Escrita de Módulo

Sophia Soares Mariano

Poliware

2025

Signed e Unsigned

- É necessário tomar cuidado com a escolha dos tipos de dados usados.
- O Verilog interpreta de forma diferente os sinais dependendo do tipo declarado.
- Valores de um `reg` ou `net` são tratados como **unsigned**, a menos que a declaração use `signed`.
- Já os valores passados para `integer`, `real` ou `realtime` são tratados como **signed**, a menos que a declaração use `unsigned`.

Exemplo de Código Verilog

```
module operator_ex_sign_unsign;
    reg [3:0] a = 4'b1111;
    reg signed [3:0] b = 4'b1111;

    initial begin
        $display("a (unsigned) = %d", a);
        $display("b (signed)   = %d", b);
    end
endmodule
```

Saída do Código

```
a (unsigned) = 15
b (signed)   = -1
```

Escrita de Módulo

A escrita de um módulo em verilog consiste em definir um bloco de código que descreve um circuito digital. Pode conter entradas, saídas, e lógica interna, e pode ser instanciado dentro de outros módulos.

A escrita correta do módulo é fundamental para o funcionamento do circuito digital. Dessa forma, preste atenção aos pequenos detalhes, como ponto e vírgula, parênteses, e a ordem das entradas e saídas.

Escrita básica da headliner do módulo

```
1  module nome_do_modulo (
2      input tipo_entrada1, //define o valor como entrada
3      input tipo_entrada2,
4      output tipo_saida1, //define o valor como saída
5      output [31:0] tipo_saida2, //define saída de tamanho diferente
6      output reg tipo_saida3 //define o valor como saída do tipo reg
7  );
8  endmodule
```

Escolha o nome do módulo e seus tipos de entradas e saídas.

Escrita da lógica interna

Para a lógica interna no módulo, é possível definir variáveis do tipo "wire" ou "reg", criando variáveis internas. Além disso, é possível utilizar blocos "always" e "assign" para definir o comportamento do circuito.

Escrita da lógica interna

```
1   module nome_do_modulo (
2       input tipo_entrada1,
3       input tipo_entrada2,
4       output tipo_saida1,
5       output reg tipo_saida3
6   );
7    wire [31:0] saida_interna; // define um wire interno
8    reg [31:0] saida_reg; // define um reg interno
9
10      always @(*) begin
11          if (tipo_entrada1 > tipo_entrada2) begin
12              tipo_saida3 <= tipo_entrada1 + tipo_entrada2; // lógica combinacion
13          end
14      end
15
16      assign tipo_saida1 = tipo_entrada1 & tipo_entrada2;
17      assign saida_interna = tipo_entrada1 + tipo_entrada2; //para um wire
18
19      always @(posedge clk) begin //para um reg
20          saida_reg <= saida_interna;
21      end
22
23  endmodule
```

Obrigado!