The University of New South Wales

ELEC2141: Digital Circuit Design

Tutorial Week 9 – Arithmetic Circuits

1.

	number	1s Complement	2s complement
a.	1001 1100	0110 0011	0110 0100
b.	1001 1101	0110 0010	0110 0011
c.	1010 1000	0101 0111	0101 1000
d.	0000 0000	1111 1111	0000 0000
e.	1000 0000	0111 1111	1000 0000

2.

$0xF2 = 1111\ 0010_2$

a. $\overset{128}{1}\overset{64}{1}\overset{32}{1}\overset{16}{1}\ \overset{8}{0}\overset{4}{0}\overset{2}{1}\overset{1}{0}_2 = 2+16+32+64+128$

$\boxed{= 242_{10}}$

b. $1\ \overset{64}{1}\overset{32}{1}\overset{16}{1}\ \overset{8}{0}\overset{4}{0}\overset{2}{1}\overset{1}{0}_2 = -(2+16+32+64)$

Negative

$\boxed{= -114_{10}}$

c. $1111\ 0010_2 \rightarrow -(\overset{128}{0}\overset{64}{0}\overset{32}{0}\overset{16}{0}\ \overset{8}{1}\overset{4}{1}\overset{2}{0}\overset{1}{1}) = -(1+4+8)$

$\boxed{= -13_{10}}$

d.

$1111\ 0010_2 \rightarrow -(\overset{128\,64\,32\,16}{0000}\ \overset{8\,4\,2\,1}{1110}) = -(2+4+8)$

$\boxed{= -14_{10}}$

3.

// . taking 2's comp. of subtrahend & then adding

a)

$$\begin{array}{r} 11010 \\ -10001 \end{array} \Rightarrow \begin{array}{r} 11010 \\ +01111 \\ \hline 1\,|\,01001 \end{array} \Rightarrow ans = 01001$$

b)

$$\begin{array}{r} 11110 \\ -01110 \end{array} \Rightarrow \begin{array}{r} 11110 \\ +10010 \\ \hline 1\,|\,10000 \end{array} \Rightarrow ans = 10000$$

c)

$$\begin{array}{r} 1111110 \\ -1111110 \end{array} \Rightarrow \begin{array}{r} 1111110 \\ +0000010 \\ \hline 1\,|\,0000000 \end{array} \Rightarrow ans = 0000000$$

d)

$$\begin{array}{r} 101001 \\ -000101 \end{array} \Rightarrow \begin{array}{r} 101001 \\ +111011 \\ \hline 1\,|\,100100 \end{array} \Rightarrow ans = 100100$$

4.

a.

$$\begin{array}{r} 1100\ 1000 \\ +\ 0011\ 1000 \\ \hline 0000\ 0000 \end{array}$$

N = 0

Z = 1

C = 1

V = 0

Unsigned: C = 1 $\Downarrow$ overflow!

Signed: V = 0 $\Downarrow$ No overflow

---

b.

```
 1000 0000
+1000 0000
----------
 0000 0000
```

N = 0

Z = 1

C = 1

V = 1

Unsigned: C = 1

⇓

Overflow!

Signed: V = 1

⇓

Overflow!

c.

```
 1010 1010
+1100 1110
----------
 0111 1000
```

N = 0

Z = 0

C = 1

V = 1

Unsigned: C = 1

⇓

overflow!

Signed: V = 1

⇓

overflow!

d.

```
 1010 1010
+1110 0010
----------
 1000 1100
```

N = 1

Z = 0

C = 1

V = 0

Unsigned: C = 1

⇓

Overflow!

Signed: V = 0

⇓

No overflow!

5.

To represent these decimal numbers we would need at least 7 bits (Range: -64 -> 63)

Express the numbers as signed binary:

+36 = 010 0100
+35 = 010 0011 => -35 = 101 1101
+24 = 0011000 => -24 = 1101000

a.

```
  11
  0100100
+ 1101000
  -------
  0001100   (=+12)
```

b.

```
  101 1101         011
- 110 1000   =>    101 1101
                 + 0011000
                   -------
                   1110101   (=-11)
```

6.

```
a)   111 0001   -15      N=0     C=1
   + 011101     +29      Z=0     V=0
   1|001110     +14      no overflow

b)   10111011   +55      N=1     C=0
   + 010111     +47      Z=0     V=1
     1100110    +102     overflow

c)   00000111    +7
   - 11110100   -(-12)
   taking 2's comp of subtrahend
     00000111    +7      N=0     C=0
 => + 00001100   +12     Z=0     V=0
     00010011    +19     no overflow
```

d) 0110111 +55

− 0101111 − 47

taking 2's comp of subtrahend

0110111	+55	N=0	C=1
+ 1010001	+ (−47)	Z=0	V=0
1 | 0001000	+8		

7.

Given two unsigned 4-bit numbers:

$A_3 A_2 A_1 A_0$ and $B_3 B_2 B_1 B_0$.

$A < B$ if the MSB of A is < the MSB of B.

i.e. $A_3 < B_3$ or: $A_3 = 0$ and $B_3 = 1$.

Otherwise, if they are equal $(\overline{A_3 \oplus B_3} = 1)$ then we need to check the next bit - if $A_2 < B_2$.

(If $B_3 < A_3$ then $B < A$ and hence we don't need to check any further).

We can continue this argument to the LSB.

This leads to the expression for X:

$$X = \bar{A}_3 B_3 + (\overline{A_3 \oplus B_3})\bar{A}_2 B_2 + (\overline{A_3 \oplus B_3})(\overline{A_2 \oplus B_2})\bar{A}_1 B_1$$
$$+ (\overline{A_3 \oplus B_3})(\overline{A_2 \oplus B_2})(\overline{A_1 \oplus B_1})\bar{A}_0 B_0$$

8.

$$S_0 = \left[\overline{A_0B_0}\,(\overline{\overline{A_0+B_0}})\right] \oplus \overline{C_0} = \overline{A_0B_0}\,(A_0+B_0) \oplus C_0$$

$$= (\overline{A_0}+\overline{B_0})(A_0+B_0) \oplus C_0 = (\overline{A_0}B_0 + A_0\overline{B_0}) \oplus C_0$$

$$= (A_0 \oplus B_0) \oplus C_0 = A_0 \oplus B_0 \oplus C_0$$

→ which is sum of full adder

$$C_1 = \overline{\overline{A_0+B_0} + \overline{(A_0B_0)}\,\overline{C_0}} = \overline{(\overline{A_0}+\overline{B_0}) + (\overline{A_0}+\overline{B_0})\overline{C_0}}$$

$$= \overline{\overline{A_0}\,\overline{B_0} + (\overline{A_0}+\overline{B_0})\overline{C_0}} = (\overline{\overline{A_0}\,\overline{B_0}})(\overline{(\overline{A}+\overline{B_0})\overline{C_0}})$$

$$= (A_0+B_0)(C_0+\overline{\overline{A_0}+\overline{B_0}}) = (A_0+B_0)(C_0+A_0B_0)$$

$$= A_0B_0 + B_0C_0 + A_0C_0$$

```
// Full Adder: Structural Verilog Description
// Problem 8
module full_adder(C1, S0, X);
        input [2:0] X; //X is the vector of inputs (A0, B0, C0).
        output C1, S0;
            wire [0:6] N;
        //N[0:6] is the six bit vector of gate outputs from upper left to lower right

        nand
           gna(N[0],X[1],X[0]);
        nor
           gno1(N[1],X[1],X[0]),
           gno2(C1,N[1],N[3]);
        not
           gn0(N[2], X[2]),
           gn1(N[4], N[1]),
           gn2(N[5], N[2]);
        and
           ga0(N[3], N[0], N[2]),
           ga1(N[6], N[0], N[4]);
        xor
           gx(S0, N[5], N[6]);
endmodule
```

9.

Let $Y = Y_3Y_2Y_1Y_0$ be the 9's comp of the BCD digit $X = X_3X_2X_1X_0$

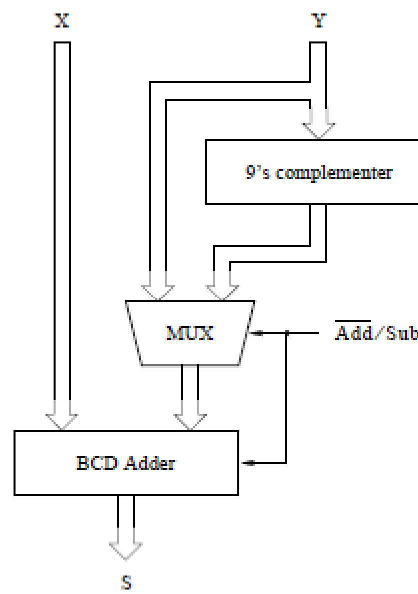
X_3	X_2	X_1	X_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	1	0	0	1
0	0	0	1	1	0	0	0
0	0	1	0	0	1	1	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	0	1
0	1	0	1	0	1	0	0
0	1	1	0	0	0	1	1
0	1	1	1	0	0	1	0
1	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0

$\Rightarrow Y_0 = \overline{X_0}$ $\qquad Y_2 = \overline{X_2}X_1 + X_2\overline{X_1} = X_1 \oplus X_2$

$Y_1 = X_1$ $\qquad Y_3 = \overline{X_3}\,\overline{X_2}\,\overline{X_1}$

10. BCD subtraction can be performed using 10's complement representation, using an approach that is similar to 2's complement subtraction. Let X and Y be BCD numbers given in 10's complement representation, such that the sign (left-most) BCD digit is 0 for positive numbers and 9 for negative numbers. Then, the subtraction operation S = X - Y is performed by finding the 10's complement of Y and adding it to X, ignoring any carry-out from the sign-digit position.

    For example, let X = 068 and Y = 043. Then, the 10's complement of Y is 957, and S = 068 + 957 = 1025. Dropping the carry-out of 1 from the sign-digit position gives S = 025. As another example, let X = 032 and Y = 043. Then, S = 032 + 957 = 989, which represents -11_{10}.
    The 10's complement of Y can be formed by adding 1 to the 9's complement of Y (using the circuit built in the previous problem). Therefore, a circuit that can add and subtract BCD operands would be as follows:


X
Y
9's complementer
MUX
$\overline{\text{Add}}$/Sub
BCD Adder
S