DISEÑO CON HARDWARE RECONFIGURABLE

Curso 2001-2002
Hortensia Mecha López
Dpto Arquitectura de Computadores y Automática
Universidad Complutense de Madrid

Indice

Práctica 1
Diseño de un circuito combinacional mediante esquemáticos. Generación de símbolos y simulación lógica

Objetivo: El objetivo fundamental de esta práctica es que el alumno tome contacto con la herramienta de diseño Xilinx Foundation, especialmente con la entrada de esquemáticos y la simulación funcional.

¿Qué hay que hacer? Diseño, simulación e implementación de un sumador de 4 bits.

Desarrollo:

En primer lugar se diseñará un sumador de un bit, se realizará la simulación lógica y se creará un símbolo asociado. A continuación, utilizando el sumador de un bit, se diseñará un sumador de 4 bits, se simulará, se implementará y se volcará sobre la placa de pruebas.

Ayuda:

1º. Entrada de esquemáticos:

Para generar un nuevo proyecto hay que seguir los siguientes pasos:

- Pulsar en **Foundation Project Manager**
- Seleccionar **New Project**
- En la ventana que aparece hay que seleccionar las FPGAs **XC4000X**L, en particular la **4010XLPC84 con speed 3** y el tipo **Schematic**
- Pulsar **OK**.

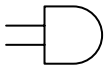
Si la entrada del diseño va a ser mediante esquemáticos hay que seleccionar en el menú:

Tools->Design Entry->Schematic Editor

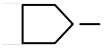
o bien el símbolo de esquemático que aparece en el flujo de diseño 

Para añadir elementos al diseño se selecciona en el menú principal:

Mode->Symbols

o bien el símbolo  del menú de la izquierda.

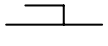
Aparecerá una ventana de símbolos donde se deben seleccionar las puertas y módulos necesarios.

Para añadir los pines de entradas y salidas seleccionar  en la esquina superior izquierda de la ventana de símbolos

Es necesario añadir buffers entre los terminales de E/S y las puertas lógicas para indicar que las señales utilizan puertos de E/S. Los nombres de estos buffers son **IBUF** y **OBUF**. Ojo, esto sólo es necesario en el diseño de más alto nivel, es decir, para esta práctica en el sumador de 4 bits, que es el que tiene los puertos de E/S.

Para añadir interconexiones pulsar en:

Mode->Draw wires

o bien en el símbolo de cable  que aparece en el menú de la izquierda

Una vez terminado el diseño hay que generar la netlist mediante:

Options->Create Netlist
Options-> Integrity Test

Este último paso permite chequear los errores de la netlist.
No hay que olvidarse de salvar el esquemático:

File->Save

A continuación es necesario exportar la netlist para que puedan usarla otras herramientas. Para ello seleccionar

Options->Export Netlist

y en la ventana que aparece seleccionar **Edif200**{*.edn}

Una vez realizados todos los pasos de generación del esquemático, para salir de la ventana pulsar:

File->Exit

Por último es necesario añadir el esquemático al proyecto. Para ello pulsar:

Document->Add.

y seleccionar el fichero correspondiente.

2º. Simulación Funcional:

1. Pulsar el botón de simulación en la ventana de flujo de diseño
2. Añadir las señales que se deseen visualizar mediante
   Signal->Add Signal
   y a continuación seleccionar las señales y pulsar
   Add
   Para cerrar la ventana de selección de señales pulsar **Close**
3. Aplicar los estímulos al circuito. Para ello pulsar en.
   Signal->Add Stimulators
   En este caso se puede seleccionar las distintas salidas del contador binario de 16 bits **BC** (BC0 para la entrada menos significativa, BC1 para la siguiente y así hasta las n entradas).
   Una vez terminado pulsar
   Close
   Se puede seleccionar la frecuencia de dicho contador. Para ello se pulsa en el menú principal
   Options->preferences, y por ejemplo puede seleccionarse 50MHz

4. Simular. Como en este caso estamos realizando una simulación funcional, ya que el diseño no ha sido todavía implementado, tiene que aparecer la opción **functional** seleccionada en la ventana principal. Cada paso de simulación se produce pulsando el botón que se encuentra a la derecha del **ON**.

5. Para salvar la forma de ondas hay que pulsar en:
   File->Save Simulation State
6. Para salir del simulador pulsar en
   File->Exit

3º. Implementación del diseño lógico:

Antes de implementar el diseño, hay que selecionar qué pines de la FPGA queremos que vayan a las distintas E/S del diseño. En este caso, para el sumador de 4 bits, podemos utilizar como entrada los 8 switches que se encuentran en la placa de pruebas, y como salida el banco de leds.
Hay dos formas para asignar a cada E/S un pin:

1. En el propio esquemático, pulsando dos veces en los IBUF o en los OBUF, aparece la ventana de atributos. Añadir un nuevo atributo llamado LOC y asignar el pin correspondiente (por ejemplo p69).
2. En un fichero *.ucf. El formato de las asignaciones es:
   NET nombre_del_pin LOC=pnº;
   Para cada asignación de E/S a pin es necesario colocar una línea.
   Por ejemplo en nuestro caso, donde vamos a utilizar las E/S correspondientes a los switches y a los leds, el fichero *.ucf tendría el siguiente formato:

```
NET data_a<3> LOC=P69;        #switch 8
NET data_a<2> LOC=P66;        #switch 7
NET data_a<1> LOC=P70;        #switch 6
NET data_a<0> LOC=P77;        #switch 5
NET data_b<3> LOC=P6;         #switch 4
NET data_b<2> LOC=P9;         #switch 3
NET data_b<1> LOC=P8;         #switch 2
NET data_b<0> LOC=P7          #switch 1
NET salida<4> LOC=P35;        #led 5
NET salida<3> LOC=P38;        #led 4
NET salida<2> LOC=P39;        #led 3
NET salida<1> LOC=P40;        #led 2
NET salida<0> LOC=P41;        #led 1
```

Hay que tener en cuenta que los leds son activos a baja, es decir, se encienden cuando a su entrada hay un 0 lógico. Los comentarios de la derecha no deben aparecer en dicho fichero.

Para generar el mapa de bits a partir de la netlist pulsar en
Implementation
en la ventana del flujo de diseño. Aparece una ventana, que nos permite seleccionar un fichero de restricciones si es necesario (por ejemplo, en el caso de haber utilizado la segunda opción para asignar pines de E/S del diseño). Si se desea añadir dicho fichero hay que pulsar en:
Set
y en
Use Constraints file from
seleccionar **custom**
y en la nueva ventana entrar el nombre del fichero *.ucf
Una vez dado el fichero de restricciones pulsar en:
OK.

Para implementar hay que pulsar en:

Run

Todos los pasos de la implementación se realizan automáticamente:

Translate: se compila la netlist a un formato propio
Map: se realizan optimizaciones para minimizar el número de puertas
Place&Route: se asignan las puertas a CLBs
Timing: se computan los retardos de los CLBs y los PSMs
Configure: se genera el mapa de bits.

4º. Descarga del diseño en la placa XS40:

Desde una ventana MSDos se utiliza el comando

XSLOAD nombre_fich.bit

Con el comando:

XSPORT b7 b6 b5 b4 b3 b2 b1 b0

se vuelca una palabra de 8 bits en el puerto paralelo. Esto es útil si se han utilizado los pines de la FPGA conectados al puerto paralelo como entradas al diseño.

5º Generación de símbolos:

Una vez generado el esquemático del módulo, para generar un símbolo asociado pulsar en:

Hierarchy->Create Macro Symbol for Current Sheet

En la ventana que aparece se introduce un nombre para el símbolo y se pulsa

OK

Una vez generado el símbolo para el sumador de un bit, abrir un nuevo esquemático (o proyecto si se desea) para realizar el sumador de 4 bits. Al abrir la ventana de símbolos pulsando

Mode->Symbols

aparecerá el nombre del símbolo del sumador de un bit. Este debe utilizarse para realizar el nuevo diseño.

Práctica 2 Diseño de un circuito combinacional utilizando HDL. Generación de macros

Objetivo: El objetivo fundamental de esta práctica es que el alumno tome contacto con la entrada VHDL de la herramienta de diseño Xilinx Foundation, y con la generación de macros.

¿Qué hay que hacer? Diseño, simulación e implementación de un conversor de código binario a 7 segmentos utilizando VHDL. Utilización de este diseño para visualizar la salida del sumador realizada en la práctica anterior.

Desarrollo:

En primer lugar se abrirá un nuevo proyecto VHDL para diseñar un conversor de código binario a 7 segmentos, se realizará la simulación lógica, se implementará y se volcará sobre la placa. La entrada de este diseño pueden ser 4 switches y la salida uno de los 3 displays 7 segmentos.

A continuación se abrirá un nuevo proyecto con entrada de esquemático, que debe ser una copia del sumador de 4 bits diseñado en la práctica anterior. Se generará una macro para el conversor de código, y se utilizará para convertir la salida del sumador de 4 bits a código 7 segmentos. Este diseño debe simularse, implementarse y se volcarse sobre la placa de pruebas. Como entrada pueden utilizarse los switches y como salida 2 de los displays 7 segmentos (el resultado de la suma está entre 0 y 30, luego se necesitan dos dígitos).

Ayuda:

1º. Entrada de un diseño en VHDL:

Para generar un nuevo proyecto con entrada en VHDL, se siguen los mismos pasos que para la entrada de esquemáticos, pero en la primera ventana que aparece hay que seleccionar el tipo **HDL**. Es decir, hay que seguir los siguientes pasos:

- Pulsar en **Foundation Project Manager**
- Seleccionar **New Project**
- En la ventana que aparece hay que dar el nombre y seleccionar el tipo **HDL**
- Pulsar **OK**.

Para abrir el editor de VHDL hay que seleccionar en el menú:

Tools->Design Entry->HDL Editor

o bien el símbolo de **HD**L que aparece en el flujo de diseño 

Aparece una ventana en la cual se pregunta si se desea utilizar el **Design Wizard**. Contestar afirmativamente y seleccionar **VHDL** como lenguaje.

En la nueva ventana se añaden las entradas y salidas del diseño y se pulsa

Finish

Se abrirá automáticamente una nueva ventana del editor de VHDL con el esqueleto del diseño, según las entradas y salidas que hemos colocado en el paso anterior. Añadir las sentencias VHDL necesarias para completar el diseño.

Para chequear la sintaxis hay que pulsar en:

Synthesis->Check Syntax

OK

Cada error sintáctico cometido aparece subrayado en el editor de HDL y una explicación en la ventana inferior.

Existe una herramienta de ayuda sobre la sintaxis de VHDL que puede llamarse mediante:

Tools->Language Assistant

Una vez el diseño tiene una sintaxis correcta hay que salvarlo:

File->Save

File->Exit

Por último es necesario añadir el documento *.vhd al proyecto. Para ello pulsar:

Document->Add.

seleccionar el fichero correspondiente y pulsar

Open

A continuación hay que sintetizar la netlist mediante:

Synthesis->Force Analysis of All HDL Source Files

Synthesis

En la ventana que aparece se coloca como **Top Level** el nombre del diseño. Además hay que seleccionar el dispositivo sobre el cual va a realizarse la implementación:

Target Device – XC4010XL

4010XLPC84

La opción **Insert I/O pads** controla si se deben añadir buffers a todas las señales de E/S. Debe seleccionarse si el circuito que va a sintetizarse se corresponde con el más alto nivel de nuestro diseño. A continuación pulsar:

Run

Los siguientes pasos, la simulación funcional, la implementación y la generación del mapa de bits se realiza según se ha explicado en la práctica anterior.

2º Código VHDL

Para el conversor de código correspondiente a la primera parte de esta práctica el código VHDL es el siguiente:

```
library IEEE;
use IEEE.std_logic_1164.all;

entity conversor_7seg is
   port (
      salida: out STD_LOGIC_VECTOR (6 downto 0);
      data: in STD_LOGIC_VECTOR (3 downto 0)
   );
end conversor_7seg;

architecture conversor_7seg_arch of  conversor_7seg is
  begin
-- codificación de los segmentos
--            0
--          ----
-- 5       |      | 1
--           ---   <- 6
-- 4       |      | 2
--           ---
--            3
   with data select
         salida<="1111001" when "0001",   --1
                          "0100100" when "0010",   --2
                          "0110000" when "0011",   --3
                          "0011001" when "0100",   --4
                          "0010010" when "0101",   --5
                          "0000010" when "0110",   --6
                          "1111000" when "0111",   --7
                          "0000000" when "1000",   --8
                          "0010000" when "1001",   --9
                          "0001000" when "1010",   --A
                          "0000011" when "1011",   --b
                          "1000110" when "1100",   --C
                          "0100001" when "1101",   --d
                          "0000110" when "1110",   --E
                          "0001110" when "1111",   --F
                          "1000000" when others;    --0
end conversor_7seg_arch;
```

3º. Generación de macros:

Para generar una macro a partir de un diseño descrito en VHDL, hay que editar el fichero con el editor HDL y seleccionar

Synthesis->Options

En la nueva ventana seleccionar

Macro

OK

El siguiente paso es crear una netlist para dicho módulo, y así poder incluirlo en un diseño más grande.

Project->Create macro

4º. Utilización de macros en esquemáticos

Para utilizar macros generadas a partir de ficheros VHDL en proyectos basados en esquemáticos, es necesario copiar el fichero *.vhd en el directorio del proyecto, abrirlo con el editor de HDL y seguir los pasos de generación de macros. A continuación, desde el esquemático, en la ventana de símbolos tiene que aparecer el correspondiente a la macro creada. Se utiliza como cualquier otro símbolo.

Además, en la ventana del **Project Manager**, si se pulsa en la pestaña de **Synthesis**, aparecen todas las macros disponibles en el diseño, y pueden actualizarse todas a la vez.

5º. Restricciones

Para volcar la salida sobre los display 7 segmentos es necesario añadir en el fichero *.ucf los pines correspondientes a los dos displays de la placa extendida.

```
display de la izda
        NET display1<0> LOC=P83;
        NET display1<1> LOC=P79;
        NET display1<2> LOC=P4;
        NET display1<3> LOC=P3
        NET display1<4> LOC=P5;
        NET display1<5> LOC=P82;
        NET display1<6> LOC=P78;
display de la drcha
        NET display2<0> LOC=P60;
        NET display2<1> LOC=P50;
        NET display2<2> LOC=P57;
        NET display2<3> LOC=P59;
        NET display2<4> LOC=P51;
        NET display2<5> LOC=P58;
        NET display2<6> LOC=P56;
```

6º. Generación de buses en un esquemático

Para añadir buses en un esquemático se selecciona:

Mode->Draw Buses

o bien se pulsa en la casilla de bus del menú de la izquierda.
Se dibuja el bus pulsando una vez en el punto de comienzo y dos veces en el final. Aparecerá una ventana **Edit Bus** y hay que rellenarla con el nombre y el rango del bus. También hay una opción que permite añadir terminales (en el caso de que el bus vaya a pines de la FPGA). A continuación pulsar:

OK.

Para conectar un bus a la entrada de un módulo de 1 bit se selecciona:

Mode->Draw Bus Taps

o bien se pulsa en la casilla correspondiente del menú de la izquierda. Se dibuja la conexión pinchando en el bus y a continuación en el módulo al que hay que conectarla. No olvidar poner nombre a las conexiones.

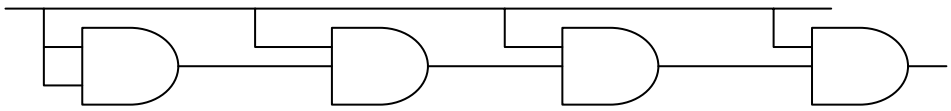
Práctica 3. Simulación eléctrica

Objetivo: El objetivo fundamental de esta práctica es que el alumno comprenda la importancia de considerar el comportamiento eléctrico de los circuitos, por los problemas de retardos, carreras, gliches, etc.

¿Qué hay que hacer? Simulación eléctrica de algunos diseños

Desarrollo:

a.- En primer lugar se realizará una simulación eléctrica antes de realizar la implementación del circuito. Para este paso se utilizará el circuito siguiente:



Colocar cuatro salidas, una después de cada puerta and. La simulación será del tipo Unit.

b.- En segundo lugar, diseñar un contador módulo 8 con biestables D e implementarlo. A continuación realizar una simulación eléctrica. Con el Analizador de tiempos visualizar la distribución de los módulos e interconexiones dentro de la FPGA, calcular el retardo de todos los caminos y calcular la frecuencia máxima del circuito.

Ayuda:

Simulación Tipo Unit

En el simulador, en lugar de seleccionar la simulación **Functional**, seleccionar **Unit**. Esta opción permite simular considerando el mismo retardo para todas las puertas, lo que se llama retardo unidad. Al realizar la simulación puede observarse que no todas las puertas cambian simultáneamente, sino que en cada unidad de tiempo sólo una de ellas cambia. Para seleccionar el retardo unidad debe pulsarse en

Options->Preferences

y colocar 1ns en la casilla de Precisión.

Simulación eléctrica

Para hacer una simulación eléctrica que tenga en cuenta los retardos reales de las puertas e interconexiones del circuito, es necesario realizar la implementación, y automáticamente la herramienta genera información detallada sobre dichos retardos.

Una vez realizada la implementación, se realiza la simulación seleccionando el modo **Timing** en lugar de **Functional,** o bien pulsando el botón de **verification** del flujo de diseño. Comparar los resultados obtenidos con la simulación funcional. Calcular a la frecuencia máxima a la que puede trabajar el circuito.

Para visualizar la distribución de los distintos módulos e interconexiones del circuito pulsar en

Tools->Implementation->FPGA Editor

Seleccionando un punto inicial y otro final puede obtenerse el retardo de los distintos caminos del diseño. De esta forma puede calcularse el máximo retardo, y la frecuencia a la que puede trabajar el circuito.

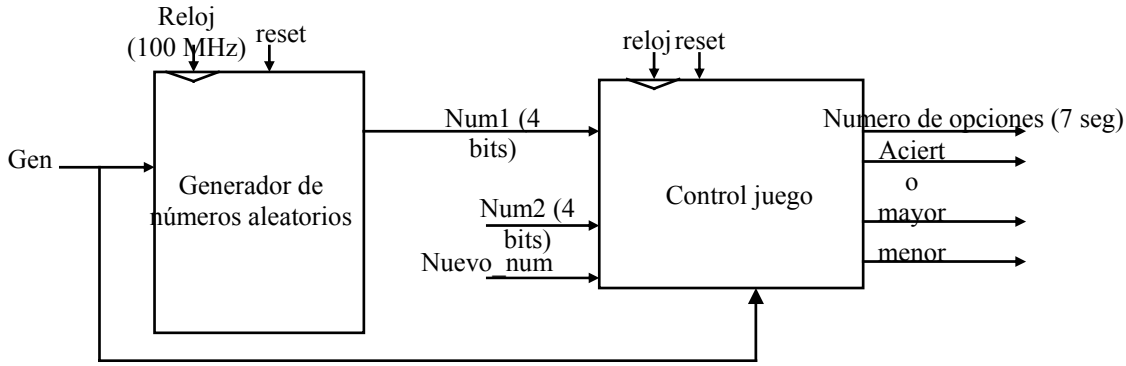
Práctica 4. . Diseño de un circuito secuencial mediante máquinas de estado

Objetivo: El objetivo fundamental de esta práctica es que el alumno aprenda a diseñar máquinas de estado ayudándose de la herramienta disponible en Xilinx Foundation

¿Qué hay que hacer? Diseñar un juego para adivinar números entre 0 y 15

Desarrollo:

El circuito debe tener dos máquinas de estado, según el diseño que se presenta a continuación:



- Un generador de números aleatorios. Este módulo será un contador mod. 16 que trabaja a 100 MHz y dispone de una entrada Gen (1 switch), que al activarse permite al contador comenzar la cuenta, y al desactivarse para la cuenta. La entrada Gen se conectará a uno de los switches de la placa base. Dada la alta frecuencia del circuito, el jugador no podrá controlar la cuenta, y el contador se comportará como un generador de números aleatorio.
- Un módulo que controla el juego. El sistema tiene las siguientes entradas:
    1. **Num1**: número que hay que adivinar
    2. **Num2**: número que introduce el usuario (de los switches)
    3. **Nuevo_num**: a 1 indica que se ha introducido un nuevo número (1 switch)
    4. **Gen**: cuando se pone a 1 se reinicializa el juego, y a 0 empieza a funcionar

En el juego hay tres oportunidades de acierto. Por la salida "**Número de opciones**" se visualiza en uno de los displays las opciones que quedan (inicialmente 3 y cada vez que se introduce una nueva y se falla se decrementa en uno). Cada vez que se introduce un nuevo número se encenderá el led correspondiente a la salida mayor si Num1>Num2, el led correspondiente a menor si Num1<Num2 y el led correspondiente a acierto si Num1=Num2. Una vez gastadas las 3 opciones, si no se ha adivinado el número, se pasa a un estado de espera hasta que se activa Gen.

Ayuda:

Utilización del editor de máquinas de estados:

Para utilizar el editor de máquinas de estados, desde el menú principal pulsar en:

Tools->Design Entry -> State Editor

o bien en el icono correspondiente a una máquina de estados de la ventana del Project Manager

Para añadir estados pulsar en

FSM->State

o bien en el icono 

Para dar nombre a los estados, se pincha sobre ellos y se pulsa dos veces.

Para añadir transiciones entre estados:

FSM->Transition

o bien pulsar en el icono 

Primero llevar el ratón al estado origen, pinchar, y luego ir al estado destino y pinchar también.

Para añadir las condiciones para que se produzca una transición pulsar en

FSM->Condition

o bien en el icono 

A continuación se pincha en la transición donde se va a añadir la expresión. El lenguaje que debe usarse es VHDL.

Para añadir la señal de reset

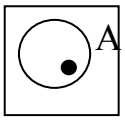
FSM->Reset

o bien en el botón 

Debe añadirse a la transición una expresión booleana que si se cumple hará a la máquina entrar en el estado de Reset. Si se pulsa con el botón derecho en el símbolo del Reset, puede seleccionarse si el reset es síncrono o asíncrono.

Para dar las salidas de un estado pulsar en:

FSM->Action->State

o bien en el símbolo: 

y pulsar en el estado donde desee añadirse.

Una vez terminado el diagrama de estados hay que salvarlo:

File->Save

y generar el código VHDL

Synthesis->HDL Code Generation

Synthesis->Synthesize

Por último, en el circuito global hay que añadir el fichero *.vhd mediante

Document->Add

Práctica 5. . Diseño de un circuito secuencial en VHDL

Objetivo: En esta práctica se pretende que el alumno diseñe un reloj despertador usando el lenguaje VHDL y aprenda a utilizar las memorias internas de la FPGA como elementos de almacenamiento.

¿Qué hay que hacer? Diseñar un reloj despertador programable. Los datos de la hora real y la alarma del reloj se almacenarán en la memoria interna de la FPGA.

Desarrollo:
Dado que la salida del reloj va a volcarse sobre los bancos de displays, lo adaptaremos de forma que cada minuto conste de 16 segundos, cada hora conste de 16 minutos y un dia tenga 16 horas. El intefaz es el siguiente:

-Una entrada de programación **prog** (1 switch). Sirve para poner el reloj en hora

-Una entrada de **alarm** (1 switch) que sirve para programar la alarma

-Una entrada de **on/off** que permite conectar y desconectar la alarma

-Una entrada **in** (1 switch) que sirve para incrementar la hora

-Tres salidas de 7 bits cada una, que irán a los 3 displays 7 segmentos (**display1**[6:0], **display2**[6:0] y **display3**[6:0]). En display1 se visualiza la hora (sólo puede contar de 0 a 15 horas), en display2 los minutos (de 0 a 15) y en display3 los segundos (de 0 a 15).

-Una salida en uno de los leds, que al apagarse y encenderse alternativamente indicará ha saltado la alarma.

El reloj tiene 4 modos de funcionamiento: normal, programación de hora, programación alarma y alarma.

En modo normal de funcionamiento, el reloj va incrementando los segundos, minutos y horas como un reloj digital normal, salvo el cambio de base (es hexadecimal).

Si se pulsa prog, el reloj entra en modo de programación de hora. En primer lugar parpadearán las horas, y con la entrada in se irá incrementando la hora hasta que el usuario desee. Una vez colocada la hora se pulsa prog, con lo cual se almacena la nueva hora y empiezan a parpadear los minutos. Nuevamente con la entrada in el usuario incrementa hasta llegar al valor adecuado y pulsa prog. Así se almacenan los minutos y se pasa a programar los segundos de la misma forma. Una vez pulsado prog por cuarta vez, se pasa a modo normal.

Si se pulsa alarm, se visualiza en los displays la hora de alarma programada, con el dígito de las horas parpadeante. Con la entrada in se incrementa hasta que el usuario pulse alarm, entonces se pasa a programar los minutos (estos parpadean) y después los segundos, de forma similar a la programación de la hora. Una vez pulsado alarm por 4ª vez se pasa a modo normal.

Cuando la hora del reloj coincida con la de la alarma y si el botón de on/off se encuentra en la posición on , se pasa a modo alarma. En este caso, con una frecuencia de un segundo, se encenderá y apagará uno de los leds hasta que se pulse el botón de on/off se coloque en posición de off.

Ayuda

Utilización de memorias

Dentro de la FPGA, se pueden utilizar las LUTs para generar memorias RAM. Cada CLB tiene 2 LUTs de 4 entradas, es decir, 32 bits de datos. Estas LUT pueden organizarse como RAM de 16*2, 32*1 o duales de 16*1.
Por ejemplo, la RAM de 16*4 construida a partir de LUTs tiene el siguiente interfaz:

```
Component ram_16x4
PORT
(A0,A1,A2,A3: IN STD_LOGIC;  líneas de dirección
 D0,D1,D2,D3: : IN STD_LOGIC; líneas de entrada de datos
we: : IN STD_LOGIC;                señal de escritura (activa en alta)
O0, O1, O2, O3: : IN STD_LOGIC;          líneas de salida de datos
); END COMPONENT
```

Este y otros módulos diseñados a partir de LUTs pueden utilizarse directamente en esquemáticos (aparece su nombre en la lista de símbolos) y en los ficheros VHDL. Su utilidad fundamental es para reducir al máximo el número de CLBs necesarios para implementar un diseño, ya que si en lugar de las LUT se usan los flip-flop existentes en los CLBs, para generar un registro de 32 bits se necesitan 16 CLBs (en lugar de 1 como es el caso).
En el diseño del reloj con alarma vamos a utilizar una memoria de n*4 para almacenar la hora real y la hora programada para que salte la alarma (en total se necesitan 6 palabras de 4 bits). La elección del tamaño del módulo de memoria se deja al alumno.