IE-0523 Circuitos Digitales II
I Ciclo 2022
Prof. E.Coen 24 de mayo de 2022

Tarea #4
(Entrega 5 de junio del 2022)
Diseño de un generador de transacciones MDIO

1. Diseñar un generador de transacciones MDIO de acuerdo con las especificaciones estipuladas en la cláusula 22 del estándar IEEE 802.3 (disponible en la página de Mediación Virtual del curso), particularmente las secciones 22.2.2.13 y 22.2.2.14.

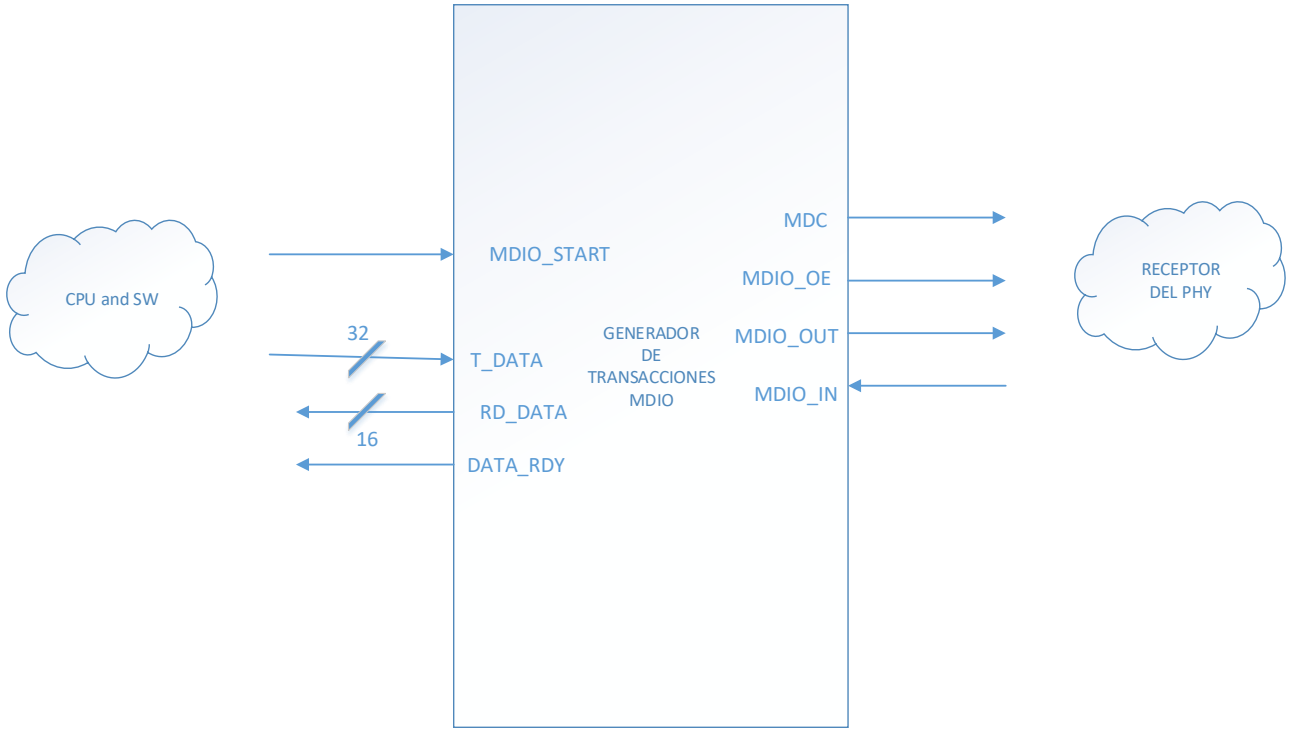

Figura #1: Generador de transacciones MDIO

Interfaces del generador:

a) ***CLK (no se muestra en la figura) –*** Es una entrada que llega al generador de transacciones desde el CPU con una frecuencia determinada. El flanco activo de CLK es el flanco creciente.
b) **MDC**– Salida de reloj para el MDIO. El flanco activo de la señal *MDC* es el flanco creciente. Observe que MDC es una salida del generador, que deberá tener una frecuencia de la mitad de la frecuencia de la entrada CLK. El generador debe generar MDC con la frecuencia correcta para cualquier posible valor de la frecuencia de entrada CLK.
c) ***RESET (no se muestra en la figura)*** – Entrada de reinicio del generador. Si *RESET*=1 el generador funciona normalmente. En caso contrario, el generador vuelve a su estado inicial y todas las salidas toman el valor de cero.
d) ***MDIO_START*** – Strobe (pulso de un ciclo de reloj). Indica al generador que se ha cargado un valor en la entrada T_DATA y que se debe iniciar la transmisión de los datos a través de la salida serial.
e) ***MDIO_OUT*** – Salida serial. Cuando se habilita MDIO_START=1 se envía a través de la

salida MDIO_OUT los bits que se observan en la entrada T_DATA, empezando por el bit más significativo y hasta completar los 32 bits.

f) ***MDIO_OE*** – Habilitación de MDIO_OUT. Esta salida debe detectar si la transacción que se está ejecutando es una transacción de lectura o de escritura. En una transacción de escritura, debe permanecer en alto durante los 32 ciclos de la transacción, pero ponerse en bajo al terminar la transacción. En una transacción de lectura, debe permanecer en alto durante los primeros 16 ciclos de la transacción, pero debe ponerse en cero durante los siguientes 16 ciclos, mientras se recibe el dato de MDIO_IN. Al final de la transacción de lectura, la salida también debe permanecer en cero.

g) ***MDIO_IN*** – Entrada serie. Durante una operación de lectura (de acuerdo a la cláusula 22 del estándar), se debe leer el valor de esta entrada durante los últimos 16 ciclos de la transacción de MDIO y escribirlos en la salida RD_DATA.

h) ***T_DATA[31:0]*** – Entrada paralela. Cuando se habilita MDIO_START, en el siguiente ciclo de reloj se transmite el bit T_DATA[31] por la salida MDIO_OUT y durante los siguientes ciclos se transmite un bit por ciclo hasta completar el envío de la palabra completa.

i) ***RD_DATA[15:0]*** – Esta salida debe producir los 16 bits que se reciben desde el PHY durante una transacción de lectura recibida en MDIO_IN. El valor de RD_DATA solo es válido cuando DATA_RDY es igual a 1.

j) ***DATA_RDY*** – Esta salida se pone en 1 cuando se ha completado la recepción de una palabra serial completa durante una transacción de lectura.

De acuerdo a la cláusula 22 del estándar IEEE 802.3, una transacción básica de MDIO es una transacción serial de 32 bits, cuyos campos se definen como se muestra en la figura #2 (figura #13 en la referencia):

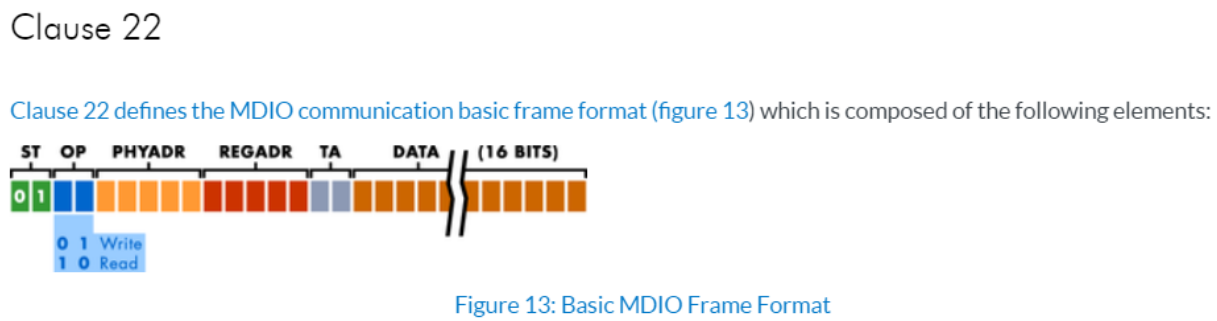

Figura #2: Transacción de MDIO, tomada de [este sitio web](#).

2. Escribir una descripción conductual del generador de transacciones de MDIO usando Verilog. Esta descripción servirá como una especificación detallada y formal del funcionamiento del dispositivo diseñado.

3. La descripción en Verilog deberá tener al menos un módulo de banco de pruebas, un módulo probador, y un módulo con la descripción del generador. Use Icarus Verilog.

4. Definir un plan de pruebas para garantizar el funcionamiento del diseño. El plan de pruebas debe cubrir todos los modos de operación del generador, es decir, transacciones de lectura, de escritura y RESET. El módulo probador debe suministrar las señales necesarias para que las pruebas se realicen.

5. Obtenga una descripción estructural a partir del programa de síntesis Yosys, mapeando su diseño a la biblioteca cmos_cells.lib. Debe entregar un archivo de síntesis que permita

correr todos los pasos de síntesis en una sola corrida e incluir una entrada en el Makefile para correr síntesis (ver ejemplo en las figuras 3 y 4 al final de este enunciado).

6. Para esta tarea NO se le pide que simule el circuito sintetizado, solamente la descripción conductual. Debe incluir en su reporte las formas de onda correspondientes a una transacción de lectura y una transacción de escritura.

7. Con el fin de facilitar el proceso de revisión, se le solicita organizar los entregables de la tarea de la siguiente manera:

   a) Entregar un solo archivo comprimido, y nombrado según el patrón <# de carné>.<formato de compresión>, por ejemplo B41047.zip

   b) El archivo comprimido descrito en el rubro a) deberá contener específicamente los siguientes archivos:

      - Reporte en formato PDF cuyo nombre debe seguir el patrón <# de carné>.pdf, por ejemplo, B41047.pdf
      - Uno o varios archivos de Verilog con el formato <nombre de archivo>.v que construyan la solución que se solicita en la tarea.
      - Un solo archivo probador llamado tester.v
      - Un solo archivo de banco de pruebas llamado testbench.v
      - Un script de síntesis para Yosys cuyo nombre siga el patrón <nombre de archivo>.ys, por ejemplo mdio.ys
      - Un archivo Makefile que permita correr todos los pasos de simulación y síntesis con una sola línea de comando, tal como se muestra en el ejemplo de la figura #4.

   c) El banco de pruebas, testbench.v, deberá incluir a todos los demás archivos *.v, de modo que la compilación y simulación del testbench implique la compilación y simulación de todos los archivos internos.

   d) El probador, tester.v, debe escribirse de forma tal que una sola simulación contenga los resultados de una escritura, una lectura y un proceso de RESET, que ejemplifiquen el funcionamiento esperado del módulo.

   e) El script de síntesis, por ejemplo mdio.ys, deberá tener una estructura similar a la que se muestra en la figura #3:

```
# read design
read_verilog lfsr16.v

# elaborate design hierarchy
hierarchy -check -top lfsr16

# the high-level stuff
proc; opt; fsm; opt; memory; opt

# mapping to internal cell library
techmap; opt

# mapping flip-flops to cmos_cells.lib
dfflibmap -liberty cmos_cells.lib

# mapping logic to mycells.lib
abc -liberty cmos_cells.lib

# write synthesized design
write_verilog synth.v
```

Figura #3: Script de síntesis en Yosys

f) El Makefile debe contener, como mínimo, los comandos necesarios para correr la compilación, simulación y síntesis de los módulos de la tarea, tal como se muestra en el ejemplo de la figura #4.

```
tarea: testbench.v mdio.ys #Archivos requeridos
	yosys -s mdio.ys    #Corre sìntesis
	iverilog -o salida testbench.v #Corre Icarus
	vvp salida #Corre la simulación
	gtkwave resultados.vcd #Abre las formas de onda
```

Figura #4: Ejemplo de Makefile

Rúbrica de Calificación

Tarea #4: Diseño de un generador de transacciones MDIO:	Categoría	% Categoría	% Rubro	% Total
Existe una descripción conductual en Verilog del diseño solicitado. Esta descripción incluye exactamente un módulo de banco de pruebas (testbench.v), exactamente un módulo probador (tester.v) y un módulo para el dispositivo bajo prueba (DUT).	Código	40%	20%	8%
Las descripciones de Verilog se entregan en archivos distintos al reporte, listos para ser simulados, e incluyen un archivo de Makefile, de modo que la simulación se corre con una sola línea de comando.	Código	40%	20%	8%
Las descripciones en Verilog están comentadas adecuadamente para que otras personas entiendan la lógica de la descripción.	Código	40%	20%	8%
Las descripciones en Verilog compilan sin producir errores.	Código	40%	20%	8%
Las descripciones en Verilog ejecutan correctamente. Es decir, corren, entregan algunos resultados y finalizan.	Código	40%	20%	8%
El dispositivo completa una operación de escritura de forma correcta de acuerdo con la especificación dada.	Pruebas	40%	35%	14%
El dispositivo completa una operación de lectura de forma correcta de acuerdo con la especificación dada.	Pruebas	40%	35%	14%
El código de Verilog sintetiza correctamente y produce los resultados de cantidad de compuertas por tipo. Estos resultados se incluyen en el reporte. Existe un script de síntesis con todos los comandos necesarios y este se ejecuta desde el Makefile.	Pruebas	40%	30%	12%
El reporte contiene las seguientes secciones: Resumen, descripción arquitectónica, plan de pruebas, instrucciones de utilización de la simulación para quien califica, ejemplos de resultados, conclusiones y recomendaciones.	Reporte	20%	40%	8%
El reporte explica con claridad los detalles relevantes del diseño particular que se hizo, las partes del diseño que dieron más trabajo para completar y por qué, una explicación de los problemas que se presentaron y cómo se solucionaron.	Reporte	20%	40%	8%
La longitud del reporte no excede 10 páginas.	Reporte	20%	20%	4%

Guía para el reporte (Sigue los mismo lineamientos del reporte de los proyectos)

Se debe entregar en forma electrónica un documento, a lo sumo de 10 páginas de longitud, que incluya los siguientes puntos:

1. **Resumen**: Breve (Media página máximo) descripción de **todo** el proyecto. Esta sección es fundamental pues puede determinar si el lector se interesa o no en leer los detalles del proyecto. Un resumen mal hecho puede esconder un excelente proyecto. El resumen debería incluir:

    a) Descripción breve del sistema, es decir, qué hace. Incluya alguna característica que considere que distingue este diseño en particular.
    b) Las pruebas que se realizaron y qué resultados se obtuvieron. Indique problemas que se tuvieron que considere importante resaltar.
    c) Conclusiones más importantes y recomendaciones para un diseño posterior.

2. **Descripción Arquitectónica**: Incluye un diagrama de bloques con las señales más importantes que sirve como base para describir el funcionamiento del sistema. La descripción va en términos de lo que se espera que el sistema haga. Es decir, se debe detallar la funcionalidad del sistema, el protocolo de las señales que se usan para que funcionen cada una de las partes y las secuencias de eventos que se deben dar. Esta descripción podría ir acompañada de tablas de verdad, tablas de transición de estados, diagramas de estados, diagramas temporales, etc.

3. **Plan de Pruebas**: Aquí se deben enumerar, esto es, se debe presentar una **lista detallada** de las pruebas que se le van a hacer al diseño para verificar que está funcionando de acuerdo a las especificaciones dadas. La lista debe contener por lo menos los siguientes elementos i) Nombre/número de prueba, ii) Descripción de la prueba, y iii) Una indicación de si el diseño la falló o la pasó. Estas pruebas podrían incluir la generación de vectores de entrada para probar en forma exhaustiva todas las líneas de una tabla de verdad o tabla de estados, patrones aleatorios de entradas para tratar de causar errores en la respuesta del diseño, o patrones específicos que ejerciten un cierto modo de funcionamiento. Cada prueba debería ser claramente enumerada en el plan para que también se pueda hacer referencia a ella en el código del banco de pruebas del diseño.

4. **Instrucciones de utilización de la simulación**: Esta sección debe mostrar los comandos necesarios para hacer funcionar la simulación en todos los casos que especifica el plan de pruebas. Hay que suponer que el diseño de un grupo puede ser utilizado por otro grupo o el profesor. Si los resultados no se pueden repetir porque no se conocen los comandos para hacer funcionar la simulación entonces es como si el diseño no funcionara del todo. Se recomienda crear un Makefile de modo que se pueda correr todas las pruebas del caso con un solo comando en Icarus Verilog y GTKwave.

5. **Ejemplos de los resultados**: Una descripción de los resultados más importantes acompañados de los diagramas temporales de la simulación (GTkWave) o cualquier otra salida que demuestre claramente el comportamiento descrito. No es necesario incluir una muestra exhaustiva de resultados, sino que los más representativos del diseño. El punto es mostrarle al lector los comportamientos más sobresalientes para formarle una idea clara

del funcionamiento del diseño. Ya verá el lector si desea más detalles, entonces podrá correr una simulación.

6. **Conclusiones y recomendaciones**: Basado en los resultados obtenidos se indica aquí qué se logró con el proyecto. Puede ser que se concluya que con el diseño propuesto se tiene una limitación en la velocidad de respuesta de… etc. O que con ciertas combinaciones de entradas el diseño se vuelve inestable o los resultados no son los esperados. También se puede concluir qué ventajas o problemas encontraron al seguir el plan de trabajo. A raíz de las conclusiones se puede también recomendar cómo se podría mejorar el diseño o qué otras pruebas se le podrían hacer para garantizar su funcionamiento en otras condiciones que al principio no se consideraron, o también cómo se debería planear el siguiente proyecto para poder cumplirlo a tiempo.