

Sistemas Digitales

Laboratorio

Semestre 2022-1

Sesión 2

Profesor: Kalun José Lau Gan

1

Preguntas previas

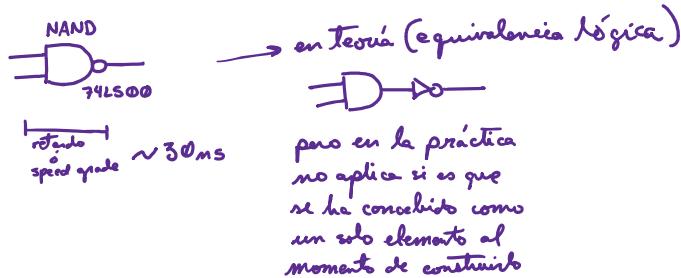
- ¿Los pines del conversor bidireccional 3.3V-5V para qué lado lo sueldo?
 - De preferencia por debajo para que puedas colocarlo en el breadboard
- ¿Para cuando vamos a emplear los materiales?
 - Las sesiones dirigidas con pruebas de circuito en físico a partir de semana 2
- ¿El modelo en 3D del case de la tarjeta requiere de alguna modificación antes de imprimirlo?
 - Ninguna, se descarga el archivo STL (la última versión), se lo llama al software slicer de la impresora y ajustas los parámetros de impresión por defecto (layer height 0.2mm ó 0.3mm, walls 3 á 4, bottom y top layers 3 á 4, sin soportes). Se recomienda material PLA.
- Encontré una fuente de 12V. ¿Se lo podrá usar para alimentar la tarjeta de FPGA?
 - Para periodos muy cortos de tiempo si, pero empezará a calentar los reguladores de voltaje de la tarjeta y si no se tiene cuidado pueden dejar de funcionar.



2

Preguntas previas

- Acerca de retardos:



- ¿El voltaje de I/O del FPGA cuál es?

- El voltaje de los puertos del FPGA para I/O es de 3.3V (LVTTL), si es que se require conectar un periférico que maneja 5V se deberá de emplear el conversor de niveles lógicos 5V/3.3V solicitado en la lista de materiales.

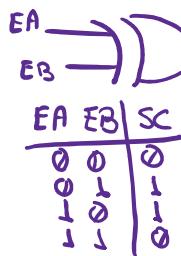
3

Agenda:

- Modelos de circuitos lógicos digitales en VHDL
 - Estilos de descripción en VHDL
- Modelo de decodificador binario en VHDL
- Modelo de decodificador BCD-7SEG en VHDL

4

Modelo de funciones básicas en VHDL en Altera Quartus II



$$SC = \overline{EA} EB + EA \overline{EB}$$

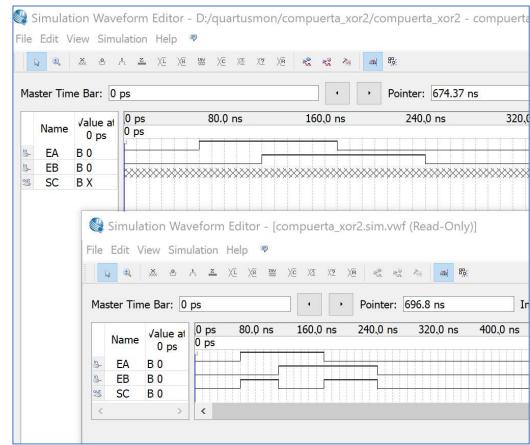
EA	EB	SC
0	0	0
0	1	1
1	0	1
1	1	0

```

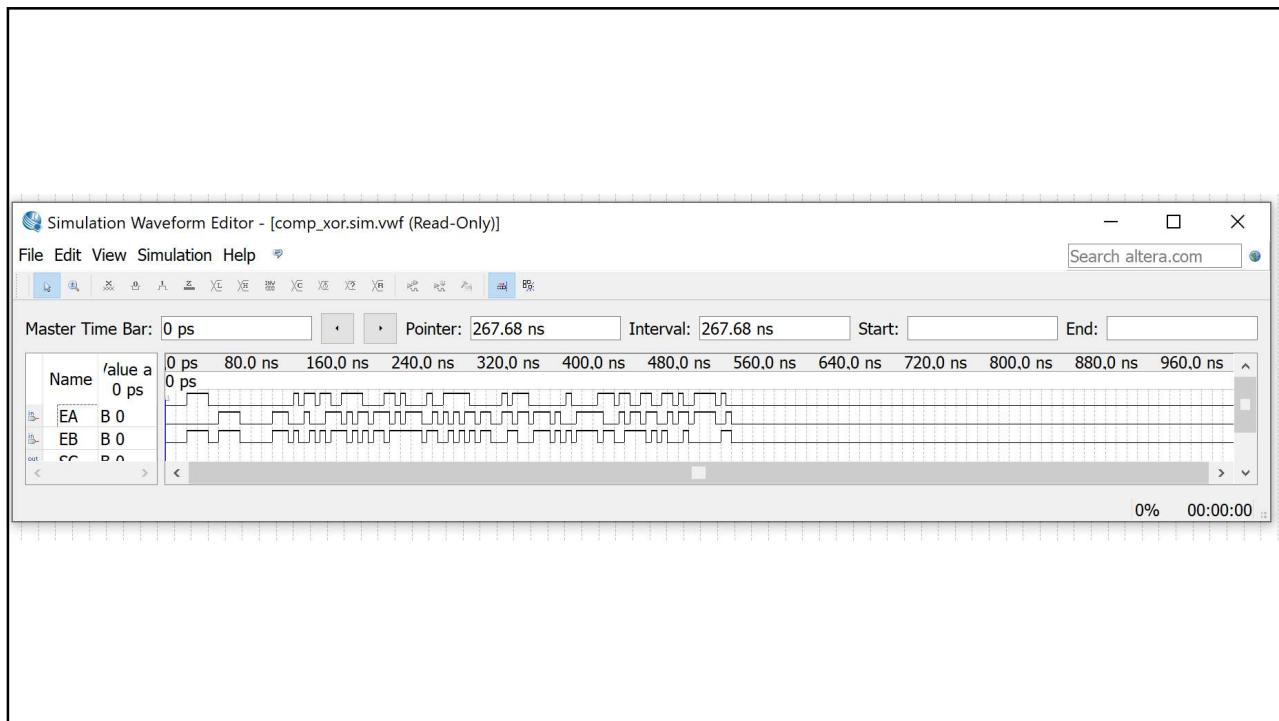
1 --Declaración de librerías IEEE --
2 library IEEE;
3 use IEEE.std_logic_1164.all;
4 use IEEE.std_logic_arith.all;
5 use IEEE.std_logic_unsigned.all;
6
7 --Declaración de la entidad--
8 entity compuerta_xor2 is
9   port(EA, EB :  in std_logic;
10        SC :  out std_logic);
11 end compuerta_xor2;
12
13 --Declaración de la arquitectura--
14 architecture constructo of compuerta_xor2 is
15 begin
16   SC <= EA xor EB;
17 end constructo;
18
19 --architecture constructo2 of compuerta_xor2 is|
20 --signal conexión:  std_logic_vector(1 downto 0);
21 --begin
22 --  conexión <= EA&EB;
23 --  with conexión select
24 --    SC <= '1' when "01",
25 --          '1' when "10",
26 --          '0' when "00",
27 --          '0' when "11",
28 --          'Z' when others;
29 end constructo2;

```

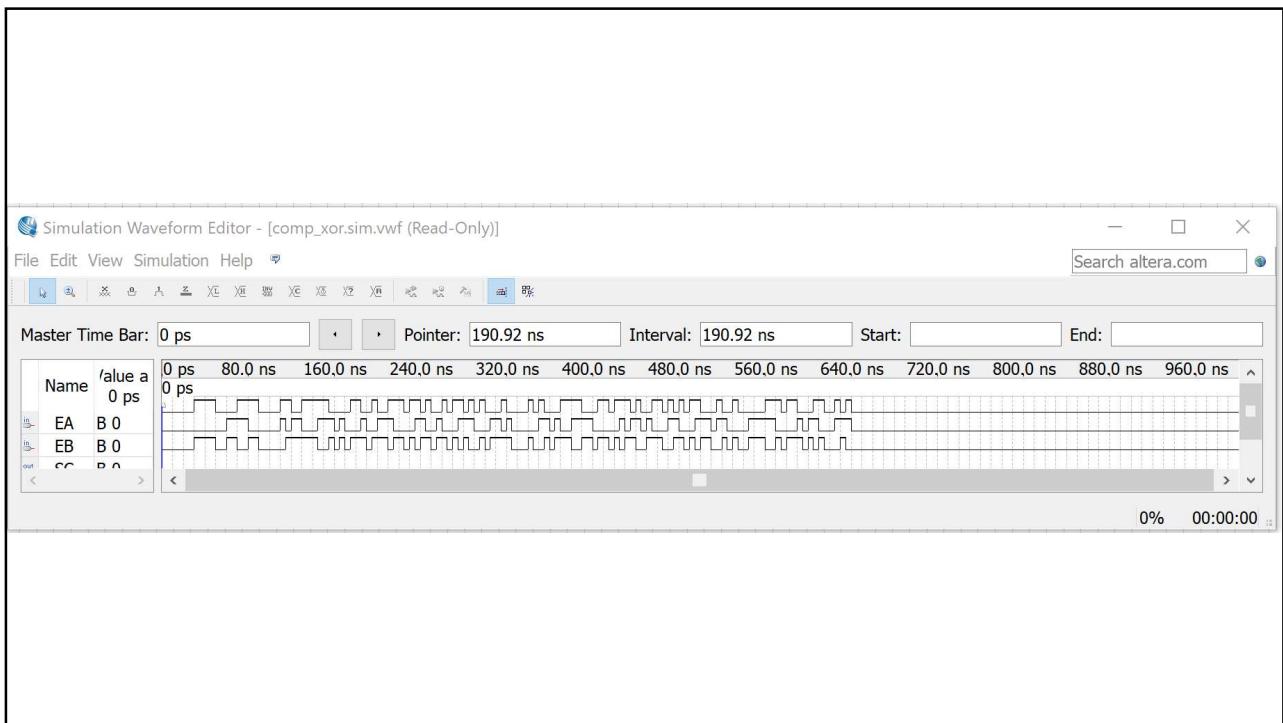
} selectiva (with select)



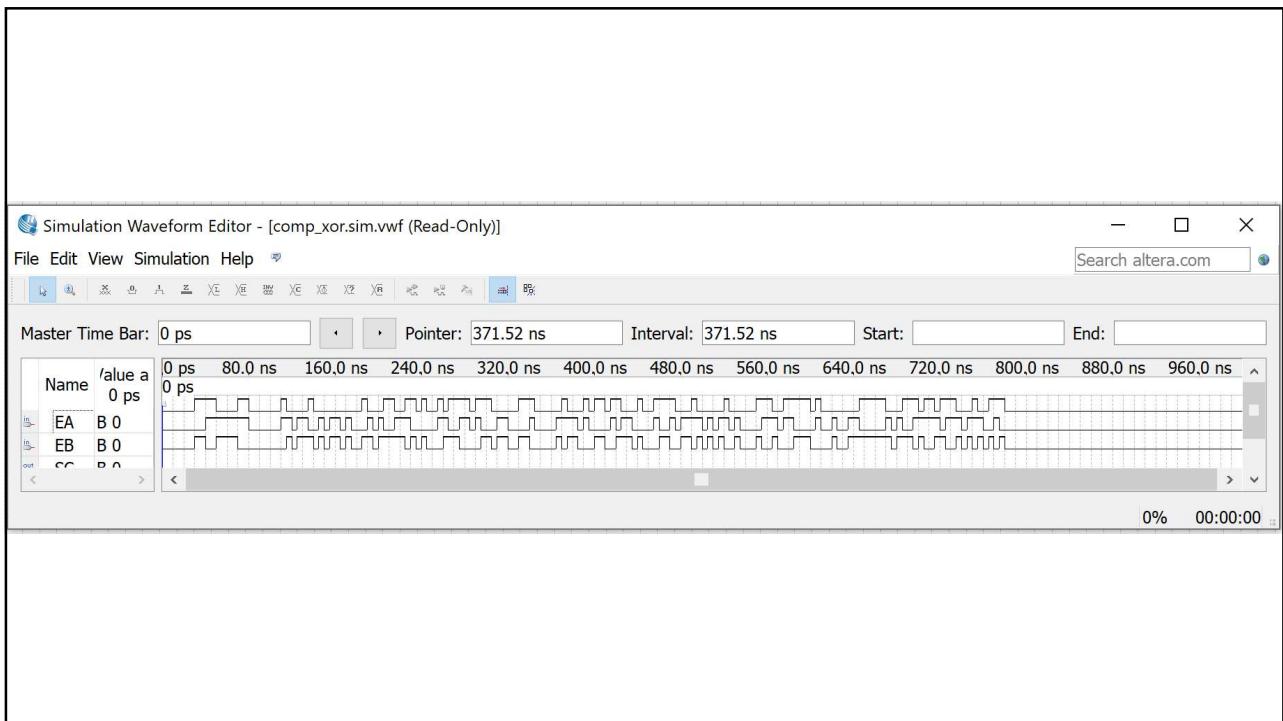
5



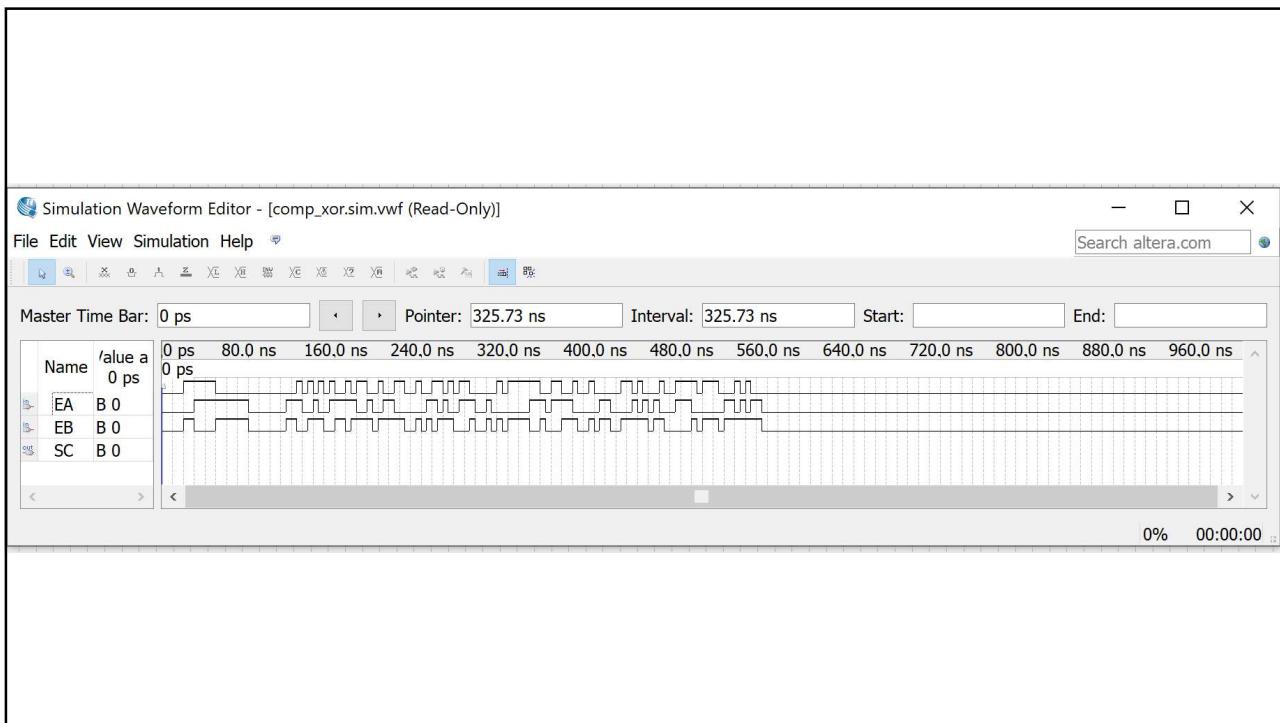
6



7

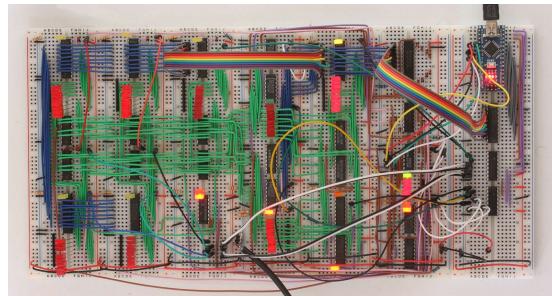
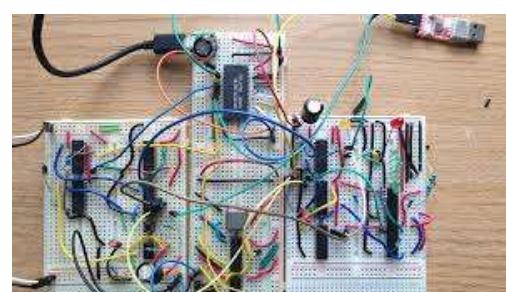
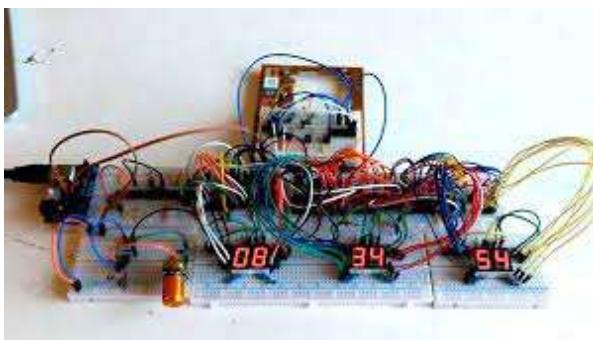


8

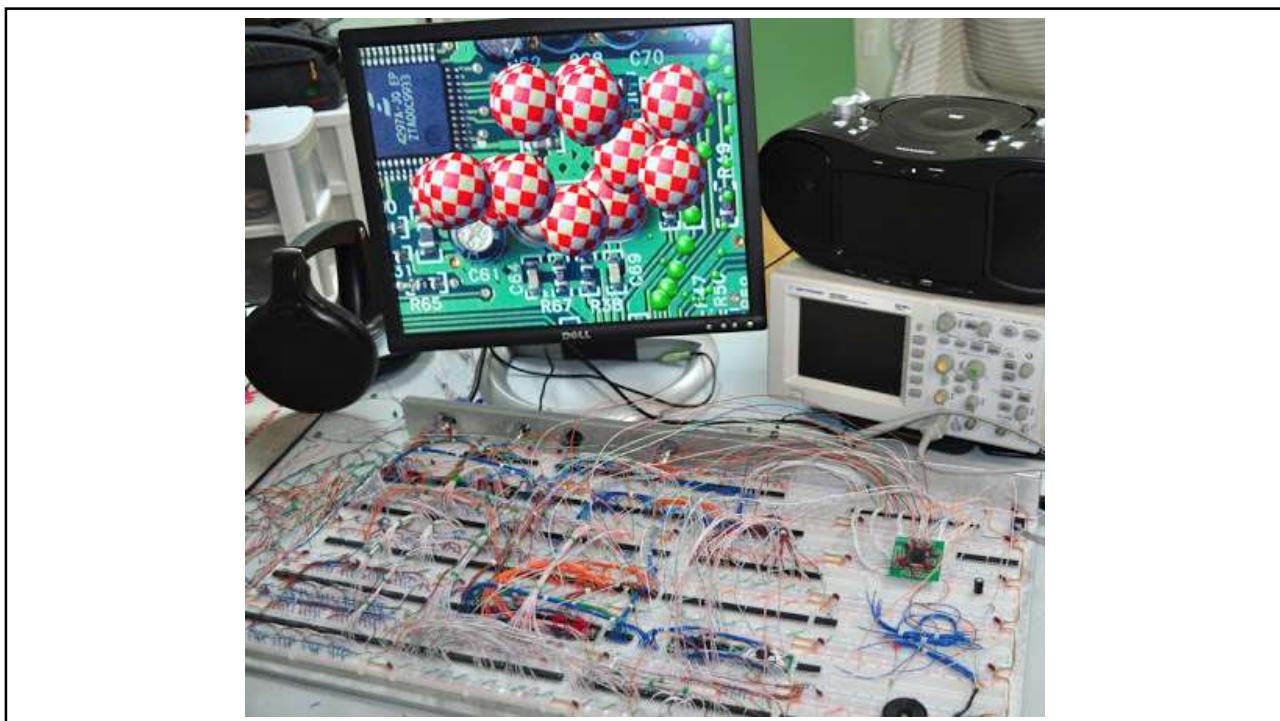


9

En el curso de circuitos lógicos digitales:



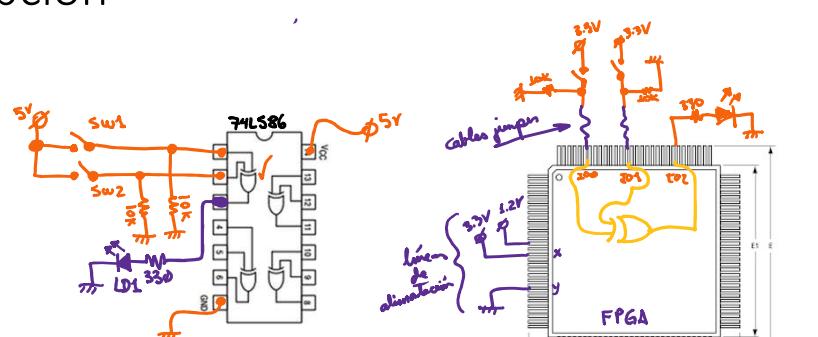
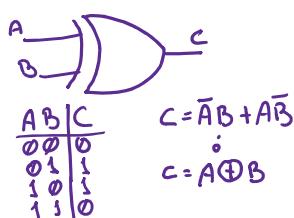
10



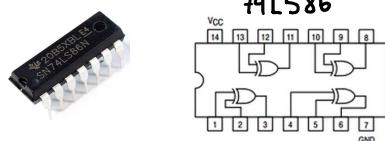
11

Ejemplo: Implementar una compuerta XOR en el EP2C5T144C8, empleando VHDL en sus diferentes estilos de descripción

Análisis:



Implementación con 74xxxx

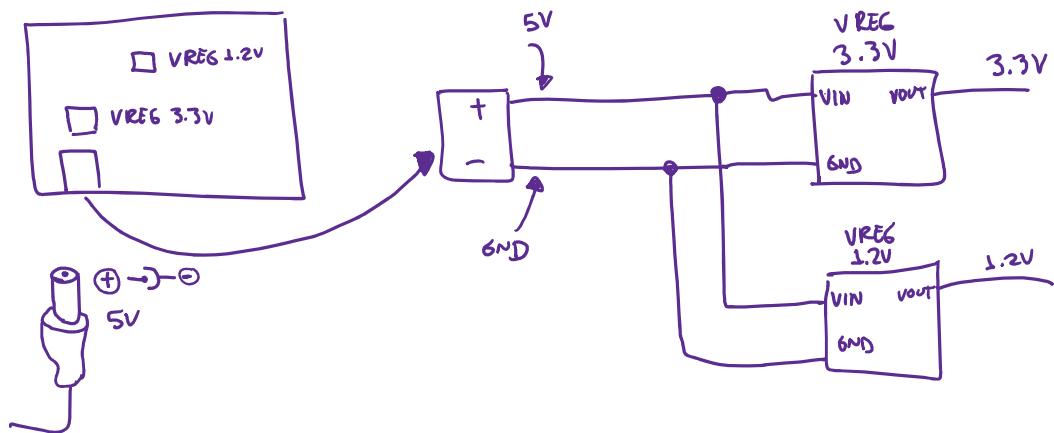


Nota: La familia 74 se considera obsoleta.

En la actualidad

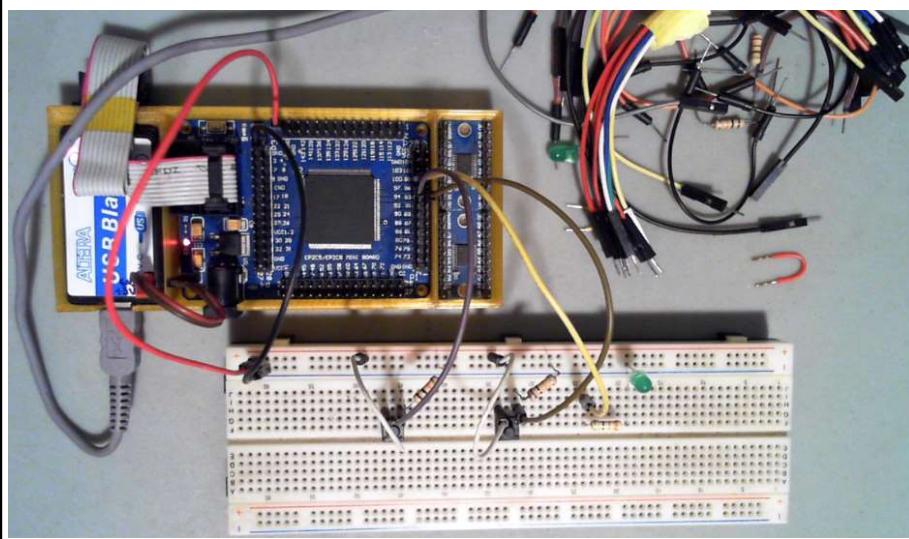
12

Detalle de la alimentación en la Tarjeta de FPGA



13

Circuito implementado en el mundo real



Acerca del LED:



14

Empleando el software Altera Quartus II:

The screenshot shows the Quartus II interface. On the left is the Project Navigator with an Entity named 'xor_vhdl'. The main window shows the VHDL code for a XOR gate. The code includes comments explaining different styles of VHDL description. The simulation waveform editor below shows the inputs IN_A and IN_B, and the output OUT_C, all starting at 0 ps and transitioning to 1 at 80.0 ns.

```

1 --Este es un comentario
2 --Codificado por Kalun
3 --Este programa en VHDL modela una compuerta XOR
4 --en diferentes estilos descriptivos
5
6 --Primera sección: declaración de las librerías
7 library IEEE;
8 use IEEE.std_logic_1164.all;
9 use IEEE.std_logic_arith.all;
10 use IEEE.std_logic_unsigned.all;
11
12 --Segunda sección: declaración de la entidad
13 entity xor_vhdl is
14   port( IN_A, IN_B: in std_logic;
15        OUT_C: out std_logic);
16 end xor_vhdl;

```

```

18 --Tercera sección: declaración de la arquitectura
19 architecture funcionamiento of xor_vhdl is
20 begin
21   --Estilo: Asignación directa
22   --Aclaración: el símbolo <= corresponde a asignación de señal
23   OUT_C <= IN_A xor IN_B;
24 end funcionamiento;
25
26 --Estilo de descripción selectiva
27 architecture funcionamiento of xor_vhdl is
28   signal s_interna: std_logic_vector(1 downto 0);
29 begin
30   s_interna <= IN_A & IN_B;
31   with s_interna select
32     OUT_C <= '0' when "00",
33                  '1' when "01",
34                  '1' when "10",
35                  '0' when "11",
36                  'Z' when others;
37 end funcionamiento;
38
39 --Estilo de descripción condicional
40 architecture funcionamiento of xor_vhdl is
41   signal s_interna: std_logic_vector(1 downto 0);
42 begin
43   s_interna <= IN_A & IN_B;
44   OUT_C <= '0' when s_interna = "00" else
45                  '1' when s_interna = "01" else
46                  '1' when s_interna = "10" else
47                  '0';
48 end funcionamiento;

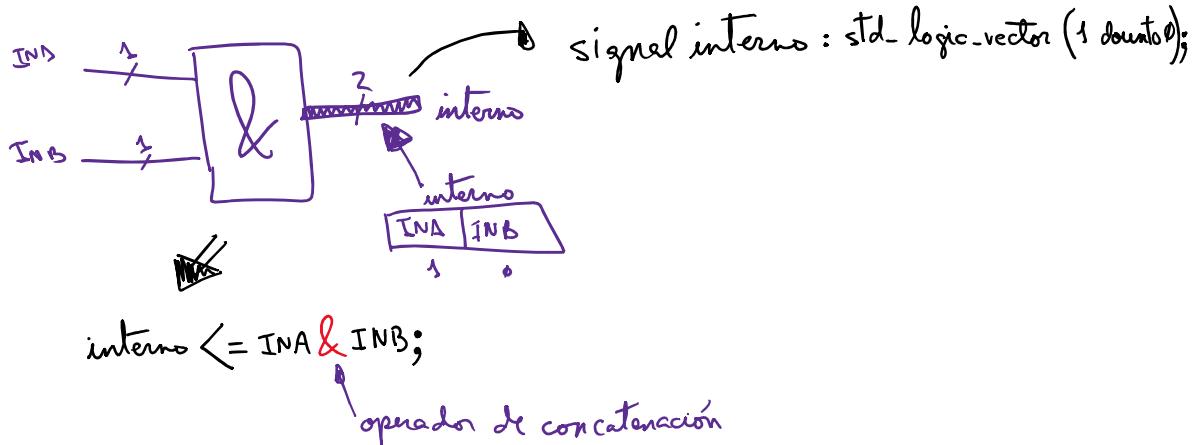
```

Diferentes estilos de descripción emplean solo una!

15

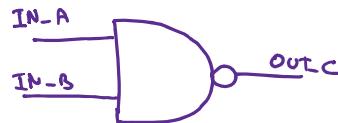
¿Concatenación?

- Unión de señales independientes en un vector

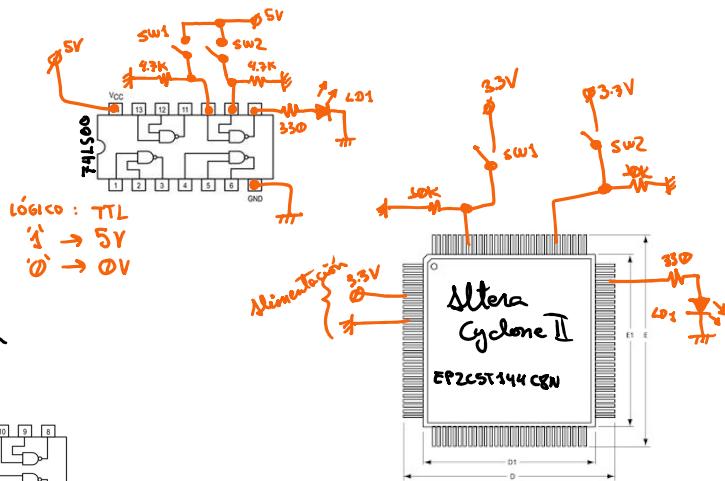


16

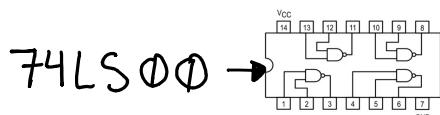
Ejemplo: Implementar una compuerta NAND en el dispositivo EP2C5T144C8



IN-A	IN-B	OUT-C
0	0	1
0	1	1
1	0	1
1	1	0



Empleando tecnología 74 para implementar:



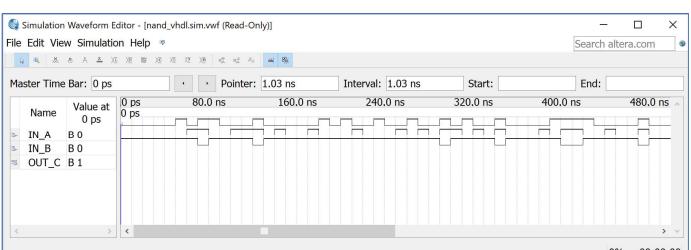
17

Empleando el software Altera Quartus II:

```

5 --Primera sección: declaración de las librerías
6 library IEEE;
7 use IEEE.std_logic_arith.all;
8 use IEEE.std_logic_1164.all;
9 use IEEE.std_logic_unsigned.all;
10
11 --Segunda sección: declaración de la entidad
12 entity nand_vhdl is
13   port( IN_A, IN_B: in std_logic;
14         OUT_C: out std_logic);
15 end nand_vhdl;
16
17 --Tercera sección: declaración de la arquitectura
18 --Utilizando estilo de asignación directa
19 architecture working of nand_vhdl is
20 begin
21   OUT_C <= IN_A nand IN_B;
22 end working;
23
24 --Utilizando estilo de asignación selectiva
25 architecture working of nand_vhdl is
26   signal previo: std_logic_vector(1 downto 0);
27 begin
28   previo <= IN_A & IN_B;
29   with previo select
30     OUT_C <= '1' when "00",
31                 '1' when "01",
32                 '1' when "10",
33                 '0' when "11",
34                 'Z' when others;
35 end working;

```



→ Dos estilos de descripción, emplean solo uno al momento de compilar.

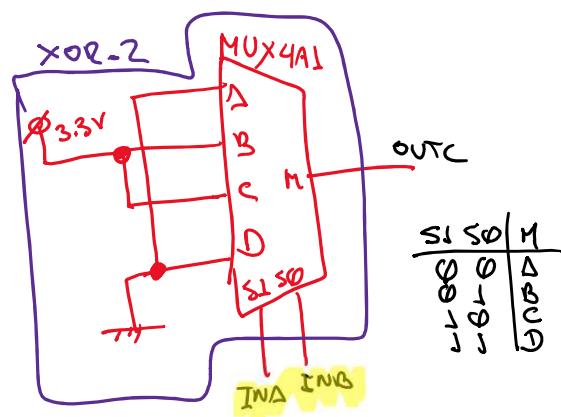
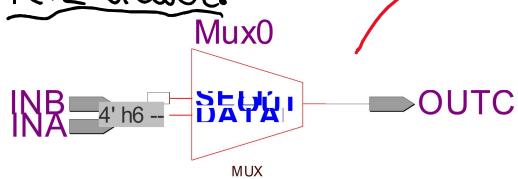
18

¿Qué es lo que interpretó el Quartus?

Hemos descrito el funcionamiento de una XOR

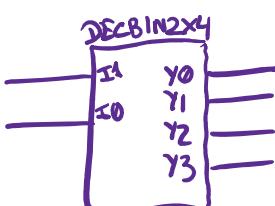


RTL Viewers



19

Modelo de decodificador binario 2x4 en VHDL



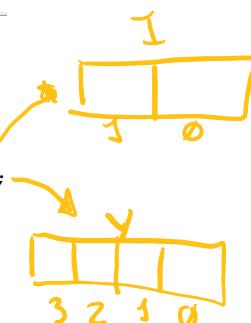
I1	I0	Y3	Y2	Y1	Y0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

Tabla de verdad
Tabla de decodificación

```

1 library IEEE;
2 use IEEE.std_logic_1164.all;
3 use IEEE.std_logic_arith.all;
4 use IEEE.std_logic_unsigned.all;
5
6 entity decbin2x4 is
7   port(I:  in std_logic_vector(1 downto 0);
8        Y:  out std_logic_vector(3 downto 0));
9 end decbin2x4;
10
11 architecture constructo of decbin2x4 is
12 begin
13   with I select
14     Y <= "0001" when "00",
15                  "0010" when "01",
16                  "0100" when "10",
17                  "1000" when "11",
18                  "ZZZZ" when others;
19 end constructo;

```



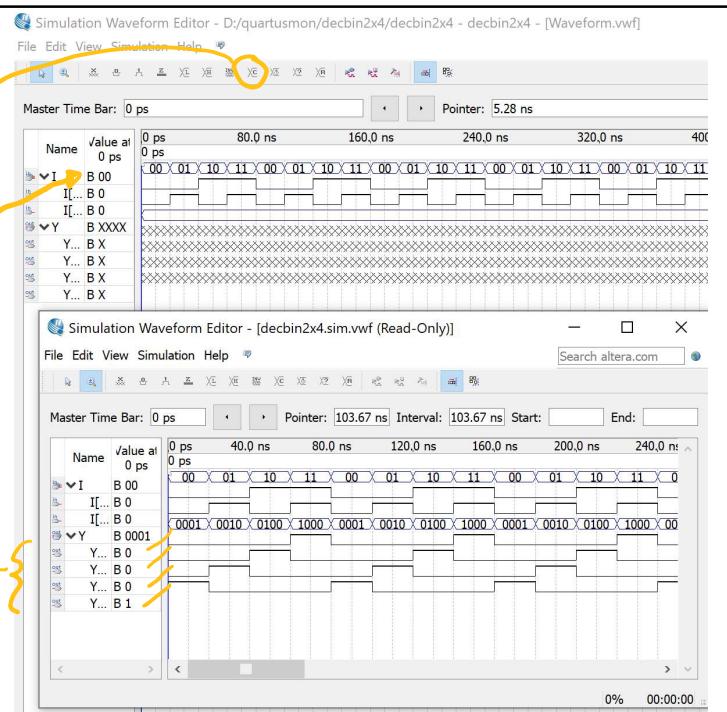
Al usar std_logic_1164
tendremos que
declarar las 2^9
alternativas de selección

20

Simulación funcional del decbin2x4

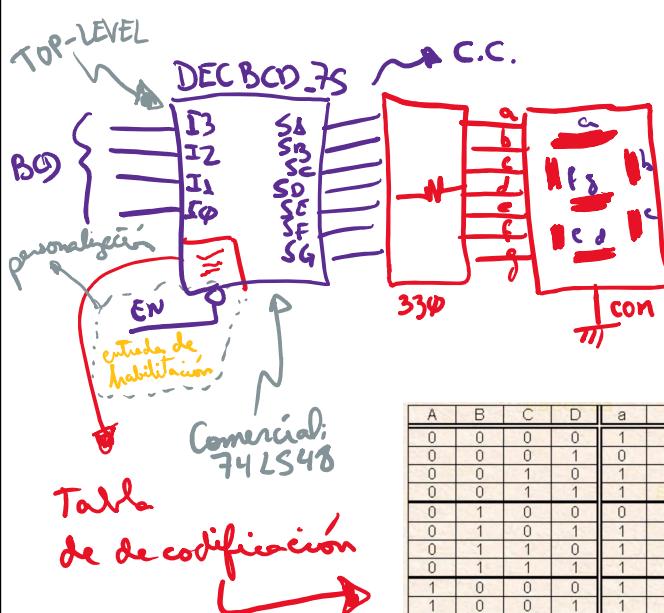
Señal de cuenta generada

Salida Simulada

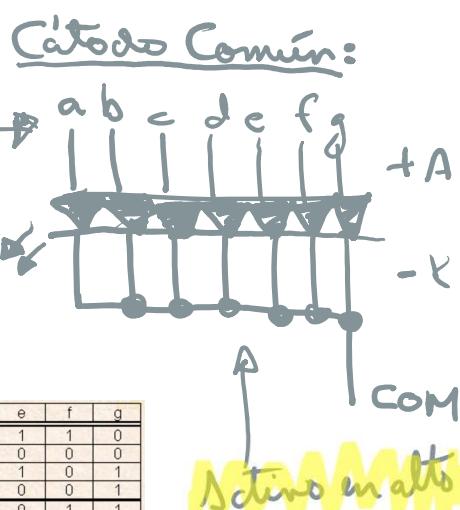


21

Modelo de decodificador BCD-7S en VHDL



A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	1	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	1	0	0	1
0	1	0	1	1	0	1	1	1	0	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	1	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	0	0	1	1



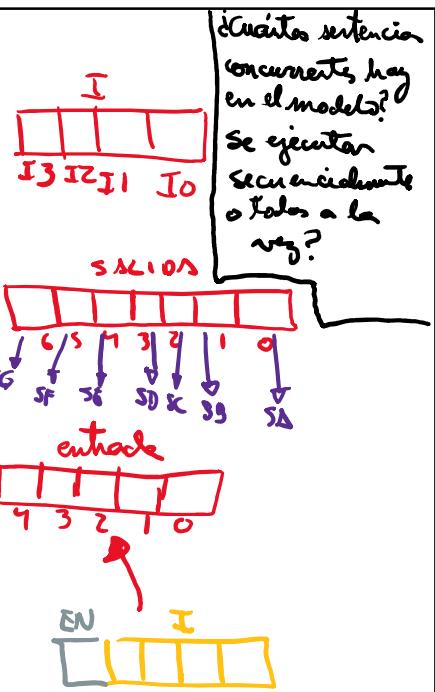
22

Modelo del decbcd_7s en VHDL

```

6 entity decbcd_7s is
7 port( I : in std_logic_vector(3 downto 0);
8      SA, SB, SC, SD, SE, SF, SG : out std_logic);
9      EN : in std_logic); --entrada de habilitación--
10 end decbcd_7s;
11
12 architecture constructo of decbcd_7s is
13 signal salida : std_logic_vector(6 downto 0);
14 signal entrada : std_logic_vector(4 downto 0);
15 begin
16     entrada <= EN&I;
17     SA <= salida(0);
18     SB <= salida(1);
19     SC <= salida(2);
20     SD <= salida(3);
21     SE <= salida(4);
22     SF <= salida(5);
23     SG <= salida(6);
24     with entrada select
25         salida <= "0111111" when "00000";
26         "0000110" when "00001";
27         "1011011" when "00010";
28         "1001111" when "00011";
29         "1100110" when "00100";
30         "1101101" when "00101";
31         "1111101" when "00110";
32         "0000111" when "00111";
33         "1111111" when "01000";
34         "1100111" when "01001";
35         "ZZZZZZZ" when others;
36 end constructo;

```



23

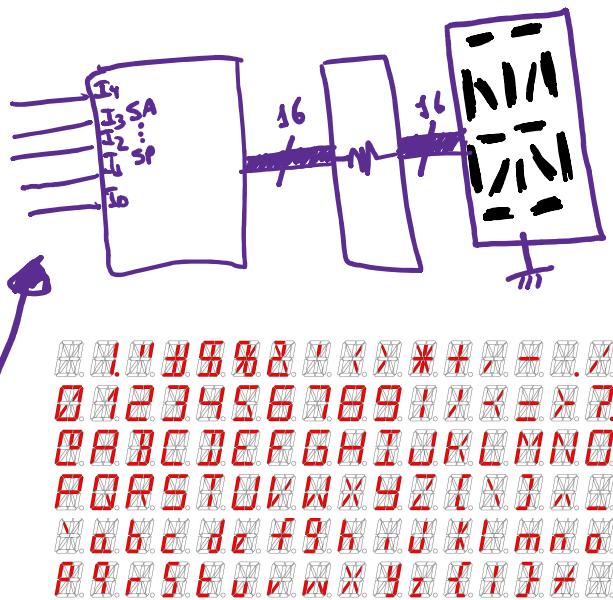
Pendientes:

- Verificar en simulación el funcionamiento del decodificador binario DEC2x4
- Cambiar el estilo de descripción del DEC2x4 hacia estilo de descripción condicional
- Verificar en simulación el funcionamiento del decodificador BCD-7SEG visto anteriormente.
- Agregarle al decodificador anterior una entrada adicional para que se puede cambiar entre display de cátodo común y ánodo común.

24

Cuestionario:

- Cuando se escoge el dispositivo EP2C5T144C8, ¿“C8” qué significado tiene?
- Desarrollar y simular un codificador binario de prioridad 8x3 en VHDL
- Desarrollar un decodificador en VHDL para el siguiente circuito (Solo para las letras mayúsculas del mapa de caracteres mostrado)



25

Fin de la sesión

26