ECEn 320

HW#14

Jae Lee

Lecture 36

|  |
| --- |
| 1.  module even detector (a , even ) ;  input [ 2 : 0 ] a ;  output even ;  wire p1 , p2 , p3 , p4 ;  assign even = (p1 | p2 ) | (p3 | p4 ) ;  assign p1 = (˜ a [ 0 ] ) & (˜ a [ 1 ] ) & (˜ a [ 2 ] ) ;  assign p2 = (˜ a [ 0 ] ) & a [ 1 ] & a [ 2 ] ;  assign p3 = a [ 0 ] & (˜ a [ 1 ] ) & a [ 2 ] ;  assign p4 = a [ 0 ] & a [ 1 ] & (˜ a [ 2 ] ) ;  endmodule |

|  |
| --- |
| 2.  module mux41(q , s e l , a , b ) ;  input[1:0] sel ;  input a, b, c, d ;  output q ;  assign q = (sel=2’b00) ? a:       (sel=2’b01) ? b:       (sel=2’b10) ? c:       d;  endmodule |

|  |
| --- |
| 3.  module decode24(q, a) ;  input[1:0] a ;  output[3:0] q ;  assign q = (a=2’b00) ? 4’b0001:       (a=2’b01) ? 4’b0010:       (a=2’b10) ? 4’b0100:       4’b1000;  endmodule |

|  |
| --- |
| 4.  module multi\_level(y, amt, a) ;  input[7:0] a ;  input[2:0] amt;  output[7:0] y ;  wire[7:0] le0\_out, le1\_out, le2\_out;  assign le0\_out = (amt[0]=1’b1) ? {a[0], a[7:1]}:     a;  assign le1\_out = (amt[1]=1’b1) ? {le0\_out[1:0], le0\_out[7:2]}:     le0\_out;  assign le2\_out = (amt[2]=1’b1) ? {le1\_out[3:0], le1\_out[7:4]}:     le1\_out;  assign y = le2\_out;    endmodule |

Lecture 37

|  |
| --- |
| 1.  module timer(clk, reset, sec, min) ;  input clk, reset;  output[5:0] sec, min;    reg[19:0] r\_reg, r\_next;  reg[5:0] s\_reg, s\_next, m\_reg, m\_next;    wire s\_en, m\_en;    always@(posedge clk or negedge reset)  begin  if(!reset)  r\_reg <= 1’d0;  s\_reg <= 1’d0;  m\_reg <= 1’d0;  else  r\_reg <= r\_next;  s\_reg <= s\_next;  m\_reg <= m\_next;  end    assign r\_next = (r\_reg=20’d99999) ? 20’d0:    r\_reg + 1;  assign s\_en = (r\_reg=20’d50000) ? 1’b1:  1’b0;  assign s\_next= (s\_reg=6’d59 && s\_en=1’b1) ? 6’d0:   (s\_en=1’b1) ? s\_reg + 1:    s\_reg;  assign m\_en = (s\_reg=6’d30) ? 1’b1:  1’b0;  assign m\_next= (m\_reg=6’d59 && m\_en=1’b1) ? 6’d0:   (m\_en=1’b1) ? m\_reg + 1:    m\_reg;    assign sec = s\_reg;  assign min = m\_reg;    endmodule |

|  |
| --- |
| 2.  module edge\_detector(clk, reset, strobe, p1) ;  input clk, reset, strobe;  output p1;    reg[1:0] state\_reg, state\_next;    parameter zero = 2’b00;  parameter edge = 2’b01;  parameter one = 2’b10;    always@(posedge clk or negedge reset)  begin  if(!reset)  state\_reg <= zero;  else  state\_reg <= state\_next;  end    always@(state\_reg or strobe)  begin  case(state\_reg)  zero:  if(strobe=1’b1)  state\_next = edge;  else  state\_next = zero;  edge:  if(strobe=1’b1)  state\_next = one;  else  state\_next = zero;  one:  if(strobe=1’b1)  state\_next = one;  else  state\_next = zero;  end case;  end  assign p1= (state\_reg=edge) ? 1’b1:  1’b0;    endmodule |