Circuitos aritméticos - Somador Completo

Autores	
Gabriel A. F. Souza, Gustavo D. Colletta, Leonardo B. Zoccal, Odilon O. Dutra	Unifei

Histórico de Revisões		
10 de janeiro de 2025	1.0	Primeira versão do documento.

Tópicos

- Soma Binária - Revisão
- Circuitos Digitais para Soma Binária
- Funcionamento do Somador Completo
- Implementações do circuito Somador Completo
- Somador de múltiplos bits
- Exercícios

Soma Binária - Revisão

O que é a soma binária?

A soma binária é uma operação fundamental em circuitos digitais e sistemas computacionais, sendo a base para a realização de operações aritméticas. Ela segue as mesmas regras básicas da adição decimal, mas utiliza o sistema de numeração binário, composto apenas pelos dígitos 0 e 1.

Regras Básicas da Soma Binária

Bit 1	**Bit 2**	**Soma (S)**	**Carry (C)**
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Aqui:

- S (Sum) é o bit de soma resultante.
- C (Carry) é o bit de transporte, que é "carregado" para a próxima coluna de soma.

Soma de Múltiplos Bits

Quando somamos números binários com mais de um bit, o transporte (C) da soma de um par de bits é propagado para a próxima coluna, assim como na soma decimal. Por exemplo:

Considere somar dois números binários de 4 bits:

$$A = 1011 \quad \text{e} \quad B = 0111$$

Soma de Múltiplos Bits

Realizando a soma bit a bit:

Posição	A	B	**Carry In**	**Soma**	**Carry Out**
Bit menos significativo (LSB)	1	1	0	0	1
Próximo bit	1	1	1	1	1
Próximo bit	0	1	1	0	1
Bit mais significativo (MSB)	1	0	1	0	1

O resultado final da soma é:

$$1011 + 0111 = 10010$$

Aqui, o resultado é um número de 5 bits, com o bit mais à esquerda (1) sendo o transporte final.

Circuitos Digitais para Soma Binária

1. Meio Somador (Half Adder)

O meio somador é um circuito que realiza a soma de dois bits, mas não considera o transporte de entrada (Carry). Ele tem duas entradas (A e B) e duas saídas:

- Soma ($S = A \oplus B$, onde $\oplus$ é a operação XOR)
- Transporte ($C = A \cdot B$, onde $\cdot$ é a operação AND)

2. Somador Completo (Full Adder)

O somador completo realiza a soma de três bits: os dois bits de entrada (A e(B) e um bit de transporte de entrada (C_{in}). Ele tem duas saídas:

- Soma ($S = A \oplus B \oplus C_{\text{in}}$)
- Transporte ($C_{\text{out}} = (A \cdot B) + (C_{\text{in}} \cdot (A \oplus B))$)

Um somador completo pode ser construído conectando dois meios somadores e uma porta OR para o transporte.

3. Somadores de Múltiplos Bits

Para somar números binários com mais de um bit, os somadores completos podem ser conectados em cascata. O transporte de saída de um somador é propagado como transporte de entrada para o próximo.

Funcionamento do Somador Completo

Tabela Verdade do Somador Completo

A	B	Cin	S (Soma)	Cout (Carry)
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

Regras de Soma

As regras de soma binária para cada bit são:

1. **Soma de 0 + 0**:
   - O resultado é 0, sem transporte (Cout = 0).
2. **Soma de 0 + 1 ou 1 + 0**:
   - O resultado é 1, sem transporte (Cout = 0).
3. **Soma de 1 + 1**:
   - O resultado é 0, com transporte (Cout = 1).
4. **Soma com Cin (Carry in)**:
   - Quando a entrada **Cin** é 1, a operação é equivalente a adicionar o carry para a soma dos bits A e B. Se o carry for gerado, **Cout** será 1; caso contrário, será 0.

Implementações do circuito Somador Completo

Descrição por fluxo de dados

```
1  module full_adder (
2      input A,          // Entrada A
3      input B,          // Entrada B
4      input Cin,        // Carry de entrada
5      output S,         // Soma
6      output Cout       // Carry de saída
7  );
8
9      // Fluxo de dados para a soma e carry de saída
10     assign S = A ^ B ^ Cin;          // Soma é a XOR de A, B e Cin
11     assign Cout = (A & B) | (Cin & (A^B)); // Carry de saída é
           a combinação dos carries de A, B e Cin
12
13 endmodule
```

Diagrama esquemático

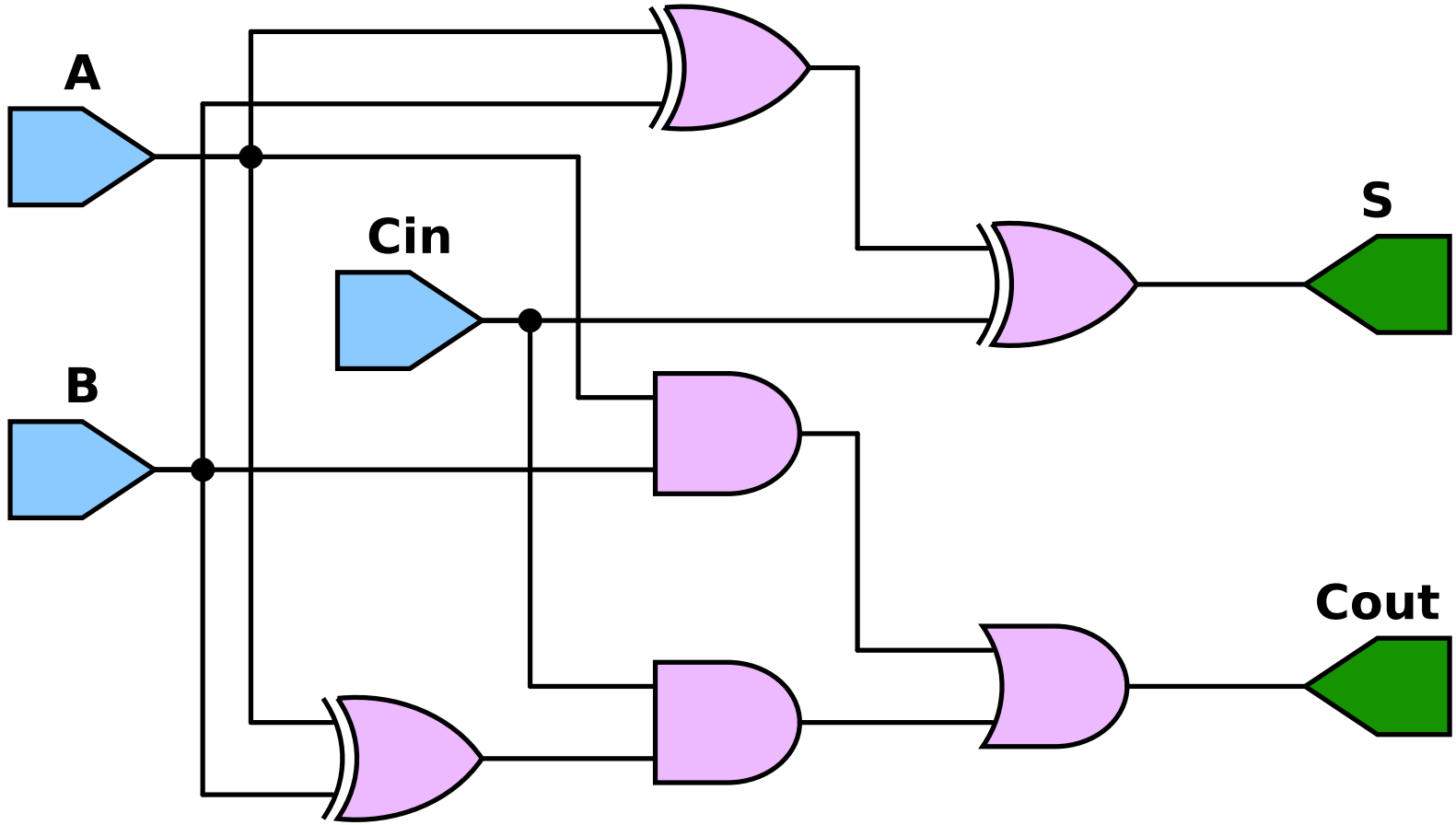

Figura 1: Esquemático do circuito somador completo.

Descrição estrutural 1

```
1  module somador_completo (
2      input A,          // Primeiro bit
3      input B,          // Segundo bit
4      input Cin,        // Carry in (entrada de transporte)
5      output S,         // Soma
6      output Cout       // Carry out (transporte de saída)
7  );
8
9  // Definição das portas lógicas
10 wire xor_ab;       // Resultado da operação A XOR B
11 wire and_ab;       // Resultado da operação A AND B
12 wire and_cin_ab;   // Resultado da operação Cin AND (A XOR B)
```

Descrição estrutural 1

```
13 // A operação XOR entre A e B
14 xor (xor_ab, A, B);
15 // A operação AND entre A e B
16 and (and_ab, A, B);
17 // A operação AND entre Cin e (A XOR B)
18 and (and_cin_ab, Cin, xor_ab);
19 // A operação XOR final para gerar a soma S
20 xor (S, xor_ab, Cin);
21 // A operação OR para gerar o carry out Cout
22 or (Cout, and_ab, and_cin_ab);
23
24 endmodule
```

Diagrama esquemático 1

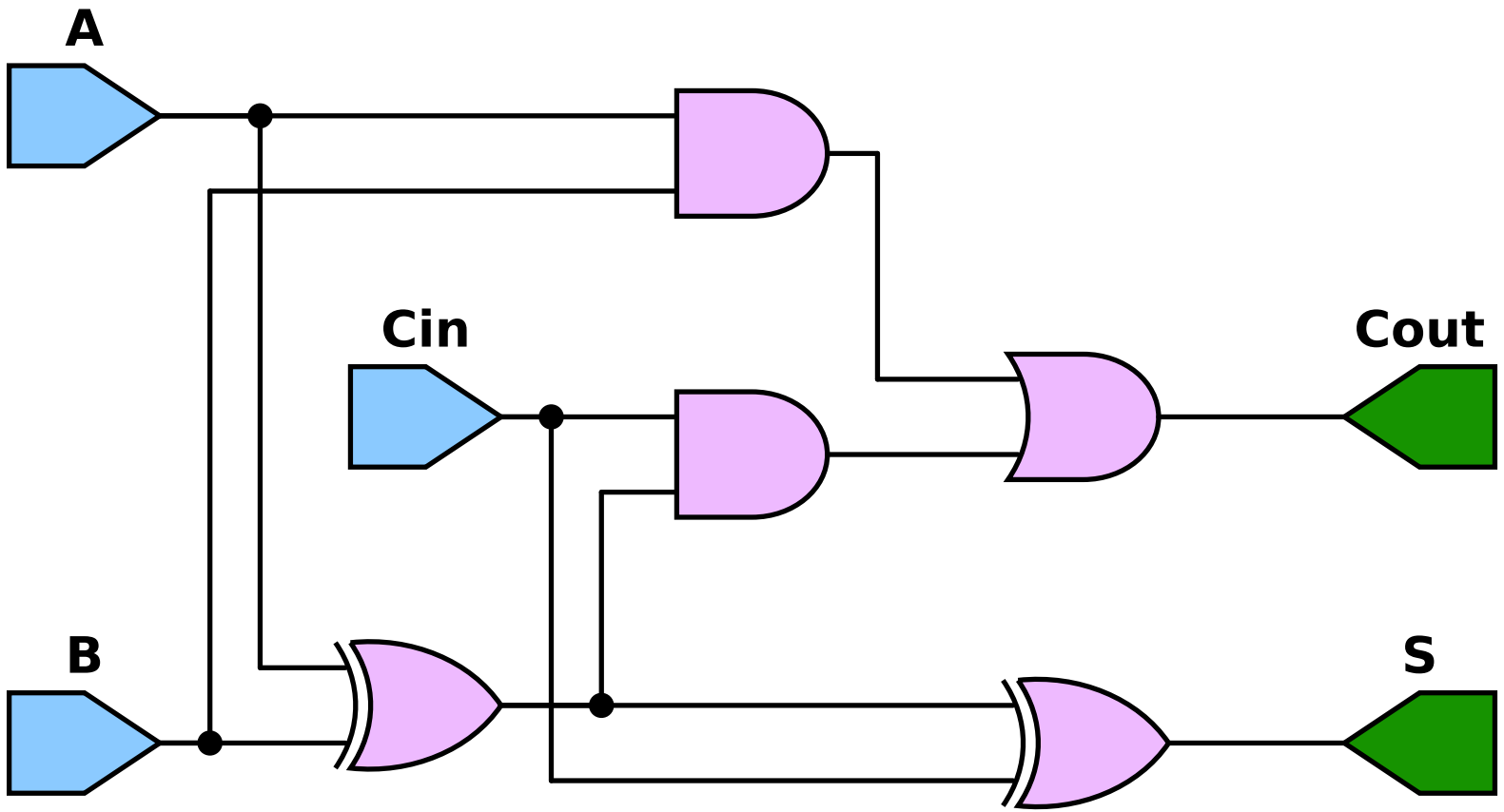

Figura 2: Esquemático do circuito somador completo.

Descrição estrutural 2

```
1  module somador_completo (
2      input A,          // Primeiro bit
3      input B,          // Segundo bit
4      input Cin,        // Carry in (entrada de transporte)
5      output S,         // Soma
6      output Cout       // Carry out (transporte de saída)
7  );
8  // Fios internos
9  wire S1, Cout1, Cout2;
10 // Primeiro meio somador: soma A e B
11 meio_somador ms1 (
12     .A(A),
13     .B(B),
14     .S(S1),
15     .Cout(Cout1)
16 );
```

Descrição estrutural 2

```
17 // Segundo meio somador: soma S1 e Cin
18 meio_somador ms2 (
19     .A(S1),
20     .B(Cin),
21     .S(S),
22     .Cout(Cout2)
23 );
24 // OR dos carries intermediários para gerar Cout
25 or (Cout, Cout1, Cout2);
26 endmodule
```

Diagrama esquemático 2

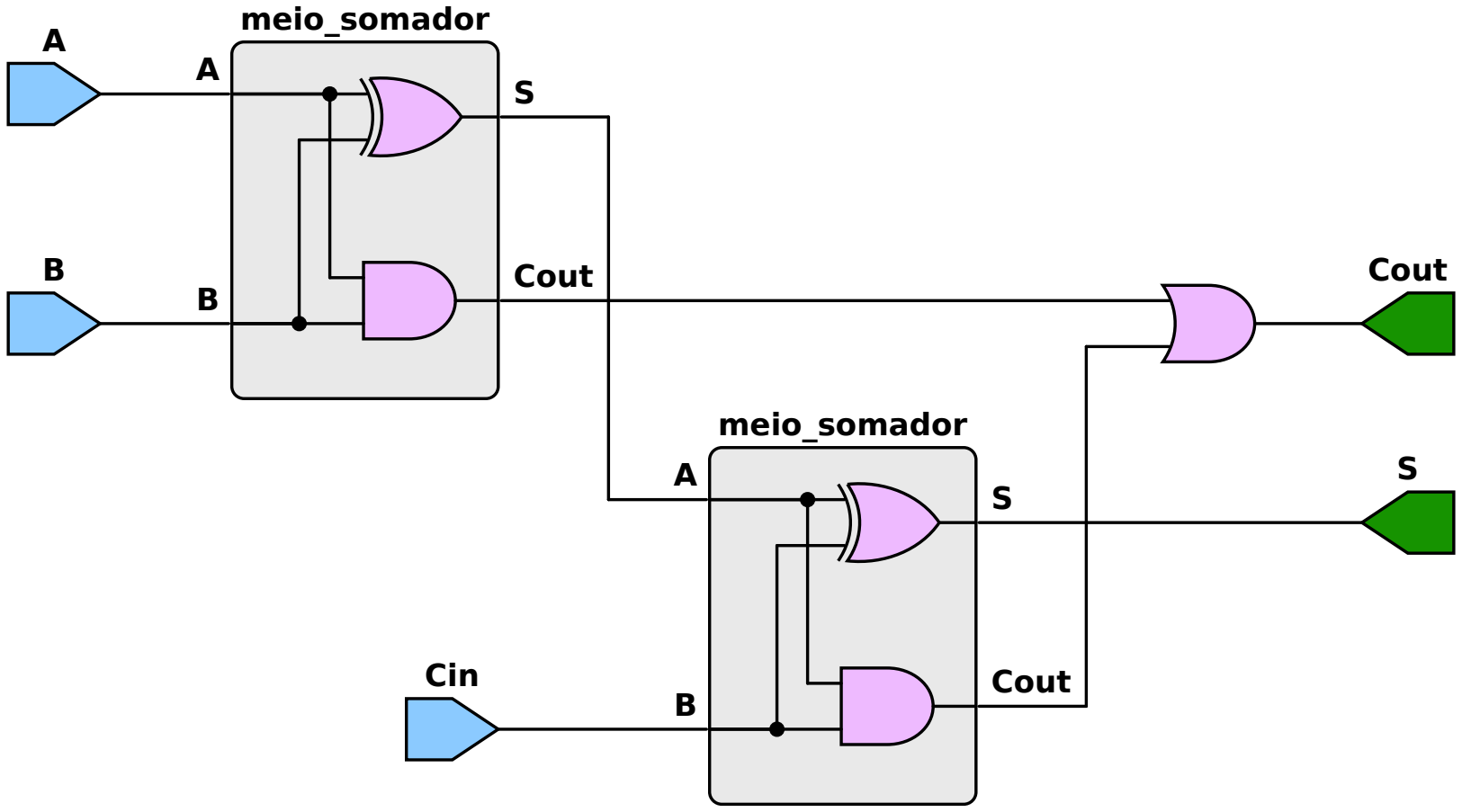

Figura 3: Esquemático do circuito somador completo.

Descrição comportamental 1

```
1  module somador_completo (
2      input A,          // Primeiro bit
3      input B,          // Segundo bit
4      input Cin,        // Carry in
5      output reg S,     // Soma
6      output reg Cout // Carry out
7  );
8  always @(*) begin
9      // Soma final
10     S = A ^ B ^ Cin; // XOR para a soma binária
11     // Carry out utilizando porta XOR
12     Cout = (A & B) | (Cin & (A ^ B));
13     // A explicação detalhada está abaixo.
14 end
15 endmodule
```

Diagrama esquemático 1

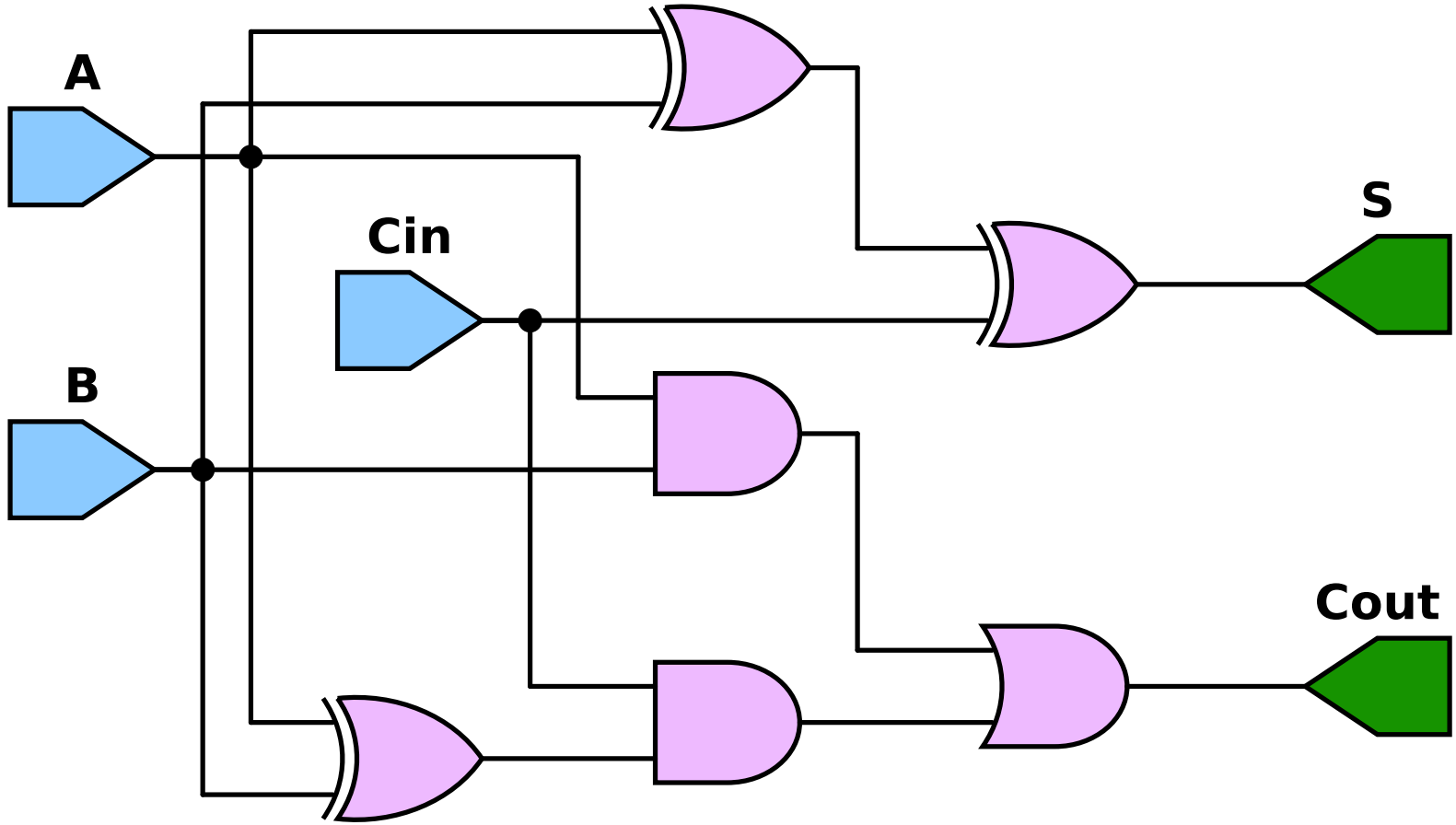

Figura 4: Esquemático do circuito somador completo.

Descrição comportamental 2

```
1  module somador_completo (
2      input A,          // Primeiro bit
3      input B,          // Segundo bit
4      input Cin,        // Carry in
5      output reg S,     // Soma
6      output reg Cout // Carry out
7  );
8  always @(*) begin
9      // Soma dos bits A, B e Cin
10     {Cout, S} = A + B + Cin;
11     // O operador "+" soma os três bits, gerando a soma (S) e o
           carry out (Cout).
12 end
13 endmodule
```

Diagrama esquemático 2

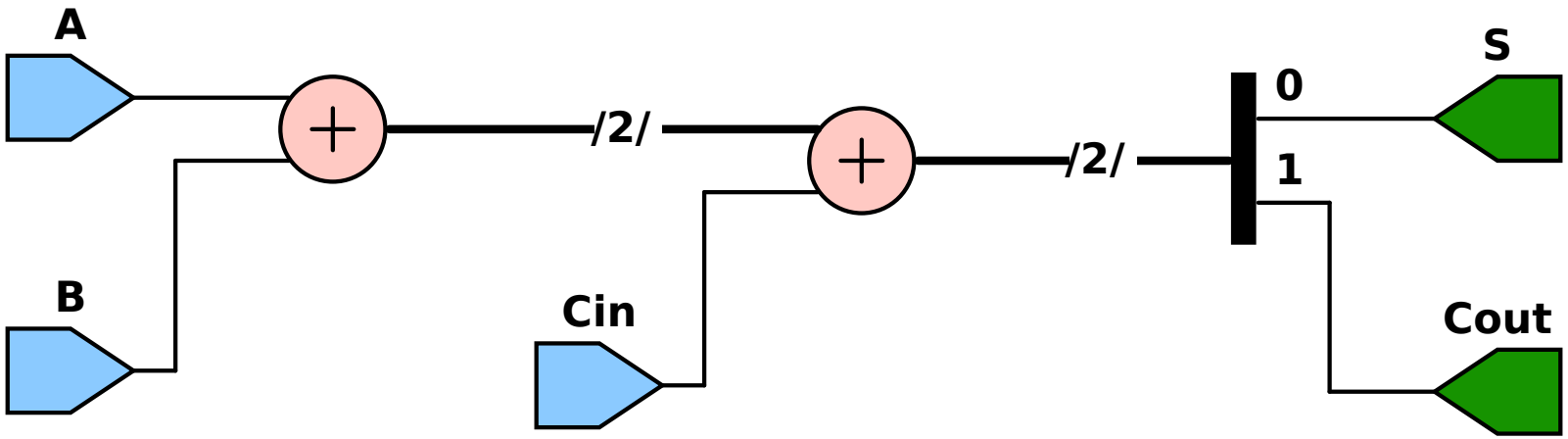

Figura 5: Esquemático do circuito somador completo.

Somador de múltiplos bits

Somador de 4-bits estrutural

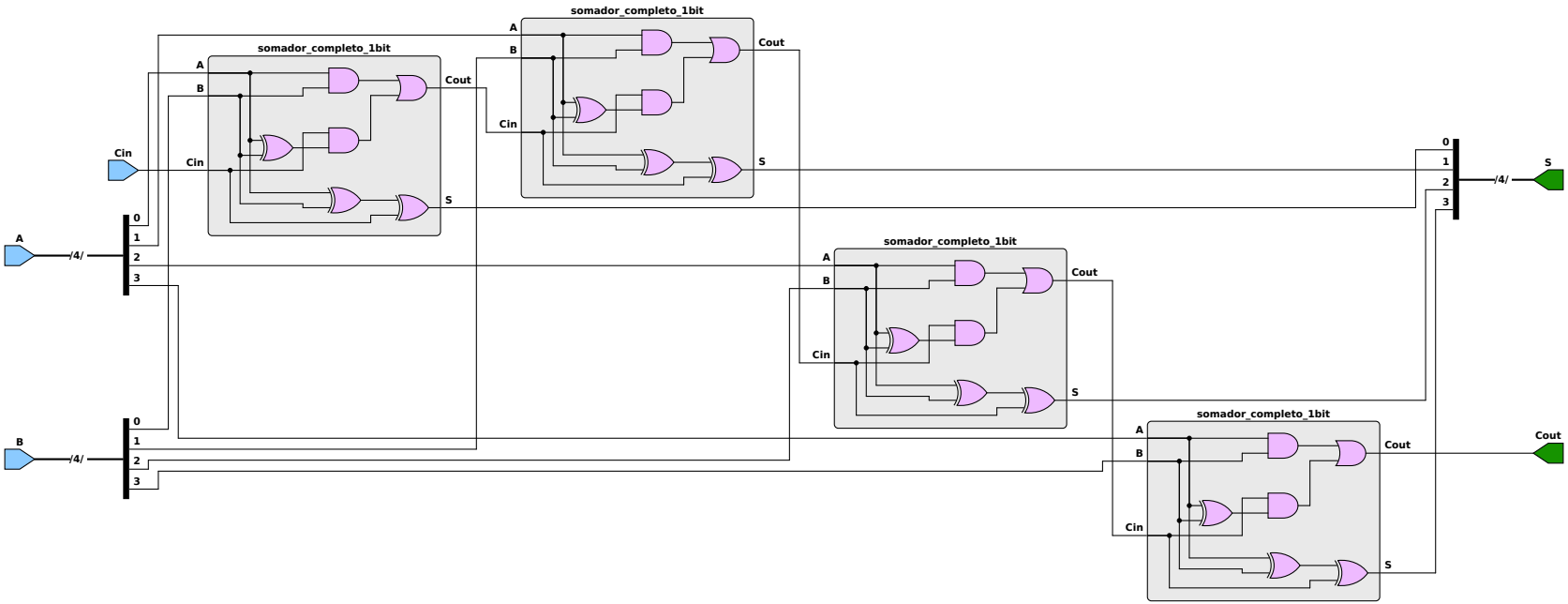

Figura 6: Somador de 4-bits usando 4 módulos somadores completos de 1-bit.

Implementação em Verilog

```
1  // Módulo do somador completo de 4 bits
2  module somador_completo_4bits (
3      input [3:0] A,        // Primeiro número de 4 bits
4      input [3:0] B,        // Segundo número de 4 bits
5      input Cin,            // Carry in
6      output [3:0] S,       // Resultado da soma (4 bits)
7      output Cout           // Carry out
8  );
9      // Sinais intermediários para os carries
10     wire C1, C2, C3;
```

Implementação em Verilog

```
11      // Instâncias do somador completo de 1 bit
12      somador_completo_1bit U0 (
13          .A(A[0]),
14          .B(B[0]),
15          .Cin(Cin),
16          .S(S[0]),
17          .Cout(C1)
18      );
19      somador_completo_1bit U1 (
20          .A(A[1]),
21          .B(B[1]),
22          .Cin(C1),
23          .S(S[1]),
24          .Cout(C2)
25      );
```

Implementação em Verilog

```
26      somador_completo_1bit U2 (
27          .A(A[2]),
28          .B(B[2]),
29          .Cin(C2),
30          .S(S[2]),
31          .Cout(C3)
32      );
33      somador_completo_1bit U3 (
34          .A(A[3]),
35          .B(B[3]),
36          .Cin(C3),
37          .S(S[3]),
38          .Cout(Cout)
39      );
40  endmodule
```

Implementação em Verilog

```
41 // Módulo do somador completo de 1 bit
42 module somador_completo_1bit (
43     input A,              // Bit do primeiro número
44     input B,              // Bit do segundo número
45     input Cin,            // Carry in
46     output S,             // Resultado da soma
47     output Cout           // Carry out
48 );
49     assign S = A ^ B ^ Cin;           // Soma (XOR dos bits)
50     assign Cout = (A & B) | ((Cin & (A ^ B))); // Carry out
51 endmodule
```

Somador de 4-bits comportamental

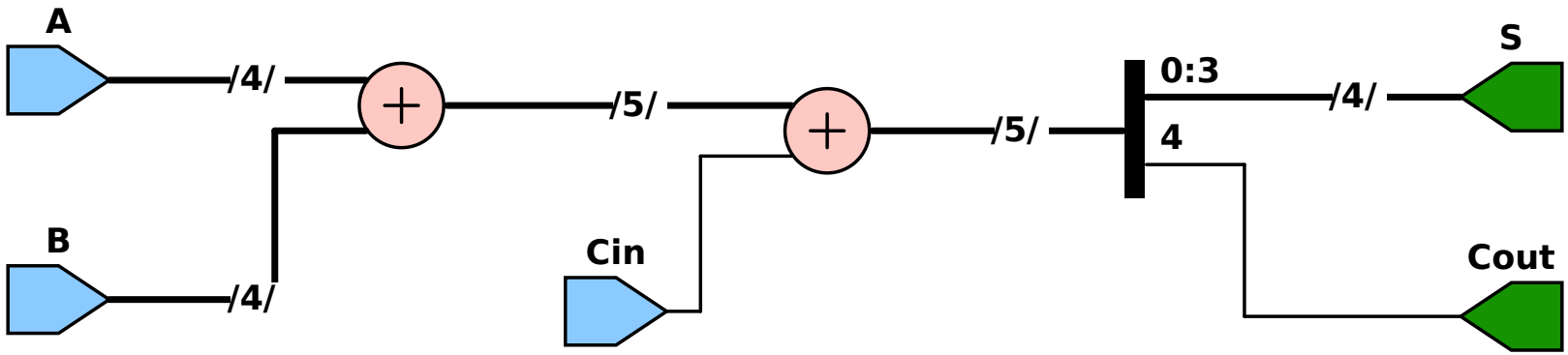

Figura 7: Somador de 4-bits comportamental.

Implementação em Verilog

```
1  module somador_completo_4bits (
2      input [3:0] A,          // Primeiro número de 4 bits
3      input [3:0] B,          // Segundo número de 4 bits
4      input Cin,              // Carry in
5      output [3:0] S,         // Resultado da soma (4 bits)
6      output Cout             // Carry out
7  );
8  reg [4:0] resultado;        // Registrador para armazenar o
       resultado da soma (5 bits: soma + carry out)
9  // Bloco procedural para realizar a soma
10 always @(*) begin
11     resultado = A + B + Cin; // Soma dos dois números de 4 bits
           mais o carry in
12 end
13 assign S = resultado[3:0];  // Os 4 bits menos significativos
       são atribuídos à saída S
14 assign Cout = resultado[4]; // O bit mais significativo é o
       carry out
15 endmodule
```

Exercícios

Exercício 1

Modifique o módulo somador de 4-bits comportamental, fornecido, afim de parametrizá-lo para ser usado para criar somadores de qualquer resolução (quantidade de bits).

Exercício 2

Para testar o somador parametrizado, crie um arquivo de *testbench* que instancie um somador de 5-bits e teste as seguintes condições:

1. Soma sem transporte (*carry*).
2. Soma com carry interno.
3. Soma com carry in.
4. Soma com overflow.
5. Soma com carry in e overflow.
6. Soma com ambos os operandos zerados.