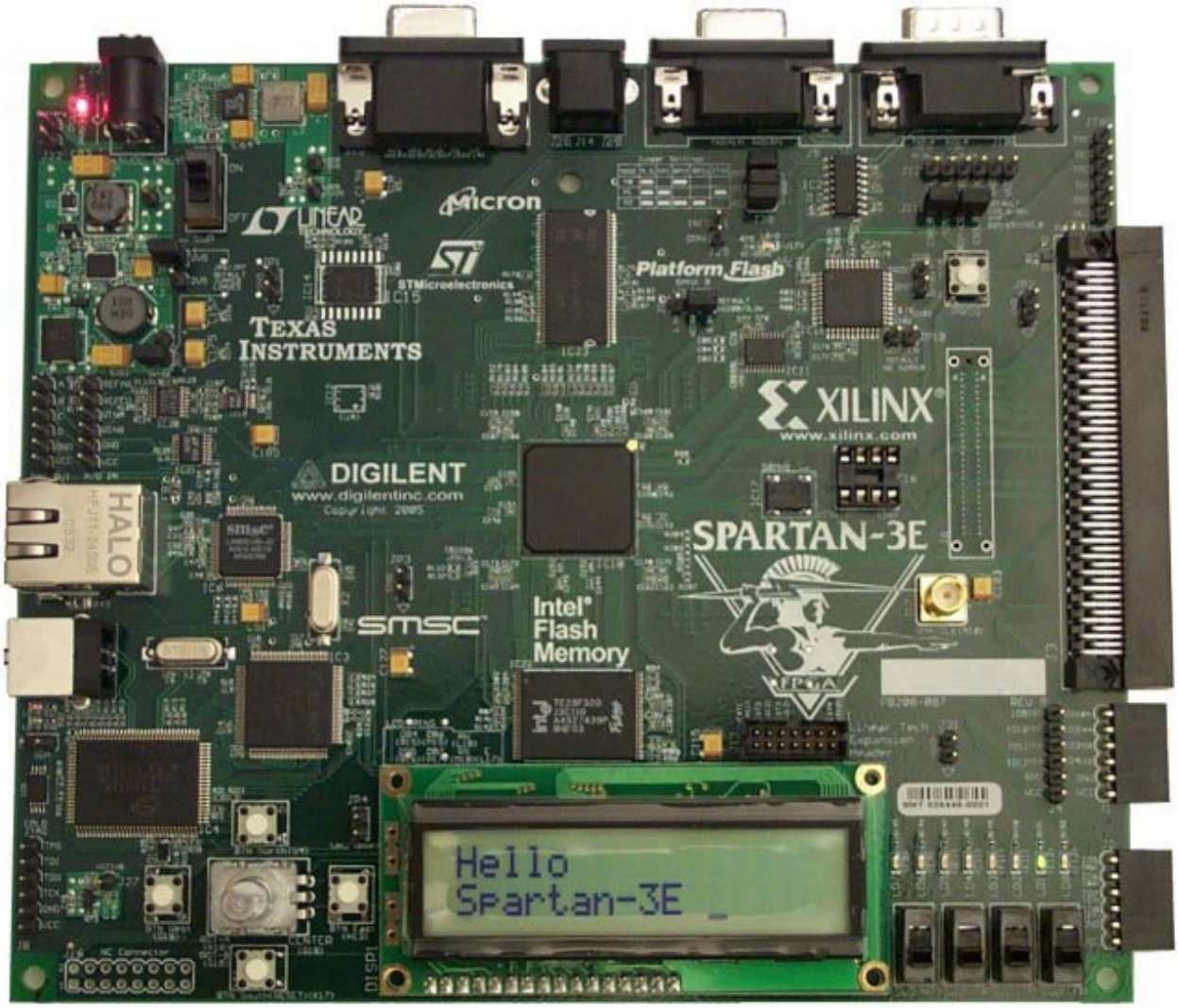
Memoria de Practicas
de
Fundamentos de Computadores



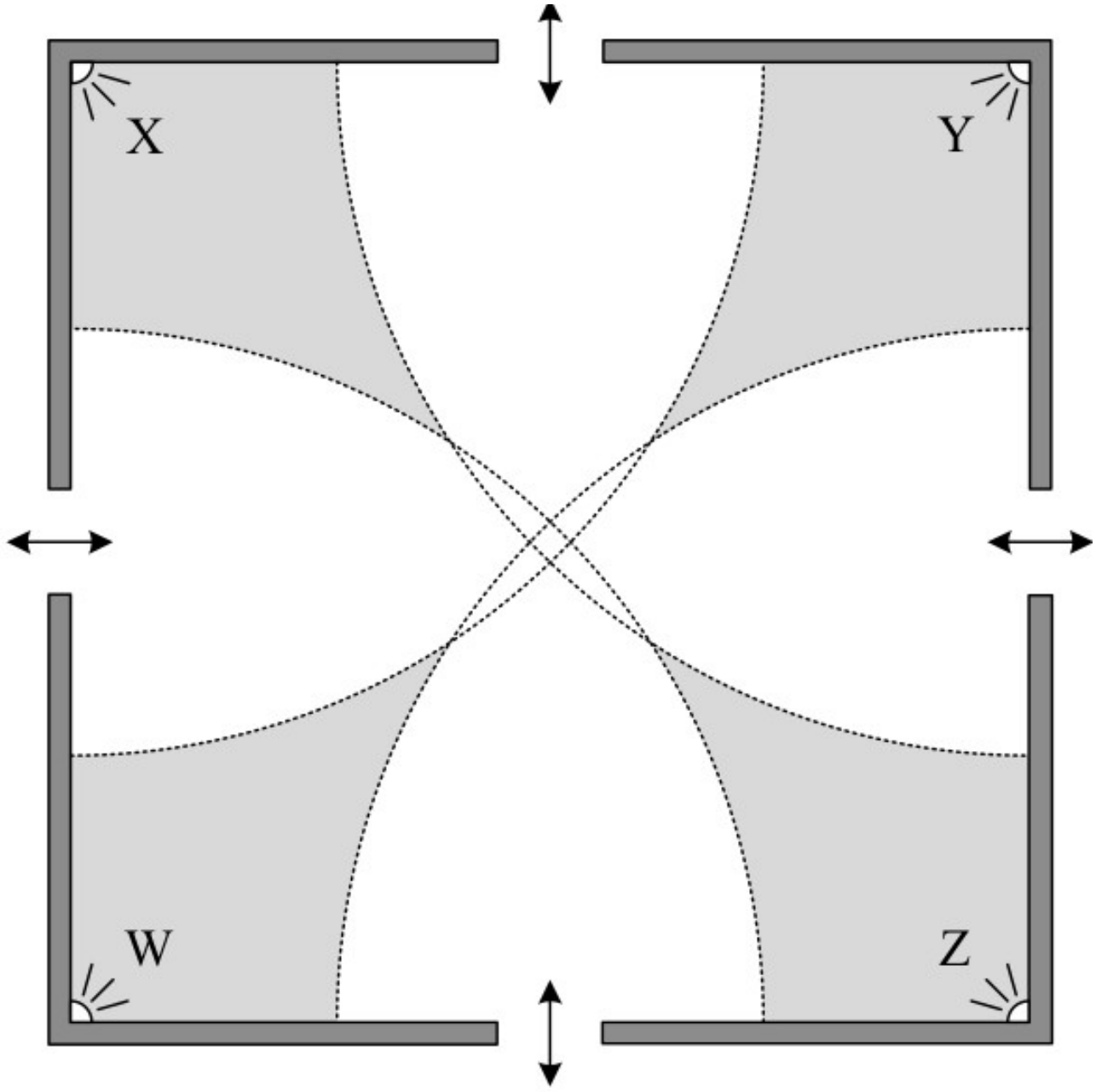
Fundamentos de Computadores
Almudena Garcia Jurado-Centurion
Practicas 5 y 6
Grupo FC2, Puesto 1

Índice

Practica 5: Implementación de un circuito combinacional mediante un dispositivo lógico programable

Especificaciones:

En las cuatro esquinas de una sala de exposiciones
de planta cuadrada se han instalado sendos detectores de presencia (X, Y, Z y W) cuyos radios de alcance se representan en la siguiente figura mediante arcos en línea discontinua.



Se desea implementar un sistema combinacional que asista al guarda de seguridad del recinto de la siguiente manera:

- Manteniendo encendida una lámpara L en un panel de monitorización mientras se encuentre alguna persona en el interior del recinto.
- Activando una señal acústica A cuando algún visitante se adentre en cualquiera de las zonas no transitables (áreas sombreadas).

Proceso operativo:

1. Representar la tabla de verdad del sistema con el siguiente orden en las variables de entrada: XYZW.

X	Y	Z	W	L	A
0	0	0	0	0	0
0	0	0	1	1	1
0	0	1	0	1	1
0	0	1	1	1	0
0	1	0	0	1	1
0	1	0	1	1	X
0	1	1	0	1	0
0	1	1	1	1	0
1	0	0	0	1	1
1	0	0	1	1	0
1	0	1	0	1	X
1	0	1	1	1	0
1	1	0	0	1	0
1	1	0	1	1	0
1	1	1	0	1	0
1	1	1	1	1	0

Comentarios respecto al diseño:

- La lampara se enciende siempre que se active al menos un sensor, con lo cual solo se apagara en la cuando la combinación de entrada sea 0000, es decir, cuando todos los sensores estén apagados.
- Las zonas no transitables son aquellas en las que solo un sensor esta activado, por lo tanto, la alarma solo se activara en esas combinaciones.

- Existen dos condiciones de no importa, debido a que es físicamente imposible situarse de forma que Y e W, o X y Z, estén activos sin que ningún otro sensor se active. De esta forma, es imposible activar la alarma en esas combinaciones.

2. Obtener las expresiones canónicas numéricas de las funciones de salida a partir de la tabla de verdad.

$L = \Pi_4 (0)$
$A = \sum_4 (1, 2, 4, 8) + \sum_\phi (5, 10)$

Comentario: En el caso de la función L, he escogido la expresión canónica conjuntiva para usar menos términos.

3. Haciendo uso del entorno Xilinx ISE WebPack, modelar en VHDL el circuito a partir de las expresiones canónicas obtenidas en el apartado 2.

Código fichero VHDL:

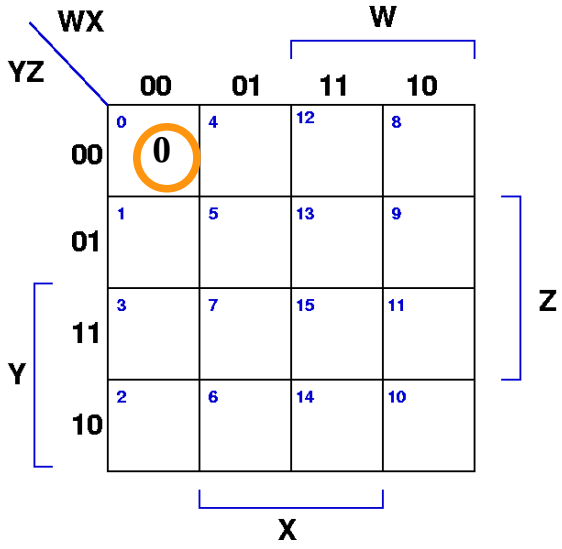
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;


entity Practica5 is
    Port ( X : in  STD_LOGIC;
           Y : in  STD_LOGIC;
           Z : in  STD_LOGIC;
           W : in  STD_LOGIC;
           L : out  STD_LOGIC;
           A : out  STD_LOGIC);
end Practica5;

architecture Behavioral of Practica5 is
           signal sensor: integer range 0 to 15;
           begin

                    sensor<=conv_integer(X&Y&Z&W);
                    with sensor select
                              A<='1' when 1|2|4|8,
                                        '0' when others;

                              L<='0' when sensor=0
                              else '1';
end Behavioral;
```

Comentarios respecto a la implementación:

- Al implementar el circuito a través de la tabla de verdad, las condiciones de “no importa” no hace falta ponerlas.
- Para facilitar la implementación, he unido las cuatro entradas en un vector, representando cada combinación en base 10; y he utilizado el operador | para ahorrar código dentro del with-select

- Para poder usar la función conv_integer, he tenido que añadir el paquete UNSIGNED.ALL

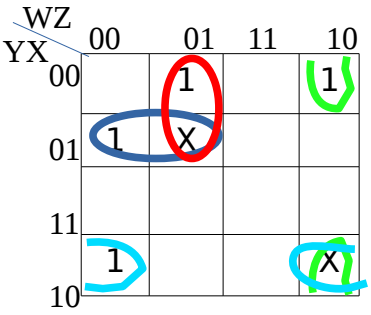
4. Simplificar por el método de Karnaugh la expresión canónica en forma de producto de sumas correspondiente a la función L.



L= X + Y + Z + W

Comentarios: L no se puede simplificar, ya que solo tiene un termino, por lo tanto la función simplificada es la misma que la original.

5. Simplificar por el método de Karnaugh la expresión canónica en forma de suma de productos correspondiente a la función A.



A= W' Z Y' + W Z' X' + Y' X W' + Y X' Z'

Comentarios: Las condiciones de no importa se usan como si fueran unos.

6. Haciendo uso del entorno Xilinx ISE WebPack, modelar en VHDL el circuito a partir de las expresiones simplificadas obtenidas en los apartados 4 y 5

Código fichero VHDL:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity Practica5_simplificada is
    Port ( X : in  STD_LOGIC;
                Y : in  STD_LOGIC;
                Z : in  STD_LOGIC;
                W : in  STD_LOGIC;
                L : out  STD_LOGIC;
                A : out  STD_LOGIC);
end Practica5_simplificada;
architecture Behavioral of Practica5_simplificada is
begin
    L<=X or Y or Z or W;
    A<=((not W) and Z and (not Y)) or (W and (not Z) and (not X)) or ((not Y) and X and (not W)) or (Y and (not X) and (not Z));
end Behavioral;
```

Comentarios sobre la implementación:

- La función L esta expresada como producto de sumas. Al tener solo un termino, y ser imposible de simplificar, su expresión equivale a la expresión canónica conjuntiva de dicho termino.
- La función A esta expresada en forma de suma de productos.
- El operador + se expresa mediante la sentencia “or”, y el operador * se expresa mediante la sentencia “and”

7. Ejecutar la simulación del sistema sobre el módulo VHDL obtenido en el apartado anterior, contrastando los resultados obtenidos con la tabla de verdad del apartado 1

Código VHDL del fichero de simulación:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use ieee.std_logic_arith.all;

ENTITY Test_pr5simp IS
END Test_pr5simp;


ARCHITECTURE behavior OF Test_pr5simp IS
-- Component Declaration for the Unit Under Test (UUT)

   COMPONENT Practica5_simplificada
   PORT(
        X : IN  std_logic;
        Y : IN  std_logic;
        Z : IN  std_logic;
        W : IN  std_logic;
        L : OUT  std_logic;
        A : OUT  std_logic
       );
   END COMPONENT;
```

```
 --Inputs
   signal X : std_logic := '0';
   signal Y : std_logic := '0';
   signal Z : std_logic := '0';
   signal W : std_logic := '0';

--Outputs
   signal L : std_logic;
   signal A : std_logic;
          signal valor: std_logic_vector(3 downto 0):="0000";
   signal valor_int: integer range 0 to 15;

BEGIN
          -- Instantiate the Unit Under Test (UUT)
   uut: Practica5_simplificada PORT MAP (
         X => X,
         Y => Y,
         Z => Z,
         W => W,
         L => L,
         A => A
        );

   -- Stimulus process
   stim_proc: process

   begin
      -- hold reset state for 100 ns.
      wait for 100 ns;
```

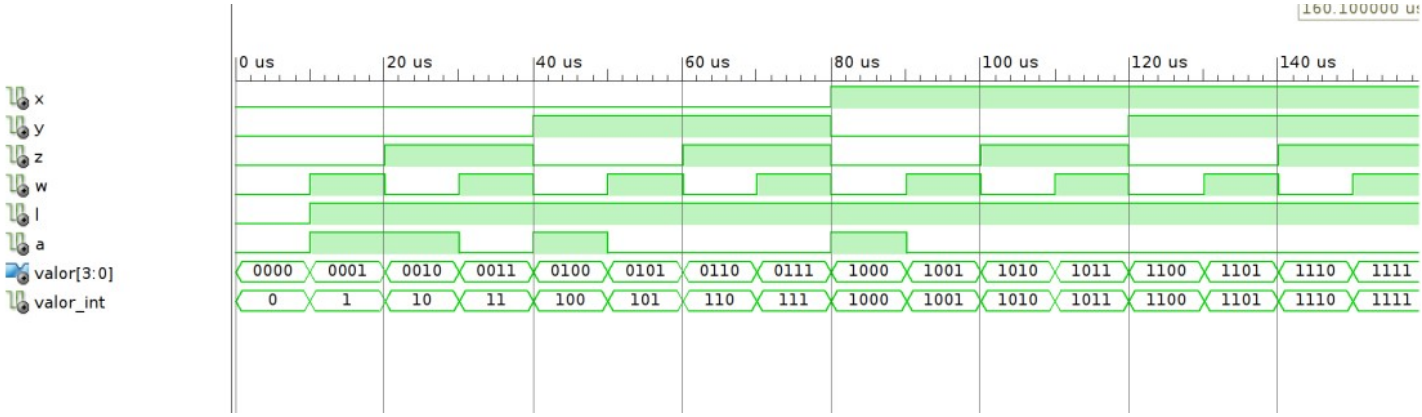
```
-- insert stimulus here
                for i in 0 to 15 loop
                        valor_int<=i; wait for 10 us;
                end loop;


                wait; wait;
   end process;
        valor<=conv_std_logic_vector(valor_int, 4);
        x<=valor(3);
        y<=valor(2);
        z<=valor(1);
        w<=valor(0);


END;
```

Comentarios respecto al fichero testbench:

- He implementado los estímulos del circuito a través de un bucle for, que va asignando valores del 0 al 15 a las entradas.
- El bucle for asigna los valores de entrada a una señal de tipo integer, que luego es convertida a un vector binario de tipo STD_LOGIC. Finalmente, cada componente del vector se asigna a una entrada del circuito
- Después de cada iteración del bucle, se espera un tiempo, para que el circuito termine de procesar la salida. Ese tiempo tiene que ser algo mayor que el tiempo de respuesta medio del circuito.

Salida del fichero de simulación:

La salida de la simulación es la siguiente:



8. Implementar en el laboratorio el circuito del apartado 6 haciendo uso de la tarjeta de desarrollo Spartan-3E de Xilinx y comprobar su correcto funcionamiento.

Copia del fichero de restricciones:

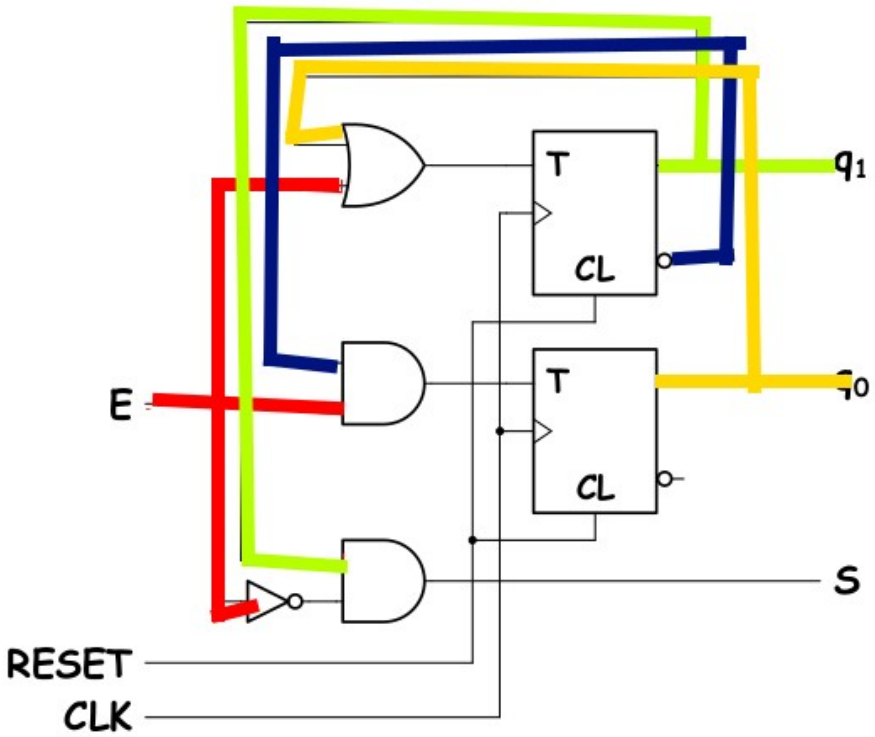
```
# ==== Slide Switches (SW) ====
NET W LOC = "L13" | IOSTANDARD = LVTTL | PULLUP ;
NET Z LOC = "L14" | IOSTANDARD = LVTTL | PULLUP ;
NET Y LOC = "H18" | IOSTANDARD = LVTTL | PULLUP ;
NET X LOC = "N17" | IOSTANDARD = LVTTL | PULLUP ;

# ==== Pushbuttons (BTN) ====
#NET "BTN_EAST" LOC = "H13" | IOSTANDARD = LVTTL | PULLDOWN ;
#NET "BTN_NORTH" LOC = "V4" | IOSTANDARD = LVTTL | PULLDOWN ;
#NET "BTN_SOUTH" LOC = "K17" | IOSTANDARD = LVTTL | PULLDOWN ;
#NET "BTN_WEST" LOC = "D18" | IOSTANDARD = LVTTL | PULLDOWN ;

# ==== Clock inputs (CLK) ====
#NET "CLK_50MHZ" LOC = "C9" | IOSTANDARD = LVCMOS33 ;
# Define clock period for 50 MHz oscillator (40%/60% duty-cycle)
#NET "CLK_50MHZ" PERIOD = 20.0ns HIGH 40%;
#NET "CLK_AUX" LOC = "B8" | IOSTANDARD = LVCMOS33 ;
#NET "CLK_SMA" LOC = "A10" | IOSTANDARD = LVCMOS33 ;

# ==== 6-pin header J1 ====
#NET "J1<0>" LOC = "B4" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 6 ;
#NET "J1<1>" LOC = "A4" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 6 ;
#NET "J1<2>" LOC = "D5" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 6 ;
#NET "J1<3>" LOC = "C5" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 6 ;
```

```
# ==== 6-pin header J2 ====
#NET "J2<0>" LOC = "A6" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 6 ;
#NET "J2<1>" LOC = "B6" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 6 ;
#NET "J2<2>" LOC = "E7" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 6 ;
#NET "J2<3>" LOC = "F7" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 6 ;

# ==== Character LCD (LCD) ====
#NET "LCD_E" LOC = "M18" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
#NET "LCD_RS" LOC = "L18" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
#NET "LCD_RW" LOC = "L17" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
#NET "SF_D<8>" LOC = "R15" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
#NET "SF_D<9>" LOC = "R16" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
#NET "SF_D<10>" LOC = "P17" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
#NET "SF_D<11>" LOC = "M15" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW =
SLOW ;

# ==== Discrete LEDs (LED) ====
#NET "LED<0>" LOC = "F12" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
#NET "LED<1>" LOC = "E12" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
#NET "LED<2>" LOC = "E11" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
#NET "LED<3>" LOC = "F11" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
#NET "LED<4>" LOC = "C11" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
#NET "LED<5>" LOC = "D11" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
NET A LOC = "E9" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;
NET L LOC = "F9" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;

# ==== Rotary Pushbutton Switch (ROT) ====
#NET "ROT_A" LOC = "K18" | IOSTANDARD = LVTTL | PULLUP ;
#NET "ROT_B" LOC = "G18" | IOSTANDARD = LVTTL | PULLUP ;
#NET "ROT_CENTER" LOC = "V16" | IOSTANDARD = LVTTL | PULLDOWN ;
```

Practica 6: Análisis y descripción de sistemas con biestables

Especificaciones:

Dado el circuito de la figura:



- Realizar el análisis del mismo, obteniendo su diagrama de estados.
- Describir el circuito en VHDL, simularlo e implementarlo en una FPGA.

Proceso operativo:

1. Analizar el circuito mostrado en las especificaciones y obtener su diagrama de estados

- Proceso de análisis:

Para comenzar el análisis del circuito, partimos de la ecuación del biestable T.

La ecuación del biestable T es: **T Qn' + T ' Qn**

Para obtener las ecuaciones del circuito, sustituimos T por las entradas de los biestables, y Qn por las salidas de los mismos.

De esta forma las ecuaciones quedarían:

Q1 = (q0 + E)* q1 ' + (q0 + E) ' * q1

Q0 = E q1 ' q0 ' + (E*q1 ') ' * q0

S = E ' * q1

- Expresión de ecuaciones en forma de suma de productos:

Para facilitar su identificación en la tabla de estados, vamos a expresar estas mismas ecuaciones en forma de suma de productos.

Tras los cálculos, las ecuaciones quedarían así:

Q1 = q0 q1 ' + E q1 ' + q0 ' E ' q1

Q0 = E q1 ' q0 ' + E ' q0 + q1 q0

S = E ' q1

- Tabla de estados

Ahora vamos a obtener la tabla de estados.

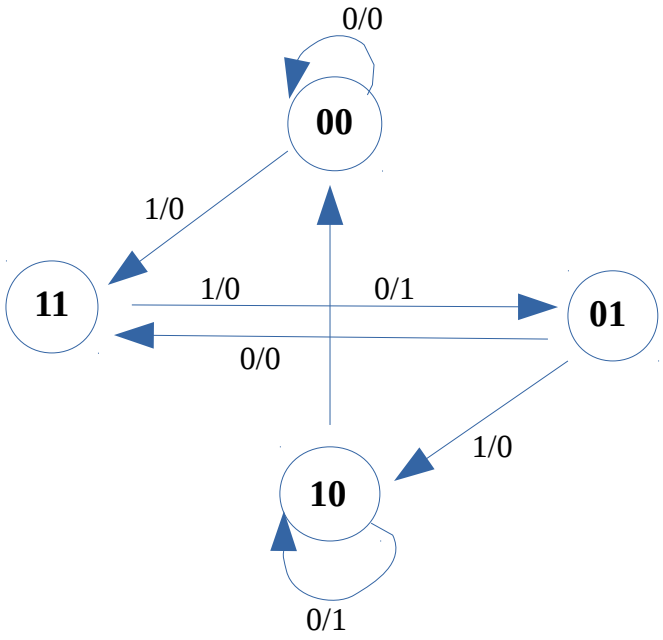
Para rellenarla, buscamos los términos de las ecuaciones en la tabla, ponemos 1 en las salidas de las funciones correspondiente a los términos que se muestran en la ecuación, y 0 en todos los demás.

Entrada	Estado presente		Estado futuro		Salida
E	q1	q0	Q1	Q0	S
0	0	0	0	0	0
0	0	1	1	1	0
0	1	0	1	0	1
0	1	1	0	1	1
1	0	0	1	1	0
1	0	1	1	0	0
1	1	0	0	0	0
1	1	1	0	1	0

- Diagrama de estados

Para hacer el diagrama de estados, colocamos un circulo representando cada estado, y unas flechas que muestren las transiciones entre estados en función de las entradas del sistema.

Al ser este un sistema tipo Mealy, las transiciones también deben mostrar las salidas.



2. Modelar en un fichero fuente VHDL un flip-flop tipo T activado por flancos de subida con una entrada de puesta a cero asíncrona activa a nivel alto

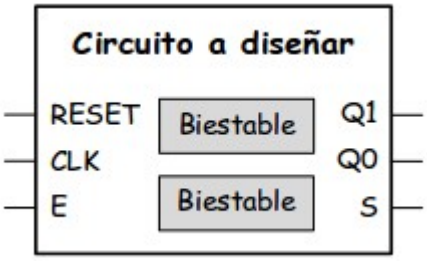
Código VHDL del Biestable T:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity Biestable_T is
    Port ( T : in  STD_LOGIC;
           CLK : in  STD_LOGIC;
           CL : in  STD_LOGIC;
           q : inout  STD_LOGIC);
end Biestable_T;

architecture Behavioral of Biestable_T is
signal q_interna: std_logic;
begin
           process(CL, CLK, T, q_interna)
           begin
                    if CL='1' then q_interna<='0';
                    elsif CLK'event and CLK='1' then
                              if q_interna='0' and T='1' then q_interna<='1';
                                        elsif q_interna='1' and T='1' then q_interna<='0';
                              end if;
                    end if;
           end process;
           q<=q_interna;
end Behavioral;
```

Comentarios sobre la implementación VHDL del Biestable T:

- El biestable ha sido implementado a través de su tabla de verdad.
- Al no poder tomar una salida como señal de entrada, hemos declarado una señal auxiliar q_interna para poder realizar las operaciones.
- He declarado la salida q del biestable como inout, para poder realizar la realimentación del circuito sin problemas.

3. Crear un segundo fichero fuente para describir el circuito mostrado en las especificaciones. Declarar dentro de este fichero como componente el biestable modelado en el apartado anterior. A continuación, insertar dicho biestable tantas veces como sea preciso y añadir la lógica necesaria



Código VHDL del circuito base:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Practica6 is
    Port ( E : in  STD_LOGIC;
           q1 : inout  STD_LOGIC;
           CLK: in STD_LOGIC;
           CL: in STD_LOGIC;
           q0 : inout  STD_LOGIC;
           S : out  STD_LOGIC);
end Practica6;
architecture Behavioral of Practica6 is
           COMPONENT Biestable_T is
                    Port (T : in  STD_LOGIC;
                             CLK : in  STD_LOGIC;
                             CL : in  STD_LOGIC;
                             q : inout  STD_LOGIC);
           end component;
           signal T1, T2: STD_LOGIC;
```

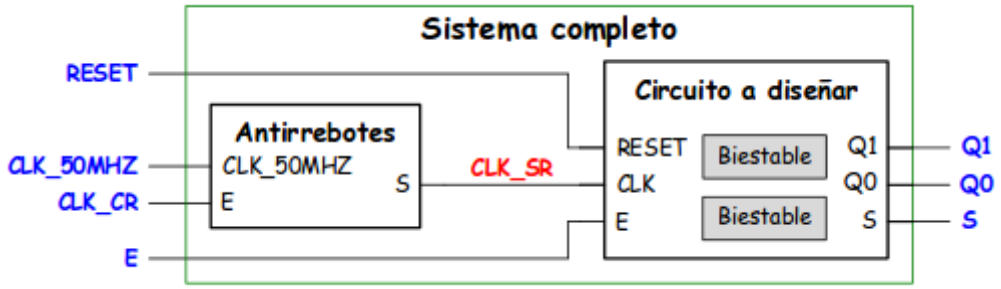
```
begin
        biestable1: Biestable_T port map(T=>T1, CLK=>CLK, CL=>CL, q=>q0);
        biestable2: Biestable_T port map(T=>T2, CLK=>CLK, CL=>CL, q=>q1);

        T2<=q0 or E;
        T1<=(not q1) and E;
        S<=(not E) and q1;
end Behavioral;
```

Comentarios respecto a la implementación:

- Al igual que en el biestable, aquí también he declarado las salidas q1 y q0 como inout, para poder realimentar el circuito y conectarlas al puerto q del biestable sin problemas.
- Para implementar el circuito, he declarado el biestable T (definido en el anterior fichero) como componente, asignándole los mismos puertos que este tiene en su entidad.

  A continuación, he creado dos instancias de este, para representar los dos biestables.
- He declarado dos señales T1 y T2, las cuales he conectado a los puertos T de cada biestable. A la salida de los biestables, he conectado las salidas q1 y q0, declaradas en la entidad del circuito.
- Finalmente, he asignado a T1, T2 y S sus respectivas entradas representadas en el circuito, a través de sentencias lógicas.

4. Crear un tercer fichero fuente para describir el circuito completo que se implementará en la FPGA. Declarar dentro de este fichero dos componentes, como se muestra en la siguiente figura.



- **El primer componente es un circuito antirrebotes para eliminar los rebotes producidos por el pulsador que se utilizará como señal de reloj. El código de este circuito puede descargarse de la sección de prácticas de la plataforma moodle de la asignatura.**
- **El segundo componente es el circuito modelado en el apartado 3.**

Código VHDL del circuito completo:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Pr6_Completo is
    Port ( RESET : in  STD_LOGIC;
           CLK_50MHZ : in  STD_LOGIC;
           CLK_CR : in  STD_LOGIC;
           CLK_SR: inout STD_LOGIC;
           E : in  STD_LOGIC;
           Q1 : inout  STD_LOGIC;
           Q0 : inout  STD_LOGIC;
           S : out  STD_LOGIC);
end Pr6_Completo;
```

```
architecture Behavioral of Pr6_Completo is

    Component Antirrebotes is
        GENERIC (SIMULAR: STD_LOGIC := '0');
        PORT (CLK_50MHZ, E: IN STD_LOGIC; S: OUT STD_LOGIC);
    End component;


    Component Practica6 is
        Port ( E : in  STD_LOGIC;
               q1 : inout  STD_LOGIC;
               CLK: in STD_LOGIC;
               CL: in STD_LOGIC;
               q0 : inout  STD_LOGIC;
               S : out  STD_LOGIC);
    End Component;
begin
    AR: Antirrebotes port map(CLK_50MHZ=>CLK_50MHZ, E=>CLK_CR, S=>CLK_SR);
    Pr6: Practica6 port map(E=>E, q1=>Q1, CLK=>CLK_SR, CL=>RESET, q0=>q0, S=>S);
end Behavioral;
```

Comentarios sobre la implementación:

Siguiendo el mismo procedimiento que en el fichero anterior, he declarado dos componentes: uno para el antirrebotes, y otro para el circuito.

He creado una instancia de cada uno y, en el port map, he conectado los puertos tal y como se muestra en la figura.

5. Simular el sistema completo del apartado 4 y contrastar los resultados con el diagrama de estados obtenido en el apartado 1. Para ello, se aplicará a la entrada E una secuencia de valores que permita comprobar todas las transiciones y valores de salida del diagrama.

Código VHDL del fichero de simulacion:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY Test_Pr6 IS
END Test_Pr6;

 ARCHITECTURE behavior OF Test_Pr6 IS

-- Component Declaration for the Unit Under Test (UUT)
   COMPONENT Pr6_Completo
   PORT(
        RESET : IN  std_logic;
        CLK_50MHZ : IN  std_logic;
        CLK_CR : IN  std_logic;
        CLK_SR : INOUT  std_logic;
        E : IN  std_logic;
        Q1 : INOUT  std_logic;
        Q0 : INOUT  std_logic;
        S : OUT  std_logic
       );
   END COMPONENT;
```

```
--Inputs
  signal RESET : std_logic := '0';
  signal CLK_50MHZ : std_logic := '0';
  signal CLK_CR : std_logic := '0';
  signal E : std_logic := '0';

--BiDirs
  signal CLK_SR : std_logic;
  signal Q1 : std_logic;
  signal Q0 : std_logic;

--Outputs
  signal S : std_logic;

  -- Clock period definitions
  constant CLK_50MHZ_period : time := 10 ns;
  constant CLK_CR_period : time := 10 ns;
  constant CLK_SR_period : time := 10 ns;
BEGIN

        -- Instantiate the Unit Under Test (UUT)
  uut: Pr6_Completo PORT MAP (
       RESET => RESET,
       CLK_50MHZ => CLK_50MHZ,
       CLK_CR => CLK_CR,
       CLK_SR => CLK_SR,
       E => E,
       Q1 => Q1,
       Q0 => Q0,
       S => S);
```

```
-- Clock process definitions
   CLK_50MHZ_process :process
   begin
                CLK_50MHZ <= '0';
                wait for CLK_50MHZ_period/2;
                CLK_50MHZ <= '1';
                wait for CLK_50MHZ_period/2;
   end process;

   CLK_CR_process :process
   begin
                CLK_CR <= '0';
                wait for CLK_CR_period/2;
                CLK_CR <= '1';
                wait for CLK_CR_period/2;
   end process;

   CLK_SR_process :process
   begin
                CLK_SR <= '0';
                wait for CLK_SR_period/2;
                CLK_SR <= '1';
                wait for CLK_SR_period/2;
   end process;

-- Stimulus process
   stim_proc: process
   begin
      -- hold reset state for 100 ns.
      wait for 100 ns;
```

```
wait for CLK_50MHZ_period*10;

-- insert stimulus here

        Reset <= '1';
        wait for 10 ns;

        Reset <= '0';
        wait for 10 ns;

        wait until CLK_CR ='1';
        E <= '0';
        wait until CLK_CR ='1';
        E <= '1';
        wait until CLK_CR ='1';
        E <= '1';
        wait until CLK_CR ='1';
        E <= '1';
        wait until CLK_CR ='1';
        E <= '0';
        wait until CLK_CR ='1';
        E <= '0';
        wait until CLK_CR ='1';
        E <= '1';
        wait until CLK_CR ='1';
        E <= '1';
        wait until CLK_CR ='1';
        E <= '0';
        wait until CLK_CR ='1';
        E <= '1';
```

```
        wait until CLK_CR ='1';
        E <= '0';
        wait until CLK_CR ='1';
        E <= '0';
        wait until CLK_CR ='1';
        E <= '1';
        wait until CLK_CR ='1';
        E <= '1';
        wait until CLK_CR ='1';
        E <= '1';
        wait until CLK_CR ='1';
        E <= '0';

    wait; wait;
  end process;

END;
```

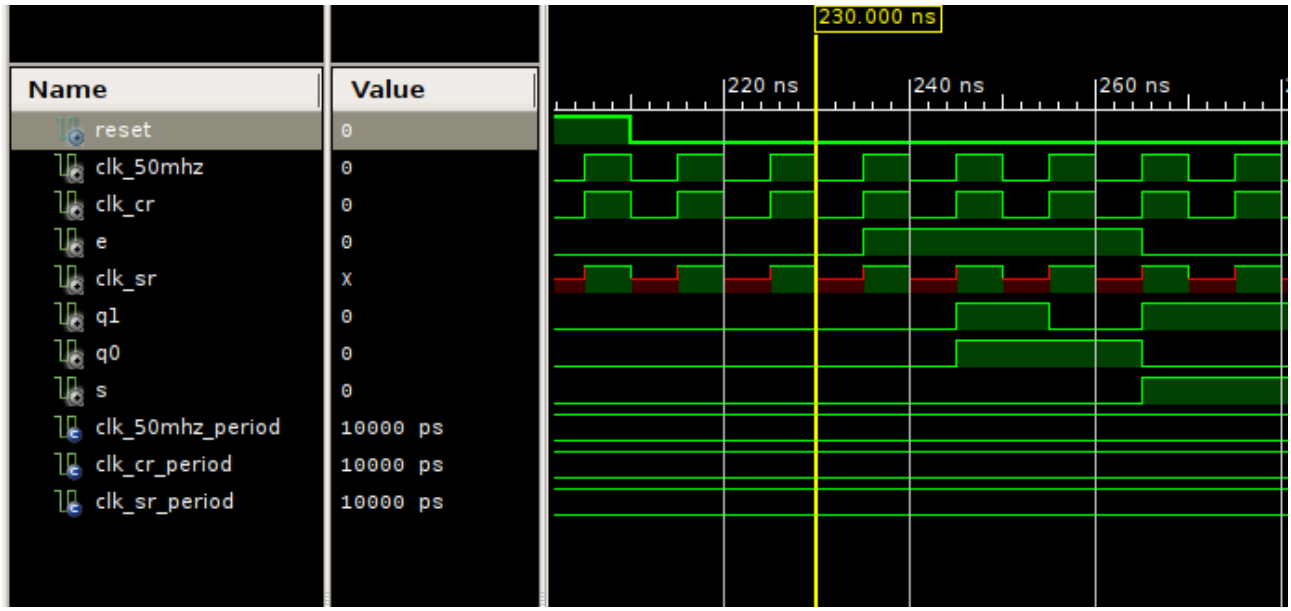
Comentarios respecto al fichero de simulación:

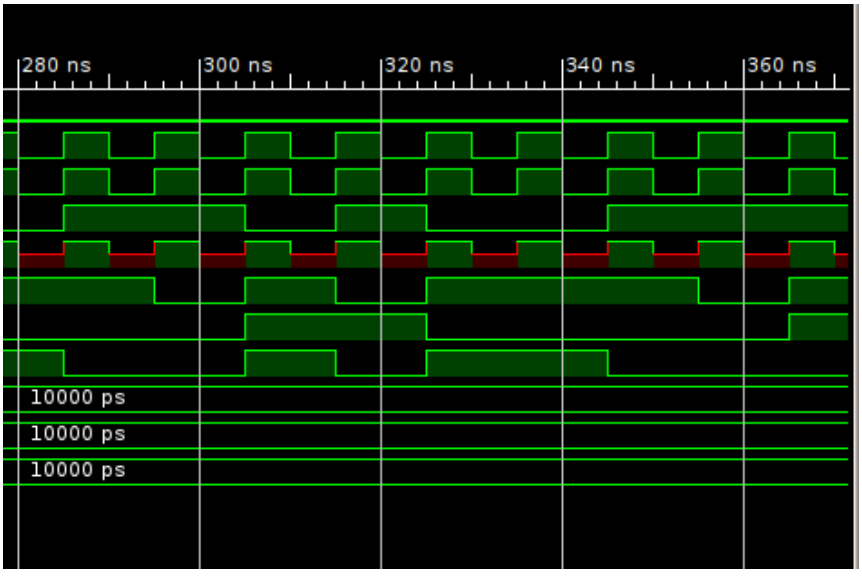
- El fichero de simulación es bastante parecido al de la Practica 5, con las diferencias de que en esta practica si existe un reloj, y que en el proceso de estímulos el wait tiene que esperar al flanco de reloj (en lugar de un valor fijo) antes de asignar las entradas.
- Como el circuito solo tiene una entrada, solo hay que asignarle valores a esta.

Salida de la simulación:

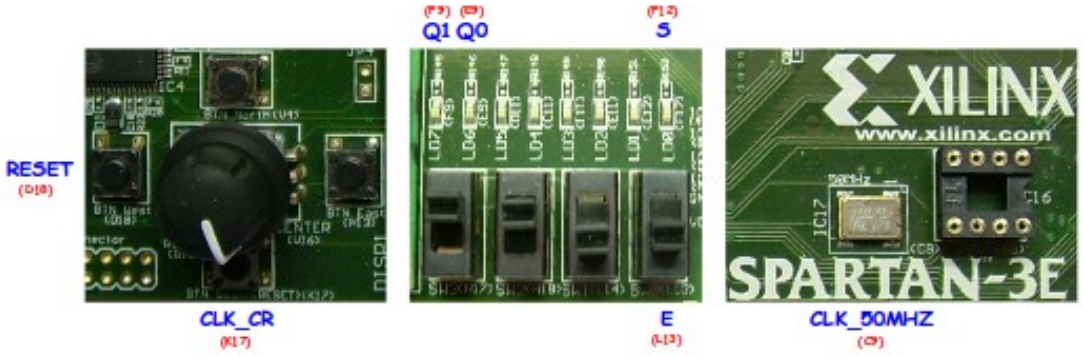
La salida de la simulación es la siguiente:

(La pongo en dos imágenes para que se vea mejor)





6. Implementar el circuito obtenido en el apartado 4 en la tarjeta de desarrollo Spartan-3E de Xilinx y comprobar su correcto funcionamiento. Para ello, conectar los diferentes terminales del circuito tal como se muestra en la siguiente figura.



Copia del fichero de restricciones:

```
############################################################
### SPARTAN-3E STARTER KIT BOARD CONSTRAINTS FILE
############################################################

# ==== Slide Switches (SW) ====
NET E LOC = "L13" | IOSTANDARD = LVTTL | PULLUP ;
#NET "SW<1>" LOC = "L14" | IOSTANDARD = LVTTL | PULLUP ;
#NET "SW<2>" LOC = "H18" | IOSTANDARD = LVTTL | PULLUP ;
#NET "SW<3>" LOC = "N17" | IOSTANDARD = LVTTL | PULLUP ;

# ==== Pushbuttons (BTN) ====
#NET "BTN_EAST" LOC = "H13" | IOSTANDARD = LVTTL | PULLDOWN ;
#NET "BTN_NORTH" LOC = "V4" | IOSTANDARD = LVTTL | PULLDOWN ;
NET CLK_CR LOC = "K17" | IOSTANDARD = LVTTL | PULLDOWN ;
NET "CLK_CR" CLOCK_DEDICATED_ROUTE = FALSE;
NET RESET LOC = "D18" | IOSTANDARD = LVTTL | PULLDOWN ;
```

```
# ==== Clock inputs (CLK) ====
NET CLK_50MHZ LOC = "C9" | IOSTANDARD = LVCMOS33 ;
# Define clock period for 50 MHz oscillator (40%/60% duty-cycle)
NET CLK_50MHZ PERIOD = 20.0ns HIGH 40%;
#NET "CLK_AUX" LOC = "B8" | IOSTANDARD = LVCMOS33 ;
#NET "CLK_SMA" LOC = "A10" | IOSTANDARD = LVCMOS33 ;


# ==== 6-pin header J1 ====
#
#NET "J1<0>" LOC = "B4" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 6 ;
#NET "J1<1>" LOC = "A4" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 6 ;
#NET "J1<2>" LOC = "D5" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 6 ;
#NET "J1<3>" LOC = "C5" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 6 ;

# ==== 6-pin header J2 ====
#
#NET "J2<0>" LOC = "A6" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 6 ;
#NET "J2<1>" LOC = "B6" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 6 ;
#NET "J2<2>" LOC = "E7" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 6 ;
#NET "J2<3>" LOC = "F7" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 6 ;

# ==== Character LCD (LCD) ====
#NET "LCD_E" LOC = "M18" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
#NET "LCD_RS" LOC = "L18" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
#NET "LCD_RW" LOC = "L17" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
#
#NET "SF_D<8>" LOC = "R15" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
#NET "SF_D<9>" LOC = "R16" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
#NET "SF_D<10>" LOC = "P17" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
```

#NET "SF_D<11>" LOC = "M15" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;

==== Discrete LEDs (LED) ====

#

NET S LOC = "F12" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;

#NET "LED<1>" LOC = "E12" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;

#NET "LED<2>" LOC = "E11" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;

#NET "LED<3>" LOC = "F11" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;

#NET "LED<4>" LOC = "C11" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;

#NET "LED<5>" LOC = "D11" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;

NET Q0 LOC = "E9" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;

NET Q1 LOC = "F9" | IOSTANDARD = LVTTL | SLEW = SLOW | DRIVE = 8 ;

==== Rotary Pushbutton Switch (ROT) ====

#NET "ROT_A" LOC = "K18" | IOSTANDARD = LVTTL | PULLUP ;

#NET "ROT_B" LOC = "G18" | IOSTANDARD = LVTTL | PULLUP ;

#NET "ROT_CENTER" LOC = "V16" | IOSTANDARD = LVTTL | PULLDOWN ;