Universidad de Costa Rica
Facultad de Ingeniería

Escuela de Ingeniería Eléctrica
IE-0523 Circuitos Digitales II
Grupo 02 - I Ciclo 2024

Proyecto Final
Diseño de un Generador y un Receptor de Transacciones MDIO

Profesora: Ing. Ana Eugenia Sánchez Villalobos

Estudiantes:
Alexa López Marcos - B94353
Sylvia Elena Fonseca Cruz - C03039
Manfred Soza García - B97755
Ángeles Ulate Jarquín - C07908

Fecha: 01 de Julio del 2024

Índice

1. Resumen

Este proyecto consiste en el diseño de un generador y receptor de transacciones MDIO en Verilog. El mismo se apega a la cláusula 22 del estándar IEEE 802.3 (Ethernet), para una transacción básica de MDIO como lo es una transacción serial de 32 bits. Cabe resaltar que, el diseño y pruebas de funcionamiento de ambas partes se llevaron a cabo de forma independiente, al igual que su descripción conductual realizada a partir de FSMs. Para poner a prueba su funcionamiento, tanto para el generador como el receptor se definió un plan de pruebas en sus correspondientes archivos tester.v. Con ello se comprobó que, el generador diseñado cumplía exitosamente las pruebas de escritura, lectura y reset definidas. Al tiempo que, el receptor funcionaba perfectamente en sus tres modos de operación, es decir, en las transacciones de lectura, escritura y carga de memoria. A razón de lo anterior, se notó que el diseño ejecutado para el generador y receptor funciona óptimamente para las distintas transacciones de MDIO bajo las cuales ambas partes fueron testeadas. Sin embargo, entre las limitaciones está precisamente el funcionamiento independiente o la no conexión entre ambas partes, lo cual sería recomendable mejorar. Al igual que, la adición de un manejo de excepciones para cuando en los bits 3 y 4 del frame de MDIO no corresponda al OP Code de lectura o escritura, que acá fueron los únicos tomados en cuenta.

2. Descripción Arquitectónica

2.1. Generador de transacciones MDIO

Si bien la descripción de cada una de las señales de entrada, internas y de salida se realizó en el .v correspondiente al DUT de esta parte, un diagrama para su entendimiento gráfico se visualiza en la figura 1, donde se nota que este componente se interconecta con el CPU y SW y con la otra parte de este proyecto, el receptor del PHY o de transacciones de MDIO, a través de las distintas señales.

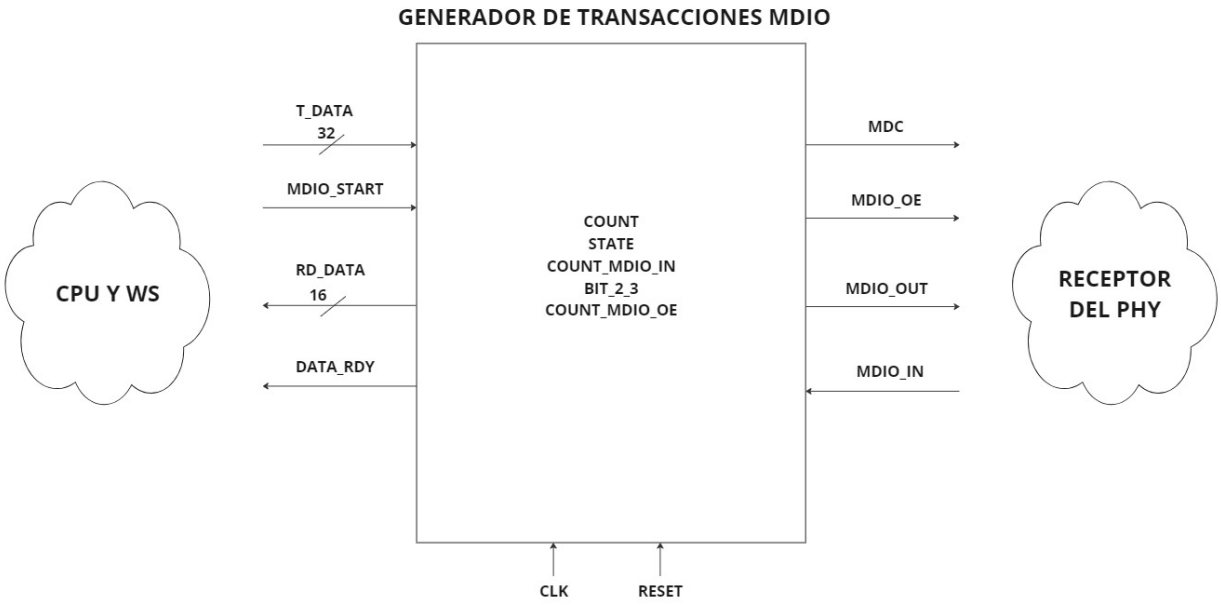

Figura 1: Diagrama de señales y salidas del generador de transacciones MDIO

Ahora bien, para la realización o descripción del funcionamiento del generador, se realizaron en Verilog, propiamente, tres archivos, correspondientes al DUT, Tester y Testbench. En el Testbench se dio la interconexión de estas otras dos partes, mientras que, en el DUT fue donde se explicitó el comportamiento de este componente (utilizando una FSM) y, en el tester, fue donde se dio la inicialización de su señal de reloj y se especificaron las pruebas (los 3 casos: lectura, reset y escritura) que debía superar el generador.

Una síntesis intuitiva de la lógica y flujo seguido para la confección del DUT se aprecia en la figura 2, donde también se comprende el funcionamiento deseado a cumplir por el generador.

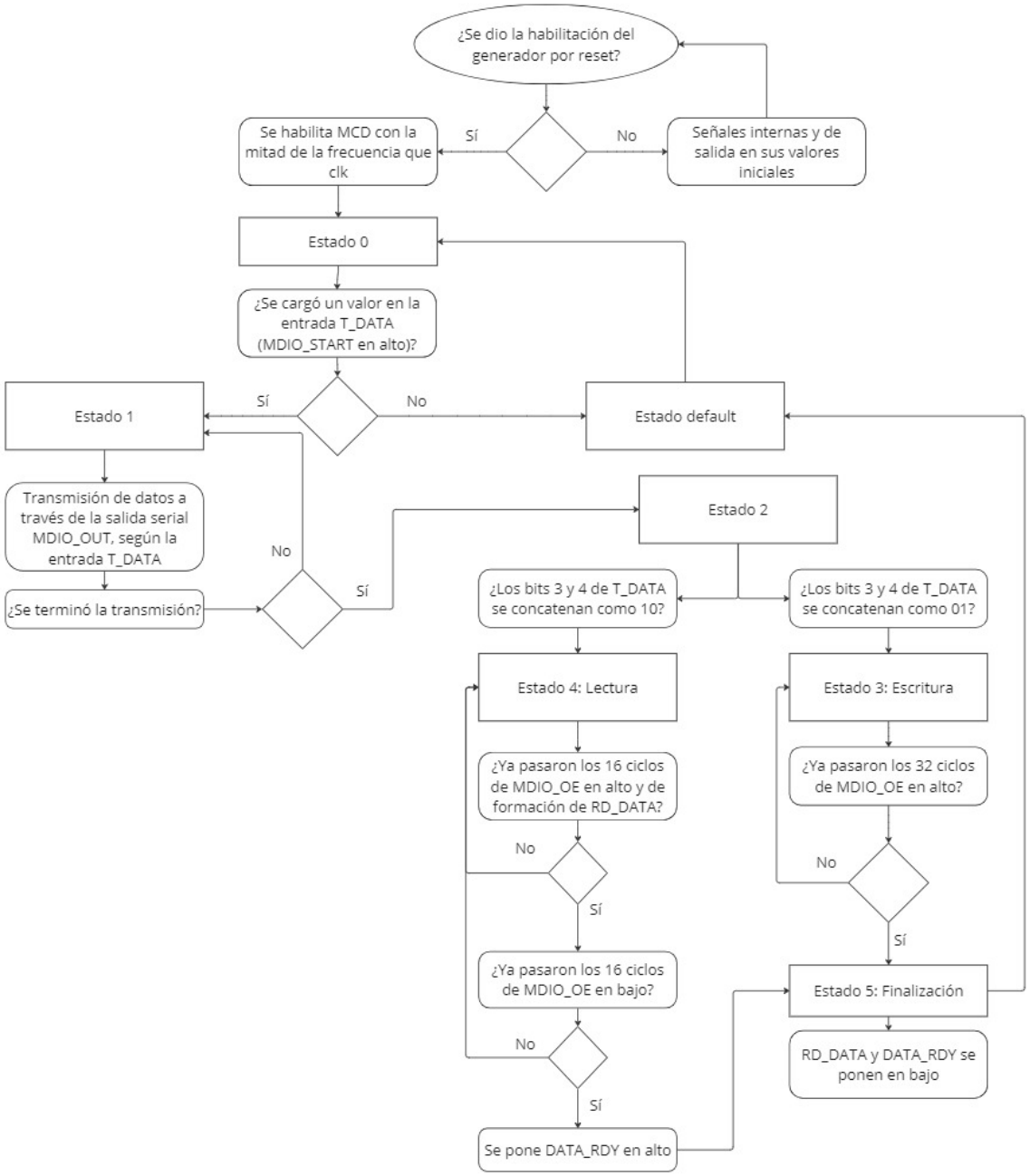

Figura 2: Diagrama lógico o de flujo empleado en el diseño del generador de transacciones MDIO

2.2. Receptor de transacciones MDIO

Al igual que con el generador, la descripción de cada una de las señales de entrada, internas y de salida se realizó en el .v correspondiente al DUT de esta parte. Un diagrama para su entendimiento gráfico se visualiza en la figura 1, donde se nota que este componente se interconecta con el CPU y SW y con la otra parte de este proyecto, el receptor del PHY o de transacciones de MDIO, a través de las distintas señales.

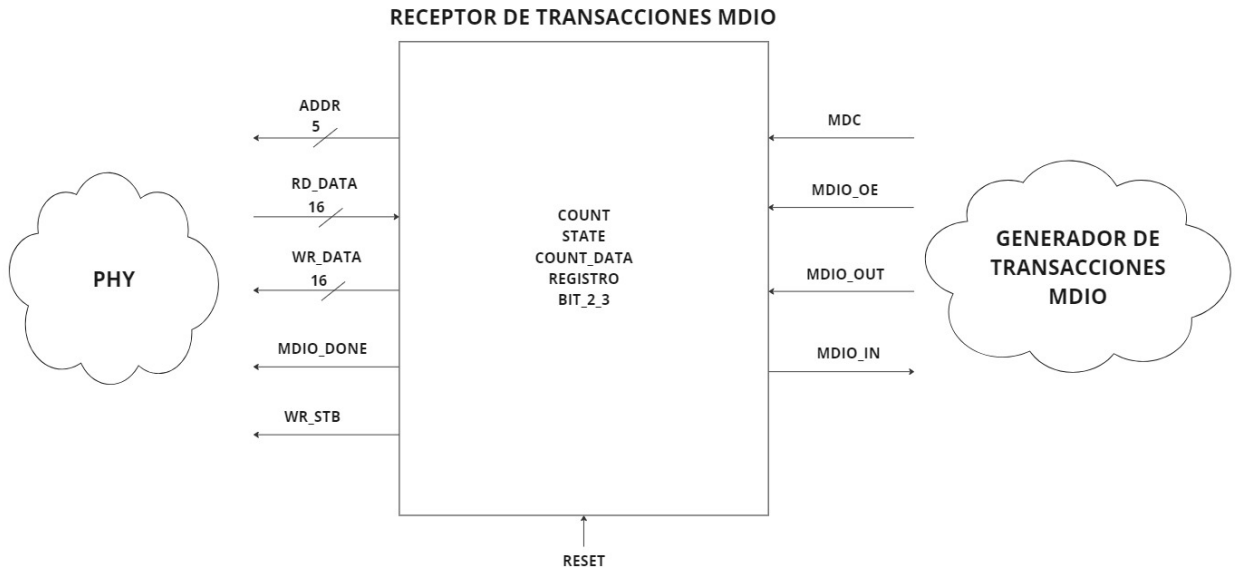

Figura 3: Diagrama de señales y salidas del receptor de transacciones MDIO

Ahora bien, para la realización o descripción del funcionamiento del receptor, se realizaron en Verilog, propiamente, tres archivos, correspondientes al DUT, Tester y Testbench. En el Testbench se dio la interconexión de estas otras dos partes, mientras que, en el DUT fue donde se explicitó el comportamiento de este componente (utilizando una FSM) y, en el tester, fue donde se especificaron las pruebas (los 3 casos: lectura, reset y escritura en memoria) que debía superar el receptor.

Una síntesis intuitiva de la lógica y flujo seguido para la confección del DUT se aprecia en la figura 4, donde también se comprende el funcionamiento deseado a cumplir por el generador.

3. Plan de Pruebas

3.1. Generador de transacciones MDIO

El plan de pruebas para el generador se especificó en su respectivo archivo tester.v, mismo que consta de 3 casos. Estos casos se especifican a continuación.

3.1.1. Para transacción de Escritura (bits 3 y 4 del frame de MDIO: 01)

De las transacciones de MDIO a evaluar, la de escritura fue la más sencilla por la misma funcionalidad lógica de este componente (apreciable en la figura 2). Por tanto, acá bastaba con haber inicializado el clk, que el rst estuviera en alto, especificar T_DATA y, por último, formar el pulso de MDIO_START. Propiamente, se utilizó:

```
// Se define T_DATA con el tercer y cuarto bit en 01
T_DATA = 32'b1001_0101_0011_0110_1010_1110_0101_0011;
```

Código 1: Prueba de escritura en generador.

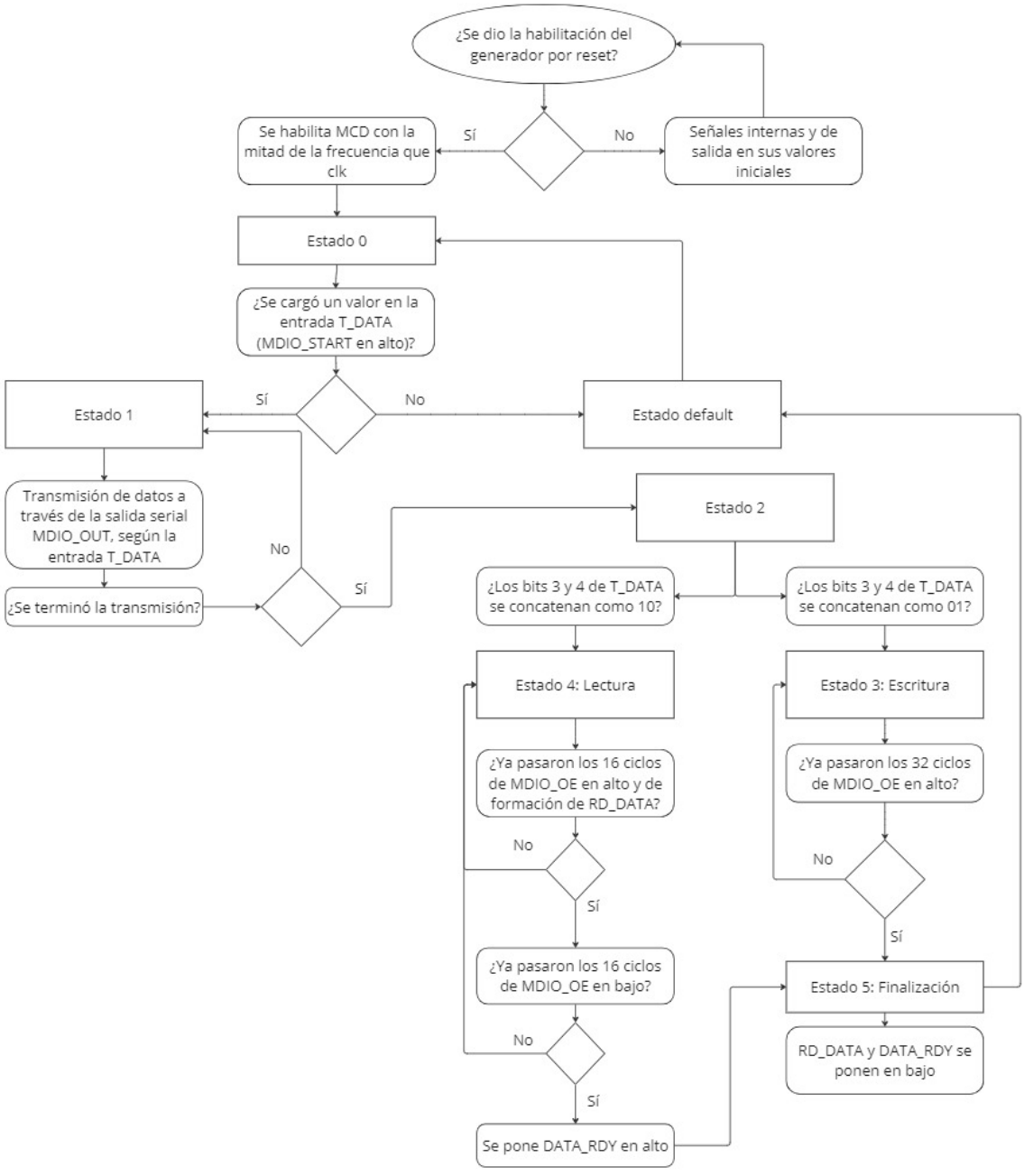

Figura 4: Diagrama lógico o de flujo empleado en el diseño del receptor de transacciones MDIO

3.1.2. Para resetear y continuar al siguiente caso

Este caso se realizó en seguida del anterior. Bastó únicamente con poner en bajo la señal de rst, se hizo por 4 ciclos de reloj, lo que inhalitaría el funcionamiento normal del generador y pondría los valores bajos en las salidas y los iniciales en los registros e internas.

3.1.3. Para transacción de Lectura (bits 3 y 4 del frame de MDIO: 10)

Este comprendió el caso más complejo para el generador, nuevamente por la misma funcionalidad lógica de este componente (apreciable en la figura 2). Sin embargo, se debió poner en alto rst, especificar T_DATA, formar el pulso de MDIO_START y, por último, pasar bit a bit, por ciclo de reloj, los últimos 16 bits de T_DATA en MDIO_IN. Propiamente, se utilizó:

```
// Se define T_DATA con el tercer y cuarto bit en 10
T_DATA = 32'b1010_0101_0011_0110_1010_1110_0101_0011;
```

Código 2: Prueba de lectura en el generador.

3.2. Receptor de transacciones MDIO

El plan de pruebas para el receptor, al igual que con el generador, se especificó en su respectivo archivo tester.v, mismo que consta de 3 casos. Estos casos se especifican a continuación.

3.2.1. Para transacción de Lectura (bits 3 y 4 del frame de MDIO: 10)

Para la transacción de lectura de MDIO a evaluar, se siguió la funcionalidad lógica de este componente (apreciable en la figura 4). Por tanto, acá se debía haber inicializado el MDC y haber puesto el rst en alto. Seguidamente, se debió especificar bit a bit los 32 bits del frame básico de MDIO. El acá utilizado se visualiza por simplicidad como:

```
// Se define MDIO_OUT con el tercer y cuarto bit en 10
    // MDIO_OUT = 32'b0110_0101_1000_0000_0000_0000_0000_0000; CASO PHY, es el que se
       utiliza
```

Código 3: Prueba de lectura en receptor.

Además, se debió especificar RD_DATA y se puso la entrada MDIO_OE en alto durante 16 ciclos de MDC y 16 en bajo, tras tratarse de tipo de transacción de lectura.

```
// Informacion de RD_DATA
    RD_DATA = 16'b1100_1001_0000_0001;

    // MDIO_OE segun LECTURA
    MDIO_OE=1;
    #64;
    MDIO_OE=0;
    #64;
```

Código 4: Prueba de lectura en receptor.

3.2.2. Para resetear y continuar al siguiente caso

Este caso se realizó en seguida del anterior. Bastó únicamente con poner en bajo la señal de rst, se hizo por 2 ciclos de MDC, lo que inhalitaría el funcionamiento normal del receptor y pondría los valores bajos en las salidas y los iniciales en los registros e internas.

3.2.3. Para transacción de Escritura (bits 3 y 4 del frame de MDIO: 01)

Para la transacción de escritura de MDIO a evaluar, se siguió la funcionalidad lógica de este componente (apreciable en la figura 4). Por tanto, acá se debía haber inicializado el MDC y haber puesto el rst en alto. Seguidamente, se debió especificar bit a bit los 32 bits del frame básico de MDIO. El acá utilizado se visualiza por simplicidad como:

```
// Se define MDIO_OUT con el tercer y cuarto bit en 01
    // MDIO_OUT = 32'b0101_1101_1000_0000_1000_1000_1000_0001; CASO PHY, caso utilizado
```

Código 5: Prueba de escritura en receptor.

Además, se puso la entrada MDIO_OE en alto durante 32 ciclos de MDC, tras tratarse de tipo de transacción de escritura.

```
// MDIO_OE segun ESCRITURA
    MDIO_OE=1;
    #64;
    MDIO_OE=1;
    #64;
    MDIO_OE=0;
```

Código 6: Prueba de escritura en receptor.

4. Instrucciones de utilización de la simulación

Como primera aclaración conviene mencionar que se trabajó en la aplicación Visual Studio Code con las extensiones de Verilog. Además, se utilizó la herramienta GTKWave para visualizar las señales u ondas generadas por el programa.

Para la ejecución de los diseños, una vez creados los archivos .v denominados *Trans_MDIO.v*, *Recep_MDIO.v*, *Tester_Trans.v*, *Tester_Recep.v*, *Testbench_Trans.v* y *Testbench_Recep.v*, se realizó un archivo makefile para cada parte, cuyo contenido se visualiza como en la figura 5.

Generador_MDIO > M makefile

```
1  verilog:
2      iverilog -o tb.vvp Testbench_Trans.v
3      vvp tb.vvp
4      gtkwave tb.vcd
5
6  .PHONY: clean
7  clean:
8      rm -rf tb.vvp tb.vcd
```

(a) Generador

Receptor_MDIO > M makefile

```
1  verilog:
2      iverilog -o tb.vvp Testbench_Recep.v
3      vvp tb.vvp
4      gtkwave tb.vcd
5
6  .PHONY: clean
7  clean:
8      rm -rf tb.vvp tb.vcd
```

(b) Receptor

Figura 5: Contenido de cada Makefile, el del Generador y el del Receptor

De esta forma, se tuvo la precaución de que los .v y su respectivo makefile estuvieran guardados en la misma dirección o carpeta. Además, el archivo visualizado en la figura 5 cuenta con las banderas *verilog* y *clean*. La bandera Verilog contiene las instrucciones para correr GTKWave en Linux. Con la línea 2 se logra compilar el código .v del testbench y se genera el ejecutable *tb.vvp* que es el archivo en donde se guardará la salida del compilador. En la tercera línea se ejecuta el .vvp, es decir, la simulación del programa codeado en Verilog. Por último, su cuarta línea abre GTKWave y es allí donde se carga el archivo *tb.vcd*, que es el que contiene las formas de onda generadas en la simulación, con el propósito de poderlas visualizar y analizar. La bandera clean se utiliza para limpiar los archivos generados por la compilación y simulación, es decir, en este caso, los archivos tb.vvp y tb.vcd. Esta bandera es útil para cuando se quiere limpiar el directorio antes de realizar una nueva compilación o ejecución del programa. Además, se escribe antes de esta bandera un *.PHONY: clean* que conlleva a que se ejecuten de inmediato las instrucciones o acciones definidas en la misma.

Una vez con todo lo anterior listo, se abre la terminal y se dirige al folder donde están guardados los archivos .v y makefile de la parte del proyecto que se desea correr (Generador o Receptor). Así, una vez estando allí, solamente bastará escribir make para que se corran las instrucciones descritas en el makefile. Inclusive, se puede especificar make clean o make verilog para que se corran únicamente las instrucciones de esas banderas incluidas en el makefile.

5. Análisis y Resultados

Tras la realización de los dos primeros casos especificados en el plan de pruebas para el generador, correspondientes a la transacción de escritura y reset, se obtiene el waveform de la figura 6.

5.1. Generador de transacciones MDIO

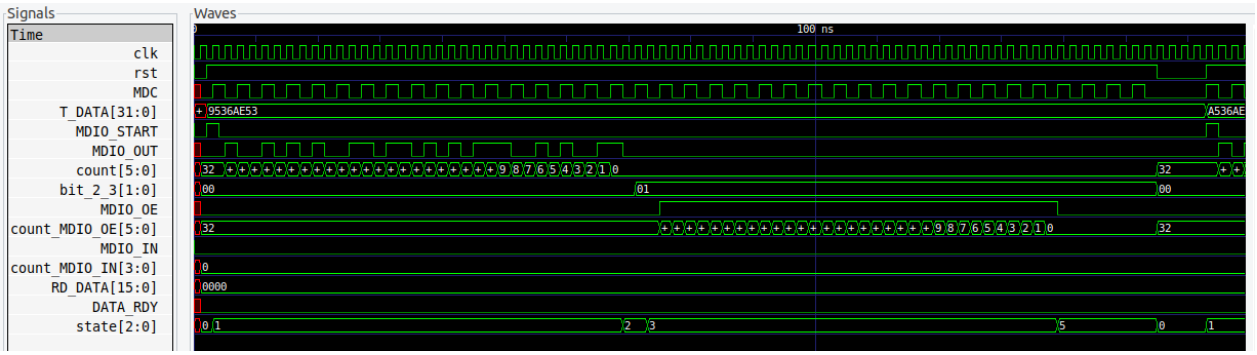

Figura 6: Waveforms del caso de escritura y Reset del Generador

En la figura 6, se nota que el reloj (clk) está funcionando correctamente, proporcionando una señal periódica, de 2 ns por ciclo, y continua esencial para la sincronización de todas las operaciones. Asimismo, la señal de reinicio (rst) se observa en alto, indicando que el sistema no está en modo de reinicio, permitiendo la operación normal del generador de transacciones; al tiempo que, la señal de reloj para el receptor, MDC, muestra correctamente la secuencia periódica que sincroniza las transacciones de datos MDIO, misma que cumple con ser de la mitad de la frecuencia que clk, o lo que equivalente, es del doble de tiempo que clk.

Los datos de la transacción (T_DATA[31:0]) se presentan en una forma específica, mostrando el valor que se está transmitiendo en la transacción, tal cual se especificó en el tester. La señal MDIO_START se activa, indicando el inicio de una transacción MDIO, ya que T_DATA[31:0] fue ingresada. Mientras que, la señal de salida de MDIO (MDIO_OUT) muestra la transmisión de bits de datos de manera secuencial, según lo ingresado en MDIO_OUT, es decir, acá los datos se transmiten bit a bit conforme avanza la transacción. Este control se realiza mediante el contador (count[5:0]), mismo que se incrementa en cada ciclo de clk, llevando un seguimiento del número de bits transmitidos, lo cual resulta crucial para manejar la longitud fija de las transacciones MDIO.

Luego, la señal bit_2_3[1:0] indica los valores concatenados de los bits 3 y 4 del frame, donde para escritura cumple satisfactoriamente con ser de 01. La señal de habilitación de salida (MDIO_OE) se mantiene activa durante los ciclos de la transacción (los 32 al tratarse de escritura), permitiendo que MDIO_OUT transmita datos. Al mismo tiempo que, el contador asociado a MDIO_OE también incrementa, manteniendo la sincronización con la transmisión de datos para controlar la duración de la habilitación de la salida. También, la señal MDIO_IN se muestra preparada para recibir datos, aunque en este caso no se activa, ya que se está analizando un caso de escritura. Caso similar con su contador de MDIO_IN, que se mantiene en cero, lo que es consistente con una operación de escritura.

Luego, la señal de los datos leídos (RD_DATA[15:0]), que debieran mostrar los valores que se están procesando en el sistema durante la transacción, se mantiene en cero, consistente con lo esperado para una transacción de escritura. Al tiempo que, la señal de datos listos (DATA_RDY) también se mantiene en bajo, tras no haberse cambiado RD_DATA[15:0].

Finalmente, en consecuente con todo lo anterior sucedido en la FSM, la señal de estado (state[2:0]) muestra los diferentes estados del sistema durante la transacción MDIO. Además, su flujo o cambio se comporta de acuerdo a lo

esperado, recalcando una simulación que muestra que el sistema está realizando una transacción de escritura MDIO de manera correcta.

Es decir que acá se obtuvo que, las señales de control y datos están sincronizadas y operan según lo esperado. La señal MDIO_START inicia la transacción, y MDIO_OUT transmite los datos bit a bit de forma sincrónica. La habilitación de salida MDIO_OE está correctamente gestionada, y los contadores (count y count_MDIO_OE) aseguran la correcta secuenciación de los bits transmitidos. Además, la señal de estado (state[2:0]) indica que el sistema está en el estado apropiado. Por tanto, el sistema está funcionando según lo esperado en un caso de escritura, con todas las señales clave operando de manera coordinada y sin errores evidentes en la transmisión de datos.

Similarmente, se nota al extremo derecho de la figura 6 el caso de prueba 2, que es el reseteo. Donde acá se pone en bajo la señal de reset, indicando la inhabilitación del generador y se nota que, mientras que se mantiene así, las señales de salida y las internas, pasan a sus respectivos valores iniciales correctamente.

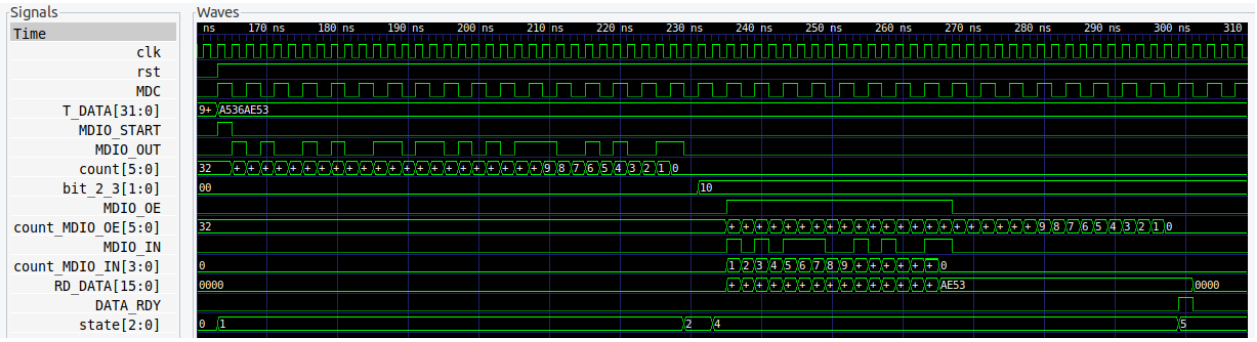

Figura 7: Waveforms del caso de lectura del Generador

Para el tercer caso de prueba, el de lectura, su waveform se muestra en la figura 7, en la cual se muestra la señal de reloj (clk), tal cual en los casos anteriores. Nuevamente, la señal de reinicio (rst) se observa en alto, indicando que el sistema no está en modo de reinicio, permitiendo la operación normal del generador de transacciones. Además, la señal MDC conserva su correcto funcionamiento.

También, se nota que los datos de la transacción (T_DATA[31:0]) se presentan en la forma especificada en la sección de plan de pruebas para este caso correspondiente, describiéndose acá el valor que se está transmitiendo en la transacción. La señal MDIO_START se activa, indicando el inicio de una transacción MDIO, ya que T_DATA[31:0] fue ingresada. Mientras que, la señal de salida de MDIO (MDIO_OUT) muestra la transmisión de bits de datos de manera secuencial, según lo ingresado en MDIO_OUT, es decir, acá los datos se transmiten bit a bit conforme avanza la transacción. Este control se realiza mediante el contador (count[5:0]), mismo que se incrementa en cada ciclo de clk, llevando un seguimiento del número de bits transmitidos, lo cual resulta crucial para manejar la longitud fija de las transacciones MDIO.

Luego, la señal bit_2_3[1:0] indica los valores concatenados de los bits 3 y 4 del frame, donde para lectura cumple satisfactoriamente con ser de 10. Asimismo, se observa que la señal de habilitación de salida (MDIO_OE) se mantiene activa durante los primeros 16 ciclos de la transacción y en bajo durante los siguientes 16, lo cual es lo esperado para este tipo de transacción. Al mismo tiempo que, el contador asociado a MDIO_OE también incrementa, manteniendo la sincronización con la transmisión de datos para controlar la duración de la habilitación de la salida. También, la señal MDIO_IN se muestra preparada para recibir datos, lo cual sucede cuando MDIO_IN se encuentra en alto, posterior a haberse enviado todos los datos por MDIO_OUT. Esto ya que se está analizando un caso de lectura, donde en los primeros 16 ciclos de la transacción, se espera que la data sea enviada en esta señal.

Seguidamente, en los datos leídos (RD_DATA[15:0]) se muestran los valores que se están procesando en el sistema durante la transacción. En este caso, los datos AE53 están siendo leídos y procesados. La señal de datos listos (DATA_RDY) se activa, indicando que los datos de la transacción están listos para ser utilizados o procesados. A su vez que, en cada flujo por la FSM, se nota que la señal de estado (state[2:0]) muestra los diferentes estados del sistema durante la transacción MDIO.

Por ende, la simulación muestra que el sistema está realizando una transacción de lectura MDIO de manera correcta. Donde sobresale el rol de la señal MDIO_IN, misma que recibe los datos leídos, que se muestran en RD_DATA[15:0] como AE53, y DATA_RDY indica que los datos están listos para ser procesados al activarse.

5.2. Receptor de transacciones MDIO

Para el plan de pruebas para el receptor, correspondientes a la transacción de lectura y reset, se obtiene el waveform de la figura 8.

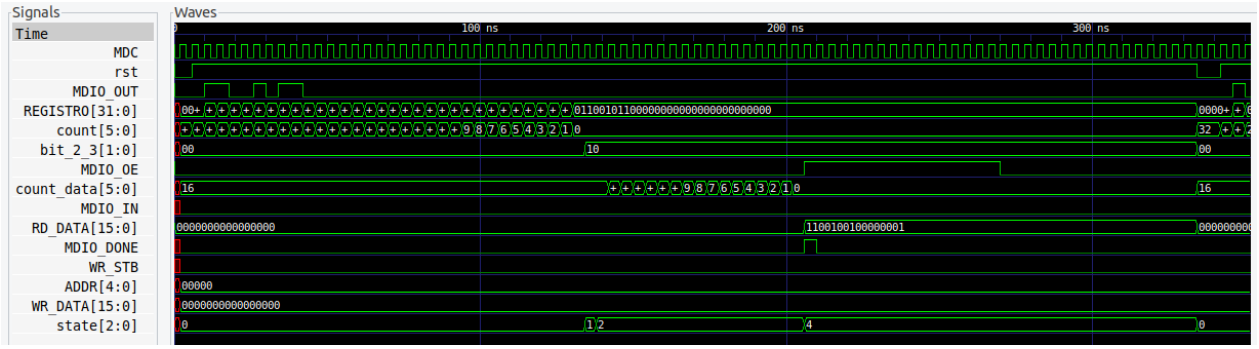

Figura 8: Waveforms del caso de lectura y Reset del Receptor

En la figura 8, se observa MDC (Management Data Clock) que es la señal de reloj utilizada para sincronizar la transmisión de datos en el bus MDIO, siendo de 4 ns por ciclo, lo que es el doble del clk del generador. En la forma de onda, se observa un tren de pulsos periódicos que indican su funcionamiento continuo. El rst (reset) es la señal que se utiliza para inicializar o reiniciar el sistema. Al observar la forma de onda, se nota que el reset está activo, poniendo el sistema en su estado inicial para ejecutar la transacción de MDIO.

También, MDIO_OUT representa los datos que están siendo transmitidos desde el generador hacia el dispositivo gestionado. En este caso, observamos un patrón de bits que comienza a transmitirse después de que la señal de reset ha sido activada. El REGISTRO[31:0] muestra el contenido del registro que se está leyendo o escribiendo. Específicamente, se configura para que los valores en la forma de onda de este registro se muestren en binario, con lo cual resulta más sencillo notar que corresponde satisfactoriamente a la transacción en curso y al MDIO_OUT ingresado. Para count[5:0] y count_data[5:0] se tiene que son señales de conteo son usadas para los bits transmitidos y recibidos, respectivamente. Notamos que los contadores avanzan conforme se transmiten o reciben los bits en el bus MDIO. El registro bit_2_3[1:0] está relacionado nuevamente con el estado o control interno del módulo, donde acá coincide con ser de 10 al tratarse de una transacción de lectura.

El MDIO_OE (Output Enable) es una señal que controla si el controlador está habilitado para enviar datos en el bus MDIO. Se observa que la señal se activa cuando se están transmitiendo datos. El MDIO_IN representa los datos que se reciben del dispositivo gestionado. Los bits recibidos se muestran claramente en la forma de onda, coincidiendo con el patrón esperado de la lectura. El RD_DATA[15:0] comprende los datos leídos del dispositivo gestionado, mismos que son almacenados en esta señal. La forma de onda muestra cómo los bits se acumulan conforme se reciben. MDIO_DONE es una señal que indica la finalización de la transacción MDIO. Se activa al finalizar la lectura, permitiendo que el sistema proceda con otras operaciones.

También, se muestran las señales de WR_STB (Write Strobe) y WR_DATA[15:0], mismas que cumplen con permanecer en bajo durante esta transacción precisamente al no ser de escritura. Caso similar sucede con la dirección del registro ADDR[4:0], en la que se desea escribir en la memoria, mas para la lectura no tiene relevancia alguna. Por último, la señal de state[2:0] indica el estado actual de la máquina de estados que controla la transacción. Los cambios en esta señal reflejan el progreso de la transacción MDIO desde el inicio hasta la finalización.

Concretamente, el proceso de lectura comienza con la activación de la señal de reloj MDC y la activación del reset. A continuación, se inicia la transmisión de un patrón predefinido en MDIO_OUT, que incluye la dirección del registro y el comando de lectura. La señal MDIO_OE se activa para permitir la transmisión de datos. Conforme avanza el conteo en count, los bits de MDIO_OUT son transmitidos y luego se recibe la respuesta del dispositivo gestionado en MDIO_IN. Al completarse la lectura, la señal MDIO_DONE se activa indicando la finalización de la transacción y, mientras WR_STB esté en bajo, la entrada RD_DATA toma el valor distinto de cero especificado en el tester, tal cual lo esperado.

Seguidamente, en el segundo caso de prueba, mostrado al final de la figura 8, cuando se da la puesta en baja del reset, todas las señales se reinician, estableciendo un estado conocido inicial y preparado para iniciar una nueva transacción, lo cual también es lo esperado.

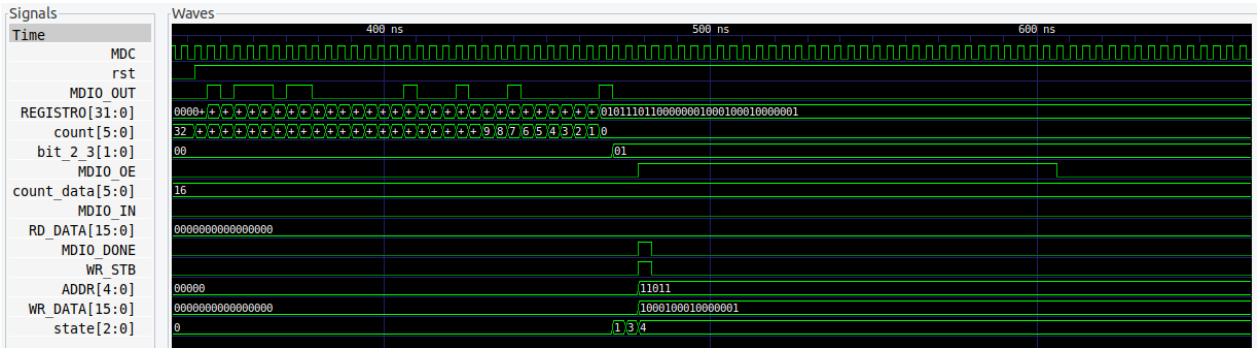

Figura 9: Waveforms del caso de escritura en memoria del Receptor

Por último, se tiene el caso de escritura en el receptor, mismo que se muestra en la figura 9. En esta, se aprecia nuevamente el correcto funcionamiento de MDC y que el rst (reset) se enuntra en alto, lo que habilita la ejecución de esta nueva transacción. MDIO_OUT y MDIO_IN son señales que representan la línea de datos bidireccional del bus MDIO. Durante la escritura, el controlador envía datos al PHY a través de MDIO_OUT. La línea MDIO_IN se utiliza cuando se necesita leer datos desde el PHY. Para el REGISTRO[31:0] se tiene que la señal muestra el registro de datos que se pretende escribir. Los bits se alinean de acuerdo con el formato del protocolo MDIO, que incluye una preámbulo, una secuencia de inicio, una operación de código, la dirección del PHY, la dirección del registro, y los datos. Además, el count[5:0] y count_data[5:0] son señales de conteo ayudan a rastrear el número de ciclos de reloj necesarios para completar la transacción. La correcta secuenciación es importante para la integridad de los datos transferidos. Con el bit_2_3[1:0] se puede saber si está en modo escritura o modo lectura, donde acá coincide con ser de 01 al tratarse de una transacción de escritura.

MDIO_OE (Output Enable) es una señal habilita la línea de salida MDIO para la transmisión de datos. Durante una operación de escritura, esta señal se activa para permitir que los datos se transmitan desde el controlador al PHY. La MDIO_DONE indica la finalización de la transacción de MDIO. Cuando se activa, significa que la escritura o lectura ha sido completada correctamente. El RD_DATA[15:0] es una señal que se utiliza para almacenar los datos leídos desde el PHY. En una operación de escritura, como la analizada en este caso, esta señal no se utiliza directamente, pero es crucial para las operaciones de lectura. Sin embargo, acá entran en relevancia las señales de escritura, mismas que abarcan WR_STB (Write Strobe), ADDR[4:0] y WR_DATA[15:0].

WR_STB (Write Strobe) es la señal que se utiliza para indicar un evento de escritura; misma que en este caso, se activa para señalar que los datos deben escribirse en el PHY. El ADDR[4:0] es una señal que muestra la dirección del registro en el PHY donde los datos serán escritos; mientras que, para WR_DATA[15:0], se tiene que contiene los datos que se escribirán en el registro del PHY. La secuencia de bits debe coincidir con los datos esperados en el protocolo MDIO.

Con todo lo acá descrito, se llega a que lo de la figura 9, correspondiente a las formas de onda del receptor para el caso 3, proporciona una visión detallada del proceso de escritura en el protocolo MDIO. En ella, cada señal tiene un papel específico en la correcta implementación y operación del protocolo, asegurando que los datos se transfieran de manera precisa y confiable entre el controlador y el dispositivo PHY. Además, es notorio que la correcta secuencia y sincronización de estas señales es requerido para el éxito de la operación de escritura, lo cual se logra a través de los cambios de estado en la FSM.

6. Conclusiones y Recomendaciones

- Objetivos cumplidos: Todos los objetivos planteados al inicio del proyecto fueron alcanzados con éxito. Esto incluye la implementación correcta de los módulos y la verificación de su funcionalidad mediante pruebas.
- Eficiencia del diseño: El diseño de los módulos demostró ser eficiente y robusto, cumpliendo con los requisitos de rendimiento establecidos. Las pruebas realizadas confirmaron la correcta operación y sincronización entre los diferentes componentes del sistema.
- Aprendizaje y desarrollo: El proyecto permitió un aprendizaje significativo en el área de diseño de sistemas digitales y la implementación de protocolos de comunicación. La experiencia adquirida en la resolución de problemas y en la adaptación del diseño a nuevas necesidades fue muy valiosa.
- Trabajo en equipo: La colaboración entre los miembros del equipo fue fundamental para el éxito del proyecto. La comunicación efectiva y la distribución de tareas permitieron un flujo de trabajo continuo y eficiente.
- Desarrollo del proyecto: En el desarrollo del proyecto parte de los desafíos era comprender realmente el comportamiento de cada una de las señales, cómo se relacionaban entre sí, en qué casos se activan o no. Buscar información sobre el protocolo y consultar dentro de lo posible a la profesora del curso fue de gran ayuda en el proceso.
- Posibilidad de casos establecidos: Parte de lo que podría ayudar a cumplir los objetivos en plazo de tiempo, es el tener visualmente ejemplos de pruebas ya realizadas. Como caso uno y caso 2, el comportamiento visible en waveform. Sin embargo, el proceso de ser nosotros quienes debamos experimentar para llegar a los objetivos o no, académicamente es bueno. Pero, tener casos visibles a los cuales llegar podría ser una forma de garantizar ciertos objetivos, que probablemente ayuden también a la hora de evaluar.
- Recomendaciones futuras: Se identificaron áreas de mejora que pueden ser abordadas en futuros trabajos. Entre ellas, se sugiere optimizar ciertos aspectos del diseño para mejorar aún más su eficiencia y explorar nuevas funcionalidades que podrían ser añadidas al sistema. También, se acató la instrucción de cambio de especificaciones, con lo cual no se unieron la Parte I y Parte II. Sería enriquecedor académicamente, ver las posibilidades de cómo realizar dicha conexión o unión de las partes.