Zuzanaczm / **Digital-electronics-1**

Code Issues Pull requests Actions Projects Wiki Securit

main

Digital-electronics-1 / Labs / 04-segment / **readme.md**

Zuzanaczm Update readme.md **History**

1 contributor

Raw Blame

274 lines (206 sloc) 7.17 KB

Lab assignment - Zuzana Czmelová (04-segment)

Part 1 - Truth table

Hex	Input	A	B	C	D	E	F	G

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	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	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

Connection of 7-segment display on Nexys board

```
AN0 => J17                      CA => T10
AN1 => J18                      CB => R10
AN2 => T9                       CC => K16
AN3 => J14                      CD => K13
AN4 => P14                      CE => P15
AN5 => T14                      CF => T11
AN6 => K2                       CG => L18
AN7 => U13                      DP => H15
```

Both anode and cathode signals are active-low.

Part 2 - Seven-segment display decoder

A) VHDL design source

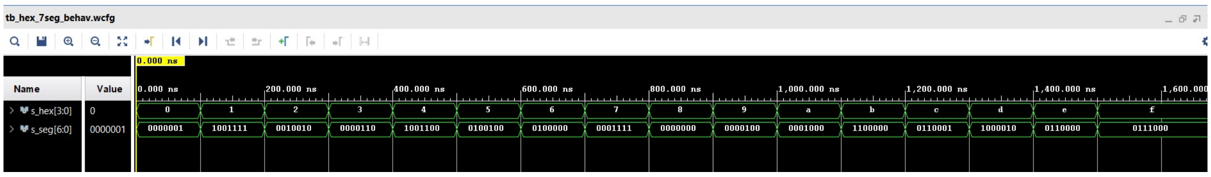
```
architecture Behavioral of hex_7seg is
begin

    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" =>              --6
                seg_o <= "0100000";

            when "0111" =>              --7
                seg_o <= "0001111";

            when "1000" =>             --8
                seg_o <= "0000000";

            when "1001" =>             --9
                seg_o <= "0000100";

            when "1010" =>             --A
                seg_o <= "0001000";

            when "1011" =>             --b
                seg_o <= "1100000";

            when "1100" =>            --C
                seg_o <= "0110001";

            when "1101" =>             --d
                seg_o <= "1000010";
            when "1110" =>
                seg_o <= "0110000";     -- E

            when others =>
                seg_o <= "0111000";     -- F
        end case;
    end process p_7seg_decoder;

end Behavioral;
```

B) VHDL testbench

```
p_stimulus : process
  begin

      report "Stimulus process started" severity note;

     s_hex <= "0000";
      wait for 100ns ;

     s_hex <= "0001";
       wait for 100ns ;

     s_hex <= "0010";
      wait for 100ns ;

     s_hex <= "0011";
      wait for 100ns ;

     s_hex <= "0100";
      wait for 100ns ;

     s_hex <= "0101";
      wait for 100ns ;

     s_hex <= "0110";
      wait for 100ns ;

     s_hex <= "0111";
     wait for 100ns ;

     s_hex <= "1000";
     wait for 100ns ;

      s_hex <= "1001";
      wait for 100ns ;

     s_hex <= "1010";
     wait for 100ns ;
```

```
        s_hex <= "1011";
        wait for 100ns ;

        s_hex <= "1100";
         wait for 100ns ;

        s_hex <= "1101";
        wait for 100ns ;

        s_hex <= "1110";
        wait for 100ns ;

         s_hex <= "1111";
        wait for 100ns ;

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

C) Screenshot of simulation



```
s_hex[3:0]  0        0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | a | b | c | d | e | f
s_seg[6:0]  0000001  0000001 | 1001111 | 0010010 | 0000110 | 1001100 | 0100100 | 0100000 | 0001111 | 0000000 | 0000100 | 0001000 | 1100000 | 0110001 | 1000010 | 0110000 | 0111000
```

D) VHDL code from TOP.vhd

```
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
       );
```

Part 3 - LED (7:4) indicators

A) Truth table

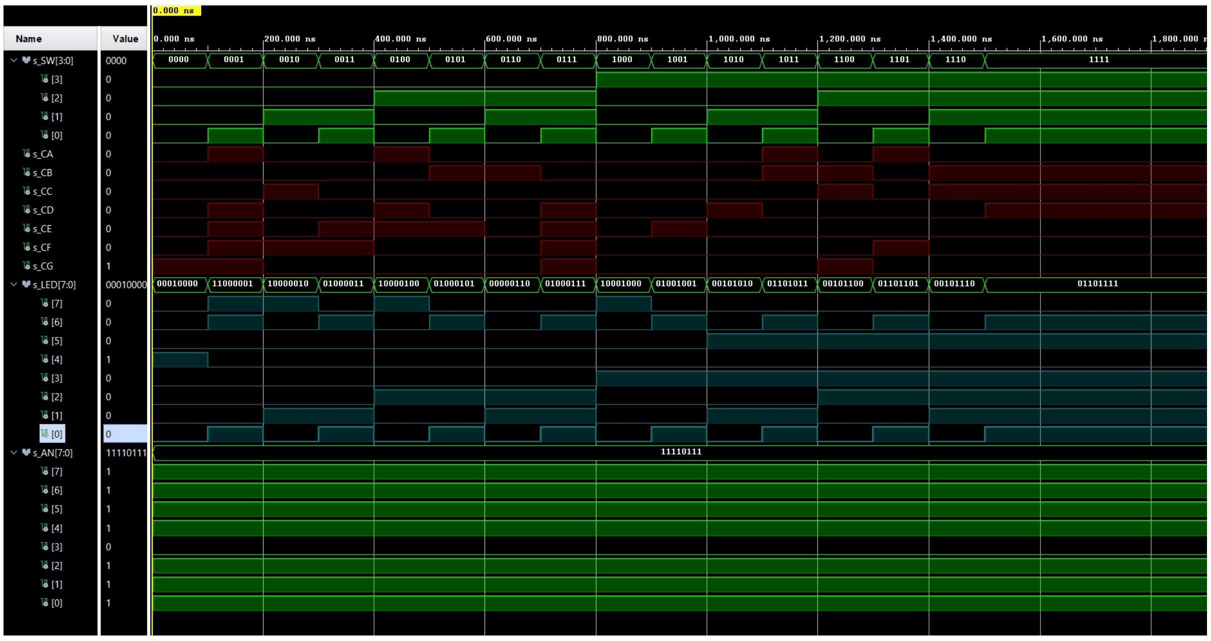
Hex	Inputs	LED4	LED5	LED6	LED7
0	0000	1	0	0	0
1	0001	0	0	1	1
2	0010	0	0	0	1
3	0011	0	0	1	0
4	0100	0	0	0	1
5	0101	0	0	1	0
6	0110	0	0	0	0
7	0111	0	0	1	0
8	1000	0	0	0	1
9	1001	0	0	1	0
A	1010	0	1	0	0
b	1011	0	1	1	0
C	1100	0	1	0	0
d	1101	0	1	1	0
E	1110	0	1	0	0
F	1111	0	1	1	0

B) VHDL code for LEDs

```
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;
           LED : out STD_LOGIC_VECTOR (8-1 downto 0);
```

```
            AN : out STD_LOGIC_VECTOR (8-1 downto 0)

    );


end top;

architecture Behavioral of top is

begin


    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";

    LED(4 - 1 downto 0) <= SW; -- Display input value

    -- Turn LED(4) on if input value is equal to 0, ie "0000"
    LED(4) <= '1' when (SW = "0000") else '0';

    -- Turn LED(5) on if input value is greater than "1001"
    LED(5) <= '1' when (SW > "1001") else '0';

    -- Turn LED(6) on if input value is odd, ie 1, 3, 5, ...
    LED(6) <= SW(0);

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

end Behavioral;
```

C) Screenshot of simulation


0.000 ns
Name
Value
s_SW[3:0] 0000
[3] 0
[2] 0
[1] 0
[0] 0
s_CA 0
s_CB 0
s_CC 0
s_CD 0
s_CE 0
s_CF 0
s_CG 1
s_LED[7:0] 00010000
[7] 0
[6] 0
[5] 0
[4] 1
[3] 0
[2] 0
[1] 0
[0] 0
s_AN[7:0] 11110111
[7] 1
[6] 1
[5] 1
[4] 1
[3] 0
[2] 1
[1] 1
[0] 1
0.000 ns
200.000 ns
400.000 ns
600.000 ns
800.000 ns
1,000.000 ns
1,200.000 ns
1,400.000 ns
1,600.000 ns
1,800.000 ns
0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
00010000 11000001 10000010 01000011 10000100 01000101 00000110 01000111 10001000 01001001 00101010 01101011 00101100 01101101 00101110 01101111
11110111