UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Departament d'Enginyeria Minera, Industrial i TIC

Arquitectura de computadors

Pràctica 1

8 de març de 2021

Disseny del banc de registres
i
del comptador de programa

Antoni Escobet i Josep Antoni Riera

PRÀCTICA 1 – Disseny del banc de registres i del PC

1. Objectius

En aquesta practica es pretén:

- Repassar els conceptes relacionats amb el banc de registres d'un processador MIPS i el seu comptador de programa.
- Dissenyar un banc de 32 registres de 32 bits.
- Dissenyar el comptador de programa (PC)
- Ampliar els coneixements sobre el llenguatge VHDL i l'edició amb QUARTUS.

2. Material

L'únic material necessari per la realització d'aquesta pràctica és el paquet de programari d'ALTERA, el Quartus instal·lat als ordinadors del laboratori, i que us podeu descarregar gratuïtament de la pàgina web d'Altera (http://www.altera.com).

La simulació del vostre disseny s'ha de fer amb el simulador que proporciona el mateix paquet de programari d'Altera (ModelSim-Altera)

3. Problema proposat

En aquesta pràctica s'ha de realitzar el disseny del banc de registres i del comptador de programa que s'ha vist a teoria.

El disseny del processador vist a classe, amb el seu camí de dades i la seva unitat de control permet executar un conjunt reduït d'instruccions relacionades amb la unitat aritmètica i lògica, es a dir, que pot realitzar sumes i restes de nombres sencers de 32 bits, i les operacions lògiques AND i OR sobre 32 bits. També pot comparar dos operants per saber si un és major que l'altre.

Per tal de poder fer els salts condicionals, serà necessari que l'ALU proporcioni a la unitat de control, l'indicador de si el resultat de l'operació val zero (Z).

El banc de registres ha de subministrar les dades dels programes a la unitat aritmètica lògica perquè realitzi els càlculs requerits. Com ja sabeu, els operands de les instruccions aritmètiques i lògiques no poden ser variables qualssevol (com les usades en els llenguatges d'alt nivell); han de procedir d'una sèrie limitada de posicions especials denominades **registres**. Cada registre és capaç d'emmagatzemar una paraula de memòria. Així, ja que l'arquitectura MIPS que heu estudiat és de 32 bits, la grandària de cada registre serà també de 32 bits. Com sabeu, el nombre de registres existents en l'arquitectura d'un processador no és il·limitat (la superfície del processador és limitada), i en el cas que ens ocupa és igual a 32. La notació emprada per a referir-nos a ells és $0, $1,…, $31.

Com s'ha dit anteriorment, les instruccions aritmètiques i lògiques necessiten utilitzar operands ubicats en registres. No obstant, inicialment les instruccions i les dades del programa es troben a la memòria. Per tant, abans de poder utilitzar un dada del programa serà necessari portar-ho des de memòria a un registre determinat. A més, és possible que necessitem escriure a la memòria algun resultat emmagatzemat en un registre. Per portar una dada des de la memòria a un registre es necessita una operació de lectura sobre memòria. De la mateixa forma, per emmagatzemar un dada continguda en un registre a la memòria s'ha de fer una operació d'escriptura sobre memòria. L'arquitectura MIPS ofereix dues instruccions específiques per a realitzar la

lectura d'una paraula de memòria i l'escriptura o emmagatzemament d'una paraula en memòria. Aquestes són: lw, *load word i* sw, *store word*, respectivament.
El banc de registres que heu de dissenyar està format per 32 registres de propòsit general, numerats del 0 al 31. **El registre $0 sempre conté el valor 0 perquè així està establert pel maquinari**. En el MIPS hi ha establert una sèrie de convencions sobre com cal utilitzar els registres (per exemple, en quins registres es passen els arguments de les funcions, quin és el punter de pila, els reservats per al sistema operatiu, etcètera), però no afecta en el disseny del maquinari que ens ocupa.

Així mateix, el comptador de programa és una altra unitat fonamental en el maquinari d'un processador. Bàsicament, el comptador de programa és un registre que s'ha d'actualitzar després d'executar una instrucció. Aquesta actualització pot ser un increment constant o bé la càrrega d'una nova adreça en el cas de les instruccions de salt. Al inici de cada instrucció, s'accedeix a l'adreça de memòria apuntada pel contingut del comptador de programa per aconseguir la instrucció que s'ha d'executar. En funció d'aquesta instrucció, el contingut del comptador de programa s'incrementa en 4 unitats (grandària en bytes de les instruccions del processador) o s'actualitza amb un valor nou que s'obté a partir del valor actual del comptador de programa i d'alguns bits codificats en la pròpia instrucció. En aquest cas, al tractar-se d'un processadors multicicle, la operació per incrementar el valor del PC es farà fora d'aquesta unitat funcional.

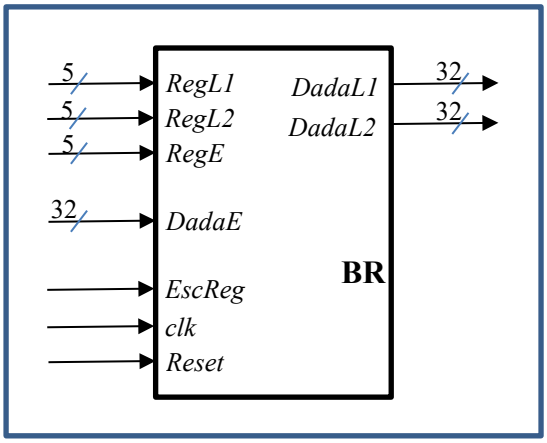
4. Descripció de la practica

En aquesta secció es descriu com realitzar el banc de registres i el comptador de programa descrits a l'apartat anterior. Anirem realitzant un disseny incremental. En primer lloc, després de recordar l'estructura estudiada a classe sobre la relació entre les diverses unitats funcionals del processador, presentarem la definició de les entitats VHDL a realitzar. Començarem pel banc de registres, partint d'un senzill registre de 32 bits que anirem ampliant en la seva funcionalitat fins a la construcció del banc complet. Finalment, implementarem el comptador de programa.

4.1. Relació del banc de registres i del comptador de programa amb la resta d'unitats

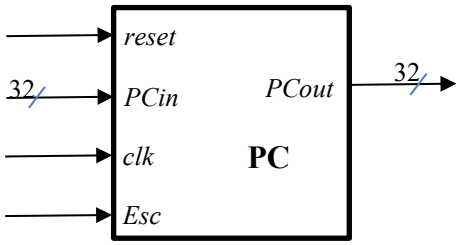
Quina informació hem de tindre present a l'hora de realitzar els dissenys?

Banc de registres:



Hi ha tres entrades per indicar sobre quins registres es vol actuar
  - ***RegL1***, ***RegL2*** indiquen els dos registres a llegir.
  - ***RegE*** indica el registre a actualitzar (escriure).
- Dues sortides per posar-hi les dades que es llegeixen:
  - ***DadaL1*** indica el bus de sortida del registre apuntat per *RegL1* que es llegeix.
  - ***DadaL2*** indica el bus de sortida del registre apuntat per *RegL2* que es llegeix.
- Una entrada de dades per a les operacions d'escriptura sobre el registre:
  - ***DadaE*** indica l'entrada de la informació a escriure sobre el registre apuntat *RegE*.
- Línies de control:
  - ***EscReg*** és el senyal d'escriptura per capturar la dada present a *DadaE* i emmagatzemar-ho al registre apuntat per *RegE*.
  - El senyal de rellotge del sistema (***clk***) per a determinar o sincronitzar el moment de l'escriptura sobre els registres, que s'activa amb el flanc de baixada.
  - El senyal de (***reset***) per deixar a zero tots els registres.

Pel comptador de programa s'ha de considerar:



- Una entrada ***clk***, que és el rellotge del processador. Serveix per sincronitzar els diferents esdeveniments que s'hauran de realitzar sobre el comptador de programa.
- Una entrada ***PCin.*** Valor que s'ha de guardar al comptador de programa.
- Una entrada ***Esc***, per memoritzar el valor de l'entrada ***PCin*** al comptador de programa.
- La sortida del comptador de programa ***PCout***. Aquesta sortida es mostra sempre.
- Afegirem un senyal de ***reset*** per a inicialitzar a zero el comptador de programa.

4.2. Definició de l'entitat: Banc de registres

A partir dels senyals indicats a l'apartat anterior, la definició de l'entitat del banc de registres serà:

```
entity BancRegistres is
    port (  clk :          in STD_LOGIC;
            reset:         in STD_LOGIC;
            EscReg:        in STD_LOGIC;
            RegL1:         in STD_LOGIC_VECTOR (4 downto 0);
            RegL2:         in STD_LOGIC_VECTOR (4 downto 0);
            RegE:          in STD_LOGIC_VECTOR (4 downto 0);
            DadaE:         in STD_LOGIC_VECTOR (31 downto 0);
            DadaL1:        out STD_LOGIC_VECTOR (31 downto 0);
            DadaL2:        out STD_LOGIC_VECTOR (31 downto 0)
        );
end BancRegistres;
```

4.2.1. Guia de disseny del Banc de registres

Una forma, entre moltes altres, de crear el banc de registres, és utilitzant els "array" de vhdl per crear una matriu de dades. Les matrius son una col·lecció d'elements del mateix tipus de dades als que s'accedeix mitjançant un índex. N'hi ha de monodireccionals (un sol índex) o multidimensionals (varis índex) i poden estar emmarcats en un rang o pot ser lliure, donant una matriu d'una dimensió infinita.

Exemples:

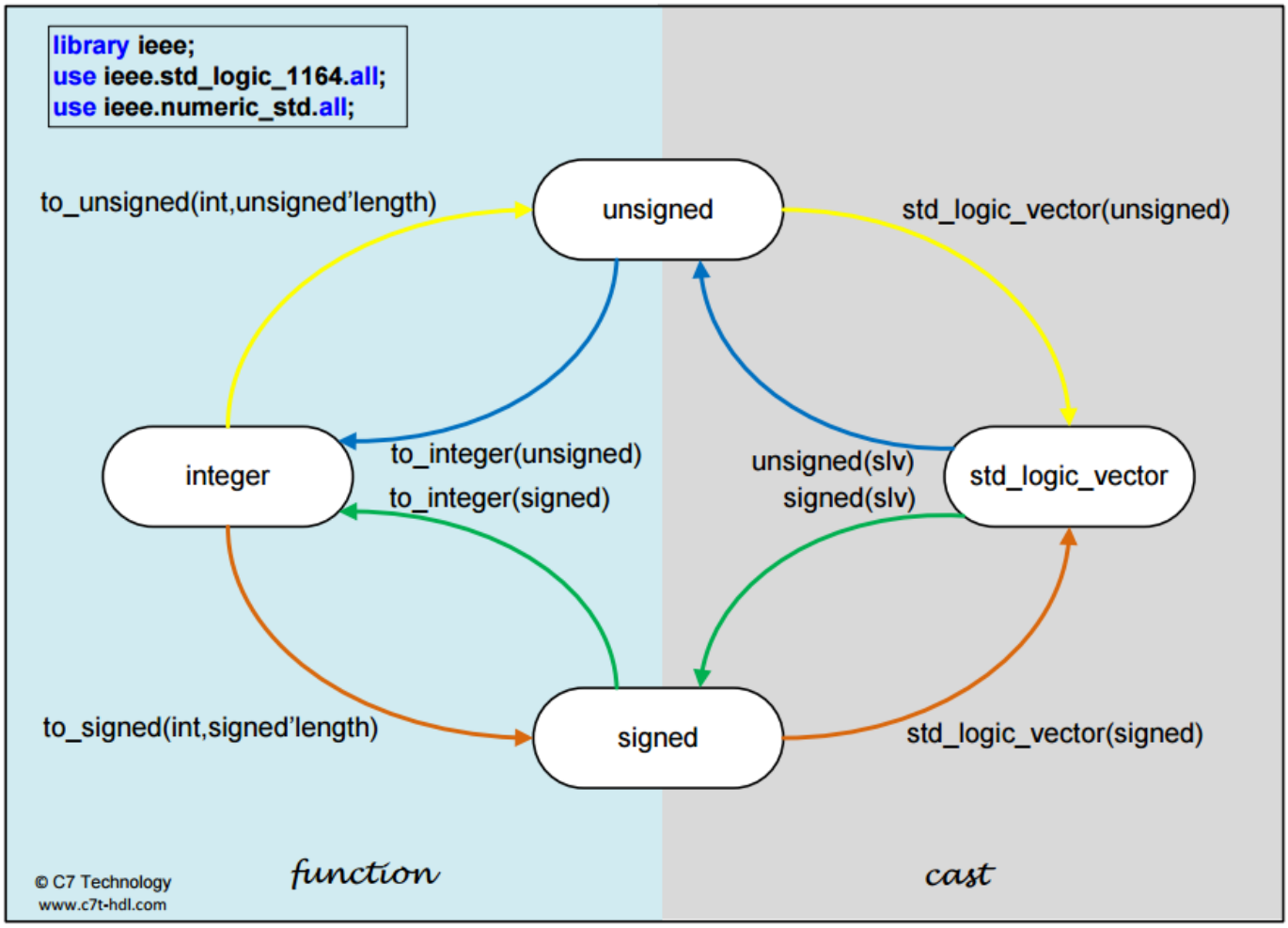
```
type word is array (31 downto 0) of bit;
type vector is array (natural range <>) of integer;
type byte_array is array (integer range <>) of std_logic_vector(7 downto 0);
....
```

Els dos últims exemples mostren una matriu amb un índex sense rang i serveix qualsevol sencer. Quan es declari la dada, s'haurà de posar límits a la matriu:

```
STD_LOGIC_VECTOR dada: vector(1 to 64);
```

Als elements d'una matriu s'accedeix mitjançant l'índex. dada(3) és l'element 3 del vector dada.

Per utilitzar les matrius és important recordar la taula de conversió de tipus de dades:



4.3. Definició de l'entitat: Comptador de programa

Després de crear el banc de registres s'ha de fer un nou projecte per a realitzar el comptador de programa. Aquesta entitat, tal com s'ha indicat tindrà la següent definició:

```
entity ComptadorPrograma is
    Port (  reset :  in STD_LOGIC;
            clk:     in STD_LOGIC;
            Esc:     in STD_LOGIC;
            PCin:  in STD_LOGIC_VECTOR (31 downto 0);
            PCout:out STD_LOGIC_VECTOR (31 downto 0)
    );
end ComptadorPrograma;
```

4.3.1. Guia de disseny del Comptador de programa

El comptador de programa ha de memoritzar el contingut de l'entrada cada vegada que hi hagi un flanc de baixada al senyal de rellotge i l'entrada Esc està activada a 1. El que té memoritzat, és el que s'ha de mostrar a la sortida. Afegim l'entrada de "Reset" per poder inicialitzar el registre a zero.

5. Realització pràctica

5.1. Exercici 1

Realitzeu el disseny corresponent a l'entitat *BancRegistres* i verifiqueu el seu funcionament correcte amb el simulador. Recordeu que cal comprovar que s'emmagatzema correctament els valors assignats a cadascun dels registres, i que aquest valors es pot veure en els dos busos de sortida.

5.2. Exercici 2

Realitzeu el disseny corresponent a l'entitat *ComptadorPrograma* i verifiqueu el seu funcionament correcte amb el simulador. En el comptador de programa només s'ha de comprovar que és realitza l'inici correctament i que es capaç d'enregistrar qualsevol valor que hi hagi a l'entrada.

6. Annexa: Com guardar un projecte en Quartus

El Quartus té una eina que permet arxivar un projecte, empaquetant-lo en un únic arxiu amb l'extensió .qar. L'avantatge de l'ús d'aquesta utilitat, és que automàticament selecciona els arxius realment necessaris. Per arxivar el projecte actual s'ha de procedir de l' següent manera:

1. Seleccionar al menú de Quartus: **Project> Archive Project**. S'obre la finestra de l'Arxiu Project, tal com es mostra a la Figura 1

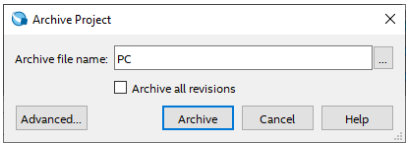


Figura 1. Finestra Archive Project

2. A la finestra de la Figura 1, cal establir un nom pel fitxer en l'apartat: Arxiu file name.

3. Prement el botó d'Advanced ..., s'obre la finestra d'opcions tal com mostra la Figura 2. Aquí es pot personalitzar la selecció de fitxers que s'han d'incloure en el procés d'arxivat de el projecte.

4. Seleccioneu al File set la configuració prèvia **Source control**, tal com es mostra a la Figura 2.

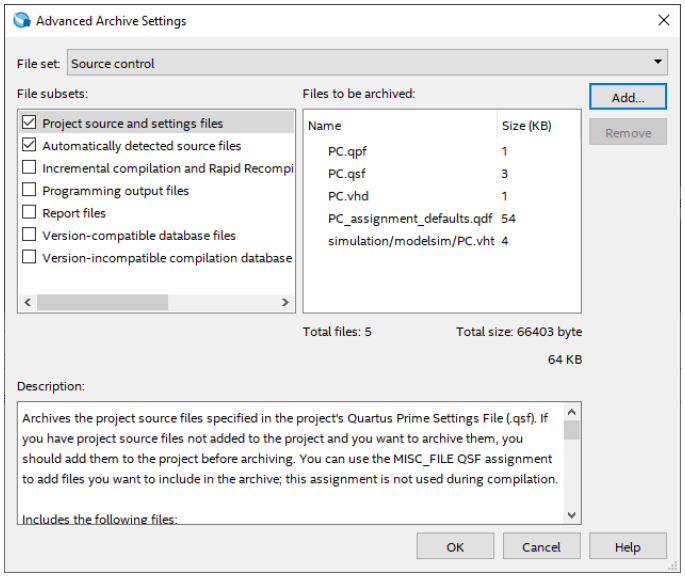


Figura 2. Opcions

5. Prement el botó OK, es torna a la finestra de l'Arxiu Project.

6. Ara, al prémer el botó Archive es crearà el fitxer on s'ha de guardar el projecte, amb el nom el que s'ha especificat en Archive file name i l'extensió .qar. Aquest fitxer es trobarà en el subdirectori del projecte, llevat que s'hagi indicat un altre lloc.

El que heu de presentar (penjar a l'Atenea) son aquests fitxers qar.

En tots els casos, heu de presentar el codi VHDL realitzat per a cadascun dels exercicis, així com les simulacions que demostren el funcionament correcte dels circuits realitzats.

Recordeu que s'ha de demostrar el funcionament correcte de la pràctica el dia assenyalar. En aquests cas el dia que s'ha de presentar la pràctica és el 22 de març.

Les pràctiques s'han de realitzar pels dos integrants del grup i la nota serà per tant la mateixa per a tots dos.

La nota de cada pràctica s'avalua segons els tres punts següents:

- Treball previ (si n'hi ha)
- Funcionament correcte.
- Documentació.

Per poder realitzar la pràctica en el temps previst és necessari que l'alumne realitzi el treball previ, quan així ho estableixi la pràctica. A l'inici de cada sessió de laboratori el professor recollirà aquest treball previ per a la seva avaluació (el que implica que l'alumne ha de tenir una còpia d'aquest treball per utilitzar-lo durant la sessió). Aquesta part valdrà el 20% de la nota.

És imprescindible per aprovar les practiques haver realitzat totes les pràctiques assignades. En el cas de suspendre una pràctica l'alumne tindrà una setmana per fer les correccions pertinents.

La durada de cada pràctica serà d'una sessió tret que s'indiqui el contrari. La memòria de la pràctica es lliurarà a l'Atenea com a màxim el dimarts de la setmana que s'ha presentat. A efectes de càlcul de la nota mitjana final, una setmana de demora significarà que la nota es dividirà per 2, si la demora és més gran la nota serà de zero punts. El fet de superar les dues setmanes de demora no exclou el fet d'haver de lliurar la pràctica, ja que cal lliurar obligatòriament totes les pràctiques.

Algunes de les pràctiques contenen un apartat opcional de caràcter voluntari que donarà l'oportunitat a l'alumne d'augmentar la nota obtinguda en els apartats obligatoris de la pràctica, sempre que aquesta sigui superior a 5.

NORMES GENERALS PER A LA REDACCIÓ DE LA DOCUMENTACIÓ (en el cas de que es demani)

1. La documentació ha de ser concisa i clara.
2. Seguir el guió de la pràctica però sense copiar l'enunciat, excepte de forma esquemàtica.
3. Per les pràctiques en el laboratori d'ordinadors, recordeu que tots els arxius generats s'esborren en finalitzar la sessió. Es recomana crear una carpeta, per exemple a l'escriptori, per emmagatzemar aquests arxius i copiar en finalitzar la sessió a un disc USB. S'obtindrà còpia impresa només dels fitxers font: esquemes i codi VHDL.
4. Si s'inclouen diagrames temporals han de tenir la mateixa escala de temps i estar sincronitzats entre si.