Universidad de Costa Rica

Escuela de Ingeniería Eléctrica

IE-0523 Circuitos Digitales II

Tarea 4

Profesor:
Enrique Coen Alfaro

Alumno:
Diego Alfaro Segura C20259

24 de octubre de 2024

1. Resumen

En el presente diseño se propone una arquitectura y lógica para dos módulos que emplean el protocolo de comunicación serial I2C, donde uno de estos es un generador (llamado Generador) que inicia y dirige la comunicación al otro (llamado Receptor) para ejecutar transacciones de lectura y escritura de registros. Se escribieron estos módulos en el lenguaje de descripción de hardware Verilog y se realizaron diversas simulaciones para verificar el funcionamiento de los mismo. Se siguieron las especificaciones del protocolo en su mayoría, con algunas excepciones como el hecho que en el código se manejó la línea bidireccional con 3 conexiones distintas (SDA_OE, SDA_OUT y SDA_IN) según una configuración de compuertas CMOS para conexiones bidireccionales. La conexión de ambos módulos en conjunto permite ejecutar operaciones de lectura/escritura de un registro correspondiente al receptor. Durante las pruebas realizadas se logró verificar que los módulos desarrollados cumplen con las acciones de lectura/escritura, se ignoran transacciones si la dirección no corresponde al receptor y que la señal de Reset reinicia el sistema en su totalidad.

2. Descripción Arquitectónica

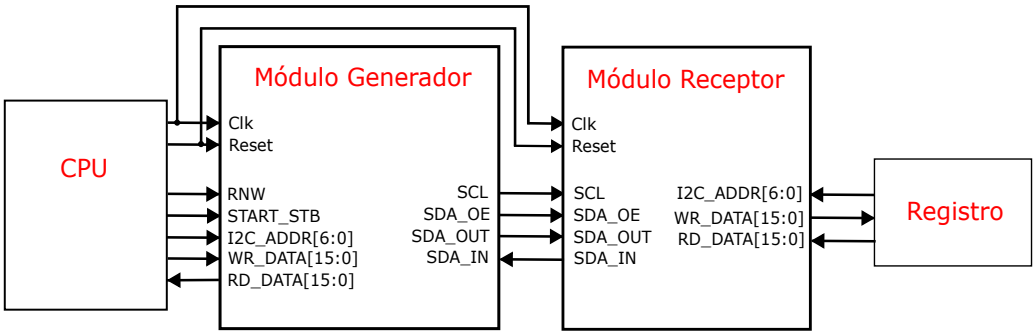

Figura 1: Diagrama de bloques de la estructura Generador-Receptor diseñada.

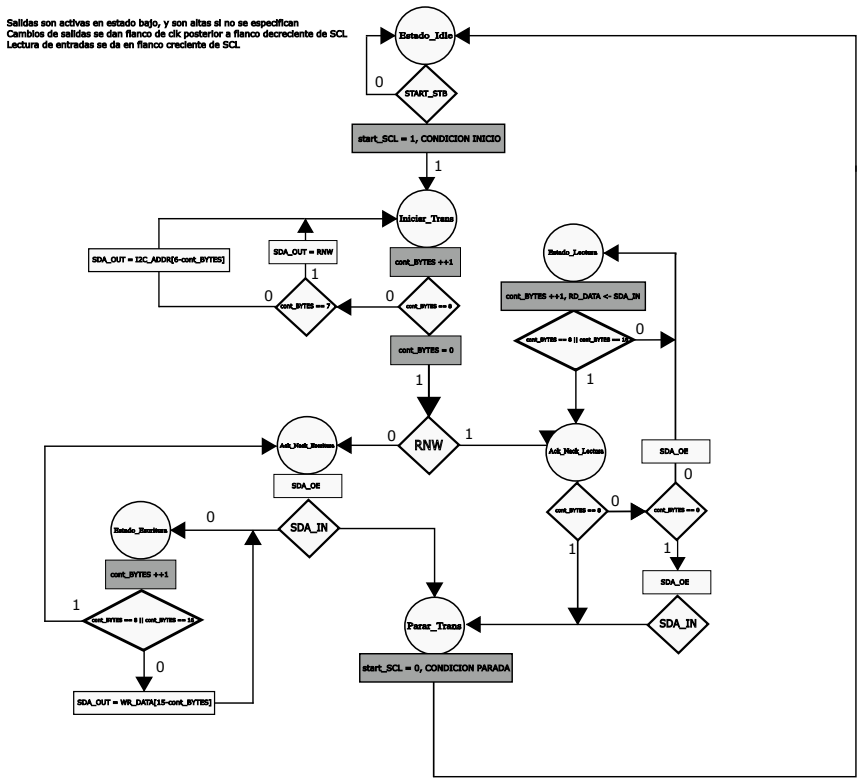

Figura 2: Diagrama ASM de la máquina de estados del generador.

En la Figura 1 se muestra la estructura general de la conexión entre los módulos generador y receptor con el CPU y los registros internos del dispositivo periférico. Se denota que los dos módulos comparten las señales de Reloj y Reset provenientes del CPU para mantenerse sincrónicos, pero que la principal comunicación entre ambos se da por el conjunto de señales correspondientes a la comunicación serial. El CPU dirige las transacciones indicando cuando inician (START_STB), la dirección en memoria del registro (I2C_ADDR), el tipo de transacción (RNW) y los datos a escribir (WR_DATA), mientras que recibe los datos leídos del registro (RD_DATA). El registro contiene y provee la dirección correspondiente al mismo (I2C_ADDR), los datos a leer (RD_DATA) y recibe los datos a escribir en el registro (WR_DATA).

Luego, en la Figura 2 se muestra el diagrama ASM realizado para la máquina de estados del módulo Generador. No se dibujó el diagrama del receptor, pero este sigue una estructura muy similar. Se usa el formato usual de dicho diagrama, con la excepción que los cambios a contadores/registros internos se muestran como cajas pintadas con gris, esto para diferenciarlos de las salidas.

2.1. Declaración de estados

Los estados asignados son los siguientes, se muestran los estados equivalentes para el módulo receptor y sus funciones:

Estado	Generador	Receptor	Próximo Estado
Estado_Idle	Espera que se de el START_STB para generar la condición de inicio (Bajar SDA_OUT con SCL en alto).	Espera a la condición de inicio proveniente de la comunicación serial.	Se pasa a Iniciar_Trans si se cumple condición de inicio.
Iniciar_Trans	Envía el primer byte con la dirección I2C_ADDR dada por el CPU y el RNW como último bit.	Recibe y compara la dirección dada por el generador con su dirección interna.	Se pasa a Ack_Nack_Lectura si RNW es 1, sino se pasa a Ack_Nack_Escritura. Si la dirección dada no corresponde a la del receptor, este pasa a Parar_Trans.
Ack_Nack_Lectura	Si es el primer byte espera el ACK/NACK del receptor, si es el último genera un NACK, sino genera un ACK.	Si es el primer byte genera un ACK, sino lee los ACK/NACK del generador para cambiar de estado.	Si se recibe/genera un NACK se pasa al estado Parar_Trans, sino se pasa al Estado_Lectura.
Estado_Lectura	Se reciben bits por medio de la comunicación serial y se agregan al registro de salida RD_DATA.	Se leen los bits del registro RD_DATA y se envían por la comunicación serial.	Si se termina de leer/enviar un byte, se pasa al estado Ack_Nack_Lectura.
Ack_Nack_Escritura	Se espera el ACK/NACK del receptor.	Si es el ultimo byte genera un NACK, sino genera un ACK.	Si se recibe/genera un NACK se pasa al estado Parar_Trans, sino se pasa al Estado_Escritura.
Estado_Escritura	Se envían los bits indicados por el CPU (WR_DATA) por medio de la comunicación serial.	Se leen los bits enviados por la comunicación serial y se agregan al registro de escritura (WR_DATA).	Si se termina de leer/enviar un byte, se pasa al estado Ack_Nack_Escritura.
Parar_Trans	Genera la condición de STOP (poniendo SDA_OUT en alto mientras que SCL está en alto), reinicia contadores y detiene SCL.	Espera la condición STOP, mantiene SDA_IN en alto y reinicia contadores.	Se pasa al Estado_Idle.

Se diseñaron los módulos de tal manera que al iniciar una transacción y el generador envíe la dirección especificada, uno o varios receptores puedan recibir esta instrucción. Si la dirección corresponde al receptor este genera un ACK (pone SDA_IN como bajo) y genera los cambios respectivos a la transacción, sino deja el SDA_IN en estado alto e ignora las instrucciones, esperando a la condición de STOP. Si el generador recibe un ACK continúa dando las instrucciones (escuchadas por el receptor acorde), sino genera la condición de STOP.

Además, se diseñó de tal manera que se dieran los cambios de salidas durante el estado bajo de SCL, la lectura de entradas en el flanco creciente de SCL y las transiciones de estados se dieran de tal manera que no se salten/acorten los periodos de comunicación establecidos por el protocolo. Se creó un registro de contador de bits para asegurar que se envíen solo 8 bits por ciclo de comunicación.

Los estados se codificaron de la siguiente manera (Tabla 1), con el objetivo de asignar un flip-flop a cada estado y así reducir la posibilidad de que se den condiciones de carrera durante la operación del chip. Se agrega el equivalente de dicha codificación en hexadecimal para tener mayor facilidad de lectura de los estados en los diagramas de tiempo de la Sección 5. Se usó la misma codificación para ambos módulos.

Cuadro 1: Asignación de estados

Estado	Codificación	Equivalente en hexadecimal
Estado_Idle	0000001	01
Iniciar_Trans	0000010	02
Ack_Nack_Lectura	0000100	04
Estado_Lectura	0001000	08
Ack_Nack_Escritura	0010000	10
Estado_Escritura	0100000	20
Parar_Trans	1000000	40

La estructura del código describe una máquina de estados que emplea flip-flops para actualizar los estados en el flanco creciente de la señal de reloj, al igual que las salidas. Además, se configuró la señal de Reset de tal manera que tome acción en estado bajo según su flanco decreciente.

3. Plan de pruebas

Para verificar el funcionamiento típico se emplearon cuatro tipos de pruebas que cubren las situaciones generales que pueden darse durante su operación normal. Se asignó el address del registro como 59 en 7 bits (0111011), los datos de lectura en el registro como 1100110011001100 y los datos a escribir como 1010101010101010.

3.1. Prueba #1, Transacción de lectura.

Se comprueba que se puede ingresar el address correcto, la instrucción de lectura con el RNW e iniciar la transacción con el START_STB el depósito. Se espera de esta prueba que se de la transición de estados $Estado\_Idle \Rightarrow Iniciar\_Trans \Rightarrow Ack\_Nack\_Lectura \Rightarrow Estado\_Lectura \Rightarrow Ack\_Nack\_Lectura \Rightarrow Estado\_Lectura \Rightarrow Ack\_Nack\_Lectura \Rightarrow Parar\_Trans \Rightarrow Estado\_Idle$. Durante estos se espera que se siga la secuencia de salidas vista en la Figura 3, con las salidas del primer byte correspondiendo al I2C_ADDR y el RNW concatenados y los bloques de DATA siendo los bytes del RD_DATA guardados en el receptor. Además se espera que se actualicen correctamente los contadores de los bits de lectura en el Generador (RD_DATA), los contadores de bytes y todo acorde a la señal SCL generada. **Note que las transiciones de estados son para los cambios de estados, no necesariamente representan ciclos individuales.**

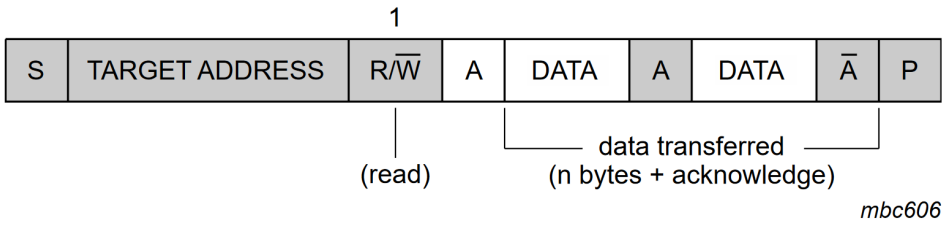

Figura 3: Diagrama de secuencia esperada para una lectura.

3.2. Prueba #2, Transacción de escritura.

Se comprueba que se puede ingresar el address correcto, con RNW puesto para realizar la transacción de escritura. Se espera de esta prueba que se de la transición de estados $Estado\_Idle \Rightarrow Iniciar\_Trans \Rightarrow Ack\_Nack\_Escritura \Rightarrow Estado\_Escritura \Rightarrow Ack\_Nack\_Escritura \Rightarrow Estado\_Escritura \Rightarrow Ack\_Nack\_Escritura \Rightarrow Parar\_Trans \Rightarrow Estado\_Idle$. Se espera que se siga la secuencia de salidas vista en la Figura 4, con las salidas del primer byte correspondiendo al I2C_ADDR y el RNW concatenados y los bloques de DATA siendo los bytes

del WR_DATA indicados por el CPU para escritura. Se espera que el registro interno para escritura del receptor WR_DATA se actualice y termine siendo igual que el indicado por el CPU.

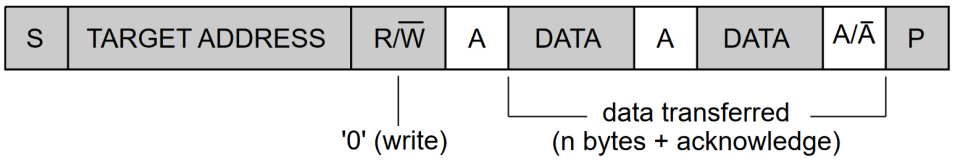

Figura 4: Diagrama de secuencia esperada para una escritura.

3.3. Prueba #3, Respuesta de receptor con address distinto al asignado

Se comprueba que al ingresar un address incorrecto el receptor responde con generar un NACK e ignora las demás instrucciones del generador. Se espera que en el módulo generador se de la transición de estados $Estado\_Idle \Rightarrow Iniciar\_Trans \Rightarrow Ack\_Nack\_Escritura \Rightarrow Parar\_Trans \Rightarrow Estado\_Idle$, mientras que en el módulo receptor se de la transición $Estado\_Idle \Rightarrow Iniciar\_Trans \Rightarrow Parar\_Trans \Rightarrow Estado\_Idle$. Se espera que el receptor genere un NACK luego de que el generador envíe el primer byte, y así se finalice la transmisión. Además, se espera que el receptor entre en un estado en el que si se continuara la transmisión de datos no cambiaría sus registros ni generaría salidas.

3.4. Prueba #4, Interrupción con Reset

Se comprueba que si se genera la señal Reset se reinician los contadores y todas las señales de comunicación serial vuelven al estado acorde a lo esperado del protocolo. Se espera que se de una transacción normal hasta que se aplique el Reset, punto en el cual todos los registros se reinician, se detiene SCL y se vuelve al estado predeterminado de cada salida serial.

4. Instrucciones de utilización de la simulación

Para repetir las simulaciones utilizadas se incluye un archivo de Makefile que permite correr los comandos necesarios para realizarlas. Se debe tener instalado las herramientas de software Icarus Verilog (compilador de verilog), GTKWave (visualizador de ondas) y GNU-make o mingw Make (comandos para ejecutar un Makefile). Además, se deben tener los archivos incluidos del diseño (tester.v, testbench.v, Generador.v y Receptor.v), el archivo Makefile en el mismo directorio que estos archivos. Se incluyen 2 comandos, notar que si se emplea la versión de make de mingw se debe escribir "mingw32-make" en vez de solo "make". **En este caso solo se puede emplear el GNU-make por las restricciones de Yosys**.

Comandos incluidos en el Makefile:

- make tarea Corre la simulación al compilar el archivo y mostrar los resultados en GTKWave, está configurado para mostrar todas las ondas importantes según se muestra en los resultados.
- make clean Elimina los archivos generados para limpiar el directorio (no elimina los de código ni el Makefile).

5. Ejemplos de los resultados

A través de las simulaciones se obtuvieron los resultados mostrados en los siguientes diagramas de tiempos. Todos estos provienen de una sola simulación, pero se separa en distintas imágenes, una para cada una de las pruebas realizadas (los tiempos de las imágenes son distintos a los de la simulación pues se separaron las pruebas a la hora de tomar las capturas). En general los resultados de cada una de las pruebas cumplieron las expectativas descritas en la Sección 3, incluyendo las transiciones de estados y las salidas generadas. Para los cambios de estados es útil consultar la Tabla 1. Debido a esto solo se resaltan los puntos clave de los resultados.

En la Figura 5 se muestra como el reloj generado SCL posee el 25 % de la frecuencia del reloj Clk, pues en el tiempo que Clk hace 4 ciclos SCL hace 1. Junto a esta señal se muestran en conjunto las de comunicación serial. También se muestra como las señales poseen valor alto por defecto y que los registros definidos por el CPU se definen desde el inicio. Además, debido a que algunas entradas poseen el mismo nombre, pero según la descripción no deben ser las mismas (como RD_DATA y WR_DATA) se agrega una señal distinta para cada módulo.

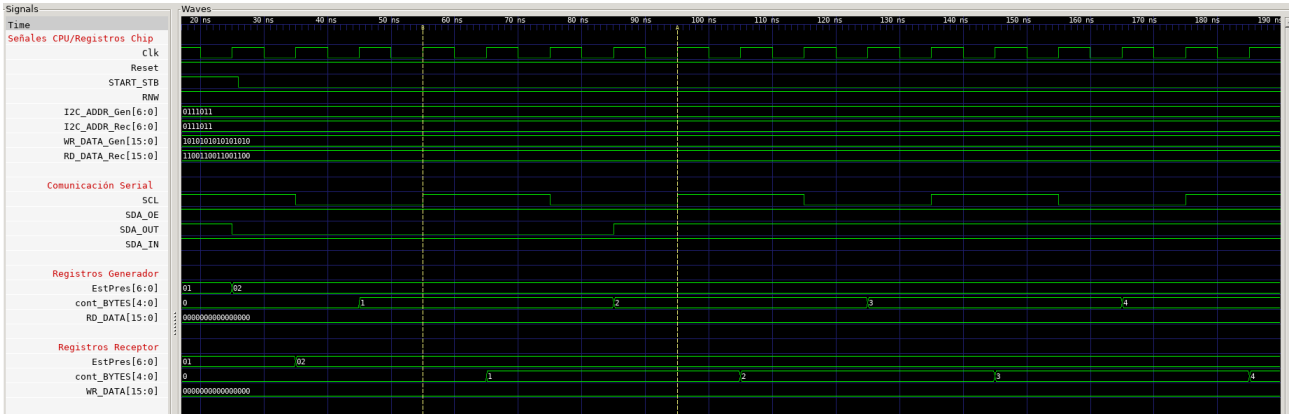

Figura 5: Resultados generales de las simulaciones.

5.1. Resultados prueba #1

En la Figura 6 se muestra la transacción de lectura. Se nota como posterior a la señal de inicio de transacción (START_STB) el generador envía los primeros 7 bits del I2C_ADDR por SDA_OUT y luego el RNW, este siendo el envío del primer byte. Posterior a esto se baja SDA_OE para ceder la línea a SDA_IN y se nota como el receptor genera el ACK al bajar la línea. Posterior a esto se evidencia como el receptor (el cual se mantiene abierto a la línea por el estado de SDA_OE) envía los datos de RD_DATA del receptor, los cuales son leídos por el generador y se agregan a RD_DATA del generador hasta que el final es igual al resultado esperado. Se nota que cuando se guarda el primer byte de datos se subió SDA_OE para devolver la línea al generador, el cual generó un ACK, mientras que al llegar a los 2 bytes genera un NACK. Finalmente se nota que se da la condición de parada correctamente y que el receptor al leer esta condición vuelve al estado idle, así cumpliento la transición de estados esperada. Con esto se concluye que se realizó la lectura con éxito.

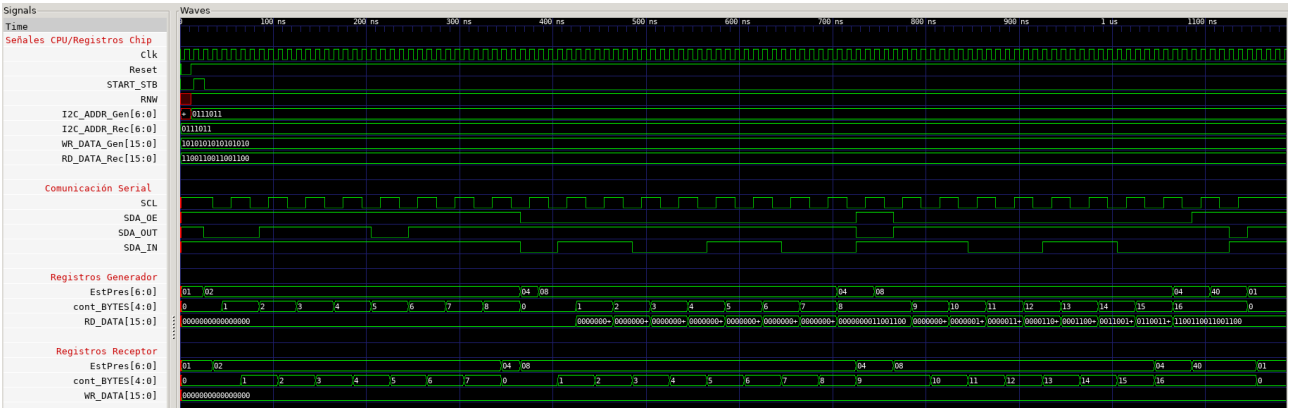

Figura 6: Diagrama de tiempo de prueba #1.

5.2. Resultados prueba #2

En la segunda prueba se obtuvieron resultados de la escritura, donde la gran mayoría de los ciclos resultaron igual que en la prueba anterior (como el bit inicial y la condición final). Las principales diferencias fueron que en este caso solo el receptor generó ACK/NACK, y que cuando se dieron fueron los únicos momentos donde se le otorgaba la línea de comunicación serial al receptor. El resto del tiempo el generador fue el que tuvo la línea y transmitió los datos del WR_DATA por SDA_OUT. Se nota que al final de la simulación el registro de WR_DATA del receptor es equivalente a lo ingresado por el CPU. Los resultados esperados de la prueba se cumplieron, por lo que se concluye que se puede hacer la escritura con éxito.

5.3. Resultados prueba #3

En la Figura 8 se muestra la respuesta del receptor, y consecuentemente el generador, cuando se intenta realizar una transacción con un módulo receptor cuya dirección no corresponde con la ingresada. En la figura se nota como el módulo compara constantemente la entrada serial con la dirección guardada, y en el momento en que se dan diferencias (en el segundo bit) el receptor pasa al estado de Parar_Trans. Este cambio permite

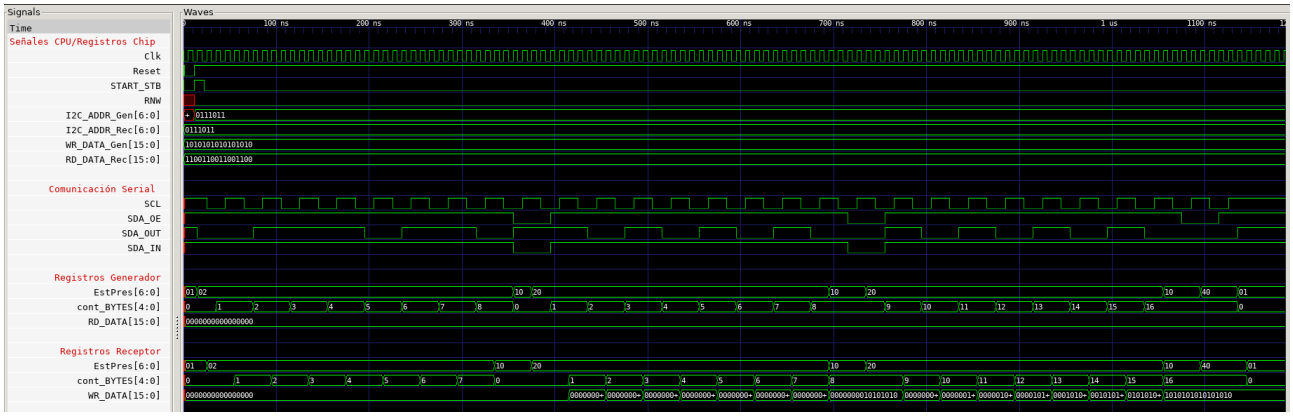

Figura 7: Diagrama de tiempo de prueba #2.

que al llegar el momento de recibir el ACK del receptor, se reciba un NACK (osea se mantenga en alto la señal) y que el receptor ignore los demás comandos hasta que se de la condición de parada. Si hubiesen más módulos conectados al sistema, y uno de estos generara un ACK, la transacción continuaría entre este y el generador sin causar cambios en los demás receptores. Con esto, y con el hecho de que se cumplen los resultados especificados en la sección de pruebas, se concluye que se maneja correctamente transacciones entre diversos dispositivos y diversas direcciones.

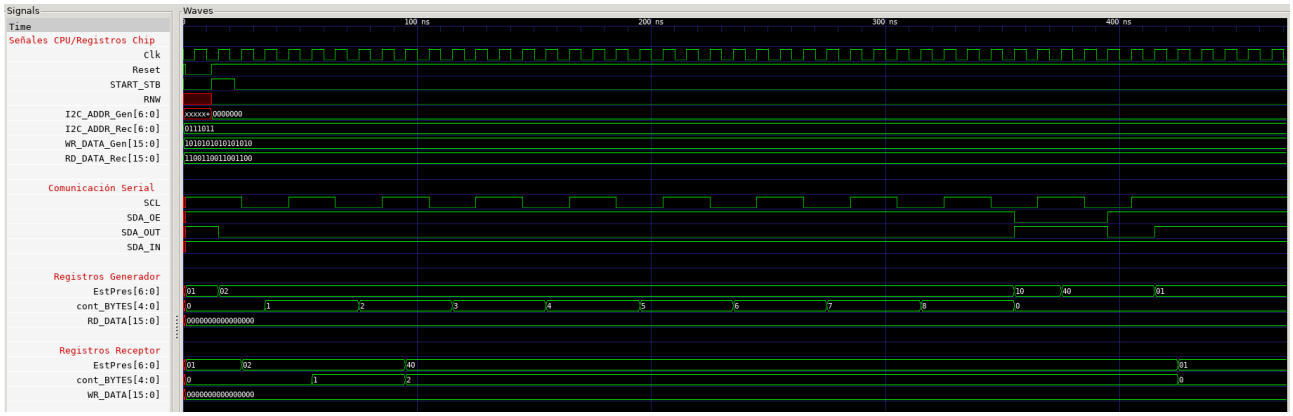

Figura 8: Diagrama de tiempo de prueba #3.

5.4. Resultados prueba #4

En la Figura 9 se muestran los cambios dados cuando se aplica una señal de reset durante una transacción. Se nota que, según se esperada, se vuelve a los estados iniciales y se reinician los registros que se hayan editado durante la misma.

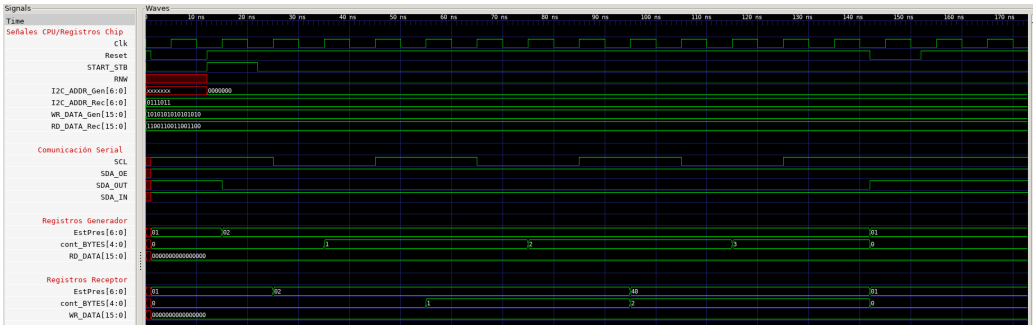

Figura 9: Diagrama de tiempo de prueba #4.

6. Conclusiones y recomendaciones

Durante el proceso de diseño se logró obtener una implementación en Verilog de ambos módulos como se especificó en el enunciado de la tarea y acorde a las especificaciones del protocolo I2C. A través de las pruebas realizadas se puede verificar que estos cumplen con las funciones requeridas, y que siguen el comportamiento requerido para comunicarse con varios dispositivos I2C. Durante el proceso se notó la necesidad de agregar varios registros internos, como el contador de bytes, que podrían ser reducidos en tamaño si se reestructura el código para dicho propósito. Debido a que la síntesis no era el enfoque de este diseño, se escogió la forma que presentara mayor legibilidad y claridad en el código. Por medio de la tarea fue posible estudiar la comunicación serial y coordinación entre módulos con señales de reloj generadas por divisores de frecuencia. Finalmente, se logró estudiar ampliamente el estándar I2C, permitiendo que se emplee en futuros proyectos como una alternativa a la comunicación en paralelo.