Proyecto Optativo 2: Diseño de un controlador de bus

Fecha de entrega: viernes 7 de Junio

Resumen

En este proyecto os pedimos que diseñéis el controlador de un dispositivo esclavo de un bus que utiliza el protocolo de sincronización semi-síncrono parecido al PCI visto en clase. El dispositivo es una pequeña memoria RAM cuyas entradas y salidas están "latcheadas" (es decir, hemos colocado un registro en cada entrada y salida de la memoria, menos en las señales de control que genera nuestro esclavo internamente). El controlador debe monitorizar el bus, identificar cuándo le realizan una solicitud y responder a ésta, ya sea escribiendo o leyendo, y enviando el dato que le piden. Por supuesto debe gestionar correctamente su parte del protocolo de sincronización.

Detalles del sistema a diseñar:

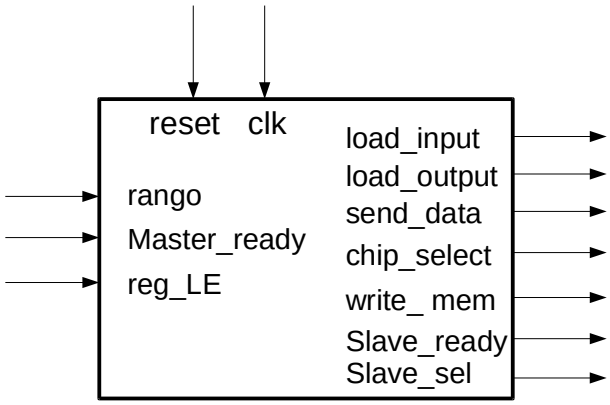
En esta práctica vamos a trabajar a partir del archivo p2_opt.cct. En este archivo tenemos:

- Un **conjunto de switches y teclados hexadecimales** que utilizamos para simular el comportamiento del master. Con ellos podemos solicitar y enviar datos al esclavo, cumpliendo con la parte del protocolo que le toca al master.
- El **dispositivo esclavo**, que es una memoria RAM de 4 palabras con registros en la entrada y la salida, y un buffer triestado que permite volcar un dato al bus de datos cuando corresponda. De las 16 palabras del espacio de direcciones **a nuestro esclavo le corresponde el siguiente rango: desde la 0000 a la 0011**. Debéis tener en cuenta esto, para identificar si el esclavo debe responder o no a una solicitud del master, y para conectar convenientemente los bits de la dirección a donde toque. En este dispositivo falta por incluir la unidad de control que se ocupe de que las transferencias funcionen correctamente. **Importante**: **la memoria RAM tiene un retardo de 22 ns**, **y el reloj del sistema trabaja a 20 ns**; debéis tenerlo en cuenta al diseñar la unidad de control.
- **Un bus** con las siguientes características:
    - 4 líneas de direcciones, por tanto hay 16 palabras direccionables
    - 4 líneas de datos, por tanto trabaja con palabras de 4 bits
    - 5 líneas de control:
        - Master:
            - *Bus_req*: la activa el master para informar de que quiere realizar una transferencia.
            - *L/E*: Si vale 0, la operación solicitada es de lectura y si vale 1, de escritura
            - *Master_ready*: se usa en las operaciones de lectura para indicar que el master no puede leer el dato en el ciclo actual. Cuando se active el esclavo, deberá mantener el dato en el bus hasta que la señal se desactive.
        - Slave:
            - *Slave_ready*: indica que el esclavo puede terminar la transferencia en el ciclo actual. En escritura se activa tan pronto como se dé la orden de escribir en la memoria. En lectura se activa cuando el dato solicitado esté en el bus.
            - *Slave_sel*: indica que el esclavo ha reconocido que la solicitud actual del bus es para él (es decir que Bus_req está a 1 y la dirección está dentro del rango del esclavo).

Diseño de la Unidad de Control:
La unidad de control debe monitorizar el bus e identificar si el master ha realizado una solicitud a la que el esclavo deba responder. En caso afirmativo, leerá la información que hay en el bus y gestionará la solicitud:

- Si el master solicita una lectura, el esclavo debe buscar el dato, colocarlo en el bus, y asegurarse de que el master lo recibe. Esto quiere decir que si la señal de *master ready* está a cero, el esclavo mantendrá el dato en el bus. En el momento en que *master_ready* esté a uno, el esclavo liberará el bus.
- Si solicita una escritura deberá coger el dato y escribirlo en la memoria RAM.

En ambos casos deberá gestionar las señales de *slave_ready* y *slave_sel* convenientemente. Las entradas y salidas de la Unidad de control se muestran en la siguiente figura:



La unidad de control se puede implementar como una máquina de estados tanto Moore como Mealy. Se valorará que uséis el menor número posible de estados. La lógica de la función de transición (la que genera el estado siguiente) se puede implementar libremente. **La lógica de la función de salida la debéis hacer con puertas lógicas** (es decir no podéis utilizar una memoria ROM o similar).

Resultados
En primer lugar debéis comprobar que el diseño funciona correctamente para todos los casos posibles.

Además, debéis presentar una memoria explicando brevemente vuestro diseño. En esta memoria hay que incluir **el diagrama de estados** de la unidad de control, explicando qué se hace en cada estado, cómo se codifican los estados, y qué señales se activan en cada uno. Adicionalmente, hay que describir la implementación realizada explicando qué hardware habéis incluido y qué función lógica realiza. Para ello no basta con poner una imagen de vuestro diseño, sino que es preciso también incluir **las fórmulas que describen las funciones lógicas** que implementáis.

En la memoria debéis evaluar vuestro diseño:

¿Cuántos ciclos tarda como mínimo en gestionar una petición de lectura? ¿Y de escritura? **Debéis incluir un cronograma** en el que se vea el funcionamiento de estas dos operaciones.