1.5.3. Carry Dependent Sum Adder (CDSA)

→ creșe un "descalibrare" pornind de la biți incorecți

→ dacă c_{i+1} = incorect ⇒ z_i devine incorect

inputs x_i	y_i	c_i	c_{i+1}	outputs z_i
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

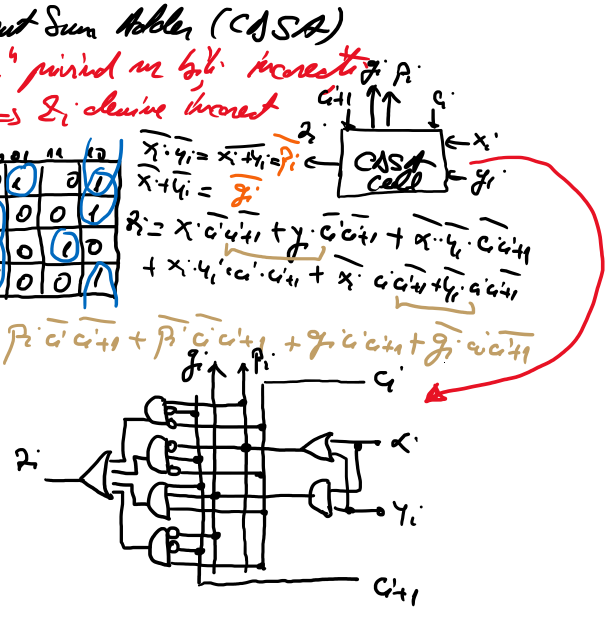
z_i	$x_i y_i$ = 00	01	11	10
c_{i+1} = 00	0	1	0	1
01	1	0	0	1
11	0	0	1	0
10	1	0	0	1

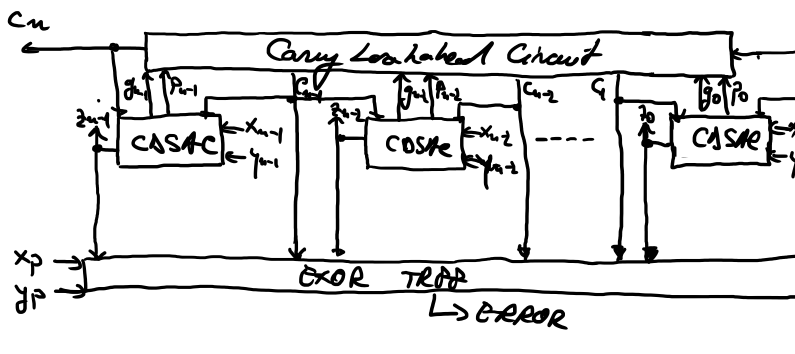
$\overline{x_i \cdot y_i} = \overline{x_i + y_i} = p_i$

$\overline{x_i + y_i} = g_i$

$z_i = x_i \cdot \overline{c_i} \cdot \overline{c_{i+1}} + y_i \cdot \overline{c_i} \cdot \overline{c_{i+1}} + \overline{x_i} \cdot \overline{y_i} \cdot \overline{c_{i+1}} + x_i \cdot y_i \cdot c_i \cdot c_{i+1} + \overline{x_i} \cdot c_i \cdot c_{i+1} + \overline{y_i} \cdot c_i \cdot c_{i+1}$

$z_i = p_i \cdot \overline{c_i} \cdot \overline{c_{i+1}} + \overline{p_i} \cdot \overline{c_i} \cdot c_{i+1} + g_i \cdot c_i \cdot c_{i+1} + \overline{g_i} \cdot c_i \cdot \overline{c_{i+1}}$





Chapter II Functional Analysis & Synthesis of Floating-Point Arithmetic Units.

2.1. F.p. arithmetic operations

IEEE 754 ↔ packed: normalized (no hidden bit), for storage, transmission
↔ unpacked: {includes hidden bit, significant unnormalized}, during computation

let $X = X_M \cdot 2^{X_E}$, $Y = Y_M \cdot 2^{Y_E}$

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

$X - Y = (X_M \ominus Y_M \cdot 2^{Y_E - X_E}) \cdot 2^{X_E}$, if $X_E \geq Y_E$

$X * Y = (X_M \circledast Y_M) * 2^{X_E + Y_E}$

$\frac{X}{Y} = \frac{X_M}{Y_M} * 2^{X_E - Y_E}$

exponent subunit: +, −

↳ significant subunit: +, −, *, /

→ operate fixed-point

$* 2^{Y_E - X_E}$ (alligment)

RightShift

2.2. Rounding: conversia unei reprezentări de precizie ridicată în o reprezentare de precizie mai scăzută

IEEE 754 → to 0 / toward −∞ / toward +∞ / to nearest even

For brevity, let X, – fixed-point number, with integer & fractional parts

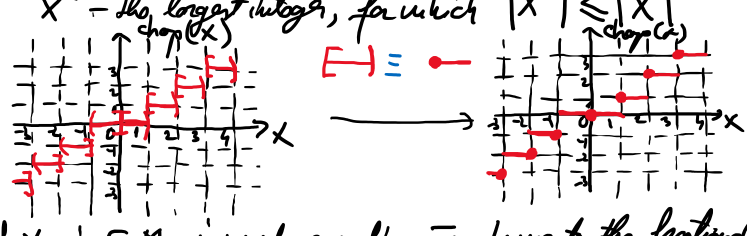
$X = \underbrace{x_{n-1} x_{n-2} \cdots x_1 x_0}_{\text{integer part}} \bullet \underbrace{x_{-1} x_{-2} \cdots x_{-m}}_{\text{fractional part}}$

X^* = rounded value of X, integer

2.91 → 2 ; −3.17 → −3

(A) to 0 (inwards rounding)

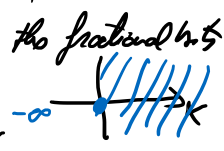
X^* – the largest integer, for which $|X^*| \leq |X|$



if X – in S.M. inwards rounding ≡ truncate the fractional bits

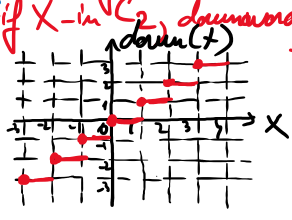
(B) towards −∞ (downwards rounding)

X^* – largest integer, for which $X^* \leq X$



– if X ≥ 0, downwards round. ≡ inwards round.

– if X – in C_2, downwards rounding ≡ truncate the fractional bits



−3.5 → −4 −4 ≤ −3.5

(−1, 0) → −1 −0.9 → −1

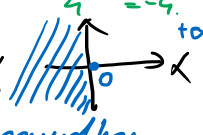
−1 → −1

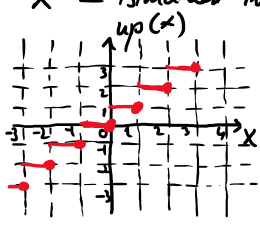
$-3.5 = 1100.1_{C2} \rightarrow 1100_{C2} = -2^3 + 100 = -8 + 4 = -4$

$+3.5 = 0011.1_{C2}$ (sign)

(C) towards +∞ (upwards rounding)

X^* – smallest integer, for which $X^* \geq X$





upwards & downwards rounding – accumulate errors in the same direction

→ 0.1 + 1.2 + 2 + 2.1 + 4.5

1 + 2 + 2 + 4 + 5 upwards

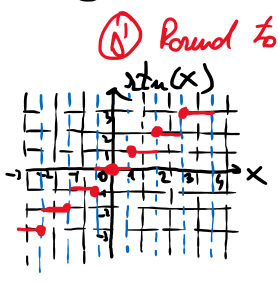
0 + 1 + 2 + 3 + 4 downwards

– upper / lower bound for result

↳ interval arithmetic

(D) to nearest even: derived from to nearest

(D') Round to nearest: if X ≥ 0, → similarly defined for X < 0



$X^* = \begin{cases} x_{n-1} x_{n-2} \cdots x_1 x_0 & \text{, if } \bullet x_{-1} x_{-2} \cdots x_{-m} < \frac{1}{2} \\ x_{n-1} x_{n-2} \cdots x_1 x_0 + 1 & \text{, if } \bullet x_{-1} x_{-2} \cdots x_{-m} \geq \frac{1}{2} \end{cases}$

ERROR analysis:

– for brevity $x_{-3} = x_{-4} = x_{-5} \cdots = x_{-m} = 0$

– only x_{-1}, x_{-2} can be ≠ 0

– a 2 bit fractional part

inputs x_{-1}	x_{-2}	X^*	$\varepsilon = X^* - X$
0	0	$x_{n-1} x_{n-2} \cdots x_1 x_0 \bullet$	0
0	1	$x_{n-1} x_{n-2} \cdots x_1 x_0 \bullet$	$-\frac{1}{4}$
1	0	$x_{n-1} x_{n-2} \cdots x_1 x_0 \bullet + 1$	$+\frac{1}{2}$ ($+\frac{1}{4}$, $-\frac{1}{4}$)
1	1	$x_{n-1} x_{n-2} \cdots x_1 x_0 \bullet + 1$	$+\frac{1}{4}$

$.00_{(2)} = .0_{(10)}$

$.01_{(2)} = .25_{(10)}$

$.10_{(2)} = .5_{(10)}$

$.11_{(2)} = .75_{(10)}$

if all 4 cases are equally probable

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

⇒ Round to nearest even: inspect x_0 of X (bit of integer part)

if $x_0 == 1$, X is odd → $x_{-1}x_{-2} = .10_{(2)}$ ↑ $X_. + 1$

if $x_0 == 0$, X is even → $x_{-1}x_{-2} = .10_{(2)}$ ↘ $X_.$

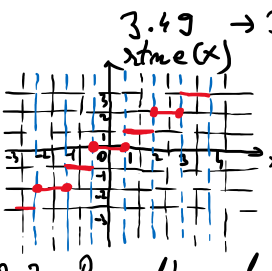
(D) Round to nearest even:

if X ≥ 0

$X^* = \begin{cases} x_{n-1} x_{n-2} \cdots x_1 x_0 \bullet & \text{, if } x_{-1} x_{-2} \cdots x_{-m} < \frac{1}{2}, \text{ OR } x_{-1} x_{-2} \cdots x_{-m} == \frac{1}{2} \text{ AND } x_0 == 0 \\ x_{n-1} x_{n-2} \cdots x_1 x_0 \bullet + 1 & \text{, if } x_{-1} x_{-2} \cdots x_{-m} > \frac{1}{2}, \text{ OR } x_{-1} x_{-2} \cdots x_{-m} == \frac{1}{2} \text{ AND } x_0 == 1 \end{cases}$

4.51 → 5

3.49 → 3



[0.5, 0.5] → 0

(0.5, 1.5) → 1

[1.5, 2.5] → 2

2.3. Rounding and normalization Rules for f.p. addition

$X_M = 1 . x_{M-2} x_{M-3} \cdots x_1 x_0$ | g r s

(2^{-1}, 2^{-2}, 2^{-M+2}, 2^{-M+1}, 2^{-M}, 2^{-M-1}, 2^{-M-2})

(M bits = storage capacity)

$Y_M = 1 . y_{M-2} y_{M-3} \cdots y_1 y_0$

and $X_E \geq Y_E$

Y_M – aligned by RightShift with d bits, $d = X_E - Y_E$

align Y_M keeping only 3 bits with weight $< 2^{-M+1}$

– g: guard bit – weight 2^{-M}; the last bit RightShifted out of Y_M

– guards against loss of precision

– r: round bit – weight 2^{-M-1}; the last but one bit RShifted out of Y_M

– s: sticky bit – weight 2^{-M-2}

– obtained by logic OR-ing of all other bits RShifted out of Y_M (excluding g, r)