جامعة نيويورك أبوظبي

NYU ABU DHABI

Advanced Digital Logic

ENGR – UH 2310

Lab 3

Group 1

Spring, 2025

This report is entirely our own work, and we have kept a copy for our own records. We are aware of the University's policies on cheating, plagiarism, and the resulting consequences of their breach.

Submitted by:

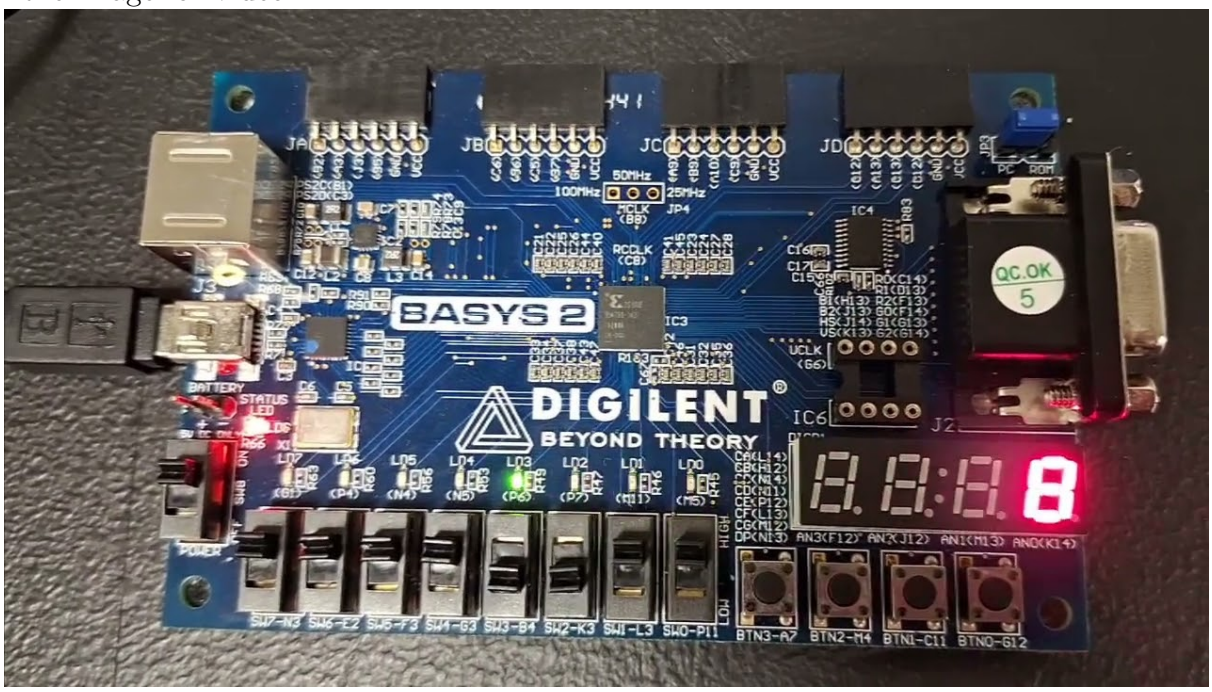
Name	**Net ID**
Damiane Kapanadze	dk4770
Seoyoon Jung	sj4260
Sipan Hovsepian	sh7437

Contents

1 Results:

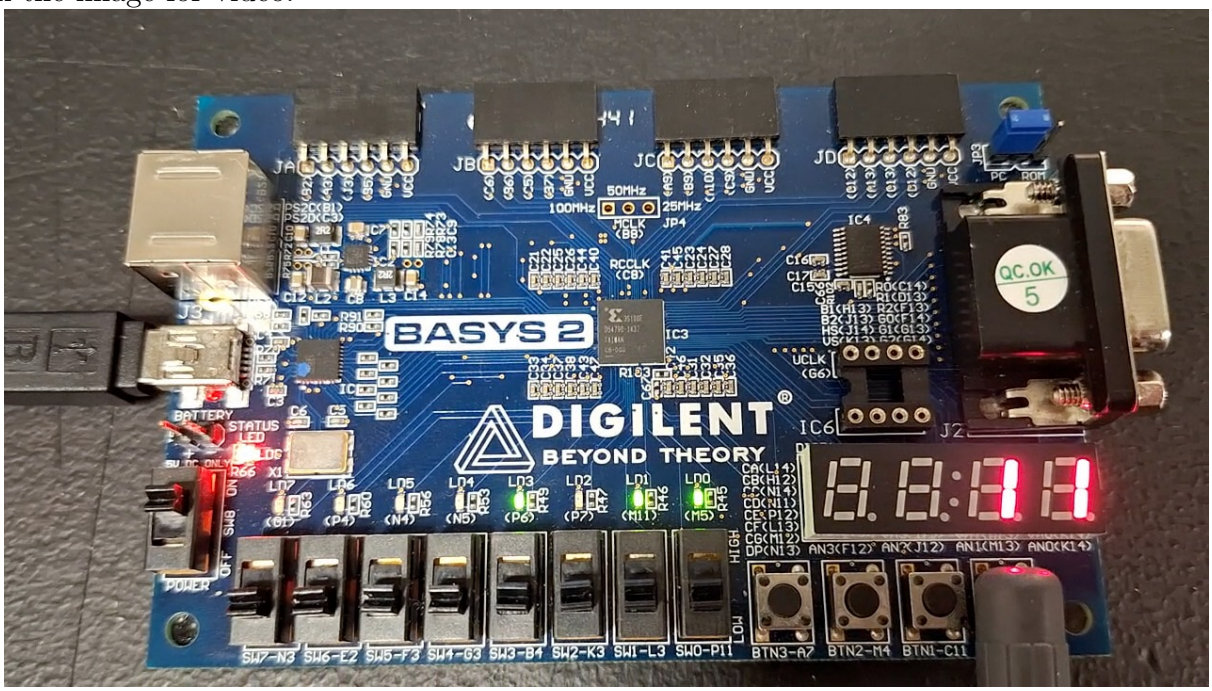
1.1 Task 1:

click the image for video:



1.2 Task 2:

click the image for video:



2 VHDL code of Task 1 modules

2.1 mainTask1 module

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.NUMERIC_STD.ALL;
4
5
6  entity mainTask1 is
7      Port (  num2Display : out  STD_LOGIC_VECTOR (3 downto 0); --
           output for LED representation
8            Reset : in  STD_LOGIC;
9            clk: in STD_LOGIC;
10           PO_seg : out  STD_LOGIC_VECTOR (0 to 7); -- output
              for each segment display
11           PO_an : out  STD_LOGIC_VECTOR (3 downto 0)); --
              output for choice of display
12 end mainTask1;
13
14
15 architecture Behavioral of mainTask1 is
16
17 --Use component single_number as a decoder
18 component single_number is
19     Port (  number : in  STD_LOGIC_VECTOR (3 downto 0); -- 4-bit
           representation of 1 to F
20           seg : out  STD_LOGIC_VECTOR (0 to 7)); -- decoded 8-
              bit representation to be displayed on seven-seg
21 end component;
22
23 -- clock divider signals
24 constant cnt_max : integer := 5*1e7; -- clock is at 50MHZ, so we
      need 5* 10^7 clock cycles to make 1Hz (=1cycle/sec)
25 signal clk_cnt : integer range 0 to cnt_max;
26
27 -- Internal signals
28 signal number2disp: STD_LOGIC_VECTOR (3 downto 0);
29 signal segment: STD_LOGIC_VECTOR (7 downto 0);
30
31
32 begin
33 -- Component instance (port => signal)
34 single_number_instance: single_number
35     port map(    number => number2disp,
36                  seg => segment);
37
38 -- Process to iterate through seg_mode
39 seg_mode_switch : process (clk, Reset)
40     begin
41         if rising_edge(clk) then
```

```
42              if (Reset = '1') then -- if reseting
43                  number2disp <= "0000";
44                  -- the reset button may not be instant upon
                       pressing since it is reflected upon each
                       cyclewe are checking it in every 1 second
45                  -- this issue is fixed in Task 2 by using another
                        variable
46              end if;
47
48              if (clk_cnt = cnt_max) then -- increment number every
                    1 second
49                  if(number2disp = "1111") then -- if max number
                       (15), reset to 0
50                      number2disp <= "0000";
51                  else -- if not, increment by 1
52                      number2disp <= std_logic_vector (unsigned(
                           number2disp) + 1);
53                  end if;
54                  -- display number
55                  num2Display <= number2disp;
56                  PO_an <= "1110";
57                  PO_seg <= segment;
58                  clk_cnt <= 0; --reset clock timer
59              else
60              clk_cnt <= clk_cnt + 1;
61                  end if;
62          end if;
63  end process;
64
65  end Behavioral;
```

Listing 1: VHDL code for mainTask1 module

2.2 Single number module

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity single_number is
5      Port (  number : in  STD_LOGIC_VECTOR (3 downto 0); -- 4-bit
           input vector
6              seg : out  STD_LOGIC_VECTOR (0 to 7)); -- 8-bit
                  output vector
7  end single_number;
8
9  architecture Behavioral of single_number is
10
11 begin
12 -- purpose: decodes compoted ALU result into binary for 7-seg
       display
13 process(number)
```

```
    begin
        case number is
                -- A_B_C_D_E_F_G_DP; 0=on, 1=off, DP 0=negative
            when "0000" => seg <= "00000011"; -- 0
            when "0001" => seg <= "10011111"; -- 1
            when "0010" => seg <= "00100101"; -- 2
            when "0011" => seg <= "00001101"; -- 3
            when "0100" => seg <= "10011001"; -- 4
            when "0101" => seg <= "01001001"; -- 5
            when "0110" => seg <= "01000001"; -- 6
            when "0111" => seg <= "00011111"; -- 7
            when "1000" => seg <= "00000001"; -- 8
            when "1001" => seg <= "00001001"; -- 9
            when "1010" => seg <= "00010001"; -- A
            when "1011" => seg <= "11000001"; -- b
            when "1100" => seg <= "01100011"; -- C
            when "1101" => seg <= "10000101"; -- d
            when "1110" => seg <= "01100001"; -- E
            when "1111" => seg <= "01110001"; -- F
            when others => seg <= "11111111"; -- All segments off
                for undefined input
        end case;
    end process;
end Behavioral;
```

Listing 2: VHDL code for single$_n$*umbermodule*

2.3 Constraints file

```
NET "PO_seg<0>" LOC = "L14"; # Bank = 1, Signal name = CA
NET "PO_seg<1>" LOC = "H12"; # Bank = 1, Signal name = CB
NET "PO_seg<2>" LOC = "N14"; # Bank = 1, Signal name = CC
NET "PO_seg<3>" LOC = "N11"; # Bank = 2, Signal name = CD
NET "PO_seg<4>" LOC = "P12"; # Bank = 2, Signal name = CE
NET "PO_seg<5>" LOC = "L13"; # Bank = 1, Signal name = CF
NET "PO_seg<6>" LOC = "M12"; # Bank = 1, Signal name = CG
NET "PO_seg<7>" LOC = "N13"; # Bank = 1, Signal name = DP

NET "PO_an<3>" LOC = "K14"; # Bank = 1, Signal name = AN3
NET "PO_an<2>" LOC = "M13"; # Bank = 1, Signal name = AN2
NET "PO_an<1>" LOC = "J12"; # Bank = 1, Signal name = AN1
NET "PO_an<0>" LOC = "F12"; # Bank = 1, Signal name = AN0

NET "clk" LOC = "B8";

#LEDS
NET "num2Display<3>" LOC = "P6";
NET "num2Display<2>" LOC = "P7";
NET "num2Display<1>" LOC = "M11";
NET "num2Display<0>" LOC = "M5";
#Reset button
```

```
23 NET "Reset" LOC = "G12";
```

Listing 3: Constraints file for Task 1

3 Task 2

3.1 mainTask2 module

```
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 primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity mainTask2 is
    Port ( num2Display : out  STD_LOGIC_VECTOR (3 downto 0); -- output for LED representation
           Reset : in  STD_LOGIC;
           clk: in STD_LOGIC;
           PO_seg : out  STD_LOGIC_VECTOR (0 to 7); -- output for each segment display
           PO_an : out  STD_LOGIC_VECTOR (3 downto 0)); -- output for choice of display unit
end mainTask2;


architecture Behavioral of mainTask2 is

--Use component single_number as a decoder
component single_number is
    Port ( number : in  STD_LOGIC_VECTOR (3 downto 0); -- 4-bit representation of 1 to 9
           seg : out  STD_LOGIC_VECTOR (0 to 7)); -- decoded 8-bit representation to be displayed on seven-seg
end component;

-- Clock divider signals
constant cnt_max : integer := 5*1e7; -- clock is at 50MHz, so we need 5*10^7 clock cycles
signal clk_cnt : integer range 0 to cnt_max; -- for keeping track of 1 sec cycle
signal seg_mode, seg_mode_new : integer range 0 to 3; -- selection of seven-segment unit
signal clk_cnt_sync : integer range 0 to 1e5; -- for visually synchronizing digits

--Internal signals
signal number2disp: STD_LOGIC_VECTOR (3 downto 0);
signal segment: STD_LOGIC_VECTOR (7 downto 0);
```

```
39
40
41 begin
42 --Component instance (port => signal)
43 single_number_instance: single_number
44     port map(    number => number2disp,
45                  seg => segment);
46
47 --Process to track time
48 seg_mode_switch : process (clk, Reset, seg_mode_new)
49 begin
50     if rising_edge(clk) then
51         -- visually synchronize digits
52         if (clk_cnt_sync = 1e5) then
53             seg_mode <= seg_mode_new;
54             clk_cnt_sync <=0;
55             -- if reset button,
56             if (Reset = '1') then
57                 number2disp <= "0000";
58             end if;
59         else
60             clk_cnt_sync <= clk_cnt_sync + 1;
61         end if;
62
63         -- Increment number
64         if (clk_cnt = cnt_max) then
65             clk_cnt <= 0;
66             if (number2disp = "1111") then -- if it hits maximum
                   number (15), reset to 0
67                 number2disp <= "0000";
68             else -- if not, increment by 1
69                 number2disp <= std_logic_vector (unsigned(
                       number2disp) + 1);
70             end if;
71         else clk_cnt <= clk_cnt + 1;
72         end if;
73
74 end if;
75 end process;
76
77 -- Process to generate output on the displays
78 display : process (seg_mode, segment, number2disp)
79 begin
80     if (seg_mode = 1) then -- display the tens
81         PO_an <= "1101"; -- activated unit
82         num2Display <= "0000"; -- to avoid latches, in reality
               this doens't matter since we are overriding the Po_seg
                anyway
83         if (unsigned(number2disp) > 9) then
84             -- activated segments (PO_seg) for tens is set as 1
                   or 0 based on value
```

```
85              PO_seg <= "10011111"; -- if two-digits number, show 1
86          else
87              PO_seg <= "00000011"; -- if one-digit number, show 0
88          end if;
89
90          seg_mode_new <= 0;
91
92      elsif (seg_mode = 0) then  -- display the ones
93          PO_an <= "1110"; -- activated unit
94          num2Display <= number2disp; -- display the number
95          PO_seg <= segment; -- activated segments (PO_seg) for
               ones follows the number counts
96          seg_mode_new <= 1;
97      else
98          num2Display <= "0000"; -- to avoid latches
99      end if;
100
101 end process;
102 end Behavioral;
```

Listing 4: main module for Task 2

3.2 single number

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity single_number is
5      Port (  number : in  STD_LOGIC_VECTOR (3 downto 0);
6              seg : out  STD_LOGIC_VECTOR (0 to 7));
7  end single_number;
8
9  architecture Behavioral of single_number is
10
11 begin -- seg => segment (here, segment is the ones digit)
12     process(number)
13     begin
14         case number is
15         -- A_B_C_D_E_F_G_DP; 0=on, 1=off, DP 0=negative
16             when "0000" => seg <= "00000011"; -- 0
17             when "0001" => seg <= "10011111"; -- 1
18             when "0010" => seg <= "00100101"; -- 2
19             when "0011" => seg <= "00001101"; -- 3
20             when "0100" => seg <= "10011001"; -- 4
21             when "0101" => seg <= "01001001"; -- 5
22             when "0110" => seg <= "01000001"; -- 6
23             when "0111" => seg <= "00011111"; -- 7
24             when "1000" => seg <= "00000001"; -- 8
25             when "1001" => seg <= "00001001"; -- 9
26             when "1010" => seg <= "00000011"; -- 0  we can
                  directly display the last digit of the number
```

```
                instead of changing the value of number
            when "1011" => seg <= "10011111"; -- 1
            when "1100" => seg <= "00100101"; -- 2
            when "1101" => seg <= "00001101"; -- 3
            when "1110" => seg <= "10011001"; -- 4
            when "1111" => seg <= "01001001"; -- 5
            when others => seg <= "11111111"; -- 6 All segments
                off for undefined input
        end case;
    end process;
end Behavioral;
```

Listing 5: single_number module

3.3 Constraints file

```
NET "PO_seg<0>" LOC = "L14"; # Bank = 1, Signal name = CA
NET "PO_seg<1>" LOC = "H12"; # Bank = 1, Signal name = CB
NET "PO_seg<2>" LOC = "N14"; # Bank = 1, Signal name = CC
NET "PO_seg<3>" LOC = "N11"; # Bank = 2, Signal name = CD
NET "PO_seg<4>" LOC = "P12"; # Bank = 2, Signal name = CE
NET "PO_seg<5>" LOC = "L13"; # Bank = 1, Signal name = CF
NET "PO_seg<6>" LOC = "M12"; # Bank = 1, Signal name = CG
NET "PO_seg<7>" LOC = "N13"; # Bank = 1, Signal name = DP

NET "PO_an<3>" LOC = "K14"; # Bank = 1, signal name = AN3
NET "PO_an<2>" LOC = "M13"; # Bank = 1, Signal name = AN2
NET "PO_an<1>" LOC = "J12"; # Bank = 1, Signal name = AN1
NET "PO_an<0>" LOC = "F12"; # Bank = 1, Signal name = AN0

NET "clk" LOC = "B8";

#LEDS
NET "num2Display<3>" LOC = "P6";
NET "num2Display<2>" LOC = "P7";
NET "num2Display<1>" LOC = "M11";
NET "num2Display<0>" LOC = "M5";

#Reset button
NET "Reset" LOC = "G12";
```

Listing 6: Constraints file for Task 2 is the same as Task 1