

Laboratorio de implementación

Con labsland

FERNEY ALBERTO BELTRAN



Objetivo

El objetivo es la implementación en una FPGA remota del HDL del display de 7 segmentos, en este sentido el estudiante se debe familiarizar con el entorno de labslab.

Proceso del Laboratorio



* Debe ingresar a la cuenta de la pagina con la invitacion enviada por correo electronico , ingrese su usuario y clave.

- Ingrese al IDE del laboratorio
- Cargue el archivo bcd2sseg.v dado en el laboratorio anterior , y realice los cambio pertinentes segun se explico en clase
- Genere la sintetización del HDL.
- Revise si se completo la sintetización y de ser asi “envie al dispositivo”
- Pruebe la funcionalidad del sistema

Una vez m termine haga el mismo procedimiento para 4 display

El IDE

[Nuevo](#) [+ Añadir](#) [Descargar](#)

blink.vhd  

Top level entity:

blink.vhd

Documentación

- Asignación de señales
- DE1-SoC.qsf
- Ejemplos de VHDL de Intel

Ejemplos

- LEDs Mirror
- Blink
- Clock (7-segments)

El programa fue correctamente verificado y compilado (16:55:19).

[Información](#) [Síntetizar](#) [Enviar al dispositivo](#) Todos los cambios guardados

```
2 -- Blink example
3 --
4 -- Using the button 0 (V_BT(0)) as a reset, this
5 -- code makes the first LED (G_LED(0)) blink
6 --
7 library ieee;
8 use ieee.std_logic_1164.all;
9 use ieee.numeric_std.all;
10
11 entity blink is
12     port (
13         G_CLOCK_50: in std_logic; --50MHz
14         V_BT: in std_logic_vector (0 downto 0);
15         G_LED: out std_logic_vector (0 downto 0)
16     );
17 end;
18
19 architecture behav of blink is
20     signal count : unsigned(31 downto 0) := (others => '0');
21     signal brightness : std_logic;
22     signal reset : std_logic;
23     constant zero : unsigned(31 downto 0) := X"00000000";
24 begin
25     G_LED(0) <= brightness;
26     reset <= V_BT(0);
27
28     process(G_CLOCK_50, reset)
29     begin
30         if rising_edge(G_CLOCK_50) then
```

consola [blink.fit.summary](#) [blink.map.summary](#)

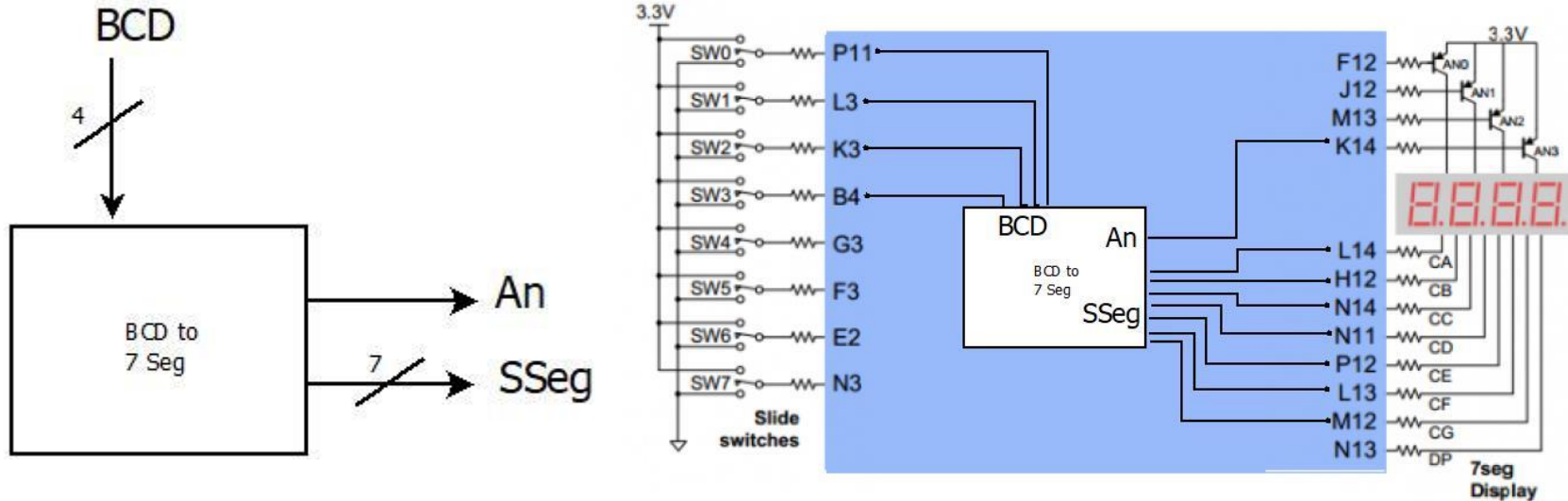
```
$ quartus_map blink --source COMPILATION_DIRECTORY/blink.vhd --family "Cyclone V" --part 5C5EMA5F31C6
Info: *****
Info: Running Quartus Prime Analysis & Synthesis
Info: Version 17.1.0 Build 590 10/25/2017 S3 Lite Edition
Info: Copyright (C) 2017 Intel Corporation. All rights reserved.
Info: Your use of Intel Corporation's design tools, logic functions
Info: and other software and tools, and its AMPP partner logic
Info: functions, and any output files from any of the foregoing
Info: (including device programming or simulation files), and any
Info: associated documentation or information are expressly subject
Info: to the terms and conditions of the Intel Program License
```

El IDE

- **Control de archivos**: Arriba a la izquierda. Aquí pueden añadirse archivos fuente, elegir cuáles editar, borrar archivos etc.
- **Documentación y ejemplos**: Abajo a la izquierda. Aquí puedes acceder a diferentes documentos y diagramas describiendo las entradas/salidas disponibles; así como diversos ejemplos de código.
- **Editor de código**: En la parte central. Permite modificar el archivo fuente actualmente seleccionado. Dispone de funciones básicas de IDE, tal como autocompletado de ciertas construcciones, resaltado de sintaxis, etc. También detectará de forma automática algunos errores, indicándose a la izquierda.
- **Salida y terminales**: Bajo el editor de código. Mostrará los resultados de las diferentes etapas de sintetización. Será particularmente útil en caso de que exista algún error, ya que se indicará información sobre éste, así como sobre el archivo y líneas en las que se ha producido.

implementación

Para la implementación física del diseño, se debe realizar el mapeo de los puertos de entrada y salida del componente diseñado, en este caso sumador de 4 bit, con los pines físicos de la FPGA que alojara el diseño electrónico.



implementación

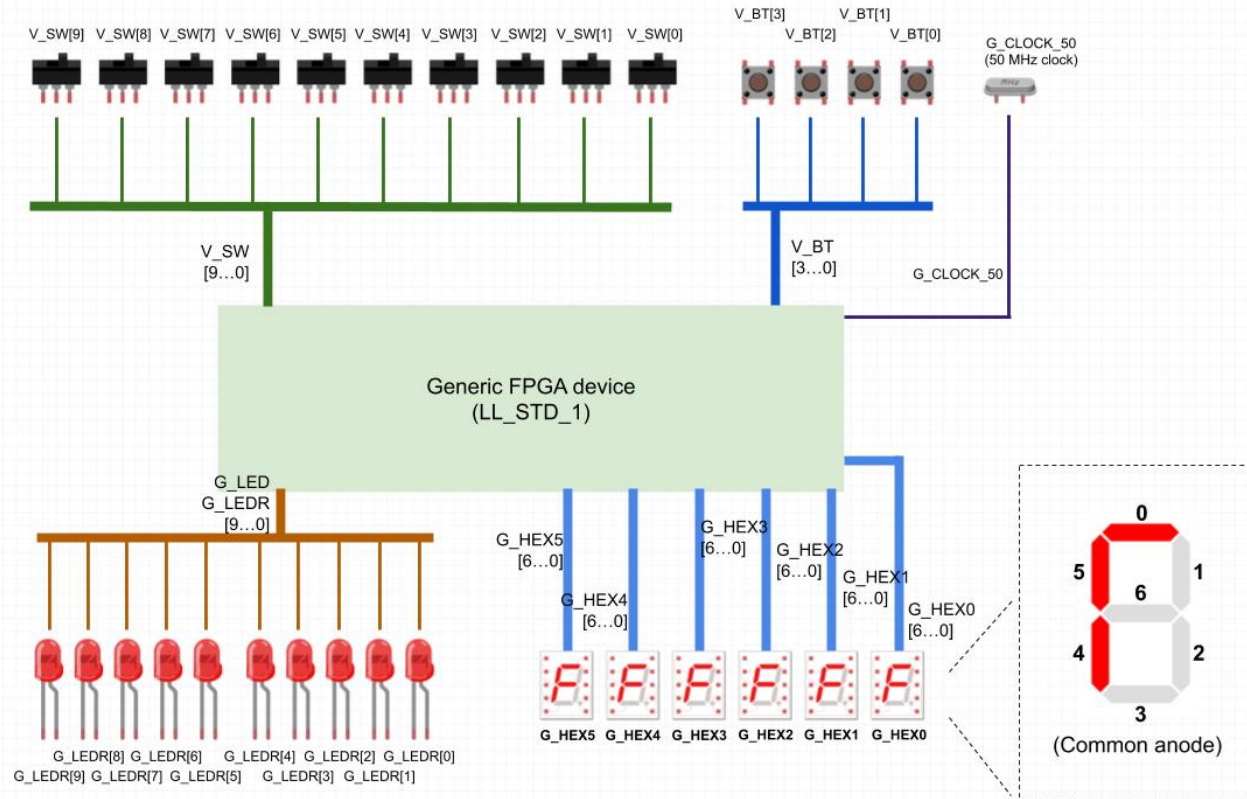


Diagrama con
periféricos elementales
de entrada/salida
disponibles.

Código inicial

Archivo: bcd2sseg.v

```
module BCDtoSseg (BCD, SSeg, an);

    input [3:0] BCD;
    output reg [0:6] SSeg;
    output [3:0] an;

    assign an=4'b1110;

    always @ ( * ) begin
        case (BCD)
            4'b0000: SSeg = 7'b0000001; // "0"
            4'b0001: SSeg = 7'b1001111; // "1"
            4'b0010: SSeg = 7'b0010010; // "2"
            4'b0011: SSeg = 7'b0000110; // "3"
            4'b0100: SSeg = 7'b1001100; // "4"
            4'b0101: SSeg = 7'b0100100; // "5"
            4'b0110: SSeg = 7'b0100000; // "6"
            4'b0111: SSeg = 7'b0001111; // "7"
            4'b1000: SSeg = 7'b0000000; // "8"
            4'b1001: SSeg = 7'b0000100; // "9"
            4'ha: SSeg = 7'b0001000;
            4'hb: SSeg = 7'b1100000;
            4'hc: SSeg = 7'b0110001;
            4'hd: SSeg = 7'b1000010;
            4'he: SSeg = 7'b0110000;
            4'hf: SSeg = 7'b0111000;
            default:
                SSeg = 0;
        endcase
    end
endmodule
```



```
module bcd2sseg (V_SW, G_HEX0);

    input wire [3:0] V_SW;
    output wire [6:0] G_HEX0;

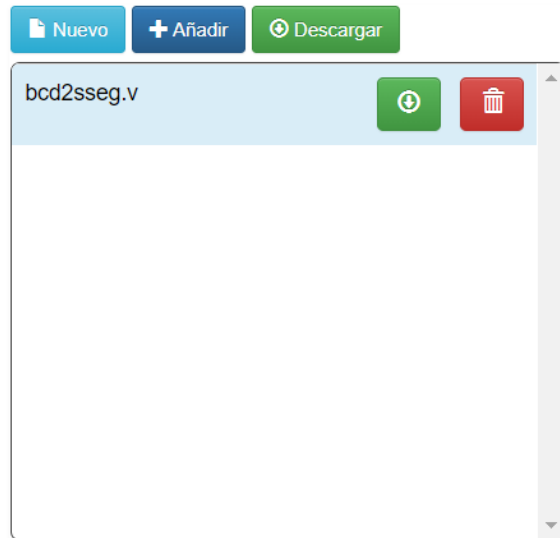
    reg [6:0] SSeg;
    wire [3:0] BCD;

    assign BCD =V_SW;
    assign G_HEX0 = SSeg;

    always @ ( * ) begin
        case (BCD)
            4'b0000: SSeg = 7'b1000000; // "0"
            4'b0001: SSeg = 7'b1111001; // "1"
            4'b0010: SSeg = 7'b0100100; // "2"
            4'b0011: SSeg = 7'b0110000; // "3"
            4'b0100: SSeg = 7'b0011001; // "4"
            4'b0101: SSeg = 7'b0010010; // "5"
            4'b0110: SSeg = 7'b0000010; // "6"
            4'b0111: SSeg = 7'b1111000; // "7"
            4'b1000: SSeg = 7'b0000000; // "8"
            4'b1001: SSeg = 7'b0011000; // "9"
            4'ha: SSeg = 7'b0001000;
            4'hb: SSeg = 7'b0000011;
            4'hc: SSeg = 7'b0100111;
            4'hd: SSeg = 7'b0100001;
            4'he: SSeg = 7'b0000100;
            4'hf: SSeg = 7'b0001110;
            default:
                SSeg = 0;
        endcase
    end
endmodule
```


La FPGA

El programa fue correctamente verificado y compilado (21:00:44).

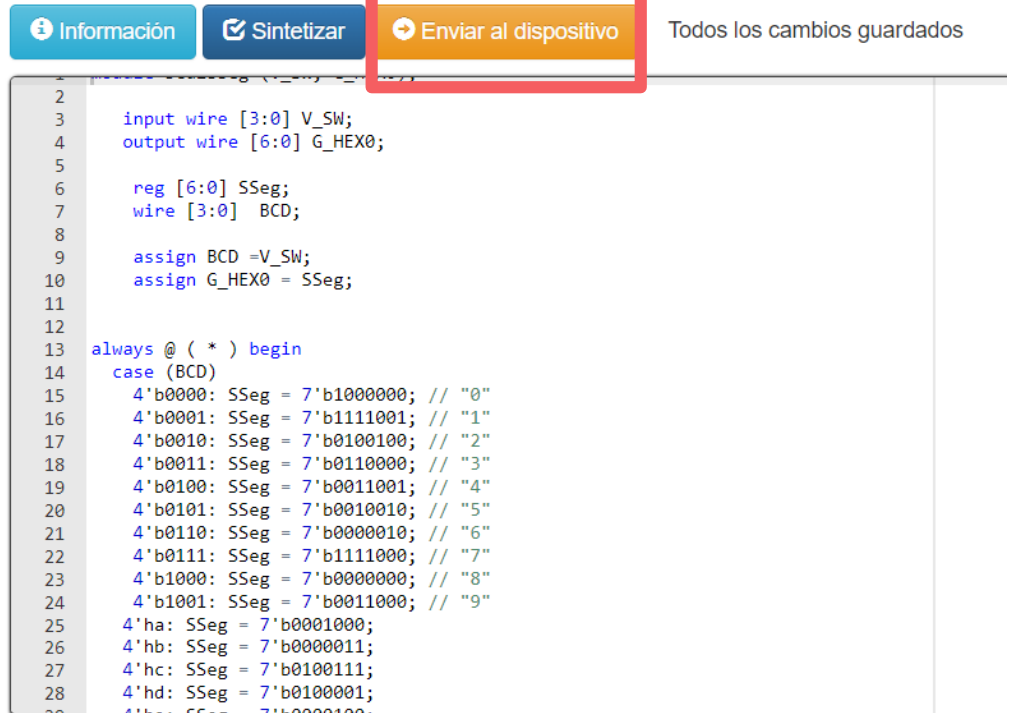


Top level entity:

bcd2sseg.v

Documentación

- Nombres de señales IO
- DE1-SoC.qsf
- Ejemplos de verilog de Intel



consola

bcd2sseg.fit.summary

bcd2sseg.map.summary

```
$ quartus_map bcd2sseg --source COMPILATION_DIRECTORY/bcd2sseg.v --family "Cyclone V"
Info: *****
Info: Running Quartus Prime Analysis & Synthesis
```

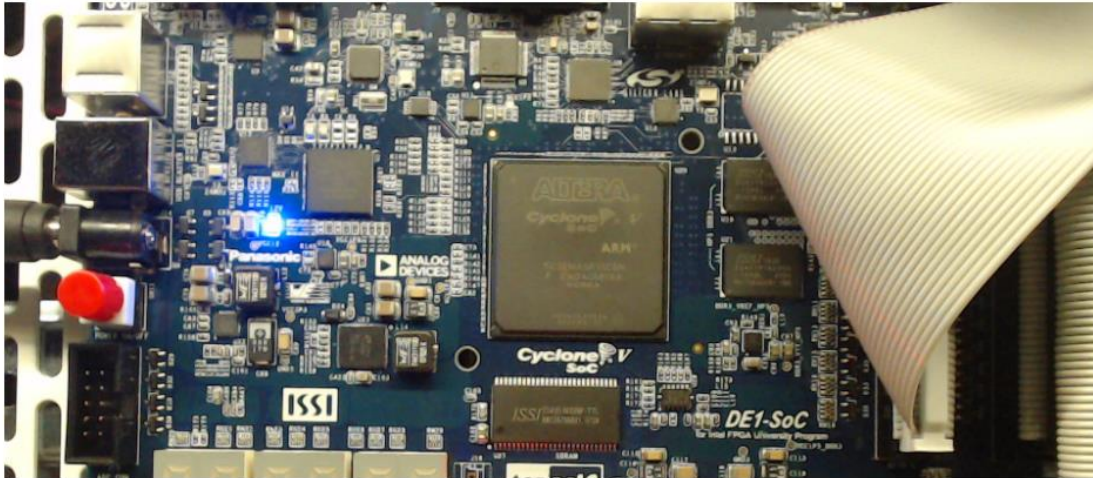
La FPGA



01:42

Salir ahora

Laboratorio Altera FPGA



Tus propios programas

Tu programa del IDE FPGA

Este es el último programa que has preparado en el IDE FPGA

Programa en la FPGA

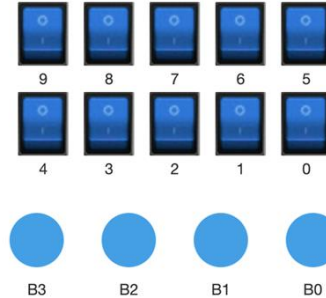
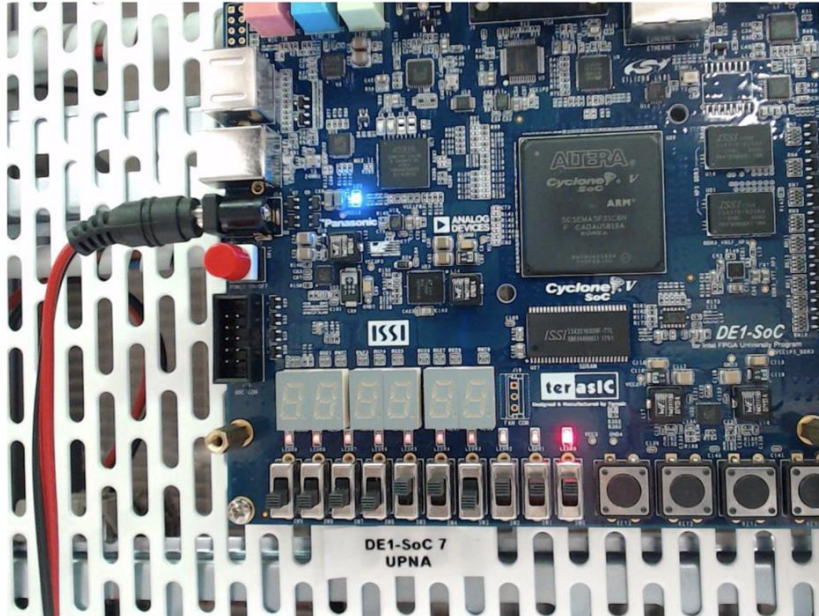
Programas de demostración

La FPGA



Salir ahora

Laboratorio Altera FPGA



La FPGA

- **Video en tiempo real**: Central a la izquierda. Muestra un video en tiempo real de la FPGA que estás controlando. De este modo es posible visualizar las salidas de la placa, incluyendo LEDs, displays 7 segmentos, y otras.
- **Controles**: A la derecha. Inicialmente, permiten elegir un programa a ser grabado. Normalmente el programa “hecho por el usuario” previamente en el IDE. Posteriormente, muestran botones e interruptores aparentemente virtuales (como en la figura). Se pueden pulsar y se puede interactuar con la placa a través de ellos. Al pulsarlos, se transmitirá una entrada real correspondiente a la placa.

Verificar

Verifica que funciona:

- Comprueba la lógica en la FPGA real.
- Introduce cada uno de los números posibles mediante los interruptores, y verifica que se muestran todos correctamente, desde el 0 hasta el 15.