**CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS –
CEFET/MG**

Celso França Neto (20203018570)
Diogo Emanuel Antunes Santos (20213002091)

AULA 9: Verilog HDL.

Laboratório de Arquitetura e Organização de Computadores

**BELO HORIZONTE
2022**

1) Apresente o código fonte, em Verilog, dos módulos que simulam os componentes descritos nos exemplos sugeridos no último slide da apresentação sobre Verilog disponibilizada.

Código fonte de um módulo de uma porta NAND e depois criação de uma porta AND a partir de 2 portas NAND o módulo de simulação desses componentes:

```
module NAND(in1, in2, out);
// define tipos das portas
    input in1, in2;
    output out;
// assinalamento contínuo
    assign out = ~(in1 & in2);
endmodule

module AND(in1, in2, out);
// Porta AND a partir de 2 portas NANDS
    input in1, in2;
    output out;
    wire w1;

// 2 instâncias do módulo NAND
    NAND NAND1(in1, in2, w1);
    NAND NAND2(w1, w1, out);
endmodule
```

Código fonte de um módulo que verifica qual valor é o mais recorrente entre três entradas. Possui apenas uma saída que indica esse valor, e seu módulo de teste:

```
module majority3 (x1, x2, x3, f);
    input x1, x2, x3;
    output f;
    assign f = (x1 & x2) | (x1 & x3) | (x2 & x3);
endmodule
```

2) Configure simulações que demonstrem o correto funcionamento de todos os módulos implementados. Apresente o código fonte desse(s) módulo(s) de simulação.

Módulo para simulação criação de portas AND, para teste, foram utilizadas 4 execuções com entradas diferentes e saída out1 para resultado da porta NAND e saída out2 para resultado da porta AND.

```
//módulo para simulação → não possui entrada nem saída
module test_AND;
    reg a, b;
```

```
    wire out1, out2;
    initial begin // Dados de teste
    a = 0; b = 0;
    #1 a = 1;
    #1 b = 1;
    #1 a = 0;
    end
    initial begin
    $monitor("Time=%0d a=%b b=%b out1=%b out2=%b",
    $time, a, b, out1, out2);
    end
    AND gate1(a, b, out2);
    NAND gate2(a, b, out1);
endmodule
```

Módulo para simulação de número mais recorrente, para teste, foram utilizadas 8 execuções com entradas referentes a números binários e saída com o valor que mais se repetiu neste número.

```
module test_majority;
    reg a, b, c;
    wire out;
    initial begin // Dados de teste
    a = 0; b = 0; c = 0;
    #1 a = 1;
    #1 b = 1;
    #1 c = 1;
    #1 b = 0;
    #1 a = 0;
    #1 b = 1;
    #1 c = 0;
    end
    initial begin
    $monitor("Time=%0d Numero Binário: %b%b%b +recorrente=%b",
    $time, a, b, c, out);
    end
    majority3 gate1(a, b, c, out);
endmodule
```

3) Demonstre e explique o funcionamento das simulações, usando fotos da tela (screenshots) da aplicação ModelSim em execução.

Resultado simulação no ModelSim do módulo de criação de portas AND:

```
VSIM 24> run
# Time=0 a=0 b=0 out1=1 out2=0
# Time=1 a=1 b=0 out1=1 out2=0
# Time=2 a=1 b=1 out1=0 out2=1
# Time=3 a=0 b=1 out1=1 out2=0
```

Números de a e b passam por porta NAND que gera resultado out1 e esse resultado passa duplicado por porta AND gerando resultado out2. Mostrando que efetivamente o código executou de maneira adequada a simulação.

Resultado simulação no ModelSim do módulo que testa qual número aparece mais vezes:

```
# vsim -gui work.test_majority
# Loading work.test_majority
# Loading work.majority3
VSIM 21> run
# Time=0 Numero Binário: 000 +recorrente=0
# Time=1 Numero Binário: 100 +recorrente=0
# Time=2 Numero Binário: 110 +recorrente=1
# Time=3 Numero Binário: 111 +recorrente=1
# Time=4 Numero Binário: 101 +recorrente=1
# Time=5 Numero Binário: 001 +recorrente=0
# Time=6 Numero Binário: 011 +recorrente=1
# Time=7 Numero Binário: 010 +recorrente=0
```

A cada tempo é gerado um número binário a partir de 3 registradores declarados no teste de simulação, e a saída recebe deste número binário recebe o valor que se repete mais vezes.