Condicionais em Verilog

Condicionais

Os principais operadores que utilizamos para condicionais em Verilog são:

- `if`
- `else`
- `case`

if e else

`if` e `else` são utilizados para executar blocos de código com base em comparações lógicas (booleanas). A estrutura básica é:

```
if (condi  o) begin
    // c digo a ser executado se a condi  o for
       verdadeira
end else begin
    // c digo a ser executado se a condi  o for falsa
end
```

Note que, dependendo do caso, não é necessário o uso de um `else` após um `if`:

```
module exemplo(
    input entrada,
    output [3:0] saida
);
    initial begin
        saida = 4'b0000; // valor inicial da sa da
    end
    if (entrada == 1'b1) begin
        saida = 4'b0010; // se entrada for 1, saida ser
           0010
    end
endmodule
```

No caso acima, se a entrada for 1, a saída será 0010, caso contrário, a saída permanecerá com o valor inicial de 0000. Portanto, não é necessário um bloco `else`.

Agora, para o caso abaixo, o uso de `else` é necessário, pois se não, nosso output `saida` apresentaria um valor indefinido (metaestado):

```
module exemplo(
    input entrada,
    output [3:0] saida
);
    reg [3:0] saida_reg;
    always @(entrada) begin
        if (entrada == 1'b1) begin
            saida_reg = 4'b0010; // se entrada for 1, saida
                ser  0010
        end
        else begin
            saida_reg = 4'b0000; // se entrada for 0, saida
                ser  0000
        end
    end
endmodule
```

case

O `case` é utilizado para comparar uma variável com múltiplos valores possíveis, semelhante ao `switch` em outras linguagens. A estrutura básica é:

```
case (vari vel)
    valor1: begin
        // c digo para valor1
    end
    valor2: begin
        // c digo para valor2
    end
    default: begin
        // c digo para caso nenhum valor corresponda
    end
endcase
```

Um exemplo simples:

```
module exemplo(
    input [1:0] entrada,
    output reg [3:0] saida
);
    always @(entrada) begin
        case (entrada)
            2'b00: saida = 4'b0001; // se entrada for 00,
                saida ser  0001
            2'b01: saida = 4'b0010; // se entrada for 01,
                saida ser  0010
```

```
            2'b10: saida = 4'b0100; // se entrada for 10,
                saida ser  0100
            2'b11: saida = 4'b1000; // se entrada for 11,
                saida ser  1000
            default: saida = 4'b0000; // para qualquer outro
                caso, saida ser  0000
        endcase
    end
endmodule
```

Nunca se esqueça de declarar um `default`, mesmo que seu `case` tenha todas as possibilidades. Isso é uma boa prática para evitar metaestados indesejados. Por exemplo, no código acima, se a sua entrada estiver em um metaestado, a saída será 0000, evitando assim a propagação de um comportamento indefinido.