

```
// -----
// Exemplo0001 - buffer
// Nome: xxx yyy zzz
// Matricula: 999999
// -----

// -----
// -- buffer
// -----

module buffer (output s, input p);
assign s = p; // criar vinculo permanente
           // (dependencia)
endmodule // buffer

// -----
// -- test buffer
// -----

module testbuffer;
// ----- dados locais
    reg a; // definir registrador
           // (variavel independente)
    wire s; // definir conexao (fio)
           // (variavel dependente )
// ----- instancia
    buffer BF1 (s, a);
// ----- preparacao
    initial begin:start
        a=0; // valor inicial
    end
// ----- parte principal
    initial begin:main
        // execucao unitaria
        $display("Exemplo 0001 - xxx yyy zzz - 999999");
        $display("Test buffer:");
        $display("\t time\t a = s");
        // execucao permanente
        $monitor("%d\t %b = %b", $time, a, s);
        // apos uma unidade de tempo
        // mudar valor do registrador para 0
        #1 a=1;
        // apos duas unidades de tempo
        // mudar valor do registrador para 1
        #2 a=0;
    end

endmodule // testbuffer
```

02.) Compilar o programa.

Se houver erros, identificar individualmente a referência para a linha onde ocorrem.
Consultar atentamente o modelo acima, próximo à linha indicada (e também linhas anteriores),
e editar as modificações necessárias.
Compilar novamente e proceder assim até que todos os erros tenham sido resolvidos.
Se não houver erros, seguir para o próximo passo.

SUGESTÃO: Para se acostumar ao tratamento de erros,
registrar a mensagem de erro (como comentário)
e o que foi feito para resolvê-lo.

03.) Executar o programa.

Observar as saídas.

04.) Copiar a versão atual do programa para outra (nova) – Exemplo0002.v.

05.) Editar mudanças no nome do programa e versão,
conforme as indicações a seguir,
tomando o cuidado de modificar todas as referências,
inclusive as presentes em comentários.
Incluir na documentação complementar as alterações feitas,
acrescentar indicações de mudança de versão e
prever novos testes.

```
// -----  
// Exemplo0002 - NOT  
// Nome: xxx yyy zzz  
// Matricula: 999999  
// -----  
  
// -----  
// -- not gate  
// -----  
module notgate (output s,  
                 input p);  
    assign s = ~p;  
endmodule // notgate  
  
// -----  
// -- test not gate  
// -----  
module testnotgate;  
// ----- dados locais  
    reg a; // definir registrador  
           // (variavel independente)  
    wire s; // definir conexao (fio)  
           // (variavel dependente )  
// ----- instancia  
    notgate NOT1 (s, a);  
// ----- preparacao  
    initial begin:start  
        a=0; // valor inicial  
    end
```

```
// ----- parte principal
initial begin
    $display("Exemplo0002 - xxx yyy zzz - 999999");
    $display("Test NOT gate");
    $display("\n~a = s\n");
    a=0;
#1 $display("~%b = %b", a, s);
    a=1;
#1 $display("~%b = %b", a, s);
end

endmodule // testnotgate
```

06.) Compilar o programa novamente.

Se houver erros, resolvê-los e compilar novamente, até que todos tenham sido resolvidos.

Se não houver erros, seguir para o próximo passo.

07.) Executar o programa.

Observar as saídas.

08.) Copiar a versão atual do programa para outra (nova) – Exemplo0003.v.

```
// -----
// Exemplo0003 - AND
// Nome: xxx yyy zzz
// Matricula: 999999
// -----

// -----
// -- and gate
// -----
module andgate ( output s,
                 input p,
                 input q);
    assign s = p & q;
endmodule // andgate

// -----
// -- test and gate
// -----
module testandgate;
// ----- dados locais
    reg a, b; // definir registradores
    wire s; // definir conexao (fio)
// ----- instancia
    andgate AND1 (s, a, b);
// ----- preparacao
initial begin:start
    // atribuicao simultanea
    // dos valores iniciais
    a=0; b=0;
end
```

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

initial begin
    $display("Exemplo0003 - xxx yyy zzz - 999999");
    $display("Test AND gate");
    $display("\na & b = s\n");
    a=0; b=0;
#1 $display("%b & %b = %b", a, b, s);
    a=0; b=1;
#1 $display("%b & %b = %b", a, b, s);
    a=1; b=0;
#1 $display("%b & %b = %b", a, b, s);
    a=1; b=1;
#1 $display("%b & %b = %b", a, b, s);
end

endmodule // testandgate
```

09.) Compilar o programa novamente.

Se houver erros, resolvê-los e compilar novamente, até que todos tenham sido resolvidos.

Se não houver erros, seguir para o próximo passo.

10.) Executar o programa.

Observar as saídas.

11.) Copiar a versão atual do programa para outra (nova) – Exemplo0004.v.

```
// -----
// Exemplo0004 - OR
// Nome: xxx yyy zzz
// Matricula: 999999
// -----

// -----
// -- or gate
// -----
module orgate ( output s,
                input  p, q);
    assign s = p | q;
endmodule // orgate

// -----
// -- test or gate
// -----
module testorgate;
// ----- dados locais
    reg  a, b; // definir registradores
    wire s;    // definir conexao (fio)
// ----- instancia
    orgate OR1 (s, a, b);
// ----- preparacao
    initial begin:start
        // atribuicao simultanea
        // dos valores iniciais
        a=0; b=0;
    end
```

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

initial begin
    $display("Exemplo0004 - xxx yyy zzz - 999999");
    $display("Test OR gate");
    $display("\na & b = s\n");
    a=0; b=0;
#1 $display("%b & %b = %b", a, b, s);
    a=0; b=1;
#1 $display("%b & %b = %b", a, b, s);
    a=1; b=0;
#1 $display("%b & %b = %b", a, b, s);
    a=1; b=1;
#1 $display("%b & %b = %b", a, b, s);
end

endmodule // testorgate
```

12.) Compilar o programa novamente.

Se houver erros, resolvê-los e compilar novamente, até que todos tenham sido resolvidos.

Se não houver erros, seguir para o próximo passo.

13.) Executar o programa.

Observar as saídas.

14.) Copiar a versão atual do programa para outra (nova) – Exemplo0005.v.

```
// -----
// Exemplo0005 - xor
// Nome: xxx yyy zzz
// Matricula: 999999
// -----

// -----
// -- xor gate
// -----

module xorgate (output [0:3] s,
                input  [0:3] p,
                input  [0:3] q);
    assign s = p ^ q;
endmodule // xor

// -----
// -- test xorgate
// -----
module testxorgate;
// ----- dados locais
    reg [0:3] a,b; // definir registrador
    wire [0:3] s; // definir conexao (fio)
// ----- instancia
    xorgate XOR1 (s, a, b);
```

```

// ----- preparacao
initial begin:start
    a=4'b0011; // 4 valores binarios
    b=4'b0101; // 4 valores binarios
end

// ----- parte principal
initial begin:main
    $display("Exemplo0005 - xxx yyy zzz - 999999");
    $display("Test xor gate");
    $display("\n a ^ b = s\n");
    $monitor("%b ^ %b = %b", a, b, s);
#1 a=0; b=0; // valores decimais
#1 a=4'b0010; b=4'b0001; // valores binarios
#1 a=4'd1; b=3; // decimal e decimal
#1 a=4'o5; b=2; // octal e decimal
#1 a=4'hA; b=3; // hexadecimal e decimal
#1 a=4'h9; b=4'o3; // hexadecimal e octal

end

endmodule // testxorgate

```

15.) Compilar o programa novamente.

Se houver erros, resolvê-los e compilar novamente, até que todos tenham sido resolvidos.

Se não houver erros, seguir para o próximo passo.

16.) Executar o programa.

Observar as saídas.

Exercícios:

DICAS GERAIS: Consultar a referência para Verilog na apostila para outros exemplos.

Para cada um dos enunciados abaixo,
definir o módulo correspondente e os procedimentos de testes.

- 01.) Construir a tabela-verdade para a porta NAND com 2 entradas.
Obs.: Usar operador ($\sim(a\&b)$) na definição do módulo.
Usar `$display ()`.
- 02.) Construir a tabela_verdade para a porta NOR com 2 entradas.
Obs.: Usar operador ($\sim(a|b)$) na definição do módulo.
Usar `$monitor ()`.
- 03.) Construir a tabela_verdade para a porta XNOR com 2 entradas.
Obs.: Usar operador ($\sim(a^b)$) na definição do módulo.
- 04.) Construir a tabela_verdade para a porta NAND com 2 entradas.
Obs.: Usar na definição do módulo a propriedade de Morgan.
- 05.) Construir a tabela_verdade para a porta NOR com 2 entradas.
Obs.: Usar na definição do módulo a propriedade de Morgan.
- 06.) Construir a tabela_verdade para a porta XOR com 2 entradas.
Obs.: Usar a expressão: $a'b+a b'$ na definição do módulo.
- 07.) Construir a tabela_verdade para a porta XNOR com 2 entradas.
Obs.: Usar a expressão: $a b+a'b'$ na definição do módulo.
- 08.) Construir a tabela_verdade para a porta AND com 3 entradas.
Obs.: Usar expressão na definição do módulo.
- 09.) Construir a tabela_verdade para a porta AND com 3 entradas.
Obs.: Criar um módulo que use outros módulos AND de 2 entradas.
- 10.) Construir a tabela_verdade para a porta OR com 3 entradas.
Obs.: Criar um módulo que use outros módulos OR de 2 entradas.

Extras

- 11.) Definir e testar um módulo com uma expressão envolvendo portas de um circuito capaz de receber um byte e retornar 1, se todos os seus bits forem iguais a 0.
É recomendável simular o módulo no Logisim.
- 12.) Definir e testar um módulo com uma expressão envolvendo portas de um circuito capaz de receber um byte e retornar 1, se algum de seus bits for igual a 1.
É recomendável simular o módulo no Logisim.