

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

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:

Atividades previstas como parte da avaliação

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

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

Atividades extras e opcionais

Outras formas de solução serão opcionais; não servirão para substituir as atividades a serem avaliadas. Se entregues, contarão apenas como atividades extras.

As execuções deverão, preferencialmente, serão testadas mediante uso 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.

Os *layouts* de circuitos deverão ser entregues no formato (.circ), identificados internamente. Figuras exportadas pela ferramenta serão aceitas como arquivos para visualização, e **não** terão validade para fins de avaliação. Separar versões completas (a) e simplificadas (b).

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

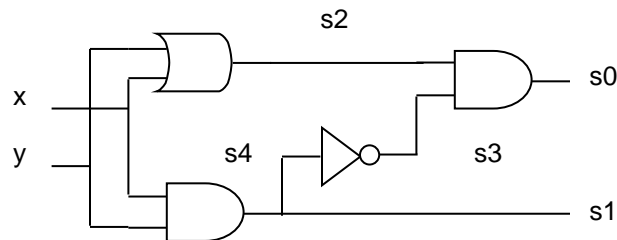
## Projeto de circuitos lógicos

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

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

## 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:

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

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

$$s2 = x + y$$

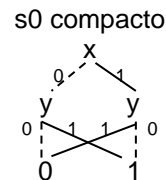
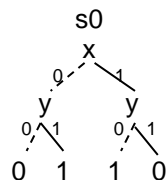
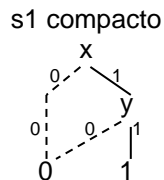
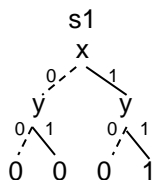
$$s3 = s4'$$

$$s4 = x \cdot y$$

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

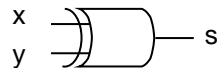
ou seja, o circuito é o responsável por uma soma binária de dois dígitos binários (bits), e é chamado de circuito de **meia-soma**. (*half-adder*). Para se operar três ou mais bits, vários desses serão combinados em cascata para formar um circuito de **soma-completa** (descrito mais adiante).

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

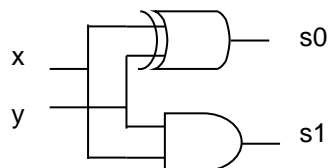
Essa relação poderá ser definida por uma soma dos produtos (SoP):

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

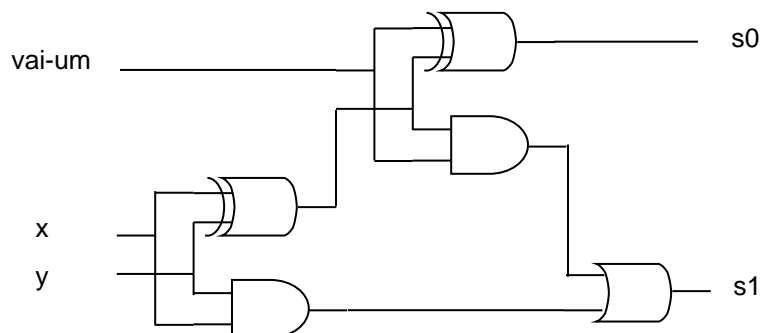
ou definida por um produto das somas (PoS):

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

O circuito de **meia-soma** poderá ser feito 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):

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

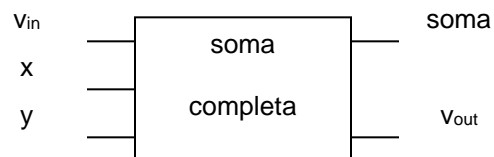
A relação ( **s1** ) também poderá ser definida por uma soma dos produtos (SoP):

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

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):

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

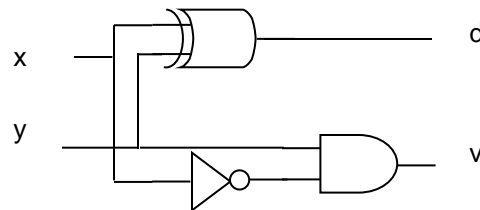
ou definida por um produto das somas (PoS):

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

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):

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

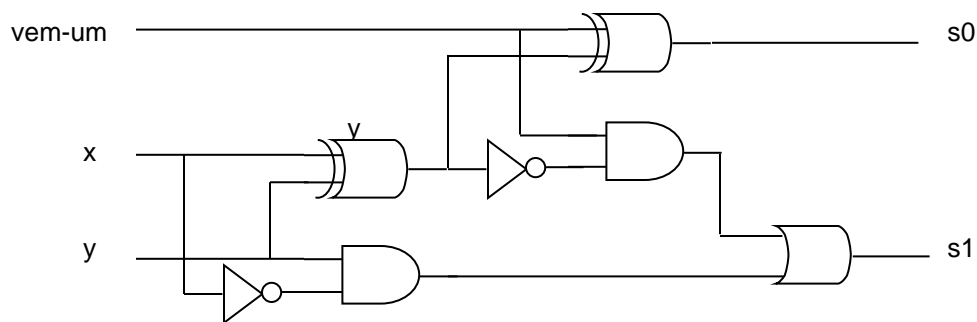
A relação ( **s1** ) poderá ser definida pela soma dos produtos (SoP):

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

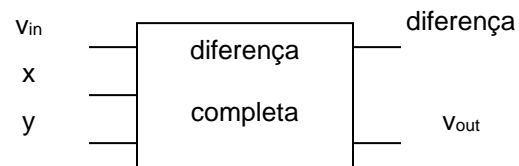
ou definida por um produto das somas (PoS):

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

O circuito equivalente é mostrado abaixo.



O diagrama de blocos descrito a seguir resume esse circuito:



## Atividade: Projeto de unidade lógica e aritmética

- 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 05 bits (sinal=1+amplitude=4). O nome do arquivo deverá ser Exemplo\_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

↓

S<sub>51</sub> S<sub>41</sub> S<sub>31</sub> S<sub>21</sub> S<sub>01</sub> 0 ← “vai-um” inicial arbitrário

a<sub>4</sub> a<sub>3</sub> a<sub>2</sub> a<sub>1</sub> a<sub>0</sub> +

b<sub>4</sub> b<sub>3</sub> b<sub>2</sub> b<sub>1</sub> b<sub>0</sub>

---

S<sub>05</sub> S<sub>04</sub> S<sub>03</sub> S<sub>02</sub> S<sub>01</sub> S<sub>00</sub>

```
// -----  
// Exemplo_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("Exemplo0800 - 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 05 bits (sinal=1+amplitude=4). O nome do arquivo deverá ser Exemplo\_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:

```

“vem-um” final
↓
S51 S41 S31 S21 S01 0 ← “vem-um” inicial arbitrário
  a4 a3 a2 a1 a0 +
  b4 b3 b2 b1 b0
-----
S05 S04 S03 S02 S01 S00

```

- 03.) Projetar e descrever em Verilog, usando portas nativas, uma unidade lógica (LU) com um comparador para igualdade, para 05 bits (sinal=1+amplitude=5). O nome do arquivo deverá ser Exemplo\_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:

```

x y s = igual
0 0 1
0 1 0
1 0 0
1 1 1

  a4 a3 a2 a1 a0 ~^ ← “xnor”
  b4 b3 b2 b1 b0
-----
  S4 S3 S2 S1 S0

s = f ( s4 s3 s2 s1 s0 ); // todos iguais a 1

```

04.) Projetar e descrever em Verilog, usando portas nativas, uma unidade lógica (LU) com um comparador para desigualdade, para 05 bits (sinal=1+amplitude=4).

O nome do arquivo deverá ser Exemplo\_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

a<sub>4</sub> a<sub>3</sub> a<sub>2</sub> a<sub>1</sub> a<sub>0</sub>  $\wedge \leftarrow$  "xor"

b<sub>4</sub> b<sub>3</sub> b<sub>2</sub> b<sub>1</sub> b<sub>0</sub>

---

s<sub>4</sub> s<sub>3</sub> s<sub>2</sub> s<sub>1</sub> s<sub>0</sub>

s = f ( s<sub>4</sub> s<sub>3</sub>, s<sub>2</sub>, s<sub>1</sub>, s<sub>0</sub> ); // 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 05 bits.

O nome do arquivo deverá ser Exemplo\_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 06 bits (sinal=1+amplitude=5), e selecionar o resultado por uma segunda entrada extra (chave). O nome do arquivo deverá ser Exemplo\_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 06 bits (sinal=1+amplitude=5), e selecionar o resultado por uma entrada extra (chave). O nome do arquivo deverá ser Exemplo\_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.