## EE 231 – Homework 5
## Due October 1, 2010

1. For the circuit shown in Fig. 4.26 (page 155 of the text),

   (a) Write the Boolean function for the outputs in terms of the input variables.

   $Y_0 = (A_0S'E')|(B_0SE')$
   $Y_1 = (A_1S'E')|(B_1SE')$
   $Y_2 = (A_2S'E')|(B_2SE')$
   $Y_3 = (A_3S'E')|(B_3SE')$

   (b) If the circuit is listed in a truth table, how many rows (and columns) would there be in the table?

   There are 10 inputs and 4 outputs, so there will be $2^{10} = 1024$ rows, and 14 columns (one for each of the 10 inputs and 4 outputs). (Homework as posted did not ask for columns.)

   (c) Write a Verilog dataflow model for the circuit.

   Here are two ways:

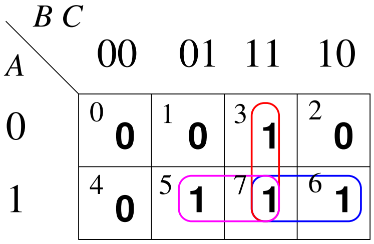
```
module hw5_p1(output [3:0] Y, input [3:0] A, B, input S, E);

assign Y[0] = (A[0] & ~S & ~E) | (B[0] & ~S & E);
assign Y[1] = (A[1] & ~S & ~E) | (B[1] & ~S & E);
assign Y[2] = (A[2] & ~S & ~E) | (B[2] & ~S & E);
assign Y[3] = (A[3] & ~S & ~E) | (B[3] & ~S & E);

endmodule

----------------

module hw5_p1(output [3:0] Y, input [3:0] A, B, input S, E);

assign Y = (E == 1’b1) ? 4’h0 :   // Output all 0’s if E is high
           (S == 1’b0) ? A    :   // Output is A if E is 0
                         B;       // Output is B is E is 1
                         I[3];
endmodule
```

2. A majority circuit is a circuit with an odd number of inputs whose output is a 1 if and only if a maority of its inputs are 1.

   (a) Find the truth table for a three-input maority circuit.

| $x$ | $y$ | $z$ | $F$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

   (b) From the truth table, find the Boolean equation for the circuit.



| $A$ \ $B\,C$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0<br>**0** | 1<br>**0** | 3<br>**1** | 2<br>**0** |
| 1 | 4<br>**0** | 5<br>**1** | 7<br>**1** | 6<br>**1** |

$F = AB + AC + BC$

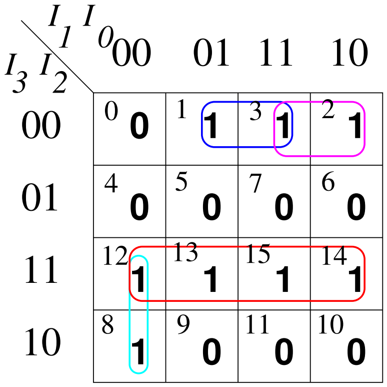
   (c) Write a Verilog dataflow model of the circuit.

```
module hw5_p2(output F, input x, y, z);

assign F = (x & y) | (x & z) | (y & z);

endmodule
```

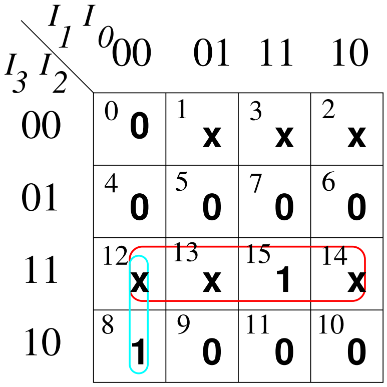
3. Problem 4.8. Treat this as a 4-input, 4-output combinational circuit, find the truth table, and use Karnaugh maps to simplify.

   Take the 8, 4, -2, -1 column as the inputs and the 8421 column as the outputs. The ”Unused bit combinations” can be treated as a ”don’t care”, or can result in the last six rows of the table. I have done it both ways. Reaarange the rows so the inputs are in numerical order (0000, 0001, ..., 1111):

| Inputs | | | | With Complete Decoding | | | | With Don't Cares | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $I_3$ | $I_2$ | $I_1$ | $I_0$ | $O_3$ | $O_2$ | $O_1$ | $O_0$ | $O_3$ | $O_2$ | $O_1$ | $O_0$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | x | x | x | x |
| 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | x | x | x | x |
| 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | x | x | x | x |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | x | x | x | x |
| 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | x | x | x | x |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | x | x | x | x |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |

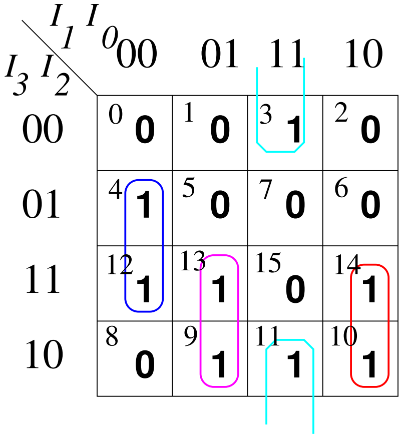


| $I_3I_2$ \ $I_1I_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 **0** | 1 **1** | 3 **1** | 2 **1** |
| 01 | 4 **0** | 5 **0** | 7 **0** | 6 **0** |
| 11 | 12 **1** | 13 **1** | 15 **1** | 14 **1** |
| 10 | 8 **1** | 9 **0** | 11 **0** | 10 **0** |

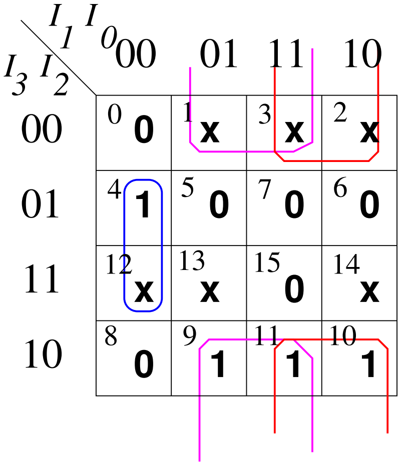


| $I_3I_2$ \ $I_1I_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 **0** | 1 **x** | 3 **x** | 2 **x** |
| 01 | 4 **0** | 5 **0** | 7 **0** | 6 **0** |
| 11 | 12 **x** | 13 **x** | 15 **1** | 14 **x** |
| 10 | 8 **1** | 9 **0** | 11 **0** | 10 **0** |

$O_3 = I_3I_2 + I_3I_1'I_0' + I_3'I_2'I_0 + I_3'I_2'I_1$ or $O_3 = I_3I_2 + I_3I_1'I_0'$

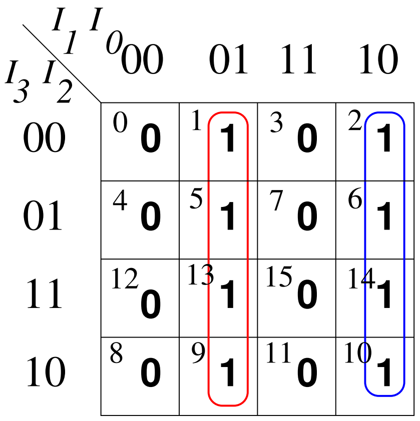


| $I_3I_2$ \ $I_1I_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 **0** | 1 **0** | 3 **1** | 2 **0** |
| 01 | 4 **1** | 5 **0** | 7 **0** | 6 **0** |
| 11 | 12 **1** | 13 **1** | 15 **0** | 14 **1** |
| 10 | 8 **0** | 9 **1** | 11 **1** | 10 **1** |



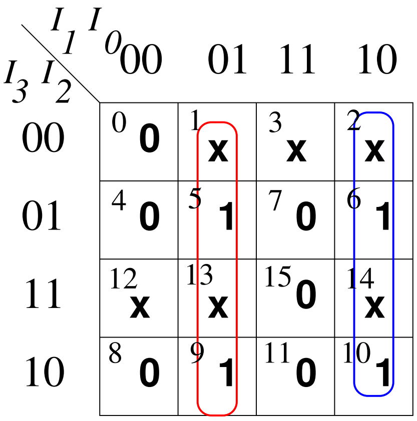
| $I_3I_2$ \ $I_1I_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 **0** | 1 **x** | 3 **x** | 2 **x** |
| 01 | 4 **1** | 5 **0** | 7 **0** | 6 **0** |
| 11 | 12 **x** | 13 **x** | 15 **0** | 14 **x** |
| 10 | 8 **0** | 9 **1** | 11 **1** | 10 **1** |

$O_2 = I_2I_1'I_0' + I_3I_1'I_0 + I_2'I1I0 + I_3I_1I_0'$ or $O_2 = I_2'I1 + I_2'I_0 + I_2I_1'I_0'$

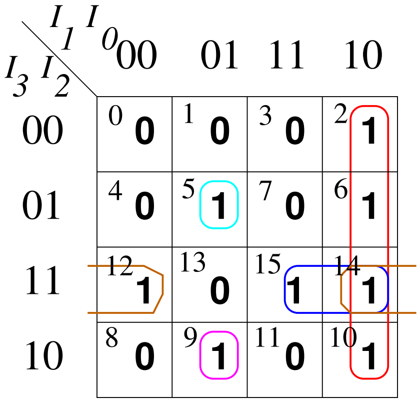


$O_1 = I_1'I_0 + I_1I_0'$

or

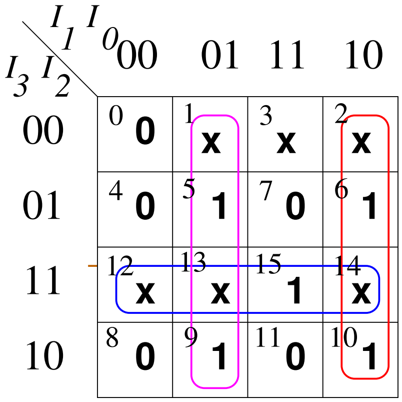


$O_1 = I_1'I_0 + I_1I_0'$



$O_0 = I_1I_0' + I_3I_2I_1 + I_3I_2I_0' + I_3'I_2I_1'I_0 + I_3I_2'I_1'I_0$

or



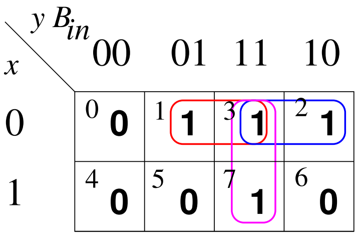
$O_0 = I_1'I_0 + I_1I_0' + I_3I_2$

4. (a) Design a full subtractor circuit with three inputs $x$, $y$, $B_{in}$, and two outputs $D$ and $B_{out}$. The circuit subtracts $x - y - B_{in}$. $B_{out}$ is 0 if no borrow is needed to complete the subtraction, and 1 if a borrow is needed. Find the truth table, and simplify the equations for $D$ and $B_{out}$ using Karnaugh maps.

$x - y - B_{in}$

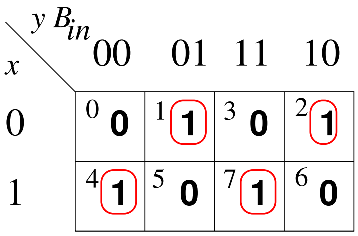
For example, $x = 0$, $y = 1$ and $B_{in} = 1$: 0 - 1 - 1, or 0 - 2. To do this, you need to borrow from the next column, so $B_{out}$ will be a 1. When you borrow from the next column, the borrow in brings you a 2, so the equation for $D$ is 2 - 2 = 0, so $0 - 1 - 1$ gives $B_{out} = 1$, $D = 0$.

| $x$ | $y$ | $B_{in}$ | $B_{out}$ | $Diff$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |



| $x \backslash yB_{in}$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | $^0$ 0 | $^1$ 1 | $^3$ 1 | $^2$ 1 |
| 1 | $^4$ 0 | $^5$ 0 | $^7$ 1 | $^6$ 0 |

$B_{out}$

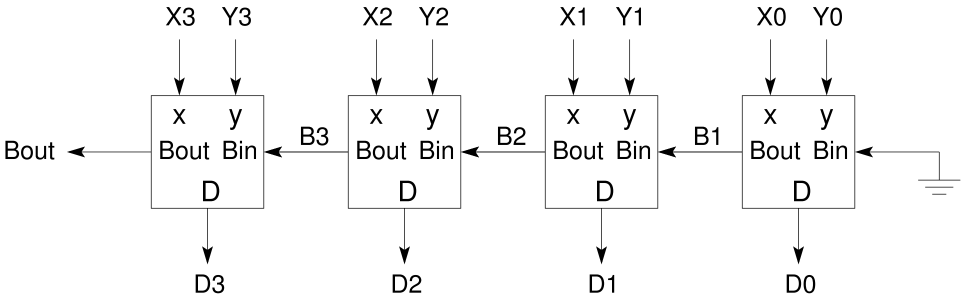


| $x \backslash yB_{in}$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | $^0$ 0 | $^1$ 1 | $^3$ 0 | $^2$ 1 |
| 1 | $^4$ 1 | $^5$ 0 | $^7$ 1 | $^6$ 0 |

$D$

$B_{out} = x'y + x'B_{in} + yB_{in}$ $D = x \oplus y \oplus B_{in}$

(b) Draw a block diagram showing how four full subtractors can be used to implement a 4-bit subtraction.



(c) Write a Verilog dataflow model to implement the circuit of Part (b).
Easy way:

```
module sub_4_bit(input [3:0] X, Y, output [3:0] D, output B);

// Calculate 4-bit difference D = X - Y, and output borrow B_out

assign {B,D} = X - Y;

endmodule
```

Harder way:

```
module full_sub(output D, B_out, input x, y, B_in);

// Calculate 1-bit difference D = x - y - Bin, and output borrow B_out

assign D = (x ^ y) ^ B_in;
assign B_out = (!x & y) | (!x & B_in) | (y & B_in);

endmodule

module sub_4_bit(input [3:0] X, Y, output [3:0] D, output B_out);

// Calculate 4-bit difference D = X - Y, and output borrow B_out


wire B1, B2, B3

full_sub FS0 (D[0], B1,    X[0], Y[0], 1’b0);
full_sub FS1 (D[1], B2,    X[1], Y[1], B1);
full_sub FS2 (D[2], B3,    X[2], Y[2], B2);
full_sub FS3 (D[3], B_out, X[3], Y[3], B3);

endmodule
```

5. (a) The adder-subtractor circuit of Fig. 4.13 has the following values for mode input M and data inputs A and B:

| | M | A | B |
|---|---|---|---|
| (a) | 0 | 0011 | 0101 |
| (b) | 0 | 1101 | 1101 |
| (c) | 1 | 0100 | 0011 |
| (d) | 1 | 0000 | 0001 |

In each case determine the values of the four SUM outputs, the carry C, and overflow V.

The circuit of Figure 4.13 gives the following outputs. Note that the C bit is incorrect for subtraction. (For subtraction, the C bit should be 1 if a borrow is needed and 0 if it is not; the circuit of Figure 4.13 gives the inverse of this.)

| | M | A | B | SUM | C | V |
|---|---|---|---|---|---|---|
| (a) | 0 | 0011 | 0101 | 1000 | 0 | 1 |
| (b) | 0 | 1101 | 1101 | 1010 | 1 | 0 |
| (c) | 1 | 0100 | 0011 | 0001 | 1 | 0 |
| (d) | 1 | 0000 | 0001 | 1111 | 0 | 0 |

I will accept the following, where the C bit in now the correct borrow out on subtraction. You can get this output by modifying Figure 4.13 so that $C_{out}$ is the XOR of $C_4$ and $M$.

| | M | A | B | SUM | C | V |
|---|---|---|---|---|---|---|
| (a) | 0 | 0011 | 0101 | 1001 | 0 | 1 |
| (b) | 0 | 1101 | 1101 | 1010 | 1 | 0 |
| (c) | 1 | 0100 | 0011 | 0001 | 0 | 0 |
| (d) | 1 | 0000 | 0001 | 1111 | 1 | 0 |

(b) Using the conditional operator (?:), write a Verilog dataflow description of the four-bit adder-subtractor of Fig. 4.13.

Easy way:

```
module add_sub_4_bit (output [3:0] S, output C, V, input [3:0] A, B, input M);

// To get the carry (borrow) and sum (difference), just add (subtract) based
// on the value of M

assign {C,S} = (M == 1’b0) ? A + B : A - B;

// To get the overflow:
// On add (M==0), overflow when P + P = N or N + N = P
// On subtract (M==0), overflow when P - N = N or N - P = P

assign V = ((M==0) && ((!A[3] & !B[3] & S[3]) | (A[3] &  B[3] & !S[3]))) ? 1’b1 :
           ((M==1) && ((!A[3] &  B[3] & S[3]) | (A[3] & !B[3] & !S[3]))) ? 1’b1 :
   1’b0;

endmodule
```

Harder way:

```
module full_add(output S, C_out, input x, y, C_in);

// Calculate 1-bit sum S = x + y + C_in, and output carry C_out

assign S = (x ^ y) ^ C_in;
assign C_out = (x & y) | (x & C_in) | (y & C_in);

endmodule

module add_sub_4_bit (output [3:0] S, output C, V, input [3:0] A, B, input M);

// Calculate 4-bit sum S = X + Y + C_in, and output carry C_out

wire C1, C2, C3, C4;

full_add FA0 (S[0], C1, A[0], B[0]^M, M);
full_add FA1 (S[1], C2, A[1], B[1]^M, C1);
full_add FA2 (S[2], C3, A[2], B[2]^M, C2);
full_add FA3 (S[3], C4, A[3], B[3]^M, C3);

assign V = C4 ^ C3;
assign C = C4 ^ M;

endmodule
```

6. For the circuit shown in Fig. 4.13 of the text, verify that the V output bit is correct for the addition operation. That is, show that (a) V will be 1 when you add two positive numbers together ($B_3 = 0$ and $A_3 = 0$) and get a negative number ($S_3 = 1$), (b) V will be 1 when you add two negative numbers together ($B_3 = 1$ and $A_3 = 1$) and you get a positive number ($S_3 = 0$), and (c) the V output will be 0 in all other circumstances (adding two positives and getting a positive, adding two negatives and getting a negative, or adding a positive and a negative number).

There are six possibilities: P + P = P, P + P = N, P + N = P, P + N = N, N + N = P, or N + N = N. Note that for addition, M is 0, so the inputs to the last full adder are $A_3$ and $B_3$.

**P + P = P:** Should get $V = 0$.

$A_3 = 0$, $B_3 = 0$, and $S_3 = 0$. With $A_3 = 0$ and $B_3 = 0$ $S_3$ will be 0 only if $C_3$ is 0. Since $A_3 = 0$, $B_3 = 0$, and $C_3 = 0$, the carry $C_4$ will be 0. $C_3 = 0$ and $C_4 = 0$, so $V$ will be 0.

**P + P = N:** Should get $V = 1$.

$A_3 = 0$, $B_3 = 0$, and $S_3 = 1$. With $A_3 = 0$ and $B_3 = 0$ $S_3$ will be 1 only if $C_3$ is 1. Since $A_3 = 0$, $B_3 = 0$, and $C_3 = 1$, the carry $C_4$ will be 0. $C_3 = 1$ and $C_4 = 0$, so $V$ will be 1.

**P + N = P:** Should get $V = 0$.

$A_3 = 0$, $B_3 = 1$, and $S_3 = 0$. With $A_3 = 0$ and $B_3 = 1$ $S_3$ will be 0 only if $C_3$ is 1. Since $A_3 = 0$, $B_3 = 1$, and $C_3 = 1$, the carry $C_4$ will be 1. $C_3 = 1$ and $C_4 = 1$, so $V$ will be 0.

**P + N = N:** Should get $V = 0$.

$A_3 = 0$, $B_3 = 1$, and $S_3 = 1$. With $A_3 = 0$ and $B_3 = 1$ $S_3$ will be 1 only if $C_3$ is 0. Since $A_3 = 0$, $B_3 = 1$, and $C_3 = 0$, the carry $C_4$ will be 0. $C_3 = 0$ and $C_4 = 0$, so $V$ will be 0.

**N + N = P:** Should get $V = 1$.

$A_3 = 1$, $B_3 = 1$, and $S_3 = 0$. With $A_3 = 1$ and $B_3 = 1$ $S_3$ will be 0 only if $C_3$ is 0. Since $A_3 = 1$, $B_3 = 1$, and $C_3 = 0$, the carry $C_4$ will be 1. $C_3 = 0$ and $C_4 = 1$, so $V$ will be 1.

**N + N = N:** Should get $V = 0$.

$A_3 = 1$, $B_3 = 1$, and $S_3 = 1$. With $A_3 = 1$ and $B_3 = 1$ $S_3$ will be 1 only if $C_3$ is 1. Since $A_3 = 1$, $B_3 = 1$, and $C_3 = 1$, the carry $C_4$ will be 1. $C_3 = 1$ and $C_4 = 1$, so $V$ will be 0.

7. Assume that inverter gates have a propagation delay of 5 ns and that AND, OR, NAND and NOR gates have a propagation delay of 10 ns. What is the total propagation delay of the four-bit magnitude comparator circuit of Fig. 4.17?

   Each output goes through five gates (NOT, AND, NOR, AND, OR), so the propagation delay for every output is 5 ns + 10 ns + 10 ns + 10 ns + 10 ns = 45 ns.