DISEÑO AVANZADO DE SISTEMAS DIGITALES - IPD 432

COMUNICACIÓN SPI EN NEXYS A7

FELIPE TORRES OLGUÍN 201821053-6

1 Introducción

La comunicación SPI (Serial Peripheral Interface) es un protocolo de comunicación síncrona ampliamente utilizado en sistemas digitales para la transferencia de datos entre dispositivos integrados, como microcontroladores, sensores y módulos de memoria. Su importancia radica en su simplicidad y eficiencia para la transmisión de datos a altas velocidades, permitiendo una comunicación directa y rápida entre un maestro y múltiples esclavos.

En el presente documento, se busca como objetivo principal describir un módulo correspondiente al maestro en la comunicación SPI, diseñando este módulo de manera que sea lo más general posible para interactuar con diferentes dispositivos esclavos. Este enfoque generalizado facilita la adaptabilidad del sistema a diversas aplicaciones y dispositivos, optimizando su reutilización en diferentes contextos.

Para validar la funcionalidad de este módulo maestro SPI, se implementará una prueba de comunicación con el acelerómetro ADXL362 [1] presente en la placa NEXYS A7 [2]. Esta tarea requiere una comprensión detallada de la información esperada por el acelerómetro desde el maestro SPI, lo que asegura una correcta configuración y transmisión de los datos.

El documento se estructura en cuatro secciones principales. La segunda sección se centra en una revisión detallada del protocolo de comunicación SPI. La tercera sección da consideraciones para la descripción de un maestro SPI generalizado, destacando su diseño y los parámetros configurables que permiten su adaptabilidad. En la cuarta sección, se expone la implementación específica de este módulo maestro SPI y otros módulos necesarios para establecer la comunicación con el acelerómetro ADXL362 de la NEXYS A7, detallando los pasos y consideraciones técnicas involucradas. Finalmente, la última sección presenta las conclusiones del estudio, resumiendo los resultados obtenidos y las posibles mejoras futuras para el sistema descrito.

2 Protocolo de comunicación SPI

El protocolo SPI se basa en una arquitectura maestro-esclavo, donde el dispositivo maestro controla la comunicación y los dispositivos esclavos responden a las instrucciones del maestro. La comunicación en SPI se lleva a cabo mediante cuatro conexiones principales asegurando una comunicación full-duplex [3].

SPI se compara frecuentemente con otros protocolos de comunicación, como I2C y UART. A diferencia de I2C, que utiliza un bus bidireccional y soporta múltiples maestros y esclavos en la misma línea, SPI utiliza líneas separadas para la transmisión y la recepción de datos, lo que permite una comunicación full-duplex y elimina la necesidad de arbitraje en el bus. Comparado con UART, que es un protocolo asíncrono y generalmente más lento, SPI ofrece mayores velocidades de transferencia y una comunicación síncrona más robusta.

Entre las ventajas más destacadas de SPI se encuentran su alta velocidad de transferencia, la simplicidad del protocolo y la capacidad de comunicación full-duplex. SPI puede operar a velocidades mucho mayores que otros protocolos seriales como I2C y UART, lo que lo hace ideal para aplicaciones que requieren un rápido intercambio de datos. La arquitectura de SPI es simple y directa, lo que facilita su implementación y configuración, y la capacidad de transmitir y recibir datos simultáneamente mejora la eficiencia en la comunicación. Sin embargo, SPI también presenta ciertas desventajas. Requiere más pines de conexión en comparación con I2C y UART, lo que puede ser un inconveniente en aplicaciones con limitaciones de espacio. Además, la gestión de múltiples esclavos en SPI puede complicar el diseño del sistema [4].

Como se mencionaba anteriormente, los dispositivos con comunicación SPI tienen cuatro señales principales: Reloj (SPI CLK, SCLK), Selección de chip (CS), Salida del maestro, entrada del esclavo (MOSI), y Entrada del maestro, salida del esclavo (MISO). En la Fig. 1 se observa una conexión entre maestro y esclavo SPI, donde se ven las cuatro conexiones necesarias para este protocolo.

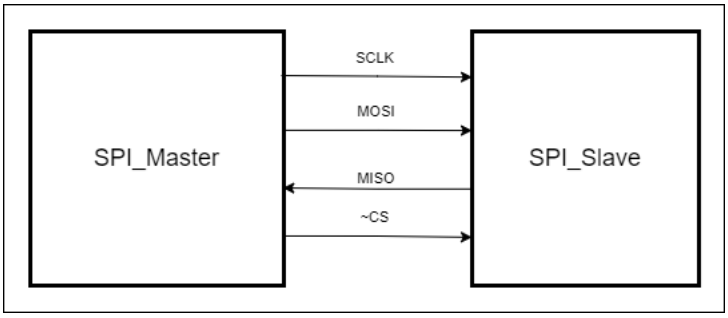

Figura 1: Configuración de conexiones protocolo SPI.

La señal de reloj es generada por el maestro con el fin de sincronizar el intercambio de información. La frecuencia de este reloj puede variar dependiendo de la frecuencia específica que requiera el esclavo. La conexión MOSI transmite datos del maestro al esclavo, mientras que MISO transmite datos del esclavo al maestro, en donde los datos se transmiten en paquetes de 1 byte, enviados bit a bit. Finalmente, la señal de selección de chip del maestro se utiliza para seleccionar el esclavo, en donde, esta señal es normalmente activa en bajo y se pone en alto para desconectar el esclavo del bus SPI. Cuando se utilizan varios esclavos, el maestro requiere una señal de selección de chip individual para cada esclavo, aunque existen otras opciones para gestionar múltiples esclavos.

La comunicación en un sistema SPI comienza con la caída de la señal de CS, activando el esclavo seleccionado para la transmisión de datos. A partir de este momento, el maestro genera la señal SCLK que sincroniza la transferencia de información. Los datos se transmiten desde el maestro al esclavo a través de la línea MOSI y, simultáneamente, el esclavo puede enviar datos de vuelta al maestro mediante la línea MISO. Cada bit de datos se sincroniza con los pulsos del reloj, asegurando una transferencia ordenada y precisa. La comunicación se mantiene activa mientras la señal CS esté en bajo, y finaliza cuando esta señal vuelve a alto.

Una de las características principales del protocolo SPI es que puede configurarse en cuatro modos diferentes según la polaridad (CPOL) y la fase (CPHA) del reloj, más detalles en la Fig. 2. Cada modo define en qué momento del ciclo del reloj se capturan y cambian los datos, permitiendo flexibilidad para diferentes dispositivos y aplicaciones.

- Modo 0 (CPOL = 0, CPHA = 0): El reloj está en bajo cuando está inactivo. Los datos se capturan en el flanco ascendente (de bajo a alto) del reloj y se cambian en el flanco descendente (de alto a bajo).
- Modo 1 (CPOL = 0, CPHA = 1): El reloj está en bajo cuando está inactivo. Los datos se capturan en el flanco descendente del reloj y se cambian en el flanco ascendente.
- Modo 2 (CPOL = 1, CPHA = 0): El reloj está en alto cuando está inactivo. Los datos se capturan en el flanco descendente del reloj y se cambian en el flanco ascendente.
- Modo 3 (CPOL = 1, CPHA = 1): El reloj está en alto cuando está inactivo. Los datos se capturan en el flanco ascendente del reloj y se cambian en el flanco descendente.

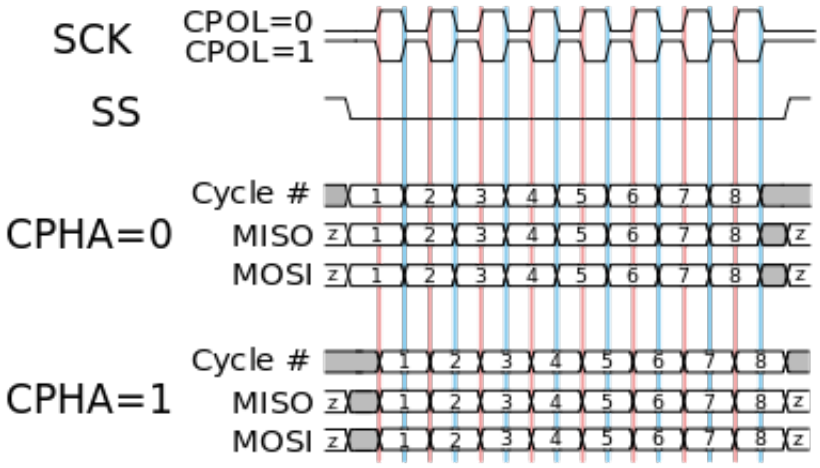

Figura 2: Diferente configuración de modos en SPI. [3]

La conexión del maestro con los esclavos SPI puede realizarse de varias maneras, cada una con sus ventajas y desventajas. La forma más sencilla es utilizar una señal CS individual para cada esclavo, donde el maestro activa la señal CS correspondiente al esclavo con el que desea comunicarse. Esto garantiza una selección precisa, pero requiere más pines en el maestro a medida que aumenta el número de esclavos. Otra opción es la conexión en cadena (*daisy chain*), donde los esclavos están conectados en serie y comparten una única línea de datos. En este caso, los datos fluyen secuencialmente a través de cada esclavo, lo que simplifica el cableado, pero puede aumentar la latencia y complicar la sincronización de datos, especialmente en sistemas con muchos esclavos. Finalmente, una solución más avanzada es usar una única señal CS combinada con un bloque combinacional que determine a qué esclavo se dirige la señal CS, mientras mantiene las señales CS de los demás esclavos en alto. Este enfoque reduce significativamente el número de pines necesarios en el maestro y permite una gestión dinámica de los esclavos, aunque requiere un diseño adicional en el circuito combinacional para asegurar una selección adecuada y evitar conflictos en la comunicación. En la figura 3, se ve cada tipo de conexión respectivamente mencionada.

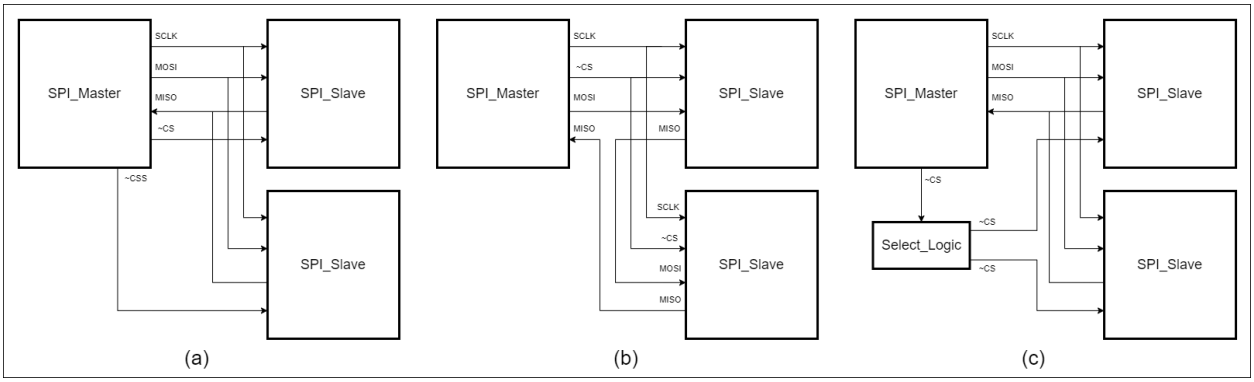

Figura 3: Configuración de las conexiones en SPI. (a) señal CS para cada esclavo, (b) *daisy chain* y (c) con lógica de selección.

3 Consideraciones para la descripción de un maestro SPI generalizado

La creación de un módulo maestro SPI capaz de generalizarse independientemente del esclavo requiere la identificación de las constantes y variables inherentes a este protocolo de comunicación. Al analizar

las cuatro conexiones principales, se observa que las más estables, independientemente de la cantidad de esclavos conectados, son SCLK, MOSI y MISO.

El SCLK debe ser un múltiplo del reloj principal del sistema. La frecuencia del reloj SPI debe ser configurable mediante contadores y debe poder ajustarse según el modo de operación definido por los parámetros CPOL y CPHA. Estos parámetros determinan el tipo de reloj que finalmente se genera a la salida del maestro, garantizando la correcta sincronización de los datos.

La señal MOSI, encargada de transmitir datos del maestro al esclavo, debe estar sincronizada con el reloj y el modo de operación. Esto puede lograrse utilizando *flags* que indiquen los flancos del reloj generado. Además, se debe permitir la configuración de la información contenida en el byte a transmitir, asegurando que el módulo pueda avisar cuando la transmisión se haya realizado con éxito y cuándo iniciar la siguiente transmisión.

Para la recepción de datos a través de la señal MISO, es esencial sincronizar la captura de los bits del byte recibido según los *flags* de los flancos del reloj, que dependerán del modo de operación configurado. El módulo debe ser capaz de almacenar el byte recibido y avisar cuando se haya completado la recepción.

En el caso del CS, esta señal es menos generalizable ya que depende del tipo de conexión que se establezca entre el maestro y los esclavos. Sin embargo, como directrices generales, es necesario asegurar que la dinámica antes mencionada de las tres otras señales del maestro (SCLK, MOSI y MISO) se cumpla cuando la señal CS está en su posición correcta. Esta dinámica debe volverse estática cuando la transmisión/recepción se haya completado o esté inactiva.

4 Implementación en SystemVerilog del protocolo SPI para NEXYS A7

En el diagrama de alto nivel de la Fig. 4, se observa la relación entre los módulos a implementar en el sistema. El módulo *SPI_Master* es el encargado de generalizar la comunicación con el esclavo, que en este caso es el acelerómetro ADXL362 en la FPGA NEXYS A7. La comunicación entre el maestro y el esclavo está definida por el módulo *Comunicacion_Acc*, que especifica los bytes a enviar y recibe los bytes del acelerómetro, para luego entregarlos al módulo *Seven_Seg_Logic*, donde se determina el formato de visualización en el *display* de la FPGA.

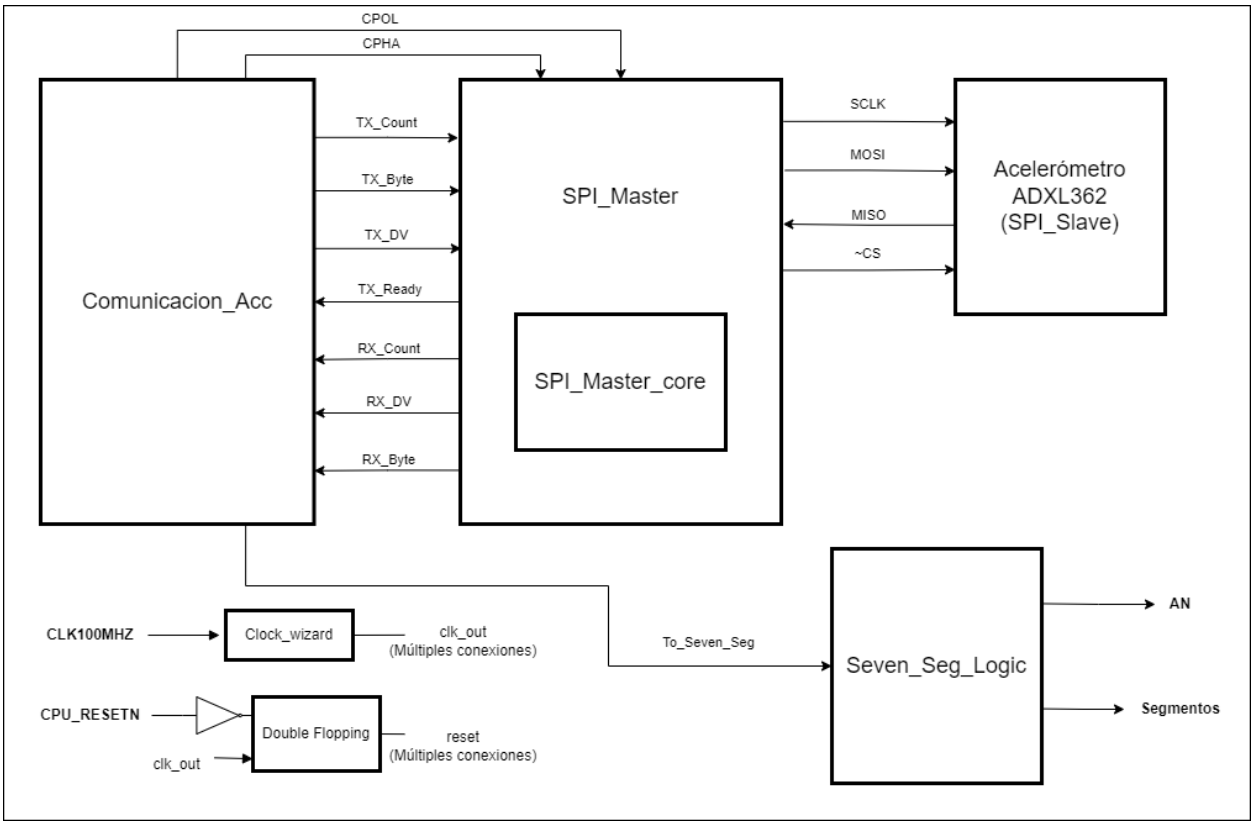

Figura 4: Diagrama de alto nivel de la implementación realizada.

El módulo *SPI_Master* controla la dinámica entre las señales de SCLK, MISO y MOSI con la señal de CS mediante una máquina de estados. Esta máquina asegura que las otras tres señales funcionen solo cuando la señal CS está en la posición correcta. Además, permite configurar cuántos bytes se espera enviar o recibir y la frecuencia del reloj SCLK. Este módulo tiene múltiples entradas y salidas, siendo las más importantes las cuatro conexiones que van al esclavo, los bytes de entrada y salida que transmite o recibe el maestro, y otras señales de *flags* que facilitan la comunicación y coordinación con el bloque que proporciona los tiempos e información al *SPI_Master*.

El módulo antes mencionado contiene al *SPI_Master_core*, que genera la señal de reloj que va al esclavo, considerando el modo de comunicación y la frecuencia necesaria para sincronizarse con el esclavo. Este módulo también contiene la lógica que envía y recibe información entre el maestro y el esclavo, coordinada por *flags* según el modo de operación.

El módulo *Comunicacion_Acc* se encarga de proporcionar la información que el *SPI_Master* debe entregar al esclavo y de coordinar los tiempos de espera que requiere el acelerómetro. En la Fig. 5, se presenta un esquema de la lógica requerida para comunicarse con el acelerómetro. Esta lógica requiere la lectura de la hoja de datos del componente [1] para obtener información sobre cómo configurar el acelerómetro en modo de toma de datos, qué instrucciones enviar, qué direcciones leer, cuánto esperar al energizar el acelerómetro, los tiempos entre cada lectura y las direcciones que almacenan la información de los tres ejes. Con esta información, se puede implementar una máquina de estados que dará las señales al maestro y guardará la información recibida del esclavo. Otra función de este módulo es entregar la información de los tres ejes al módulo *Seven_Seg_Logic*.

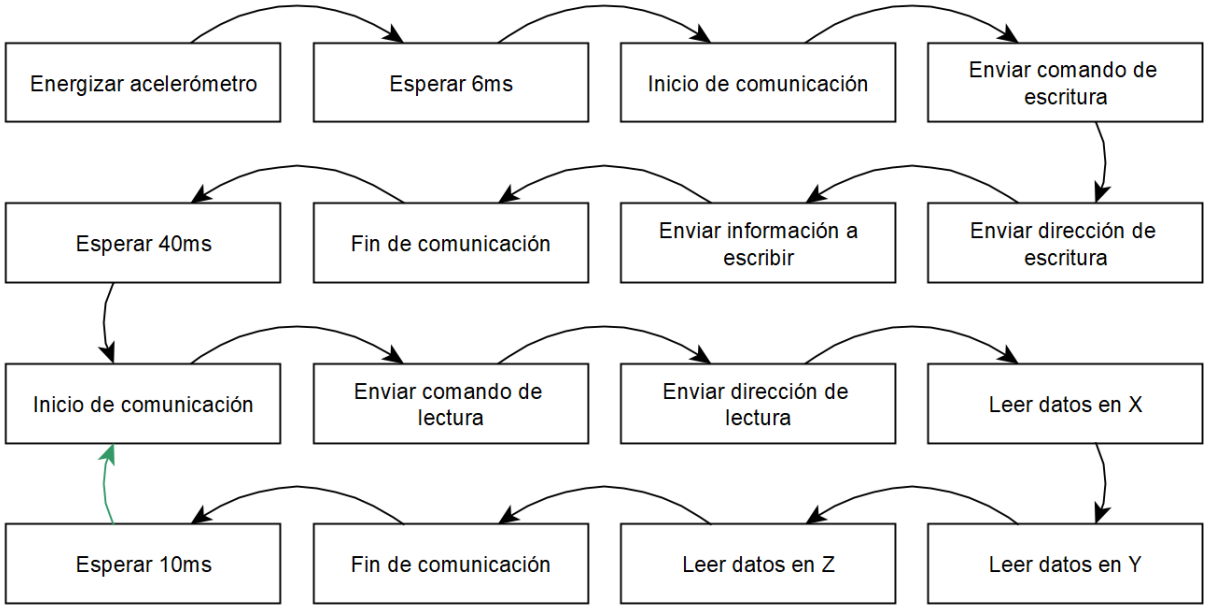

Figura 5: Lógica requerida para la comunicación con el acelerómetro.

El módulo *Seven_Seg_Logic* recibe un bus de información con los datos medidos en los tres ejes, correspondientes a 16 bits por eje. La hoja de datos del acelerómetro especifica qué bits corresponden a qué tipo de información (signo, MSB o LSB). Con la información recibida y la cantidad de bits a mostrar seleccionados para el *display*, el módulo realiza la conversión de *unsigned* a BCD y da las señales necesarias para visualizar el valor de cada eje en el *display*.

5 Resultados y conclusiones

Se cumplió con el objetivo de implementar un módulo maestro SPI lo más general posible, comprobando su funcionamiento al ver la información del *display* cambiar a medida que se rotaba la FPGA en las tres direcciones de medición, además de verificar la comunicación SPI en sus cuatro conexiones con el analizador lógico.

El módulo *SPI_Master* es capaz de cambiar su modo de operación, la cantidad de bytes a enviar, la información de los bytes a enviar, la frecuencia del reloj, entre otros parámetros configurables. La generalización realizada para el maestro SPI constituye la mitad del trabajo necesario para lograr una comunicación efectiva con cualquier esclavo SPI, ya que la otra mitad depende de la lógica específica que requiere cada esclavo, es decir, la información que necesita recibir y los tiempos entre transmisiones.

La implementación se realizó considerando el caso de una FPGA con un maestro y un esclavo, en este caso un acelerómetro. Sin embargo, este bloque maestro SPI requeriría modificaciones si se tuvieran más esclavos SPI. Estas modificaciones deberían centrarse solo en el módulo *SPI_Master*, que es el encargado de gestionar la señal CS, mientras que el módulo *SPI_Master_core* puede permanecer inalterado independientemente del número de esclavos.

Como trabajo futuro, se recomienda generalizar el módulo *SPI_Master* para ser capaz de comunicarse con múltiples esclavos, teniendo en cuenta las configuraciones de conexión (a) y (b) de la Fig. 3, ya que en esta implementación solo se consideró el tipo (c). Dado que solo había un esclavo en esta implementación, no fue necesario incluir un bloque de lógica adicional. Además, se podría probar la implementación realizada para múltiples esclavos para verificar su funcionalidad y robustez.

Finalmente, otras posibles líneas de desarrollo a partir de este proyecto incluyen la modificación del módulo *Comunicacion_Acc* para probar otras funcionalidades del acelerómetro, o utilizar la implementación actual para aplicaciones distintas, como mover un servomotor, encender LEDs, interactuar con lógica en

una pantalla mediante VGA, entre otros. Estas expansiones permitirían explorar y aprovechar al máximo las capacidades del sistema SPI en diversas aplicaciones prácticas y experimentales.

References

[1] A. Devices, "ADXL362 Micropower, 3-Axis, ±2 g/±4 g/±8 g Digital Output MEMS Accelerometer," May 2023. [Online]. Available: https://www.analog.com/en/products/adxl362.html

[2] Digilent, "Nexys A7 Reference Manual," May 2023. [Online]. Available: https://digilent.com/reference/programmable-logic/nexys-a7/reference-manual

[3] P. Dhaker, "Introduction to SPI Interface," Sep 2018. [Online]. Available: https://www.analog.com/en/resources/analog-dialogue/articles/introduction-to-spi-interface.html

[4] J. Chen and S. Huang, "Analysis and Comparison of UART, SPI and I2C," in *2023 IEEE 2nd International Conference on Electrical Engineering, Big Data and Algorithms (EEBDA)*, 2023, pp. 272–276.