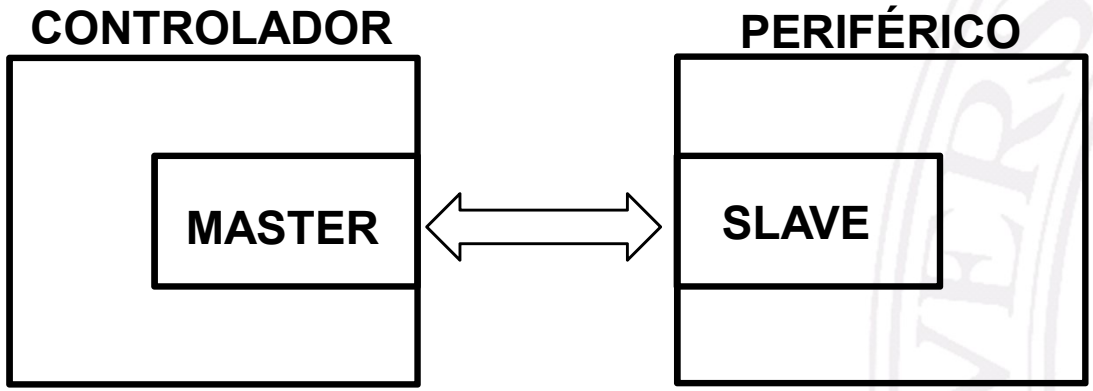
Diseño Digital I

Diseño y modelado de un Master SPI

Interconexión de sensores y periféricos a controladores

- Buses serie síncronos:
  - I2C, SPI
  - Interfaces con pocas líneas ☺
  - Baja velocidad 😐
  - Protocolos de trama relativamente complejos 😐
- Dos tipos de interfaces: MAESTROS y ESCLAVOS
  - El MAESTRO dirige las transferencias
  - El ESCLAVO recibe los datos (escritura) o los entrega (lectura)



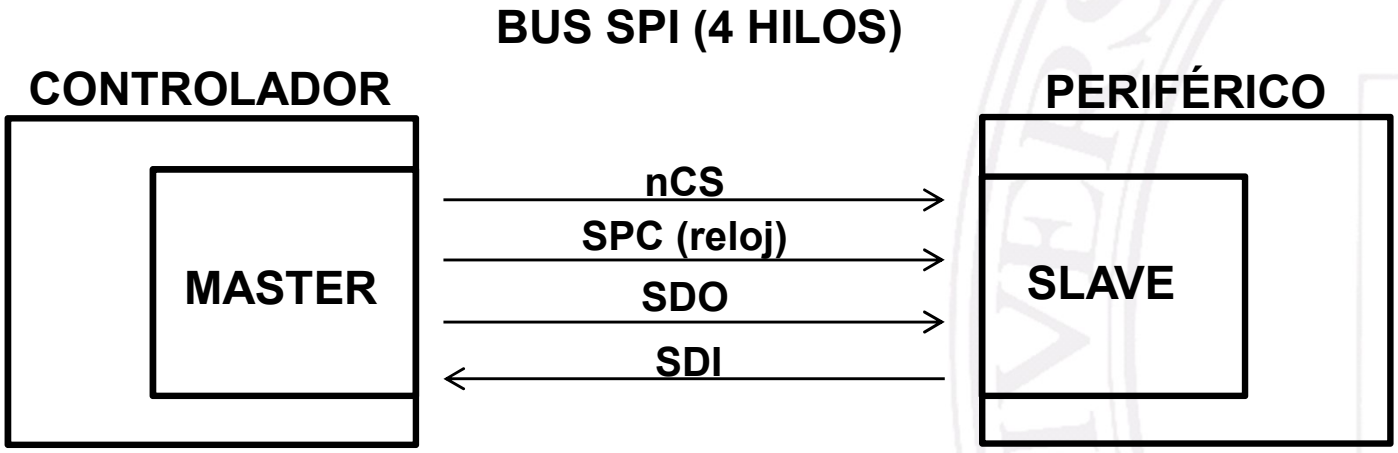
I2C vs SPI

- I2C
  - Estándar de facto, muy formalizado
  - Interfaz a 2 hilos, *half-duplex*
  - 400 Kbps (I2C Fast) o 5 Mbps (I2C UltraFast)
  - Hasta 127 circuitos compartiendo el bus (con 2 hilos)

- SPI
  - Estándar de facto, menos formalizado (variantes)
  - Interfaz a 3 hilos o 4 hilos, *full-duplex*
  - Velocidad máxima no especificada (+10 Mbps)
  - N circuitos compartiendo el bus
    - con N+3 hilos (N es el número de CS)
    - con 4 hilos (en modo Daisy-chain)

Buses SPI a 3 o 4 hilos

- Variante a 3 hilos
  - reloj (SPC)
  - *chip select* (nCS)
  - datos (SDI_O, bidireccional)
- Variante a 4 hilos
  - reloj (SPC)
  - *chip select* (nCS)
  - Entrada de datos (SDI, MISO)
  - Salida de datos (SDO, MOSI)



Características del bus SPI

- Múltiples variantes
  - Características comunes:

    1.- Duración de una transferencia = tiempo que el *maestro* mantiene nCS activa.
    2.- Durante la transferencia el *master* genera la señal de reloj del bus (SPC).
    3.- Cuando no hay una transferencia en curso, la línea de reloj (SPC) está en reposo (0 o 1)
    4.- En cada periodo de SPC se transfiere un bit.
      - Escrituras en uno de los flancos de SPC (subida o bajada)
      - Lecturas en el flanco contrario
  - Diferencias (compartidas por maestro y esclavo):

    1.- CPOL (Clock Polarity): CPOL = 0 -> nivel bajo en reposo; CPOL = 1 -> nivel alto en reposo
    2.- CPHA (Clock Phase): flanco de SPC en el que se escriben los datos

CPOL	CPHA	Flanco de SPC en el que se **escriben** los datos
0	0	Bajada (el primer dato tras la bajada de nCS)
0	1	Subida
1	0	Subida (el primer dato tras la bajada de nCS)
1	1	Bajada

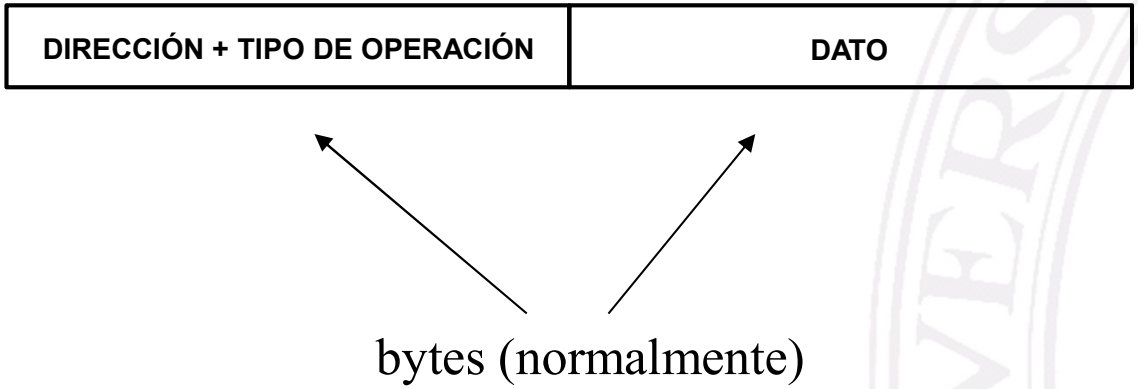
    3.- Número de hilos: 3 o 4
    4.- Formato y duración de las tramas

Tramas SPI (I)

- Transferencias: acceso del master (lectura o escritura) a los registros direccionables del esclavo

- Transferencia de un dato:

  - El master envía la dirección del registro y el tipo de operación (R o W)
  - W: el master escribe un dato
  - R: el master lee un dato
  - Primero el MSb (normalmente)



Tramas SPI (II)

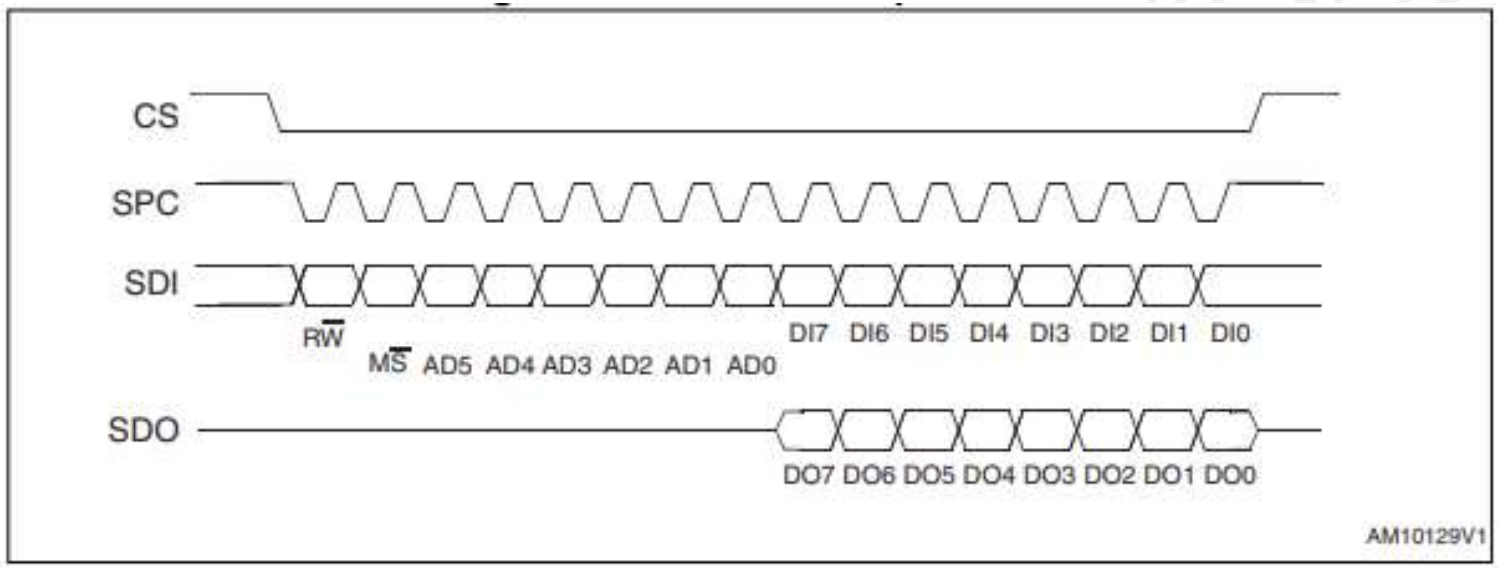
- Transferencias múltiples (en muchos casos):
  - Frecuentemente el *esclavo* funciona con un puntero con autoincremento
  - Al inicio de una transferencia el puntero toma el valor de la dirección indicada por el maestro
  - El primer dato transferido (por el *maestro* o el *esclavo*) corresponde a la dirección enviada por el *maestro*
  - Mientras el *maestro* prolongue la duración de la transferencia el puntero se va incrementando dando acceso a un nuevo registro

DIRECCIÓN N	DATO N	DATO N+1	DATO N+2

Diseño de un *master* SPI para el acelerómetro LIS2DH12 (I)

Circuito digital síncrono capaz de comunicarse con un acelerómetro **LIS2DH12** de *STMicroelectronics*, capaz de realizar medidas de aceleración en tres ejes con una periodicidad y rango de aceleraciones configurables.

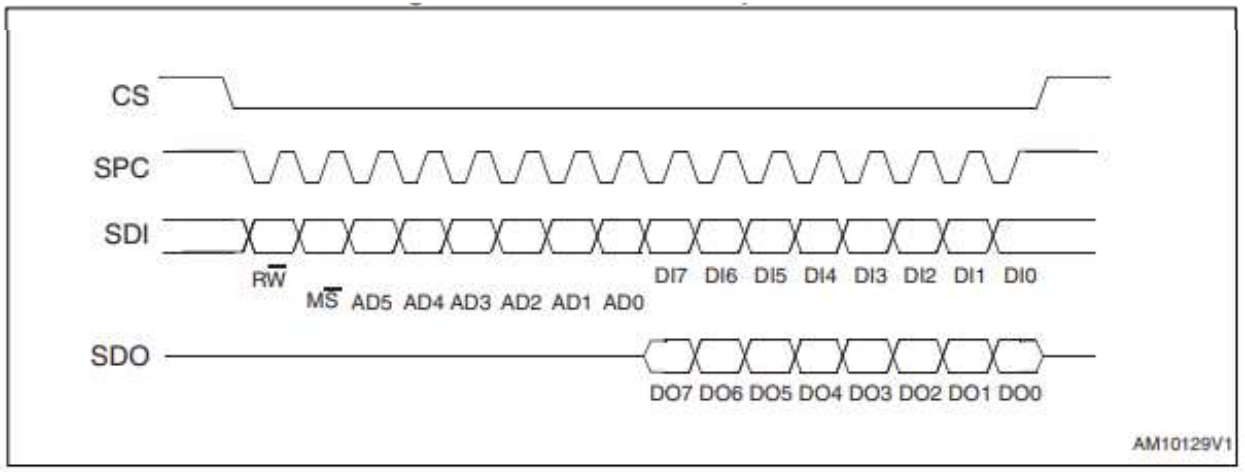
El acelerómetro cuenta con una interfaz esclava SPI cuyo funcionamiento básico ilustra la figura adjunta.



Diseño de un *master* SPI para el acelerómetro LIS2DH12 (II)

De la figura puede deducirse inmediatamente que:

- CS es activo a nivel bajo
- La interfaz puede funcionar a 4 hilos (se puede configurar a 3) y emplea polaridad 1 y fase 1



De la hoja de datos del chip se extrae la siguiente información adicional:

- Primer byte:
  - RW indica transferencia de lectura (1) o escritura (0)
  - MS indica si el esclavo debe (1) o no (0) autoincrementar el puntero
  - AD5-AD0 es la dirección del registro al que el master quiere acceder

- Segundo byte:
  - escritura: el maestro transmite por la línea SDI del esclavo el dato (1 byte) que desea escribir
  - lectura: el esclavo envía el byte del registro direccionado por SDO

Diseño de un *master* SPI para el acelerómetro LIS2DH12 (III)

- Del estudio del mapa de registros del sensor y de la función de los registros, se deduce la conveniencia de que:
  - las **escrituras** del maestro – para configurar el sensor - se harán con transferencias de un byte (1+1, que no requieren autoincremento del puntero)
  - las **lecturas** –para recoger las medidas de aceleración en un eje y consistirán en la transferencia de 2 bytes (1+2), de registros mapeados en direcciones consecutivas- sí **harán uso del incremento automático del puntero**
  - se decide que el master SPI sea capaz de realizar lecturas **en los 3 ejes** para facilitar posibles ampliaciones futuras (transferencias de hasta **1+6** bytes)

Especificación de un *master* SPI para el acelerómetro LIS2DH12 (I)

- Interfaz

Pin	Dirección	Descripción
nCS	Out	Chip select del bus SPI. Activa a nivel bajo
SPC	Out	Reloj del bus SPI
SDI_MASTER	In	Entrada de datos serie
SDO_MASTER	Out	Salida de datos serie

Especificación de un *master* SPI para el acelerómetro LIS2DH12 (II)

- Interfaz

Pin	Dirección	Descripción
RDY	Out	Modulo listo para realizar una nueva transferencia. Activa a nivel alto
START	In	Comienzo de transferencia. Debe ser un pulso a nivel alto de la duración de un ciclo del reloj del sistema. Solo funciona si RDY activo
nWR_RD†	In	Indica si la transferencia es de lectura (1) o escritura (0)
DIR_REG[6:0] †	In	Bit 6: autoincremento sí (1) o no (0). Bits 5..0: dirección del registro
DATO_WR[7:0] †	In	Valor a escribir en el registro direccionado, si nWR_RD = 0
NO_BYTES[2:0] ‡	In	#bytes de la transferencia (2 en escritura y entre 3 y 7 en lectura)

†Esta entrada debe tomar su valor durante el pulso de start
‡Esta entrada debe tomar su valor durante el puldo de start y permanecer hasta la finalización de la transferencia

Especificación de un *master* SPI para el acelerómetro LIS2DH12 (III)

- Interfaz

Pin	Dirección	Descripción
DATO_RD[7:0]	Out	Datos leídos del esclavo
ENA_RD	Out	Validación de los datos leídos del esclavo. Pulso a nivel alto con duración de un período del reloj del sistema

Adicionalmente el circuito tendrá una entrada de reset asíncrono, **nRst**, activa a nivel bajo y una entrada de reloj, **clk**

Especificación de un *master* SPI para el acelerómetro LIS2DH12 (IV)

- Especificaciones funcionales

#	Explicación
ESP1	La interfaz debe ser la descrita en las dispositivas anteriores
ESP2	Debe implementar un master SPI a 4 hilos con polaridad 1 y fase 1 compatible con el esclavo en el acelerómetro LIS2DH12
ESP3	Debe implementar transferencias de escritura de 1 byte
ESP4	Las transferencias de lectura serán de 2, 4 o 6 bytes consecutivos
ESP5	RDY debe permanecer activo (1) mientras no haya transferencias en curso y desactivarse (0) en el siguiente ciclo de reloj después de la activación de START
ESP6	START debe activarse (1) durante un ciclo de reloj para iniciar una transferencia. La transferencia solo debe iniciarse si RDY está activo
ESP6	En las transferencias de **escritura**, simultáneamente a la activación de START, nWR_RD =0 y DIR_REG(6) = 0, DIR_REG(5..0) y DATO_WR deben contener la dirección del registro del esclavo y el dato que se va a escribir, respectivamente, y NO_BYTES = 2

Especificación de un *master* SPI para el acelerómetro LIS2DH12 (V)

- Especificaciones funcionales

#	Explicación
ESP7	En las transferencias de **lectura**, simultáneamente a la activación de START, nWR_RD =1 y DIR_REG(6) = 1, DIR_REG(5..0) debe contener la dirección del registro del esclavo del que se va a leer y NO_BYTES = 3 (lectura del eje X), 5 (X e Y) o 7 (X, Y y Z)
ESP8	NO_BYTES debe permanecer estable durante toda la transferencia
ESP9	Durante las transacciones de **lectura** ENA_RD debe activarse 2 (si NO_BYTES = 3), 4 (NO_BYTES = 5) o 6 (NO_BYTES = 7) veces, para habilitar la lectura de los bytes que se van leyendo del esclavo

- Especificaciones no funcionales

#	Explicación
ESP10	El diseño debe realizarse conforme a las metodologías de diseño síncrono
ESP11	El reloj del sistema debe ser de 50 MHz
ESP12	El sistema de prototipará en una tarjeta DECA MAX10 de Intel-Altera

Especificación de un *master* SPI para el acelerómetro LIS2DH12 (VI)

- Especificación de tiempos
  - La compatibilidad con el acelerómetro LISDH12 (ESP2) implica que deben cumplirse sus especificaciones de tiempos:

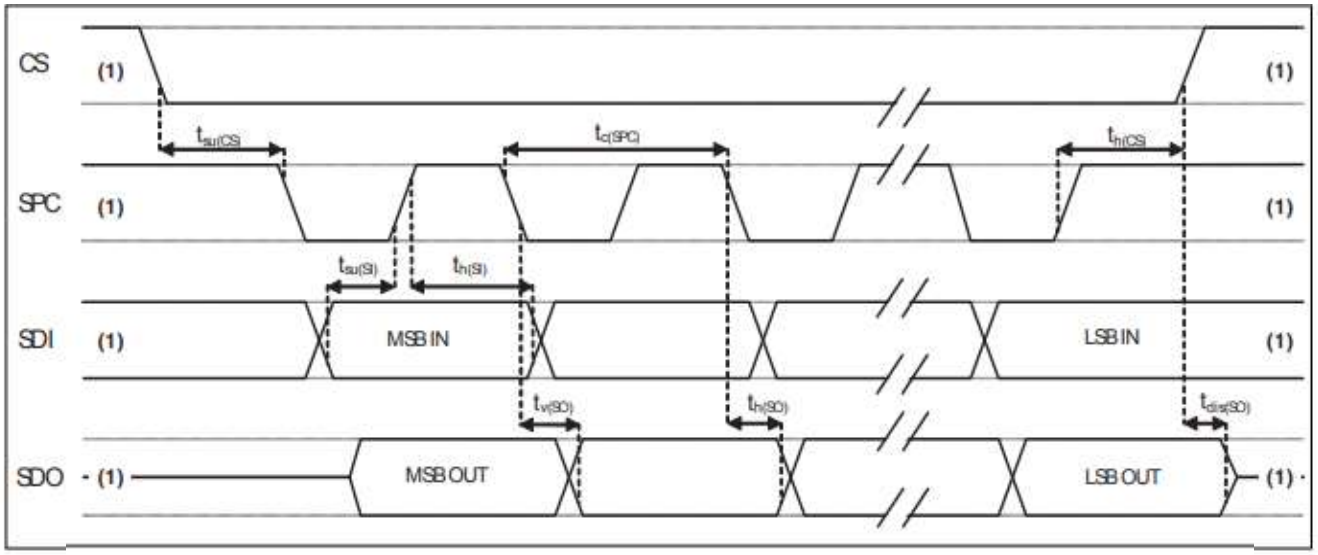


Table 6. SPI slave timing values

Symbol	Parameter	Value (1)		Unit
		Min	Max	
$t_{c(SPC)}$	SPI clock cycle	100		ns
$f_{c(SPC)}$	SPI clock frequency		10	MHz
$t_{su(CS)}$	CS setup time	5		ns
$t_{h(CS)}$	CS hold time	20		
$t_{su(SI)}$	SDI input setup time	5		
$t_{h(SI)}$	SDI input hold time	15		
$t_{v(SO)}$	SDO valid output time		50	
$t_{h(SO)}$	SDO output hold time	5		
$t_{dis(SO)}$	SDO output disable time		50	

Especificación de un *master* SPI para el acelerómetro LIS2DH12 (VII)

- Teniendo en cuenta que la frecuencia del reloj del sistema es de 50 MHz (20 ns):

  1.- Se elige una frecuencia de 5 MHz para **SPC** (< 10 MHz, periodo equivalente a 10 ciclos de reloj del sistema, CT 50%)

  2.- Set-up y hold de **nCS**: suficiente si **SPC** a nivel alto un ciclo de reloj después de su activación y uno antes de su desactivación

  3.- Set-up y hold de **SDI**: suficiente si **SDO_master** cambia un ciclo de reloj después de la bajada de **SPC**

  4.- La temporización de **SDO** garantiza que el master puede leer bits del esclavo un ciclo de reloj después de la subida de **SPC**

Table 6. SPI slave timing values

Symbol	Parameter	Value (1)		Unit
		Min	Max	
$t_{c(SPC)}$	SPI clock cycle	100		ns
$f_{c(SPC)}$	SPI clock frequency		10	MHz
$t_{su(CS)}$	CS setup time	5		ns
$t_{h(CS)}$	CS hold time	20		
$t_{su(SI)}$	SDI input setup time	5		
$t_{h(SI)}$	SDI input hold time	15		
$t_{v(SO)}$	SDO valid output time		50	
$t_{h(SO)}$	SDO output hold time	5		
$t_{dis(SO)}$	SDO output disable time		50	

Estructura del ejercicio

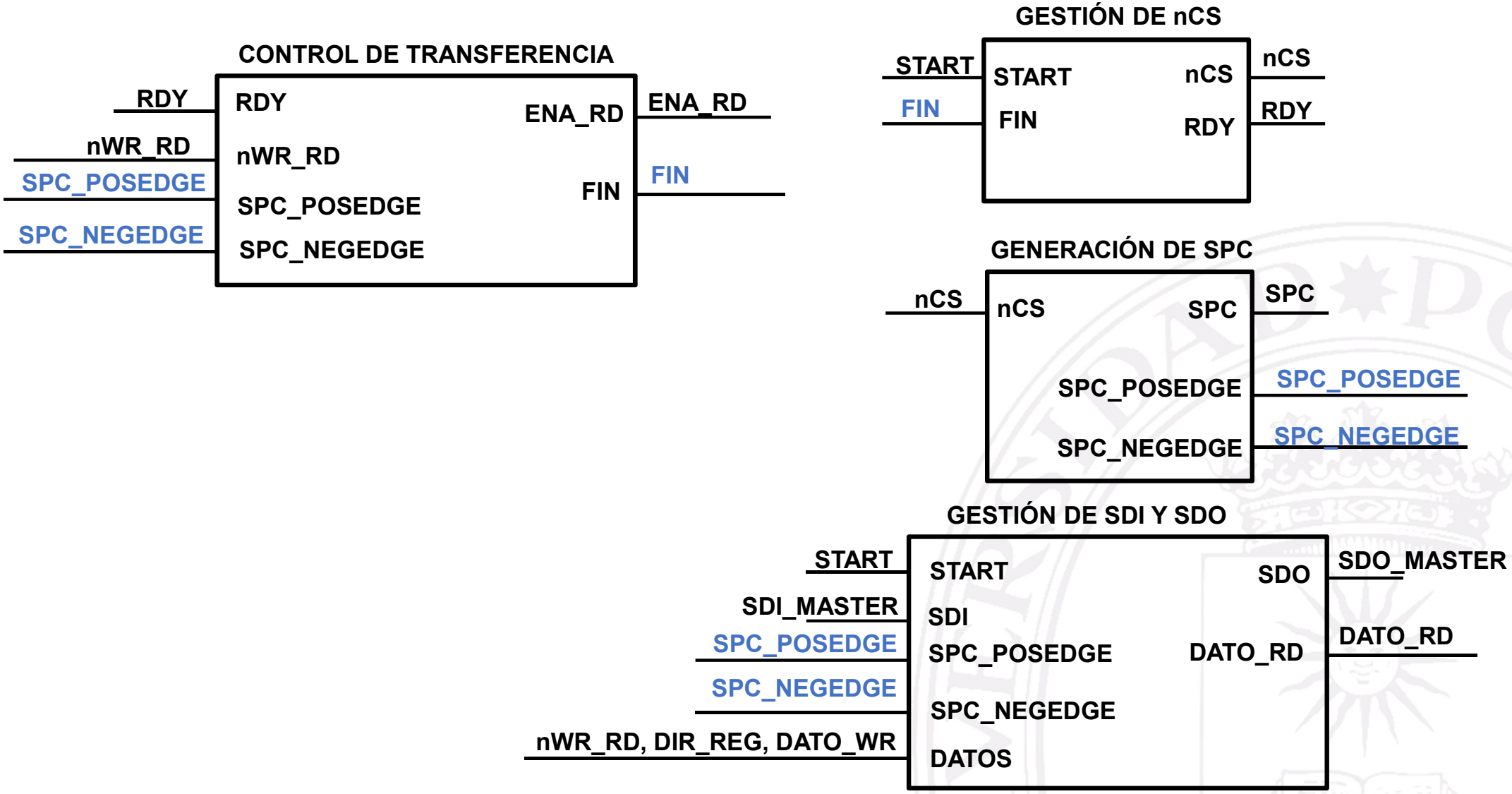
El ejercicio de diseño discurrirá de la siguiente manera:

1.- Se presentará primero un diagrama de bloques que representa los subsistemas que componen la estructura del *master*. Advierta que se trata de un diagrama conceptual que no recoge minuciosa ni detalladamente los detalles de interconexión entre subsistemas que observará en el modelo.

2.- Se le proporcionará un diagrama incompleto de un modelo VHDL 2008 del *master*. Sobre dicho modelo se plantearán pequeños ejercicios de modelado y análisis

3.- Se le proporcionará código para la realización de una simulación que permita verificar el funcionamiento del modelo completo

Estructura del circuito



Ejercicio. Apartado 1

Complete el código del modelo del subsistema que gestiona la generación de las señales **nCS** y, identificado con el comentario *Gestión de nCS*

Recuerde que: Una transferencia se inicia con la activación de **start** cuando **rdy** vale 1; **rdy** vale 0 durante el transcurso de la transferencia.

Emplee las simplificaciones sintácticas de VHDL 2008 donde sea posible

```
-- Apartado A
-- Gestion de nCS:
process(nRst, clk)
begin
  if  nRst = '0' then                          -- Cambiar 2008
    nCS <= ;

  elsif clk'event and clk = '1' then
    if  then                                   -- Completar 2008
      nCS <= ;

    elsif fin = '1' then                       -- Fin se activa cuando debe desactivarse nCS
      nCS <= ;                                 -- Cambiar 2008

    end if;
  end if;
end process;

rdy <= ;                                       -- Ecuacion que define el nivel logico de rdy


-- Fin de Gestion nCS
```

Ejercicio. Apartado 2

El código identificado como *Generación de SPC* modela la generación de la señal de reloj y de la detección síncrona de los flancos de subida y bajada de dicha señal.

1.- Observe el uso de las simplificaciones sintácticas de VHDL-2008 que se emplean.

2.- Debe saber que más adelante este código se sintetizará con una herramienta que no dispone del paquete *numeric_std_unsigned*, que resulta indispensable para emplear la operación "*?=".* Cambie el código de las sentencias concurrentes por otro equivalente que no emplee estas operaciones –y sustituya el paquete *numeric_std_unsigned* por el *std_logic_unsigned*

3.- Compruebe que el diseño de la generación de la señal **SPC** efectivamente cumple los tiempos de *set-up* y *hold* respecto a la señal **nCS** especificados en la hoja de datos del acelerómetro. Deduzca el valor efectivo de dichos tiempos en el diseño realizado.

4.- Observe que tras el reset asíncrono o cuando se desactiva **nCS**, el contador con el que se genera **SPC** se inicializa con el valor de cuenta 2. Intente deducir el porqué, pero si tiene dificultades no dedique más de 5 minutos a analizar la causa y consulte con el profesor.

Ayuda: Observe las ecuaciones que detectan los flancos de subida y bajada.

5.- Determine los cambios que habría que realizar en el modelo para que la frecuencia de **SPC** fuera de 1 MHz. Pero no cambie el modelo.

Ejercicio. Apartado 3

El código identificado como *Gestión de SDI y SDO* modela la escritura de bits en **SDO**, la lectura de bits de **SDI** y la generación de los bytes leidos (**dato_rd**). Observe que hay dos sentencias concurrentes incompletas (las que asignan valor a **dato_rd** y a **SDO**).

1.- Determine en qué flanco se escriben y leen los bits y compruebe que se corresponde con la polaridad indicada en la especificación del sistema.

2.- Observe la señal **SDI_syn** para determinar qué modela y por qué resulta necesaria.

3.- Compruebe que los bits que se escriben cumplen los tiempos de *set-up* y *hold* establecidos para la línea **SDO** (SDI del esclavo).

4.- Compruebe que cuando se lee un bit el valor en la señal **SDI_syn** está garantizado por las especificaciones de tiempo del acelerómetro. Si tiene dudas consulte con el profesor.

5.- Complete las sentencias concurrentes

Ejercicio. Apartado 4

El código identificado como *Gestión de transferencia* genera la señal de salida que valida los bytes leidos (**ena_rd**) y la generación de una señal de control que determina el final de la transferencia (**fin**).

1.- Determine qué bits de la señal **cnt_bits_SPC** determinan el número completo de bytes transmitidos y cuáles el número de bits transmitidos del byte en curso.

2.- Considerando la respuesta que ha dado a la cuestión anterior, analice la sentencia concurrente que asigna valor a **ena_rd** para determinar cuando y por qué se activa esta señal.

3.- Traduzca la sentencia concurrente que asigna valor a VHDL pre-2008.

4.- Compile el código del modelo para depurar posibles errores sintácticos.

Ejercicio. Apartado 5

Antes de realizar este apartado el profesor le explicará el código de los ficheros que contienen un modelo del acelerómetro para *test* y un *test-bench* que le permitirá verificar el funcionamiento del *master* SPI. Una vez concluida la exposición realice el ejercicio

1.- Dedique unos minutos a repasar el código que se le acaba de explicar.

2.- Ejecute después una simulación para verificar el funcionamiento del modelo del *master* SPI. Utilice el fichero **.do** que se suministra para añadir y configurar la ventana **Wave** con las señales del diseño.

- Más información: https://blogs.upm.es/ue-upm/

- Contacto: comunidad.microelectronica@upm.es