Tarea 4 Digitales 2

Mario Enrique Brenes Arroyo
Universidad de Costa Rica
Fecha: 26 de octubre de 2024
Carnet: C11194
Grupo: 1
Curso: Circuitos Digitales 2
Profesor(a): Enrique Coen

Índice

1. Resumen

Este proyecto implementa un sistema de comunicación I2C compuesto por un generador y un receptor de transacciones, desarrollado en Verilog. El generador permite iniciar y controlar transacciones I2C para lectura y escritura de datos, mientras que el receptor gestiona la recepción y confirmación de estos datos conforme a la especificación del protocolo I2C. Las pruebas realizadas incluyen transacciones de lectura y escritura de dos bytes, verificando la correcta sincronización y transmisión de datos en ambas direcciones.

La simulación se llevó a cabo en Icarus Verilog, y los resultados fueron visualizados en GTKWave, mostrando un comportamiento acorde al protocolo y a las especificaciones. Durante el desarrollo, se resolvieron algunos problemas de sincronización en las señales de control, los cuales se ajustaron modificando la frecuencia del reloj de SCL en función de la señal de reloj principal.

2. Descripción Arquitectónica

Este sistema implementa un controlador I2C en Verilog, compuesto por dos módulos principales: un generador de transacciones I2C y un receptor de transacciones I2C. Ambos módulos permiten realizar una comunicación sincronizada, donde el generador inicia transacciones de lectura y escritura, mientras que el receptor maneja la recepción y confirmación de datos según el protocolo I2C.

Descripción Arquitectónica del Generador de Transacciones I2C

El módulo `i2c_transaction_generator` implementa la lógica de control para iniciar y gestionar transacciones en el protocolo I2C, incluyendo operaciones de lectura y escritura. Este generador se encarga de controlar el bus I2C y se estructura en base a una máquina de estados finitos (FSM) que sigue los pasos necesarios para cada transacción.

Entradas y Salidas

- **clk**: Señal de reloj del sistema que sincroniza el módulo.
- **rst**: Señal de reinicio que inicializa el módulo.

- **start_stb**: Indica el inicio de una transacción.
- **rnw**: Controla el tipo de operación, siendo lectura (1) o escritura (0).
- **i2c_addr**: Dirección I2C del dispositivo receptor.
- **wr_data**: Datos de 16 bits que se envían en una transacción de escritura.
- **sda_in**: Entrada de datos serial para recibir datos del receptor.
- **scl** (salida): Señal de reloj I2C generada con una frecuencia equivalente a un cuarto de **clk** con un funcionamiento de 1/4 de la frecuencia de reloj.
- **sda_out** (salida): Salida serial que envía datos en el bus I2C.
- **sda_oe** (salida): Habilitación de salida de datos serial, que permite la transmisión en el bus I2C.
- **rd_data** (salida): Registro de 16 bits que almacena los datos recibidos en una transacción de lectura.

Máquina de Estados Finita (FSM)

La FSM controla cada etapa de la transacción I2C, pasando por los siguientes estados principales:

- **IDLE**: Estado de espera inicial. Si `start_stb` está activo, el generador pasa al estado **START**.
- **START**: Envía la condición de inicio en el bus I2C, estableciendo `sda_out` en bajo.
- **SLAVE_ADDRESS**: Envía la dirección I2C y el bit de lectura/escritura. Una vez completada, el generador espera el **ACK**.
- **ACK**: Recibe la confirmación del receptor. Dependiendo de `rnw`, el generador continúa con **LECTURA_1** o **ESCRITURA_1**.
- **LECTURA_1** y **LECTURA_2**: Capturan los datos recibidos en una transacción de lectura, almacenándolos en `rd_data`.
- **ESCRITURA_1** y **ESCRITURA_2**: Envía los datos de `wr_data` en una transacción de escritura.

- **WAITACK** y **WAITACK_2**: Espera la confirmación del receptor después de cada byte transmitido.
- **STOP**: Genera la condición de parada, concluyendo la transacción y volviendo al estado **IDLE**.

Esta estructura permite que el generador mantenga el control del bus I2C, alternando entre los estados de transmisión y recepción según el protocolo. La señal `scl` se gestiona para operar a una frecuencia que es una cuarta parte de la frecuencia del reloj `clk`.

Descripción Arquitectónica del Receptor de Transacciones I2C

El módulo `i2c_transaction_receiver` actúa como receptor en el protocolo I2C, gestionando la recepción de datos seriales provenientes del generador de transacciones y enviando respuestas de confirmación (ACK) cuando corresponda. La arquitectura se basa en una máquina de estados finita (FSM) para seguir las fases de una transacción I2C, permitiendo la sincronización y control de datos en el bus I2C.

Entradas y Salidas

- **clk**: Señal de reloj del sistema utilizada para la sincronización de los estados.
- **rst**: Señal de reinicio que inicializa el módulo en su estado de espera.
- **i2c_addr** (salida): Dirección del receptor, que permite verificar la dirección recibida en la transacción.
- **scl**: Reloj I2C controlado por el generador y utilizado para sincronizar la transferencia de datos.
- **sda_out** (entrada): Datos seriales recibidos del generador.
- **sda_oe** (entrada): Habilitación de `sda_out`, indicando cuándo el receptor debe leer datos.
- **sda_in** (salida): Salida serial para responder con los datos o el bit de confirmación (ACK).

- **wr_data** (salida): Almacena los datos de 16 bits recibidos en una transacción de escritura.
- **rd_data** (entrada): Datos de 16 bits que el receptor envía en una transacción de lectura.

Máquina de Estados Finita (FSM)

La FSM controla las diferentes etapas de la transacción I2C, alternando entre estados de lectura y escritura según el tipo de transacción, y asegurando la sincronización adecuada con el generador:

- **ESPERA**: Estado inicial de espera. El receptor monitorea la señal `sda_out` y detecta la condición de inicio.
- **SLAVE_ADDRESS**: El receptor recibe la dirección I2C desde el generador y la compara con su propia dirección (`i2c_addr`).
- **ACK**: Si la dirección es correcta, el receptor envía un ACK para confirmar la recepción.
- **DATA** y **DATA_2**: Estados en los que el receptor recibe o envía datos de acuerdo al tipo de transacción. En modo de escritura, los datos se almacenan en `wr_data`. En modo de lectura, los datos de `rd_data` se envían al generador a través de `sda_in`.
- **WAITACK** y **WAITACK_2**: El receptor espera el ACK del generador para confirmar la recepción de cada byte de datos.

La FSM asegura que cada estado sea sincronizado por la señal de reloj I2C (`scl`) para mantener la coherencia con el protocolo. Al finalizar una transacción, el receptor regresa al estado **ESPERA** hasta que el generador inicie una nueva transacción.

Dentro de los retos que tenianmos al diseñar el receptor fue la sincronización del contador, para que pudiera aceptar todos los bits que salian del generador ya sea en addr o escritura debiamos modificar el contador para retrasarlo haciedno que en caso de dar un conteo de 15 a 0 en el momento que llega a 8 alargar el tiempo y ademas de eso debiamos modificar el (`wr_data`) del receptor para que no ingresara valores en el momento que alargabamos el contador a 8 porque si no hubieramos tenido problemas de ingresar el bit 8 dos veces.

Para esto lo que hicimos fue modificar la siguiente recepción de datos para que agregue los bit cuando el contador sea menor a 8 y asi no duplicar el valor.

3. Plan de pruebas

para estos modulos se harán 2 modulos de pruebas, uno de escritura y otro de lectura, veremos el comportamiento debido del generador y el receptor.

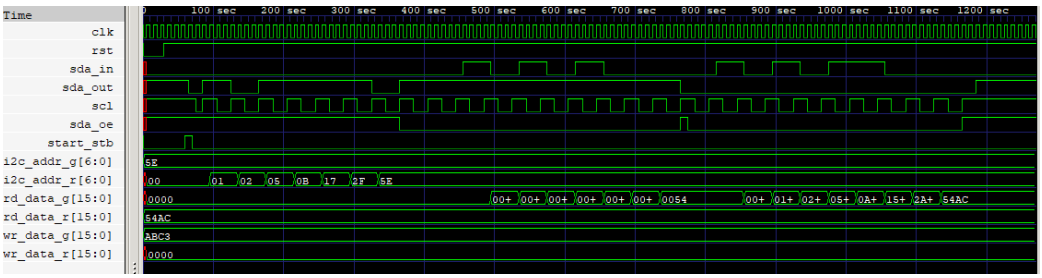

Figura 1: prueba de escritura

- **Prueba de Reinicio:** Se inicia el sistema aplicando y quitando la señal de reinicio (`rst`). Esto asegura que tanto el generador como el receptor comiencen en un estado inicial conocido y que las señales sean reiniciadas adecuadamente.

- **Transacción de Escritura:** Con la señal `rnw` en bajo (0), se envían datos de prueba (`wr_data_g` = `16'hABC3`) al receptor I2C en la dirección configurada (`i2c_addr_g` = 94). Durante esta prueba, se monitorean las señales `scl`, `sda_out` y `sda_oe` para verificar la transmisión de datos y la respuesta de confirmación (ACK) del receptor.
  Se puede ver como la señal sda out toma los valores respectivos para que se genere una señal serial tanto de la dirección, rnw y el registro que contiene lo que quiero escribir.

- **Verificación de Señal de Inicio y Parada:** Se comprueba la generación de las condiciones de inicio y parada en el bus I2C mediante la señal `start_stb`, asegurando que el generador cumpla con las transiciones de señal esperadas para iniciar y finalizar cada transacción.

Los resultados de estas pruebas se registraron en forma de ondas para cada señal relevante, confirmando el cumplimiento del protocolo I2C y el correcto funcionamiento de las transacciones de lectura y escritura.

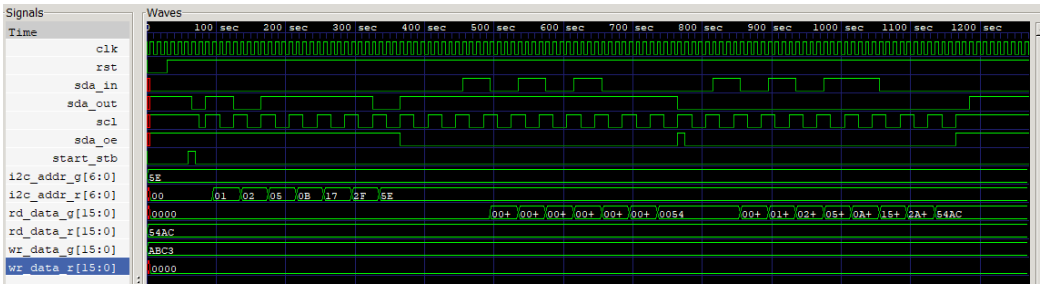

Figura 2: prueba de lectura

- **Transacción de Lectura:** Con la señal `rnw` en alto (1), se verifica la recepción de datos desde el receptor (`rd_data_g`). Los datos de prueba en el receptor están predefinidos como `rd_data_r = 16'h54AC` y se verifica que el generador reciba estos datos correctamente.
  se puede ver como se genera primero la direción de donde quiero que se ejecute la lectura y tambien podemos ver como el registro de lectura del generador se le van añadiendo valores con la señal sda in hasta completar la transacción de byte en byte.

En ambas pruebas se puede ver como la condición start y stop se generan al principio de la consulta cuando se presiona el start stb y despues cuando se finaliza toda la transación se ejecuta la condición stop con la sñal `sda_out` y la señal scl.

4. Ejemplo de resultados

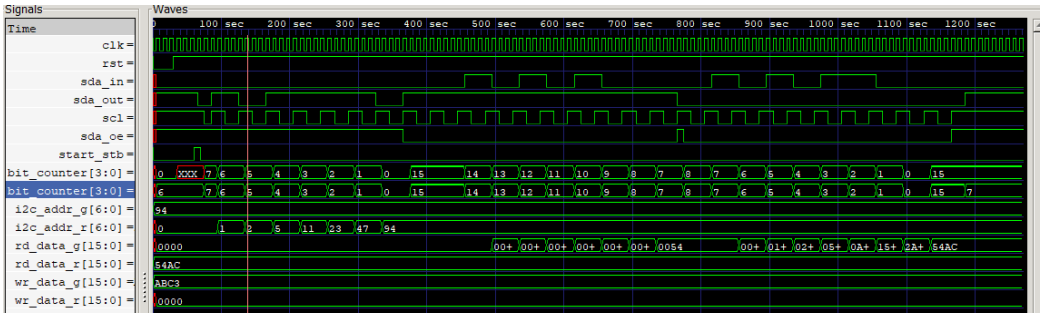

Figura 3: resultados de lectura

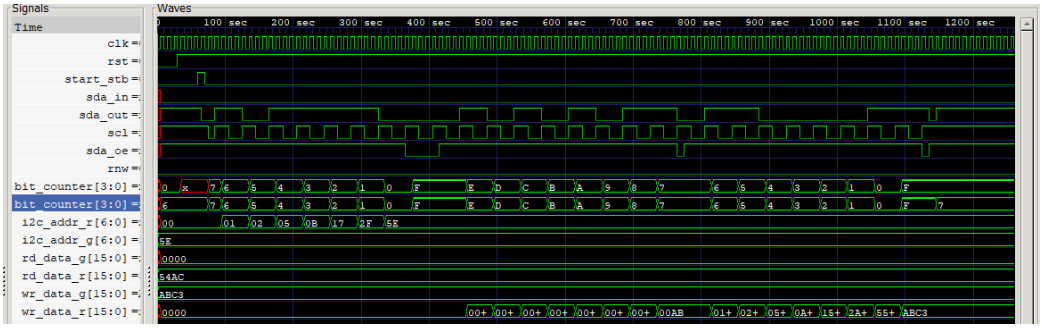

Figura 4: resultados de escritura

Para poder sincronizar ambos modulos se crearon contadores que trabajaran juntos tanto en recibir las señales ACK como sda in y sda out para las respectivas funciones. Aunque el generador tuviera mas estados que el receptor, no hizo falta crear tantos ya que solo debia enviar o recibir datos el receptor dependiendo de la comunicación que se deseara usar.

Instrucciones para Ejecutar el Programa

Para ejecutar las simulaciones de las transacciones I2C, se ha configurado un `Makefile` que facilita la compilación, simulación y visualización de los resultados. A continuación se describen los pasos necesarios:

1. **Compilación y Simulación:** Para compilar el archivo de prueba y ejecutar la simulación, utiliza el siguiente comando:

   ```
   make run
   ```

   Esto compilará el archivo `tb.v` usando `iverilog`, ejecutará la simulación con `vvp`, y abrirá el archivo de resultados `prueba1.vcd` en GTKWave para visualizar las señales.

2. **Prueba de Escritura o Lectura:** Si deseas cambiar entre pruebas de lectura y escritura, modifica el valor de la línea 25 en el archivo `tester_i2c.v`:

   ```
   rnw = 1'b1;  // Cambiar a 0 para escritura, o 1 para lectura
   ```

3. **Limpiar Archivos Generados:** Para eliminar los archivos de salida y resultados generados en la simulación, utiliza:

```
make clean
```

Este flujo de trabajo permite realizar simulaciones completas y visualizar los resultados en `GTKWave`, facilitando la verificación del funcionamiento del sistema de transacciones I2C.

5. Conclusiones y Recomendaciones

El diseño y la implementación del generador de transacciones I2C requirieron el uso de múltiples estados en la máquina de estados finita (FSM) para gestionar cada etapa del protocolo I2C. Aunque se lograron los objetivos de funcionalidad, la FSM podría optimizarse para reducir la cantidad de estados y simplificar el flujo de control.

Un desafío importante fue la sincronización del reloj entre el generador y el receptor, especialmente al activar el contador del receptor en el momento exacto de la condición de inicio (START). Lograr esta precisión fue esencial para asegurar que la transmisión y recepción de datos ocurran en los momentos correctos, evitando desajustes de tiempo.

Otro aspecto complejo fue la recepción y transmisión de datos entre el generador y el receptor. Para esto, se sincronizó el contador utilizado en ambos módulos, permitiendo que la transmisión de datos coincidiera con los flancos positivos de la señal `scl`, manteniendo la integridad del protocolo.

Todo el proyecto lo puede encontrar en este repositorio de github tarea4.