjeta, cuya implementación y funcionamiento vienen incorporadas en cada FPGA, sin necesidad de configurarlas de antemano. Ambas tarjetas se encuentran conectadas entre sí mediante un cable Ethernet que permitirá su comunicación.

Obsérvese en la imagen como en ambas FPGAs se han implementado, en su mayoría, dos tipos de bloque: memorias FIFO (*First In First Out*) y módulos que desempeñan, de arriba hacia abajo, las funciones de empaquetamiento (en la FPGA de transmisión) - y desempaquetamiento (en la FPGA de recepción) asociadas a las capas de transporte, de red y de enlace. Cada módulo agrega o quita los encabezados correspondientes para enviar el paquete de datos a la capa posterior (representado por flechas). El procesamiento de los datos de cada paquete en estos módulos se realiza por byte. Debido a esto, las memorias FIFO desempeñan la función de buffer de almacenamiento de los bytes de un mismo paquete provenientes de la capa anterior mientras este es procesado por la capa siguiente. Una vez terminado este procesamiento, el byte liberado es enviado a la memoria FIFO que sigue, y así sucesivamente. Por otra parte, el bloque MAC es un cuyo rol es comunicar agregar los encabezados de Preámbulo y Verificación de Redundancia física de la capa de enlace, y los últimos dos bloques, de abajo hacia arriba, corresponden al chip embebido en la tarjeta que incorpora las funciones de capa física, y un conversor de bytes apropiado

para que esta reciba la información del resto de la pila de protocolos en el formato que lo espera.

A continuación se describe en detalle el flujo de datos entre los módulos para la transmisión y la recepción de ambas FPGAs.

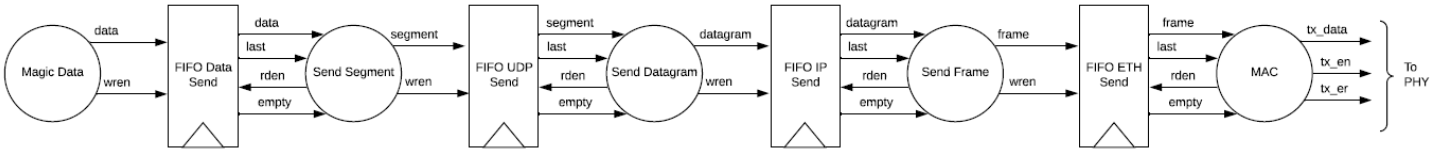

Figura 2: Camino de transmisión

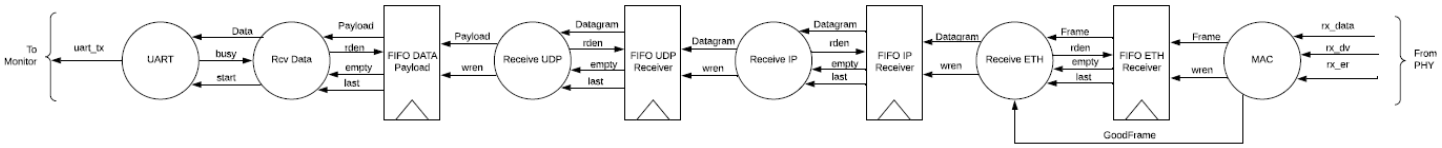

Figura 3: Camino de recepción

1. Camino de transmisión : Como podemos observar en la figura (2) el flujo de señales comienza en el módulo "Magic Data", en donde se genera la carga útil a transmitir, los cuales son guardados en la FIFO de envío de datos. La capa de transporte, al detectar que existe información útil cargada en la FIFO de datos (es decir, la FIFO no está vacía) comenzará a realizar la lectura de esta y a generar los encabezados correspondientes a la capa, enviando el segmento a la siguiente FIFO, en donde la capa inferior comenzará el mismo procesamiento. Este flujo de datos sigue la misma lógica para las demás capas inferiores y el módulo MAC, en donde se notará la modularidad e independencia que tiene cada una de las capas al integrar las memorias FIFO.

2. Camino de recepción : El camino correspondiente a la recepción comienza su flujo desde la capa física, donde los datos llegan de manera serial un byte a la vez y se dirigen a la MAC. El módulo MAC analiza el frame entrante y nos dará una señal si este se encuentra correcto o no. La señal "Goodframe" nos indica lo mencionado anteriormente y con ella comandaremos el paso de los datos hacia la capa Ethernet, a su vez la señal de "Badframe" hará de reset para la capa Ethernet de manera de desechar el paquete y no sea entregado a la capa siguiente. El movimiento de datos entre las capas siguientes consiste en esperar que los datos sean cargados en la FIFO de la capa actual y luego de comprobar los encabezados , se procede a entregar a la capa siguiente el payload de la capa actual. Cabe notar que nuestra pseudo capa aplicación consiste en recibir el mensaje desempaquetado y automáticamente enviarlo por el puerto serial hacia el computador.

Los caminos descritos anteriormente se encuentran cada uno en una FPGA diferente, esto se realizó de dicha manera para notar el flujo del mensaje de mejor forma. El funcionamiento interno de cada modulo consiste en flujos de maquinas de estado que leen y escriben memorias FIFO de la capa actual y siguiente

según corresponda, y mediante este proceso van desempaquetando y comprobando la consistencia de los datos. Cabe notar que si en alguna capa el paquete no corresponde a lo esperado este simplemente es desechado.

A continuación se describe de forma sucinta los encabezados agregados por cada una de los módulos anteriores, asociados a las capas del modelo TCP-IP.

(i) **Capa de aplicación**

La capa de aplicación no se encuentra implementada en este trabajo, no obstante, el diseño del hardware considera soportar, en un futuro, manejo de datos a nivel de capa de aplicación. En su reemplazo, se establece por defecto generar en la FPGA un paquete de capa aplicación cuyos datos son una secuencia conocida. Este paquete de datos es luego entregado a la memoria FIFO que interconecta las capas de transporte y de aplicación.

En cuanto a la recepción igualmente no existe una capa aplicación como tal, pero si se utilizó la interfaz UART para comprobar lo datos que llegan desde la capa UDP, a través del puerto serial del computador.

(ii) **Capa de transporte**

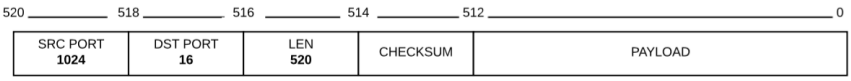

Figura 4: Formato del paquete UDP

En la figura (10) se muestra el formato del datagrama UDP, modificado en el módulo asociado a la capa de transporte. En esta etapa se añaden los encabezados correspondientes al protocolo UDP, siendo estos puerto de origen, puerto de destino, longitud del paquete, suma de comprobación y datos de la capa de aplicación. En negrita se muestran los valores establecidos constantes para todos los paquetes que son procesados. Naturalmente para este caso, la suma de comprobación será constante (pues los encabezados se mantienen constantes), y la carga útil corresponde al paquete de capa de aplicación con los datos de prueba.

(iii) **Capa de red**

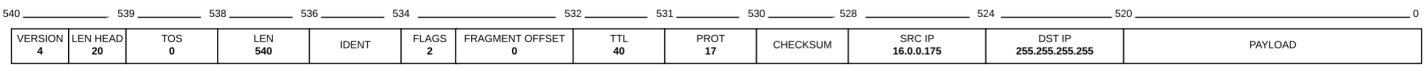

Figura 5: Formato del datagrama IP

La figura 5 muestra los encabezados añadidos por el módulo de capa de enlace. En negrita están escritos los valores que permanecen constante para todos los paquetes, y la carga útil corresponde al paquete UDP.

(iv) **Capa de enlace**

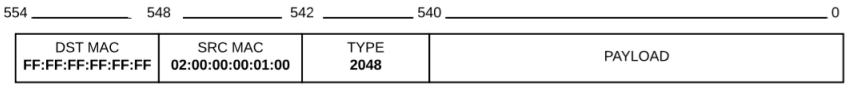

Figura 6: Formato de la trama Ethernet

En el frame del protocolo Ethernet incluye la dirección MAC de destino, la dirección MAC de fuente y el EtherType, que especifica el protocolo de capa superior, de forma similar como lo hace el campo Protocolo en IPv4. En el Payload se encuentra el datagrama de capa superior.

(v) **Capa física**

La capa física viene incorporada en un chip realtek rtl8211e. Este ejecuta las funciones de transmisión y recepción física de forma independiente, siendo necesario únicamente suministrar los valores lógicos de sus pines de entrada, y la señal digital proveniente de la interfaz Ethernet conectada.

Demostración experimental

La demostración de la transmisión y la recepción consiste en dos experimentos. Para ver que la transmisión de datos se realice de forma efectiva, se conecta por cable Ethernet la tarjeta encargada de la transmisión a un computador, en donde *Wireshark* estará realizando la captura de paquetes. Con esto será posible ver cada una de la capas empleadas en la transmisión de datos y corroborar que los campos de encabezado para cada uno de los protocolos y la carga útil corresponda con lo definido en la descripción del hardware implementado en la FPGA. En el Anexo I se encuentra la captura realizada en *Wireshark*.

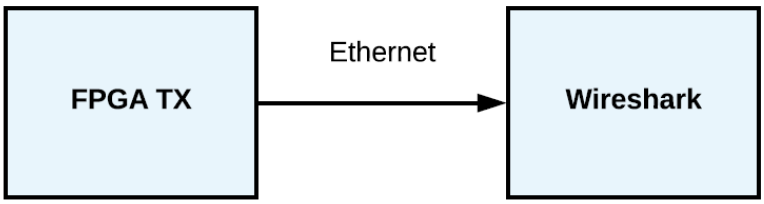

Figura 7: Experimento transmisión de paquetes

El experimento relacionado a la recepción de datos consiste en la transmisión por Ethernet desde una tarjeta a otra, en donde esta última estará disponible para realizar la recepción en cualquier momento. Una vez se haya recibido el paquete se procesarán sus datos por cada una de las capas implementadas en la tarjeta de recepción. Finalmente, cuando la capa de transporte reciba el segmento, la carga útil, asociada a la capa de aplicación, será enviada por interfaz UART a través de una conexión USB a un computador, la cual será posible observar a través de un monitor de datos ejecutándose en este.

Figura 8: Experimento captura de paquetes

Conclusiones

Este trabajo introduce un esquema de protocolo de comunicación UDP-IP, donde se vieron involucradas 4 de las 5 capas asociadas al modelo de capas TCP/IP. Aunque si bien la capa de aplicación no se vio reflejada en nuestro trabajo se emuló la existencia de esta mediante la generación de un "paquetez la recepción de este en el otro lado.

Para llevar a cabo esto se necesitó condensar lo aprendido durante las cátedras, en relación al empaquetamiento y consistencia de los paquetes referentes a una de las capas del modelo TCP-IP. Entender como se relaciona cada capa entre sí y poder separar su funcionamiento a través de una correcta descripción también forman parte de los aprendizajes relacionados a la implementación.

Como se mencionó, este proyecto nace de la implementación para comunicación de dispositivos de hardware mediante un protocolo de capa superior IPbus, donde esto decanta en una primera etapa descrita en este informe. La consistencia de los paquetes se logró observar en primera instancia a través del software Wireshark, notando la existencia de cada una de las capas y sus encabezados correspondientes. Luego en la recepción se logró un entendimiento más acabado de la forma de reconocer cada uno de los campos adyacentes al mensaje de interés y de como estos podrían interactuar con una posible aplicación.

Referencias

[1] James F. Kurose y Keith W. Ross *Redes de Computadoras - Un enfoque descendente* Quinta edición. Pearson Education, 2010.

Anexos

I. Esquema general de los módulos implementados en la FPGA.

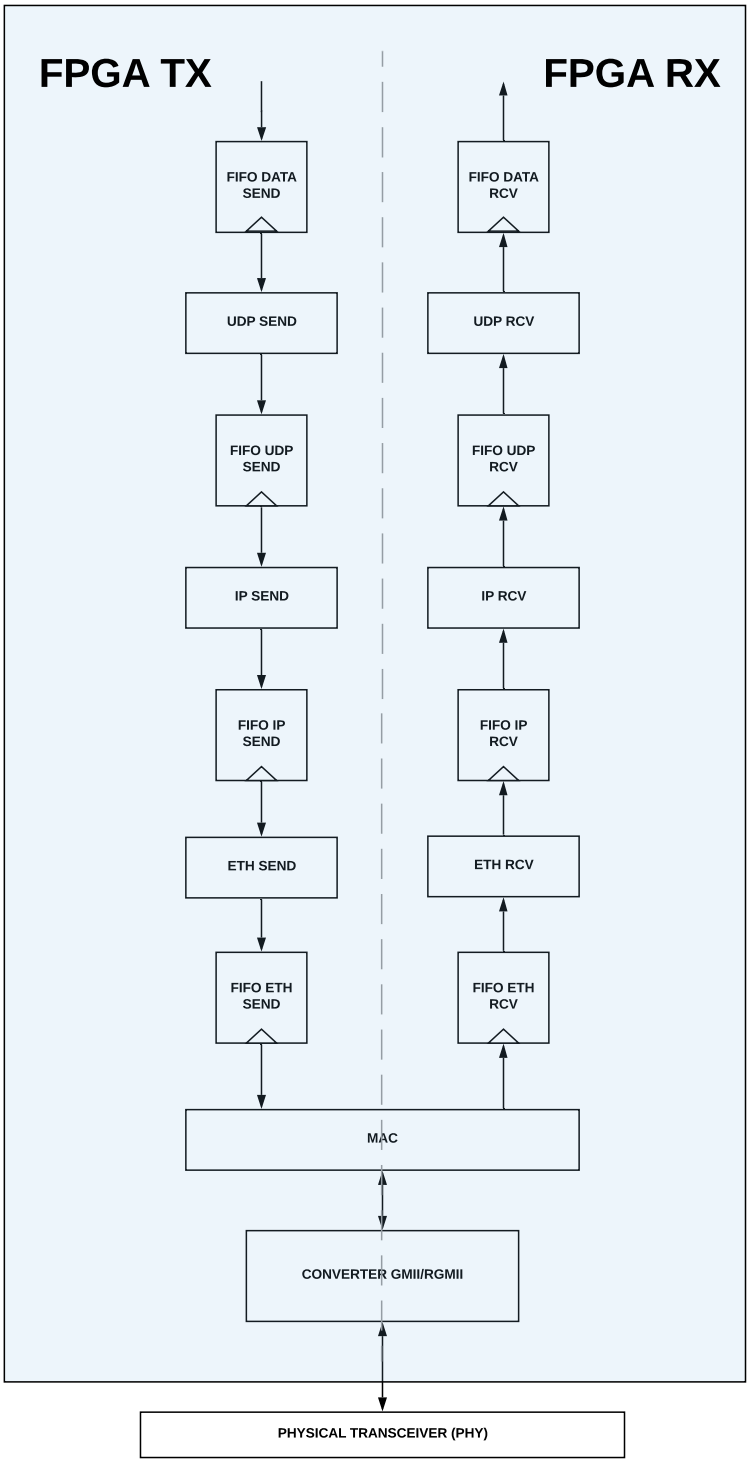

Figura 9: Diagrama de los módulos de transmisión y recepción

II. Captura realizada por *Wireshark*.

```
No.     Time           Source                Destination           Protocol Length Info
     21 9.096264       16.0.0.175            255.255.255.255       UDP      554    1024 → 16 Len=512
Frame 21: 554 bytes on wire (4432 bits), 554 bytes captured (4432 bits) on interface 0
Ethernet II, Src: 02:00:00:00:01:00 (02:00:00:00:01:00), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
    Destination: Broadcast (ff:ff:ff:ff:ff:ff)
    Source: 02:00:00:00:01:00 (02:00:00:00:01:00)
    Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 16.0.0.175, Dst: 255.255.255.255
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
    Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 540
    Identification: 0x0004 (4)
    Flags: 0x4000, Don't fragment
    Time to live: 64
    Protocol: UDP (17)
    Header checksum: 0x281f [correct]
    [Header checksum status: Good]
    [Calculated Checksum: 0x281f]
    Source: 16.0.0.175
    Destination: 255.255.255.255
User Datagram Protocol, Src Port: 1024, Dst Port: 16
    Source Port: 1024
    Destination Port: 16
    Length: 520
    Checksum: 0xf9e7 [unverified]
    [Checksum Status: Unverified]
    [Stream index: 0]
    [Timestamps]
Data (512 bytes)
0000  9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d   ................
0010  9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d   ................
0020  9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d   ................
0030  9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d   ................
0040  9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d   ................
0050  9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d   ................
0060  9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d   ................
0070  9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d   ................
0080  9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d   ................
0090  9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d   ................
00a0  9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d   ................
00b0  9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d   ................
00c0  9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d   ................
00d0  9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d   ................
00e0  9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d   ................
00f0  9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d   ................
0100  9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d   ................
0110  9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d   ................
0120  9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d   ................
0130  9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d   ................
0140  9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d   ................
0150  9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d   ................
0160  9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d   ................
0170  9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d   ................
0180  9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d   ................
0190  9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d   ................
01a0  9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d   ................
01b0  9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d   ................
01c0  9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d   ................
01d0  9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d   ................
01e0  9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d   ................
01f0  9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d 9d   ................
    Data: 9d9d9d9d9d9d9d9d9d9d9d9d9d9d9d9d9d9d9d9d9d9d9d9d9d9d…
    [Length: 512]
```

Figura 10: Experimento captura de paquetes