Universidad de Costa Rica
Facultad de Ingeniería
Escuela de Ingeniería Eléctrica

IE-0523
Circuitos Digitales II

Reporte del Proyecto #1

Por:

Luis Alonso Rodríguez B76547
Jafet Gutiérrez B73558
Gabriel Araya B80525
Andrés Arias B80661

Ciudad Universitaria Rodrigo Facio, Costa Rica

I ciclo, 2021

Índice

Resumen

El proyecto consiste en un diseño de RTL en verilog de la etapa física de un sistema PCI el cual tiene en su interior un dos módulos, un phy transmisor, y un phy receptor. Estos dos por su parte cuentan con, recirculador y lógica de muxes y demuxes encargados de enrutrar la información en los canales, separando o juntando las entradas según sea necesario para las siguientes etapas. Además, ambos cuentan con módulos serial paralelo y paralelo serial para cumplir con dicha función. El paralelo serial es el encargado de enviar los datos de un módulo a otro. Estos, como su nombre lo indica pasan los datos en serie o en paralelo a su contraparte, con una codificación que indica si es válida o no la información enviada, evitando tener lecturas no deseadas en momentos donde se indica, por ejemplo, un reset.

El objetivo del sistema es precisamente recibir y transmitir la información. Es por esto que como pruebas, se verifica que los valores de entrada aparezcan a la salida también, asegurandose así que, si coninciden, el proceso se realizó de manera satisfactoria pese a las transformaciones internas. Se obtiene un resultado satisfactorio debido a que tanto la descripción conductual en verilog como la estructural procesada por yosys entregan la salida esperada. Para un diseño posterior se recomienda siempre verificar que las entradas y salidas tienen un valor, y no son condiciones no importa y cubrir todas las posibilidades de valores condicionales.

1. Descripción arquitectónica

A continuación el diagrama de arquitectura del sistema completo construido en el proyecto:

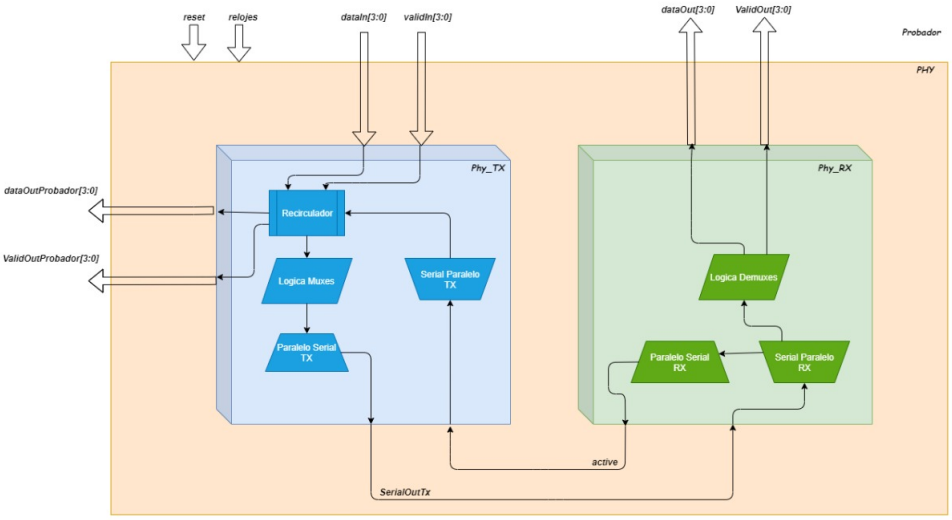

Figura 1: Conexión de Phy con sus respectivas entradas y salidas

El sistema cuenta con una primera etapa de 4 flops con 8 bits más un valid, a la frecuencia más baja del circuito. Después del recirculador, hay una segunda etapa de flops con igual a la anterior que se conecta con los muxes, estos reducen los flops de 4 a 2 flops con una frecuencia de 2 veces la menor. y esto va a una segunda lógica de muxes que nuevamente convierte la señal, esta vez de 2 a 1 y lo envía al paralelo serial. Este pasa de los valores en paralelo a una serie de bits a una frecuencia de 32 veces la más lenta (el clock más rápido) e inserta una señal de BC en caso de que no sea válida la salida. Dicha señal pasa al receptor, serial paralelo, que recupera la señal cuando es distinta de BC y la pasa a su forma en paralelo, y lo envía a demuxes que a su vez lo pasan a flops que revierten el proceso anterior de los muxes y envían el valor a la salida. Por otra parte, la señal también pasa al paralelo serial que convierte la señal para enviarla al transmisor nuevamente a su valor en serie. Este inserta 7C y BC para representar activo o inactivo y el Serial paralelo del transmisor entonces, convierte la señal a paralelo, la pasa al recirculador y este de regreso al sistema y/o al probador.

2. Plan de pruebas

Para las pruebas se decidió recrear las pruebas mostradas en el enunciado del proyecto. De esa forma se fueron diseñando los componentes, de forma que tuviera sentido su función y, a la vez pudieran obtener la misma salida utilizando los valores de entrada de las figuras del enunciado.

Por lo tanto, se prueba y verifica tanto cada componente interno del sistema de forma individual como las partes complejas. El criterio de verificación para cada parte y el sistema completo es el siguiente:

2.1. Lógica de Muxes y Demuxes

Para la primera entrega, al ser Muxes y Demuxes, se planteo el siguiente plan de trabajo:

- Se le dan 4 entradas de 8 bits al sistema de módulos de Mux con sus respectivos selectores.
- Se espera que el sistema tenga en la salida, después de pasar de los Demux, la salida que se recibe de 8 bits.

A continuación se muestra en una imagen el plan de pruebas utilizado. La lógica detrás consiste en que, si el sistema funciona adecuadamente, se debe recibir en la salida, las entradas otorgadas, pero con retraso impuesto por los flip-flop.

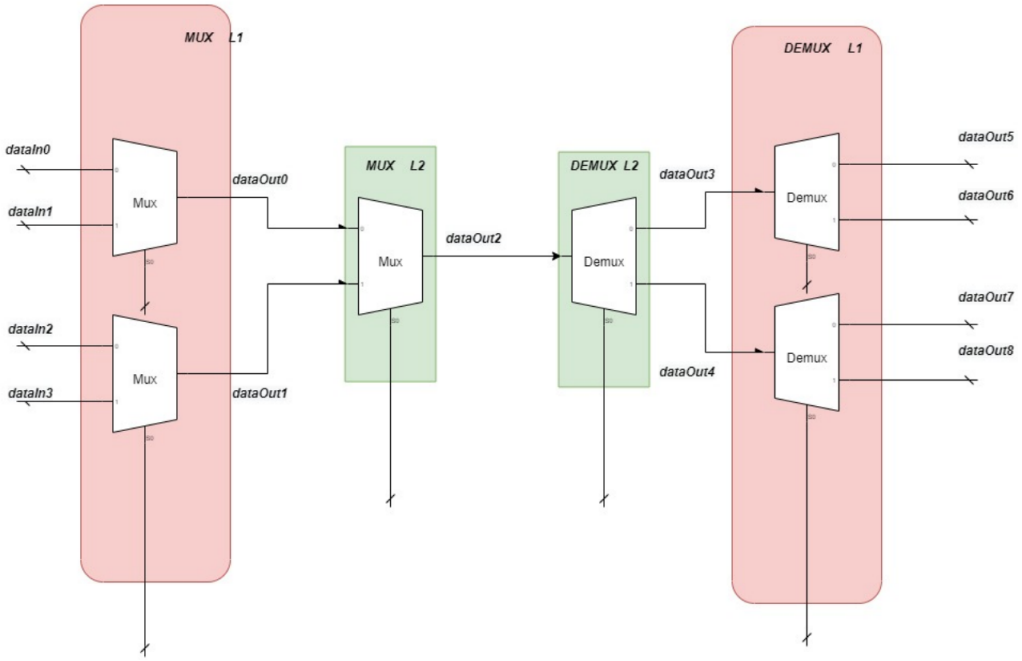

Figura 2: Diagrama del plan de pruebas para muxes y demuxes

Además se cuenta con 3 relojes distintos, para variar en distinto tiempo las señales. El clk_4f es el reloj más veloz, entre los utilizados para los muxes, con 1 unidad de tiempo, el reloj clk_2f es 2 veces más lento que el primero, y finalmente el clk_f es 4 veces más lento que el primero. Se utilizan el 1 y el 2 para variar los selectores, y se utiliza el 3 para cambiar los valores de las entradas del Mux L1.

2.2. Recirculación

El bloque de recirculación opera fundamentalmente como un demultiplexor con una entrada y dos salidas, las tres conformadas a su vez por 4 buses de 8 bits. Adicionalmente cada uno de estos buses cuenta con un cable que lleva un bit de validación. Cuando la entrada de selección (*selector_IDLE*) se encuentre en estado bajo, los valores de las entradas irán a las salidas que vuelven al bloque del probador, y por otro lado, cuando *selector_IDLE* esté en estado alto los datos en las entradas seguirán al bloque de lógica de multiplexores. También cabe mencionar que este módulo opera de manera sincrónica con el reloj de frecuencia más alta del sistema.

En la figura 3 se muestra un diagrama del bloque Recirculador, junto con sus entradas y salidas.

Para verificar la funcionalidad de este bloque se realiza una prueba, en la cual se le aplican 4 entradas de 8 bits que cambian de valor a través del tiempo. Con el propósito de comprobar el efecto del selector, la entrada *selector_IDLE* comenzará la prueba en 0 y cambiará de valor a 1 después de un pequeño periodo. Por lo que se espera que al principio de la prueba las salidas y bits de valid que van a la lógica de multiplexores mantengan un valor indefinido. Después del cambio del selector estas salidas deberían de empezar a cambiar conforme lo hacen las entradas, mientras que las salidas que van al probador tendrían que conservar su valor anterior.

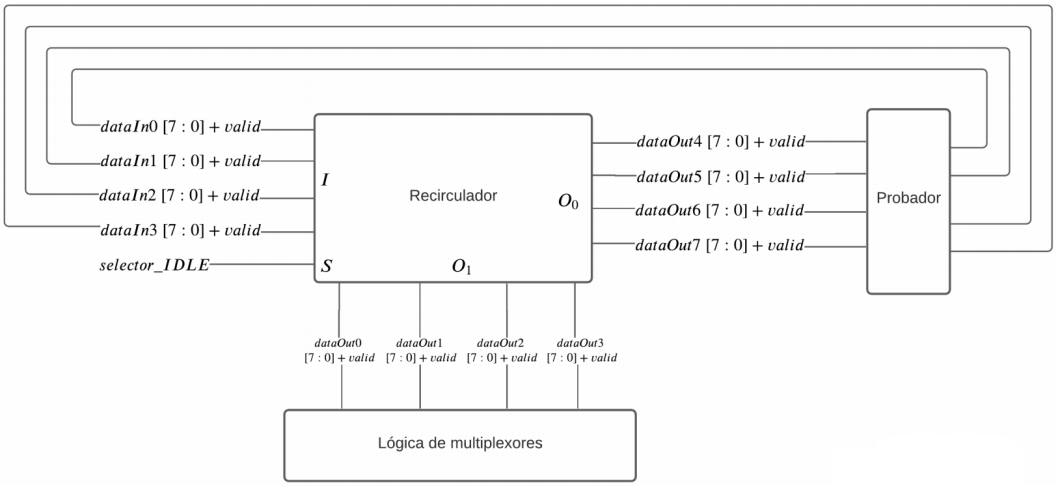

Figura 3: Diagrama de Recirculación

2.3. Serial-Paralelo y Paralelo-Serial Transmisor

2.3.1. Paralelo-Serial

El bloque funciona utilizando el reloj clk_32f para pasar las entradas en paralelo a uno serial que cambia de valor con cada flanco creciente. También toma en cuenta el bit de valid, en el caso que sea 1 se obtendrá la salida serial del valor que venía del multiplexor L2; en el caso que sea 0 se obtendría que la salida va a ser el serial del número BC.

2.3.2. Serial-Paralelo

El bloque opera de forma que primero pasa la entrada de serial a paralelo para así trabajar luego directamente con el bus de datos. Luego se comprueba con un contador cuantas entradas han sido el número BC, esto para que cuando se cuenten 4 o más entonces se proceda a pasar el bit de active a 1. En la salida del IDLE out se aplica una lógica AND con el active que se calculó anteriormente y la entrada actual. Su la entrada IDLE in es 1 y se tiene el active arriba, entonces la salida IDLE out también sera 1. Luego se esto se envía la salida al recirculador para que este realice la acción de dejar pasar los siguientes datos o de repetir el proceso para evaluar el error en los datos.

2.4. Serial-Paralelo y Paralelo-Serial Receptor

2.4.1. Paralelo-Serial

El paralelo a serial en el receptor detecta la señal de active, y dependiendo de este valor, se encarga de insertar 7C o BC en hexadecimal y pasar la señal a serial. Cuando el valor es inactivo, envía BC y cuando es activo, envía 7C.

2.4.2. Serial-Paralelo

El bloque recibe primero la entrada en serial a una alta frecuencia, detectando valores de BC, cuando recibe 4 o más valores de BC pasa a active a 1, transformando los valores a paralelo para enviarlos, tanto a la lógica de demuxes que se encarga de pasarlo a la salida dataOut y validOut, como nuevamente a un paralelo serial en el receptor. El validOut se mantendrá arriba siempre que active esté arriba y no se tenga un BC en los datos.

A continuación el diagrama de esta etapa:

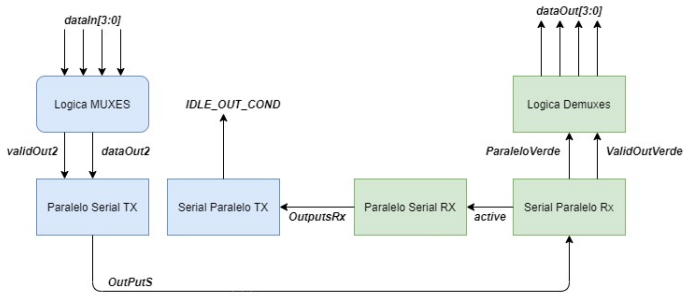

Figura 4: Diagrama interno para las pruebas de Serial Paralelo y Paralelo Serial

2.5. Sistema completo

Para esta tercera entrega se realiza un plan de pruebas similar a los previos. El enfoque principal de las pruebas consiste en alimentar el sistema con una entrada dataIn que recorrerá todos los módulos del sistema. Finalmente, debe entregar una salida dataOut que debe ser igual a data in simplemente atrasada ciertos ciclos de reloj debido a los flip flop que recorre. A continuación se muestran los diagramas que explican visualmente la información.

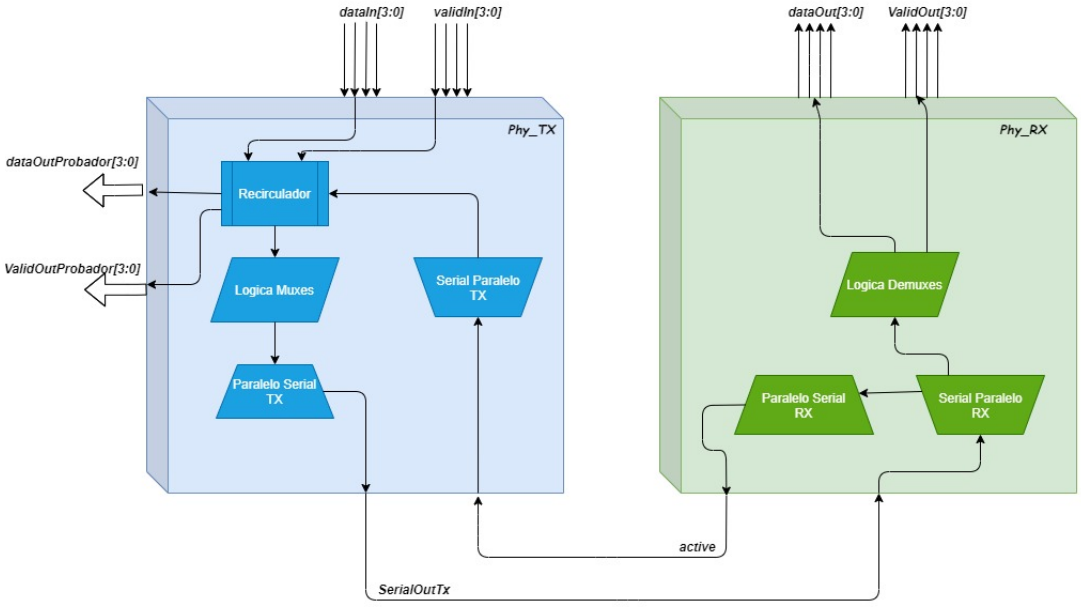

Figura 5: Conexión de PhyRX y PyTX con sus respectivas entradas y salidas

Se puede observar en el diagrama de la figura 1 que consiste en el módulo completo, sintetizado, con todos sus elementos internos, y entradas, así como las salidas hacia el medio y hacia el probador. Además recibe la señal de reloj y reset que afectan con su valor todos los módulos internos. Tanto los módulos individuales de PhyRX y PhyTX como el sistema completo, otorgan a la salida el valor esperado, lo que implica que al recorrer el sistema, las transformaciones realizadas son correctas, y de esta manera se verifica el funcionamiento.

Para esta entrega tanto el funcionamiento conductual como el estructural de todo bloque individual y general, fueron probados y se verifica su respuesta ante los distintos estímulos de entrada como las correctas y deseadas.

3. Instrucciones de utilización de la simulación

A continuación se muestra el código del *Makefile*:

```
#Entrega #3
Prueba_PHYTX/RX:
        sed -i '75 s/posedge clk_4f/*/g' SerialParalelo_PhyRX.v
        yosys -s sintesis_rx.ys
        sed -i '75 s/*/posedge clk_4f/g' SerialParalelo_PhyRX.v
        sed -i 's/cond/synth/g' Phy_Rx_synth.v
        sed -i 's/DeMux2x1/DeMux2x1_synth/g' Phy_Rx_synth.v
        sed -i 's/DeMuxL1/DeMuxL1_synth/g' Phy_Rx_synth.v
        sed -i 's/DeMuxL2/DeMuxL2_synth/g' Phy_Rx_synth.v
        sed -i 's/Mux2x1/Mux2x1_synth/g' Phy_Rx_synth.v
        sed -i 's/MuxL1/MuxL1_synth/g' Phy_Rx_synth.v
        sed -i 's/MuxL2/MuxL2_synth/g' Phy_Rx_synth.v
        sed -i 's/ParaleloSerial_PhyRX/ParaleloSerial_PhyRX_synth/g' Phy_Rx_synth.v
        sed -i 's/ParaleloSerial_PhyTX/ParaleloSerial_PhyTX_synth/g' Phy_Rx_synth.v
```

```
	sed -i 's/Phy_Rx/Phy_Rx_synth/g' Phy_Rx_synth.v
	sed -i 's/Phy_Tx/Phy_Tx_synth/g' Phy_Rx_synth.v
	sed -i 's/Recirculador/Recirculador_synth/g' Phy_Rx_synth.v
	sed -i 's/SerialParalelo_PhyRX/SerialParalelo_PhyRX_synth/g' Phy_Rx_synth.v
	sed -i 's/SerialParalelo_PhyTX/SerialParalelo_PhyTX_synth/g' Phy_Rx_synth.v
	yosys -s sintesis_tx.ys
	sed -i 's/cond/synth/g' Phy_Tx_synth.v
	sed -i 's/DeMux2x1/DeMux2x1_synth/g' Phy_Tx_synth.v
	sed -i 's/DeMuxL1/DeMuxL1_synth/g' Phy_Tx_synth.v
	sed -i 's/DeMuxL2/DeMuxL2_synth/g' Phy_Tx_synth.v
	sed -i 's/Mux2x1/Mux2x1_synth/g' Phy_Tx_synth.v
	sed -i 's/MuxL1/MuxL1_synth/g' Phy_Tx_synth.v
	sed -i 's/MuxL2/MuxL2_synth/g' Phy_Tx_synth.v
	sed -i 's/ParaleloSerial_PhyRX/ParaleloSerial_PhyRX_synth/g' Phy_Tx_synth.v
	sed -i 's/ParaleloSerial_PhyTX/ParaleloSerial_PhyTX_synth/g' Phy_Tx_synth.v
	sed -i 's/Phy_Rx/Phy_Rx_synth/g' Phy_Tx_synth.v
	sed -i 's/Phy_Tx/Phy_Tx_synth/g' Phy_Tx_synth.v
	sed -i 's/Recirculador/Recirculador_synth/g' Phy_Tx_synth.v
	sed -i 's/SerialParalelo_PhyRX/SerialParalelo_PhyRX_synth/g' Phy_Tx_synth.v
	sed -i 's/SerialParalelo_PhyTX/SerialParalelo_PhyTX_synth/g' Phy_Tx_synth.v
	emacs BancoPrueba_Entrega3.v -f verilog-batch-auto --batch
	sed -i '52 s/active/SerialOut_cond/2g' BancoPrueba_Entrega3.v
	sed -i '71 s/active/SerialOut_synth/2g' BancoPrueba_Entrega3.v
	sed -i '98 s/SerialIn/SerialOutTx_cond/2g' BancoPrueba_Entrega3.v
	sed -i '117 s/SerialIn/SerialOutTx_synth/2g' BancoPrueba_Entrega3.v
	iverilog BancoPrueba_Entrega3.v
	./a.out
	rm a.out
	gtkwave Prueba_Final_Entrega3.vcd

#Prueba Final
Prueba_PHY:
	emacs Phy.v -f verilog-batch-auto --batch
	sed -i '60 s/active/SerialOut_cond/2g' Phy.v
	sed -i '75 s/dataOutProbador0/dataOutProbador0_cond/2g' Phy.v
	sed -i '76 s/dataOutProbador1/dataOutProbador1_cond/2g' Phy.v
	sed -i '70 s/dataOut0_Rx_cond/dataOut0_cond/2g' Phy.v
	sed -i '71 s/dataOut1_Rx_cond/dataOut1_cond/2g' Phy.v
	sed -i '72 s/dataOut2_Rx_cond/dataOut2_cond/2g' Phy.v
	sed -i '73 s/dataOut3_Rx_cond/dataOut3_cond/2g' Phy.v
	sed -i '74 s/validOut0_Rx_cond/validOut0_cond/2g' Phy.v
	sed -i '75 s/validOut1_Rx_cond/validOut1_cond/2g' Phy.v
	sed -i '76 s/validOut2_Rx_cond/validOut2_cond/2g' Phy.v
	sed -i '77 s/validOut3_Rx_cond/validOut3_cond/2g' Phy.v
	sed -i '79 s/SerialIn/SerialOutTx_cond/2g' Phy.v
	sed -i '75 s/posedge clk_4f/*/g' SerialParalelo_PhyRX.v
	yosys -s sintesis_phy.ys
```

```
sed -i '75 s/*/posedge clk_4f/g' SerialParalelo_PhyRX.v
sed -i 's/module Phy(/module Phy_synth(/g' Phy_synth.v
sed -i 's/cond/synth/g' Phy_synth.v
sed -i 's/DeMux2x1/DeMux2x1_synth/g' Phy_synth.v
sed -i 's/DeMuxL1/DeMuxL1_synth/g' Phy_synth.v
sed -i 's/DeMuxL2/DeMuxL2_synth/g' Phy_synth.v
sed -i 's/Mux2x1/Mux2x1_synth/g' Phy_synth.v
sed -i 's/MuxL1/MuxL1_synth/g' Phy_synth.v
sed -i 's/MuxL2/MuxL2_synth/g' Phy_synth.v
sed -i 's/ParaleloSerial_PhyRX/ParaleloSerial_PhyRX_synth/g' Phy_synth.v
sed -i 's/ParaleloSerial_PhyTX/ParaleloSerial_PhyTX_synth/g' Phy_synth.v
sed -i 's/Phy_Rx/Phy_Rx_synth/g' Phy_synth.v
sed -i 's/Phy_Tx/Phy_Tx_synth/g' Phy_synth.v
sed -i 's/Recirculador/Recirculador_synth/g' Phy_synth.v
sed -i 's/SerialParalelo_PhyRX/SerialParalelo_PhyRX_synth/g' Phy_synth.v
sed -i 's/SerialParalelo_PhyTX/SerialParalelo_PhyTX_synth/g' Phy_synth.v
emacs BancoPrueba_PHY.v -f verilog-batch-auto --batch
sed -i '120 s/dataOut0_Rx_cond/dataOut0_cond/2g' BancoPrueba_PHY.v
sed -i '121 s/dataOut1_Rx_cond/dataOut1_cond/2g' BancoPrueba_PHY.v
sed -i '122 s/dataOut2_Rx_cond/dataOut2_cond/2g' BancoPrueba_PHY.v
sed -i '123 s/dataOut3_Rx_cond/dataOut3_cond/2g' BancoPrueba_PHY.v
sed -i '124 s/validOut0_Rx_cond/validOut0_cond/2g' BancoPrueba_PHY.v
sed -i '125 s/validOut1_Rx_cond/validOut1_cond/2g' BancoPrueba_PHY.v
sed -i '126 s/validOut2_Rx_cond/validOut2_cond/2g' BancoPrueba_PHY.v
sed -i '127 s/validOut3_Rx_cond/validOut3_cond/2g' BancoPrueba_PHY.v
sed -i '136 s/dataOut0_Rx_synth/dataOut0_synth/2g' BancoPrueba_PHY.v
sed -i '137 s/dataOut1_Rx_synth/dataOut1_synth/2g' BancoPrueba_PHY.v
sed -i '138 s/dataOut2_Rx_synth/dataOut2_synth/2g' BancoPrueba_PHY.v
sed -i '139 s/dataOut3_Rx_synth/dataOut3_synth/2g' BancoPrueba_PHY.v
sed -i '140 s/validOut0_Rx_synth/validOut0_synth/2g' BancoPrueba_PHY.v
sed -i '141 s/validOut1_Rx_synth/validOut1_synth/2g' BancoPrueba_PHY.v
sed -i '142 s/validOut2_Rx_synth/validOut2_synth/2g' BancoPrueba_PHY.v
sed -i '143 s/validOut3_Rx_synth/validOut3_synth/2g' BancoPrueba_PHY.v
iverilog BancoPrueba_PHY.v
./a.out
rm a.out
gtkwave Prueba_Final_PHY.vcd
```

Para ejecutar las distintas pruebas solo es necesario correr el comando *make Prueba_#*, donde # es el resto del nombre de la prueba como se muestra en el *Makefile*. *Prueba_PHYTX/RX* ejecuta las pruebas de los módulos *Phy_Tx.v* y *Phy_Rx.v*, mientras que *Prueba_PHY* ejecuta la simulación del módulo *Phy.v*. En términos generales cada prueba realiza lo siguiente:

- Con Yosys se realiza la síntesis de los diseños conductuales y se generan descripciones estructurales genéricas.
- Se cambia el nombre de los módulos dentro de los archivos sintetizados con el comando *sed*.

- Con Verilog se compila el archivo de banco de prueba, y se genera su correspondiente archivo .vcd.
- Se despliega GTKWave con al archivo .vcd para poder observar las respuestas temporales de los modelos conductuales y estructurales desarrollados.

4. Errores e inconvenientes el proyecto

1. Inicialmente se empezó trabajando sin inicializar las variables, y esto luego al sintetizarlo con Yosys causaba que parte de las salidas del mpdulo estructural durante unos ciclos de reloj tuvieran el valor x (color rojo en el diagrama de tiempo). Esto se muestra a continuación:

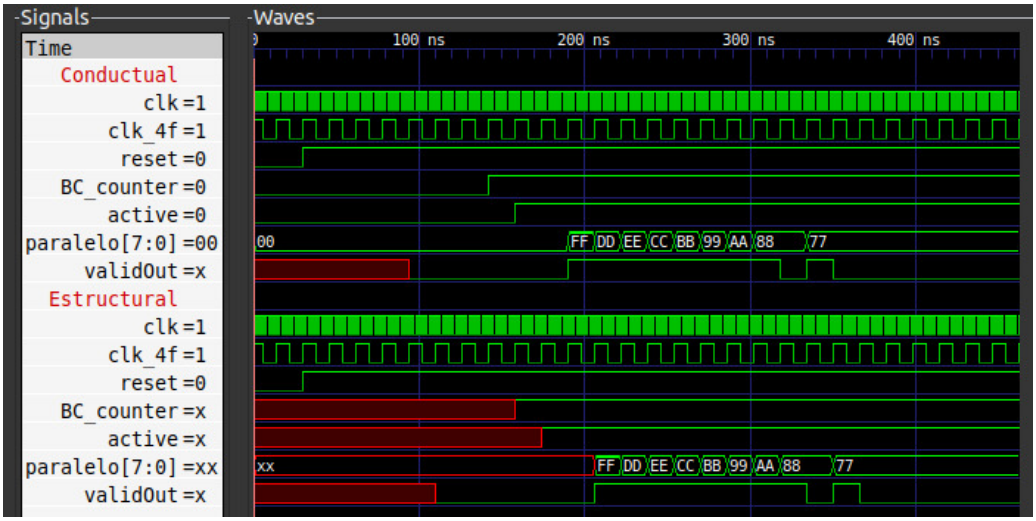

Figura 6: Error de instanciación

2. Otros diversos problemas se tuvieron con los sintetizados de Yosys, debido a que, por faltas de instanciaciones en el código de los módulos conducturales, se asignaban algunas salidas directamente en su valor de cero.
3. Se cometió el error de modificar el valor de una misma señal en distintos posedge clock, cosa que causaba al sintetizar un corto circuito, y por lo tanto las salidas mantuvieran su valor durante toda la prueba.
4. En ocasiones ocurrieron problemas al tener condicionales if sin su respectivo else o sin un if que cumpliera las demás condiciones al sintetizar, lo que causaba un estado sin valor definido.
5. Al momento de sintetizar los módulos más grande y complejos (TX, RX y PHY) se tuvo el inconveniente de que se tomaron las entradas en diferente orden comparado al módulo estructural. Esto provoco que las salidas fueran por distintos puntos/cables con un valor correcto. Esto se solucionó modificando los muxes y demuxes para evitar este tipo de errores al sintetizar.

5. Resultados y análisis

Primeramente, se logró igualar la salida del Demux L2 a la entrada del recirculador. Esto se traduce a que todos los demás componentes funcionan correctamente. También se comprobó el hecho de que si el valid, en algunos ciclos, es igual a cero, entonces la salida del parelelo-serial TX mandaría BC's hacia el serial-paralelo RX. Entonces por parte de la lógica secuencial del circuito se afirma que cumple las funciones respectivas de la capa *Data Link*.

Por parte de cada uno de los módulos, los muxes y los demuxes no representaron problema tanto en su confección como en sus pruebas. En cuanto al recirculador, al inicio presentó problemas durante la confección de su versión estructural, y posteriormente con el diseño de versión conductual. Pero los tres componentes pudieron arreglarse y funcionan correctamente cada uno por separado y juntos.

Con respecto a los seriales-paralelos y los paralelos-seriales, estos fueron los módulos que representaron mayor dificultad y consecuentemente requirieron de más trabajo y análisis que los demás. Su desempeño es bueno, a pesar de los diversos errores que se tuvieron al momento de su creación. Las señales que cada uno de ellos envía a los demás módulos concuerda con su lógica. Sin embargo la mayor dificultad fue, tal y como se menciona en la sección de errores, la síntesis del diseño conductual. Se realizaron en cierto momento varias versiones del código conductual, debido a que la síntesis continuaba fallando (el detalle de algunos problemas se explica en los errores), pero se optó por el diseño conductual entregado debido a que los cambios necesarios en el conductual para obtener a la salida un estructural funcional fueron más simples.

Entre los aspectos positivos del diseño se encuentra, que no hubieron problemas de sincronización entre los relojes, los módulos se comunican de manera correcta, y reaccionan de la manera esperada a la señal de reset en caso de ser necesaria. Dicho esto, se concluye que las pruebas realizadas exitosamente y los resultados obtenidos a nivel general fueron correctos.

En la figura 7 se muestra la prueba que demuestra los resultados correctos conforme a las entradas suministradas:

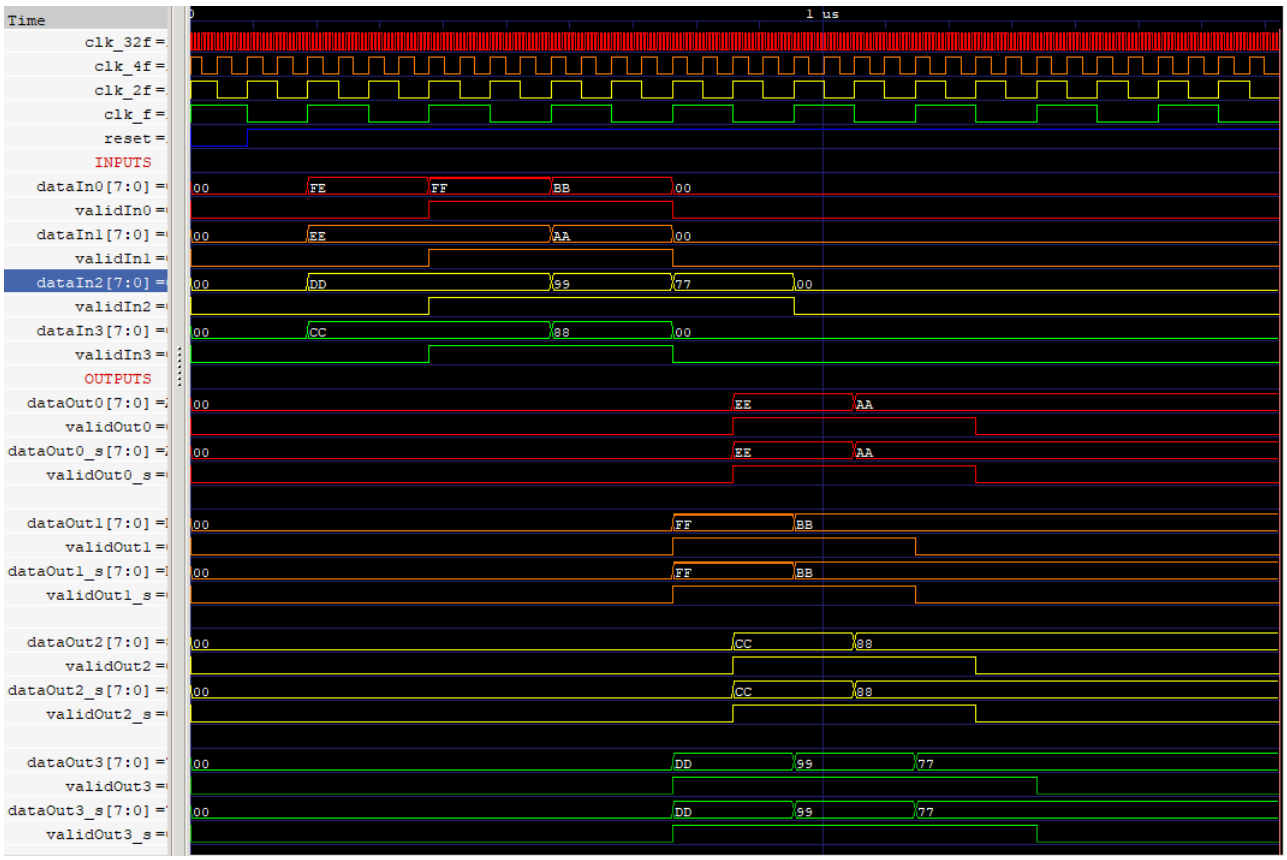

Figura 7: Resultados de la prueba del módulo Phy.v

6. Conclusiones

El diseño de hardware de la capa Data Link de un PCIE represento un reto para cada uno de los participantes del proyecto. El desarrollo de este proyecto conllevo diversos análisis de lógica y diseño de hardware que resultaran útiles para los próximos trabajos de este curso y los siguientes en la carrera. Se aprendió a través de prueba y error los conceptos básicos del diseño de RTL en el lenguaje Verilog.

Con el proyecto los estudiantes lograron familiarizarse más con el funcionamiento del hardware en distintos equipos. Por ejemplo, se aprendió sobre la utilización de determinadas señales de control como lo fueron el BC y el $7C$. También se determinó como lograr el cambio de un bus de datos a uno serial con los ciclos de un reloj rápido y uno más lento.

Se lograron cumplir los objetivos planteados, obteniendo las reacciones esperadas ante las entradas del circuito, pese a las complicaciones. El mayor reto consistió en la síntesis de los módulos y por ende se recomienda verificar siempre el diseño conductual, que este esté completo y cumpla todas las condiciones así como todas sus señales con un valor. El circuito, dado el uso de múltiples flip flop tiene un atraso normal y esperado en la salida ya que va a tardar un ciclo adicional por cada flip flop en pasar la salida. Aparte de esto, si bien están sincronizados, no todos los clock van a una misma frecuencia.

En cuanto al trabajo realizado, tomó esfuerzo y tiempo adicional al calculado debido a la gran cantidad de inconvenientes que aparecían en el proceso y al poco manejo inicialmente del lenguaje en verilog y sus necesidades o limitaciones. La mayor ventaja en cuanto al plan

de trabajo es que todo el equipo logró trabajar muy en conjunto y con apoyo constante de los distintos miembros, y esto ayudó a depurar juntos los distintos módulos y completar las entregas de manera funcional.

En cuanto a y pruebas adicionales que se podrían haber realizado, es principalmente verificar el funcionamiento del circuito fuera de condiciones estables y en un ambiente menos controlado, donde las entradas pueden ser constantes, muy cambiantes y el hecho de cumplir o no varias condiciones al mismo tiempo podría alterar el funcionamiento del circuito. Además, sería positivo verificar la síntesis por ejemplo con una biblioteca que incluya retardos. Sin embargo, pruebas como estas no se consideran ya que no solo complicarían innecesariamente el diseño de lo que se solicita, sino que además, por cuestiones de tiempo y objetivos no era pertinente.

7. Plan de Trabajo

Tema	Fecha	Encargado
Conductual Muxes	21 de mayo	Andrés Arias
Testbench Estructural/Conductual	21 de mayo	Andrés Arias
Conductual Demuxes	21 de mayo	Gabriel Araya
Plan de pruebas Muxes/Demuxes	21 de mayo	Gabriel Araya
Sintesis Modulos	23 de Mayo	Luis Rodriguez
Recircuilador completo	23 de Mayo	Jafet Gutierrez
Unir modulos Conductuales y Estructurales en 1 workbench	1 de Junio	Gabriel Araya
Paralelo serial TX	1 de Junio	Gabriel Araya
Serial Paralelo TX	1 de Junio	Jafet Gutierrez
Paralelo Serial RX	1 de Junio	Andrés Arias
Serial Paralelo RX	1 de Junio	Luis Rodriguez
Workbench Comparar estructual y conductual y probadores	8 de Junio	Andrés Arias
Phy TX	8 de Junio	Luis Rodriguez
Phy RX	8 de Junio	Gabriel Araya
Phy	8 de Junio	Jafet Gutierrez
AUTOINST	8 de Junio	Todos
Bitácora	12 de Junio	Gabriel Araya
Reporte	12 de Junio	Todos
Presentación	12 de Junio	Todos

Cuadro 1: Plan de Trabajo