DISEÑO AVANZADO DE SISTEMAS DIGITALES Y MICROPROCESADORES
3º INGENIERÍA INFORMÁTICA

PRÁCTICA 2
CONTROL DEL DISPLAY DE SIETE SEGMENTOS

➢ OBJETIVOS

- Diseñar e implementar una serie de contadores síncronos que permitirán controlar los cuatro displays de siete segmentos disponibles en la tarjeta Digilent S3
- Aprender las opciones básicas de síntesis e implementación del entorno Xilinx ISE 14.7
- Aprender a utilizar la hoja de características de la tarjeta S3 para asignar correctamente los puertos de la FPGA en función de la aplicación

➢ TARJETA S3 DE DIGILENT

La tarjeta Digilent S3 de Xilinx consiste básicamente en una FPGA (Xilinx Spartan3 con 200K puertas equivalentes) cuyos puertos están unidos directamente a una serie de periféricos. En concreto esta placa contiene displays 7-segmentos, diodos LED, pulsadores y conmutadores, que nos permiten visualizar las salidas y generar los estímulos de entrada. En la figura 1 se muestra la tarjeta que utilizaremos en la práctica junto con su diagrama de bloques básicos.

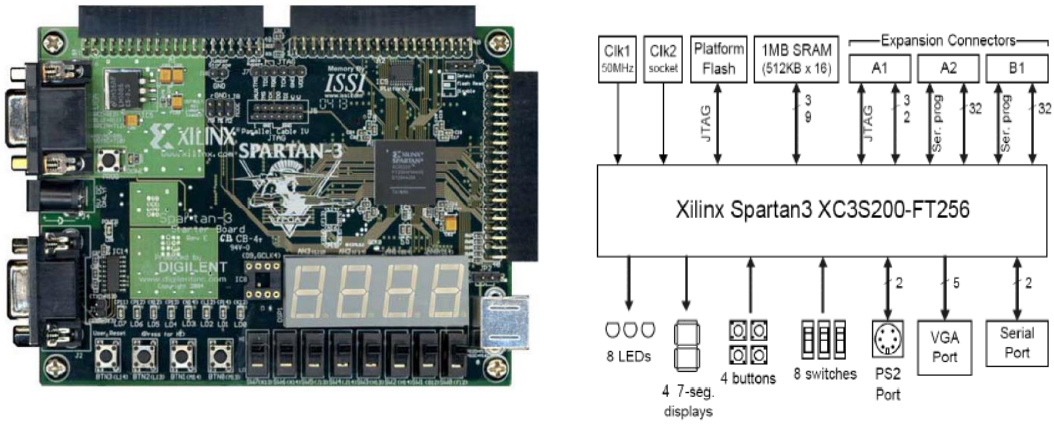

Figura 1: Tarjeta Digilent S3 de Xilinx

Los cuatros displays de 7 segmentos están situados en la parte inferior derecha de la tarjeta. Su control se puede entender visualizando la figura 2. Se utilizarán las siguientes señales (conectadas a puertos de las FPGAs):

- A, B, C, D, E, F, G y DP: señales comunes para los cuatro dígitos activas a nivel bajo. Cuando una de estas señales se activa (fijando un '0' desde la FPGA) se encenderá el segmento correspondiente.
- AN0, AN1, AN2 y AN3: Señal de habilitación (activa a nivel bajo). Tendremos una habilitación independiente para cada dígito, de forma que sólo se encenderá aquel que esté habilitado.

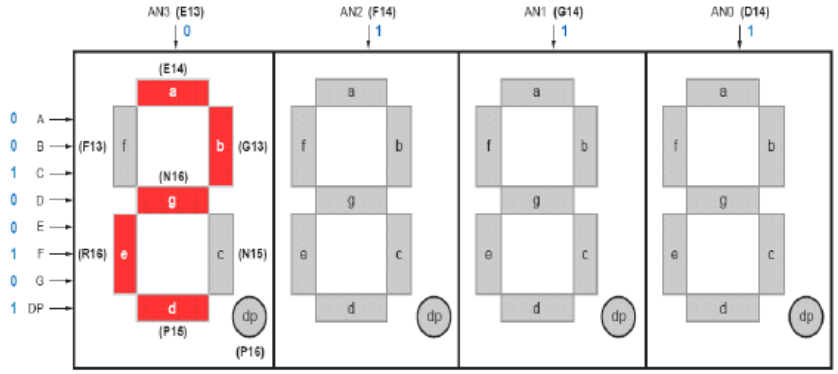

Figura 2: Control del display de 7 segmentos

Por ejemplo, en la figura 2 se ilustra los estímulos (señalados en azul) que habría que proporcionar a los displays para conseguir que aparezca un 2 en el primer dígito.

- De lo anterior es muy sencillo deducir los códigos necesarios para la representación de caracteres en el display, los cuales son mostrados en la tabla 1.

Character	a	b	c	d	e	f	g
0	0	0	0	0	0	0	1
1	1	0	0	1	1	1	1
2	0	0	1	0	0	1	0
3	0	0	0	0	1	1	0
4	1	0	0	1	1	0	0
5	0	1	0	0	1	0	0
6	0	1	0	0	0	0	0
7	0	0	0	1	1	1	1
8	0	0	0	0	0	0	0
9	0	0	0	0	1	0	0
A	0	0	0	1	0	0	0
b	1	1	0	0	0	0	0
C	0	1	1	0	0	0	1
d	1	0	0	0	0	1	0
E	0	1	1	0	0	0	0
F	0	1	1	1	0	0	0

Tabla 1: Códigos necesarios para la representación de un carácter en el display

Como hemos comentado anteriormente, los 8 bits que encienden cada uno de los segmentos son comunes para los cuatro displays. Por tanto, esto nos conduciría a pensar que los cuatros displays no pueden ser encendidos simultáneamente. Sin embargo, si observamos la figura 3 podemos comprobar que las señales de habilitación de los displays (AN0-3) pueden servir para multiplexar en el tiempo las señales de activación de los segmentos (A, B, …, DP), de forma que estaríamos sucesivamente encendiendo un display y apagando todos los demás. Si esto es realizado a una frecuencia alta, debido a la persistencia de la visión, el cerebro humano percibe que todos los caracteres se encienden simultáneamente.

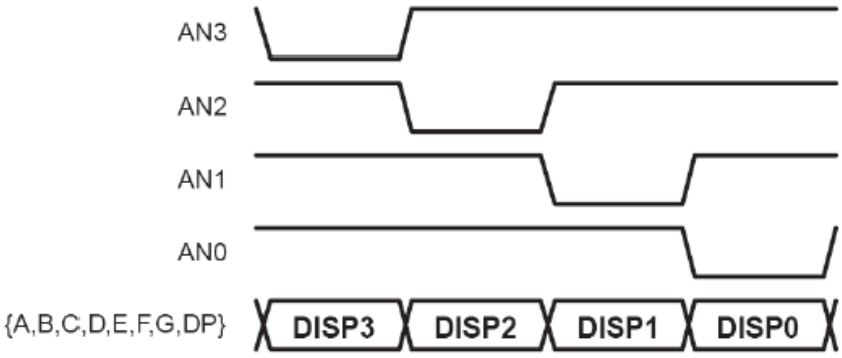

Figura 3: Uso de los 4 dígitos de forma simultánea

La tarjeta Digilent S3 dispone de un reloj de 50 MHz que es el que utilizaremos como entrada de reloj en nuestros diseños digitales. Para introducir el reset asíncrono a nuestro diseño utilizaremos uno de los cuatro pulsadores de la parte inferior izquierda de la tarjeta. Al pulsar un botón producimos un nivel lógico '1' en el puerto correspondiente de la FPGA.

Un aspecto muy importante a la hora de implementar un diseño es la selección de puertos en la FPGA. En función de los periféricos o conectores que vayamos a utilizar en nuestro diseño usaremos unos puertos concretos de la FPGA, que deberán ser especificados al realizar la implementación. Para conocer los puertos necesitamos conocer la conexión entre la FPGA en la tarjeta S3. En la tabla 2 se especifican los puertos asociados a los displays de 7 segmentos, reloj, pulsadores, switches y diodos led. El diseño final de la práctica, una vez implementado en la FPGA, debe controlar los pines de estos periféricos a través de estos puertos.

Segment	FPGA Pin
A	E14
B	G13
C	N15
D	P15
E	R16
F	F13
G	N16
DP	P16

Anode Control	AN3	AN2	AN1	AN0
FPGA Pin	E13	F14	G14	D14

Push Button	BTN3 (User Reset)	BTN2	BTN1	BTN0
FPGA Pin	L14	L13	M14	M13

Diodos Leds	LD7	LD6	LD5	LD4	LD3	LD2	LD1	LD0
FPGA Pin	P11	P12	N12	P13	N14	P14	P14	K12

Switches	SW7	SW6	SW5	SW4	SW3	SW2	SW1	SW0
FPGA Pin	K13	K14	J13	J14	H13	H14	G12	F12

Reloj	CLK
FPGA Pin	T9

Tabla 2: Pines de la FPGA asociados a distintos periféricos

➢ REALIZACIÓN DE LA PARTE 1 DE LA PRÁCTICA

Ha de diseñarse en código VHDL un sistema digital cuyo objetivo sea convertir el display en un contador que se incremente a una velocidad “visible”. En la figura 4 se muestra un esquema general de esta parte de la práctica.

Para ello necesitaremos diseñar los siguientes bloques:

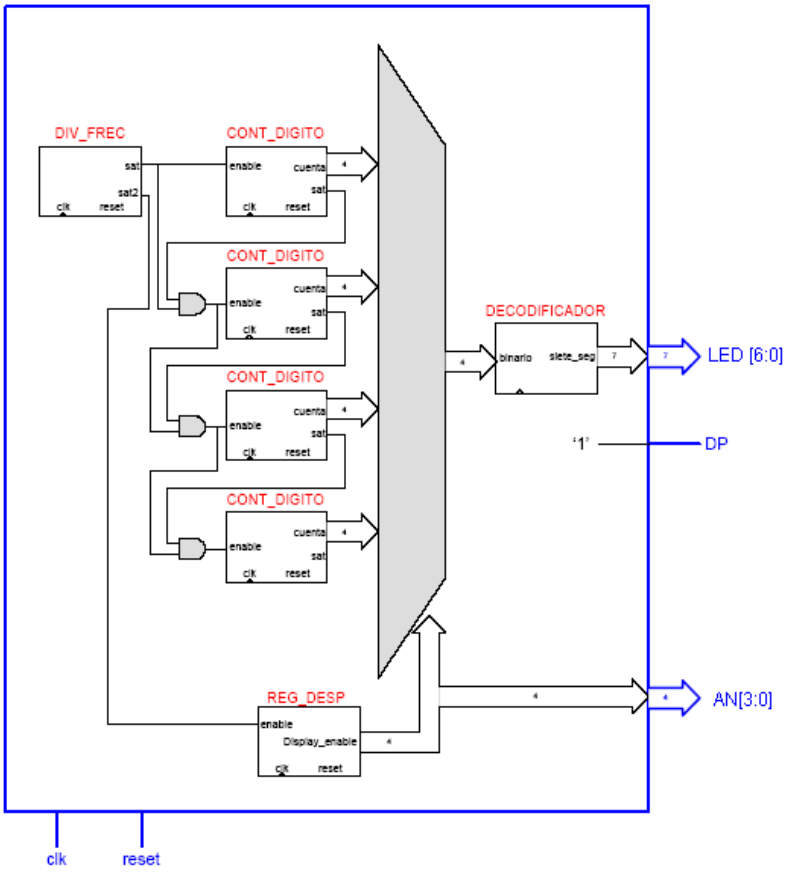

Figura 4: Esquema general de la parte 1 de la práctica

1.- Divisor de frecuencia. Consiste en un contador de 26 bits que divide la frecuencia de reloj de 50 MHz a una más visible por el ojo humano. Produce:

- Un pulso a la salida cada vez que cuenta=”11111111111111111111111111”, dividiendo la frecuencia de reloj por 2^{26} para obtener una señal a una frecuencia de 0.74 Hz.

- Un pulso a la salida cada vez que cuenta(7 downto 0)=”1111111”, dividiendo la frecuencia de reloj por 2^{8} para obtener una señal a una frecuencia de 195 KHz, que será utilizada para la conmutación entre los displays.

```
entity div_frec is
port (clk: in std_logic; -- Reloj
reset: in std_logic; -- Reset asíncrono activo a nivel alto
sat: out std:logic; -- se produce un pulso de duración un ciclo de reloj cuando se llega al final
                       de la cuenta (cuenta="11111111111111111111111111")
sat2: out std_logic); --  se produce un pulso de duración un ciclo de reloj cuando los ocho bits
                          menos significativos de cuenta llegan a "11111111"
end div_frec;
```

2.- Contador dígito. Consiste en un contador entre 0 y 9 con reset asíncrono y señal de habilitación. Cuando está activa la habilitación se contará de 0 a 9, volviendo a comenzar de nuevo por 0. Asignaremos un contador de este tipo a cada dígito del display.

```
entity cont_digito is
port (clk: in std_logic; -- Reloj
reset: in std_logic; -- Reset asíncrono activo a nivel alto
enable: in std_logic; -- Señal de habilitación activa a nivel alto (si enable='1' el contador
                         avanzará uno en la cuenta en el flanco positivo del reloj)
cuenta: out std_logic_vector (3 downto 0); -- salida de 4 bits que comienza en "0000" y
                                              termina en "1001", volviendo a empezar de
                                              nuevo
sat: out std_logic); --  si cuenta es igual a "1001", sat será activada (activa a nivel alto)
end cont_digito;
```

3.- Decodificador. Consiste en un decodificador de binario a código de siete segmentos.

```
entity decodificador is
port (binario: in std_logic_vector (3 downto 0); -- código binario sin signo
led: out std_logic_vector (6 downto 0)); -- Código de 7 segmentos según la tabla 1. Se
                                            corresponde, de más a menos significativo, con
                                            las entradas A, B, C, D, E, F y G del display
end decodificador;
```

4.- Registro de de desplazamiento. Consiste en un pequeño registro de desplazamiento circular, de forma que todos los bits se desplazarán una posición a la izquierda (hacia el bit más significativo), y el bit menos significativo pasará a tener el valor del bit más significativo. El valor del registro al producirse un reset asíncrono será "1110", por tanto, la secuencia que se producirá será: "1110" -> "1101" -> "1011" -> "0111" -> "1110" (ver figura 3). Estos desplazamientos se producirán siempre que el módulo esté habilitado. El valor del registro se utilizará para controlar las señales de habilitación de los displays.

```
entity reg_desp is
port (clk: in std_logic; -- Reloj
reset: in std_logic; -- Reset asíncrono activo a nivel alto
enable: in std_logic; -- Señal de habilitación activa a nivel alto
display_enable: out std_logic_vector (3 downto 0)); -- valor del registro de desplazamiento
                                                       (con enable ='1', se producirá un
                                                       desplazamiento cada flanco posiitivo del
                                                       reloj)
end reg_desp;
```

5.- Bloque de jerarquía superior. Una vez diseñados y simulados los bloques anteriores, pasaremos a conectarlos tal y como se muestra en la figura 4. Los bloques sombreados de la

figura 4 (multiplexor y puertas AND) se implementarán directamente, mediante órdenes concurrentes, en el nivel de jerarquía superior.

```
entity control_display is
port (clk: in std_logic; -- Reloj
reset: in std_logic; -- Reset asíncrono activo a nivel alto
led: in std_logic_vector (6 downto 0); -- Salidas activas a nivel bajo que encienden los leds
                                          correspondientes del display (ver figura 2)
dp: out std_logic; -- salida activa  a nivel bajo que enciende el punto del display (siempre
                      estará desactivada)
an: out std_logic_vector (3 downto 0)); -- habilitación de los diferentes dígitos del display. Se
                      formará una secuencia como se indica en la figura 3
end control_display;
```

Una vez terminado el diseño en VHDL del esquema de la figura 4, se procederá a la síntesis e implementación del mismo haciendo uso del entorno ISE 14.7 de Xilinx. Para la etapa de implementación será necesario utilizar el fichero de restricciones control_display.ucf.

➢ REALIZACIÓN DE LA PARTE 2 DE LA PRÁCTICA

Modificar el diseño de la parte anterior para que se ajuste al esquema de bloques que se muestra en la figura 5, consistente en un contador de 4 bits síncrono con puesta a cero asíncrona (*reset*), que dispone de señales de control de cuenta hacia arriba (*up*) y cuenta hacia abajo (*dw*).

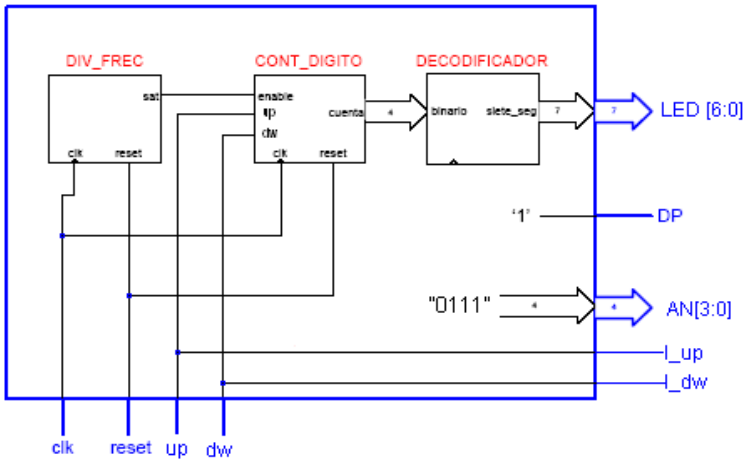

Figura 5: Esquema general de la parte 2 de la práctica

Modificar el fichero de restricciones control_display.ucf haciendo uso de la tabla 2, de manera que para generar las señales de control *up* y *dw* se empleen los conmutadores *switch1* y *switch0* respectivamente. También usaremos los diodos *LED1* y *LED0* para encenderlos cuando su correspondiente señal de control esté activa (*l_up* y *l_dw*).