中国科学技术大学计算机学院
《计算机组成原理》实验报告

实验题目：lab5_流水线 CPU

学生姓名：胡毅翔

学生学号：PB18000290

完成日期：2020 年 6 月 19 日

计算机实验教学中心制

2019 年 09 月

实验目的

1. 理解计算机硬件的基本组成、结构和工作原理
2. 掌握数字系统的设计和调试方法
3. 熟练掌握数据通路和控制器的设计和描述方法

实验环境

1. PC 一台
2. Windows 或 Linux 操作系统
3. Vivado
4. vlab.ustc.edu.cn
5. Logisim
6. Nexys4DDR 开发板

实验设计

流水线 CPU

设计目标

设计实现有相关处理的流水线 CPU 并可执行 add，addi，lw，sw，beq，j 共 6 条指令。

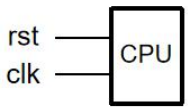

图 1

指令功能与格式如下所示：

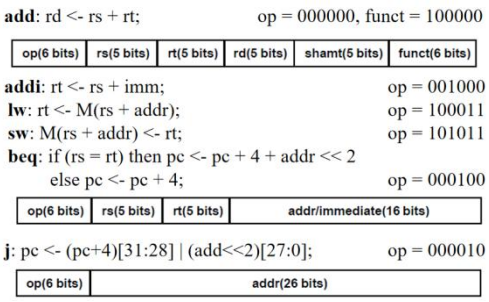

图 2

端口声明

端口声明如下：

```
module CPU(
        input clk, rst //clock,reset
      );
```

设计实现

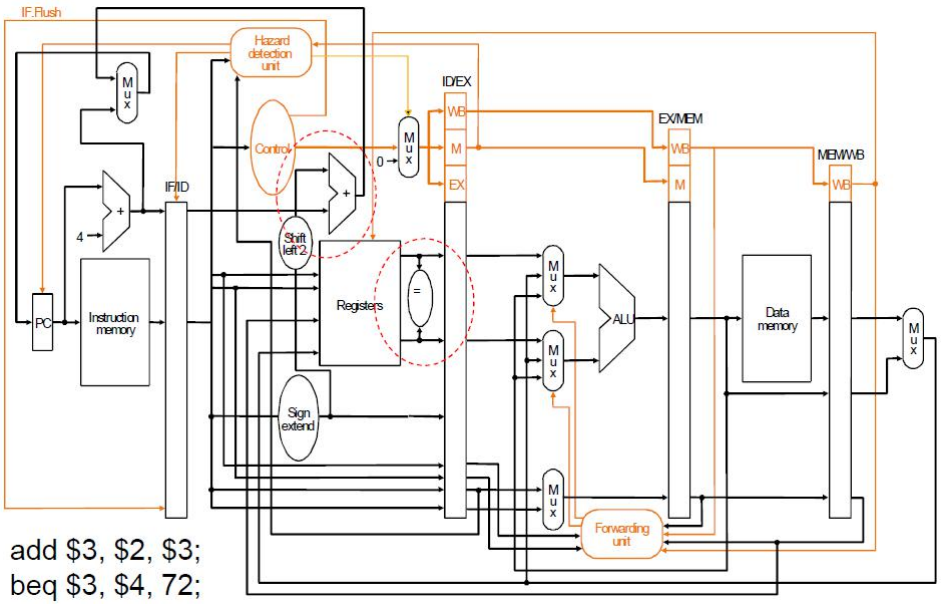

图 3

数据通路如上图所示。

而控制信号的输出，在译码段输出并保存到寄存器内，逐级传递到需要使用的地方。

用 Forwarding 和 Hazard 单元解决相关处理。

核心代码

核心代码(完整代码链接)(根据指令输出控制信号)如下：

ALU 控制及控制信号控制：

```
///alu control
always @ *
  begin
    case ( ID_EX_ALUOp_o )
      2'b00:
        alu_m = 3'b0;
      2'b01:
        alu_m = 3'b1;
      2'b10:
        if ( ID_EX_ir_o[ 5: 0 ] == 6'b100000 )
```

```
          begin
            alu_m = 3'b0;
          end
      default:
        alu_m = 3'b0;
    endcase
  end
//control
always @ *
  begin
    if ( rst )
      begin
        { ID_EX_ALUSrc_i, ID_EX_RegDst_i, ID_EX_MemWrite_i, ID_EX_MemRead_i, ID_EX_Branch_i, I
D_EX_RegWrite_i, ID_EX_MemtoReg_i, ID_EX_ALUOp_i } = 9'b000000000;
      end
    else
      begin
        case ( IF_ID_ir[ 31: 26 ] )
          //add
          6'b0:
            { ID_EX_ALUSrc_i, ID_EX_RegDst_i, ID_EX_MemWrite_i, ID_EX_MemRead_i, ID_EX_Branch_
i, ID_EX_RegWrite_i, ID_EX_MemtoReg_i, ID_EX_ALUOp_i } = 9'b010001010;
        //addi
        6'b001000:
          { ID_EX_ALUSrc_i, ID_EX_RegDst_i, ID_EX_MemWrite_i, ID_EX_MemRead_i, ID_EX_Branch_i,
 ID_EX_RegWrite_i, ID_EX_MemtoReg_i, ID_EX_ALUOp_i } = 9'b100001000;
        //lw
        6'b100011:
          { ID_EX_ALUSrc_i, ID_EX_RegDst_i, ID_EX_MemWrite_i, ID_EX_MemRead_i, ID_EX_Branch_i,
 ID_EX_RegWrite_i, ID_EX_MemtoReg_i, ID_EX_ALUOp_i } = 9'b100101100;
        //sw
        6'b101011:
          { ID_EX_ALUSrc_i, ID_EX_RegDst_i, ID_EX_MemWrite_i, ID_EX_MemRead_i, ID_EX_Branch_i,
 ID_EX_RegWrite_i, ID_EX_MemtoReg_i, ID_EX_ALUOp_i } = 9'b101000000;
        //beq
        6'b000100:
          { ID_EX_ALUSrc_i, ID_EX_RegDst_i, ID_EX_MemWrite_i, ID_EX_MemRead_i, ID_EX_Branch_i,
 ID_EX_RegWrite_i, ID_EX_MemtoReg_i, ID_EX_ALUOp_i } = 9'b000010001;
        //j
        6'b000010:
          { ID_EX_ALUSrc_i, ID_EX_RegDst_i, ID_EX_MemWrite_i, ID_EX_MemRead_i, ID_EX_Branch_i,
 ID_EX_RegWrite_i, ID_EX_MemtoReg_i, ID_EX_ALUOp_i } = 9'b000000000;
        default:
```

```
        { ID_EX_ALUSrc_i, ID_EX_RegDst_i, ID_EX_MemWrite_i, ID_EX_MemRead_i, ID_EX_Branch_i,
ID_EX_RegWrite_i, ID_EX_MemtoReg_i, ID_EX_ALUOp_i } = 9'b000000000;
     endcase
     end
 end
```

Forwarding 单元封装及 Hazard 单元控制：

```
FORWARD FORWARD0( ID_EX_ir_o[ 25: 21 ], ID_EX_ir_o[ 20: 16 ], EX_MEM_wa_o, MEM_WB_wa_o, EX_MEM
_RegWrite_o, MEM_WB_RegWrite_o, EX_MEM_MemRead_o, ALUSrcA, ALUSrcB );
mux4 muxa( alu_a, ID_EX_a_o, write_data, EX_MEM_y_o, MEM_WB_mdr_i, ALUSrcA );
mux4 muxb( alu_b, ID_EX_b_o, write_data, EX_MEM_y_o, MEM_WB_mdr_i, ALUSrcB );
//hazard
always @ *
 begin
   if ( PCSrc )
     begin
       { IF_ID_bubble, ID_EX_bubble, EX_MEM_bubble } <= 3'b111;
     end
   else
     begin
       { IF_ID_bubble, ID_EX_bubble, EX_MEM_bubble } <= 3'b000;
     end
 end
```

仿真结果

本次实验使用了两个不同的仿真用例。

第一个仿真用例的仿真结果如下，开始时执行 4 条 addi 指令，1 条 add 指令，及 1 条 lw 指令，在 beq 指令跳转后，不应执行的指令被 bubble 掉：

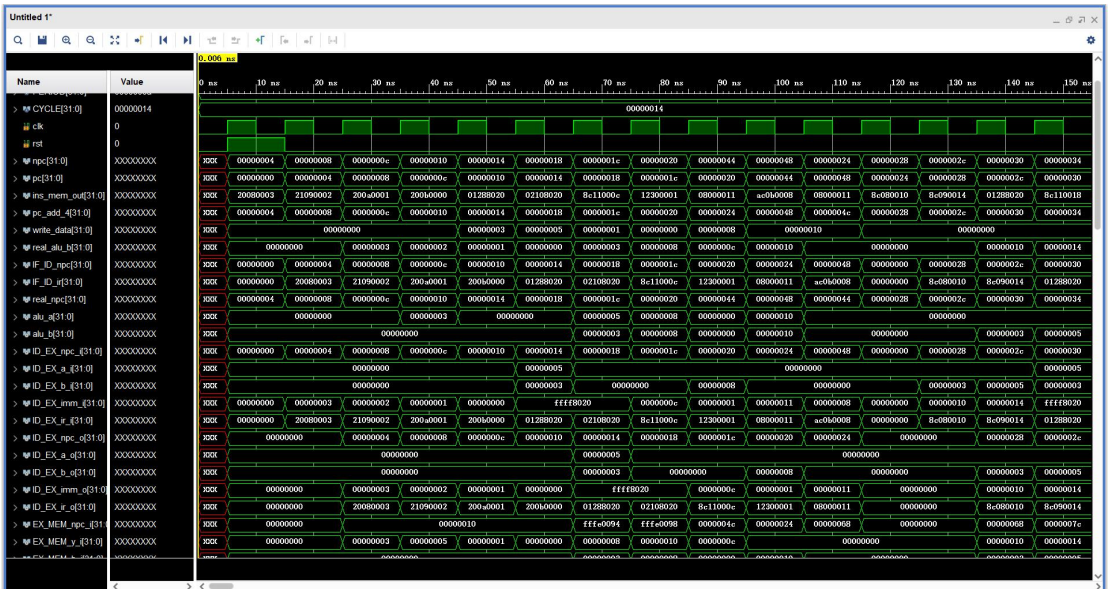

图 4

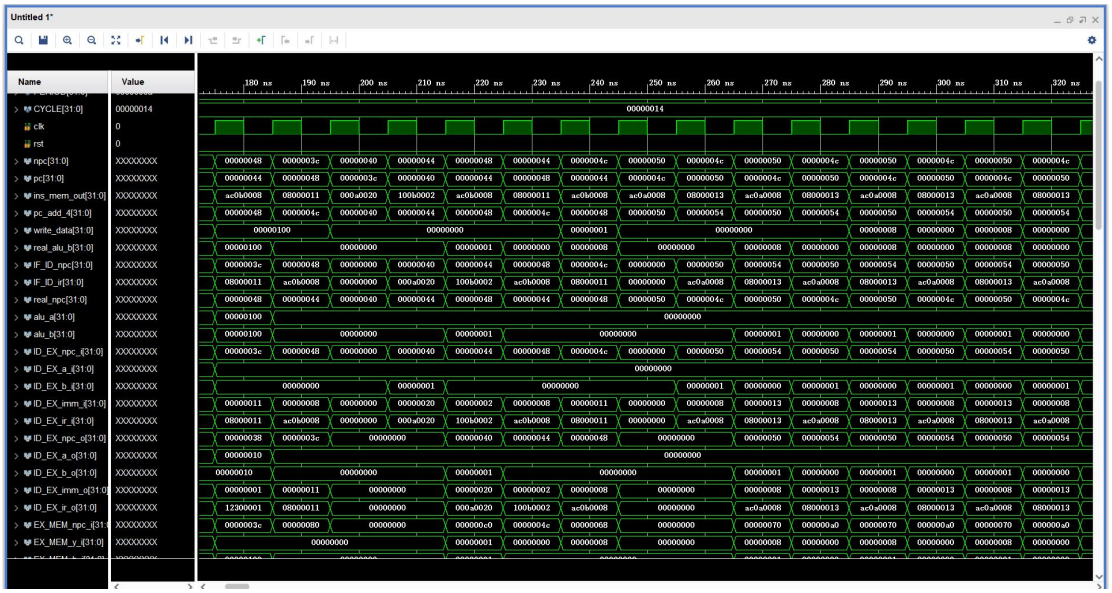

图 5

之后，继续执行其余指令，最后进入_success 循环，见上图。

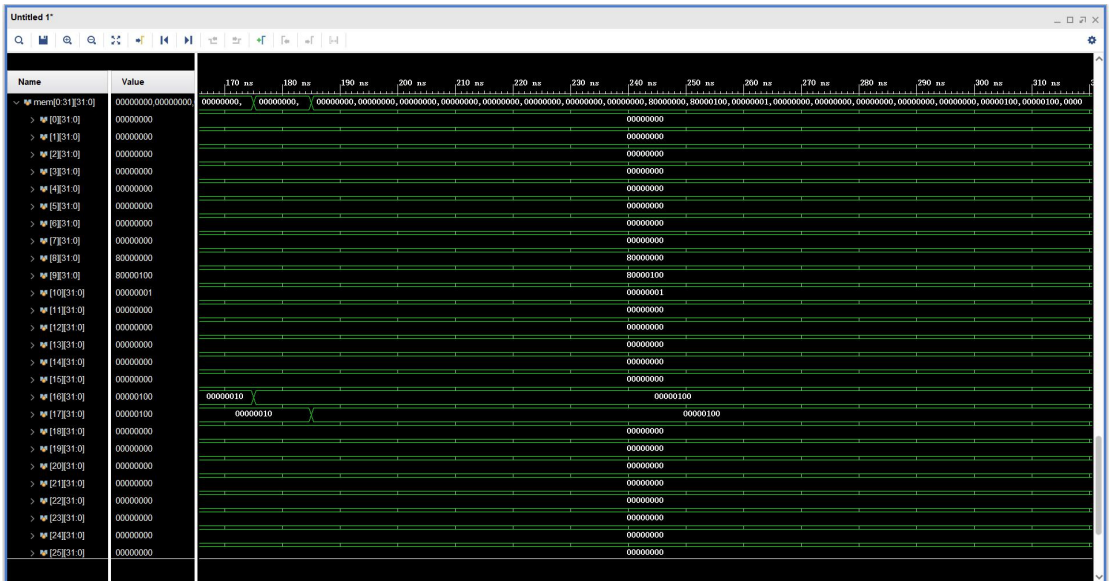

图 6

最后，检查寄存器堆中的值，与实验预期相符。

第二个仿真用例主要测试相关处理，具体仿真结果如下：

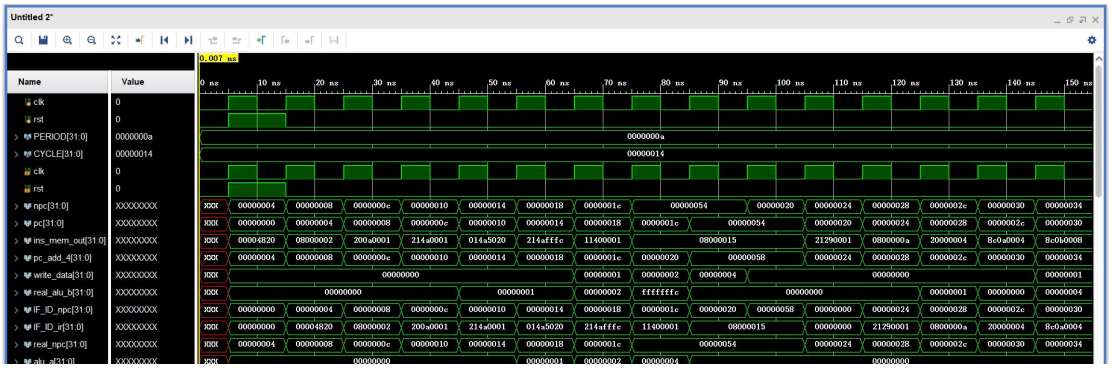

图 7

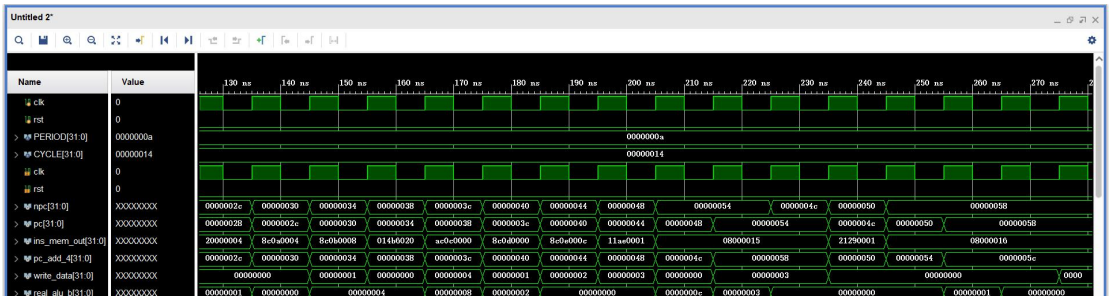

图 8

根据指令跳转结果可知，指令执行顺序与实验预期相同。

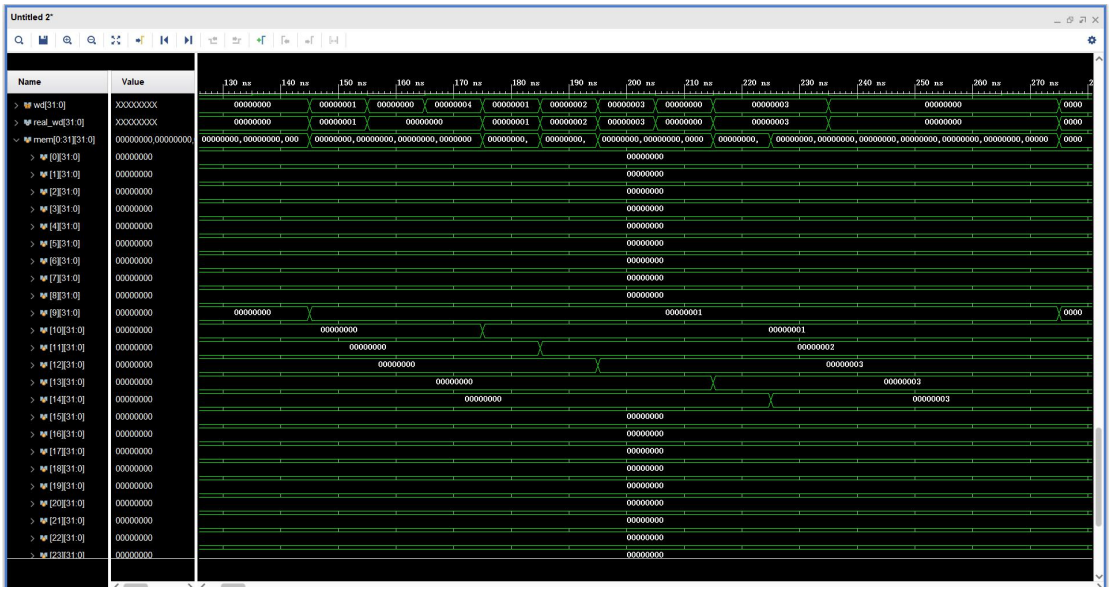

图 9

寄存器堆中的值也表明，数据相关处理正常执行。

因此，仿真结果表明所设计的流水线 CPU 满足设计要求

结果分析

仿真结果表明设计，运行效果与设计目标相同。

调试单元(DBU)

设计目标

为了方便下载调试，设计一个调试单元 DBU，该单元可以用于控制 CPU 的运行方式，显示运行过程的中间状态和最终运行结果。DBU 的端口与 CPU 以及 FPGA 开发板外设（拨动/按钮开关、LED 指示灯、7-段数码管）的连接如图所示。为了 DBU 在不影响 CPU 运行的情况下，随时监视 CPU 运行过程中寄存器堆和数据存储器的内容，可以为寄存器堆和数据存储器增加 1 个用于调试的读端口。

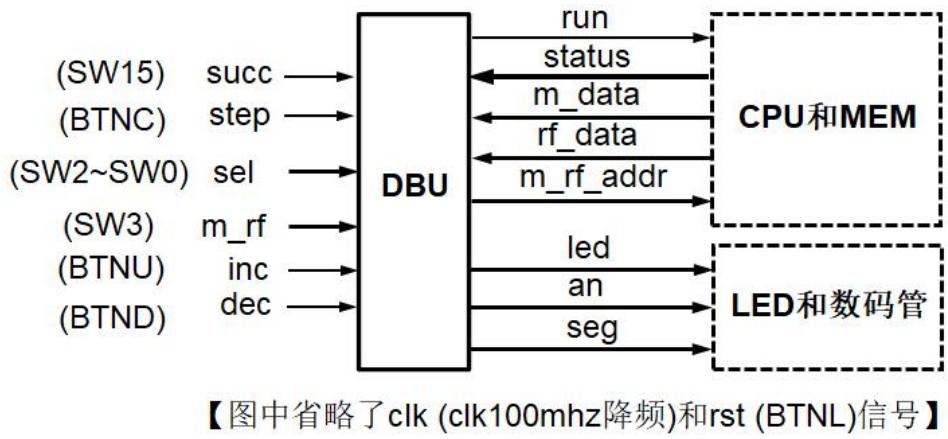

图 10

端口声明

端口声明如下：

```
module DBU(
        input clk,//clock
        input succ,//successive
        input step,//step
        input rst,//reset
        input [ 2: 0 ] sel,//select
        input m_rf,//memory_regfile
        input inc,//increase
        input dec,//decrease
        output reg [ 15: 0 ] led,//show part
        output reg [ 7: 0 ] an,
        output [ 7: 0 ] seg
      );
```

设计实现

实现思路为将需要显示的数据从 CPU 中引出，并通过数据选择器，选择以供显示。

在 CPU 中的 regfile 及 memory 的读地址前，各增加一数据选择器，以选择是 CPU 内部运行时使用的地址，还是 DBU 读取的地址。而 m_rf_addr 的修改，则由两段式有限状态机实现，根据取信号边沿后的 inc 及 dec 信号增减。

此外还需增加显示模块，根据须显示的数字，输出数码管的使能信号，及滚动显示，以保证数据的正常显示。

核心代码

m_rf_aadr 的修改及控制是否连续运行部分的核心代码如下(剩余部分为简单的连线)：

```
always @( posedge clk )
  begin
    if ( edg_inc )
      begin
```

```
        n_m_rf_addr = m_rf_addr + 8'd1;
      end
    else if ( edg_dec )
      begin
        n_m_rf_addr = m_rf_addr - 8'd1;
      end
    else
      begin
        n_m_rf_addr = m_rf_addr;
      end
  end
always @( posedge clk )
  begin
    if ( rst )
      begin
        m_rf_addr <= 8'd0;
      end
    else
      begin
        m_rf_addr <= n_m_rf_addr;
      end
  end
always @ *
  begin
    if ( succ )
      begin
        clk_cpu = clk;
      end
    else
      begin
        clk_cpu = edg_step;
      end
  end
```

结果分析

CPU 的仿真结果均符合设计目标。

实验总结

本次实验复习了上学期所学的 Verilog 的基本知识，并通过流水线 CPU 及 DBU 的设计，为接下来的综合设计打下基础。

意见建议

无。

思考题

增加一个有限状态机，用两个比特位表示 4 个状态，根据每次分支指令的跳转与否，进行状态转移，实现分支预测。