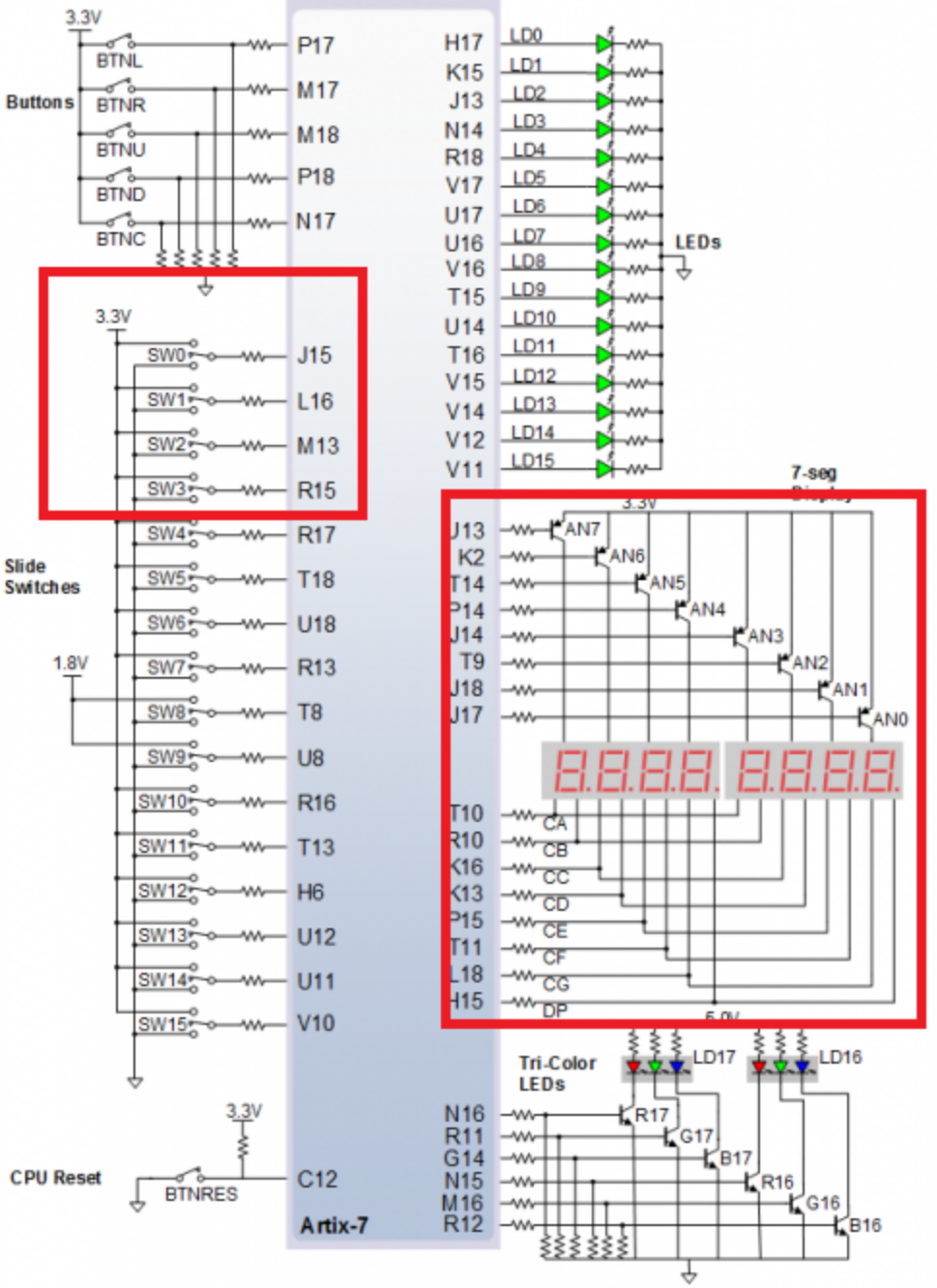
# Digital-electronics-1

## Lab assignment 3)

### Preparation tasks:




Decoder truth table for common anode 7-segment display.

| Hex | Inputs | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|---|

| 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 | 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

### Architecture code (hex_7seg.vhd)

```
    p_7seg_decoder : process(hex_i)
        begin
```

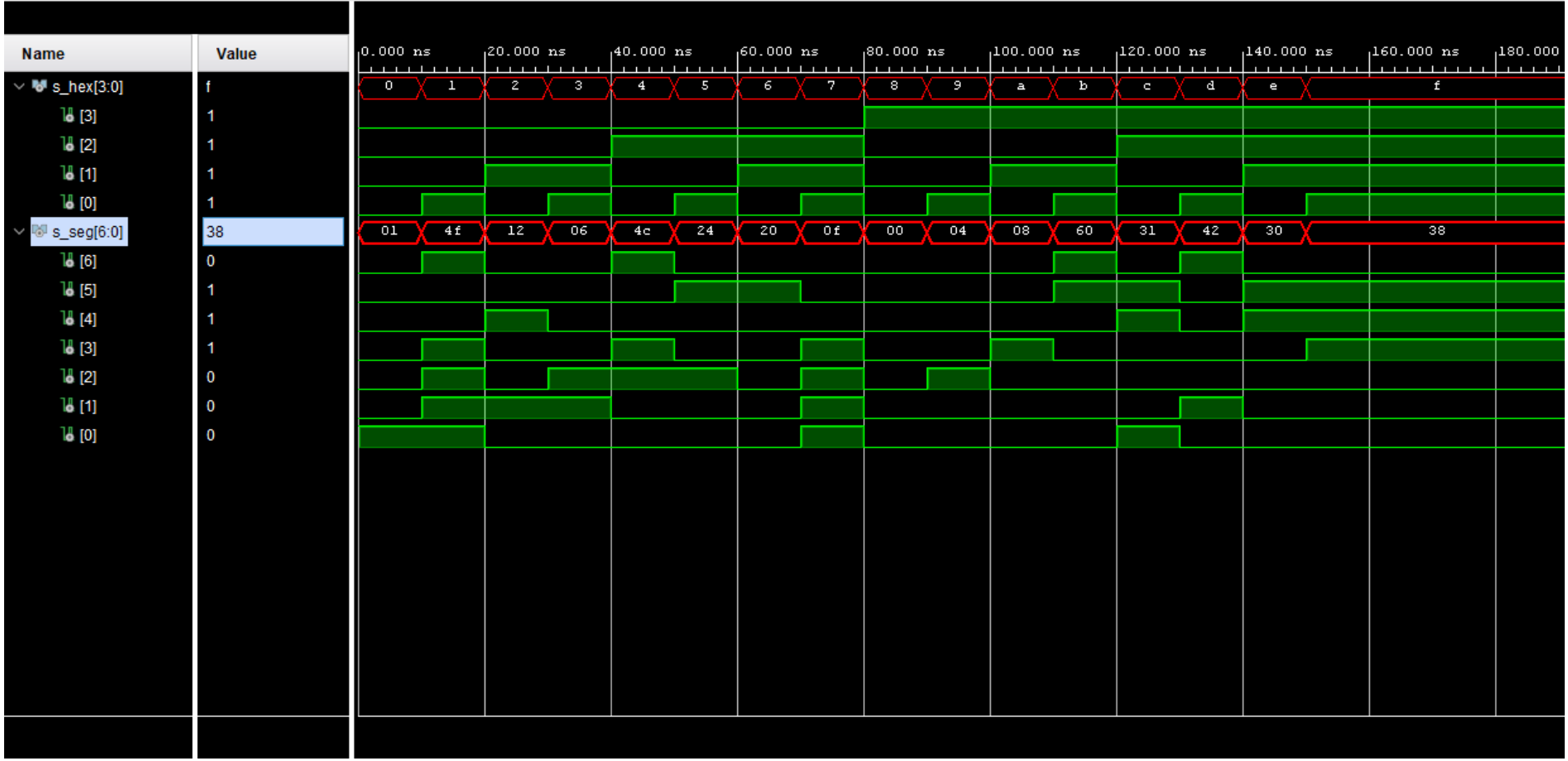
```
            case hex_i is
                when "0000" =>
                    seg_o <= "0000001";     -- 0
                when "0001" =>
                    seg_o <= "1001111";     -- 1
                when "0010" =>
                    seg_o <= "0010010";     -- 2
                when "0011" =>
                    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;
```

### Testbench code (tb_hex_7seg.vhd)

```
  p_stimulus : process
      begin
          -- Report a note at the begining of stimulus process
```

```
        report "Stimulus process started. ---------------------------------------" severity note;

        s_hex <= "0000";    wait for 10 ns;       -- 0

        s_hex <= "0001";    wait for 10 ns;       -- 1

        s_hex <= "0010";    wait for 10 ns;       -- 2

        s_hex <= "0011";    wait for 10 ns;       -- 3

        s_hex <= "0100";    wait for 10 ns;       -- 4

        s_hex <= "0101";    wait for 10 ns;       -- 5

        s_hex <= "0110";    wait for 10 ns;       -- 6

        s_hex <= "0111";    wait for 10 ns;       -- 7

        s_hex <= "1000";    wait for 10 ns;       -- 8

        s_hex <= "1001";    wait for 10 ns;       -- 9

        s_hex <= "1010";    wait for 10 ns;       -- A

        s_hex <= "1011";    wait for 10 ns;       -- B

        s_hex <= "1100";    wait for 10 ns;       -- C

        s_hex <= "1101";    wait for 10 ns;       -- D

        s_hex <= "1110";    wait for 10 ns;       -- E

        s_hex <= "1111";    wait for 10 ns;       -- F

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

### Simulated waveform




## Top 7-segment module instantiation (top.vhd)

```
    -- 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

## LED[7:4] truth table

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.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);
        LED :   out std_logic_vector(16 - 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)
    );
end top;

architecture Behavioral of top is
    signal  s_seg_o :   std_logic_vector(7 - 1 downto 0);
begin
    -- Instance (copy) of hex_7seg entity
```

```
    hex2seg : entity work.hex_7seg
        port map(
            hex_i       => SW,
            seg_o       => s_seg_o
        );

    -- Connect one common anode to 3.3V
    AN  <= b"1111_1110";

    CA  <= s_seg_o(6);
    CB  <= s_seg_o(5);
    CC  <= s_seg_o(4);
    CD  <= s_seg_o(3);
    CE  <= s_seg_o(2);
    CF  <= s_seg_o(1);
    CG  <= s_seg_o(0);

    LED(15 downto 9)    <= not s_seg_o;

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

    -- 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 9
    LED(5)              <= '1' when (unsigned(SW) > 9) else '0';

    -- Turn LED(6) on if input value is odd, ie 1, 3, 5, ...
    LED(6)              <= '1' when (unsigned(SW) mod 2 = 1) else '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" or SW = "0010" or SW = "0100" or SW = "1000") else '0';

end Behavioral;
```

| Hex | Inputs | LED[7] | LED[6] | LED[5] | LED[4] |
|---|---|---|---|---|---|
| 0 | 0000 | 0 | 0 | 0 | 1 |
| 1 | 0001 | 1 | 1 | 0 | 0 |

| Hex | Inputs | LED[7] | LED[6] | LED[5] | LED[4] |
|---|---|---|---|---|---|
| 2 | 0010 | 1 | 0 | 1 | 0 |
| 3 | 0011 | 0 | 1 | 0 | 0 |
| 4 | 0100 | 1 | 0 | 0 | 0 |
| 5 | 0101 | 0 | 1 | 0 | 0 |
| 6 | 0110 | 0 | 0 | 0 | 0 |
| 7 | 0111 | 0 | 1 | 0 | 0 |
| 8 | 1000 | 1 | 0 | 0 | 0 |
| 9 | 1001 | 0 | 1 | 0 | 0 |
| A | 1010 | 0 | 0 | 1 | 0 |
| b | 1011 | 0 | 1 | 1 | 0 |
| C | 1100 | 0 | 0 | 1 | 0 |
| d | 1101 | 0 | 1 | 1 | 0 |
| E | 1110 | 0 | 0 | 1 | 0 |
| F | 1111 | 0 | 1 | 1 | 0 |