Universidad de Costa Rica
Facultad de Ingeniería
Escuela de Ingeniería Eléctrica
IE-0523 Circuitos Digitales II
I ciclo 2024

Tarea #4 Diseño de una interfaz serial periférica (SPI)

Oscar Porras Silesky C16042
Grupo 001

Profesor: Ing. Enrique Coen

31 de mayo de 2023

Índice

1. Resumen

Este proyecto aborda el diseño e implementación de un sistema de comunicación SPI (Interfaz Periférica Serial) utilizando Verilog. El sistema consta de módulos master y slave que interactúan para realizar transacciones de datos de manera coordinada y segura.

El rendimiento del sistema SPI se demostró eficazmente en la gestión de las señales y en la coordinación de las transacciones de datos entre el master y el slave. Se implementaron mecanismos de manejo de errores y reloj para asegurar la integridad y la confiabilidad de la transmisión de datos.

Se llevaron a cabo pruebas detalladas para verificar la funcionalidad del sistema bajo varios escenarios, enfocándose en la correcta inicialización, transmisión de datos, y respuesta a errores en tiempo de ejecución. Estas pruebas validaron que el diseño del sistema SPI es robusto y capaz de manejar distintas configuraciones de comunicación y errores de transmisión.

Una característica destacada de este diseño es la habilidad para adaptarse dinámicamente a distintos modos de operación del reloj, manejando efectivamente las configuraciones de polaridad y fase del reloj (CKP y CPH). La eficiencia del sistema se mejoró significativamente mediante la optimización del control de reloj y el manejo eficiente de las líneas MOSI y MISO, asegurando así la sincronización perfecta entre los dispositivos.

Este documento detalla cada parte del diseño, la funcionalidad del testbench, y la integración de los módulos master y slave, ilustrando cómo interactúan dentro del entorno de prueba SPI.

2. Descripción Arquitectónica

2.1. Diagrama de Bloques del controlador SPI

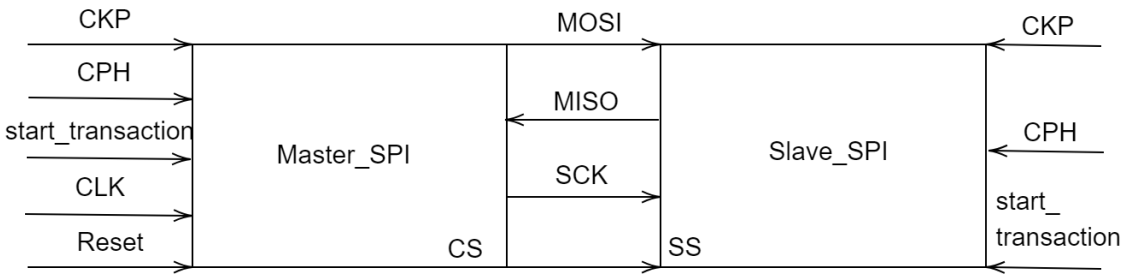

Figura 1: Diagrama de bloques del controlador SPI.

En la figura del controlador SPI, se muestran las interacciones entre las señales del sistema y el módulo `SPIMaster` y `SPISlave`. Las señales y su propósito son los siguientes:

- `CLK`: Reloj proporcionado por el sistema central (CPU) para sincronizar la transmisión y recepción de datos entre el maestro y el esclavo SPI.
- `RESET`: Señal de reinicio que devuelve el controlador SPI a su estado inicial, preparándolo para una nueva transacción o para corregir errores de transmisión.
- `CS` (Chip Select): Salida del maestro SPI que activa el esclavo deseado para comenzar una comunicación. Es activa en bajo, indicando que el esclavo seleccionado debe responder a las peticiones del maestro.

- `SS` (Slave Select): Entrada en cada dispositivo esclavo SPI. Cuando esta señal está en bajo, el esclavo sabe que debe participar en la comunicación y responder a las instrucciones del maestro.
- `SCK` (Serial Clock): Salida del reloj de SPI desde el maestro hacia el esclavo. El esclavo utiliza esta señal para sincronizar la recepción y envío de datos serialmente a través de MOSI y MISO.
- `MOSI` (Master Out Slave In): Salida de datos serial del maestro hacia el esclavo. El maestro envía datos al esclavo bit a bit a través de esta línea.
- `MISO` (Master In Slave Out): Salida de datos serial del esclavo hacia el maestro. El esclavo envía datos al maestro bit a bit a través de esta línea.
- `CKP` (Clock Polarity): Configura la polaridad del reloj en el maestro para determinar el nivel de inactividad del reloj SCK cuando no hay transacciones.
- `CPH` (Clock Phase): Configura la fase del reloj para determinar en qué flanco del reloj SCK el maestro y el esclavo deben muestrear y/o cambiar sus líneas de datos.
- `start_transaction` (Start transaction): Bandera utilizada para confirmar que se puede comenzar el proceso de transacción de datos.

2.2. Funcionamiento de los Archivos

En el contexto del sistema SPI, cada archivo tiene un propósito específico dentro del proceso de simulación y verificación:

- `Testbench.v`: Este archivo actúa como el testbench principal que genera la simulación.
- `Master.v`: Contiene la implementación del módulo master del SPI. Se encarga de iniciar las transacciones, generar el reloj SPI y manejar la comunicación mediante las señales MOSI y SCK.
- `Slave.v`: Contiene la implementación del módulo slave del SPI. Responde a las señales del master, recibiendo datos a través de MOSI y enviando respuestas por MISO.
- `Tester.v`: Proporciona el entorno de test y genera las señales de entrada para el módulo master y el módulo slave.

2.3. Diagrama ASM del Transmisor SPI

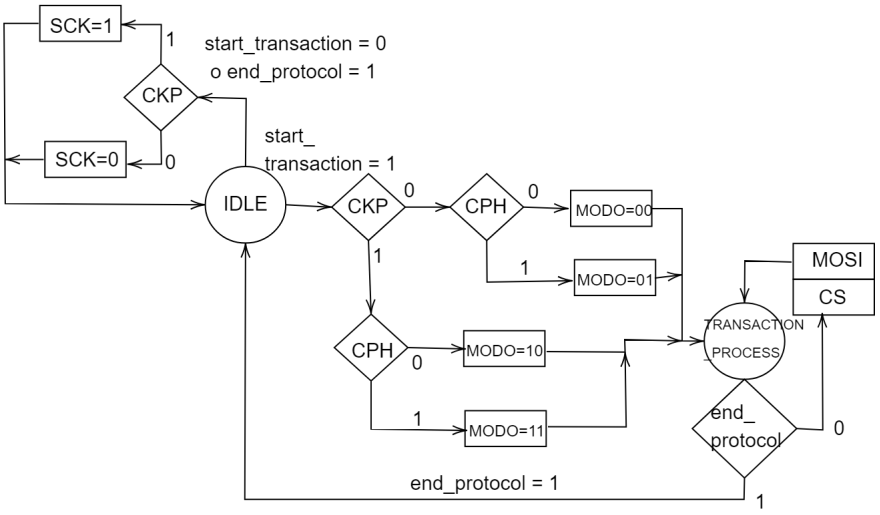

Figura 2: Diagrama de estados del módulo transmisor SPI (Master).

Los estados del módulo transmisor SPI (Master) se describen a continuación:

- `IDLE (00)`: Estado de reposo. El sistema espera una señal para iniciar una transacción. Mientras está en este estado, el CS (Chip Select) está desactivado (alto), indicando que no hay transacción en curso. El sistema permanece en este estado hasta que se recibe la señal `start_transaction` activa.
- `TRANSACTION_PROCESS (01)`: Estado de procesamiento de la transacción. Una vez que se inicia la transacción, el sistema activa la señal CS (bajo), comienza la generación del reloj SCK según los parámetros CKP y CPH, y procede a enviar datos a través de MOSI. Si el contador de bits alcanza el límite predefinido indicado por `end_protocol`, el sistema regresa al estado `IDLE`.

2.4. Diagrama ASM del Receptor SPI

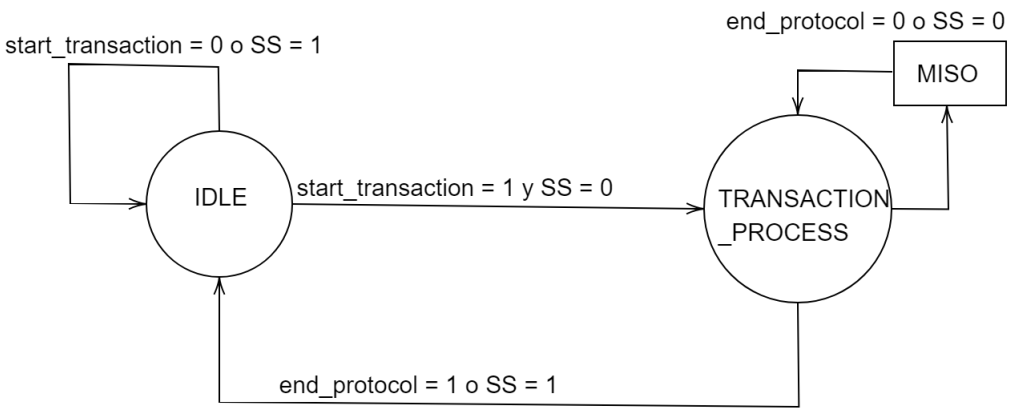

Figura 3: Diagrama de estados del módulo receptor SPI (Slave).

Los estados del módulo receptor SPI (Slave) se describen a continuación:

- `IDLE (00)`: Estado de reposo. El esclavo espera que el maestro inicie una transacción y el CS sea activado (bajo). Mientras está en este estado, no se realizan transacciones activas.
- `TRANSACTION_PROCESS (01)`: Estado de procesamiento de la transacción. Cuando el maestro inicia una transacción y activa CS, el esclavo comienza a recibir datos a través de MOSI y responde enviando datos a través de MISO, sincronizado con la señal de reloj SCK. Este estado continua hasta que el maestro señale el fin de la transacción, normalmente cuando CS vuelve a estado alto, lo que hace que el esclavo regrese al estado `IDLE`.

2.5. Configuración de Daisy Chain y Modos del Reloj en SPI

El protocolo SPI es una forma eficiente de permitir la comunicación entre varios dispositivos, especialmente usando la configuración en cadena Daisy Chain. En esta configuración, varios dispositivos esclavos están conectados en serie, lo que permite que un solo dispositivo maestro se comunique con todos ellos a través de un único bus de datos. Es importante entender cómo el maestro y los esclavos manejan las señales de control, en especial la señal de reloj, para asegurar que los datos se transmitan correctamente.

En este contexto, los modos del reloj son clave. El protocolo SPI define cuatro modos que determinan cómo se relacionan la fase y la polaridad del reloj. Estos modos especifican cuándo deben transmitirse y capturarse los datos, ya sea en el flanco ascendente o descendente del reloj.

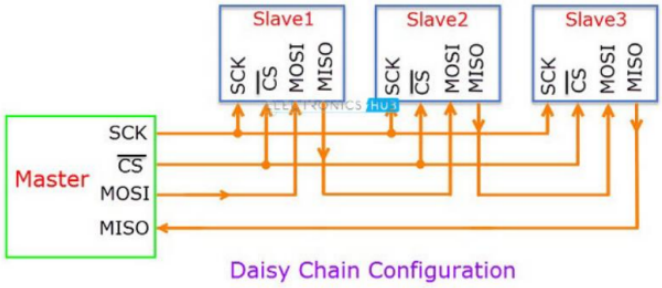

Figura 4: Diagrama de Configuración Daisy Chain.

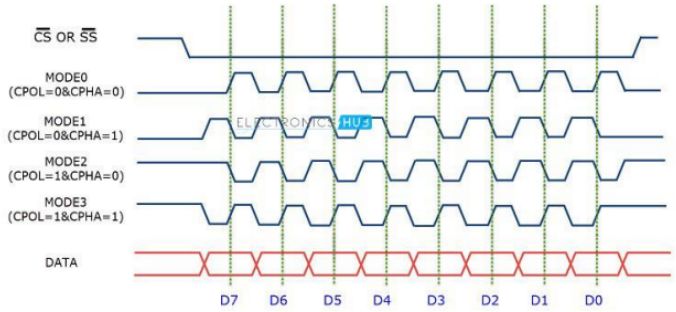

Figura 5: Diagrama de Modos del reloj SCK en la transmisión de datos.

3. Plan de Pruebas

El plan de pruebas se diseñó para validar el funcionamiento correcto del módulo SPI Master y Slave. A continuación se enumeran las pruebas realizadas, describiendo cada una con su propósito específico, los pasos ejecutados, y los resultados esperados.

1. **Prueba #1: Modo CKP=0, CPH=1 (Modo SPI 01)**
   - *Descripción:* Verificar la correcta transmisión y recepción de datos bajo la configuración de reloj con CKP=0 y CPH=1.
   - *Pasos:*
     i. Inicialización de los módulos con Reset activo.
     ii. Establecimiento de CKP=0 y CPH=1.
     iii. Activación de la señal Begi_trans para iniciar la transmisión.
     iv. Observación de la transmisión de datos y su correcta recepción en el módulo esclavo.
   - *Resultado esperado:* Los datos enviados por el master deben ser correctamente recibidos por el esclavo y el registro de datos debe coincidir al final de la transmisión.
   - *Resultado obtenido:* Exitosamente completado.
2. **Prueba #2: Modo CKP=1, CPH=0 (Modo SPI 10)**
   - *Descripción:* Evaluar la operación de los módulos bajo la configuración de CKP=1 y CPH=0.
   - *Pasos:*
     i. Reset de los módulos.
     ii. Configuración de CKP=1 y CPH=0.
     iii. Inicio de la transmisión y monitoreo de la integridad de los datos recibidos.
   - *Resultado esperado:* El esclavo debe recibir los datos exactamente como son enviados por el master, verificando la correcta configuración del reloj.
   - *Resultado obtenido:* Exitosamente completado.
3. **Prueba #3: Modo CKP=0, CPH=0 (Modo SPI 00)**
   - *Descripción:* Confirmar la funcionalidad de la comunicación con ambos parámetros de reloj en bajo.
   - *Pasos:*
     i. Configuración inicial y reset.
     ii. Ajuste de CKP=0 y CPH=0 para la prueba.
     iii. Transmisión de datos y comprobación de la recepción.
   - *Resultado esperado:* Los datos deben ser manejados correctamente entre el master y el esclavo, respetando los flancos del reloj configurados.
   - *Resultado obtenido:* Exitosamente completado.
4. **Prueba #4: Modo CKP=1, CPH=1 (Modo SPI 11)**
   - *Descripción:* Verificar la correcta transferencia de datos cuando CKP y CPH están en alto.
   - *Pasos:*
     i. Activación de la configuración CKP=1 y CPH=1.
     ii. Ejecución de una transmisión de prueba y validación de la recepción en el esclavo.
   - *Resultado esperado:* Tanto el máster como el esclavo deben operar correctamente bajo esta configuración, con una sincronización adecuada y sin pérdida de datos.
   - *Resultado obtenido:* Exitosamente completado.

3.1. Prueba en Configuración Daisy Chain

- *Descripción:* Verificar la integridad de los datos tras ser transmitidos a través de una cadena de tres esclavos.
- *Pasos:*
  i. Configurar la conexión daisy chain con 3 esclavos.
  ii. Transmitir datos desde el master, pasando por los 3 esclavos.
  iii. Verificar que los datos regresen al master intactos después de la última transmisión.
- *Resultado esperado:* Los datos iniciales deben regresar al master después de pasar por los 3 esclavos, indicando una transmisión y recepción correcta a través de toda la cadena.
- *Resultado obtenido:* Exitosamente completado.

4. Instrucciones de utilización de la simulación

4.1. Makefile

Para facilitar el proceso de compilación y simulación, se ha proporcionado un `Makefile`. Para utilizar el archivo `Makefile`, este debe estar en la misma carpeta que el resto de archivos, al igual que el archivo `gtkwaveconfig.gtkw`.

4.1.1. Para solamente compilar:

Se debe correr el comando `make compile`

4.1.2. Para solamente ejecutar la simulación:

Se debe correr el comando `make run`

4.1.3. Para solamente abrir el GTKWave:

Se debe correr el comando `make gtkwave`

4.1.4. Para compilar, simular y abrir el GTKWave:

Para ejecutar todos los comandos automáticamente, se debe correr el comando `make` en la terminal. Este comando compilará el testbench y los módulos de diseño, ejecutará la simulación para generar el archivo de ondas y abrirá GTKWave con la configuración predefinida para visualizar los resultados.

4.1.5. Para limpiar todo:

Luego de revisar las ondas y cuando se desee limpiar el directorio de trabajo, se usa el comando `make clean` para eliminar todos los archivos generados durante la compilación y la simulación.

5. Ejemplos de los resultados

Los resultados presentados incluyen simulaciones detalladas para cada uno de los cuatro modos de operación del SPI: Modo 01, Modo 10, Modo 00 y Modo 11. A continuación, se analizan las características particulares de cada modo, acompañadas de diagramas temporales para ilustrar el comportamiento de las señales durante la simulación. Cabe destacar que la señal end_protocol no se adjuntó en las capturas del GTKWave de este reporte por términos de espacio y extensión del mismo. Pero, si se corre la simulación, se pueden apreciar las señales en el Master y cada uno de los Slaves. Igualmente por términos de extensión no se agregó, pero al final de la simulación se dan cambios de polaridad en el modo reposo, para confirmar que efectivamente se comporte todo como se debe.

5.1. Modo 01 (CKP=0, CPH=1)

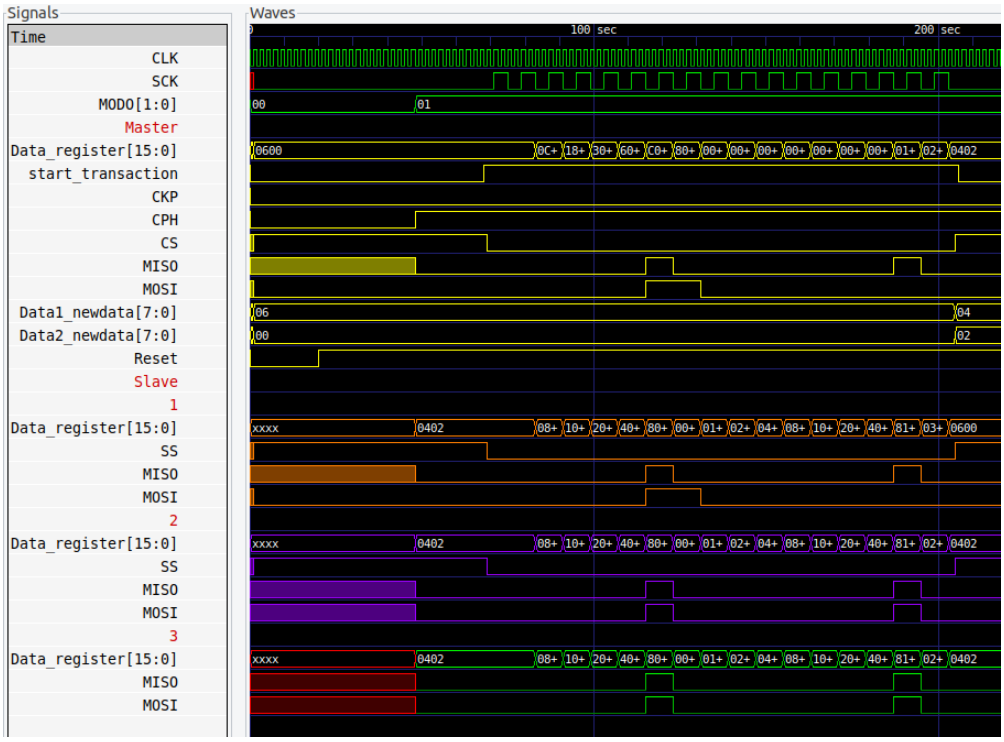

Figura 6: Simulación del Modo 01, donde las transiciones de datos ocurren en el flanco descendente del reloj y el reloj inicia en bajo.

5.2. Modo 10 (CKP=1, CPH=0)

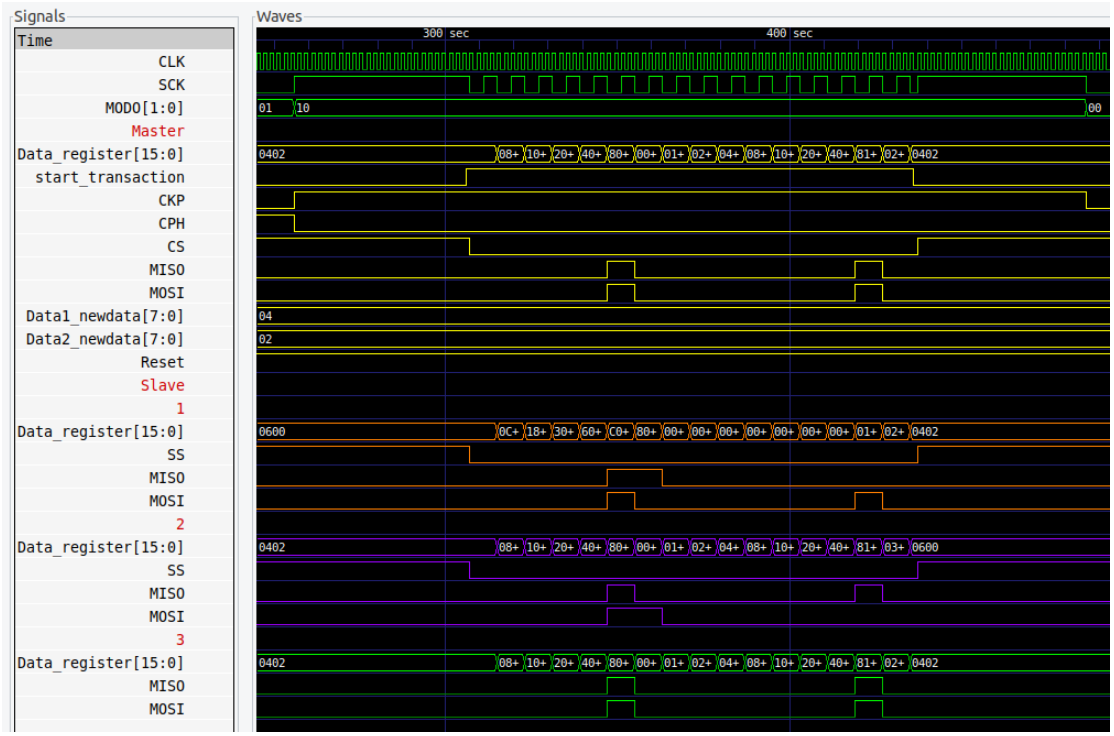

Figura 7: Simulación del Modo 10, donde las transiciones de datos ocurren en el flanco ascendente del reloj y el reloj inicia en alto.

5.3. Modo 00 (CKP=0, CPH=0)

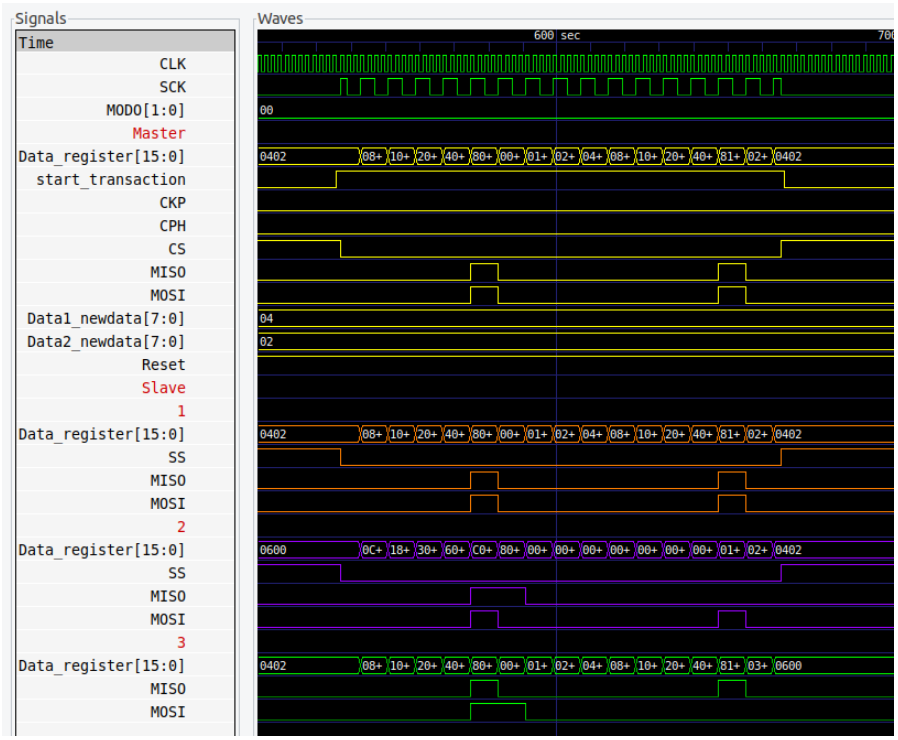

Figura 8: Simulación del Modo 00, caracterizado por transiciones de datos en el flanco ascendente del reloj y un reloj que inicia en bajo.

5.4. Modo 11 (CKP=1, CPH=1)

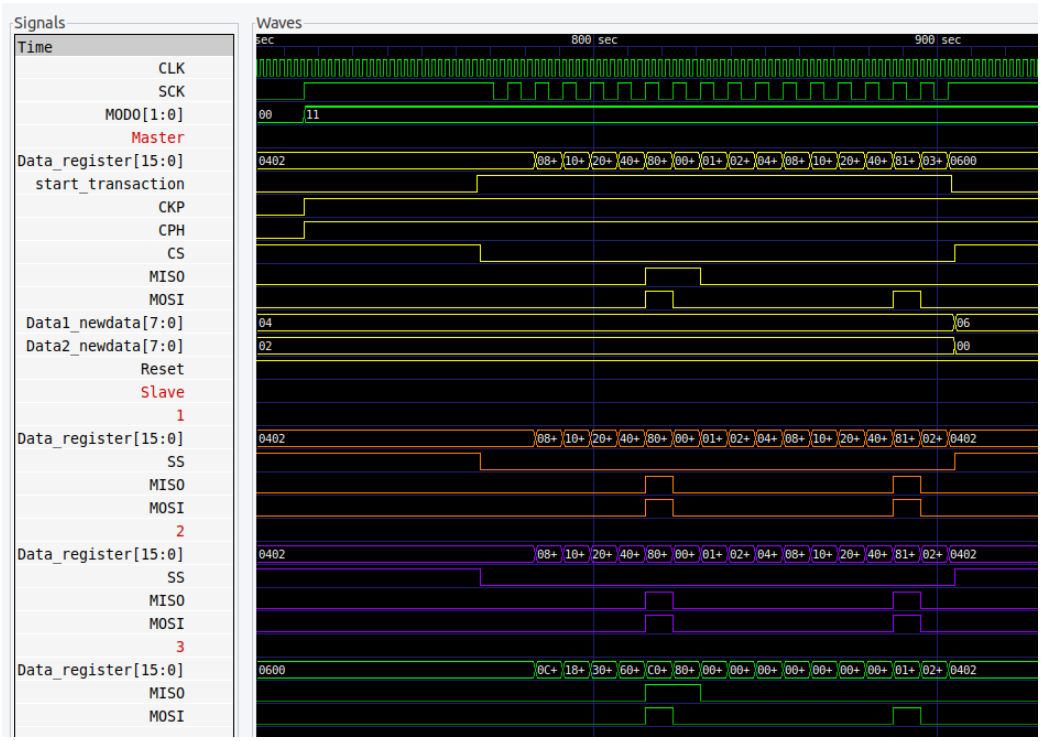

Figura 9: Simulación del Modo 11, donde las transiciones de datos ocurren en el flanco descendente del reloj y el reloj inicia en alto.

6. Análisis de los Resultados de Simulación

Tras realizar las pruebas de simulación en los cuatro modos de operación del SPI (01, 10, 00, 11), se observaron comportamientos consistentes y correctos en todas las configuraciones, validando así la funcionalidad del módulo SPI Master y Slave bajo diversas condiciones de sincronización del reloj.

6.1. Modo SPI 01 (CKP=0, CPH=1)

En este modo, la simulación confirmó que los datos se transmiten en el flanco descendente del reloj, con el reloj iniciando en estado bajo. Todos los datos enviados por el master fueron correctamente recibidos por el esclavo sin errores, cumpliendo con las expectativas de integridad de datos. Además se puede confirmar que el Reset se comportó como debía.

6.2. Modo SPI 10 (CKP=1, CPH=0)

El modo 10 mostró una perfecta sincronización en el flanco ascendente del reloj, con el reloj iniciando en estado alto. La transmisión y recepción de datos se realizaron sin discrepancias, y los datos fueron precisamente los que el master envió, verificando la correcta configuración de la polaridad del reloj.

6.3. Modo SPI 00 (CKP=0, CPH=0)

Este modo, que opera con ambos parámetros de reloj en bajo, fue exitoso en manejar las transmisiones de datos en el flanco ascendente del reloj. Los resultados mostraron una

operación correcta del sistema bajo esta configuración, con todos los datos transmitidos y recibidos intactos.

6.4. Modo SPI 11 (CKP=1, CPH=1)

Finalmente, el modo 11, con ambos parámetros en alto, funcionó correctamente, transmitiendo datos en el flanco descendente del reloj. La integridad y la sincronización de los datos fueron mantenidas a lo largo de todas las transmisiones.

6.5. Prueba en Configuración Daisy Chain

Además de las pruebas individuales, se realizó una prueba en configuración Daisy Chain conectando tres esclavos. En esta configuración, los datos iniciales 0600, que empezaron en el master, regresaron intactos al master después de la cuarta transmisión, habiendo pasado por cada esclavo correctamente. Además, se observó la correcta rotación de los datos 0402 entre los esclavos y el master, demostrando una transmisión y recepción eficaz a través de toda la cadena. Estos resultados por ende confirman la implementación efectiva del módulo SPI.

7. Detalles, retos y complicaciones durante la solución

El principal reto durante el desarrollo del diseño fue el reloj SCK para que funcionara como lo pedía el protocolo, ya que se tuvieron problemas para controlar el reloj correctamente, lo que llevó a crear la señal intermedia de 'MODO' para poder controlarlo adecuadamente. Otro reto fue que los bits registrados en la señal Data_register se atrasaban por uno luego de activarse la señal MOSI por primera vez, esto se arregló modificando la lógica de estados agregando un next_state para que no hubiera atrasos y las variables de Prox_data_register y Data_register pudieran actualizarse correctamente.

8. Conclusiones y Recomendaciones

En conclusión, el diseño implementado del módulo SPI Master y Slave ha demostrado su funcionalidad y eficacia bajo todas las configuraciones de modo de operación probadas (01, 10, 00, 11). La capacidad del sistema para manejar diferentes configuraciones de polaridad y fase del reloj sin errores en la transmisión y recepción de datos valida su robustez y adaptabilidad. Además, la prueba en configuración Daisy Chain confirmó la integridad de los datos a través de diferentes nodos, resaltando la escalabilidad y fiabilidad del diseño.

Las recomendaciones para futuros desarrollos o mejoras son las siguientes:

- **Planificación Detallada:** Implementar una fase de planificación inicial detallada antes de proceder con la estructuración del código. Esto debería incluir la definición de requisitos, la selección de metodologías de diseño y el establecimiento de un marco para pruebas preliminares.
- **Pruebas de Síntesis:** Realizar pruebas de síntesis para confirmar la viabilidad de construcción física del circuito propuesto. Esta aproximación proporciona una verificación temprana de que el diseño es no solo funcional en teoría, sino también práctico y realizable, además de evaluar el comportamiento del circuito con retardos para probarlo en un entorno más realista.