EJEMPLOS VERILOG COMBINACIONALES

Técnicas Digitales I

Luis Eduardo Toledo

UTN

EJEMPLOS VERILOG COMBINACIONALES

Escriba una descripción comportamental en Verilog de un comparador de 4 bits con entradas A y B y una salida de seis bits Y [5:0]. El bit 5 de Y es para “igual”, bit 4 para “no igual a”, bit 3 para “mayor que”, bit 2 para “menor que”, bit 1 para “mayor o igual que”, y bit 0 para “menor o igual que”.

```
module compare (input [3: 0]   A, B,     // entrada de datos de 4-bits.
                output reg [5: 0]      Y);    // salida de 6-bits del comparador.
                                              // EQ, NE, GT, LT, GE, LE
   always @ (A or B)
    if (A==B)          Y = 6'b10_0011;    // EQ, GE, LE
    else if (A < B)   Y = 6'b01_0101;    // NE, LT, LE
    else                Y = 6'b01_1010;    // NE, GT, GE
endmodule
```

EJEMPLOS VERILOG COMBINACIONALES

Escriba un módulo en Verilog de un circuito aritmético con una variable de selección S, un acarreo de entrada Cin y dos entradas de datos de 8 bits, A y B. El circuito genera las cuatro operaciones aritméticas indicadas en la siguiente tabla:

S	Cin = 0	Cin = 1
0	F = A + B (suma)	F = A + 1 (incremento)
1	F = A – 1 (decremento)	F = A + /B + 1 (sustracción)

```
module cir_arit(input [7:0] A,
                input [7:0] B,
                input S, Cin,
                output reg [7:0] F,
                output reg cout);
 always @(*)
  case ({S, Cin})
   2'b00: {cout,F} = A + B;
   2'b01: {cout,F} = A + 8'b1;
   2'b10: {cout,F} = A - 8'b1;
   2'b11: {cout,F} = A - B;      //A + ~B + 8'b1;
  endcase
endmodule
```

EJEMPLOS VERILOG COMBINACIONALES

Escriba un módulo en Verilog de un circuito digital que realice las cuatro operaciones indicadas en la siguiente tabla:

Tabla de Funciones						
SELECCIÓN		SALIDA				
S_1	S_0	Y_3	Y_2	Y_1	Y_0	Operación
0	0	D_3	D_2	D_1	D_0	No hay corrimiento
0	1	D_2	D_1	D_0	D_3	Rotación una vez
1	0	D_1	D_0	D_3	D_2	Rotación dos veces
1	1	D_0	D_3	D_2	D_1	Rotación tres veces

```
module prob_4 (input [3:0] D,
               input [1:0] S,
               output reg [3:0] Y);
  always @(*)
    case (S)
      2'b00: Y = D;
      2'b01: Y = { D[2:0], D[3] };
      2'b10: Y = { D[1:0], D[3:2] };
      2'b11: Y = { D[0], D[3:1] };
    endcase
endmodule
```