Fulla d'especificacions comunicació SPI "SPI_Master_MLF"

Autor: Miquel Lorenzo Farràs

NiUB: 16655866

Assignatura: Disseny i Síntesi de Sistemes Digitals

Data d'entrega: 15 Maig, 2020

Índex

1. Introducció

- En aquest document podrem trobar tota la informació tècnica disponible per poder entendre correctament el funcionament del SPI_Master_MLF, el qual consisteix en el Màster d'una comunicació SPI (Serial Peripherial Interface) aquesta comunicació s'ha creat a partir del Màster que podem trobar en la pàgina de Nandland (https://github.com/nandland/spi-master). La connexió SPI tracta d'una comunicació en sèrie síncrona que requereix un mínim de 3 cables: CLK, MOSI (Master Out Slave In), MISO (Master In Slave Out). Aquest Màster pot funcionar de forma autònoma, de totes maneres, recomanem l'aplicació d'un codi a més alt nivell el qual funcioni com a TOP per poder comunicar-nos amb qualsevol Slave, en aquest TOP serà on afegirem la quarta senyal característica de la comunicació SPI, Chip Select, de totes formes, en aquest datasheet només es tractarà del funcionament interior del SPI_Master_MLF.

Aplicacions:

- Compatible amb sensors que utilitzin comunicació SPI
- Parametritzable i sintetitzable
- Comptabilitat amb els 4 modes SPI
- Disseny síncron
- Configuració MSB
- Capacitat per ajustar la velocitat del rellotge SPI
- Conversió en paral·lel i en sèrie per la millor interpretació de les dades

2. SPI_Master_MLF Ports IO

2.1. Connexions d'interfície

Port	Mida (bit)	Direcció	Descripció
Senyals de control			
i_rst_n	1	Input	Reset asíncron actiu a nivell baix
i_clk	1	Input	Rellotge del sistema
Senyals de Tx (MOSI)			
i_TX_Byte	8	Input	Byte que volem enviar des del Master SPI
i_TX_DV	1	Input	Bit que ens indica quan les dades de Tx són vàlides
o_TX_Ready	1	Output	Bit que indica al codi que està llest per rebre dades
Senyals de Rx (MISO)			
o_RX_DV	1	Output	Bit que ens indica quan les dades de Rx son valides
o_RX_Byte	8	Output	Byte que rebem al Master SPI

2.1.1. i_rst_n

- Senyal de reset asíncrona i activa per nivell baix, la funció d'aquesta senyal és tornar el sistema a la seva posició inicial. Tots els registres del sistema tornen al seu valor inicial.

2.1.2. i_clk

- Tota la lògica interna del SPI_Master_MLF es registra amb el flanc de pujada del rellotge. *Anotació:* No confondre amb el rellotge del sistema SPI, parlarem d'ell a continuació

2.1.3. i_TX_Byte

- Aquest és el byte que volem enviar per MOSI, li donarem un valor inicial de 0 i un cop la senyal i_TX_DV s'activi carregarem el valor en paral·lel, aquest valor és el que podrem veure transformat en sèrie pel canal MOSI

2.1.4. i_TX_DV

- Pols d'un bit d'amplada actiu per nivell alt, La funció d'aquesta senyal és avisar al Master que les dades que volem enviar per i_TX_Byte estan preparades per enviar per la interfície SPI.

2.1.5. o_TX_Ready

- Senyal de sortida activa per nivell alt que avisa al Master que està preparat per rebre una nova dada, un cop aquesta dada és carrega amb èxit, posem la senyal a nivell baix per seguretat.

2.1.6. o_RX_DV

- Pols d'un bit d'amplada que ens indica que la dada o_RX_Byte ha acabat de rebre noves dades, s'activa quan el comptador r_RX_Bit_Count arriba a 0 indicant que tota la trama s'ha enviat

2.1.7. o_RX_Byte

- Aquesta senyal de 8 bits es construeix a partir de les dades que arriben en sèrie per MISO mitjançant el comptador r_RX_Bit_Count, el qual funciona com a indexador per omplir el byte en paral·lel de forma correcta

2.2. Connexions SPI externes

Port	Mida (bit)	Direcció	Descripció
o_SPI_clk	1	Output	Rellotge del nostre sistema SPI.
i_SPI_MISO	1	Input	Master In Slave Out
o_SPI_MOSI	1	output	Master Out Slave In

2.2.1. o_SPI_clk

- El rellotge del nostre sistema SPI, utilitzat per sincronitzar el tràfic de dades de MISO i MOSI, és genera a partir del rellotge del sistema i_clk i un paràmetre anomenat CLKS_PER_HALF_BIT on podem determinar quants cicles de i_clk volem que hi hagin cada mig bit de o_SPI_clk, la configuració d'aquest paràmetre el podem configurar mitjançant software.

2.2.2. i_SPI_MISO

- La senyal Master In Slave Out és unidireccional i arriba en sèrie al Master vinguda des del Slave corresponent que farà de sortida

2.2.3. o_SPI_MOSI

- La senyal Master Out Slave In és unidireccional i arriba en sèrie al Slave vinguda des del nostre Master, aquesta senyal s'envia a partir de la senyal en paral·lel o_TX_Ready, la qual transformem amb sèrie mitjançant lògica interna

2.2.4. Absència de o_SPI_CS_n

- Com podem veure, **no** comptem amb una senyal CS en el nostre màster SPI això és degut a que la funció d'aquests mòdul és proporcionar una correcta generació i rebuda de les senyals tal i com estipula el protocol SPI, si volem implementar la funció CS hauríem de generar un altre document, on, mitjançant una màquina d'estats on controlaríem la funció CS [(1)]

(1): Adjunt a aquest document, hauria de trobar-se un altre document "...\Lorenzo_SPI\doc\SPI_Master_MaquinaEstats_MLF.pdf" on expliquem la implementació d'una màquina d'estats, de tal manera com la implementació de la senyal o_SPI_CS_n.

3. SPI_Master_MLF paràmetres locals

3.1. Llista de registres locals

Nom	Mida (BIT)	Accés	Descripció
r_SPI_clk_count	5	R/W	Conta per calcular el SPI_clk
r_SPI_clk	1	R/W	Registre on guardem el valor de SPI_clk
r_SPI_Clk_Edges	5	R/W	Numero de flancs en un byte, sempre **16**
r_Leading_Edge	1	R/W	Registre on guardem el flanc de pujada
r_Trailing_Edge	1	R/W	Registre on guardem el flanc de baixada
r_TX_DV	1	R/W	Registre on guardem el valor de i_TX_DV
r_TX_Byte	8	R/W	Registre on guardem el valor de i_TX_Byte
r_RX_Bit_Count	3	R/W	Registre que ens indica a quin bit estem de RX
r_TX_Bit_Count	3	R/W	Registre que ens indica a quin bit estem de TX

3.1.1. r_SPI_clk_count

- Utilitzem aquest registre com a comptador per poder generar el clock del nostre SPI, comença amb valor inicial 0 el qual augmenta constantment i compara amb el paràmetre CLKS_PER_HALF_BIT, aquesta comparació ens permet escalar el i_clk al valor del o_SPI_clk.

3.1.2. r_SPI_clk

- Utilitzarem aquest registre per poder configurar de forma correcta el o_SPI_clk, per fer-ho, primer li assignarem el valor de w_CPOL per tenir la configuració correcta, a continuació, mitjançant r_SPI_clk_count i el paràmetre CLKS_PER_HALF_BIT podem generar la freqüència que desitgem al nostre clock i enviar-la a o_SPI_clk

3.1.3. r_SPI_Clk_Edges

- Utilitzem aquest registre com a comptador de flancs de byte, sabem que sempre és 16, però utilitzant aquest registre, podem activar correctament r_Trailing_Edge, r_Leading_Edge i podem generar el clock del nostre SPI de forma correcta, un cop el comptador arriba a 0, sabem que em carregat correctament un byte i activem r_TX_Ready.

3.1.4. r_Leading_Edge

- Utilitzem aquest registre per poder identificar quan ens trobem en un flanc de pujada, d'aquesta forma ens podem assegurar que la fase del nostre protocol SPI funcioni correctament

3.1.5. r_Trailing_Edge

- Utilitzem aquest registre per poder identificar quan ens trobem en un flanc de baixada, d'aquesta forma ens podem assegurar que la fase del nostre protocol SPI funcioni correctament

3.1.6. r_TX_DV

- Utilitzem aquest registre per carregar el valor que ens arriba per i_TX_DV, un cop carregat, té la mateixa funció, quan s'activa i ens trobem en fase el valor de r_TX_Byte s'envia per o_SPI_MOSI

3.1.7. r_TX_Byte

- Utilitzarem aquest registre per enviar les dades per o_SPI_MOSI, per fer-ho, primer carregarem el valor en paral·lel que ens arriba per i_TX_Byte, a continuació, mitjançant r_TX_Bit_Count enviem el byte en sèrie per o_SPI_MOSI

3.1.8. r_RX_Bit_Count

- Utilitzarem aquest registre com a indexat de o_RX_Byte, per fer-ho li donarem un valor inicial de 3'b111 i l'anirem disminuint, un cop aquest arribi a 3'b000 indicarà que el byte s'ha acabat de rebre, mitjançant aquest registre també podem convertir la senyal en sèrie MISO en un byte en paral·lel mitjançant una assignació MSB.

3.1.9. r_TX_Bit_Count

- Utilitzarem aquest registre com a indexat de o_TX_Byte, per fer-ho li donarem un valor inicial de 3'b111 i l'anirem disminuint, un cop aquest arribi a 3'b000 indicarà que el byte s'ha acabat de transmetre, mitjançant aquest registre també convertim el byte en paral·lel en sèrie pel canal MOSI.

3.2. Llista de wires locals

Nom	Mida (bit)	Actiu SPI mode	Descripció
w_CPOL	1	2, 3	Polaritat del clk
w_CPHA	1	1, 3	Fase del clk

3.2.1. w_CPOL

- Amb aquest bit podrem configurar el mode de transferència, per poder fer-ho correctament, assignarem el seu valor a o_SPI_clk al estat inicial.

3.2.2. w_CPHA

- Amb aquest bit podrem configurar el mode de transferència, per fer-ho correctament, utilitzarem lògica AND amb els registres r_Leading_Edge i r_Trailing_Edge, les quals ens indiquen els flancs de pujada i baixada del nostre o_SPI_clk.

3.3. Paràmetres locals amb capacitat de configuració

3.3.1. SPI_MODE

- Mitjançant el paràmetre SPI_MODE, podem indicar al nostre codi en quin mode SPI volem treballar, a continuació presento la taula amb tots els models disponibles. Més informació sobre els diferents modes SPI a l'apartat 4.1

SPI_MODE	w_CPOL	w_CPHA
0	0	0
1	0	1
2	1	0
3	1	1

3.3.2. CLKS_PER_HALF_BIT

- Mitjançant el paràmetre CLKS_PER_HALF_BIT podem configurar la freqüència del nostre o_SPI_clk, donada la freqüència input de i_clk, ens indica quants polsos hi haurà per cada mig pols del nostre o_SPI_clk.
- **Exemple:** Si tenim un i_clk de 100MHz i volem una freqüència a o_SPI_clk de 25MHz, tindrem 2 polsos de i_clk per cada mig bit.

$$\frac{100MHz}{25Mhz} = 4\frac{pulsos}{1bit};\quad \frac{4pulsos}{bit} * \frac{1}{2} = 2\,pulsos = \boldsymbol{CLKS\_PER\_HALF\_BIT}$$

4. Operacions SPI_Master_MLF

4.1. Models de transferències SPI

- El protocol SPI consta de 4 models diferents de transferència, tal i com hem comentat en l'apartat 3.3.1, aquests quatre models utilitzen el o_SPI_clk per poder sincronitzar les dades MISO i MOSI en fase (w_CPHA) i en polaritat (w_CPOL), a continuació presentem tots els models disponibles indicant un punt crític que ens permet identificar el mode ràpidament.
- **w_CPOL:**

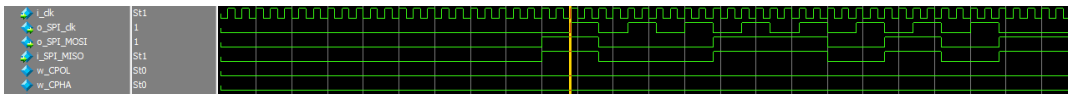
  '0' = o_SPI_clk es troba en posició baixa en repòs

  '1' = o_SPI_clk es troba en posició alta en repòs

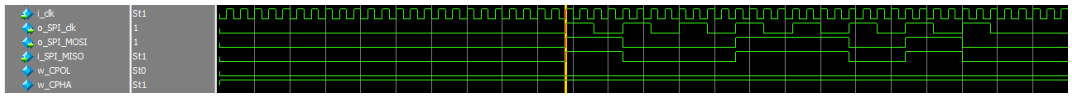
- **w_CPHA:**

  '0' = Mostregem la dada al flanc de pujada i la canviem al flanc de baixada

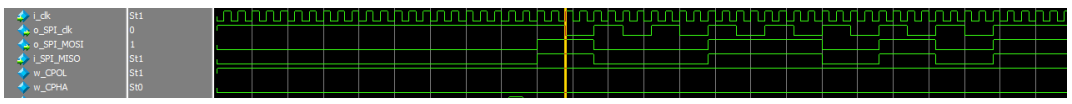
  '1' = Mostregem la dada al flanc de baixada i la canviem al flanc de pujada



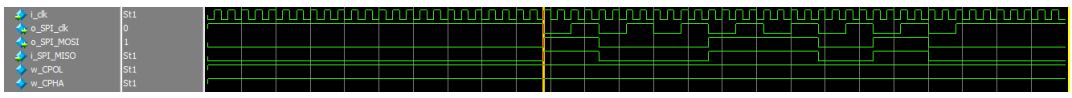
Il·lustració 1: Comportament de les senyals quan o_SPI_clk, o_SPI_MOSI i i_SPI_MISO per SPI_MODE = 0



Il·lustració 2: Comportament de les senyals quan o_SPI_clk, o_SPI_MOSI i i_SPI_MISO per SPI_MODE = 1



Il·lustració 3: Comportament de les senyals quan o_SPI_clk, o_SPI_MOSI i i_SPI_MISO per SPI_MODE = 2



Il·lustració 4: Comportament de les senyals quan o_SPI_clk, o_SPI_MOSI i i_SPI_MISO per SPI_MODE = 3

4.2. Inicialitzant transferències

4.2.1. Enviar bytes de dades

- Després de programar correctament els registres mitjançant un pols de reset inicial, podem començar a enviar bytes en sèrie per la senyal MOSI mitjançant el SPI_Master_MLF. Per fer-ho, primer carreguem el registre r_TX_Byte amb el valor en paral·lel del pin i_TX_Byte, mirarem el valor de o_TX_Ready per veure si el Màster està preparat per enviar noves dades, si o_TX_Ready = 1, el registre r_TX_DV generarà un pols d'un bit indicant que la transferència pot començar, mitjançant el comptador r_TX_Bit_Count, les dades de r_TX_Byte s'enviaran pel pin o_SPI_MOSI en sèrie mitjançant una configuració MSB.

4.2.2. Rebre bits de dades

- Després de programar correctament els registres mitjançant un pols de reset inicial, podem començar a rebre dades en sèrie per la senyal MISO mitjançant el SPI_Master_MLF, aquestes dades seran convertides en paral·lel per qualsevol futura utilització. Per fer-ho, primer mirarem la senyal o_TX_Ready, per veure si hi ha alguna transferència activa en aquell moment, si o_TX_Ready = 1, el registre comptador r_RX_Bit_Count es configurarà amb el valor màxim i començarà a indexar les dades que arribin pel pin i_SPI_MISO en paral·lel (configuració MSB) al pin o_RX_Byte, un cop el valor del comptador arribi a 0, o_RX_Byte generarà un pols d'un bit indicant que s'ha acabat de rebre el byte.

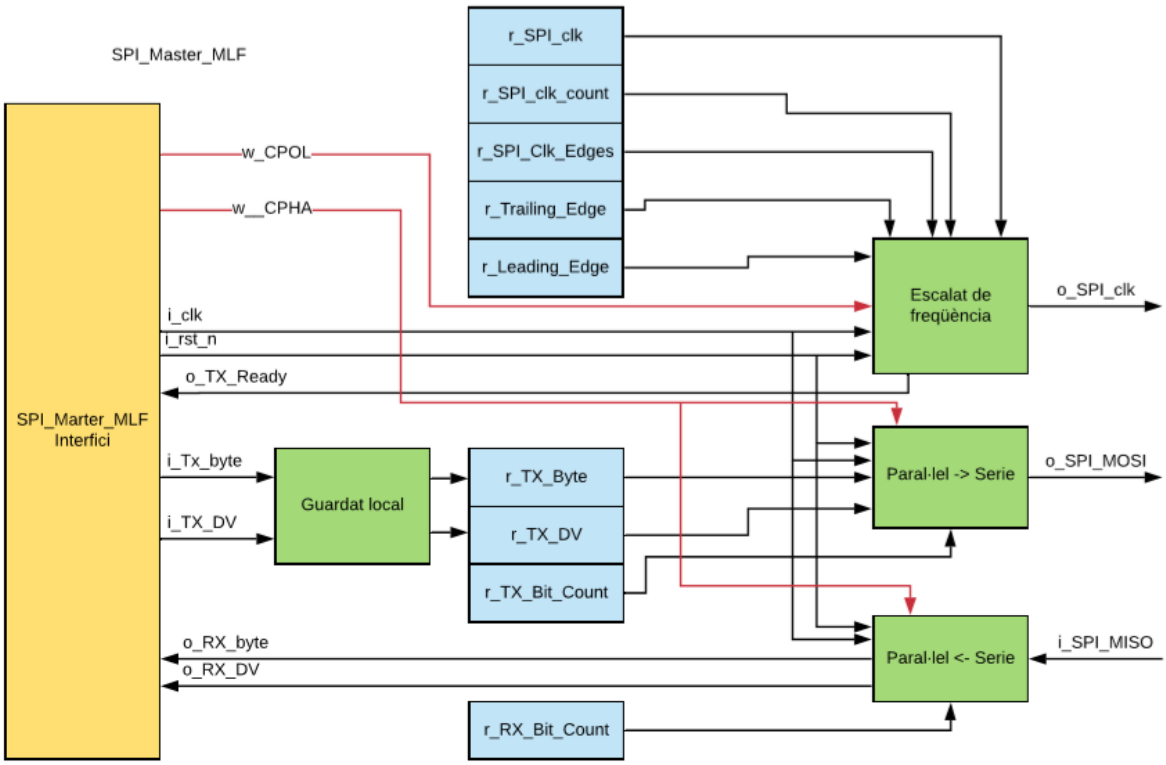
4.2.3. Echo

- La utilització del nostre SPI_Master_MLF en sensors s'ha de configurar mitjançant un TOP amb una maquina d'estats controlada amb Chip Select[(1)] el qual ens permetrà controlar qualsevol sensor que utilitzi aquest protocol de comuniciació SPI, previament a configurar el sensor, podem configurar un test echo[(2)] per veure que les senyals s'envien correctament al sensor, mitjançant un echo també podrem assegurar-nos que serem capaços de rebre correctament les dàdes que el sensor envii, per fer-ho el que haurém de realitzar és curtcircuitar els pins i_SPI_MISO i o_SPI_MOSI mitjançant un wire.

[(2):] Adjunt a aquest document, hauria de trobar-se un altre document "...\Lorenzo_SPI\tb\SPI_Master_MLF_TB.v" on donem un testbench d'exemple amb la funció echo implementada.

5. Arquitectura

5.1. Dibuix jeràrquic



5.2. RTL viewer

- Degut a la mida de la imatge generada pel Quartus, no podem mostrar la imatge del RTL viewer en aquest datasheet, però podreu trobar imatge completa adjunta a aquest fitxer: *"...\Lorenzo_SPI\doc\SPI_Master_MLF_RTL_Viewer.jpg"*

5.3. Pin Planer

- Podem trobar el fitxer que ha exportat el Quartus a *"...\Lorenzo_SPI\doc\SPI_Master_MaquinaEstats_MLF.csv"*, presentem a continuació el pin planer per la FPGA CYCLONE IV EP4CE115F29C7

Senyal	Input/Output	Pin
i_SPI_MISO	Input	K2
i_TX_Byte[7]	Input	J6
i_TX_Byte[6]	Input	L4
i_TX_Byte[5]	Input	K7
i_TX_Byte[4]	Input	L3
i_TX_Byte[3]	Input	J7
i_TX_Byte[2]	Input	M4
i_TX_Byte[1]	Input	J5
i_TX_Byte[0]	Input	K8
i_TX_DV	Input	K3
i_clk	Input	J1
i_rst_n	Input	Y2
o_RX_Byte[7]	Output	L7
o_RX_Byte[6]	Output	G1
o_RX_Byte[5]	Output	H5
o_RX_Byte[4]	Output	L5
o_RX_Byte[3]	Output	J3
o_RX_Byte[2]	Output	M3
o_RX_Byte[1]	Output	F2
o_RX_Byte[0]	Output	J4
o_RX_DV	Output	G2
o_SPI_MOSI	Output	L8
o_SPI_clk	Output	L6
o_TX_Ready	Output	M7