Lecture 5  Fixed Point vs Floating Point

Objectives:
- Understand fixed point representations
- Understand scaling, overflow and rounding in fixed point
- Understand Q-format
- Understand TMS320C67xx floating point representations
- Understand relationship between the two in C6x architecture


Q-format notation
- Q-format representation:
  - if N=16, 15 bit fractional representation ⇒ Q15 format
- Rule:
  - $Q_m + Q_n ⇒ Q_m$
  - $Q_m \times Q_n ⇒ Q_{m+n}$
- Assume 16-bit data format, $Q_{15} \times Q_{15} ⇒ Q_{30}$

How to store Q30 number to 16-bit memory?
- Storing Q30 number to 16-bit memory requires rounding or truncation:

$$ x' = -b_{N-1}2^{N-1} + b_{N-2}2^{N-2} + \ldots + b_12^1 + b_02^0 $$

- Q-Format number representation
- N-bit fixed point, 2’s complement number is given by:

$$ x = -b_{N-1}2^{N-1} + b_{N-2}2^{N-2} + \ldots + b_12^1 + b_02^0 $$

- Difficult to work with due to possible overflow & scaling problems
- Often normalise number to some fractional representation (e.g. between $±1$)

$$ x' = -b_{N-1}2^0 + b_{N-2}2^{-1} + \ldots + b_12^{-N+2} + b_02^{-N+1} $$

How to store Q30 number to 16-bit memory?
- Storing Q30 number to 16-bit memory requires rounding or truncation:

$$ x' = -b_{N-1}2^0 + b_{N-2}2^{-1} + \ldots + b_12^{-N+2} + b_02^{-N+1} $$

- Rounding:
  - if $r = 0$, round down,
  - if $r = 1$, round up

How to store Q30 number to 16-bit memory?
- Storing Q30 number to 16-bit memory requires rounding or truncation:

$$ x' = -b_{N-1}2^0 + b_{N-2}2^{-1} + \ldots + b_12^{-N+2} + b_02^{-N+1} $$

- Rounding:
  - if $r = 0$, round down,
  - if $r = 1$, round up

How to store Q30 number to 16-bit memory?
- Storing Q30 number to 16-bit memory requires rounding or truncation:

$$ x' = -b_{N-1}2^0 + b_{N-2}2^{-1} + \ldots + b_12^{-N+2} + b_02^{-N+1} $$

- Rounding:
  - if $r = 0$, round down,
  - if $r = 1$, round up

How to store Q30 number to 16-bit memory?
- Storing Q30 number to 16-bit memory requires rounding or truncation:

$$ x' = -b_{N-1}2^0 + b_{N-2}2^{-1} + \ldots + b_12^{-N+2} + b_02^{-N+1} $$

- Rounding:
  - if $r = 0$, round down,
  - if $r = 1$, round up

How to store Q30 number to 16-bit memory?
- Storing Q30 number to 16-bit memory requires rounding or truncation:

$$ x' = -b_{N-1}2^0 + b_{N-2}2^{-1} + \ldots + b_12^{-N+2} + b_02^{-N+1} $$

- Rounding:
  - if $r = 0$, round down,
  - if $r = 1$, round up

How to store Q30 number to 16-bit memory?
- Storing Q30 number to 16-bit memory requires rounding or truncation:

$$ x' = -b_{N-1}2^0 + b_{N-2}2^{-1} + \ldots + b_12^{-N+2} + b_02^{-N+1} $$

- Rounding:
  - if $r = 0$, round down,
  - if $r = 1$, round up

How to store Q30 number to 16-bit memory?
- Storing Q30 number to 16-bit memory requires rounding or truncation:

$$ x' = -b_{N-1}2^0 + b_{N-2}2^{-1} + \ldots + b_12^{-N+2} + b_02^{-N+1} $$

- Rounding:
  - if $r = 0$, round down,
  - if $r = 1$, round up
Avoid overflow with SADD

- SADD - saturation add instruction
- Always clip to max (or min) possible
- Set bit 9 of the CSR register to indicate saturation has occurred

Example

<table>
<thead>
<tr>
<th>Before instruction</th>
<th>1 cycle after instruction</th>
<th>2 cycles after instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>4367.71F2h</td>
<td>1130852850</td>
</tr>
<tr>
<td>A2</td>
<td>5A2E 51A3h</td>
<td>1512984995</td>
</tr>
<tr>
<td>A3</td>
<td>XXXX XXXXh</td>
<td>7PPP PPPPh</td>
</tr>
<tr>
<td>CSR</td>
<td>0001 0100h</td>
<td>CSR</td>
</tr>
</tbody>
</table>

SADD .L1 A1,A2,A3

Safe add routine in C to avoid overflow

```c
short safe_add(short A, short B, int *status) {
    int X,Y,result, SAT_BIT;
    X = A << 16;
    Y = B << 16;
    result = _sadd(X,Y);
    SAT_BIT = GET_REG_BIT(CSR,9);
    if(SAT_BIT==1) {
        //Overflow Occurred
        RESET_REG_BIT(CSR,9); //Reset Sat Bit
        *status = 1;
    } else {
        *status = 0;
    }
    return (result >> 16);
}
```

Single Precision Floating Point number

- Easy (and lazy) way of dealing with scaling problem
- 32-bit single precision floating point:
  - MSB is sign-bit (same as fixed point)
  - 8-bit exponent in bias-127 integer format (i.e., add 127 to it)
  - 23-bit to represent only the fractional part of the mantissa. The MSB of the mantissa is ALWAYS `1`, therefore it is not stored

```
x = -1^e \times 2^{exp-127} \times 1.frac
1.175 \times 10^{-38} < |x| < 1.7 \times 10^{38}
```

Double Precision Floating Point number

- 64-bit double precision floating point:
  - MSB is sign-bit (same as fixed point)
  - 11-bit exponent in bias-1023 integer format (i.e., add 1023 to it)
  - 52-bit to represent only the fractional part of the mantissa. The MSB of the mantissa is ALWAYS `1`, therefore it is not stored

```
x = -1^e \times 2^{exp-1023} \times 1.frac
2.2 \times 10^{-308} < |x| < 1.7 \times 10^{308}
```
Examples

■ Convert 5.75 to SP FP
  - 5.75 to binary: +1.01110000... x 2^2
  - exponent in bias-127 is 127+2 = 129 = 1000 0000b
  - The fractional part is .01110000... after we drop the hidden ‘1’ bit.
  - Answer: 0 10000001 0111000 00...00 = 40B80000 (hex)

■ Convert 0.1 to DP FP
  - 0.1 to binary: 1.10011001(1001 repeats) x 2^-4
  - exponent in bias-1023 is 1023-4 = 1019 = 011 1111 1011b
  - The fractional part is .10011001...1010 after we drop the hidden ‘1’ bit and rounding
  - Answer: 0 01111111011 1001100 ...1001 1010 = 3FB9 9999 9999 999A (hex).

Problems of Q-format

■ Wrong Q-format representation will give totally wrong results
■ Even correct use of Q-format notation may reduce precision
■ For this example, Q12 result is totally wrong, and Q8 result is imprecise:

Q12 → 7.50195 0111. 1000 0000 1000
Q12 → 7.25 0111. 0100 0000 0000
Q24 → 54.38916 0110 0110 0110 0011 1010 0000 0000
Q12 → 6.38916
Q8 → 54.38281

Special SP numbers

■ IEEE floating point standard has a set of special numbers:

<table>
<thead>
<tr>
<th>Special value</th>
<th>Sign (s)</th>
<th>Exponent (e)</th>
<th>Fraction (f)</th>
<th>Hex Value</th>
<th>Decimal value</th>
</tr>
</thead>
<tbody>
<tr>
<td>+0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0x0000 0000</td>
<td>0.0</td>
</tr>
<tr>
<td>-0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0x8000 0000</td>
<td>-0.0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>127</td>
<td>0</td>
<td>0x3F80 0000</td>
<td>1.0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>128</td>
<td>0</td>
<td>0x4000 0000</td>
<td>2.0</td>
</tr>
<tr>
<td>+Inf</td>
<td>0</td>
<td>255</td>
<td>0</td>
<td>0x7F80 0000</td>
<td>+∞</td>
</tr>
<tr>
<td>-Inf</td>
<td>1</td>
<td>255</td>
<td>0</td>
<td>0xFF80 0000</td>
<td>-∞</td>
</tr>
<tr>
<td>NaN</td>
<td>x</td>
<td>255</td>
<td>Nonzero</td>
<td>0x7FFF FFFF</td>
<td>not a number</td>
</tr>
<tr>
<td>LFPN</td>
<td>0</td>
<td>254</td>
<td>All 1’s</td>
<td>0x7F7F FFFF</td>
<td>3.40282347 e+38</td>
</tr>
<tr>
<td>SFPN</td>
<td>0</td>
<td>1</td>
<td>All 0’s</td>
<td>0x0080 0000</td>
<td>1.17549435e-38</td>
</tr>
</tbody>
</table>
Special DP numbers

- Double precision floating point special numbers:

<table>
<thead>
<tr>
<th>Special value</th>
<th>Exponent (e)</th>
<th>Fraction (f)</th>
<th>Hex Value</th>
<th>Decimal value</th>
</tr>
</thead>
<tbody>
<tr>
<td>+0</td>
<td>0</td>
<td>0</td>
<td>0x0000 0000 0000 0000</td>
<td>0.0</td>
</tr>
<tr>
<td>-0</td>
<td>0</td>
<td>0</td>
<td>0x8000 0000 0000 0000</td>
<td>-0.0</td>
</tr>
<tr>
<td>1</td>
<td>1023</td>
<td>0</td>
<td>0x3FF0 0000 0000 0000</td>
<td>1.0</td>
</tr>
<tr>
<td>2</td>
<td>1024</td>
<td>0</td>
<td>0x4000 0000 0000 0000</td>
<td>2.0</td>
</tr>
<tr>
<td>+Inf</td>
<td>2047</td>
<td>0</td>
<td>0x7FF0 0000 0000 0000</td>
<td>+∞</td>
</tr>
<tr>
<td>-Inf</td>
<td>2047</td>
<td>0</td>
<td>0xFFF0 0000 0000 0000</td>
<td>-∞</td>
</tr>
<tr>
<td>NaN</td>
<td>2047</td>
<td>Nonzero</td>
<td>0x7FFF FFFF FFFF FFFF</td>
<td>not a number</td>
</tr>
<tr>
<td>LFPN</td>
<td>2046</td>
<td>All 1's</td>
<td>0x7FFF FFFF FFFF FFFF</td>
<td>1.7976931348623157 e+308</td>
</tr>
<tr>
<td>SFPN</td>
<td>1</td>
<td>All 0's</td>
<td>0x0001 0000 0000 0000</td>
<td>2.2250738585072014 e-308</td>
</tr>
</tbody>
</table>

Four functional units for each datapath

<table>
<thead>
<tr>
<th>Functional Unit</th>
<th>Fixed-Point Operations</th>
<th>Floating-Point Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>.L unit (L1, L2)</td>
<td>32/40-bit arithmetic and compare operations&lt;br&gt;Leftmost 1 or 0 bit counting for 32 bits&lt;br&gt;Normalization count for 32 and 40 bits&lt;br&gt;32-bit logical operations</td>
<td>Arithmetic operations&lt;br&gt;Conversion operations: DP → SP, INT → DP, INT → SP</td>
</tr>
<tr>
<td>.S unit (S1, S2)</td>
<td>32-bit arithmetic operations&lt;br&gt;32/40-bit shifts and 32-bit bit-field operations&lt;br&gt;32-bit logical operations&lt;br&gt;Branches&lt;br&gt;Constant generation&lt;br&gt;Register transfers to/from the control register file (.S2 only)</td>
<td>Compare reciprocal and reciprocal square-root operations&lt;br&gt;Absolute value operations&lt;br&gt;SP to DP conversion operations</td>
</tr>
<tr>
<td>.M unit (M1, M2)</td>
<td>16 × 16 bit multiply operations</td>
<td>32 × 32 bit multiply operations&lt;br&gt;Floating-point multiply operations</td>
</tr>
<tr>
<td>.D unit (D1, D2)</td>
<td>32-bit add, subtract, linear and circular address calculation&lt;br&gt;Loads and stores with a 5-bit constant offset</td>
<td>Load double word with a 5-bit constant offset</td>
</tr>
</tbody>
</table>

Mapping of instructions to functional units

<table>
<thead>
<tr>
<th>Functional Unit</th>
<th>S Unit</th>
<th>L Unit</th>
<th>D Unit</th>
<th>M Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ADD</td>
<td>NEG</td>
<td>ADD</td>
<td>ADD</td>
</tr>
<tr>
<td></td>
<td>ADK</td>
<td>OR</td>
<td>ADD</td>
<td>SUB</td>
</tr>
<tr>
<td></td>
<td>AND</td>
<td>SET</td>
<td>OR</td>
<td>SUB</td>
</tr>
<tr>
<td></td>
<td>CLR</td>
<td>SHR</td>
<td>OR</td>
<td>SUB</td>
</tr>
<tr>
<td></td>
<td>EXT</td>
<td>SHL</td>
<td>OR</td>
<td>SUB</td>
</tr>
<tr>
<td></td>
<td>MV</td>
<td>SRL</td>
<td>SUB</td>
<td>SUB</td>
</tr>
<tr>
<td></td>
<td>MVC</td>
<td>SHL</td>
<td>SUB</td>
<td>SUB</td>
</tr>
<tr>
<td></td>
<td>MK</td>
<td>SHR</td>
<td>SUB</td>
<td>SUB</td>
</tr>
<tr>
<td></td>
<td>MKH</td>
<td>SRL</td>
<td>SUB</td>
<td>SUB</td>
</tr>
<tr>
<td></td>
<td>ADDAB</td>
<td>XOR</td>
<td>MR</td>
<td>LDB</td>
</tr>
<tr>
<td></td>
<td>LDB</td>
<td>NOT</td>
<td>LDB</td>
<td>MPY</td>
</tr>
<tr>
<td></td>
<td>LW</td>
<td>XOR</td>
<td>LW</td>
<td>MPYH</td>
</tr>
<tr>
<td></td>
<td>MV</td>
<td>XOR</td>
<td>MPYI</td>
<td>MPYH</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ZERO</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

No Unit Used

- NOP
- IDLE
Detailed internal datapaths