

Lab 4: Seven-segment display decoder



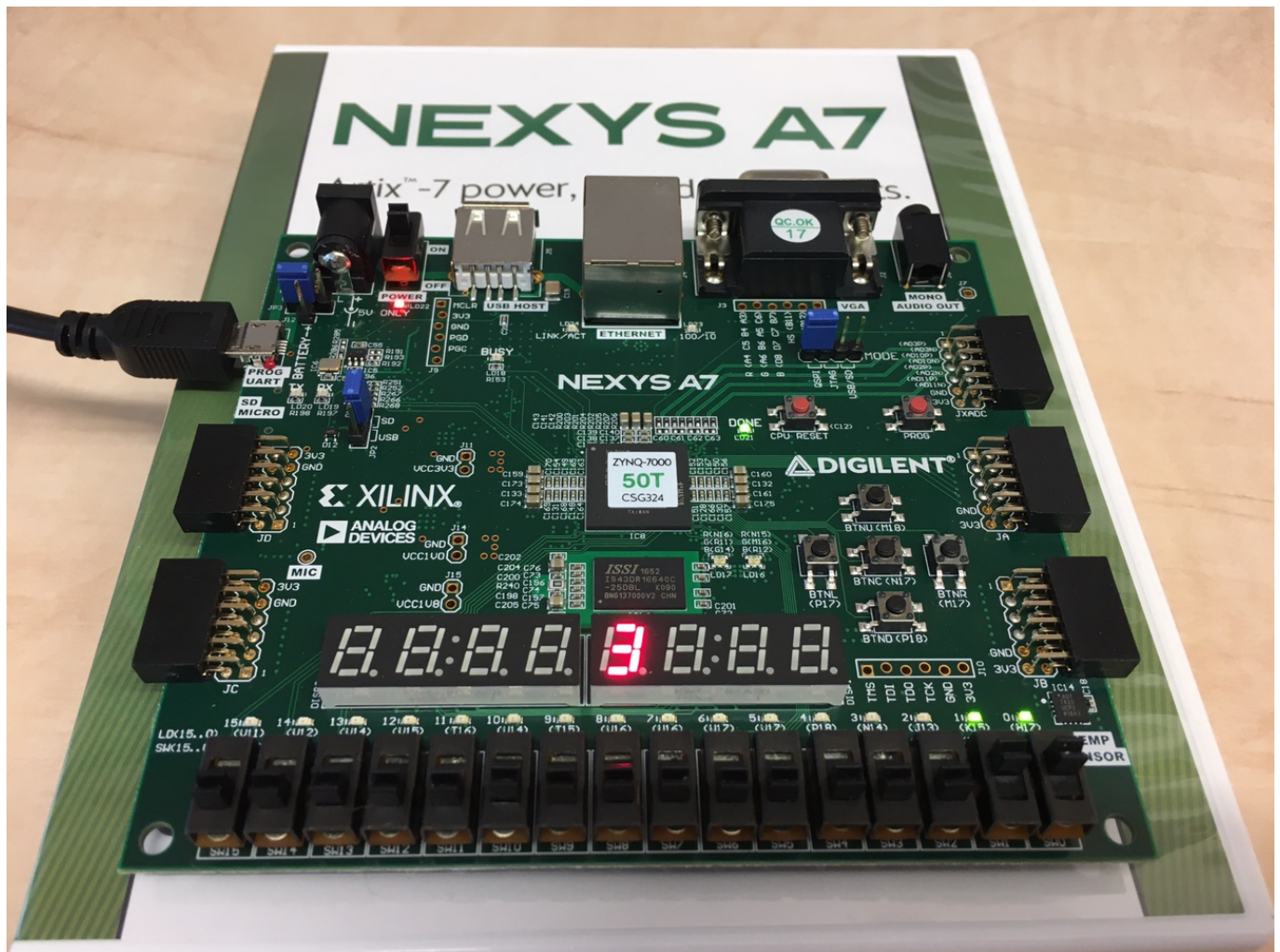
EUROPEAN UNION
European Structural and Investment Funds
Operational Programme Research,
Development and Education



The Study of Modern and Developing Engineering BUT
CZ.02.2.69/0.0/0.0/18_056/0013325

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 displays, ie to which FPGA pins are connected and how.

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

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								
3								
4								
5								
6								
7								
8	1000	0	0	0	0	0	0	0
9								
A								
b								
C								
d								
E	1110	0	1	1	0	0	0	0
F	1111	0	1	1	1	0	0	0



The image above was used from a website: [How Seven Segment Display Works & Interface it with Arduino](#).

Part 1: Synchronize repositories and create a new folder

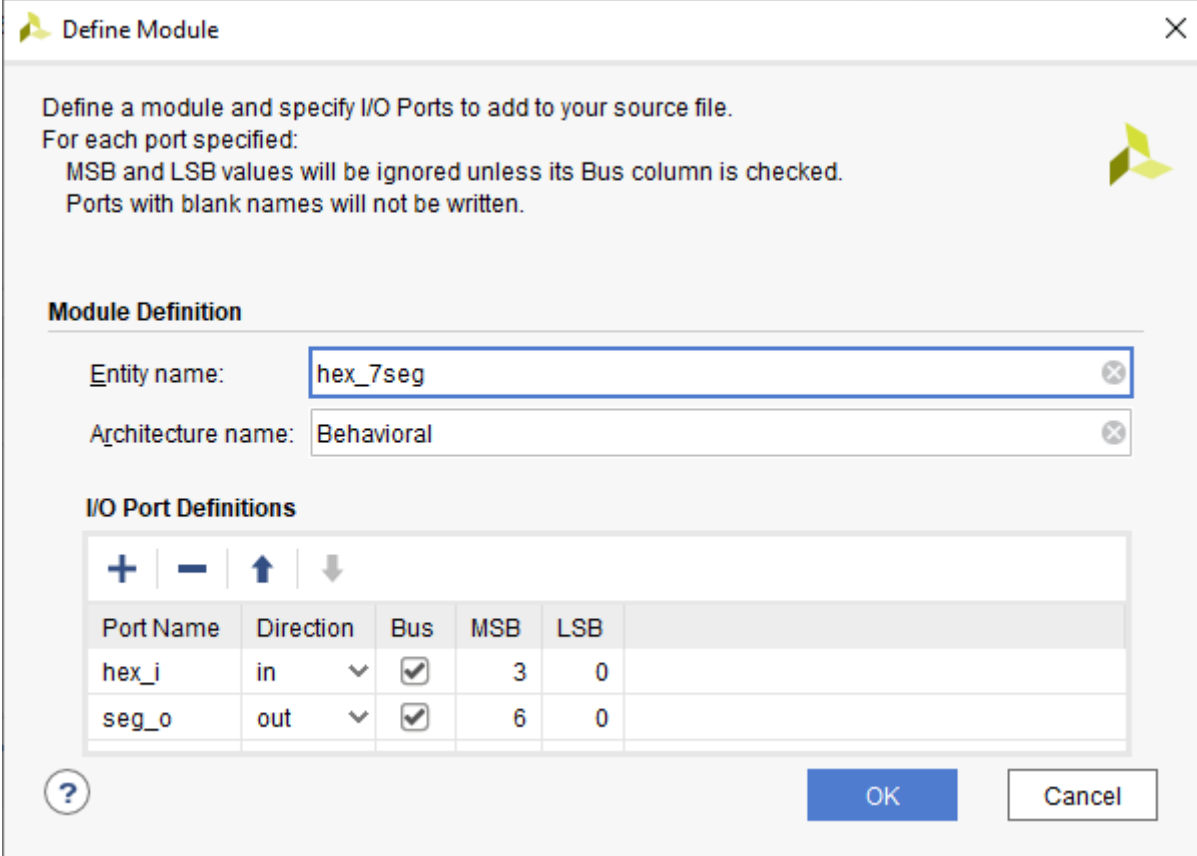
Run Git Bash (Windows) or Terminal (Linux), navigate to your working directory, and update local repository. Create a new working folder `Labs/04-segment` for this laboratory exercise.

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 RTL project `display` 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. Use **Define Module** dialog and define I/O ports of entity `hex_7seg` as follows.

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



Define a module and specify I/O Ports to add to your source file.
For each port specified:
MSB and LSB values will be ignored unless its Bus column is checked.
Ports with blank names will not be written.

Module Definition

Entity name:

Architecture name:

I/O Port Definitions

Port Name	Direction	Bus	MSB	LSB
hex_i	in	<input checked="" type="checkbox"/>	3	0
seg_o	out	<input checked="" type="checkbox"/>	6	0

OK Cancel

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 (Common Anode)
    -- decoder. Any time "hex_i" is changed, the process is "executed".
    -- Output pin seg_o(6) controls segment A, seg_o(5) segment B, etc.
    --
    --      segment A
    --      | segment B
    --      | | segment C
    --      +-+| | ... segment G
    --      ||+-+ |
    --      ||| |
    -- seg_o = "0000001"-----+
    -----

    p_7seg_decoder : process(hex_i)
    begin
        case hex_i is
            when "0000" =>
                seg_o <= "0000001";      -- 0
            when "0001" =>
                seg_o <= "1001111";      -- 1
        end case;
    end process;
end architecture;

```

```

-- WRITE YOUR CODE HERE
-- 2, 3, 4, 5, 6, 7

when "1000" =>
    seg_o <= "00000000";    -- 8

-- WRITE YOUR CODE HERE
-- 9, A, b, C, d

when "1110" =>
    seg_o <= "01100000";    -- E
when others =>
    seg_o <= "01110000";    -- F
end case;
end process p_7seg_decoder;

end architecture Behavioral;

```

6. Create a VHDL [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	<code>std_logic_vector(4 - 1 downto 0)</code>	Input binary data
CA	output	<code>std_logic</code>	Cathod A
CB	output	<code>std_logic</code>	Cathod B
CC	output	<code>std_logic</code>	Cathod C

Port name	Direction	Type	Description
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
LED	output	std_logic_vector(8 - 1 downto 0)	LED indicators

3. 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 on LEDs
    LED(3 downto 0) <= SW;

    -- LED(7:4) indicators
    -- Turn LED(4) on if input value is equal to 0, ie "0000"
    -- WRITE YOUR CODE HERE

    -- Turn LED(5) on if input value is greater than "1001", ie 10, 11, 12,
    ...
    -- WRITE YOUR CODE HERE

    -- Turn LED(6) on if input value is odd, ie 1, 3, 5, ...

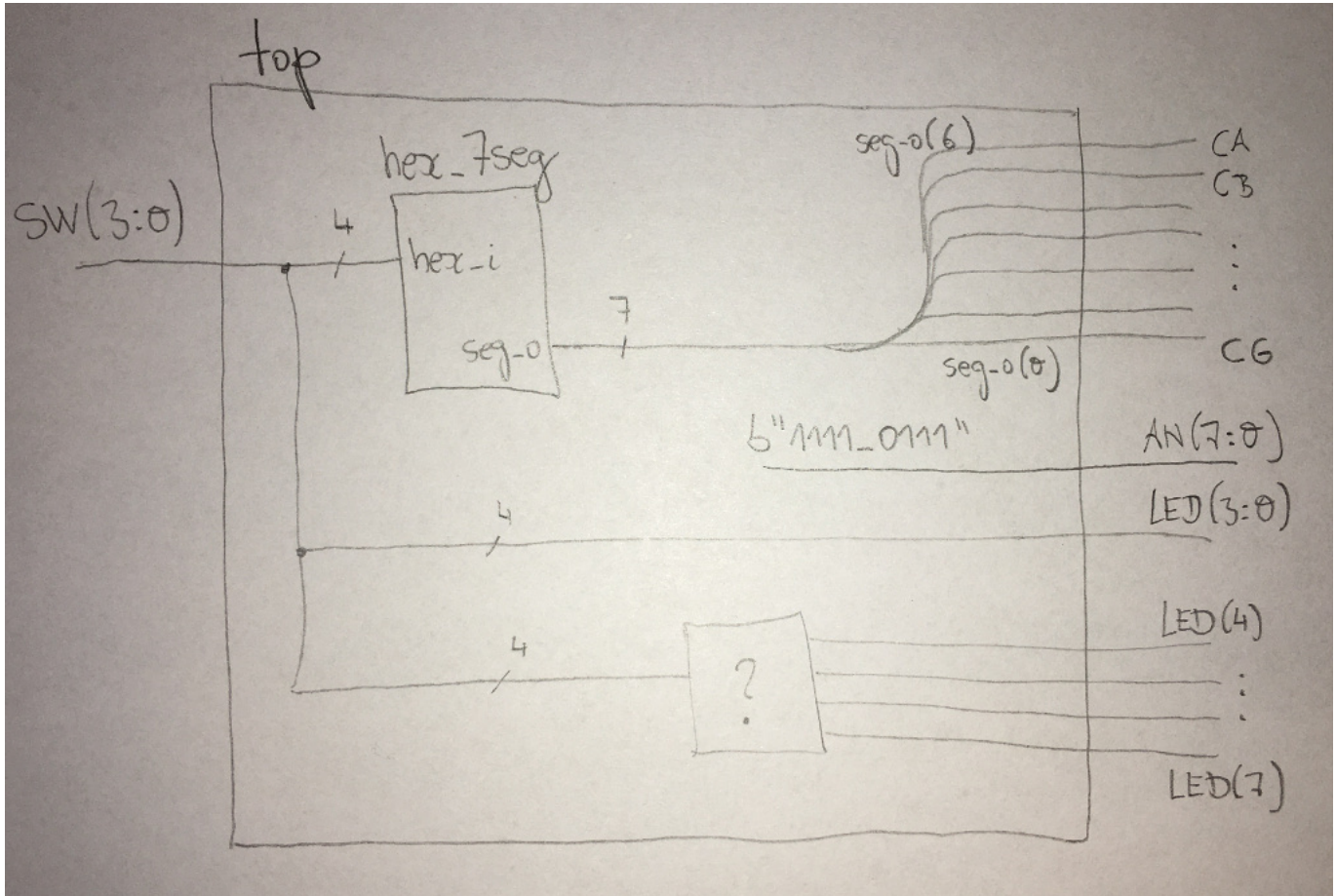
```



```
-- 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;
```



4. Create a new [constraints XDC](#) file: `nexys-a7-50t` and uncomment used pins according to the `top` entity.
5. Compile the project and download the generated bitstream `display/display.runs/impl_1/top.bit` into the FPGA chip.
6. Test the functionality of the seven-segment display decoder by toggling the switches and observing the display and LEDs. Change the binary value `AN <= b"0111_1111"` and observe its effect on the display selection.
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. Complete the truth table for LEDs according to comments in source code above. Use VHDL construction `when-else` or low-level gates `and`, `or`, and `not` and write logic functions for LED(7:4)

indicators.

Hex	Inputs	LED4	LED5	LED6	LED7
0	0000				
1	0001				
2					
3					
4					
5					
6					
7					
8	1000				
9					
A					
b					
C					
d					
E	1110				
F	1111				

Lab assignment

1. Preparation tasks (done before the lab at home). Submit:

- Figure or table with connection of 7-segment displays on Nexys A7 board,
- Decoder truth table for common anode 7-segment display.

2. Seven-segment display decoder. Submit:

- Listing of VHDL architecture from the source file `hex_7seg.vhd` with syntax highlighting,
- Listing of VHDL stimulus process from the testbench file `tb_hex_7seg.vhd` with syntax highlighting and asserts,
- Screenshot with simulated time waveforms; always display all inputs and outputs. The full functionality of the entities must be verified,
- VHDL code listing of the source file `top.vhd` with 7-segment module instantiation.

3. LED(7:4) indicators. Submit:

- Truth table and listing of VHDL code for LEDs(7:4) with syntax highlighting,

- Screenshot with simulated time waveforms; always display all inputs and outputs. The full functionality of the entities must be verified.

Prepare all parts of the assignment on a computer (not by hand), insert them in your README file [Digital-electronics-1/Labs/04-segment/README.md](#), export the formatted output (not the listing in markdown language) from [HTML to PDF](#), use [BUT e-learning](#) web page and submit a single PDF file. The deadline for submitting the task is the day before the next laboratory exercise.