<논리설계 카운터 설계>

1. register 이용한 비동기 리셋 카운터

   reg [3:0] d;

   always @ (posedge clk or negedge rst)

   if (!rst) begin q<=0; d<=0; end

   else begin d<=d+1; q<=d; end

   /*동시에 수행되므로 처음에 q에 들어가는 값은 1이 아니라 0*/

2. up-down counter logic 설계:

   제어 신호에 의하여 클럭 입력에 맞추어 출력값이 증가하거나 감소하는 카운터 → mux에 의해 출력값을 +1과 -1한 결과 중 하나를 선택하게 되어있음

   reg [3:0] d;

   always @ (posedge clk or negedge rst)

   if (!rst) begin q<=0; d<=0; end

   else begin

   if(UpDn==1) begin d<=d+1; q<=d; end

   else if (UpDn==0) begin d<=d-1; q<=d; end

   end

3. counter with enable logic 설계:

   클럭 신호의 상승 에지에서 EN신호의 논리값이 1일 때에는 카운터로 동작하며, 0일 때에는 현재의 값을 그대로 유지하는 동작 수행

   reg [3:0] d;

   always @ (posedge clk or negedge rst)

   if (!rst) begin q<=0; d<=0; end

   else begin

```
if(EN==1) begin d<=d+1; q<=d; end
else if (EN==0) begin d<=d; q<=d; end
end
```

4. 비동기 계수기 설계

1) CNT0

```
module cnt10(rst, clk, out0, clk0); //10분주하는 코드
input rst, clk;
output clk0;
output [3:0] out0;
reg out_clk;
reg [2:0] tmp2; //분주에 사용하는 변수
reg [3:0] out0, tmp; //output 계산에 사용하는 변수 (tmp는 0부터 세기 위해 쓰는 변수임)

always @ (posedge clk or negedge rst)
begin
  if(!rst)
    begin
    out0<=4'b0000;
    tmp<=4'b0000;
    end
  else
    begin
      tmp<=tmp+1;
      out0<=tmp; //15까지 세는 코드
```

```
    /*else
      begin
      if(temp == 9)
        temp <= 0;
      else begin
        temp<=temp+1;
        q <= temp;
        end
       end*/ //9까지 세는 코드
     end
  end
always @ (posedge clk or negedge rst)
begin
  if(!rst)
    begin
      out_clk<=0;
      tmp2<=0;
    end
  else
    begin
      tmp2<=tmp2+1;
      if(tmp2==4)
        tmp2<=0;
      else if(tmp2==0)
        out_clk<=!out_clk;
    end
```

```
    end
  assign clk0=out_clk;
  endmodule
```

2) async counter

```
  module async_cnt(rst, clk, out0, out1, out2, out3);
  input rst, clk;
  output [3:0] out0, out1, out2, out3;
  wire clk1, clk2, clk3;

  cnt10 u0(rst, clk, out0, clk1);
  cnt10 u1(rst, clk1, out1, clk2);
  cnt10 u2(rst, clk2, out2, clk3);
  counter u3(rst, clk3, out3);
```

5. 동기 계수기 설계:

각 계수기에 입력되는 클럭신호가 동일 → 이 클럭을 이용하여 각 계수기에서는 10분주(==4), 100분주(==49), 1000(==499)분주되는 신호를 생성해야 함 → 이 신호에 맞게 출력 발생 (각 cnt마다 하위모듈 만들어줘야함)

2번째 cnt부터는 always @ (posedge out_clk or negedge rst)

6. FPGA 실습 예제 코드

```
module top_seg7(rst, clk, seg);
  input   rst, clk;
  output[6:0] seg;
  wire[3:0] digit;
  wire out_clk;
```

```
    // 50MHz -> 1Hz   divide
    clk_dll u0(rst, clk, out_clk);
    // counter
    counter u1(rst, out_clk, digit);
    // 7-segment decoder
    seg7 u2(digit, seg);
endmodule

module seg7(digit, seg);
    input [3:0] digit;
    output [6:0] seg;
    reg [6:0] seg;
    always@(digit)
    begin
       case(digit)
         4'b0000 : seg = 7'b1000000;
         4'b0001 : seg = 7'b1111001;
         4'b0010 : seg = 7'b0100100;
         4'b0011 : seg = 7'b0110000;
         4'b0100 : seg = 7'b0011001;
         4'b0101 : seg = 7'b0010010;
         4'b0110 : seg = 7'b0000010;
         4'b0111 : seg = 7'b1111000;
         4'b1000 : seg = 7'b0000000;
         4'b1001 : seg = 7'b0010000;
       endcase
```

```
    end
endmodule


module clk_dll(rst, clk, out_clk);
    input clk, rst;
    output out_clk;
    reg out_clk;
    reg [24:0]cnt_clk;
    always@(posedge clk or negedge rst)
    begin
        if(!rst)
            begin
            out_clk = 0;
            cnt_clk = 0;
        end
        else
            begin
            cnt_clk <= cnt_clk+1;
            if(cnt_clk == 24999999) //1Hz
          // if(cnt_clk == 24) 1MHz
            cnt_clk <=0;
            else if(cnt_clk==0)
            out_clk = !out_clk;
        end
    end
endmodule
```

```
module counter(rst, in_clk, q);
   input in_clk, rst;
   output [3:0]q;
   reg [3:0]q, temp;
   always@(posedge in_clk or negedge rst)
   begin
      if(!rst)
            begin
         temp<=0;
         q<=0;
      end
      else
            begin
         if(temp == 9)
            temp <= 0;
         else begin
            temp<=temp+1;
            q <= temp;
               end
      end
endmodule

`timescale 1ns/1ns

module tb_top_seg7();
```

```
reg rst, clk;

wire [6:0] seg;

top_seg7 u0(rst, clk, seg);

initial

begin

rst=0;

#33; rst=1;

end

initial

begin

clk=0;

forever #5 clk=~clk;

end

endmodule
```