Lab 3

COUNTER:

```
module counter(
  input clock_in,
  input clr,
  output reg [width-1:0] count
  );

  parameter width = 5;
  //reg [width-1:0] count ;

  always@(posedge clock_in)
  begin
     if (clr!=1)
        count = count + 1;
     else
        count=0;
  end

endmodule
```

FREQUENCY DIVIDER:

```
module freq_divider(
  input clock_in,
  output clock_out
  );
  parameter width=25;

  reg [width-1:0] count;

  always @ (posedge clock_in)
  begin
     count<=count+1;
  end
```

```
   assign clock_out=count[width-1];
endmodule
```

DEBOUNCER:

```
module debouncing(
  input clock_in,
  input clr_in,
  output clr_out
  );
  reg D1,D2,D3;
      always@(posedge clock_in)
      begin
          D1 <= clr_in;
          D2 <= D1;
          D3 <= D2;
      end

      assign clr_out = D1 && D2 && D3 ;

endmodule
```

TOP FREQUENCY :

```
module top_freq(
   input clock_in,
   output  clock_out_original,
   output  clock_out_slow
   );
   assign clock_out_original = clock_in;
   frequency_divider #25 f1 (clock_in,clock_out_slow);

endmodule
```

TOP MODULE:

```
module top_module(
   input clk,
   input clr,
   input [1:0] sel_time,
   output reg [4:0] out
   );
   wire [4:0]count1,count2,count3,count4;
   reg pa;

   divider_one object(clk,out_clk);

   freq_divider #(0) object1(out_clk,oc);
   debouncing object2(out_clk,clr,deb_out);
   counter object3(oc,deb_out,count1);


   freq_divider #(1) o1(out_clk,oc1);
   debouncing o2(out_clk,clr,deb_out);
   counter o3(oc1,deb_out,count2);


   freq_divider #(3) ob1(out_clk,oc2);
   debouncing ob2(out_clk,clr,deb_out);
   counter ob3(oc2,deb_out,count3);


   freq_divider #(7) obc1(out_clk,oc3);
   debouncing obc2(out_clk,clr,deb_out);
   counter obc3(oc3,deb_out,count4);

   always @(*)
   begin
   if (sel_time == 2'b00)
      begin
      out <= count1;
      end
   else if(sel_time == 2'b01)
      begin
```

```
        out<= count2;
        end
    else if(sel_time == 2'b10)
        begin
        out <= count3;
        end
    else if (sel_time == 2'b11)
        begin
        out <= count4;
        end
    else
        out <= count1;

    end

endmodule
```

1 HZ FREQUENCY:

```
module divider_one(
    input clk,
    output reg outclk
    );
    reg [26:0] count;
    always @(posedge clk)
        begin
            count <= count + 1;
            if (count == 50000000)
                begin
                    count <= 0;
                    outclk <= ~outclk;
                end
        end
endmodule
```

CONSTRAINT FILE:

```
set_property PACKAGE_PIN Y9 [get_ports {clk}]
set_property IOSTANDARD LVCMOS33 [get_ports {clk}]
set_property PACKAGE_PIN T18 [get_ports {clr}]
set_property IOSTANDARD LVCMOS33 [get_ports {clr}]
set_property PACKAGE_PIN T22 [get_ports {out[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {out[0]}]
set_property PACKAGE_PIN T21 [get_ports {out[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {out[1]}]
set_property PACKAGE_PIN U22 [get_ports {out[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {out[2]}]
set_property PACKAGE_PIN U21 [get_ports {out[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {out[3]}]
set_property PACKAGE_PIN F22 [get_ports {sel_time[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sel_time[0]}]
set_property PACKAGE_PIN G22 [get_ports {sel_time[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {sel_time[1]}]
```