EE365 – Advanced Digital Circuit Design
Fall 2003
Design Project 1 – 4-Function ALU

Project Overview:

Your task is to implement a four function arithmetic logic unit (ALU) of single bit width in VHDL. You will be required to write VHDL code for each part of the design, and to test it via simulation. The testing will include both functional and timing simulations based on specific device families.

Project Background and Design:

An ALU is a piece of hardware that takes two inputs and uses a control input to choose between a variety of operations to be performed on the data. Our simplified version of the ALU will implement four functions on two inputs of single bit width: logical AND, logical OR, logical XOR, and inversion. These functions will be selected using a multiplexer with a 2-bit control word. Here is a drawing of the logic design:

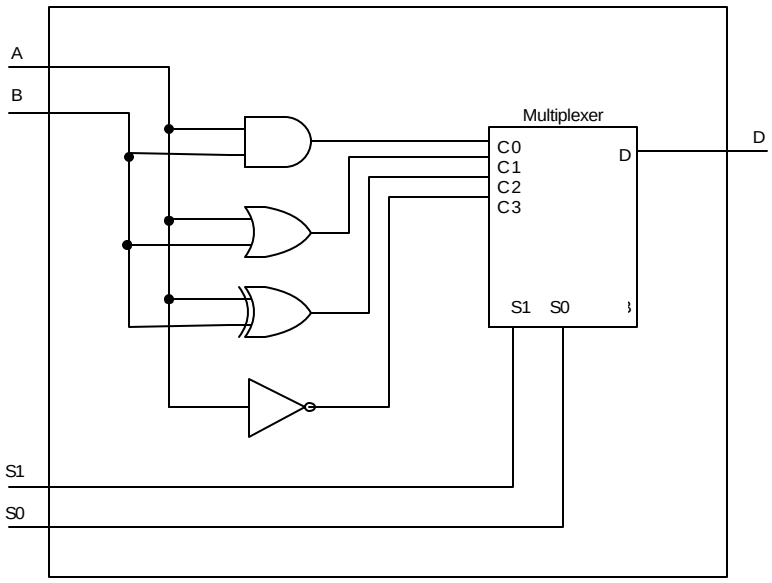

Figure 1: Design drawing for Project 1.

The control interface to the ALU will operate as follows:

S1	S0	Output
0	0	D = A · B (AND)
0	1	D = A + B (OR)
1	0	D = A ⊕ B (XOR)
1	1	D = A’ (NOT)

Project specifications:

You are to implement the design in VHDL under the following constraints:

1) The top level design must use structural VHDL techniques (component instantiation etc.)
2) The AND gate module's architecture must be implemented using one or more concurrent *conditional* signal assignment statements (Wakerly p. 287).
3) The OR gate module's architecture must be implemented using one or more concurrent *selected* signal assignment statements (Wakerly p. 287-288).
4) The XOR gate module's architecture must be implemented using a behavioral description (i.e. a process statement). (Wakerly p. 289).
5) The inverter and 4 to 1 multiplexer may be implemented in the style of your choosing. If you need help with how to implement a multiplexer, Wakerly presents an example of a 4 to 1 multiplexer in VHDL on p.410; however, note the inputs in the example are 8 bits wide instead of the single bit inputs you will need for this project.

You will also need to implement a test bench for the purposes of simulating and testing your design. http://www.vhdl-online.de/~vhdl/TB-GEN/ will help you generate a generic test bench from your top level entity statement.

For this project, you will need to provide both functional and timing verification of your design. Functional testing checks only the logic itself; it does not factor in parameters like propagation and transport delay, but it is useful for determining whether your implementation is logically correct. Timing simulation uses actual device parameters in order to determine whether an actual circuit would behave as expected from the functional simulation. Timing information can be incorporated into VHDL by placing the statement "after __ ns" directly after the value being assigned to a signal. Example:

OUTPUT <= '1' when INPUT = '0';

becomes

OUTPUT <= '1' after 5 ns when INPUT = '0';

See examples in the Wakerly book for more information.

The device timing information to use for each module is as follows:

Device	Use timing for:
AND gate	74F08
OR gate	74ALS32
XOR gate	74LS86
Inverter	74LS04
4:1 Mux	74ALS153

Timing information for each device can be found at http://www.ti.com. For each device, just use the worst case (highest maximum time) for all signal transitions; don't worry about differentiating between low-to-high and high-to-low transitions at this point. For the 4:1 Mux, use the worst case from the "A or B to Y" timing characteristics.

Remember, this timing information is only for the timing simulation. Do not use timing information for the functional simulation!

When you turn your project in, your report should include the following:

1) A brief description of the project and your method of implementation.
2) A description of your test plan.
3) Timing diagrams for both the functional and timing simulations, along with an explanation (in words) of what the timing diagrams show in terms of design verification.
4) A copy of the VHDL code for each module and the testbench (you only need to turn in the code you used for your timing simulation; it should be exactly the same as it was for your functional simulation except for the added timing values).
5) Answers to the following questions:
    a. You should have found that you need to have the following statement in your XOR gate architecture:
    ```
    process (A, B)
    ```

    What is the significance of listing the input signal names in the parentheses that follow the word "process"? What would happen if one or both of those signals were not listed? (Hint: Look up the name for the parentheses statement; it should point you in the right direction).

    b. Run the following test pattern in your timing simulation:
    At t = 0 ns: S1 = 0, S0 = 0, A = 0, B = 1
    At t = 100 ns: S1 = 1, S0 = 0, A = 0, B = 0

    With what functions do the control codes correspond? What is the expected output for each test pattern? Does the waveform match this expectation? If not, why do you believe there is a discrepancy?