LSU EE 3755 Homework 2 Due: 3 October 2012

Code for the following assignment can be found at `http://www.ece.lsu.edu/ee3755/2012f/hw02.v.html`. *Some parts are reproduced here.*

Problem 1: Module `adder_r4_c3` is a two-level adder similar to `cla_12_two_level` covered in Lecture Set 5, `http://www.ece.lsu.edu/ee3755/2012f/l05.v.html`, except that it consists of 3 interconnected 4-bit ripple adders rather than 3 interconnected 4-bit CLA adders.

```
module adder_r4_c3(sum,a,b);
   input [11:0]  a, b;
   output [12:0] sum;
   wire [2:0]  P, G, carry;
   wire [2:0]  CO; // Unused.

   // Each 4-bit ripple adder computes a piece of the sum.
   ripple_4_block ad0(sum[3:0],  CO[0], a[3:0],  b[3:0],  carry[0]);
   ripple_4_block ad1(sum[7:4],  CO[1], a[7:4],  b[7:4],  carry[1]);
   ripple_4_block ad2(sum[11:8], CO[2], a[11:8], b[11:8], carry[2]);

   // Use carry lookahead logic to compute carry in signals for ripple adders.
   gen_prop_4 gp0(G[0], P[0], a[3:0],  b[3:0]);
   gen_prop_4 gp1(G[1], P[1], a[7:4],  b[7:4]);
   gen_prop_4 gp2(G[2], P[2], a[11:8], b[11:8]);

   assign      carry[0] = 1'b0;
   assign      carry[1] = G[0];
   assign      carry[2] = G[0] & P[1] | G[1];
   assign      sum[12] = G[0] & P[1] & P[2] | G[1] & P[2] | G[2];
endmodule
```

(*a*) Draw a diagram of `adder_r4_c3` in which the instantiated modules (`gp0`, `gp1`, etc.) are shown as boxes (don't show their contents). Be sure to show the connections to and from the instantiated modules and also show logic within module `adder_r4_c3`, including the logic to generate the carry signals.

Problem 2: In the `gen_prop_4` module below there are two lines to compute `P`, one labeled Method E, the other labeled Method G (Method E is commented out). Both are correct and a decent synthesis program should generate the same logic for both.

```
module gen_prop_4(G,P,a,b);
   output G;
   output P;
   input [3:0] a, b;

   wire [3:0]  g = a & b;
   wire [3:0]  p = a | b;

   assign G = g[0] & p[1] & p[2] & p[3]
            | g[1] & p[2] & p[3]
            | g[2] & p[3]
            | g[3];

   // assign P = p == 4'hf;   // Method E
   assign P = p[0] & p[1] & p[2] & p[3];  // Method G
```

```
endmodule
```

(*a*) Recall that a synthesis program will, in the inference step, replace some Verilog operators with basic gates, others will be replaced by the equivalent of library functions. For the two Methods of computing `P` show the results of the inference step. That is, **write a structural description** that either includes **basic gates** or **the instantiation of a library module**. Make up a reasonable name for the library module. *Hint: This part is easy once the concept of inference is understood.*

(*b*) Show a possible implementation for this inferred library module. Show a description of the library module, remember that the library module should work for any inputs, not just the inputs used in the prior problem. Show your answer as either a logic diagram or as a Verilog module. *Hint: Don't use AND gates naïvely.*

Problem 3: Notice that the module `adder_r4_c3` uses instances of `gen_prop_4` to compute the generate (`G`) and propagate (`P`) signals while in module `adder_c4_c3` those same signals are generated by the `cla_4_block` modules. (Use link above to get full listing of code.)

(*a*) Modify `adder_c4_c3` so that it too uses `gen_prop_4` to create the generate and propagate signals, and modify other modules as necessary (such as `cla_4_block` to remove the generate and propagate outputs).

(*b*) How does this change affect cost with and without optimization?