LSU EE 3755 Homework 1 Due: 11 September 2013

Problem 1: Draw a schematic of the logic circuit described by the Verilog code below.

```
module pie( x, y, a, b, c);
   input a, b, c;             output x, y;
   wire   t1, t2;

   xor x1(t1,a,b);
   not n1(x,t1);
   and a1(t2,x,c);
   or  o1(y,t2,b);

endmodule
```

Problem 2: Draw a schematic of the logic circuit described by the Verilog module `twoterms` below. *Note: this problem is very similar to one given last year. Try to solve this one before looking at the solution to last year's problem.*

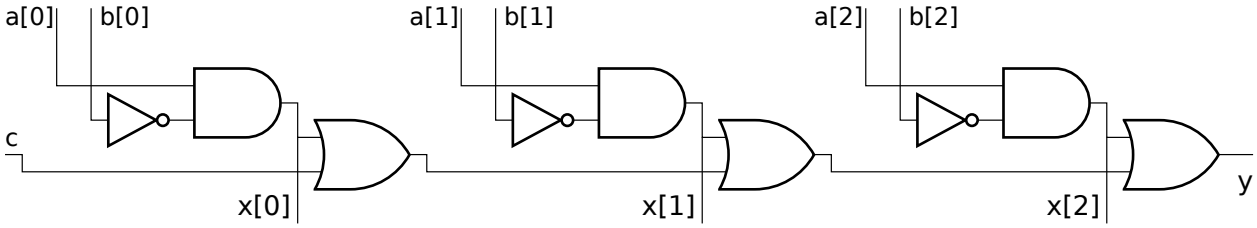
- Show the contents of each instantiated module. (That is, do **not** just show a box labeled `term1100` or `twoterms`.)
- Show using AND, OR, and NOT gates, inferring the correct gate for the Verilog operators used in the `assign` expression.
- To the extent possible, label the diagram using the port names defined by `twoterms` (`x`, `i`, `j`, `k`, and `m`).

```
module term1100(x,a,b,c,d);
   input a, b, c, d;              output x;
   assign x = a && b && !c && !d;
endmodule

module twoterms_bundle(x,a);
   input [3:0] a;             output      x;
   wire        tx101, t100x;
   term1100 t0(tx101, a[2], a[0], a[1], 1'b0);
   term1100 t1(t100x, a[3], a[3], a[1], a[2]);
   or o1(x, tx101, t100x);
endmodule

module twoterms(x,i,j,k,m);
   input i,j,k,m;               output x;
   wire [3:0] bundle;
   assign bundle[3:0] = {i,j,k,m};
   twoterms_bundle t(x,bundle);
endmodule
```

Problem 3: Notice that the logic below consists of three repeated parts. Write a Verilog explicit structural description of the logic which consists of two modules, one module, name it `part`, will be for the part that's repeated, the other, name it `whole`, will instantiate `part` three times and interconnect them appropriately. Choose appropriate inputs and outputs for the two modules based on the diagram.



Problem 4: Replace each assign statement below with explicit structural code. Consider each assign statement in isolation (they are not part of the same module). There is no need to show the module declarations.

```
assign x = a & b ? 1 : 0;

assign x = a == b ? 0 : 1;

assign x = a ? b : c;
```