5 BASIC ADDITION AND COUNTING

Chapter Goals

Study the design of ripple-carry adders, discuss why their latency is unacceptable, and set the foundation for faster adders

Chapter Highlights

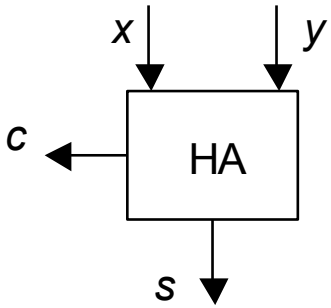
Full adders are versatile building blocks
Longest carry chain on average: $\log_2 k$ bits
Fast asynchronous adders are simple
Counting is relatively easy to speed up
Key part of a fast adder is its carry network

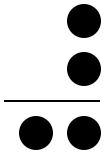
BASIC ADDITION AND COUNTING: TOPICS

Topics in This Chapter
5.1 Bit-Serial and Ripple-Carry Adders
5.2 Conditions and Exceptions
5.4 Carry Completion Detection
5.5 Addition of a Constant
5.6 Manchester Carry Chains and Adders

5.1 BIT-SERIAL AND RIPPLE-CARRY ADDERS

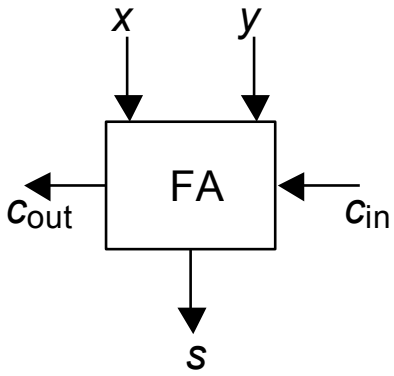
Inputs		Outputs	
x	y	c	s
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

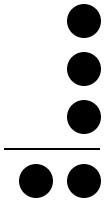




Half-adder (HA): Truth table and block diagram

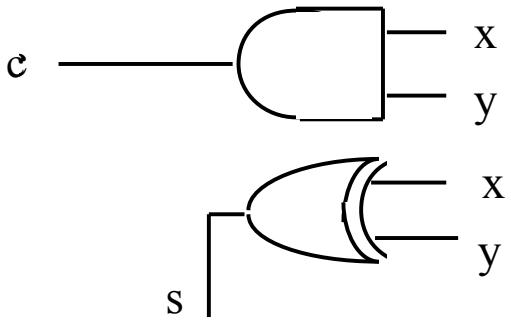
Inputs			Outputs	
x	y	c_{in}	c_{out}	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



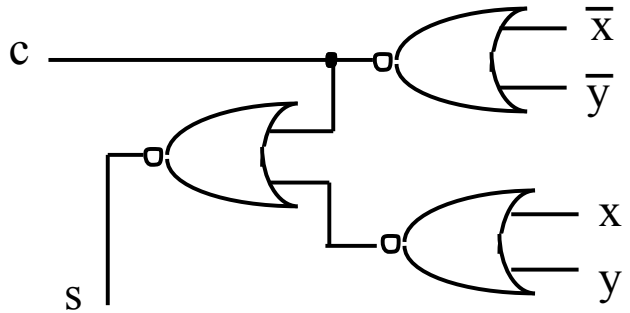


Full-adder (FA): Truth table and block diagram

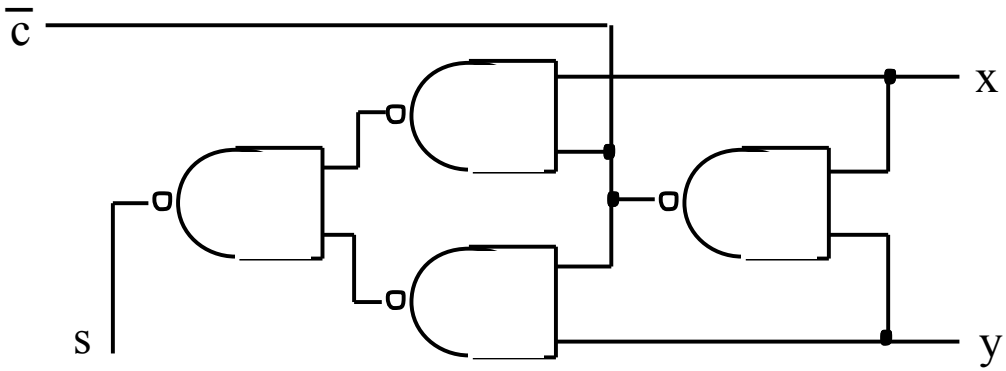
HALF-ADDER IMPLEMENTATIONS



(a) AND/XOR half-adder.



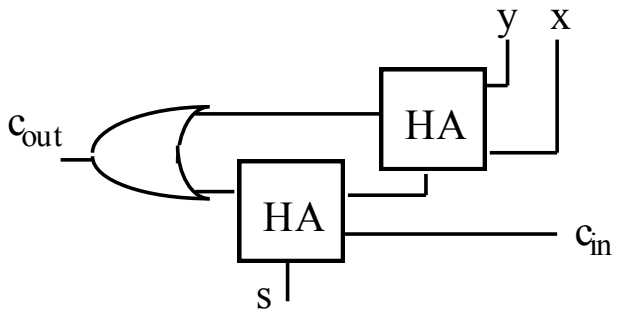
(b) NOR-gate half-adder.



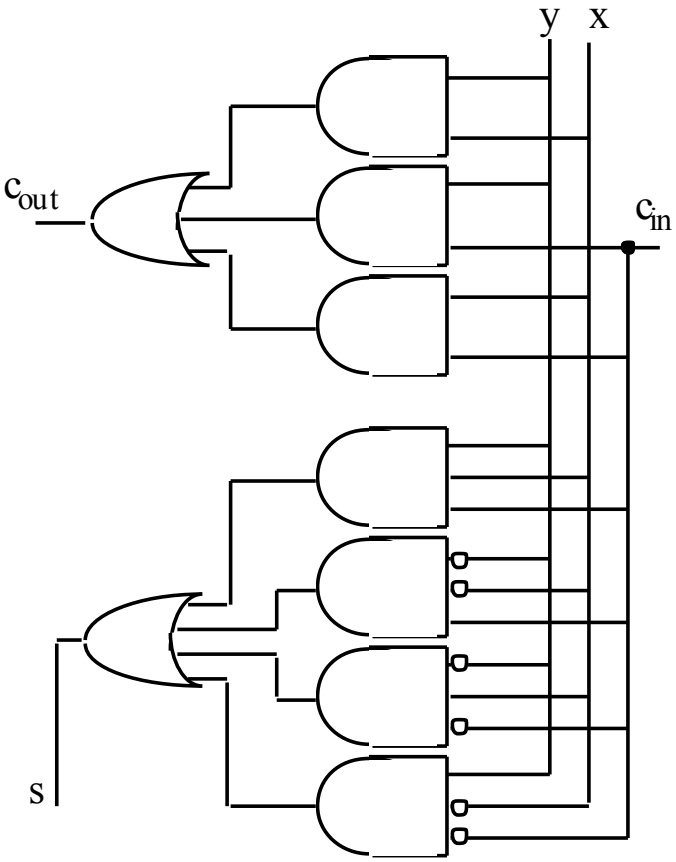
(c) NAND-gate half-adder with complemented carry.

Fig. 5.1 Three implementations of a half-adder.

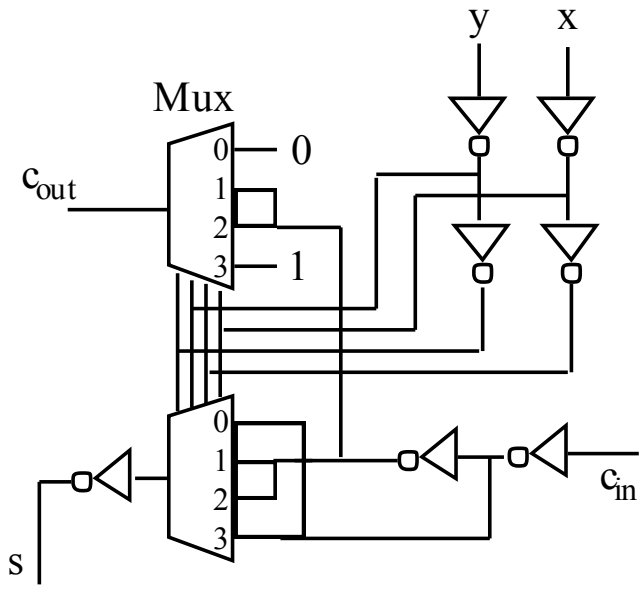
FULL-ADDER IMPLEMENTATIONS



(a) Built of half-adders.



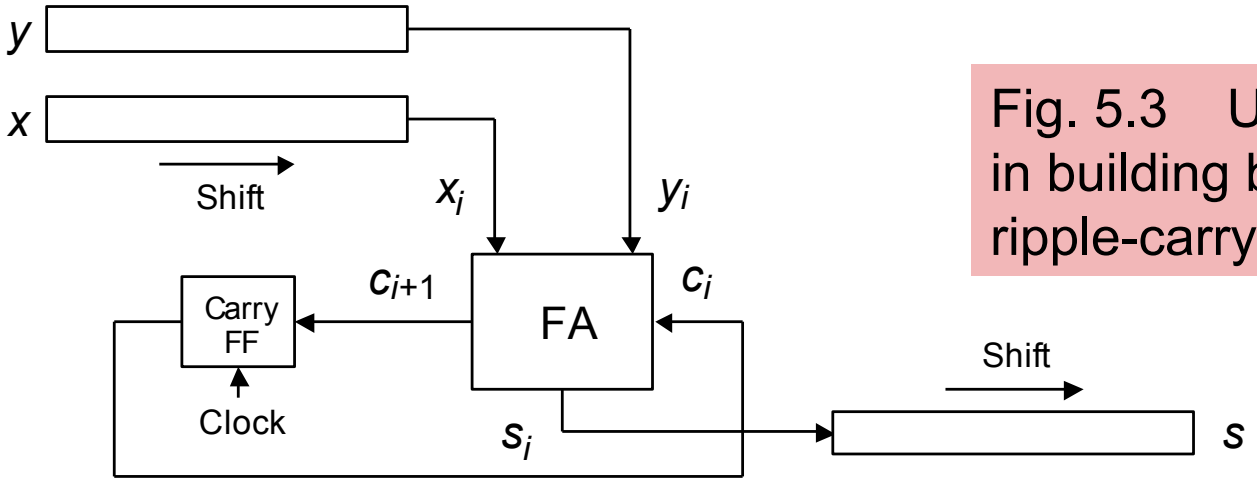
(b) Built as an AND-OR circuit.



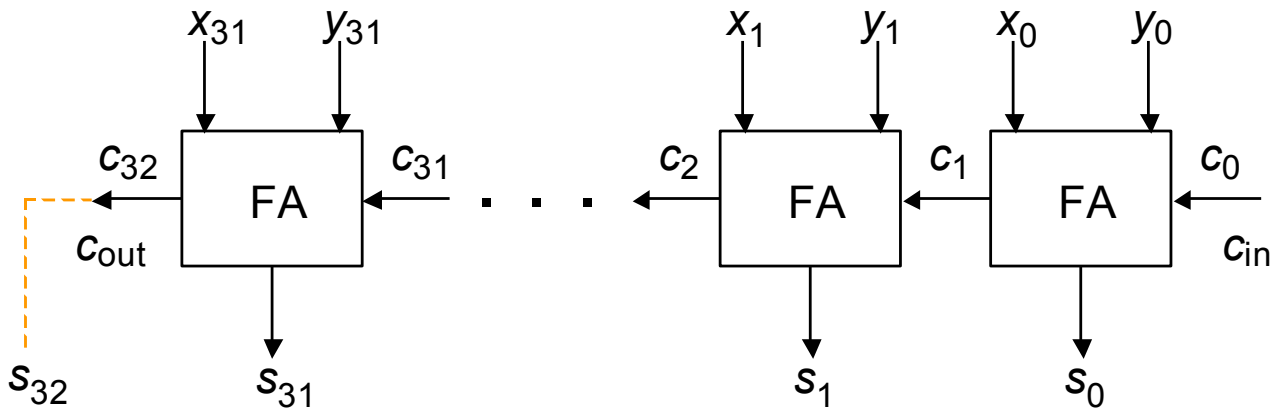
(c) Suitable for CMOS realization.

Fig. 5.2 Possible designs for a full-adder in terms of half-adders, logic gates, and CMOS transmission gates.

SIMPLE ADDERS BUILT OF FULL-ADDERS



(a) Bit-serial adder.



(b) Ripple-carry adder.

Fig. 5.3 Using full-adders in building bit-serial and ripple-carry adders.

CRITICAL PATH THROUGH A RIPPLE-CARRY ADDER

$$T_{\text{ripple-add}} = T_{\text{FA}}(x,y \rightarrow c_{\text{out}}) + (k-2) \times T_{\text{FA}}(c_{\text{in}} \rightarrow c_{\text{out}}) + T_{\text{FA}}(c_{\text{in}} \rightarrow s)$$

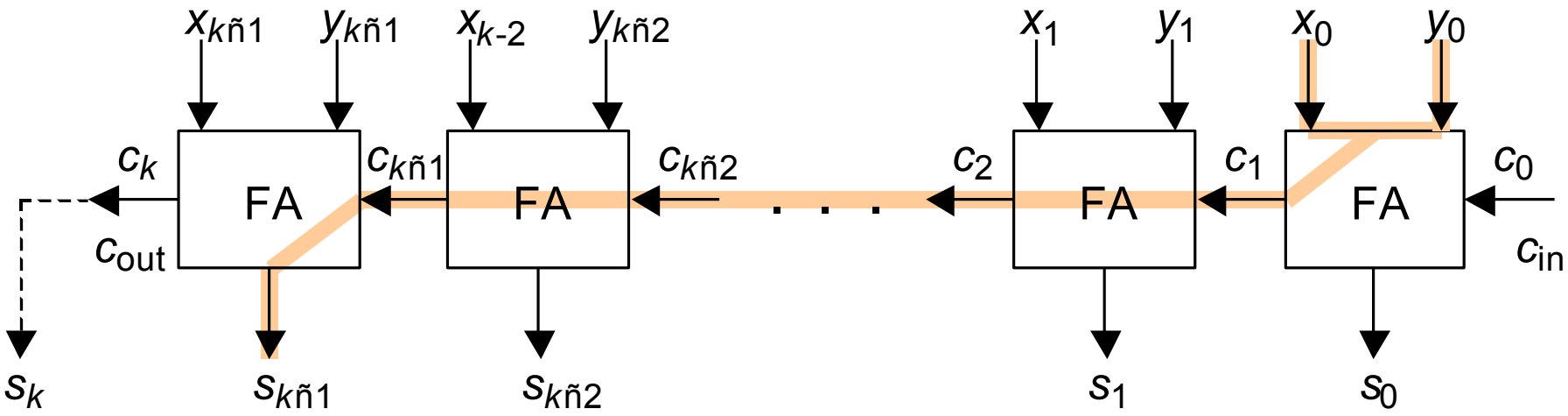
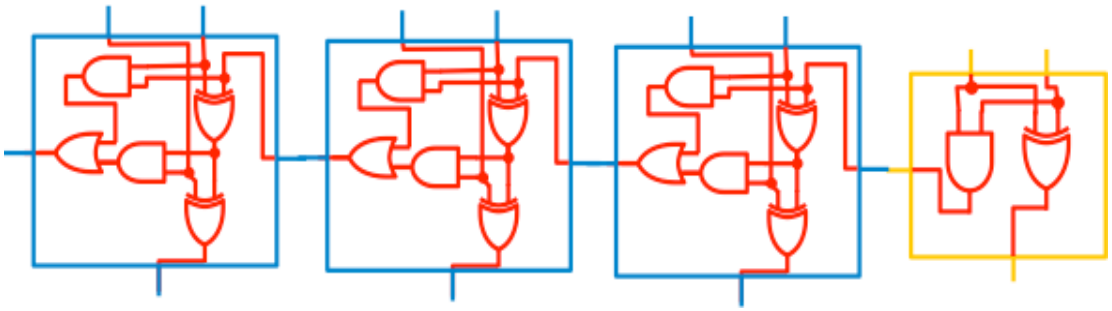


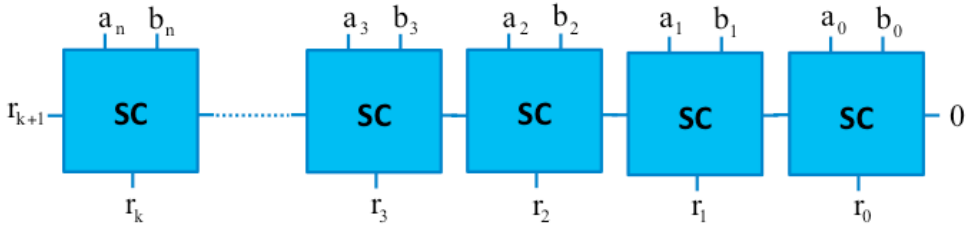
Fig. 5.5 Critical path in a *k*-bit ripple-carry adder.

PROBLEMAS

Problema 5.1 Qual o atraso do caminho crítico do circuito abaixo, considerando que cada porta lógica resulta em um atraso de 3 ns?



Problema 5.2 Qual o atraso do caminho crítico do circuito abaixo, considerando que cada bloco resulta em um atraso de 5 ns?

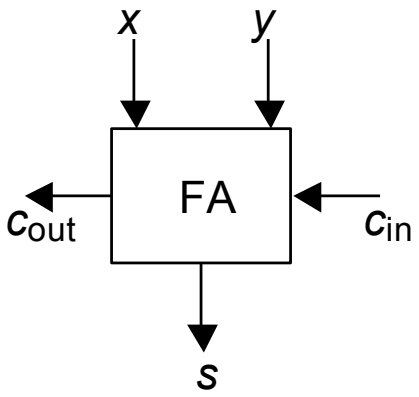


BINARY ADDERS AS VERSATILE BUILDING BLOCKS

Set one input to 0: c_{out} = AND of other inputs

Set one input to 1: c_{out} = OR of other inputs

Set one input to 0 and another to 1: s = NOT of third input



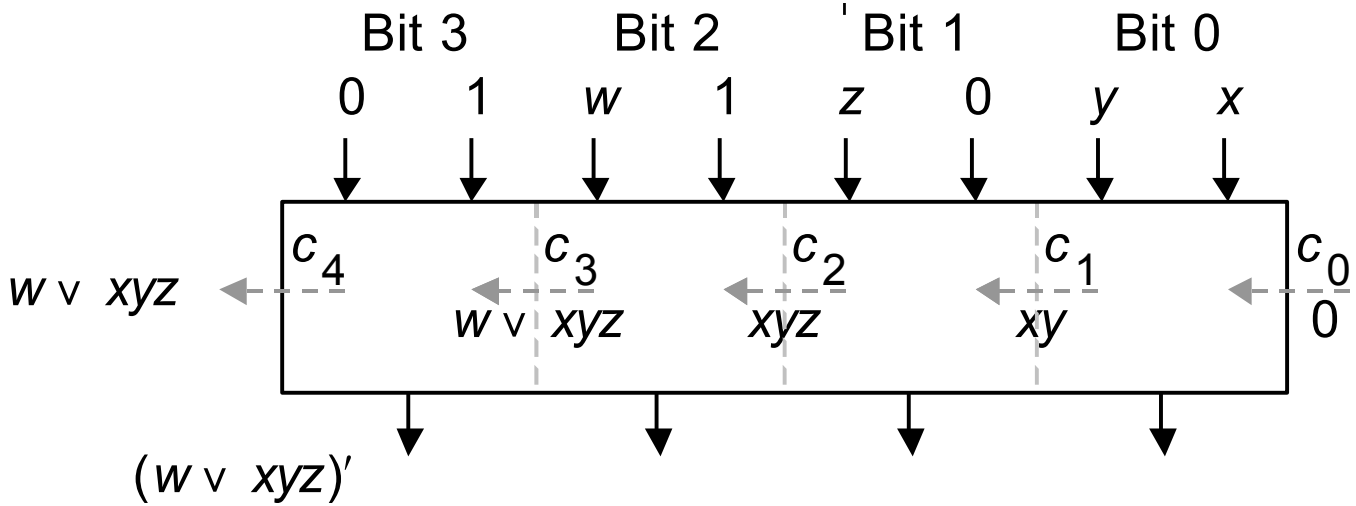


Fig. 5.6 Four-bit binary adder used to realize the logic function $f = w + xyz$ and its complement.

PROBLEMAS

Problema 5.5. Usando unicamente um somador de 4 bits implemente:

a) Um somador de 3 bits, com *carry-in* e *carry-out*;

b) Dois somadores independentes de 1 bit (*Full-Adder*);

c) Um somador de um bit (*Full-Adder*) e um somador de dois bits operando de forma independente.

d) Um gerador de imparidade de 4 bits (4-bit XOR).

e) Dois geradores independentes de imparidade de 3 bits.

f) Uma porta AND de 5 entradas.

g) Uma porta OR de 5 entradas.

h) Um circuito que implemente a função lógica de 4 variáveis $wx+yz$.

i) Um circuito que implemente a função lógica de 4 variáveis $wxy+wxz+wyz+xyz$.

j) Um multiplicador $f=15y$, onde entrada y é de dois bits e f de 6 bits.

k) Um circuito que compute $x+4y+8z$, onde x, y, e z são números de 3 bits sem sinal.

l) Um contador paralelo de 5 entradas, produzindo uma saída de 3 bits.

5.2 CONDITIONS AND EXCEPTIONS

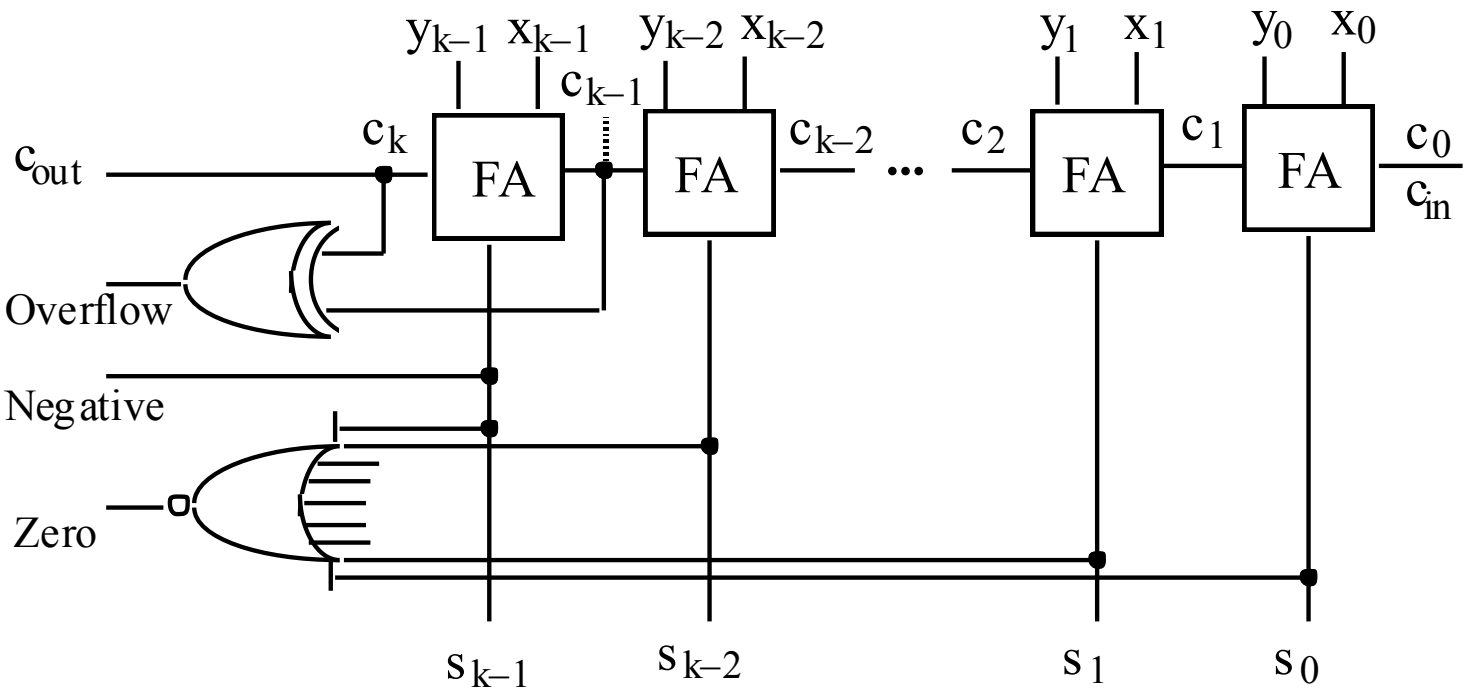


Fig. 5.7 Two's-complement adder with provisions for detecting conditions and exceptions.

PROBLEMAS

Problema 5.6. Considere os seguintes números, representados com 4 bits em complemento para dois:

A = 0011 ; B = 1001

Indique, para a operação A + B:

a)o vector de soma (S) resultante;

b)o vector constituído pelos vários bits de transporte (C) gerados ao longo da operação;

o valor das *flags* zero (Z), negativo (N) e *overflow* (V) à saída da unidade aritmética.

SATURATING ADDERS

Saturating (saturation) arithmetic:

When a result's magnitude is too large, do not wrap around; rather, provide the most positive or the most negative value that is representable in the number format

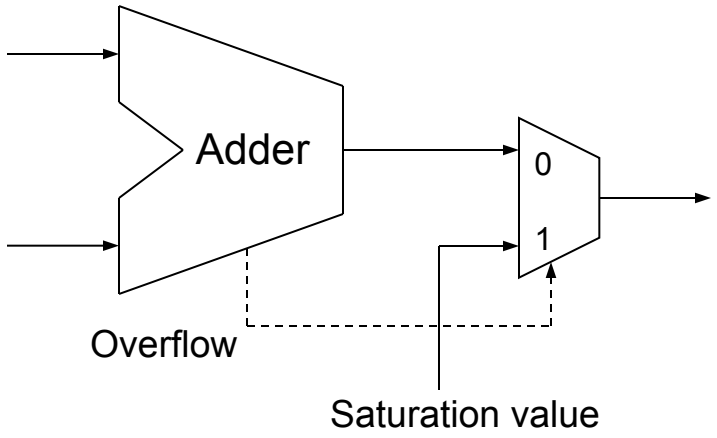
Example – In 8-bit 2's-complement format, we have:
120 + 26 → 18 (wraparound); 120 $+_{sat}$ 26 → 127 (saturating)

Saturating arithmetic in desirable in many DSP applications

Designing saturating adders

Unsigned (quite easy)

Signed (only slightly harder)



PROBLEMAS

Problema 5.7. Usando a ideia de somadores com saturação:

a) Implemente a operação $A+B$ (4 bits) quando B é par e $2(A+B)$ quando B é impar.

b) Adicione um valor de saturação de 15_{10} quando exista *overflow*.

5.4 CARRY COMPLETION DETECTION

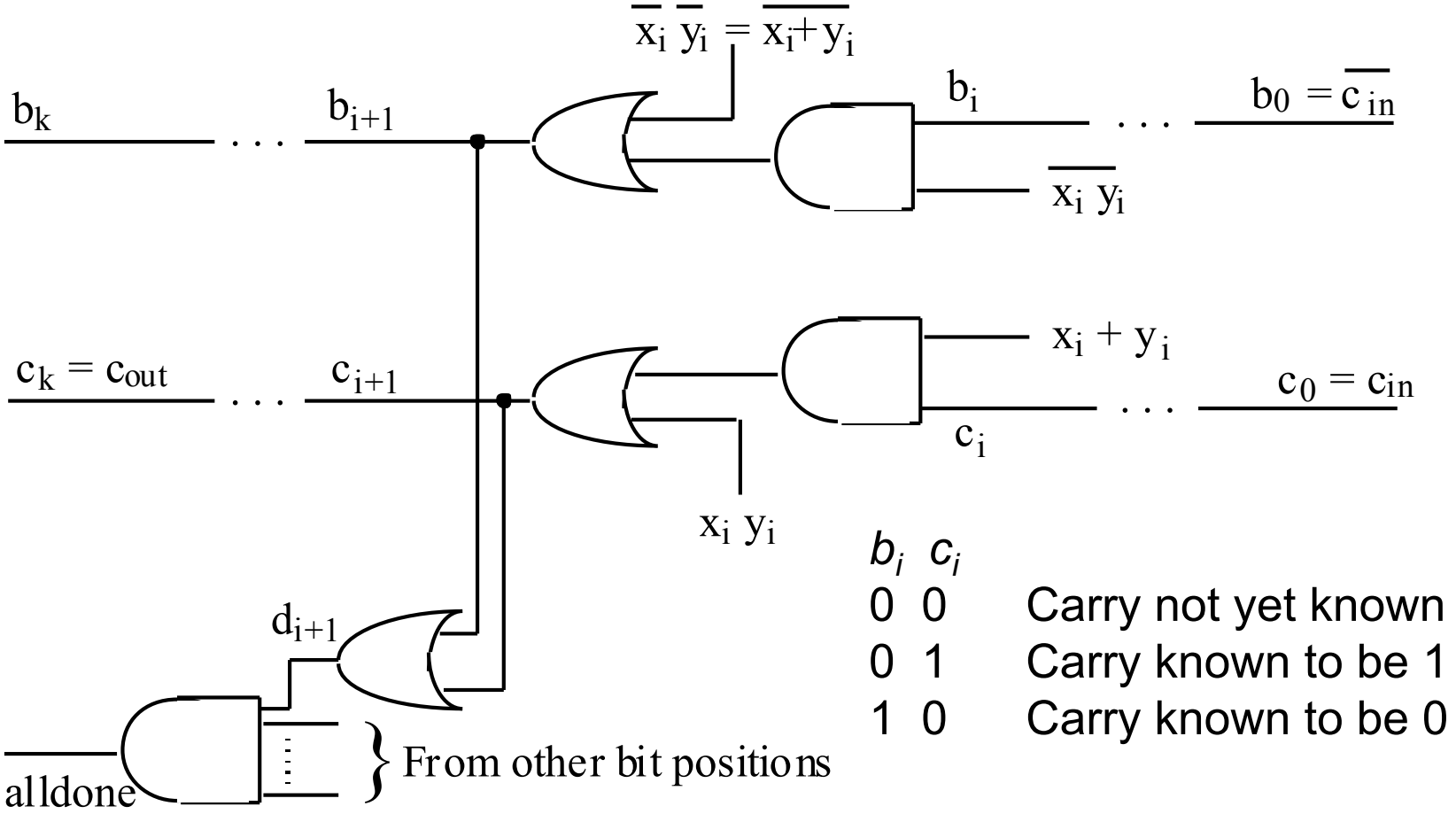


Fig. 5.9 The carry network of an adder with two-rail carries and carry completion detection logic.

5.5 ADDITION OF A CONSTANT: COUNTERS

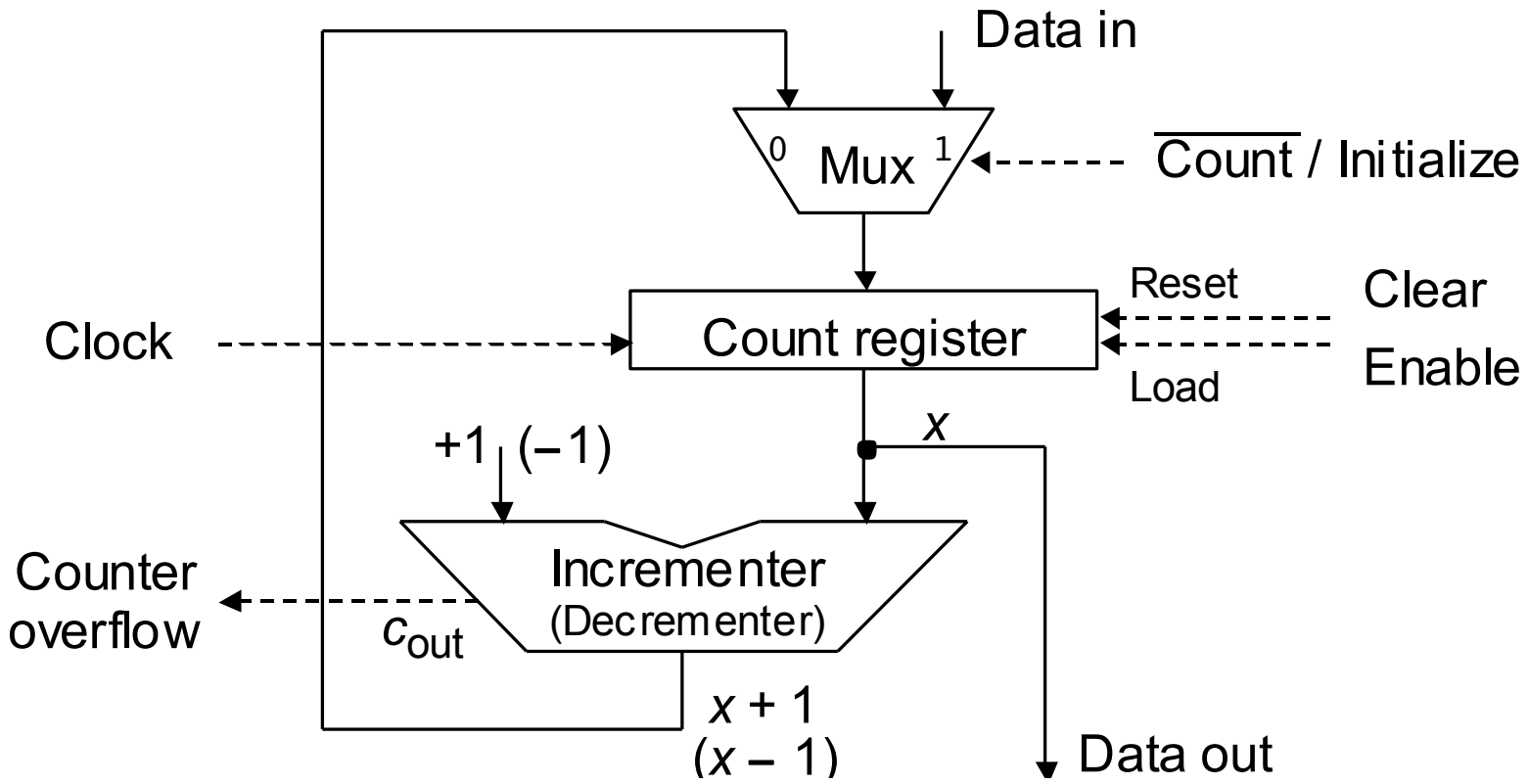


Fig. 5.10 An up (down) counter built of a register, an incrementer (decrementer), and a multiplexer.

PROBLEMAS

Problema 5.8. Usando incrementadores/decrementadores, registradores, deslocadores e multiplexadores implemente os sequenciadores:

a) 0 → 2 → 6 → 14 → 30 → ...

b)0 → 3 → 7 → 15 → 31 → ...

c)4 → 6 → 10 → 18 → 34 →...

5.6 MANCHESTER CARRY CHAINS AND ADDERS

Sum binary digit $s_i = x_i \oplus y_i \oplus c_i$

Computing the carries c_i is thus our central problem
For this, the actual operand digits are not important
What matters is whether in a given position a carry is

generated, *propagated*, or *annihilated* (*absorbed*)

For binary addition:

$g_i = x_i\, y_i$ $\quad p_i = x_i \oplus y_i$ $\quad a_i = x_i'\, y_i' = (x_i \vee y_i)'$

It is also helpful to define a *transfer* signal:

$t_i = g_i \vee p_i = a_i' = x_i \vee y_i$

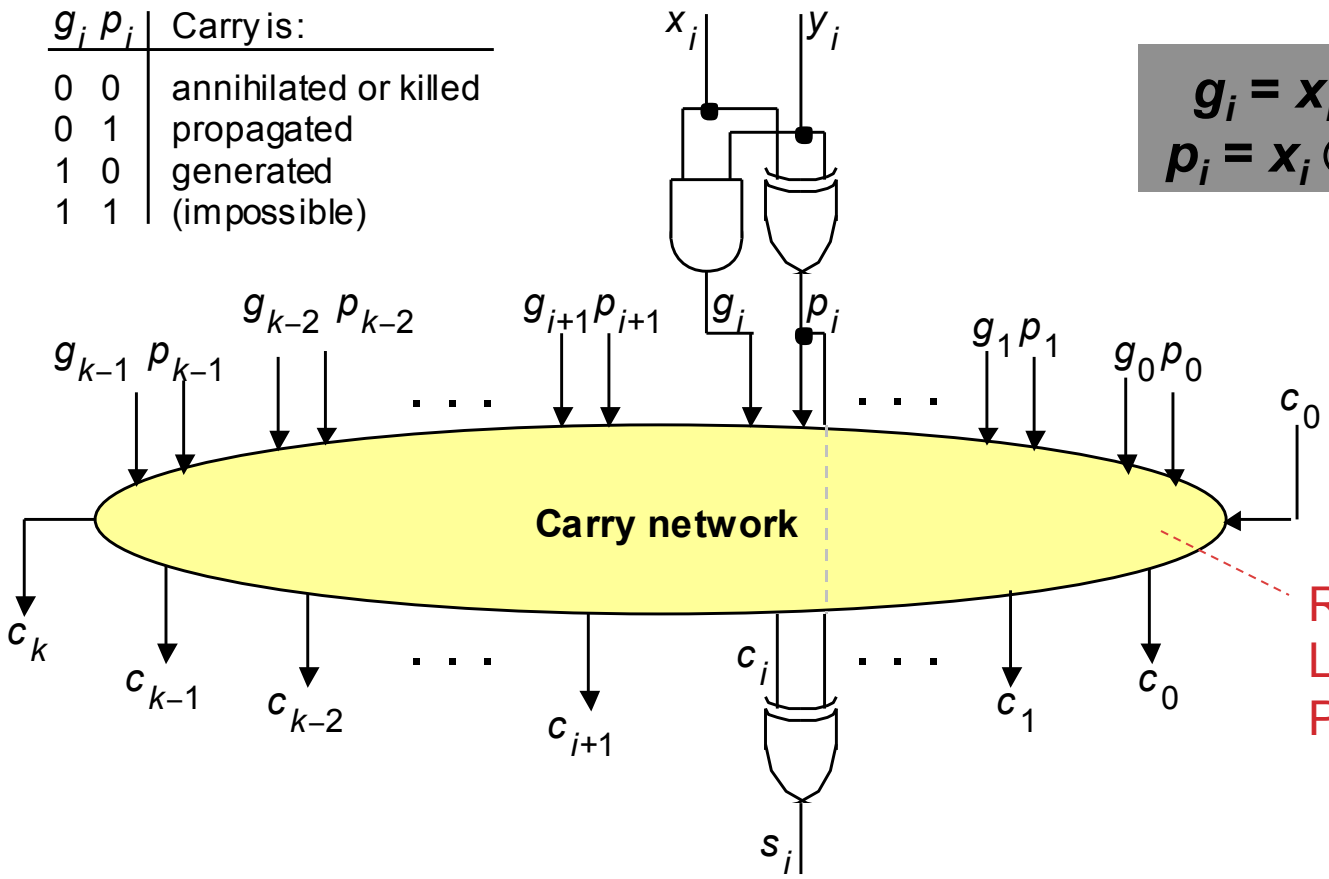
Using these signals, the *carry recurrence* is written as

$c_{i+1} = g_i \vee c_i\, p_i = g_i \vee c_i\, g_i \vee c_i\, p_i = g_i \vee c_i\, t_i$

CARRY NETWORK IS THE ESSENCE OF A FAST ADDER

g_i	p_i	Carry is:
0	0	annihilated or killed
0	1	propagated
1	0	generated
1	1	(impossible)

$$g_i = x_i\, y_i$$
$$p_i = x_i \oplus y_i$$



Ripple; Skip;
Lookahead;
Parallel-prefix

Fig. 5.14 Generic structure of a binary adder, highlighting its carry network.

RIPPLE-CARRY ADDER REVISITED

The carry recurrence: $c_{i+1} = g_i \vee p_i c_i$

Latency of k-bit adder is roughly $2k$ gate delays:

1 gate delay for production of p and g signals, plus
$2(k - 1)$ gate delays for carry propagation, plus
1 XOR gate delay for generation of the sum bits

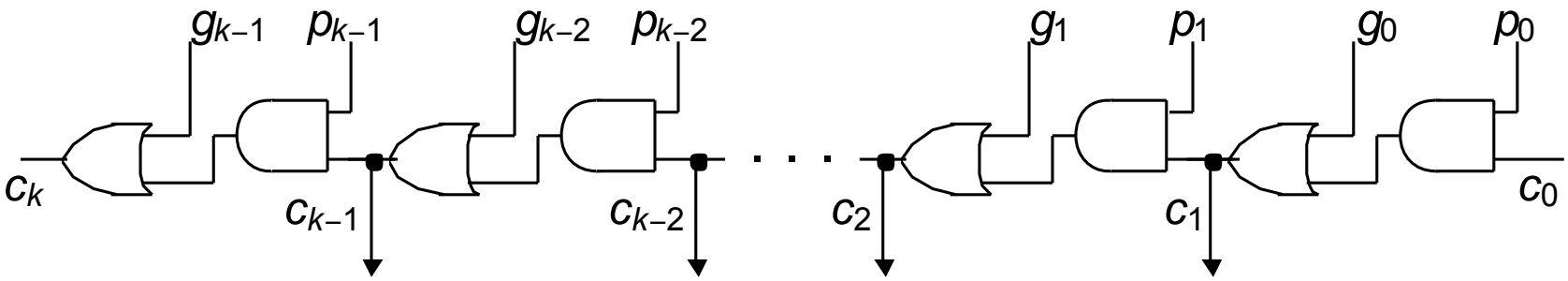


Fig. 5.15 Alternate view of a ripple-carry network in connection with the generic adder structure shown in Fig. 5.14.

THE COMPLETE DESIGN OF A RIPPLE-CARRY ADDER

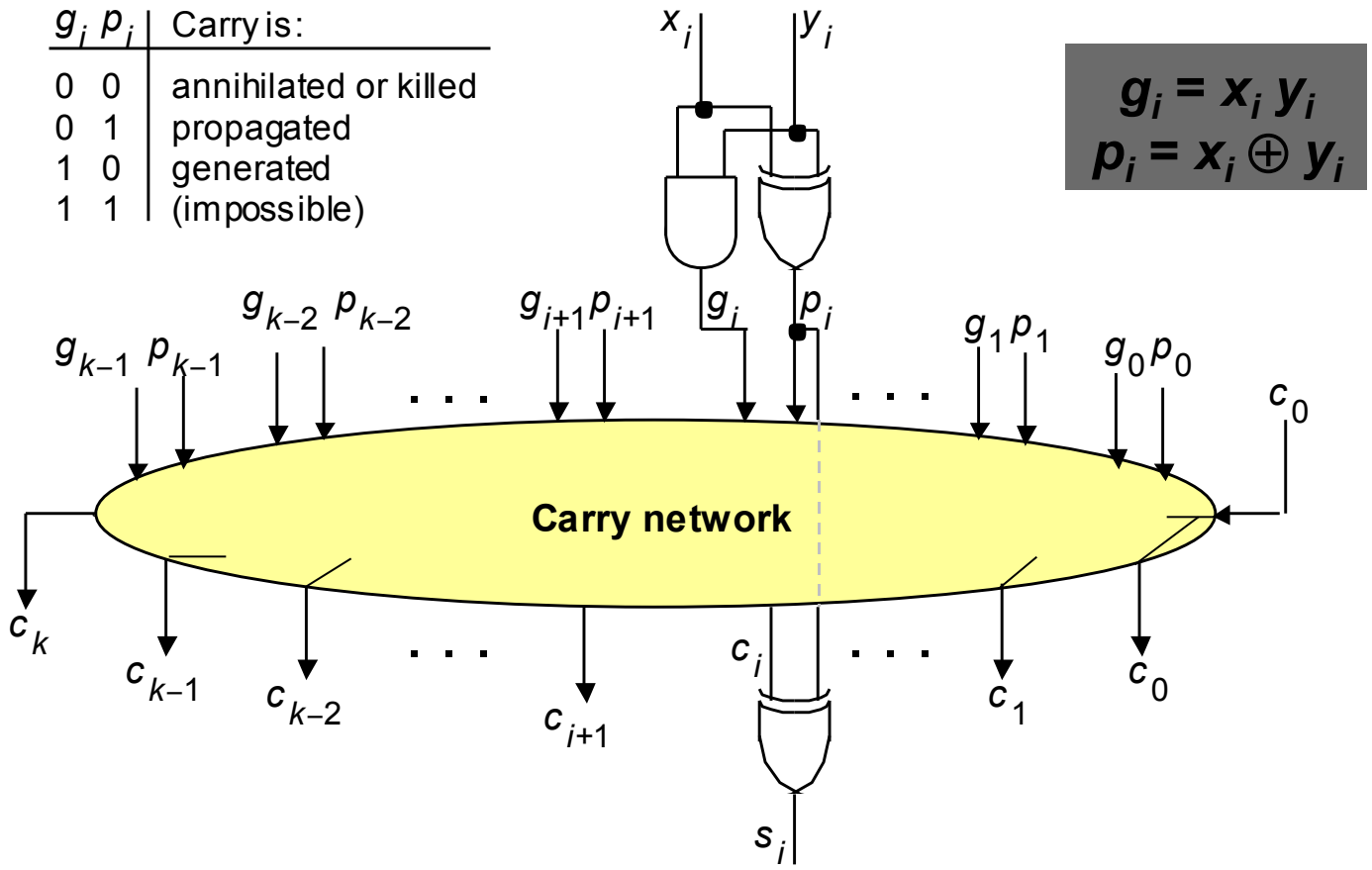


Fig. 5.15 (ripple-carry network) superimposed on Fig. 5.14 (generic adder).

PROBLEMAS

Problema 5.9. Compare o atraso do Carry-Propagate-Adder (CPA) e o Ripple-Carry-Adder (RCA) para 4 bits, 8 bits e 16 bits. Considere as portas AND-2 e OR-2 com um atraso de 2ns e XOR-2 com um atraso de 3ns.