# COD LAB1 实验报告

姓名：刘芷辰

学号：PB21111728

日期：2023.3.29

## 1. 实验题目

运算器及其运用

## 2. 实验目标

- 掌握算术逻辑单元 (ALU) 的功能
- 掌握数据通路和控制器的设计方法
- 掌握组合电路和时序电路，以及参数化和结构化的 Verilog 描述方法

## 3. 实验内容

### 3.1 PART 1（ALU及测试）

> 逻辑设计(包括选做)

- ALU模块设计

```
module alu #(parameter WIDTH =6) // data width
(
 input [WIDTH - 1:0] a, b, // 输入
 input [3:0] f, // 操作码
 output reg [WIDTH - 1:0] y, // 输出
 output reg of // 加减法溢出置1
);

 always @(*) begin
 case (f)
 4'b0000:begin
 y = a + b;
```

```
13  of=(a[WIDTH-1]==b[WIDTH-1])&&(y[WIDTH-1]!=a[WIDTH-1]);
14  end
15
16  4'b0001:begin
17  y = a - b;
18  of=(a[WIDTH-1]!=b[WIDTH-1])&&(y[WIDTH-1]!=a[WIDTH-1]);
19  end
20
21  4'b0010:begin
22  of=0;
23  if (a==b) y = 1; else y=0;
24  end
25
26  4'b0011:begin
27  of=0;
28  if (a<b) y = 1; else y=0;
29  end
30
31  4'b0100:begin
32  of=0;
33  if ($signed(a)<$signed(b)) y = 1; else y=0;
34  end
35
36  4'b0101:begin
37  of=0;
38  y = a&b;
39  end
40
41  4'b0110:begin
42  of=0;
43  y = a|b;
44  end
45
46  4'b0111:begin
47  of=0;
48  y = a ^ b;
49  end
50
51  4'b1000:begin
52  of=0;
53  y = a>>b;
54  end
55
56  4'b1001:begin
57  of = 0;
```

```
 y = a<<b;
 end

 default:begin
 of=0;
 y = {WIDTH{1'h0}};
 end
 endcase
 end
endmodule

```

通过不同操作码f来确定不同的功能

- 译码器模块

```
module decoder(
 input en,
 input [1:0] sel,
 output ea, eb, ef
);
 assign ea = en & (sel == 2'b00);
 assign eb = en & (sel == 2'b01);
 assign ef = en & (sel == 2'b10);
endmodule
```

为了将输入的x根据sel的不同复用为a,b,f，通过decoder模块，将en信号在不同的sel下复用为ea，eb，ef的使能信号

- main模块（调用decoder和ALU）

```
module main
(
 input clk,
 input en,
 input [1:0] sel,
 input [5:0] x,//输入，分时f，a，b
 output reg [5:0] y,//输出
 output reg of//溢出
);
 // wire mapping
 wire enf, ena, enb;
 wire alu_of;

```

```
 wire [5:0] alu_y;
 reg [3:0] f;
 reg [5:0] a, b;

 decoder dec(
 .en(en),
 .sel(sel),
 .ef(enf),
 .ea(ena),
 .eb(enb)
 );

 alu  alu_test
 (
 .a(a),
 .b(b),
 .f(f),
 .y(alu_y),
 .of(alu_of)
 );
 // registers
 always @(posedge clk) begin
 if (enf) f <= x[3:0]; else f<=f;
 if (ena) a <= x;else a<=a;
 if (enb) b <= x;else b<=b;
 y <= alu_y;
  of <= alu_of;
 end
endmodule

```

> 测试电路的逻辑设计和功能仿真

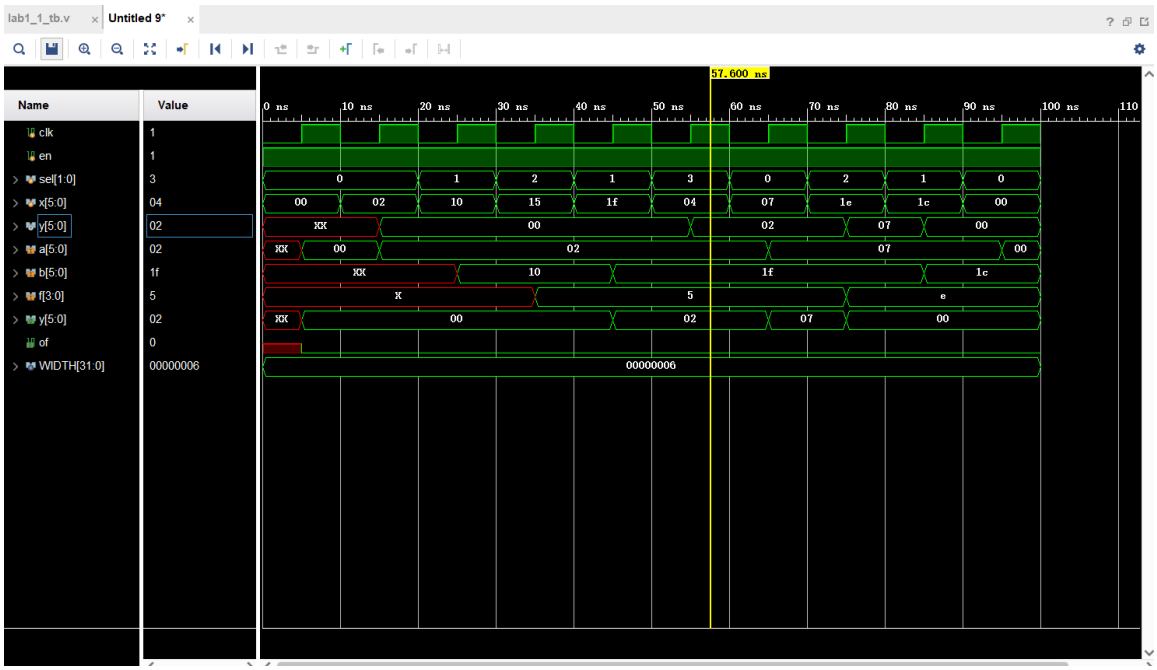
- 仿真文件：使用testbench插件生成并做一定修改得到

```
`timescale 1ns / 1ps

module main_tb;

  reg clk;
  reg en;
  reg [1:0] sel;
  reg [5:0] x;
```

```
  wire [5:0] y;

  main uut (
    .clk(clk),
    .en(en),
    .sel(sel),
    .x(x),
    .y(y)
  );

  initial begin
    // Initialize Inputs
    clk = 0;
    en = 1;
    sel = 0;
    x = 6'b000000;

    #10;

    x = 6'b000010;
    sel=2'b00;
    #10;
    x = 6'b010000;
    sel=2'b01;
    #10;
    x = 6'b010101;
    sel=2'b10;
    #10;
    x = 6'b011111;
    sel=2'b01;
    #10;
    x = 6'b000100;
    sel=2'b11;
    #10;
    x = 6'b000111;
    sel=2'b00;
    #10;
    x = 6'b011110;
    sel=2'b10;
    #10;
    x = 6'b011100;
    sel=2'b01;
    #10;
    x = 6'b000000;
    sel=2'b00;
```

```
54      #10;
55      $finish;
56    end
57
58    always #5 clk = ~clk;
59
60  endmodule
61
```

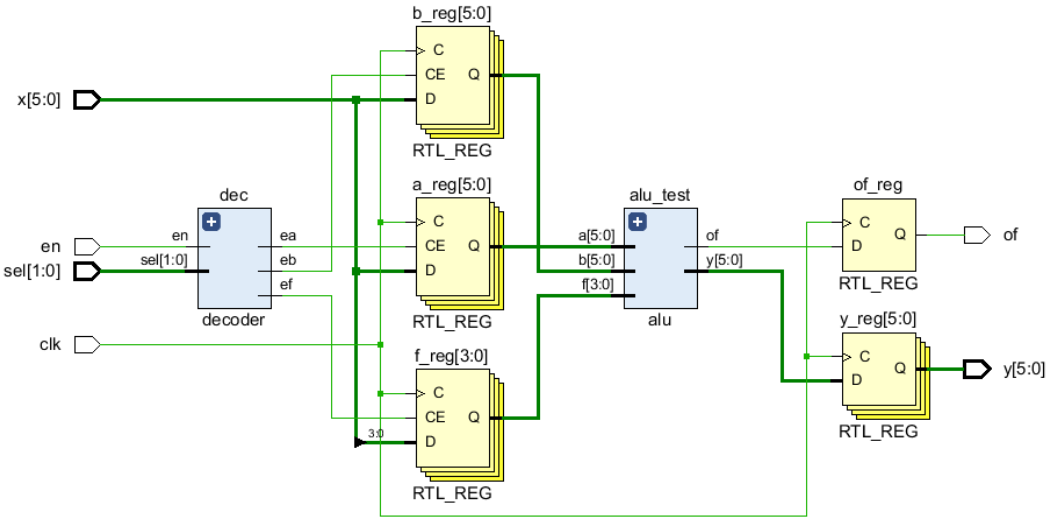
- 仿真波形



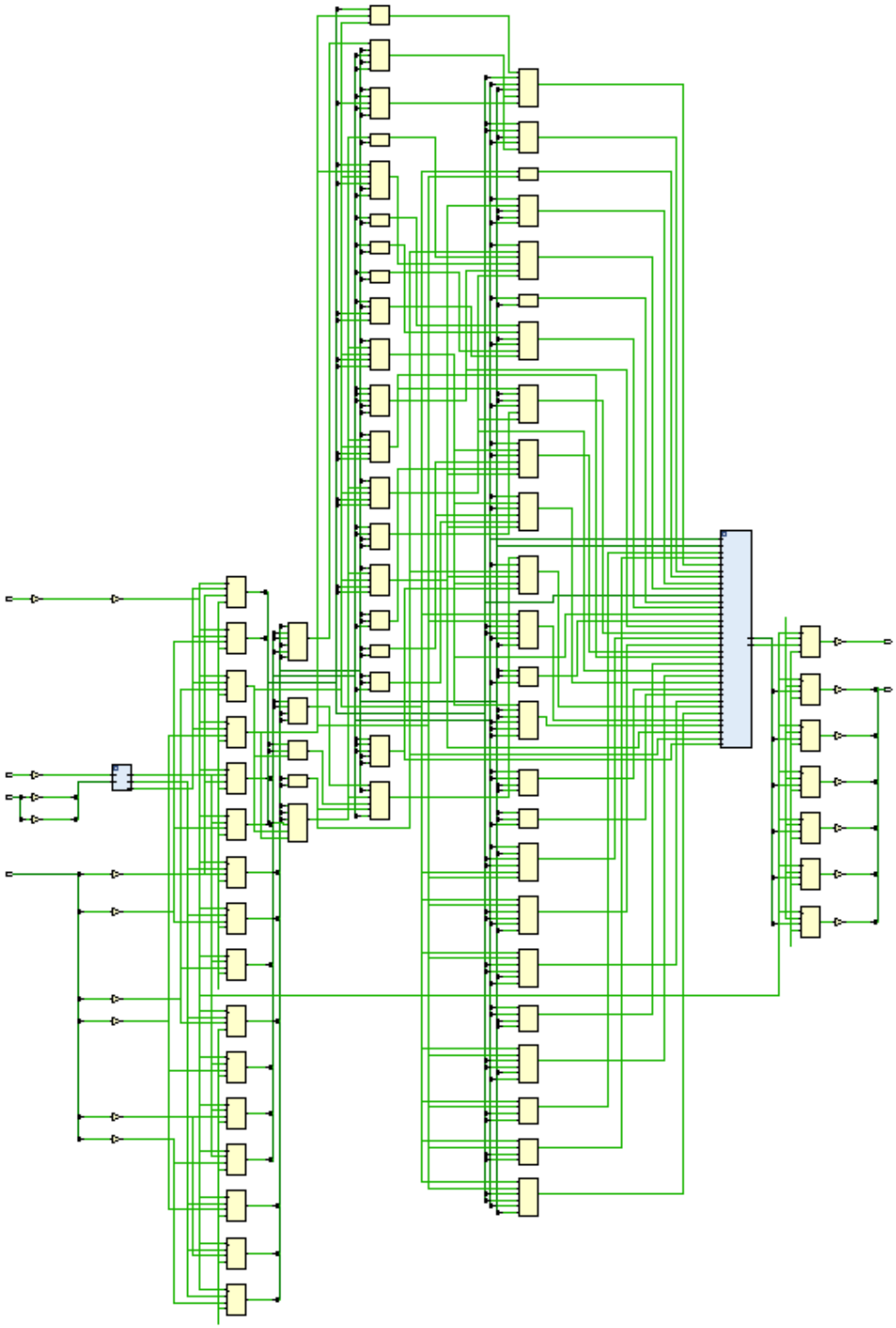

可以看到，在得到数据后的绿色波形中，sel和x在变化的过程中，a、b、y、f的值符合预期

> 测试电路的RTL分析和综合后电路图

- RTL分析电路：


b_reg[5:0]
C
CE
Q
D
RTL_REG
x[5:0]
dec
en
sel[1:0]
ea
eb
ef
decoder
a_reg[5:0]
RTL_REG
alu_test
a[5:0]
b[5:0]
f[3:0]
of
y[5:0]
alu
of_reg
RTL_REG
of
y_reg[5:0]
RTL_REG
y[5:0]
clk
f_reg[3:0]
3:0
RTL_REG

- 综合分析电路：



异同：

同：都展示了实现该功能的逻辑电路

异：RTL分析电路使用各个模块展示整体的逻辑结构

综合分析电路使用FPGA内部逻辑资源来展示，将各个模块内部的详细情况展示出来

> 测试电路下载测试

编写约束文件与引脚对应后，在FPGAOL上烧写bit文件

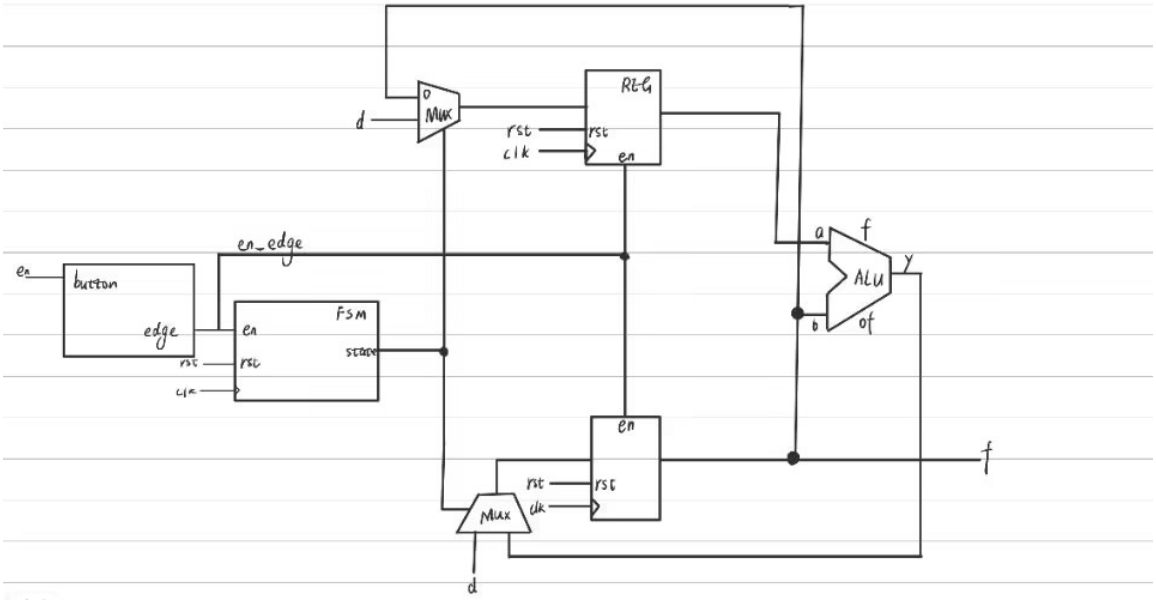
与老师所发视频对比结果完全一致

已在线下检查完毕，报告中不再展示

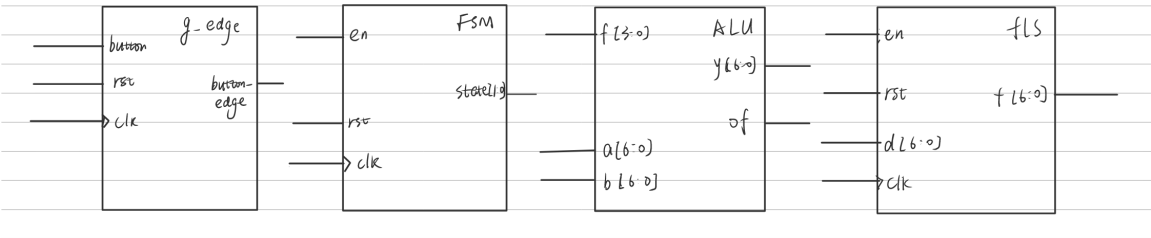
## 3.2 PART 2（FLS）

> 逻辑设计

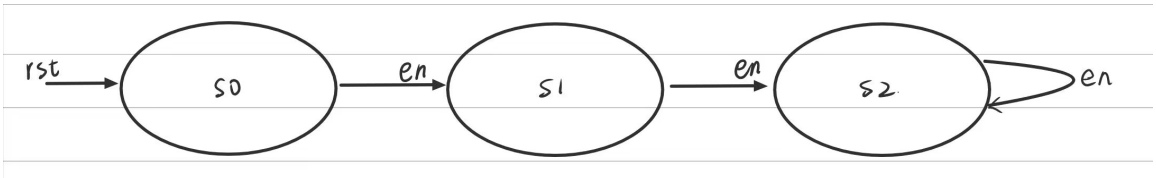
1. 数据通路




各个模块框图：




2. 状态机




3. 代码

- 取边缘信号模块

```
module g_edge(
 input clk,
 input rst,
 input button,
 output button_edge
);
 reg button_r1, button_r2;
 always @(posedge clk) button_r1 <= ~rst & button;
 always @(posedge clk) button_r2 <= button_r1;
 assign button_edge = button_r1 & ~button_r2;
endmodule
```

- 有限状态机模块（三段式）

```
module FSM(
 input clk,
 input rst,
 input en,
 output [1:0] state
);
 reg [1:0] cs;
 reg [1:0] ns;

 parameter s0 = 2'b00;
 parameter s1 = 2'b01;
 parameter s2 = 2'b10;
 // 描述 cs
 always @(posedge clk) begin
 if (rst) cs <= s0;
 else if (en) cs <= ns;
 end

 // 描述ns
 always @(*) begin
    ns=cs;
 case (cs)
 s0: ns = s1;
 s1: ns = s2;
 s2: ns = s2;
 default: ns = s0;
 endcase
 end
```

```
 //描述output
 assign state = cs;
endmodule
```

- FLS（调用ALU、FSM、g_edge）

```
module fls(
 input clk,
 input rst,
 input en,
 input [6:0] d,
 output reg [6:0] f
);

 parameter s0 = 2'b00;
 parameter s1 = 2'b01;
 parameter s2 = 2'b10;

 // get edge
 wire enEdge;
 g_edge g_enEdge(
 .clk(clk),
 .rst(rst),
 .button(en),
 .button_edge(enEdge)
 );

 // ALU
 reg [6:0] a;
 wire [6:0] alu_out;
 alu #(.WIDTH(7)) add(
 .a(a),
 .b(f),
 .f(4'b0000),
 .y(alu_out)
 );

 // FSM
 wire [1:0] state;
 FSM FSM1(
 .clk(clk),
 .rst(rst),
 .en(enEdge),
 .state(state)
```

```
  );


  always @(posedge clk) begin
  if (rst) a <= 7'b00;
  else if (enEdge) begin
  case (state)
  s0: a <= d;
  s2: a <= f;
  default: a <= a;
  endcase
  end
  end

  always @(posedge clk) begin
  if (rst) f <= 7'b00;
  else if (enEdge) begin
  case (state)
  s0: f <= d;
  s1: f <= d;
  s2: f <= alu_out;
  default: f <= f;
  endcase
  end
  end
endmodule
```

在FSM调用后对a和f进行更新

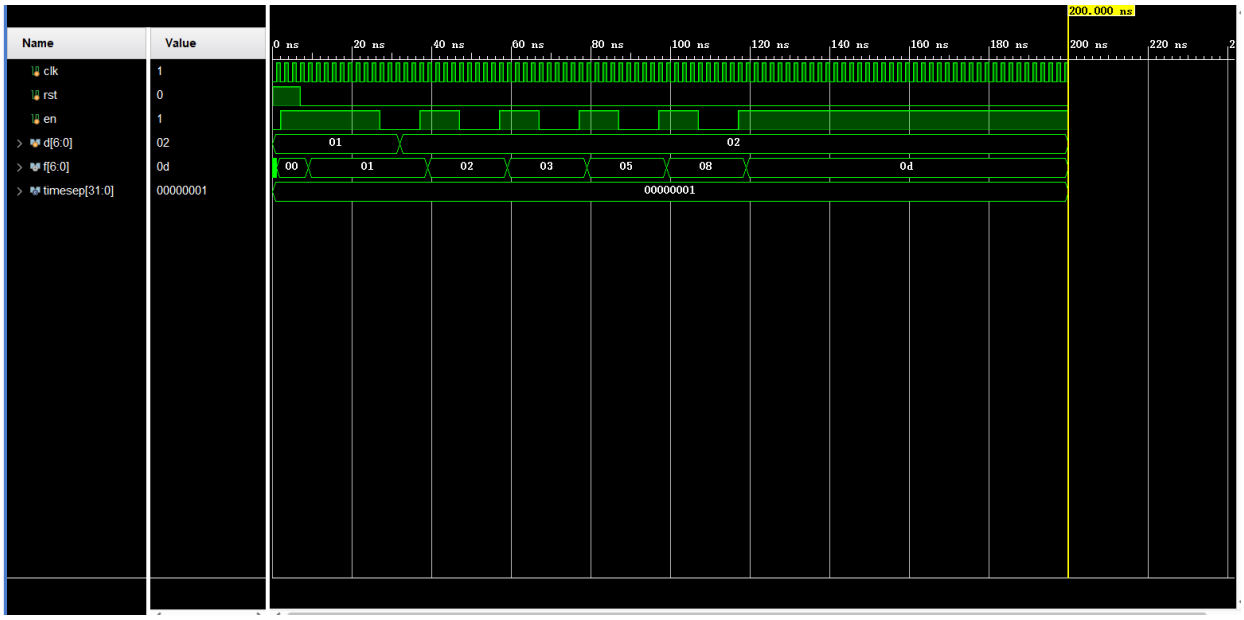
> 功能仿真

- 测试文件

```
module tb();
 reg clk;
 reg rst;
 reg en;
 reg [6:0] d;
 wire [6:0] f;
 fls test(
 .clk(clk),
 .rst(rst),
 .en(en),
 .d(d),
```

```
  .f(f)
  );
  parameter timesep = 1;
  always #(timesep) clk = ~clk;
  initial begin
  clk = 1'b0;
  #200 $finish;
  end
  initial begin
  rst = 1'b1;
  #7 rst = 1'b0;
  end
  initial begin
  en = 1'b0;
  #2 en = 1'b1;
  #25 en = 1'b0;
  #10 en = 1'b1;
  #10 en = 1'b0;
  #10 en = 1'b1;
  #10 en = 1'b0;
  #10 en = 1'b1;
  #10 en = 1'b0;
  #10 en = 1'b1;
  #10 en = 1'b0;
  #10 en = 1'b1;
  end
  initial begin
  d = 7'h01;
   #32 d = 7'h02;
  end
endmodule
```

- 仿真结果




可以看到在给定前两个数为1和2的情况下，后续的数为3、5、8……符合斐波拉契数列

> 下载测试

编写约束文件与引脚对应后，在FPGAOL上烧写bit文件

与老师所发视频对比结果完全一致

已在线下检查完毕，报告中不再展示

# 4. 总结

本次实验难度适中，但是工作量较大

在完成的过程中也复习到了很多上学期学过的知识，比如取信号边缘、信号复用等代码

但是本次实验的PPT对内容的讲解有点混乱，希望下一次能够更加清晰的罗列出实验内容以及具体要求，而把需要注意的点或者提醒单独列出来，方便更快地读懂实验内容