Class Notes
for EE 3755
Part I

Author: Alexander Skavantzos

EE 3755

Computer Arithmetic

Handout #1

Fixed Point (FXP) Systems: Addition, Subtraction

In this handout we present the basics of fixed point systems as well as how to do fixed point addition and subtraction. Both cases of unsigned as well as signed systems are considered.

A fixed point number is represented in general as

$$X = \underbrace{x_{n-1}x_{n-2}\cdots x_1 x_0}_{\text{integer part}} \bullet \underbrace{x_{-1}x_{-2}\cdots x_{-k}}_{\text{fractional part}}$$

(radix point)

If the radix used is r then the value of such number X is

$$X_{value} = x_{n-1}\cdot r^{n-1} + x_{n-2}\cdot r^{n-2} + \cdots + x_1 r^1 + x_0 r^0 + x_{-1}\cdot r^{-1} + x_{-2} r^{-2} + \cdots$$
$$\cdots + x_{-k}\cdot r^{-k} = \sum_{i=-k}^{n-1} x_i \cdot r^i$$

while the digits x_is belong in the set $\{0, 1, \cdots, r-1\}$.

The system is called fixed point because the radix point ($\bullet$) assumes a fixed location. The radix point can be located someplace in the middle of the number, in which case the number will consist of an integer as well as a fractional part (like in the number X above); it can be located after the right most digit (like $X = x_{n-1}\cdots x_1 x_0 \bullet$) in which case the number is 100% integer; or it can be located before the leftmost digit (like $X = \bullet x_{-1}x_{-2}\cdots x_{-n}$) in which case the number is 100% fractional.

If the radix is $r=10$ we then have a decimal system

if r=8 octal, if r=2 binary etc.

In this handout we will be concerned with binary systems only (r=2). Also our fixed point numbers will be assumed to be 100% integers (like $X = x_{n-1} \cdots x_1 x_0$).

Unsigned binary Fixed Point (FXP) systems:

In such a system an n-bit unsigned number is of the form $X = x_{n-1} x_{n-2} \cdots x_1 x_0$. Here, all the bits of X represent magnitude (there is no sign bit included). The bit x_{n-1} is the most significant bit (MSB) while x_0 is the least significant bit (LSB). The Dynamic Range (DR) of an n-bit binary unsigned FXP system is $DR = [0 \quad 2^n - 1]$.

Signed binary FXP systems:

In such systems an n-bit signed number is of the form $X = x_{n-1} x_{n-2} \cdots x_1 x_0$ where the left most bit x_{n-1} is the sign bit (SB). If $x_{n-1} = 0$ that means that X is positive while if $x_{n-1} = 1$ that means that X is negative.

There are three different systems for representing signed FXP numbers: The sign magnitude system, the 1's complement system and the 2's complement system

- ### Sign magnitude system

  Here any positive X (X>0) is represented as

  $X = \underbrace{0}_{\text{sign bit}} \underbrace{x_{n-2} x_{n-3} \cdots x_1 x_0}_{\text{magnitude}}$

The additive inverse of X is then represented as

$$-X = \underbrace{1}_{SB}\ \underbrace{x_{n-2}x_{n-3}\cdots x_1 x_0}_{\text{magnitude}}$$

The Dynamic Range of an n-bit binary sign magnitude system is

$$DR = [-(2^{n-1}-1) \qquad 2^{n-1}-1]$$

In a sign magnitude system the number zero has two representations

$+0 = 000\cdots00$
$-0 = 100\cdots00$

- 1's complement system

Here any positive number X $(X>0)$ is represented as

$$X = \underbrace{0}_{SB}\ x_{n-2}x_{n-3}\cdots x_1 x_0$$

The additive inverse of X is then represented as

$$-X = \underbrace{1}_{SB}\ \overline{x_{n-2}}\ \overline{x_{n-3}}\cdots\overline{x_1}\ \overline{x_0}$$ (where $\overline{\ }$ means the NOT operator)

The Dynamic Range of an n-bit 1's complement system is the same as the DR of the sign magnitude system or

$$DR = [-(2^{n-1}-1) \qquad 2^{n-1}-1]$$

In the 1's complement system the number zero has again two different representations:

$+0 = 000\cdots00$
$-0 = 111\cdots11$

- 2's complement system

Here, again, any positive number X $(X>0)$ is represented as $X = 0\, x_{n-2} x_{n-3} \cdots x_1 x_0$

SB

The additive inverse of X is then

$$-X = (1\, \overline{x_{n-2}}\, \overline{x_{n-3}} \cdots \overline{x_1}\, \overline{x_0}) + 1$$

The Dynamic Range of an n-bit 2's complement system is by one number larger than the DR of the 1's compl. or sign magnitude systems or

$$DR = [-2^{n-1} \quad 2^{n-1}-1]$$

In the 2's complement system the number zero has a unique representation. Observe that

$+0 = 000\ldots00$

while $-0 = 111\ldots11$

$+)\quad 1$

$000\ldots00$

ignore carry out

-0

The reason why the 2's complement system has one more slot in its Dynamic Range (compared to the sign magnitude or the 1's complement) is the fact that the zero has one unique representation (as opposed to two different that the other systems require).

Addition and Subtraction in signed binary FXP systems

In this section we will show how additions and subtractions are performed in the sign magnitude, 1's complement and 2's complement systems. Discussions about overflow and underflow will be included.

- **Addition/subtraction in the sign magnitude system**

When doing additions or subtractions in a sign magnitude system we basically have to add or subtract the magnitudes of the two numbers and make sure that we return the correct sign bit of the result. There are two cases: The case of a true addition and the case of a true subtraction. A true addition has to take place when adding numbers of the same sign or subtracting numbers of different signs. A true subtraction has to take place when subtracting numbers of the same sign or adding numbers of different signs. Since sign magnitude additions/subtractions involve additions/subtractions between unsigned magnitudes, we have to show how we can do additions/subtractions between unsigned numbers.

– **Addition between unsigned numbers**

Suppose that X and Y are two fixed point n-bit unsigned numbers $X = x_{n-1}\, x_{n-2} \cdots x_1\, x_0$ and $Y = y_{n-1} y_{n-2} \cdots y_1 y_0$

(x_{n-1}: MSB, x_0: LSB; y_{n-1}: MSB, y_0: LSB)

In order to compute $X+Y$ we just perform the regular binary addition between vectors $X = x_{n-1} \cdots x_1 x_0$ and $Y = y_{n-1} \cdots y_1 y_0$.

Suppose that

$$\begin{array}{rl} X+Y = & x_{n-1} \cdots x_1 x_0 \\ +) & y_{n-1} \cdots y_1 y_0 \\ \hline C & z_{n-1} \cdots z_1 z_0 \end{array}$$

Carry out of addition ——→ C

In the above, c is the carry out of the addition. Then if $c=0$, the correct sum $X+Y$ is $Z = z_{n-1} z_{n-2} \cdots z_1 z_0$. However, if $c=1$, then an overflow occured which means that the result $X+Y$ is outside the Dynamic Range (or $X+Y > 2^n - 1$). In this case the vector $z_{n-1} z_{n-2} \cdots z_1 z_0$ is not the correct sum $X+Y$.

Example 1: Add the following two 4-bit unsigned numbers:
$X = (5)_{10} = (0101)_2$; $Y = (6)_{10} = (0110)_2$.

The Dynamic Range of a 4-bit unsigned system is $[0 \quad 15]$.

$$X+Y = \begin{array}{r} 0101 \\ 0110 \\ \hline 1011 \end{array}$$

Since $c=0$ there is no overflow and $X+Y = (1011)_2 = (11)_{10}$ $\left(11 \in [0 \quad 15]\right)$.

Example 2: Add the following two 4-bit unsigned numbers:
$X = (10)_{10} = (1010)_2$; $Y = (8)_{10} = (1000)_2$.

The DR of a 4-bit unsigned system is again $[0 \quad 15]$.

$$X+Y = \begin{array}{r} 1010 \\ 1000 \\ \hline 10010 \end{array}$$

→ since $c=1$ that means that an overflow occured.
Observe that $X+Y = (18)_{10} > 15$ (outside DR).

— Subtraction between unsigned numbers

Subtraction between two unsigned numbers might result in a negative result which will then need a sign bit.

Suppose that X and Y are two fixed point n-bit unsigned numbers $X = x_{n-1} x_{n-2} \cdots x_1 x_0$ and $Y = y_{n-1} y_{n-2} \cdots y_1 y_0$

(x_{n-1}: MSB, x_0: LSB, y_{n-1}: MSB, y_0: LSB)

In order to compute $X-Y$ we have to perform the binary addition $X + (2\text{'s complement of } Y)$. Suppose that the binary addition $X + (2\text{'s complement of } Y)$ results in the binary vector

$(c \; z_{n-1} z_{n-2} \cdots z_1 z_0)_2$

carry out of addition → c

In the above, c is the carry out of the addition. Then

- if $c=1$ that means that result $= X-Y \geq 0$ (or $X \geq Y$). In this case $X-Y = (z_{n-1} z_{n-2} \cdots z_1 z_0)_2$
- if $c=0$ that means that result $= X-Y < 0$ (or $X < Y$). In this case the result $X-Y$ must have a negative (−) sign while the magnitude of the difference will be $=$ (2's complem. of $(z_{n-1} \cdots z_1 z_0)$). In other words if $c=0$ then $X-Y = -($(2's compl. of $(z_{n-1} \cdots z_1 z_0)$)).

Example 3: Perform the subtraction $X-Y$ where X and Y are the following 4-bit unsigned numbers: $X=(7)_{10}=(0111)_2$; $Y=(5)_{10}=(0101)_2$.

$7 = \quad 0111$

2's compl. of $5 = 1011$

$+) \quad 10010$

↳ since $c=1 \Rightarrow$ result $\geq 0 \Rightarrow 7-5 = (0010)_2 = (2)_{10}$.

Example 4: Perform the subtraction $X-Y$ where X and Y are the following 4-bit unsigned numbers: $X=(5)_{10}=(0101)_2$; $Y=(7)_{10}=(0111)_2$.

$5 = 0101$

2's compl. of $7 = 1001$

$+) \quad 01110$

↳ since $c=0 \Rightarrow$ result $< 0 \Rightarrow 5-7 = -($(2's compl. of (1110))$)$

$= -(0010)_2 = (-2)_{10}$.

Example 5: Perform the subtraction $X-Y$ where X and Y are the following 4-bit unsigned numbers: $X=(5)_{10}=(0101)_2$; $Y=(5)_{10}=(0101)_2$.

$5 = 0101$

2's compl. of $5 = 1011$

$+) \quad 10000$

↳ since $c=1 \Rightarrow$ result $\geq 0 \Rightarrow 5-5 = (0000)_2 = (0)_{10}$

Question: When performing subtraction between two unsigned numbers can we check the sign bit (the bit just right of the carry out bit) instead of the carry out bit, in order to determine if difference ≥ 0 or difference < 0?

Answer: NO

The following two examples will clarify the above issue.

Example 6: Perform the subtraction $X-Y$ where X and Y are the following 4-bit unsigned numbers: $X=(15)_{10}=(1111)_2$; $Y=(1)_{10}=(0001)_2$.

$$\begin{array}{rl} & 15 = 1111 \\ 2\text{'s complement of} & 1 = 1111 \\ \hline +) & 1\ 1110 \end{array}$$

sign bit = 1

carry out = 1

The carry out being equal to 1 suggests that the difference $X-Y$ is ≥ 0 (or that $15-1=(1110)_2=(+14)_{10}$ which is correct). On the other hand, the sign bit being equal to 1 suggests that the difference $X-Y$ is <0 (which is wrong).

Example 7: Perform the subtraction $X-Y$ where X and Y are the following 4-bit unsigned numbers: $X=(1)_{10}=(0001)_2$; $Y=(15)_{10}=(1111)_2$.

$$\begin{array}{rl} & 1 = 0001 \\ 2\text{'s complement of } 15 = & 0001 \\ \hline +) & 0\ 0010 \end{array}$$

sign bit = 0.

carry out bit = 0.

The carry out bit being equal to 0 suggests that the difference $X-Y<0$. Thus $1-15=-(2\text{'s complement of } (0010))=-(1110)_2=(-14)_{10}$

(which is the correct result). On the other hand, the sign bit being equal to 0 suggests that the difference $X-Y \geq 0$ (which is wrong).

Example 8: Perform the addition $X+Y$ where X and Y are the following 5-bit sign magnitude numbers: $X = (01100)_2 = (+12)_{10}$, $Y = (11111)_2 = (-15)_{10}$.

Here the two numbers X and Y are of different signs and the addition $X+Y$ is to be performed. We thus have to perform the following subtraction:

(magnitude of X) − (magnitude of Y) $= (1100)_2 - (1111)_2$

$= (1100)_2 + (2\text{'s complement of } (1111)_2) = (1100)_2 + (0001)_2$

$$\begin{array}{r} 1100 \\ +)\ 0001 \\ \hline 0\ 1101 \end{array}$$

↳ since $c=0 \Rightarrow$ result $< 0 \Rightarrow$

$\Rightarrow$ (magnitude of X) − (magnitude of Y) < 0

$\Rightarrow$ magnitude of $X <$ magnitude of Y

$\Rightarrow$ { sign bit of result = sign bit of number with the largest magnitude = sign bit of $Y = 1$ and magnitude of $X+Y$ = 2's compl. of $(1101) = 0011$ }

Thus $X+Y = (10011)_2 = (-3)_{10}$.

Note: In the case of subtraction between unsigned numbers, an overflow can never occur.

• Addition/subtraction in the 1's complement system

– Addition in the 1's complement system

Suppose that X and Y are two n-bit signed numbers (the 1's complement system is used for representing signed numbers here). Suppose that $X = x_{n-1}x_{n-2}\cdots x_1x_0$ and $Y = y_{n-1}y_{n-2}\cdots y_1y_0$ where x_{n-1} and y_{n-1} are the sign bits of X and Y. Then in order to compute $X+Y$ we have to perform the binary addition between vectors $X = x_{n-1}\cdots x_1x_0$ and $Y = y_{n-1}\cdots y_1y_0$ and add to the result the created carry out (end around carry out etc). In other words we do

$$\begin{array}{r} x_{n-1}\,x_{n-2}\cdots x_1\,x_0 \\ +)\ y_{n-1}\,y_{n-2}\cdots y_1\,y_0 \\ \hline c\ \ z_{n-1}\,z_{n-2}\cdots z_1\,z_0 \end{array}$$

(c: carry out of addition)

Then $X+Y =$

$$\begin{array}{r} z_{n-1}\,z_{n-2}\cdots z_1\,z_0 \\ +)\qquad\qquad\qquad c \\ \hline w_{n-1}\,w_{n-2}\cdots w_1\,w_0 \end{array}$$

adding carry out

(w_{n-1}: sign bit; $w_{n-2}\cdots w_1w_0$: $X+Y$)

Example 9: Perform the addition $X+Y$ where X and Y are the following 4-bit signed numbers (1's complement system is used for representing signed numbers):

$X = (1110)_2 = (-1)_{10}$; $Y = (0100)_2 = (+4)_{10}$.

$$\begin{array}{r} 1110 \\ +)\ 0100 \\ \hline 0010 \end{array}$$

carry out = 1

$$\begin{array}{r} 0010 \\ +)\qquad 1 \\ \hline 0011 \end{array}$$

add carry out

$\hookrightarrow X+Y = (0011)_2 = (3)_{10}$

— Subtraction in the 1's complement system

Suppose that $X = x_{n-1} \cdots x_1 x_0$ and $Y = y_{n-1} \cdots y_1 y_0$ are two n-bit signed numbers (1's complement system used for representing signed numbers). Then

$X - Y = X +$ 1's complement of Y (end around carry out etc).

Example 10: Perform $X - Y$ where X and Y are the following 4-bit signed numbers (1's compl. system is used for representing signed numbers): $X = (1110)_2 = (-1)_{10}$;

$Y = (1011)_2 = (-4)_{10}$.

$X - Y = X +$ 1's compl. of $Y = (1110) +$ 1's compl. of (1011)

$= (1110) + (0100)$

$$\begin{array}{r} 1110 \\ 0100 \\ \hline 10010 \end{array} \qquad \begin{array}{r} 0010 \\ \quad 1 \text{ add carry out} \\ \hline 0011 \end{array}$$

$\hookrightarrow X - Y = (0011)_2 = (3)_{10}$

— Overflow/Underflow:

An overflow/underflow might occur when adding numbers of the same sign. When adding numbers of different signs, overflow or underflow can never occur. Overflows and underflows are easily detectable as follows: If two positive numbers added together result in a negative result, this indicates an overflow; and if two negative numbers added together result in a positive result, this indicates an underflow.

Example 11: Using the 1's complement system perform the addition of the 4-bit numbers $X = (0101)_2 = (+5)_{10}$ and $Y = (0011)_2 = (+3)$.

```
  0101
+)0011
----
 1000
```

↳ sign bit = 1 means that result is negative. Since the two numbers that were added were both positive that means that an overflow occured. This was expected since $+5+3=8$ and 8 is outside the Dyn. Range of a 4-bit 1's compl system ($DR = [-7 \quad 7]$).

Example 12: Using the 1's complement system perform the addition of the 4-bit numbers $X=(1010)_2=(-5)_{10}$ and $Y=(1011)_2=(-4)_{10}$.

```
  1010
+)1011
----
1 0101
```

carry out

```
  0101
+)   1   add carry out
----
 0110
```

↳ sign bit = 0 means that result is positive. Since the two numbers that were added were both negative, that means that an underflow occured. This was expected since $(-5)+(-4)=-9$ and -9 is outside the DR ($DR = [-7 \quad 7]$).

- Addition/subtraction in the 2's complement system

Suppose that $X = x_{n-1}x_{n-2}\cdots x_1x_0$ and $Y = y_{n-1}y_{n-2}\cdots y_1y_0$ are two n-bit signed numbers (the 2's complement system is used here for representing signed numbers). The bits x_{n-1} and y_{n-1} are the sign bits of X and Y. In order to do the addition $X+Y$ we have to

perform the binary addition between vectors $X = x_{n-1}x_{n-2}\cdots x_1x_0$ and $Y = y_{n-1}y_{n-2}\cdots y_1y_0$ and just ignore the created carry out. Or in other words

$$\begin{array}{r} x_{n-1}\,x_{n-2}\,\cdots\,x_1\,x_0 \\ +)\ y_{n-1}\,y_{n-2}\,\cdots\,y_1\,y_0 \\ \hline \not{c}\ \ z_{n-1}\,z_{n-2}\,\cdots\,z_1\,z_0 \end{array}$$

ignore carry out

$z_{n-1}\,z_{n-2}\cdots z_1\,z_0 = X+Y$

sign bit.

In order to do the subtraction $X-Y$ all we do is X + 2's complement of Y (ignore carry out). of course 2's complement of Y = (1's complement of Y) + 1.

Example 13: Using the 2's complement system perform the addition $X+Y$ where $X = (1110)_2 = (-2)_{10}$ and $Y = (0110)_2 = (+6)_{10}$.

$$\begin{array}{r} 1\,1\,1\,0 \\ +)\ 0\,1\,1\,0 \\ \hline \not{1}\ \ 0\,1\,0\,0 \end{array}$$

ignore

$\hookrightarrow X+Y = (0100)_2 = (+4)_{10}$

Example 14: Using the 2's complement system perform the subtraction $X-Y$ where $X = (1110)_2 = (-2)_{10}$ and $Y = (1010)_2 = (-6)_{10}$.

$X-Y = X$ + 2's complement of Y = (1110) + 2's compl. of (1010)

$= (1110) + \big((0101)+1\big) = (1110)+(0110)$

$$\begin{array}{r} 1\,1\,1\,0 \\ +)\ 0\,1\,1\,0 \\ \hline \not{1}\ \ 0\,1\,0\,0 \end{array}$$

ignore

$\hookrightarrow X-Y = (0100)_2 = (+4)_{10}$.

– Overflow / Underflow:

An overflow or underflow might occur only when adding numbers of the same sign. When adding numbers of different signs, overflow or underflow can never occur. Overflows and underflows are easily detectable as follows: If two positive numbers added together result in a negative result, this indicates an overflow; and if two negative numbers added together result in a positive result, this indicates an underflow.

Example 15: Using the 2's complement system perform the addition of the 4-bit numbers $X = (0110)_2 = (+6)_{10}$ and $Y = (0101)_2 = (+5)_{10}$

$$\begin{array}{r} 0110 \\ +)\ 0101 \\ \hline 1011 \end{array}$$

ignore ⊗

→ since sign bit = 1 that means that an overflow occured (observe that the D.R of a 4-bit 2's complem. system is DR = [−8 7] and the sum +6 + 5 = +11 while 11 > 7).

Example 16: Using the 2's complement system perform the addition of the 4-bit numbers $X = (1100)_2 = (-4)_{10}$ and $Y = (1011)_2 = (-5)_{10}$.

$$\begin{array}{r} 1100 \\ +)\ 1011 \\ \hline 0111 \end{array}$$

ignore ⊗

→ Here, since the two numbers added are negative and a positive result is returned that means that an underflow occured. This was expected since DR = [−8 +7] and $(-4) + (-5) = -9 < -8$.