Pràctica 1

Prototipat de matrius sistòliques

1. Objectius

L'objectiu de la present pràctica és introduir el concepte de processador sistòlic. Es pretén que l'alumne conega este tipus d'arquitectura i que implemente una operació de forma paral·lela utilitzant una matriu sistòlica.

Per a dur a terme este objectiu es proposa la realització d'una matriu d'elements de procés que de forma sistòlica multipliquen una matriu quadrada per un vector. Per a això s'utilitzarà la ferramenta de disseny OrCad. El circuit es realitzarà amb la ferramenta de captura d'esquemes i l'element de procés es modelarà amb VHDL. Finalment se simularà tot el circuit per a veure el seu funcionament.

2. Introducció

L'execució paral·lela d'algoritmes és una matèria que afecta quasi tots els camps d'investigació i desenrotllament de la informàtica actual. Algunes de les arquitectures de major rendiment són els processadors matricials, els sistemes multiprocessadors, les matrius de front d'onda (*wave-front arrays*) i les matrius sistòliques (*sistolic arrays*).

Per matrius sistòliques entendrem estructures d'Elements de Procés (EP) amb una disposició cel·lular i amb un flux de dades canalitzat. La transmissió de dades es realitza de forma síncrona entre els diversos EP, al contrari dels *wave-front arrays,* el flux d'informació dels quals és asíncron. Les matrius sistòliques prenen el seu nom del cert paregut que té el flux d'informació entre els diversos EPs de la xarxa cel·lular amb el flux sanguini. Cada colp de rellotge s'assembla a un batec en què la informació fluïx de cel·la a cel·la.

2.1 L'anell sistòlic matriu/vector

L'objectiu és el de realitzar un sistema processador que calcule el producte d'una matriu quadrada A, de nxn elements, per un vector lineal x de n elements.

Per a fer front a este objectiu es realitzarà un vector sistòlic que calcula el vector resultant de multiplicar repetidament una matriu per un vector inicial. Són necessàries n iteracions d'esta matriu sistòlica per a completar la multiplicació de la matriu pel vector, però es pot repetir més vegades per a obtindre una multiplicació recursiva. La solució maquinari és l'anell sistòlic de la figura 1. Esta estructura implementa equacions recursives de la forma:

$$x^{(k)} = A\, x^{(k-1)}$$

Sent x un vector de dimensió n i A una matriu de grandària nxn amb els elements a_{ij} on i és el nombre de fila i j la columna. El superíndex k correspon al número de la iteració, sent $x^{(0)}$ el vector inicial.

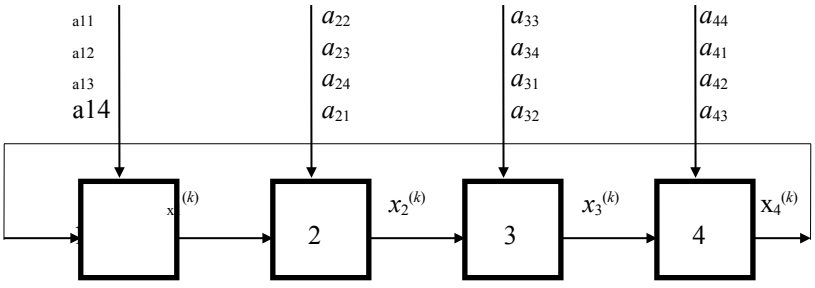

Figura 1: Anell sistòlic de multiplicació recursiva matriu/vector.

3. Realització

Tal com s'ha comentat en la introducció es farà ús en esta pràctica de les ferramentes de disseny d'OrCad per a dur a terme la realització del processador sistòlic proposat. Per a això s'utilitzaran la captura d'esquemes, la descripció VHDL i la ferramenta de simulació. S'expliquen a continuació els passos que s'ha de seguir per a realitzar la descripció del disseny per a la seua posterior simulació.

3.1 Descripció de l'anell sistòlic

L'esquema aproximat de l'anell sistòlic és el que apareix en la figura 1. Està format per 4 elements de procés disposats com un *pipeline*. Els quatre elements de procés realitzen exactament la mateixa funció, és a dir, multipliquen l'entrada de l'esquerra (element *n* del vector) i ho multipliquen per l'element a_{nj} (element *j* de la fila *n*) i guarden el resultat en una suma interna que es va acumulant; quan s'han realitzat 4 iteracions els registres interns de suma contens el resultat de la multiplicació, per la qual cosa han de ser trets a l'exterior.

Esta funcionalitat es descriurà per mitjà del llenguatge VHDL per a crear un símbol que després s'utilitzarà per a la realització de l'esquema de l'anell sistòlic. Una possible descripció en VHDL d'este processador es mostra a continuació:

```
Entity proc is
port ( x,x_ini,a,reset,clk: IN integer;
y: OUT integer);
end proc;

architecture descripcio of proc is
signal suma,compte: integer;
begin
process(clk,reset)
begin
if reset=1 then
compte<=0;
elsif (clk=1 and clk'event) then
if compte<3 then
compte<=compte+1;
else
compte<=0;
end if;
end if;
end process;
```

```
process(clk,reset)
begin
if reset=1 then
y<=x_ini;
suma<=0;
elsif (clk=1 and clk'event) then
if compte<3 then
y<=x;
suma<=suma+x*a;
else
i<=suma+x*a;
suma<=0;
end if;
end if;
end process;

end descripcio;
```

Les entrades i eixides de cada element de procés (anomenat `proc` en esta descripció) vénen definides en el bloc d'entitat (`entity`). L'entrada `x` és el valor de l'element del vector que es pren cada vegada i que ve de l'element de procés anterior. L'entrada `x_ini` és el valor inicial d'eixida de cada EP i conté inicialment el valor del vector a multiplicar. L'entrada `a` correspon a l'element de matriu corresponent en cada iteració. L'entrada `clk` sincronitza per flanc de pujada l'EP i el senyal de `reset` inicialitza l'EP. L'eixida `i` trasllada el valor d'entrada `x` durant el càlcul, i saca el valor de les sumes internes quan el càlcul ha sigut completat.

En principi els senyals de rellotge i reset haurien de ser de tipus bit, però atés que açò donaria problemes a l'hora de la seua implementació en OrCad s'ha optat per definir-los com sencers.

El funcionament intern de l'EP, especificat en el bloc d'arquitectura, és simple: s'han definit dos senyals internes, una servix per a comptar quatre cicles (`compte`) i l'altra per a portar la suma del resultat (`suma`). S'ha separat la descripció en dos processos, un s'ocupa del comptador i l'altre de les eixides; es podria haver fet la descripció en un únic *process*, ja que tenen la mateixa estructura, però separant-ho queda un poc més clar. El *process* del comptador, el primer, posa a zero el compte quan es produïx un reset (`reset=1`) i compte quan es produïx un flanc de pujada del rellotge (`clk=1 and clk'event`). Quan el compte arriba a 3 es posa a zero per a iniciar un nou compte de 4 cicles. El segon procés s'ocupa de la suma i l'eixida. Quan es produïx un reset l'eixida presa el valor de `x_ini`, que és el vector a multiplicar, i la suma es posa a zero. En funcionament normal, cada vegada que es produïx un flanc de pujada del rellotge, l'eixida presa el valor de l'entrada `x` i la suma presa el valor de la suma anterior mes `x*a`; només quan el compte arriba al final (`compte=3`) traiem per l'eixida el valor de la suma total i posem la suma a zero.

Per a introduir esta descripció en OrCad realitzarem els següents passos (convé anar pas a pas amb atenció perquè tot vaja bé, en cas contrari no funcionarà i serà difícil saber per què):

- Arranquem la ferramenta Capture Cis d'OrCad.
- Creguem un nou projecte per mitjà de *File>New>Project*. Al fer açò apareix una finestra on se'ns pregunten diverses coses; per a **començar hem de seleccionar el *Programmable Logic Wizard***, el nom pot ser qualsevol, per exemple, *sistòlic*. Com a família de PLDs podem triar *Other*. En la localització haurem d'especificar el directori on volem que es creu el projecte, resulta per tant convenient haver creat este directori de treball en algun lloc. **Este punt és molt important, si no ho hem fet exactament com s'explica després no funcionarà res.**

- Una vegada creat el projecte s'ha de crear el disseny. Açò es fa per mitjà de *File>New>Design*. Resulta interessant salvar-ho inclús abans de fer res, ja que d'esta manera li podrem posar nom. Per a això fem *File>Save*. És **important especificar el directori de treball que hem definit abans perquè per defecte no ens ho salva ací sinó en qualsevol altre lloc**. A este disseny el podem anomenar *raiz.dsn* encara que qualsevol altre nom val.
- A continuació cal crear el fitxer que contindrà la definició de l'element de procés (si algú se'l va portar ja fet de casa pot passar al punt següent). Per a crear un fitxer vhdl cal fer *File>New>VHDL File* amb el que s'obrirà una finestra de text. Cal introduir ací la descripció del processador tal com s'ha descrit anteriorment. Este fitxer s'ha de salvar com *proc.vhd* per exemple. És **molt important que el nom de fitxer i el nom de l'entitat coincidisquen**. Al final es pot comprovar la sintaxi per mitjà d'Edit>*Check VHDL Syntax. (Alt-C)*.
- Una vegada creada la descripció en VHDL cal incorporar-la a la col·lecció de fitxers del projecte. Per a això obrim la finestreta del projecte i ens situem amb el ratolí damunt de la carpeta crida *Design Resources*, en este moment polsem el botó dret del ratolí i triem *Add File* del menú que ens apareix. Triem el fitxer vhdl que acabem de crear de manera que quede afegit al projecte.
- A continuació cal crear un símbol (*part*) de l'element de procés per a poder usar-ho en l'esquema que fem. Per a això seleccionarem *proc.vhd* en la finestreta del projecte i després farem *Tools>Generate Part* (és **important donar-se compte que els menús canvien depenent del que es tinga seleccionat en cada moment**). Al fer açò apareix una finestra amb unes quantes caselles a omplir. Açò és el que hem d'introduir en cada cas: en el *Netlist file* devem d'especificar el fitxer vhdl que hàgem creat; en el *Vendor* posarem *VHDL Netlist*; com *Part Name* posarem el nom de l'entitat, en este cas *proc*; com *Part lib* podem posar *proc* també; en l'Implementation *type* cal posar *VHDL* i com a nom el de l'entitat (*proc*), finalment en el fitxer de més avall posarem el nostre fitxer vhdl una altra vegada. Açò crearà una biblioteca nova en la carpeta *Outputs*; esta biblioteca es cridarà *proc.olb* (o com li hàgem dit) i contindrà un símbol anomenat *proc*.
- Una vegada es té un símbol per a l'element de procés (proc) ja es pot realitzar l'esquema. Per a això anirem a la finestra on es troba l'esquema i posarem 4 elements de procés. Açò es fa per mitjà de *Place>Part* i triant *proc* dels símbols que apareguen.
- Una vegada col·locats els elements de procés cal posar les entrades i eixides del sistema. Per una costat es tenen les 4 entrades del vector inicial, per un altre es tenen les 4 entrades per a les files de la matriu a multiplicar, i finalment es tenen els senyals de reset i rellotge. Com a única eixida es té l'eixida de l'últim element de procés. Per a posar estes entrades i eixida s'utilitza *Place>Hierarchical Port* i després es tria el port: per als d'eixida **podem utilitzar *PORTLEFT-L*** i per als d'entrada ***PORTRIGHT-R***. A estos ports els haurem de posar nom, per a això es fa *doble clic* damunt de cada port i se li posa el nom corresponent. Els noms han de fer-se únicament amb lletres i números (sempre el nom ha de començar amb una lletra) mai **s'han d'utilitzar espais ni altres caràcters que no siguen lletres o números.**

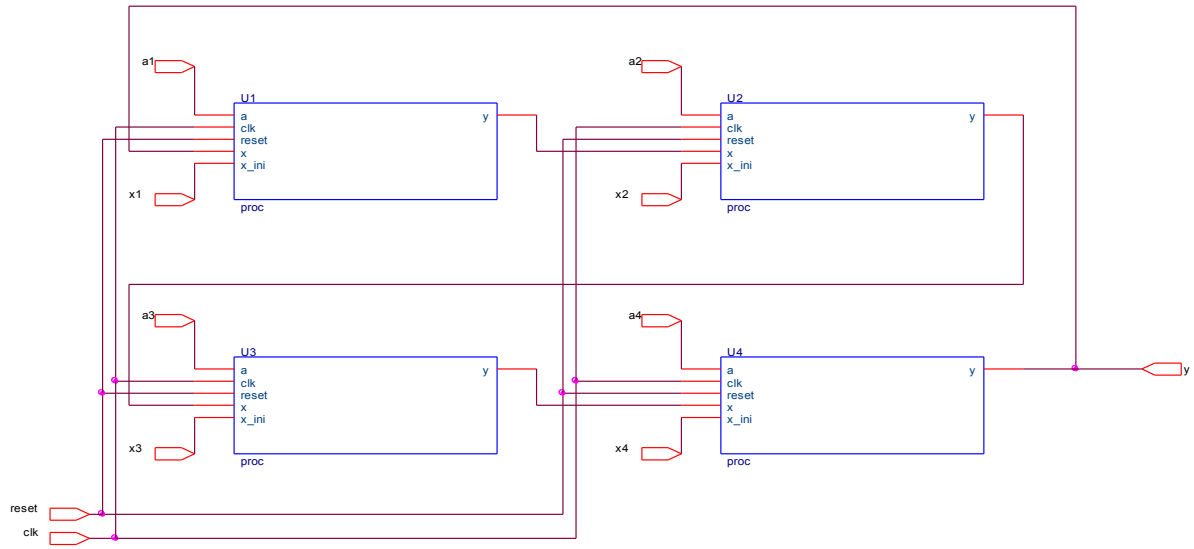

Figura 2: Esquema d'OrCad de l'anell sistòlic mostrant el cablejat, les entrades i l'eixida.

- Quan es tenen els 4 elements de procés i els ports llavors es pot començar a connectar-ho tot per mitjà de *Place>Wire*. En la figura 2 es mostra l'esquema tal com queda després de realitzar totes estes operacions.

- Per a acabar amb el processat de l'esquema cal anotar les referències amb *Tools>Annotate* i comprovar si hi ha algun error amb *Tools>Design Rules Check*. No cal ni dir-ho que el disseny se salva de tant en tant per a no perdre els canvis.
- **Els errors que es puguen produir durant el disseny es troben en la finestra de log**, per la qual cosa resulta interessant tirar-li una ullada de tant en tant.

3.2 Simulació de l'anell sistòlic

Una vegada creat i segó l'esquema haurem de realitzar una simulació del mateix per a veure que funciona. El simulador d'OrCad és una ferramenta diferent de la de captura i la podem invocar des de l'esquema que s'està fent (*Tools>Simulate*) o es pot arrancar independentment com un programa més de windows.

A l'arrancar el simulador es realitza de forma automàtica un *Netlist* que és el que el simulador llegirà per a poder simular el circuit. Este *netlist* és una descripció en VHDL de tot el que s'ha dissenyat fins ara, des de l'element de procés individual fins a tot l'esquema. No obstant això, la generació automàtica d'este *Netlist* falla en OrCad ja que està preparat per a la síntesi de circuits i llavors les connexions d'entrada o eixida només poden ser de tipus bit, i en este disseny són de tipus sencer. Per a solucionar açò i per a simular el circuit correctament cal seguir els següents passos de forma acurada:

- El primer és arrancar la ferramenta de simulació digital. El més còmode és llançar la simulació des de la pròpia ferramenta de captura on es realitza el disseny de l'anell sistòlic. Per a això fem *Tools>Simulate*. En este moment ens diu que hem de triar entre *In Design* o *Timed*, triarem *In Design* que a més és l'opció per defecte.
- Després d'acceptar els diferents missatges que van eixint veurem que l'últim és d'error. Este error es produirà a l'hora de compilar el *netlist* de VHDL que conté la descripció del circuit. Inclús si tot s'ha especificat correctament donarà error degut al que s'ha explicat abans sobre el tipus bit i el sencer. Per a solucionar-ho cal editar a mà el fitxer que genera automàticament al simular i canviar **totes les idees de `std_logic` per `integer` excepte** la de llibreria `std_logic_1164` del principi. Este fitxer ho podem obrir des del propi projecte, ja que es troba en la carpeta *Simulation Resources\InDesig,* el nom del fitxer en esta carpeta serà quelcom com *.\indesign\raiz.vhd* així que li clicamos dos vegades, s'obrirà l'entitat (schematic1) i al clicar dos vegades sobre schematic1 s'obrirà l'editor de text, on caldrà reemplaçar tots els `std_logic` per `integer` i després salvar. També podem obrir directament el fitxer clicando dos vegades sobre l'error. Normalment esta operació es fa només la primera vegada, llevat que es facen modificacions en l'esquema, i en este cas tornarà a repetir-se este error.
- Una vegada realitzat el pas anterior i **cada vegada que es modifique quelcom, s'ha de recarregar novament el projecte per mitjà de Simula't>*Reload Project* en el menú superior de la ferramenta de simulació**. Si tot s'ha especificat correctament llavors no es tindrà cap missatge d'error (veure finestra de log) i podrem començar a simular. Si hi ha algun error significa que es va fer malament la descripció en VHDL dels elements de procés o de l'esquema. Els **errors es poden veure en la finestra de log del simulador**, polsant sobre l'error s'obrirà el fitxer font on es troba l'error. Una vegada detectat l'error cal corregir-ho i tornar a llançar el simulador fins que ja no hi haja més errors.
- Abans de començar la simulació en si cal definir els estímuls o vectors de test. Hi ha diverses formes de fer açò. Una de les més senzilles és **generar automàticament un banc de proves** (*Test Bench*) en VHDL que podem modificar per a posar els senyals d'entrada que vullguem. Este banc de proves es genera amb *Stimulus>Create Test Bench* amb el que ix una finestra on triem l'entitat sobre la qual crear el banc de proves. És **molt important triar l'entitat correcta que en este cas ha de ser *SCHEMATIC.*** Triarem també un nom per al fitxer d'estímuls com per exemple *estimulos.vhd*. Veurem que al fer açò s'ha generat un fitxer VHDL que conté a *proc* com un component i una sèrie de senyals que són les entrades i eixides de *proc* que podem modificar per a donar-li els estímuls a la simulació. En este fitxer generat ens trobem una línia cap al final que posa `-- Place stimulus and analysis statements here` invitant-nos a posar els estímuls a partir d'ací.
- Per tant el següent pas consistix a posar els estímuls en este fitxer generat per OrCad. A continuació es mostren uns estímuls d'exemple que permetran demostrar el funcionament de l'anell sistòlic (es mostren només aquelles parts del fitxer d'estímuls que s'han afegit o modificat per a generar el patró de test):

```
.
.
.
--Place stimulus and analysis statements here

clk<=1,0 after 50 ns when clk=1 else
1 after 50 ns when clk=0;

reset<=1, 0 after 60 ns;

x1<=4;
x2<=3;
x3<=2;
x4<=1;

process
type matriu is array
(1 to 4, 1 to 4) of
integer;

constant a: matriu :=
((4, 3, 2, 1),
(8, 7, 6, 5),
(9,10,11,12),
(13,14,15,16));
begin
wait for 5 ns;
loop
a1<=a(1,4);
a2<=a(2,1);
a3<=a(3,2);
a4<=a(4,3);

wait for 100 ns;
a1<=a(1,3);
a2<=a(2,4);
a3<=a(3,1);
a4<=a(4,2);

wait for 100 ns;
a1<=a(1,2);
a2<=a(2,3);
a3<=a(3,4);
a4<=a(4,1);

wait for 100 ns;
a1<=a(1,1);
a2<=a(2,2);
a3<=a(3,3);
a4<=a(4,4);
wait for 100 ns;
end loop;
end process;
.
.
.
```

Este patró de test multiplica la matriu que ve definida com a en el fitxer pel vector (4,3,2,1) traslladat. Cal observar l'orde en què es van introduint els valors en els elements de procés. Estes instruccions també posen a 1 el reset al principi per a inicialitzar tot. El rellotge simplement va canviant de zero a un amb un període de 100 ns.

· **És molt important no llevar res del que ja estiga en este banc de proves**, és a dir, les línies anteriors s'han d'introduir tal qual, afegint-les al fitxer i sense llevar res del que ja continga.

· Una vegada es té llest el fitxer d'estímuls només queda triar els senyals que es volen veure. Per a això cal fer *Trace>New Wave Window* de manera que podrem triar els senyals per a mostrar. Es poden veure senyals interns inclús de cada element de procés, per a això cal aprofundir en la jerarquia. És aconsellable indicar el tipus de base (*Radix*) de cada senyal i posar-la a decimal (per defecte es mostraran tots els senyals en hexadecimal). Totes estes operacions es poden realitzar en la finestra que s'obri al fer *Trace>New Wave Window*.

· Finalment farem Simula't>*Run* perquè simule durant el temps que se li indique, normalment el que ve per defecte és més que suficient per a veure diverses multiplicacions recursives.

· És important fer Simula't>*Reload>Project* cada vegada que fem qualsevol xicotet canvi i vulguem simular, açò inclou naturalment la primera vegada que es fa *Run*.

· És possible que al fer *Run* ens pregunte què volem simular. Triarem **aquella entitat que tinga *test* en el nom (per exemple, *test_schematic1*).** El que estem fent amb esta operació és definir qual és l'entitat arrel en VHDL. Açò ho podem fer també de forma manual des de la finestra de projecte; l'entitat arrel es distingix per tindre una barra damunt. Per a estar segurs de que simulem el fitxer d'estímuls ens assegurarem que la barra està en l'entitat corresponent a este fitxer. Si volem canviar l'arrel polsarem amb el botó de la dreta del ratolí i triarem *Make root*.

4. Resultats i treball addicional

Cal comprovar que l'anell fa el que toca, és a dir, multiplica correctament la matriu pel vector i després continua multiplicant la mateixa matriu pel resultat i així successivament.

Resulta interessant canviar els vectors de test i comprovar el funcionament amb altres valors per als vectors i les matrius.

En el transcurs de la classe el professor demanarà algun canvi en el disseny.

En la memòria que s'entregue d'esta pràctica caldrà comentar el rendiment d'este tipus de processador comparat amb altres (escalares, matricials, vectorials, etc.). També caldrà mostrar els avantatges i inconvenients d'esta solució en front d'altres, cost del sistema implementat, etc.