AC C8

1.5.1 Binary adders with parity control

- use parity control for protecting adders against errors
  - ≡ attach 1 parity bit to operands.

$$X, Y, c_0 \rightarrow \{c_n, Z\} = X + Y + c_0$$

$$\text{inputs: } \begin{cases} x_p = x_{n-1} \oplus x_{n-2} \oplus x_{n-3} \oplus \dots \oplus x_1 \oplus x_0 \\ y_p = y_{n-1} \oplus y_{n-2} \oplus \text{———} \oplus y_1 \oplus y_0 \end{cases}$$

$$z_p = z_{n-1} \oplus z_{n-2} \oplus \dots \oplus z_1 \oplus z_0 \quad (1)$$

$$(z_i = x_i \oplus y_i \oplus c_i$$

generat

Notation $c_p = c_{n-1} \oplus c_{n-2} \oplus \dots \oplus c_1 \oplus c_0$

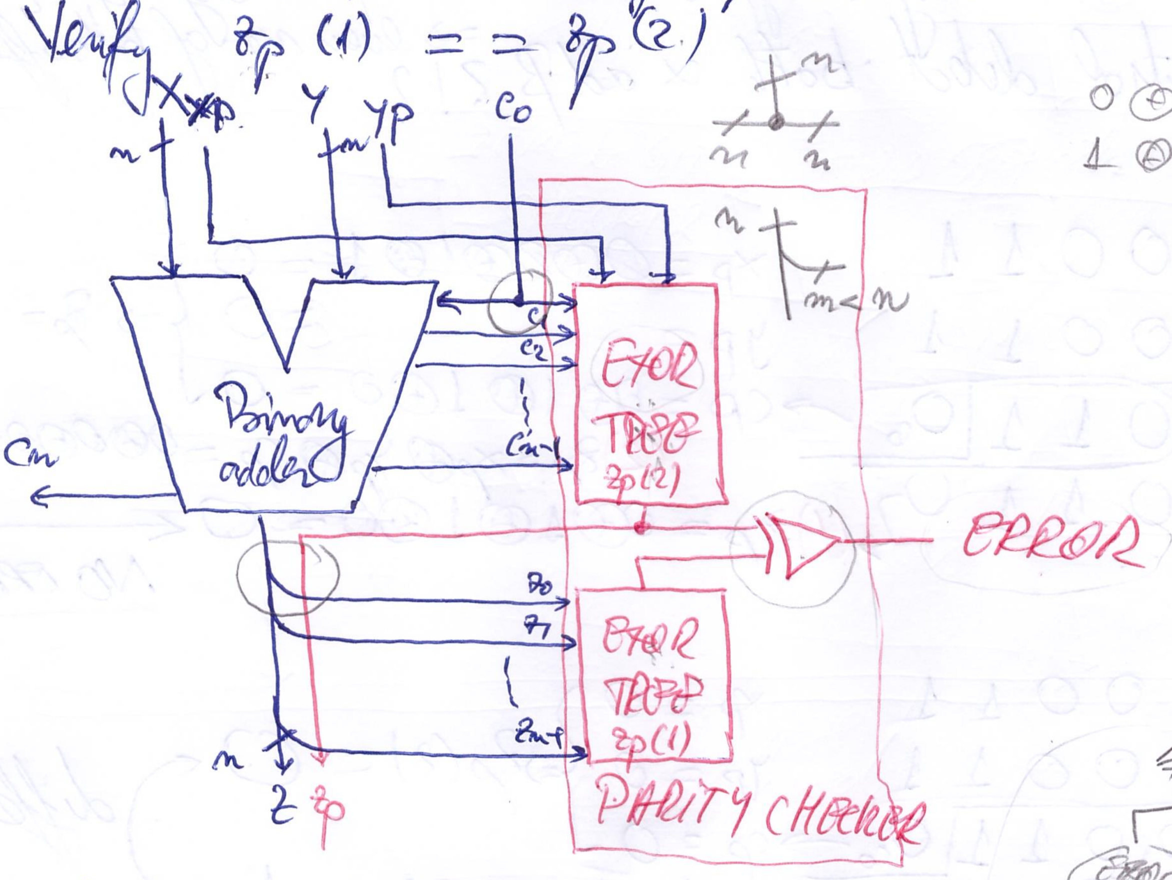
intrare

$$z_p = x_{n-1} \oplus y_{n-1} \oplus c_{n-1} \oplus x_{n-2} \oplus y_{n-2} \oplus c_{n-2} \oplus \dots x_0 \oplus y_0 \oplus c_0$$

$$z_p = x_p \oplus y_p \oplus c_p \quad (2)$$

- faster than (1)
- anticipates value of z_p (1)

Verify $z_p(1) == z_p(2)$



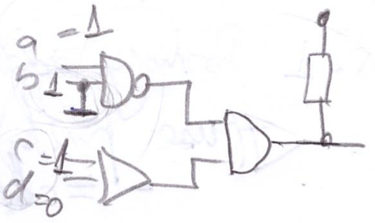
0 ⊕ 0 → 0
1 ⊕ 1 → 0

Parity checker ≡ EXOR tree

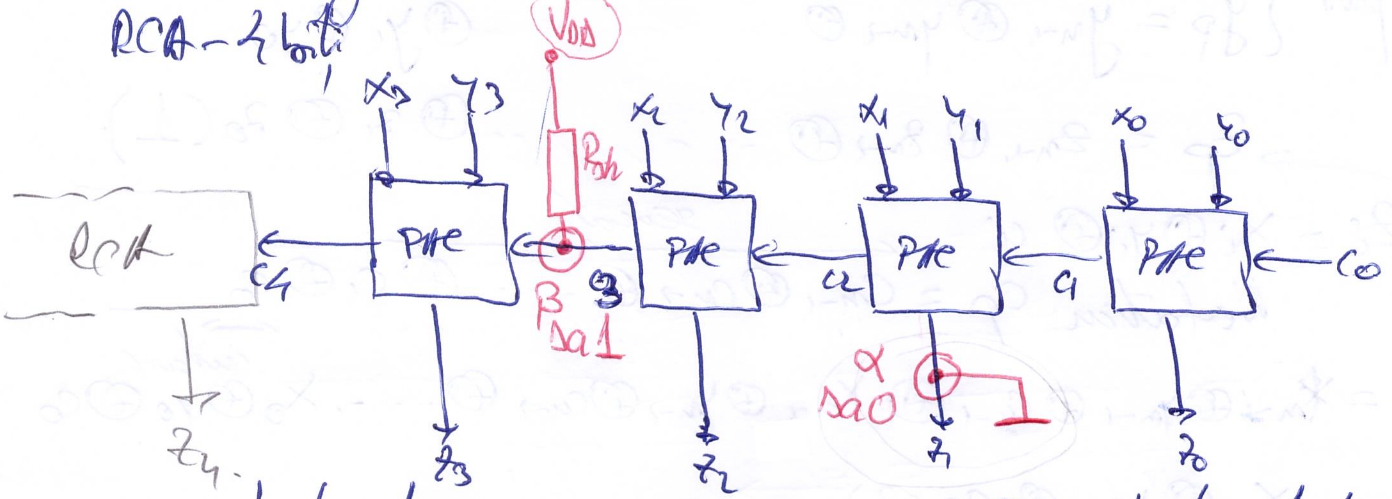
ERRORS that can affect a binary adder:

- errors
  - single: higher probability
  - multiple: harder to detect
- logic manifestation

→ Single Stuck at Fault (SSaF)



RCA - 4 bits



α: stuck-at 0
only z_1 is affected.
⇒ odd no of bits affected.

β: stuck-at 1
c_3, z_3 = affected.
- possibly c_4, z_4
⇒ even no of bits affected

Can parity control detect both α and β ?!?

fault-free

$$
\begin{array}{lllll}
X = & 0 & 0 & 1 & 1 \\
Y = & 0 & 0 & 1 & 1 \\
\hline
c = {}_{c_4}0 & 0 & 1 & 1 & 0_{c_0} \\
z = & 0 & 1 & 1 & 0
\end{array}
$$

$x_p = \cancel{0} \oplus \cancel{0} \oplus 1 \oplus 1 = 0$
$y_p = \text{---} = 0$
$c_p = 0 \oplus 1 \oplus 1 \oplus 0 = 0$

(2) $z_p = x_p \oplus y_p \oplus c_p = 0 \oplus 0 \oplus 0 = 0$

(1) $z_p = 0 \oplus 1 \oplus 1 \oplus 0 = 0$

NO ERROR

α: z_1 sa0:

$$
\begin{array}{lllll}
X = & 0 & 0 & 1 & 1 \\
Y = & 0 & 0 & 1 & 1 \\
\hline
c = {}_{c_4}0 & 0 & 1 & 1 & 0_{c_0} \\
z = & 0 & 1 & 0 & 0
\end{array}
$$

$x_p = 0$
$y_p = 0$
$c_p = 0$

$\Rightarrow z_p(2) = 0$

$z_p(1) = 0 \oplus 1 \oplus 0 \oplus 0 = 1$

differ

! ERROR

B: C_3 sa 1:

$X = 0\ 0\ 1\ 1 \qquad x_p = 0$

$Y = 0\ 0\ 1\ 1 \qquad y_p = 0 \qquad \Big\}\ z_p(2) = 1$

$C = {}_{c_4}0\ 1\ 1\ 1\ 0_{c_0} \rightarrow c_p = 1 \oplus 1 \oplus 1 \oplus 0 = 1$

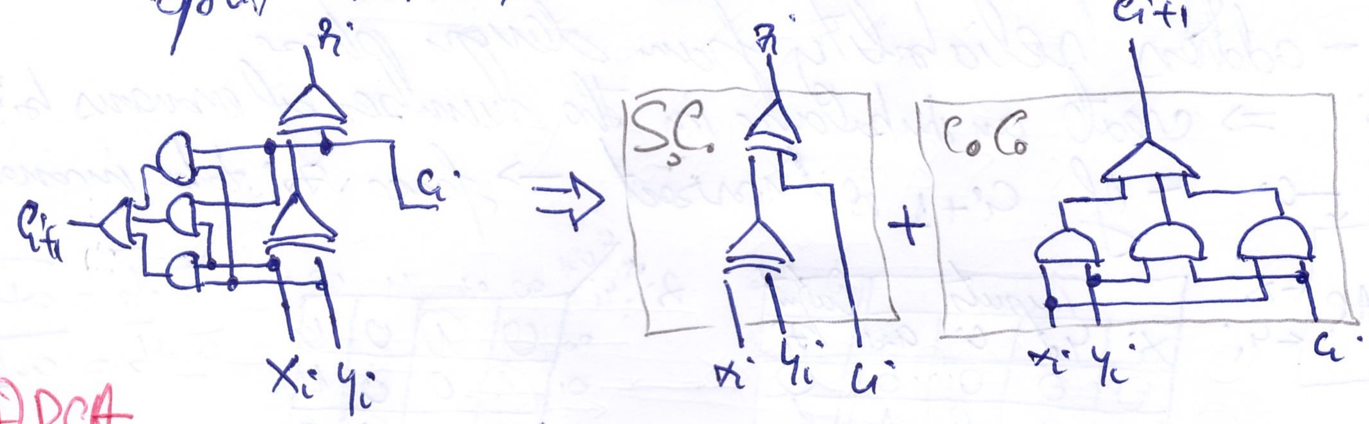
$Z = 1\ 1\ 1\ 0 \qquad z_p(1) = 1 \oplus 1 \oplus 1 \oplus 0 = 1$

NO ERROR?!?

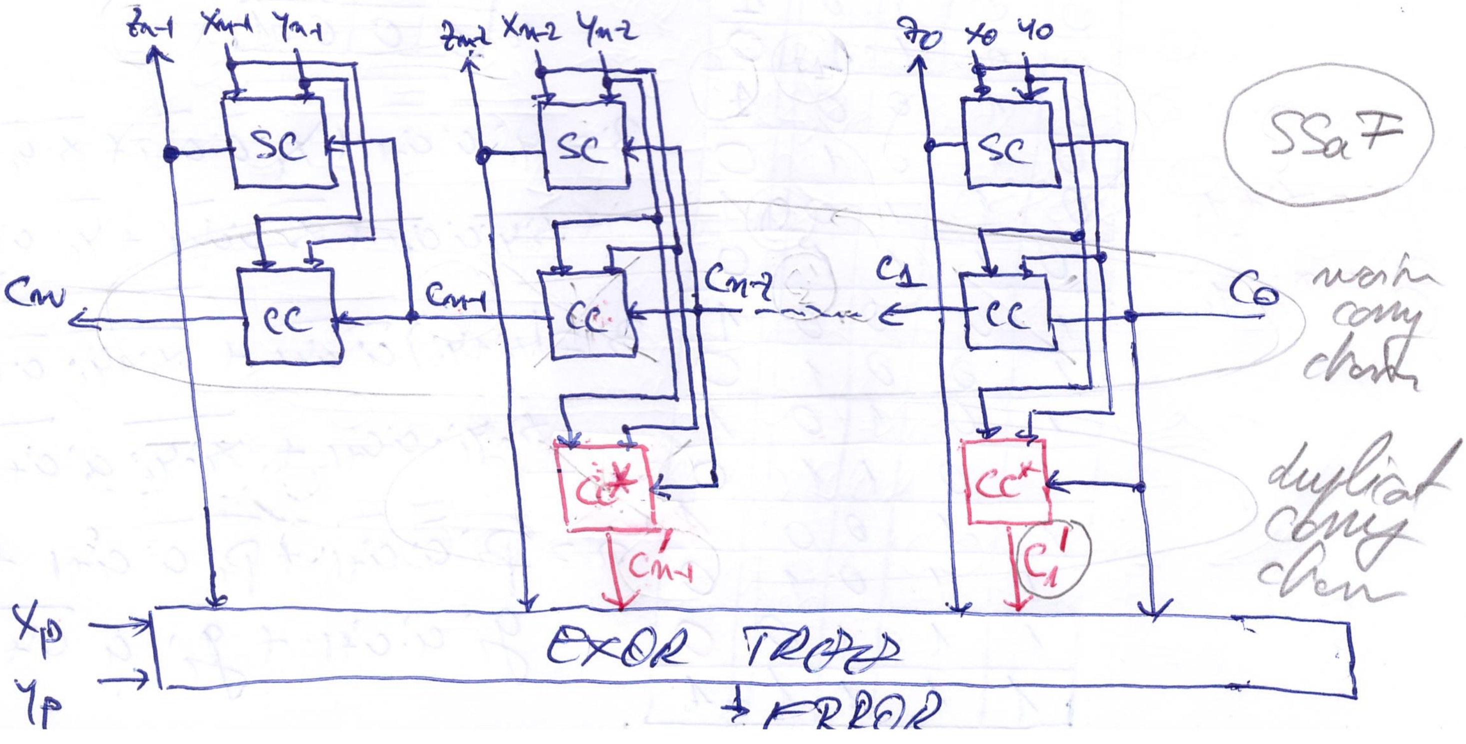
Single parity bit cannot detect errors affecting the carry chain!!

Solution:
- carry chain duplication
- Carry Dependent Sum Adder

1.5.2. Carry Chain Duplication
- adds an additional carry chain
  - used for calculating $z_p(2)$
- split FAC in 2 cells

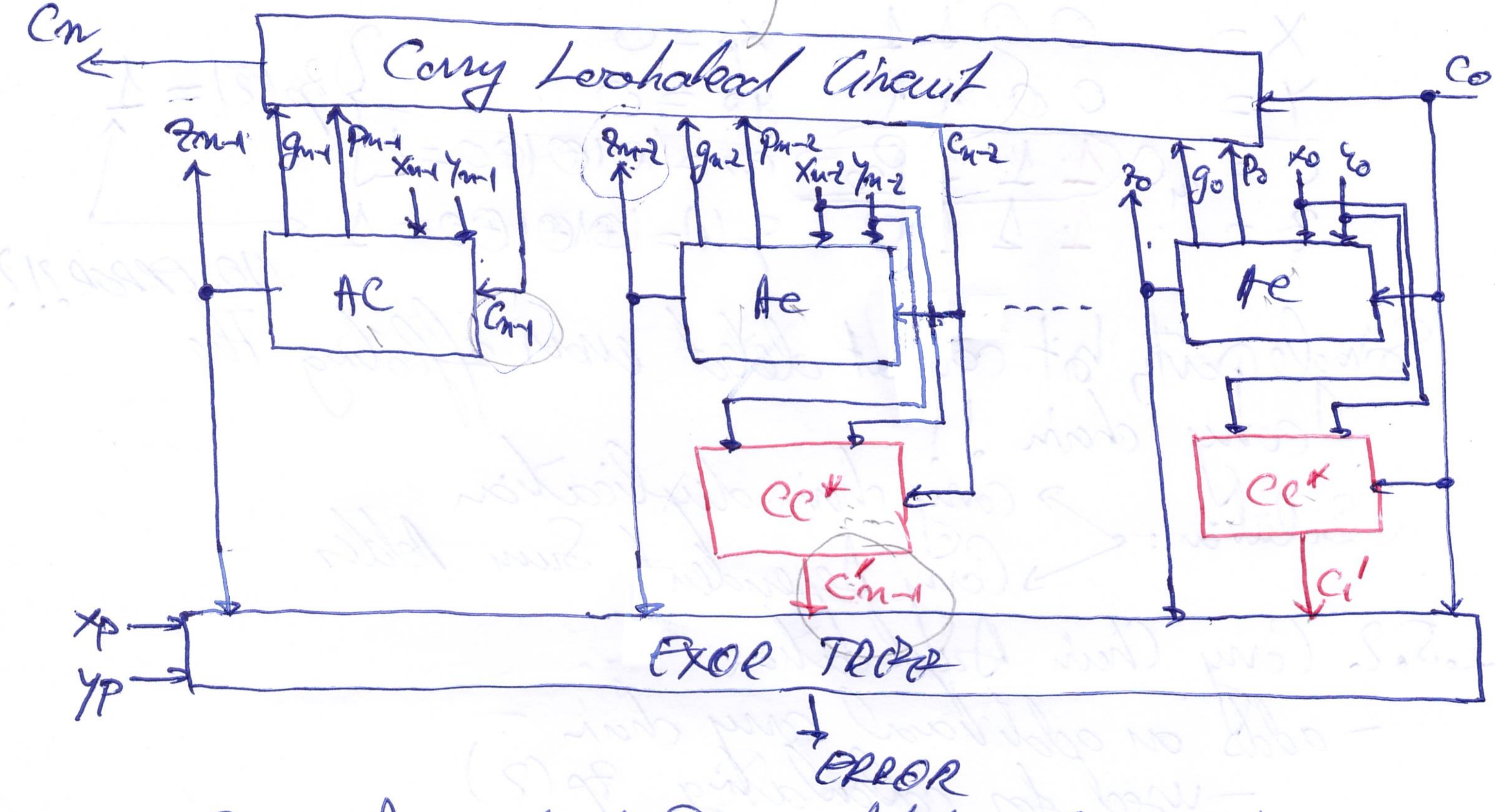


A) RCA



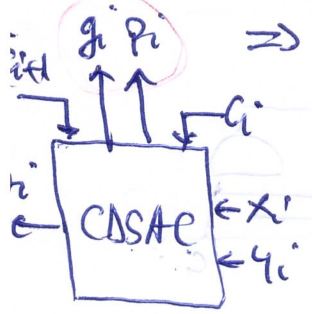
(B) CLA

c_{i+1}: $p_0, \dots, p_i$; $g_0, \dots, g_i, 0$



1.5.3. Carry Dependent Sum Adder (CDSA)

- address reliability from design phase
  - ⇒ create an imbalance in the number of erroneous bits
- if c_{i+1} is incorrect → force z_i to be incorrect



Inputs				Output
x_i	y_i	c_i	c_{i+1}	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

$p_i = x_i + y_i$

$g_i = x_i \cdot y_i$

z_i: $x_i y_i$ \ $c_i c_{i+1}$	00	01	11	10
00	0	1	0	1
01	1	0	0	1
11	1	0	1	0
10	1	0	0	1

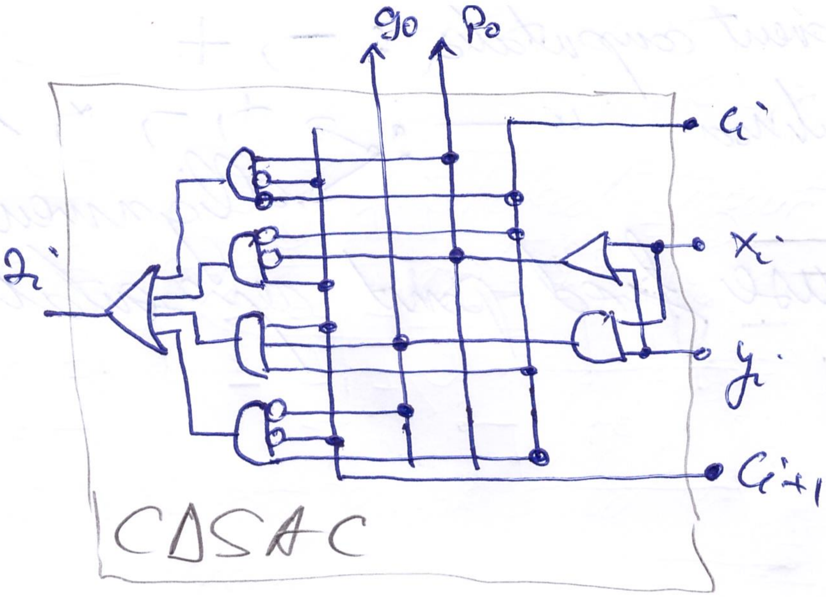
$\overline{a \cdot b} = \overline{a} + \overline{b}$

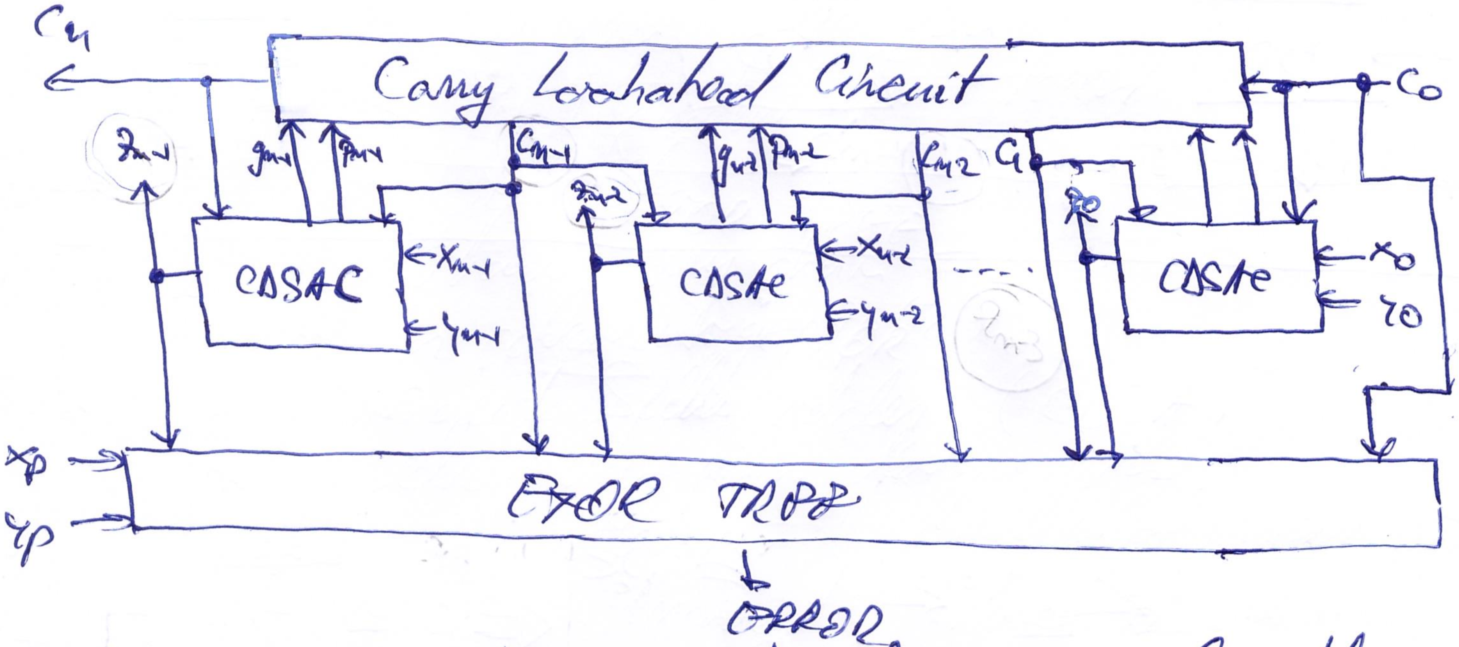
$\overline{a + b} = \overline{a} \cdot \overline{b}$

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

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

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





Chapter 2 Functional Analysis and Synthesis of Floating Point Arithmetic Units

2.1. F.p. arithmetic operations and architectures

IEEE 754 → packed (normalised): storage / transmission
→ unpacked: during computations

Let $X = X_M * 2^{X_E}$, $Y = Y_M * 2^{Y_E}$

allignment

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

$$X - Y = (X_M \ominus Y_M * 2^{Y_E - X_E}) * 2^{X_E}, \text{ if } X_E \ge Y_E$$

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

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

2 subunits

- exponent computation : –, +
- mantissa — " — :
  - +, –, *, /
  - alignment $\times 2^{y_E - x_E}$

both subunits use fixed-point representations