

ARQ1_ Aula_08

Tema: Introdução à linguagem Verilog e simulação em Logisim

Preparação

Como preparação para o início das atividades, recomendam-se

- a.) leitura prévia do resumo teórico, do detalhamento na apostila e referências recomendadas
- b.) estudo e testes dos exemplos
- c.) assistir aos seguintes vídeos:

<https://www.youtube.com/watch?v=Zkkck2MovCc>

<https://www.youtube.com/watch?v=cG7wemiantQ>

<https://www.youtube.com/watch?v=YCq9L0hAWyM>

Orientação geral:

Apresentar todas as soluções em apenas um arquivo com formato texto (.txt).

Sugere-se usar como nome Guia_xx.txt, onde xx indicará o guia, exemplo Guia_01.txt.

Todos os arquivos deverão conter identificações iniciais com o nome e matrícula, no caso de programas, usar comentários.

As implementações e testes dos exemplos em Verilog (.v) fornecidos como pontos de partida, também fazem parte da atividade e deverão ter os códigos fontes entregues **separadamente**, com o código fonte, a fim de que possam ser compilados e testados. Entregar os módulos de testes.

Sugere-se usar como nomes Guia_01yy.v, onde yy indicará a questão, exemplo Guia_0101.v

As saídas de resultados, opcionalmente, poderão ser copiadas ao final do código, em comentários.

Quaisquer outras anotações, observações ou comentários poderão ser colocadas em arquivo texto (README.txt) acompanhando a entrega.

Outras formas de solução serão **opcionais**; não servirão para substituir as atividades a serem avaliadas. Caso entregues, poderão contar apenas como atividades extras.

Os programas com funções desenvolvidas em C, Java ou Python (c, .java, py),

como os modelos usados para verificação automática de testes das respostas;

caso entregues, também deverão estar em arquivos **separados**, com o código fonte, a fim de serem compilados e testados.

As execuções deverão, preferencialmente, serão testadas mediante uso de redirecionamento de entradas e saídas padrões, cujos dados/resultados deverão ser armazenados em arquivos textos.

Os resultados poderão ser anexados ao código, ao final, como comentários.

Planilhas, caso venham a ser solicitadas, deverão ser **programadas** e/ou usar funções nativas.

Serão suplementares e opcionais, e deverão ser entregues em formato texto, preferencialmente,

com colunas separadas por tabulações ou no formato (.csv), acompanhando a solução em texto.

Arquivos em formato (.pdf), fotos, cópias de tela ou soluções manuscritas também poderão ser aceitos como recursos suplementares para visualização, mas não servirão como substitutos e **não** terão validade para fins de avaliação.

Os *layouts* de circuitos deverão ser entregues no formato (.circ), identificados internamente.

Figuras exportadas pela ferramenta serão aceitas apenas como arquivos para visualização,

mas **não** terão validade para fins de avaliação. Separar versões completas (a) e simplificadas (b).

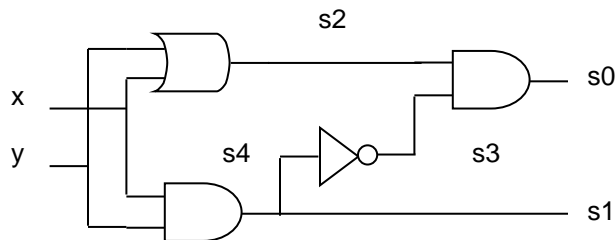
Projeto de circuitos lógicos

A síntese de circuitos lógicos pode ser executada em cinco níveis:

N	Nível	Atividades
1	Sistema	<ul style="list-style-type: none">- especificação de requisitos- particionamento
2	Algoritmo	<ul style="list-style-type: none">- especificação de comportamento- concorrência- complexidade- representação de dados
3	Arquitetura	<ul style="list-style-type: none">- representação de dados- sinais e controle- paralelismo e <i>pipelining</i>- <i>data paths</i>
4	Lógico	<ul style="list-style-type: none">- circuitos- otimizações em portas e transistores- mapeamento em bibliotecas
5	Físico	<ul style="list-style-type: none">- otimização lógica- planejamento de <i>layout</i>- fabricação e encapsulamento

Aplicações aritméticas de expressões e circuitos lógicos

Dado o circuito lógico:



As relações abaixo descrevem os sinais de saída em função dos sinais de entrada:

$$s0 = s2 \cdot s3 = s2 \cdot s4' = (x + y) \cdot (x \cdot y)' \quad (a)$$

$$s1 = s4 = x \cdot y \quad (b)$$

$$s2 = x + y$$

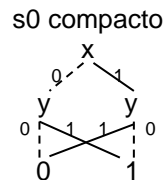
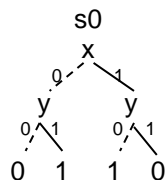
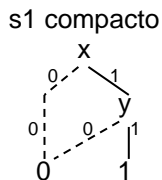
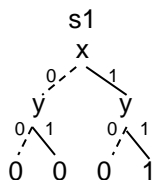
$$s3 = s4'$$

$$s4 = x \cdot y$$

A partir das relações (a) e (b) pode-se construir a tabela-verdade:

x y	x+y	s1=x•y	(x•y)'	s0=(x+y)•(x•y)'
0 0	0	0	1	0
0 1	1	0	1	1
1 0	1	0	1	1
1 1	1	1	0	0

e os diagramas (or árvores) de decisão (*BDDs – Binary Decision Diagrams*):



O resultado também poderá ser apresentado de outra forma:

x + y =	s1	s0
0 + 0 =	0	0
0 + 1 =	0	1
1 + 0 =	0	1
1 + 1 =	1	0

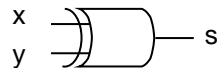
ou seja, o circuito é o responsável por uma soma binária de dois dígitos binários (bits), e é chamado de circuito de **meia-soma**. (*half-adder*). Para se operar três ou mais bits, esses serão combinados em cascata para formar um circuito de **soma-completa** (descrito mais adiante).

A equação da saída (s0) do circuito poderá ser simplificada pelas propriedades da álgebra:

$(x + y) \cdot (x \cdot y)'$	Propriedades
$(x + y) \cdot (x' + y')$	- De Morgan
$x \cdot x' + y \cdot x' + x \cdot y' + y \cdot y'$	- Distributiva
$0 + y \cdot x' + x \cdot y' + 0$	- Complementar
$(0 + y \cdot x') + (x \cdot y' + 0)$	- Associativa
$y \cdot x' + x \cdot y'$	- Identidade
$x' \cdot y + x \cdot y'$	- Comutativa

A relação obtida é equivalente à porta lógica - **OU exclusivo** (XOR) - cuja representação encontra-se abaixo:

Porta OU-exclusivo (XOR)



Antivalência (Diferença)

x	y	$x \oplus y$	mintermos	MAXTERMOS	N
0	0	0	0	$(X' + Y')$	0
0	1	1	$(x' \cdot y)$	1	1
1	0	1	$(x \cdot y')$	1	2
1	1	0	0	$(X + Y)$	3

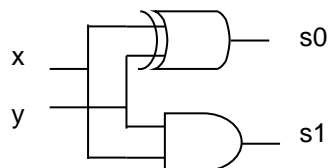
Essa relação poderá ser definida por uma soma dos produtos (SoP):

$$0 + (x' \cdot y) + (x \cdot y') + 0 = (x' \cdot y) + (x \cdot y') = \sum(1, 2)$$

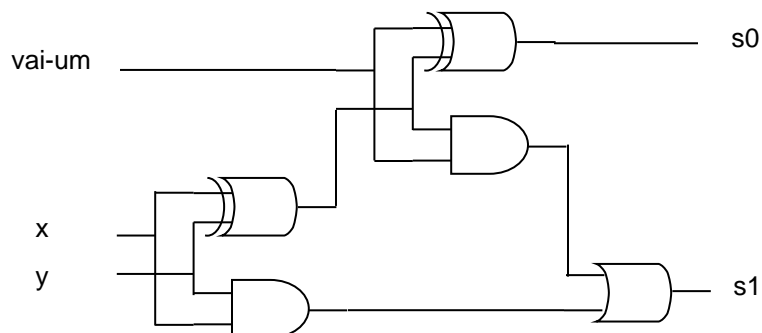
ou definida por um produto das somas (PoS):

$$(X' + Y') \cdot 1 \cdot 1 \cdot (X + Y) = (X' \cdot X) + (X' \cdot Y) + (Y' \cdot X) + (Y' \cdot Y) = (X' \cdot Y) + (Y' \cdot X) = \prod(0, 3)$$

O circuito de **meia-soma** poderá ser refeito com a porta **XOR** :



O circuito de **soma-completa** (*full-adder*), para somar três bits, também poderá ser feito com essa porta:



As relações expressas pelo circuito de **soma-completa** poderão ser encontradas abaixo:

vai-um	+ x	+ y	s1	s0	mintermos	MAXTERMOS	N
0	0	0	0	0	$(v' \cdot x' \cdot y')$	$(V + X + Y)$	0
0	0	1	0	1	$(v' \cdot x' \cdot y)$	$(V + X + Y')$	1
0	1	0	0	1	$(v' \cdot x \cdot y')$	$(V + X' + Y)$	2
0	1	1	1	0	$(v' \cdot x \cdot y)$	$(V + X' + Y')$	3
1	0	0	0	1	$(v \cdot x' \cdot y')$	$(V' + X + Y)$	4
1	0	1	1	0	$(v \cdot x' \cdot y)$	$(V' + X + Y')$	5
1	1	0	1	0	$(v \cdot x \cdot y')$	$(V' + X' + Y)$	6
1	1	1	1	1	$(v \cdot x \cdot y)$	$(V' + X' + Y')$	7

A relação (**s0**) poderá ser definida por uma soma dos produtos (SoP):

$$(v' \cdot x' \cdot y) + (v' \cdot x \cdot y') + (v \cdot x' \cdot y') + (v \cdot x \cdot y) = \sum(1, 2, 4, 7)$$

ou definida por um produto das somas (PoS):

$$(V + X + Y) \cdot (V + X' + Y') \cdot (V' + X + Y') \cdot (V' + X' + Y) = \prod(0, 3, 5, 6)$$

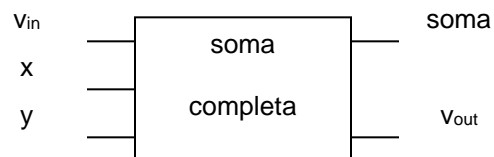
A relação (**s1**) também poderá ser definida por uma soma dos produtos (SoP):

$$(v' \cdot x \cdot y) + (v \cdot x' \cdot y) + (v \cdot x \cdot y') + (v \cdot x \cdot y) = \sum(3, 5, 6, 7)$$

ou definida por um produto das somas (PoS):

$$(V + X + Y) \cdot (V + X + Y') \cdot (V + X' + Y) \cdot (V' + X + Y) = \prod(0, 1, 2, 4)$$

O diagrama de blocos abaixo resume esse circuito:



A diferença entre dois **bits** também pode ser projetada de modo semelhante.

x - y	d	v	mintermos	MAXTERMOS	N
0 0	0	0	$(x' \cdot y')$	$(X + Y)$	0
0 1	1	1	$(x' \cdot y)$	$(X + Y')$	1
1 0	1	0	$(x \cdot y')$	$(X' + Y)$	2
1 1	0	0	$(x \cdot y)$	$(X' + Y')$	3

A diferença (d) poderá ser definida por uma soma dos produtos (SoP):

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

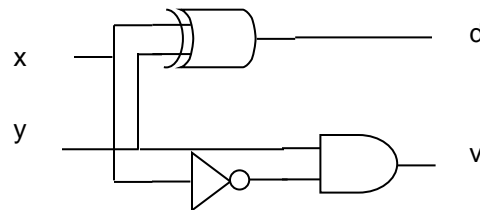
ou definida por um produto das somas (PoS):

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

A necessidade de empréstimo (“vem-um”) poderá ser expressa pela relação:

$$(x' \cdot y) = v$$

O circuito de **meia-diferença** (*half-difference*) é mostrado a seguir:



O circuito de **diferença-completa** (x-y-empréstimo) (*full-difference*) terá a definição abaixo:

x - y - vem-um	s1	s0	mintermos	MAXTERMOS	N
0 0 0	0	0	$(x' \cdot y' \cdot v')$	$(X + Y + V)$	0
0 0 1	1	1	$(x' \cdot y' \cdot v)$	$(X + Y + V')$	1
0 1 0	1	1	$(x' \cdot y \cdot v')$	$(X + Y' + V)$	2
0 1 1	1	0	$(x' \cdot y \cdot v)$	$(X + Y' + V')$	3
1 0 0	0	1	$(x \cdot y' \cdot v')$	$(X' + Y + V)$	4
1 0 1	0	0	$(x \cdot y' \cdot v)$	$(X' + Y + V')$	5
1 1 0	0	0	$(x \cdot y \cdot v')$	$(X' + Y' + V)$	6
1 1 1	1	1	$(x \cdot y \cdot v)$	$(X' + Y' + V')$	7

A relação (**s0**) também poderá ser definida pela soma dos produtos (SoP):

$$(x' \cdot y' \cdot v) + (x' \cdot y \cdot v') + (x \cdot y' \cdot v') + (x \cdot y \cdot v) = \sum m(1, 2, 4, 7)$$

ou ainda definida pelo produto das somas (PoS):

$$(X + Y + V) \cdot (X + Y' + V') \cdot (X' + Y + V') \cdot (X' + Y' + V) = \prod M(0, 3, 5, 6)$$

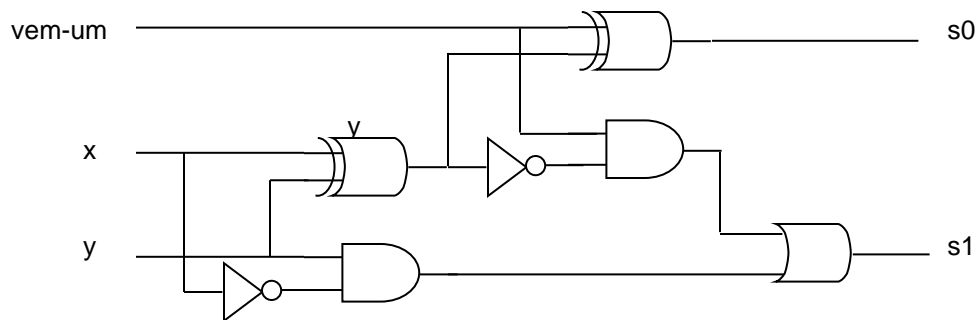
A relação (**s1**) poderá ser definida pela soma dos produtos (SoP):

$$(x' \cdot y' \cdot v) + (x' \cdot y \cdot v') + (x \cdot y' \cdot v) + (x \cdot y \cdot v') = \sum m(1, 2, 3, 7)$$

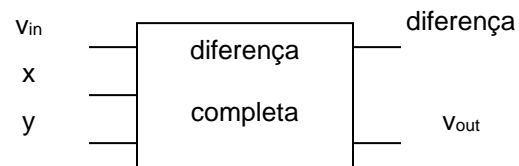
ou definida por um produto das somas (PoS):

$$(X + Y + V) \cdot (X' + Y + V) \cdot (X' + Y' + V') \cdot (X' + Y' + V) = \prod M(0, 4, 5, 6)$$

O circuito equivalente é mostrado abaixo.



O diagrama de blocos descrito a seguir resume esse circuito:



Atividade: Projeto de unidade lógica e aritmética

Orientação geral:

Apresentar todas as soluções em apenas um arquivo com formato texto (.txt). Sugere-se usar como nome Guia_xx.txt, onde xx indicará o guia, exemplo Guia_01.txt. Todos os arquivos deverão conter identificações iniciais com o nome e matrícula, no caso de programas, usar comentários. As implementações e testes dos exemplos em Verilog (.v) fornecidos como pontos de partida, também fazem parte da atividade e deverão ter os códigos fontes entregues **separadamente**, com o código fonte, a fim de que possam ser compilados e testados. Sugere-se usar como nomes Guia_01yy.v, onde yy indicará a questão, exemplo Guia_0101.v. As saídas de resultados, opcionalmente, poderão ser copiadas ao final do código, em comentários. Quaisquer outras anotações, observações ou comentários poderão ser colocadas em arquivo texto (README.txt) acompanhando a entrega.

Outras formas de solução serão **opcionais**; não servirão para substituir as atividades a serem avaliadas. Caso entregues, poderão contar apenas como atividades extras. Os programas com funções desenvolvidas em C, Java ou Python (c, .java, py), como os modelos usados para verificação automática de testes das respostas; caso entregues, também deverão estar em arquivos **separados**, com o código fonte, a fim de serem compilados e testados. As execuções deverão, preferencialmente, serão testadas mediante uso de redirecionamento de entradas e saídas padrões, cujos dados/resultados deverão ser armazenados em arquivos textos. Os resultados poderão ser anexados ao código, ao final, como comentários. Planilhas, caso venham a ser solicitadas, deverão ser **programadas** e/ou usar funções nativas. Serão suplementares e opcionais, e deverão ser entregues em formato texto, preferencialmente, com colunas separadas por tabulações ou no formato (.csv), acompanhando a solução em texto. Arquivos em formato (.pdf), fotos, cópias de tela ou soluções manuscritas também poderão ser aceitos como recursos suplementares para visualização, mas não servirão como substitutos e **não** terão validade para fins de avaliação. Os *layouts* de circuitos deverão ser entregues no formato (.circ), identificados internamente. Figuras exportadas pela ferramenta serão aceitas apenas como arquivos para visualização, mas **não** terão validade para fins de avaliação. Separar versões completas (a) e simplificadas (b).

01.) Projetar e descrever em Verilog, usando **portas nativas**, uma unidade aritmética (AU) com um somador completo, utilizando o modelo compacto para a “meia-soma”, para operandos de 06 bits (sinal=1+amplitude=4). O nome do arquivo deverá ser Guia_0801.v, e poderá seguir o modelo descrito abaixo. Incluir previsão de testes. Simular o módulo no Logisim e apresentar layout do circuito e subcircuitos.

Exemplo:

“vai-um” final

↓

S₄₁ S₃₁ S₂₁ S₀₁ 0 ← “vai-um” inicial arbitrário

a₃ a₂ a₁ a₀ +

b₃ b₂ b₁ b₀

S₀₄ S₀₃ S₀₂ S₀₁ S₀₀

// -----

// Guia_0800 - FULL ADDER

// Nome: xxx yyy zzz

// Matricula: 999999

// -----

// -----

// half adder

// -----

```
module halfAdder (output s1,
                  output s0,
                  input  a,
                  input  b);
```

// descrever por portas

xor XOR1 (s0, a, b);

and AND1 (s1, a, b);

endmodule // halfAdder

// -----

// full adder

// -----

```

module fullAdder ( output s1,
                  output s0,
                  input  a,
                  input  b,
                  input  carryIn );

// descrever por portas e/ou modulos
// (valores arbitrarios escolhidos apenas para exemplo)
    not NOT1 ( s1, a ); // valor arbitrario
    not NOT2 ( s0, b ); // valor arbitrario

endmodule // fullAdder

module test_fullAdder;
// ----- definir dados
    reg  [3:0] x;
    reg  [3:0] y;
    wire [3:0] carry; // "vai-um"
    wire [4:0] soma;

// halfAdder HAO ( carry[0], soma[0], x[0], y[0] );
    fullAdder FA0 ( carry[0], soma[0], x[0], y[0], 1'b0 );

// ----- parte principal
    initial begin
        $display("Guia_0800 - xxx yyy zzz - 999999");
        $display("Test ALU's full adder");

// projetar testes do somador completo
    end

endmodule // test_fullAdder

```

- 02.) Projetar e descrever em Verilog, usando **portas nativas**, uma unidade aritmética (AU) com um subtrator completo, utilizando o modelo compacto para a “meia-diferença”, para operandos de 06 bits (sinal=1+amplitude=4). O nome do arquivo deverá ser Guia_0802.v, e poderá seguir o modelo descrito anteriormente. Incluir previsão de testes. Simular o módulo no Logisim e apresentar *layout* do circuito e subcircuitos.

Exemplo:

$$\begin{array}{r}
 \text{“vem-um” final} \\
 \downarrow \\
 \begin{array}{r}
 s_{41} \ s_{31} \ s_{21} \ s_{01} \ 0 \ \leftarrow \text{“vem-um” inicial arbitrário} \\
 a_3 \ a_2 \ a_1 \ a_0 \ + \\
 b_3 \ b_2 \ b_1 \ b_0 \\
 \hline
 s_{04} \ s_{03} \ s_{02} \ s_{01} \ s_{00}
 \end{array}
 \end{array}$$

- 03.) Projetar e descrever em Verilog, usando portas nativas, uma unidade lógica (LU) com um comparador para igualdade, para 06 bits (sinal=1+amplitude=4). O nome do arquivo deverá ser Guia_0803.v, e poderá seguir o modelo descrito anteriormente. Incluir previsão de testes. Simular o módulo no Logisim e apresentar *layout* do circuito e subcircuitos.

Exemplo:

$$\begin{array}{r}
 x \ y \ s = \text{igual} \\
 0 \ 0 \ 1 \\
 0 \ 1 \ 0 \\
 1 \ 0 \ 0 \\
 1 \ 1 \ 1 \\
 \\
 \begin{array}{r}
 a_3 \ a_2 \ a_1 \ a_0 \ \sim \wedge \leftarrow \text{“xnor”} \\
 b_3 \ b_2 \ b_1 \ b_0 \\
 \hline
 s_3 \ s_2 \ s_1 \ s_0
 \end{array} \\
 \\
 s = f (s_3, s_2, s_1, s_0); \quad // \text{ todos iguais a 1}
 \end{array}$$

- 04.) Projetar e descrever em Verilog, usando **portas nativas**, uma unidade lógica (LU) com um comparador para desigualdade, para 06 bits (sinal=1+amplitude=4).
 O nome do arquivo deverá ser Guia_0804.v,
 e poderá seguir o modelo descrito anteriormente.
 Incluir previsão de testes.
 Simular o módulo no Logisim e
 apresentar *layout* do circuito e subcircuitos.

Exemplo:

```
x y s = desigualdade
0 0 0
0 1 1
1 0 1
1 1 0
```

```
a3 a2 a1 a0 ^ ← "xor"
b3 b2 b1 b0
```

```
s3 s2 s1 s0
```

```
s = f ( s3, s2, s1, s0 ); // pelo menos algum diferente de 0
```

- 05.) Projetar e descrever em Verilog, usando **portas nativas**, uma unidade lógica (LU) com um módulo para calcular o complemento de 2 de um valor binário com 06 bits.
 O nome do arquivo deverá ser Guia_0805.v,
 e poderá seguir o modelo descrito anteriormente.
 Incluir previsão de testes.
 Simular o módulo no Logisim e
 apresentar *layout* do circuito e subcircuitos.
 DICA: Construir um subcircuito para calcular o complemento de 1 e
 usar um somador completo.

Extras

- 06.) Projetar e descrever em Verilog, usando **portas nativas**, uma unidade aritmética (AU) com um somador algébrico (somador completo com uma das entradas invertida, caso seja escolhida a subtração, conforme seleção pela entrada **carryIn** do primeiro subcircuito (soma=**carryIn**=0; subtração=**carryIn**=1), para calcular simultaneamente a igualdade e a desigualdade, para 05 bits (sinal=1+amplitude=5), e selecionar o resultado por uma segunda entrada extra (chave). O nome do arquivo deverá ser Guia_0806.v, e poderá seguir o modelo descrito anteriormente. Incluir previsão de testes. Simular o módulo no Logisim e apresentar *layout* do circuito e subcircuitos. DICA: Usar o subcircuito para calcular o complemento de 1 condicionado à seleção.
- 07.) Projetar e descrever em Verilog, usando **portas nativas**, uma unidade lógica (LU) com um comparador para calcular simultaneamente a igualdade (=0) ou a desigualdade (=1), para 05 bits (sinal=1+amplitude=5), e selecionar o resultado por uma entrada extra (chave). O nome do arquivo deverá ser Guia_0807.v, e poderá seguir o modelo descrito anteriormente. Incluir previsão de testes. Simular o módulo no Logisim e apresentar *layout* do circuito e subcircuitos. DICA: Usar como referência os subcircuitos 03 e 04 acima.