Homework Assignment #6 - SOLUTIONS

Assigned Thursday, March 3, 2005
Due Thursday, March 10, 2005

(Note – Solutions are posted on the course web site after the due date)

1. **(30 pts) You are to design a circuit which inputs a two's complement binary number between –7 and +7. Its output is the absolute value (abs(-5) = abs(5) = 5, etc.). Derive reduced expressions for each output signal. Be sure to take advantage of any "don't care" combinations to simplify the circuit.**

Solution:

	x3 x2 x1 x0	*a3 a2 a1 a0*
0	0 0 0 0	0 0 0 0
1	0 0 0 1	0 0 0 1
2	0 0 1 0	0 0 1 0
3	0 0 1 1	0 0 1 1
4	0 1 0 0	0 1 0 0
5	0 1 0 1	0 1 0 1
6	0 1 1 0	0 1 1 0
7	0 1 1 1	0 1 1 1
-8	1 0 0 0	d d d d
-7	1 0 0 1	0 1 1 1
-6	1 0 1 0	0 1 1 0
-5	1 0 1 1	0 1 0 1
-4	1 1 0 0	0 1 0 0
-3	1 1 0 1	0 0 1 1
-2	1 1 1 0	0 0 1 0
-1	1 1 1 1	0 0 0 1

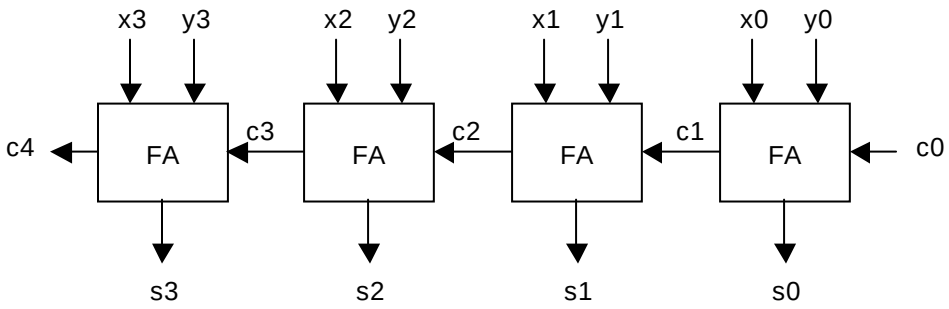
Doing the Kmaps, we get

a3 = 0
a2 = x3' x2 + x3 x2' + x2 x1' x0'
a1 = x3' x1 + x1 x0' + x3 x1' x0
a0 = x0

2. **(20 pts) A circuit adds two 4-bit numbers (x3,x2,x1,x0) and (y3,y2,y1,y0) to produce a 4-bit sum (s3,s2,s1,s0) and a carry out bit c4. Design another circuit to detect a two's complement overflow and output a logic HIGH if an overflow occurs. Use only the signals listed above.**

Overflow occurs if you add two positive numbers and get a negative number, or two negative numbers and get a positive number:
OVERFLOW = x3'*y3'*s3 + x3*y3*s3'

3. (20 pts) The following diagram shows a ripple adder that adds two 4-bit numbers, x = (x3 x2 x1 x0) and y = (y3 y2 y1 y0), and a carry-in bit c0. Consider the inputs x = 1010, y = 1110, c0 = 1.



(a) Give the binary values for the sum s = (s3 s2 s1 s0) and carry-out bits c = (c4 c3 c2 c1).

Solution: s = 1001, c = 1110

(b) Give the decimal values of x, y, and s if the numbers are 4-bit unsigned numbers. Is there overflow?

Solution: x = 10, y = 14, s = 9. There is overflow.

(c) Give the decimal values of x, y, and s if the numbers are 4-bit two's complement numbers. Is there overflow?

Solution: x = -6, y = -2, s = -7. No overflow.

4. (30 pts) Consider a 4-bit carry-lookahead adder.

(a) Write the equations to compute the carry out bits (c1,c2,c3,c4), following section 5.4 in the text.

(b) Draw the complete circuit for this adder (you can use Quartus).

(c) Determine the number of gates needed to implement the 4 bit carry-lookahead adder, assuming no fan-in constraints. Note that your total should include the four 3-input XOR gates needed to generate the sum bits. (Hint: the total is greater than 25 but less than 30.)

Solution:
(a) Following section 5.4, we have:

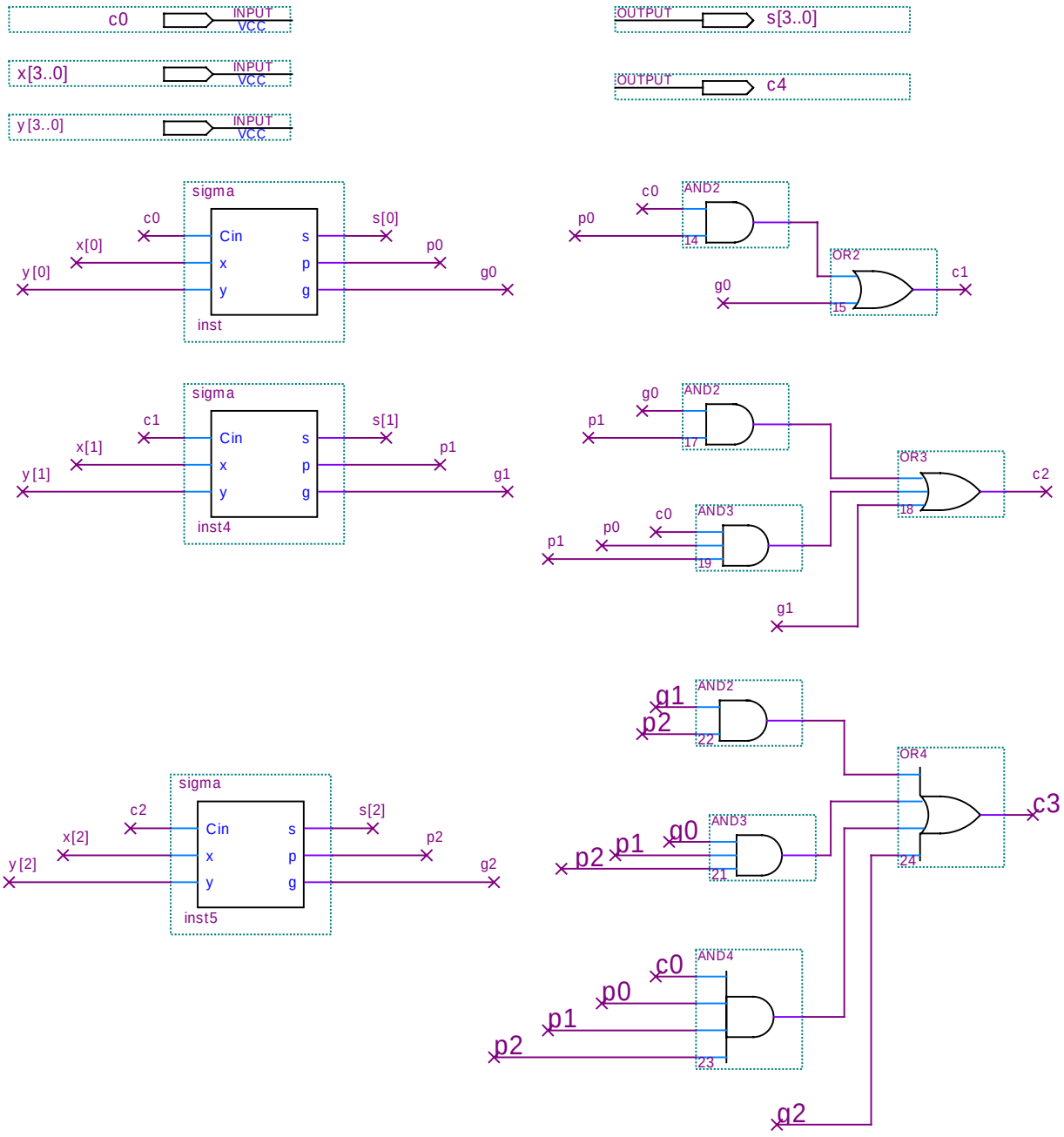
c1 = g0 + p0 c0, where gi = xi yi, pi = xi + yi

c2 = g1 + p1 c1
= g1 + p1 (g0 + p0 c0)
= g1 + p1 g0 + p1 p0 c0

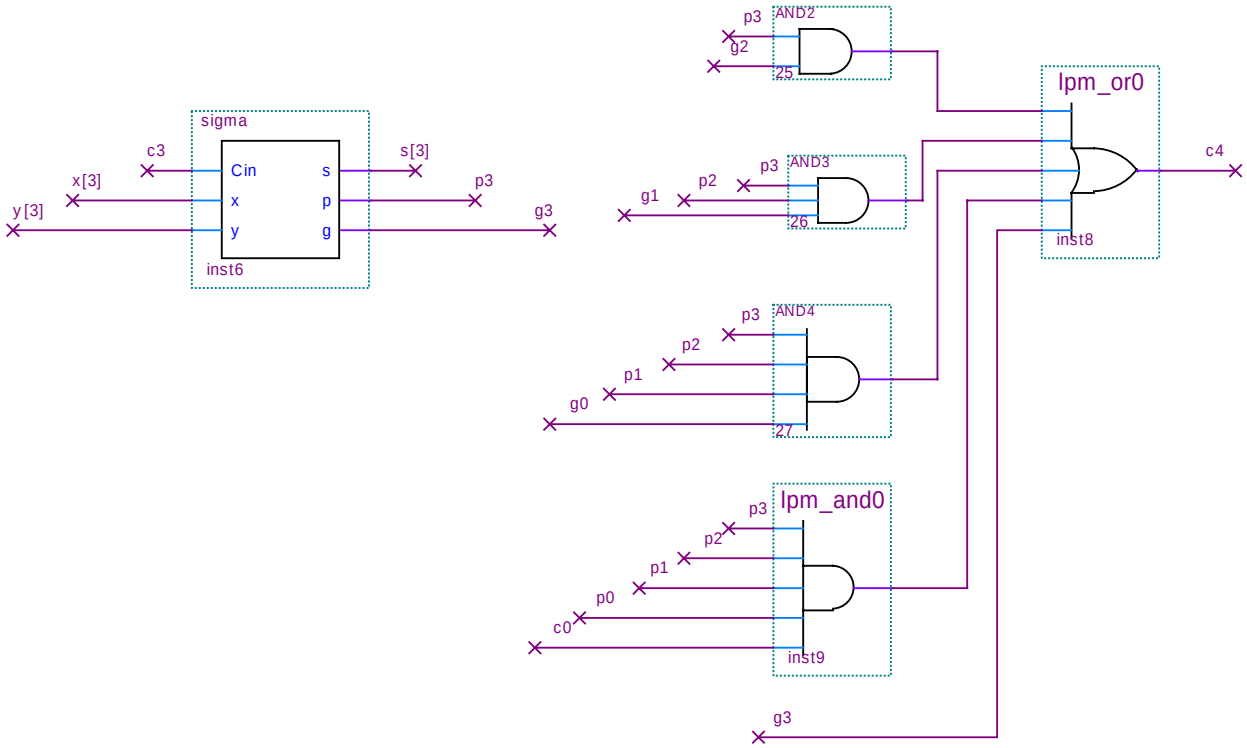
c3 = g2 + p2 c2
= g2 + p2 (g1 + p1 g0 + p1 p0 c0)
= g2 + p2 g1 + p2 p1 g0 + p2 p1 p0 c0

c4 = g3 + p3 c3
= g3 + p3 (g2 + p2 g1 + p2 p1 g0 + p2 p1 p0 c0)
= g3 + p3 g2 + p3 p2 g1 + p3 p2 p1 g0 + p3 p2 p1 p0 c0

(b) The first three stages are (we did this in computer lab 2):



The fourth stage for c4 = g3 + p3 g2 + p3 p2 g1 + p3 p2 p1 g0 + p3 p2 p1 p0 c0:



(c) We require 4 XOR gates to compute the sum bits. To compute the carry bits, look at the equations above.

To determine the number of gates, we note that each gi takes one AND gate and each pi takes one OR gate. At each stage, some of the g's and p's have already been computed from a previous stage, so we don't need to re-compute them (ie., we just run a wire from the previous stage). For each stage, there is only one new g and one new p - all the others are available from previous stages.

Expression	*#AND gates*	*# OR gates*	*Total for this stage*
c1 = g0 + p0 c0	2	2	4
c2 = g1 + p1 g0 + p1 p0 c0	3	2	5
c3 = g2 + p2 g1 + p2 p1 g0 + p2 p1 p0 c0	4	2	6
c4 = g3 + p3 g2 + p3 p2 g1 + p3 p2 p1 g0 + p3 p2 p1 p0 c0	5	2	7

Thus, it takes 4+5+6+7=22 gates to compute c1..c4.

Including the 4 gates needed to compute s0..s3, the total number of gates is 22+4 = 26.