Preparation for Evaluation 1 (Practical part)

A few exercises to get you familiar with INGInious and SystemVerilog

1. (Homework 1, 2023) Implement an 8-bit combinatorial circuit which adds or subtracts a selected constant to an input value.

```
module Exercise_1 (
MyInput,
MyConstantSelect,
MyOperation,
MyOutput
);
input  [7:0] MyInput;
input  [1:0] MyConstantSelect;  // 0: Constant = 1, 1: Constant = 3, 2: Constant = 5, 3: Constant = 7
input  MyOperation;       // 1 for Subtraction, 0 for Addition
output  [7:0]   MyOutput;
```

2. (Question 1, Evaluation 1 2023) Sarting from exercise 1, add some new functionalities : NAND, NOR and XOR operations and a status bit.

```
module Exercise_2 (
MyInput,
MyConstantSelect,
MyOperation,
MyStatus,
MyOutput
);
input [7:0] MyInput;
input [1:0] MyConstantSelect; // 0: Constant = 1, 1: Constant = 3,
  // 2: Constant = 5, 3: Constant = 7
input [2:0] MyOperation;    // 0 for Addition : MyOutput = MyInput + Constant
      // 1 for Subtract : MyOutput = MyInput - Constant
      // 2 for NAND : MyOutput = MyInput NAND Constant
      // 3 for NOR : MyOutput = MyInput NOR Constant
      // 4 for XOR : MyOutput = MyInput XOR Constant
      // 5-7 : not used
output MyStatus;        // if MyOutput is equal to 0 then MyStatus = 1
      // else MyStatus = 0
output [7:0] MyOutput;
```

3. (Homework 2, 2023) Implement an 8-bit counter with the following features : (1) Increment or Decrement, (2) initialized serially from left to right (MSB) or from right to left (LSB), only one command can be enabled on a given time and all commands are active on the positive edge of the clock.

```
module Exercise_3 (
Clock,
CounterOut,
CounterInMSB, CounterInLSB,
DoReset,
DoIncrement, DoDecrement,
DoShiftL2R, DoShiftR2L
);
input  logic Clock;
output logic [7:0] CounterOut;
input  logic  CounterInMSB, CounterInLSB; // Input bit when loading serially
input  logic DoReset; // DoReset = 1     -> CounterOut = 0
input  logic DoIncrement;         // DoIncrement = 1 -> CounterOut++
input  logic  DoDecrement;         // DoDecrement = 1 -> CounterOut--
input  logic DoShiftL2R; // DoShiftL2R = 1  -> CounterOut = (CounterOut >> 1) + CounterInMSB * (2**7)
input  logic DoShiftR2L; // DoShiftR2L = 1  -> CounterOut = (CounterOut << 1) + CounterInLSB
```

4. (Question 3, Evaluation 1 2023) Starting from the FSM provided in the lab 2, add the brake, fog and alarm inputs. When brake is active, the 6 lights (la, lb, lc, ra, rb, rc) are on during 1 cycle. The lights should remain on if brake is active more than one cycle. When fog is active, the 1 lights (la, ra) are on during 1 cycle. The lights should remain on if fog is active more than one cycle. When alarm is active, the 6 lights (la, lb, lc, ra, rb, rc) blink : the 6 lights are on during 1 cycle, then they off during the next cycle. The lights should alternate on and off if alarm is active more than one cycle. For simplicity, we suppose that only one input can be enabled on a given time and that the brake, fog or alarm will not occur during the left / right 3-cycle sequences.

```
module Exercise_4 (
input logic clk,
input logic reset,
input logic left, right, brake, fog, alarm,
output logic la, lb, lc, ra, rb, rc);
```