Lab session #04

October 17, 2024

Michelangelo Barocci, michelangelo.barocci@polito.it

Politecnico di Torino,

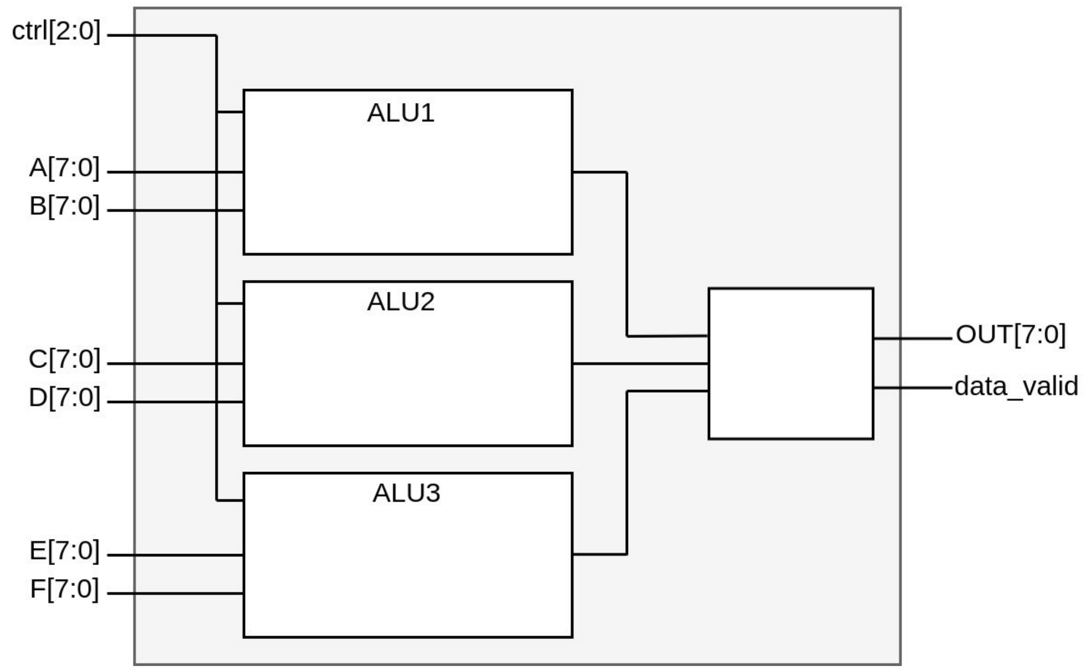
PhD Student @ EDA Group - DAUIN

Dpt. of Computer and Control engineering (DAUIN)

Before we start

- Everything ok with Lab#03?

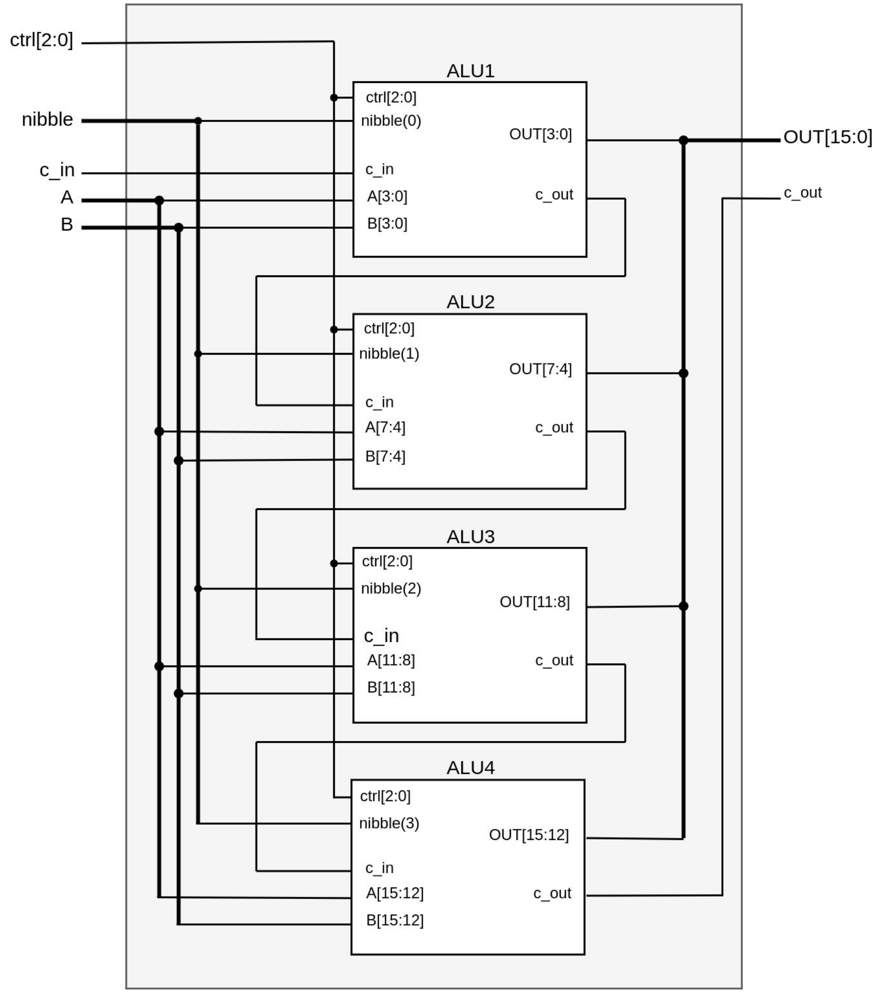
Majority-vote ALU



- Design a circuit by integrating three ALUs as described in LAb#03 (operations controlled by *ctrl* signal, see table below), with three sets of 8-bit signals as inputs.
- Design an additional component that decides the final output of the circuit, according to the rules explained in the Lab guidelines.
- Find a suitable set of relevant inputs to test your design within a testbench
- Single ALU:

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

Reconfigurable ALU



- Design an improved version of the ALU designed in Lab#03:
  - the inputs A and B are 16-bit wide, divided into four 4-bit segments.
  - There are four ALUs enabled by the respective bit of the input signal nibble, each one receives one segment of A and B as input data (look at the diagram on the left).
  - Remember to modify the ALU VHDL description by adding the carry in bit as input and the enable signal
- Write a suitable testbench and simulate your design

Adder-based multiplier

$$
\begin{array}{cccccccccl}
 & & & & & a_3 & a_2 & a_1 & a_0 & \text{multiplicand} \\
\times & & & & & b_3 & b_2 & b_1 & b_0 & \text{multiplier} \\
\hline
 & & & & & a_3b_0 & a_2b_0 & a_1b_0 & a_0b_0 & \\
 & & & & a_3b_1 & a_2b_1 & a_1b_1 & a_0b_1 & & \\
 & & & a_3b_2 & a_2b_2 & a_1b_2 & a_0b_2 & & & \\
+ & & a_3b_3 & a_2b_3 & a_1b_3 & a_0b_3 & & & & \\
\hline
 & y_7 & y_6 & y_5 & y_4 & y_3 & y_2 & y_1 & y_0 & \text{product}
\end{array}
$$

- Design a 8-bit (signed) multiplier using combinational logic (Adders and logic gates) by following the suggested steps:
  - AND operation between bi and A
  - Shift the obtained signals by i bits (left-wise)
  - Sum the partial results
- Check the resulted RTL for the number of synthesized adders
- Check your design through a suitable testbench

Adder-based multiplier - optimized

$$
\begin{array}{cccccccccl}
 & & & & & a_3 & a_2 & a_1 & a_0 & \text{multiplicand} \\
\times & & & & & b_3 & b_2 & b_1 & b_0 & \text{multiplier} \\
\hline
 & & & & & a_3b_0 & a_2b_0 & a_1b_0 & a_0b_0 & \\
\cline{5-9}
 & & & & pp0_4 & pp0_3 & pp0_2 & pp0_1 & pp0_0 & \text{partial product } pp0 \\
 & & & + & a_3b_1 & a_2b_1 & a_1b_1 & a_0b_1 & & \\
\cline{4-8}
 & & & pp1_4 & pp1_3 & pp1_2 & pp1_1 & pp1_0 & & \text{partial product } pp1 \\
 & & + & a_3b_2 & a_2b_2 & a_1b_2 & a_0b_2 & & & \\
\cline{3-7}
 & & pp2_4 & pp2_3 & pp2_2 & pp2_1 & pp2_0 & & & \text{partial product } pp2 \\
 & + & a_3b_3 & a_2b_3 & a_1b_3 & a_0b_3 & & & & \\
\cline{2-6}
 & pp3_4 & pp3_3 & pp3_2 & pp3_1 & pp3_0 & & & & \text{partial product } pp3 \\
\hline
 & pp3_4 & pp3_3 & pp3_2 & pp3_1 & pp3_0 & pp2_0 & pp1_0 & pp0_0 & \text{product } prod
\end{array}
$$

- ❏ Design the same multiplier by adding some modifications: we want to exploit the real number of sums that are really necessary by optimizing the utilization of the intermediate results.
- ❏ Follow the suggestions in the Lab guidelines and check the resulted RTL (Schematic section): has the allocation of resources improved? How?
- ❏ Check your design through a suitable testbench