TCES 330, Spring 2008
Midterm Number 1

General Comments:
Follow the directions very carefully. You will be partly graded on how well you can follow these directions. This exam requires you to create three Quartus II projects; these are described below. When you are finished put all three projects in a zip file and upload it to Moodle.

Turn in this paper copy of the exam, too. Be sure your name is on this page.

Do NOT connect to the Internet until you are given permission to upload to Moodle. Do NOT use your cell phone in any fashion during this exam except for emergency purposes; keep your cell phone out of view. Do not consult any other individual except for an instructor. This is an **open book** exam. You may use your text books, your notes, my slides, the lab assignments, Altera tutorials, etc.

When you set up your projects, consider the following:

- Pick the Cyclone II EP2C35F672C6 device, even though we won't be using the DE2 boards.
- Assign unused pins properly.
- Enter a value for 'Capacitive Laoding.'
- Minimize the number of warnings you get when you compile.
- Use comments appropriately. As a minimum, your name should appear in **every** Verilog module.

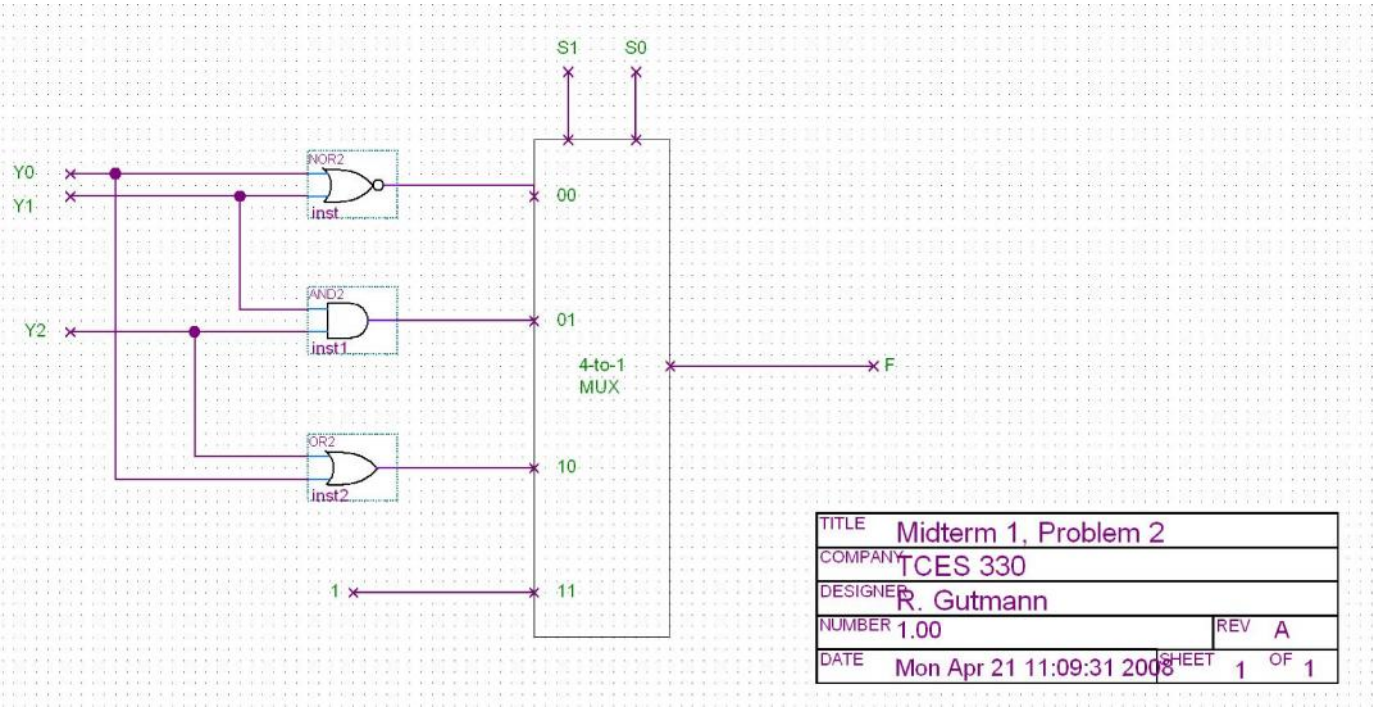
There are no 'trick' questions. This exam is a straight-forward test of your abilities to write Verilog HDL programs and to verify them.

Problem 1. This tests your ability to implement a truth table two different ways and your ability to verify your results. The following truth table is for a binary to Gray code converter Recall that the Gray code is what you used when you constructed Karnaugh maps: Only one bit changes from line to line. In this truth table the inputs are b_2, b_1, b_0 and the outputs are g_2, g_1, g_0.

b_2	b_1	b_0	g_2	g_1	g_0
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	1
0	1	1	0	1	0
1	0	0	1	1	0
1	0	1	1	1	1
1	1	0	1	0	1
1	1	1	1	0	0

a) Using only “assign” statements to implement your logic, write a Verilog module that implements the truth table. Your inputs and outputs should be vectors. Call this project “MT1_1a” and call the module “MT1_1a.v”.
b) Come up with a testbench that tests all input possibilities for your module of part a). Use this testbench to verify that your design in part a) works correctly by performing a functional simulation using ModelSim.
c) Create a new Quartus project called “MT1_1c” that implements the truth table, this time using an “always @” procedure. Call this module “MT1_1c.v”. Your inputs and outputs should be very similar to part a).
d) Repeat the test from part b) on this new circuit.

Problem 2. This problem tests your ability to use sub-circuits and built-in gates in a top-level Verilog module. We need to implement the following circuit in Verilog.



Here is the Verilog module for the Mux module:

```
// 4-to-1 multiplexer for TCES 330 Midterm #1
// Problem 2
// … your comments

module MT1MUX( I, S, W );
  input [3:0] I;  // mux input lines
  input [1:0] S;  // mux select lines
  output reg W;

  always @ * begin
    W = I[S];
  end

endmodule
```

Here's how you should structure your module:

```
// Your comments

module MT1_2( Y, S, F );
  input [2:0]Y;
  input [1:0]S;
  output F;

… your code goes here

endmodule
```

Instantiate the MT1MUX module in your module and use Verilog's primitive gates (**and**, **or,** and so on**)** to complete the circuit. Verify your design by comparing the circuit displayed by the RTL Netlist Viewer with the circuit above. Comment below regarding any differences.