

Lab 3: Hexadecimal digit to seven-segment decoder

Objectives

The purpose of this laboratory exercise is to design a 7-segment display decoder and to become familiar with the VHDL structural description that allows you to build a larger system from simpler or predesigned components.

Materials

You will use push buttons and slide switches on the CoolRunner-II CPLD starter board (XC2C256-TQ144, manual, schematic) as inputs and light emitting diodes (LEDs) and 7-segment display as output devices.

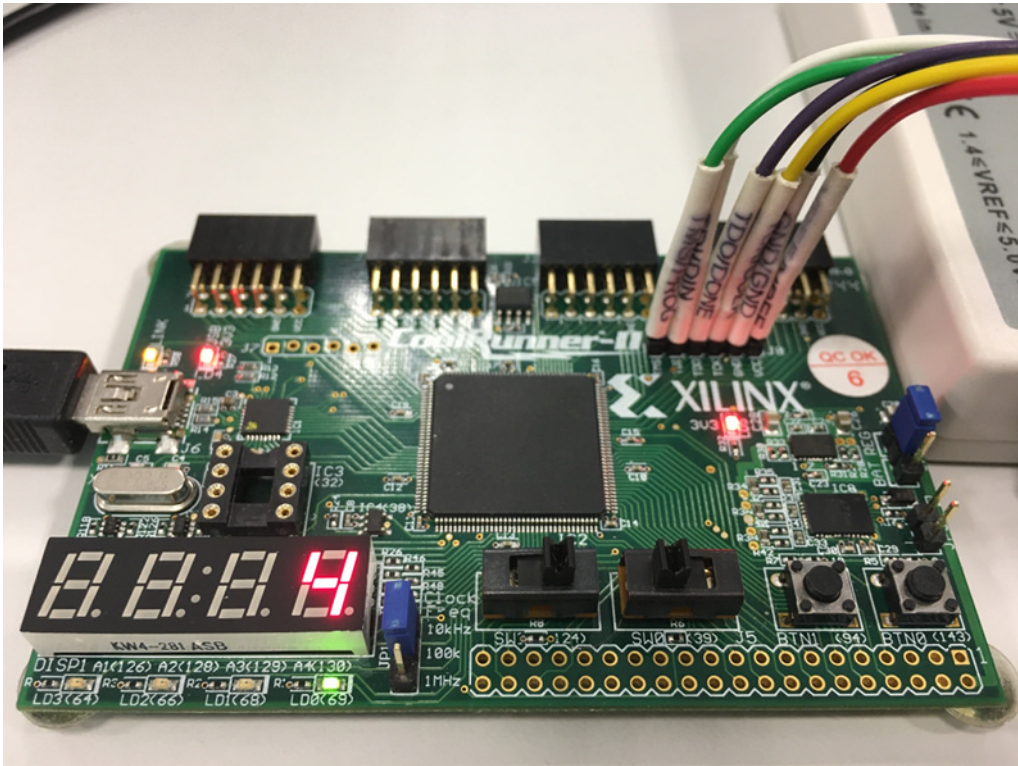


Figure 1: CoolRunner-II CPLD starter board

1 Preparation tasks (done before the lab at home)

1. See schematic or reference manual of the board and find out the connection of 7-segment display. How can you change the position of the character on the display?
2. Draw the patterns for hexadecimal digits and complete the decoder conversion table for **common anode** display.

Hex	Input	a	b	c	d	e	f	g
0	0000	0	0	0	0	0	0	1
1	0001	1	0	0	1	1	1	1
2								
3								
4								
5								
6								
7								
8								

Hex	Input	a	b	c	d	e	f	g
9								
A								
b								
C								
d								
E	1110	0	1	1	0	0	0	0
F	1111	0	1	1	1	0	0	0

2 Synchronize Git and create a new folder

1. Open a Linux terminal, use `cd` commands to change path to your Digital-electronics-1 working directory, and synchronize the contents with GitHub.

```
$ pwd
/home/lab661
$ cd Documents/your-name/Digital-electronics-1/
$ pwd
/home/lab661/Documents/your-name/Digital-electronics-1
$ git pull
```

2. Create a new folder `Labs/03-segment`

```
$ cd Labs/
$ mkdir 03-segment
$ cd 03-segment/
$ touch README.md
$ ls
README.md
```

3 VHDL code for hexadecimal digit to seven-segment decoder

1. Follow instructions from wiki, create a new project in ISE titled `hex_to_segment` for XC2C256-TQ144 CPLD device. Make sure the project location is `/home/lab661/Documents/your-name/Digital-electronics-1/La`, ie in **your** local folder.
2. Create a new source file **Project** > **New Source...** > **VHDL Module**, name it `hex_to_7seg` and copy/paste the following code template.

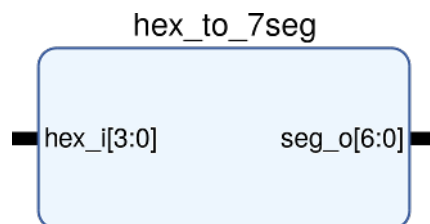


Figure 2: Ports of hexadecimal digit to seven-segment decoder

```
-----
--
-- Hexadecimal digit to seven-segment decoder.
-- Xilinx XC2C256-TQ144 CPLD, ISE Design Suite 14.7
--
-- Copyright (c) 2019-2020 Tomas Fryza
-- Dept. of Radio Electronics, Brno University of Technology, Czechia
-- This work is licensed under the terms of the MIT license.
```

```

--
-----

library ieee;
use ieee.std_logic_1164.all;

-----

-- Entity declaration for hexadecimal digit to seven-segment decoder
-----

entity hex_to_7seg is
port (
    hex_i : in  std_logic_vector(4-1 downto 0);
    seg_o : out std_logic_vector(7-1 downto 0)
);
end entity hex_to_7seg;

-----

-- Architecture declaration for hex to seven-segment decoder
-----

architecture Behavioral of hex_to_7seg is
begin

    -----

    --
    --      a
    --      -----
    --      f |      | b      a: seg_o(6)
    --      |  g  |          b: seg_o(5)
    --      |-----|          c: seg_o(4)
    --      e |      | c      d: seg_o(3)
    --      |      |          e: seg_o(2)
    --      |      |          f: seg_o(1)
    --      |      |          g: seg_o(0)
    --      -----
    --      d
    --
    -----

    seg_o <= "0000001" when (hex_i = "0000") else -- 0
            "1001111" when (hex_i = "0001") else -- 1

            -- WRITE YOUR CODE HERE

            "0110000" when (hex_i = "1110") else -- E
            "0111000";                          -- F

end architecture Behavioral;

```

3. See how signals can be assigned outside the process, complete the decoding table for all input combinations, and define the output signals to display hexadecimal symbols (0, 1, ..., 9, A, b, C, d, E, F). Save all files in menu **File > Save All**.
4. In menu **Tools > Schematic Viewer > RTL...** select **Start with a schematic of top-level block** and check the hierarchical structure of the module.

4 VHDL code for top level

1. Create a new source file **Project > New Source...** > **VHDL Module**, name it `top` and copy/paste the following code template.

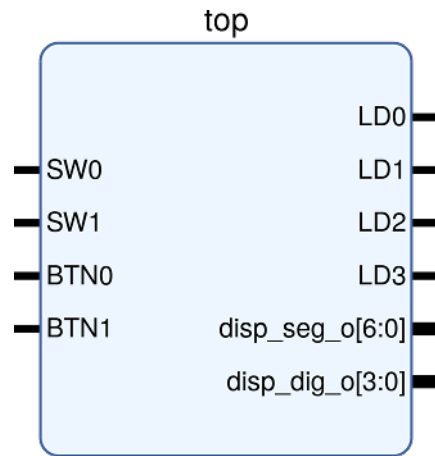


Figure 3: Ports of top module

```

-----
--
-- Implementation of hexadecimal digit to seven-segment decoder.
-- Xilinx XC2C256-TQ144 CPLD, ISE Design Suite 14.7
--
-- Copyright (c) 2018-2020 Tomas Fryza
-- Dept. of Radio Electronics, Brno University of Technology, Czechia
-- This work is licensed under the terms of the MIT license.
--
-----

library ieee;
use ieee.std_logic_1164.all;

-----
-- Entity declaration for top level
-----

entity top is
port (
    SW0, SW1 :      in  std_logic;
    BTN0, BTN1 :    in  std_logic;
    LD0, LD1, LD2, LD3 : out std_logic;
    disp_seg_o :    out std_logic_vector(7-1 downto 0);
    disp_dig_o :    out std_logic_vector(4-1 downto 0)
);
end entity top;

-----
-- Architecture declaration for top level
-----

architecture Behavioral of top is
    signal s_hex : std_logic_vector(4-1 downto 0); -- Internal signals
begin

    -- Combine inputs [SW1, SW0, BTN1, BTN0] into internal vector
    s_hex(3) <= SW1;
    s_hex(2) <= SW0;
    s_hex(1) <= not BTN1;
    s_hex(0) <= not BTN0;

    -----

```

```

-- Sub-block of hex_to_7seg entity
HEX2SSEG : entity work.hex_to_7seg
port map (
    -- <component_signal> => <actual_signal>,
    -- <component_signal> => <actual_signal>,
    -- <other signals>...
    -- WRITE YOUR CODE HERE
);

-- Select display position
disp_dig_o <= "1110";

-- Turn on LD3 if the input value is equal to "0000"
-- WRITE YOUR CODE HERE

-- Turn on LD2 if the input value is A, B, C, D, E, or F
-- WRITE YOUR CODE HERE

-- Turn on LD1 if the input value is odd, ie 1, 3, ..., F
-- WRITE YOUR CODE HERE

-- Turn on LD0 if the input value is a power of two, ie 1, 2, 4, or 8
-- WRITE YOUR CODE HERE

end architecture Behavioral;

```

2. Use onboard push buttons and slide switches as 4-bit input. How is the sub-block of hex to 7-segment decoder connected to the top module?
3. What coding style is used to name the input, output, and internal signals in VHDL?
4. Follow instructions from wiki, create a constraints file, and implement your design to CoolRunner-II CPLD starter board.
5. Write logic functions for LEDs. Let two functions are defined using VHDL construction `when - else` and two functions using low-level gates `and` , `or` , `not` , etc.
6. In menu **Tools > Schematic Viewer > RTL...** select **Start with a schematic of top-level block** and check the hierarchical structure of the module.
7. In menu **Project > Design Summary/Reports** check **CPLD Fitter Report (Text)** for implemented functions in section `***** Mapped Logic *****` .

5 Clean project and synchronize git

1. In Xilinx ISE, clean up all generated files in menu **Project > Cleanup Project Files...** and close the project using **File > Close Project**.

Warning: In any file manager, make sure the project folder does not contain any **large** (gigabyte) files. These can be caused by incorrect simulation in ISim. Delete such files.

2. Use `cd ..` command in Linux terminal and change working directory to `Digital-electronics-1` . Then use git commands to add, commit, and push all local changes to your remote repository. Check the repository at GitHub web page for changes.

```

$ pwd
/home/lab661/Documents/your-name/Digital-electronics-1/Labs/03-segment

$ cd ..
$ cd ..
$ pwd

```

```
/home/lab661/Documents/your-name/Digital-electronics-1  
  
$ git status  
$ git add <your-modified-files>  
$ git status  
$ git commit -m "[LAB] Adding 03-segment lab"  
$ git status  
$ git push  
$ git status
```

Experiments on your own

1. Program and simulate a 4-to-1 multiplexer consists of four data input lines `data_i` two select lines `sel_i` and a single output line `y_o` .
2. Complete your `README.md` file with notes and screenshots from the implementation.