Reporte Proyecto 2

Universidad de Costa Rica
Escuela de Ingeniería Eléctrica
Circuitos Digitales II
Profesor: Jorge Soto Avendaño

Capa de transacción PCIE (Adaptada) y lógica de conmutación

César Valverde Zúñiga
Douglas González Parra
Alberto Martínez Ríos

06 de julio 2019

RESUMEN

Se espera desarrollar la capa de transición de un PCIE con las secciones solicitadas en las especificaciones del proyecto -memoria, FIFO, fsm, el flow control interno junto con mux-round robin.

1. Nota Teórica.

1.1. Calidad de Servicio (QoS).

QoS o Calidad de Servicio (Quality of Service) es un término que abarca el conjunto de tecnologías que garantizan la transmisión de cierta cantidad de información en un tiempo dado. Calidad de servicio es la capacidad para administrar el tráfico de red de manera rentable y así mejorar las experiencias del usuario al recibir un servicio. Es especialmente importante para ciertas aplicaciones tales como la transmisión de video o voz.

Según la UIT (Unión Internacional de Telecomunicaciones, por sus siglas en inglés), define a la calidad de servicio, como el efecto colectivo del rendimiento de un servicio que determina el grado de satisfacción del usuario de dicho servicio. Conformado por tres grandes aspectos:

- El Usuario.
- La Clase de Servicio (CoS).
- El tipo de Servicio (ToS).

El objetivo fundamental de QoS, es proporcionar un mejor servicio a ciertos flujos de datos, haciéndolo mediante la priorización de los paquetes entre sí, dependiendo de su naturaleza y asegurando la recepción exitosa de cada paquete a su destino.

1.1.1. Calidad de servicio en redes de conmutación de paquetes.

La Calidad de servicio en redes de conmutación de paquetes, se puede decir, que consiste en el rendimiento de una red de telefonía o de computadores, donde un administrador o el mismo usuario puede ver lo que sucede con programas especializados. Para definir la calidad de los servicios se consideran varios aspectos tales como tasas de errores, ancho de banda, rendimiento, retraso en la transmisión, disponibilidad, etc. Entre los parámetros que definen la calidad de servicio están:

- Ancho de Banda, consiste en la máxima cantidad de datos que pueden pasar por un camino de comunicación en un momento dado, normalmente medido en segundos,
- Jitter, es efecto de desplazamiento de la señal o bit con la posición ideal que debería ocupar en el tiempo.
- Estado latente de la red, el nivel de tráfico a traves del tiempo, es útil especialmente en el tránsito de tráfico de tiempo real e interactivo.
- Pérdida de paquetes, paquetes de datos que son desechados cuando las colas de tráfico estan colapsadas, perdiéndose información en el camino.

Típicamente, las redes y protocolos digitales que admiten QoS (por ejemplo, Frame Relay, ATM, etc), permiten controlar algunas de las perturbaciones más comunes en comunicaciones, entre ellas:

- El Jitter o fluctuación, consiste en la llegada de una secuencia de paquetes con retardos dispares para cada uno de ellos, aspecto que perjudica en gran medida a las comunicaciones ordenadas, como por ejemplo a las secuencias de audio,
- Llegada en desorden, causada por el encaminamiento de los paquetes de una secuencia por distintos trayectos, que sólo puede ser corregido por determinados protocolos de transmisión.
- Errores en la transmisión, que provocan la corrupción de los datos o la combinación errónea de paquetes.
- La pérdida de paquetes, debido a la imposibilidad de entregarlos a un receptor que tiene un buffer (cola de entrada) lleno, lo que puede obligar a la retransmisión de los paquetes pérdidos.
- Retardos debidos a las esperas de los paquetes en distintos nodos de la red (colas) o, debido al encaminamiento a través de un camino más largo que el directo para evitar congestiones.

1.1.2. Herramientas de la calidad de servicio.

- Administración de la congestión:
  - FIFO, el primero que entra es el primero que sale.
  - Enviar a una cola priorizada.
- Administración de cola:
  - Mantener espacio libre en cola para paquetes prioritarios.

- Desechar paquetes con menor prioridad.

▶ Eficiencia del enlace:

- Mantener espacio libre en cola para paquetes prioritarios.
- Desechos de over head.

▶ Políticas y Formas de tráfico:

- Definición del ritmo de transferencia entre host.
- Desecho de paquetes del host de mayor velocidad.

1.1.3. Niveles de QoS.

La medida de la calidad de servicio en las telecomunicaciones se asocia generalmente a la satisfacción del cliente, a la percepción que éste tiene del servicio que se le presta. Entre las medidas habituales de calidad proporcionada se incluyen cuestiones como la disponibilidad de las redes, los tiempos que se tarda en realizar la comunicación o la velocidad y la tasa de errores en la descarga de un archivo con una conexión a Internet comercial. Existen tres niveles de calidad de servicio. Estos pueden ser elegidos según los intereses de los usuarios. Los niveles son:

▶ Servicio Mejor Esfuerzo (Best-effort service).

▶ Servicio Diferenciado o QoS suave.

▶ Servicio Garantizado o QoS fuerte.

1.1.4. Herramientas de administración del tráfico.

Entre las herraminetas de administración de tráfico más utilizadas están:

▶ Encolamiento FIFO.

- No requiere configuración por lo que es la opción de encolamiento más simple.

▶ Encolamiento Priorizado .

- Asegura que los tráficos importantes tengan un rápido manejo en cualquier punto.
- Clasifica los paquetes y los agrupa en cuatro niveles de prioridad, aquellos que no tienen ningun tipo de prioridad se van a una cola normal (FIFO).
- Puede manejar distintos protocolos.

▶ Encolamiento típico: Ancho de Banda Garantizado.

- Se tiene un ancho de banda garantizado para cada usuario.
- El ancho de banda sería compartido en forma proporcional por el número de usuarios.
- Maneja el trafico asignando un monto especifico de espacio de cola para cada tipo de paquete.

- Flujo basado en WFQ (Weighted Fair Queuing).
  - Espera Equitativa Ponderada, cada flujo de datos tiene una cola FIFO separada.
  - Envia paquetes de distintos tamaño a distintas tasas cada uno, de manera que el envio de datos sea de forma justa.
- Clase Basada en WFQ.
  - Separa los paquetes con más alta prioridad que se estan enviando, proporcionandoles un ancho de banda especifico, garantizando su envío satisfactorio.
  - El resto de los paquetes son enviados mediante WFQ.

1.2. Arbitraje en sistemas digitales.

La administración de los datos en los sistemas digitales esta compuesta por varias fases entre ellas el arbitraje, este está definido como la función que realiza el árbitro, el cual es componente que resuelve conflictos cuando 2 o más maestros quieren usar el bus simultáneamente. Cuando un maestro quiere usar el bus levanta una solicitud al árbitro, y éste concede el control del bus en base a criterios de prioridad o equidad.

1.2.1. Arbitraje centralizado.

En esquemas de arbitración centralizada, utilizan un único árbitro para seleccionar al próximo maestro del . Típicamente se usan las señales de arbitración request (solicitud), grant (conceder) y busy (ocupado). Estas señales pueden ser compartidas por los potenciales maestros (esquema daisy-chain) o pueden ser independientes para cada maestro.

1.2.2. Arbitraje descentralizado.

En el arbitraje descentralizado cada uno de los dispositivos involucrados seleccionan al próximo maestro del bus. A cada dispositivo del bus se le asigna un número de identificación. Cada dispositivo compara el código formado en la línea de arbitraje con su propia ID, comenzando desde el bit más significativo. Si encuentra la diferencia en cualquier posición de bit, desactiva sus unidades en esa posición de bit y para todos los bits de orden inferior. Este arbitraje ofrece alta confiabilidad porque el funcionamiento del bus no depende de ningún dispositivo individual.

1.3. Priority Flow Control.

Cada puerto al final de un enlace en PCI Express debe implementar el control de flujo. Es esencialmente un método para comunicar el estado del búfer de recepción de un receptor a un transmisor, proporcionando formas de evitar el desbordamiento del búfer, pero también el poco uso del búfer . El control de flujo es punto a punto y no de extremo a extremo.

1.4. Relación de los créditos o pesos con el Flow Control.

Se puede utilizar los créditos en un diseño, al implementar el flow control al utilizar canales virtuales. El sistema de flow control establece cuantos paquetes de datos puede recibir de acuerdo al espacio libre. El sistema de flow control le informa a las lineas de transmisición si pueden seguir enviando paquetes o deben pausar de acuerdo al nivel de peso o prioridad que posean. [**credits**]

2. DESCRIPCIÓN ARQUITECTÓNICA

2.1. Memoria.

Se emplea una memoria con un puerto de entrada y un puerto de salida (se debe poder leer y escribir al mismo tiempo).

Entradas:

- Clk, señal de reloj.
- iWriteEnable, señal de activación de lectura o escritura.
- iReadAddress0, puntero de lectura.
- iWriteAddress, puntero de escritura.
- iDataIn, datos de ingreso.

Salidas:

- oDataOut0, datos de salida.

2.2. FIFO.

El fifo emplea la memoria anteriormente descrita. Se debe evitar leer el fifo cuando se encuentre vacio o escribir cuando se encuntre lleno.

Entradas:

- Posee una señal de reinicio.
- Señal de indicación de entrada de datos (Push).
- Posee una señal de indicación de salida de datos (Pop).
- Señal de datos de entrada al FIFO (Fifo_Data_in)

Salidas:

- Señal de datos de salida del FIFO (Fifo_Data_out)
- Señal de fifo vacio (FIFO_Empty).
- Señal de fifo casi vacio (Almost_Empty).
- Señal de fifo casi lleno (Almost_Full).
- Señal de fifo lleno (FIFO_Full).
- Señal de pausa en el FIFO (Pausa).
- Señal de error en el FIFO (Error_Fifo).
- Señal indicadora del valor umbral (Error_Fifo).

2.3. Flow control

El Flow Control envía señales al sistema FIFO para indicarle cuando puede ingresar datos y cuando no debe hacerlo, pausando o permitiendo la entrada de datos en el stack de los FIFO.

Entradas (vienen de cada uno de los FIFO al cual pernecen):

- Señal de fifo vacio (FIFO_Empty).
- Señal de fifo lleno (FIFO_Full).

Salidas:

- empty_main_FIFO
- pause_main_FIFO
- pause_vc0
- pause_vc1
- empty_vc0
- empty_vc1
- pause_DO
- pause_D1

2.4. Mux Round Robin.

Es el encargado de la tabla de arbitraje (Arbitro),donde se analizan los pesos de cada solicitud, en nuestro caso dar prioridad a los datos del canal VCO sobre los datos del canal VC1.

Por el diseño de la lógica, VC0 y VC1 no realizan pop simultàneos, es decir, sacar datos, si uno realiza pop, el otro espera por lo tanto debe esperar.

Entradas:

- data_in_VC0.
- data_in_VC1.
- reset.
- clk.

Salidas:

- valid_in_VC0.
- valid_in_VC1.
- dataout.
- valid_out.

2.5. Máquina de Estados de Control.

Se diseña una máquina de estados finita, sincrónica con las siguientes especificaciones:

- RESET: Estado de reset, cambia a INIT.
- INIT: Forzado mediante señal “init”, permite la modificación de registros “Umbrales”. Precedencia sobre IDLE, cambia a IDLE.
- IDLE: Todos los FIFOs están vacíos. Salida “idle” en 1 sólo en este estado. Cambia a ACTIVE al tener un FIFO no vacío.
- ACTIVE: Modo de transmisión de datos por defecto.
- ERROR: Escritura y no lectura en uno o más FIFOs cuando están llenos (señal de error). Indicar el ID en .^{er}ror_full”. Sale hacia RESET únicamente al aplicar reset.

Entradas:

- Clk.
- reset_L.
- init.
- FIFO_error.
- FIFO_empy.
- Umbral_MF.
- Umbral_VC
- Umbral_D.

Salidas:

- active_out.
- idle_out.
- error_out.
- Umbrales_I.

2.6. Interconexión completa.

En el Apéndice 2, se define el diseño estructural a seguir para la interconexión de todos los módulos, realizando los enlaces entre ellos, mediante elementos de control como lo son los demuxes y muxes complementarios al diseño y que integran las diferentes secciones de la capa de transacción .

3. Plan de pruebas

3.1. Memoria

Se diseña un DUT, el cual contiene un probador que sera aplicado tanto al modelo conductual como al modelo estructural. El probador enviara señales que estimulen las entradas con la intención de obtener los valores esperados en las salidas. Se realizara la verificación del correcto funcionamiento del modelo conductual al analizar las señales de salida y entrada empleando GTKWave. Se comparará el comportamiento del modelo estructural con el conductual, ambos modelos deben mantener el mismo funcionamiento. La memoria debe permitir la escritura y lectura de datos de forma simultanea.

3.2. FIFO

Se diseña un DUT, el cual contiene un probador que sera aplicado tanto al modelo conductual como al modelo estructural. El probador enviara señales que estimulen las entradas con la intención de obtener los valores esperados en las salidas. Se realizara la verificación del correpto funcionamiento del modelo conductual al analizar las señales de salida y entrada empleando GTKWave. Se comparará el comportamiento del modelo estructural con el conductual, ambos modelos deben mantener el mismo funcionamiento.

Entre las pruebas contenidas en el probador se debe verificar el correptofuncionamiento de las intrucciones Push y Pop. No se realizan lecturas en el fifo estando vacío. Tampoco se debe intentar escribir en el FIFO mientras se encuentre lleno. Además que se esta cumpliendo el principio de un FIFO, donde los primeros datos en ingresar son los primeros en salir y no se realiza la perdida de ningún valor.

3.3. Flow Control

Para el Flow Control, el probador enviará las señales de entrada FIFO empty y FIFO full. Se iniciará todo como FIFO empty y sin pausar para generar un tráfico inicial como si fuese la primera utilización de dicha implementación. Luego se requiere el escenario en donde Main FIFO está vacío y que FIFO VC0 o FIFO VC1 no estén pausados para demostrar que no se hace POP en ese caso.

Para el caso del POP delay se necesita que solo uno o ninguno de los FIFO D0 y D1 estén activos y que un VC0 esté lleno y el otro vacío y alternar estos casos para generar cambios en el POP delay que funciona de selector del MUX que recibe los datos de VCO y VC1.

3.4. Mux-Round Robin

El round robin funciona como el arbitro y distribuye las señales por los diferentes canales. Se debe verificar que cumple con la distribución de acuerdo a los pesos y el FIFO destino correpto. Comprobar que los paquetes con id = 0, son transferidas a la línea cero. Verificar que las paquetes con id = 1, son transferidas a la línea uno. Se diseña un DUT, con un probador que estimula la lógica conductual y estructural y se verifica que ambas tienen un comportamiento correpto.

3.5. FSM

Se define un circuito conductual, donde el probador contiene las pruebas necesarias para considerar todos los estados de la máquina, que seran aplicadas tanto al modelo conductual como al

modelo estructural. Desde el probador se envian las señales, estimulando las entradas con la intención de obtener los valores esperados en las salidas. Se realiza la verificación del funcionamiento correcto al analizar las señales de salida y entrada empleando GTKWave. Se obtiene que tanto el modelo conductual y el modelo estructural se comporten de manera homóloga.

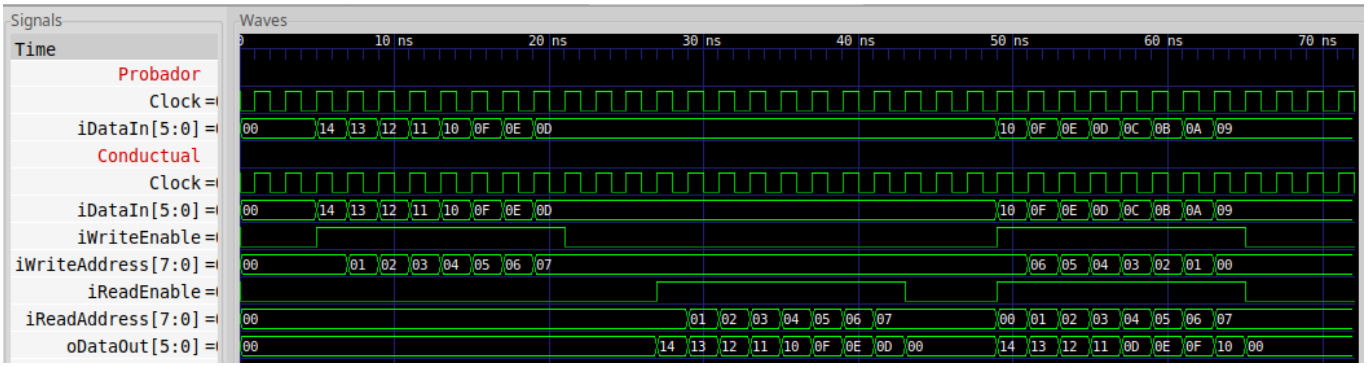
4. EJEMPLOS DE LOS RESULTADOS

4.1. Memoria.

4.1.1. Descripción Conductual.

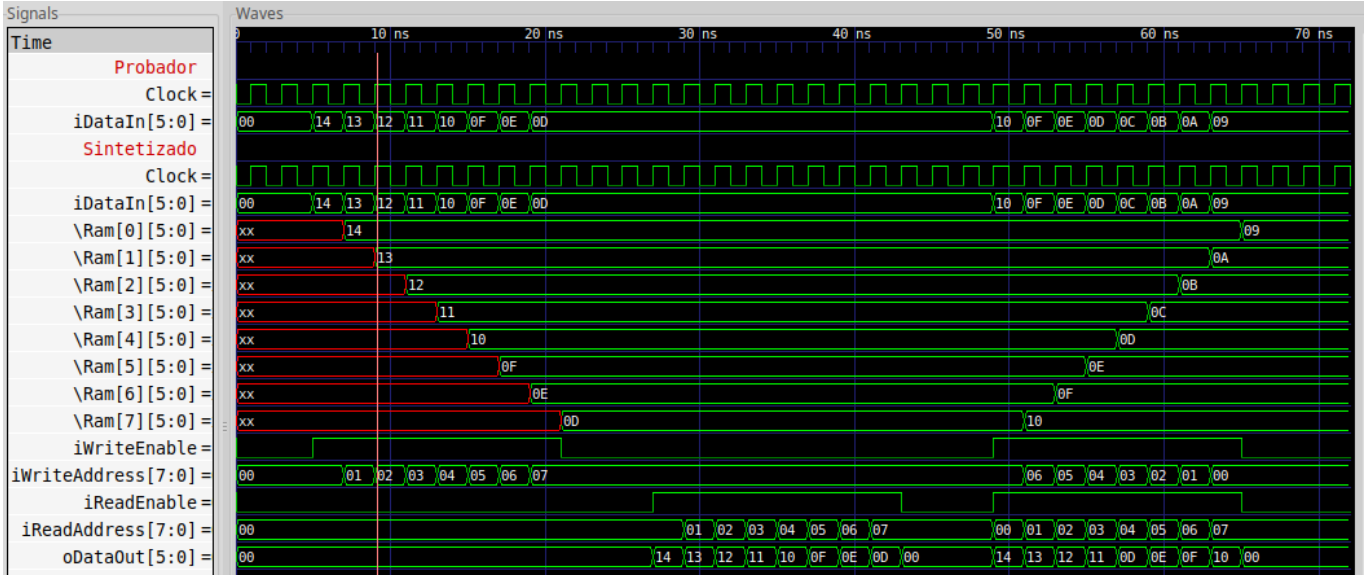
Según lo propuesto en la sección 3.1 se realizan las pruebas de la memoria la cual tiene una señal de entrada y otra de salida de datos, un habilitador de escritura de datos en memoria y de lectura, punteros de lectura y escritura. Las pruebas mostradas en las figuras Figura 1, Figura 2 y Figura 3 consisten en primeramente escribir y leer de forma separada y luego de forma simultánea, mostrando el flujo correcto mediante la visualización de espacios de memoria asignados por la acción de entrada y salida de datos.

Figura 1. Simulación de Memoria (Condutual)



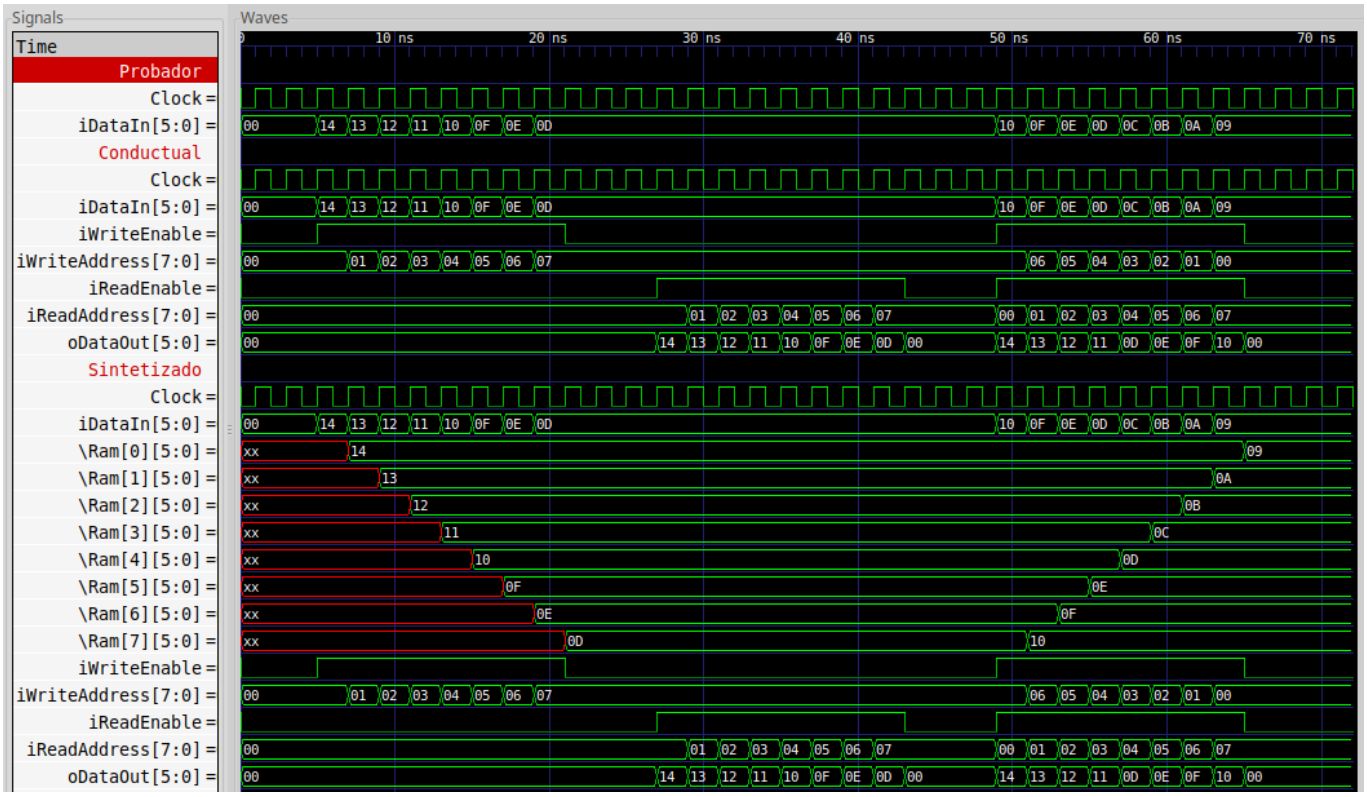
4.1.2. Descripción Estructural.

Figura 2. Simulación de Memoria (Estructural)



4.1.3. Funcionamiento de ambas descripciones.

Figura 3. Simulación de Final de la Memoria



4.2. FIFO.

Según lo propuesto en la sección 3.2 se realizan las pruebas del FIFO, del cual primeramente de realizan las acciones push y pop por separado llenando la memoria sin rebasarla, seguidamente se realizan las funciones push y pop de forma simultánea y finalmente se prueban las condiciones de error tanto para un push en el momento que se tiene memoria llena y un pop cuando se tiene memoria vacía. A la vez se muestran las señales de FIFO empty, Almost Empty, Almost Full, Pausa, FIFO Full y Error Fifo, que varía su activación dependiendo la cantidad de espacio de memoria manejada por el FIFO. En el caso de una Main FIFO, FIFO D0 y FIFO D1 se maneja un dato en memoria como Almost Empty y 3 espacios llenos de memoria como Almost Full, como se muestra en las figuras Figura 4, Figura 6 y Figura 8. En el caso de los FIFO correspondientes a VC0 y VC1, como se muestra en las figuras Figura 5, Figura 7 y Figura 9 se tiene que para 4 datos de memoria o menos es Almost Empty y 12 datos de memoria en adelante activan la señal de Almost Full.

4.2.1. Descripción Conductual.

Figura 4. Simulación de FIFO 4 espacios(Condutual)

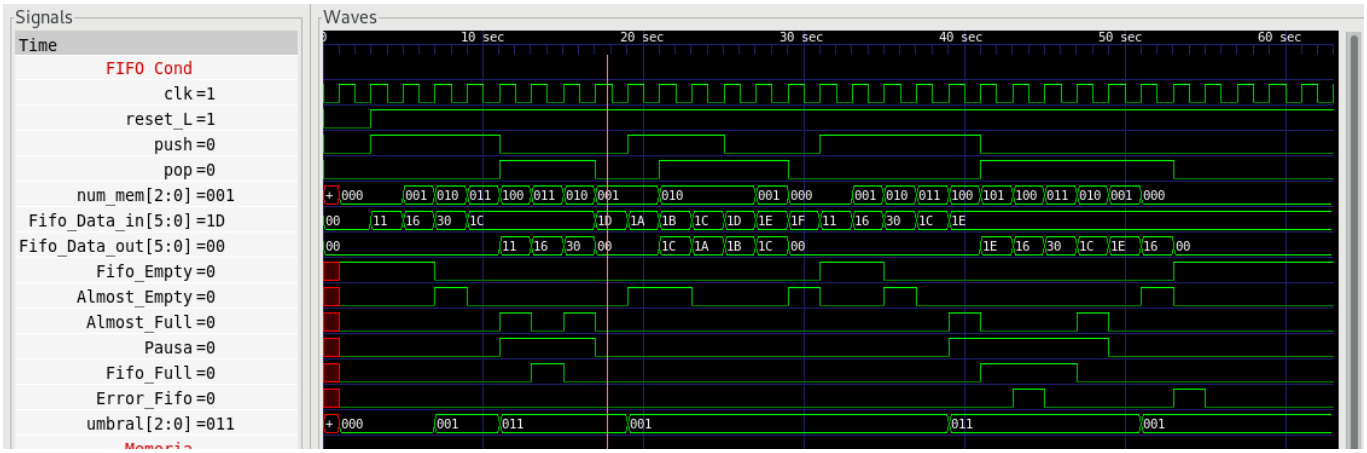
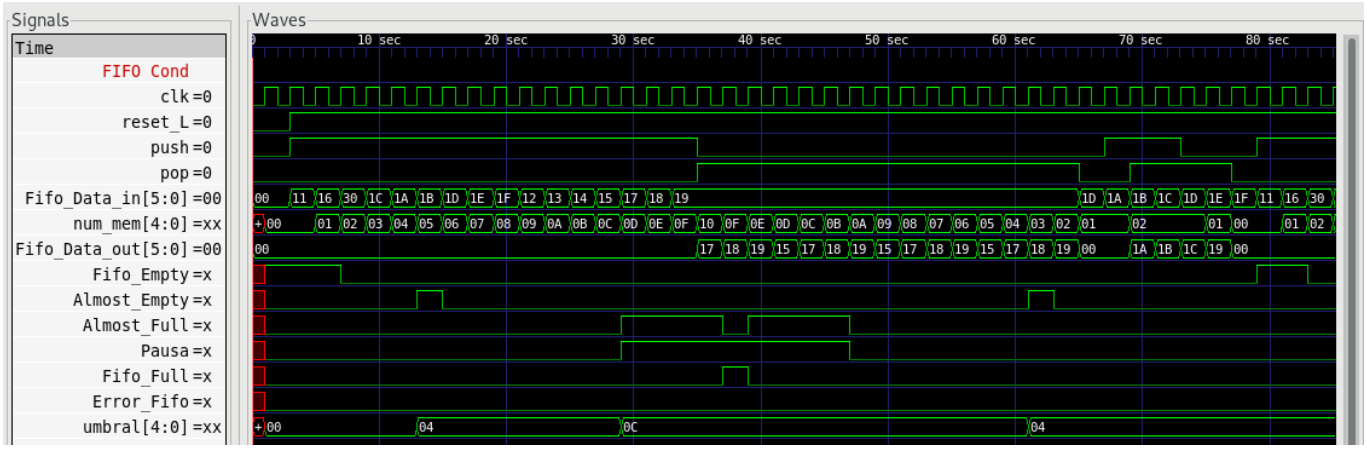


Figura 5. Simulación de FIFO 16 espacios(Condutual)



4.2.2. Descripción Estructural.

Figura 6. Simulación de FIFO 4 espacios (Estructual)

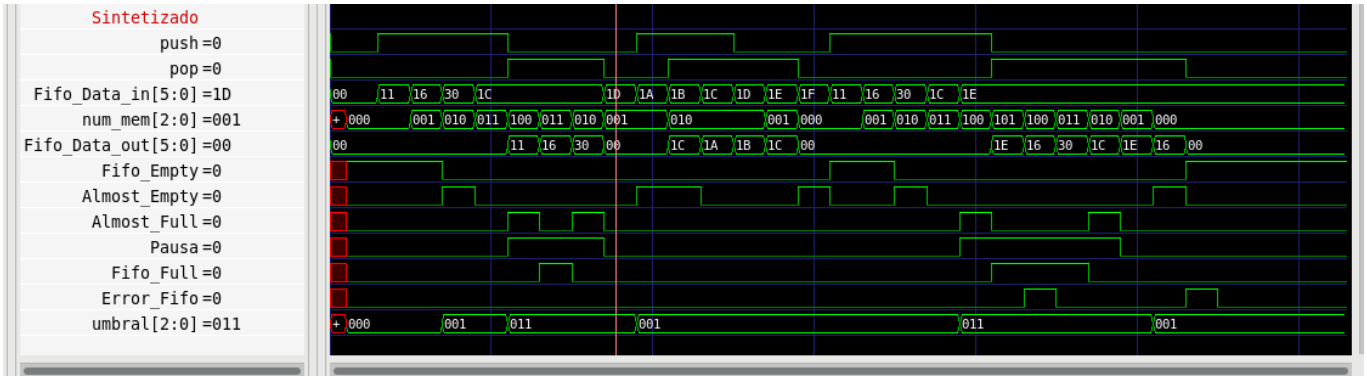
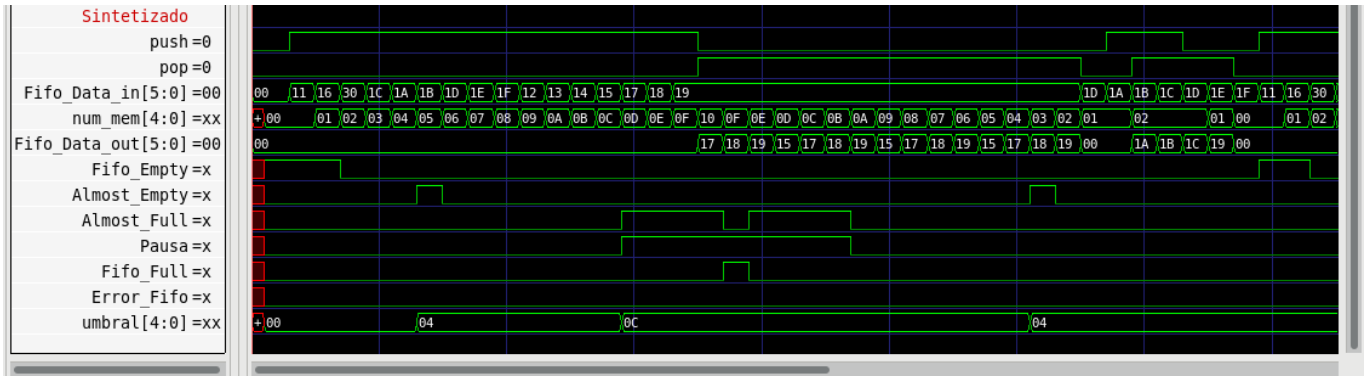


Figura 7. Simulación de FIFO 16 espacios (Estructual)



4.2.3. Funcionamiento de ambas descripciones.

Figura 8. Simulación final del FIFO 4 espacios

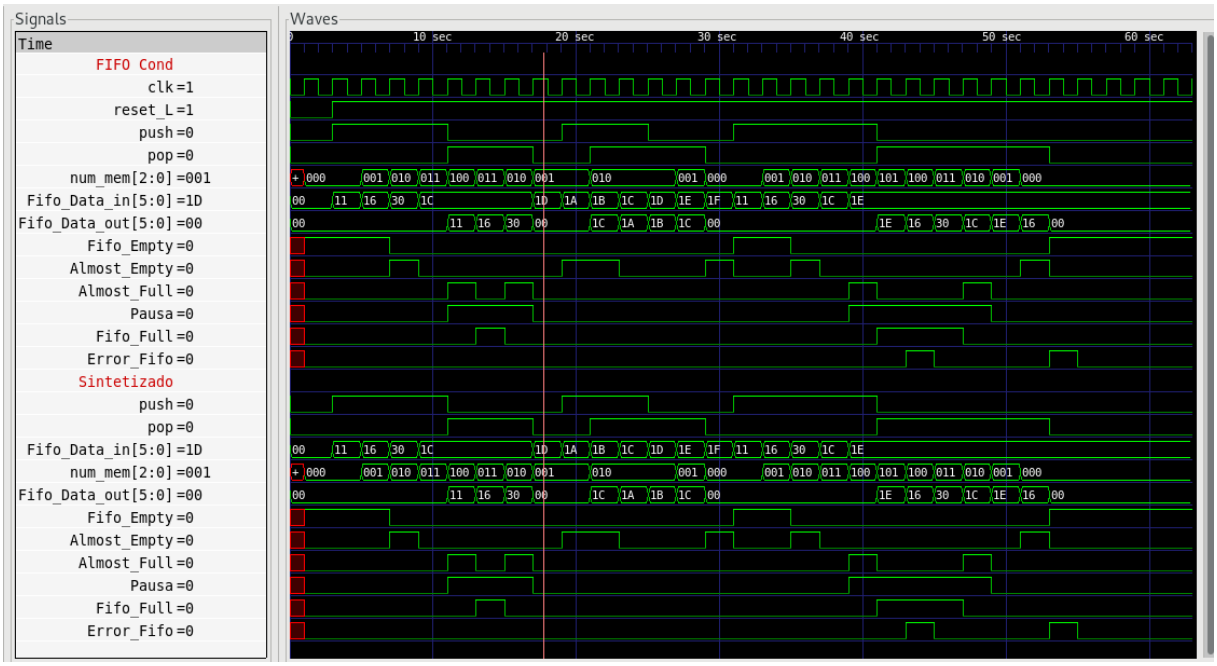
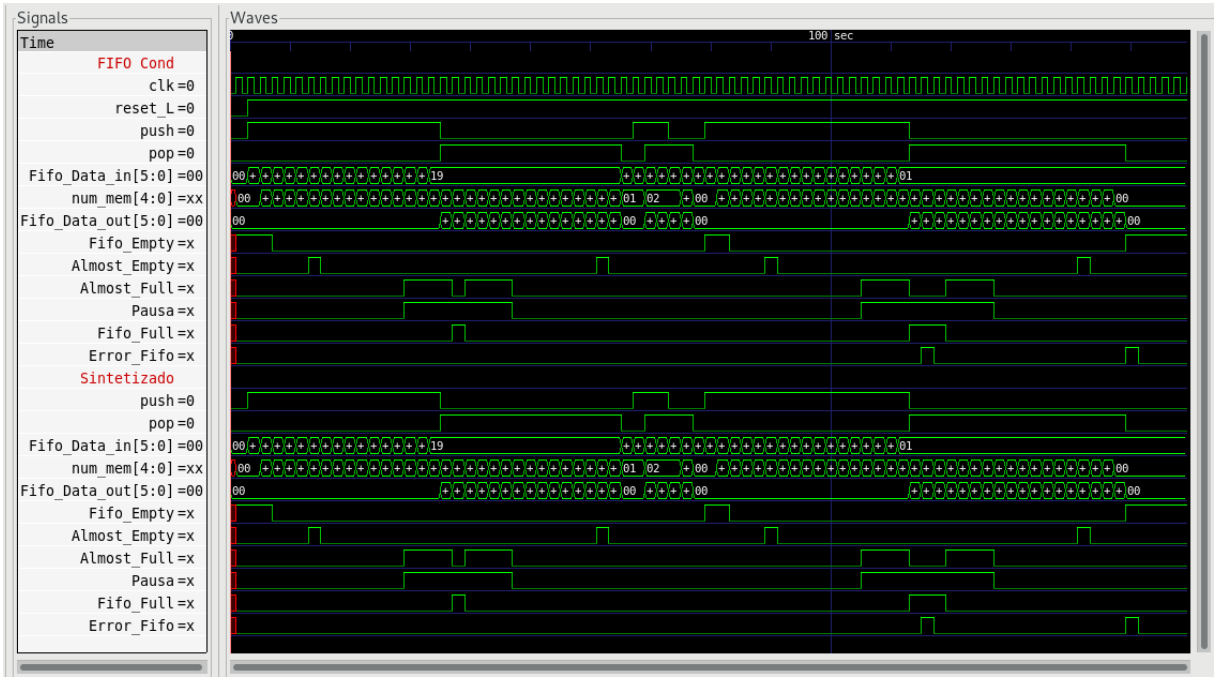


Figura 9. Simulación final del FIFO 16 espacios



4.3. Flow control.

Cada fifo contiene una pequeña parte de flow control, que determina los estados de umbral y los transfiere a la máquina de estados que controla el diseño.

4.4. Máquina de Estados.

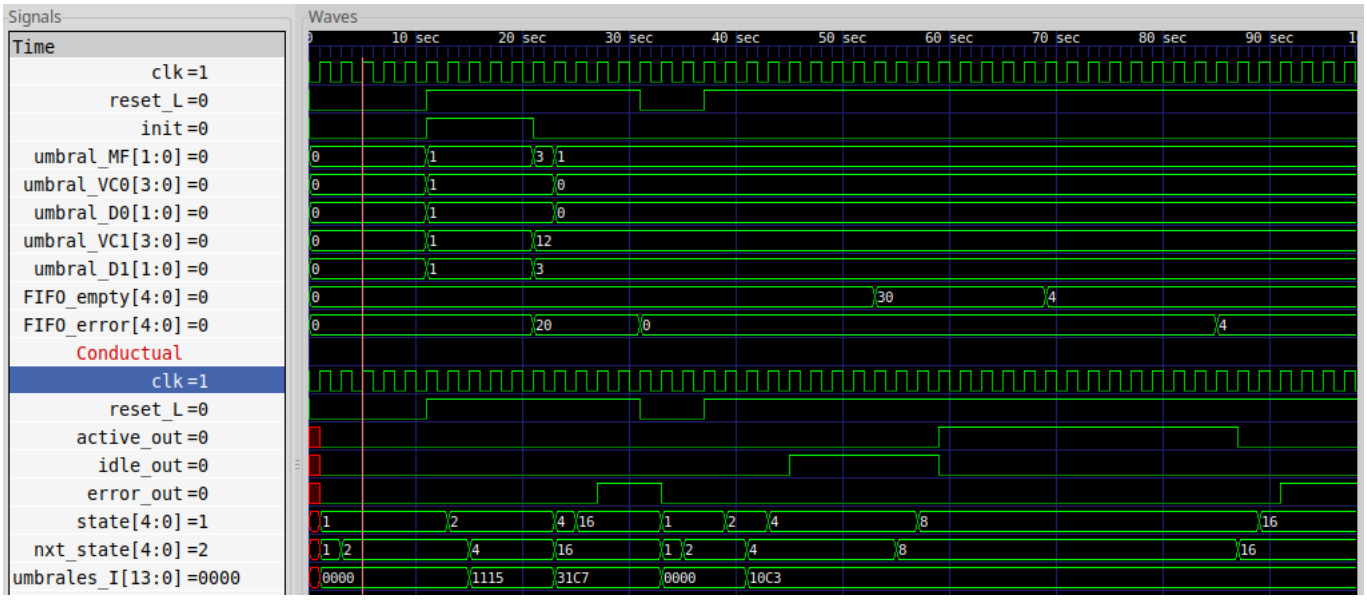
4.4.1. Descripción Conductual.

La máquina de estados es la unidad de control de la capa de transación. En reset en bajo todo el diseño se reinicia y establece los valores por defecto de arranque. La máquina inicia en estado reset, para luego con la señal de init establecer el estado de init. Después de init el siguiente estado por defecto es idle. Si no se ha producido ninguna señal de error, el siguiente estado es el de active, donde la capa completa debe recibir y transmitir los datos.

En la Figura 10 se evidencia que el comportamiento de la descripción conductual de la máquina de estados, en esta se puede ver como en reset en bajo, los valores de umbral son cero y la máquina se encuentra en estado 1, el cuál es la codificación para reset. La codificación para init es 2, el estado idle se representa con 4, el estado active con 8 y error con la codificación 16. Se aprecia en la simulación como al recibir la señal de FIFO_error después de las 20 unidades de tiempo de la simulación se activa la señal de error_out y la máquina de estados ingresa al estado de error. Al ingresar a error solo una señal de reset en bajo reinicia el sistema.

Se puede apreciar como al reiniciar y llegar al estado active, la señal de salida idle_out cambia a cero y la de active_out indica que se transfieren archivos, pasando a 1.

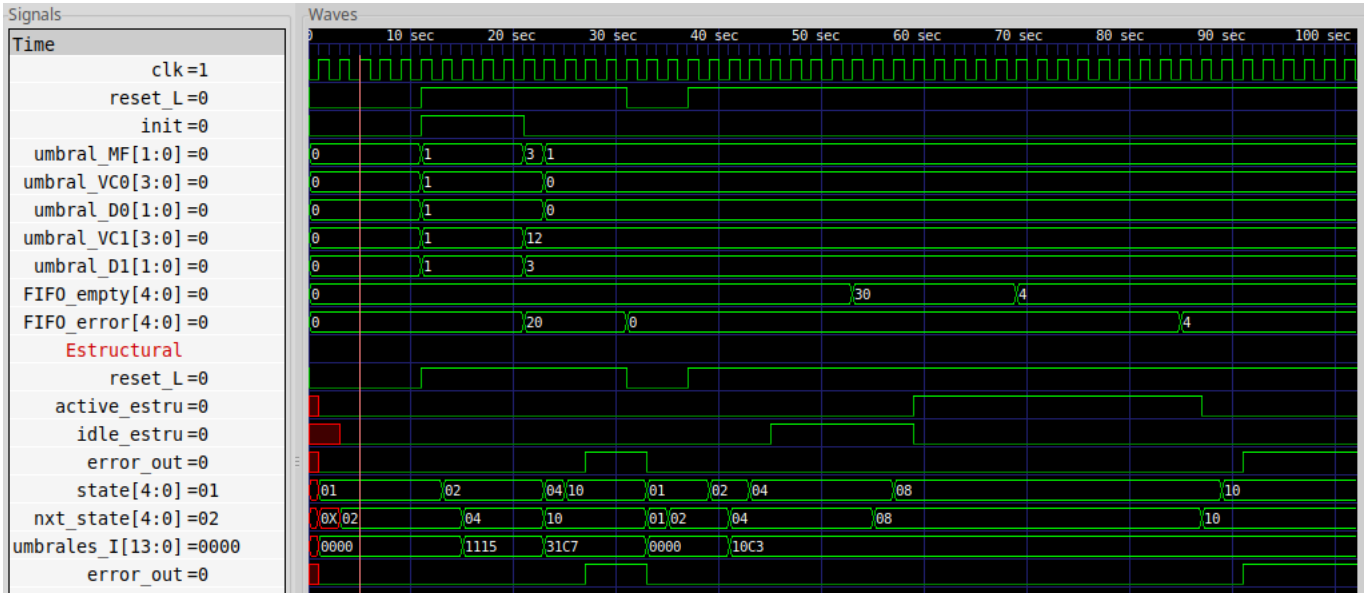
Figura 10. Simulación de la FSM (Condutual)



4.4.2. Descripción Estructural.

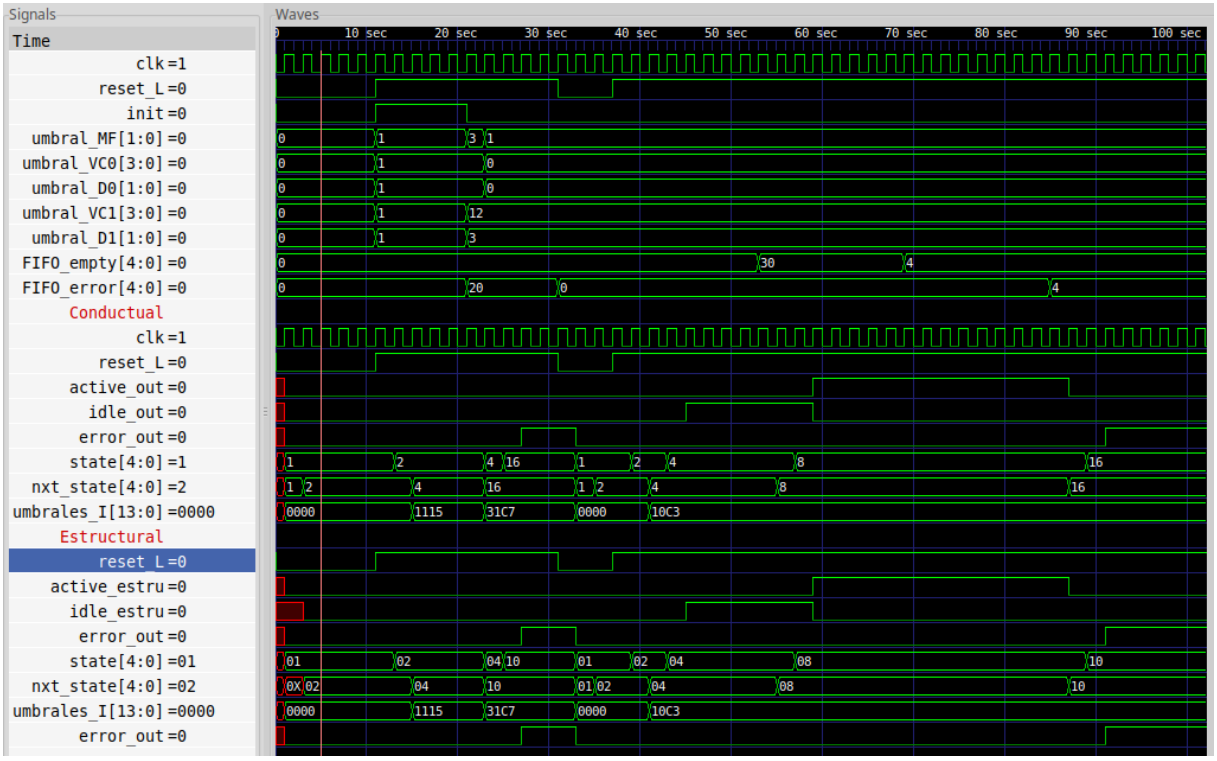
La descripción estructural obtenida de la síntesis elabora por yosys, de la FSM presenta un comportamiento igual al de la conductual, por ello se toman como diseños equivalentes.

Figura 11. Simulación de la FSM (Estructural)



4.4.3. Funcionamiento de ambas descripciones.

Figura 12. Simulación de la FSM Final

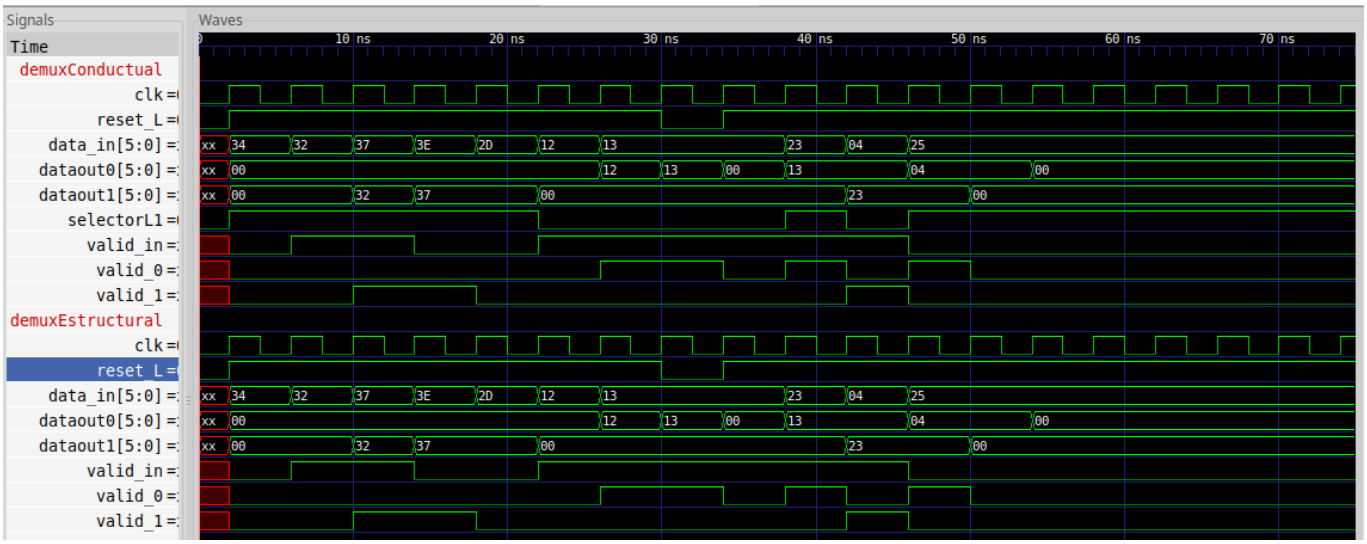


4.5. Interconexión de módulos.

4.5.1. Descripción Conductual del DEMUX de pop válidos según vc_id .

Como parte de la lógica complementaria para lograr la interconexión de la capa de transferencia se elabora un demultiplexor que se encontrara entre el modulo del Fifo Main y los Fifos VC0 y VC1. La idea principal de este demux es funcionar como Demux de pop válidos según vc_id de datos del main fifo hacia los canales VC0 y VC1. Esto se determina al leer el bit 5 del paquete de datos de entrada. El demux debe recibir una señal de la lógica interna del modulo top, que le indica cuales paquetes son validos y seleccionar por cual canal enviarlo. En la imagen Figura 13 se puede apreciar una simulación donde se analiza el funcionamiento de este demux tanto en su versión estructural síntetizada como en la descripción conducutal.

Figura 13. Demux de pop válidos según vc_id



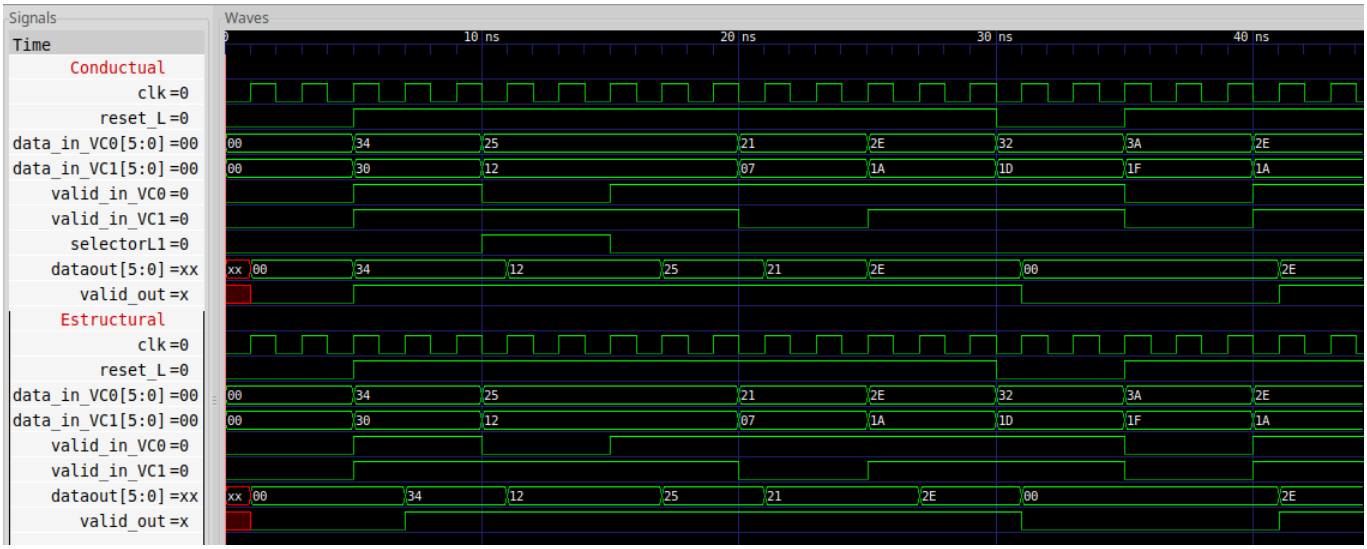
4.6. Mux Round Robin

Después de la etapa del Fifo VC0 y el Fifo VC1, se encuntra un Mux, el cual funciona como un tipo de arbitro round robin, el cual debe seleccionar cual canal transmite sus datos. Dicho mux debe dar prioridad a los datos del canal VC0.

En la figura Figura 14 se aprecia la descripción conductual y estructural síntetizada de esta parte del diseño.

Por el diseño de la lógica, VC0 y VC1 no realizan pop (sacar datos) de forma simultanea, si uno de ellos realiza pop, el otro espera.

Figura 14. Mux arbitro de paquetes entre VC0 y VC1.



4.7. Demux posterior al Mux Round Robin

Se emplea un selector de canal de destino que funciona similar a un demux, el cual en su diseño es muy similar al Demux de pop válidos según vc_id, con la diferencia que este lee el 4 bit para determinar si el paquete debe ser transmitido por el canal D0 o el canal D1.

5. UTILIZACIÓN DEL CÓDIGO

El código fuente del del proyecto se en encuentra en GitHub, el enlace para poder clonar el diseño es *https://github.com/Cesartarantula/Proyecto2*.

Para la ejecución por separado se tiene un make para cada una de las etapas, se procede de la siguiente manera:

- Tener la carpeta **Proyect2** suministrada y ubicarse en ella.
- Ya ubicado en la raíz escoja la carpeta que desee comprobar (Memoria, FSM, FIFO, etc) y escriba uno de los siguientes tres comandos:
    - make conductual (muestra la simulación del código conductual)
    - make estructural (muestra la simulación del código obtenido de YOSYS)
    - make clean (elimina archivos auxiliares)

Los nombres de los comandos son descriptivos de su funcionamiento, además para cada etapa se incluyó el archivo de YOSYS por si se desea verificar que efectivamente el código de verilog sintetiza correctamente (puede ejecutar la prueba estructural directamente despúes de sintetizar en YOSYS)

6. CONCLUSIONES Y RECOMENDACIONES

- Un FIFO cumple el principio de una fila, donde el primer paquete de datos en ingresar es el primero en salir o ser transmitido.
- Un round robin funciona como un arbitro o switch de selección, el cual posee pesos, los cuales determinan la prioridad de transmisición.
- Una forma de activar señales o transferir datos que están desfasados, de forma simultanea es es colacando flip flops que permitan retrasar la señal que adelanta al mismo punto que la que transmite más retrasada.

Referencias

[1] QoS-TCs-VCs y arbitraje en la capa de transacción PCIE (Adaptación de la arquitectura para el proyecto de diseño 2); Jorge Soto Avendaño, UCR, 2019.

[2] "The Verilog Hardware Description Language"; 5ta Edición; Donald E. Thomas; Philip R. Moorby; Kluwer Academic Publishers

[3] "Digital Design, Principles & Practices"; John F. Wakerly, 3era Edición, Prentice Hall

[4] Wikipedia QoS *https://en.wikipedia.org/wiki/Quality_of_service* Recuperado el 18-6-19.

[5] Wikipedia FPGAs *http://es.wikipedia.org/wiki/FPGA* Recuperado el 18-6-19.

[6] Cisco Systems, Inc.: Internetworking Technology Handbook. *http://www.cisco.com* Recuperado el 18-6-19.

[7] Manolis Katevenis. (1997) Buffer Requirements of Credit-Based Flow Controlwhen a Minimum Draining Rate is Guaranteed. *http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.74.6956rep=rep1type=pdf* Recuperado 19 - 06 - 19.

7. Apéndice A Cronograma del plan de trabajo

Figura 15. Cronograma del plan de trabajo.

	Fecha	Tarea	Responsable
Avance 1	13/6/19 – 19/6/19	Plan de trabajo	César Valverde
	13/6/19 – 19/6/19	FSM (Máquina de Estados)	César Valverde
	13/6/19 – 19/6/19	Memoria	Douglas Gonzalez
	19/06/2019	Reunión semanal	Todos
Avance 2	20/06/19 – 26/06/19	FIFO	Alberto Martínez
	20/06/19 – 26/06/19	Optimización de FSM	César Valverde
	20/06/19 – 26/06/19	Optimización Memoria	Douglas Gonzalez
	26/06/2019	Reunión semanal	Todos
Avance 3	27/06/18 – 03/06/19	Lógica miscelánea e interconexión completa	Todos
	07/03/19	Reunión semanal	Todos
Avance 4	04/07/19 – 06/07/19	Reporte del Proyecto	Todos
	06/07/19 – 09/07/19	Presentación	Todos
	07/10/19	Presentación en clase	Todos

8. Apéndice 2 Arquitectura proyecto de diseño #2

Figura 16. PCIE-Capa de Transacción Adaptada