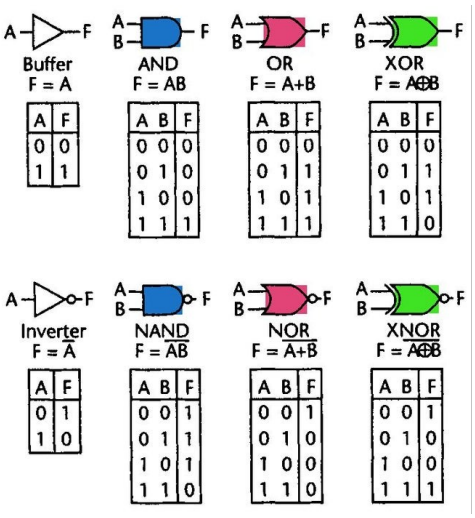
Computer Systems 414

Practical 6

2021

Aim of Practical 6:

1. To learn by example and gain practical experience in writing, compiling and simulating VHDL code. This week we will look at sequential systems.



Rules of Engagement

1. Attendance is not required for this practical – students can complete the practical on their own time.

2. Students are allowed to work in groups of two to three, but each student must write and submit their own code.

3. Supporting documentation is available on learn.sun.ac.za, but students also have to do their own research.

4. The report is to be submitted on learn.sun.ac.za. The deadline for the reports is (14:00) Wednesday, 26 May 2021. **No late submissions will be accepted.**

5. Do not forget to cite and give credit for any information reported which is not your property.

8. Google is your friend. Any information not given is left out on purpose. Search for your solution and on the internet or relevant manuals and documentation.

Task (Use the uploaded files from SunLearn as required)

Assignment 5A

For this practical we will use GHDL, an Opensource VHDL compiler.

Please refer to the previous practical for the GHDL installation instructions.

Task (Use the uploaded files from SunLearn as required)

1. Download the attached VHDL code for an up counter and the associated testbench.

2. Compile the counter by typing *ghdl -a counter.vhdl*.

3. Now compile the testbench by typing *ghdl -a counter_sim.vhdl*

4. Run the testbench by typing *ghdl -r counter_sim --vcd=wave.vcd.* After a few seconds stop the simulation by pressing <ctrl-C>.

5. View the results by typing *gtkwave wave.vcd.* Adjust the zoom controls until you can see the counter sequence.

6. Why does the counter only count to 10 decimal? Change the code to let the counter count to 'F' hex and simulate it again.

6. Open the counter.vhdl and counter_sim.vhdl files in a text editor. Observe the changes in the testbench code as compared to last week – *counter_sim* is declared as the testbench entity. The counter itself is instantiated by the *counter_0: entity work.counter c*ommand and its pins are mapped to the signals declared just above it. Two processes are then declared, one to generate the clock (with a 1ns cycle time by the *wait* command) and one that lowers the reset line for 5 clock cycles.

7. Now change the counter to a down counter and simulate the results.

8. Then add an extra signal and use that change the counter into a up or down counter depending on the state of the signal and simulate the results.

9. Submit your code, simulation results and the names of your group members to Sunlearn before the cut-off time.