Universidad de Costa Rica | Facultad de Ingeniería | Escuela de Ingeniería Eléctrica

Tarea 6: Implementación de una entrada valid y muxes de 4 bits

Roberto Sánchez Cárdenas - B77059

San José, 17 de septiembre de 2020
Digitales II

Índice

1. Introducción

La idea de esta tarea es utilizar las descripciones desarrolladas previamente para implementar nuevos componentes a partir de esos resultados. Como punto inicial, se desea implementar una entrada de valid, que se encarga de decir si un valor es válido o no, por lo tanto si la entrada valid está en 0, el valor anterior se mantiene en la salida.

Una vez se tiene desarrollado el mux 2:1 de 2bits con una entrada valid, el módulo se debe usar par formar un mux 2 a 1 de 4 bits y uno 4 a 1 de 4 bits. Por cada entrada debe haber una entrada de validación. A la salida se tiene una salida de validación, que nos dice si el valor actual de la salida es uno nuevo o se mantuvo el previo debido al valid.

Cada uno de los módulos diseñados se sintetizó automáticamente a partir de compuertas diponibles en cmos_cells con la herramienta Yosys.

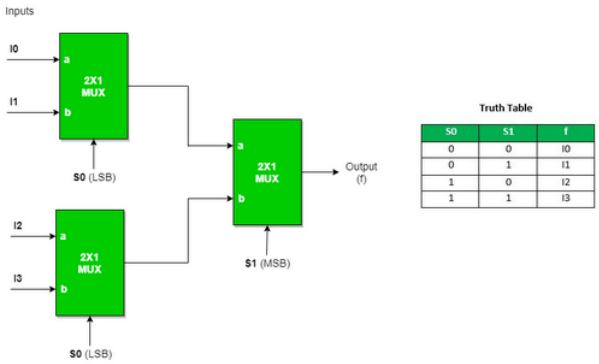

Figura 1: Mux 4 a 1 formado con muxes 2 a 1. (Tomada de GeeksforGeeks)

2. Plan de pruebas

Para probar los componentes diseñados y que la opción valid funcione bien se pensó en una serie de pulsos que prueban la gran mayoría de casos posibles. A continuación se muestran dos imágenes donde se observan las salidas con dos bits y con 4 bits.

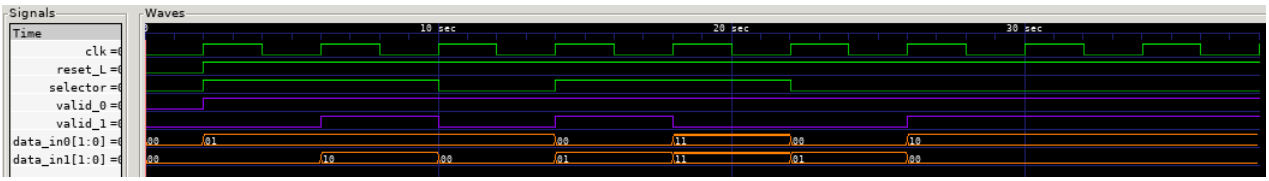

Figura 2: Plan de pruebas para dos bits y 2 entradas

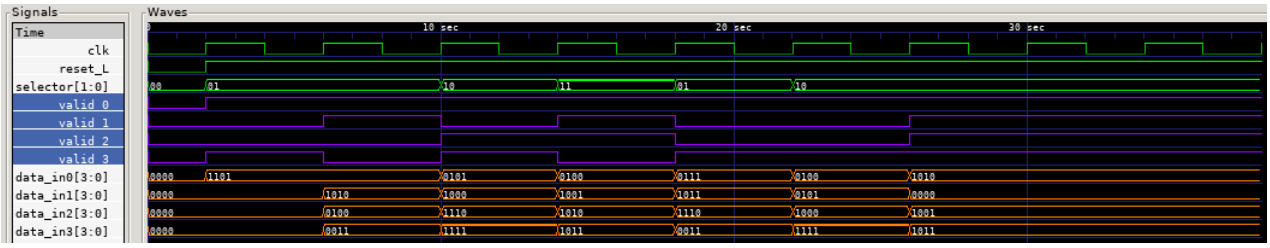

Figura 3: Plan de pruebas para cuatro bits y hasta 4 entradas

El caso de 4 bits se usa para el mux 2 a 1 de 4 bits. Cabe destacar que el plan para dos bits también es tomado de esta última, sin embargo solo se toman los dos últimos bits de la entrada. Solo se usó un probador para todos los casos.

3. Instrucciones de simulación

Para correr el programa de diseñó un makefile, que compila automáticamente todo y modifica ciertos nombres de variables. Se puede correr simplemente escribiendo make en la terminal,

estando en la dirección en donde están los archivos. Se puede correr cada mux individual usando los siguientes comandos.

- $make mux2a1_2b <- sintetiza y corre mux 2 a 1 de 2 bits
- $make mux2a1_4b <- sintetiza y corre mux 2 a 1 de 4 bits
- $make mux4a1_4b <- sintetiza y corre mux 4 a 1 de 4 bits

4. Resultado y análisis

Usando los planes de prueba mostrados en las figuras 2 y 3, se probaron los diseños realizados. En la siguiente imagen se muestran los resultados del mux 2 a 1 para 2 bits con entradas de validez. La salida valid está en 1 cuando la señal se actualizó.

En la siguiente figura se muestra el diagrama sintetizado con Yosys del mux 2a1, donde se pueden observar las entradas de valid.

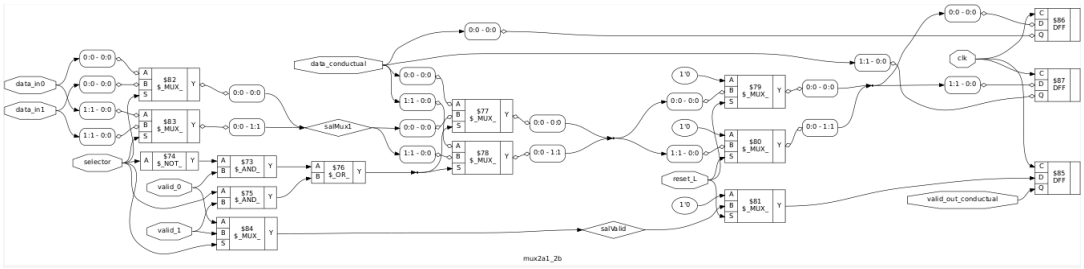

Figura 4: Diagrama sintetizado de mux 2:1 para 2 bits

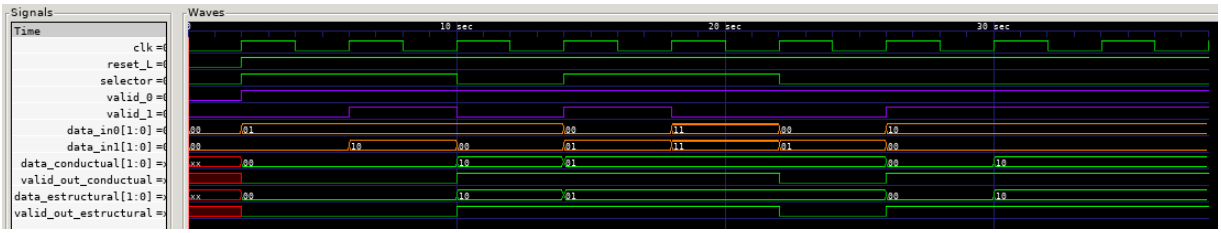

Figura 5: Resultado de mux 2 a 1 para dos bits

Se puede notar que las salidas comienzan reseteadas debido al reset_L, por este motivo la salida de valid comienza en 0 y confirma que la primera entrada es correcta. En el tercer ciclo de reloj el mux leyó un un 1 y en valid_1 hay un 1, por lo tanto en la salida este cambio se ve reflejado, lo mismo ocurre en el siguiente flanco positivo, más el selector está en 0 y valid_0 en 1. En el sexto ciclo se observa un error porque valid_1 está en 0. Ambas descripciones tienen el mismo comportamiento.

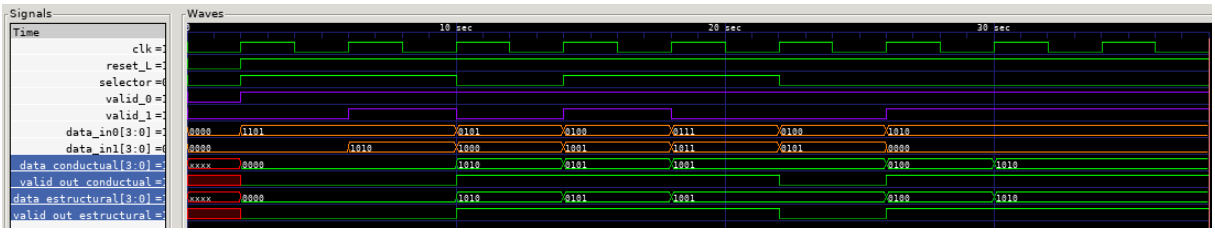

Figura 6: Resultado de mux 2 a 1 para cuatro bits

En el caso del mux 2:1 para 4 bits de entrada se muestra en la figura 6. Este caso se puede comparar con el primero puesto que el selector utilizado y señales de valid son las mismas, por lo tanto tiene un comportamiento de cambios idéntico, si se observa con atención, los últimos dos bits de la salida, concuerdan con los del caso previo. Este mux fue creado a partid del mux 2:1 de 2 bits, al usar dos en paralelo, de uno se toman los dos primeros bits y del otro se toman los otros dos, que se pasan a un bus de datos.

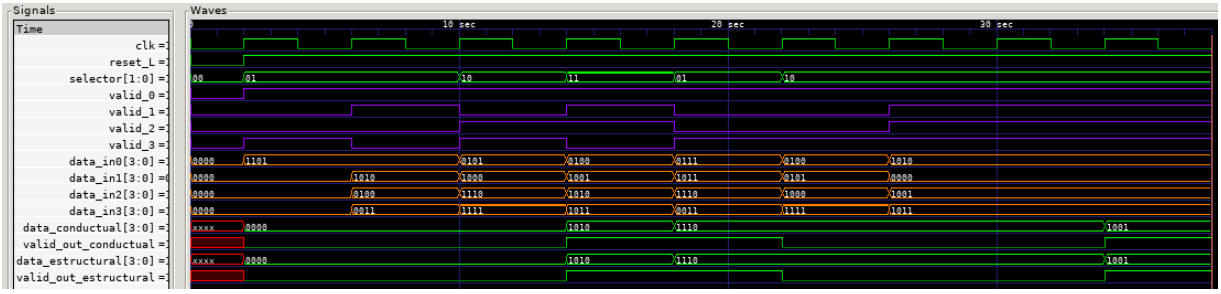

Figura 7: Resultado de mux 4 a 1 para cuatro bits

Finalmente se tiene un mux 4:1 para 4 bits, el cuál fue diseñado utilizando el caso anterior. En este caso es importante notar que como se tiene dos flipflops que conforman el camino más largo, la señal se ve retardada 2 ciclos de reloj. En el primer flanco positivo se ponen las salidas en 0 debido al reset previo. En el segundo flanco positivo se puede ver un comportamiento adecuado, el selector marca 01, el valid_1 está en 1, por lo tanto, dos flancos de reloj posteriores se ve reflejado el dato 1010. En el tercer flanco también se da otra señal válida, pero en este caso se toma la señal data_in2. Cuando se llega al cuarto flanco se pide la señal de 11, pero el valid_3 está en 0, por lo tanto el valor no se ve reflejado, dos flancos de reloj después en valid_out hay un 0.

Para este caso se tuvo que modificar la señal del selector que llega al mux de salida. La señal del selector en el bit más significativo se pasó por un flip flip para que se tome esa señal un ciclo de reloj después, que es cuando se lee en ese mux. Si no se hacía esto, se leeía el selector en dos puntos distintos para una misma salida.

5. Conclusiones y recomendaciones

Es muy sencillo crear circuitos sencillos que tomen mas bits a partir de diseños para una cantidad inferior de bits, basta con conectar dos elementos en una especie de paralelo donde uno toma cierta cantidad de bits y otro el resto, para pasar todo por un bus.

Se deben tener cuidado al trabajar con Yosys y los nombres que este genere en los archivos sintetizados, puesto que pueden tener conflictos al ser probados con la descripción conductual por los nombres.

Si se usan muxes con flops a la salida es necesario que se añadan retardos a las señales intermedias para no perder la info que queríamos al inicio del ciclo de interés.

Todos los diseños realizados funcionaron como era esperado.

6. Tiempo empleado

- Descripciones: 1.5 horas horas
- Makefile: 30 minutos
- Debuggeo debido a errores extraños: 2.5 horas
- Reporte: 1 hora y 30 minutos