Universidad de Costa Rica
Facultad de Ingeniería
Escuela de Ingeniería Eléctrica
Departamento de Automática

`IE-0523` Circuitos Digitales II

Reporte del Proyecto I: Diseño de la capa *PHY* de la interfaz *PCIe*

B61325 - Alexander Calderón Torres - `alexander.calderontorres@ucr.ac.cr`
A45967 - Freddy Zúñiga Cerdas - `frenzuce@gmail.com`

Sábado 12 de junio
I ciclo 2021

Índice

1. Resumen

En el presente proyecto, se presenta el resultado final de la implementación de la capa física de la interfaz *PCIe* en *Verilog*. Para realizar esta implementación, se empezó por analizar la micro-arquitectura de esta capa física y a estudiar los distintos comportamientos que se esperaban como resultado, principalmente de cada uno de los módulos individuales a utilizar, pero también de los módulos principales que requerían de una mayor integración a nivel general.

Se diseñó la capa física de manera tal que se recibe una señal inicial de **cuatro líneas de 8 bits con un bit de valid y a una frecuencia inicial f**. Esta señal atraviesa una etapa de transmisión que gradualmente se encargará de serializar la información, mantener la coherencia de los datos y de las señales valid, y de aumentar la frecuencia según se necesite como consecuencia de dicha serialización. Después de esto, una etapa receptora se encarga de esperar cuatro señales `0xBC` para activarse, y de esta forma iniciar con la lectura y paralelización de datos, así como una disminución paulatina de la frecuencia de entrada, hasta que del receptor sale la señal que inicialmente se había enviado a través de todo el sistema.

Se realizaron pruebas conocidas y pruebas aleatorias al diseño que se creó, y finalmente los resultados que se obtuvieron fueron congruentes y funcionales.

2. Descripción arquitectónica del diseño

El sistema se creó de manera tal que su funcionamiento a nivel general cumpliera con lo detallado en la descripción de la figura 1, dada como parte del enunciado del proyecto. La creación de los distintos módulos del proyecto se realizó con base en lo descrito en este diagrama, y se respetó en su mayor parte la forma en que estos interactúan entre sí, con el fin de mantener el orden y garantizar el comportamiento deseado.

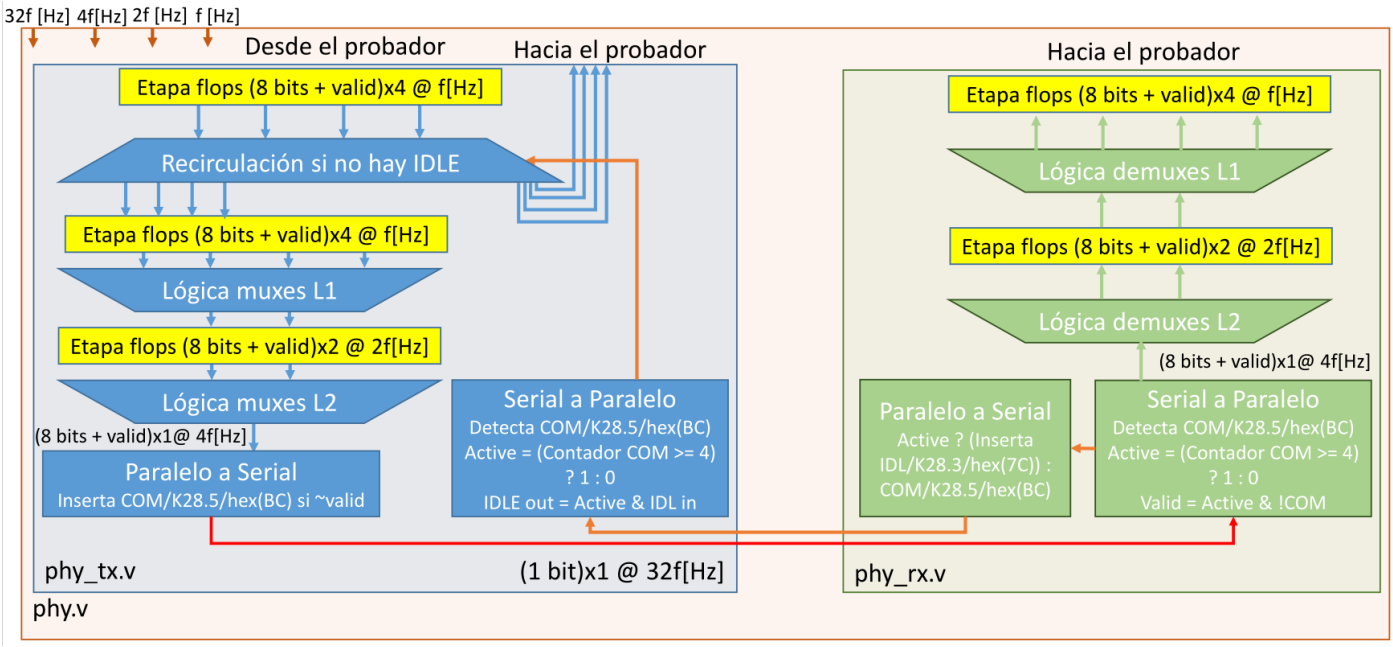

Figura 1: Diagrama general de la capa física de la interfaz *PCIe*

A continuación, se describe muy brevemente el funcionamiento esperado de las principales partes de la capa física diseñada y los principales módulos utilizados:

- `recirculacion.v`: Si no hay IDL, la señal sale hacia el probador; si hay IDL, los datos circulan hacia por el transmisor y hacia el receptor normalmente.
- `mux4x2_8bits.v`: Convierte la entrada de cuatro líneas de *8 bits + valid* a 2 líneas de *8 bits + valid*.
- `mux2x1_8bits.v`: Convierte la entrada de dos líneas de *8 bits + valid* a una líneas de *8 bits + valid*.
- `demux1x2_8bits.v`: Convierte la entrada de una línea de *8 bits + valid* a dos líneas de *8 bits + valid*.
- `demux2x4_8bits.v`: Convierte la entrada de dos líneas de *8 bits + valid* a cuatro líneas de *8 bits + valid*.
- `paralelo_serial1.v`: Convierte la entrada que viene en múltiples bits en paralelo, a una salida serial de un bit con una frecuencia muy alta.
- `paralelo_serial2.v`: Se encarga de enviar las señales *COM* e *IDL* al módulo `serial_paralelo2`.

- `serial_paralelo1.v`: Se encarga de convertir la entrada serial de 1 bit, a una salida paralela de 8 bits + valid. Además cuenta las señales *COM* y se activa al recibir cuatro de estas.
- `serial_paralelo2.v`: Recibe los COM e IDL enviados por el módulo `paralelo_serial2` y envía la señal *IDLE_OUT* al módulo de recirculación.
- `phy_tx.v`: Módulo que procesa una señales para transmitirlas posteriormente.
- `phy_rx.v`: Módulo de recepción y procesamiento de las señales transmitidas.
- `phy.v`: Módulo que representa, en sí, a la capa física diseñada.

3. Plan de pruebas

Por cuestiones de tiempo y simplicidad, ambas pruebas aquí descritas se implementaron dentro del mismo archivo `probador.v` del módulo `phy`.

3.1. Prueba con las señales dadas en el enunciado

Esta prueba pretende implementar, aproximadamente, las señales que se mostraban en la descripción de la microarquitectura del proyecto que se discutió en clase. Esto se realiza de forma manual manipulando las señales para que coincidan con las dadas, en el archivo `probador.v` del módulo `phy`.

Una vez finalizado el proyecto, el diseño implementado **pasó esta prueba**.

3.2. Prueba con señales generadas aleatoriamente

En esta prueba se generan las señales necesarias para probar a profundidad esta capa física diseñada, pero de manera completamente aleatoria. Se espera que esto permita, también, encontrar posibles errores en la lógica desarrollada y poder corregirlos.

Una vez finalizado el proyecto, el diseño implementado **pasó esta prueba** y no se notaron errores en el diseño.

4. Instrucciones de uso para la simulación

Todo el código empleado en la elaboración del proyecto, necesario para replicar la simulación del sistema, se puede encontrar en el repositorio de *GitHub*: https://github.com/Darkgambler/PROYECTO1_DIGITALESII.

En primer lugar, cabe destacar que cada uno de los módulos implementados posee, a nivel individual, su propio archivo *makefile* con los siguientes comandos:

- `make all`: Ejecuta la síntesis en yosys y compila el banco de pruebas respectivo. Genera un dumpfile en formato `.vcd`.
- `make compilar`: Compila el banco de pruebas respectivo. Genera un dumpfile en formato `.vcd`.
- `make yosys`: Realiza la síntesis de la descripción conductual respectiva y genera automáticamente la descripción estructural.

- `make cambiar`: Comando que se utilizó como herramienta durante la elaboración del proyecto. Cambia distintos parámetros o nombres de archivos según necesidad de los distintos módulos.

Sin embargo, el proyecto queda finalmente organizado de manera en que tan solo es necesaria la utilización del comando `make all` (o `make`) para recompilar los archivos de un cierto componente, por lo que se recomienda la utilización de este, únicamente.

Para replicar las pruebas aquí descritas y obtener los resultados mostrados en el presente documento, se deben compilar los principales módulos del proyecto, llamados `phy`, `phy_rx` y `phy_tx`, que se encuentran organizados en carpetas con sus respectivos nombres. Para llevar a cabo esto, tan solo se debe ejecutar el comando `make` desde la terminal una vez que se ha posicionado en cada una de las carpetas mencionadas. Una vez hecho esto, se genera un archivo de output con extensión `.vcd` para cada módulo, que contendrá las señales de las simulaciones listas para visualizar en el software *GTKWave*.

Por otro lado, si tan solo desea visualizar la replica exacta de los resultados aquí mostrados, se incluyen a lo largo del proyecto archivos con extensión `.gtkw` que los contienen para su fácil visualización.

5. Ejemplos de los resultados obtenidos

A continuación se presentan algunos ejemplos de resultados obtenidos al simular algunos de los componentes principales del sistema creado, en las figuras 2, 3 y 4.

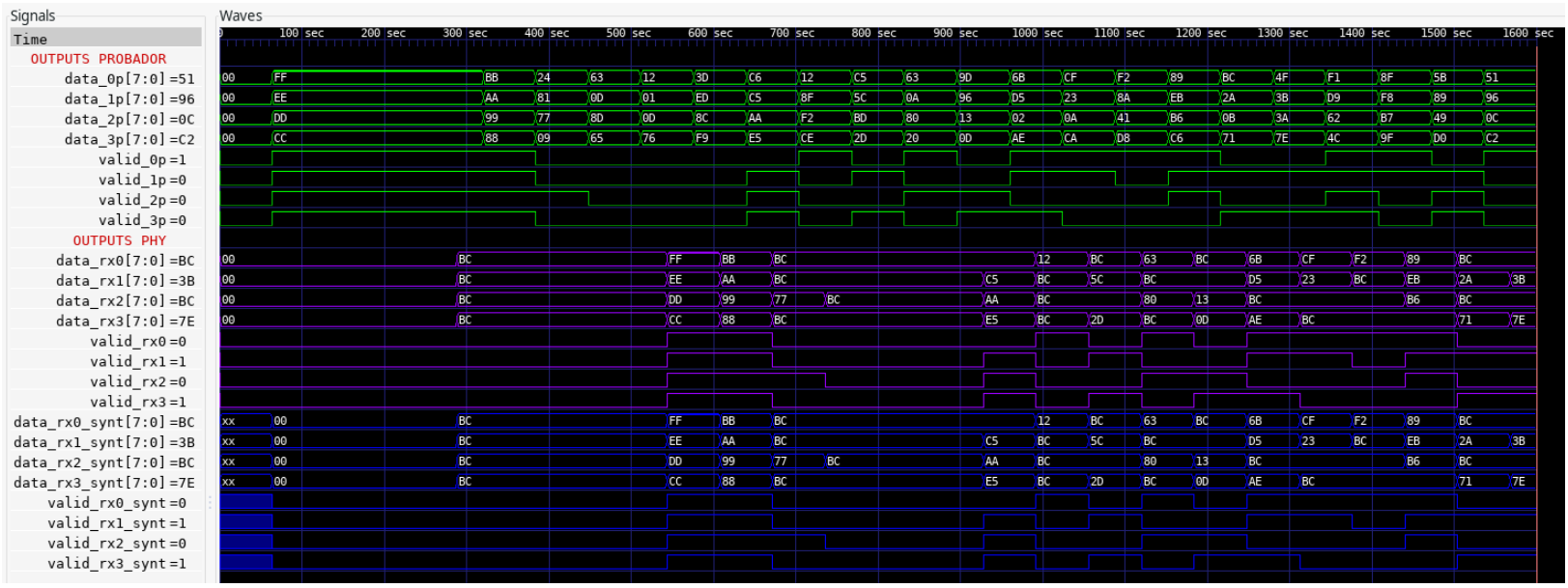

Figura 2: Visualización del comportamiento del módulo principal de la capa física, `phy`.

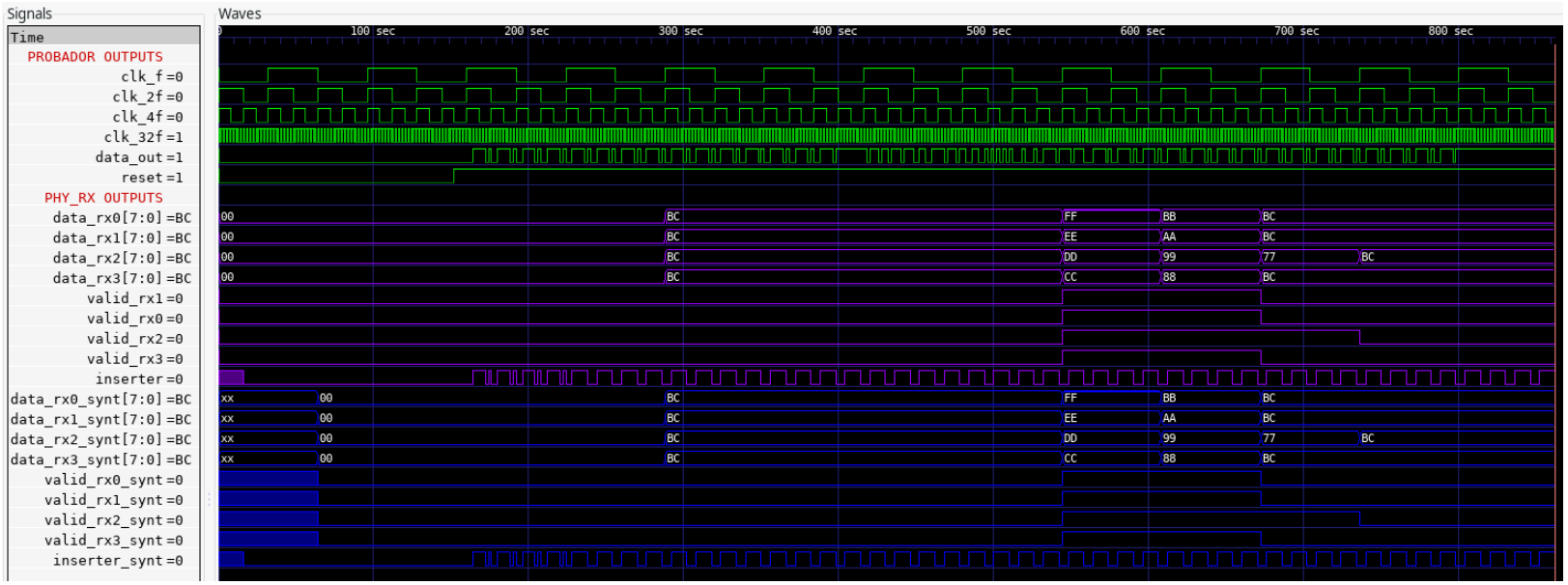

Figura 3: Visualización del comportamiento del módulo receptor de la capa física, `phy_rx`.

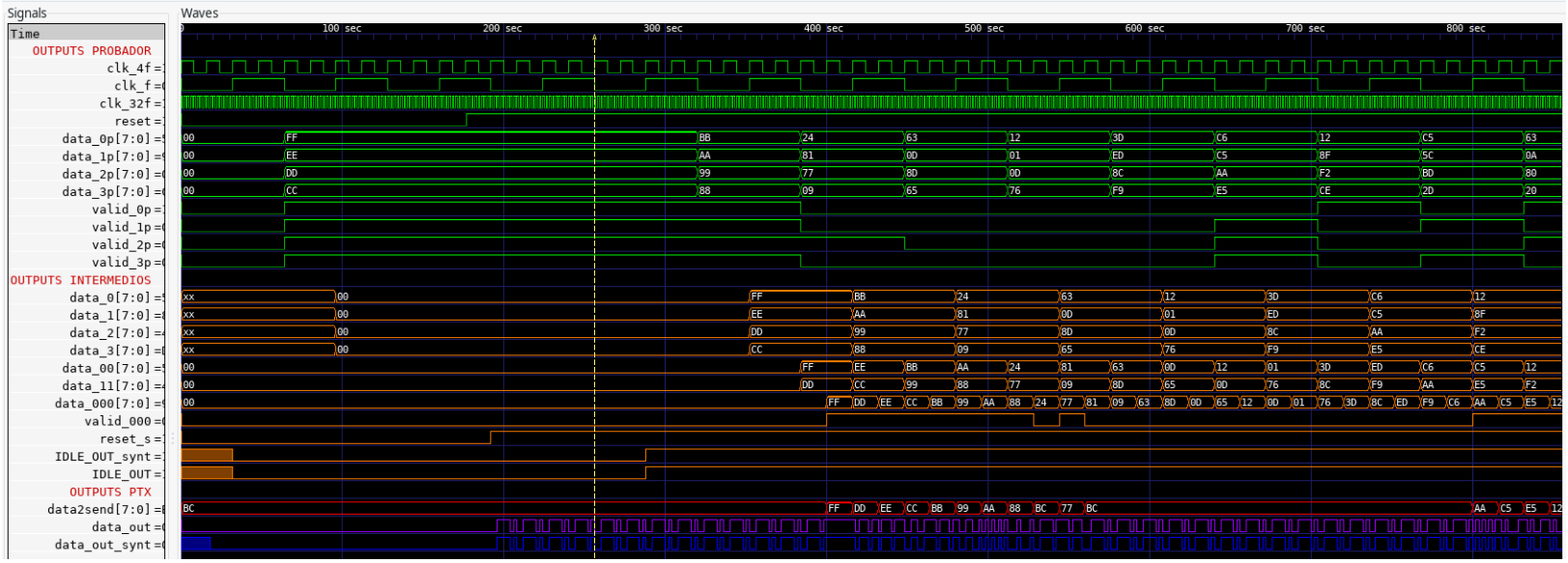

Figura 4: Visualización del comportamiento del módulo de transmisión del sistema, `phy_tx`.

6. Conclusiones y recomendaciones

De la realización del proyecto a nivel general, el funcionamiento observado y los resultados obtenidos, se concluye que:

- La utilización de múltiples etapas para la serialización y/o paralelización de datos resulta bastante ingeniosa y útil, y se puede llevar a cabo sin inconvenientes.
- La lógica a seguir en la creación de los distintos módulos y etapas puede tornarse compleja fácilmente, pero la automatización de algunos procedimientos metódicos, como la síntesis de yosys, puede ser de gran ayuda.
- La idea de integrar las distintas partes de la lógica creada, desde etapas tempranas, puede facilitar mucho el trabajo a futuro.
- Los resultados obtenidos muestran un comportamiento bastante bueno por parte de los distintos módulos creados, y realmente permiten observar el funcionamiento de la capa física.

7. Plan de trabajo y bitácora

El proyecto se inició de manera individual por la incapacidad de encontrar otro integrante, por lo que la programación de los modulos de recirculación, muxes, demuxes y el plan de trabajo fueron realizados antes de que se formara el equipo. Luego cuando se consiguió a un integrante para formar el equipo de trabajo éste manifestó que debido a problemas de salud no estaba al día con la materia del curso, por lo que a mitad de semana se tomó la decisión de que el primer integrante seguiría con todo lo referente a la programación de los módulos restantes y la integración, mientras que el nuevo integrante se encargaría del reporte y la presentación final del proyecto.

El diagrama de la figura 5 resume el plan de trabajo presentado inicialmente, el cual sirvió como guía para la realización del proyecto.

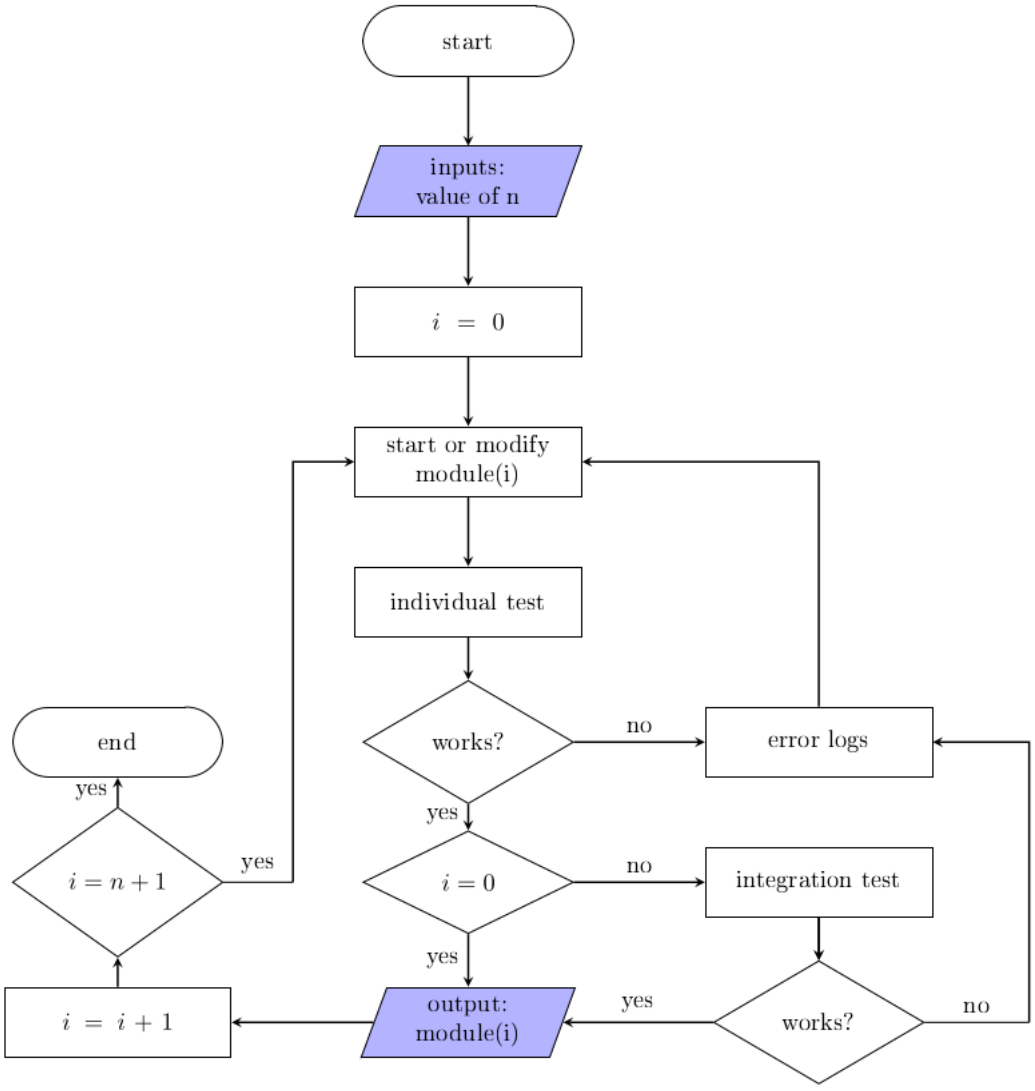

Figura 5: Diagrama de flujo de trabajo

Cuadro 1: Detalle de la primera entrega de módulos

Etapa/Tarea	Nombre	Tiempo	Entrega	Comentario
Muxes/Demuxes	recirculacion.v	3.5 horas	25/05/21	El módulo era sencillo, lo complicado fue programar el probador que hiciera lo esperado, o sea volver a repetir la salida cuando el módulo retornaba la entrada.
Muxes/Demuxes	mux4x2_8bits.v	1.5 horas	25/05/21	El módulo convierte la entrada de 4 líneas de 8 bits + valid a 2 líneas de 8 bits + valid.
Muxes/Demuxes	mux2x1_8bits.v	1.0 horas	25/05/21	El módulo convierte la entrada de 2 líneas de 8 bits + valid a una líneas de 8 bits + valid.
Muxes/Demuxes	demux1x2_8bits.v	2.5 horas	25/05/21	El módulo convierte la entrada de una líneas de 8 bits + valid a dos líneas de 8 bits + valid. Por la funcionalidad fue más complicado de realizar que los muxes.
Muxes/Demuxes	demux2x4_8bits.v	3.0 horas	25/05/21	El módulo convierte la entrada de 2 líneas de 8 bits + valid a 4 líneas de 8 bits + valid. Por la funcionalidad fue más complicado de realizar que los muxes

Cuadro 2: Detalle de la segunda entrega de módulos

Etapa/Tarea	Nombre	Tiempo	Entrega	Comentario
Paralelo/Serial	paralelo_serial1.v	3.5 horas	01/06/21	El módulo convierte la entrada que viene del mux2x1_8bits.v a una salida serial con la frecuencia máxima.
Paralelo/Serial	paralelo_serial2.v	2.5 horas	01/06/21	El módulo se encarga de enviar los COM e IDL al serial_paralelo2.v del módulo transmisor.
Paralelo/Serial	serial_paralelo1.v	4.0 horas	01/06/21	El módulo se encarga de convertir la entrada serie que viene del paralelo serial 1 en una señal de 8bits + valid, cuenta los COM y se activa según las especificaciones.
Paralelo/Serial	serial_paralelo2.v	2.0 horas	01/06/21	El módulo recibe los COM e IDL enviados por el paralelo serial 2 y cumple la importante función de enviar el IDLE_OUT hacia el módulo de recirculación.

Cuadro 3: Detalle de la tercera entrega de módulos

Etapa/Tarea	Nombre	Tiempo	Entrega	Comentario
PHY TX	phy_tx.v	1.5 horas	08/06/21	Módulo de transmisión integrado, llevó poco tiempo debido a que el trabajo grueso se realizó como parte de las pruebas de integración según el plan de trabajo.
PHY RX	phy_rx.v	1.5 horas	08/06/21	Módulo de recepción integrado, llevó poco tiempo debido a que el trabajo grueso se realizó como parte de las pruebas de integración según el plan de trabajo.
PHY	phy.v	2.5 horas	08/06/21	Módulo final, llevó poco tiempo debido a que el trabajo grueso se realizó como parte de las pruebas de integración según el plan de trabajo.

Cuadro 4: Detalle del trabajo extra realizado

Etapa/Tarea	Nombre	Tiempo	Entrega	Comentario
Auxiliar	script bash	3 horas	no aplica	Se creó un script para mejorar la integración de los módulos para cumplir con lo esperado en el plan de trabajo, el script es sencillo pero el tiempo consumido fue en su mayoría debido a su diseño funcional, pruebas y correcciones.
Auxiliar	idletemporal.v	0.5 horas	no aplica	Se creó un módulo auxiliar que sustituyó tempranamente al módulo que enviaba la señal de IDLE_OUT que entra al módulo de recirculación
Auxiliar	integraciones	15.0 horas	no aplica	Para cumplir el plan de trabajo se integraron desde el principio los módulos para garantizar su funcionalidad, esto aumentó mucho el tiempo de trabajo inicial pero a cambio garantizó la funcionalidad del sistema y aceleró la integración de los entregables finales.

Referencias

[1] R. Budruk, D. Anderson, T. Shanley, and I. MindShare, *PCI Express System Architecture*, ser. Mindshare PC System Architecture. Addison-Wesley, 2004. [Online]. Available: https://www.mindshare.com/files/ebooks/PCI%20Express%20System%20Architecture.pdf