Laboratorio 02: El decodificador BCD/7-segmentos

Por:
Ian Gabriel Cañas Fernández, 1092228

.
.

Decodificador BCD/7-Segmentos:
Mostrando números en un display

Objetivos:
Conocer y comprobar el funcionamiento de los decodificadores 7-segmentos e implementarlo en el kit de desarrollo de la FPGA.

Procedimiento:
En el presente experimento se procura generar un código que será aplicado a la FPGA que, va a ser capaz de recibir como entrada 4 bits que representarán los números del 0 al 15 en binario, que serán presentados en un display 7 segmentos con los decimales del 0 al 9 correspondientes a dichos números y con las letras de la A a la F.

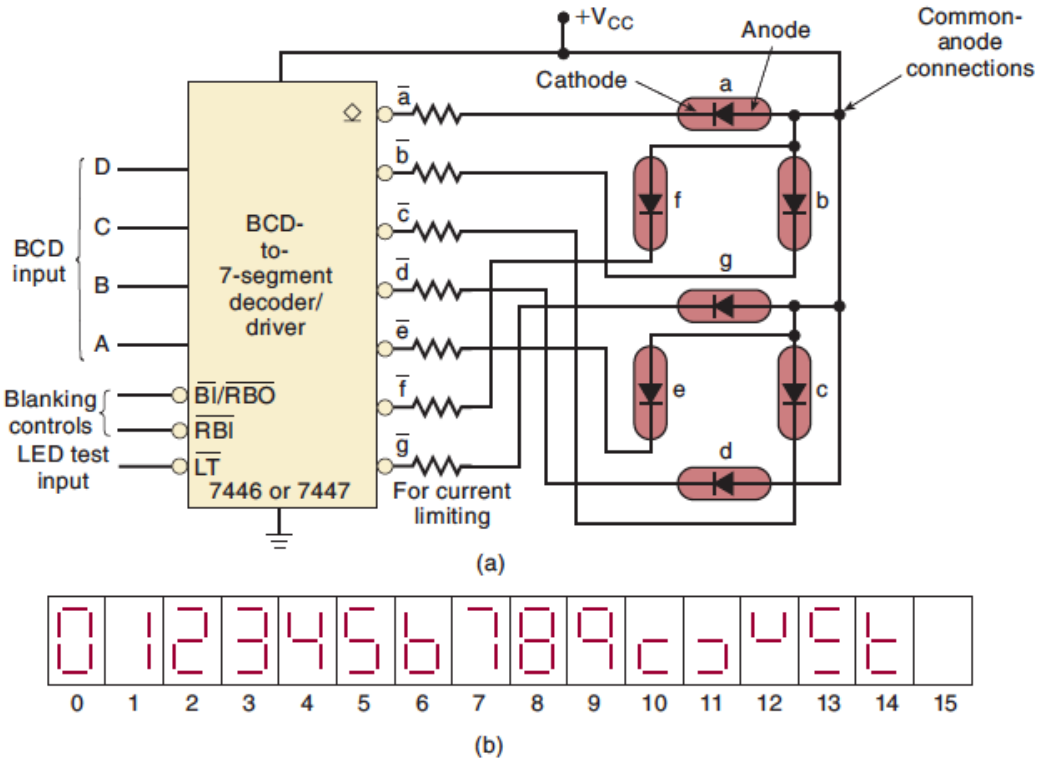


Figura 1. (a) Decodificador/controlador de BCD a 7 segmentos con ánodo común; (b) patrones de segmentos para todos los posibles códigos de entrada.

El display recibirá un vector lógico con 7 bits en el que a cada uno se le asignará un segmento del display que, dependiendo del estado del bit, este estará encendido o apagado, tomando en cuenta que, en un display de ánodo común, los segmentos se activan con un cero.

A continuación, se presenta la definición del circuito que va a representar dicho comportamiento junto a su respectivo *constraint*:

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

entity Laboratorio02 is
 Port (
        NumBin: in std_logic_vector (3 downto 0);
        NumDec: OUT std_logic_vector (6 DOWNTO 0);
        an: OUT std_logic_vector (7 DOWNTO 0)

        );
end Laboratorio02;


architecture Behavioral of Laboratorio02 is

begin
    with NumBin select
        -- Los 7 dígitos de NumDec corresponden a las salidas abcdefg respectivamente
    NumDec <=    ("0000001") WHEN ("0000"),
             ("1001111") WHEN ("0001"),
             ("0010010") WHEN ("0010"),
             ("0000110") WHEN ("0011"),
             ("1001100") WHEN ("0100"),
             ("0100100") WHEN ("0101"),
             ("0100000") WHEN ("0110"),
             ("0001111") WHEN ("0111"),
             ("0000000") WHEN ("1000"),
             ("0000100") WHEN ("1001"),
             ("0001000") WHEN ("1010"),
             ("1100000") WHEN ("1011"),
             ("0110001") WHEN ("1100"),
             ("1000010") WHEN ("1101"),
             ("0110000") WHEN ("1110"),
             ("0111000") WHEN OTHERS;

    an <= "11111110";
end Behavioral;
```

Constraint:

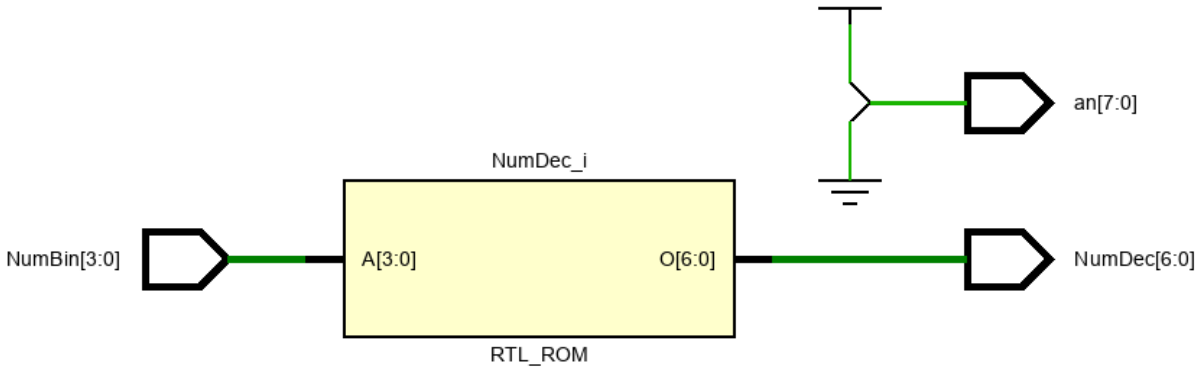
```
##Switches
set_property -dict { PACKAGE_PIN J15   IOSTANDARD LVCMOS33 } [get_ports { NumBin[0] }]; #IO_L24N_T3_RS0_15 Sch=sw[0]
set_property -dict { PACKAGE_PIN L16   IOSTANDARD LVCMOS33 } [get_ports { NumBin[1] }]; #IO_L3N_T0_DQS_EMCCLK_14 Sch=sw[1]
set_property -dict { PACKAGE_PIN M13   IOSTANDARD LVCMOS33 } [get_ports { NumBin[2] }]; #IO_L6N_T0_D08_VREF_14 Sch=sw[2]
set_property -dict { PACKAGE_PIN R15   IOSTANDARD LVCMOS33 } [get_ports { NumBin[3] }]; #IO_L13N_T2_MRCC_14 Sch=sw[3]


##7 segment display
set_property -dict { PACKAGE_PIN T10   IOSTANDARD LVCMOS33 } [get_ports { NumDec[6] }]; #IO_L24N_T3_A00_D16_14 Sch=ca
set_property -dict { PACKAGE_PIN R10   IOSTANDARD LVCMOS33 } [get_ports { NumDec[5]  }]; #IO_25_14 Sch=cb
set_property -dict { PACKAGE_PIN K16   IOSTANDARD LVCMOS33 } [get_ports { NumDec[4] }]; #IO_25_15 Sch=cc
set_property -dict { PACKAGE_PIN K13   IOSTANDARD LVCMOS33 } [get_ports { NumDec[3] }]; #IO_L17P_T2_A26_15 Sch=cd
set_property -dict { PACKAGE_PIN P15   IOSTANDARD LVCMOS33 } [get_ports { NumDec[2] }]; #IO_L13P_T2_MRCC_14 Sch=ce
set_property -dict { PACKAGE_PIN T11   IOSTANDARD LVCMOS33 } [get_ports { NumDec[1] }]; #IO_L19P_T3_A10_D26_14 Sch=cf
set_property -dict { PACKAGE_PIN L18   IOSTANDARD LVCMOS33 } [get_ports { NumDec[0] }]; #IO_L4P_T0_D04_14 Sch=cg
#set_property -dict { PACKAGE_PIN H15   IOSTANDARD LVCMOS33 } [get_ports { NumDec[0] }]; #IO_L19N_T3_A21_VREF_15 Sch=dp
set_property -dict { PACKAGE_PIN J17   IOSTANDARD LVCMOS33 } [get_ports { an[0] }]; #IO_L23P_T3_FOE_B_15 Sch=an[0]
set_property -dict { PACKAGE_PIN J18   IOSTANDARD LVCMOS33 } [get_ports { an[1] }]; #IO_L23N_T3_FWE_B_15 Sch=an[1]
set_property -dict { PACKAGE_PIN T9    IOSTANDARD LVCMOS33 } [get_ports { an[2] }]; #IO_L24P_T3_A01_D17_14 Sch=an[2]
set_property -dict { PACKAGE_PIN J14   IOSTANDARD LVCMOS33 } [get_ports { an[3] }]; #IO_L19P_T3_A22_15 Sch=an[3]
set_property -dict { PACKAGE_PIN P14   IOSTANDARD LVCMOS33 } [get_ports { an[4] }]; #IO_L8N_T1_D12_14 Sch=an[4]
set_property -dict { PACKAGE_PIN T14   IOSTANDARD LVCMOS33 } [get_ports { an[5] }]; #IO_L14P_T2_SRCC_14 Sch=an[5]
set_property -dict { PACKAGE_PIN K2    IOSTANDARD LVCMOS33 } [get_ports { an[6] }]; #IO_L23P_T3_35 Sch=an[6]
set_property -dict { PACKAGE_PIN U13   IOSTANDARD LVCMOS33 } [get_ports { an[7] }]; #IO_L23N_T3_A02_D18_14 Sch=an[7]
```

Resultados:

Cargando lo anterior en la FPGA se tienen los resultados esperados. Precisamente los siguientes valores:

NumBin	Número presentado
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	b
1100	C
1101	d
1110	E
1111	F

A parte, el programa genera el siguiente diagrama, en el que se aprecia que el valor de an ha sido independizado del sistema.



Análisis:
Luego de la práctica de este laboratorio hemos comprobado que la FPGA responde con claridad ante las variantes asignadas de manera tal de que la combinación de diodos encendidos muestre el símbolo correspondiente.

Hemos aprendido cómo configurar y trabajar con un 7 segmento y cómo adaptar su comportamiento para que muestre ciertos símbolos. El diseño no es "mejorable" del todo, pero si es modificable para que la respuesta presentada sea resultado de funciones lógicas.