

Universidad de Costa Rica

Facultad de Ingeniería
Escuela de Ingeniería Eléctrica
IE-0523 Circuitos Digitales 2
II ciclo 2018

Proyecto 1
Diseño capa PHY de una interfaz PCIe

Esteban Vargas, B16998
Yasser Wagon, B47732
Paula Góchez, B53375

Profesor: Jorge Soto

13 de octubre de 2018

Índice

| | |
|-------------------------------------|----------|
| 1. Diseño | 3 |
| 1.1. Generador de relojes | 3 |
| 1.2. Lógica muxes | 4 |
| 1.3. Paralelo - 2b serial | 6 |
| 1.4. 2b Serial a Paralelo | 7 |
| 1.5. Lógica demuxes | 9 |

Índice de figuras

| | |
|--|---|
| 1. Señales generador de relojes [Autoría Propia] | 3 |
| 2. Arquitectura de la lógica de muxes [Creación propia] | 4 |
| 3. Señales lógica de muxes prueba 1 [Creación propia] | 5 |
| 4. Señales lógica de muxes prueba final [Creación propia] | 5 |
| 5. Convertidor paralelo a 2b serial [autoría propia] | 6 |
| 6. Comprobación funcionamiento Paralelo - 2b Serial [Autoría Propia] | 7 |
| 7. Convertidor 2b serial a paralelo a [autoría propia] | 8 |
| 8. Señales convertidor 2b serial a paralelo, conductual y estructural [autoría propia] . . . | 9 |
| 9. Señales conductual y estructural lógica demuxes [autoría propia] | 9 |

Índice de tablas

| | |
|------------------------------|---|
| 1. Plan de trabajo | 3 |
|------------------------------|---|

Plan de trabajo

| Parte del diseño | Pruebas a realizar |
|----------------------|---|
| Generador de relojes | Observar la respuesta del generador a una frecuencia de 16f |
| Lógica muxes | Comprobar salida ante entradas con valores válidos en las 4 entradas, en 3 , en 2 y con solo 1 válida |
| Lógica demuxes | Comparar sus salidas ante el ingreso de una secuencia de 2 válidos, 1 válido y 0 |
| Paralelo a 2b Serial | Colocar un valor válido en la entrada y comprobar el envío de código correcto |
| 2b Serial a Paralelo | Recibir el código BC 4 veces y después el dato enviado en la etapa anterior |
| Módulo phy_tx.v | Comprobar la congruencia de la salida del módulo con la secuencia que se ingresa |
| Módulo phy_rx.v | Corrobar la salida del módulo con la secuencia ingresada en la etapa anterior |
| Módulo phy.v | Al integrarse todo |

Tabla 1: Plan de trabajo

1. Diseño

1.1. Generador de relojes

Como primer paso del diseño se creó una descripción conductual de un generador de señales de reloj, el cual recibe una señal de reloj con una frecuencia de 16f (*clk16f*) y mediante bloques *always* y contadores genera otras tres señales con frecuencia reducida a 4f (*clk4f*) , 2f (*clk2f*) y f (*clkf*).

En la figura 1, se muestra la simulación del modulo generador de relojes, donde si se observa con detenimiento se aprecia como efectivamente la señal *clk16f* de color rojo tiene una frecuencia 16 veces más grande que la señal *clkf* de color verde. De forma similar, las señales *clk4f* (de color naranja) y *clk2f* (de color amarillo) tienen una frecuencia 4 y 2 veces más grande que la señal *clkf*, respectivamente. Se observa como la descripción conductual y la estructural, sintetizada por Yosys, se comportan de manera idéntica.

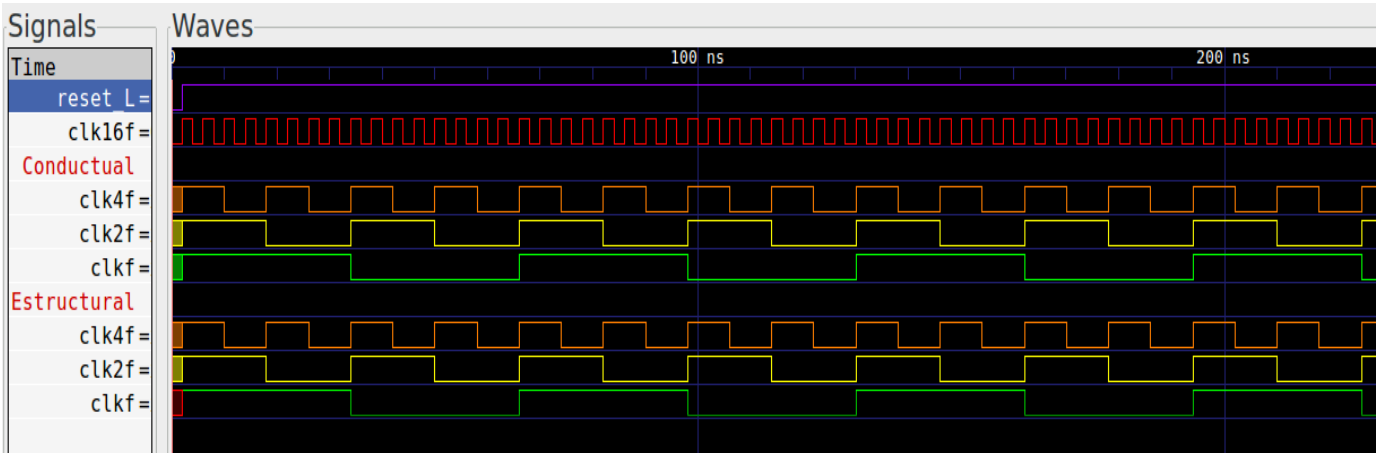


Figura 1: Señales generador de relojes [Autoría Propia]

1.2. Lógica muxes

Para los multiplexores de transmisión, en la primera etapa se requiere uno 4 a 2, el cual manipula el dato según su bit de válido. Ante una entrada válida transmite el dato como se encuentra con su bit de válido en alto, en caso contrario propaga el último valor válido que hubo en esa entrada con el bit de válido en 0.

El objetivo de estos mux es transmitir los 4 datos que se encuentran en la entrada y propagarlos cada vez con menos bits para enviarlos al probador. Debido a esto se elige una arquitectura como la que se muestra en la Fig. 1.2 en donde la frecuencia la primera etapa es el doble de rápida que el segundo elemento y así el segundo puede propagar correctamente los valores de sus entradas; para esto se asigna en las entradas del clk de cada módulo mux el reloj correspondiente, generado en la sección anterior.

Para describir el mux 4:1 se decide utilizar 2 mux 2:1 ya que de esta forma puede utilizarse un selector automático que siempre itere entre las entradas, para la segunda etapa se requiere solo un mux 2:1, teniendo el cuidado de asignar correctamente los clock

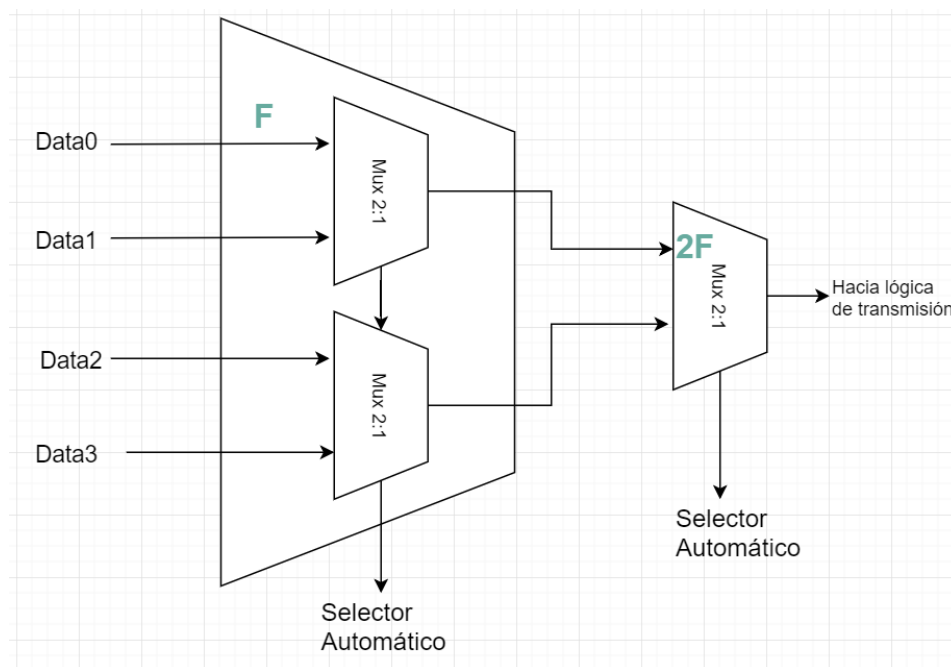


Figura 2: Arquitectura de la lógica de muxes [Creación propia]

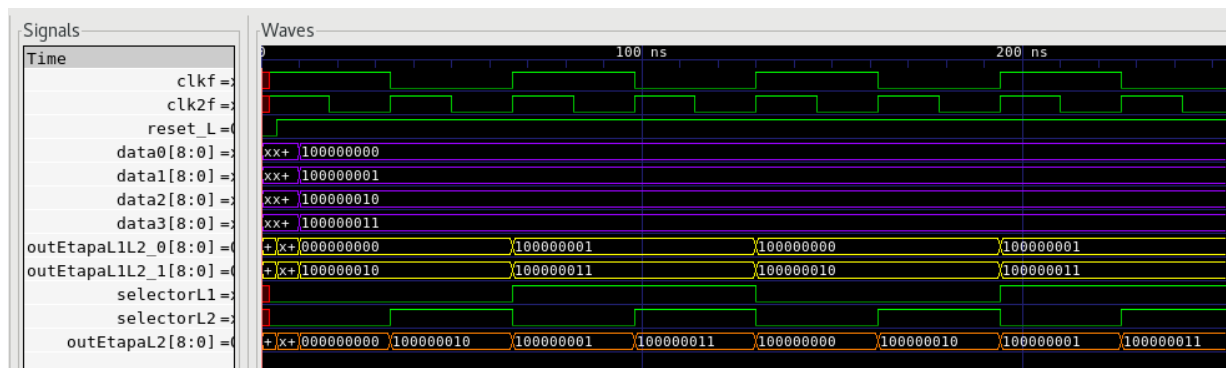


Figura 3: Señales lógica de muxes prueba 1 [Creación propia]

En la figura 3 se muestra la simulación de la lógica de muxes. Se observa que cuando todas salidas son válidas, en la salida se pone según la señal de reloj, cada una de las entradas. Para facilitar el funcionamiento de la lógica de muxes, se puso como dato en las entradas el número de entrada como se observa en la figura 3.

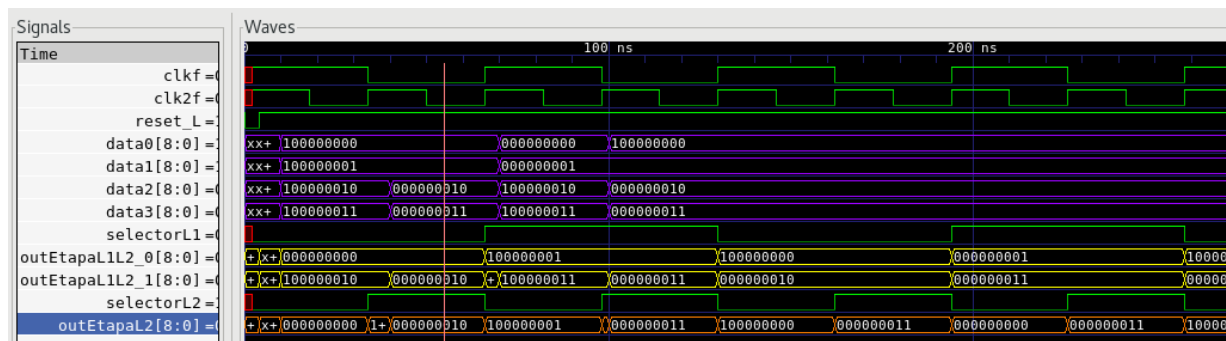


Figura 4: Señales lógica de muxes prueba final [Creación propia]

En la figura 4 se muestran la simulación invalidando algunas señales de entrada. Se observa que en ese caso, la lógica de muxes se comporta según lo solicitado al poner el último valor válido con el bit de valido en cero. Cuando las señales se validan, se ponen según el orden que se mostró en el caso cuando todas las entradas son validas.

1.3. Paralelo - 2b serial

Se hizo un convertidor paralelo a "serial" de 2 bits que recibe como entrada un bus de 9 bits cuyo bit más significativo indica si el dato que se está transmitiendo es válido y los restantes 8 son el dato en sí. Un diagrama simplificado del módulo se presenta en la figura 5.

El diseño realizado contiene las siguientes entradas y salidas:

- Entrada **clk4f**: es el reloj que controla la llegada de datos de 9 bits al módulo
- Entrada **clk16f**: es el reloj que controla el muestreo de los datos dentro del módulo. Es importante recalcar que como se deben pasar 8 bits (los 8 del dato, el de valido no se transmite en esta etapa) en un bus de solamente 2 bits, esta frecuencia de muestreo para enviar los 8 bits de 2 en 2 es cuatro veces más alta que la frecuencia a la que se reciben los datos a la entrada del módulo.
- Entrada **reset**: Es el reset del sistema de transmisión de datos, el cual cuando se encuentra en 1 hace que se mande el código \$BC a través del bus de 2 bits.
- Entrada **reset_L**: es una señal que cuando está en bajo inicializa los valores de las variables internas del módulo y de la salida.
- Entrada **paralelo**: es el bus que ingresa al módulo que contiene 9 bits. Como se indicó anteriormente, el bit más significativo indica si el dato es válido o no (1 es válido, 0 es inválido). Los bits restantes son el dato que se quiere transmitir.
- Salida **serial**: Es el bus de 2 bits de salida del módulo. Cuando la entrada reset está en alto o el bit valid del dato está en bajo, se pone en este bus el valor de \$BC. Sino, se transmite el dato que viene desde el bus de entrada.

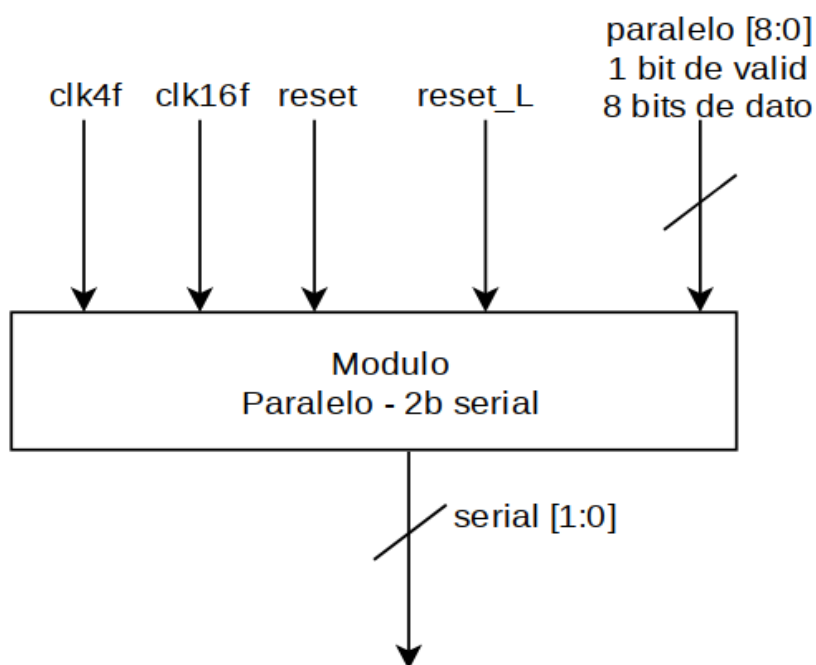


Figura 5: Convertidor paralelo a 2b serial [autoría propia]

En la figura 6 se muestran los resultados de las pruebas realizadas al módulo. En el primer ciclo del reloj *clk4f* la entrada *reset* se tiene en 1, por ende la salida *serial* presenta los valores \$BC (aunque la entrada \$FF tenga el valid en 1), los cuales están codificados en binario como 1011 1100. Luego de esto, a la entrada *reset* se pone en bajo y la salida *serial* empieza a pasar el valor de la entrada *paralelo*, que en este caso es \$55 con el bit valid en 1, el cual codificado en decimal es 0101 0101. Un ciclo después del reloj *clk4f* la entrada paralelo toma el valor de \$00 también con el bit de valid en 1, por ende pasa a la salida. Un ciclo del reloj *clk4f* después se pone a la entrada el valor \$FF pero con el valid en cero, por ende a la salida se ponen de nuevo los valores \$BC codificados en binario como 1011 1100, mismo caso para cuando después se pone la entrada \$55 con el bit valid en cero. Finalmente, se pone de nuevo a la entrada el valor \$00 con el valid en 1 por el resto de la simulación, y se observa que este valor se propaga a la salida. Es importante recalcar el funcionamiento idéntico entre las descripciones conductual y estructural.

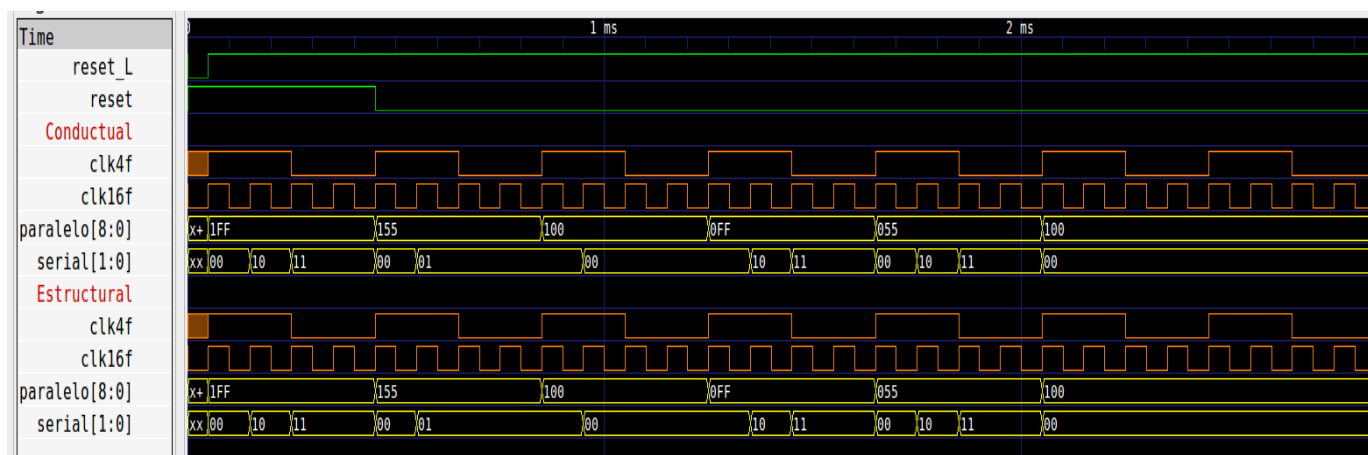


Figura 6: Comprobación funcionamiento Paralelo - 2b Serial [Autoría Propia]

1.4. 2b Serial a Paralelo

Para esta sección se toma la salida de la sección anterior y se prueba recuperar su señal de entrada a en la salida del modulo *serialParalelo*. Se conectan los módulos como se muestran en la Fig. 7; la entrada serial de 2 bits corresponde a la de la etapa anterior, el cl16 se genera desde el probador así como el reset16, por salida se tendrá el registro outParalelo de 9 bits, el cual se pasa a la etapa de demuxes.

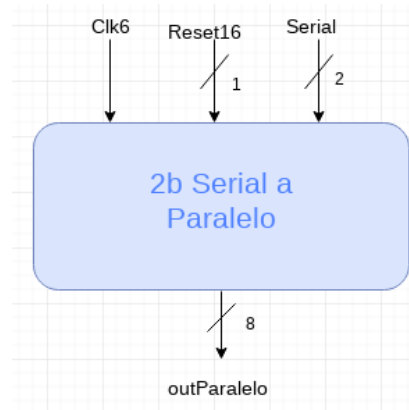


Figura 7: Convertidor 2b serial a paralelo a [autoría propia]

Para la lógica de esta sección se toma utilizan las siguientes variables :

- contador[2:0], registro en que se cuenta los BC recibidos.
- i[1:0], registro contador de pares de serial que se han recibido, si es 4 se propaga el dato hacia outParalelo.
- paralelo[7:0], registro de 8 bits que almacena las entradas de serial
- active, wire que está alto cuando contador es ≥ 4
- com , wire que se encuentra en 1 si paralelo es \neq BC
- valid, señal que se activa cuando active y com están en alto

Para detectar los cambios de paralelo, se revisa en cada ciclo si ya alcanzó el valor de BC, si lo hizo se asigna este valor a paralelo y posteriormente en outParalelo, como la concatenación valid,paralelo.

Para poder probar su funcionalidad se envían primero 3 datos inválidos, los cuales incrementan *contador* y se observa en la señal *paralelo* como hay 3 BC antes de los primeros 3 marcadores de la Fig.8. Posteriormente se envía el dato válido 100, obteniéndose a la salida un 000, como es esperado ya que *contador* no es aún mayor o igual a 4 , después se envía un dato inválido más y dos válidos para observar como al aumentar el contador a 4 la señal de valid se activa y se tienen señales válidas en *outParalelo* comprobándose así la correcta propagación y recibimiento de las señales.

En esa misma Figura se tiene la gráfica del convertidor estructural en la que las salidas son iguales a las del conductual.

Utilizando la instrucción make all se puede correr todo el ejemplo con la gráfica en gtkwave de las señales de salida.

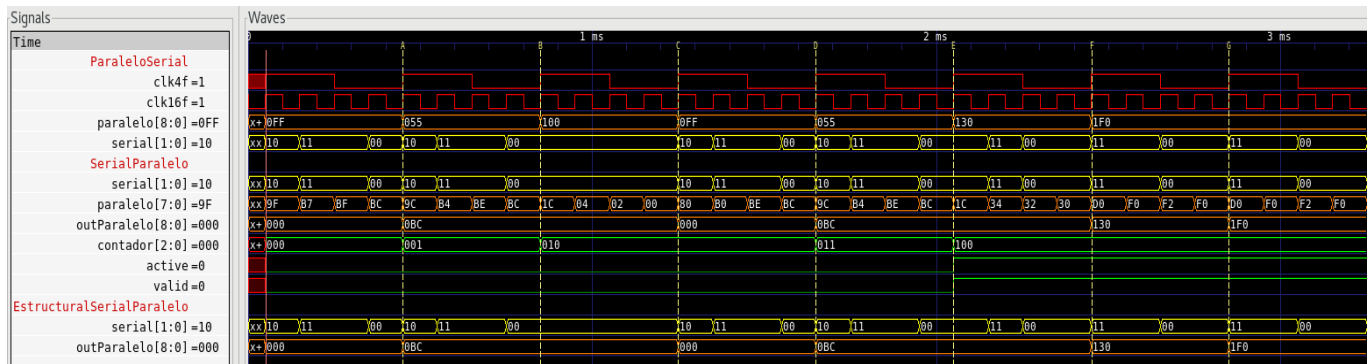


Figura 8: Señales convertidor 2b serial a paralelo, conductual y estructural [autoría propia]

1.5. Lógica demuxes

Para la lógica de demuxes en el receptor, se tiene un funcionamiento igual al de la lógica de muxes, con el pequeño cambio de que los datos viajan en sentido opuesto. Por lo que el diseño arquitectónico es igual al de la figura 1.2 con todas las flechas en sentido contrario.

Se parte de un mux 1 a 2, esto es, que los datos entran por una sola entrada y salen por dos salidas. Se transmitirán sin ningún problema los datos con bit de valid igual a 1, en caso contrario, se pone en las salidas el último dato válido, pero con el bit de valid en cero.

La siguiente etapa consiste en tomar las anteriores 2 salidas y obtener 4 salidas que corresponden a los datos transmitidos. Para esto se utilizan dos muxes 1 a 2.

Es fundamental utilizar los relojes correctos para la efectiva propagación de los datos. Se tienen frecuencias más altas en el principio de la lógica de demuxes, disminuyendola conforme se van teniendo más salidas.

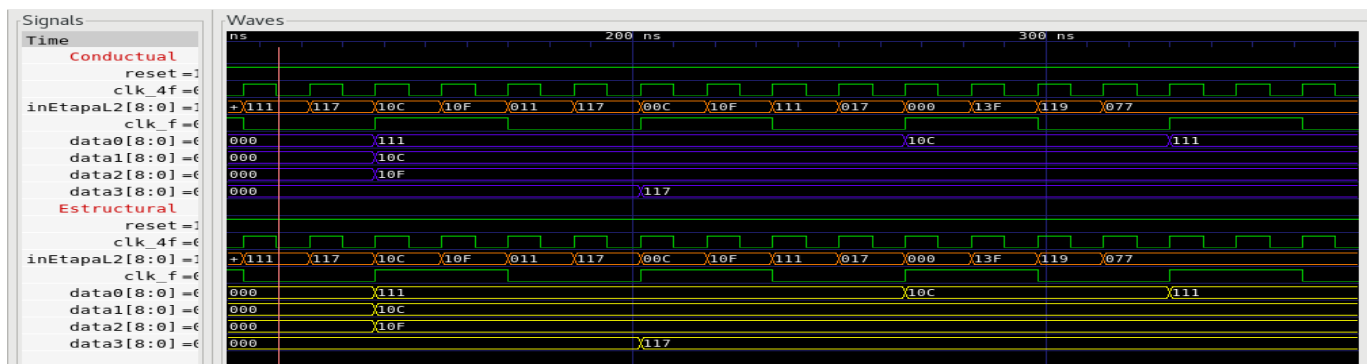


Figura 9: Señales conductual y estructural lógica demuxes [autoría propia]

En la figura 9 se muestran los resultados obtenidos al simular la descripción conductual y la descripción estructural obtenida con yosys. Se procede primeramente a enviar una secuencia de datos varias veces, se observa que en las salidas finales de ambas descripciones, se tienen los datos definidos en la secuencia. Cada salida tiene su dato de entrada. Seguidamente, se procedió a invalidar ciertas entradas y se observa que el demux funciona adecuadamente, conservando en la salida el último

valor válido pero con el bit de valid igual a cero. Para compilar, sintetizar y ejecutar las simulaciones de ambas descripciones, se debe de ejecutar el comando **make** dentro de la carpeta **DemuxesLogic**

Referencias

- [1] Material del curso Circuitos Digitales 2 IE-0523
- [2] Enunciado del proyecto brindado por el profesor