# Homework 4

Please turn in all verilog code and a hardcopy of your simulation results. You can demonstrate proper operation by either a printout of the "monitor" output (the table printed once the simulation has been run) or by a printout of the waveforms (the results of some problems are easier to see one way and the results of other problems are easier to see another). **Be sure to annotate your simulation results telling me how your results prove that you have met all specifications.**

## Problem 1:

Design a 4-bit arithmetic block that performs the following functions. You may only use muxes, NOR gates, and 4 full adders. Assume all inputs are 4-bit binary numbers in 2's complement form. (Hint: This problem should sound very familiar! Be sure to think about the hardware before implementing this design to ensure an efficient design.)

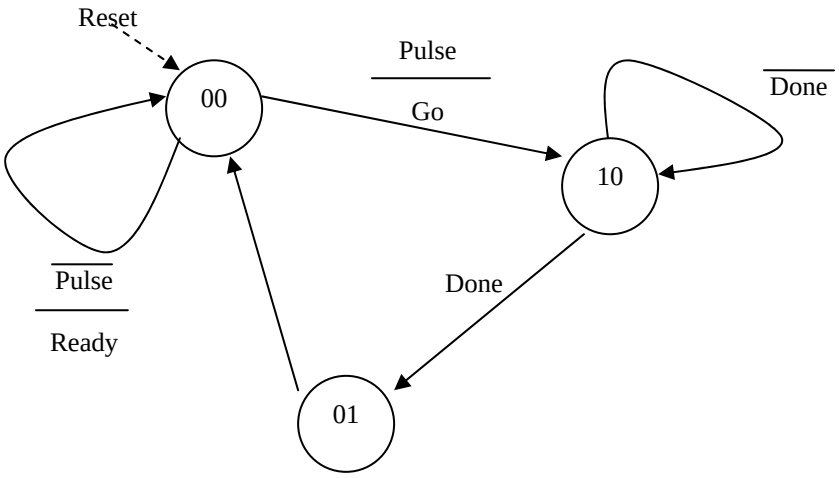
| S1 | S0 | Function |
|---|---|---|
| 0 | 0 | A+B |
| 0 | 1 | A-B |
| 1 | 0 | A+1 |
| 1 | 1 | A-1 |

You must implement and test each module as distinct and separate entities. Tie the pieces together in a top-level module and design a test for it as well. Each functional unit should have its own module and file -- try to reuse modules as often as possible.

## Problem 2:

You have been given a finite state machine that is to accomplish the functionality in the following figure. The compiled module that implements this machine is available online. Download and uncompress the compressed folder problem2.rar. You should set your working design directory within NC-SIM to the problem2 folder extracted from the archive. (Note: you will not be able to ready the actual module code. However, the image of the FSM will appear in your worklib.) The interface to the machine is **Problem2FSM( clk, Reset, Pulse, Done, Go, Ready )**.

Write a test bench for this machine and determine whether or not it is working to specifications. Be sure to annotate your work.




## Problem 3

Write a verilog description of the FSM in Problem #2. Run your test from Problem #2 on it to verify that it works.

## Problem 4

Write a verilog description of the FSM in Problem #2, but implement it as a Moore Machine. Run your test from Problem #2 on it to verify that it works.