Labs - 06 - Display_driver

Experiments on your own

1. On your smartphone, set slow motion video recording and observe the seven-segment display behavior:)
2. Extend the duration of one symbol on the 7-segment display ie. generic `g_MAX` in `driver_7seg_4digit.vhd` file and experimentally determine the maximum value at which switching by the human eye is not yet observable.
3. Design the structure of `driver_7seg_8digits` module, which controls all eight 7-segment displays.

Lab assignment

1. Preparation tasks (done before the lab at home). Submit:
    - Timing diagram figure for displaying value `3.142`.
2. Display driver. Submit:
    - Listing of VHDL code of the process `p_mux` with syntax highlighting.
    - Listing of VHDL testbench file `tb_driver_7seg_4digits` with syntax highlighting and asserts,
    - Screenshot with simulated time waveforms; always display all inputs and outputs,

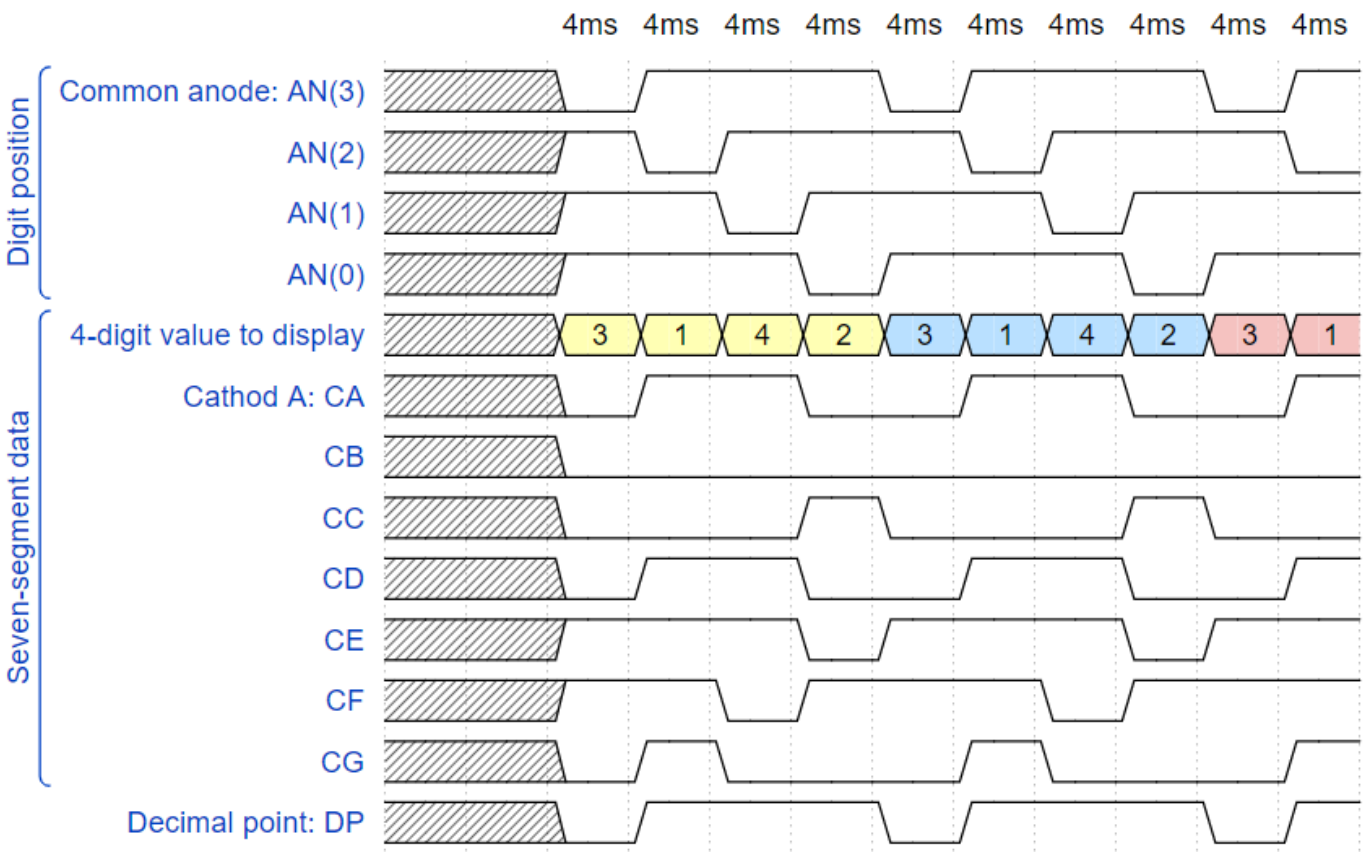
- Listing of VHDL architecture of the top layer.

3. Eight-digit driver. Submit:

   - Image of the driver schematic. The image can be drawn on a computer or by hand.

1. Preparation tasks

1.1. Timing diagram figure for displaying value `3.142`




```
{
  signal:
  [
    ['Digit position',
      {name: 'Common anode: AN(3)', wave: 'xx01..01..01'},
      {name: 'AN(2)', wave: 'xx1'},
      {name: 'AN(1)', wave: 'xx1'},
      {name: 'AN(0)', wave: 'xx1'},
    ],
    ['Seven-segment data',
      {name: '4-digit value to display', wave: 'xx3333555599', data: ['3','1','4','2','3','
      {name: 'Cathod A: CA', wave: 'xx01.0.1.0.1'},
      {name: 'CB', wave: 'xx0'},
      {name: 'CC', wave: 'xx0'},
      {name: 'CD', wave: 'xx0'},
      {name: 'CE', wave: 'xx1'},
      {name: 'CF', wave: 'xx1'},
      {name: 'CG', wave: 'xx0'},
    ],
    {name: 'Decimal point: DP', wave: 'xx01..01..01'},
```

```
  ],
  head:
  {
    text: '                    4ms    4ms    4ms    4ms    4ms    4ms    4ms    4ms    4ms    4ms',
  },
}
```

2. Display driver

2.1. Listing of VHDL code of the process `p_mux` with syntax highlighting

```
    p_mux : process(s_cnt, data0_i, data1_i, data2_i, data3_i, dp_i)
    begin
        case s_cnt is
            when "11" =>
                s_hex <= data3_i;
                dp_o  <= dp_i(3);
                dig_o <= "0111";

            when "10" =>
                s_hex <= data2_i;
                dp_o  <= dp_i(2);
                dig_o <= "1011";

            when "01" =>
                s_hex <= data1_i;
                dp_o  <= dp_i(1);
                dig_o <= "1101";

            when others =>
                s_hex <= data0_i;
                dp_o  <= dp_i(0);
                dig_o <= "1110";
        end case;
    end process p_mux;
```

2.2. Listing of VHDL testbench file `tb_driver_7seg_4digits` with syntax highlighting and asserts

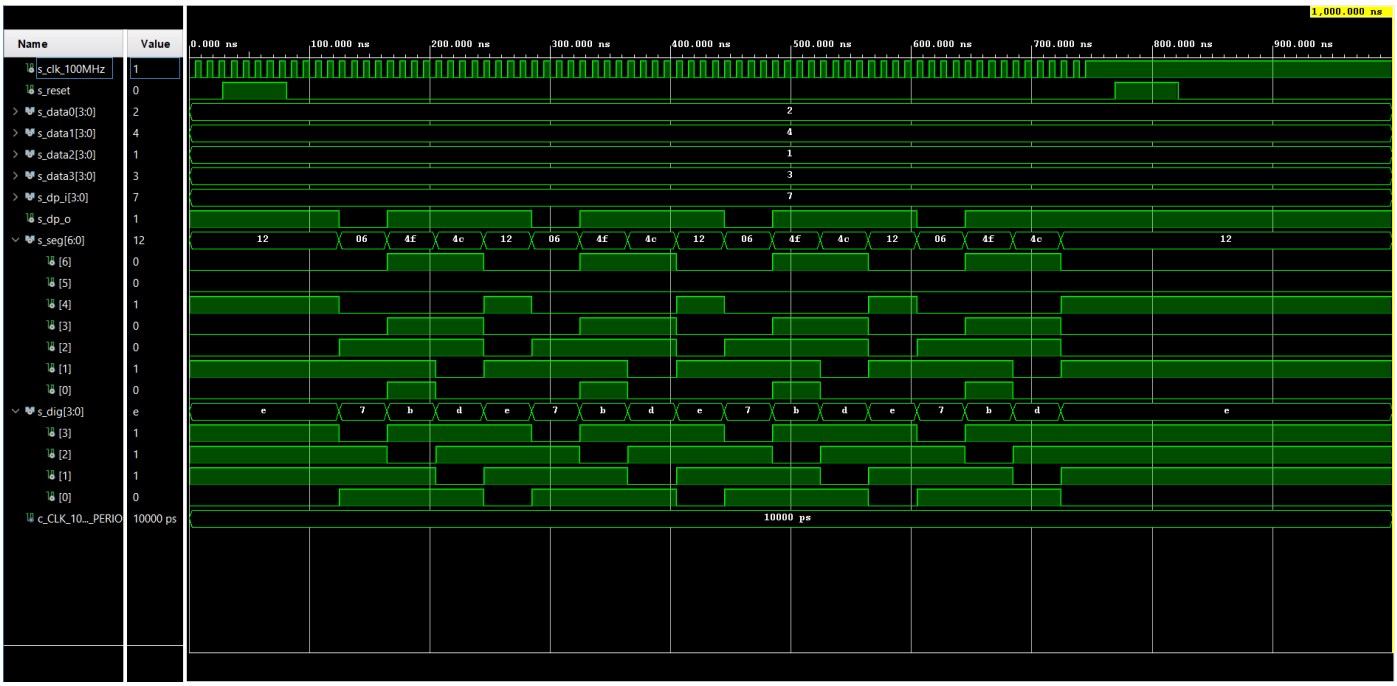
```
----------------------------------------------------------------------
--
-- Template for 4-digit 7-segment display driver testbench.
-- Nexys A7-50T, Vivado v2020.1.1, EDA Playground
--
-- Copyright (c) 2020 Tomas Fryza
-- Dept. of Radio Electronics, Brno University of Technology, Czechia
-- This work is licensed under the terms of the MIT license.
--
----------------------------------------------------------------------
```

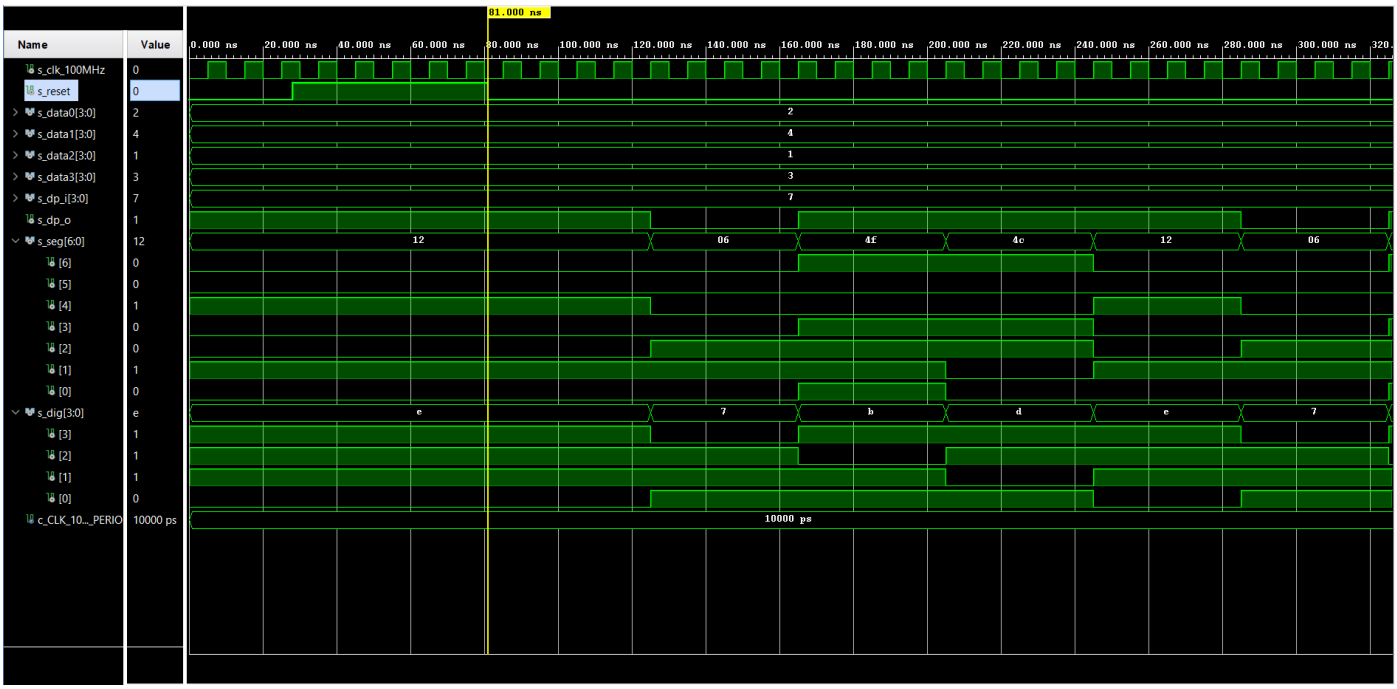
```
library ieee;
use ieee.std_logic_1164.all;

----------------------------------------------------------------------
-- Entity declaration for testbench
----------------------------------------------------------------------
entity tb_driver_7seg_4digits is
    -- Entity of testbench is always empty
end entity tb_driver_7seg_4digits;

----------------------------------------------------------------------
-- Architecture body for testbench
----------------------------------------------------------------------
architecture testbench of tb_driver_7seg_4digits is

    -- Local constants
    constant c_CLK_100MHZ_PERIOD : time    := 10 ns;

    --Local signals
    signal s_clk_100MHz : std_logic;
    --- WRITE YOUR CODE HERE
    signal s_reset : std_logic;

    signal s_data0 : std_logic_vector(4 - 1 downto 0);
    signal s_data1 : std_logic_vector(4 - 1 downto 0);
    signal s_data2 : std_logic_vector(4 - 1 downto 0);
    signal s_data3 : std_logic_vector(4 - 1 downto 0);

    signal s_dp_i  : std_logic_vector(4 - 1 downto 0);
    signal s_dp_o  : std_logic;
    signal s_seg   : std_logic_vector(7 - 1 downto 0);

    signal s_dig   : std_logic_vector(4 - 1 downto 0);


begin
    -- Connecting testbench signals with driver_7seg_4digits entity
    -- (Unit Under Test)
    uut_driver_7seg_4digits : entity work.driver_7seg_4digits
    port map (
        clk => s_clk_100Mhz,
        reset => s_reset,

        data0_i => s_data0,
        data1_i => s_data1,
        data2_i => s_data2,
        data3_i => s_data3,

        dp_i => s_dp_i,

        dp_o => s_dp_o,
        seg_o => s_seg,
        dig_o => s_dig
    );
    ----------------------------------------------------------------------
    -- Clock generation process
```

```
    ----------------------------------------------------------------------
    p_clk_gen : process
    begin
        while now < 750 ns loop         -- 75 periods of 100MHz clock
            s_clk_100MHz <= '0';
            wait for c_CLK_100MHZ_PERIOD / 2;
            s_clk_100MHz <= '1';
            wait for c_CLK_100MHZ_PERIOD / 2;
        end loop;
        wait;
    end process p_clk_gen;

    ----------------------------------------------------------------------
    -- Reset generation process
    ----------------------------------------------------------------------
    p_reset_gen : process
    begin
        s_reset <= '0';
        wait for 28 ns;

        -- Reset activated
        s_reset <= '1';
        wait for 53 ns;

        -- Reset deactivated
        s_reset <= '0';
        wait for 660 ns;


    end process p_reset_gen;

    ----------------------------------------------------------------------
    -- Data generation process
    ----------------------------------------------------------------------
p_stimulus : process
    begin
        report "Stimulus process started" severity note;
        -- 3.142
        s_data3 <= "0011";
        s_data2 <= "0001";
        s_data1 <= "0100";
        s_data0 <= "0010";

        s_dp_i  <= "0111";

        --wait for 350 ns;

        --s_data3 <= "0001";
        --s_data2 <= "0000";
        --s_data1 <= "0001";
        --s_data0 <= "0000";


        report "Stimulus process finished" severity note;
        wait;
    end process p_stimulus;
```

```
end architecture testbench;
```

2.3. Screenshot with simulated time waveforms; always display all inputs and outputs





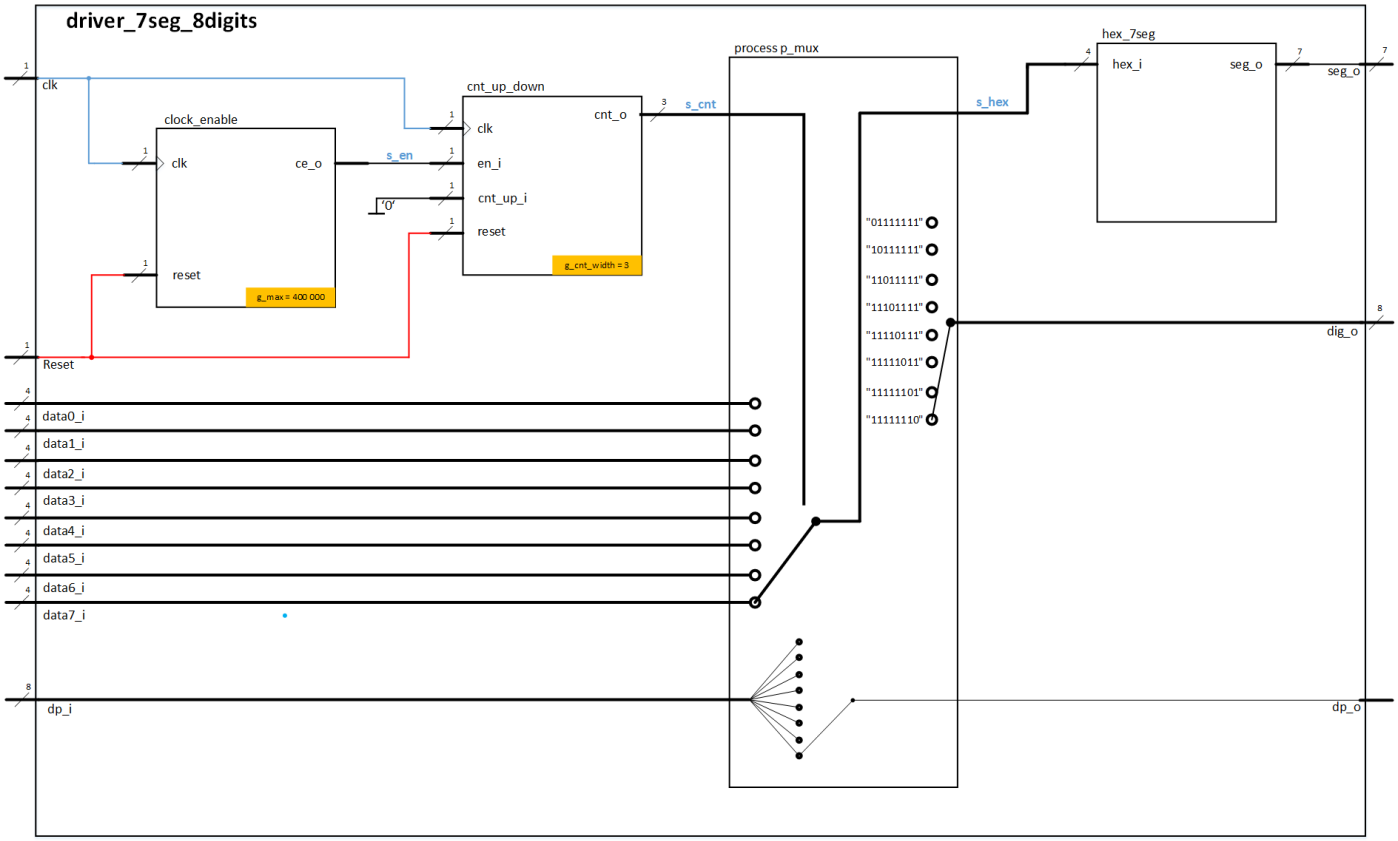
2.4. Listing of VHDL architecture of the top layer

```
architecture Behavioral of top is
begin
    ------------------------------------------------------------------
    -- Instance (copy) of driver_7seg_4digits entity
    driver_seg_4 : entity work.driver_7seg_4digits
        port map
        (
            clk         => CLK100MHZ,
```

```
            reset       => BTNC,

            data0_i(3)  => SW(3),
            data0_i(2)  => SW(2),
            data0_i(1)  => SW(1),
            data0_i(0)  => SW(0),

            data1_i(3)  => SW(7),
            data1_i(2)  => SW(6),
            data1_i(1)  => SW(5),
            data1_i(0)  => SW(4),

            data2_i(3)  => SW(11),
            data2_i(2)  => SW(10),
            data2_i(1)  => SW(9),
            data2_i(0)  => SW(8),

            data3_i(3)  => SW(15),
            data3_i(2)  => SW(14),
            data3_i(1)  => SW(13),
            data3_i(0)  => SW(12),

            seg_o(6)    => CA,
            seg_o(5)    => CB,
            seg_o(4)    => CC,
            seg_o(3)    => CD,
            seg_o(2)    => CE,
            seg_o(1)    => CF,
            seg_o(0)    => CG,

            dig_o    => AN(4 - 1 downto 0),

            dp_i        => "0111",
            dp_o        => DP
        );
end architecture Behavioral;
```

3. Eight-digit driver

3.1. Image of the driver schematic


driver_7seg_8digits
clk
Reset
clock_enable
clk
reset
ce_o
g_max = 400 000
s_en
'0'
cnt_up_down
clk
en_i
cnt_up_i
reset
cnt_o
g_cnt_width = 3
s_cnt
process p_mux
"01111111"
"10111111"
"11011111"
"11101111"
"11110111"
"11111011"
"11111101"
"11111110"
s_hex
hex_7seg
hex_i
seg_o
seg_o
dig_o
data0_i
data1_i
data2_i
data3_i
data4_i
data5_i
data6_i
data7_i
dp_i
dp_o