lab4 单周期CPU设计

曾郅琛　PB20071431　物理学院辅修计算机

一、 实验题目：

Lab4 单周期 CPU 设计

二、 实验目的：

理解 CPU 的结构和工作原理
掌握单周期 CPU 的设计和调试方法
熟练掌握数据通路和控制器的设计和描述方法

三、 实验平台：

Vivado

四、实验过程展示

1.寄存器堆使用

```
module register
#(parameter WIDTH=32)
(
input clk,we,
input [4:0]wa,ra0,ra1,ra2,
input [WIDTH-1:0]wd,
output [WIDTH-1:0]rd0,rd1,rd2
    );
reg [WIDTH-1:0] regfile[0:31];

assign rd0 = regfile[ra0];
assign rd1 = regfile[ra1];
assign rd2 = regfile[ra2];

always@(posedge  clk)
begin
    if(we)
    if(wa==5'b00000)regfile[wa]<=5'b00000;
    else
    regfile[wa]<=wd;
end

endmodule
```

```
module PCregister(
input[31:0]a,
input clk,rst,
output [31:0]b
    );
reg[31:0]PC;
always@(posedge clk,posedge rst)
begin
    if(rst)PC<=32'b0000_0000_0000_0000_0011_0000_0000_0000;
    else PC<=a;
end
assign b=PC;
endmodule
```

对lab2中的寄存器堆重新设计，加入一个读接口，并使wa恒为0

2.CPU代码模块

1）控制器control

```
module control
#(parameter
ADDI=7'b0010011,
ADD=7'b0110011,
SUB=7'b0110011,
AUIPC=7'b001011,
JAL=7'b1101111,
JALR=7'b1100111,
BEQ=7'b1100011,
BLT=7'b1100011,
LW=7'b0000011,
SW=7'b0100011)
(
input [6:0]in,
input branch,
output reg[2:0]select,
output reg mux1,mux2,
output reg[1:0]mux4,
output reg register_we,
output reg D_mem_we,
output reg D_mem_ena
    );
always@(*)
begin
    case(in)
    ADDI:begin
select=3'b000;mux1=1'b1;mux2=1'b1;mux4=2'b10;register_we=1'b1;D_mem_we=1'b0;D_me
m_ena=1'b0;end
    ADD:begin
select=3'b000;mux1=1'b1;mux2=1'b0;mux4=2'b10;register_we=1'b1;D_mem_we=1'b0;D_me
m_ena=1'b0;end
    AUIPC:begin
select=3'b000;mux1=1'b0;mux2=1'b1;mux4=2'b10;register_we=1'b1;D_mem_we=1'b0;D_me
m_ena=1'b0;end
```

```
    JAL:begin
select=3'b001;mux1=1'bx;mux2=1'bx;mux4=2'b00;register_we=1'b1;D_mem_we=1'b0;D_me
m_ena=1'b0;end
    JALR:begin
select=3'b111;mux1=1'b1;mux2=1'b1;mux4=2'b00;register_we=1'b1;D_mem_we=1'b0;D_me
m_ena=1'b0;end
    BEQ:if(branch)begin
select=3'b011;mux1=1'b1;mux2=1'b0;mux4=2'bxx;register_we=1'b0;D_mem_we=1'b0;D_me
m_ena=1'b0;end else begin
 select=3'b101;mux1=1'b1;mux2=1'b0;mux4=2'bxx;register_we=1'b0;D_mem_we=1'b0;D_m
em_ena=1'b0;end
    LW:begin
select=3'b000;mux1=1'b1;mux2=1'b1;mux4=2'b01;register_we=1'b1;D_mem_we=1'b0;D_me
m_ena=1'b1;end
    SW:begin
select=3'b000;mux1=1'b1;mux2=1'b1;mux4=2'b01;register_we=1'b0;D_mem_we=1'b1;D_me
m_ena=1'b0;end
    default:begin
select=3'b000;mux1=1'bx;mux2=1'bx;mux4=2'bxx;register_we=1'b0;D_mem_we=1'b0;D_me
m_ena=1'b0;end
    endcase
end
endmodule
```

通过设计控制器来进行各个操作`addi,add,sub,jal,jarl,lw,sw,beq`的选择控制。

2)PC

```
module PCregister(
input[31:0]a,
input clk,rst,
output [31:0]b
    );
reg[31:0]PC;
always@(posedge clk,posedge rst)
begin
    if(rst)PC<=32'b0000_0000_0000_0000_0011_0000_0000_0000;
    else PC<=a;
end
assign b=PC;
endmodule
```

设置PC计数器，高电平传输时PC+4

3）立即数Immidiate模块

```
module imm
#(parameter
ADDI=7'b0010011,
ADD=7'b0110011,
SUB=7'b0110011,
AUIPC=7'b001011,
JAL=7'b1101111,
JALR=7'b1100111,
BEQ=7'b1100011,
BLT=7'b1100011,
```

```
LW=7'b0000011,
SW=7'b0100011)
(
input[31:0]in,
output reg[31:0]data
    );
always@(*)
begin
    case(in[6:0])
    ADDI:begin if(~in[31])data={20'b0,in[31:20]};else data=
{20'b1111_1111_1111_1111_1111,in[31:20]};end
    AUIPC:data={in[31:12],12'b0};
    JAL:begin if(~in[31])data=
{11'b0,in[31],in[19:12],in[20],in[30:21],1'b0};else data=
{11'b1111_1111_111,in[31],in[19:12],in[20],in[30:21],1'b0};end
    JALR:begin if(~in[31])data={20'b0,in[31:20]};else data=
{20'b1111_1111_1111_1111_1111,in[31:20]};end
    BEQ:begin if(~in[31])data={19'b0,in[31],in[7],in[30:25],in[11:8],1'b0};else
data={19'b1111_1111_1111_1111_111,in[31],in[7],in[30:25],in[11:8],1'b0};end
    LW:begin if(~in[31])data={20'b0,in[31:20]};else data=
{20'b1111_1111_1111_1111_1111,in[31:20]};end
    SW:begin if(~in[31])data={20'b0,in[31:25],in[11:7]};else data=
{20'b1111_1111_1111_1111_1111,in[31:25],in[11:7]};end
    default:data=32'b0;
    endcase
end
endmodule
```

根据操作数完成对应操作

4）ALU模块

```
module alu#(parameter WIDTH=32)
(
input [WIDTH-1:0]a,b,
input sel,
output wire[WIDTH-1:0]c,
output zero,less
    );
wire d;
assign d=c[31]&1'b1;
assign c= sel==0? a+b:a-b;
assign zero= c==0? 1:0;
assign less= d==1? 1:0;
endmodule

module alu_control
#(parameter
ADDI=7'b0010011,
ADD=7'b0110011,
SUB=7'b0110011,
AUIPC=7'b0010111,
JAL=7'b1101111,
JALR=7'b1100111,
BEQ=7'b1100011,
BLT=7'b1100011,
LW=7'b0000011,
```

```
SW=7'b0100011)
(
input [6:0]in,
input In,
output reg sel
    );
always@(*)
begin
    case(in)
    SUB:if(In)sel=1'b1;else sel=1'b0;
    default:sel=1'b0;
    endcase
end
endmodule
```

算术逻辑运算器根据7位二进制操作码执行算术运算

5）PDU模块

展示在老师框架下的主要操作：

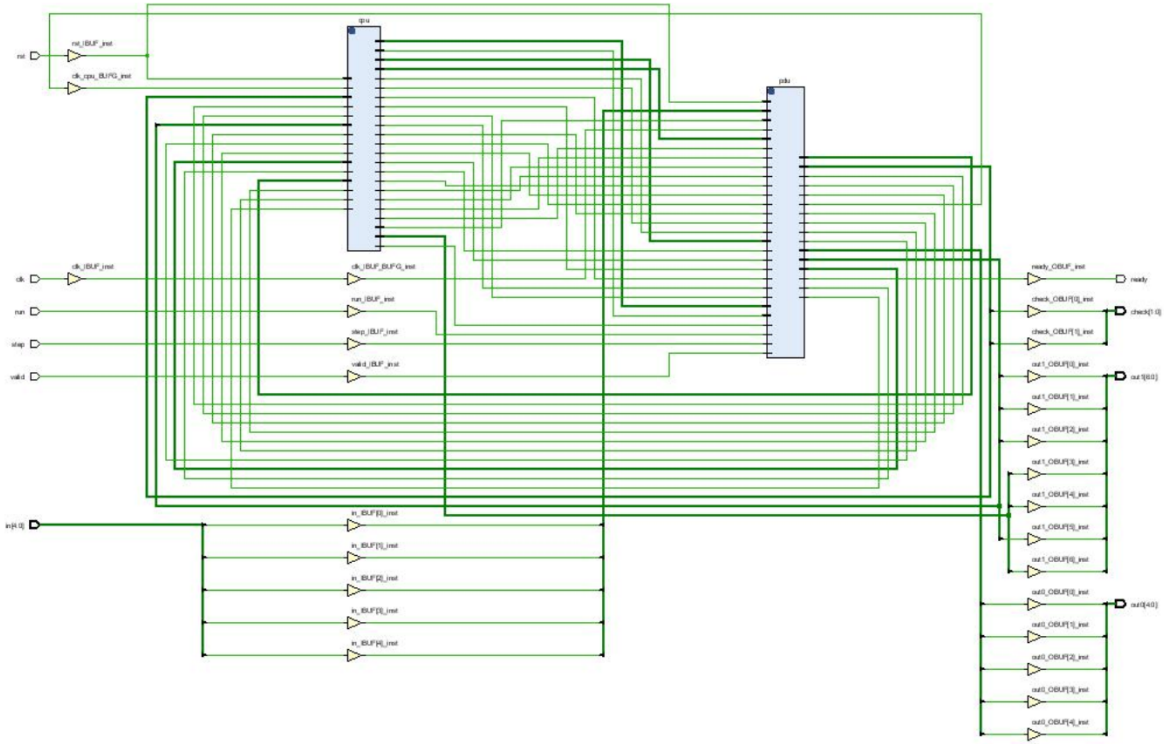
```
/CPU工作方式
always @(posedge clk, posedge rst) begin
  if(rst)
    clk_cpu_r <= 0;
  else if (run_r)
    clk_cpu_r <= ~clk_cpu_r;
  else
    clk_cpu_r <= step_p;
end

//读外设端口
always @* begin
  case (io_addr)
    8'h0c: io_din_a = {{27{1'b0}}, in_r};
    8'h10: io_din_a = {{31{1'b0}}, valid_r};
    default: io_din_a = 32'h0000_0000;
  endcase
end

//写外设端口
always @(posedge clk, posedge rst) begin
if (rst) begin
  out0_r <= 5'h1f;
  out1_r <= 32'h1234_5678;
  ready_r <= 1'b1;
end
else if (io_we)
  case (io_addr)
    8'h00: out0_r <= io_dout[4:0];
    8'h04: ready_r <= io_dout[0];
    8'h08: out1_r <= io_dout;
    default: ;
  endcase
end

//LED和数码管查看类型
```

```
always @(posedge clk, posedge rst) begin
if(rst)
    check_r <= 2'b00;
  else if(run_r)
    check_r <= 2'b00;
  else if (step_p)
    check_r <= 2'b00;
  else if (valid_pn)
    check_r <= check - 2'b01;
end

//LED和数码管显示内容
always @* begin
  ready_a = 1'b0;
  case (check_r)
    2'b00: begin
      out0_a = out0_r;
      out1_a = out1_r;
      ready_a = ready_r;
    end
    2'b01: begin
      out0_a = in_r;
      out1_a = rf_data;
    end
    2'b10: begin
      out0_a = in_r;
      out1_a = m_data;
    end
    2'b11: begin
      out0_a = 5'b00000;
      out1_a = pc;
    end
  endcase
end

//扫描数码管
always @(posedge clk, posedge rst) begin
  if (rst) cnt <= 20'h0_0000;
  else cnt <= cnt + 20'h0_0001;
end

always @* begin
  case (an)
    3'd0: seg_a = out1_a[3:0];
    3'd1: seg_a = out1_a[7:4];
    3'd2: seg_a = out1_a[11:8];
    3'd3: seg_a = out1_a[15:12];
    3'd4: seg_a = out1_a[19:16];
    3'd5: seg_a = out1_a[23:20];
    3'd6: seg_a = out1_a[27:24];
    3'd7: seg_a = out1_a[31:28];
    default: ;
  endcase
end
```

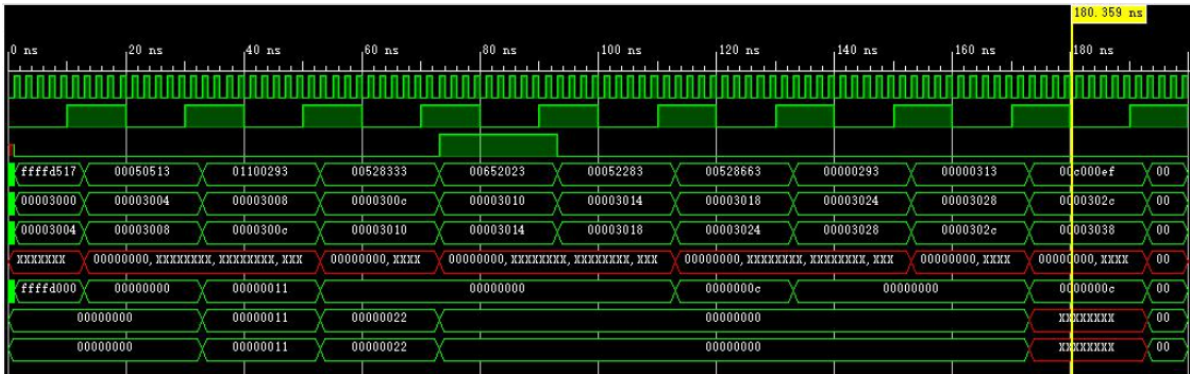
6）顶层文件

```
module top(
input run,step,valid,rst,clk,
input[4:0] in,
output[1:0]check,
output[4:0]out0,
output ready,
output [2:0]an,
output [3:0]seg
    );
    wire clk_cpu,clk_rst;
    wire[31:0]pc;
    wire[7:0]m_rf_addr;
    wire[31:0]rf_data;
    wire[31:0]m_data;
    wire[7:0]io_addr;
    wire[31:0]io_dout;
    wire[31:0]io_din;
    wire io_we;
pdu(clk,rst,run,step,clk_cpu,valid,in,check,out0,an,seg,ready,io_addr,io_dout,io
_we,io_din,m_rf_addr,rf_data,m_data,pc);
cpu(clk_cpu,rst,pc,m_rf_addr,rf_data,m_data,io_addr,io_dout,io_din,io_we);
endmodule
```

三、数据通路



四、波形仿真

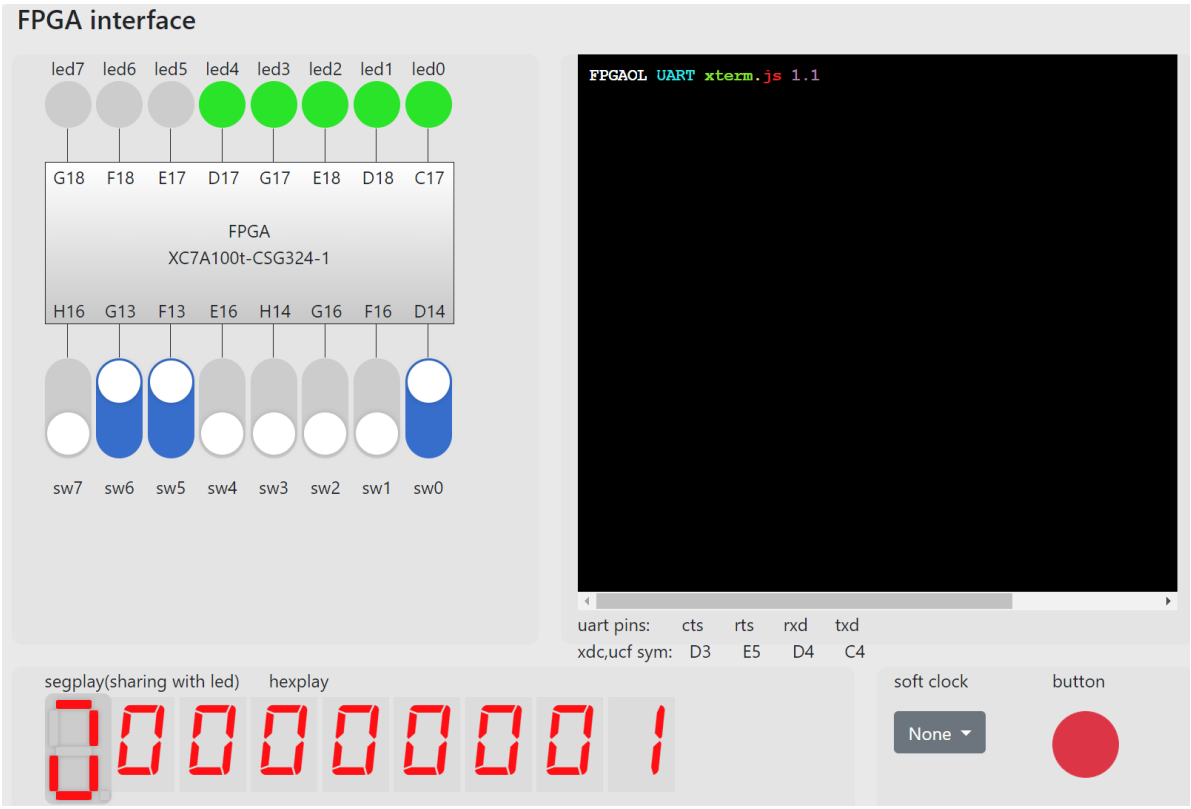


五、FPGA测试

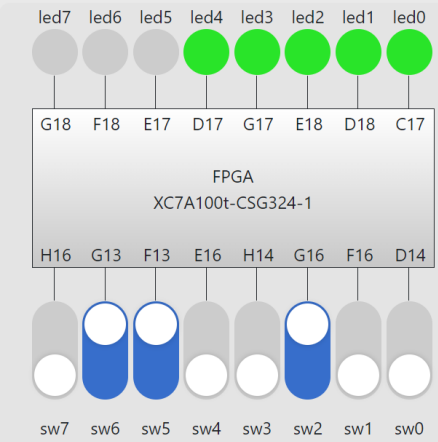
1）现场检验PC、寄存器值与RARS对照

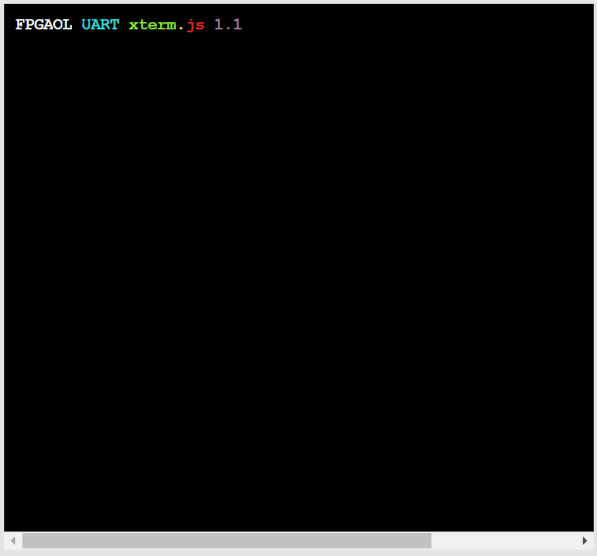
单步执行检验地址，现场检验正确。

2）斐波拉契数



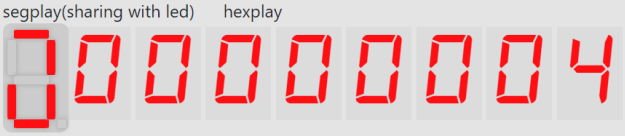
FPGA interface





uart pins: cts rts rxd txd
xdc,ucf sym: D3 E5 D4 C4

segplay(sharing with led) hexplay



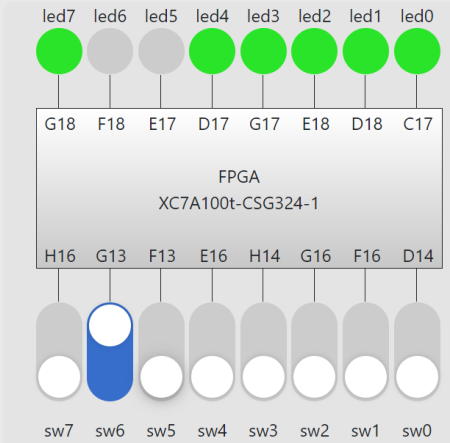
segplay pin: dot seg_g seg_f seg_e seg_d seg_c seg_b seg_a
xdc,ucf sym: G18 F18 E17 D17 G17 E18 D18 C17

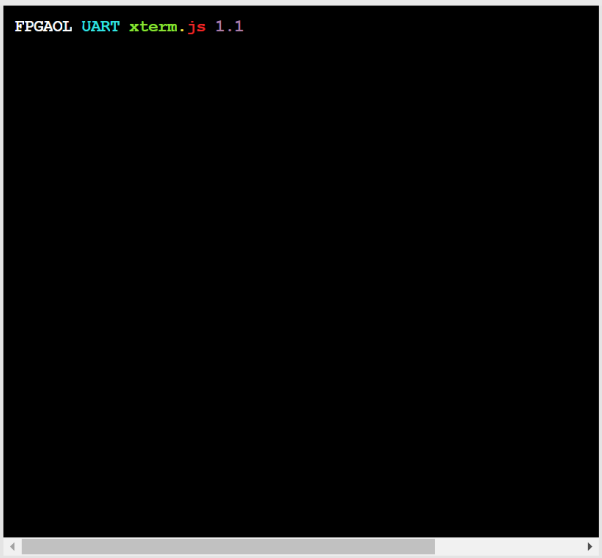
soft clock button

None

clk btn pins: clk_btn
xdc,ucf sym: B18

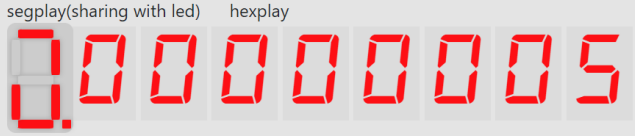
FPGA interface





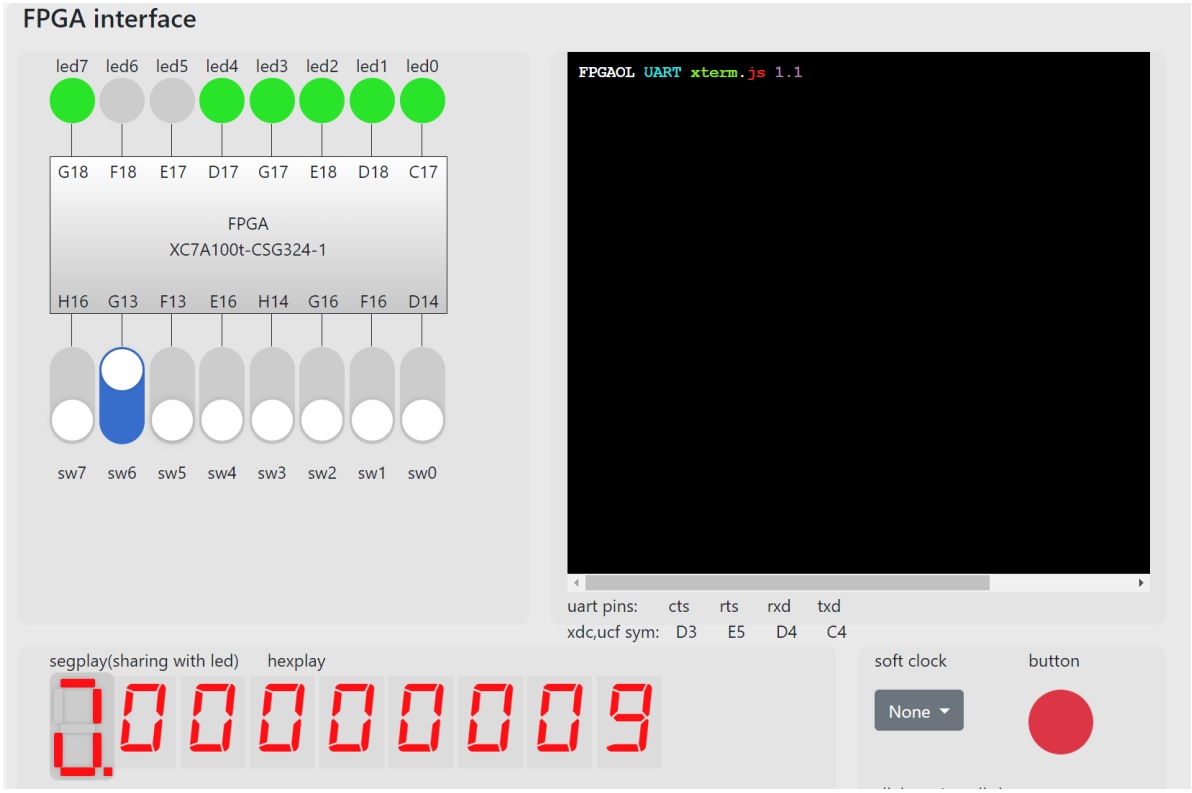
uart pins: cts rts rxd txd
xdc,ucf sym: D3 E5 D4 C4

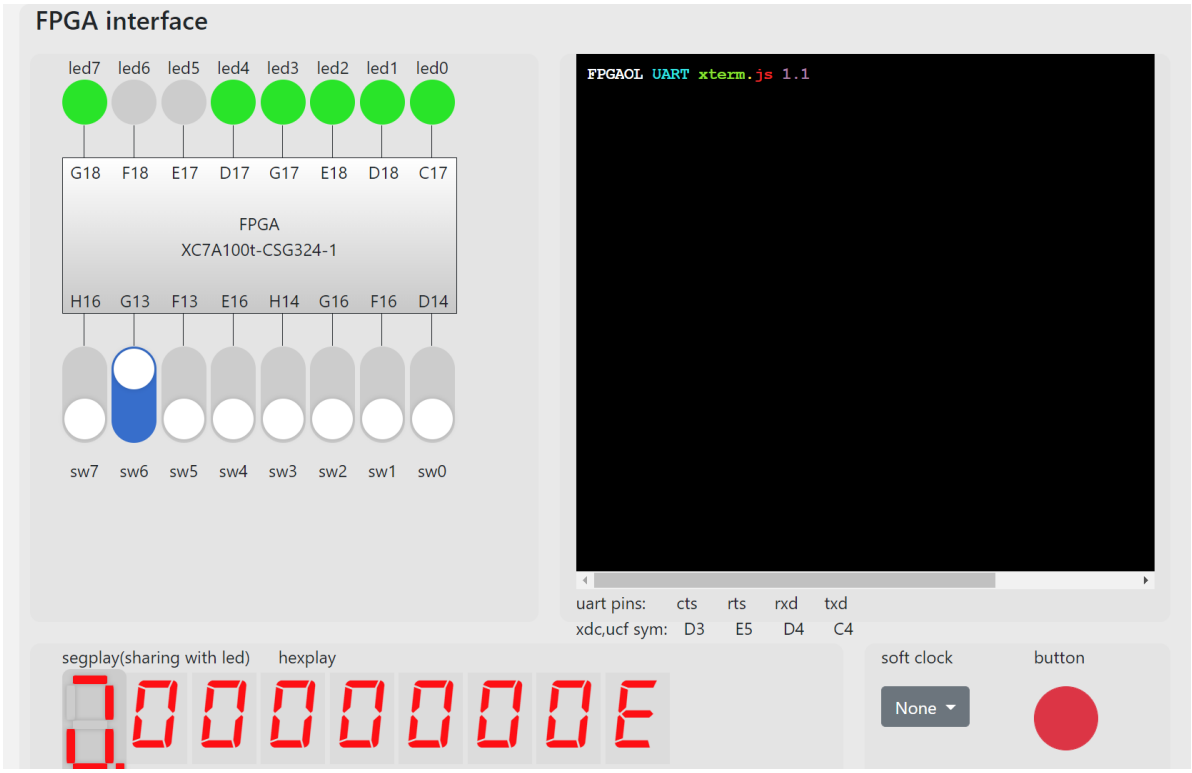
segplay(sharing with led) hexplay



soft clock button

None





取前两个数分别为1，4.得到后几个数值符合运算。