Pontifícia Universidade Católica de Minas Gerais
Instituto de Ciências Exatas e Informática – ICEI
Arquitetura de Computadores I

ARQ1 _ Aula_04
Tema: Introdução à Álgebra de Boole

Atividade: Álgebra de Boole

Em Matemática, chama-se ***proposição*** ao enunciado de uma verdade que se quer demonstrar, ou como usaremos: uma sentença que pode ser falsa (0), ou verdadeira (1), mas nunca ambos ao mesmo tempo.

A conjunção é uma relação entre sentenças que estabelece um resultado **verdadeiro** (1) quando associadas duas proposições (p e q), ambas **verdadeiras** (iguais a 1). Basta uma delas ser **falsa** (0), para que a conjunção (s) também seja **falsa** (0).

A porta **AND** (E) é um componente de circuito lógico que implementa essa relação; pode ter duas (p, q), ou mais entradas, e a saída (s) assumirá o valor 1 (**verdadeiro**) se, e somente se, todas as entradas forem iguais a 1 (**verdadeiras**); caso uma, ou mais entradas forem iguais a 0 (**falso**), a saída terá valor igual a 0 (**falso**).

A disjunção é uma relação entre sentenças que estabelece um resultado **falso** (0) quando duas proposições (p e q) forem **falsas** (0). Basta uma delas ser **verdadeira** (1), para que a disjunção também seja **verdadeira** (1).

A porta **OR** (OU) é um componente de circuito lógico que implementa essa relação; pode ter duas (p, q), ou mais entradas, e a saída (s) assumirá o valor 0 (**falso**) se, e somente se, todas as entradas forem iguais a 0 (**falso**); caso uma, ou mais entradas forem iguais a 1 (**verdadeiro**), a saída terá valor 1 (**verdadeiro**).

A negação determina que se uma proposição (p) for **falsa** (0), seu resultado será **verdadeiro** (1), ou vice-versa.

A porta **NOT** (NÃO) é um componente de circuito lógico que implementa essa relação, também chamada de **INVERTER** (INVERSOR), só possui uma entrada (p), e a saída assumirá o valor 1 (**verdadeiro**), se a entrada for igual a 0 (**falso**); senão, a saída terá valor 0 (**falso**), se a entrada for igual a 1 (**verdadeiro**).

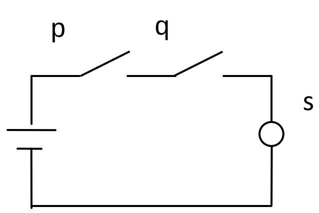
- Analogias com circuitos elétricos

O primeiro circuito a seguir (conjunção) determina que se duas chaves (p e q) forem fechadas (1), o resultado (s) será o de um circuito fechado com uma lâmpada acesa (1), por exemplo; basta que uma delas seja aberta (0), para que o circuito se abra, e a lâmpada apague (0). O circuito poderá ter duas (p, q), ou mais chaves, em série que a saída (s) terá o mesmo resultado (1) se, e somente se, todas as chaves forem fechadas (1); caso uma, ou mais chaves forem abertas (0), o resultado será um circuito aberto com a lâmpada apagada (0).

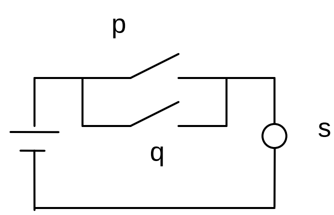
O segundo circuito a seguir (disjunção) determina que se duas chaves (p e q) forem abertas (0), o resultado (s) será o de um circuito aberto com uma lâmpada apagada (0), por exemplo; basta que uma delas seja fechada (1), para que o circuito se feche. O circuito poderá ter duas (p, q), ou mais chaves, em paralelo que a saída (s) terá o mesmo resultado (0), se, e somente se, todas as entradas forem abertas (0); caso uma, ou mais chaves forem fechadas (1), o resultado será um circuito fechado coma lâmpada acesa (1).

O terceiro circuito a seguir (negação) determina que se uma chave (p) for acionada (1), o resultado (s) será o de um circuito aberto com uma lâmpada apagada (0); caso contrário, o circuito permanecerá fechado, e a lâmpada se manterá acesa (1).

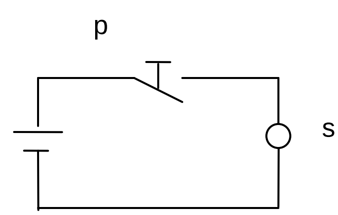
- Representações por circuitos



Circuito série (AND)
(Conjunção)



Circuito paralelo (OR)
(Disjunção)



Curto-circuito (NOT)
(Negação)

- Representações de relações lógicas

Notações

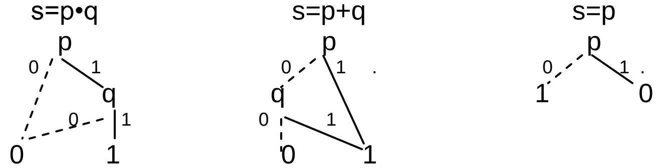
Conjunção (E) (p e q)	Disjunção (OU) (p ou q)	Negação (NÃO) (não p)
$p \wedge q$	$p \vee q$	$\neg p$
p • q = p q	p + q	/p = $\bar{p}$ = p'
p & q	p | q	~p
p && q	p || q	**!** p

Tabela-verdade

Conjunção (E)	Disjunção (OU)	Negação (NÃO)
p q s	p q s	p s
0 • 0 = 0	0 + 0 = 0	
0 • 1 = 0	0 + 1 = 1	0' = 1
1 • 0 = 0	1 + 0 = 1	1' = 0
1 • 1 = 1	1 + 1 = 1	

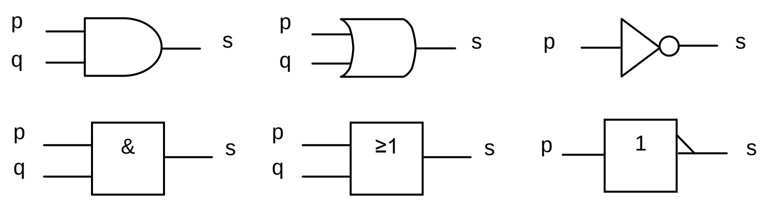
Diagrama de Decisão Binária (BDD)

s=p•q s=p+q s=p

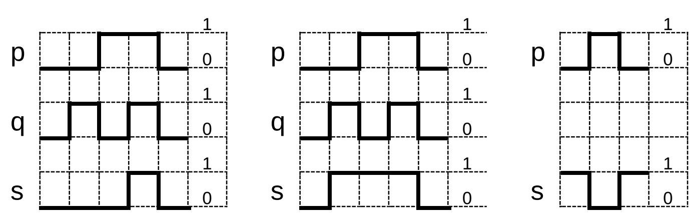


Portas Lógicas

s = AND (p, q) s = OR (p, q) s = NOT (p)



Diagramas de tempo para as portas lógicas



- Prioridade de conectivos

Estabelece-se que a ordem de avaliação de uma expressão, envolvendo conectivos lógicos, será da esquerda para a direita, respeitando-se as prioridades dos conectivos na ordem mostrada abaixo, sendo a primeira a mais alta quando aplicada imediatamente a um valor.

NÃO
E
OU

Pode-se mudar a ordem de avaliação por meio de parênteses.

Exemplo:

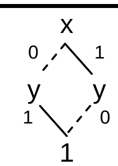
Considere a expressão lógica: (! x && y) || (x && ! y)
de forma mais simples como (x' • y) + (x • y')
A sua avaliação será feita na seguinte ordem de prioridade:

- negação de (x) : x'
- conjunção com (y) : x' • y
- negação de (y) : y'
- conjunção com (x) : x • y'
- disjunção das conjunções : (x' • y) + (x • y')

A expressão poderá ser representada nas formas tabular (***tabela-verdade***) ou por BDD (*Binary Decision Diagram*):

x y	(x' • y)	(x • y')	(x'•y)+(x•y')
0 0	0	0	0
0 1	1	0	1
1 0	0	1	1
1 1	0	0	0

BDD



Resumidamente as relações em uma tabela também poderão ser indicadas

- pela disjunção (+) das conjunções iguais a 1 (ou mintermos)

x y	(x' • y)	(x • y')	(x'•y)+(x•y')	mintermos (=1)
0 0	0	0	0	m0 = x' • y' = 0 .
0 1	1	0	1	m1 = x' • y = 1 ←
1 0	0	1	1	m2 = x • y' = 2 ←
1 1	0	0	0	m3 = x • y = 3 .

f (x, y) = (x'•y)+(x•y') = m1 + m2 = $\sum$m(1,2) = SoP(1,2)

- pela conjunção (•) das disjunções iguais a 0 (ou MAXTERMOS).

X Y	(X+Y')	(X'+Y)	(X+Y')•(X'+Y)	MAXTERMOS (=0)
0 0	0	0	0	m0 = X + Y = 0 ←
0 1	1	0	1	m1 = X + Y' = 1 .
1 0	0	1	1	m2 = X' + Y = 2 .
1 1	0	0	0	m3 = X' + Y' = 3 ←

F (X, Y) = (X+Y') • (X'+Y) = M1 • M2 = ΠM(0,3) = PoS(0,3)

Principais relações da álgebra de Boole

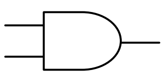
s=(p&q)	AND	p/q
0	1	
0	0	0
0	1	1

s=(p | q)	OR
0	1
0	1
1	1

s=(p^q)	XOR	p/q
0	1	
0	1	0
1	0	1

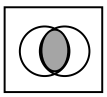
BUFFER
s = p
0
1

Conjunção

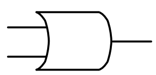
p
q

s

AND

s = p • q

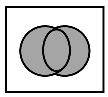


Disjunção

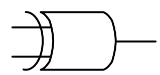
p
q

s

OR

s = p + q

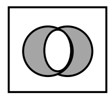


Disjunção Exclusiva

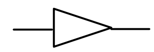
p
q

s

XOR

$s = p \oplus q$

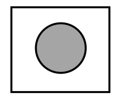


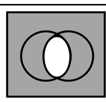
Cópia

p

s

BUFFER

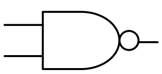
s = p



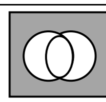


s = (p • q)'

NAND

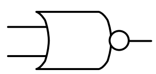
p
q

s

Alternative Denial

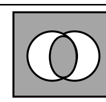


s = (p + q)'

NOR

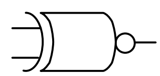
p
q

s

Joint Denial

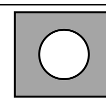


s = (p + q)'

XNOR

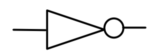
p
q

s

Equivalência



$s = p' = \overline{P}$

NOT

p

s

Negação

1	1	0
1	0	1
0	1	p\q
s=~(p&q)	NAND	

1	0
0	0
0	1
s=~(p|q)	NOR

1	0	0
0	1	1
0	1	p\q
s=~(p^q)	XNOR	

1
0
s=~p
NOT

Resumo

	m	M	**p**	**q**	**AND** **p && q**	**OR** **p || q**	**XOR** **p ^ q**	**XNOR** **! (p ^ q)**	**NOR** **! (p || q)**	**NAND** **! (p && q)**
0	p' • q'	P+Q	0	0	0	0	0	1	1	1
1	p' • q	P+Q'	0	1	0	1	1	0	0	1
2	p • q'	P'+Q	1	0	0	1	1	0	0	1
3	p • q	P'+Q'	1	1	1	1	0	1	0	0
mintermos		SoP	(+)	[=1]	3	1,2,3	1,2	0,3	0	0,1,2
MAXTERMOS		PoS	(•)	[=0]	0,1,2	0	0,3	1,2	1,2,3	3

Principais propriedades da Álgebra de Boole

Idempotência	Comutativa	Associativa
$p + p = p$	$p + q = q + p$	$(p + q) + r = p+(q + r)$
$p \bullet p = p$	$p \bullet q = q \bullet p$	$(p \bullet q) \bullet r = p \bullet (q \bullet r)$
Identidade	**De Morgan**	**Distributiva**
$p + 0 = p \quad p \bullet 0 = 0$	$\overline{(p+q)} = \bar{p} \bullet \bar{q}$	$p + (q \bullet r)=(p + q) \bullet (p + r)$
$p + 1 = 1 \quad p \bullet 1 = p$	$\overline{(p \cdot q)} = \bar{p} + \bar{q}$	$p \bullet (q+r)=(p \bullet q) + (p \bullet r)$
Complementar	**Absorção**	**Consenso**
$p + \bar{p} = 1$ (tautologia)	$p + (\bar{p} \bullet q)=(p + q)$	$(p \bullet q)+(\bar{p} \bullet r)+(q \bullet r)$ $= (p \bullet q) + (\bar{p} \bullet r)$
$p \bullet \bar{p} = 0$ (contradição)	$\bar{p} + (p \bullet q)=(\bar{p} + q)$	$(p+q) \bullet (\bar{p}+r) \bullet (q+r)$ $= (p+q) \bullet (\bar{p}+r)$
$\bar{\bar{p}} = p$ (dupla negação)	$p + (p \bullet q) = p$	

Principais propriedades da álgebra com XOR.

Básicas	Identidade	Complementar
$p \oplus p = 0$	$p \oplus 0 = p$	$\bar{p} \oplus \bar{q} = p \oplus q$
$p \oplus \bar{p} = 1$	$p \oplus 1 = \bar{p}$	$\overline{(p \oplus q)} = \bar{p} \oplus q = p \oplus \bar{q}$
Associativa		**Comutativa**
$(p \oplus q) \oplus r = p \oplus (q \oplus r)$		$p \oplus q = q \oplus p$
Disjunção	**Distributiva**	**Transposição**
se: $p = q \oplus r$ e $q \bullet r=0$ então: $p = q + r$	$p \bullet (q \oplus r) = (p \bullet q) \oplus (p \bullet r)$	se: $p = q \oplus r$ então: $q = p \oplus r$ e $r = p \oplus q$

Tabela-verdade

Expressões lógicas podem ser expressas na forma tabular (tabela-verdade):

Exemplo:
Avaliar a expressão: x' • y + x • y'
considerando a ordem de prioridades entre negação, conjunção e disjunção.

x y	x' • y	x • y'	x'• y + x • y'
0 0	0	0	0
0 1	1	1	1
1 0	0	0	1
1 1	0	0	0

A descrição equivalente em Verilog será

```
// ---------------------
// TRUTH TABLE
// Nome: xxx yyy zzz
// Matricula: 999999
// ---------------------

// ---------------------
// -- expression
// ---------------------

module fxy (output s,
            input  x, y);
 assign s = ~x & y | x & ~y;

endmodule // fxy
```

Uma função lógica também pode ser descrita pela soma de produtos (mintermos) ou SoP (disjunção das conjunções dos termos na tabela onde a função for igual a 1).

#mintermo	mintermo	x y	f(x,y)	
0	x'• y'	0 0	0	
1	x'• y	0 1	1	←
2	x • y'	1 0	1	←
3	x • y	1 1	0	

$$f(x,y) = (x' \cdot y) + (x \cdot y') = \sum m(1,\ 2)$$

A descrição equivalente em Verilog será

```
// ---------------------
// -- SoP
// ---------------------

module SoP (output s,
                      input  x, y);
                                 // mintermos
 assign s = ~x &   y   // 1
              |   x & ~y;  // 2
endmodule // SoP
```

Uma função lógica pode ser descrita pelo produto de somas (MAXTERMOS) ou PoS, (conjunção das disjunções dos termos na tabela onde a função for igual a 0), cujo resultado é equivalente à soma de produtos complementar

#MAXTERMOS	MAXTERMOS	X Y	F(X,Y)	
0	X +Y	0 0	0	←
1	X +Y'	0 1	1	
2	X'+Y	1 0	1	
3	X'+Y'	1 1	0	←

$$F(X,Y) = (X + Y) \cdot (X' + Y') = \prod M(0,\ 3)$$

A descrição equivalente em Verilog será

```
// ---------------------
// -- PoS
// ---------------------

module PoS (output S,
                      input  X, Y);
                                   // MAXTERMOS
 assign S = (  X |  Y  )  // 0
              & (  X | ~Y ); // 3

endmodule // PoS
```

cujo módulo com os conjuntos de testes em Verilog poderá ser

```
// ---------------------
// -- test_module
// ---------------------

module test_module;
 reg   x, y;
 wire  s1, s2, s3;
         // instancias
 fxy   FXY1  (s1, x, y);
 SoP SOP1  (s2, x, y);
 PoS POS1  (s3, x, y);

         // valores iniciais
 initial begin: start
     x=1'bx; y=1'bx;   // indefinidos
 end
         // parte principal
 initial begin: main
   // identificacao
     $display("Exemplo- xxx yyy zzz - 999999");
     $display("Test boolean expression");
     $display("\nx'&y+x&y' = s\n");
   // monitoramento
     $display("x  y   = s1 s2 s3");
     $monitor("%2b %2b  = %2b %2b %2b", x, y, s1, s2, s3);
   // sinalizacao
  #1  x=0; y=0;
  #1  x=0; y=1;
  #1  x=1; y=0;
  #1  x=1; y=1;
end

endmodule // test_module
```

Preparação

Vídeos recomendados

Como preparação para o início das atividades,
recomenda-se assistir os seguintes vídeos:

http://www.youtube.com/watch?v=Tb1qLGR2hvU
http://www.youtube.com/watch?v=UrA-miNZ6ag
http://www.youtube.com/watch?v=wAqlu7M4xvA

Exercícios:

01.) Construir a tabela-verdade para as proposições
e verificar pelas respectivas tabelas-verdades implementadas em Verilog:
Exemplo:

$$\overline{x} + (\overline{y} \cdot \overline{z})$$

#mintermos	mintermos	x y z	x'	y'	z'	y'•z'	x'+(y'•z')
0	x'•y'•z'	0 0 0	1	1	1	1	1
1	x'•y'•z	0 0 1	1	1	0	0	1
2	x'•y•z'	0 1 0	1	0	1	0	1
3	x'•y •z	0 1 1	1	0	0	0	1
4	x•y'•z'	1 0 0	0	1	1	1	1
5	x•y'•z	1 0 1	0	1	0	0	0
6	x•y•z'	1 1 0	0	0	1	0	0
7	x•y•z	1 1 1	0	0	0	0	0

SoP (0,1,2,3,4)

```
module fxyz (output s,
                    input  x, y, z);
 assign s = ~x | (~y & ~z);

endmodule // fxyz
```

a.) $x \cdot \overline{(\overline{y} + z)}$
b.) $(x + \overline{y}) \cdot \overline{z}$
c.) $\overline{(\overline{x} + y)} \cdot z$
d.) $\overline{(x \cdot y)} + \overline{z}$
e.) $(\overline{x} + \overline{y}) \cdot (y + z)$

02.) Simplificar as expressões abaixo pelas propriedades da álgebra de Boole
e verificar pelas respectivas tabelas-verdades implementadas em Verilog:
Exemplo:

$$(\bar{x}+\bar{y})\cdot(\bar{x}+\bar{z})$$

= x'+(y'•z') (propriedade distributiva)

```
module fxyz (output s1, output s2
                    input  x, y, z);
 assign s1 = (~x | ~y ) & (~x | ~z);
 assign s2 = ~x | (~y & ~z);
endmodule // fxyz
```

a.) $x \cdot \overline{x + \bar{y}}$

b.) $(x + y) + (y \cdot \bar{x})$

c.) $\overline{(x \cdot y)} \cdot (\bar{y} + x)$

d.) $\overline{(\bar{x} \cdot \bar{y})} + (\bar{x} + \bar{y})$

e.) $(y + x) \cdot (\bar{y} + \bar{x})$

03.) Montar as tabelas-verdades expressas pelas somas de produtos abaixo
e verificar pelas respectivas tabelas-verdades implementadas em Verilog:
Exemplo:

$$f(x,y,z) = \sum m(0, 1, 2, 3, 4) = SoP\ (0,1,2,3,4) = 1$$

```
module SoP (output s, input  x, y);          // mintermos
 //       m    0                   1                  2                  3               4
 assign s = (~x&~y~&~z) | (~x&~y&z) | (~x&y&~z) | (~x&y&z) | (x&~y&~z);

endmodule // SoP
```

x y z	mintermos	SoP (0,1,2,3,4)
0 0 0	x'•y'•z' = m0	1
0 0 1	x'•y'•z = m1	1
0 1 0	x'•y •z' = m2	1
0 1 1	x'•y •z = m3	1
1 0 0	x•y'•z' = m4	1
1 0 1	x•y'•z	0
1 1 0	x•y•z'	0
1 1 1	x•y•z	0

a) f(x,y,z) = $\sum m(1, 2, 4, 5)$

b) f(x,y,z) = $\sum m(0, 2, 3, 6)$

c) f(x,y,w,z) = $\sum m(0, 1, 3, 5, 7, 9, 10, 11)$

d) f(x,y,w,z) = $\sum m(0, 2, 4, 6, 8, 10, 13)$

e) f(x,y,w,z) = $\sum m(0, 1, 2, 3, 7, 13, 15)$

04.) Montar as expressões PoS equivalentes aos produtos das somas abaixo
e verificar pelas respectivas tabelas-verdades implementadas em Verilog:

Exemplo:

$F (X,Y,Z) = \prod M(5, 6, 7) = PoS (5,6,7) = 0$

```
module PoS (output S, input  X, Y);        // MAXTERMOS
//          M     5                  6                    7
  assign S = (~X| Y|~Z)  & (~X|~Y| Z) & (~X|~Y|~Z);

endmodule // PoS
```

x y z	MAXTERMOS	PoS (5,6,7)
0 0 0	X+Y+Z	1
0 0 1	X+Y+Z'	1
0 1 0	X+Y'+Z	1
0 1 1	X+Y'+Z'	1
1 0 0	X'+Y+Z	1
1 0 1	X'+Y+Z' = M5	**0**
1 1 0	X'+Y'+Z = M6	**0**
1 1 1	X'+Y'+Z' = M7	**0**

a) $F(X,Y,Z) = \prod M(1, 2, 4, 7)$

b) $F(X,Y,Z) = \prod M(0, 1, 2, 43\ 5)$

c) $F(X,Y,W,Z) = \prod M(0, 1, 2, 5, 7\ 9, 10, 11)$

d) $F(X,Y,W,Z) = \prod M(0, 2, 4, 6, 7, 9, 13)$

e) $F(X,Y,W,Z) = \prod M(0, 1, 2, 3, 7, 14, 15)$

05.) Identificar as expressões SoP e PoS equivalentes às tabelas abaixo
e verificar pelas respectivas tabelas-verdades implementadas em Verilog:

a.)

n	x y	f(x,y)
0	0 0	1
1	0 1	1
2	1 0	0
3	1 1	1

SoP( ) = ____________

PoS( ) = ____________

b.)

n	x y	f(x,y)
0	0 0	1
1	0 1	0
2	1 0	1
3	1 1	0

SoP( ) = ____________

PoS( ) = ____________

c.)

n	x y z	f(x,y,z)
0	0 0 0	1
1	0 0 1	0
2	0 1 0	0
3	0 1 1	1
4	1 0 0	1
5	1 0 1	1
6	1 1 0	1
7	1 1 1	0

SoP( ) = ____________

PoS( ) = ____________

d.)

n	x y z	f(x,y,z)
0	0 0 0	1
1	0 0 1	0
2	0 1 0	1
3	0 1 1	1
4	1 0 0	1
5	1 0 1	0
6	1 1 0	1
7	1 1 1	0

SoP( ) = ____________

PoS( ) = ____________

e.)

n	x y w z	f(x,y,w,z)
0	0 0 0 0	1
1	0 0 0 1	1
2	0 0 1 0	0
3	0 0 1 1	1
4	0 1 0 0	1
5	0 1 0 1	0
6	0 1 1 0	1
7	0 1 1 1	0
8	1 0 0 0	1
9	1 0 0 1	0
10	1 0 1 0	1
11	1 0 1 1	1
12	1 1 0 0	0
13	1 1 0 1	0
14	1 1 1 0	1
15	1 1 1 1	1

SoP( ) = ____________

PoS( ) = ____________