FranciscaCampos / **Digital-electronics-1**

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main

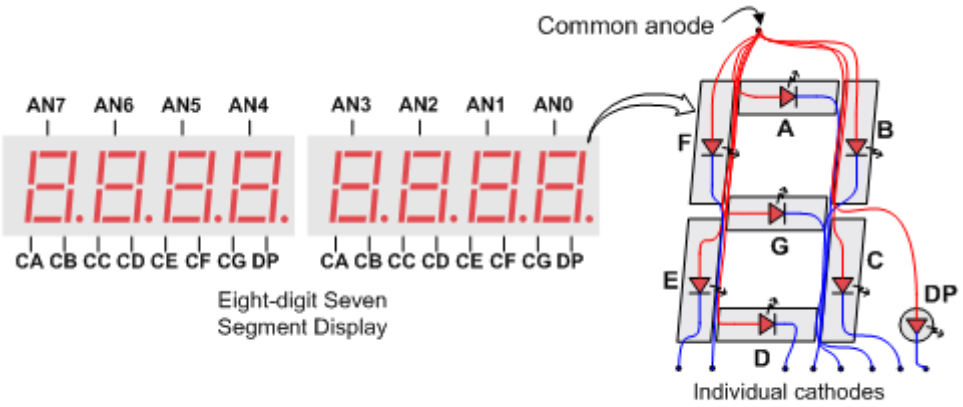
Digital-electronics-1 / Labs / **04-segment** /

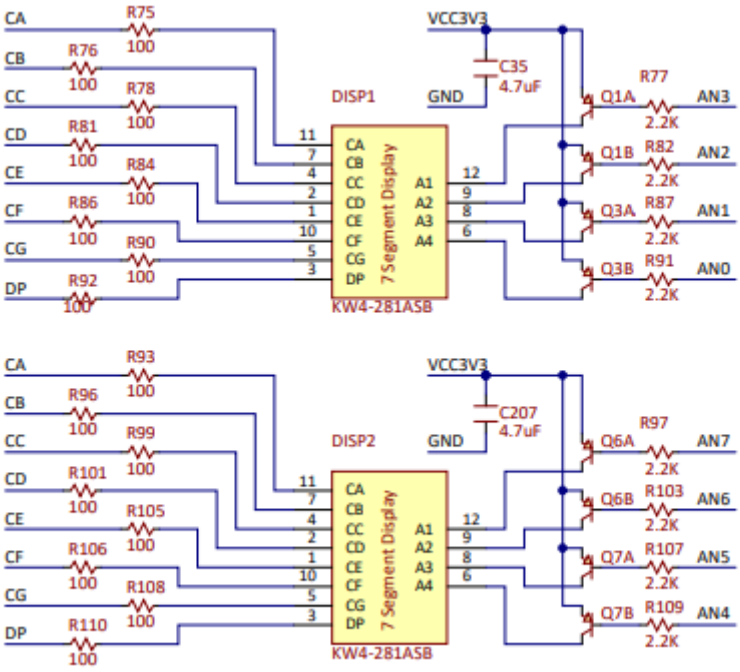
FranciscaCampos ...		20 seconds ago
..		
display		25 minutes ago
7segCon.png		25 minutes ago
Lab4.md		25 minutes ago
README.md		22 minutes ago
Schematic7Seg.PNG		25 minutes ago
Sim1.PNG		20 seconds ago
Sim2.PNG		20 seconds ago
SimuFinal.PNG		25 minutes ago

README.md

Lab 4: Seven-segment display decoder

1. Preparation tasks





Hex	Inputs	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	0010	0	0	1	0	0	1	0
3	0011	0	0	0	0	1	1	0
4	0100	1	0	0	1	1	0	0
5	0101	0	1	0	0	1	0	0

Hex	Inputs	A	B	C	D	E	F	G
6	0110	0	1	0	0	0	0	0
7	0111	0	0	0	1	1	1	1
8	1000	0	0	0	0	0	0	0
9	1001	0	0	0	0	1	0	0
A	1010	0	0	0	1	0	0	0
b	1011	1	1	0	0	0	0	0
C	1100	0	1	1	0	0	0	1
d	1101	1	0	0	0	0	1	0
E	1110	0	1	1	0	0	0	0
F	1111	0	1	1	1	0	0	0

2. Seven-segment display decoder

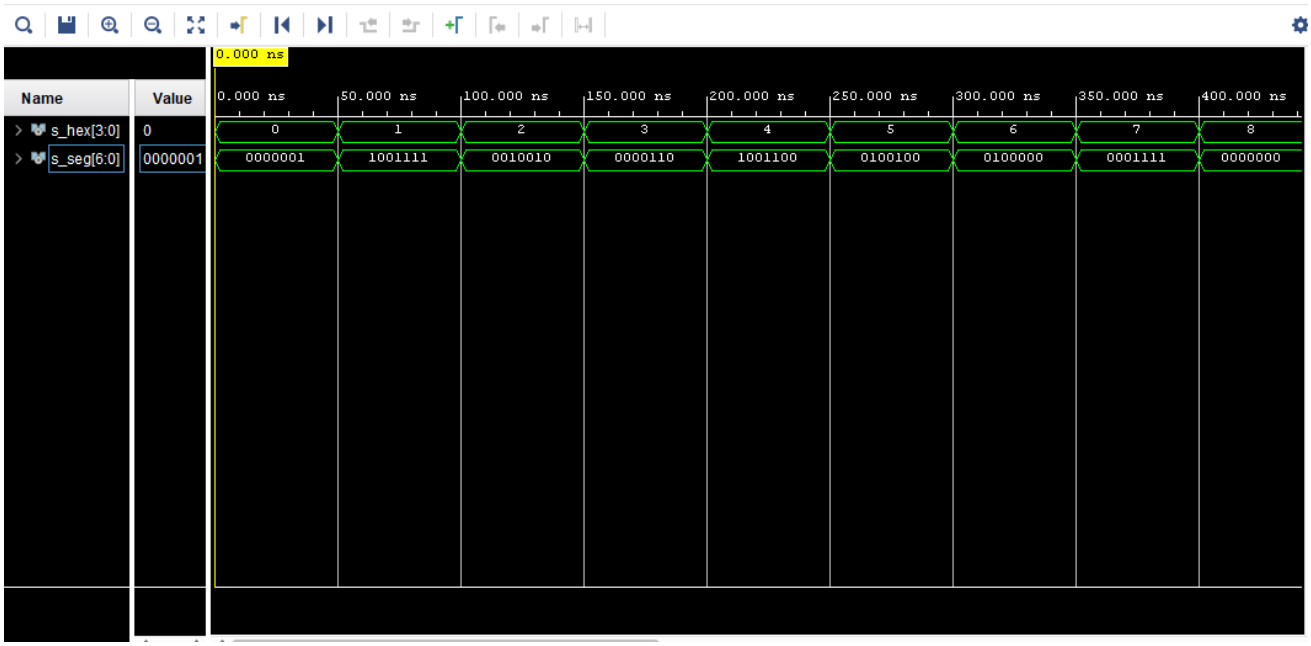
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity hex_7seg is
   Port ( hex_i : in STD_LOGIC_VECTOR (3 downto 0);
          seg_o : out STD_LOGIC_VECTOR (6 downto 0));
end hex_7seg;


------------------------------------------------------------------------
-- Architecture body for seven-segment display decoder
------------------------------------------------------------------------
architecture Behavioral of hex_7seg is
begin

   ------------------------------------------------------------------
   -- p_7seg_decoder:
   -- A combinational process for 7-segment display (Common Anode)
   -- decoder. Any time "hex_i" is changed, the process is "executed".
```

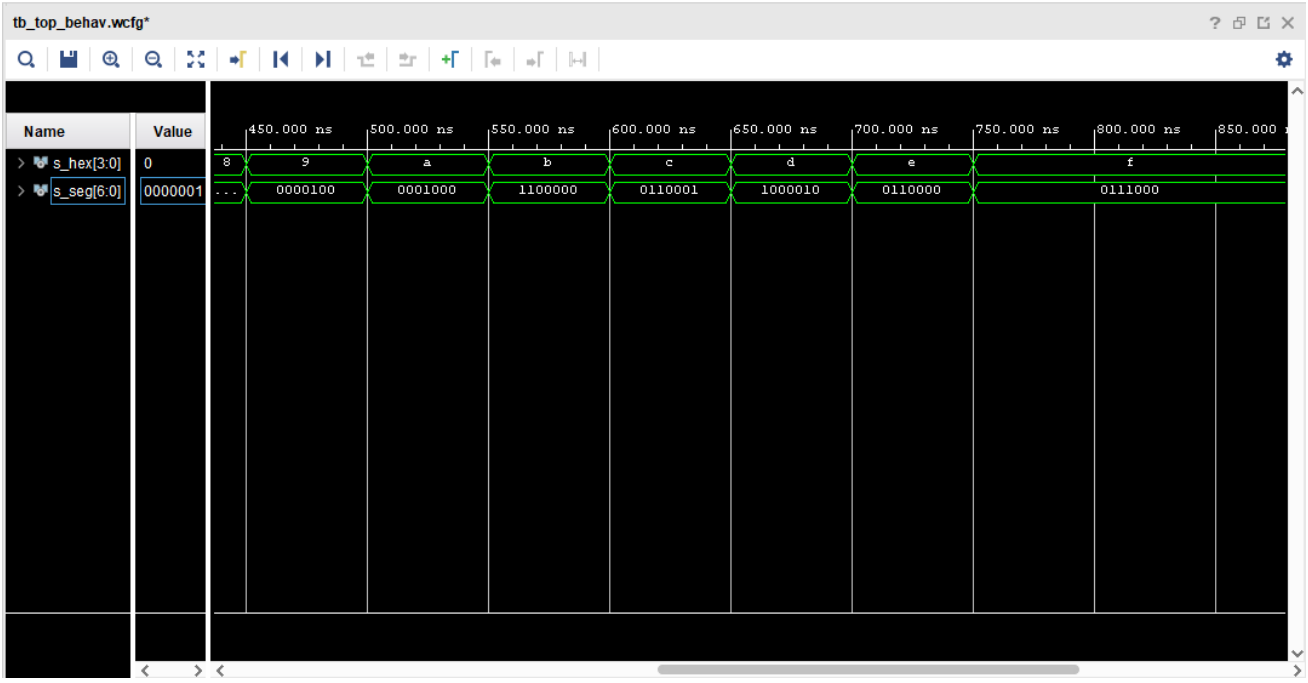
```
    -- Output pin seg_o(6) controls segment A, seg_o(5) segment B, etc.
    --       segment A
    --        | segment B
    --        |  | segment C
    --        |  |  |   ...   segment G
    --        +-+|+-+          |
    --          |||            |
    -- seg_o = "0000001"-------+
    ------------------------------------------------------------------------
    p_7seg_decoder : process(hex_i)
    begin
        case hex_i is
            when "0000" =>
                seg_o <= "0000001";     -- 0
            when "0001" =>
                seg_o <= "1001111";     -- 1
            when "0010" =>
                seg_o <= "0010010";     -- 2
            when "0011" =>
                seg_o <= "0000110";     -- 3
            when "0100" =>
                seg_o <= "1001100";     -- 4
            when "0101" =>
                seg_o <= "0100100";     -- 5
            when "0110" =>
                seg_o <= "0100000";     -- 6
            when "0111" =>
                seg_o <= "0001111";     -- 7
            when "1000" =>
                seg_o <= "0000000";     -- 8
            when "1001" =>
                seg_o <= "0000100";     -- 9
            when "1010" =>
                seg_o <= "0001000";     -- A
            when "1011" =>
                seg_o <= "1100000";     -- b
            when "1100" =>
                seg_o <= "0110001";     -- C
             when "1101" =>
                seg_o <= "1000010";     -- d
            when "1110" =>
                seg_o <= "0110000";     -- E
            when others =>
                seg_o <= "0111000";     -- F
        end case;
    end process p_7seg_decoder;

end architecture Behavioral;



------------------------------------------------------------------------
--
-- Testbench for 7-segment display decoder.
-- Nexys A7-50T, Vivado v2020.1, EDA Playground
```

```
--
-- Copyright (c) 2020-Present 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_hex_7seg is
    -- Entity of testbench is always empty
end entity tb_hex_7seg;

------------------------------------------------------------------------
-- Architecture body for testbench
------------------------------------------------------------------------
architecture testbench of tb_hex_7seg is

    -- Local signals
    signal s_hex  : std_logic_vector(4 - 1 downto 0);
    signal s_seg  : std_logic_vector(7 - 1 downto 0);

begin
    -- Connecting testbench signals with decoder entity (Unit Under Test)
    uut_hex_7seg : entity work.hex_7seg
        port map(
            hex_i => s_hex,
            seg_o => s_seg
        );

    ------------------------------------------------------------------------
    -- Data generation process
    ------------------------------------------------------------------------
    p_stimulus : process
    begin
        report "Stimulus process started" severity note;

        -- First test value
        s_hex <= "0000"; wait for 50 ns;
        assert (s_seg = "0000001")
        report "Test failed for input combination: 0000" severity error;


        s_hex <= "0001"; wait for 50 ns;
        assert (s_seg = "1001111")
        report "Test failed for input combination: 0000" severity error;

        s_hex <= "0010"; wait for 50 ns;
        assert (s_seg = "0010010")
        report "Test failed for input combination: 0000" severity error;
```

```
        s_hex <= "0011"; wait for 50 ns;
        assert (s_seg = "0000110")
        report "Test failed for input combination: 0000" severity error;

        s_hex <= "0100"; wait for 50 ns;
        assert (s_seg = "1001100")
        report "Test failed for input combination: 0000" severity error;

        s_hex <= "0101"; wait for 50 ns;
        assert (s_seg = "0100100")
        report "Test failed for input combination: 0000" severity error;

        s_hex <= "0110"; wait for 50 ns;
        assert (s_seg = "0100000")
        report "Test failed for input combination: 0000" severity error;

        s_hex <= "0111"; wait for 50 ns;
        assert (s_seg = "0001111")
        report "Test failed for input combination: 0000" severity error;

        s_hex <= "1000"; wait for 50 ns;
        assert (s_seg = "0000000")
        report "Test failed for input combination: 0000" severity error;

        s_hex <= "1001"; wait for 50 ns;
        assert (s_seg = "0000100")
        report "Test failed for input combination: 0000" severity error;

        s_hex <= "1010"; wait for 50 ns;
        assert (s_seg = "0001000")
        report "Test failed for input combination: 0000" severity error;

        s_hex <= "1011"; wait for 50 ns;
        assert (s_seg = "1100000")
        report "Test failed for input combination: 0000" severity error;

        s_hex <= "1100"; wait for 50 ns;
        assert (s_seg = "0110001")
        report "Test failed for input combination: 0000" severity error;

        s_hex <= "1101"; wait for 50 ns;
        assert (s_seg = "1000010")
        report "Test failed for input combination: 0000" severity error;

        s_hex <= "1110"; wait for 50 ns;
        assert (s_seg = "0110000")
        report "Test failed for input combination: 0000" severity error;

        s_hex <= "1111"; wait for 50 ns;
        assert (s_seg = "0111000")
        report "Test failed for input combination: 0000" severity error;



        report "Stimulus process finished" severity note;
```

```
        wait;
    end process p_stimulus;

end architecture testbench;
```






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

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;
```

```
entity top is
    Port ( SW : in std_logic_vector(4 - 1 downto 0);
           CA : out std_logic;
           CB : out std_logic;
           CC : out std_logic;
           CD : out std_logic;
           CE : out std_logic;
           CF : out std_logic;
           CG : out std_logic;
           AN : out std_logic_vector(8 - 1 downto 0);
           LED : out std_logic_vector(7 downto 0));
end top;
------------------------------------------------------------------------
-- Architecture body for top level
------------------------------------------------------------------------
architecture Behavioral of top is
begin

    ------------------------------------------------------------------------
    -- Instance (copy) of hex_7seg entity
    hex2seg : entity work.hex_7seg
        port map(
            hex_i    => SW,
            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
        );

    -- Connect one common anode to 3.3V
    AN <= b"1111_0111";

    -- Display input value on LEDs
    LED(3 downto 0) <= SW;


    -- LED(7:4) indicators
    -- Turn LED(4) on if input value is equal to 0, ie "0000"
    -- WRITE YOUR CODE HERE
    LED(4) <= '1' when SW = "0000" else
              '0';

    -- Turn LED(5) on if input value is greater than "1001", ie 9
    -- WRITE YOUR CODE HERE
    LED(5) <= '1' when SW >= "1001" else
              '0';

    -- Turn LED(6) on if input value is odd, ie 1, 3, 5, ...
    -- WRITE YOUR CODE HERE
    LED(6) <= '1' when SW(0) = '1' else
              '0';
```

```
    -- Turn LED(7) on if input value is a power of two, ie 1, 2, 4, or 8
    -- WRITE YOUR CODE HERE
    LED(7) <= '1' when ((SW = "0001") or (SW = "0010") or (SW = "0100") or (SW = "1000
              '0';

end architecture Behavioral;
```

3. LED(7:4) indicators.

Hex	Inputs	LED4	LED5	LED6	LED7
0	0000	1	0	0	0
1	0001	0	0	1	1
2	0010	0	0	0	1
3	0011	0	0	1	0
4	0100	0	0	0	1
5	0101	0	0	1	0
6	0110	0	0	0	0
7	0111	0	0	1	0
8	1000	0	0	0	1
9	1001	0	1	1	0
A	1010	0	1	0	0
b	1011	0	1	1	0
C	1100	0	1	0	0
d	1101	0	1	1	0
E	1110	0	1	0	0
F	1111	0	1	1	0

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

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
```

```
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity tb_top is
end entity tb_top;

architecture testbench of tb_top is

signal s_SW  : std_logic_vector(4 - 1 downto 0);
signal s_CA : std_logic;
signal s_CB : std_logic;
signal s_CC : std_logic;
signal s_CD : std_logic;
signal s_CE : std_logic;
signal s_CF : std_logic;
signal s_CG : std_logic;
signal s_AN  : std_logic_vector(8 - 1 downto 0);
signal s_LED  : std_logic_vector(7 downto 0);

begin

uut_top : entity work.top
       port map(
           SW => s_SW,
           CA => s_CA,
           CB => s_CB,
           CC => s_CC,
           CD => s_CD,
           CE => s_CE,
           CF => s_CF,
           CG => s_CG,
           AN => s_AN,
           LED => s_LED
        );

p_stimulus : process
   begin


report "Stimulus process started" severity note;

       -- First test value
       s_SW <= "0000"; wait for 50 ns;
       assert (s_LED(4) = '1')
       report "Test failed for input combination: 0000 the LED 4 does not turn on" se


       s_SW <= "1011"; wait for 50 ns;
       assert (s_LED(5) = '1')
       report "Test failed for input combination: 1011 the LED 5 does not turn on" se

       s_SW <= "0001"; wait for 50 ns;
       assert (s_LED(6) = '1')
       report "Test failed for input combination: 0001 the LED 6 does not turn on wit
```

```
        assert (s_LED(7) = '1')
        report "Test failed for input combination: 0001 the LED 1 does not turn on wit


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


end architecture testbench;
```