TSTE18 Digital Arithmetic
Seminar 8

Oscar Gustafsson

Floating-point arithmetic

- **Floating-point representations**
- **IEEE 754**
- **Rounding**

- Floating-point addition/subtraction
- Floating-point multiplication
- Floating-point division
- Fused floating-point operation

Dynamic range

- Using a finite word length of N radix-b digits there are at most b^N numbers that can be represented
- How to select the distribution?
- Often, it is useful to be able to use the same number representation for both large and small numbers
- The precision requirements are often relative to the represented numbers
- Example:
  - Calculate distances between stars – OK with a precision in meters
  - Calculate distances between atoms – Not OK with a precision in meters

Dynamic range

- Consider two binary fixed-point numbers

  ```
  0000 0000.0000 1011
  1011 0000.0000 0000
  ```
- Computing the square of any of these numbers will lead to results which are not representable

Floating-point representation

- The value of a generic floating-point number is defined by a triple (S_X, E_X, F_X) as

$$X = (-1)^{S_X} b^{E_X} F_X \tag{1}$$

  where S_X denotes the sign, E_X the exponent (characteristic), and F_X the significand (mantissa)
- This leads to a much higher dynamic range of the numbers, since the exponent can be used to "move" the position of the binary point
- A floating-point representation is in general redundant since shifting F_X one position (assuming radix-b representation) corresponds to addition/subtracting 1 to E_X
- It is therefore common to use a normalized significand such that it only can take on a certain range of values $0 < F_{\min} \leq F \leq F_{\max}$

Floating-point representation

- To simplify the discussion we assume, without loss of generality, that $1 \leq F < b$
- This leads to that $F = f_0.f_1 f_2 \ldots$ with, assuming a non-redundant digit set typically $f_i \in \{0, 1, \ldots, b-1\}$, leads to a non-redundant representation with $1 \leq f_0 \leq b-1$
- For a binary representation this gived $F = 1.f_1 f_2 \ldots$, and, hence, only the fractional bits must be stored

Dynamic range and resolution

- The ulp of a floating-point representation depends on the exponent
- Therefore, the resolution of a floating-point number is highly dependent on the order of the number
- On average, the resolution is lower for floating-point compared to fixed-point
- However, the relative resolution is more or less constant (within one digit)
- Also, the dynamic range (ratio between largest and smallest number) is significantly higher

IEEE 754-2008

- IEEE started standardizing floating-point formats in the seventies
- Before that the number of bits and the partitioning was dependent on the manufacturer
- This also included rounding and how to handle overflows etc
- IEEE 754-2008 defines several formats
- We will initially consider the single precision format, called binary32

IEEE 754-2008

- The binary32 format has a sign bit, eight exponent bits using excess-127 representation, and 23 bits for the significand plus a hidden leading one
- The excess-127 means that 127 should be added to the actual exponent to obtain the stored value, hence, a stored value of 130 would correspond to an exponent of $130 - 127 = 3$
- The representation can be visualized as

$$\underbrace{s}_{\text{Sign}}\ \underbrace{e_{-7}e_{-6}e_{-5}e_{-4}e_{-3}e_{-2}e_{-1}e_0}_{E\ \text{8 bits biased exponent}}\ \underbrace{f_1f_2f_3f_4f_5f_6\ldots f_{22}f_{23}}_{F\ \text{23 bits unsigned fraction}}$$

- The value of the floating-point number is given by

$$X = (-1)^s\, 1.F\, 2^{E-127}. \tag{2}$$

- Note the hidden one due to the normalized number system, so $M = 1.F$
- This means that the actual mantissa value will be in the range $1 \leq M < 2 - 2^{-23}$

IEEE 754-2008

- Out of the 256 possible values for the exponent, two have special meanings to deal with zero value, $\pm\infty$, and undefined results (NaN)

	$F = 0$	$F \neq 0$
$E = 0$	0	Denormalized
$E = 255$	$\pm\infty$	NaN

- The denormalized numbers are used to extend the dynamic range as the hidden one otherwise limits the smallest positive number to $2^{1-127} = 2^{-126}$
- A denormalized number has a value of

$$X = (-1)^s\, 0.F\, 2^{-126}. \tag{3}$$

- Using denormalized numbers it is possible to represent $2^{-23}2^{-126} = 2^{-149}$

IEEE 754-2008

- There is an extended format defined that is used for intermediate results in certain complex functions
- The extended binary32 format uses at least 11 bits for the exponent and at least 32 bits for the significand (now without a hidden bit)
- The other formats are defined similarly to binary32 as

Property	binary32	binary64	binary128
Total bits	32	64	128
Mantissa bits	$23 + 1$	$52 + 1$	$112 + 1$
Exponent bits	8	11	15
Bias	127	1023	16383
Ext. mantissa bits	≥ 32	≥ 64	≥ 128
Ext. exponent bits	11	15	20

Floating-point special values and exceptions

- In the IEEE 754-2008 four different special values are defined: 0, NaN, $\pm\infty$
- In addition exception flags for NaN, exponent over-/underflow, and division by zero are defined
- These will occur under the following circumstances
  - The result is 0 if
    - An addition/subtraction yield that result
    - Multiplication by 0, dividing 0 with a non-zero value etc
    - Rounding causes the exponent to be smaller than the smallest possible number, will also raise the underflow flag
  - The result is $\pm\infty$ if
    - Rounding causes the exponent to be larger than the largest possible number, will also raise the overflow flag
  - The result is NaN if
    - Some operations, e.g., $0 - \infty$

Floating-point special values and exceptions

- Note that the mathematically defined operations should work

$$X + \infty = \infty \tag{4}$$

$$\frac{X}{\infty} = 0 \tag{5}$$

- Also for NaN

$$NaN + X = NaN \tag{6}$$

etc

Floating-point rounding

- To round a value is to map a number X_e to a representable number X
- Let $X = R_{\text{mode}}(X_e)$ denote the result of rounding X_e using rounding mode "mode"
- Basic relations
  - If $X_e \leq Y_e$ then $X \leq Y$
  - If X_e is a representable number $X = X_e$
  - If $\check{X}$ and $\hat{X}$ are two consecutive representable numbers with $\check{X} \leq X_e \leq \hat{X}$ then X is either $\check{X}$ or $\hat{X}$

Floating-point rounding

- There are typically four different rounding modes used in practice:
  - Round to nearest, tie to even: R_N

$$R_N(X_e) = \begin{cases} \check{X}, & |X_e - \check{X}| < |X_e - \hat{X}| \\ \hat{X}, & |X_e - \check{X}| > |X_e - \hat{X}| \\ \text{even}\{\check{X}, \hat{X}\} & |X_e - \check{X}| = |X_e - \hat{X}| \end{cases} \tag{7}$$

  - Round toward zero (magnitude truncation): R_Z

$$R_Z(X_e) = \begin{cases} \check{X}, & X_e \geq 0 \\ \hat{X}, & X_e < 0 \end{cases} \tag{8}$$

  - Round toward plus infinity: $R_{+\infty}$

$$R_{+\infty}(X_e) = \hat{X} \tag{9}$$

  - Round toward minus infinity: $R_{-\infty}$

$$R_{-\infty}(X_e) = \check{X} \tag{10}$$

Floating-point rounding

- Assume that we have a normalized result after an operation with m fractional bits and we want to round to f fractional bits
- How many bits are required?
  - R_Z: f fractional bits
  - R_N: $f + 1$ fractional bits
  - $R_{+\infty}$: must detect when all discarded bits are zero
  - $R_{-\infty}$: must detect when all discarded bits are zero
- Introduce a <u>sticky bit</u>, denoted T, which is one if any of the discarded bits are one
- Denote the extra bit required for rounding, R

Floating-point addition/subtraction

- To add/subtract two floating-point numbers the exponents must be the same
- Required to align the two significands before adding/subtracting
- Sign-magnitude representation, so one need to figure out if the effective operation is an addition or subtraction
- Should normalize and round the result

Floating-point addition/subtraction

- Adding two numbers can lead to two cases
  - The result is already normalized

```
  1.000110
+ 0.001101
----------
  1.100011
```

  - An overflow is obtained

```
   1.000110
 + 1.111101
 ----------
  11.000011
```

  which can easily be normalized by a right-shift

```
   11.000011
>>  1.1000011
```

Floating-point addition/subtraction

- Subtracting two numbers can lead to three cases
  - The result is already normalized
  - The difference in exponent is more than one
    - The smaller term has more than one leading zero
    - The result is either normalized or has one leading zero
    - In the case of one leading zero, left shift, so a third additional bit, the guard bit G must be kept
  - The difference in exponent is zero or one
    - Cancellation may occur

```
  1.000110
- 0.1111011
----------
  0.0010011
```

    - Left-shift multiple positions

```
     0.0010011
<<   1.001100
```

Floating-point rounding

- Three additional least significant bits required before post-addition/subtraction alignment: guard bit, G, round bit, R, sticky bit, T

```
CMM       LGRT
xxx.xxxxxxxxxx
    -- f --
```

- Depending on the amount of pre-addition/subtraction alignment
  - Zero or one bit: The possibly shifted bit is in G and no precision is lost
  - Two or more bits: Non-shifted significand $1 \leq X_1 < 2$, shifted significand $0 \leq X_2 < \frac{1}{2}$, result $\frac{1}{2} < X_1 \pm X_2 < \frac{5}{2}$
- If left-shift post-subtraction alignment happens
  - $R = 0$ means discarded part is $<$ ulp/2
  - $R = 1$ means discarded part is $\geq$ ulp/2
- Problems solved with the sticky bit, T
  - R_∞: with $G = 0,\ R = 0$
  - R_E: to determine if the discarded part is exactly ulp/2