Universidad de Costa Rica

Facultad de Ingeniería
Escuela de Ingeniería Eléctrica
IE0523 - Circuitos Integrados Digitales
II ciclo de 2024

Tarea #4:
Generador y receptor de transacciones I^2C

Profesor:
Enrique Coen Alfaro

Estudiante:
Gabriel Siles Chaves - C17530

Grupo 01

27 de octubre 2024

Índice

1. Resumen

En esta Tarea #4 se diseña un generador al igual que un receptor de transacciones con un protocolo I^2C conforme a las especificaciones del manual I^2C UM1024. Este protocolo permite la comunicación síncrona de datos entre dispositivos en un bus bidireccional usando solo dos líneas: una línea de datos (*SDA*) y una de reloj (*SCL*). El generador de transacciones cumple el rol de *master*, controlando la comunicación mediante señales de inicio y parada y manteniendo la sincronización de datos.

Para iniciar una transacción, el generador activa una condición de inicio, que consiste en bajar la señal *SDA* mientras *SCL* permanece en alto. Luego, en cada flanco positivo de *SCL*, el generador transmite bits del mensaje. Cada byte transmitido es confirmado con una señal de reconocimiento (*ACK*) por el receptor, lo que asegura la integridad de la comunicación. En el diseño, se implementaron mecanismos de lectura y escritura de 16 bits utilizando la señal *RNW* para definir la dirección de la transacción. Una transacción de escritura implica enviar dos bytes de datos al receptor, mientras que una transacción de lectura permite recibir dos bytes desde el receptor hacia el generador.

Al final de cada transacción, el generador envía una condición de parada subiendo la señal *SDA* mientras *SCL* está en alto, finalizando así la comunicación. El diseño se verifica mediante simulaciones que comprueban el correcto funcionamiento de las transacciones de lectura y escritura bajo distintos escenarios, manteniendo la coherencia del protocolo I^2C.

En este reporte se describe todo el proceso y el funcionamiento del controlador, detallando paso por paso la estructura, pruebas, instrucciones de ejecución y los resultados obtenidos con sus respectivas conclusiones.

2. Descripción Arquitectónica

En este caso a la hora de realizar un generador para las transacciones se emplea una diseño de máquina de estados que contenga estados y transiciones con respecto a las entradas para así activar las salidas. La máquina de estados que se puede observar en la Figura 3 tiene un total de 5 estados los cuales son los siguientes:

- **a Estado inicial, Esperando Inicio**: En este este estado se habilita el output enable y se espera a que se obtenga el inicio de la trasacción, desactivando al el sda al recibir inicio.
- **b Envío Address**: Se activa el SDA OE para que en el flanco positivo del scl se genere un contador y envíe los datos de la dirrección hasta llegar a 8 bits.
- **c Esperando ACK**: En este caso se desactiva el SDA OE para recibir el ACK depende lo si se requiere escritura o lectura va a pasar al siguiente estado correspondiente.
- **d Envío de datos W**: Este estado dependiendo los ACK recibidos en el flanco positivo de scl irá enviando los valores de los bytes, al llegar a 8 bits pasa a preguntar al estado C así hasta enviar 2 bytes.
- **e Recibo de datos R**: Desactiva SDA OE para obtener la lectura y en cada flanco positivo de scl guardará el byte correspondiente recibido hasta llegar al tope y enviar el ACK

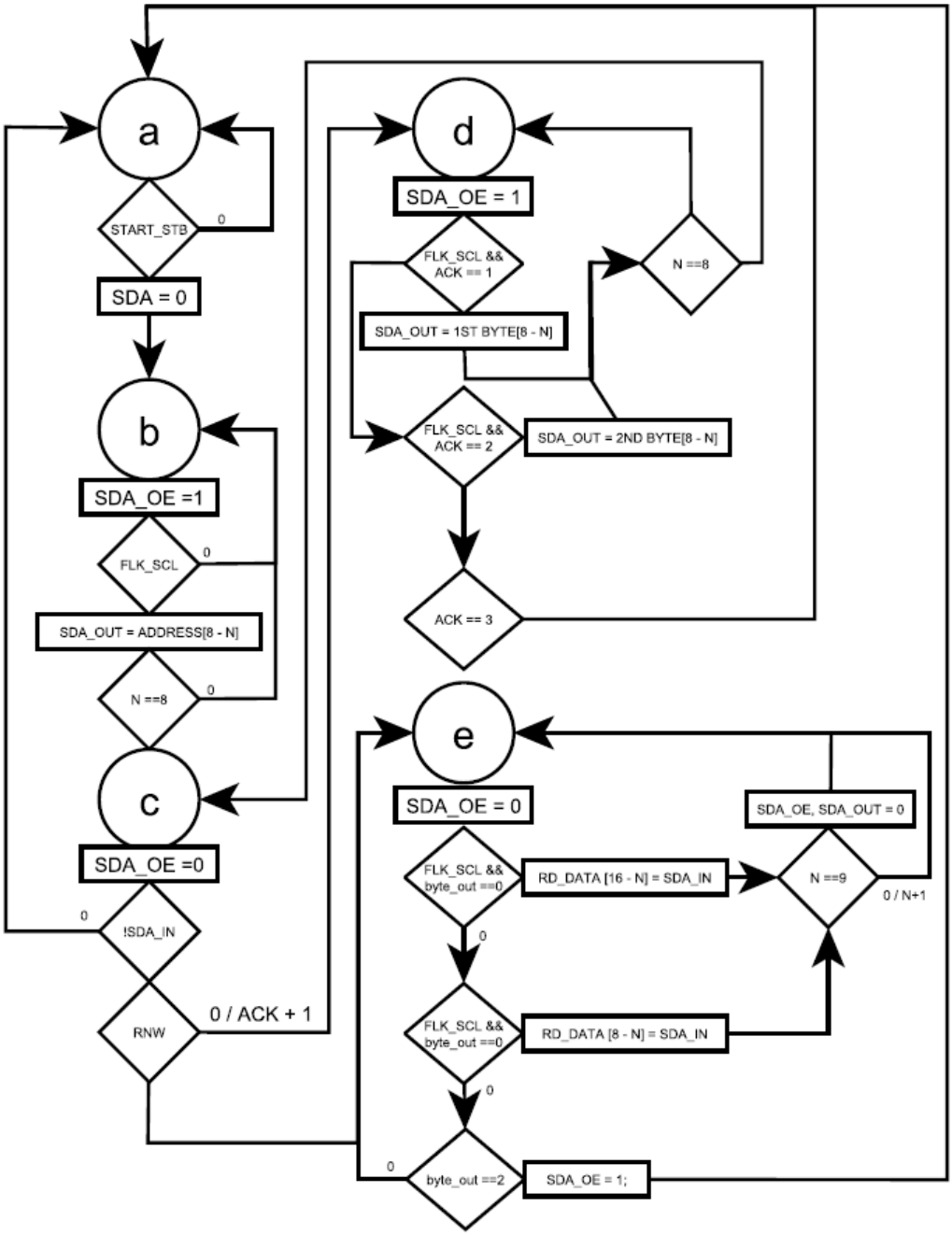

Figura 1: Diagrama ASM máquinas de estado Generador

Para el caso del receptor se debe realizar otra máquina de estado que funcione en sincronización del generador, para así funcionar de manera correcta con el comportamiento del mismo por lo que en la Figura **??** se puede observar el diagrama ASM del receptor en donde se ve la transición de estados y activación de salidas correspondientes. El modulo receptor cuenta con un total de 7 estados que se detallan a continuación:

- **a Estado inicial, Esperando Inicio**: En este se espera que el generador inicie la transacción bajando SDA

- **b Recibo de Address**: El generador envía el address serial por lo que al ser recibido comprueba si es valido para continuar.
- **c Tipo de transacción**: Este estado se encarga de recibir el tipo de transacción que se va a realizar, teniendo en cuenta el ultimo bit del primer byte
- **d Se reciben los datos W**: En este caso el receptor recibe los datos por parte del generador y se realiza la escritura del mismo en el cpu
- **e Envío de datos R**: El receptor envía al generador el dato que se tiene registrado en address indicando enviando por medio de sda_in el valor

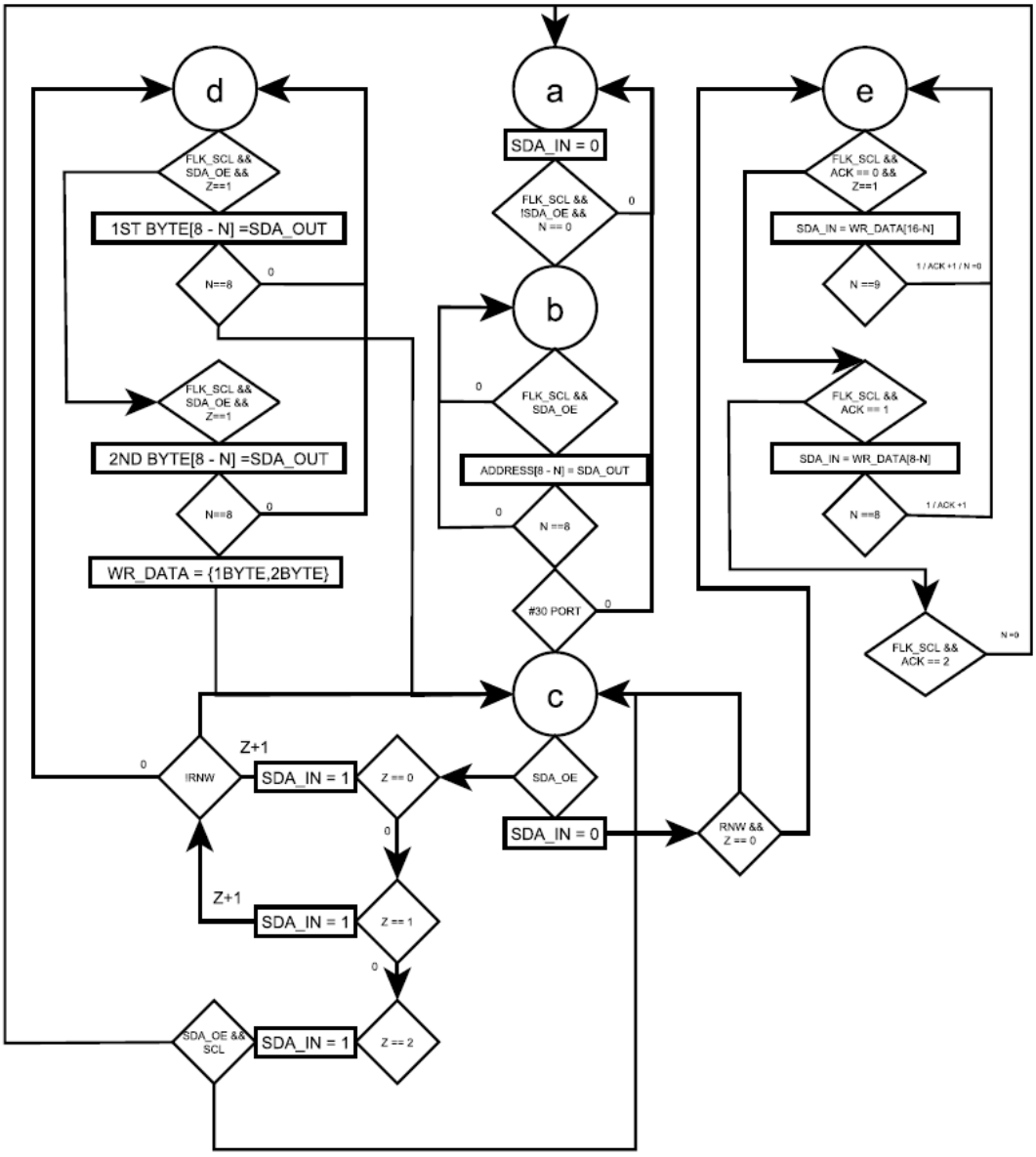

Figura 2: Diagrama ASM máquinas de estado Receptor

Una vez establecido estos dos módulos se conectan de manera compartida en el testbench donde se van a comunicar uno con el otro por medio de wires. Es importante tener en consideración que

para sinconizar ambos módulos se debe asignar continuamente los valores del SCL en la Figura **??** se puede observar como se realiza la asignación de las variables importantes para la comunicación receptor/generador.

```
// Asignaciones Continuas
assign transaccion = {i2caddr, rnw};                    // Concatenacion de la dirección I2C con rnw
assign primer_byte = {wr_data[15:8]};                   // Creacion del primer bloque de data
assign segundo_byte = {wr_data[7:0]};                   // Creacion del segundo bloque de data
assign scl = (inicio_reloj) ? scl_clk[1] : 1;         // Se genera el reloj de scl 25% del clk
assign posedge_scl = !scl_delayed && scl;               // Se crea el flanco poitivo del scl
```

Figura 3: Diagrama ASM máquinas de estado Generador

3. Plan de Pruebas

En esta sección se presenta la lista detallada de las pruebas de cada estado y corroborar el funcionamiento del sistema de acuerdo a las especificaciones. Para esta Tarea se plantearon 3 pruebas.

3.1. Prueba #1. Comunicación para escritura

Se realiza prueba en la situación en donde el cpu desea escribir en un registro específico por lo que enviará el puerto serial con el receptor que se desea comunicar y una vez establecida la conexión se le transmite 2 bytes de datos a lo que el receptor debe responder. La comunicación entre el generador y receptor se realiza para el envío de datos principalmente por el sda_out y se reciben los acknowledge. Al realizar la prueba el diseño lo realizó correctamente

- **02**: Se realiza reset de inicio
- **02**: El cpu carga en las entradas de I2CADDR puerto 30
- **03**: Se indica que se desea escribir pot lo que se manda 0
- **04**: Se guarda en wr_data el valor que se desea escribir
- **05**: Se inicia la transacción indicando el start

3.2. Prueba #2.Comunicación para lectura

Esta segunda prueba simula la situación en donde el cpu desea realizar una lectura del puerto específico seleccionado por lo que da inicio a la transacción por lo que se indica que se desea leer, recibiendo estos parámetros el receptor enviará el acknowledge una vez confirme que el puerto es correcto e iniciará el envío de datos por medio de el hilo de sda_in. Al realizar esta prueba el diseño realizó todo de manera correcto.

- **01**: Se ingresa el puerto 30
- **02**: Se ingresa que se desea escribir 1
- **03**: Se inicia la comunicación al indicar START

3.3. Prueba #3. Puerto no correspondiente

En la tercera prueba el cpu indica un puerto que no es valido para el receptor por lo que no procede a realizar ninguna comunicación, evitando la lectura y escritura manteniéndose en el estado inicial. Esta prubea fue exitosa, indicando que el diseño trabaja acorde a lo esperado con las diferentes pruebas.

- **01**: Se realiza un reset
- **02**: Se ingresa la dirección 31
- **03**: Se indica que se desea escribir
- **04**: Se carga a wr_data el valor de escritura

4. Instrucciones de utilización de la simulación

Para ejecutar los comandos necesarios para realizar la simulación se debe contar con unos requisitos previos para la utilización del mismo, se debe tener instalado las siguientes aplicaciones:

- Icarus Verilog
- GTKWave

En caso de que se desee ejecutar en alguna extensión de Linux (Ubuntu, Debian, Xubuntu) se debe correr una terminal en la carpeta donde se encuentran todos lo archivos necesarios para la ejecución de cada parte. Para ejecutar el diseño conductual con sus respectivas ondas se debe ingresar el siguiente comando:

```
>> make
```

5. Resultados

A la hora de ejecutar el testbench se genera un archivo llamado *results.vcd* el cual al abrirse con el GTKWave se podrá visualizar las ondas y comprobar el funcionamiento del sistema.

5.1. Resultado Prueba #1 Escritura

Al aplicar el tester al *transmisor.v* se obtiene una serie de resultados, las salidas que se activan o cambian con respecto a las entradas. En la Figura 4 se visualiza el funcionamiento correcto del la primera prueba.

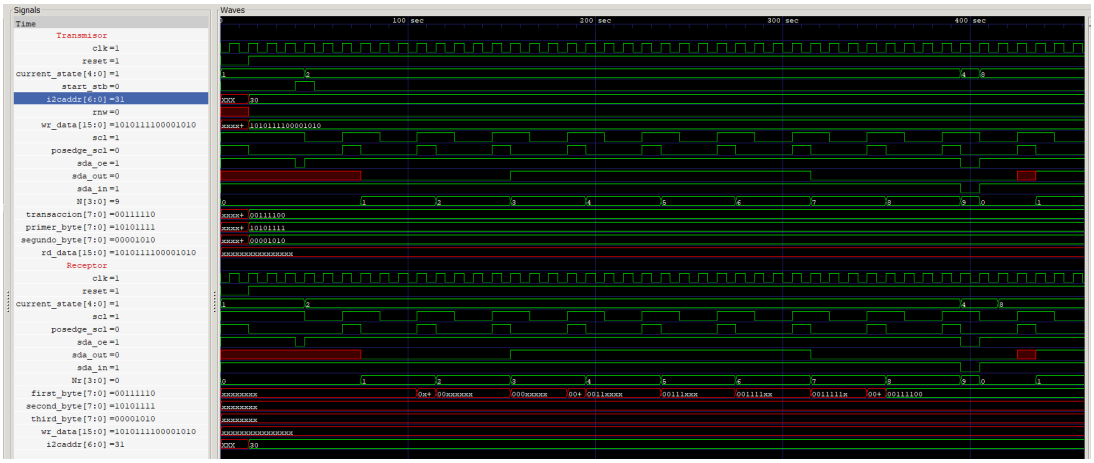

Figura 4: Ondas del GTKwave Escritura etapa Address

Como primer parte de la transacción se puede observar tanto el generador y el receptor. El generador desactiva el SDA por lo que el SCL inicia a oscilar indicando el clk, dando inicio a la transacción. Existe un contador N el cual lleva la cuenta de los flancos del scl y se puede obsrvar como el sda_out cambia con respecto el tiempo mandando el valor del address hasta llegar al 9no flanco, en donde el receptor baja el sda_in, indicando el ACK por lo que se procede con el envío del primer byte. Se puede corroborar el valor transmitido en la variable first_byte que indica que es 00111100 que equivale a 30.

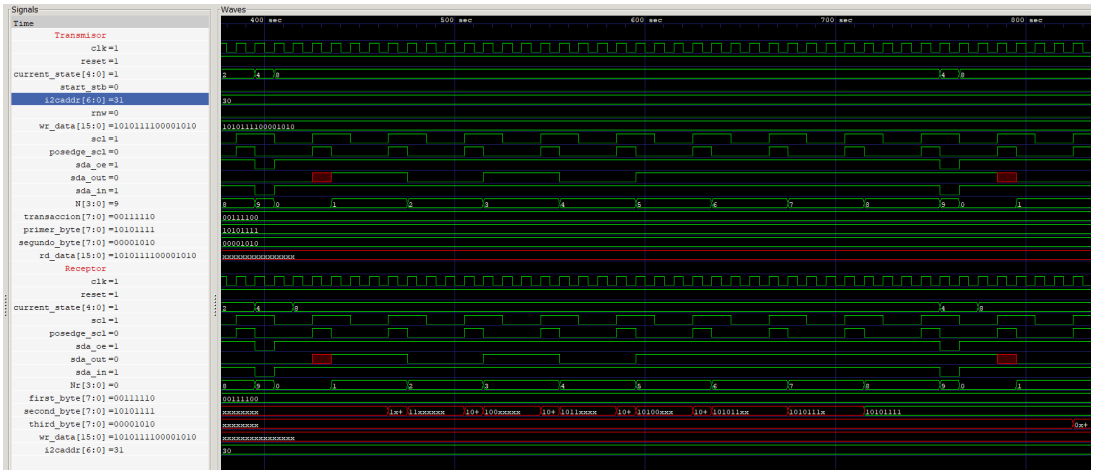

Figura 5: Ondas del GTKwave Escritura etapa primer byte

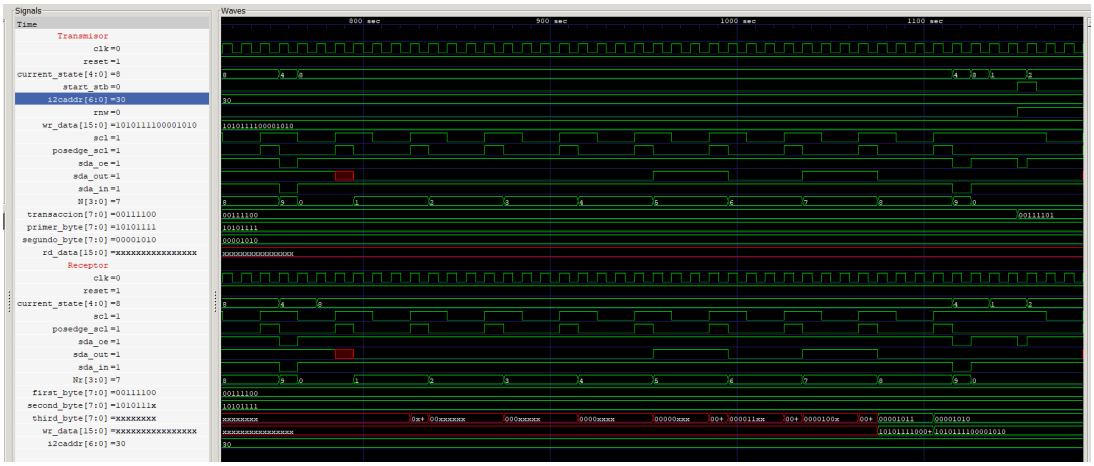

Figura 6: Ondas del GTKwave Escritura etapa segundo byte

Tanto como en el envío del primer byte como en el segundo se puede observar un funcionamiento correcto con forme se va haciendo la transacción de los bits en serie se va completando, en la variable second_byte en la Figura 5 se puede corroborar que al terminar los 8 bits se obtiene 10101111 verificando el envío. Ya en el segundo byte se peude observar el wr_data completo donde ya ambos bytes son recibidos por el receptor. En ambas transacciones se realiza correctamente el ACK para que al final el generador haga la marca de parada subiendo tanto sda_oe y scl.

5.2. Resultado Prueba #2 Lectura

Para la segunda prueba se puede observar en la Figura 7 como en vez de escritura se realiza transacción de lectura, recibe de la misma manera el el puerto serial por lo que es lo mismo unicamente que el último bit transmitido en sda_out es un 1 indicando la lectura. Ya en la Figura 8 se puede observar como el sda output enable se mantiene en 0 permitiendo que sda_in envíe los datos de lectura, por lo que se puede observar como con el tiempo se van completando los espacios del byte.

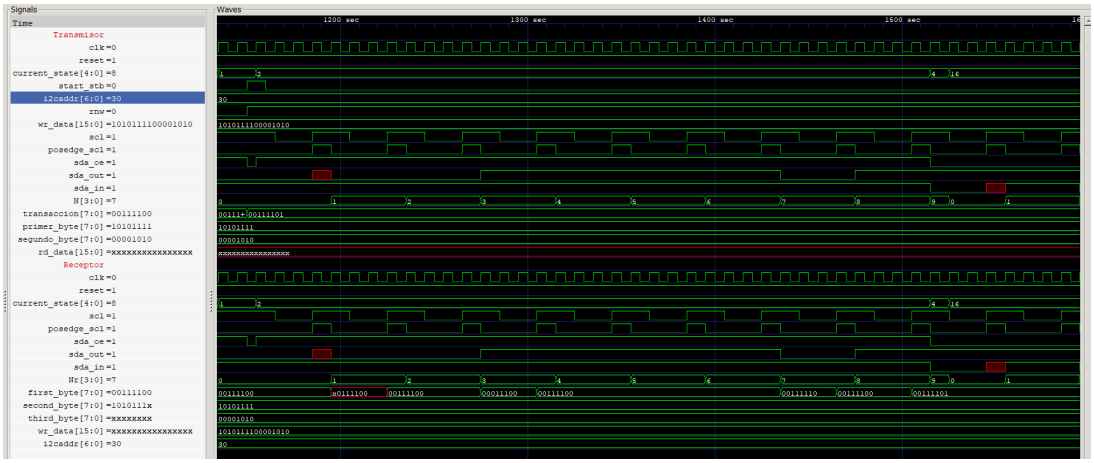

Figura 7: Ondas del GTKwave Lectura etapa Address

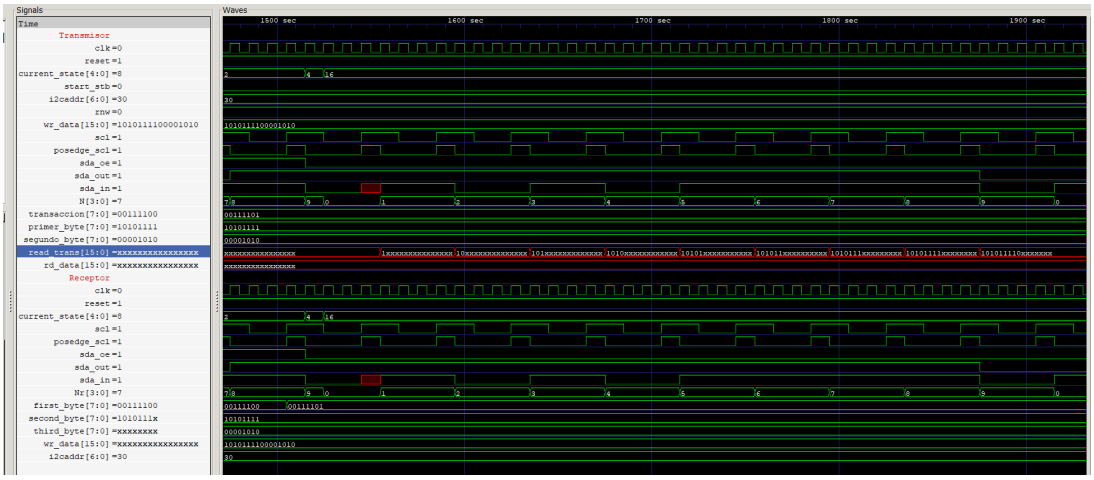

Figura 8: Ondas del GTKwave Lectura etapa primer byte

Figura 9: Ondas del GTKwave Lectura etapa segundo byte

Ya en la Figura 9 Se comprueba que se transmite de manera serial el valor que se tiene registrado en el address unicamente que en este caso el generador indica los ACK de recibido de los valores, manteniendo en bajo tanto output enable como sda out.

5.3. Resultado Prueba #2 Puerto No correspondiente

Para la pultima prueba se puede observar en la Figura 10 como el diseño reconoce que el puerto no es el asociado al receptor por lo que no se genera el proceso de transacción con el mismo. En este caso como anteriormente se había guardado en las variables valores se mantienen así, en este caso unicamente cambió el puerto con que se deseaba realizar la transacción por lo que se puede observar que solo recibe pero no cotinúa con la comunicación quedandose al final en el estado inicial, como se esperaba.

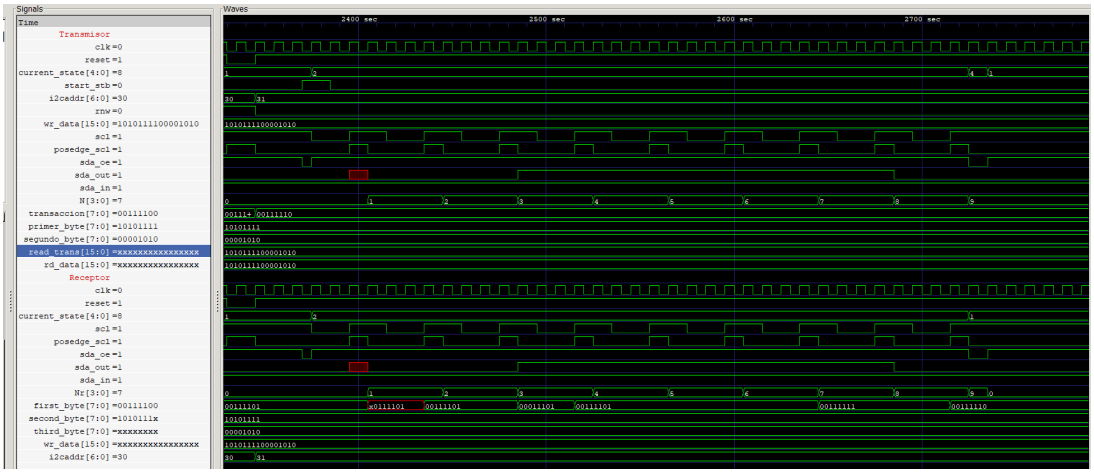

Figura 10: Ondas del GTKwave Puerto No correspondiente etapa address

6. Conclusiones y recomendaciones

En conclusión, se realizó el protocolo de transacción I^2C obteniendo el comportamiento esperado en ambos casos, tanto escritura y lectura. Se debe considerar que para realizar esto, se debía sincronizar y conectar dos módulos por aparte de verilog, generador y receptor, respectivamente. Por medio del testbench estos dos módulos comparte un wire específico que eran los hilos de comunicación del protocolo.

Para el desarrollo de los diagramas y la estructura del código se tuvo que realizar cada paso con cautela manteniendo cuidado y agregando detalles para el funcionamiento correcto del protocolo. En general se puedo analizar y obtener las respuestas de los errores que se presentaban. Es importante tener en cuenta que entender el protocolo I^2C es el aspecto más importante a la hora de realizar el diseño de la estructura conductual del código, debido a que influye en que paso a paso se mantenga el comportamiento para poder obtener éxito.

Es recomendable estudiar de cerca tener orden a la hora de aplicar los diagramas al código debido a que se puede simplificar muchas veces estados que no son necesarios y mantener un sistema más simple y permitiendo cumplir con el objetivo. Muchas de las ocasiones los estados se podrían diseñar de manera que cumplan las tareas en un mismo estado en vez de varios por lo que poder definir claramente en que se encarga cada uno es vital para poder concluir con el código.