Introdução à Arquitetura de Computadores

Aula 3
Lógica Binária e Álgebra de Boole

Pedro M. Lavrador

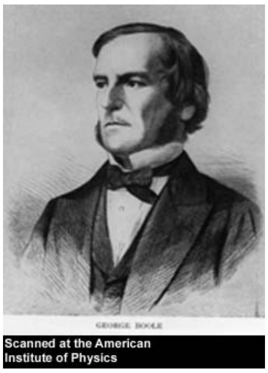
Departamento de Electrónica, Telecomunicações e Informática
Universidade de Aveiro
plavrador@ua.pt

Índice

- Lógica binária.
  - As operações lógicas básicas.
  - Portas Lógicas.
- Álgebra de Boole.
  - Axiomas e teoremas.
  - Soma de Produtos e Produto de Somas.
  - Minimização de funções booleanas.

Introdução

- George Boole (1815-1864)



- Introduziu as variáveis binárias e os três operadores lógicos fundamentais: AND, OR e NOT
- Foi o percursor da lógica binária na qual se baseiam atualmente os sistemas digitais.

Operações Lógicas:

- Variáveis lógicas (ou Booleanas):
  - Podem assumir dois valores: 1 ou 0 (V ou F).
- A álgebra booleana fornece as ferramentas matemáticas necessárias: AND, OR e NOT

A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

A	NOT A
0	1
1	0

- Operandos com n-bits são tratados como um coleção de n valores:
  - A operação é efetuada de modo independente em cada bit.

Operações Lógicas:

- Operações NAND e NOR

A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

A	B	A NAND B
0	0	1
0	1	1
1	0	1
1	1	0

NAND: Not AND

A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

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

NOR: Not OR

Operações Lógicas:

- Operação *Xclusive* OR

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

$$A\ XOR\ B = A.\bar{B} + \bar{A}.B$$

- Operação *Xclusive* NOR

A	B	A XNOR B
0	0	1
0	1	0
1	0	0
1	1	1

$$A\ XNOR\ B = A.B + \overline{A.B}$$

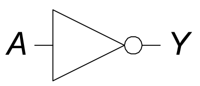
Introdução:

- Claude Shannon (1916-2001) provou que era possível construir circuitos elétricos digitais que resolvessem todos os problemas que a Álgebra de Boole pode resolver.
- Dá-se o nome de Porta Lógica (*Logic Gate*) aos circuitos lógicos que implementam as funções lógicas.
  - Uma entrada: Porta NOT
  - Duas Entradas: AND, OR, NAND, NOR, XOR, XNOR
  - Entradas Múltiplas

Introdução:

- Portas Lógicas

NOT



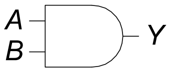
$$Y = \overline{A}$$

A	Y
0	
1	

Introdução:

- Portas Lógicas

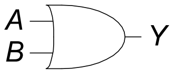
AND



$Y = AB$

A	B	Y
0	0	
0	1	
1	0	
1	1	

OR



$Y = A + B$

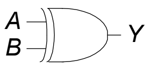
A	B	Y
0	0	
0	1	
1	0	
1	1	

Lógica Binária

Introdução:

- Portas Lógicas

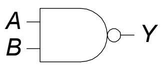
XOR



$Y = A \oplus B$

A	B	Y
0	0	
0	1	
1	0	
1	1	

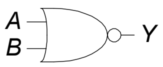
NAND



$Y = \overline{AB}$

A	B	Y
0	0	
0	1	
1	0	
1	1	

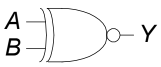
NOR



$Y = \overline{A + B}$

A	B	Y
0	0	
0	1	
1	0	
1	1	

XNOR



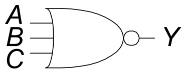
$Y = \overline{A \oplus B}$

A	B	Y
0	0	
0	1	
1	0	
1	1	

Introdução:

- Portas Lógicas

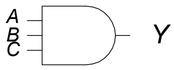
NOR3



$Y = \overline{A+B+C}$

A	B	C	Y
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

AND3



$Y = ABC$

A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

Introdução:

- Níveis Lógicos
- Os valores 0 e 1 existem nos circuitos como tensões. Por exemplo:
  - 0 corresponde a *ground* ou 0V
  - 1 corresponde à tensão de alimentação ou 5V
- Então e 4,99V? Corresponde a 0 ou 1?
  - E 3.2V?
- Há uma gama de tensões que correspondem ao nível lógico 0 e uma gama que corresponde ao 1.

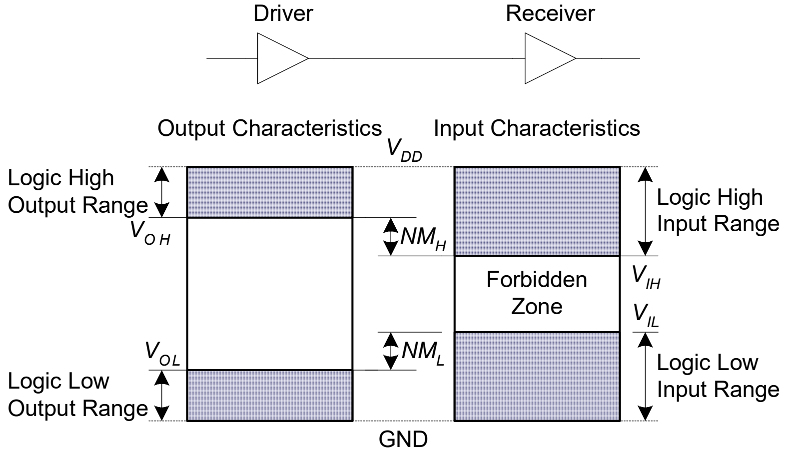
Introdução:

- Ruído?
- O que é o ruído?
  - Tudo o que degrade a qualidade do sinal.
  - Pode ser uma resistência, imperfeições da fonte, acoplamento aos circuitos vizinhos…

- Vamos definir agora **margem de ruído**

Introdução:

- Margem de Ruído (Noise Margin)



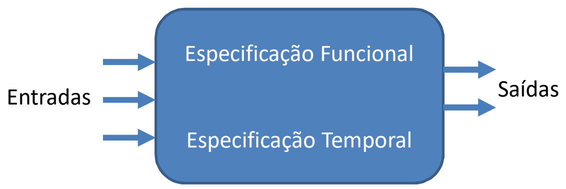
$$NM_H = V_{OH} - V_{IH} \qquad NM_L = V_{IL} - V_{OL}$$

Introdução:

- Os circuitos lógicos podem ser de 2 tipos:
  - Lógica combinatória
    - Sem memória, i.e.,
    - As saídas são determinadas apenas pelos valores atuais das entradas.
  - Lógica Sequencial
    - Com memória
    - As saídas são determinadas pelas entradas atuais e passadas.

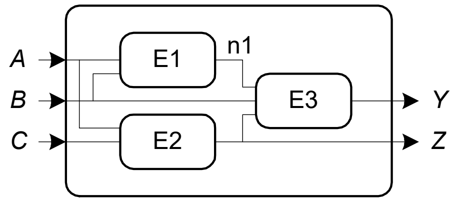
Introdução:

- Um **circuito lógico** tem:
  - Entradas
  - Saídas
  - Especificação Funcional
  - Especificação Temporal



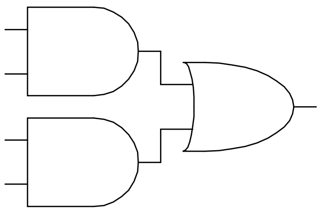
Introdução:

- Um circuito lógico pode ser composto por sub-circuitos e, nesse caso, definem-se:
  - Elementos de circuito (E1, E2, …)
  - Nós
    - Entradas
    - Saídas
    - Internos: n1



Introdução:

- Num circuito combinatório:
  - Cada elemento tem que ser combinatório;
  - Cada nó é uma entrada ou liga a uma (e uma só) saída de um elemento;
  - Não existem caminhos cíclicos.



Introdução:

- Especificações temporais:
  - Os componentes eletrónicos usados nos circuitos lógicos são contínuos e não discretos:
    - Logo, as transições entre estado não são instantâneas e por isso há um intervalo de tempo desde que as entradas mudam até que o valor da saída estabilize.
    - Durante esse período de tempo pode ocorrer que o valor da saída seja indefinido.
    - Chamamos a esse período **tempo de propagação** e está associado à frequência máxima a que um circuito digital pode operar.
- A álgebra de Boole descreve o comportamento dos sistemas digitais em regime estacionário sem considerar estas variações nas transições.

Introdução:

- Especificação Funcional descreve a relação (lógica) entre as entradas e a(s) saída(s) do circuito lógico.
- Normalmente é especificada como uma tabela de verdade ou como uma equação Booleana.
  - Vamos estudar agora como obter uma equação Booleana a partir de uma tabela de verdade, e como fazer a sua simplificação, usando:
    - Álgebra de Boole
    - Mapas de Karnaugh
  - Vamos ver também como implementar uma equação usando portas lógicas.

Índice

- Lógica binária.
  - As operações lógicas básicas.
- Álgebra de Boole.
  - Axiomas e teoremas.
  - Soma de Produtos e Produto de Somas.
  - Minimização de funções booleanas.

Motivação: O piquenique do Ben

- O Ben foi fazer um piquenique. Ele gosta de piqueniques se não chover e se não aparecerem formigas.
- Escreva uma equação lógica que determine em função de chover e aparecerem formigas se o Ben gosta do piquenique.
  - Definir as entradas
  - Definir a saída
  - Formalizar o sistema.

Motivação: O piquenique do Ben

- O problema consiste em determinar se o Ben gostou ou não do piquenique.
  - Chamamos à variável de saída G.
- As entradas do sistema são:
  - A Chuva: C
  - As Formigas: F
- Podemos escrever a tabela de verdade do problema:

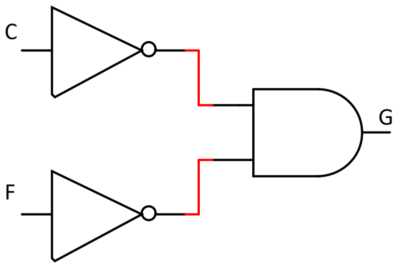
C	F	G
0	0	1
0	1	0
1	0	0
1	1	0

Motivação: O piquenique do Ben

- Podemos agora descrever a saída do problema como:

$$G = \bar{C}\bar{F}$$

- E G pode ser obtido de C e F através do seguinte circuito:



C	F	G
0	0	1
0	1	0
1	0	0
1	1	0

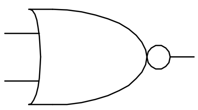
Motivação: O piquenique do Ben

- Podemos também reconhecer a tabela de verdade da função NOR:

$$G = \overline{C + F}$$

- E G pode ser obtido de C e F através do seguinte circuito:

C	F	G
0	0	1
0	1	0
1	0	0
1	1	0



Representação de Funções

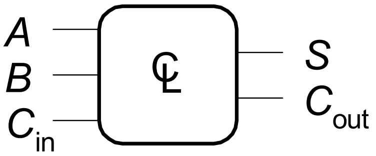
- Uma função lógica pode ser representada, como:
  - Uma tabela de verdade;
  - Uma equação algébrica
  - Um circuito lógico
- Como as três formas representam a mesma função têm que ser equivalentes.
- Vamos estudar:
  - Como passar da Tabela de Verdade para a equação algébrica
  - Como simplificar a equação algébrica
  - Como representar o circuito Lógico

Equação Booleana

- Uma equação Booleana permite fazer a especificação funcional da saída em função das entradas:

$$S = F(A,B,C_{in})$$

$$C_{out}=F(A,B,C_{in})$$



$$S = A \oplus B \oplus C_{in}$$

$$C_{out} = AB + AC_{in} + BC_{in}$$

Álgebra de Boole: Definições

- Complemento:
  - $\bar{A}, \bar{B}, \bar{C}$
- Literal: variável ou o seu complemento
  - $A, \bar{B}, C$
- Implicante: Produto de Literais
  - $AB, \bar{B}C, AB\bar{C}$
- Mintermo: Produto que inclui todas as variáveis
  - $ABC, A\bar{B}C, AB\bar{C}$
- Maxtermo: Soma que inclui todas as variáveis
  - $(A+B+C), (A+\bar{B}+C), (A+B+\bar{C})$

Precedências

- **Precedência** dos operadores:

$$Y = A + BC$$

- Lê-se

$$Y = (A + B)C \text{ ou } Y = A + (BC)?$$

- A prioridade mais elevada é do operador NOT, seguindo-se o AND e depois o OR.
  - Logo neste caso lê-se $Y = A + (BC)$

Álgebra de Boole: Definições e Teoremas

- A tabela de verdade que descreve um problema lógico pode **sempre** ser descrita como:
  - Uma Soma de Produtos (SoP)
  - Um Produto de Somas (PoS)
- A descrição de uma função Booleana como uma Soma de Produtos (mintermos) corresponde à **primeira forma canónica**.
- A descrição como um Produto de Somas (maxtermos) corresponde à **segunda forma canónica** de representação.

Soma de Produtos (SoP)

- Uma tabela de verdade de N entradas tem 2^N linhas que descrevem o valor da saída para cada combinação diferente das entradas
- Cada linha corresponde a um mintermo, i.e., um produto (AND) de literais
- Cada mintermo assume o valor 1 na sua linha e só nessa linha
- Somam-se (OR) os mintermos que assumem o valor 1.

Álgebra de Boole: Definições e Teoremas

Soma de Produtos: Exemplo

- Escreva a forma Soma de Produtos de Y que é função de duas entradas, A e B, descrita pela seguinte tabela verdade:

A	*B*	*Y*	minterm	minterm name
0	0	0	$\overline{A}\ \overline{B}$	m_0
0	1	1	$\overline{A}\ B$	m_1
1	0	0	$A\ \overline{B}$	m_2
1	1	1	$A\ B$	m_3

$$Y = \mathrm{F}(A, B) = \overline{\mathrm{A}}\mathrm{B} + \mathrm{AB} = \Sigma(1, 3)$$

Produtos de Somas (PoS)

- Cada linha corresponde a um maxtermo, i.e., uma soma (OR) de literais
- Cada maxtermo assume o valor 0 na sua linha e só nessa linha
- Multiplicam-se (AND) os maxtermos que assumem o valor 0.

A	*B*	*Y*	maxterm	maxterm name
0	0	0	$A + B$	M_0
0	1	1	$A + \overline{B}$	M_1
1	0	0	$\overline{A} + B$	M_2
1	1	1	$\overline{A} + \overline{B}$	M_3

$$Y = F(A, B) = (A + B)(\overline{A} + B) = \Pi(0, 2)$$

Exemplo: O Almoço do Carlos

- O Carlos foi comer à cantina. Ele não come ($\bar{C}$) se:
  - A cantina não estiver aberta ($\bar{A}$)
  - A cantina apenas servir brócolos (B)
- Escreva a tabela de verdade que determina as condições em que o Carlos almoça (C)

A	*B*	*C*
0	0	
0	1	
1	0	
1	1	

- Escreva C na forma SoP e PoS.

Produto de Somas e Soma de Produtos

- Qualquer tabela de verdade pode ser escrita usando PoS ou SoP
- Mas…
- no caso geral a descrição pode ser demasiado complexa para ser útil.
- Vamos estudar álgebra de Boole que nos vai permitir simplificar as expressões:
  - Os fundamentos são os mesmos da Álgebra, mas mais simples (apenas temos dois valores possíveis)
- Todos os Axiomas e teoremas tem um dual, i.e., continuam a verificar-se se trocarmos os 0's por 1's e os AND's por OR's e vice-versa.

Axiomas

- A1: Valor Binário

  B=0 se B$\neq$1
- A2: NOT

  $\overline{0} = 1$
- A3,A4 e A5: AND

  0.0 = 0

  1.1=1

  0.1=1.0 = 0

- A1': Valor Binário

  B=1 se B$\neq$0
- A2: NOT

  $\overline{1} = 0$
- A3,A4 e A5: OR

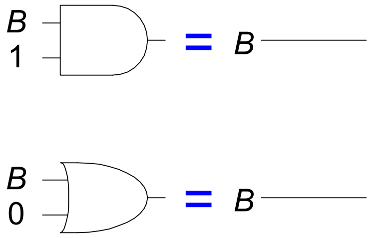
  1+1 = 1

  0+0 = 0

  1+0=0+1 = 1

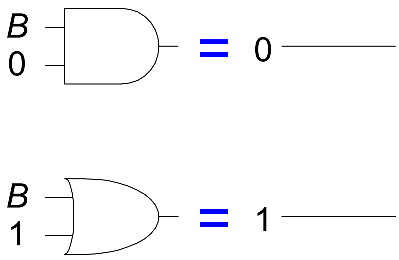
Teorema 1: Identidade

- B.1 = B
- B+0 = B



Teorema 2: Elemento Absorvente

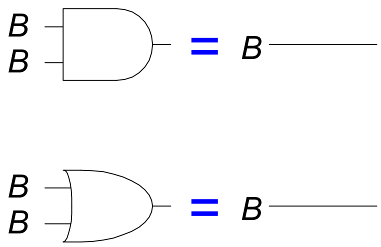
- B.0 = 0
- B+1 = 1



Álgebra de Boole: Definições e Teoremas

Teorema 3: Idempotência

- B.B = B
- B+B = B

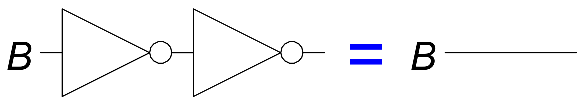


09/03/2018 PML – IAC - 2018 39

Álgebra de Boole: Definições e Teoremas

Teorema 4: Involução

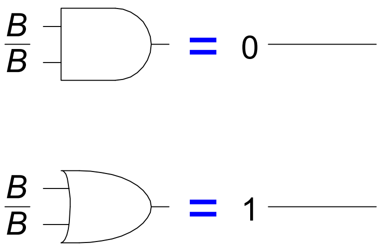
- $\overline{\overline{B}} = B$



09/03/2018 PML – IAC - 2018 40

Teorema 5: Complementaridade

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



Sumário dos Teoremas

	Theorem		Dual	Name
T1	$B \bullet 1 = B$	T1′	$B + 0 = B$	Identity
T2	$B \bullet 0 = 0$	T2′	$B + 1 = 1$	Null Element
T3	$B \bullet B = B$	T3′	$B + B = B$	Idempotency
T4		$\bar{\bar{B}} = B$		Involution
T5	$B \bullet \bar{B} = 0$	T5′	$B + \bar{B} = 1$	Complements

Teoremas de várias variáveis

- Teorema 6: Comutatividade
  - B.C = C.B
  - B+C = C+B
- Teorema 7: Associatividade
  - (B.C).D = B.(C.D)
  - (B+C)+D = B+(C+D)
- Teorema 8: Distributividade
  - (B.C)+(B.D) = B.(C+D)
  - (B+C).(B+D) = B+(C.D)*

Teoremas de várias variáveis

- Teorema 9: Absorção (*Covering*)
  - B.(B+C) = B
  - B+(B.C) = B
- Teorema 10: Adjacência (Combining)
  - (B.C)+(B.$\overline{C}$) = B
  - (B+C).(B+$\overline{C}$) = B
- Teorema 11: Consenso
  - (B.C)+($\overline{B}$.D)+(C.D) = (B.C)+($\overline{B}$.D)
  - (B+C).($\overline{B}$+D).(C+D) = (B+C).($\overline{B}$+D)

Demonstração dos Teoremas

- Estes teoremas podem ser demonstrados?
- Uma forma é usar a indução perfeita que é simples para teoremas com um número finito de variáveis.
  - Consiste em demonstrar na tabela de verdade que o teorema se verifica para todos os valores das variáveis.
- Exemplo teorema da adjacência (*Combining*)

B	C	$(B.C)+(B.\bar{C})$	B
0	0	0	0
0	1	0	0
1	0	1	1
1	1	1	1

Simplificação de Equações Lógicas

- Exemplo 1:

Y = A(AB + ABC)

=A(AB(1+C)) (T8 - Distributividade)

=A(AB.(1)) (T2' - El. Absorvente)

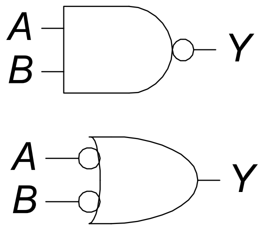
= A(AB) (T1 - Identidade)

= (AA)B (T7 - Associatividade)

= AB (T3 – Idempotência)

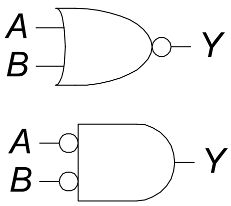
Teorema 12: Leis de DeMorgan

- A negação do produto é a soma das negações:
$$\overline{B_0 . B_1 . B_2 \dots} = \overline{B_0} + \overline{B_1} + \overline{B_2} + \cdots$$
- Corolário:
  - Uma porta NAND equivale a uma porta OR com entradas negadas;
- $Y = \overline{A.B} = \bar{A} + \bar{B}$



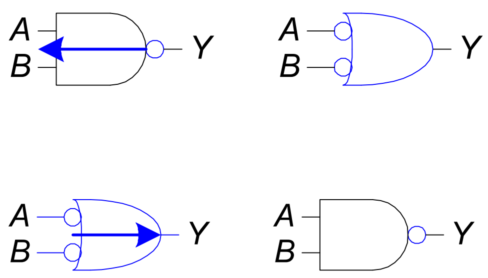
Teorema 12: Leis de DeMorgan

- A negação da soma é o produto das negações
$$\overline{B_0 + B_1 + B_2 \dots} = \overline{B_0} . \overline{B_1} . \overline{B_2} . \dots$$
- Corolário:
  - Uma porta NOR equivale a uma porta AND com entradas negadas.
- $Y = \overline{A + B} = \bar{A} . \bar{B}$



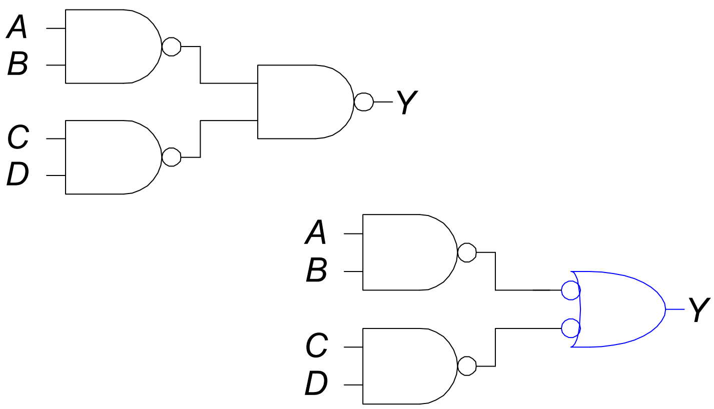
"Bubble Pushing"

- Troca-se a gate OR/AND por AND/OR
- E empurram-se as bolhas



"Bubble Pushing"

- Qual é a expressão Booleana deste circuito:



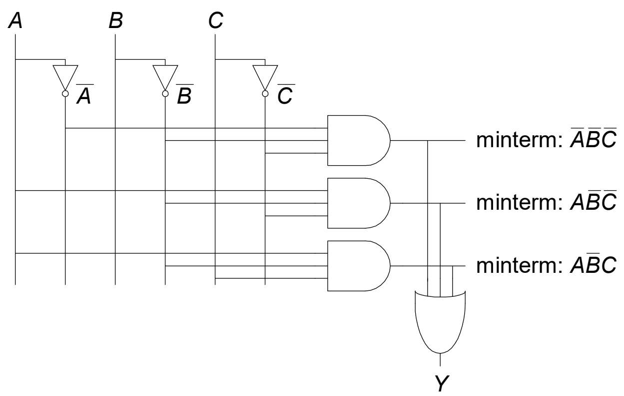
$$Y = AB + CD$$

Passar da lógica para os circuitos

- A implementação direta da forma Soma de Produtos origina uma implementação lógica a dois níveis:
  - ANDs seguidos de ORs
- Convenção de desenho:
  - Entradas à esquerda (ou em cima)
  - Saídas à direita (ou em baixo)
  - Usam-se linhas retas
  - O fluxo das *Gates* é da esquerda para a direita

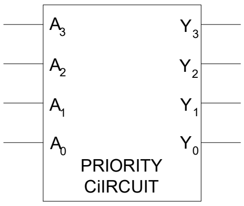
Passar da lógica para os circuitos

- Exemplo:
  - $Y = \bar{A}\bar{B}\bar{C}$+A$\bar{B}\bar{C}$+A$\bar{B}C$



Circuitos com múltiplas saídas

- Exemplo:
- Circuito de Prioridades
  - Tem 4 entradas e 4 saídas.
  - Fica ativa a saída correspondente à entrada ativa mais significativa.



A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	0	1	1	0	0	1	0
0	1	0	0	0	1	0	0
0	1	0	1	0	1	0	0
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	0
1	0	0	0	1	0	0	0
1	0	0	1	1	0	0	0
1	0	1	0	1	0	0	0
1	0	1	1	1	0	0	0
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	0
1	1	1	0	1	0	0	0
1	1	1	1	1	0	0	0

Don’t Cares

- Podemos simplificar a tabela de verdade usando o símbolo X para representar os casos em que a saída é independente de alguma(s) das entradas.

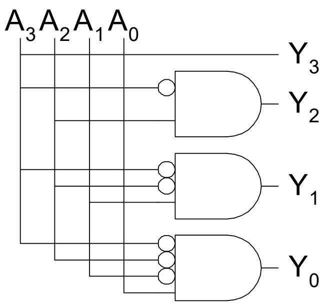
A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	0
0	0	1	1	0	0	1	0
0	1	0	0	0	1	0	0
0	1	0	1	0	1	0	0
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	0
1	0	0	0	1	0	0	0
1	0	0	1	1	0	0	0
1	0	1	0	1	0	0	0
1	0	1	1	1	0	0	0
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	0
1	1	1	0	1	0	0	0
1	1	1	1	1	0	0	0

A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	X	0	0	1	0
0	1	X	X	0	1	0	0
1	X	X	X	1	0	0	0

Circuitos com múltiplas saídas

- Circuito de Prioridades: Implementação

A_3	A_2	A_1	A_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	X	0	0	1	0
0	1	X	X	0	1	0	0
1	X	X	X	1	0	0	0



Conjunto completo de operadores

- Define-se **conjunto completo de operadores** o conjunto de operadores que permite a implementação de qualquer função booleana.
- São Conjuntos Completos de Operadores:
  - {AND, OR, NOT}
  - {AND, NOT}
  - {OR, NOT}
  - {NAND}
  - {NOR}
- Um conjunto completo com apenas um tipo de portas pode facilitar o trabalho de quem projeta os circuitos.

Operadores NAND e NOR

- Para escrever uma função apenas com operadores NAND coloca-se na forma de SoP e de seguida aplica-se o teorema da involução (dupla negação) e as leis de DeMorgan.
- Para escrever só com operadores NOR começa-se com a forma PoS.
- Exemplo:
  - $x + (y.\bar{z}) = \overline{\overline{x + (y.\bar{z})}} = \overline{\bar{x}.\overline{y.\bar{z}}}$
  - $x + (y.\bar{z}) = \overline{\overline{(x + y).(x + \bar{z})}} = \overline{\overline{x + y} + \overline{x + \bar{z}}}$

Outras formas canónicas de representação

- Formalizando o caso geral da representação canónica de funções booleanas temos:
- 1ª forma canónica: SoP AND-OR

$$f(x_0, x_1, \dots, x_{n-1}) = \sum_{i=0}^{2^n-1} f_i . m_i$$

- 2ª forma canónica: PoS OR-AND

$$f(x_0, x_1, \dots, x_{n-1}) = \prod_{i=0}^{2^n-1} (f_i + M_i)$$

- E ainda …

Outras formas canónicas de representação

- 3ª forma canónica: SoP NAND-NAND

$$f(x_0, x_1, \dots, x_{n-1}) = \overline{\prod_{i=0}^{2^n-1} \overline{f_i . m_i}}$$

- 4ª forma canónica: PoS NOR-NOR

$$f(x_0, x_1, \dots, x_{n-1}) = \overline{\sum_{i=0}^{2^n-1} \overline{f_i + M_i}}$$

Formas canónicas de representação

- Exercício:
- Determinar as formas canónicas da função:
- $f(x, y, z) = x.y + \bar{z}$
  - Sugestão: Escrevendo a tabela de verdade obtêm-se facilmente a 1ª e a 2ª formas. As formas NAND-NAND e NOR-NOR são obtidas a partir destas.

Índice

- Lógica binária.
  - As operações lógicas básicas.
  - Portas Lógicas.
- Álgebra de Boole.
  - Axiomas e teoremas.
  - Soma de Produtos e Produto de Somas.
  - Minimização de funções booleanas.

Mapas de Karnaugh

- As expressões Booleanas podem ser minimizadas combinando algebricamente os termos.
- Os mapas de Karnaugh são um método de determinar graficamente a minimização das equações.
- Começamos por representar a tabela de verdade num Mapa e desenhamos "círculos" sempre que houver 1's em quadrados adjacentes.
  - Entre cada linha (coluna) apenas pode variar um bit
- Na expressão final usamos apenas os literais que são invariantes em cada "círculo".

Mapas de Karnaugh

- Exemplo com 3 entradas

Truth Table

A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

K-Map

Y — C \ AB	00	01	11	10
0	0	1	0	0
1	0	1	1	0

$$Y = \bar{A}B + BC$$

Mapas de Karnaugh: Regras

- Cada 1 no mapa tem que estar dentro de pelo menos um “círculo”
- O número de quadrados cobertos por um círculo tem que ser uma potência de 2, em cada direção
- O “círculo“ pode fechar-se pelas arestas do mapa
- Um *don’t care* (X) inclui-se num “círculo” se ajudar a simplificar a equação

Mapas de Karnaugh

- Exemplo com 4 entradas:

A	B	C	D	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

Y CD \ AB	00	01	11	10
00	1	0	0	1
01	0	1	0	1
11	1	1	0	0
10	1	1	0	1

$$Y = \bar{A}C + \bar{B}\bar{D} + \bar{A}BD + A\bar{B}\bar{C}$$

Mapas de Karnaugh com *don't care*

- Exemplo com 4 entradas:

A	B	C	D	Y
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	X
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	X
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X

Y CD \ AB	00	01	11	10
00	1	0	X	1
01	0	X	X	1
11	1	1	X	X
10	1	1	X	X

$$Y = A + \bar{B}\bar{D} + C$$