PROTOCOLO SPI

La interfaz de periféricos en serie (SPI) es un bus de interfaz utilizado habitualmente para enviar datos entre microcontroladores y pequeños periféricos como registros de desplazamiento, sensores y tarjetas SD. Utiliza líneas de reloj y datos separadas, junto con una línea de selección para elegir el dispositivo con el que se desea hablar.

Dentro de este protocolo se define un maestro que será aquel dispositivo encargado de transmitir información a sus esclavos. Los esclavos serán aquellos dispositivos que se encarguen de recibir y enviar información al maestro. El maestro también puede recibir información de sus esclavos, cabe destacar. Para que este proceso se haga realidad es necesario la existencia de dos registros de desplazamiento, uno para el maestro y uno para el esclavo respectivamente. Los registros de desplazamiento se encargan de almacenar los bits de manera paralela para realizar una conversión paralela a serial para la transmisión de información.

Modo de funcionamiento

Presenta 4 señales lógicas principales

- **MOSI (Master Out Slave In):** Línea utilizada para llevar los bits que provienen del maestro hacia el esclavo.
- **MISO (Master in Slave Out):** Línea utilizada para llevar los bits que provienen del esclavo hacia el maestro.
- **CLK (Clock).** Línea proveniente del maestro encarga de enviar la señal de reloj para sincronizar los dispositivos.
- **SS (Slave Select):** Línea encargada de seleccionar y a su vez, habilitar un esclavo.

En nuestro caso estamos utilizando un solo esclavo el cual es la memoria flash M25P16.

Durante cada ciclo de reloj SPI, se produce una transmisión full-dúplex de un solo bit. El principal envía un bit en la línea MOSI mientras que el secundario envía un bit en la línea MISO, y luego cada uno lee su bit entrante correspondiente. Esta secuencia se mantiene incluso cuando sólo se pretende una transferencia de datos unidireccional.

Polaridad y fase del reloj

Además de configurar la frecuencia del reloj, el principal también debe configurar la polaridad y la fase del reloj con respecto a los datos.

- CPOL representa la polaridad del reloj. Las polaridades se pueden convertir con un simple inversor .
  - SCLK $_{CPOL=0}$ es un reloj que está inactivo con un voltaje lógico bajo .
  - SCLK $_{CPOL=1}$ es un reloj que está inactivo con el alto voltaje lógico.
- CPHA representa la fase del ciclo de transmisión de cada bit de datos en relación con SCLK.
  - Para CPHA=0:
    - El primer bit de datos se emite *inmediatamente* cuando se activa CS.
    - Los bits posteriores se emiten cuando SCLK pasa *a* su nivel de voltaje inactivo.
    - El muestreo ocurre cuando SCLK pasa *de* su nivel de voltaje inactivo.
  - Para CPHA=1:
    - El primer bit de datos se emite en el primer flanco del reloj de SCLK *después de que se activa* CS.
    - Los bits posteriores se emiten cuando SCLK pasa *de* su nivel de voltaje inactivo.
    - El muestreo ocurre cuando SCLK pasa *a* su nivel de voltaje inactivo.

La M25P16 es una memoria serial paginada de 16Mb. El dispositivo es accedido a través de su compatibilidad con el protocolo SPI de alta velocidad.

Figure 1: Logic Diagram

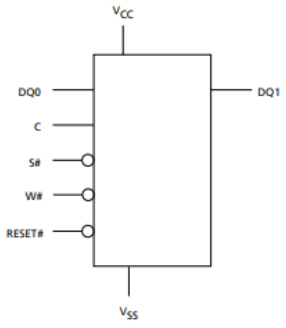


Table 1: Signal Names

Signal Name	Function	Direction
C	Serial clock	Input
DQ0	Serial data input	Input
DQ1	Serial data output	Output
S#	Chip select	Input
W#	Write protect	Input
RESET#	Reset	Input
V_{CC}	Supply voltage	–
V_{SS}	Ground	–

En el diagrama lógico del dispositivo se logra observar que este consta de 5 pines de entrada y 1 pin de salida.

Entre los pines de entrada se encuentra el reloj con el cual trabaja la memoria el cual debe estar sincronizado con el reloj del fpga para el correcto funcionamiento de las funciones de lectura y escritura en memoria.

Además, presenta la entrada DQ0 la cual corresponde a la entrada de comunicación de la memoria a través de el bus spi de datos en serie, este pin tiene la misma función que el pin de DQ1 solo que el primero es para la comunicación Memoria-FPGA y el segundo para la comunicación FPGA-Memoria.

El pin de Chip select, en un principio su aplicación es para elegir entre varios, el dispositivo con el cual se estará comunicando, sin embargo, su importancia va más allá debido a que toda instrucción para poder ser leída por la memoria se necesita, primeramente, que este pin sea activado, en este caso a 0.

Presenta además otros pines como la protección contra escritura para el tratamiento de sectores de memoria los cuales tienen un acceso con ciertas especificaciones, la alimentación de la memoria y tierra.

SPI Modes

These devices can be driven by a microcontroller with its serial peripheral interface (SPI) running in either of the following two SPI modes:

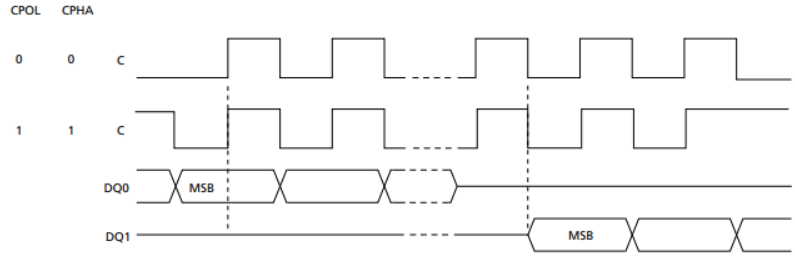
- CPOL=0, CPHA=0
- CPOL=1, CPHA=1

For these two modes, input data is latched in on the rising edge of serial clock (C), and output data is available from the falling edge of C.

The difference between the two modes is the clock polarity when the bus master is in STANDBY mode and not transferring data:

- C remains at 0 for (CPOL=0, CPHA=0)
- C remains at 1 for (CPOL=1, CPHA=1)

Figure 3: SPI Modes Supported



Modos de trabajo SPI de la memoria

Después de haberse hablado, en general de los modos de trabajo del protocolo SPI, pasaremos al caso particular de la memoria M25P16, en la cual

principalmente existen 2 modos de Trabajo. CPOL=CPHA=0 y CPOL=CPHA=1, lo que nos indica que la memoria siempre, en cualquiera de los modos que esté trabajando recibirá los bits e el frente de subida del reloj, luego como se menciono anteriormente la polaridad del reloj es irrelevante ya que puede ser cambiada simplemente colocando un inversor.

Disposición de la memoria ROM:

Como se dijo anteriormente la memoria se encuentra dividida en páginas. Cada una de estas paginas puede ser programada individualmente.

Se encuentra organizada en 32 sectores, cada uno de estos conteniendo 256 paginas y cada paginas contiene en su interior 256 bytes. Es importante recalcar que cada dirección de memoria esta constituida por 3 bytes.

En la figura se muestra uno de los sectores de la memoria

Sector	Address Range	
	Start	End
31	001F 0000	001F FFFF
30	001E 0000	001E FFFF
29	001D 0000	001D FFFF
28	001C 0000	001C FFFF
27	001B 0000	001B FFFF
26	001A 0000	001A FFFF
25	0019 0000	0019 FFFF
24	0018 0000	0018 FFFF
23	0017 0000	0017 FFFF
22	0016 0000	0016 FFFF
21	0015 0000	0015 FFFF
20	0014 0000	0014 FFFF
19	0013 0000	0013 FFFF
18	0012 0000	0012 FFFF

Frecuencia de Trabajo de la operaciones de la memoria

Table 24: AC Specifications (50 MHz operation)

Test conditions are specified in the Operating Conditions and AC Measurement Conditions tables.

Symbol	Alt.	Parameter	Min	Typ	Max	Unit	Notes
f_C	f_C	Clock frequency for following commands: FAST_READ, RDLR, PW, PP, WRLR, PE, SE, SSE, DP, RDP, WREN, WRDI, RDSR, WRSR	D.C.	–	50	MHz	
f_R	–	Clock frequency for READ command	D.C.	–	33	MHz	

Como se observa en la figura la frecuencia de reloj máxima es diferente para si vamos a escribir en memoria o vamos a leer de memoria, por lo que será necesario un regulador de reloj (divisor de frecuencia), para lograr adecuar el circuito a las acciones que deseemos realizar en cada momento. Para la escritura en memoria la frecuencia máxima de reloj es 50MHz mientras que para la lectura de memoria la frecuencia máxima es de 33MHz.

La frecuencia del reloj que presenta el FPGA es, originalmente de 55MHz, de ahí la necesidad de reducirla para el correcto funcionamiento de la memoria.

Funcionamiento General De los Comandos.

- Todos los comandos, direcciones y datos son rotados dentro y fuera de la memoria, el byte más significativo primero.
- Los inputs de datos DQ0 y DQ1son muestreados en el primer flanco de subida de reloj después de que #CS es activado
- Luego el comando de un byte correspondiente a la instrucción debe ser desplazado hacia el dispositivo (MSB primero), cada bit en un flanco de subida de reloj

Dependiendo del comando que sea utilizado, este debe ser continuado por una dirección, un set de datos, ambos o ninguno de estos. (Sin desactivar el CS)

Para los comandos de lectura de memoria, luego de introducido el comando de la instrucción, es seguido por una secuencia de datos salientes. CS puede ser desactivado en cualquier momento luego de que un bit haya sido desplazado por la salida de la memoria.

Para los comandos de escritura en memoria, CS debe ser desactivado exactamente después de que se halla enviado un múltiplo de 8 pulsos de reloj, de no ser así la instrucción será cancelada y no abra modificaciones en la memoria

Command Set Overview

All commands, addresses, and data are shifted in and out of the device, most significant bit first.

Serial data inputs DQ0 and DQ1 are sampled on the first rising edge of serial clock (C) after chip select (S#) is driven LOW. Then, the one-byte command code must be shifted in to the device, most significant bit first, on DQ0 and DQ1, each bit being latched on the rising edges of C.

Every command sequence starts with a one-byte command code. Depending on the command, this command code might be followed by address or data bytes, by address and data bytes, or by neither address or data bytes. For the following commands, the shifted-in command sequence is followed by a data-out sequence. S# can be driven HIGH after any bit of the data-out sequence is being shifted out.

- READ DATA BYTES (READ)
- READ DATA BYTES at HIGHER SPEED
- READ IDENTIFICATION
- READ STATUS REGISTER
- READ TO LOCK REGISTER

For the following commands, S# must be driven HIGH exactly at a byte boundary. That is, after an exact multiple of eight clock pulses following S# being driven LOW, S# must be driven HIGH. Otherwise, the command is rejected and not executed.

- PAGE WRITE
- PAGE PROGRAM
- WRITE to LOCK REGISTER
- PAGE ERASE
- SUBSECTOR ERASE
- SECTOR ERASE
- BULK ERASE
- WRITE STATUS REGISTER
- WRITE ENABLE
- WRITE DISABLE
- DEEP POWER-DOWN
- RELEASE FROM DEEP POWER-DOWN

All attempts to access the memory array are ignored during a WRITE STATUS REGISTER command cycle, a PROGRAM command cycle, or an ERASE command cycle. In addition, the internal cycle for each of these commands continues unaffected.

La lista de todas las instrucciones que pueden realizarse en la memoria es la siguiente, sin embargo, no todas estas instrucciones serán utilizadas en este proyecto, ni explicadas en la exposición, de todas estas instrucciones solo nos quedaremos con las de habilitación y deshabilitacion de escritura, instrucciones necesarias para escribir en memoria y la instrucción de page program, encargada de enviar las instrucciones a la memoria. Por otra parte, también tendremos la

instrucción de Read Data Byte, instrucción encargada de leer de memoria. Estas instrucciones serán Expuestas más a detalle a continuación.

Table 6: Command Set Codes

Command Name	One-Byte Command Code		Bytes		
			Address	Dummy	Data
WRITE ENABLE	0000 0110	06h	0	0	0
WRITE DISABLE	0000 0100	04h	0	0	0
READ IDENTIFICATION	1001 1111	9Fh	0	0	1 to 3
READ STATUS REGISTER	0000 0101	05h	0	0	1 to ∞
WRITE STATUS REGISTER	0000 0001	01h	0	0	1
WRITE to LOCK REGISTER	1110 0101	E5h	3	0	1
READ LOCK REGISTER	1110 1000	E8h	3	0	1
READ DATA BYTES	0000 0011	03h	3	0	1 to ∞
READ DATA BYTES at HIGHER SPEED	0000 1011	0Bh	3	1	1 to ∞
PAGE WRITE	0000 1010	0Ah	3	0	1 to 256
PAGE PROGRAM	0000 0010	02h	3	0	1 to 256
PAGE ERASE	1101 1011	DBh	3	0	0
SUBSECTOR ERASE	0010 0000	20h	3	0	0
SECTOR ERASE	1101 1000	D8h	3	0	0
BULK ERASE	1100 0111	C7h	0	0	0
DEEP POWER-DOWN	1011 1001	B9h	0	0	0
RELEASE from DEEP POWER-DOWN	1010 1011	ABh	0	0	0

Command Name	One-Byte Command Code		Bytes		
			Address	Dummy	Data
WRITE ENABLE	0000 0110	06h	0	0	0
WRITE DISABLE	0000 0100	04h	0	0	0
READ DATA BYTES	0000 0011	03h	3	0	1 to ∞
PAGE WRITE	0000 1010	0Ah	3	0	1 to 256

Proceso de escritura en memoria

El proceso de escritura en memoria es por mucho, mas complejo que el proceso de lectura que será especificado más adelante.

La primera instrucción que debe ser mandada a la memoria es la instrucción de habilitación de escritura:

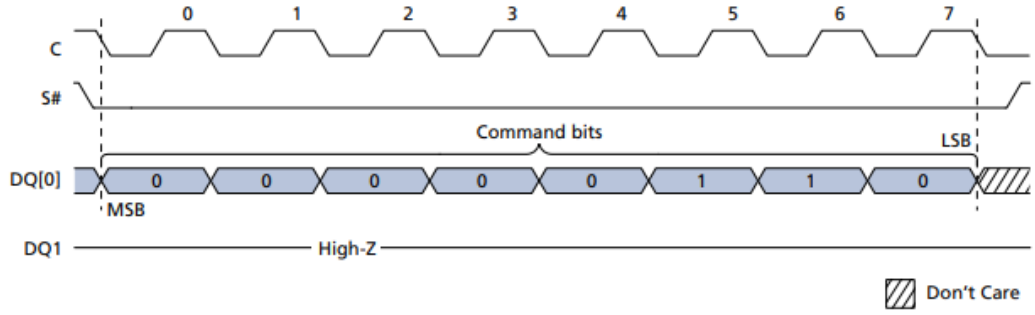
Habilitación PARA ESCRITURA

ACTIVA EL BIT DE WEL

El comando de write enable habilita el bit de WEL. Este bit debe estar obligatoriamente activado cuando se realice cualquier operación de escritura en página o escritura en programa, incluye además otras como el borrado de sector escritura en el registro de estados y otras que no serán referidas

Este comando es introducido en el chip luego de activar el chip select enviando el código correspondiente de la instrucción.

Figure 6: WRITE ENABLE Command Sequence



Page Write

Es una vía conveniente para la manipulación de datos (hasta 1 sector) dentro de la memoria.

Primeramente, se envía el byte correspondiente a la instrucción. Luego se envían los 3 bytes de dirección donde se desea iniciar la escritura. Comenzando primeramente por el bit mas significativo hasta el menos significativo.

Por ultimo se envía, como mínimo 1 byte de datos a la memoria, el cual será escrito en esa dirección de memoria.

Luego de enviado el dato se desactiva el chi select

Simple Data Modification

The Page Write (PW) instruction provides a convenient way of modifying data (up to 256 contiguous bytes at a time), and simply requires the start address, and the new data in the instruction sequence.

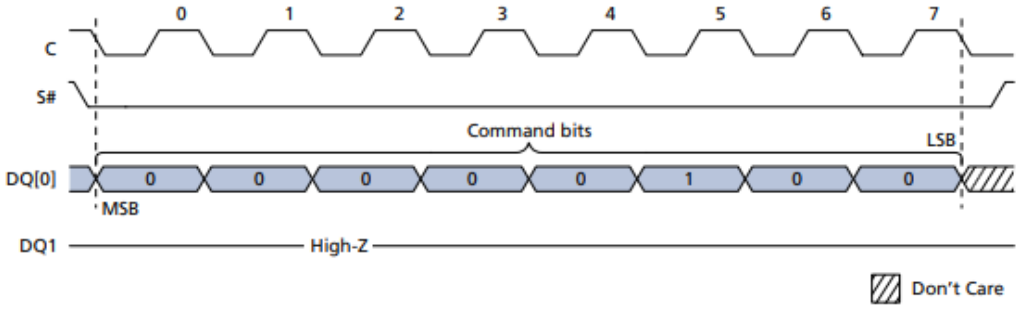
The Page Write (PW) instruction is entered by driving Chip Select (S#) LOW, and then transmitting the instruction byte, three address bytes (A23-A0) and at least one data byte, and then driving S# HIGH. While S# is being held LOW, the data bytes are written to the data buffer, starting at the address given in the third address byte (A7-A0). When Chip S# is driven HIGH, the Write cycle starts. The remaining unchanged bytes of the data buffer are automatically loaded with the values of the corresponding bytes of the addressed memory page. The addressed memory page is then automatically put into an Erase cycle. Finally, the addressed memory page is programmed with the contents of the data buffer.

All of this buffer management is handled internally, and is transparent to the user. The user is given the facility of being able to alter the contents of the memory on a byte-by-byte basis. For optimized timings, it is recommended to use the PAGE WRITE (PW) instruction to write all consecutive targeted bytes in a single sequence versus using several PAGE WRITE (PW) sequences with each containing only a few bytes.

DESABILITACION DE ESCRITURA UNA VEZ FUE ESCRITA UNA PAGINA

La función de la deshabilitacion de escritura es indicar a la memoria que ha terminado el proceso de escritura en memoria y permitirá a esta recibir nuevos comando de instrucciones

Figure 7: WRITE DISABLE Command Sequence

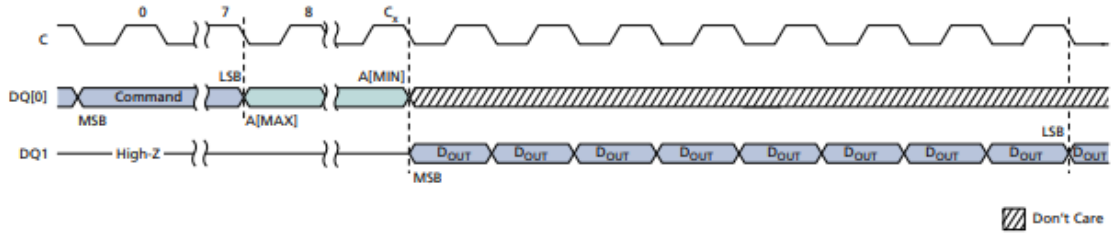


Read Data Bytes (Proceso de lectura)

El proceso de lectura es mucho mas simple que el de escritura, cuenta, en su forma más básica de cuatro etapas, primera mente se envían los 8 bits correspondientes a la instrucción de read data bytes, donde el contenido de memoria de la dirección que se especificará en los próximos 3 bytes será rotado fuera de la memoria a través de dq1, cada bit en un frente positivo de reloj.

Una observación es que la dirección de memoria es automáticamente incrementada luego de que haya sido leída con éxito la anterior.

Figure 14: READ DATA BYTES Command Sequence



Posibles Bloques Lógicos para la implementación:

1-Generador de reloj: genera la señal de reloj necesaria para sincronizar la comunicación SPI entre el FPGA y el dispositivo esclavo.

2- Divisor de frecuencia: divide la frecuencia del reloj para obtener la frecuencia deseada para la comunicación. Útil si la frecuencia del reloj es demasiado alta para el dispositivo.

3- Controlador de selección de esclavos: genera la señal SS para seleccionar la memoria M25P16 durante las operaciones de lectura y escritura.

4- Controlador de transmisión de datos: gestiona la transferencia de datos a través de la línea MOSI y controla el tiempo de transmisión según SPI.

5- Controlador de recepción de datos: gestiona la transferencia de datos a través de la línea MISO y el tiempo de recepción según SPI.

6- Controlador de estado: implementa la lógica de la máquina de estados finitos necesaria para controlar el flujo de comunicación SPI. Supervisa el estado actual y la transición entre estados según la señal de control y la temporización.

7**- Buffer de datos**: Almacenar temporalmente los datos que se transmiten entre el FPGA y el dispositivo esclavo (Registros).