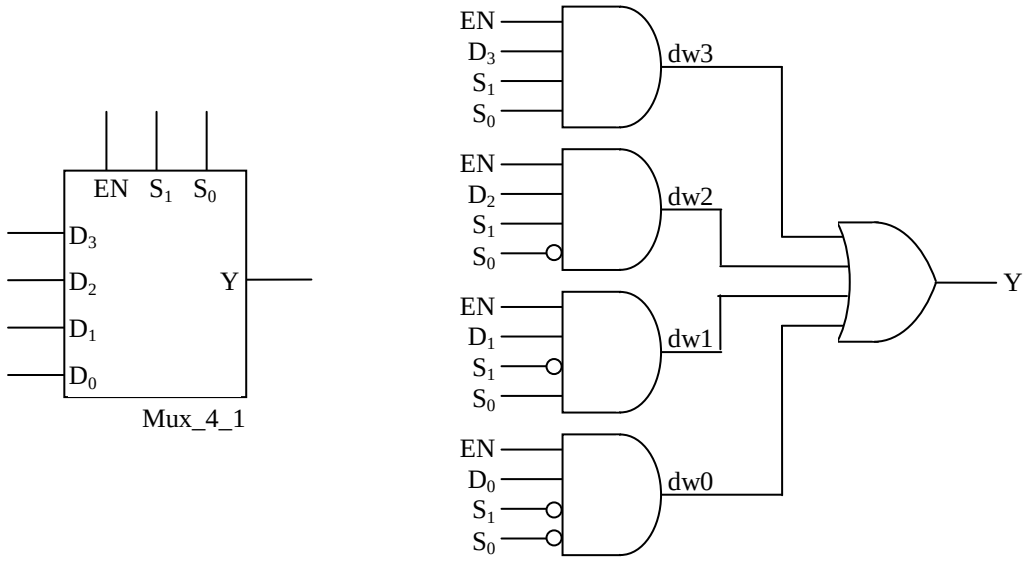
ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO
Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos

Verilog – Exemplos Básicos

Prof. Dr. Celso Massatoshi Furukawa

1 Multiplexador 4:1 com enable

Figura 1 Multiplexador Mux_4_1



1.1 Descrição não procedural (atribuições *assign*)

```
module mux_4_1_assign (

    input [3:0] D,   // Por default, inputs e outputs são wires
    input [1:0] S,
    input EN,
    output Y         // não tem virgula após o último parâmetro!
    );               // mas tem ';' após fechar o parêntesis!

    wire [3:0] dw;   // nets internas (auxiliares)

    assign dw[0] = EN & D[0] & ~S[1] & ~S[0];
    assign dw[1] = EN & D[1] & ~S[1] &  S[0];
    assign dw[2] = EN & D[2] &  S[1] & ~S[0];
    assign dw[3] = EN & D[3] &  S[1] &  S[0];

    assign Y = dw[0] | dw[1] | dw[2] | dw[3]; // Y é uma net do tipo wire

endmodule           // não tem ';' após endmodule!
```

1.2 Descrição por bloco procedural (*initial*) usando índice

```
module mux_4_1_index (
    input [3:0] D,
    input [1:0] S,
    input EN,
    output Y    // Por default, inputs e outputs são wires
    );

    reg yr;     // variável auxiliar para o bloco always

    always @ (*) begin    // '*' representa "todas as entradas"
        yr = EN & D[ S ]; // 'regs' podem ser modificadas no bloco always
    end // always

    assign Y = yr;  // nets podem ser modificadas fora do bloco always

endmodule
```

1.3 Descrição procedural usando *for*

```
module mux_4_1_for (
    input [3:0] D,
    input [1:0] S,
    input EN,
    output Y
    );

    reg di, yr;
    integer i;

    always @ (*) begin
        yr = 0;
        for (i = 0 ; i <= 3; i = i + 1) begin
            // 'i == S' resulta em 1 quando verdadeiro
            di = EN & D[i] & ( i[1:0] == S[1:0] );
            yr = yr | di;
        end // for
    end // always

    assign Y = yr;

endmodule
```

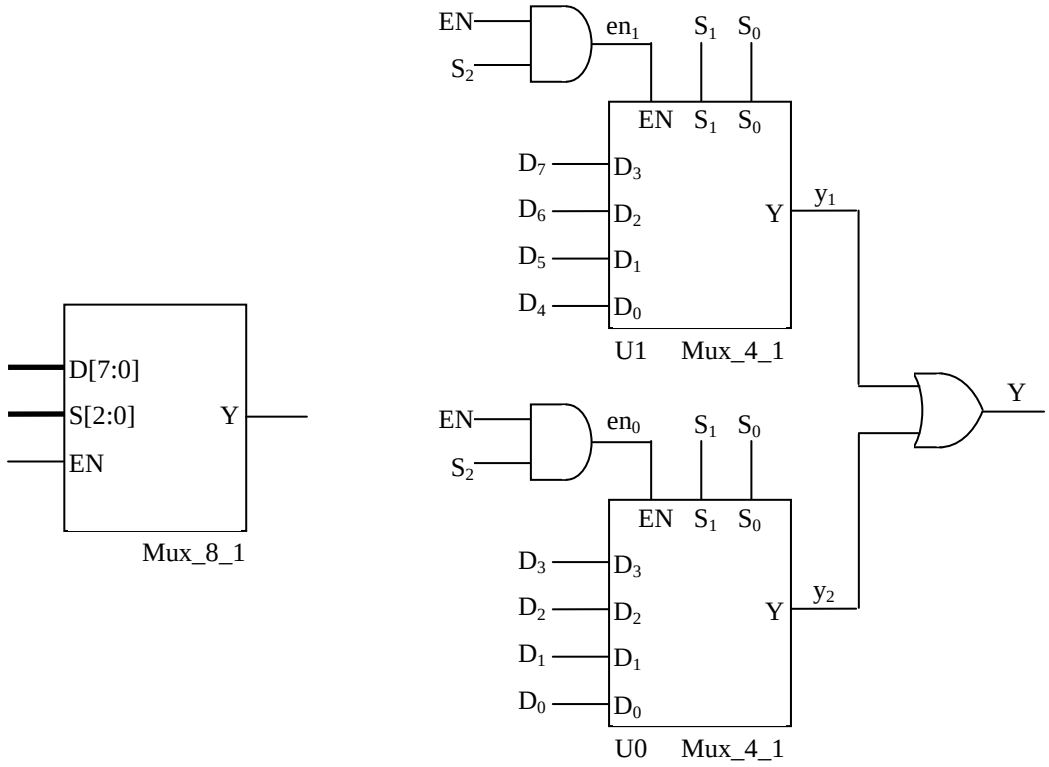
1.4 Descrição procedural usando *if-then-else* e *case*

```
module mux_4_1_case (
    input [3:0] D,
    input [1:0] S,
    input EN,
    output Y
    );

    reg yr;

    always @ (*) begin
        if (EN==0) begin
            yr = 0;
        end // if
        else begin
            case (S)
                2'b00 : yr = D[0]; // "2'b00" 2 bits, 0 em binário
                2'b01 : yr = D[1]; // "2'b01" 2 bits, 1 em binário
                2'b10 : yr = D[2]; // "2'b10" 2 bits, 2 em binário
                2'b11 : yr = D[3]; // "2'b11" 2 bits, 3 em binário
                default: yr = 0;   // opcional
            endcase;
        end // else
    end // always

    assign Y = yr;

endmodulemodule
```

2 Instanciação de um módulo em outro

Exemplo: multiplexador 8:1 com *enable* usando o módulo mux_4_1_assign

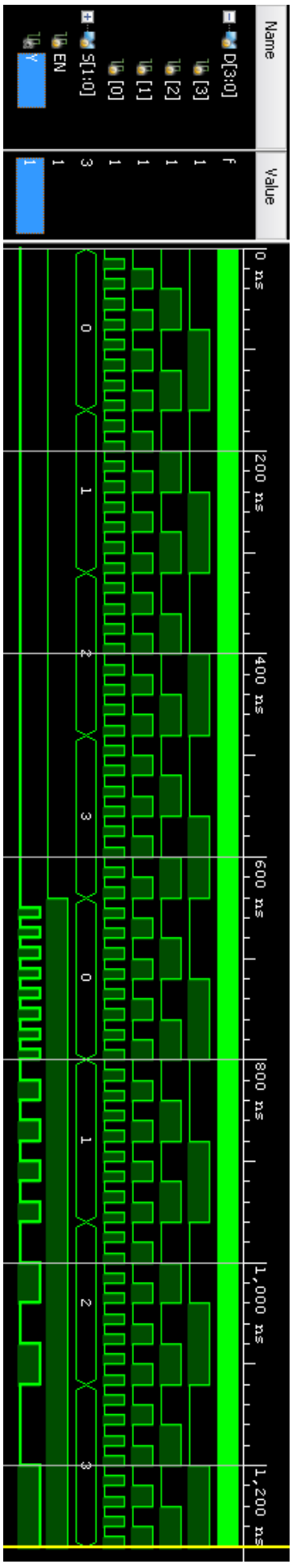
Figura 2 Multiplexador 8:1 cascateando multiplexadores 4:1



```
module mux_8_1_assign (
    input [7:0] D,
    input [2:0] S,
    input EN,
    output Y
    );

    wire y0, y1, en0, en1;  // nets internas (auxiliares)

    // instanciando os multiplexadores 4:1
    mux_4_1_assign U0 (D[3:0], S[1:0], en0, y0);
    mux_4_1_assign U1 (D[7:4], S[1:0], en1, y1);

    // conexões entre os módulos
    assign en0 = EN & ~S[2];
    assign en1 = EN &  S[2];
    assign Y = y0 | y1;

endmodule
```

3 Vetor de teste para simulação (bloco *initial*)

```
`timescale 1ns / 1ps   // unidade de tempo / precisão da simulação

module mux4_1_sim(
    );

    reg [3:0] D;
    reg [1:0] S;
    reg EN;
    wire Y;
    integer i;

    mux_4_1_assing UUT (D, S, EN, Y);

    initial begin
        // i varre todas as combinações de 7 variáveis
        for (i = 0; i < 128; i = i + 1) begin
            D[3:0] = i[3:0];
            S[1:0] = i[5:4];
            EN = i[6];
            #10;  // avança uma unidade tempo definida por timescale
        end // for
    end // initial

endmodule
```


Name
Value
D[3:0]
[3]
[2]
[1]
[0]
S[1:0]
EN
V
0 ns
200 ns
400 ns
600 ns
800 ns
1,000 ns
1,200 ns