D0011E Lab 2 (Design and verification of logic circuits in VHDL)

The goal of this lab is to implement simple logic in VHDL and to verify own designs using simulations. This lab must be completed in groups of two.

All the VHDL code together with a README file will be uploaded to GitLab. You can commit the whole Vivado project to facilitate reviewers in their task. The README file will include the table of the preparation and the answers to the questions in this document, include the text and the index of the question for it to be traced.

Preparation (to be completed BEFORE the lab session):

Complement the truth table from lab 1 with the function hieq3: 1 if (x>=3) and (x<=9), 0 otherwise.

Input (x)	max	min	even	lo3	noBCD	hieq3
0						
1						
2						
3						
4						
5						
6						
7						
8						
9						
10 (A)						
11 (B)						
12 (C)						
13 (D)						
14 (E)						
15 (F)						

Obtain the Boolean equation for the new signal hieq3 as a function of the inputs *x0*, *x1*, *x2*, *x3*.

Part 1.

a) Modify *bcdcheck2.vhd* from lab 1 by adding the output *hieq3*, determined directly from the input signals. Change the entity name to BCDcheck3. Save your code as *bcdcheck3.vhd* and add it to the project using the menu Project > Add Copy of Source… (if you saved the file outside the project directory) or Project > Add Source (if you saved the file inside the project directory).

   Simulate and verify your design, make sure to test all possible inputs (you can modify the test bench *bcdcheck2_tb.vhd* to run the simulations).

   Now you can answer the following questions:

   - What types for signals are predefined in VHDL?
   - What are the differences between an Integer and a Std_logic in VHDL?

b) A problem with the solution in 1a is an unnecessary duplication of comparators, you can see the circuit in "Open Elaborated Design". To avoid this, modify *bcdcheck3.vhd* so that *hieq3* is determined by reusing the comparators from the calculations of *lo3* and *noBCD*. Change the entity name to BCDcheck4 and save your code as *bcdcheck4.vhd*. Simulate and verify your design, make sure to test all possible inputs. To check that you have reused the comparator, reload the "Open Elaborated Design".

Hint: Since you cannot read output signals in VHDL, you will need to introduce intermediate signals *lo3_int* and *noBCD_int* (internal to the architecture of *BCDcheck4*) and then use them to assign the output signals *hieq3*, *lo3*, and *noBCD*. If you see more comparators than before, try using the code as in *bcdcheck1.vhd* and use a NOR gate instead.

In this solution you removed comparators but added other logic gates. At the end the RTL schematic does not show much savings in space. Was it then a good decision to remove the comparator?

c) Try to move the assignment of *hieq3* by placing it above and below the assignments of the intermediate signals *lo3_int* and *noBCD_int*. Simulate your design. **Does this change affect the behavior of the entity, i.e. does the order of assignments in VHDL matter?**

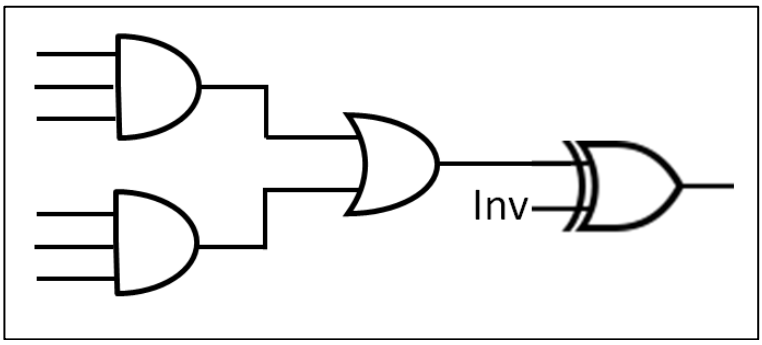
Part 2.

a) Design a VHDL component that corresponds to the logic function $f = bc'd' + a'd' + ac' + a'c$ (without making any optimizations). Generate RTL schematic. Then perform the synthesis of your design and open the synthesized schematic to compare. Has the synthesis tool optimized the design? How?

b) Write a truth table for the function *f* and design a test bench to verify the component from 2a. Verify your design by comparing the simulation results to the truth table.

Part 3.

When designing hardware, we often need to express certain logic using a grid of PLD cells (PLD = programmable logic device). In this assignment, you will express the function *f* from 2a using the following PLD cell:



a) Write a VHDL component corresponding to the schematic of the PLD cell above. Use this interface:

```
entity PLDcell is
  port(x5, x4, x3, x2, x1, x0, inv : in STD_LOGIC;
       y : out STD_LOGIC);
end entity PLDcell;
```

b) Design a VHDL component for the logic function $f = bc'd' + a'd' + ac' + a'c$ that uses a *single* PLD cell from 3a and no other logical circuits. You may, however, invert the input signals (*a*,*b*,*c*,*d*) before sending them to the PLD cell or set its inputs to constants 0 or 1. Use the test bench from 2b to simulate and verify your design.

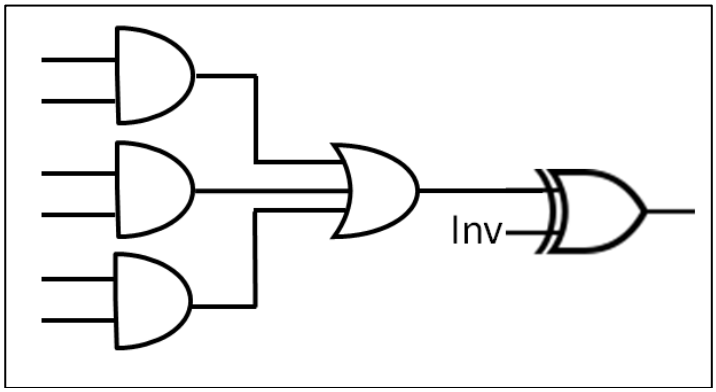
Hint: **you will need to perform Karnaugh minimization of the function *f*.**

c) Design a VHDL component for the logic function $g = a'b'c' + abcde$ that uses *two* PLD cells from

3a. You may invert the input signals (*a,b,c,d, e*) before sending them to the PLD cell or set its inputs to constants (0, 1). You may also send the output from one cell as an input to another cell. Simulate and verify your design, make sure to test all possible inputs.

Part 4.

Let *x = (x3, x2, x1, x0)* be a BCD encoding of a number from 0 to 9. Design a VHDL component that takes *x3, x2, x1, x0* as inputs and produces *y3, y2, y1, y0* as outputs, where *y = (y3, y2, y1, y0) = 9 – x*, i.e. *y* is the 9-complement of *x*. Use exactly four PLD cells depicted below, one for each bit of *y*. You may *not* invert the input signals before sending them to the PLD cells but you may set their inputs to constants 0 or 1. Design a testbench to test your design, as you did in the previous parts.



Hint: write 4-bit encodings for each value of *x* and the corresponding value of *y*. Use these to formulate separate logic functions for each bit of *y*, using only *x1*, *x2*, *x3*, and *x4* as inputs. You should also use the fact that *x* can only be a number from 0 to 9, i.e. not all combinations of bits are possible as inputs.

Part 5 Upload the Vivado project and the README file created during the lab, remember:

- **Update the truth table with the new signal**
- **Answer the questions in part 1 & 2**
- **Upload the truth table for function f**
- **Simplification of f in part 3 and schematic implementation of f and g on the PLDcell**
- **Schematic implementation of y = 9-x in part 4 on the PLDcell**