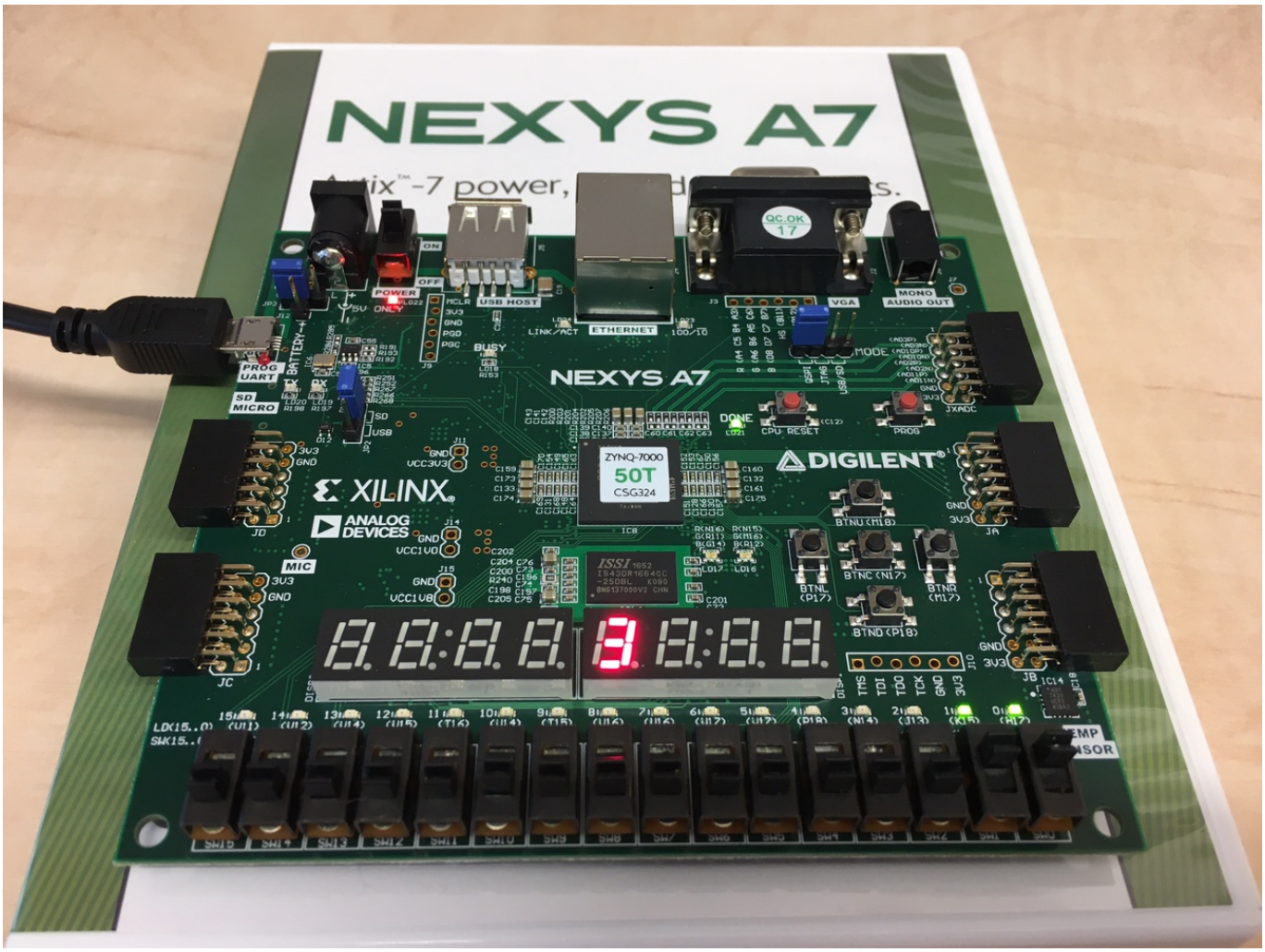
Lab 4: Seven-segment display decoder

EUROPEAN UNION
European Structural and Investment Funds
Operational Programme Research,
Development and Education

MINISTRY OF EDUCATION,
YOUTH AND SPORTS

Learning objectives

The purpose of this laboratory exercise is to design a 7-segment display decoder and to become familiar with the VHDL structural description that allows you to build a larger system from simpler or predesigned components.



Preparation tasks (done before the lab at home)

The Nexys A7 board provides two four-digit common anode seven-segment LED displays (configured to behave like a single eight-digit display). See schematic or reference manual of the Nexys A7 board and find out the connection of 7-segment display.

Complete the decoder conversion table for common anode 7-segment display.

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								
3								
4								
5								
6								
7								
8								
9								
A								
b								
C								
d								
E	1110	0	1	1	0	0	0	0
F	1111	0	1	1	1	0	0	0

Part 1: Synchronize repositories and create a new folder

Run Git Bash (Windows) of Terminal (Linux) and synchronize local and remote repositories. Create a new working folder `Labs/04-segment` for this exercise.

Part 2: VHDL code for seven-segment display decoder

Perform the following steps to simulate the seven-segment display decoder.

1. Create a new Vivado project in your `Labs/04-segment` working folder.

2. Create a VHDL source file `hex_7seg` for the decoder.
3. Choose default board: `Nexys A7-50T`.
4. In source file, define an entity `hex_7seg` as follows.

Port name	Direction	Type	Description
`hex_i`	input	`std_logic_vector(4 - 1 downto 0)`	Input binary data
`seg_o`	output	`std_logic_vector(7 - 1 downto 0)`	Cathode values in the order A, B, C, D, E, F, G

5. Use combinational process and define an architecture of the decoder. Note that, inside a process, `case-when` assignments can be used.

```
------------------------------------------------------------------------
-- Architecture body for seven-segment display decoder
------------------------------------------------------------------------
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 "1110" =>
                seg_o <= "0110000";     -- E
            when others =>
                seg_o <= "0111000";     -- F
        end case;
    end process p_7seg_decoder;

end architecture behavioral;
```

6. Create a simulation source `tb_hex_7seg` and verify the functionality of your decoder.

Part 3: Top level VHDL code

VHDL provides a mechanism how to build a larger system from simpler or predesigned components. It is called an instantiation. Each instantiation statement creates an instance (copy) of a design entity.

VHDL-93 and later offers two methods of instantiation: direct instantiation and component instantiation. In direct instantiation, the entity itself is directly instantiated in an architecture. Its ports are connected using the port map. Let the top-level design `top.vhd`, implements an instance of the module defined in `hex_7seg.vhd`.

Perform the following steps to implement the seven-segment display decoder on the Nexys A7 board.

1. Create a new design source `top` in your project.
2. Define an entity `top` as follows.

Port name	Direction	Type	Description
`SW`	input	`std_logic_vector(4 - 1 downto 0)`	Input binary data
`LED`	output	`std_logic_vector(8 - 1 downto 0)`	LED indicators
`CA`	output	`std_logic`	Cathod A
`CB`	output	`std_logic`	Cathod B
`CC`	output	`std_logic`	Cathod C
`CD`	output	`std_logic`	Cathod D
`CE`	output	`std_logic`	Cathod E
`CF`	output	`std_logic`	Cathod F
`CG`	output	`std_logic`	Cathod G
`AN`	output	`std_logic_vector(8 - 1 downto 0)`	Common anode signals to individual displays

3. Create a new constraints XDC file: `nexys-a7-50t` and uncomment used pins according to the entity.
4. Use direct instantiation and define an architecture of the top level.

```
----------------------------------------------------------------------------
-- Architecture body for top level
----------------------------------------------------------------------------
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(0) => CG
        );

    -- Connect one common anode to 3.3V
    AN <= b"1111_0111";

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

    -- Turn LED(4) on if input value is equal to "0000"
    -- WRITE YOUR CODE HERE

    -- Turn LED(5) on if input value is A, b, C, d, E, or F
    -- WRITE YOUR CODE HERE

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

    -- Turn LED(7) on if input value is a power of two, ie 1, 2, 4, or 8
    -- WRITE YOUR CODE HERE

end architecture behavioral;
```

5. Compile the project and download the generated bitstream into the FPGA chip.
6. Test the functionality of the seven-segment display decoder by toggling the switches and observing the display and LEDs.
7. Use **IMPLEMENTATION > Open Implemented Design > Schematic** to see the generated structure.

Synchronize repositories

Use git commands to add, commit, and push all local changes to your remote repository. Check the repository at GitHub web page for changes.

Experiments on your own

1. Write logic functions for LEDs(7:4) according to comments in source code above. Use VHDL construction `when-else` or low-level gates `and`, `or`, and `not`.

Lab assignment

1. Seven-segment display decoder. Submit:
    - VHDL code of the decoder (`hex_7seg.vhd`),
    - VHDL testbench (`tb_hex_7seg.vhd`).
2. LED indicators. Submit:
    - VHDL code for LEDs(7:4).

The deadline for submitting the task is the day before the next laboratory exercise. Use BUT e-learning web page and submit a single PDF file.