CA4

$612.8046875_{10} = 1001100100.1100111_2$

Normalize

$= 1.\underbrace{0011001001100111} * 2^9$

$X_e = 9 + \text{bias} = 9 + 127 = 128 + 8 = 10001000_{(2)}$

$X_s^* = .00110010011001111$

S	X_e																														
0	1	0	0	0	1	0	0	0	0	0	1	1	0	0	1	0	0	1	1	0	0	1	1	1	0	0	0	0	0	0	0
4				4				1				9				3				3				8				0$_{(16)}$			

Etapa 2:

$412A0000_{(16)}$

$\Downarrow$

S	X_e								X_s^*																						
0	1	0	0	0	0	0	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

$S = 0 \rightarrow$ positive

$X_e = 10000010 = 128 + 2 = 130$

$X_s^* = .010101$

hidden bit

$X = (-1)^S * 2^{X_e - \text{bias}} * (1.X_s^*) =$ fractional part of significand

$= (-1)^0 * 2^{130-127} * (1.010101) =$

$= 2^3 * 1.010101 = \underbrace{1010}_{10}.101 =$

$= 10 + 2^{-1} + 2^{-3} = 10 + 0.5 + 0.125 = 10.625$

1.4.3 IBM Floating-Point format

- IBM S/360, S/370
- 32, 64, 128 bit formats.

- 32-bit format.

S	X_e	X_m^*
<1>	<7>	<24>

- Radix $16 = 2^4$
- no hidden bit

value: $X = (-1)^S * 16^{X_E - bias} * (0.X_n^*)$

bias = 64

.0000

- mantissa's normalization:
  - can have at most 3 leading 0s.
- Special case: zero $X_E = 0$, $X_n^* = 0$

Example: $612.8046875_{10} = 0.001001100100.1100111_{(2)}$

$= 0.001001100100\,1100111 * 16^3$

$X_E = 3 + bias = 64 + 3 = 1000011_{(2)}$

$X_n^* = .00100110010011 00111$

S	X_E							X_n^*																							
0	1	0	0	0	0	1	1	0	0	1	0	0	1	1	0	0	1	0	0	1	1	0	0	1	1	1	0	0	0	0	0
4		3		2		6		4		C		E		0$_{(16)}$																	

Example2: $412A0000_{(16)}$

0 | 100 | 0001 | 0010 | 1010 | 0 ——— 0

S | X_E | X_n^*

$S=0$, $X_E = 1000001 = 64+1 = 65$

$X_n^* = .00101010$

$X = (-1)^S * 16^{X_E - bias} * (0.X_n^*)$

$= (-1)^0 * 16^{65-64} * (0.00101010)$

$= 2^4 * 0.0010101 = 10.101$

$= 2.625$

Chapter II Functional Analysis and Synthesis of Binary and Decimal Adders and Subtractors Devices.

2.1 Serial Adders

- adds one pairs of bits from the 2 operands, each clock cycle ②

advantages: area ↓, power ↓, frequency ↑

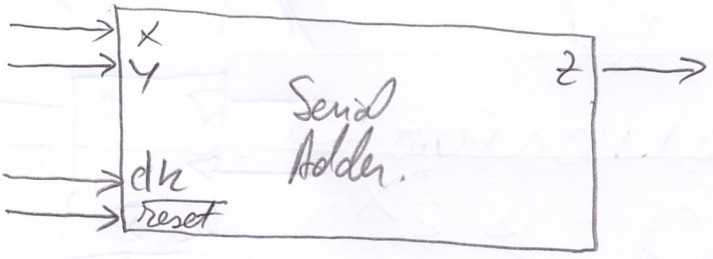
disadvantage: latency for final result ↑

2 types of serial adders → LSDF (Least Significant Digit First)
→ MSDF (Most —"—)

- carry propagate: ⟼
- carry-save:

LSDF:

$$
\begin{array}{lll}
X: & x_{n-1}\, x_{n-2} \text{ ——— } x_1\, x_0 & \\
Y: & y_{n-1}\, y_{n-2} \text{ ——— } y_1\, y_0 & + \\
\hline
Z: & z_{n-1}\, z_{n-2} \text{ ——— } z_1\, z_0 &
\end{array}
$$

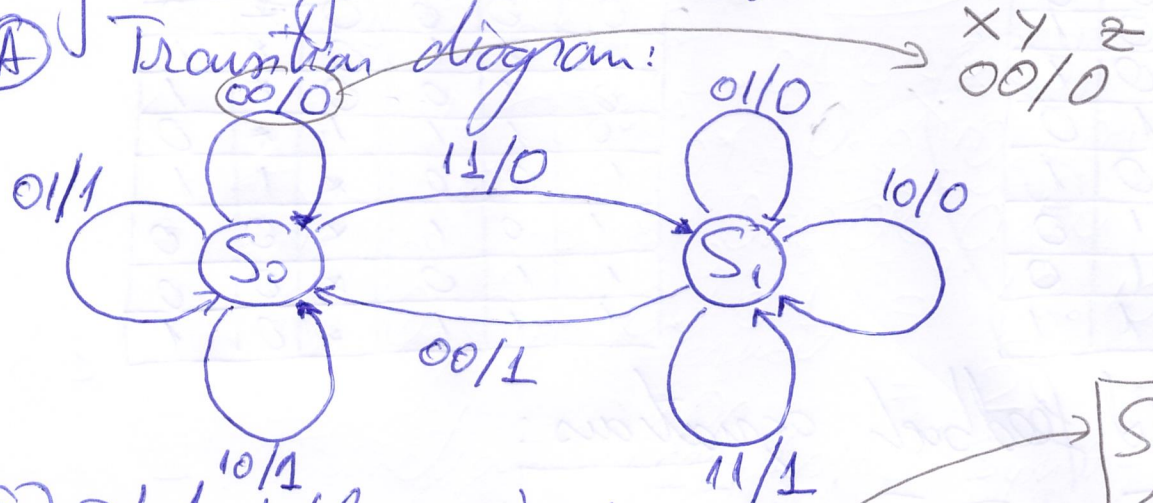


carry propagation: uses the internal state

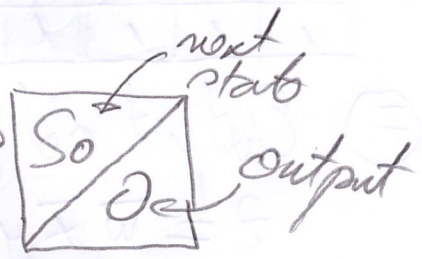
→ 2 states → S_0: no carry from prev. bits.
→ S_1: carry from —"—

Synthesis of a serial adder

Ⓐ Transition diagram:



Ⓑ State table



Input (x, y) / State	00	01	11	10
S_0	S_0 / 0	S_0 / 1	S_1 / 0	S_0 / 1
S_1	S_0 / 1	S_1 / 0	S_1 / 1	S_1 / 0

C) State encoding

- minimum no of state variables with which the states can be represented. $\lceil \log_2 s \rceil$ s = number of states

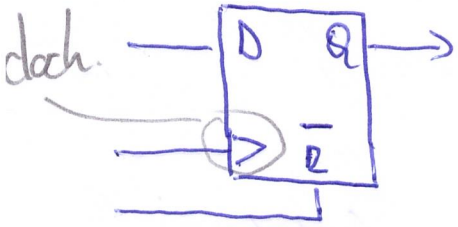
$s = 2 \Rightarrow$ one state variable.

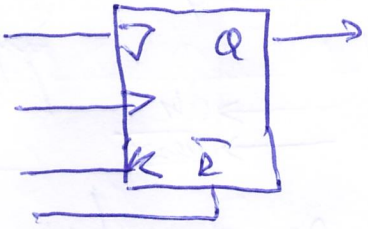
$w \begin{cases} 0: \text{ encodes } S_0 \\ 1: \text{ — " — } S_1 \end{cases}$

D) Transition table:

State variables, w \ Input config.	(x,y) 00	01	11	10
0	0/0	0/1	1/0	0/1
1	0/1	1/0	1/1	1/0

E) Excitation tables: specify the type of storage elements used





$Q(t+1) = D$ next state

$Q(t+1) = J \cdot \overline{Q(t)} + \overline{K} \cdot Q(t)$ next state

Inputs w	x	y	Outputs D	z
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Inputs w	x	y	Outputs J	K	z
0	0	0	0	*	0
0	0	1	0	*	1
0	1	0	0	*	1
0	1	1	1	*	0
1	0	0	*	1	1
1	0	1	*	0	0
1	1	0	*	0	0
1	1	1	*	0	1

F) Output and feedback equations:

$$z = \overline{w}\,\overline{x}\,y + \overline{w}\,x\,\overline{y} + w\,\overline{x}\,\overline{y} + w\,x\,y =$$

$$= \overline{w}\,(\underbrace{\overline{x}\,y + x\,\overline{y}}) + w\,(\underbrace{\overline{x}\,\overline{y} + x\,y}) =$$

$$\overline{w}\,(x \oplus y) + w\,(\overline{x \oplus y}) \Rightarrow$$

$z = x \oplus y \oplus w$

Feedback equations:

D type f.f.

D:

w \ xy	00	01	11	10
0	0	0	1	0
1	0	1	1	1

$D = wx + wy + xy$

JK type f.f.

J:

w \ xy	00	01	11	10
0	0	0	1	0
1	*	*	*	*

$J = x \cdot y$

K:

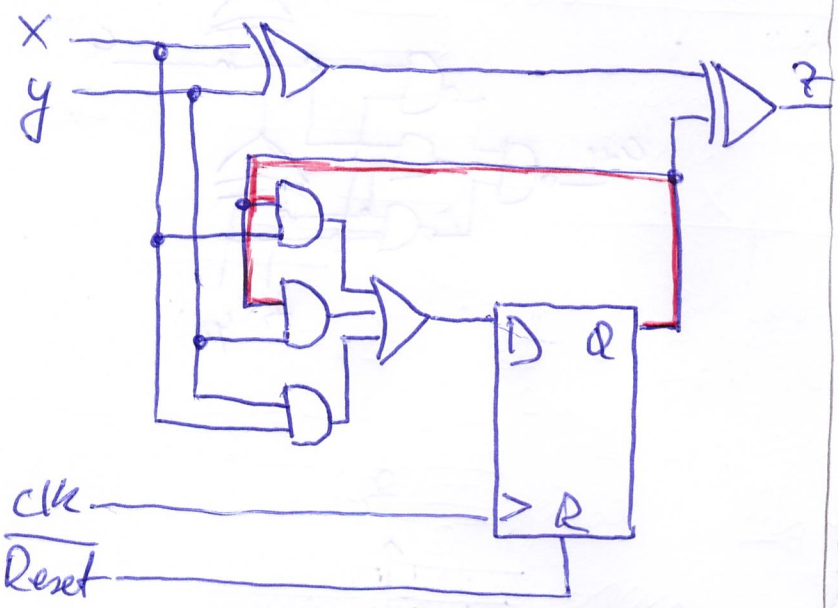
w \ xy	00	01	11	10
0	*	*	*	*
1	1	0	0	0

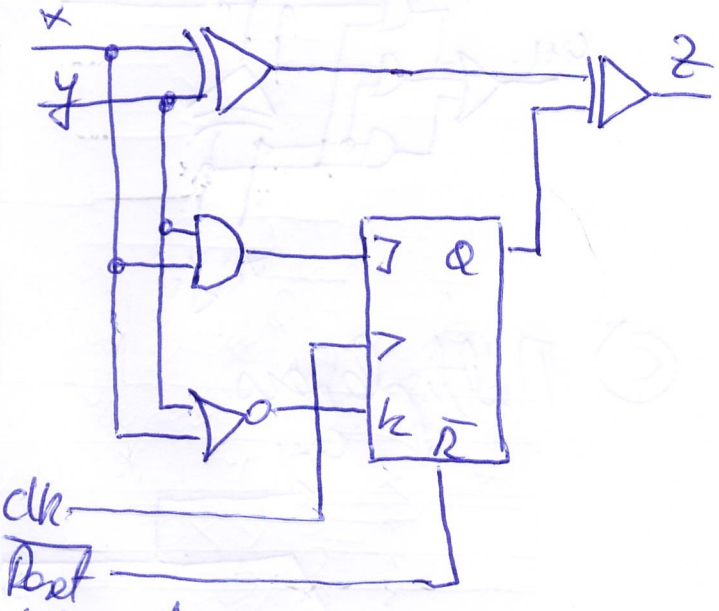
$K = \overline{x} \cdot \overline{y} = \overline{x+y}$

JK type f.f.

Q Synthesis result:

D type f.f.



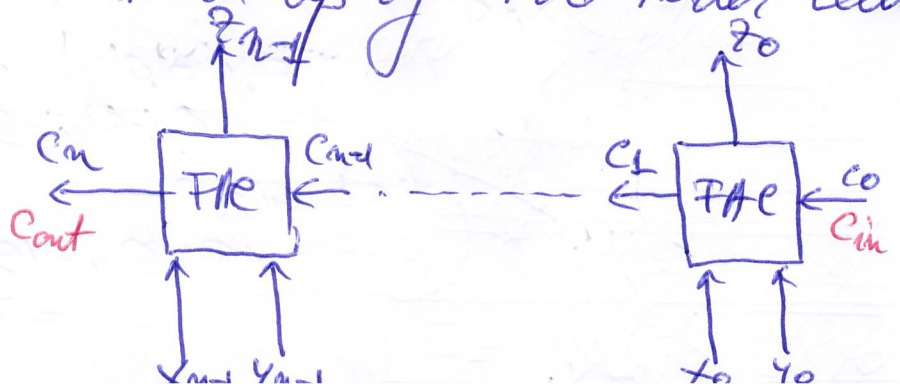


2.2. Parallel Adders and Subtractors

2.2.1. Parallel Adders based on serial carry propagation

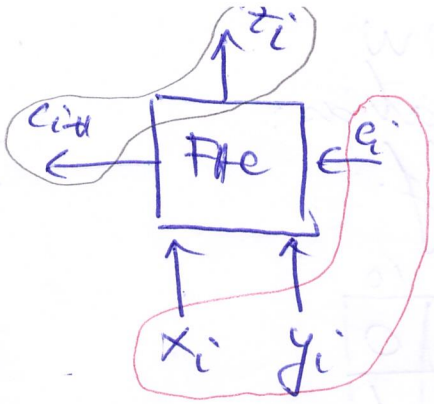
Ripple Carry Adder (RCA)

– made up of Full Adder Cells (FACs)



FAc: truth table

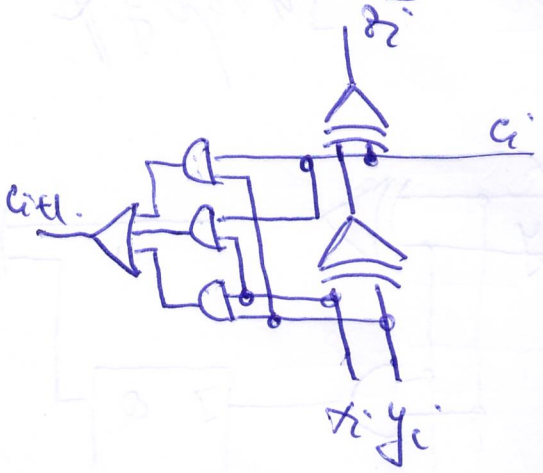
Inputs			Outputs	
x_i	y_i	c_i	c_{i+1}	z_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



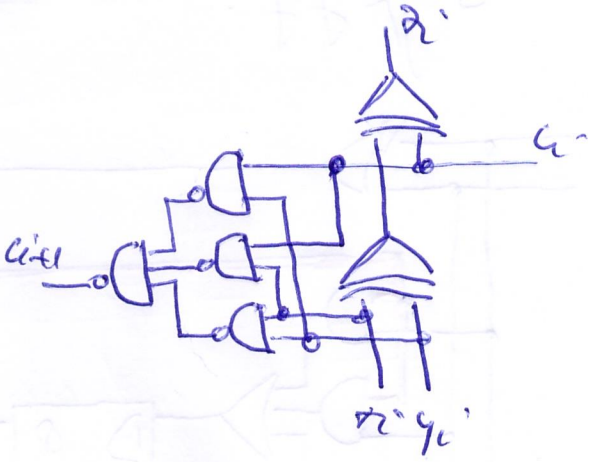
$$z_i = x_i \oplus y_i \oplus c_i$$
$$c_{i+1} = x_i \cdot y_i + x_i \cdot c_i + y_i \cdot c_i$$

FAc's synthesis.

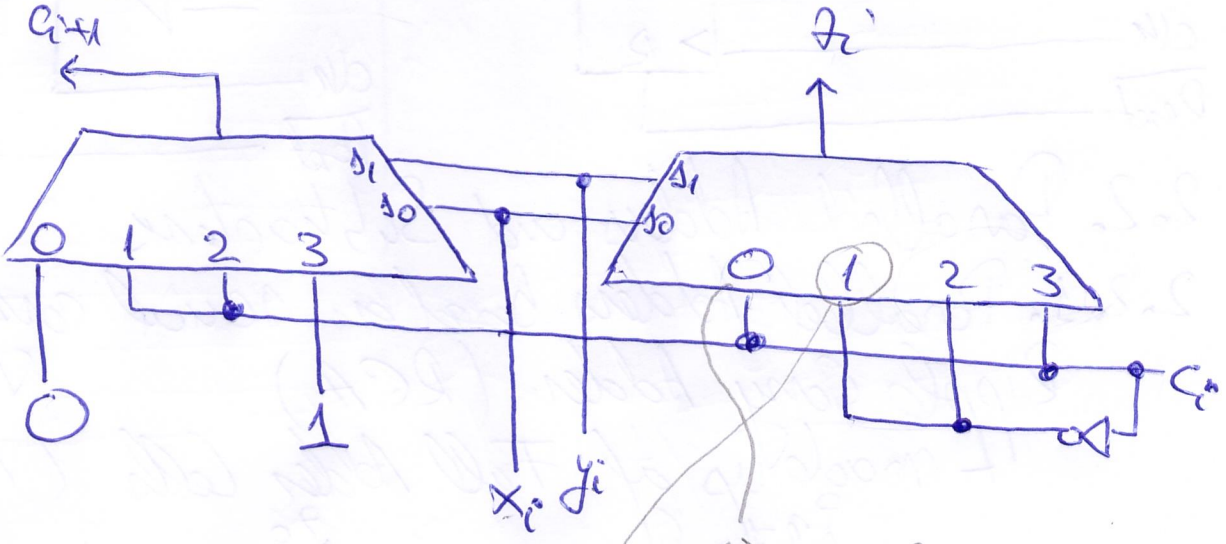
(A) EXOR – AND – OR



(B) EXOR – NAND



(C) Multiplexers.



$x_i = y_i = 0$

$z_i = x_i \oplus y_i \oplus c_i = \cancel{0 \oplus 0 \oplus} c_i$

$x_i = 1 \quad y_i = 0$

$z_i = x_i \oplus y_i \oplus c_i = 1 \cancel{\oplus 0 \oplus} c_i$

$= \overline{c_i}$