Pre-lab (10 주차) : 7-Segment, Keypad, Dot Matrix 학번 : 이름 :

문제 1) 다음 예제는 7-Segment 6 개를 이용한 시계 구현 예제이다. 아래 Code 를 설명하시오. (동작 클럭 : 1KHz)

Top file : watch.v

```
module watch(clk, seg_com, seg_data);

input clk;
output reg[7:0] seg_com;
output reg[7:0] seg_data;

integer cnt;
integer cnts;
reg[5:0] hour, min, sec ;
wire[7:0] seg_s0, seg_s1,seg_m0,seg_m1,seg_h0,seg_h1;
reg s_clk, m_clk, h_clk;

initial
begin
    hour=0;
    min=0;
    sec=0;
    cnt=0;
    cnts=0;
    h_clk=1'b0;
    s_clk=1'b0;
    m_clk=1'b0;
end

decod_0 sec0(clk,sec,seg_s0);
decod_1 sec1(clk,sec,seg_s1);
decod_0 min0(clk,min,seg_m0);
decod_1 min1(clk,min,seg_m1);
decod_0 hour0(clk,hour,seg_h0);
decod_1 hour1(clk,hour,seg_h1);

always@(posedge clk)
begin
    if(cnts == 499)
    begin
        cnts <= 0;
        s_clk <= ~(s_clk);
    end
    else
      begin
          cnts <= cnts + 1;
      end
end
```

```
always@(posedge s_clk)
begin
    if (sec >= 59)
    begin
        m_clk <= 1'b1;
        sec <= 0;
    end
    else
    begin
        sec <= sec + 1;
        m_clk <= 1'b0;
    end
end

always@(posedge m_clk)
begin
   if (min >= 59)
   begin
       h_clk <= 1'b1;
       min <= 0;
    end
    else
    begin
        min <= min + 1;
        h_clk <= 1'b0;
    end
end

always@(posedge h_clk)
begin
    if (hour >= 23)
    begin
        hour <= 0;
    end
    else
    begin
        hour <= hour +1;
    end
end

always@(posedge clk)
begin
    if(cnt==9)
    begin
        cnt<=0;
    end
    else
    begin
        cnt<=cnt+1;
    end

    case (cnt)
        0 :
        begin
            seg_com <= 8'b01111111;
```

```
            seg_data <= seg_s0;
        end
        1 :
        begin
            seg_com <= 8'b10111111;
            seg_data <= seg_s1;
        end
        2 :
        begin
            seg_com <= 8'b11011111;
            seg_data <= seg_m0;
        end
        3 :
        begin
            seg_com <= 8'b11101111;
            seg_data <= seg_m1;
        end
        4 :
        begin
            seg_com <= 8'b11110111;
            seg_data <= seg_h0;

        end
        5 :
        begin
            seg_com <= 8'b11111011;
            seg_data <= seg_h1;
        end
        6 :
        begin
            seg_com <= 8'b11110111;
            seg_data <= 8'b000000001;
        end
        7 :
        begin
            seg_com <= 8'b11011111;
            seg_data <= 8'b000000001;
        end
        8 :
        begin
            seg_com <= 8'b01111111;
            seg_data <= 8'b000000001;
        end
    endcase
end

endmodule
```

```
module decod_0(clk,num,seg_0);

input clk;
input[5:0] num;
output reg[7:0] seg_0;

always@(posedge clk)
begin
    case((num%10))
        0 : seg_0<=8'b11111100;
        1 : seg_0<=8'b01100000;
        2 : seg_0<=8'b11011010;
        3 : seg_0<=8'b11110010;
        4 : seg_0<=8'b01100110;
        5 : seg_0<=8'b10110110;
        6 : seg_0<=8'b10111110;
        7 : seg_0<=8'b11100000;
        8 : seg_0<=8'b11111110;
        9 : seg_0<=8'b11100110;
        default: seg_0<=8'b11111100;
    endcase
end

endmodule



module decod_1(clk,num,seg_1);

input clk;
input[5:0] num;
output reg[7:0] seg_1;

always@(posedge clk)
begin
    case((num/10))
        0 : seg_1<=8'b11111100;
        1 : seg_1<=8'b01100000;
        2 : seg_1<=8'b11011010;
        3 : seg_1<=8'b11110010;
        4 : seg_1<=8'b01100110;
        5 : seg_1<=8'b10110110;
        6 : seg_1<=8'b10111110;
        7 : seg_1<=8'b11100000;
        8 : seg_1<=8'b11111110;
        9 : seg_1<=8'b11100110;
        default: seg_1<=8'b11111100;
    endcase
end

endmodule
```

문제 2) 다음 예제는 입력 장치인 키패드에 대한 예제를 보여주고 있다. 이 예제는 4 X 3 의 총 12 개의 키 버튼을 가진 모듈을 통해 입력을 받는 모습을 설계하고 있다. 이러한 키패드의 구동은 스캔 방식을 통해 구동을 하고 있다. 따라서 port 에는 scan 라인과 data 라인이 따고 구성이 되어 있다. 키패드를 구동하기 위해서는 아래 설계 소스와 같이 scan 라인을 클럭에 따라 계속 '1' 의 값을 키패드로 입력을 하고 키 버튼에 의해 data 라인에서 나오는 값을 검출하여 어떤 버튼이 눌려 졌는지 알게 된다. 따라서 현재 scan 라인의 '1' 의 값을 주는 시점과 버튼이 눌려졌을 때의 data 라인의 출력 시점을 정확히 계산하여 버튼의 동작 상태를 확인하게 된다. 이렇게 확인된 결과를 led 를 통해 어떠한 버튼이 눌려 졌는지 이진법으로 표시하게 된다. 소스 Code 를 설명하시오. (동작 클럭 : 1KHz)

file : keypad.v	

```
module keypad(clk, led_data, k_s,k_d);

input clk;
output reg[7:0] led_data;
input [3:0] k_d;
output reg [2:0] k_s;

integer key_data;
wire k_stop;
assign k_stop = (k_d[3])||(k_d[2])||(k_d[1])||(k_d[0]);

always@(posedge clk)
begin
   if(k_stop==1'b1);
   else
   begin
      case(k_s)
         3'b000 : k_s <= 3'b001;
         3'b001 : k_s <= 3'b010;
         3'b010 : k_s <= 3'b100;
         3'b100 : k_s <= 3'b001;
      endcase
   end

   case(k_s)
      3'b001:
         case(k_d)
            4'b0001: key_data<=1;
            4'b0010: key_data<=4;
            4'b0100: key_data<=7;
            4'b1000: key_data<=10;
            default : key_data <= 0;
         endcase
      3'b010:
         case(k_d)
            4'b0001: key_data<=2;
            4'b0010: key_data<=5;
            4'b0100: key_data<=8;
            4'b1000: key_data<=11;
            default : key_data <= 0;
```

```
            endcase
        3'b100:
            case(k_d)
                4'b0001: key_data<=3;
                4'b0010: key_data<=6;
                4'b0100: key_data<=9;
                4'b1000: key_data<=12;
                default : key_data <= 0;
            endcase
        default : key_data <=0;
endcase

    case(key_data)
        0   : led_data<=8’b00000000;
        1   : led_data<=8’b10000000;
        2   : led_data<=8’b01000000;
        3   : led_data<=8’b11000000;
        4   : led_data<=8’b00100000;
        5   : led_data<=8’b10100000;
        6   : led_data<=8’b01100000;
        7   : led_data<=8’b11100000;
        8   : led_data<=8’b00010000;
        9   : led_data<=8’b10010000;
        10 : led_data<=8’b01010000;
        11 : led_data<=8’b11010000;
        12 : led_data<=8’b00110000;
        default : led_data<=8'b00000000;
    endcase
end

endmodule
```

문제 3) 다음 예제는 Dot-Matrix 를 이용한 코드이다. 10 X 14 의 점들을 표현할 수 있는 dot led 를 통해 square wave 가 클럭에 따라 이동하여 표현되도록 구현되었다. 소스 Code 를 설명하시오. (동작 클럭 : 1KHz)

file : dot_matrix.v

```
module dot_matrix( clk, dot_d, dot_scan);

input clk;
output reg[13:0] dot_d;
output reg[9:0] dot_scan;

reg[13:0] dot_d0, dot_d1, dot_d2, dot_d3, dot_d4, dot_d5,
dot_d6, dot_d7, dot_d8, dot_d9;
reg clk_100;
integer cnt;
integer cnt_100;

initial
begin
    clk_100=1'b0;
    cnt_100=0;
    cnt=0;
    dot_d0 = 14'b00011111111000;
    dot_d1 = 14'b00010000000000;
    dot_d2 = 14'b00010000000000;
    dot_d3 = 14'b00010000000000;
    dot_d4 = 14'b00010000000000;
    dot_d5 = 14'b00011111111000;
    dot_d6 = 14'b00000000001000;
    dot_d7 = 14'b00000000001000;
    dot_d8 = 14'b00000000001000;
    dot_d9 = 14'b00000000001000;
end



always@(posedge clk)
begin
    if(cnt_100==100)
    begin
        cnt_100<=0;
        clk_100<=~(clk_100);
    end
    else
    begin
        cnt_100<=cnt_100+1;
    end

    if(cnt==11)
    begin
        cnt<=0;
    end
```

```
else
begin
   cnt<=cnt+1;
end

case(cnt)
   0 :
   begin
      dot_scan<=10'b0000000001;
      dot_d<=dot_d0;
   end
   1 :
   begin
      dot_scan<=10'b0000000010;
      dot_d<=dot_d1;
   end
   2 :
   begin
      dot_scan<=10'b0000000100;
      dot_d<=dot_d2;
   end
   3 :
   begin
      dot_scan<=10'b0000001000;
      dot_d<=dot_d3;
   end
   4 :
   begin
      dot_scan<=10'b0000010000;
      dot_d<=dot_d4;
   end
   5 :
   begin
      dot_scan<=10'b0000100000;
      dot_d<=dot_d5;
   end
   6 :
   begin
      dot_scan<=10'b0001000000;
      dot_d<=dot_d6;
   end

  7 :
  begin
     dot_scan<=10'b0010000000;
     dot_d<=dot_d7;
  end

  8 :
  begin
     dot_scan<=10'b0100000000;
     dot_d<=dot_d8;
  end

  9 :
```

```
      begin
         dot_scan<=10'b1000000000;
         dot_d<=dot_d9;
      end

   endcase
end

always@(posedge clk_100)
begin
   dot_d0 <= dot_d1;
   dot_d1 <= dot_d2;
   dot_d2 <= dot_d3;
   dot_d3 <= dot_d4;
   dot_d4 <= dot_d5;
   dot_d5 <= dot_d6;
   dot_d6 <= dot_d7;
   dot_d7 <= dot_d8;
   dot_d8 <= dot_d9;
   dot_d9 <= dot_d0;
end

endmodule
```