Homework 2

Please complete Problem 1, 2, 3, and 4 and turn these problems in at the beginning of class on Monday, Sept 22.
The solutions to all questions will be posted after class on Monday so that you will have the solutions to study for the exam. I suggest you try to work Problem 5 and 6 BEFORE looking at the solutions.

For problems 2, 3, 5, and 6, please perform each of the following steps:

(a) Draw a symbol of the device showing all inputs and outputs.
(b) Develop a synthesizable verilog description.
(c) Develop a testbench to test the operation of the device
(d) Verify that your circuit meets specifications.

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:

Provide a schematic of the circuit that the following Verilog description should produce. Once you have produced the schematic, answer the following questions.

a) Could any lines of code be removed without changing the functionality? Explain your answer
b) Under what conditions of A and B is the output 0000_2?

```
/*
 * Module comb:
 * performs combinational logic...
 * what DOES it do?
 */
module comb( A, B, O );

   // inputs
   input  [3:0] A, B;

   // outputs
   output [3:0] O;

   // identifiers
   wire cl, cg, ce;
   reg [1:0] e;
   reg [3:0] O;


   /* Functionality */
   assign cl = (A < B);
   assign ce = (A == B);
```

```
   assign cg = (A > B);

   always @(cg or ce or cl)
   case( {1'b0,cg,ce,cl} )
      4'b0001: e <= 2'b00;
      4'b0010: e <= 2'b01;
      4'b0100: e <= 2'b10;
      4'b1000: e <= 2'b11;
      default: e <= 2'b00;
   endcase

   always @(e or A or B)
   if( e == 2'b00 )
      O <= B;
   else if( e == 2'b01 )
      O <= ~A;
   else if( e == 2'b10 )
      O <= A;
   else
      O <= 4'b0000;

endmodule
```

Problem 2:

Design a 5-to-32 binary decoder. Outputs are active high. Thus, only one output is high, determined by the binary input, and all others are low. The decoder also has an active low enable. If the enable is low, all outputs are low. If the enable is high, the decoder operates as described.

Problem 3:

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

.

Problem 4:

Provide a schematic of the circuit that the following Verilog description should produce. Once you have produced the schematic, answer the following questions.

(a) Is the reset for the flip-flops asynchronous or synchronous? Explain your answer.

(b) What is the purpose of the line shown in bold?

```
module accum(In, clock, Reset, Out);

parameter Bit=4;

//Port modes
input [Bit-1:0] In;
input clock, Reset;
output [Bit-1:0] Out;

//Registered identifiers
reg [Bit-1:0] Out;
reg [Bit-1:0] Q;

//Functionality

always @ (posedge clock or posedge Reset)
    if (Reset == 1)
        Q<=4'b0;
    else
        Q<=Out;

always @ (In or Q)
    Out <= In + Q;

endmodule
```

Problem 5:

Design a 4 bit counter that has a select line that allows you to count up or down. When the counter counts up, it should increment by 3. When the counter counts down, it should decrement by 5. In addition, the counter has an active high enable. When the enable is high, the counter should hold its current count. When the counter is low, it should count in the manner described previously. The counter also has an active low reset. When the reset is low, the output of the counter should reset to a binary zero.

Problem 6:

Design a 4-bit shift register that performs the functions shown in the following table. "Shift right" denotes that the bits in the most significant flip-flop will move into to flip-flop of lesser significance. A zero will be moved into the most significant flip-flop. . "Shift left" denotes that the bits in the least significant flip-flop will move into to flip-flop of greater significance. A zero will be moved into the least significant flip-flop "Hold" denotes that the flip-flops will not change their value. "Parallel load" denotes that a new binary number will be loaded in parallel.

Before writing your verilog descriptions, draw a schematic of the circuit. Be sure to think about the hardware before implementing this design to ensure an efficient design.

S1	S0	Function
0	0	Shift right
0	1	Shift left
1	0	Hold
1	1	Parallel load