Lab 6: Driver for seven-segment display

Objectives

In this laboratory exercise, you will study the creation of a sequential circuit for multiplexing a 7-segment display. This allows you to display 4-digit values including the decimal point on the display.

Materials

You will use a push button on the CoolRunner-II CPLD starter board (XC2C256-TQ144, manual, schematic) as reset device, onboard clock signal with frequency of 10 kHz for synchronization, and 7-segment display as output device. You will also use slide switches on the CPLD expansion board (schematic) as inputs.

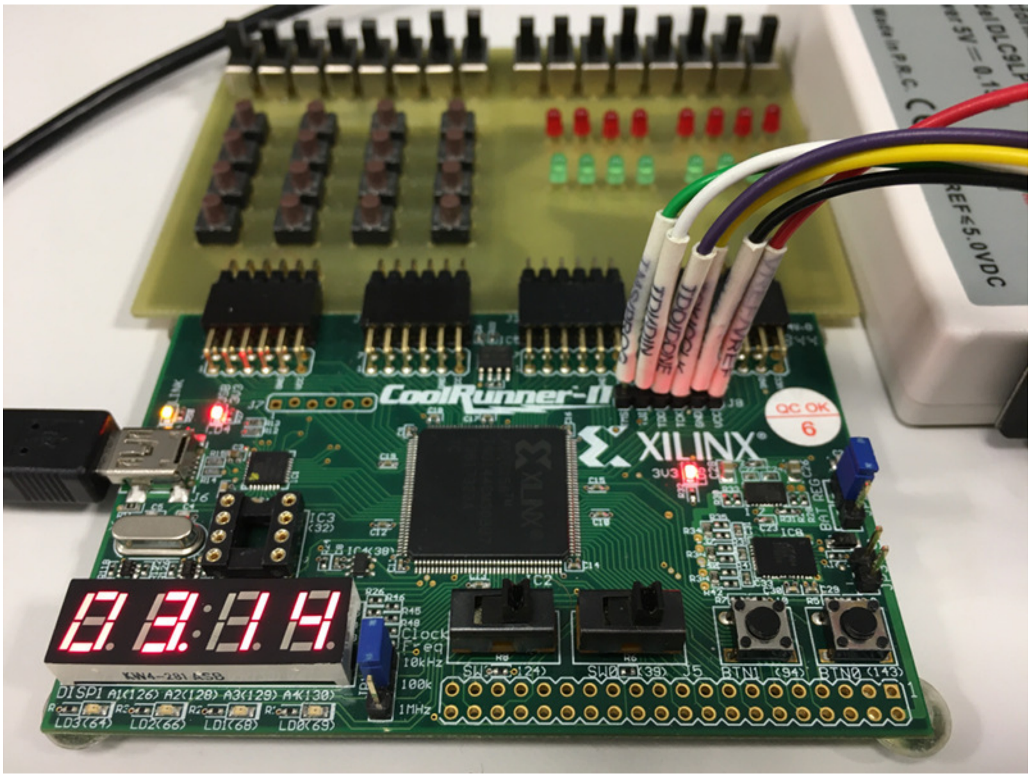

Figure 1: CoolRunner-II CPLD starter board

1 Preparation tasks (done before the lab at home)

1. See reference manual of the Coolrunner board, find out the connection of 7-segment display, and complete the signal timing to display `03.14` value. Note that the duration of one symbol is 4 ms.

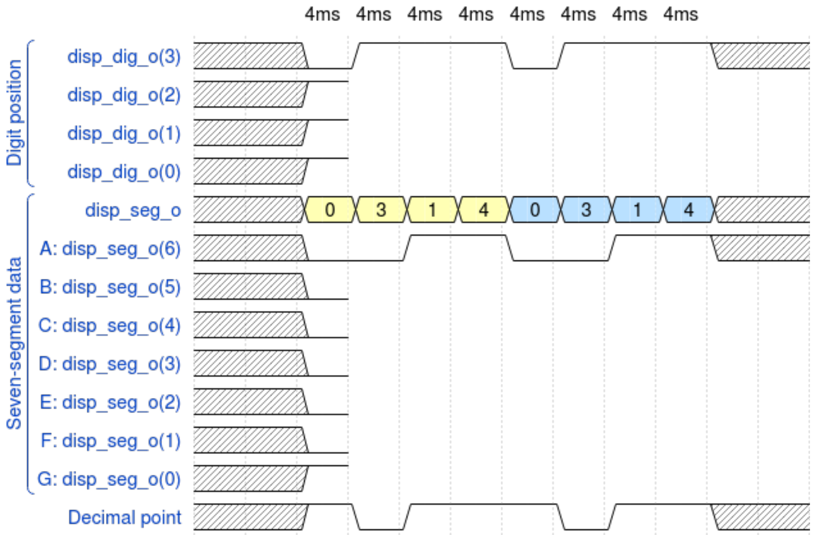

Figure 2: Timing of seven-segment display

The figure above was created in WaveDrom digital timing diagram online tool. The figure source code is as follows:

```
{signal: [
  ['Digit position',
    {name: 'disp_dig_o(3)', wave: 'xx01..01..xx', },
    {name: 'disp_dig_o(2)', wave: 'xx1', },
    {name: 'disp_dig_o(1)', wave: 'xx1', },
    {name: 'disp_dig_o(0)', wave: 'xx1', },
  ],
  ['Seven-segment data',
    {name: 'disp_seg_o',       wave: 'xx33335555xx', data: ['0','3','1','4','0','3','1','4'], },
    {name: 'A: disp_seg_o(6)', wave: 'xx0.1.0.1.xx', },
    {name: 'B: disp_seg_o(5)', wave: 'xx0',          },
    {name: 'C: disp_seg_o(4)', wave: 'xx0',          },
    {name: 'D: disp_seg_o(3)', wave: 'xx0',          },
    {name: 'E: disp_seg_o(2)', wave: 'xx0',          },
    {name: 'F: disp_seg_o(1)', wave: 'xx0',          },
    {name: 'G: disp_seg_o(0)', wave: 'xx1',          },
  ],
  {name: 'Decimal point', wave: 'xx101..01.xx', },
],
  head: {
    text: '4ms   4ms   4ms   4ms   4ms   4ms   4ms   4ms',
  },
}
```

2. See how to make signal assignments outside and inside a process. What is the difference between combinational and sequential processes?

2 Synchronize Git and create a new folder

1. Open a Linux terminal, change path to your Digital-electronics-1 working directory, and synchronize the contents with GitHub.
2. Create a new folder `Labs/06-display_driver`

3 Display driver VHDL code

Multiplexer or MUX is a digital switch. It allows to route binary information from several input lines or sources to one output line or channel.

1. Create a new project in ISE titled `display_driver` for XC2C256-TQ144 CPLD device in location `/home/lab661/Documents/your-name/Digital-electronics-1/Labs/06-display_driver`
2. Create a new VHDL module `driver_7seg` and copy/paste the following code template.

```vhdl
-------------------------------------------------------------------------
--
-- Driver for seven-segment displays.
-- 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;
use ieee.std_logic_unsigned.all;    -- Provides unsigned numerical computation

-------------------------------------------------------------------------
-- Entity declaration for display driver
-------------------------------------------------------------------------
entity driver_7seg is
port (
    clk_i    : in  std_logic;
    srst_n_i : in  std_logic;   -- Synchronous reset (active low)
    data0_i  : in  std_logic_vector(4-1 downto 0);  -- Input values
    data1_i  : in  std_logic_vector(4-1 downto 0);
    data2_i  : in  std_logic_vector(4-1 downto 0);
    data3_i  : in  std_logic_vector(4-1 downto 0);
    dp_i     : in  std_logic_vector(4-1 downto 0);  -- Decimal points

    dp_o     : out std_logic;                       -- Decimal point
    seg_o    : out std_logic_vector(7-1 downto 0);
    dig_o    : out std_logic_vector(4-1 downto 0)
);
end entity driver_7seg;

-------------------------------------------------------------------------
-- Architecture declaration for display driver
-------------------------------------------------------------------------
architecture Behavioral of driver_7seg is
    signal s_en  : std_logic;
    signal s_hex : std_logic_vector(4-1 downto 0);
    signal s_cnt : std_logic_vector(2-1 downto 0) := "00";
begin

    ---------------------------------------------------------------------
    -- Sub-block of clock_enable entity. Create s_en signal.
    --- WRITE YOUR CODE HERE


    ---------------------------------------------------------------------
    -- Sub-block of hex_to_7seg entity
    --- WRITE YOUR CODE HERE


    --------------------------------------------------------------------
    -- p_select_cnt:
    -- Sequential process with synchronous reset and clock enable,
    -- which implements an internal 2-bit counter s_cnt for multiplexer
    -- selection bits.
    --------------------------------------------------------------------
    p_select_cnt : process (clk_i)
    begin
        if rising_edge(clk_i) then  -- Rising clock edge
            if srst_n_i = '0' then  -- Synchronous reset (active low)
                -- WRITE YOUR CODE HERE
            elsif s_en = '1' then
                -- WRITE YOUR CODE HERE
            end if;
        end if;
    end process p_select_cnt;

    --------------------------------------------------------------------
    -- p_mux:
    -- Combinational process which implements a 4-to-1 mux.
    --------------------------------------------------------------------
    p_mux : process (s_cnt, data0_i, data1_i, data2_i, data3_i, dp_i)
    begin
        case s_cnt is
        when "00" =>
            -- WRITE YOUR CODE HERE
        when "01" =>
            -- WRITE YOUR CODE HERE
        when "10" =>
            -- WRITE YOUR CODE HERE
        when others =>
            -- WRITE YOUR CODE HERE
        end case;
    end process p_mux;

end architecture Behavioral;
```

2. Connect clock enable (period of 4 ms) and hex to seven-segment decoder sub-blocks. Define internal signals if needed. Copy VHDL file of clock enable and seven-segment decoder to the working folder. Add these files to the project.

Implement an internal 2-bit binary counter and use the value in a combinational process to perform 4-to-1 mux. For each selection, define output value, decimal point, and digit position.

4 Top level implementation of display driver

1. Create a new VHDL module `top` and copy/paste the following code template.

```
------------------------------------------------------------------------
--
-- Implementation of seven-segment display driver.
-- 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 top level
-------------------------------------------------------------------------
entity top is
port (
    clk_i : in std_logic;       -- 10 kHz clock signal
    BTN0  : in std_logic;       -- Synchronous reset
    SW0_CPLD,  SW1_CPLD,  SW2_CPLD,  SW3_CPLD  : in std_logic; -- Input 0
    SW4_CPLD,  SW5_CPLD,  SW6_CPLD,  SW7_CPLD  : in std_logic; -- Input 1
    SW8_CPLD,  SW9_CPLD,  SW10_CPLD, SW11_CPLD : in std_logic; -- Input 2
    SW12_CPLD, SW13_CPLD, SW14_CPLD, SW15_CPLD : in std_logic; -- Input 3

    disp_dp    : out std_logic; -- Decimal point
    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_data0, s_data1 : std_logic_vector(4-1 downto 0);
    signal s_data2, s_data3 : std_logic_vector(4-1 downto 0);
begin

    -- Combine 4-bit inputs to internal signals
    -- WRITE YOUR CODE HERE



    ---------------------------------------------------------------------
    -- Sub-block of driver_7seg entity
    --- WRITE YOUR CODE HERE

end architecture Behavioral;
```

2. Connect seven-segment driver and implement it on the Coolrunner-II board. Use slide switches on the CPLD expansion board as data inputs and display the values on the 7-segment display. Connect the reset to BTN0 push button and make sure the 10kHz clock frequency is selected by JP1 jumper. Copy Coolrunner and Expansion UCF files to the working folder. Add these files to the project.

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 git commands to add, commit, and push all local changes to your remote repository. Check the repository at GitHub web page for changes.

Experiments on your own

1. On your smartphone, set slow motion video recording and observe the seven-segment display behavior:)
2. Display 4-bit input values with green and red LEDs on the CPLD expansion board.
3. Display digit position value with LEDs on the Coolrunner board.
4. Extend the duration of one symbol on the 7-segment display (ie. generic `g_NPERIOD` in `driver_7seg.vhd` file) and experimentally determine the maximum value at which switching by the human eye is not yet observable.
5. Complete your `README.md` file with notes and screenshots from simulation and implementation.