

## Algebra de Boole

### AND

$$L = A \cdot B$$

A	B	L
1	1	1
0	0	0
0	1	0
1	0	0

- $A \cdot 1 = A$
- $A \cdot 0 = 0$
- $A \cdot A = A$
- $A \cdot \bar{A} = 0$
- $A \cdot B = B \cdot A$

### Teorema da Absorção

$$A + A \cdot B = A$$

- $A + \bar{A} \cdot B = A + B$
- $\bar{A} + A \cdot B = \bar{A} + B$

### OR

$$L = A + B$$

A	B	L
1	1	1
1	0	1
0	1	1
0	0	0

- $A + 1 = 1$
- $A + 0 = A$
- $A + A = A$
- $A + \bar{A} = 1$
- $A + B = B + A$

### Teorema de DeMorgan

$$\overline{A \cdot B} = \bar{A} + \bar{B}$$

$$\overline{A + B} = \bar{A} \cdot \bar{B}$$

### Propriedades AND / OR

$$A + B + C = (A + B) + C = A + (B + C)$$

$$ABC = (AB)C = A(BC)$$

$$A(B+C) = AB + AC$$

$$A + BC = (A + B) \cdot (A + C)$$

### NOT

$$L = \bar{A}$$

A	L
1	0
0	1

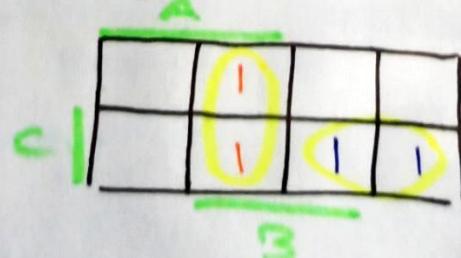
$$\bar{\bar{A}} = A$$

$$L = \bar{A} \Leftrightarrow \bar{L} = A$$

Ex:

$$F = \underline{A \cdot B} + \underline{\bar{A} \cdot C} + A \cdot \bar{B} \cdot C + \bar{A} \cdot \bar{B} \cdot C$$

$$2^3 = 8 \text{ Quadriculos}$$



$$F = AB + AC$$

### Teorema do Consenso

$$\cancel{\bar{A} \cdot \cancel{B}} + \cancel{A \cdot \cancel{C}} + \cancel{B \cdot \cancel{C}} = \bar{A} \cdot B + A \cdot C$$

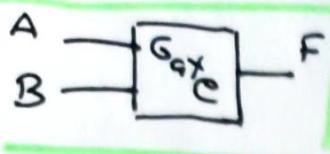
### Mapas de Karnaugh

$$\text{Número de quadriculos} = 2^n$$

$n \rightarrow n = \text{de variáveis}$

## Portas Lógicas

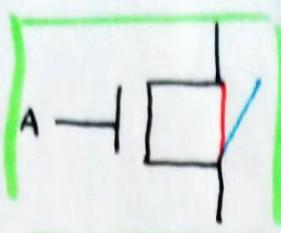
### Componição



A } input

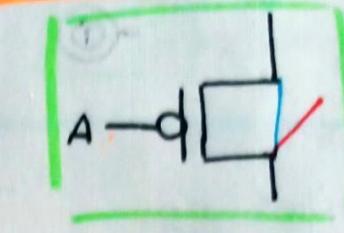
F → output

### Transistor



- Abre com OV

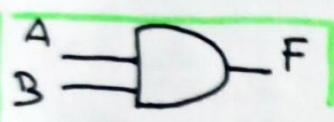
- Fecha com +V



- Abre com +V

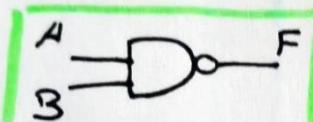
- Fecha com OV

### AND



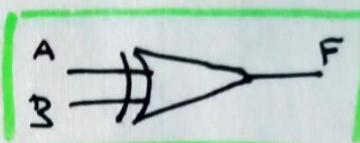
$$F = A \cdot B$$

### NAND



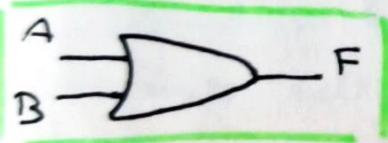
$$F = \overline{A \cdot B}$$

### XOR



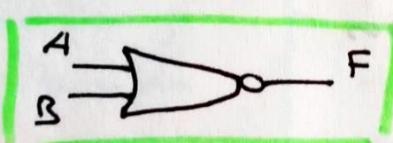
$$F = A \oplus B$$

### OR



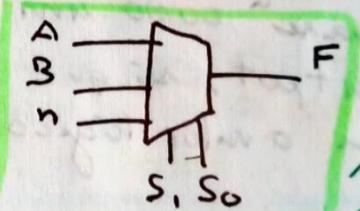
$$F = A + B$$

### NOR



$$F = \overline{A + B}$$

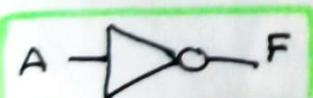
### Multiplexor



\* Mux auto  
a - input  
b - outputs  
n - contar com  
seletores !!!

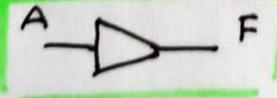
escolhe 1 de n inputs  
através do descodificador

### NOT



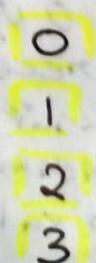
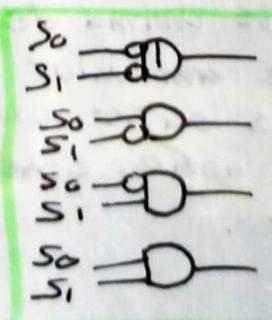
$$F = \overline{A}$$

### IDENT



$$F = A$$

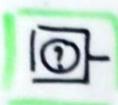
### Descodificador



funciona com números binários

## Componentes

### Pin Input



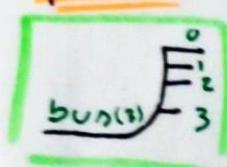
### Pin Output



### "Bus"

- Quando ligamos várias fios ~~umas~~ uns ~~ao~~ ~~outro~~, ficam pretos e precisamos de um splitter para os separar

### Splitter

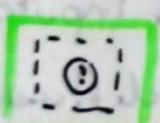


Separar os cabos de um "bus"

### Probe

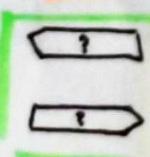
→ "Quase" como um pin de output, só que apenas mostra o valor lógico

### Constant



Como um pin de input só que constante em vez de variável

### Tunnel



→ O Nyarl chama de "portal", recebe um valor e reenvia o outro com o mesmo nome, esse recebe o valor.

## Números e Operações

### Representações

$$N = \sum A_i B^i$$

$B$  → base

$A_i$  → Algarismos

$i$  → índice posicional do algarismo

Converter  $10 \rightarrow ?_2$

$$158_{10} = ?_2$$

$$\begin{array}{r} 158 \\ \underline{-79} \\ 79 \\ \underline{-39} \\ 39 \\ \underline{-19} \\ 19 \end{array}$$

usamos os algarismos

dos restos ao contrário

Ex: 110 (inverte o de cima)

### Operações adição binária

$$\text{Ex: } 100101_2 + 110111_2$$

$$\begin{array}{r} 0011 \quad B \\ A = 100101 \\ B = 110111 \\ S = 1011100 \end{array}$$

Diagrama de somador binário:

Entradas:  $A_n, B_n$  (verde) → Somador (verde) → Soma ( $S_n$ ) e Carry ( $C_{n+1}$ ). O carry é somado ao próximo dígito.

### Semi-Somador

$A_n$	$B_n$	$S_n$	$C_{n+1}$
0 = 0 + 0	00	0	0
1 = 0 + 1	01	0	0
1 = 1 + 0	01	0	0
2 = 1 + 1	10	1	1

### Tipos de Bases

10 - Decimal

2 - Binária

16 - Hexadecimal

1009 e Aa F

...

### Binários Negativos

0 → Positivo

1 → Negativo

$$23_{10} = 10111_2$$

$$+23 = 010111$$

$$-23 = 101000 + 1$$

↳ exemplo

### Operações Subtração Binária

Ex:

$$\begin{array}{r} 11000000 \\ \underline{-1101000} \\ 011010 \end{array}$$

BN → Borrow

BN → Borrow

### Código dos Complementos

Simétrico de  $A$  é o que somamos com  $A$  para dar 0

$$A = 011010010_2$$

$$\bar{A} = 100101101_2$$

↳ Complemento de  $A$  (inverter dígitos)

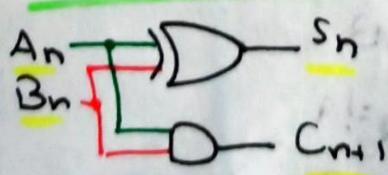
$$-A = \bar{A} + 1 \quad A + (-A) = 0$$

\* Próxima página

### Carry (Arrasto)

$$S_n = A_n \oplus B_n$$

$$C_{n+1} = A_n \cdot B_n$$



### Somador Completo

$C_n$	$A_n$	$B_n$	$S_n$	$C_{n+1}$
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	1	0	0	1
1	0	1	0	1
1	1	1	0	1

## \* Código de complementos

$$A = 011010010$$

$$\bar{A} = 100101101$$

$$100101110 = -A$$

$$011010010 + A$$

$$+ 100101110 - A$$

$$\hline 000000000 \quad 0$$

$-A$  é o análogo de  $A$

Calcular rapidamente negativos

$$\text{ex: } (-3) = -8 + 5$$

$$\begin{array}{|c|c|} \hline 1 & 1 & 0 & 1 \\ \hline \end{array}$$

$$-6 = -8 + 2$$

$$\begin{array}{|c|c|} \hline 1 & 0 & 1 & 0 \\ \hline \end{array}$$

Solver Adisoén (exercício)

$$\begin{array}{r} B \ 1 \ 0 \\ + !A \ B \ A \\ \hline C_1 \ R_2 \ R_1 \ D_0 \end{array}$$

Descobrir  $C_1$  e  $R_1(a=0)$ :

$$\begin{array}{|c|c|c|} \hline A & 0 & R_1 & C_1 \\ \hline 0 & 0 & 0 & 0 \\ \hline 1 & 0 & 1 & 0 \\ \hline \end{array} \rightarrow R_1 = A \text{ e } C_1 = 0$$

$$\begin{array}{|c|c|c|} \hline B & 1 & R_1 & C_2 \\ \hline 0 & 1 & 1 & 0 \\ \hline 1 & 0 & 0 & 1 \\ \hline \end{array} \rightarrow R_1 = !B \text{ e } C_2 = B$$

$$\begin{array}{|c|c|c|c|} \hline B & !A & C_2 & R_2 & C_4 \\ \hline 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 1 & 0 & 1 & 1 \\ \hline \end{array} \rightarrow R_2 = !A \text{ e } C_4 = B$$

Resolver Tabela de Somador/Sub.

$$R = A - B - C_y \text{ (3 bits)} \quad e \sim 5 \text{ bits}$$

base	A	B	$C_y B_{w1}$	R	$C_{B_0}$	Ov/c
2	00100	10100	0	10000		
	4	20	0	16		
10 IN						
10/2	+4	-12	0	-16		

Calcular na coluna do B e R:

$$(20)_{10} = (10100)_2 = -12 \text{ em complemento}$$

$$(-16)_{10} = (10000)_2 = -16 \text{ em complemento}$$

Usar uma linha:

$$-16 = A - 20 - 0 \Leftrightarrow A = 36$$

frente do D.

$$36 - 32 = \boxed{4} \rightarrow \text{dentro do Domínio}$$

$$\text{e } C_{B_0} = 1$$

Depois calcular a coluna A e verificar o Ov e G.

$$IN \rightarrow C_y B_{w0} \rightarrow [0, 31]$$

$$Z \rightarrow Ov \rightarrow [-16, 15] \quad \leftarrow D.$$

Conversão para base 10

$$(abc)_k = a \times k^2 + b \times k^1 + c \times k^0$$

Conversão para base 2

Dividir e ver os restos ao contrário

## C. Lógica Sequencial

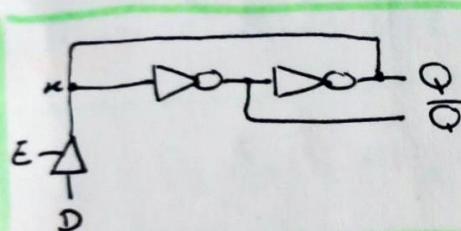
### Introdução

O circuito reage à entrada em funscos de acontecimentos passado.

Logo para guardar esses acontecimentos é necessário existir Memória

### Disponibilidade Memória

- Chamam células unitárias de memória → Flip-flop
- Permitem memorizar um bit de informação:



Quando...

- $E \rightarrow 1 \Rightarrow Q = D$
- $E \rightarrow 0 \Rightarrow$  O valor de D é trincado na saída Q enquanto E permanecer a zero

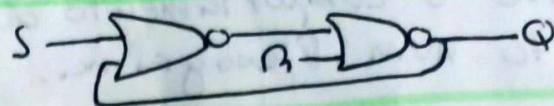
### ↳ Flip-flop D-latch

### Flip-flop S-R

Tem 2 entradas:

- S(set) - para impor 1
- R(reset) - para impor 0

Memoriza quando estão desativados ambos



Quando...

$S \rightarrow 0$  e ativarmon S, Q vai ficar 1 até ativarmon R.

$S \rightarrow 0$  e ativarmon R, Q vai ficar 0 até ativarmon S.

### Flip-flop Latch

Quando E está desativado, Q mantem o valor lógico, independente do D.

Quando E está ativa, Q fica com o valor lógico de D.

### Flip-flop Edge-Triggered

A saída não reage à entrada quando o clock transita de 0 para 1, e nas durante o tempo que permanece a 1

$0 \rightsquigarrow 1$  Transição ascendente  
↓ reagindo a nível

\* ver exemplo no PDF

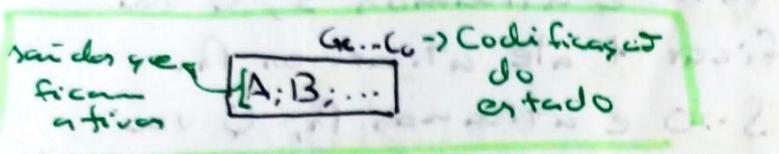
### Registers

Pode-se registar uma palavra constituída por n bits.

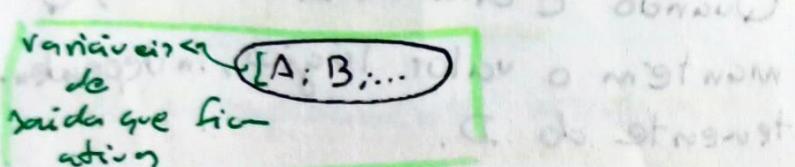
Usam-se os flip-flops que forem necessários.

## ASM - Algorithm Stat Machine

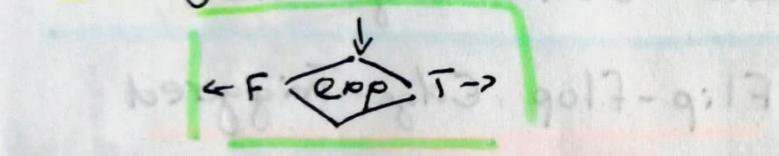
- Descreve o comportamento do circuito num fluxograma.
- As tângulas → estado corrente do circuito



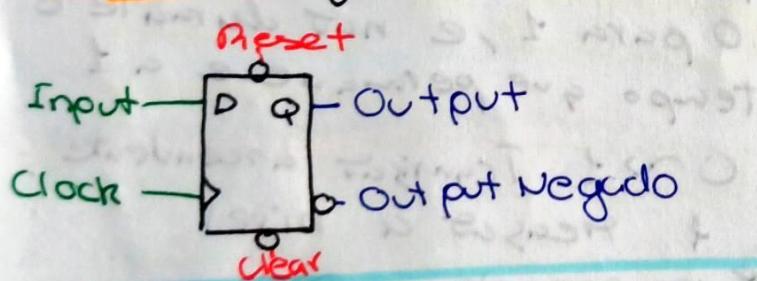
- Oval → ativação de saídas em função do estado presente e das variáveis de entrada



- Laranja - decisão binária



## Flip-Flop Logístico



Logístico → Edge-Triggered

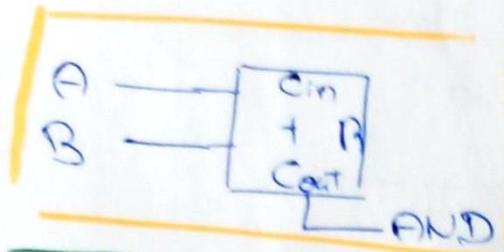
Tipo D → Muda com valor D

Tipo E → Muda quando T=1

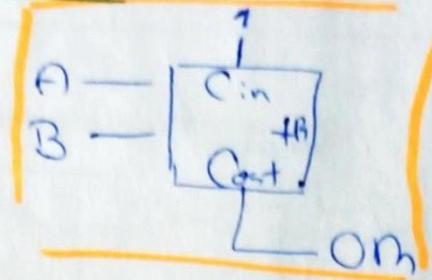
# LSD - Geral

## Portas Lógicas com Adders

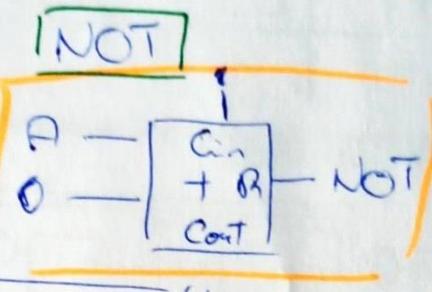
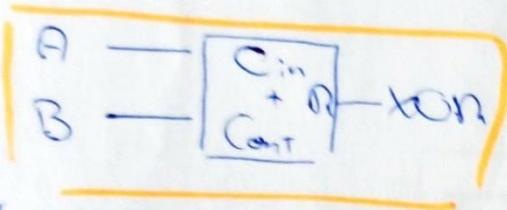
### AND



### OR



### XOR



### Flags

$$CyB_{wo} = *S_n A \oplus C_4 \rightarrow [A < B \text{ em } LN]$$

} N - naturais

$$Ov = \bar{B}_3 \cdot \bar{A}_3 \cdot \bar{B}_3 + \bar{B}_3 \cdot A_3 \cdot B_3 \rightarrow \begin{array}{l} \text{O resultado obtido} \\ \text{é diferente do esperado} \end{array}$$

}  $\mathbb{Z}$  - inteiros com sinal

$$E = \bar{B}_0 \bar{B}_1 \bar{B}_2 \bar{B}_3 \rightarrow [A = B]$$

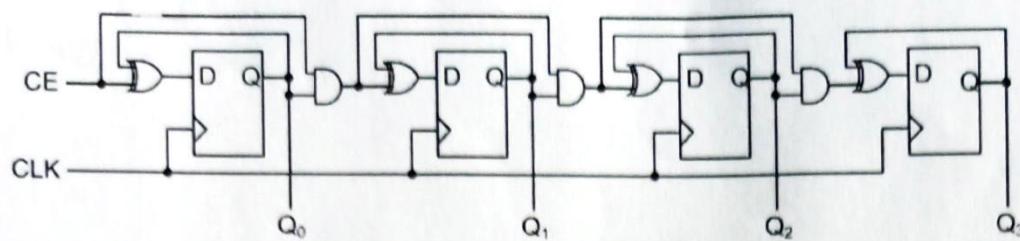
} Apenas significado na subtração

$$GE = Ov \oplus \bar{B}_3 \rightarrow [A \geq B]$$

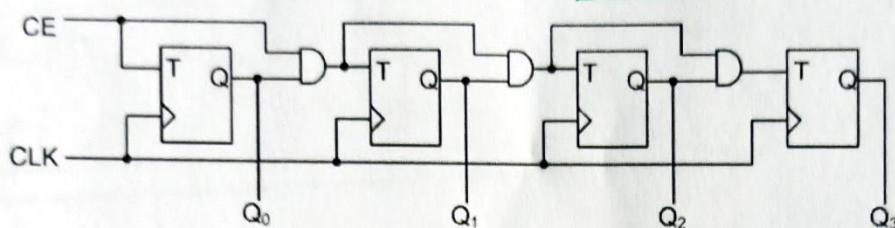
### Contadores

- And entre o valor que queremos no anterior e o valor atual do anterior (valor atual negado se for down)

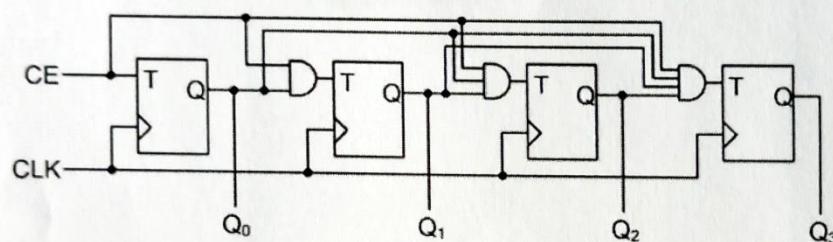
## Contador Up



a) Implementação com flip-flop tipo D



b) Implementação com flip-flop tipo T modo série

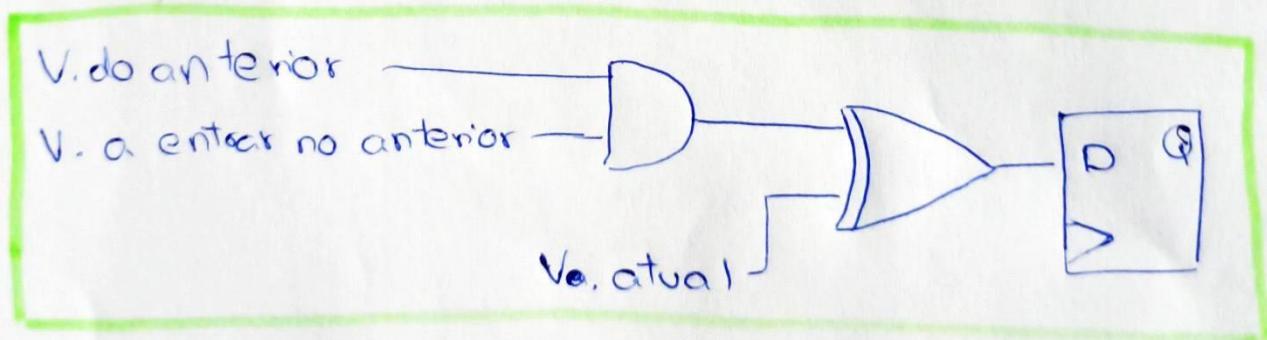


c) Implementação com flip-flop tipo T modo paralelo

Figura 8-7

Tipo D → Xor do valor atual com o que queremos

AND entre o valor do anterior e o valor atual



↳ muda quando o anterior muda de 1 para 0.

## Contador Up/Down $\rightarrow$ Tipo T

UnD  $\begin{cases} 1 \rightarrow \text{Up} \\ 0 \rightarrow \text{Down} \end{cases}$

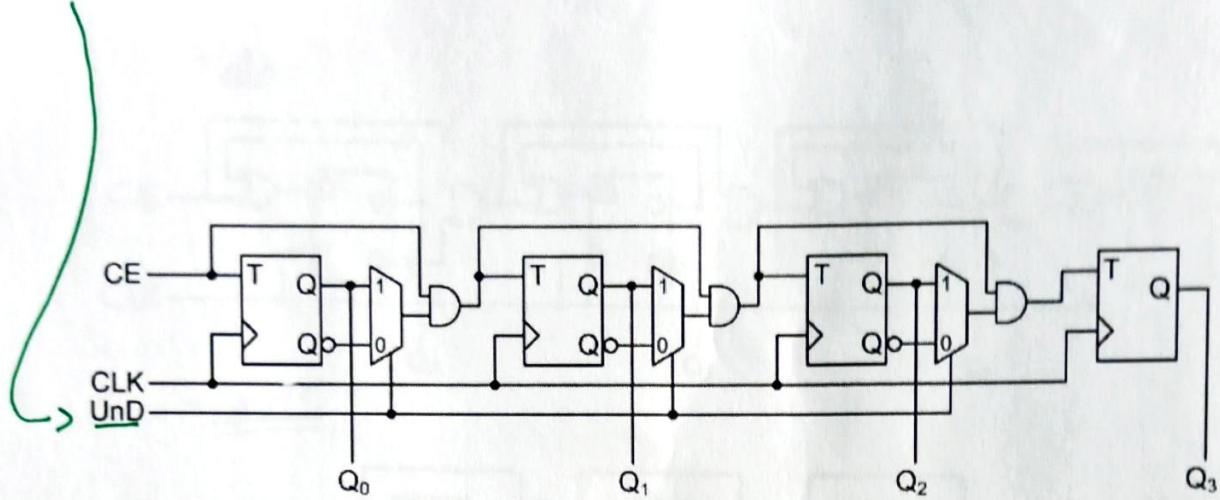


Figura 8-10

# Contador Down

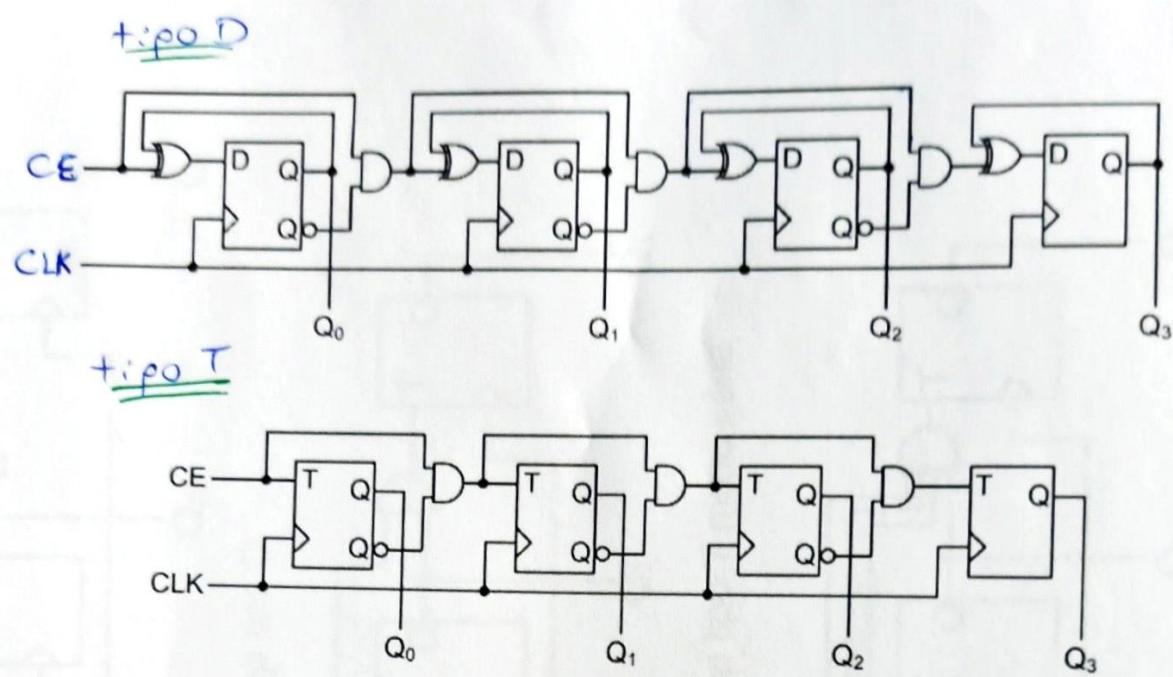
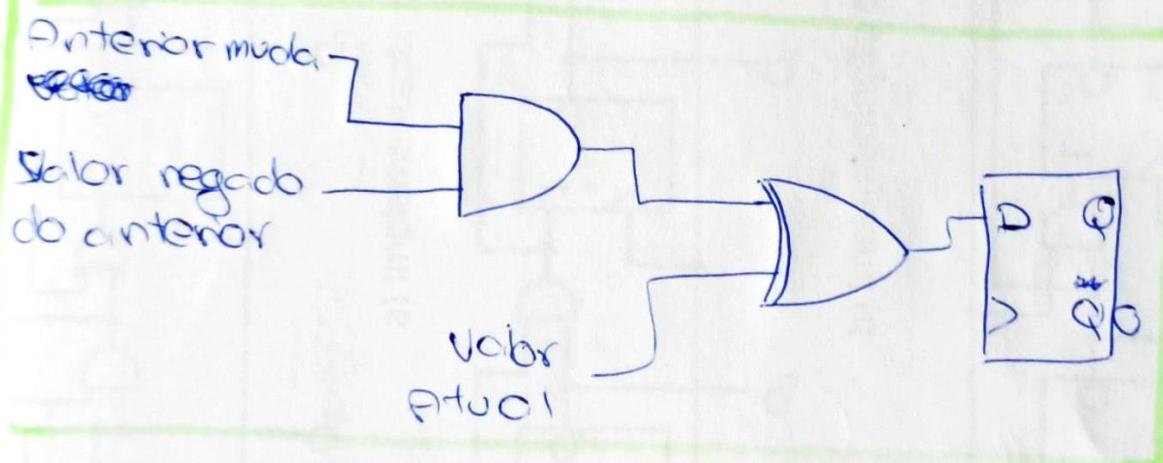
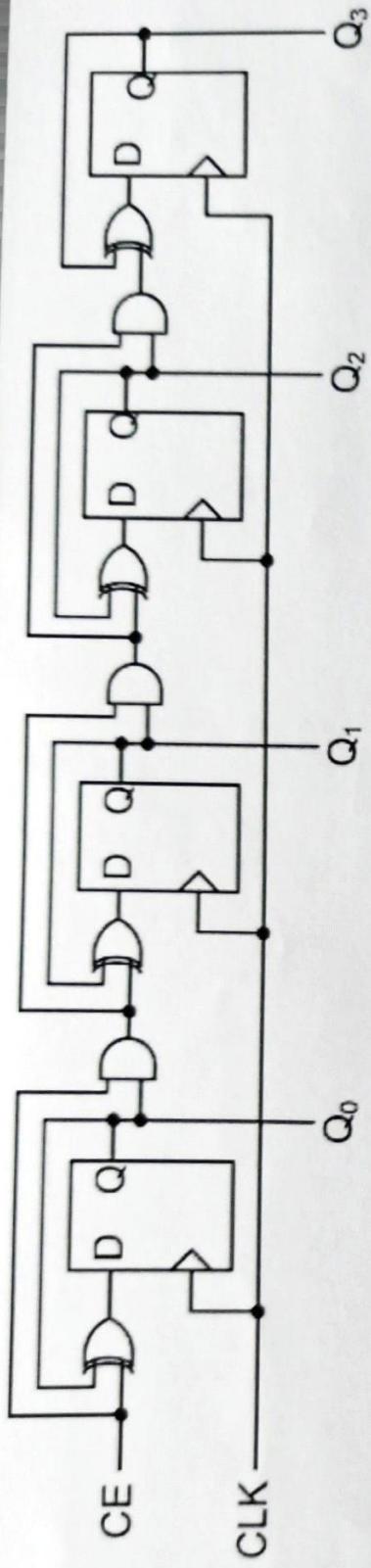
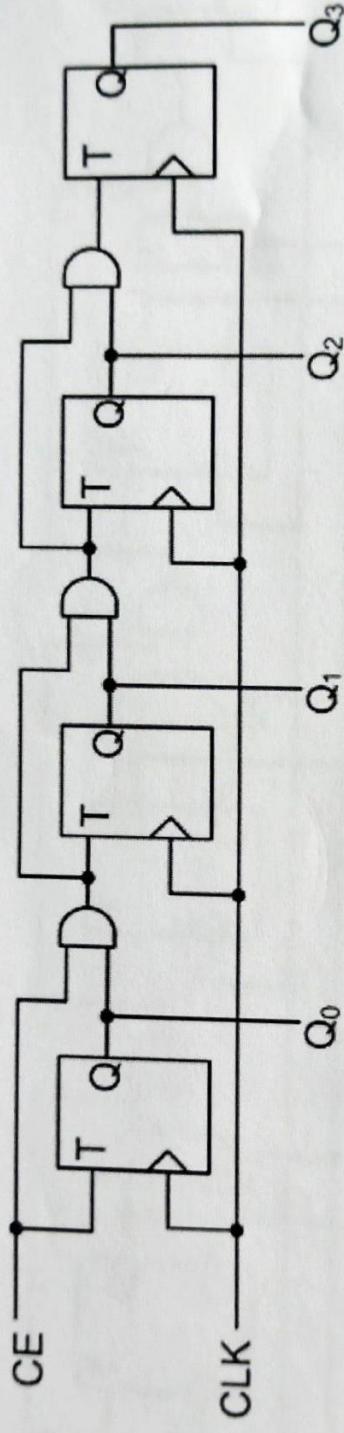


Figura 8-9

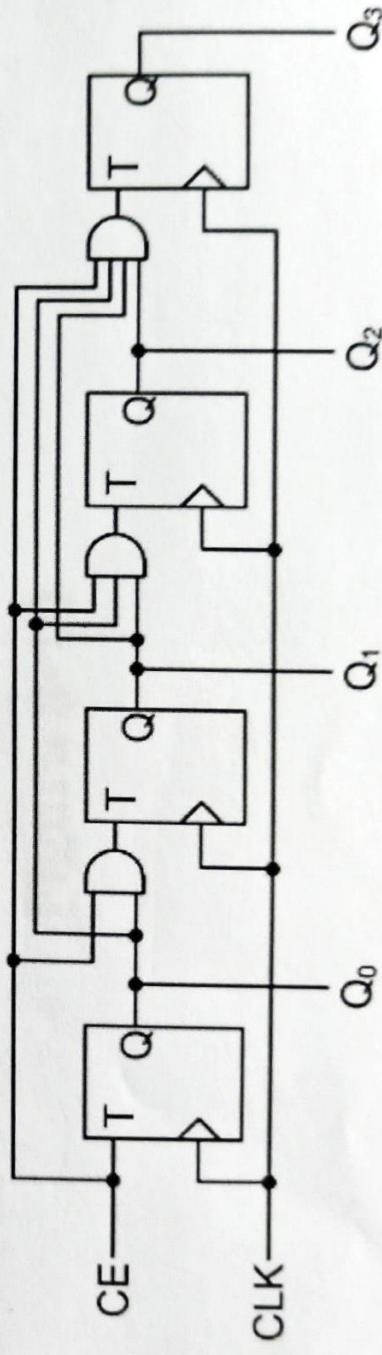




a) Implementação com *flip-flop* tipo D



b) Implementação com *flip-flop* tipo T modo série



c) Implementação com *flip-flop* tipo T modo paralelo

Figura 8-7