

# Arquitetura de Computadores

## Aula 06

### Capítulo 3 - Nível Lógico Digital

# Trabalho

Equipe 1: CD-ROMs

CDs graváveis

CDs regraváveis

Equipe 2: DVD + blu-ray

Equipe 3: Pen Drives

Equipe 4: Entrada e Saída

barramentos

teclados

Equipe 5: Entrada e Saída

terminais (monitor touchscreen)

monitores de tela plana

Equipe 6: Entrada e Saída

mouses (touchpad)

controladores de jogos

Equipe 7: Entrada e saída

impressoras (e scanner)

Equipe 8: Entrada e Saída

equipamentos de telecomunicações

Equipe 9: Entrada e Saída

Câmeras digitais (+ web cam)

**Parte escrita +  
apresentação!**

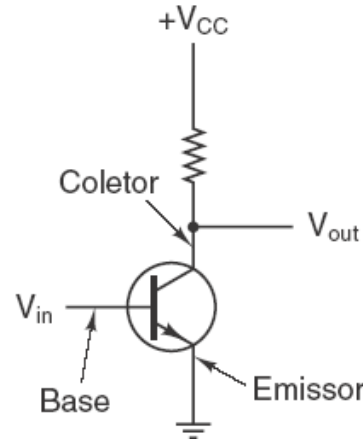
# 3 O nível lógico digital

## Objetivo:

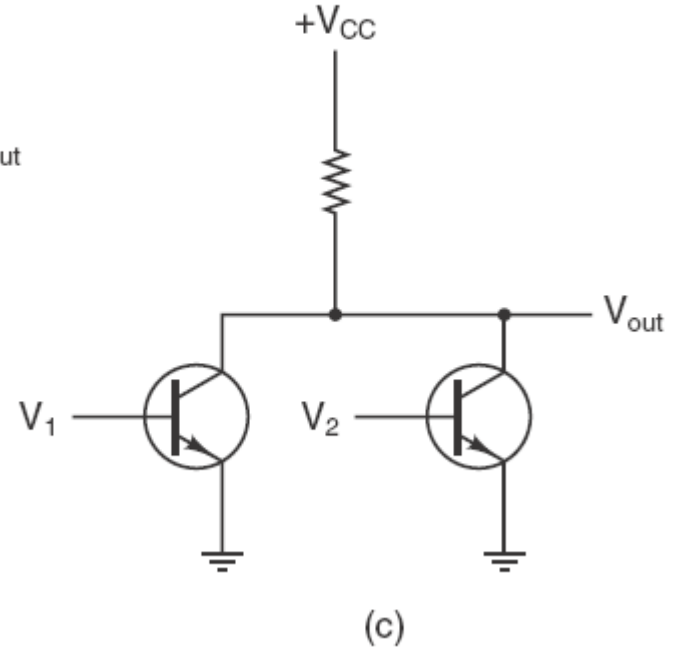
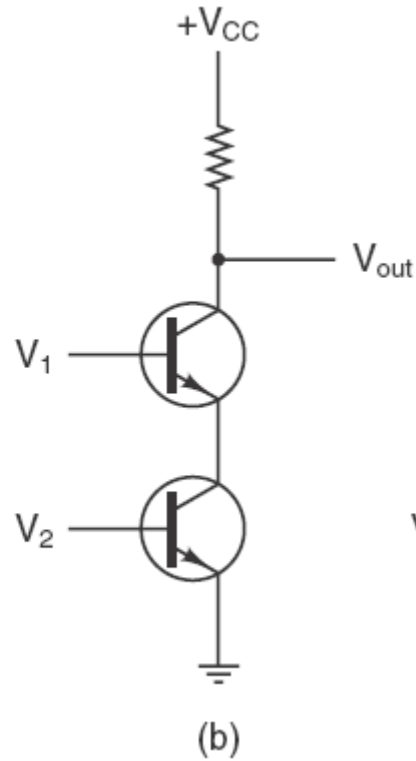
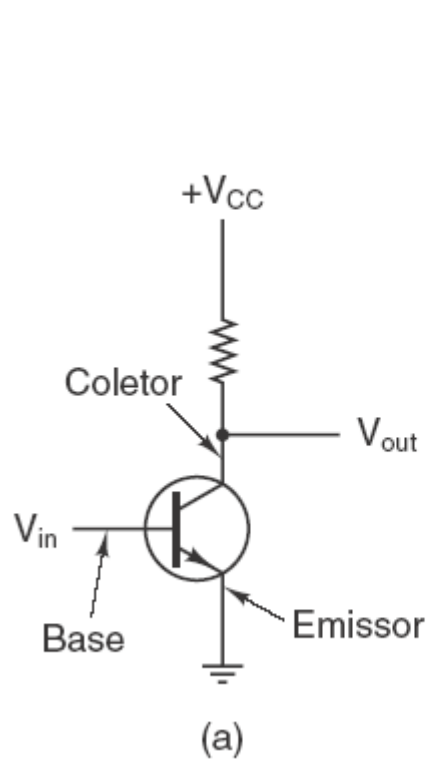
- examinar muitos aspectos da lógica digital, como um fundamento para o estudo de níveis mais altos

## 3.1.1 Portas

- **Portas:** formam a base do hardware sobre qual todos os computadores digitais são construídos.
- Detalhes do funcionamento interno pertencem ao **nível de dispositivo** (que está abaixo do nível 0)



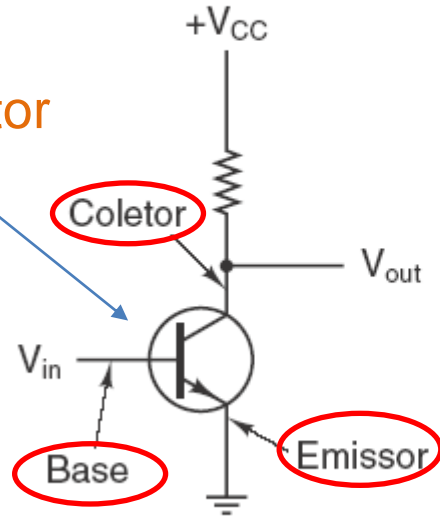
## 3.1.1 Portas



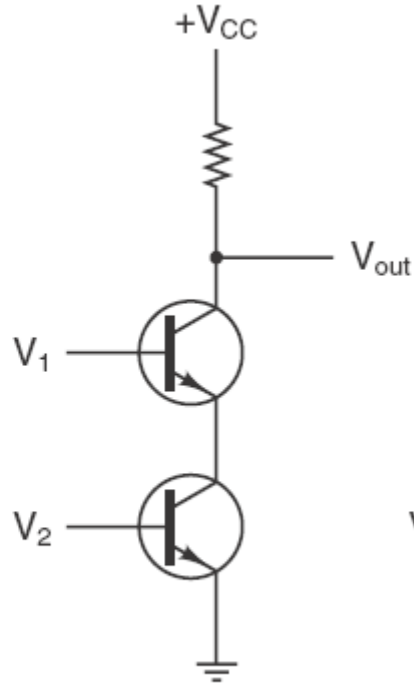
- (a) Inversor com transistor.
- (b) Porta NAND.
- (c) Porta NOR.

## 3.1.1 Portas

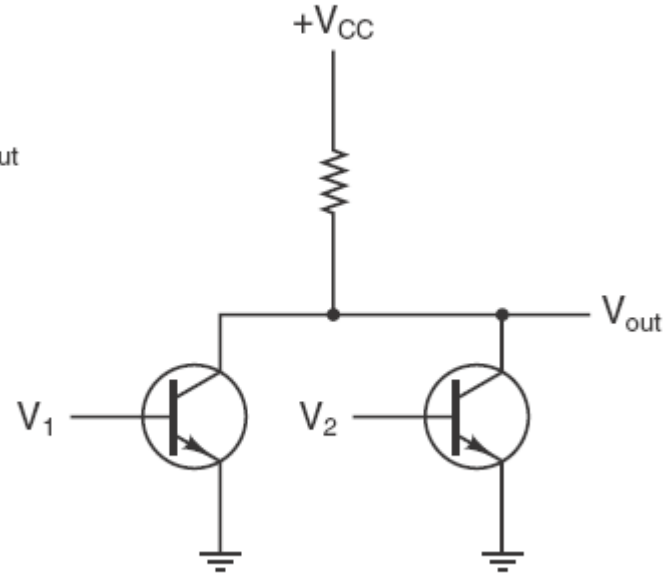
transistor



(a)



(b)



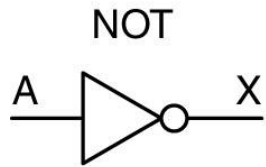
(c)

(a) Inversor com transistor.

(b) Porta NAND.

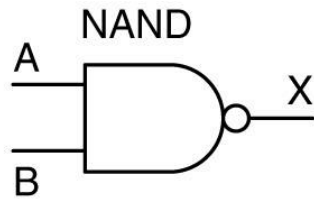
(c) Porta NOR.

## 3.1.1 Portas



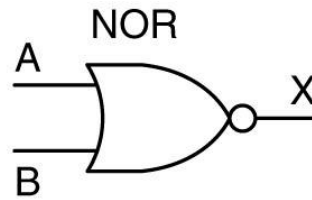
A	X
0	1
1	0

(a)



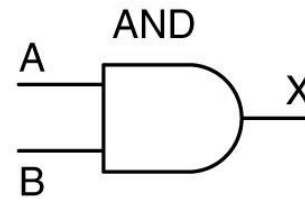
A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

(b)



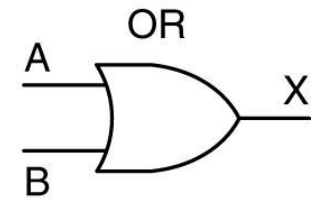
A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

(c)



A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

(d)

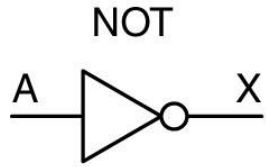


A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

(e)

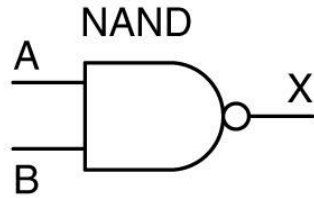
Símbolos e comportamento funcional das cinco portas básicas.

## 3.1.1 Portas



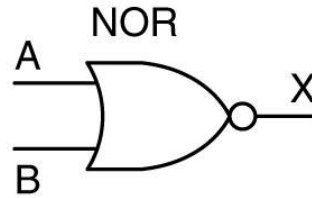
A	X
0	1
1	0

(a)



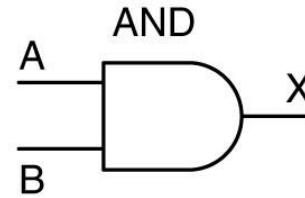
A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

(b)



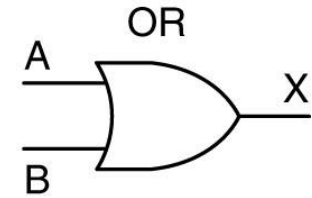
A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

(c)



A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

(d)



A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

(e)

2 transistores

3 transistores

Símbolos e comportamento funcional das cinco portas básicas.



## 3.1.2 Álgebra booleana

- Variáveis e funções podem assumir somente 2 valores: 0 e 1
- Álgebra booleana = Álgebra de comutação
- Ex. função simples:

### Função NOT

$$f(A) = 1 \text{ se } A \text{ for } 0$$

$$f(A) = 0 \text{ se } A \text{ for } 1$$

- Função booleana de  $n$  variáveis tem  $2^n$  combinações possíveis de valores de entrada
- Pode ser completamente descrita por uma tabela  $2^n$  com linhas:  
Tabela Verdade

## 3.1.2 Álgebra booleana

- As portas NAND e NOR são denominadas completas porque qualquer função booleana pode ser calculada usando quaisquer das duas. Nenhuma outra porta tem essa propriedade, essa é uma das razões porque elas costumam ser preferidas como blocos de construção de circuitos.

## 3.1.2 Álgebra booleana

### Função de lógica majoritária

- 0 se a maioria de suas entradas for 0.
- 1 se a maioria de suas entradas for 1.

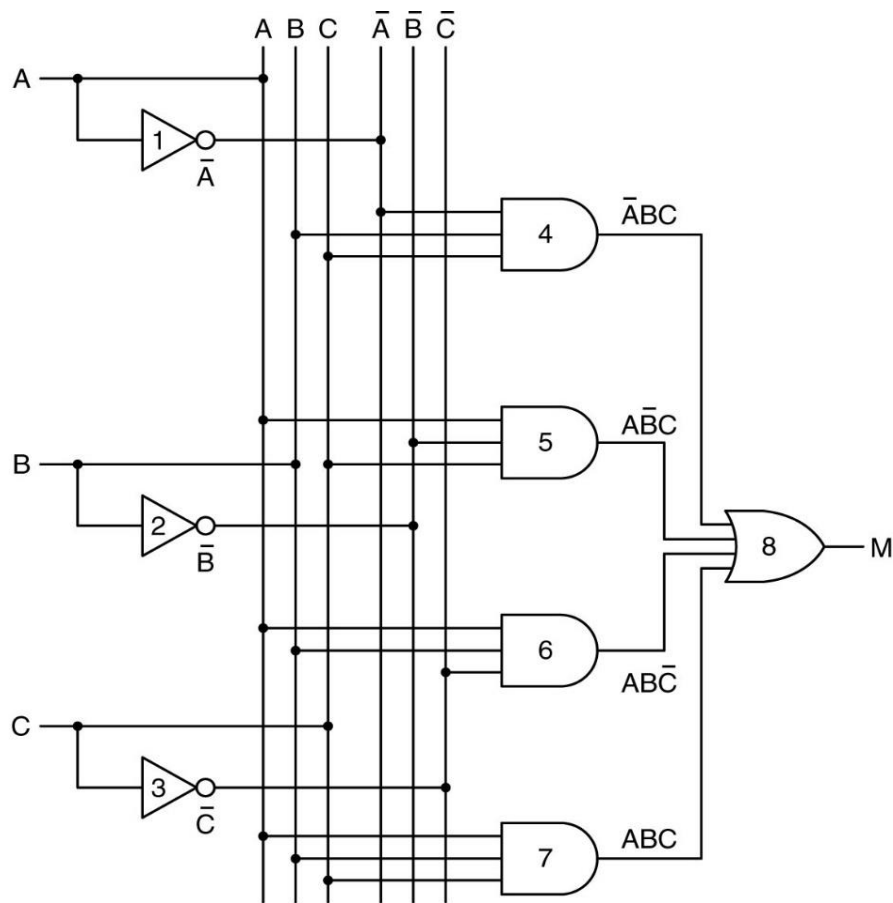
## 3.1.2 Álgebra booleana

### Função de lógica majoritária

A	B	C	M
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

(a)

(a) Tabela-verdade para a função majoritária de três variáveis.



(b)

(b) Um circuito que implementa a função descrita em (a).

## 3.1.2 Álgebra booleana

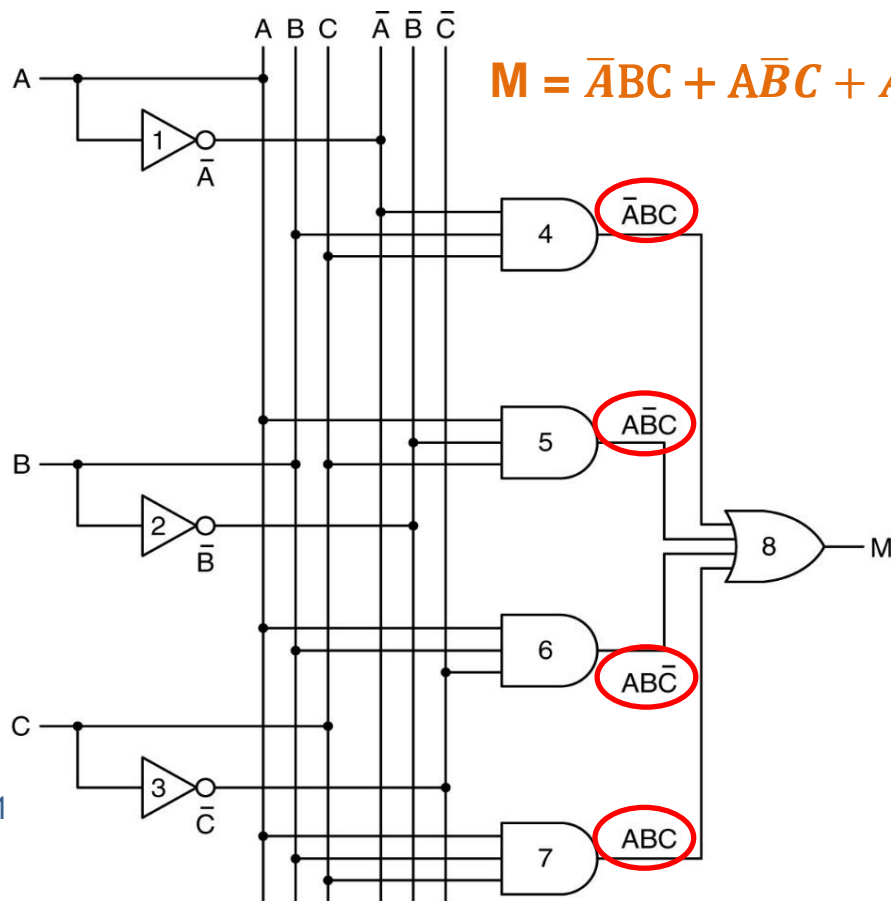
### Função de lógica majoritária

A	B	C	M
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

(a)

(a) Tabela-verdade para a função majoritária de três variáveis.

(b) Um circuito que implementa a função descrita em (a).

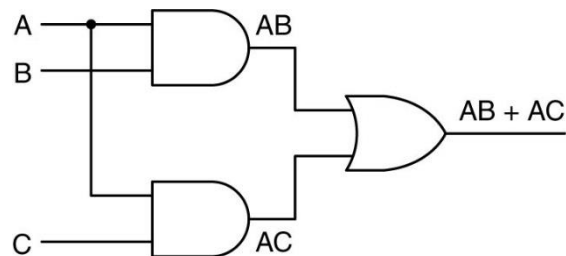


(b)

## 3.1.4 Equivalência de circuito

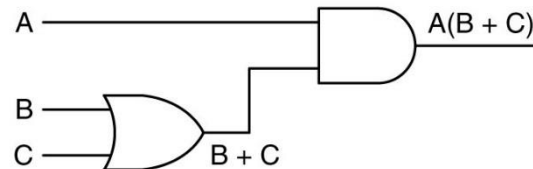
- Projetistas tentam reduzir o número de portas em seus produtos para reduzir custo de componentes, espaço de placa de circuito interno, consumo de energia elétrica....
- Duas funções são equivalentes se, e somente se, elas tiverem a mesma saída para todas as entradas possíveis.

## 3.1.4 Equivalência de circuito



A	B	C	AB	AC	AB + AC
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	0	0
1	0	1	0	1	1
1	1	0	1	0	1
1	1	1	1	1	1

(a)



A	B	C	A	B + C	A(B + C)
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	1	0
0	1	1	0	1	0
1	0	0	1	0	0
1	0	1	1	1	1
1	1	0	1	1	1
1	1	1	1	1	1

(b)

Duas funções equivalentes (a)  $AB + AC$ , (b)  $A(B + C)$ .

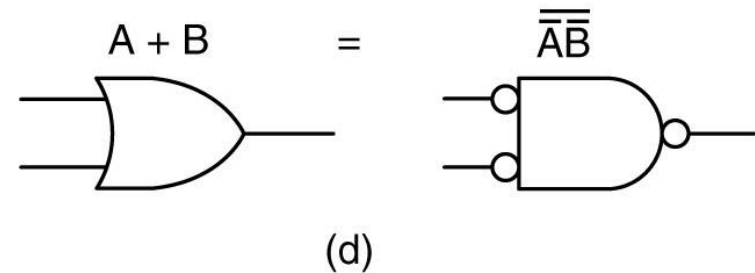
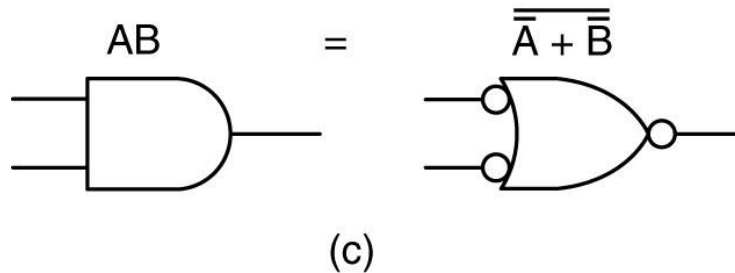
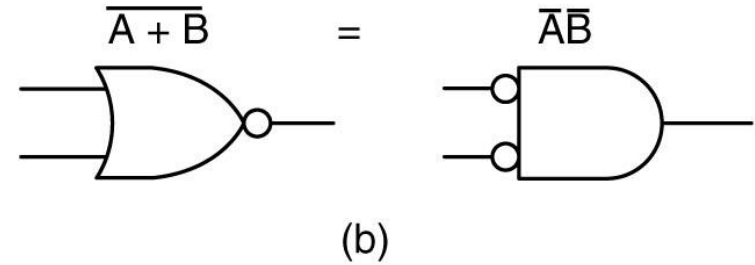
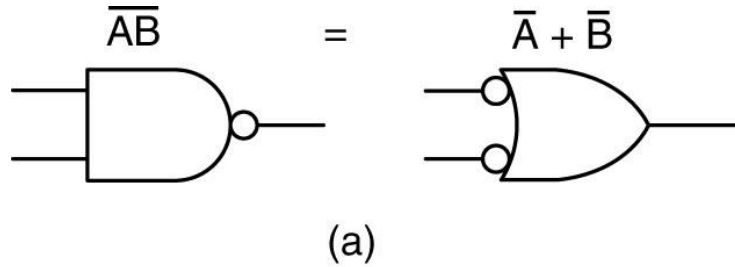
## 3.1.4 Equivalência de circuito

Nome	Forma AND	Forma OR
Lei da identidade	$1A = A$	$0 + A = A$
Lei do elemento nulo	$0A = 0$	$1 + A = 1$
Lei idempotente	$AA = A$	$A + A = A$
Lei do inverso	$A\bar{A} = 0$	$A + \bar{A} = 1$
Lei comutativa	$AB = BA$	$A + B = B + A$
Lei associativa	$(AB)C = A(BC)$	$(A + B) + C = A + (B + C)$
Lei distributiva	$A + BC = (A + B)(A + C)$	$A(B + C) = AB + AC$
Lei da absorção	$A(A + B) = A$	$A + AB = A$
Lei de DeMorgan	$\overline{AB} = \bar{A} + \bar{B}$	$\overline{A + B} = \bar{A}\bar{B}$

Algumas identidades da álgebra Booleana.



## 3.1.4 Equivalência de circuito



Símbolos alternativos para algumas portas:  
(a) NAND, (b) NOR, (c) AND, (d) OR

## 3.2.1 Circuitos Integrados

### ICs ou Chips

Um pedaço quadrado de silício de aproximadamente 5mm x 5mm no qual foram depositadas algumas portas.

ICs pequenos costumam ser montados em pedaços retangulares de plástico ou cerâmica medindo 5 a 15mm de largura, e 20 a 50mm de comprimento.



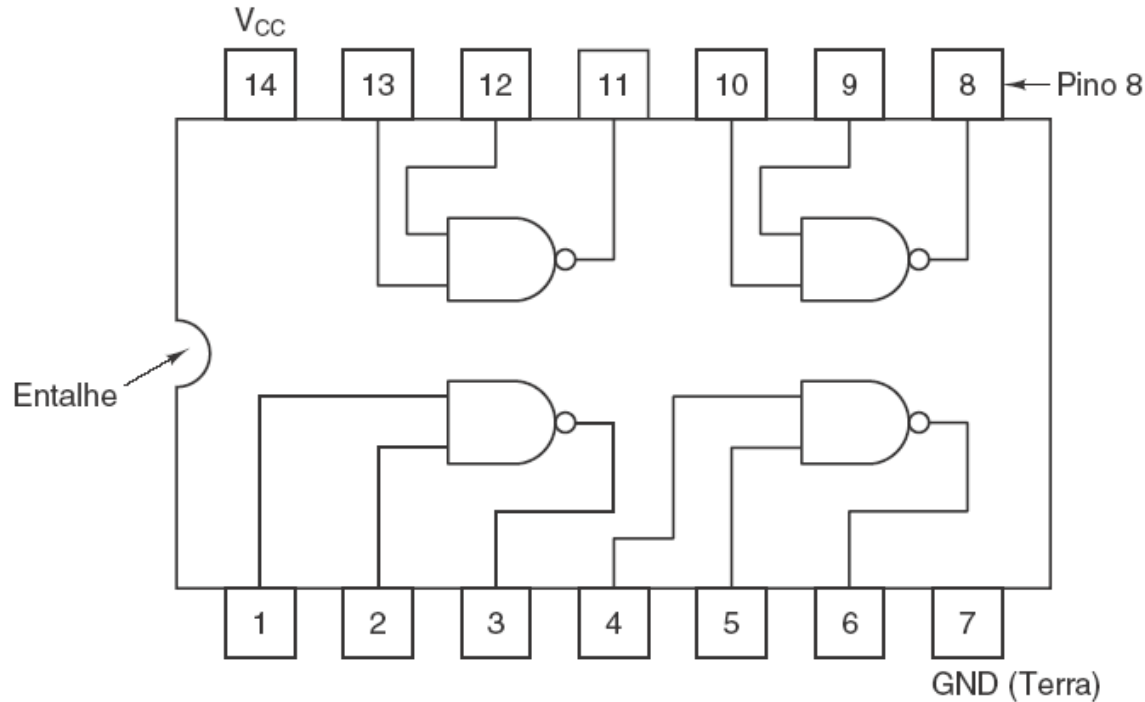
## 3.2.1 Circuitos Integrados

### Chips

Podem ser divididos em classes gerais com base no número de portas que o contém:

- **SSI** -> circuito de integração em pequena escala – 1 a 10 portas
- **MSI** -> circuito de integração em média escala – 10 a 100 portas
- **LSI** -> circuito de integração em grande escala – 100 a 100.000 portas
- **VLSI** -> circuito de integração em escala muito grande – > 100.000 portas

## 3.2.1 Circuitos Integrados



Chip SSI que contém quatro portas.

## 3.2.2 Circuitos Combinacionais

- Circuitos MSI simples que combinam uma quantidade de portas internamente para fornecer uma função útil que requer apenas um número limitado de conexões externas.
- Circuito com múltiplas entradas e múltiplas saídas, no qual as saídas são determinadas exclusivamente pelas entradas em questão

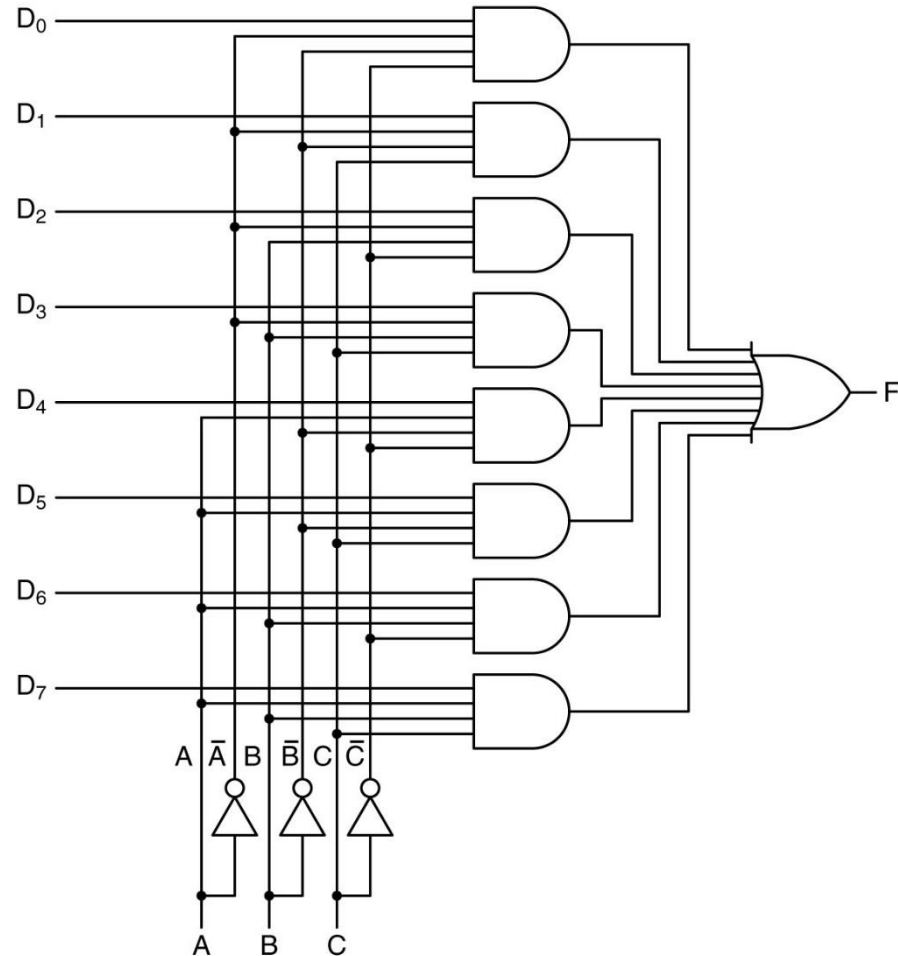
### Multiplexadores

- É um circuito com  **$2^n$  entradas de dados**, **uma saída de dados**, e  **$n$  entradas de controle** que selecionam uma das entradas de dados.
- Por exemplo, podemos implementar um chip da função majoritária.
- Outra aplicação é converter dados paralelos para serial (teclado).
- O inverso do multiplexador é um **Demultiplexador**, que dirige sua única entrada até uma das  $2^n$  saídas dependendo dos valores das  $n$  linhas de controle.

## 3.2.2 Circuitos Combinacionais

### Multiplexadores

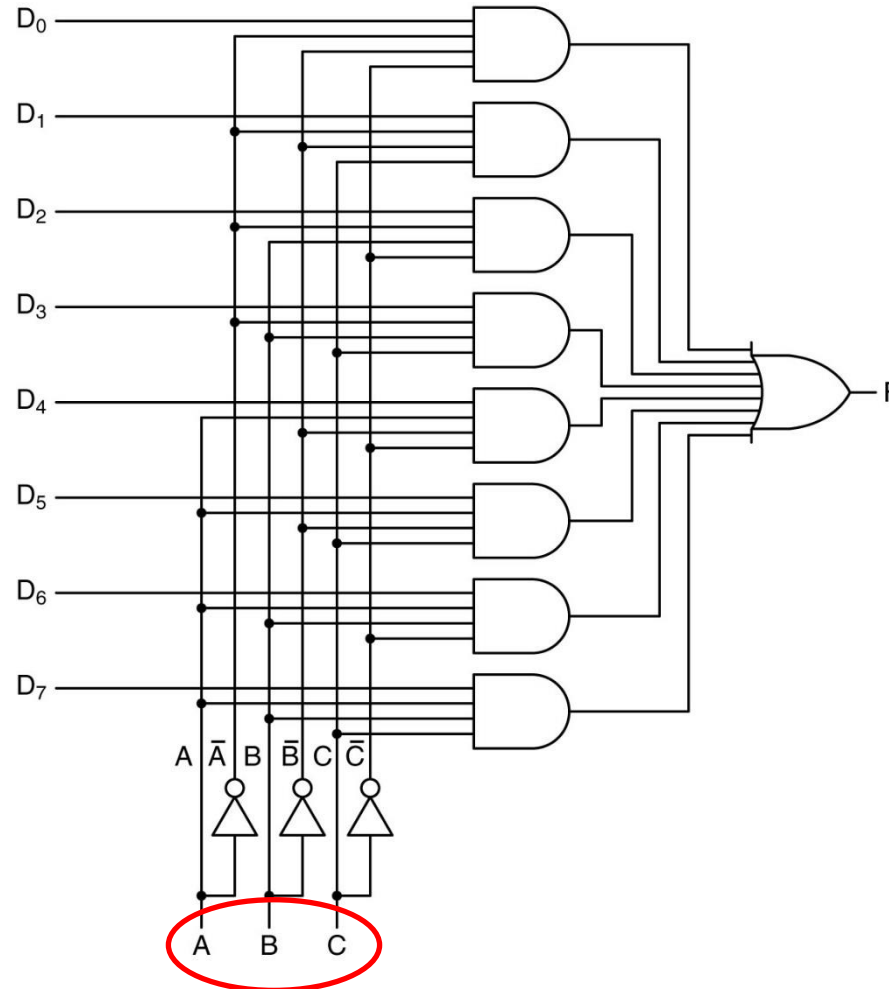
Circuito multiplexador de oito entradas.



## 3.2.2 Circuitos Combinacionais

### Multiplexadores

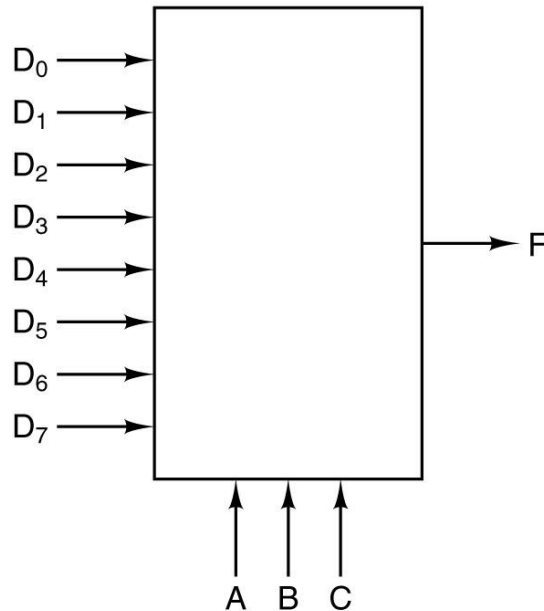
As 3 linhas de controle A, B e C, codificam um número de 3 bits que especifica qual das 8 linhas de entrada é direcionada a porta OR e dali até a saída



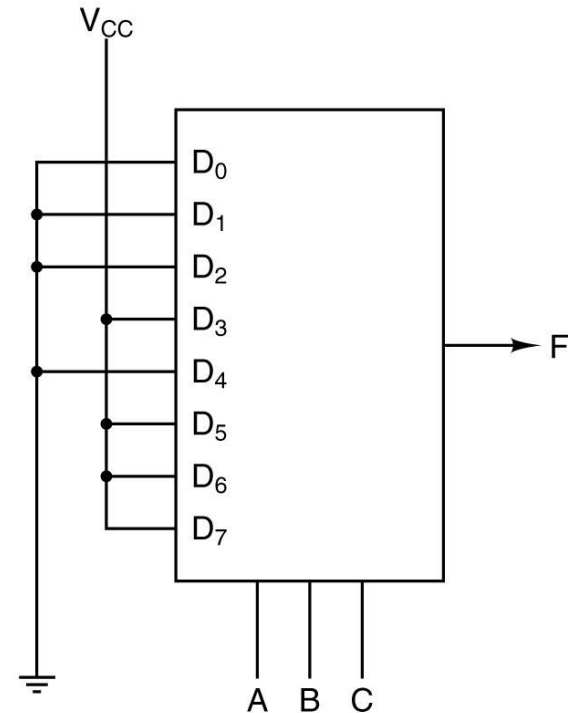


## 3.2.2 Circuitos Combinacionais

### Multiplexadores



(a)



(b)

(a) Multiplexador MSI.

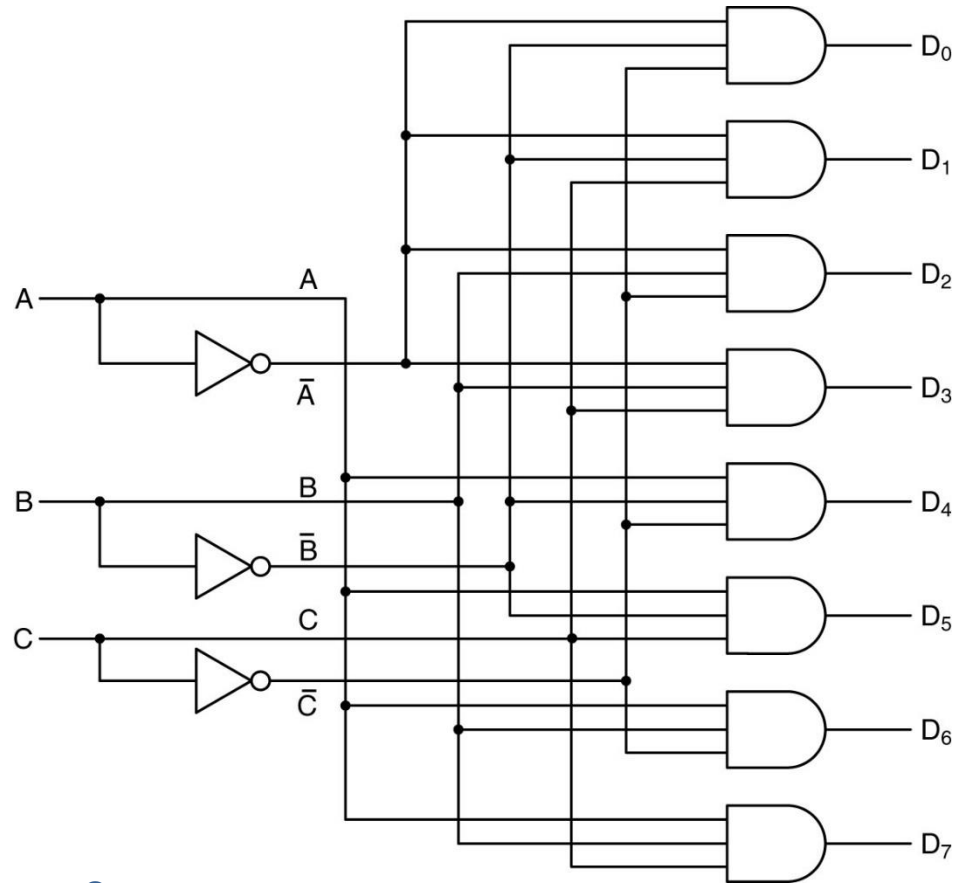
(b) O mesmo multiplexador ligado para calcular a função majoritária.

### Decodificadores

- Circuito que toma um número de  $n$  bits como entrada e a usa para selecionar (isto é colocar em 1) exatamente uma das  $2^n$  linhas de saída.
- Ex.
  - pequena memória que consiste em 8 chips;
  - cada chip contém 1M
  - o chip 0 tem endereço de 0 a 1M, o chip 1 tem endereço de 1M a 2M...
  - quando um endereço é apresentado a memória, 3 bits de ordem alta são usados para selecionar um dos 8 chips
  - esses 3 bits são representados por 3 entrada: A, B e C

## 3.2.2 Circuitos Combinacionais

### Decodificadores



Circuito decodificador 3 para 8.

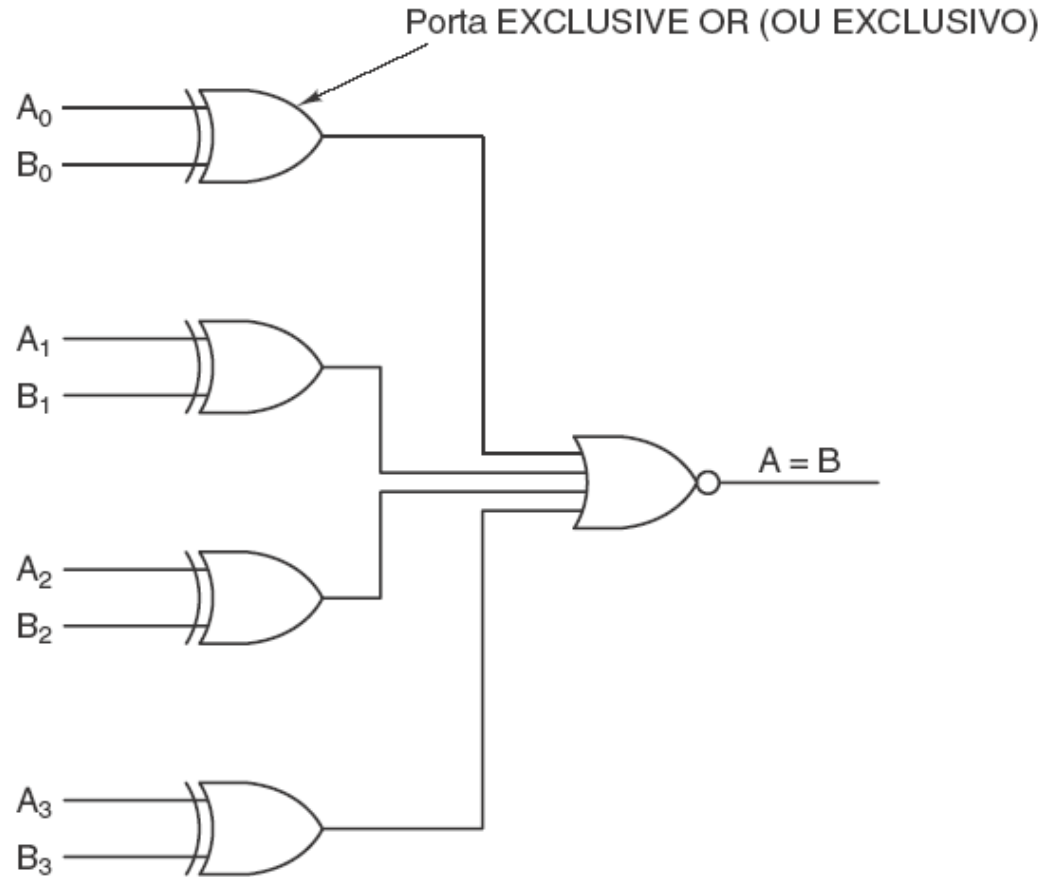
### Comparadores

- Compara duas palavras de entrada
- Produz um 1 se elas forem iguais, e um 0 se elas não forem iguais
- Ex.
  - comparador tem duas entradas A e B
  - cada uma de 4 bits de comprimento
  - um circuito baseado na porta XOR produz um 0 se se suas entradas forem iguais, e um 1 se elas forem diferentes
  - uma porta NOR reverte o estágio final: 1 significa igual e 0 significa diferente.

## 3.2.2 Circuitos Combinacionais

### Comparadores

Comparador simples de 4bits



## 3.2.2 Circuitos Combinacionais

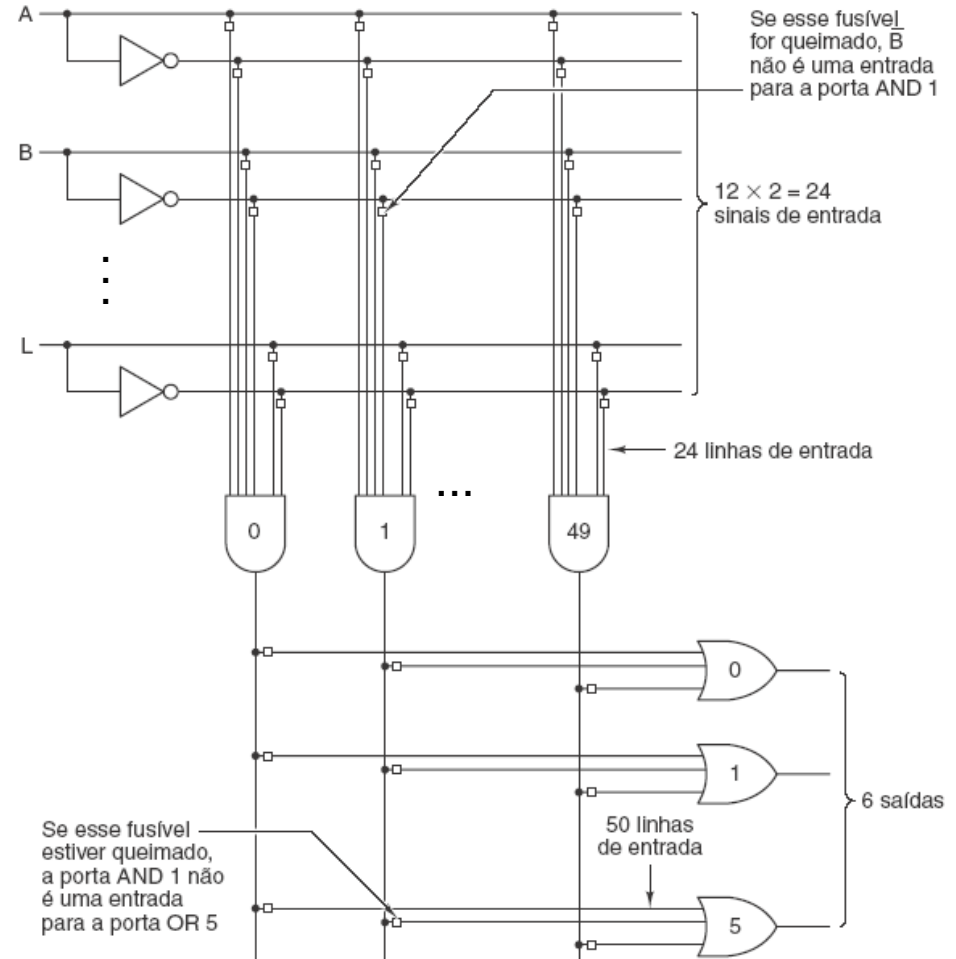
### Arranjos lógicos programáveis (PLA – Programmable Logic Array)

- Um chip muito geral para formar somas de produtos
- Ex.
  - um chip tem linhas de entrada para 12 variáveis
  - o complemento de cada entrada é gerado internamente, perfazendo 24 sinais de entrada no total
  - o coração do circuito é um arranjo de 50 portas AND
  - cada uma delas pode ter como uma entrada qualquer subconjunto dos 24 sinais de entrada
  - uma matriz 24x50 bits fornecida pelo usuário determina qual dos sinais de entrada vai para qual porta AND
  - cada linha de entrada para as 50 portas AND contém um fusível
  - quando expedidos de fábrica, todos os 1200 estão intactos
  - para programar a matriz o usuário queima fusíveis selecionados aplicando uma alta tensão no chip

## 3.2.2 Circuitos Combinacionais

### Arranjos lógicos programáveis

Arranjo lógico programável de 12 entradas e 6 saídas. Os quadradinhos representam fusíveis que podem ser queimados para determinar a função a ser calculada. Os fusíveis são arranjados em duas matrizes: a superior para as portas AND e a inferior para as portas OR.



### 3.2.3 Circuitos Aritméticos

- Circuito MSI combinacionais usados para efetuar aritmética.



## 3.2.3 Circuitos Aritméticos

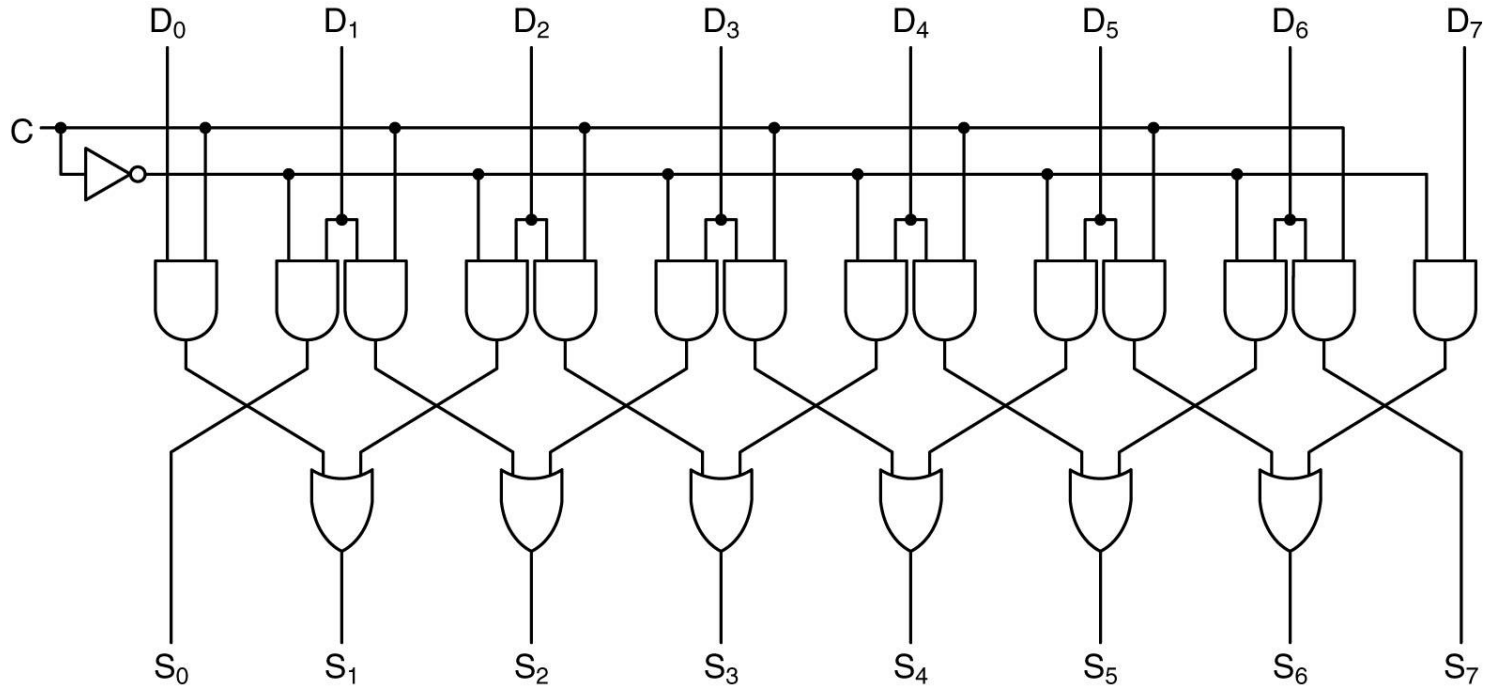
### Deslocadores

Ex.:

- Deslocador de 8 entradas e 8 saídas
- A saída é apenas a entrada deslocada 1 bit
- A linha de controle C determina a direção do deslocamento: **0 para a esquerda e 1 para a direita.**
- Quando o deslocamento for para a esquerda, um 0 é inserido no bit 7
- Quando for para a direita, um 1 é inserido no bit 0

## 3.2.3 Circuitos Aritméticos

### Deslocadores



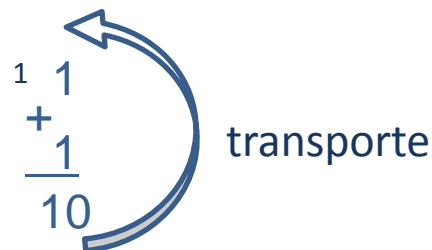
Deslocador esquerda/direita de 1 bit

## 3.2.3 Circuitos Aritméticos

### Somadores

Meio Somador

0	0	1	<sup>1</sup> 1	
+ 0	+ 1	+ 0	+ 1	
—	—	—	—	
0	1	1	10	transporte

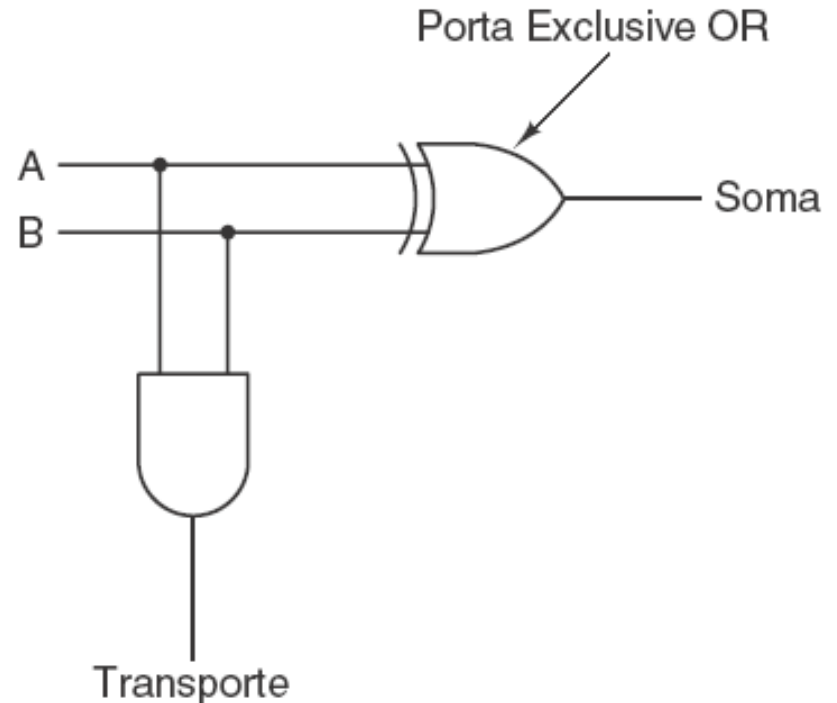


Em um circuito para calcular o bit de soma e o bit de transporte há duas saídas presentes: a soma das entradas, A e B, e o transporte (vai-um) para a posição seguinte (à esquerda)

## 3.2.3 Circuitos Aritméticos

### Somadores

A	B	Soma	Transporte
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



- (a) Tabela-verdade para adição de 1 bit.  
(b) Circuito para um meio-somador.

## 3.2.3 Circuitos Aritméticos

### Somadores

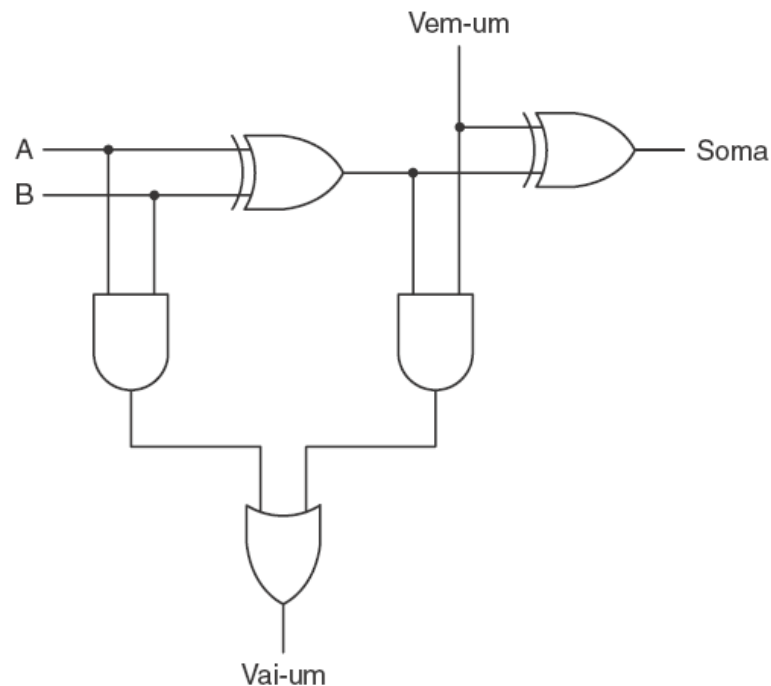
- É composto de dois meio-somadores
- Meio-somador é adequado para somar os bits de ordem baixa de duas palavras de entrada de múltiplos bits. Não serve para uma posição de bit no meio da palavra, porque não trata transporte de bit da posição à direita (vem-um)
- A linha de saída Soma é 1 se um número ímpar A, B e o vem-um forem 1
- O vai-um é 1 se A ou B forem ambos 1 ou exatamente um deles for 1 e o bit de vem-um também for 1
- Juntos, os dois meio-somadores geram a soma e também os bits de transporte

## 3.2.3 Circuitos Aritméticos

### Somadores

A	B	Vem-um	Soma	Vai-um
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

(a)



(b)

(a) Tabela-verdade para um somador completo.

(b) Circuito para um somador completo.

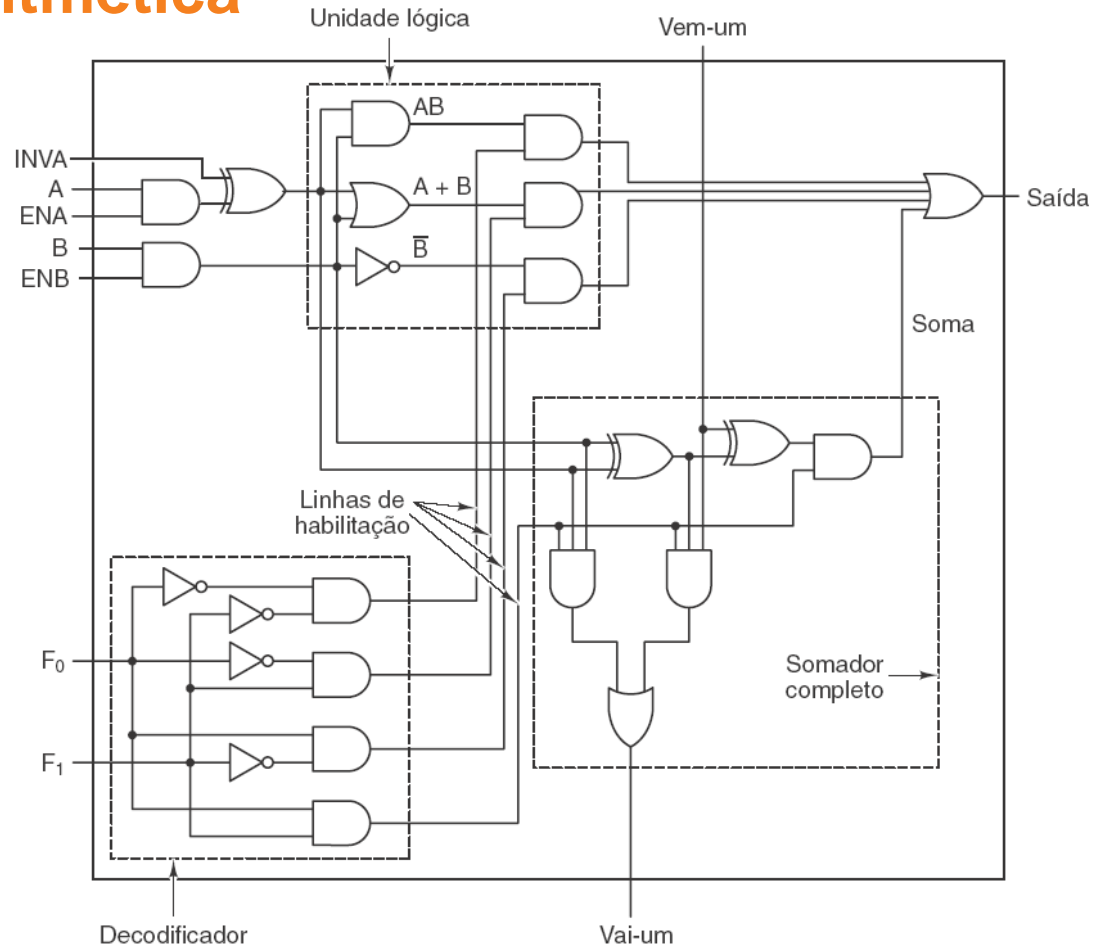
## 3.2.3 Circuitos Aritméticos

### Unidade lógica e aritmética (ULA)

- Um único circuito para efetuar AND, OR e soma de duas palavras de máquina
- Tal circuito para palavras de  $n$  bits é composto de  $n$  circuitos idênticos para as posições individuais de bits
- Ex.:
  - um circuito simples (ULA) que pode calcular qualquer uma das quatro funções:  $A \text{ AND } B$ ,  $A \text{ OR } B$ ,  $\bar{B}$  ou  $A + B$
  - dependem das linhas de entrada de seleção de função  $F_0$  e  $F_1$  conterem: 00, 01, 10 ou 11 (binário)

## 3.2.3 Circuitos Aritméticos

### Unidade lógica e aritmética



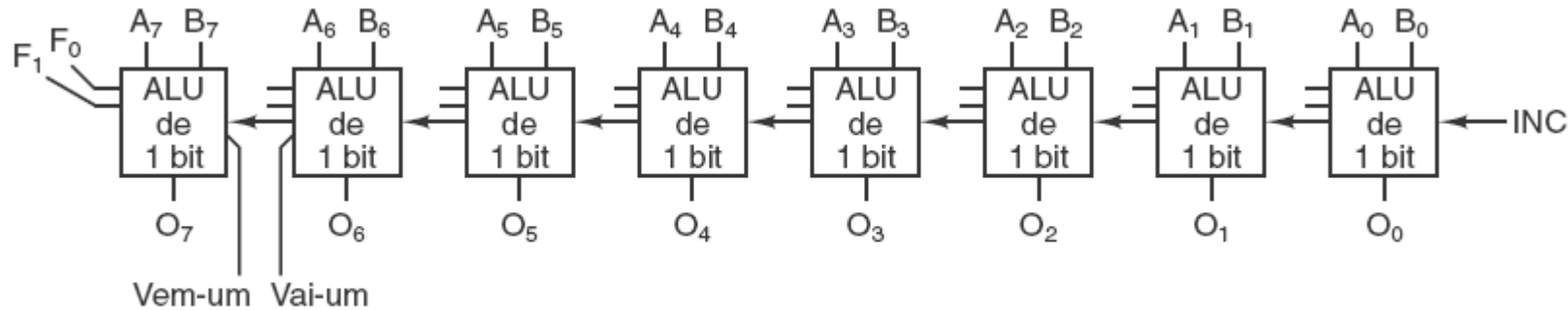
ULA de 1 bit



## 3.2.3 Circuitos Aritméticos

### Unidade lógica e aritmética

- Vários desses circuitos são ligados juntos para efetuar operações de palavra inteira, conhecidos como **segmento de bits**.
- Permite que o projetista monte uma ULA de qualquer largura que quiser
- A figura mostra uma ULA de 8 bits montada com 8 segmentos de ULA de 1 bit:



Oito segmentos de ALU de 1 bit conectados para formar uma ALU de 8 bits.  
Os sinais de habilitação e de inversão não são mostrados por simplicidade.

## 3.2.4 Relógios

- A ordem que eventos ocorre em circuitos digitais é crítica
- As vezes um evento deve preceder a outro, as vezes devem ocorre simultaneamente
- Relógios são usados para providenciar essas sincronizações

### Relógio

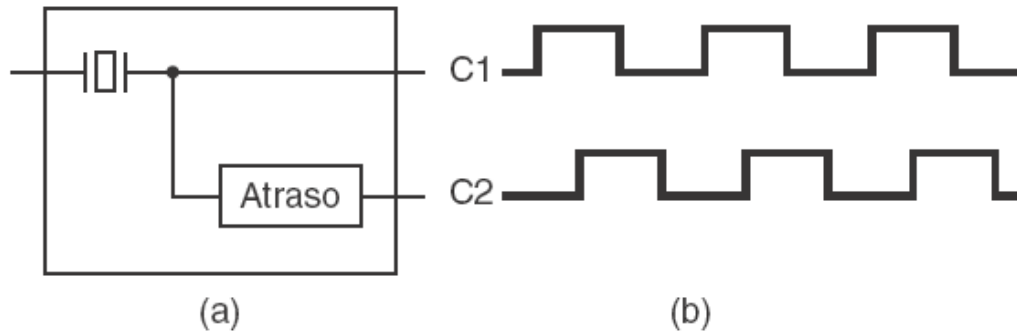
- ✓ É um circuito que emite uma série de pulsos com uma largura de pulso precisa e intervalos precisos entre pulsos consecutivos

### Tempo de ciclo de relógio

- ✓ É o intervalo de tempo entre as arestas correspondentes de dois pulsos consecutivos

## 3.2.4 Relógios

- Muitos eventos podem ocorrer dentro durante um único ciclo de relógio
- Se devem ocorrer em ordem específica, o ciclo de relógio deve ser subdividido em subciclos.



(a) Relógio.  
(b) Diagrama de temporização para o relógio.

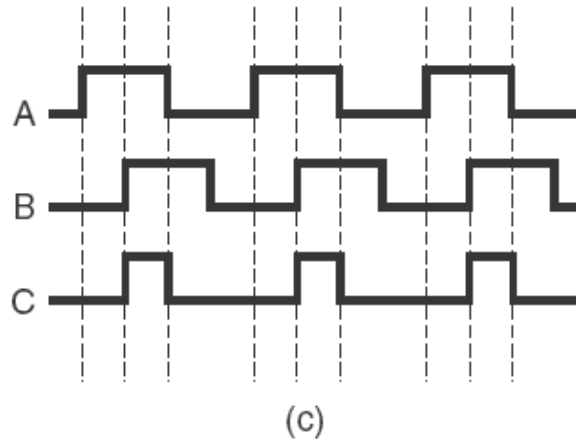
1. Fase ascendente de C1
2. Fase descendente de C1
3. Fase ascendente de C2
4. Fase descendente de C2

- Em alguns circuitos estamos interessados em intervalos de tempo em vez de instantes discretos de tempo.

Ex.: pode-se permitir que algum evento aconteça toda vez que C1 estiver no alto, em vez de exatamente na fase ascendente.

## 3.2.4 Relógios

- Relógios são simétricos: o tempo gasto no estado alto é igual ao tempo gasto no estado baixo
- Para gerar um trem de pulso assimétrico, o relógio básico é deslocado usando um circuito de atraso e efetuando uma operação AND com o sinal original.



(c) Geração de um relógio assimétrico.

## 3.3 Memória

- Usada para armazenar instruções a serem executadas e dados
- Examinaremos os componentes básicos de um sistema de memória começando no nível da porta para ver como eles funcionam e como são combinados para produzir memórias de grande porte

## 3.3.1 Memória de 1 bit

- Memórias de 1 bit (“latch”) pode ser construído com base em duas portas NOR

### Latch SR

- ✓ Tem duas entradas: S para ativar o latch, e R para restaurá-lo (liberá-lo)
- ✓ Tem duas saídas  $Q$  e  $\bar{Q}$ , que são complementares
- ✓ As saídas são exclusivamente determinadas pelas entradas atuais

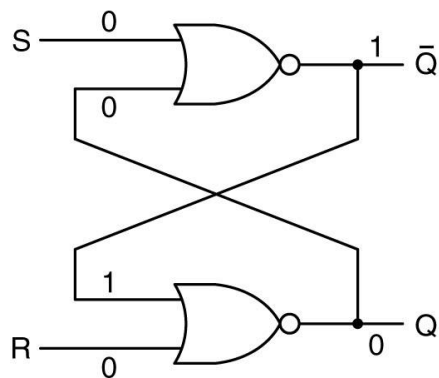
S	R	Qa	Qf	$\bar{Qf}$	
0	0	0	0	1	} Fixa Qf = Qa
0	0	1	1	0	
0	1	0	0	1	} Fixa Qf em 0
0	1	1	0	1	
1	0	0	1	0	} Fixa Qf em 1
1	0	1	1	0	
1	1	0	1	1	} Não permitido
1	1	1	1	1	

S	R	Qf
0	0	Qa
0	1	0
1	0	1
1	1	

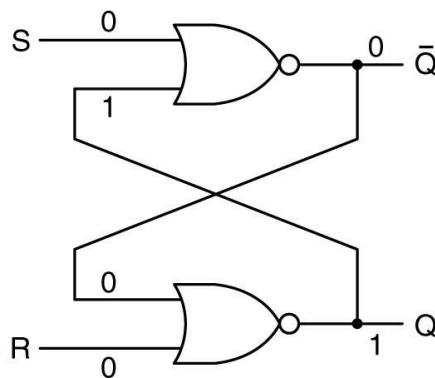
## 3.3.1 Memória de 1 bit

### Resumindo

- ✓ Quando S é ativado para 1 momentaneamente, o latch acaba no estado  $Q=1$ , independentemente do estado que estava antes
- ✓ Da mesma maneira ativar R para 1 momentaneamente força o latch ao estado  $Q=0$
- ✓ O circuito “se lembra”, se foi S ou R, ligado por ultimo.



(a)



(b)

A	B	NOR
0	0	1
0	1	0
1	0	0
1	1	0

(c)

(a) Latch NOR no estado 0.

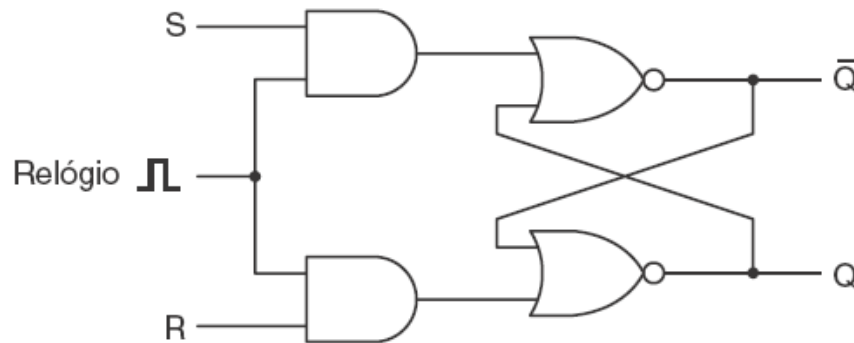
(b) Latch NOR no estado 1.

(c) Tabela-verdade para NOR.

## 3.3.1 Memória de 1 bit

### Latch com relógio

- ✓ Muitas vezes é conveniente impedir que o latch mude de estado, a não ser em certos momento específicos
- ✓ Uma ligeira modificação no circuito básico obtém um latch com relógio



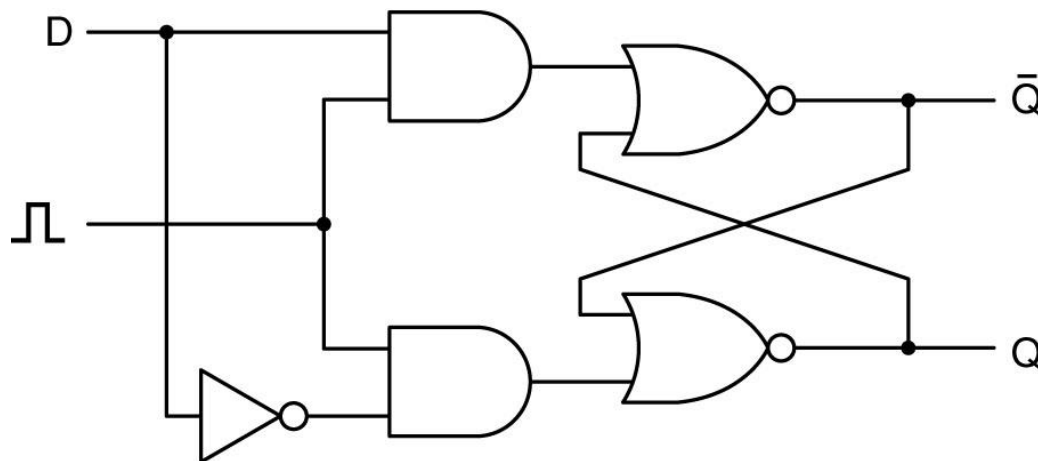
- ✓ Relógio em 0, ambas as portas AND geram saída 0, independentemente de ser S ou R, e o latch não muda de estado
- ✓ Relógio em 1, o efeito das portas AND desaparece e o latch se torna sensível a S e R



### 3.3.1 Memória de 1 bit

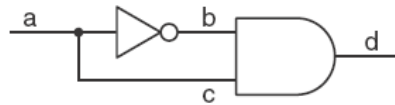
#### Latch D com relógio

- ✓ Uma boa maneira de resolver a ambiguidade do latch SR ( $S=R=1$ ) é evitar que ela ocorra
- ✓ **Latch D com relógio** apresenta um circuito com somente uma entrada D, onde a entrada para a porta inferior AND é sempre complemento da entrada para a porta superior
- ✓ Nunca ocorre de ambas entradas serem 1

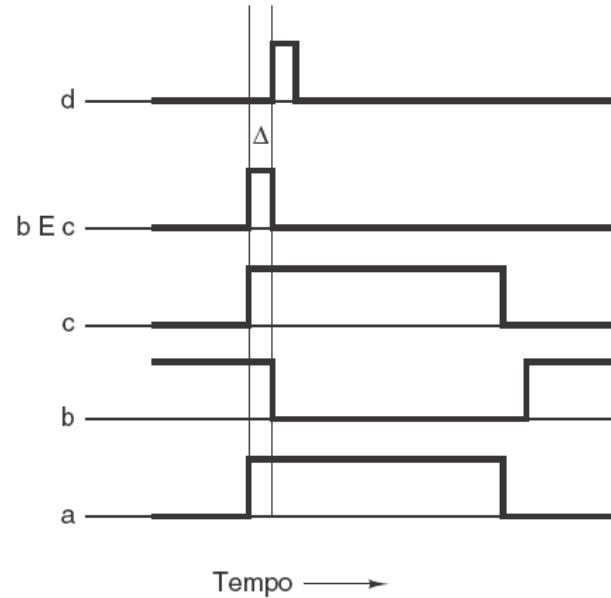


## 3.3.2 Flip-flop

- ✓ A transição de estado não ocorre quando o relógio é 1, mas **durante a transição do relógio** de 0 para 1 (borda ascendente), ou de 1 para 0 (borda descendente)



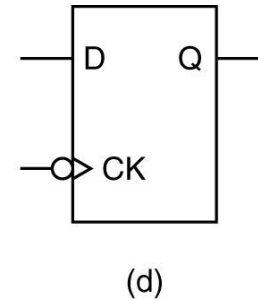
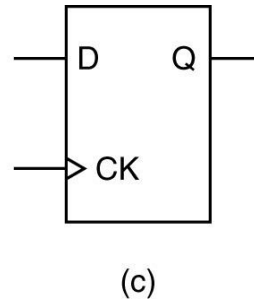
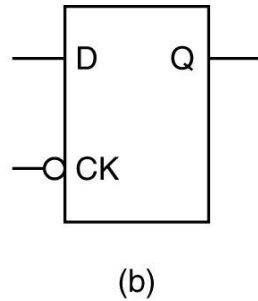
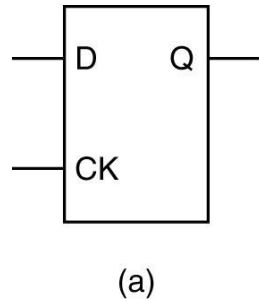
(a)



(b)

## 3.3.2 Flip-flop

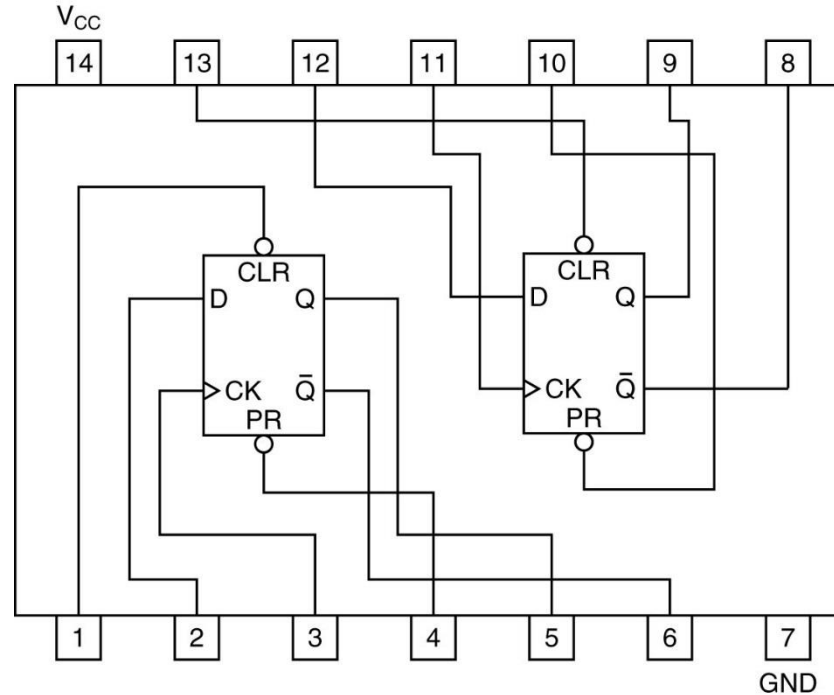
- ✓ Esse deslocamento de tempo significa apenas que o lantch de D será ativado com um atraso fixo após a fase ascendente do relógio, mas não tem efeito sobre a largura do pulso



- ✓ a. lantch cujo estado é carregado quando o relógio é 1
- ✓ b. um lantch cujo relógio é 1, mas cai para 0 momentaneamente para carregar o estado a partir de D
- ✓ c. flip-flop: muda de estado na borda ascendente do pulso de relógio (transição de 0 para 1)
- ✓ d. flip-flop: muda de estado na borda descendente do pulso de relógio (transição de 1 para 0)

### 3.3.3 Registradores

- ✓ Há uma variedade de configurações de flip-flops disponíveis
- ✓ Ex1.

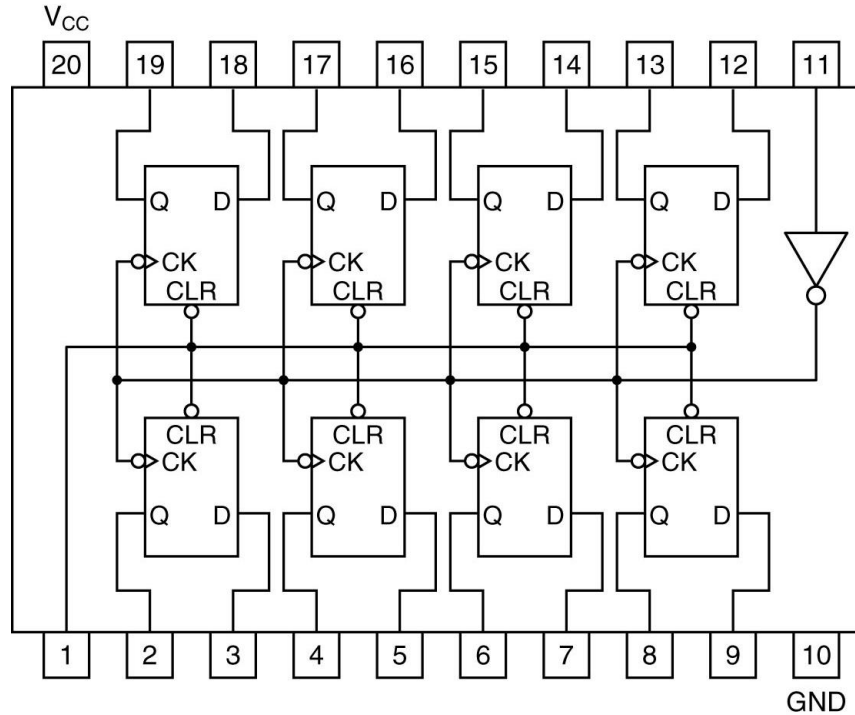


(a)

(a) Flip-flop D dual: contém dois flip-flops D independentes com sinais *clear* e *preset*

## 3.3.3 Registradores

✓ Ex 2.



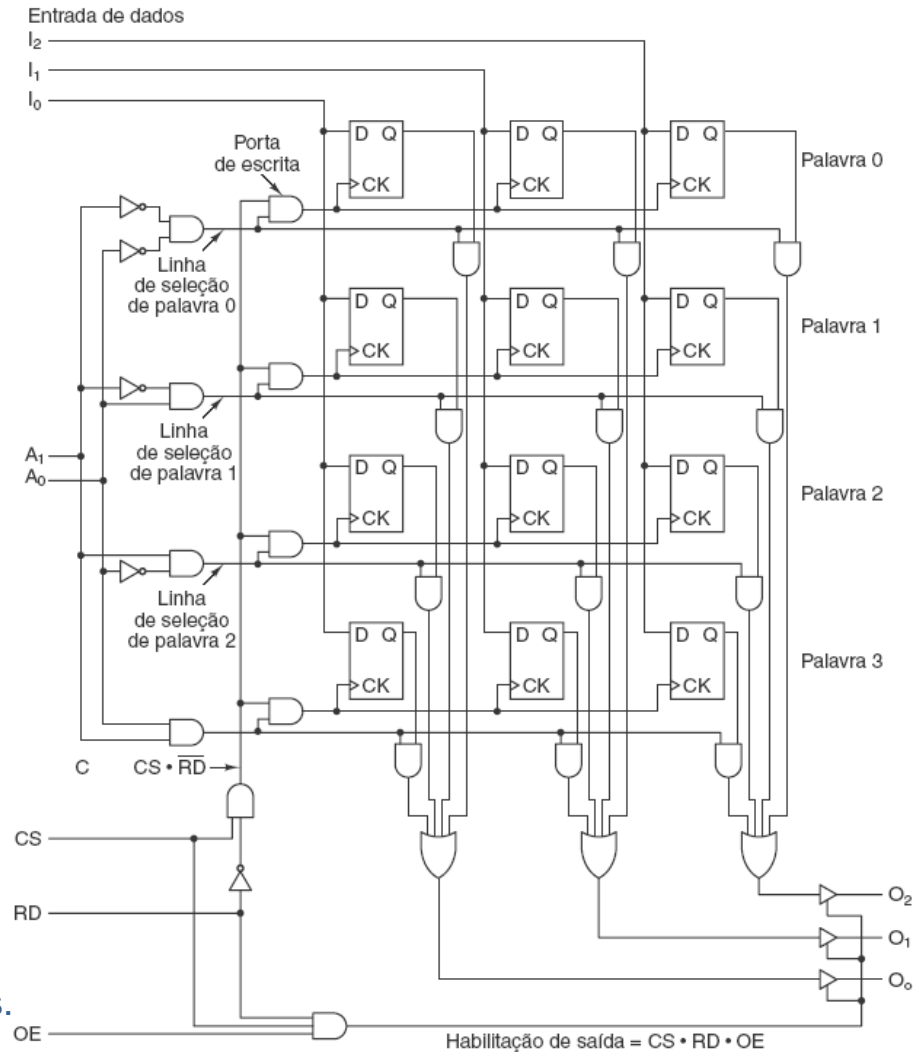
(b)

(b) Flip-flop octal: são carregados na borda ascendente, é usado como um registrador único de 8 bits

## 3.3.4 Organização da memória

- ✓ Para construir memórias grandes é preciso uma organização na qual palavras individuais podem ser endereçadas
- ✓ Ex.: uma memória com 4 palavras de 3 bits
  - Cada operação lê ou escreve uma palavra completa de 3 bits

Diagrama lógico para uma memória 4 x 3.  
Cada linha é uma das quatro palavras de 3 bits.

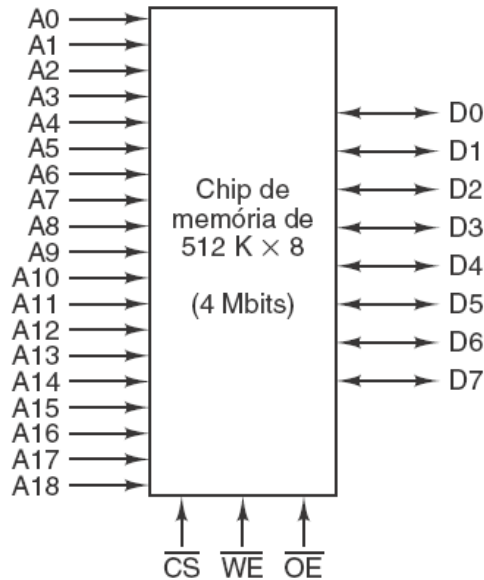


### 3.3.5 Chips de memória

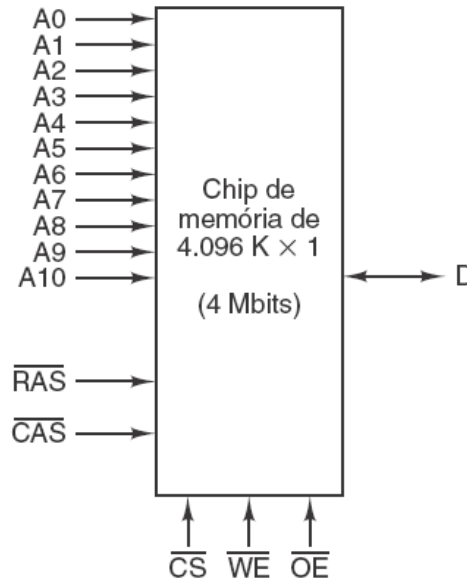
- ✓ No exemplo a memória é 4x3, isto é, quatro palavras de 3 bits cada
- ✓ Para ampliá-la para **4x8**, basta adicionar mais 5 colunas de 4 flip-flops cada, bem como mais 5 linhas de entrada e mais 5 linhas de saída
- ✓ Para passar de 4x3 para **8x3**, devemos acrescentar mais 4 linhas de três flop-flop cada, bem como uma linha de endereço  $A_2$ .
- ✓ Nesta estrutura o número de palavras na memória deve ser uma potência de 2 para eficiência máxima, mas o número de bits pode ser qualquer um
- ✓ Há vários modos de organizar o chip para qualquer tamanho de memória dado

### 3.3.5 Chips de memória

- ✓ Duas organizações possíveis para um chip de memória mais antigo, de 4 Mbits, de tamanho: **512k x 8** e **4.096k x 1**
- (a) São necessárias 19 linhas de endereço para endereçar em do  $2^{19}$  bytes e 8 linhas de dados para carregar e armazenar o byte selecionado
- (b) O chip é organizado internamente como uma matriz 2.048 x 2.048 de células de 1 bit, o que dá 4Mbits



(a)



(b)



### 3.3.5 Chips de memória

- ✓ Chips de memória de grande porte costumam ser construídos como matrizes  $n \times n$  endereçadas por linhas e colunas
- ✓ Essa organização reduz o numero de pinos requerido, mas também torna mais lento o endereçamento do chip, uma vez que são necessários dois ciclos de endereçamento: uma para linha e outro para a coluna
- ✓ Para recuperar um pouco da velocidade perdida por esse projeto, alguns chips de memória podem receber um endereço de linha acompanhado por uma sequência de endereços de coluna para acessar bits consecutivos em uma linha

## 3.3.6 RAMs e ROMs

### RAMs (Random Access Memories - memórias de acesso aleatório)

- ✓ Tem um nome suspeito, porque todos os chips de memória tem acesso aleatório
- ✓ Podem ser escritas e lidas
- ✓ Podem ter duas variedades: estáticas e dinâmicas



## 3.3.6 RAMs e ROMs

### SRAMs (Static RAMs)

- ✓ Seus conteúdos são conservados enquanto houver fornecimento de energia
- ✓ Usa circuitos similares a flip-flop D básico
- ✓ São muito rápidas, por isso são muito usadas como memória cache de nível 2

## 3.3.6 RAMs e ROMs

### DRAMs (Dynamic RAMs)

- ✓ Não usam flip-flops
- ✓ É um arranjo de células, cada uma contendo um transistor e um pequenino capacitor
- ✓ Os capacitores podem ser carregados e descarregados, permitindo que 0s e 1s sejam armazenados
- ✓ Como a carga elétrica tende a vaziar, cada bit em uma RAM dinâmica deve ser **renovado**.
- ✓ A lógica externa que tem de executar essa renovação, por isso precisam de uma interface mais complexa do que as estáticas
- ✓ Essa desvantagem é compensada por maiores capacidades. Densidades muito mais altas (bits por chip)
- ✓ LENTAS

## 3.3.6 RAMs e ROMs

### SDRAMs (Synchronous DRAM – DRAM síncrona)

- ✓ Híbrida de RAM estática e dinâmica, comandada pelo relógio do sistema principal
- ✓ O relógio elimina a necessidade de sinais de controle para informar ao chip de memória quando responder
- ✓ A CPU informa à memória por quanto ciclos ela deve funcionar e então inicia
- ✓ Eliminar a necessidade de sinais de controle aumenta a taxa de dados entre CPU e memória

## 3.3.6 RAMs e ROMs

### ROMS (Read-Only-Memores)

- ✓ Dados devem permanecer armazenados mesmo quando o fornecimento de energia for interrompido
- ✓ Uma vez instalados, nem programas nem dados são alterados
- ✓ Dados são inseridos durante sua fabricação
- ✓ A única maneira de mudar o programa é substituir o chip inteiro
- ✓ São mais baratas

Tipos: PROM, EPROM, EEPROM

## 3.3.6 RAMs e ROMs

### PROM (Programmable ROM)

- ✓ Pode ser programada (uma vez) em campo, eliminando o tempo de produção e entrega

### EPROM (Erasable PROM – PROM apagável)

- ✓ Não somente pode ser programada, mas também apagada em campo
- ✓ Exposta a uma forte luz ultravioleta durante 15 minutos, todos os bits são fixados em 1

### EEPROM

- ✓ Pode ser apagada aplicando-se pulsos em vez de ser exposta à luz ultravioleta
- ✓ Um tipo recente: memória flash

## 3.3.6 RAMs e ROMs

Comparação entre vários tipos de memória:

Tipo	Categoria	Modo de apagar	Byte alterável	Volátil	Utilização típica
SRAM	Leitura/escrita	Elétrico	Sim	Sim	Cache de nível 2
DRAM	Leitura/escrita	Elétrico	Sim	Sim	Memória principal (antiga)
SDRAM	Leitura/escrita	Elétrico	Sim	Sim	Memória principal (nova)
ROM	Somente de leitura	Não é possível	Não	Não	Equipamentos e grande volume
PROM	Somente de leitura	Não é possível	Não	Não	Equipamentos e pequeno volume
EPROM	Principalmente leitura	Luz UV	Não	Não	Prototipagem de dispositivos
EEPROM	Principalmente leitura	Elétrico	Sim	Não	Prototipagem de dispositivos
Flash	Leitura/escrita	Elétrico	Não	Não	Filme para câmera digital