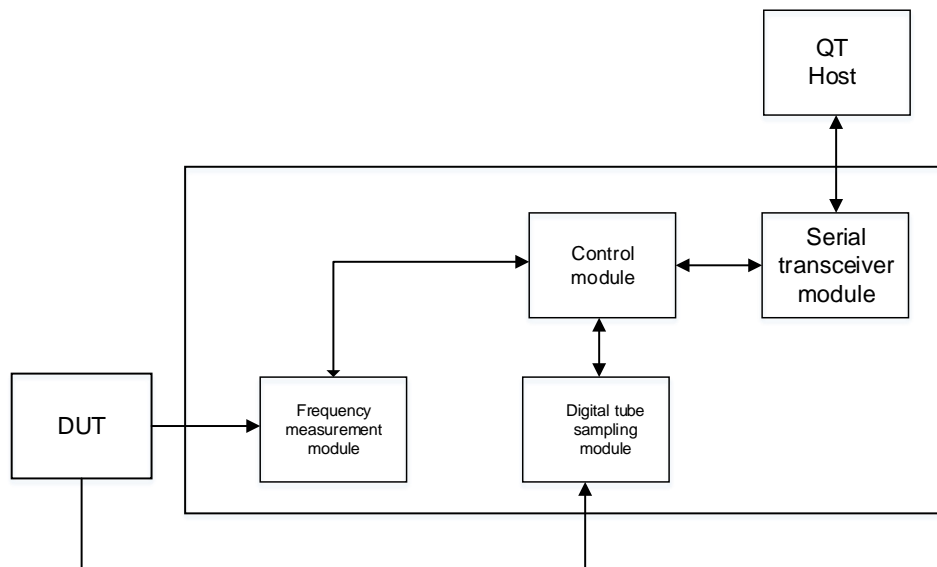


Project Design

I prepare to design a verification module which can be inserted into the designer's module. In order to facilitate a quick analysis, it is therefore necessary to do it in a host. The data is downloaded to the FPGA to start testing according to the instructions of the host.

Design Diagram

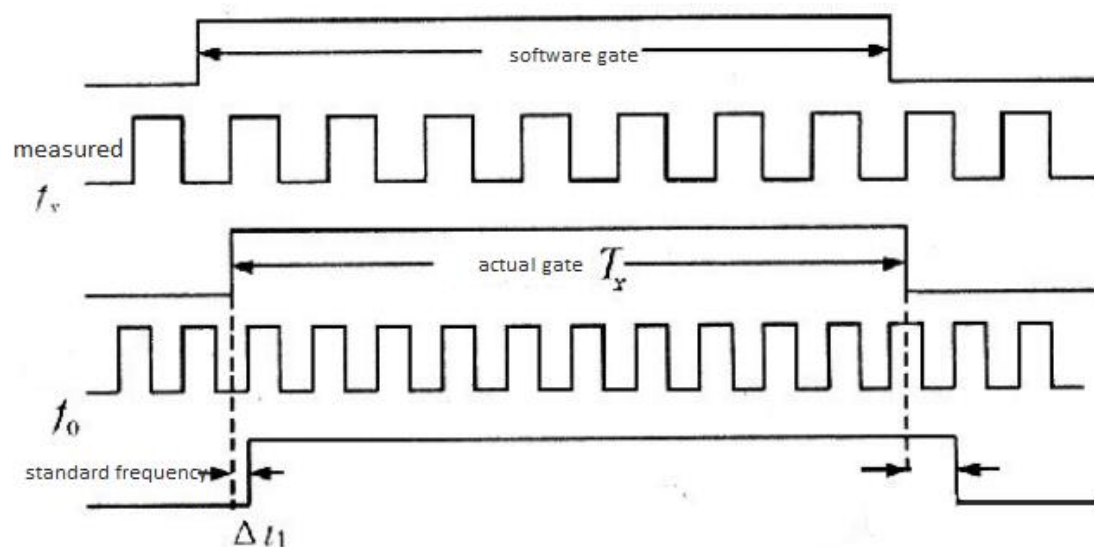
The design diagram is as shown below.



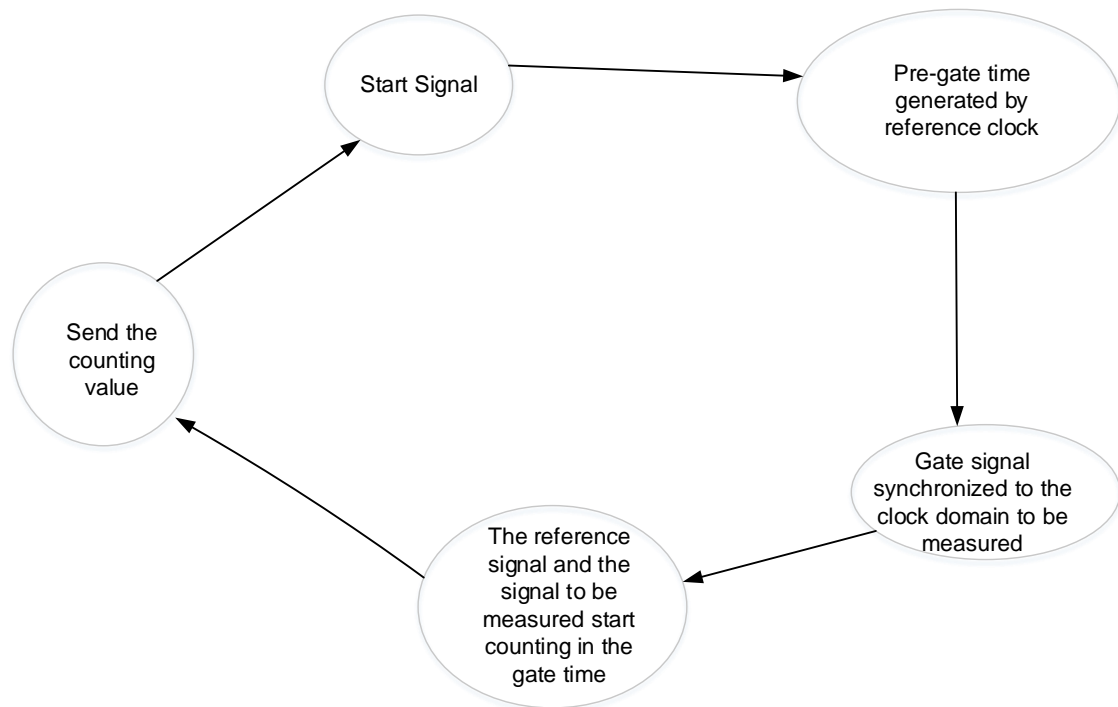
It will describe the four modules in details in the followings.

1. Frequency measurement module

Equal precision frequency measurement principle is as shown below.



According to the principle of equal precision frequency measurement, the coding diagram is as follows.



The pre-gate coding is shown below.

```

always @(posedge I_sys_clk or negedge I_rst_n) begin
•   if(!I_rst_n) begin
•       gate_cnt <= 32'd0;
•       R_done_pre <= 0;
•       gate_time_pre <= 0;
•       R_start <= 0;
•   end
•   else begin
•       if(W_start_pos) begin
•           R_start <= 1;
•       end
•       if(R_start) begin
•           if(gate_cnt == SET_1S_PERD) begin
•               gate_time_pre <= 0;
•               R_done_pre <= 1;
•               R_start <= 0;
•           end
•       end
•   end

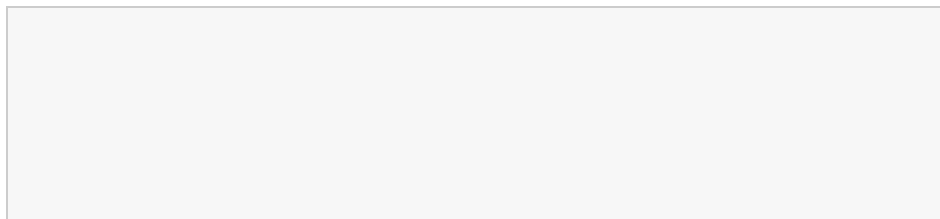
```

```

•         else begin
•             gate_cnt <= gate_cnt + 32'd1;
•             gate_time_pre <= 1;
•             R_done_pre <= 0;
•         end
•     end
•     else begin
•         gate_time_pre <= 0;
•         gate_cnt <= 32'd0;
•         R_done_pre <= R_done_pre;
•     end
• end
• end
• end

```

The code for pre-gate synchronized to the clock domain to be measured is shown below.



```

1. reg gate_time;
2.
3. always @(posedge I_clk_fx or negedge I_rst_n) begin
4.     if(!I_rst_n) begin
5.         gate_time <= 0;
6.     end
7.     else begin
8.         gate_time <= gate_time_pre;
9.     end
10.
11. end

```

The code for the reference signal and the signal to be measured is shown below.

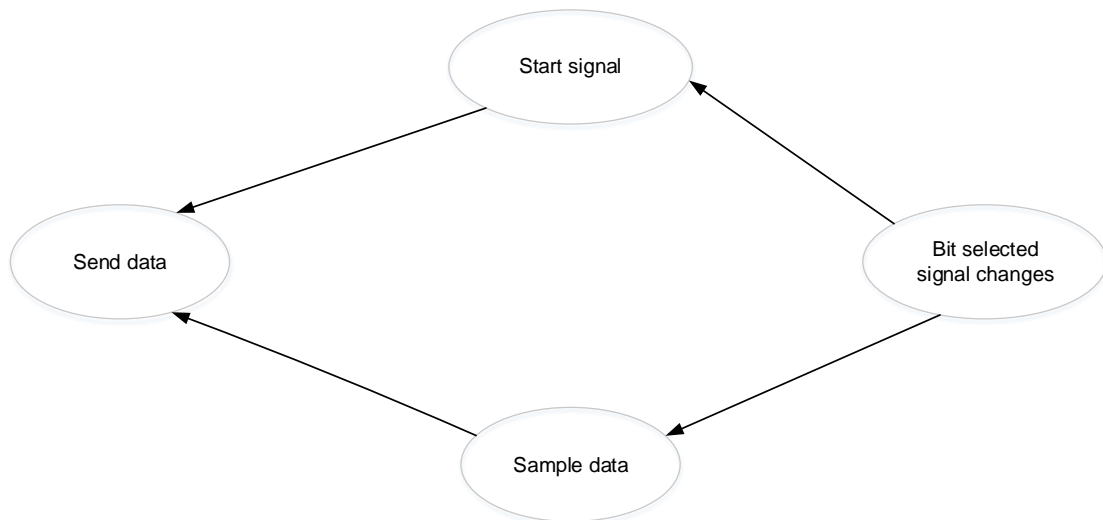
```

1. assign O_done = ({R_done_pre,gate_time} == 2'b10) ? 1 : 0;
2.
3. always @(posedge I_clk_fx or negedge I_rst_n) begin
4.     if(!I_rst_n) begin
5.         O_fx_cnt <= 32'd0;
6.     end
7.     else begin
8.         if(gate_time) begin
9.             O_fx_cnt <= O_fx_cnt + 32'd1;
10.        end
11.        else begin
12.            O_fx_cnt <= O_fx_cnt;
13.        end
14.    end
15. end
16.
17. always @(posedge I_sys_clk or negedge I_rst_n) begin
18.     if(!I_rst_n) begin
19.         O_f0_cnt <= 32'd0;
20.     end
21.     else begin
22.         if(gate_time) begin
23.             O_f0_cnt <= O_f0_cnt + 32'd1;
24.        end
25.        else begin
26.            O_f0_cnt <= O_f0_cnt;
27.        end
28.    end
29. end

```

2. Digital tube sampling module

The diagram is as shown below.



The code is as follows.

```

1. always @(posedge I_sys_clk or negedge I_rst_n) begin
2.     if(~I_rst_n) begin
3.         R_test_num <= 0;
4.         O_data1 <= 0;
5.         O_data0 <= 0;
6.         R_01_flag <= 1'b0;
7.         R_02_flag <= 1'b0;
8.         R_03_flag <= 1'b0;
9.         R_04_flag <= 1'b0;
10.    end
11.    else begin
12.        if(R_start_en & I_test_en) begin
13.            case (I_test_sel)
14.                4'b0001 : begin
15.                    O_data1[15:0] <= I_test_disp_data;
16.                    O_data0[7:0] <= I_test_seg;
17.                    R_01_flag <= 1'b1;
18.                    if(R_01_flag) begin
19.                        R_test_num <= R_test_num;
20.                    end
21.                else begin
22.                    R_test_num <= R_test_num + 1;
23.                end
24.            end

```

```

25.         end
26.         4'b0010 : begin
27.             O_data0[15:8] <= I_test_seg;
28.             R_02_flag <= 1'b1;
29.             if(R_02_flag) begin
30.                 R_test_num <= R_test_num;
31.             end
32.             else begin
33.                 R_test_num <= R_test_num + 1;
34.             end
35.         end
36.         4'b0100 : begin
37.             O_data0[23:16] <= I_test_seg;
38.             R_03_flag <= 1'b1;
39.             if(R_03_flag) begin
40.                 R_test_num <= R_test_num;
41.             end
42.             else begin
43.                 R_test_num <= R_test_num + 1;
44.             end
45.         end
46.         4'b1000 : begin
47.             O_data0[31:24] <= I_test_seg;
48.             R_04_flag <= 1'b1;
49.             if(R_04_flag) begin
50.                 R_test_num <= R_test_num;
51.             end
52.             else begin
53.                 R_test_num <= R_test_num + 1;
54.             end
55.         end
56.         default: O_data0 <= O_data0;
57.     endcase
58. end
59. else begin
60.     if(I_test_en) begin

```

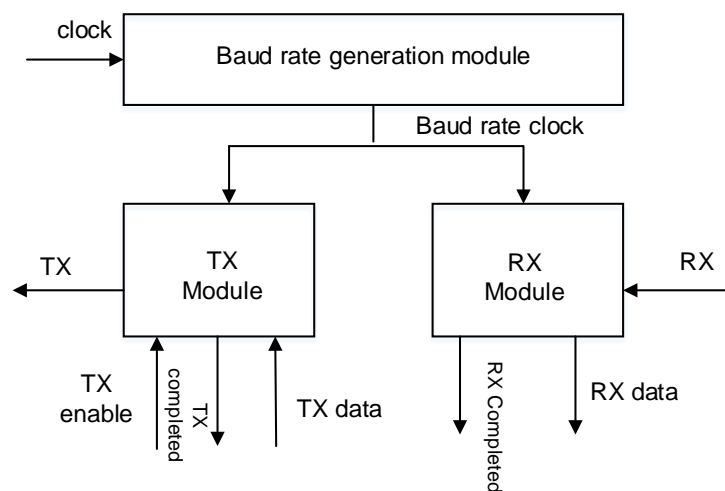
```

61.         R_test_num <= R_test_num;
62.         O_data1 <= O_data1;
63.         O_data0 <= O_data0;
64.     end
65.     else begin
66.         R_test_num <= 0;
67.         O_data1 <= 0;
68.         O_data0 <= 0;
69.         R_01_flag <= 1'b0;
70.         R_02_flag <= 1'b0;
71.         R_03_flag <= 1'b0;
72.         R_04_flag <= 1'b0;
73.     end
74. end
75. end
76. end
77.

```

3. Serial transceiver module

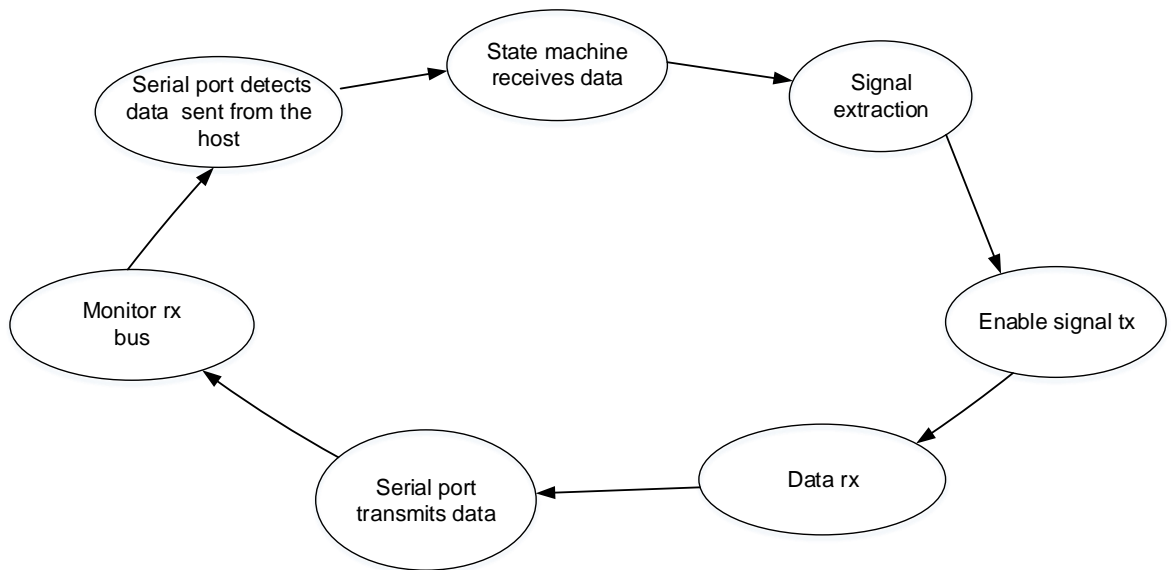
The diagram is as shown below.



For the code, you can refer to the source.

4. Control module

The diagram is as shown below.



Host Design

The source code is as follows.

```

1. HeadByte[0] = 0xEC;
2.   HeadByte[1] = 0x01;
3.   HeadByte[2] = 0x00;
4.   HeadByte[3] = 0x00;
5.   HeadByte[4] = 0x00;
6.   HeadByte[5] = 0x00;
7.   HeadByte[6] = 0x00;
8.   HeadByte[7] = 0xa5;
9.
10. void Widget::on_pushButtonP1_clicked()
    11. {
    12.
    13.   HeadByte[1] = 0x01;
    14.
    15.   //serialport->write(HeadByte);
    16.
    17.   QByteArray tmp;
    18.   tmp.resize(1);
    19.   for(int idx = 0;idx < 8;idx++){
    20.     tmp[0] = HeadByte.at(idx);
    21.     serialport->write(tmp);
    22.     Sleep(10);
  
```



```

23.     }
24.
25.     ui->pushButtonP1->setEnabled(false);
26.     ui->pushButtonP2->setEnabled(false);
27.     flag_state = 1;
28.     ui->textCommd->append("Frequency measure cmd has
        send!\n");
29.
30. }
31.
32.

```

Testing

Frequency testing

```

Gowin_PLLVR pll_inst0(
    .clkout(clkout_50m), //output clkout
    .lock(lock_o), //output lock
    .reset(~I_rst_n), //input reset
    .clkin(I_sys_clk) //input clkin
);

```

FPGA PLL 50MHz Clock Source

```

always @(posedge clkout_50m or negedge grst_n) begin
    if(!grst_n) begin
        cnt <= 0;
        clk_div <= 0;
    end
    else begin
        if(cnt == DIV_NUMBER) begin
            cnt <= 0;
            clk_div <= ~clk_div;
        end
        else begin
            cnt <= cnt + 1;
        end
    end
end
end

```

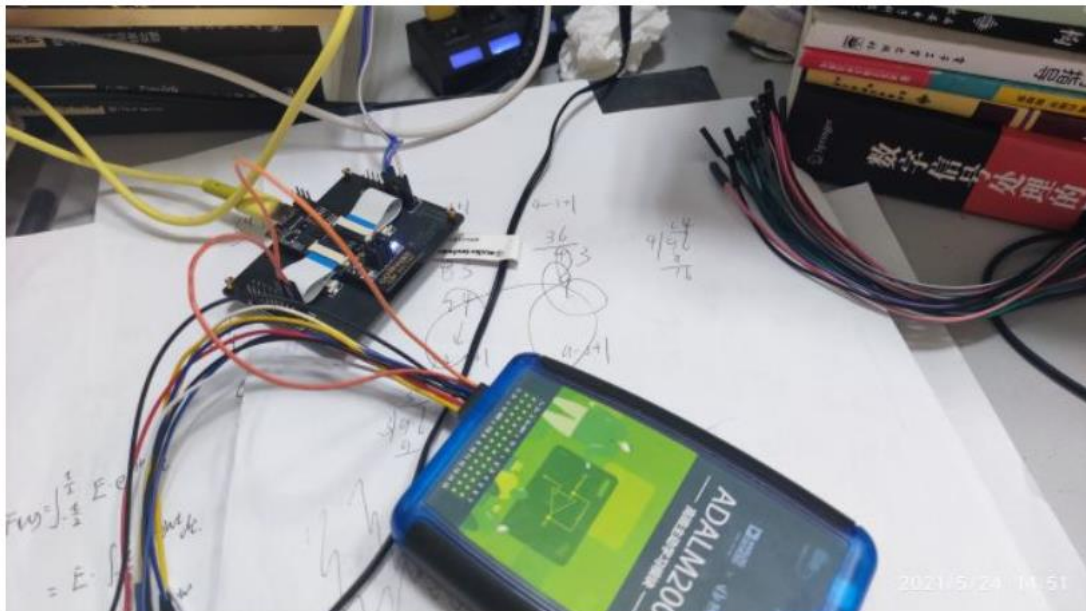
Division code

```
parameter DIV_NUMBER = 249;
```

100KHz division coefficient

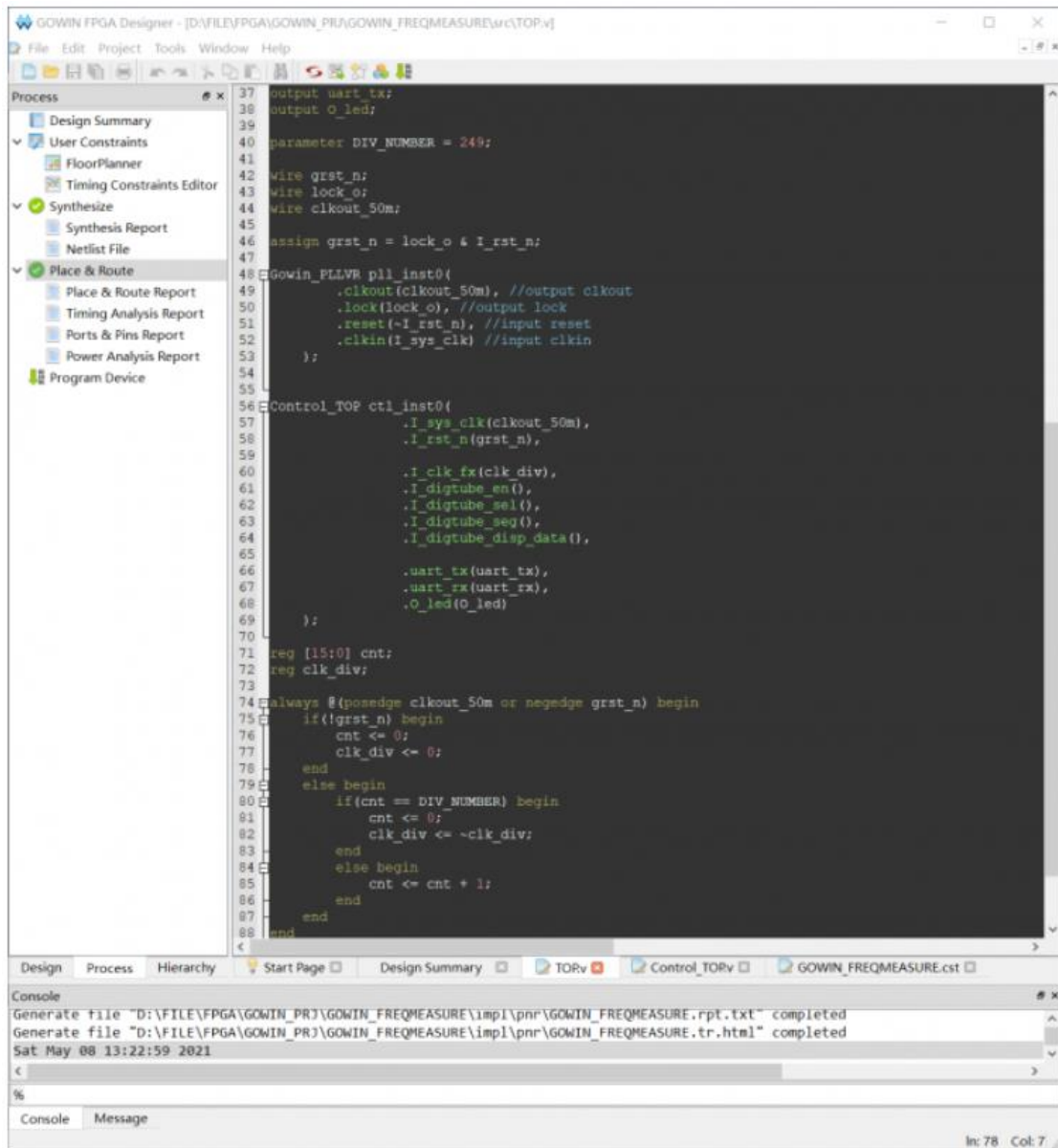
```
IO_LOC "clk_div" 27;  
IO_PORT "clk_div" IO_TYPE=LVC MOS33
```

Pinout



Connection

100K frequency Testing



User design clock 100KHz

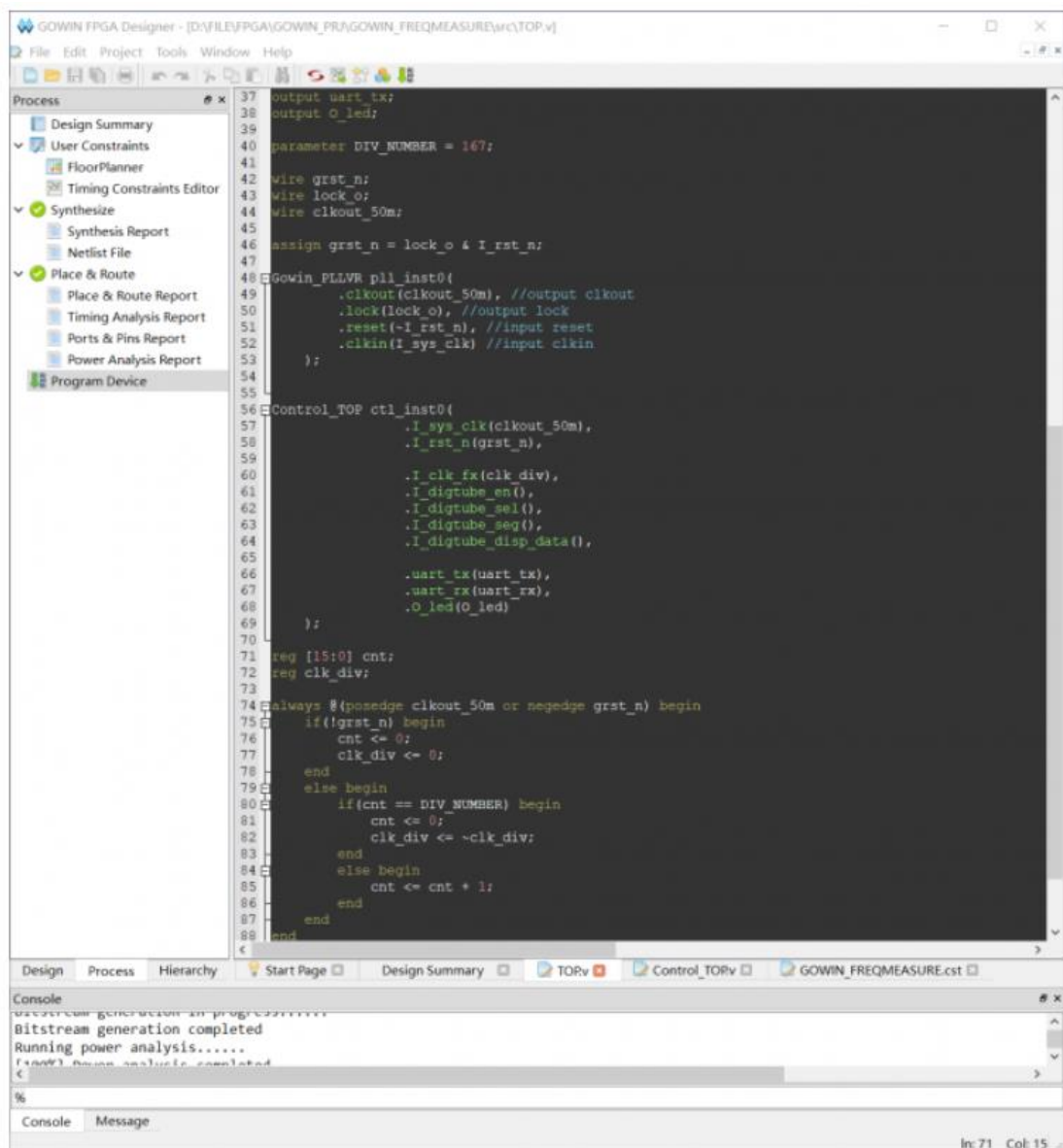


Testing by host



Testing by ADALM2000

149K frequency Testing



The screenshot displays the Gowin FPGA Designer interface. The left sidebar shows the project process, with 'Synthesize' and 'Place & Route' completed. The main editor shows the Verilog code for the design, which includes a PLL instance and a control logic block. The console at the bottom indicates that the bitstream generation is complete and the power analysis has been successfully run.

```
37 output uart_tx;
38 output O_led;
39
40 parameter DIV_NUMBER = 167;
41
42 wire grst_n;
43 wire lock_o;
44 wire clkout_50m;
45
46 assign grst_n = lock_o & I_rst_n;
47
48 Gowin_PLLVR pll_inst0(
49     .clkout(clkout_50m), //output clkout
50     .lock(lock_o), //output lock
51     .reset(~I_rst_n), //input reset
52     .clkin(I_sys_clk) //input clkin
53 );
54
55
56 Control_TOP ct1_inst0(
57     .I_sys_clk(clkout_50m),
58     .I_rst_n(grst_n),
59
60     .I_clk_fx(clk_div),
61     .I_digtube_en(),
62     .I_digtube_sel(),
63     .I_digtube_seg(),
64     .I_digtube_disp_data(),
65
66     .uart_tx(uart_tx),
67     .uart_rx(uart_rx),
68     .O_led(O_led)
69 );
70
71 reg [15:0] cnt;
72 reg clk_div;
73
74 always @(posedge clkout_50m or negedge grst_n) begin
75     if(!grst_n) begin
76         cnt <= 0;
77         clk_div <= 0;
78     end
79     else begin
80         if(cnt == DIV_NUMBER) begin
81             cnt <= 0;
82             clk_div <= ~clk_div;
83         end
84         else begin
85             cnt <= cnt + 1;
86         end
87     end
88 end
```

Design Process Hierarchy Start Page Design Summary TORv Control_TORv GOWIN_FREQMEASURE.cst

Console

```
Bitstream generation in progress.....
Bitstream generation completed
Running power analysis.....
Power analysis completed
```

Console Message

In: 71 Col: 15

User design 149KHz



Testing by host



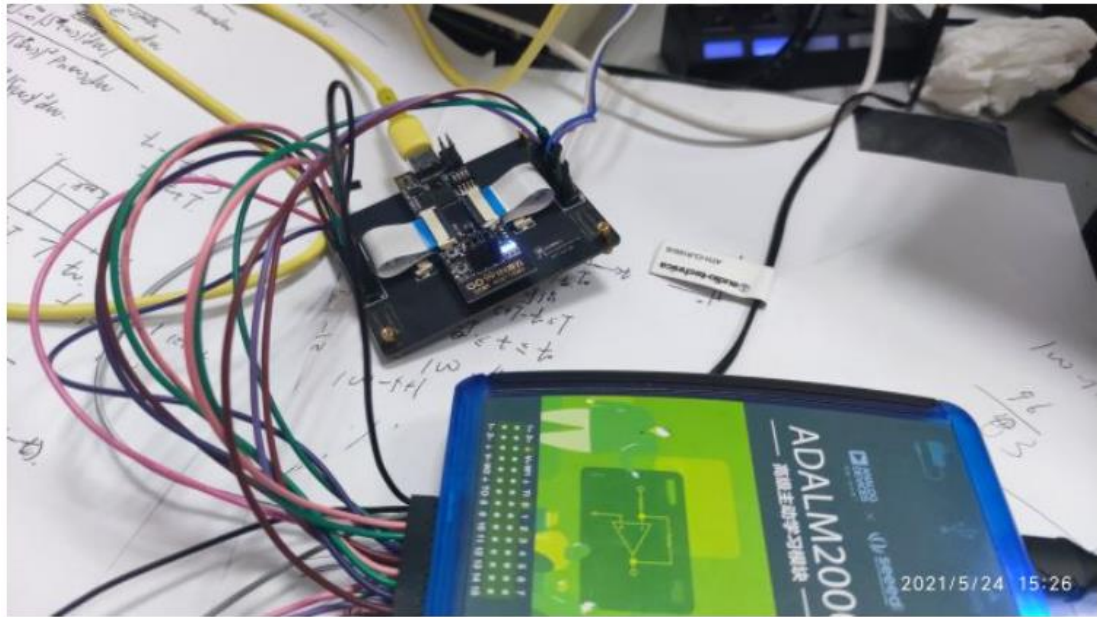
Testing by ADALM2000

Digital tube testing

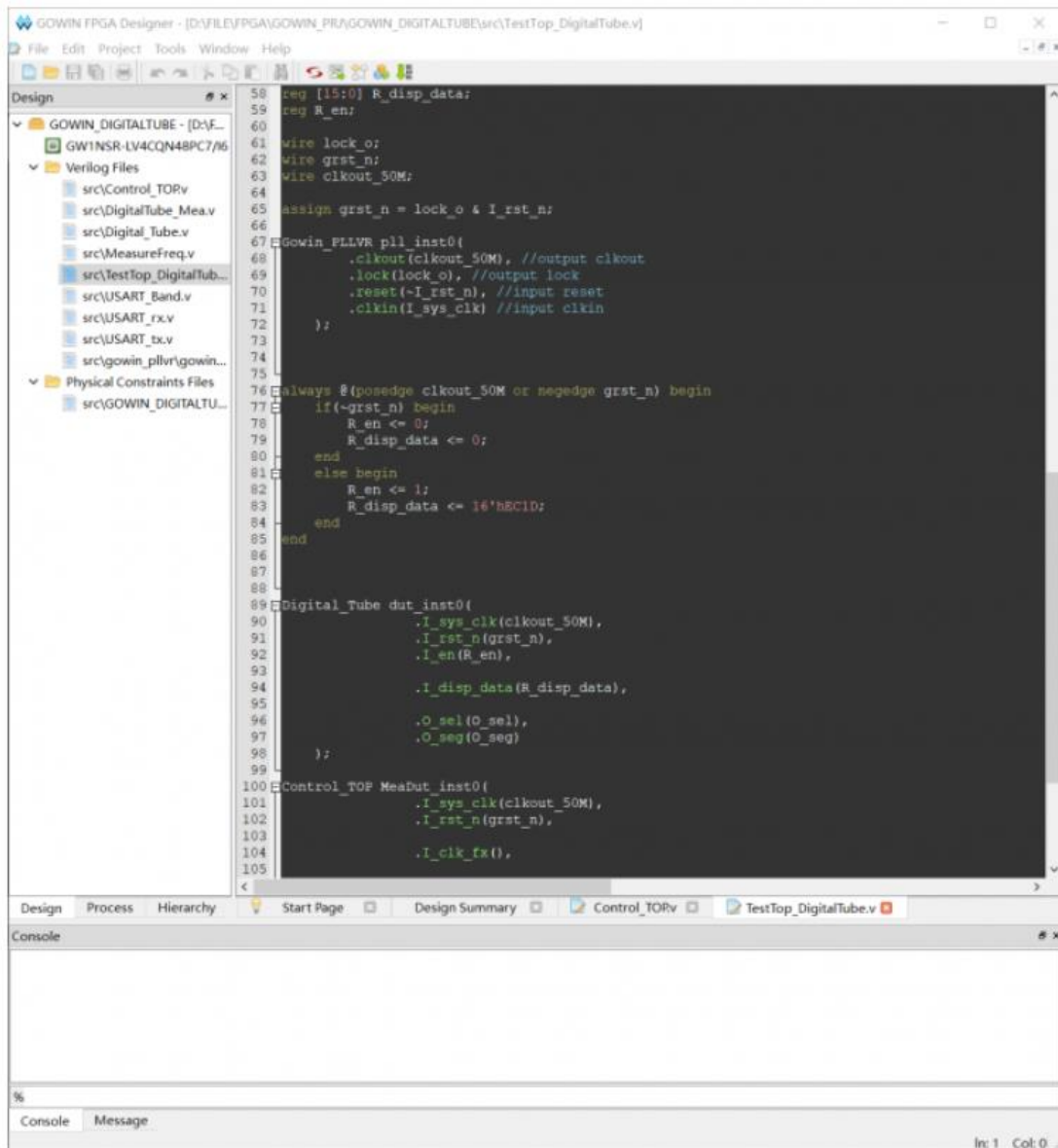
```
IO_LOC "O_sel[0]" 27;
IO_PORT "O_sel[0]" IO_TYPE=LVC MOS33
IO_LOC "O_sel[1]" 28;
IO_PORT "O_sel[1]" IO_TYPE=LVC MOS33
IO_LOC "O_sel[2]" 29;
IO_PORT "O_sel[2]" IO_TYPE=LVC MOS33
IO_LOC "O_sel[3]" 30;
IO_PORT "O_sel[3]" IO_TYPE=LVC MOS33

IO_LOC "O_seg[0]" 31;
IO_PORT "O_seg[0]" IO_TYPE=LVC MOS33
IO_LOC "O_seg[1]" 32;
IO_PORT "O_seg[1]" IO_TYPE=LVC MOS33
IO_LOC "O_seg[2]" 8;
IO_PORT "O_seg[2]" IO_TYPE=LVC MOS33
IO_LOC "O_seg[3]" 34;
IO_PORT "O_seg[3]" IO_TYPE=LVC MOS33
IO_LOC "O_seg[4]" 35;
IO_PORT "O_seg[4]" IO_TYPE=LVC MOS33
IO_LOC "O_seg[5]" 41;
IO_PORT "O_seg[5]" IO_TYPE=LVC MOS33
IO_LOC "O_seg[6]" 42;
IO_PORT "O_seg[6]" IO_TYPE=LVC MOS33
IO_LOC "O_seg[7]" 43;
IO_PORT "O_seg[7]" IO_TYPE=LVC MOS33
```

Output Pinout



Connection



Gowin editor



Testing by host



Testing by ADALM2000

Source code

You can click this link

https://github.com/kevinliuyunfeng/GOWIN_FPGA.git to get the source code.

