数字电路实验 Lab 4 实验报告

奚项正 PB23000020

必做内容

题目 1：编码器 Pro（2 分）

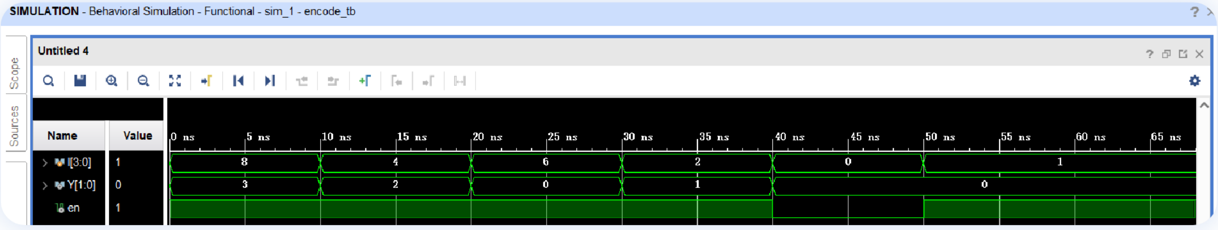
设计

```
module encode(
    input [3:0]         I,
    output reg [1:0]    Y,
    output reg          en
);
// Write your codes here
always @(*) begin
    case (I)    //据输入信号确定Y的值，非真值表中输入对应输出为00
        4'b1000: Y = 2'b11;
        4'b0100: Y = 2'b10;
        4'b0010: Y = 2'b01;
        4'b0001: Y = 2'b00;
        default: Y = 2'b00;
    endcase
end

always @(*) begin   //仅I为4'b0000时en取0
    if (I == 4'b0000) begin
        en = 2'b0;
    end
    else begin
        en = 2'b1;
    end
end
// End of your codes
endmodule
```

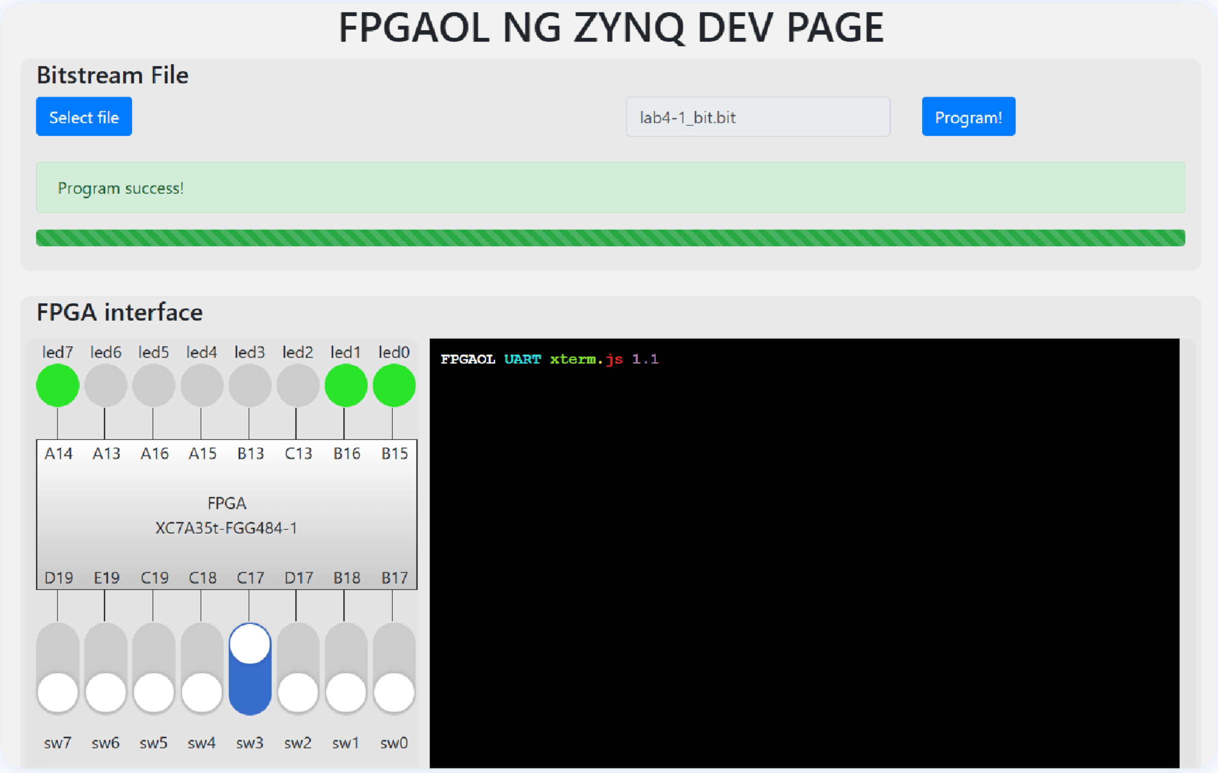
仿真

```
module encode_tb();
reg [3:0]   I;
```

```
wire [1:0]  Y;
wire        en;

initial begin
    I = 4'b1000;
    #10 I = 4'b0100;
    #10 I = 4'b0110;
    #10 I = 4'b0010;
    #10 I = 4'b0000;
    #10 I = 4'b0001;
end

encode encode(
    .I(I),
    .Y(Y),
    .en(en)
);
endmodule
```



实现



Program success!

```
FPGAOL UART xterm.js 1.1
```

题目 2：2bits 半加器（3 分）

设计

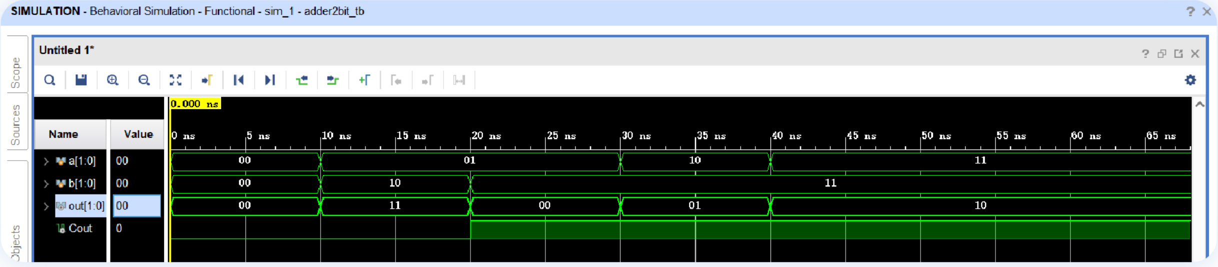
```
module HalfAdder(   //半加器，实验文档提供
    input           a, b,
    output          s, c
);
    assign s = a ^ b;
    assign c = a & b;
endmodule
```

```
module FullAdder (  //全加器，实验文档提供
    input       a, b, cin,
    output      s, cout
);
wire temp_s, temp_c_1, temp_c_2;
HalfAdder ha1(
    .a(a),
    .b(b),
    .s(temp_s),
    .c(temp_c_1)
);

HalfAdder ha2(
    .a(temp_s),
    .b(cin),
    .s(s),
    .c(temp_c_2)
);

HalfAdder ha3(
    .a(temp_c_1),
    .b(temp_c_2),
    .s(cout),
    .c()
);
endmodule
```

```
module adder2bit(   //2bit全加器
    input           [1:0]         a,
    input           [1:0]         b,
    output          [1:0]         out,
    output                        Cout
);

// Write your code here
wire tmp;
FullAdder fa1(  //个位1bit加法
    .a(a[0]),
    .b(b[0]),
    .cin(0),
    .s(out[0]),
    .cout(tmp)
);

FullAdder fa2(  //十位1bit加法
    .a(a[1]),
    .b(b[1]),
    .cin(tmp),
    .s(out[1]),
    .cout(Cout)
);
// End of your code
endmodule
```

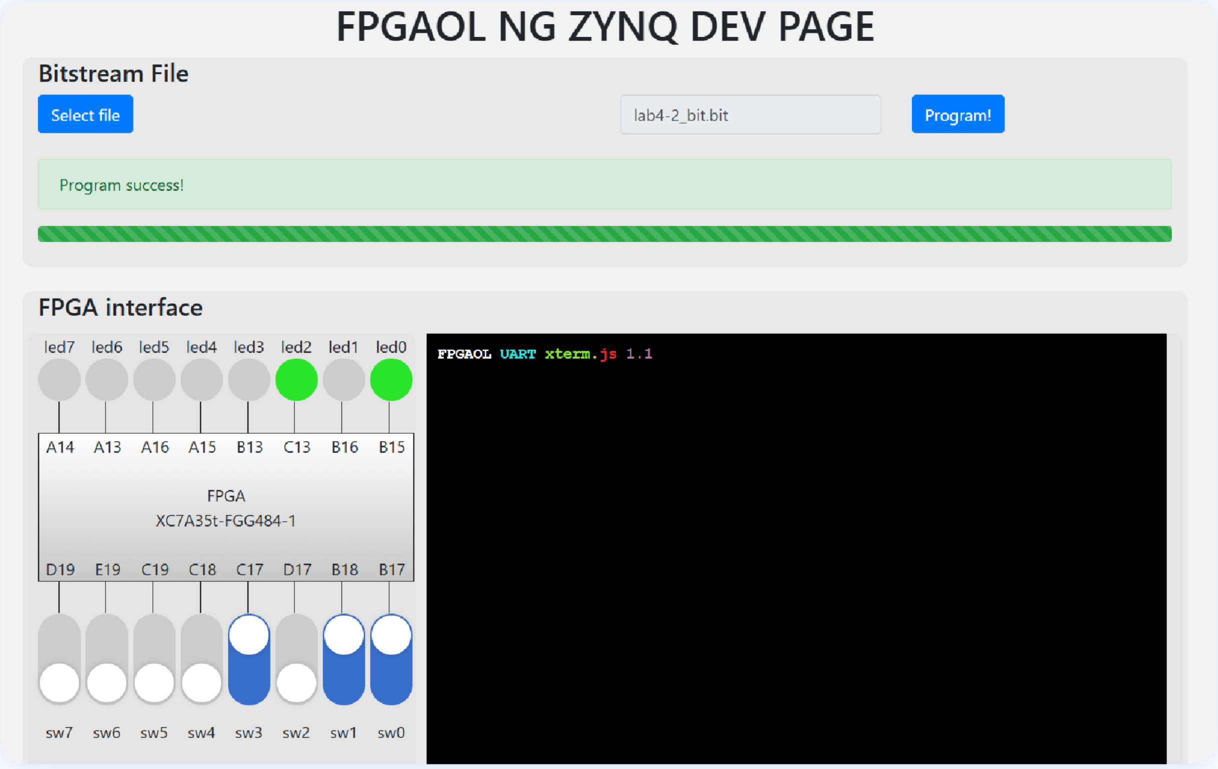
仿真

```
module adder2bit_tb();
reg [1:0] a;
reg [1:0] b;
wire [1:0] out;
wire Cout;

initial begin
    a = 2'b00;
    b = 2'b00;
    #10;
    a = 2'b01;
    b = 2'b10;
    #10;
    a = 2'b01;
    b = 2'b11;
    #10;
    a = 2'b10;
    b = 2'b11;
    #10;
    a = 2'b11;
    b = 2'b11;
end

adder2bit add2b(
    .a(a),
    .b(b),
    .out(out),
    .Cout(Cout)
);
endmodule
```



SIMULATION - Behavioral Simulation - Functional - sim_1 - adder2bit_tb

实现



FPGAOL NG ZYNQ DEV PAGE

Bitstream File

lab4-2_bit.bit

Program success!

FPGA interface

FPGAOL UART xterm.js 1.1

选择性必做内容

题目 2：8bits 5 的倍数检测器（5 分）

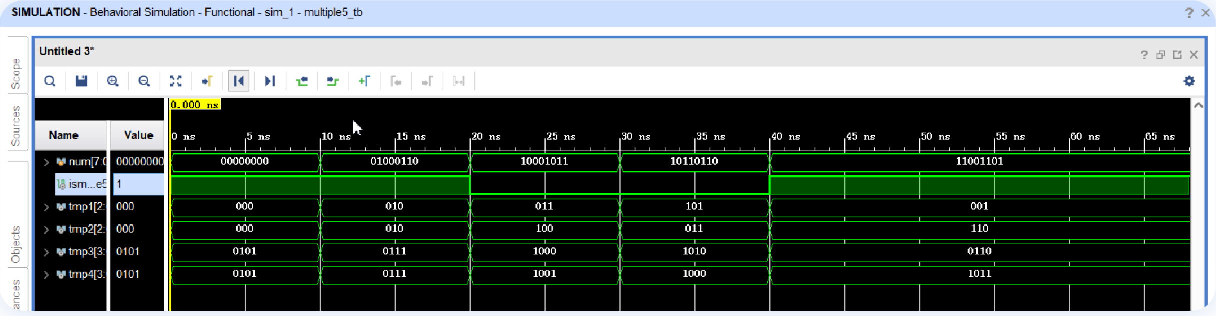
```
module adder3bit(    //3bit加法器
    input           [2:0]         a,
    input           [2:0]         b,
    output          [2:0]         out,
    output                        Cout
);

// Write your code here
wire tmp1;
FullAdder fa1(
    .a(a[0]),
    .b(b[0]),
    .cin(0),
    .s(out[0]),
    .cout(tmp1)
);

FullAdder fa2(
    .a(a[1]),
    .b(b[1]),
    .cin(tmp1),
    .s(out[1]),
    .cout(tmp2)
);

FullAdder fa3(  //比2bit加法器额外多一次调用全加器
    .a(a[2]),
    .b(b[2]),
    .cin(tmp2),
    .s(out[2]),
    .cout(Cout)
);
// End of your code
endmodule
```

```
module multiple5(
    input           [7:0]          num,
    output   reg                   ismultiple5
);

// Write your code here
// Use the 2-bits adder, or you will not get the score!
wire [2:0] tmp1;
wire [2:0] tmp2;
wire [3:0] tmp3;
wire [3:0] tmp4;

adder2bit a2bit1(
    .a(num[1:0]),
    .b(num[5:4]),
    .out(tmp1[1:0]),
    .Cout(tmp1[2])  //暂存num[1:0]与num[5:4]的和
);

adder2bit a2bit2(
    .a(num[3:2]),
    .b(num[7:6]),
    .out(tmp2[1:0]),
    .Cout(tmp2[2])  //暂存num[3:2]与num[7:6]的和
);

adder3bit a3bit1(
    .a(tmp1),
    .b(3'b101),
    .out(tmp3[2:0]),
    .Cout(tmp3[3])  //暂存tmp1 + 5
);


adder3bit a3bit2(
    .a(tmp2),
    .b(3'b101),
    .out(tmp4[2:0]),
    .Cout(tmp4[3])  //暂存tmp2 + 5
);


always @(*) begin
    if (tmp1 == tmp2 | {0, tmp1} == tmp4 | tmp3 == {0, tmp2}) begin     //希望tmp1-tmp2为5的倍数，而这仅有上述三种可能，即差+5,0,-5
        ismultiple5 = 1;
    end
    else begin
        ismultiple5 = 0;
    end
end

// End of your code
endmodule
```

> 注：HalfAdder, FullAdder, adder2bit 模块代码与必做第 2 题中完全相同

仿真

```
module multiple5_tb();
reg [7:0] num;
wire ismultiple5;

initial begin
    num = 8'b00000000;
    #10 num = 8'b01000110;
    #10 num = 8'b10001011;
    #10 num = 8'b10110110;
    #10 num = 8'b11001101;
end

multiple5 multiple5(
    .num(num),
    .ismultiple5(ismultiple5)
);
endmodule
```

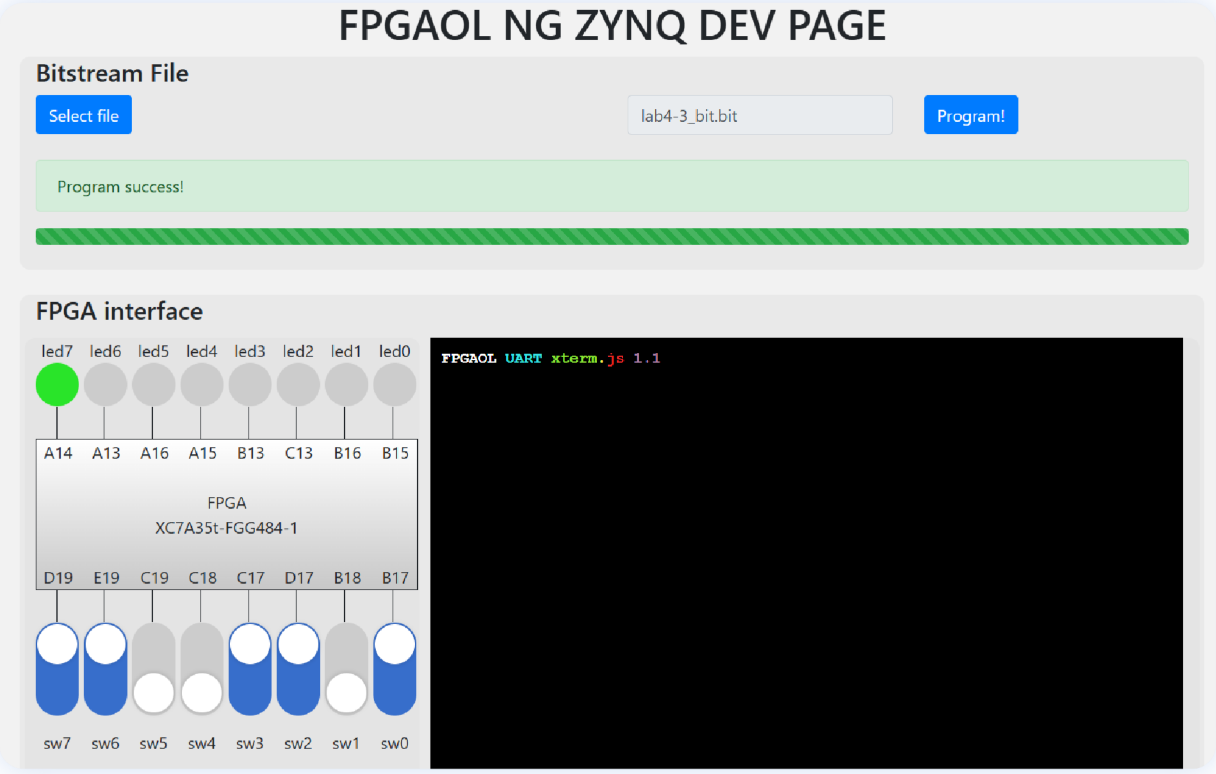


SIMULATION - Behavioral Simulation - Functional - sim_1 - multiple5_tb

Untitled 3*

Name	Value
num[7:0	00000000
ism...e5	1
tmp1[2:	000
tmp2[2:	000
tmp3[3:	0101
tmp4[3:	0101

实现



FPGAOL NG ZYNQ DEV PAGE

Bitstream File

Select file

lab4-3_bit.bit

Program!

Program success!

FPGA interface

```
FPGAOL UART xterm.js 1.1
```