# Lab session #03

October 10, 2024

Michelangelo Barocci, michelangelo.barocci@polito.it

*Politecnico di Torino,*

*PhD Student @ EDA Group - DAUIN*

*Dpt. of Computer and Control engineering (DAUIN)*

## Before we start

- Everything ok with Lab#02?

# Hamming distance

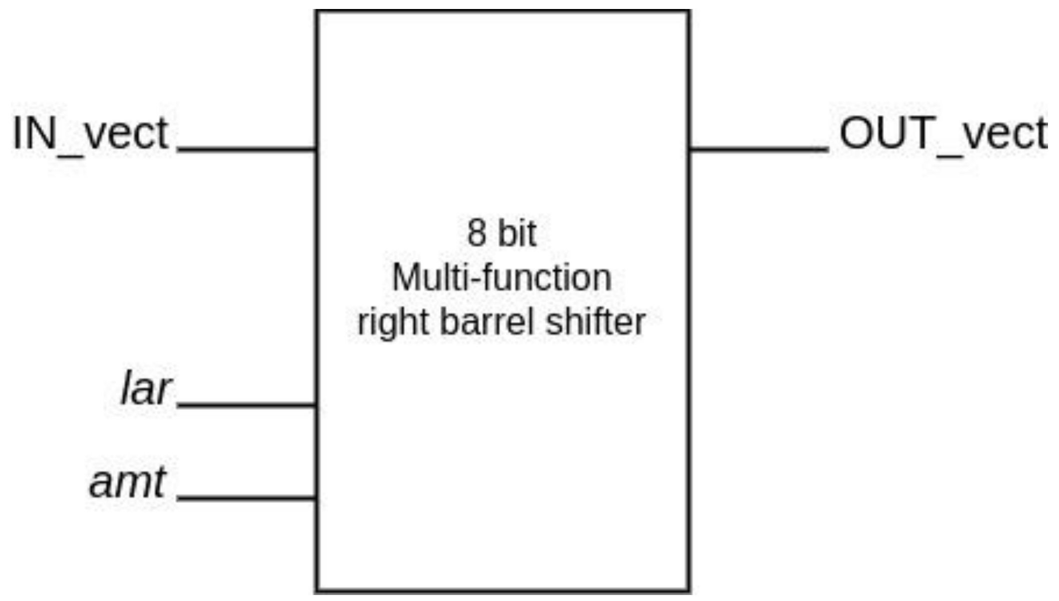
| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| std_logic_vector(7 DOWNTO 0) | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| std_logic_vector(7 DOWNTO 0) | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| std_logic_vector(7 DOWNTO 0) (Hamming distance = 3) | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

- ❑ Design a circuit capable of calculating the Hamming distance between two signals:
  - ❑ The two input signals should be **8 bit std_logic_vector**
  - ❑ Also the output should be described as a **8 bit std_logic_vector**

**To do:**

- ❑ Behavioral/Dataflow description
- ❑ Structural description
- ❑ Testbench

# 8-bit Multi-function Shifter



Without recurring to pre-defined VHDL functions, design a 8-bit Right barrel shifter circuit capable of shifting the input data in three possible ways, using behavioral or dataflow description styles:

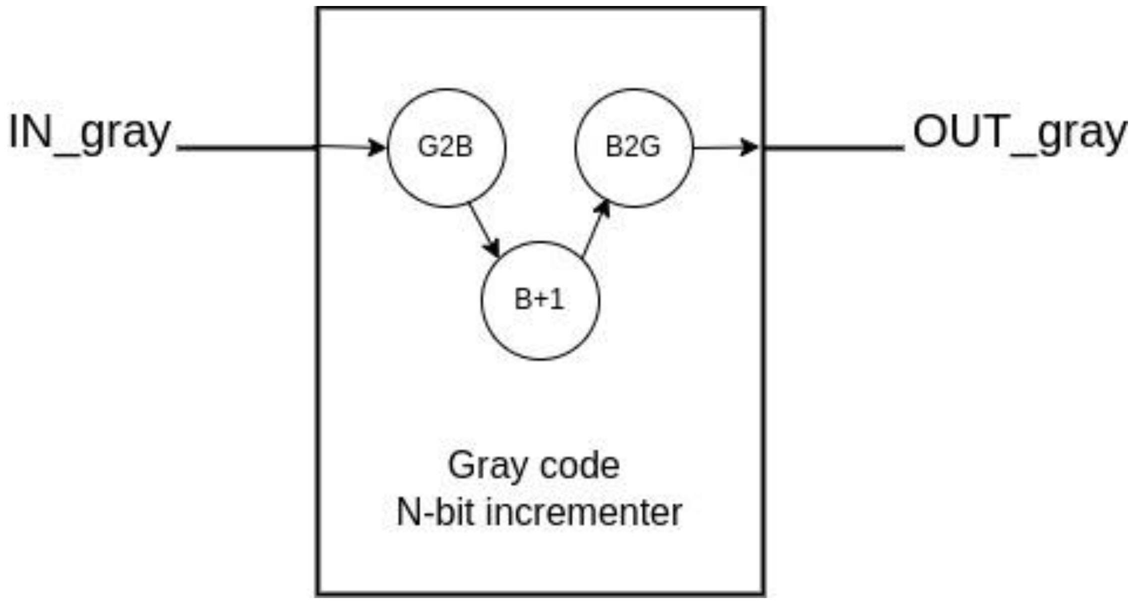
- ❑ rotate-right by *n* positions;
- ❑ logic shift by *n* positions;
- ❑ arithmetic shift by *n* positions;

where the number of positions *n* and the choice of operation to perform are given by the two signals, respectively, *amt* and *lar.*

For all signals, use the *std_logic* and *std_logic_vector* data types.

- ❑ Develop an appropriate testbench to perform the simulation and verify if the design is correct

# Gray code incrementer

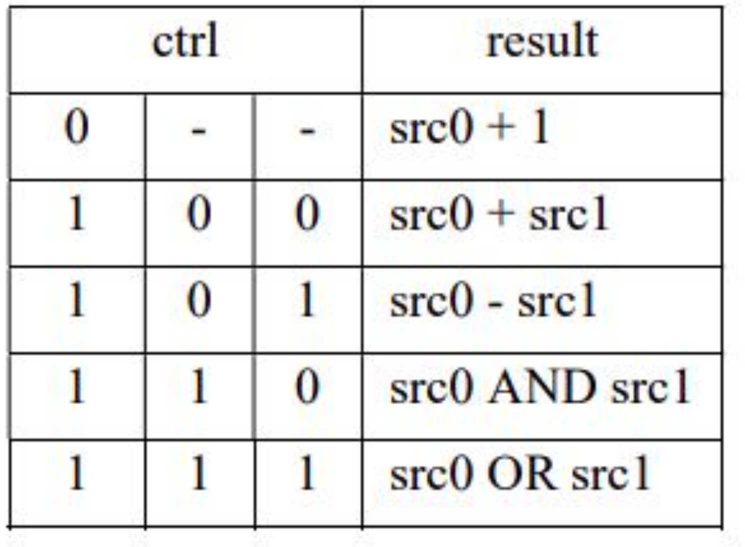


- Design a generic N-bit Gray code incrementer such that given a Gray-coded N-bit input, it increases its value by 1 following the Gray code rules (only one bit change per increment).
- Use dataflow description style

- Suggestion: follow the Lab#03 guidelines to design the circuit in three steps:
  - GRAY to BIN
  - BIN +1
  - BIN to GRAY

- Develop an appropriate testbench to perform the simulation and verify if the design is correct

# ALU - lab#03



| ctrl | | | result |
|---|---|---|---|
| 0 | - | - | src0 + 1 |
| 1 | 0 | 0 | src0 + src1 |
| 1 | 0 | 1 | src0 - src1 |
| 1 | 1 | 0 | src0 AND src1 |
| 1 | 1 | 1 | src0 OR src1 |

- Design a configurable Arithmetic and Logic Unit (ALU) using the dataflow description style
- The ALU should be kept generic in N: *src0* and *src1* are the two input signals, to be considered signed and N-bit wide.
- The operations (5 possibilities) are controlled through an additional input signal called *ctrl* (see table on the left)

- To validate your design, write a testbench covering all possible operations that the ALU should perform with the input datactrl