lab05 流水线CPU设计

曾郅琛　PB20071431　物理学院辅修计算机

一、 实验题目：

Lab05 流水线 CPU 设计

二、 实验目的：

理解流水线CPU的结构和工作原理

掌握流水线CPU的设计和调试方法，特别是流水线中数据相关和控制相关的处理

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.CPU_pl

```
module  cpu_pl (
  input clk,
  input rst,

  //IO_BUS
  output [7:0] io_addr,      //led和seg的地址
  output [31:0] io_dout,     //输出led和seg的数据
  output io_we,                 //输出led和seg数据时的使能信号
  input [31:0] io_din,        //来自sw的输入数据

  //Debug_BUS
  input [7:0] m_rf_addr,   //存储器(MEM)或寄存器堆(RF)的调试读口地址
  output [31:0] rf_data,    //从RF读取的数据
  output [31:0] m_data,    //从MEM读取的数据

  //PC/IF/ID 流水段寄存器
  output [31:0] pc,
  output [31:0] pcd,
  output [31:0] ir,
  output [31:0] pcin,

  // ID/EX 流水段寄存器
  output [31:0] pce,
  output [31:0] a,
  output [31:0] b,
  output [31:0] imm,
  output [4:0] rd,
  output [31:0] ctrl,

  // EX/MEM 流水段寄存器
```

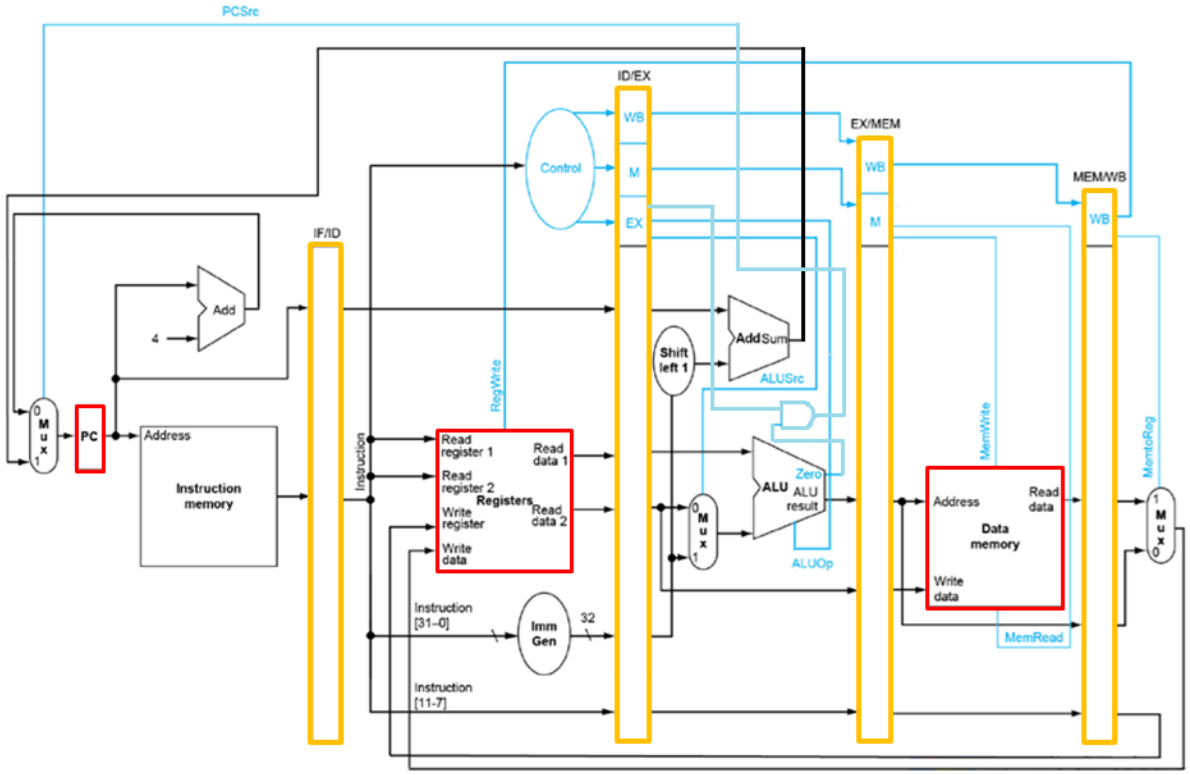
```
  output [31:0] y,
  output [31:0] bm,
  output [4:0] rdm,
  output [31:0] ctrlm,

  // MEM/WB 流水段寄存器
  output [31:0] yw,
  output [31:0] mdr,
  output [4:0] rdw,
  output [31:0] ctrlw
);
//IF信号
wire [31:0]nouse0,nouse1,nouse2,nouse3,nouse4,nouse5,nouse6;
wire [31:0]ir0,pc4;
wire [31:0]pcsrc;//beq or jump
wire pcsel;//
wire stop,rst_IF_ID;//////////////

wire[31:0]pc4d;
//IF
mux2 pcmux(pc4,pcsrc,pcsel,pcin);
PCregister PC(pcin,clk,rst,stop,pc);
add pcadd(pc,32'b0100,pc4);
I_mem Imem(pc[9:2],32'b0,clk,1'b0,ir0);
SegmentRegister
IF_ID({32'b0,pc,pc4,32'b0,32'b0,ir0,32'b0},clk,rst_IF_ID,stop,nouse0,pcd,pc4d,no
use1,nouse2,ir,nouse3);


wire RegWrite,rst_ID_EX,rst_ID_EX0,rst_ID_EX1;
wire [31:0]wd,rd1,rd2,immd,ctrld,pc4e,ire;
register
registerfile(clk,RegWrite,rdw,ir[19:15],ir[24:20],m_rf_addr[4:0],wd,rd1,rd2,rf_d
ata);
Imm Immnumber(ir,immd);
control Control(ir[6:0],ctrld);
SegmentRegister
ID_EX({ctrld,pcd,pc4d,rd1,rd2,ir,immd},clk,rst_ID_EX,1'b0,ctrl,pce,pc4e,a,b,ire,
imm);
assign rd=ire[11:7];
Hazard_Detection_Unit
Hazard_Detection(ir[19:15],ir[24:20],ire[11:7],ctrl[13],stop,rst_ID_EX0);

wire [31:0]rs2_or_imm,alunum1,alunum2,aluresult,irm,irw;
wire [1:0]afwd,bfwd;
wire zero;
wire [31:0]pc4m;

mux2 rd2_imm(alunum2,imm,ctrl[4],rs2_or_imm);
add addar(pce,imm,pcsrc);
mux3 fwda(a,y,wd,afwd,alunum1);
mux3 fwdb(b,y,wd,bfwd,alunum2);
Forwarding_Unit
forwarding_unit(ctrlm[18],ctrlw[18],ire[19:15],ire[24:20],irm[11:7],irw[11:7],af
wd,bfwd);
alu ALU(alunum1,rs2_or_imm,ctrl[0],aluresult,zero);
assign rst_ID_EX1 = (zero&ctrl[8])|ctrl[9];
assign rst_ID_EX = rst_ID_EX0|rst_ID_EX1;
```

```
assign rst_IF_ID = (zero&ctrl[8])|ctrl[9];
assign pcsel = (zero&ctrl[8])|ctrl[9];
SegmentRegister
EX_MEM({ctrl,32'b0,pc4e,aluresult,alunum2,ire,32'b0},clk,1'b0,1'b0,ctrlm,nouse4,
pc4m,y,bm,irm,nouse5);
assign rdm=irm[11:7];

wire [31:0]lw_data,pc4w,bmw;
wire mem_write;
assign mem_write=ctrlm[12]&~y[10];
D_mem Dmem(y[9:2],bm,m_rf_addr,clk,mem_write,lw_data,m_data);
SegmentRegister
MEM_WB({ctrlm,bm,pc4m,lw_data,y,irm,32'b0},clk,1'b0,1'b0,ctrlw,bmw,pc4w,mdr,yw,i
rw,nouse6);
assign rdw=irw[11:7];

wire [31:0]wrdata;
mux2 mdr_or_io(mdr,io_din,yw[10],wrdata);
mux3 rfwd(pc4w,wrdata,yw,ctrlw[17:16],wd);
assign RegWrite=ctrlw[18];
assign io_addr=yw[7:0];
assign io_dout=bmw;
assign io_we=yw[10]&ctrlw[12];
endmodule
```

PDU模块老师已给出，只需把CPU与PDU连接，此处不展示。

2.流水段Register

```
module SegmentRegister(
input[223:0]in,
input clk,rst,wr,
output[31:0]ctrl,pc,pc4,a,b,ir,imm
    );
reg[223:0]out;
initial out=0;
always@(posedge clk)
begin
    if(rst)out<=0;
    else if(!wr)out<=in;
    else out<=out;
end
assign imm=out[31:0];
assign ir=out[63:32];
assign b=out[95:64];
assign a=out[127:96];
assign pc4=out[159:128];
assign pc=out[191:160];
assign ctrl=out[223:192];
endmodule
```

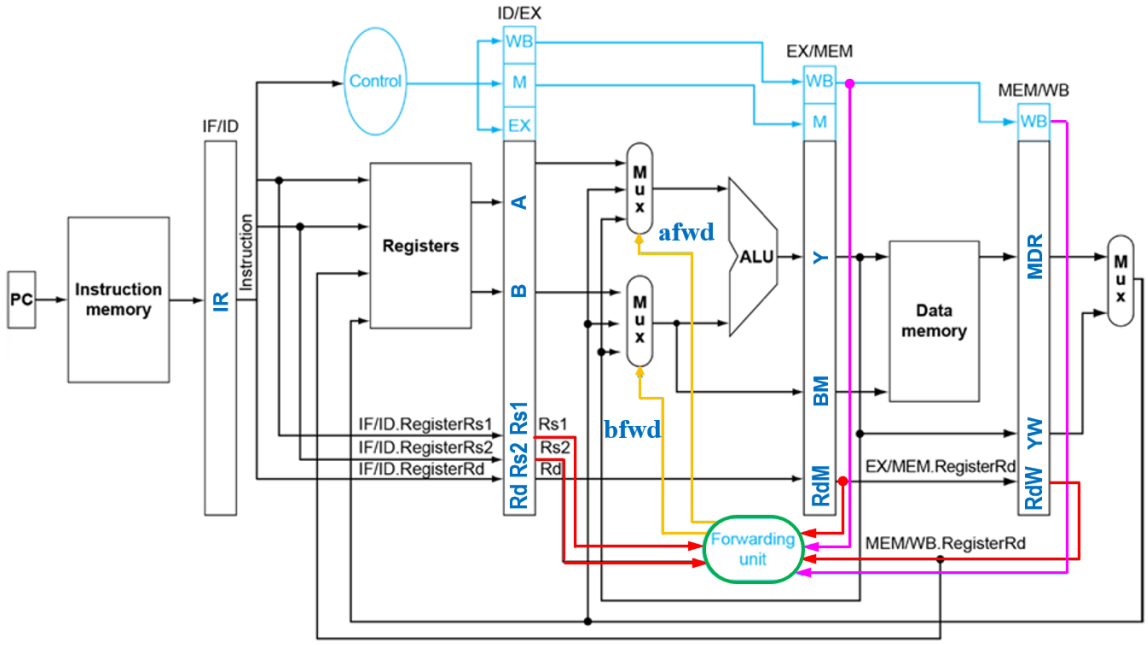


如图，在PC/IF/ID，ID/EX，EX/MEM ，MEM/WB 流水段插入寄存器，满足流水线需要。

3.Forwarding模块

```
module Forwarding_Unit(
    input EX_MEM_RegWrite,
    input MEM_WB_RegWrite,
    input [4:0] ID_EX_RegRs1,
    input [4:0] ID_EX_RegRs2,
    input [4:0] EX_MEM_RegRd,
    input [4:0] MEM_WB_RegRd,
    output [1:0]ForwardA,
    output [1:0]ForwardB
    );
reg [1:0]Reg_ForwardA;
reg [1:0]Reg_ForwardB;

always@(*)
begin
    if(EX_MEM_RegWrite && (EX_MEM_RegRd != 0)
      && (EX_MEM_RegRd == ID_EX_RegRs1) )
       Reg_ForwardA <= 1;
    else if(MEM_WB_RegWrite && (MEM_WB_RegRd != 0)
      && (MEM_WB_RegRd == ID_EX_RegRs1)&&~(EX_MEM_RegWrite &&
(EX_MEM_RegRd!=0)&&(EX_MEM_RegRd==ID_EX_RegRs1)) )
       Reg_ForwardA <= 2;
    else
       Reg_ForwardA <= 0;
end

always@(*)
begin
     if(EX_MEM_RegWrite && (EX_MEM_RegRd != 0)
      && (EX_MEM_RegRd == ID_EX_RegRs2) )
       Reg_ForwardB <= 1;
```

```
      else if(MEM_WB_RegWrite && (MEM_WB_RegRd != 0)
       && (MEM_WB_RegRd == ID_EX_RegRs2)&&~(EX_MEM_RegWrite &&
(EX_MEM_RegRd!=0)&&(EX_MEM_RegRd==ID_EX_RegRs2)) )
        Reg_ForwardB <= 2;
      else
        Reg_ForwardB <= 0;
end

assign ForwardA = Reg_ForwardA;
assign ForwardB = Reg_ForwardB;
endmodule
```



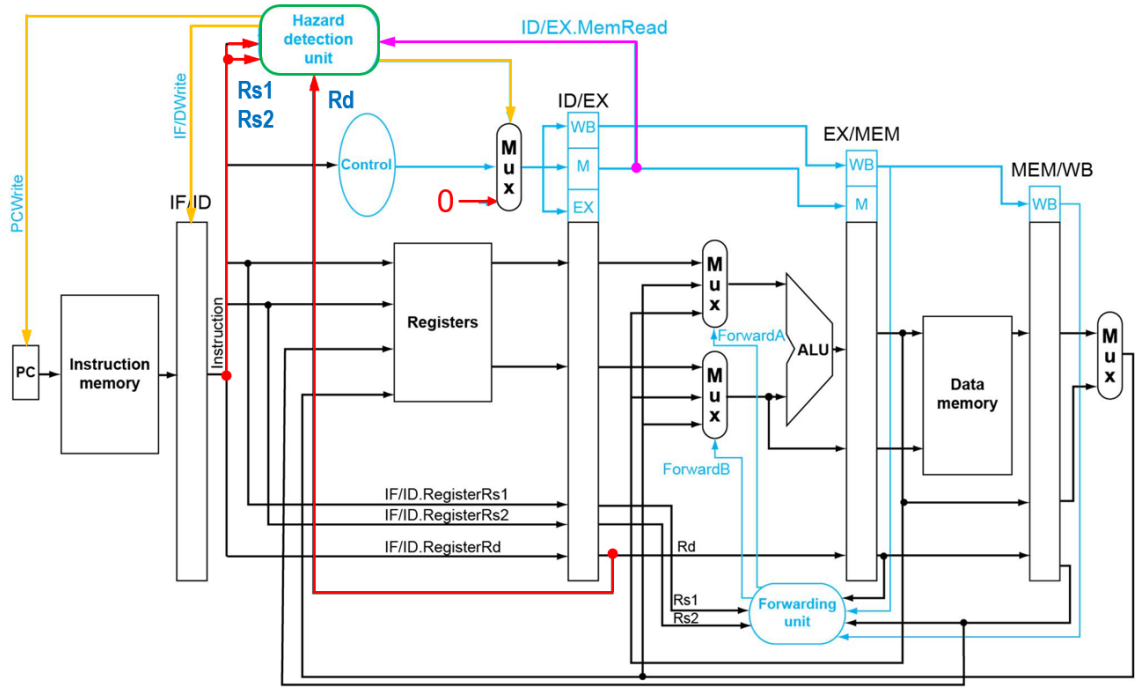
可能存在数据相关，即当一条指令需要等待前面指令的执行结果时，这个module利用数据定向（Forwarding)将执行结果提前传递至之前流水段，从而解决数据相关问题。

4.Hazard模块

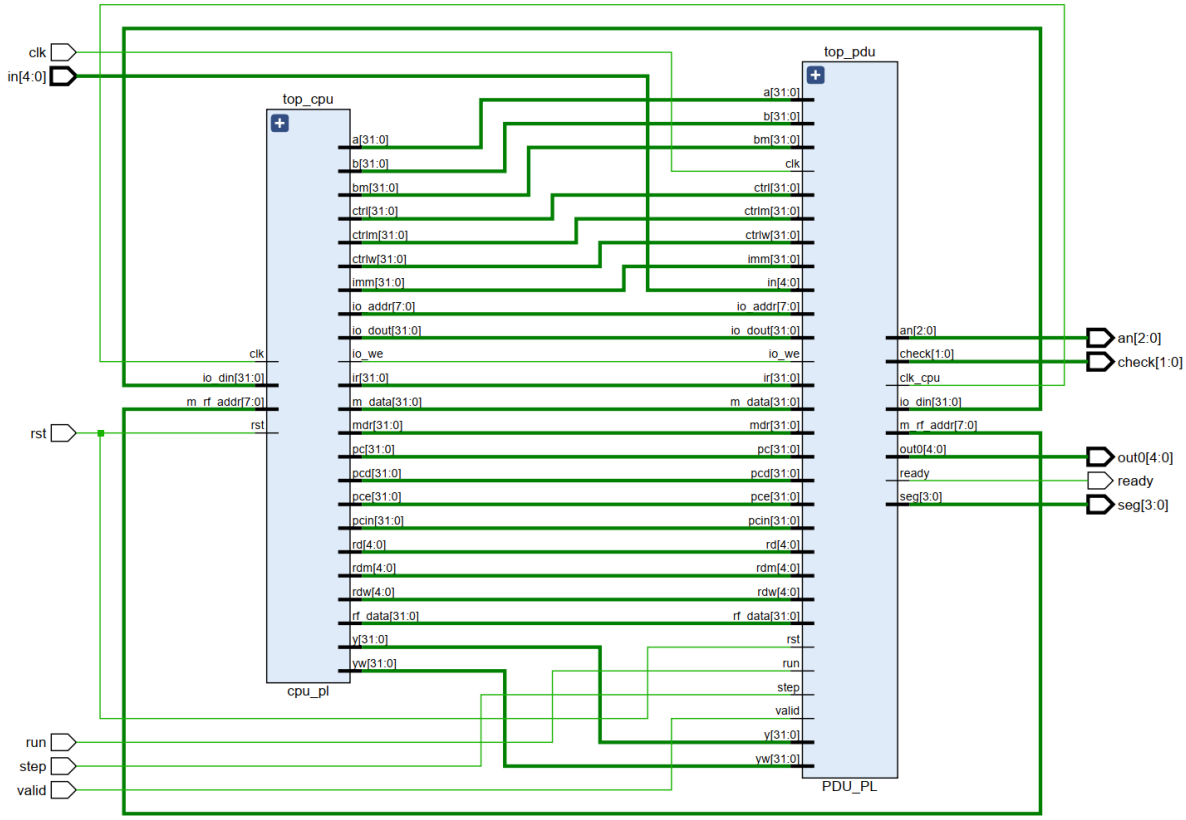
```
module Hazard_Detection_Unit(
    input [4:0] IF_ID_RegRs1,
    input [4:0] IF_ID_RegRs2,
    input [4:0] ID_EX_RegRd,
    input ID_EX_MemRead,
    output reg stop,rst
    );
    always@(*)
    begin
    if(ID_EX_MemRead&&((IF_ID_RegRs1!=0&&IF_ID_RegRs1==ID_EX_RegRd)||
(IF_ID_RegRs2!=0&&IF_ID_RegRs2==ID_EX_RegRd)))
    begin
    stop=1'b1;
    rst=1'b1;
    end
    else
    begin
    stop=1'b0;
    rst=1'b0;
    end
    end
```

```
endmodule
```

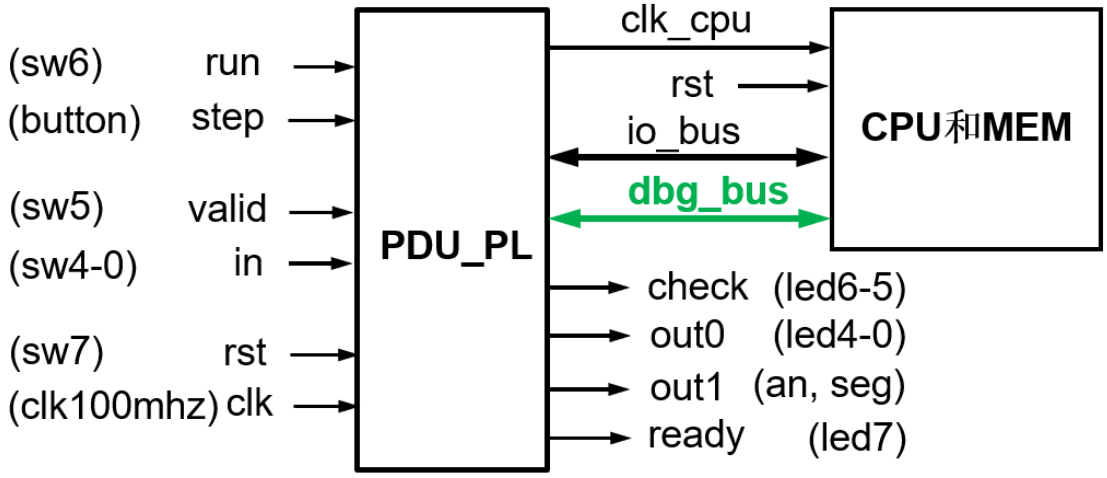
加载-使用相关（Load-use hazard），目的是阻止紧随Load已进入流水线的指令流动（Stall），向后续流水段插入空操作（Bubble）



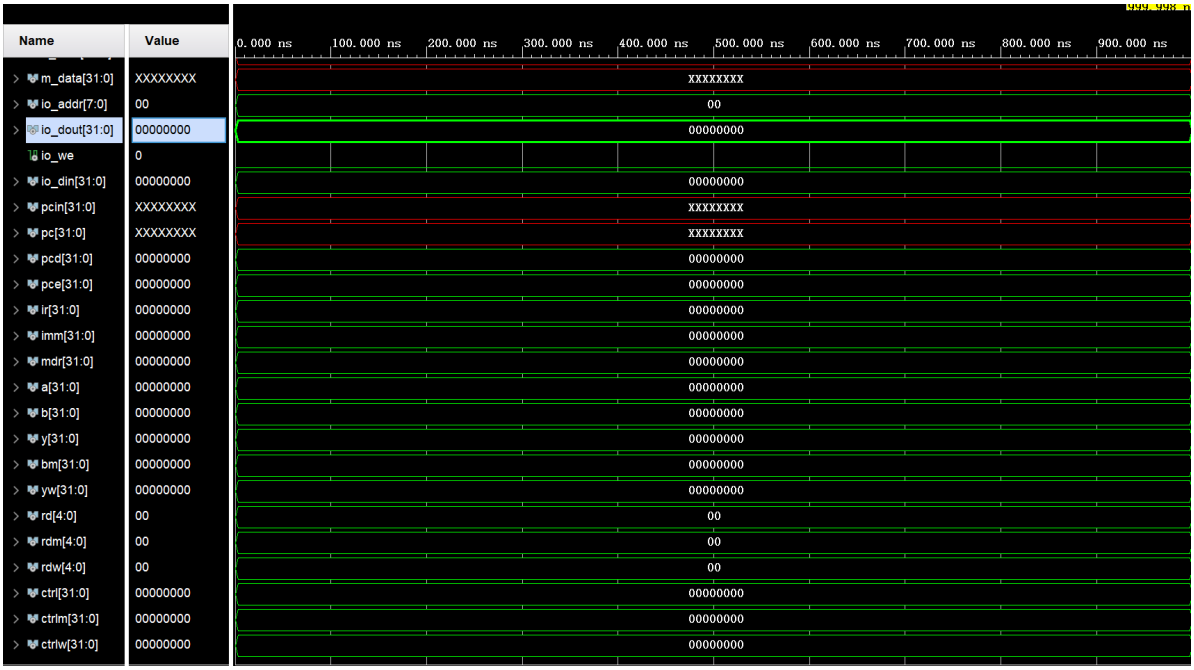
三、数据通路



寄存器堆和数据存储器均增加一个读端口用于调试



四、波形仿真

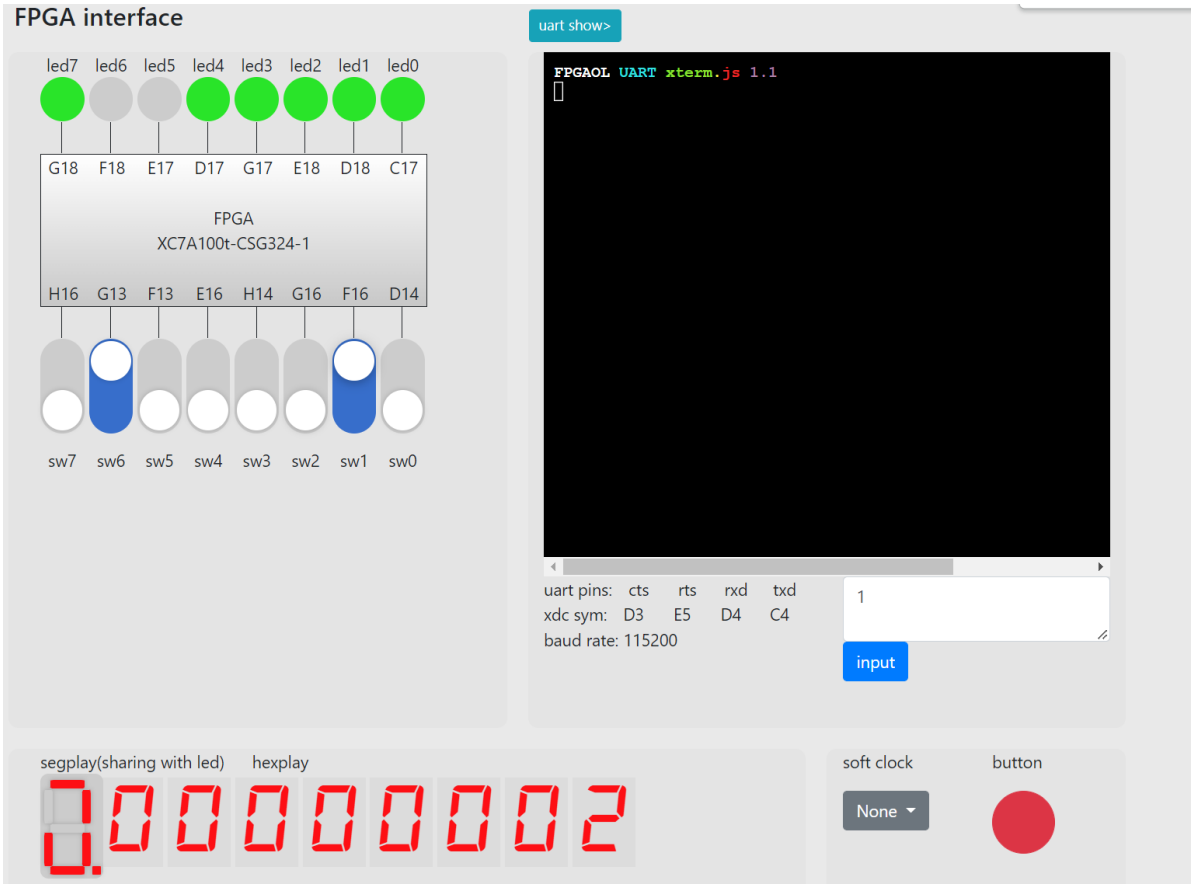


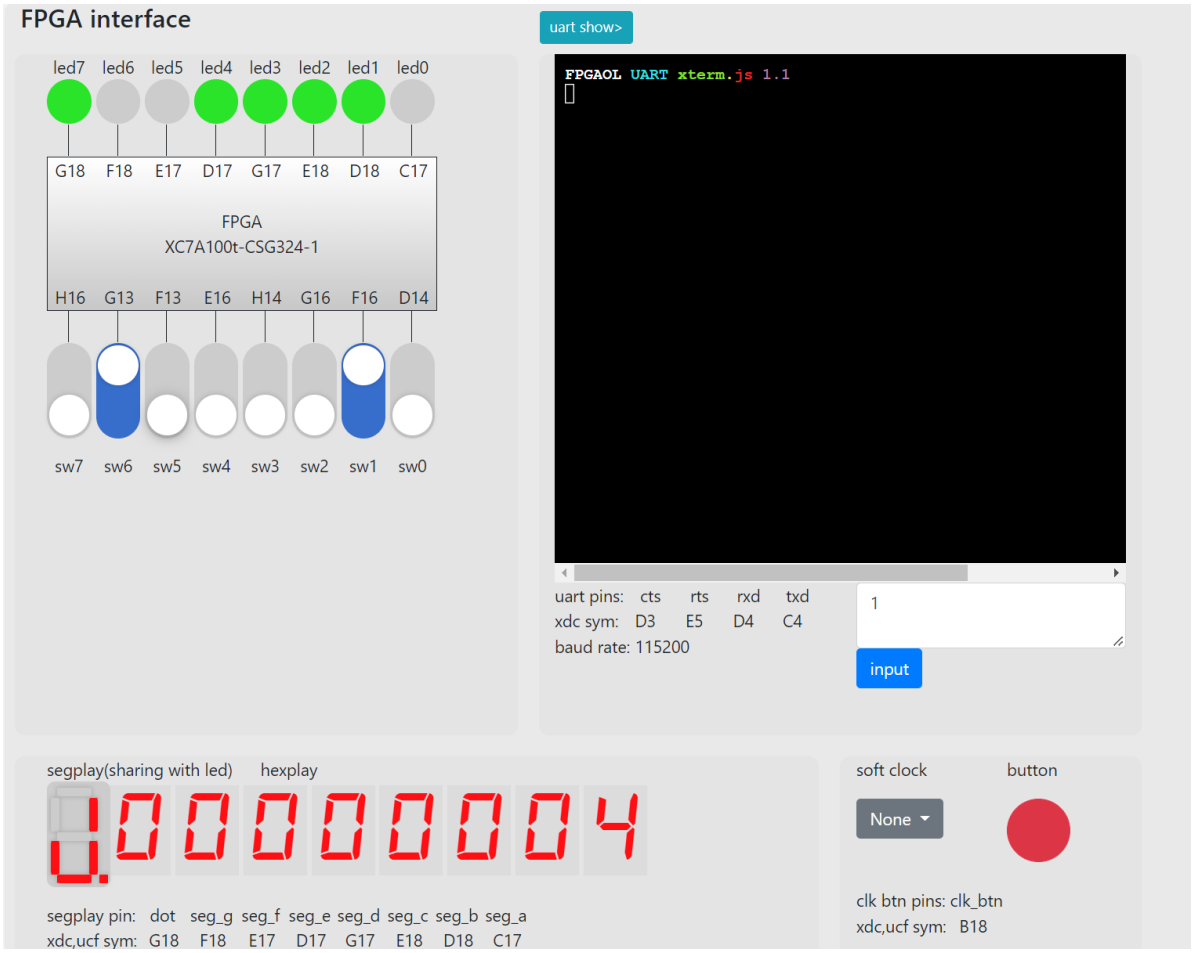
五、FPGA测试

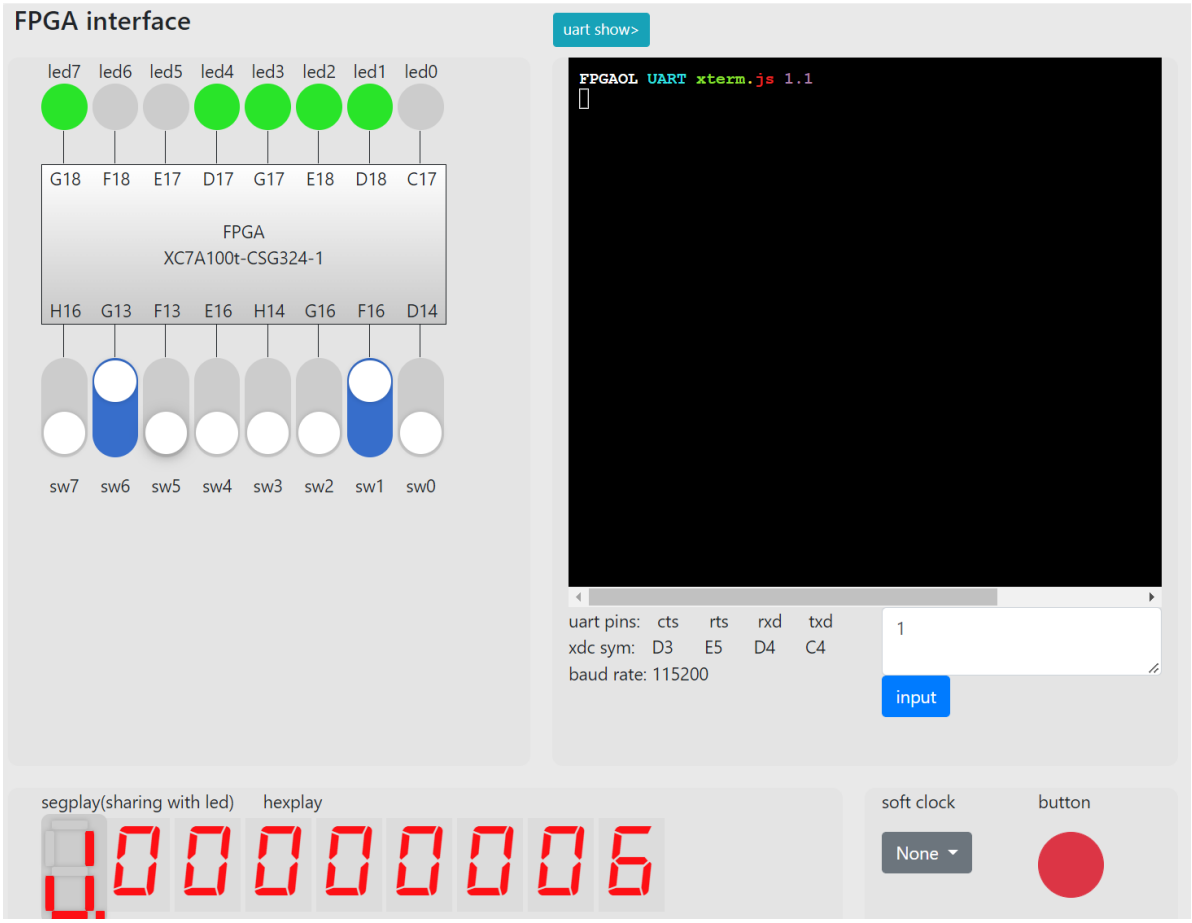
1）现场检验两个汇编测试程序（hazards_test.s，fib_test.s）

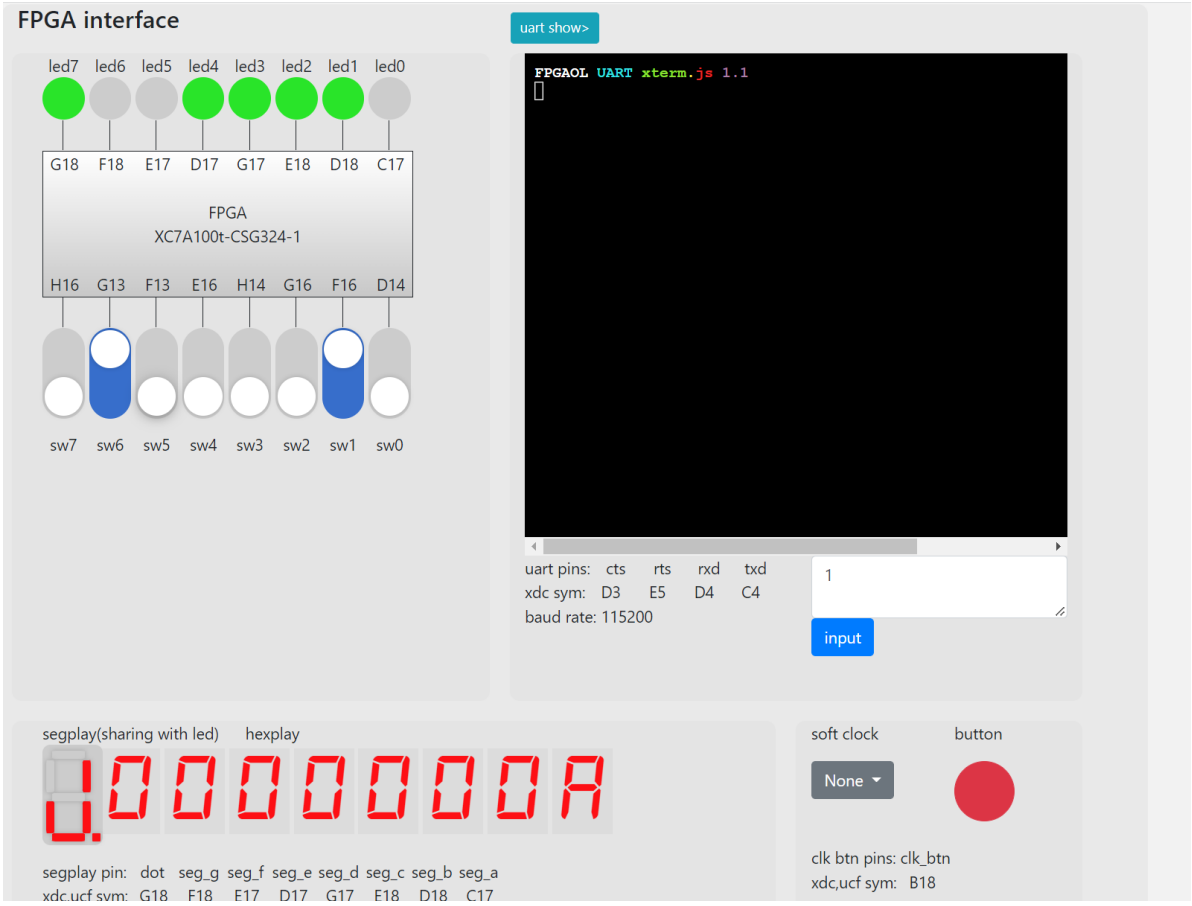
现场检验正确。

2）斐波拉契数检验


FPGA interface
uart show>
led7 led6 led5 led4 led3 led2 led1 led0
G18 F18 E17 D17 G17 E18 D18 C17
FPGA
XC7A100t-CSG324-1
H16 G13 F13 E16 H14 G16 F16 D14
sw7 sw6 sw5 sw4 sw3 sw2 sw1 sw0
FPGAOL UART xterm.js 1.1
uart pins: cts rts rxd txd
xdc sym: D3 E5 D4 C4
baud rate: 115200
1
input
segplay(sharing with led)
hexplay
soft clock
button
None


FPGA interface
uart show>
led7 led6 led5 led4 led3 led2 led1 led0
G18 F18 E17 D17 G17 E18 D18 C17
FPGA
XC7A100t-CSG324-1
H16 G13 F13 E16 H14 G16 F16 D14
sw7 sw6 sw5 sw4 sw3 sw2 sw1 sw0
FPGAOL UART xterm.js 1.1
uart pins: cts rts rxd txd
xdc sym: D3 E5 D4 C4
baud rate: 115200
1
input
segplay(sharing with led)
hexplay
soft clock
button
None
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
clk btn pins: clk_btn
xdc,ucf sym: B18





六、心得体会

本次实验以编写流水线CPU为主要目的，首先是写流水线reg时wire错误，但自己发现不了，后来问助教和同学才找到原因。然后.coe文件一定要加前面那一堆东西要不就完蛋，我自己搞忘记了，一直以为是bug。