Universidad de Costa Rica
Facultad de Ingeniería
Escuela de Ingeniería Eléctrica

IE-0523
Circuitos Digitales II

Reporte del Proyecto #2

Por:

Luis Alonso Rodríguez B76547
Jafet David Gutiérrez Guevara B73558
Gabriel Araya Mora B80525
Andrés Arias Campos B80661

Ciudad Universitaria Rodrigo Facio, Costa Rica

I ciclo, 2021

Índice

Resumen

El proyecto se basa en la creación de la capa de transacción del PCIE. Esta capa se modela de tal forma que lee los datos de entrada, los guarda en una memoria de entrada para posteriormente redirigirlos al puerto de salida en el que se requieran bajo una prioridad dada. Además cuenta con un árbitro para el manejo de las memorias internas (en este caso FIFOs), que se encarga de manejar las prioridades de los FIFOS de entrada, y limitar el paso de información cuando se llenan los FIFOs. Por otra parte, la capa de transacción también cuenta con una máquina de estados que controla la etapa en la que se encuentra el módulo.

1. Investigación

1.1. QoS (Calidad de Servicio)

El QoS o Quality of Service es la capacidad de controlar los mecanismos de gestión de tráfico en la red de manera que la red satisfaga las necesidades de servicio de aplicaciones específicas y usuarios sujetos a las políticas de la red. Las redes QoS deben tener mecanismos para controlar la asignación de recursos entre aplicaciones y usuarios. [1] Para poder definir que es QoS existen multiples parámetros cuantitativos medibles.

- Bandwidth: Traducido al español como ancho de banda, define la capacidad de transmisión de una linia eléctrica. Describe el rango de la la gama de velocidades de transmisión posibles, o frecuencias. En la realidad, describe el tamaño del "pipe"que una aplicación necesita para comunicarse por la red. El ancho de banda de canal determina la capacidad de canal, que corresponde a la máxima tasa de información que puede ser transmitida.
- Packet Delay and Jitter: El delay o retraso, consiste en tres tipos diferentes, retardo de serialización, retardo de propagación, y retardo de conmutación. El retardo de serialización o transmisión, es el tiempo que toma al dispositivo sincronizar un paquete a una tasa de salida especifica. El retardo de transmisión es una función del ancho de banda y el tamaño del paquete, su valor depende directamente de ambos. El retardo de propagación es el tiempo que toma a un bit viaja del transmisor al receptor, es una función de la distancia recorrida y el medio.
- Packet Lost: La perdida de paquetes puede ocurrir debido a puntos de congestionamiento cuando el número de paquetes entrantes excede significativamente el tamaño de la cola. Además, paquetes corruptos del cable de transmisión también puede causar perdida de paquetes, y una alta cantidad de perdida de paquetes puede causar que las aplicaciones no funcionen correctamente.

Además, el QoS cuenta con distintos niveles:

- Best effort service: En este nivel no proporciona garantía alguna. Representa el extremo más bajo de los niveles. Muchas aplicaciones funcionan bien con este nivel. El único criterio es si la transferencia fue completada exitosamente o no.
- Soft QoS: También es conocido como servicio diferenciado. No cuenta con garantías absolutas, por el contrario, prioridades distintas se asignan a tareas distintas, se agrupan las aplicaciones en distintas clases de prioridades.
- Hard QoS: También es comocida como servicio garantizado. Representa el nivel de aplicaciones de QoS que requieren garantias absolutas en el minimo recurso necesitado de la red para funcionar correctamento o funcionar del todo. La reserva previa de recursos de red en un trayecto suele realizarse para que la red pueda proporcionar, o denegar, la garantía requerida.

1.2. Arbitraje en sistemas digitales

Cuando a una unidad de hardware llegan varios buses de datos, se utiliza los protocolos de arbitraje. Esto para evitar que más de uno entre al mismo tiempo y produzca fallos en la transmisión de los datos. Los protocolos organizan el uso compartido del bus, estableciendo cuál posee prioridad en la unidad solicitada, y garantizando que el acceso al bus es realizado por un solo *master*. [2]

Los protocolos de arbitraje se dividen principalmente en dos tipos: los centralizados, que utilizan una unidad llamado árbitro del bus, que se encarga de gestionar de forma centralizada el uso del bus desde una unidad individual o integrado con otro componentes/sección; y los distribuidos, en esta no hay un árbitro como tal, sino que cada bus de datos se gestiona desde forma distribuida entre las unidades de acceso. Algunos tipos de protocolos de arbitraje son:

1. Encadenamiento (*daisy chaining*) de dos/tres/cuatro señales: se basa en la utilización de cuatro señales para el control de todos los buses, estas serían: concesión, petición, ocupación, reconocimiento. La primera entra al árbitro mientras que las demás van a los master.
2. Concesión por encuesta (*polling*): sustituye la línea encadenada de concesión del bus por un conjunto de líneas que permiten acceder de forma selectiva a la dirección asignada cada *master* sobre estas líneas.
3. Señales independientes: utiliza una línea de concesión específica para cada línea de petición independiente. Esta alternativa tienen la ventaja que el árbitro puede aplicar distintos algoritmos de decisión en caso de peticiones simultaneas.

1.3. Priority Flow Control

El Priority Flow Control (PFC; IEEE 802.1bb) es un mecanismo que previene la pérdida de datos debido a la falta de espacio de buffer o tráfico en ráfaga [3]. Para explicar el PFC es útil entender primeramente el funcionamiento de otro mecanismo, conocido como LFC (Link-level Flow Control). En el momento que el umbral de un buffer es excedido debido a congestión, el LFC envía una trama *pause* para detener toda la transmisión de datos en el enlace por un periodo de tiempo específico. Cuando la congestión se mitiga, es decir, el tráfico vuelve a encontrarse bajo el umbral configurado, una trama *resume* se genera para reiniciar la transmisión de datos en el enlace [4].

En contraste, durante la congestión, el PFC envía una trama *pause* que indica cual valor de CoS (Class of Service) necesita ser pausado. Una trama *pause* de PFC contiene un valor de temporizador de 2 octetos para cada CoS que indica la longitud de tiempo que el tráfico necesita ser pausado. La unidad de tiempo para el temporizador se específica en el *pause quanta*. Un *quanta* es el tiempo requerido para transmitir 512 bits a la velocidad del puerto. El rango va de 0 a 65535. Una trama *pause* con un *quanta* de cero le indica a una trama *resume* que reinicie el tráfico pausado.

El PFC le solicita al sistema que detenga las tramas de un valor particular de CoS al enviar una trama *pause* a una bien conocida dirección de multidifusión, como se puede observar en la figura 1. Esta trama *pause* es una trama de un salto que no es reenviada cuando lo recibe el sistema. Cuando la congestión es mitigada, el PFC puede solicitar al sistema que reinicie la transmisión de tramas.

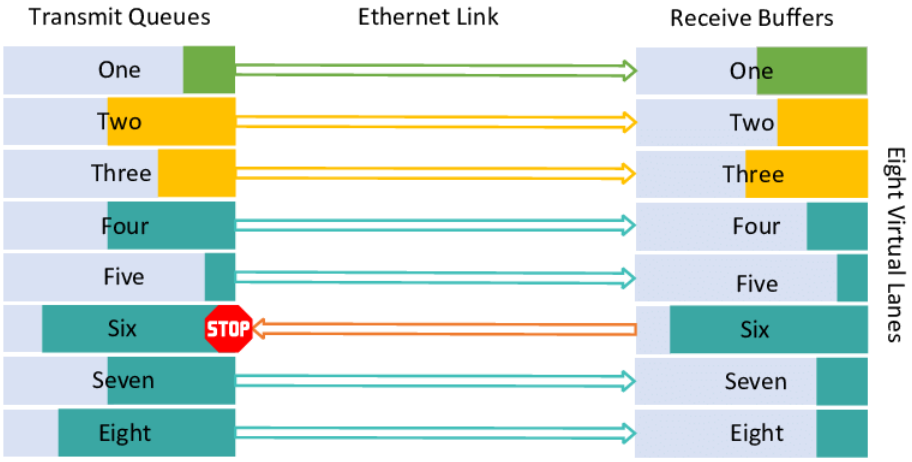

Figura 1: Priority Based Flow-Control (PFC) [241], [243]: IEEE 802.3 trama *pause* con datos PFC extendidos se envía a la estación ascendente que solicita al sexto nivel de prioridad detener la transmisión. [3]

1.4. ¿Cómo se relacionan los créditos con Flow Control?

- Un mecanismo de *flow control* básicamente se encarga de prevenir que los recibidores lentos, o bien los recibidores que están ocupados haciendo otras tareas dentro del circuito se llenen y se cause un "overflow" debido a que los transmisores sacan datos muy rápido. Existen dos tipos de mecanismos de *flow control*, los estáticos y los dinámicos. Sin embargo ambos se basan en el uso de *Créditos* para mantener récord de la cantidad de casillas disponibles en los recibidores.[5]

  - El mecanismo de *flow control* basado en créditos más básico, asegura que los emisores tengan cierto espacio del lado de los receptores. La cantidad exacta de espacio que tienen los *buffers* es explícitamente denotado por el número de créditos que le manda el emisor al iniciar todo el sistema. Cabe decir que el valor de los créditos es constante durante toda la ejecución del programa. (Como en nuestro proyecto)[5]

    Además un mismo emisor puede mandar datos a varios receptores y los datos no se combinan, ya que se tiene una lógica que distribuye los datos dependiendo de la caja a la que se quiera mandar.

    Los créditos o valores de umbral se pueden cambiar durante la ejecución, para esto se implementó la maquina de estados en la cual hay un estado exclusivo para cambiar los umbrales. Después es el árbitro el que se encarga de hacer cumplir el algoritmo de flow control y velar porque la marca de agua nunca sea sobrepasada.

2. Descripción arquitectónica

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

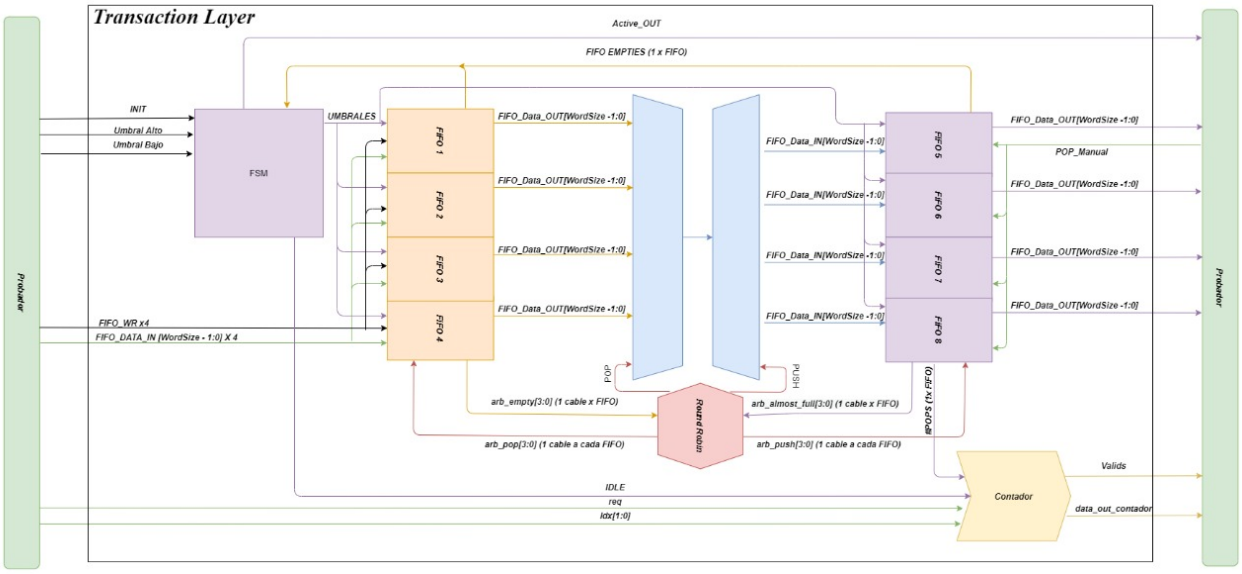

Figura 2: Conexión de capa de transacción PCIE con sus respectivas entradas y salidas

El proyecto consta de los siguientes elementos:

- FIFO: Se utilizan ocho unidades en total, las cuales se encargan del manejo de memoria en la arquitectura.
- Árbitro: Se utiliza para el manejo de los buses de memoria en los FIFOs, esto para evitar que dos de ellos traten de pasar por un cable al mismo tiempo.
- Multiplexor y demultiplexor: El multiplexor se compone por 4 entradas y una salida de datos. Su propósito es pasar las palabras de salida de cada FIFO al demultiplexor. Por otra parte, el demultiplexor está conformado por una entrada y 4 salidas de datos. Su función es la de direccionar las palabras que le pasa el multiplexor al FIFO correspondiente. El multiplexor utiliza la salida de pop del árbitro como entrada de selección, mientras que el demultiplexor escoge la salida de datos con base en los dos bits más significativos de las palabras de entrada.
- Contadores: Se utilizan para verificar que la cantidad de palabras de entrada y salida sean la misma, esto para corroborar que no se perdieron datos en el proceso.
- FSM: También llamada máquina de estados, es la encargada de indicar en cuál estado se encuentra la capa. Entre los estados se encuentran: el de reinicio (**reset**), el de inicio (**init**), el de inactividad (**idle**) y el de activo (**active**).

3. Plan de pruebas

Para el diseño completo se utilizaron las pruebas mencionadas en el documento *QoS / TCs / VCs y arbitraje en la capa de transacción PCIE* que se encuentra en Mediación Virtual del curso, cada una de las siete pruebas se aplicaró para determinar el funcionamiento correcto de la interconexión completa:

1. Saque el bloque de RESET, manteniendo el estado de INIT (señal init).
2. Modifique 2 veces los umbrales altos y bajos de los FIFOs (son el mismo umbral bajo y alto para todos). Libere la señal init.
3. Provoque un Almost Full en todos los FIFOs de salida, el árbitro no le permitirá hacerlo de forma simultánea. Desde el probador, haga la menor cantidad de POPs. Verifique que las palabras que salieron son las mismas que entraron y que salieron por la salida correcta en la prioridad correcta.
4. Provoque un Almost Full en todos los FIFOs de entrada. Luego usando POPs del probador deje todos los FIFOs vacíos. Verifique que las palabras que salieron son las mismas que entraron y que salieron por la salida correcta en la prioridad correcta.
5. Lea los contadores de palabras.
6. Envíe 16 palabras, 4 a cada FIFO de entrada, y cada set de 4 palabras por FIFO de entrada estén con destino a cada FIFO de salida (las 4x4=16 combinaciones posibles). Deje todos los FIFOs vacíos. Verifique que las palabras que salieron son las mismas que entraron y que salieron por la salida correcta en la prioridad correcta.
7. Lea los contadores de palabras y valide un aumento de 4 palabras por contador.

El plan de pruebas completo, con los módulos individuales fue entregado junto al avance 1 y además se adjunta en el Zip del proyecto.

4. Instrucciones de utilización de la simulación

A continuación se muestra el código del *Makefile*:

```
test:
        yosys synth.ys
        sed -i 's/Arb_demux_cond/Arb_demux_cond_synth/' transactionLayer_synth.v
        sed -i 's/Arb_mux_cond/Arb_mux_cond_synth/' transactionLayer_synth.v
        sed -i 's/Bloque_mxdx_cond/Bloque_mxdx_cond_synth/' transactionLayer_synth.v
        sed -i 's/contador/contador_synth/' transactionLayer_synth.v
        sed -i 's/maquina_est/maquina_est_synth/' transactionLayer_synth.v
        sed -i 's/transactionLayer/transactionLayer_synth/' transactionLayer_synth.v
        iverilog BancoPrueba.v cmos_cells.v
        ./a.out
        rm a.out
        gtkwave prueba.gtkw
```

Como se muestra anteriormente, para correr la prueba final del proyecto solo es necesario utilizar el comando *make* o *make test* para que despliegue la prueba ya personalizada en GTKWave

5. Errores e inconvenientes el proyecto

Los problemas y errores que encontramos en el trayecto del diseño e implementación del proyecto fueron:

1. El árbitro no funcionaba correctamente: durante las pruebas del árbitro este funcionó de manera correcta, pero al momento de implementarlo en conjunto con los demás componentes se tuvo el error de que bloqueaba todos los FIFOs si uno solo estaba casi lleno o vacío. Esto condujo a rediseñar la lógica del árbitro para asegurarse de que todos los FIFOs detuvieran sus procesos de escritura y lectura de la forma correcta.

2. Síntesis de latches en la FSM: al momento de sintetizar la FSM con Yosys y luego probarla, se advirtió la formación de latches cuando se manejaban los cambios en los umbrales. Esto se pudo solucionar implementando umbrales internos con valores definidos, que luego se sobrescribían en el estado INIT y en los demás estados se asignaban a los umbrales de salida.

3. Además al sintetizar la máquina de estados, el software de síntesis, le asignó valores distintos a los estados definidos en el módulo conductual. Después de verificar el diseño, se llegó a la conclusión de que esto no afecta el diseño general, es simplemente que la máquina sintetizada entiende los estados bajo diferentes valores, pero las salidas de umbrales son las mismas, sin retardos, y son exactamente iguales al conductual.

4. Síntesis de la memoria: Se tuvieron latches al sintetizar el módulo de la memoria, debido a los esfuerzos de hacer que no existieran retardos a la salida del FIFO. Para corregir esto, se tuvo que cambiar una pequeña parte del código que modela a la memoria.

5. Conseguir que los FIFOs saquen el valor al instante con la señal de POP: debido a que se quería coordinar los FIFOs con el árbitro para evitar tener que trabajar a destiempos los valores, se tuvo que modificar la estructura originalmente diseñada del FIFO. Se decidió implementar la salida de datos cada vez que hubiera un cambio en las entradas, para los demás procesos se siguió coordinando con el ciclo de reloj.

6. Resultados y análisis

Cada una de las pruebas propuestas para cada sección de la capa de transacción funcionó de manera correcta, tanto en su prueba individual como en la prueba de todos los elementos interconectados. Conforme se fue avanzando se descubrieron formas de implementar la lógica de los componentes y se fue corrigiendo para que funcionase de mejor forma. A continuación se muestra la prueba final de los elementos interconectados, dada la gran cantidad de señales, se divide en tres partes:

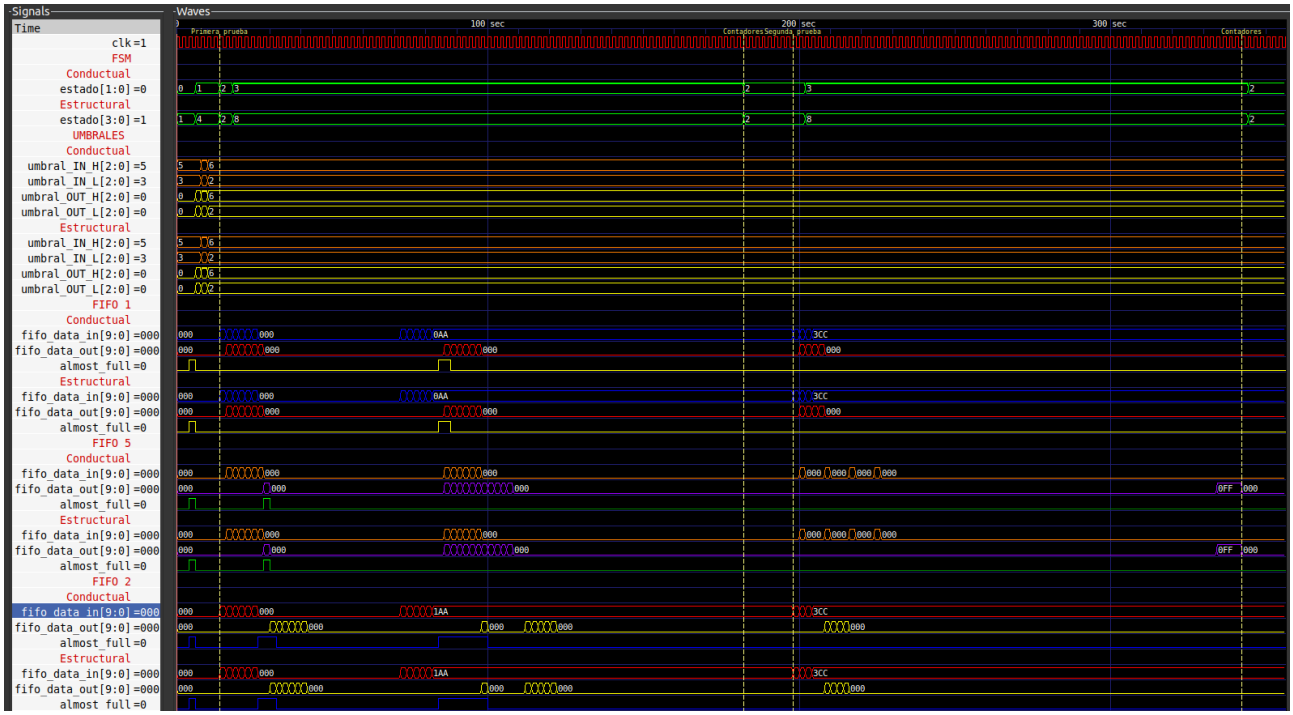

Figura 3: Salidas resultantes del sistema parte 1

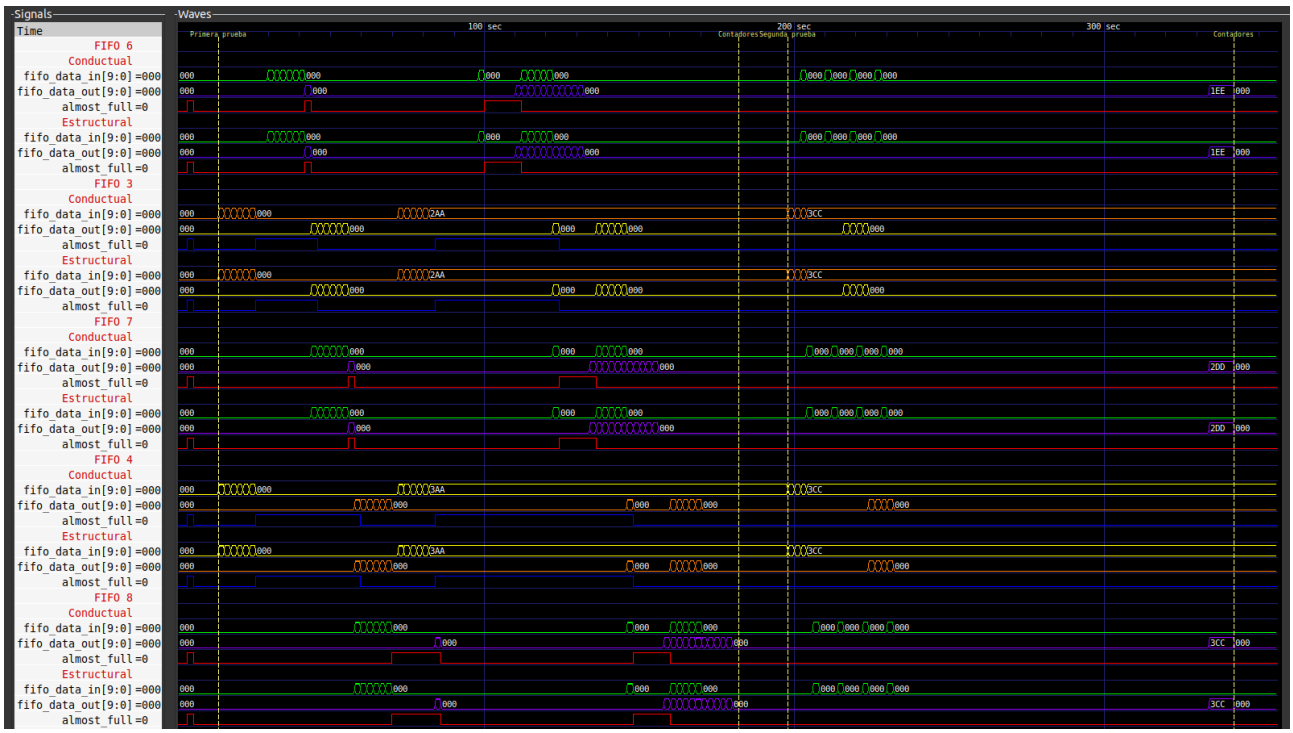

Figura 4: Salidas resultantes del sistema parte 2

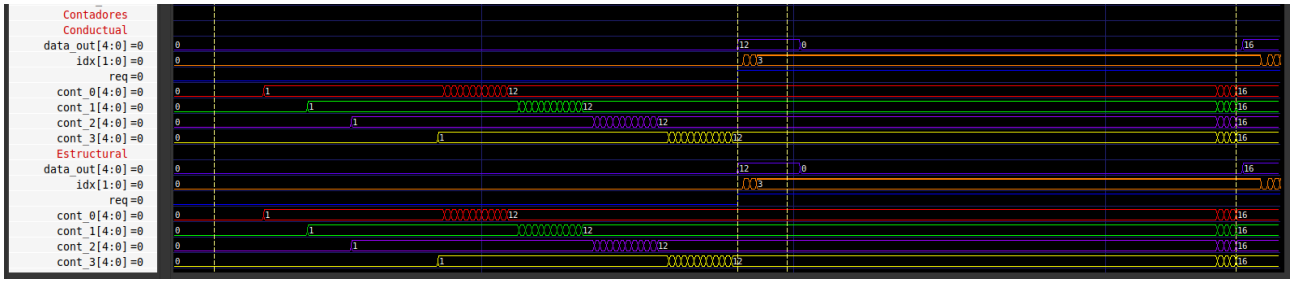

Figura 5: Salidas resultantes del sistema parte 3

Se puede observar en la imagen, la división de las pruebas explicadas en el plan de pruebas. Se inicia modificando los umbrales previo a la prueba #1. Se puede observar en la imagen como el sistema cumple correctamente lo siguiente:

- Todas las señales conductuales corresponden de manera correcta con su respectiva señal estructural. No hay ningún desfase entre ambos sistemas.
- Se observa una diferencia en el valor decimal en los estados conductual y estructural. Esto se debe, considerando el hecho de que todas las señales internas de ambos sistemas tienen exactamente la misma salida, que yosys al sintetizar, utiliza una codificación distinta, cambiando un único bit a la vez entre cada estado, para evitar que en la implementación del circuito real, se pueda repetir accidentalmente un estado al transicionar y afectar el funcionamiento del sistema. Es por esto que los estados pasan a ser: 0001, 0010, 0100, 1000. Como son 4 estados, utiliza 4 bits. Sin embargo, los estados cambian de manera correcta.

- El sistema implementa la prioridad de los FIFOs de manera correcta, limita el paso de información cuando los FIFOs de salida se encuentran casi llenos, y dirige desde cualquier FIFO de entrada, la información de manera correcta a cada FIFO de salida según el destino. Esto se puede observar tanto en la imagen 3 con las señales de almost full y en cómo se paraliza el sistema esperando un pop cuando un FIFO de salida esta casi lleno, así como en la figura 4 donde se muestran los FIFOs restantes y como (en combinación con las señales de la figura 3) todos los FIFOs de entrada se comunican correctamente con las salida durante la segunda prueba.
- Los aumentos en los contadores de salida son los esperados. Se inicia sacando 12 elementos por cada FIFO de salida, y se termina con 4 elementos más por cada FIFO, por lo tanto se espera un aumento de 16 elementos por FIFO, para un total de 64 elementos finales.

7. Conclusiones

El diseño de hardware de la capa de transacción de un PCIE represento un reto para cada uno de los participantes del proyecto. Para desarrollar este proyecto se requirió realizar varios análisis y estudios de la lógica que rige el comportamiento de todos los módulos que componen la capa de transacción. Dicho esto, la realización de este trabajo resultó muy útil para reforzar los conceptos básicos del diseño de RTL en el lenguaje Verilog. Análogamente este proyecto también sirvió para fortalecer las habilidades y conocimientos de los estudiantes en verificación de hardware, ya que se tuvo que diseñar el plan de pruebas de cada módulo desarrollado para el transaction layer.

Por otra parte, el proyecto representó una clara mejoría en comparación al proyecto anterior, esto debido a los avances logrados en la lógica y habilidad de cada uno de los integrantes en lo que es diseño de hardware con Verilog. También la solución de problemas fue más fluida al saber como resolver cada uno de los problemas típicos que ocurría, ya fuese con los módulos o la síntesis con Yosys. La mayor ventaja en cuanto al plan de trabajo fue que todo el equipo logró trabajar en conjunto y con apoyo constante de los distintos miembros. Esto ayudó a depurar juntos los distintos módulos y completar las entregas de manera funcional.

Finalmente, cabe destacar que cada módulo diseñado pasó las pruebas realizadas exitosamente, así como también lo hizo el transaction layer construido con ellos. Estos resultados demuestran que esta capa del PCIE funciona de acuerdo con las especificaciones solicitadas. Por lo tanto, se puede concluir que la capa de transacción se diseñó correctamente.

8. Plan de trabajo

Tema	Fecha	Encargado
Memoria	29 Junio	Andrés Arias Gabriel Araya
FIFO	29 Junio	Todos
Control Logic	29 Junio	Jafet Gutierrez
Write Logic	29 Junio	Luis Rodríguez
Read Logic	29 Junio	Luis Rodriguez
Síntesis total del FIFO	29 Junio	Andres Arias
Plan de Trabajo	29 Junio	Jafet Gutiérrez
Contadores	6 de Julio	Andres Arias
Arbitro	6 de Julio	Jafet Gutierrez Luis Rodríguez
Máquina de estados	6 de Julio	Andrés Arias
MUX/DEMUX	13 de Julio	Jafet Gutiérrez
Interconexión de la capa de transacción	13 de Julio	Gabriel Araya
Pruebas y Síntesis	13 de Julio	Luis Rodriguez Andres Arias
MakeFile	13 de Julio	Gabriel Araya
AUTOINST	13 de Julio	Todos
Bitácora	17 de Julio	Todos
Reporte Final	17 de Julio	Todos
Presentación	17 de Julio	Todos

Cuadro 1: Plan de Trabajo

9. Referencias

[1] A. P. Mrs. Amandeep Kaur, *AN OVERVIEW OF QUALITY OF SERVICE COMPUTER NETWORK*. peejay Institute of Management, 2011.

[2] J. L. Hennessy, D. A. Patterson y J. R. Larus, *Estructura y diseño de computadores: interficie circuiteria*. Reverté, 2000.

[3] A. Nasrallah, A. Thyagaturu, Z. Alharbi, C. Wang, X. Shao, M. Reisslein y H. Elbakoury, "Ultra-Low Latency (ULL) Networks: A Comprehensive Survey Covering the IEEE TSN Standard and Related ULL Research", mar. de 2018.

[4] *Configuring Priority Flow Control*, 5.ª ed., Cisco Networking Academy, ene. de 2018.

[5] J. P. Gasulla, "A New Credit-Based End-to-End Flow Control Protocol for High Performance Interconnects", *Universitat Politecnica De Valencia*, 2015.