Práctica 1

Prototipado de matrices sistólicas (I)

1. Objetivos

El objetivo de la presente práctica es introducir el concepto de procesador sistólico. Se pretende que el alumno conozca este tipo de arquitectura y que implemente una operación de forma paralela utilizando una matriz sistólica.

Para llevar a cabo este objetivo se propone la realización de una matriz de elementos de proceso que de forma sistólica multipliquen una matriz cuadrada por un vector. Para ello se utilizará la herramienta de diseño OrCad. El circuito se realizará con la herramienta de captura de esquemas y el elemento de proceso se modelará con VHDL. Finalmente se simulará todo el circuito para ver su funcionamiento.

2. Introducción

La ejecución paralela de algoritmos es una materia que afecta a casi todos los campos de investigación y desarrollo de la informática actual. Algunas de las arquitecturas de mayor rendimiento son los procesadores matriciales, los sistemas multiprocesadores, las matrices de frente de onda (*wave-front arrays*) y las matrices sistólicas (*sistolic arrays*).

Por matrices sistólicas entenderemos estructuras de Elementos de Proceso (EP) con una disposición celular y con un flujo de datos encauzado. La transmisión de datos se realiza de forma síncrona entre los diversos EP, al contrario de los *wave-front arrays,* cuyo flujo de información es asíncrono. Las matrices sistólicas toman su nombre del cierto parecido que tiene el flujo de información entre los diversos EPs de la red celular con el flujo sanguíneo. Cada golpe de reloj se asemeja a un latido en el que la información fluye de celda a celda.

2.1 El anillo sistólico matriz/vector

El objetivo es el de realizar un sistema procesador que calcule el producto de una matriz cuadrada A, de nxn elementos, por un vector lineal x de n elementos.

Para hacer frente a este objetivo se realizará un vector sistólico que calcula el vector resultante de multiplicar repetidamente una matriz por un vector inicial. Son necesarias n iteraciones de esta matriz sistólica para completar la multiplicación de la matriz por el vector, pero se puede repetir más veces para obtener una multiplicación recursiva. La solución hardware es el anillo sistólico de la figura 1. Esta estructura implementa ecuaciones recursivas de la forma:

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

Siendo x un vector de dimensión n y A una matriz de tamaño n×n con los elementos a_{ij} donde i es el número de fila y j la columna. El superíndice k corresponde al número de la iteración, siendo $x^{(0)}$ el vector inicial.

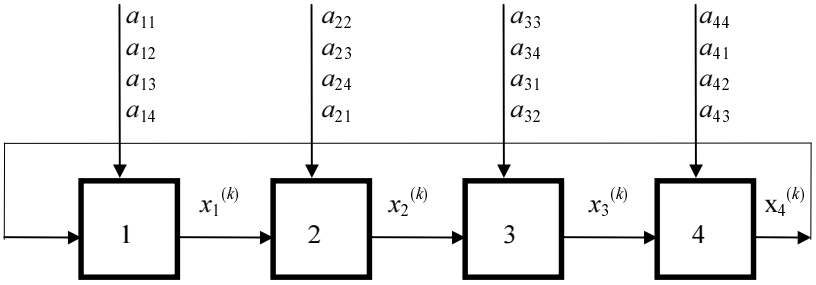

Figura 1: Anillo sistólico de multiplicación recursiva matriz/vector.

3. Realización

Tal y como se ha comentado en la introducción se hará uso en esta práctica de las herramientas de diseño de OrCad para llevar a cabo la realización del procesador sistólico propuesto. Para ello se utilizarán la captura de esquemas, la descripción VHDL y la herramienta de simulación. Se explican a continuación los pasos a seguir para realizar la descripción del diseño para su posterior simulación.

3.1 Descripción del anillo sistólico

El esquema aproximado del anillo sistólico es el que aparece en la figura 1. Está formado por 4 elementos elementos de proceso dispuestos como una cauce segmentado. Los cuatro elementos de proceso realizan exactamente la misma función, es decir, multiplican la entrada de la izquierda (elemento *n* del vector) y lo multiplican por el elemento a_{nj} (elemento *j* de la fila *n*) y guardan el resultado en una suma interna que se va acumulando; cuando se han realizado 4 iteraciones los registros internos de suma contienes el resultado de la multiplicación, por lo que deben ser sacados al exterior.

Esta funcionalidad se describirá mediante el lenguaje VHDL para crear un símbolo que luego se utilizará para la realización del esquema del anillo sistólico. Una posible descripción en VHDL de este procesador se muestra a continuación:

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

architecture descripcion of proc is
signal suma,cuenta: integer;
begin
 process(clk,reset)
 begin
  if reset=1 then
    cuenta<=0;
  elsif (clk=1 and clk'event) then
    if cuenta=3 then
      cuenta<=0;
    else
      cuenta<=cuenta+1;
    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 cuenta=3 then
       y<=suma+x*a;
       suma<=0;
     else
       y<=x;
       suma<=suma+x*a;
     end if;
   end if;
 end process;

end descripcion;
```

Las entradas y salidas de cada elemento de proceso (llamado `proc` en esta descripción) vienen definidas en el bloque de entidad (`entity`). La entrada `x` es el valor del vector que se coge cada vez y que viene del elemento de proceso anterior. La entrada `x_ini` es el valor inicial de salida de cada EP y contiene inicialmente el valor del vector a multiplicar. La entrada `a` corresponde al elemento de matriz correspondiente en cada iteración. La entrada `clk` sincroniza por flanco de subida el EP, y la señal de `reset` inicializa el EP. La salida `y` traslada el valor de entrada `x` durante el cálculo, y saca el valor de las sumas internas cuando el cálculo ha sido completado.

En principio las señales de reloj y reset deberían ser de tipo bit, pero dado que esto daría problemas a la hora de su implementación en OrCad se ha optado por definirlos como enteros.

El funcionamiento interno del EP, especificado en el bloque de arquitectura, es simple: se han definido dos señales internas, una sirve para contar cuatro ciclos (`cuenta`) y la otra para llevar la suma del resultado (`suma`). Se ha separado la descripción en dos procesos, uno se ocupa del contador y el otro de las salidas; se podría haber hecho la descripción en un único process, ya que tienen la misma estructura, pero separándolo queda un poco más claro. El process del contador, el primero, pone a cero la cuenta cuando se produce un reset (`reset=1`) y cuenta cuando se produce un flanco de subida del reloj (`clk=1 and clk'event`). Cuando la cuenta llega a 3 se pone a cero para iniciar una nueva cuenta de 4 ciclos. El segundo proceso se ocupa de la suma y la salida. Cuando se produce un reset la salida toma el valor de `x_ini`, que es el vector a multiplicar, y la suma se pone a cero. En funcionamiento normal, cada vez que se produce un flanco de subida del reloj, la salida toma el valor de la entrada `x` y la suma toma el valor de la suma anterior mas `x*a`; sólo cuando la cuenta llega al final (`cuenta=3`) sacamos por la salida el valor de la suma total y ponemos la suma a cero.

Para introducir esta descripción en OrCad realizaremos los siguientes pasos (conviene ir paso a paso con cuidado para que todo vaya bien, de lo contrario no funcionará y será difícil saber por qué):

- Arrancamos la herramienta Capture Cis de OrCad.
- Creamos un nuevo proyecto mediante *File>New>Project*. Al hacer esto aparece una ventana donde se nos preguntan varias cosas; para empezar debemos seleccionar el *Programmable Logic Wizard*, el nombre puede ser cualquiera, por ejemplo, *sistolico*. Como familia de PLDs podemos elegir *Other*. En la localización deberemos especificar el directorio donde queremos que se cree el proyecto, resulta por tanto conveniente haber creado este directorio de trabajo en algún sitio.
- Una vez creado el proyecto se debe crear el diseño. Esto se hace mediante *File>New>Design*. Resulta interesante salvarlo ya antes de hacer nada, ya que de esta manera le podremos poner nombre. Para ello hacemos *File>Save*. **Es importante especificar el directorio de trabajo que hemos definido antes porque por defecto no nos lo salva ahí sino en cualquier otro lugar**. A este diseño lo podemos nombrar *raiz.dsn* aunque cualquier otro nombre vale.
- A continuación hay que crear el fichero que contendrá la definición del elemento de proceso (si alguien se lo trajo ya hecho de casa puede pasar al punto siguiente). Para crear un fichero vhdl hay que hacer *File>New>VHDL File* con lo que se abrirá una ventana de texto. Hay que introducir aquí la descripción del procesador tal y como se ha descrito anteriormente. Este fichero se puede salvar como *proc.vhd* por ejemplo. Al final se puede comprobar la sintaxis mediante *Edit>Check VHDL Syntax. (Alt-C)*.
- Una vez creada la descripción en VHDL hay que incorporarla a la colección de ficheros del proyecto. Para ello abrimos la ventanita del proyecto y nos situamos con el ratón encima de la carpeta llamada *Design Resources*, en este momento pulsamos el botón derecho del ratón y elegimos *Add File* del menú que nos aparece. Elegimos el fichero vhdl que acabamos de crear de manera que quede añadido al proyecto.
- A continuación hay que crear un símbolo (*part*) del elemento de proceso para poder usarlo en el esquema que hagamos. Para ello seleccionaremos *proc.vhd* en la ventanita del proyecto y luego haremos *Tools>Generate Part* (**es importante darse cuenta que los menús cambian dependiendo de lo que se tenga seleccionado en cada momento**). Al hacer esto aparece una ventana con varias casillas a rellenar. Esto es lo que debemos introducir en cada caso: en el *Netlist file* debemos de especificar el fichero vhdl que hayamos creado; en el *Vendor* pondremos *VHDL Netlist*; como *Part Name* pondremos el nombre de la entidad, en este caso *proc*; como *Part lib* podemos poner *proc* también; en el *Implementation type* hay que poner *VHDL* y como nombre el de la entidad (*proc*), por último en el fichero de más abajo pondremos nuestro fichero vhdl otra vez. Esto creará una biblioteca nueva en la carpeta *Outputs*; esta biblioteca se llamará *proc.olb* (o como le hayamos dicho) y contendrá un símbolo llamado *proc*.
- Una vez se tiene un símbolo para el elemento de proceso (proc) ya se puede realizar el esquema. Para ello iremos a la ventana donde se encuentra el esquema y pondremos 4 elementos de proceso. Esto se hace mediante *Place>Part* y elegiendo *proc* de los símbolos que aparezcan.
- Una vez colocados los elementos de proceso hay que poner las entradas y salidas del sistema. Por una lado se tienen las 4 entradas del vector inicial, por otro se tienen las 4 entradas para las filas de la matriz a multiplicar, y por último se tienen las señales de reset y reloj. Como única salida se tiene la salida del último elemento de proceso. Para poner estas entradas y salida se utiliza *Place>Hierarchical Port* y luego se elige el puerto: para los de salida podemos utilizar *PORTLEFT-L* y para los de entrada *PORTRIGHT-R*. A estos puertos les tendremos que poner nombre, para ello se hace *doble click* encima de cada puerto y se le pone el nombre correspondiente.
- Cuando se tienen los 4 elementos de proceso y los puertos entonces se puede empezar a conectarlo todo. mediante *Place>Wire*. En la figura 2 se muestra el esquema tal y como queda después de realizar todas estas operaciones. Resulta interesante ponerle nombre a los cables que unen unos elementos con otros para luego identificarlos durante la simulación. Los cables unidos a los puertos toman su nombre de estos puertos, por lo que no es necesario renombrarlos, pero los que unen unos elementos de proceso con otros no tienen nombre y conviene ponerle alguno.

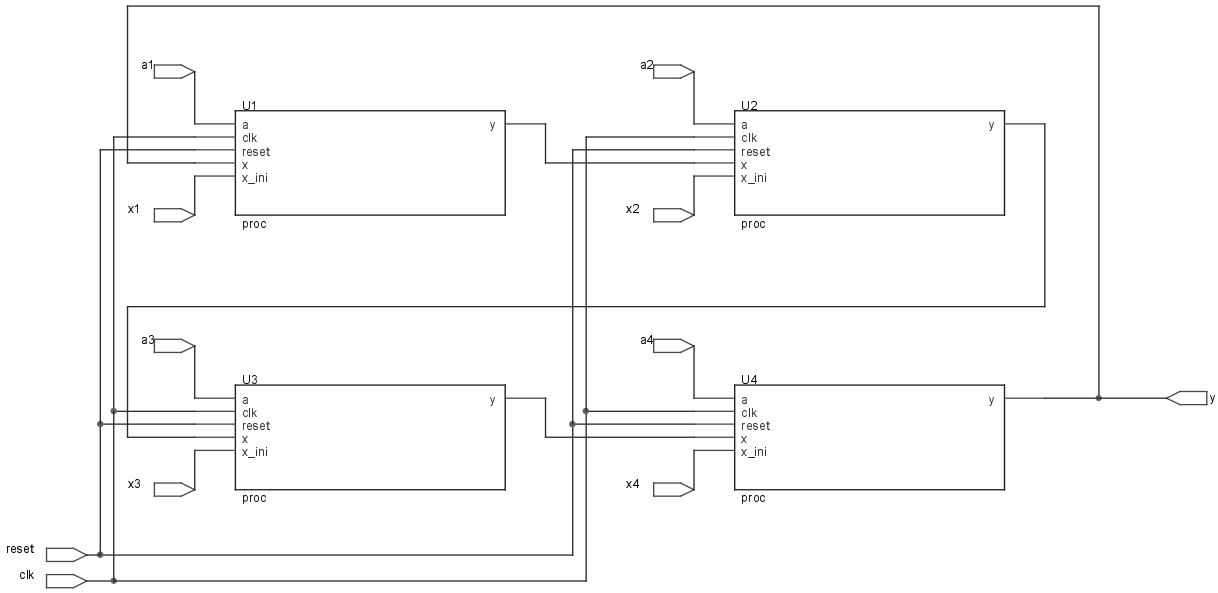

Figura 2: Esquema de OrCad del anillo sistólico mostrando el cableado, las entradas y la salida.

- Para terminar con el procesado del esquema hay que anotar las referencias con *Tools>Annotate* y comprobar si hay algún error con *Tools>Design Rules Check*. Ni que decir tiene que el diseño se salva de vez en cuando para no perder los cambios.

3.2 Simulación del anillo sistólico

Una vez creado y salvado el esquema tendremos que realizar una simulación del mismo para ver que funciona. El simulador de OrCad es una herramienta diferente a la de captura y la podemos invocar desde el esquema que se está haciendo (*Tools>Simulate*) o se puede arrancar independientemente como un programa más de windows.

Al arrancar el simulador se realiza de forma automática un *Netlist* que es el que el simulador leerá para poder simular el circuito. Este *netlist* es una descripción en VHDL de todo lo que se ha diseñado hasta ahora, desde el elemento de proceso individual hasta todo el esquema. Sin embargo, la generación automática de este *Netlist* falla en OrCad puesto que está preparado para la síntesis de circuitos y entonces las conexiones de entrada o salida sólo pueden ser de tipo bit, y en este diseño son de tipo entero. Para solucionar esto y para simular el circuito correctamente hay que seguir los siguientes pasos de forma cuidadosa:

- Lo primero es arrancar la herramienta de simulación digital. Lo más cómodo es lanzar la simulación desde la propia herramienta de captura donde se realiza el diseño del anillo sistólico. Para ello hacemos *Tools>Simulate*.

- Después de aceptar los diferentes mensajes que van saliendo veremos que el último es de error. Este error se producirá a la hora de compilar el *netlist* de VHDL que contiene la descripción del circuito. Incluso si todo se ha especificado correctamente dará error debido a lo que se ha explicado antes sobre el tipo bit y el entero. Para solucionarlo hay que editar a mano el fichero que genera automáticamente al simular y cambiar todas las ocurrencias de `std_logic` por `integer`. Este fichero lo podemos abrir desde el propio proyecto ya que se encuentra en la carpeta *Simulation Resources\InDesig,* el nombre del fichero en esta carpeta será algo como *.\indesign\raiz.vhd* así que le clicacamos dos veces, se abrirá la entidad (schematic1) y al clicar dos veces sobre schematic1 se abrirá el editor de texto, donde habrá que reemplazar todos los `std_logic` por `integer` en todo el fichero y luego salvar. Normalmente esta operación se hace sólo la primera vez, sin importar las veces que se modifica el esquema original, pero si salimos completamente del simulador y volvemos a arrancar es posible que haya que repetir este paso.

- Una vez realizado el paso anterior se debe invocar de nuevo la herramienta de simulación. Si todo se ha especificado correctamente entonces no se tendrá ningún mensaje de error y podremos empezar a simular. Si

hay algún error significa que se hizo mal la descripción en VHDL de los elementos de proceso o del esquema. Los errores se pueden ver en la ventana de log del simulador, pulsando sobre el error se abrirá el fichero fuente donde se encuentra el error. Una vez detectado el error hay que corregirlo y volver a lanzar el simulador hasta que ya no haya más errores.

- Antes de empezar la simulación en sí hay que definir los estímulos o vectores de test. Hay diversas formas de hacer esto. Una de las más sencillas es generar automáticamente un banco de pruebas (*Test Bench*) en VHDL que podemos modificar para poner las señales de entrada que queramos. Este banco de pruebas se genera con *Stimulus>Create Test Bench* con lo que sale una ventana donde elegimos la entidad a simular (*proc*) y el nombre para el fichero de estímulos como por ejemplo *estimulos.vhd*. Veremos que al hacer esto se ha generado un fichero VHDL que contiene a *proc* como un componente y una serie de señales que son las entradas y salidas de *proc* que podemos modificar para darle los estímulos a la simulación. En este fichero generado nos encontramos una línea que pone `-- Place stimulus and analysis statements here` invitándonos a poner los estímulos a partir de ahí.
- Por lo tanto el siguiente paso consiste en poner los estímulos en este fichero generado por OrCad. A continuación se muestran unos estímulos de ejemplo que van a permitir demostrar el funcionamiento del anillo sistólico (se muestran solo aquellas partes del fichero de estímulos que se han añadido o modificado para generar el patrón de test):

```
.
.
.
signal   clk : INTEGER:=1;

begin

-- Place stimulus and analysis statements here

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

reset<=1, 0 after 60 ns;

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

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

 constant a: matriz :=
      (( 1, 2, 3, 4),
       ( 5, 6, 7, 8),
       ( 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 patrón de test multiplica la matriz que viene definida como `a` en el fichero por el vector (1,2,3,4) traspuesto. Hay que observar el orden en que se van introduciendo los valores a los elementos de proceso. Estas instrucciones también ponen a 1 el reset al principio para inicializar todo. El reloj simplemente va cambiando de cero a uno con un periodo de 100 ns.

- Una vez se tiene listo el fichero de estímulos sólo queda elegir las señales que se quieren ver. Para ello hay que hacer *Trace>New Wave Window* de manera que podremos elegir las señales para mostrar. Se pueden ver señales internas incluso de cada elemento de proceso, para ello hay que profundizar en la jerarquía. Es aconsejable elegir indicar el tipo de base (*Radix*) de cada señal y ponerla a decimal (por defecto se mostrarán todas las señales en hexadecimal). Todas estas operaciones se pueden realizar en la ventana que se abre al hacer *Trace>New Wave Window.*
- Una vez realizado todo esto sólo resta por hacer *Simulate>Run* que ejecutará la simulación durante el tiempo que se le indique, normalmente el que viene por defecto es más que suficiente para ver varias multiplicaciones recursivas.

4. Resultados y trabajo adicional

Hay que comprobar que el anillo hace lo que toca, es decir, multiplica correctamente la matriz por el vector y luego sigue multiplicando la misma matriz por el resultado y así sucesivamente.

Resulta interesante cambiar los vectores de test y comprobar el funcionamiento con otros valores para los vectores y las matrices.

Si hay tiempo se puede añadir otro elemento de proceso al esquema para multiplicar vectores y matrices de dimensión 5.

También se puede intentar modificar el símbolo, moviendo los pines y cambiando el tamaño, para que el cableado sea más sencillo.

En la memoria que se entregue de esta práctica habrá que comentar el rendimiento de este tipo de procesador comparado con otros que realicen la misma tarea. También habrá que mostrar las ventajas e inconvenientes de esta solución frente a otras, coste del sistema implementado, etc.

Es importante aprender bien los conceptos de esta primera práctica puesto que para la segunda no se dará la descripción del elemento de proceso, no se dará el esquema final y no se darán los vectores de test. Por lo tanto, conviene ir despacio en esta primera sesión y repasar cada concepto hasta tenerlo claro.