ECEn 220 –
Chapter 9 Homework Solutions

9.1. Convert the following to unsigned binary, add the numbers together in binary, and check your result by converting it back to base-10: 4 + 5 =?. Use four bits for your operands and result.

```
   0 1 0 0          4
+  0 1 0 1        + 5
 -----------      -----
   1 0 0 1          9
```

9.2. Repeat the above for 8 + 5 =?.

```
   1 0 0 0        8
+  0 1 0 1      + 5
-----------     -----
   1 1 0 1      1 3
```

9.3. Repeat the above for 12 + 15 =? but use five bits for the result.

```
   1 1 0 0        12
+  1 1 1 1      + 15
-----------     ------
 1 1 0 1 1        27
```

9.5. Repeat the three additions from the previous 3 problems but for 2's complement. Use four bits for the operands if possible, use five bits if needed. In each case first do the addition and compute a result with the same number of bits as the operands. Then, determine which operations overflowed and repeat those after first sign-extending the operands by one bit position.

(8.1)

```
  0 1 0 0             4
+ 0 1 0 1           + 5
---------           ---
  1 0 0 1            -7        Overflow!

  0 0 1 0 0           4
+ 0 0 1 0 1         + 5
-----------         ---
  0 1 0 0 1           9
```

```
(8.2)            0 1 0 0 0        8     Sign extend to represent +8
               + 0 0 1 0 1      + 5
               -----------      ---
                 0 1 1 0 1       13

(8.3)            0 1 1 0 0        12    Sign extend to represent +12, +15
               + 0 1 1 1 1      + 15
               -----------      ----
                 1 1 0 1 1        -5    Overflow!

                 0 0 1 1 0 0      12
               + 0 0 1 1 1 1    + 15
               -------------    ----
                 0 1 1 0 1 1      27
```

9.6. Do the following addition using 2’s complement and 4-bit numbers: -8+ 7 =?. Try using the most significant carry-out as the MSB for a 5-bit result. Show why it doesn’t work. Repeat by first sign-extending the operands to be 5-bit values and then compute a 5-bit result.

```
                 1 0 0 0         -8
               + 0 1 1 1        + 7     Carry out of MSB = ‘0’
               ---------        ---
               0 1 1 1 1         15     Incorrect!

                 1 1 0 0 0       -8     Sign extend operands to 5-bits
               + 0 0 1 1 1      + 7
               -----------      ---
                 1 1 1 1 1       -1
```

9.9. Using Figure 8.6 and Figure 8.7 as a guide, draw the full gate-level schematic for a 3-bit adder. Can you write how many gates are required, in general, for an *n*-bit ripple carry adder?

If every stage uses a full-adder, then each stage requires 5 logic gates. An n-bit adder requires 5•n logic gates.

However, since we know the carry in is 0, we can simplify the first full adder down to two gates. This results in what is called a *half adder* – it only has A and B inputs and Sum and Carry outputs. The logic for the half adder is as follows:

$$S = A \oplus B \qquad C_{out} = A \bullet B$$

Thus the complete ripple carry adder requires 5•(n-1)+2 logic gates

9.10 The chapter discussed detecting overflow in 2’s complement addition by looking at the sign of both operands and the sign of the result. Derive the logic equation required to detect overflow this way and draw a gate-level schematic of the resulting overflow detection circuit.

Suppose we label the operand bits and result bits as follows:

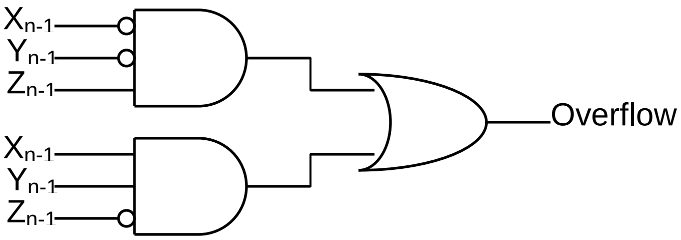
$$\begin{array}{ll} X_{n-1}\ X_{n-2}\ \ldots\ X_0 & \text{Operands} \\ Y_{n-1}\ Y_{n-2}\ \ldots\ Y_0 & \\ \hline Z_{n-1}\ Z_{n-2}\ \ldots\ Z_0 & \text{Result} \end{array}$$

The following truth table gives the value of the overflow signal based on the most significant bit of the operands and result. The K-map is shown on the right.

X_{n-1}	Y_{n-1}	Z_{n-1}	Overflow
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

$Y_{n-1}Z_{n-1}$ \ X_{n-1}	0	1
00		
01	1	
11		
10		1

Overflow = $X_{n-1}'Y_{n-1}'Z_{n-1} + X_{n-1}Y_{n-1}Z_{n-1}'$



X_{n-1}
Y_{n-1}
Z_{n-1}
X_{n-1}
Y_{n-1}
Z_{n-1}
Overflow