ELD lab-GCD_Switches

Nishant Kumar Bundela

2017171

GCD_TOP

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
module gcd_top(
input clk,
input clr,
input go,
input [3:0] xin,
input [3:0] yin,
output [7:0]cathode,
output [3:0] anode
);
wire [3:0] greg;
wire xmsel;
wire ymsel;
wire xld;
wire yld;
wire gld;
wire eqflg;
wire ltflg;
wire clock;
wire gotemp;
wire gotemp1;
wire clrout;
wire clock19;
wire [3:0] ones;
```

```
wire [3:0] tens;

wire [3:0] hundreds;

wire [3:0] thousands;



clockdivid19 clk2(.inclk(clk),.outclk(clock19));

clockdivid25 clk1(.inclk(clk),.outclk(clock));

gcd_data data
(.clk(clock),.clr(clrout),.xin(xin),.yin(yin),.greg(greg),.xld(xld),.yld(yld),.gld(gld),.xmsel(xmsel),.ymsel(y
msel),.eqflg(eqflg),.ltflg(ltflg));

gcd_ctrl
control(.clk(clock),.clr(clrout),.go(gotemp1),.xld(xld),.yld(yld),.gld(gld),.xmsel(xmsel),.ymsel(ymsel),.e
qflg(eqflg),.ltflg(ltflg));

clk_pulse clk_pulse(.clk(clock),.deb_in(gotemp),.deb_out(gotemp1));

debouncer clear(.inclk(clock19),.clr(clr),.outclk(clrout));

debouncer goto(.inclk(clock19),.clr(go),.outclk(gotemp));

bin2bcd
bin2bcd(.number(greg),.ones(ones),.tens(tens),.hundreds(hundreds),.thousands(thousands));

sevenseg_all
sevseg(.clk(clk),.cathode(cathode),.anode(anode),.ones(ones),.tens(tens),.hundreds(hundreds),.thou
sands(thousands));



endmodule
```

GCD_CONTROL

```
`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

module gcd_ctrl(

input clk,

input clr,

input go,
```

```
input eqflg,
input ltflg,
output reg xmsel,
output reg ymsel,
output reg xld,
output reg yld,
output reg gld
);

reg [2:0] ps=0;
reg [2:0] ns=0;
parameter
start=3'b000,inp=3'b001,test1=3'b010,test2=3'b011,done=3'b100,update1=3'b101,update2=3'b110;

always@(posedge clk or posedge clr)
begin
   if(clr)
      ps<=start;
   else if(~clr)
      ps<=ns;
end

always@(*)
begin
   case(ps)
      start:begin
            if(go)
               ns<=inp;
            else if(~go)
               ns<=start;
            end
```

```
inp:
begin
    ns<=test1;
    xmsel<=1;
    ymsel<=1;
    xld<=1;
    yld<=1;
    gld<=0;
   end

test1:
begin
      if(eqflg)
       begin
       ns<=done;
       gld<=0;
       xld<=0;
       yld<=0;
      end
      else if(~eqflg)
      begin
      ns<=test2;
      gld<=0;
      xld<=0;
      yld<=0;
     end
end
done: begin
          if(clr)
          begin
          ns<=start;
```

```
            gld<=1;
            end
            else if(~clr)
            begin
            ns<=done;
            gld<=1;

            end
          end
     test2:  begin if(ltflg)begin ns<=update1;
//                  xmsel<=0;
//                  ymsel<=0;
//                  xld<=0;
//                  yld<=0;
//                  gld<=0;
                 end
          else if (~ltflg)begin
                ns<=update2;
//                  xmsel<=0;
//                  ymsel<=0;
//                  xld<=0;
//                  yld<=0;
//                  gld<=0;
            end
             end
     update1: begin yld<=1;
          ymsel<=0;
          ns<=test1; end
     update2: begin xld<=1;
          xmsel<=0;
          ns<=test1; end
```

```
      default :ns<=start;
    endcase
end
endmodule
```

GCD_DATAPATH

```
`timescale 1ns / 1ps
////////////////
module gcd_data(
input wire clk,
input wire clr,
input wire xmsel,
input wire ymsel,
input wire xld,
input wire yld,
input wire gld,
input wire [3:0] xin,
input wire [3:0] yin,
output reg [3:0] greg=0,
output reg eqflg=0,  //equal flag
output reg ltflg=0   //less than flag
);
reg [3:0] xreg;
reg [3:0] yreg;
wire [3:0] xmy;
reg  [3:0] x;
reg  [3:0] y;
wire [3:0] ymx;
assign xmy=xreg-yreg;
assign ymx=yreg-xreg;
```

```
always @(*)    //initial x mux {2X1}
begin
   case (xmsel)
      1'b1:x<=xin;
      1'b0:x<=xmy;
   endcase
end

always @(*)    //initial y mux (2X1)
begin
   case (ymsel)
      1'b1:y<=yin;
      1'b0:y<=ymx;
   endcase
end
always @(posedge clk or posedge clr)
begin
   if(clr==1)
   begin
      xreg<=0;
      yreg<=0;
      greg<=0;
   end
   else
   begin
      if(xld) xreg<=x;
      else if (~xld) xreg<=xreg;

      if(yld) yreg<=y;
      else if (~yld) yreg<=yreg;
```

```
        if(gld) greg<=xreg;
        else if (~gld) greg<=greg;
    end
end

always@(*)     //equality checker
begin
    if(xreg==yreg)
        eqflg<=1;
    else
        eqflg<=0;
end

always@(*)      //less than checker
begin
    if(xreg<yreg)
        ltflg=1;
    else
        ltflg=0;
end
endmodule
```

Constraint file

```
set_property PACKAGE_PIN W5 [get_ports clk]
            set_property IOSTANDARD LVCMOS33 [get_ports clk]
            create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clk]
set_property PACKAGE_PIN U18 [get_ports clr]
    set_property IOSTANDARD LVCMOS33 [get_ports clr]
set_property PACKAGE_PIN T18 [get_ports go]
set_property IOSTANDARD LVCMOS33 [get_ports go]
```

```
set_property PACKAGE_PIN V17 [get_ports {xin[0]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {xin}]
set_property PACKAGE_PIN V16 [get_ports {xin[1]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {sw[1]}]
set_property PACKAGE_PIN W16 [get_ports {xin[2]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {sw[2]}]
set_property PACKAGE_PIN W17 [get_ports {xin[3]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {sw[3]}]
set_property PACKAGE_PIN W15 [get_ports {yin[0]}]
        set_property IOSTANDARD LVCMOS33 [get_ports {yin}]
set_property PACKAGE_PIN V15 [get_ports {yin[1]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {sw[5]}]
set_property PACKAGE_PIN W14 [get_ports {yin[2]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {sw[6]}]
set_property PACKAGE_PIN W13 [get_ports {yin[3]}]
        #set_property IOSTANDARD LVCMOS33 [get_ports {sw[7]}]
set_property PACKAGE_PIN T1 [get_ports {control1}]
      set_property IOSTANDARD LVCMOS33 [get_ports {control1}]
set_property PACKAGE_PIN R2 [get_ports {control}]
set_property IOSTANDARD LVCMOS33 [get_ports {control}]

set_property PACKAGE_PIN W7 [get_ports {cathode[7]}]
      set_property IOSTANDARD LVCMOS33 [get_ports {cathode}]
   set_property PACKAGE_PIN W6 [get_ports {cathode[6]}]
      #set_property IOSTANDARD LVCMOS33 [get_ports {seg[1]}]
   set_property PACKAGE_PIN U8 [get_ports {cathode[5]}]
      #set_property IOSTANDARD LVCMOS33 [get_ports {seg[2]}]
   set_property PACKAGE_PIN V8 [get_ports {cathode[4]}]
      #set_property IOSTANDARD LVCMOS33 [get_ports {seg[3]}]
   set_property PACKAGE_PIN U5 [get_ports {cathode[3]}]
      #set_property IOSTANDARD LVCMOS33 [get_ports {seg[4]}]
```

```
set_property PACKAGE_PIN V5 [get_ports {cathode[2]}]
   #set_property IOSTANDARD LVCMOS33 [get_ports {seg[5]}]
set_property PACKAGE_PIN U7 [get_ports {cathode[1]}]
   #set_property IOSTANDARD LVCMOS33 [get_ports {seg[6]}]

set_property PACKAGE_PIN V7 [get_ports cathode[0]]
   #set_property IOSTANDARD LVCMOS33 [get_ports cathode]

set_property PACKAGE_PIN U2 [get_ports {anode[0]}]
   set_property IOSTANDARD LVCMOS33 [get_ports {anode}]
set_property PACKAGE_PIN U4 [get_ports {anode[1]}]
   #set_property IOSTANDARD LVCMOS33 [get_ports {an[1]}]
set_property PACKAGE_PIN V4 [get_ports {anode[2]}]
   #set_property IOSTANDARD LVCMOS33 [get_ports {an[2]}]
set_property PACKAGE_PIN W4 [get_ports {anode[3]}]
   #set_property IOSTANDARD LVCMOS33 [get_ports {an[3]}]
```