

Electronics and Communication Systems

Electronics Systems

Luca Fanucci

Dipartimento di Ingegneria dell'Informazione

Via Caruso 16 - I-56122 - Pisa - Italy

Phone: +39 050 2217 668

Fax: +39 050 2217 522

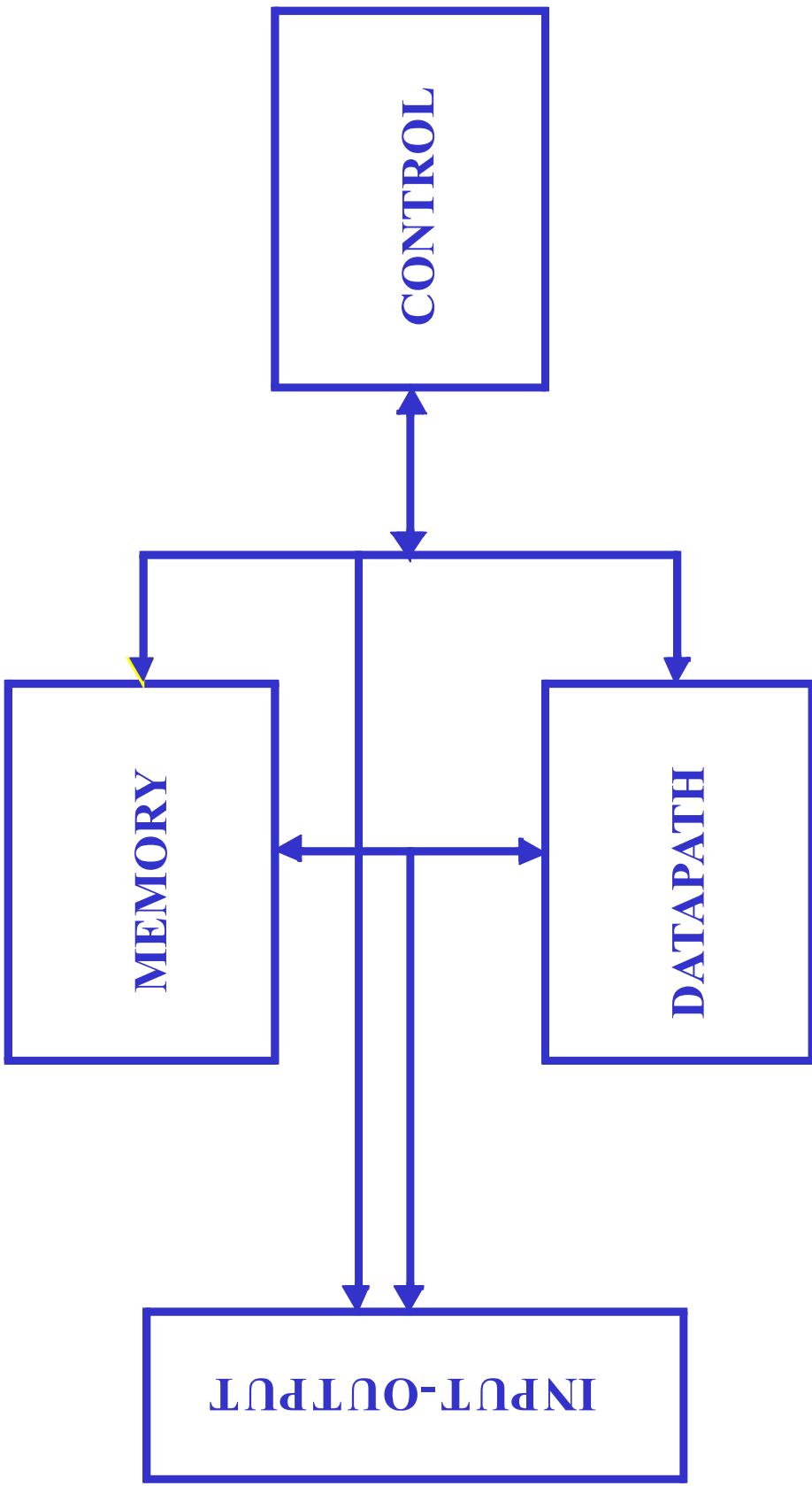
Mobile: +39 347 5013837

Email: luca.fanucci@unipi.it

Outline

- ❖ **Full Adder**
- ❖ **Serial Adder**
- ❖ **Parallel Adder**
 - **Ripple Carry Adder**
 - **Carry Look Ahead Adder**
- ❖ **Pipeline**
- ❖ **Subtractor**
- ❖ **ALU**
- ❖ **Multiplier**

A Generic Digital Processor

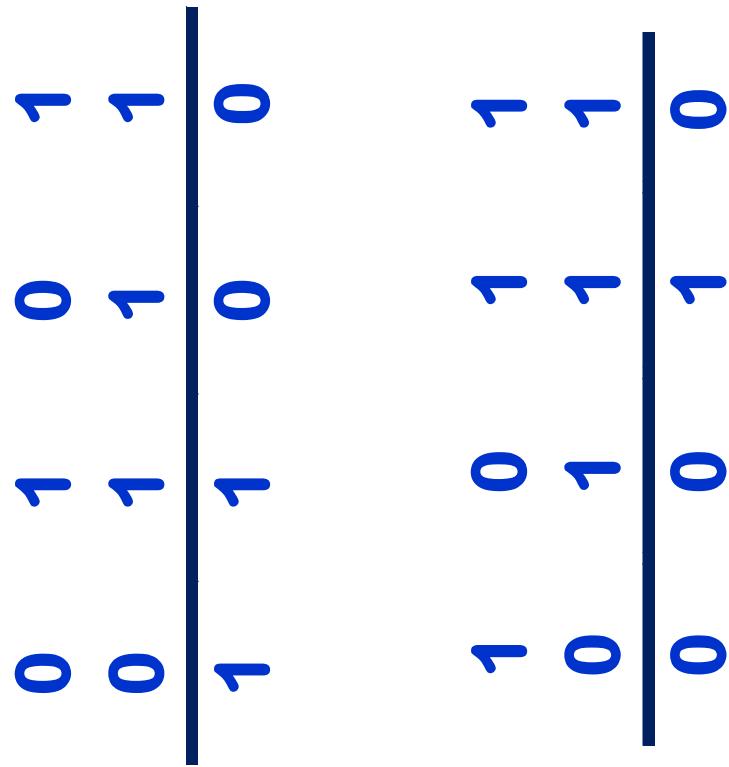


Basic Building Blocks

- ❖ **Datapath**
 - **Execution units**
 - **Adder, multiplier, divider, shifter, etc.**
 - **Register file and pipeline registers**
 - **Multiplexers, decoders**
- ❖ **Control**
 - **Finite state machines (PLA, ROM, random logic)**
- ❖ **Interconnect**
 - **Switches, arbiters, buses**
- ❖ **Memory**
 - **Caches (SRAMs), TLBs, DRAMs, buffers**

Example

$$\star 5 + 7 = 12$$



$$\star (-5) + 7 = 2$$

1 bit adder

❖ full adder

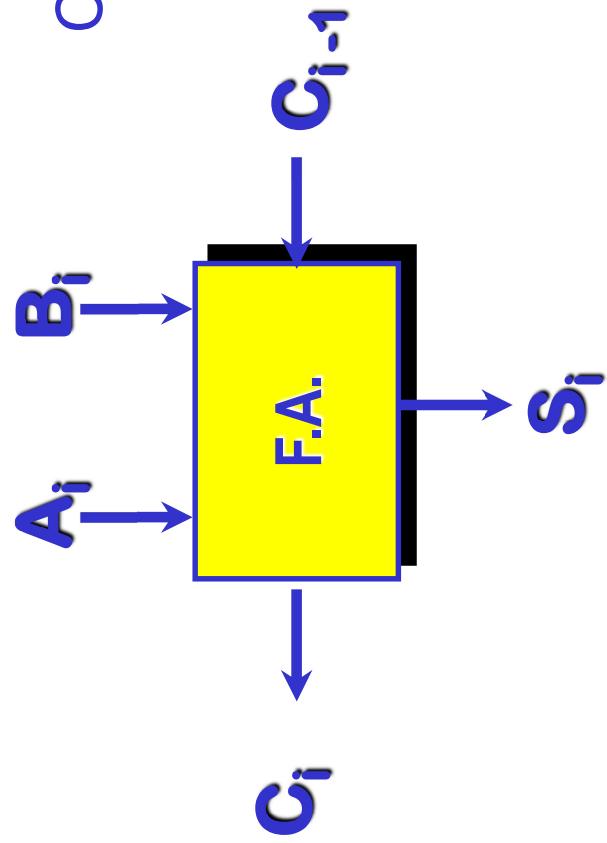
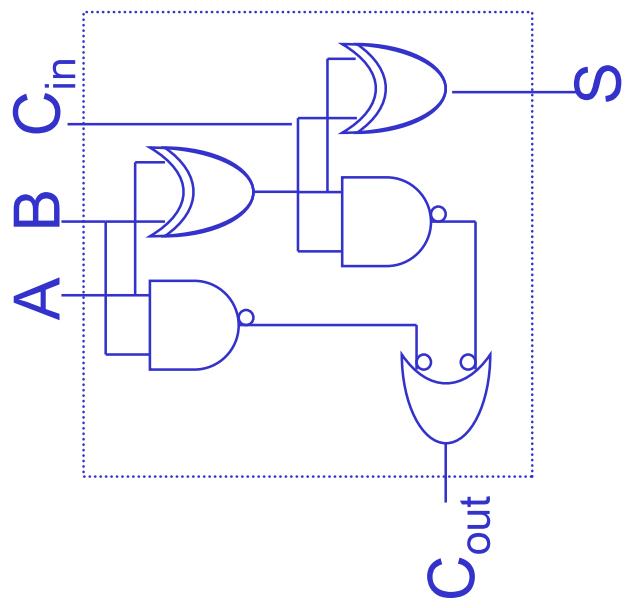
- INPUT
 - 2 1-bit input (A_i, B_i)
 - 1-bit carry (C_{i-1})
- OUTPUT
 - 1-bit sum (S_i)
 - 1-bit carry (C_i)

Truth Table

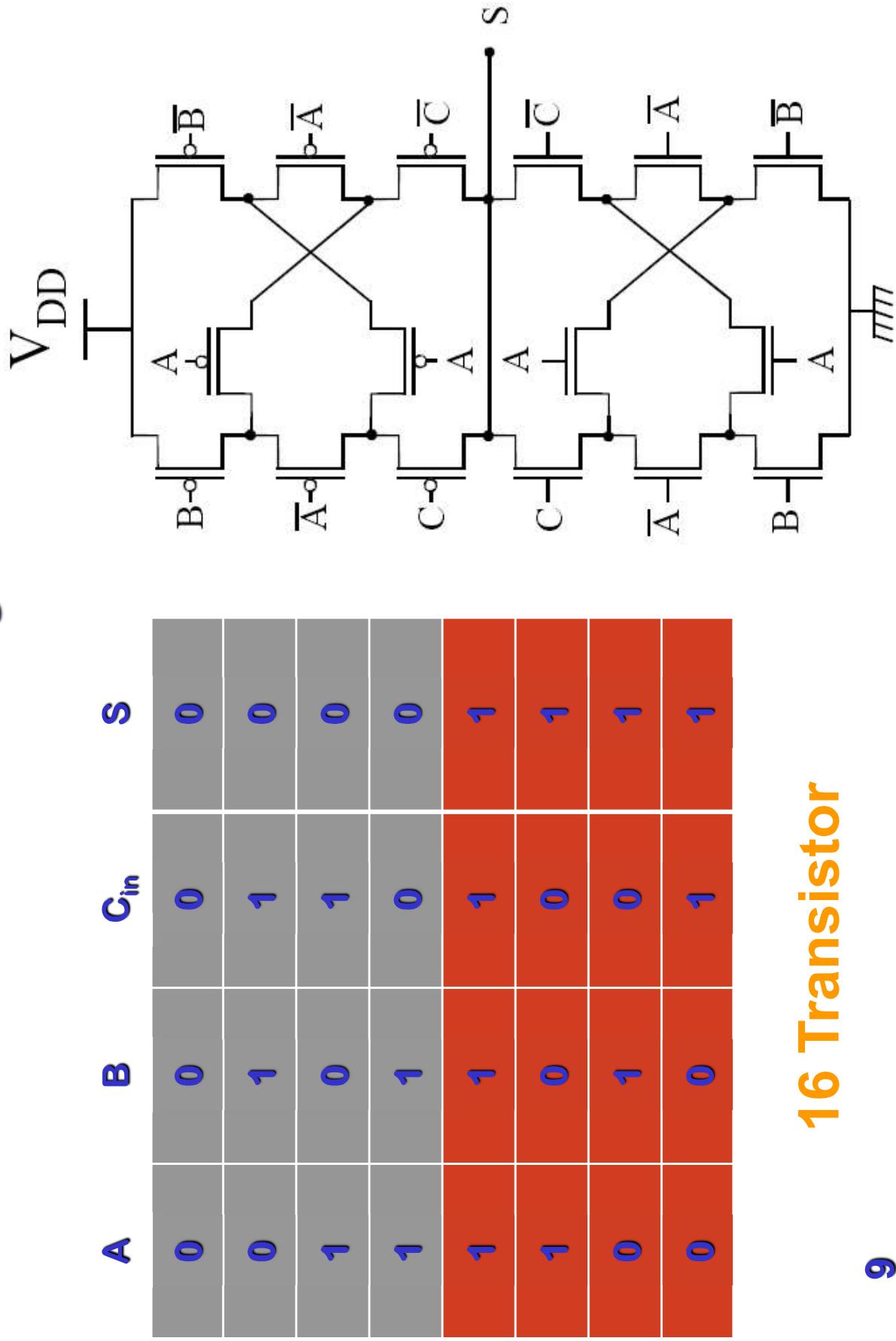
A_i	B_i	C_{i-1}	S_i	C_i
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	0	0

Logic

$$\begin{aligned} \star S_i &= A_i + B_i + C_{i-1} \\ \star C_i &= A_i \cdot B_i + A_i \cdot C_{i-1} + B_i \cdot C_{i-1} \end{aligned}$$

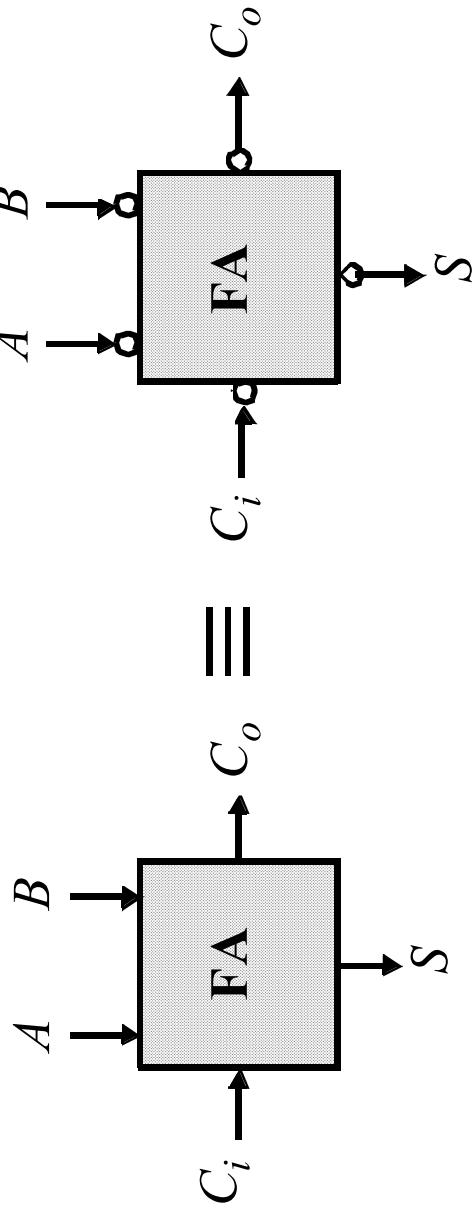


The Binary Adder



16 Transistor

Inversion Property

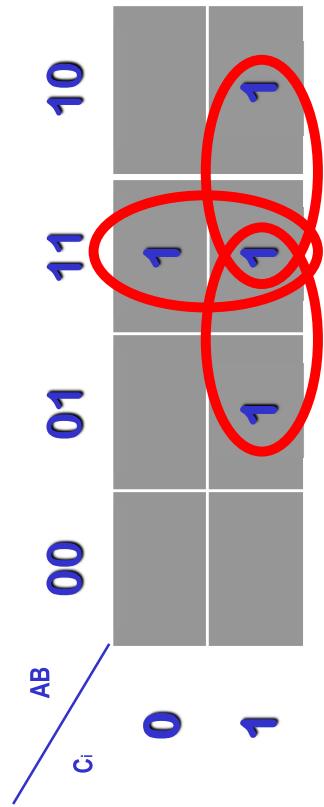


$$\bar{S}(A, B, C_i) = S(\bar{A}, \bar{B}, \bar{C}_i)$$

$$\bar{C}_o(A, B, C_i) = C_o(\bar{A}, \bar{B}, \bar{C}_i)$$

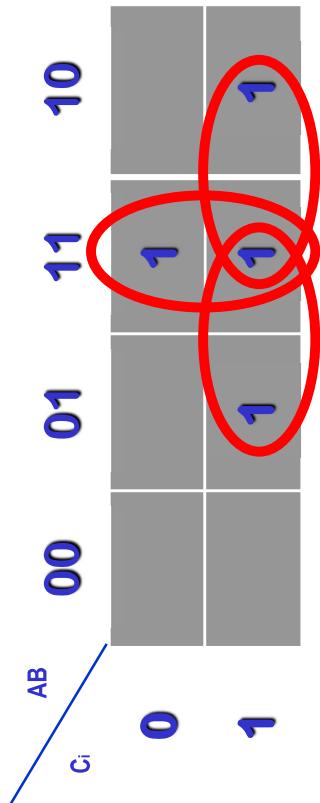
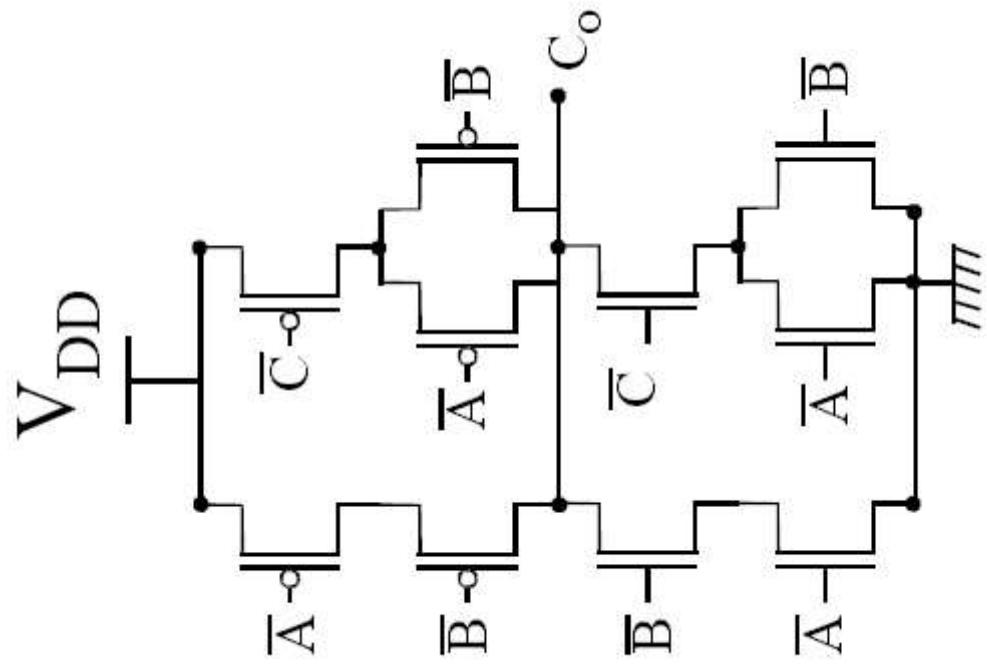
The Binary Adder

A	B	C_{in}	C_{out}
0	0	0	0
0	0	1	0
1	0	0	0
0	1	0	0
1	1	1	1
1	1	0	1
0	1	1	1
1	0	1	1



11

The Binary Adder



10 Transistor

12 Total: $16+10+6=32$ Transistor

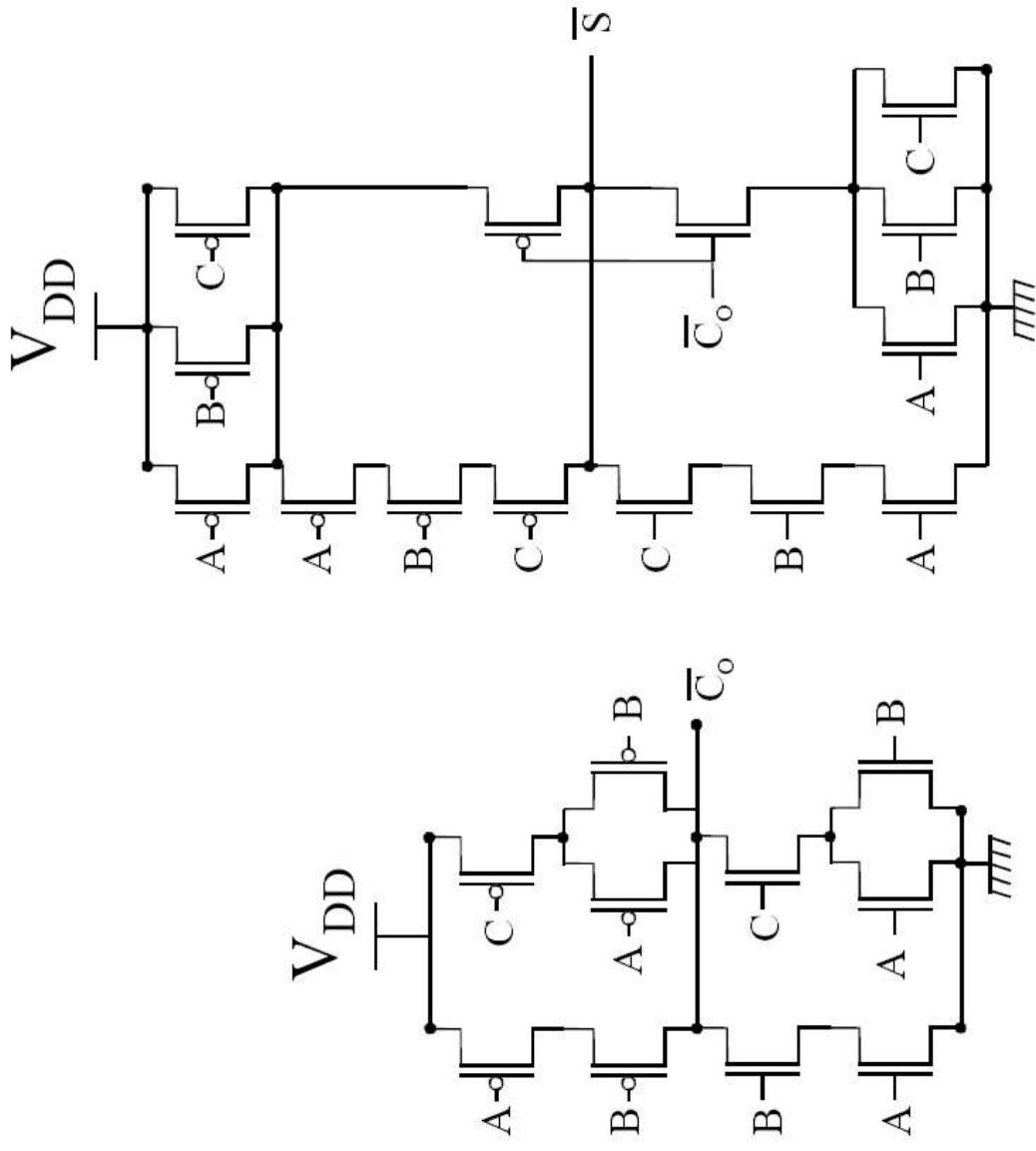
The Binary Adder

$$S = \overline{C_o} (A + B + C) + ABC$$

A	B	C_{in}	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
0	1	1	0	1
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

The Binary Adder

28 Transistors



The 1-bit Binary Adder

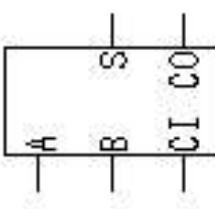
0.35 μm CMOS

FA1

FA1 is a one-bit full adder with 1x drive strength.

Truth Table

A	B	C _i	CO	S
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



Capacitance

Pin	Cap [pF]
A	0.038
B	0.037
C _i	0.032

Power

1.632 $\mu\text{W}/\text{MHz}$

Area

0.564 mils²
364 μm^2

The 1-bit Binary Adder

AC Characteristics

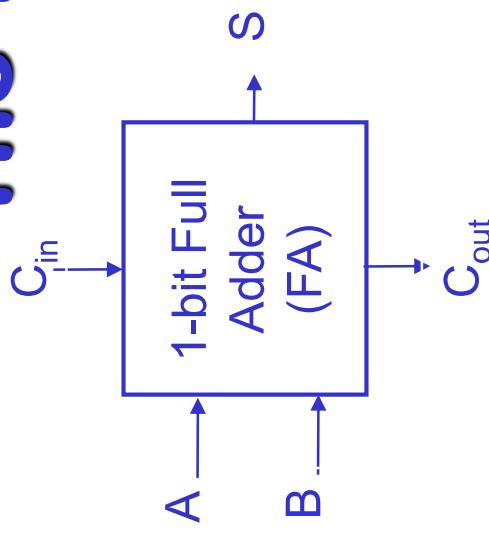
Delay [ns] = $t_{pd..} = f(SL, L)$
Output Slope [ns] = $op_sl.. = f(SL, L)$

with SL = Input Slope [ns]; L = Output Load [pF]
with L = Output Load [pF]

AC Characteristics: $T_j = 25^\circ\text{C}$ $VDD = 3.3\text{V}$ Typical Process

Slope [ns] Load [pF]	Rise			Fall		
	0.1 0.015	0.15 0.015	0.15 0.015	0.1 0.015	0.15 0.015	2 0.15
Delay A => CO	0.47	0.99	0.67	1.18	0.57	1.14
Delay A01 => S	0.55	1.07	0.68	1.21	0.78	1.35
Delay A10 => S	0.83	1.32	0.96	1.44	0.71	1.32
Delay B => CO	0.49	1.01	0.82	1.33	0.54	1.14
Delay B01 => S	0.55	1.07	0.78	1.3	0.8	1.36
Delay B10 => S	0.84	1.33	1.01	1.5	0.7	1.3
Delay C1 => CO	0.41	0.92	0.71	1.22	0.48	1.07
Delay C101 => S	0.59	1.11	0.9	1.42	0.71	1.26
Delay C110 => S	0.76	1.24	1	1.5	0.66	1.27

The 1-bit Binary Adder



- How can we use it to build a 64-bit adder?
- How can we make it faster?
- How can we modify it easily to build an adder/subtractor?

Outline

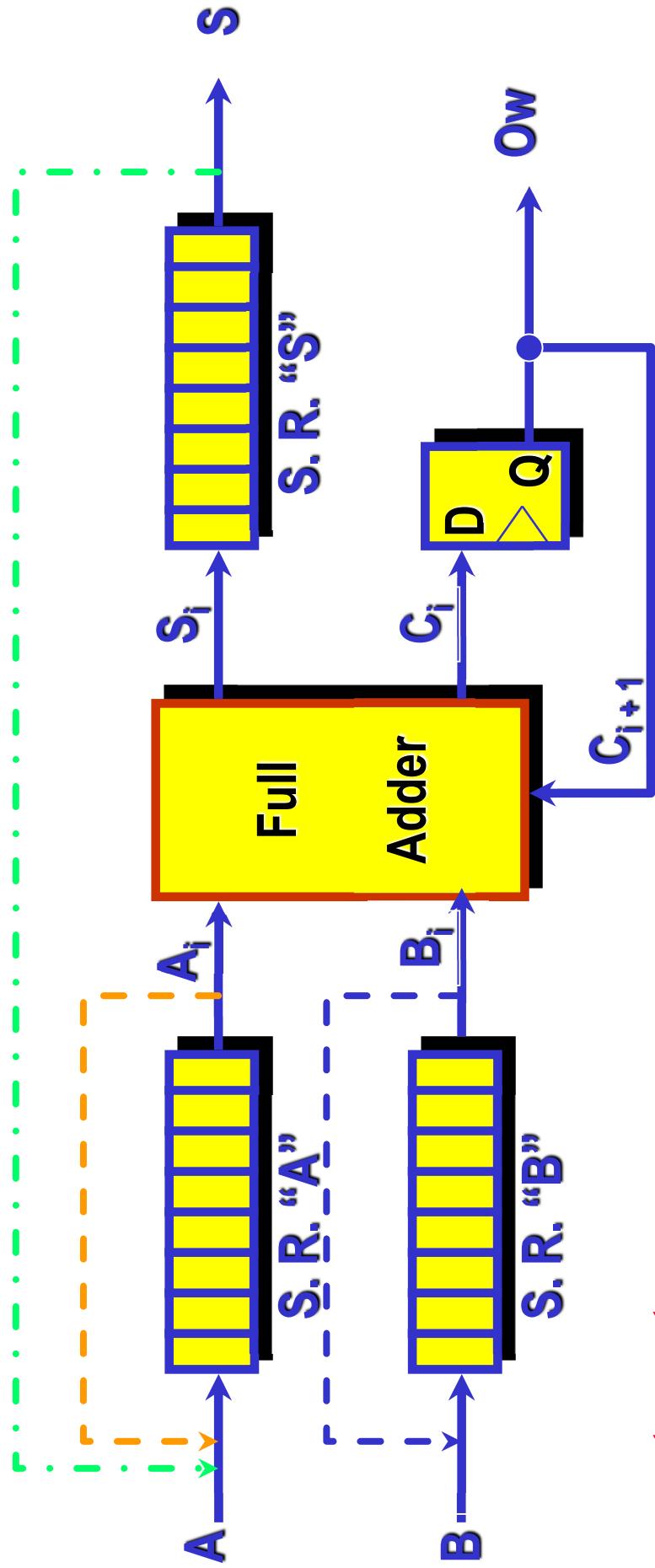
- ❖ Full Adder
- ❖ Serial Adder
- ❖ Parallel Adder
 - Ripple Carry Adder
 - Carry Look Ahead Adder
- ❖ Pipeline
- ❖ Subtractor
- ❖ ALU
- ❖ Multiplier

Adding two “N” bit words

★ Serial Adder

- Need:
 - 1 1-bit Full Adder
 - 3 N-bit Shift Register
 - 1 D edge triggered
 - ⇒ a clock signal
- Addition is performed in N clock cycles

Block Diagram



hp. $t_c > t_s$

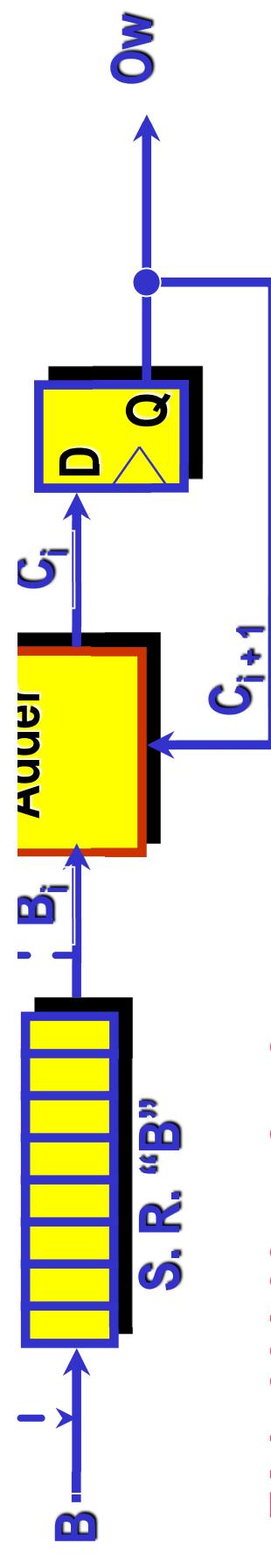
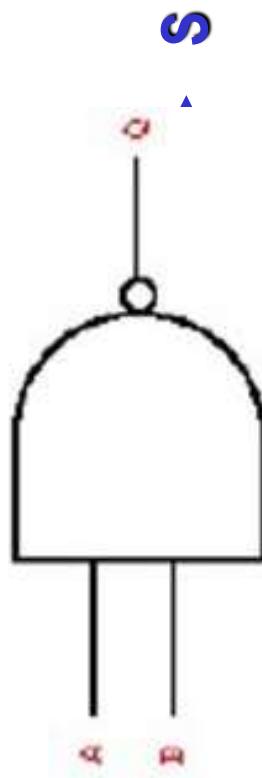
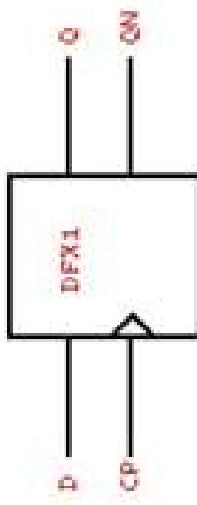
$$2t_{CK} \geq t_{c-q} + t_c + t_{su}$$

$$t_{tot} = N T_{CK}$$

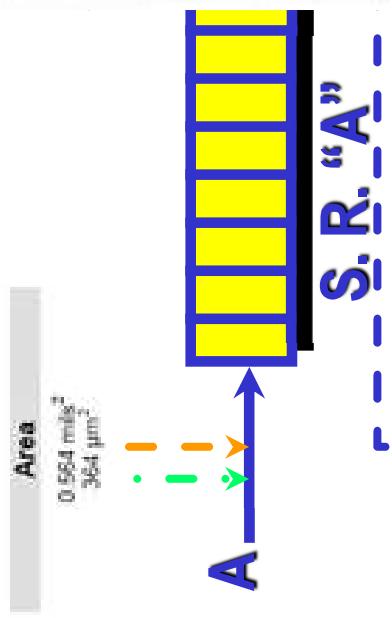
FA1 is a one-bit full adder with 1x drive strength
DFX1 is a edge Flip-Flop with drive strength X1

Truth Table				
A	B	Ci	CO	S
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

Strength	1
Cell Area	232.960 μm^2
Equation	$Q = "D"$ $QN = "I(D)"$
Clock	CP
Type	Sequential
Input	D
Output	Q, QN



FA1: 364,00 μm^2 ~ 8 gate
DFX1: 232,96 μm^2 ~ 5 gate



Notes

❖ Pro

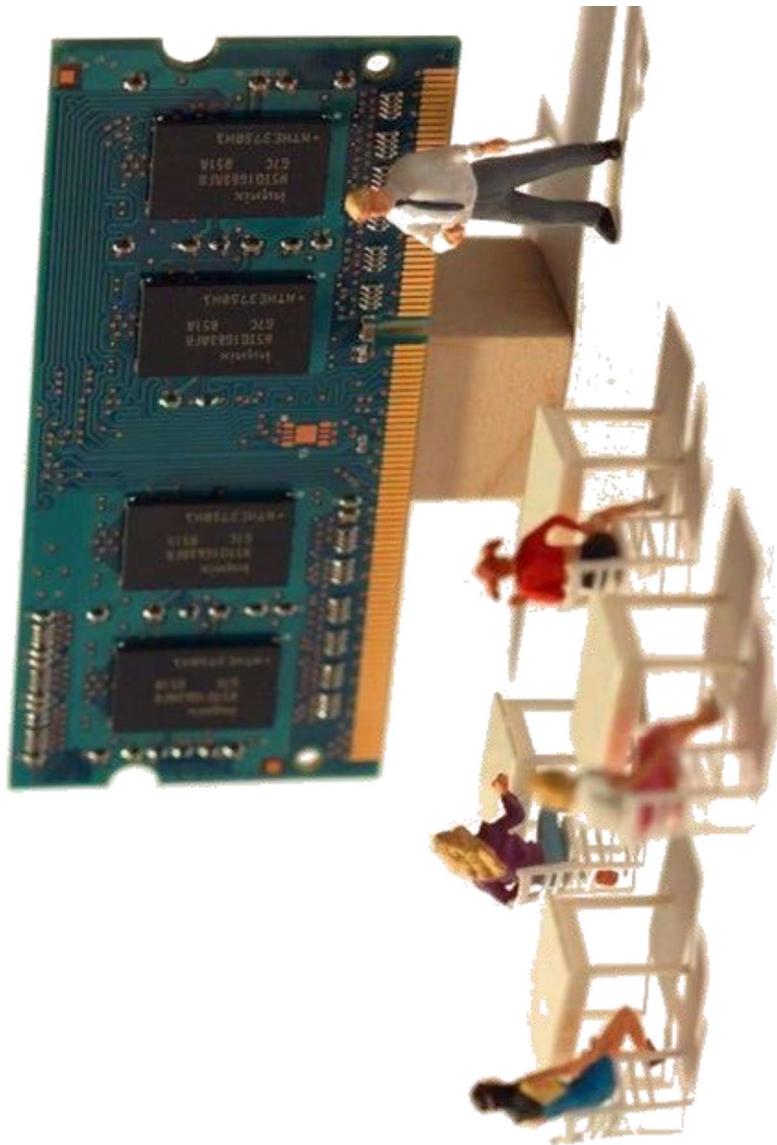
- Low Hardware complexity
- Perform the operation:
$$S \leftarrow A + B$$
- Inputs are kept
- Might perform the operation $A \leftarrow A + B$
saving one Shift Register

❖ Contra

- Long execution time

End, Questions ?

- Full Adder
- Serial Adder



Outline

- ❖ Full Adder
- ❖ Serial Adder
- ❖ Parallel Adder
 - Ripple Carry Adder
 - Carry Look Ahead Adder
- ❖ Pipeline
- ❖ Subtractor
- ❖ ALU
- ❖ Multiplier

Parallel Schemes

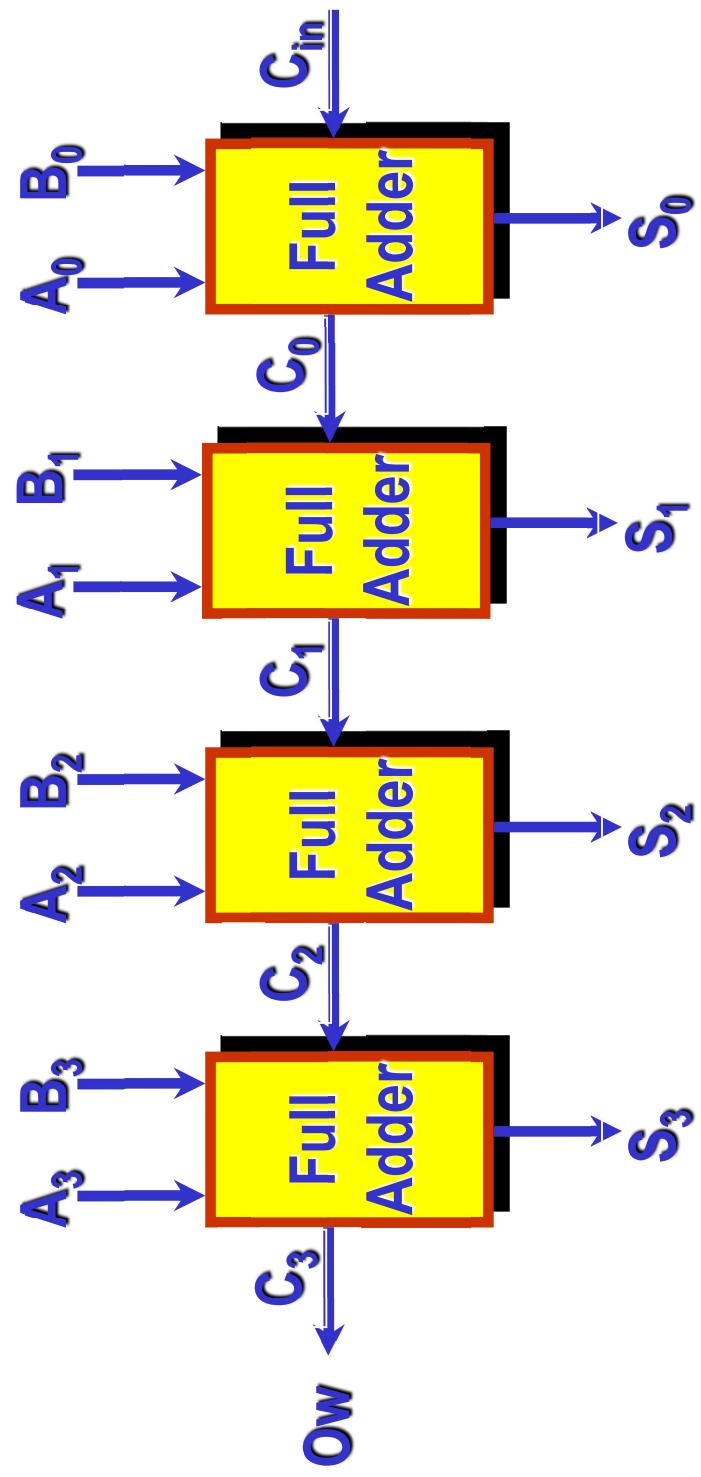
- ❖ Ripple Carry Adder (Carry Propagate Adder)
- ❖ Carry Look Ahead Adder

Adding two “N” bit words

★ Ripple Carry Adder

- Needs:
 - N 1-bit Full Adder
- A **combinational network**
- **Easy to be change to a subtractor**

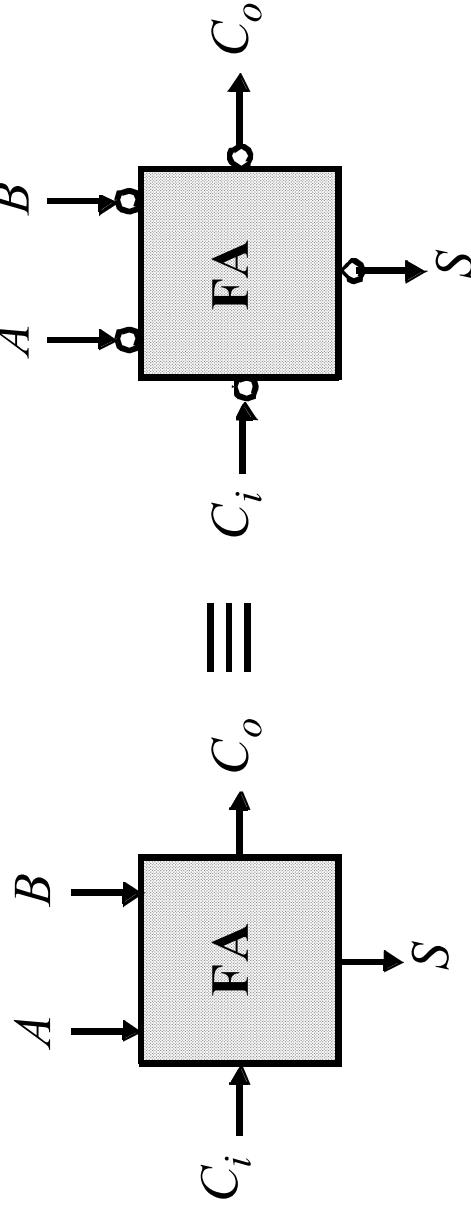
Block Diagram



Notes

- ❖ Faster than the Serial Adder
- ❖ Highly Modular scheme
- ❖ $T_c = \text{Worst Case Carry propagation delay for a 1-bit Full Adder}$
- ❖ $T_{\text{tot}} = N T_c$
- ❖ Required FA circuit with reduced T_c

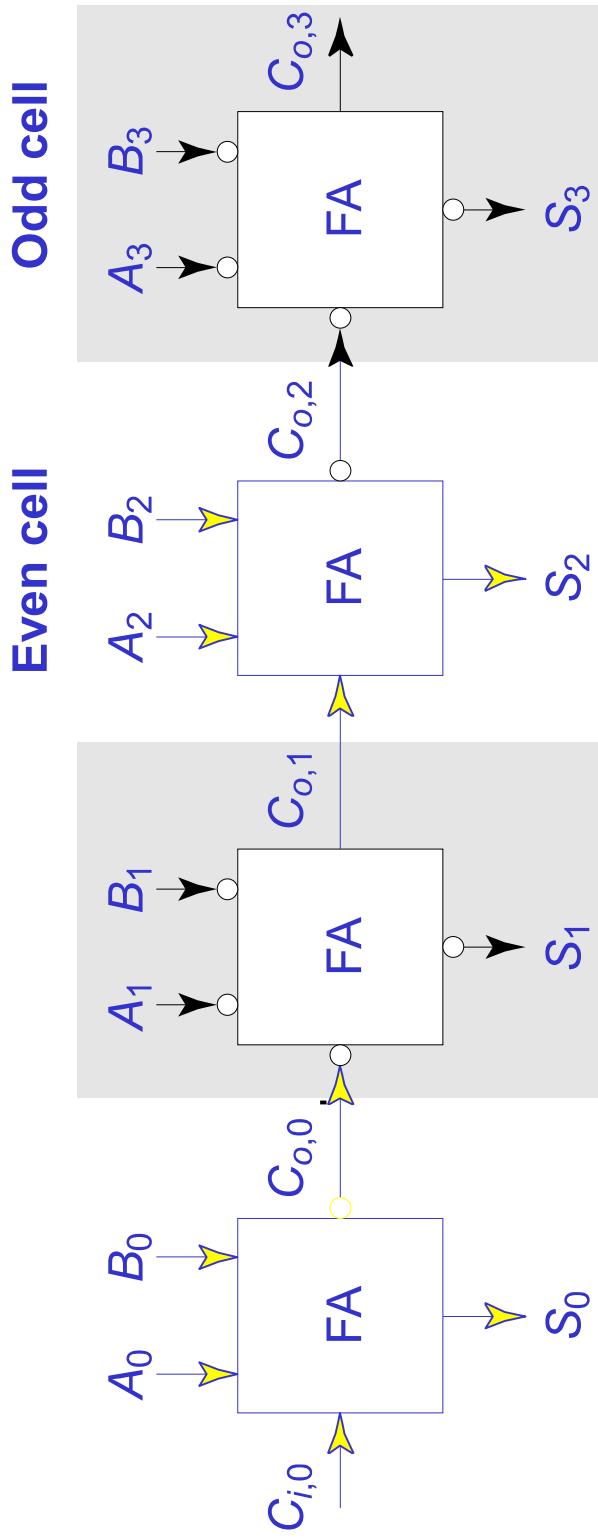
Inversion Property



$$\bar{S}(A, B, C_i) = S(\bar{A}, \bar{B}, \bar{C}_i)$$

$$\bar{C}_o(A, B, C_i) = C_o(\bar{A}, \bar{B}, \bar{C}_i)$$

Minimize Critical Path by Reducing Inverting Stages



Exploiting Inversion Property

Outline

- ❖ Full Adder
- ❖ Serial Adder
- ❖ Parallel Adder
 - Ripple Carry Adder
 - Carry Look Ahead Adder
- ❖ Pipeline
- ❖ Subtractor
- ❖ ALU
- ❖ Multiplier

Notes on CARRY Generation

- ❖ From Full Adder Truth Table
 - ❖ $C_i = A_i B_i C_{i-1} + A_i B_i \overline{C}_{i-1} + A_i \overline{B}_i C_{i-1} + \overline{A}_i B_i C_{i-1}$
 - ❖ $= A_i B_i + C_{i-1} (A_i \oplus B_i)$
 - ❖ Meaning
 - $A_i B_i = 1$ Carry Generate
 - $A_i \oplus B_i = 1$ Carry Propogate
- $G_i = A_i B_i$ $P_i = A_i \oplus B_i$

Notes on CARRY Generation

$$\star C_i = A_i B_i C_{i-1} + A_i B_i \overline{C}_{i-1} + A_i \overline{B}_i C_{i-1} + \overline{A}_i B_i C_{i-1}$$

$$\star = A_i B_i + C_{i-1} (A_i \oplus B_i)$$

$$\boxed{P_i = A_i \oplus B_i}$$

$$\star G_i = A_i B_i$$

$$\star P_i = A_i + B_i$$

A	B	XOR	OR
0	0	0	0
0	1	1	1
1	0	1	1
1	1	0	1

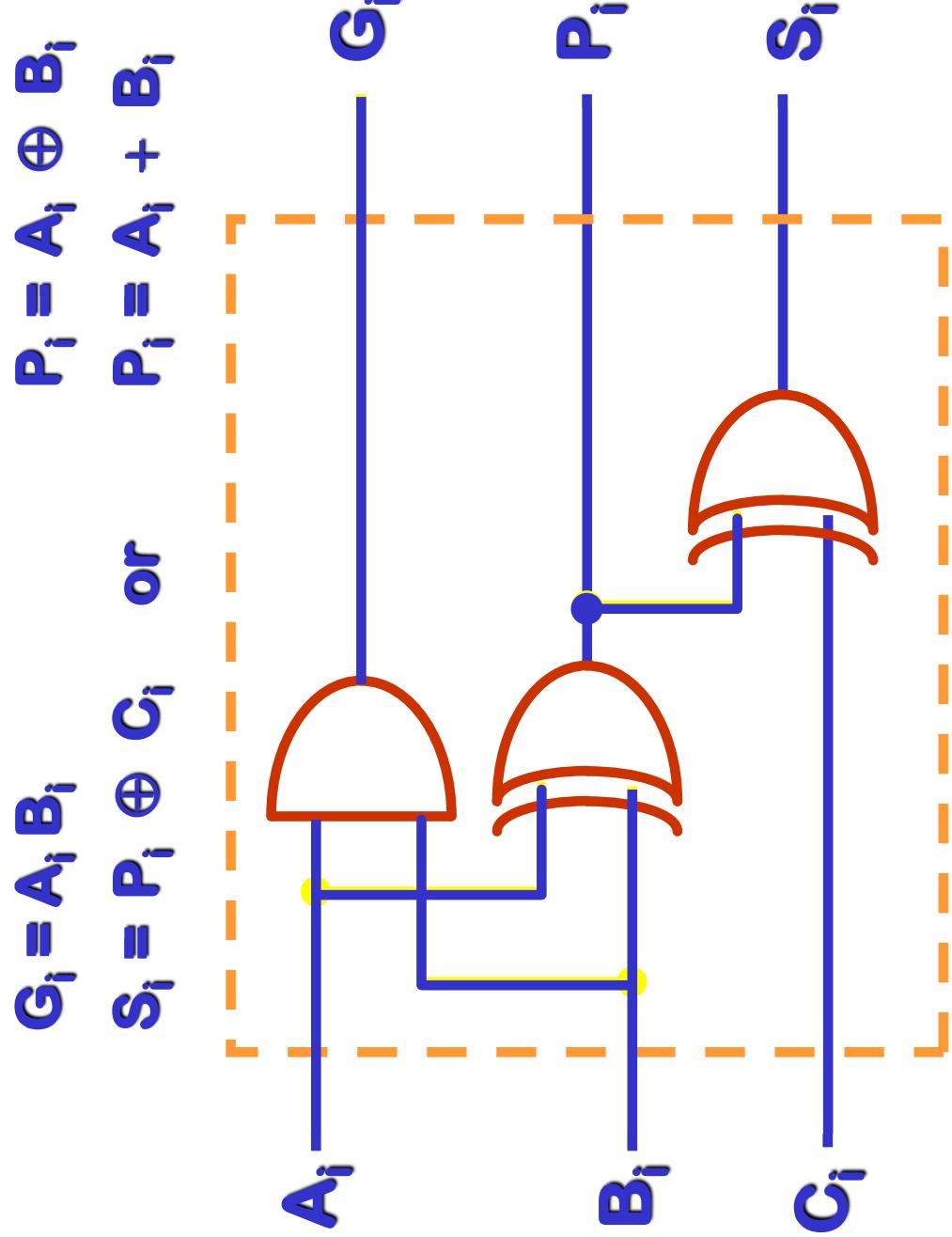
Carry Look -Ahead Adder

So...

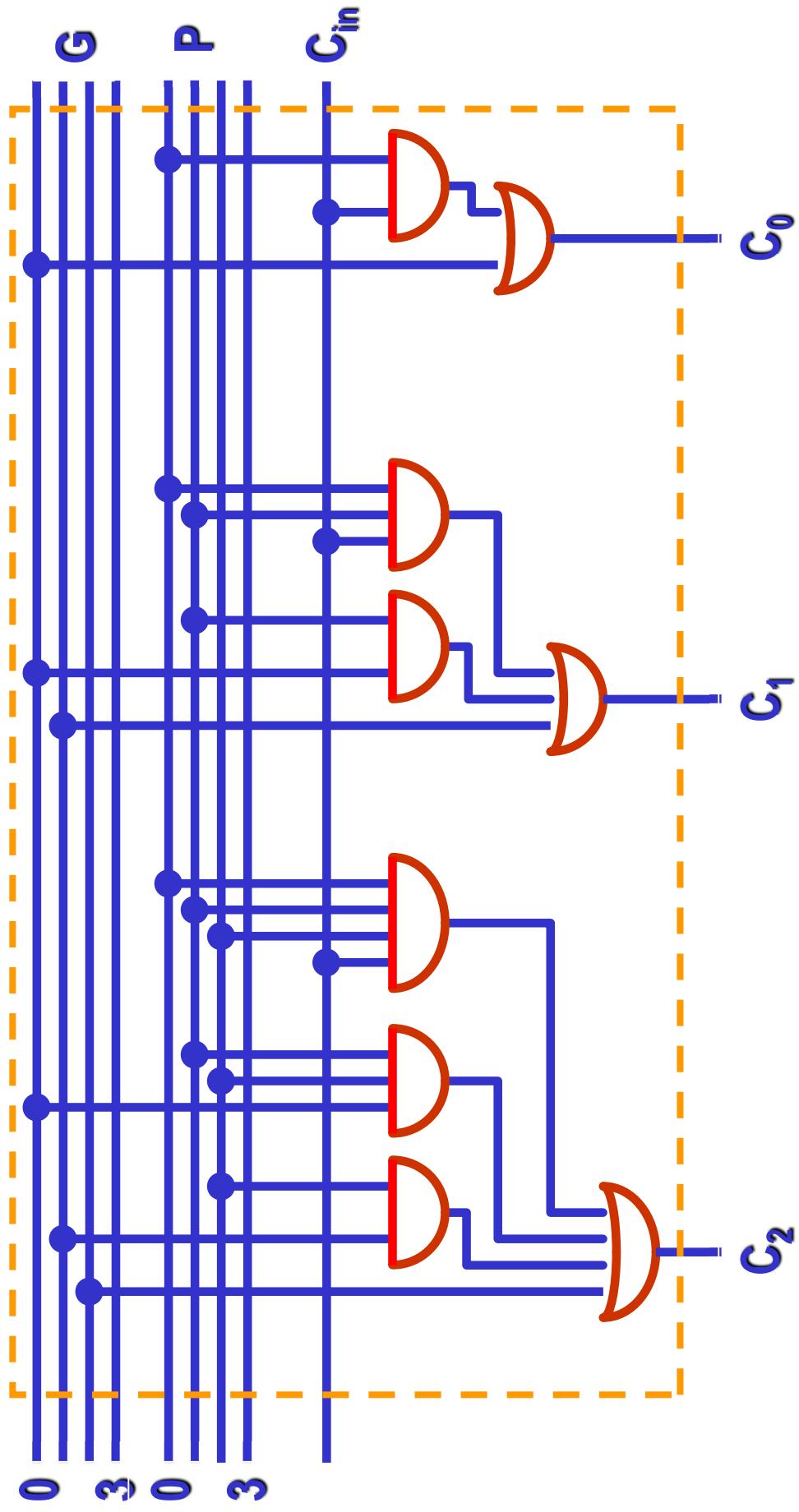
- $C_0 = G_0 + P_0 C_{in}$
- $C_1 = G_1 + P_1 C_0 = G_1 + P_1 G_0 + P_1 P_0 C_{in}$
- $C_2 = G_2 + P_2 C_1 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_{in}$
- $C_3 = G_3 + P_3 C_2 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 P_0 G_0 +.$

All Carry Signals can be generated at
the same time

Basic block

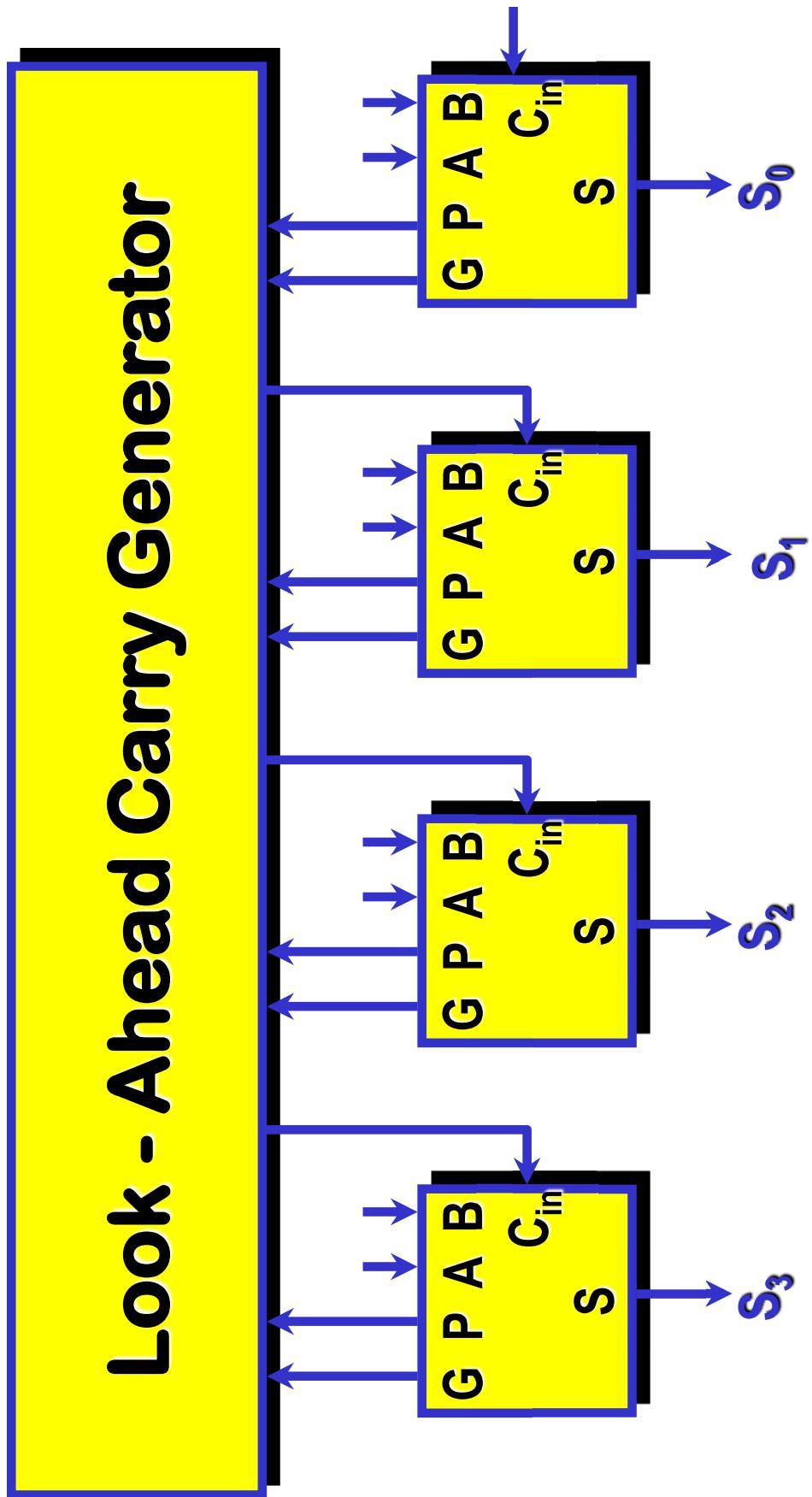


Look - Ahead Carry Generator Scheme



Adder Scheme

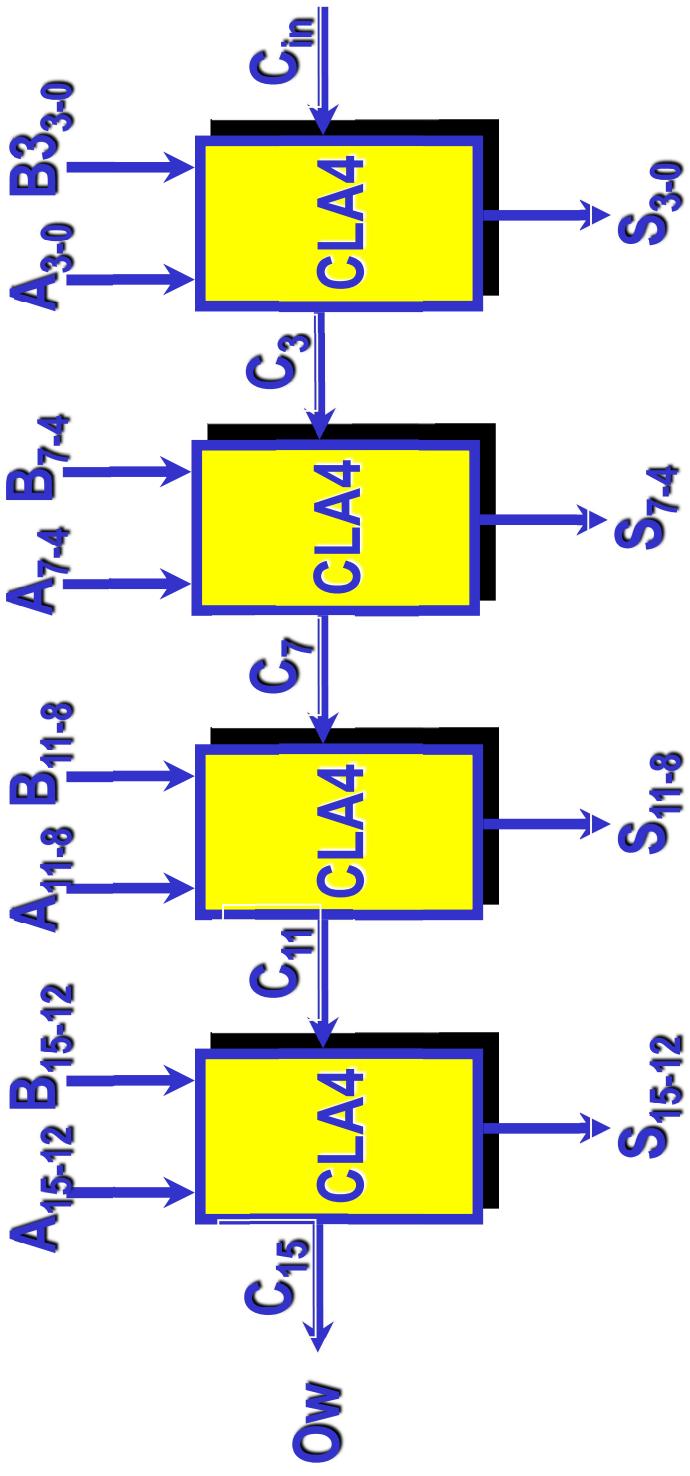
Look - Ahead Carry Generator



Notes

- ❖ **Fast Adder w.r.t. Serial One**
- ❖ **Requires wide BUS (G and P)**
- ❖ **Requires AND gates with high fan-in**
- ❖ **For N greater than 4:**
 - CLA4 ripple carry solution
 - CLA Hierarchical solution

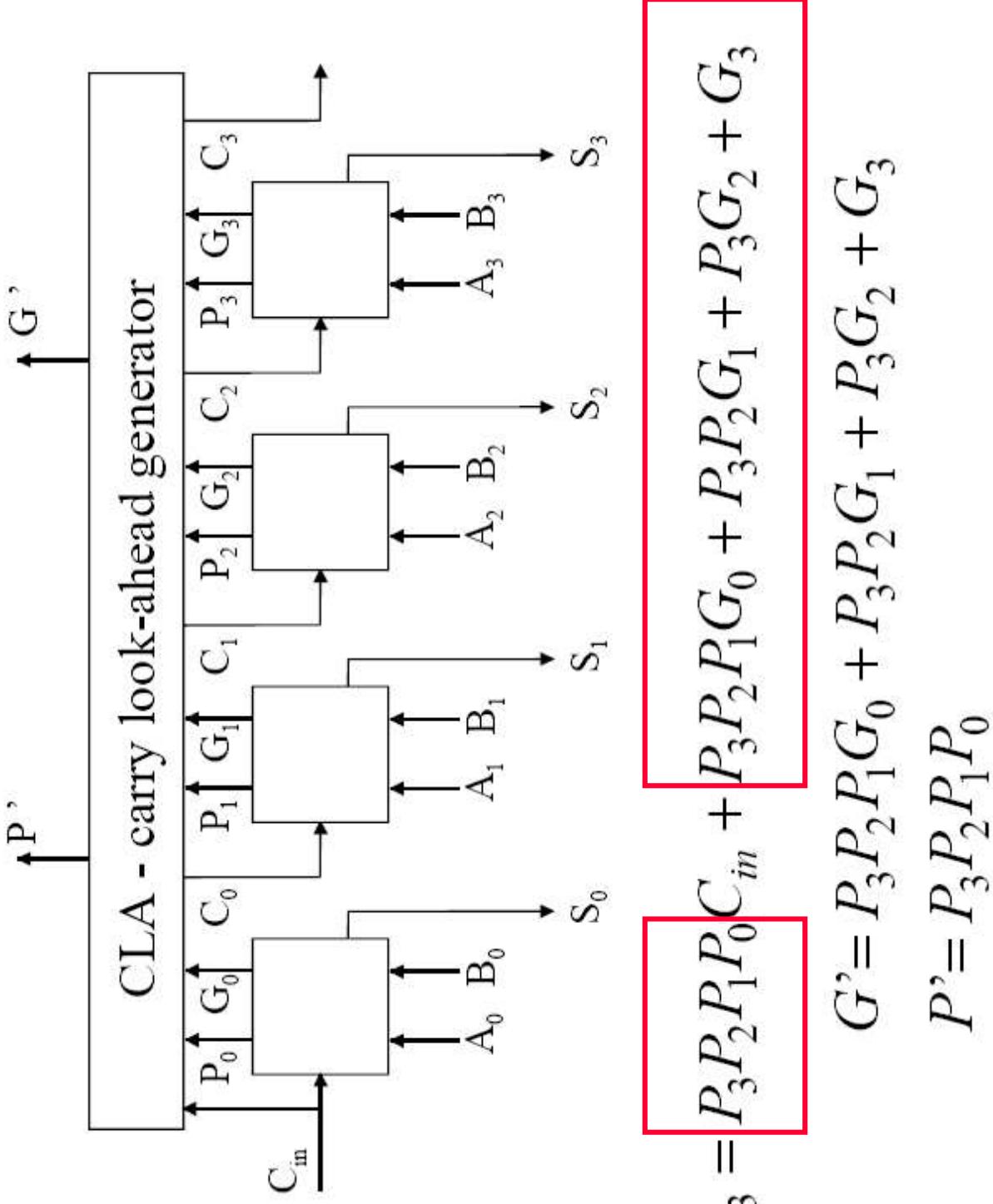
CLA4 ripple carry solution



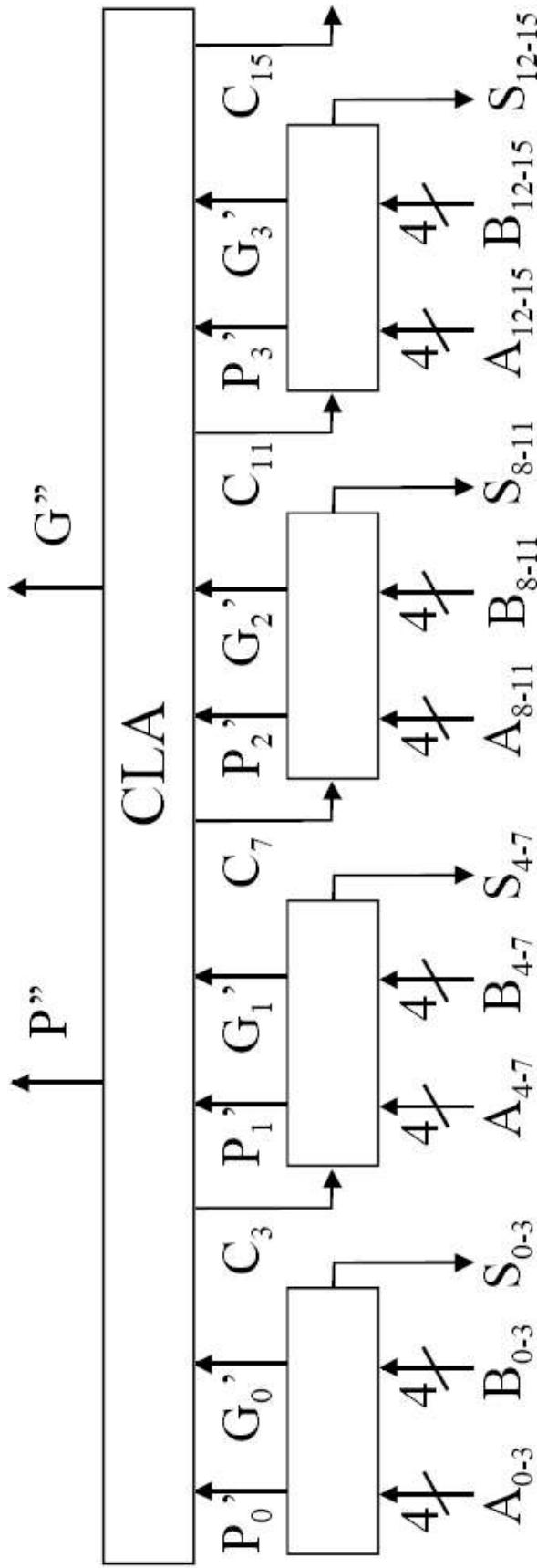
$T_{CLA4} = \text{CLA4 propagation delay}$

$T_{tot} = (N/4) T_{CLA4}$

Hierarchical CLA solution



Soluzione CLA gerarchica (N=16)



$T_{CLA4} = CLA4 \text{ propagation delay}$

$T_{tot} = \log_4(N) T_{CLA4} = 2 T_{CLA4}$

Comparison for N=64

$T_{CLA4} = CLA4$ propagation delay

$T_c = WC$ Carry propagation delay for a 1-bit Full Adder

Ripple Carry Adder: $T_{tot} = N T_c = 64 T_c$

CLA4- Ripple: $T_{tot} = N/4 T_{CLA4} = 16 T_{CLA4}$

CLA hierarchical: $T_{tot} = \log_4(N) T_{CLA4} = 3 T_{CLA4}$

For CMOS 0,25 we have $T_c \sim 0,5$ ns and $T_{CLA4} \sim 1$ ns

RPA 32 ns

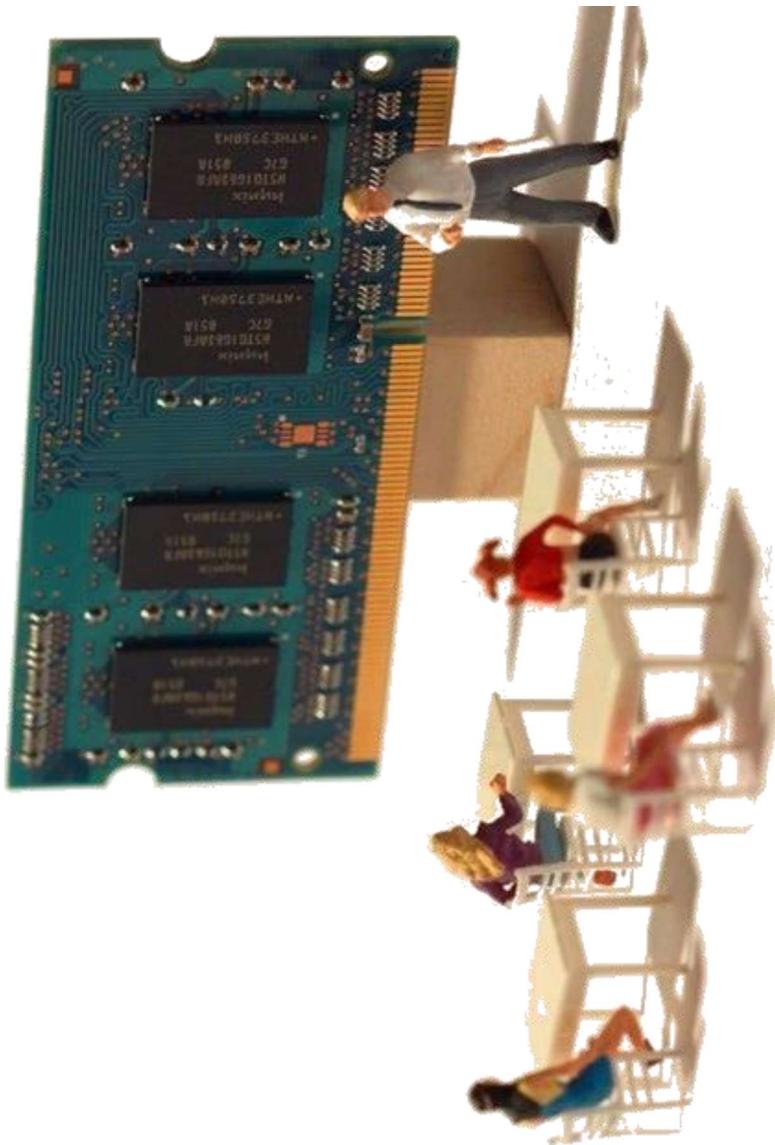
CLA4-R 16 ns

CLA4-G 3 ns

$T_{CK} \geq t_{c-q} + t_{\text{plogic}} + t_{su}$

End, Questions ?

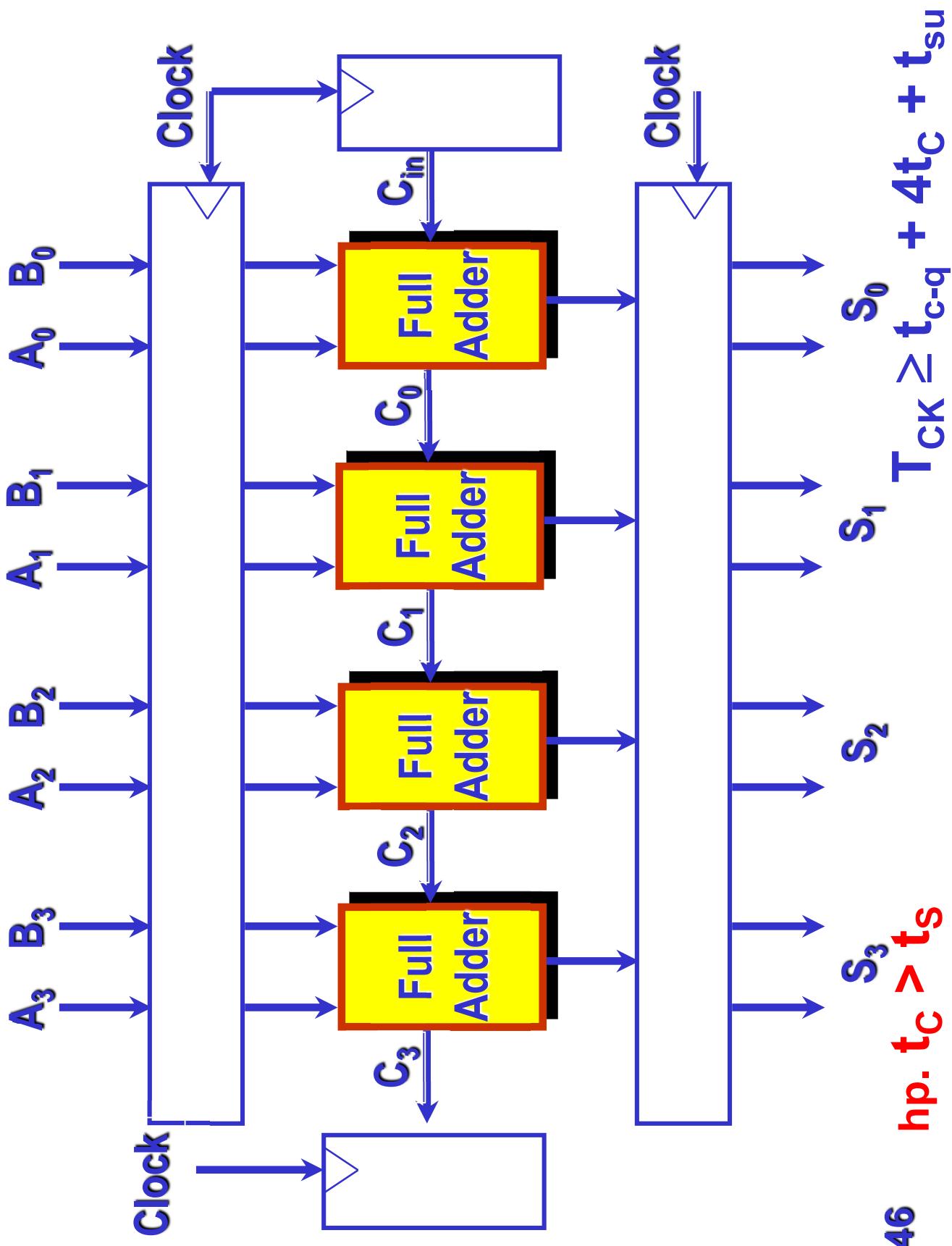
- Full Adder
- Serial Adder
- Parallel Adder
- Ripple Carry Adder
- Carry Look Ahead Adder



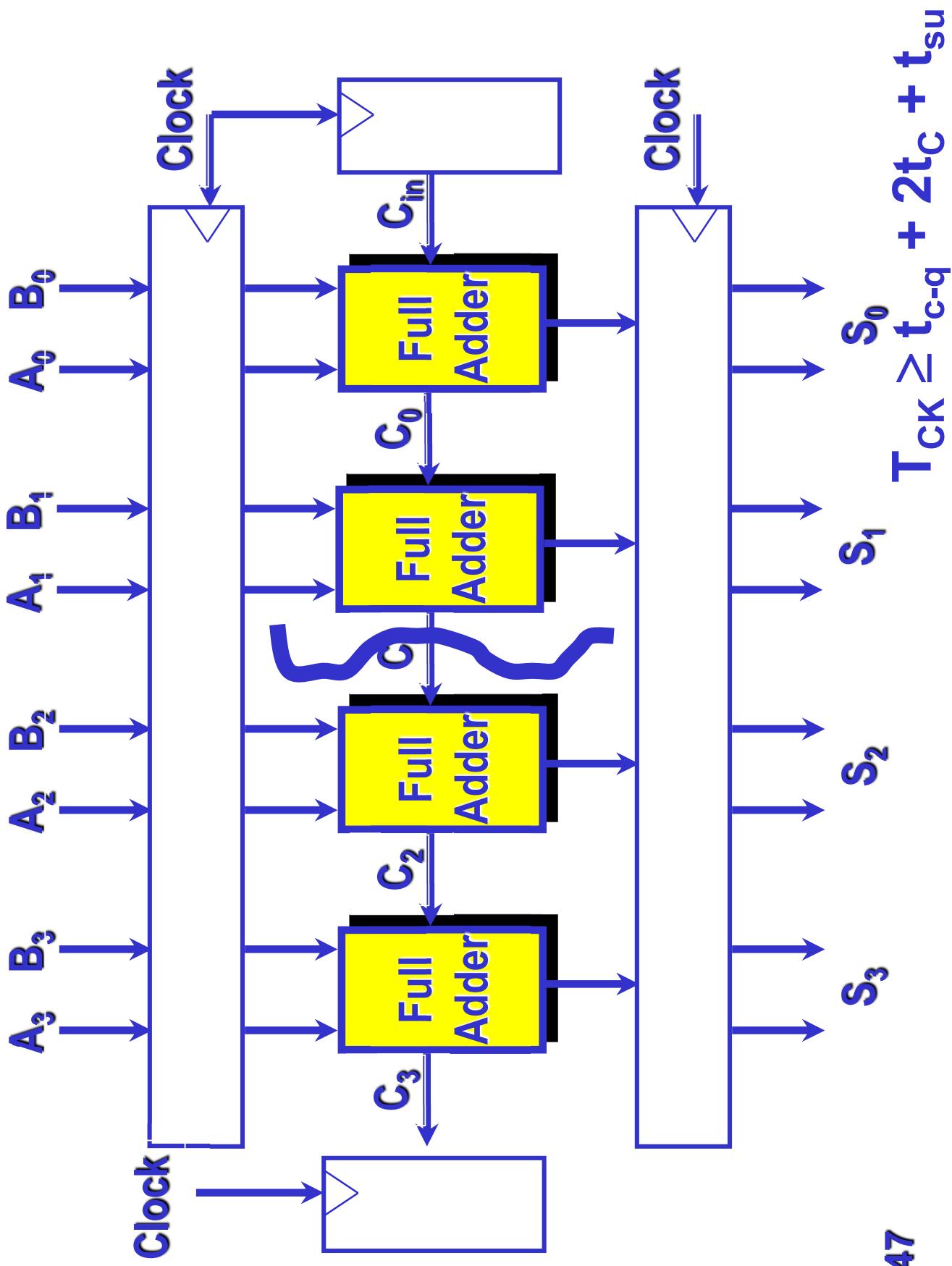
Outline

- ❖ **Full Adder**
- ❖ **Serial Adder**
- ❖ **Parallel Adder**
 - **Ripple Carry Adder**
 - **Carry Look Ahead Adder**
- ❖ **Pipeline**
- ❖ **Subtractor**
- ❖ **ALU**
- ❖ **Multiplier**

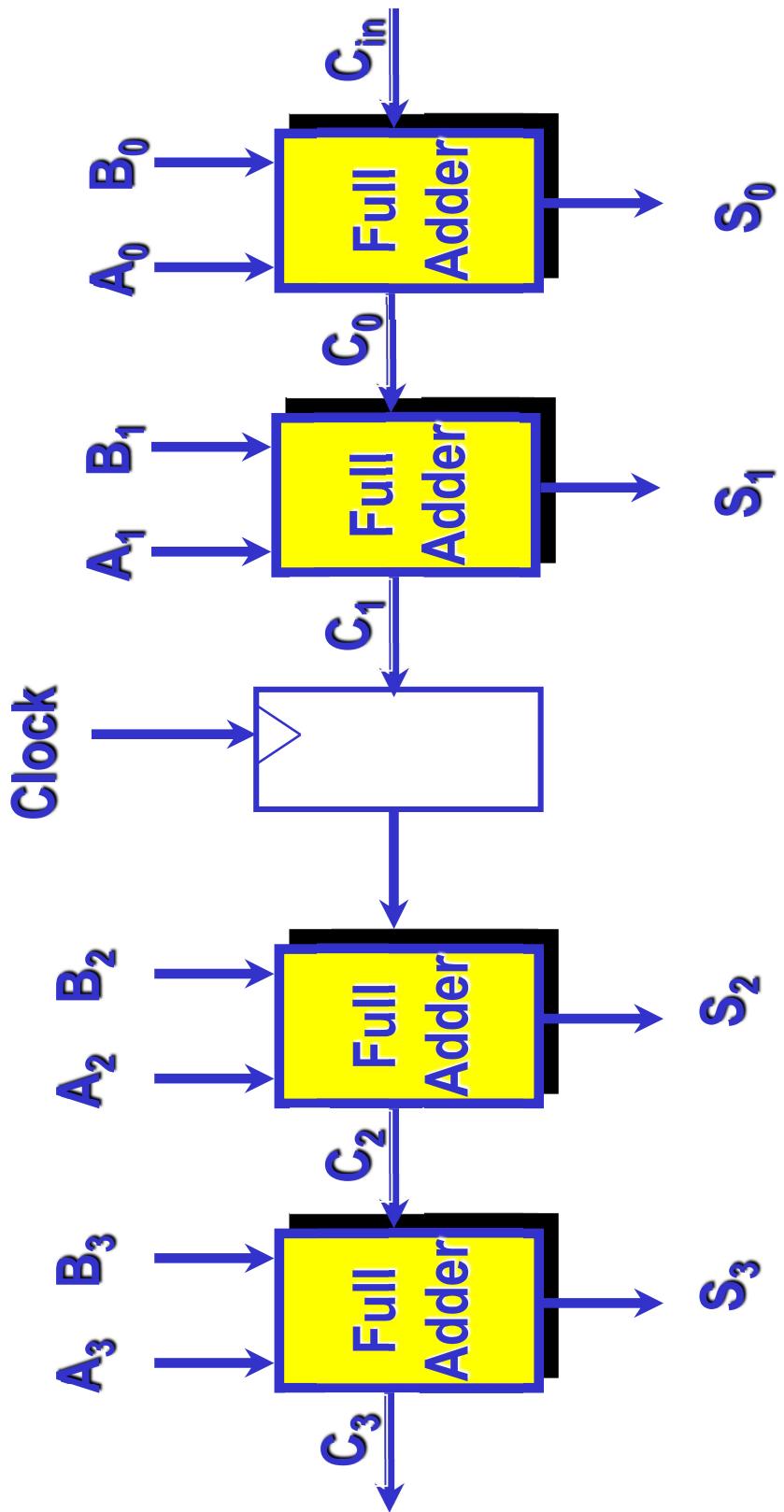
Pipeline Solutions



Pipeline Solutions

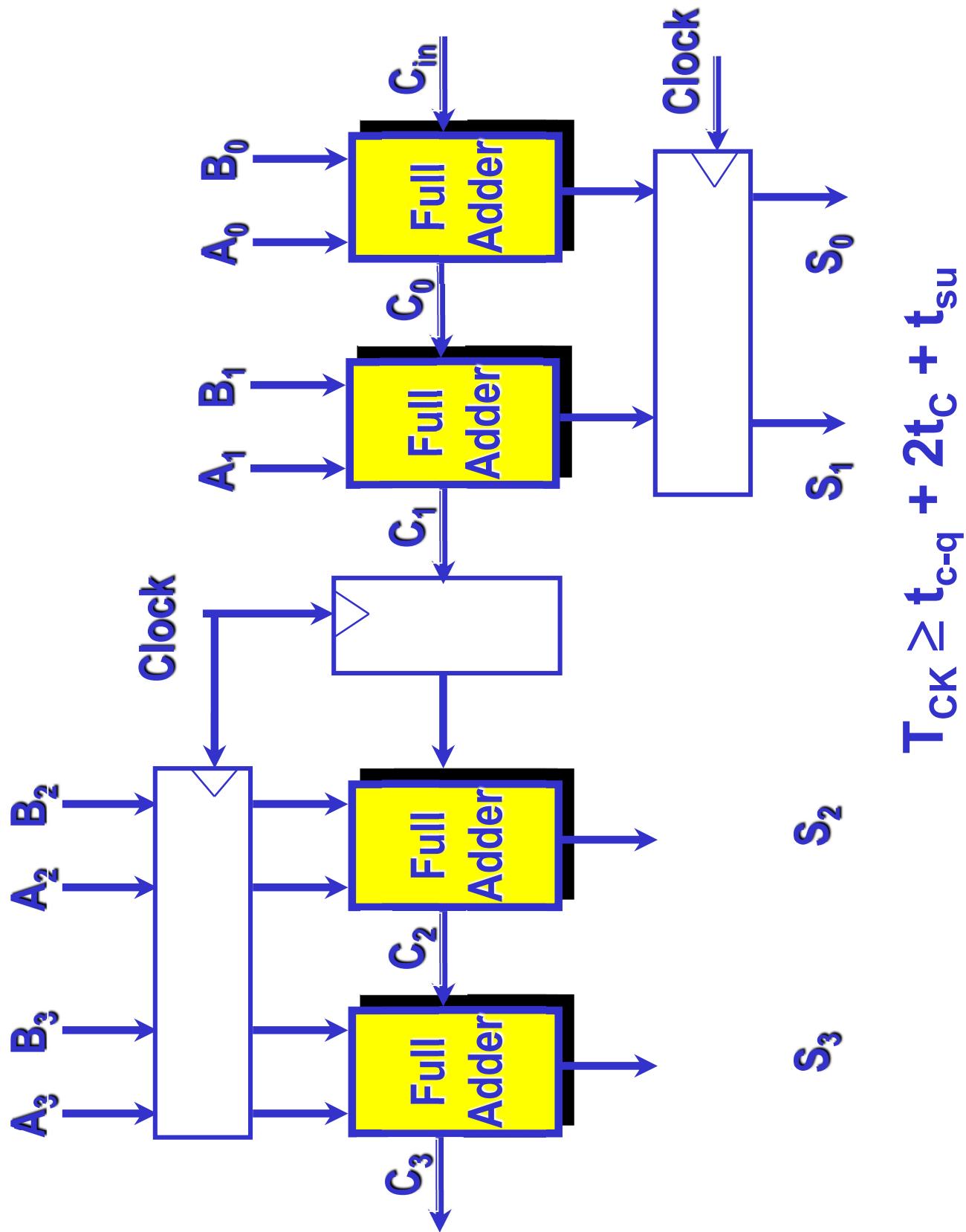


Pipeline Solutions



$$T_{CK} \geq t_{c-q} + 2t_C + t_{su}$$

Pipeline Solutions



Outline

- ❖ **Full Adder**
- ❖ **Serial Adder**
- ❖ **Parallel Adder**
 - **Ripple Carry Adder**
 - **Carry Look Ahead Adder**
- ❖ **Pipeline**
- ❖ **Subtractor**
- ❖ **ALU**
- ❖ **Multiplier**

2 complement Subtractor

★ Subtractor

$$D = A - B = A + (-B)$$

★ In 2's complement

B^* = bit-to-bit complement of B

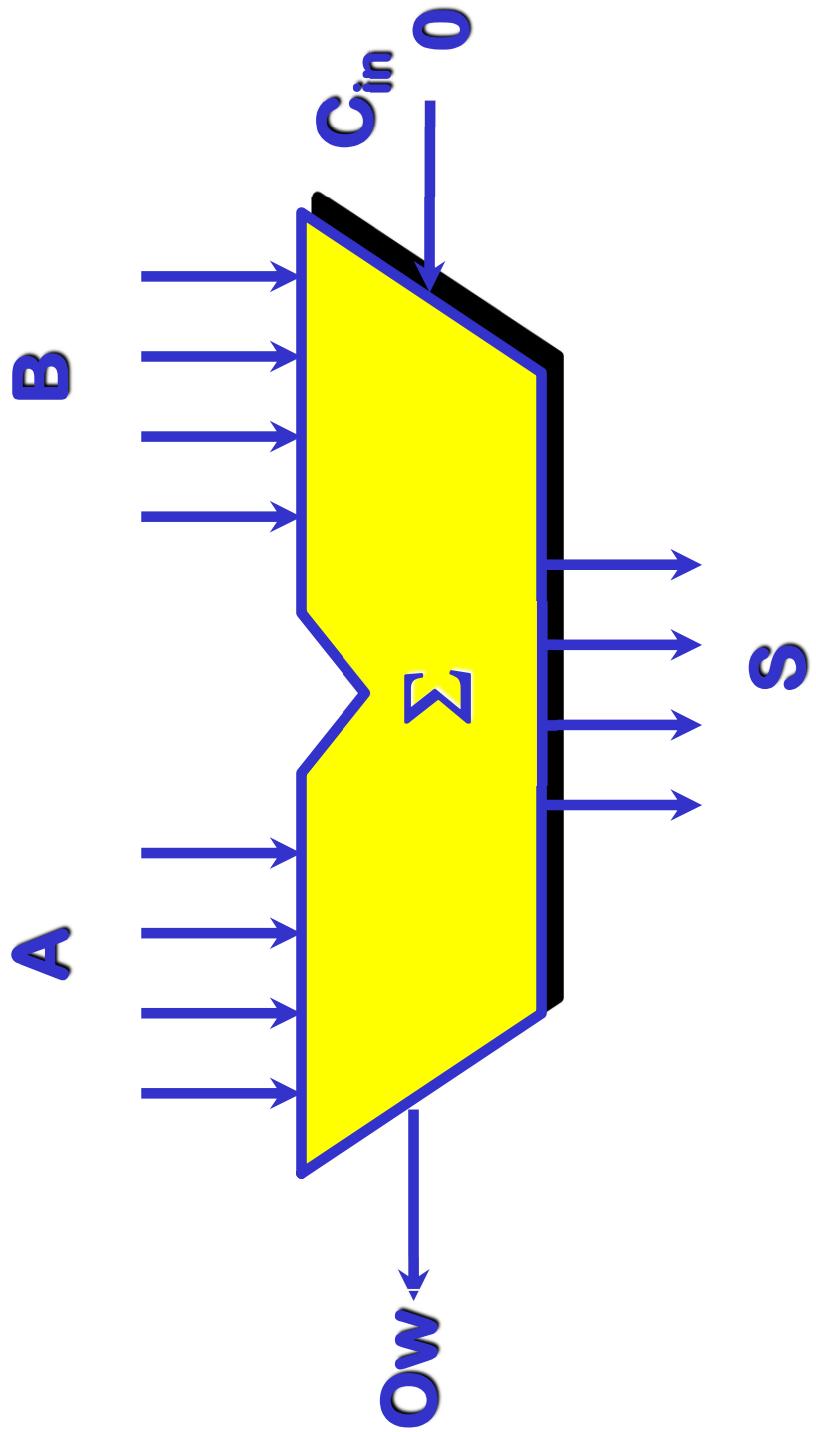
$$B = 01101 \quad B^* = 10010$$

★ $-B = B^* + 1$

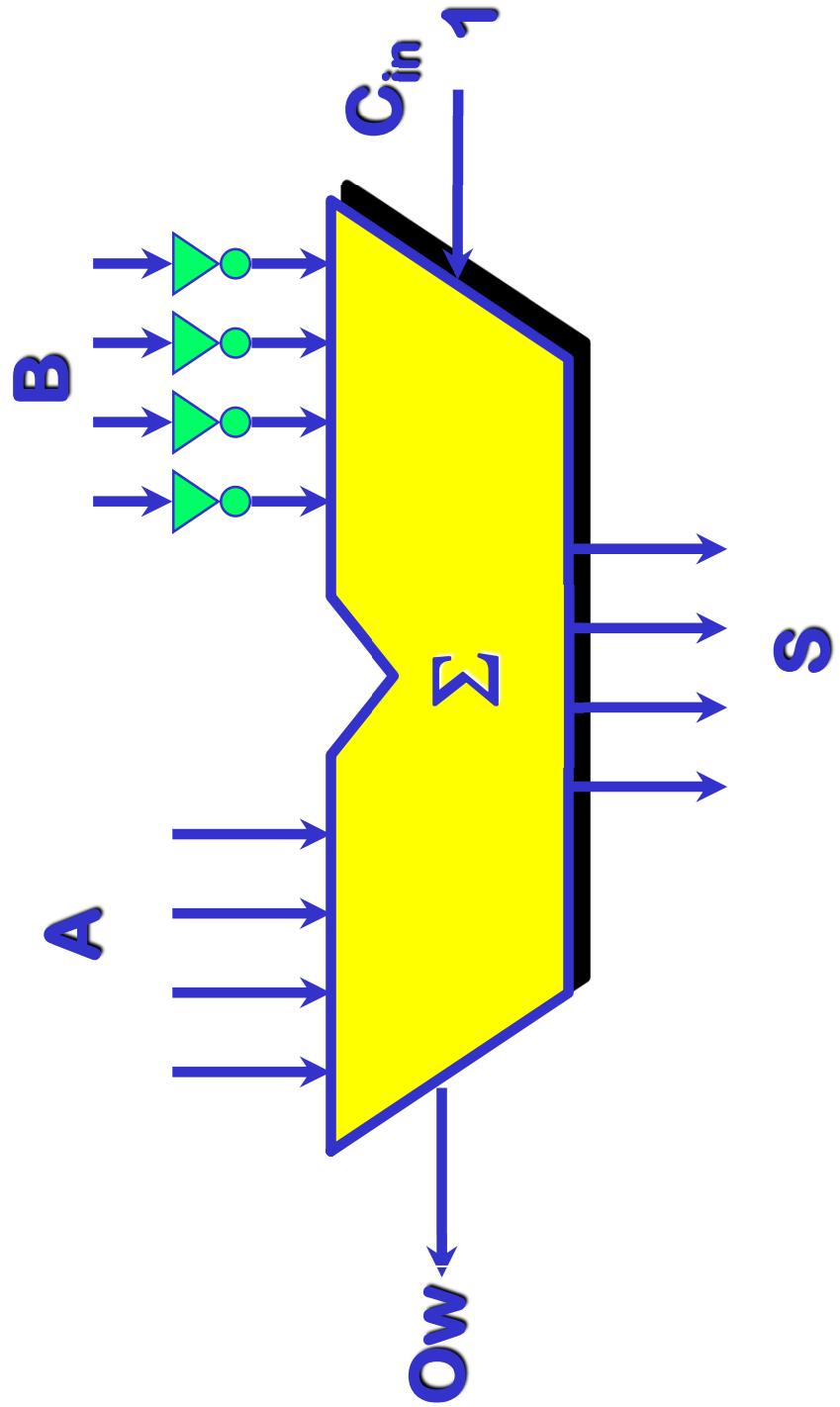
$$D = A - B = A + B^* + 1$$

Adder

★ Adding two 4-bit words



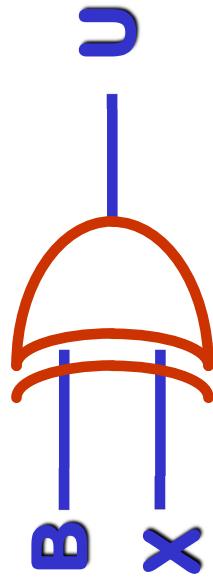
Subtractor



Notes on exclusive OR

★ Truth Table

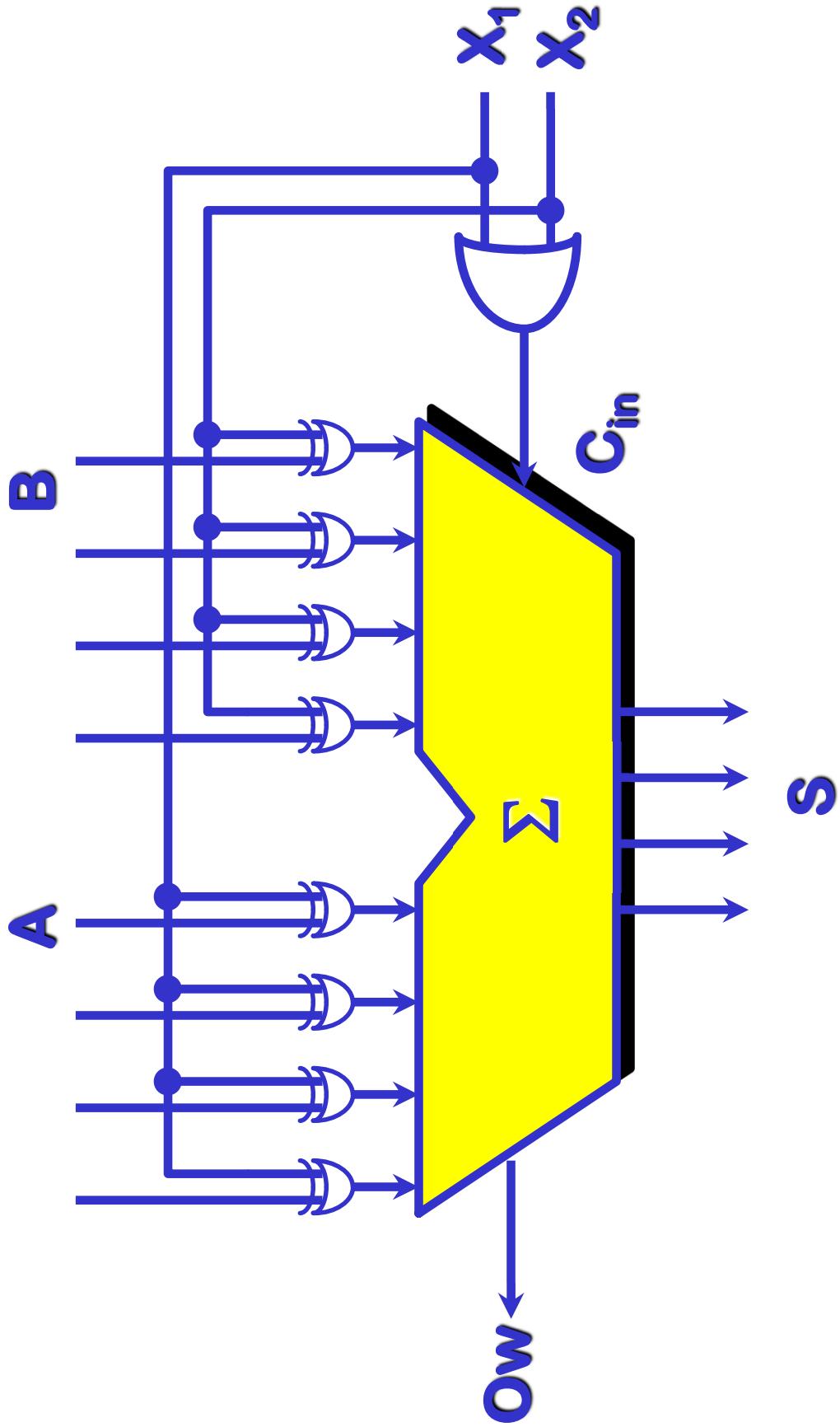
B	X	U
0	0	0
1	0	1
0	1	1
1	1	0



★ For $X = 0$ results $U = B$

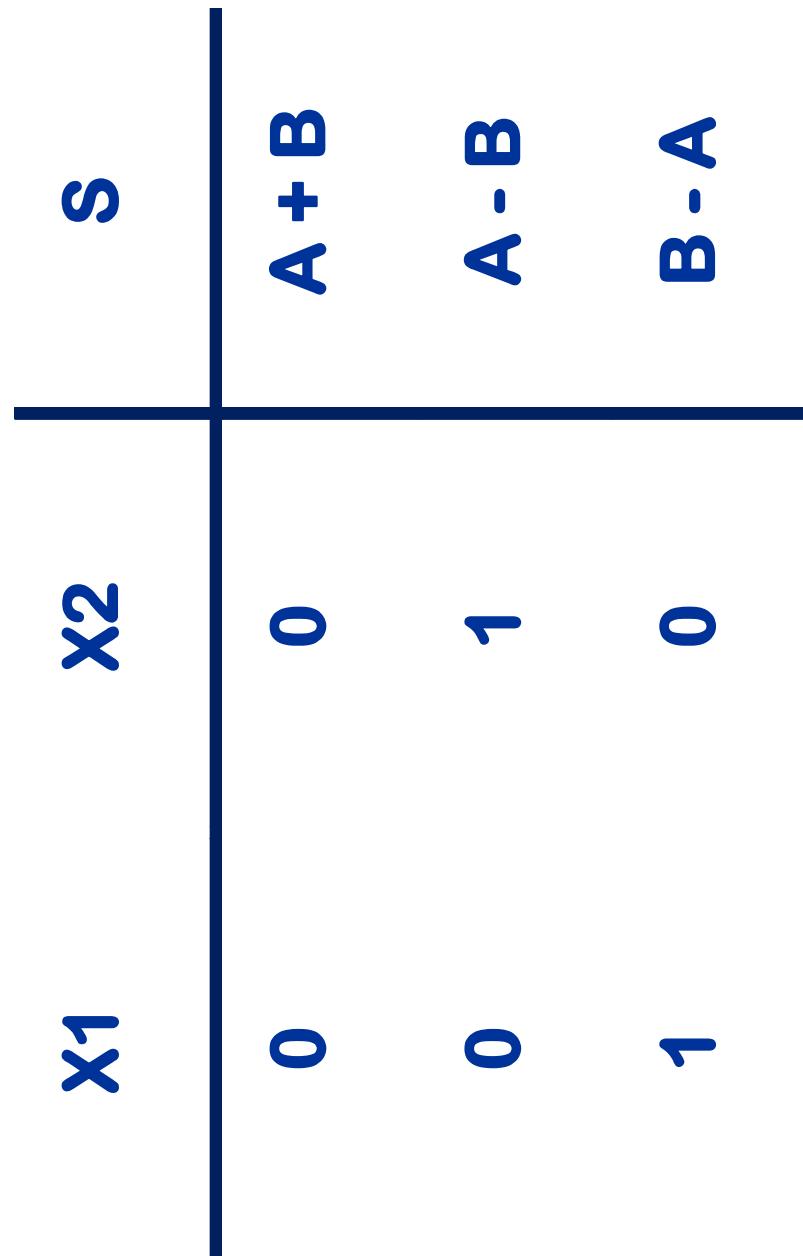
★ For $X = 1$ results $U = B^*$

Adder - Subtractor



Control variables

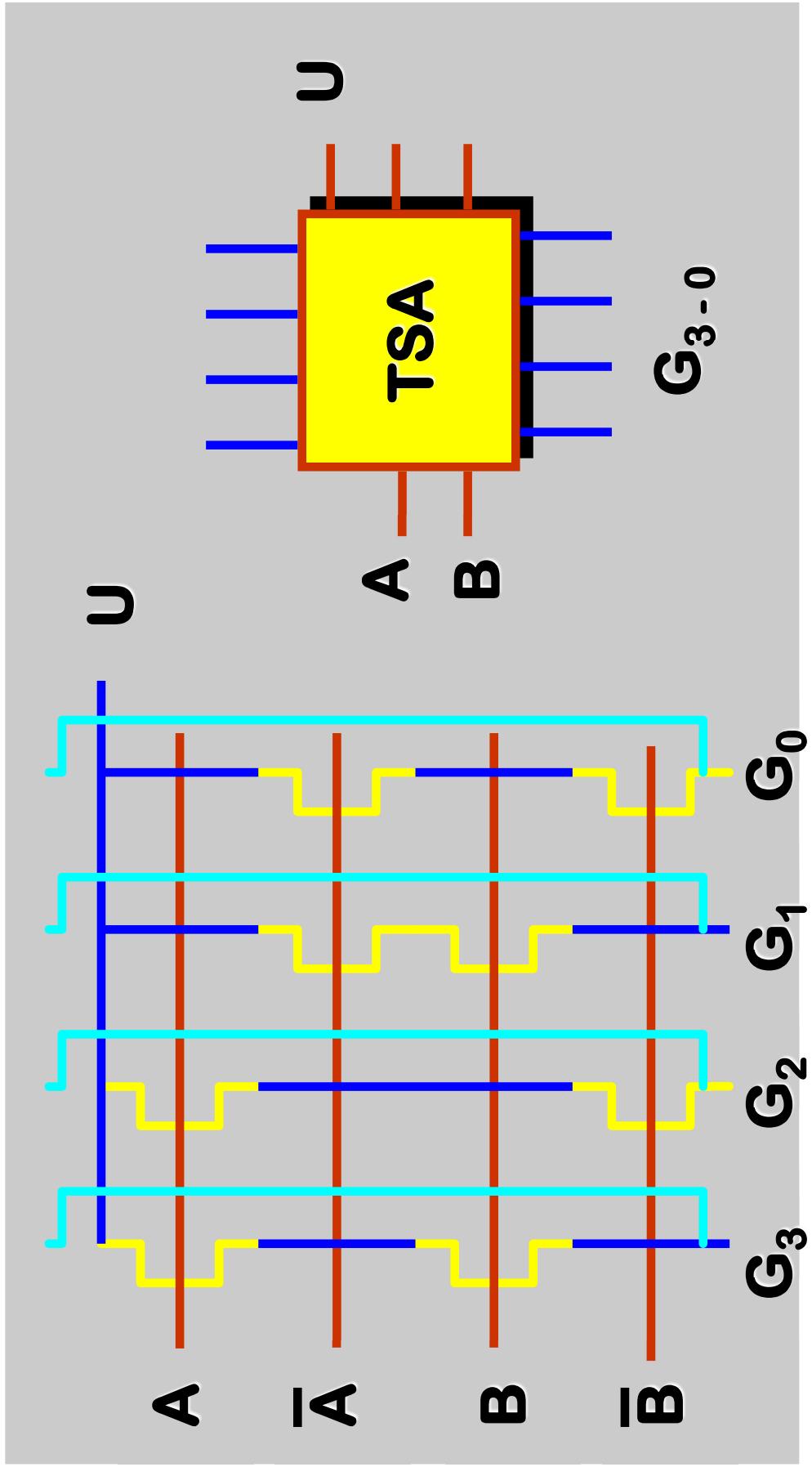
★ Coding Table



Outline

- ❖ Full Adder
- ❖ Serial Adder
- ❖ Parallel Adder
 - Ripple Carry Adder
 - Carry Look Ahead Adder
- ❖ Pipeline
- ❖ Subtractor
- ❖ ALU
- ❖ Multiplier

Programmable TSA

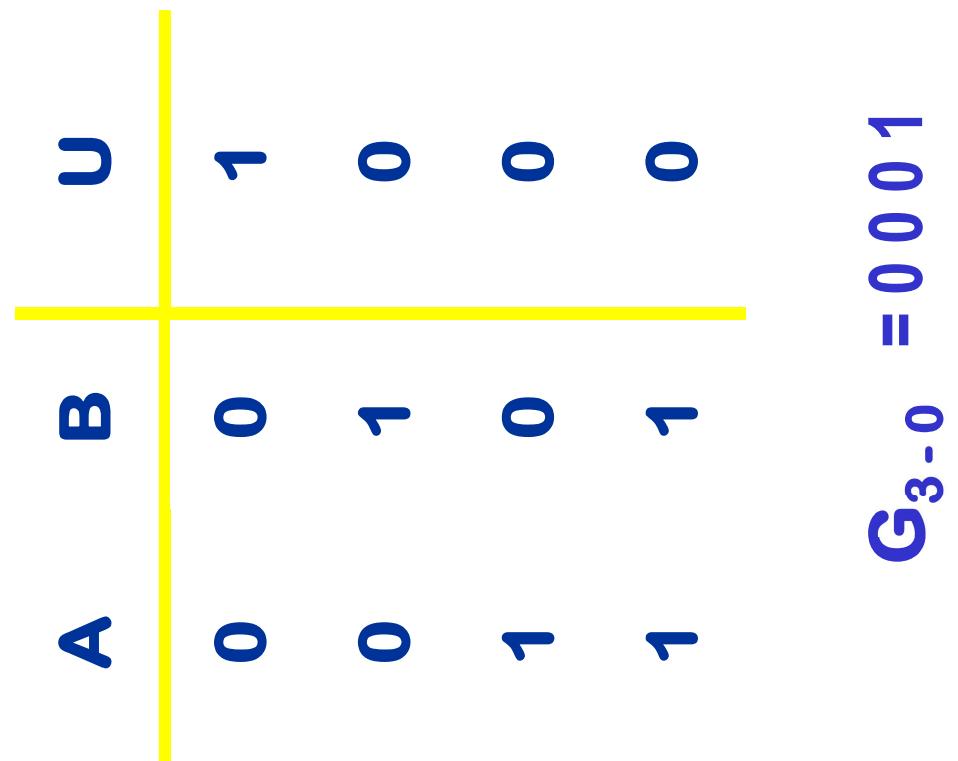


Funzione logica NOR

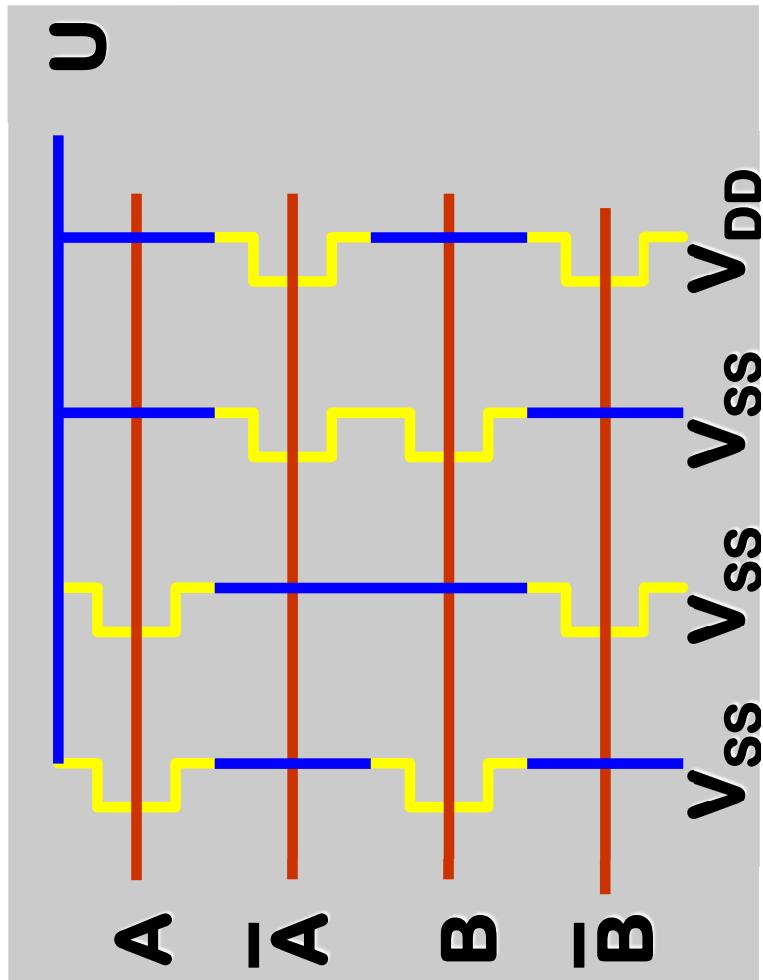
☆ L'uscita va a livello logico alto solo se

$$\overline{A} = 1 \text{ e } \overline{B} = 1$$

$$\star U = \overline{A} * \overline{B} = \overline{(A + B)}$$



$$G_{3-0} = 0001$$



Funzioni realizzabili con la PTL 1/4

★ Tabella di programmazione 1

HEX	G ₃	G ₂	G ₁	G ₀	U
0	0	0	0	0	0
1	0	0	0	1	$\overline{A + B}$
2	0	0	1	0	$\overline{A \cdot B}$
3	0	0	1	1	\overline{A}

Funzioni realizzabili con la PTL 2/4

⇒ Tabella di programmazione 2

HEX	G ₃	G ₂	G ₁	G ₀	U
4	0	1	0	0	$A \cdot \overline{B}$
5	0	1	0	1	\overline{B}
6	0	1	1	0	$A \oplus B$
7	0	1	1	1	$\overline{A \cdot B}$

Funzioni realizzabili con la PTL 3/4

⇒ Tabella di programmazione 3

HEX	G ₃	G ₂	G ₁	G ₀	U
8	1	0	0	0	A • B
9	1	0	0	1	$\overline{A \oplus B}$
A	1	0	1	0	B
B	1	0	1	1	$\overline{A + B}$

Funzioni realizzabili con la PTL 4/4

⇒ Tabella di programmazione 4

HEX	G ₃	G ₂	G ₁	G ₀	U
C	1	1	0	0	A
D	1	1	0	1	$\overline{A + B}$
E	1	1	1	0	A + B
F	1	1	1	1	1

Logic Functions

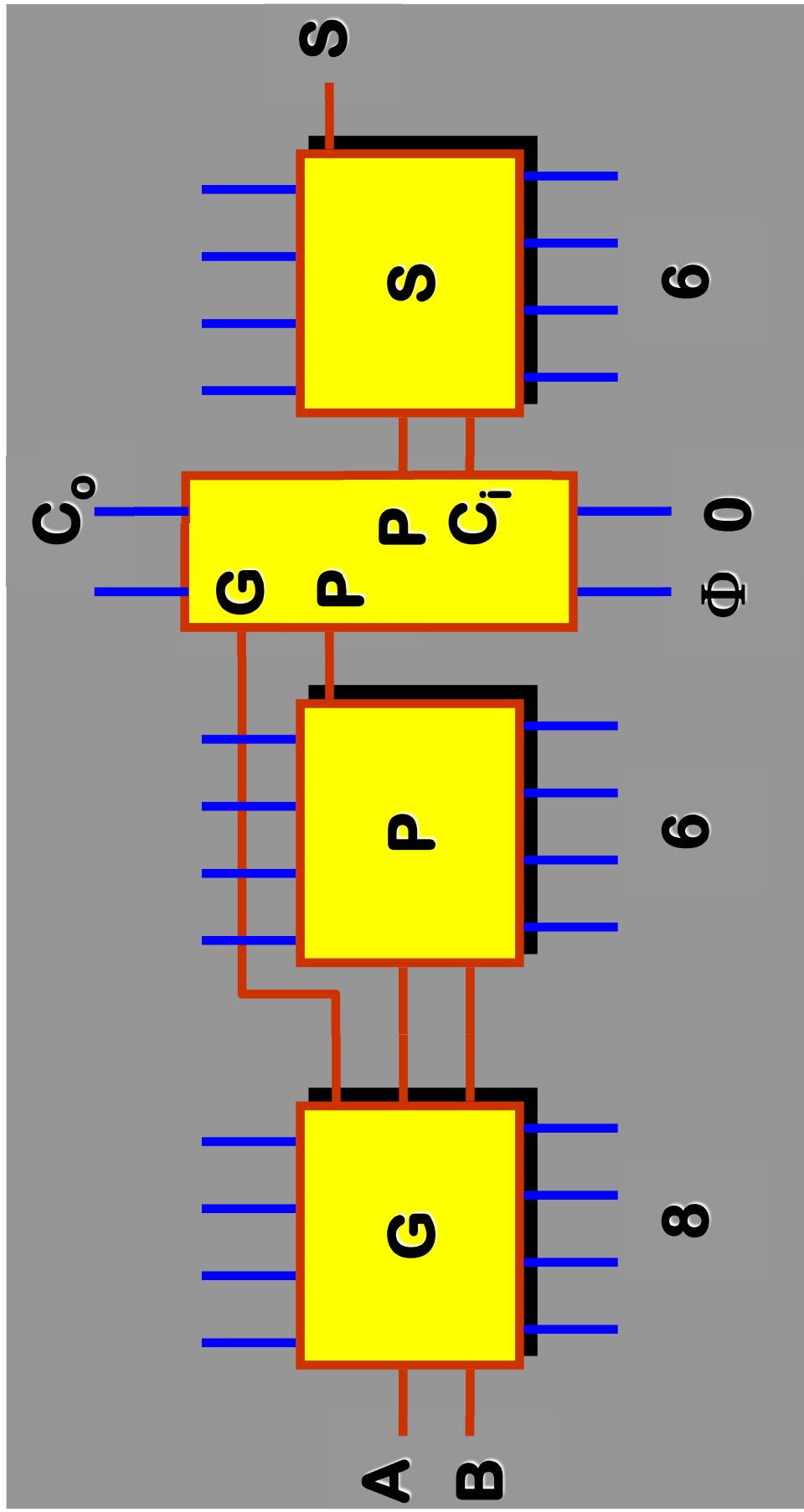
★ Programming Table

HEX	G ₃	G ₂	G ₁	G ₀	U
8	1	0	0	0	$A \cdot B$
5	0	1	0	1	\overline{B}
6	0	1	1	0	$A \oplus B$
7	0	1	1	1	$\overline{A \cdot B}$

$$G_i = A_i B_i$$
$$S_i = P_i \oplus C_i \quad \text{or}$$

$$P_i = A_i \oplus B_i$$
$$P_i = A_i + B_i$$

ALU

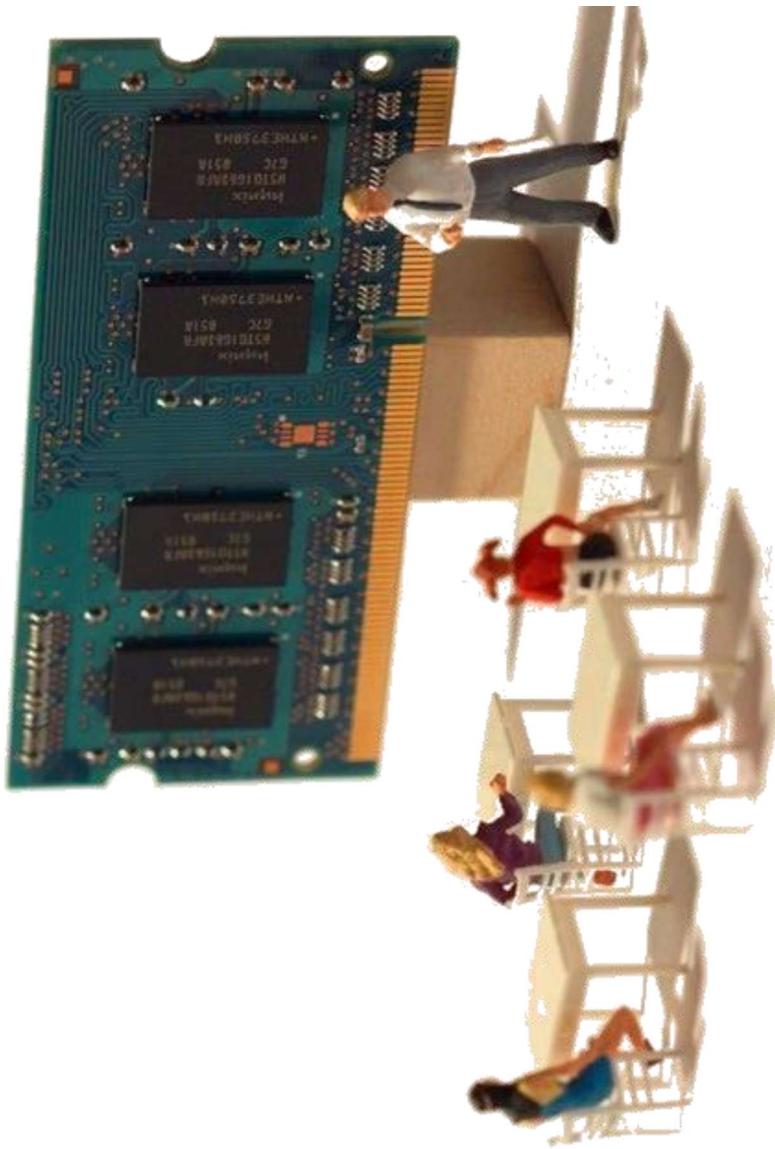


Notes

- N-bit ALU is obtained by using N blocks as the previous one
- Multiple functions can be obtained by changing programming bits
 - Adder, Subtractor, AND, OR, XOR,
- The programming bits represents the «microcode» for the ALU

End, Questions ?

- Full Adder
- Serial Adder
- Parallel Adder
- Ripple Carry Adder
- Carry Look Ahead Adder
- Pipeline
- Subtractor
- ALU



Outline

- ❖ Full Adder
- ❖ Serial Adder
- ❖ Parallel Adder
 - Ripple Carry Adder
 - Carry Look Ahead Adder
 - Manchester Carry Adder
- ❖ Pipeline
- ❖ Subtractor
- ❖ ALU
- ❖ **Multiplier**
68

Parallel Multiplier

★ Integer Multiplication

$$\star A = 1011 \quad (10)_{10}$$

$$\star B = 0110 \quad (6)_{10}$$

$$\star P = 0111100 \quad (60)_{10}$$

Example

$$\star P = A \cdot B$$

A	B	P
1	0	0
0	1	0
0	0	0
1	1	0
0	0	0
0	1	0
1	0	0
1	1	1
0	0	0
1	1	1
0	1	0
0	0	0

ROM-based implementation

Complexity for a 16×16 MPY

- A = 16 bit, B = 16 bit
- 32 address bit
- Result on 32 bit
 - In total 2^{32} words of 32 bit are required

ROM Size

16 GBYTE !!!

Notes

- ❖ Partial products can be evaluated in parallel
- ❖ Given two words, X and Y, the product P results:

$$X = \sum_{i=0}^{m-1} x_i 2^i \quad Y = \sum_{j=0}^{n-1} y_j 2^j$$

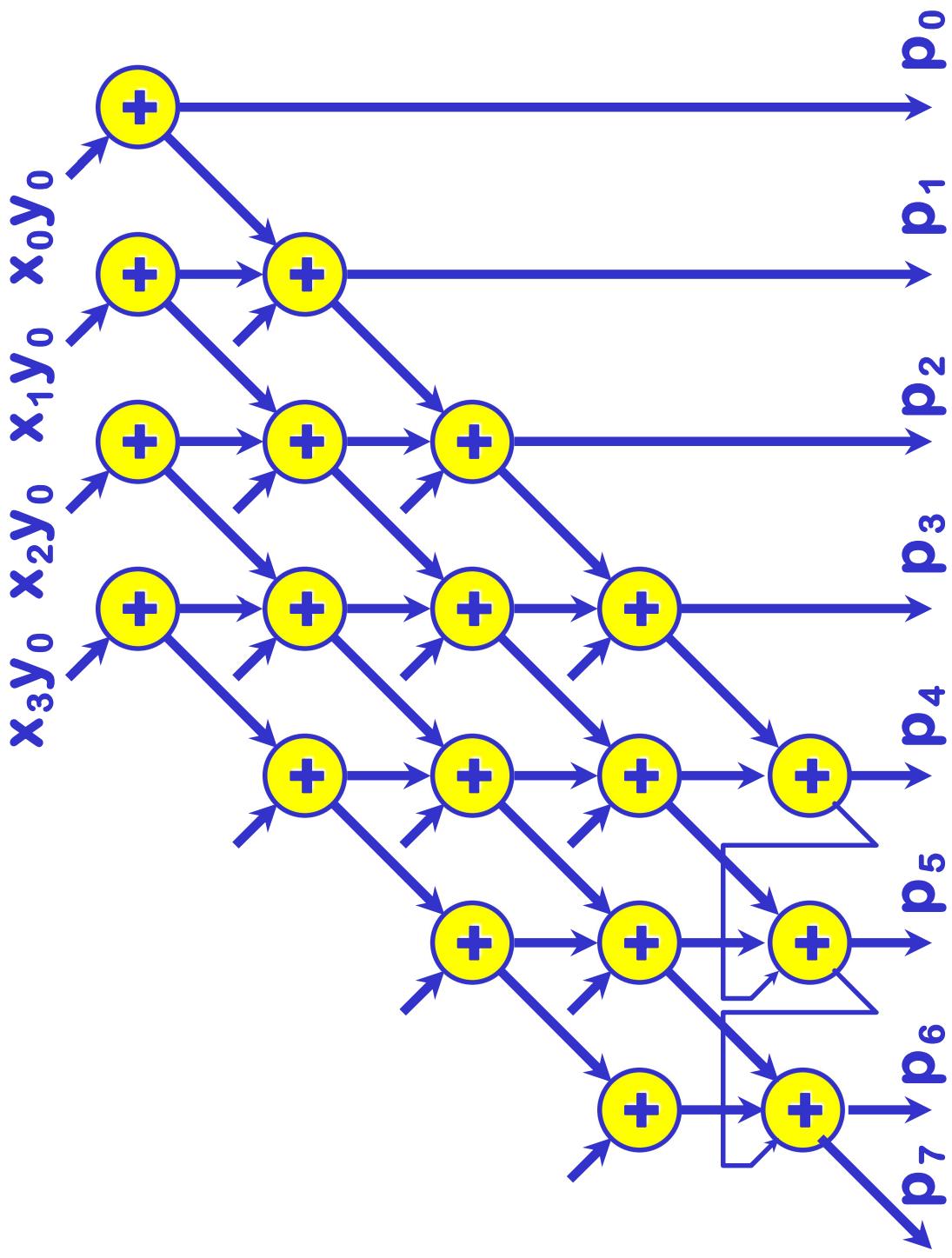
$$P = \sum_{i=0}^{m-1} x_i 2^i \bullet \sum_{j=0}^{n-1} y_j 2^j = \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} (x_i y_j) \bullet 2^{i+j} = \sum_{k=0}^{m+n-1} p_k \bullet 2^k$$

Example

★ Product of two 4-bits words

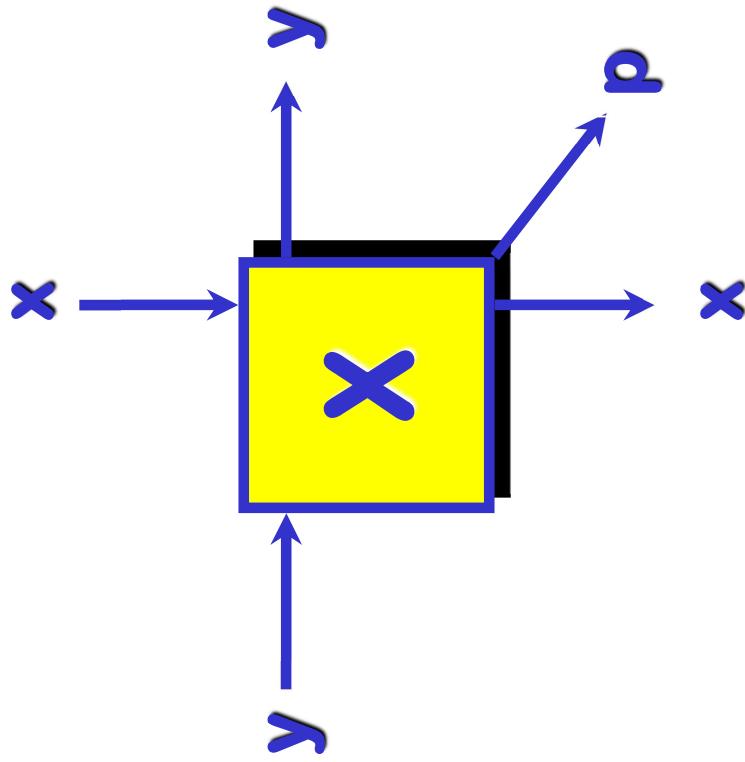
x_3	x_2	x_1	x_0				
y_3	y_2	y_1	y_0				
<hr/>	<hr/>	<hr/>	<hr/>				
x_3y_0	x_2y_0	x_1y_0	x_0y_0				
x_3y_1	x_2y_1	x_1y_1	x_0y_1				
x_3y_2	x_2y_2	x_1y_2	x_0y_2				
x_3y_3	x_2y_3	x_1y_3	x_0y_3				
<hr/>	<hr/>	<hr/>	<hr/>				
p_7	p_6	p_5	p_4	p_3	p_2	p_1	p_0

Scheme



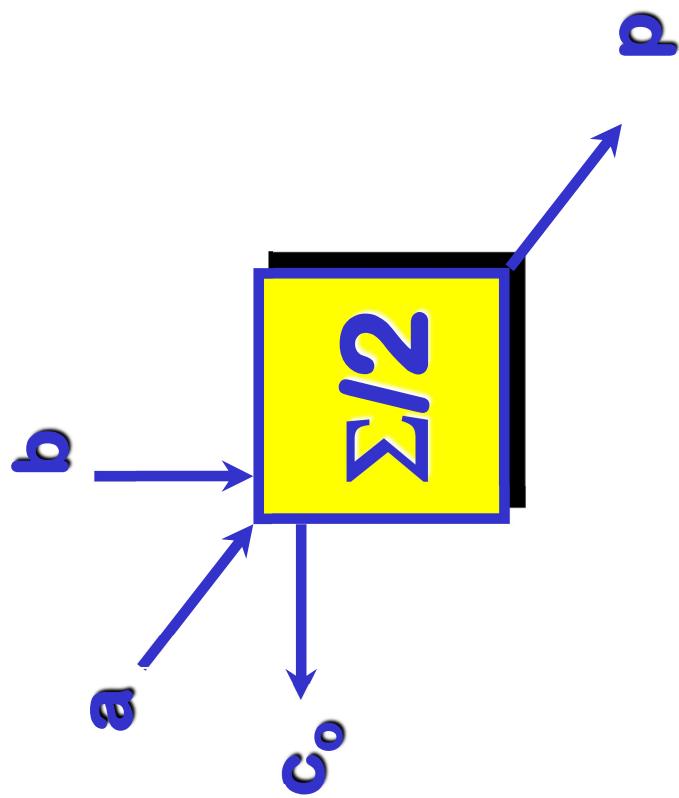
Basic Block 1

★ Product



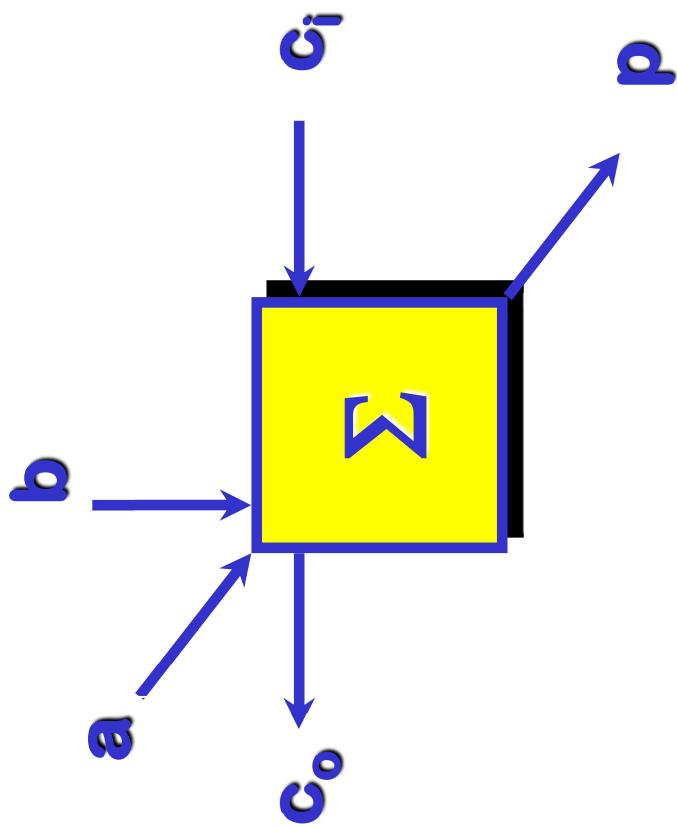
Basic Block 2

★ Half Adder



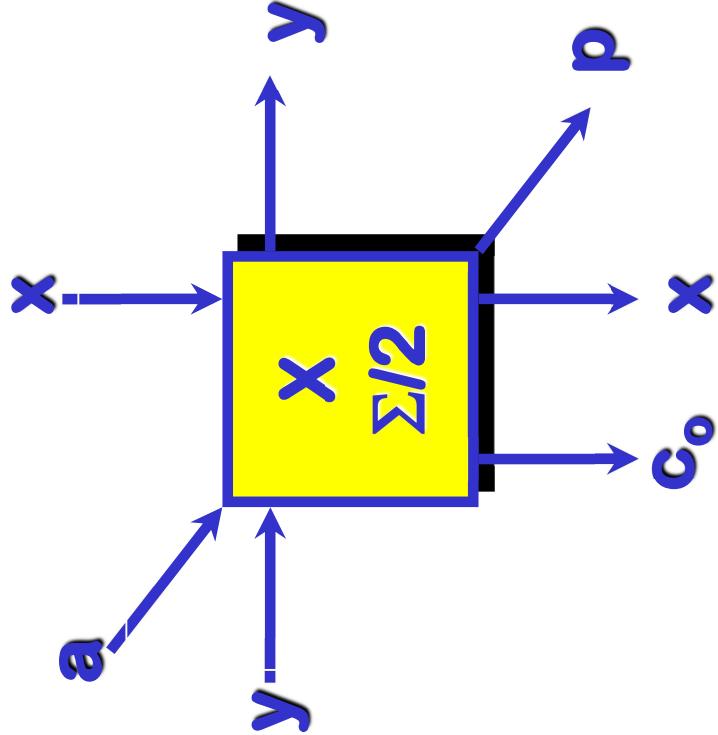
Basic Block 3

★ Full Adder



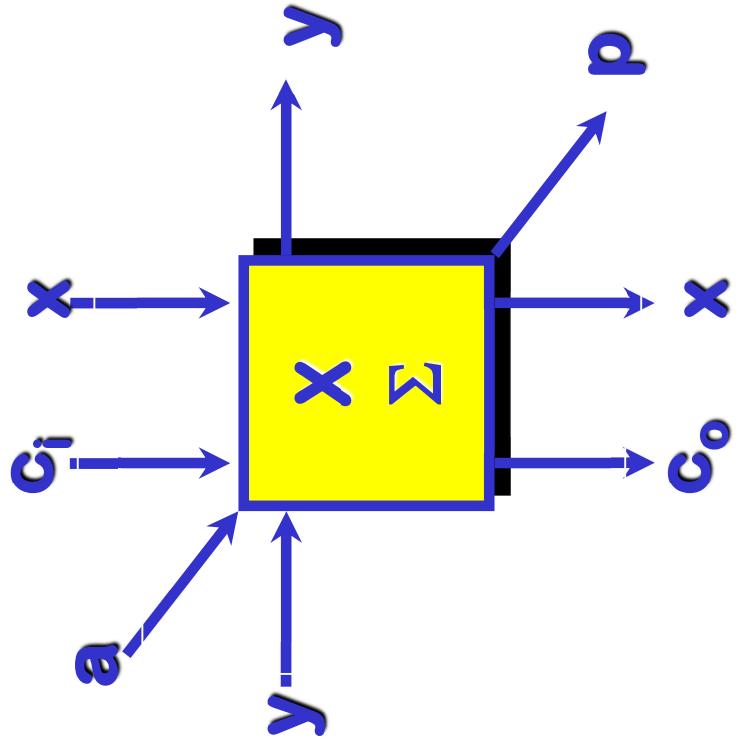
Basic Block 4

★ Product + half adder

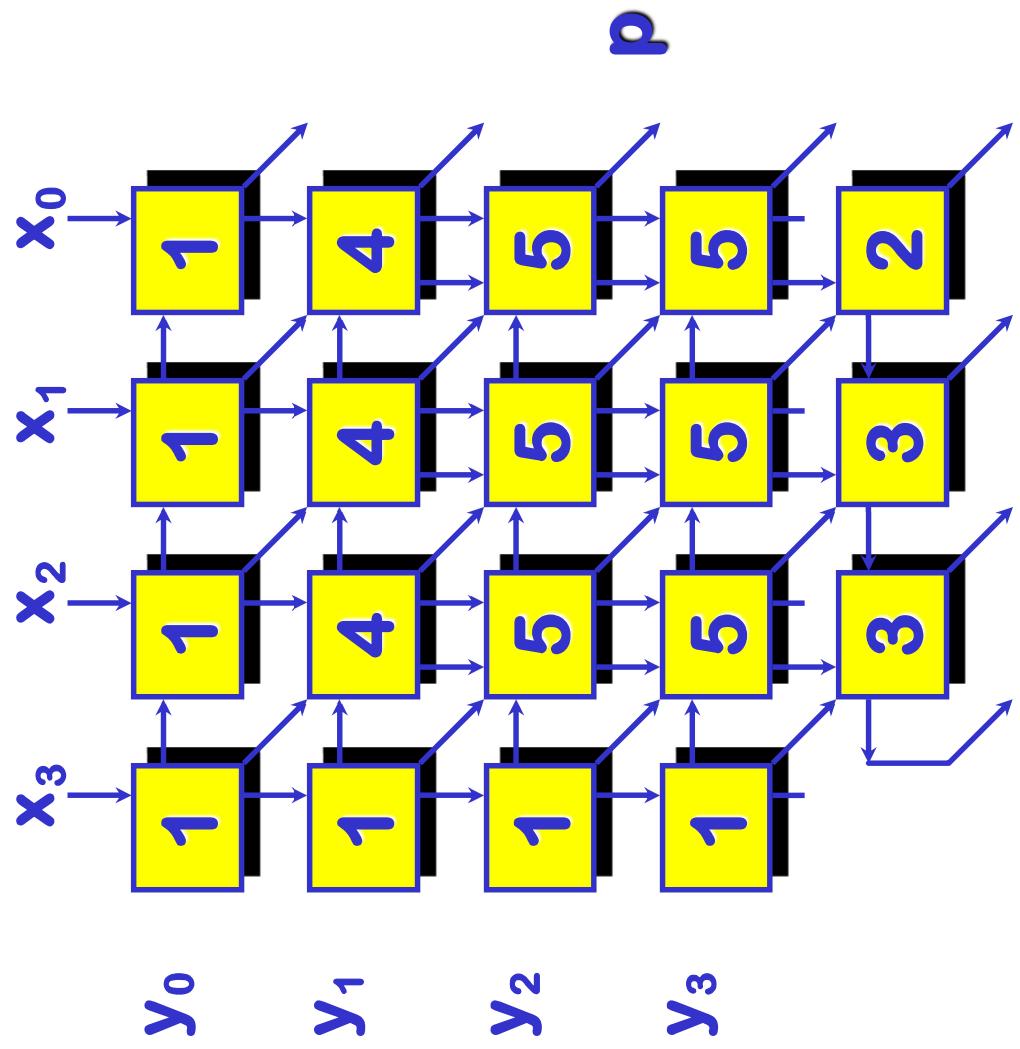


Basic Block 5

❖ Product + full adder

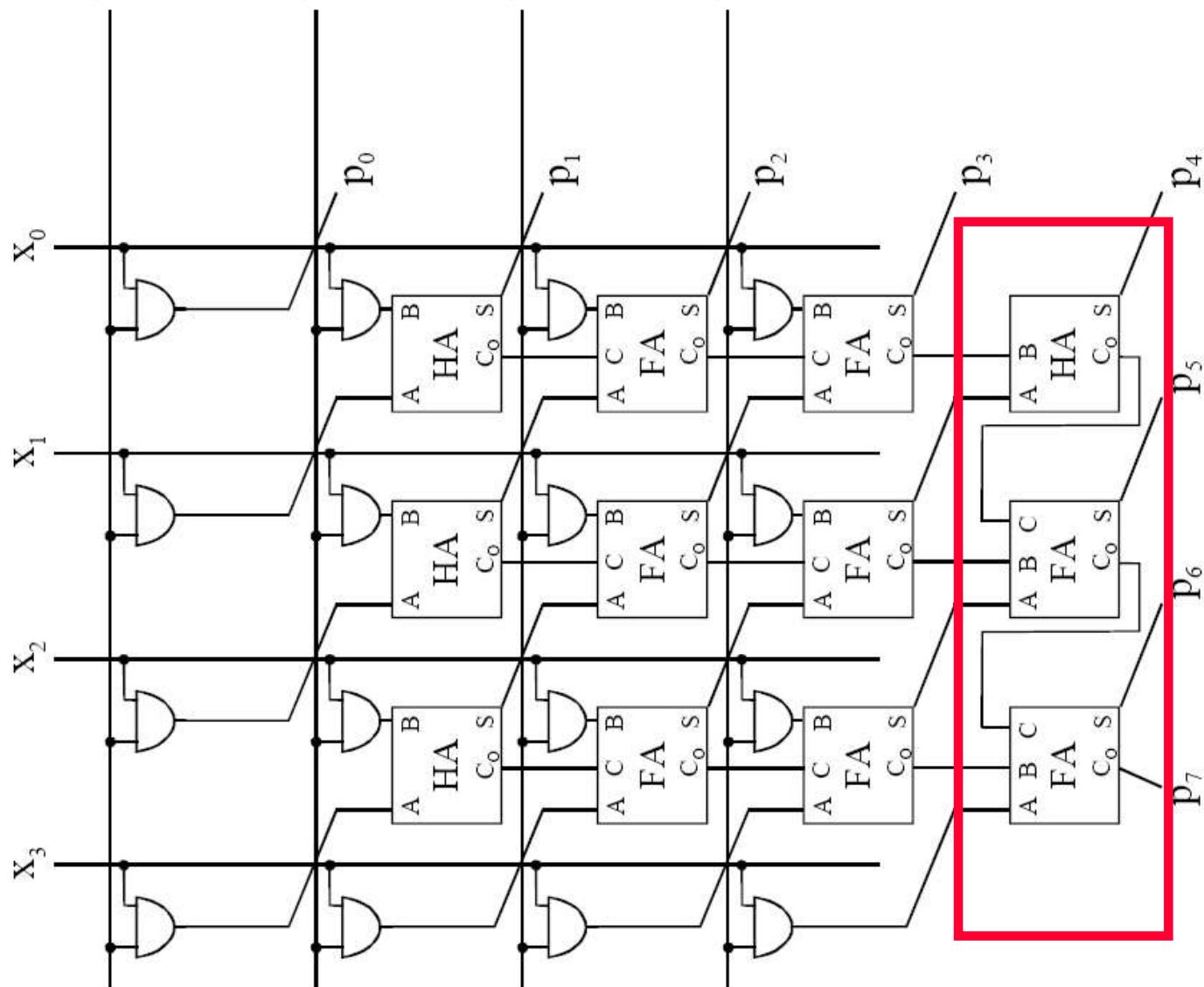


Parallel Multiplier



Complexity

- Requires
 - N^2 AND
 - $N(N - 2)$ Full Adder
 - N Half Adder
- Total propagation delay
 - $T_b = \text{Single Cell propagation delay}$
 - $T_{\text{tot}} = 2(N-1) T_b$



End, Questions ?

- Full Adder
- Serial Adder
- Parallel Adder
- Ripple Carry Adder
- Carry Look Ahead Adder
- Pipeline
- Subtractor
- ALU
- Multiplier

