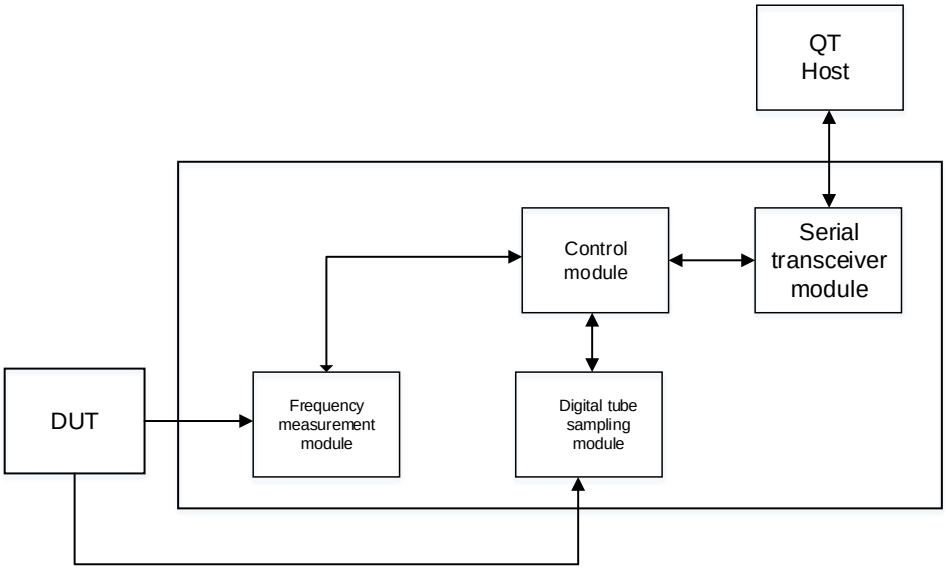
Project Design

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

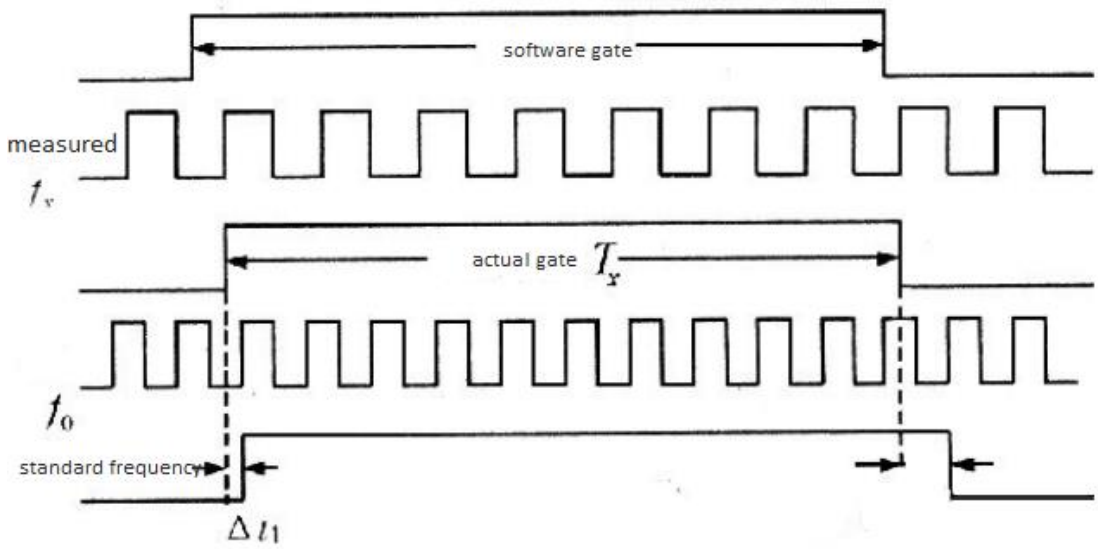
The design diagram is as shown below.



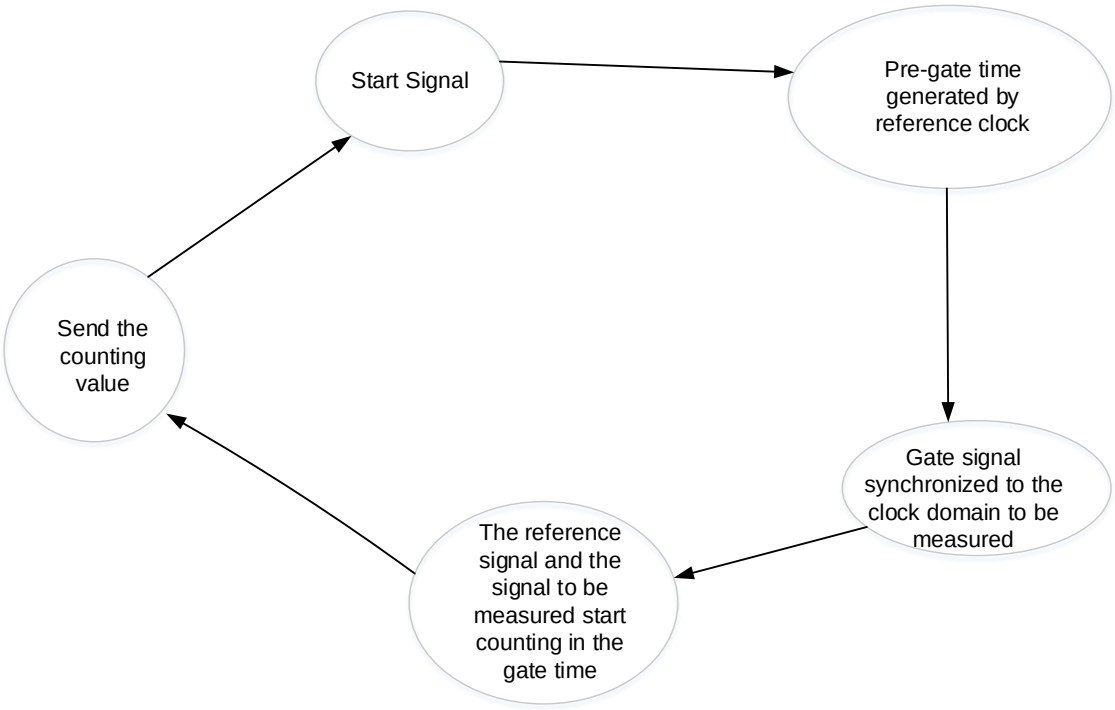
It will describe the four modules in details in the followings.

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
```

```
            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
```

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

```
reg gate_time;

always @(posedge I_clk_fx or negedge I_rst_n) begin
    if(!I_rst_n) begin
        gate_time <= 0;
    end
    else begin
        gate_time <= gate_time_pre;
    end

    end
```

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

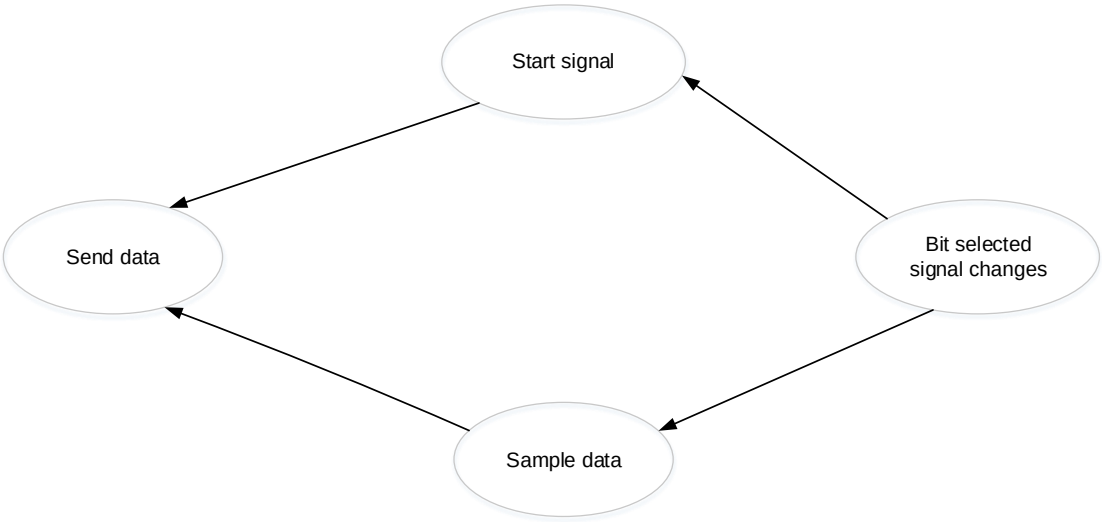
```
assign O_done = ({R_done_pre,gate_time} == 2'b10) ? 1 : 0;

always @(posedge I_clk_fx or negedge I_rst_n) begin
    if((!I_rst_n)) begin
        O_fx_cnt <= 32'd0;
    end
    else begin
        if(gate_time) begin
            O_fx_cnt <= O_fx_cnt + 32'd1;
        end
        else begin
            O_fx_cnt <= O_fx_cnt;
        end
    end
end

always @(posedge I_sys_clk or negedge I_rst_n) begin
    if((!I_rst_n)) begin
        O_f0_cnt <= 32'd0;
    end
    else begin
        if(gate_time) begin
            O_f0_cnt <= O_f0_cnt + 32'd1;
        end
        else begin
            O_f0_cnt <= O_f0_cnt;
        end
    end
end
```

2. Digital tube sampling module

   The diagram is as shown below.



The code is as follows.

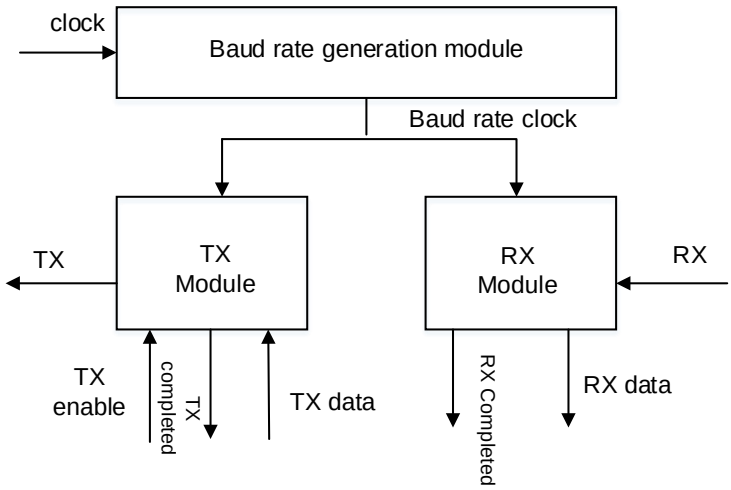
```
always @(posedge I_sys_clk or negedge I_rst_n) begin
    if(~I_rst_n) begin
        R_test_num <= 0;
        O_data1 <= 0;
        O_data0 <= 0;
        R_01_flag <= 1'b0;
        R_02_flag <= 1'b0;
        R_03_flag <= 1'b0;
        R_04_flag <= 1'b0;
    end
    else begin
        if(R_start_en & I_test_en) begin
            case (I_test_sel)
                4'b0001 : begin
                    O_data1[15:0] <= I_test_disp_data;
                    O_data0[7:0] <= I_test_seg;
                    R_01_flag <= 1'b1;
                    if(R_01_flag) begin
                        R_test_num <= R_test_num;
                    end
                    else begin
                        R_test_num <= R_test_num + 1;
                    end
```

```
                end
            4'b0010 : begin
                O_data0[15:8] <= I_test_seg;
                R_02_flag <= 1'b1;
                if(R_02_flag) begin
                    R_test_num <= R_test_num;
                end
                else begin
                    R_test_num <= R_test_num + 1;
                end
                end
            4'b0100 : begin
                O_data0[23:16] <= I_test_seg;
                R_03_flag <= 1'b1;
                if(R_03_flag) begin
                    R_test_num <= R_test_num;
                end
                else begin
                    R_test_num <= R_test_num + 1;
                end
                end
            4'b1000 : begin
                O_data0[31:24] <= I_test_seg;
                R_04_flag <= 1'b1;
                if(R_04_flag) begin
                    R_test_num <= R_test_num;
                end
                else begin
                    R_test_num <= R_test_num + 1;
                end
                end
            default: O_data0 <= O_data0;
            endcase
        end
        else  begin
            if(I_test_en) begin
```

```
                R_test_num <= R_test_num;
                O_data1 <= O_data1;
                O_data0 <= O_data0;
            end
        else begin
                R_test_num <= 0;
                O_data1 <= 0;
                O_data0 <= 0;
                R_01_flag <= 1'b0;
                R_02_flag <= 1'b0;
                R_03_flag <= 1'b0;
                R_04_flag <= 1'b0;
            end
        end
    end
end
```

3. Serial transceiver module

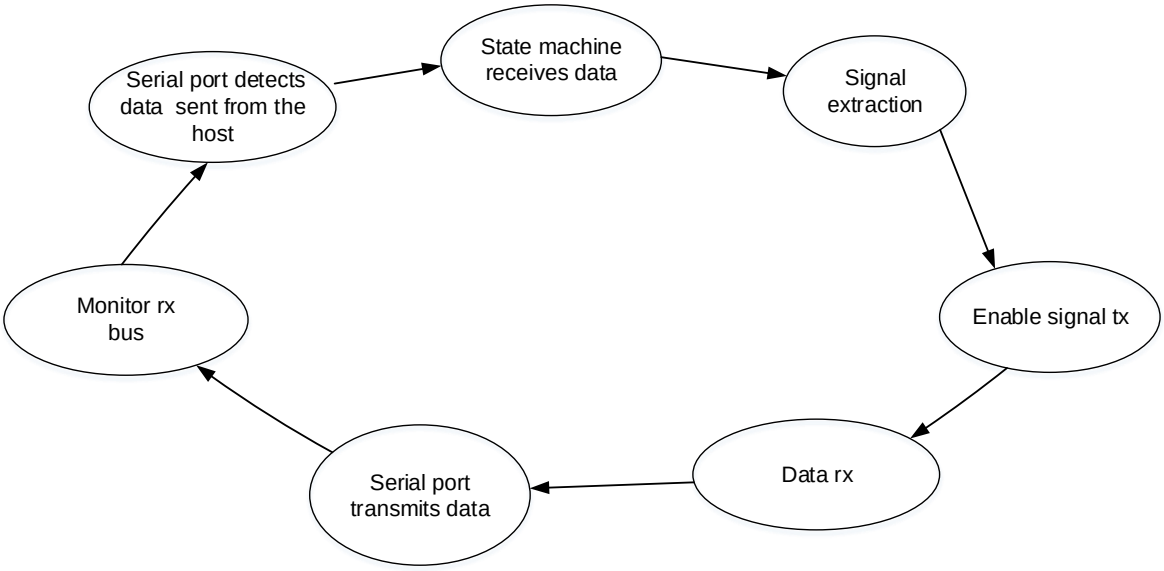
   The diagram is as shown below.



   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.

```
HeadByte[0] = 0xEC;
    HeadByte[1] = 0x01;
    HeadByte[2] = 0x00;
    HeadByte[3] = 0x00;
    HeadByte[4] = 0x00;
    HeadByte[5] = 0x00;
    HeadByte[6] = 0x00;
    HeadByte[7] = 0xa5;

void Widget::on_pushButtonP1_clicked()
    {

    HeadByte[1] = 0x01;

    //serialport->write(HeadByte);

    QByteArray tmp;
    tmp.resize(1);
    for(int idx = 0;idx < 8;idx++){
        tmp[0] = HeadByte.at(idx);
        serialport->write(tmp);
        Sleep(10);
```

```
    }

    ui->pushButtonP1->setEnabLED(false);
    ui->pushButtonP2->setEnabled(false);
    flag_state = 1;
    ui->textCommd->append("Frequence measure cmd has sended!\n");

}
```

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
```

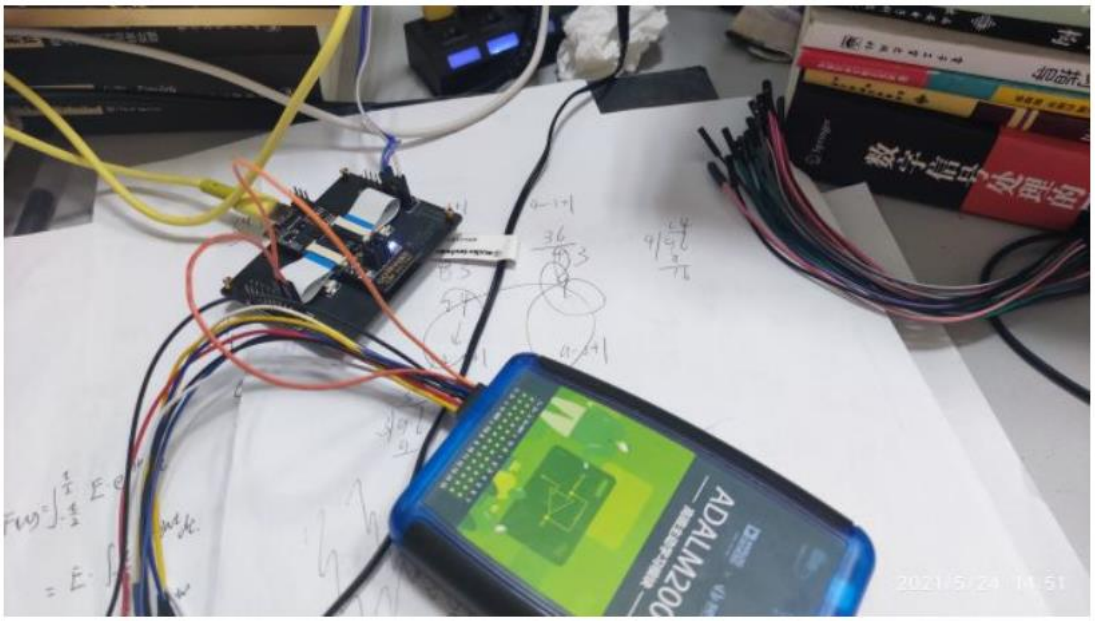
Division code

```
parameter DIV_NUMBER = 249;
```

100KHz division coefficient

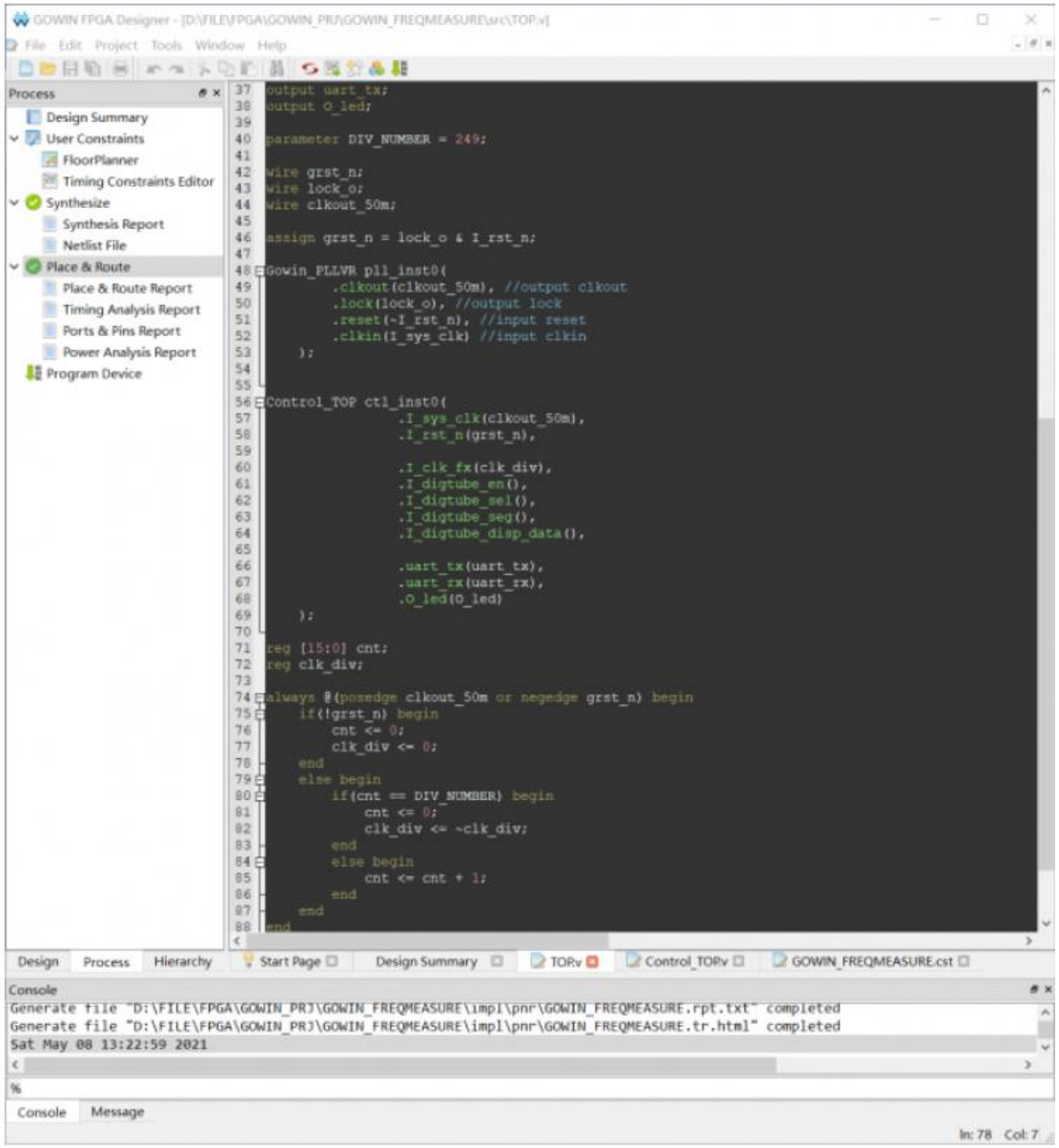
```
IO_LOC "clk_div" 27;
IO_PORT "clk_div" IO_TYPE=LVCMOS33
```

Pinout



Connection

100K frequency Testing

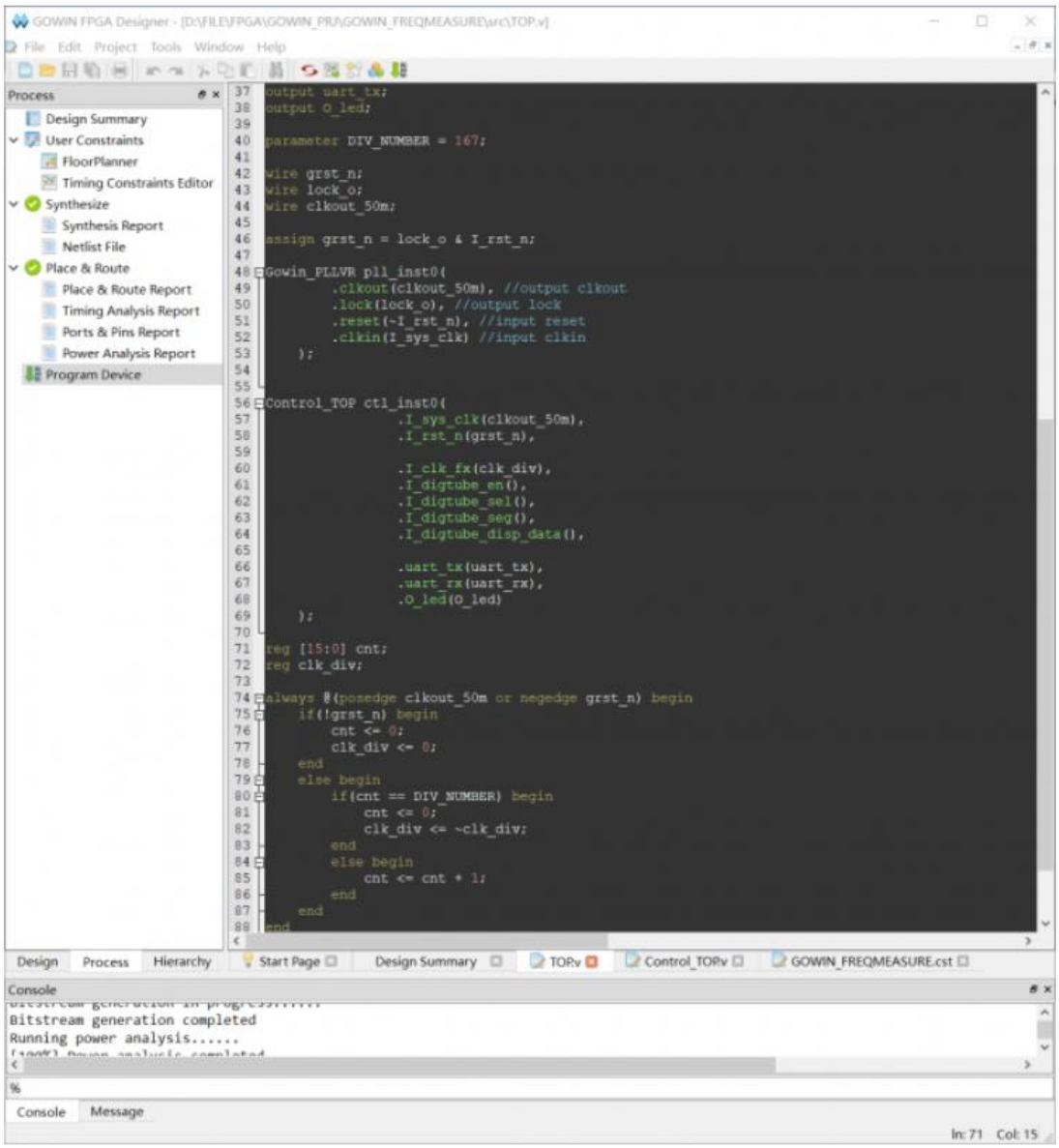


User design clock 100KHz

Testing by host

Testing by ADALM2000

149K frequency Testing



User design 149KHz

Testing by host

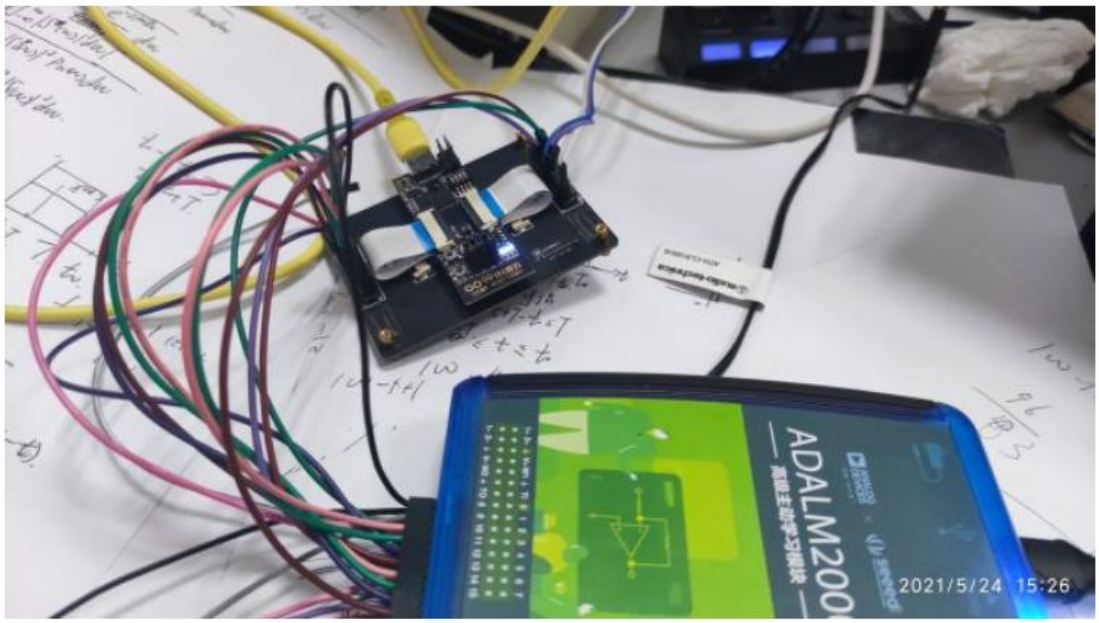
Testing by ADALM2000

Digital tube testing

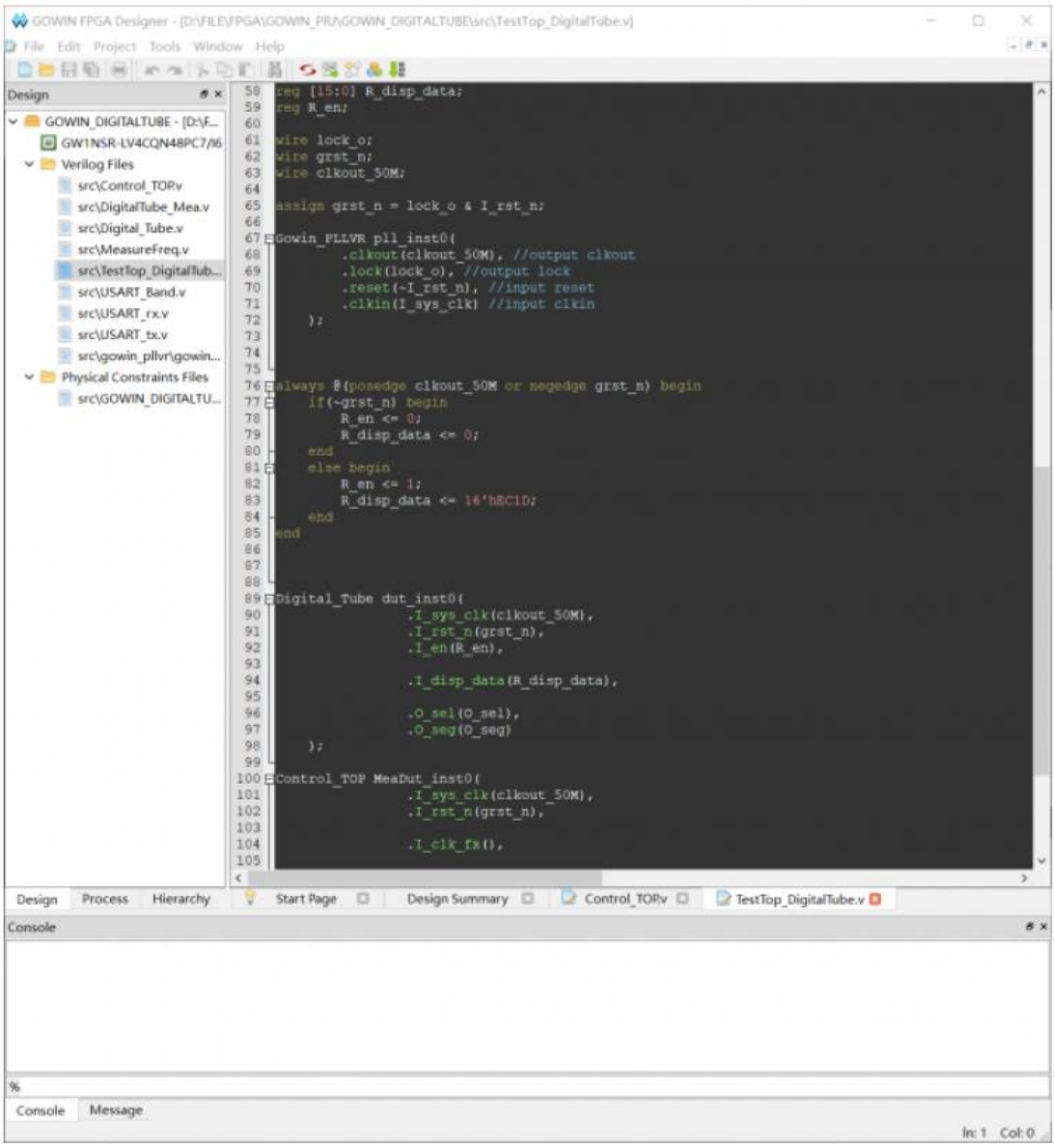
```
IO_LOC "O_sel[0]" 27;
IO_PORT "O_sel[0]" IO_TYPE=LVCMOS33
IO_LOC "O_sel[1]" 28;
IO_PORT "O_sel[1]" IO_TYPE=LVCMOS33
IO_LOC "O_sel[2]" 29;
IO_PORT "O_sel[2]" IO_TYPE=LVCMOS33
IO_LOC "O_sel[3]" 30;
IO_PORT "O_sel[3]" IO_TYPE=LVCMOS33

IO_LOC "O_seg[0]" 31;
IO_PORT "O_seg[0]" IO_TYPE=LVCMOS33
IO_LOC "O_seg[1]" 32;
IO_PORT "O_seg[1]" IO_TYPE=LVCMOS33
IO_LOC "O_seg[2]" 8;
IO_PORT "O_seg[2]" IO_TYPE=LVCMOS33
IO_LOC "O_seg[3]" 34;
IO_PORT "O_seg[3]" IO_TYPE=LVCMOS33
IO_LOC "O_seg[4]" 35;
IO_PORT "O_seg[4]" IO_TYPE=LVCMOS33
IO_LOC "O_seg[5]" 41;
IO_PORT "O_seg[5]" IO_TYPE=LVCMOS33
IO_LOC "O_seg[6]" 42;
IO_PORT "O_seg[6]" IO_TYPE=LVCMOS33
IO_LOC "O_seg[7]" 43;
IO_PORT "O_seg[7]" IO_TYPE=LVCMOS33
```

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.