TaaviSalum ...	29 seconds ago
..	
Pictures	4 minutes ago
display	30 seconds ago
README.md	30 seconds ago

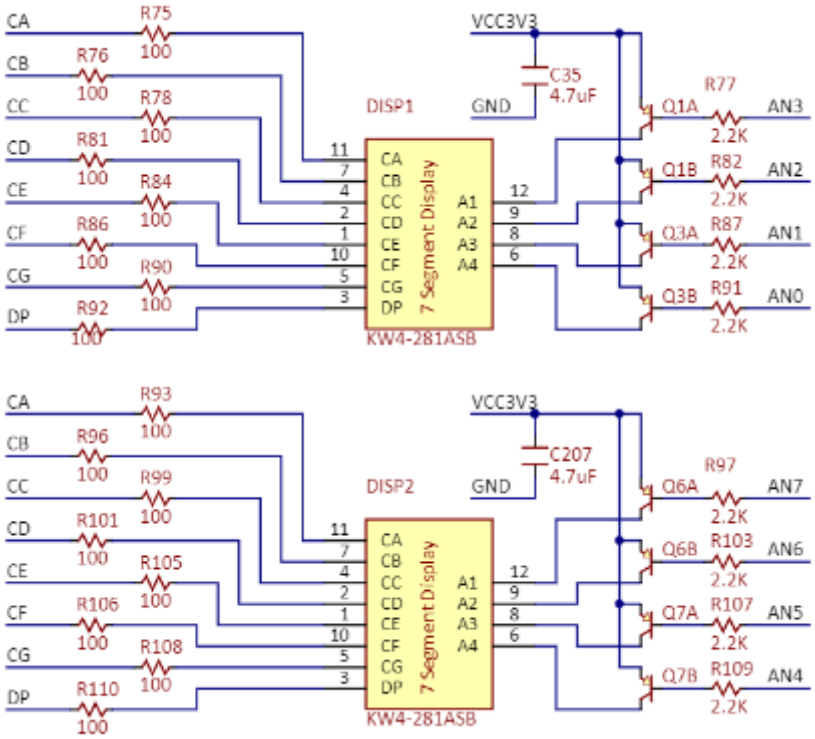
README.md

Digital-electronics-1

Laboratory #4

Exercise 1: Preparation tasks

Connection of 7-segment displays on Nexys A7 board



7-segment display truth table

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
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	1	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

Hex	Inputs	A	B	C	D	E	F	G
E	1110	0	1	1	0	0	0	0
F	1111	0	1	1	1	0	0	0

Exercise 2: Seven-segment display decoder

VHDL architecture from source file:

```
architecture Behavioral of hex_7seg is
begin

    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

            -- 9, A, B, C, D, E, F

            when "1001" =>
                seg_o <= "0001100";     -- 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 =>
```

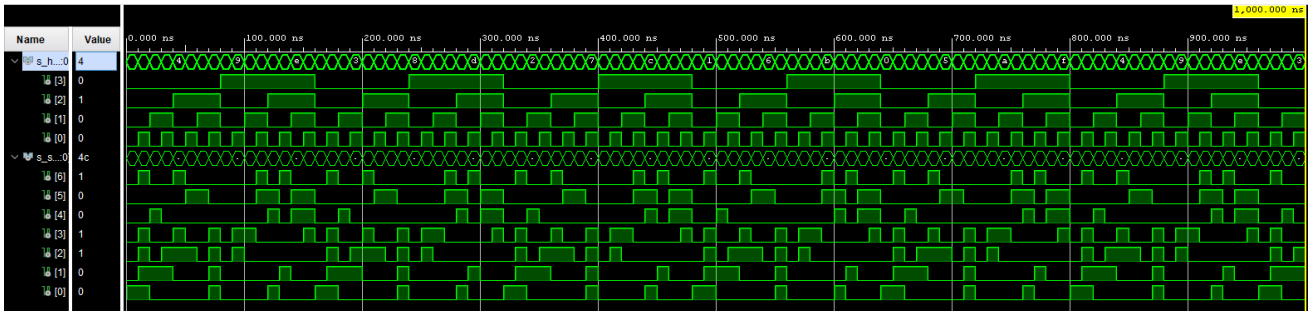
```
                seg_o <= "0111000";     -- F
        end case;
    end process p_7seg_decoder;

end architecture Behavioral;
```

VHDL stimulus process from testbench file:

```
    p_stimulus : process
    begin
        report "Stimulus process started" severity note;

        -- 0, 1, 2, 3, 4, 5, 6, 7, 8

        s_hex <= "0000"; wait for 10 ns;
        assert (s_seg = "0000001")
        report "Test failed for input: 0000 which is 0" severity error;

        s_hex <= "0001"; wait for 10 ns;
        assert (s_seg = "1001111")
        report "Test failed for input: 0001 which is 1" severity error;

        s_hex <= "0010"; wait for 10 ns;
        assert (s_seg = "0010010")
        report "Test failed for input: 0010 which is 2" severity error;

        s_hex <= "0011"; wait for 10 ns;
        assert (s_seg = "0000110")
        report "Test failed for input: 0011 which is 3" severity error;

        s_hex <= "0100"; wait for 10 ns;
        assert (s_seg = "1001100")
        report "Test failed for input: 0100 which is 4" severity error;

        s_hex <= "0101"; wait for 10 ns;
        assert (s_seg = "0100100")
        report "Test failed for input: 0101 which is 5" severity error;

        s_hex <= "0110"; wait for 10 ns;
        assert (s_seg = "0100000")
        report "Test failed for input: 0110 which is 6" severity error;

        s_hex <= "0111"; wait for 10 ns;
        assert (s_seg = "0001111")
        report "Test failed for input: 0111 which is 7" severity error;

        s_hex <= "1000"; wait for 10 ns;
        assert (s_seg = "0000000")
        report "Test failed for input: 1000 which is 8" severity error;

        -- 9, A, B, C, D, E, F
```

```
        s_hex <= "1001"; wait for 10 ns;
        assert (s_seg = "0001100")
        report "Test failed for input: 1001 which is 9" severity error;

        s_hex <= "1010"; wait for 10 ns;
        assert (s_seg = "0001000")
        report "Test failed for input: 1010 which is A" severity error;

        s_hex <= "1011"; wait for 10 ns;
        assert (s_seg = "1100000")
        report "Test failed for input: 1011 which is B" severity error;

        s_hex <= "1100"; wait for 10 ns;
        assert (s_seg = "0110001")
        report "Test failed for input: 1100 which is C" severity error;

        s_hex <= "1101"; wait for 10 ns;
        assert (s_seg = "1000010")
        report "Test failed for input: 1101 which is D" severity error;

        s_hex <= "1110"; wait for 10 ns;
        assert (s_seg = "0110000")
        report "Test failed for input: 1110 which is E" severity error;

        s_hex <= "1111"; wait for 10 ns;
        assert (s_seg = "0111000")
        report "Test failed for input: 111 which is F" severity error;

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

Simulated time waveforms



VHDL code from source file with 7-segment module instantiation:

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

entity top is
    Port ( SW : in STD_LOGIC_VECTOR (3 downto 0);
           LED : out STD_LOGIC_VECTOR (7 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 (7 downto 0));
end top;

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,

            -- WRITE YOUR CODE HERE
            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
    LED(3 downto 0) <= SW;
```

Exercise 3: LED(7:4) indicators

Truth table for LEDs(7:4)

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

Hex	Inputs	LED4	LED5	LED6	LED7
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	0	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

VHDL code for LEDs(7:4):

```
-- 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
           '0';

-- Turn LED(5) on if input value is greater than 9
-- WRITE YOUR CODE HERE
LED(5)  <= '1' when (SW = "1010") else  -- A (10)
           '1' when (SW = "1011") else  -- B (11)
           '1' when (SW = "1100") else  -- C (12)
           '1' when (SW = "1101") else  -- D (13)
           '1' when (SW = "1110") else  -- E (14)
           '1' when (SW = "1111") else  -- F (15)
           '0';

-- Turn LED(6) on if input value is odd, ie 1, 3, 5, ...
-- WRITE YOUR CODE HERE
LED(6)  <= '1' when (SW = "0001") else  -- 1
           '1' when (SW = "0011") else  -- 3
           '1' when (SW = "0101") else  -- 5
           '1' when (SW = "0111") else  -- 7
           '1' when (SW = "1001") else  -- 9
           '1' when (SW = "1011") else  -- B (11)
           '1' when (SW = "1101") else  -- D (13)
           '1' when (SW = "1111") else  -- F (15)
           '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") else  -- 1
               '1' when (SW = "0010") else  -- 2
               '1' when (SW = "0100") else  -- 4
               '1' when (SW = "1000") else  -- 8
               '0';
```

Simulated time waveforms