Aritmética Digital

Prof. Fernando Passold

Soma binária:

- Exemplos: $\sum = A + B$

			1	<- Carry-
0	0	1	1	<- A
+ 0	+ 1	+ 0	+ 1	<- B
0	1	1	0	<- $\sum$

Problema: útil, mas só posso realizar somas envolvendo palavras de 1-bit. E mesmo assim, pode ser gerado um sinal “extra”, o bit de carry-out.
Note que apesar de manipular palavras de 1-bit, gera até 2-bits de saída.

Soma binária:

- Exemplos: Σ=A+B << palavras de 8-bits

Decimal:	Hex:	Binário:	Obs:
1	1	11 1	<- bits de Carry-
28	1C	0001 1100	<- A
+ 22	+16	+ 0001 0110	<- B
50	32	0011 0010	<- Σ

Soma binária:

- Exemplos: $\Sigma=A+B$ << palavras de 8-bits

Decimal:	Hex:	Binário:	Obs:
1	1	11 1	<- bits de Carry-
28	1C	0001 1100	<- A
+ 22	+16	+ 0001 0110	<- B
50	32	0011 0010	<- Σ

Repare: Aqui entra o "**somador parcial**" (não trabalha com bit de carry-in).
mas...

Soma binária:

- Exemplos: Σ=A+B << palavras de 8-bits:

necessidade do bit de carry-in!

Decimal:	Hex:	Binário:	Obs:
1	1	11 1	<- bits de Carry-
28	1C	0001 1100	<- A
+ 22	+16	+ 0001 0110	<- B
50	32	0011 0010	<- Σ

Repare: Aqui entra o "**somador parcial**" (não trabalha com bit de carry-in).
mas...

sexta-feira, 25 de outubro de 13

Somador binário parcial (semi-somador):

- Projeto do circuito:

$$\begin{array}{rr} & A \\ + & B \\ \hline C_{OUT} & \Sigma \end{array}$$

$$\begin{array}{r} 0 \\ +\,0 \\ \hline 0 \end{array} \qquad \begin{array}{r} 0 \\ +\,1 \\ \hline 1 \end{array} \qquad \begin{array}{r} 1 \\ +\,0 \\ \hline 1 \end{array} \qquad \begin{array}{r} 1 \\ +1 \\ \hline 1\ 0 \end{array}$$

Bit de "vai-um" (*carry-out*)

Entradas: A y B
Saídas: Σ e C_{OUT}

Entradas		Saídas	
A	B	COUT	Σ
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$$\left.\begin{array}{l} = \overline{A} \cdot B \\ = A \cdot \overline{B} \end{array}\right\} \quad \Sigma = A \oplus B$$

$$\longrightarrow \quad C_{OUT} = A \cdot B$$

Somador binário parcial (semi-somador):

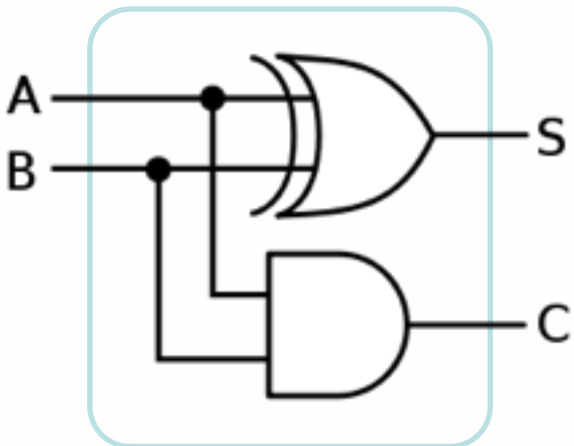
- Circuito:

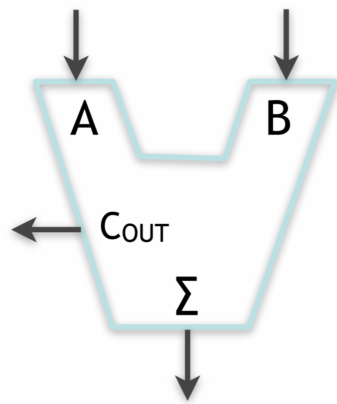
Entradas: A y B
Salidas: Σ y C_{OUT}

$$\Sigma = A \oplus B$$

$$C_{OUT} = A \cdot B$$

Entradas		Salidas	
A	B	COUT	Σ
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0





Somador Completo

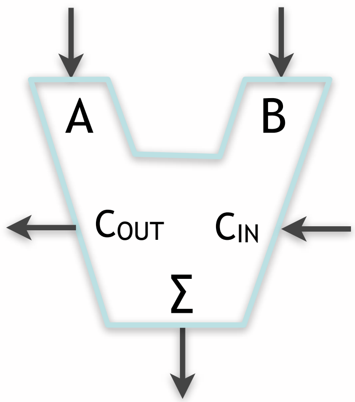
- Projeto:

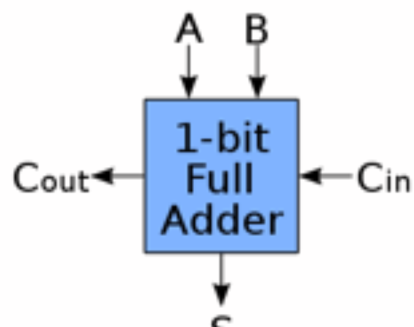
Entradas			Salidas	
C_{IN}	A	B	C_{OUT}	Σ
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

carry-in

$$\begin{array}{ccccc} 1 & 1 & 1 & 1 & C_0 \\ & 1 & 1 & 1 & 1 \\ + & 0 & 0 & 0 & 1 \\ \hline 1 & 0 & 0 & 0 & 0 \\ C_4 & S_3 & S_2 & S_1 & S_0 \end{array}$$

carry-out





sexta-feira, 25 de outubro de 13

Somador Completo

					carry-in
1	1	1	1	C_0	
	1	1	1	1	
+	0	0	0	1	
1	0	0	0	0	
C_4	S_3	S_2	S_1	S_0	
carry-out					

- Projeto:

Entradas			Salidas	
C_{IN}	A	B	C_{OUT}	Σ
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

Σ: AB

C_{IN}	00	01	11	10
0		1		1
1	1		1	

$$\Sigma = \overline{C}_{IN}\left(\overline{A}B + A\overline{B}\right) + C_{IN}\left(\overline{A}\,\overline{B} + AB\right)$$

$$\Sigma = \overline{C}_{IN}\left(A \oplus B\right) + C_{IN}\left(\overline{A \oplus B}\right)$$

$$\Sigma = C_{IN} \oplus \left(A \oplus B\right)$$

C_{OUT}: AB

C_{IN}	00	01	11	10
0			1	
1		1	1	1

$$C_{OUT} = AB + C_{IN}\left(\overline{A}B + A\overline{B}\right)$$

$$C_{OUT} = AB + C_{IN}\left(A \oplus B\right)$$

Somador Completo

carry-in

$$\begin{array}{ccccc} 1 & 1 & 1 & 1 & C_0 \\ & 1 & 1 & 1 & 1 \\ + & 0 & 0 & 0 & 1 \\ \hline 1 & 0 & 0 & 0 & 0 \\ C_4 & S_3 & S_2 & S_1 & S_0 \end{array}$$

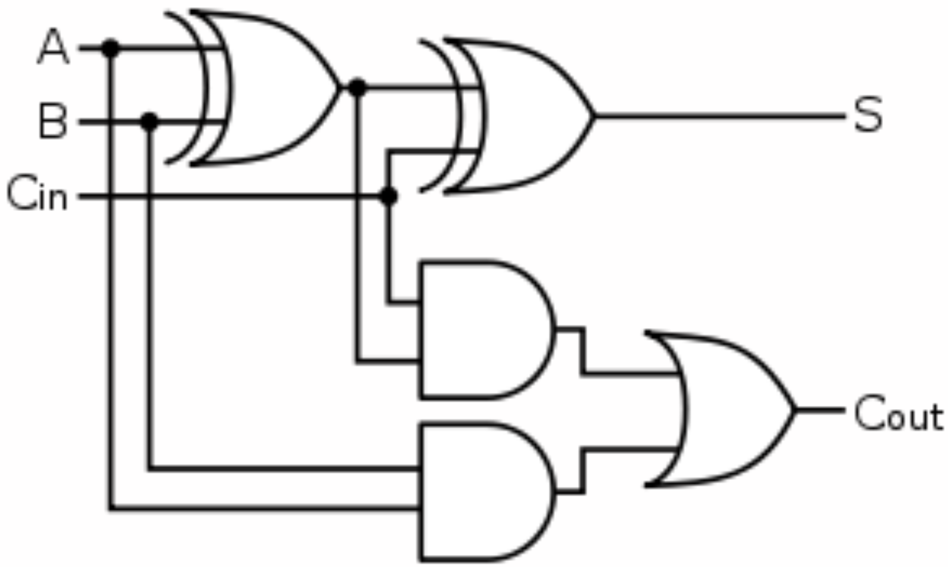
carry-out

- Projeto:

Entradas			Salidas	
C_{IN}	A	B	C_{OUT}	Σ
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

$$\Sigma = (A \oplus B) \oplus C_{IN}$$

$$C_{OUT} = AB + (A \oplus B)C_{IN}$$



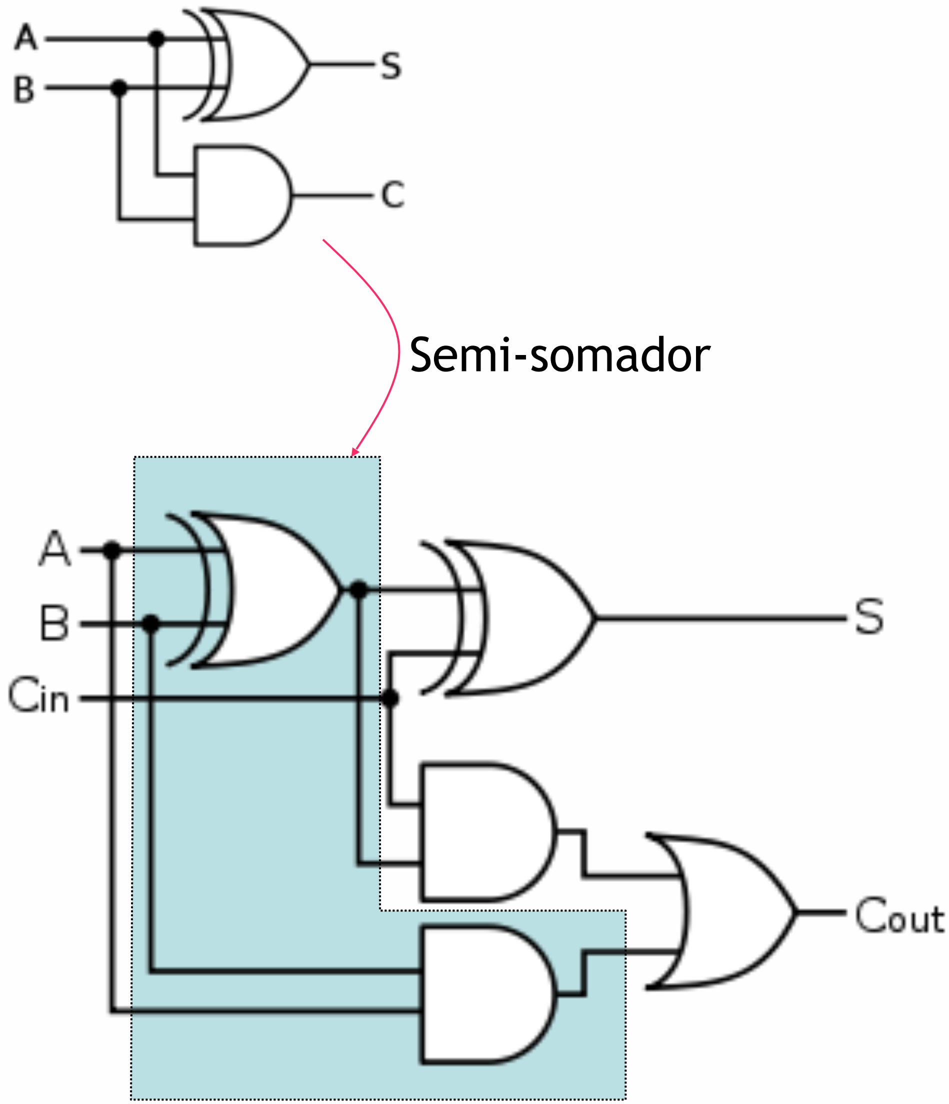
Somador Completo

- Projeto:

Entradas			Salidas	
C_{IN}	A	B	C_{OUT}	Σ
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

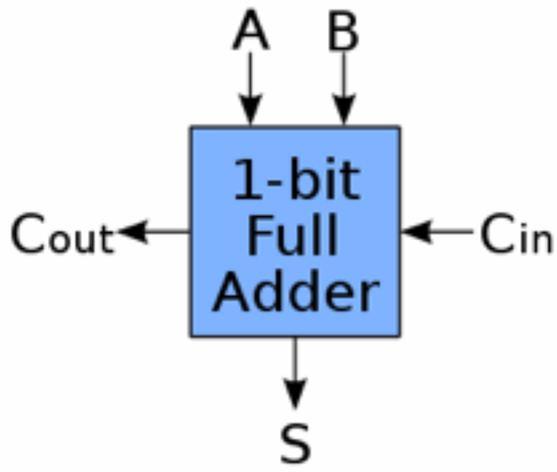
$$\Sigma = (A \oplus B) \oplus C_{IN}$$

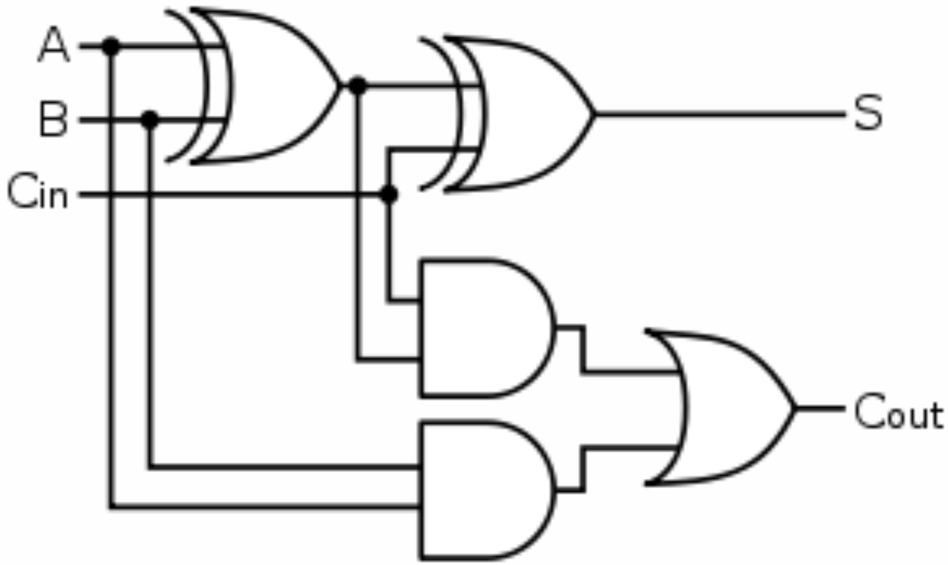
$$C_{OUT} = AB + (A \oplus B) C_{IN}$$

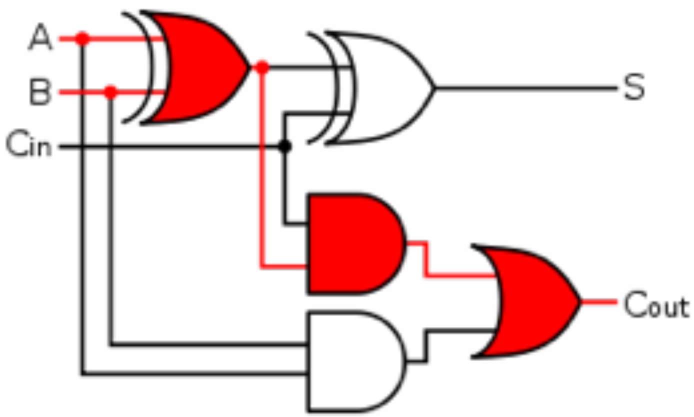


Somador Completo

- Ex$_1$:







Somador Completo

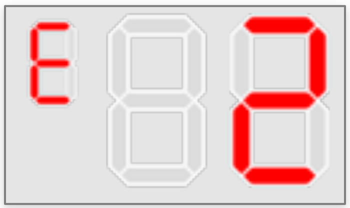
- Ex_2:

3 + 7 = ?

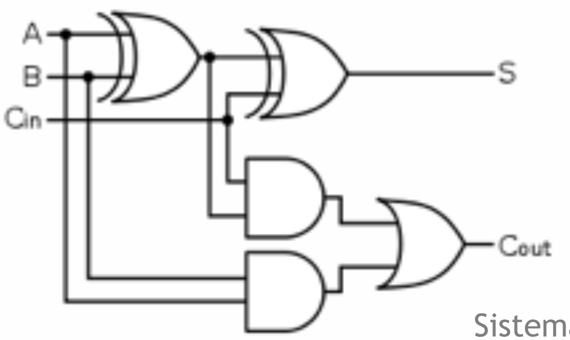
```
        1      1110
 A       3       11
+B     + 7    + 111
--     ---    -----
 Σ      10      010
```

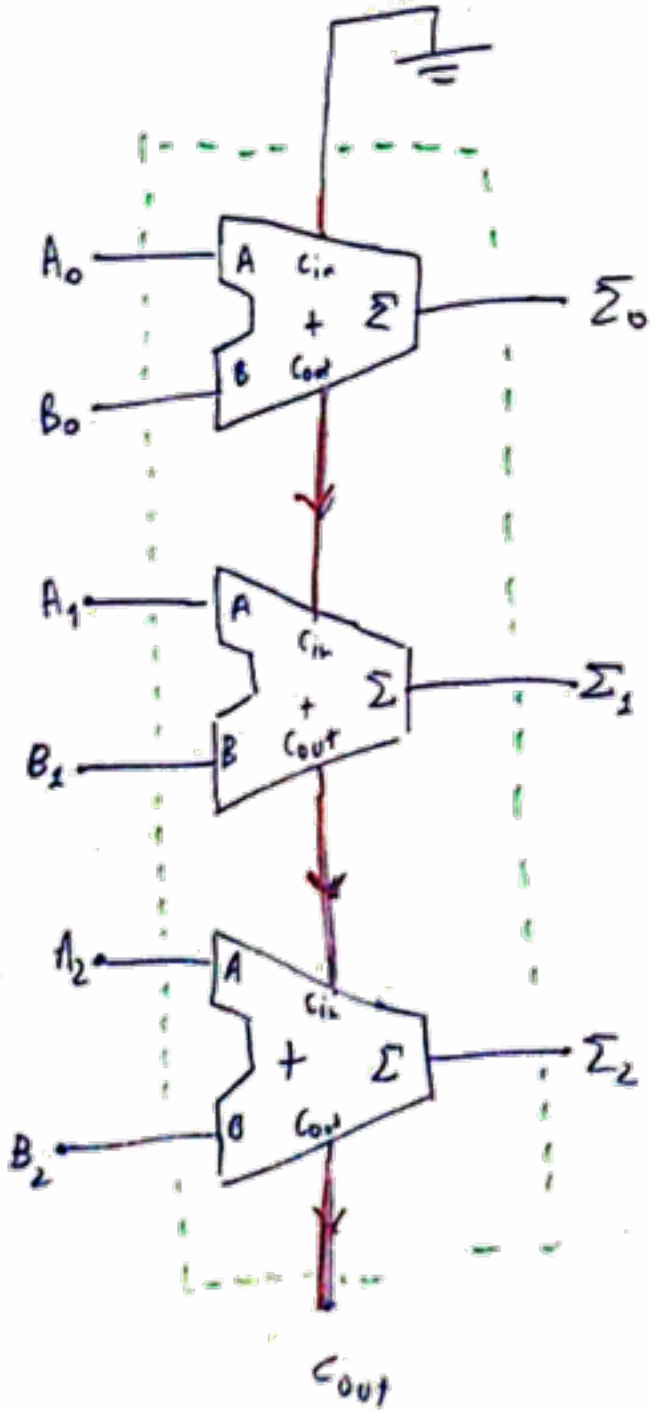
Carry-out!

Carry-in = 0



Overflow
no caso de
representação de
números usando
apenas 3-bits





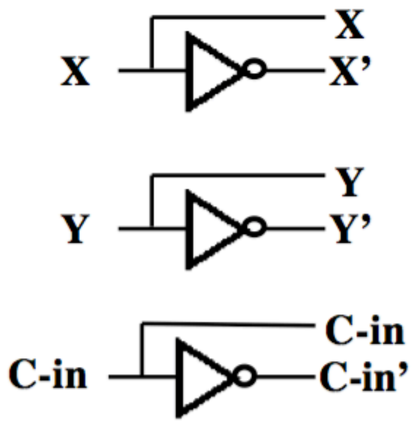
Somador Completo

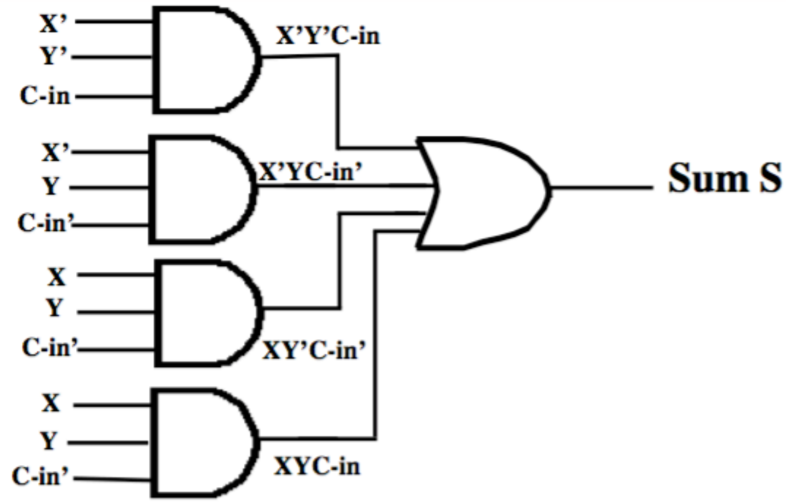
$$\Sigma = (A \oplus B) \oplus C_{IN}$$

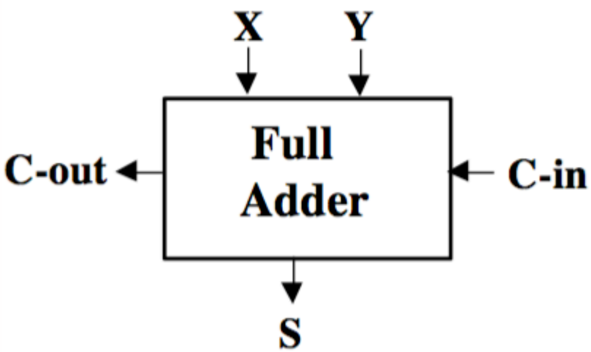
$$C_{OUT} = AB + (A \oplus B)C_{IN}$$

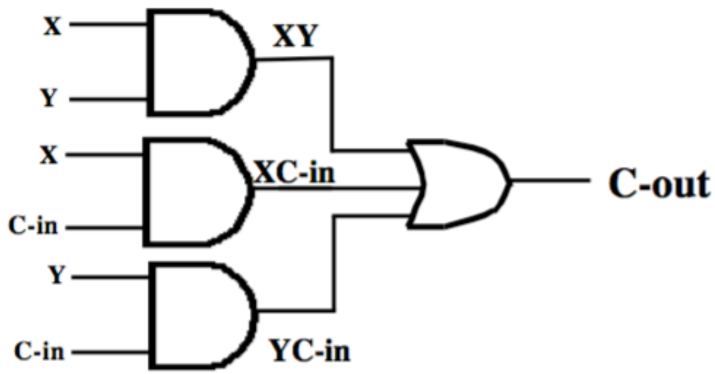
- Ex_2: Somador Completo Usando AND-OR

Entradas			Salidas	
C_{IN}	A	B	C_{OUT}	Σ
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



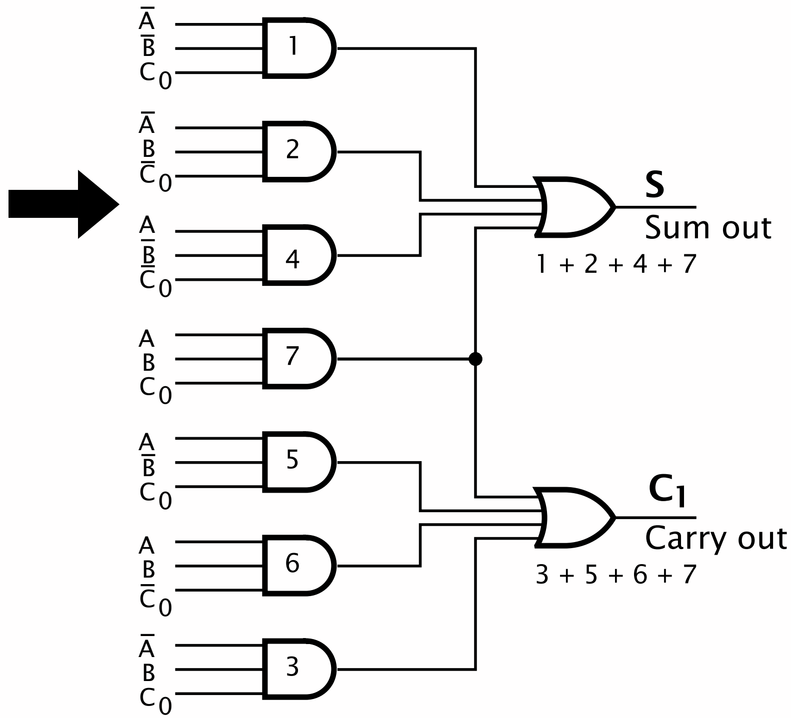






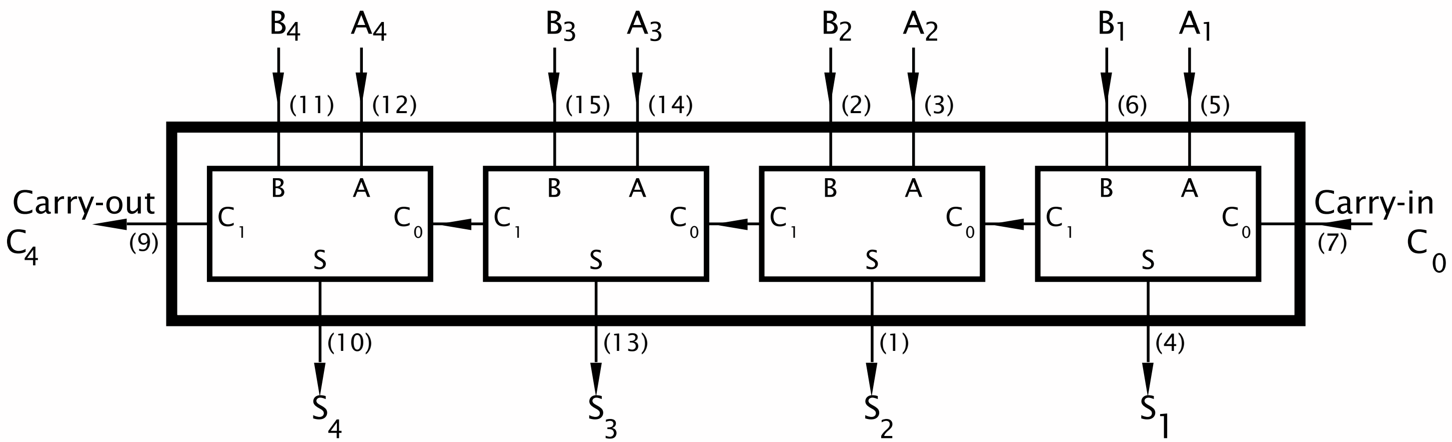
Somador Completo

	A	B	C_0	S	C_1
0	0	0	0	0	0
1	0	0	1	1	0
2	0	1	0	1	0
3	0	1	1	0	1
4	1	0	0	1	0
5	1	0	1	0	1
6	1	1	0	0	1
7	1	1	1	1	1



- Ex$_4$: C.I. 74LS283

(a) One-bit addition

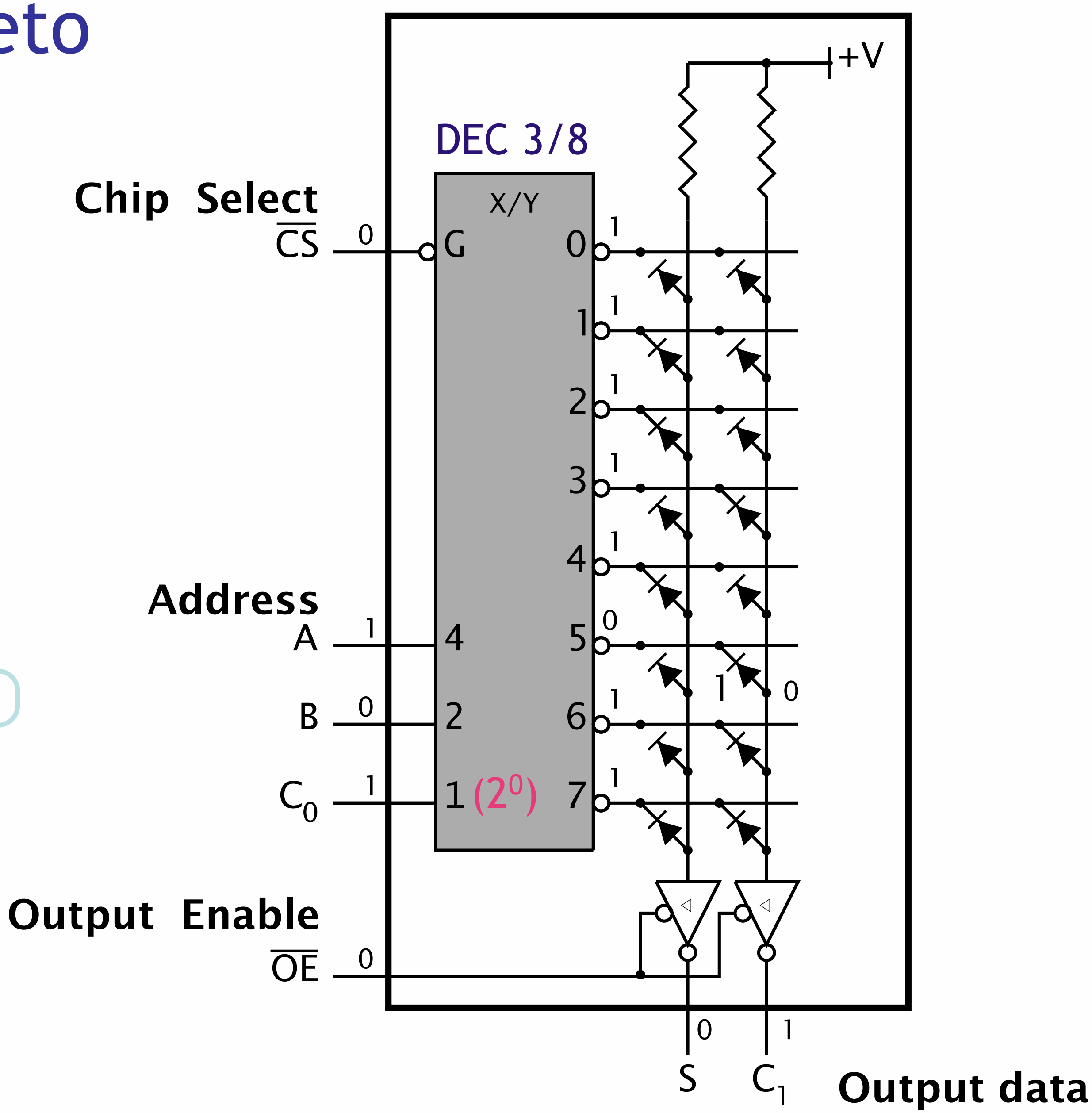


(b) The 74LS283 4-bit adder

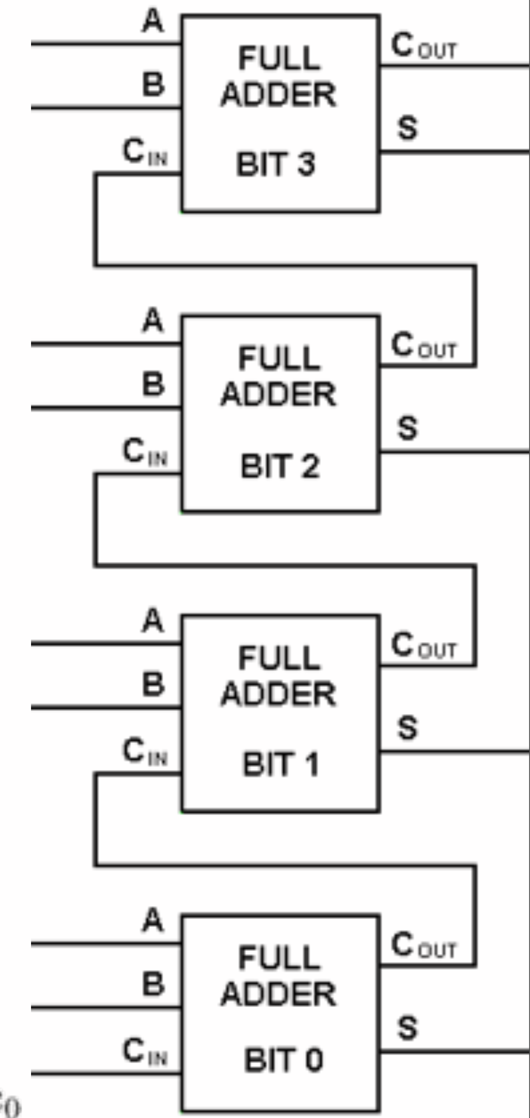
Somador Completo

Usando PROM (Programmable Read-Only Memory)

Entradas			Salidas	
C_{IN}	A	B	C_{OUT}	Σ
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



Somador Completo (*ripple-carry-adder*)



- Problema:

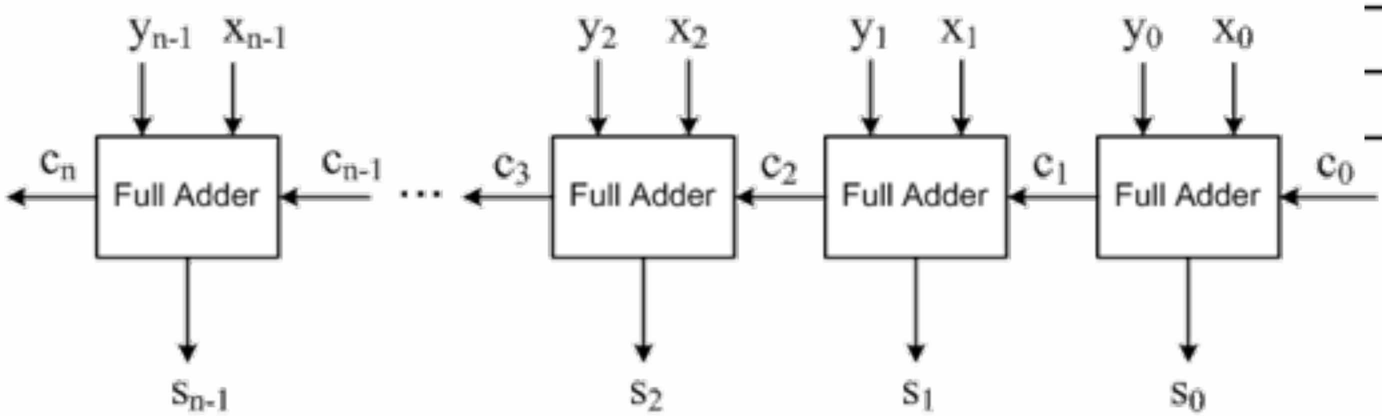
Atrasos de propagação de sinais, principalmente dos "*carry-out*" (estágios internos):

Tempo total do atraso = 2 nΔ = 8Δ = atraso de 8 portas
onde: Δ = atraso de propagação de 1 porta; 2Δ é o atraso associado com o tempo que cada somador completo leva para gerar seu carry-out.
Note que a saída de um somador completo com respeito ao bit na posição j é dado por:

$$S_j = X_j \oplus Y_j \oplus C_j$$

$$C_{j+1} = X_j \cdot Y_j + X_j \cdot C_j + Y_j \cdot C_j$$

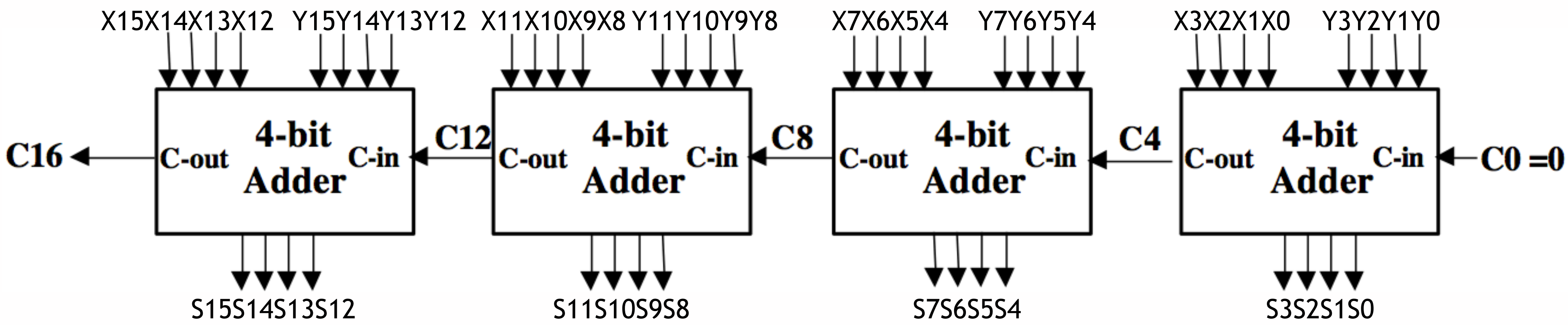
- *Ripple-Carry-Adder*:



Somador Completo (*ripple-carry-adder*)

- Exemplo:
  Somador de 16-bits, usando 4 x Somadores de 4-bits

- Soma duas palavras de entrada de 16-bits: X (bits X0 até X15) e Y (bits Y0 até Y15) produzindo o resultado de 16-bits S (bits S0 até S15) e 1-bit de Carry-Out na posição mais significativa: C16.



- Atraso total de propagação = 4 x propagação de cada somador de 4-bits;
  = 4 x 2 nΔ = 4 x 8Δ = 32Δ;
  = atraso de 32 portas!

• Carry Look-Ahead Adders, ou
Somadores Baseados em Antecipação do Sinal de Carry

- Desvantagem do ripple carry adder: o atraso na propagação do sinal de carry (2 *n*Δ) aumenta conforme aumenta o tamanho do somador (*n*-bits).
- Somadores do tipo "carry look adders" usam um método diferente para criar os bits de carry necessários para cada somador completo com a menor constante de atraso sendo igual ao atraso de 3 portas.
- O carry-out, C_{out} de cada somador completo na posição *i* ou C_{j+1} é dado por:

$$C_{out} = C_{i+1} = \boxed{X_i \cdot Y_i} + \boxed{(X_i + Y_i)} \cdot C_i$$

- Definindo que:

$\boxed{G_i = X_i \cdot Y_i}$ = Função geradora de carry da posição i (atraso de 1 porta)
Note que se $G_i = 1$, C_{i+1} será gerado independente do valor de C_j

$\boxed{P_i = X_i \cdot Y_i}$ = Função propagadora de carry da posição i (1 porta de atraso)
Note que se $P_i = 1$, C_i será propagado até C_{i+1}

- Usando função geradora de carry G_i e a função propagadora de carry P_i, então C_{i+1} pode ser escrito como:

$$C_{out} = C_{i+1} = \boxed{G_i} + \boxed{P_i} \cdot C_i$$

- Para eliminar o ripple de carry, o termo C_i é recursivamente expandido e multiplicado, resultando numa expressão AND-OR para cada C_{i+1}.

• Carry Look-Ahead Adders, ou
Somadores Baseados em Antecipação do Sinal de Carry

- Para um somador carry look-ahead de 4-bits, as expressões expandidas associadas com todos os bits de carry são dadas por:

$$C_{i+1} = \underbrace{X_i \cdot Y_i}_{G_i} + \underbrace{(X_i + Y_i)}_{P_i} \cdot C_i$$

$$C_1 = G_0 + P_0 \cdot C_0$$

$$C_2 = G_1 + P_1 \cdot C_1 = G_1 + P_1 \cdot G_0 + P_1 \cdot P_0 \cdot C_0$$

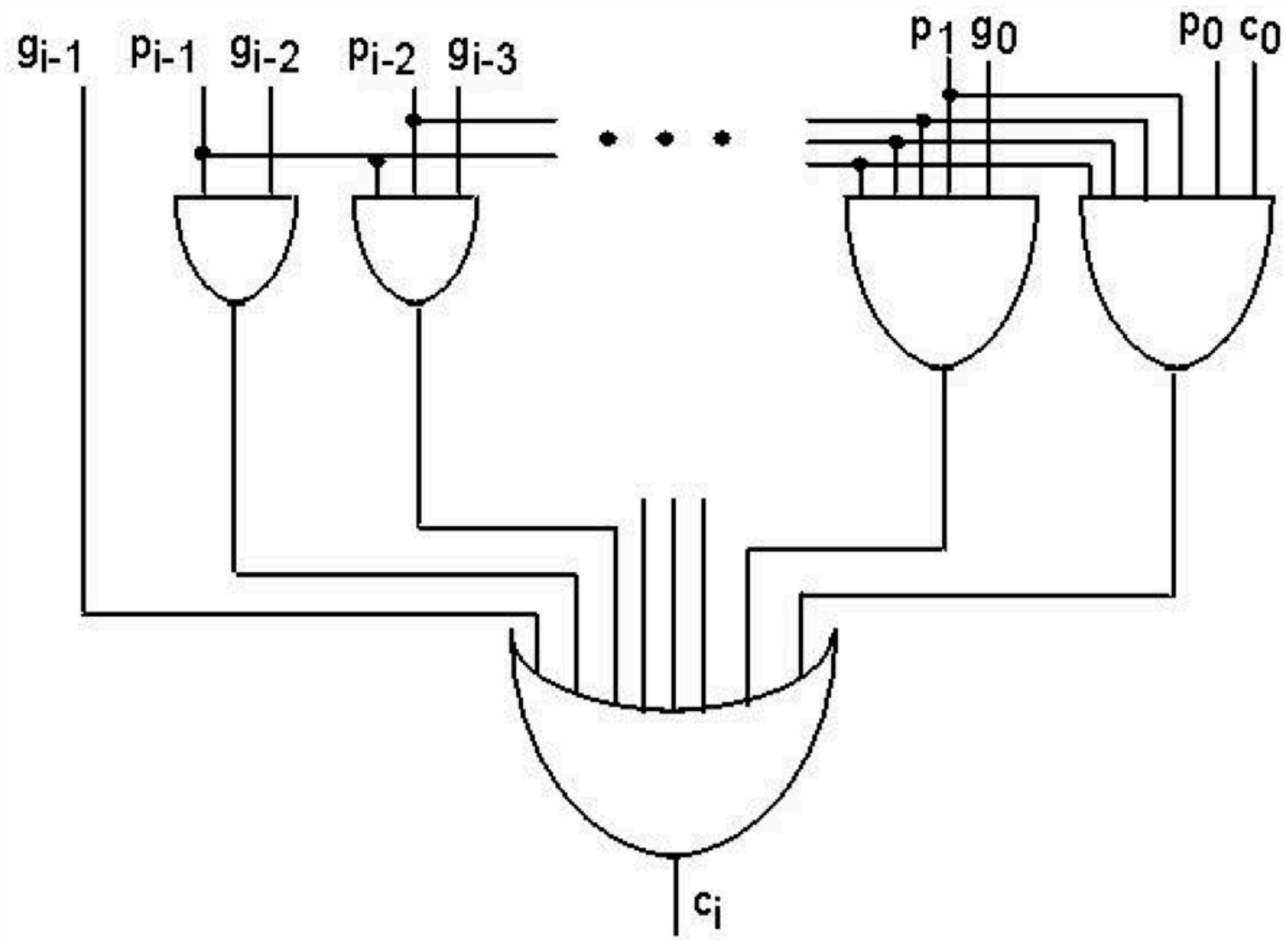
$$C_3 = G_2 + P_2 \cdot C_2 = G_2 + P_2 \cdot G_1 + P_2 \cdot P_1 \cdot G_0 + P_2 \cdot P_1 \cdot P_0 \cdot C_0$$

$$C_4 = G_3 + P_3 \cdot G_2 + P_3 \cdot P_2 \cdot G_1 + P_3 \cdot P_2 \cdot P_1 \cdot G_0 + P_3 \cdot P_2 \cdot P_1 \cdot P_0 \cdot C_0$$

- Os circuitos adicionais necessários para implementar as expressões são usualmente referencidas como a lógica do carry look-ahead.
- Usando a lógica de carry-ahead, todas os bits de carry estão disponíveis depois do atraso de propagação de 3 portas, **independente do tamanho do somador**.

• Carry Look-Ahead Adders, ou Somadores Baseados em Antecipação do Sinal de Carry

- Circuito:



$$C_i = G_{i-1} + P_{i-1}\,G_{i-2} + \ldots + P_{i-1}\,P_{i-2}\cdots P_1\,G_0 + P_{i-1}\,P_{i-2}\cdots P_0\,C_0$$

+ Ref. sobre Somadores de Carry Antecipado (opcional):

- Lab 3) Carry-Look-Ahed Adder (University of Pensylvania) Dispobible en 25/11/2008: http://www.seas.upenn.edu/~ese201/syll.html

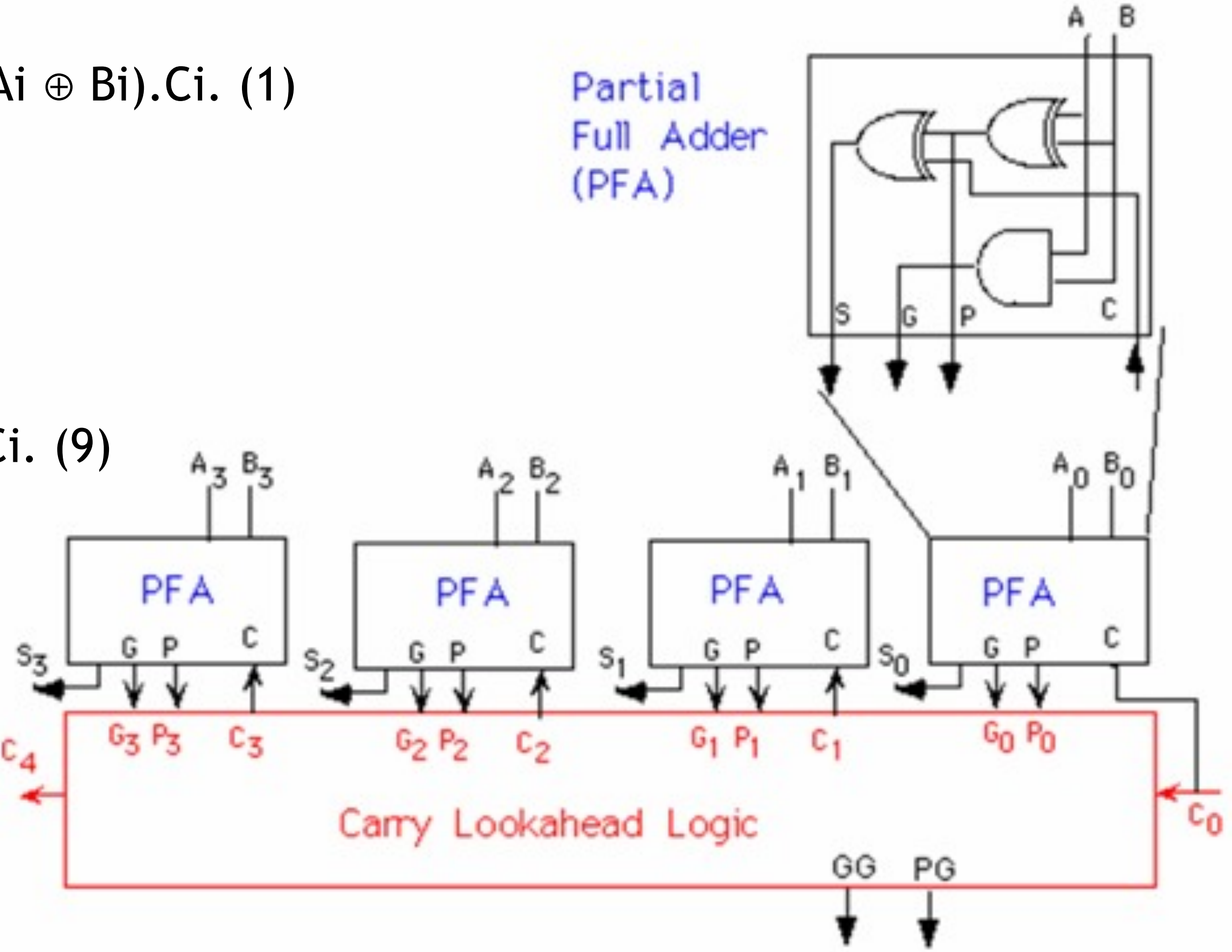
$COUT = C_{i+1} = A_i.B_i + (A_i \oplus B_i).C_i.$ (1)

$C_{i+1} = G_i + P_i.C_i$ (2)

$G_i = A_i.B_i$ (3)

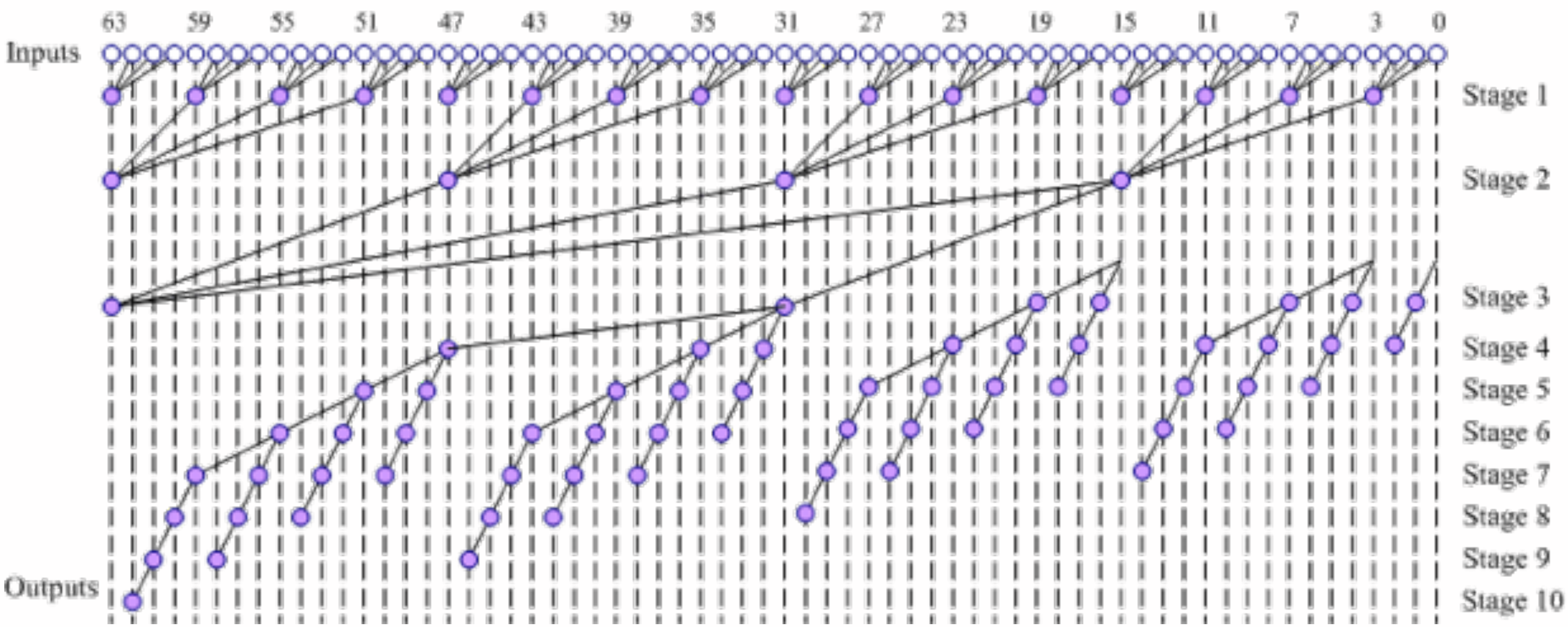
$P_i = (A_i \oplus B_i)$ (4)

$S_i = A_i \oplus B_i \oplus C_i = P_i \oplus C_i.$ (9)



+ Ref. sobre Somadores de Carry Antecipado (opcional):

- Hardware algorithms for arithmetic modules (http://www.aoki.ecei.tohoku.ac.jp/arith/mg/algorithm.html _ 25/11/2008) - Trata de Ripple Carry, Paraller prefix adders, conditional sum adder, fixed block size carry skip adder, variable block size carry skip adder)



sexta-feira, 25 de outubro de 13

Introdução

- Como representar **números com sinal** no sistema binário de numeração?

- 3 opções:
  - Notação sinal-magnitude;
  - Notação Complemento 1;
  - Notação Complemento 2.

1) Formato Sinal-Magnitude

- Exemplos:

	7	6	5	4	3	2	1	0
+52=	0	0	1	1	0	1	0	0

	7	6	5	4	3	2	1	0
-52=	1	0	1	1	0	1	0	0

No formato sinal-magnitude um número negativo possui os mesmos bits de magnitude que o correspondente número positivo, mas o bit de sinal é um “1” no lugar de um “0”.

1) Formato Sinal-Magnitude

- Problemas:

	7	6	5	4	3	2	1	0
+52=	0	0	1	1	0	1	0	0

	7	6	5	4	3	2	1	0
-52=	1	0	1	1	0	1	0	0

Ref	Binario	Sem Sinal	Com Sinal
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	8	-0
9	1001	9	-1
10	1010	10	-2
11	1011	11	-3
12	1100	12	-4
13	1101	13	-5
14	1110	14	-6
15	1111	15	-7

1) Formato Sinal-Magnitude

- Problemas:

	7	6	5	4	3	2	1	0
+52=	0	0	1	1	0	1	0	0
-52=	1	0	1	1	0	1	0	0

Ref	Binario	Sem Sinal	Com Sinal
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	8	-0 !?
9	1001	9	-1
10	1010	10	-2
11	1011	11	-3
12	1100	12	-4
13	1101	13	-5
14	1110	14	-6
15	1111	15	-7

2) Formato do Complemento-1 ($\overline{C_1}$)

- Exemplo:

	7	6	5	4	3	2	1	0
+52=	0	0	1	1	0	1	0	0

	7	6	5	4	3	2	1	0
-52=	1	1	0	0	1	0	1	1

No formato do complemento 1, um numero negativo é o complemento a 1 do correspondente número positivo.

2) Formato do Complemento-1 ($\overline{C_1}$)

- Problemas:

	7	6	5	4	3	2	1	0
+52=	0	0	1	1	0	1	0	0

	7	6	5	4	3	2	1	0
-52=	1	1	0	0	1	0	1	1

Ref	Binario	com Sinal	Sinal-Magnitude	Complemento-1
0	0000	0	0	0
1	0001	1	1	1
2	0010	2	2	2
3	0011	3	3	3
4	0100	4	4	4
5	0101	5	5	5
6	0110	6	6	6
7	0111	7	7	7
8	1000	8	-0	-7
9	1001	9	-1	-6
10	1010	10	-2	-5
11	1011	11	-3	-4
12	1100	12	-4	-3
13	1101	13	-5	-2
14	1110	14	-6	-1
15	1111	15	-7	-0

2) Formato do Complemento-1 ($\overline{C_1}$)

- Problemas:

7	6	5	4	3	2	1	0
0	0	1	1	0	1	0	0

+52=

7	6	5	4	3	2	1	0
1	1	0	0	1	0	1	1

-52=

Ref	Binario	sem Sinal	Sinal-Magnitude	Complemento-1
0	0000	0	0	0
1	0001	1	1	1
2	0010	2	2	2
3	0011	3	3	3
4	0100	4	4	4
5	0101	5	5	5
6	0110	6	6	6
7	0111	7	7	7
8	1000	8	-0	-7
9	1001	9	-1	-6
10	1010	10	-2	-5
11	1011	11	-3	-4
12	1100	12	-4	-3
13	1101	13	-5	-2
14	1110	14	-6	-1
15	1111	15	-7	-0 !?

2) Formato do Complemento-1 ($\overline{C_1}$)

- Problemas:

```
   11111110       -1
+  00000010       +2
 ..............  ...
 1 00000000        0  ← não é a resposta correta!
           1      +1  ← Somar carry (add carry)
 ..............  ...
   00000001        1  ← resposta correta
```

Para somar dois números neste sistema, se realiza uma soma convencional, mas se faz necessário voltar a somar o bit de carry-out (se ele foi ativado na primeira soma) para completar a soma final.

O exemplo acima demostra esta necesidade:

Considere o caso da soma de -1 (11111110) com +2 (00000010).

3) Formato do Complemento-2 ($\overline{C_2}$)

- Exemplo:

	7	6	5	4	3	2	1	0
+52=	0	0	1	1	0	1	0	0

Complemento -1

	7	6	5	4	3	2	1	0
-52=	1	1	0	0	1	0	1	1

+ 1

Complemento-2

	7	6	5	4	3	2	1	0
-52=	1	1	0	0	1	1	0	0

No formato do complemento-2, um número negativo é resultado do complemento do correspondente número positivo acrescido da soma de +1.

3) Formato do Complemento-2 ($\overline{C_2}$)

- Exemplo para 4-bits:

Ref	Binario	sem Sinal	Sinal-Magnitude	Comple-Mento-1	Comple-Ment- 2
0	0000	0	0	0	0
1	0001	1	1	1	1
2	0010	2	2	2	2
3	0011	3	3	3	3
4	0100	4	4	4	4
5	0101	5	5	5	5
6	0110	6	6	6	6
7	0111	7	7	7	7
8	1000	8	-0	-7	-8
9	1001	9	-1	-6	-7
10	1010	10	-2	-5	-6
11	1011	11	-3	-4	-5
12	1100	12	-4	-3	-4
13	1101	13	-5	-2	-3
14	1110	14	-6	-1	-2
15	1111	15	-7	-0	-1

3) Formato do Complemento-2 ($\overline{C_2}$)

- Exemplo:

	3	2	1	0
+ 3 =	0	0	1	1

Complemento-1

	3	2	1	0
	1	1	0	0

+ 1

Complemento-2

	3	2	1	0
- 3 =	1	1	0	1

Ref	Binario	sem Sinal	Sinal-Magnitude	Comple-Mento-1	Comple-Ment- 2
0	0000	0	0	0	0
1	0001	1	1	1	1
2	0010	2	2	2	2
3	0011	3	3	3	3
4	0100	4	4	4	4
5	0101	5	5	5	5
6	0110	6	6	6	6
7	0111	7	7	7	7
8	1000	8	-0	-7	-8
9	1001	9	-1	-6	-7
10	1010	10	-2	-5	-6
11	1011	11	-3	-4	-5
12	1100	12	-4	-3	-4
13	1101	13	-5	-2	-3
14	1110	14	-6	-1	-2
15	1111	15	-7	-0	-1

3) Formato do Complemento-2 ($\overline{C_2}$)

- Repare:

3 2 1 0

- 8 = $\boxed{1}\,\boxed{0}\,\boxed{0}\,\boxed{0}$

- 1

3 2 1 0

$\boxed{0}\,\boxed{1}\,\boxed{1}\,\boxed{1}$

Complemento

3 2 1 0

+ 8 = $\boxed{1}\,\boxed{0}\,\boxed{0}\,\boxed{0}$

Ref	Binario	sem Sinal	Sinal-Magnitude	Comple-Mento-1	Comple-Mento-2
0	0000	0	0	0	0
1	0001	1	1	1	1
2	0010	2	2	2	2
3	0011	3	3	3	3
4	0100	4	4	4	4
5	0101	5	5	5	5
6	0110	6	6	6	6
7	0111	7	7	7	7
8	1000	8	-0	-7	-8
9	1001	9	-1	-6	-7
10	1010	10	-2	-5	-6
11	1011	11	-3	-4	-5
12	1100	12	-4	-3	-4
13	1101	13	-5	-2	-3
14	1110	14	-6	-1	-2
15	1111	15	-7	-0	-1

3) Formato do Complemento-2 ($\overline{C_2}$)

- Exemplo:

Ref	Binario	sin Signo	Signo-Magnitud	Complemento 1	Complemento 2
0	0000	0	0	0	0
1	0001	1	1	1	1
2	0010	2	2	2	2
3	0011	3	3	3	3
4	0100	4	4	4	4
5	0101	5	5	5	5
6	0110	6	6	6	6
7	0111	7	7	7	7
8	1000	8	-0	-7	-8
9	1001	9	-1	-6	-7
10	1010	10	-2	-5	-6
11	1011	11	-3	-4	-5
12	1100	12	-4	-3	-4
13	1101	13	-5	-2	-3
14	1110	14	-6	-1	-2
15	1111	15	-7	-0	-1

Binario	Complemento-2	Interpretação sem sinal.
00000000	0	0
00000001	1	1
...	...	...
01111110	126	126
01111111	127	127
10000000	−128	128
10000001	−127	129
10000010	−126	130
...	...	...
11111110	−2	254
11111111	−1	255

3) Formato do Complemento-2 ($\overline{C_2}$)

- Exemplo:

	7	6	5	4	3	2	1	0
+52=	0	0	1	1	0	1	0	0

	7	6	5	4	3	2	1	0
-52=	1	1	0	0	1	0	1	1

+ 1

	7	6	5	4	3	2	1	0
-52=	1	1	0	0	1	1	0	0

Notar que:
Con 16 bits é possível representar números entre −32,768 à 32,767, e com 32 bits é possível codificar números entre −2,147,483,648 à 2,147,483,647.
Lembrar de Linguagem C: variáveis do tipo "*int*"

Ref	Binario	sem Sinal	Sinal-Magnitude	Comple-Mento-1	Comple-Mento-2
0	0000	0	0	0	0
1	0001	1	1	1	1
2	0010	2	2	2	2
3	0011	3	3	3	3
4	0100	4	4	4	4
5	0101	5	5	5	5
6	0110	6	6	6	6
7	0111	7	7	7	7
8	1000	8	-0	-7	-8
9	1001	9	-1	-6	-7
10	1010	10	-2	-5	-6
11	1011	11	-3	-4	-5
12	1100	12	-4	-3	-4
13	1101	13	-5	-2	-3
14	1110	14	-6	-1	-2
15	1111	15	-7	-0	-1

Soma/Subtração:

- Exemplos:

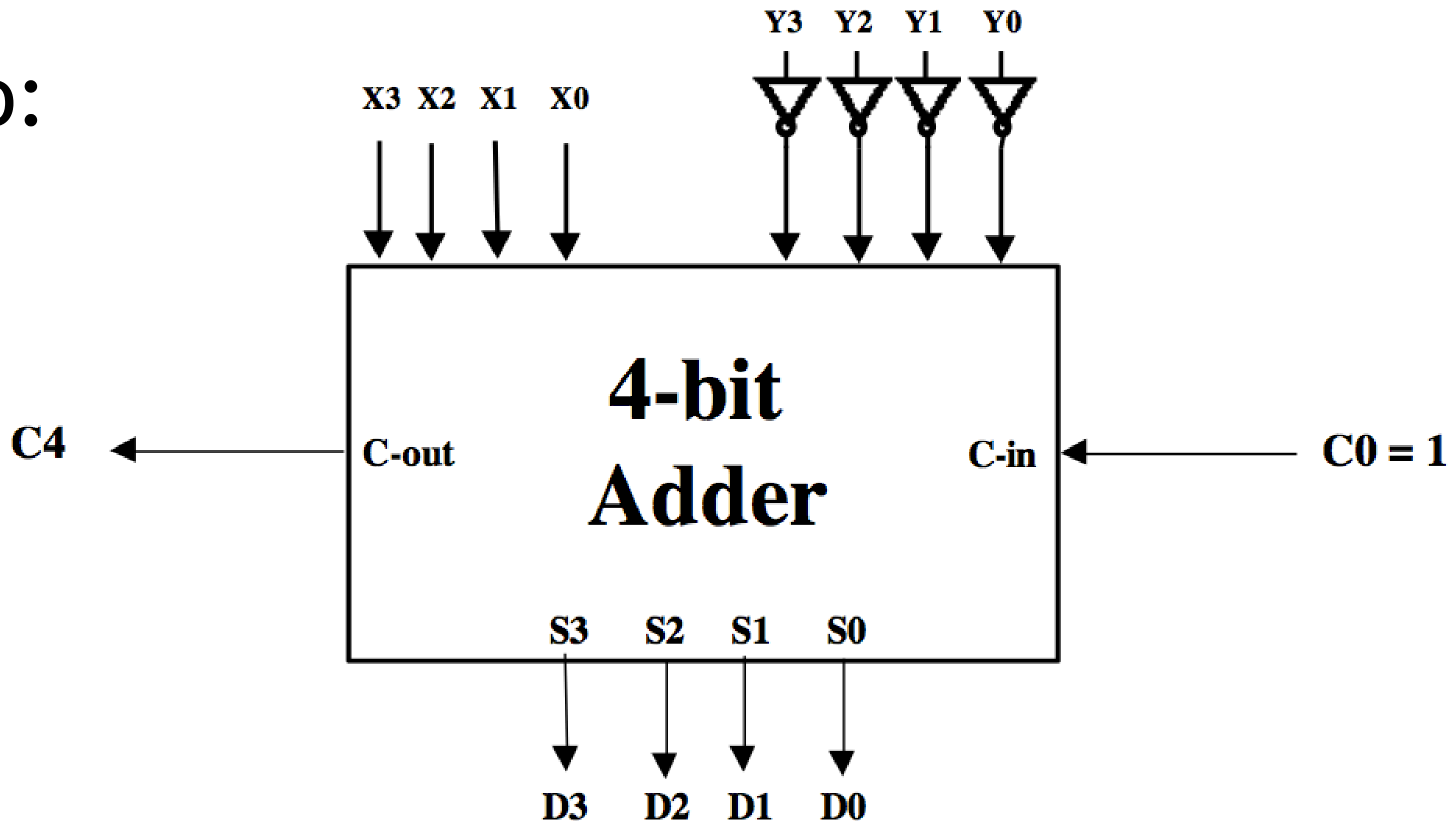
```
         1 1111 111.  ←  carry-outs
  7        0000 0111
 -5     +  1111 1011  ←  +5=0000 0101 →  -5=1111 1010 (2,C1)
 ...    .............
  2        0000 0010


                  1   ←  carry-outs
  5        0000 0101
 -7     +  1111 1001  ←  +7=0000 0111 →  -7=1111 1000 (2,C1)
 ...    ............. ...
 -2        1111 1110 →   -1 →  1111 1101 →  0000 0010 = 2 (10)
```

Interpretação sem sinal

Subtração:

- Exemplo:

X
−Y
. .
D →



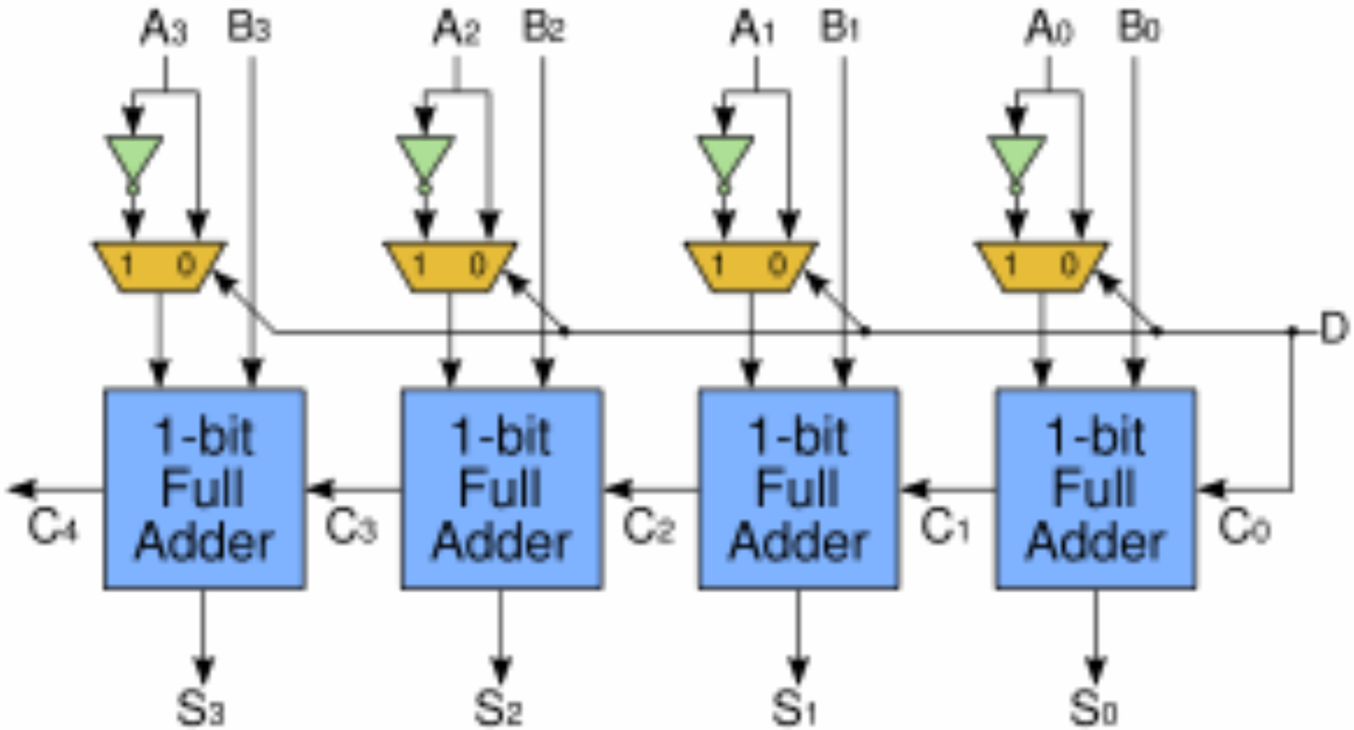

```
                 1   ←  carry-outs
 5         0000 0101
-7       + 1111 1001 ←  +7=0000 0111 →  -7=1111 1000 (2,C1)
...      .............. ...
-2         1111 1110 →    -1 →  1111 1101 →  0000 0010 = 2 (10)
```

Circuto de $\overline{\text{Soma}}$/Subtração

- Exemplo:

B
−A
. . →
S

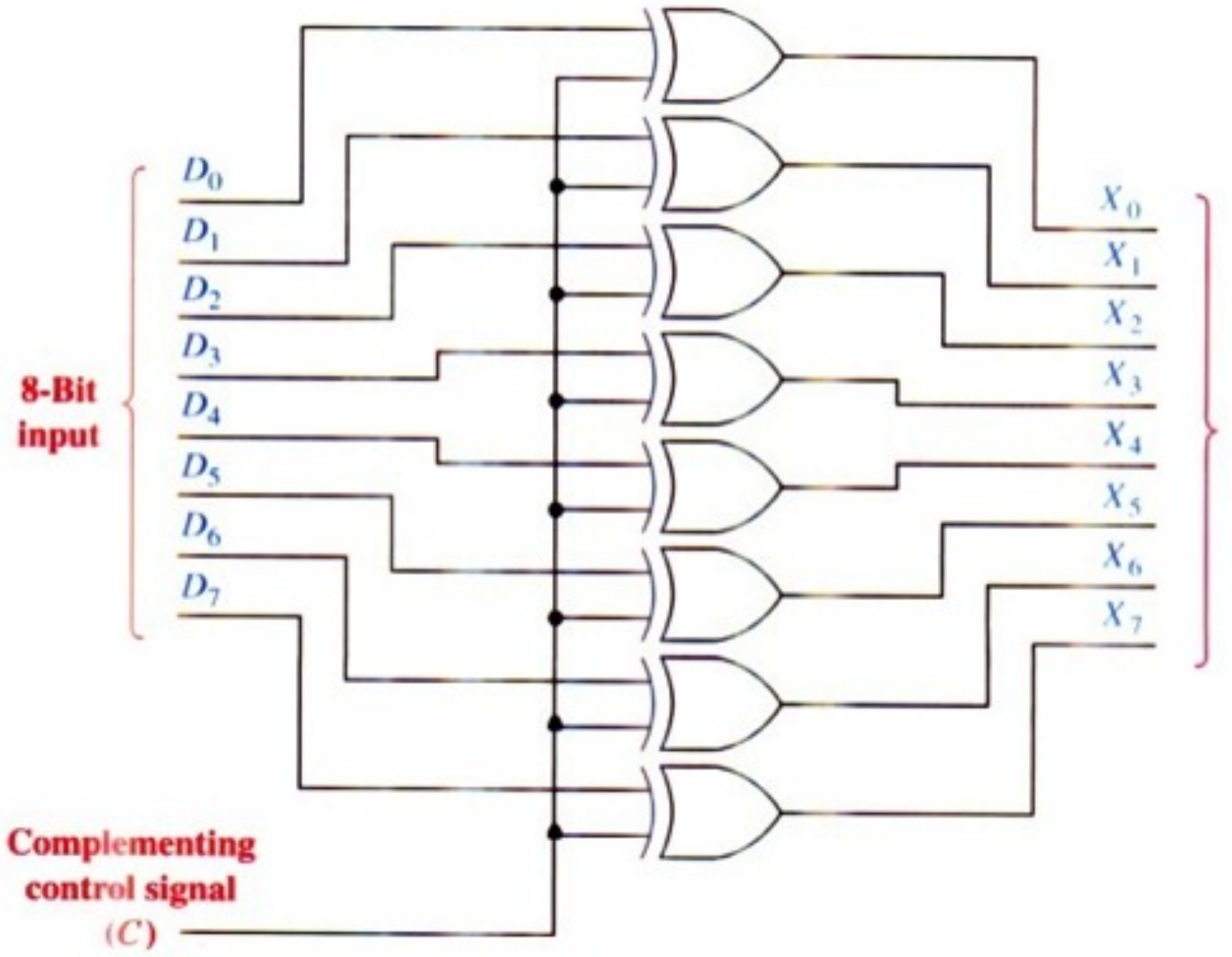
Neste caso:
D=1




```
                1   ←  carry-outs
 5       0000 0101
-7     + 1111 1001 ←  +7=0000 0111 →  -7=1111 1000 (2,c1)
...    ............ ...
-2       1111 1110 →    -1 →  1111 1101 →  0000 0010 = 2 (10)
```

Sumador/Subtrador:

Uso de "Inversor Controlado":



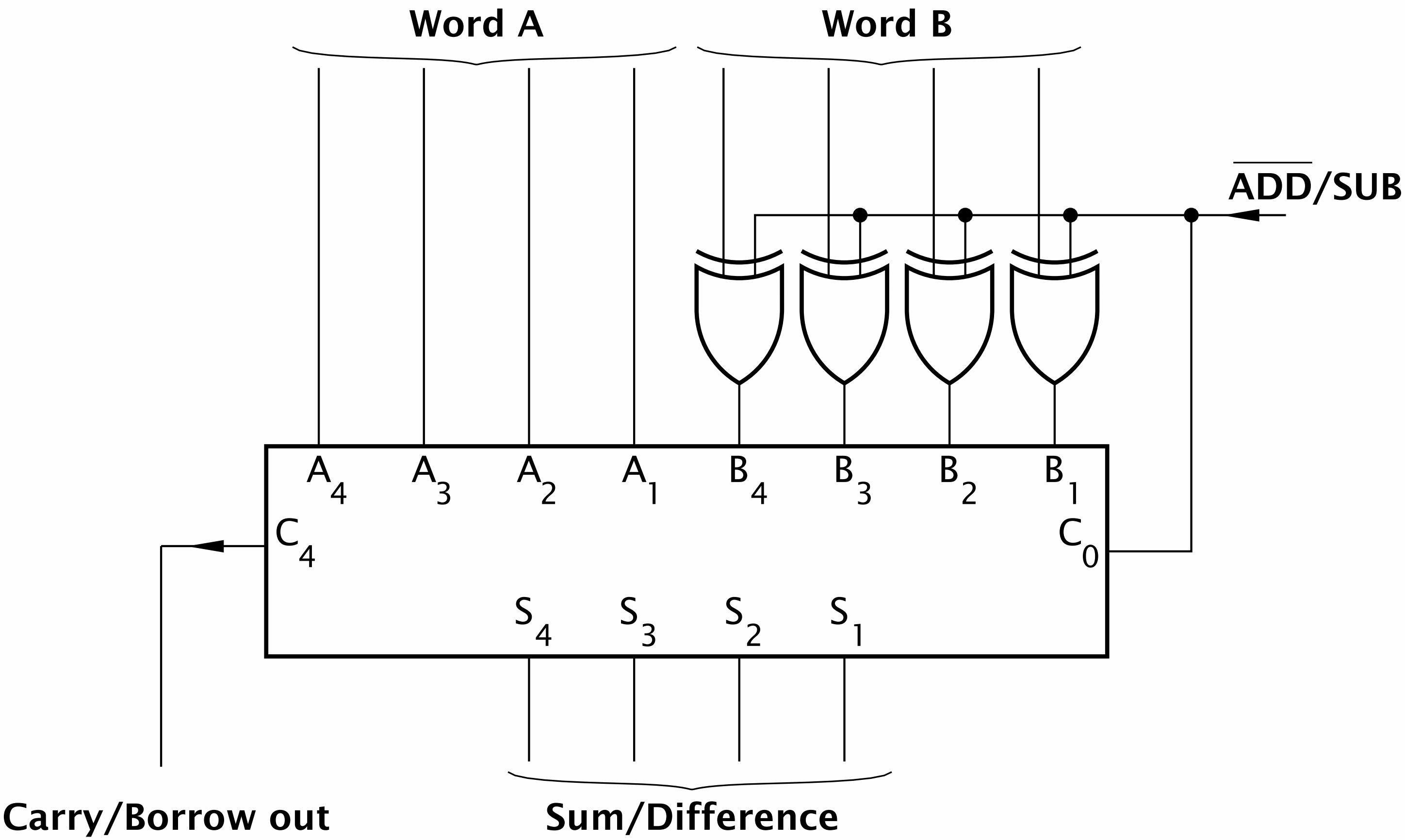
Saída Controlada:

$$Si\ C = 1,\ X_{0-7} = \overline{D_{0-7}}$$
$$Si\ C = 0,\ X_{0-7} = D_{0-7}$$

Fig.: Exemplo de solução típica empregada em circuitos aritméticos binários.

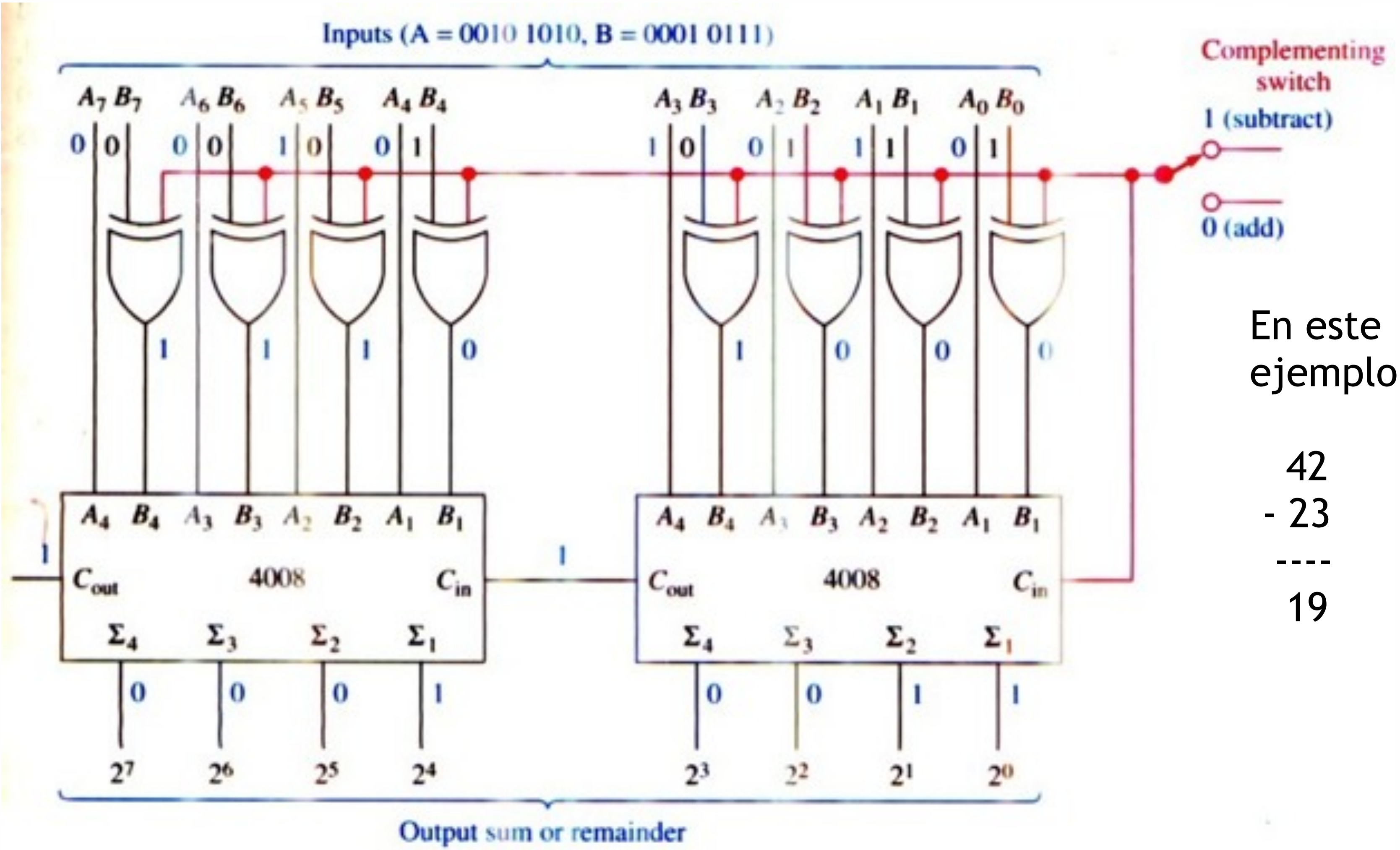
Circuto de $\overline{\text{Soma}}$/Subtração

Usando "Inversor Controlado":



Somador/Subtrador:

- Circuito de soma/substração: $\Sigma=A+B$ ou $\Sigma=A-B$

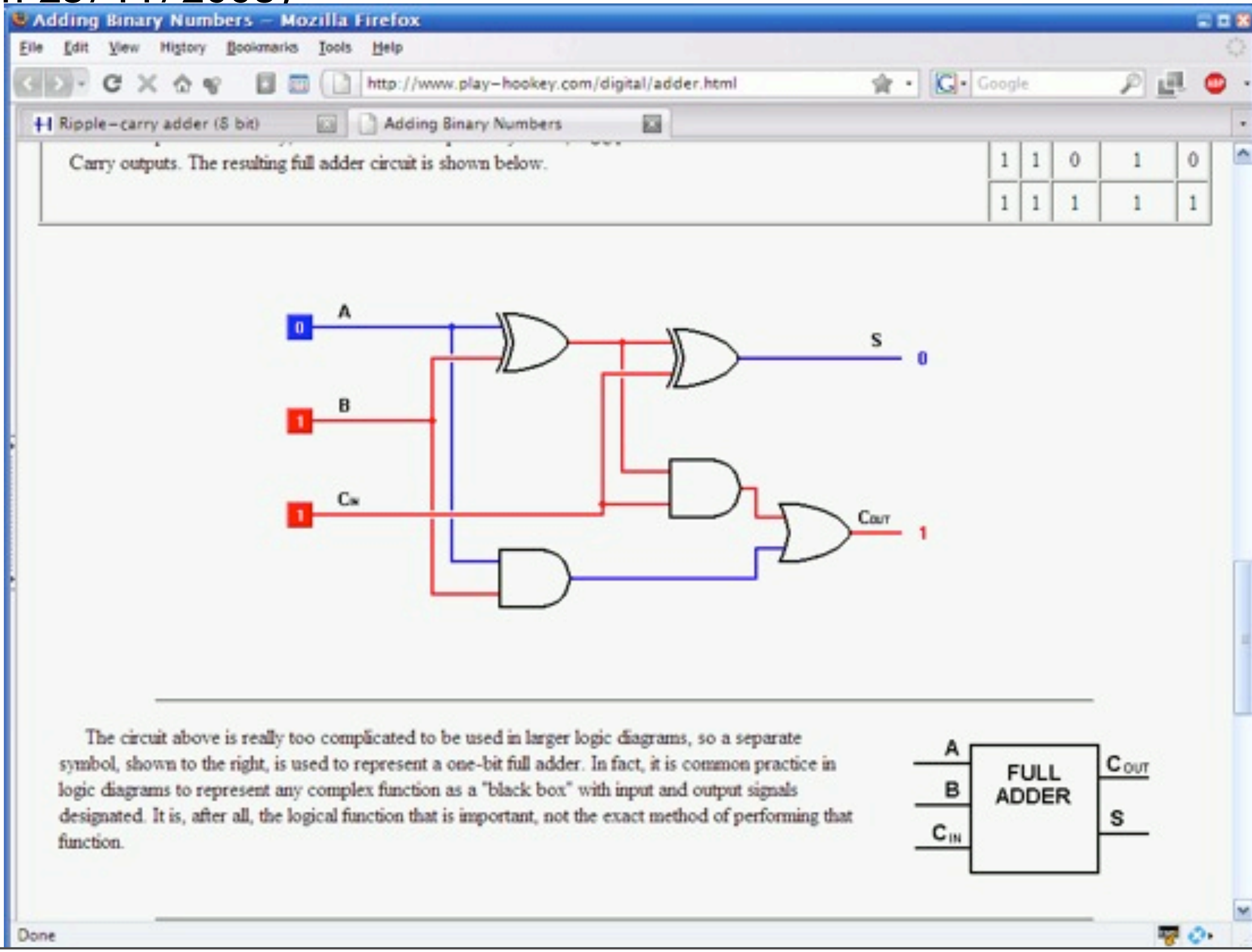


En este ejemplo:

$$\begin{array}{r} 42 \\ -\ 23 \\ \hline 19 \end{array}$$

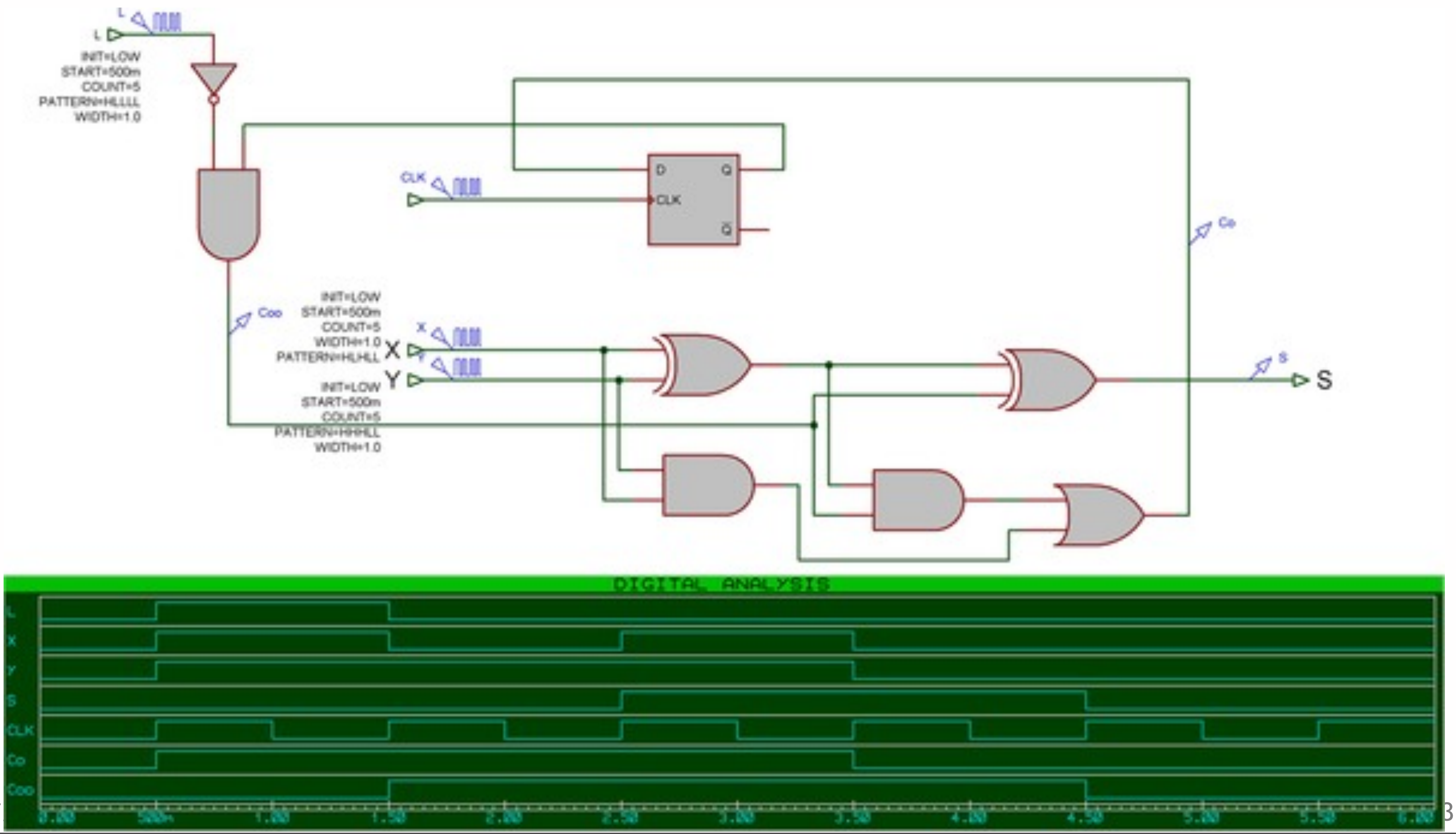
Simuladores em Applets-Java:

- http://www.play-hookey.com/digital/adder.html
  (Disponível em 25/11/2008)

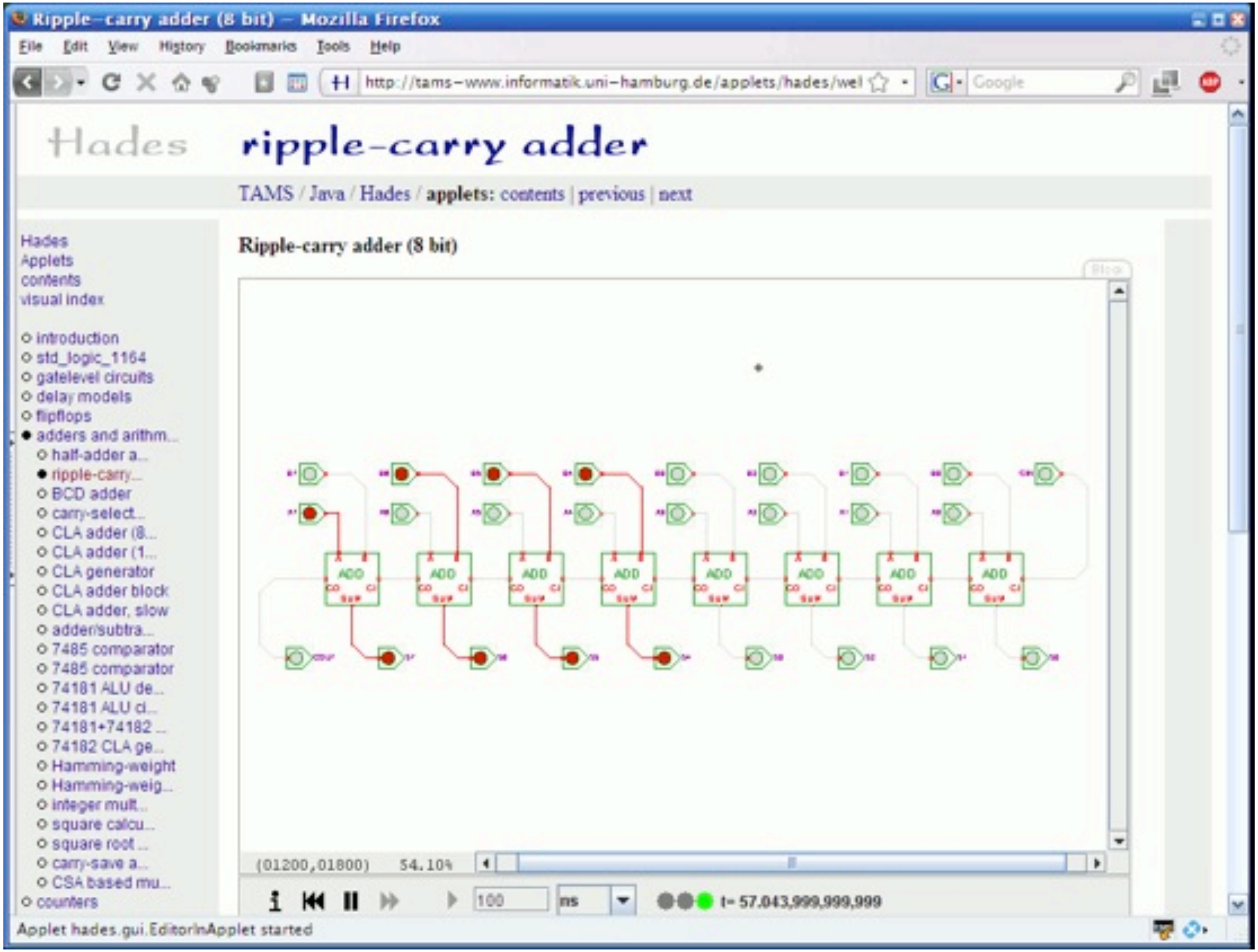


+ Ref.:

- Microprocessor Design/Add and Subtract Blocks, trata de circuitos de adição e subtração http://en.wikibooks.org/wiki/Microprocessor_Design/Add_and_Subtract_Blocks - Disponível em 25/11/2008



Simuladores em Applets-Java:



http://tams-www.informatik.uni-hamburg.de/applets/hades/webdemos/20-arithmetic/10-adders/ripple.html
Disponível em 25/11/2008.

Simuladores em Applets-Java:

- http://tams-www.informatik.uni-hamburg.de/applets/hades/webdemos/20-arithmetic/20-carryselect/adder_carryselect.html
  (Disponível em 25/11/2008)