Digital Clock using Verilog Programming NEXYS-4

```
module clockdivide(clk,newclk);
 input clk;
output reg newclk=0;
integer clkc=0;
localparam onesec=100000000;
always@(posedge clk)
 begin
      if(clkc==onesec)
      begin
            newclk<=1'b1;
            clkc<=0;
 end
else
begin
      newclk<=1'b0;
      clkc<=clkc+1;
end
end
endmodule
```

// refresh the clock after 1Hz

```
module refreshclk(clk,rclk);
input clk;
 output reg rclk;
 reg [31:0]cnt=32'd0;
always@(posedge clk)
begin
     cnt=cnt+1;
     rclk=cnt[12];
end
```

```
 endmodule
```

// refresh counter

```
module refresh(clk,rcount);
input clk;
output reg [2:0]rcount=0;
always@(posedge clk)
begin
     if(rcount<5)
     rcount=rcount+1;
else
rcount=0;
end
endmodule
```

// anode control

```
module anode_control(count,anode);
input [2:0] count;
output reg [5:0] anode=0;
always@(count)
begin
case(count)
3'd0:anode=6'b111110;
3'd1:anode=6'b111101;
3'd2:anode=6'b111011;
3'd3:anode=6'b110111;
3'd4:anode=6'b101111;
3'd5:anode=6'b011111;
endcase
end
endmodule
```

// 7- segment display

```
module segment_7(bcd,seg);
input [3:0]bcd;
output reg [6:0]seg;
always @(bcd)
begin
    if(bcd==4'd0)
    seg=7'b1000000;
    else if(bcd==4'd1)
    seg=7'b1111001;
    else if(bcd==4'd2)
    seg=7'b0100100;
    else if(bcd==4'd3)
    seg=7'b0110000;
    else if(bcd==4'd4)
    seg=7'b0011001;
    else if(bcd==4'd5)
    seg=7'b0010010;
    else if(bcd==4'd6)
    seg=7'b0000010;
    else if(bcd==4'd7)
    seg=7'b1111000;
    else if(bcd==4'd8)
    seg=7'b0000000;
    else if(bcd==4'd9)
    seg=7'b0010000;
    else seg=7'b1111111;
end
endmodule
```

// binary to bcd

```
module binaryToBCD(bin,tens,ones);
input [5:0] bin;
```

```
output reg [3:0] tens,ones;
reg [5:0] bin_data=0;
always@(bin)
begin
bin_data = bin;
tens <= bin_data/10;
ones <= bin_data%10;
end
endmodule
```

// code for digital clock

```
module digitalclk(clk,rst,en,minup,hrup,h2,h1,m2,m1,s2,s1);
input clk,rst=0;
input minup,hrup,en;
output [3:0] h2,h1,m2,m1,s2,s1;
reg [5:0] hours=0,mins=0,secs=0;
always@(posedge clk or posedge rst)
begin
    if(rst)
    {hours,mins,secs}<=0;
    else if (minup==1'b1)
    if (mins==6'd59)
     mins<=0;
     else mins<=mins+1'd1;
     else if (hrup ==1'b1)
     if (hours==6'd23)
     hours<=0;
      else hours<=hours+1'd1;
      else if(en)
begin
    if(secs==6'd59)
begin
secs<=0;
```

```
    if (mins==6'd59)
begin
   mins<=0;
   if (hours==6'd23)
   hours<=0;
    else
    hours<=hours+1'd1;
end
   else mins<=mins+1'd1;
end
    else secs<=secs+1'd1;
end
end
```

// show hour, min, sec in bcd

```
binaryToBCD d1(hours,h2,h1);
binaryToBCD d2(mins,m2,m1);
binaryToBCD d3(secs,s2,s1);
endmodule
```

// bcd controller

```
module bcdControl(h2,h1,m2,m1,s2,s1,count,one_digit);
input [3:0] h2,h1,m2,m1,s2,s1;
input [2:0] count;
output reg [3:0] one_digit=0;
always@(count)
begin
        case(count)
        3'd0:one_digit=s1;
        3'd1:one_digit=s2;
        3'd2:one_digit=m1;
        3'd3:one_digit=m2;
        3'd4:one_digit=h1;
```

```
     3'd5:one_digit=h2;
endcase
end
endmodule
```

// main module

```
module main(clk,rst,en,minup,hrup,anode,seg);
input clk,rst,en,minup,hrup;
output [7:0] anode;
output [6:0] seg;
wire [3:0] h2,h1,m2,m1,s2,s1,bcd_out;
wire newclk,rclk;
wire [2:0] rcount;
wire [5:0] an;
clockdivide mm1(clk,newclk);
refreshclk mm2(clk,rclk);
refresh mm3(rclk,rcount);
anode_control mm4(rcount,an);
assign anode={2'b11,an};
digitalclk mm5(newclk,rst,en,minup,hrup,h2,h1,m2,m1,s2,s1);
bcdControl mm6(h2,h1,m2,m1,s2,s1,rcount,bcd_out);
segment_7 mm7(bcd_out,seg);
endmodule
```

// top module

```
module top_module(sw,minup,hrup,clk,an,seg);
input clk; input [1:0]sw;
input minup,hrup;
output [6:0] seg;
output [7:0] an;
main tt1(clk,sw[0],sw[1],minup,hrup,an,seg);
endmodule
```