

CIRCUITOS ELECTRÓNICOS DIGITALES (CED-ISW)

Práctica 6: Descripción de Subsistemas combinacionales con Verilog e implementación en FPGA)

Objetivos de la práctica

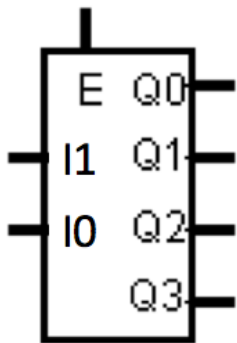
- Introducción al entorno de diseño lógico Xilinx ISE webpack 12.4, con sus funciones básicas
- Especificación de subsistemas combinacionales básicos en Verilog
- Simulación lógica de subsistemas combinacionales
- Implementación en un dispositivo programable FPGA

Material:

- Ordenador con Xilinx ISE Design Suite 12.4 instalado y placa de entrenamiento de FPGA Basys2

1. ESTUDIO TEÓRICO (Debe presentarse antes de empezar la práctica)

- a) Complete la siguiente tabla de descripción funcional de un DEC2:4 con salidas activas en alto y ENABLE también en alto, y obtenga las ecuaciones de salida Q_i :



E	I1	I0	Q0	Q1	Q2	Q3
0	X	X				
1	0	0				
1	0	1				
1	1	0				
1	1	1				

Funciones de salida:

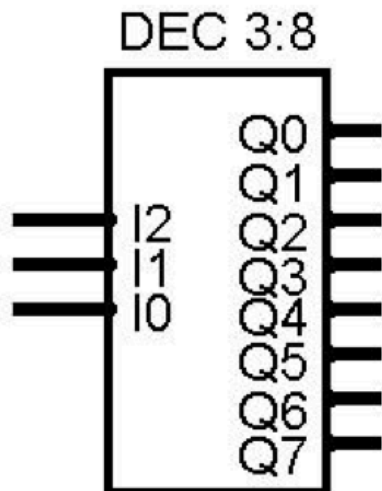
Q0=

Q1=

Q2=

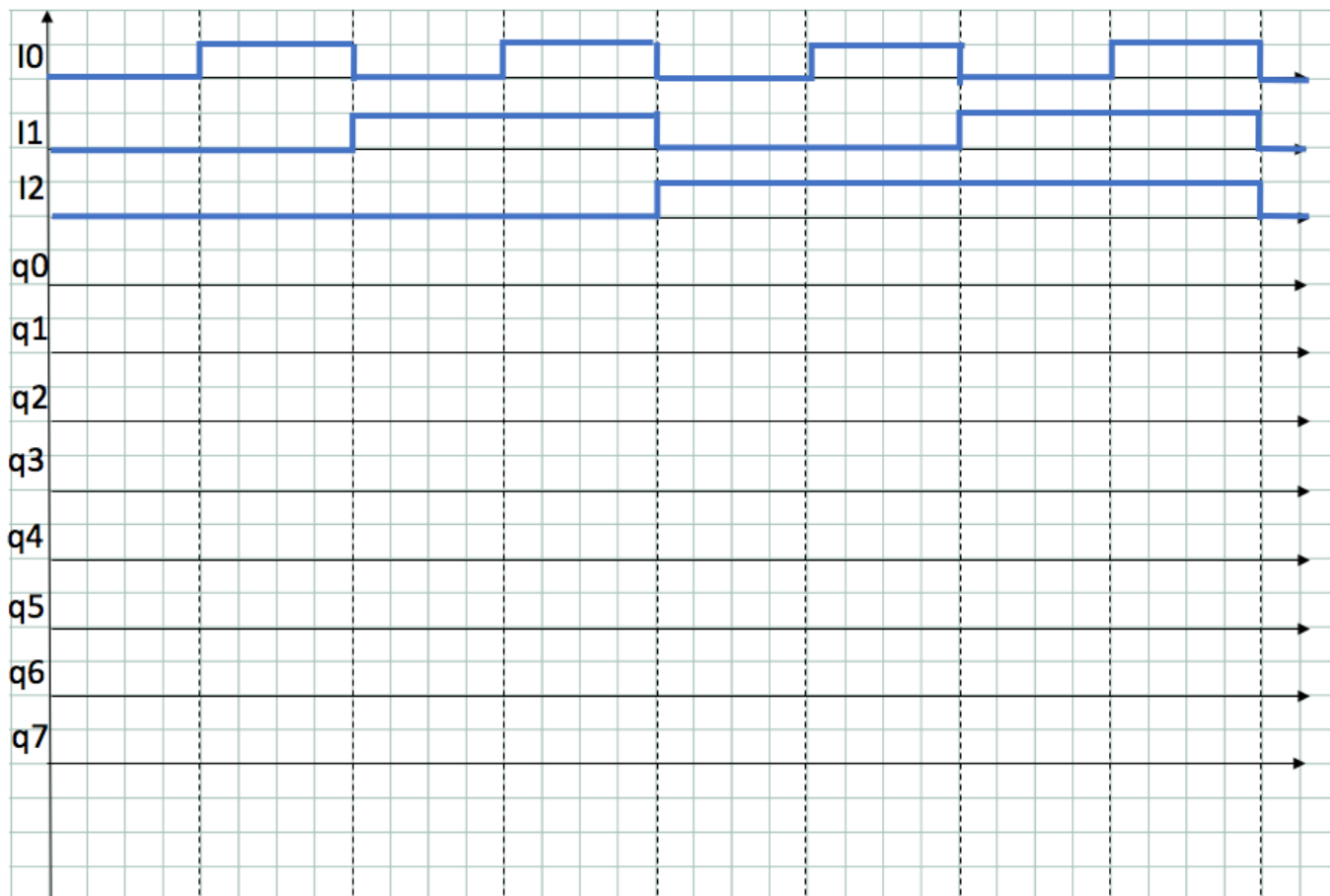
Q3=

- b) Complete la tabla de la descripción funcional de un DEC 3:8 con entradas y salidas activas en alto:

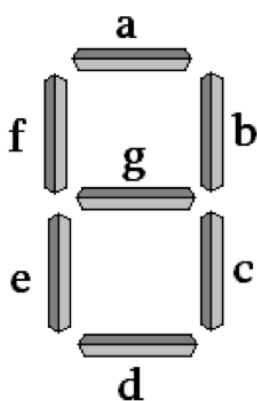


I2	I1	I0	Q7	Q6	Q5	Q4	Q3	Q2	Q1	Q0
0	0	0								
0	0	1								
0	1	0								
0	1	1								
1	0	0								
1	0	1								
1	1	0								
1	1	1								

- c) Represente las formas de onda de las salidas del DEC3:8 para la secuencia de valores de entrada del siguiente cronograma:



- d) Obtenga la tabla de funcionamiento de un **convertidor binario/7 segmentos**, esto es, obtenga la tabla de un circuito que devuelva una representación en siete segmentos del carácter que en hexadecimal tiene asociado el valor que representa en binario la entrada. En la representación, los segmentos activos deben valer 0 y los inactivos 1.



I3	I2	I1	I0	a	b	c	d	e	f	g
0	0	0	0							
0	0	0	1							
0	0	1	0							
0	0	1	1							
0	1	0	0							
0	1	0	1							
0	1	1	0							
0	1	1	1							
1	0	0	0							
1	0	0	1							
1	0	1	0							
1	0	1	1							
1	1	0	0							
1	1	0	1							
1	1	1	0							
1	1	1	1							

2. TRABAJO EXPERIMENTAL EN EL LABORATORIO

Diseñaremos en los siguientes apartados varios subsistemas combinacionales conocidos. En todos los diseños, el proceso a seguir será el mismo:

- 1) crear un proyecto nuevo;
- 2) añadir un módulo nuevo y especificar el diseño pedido en Verilog;
- 3) realizar una simulación;
- 4) programar la FPGA y comprobar si el funcionamiento es el previsto.

El primer diseño se realizará en Verilog funcional, a partir de las ecuaciones algebraicas; los restantes apartados se abordarán con Verilog procedimental, más potente para subsistemas.

2.1 DECODIFICADOR 2:4 (Descripción con Verilog funcional y simulación)

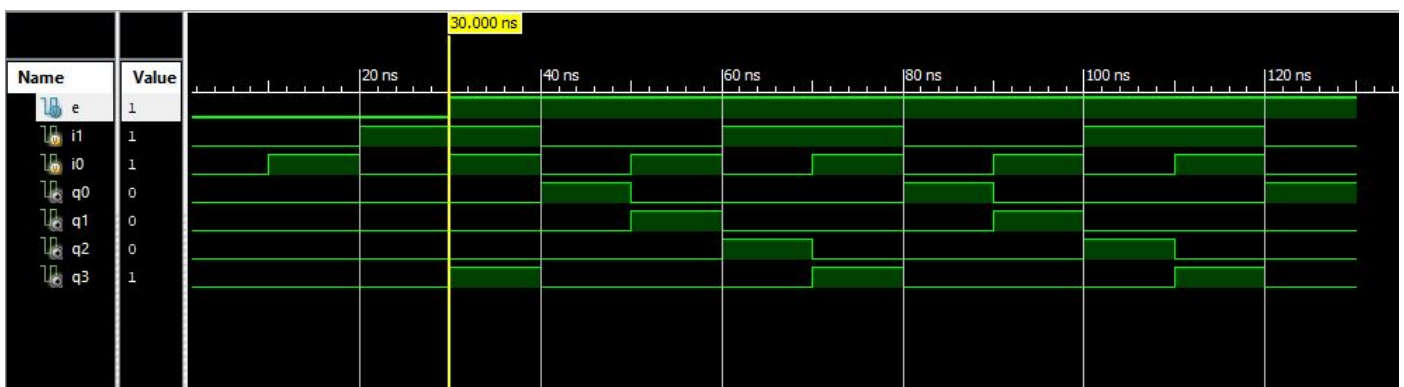
- a) A partir de las ecuaciones obtenidas en el apartado 1.a del estudio teórico, realice la especificación del DEC2:4 en Verilog mediante una descripción funcional.

```
module dec2a4(  
    input i1,i0,e,  
    output q0,q1,q2,q3  
);  
    // complete aquí la especificación funcional con assign  
  
    assign q0= ~i1 & ~i0 & e;  
  
endmodule
```

- b) Active “modo simulación” e incorpore ahora un fichero de simulación de este circuito y compruebe que funciona según la tabla del apartado 1.a del estudio teórico. Para ello, se propone utilizar las siguientes órdenes de simulación:

```
always #10 {i1,i0} = {i1,i0}+1;  
//Initialize Inputs  
initial begin  
    i1=0;  
    i0=0;  
    e=0;  
    #30;  
    e=1; //activo enable  
    #100;  
    $finish;  
end
```

ISim debe genera unas ondas equivalentes a estas:



2.2 DECODIFICADOR 3:8 (Descripción Verilog procedimental, simulación e implementación en FPGA)

Para subsistemas combinacionales más complejos, se aconseja utilizar descripciones procedimentales en Verilog, de tipo algorítmico, con bloques **always**, junto con el uso de variables multidimensionales (tipo BUS).

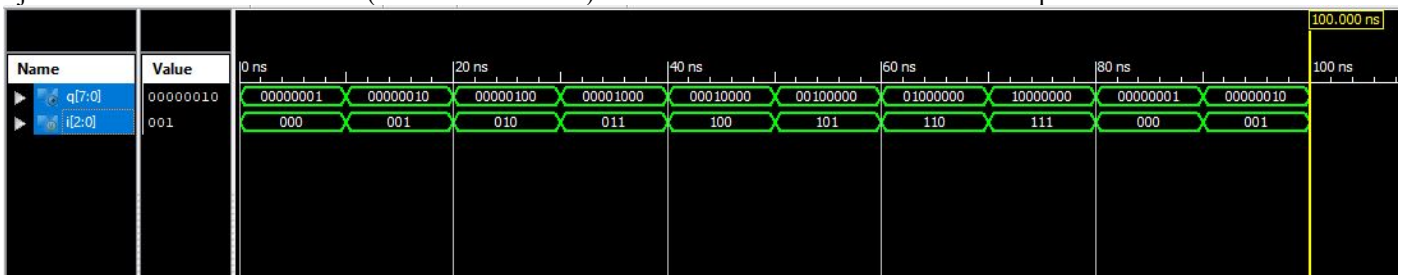
- a) Describa en Verilog procedimental el decodificador 3:8 del estudio teórico del apartado 1.b. (Se aconseja usar estructura case y variables multidimensionales para entradas y salidas)

```
module DEC3a8(  
    input [2:0] in,  
    output reg [7:0] q  
);  
//complete la descripción del dec3:8  
always @(in)  
    case (in)  
        0: q=8'b00000001;  
  
    endcase  
endmodule
```

- b) Active “modo simulación” e incorpore ahora un fichero de simulación de este circuito y compruebe que funciona según la tabla del apartado 1.a del estudio teórico. Para ello, se propone utilizar las siguientes órdenes de simulación:

```
always #10 in=in+1;  
  
//Initialize Inputs  
initial begin  
    in=0;  
    #100;  
    $finish;  
end
```

Ejecute la simulación del circuito (con ISIm Simulator). Deben obtenerse unas formas de onda equivalentes a:



- c) Vamos a implementar ahora el diseño en la placa de la FPGA. El proceso es el siguiente:

- Pase a modo implementación, **añada una copia el fichero de asociaciones basys2.ucf**, y realice las modificaciones necesarias para asignar 3 interruptores (SW2,SW1,SW0) como variables de entrada del DEC3:8, y 8 leds (LD7 a LD0) como salidas del decodificador. (Consulte la denominación de los pines del esquema de la placa en apéndice de la práctica). Sólo deben quedar sin comentarios, las líneas correspondientes a las asignaciones usadas, con el formato:

```
//está incompleto, debes seguir este formato para el resto de pines usados  
  
# asignación de Pin para el LED LD7  
NET "q<7>" LOC = "G1"; # Bank = 3, Signal name = LD7  
//así con todos los restantes  
  
# asignación de Pin para el switch SW2  
NET "in<2>" LOC = "K3"; # Bank = 3, Signal name = SW2
```

Fichero basys2.ucf incompleto para el dec3:8

- Generamos el fichero de programación, con la opción “Generate Programming File” (debe generarse un fichero con extensión .bit en la carpeta de nuestro proyecto).
- Realizamos la programación de la FPGA con ayuda de la aplicación ADEPT.

2.3 CONVERTIDOR BINARIO-7SEGMENTOS (Descripción Verilog estructural, simulación e implementación en FPGA)

- a) Describa en Verilog procedimental el convertidor de código binario de 4 bits a código 7-segmentos. Tenga en cuenta que en la placa BASYS2, los segmentos son activos en bajo y que disponen de una señal de activación (ánodo común) también activa en bajo. Es importante resaltar que los códigos de los segmentos (a,b,c,d,e,f,g) son comunes a los cuatro displays, por lo que, si activamos más de un display, se visualizará el mismo código en todos ellos.

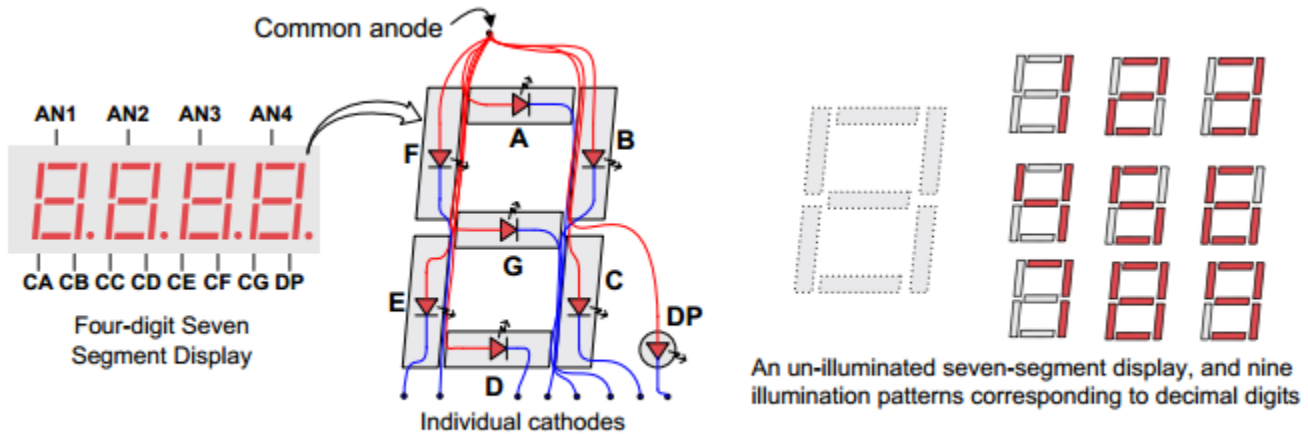


Figure 7. Seven-segment display

```
module convertidor_bin7seg(
    input  [3:0] in,           // entrada binaria 4 bits
    output [3:0] an,          //salidas para activar displays
    output reg a,b,c,d,e,f,g); // salida 7-segmentos

    always @*
    begin
        case (in)
            4'h0: {a,b,c,d,e,f,g} = 7'b0000001; //--0
            4'h1: {a,b,c,d,e,f,g} = 7'b1001111; //--1

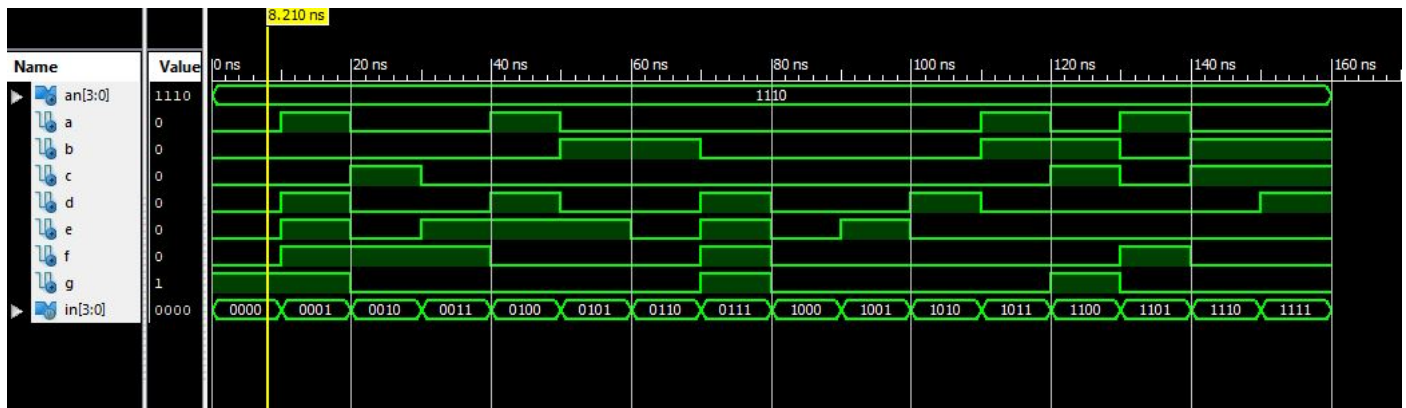
            // complete aquí la especificación

        endcase
    end
    assign an=4'b1110; //activo ánodo display derecha AN4
endmodule // convertidor_bin7seg
```

- b) Active “modo simulación” e incorpore ahora un fichero de simulación de este circuito y compruebe que funciona según la tabla del apartado 1.a del estudio teórico. Pare ello, se propone utilizar las siguientes órdenes de simulación:

```
always #10 in=in+1;
//Initialize Inputs
initial begin
    in=0;
    #160;
    $finish;
end
```

Ejecute la simulación del circuito (con ISIm Simulator). Deben obtenerse unas formas de onda equivalentes a:



- c) El fabricante de la placa, suministra un fichero de asociaciones “basys2.ucf” que tendremos que modificar para asignar las entradas y salida de nuestro módulo verilog, a pines (patitas o terminales) de la FPGA Spartan 3E que, a su vez, están conectados a elementos físicos de la placa (LEDs, switches, pulsadores, displays 7 segmentos, etc).
Incorpore al proyecto el fichero basys2.ucf suministrado y realice las modificaciones necesarias para asignar correctamente entradas y salidas del módulo descrito en verilog con los componentes físicos de la placa de la FPGA. Debe poner mucho cuidado en comprobar que los nombres de las entradas y salidas de su módulo verilog coinciden con los del fichero basys2.ucf. En este caso tenemos que asignar las salidas de activación de los segmentos (a,b,c,d,e,f,g), las salidas de activación de los ánodos (an) y las entradas del dato (in) a los pines correspondientes de la FPGA (ver el esquema del apéndice).

El proceso de generación del fichero de programación se realiza siguiendo los mismos pasos que en el proyecto del apartado anterior (decodificador).

APÉNDICE: ESQUEMA DE LA PLACA BASYS2

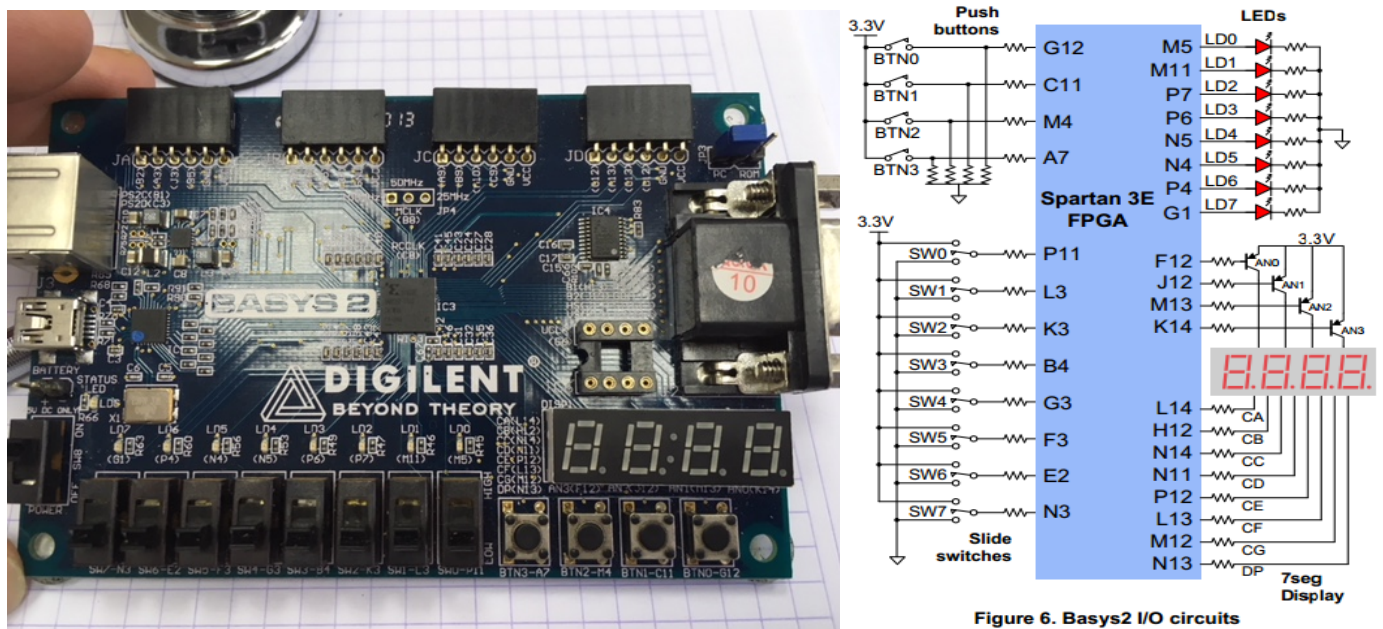


Figure 6. Basys2 I/O circuits