Lab 4: Seven-segment display decoder

Preparation tasks

Nexys A7 anode table

- 7-seg displays have common anode each.
- Logic 0 value allows the current to flow through a transistor to 7-seg display.
- Logic 1 value prevents the current to flow through a transistor to 7-seg display.
- In the DISPLAY column, the value 0 means rightmost and 7 means leftmost.

AND	PIN	DISPLAY
AN0	J17	0
AN1	J18	1
AN2	T9	2
AN3	J14	3
AN4	P14	4
AN5	T14	5
AN6	K2	6
AN7	U13	7

Nexys A7 cathode table

- Logic 0 value allows the current to flow through the LEDs in 7-seg display.
- Logic 1 value prevents the current to flow through the LEDs in 7-seg display.

CTD	PIN
CA	T10

CTD	PIN
CB	R10
CC	K16
CD	K13
CE	P15
CF	T11
CG	L18
DP	H15

Decoder 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	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

Hex	Inputs	A	B	C	D	E	F	G
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

Seven-segment display decoder

VHDL architecture from source file `hex_7seg.vhd`

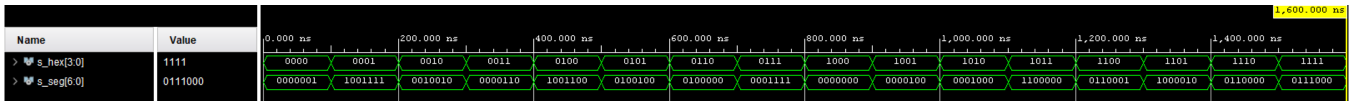
```
architecture Behavioral of hex_7seg is

begin

    --------------------------------------------------------------------
    -- p_7seg_decoder:
    -- A combinational process for 7-segment display decoder.
    -- Any time "hex_i" is changed, the process is "executed".
    -- Output pin seg_o(6) corresponds to segment A, seg_o(5) to B, etc.
    --------------------------------------------------------------------
    p_7seg_decoder : process(hex_i)
    begin
        case hex_i is
            when "0000" =>
                seg_o <= "0000001";     -- 0
            when "0001" =>
                seg_o <= "1001111";     -- 1

            -- WRITE YOUR CODE HERE
            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 Behavioral;
```

VHDL stimulus process from testbench file `tb_hex_7seg.vhd`

```
p_stimulus : process
    begin
        -- Report a note at the begining of stimulus process
        report "Stimulus process started" severity note;


        -- First test values
        s_hex <= "0000"; wait for 100 ns;

        -- WRITE OTHER TESTS HERE
        s_hex <= "0001"; wait for 100 ns;
        s_hex <= "0010"; wait for 100 ns;
        s_hex <= "0011"; wait for 100 ns;
        s_hex <= "0100"; wait for 100 ns;
        s_hex <= "0101"; wait for 100 ns;
        s_hex <= "0110"; wait for 100 ns;
        s_hex <= "0111"; wait for 100 ns;
        s_hex <= "1000"; wait for 100 ns;
        s_hex <= "1001"; wait for 100 ns;
        s_hex <= "1010"; wait for 100 ns;
        s_hex <= "1011"; wait for 100 ns;
        s_hex <= "1100"; wait for 100 ns;
        s_hex <= "1101"; wait for 100 ns;
        s_hex <= "1110"; wait for 100 ns;
        s_hex <= "1111"; wait for 100 ns;


        -- Report a note at the end of stimulus process
        report "Stimulus process finished" severity note;
        wait;
    end process p_stimulus;
```

Simulated time waveforms



VHDL code from source file `top.vhd` with 7-segment module instantiation

```
  -- 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
          );
```

LED(7:4) indicators

Truth table

Hex	Inputs	LED4	LED5	LED6	LED7
0	0000	1	0	0	0

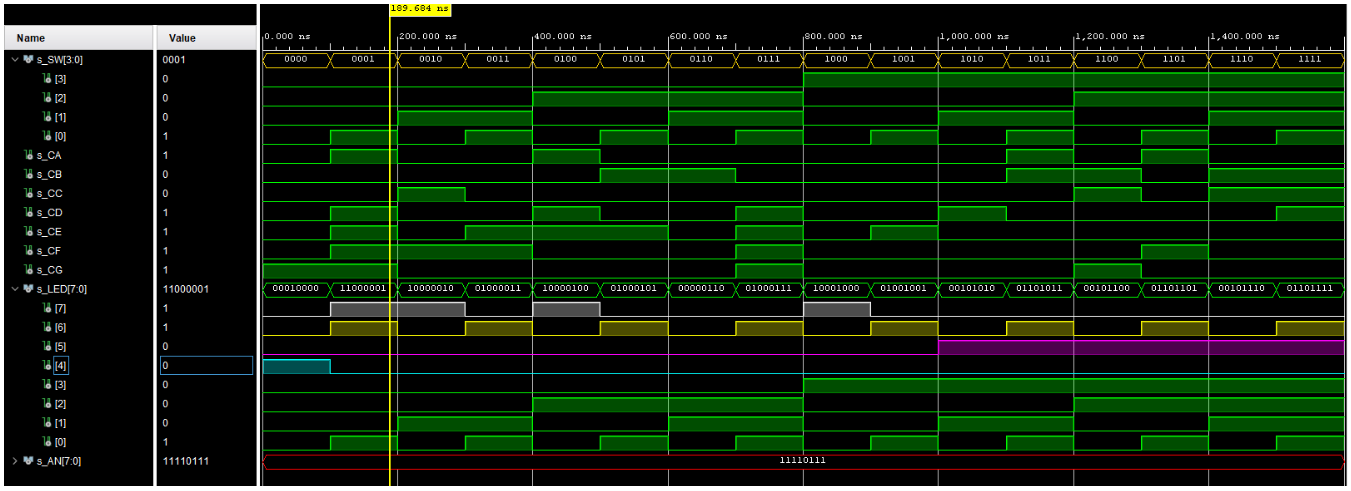
Hex	Inputs	LED4	LED5	LED6	LED7
1	0001	0	0	1	1
2	0010	0	0	0	1
3	0011	0	0	1	0
4	0100	0	0	0	1
5	0101	0	0	1	0
6	0110	0	0	0	0
7	0111	0	0	1	0
8	1000	0	0	0	1
9	1001	0	0	1	0
A	1010	0	1	0	0
b	1011	0	1	1	0
C	1100	0	1	0	0
d	1101	0	1	1	0
E	1110	0	1	0	0
F	1111	0	1	1	0

VHDL code for LEDs(7:4)

```
-- 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) <= (SW(3) and SW(1)) or (SW(3) and SW(2));

-- Turn LED(6) on if input value is odd, ie 1, 3, 5, ...
-- WRITE YOUR CODE HERE
```

```
              LED(6) <= SW(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")
    else
                        '0';
```

Simulated time waveforms


189.684 ns
Name
Value
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
s_SW[3:0] 0001
[3] 0
[2] 0
[1] 0
[0] 1
s_CA 1
s_CB 0
s_CC 0
s_CD 1
s_CE 1
s_CF 1
s_CG 1
s_LED[7:0] 11000001
[7] 1
[6] 1
[5] 0
[4] 0
[3] 0
[2] 0
[1] 0
[0] 1
s_AN[7:0] 11110111
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