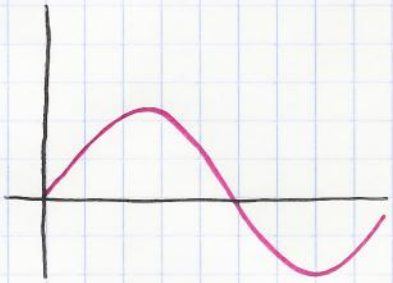
SISTEMAS DIGITAIS

Aula_01
Introdução

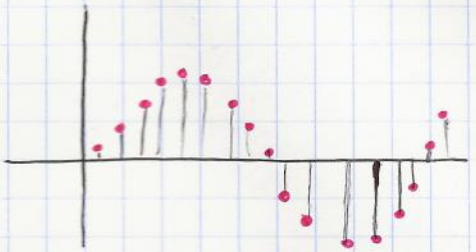
Sinal Analógico → Sinal Digital

Amostragem



D : IR

Sinal Contínuo



D : ℚ

Sinal Discreto

Aula_02
Sistemas de Numeração e Códigos

Componentes de um sistema de numeração:
- Base
- Alfabeto Ordenado
- Número (seq. de dígitos)
- Valor do Dígito (peso)

Base b → Base 10

$n_3\ n_2\ n_1\ n_0$ → $n_3 b^3 + n_2 b^2 + n_1 b + n_0$

$0 < n < b$

Aula 02

Sistemas de numeração e Códigos

(continuação)

Base 10	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Base 16	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10
Base 2	0	1	10	11	100	101	110	111	1000	1001	1010	1011	1100	1101	1110	1111	10000
Base 3	0	1	2	10	11	12	20	21	22	100	101	102	110	111	112	120	121
Base 4	0	1	2	3	10	11	12	13	20	21	22	23	30	31	32	33	100

Conversão CBN ⇆ DEC e BCD ⇄ HEX

DEC → CBN

- Subtrações sucessivas pelas potências de 2 — Ou
- Divisões sucessivas por 2

↳ Parte inteira

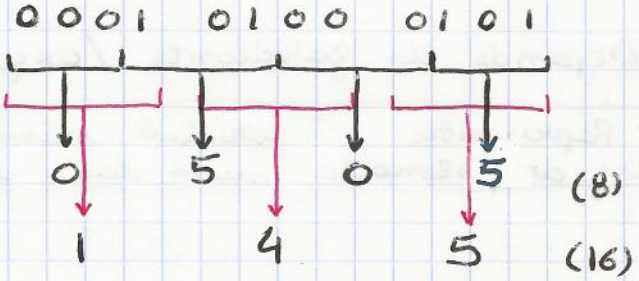
- Multiplicações sucessivas por 2 (tomando a parte inteira do produto significativa e a parte decimal do produto o novo nº a multiplicar)

↳ Parte decimal

↳ Métodos utilizados também na conversão DEC → HEX

CBN → Qualquer base 2^t

- Agrupar os dígitos em grupos de t elementos e proceder para a conversão em decimal



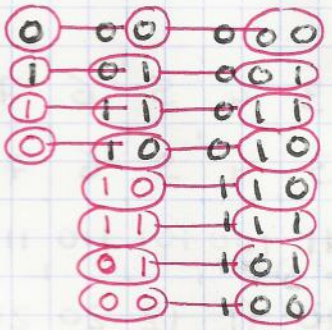
Códigos Binários

- CBN (C. B. Natural)
- CBR (C.B. refletido / Código de Gray)

0	000	000
1	001	001
2	010	011
3	011	010
4	100	110
5	101	111
(...)		

Não Posicional

Posicional



- BCD (Binary - coded decimal)

0001 0010 (BCD)

1 2

12

cada 4 bits representam 1 algarismo decimal

↳ Nibble

Códigos Alfanuméricos 7x16

- ASCII
  - ASCII - 7 bits
    - Tabela 7x16 que permite codificar certos caracteres alfa-numéricos em números de base 16 (hexadecimal)
  - Extended ASCII
- KOI (ASCII soviético / Russo)
  - KOI-7
  - KOI-8
- ISO-8859-1 (codificação do Alfabeto Latino)

Word de um processador

- Depende do fabricante / arquitetura
- Representa a unidade mínima processada ou armazenada num dado sistema

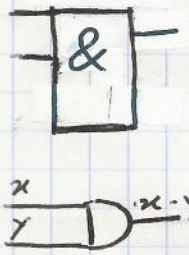
(Aula - 03)
Álgebra de Boole

Operações Básicas

Portas Lógicas correspondentes

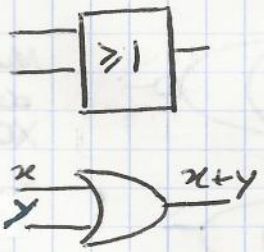
AND → produto lógico

x	y	x · y
0	0	0
0	1	0
1	0	0
1	1	1



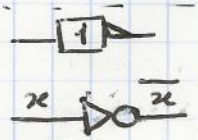
OR → soma lógica

x	y	x + y
0	0	0
0	1	1
1	0	1
1	1	1



NOT → complemento

x	$\bar{x}$
0	1
1	0



Propriedades Características da Álgebra Booleana

- **Distributividade da soma**

$$x + (y \cdot z) = x + y \cdot x + z$$

- **Leis de De Morgan**

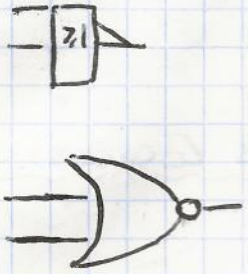
$$\overline{x + y} = \bar{x} \cdot \bar{y} \quad \wedge \quad \overline{x \cdot y} = \bar{x} + \bar{y}$$

- **Adjacência**

$$x \cdot y + x \cdot \bar{y} =$$
$$= (x) \cdot (y + \bar{y}) =$$
$$= (x) \cdot 1 =$$
$$= x$$

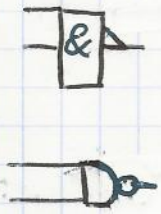
NOR

x	y	$\overline{x+y}$
0	0	1
0	1	0
1	0	0
1	1	0



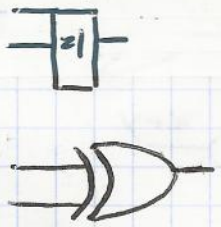
NAND

x	y	$\overline{x \cdot y}$
0	0	1
0	1	1
1	0	1
1	1	0



XOR

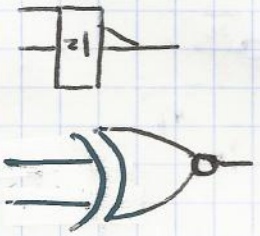
x	y	$x \oplus y$
0	0	1
0	1	0
1	0	0
1	1	1



Numa porta de n entradas XOR =1 se o número de entradas for impar

XNOR

x	y	$x \odot y$
0	0	0
0	1	1
1	0	1
1	1	0

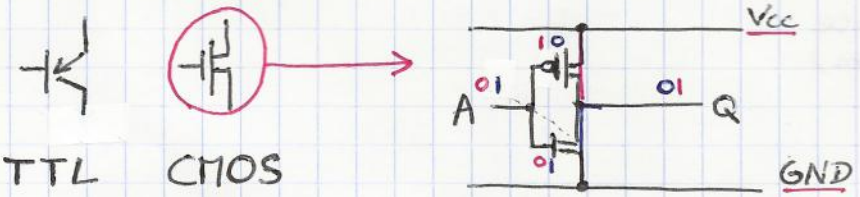


Aula - 04
Elementos da Tecnologia

CI
- Classificação por nº de portas ( )
- Classificação por tecnologia utilizada (TTL , CMOS , (...))

Compostos por ...

Transistores
- "Torneira"
- Permitem fazer fisicamente portas lógicas



PORTA NOT EM CMOS

Nível lógico

V_{xy} → Tensão

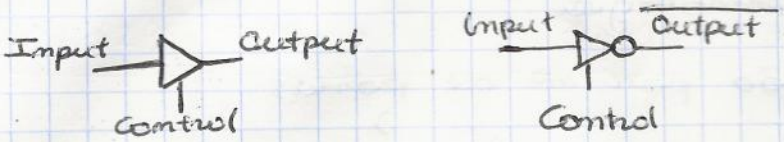
- x: 0 - Output → saída; I - Input → entrada
- y: H - High → Máxima; L - Low → Mínima

Os limites quantitativos variam de acordo com a família lógica

Fan-In → Nº de entradas disponíveis (de uma porta log)

Fan-Out → Nº de entradas que podem ser disponibilizadas sem degenerar o funcionamento do circuito

Tri-State Buffer
(Buffer de 3 estados)

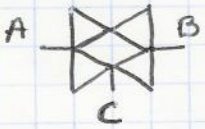


Crtl	IN	OUT	$\overline{OUT}$
0	ϕ	OFF	OFF
1	0	0	1
1	1	1	0

→ Ponto de saída em alta impedância

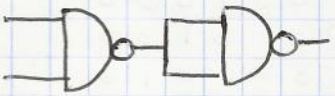
↳ Permite a selecção de sinais
↳ Permite realizar linhas bidirecionais funcionais

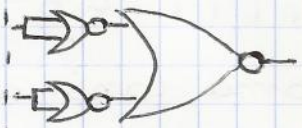
Transmission gate
(Porta de passagem)



Aula – 05
Funções lógicas

NAND & NOR LOGIC

PORTA BASE	NAND LOGIC	
NOT		$\overline{A \cdot A} = \overline{A}$
AND		$\overline{\overline{A \cdot B}} = A \cdot B$
OR		$\overline{\overline{A} \cdot \overline{B}} = A + B$

PORTA BASE	NOR LOGIC	
NOT		$\overline{A + A} = \overline{A}$
OR		$\overline{\overline{A + B}} = A + B$
AND		$\overline{\overline{A} + \overline{B}} = A \cdot B$

Forma canónica da forma normal disjuntiva

$x_1 x_2 + \overline{x_1}\,\overline{x_2}\,\overline{x_3} =$

$x_1 x_2 x_3 + x_1 x_2 \overline{x_3} + \overline{x_1}\,\overline{x_2}\,\overline{x_3}$

Contém todos os literais da função

x_3	x_2	x_1	
0	0	0	$\overline{x_3}\,\overline{x_2}\,\overline{x_1}$
0	0	1	$\overline{x_3}\,\overline{x_2}\,x_1$
0	1	0	$\overline{x_3}\,x_2\,\overline{x_1}$
0	1	1	$\overline{x_3}\,x_2\,x_1$
1	0	0	$x_3\,\overline{x_2}\,\overline{x_1}$
1	0	1	$x_3\,\overline{x_2}\,x_1$
1	1	0	$x_3\,x_2\,\overline{x_1}$
1	1	1	$x_3\,x_2\,x_1$

→ Tabelas de verdade correspondentes aos mintermos

m_0	m_1	m_2	m_3	m_4	m_5	m_6	m_7
1	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0
0	0	1	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	0	0	1	0	0	0
0	0	0	0	0	1	0	0
0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	1

Mintermo → Produto canónico

Forma canónica da forma normal conjuntiva

$(x_1 + x_2) \cdot (\overline{x_1} + \overline{x_2} + \overline{x_3}) =$

$= (x_1 + x_2 + x_3) \cdot (x_1 + x_2 + \overline{x_3}) \cdot (\overline{x_1} + \overline{x_2} + \overline{x_3})$

x_3	x_2	x_1	
0	0	0	$x_3 + x_2 + x_1$
0	0	1	$x_3 + x_2 + \overline{x_1}$
0	1	0	$x_3 + \overline{x_2} + x_1$
0	1	1	$x_3 + \overline{x_2} + \overline{x_1}$
1	0	0	$\overline{x_3} + x_2 + x_1$
1	0	1	$\overline{x_3} + x_2 + \overline{x_1}$
1	1	0	$\overline{x_3} + \overline{x_2} + x_1$
1	1	1	$\overline{x_3} + \overline{x_2} + \overline{x_1}$

M_0	M_1	M_2	M_3	M_4	M_5	M_6	M_7
0	1	1	1	1	1	1	1
1	0	1	1	1	1	1	1
1	1	0	1	1	1	1	1
1	1	1	0	1	1	1	1
1	1	1	1	0	1	1	1
1	1	1	1	1	0	1	1
1	1	1	1	1	1	0	1
1	1	1	1	1	1	1	0

Maxtermos → soma canónica

$$m_j = \overline{M_j}$$

Funções Incompletamente Especificadas

x_3	x_2	x_1	f
0	0	0	X_1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	X_2

→ Dont care

$$\sum_{(3,6)} + \sum_{d(0,7)} = m_3 + m_6 + m d_0 + m d_7$$

$$\prod_{(1,2,4,5)} \cdot \prod_{d(0,7)} = M_1 \cdot M_2 \cdot M_4 \cdot M_5 \cdot M d_0 \cdot M d_7$$

Função que detecta se a multiplicidade por 3 do valor é verdadeira

Lado de um d_7

Conversão para a praticalidade

$X \begin{cases} 0 \\ 1 \end{cases}$, logo a conversão para 0 ou 1 de X_1 e X_2 é necessária (sendo que a conversão para 0 ou 1 depende da simplificação).

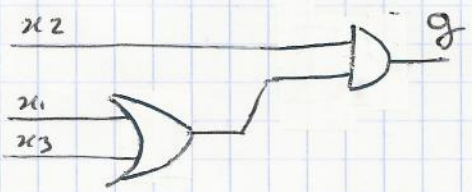
f →	g
X_1 →	0
0	0
0	0
1	1
0	0
0	0
1	1
X_2 →	1

$$g = \sum(3,6,7) =$$

$$= (\bar{x}_3 x_2 x_1) + (x_3 x_2 \bar{x}_1) + (x_3 x_2 x_1) =$$

$$= (x_2 x_1) + (x_3 x_2) =$$

$$= x_2 (x_1 + x_3)$$



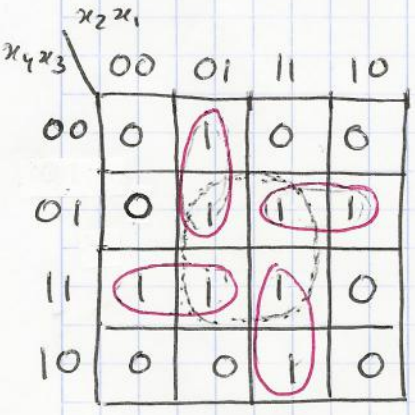
Aula 06
Minimização de Funções Booleanas

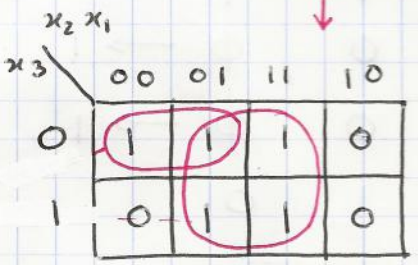
x_3	x_2	x_1	f
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Reordenação da Tabela →

x_3	x_2	x_1	f
0	0	0	1
0	0	1	1
0	1	1	1
0	1	0	0
1	1	0	0
1	1	1	1
1	0	1	1
1	0	0	0

x_4	x_3	x_2	x_1	f
0	0	0	0	0
0	0	0	1	1
0	0	1	1	0
0	0	1	0	0
0	1	1	0	0
0	1	1	1	1
0	1	0	1	1
0	1	0	0	1
1	1	0	0	1
1	1	0	1	1
1	1	1	1	1
1	1	1	0	0
1	0	1	0	0
1	0	1	1	0
1	0	0	1	1
1	0	0	0	0

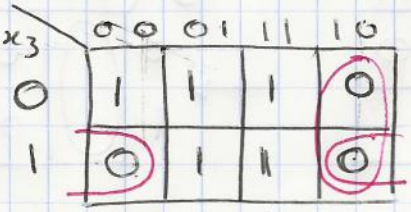




Agrupamento de Minitermos

$$\overline{x_2}\,\overline{x_3} + x_1$$

ou

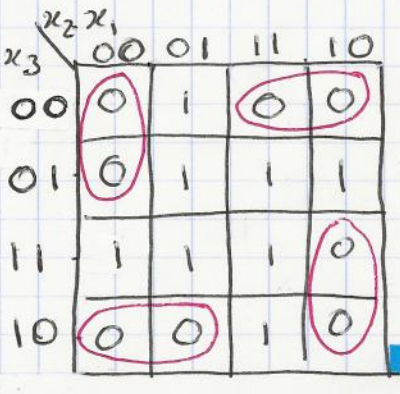


Agrupamento de Maxtermos

$$\overline{x}_3 + x_1 \cdot \overline{x}_2 + x_1 =$$

$$= (\overline{x_2}\,\overline{x_3}) + x_1$$

- Implicantes primos essenciais
- Implicantes primos
- Implicados primos essenciais



Algoritmo de Minimização

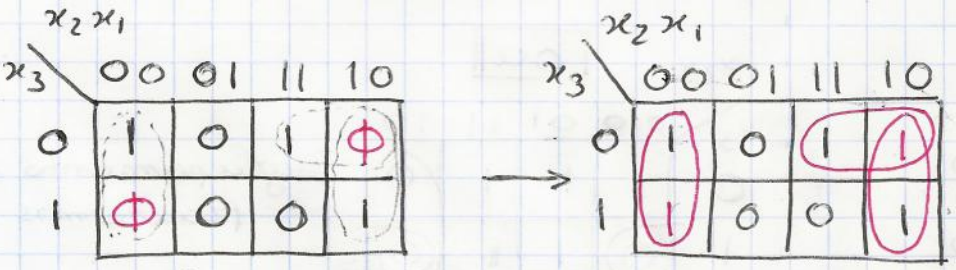
1- Identificação dos dos implicantes/cados primos essenciais

2- Agrupamento dos implicantes/cados restantes

3- Conclusão e escrita da expressão minimizada

Funções Incompletamente Especificadas

x_3	x_2	x_1	f
0	0	0	1
0	0	1	0
0	1	1	1
0	1	0	$\phi \rightarrow 1$
1	1	0	$\phi \rightarrow 1$
1	1	1	0
1	0	1	0
1	0	0	1



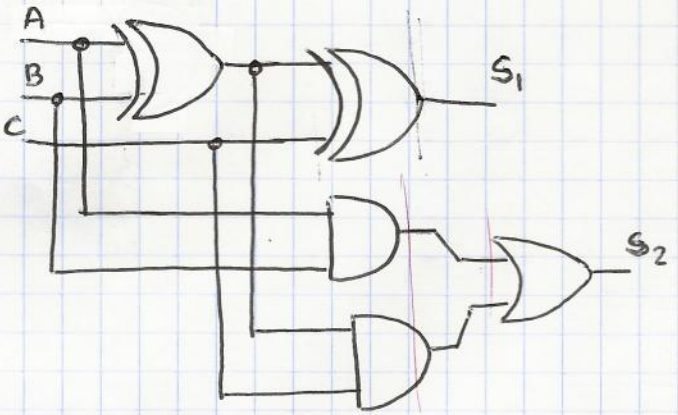
Para

A Completar os agrupamentos torna-se vantajoso, por exemplo a conversão dos DON'T CARE (ϕ) para 1

Aula_07
Circuitos Combinatórios e Tempo de Propagação

Circuito Combinatório

- Circuito sem memória associada, logo a saída apenas depende das entradas

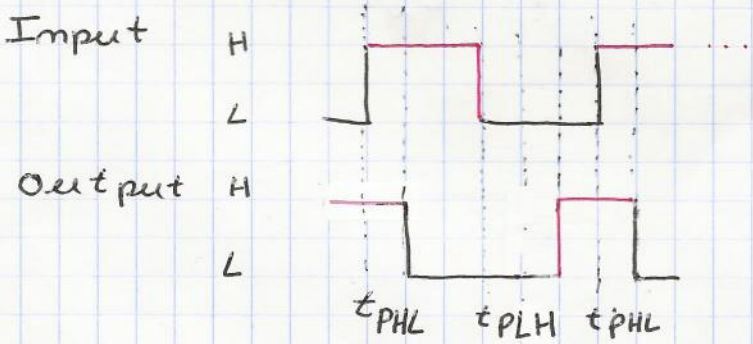


IDEALMENTE

$t_{S_1} = t_{S_2} = \delta_s$ ↓ Mudança insL

Conteudo numa situação REAL

$t_{S_1} < t_{S_2} \neq \delta_s$



Logo verifica-se que $t_{PLH} = t_{PHL}$ não é necessáriamente verdade

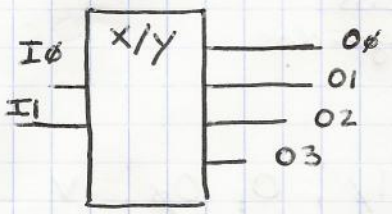
Pela ideia de Fan-out um aumento no nº de portas ligadas a uma saída aumenta o tempo de propagação da referida saída ...

Aula 08

Encoders , Decoders
Multiplexers , Demultiplexers

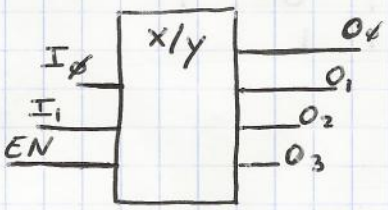
Decoder

• Exemplo 2:4



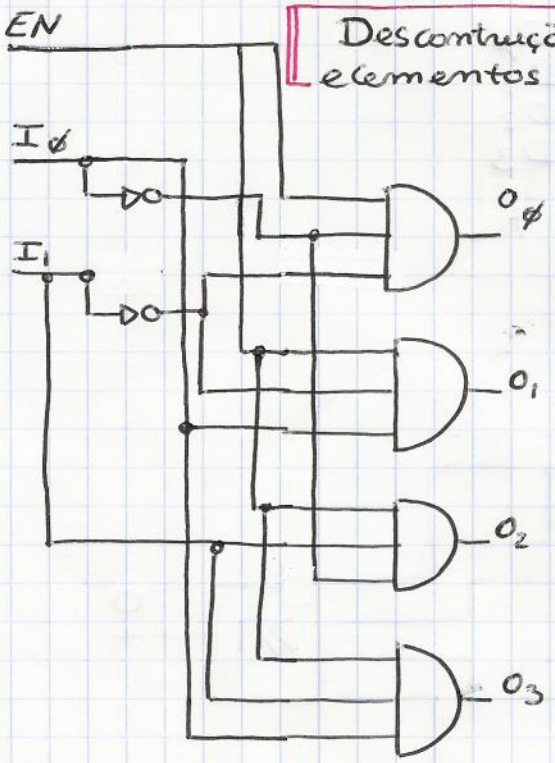
I_1	I_0	O_0	O_1	O_2	O_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

• Exemplo - 2:4 , com entrada Enable



EN	I_1	I_0	O_0	O_1	O_2	O_3
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1
0	ϕ	ϕ	0	0	0	0

Descontrução do decoder em elementos lógicos mais simples



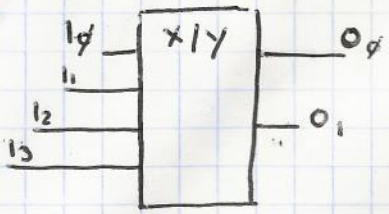
$O_0 = EN.\overline{I_1}.\overline{I_0}$ $O_2 = EN.I_1.\overline{I_0}$

$O_1 = EN.\overline{I_1}.I_0$ $O_3 = EN.I_1.I_0$

Exemplo 2:4 Dual

Encoder

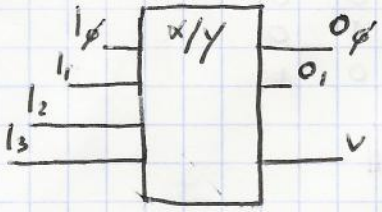
Exemplo simples



I_3	I_2	I_1	I_0	O_1	O_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

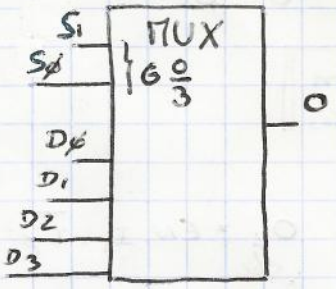
Os outros casos não são distinguidos

Exemplo - Priority Encoder

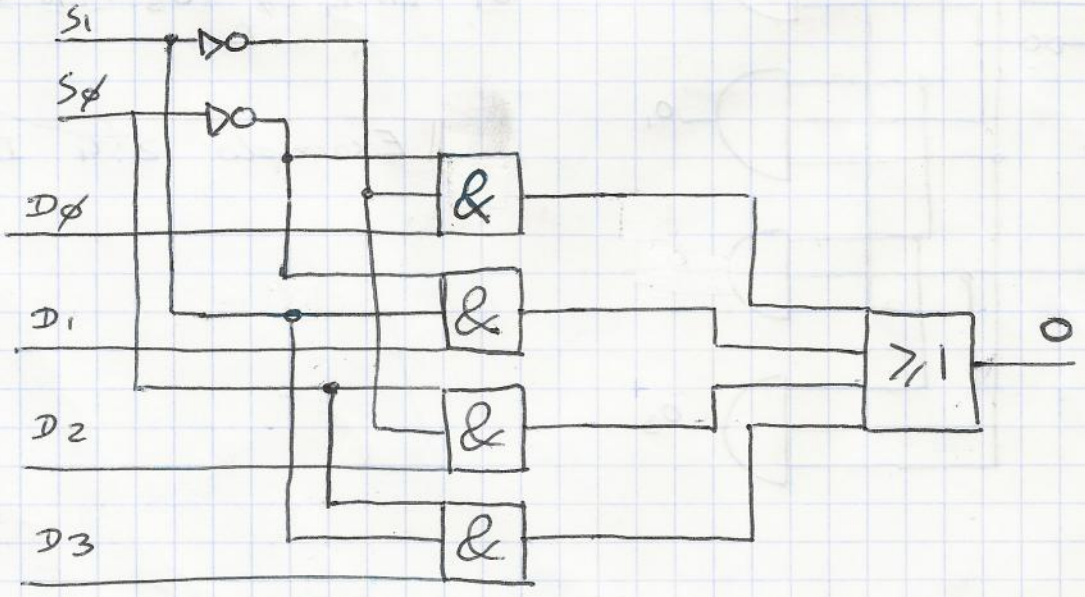


I_3	I_2	I_1	I_0	O_1	O_0	V
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	Φ	0	1	1
0	1	Φ	Φ	1	0	1
1	Φ	Φ	Φ	1	1	1

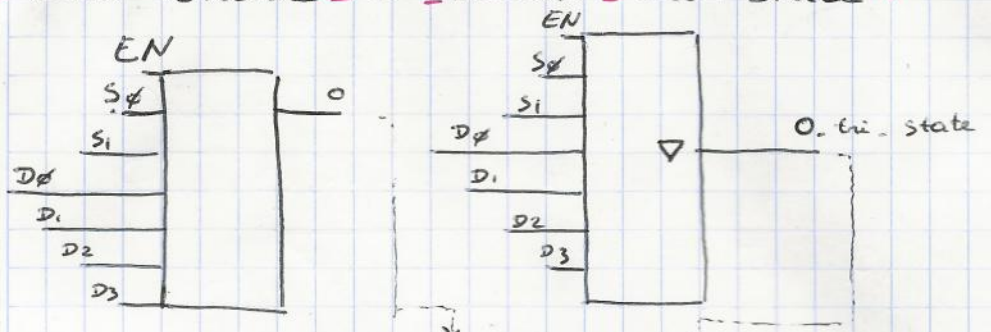
Multiplexer



S_1	S_0	O
0	0	D_0
0	1	D_1
1	0	D_2
1	1	D_3



Mux com enable e Mux com enable e saída tri-state

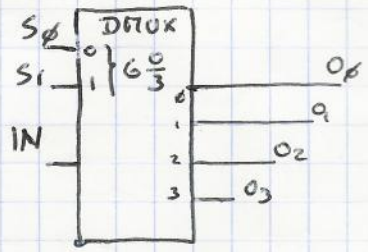


S_1	S_ϕ	EN	O
0	0	1	D_ϕ
0	1	1	D_1
1	0	1	D_2
1	1	1	D_3
ϕ	ϕ	0	0

O - tri - state
D_ϕ
D_1
D_2
D_3
Z

→ Saída em alta impedância

Demultiplexer



S_1	S_ϕ	O_ϕ	O_1	O_2	O_3
0	0	IN	0	0	0
0	1	0	IN	0	0
1	0	0	0	IN	0
1	1	0	0	0	IN

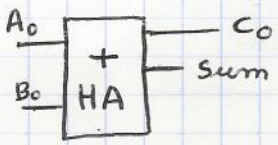
Aula 09
Somadores Subtractores e Comparadores

Somadores

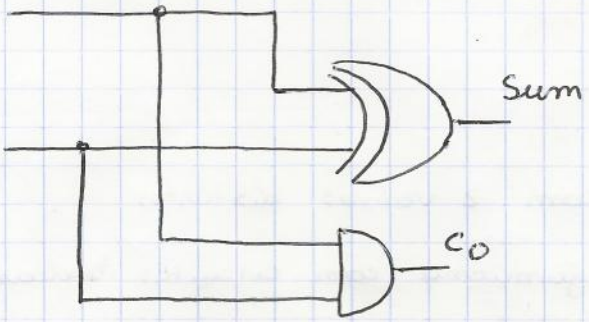
$$\begin{array}{r} 7 \\ +\ 2 \\ \hline 9 \end{array} \rightarrow$$

$$\begin{array}{r} {\scriptstyle 1}\ {\scriptstyle 1}\ {\scriptstyle 1}\ {\scriptstyle 0} \\ 0\ 1\ 1\ 1 \\ +\ 0\ 0\ 1\ 0 \\ \hline 1\ 0\ 0\ 1 \end{array} \quad \text{carry}$$

Half Adder



A_0	B_0	C_0	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0



Aoldor

C_1	A_1	B_1	C_0	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	0	1
1	1	0	1	0
1	1	1	1	1

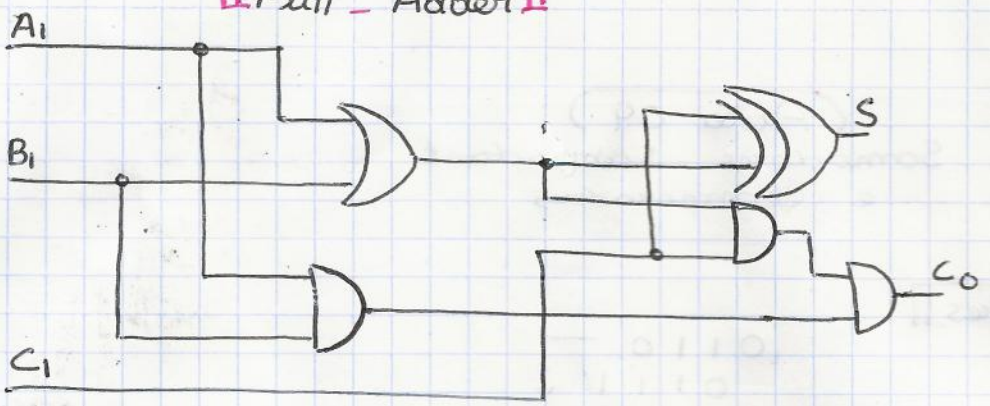
C_1 \ A_1B_1	00	01	11	10
0	0	0	1	0
1	0	1	1	1

$C_0 = AB + C_1(A \oplus B)$

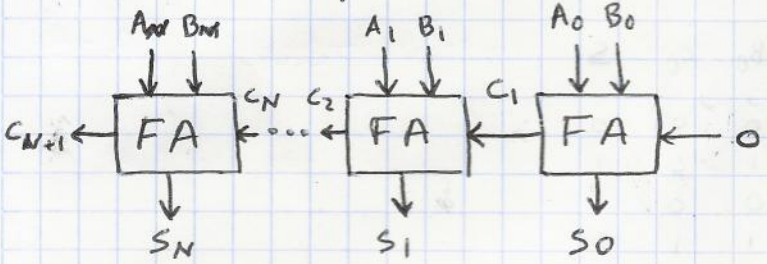
C_1 \ A_1B_1	00	01	11	10
0	0	0	1	0
1	1	1	0	1

$S = A \oplus B \oplus C_1$

Full_Adder



Ripple_Carry_Adder



Representação de números negativos

Bit extra como sinal $\begin{cases} 0 \text{ com } 2 \text{ valores distintos} \\ \overline{N} \text{ funciona com circuitos booleanos} \end{cases}$

↓

✗

Complemento para 1 → ✗

(=) ↳ • $(2^m - 1) - N$
• Inverter todos os bits

+0 0000		-0 1111	→ Zero continua com 2 valores distintos
+1 0001	→	-1 1110	
+2 0010		-2 1101	
+3 0011		-3 1100	
+4 0100		-4 1011	

Complemento para 2

<=>
- $2^n - N$
- Complementar para 1 e somar 1

+0	0000		-0	1111 +1 → 0000
+1	0001		-1	1111
+2	0010	→	-2	1110
+3	0011		-3	1101
+4	0100		-4	1100

- ZERO com uma única representação
- Funcional com circuitos aritméticos e booleanos

Soma com números em complemento para dois

$$\begin{array}{r} +3 \\ +\ -2 \\ \hline 1 \end{array} \longrightarrow \begin{array}{r} [1\ 1\ 1] \\ 0011 \\ +\ 1110 \\ \hline 0001 \end{array}$$

→ O último bit do carry out não é significativo

Subtracção com números em complemento para dois

$$\begin{array}{r} +3 \\ -\ +4 \\ \hline -1 \end{array} \longrightarrow \begin{array}{r} 0011 \\ -\ 0100 \\ \hline 1111 \\ [1\ 1\ 1] \end{array}$$

↓

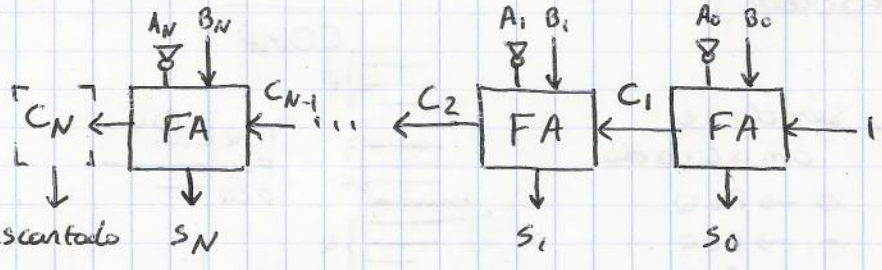
O último borrow não é significativo

$$\begin{array}{r} +3 \\ +\ -4 \\ \hline -1 \end{array} \longrightarrow \begin{array}{r} 0011 \\ +\ 1100 \\ \hline 1111 \end{array}$$

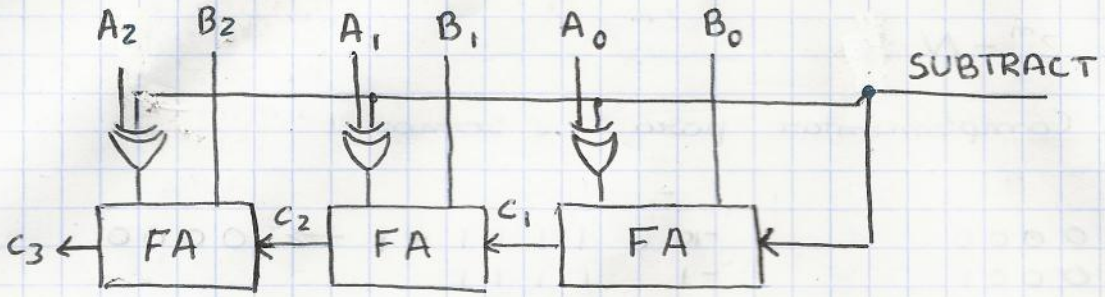
Mais simples (soma do complemento)

Estrutura Subtractora

-A+B = B-A:



Circuito Somador e Subtractor



B+A : SUBTRACT = 0

B-A : SUBTRACT = 1

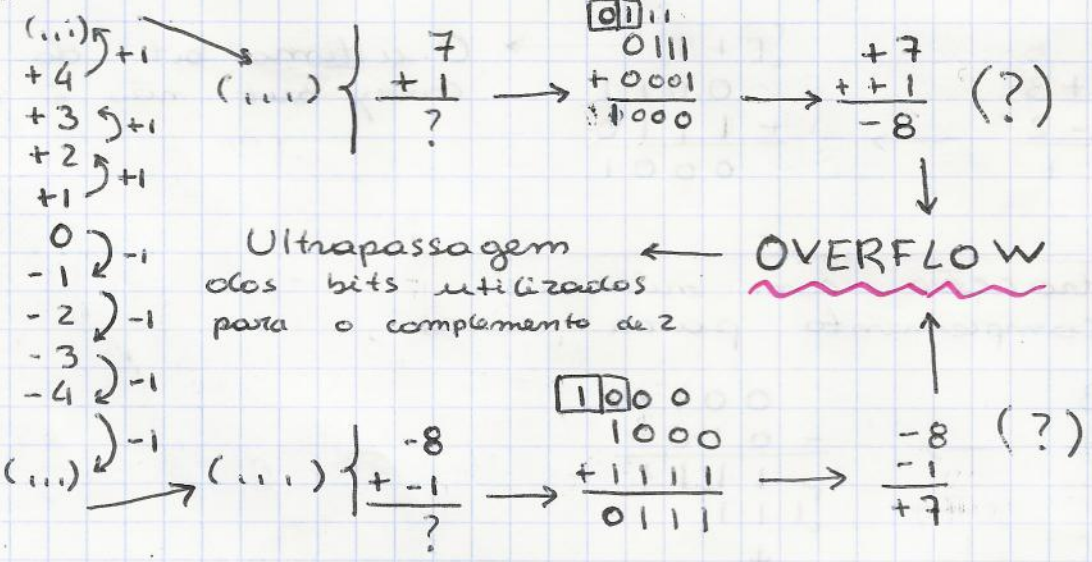
$A \oplus 0 = A$

$A \oplus 1 = \overline{A}$

Carry Inicial (0) = SUBTRACT

0 para Soma
1 para subtracção

Overflow



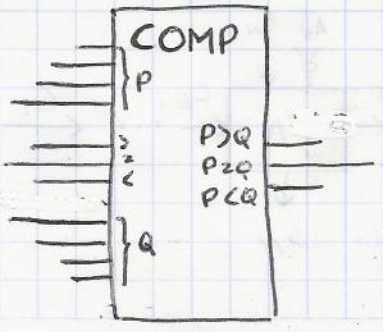
$Overflow = CarryOut_{N-1} \oplus CarryOut_{N-2}$

Logo o $CarryOut_{N-1}$ indica ultrapassagem de bits em operações aritméticas SEM sinal e o Overflow em operações aritméticas COM SINAL

Comparador

Faz P-Q, sendo o resultado analisado

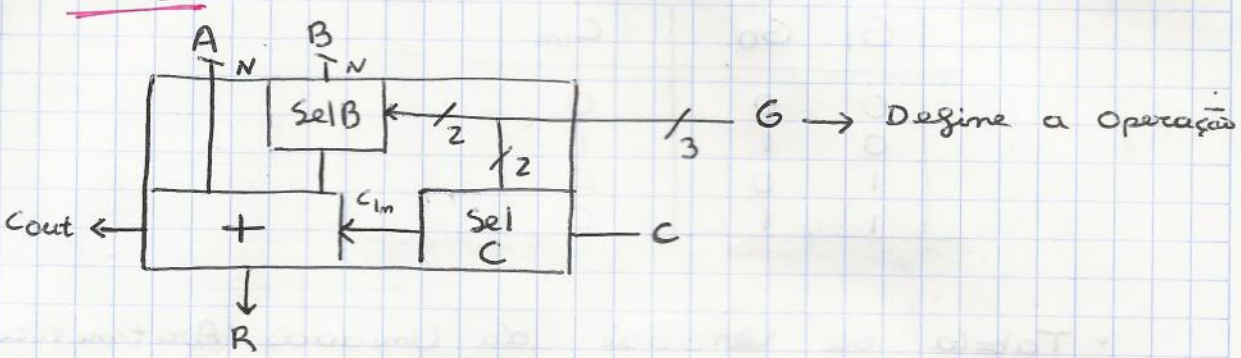
0 → P = Q
− → P < Q
+ → P > Q



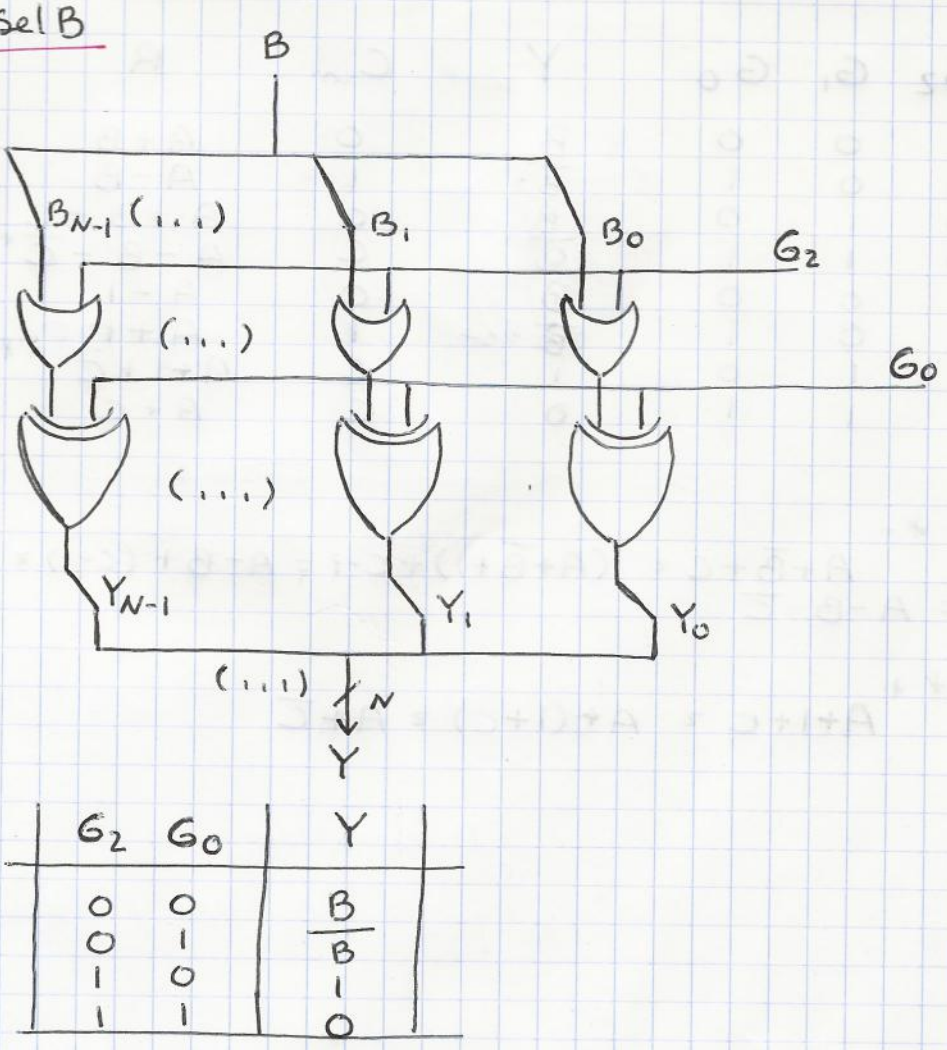
Aula 10
Unidade Lógica e Aritmética

Unidade_Aritmética

- Base

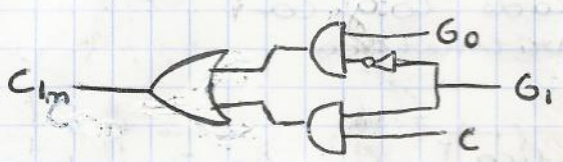


- SelB



G2	G0	Y
0	0	B
0	1	B
1	0	1
1	1	0

° Sel C



G_1	G_0	C_{in}
0	0	0
0	1	1
1	0	C
1	1	C

° Tabela de Verdade da Unidade Aritmética Base

G_2	G_1	G_0	Y	C_{in}	R	Operação
0	0	0	B	0	A + B	Soma
0	0	1	$\overline{B}$	1	A − B	Subtração
0	1	0	B	C	A + B + C	Soma c/ bit trans.
0	1	1	$\overline{B}$	C	A − B − $\overline{C}$ *	Subtração c/ bit $\overline{\text{trans.}}$
1	0	0	1	0	A − 1	Decremento
1	0	1	0	1	A + 1	Incremento
1	1	0	1	C	A − 1 + C **	Decremento se C = 0
1	1	1	0	C	A + C	Incremento se C = 1

* -

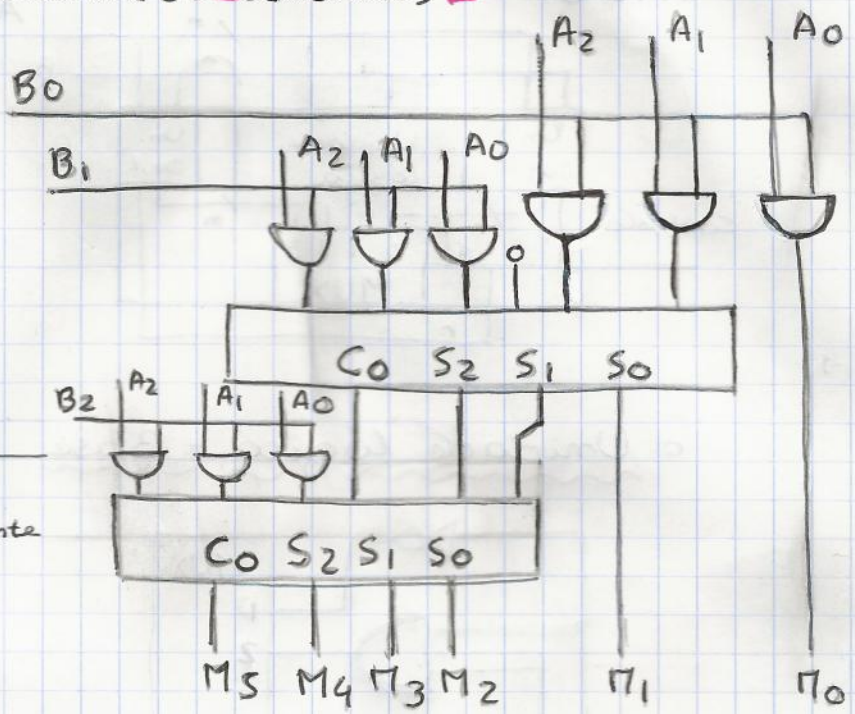
$A+\overline{B}+C = (A+\overline{B}+1)+C-1 = A-B+(C-1) = A-B-(1-C) = A-B-\overline{C}$

* *

$A-1+C = A-(1+C) = A-\overline{C}$

Multiplicação, Divisão e circuitos multiplicativos binários

```
   0 1 1      A
 x 1 0 1      B
-------
   0 1 1
 0 0 0
+0 1 1
-------
0 0 1 1 1 1
```



Estrutura simples mas não é particularmente eficiente

Multiplicação e divisão por potência de base 2

101 x 010 = 1010

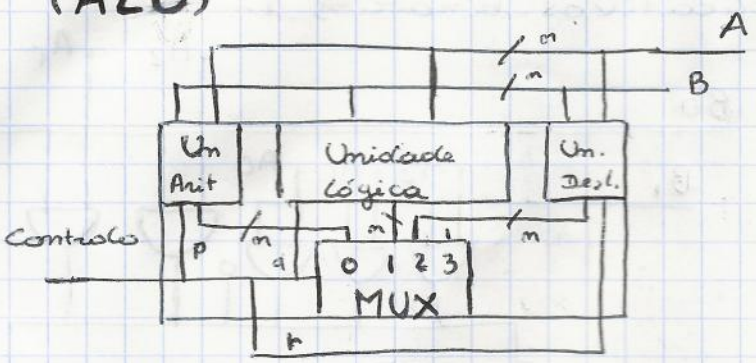
101 x 100 = 10100

10100 / 100 = 00101

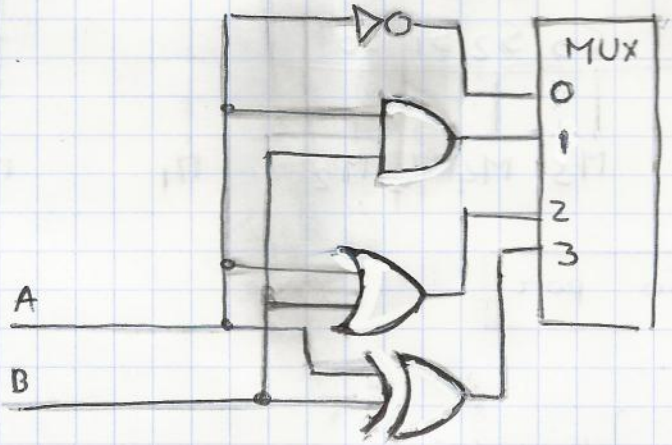
"Deslocamento" de k bits na divisão / multiplicação por 2^k

- Num caso geral as divisões são operações raras

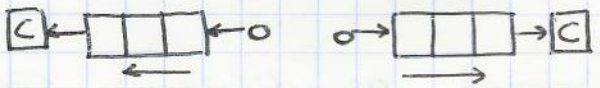
Unidade Lógica e Aritmética (ALU)



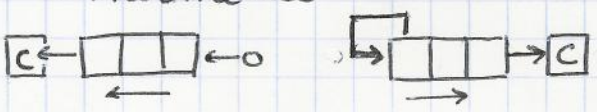
- Unidade Lógica - Base



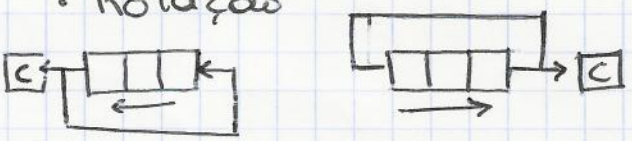
- Unidade de deslocamento
  - Operações de deslocamento
    - Simples



    - Aritmético



    - Rotação



Rotação com transporte

Unidade de Deslocamento

J_2	J_1	J_0		Operação		
0	0	0	D_i	Lógico	à	dir
0	0	1	D_i	Lógico	à	esq
0	1	0	D_i	Aritmético	à	dir
0	1	1	D_i	Aritmético	à	esq
1	0	0	R_i	-	à	dir
1	0	1	R_i	-	à	esq
1	1	0	R_i	C/ transporte	à	dir
1	1	1	R_i	C/ transporte	à	esq