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

ARQ1 _ Aula_05
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=TdDCWG2inoY
https://www.youtube.com/watch?v=WExVhr583vA
https://www.youtube.com/watch?v=ZgAtsWlyd5I

Orientação geral:

Atividades previstas como parte da avaliação

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 e os módulos de testes, a fim de que possam ser compilados e verificados.
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.

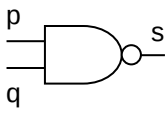
Atividades extras e opcionais

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).

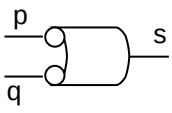
Universalidade das portas NAND e NOR

As portas NAND e NOR podem ser usadas para substituir outras funções lógicas básicas por composições equivalentes, como mostrado a seguir.

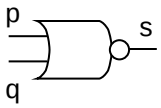
Porta NAND



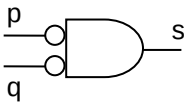
Conjugado NAND



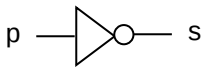
Porta NOR



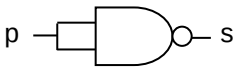
Conjugado NOR



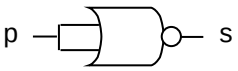
Porta NÃO (NOT)



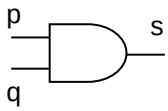
Porta NÃO (NOT)



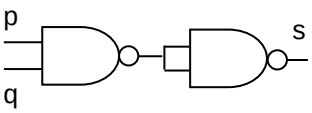
Porta NÃO (NOT)



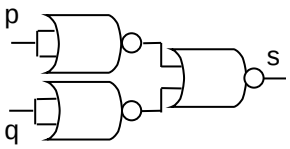
Porta AND



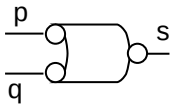
Porta AND



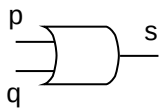
Porta AND



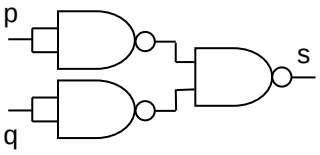
Conjugado AND



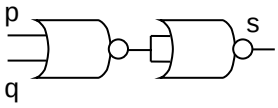
Porta OR



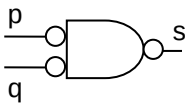
Porta OR



Porta OR



Conjugado OR



Atividades

Para os exercícios a seguir, primeiro, implementar e testar o exemplo abaixo em Verilog.

```
// -------------------------
// Guia_0500.v - GATES
// Nome: xxx yyy zzz
// Matricula: 999999
// -------------------------

// -------------------------
//  f5_gate
//  m  a  b  s
//  0  0  0  0
//  1  0  1  1 <- a'.b
//  2  1  0  0
//  3  1  1  0
//
// -------------------------
module f5a ( output s,
                    input  a,
                    input  b );
// definir dado local
   wire not_a;
// descrever por portas
   not NOT1  ( not_a, a );
   and AND1 ( s, not_a, b );
endmodule // f5a

// -------------------------
//  f5_gate
//  m  a  b  s
//  0  0  0  0
//  1  0  1  1 <- a'.b
//  2  1  0  0
//  3  1  1  0
//
// -------------------------
module f5b ( output s,
                    input  a,
                    input  b );
// descrever por expressao
   assign s = ~a & b;
endmodule // f5b
```

```
module test_f5;
// ------------------------- definir dados
      reg  x;
      reg  y;
      wire a, b;

      f5a moduloA ( a, x, y );
      f5b moduloB ( b, x, y );

// ------------------------- parte principal

  initial
  begin : main
        $display("Guia_0500 - xxx yyy zzz - 999999");
        $display("Test module");
        $display("   x     y    a    b");

      // projetar testes do modulo
        $monitor("%4b %4b %4b %4b", x, y, a, b);
           x = 1'b0;  y = 1'b0;
  #1       x = 1'b0;  y = 1'b1;
  #1       x = 1'b1;  y = 1'b0;
  #1       x = 1'b1;  y = 1'b1;

  end

endmodule // test_f5
```

Exercícios

01.) Projetar e descrever em Verilog, usando apenas portas nativas **nor**
um módulo equivalente à expressão (~a & ~b).
O nome do arquivo deverá ser Guia_0501.v,
e poderá seguir o modelo descrito anteriormente.
Incluir previsão de testes.
Simular o módulo no Logisim e
apresentar *layout* do circuito e subcircuitos.

02.) Projetar e descrever em Verilog, usando apenas portas nativas **nand**
um módulo equivalente à expressão (~a | ~b).
O nome do arquivo deverá ser Guia_0502.v,
e poderá seguir o modelo descrito anteriormente.
Incluir previsão de testes.
Simular o módulo no Logisim e
apresentar *layout* do circuito e subcircuitos.

03.) Projetar e descrever em Verilog, usando apenas portas nativas **nor**
módulo equivalente à expressão ~(~a | b).
O nome do arquivo deverá ser Guia_0503.v,
e poderá seguir o modelo descrito anteriormente.
Incluir previsão de testes.
Simular o módulo no Logisim e
apresentar *layout* do circuito e subcircuitos.

04.) Projetar e descrever em Verilog, usando apenas portas nativas **nand**
módulo equivalente à expressão ~(a & b).
O nome do arquivo deverá ser Guia_0504.v,
e poderá seguir o modelo descrito anteriormente.
Incluir previsão de testes.
Simular os módulos no Logisim e
apresentar *layout* dos circuitos e subcircuitos.

05.) Projetar e descrever em Verilog, usando apenas portas nativas **nor**
um módulo equivalente à disjunção exclusiva (~(a ^ b) = a **xnor** b).
O nome do arquivo deverá ser Guia_0505.v,
e poderá seguir o modelo descrito anteriormente.
Incluir previsão de testes.
Simular o módulo no Logisim e
apresentar *layout* do circuito e subcircuitos.

06.) Projetar e descrever em Verilog, usando apenas portas nativas **nand**
um módulo equivalente à negação da disjunção exclusiva ((a ^ b) = a **xor** b).
O nome do arquivo deverá ser Guia_0506.v,
e poderá seguir o modelo descrito anteriormente.
Incluir previsão de testes.
Simular o módulo no Logisim e
apresentar *layout* do circuito e subcircuitos.

Extras

07.) Projetar e descrever em Verilog, usando apenas portas nativas **nor**
um módulo equivalente à expressão ~ (a ^ ~b).
O nome do arquivo deverá ser Guia_0507.v,
e poderá seguir o modelo descrito anteriormente.
Incluir previsão de testes.
Simular o módulo no Logisim e
apresentar *layout* do circuito e subcircuitos.

08.) Projetar e descrever em Verilog, usando apenas portas nativas **nand**
um módulo equivalente à expressão (~a | b).
O nome do arquivo deverá ser Guia_ 0508.v,
e poderá seguir o modelo descrito anteriormente.
Incluir previsão de testes.
Simular o módulo no Logisim e
apresentar *layout* do circuito e subcircuitos.