Laboratori de
Disseny i Síntesi de Sistemes Digitals

Grau Enginyeria Electrònica de Telecomunicació

Autor

Joan Canals Gil

Contribuïdors

Sergio Moreno Martín,
Oscar Alonso Casanovas,
Ángel Diéguez Barrientos

UNIVERSITAT DE BARCELONA

Índex

Introducció

L'objectiu d'aquestes pràctiques és aprendre a dissenyar, sintetitzar i verificar un sistema digital complet. El flux genèric de disseny d'un sistema digital es pot dividir en diverses etapes tal com mostra la Figura 1. Cal notar que la síntesi física en el cas de disseny en ASIC inclou més subprocessos (*power planing, IO planning, clock tree*, entre d'altres) i les respectives verificacions en comparació en la implementació en una FPGA. Això és degut a l'estructura interna les FPGAs on l'encaminament dels senyals, alimentació, propagació de rellotges, nombre i tipus de portes lògiques disponibles, disposició d'entrades i sortides estan preestablertes.

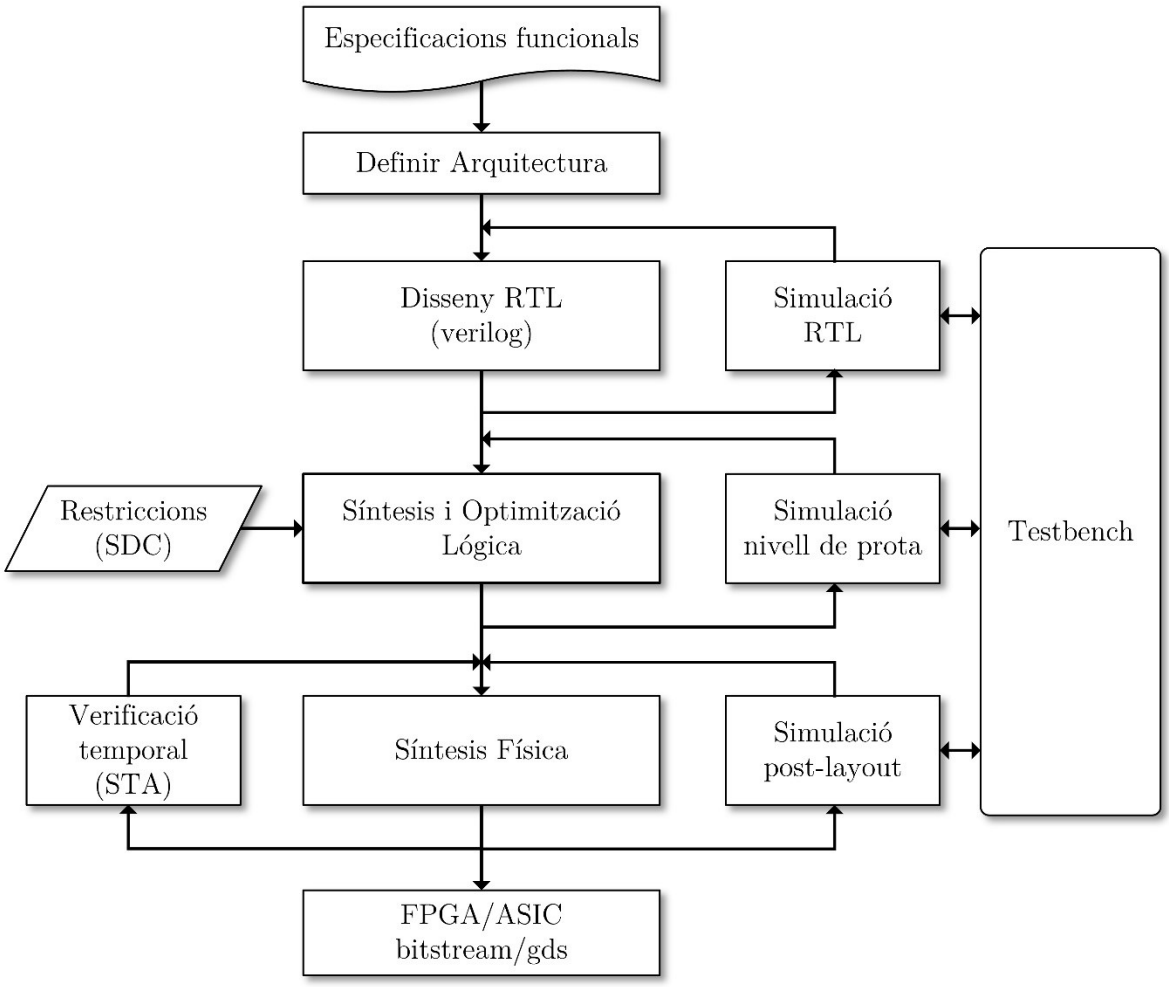

Figura 1. Flux de disseny i verificació genèric d'un sistema digital.

Durant el transcurs de les diferents sessions pràctiques, veureu cadascuna de les diferents etapes aplicades al disseny d'una interfície SPI (de l'anglès, Serial Peripheral Interface) completa, que s'utilitzarà com interfície de comunicació en una estació meteorològica per interrogar el sensor de temperatura, pressió i humitat (model BME280 de Bosch) integrat en una placa de desenvolupament Adafruit-2652.

Interfície SPI

El bus SPI va ser desenvolupat per Motorola per proporcionar una comunicació sèrie síncrona dúplex completa entre dispositius mestres i esclaus. El bus SPI s'utilitza per a la comunicació amb memòries flash, sensors, rellotges en temps real (RTCS), convertidors analògics a digitals, entre d'altres. Com es mostra a la Figura 2, els mestres SPI estàndard es comuniquen amb esclaus que utilitzen el rellotge sèrie (**SCK**), dues línies de dades mestre a l'esclau (**MOSI**, de l'anglès Master Output Slave Input), esclau a mestre (**MISO** de l'anglès, Master Input Slave Output) i les línies de selecció d'esclau (**SS** de l'anglès Slave Select). Els senyals SCK, MOSI i MISO són comuns a tots els esclaus del bus, mentre cada esclau té una línia de selecció SS única.

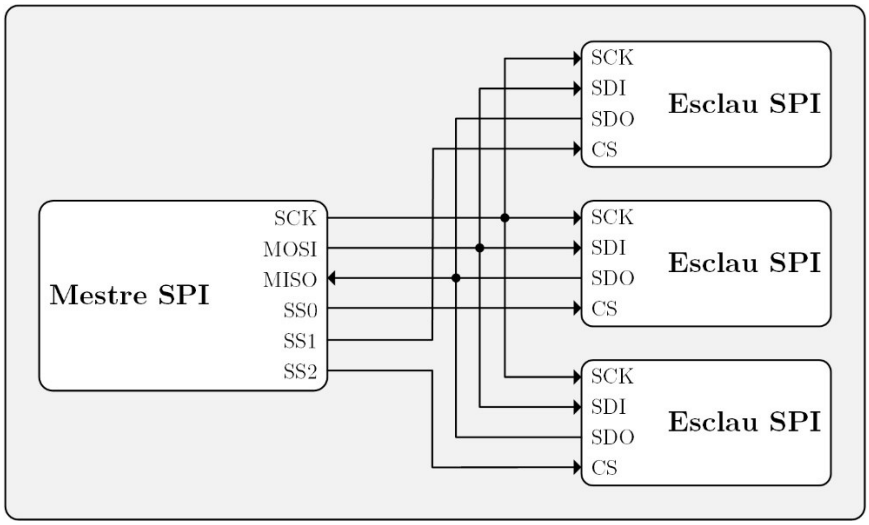

Figura 2. Exemple de bus SPI amb un mestre i controlant tres esclaus.

El bus SPI no defineix cap protocol per a l'intercanvi de dades, és a dir, no fixa ni el nombre de bytes a transmetre ni el contingut d'aquests, sinó que limita la sobrecàrrega del bus (el nombre d'esclaus que es poden utilitzar) per transmetre dades a alta velocitat (desenes de MHz). Existeixen **quatre modes de comunicació únics** en funció de la polaritat (CPOL) i la fase (CPHA) del rellotge SCK, tal com es mostra a la Figura 3.

- **CPOL = CPHA = "0" (mode 0):** les dades són capturades per l'esclau en el flanc de pujada de rellotge. El mode 0 és, amb diferència, el mode més comú per a la comunicació SPI.
- **CPOL = "0" i CPHA = "1" (mode 1):** les dades es mostregen en el flanc de baixada rellotge.
- **CPOL = "1" i CPHA = "0" (mode 2):** les dades es mostregen en el flanc de baixada de rellotge.
- **CPOL = CPHA = '1' (mode 3):** les dades són mostrejades en el flanc de pujada de rellotge.

El protocol SPI no defineix una estructura del flux de dades. La composició de les dades depèn totalment del dissenyador de l'esclau. No obstant això, molts dispositius segueixen el mateix format bàsic per a l'enviament i la recepció de dades organitzades per bytes, permetent la interoperabilitat entre parts de diferents proveïdors.

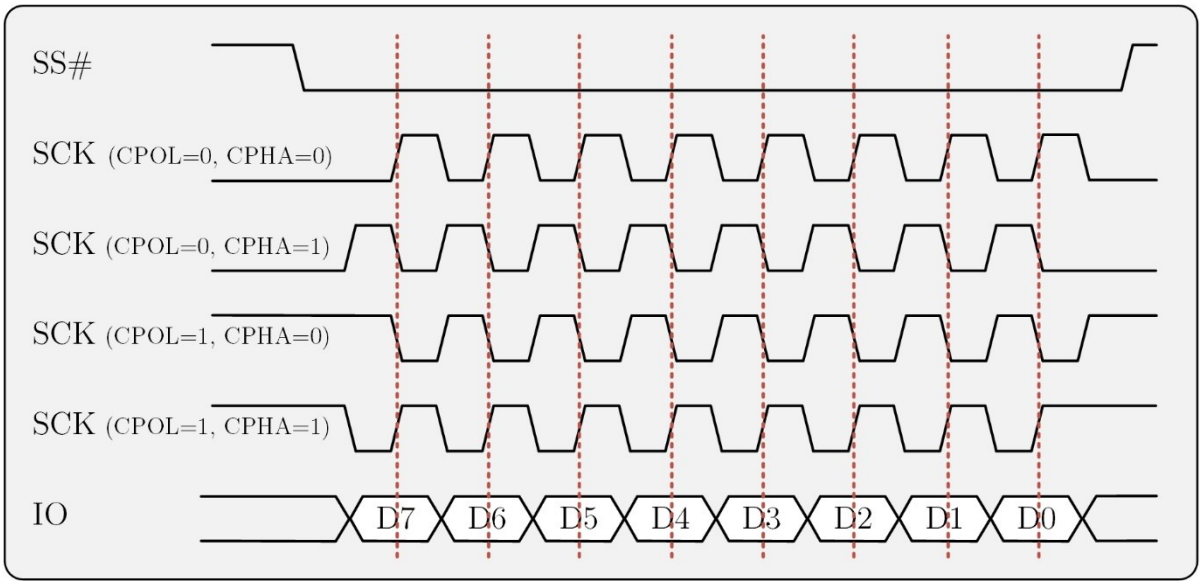

Figura 3. Modes de comunicació del bus SPI.

La Figura 4 mostra una transició d'escriptura d'un registre que consta d'un byte d'instrucció i un byte d'adreça i un bytes de dades. Per escriure al registre, el mestre SPI selecciona l'esclau posant a zero la línia de selecció. El Mestre produeix la instrucció adequada seguida del byte de direccionalment i el de dades a escriure al registre. Atès que la transacció no necessita retornar cap dada, el dispositiu esclau manté la línia MISO en un estat d'alta impedància i el mestre emmascara les dades entrants. Finalment, es deshabilita la senyal de selecció per completar la transacció.

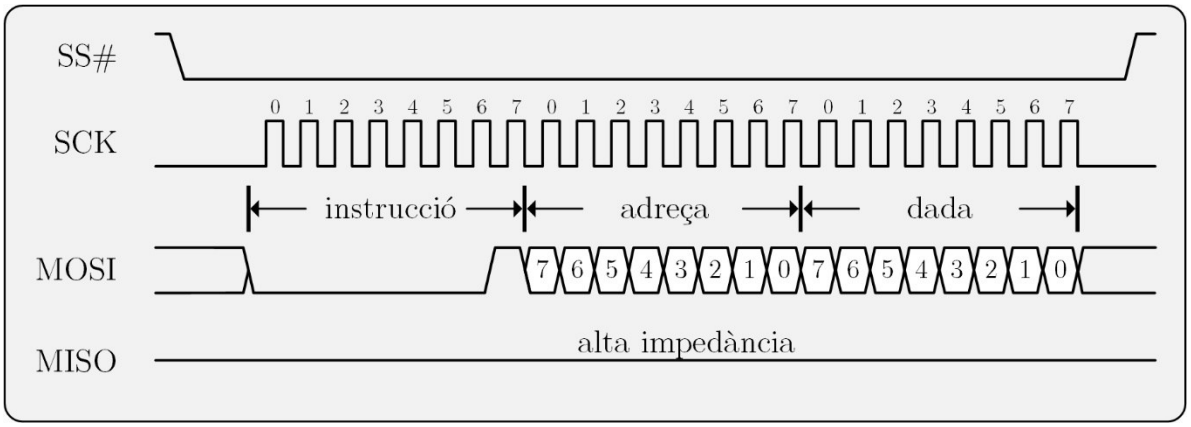

Figura 4. Exemple escriptura a un registre.

La transmissió de lectura és molt similar a la d'escriptura, un cop enviada la instrucció de lectura, i l'adreça el mestre ha de generar els cicles de rellotges necessaris per a que l'esclau pugui enviar les dades. La Figura 5 mostra un exemple de lectura d'un registre de 8 bits.

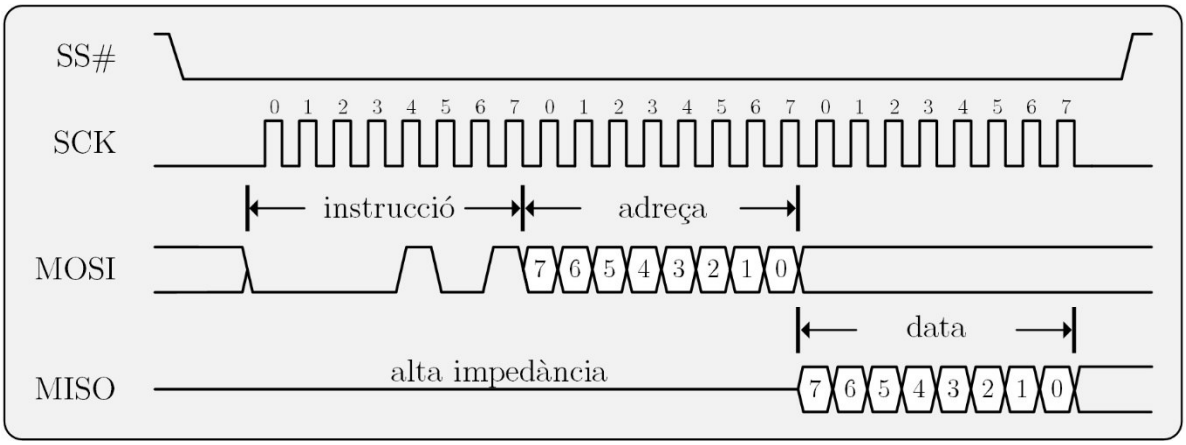

Figura 5. Exemple lectura d'un registre.

Llistat de Pràctiques

- Les pràctiques es realitzen en sessions de laboratori de 3 hores.
- Cada pràctica consta d'un nombre de sessions i entregues determinats.
- Es disposa d'una sessió de recuperació abans de l'última pràctica.
- Al final de les sessions de pràctiques hi ha un examen pràctic.

Pràctica	**Títol**	**#Sessions**	**#Entregues**	**Puntuació**
1	Arquitectura d'un sistema digital.	2	1	20 %
2	Introducció al flux de disseny i síntesis en FPGA.	1	1	10 %
3	Disseny RTL i Verificació.	4	3	30 %
-	*Recuperació*	*1*	-	
4	Disseny i síntesis d'una estació Meteorològica.	2	1	20 %
	Examen	*1*	-	20 %

Criteris d'avaluació

- Per aprovar una pràctica cal fer tots els apartats descrits en els guions i informes.
- Per aprovar les pràctiques s'han d'aprovar totes les pràctiques (mínim de 5 sobre 10).
- Per aprovar les pràctiques s'ha d'aprovar l'examen (mínim de 5 sobre 10).
- L'assistència al laboratori és obligatòria, la falta es considerarà un no presentat.
- Les pràctiques recuperades tindran una puntuació màxima de 5 sobre 10.
- Els criteris d'avaluació is generals:
  - → Codi RTL: estructura i sintetitzable.
  - → Testbench: estructura i codificació.
  - → Verificació completa de la funcionalitat del mòdul.
  - → Comentaris al codi tant del testbench com del RTL.
  - → Informe de pràctiques amb tots els apartats degudament complimentat amb explicacions clares i concises.
  - → Implementació i demostració en FPGA.
  - → Seguir els convenis d'estructura del directori de treball i codificació Verilog.

Material de Pràctiques

Per la realització de les pràctiques utilitzarem:

- Verilog HDL standard (IEEE 1364-2005) per descriure a nivell RTL els diferents mòduls i els test corresponents.
- ModelSim Intel® FPGA Starter Edition de Siemens que utilitzarem per simular i verificar els diferents mòduls dissenyats tant a nivell RTL com post-implementació.
- Intel Quartus Prime Lite Edition Design Software v20.1 que utilitzarem per sintetitzar el nostre disseny.
- Encara que tant el ModelSim com el Quartus incorporen editors de text, és recomana utilitzar un editor de codi extern. Algunes de les opcions més populars són: SublimeText, Neovim, VS Code o GNU Emacs.
- Placa de desenvolupament DE0-CV de terasIC, que consta d'una FPGA Intel 28-nm Cyclone V model 5CEBA4F23C7N, on implementarem alguns dels dissenys al llarg de les sessions pràctiques.

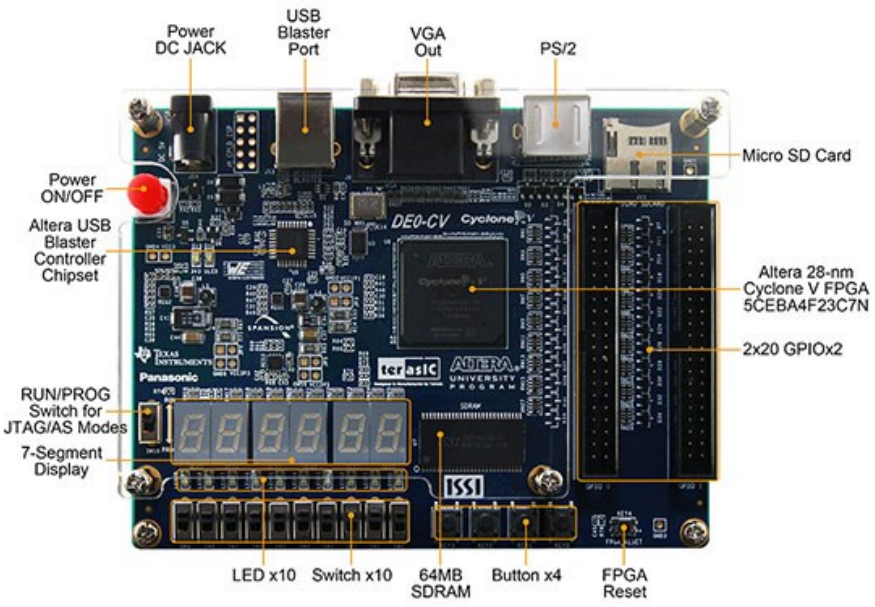

Figura 6 Vista superior de la placa de desenvolupament DE0-CV amb els diferents elements identificats.

Al campus virtual trobareu:

- Link de descarregar del ModelSim i el Quartus així com els fitxers de la tecnologia necessaris.
- Guia d'instal·lació i ús tant el Modelsim com del Quartus.
- Guia ràpida sobre el llenguatge Verilog.
- Lectures recomanades sobre el codificació, disseny i síntesis de sistemes digitals.
- Fitxer necessaris per la realització de les diferents sessions pràctiques.

Convenis

Estructura del directori de treball:

Prac_x

|--- doc documentació / informe
|--- ip IP generades amb el Quartus
|--- misc blocs i models funcionals utilitzats per test.
|--- rtl codi sintetitzable del disseny.
|--- sdc fitxers amb les restriccions del disseny, utilitzat per l'eina de síntesis.
|--- sim directori de treball del ModelSim.
|--- syn directori de treball del Quartus.
|--- tb fitxers de test utilitzats per l'eina de simulació.

Codificació Verilog

1. Un mòdul per fitxer, on el fitxer ha de tenir el mateix nom que el mòdul.
2. Noms de senyals ports, registres:
   a. Han de ser descriptius.
   b. Entrades i Sortides primera lletra en majúscula.
   c. Senyals internes nomes de registres etc en minúscules.
   d. Paràmetres i directives en majúscules.
3. Estructura fitxer Verilog:

```
module name #(
   // declaració paràmetres

)(
   //llista input outputs comentant la seva funció

);
   // declaració de paràmetres locals

   // declaració de regs & wires comentant la seva funció

   // descripció del disseny: lògica combinacional i seqüencial.

endmodule
```

4. Comentaris al codi:
   a. Utilitzeu el comentari en línia // per explicar la funció de les senyals.
   b. Comenteu estructures de codi no cal comentar línia a línia, a menys que sigui necessari per la correcta comprensió del que fa.

Pràctica 1: Arquitectura d'un sistema digital

Durada: 2 sessions.

Objectius:

Aprendre a definir l'arquitectura d'un sistema digital en base a unes especificacions tècniques donades. En aquest cas un mestre SPI que compleixi amb els següents requisits tècnics.

Especificacions tècniques:

1. Mode de comunicació seleccionable, mitjançant CPol i CPha.
2. Freqüència de transmissió (del SCK) seleccionable, mitjançant un pre-escalat CPre de 4-bits.
3. Freqüència màxima del bus SPI 50 MHz.
4. Unitat mínima d'informació a transmetre 1 byte.
5. S'ha de poder habilitar i deshabilitar el mòdul.
6. Ha de poder controlar fins a 8 esclaus diferents.
7. El control del mòdul ha de ser a traves d'un conjunt de registres de 8 bits de dades i 2 bits d'adreces. L'escriptura d'aquests ha de ser síncrona i la lectura asíncrona. Ha d'incloure entre d'altres: Un registre de control on el MSB sigui el bit d'estat (ocupat o no). Un registre de memòria intermèdia (buffer) on s'escriurà la dada a transmetre. El mateix registre buffer s'actualitzarà amb l'últim byte rebut al finalitzar cada transmissió de 8 bits.
8. La transmissió ha d'iniciar-se automàticament després d'escriure en el buffer.

Tasques a realitzar:

En base als requisits tècnics enumerats anteriorment, determineu els mòduls necessaris per implementar el SPI mestre i descriviu la seva funcionalitat.

1. Definiu les entrades i sortides del mestre SPI a implementar.
2. Feu el diagrama de blocs bàsic (no RTL) que mostri les interconnexions entre blocs i les E/S del mestre SPI.
3. De cada bloc, dibuixeu el diagrama RTL detallat i enumereu-ne les seves entrades i sortides, exceptuant les màquines d'estats (on només cal indicar entrades i sortides).
4. Descriviu breument la funcionalitat de la màquina d'estats.
5. Descriviu la funcionalitat dels registres bit a bit.

Entrega:

L'informe que trobareu al campus virtual, que constarà de:

- Descripció breu de l'arquitectura del mestre SPI, enumerant cada mòdul i descrivint la seva funcionalitat.

- Diagrama RTL detallat complet. Indicant noms dels senyals i la seva mida (si són de més d'un bit). Feu l'esquema amb ordinador amb Visio, Draw.io, SchemDraw, InkScape, PowerPoint. Es desaconsella l'ús del Paint o similars.
- Taula amb les entrades i sortides del mòdul SPI i la seva funcionalitat.
- Taula de registres indicant si són d'escriptura i/o lectura i la funció de cada bit.
- Descripció breu l'operació (els passos a segueix) per realitzar la transmissió d'un byte.

Pràctica 2: Introducció al flux de disseny i síntesis en FPGA

Durada: 1 sessió

Introducció:

La Figura 7 mostra el diagrama RTL simplificat del mestre SPI. Un cop definida l'arquitectura del nostre sistema digital, els següents passos són codificar a nivell RTL i verificar-ne la funcionalitat, per finalment sintetitzar el sistema complet. En aquesta pràctica ens familiaritzarem amb els passos bàsics a seguir pel flux de disseny i síntesis en FPGA. Per fer-ho ens centrarem en la implementació dels registres de desplaçament que es mostren al diagrama RTL de la Figura 7.

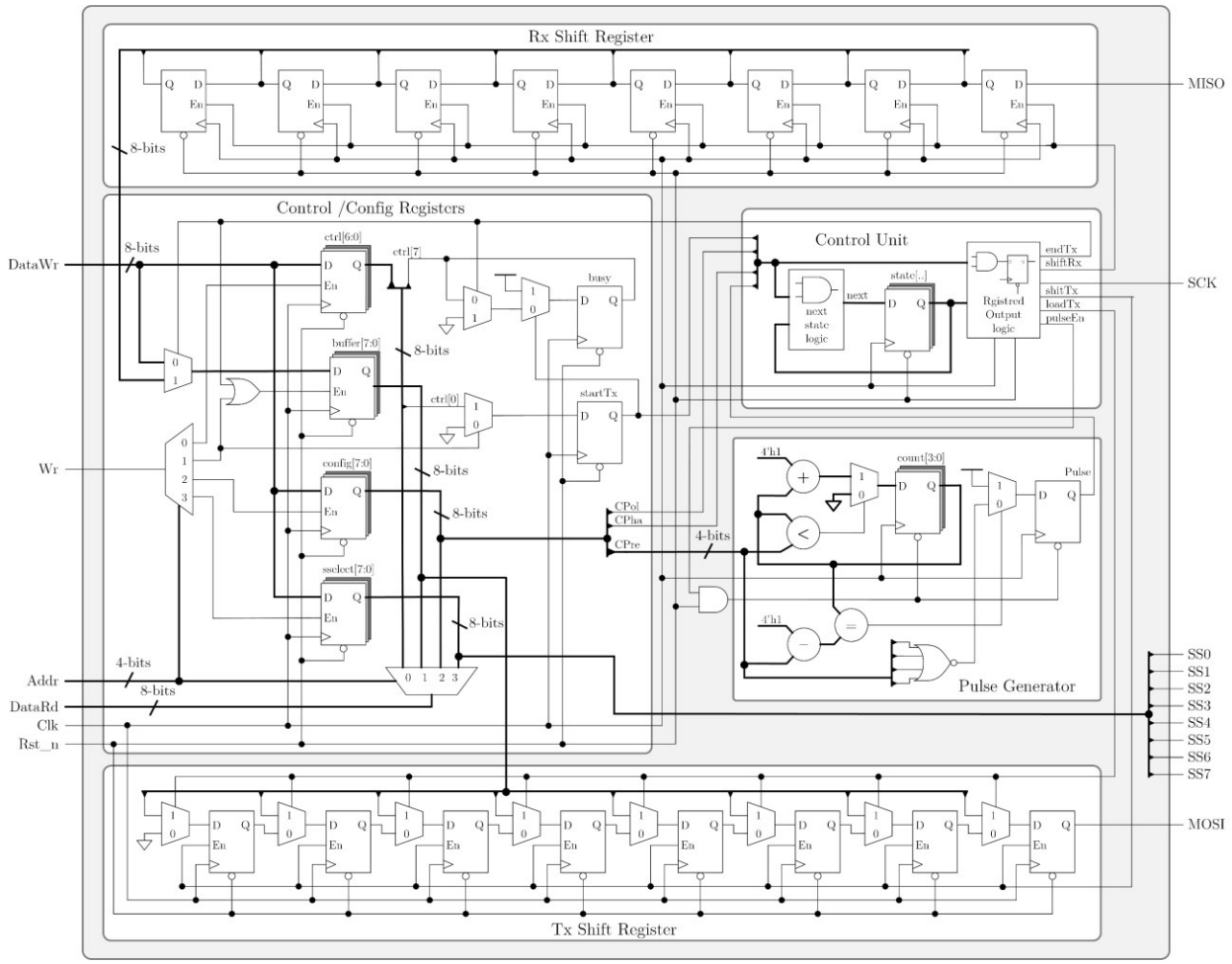

Figura 7. Arquitectura del mestre SPI a implementar.

Objectius:

- Codificar RTL de l'estructura descrita en base a les especificacions donades.
- Familiaritzar-se en l'ús de l'entorn de simulació ModelSim.
- Introducció al *self-checking* mitjançant l'ús de tasques, funcions pròpies del Verilog (*$display, $monitor, $wait, $force, $release*, ...)i la monitorització de senyals internes del dispositiu a verificar (DUT, de l'anglès Device Under Test).
- Familiaritzar-se en l'ús de l'entorn de síntesis Quartus Prime d'Altera i simulació post-implementació.

Tasques a realitzar:

1. Dissenyeu un registre de desplaçament genèric no-cíclic parametritzable de N-bits, que anomenarem *shiftreg*. El registres de desplaçament ha de permetre la càrrega en paral·lel de dades. Ha de tenir una entrada i sortida en sèrie i la sortida en paral·lel del contingut del registres. El registres ha de tenir una senyal d'habilitació i un reset asíncron actiu per nivell baix. Els ports d'entrada i sortida seran doncs:

Clk:	Rellotge del sistema.
Rst_n	Reset asíncron actiu per nivell baix.
Load	Habilita la carrega de dades en paral·lel. Actiu per nivell alt.
En	Habilitació del desplaçament. Actiu per nivell alt.
DataIn	Entada de dades en paral·lel de N-bits.
DataOut	Sortida de dades en paral·lel de N-bits.
SerIn	Entrada de dades sèrie, que s'insereixen per el bit menys significatiu (LSB).
SerOut	Sortida de dades sèrie que es correspon amb el bit més significatiu (MSB).

2. Examineu el fitxer de testbench (*tb_shiftreg.v*) que trobareu al campus virtual i instancieu el vostre registre de desplaçament com a DUT i verifiqueu-ne el funcionament.
3. Completeu el test del registre de desplaçament creant la següent tasca:

   test_serin, que carregui pel port d'entrada sèrie i comprovar que després de N cicles el valor de dins el registres es correspon al valor esperat. El valor esperat s'ha de calcular automàticament dins la mateixa tasca. Per tant, la tasca ha de tenir com a inputs: la dada a rebre pel port SerIn, i el nombre de desplaçaments que volem que fer.
4. Utilitzeu la tasca del sistema *$monitor* per veure l'estat de les entrades i sortides del registre de desplaçament en tot moment (excloent els senyals de Clk i Rst_n).
5. Simuleu i verifiqueu el correcte funcionament per un registre de 8-bits per diferents valors d'entrada i nombre de desplaçaments.
6. Verifiqueu visualment mitjançant el diagrama d'ones cada un dels test. Examineu els estímuls externs, les senyals internes i les sortides del vostre registre de desplaçament. També podeu examinar les variable de les tasques.

7. Sintetitzeu el registre de desplaçament de 8 bits amb el Quartus per una Cyclone V model 5CEBA4F23C7N.
   - Creeu un projecte nou. Consulteu la guia de Quartus que teniu al campus virtual.
   - Elaboreu el disseny i comproveu quins missatge d'*error*, *info*, i *alertes* ús dona.
   - Examineu el esquema RTL generat per el Quartus. És l'esperat?
   - Creeu un fitxer de restriccions temporals (shiftreg.sdc) dins la carpeta misc, que contingui la següent comanda.

     create_clock -name clk100MHz -period 10.0 [get_ports Clk]

     Aquesta comanda defineix un rellotge de 100 MHz al port d'entrada *Clk*. Aquest rellotge serà utilitzat per l'eina de síntesis per realitzar l'anàlisi temporal estàtic i comprova si el disseny implementat pot funcionar aquesta freqüència.
   - Afegiu el fitxer SDC al projecte de Quartus.
   - Assigneu les entrades i sortides als pins de la FPGA indicats en La seguent taula:

Senyal	**Element**	**Pin**	**Senyal**	**Element**	**Pin**
En	SW9	PIN_AB12	DataOut[7]	LED7	PIN_U1
SerIn	SW8	PIN_AB13	DataOut[6]	LED6	PIN_U2
DataIn[7]	SW7	PIN_AA13	DataOut[5]	LED5	PIN_N1
DataIn[6]	SW6	PIN_AA14	DataOut[4]	LED4	PIN_N2
DataIn[5]	SW5	PIN_AB15	DataOut[3]	LED3	PIN_Y3
DataIn[4]	SW4	PIN_AA15	DataOut[2]	LED2	PIN_W2
DataIn[3]	SW3	PIN_T12	DataOut[1]	LED1	PIN_AA1
DataIn[2]	SW2	PIN_T13	DataOut[0]	LED0	PIN_AA2
DataIn[1]	SW1	PIN_V13	Clk	KEY0	PIN_U7
DataIn[0]	SW0	PIN_U13	Rst_n	KEY1	PIN_W9
SerOut	LED9	PIN_L1	Load	KEY2	PIN_M7

8. Sintetitzeu de nou el disseny i examineu el *Compilation Report*.
   - Compleix els requisits temporals? Quina és la freqüència màxima del disseny?
   - Hi ha alguna E/S del vostre mòdul sense definir (*unconstrained*)?
   - Quants i quins recursos s'utilitzen?
9. Configureu el projecte de Quartus per poder simular la *netlist* generada. Consulteu la guia *guia_SimulacióNetlisQuartus* que trobareu al campus virtual.
   - Quina diferencies veieu respecte al simulació RTL? Examineu el retard de les senyals respecte el flanc de rellotge.
10. Per últim carregeu el disseny a la placa de desenvolupament DE0-CV i comproveu que efectivament funciona el registre de desplaçament.

Entrega:

Fer demostració al professor un cop s'ha implementat a la DE0-CV. En cas de no poder fer la demostració a l'aula, heu de gravar un vídeo que adjuntareu dins la carpeta doc.

Un fitxer ZIP amb el directori de treball:

1. A la carpeta **rtl**: el codi RTL sintetitzable.
2. A la carpeta **tb**: el codi del testbench complet.
3. A la carpeta **misc**: el fitxer de restriccions temporals (.sdc).
4. Dins la carpeta **doc** hi heu de posar l'informe complimentat que trobareu al campus virtual, que constarà de:
   - Enumerar les tasques del testbench i explicar breument què fan.
   - Captures de les simulacions, amb una explicació breu i ressaltant les zones d'interès.
   - Captura del terminal del ModelSim amb els missatges de l'auto verificació.
   - Captura del esquema RTL resultant de la síntesi amb el Quartus (expandiu les caixetes!).
   - Taula on consti freqüència màxima d'operació, nombre de portes utilitzades de cada tipus i percentatge d'ocupació de la Cyclone V model 5CEBA4F23C7N.

Pràctica 3: Disseny RTL i Verificació

Durada: 4 sessions

Introducció:

En la pràctica 2 ens vàrem familiaritzar amb el flux de disseny aplicat a un registre de desplaçament. Durant les pròximes 4 sessions dissenyarem i verificarem la resta de blocs que composen el mestre SPI (Figura 7 del guió Lab2).

Objectius:

- Codificar RTL diferents estructures descrites en base a les especificacions donades.
- Planificació del test d'un bloc digital amb *self-checking*.
- Verificació funcional pre- i post-síntesis en FPGA.

P3 Sessió 1: Generador de polsos

Introducció:

El següent mòdul del mestre SPI que dissenyarem serà el generador de polsos, que s'utilitza per temporitzar les transicions de les senyals del bus SPI.

Objectius:

- Codificar RTL en base a les especificacions donades.
- Verificació pre- i post-síntesis.

Tasques a realitzar:

11. Dissenyeu el generador de polsos de la Figura 7 (del guió Lab2) anomenat *pulse_generator*. Aquest ha de ser parametritzable i ha de generar un pols cada N cicles de rellotge. El reset del mòdul ha de ser asíncron i actiu per nivell baix. Noteu que el senyal d'habilitació i del mòdul ataca el mateix reset.
12. Dissenyeu el testbench (*tb_pulsegenerator.v*) per verificar el correcte funcionament del generador de polsos que utilitzi tasques i/o lògica per automatitzar-ne la verificació. El test ha de verificar que el nombre de cicles de rellotge entre polsos consecutius és el desitjat i indicar temps transcorregut entre dos polsos consecutius. Ajuda: *$realtime $time $monitor $display $while*.
13. Simuleu i verifiqueu el correcte funcionament per un generador de polsos de 8-bits pels valors 0, 1, 63 i 255.
14. Sintetitzeu un generador de polsos de 8 bits amb el Quartus i simuleu la *netlist* generada per verificar-ne el funcionament per una Cyclone V model 5CEBA4F23C7N. Pareu atenció als missatges d'informació i alertes. No us oblideu del fitxer de restriccions (SDC). Els pins d'entrada i sortida han d'anar connectats als següents elements.

- Utilitzeu els interruptors del SW0 al SW7 per introduir el nombre de cicles entre polsos.
- Utilitzeu un polsador (Key0) per generar el rellotge del sistema.
- Utilitzeu el polsador (Key1) com a reset del sistema.
- Feu servir el LED0 per connectar el senyal de sortida del temporitzador.

Nota: consulteu el manual de la placa de desenvolupament per saber quins pins de configuració es corresponen amb cadascun dels elements esmentats.

15. Comproveu que el disseny compleix els requisits temporals.

Entrega:

Fer demostració al professor un cop s'ha implementat a la DE0-CV. En cas de no poder fer la demostració a l'aula, heu de gravar un vídeo que adjuntareu dins la carpeta doc.

Un fitxer ZIP amb el directori de treball:

5. A la carpeta **rtl**: el codi RTL sintetitzable.
6. A la carpeta **tb**: el codi del testbench complet.
7. A la carpeta **misc**: el fitxer de restriccions temporals (.sdc).
8. Dins la carpeta **doc** hi heu de posar l'informe complimentat que trobareu al campus virtual, que constarà de:
   - Enumerar les tasques del testbench i explicar breument què fan.
   - Captures de les simulacions, amb una explicació breu i ressaltant les zones d'interès.
   - Captura del terminal del ModelSim amb els missatges de l'auto verificació.
   - Captura del esquema RTL resultant de la síntesi amb el Quartus (expandiu les caixetes!).
   - Taula on consti freqüència màxima d'operació, nombre de portes utilitzades de cada tipus i percentatge d'ocupació de la FPGA Cyclone V model 5CEBA4F23C7N.

P3 Sessió 2: Registres de Configuració i Control

Introducció:

En aquesta sessió ens centrarem el disseny del test per verificar el conjunt de registres de configuració i control que us podeu descarregar del campus virtual.

Objectius:

- Planificació del test exhaustiu d'un bloc digital amb *self-checking*.
- Utilització de declaracions seqüencials aplicades a testbench com són, *fork-join, wait, repeat*, entre d'altres.
- Disseny de tasques i funcions.
- Access jeràrquic a elements (com registres o *wires*) de sub-blocs.

Tasques a realitzar:

1. Examineu el contingut del fitxer *spi_regs.v*, identifiqueu les diferents estructures i comenteu el codi.
2. Feu una llista amb les verificacions/tests a fer. Un cop feta comenteu-la amb el professor.
3. Examineu el contingut del fitxer *tb_spi_regs.v*.
4. Simuleu i verifiqueu el conjunt de registres (*spi_reg.v*) tot completant el testbench (*tb_spi_regs.v*) on s'indica. Seguiu l'ordre establert que s'indica en el mateix fitxer.
5. Sintetitzeu i verifiqueu el disseny per la FPGA Cyclone IV E EP4CE22F17C6.
    - Creeu el fitxer de restriccions de disseny (*spi_regs.sdc*) on hi heu de definir un rellotge del sistema de 50MHz.
    - Creeu un projecte de Quartus a la carpeta (syn) anomenat spi_reg, per la
    - Afegiu els fitxer verilog i el SDC al projecte.
    - Configureu el Quartus per simular a nivell de porta (gate level). Seguiu la guia que trobareu al campus virtual.
    - Sintetitzeu el conjunt de registres (*Processing>Start>Start Analysis & Synthesis*). Pareu atenció als missatges d'informació i alertes.
    - Assigneu els pins d'entrada i sortida als GPIOs (a la vostra elecció). Mireu la fulla d'especificacions de la DE0-Nano que trobareu al campus virtual.
    - Abans de compilar el disseny assegureu-vos de desactivar l'opció de llençar la simulació automàticament. Premeu a *Assignments>Settings>EDA Tool Settings* desmarqueu l'opció *Run gate-level simulation...*. Ja podeu compilar el disseny.
        - Comproveu que el disseny compleix els requisits temporals.
        - Inspeccioneu el esquemàtics generats (*Tools → Netlist Viewers*) a nivell:
            - *RTL Viewer*
            - *Technology Map Viewer (Post-Fitting).*

- Quines diferencies hi trobeu? Busqueu el registre de la senyal busy.

- Examineu la netlist (.vo) i el fitxer de retards (.sdo) generats al directori del vostre projecte de Quartus: *\Lab3s2\syn\simulation\modelsim*

6. Simuleu la netlist generada amb el mateix fitxer de testbench. Per simular a nivell de porta i veure els retards interns seguiu la guia (Simulacio Netlist Quartus) que trobareu al campus virtual.
   - Per la verificació post-síntesis, cal modificar la referència jeràrquica al senyal intern busy fet al testbench. Substituïu-la per la següent: *tb_spi_regs.DUT.busy.q.*
     - A què respon aquesta modificació?
   - Un cop guardats els canvis en el fitxer de testbench, inicieu la simulació. Per fer-ho cliqueu a Tools > Run Simulation Tool > Gate Level Simulation...
     - Examineu el diagrama d'ones i la finestra de transcripció de l'eina de simulació.
     - Examineu el comportament de la senyal de busy (registre busy.q).

Entrega:

Un fitxer ZIP amb el directori de treball:

9. A la carpeta **rtl**: el codi RTL.
10. A la carpeta **tb**: el codi del testbench complet.
11. A la carpeta **sdc**: el fitxer de restriccions temporals (.sdc).
12. Dins la carpeta **doc** hi heu de posar l'informe complimentat que trobareu al campus virtual, que constarà de:
    - Enumerar les tasques del testbench i explicar breument què fan.
    - Captures de les simulacions funcionals, amb una explicació breu i ressaltant les zones d'interès.
    - Captura del terminal del ModelSim amb els missatges de l'auto verificació.
    - Captura del esquema RTL resultant de la síntesi amb el Quartus i taula on consti freqüència màxima d'operació, i recursos utilitzats de la FPGA Cyclone IV E EP4CE22F17C6.
    - Captures de les simulacions post-síntesis, amb una explicació breu i ressaltant les zones d'interès.

P3 Sessió 3-4: Unitat de Control del mestre SPI

Introducció

Un cop dissenyats els mòduls bàsics del nostre mestre SPI, només ens falta dissenyar la unitat de control encarregada de gestionar la transmissió d'un byte i la generació del senyal SCK. Aquesta unitat de control l'implementarem per mitja d'una màquina d'estats.

Objectius:

- Disseny RTL d'una màquina d'estats.
- Disseny del top amb les instàncies dels diferents sub-mòduls del mestre SPI seguint l'arquitectura definida.
- Auto-Test i verificació del mestre SPI, reaprofitant tasques i funcions ja desenvolupades entre d'altres.

Tasques a realitzar:

1. Determineu les entrades i sortides de la màquina d'estats, en base a l'arquitectura del sistema.
2. Identifiqueu i enumereu la seqüencia que ha de seguir per enviar/rebre 1 byte. Dibuixeu el diagrama temporal de les entrades i sortides de la unitat de control.
3. Dibuixeu el diagrama d'estats que implementi la seqüencia establerta.

   Lectura obligatòria del campus virtual sobre codificació de màquines d'estat.
4. Codifiqueu la màquina d'estats que anomenarem *spi_cu*.
5. Genereu un fitxer anomenat *spi_top* que serà el top del nostre mestre SPI on heu d'instanciar tots els mòduls generats en les sessions anteriors incloent el *spi_cu*.
6. Dissenyeu el test que verifiqui la transmissió i recepció de dades pels 4 modes d'operació on el DUT és el *spi_top*.
   - Feu el test utilitzant els registres de configuració i control per realitzar transferències de 1 byte en els 4 modes de comunicació que permet el mestre SPI. Poder reutilitzar les tasques dissenyades durant les sessions anteriors.
   - Comproveu que escriviu i llegiu correctament els registres de configuració i control.
   - Dissenyeu una tasca (waitEnd) que monitoritzi el bit d'estat *busy* del registre de control, és a dir, ha d'esperar que el bit *busy* passi de 1 a 0.
   - Al campus virtual trobareu un model d'esclau SPI molt senzill compatible amb tots els modes de comunicació del bus SPI. En funció del mode de funcionament (que s'indica mitjançant un port d'entrada) l'esclau envia un byte de diferent valor. És el fitxer anomenat *spislave_fm.v*. Recordeu adjuntar el model funcional als projectes de ModelSim.
7. Creu un fitxer SDC on heu d'especificar:

- Un rellotge del sistema de 100 MHz amb la comanda *create_clock*.
- Els senyals d'entrada i sortida del xip (FPGA/ASIC) tenen un retard associat. Aquest retard és pot especificar en el fitxer de restriccions SDC per tal que les eines de síntesis el tinguin present a l'hora de sintetitzar i de "mapejar" i implementar el nostre disseny. Per fer-ho, s'utilitzen les comandes *set_input_delay* i *set_output_delay*.

  set_input_delay -clock <clockName> <delay value in ns> [get_ports <portName>]

  set_output_delay -clock <ClockName> <delay value in ns> [get_ports <portName>]

  La ordre *set_input_delay* especifica el temps d'arribada de dades als ports d'entrada especificats en relació amb el rellotge especificat per l'opció -clock. On el rellotge ha de fer referència al nom del rellotge al disseny. Per tant, determinarà el temps disponible per dur la senyal d'entrada des del port fins el registres intern (Tclk - InputDelay). De la mateixa manera l'ordre *set_ouptut_delay* especifica el temps requerit en que dades han d'estar als ports de sortida especificats en relació amb el rellotge especificat per l'opció *-clock*. (Noteu que les comandes tenen més paràmetres que els esmentats. Es recomana consultar l'ajuda del Quartus referent a les ordres TCL.)

  En el nostre cas modelitzarem totes les entrades i sortides amb un retard de 2 ns. Per fer-ho heu d'incloure les següents comandes al fitxer SDC, on heu de substituir el <nomClk> pel nom que heu donat al rellotge de 100 MHz que heu definit.

  set_input_delay -clock <nomClk> 2.0 [all_inputs]

  set_output_delay -clock <nomClk> 2.0 [all_outputs]

8. Sintetitzeu i verifiqueu el disseny per FPGA Cyclone IV E EP4CE22F17C6. No us oblideu d'incloure el fitxer de SDC. Definiu els pins d'entrada i sortida al GPIO.
9. Comproveu que el disseny està completament definit (no té cap *unconstrained path*) i que compleix els requisits temporals.
10. Simuleu la *netlist* generada amb el mateix fitxer de testbench. Per simular a nivell de porta i veure els retards interns seguiu la guia (Simulacio Netlist Quartus) que trobareu al campus virtual.

Entrega:

Un fitxer ZIP amb el directori de treball:

13. A la carpeta **rtl**: el codi RTL.
14. A la carpeta **tb**: el codi del testbench complet.
15. A la carpeta **sdc**: el fitxer de restriccions temporals (.sdc).
16. Dins la carpeta **doc** hi heu de posar l'informe complimentat que trobareu al campus virtual, que constarà de:
    - Tipus i explicació de la màquina d'estats implementada i diagrama d'estats.
    - Enumerar les tasques del testbench i explicar breument què fan.

- Captures de les simulacions funcionals, amb una explicació breu i ressaltant les zones d'interès.
- Captura del terminal del ModelSim amb els missatges de l'auto verificació.
- Captura del esquema RTL resultant de la síntesi (no oblideu el diagrama d'estats) amb el Quartus i taula on consti freqüència màxima d'operació, i recursos utilitzats de la FPGA Cyclone IV E EP4CE22F17C6.
- Captures de les simulacions post-síntesis, amb una explicació breu i ressaltant les zones d'interès.

Pràctica 4: Disseny i Síntesis d'una Estació Meteorològica

Durada: 2 sessions

Introducció

Un cop dissenyat i verificat el mestre SPI, l'utilitzarem com a interfície de comunicacions SPI d'una estació meteorològica simple basada en el sensor BME280. El sensor BME280 és un sensor digital combinat d'humitat, pressió i temperatura, que permet monitoritzar les condicions ambientals en temps real.

Objectius:

- Analitzar i combinar mòduls donats per generar un sistema digital complex amb interfície d'usuari, adquisició, post-processat i visualització de dades.
- Definir l'arquitectura del l'estació meteorològica, en base a blocs donats. Seguint els criteris de disseny adients, tal com l'ús de sincronitzadors, restabliments per encesa (power-on-reset), implementació d'IPs (de l'anglès Intellectual Property), entre d'altres.
- Definició de restriccions temporals necessàries per el correcte funcionament del sistema (SDC).
- Implementació en FPGA Cyclone V 5CEBA4F23C7N de la placa de desenvolupament DE0-CV.

Requisits tècnics de l'estació meteorològica.

11. Ha de fer una adquisició de temperatura, pressió i humitat per segon.
12. Les dades es mostraran per les pantalles de set segments, en format BCD (de l'anglès Binary-Coded Decimal)
13. L'usuari ha de poder seleccionar quina magnitud (temperatura, pressió o humitat) es mostra per les pantalles, utilitzant els interruptors.

SW2	SW1	SW0	Magnitud
x	x	1	Temperatura
x	1	0	Pressió
1	0	0	Humitat

14. Ha de tenir una bandera d'error en el LED0.
15. Ús d'un PLL per generar dos dominis de rellotge diferents: un de 100MHz que serà el rellotge del sistema, i un de 1MHz per controlar quan es realitza una nova mesura.
16. Reset extern asíncron actiu per nivell baix (al Key0).

17. Recordeu d'implementar el que es coneix com restabliment d'encesa (power-on-reset) per cada domini de rellotge.
18. El mòdul del sensor BME280 es connectarà al GPIO 1, de la següent manera:

Senyal	BME PIN	DE0-CV
CS	CS	GPIO_1_D33
SCK	SCK	GPIO_1_D27
MOSI	SDI	GPIO_1_D31
MISO	SDO	GPIO_1_D29
VDD	VIN	VCC3P3
GND	GND	GND

Tasques a realitzar:

1. Descarregueu-vos i descomprimiu el directori lab4 del campus virtual.
2. Examineu el contingut de les diferents carpetes i fitxers.
3. Enumereu i descriviu els estats del mòdul ***bme280_reader***, i dibuixeu-ne el diagrama d'estats.
4. Feu el diagrama de blocs (no RTL) de l'arquitectura de l'estació meteorològica que implementareu, en base al requisits anteriors.
5. Feu una taula enumerant els mòduls que utilitzeu i la seva funció principal.
6. Genereu l'entitat de primer nivell (top de l'estació meteorològica) en base a l'arquitectura definida. Tant el mòdul com el fitxer s'han d'anomenar ***top_meteo_de0cv.*** Recordeu d'utilitzar sincronitzadors per les senyals que creuen dominis de rellotge.
7. Penseu i llisteu quins tests i simulacions RTL heu de fer per verificar que el sistema funciona. Especifiqueu-los per cada mòdul.
8. Verifiqueu el correcte funcionament de la vostra estació meteorològica simulant a nivell RTL els tests que heu definit abans. Podeu utilitzar el fitxer de test així com els diferents models funcionals que trobareu a les carpetes *tb* i *misc*.
9. Implementeu el disseny a la DE0-CV (Cyclone V 5CEBA4F23C7N):
   A. Afegiu el fitxer de restriccions temporals (SDC) al projecte i examineu-ne el contingut. Podeu consultar l'ajuda del mateix Quartus per saber que fa cada una de les comandes.
   B. Actualitzeu el nom de les entrades i sortides del fitxer SDC així com el nom de la instancia del PLL.
   C. Per generar la IP del PLL, heu de buscar a la catàleg d'IPs del Quartus (*Tools > IP Catalog > PLL Intel FPGA IP*).
   D. Genereu la IP a la carpeta *ips*, amb el nom de *pll_cv.v*, i configurar-lo com es mostra en la Figura 8.

E. Aneu a la carpeta *ips* i comproveu els fitxers generats. Per instanciar el PLL al vostre disseny utilitzeu el fitxer *pll_cv.v*. Quan l'instancieu poseu la senyal de reset a zero.
F. Per fer l'assignació de pins, consulteu el manual de la DE0-CV que trobareu al campus virtual.

10. Comproveu els missatges d'alerta i error així com l'anàlisi temporal, es compleixen els requisits temporals? Quin és el problema? Com es podria solucionar?
11. Connecteu el sensor BME280 a la placa de desenvolupament DE0-CV i comproveu que podeu comunicar-vos amb ell.

Entrega:

Fer demostració al professor un cop s'ha implementat a la DE0-CV. En cas de no poder fer la demostració a l'aula, heu de gravar un vídeo que adjuntareu dins la carpeta doc.

Un fitxer zip amb el directori de treball:

17. A la carpeta **rtl**: el codi RTL, amb el *top_meteo_de0cv* i *bme280_reader* comentats.
18. A la carpeta **tb**: el codi del testbench comentat.
19. A la carpeta **sdc**: el fitxer de restriccions temporals (.sdc) comentat**.**
20. Dins la carpeta **doc** hi heu de posar l'informe complimentat que trobareu al campus virtual, que constarà de:
    - Diagrama de blocs de l'arquitectura de l'estació meteorològica implementada i descripció breu de l'arquitectura, enumerant cada mòdul i descrivint la seva funcionalitat, així com les seves entrades i sortides.
    - Diagrama d'estats del mòdul *bme280_reader*, així com una descripció breu del seu funcionament enumerant els diferents passos que fa fins a realitzar la segona tanda d'adquisició de dades.
    - La llista de tests i les captures de les simulacions RTL corresponents, amb una explicació breu i ressaltant les zones d'interès.
    - Captura (o captures) del esquema RTL generat pel Quartus (expandiu les caixetes) i taula de recursos utilitzats.
    - Preguntes que trobareu a l'informe.

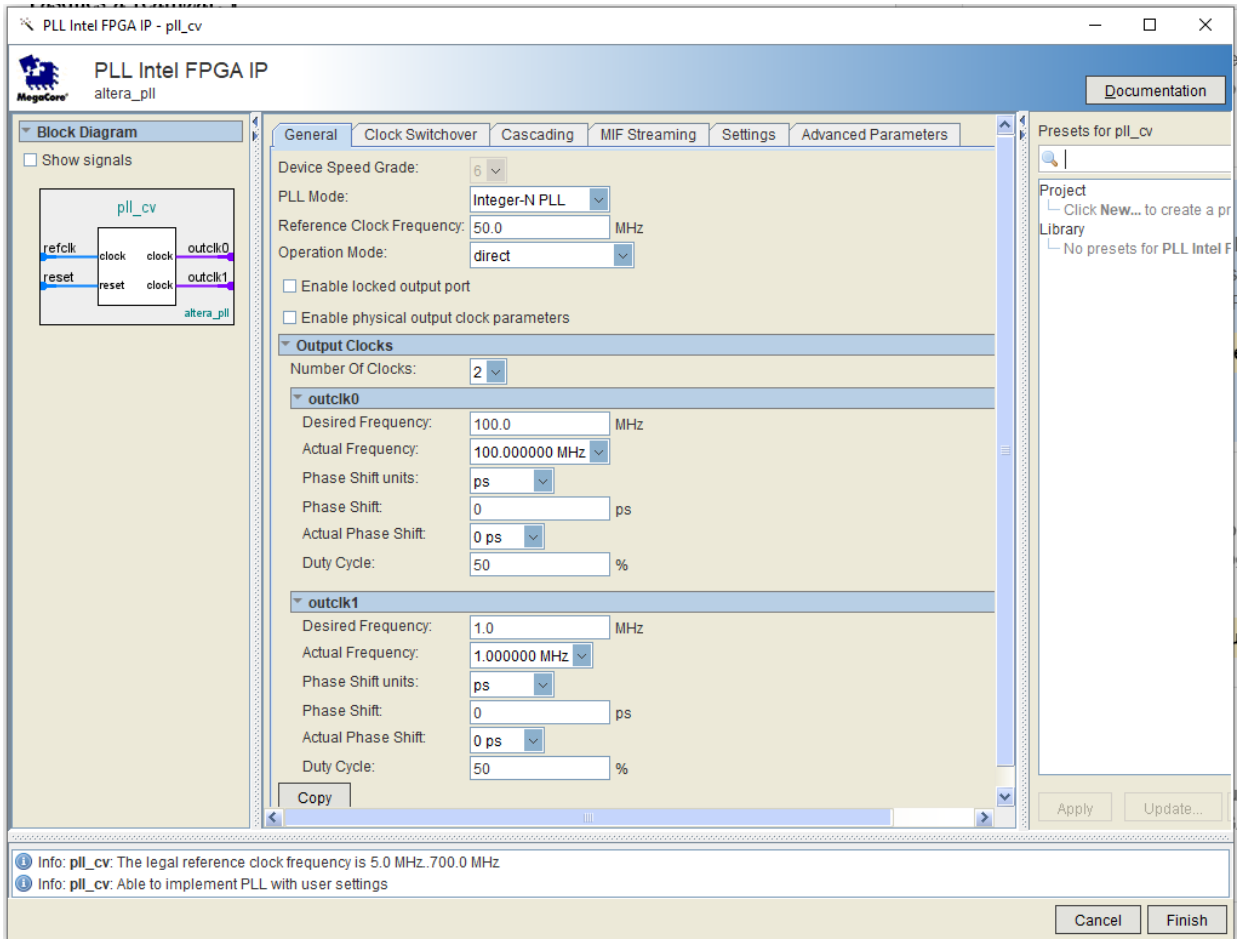

Figura 8. Paràmetres de configuració de la IP del PLL.