Computer Architecture

Oprițoiu Flavius
flavius.opritoiu@cs.upt.ro

November 21, 2023
November 28, 2023

Chap. 3 Functional Analysis and Synthesis of Floating Point Arithmetic Units

3.1 - Floating point operations and architectures

In general, IEEE 754 floating point (FP) operands are considered, unless otherwise stated.
IEEE 754 numbers can be either:

packed (or normalized) :	for storage / data transmission
unpacked :	used during computation

Let $X = X_M 2^{X_E}$ and $Y = Y_M 2^{Y_E}$. The four fundamental arithmetic operations on X and Y are define bellow:

- $X + Y = (X_M + Y_M 2^{Y_E - X_E}) 2^{X_E}$, if $X_E \geq Y_E$
- $X - Y = (X_M - Y_M 2^{Y_E - X_E}) 2^{X_E}$, if $X_E \geq Y_E$
- $XY = X_M Y_M 2^{X_E + Y_E}$
- $\dfrac{X}{Y} = \dfrac{X_M}{Y_M} 2^{X_E - Y_E}$

3.1 - FP operations and architectures (contd.)

Based on the definitions of the fundamental FP arithmetic operations, a FP arithmetic unit has 2 subunits:

- exponent computation: performing $+$ or $-$ of exponents
- significand computation: performing

$$\begin{tikzcd} & +,\ -,\ *\text{ or } / \quad \text{of significands} \\ {} \arrow[ur] \arrow[dr] & \\ & *2^{Y_E-X_E} \quad \text{for significand alignment} \end{tikzcd}$$

The two subunits operates only with fixed-point operands:

- exponent subunit operates with (biased) fixed-point integers
- significand subunit operates with fixed-point fractionals

3.2 - Rounding

Rounding: converting a higher precision representation to a lower precision representation (for storage or transmission).
IEEE 754 rounding modes:

$$\begin{tikzcd} & \text{towards 0} \\ & \text{to } +\infty \\ {} \arrow[uur] \arrow[ur] \arrow[r] \arrow[dr] & \text{to } -\infty \\ & \text{to nearest even} \end{tikzcd}$$

In IEEE 754, rounding concerns fractional bits with weights smaller than the weight of the least significant bit of the significand. For brevity, in this paragraph only, the rounding converts a number with integer and fractional parts into an integer.

3.2 - Rounding (contd.)

Consider X with:

$$X = x_{n-1}x_{n-2}\ldots x_1x_0.x_{-1}x_{-2}\ldots x_{-m}$$

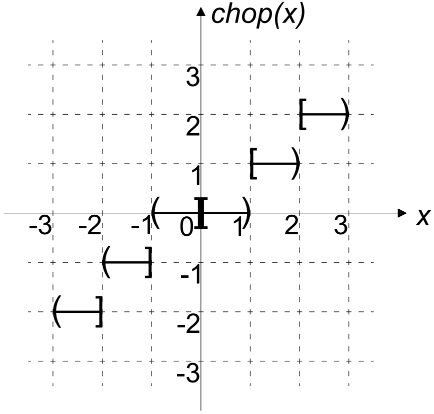
Let $X^\star$ represent the rounded value of X, with $X^\star$ being an integer:

$$X^\star = x^\star_{n-1}x^\star_{n-2}\ldots x^\star_1x^\star_0$$

3.2 - Rounding (contd.)

Ⓐ Round towards 0 (inward rounding)

$X^\star$ is the largest integer for which $|X^\star| \leq |X|$

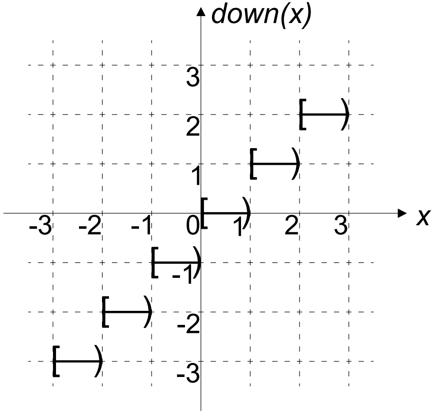


If X is in SM, inward rounding is equivalent to truncating to the integer part

3.2 - Rounding (contd.)

Ⓑ Round to $-\infty$ (downward rounding)

$X^\star$ is the largest integer for which $X^\star \leq X$



If X is positive, downward rounding is equivalent to inward rounding.

3.2 - Rounding (contd.)

$\textcircled{B}$ Round to $-\infty$ (downward rounding)
If X is in C2, downward rounding is equivalent to truncating to the integer part.

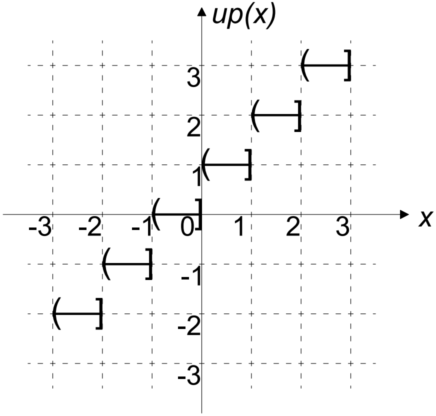
For X in SM, downward rounding is equivalent to:

$\nearrow$ truncate to integer part, if $X \geq 0$

$$\searrow X^\star = \begin{cases} x_{n-1}x_{n-2}\ldots x_1x_0 - 1, & \text{if } .x_{-1}x_{-2}\ldots x_{-m} \neq 0 \\ x_{n-1}x_{n-2}\ldots x_1x_0, & \text{if } .x_{-1}x_{-2}\ldots x_{-m} = 0 \end{cases} \text{, if } X < 0$$

3.2 - Rounding (contd.)

Ⓒ Round to $+\infty$ (upward rounding)
$X^\star$ is the smallest integer for which $X^\star \geq X$



If X is negative, upward rounding is equivalent to inward rounding.

3.2 - Rounding (contd.)

Upward and downward rounding characteristics:

- errors are in the same dirrection $\Rightarrow$ errors accumulates faster
- provide some upper/lower bounds for a result $\Rightarrow$ interval arithmetic

3.2 - Rounding (contd.)

D' Round to nearest

IEEE 754's round to nearest even rounding mode is derived from the *round to nearest* mode

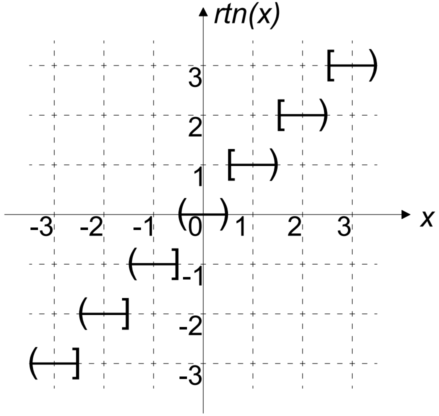
Without loosing generality, consider X to be positive. $X^\star$ rounded to nearest is defined as:

$$X^\star = \begin{cases} x_{n-1}x_{n-2}\dots x_1x_0, & \text{if } .x_{-1}x_{-2}\dots x_{-m} < \frac{1}{2} \\ x_{n-1}x_{n-2}\dots x_1x_0 + 1, & \text{if } .x_{-1}x_{-2}\dots x_{-m} \geq \frac{1}{2} \end{cases}$$

In a similar manner can be defined the round to nearest mode for negatives.

3.2 - Rounding (contd.)

D' Round to nearest



3.2 - Rounding (contd.)

D' Round to nearest

Error accumulation analysis: consider X to be positive and to have only 2 fractional bits.

Inputs		Outputs	
x_{-1}	x_{-2}	$X^\star = rtn(X)$	$\epsilon = X^\star - X$
0	0	$x_{n-1}x_{n-2}\ldots x_1x_0$	0
0	1	$x_{n-1}x_{n-2}\ldots x_1x_0$	$-\frac{1}{4}$
1	0	$x_{n-1}x_{n-2}\ldots x_1x_0 + 1$	$\frac{1}{2}$
1	1	$x_{n-1}x_{n-2}\ldots x_1x_0 + 1$	$\frac{1}{4}$

If all 4 above cases are equally probable to appear during a sequence of computations, the mean error is obtained as

$$\epsilon_{mean} = \frac{0 - \frac{1}{4} + \frac{1}{2} + \frac{1}{4}}{4} = \frac{1}{8}$$

If line 3 of the table above is more probable to appear, compared to the other, ϵ_{mean} can become larger than $\frac{1}{8}$

3.2 - Rounding (contd.)

$\text{\textcircled{D'}}$ Round to nearest

Solution to the error accumulation problem: split the case $.x_{-1}x_{-2} = .10$ into two sub-cases, with equal probability (or as close to equal as possible) so that one sub-case rounds upwards and the other downwards.

One possible approach to splitting the rounding of a fractional part of $\frac{1}{2}$ with equal probabilities would be to inspect the least significant bit of the integer part, x_0, thus differentiating between even and odd numbers. For positive numbers, if $x_0 = 0$ (even numbers) the rounding could be done downwards while for $x_0 = 1$ (odd numbers), rounding would be upwards.

3.2 - Rounding (contd.)

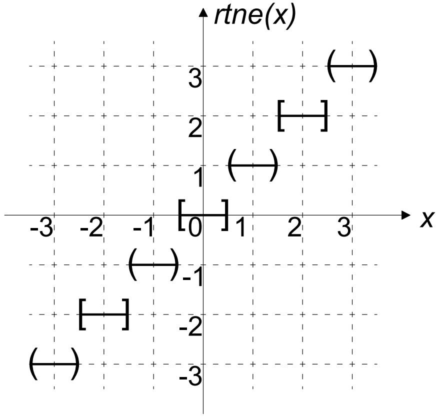
Ⓓ Round to nearest even

Without loosing generality, consider X to be positive. $X^\star$ rounded to nearest even is defined as:

$$X^\star = \begin{cases} x_{n-1}x_{n-2}\ldots x_1x_0, & \text{if } .x_{-1}x_{-2}\ldots x_{-m} < \frac{1}{2}, \text{ OR} \\ & \text{if } .x_{-1}x_{-2}\ldots x_{-m} = \frac{1}{2} \text{ AND } x_0 = 0 \\ & \\ x_{n-1}x_{n-2}\ldots x_1x_0 + 1, & \text{if } .x_{-1}x_{-2}\ldots x_{-m} > \frac{1}{2}, \text{ OR} \\ & \text{if } .x_{-1}x_{-2}\ldots x_{-m} = \frac{1}{2} \text{ AND } x_0 = 1 \end{cases}$$

3.2 - Rounding (contd.)

Ⓓ Round to nearest even



3.3 Normalization and rounding rules for a FP result

Consider two significands X_M and Y_M, on m bits:

$$X_M = 1.x_{m-2}x_{m-3}\ldots x_i \ldots x_1x_0$$
$$Y_M = 1.y_{m-2}y_{m-3}\ldots y_i \ldots y_1y_0$$

Consider also that $X_E \geq Y_E$ so that in order to add the two significands, Y_M need to be aligned by right-shifting with $d = |X_E - Y_E|$ positions.

Y_M's alignment can be done:

- with infinite precision: keep all bits of Y_M including bits with weights smaller than 2^{-m+1}
- with finite precision: keep only 3 bits out of all positions having weights smaller than 2^{-m+1}

3.3 Normalization and rounding rules for a FP result (contd.)

The 3 preserved bits during Y_M's alignment with finite precision are called *sticky bits*:

- g: the guard bit, with weight 2^{-m}; it guards against loss of precision
- r: the round bit, with weight 2^{-m-1}; it is used for result rounding
- s: the sticky bit, with weight 2^{-m-2}; it is obtained as a logic *OR* of all other less significant bits that were right-shifted out of Y_M, except the g and r bits

After alignment, Y_M becomes Y_{Mal}.

3.3 Normalization and rounding rules for a FP result (contd.)

Consider the FP result to be a sum $\Rightarrow Z_M = X_M + Y_{Mal}$, provided that $X_E \geq Y_E$.

Since the significands addition can produce a carry out, it follows that the form of Z_M is:

$$Z_M = z_m \; z_{m-1} \; . \; z_{m-2} \; z_{m-3} \ldots z_1 \; z_0 \mid g \;\; r \;\; s$$

The normalization operation for Z_M will produce Z_{M_n}, with

$$Z_{M_n} = 1 \; . \; z_{m-2_n} \; z_{m-3_n} \ldots z_{1_n} \; z_{0_n} \mid R \;\; S$$

The two bits, R and S are needed for performing the rounding operations, subsequent to normalization.

3.3 Normalization and rounding rules for a FP result (contd.)

Normalization cases:

$Z_{M_n} =$		1.	z_{m-2_n}	z_{m-3_n}	$\dots$	z_{1_n}	z_{0_n}	$\mid$	R	S
Case 1)	$z_m = 1$	1.	z_{m-1}	z_{m-2}	$\dots$	z_2	z_1	$\mid$	z_0	$(g \underline{OR} r \underline{OR} s)$
$\Rightarrow$ 1-bit right-shift, Z_E++										
Case 2)	$z_m = 0$, $z_{m-1} = 1$	1.	z_{m-2}	z_{m-3}	$\dots$	z_1	z_0	$\mid$	g	$(r \underline{OR} s)$
$\Rightarrow Z_M$ is already normalized										
Case 3)	$z_m = 0$, $z_{m-1} = 0$, $z_{m-2} = 1$	1.	z_{m-3}	z_{m-4}	$\dots$	z_0	g	$\mid$	r	s
$\Rightarrow$ 1-bit left-shift, Z_E--										
Case 4)	$z_m = 0$, $z_{m-1} = 0$, $z_{m-2} = 0$, $z_{m-3} = 1$	1.	z_{m-4}	z_{m-5}	$\dots$	g	0	$\mid$	0	0
$\Rightarrow$ 2-bit left-shift, $Z_E -= 2$										

3.3 Normalization and rounding rules for a FP result (contd.)

If normalization of Z_M requires a left shift operation of 2 or more bit:

- append the g bit to Z_M, after z_0
- complete all remaining bits of Z_M with 0s
- set $R = S = 0$

Rounding of Z_{M_n} uses bits R and S previously determined

- make use of 2 bits to be eliminated in order to implement all 4 rounding modes

3.3 Normalization and rounding rules for a FP result (contd.)

Rounding rules

Rounding mode	$Z_{M_n} > 0$	$Z_{M_n} < 0$
towards 0	——— (discard R and S)	———
to $-\infty$	———	$if\,(R\ \underline{or}\ S)\ then\ Z_{M_n} - 1$
to $+\infty$	$if\,(R\ \underline{or}\ S)\ then\ Z_{M_n} + 1$	———
to nearest even	$if\,(R\ \underline{and}(S\ \underline{or}\ z_{0_n}))\ then\ Z_{M_n} + 1$	$if\,(R\ \underline{and}(S\ \underline{or}\ z_{0_n}))\ then\ Z_{M_n} - 1$

3.4 FP addition/subtraction with rounding

Rounding error is not correlated with exponents' difference (see Fig. 5.12 in [Vlad12]).

A simplified, scaled down, FP format will be used:

- inspired by IEEE 754, with narrower fields
  - 1 bit for sign
  - 3 bits for exponent field ($e = 3$)
  - 3 bits for fractional part of significand
- same bias calculation relation as IEEE 754
  $bias = 2^{e-1} - 1 = 3$
- same exceptions as IEEE 754

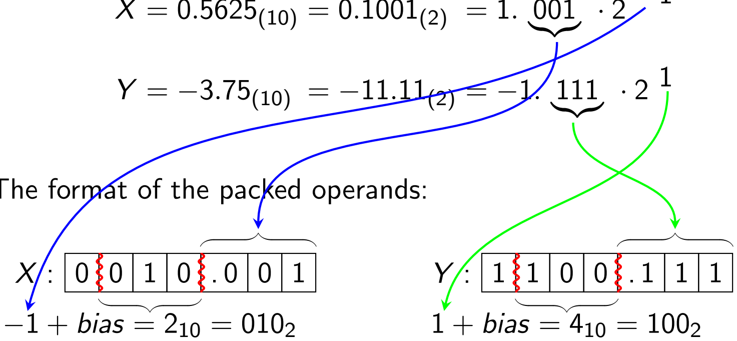
3.4 FP addition/subtraction with rounding (contd.)

Consider the following operands:

$$X = 0.5625_{(10)} = 0.1001_{(2)} = 1.\underbrace{001} \cdot 2^{-1}$$

$$Y = -3.75_{(10)} = -11.11_{(2)} = -1.\underbrace{111} \cdot 2^{1}$$

The format of the packed operands:



3.4 FP addition/subtraction with rounding (contd.)

Floating point addition with rounding algorithm:

Step $\textcircled{1}$: Unpack operands

- add the hidden bit
- check for exceptions: one of the operands is 0, $\pm\infty$, *NaN*

X :

0	0	1	0	1	.0	0	1

Y :

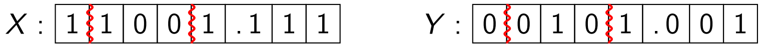
1	1	0	0	1	.1	1	1

Step $\textcircled{2}$: Calculate exponents difference, $d = X_E - Y_E$

- if $d < 0$ it follows that $|X| < |Y|$
  - Swap the two operands
  - set result's exponent, $Z_E = Y_E$
- if $d \geq 0$
  - set result's exponent, $Z_E = X_E$

Operands swapping reduce device's area (provides only Y with right-shift alignment).

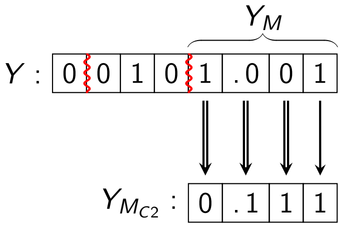
Calculate $d = X_E - Y_E = -2$. Because d is negative $\Rightarrow$ SWAP the operands and set $Z_E = Y_E = 4$.



3.4 FP addition/subtraction with rounding (contd.)

Step $\textcircled{3}$: If $sign(X) \neq sign(Y) \Rightarrow$ two's complement Y_M

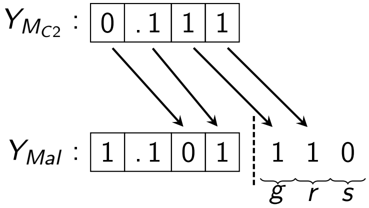
- differing signs indicates subtraction
  - two's complementing allows using binary adders for significands subtraction
- only two's complement significand Y_M (reduce device's area by avoiding two's complementation for X_M)



3.4 FP addition/subtraction with rounding (contd.)

Step $\textcircled{4}$: Align Y_M by shifting it to the right with $|d|$ positions, aligned Y_M is referred to as $Y_{M}al$

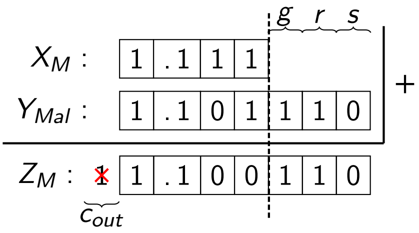
- if in Step 3 Y_M was two's complemented, when right-shifting, introduce bits of 1s in the most significant positions of Y_M, instead of 0s
- preserve the sticky bits: g, r and s



3.4 FP addition/subtraction with rounding (contd.)

Step ⑤: Add the two significands: $Z_M = X_M + Y_{Mal}$

- if $sign(X) = sign(Y)$ the potential carry out is preserved (it is part of the result)
- if $sign(X) \neq sign(Y)$ and no carry out is generated, Z_M is negative and it must be two's complemented
- if $sign(X) \neq sign(Y)$ and a carry out is generated, Z_M is positive and the carry out is discarded



3.4 FP addition/subtraction with rounding (contd.)

Step $\textcircled{6}$: Prenormalization

- according to the normalization cases from section 3.3
  - determine Z_{M_n}
  - can update Z_E
- check for exceptions:
  - if $Z_E = Z_{E_{MAX}}(2^e - 2 = 6)$ and Z_M requires a 1-bit right shift $\Rightarrow$ *Overflow*
  - if $Z_E = Z_{E_{min}}(1)$ and Z_M requires 1-bit (or more) left shift $\Rightarrow$ *Underflow*

Z_M is already normalized (normalization case 2 from 3.3)

$Z_M : \boxed{1}\boxed{.1}\boxed{0}\boxed{0}\,\vdots\,\boxed{1}\boxed{1}\boxed{0}$ (from Step 5)

Z_M is already normalized

$Z_{M_n} : \boxed{1}\boxed{.1}\boxed{0}\underbrace{\boxed{0}}_{z_{0_n}}$

$Z_E = 4$ (from Step 2)

3.4 FP addition/subtraction with rounding (contd.)

Step ⑦: Determine values of R and S (according to the same normalization cases from section 3.3)

$$Z_M : \boxed{1}\boxed{.1}\boxed{0}\boxed{0}\vdots\overbrace{\boxed{1}}^{g}\overbrace{\boxed{1}}^{r}\overbrace{\boxed{0}}^{s} \text{ (from Step 5)}$$

Z_M is already normalized

$R = g = 1$

$S = (r \ \underline{or} \ s) = 1$

3.4 FP addition/subtraction with rounding (contd.)

Step ⑧: Round Z_{M_n} to obtain $Z_M^\star$:

- according to the rounding rules of section 3.3
- if rounding generates carry out
  - postnormalize result
    - shift $Z_M^\star$ to right by 1 bit
    - increment Z_E
      - check for *Overflow* exceptions

Consider rounding to nearest even mode. According to rules from 3.3 , condition R <u>*and*</u> $(S$ <u>*or*</u> $z_{0_n})$ is True $\Rightarrow$ increment $Z_M^\star$.

$$\begin{array}{rr|l}
Z_{M_n}: & \boxed{1}\boxed{.1}\boxed{0}\boxed{0} & \\
 & 1 & + \\
\hline
Z_M^\star: \underbrace{0}_{c_{out}} & \boxed{1}\boxed{.1}\boxed{0}\boxed{1} &
\end{array}$$

$Z_E = 4$ (from Step 6)

3.4 FP addition/subtraction with rounding (contd.)

Step $\textcircled{9}$: Determine sign of the result:

- if $sign(X) = sign(Y) \Rightarrow sign(Z) = sign(X)$
- if $sign(X) \neq sign(Y)$, the sign is obtained from the table below

Inputs				Output
SWAP (in Step 2)	Two's complement (in Step 5)	$sign(X)$	$sign(Y)$	$sign(Z)$
YES		+	-	−
YES		-	+	+
NO	YES	+	-	−
NO	YES	-	+	+
NO	NO	+	-	+
NO	NO	-	+	−

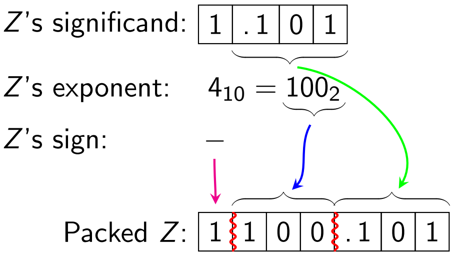
Important: The $sign(X)$ and $sign(Y)$ columns above refer to the sign of the operands prior to the potential SWAP in Step 2.

For the considered example, because of the SWAP in Step 2, according to table's first line $\Rightarrow sign(Z) = -$

3.4 FP addition/subtraction with rounding (contd.)

Step ⑩: Pack the result

Use the FP fields of result Z, determined in the previous steps:



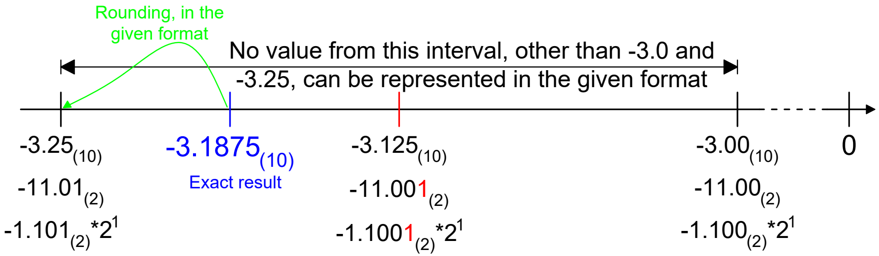
Z's significand: $\boxed{1}\boxed{.1}\boxed{0}\boxed{1}$
Z's exponent: $4_{10} = 100_2$
Z's sign: $-$
Packed Z: $\boxed{1}\boxed{1}\boxed{0}\boxed{0}\boxed{.1}\boxed{0}\boxed{1}$

3.4 FP addition/subtraction with rounding (contd.)

Verification: The exact result is:
$X + Y = 0.5625 - 3.75 = -3.1875.$

The value of Z:
$Z = (-1)^{Z_{sign}} * 2^{Z_E - bias} * Z_M = -1 * 2^{4-3} * 1.101_2 = -3.25$

In the $[-3.25, -3.00]$ interval, only the margins have valid representations in the given format. Considering the position of the exact result relative to the middle of the interval, -3.125, Z is rounded to -3.25.



Rounding, in the given format
No value from this interval, other than -3.0 and -3.25, can be represented in the given format
-3.25(10)
-11.01(2)
-1.101(2)*2^1
-3.1875(10)
Exact result
-3.125(10)
-11.001(2)
-1.1001(2)*2^1
-3.00(10)
-11.00(2)
-1.100(2)*2^1
0

3.4 FP addition/subtraction with rounding (contd.)

Design of the pre-normalization shifter:

- covers operations in Step 6 and Step 7
- designed according to the rules in section 3.3
- purely combinational design

The result of Step 5, for the considered format is:

$$Z_M = z_{4_n} z_{3_n} \ . \ z_{2_n} \ z_{1_n} z_{0_n} \mid g \ r \ s$$

Output of Step 6 combined with Step 7 is:

$$Z_{M_n} = 1 \ . \ z_{2_n} \ z_{1_n} z_{0_n} \mid R \ S$$

3.4 FP addition/subtraction with rounding (contd.)

According to the normalization rules in section 3.3, for the given format, it follows that Z_M must be either:

- shifted to the right with 1 bit (r_1), or
- left unchanged since it is normalized (l/r_0), or
- shifted to the left with 1 bit (l_1), or
- shifted to the left with 2 bits (l_3), or
- shifted to the left with 3 bits (l_3)

The 5 normalization cases are identified by the 5 conditions/variables in parentheses: r_1, l/r_0, l_1, l_2, l_3.

One and only one of the 5 conditions must be active for a given pair of FP operands to be added by the FP addition algorithm with rounding.

3.4 FP addition/subtraction with rounding (contd.)

The normalization cases with the associated conditions are presented below:

Z_{M_n}	1.	z_{2_n}	z_{1_n}	z_{0_n}	R	S
Z_M is normalized (l/r_0)	1.	z_2	z_1	z_0	g	$(r\ \underline{OR}\ s)$
Z_M needs 1-bit left-shift (l_1)	1.	z_1	z_0	g	r	s
Z_M needs 2-bit left-shift (l_2)	1.	z_0	g	0	0	0
Z_M needs 3-bit left-shift (l_3)	1.	g	0	0	0	0
Z_M needs 1-bit left-shift (r_1)	1.	z_{m-1}	z_{m-2}	z_1	z_0	$(g\ \underline{OR}\ r\ \underline{OR}\ s)$

3.4 FP addition/subtraction with rounding (contd.)

Consequently, one can write boolean equations for the 5 bits generated at the output of Step 6 combined with Step 7:

$$\begin{array}{lcllllll}
z_{2_n} & = & z_2 & \cdot l/r_0 + z_1 & \cdot l_1 + z_0 & \cdot l_2 + g & \cdot l_3 & + z_3 & \cdot r_1 \\
z_{1_n} & = & z_1 & \cdot l/r_0 + z_0 & \cdot l_1 + g & \cdot l_2 & & + z_3 & \cdot r_1 \\
z_{0_n} & = & z_0 & \cdot l/r_0 + g & \cdot l_1 & & & + z_3 & \cdot r_1 \\
R & = & g & \cdot l/r_0 + r & \cdot l_1 & & & + z_0 & \cdot r_1 \\
S & = & (r \ \underline{OR} \ s) & \cdot l/r_0 + s & \cdot l_1 & & & + (g \ \underline{OR} \ r \ \underline{OR} \ s) & \cdot r_1
\end{array}$$

3.4 FP addition/subtraction with rounding (contd.)

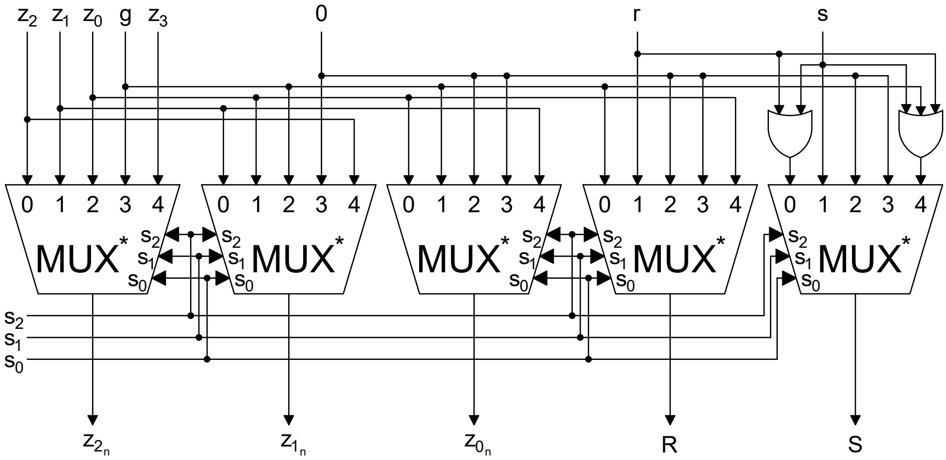
Because for a given pair of FP operands added by the FP addition algorithm with rounding, one and only one of the 5 conditions can be active, $\Rightarrow$ the 5 conditions can be encoded on fewer bits.

Consider variables s_2, s_1 and s_0 for encoding the 5 variables. The encoding is described in the table below:

Inputs					Outputs		
r_1	l_3	l_2	l_1	l/r_0	s_2	s_1	s_0
0	0	0	0	1	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	1	0	0	0	0	1	1
1	0	0	0	0	1	0	0

3.4 FP addition/subtraction with rounding (contd.)

The architecture of the pre-normalization shifter is depicted below:



The *MUX** are degenerate multiplexors, with 3 selection lines but having only 5 data inputs.

References

[Vlad12] M. Vlăduțiu, *Computer Arithmetic: Algorithms and Hardware Implementations*. Springer, 2012.