Prof. Yann Thoma

Exercices du cours VSE
Assertion-based verification
Verification of a arithmetic logic unit (ALU)
semestre automne 2025 - 2026

We are interested in the verification of an arithmetic logic unit able to manage 8 operations, on two input vectors.
A generic parameter SIZE allows to define the operands size.
The input/output are :

Nom	Taille	Description
a_i	SIZE	First operand
b_i	SIZE	Second operand
mode_i	3	Operation mode
c_o	1	Carry out
s_o	SIZE	Result

The expected results are the following :

mode_i	s_o	c_o
"000"	a_i+b_i	carry out
"001"	a_i-b_i	carry out
"010"	a_i or b_i	undefined
"011"	a_i and b_i	undefined
"100"	a_i	undefined
"101"	b_i	undefined
"110"	s_o(0)= '1' when a_i=b_i else '0'	undefined
"111"	0	undefined

The mathematical operations are computed with unsigned numbers.
A second generic parameter ERRNO allows to inject errors in the design. Its behavior is the following :

1. When in the $[0, 15]$ interval the result is valid ;
2. When in the $[16, 24]$ interval, the result is invalid.

This generic parameter allows to test your assertions by trying various ERRNO values
A simple testbench simply runs different scenarios. A file alu_assertions.sv is bind to the DUV, and the assertions you write in will be evaluated during simulation. So, add the necessary assertions to verify the correct behavior of the ALU.
After having implemented the assertions, you can transform your testbench to use random-based coverage-driven simulation. Add simple randomization, and some coverage to decide when to finish the simulation.

The script `sim.do` allows to start the simulation. You can run it as

```
do sim.do all <SIZE> <ERRNO>
```

to test various sizes and error numbers.

There is also a file `check.do` that allows to perform formal verification. We haven't seen that yet, but you can try

```
qverify -do ../scripts/check.do
```

from the `comp` folder to see how it goes.