CA C9

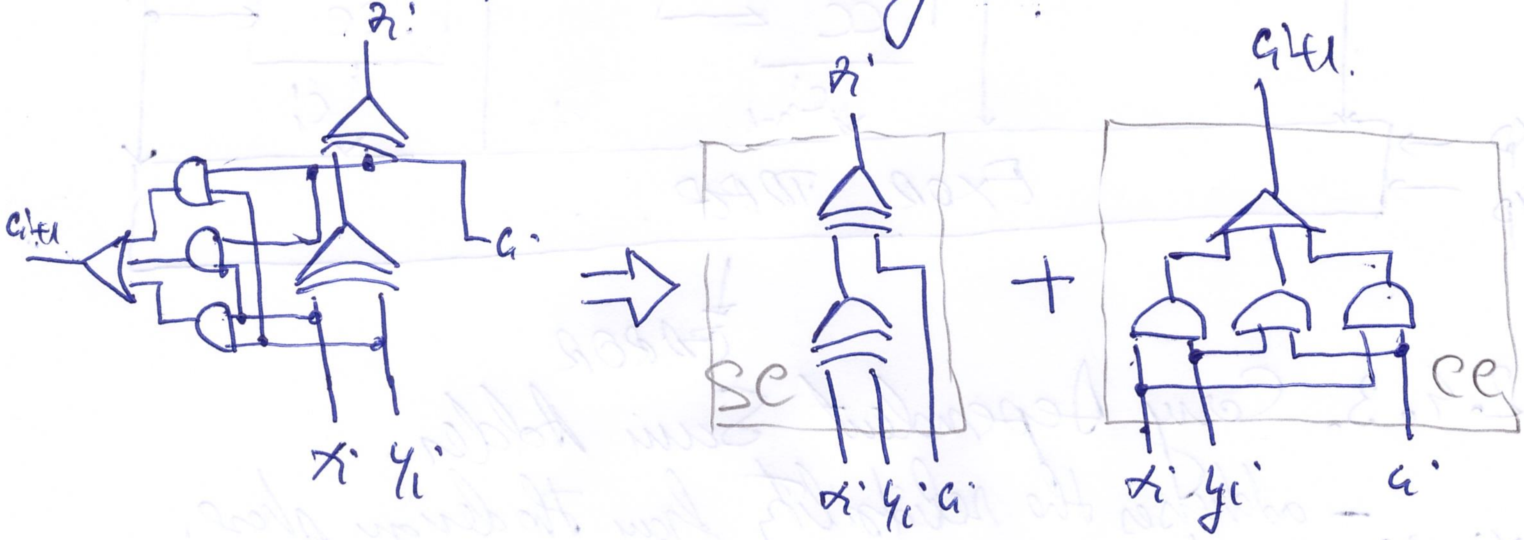
①

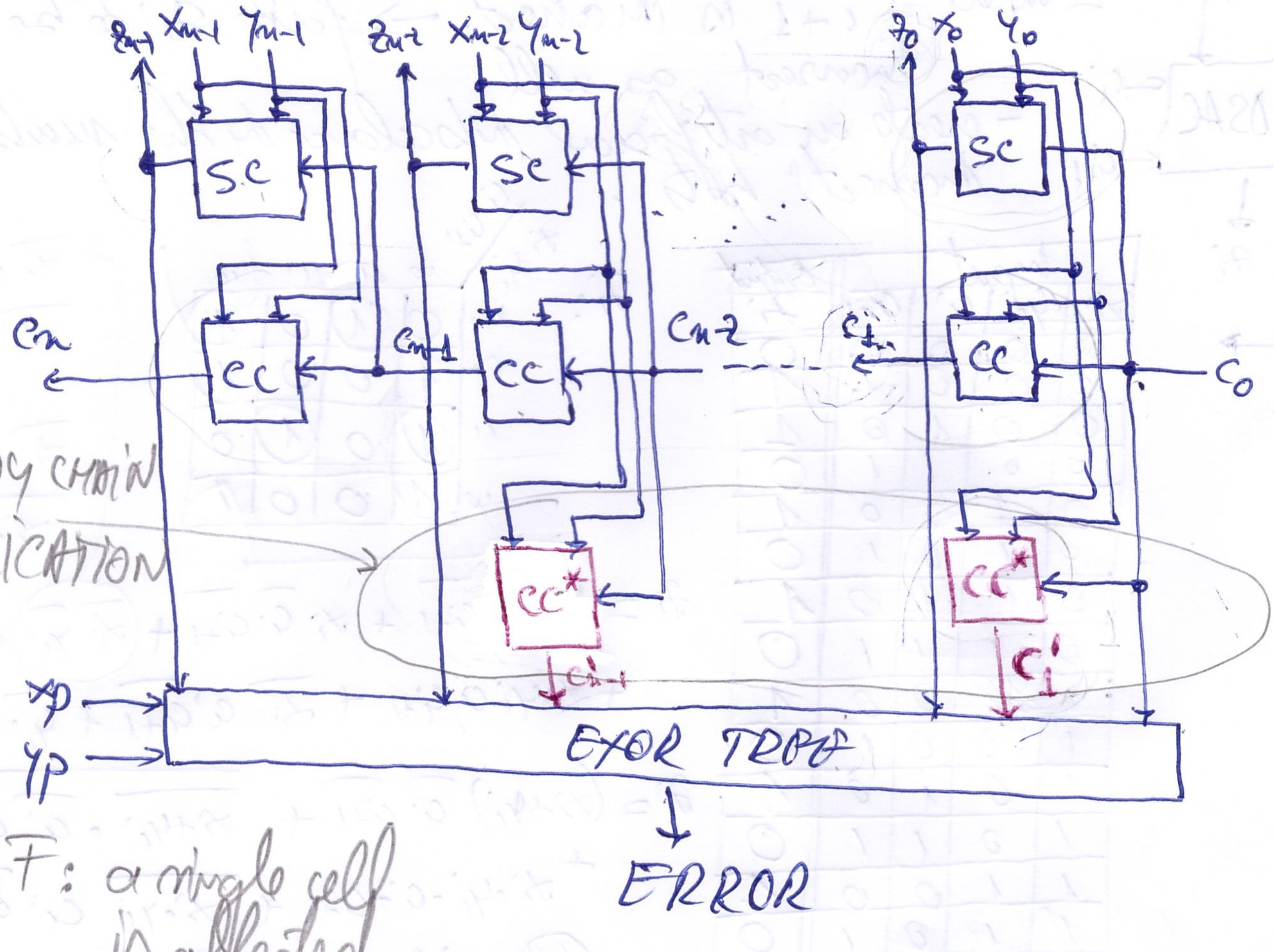
Detecting carry chain stuck-at faults
- Carry Chain Duplication
- Carry Dependent Sum Adder

2.4.2 Carry Chain Duplication

Ⓐ Applied to RCA

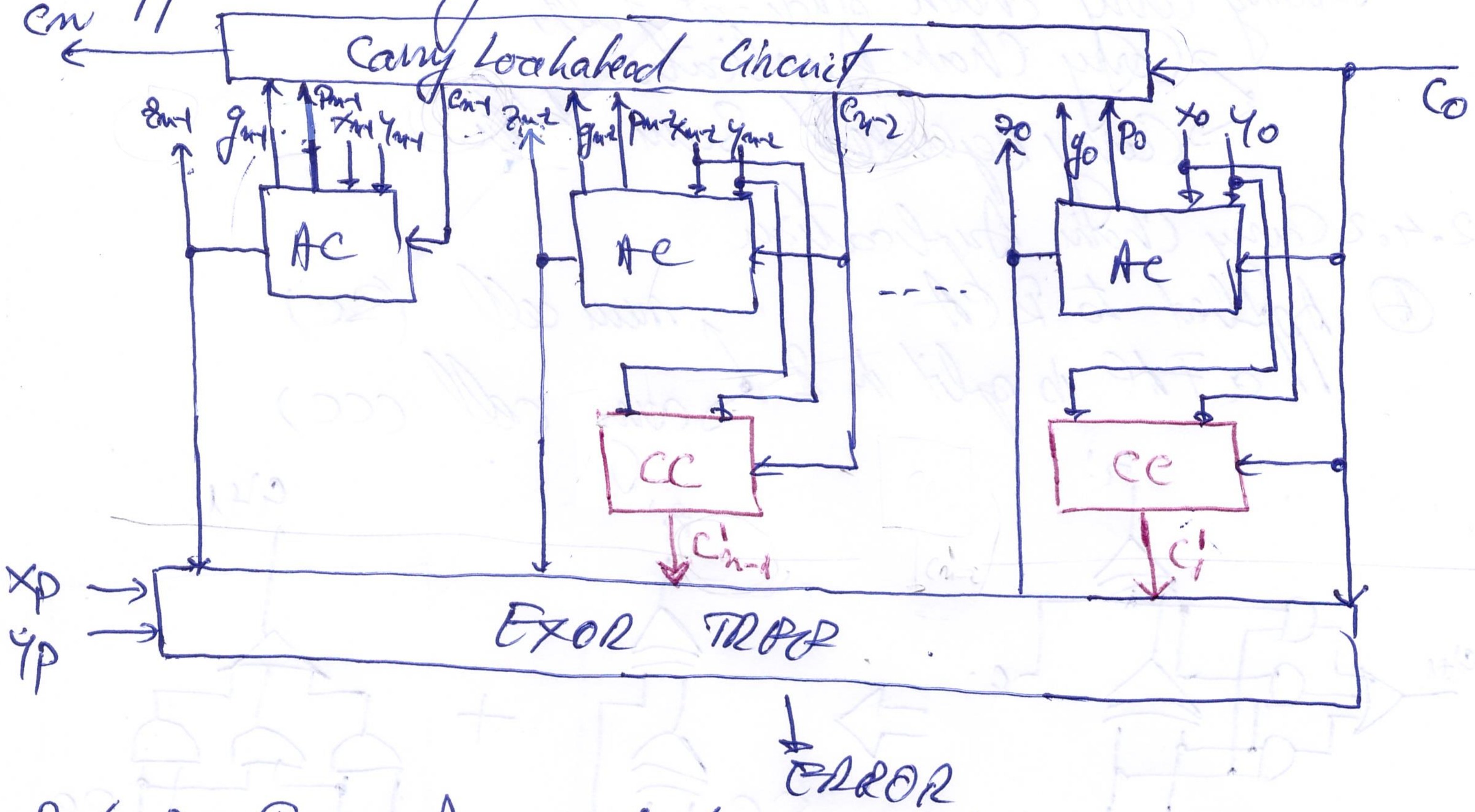
// a FA is split in 2
- sum cell (SC)
- carry cell (CC)





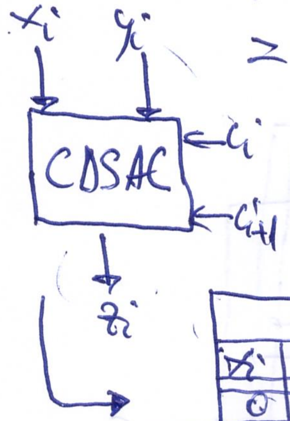
SSaF: a single cell is affected.

(B) Applied to Carry Lookahead Adder.



2.4.3. Carry Dependent Sum Adder.

- addresses the reliability from the design phase.
  - when c_{i+1} is incorrect → force z_i to be incorrect as well
- create an artificial imbalance in the number of incorrect bits.



Input				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

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

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

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

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

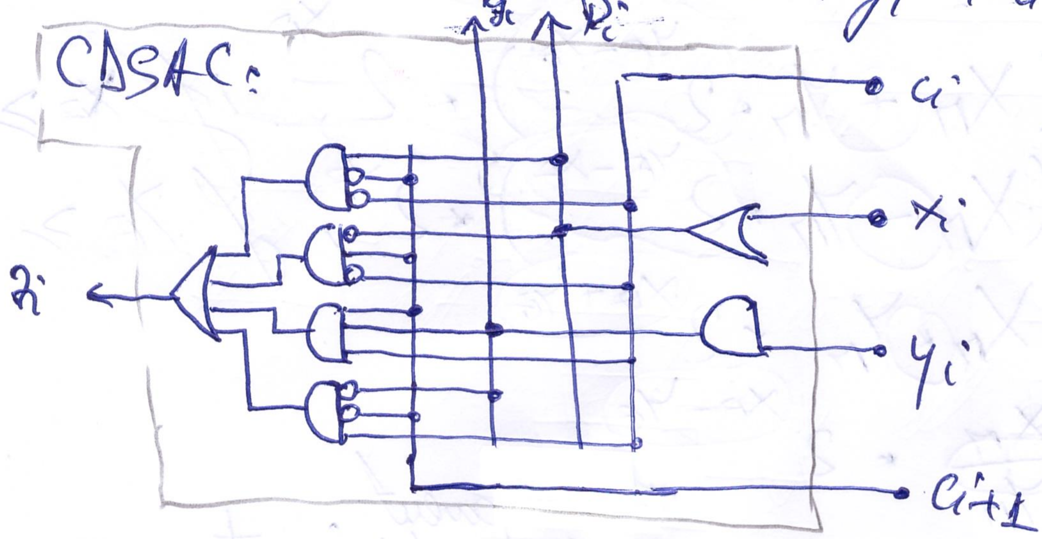
$$z_i = (x_i + y_i)\overline{c_i}\,\overline{c_{i+1}} + \overline{x_i + y_i} \cdot \overline{c_i} c_{i+1} + x_i \cdot y_i \cdot c_i \cdot c_{i+1} + \overline{x_i \cdot y_i}\, c_i \overline{c_{i+1}}$$

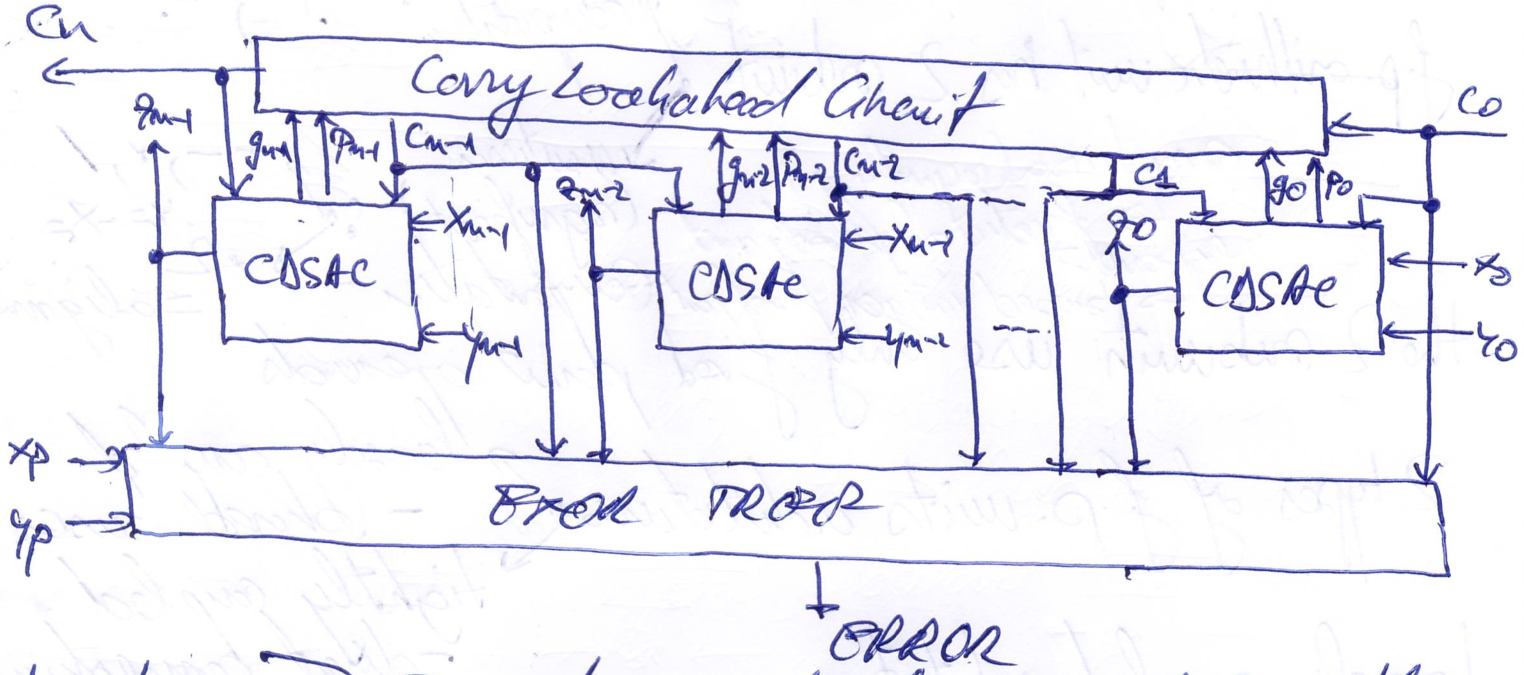
$p_i = x_i + y_i \quad g_i = x_i \cdot y_i$

(2)

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

CDSAC:





Chapter III Functional Analysis and Synthesis of Floating Point Arithmetic Units.

3.1. F. p. operations and architectures

IEEE 754 format of operands

- can be represented in 2 distinct manners.
  - packed: for storage / data transmission.
    - → normalized
  - unpacked: during computations.

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

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

alignment

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

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

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

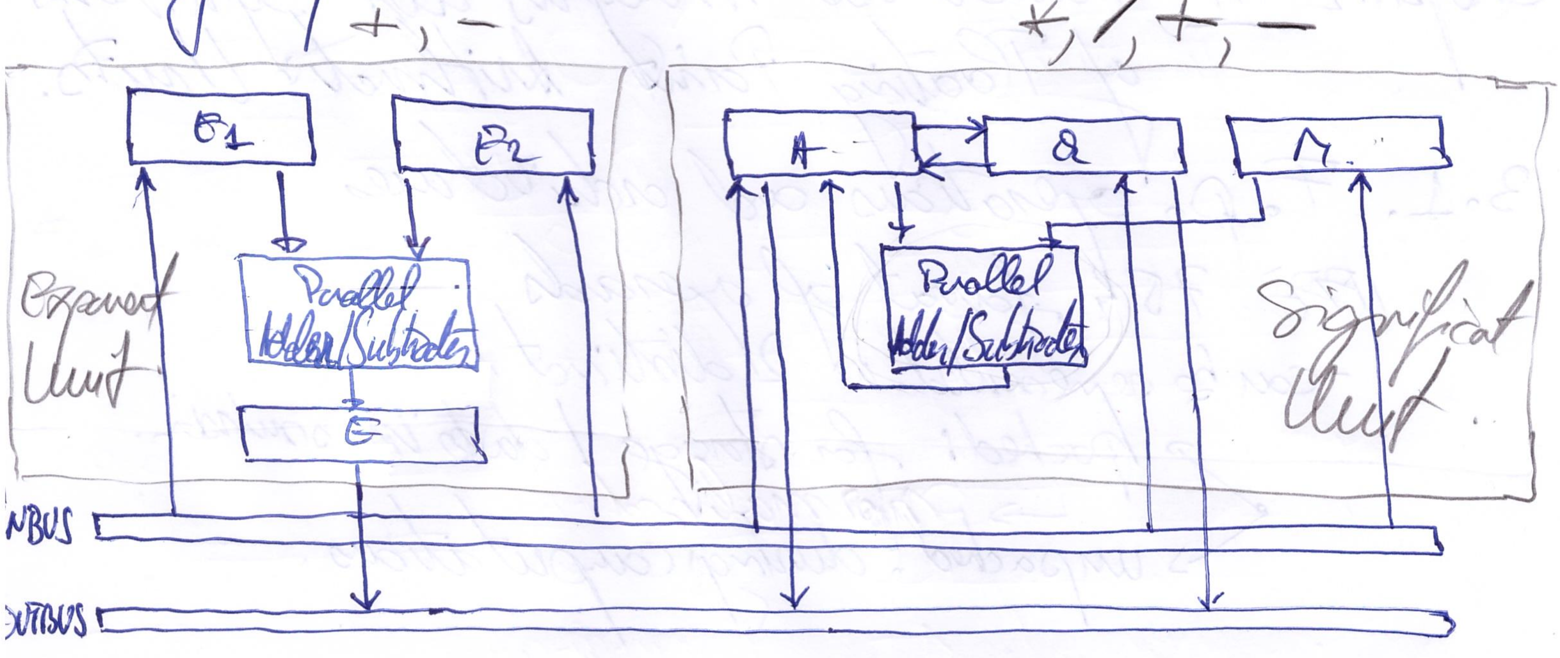
f.p. arithmetic unit has 2 subunits
- exponent computation: +, −
- mantissa (significant) computation: +, −, *, / ; $* 2^{Y_E - X_E}$ = alignment

mantissa (significand) → fractional fixed point
exponent → biased integer fixed point

the 2 subunits use only fixed point operands

2 types of f.p. units architectures
- loosely coupled – shared buses
- tightly coupled – direct connections

Loosely coupled architecture



Tightly coupled architecture
- expensive: direct connections between the exp. & signif. subunits
- faster operation.

- for IBM floating point format
  - IBM's S360 Model 91
  - 2 f.p. units: +, − ; *, /
  - operands were on 32/64 bits.