**Only Logisim version 3.7.2 and 3.8.0 are supported for this lab.**
**Submission deadline: 31.03.2023 23:59**

In this lab, you will design a base 5 modulo $125_{10}$ 3-digit counter. Binary encodings of the base 5 digits are given in the table below.

| Digit | Code |
|---|---|
| $0_{B=5}$ | $010_b$ |
| $1_{B=5}$ | $101_b$ |
| $2_{B=5}$ | $111_b$ |
| $3_{B=5}$ | $001_b$ |
| $4_{B=5}$ | $100_b$ |

Table 1: Binary encoding of digits.

Unused binary sequences are considered *don't cares.*

Please download the template file called *tp*34*.circ* from the moodle page of the course and use it for your design. The top level module of the file is named $TP34$. You must not change that name nor make any other module top. You must also neither modify nor rename any of the components existing in the template file. Finally, you must not change the mapping of any of the components on the FPGA board. This also means that you must not add more LEDs, buttons, etc.

Besides the components that are already in the template, you can also use NOT, AND, OR, NAND, NOR, XOR, and XNOR gates, D-Flipflops, any wiring, and subcircuits that you created out of these components. If you use any other components, the total number of points awarded to your design will be halved.

The solution .circ file is to be submitted for automated grading at the following link: `https://digsys.epfl.ch`. You are allowed to make ten submissions, out of which only the last one will be counted. Functionality of each of the questions below assumes functionality of all the previous questions. Hence, the grader will check for functionality of the questions in the reverse order, and award the points for the first correctly answered one and all the preceding ones. For example, if the solution successfully solves the problem posed in Question 3, you will get all points for Questions 1, 2, and 3. Note that it is only necessary to upload the file containing the solution of the last question that you have solved.

**A Note on Using the Grader**
Besides producing a brief report with the scores awarded, the grader outputs two plain text files for each graded question: *golden_qN.txt* and

*trace_qN.txt*, where $N$ is the question number. Both files contain the values of the output signals for each clock cycle of the simulation. Outputs of the submitted solution are dumped into *trace_qN.txt*, while those of the reference model are stored in *golden_qN.txt*. The console output of the grader will point you to the first cycle at which a mismatch between the submitted solution and the reference model occurred. You may then use any file comparison tool to track the issue. For instance, running *vim* with a *-d* switch on the two files will open them side by side, highlighting the mismatches (e.g., you may run *vim -d golden_q1.txt trace_q1.txt*).

Please carefully look at the console output as it may help you to find the problems in your solution.

**Question 1 [30 pts]:** Design an upward counter that counts from $000_{B=5}$ to $444_{B=5}$ in a loop, and displays the count on 7-segment displays included in the template file. Display $Disp2$ should display the most significant digit, and display $Disp0$ the least significant one. Furthermore, bit $j \in [0, 3)$ of the encoding of digit $i \in [0, 3)$ should be displayed on LED $Q_{ij}$, of the template. For example, LED $Q_{22}$ should display the most significant bit of the encoding of the most significant digit. System should be reset by the button *reset*. State upon reset should be $130_{B=5}$ and the counter should start counting upwards. Clock generator from the template should be the only clock generator used.

**Question 2 [30 pts]:** Extend the functionality of the counter, so that it counts upwards while button *But* from the template is pressed and downwards while it is released. Again, state upon reset should be $130_{B=5}$, but now the counting direction should depend on whether the button is pressed or not.

**Question 3 [40 pts]:** Change the functionality of the counter so that briefly pressing the button *But* from the template achieves the following: Once the button is pressed, a change in the direction of counting is scheduled for the clock cycle immediately after the next transition to the value $423_{B=5}$. For example, if the counter is counting upwards at the time of pressing the button, and holds the value $421_{B=5}$, a partial sequence of values observed in the future is: $421_{B=5}, 422_{B=5}, 423_{B=5}, 422_{B=5}, 421_{B=5}, 420_{B=5}$. You can assume that the button will not be pressed when the counter is in the state $423_{B=5}$. Hence, it is up to you to define the behavior in that case. Please note that your system should be able to react to arbitrarily short button presses (disregarding physical limitations, of course). State upon reset should be $130_{B=5}$ and the counter should start counting upwards.