Lab 1: 8 Function 4-Bit ALU Written Using Structural VHDL and Tested With NC-VHDL

Jordan Kieltyka
University of Maryland, Baltimore County

CMPE 640
Custom VLSI Design
Fall 2023

Simulation Procedure

The complete simulation procedure is rather straightforward thanks to the project's included makefile. The makefile does require a little modification in order to be utilized. The below variables in the makefile should be set to the correct paths and will vary depending on the user/machine.

```
NCVHDL = <path to run_ncvhdl.bash>
NCELAB = <path to run_ncelab.bash>
NCSIM = <path to run_ncsim.bash>
CDSLIB = <path to cds.lib>
HDLVAR = <path to hdl.var>
```

Once the makefile variables are set, the input text file should also be configured. The name of the input file is '`alu_4_in.txt`' and should follow the pattern listed in the *Input* column in *Table 1*. It should be noted that the simulation by default will be expecting 8 input patterns, for a total of 40 lines in the input file.

Table 1: Format for input and output text file for simulating the 8 function 4-bit ALU.

Input	Output
`A` (4 bits) `B` (4 bits) `Cin` (1 bit) `S0` (1 bit) `S1` (1 bit)	`G` (4 bits) `Cout` (1 bit)

Once the makefile and input file are configured, the below commands can be run to compile, elaborate and simulate the ALU.

1. `make alu_4_test`
2. `make alu_4_test_elaborate`
3. `make alu_4_test_simulate`

Additionally, simply using the command '`make`' will compile, elaborate, and then simulate all in one command.

The output of the simulation can be found in '`alu_4_out.txt`'. Unfortunately, due to limitations of NCSIM, the output file will only contain 7 outputs (14 lines) instead of the expected 8 (16 lines). The output file follows the pattern in the *Output* column of *Table 1*.

Test Results

A variety of test inputs were used to confirm the functionality of the 8 function 4-bit ALU via simulation. A sample of these test inputs and corresponding outputs demonstrating all 8 functions of the ALU can be seen in *Table 2* below.

Table 2: Simulated input and resulting output for 8 function 4-bit ALU.

A (in)	B (in)	Cin (in)	S0 (in)	S1 (in)	G (out)	Cout (out)
0000	1111	0	0	0	0000	0
0000	1111	1	0	0	0001	0
1010	0101	0	1	0	1111	0
1010	0101	1	1	0	0000	1
1010	1010	0	0	1	1111	0
1010	1010	1	0	1	0000	1
0001	0000	0	1	1	0000	1
0001	0000	1	1	1	0001	1

Design Hierarchy

Figure 1 displays how entities utilize other entities in a hierarchical way from basic logic gates up to the 8 function 4-bit ALU. *Table 3* explains the operation of each entity.

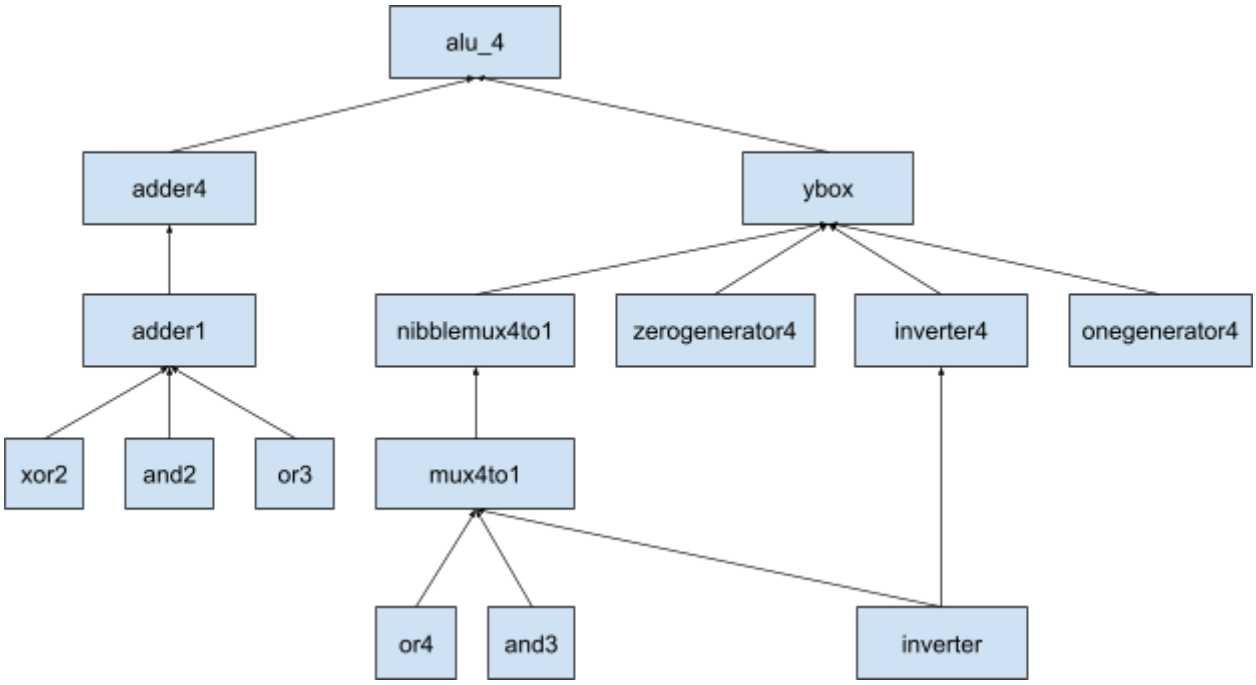

Figure 1: Hierarchy of entities used to create the 8 function 4-bit ALU. The top of the diagram has the most complex entity, while the bottom entities are the least complex.

Table 3: All entities used to construct the 8 operation 4-bit ALU with a description of each of their operations.

Name	Operation
alu_4	Performs 8 arithmetic operations on 4 bit inputs `A`, `B` and 1 bit input `Cin` and outputs the result as a 4 bit value `G` and 1 bit carry-out `Cout`. 1 bit `S0` and `S1` inputs determine which arithmetic operation is performed.
adder4	Adds two 4 bit values with a carry-in bit and outputs the sum as 4 bits with one carry-out bit.
adder1	Adds two data input bits and carry-in bit and outputs the sum along with a carry-out bit.
xor2	Takes in 2 input bits and XORs them.
and2	Takes in 2 input bits and ANDs them.
or3	Takes in 3 input bits and ORs them.
ybox	Determines what value to add with `A` and `Cin` based on `B`, `S0` and `S1` inputs.
nibblemux4to1	A 4 to 1 multiplexer with data inputs as four 4 bit buses and an output as a 4 bit bus determined by two 1 bit inputs.
mux4to1	4 bit to 1 bit multiplexer.
or4	Takes in 4 input bits and ORs them.
and3	Takes in 3 input bits and ANDs them.
zerogenerator4	Generates a 4 bit bus of zeroes.
onegenerator4	Generates a 4 bit bus of ones.
inverter4	Inverts four bit bus.
inverter	Inverts one bit.