Memoria TOP_SYSTEM

En este apartado se crea el modulo top_system en el cual esta incluido el modulo cnt_epp, con este modulo se pretende simular completamente la funcionalidad de dicho componente y una vez verificado se volcara a la placa para su comprobación final.

Las entradas y salidas de este apartado son las citadas a continuación ademas se especifica el tipo de dato que son y si son de entrada o de salida:

```
CLK : in std_logic;
RST : in std_logic;
ASTRB : in std_logic;
DSTRB : in std_logic;
DATA : inout std_logic_vector(7 downto 0);
PWRITE : in std_logic;
PWAIT : out std_logic;
SWITCHES_I : in std_logic_vector(7 downto 0);
PSH_BUTTON:in std_logic;
LEDS_O : out std_logic_vector (7 downto 0));
```

Señales auxiliares utilizadas en el programa para poder realizar las interconexiones entre las distintas partes del los componentes:

```
signal DIR ,DIR_REG, DATO, DATO_REG, DATO_RD   : std_logic_vector (7 downto 0);
signal DIR_VLD, DATO_VLD, CE_RD : std_logic;
```

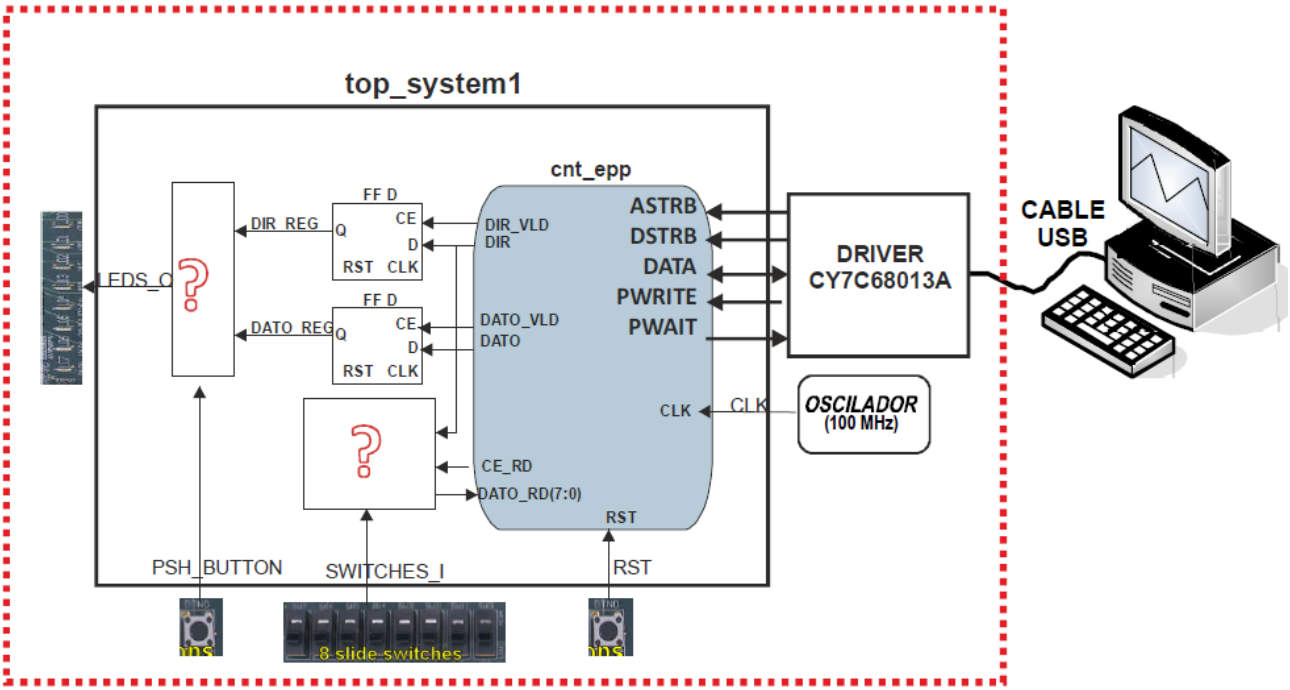
Se ha de especificar el componente que hemos utilizado dentro del sistema para poder usarlo de no ser así el sistema no funcionara.

```
DUT : entity work.cnt_epp
port map (
CLK      => CLK,
RST      => RST,
ASTRB    => ASTRB,
DSTRB    => DSTRB,
DATA     => DATA,
PWRITE   => PWRITE,
PWAIT    => PWAIT,
DATO_RD  => DATO_RD,
CE_RD    => CE_RD,
DIR      => DIR,
DIR_VLD  => DIR_VLD,
DATO     => DATO,
DATO_VLD => DATO_VLD);
```

Para su implementación dentro del archivo adjunto entregado por el profesor se ha de realizar el circuito descrito en la imagen adjunta a continuación.

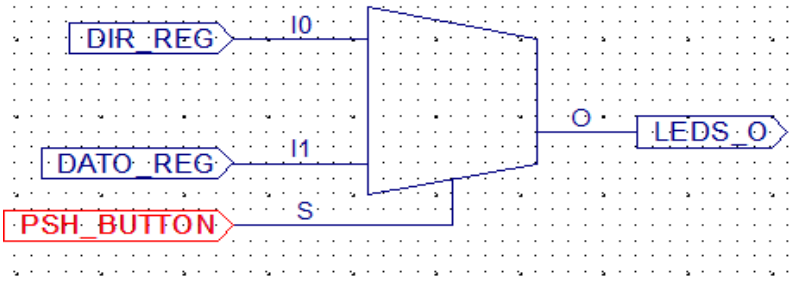
Los dos circuitos con ? Corresponden a lo descrito en el siguiente apartado:

- El bloque controlado con la señal *PSH_BUTTON* permite seleccionar que información se visualiza en los LEDS de forma que con un nivel bajo se selecciona la dirección registrada y con un alto el dato registrado.
- El bloque al que tiene conectado los 8 switches de la placa (puerto *SWITCHES_I*) deberá llevarlos a la entrada ***DATO_RD*** del componente ***cnt_epp*** cuando se realice un ciclo de lectura en la posición 32HEX.

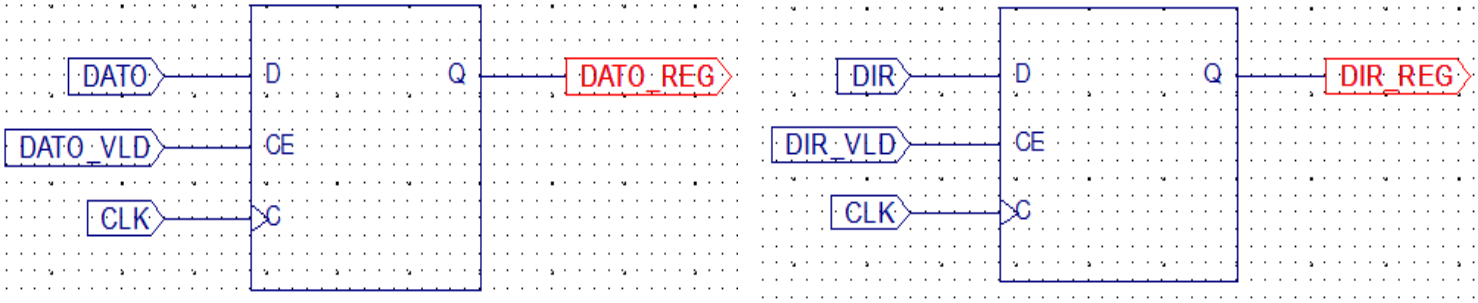

top_system1
cnt_epp
FF D
DIR_REG
CE
D
Q
RST CLK
DIR_VLD
DIR
FF D
DATO_REG
CE
D
Q
RST CLK
DATO_VLD
DATO
ASTRB
DSTRB
DATA
PWRITE
PWAIT
CLK
CE_RD
DATO_RD(7:0)
RST
LEDS_O
PSH_BUTTON
SWITCHES_I
RST
8 slide switches
DRIVER
CY7C68013A
CABLE
USB
OSCILADOR
(100 MHz)
CLK

Circuitos Digitales con explicación

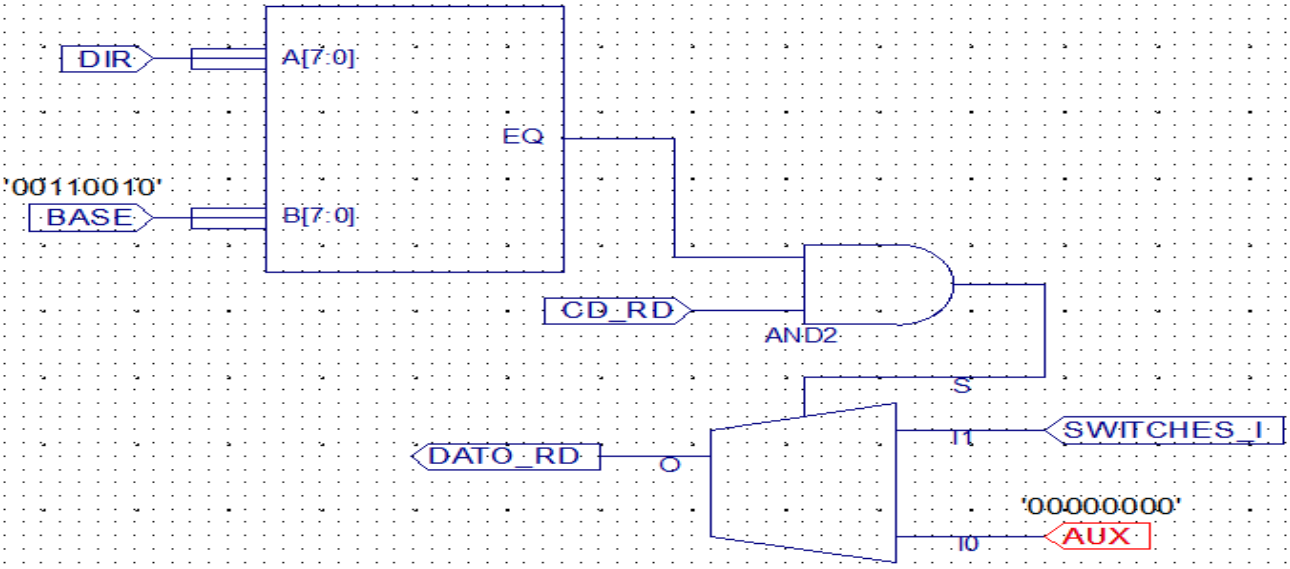
Presente en: top_system
Función: Su funcionamiento es el de elegir una de las dos entradas y pasarla a los leds para ello se utiliza el botón psh_button en cual selecciona a nivel bajo la dirección y a nivel alto el dato



Presente en: top_system
Función: Su función es la de almacenar en su salida el contenido de la entrada ya sea DATO o DIR, estos se consigue cuando se realiza un pulso de reloj y CE esta activo para cada uno de ello (DIR_VLD o DATO_VLD), al cumplirse esta condición se almacena en la salida el contenido de ese momento en la entrada permaneciendo ahí hasta la próxima vez que se cumpla la condición.



Presente en: top_system
Función: La finalidad de este circuito es la de transferir el contenido de SWITCHES_I a DATO_RD cuando se cumpla la condición y durante cierto periodo de tiempo, para ello primero ha de cumplir que en DIR se encuentre el valor x"32" para que pueda continuar, de ser así si la señal CD_RD esta activa entonces permite el paso del contenido de SWITCHES_I para que sea copiado en DATO_RD, si no se cumplen las condiciones anteriores se transmite la señal aux que contiene "00000000" como información.



Top_System Code

```
library ieee;
use ieee.std_logic_1164.all;

entity top_system1 is
  port(
    CLK        : in    std_logic;
    RST        : in    std_logic;
    ASTRB      : in    std_logic;
    DSTRB      : in    std_logic;
    DATA       : inout std_logic_vector(7 downto 0);
    PWRITE     : in    std_logic;
    PWAIT      : out   std_logic;
    SWITCHES_I : in    std_logic_vector(7 downto 0);
    PSH_BUTTON : in    std_logic;
    LEDS_O     : out   std_logic_vector (7 downto 0));
end top_system1;

architecture rtl of top_system1 is
  signal DIR ,DIR_REG, DATO, DATO_REG, DATO_RD   : std_logic_vector (7 downto 0);
  signal DIR_VLD, DATO_VLD, CE_RD : std_logic;
begin  -- rtl


  DUT : entity work.cnt_epp
    port map (
      CLK      => CLK,
      RST      => RST,
      ASTRB    => ASTRB,
      DSTRB    => DSTRB,
      DATA     => DATA,
      PWRITE   => PWRITE,
      PWAIT    => PWAIT,
      DATO_RD  => DATO_RD,
      CE_RD    => CE_RD,
      DIR      => DIR,
      DIR_VLD  => DIR_VLD,
      DATO     => DATO,
      DATO_VLD => DATO_VLD);

  --DIR_REG biestabale que almacena la información de la dir cada DIR_VLD
  process (CLK, RST)
  begin
   if RST='0' then
      DIR_REG <= (others => '0');
   elsif (CLK'event and CLK='1') then
      if DIR_VLD = '1' then
         DIR_REG <= DIR;
      end if;
   end if;
  end process;
```

```
  --DATO_REG biestable que almacena la información de la dir cada DATO_REG
 process (CLK, RST)
 begin
  if RST='0' then
     DATO_REG <= (others => '0');
  elsif (CLK'event and CLK='1') then
     if DATO_VLD = '1' then
        DATO_REG <= DATO;
     end if;
  end if;
 end process;

--Multiplexor que en función de el estado de PSH_BUTTON elige DATO_REG o DIR_REG
LEDS_O <= DATO_REG WHEN PSH_BUTTON ='1' ELSE  DIR_REG;

  --  Transfiere el contenido de SWITCHES_I a DATO_RD cuando es el ciclo de lectura x32
  process (DIR, CE_RD, SWITCHES_I) is
  begin
   if DIR=x"32" and CE_RD='1' then
     DATO_RD <= SWITCHES_I;
   else
     DATO_RD <= (others => '0');
   end if;
  end process;


end rtl;
```

TestBench

Una vez creado el código en base a los circuitos digitales que queremos implementar, pasamos a la creación del testbench de este apartado, para verificar su correcto funcionamiento para ello creamos el archivo top_system_tb en el cual implementamos toda la lógica necesaria para poder simular el comportamiento del componente, hemos de incluir el archivo epp_device para poder probar correctamente su funcionamiento, el archivo cnt_epp no es necesario declararlo ya que esta incluido en el propio top_system.

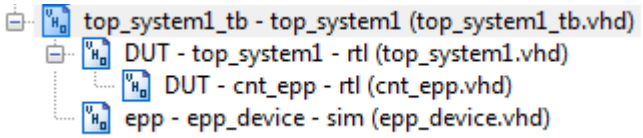
En el archivo epp_device se ha forzado la dir x"32" para verificar una de sus funciones no se añade el código ya que es el mismo que en el apartado cnt_epp

El código TestBench con todos los datos seria el citado a continuación:

```
library ieee;
use ieee.std_logic_1164.all;

-------------------------------------------------------------------------------

entity top_system1_tb is

end entity top_system1_tb;

-------------------------------------------------------------------------------

architecture top_system1 of top_system1_tb is

  -- component ports
  signal CLK        : std_logic := '0';
  signal RST        : std_logic := '0';
  signal ASTRB      : std_logic := '1';
  signal DSTRB      : std_logic := '1';
  signal DATA       : std_logic_vector(7 downto 0);
  signal PWRITE     : std_logic := '1';
  signal PWAIT      : std_logic;
  signal SWITCHES_I : std_logic_vector(7 downto 0) := x"30";
  signal PSH_BUTTON : std_logic := '0';
  signal LEDS_O     : std_logic_vector (7 downto 0);

begin  -- architecture top_system1

  -- component instantiation
  DUT: entity work.top_system1
    port map (
      CLK        => CLK,
      RST        => RST,
      ASTRB      => ASTRB,
      DSTRB      => DSTRB,
      DATA       => DATA,
      PWRITE     => PWRITE,
      PWAIT      => PWAIT,
      SWITCHES_I => SWITCHES_I,
```

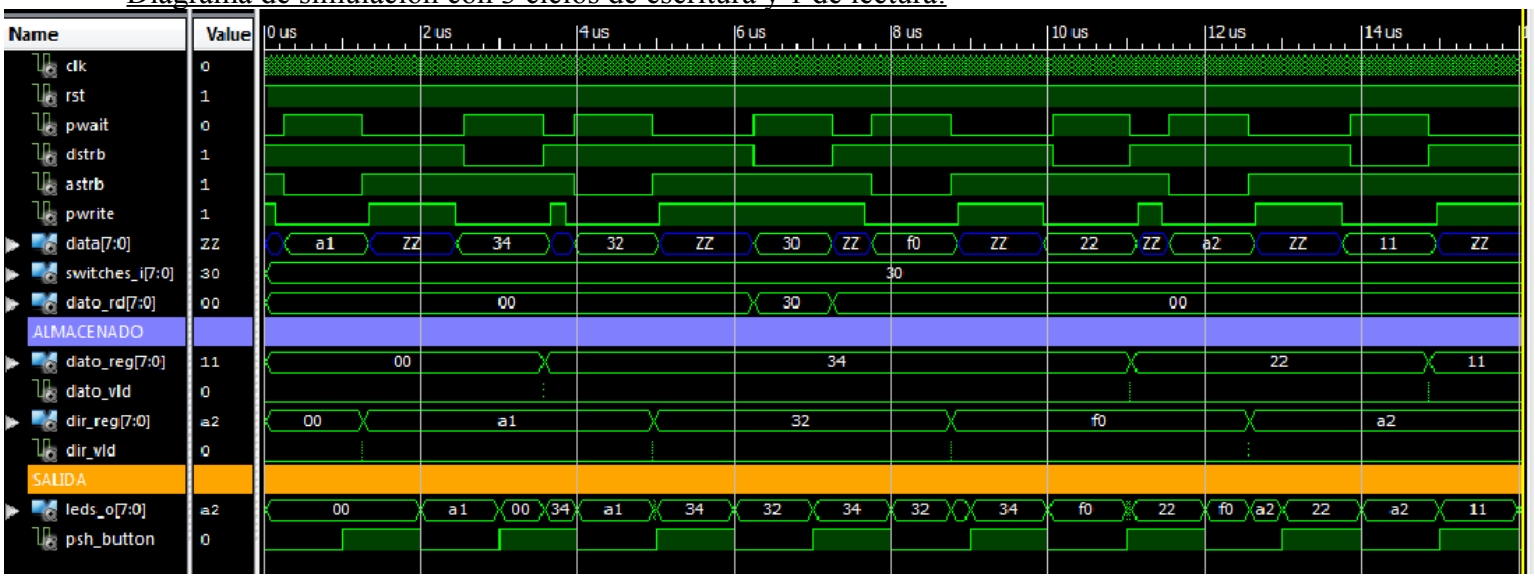
```
    PSH_BUTTON => PSH_BUTTON,
    LEDS_O     => LEDS_O);

        epp: entity work.epp_device
   port map (
                    DATA   => DATA,
                          PWRITE => PWRITE,
                          DSTRB  => DSTRB,
                          ASTRB  => ASTRB,
                          PWAIT  => PWAIT);


  PSH_BUTTON <= not PSH_BUTTON after 1000 ns;
  -- clock generation
  CLK <= not CLK after 5 ns;
  RST <= '0', '1' after 25 ns;


end architecture top_system1;
```

En el programa hemos de obtener algo parecido a lo mostrado en la imagen para que su funcionamiento sea el correcto.



Simulación Funcional

Diagrama de simulación con 3 ciclos de escritura y 1 de lectura:



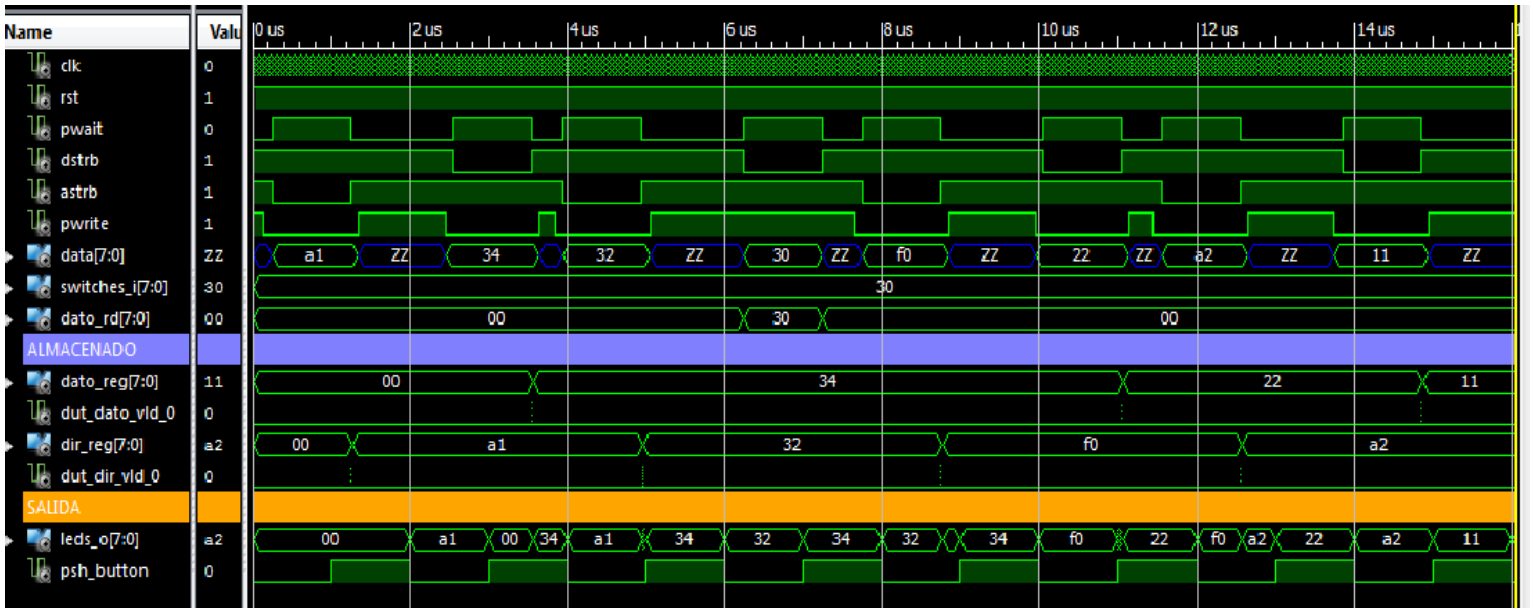
Una vez analizados los datos obtenidos de las gráficas y comparados con los de la practica si los resultados son satisfactorios podemos continuar con el siguiente paso.

Simulación Temporal

Después de verificar que los ciclos son correctos pasamos a realizar la simulación temporal para ello seguimos los paso explicados en clase:

- Realizamos en el apartado de implementación el Post&Place and Route
- Cambiamos a la ventana de simulación
- Elegimos Post-Rute en el desplegable
- Chequeamos la síntesis del código
- Realizamos la simulación temporal en la cual se tienen en cuenta los retardos de las puertas lógicas y del resto de los componentes

Diagrama de simulación temporal con 3 ciclos de escritura y 1 de lectura:



Una vez analizados los datos obtenidos de las gráficas y comparados con los de la practica si los resultados son satisfactorios podemos continuar con el siguiente paso.

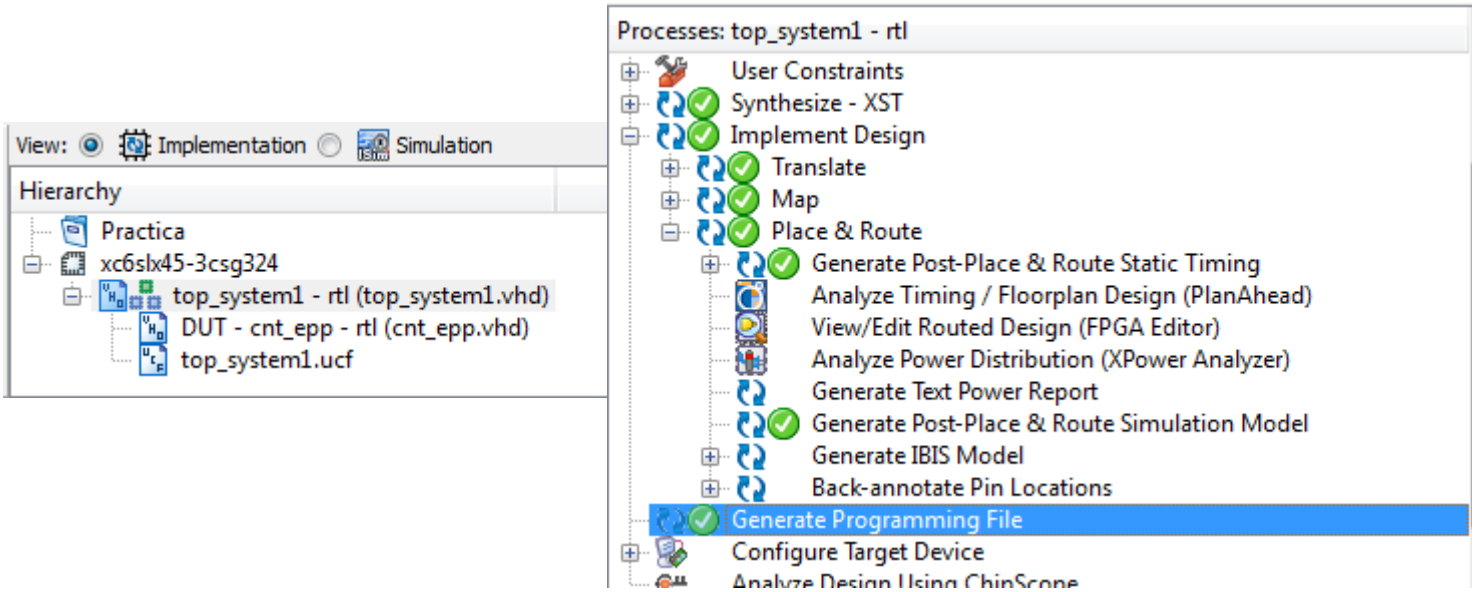
Recursos Utilizados

Slice Logic Utilization	Used	Available	Utilization
Number of Slice Registers	36	54,576	1,00%
Number used as Flip Flops	36		
Number used as Latches	0		
Number used as Latch-thrus	0		
Number used as AND/OR logics	0		
Number of Slice LUTs	36	27,288	1,00%
Number used as logic	28	27,288	1,00%
Number using O6 output only	18		
Number using O5 output only	0		
Number using O5 and O6	10		
Number used as ROM	0		
Number used as Memory	0	6,408	0,00%
Number used exclusively as route-thrus	8		
Number with same-slice register load	8		
Number with same-slice carry load	0		
Number with other load	0		
Number of occupied Slices	14	6,822	1,00%
Number of MUXCYs used	0	13,644	0,00%
Number of LUT Flip Flop pairs used	36		
Number with an unused Flip Flop	10	36	27,00%
Number with an unused LUT	0	36	0,00%
Number of fully used LUT-FF pairs	26	36	72,00%
Number of unique control sets	3		
Number of slice register sites lost to control set restrictions	4	54,576	1,00%
Number of bonded IOBs	31	218	14,00%
Number of RAMB16BWERs	0	116	0,00%
Number of RAMB8BWERs	0	232	0,00%
Number of BUFIO2/BUFIO2_2CLKs	0	32	0,00%
Number of BUFIO2FB/BUFIO2FB_2CLKs	0	32	0,00%
Number of BUFG/BUFGMUXs	1	16	6,00%
Number used as BUFGs	1		
Number used as BUFGMUX	0		
Number of DCM/DCM_CLKGENs	0	8	0,00%
Number of ILOGIC2/ISERDES2s	0	376	0,00%
Number of IODELAY2/IODRP2/IODRP2_MCBs	0	376	0,00%
Number of OLOGIC2/OSERDES2s	0	376	0,00%
Number of BSCANs	0	4	0,00%
Number of BUFHs	0	256	0,00%
Number of BUFPLLs	0	8	0,00%
Number of BUFPLL_MCBs	0	4	0,00%
Number of DSP48A1s	0	58	0,00%
Number of ICAPs	0	1	0,00%
Number of MCBs	0	2	0,00%
Number of PCILOGICSEs	0	2	0,00%
Number of PLL_ADVs	0	4	0,00%
Number of PMVs	0	1	0,00%
Number of STARTUPs	0	1	0,00%
Number of SUSPEND_SYNCs	0	1	0,00%
Average Fanout of Non-Clock Nets	01/02/76		

Generación del archivo a descargar en placa

Una vez comprobado que la simulación temporal es correcta pasamos a añadir al programa el archivo top_system1.uucf en el cual se especifican cuales son los puertos que se usan y a que salida o entrada corresponden en la placa. Quedando igual que en la imagen.

Después generamos el archivo.bit el cual sera el descargado en la placa para su posterior testeo final

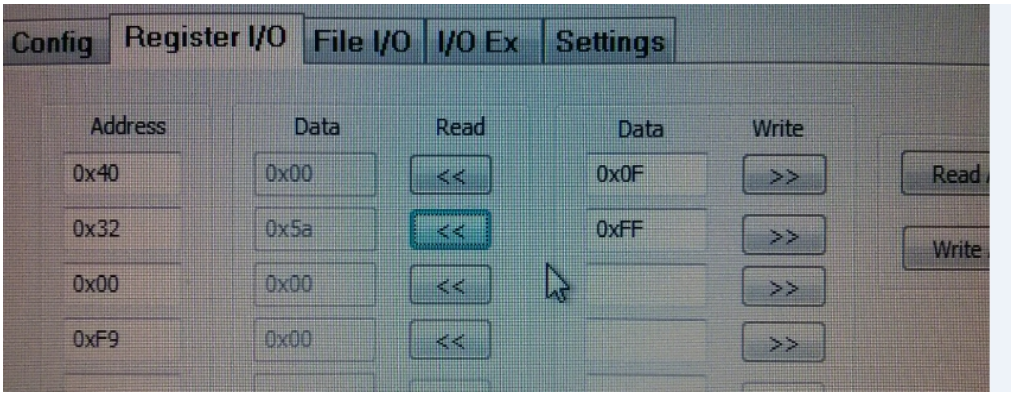


Top_system.ucf Code

```
NET "CLK"   LOC = "L15";


# onBoard USB controller
 NET "ASTRB"   LOC = "B9";
 NET "DSTRB"   LOC = "A9";
 NET "PWRITE"   LOC = "C15";
 NET "PWAIT"   LOC = "F13";
 NET "DATA<0>"  LOC = "A2";
 NET "DATA<1>"  LOC = "D6";
 NET "DATA<2>"  LOC = "C6";
 NET "DATA<3>"  LOC = "B3";
 NET "DATA<4>"  LOC = "A3";
 NET "DATA<5>"  LOC = "B4";
 NET "DATA<6>"  LOC = "A4";
 NET "DATA<7>"  LOC = "C5";

# onBoard Pushbuttons

 NET "PSH_BUTTON"  LOC = "F5";
 NET "RST"   LOC = "T15";
# onBoard LEDS
 NET "LEDS_O<0>" LOC = "U18";
 NET "LEDS_O<1>" LOC = "M14";
 NET "LEDS_O<2>" LOC = "N14";
 NET "LEDS_O<3>" LOC = "L14";
 NET "LEDS_O<4>" LOC = "M13";
 NET "LEDS_O<5>" LOC = "D4";
 NET "LEDS_O<6>" LOC = "P16";
 NET "LEDS_O<7>" LOC = "N12";


# onBoard SWITCHES
 NET "SWITCHES_I<0>" LOC = "A10";
 NET "SWITCHES_I<1>" LOC = "D14";
 NET "SWITCHES_I<2>" LOC = "C14";
 NET "SWITCHES_I<3>" LOC = "P15";
 NET "SWITCHES_I<4>" LOC = "P12";
 NET "SWITCHES_I<5>" LOC = "R5";
 NET "SWITCHES_I<6>" LOC = "T5";
 NET "SWITCHES_I<7>" LOC = "E4";
```

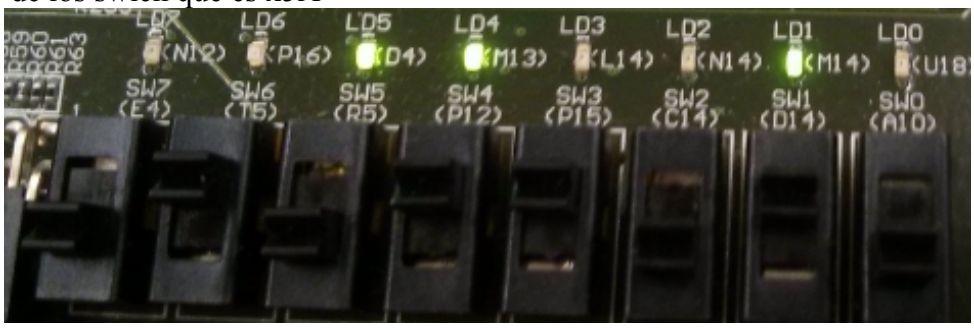
Descarga y test en placa Top_system

Para este apartado una vez generado el archivo .bit hemos de ir al ordenador del profesor en el cual se encuentra instalado el programa para volcar los datos a la placa para su posterior prueba.



En el programa cargamos en la parte de la izquierda las direcciones de donde queremos leer la informacion que en este caso la información se encuentra almacenada en los swich de la placa, pero solo puede ser leída si se elige la dirección x32 el resto no leen nada

En el caso de la imagen superior cuando leemos la dirección x32 mostrada en los leds nos muestra el contenido de los swich que es x5A



Y en la parte de la derecha podemos insertar una dirección de escritura la cual se mostrara en los leds cuando se pulse el botón que se a definido para alternar la información entre la dirección de lectura y la de escritura en este caso la dirección introducida es xF0.