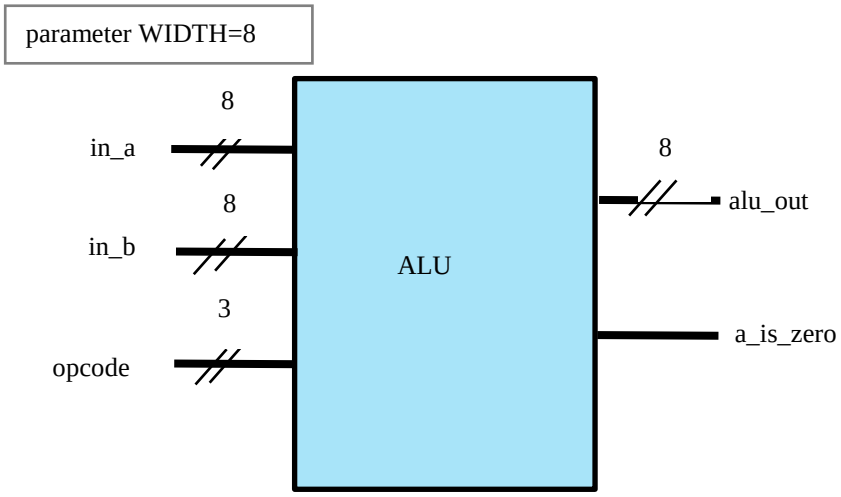
Module 5: Using Verilog Operators

Lab 5-1 Modeling the Arithmetic Logic Unit

Objective: To use Verilog operators to describe a parameterized-width arithmetic logic unit (ALU).

ALU performs arithmetic operations on numbers depending upon the operation encoded in the instruction. This ALU will perform 8 operations on the 8-bit inputs (see table in the S*pecification*) and generate an 8-bit output and single-bit output.

In this lab, you create an ALU design as per the specification and verify it using the provided testbench. Please make sure to use the same variable name as the ones shown in block diagrams.



Specifications

- `in_a, in_b,` and `alu_out` are all 8-bit long. The `opcode` is a 3-bit value for the CPU operation code, as defined in the following table.
- `a_is_zero` is a single bit asynchronous output with a value of 1 when `in_a` equals 0. Otherwise, `a_is_zero` is 0.
- The output `alu_out` value will depend on the `opcode` value as per the following table.
- To select which of the 8 operations to perform, you will use `opcode` as the selection lines.
- The following table states the opcode/instruction, opcode encoding, operation, and output.

Opcode/ Instruction	Opcode Encoding	Operation	Output
HLT	000	PASS A	in_a => alu_out
SKZ	001	PASS A	in_a => alu_out
ADD	010	ADD	in_a + in_b => alu_out
AND	011	AND	in_a & in_b => alu_out
XOR	100	XOR	in_a ^ in_b => alu_out
LDA	101	PASS B	in_b => alu_out
STO	110	PASS A	in_a => alu_out
JMP	111	PASS A	in_a => alu_out

Designing an ALU

1. Change to the *lab6-alu* directory and examine the following files.

alu_test.v	ALU test
filelist.txt	File listing all modules to simulate

2. Use your favorite editor to create the *alu.v* file. Describe the ALU module named as *alu*.

   a. Parameterize the ALU input and output width so that the instantiating module can specify the width of each instance.

   b. Assign a default value to the parameter.

Verifying the ALU Design

1. Using the provided testbench module, check your ALU design using the following command with VCS

   ```
   vcs alu.v alu_test.v (Batch Mode)
   ```

   or

   ```
   vcsalu.v alu_test.v -gui -access +rwc ( GUI Mode)
   ```

2. You might find it easier to list all the files and simulation options in a text file and pass the file into the simulator using the `-f xrun` option.

```
xrun -f filelist.txt -access rwc
```

You should see the following results.

```
At time 1 opcode=000 in_a=01000010 in_b=10000110 a_is_zero=0
alu_out=01000010
At time 2 opcode=001 in_a=01000010 in_b=10000110 a_is_zero=0
alu_out=01000010
At time 3 opcode=010 in_a=01000010 in_b=10000110 a_is_zero=0
alu_out=11001000
At time 4 opcode=011 in_a=01000010 in_b=10000110 a_is_zero=0
alu_out=00000010
At time 5 opcode=100 in_a=01000010 in_b=10000110 a_is_zero=0
alu_out=11000100
At time 6 opcode=101 in_a=01000010 in_b=10000110 a_is_zero=0
alu_out=10000110
At time 7 opcode=110 in_a=01000010 in_b=10000110 a_is_zero=0
alu_out=01000010
At time 8 opcode=111 in_a=01000010 in_b=10000110 a_is_zero=0
alu_out=01000010
At time 9 opcode=111 in_a=00000000 in_b=10000110 a_is_zero=1
alu_out=00000000
TEST PASSED
```

3. Correct your ALU design as needed.