# Aritmética Digital Inteira

Aritmética inteira → trata das operações com números inteiros

Aritmética em ponte flutuante → trata das operações com números reais

Números reais → representados por dois campos: base e mantissa

**Os algoritmos para tratar de números reais são baseados na tarefa de normalização dos operandos**

# Aritmética Digital Inteira

Aritmética inteira → Sinalização

**Duas representações:**

a) Sinal e magnitude → adiciona-se 1 bit de sinal

EX: $(13)_{10}$ → 1101 (natural ) com sinal +13: 01101; -13: 11101

b) Complemento de 2:

EX: +13 → 01101; -13 → 10010 (complemento de 1) e soma 1

```
   10010
+      1
-----------
   10011  (complemento de 2)
```

# Aritmética Digital Inteira

Aritmética inteira → operação adição

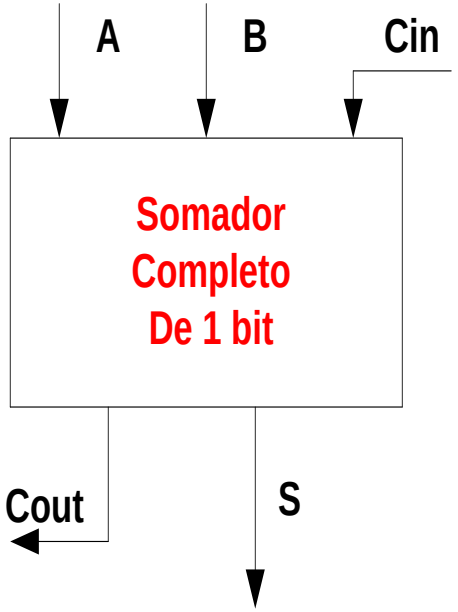
Regras:

0+0=0

0+1=1

1+0=1

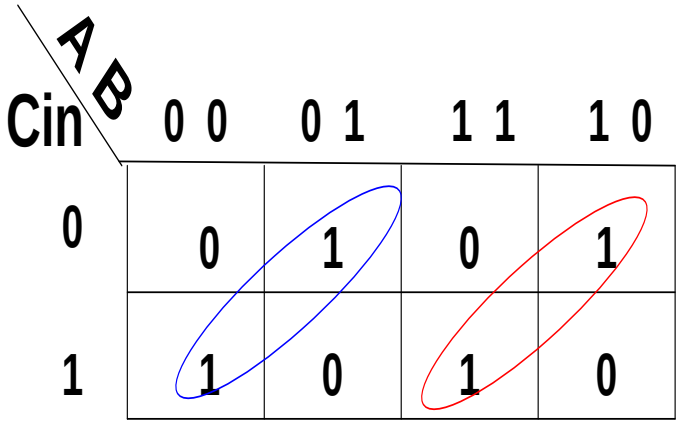
1+1= 0 e vai 1

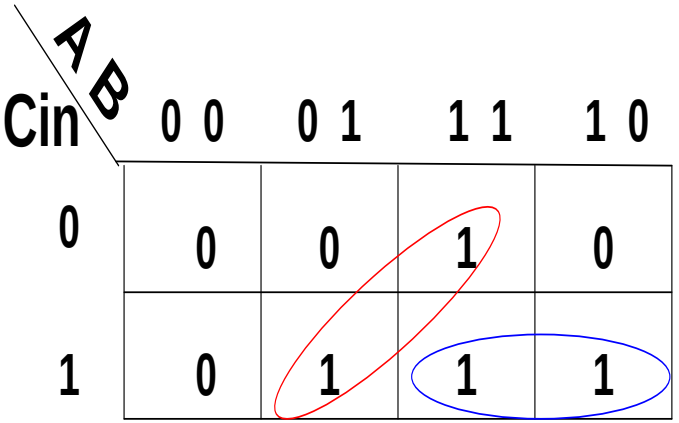



| A | B | Cin | S | Cout |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

# Aritmética Digital Inteira

## Aritmética inteira → operação adição




$S = A \oplus B \oplus Cin$

$Cout = A\,Cin + B(A \oplus Cin)$

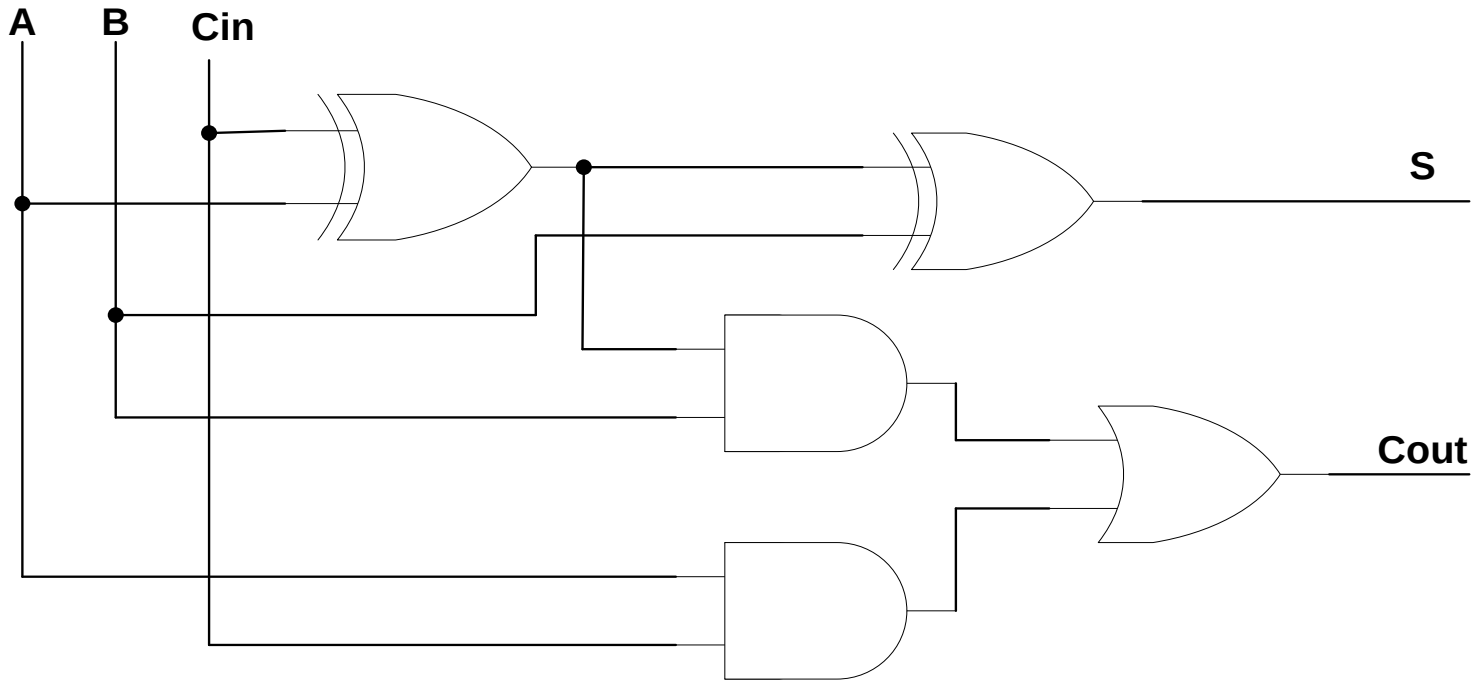
$$S = \overline{Cin}\,(A\,\overline{B} + \overline{A}\,B) + Cin\,(A\,B + \overline{A}\,\overline{B})$$

$$S = \overline{Cin}\,\underbrace{(A \oplus B)}_{X} + Cin\,\underbrace{\overline{(A \oplus B)}}_{\overline{X}}$$

$$S = Cin \oplus A \oplus B$$

# Aritmética Digital Inteira

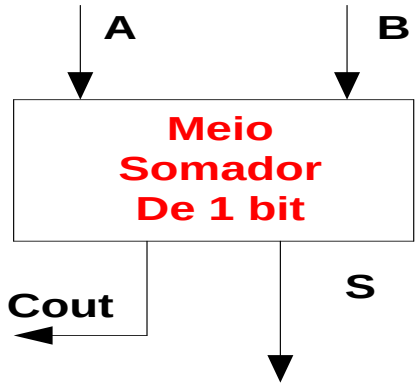
## Somador completo de 1 bit → circuito lógico




**S=A ⊕ B ⊕Cin;  Cout = ACin + B (A⊕ Cin)**

# Aritmética Digital Inteira

## Aritmética inteira → operação adição

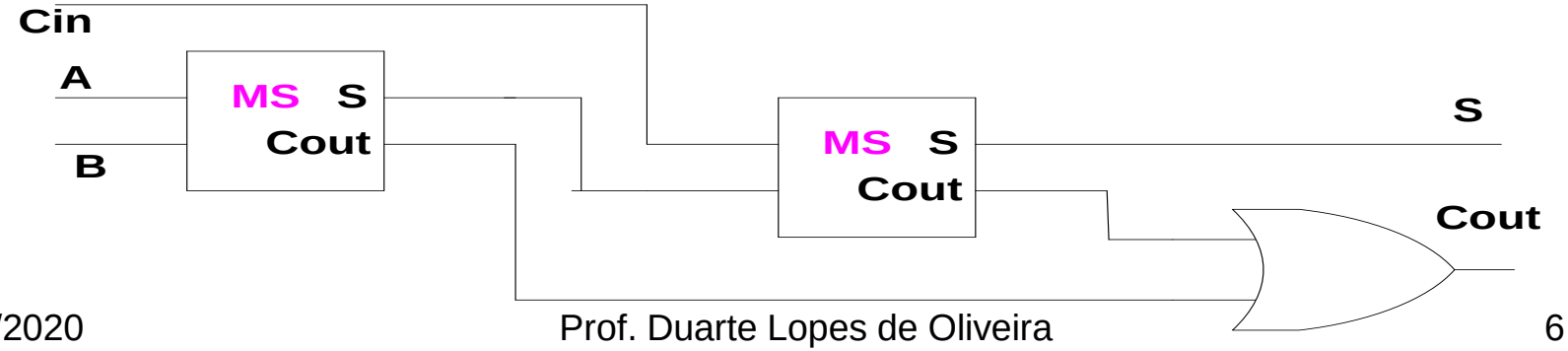



| A | B | S | Cout |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

$S = A \oplus B$

$Cout = A\ B$

**Somador completo usando meio somador**

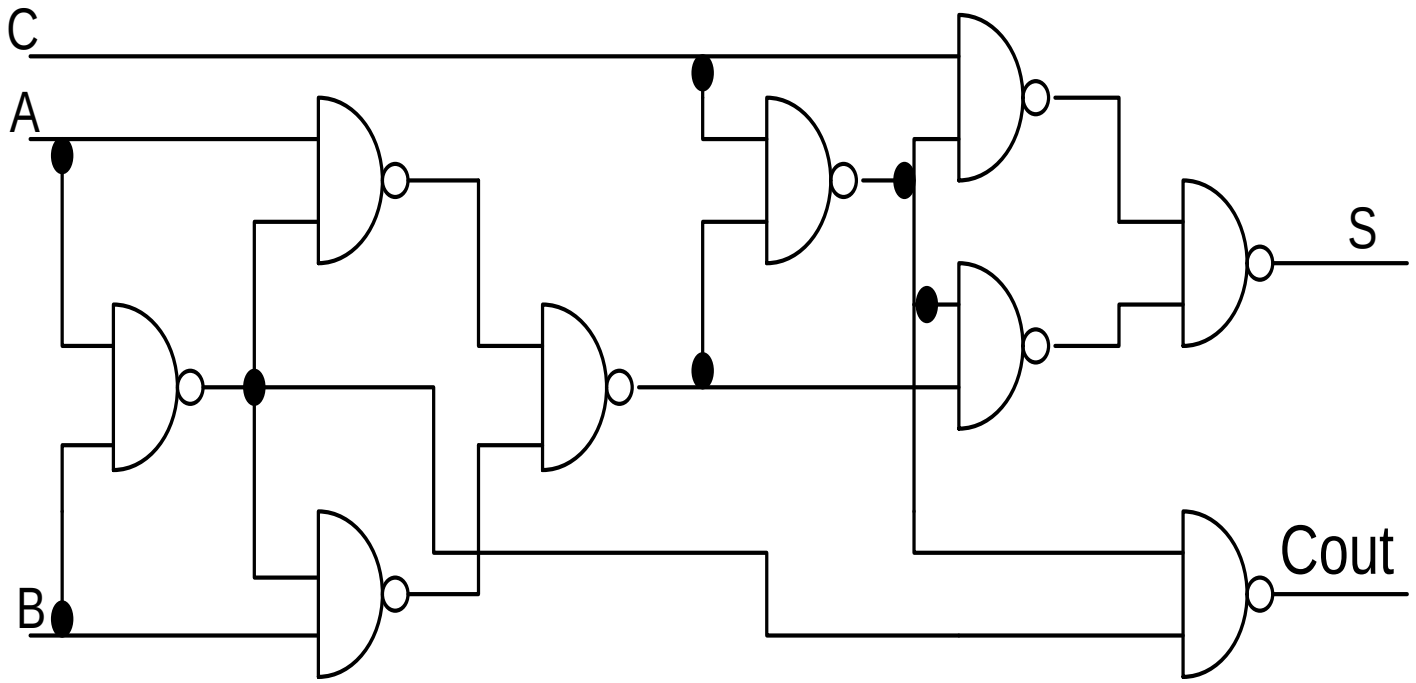


# Aritmética Digital Inteira

## Somador completo de 1 bit → **configurações**

(36 transistores – 18 número total de literais)

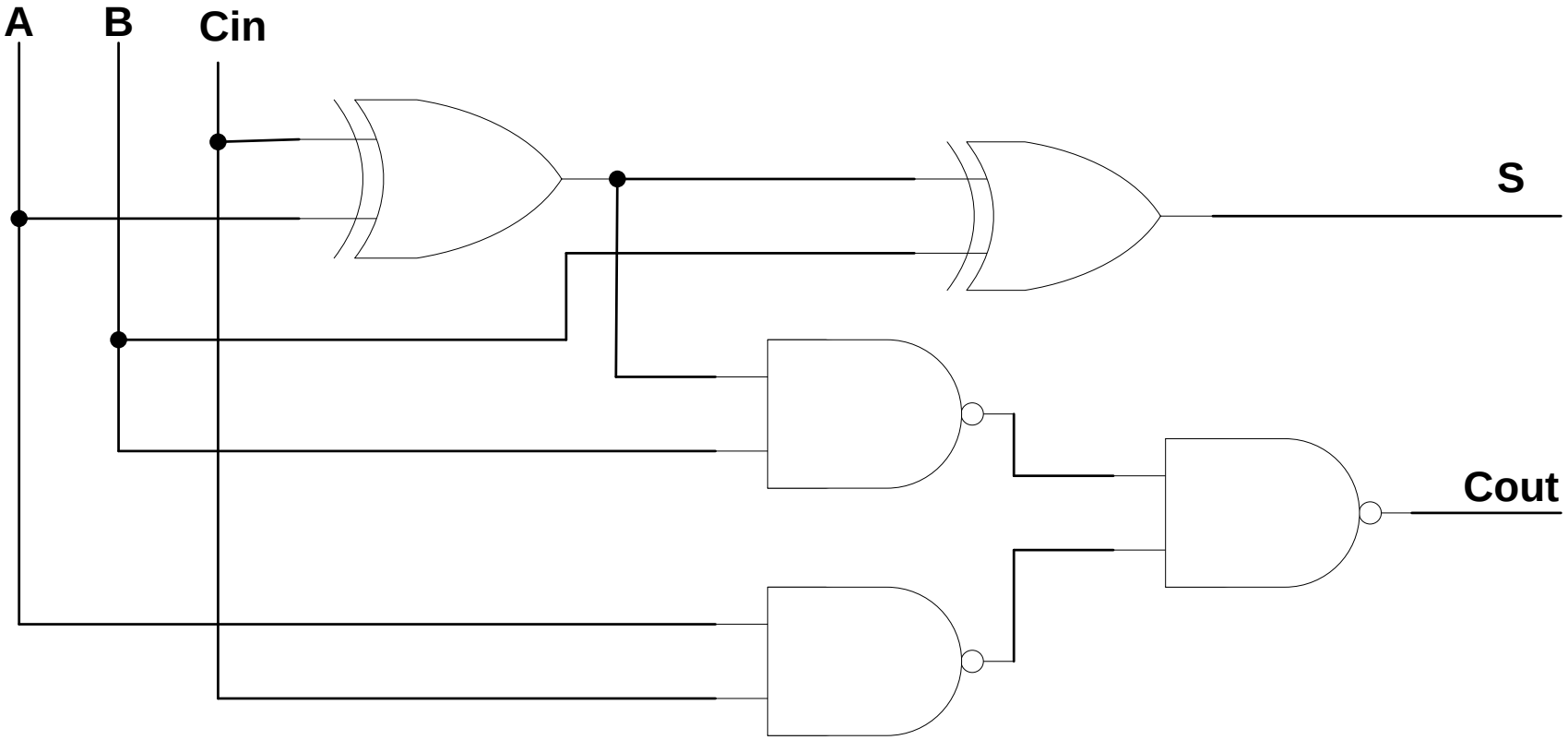
**Para porta NAND → Número de transistores = 2 Fan-in**



# Aritmética Digital Inteira

## Somador completo de 1 bit → **configurações**

(24 transistores – 10 número total de literais)

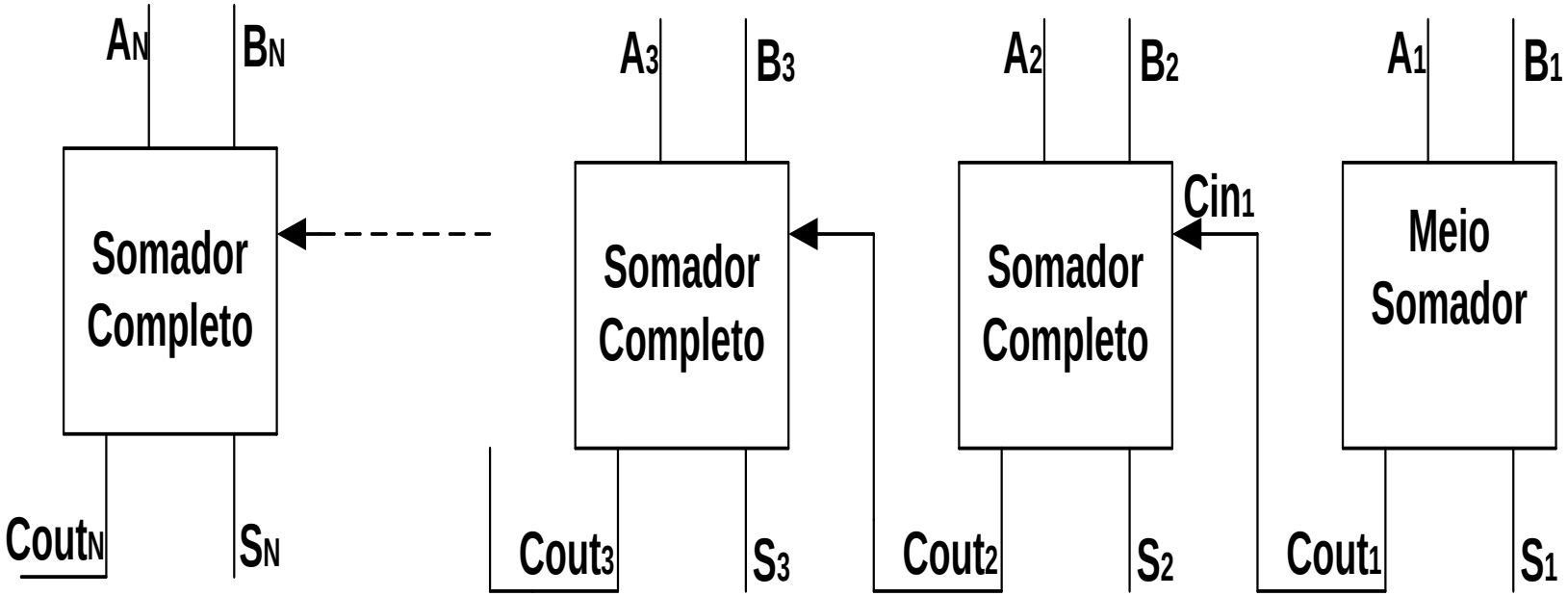



**Porta XOR → 6 transistores**

# Aritmética Digital Inteira

## Somador completo de 1 bit → **célula básica**

(Somador de N bits -→ propagação do Carry)

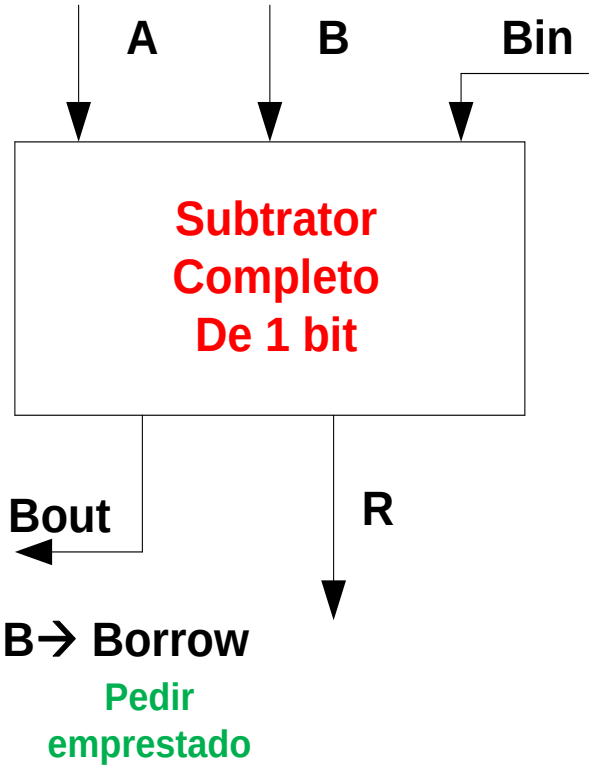



**Carry Propagation**

# Aritmética Digital Inteira

## Aritmética inteira → operação subtração

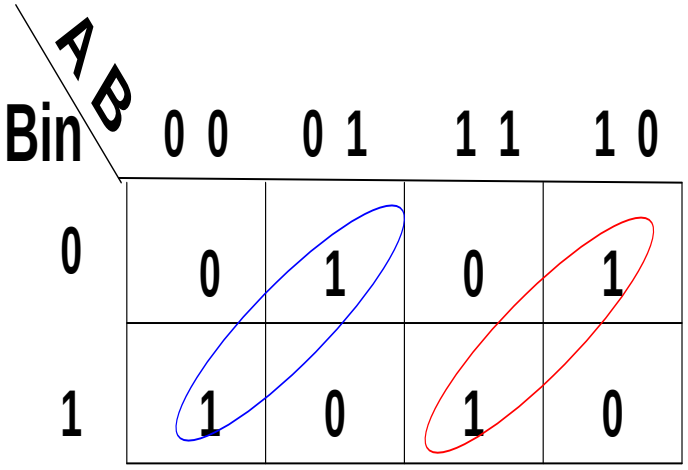
Regras:
0-0=0
0-1=1 e empresta 1
1-0=1
1-1=0



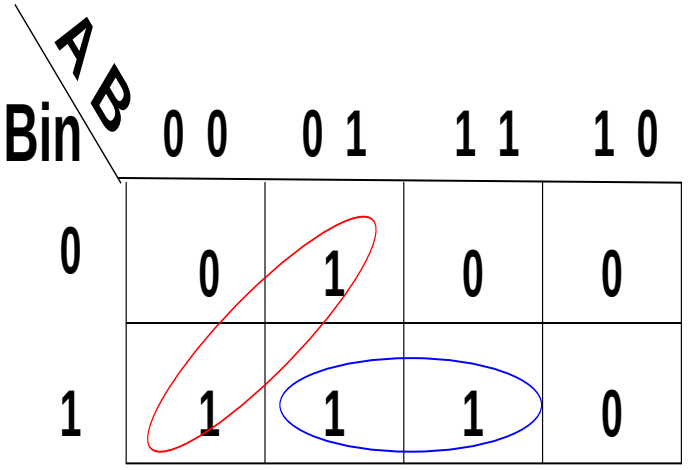

| A | B | Bin | R | Bout |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 |

# Aritmética Digital Inteira

Aritmética inteira → operação subtração



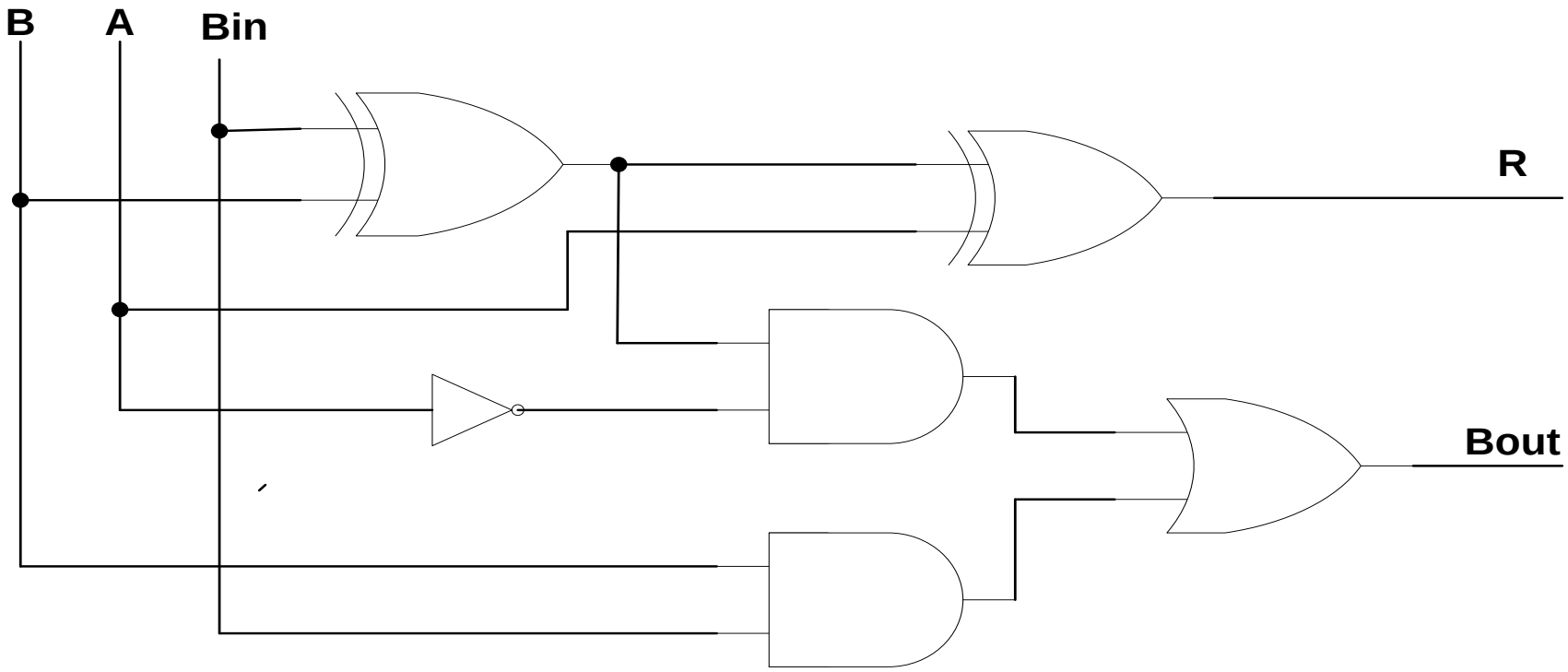

$R = A \oplus B \oplus Bin$




$Bout = B\,Bin + \overline{A}(B \oplus Bin)$

# Aritmética Digital Inteira

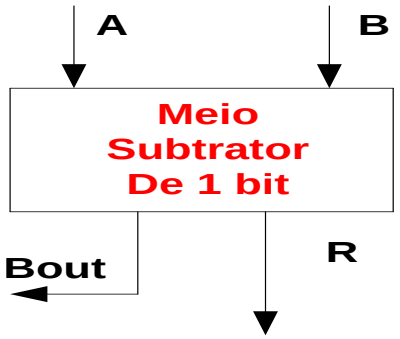
## Subtrator completo de 1 bit → circuito lógico




**R = A ⊕B ⊕ Bin;  Bout = BBin + A' (B ⊕Bin)**

# Aritmética Digital Inteira

## Aritmética inteira → operação subtração



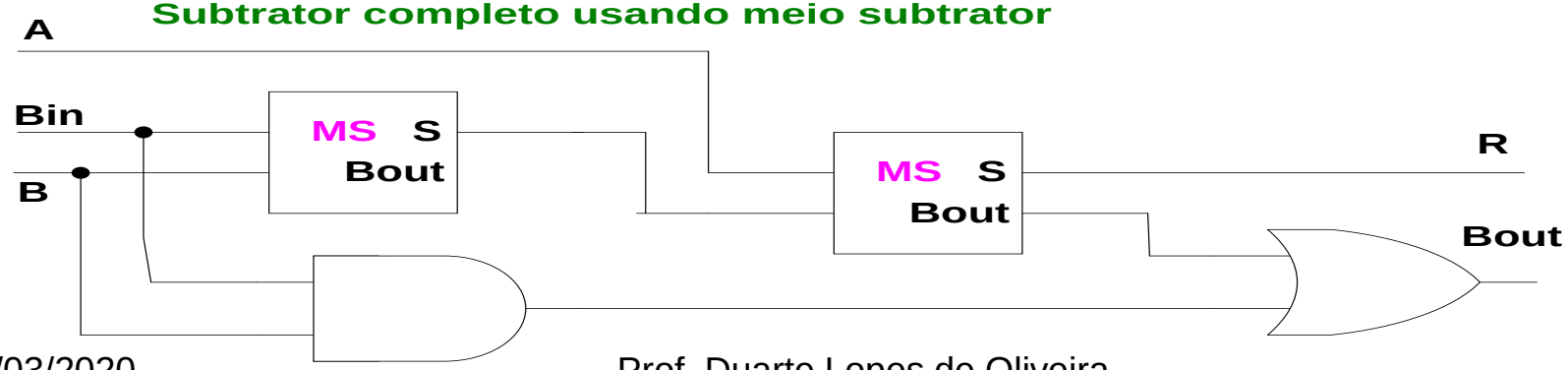

R=A menos B

| A | B | R | Bout |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |

$R = A \oplus B$

$Bout = \overline{A}\,B$

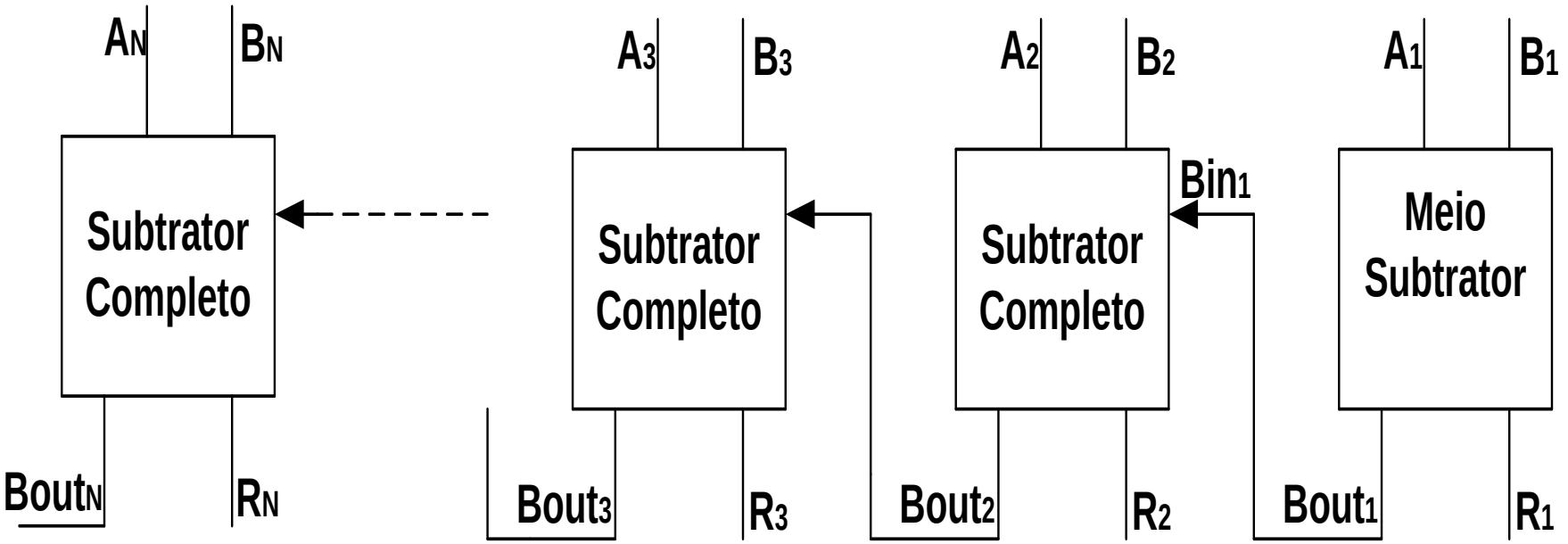
Subtrator completo usando meio subtrator



# Aritmética Digital Inteira

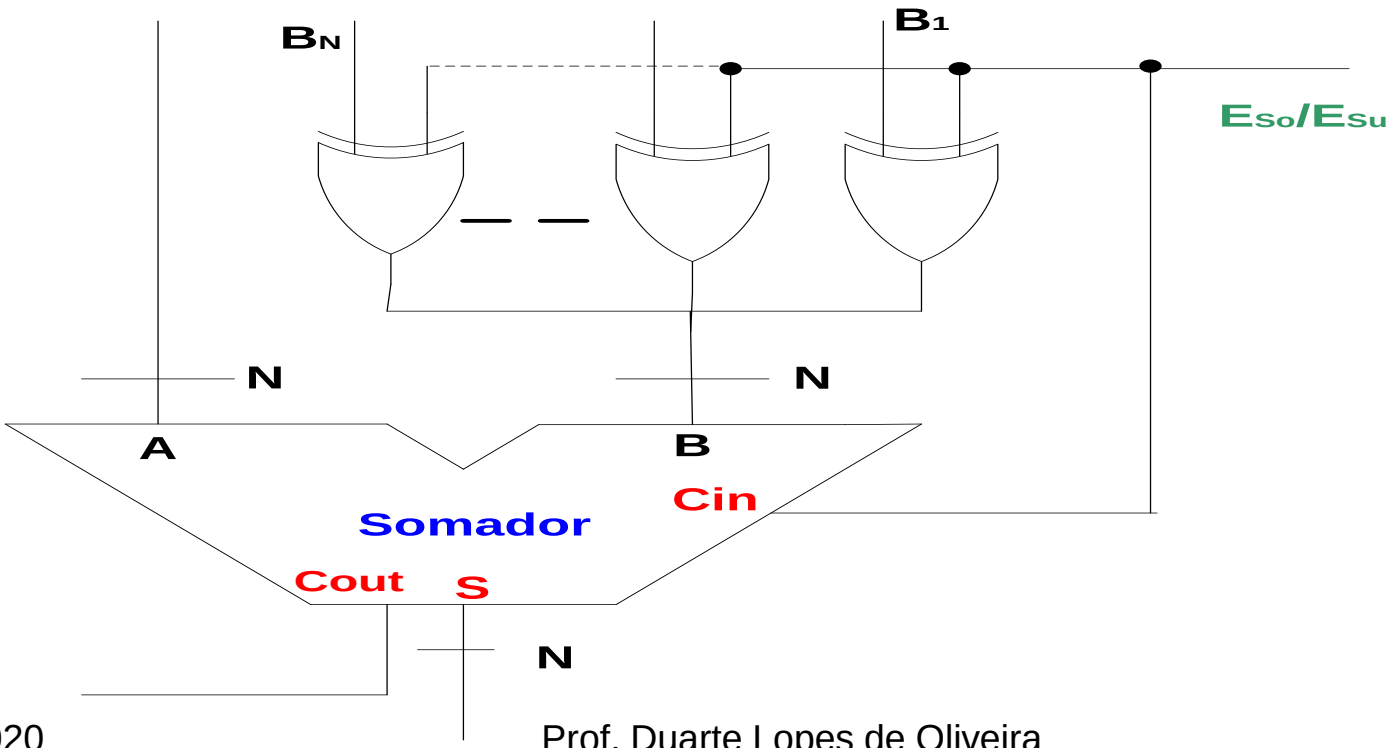
Subtrator completo de 1 bit → **célula básica**

**Subtrator de N bits → propagação de Borrow**



# Aritmética Digital Inteira

Aritmética inteira → **Subtração por complemento de 2**



# Aritmética Digital Inteira

Aritmética inteira → Paralela

Sejam Ai e Bi os bits de entrada da i-esima coluna dos operandos A e B

Variável de geração

$G_i = A_i B_i$ (1)

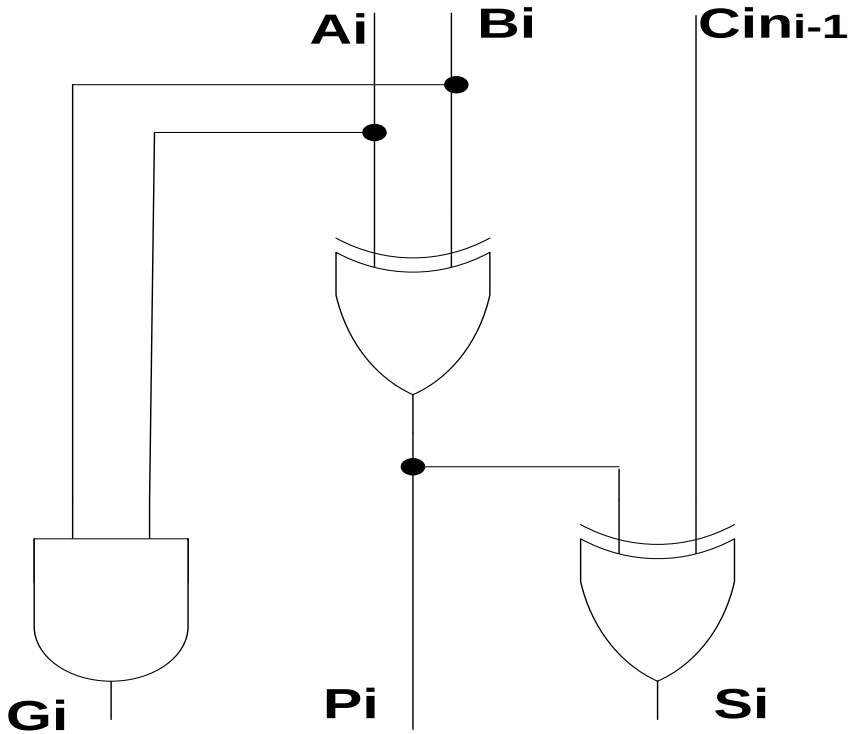
Variável de propagação:

$P_i = A_i \oplus B_i$ (2)

Somador

$S_i = A_i \oplus B_i \oplus Cin_{i-1}$ (3)

(1),(2) e (3) → gera a unidade somadora



# Aritmética Digital Inteira

Aritmética inteira → Paralela

Sejam Ai e Bi os bits de entrada da i-esima coluna dos operandos A e B

Variável de geração Gi=AiBi (1)

Variável de propagação: Pi=Ai⊕Bi (2)

Sabendo que, $C_{i+1}$=AiBi +Ci(Ai⊕Bi) (3)

Usando (1) e (2) em (3) temos:

**$C_{i+1}$=Gi +CiPi (4)**

# Aritmética Digital Inteira

Aritmética inteira → Paralela: $C_{i+1}=Gi + CiPi$ **(4)**

Aplicando (4) em cada estágio:

$C1=G0+CoPo$

$C2=G1+C1P1=G1+(G0+CoPo)P1$

$= G1 + G0P1 + CoPoP1$

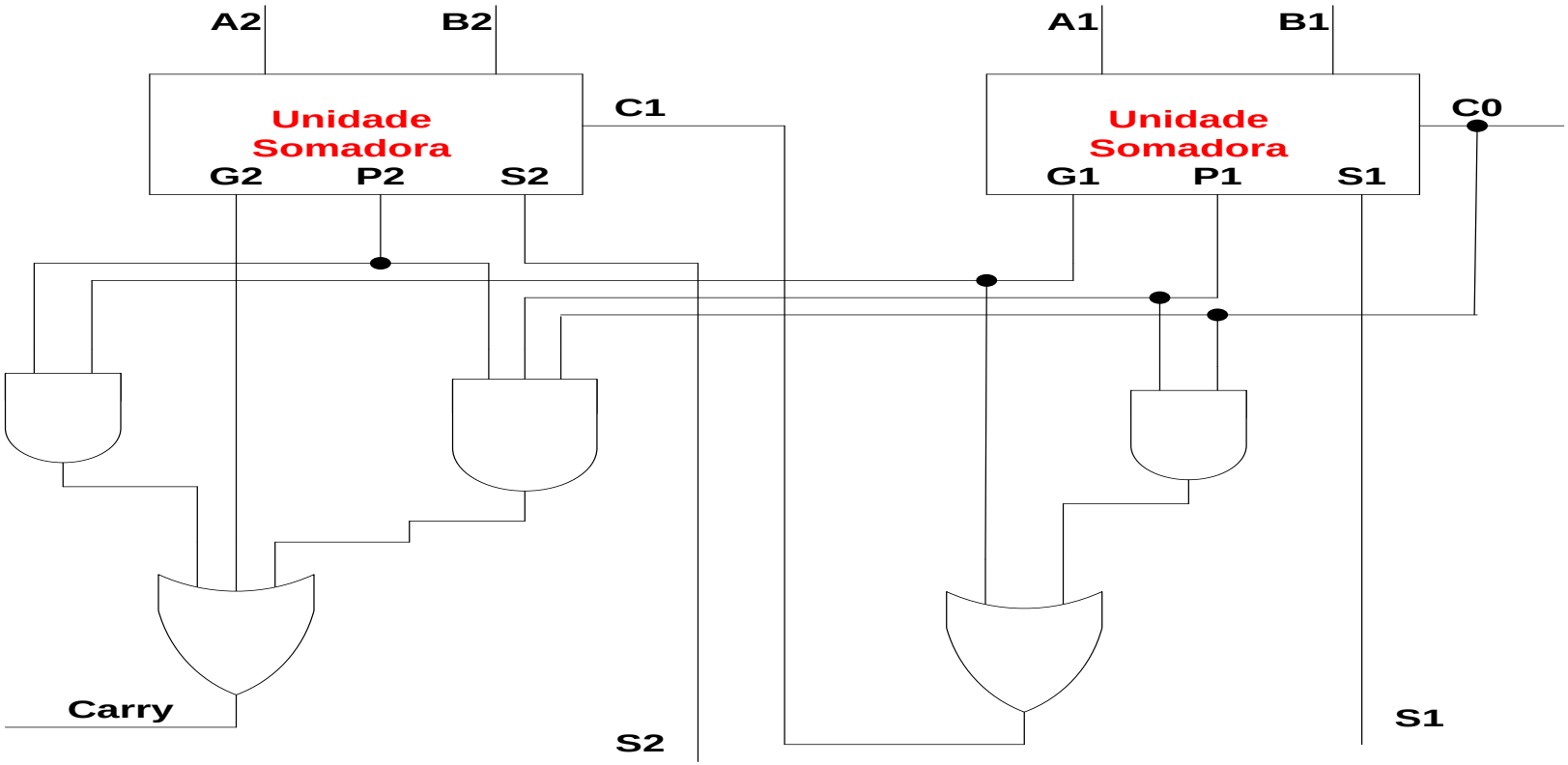
$C3=G2+C2P2=G2+(G1+G0P1+CoPoP1 + CoPoP1)P2$

$=G2+G1P2+GoP1P2+CoPoP1P2$

# Aritmética Digital Inteira

Adição inteira paralela→ Circuito lógico de 2 bits

**Performance: para N bits → 4 níveis de atraso**

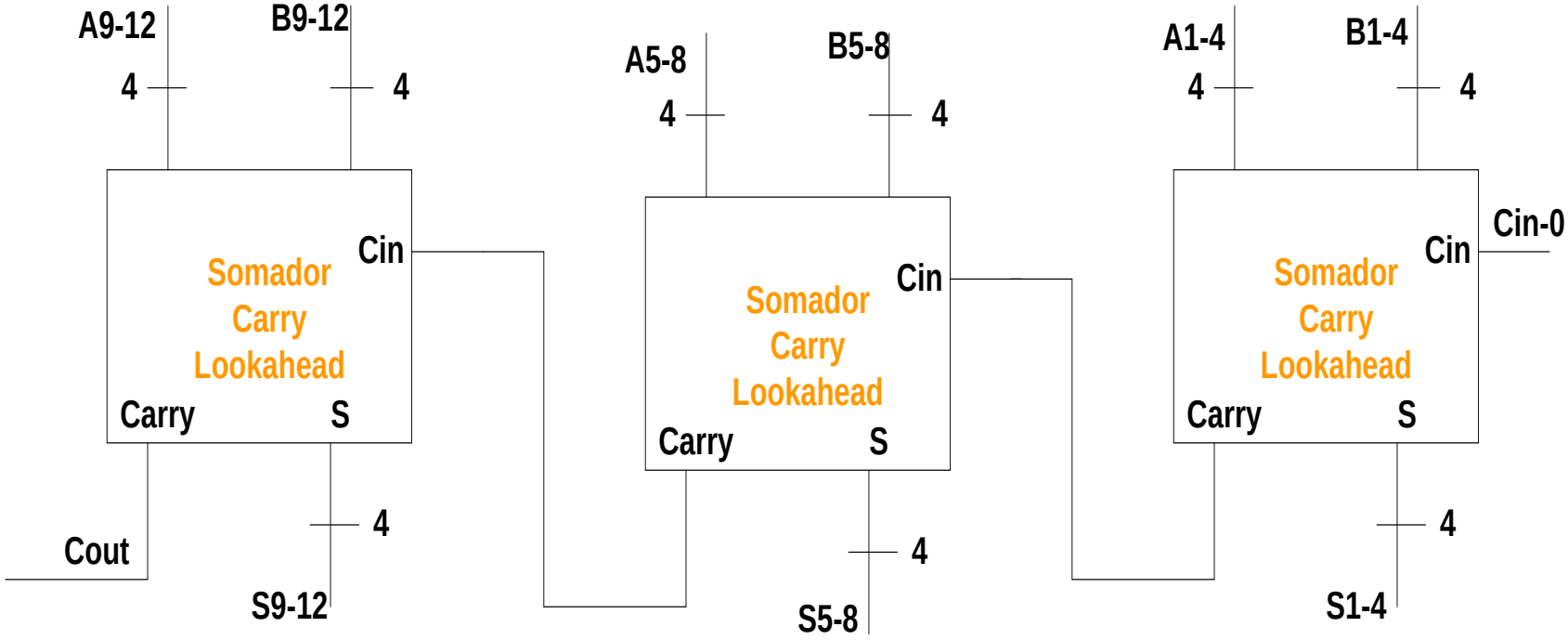


# Aritmética Digital Inteira

## Adição inteira paralela→ Solucionar problema de Fan-in →

**Performance: 12 bits (Ripple Carry– 36 níveis de atraso ; Lockahead – 4 níveis e fan-in 12)**

**Método: mistura Carry Propagation com Carry Lookahead → 12 níveis**



# Aritmética Digital Inteira

Adição inteira paralela→ Solucionar problema de Fan-in (Geração de grupos)

Geração de grupo:

$G(1\text{-}4)=G4 + G3P4 + G2P4P3 + G1P4P3P2$

Propagação do grupo:

$P(1\text{-}4)=P4P3P2P1$

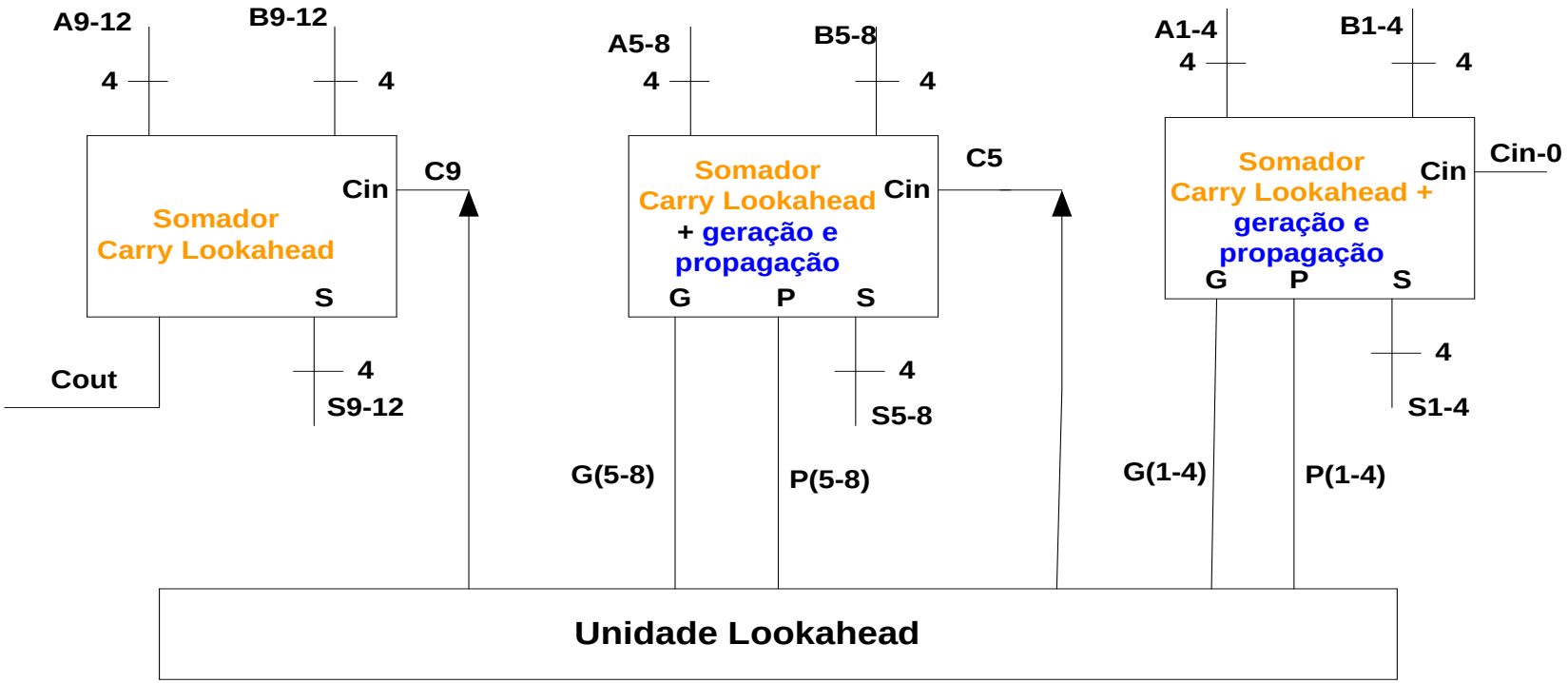
Geração dos Carry lookahead

$C5=G(1\text{-}4) + P(1\text{-}4)C0$

$C9=G(5\text{-}8) + P(5\text{-}8)G(1\text{-}4) + P(5\text{-}8)P(1\text{-}4)C0$

# Aritmética Digital Inteira

Adição inteira paralela→ Solucionar problema de Fan-in (Geração de grupos); **Performance → 6 níveis**



# Operações Relacionais

Comparador paralelo→ Comparador de 1 bit (célula) + lógica


| A | B | EQ | MA | MB |
|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 |

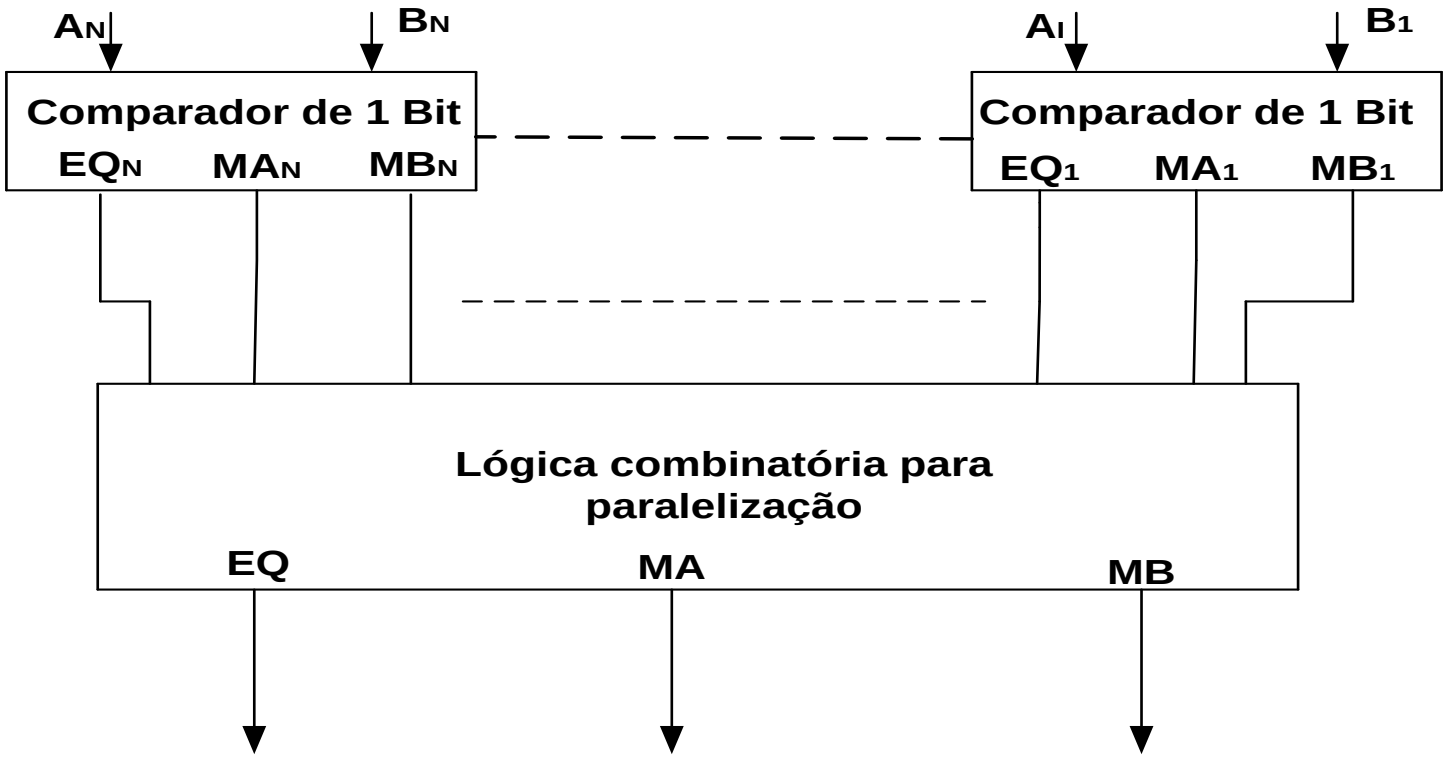
$$EQ = \overline{A}\,\overline{B} + A\,B = A \odot B$$

$$MA = A\,\overline{B}$$

$$MB = \overline{A}\,B$$

# Operações Relacionais

Comparador paralelo de N bits→ Comparador de 1 bit (célula) + lógica de paralelização



# Operações Relacionais

## Comparador paralelo de N Bits→

### Desenvolvimento da lógica de paralelização

**Para 2 Bits:**

$EQ_{Geral} = EQ_2 \; EQ_1$

$MA_{Geral} = MA_2 + EQ_2 \; MA_1$

$MB_{Geral} = MB_2 + EQ_2 \; MB_1$

**Para 3 Bits:**

$EQ_{Geral} = EQ_3 \; EQ_2 \; EQ_1$

$MA_{Geral} = MA_3 + EQ_3 \; MA_2 + EQ_3 \; EQ_2 \; MA_1$

$MB_{Geral} = MB_3 + EQ_3 \; MB_2 + EQ_3 \; EQ_2 \; MB_1$

# Operações Relacionais

## Comparador paralelo de 2 Bits→ Circuito lógico

**Para 2 Bits:**

$EQ_{Geral} = EQ_2 \; EQ_1$

$MA_{Geral} = MA_2 + EQ_2 \; MA_1$

$MB_{Geral} = MB_2 + EQ_2 \; MB_1$