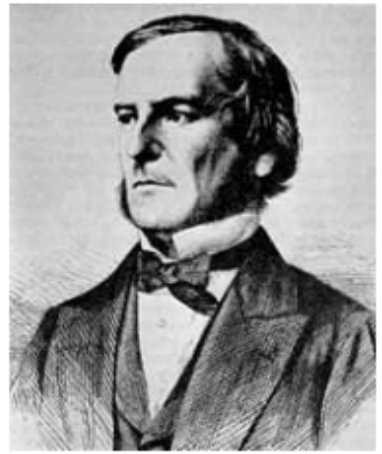
Funções & Portas lógicas

Uniube – campus Uberlândia

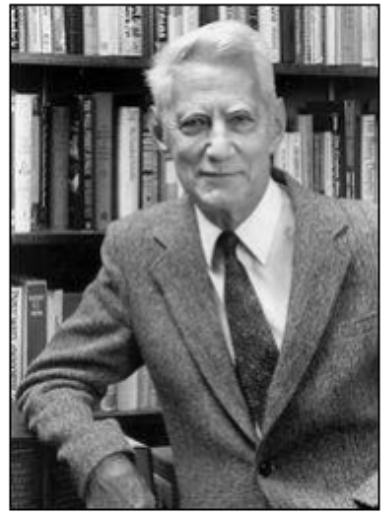
Prof. Ms Mário Peixoto

Histórico

- Em meados do século XIX o matemático inglês George **Boole** desenvolveu um sistema matemático de análise lógica
- Em meados do século XX, o americano Claude Elwood **Shannon** sugeriu que a Álgebra Booleana poderia ser usada para análise e projeto de circuitos de comutação



George Boole (1815-1864)



Claude Elwood Shannon (1916-2001)

Histórico

- Nos primórdios da eletrônica, todos os problemas eram solucionados por meio de sistemas analógicos
- Com o avanço da tecnologia, os problemas passaram a ser solucionados pela eletrônica digital
- Na eletrônica digital, os sistemas (computadores, processadores de dados, sistemas de controle, codificadores, decodificadores, etc) empregam um pequeno grupo de circuitos lógicos básicos, que são conhecidos como portas **e**, **ou**, **não** e ***flip-flop***
- Com a utilização adequadas dessas portas é possível implementar todas as expressões geradas pela álgebra de Boole

Álgebra Booleana

- Na álgebra de Boole, há somente dois **estados** (**valores** ou **símbolos**) permitidos
  - Estado **0** (zero)
  - Estado **1** (um)
- Em geral
  - O estado zero representa **não**, **falso**, aparelho desligado, ausência de tensão, chave elétrica desligada, etc
  - O estado um representa **sim**, **verdadeiro**, aparelho ligado, presença de tensão, chave ligada, etc

Álgebra Booleana

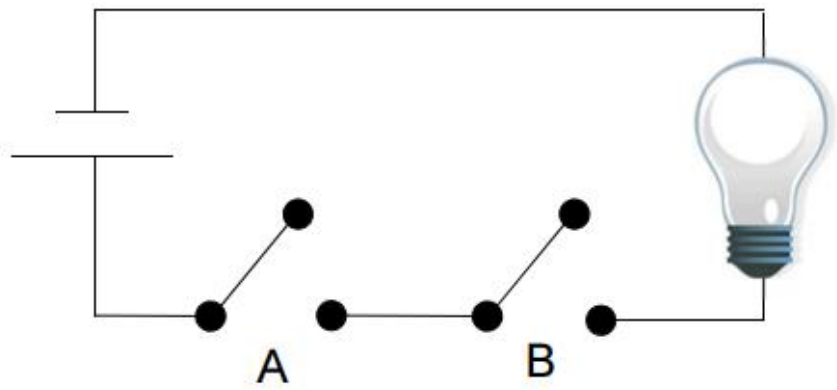
- Assim, na álgebra booleana, se representarmos por 0 uma situação, a situação contrária é representada por 1
- Portanto, em qualquer bloco (porta ou função) lógico somente esses dois estados (0 ou 1) são permitidos em suas entradas e saídas
- Uma variável booleana também só assume um dos dois estados permitidos (0 ou 1)

Álgebra Booleana

- Nesta apresentação trataremos dos seguintes blocos lógicos
  - E (AND)
  - OU (OR)

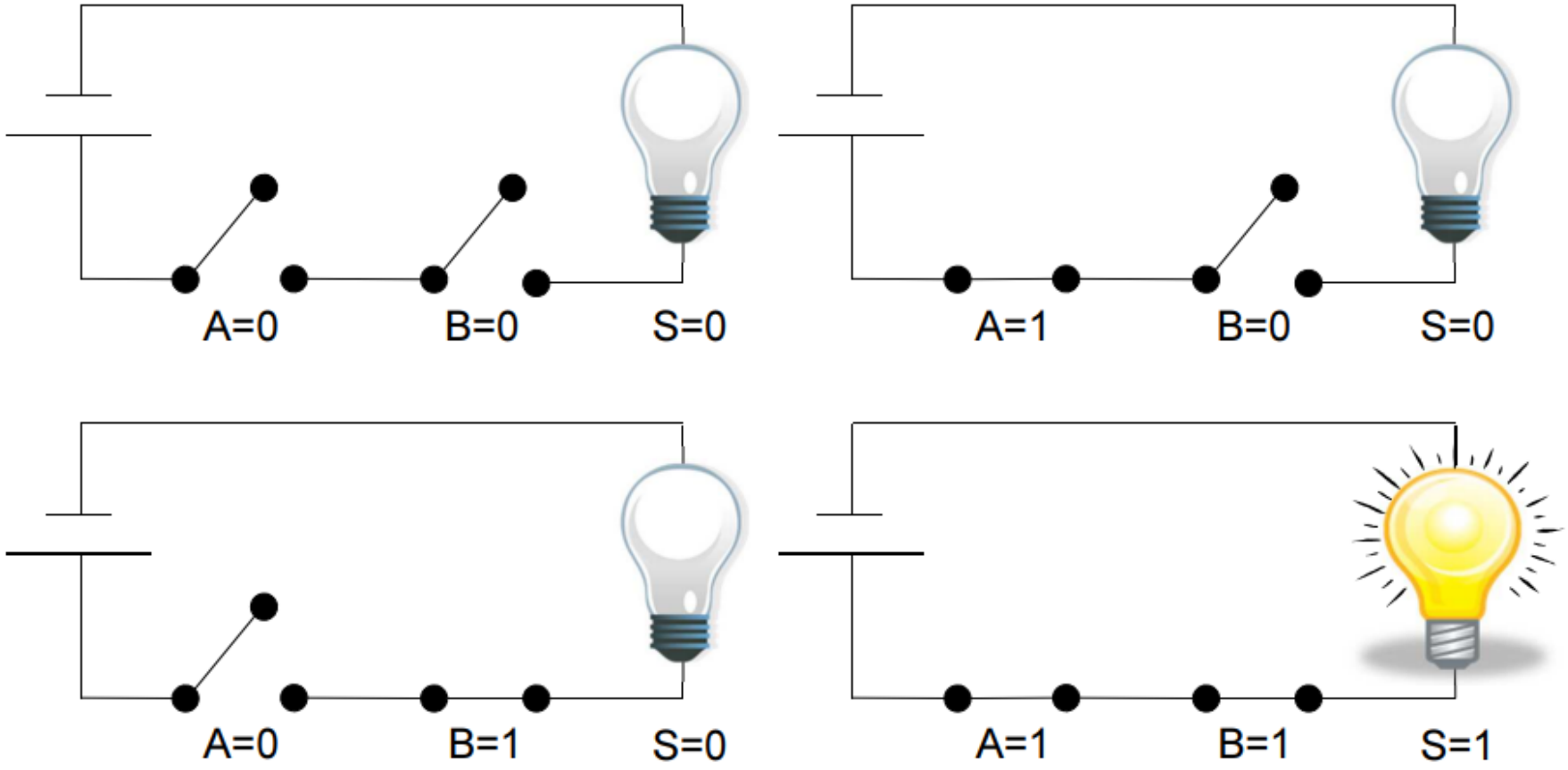
Função **E** (**AND**)

- Executa a **multiplicação** (**conjunção**) booleana de duas ou mais variáveis binárias
- Por exemplo, assuma a convenção no circuito
  - Chave aberta = 0; Chave fechada = 1
  - Lâmpada apagada = 0; Lâmpada acesa = 1



Função **E** (**AND**)

- Situações possíveis:



Função **E** (**AND**)

- Se a chave A está aberta (A=0) e a chave B aberta (B=0), não haverá circulação de energia no circuito, logo a lâmpada fica apagada (S=0)
- Se a chave A está fechada (A=1) e a chave B aberta (B=0), não haverá circulação de energia no circuito, logo a lâmpada fica apagada (S=0)
- Se a chave A está aberta (A=0) e a chave B fechada (B=1), não haverá circulação de energia no circuito, logo a lâmpada fica apagada (S=0)
- Se a chave A está fechada (A=1) e a chave B fechada (B=1), haverá circulação de energia no circuito e a lâmpada fica acesa (S=1)
- Observando todas as quatro situações possíveis (interpretações), é possível concluir que a lâmpada fica acesa somente quando as chaves A e B estiverem simultaneamente fechadas (A=1 e B=1)

Função **E** (**AND**)

- Para representar a expressão
  - S = A **e** B
- Adotaremos a representação
  - S = A.B, onde se lê S = A **e** B
- Porém, existem notações alternativas
  - S = A & B
  - S = A, B
  - S = A $\wedge$ B

Tabela Verdade

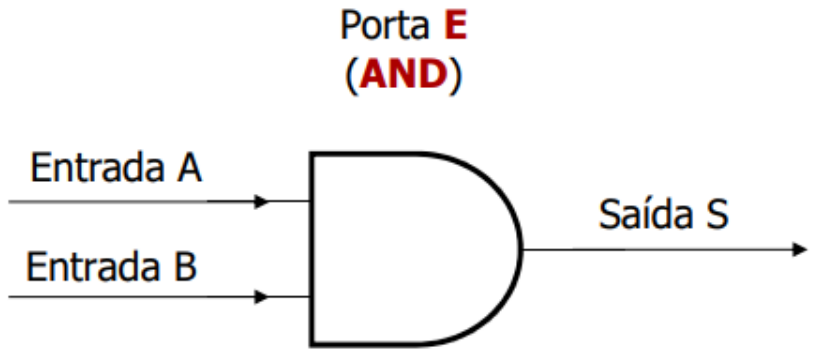
- A tabela verdade é um mapa onde são colocadas todas as possíveis interpretações (situações), com seus respectivos resultados para uma expressão booleana qualquer
- Como visto no exemplo anterior, para 2 variáveis booleanas (A e B), há 4 interpretações possíveis
- Em geral, para *N* variáveis booleanas de entrada, há 2^N interpretações possíveis

Tabela Verdade da Função **E** (**AND**)

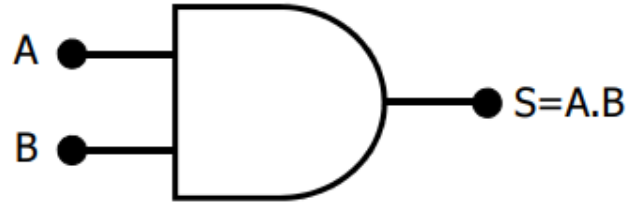
A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

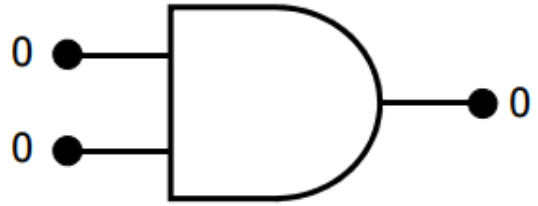
Porta Lógica **E** (**AND**)

- A porta **E** é um circuito que executa a função **E**
- A porta **E** executa a tabela verdade da função **E**
  - Portanto, a saída será 1 somente se ambas as entradas forem iguais a 1; nos demais casos, a saída será 0
- Representação

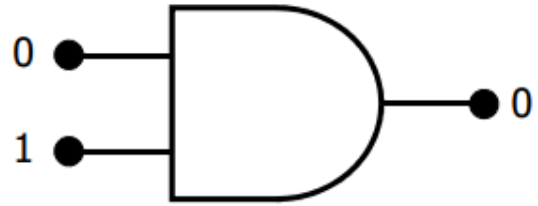


Porta Lógica **E** (**AND**)

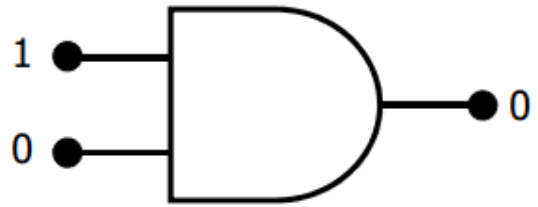




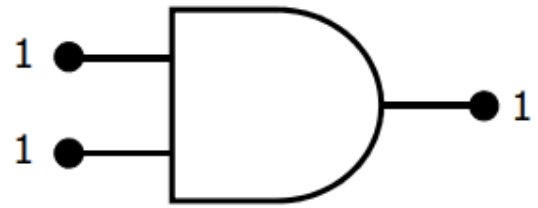
A	B	S=A.B
0	0	0
0	1	0
1	0	0
1	1	1



A	B	S=A.B
0	0	0
0	1	0
1	0	0
1	1	1



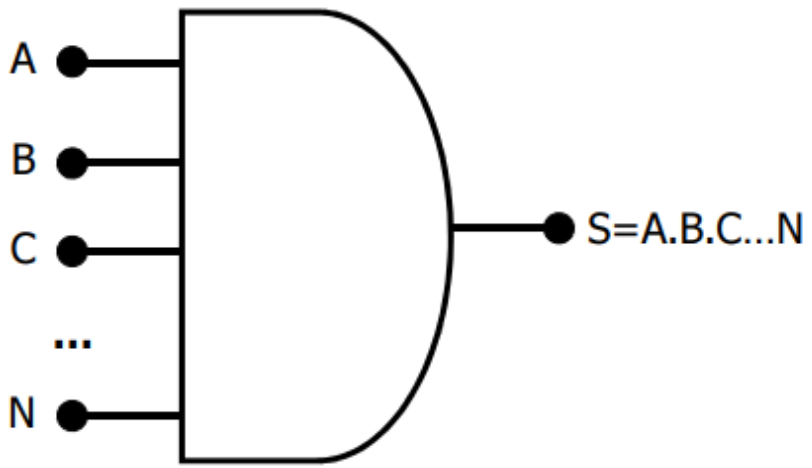
A	B	S=A.B
0	0	0
0	1	0
1	0	0
1	1	1



A	B	S=A.B
0	0	0
0	1	0
1	0	0
1	1	1

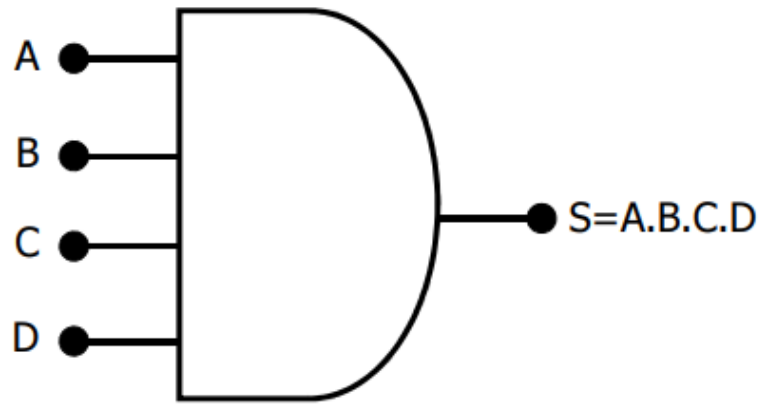
Porta Lógica **E** (**AND**)

- É possível estender o conceito de uma porta **E** para um número qualquer de variáveis de entrada
- Nesse caso, temos uma porta **E** com N entradas e somente uma saída
- A saída será 1 se e somente se as N entradas forem iguais a 1; nos demais casos, a saída será 0



Porta Lógica **E** (**AND**)

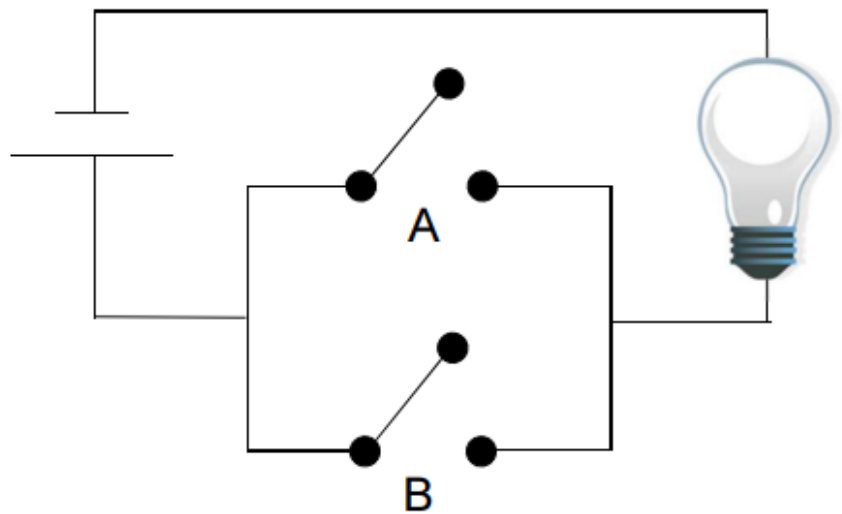
- Por exemplo, S=A.B.C.D



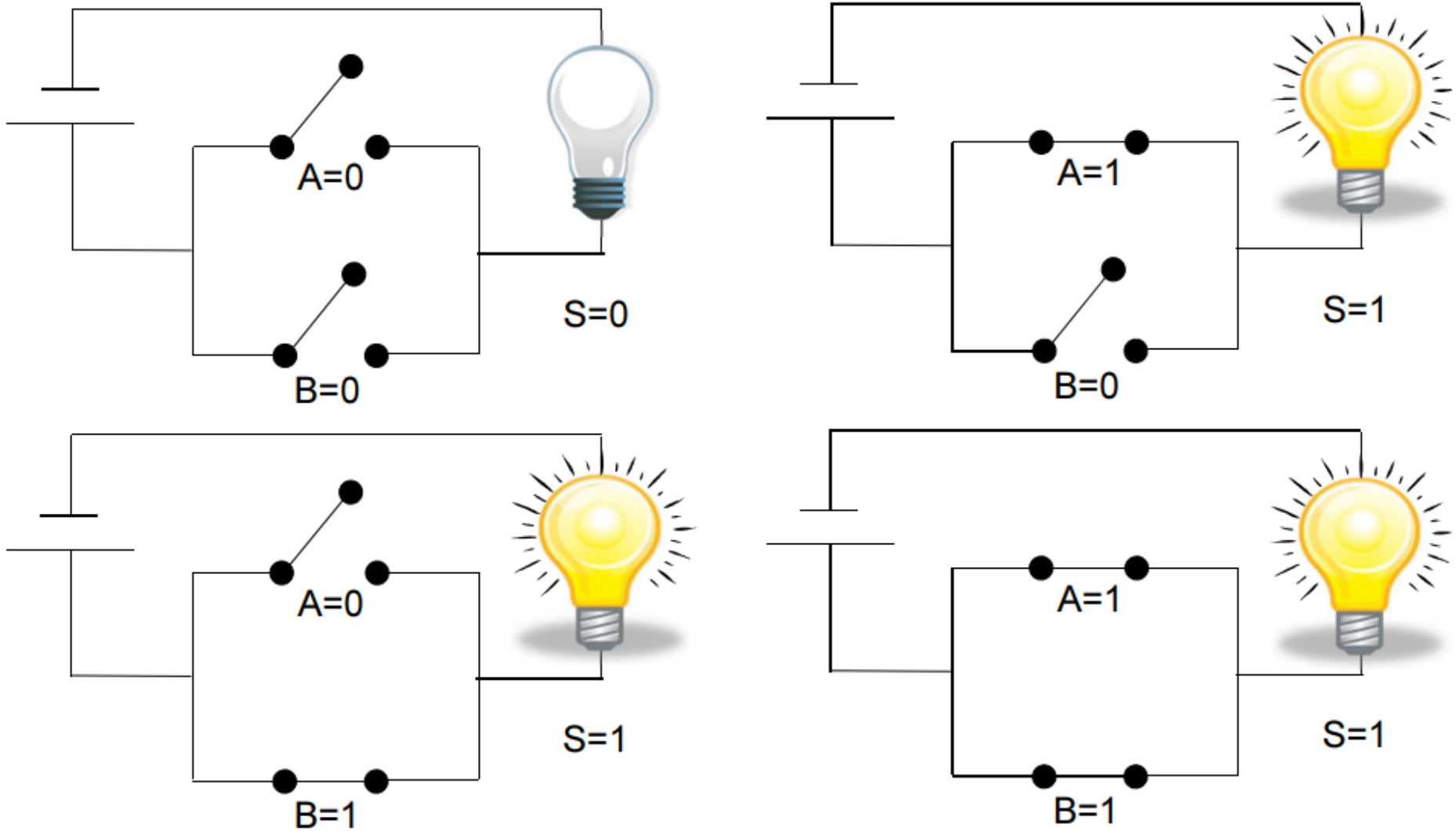
A	B	C	D	S
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

Função **OU** (**OR**)

- Executa a **soma** (**disjunção**) booleana de duas ou mais variáveis binárias
- Por exemplo, assuma a convenção no circuito
  - Chave aberta = 0; Chave fechada = 1
  - Lâmpada apagada = 0; Lâmpada acesa = 1



Função **OU** (**OR**)



Função **OU** (**OR**)

- Para representar a expressão
  - S = A **ou** B
- Adotaremos a representação
  - S = A**+**B, onde se lê S = A **ou** B
- Porém, existem notações alternativas
  - S = A | B
  - S = A; B
  - S = A $\vee$ B

Tabela Verdade da Função **OU** (**OR**)

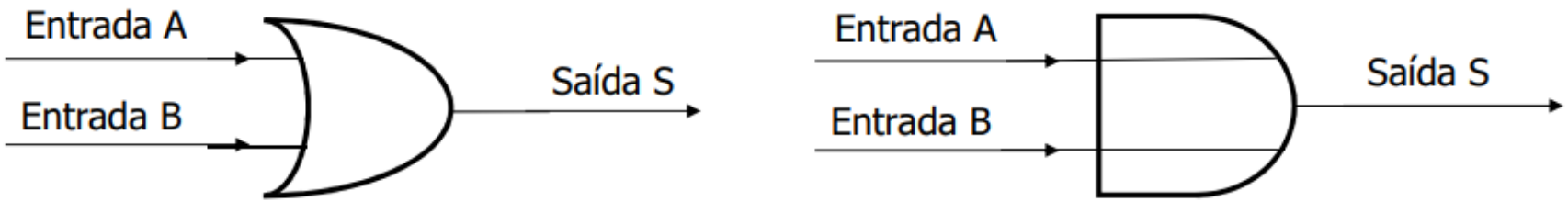
- Observe que, no sistema de numeração binário, a soma 1+1=10
- Na álgebra booleana, 1+1=1, já que somente dois valores são permitidos (0 e 1)

A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

Porta Lógica **OU** (**OR**)

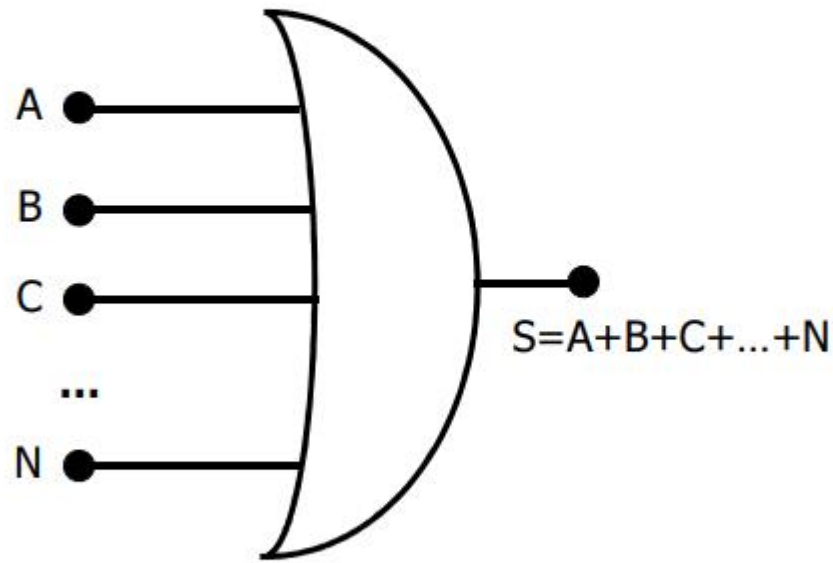
- A porta **OU** é um circuito que executa a função **OU**
- A porta **OU** executa a tabela verdade da função **OU**
  - Portanto, a saída será 0 somente se ambas as entradas forem iguais a 0; nos demais casos, a saída será 1
- Representação

Porta **OU**
(**OR**)



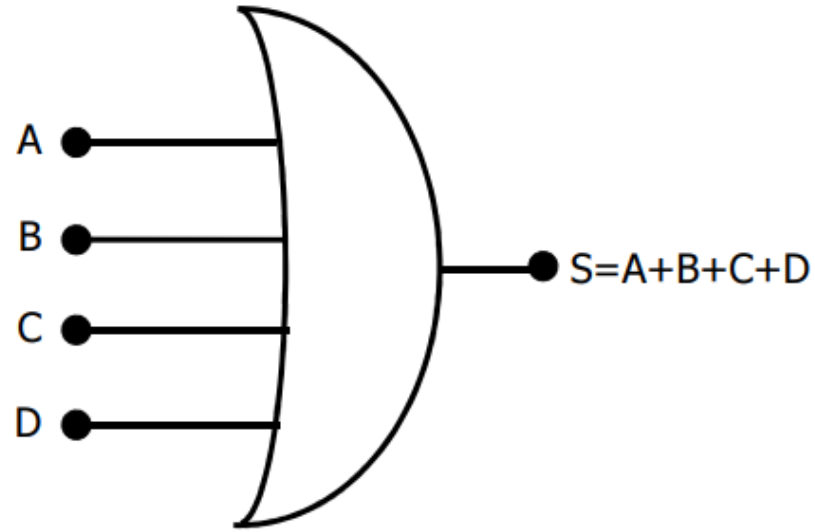
Porta Lógica **OU** (**OR**)

- É possível estender o conceito de uma porta **OU** para um número qualquer de variáveis de entrada
- Nesse caso, temos uma porta **OU** com N entradas e somente uma saída
- A saída será 0 se e somente se as N entradas forem iguais a 0; nos demais casos, a saída será 1



Porta Lógica **OU** (**OR**)

- Por exemplo, S=A+B+C+D



A	B	C	D	S
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
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	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Função **NÃO** (**NOT**)

- Executa o **complemento** (**negação**) de uma variável binária
  - Se a variável estiver em 0, o resultado da função é 1
  - Se a variável estiver em 1, o resultado da função é 0
- Essa função também é chamada de **inversora**

Exercício

- Pegue seu primeiro nome
- Converta em binário
- Agora aplique a função NOT
- Depois veja quanto este binário após o NOT vale em decimal
- Em seguida consulte a tabela ASCII e veja qual(is) letra(s) representa
- Envie por e-mail ou entregue numa folha ao final da aula.