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：lab3_单周期 CPU

学生姓名：胡毅翔

学生学号：PB18000290

完成日期：2020 年 5 月 13 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

单周期 CPU

设计目标

设计实现单周期 CPU 并可执行 add，addi，lw，sw，beq，j 共 6 条指令。

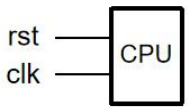

图 1

指令功能与格式如下所示：

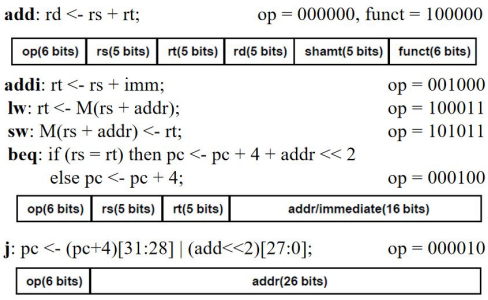

图 2

端口声明

端口声明如下：

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

设计实现

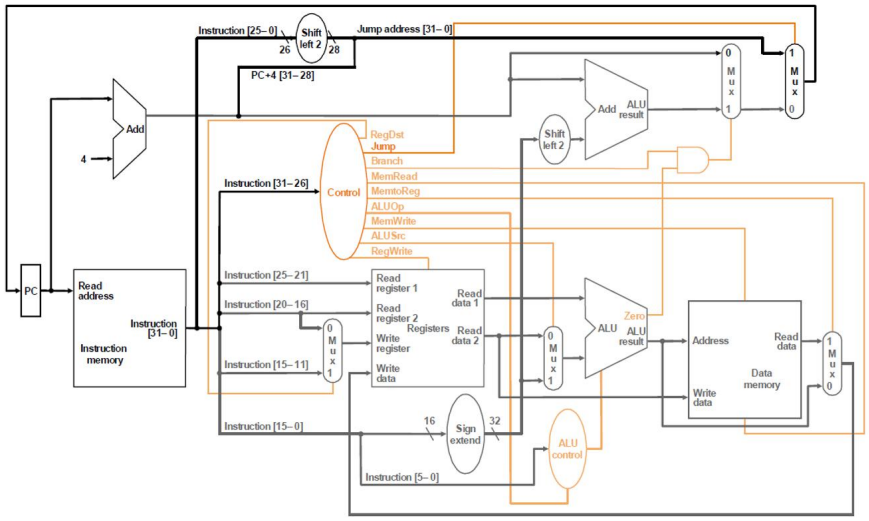

图 3

数据通路如上图所示，只需根据指令输出控制信号。

核心代码

核心代码(根据指令输出控制信号)如下：

```
//Control Unit
always @*
    begin
        case(instruct[31:26])
        6'b000000:begin//add
                    RegDst=1;
                    jump=0;
                    branch=0;
                    MemtoReg=0;
                    ALUop=2'd00;
                    Memwe=0;
                    ALUSrc=0;
                    Regwe=1;
                   end
        6'b001000:begin//addi
                    RegDst=0;
```

```
                jump=0;
                branch=0;
                MemtoReg=0;
                ALUop=2'd00;
                Memwe=0;
                ALUSrc=1;
                Regwe=1;
               end
      6'b100011:begin//lw
                RegDst=0;
                jump=0;
                branch=0;
                MemtoReg=1;
                ALUop=2'd00;
                Memwe=0;
                ALUSrc=1;
                Regwe=1;
               end
      6'b101011:begin//sw
                RegDst=1;
                jump=0;
                branch=0;
                MemtoReg=0;
                ALUop=2'd00;
                Memwe=1;
                ALUSrc=1;
                Regwe=0;
               end
      6'b000100:begin//beq
                RegDst=0;
                jump=0;
                branch=1;
                MemtoReg=0;
                ALUop=2'd01;
                Memwe=0;
                ALUSrc=0;
                Regwe=0;
               end
      6'b000010:begin//j
                RegDst=0;
                jump=1;
                branch=0;
                MemtoReg=0;
                ALUop=2'd00;
```

```
                    Memwe=0;
                    ALUSrc=0;
                    Regwe=0;
                   end
        endcase
    end
//ALU Control Unit
always @*
    begin
        case(ALUop)
            2'b00:alu_op=3'b000;
            2'b01:alu_op=3'b001;
        endcase
    end
```

仿真结果

仿真结果如下，最后进入“sw $s2,8($0)”及“j _success”的循环：

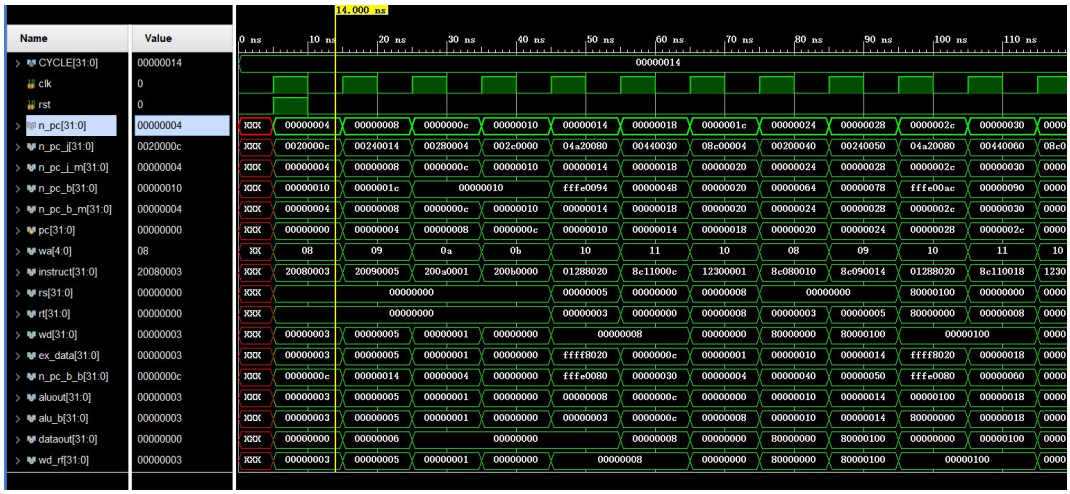

图 4

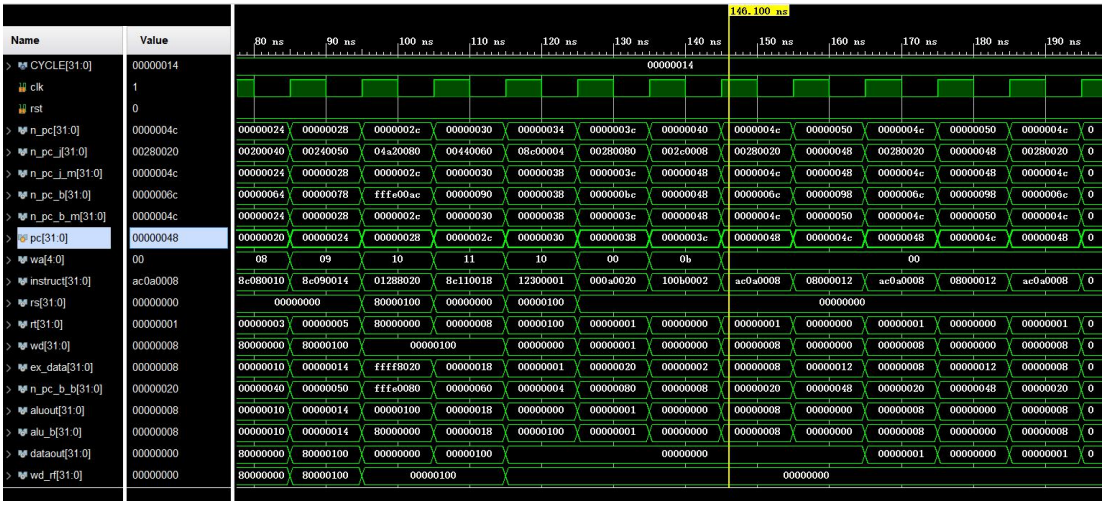

图 5

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。

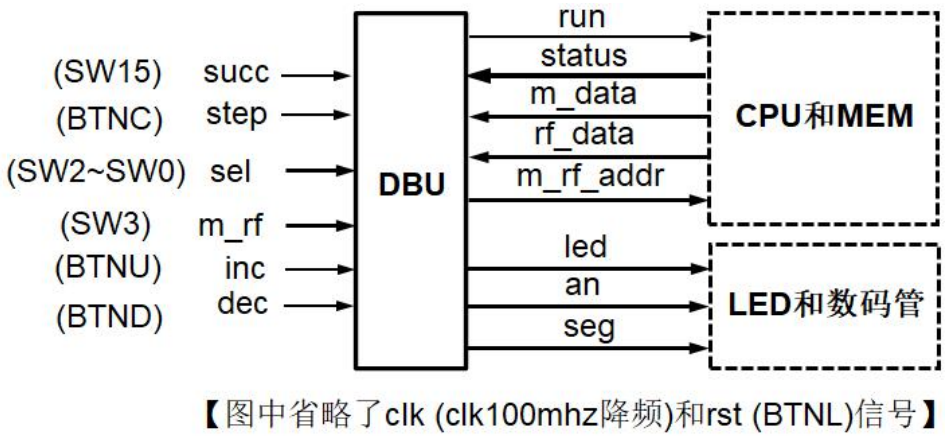

图 6

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 及数据 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：

```
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
```

仿真结果

仿真结果如下：

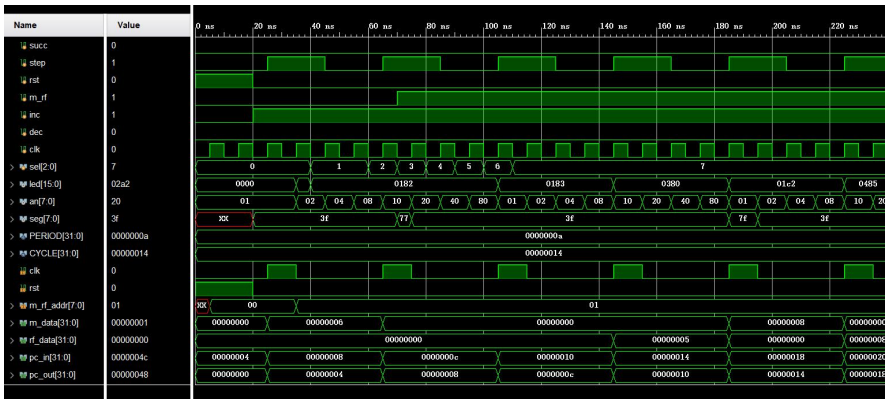

图 7

思考题(含 accm 指令的单周期 CPU)

设计目标

在原有的单周期 CPU 中增加 accm 指令。

指令格式及功能如下：

accm: rd <- M(rs) + rt;　　op = 000000, funct = 101000

op(6 bits)	rs(5 bits)	rt(5 bits)	rd(5 bits)	shamt(5 bits)	funct(6 bits)

图 8

端口声明

端口声明与原单周期 CPU 相同。

设计实现

实现思路为在原 CPU 中增加如下两条线(绿线及红线)，及两个二选一选择器，并在控制单元增加对这两个选择器的控制。

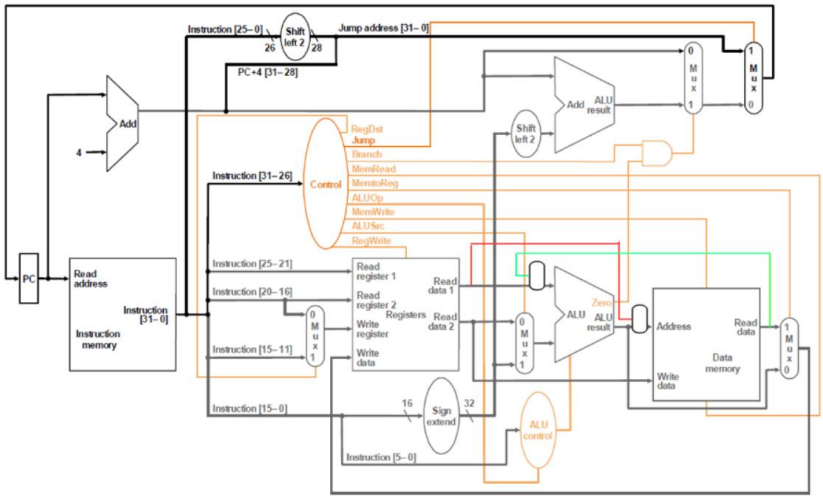

图 9

核心代码

控制单元核心代码(只含新增的控制信号，其余同上，略去)，如下：

```
//Control Unit
always @*
    begin
        case(instruct[31:26])
        6'b000000:begin
                    if(instruct[5:0]==6'b100000)begin//add
                    mem_r_d_sel=0;
                    alu_a_sel=0;
                    end
                    else if(instruct[5:0]==6'b101000)begin//accm
                    mem_r_d_sel=1;
                    alu_a_sel=1;
                    end
                   end
        6'b001000:begin//addi
                    mem_r_d_sel=0;
                    alu_a_sel=0;
                   end
        6'b100011:begin//lw
                    mem_r_d_sel=0;
                    alu_a_sel=0;
                   end
        6'b101011:begin//sw
                    mem_r_d_sel=0;
                    alu_a_sel=0;
                   end
```

```
        6'b000100:begin//beq
                    mem_r_d_sel=0;
                    alu_a_sel=0;
                   end
        6'b000010:begin//j
                    mem_r_d_sel=0;
                    alu_a_sel=0;
                   end

        endcase
    end
```

仿真结果

仿真使用的指令(伪代码)，图中为修改部分，其余部分同原 CPU 所用指令：

```
_success:
        sw   $t2,8($0)     #全部测试通过，存储器地址0x08里值为1
        accm: $t0<-M(0x01)+$t2
        j    _success
```

图 10

仿真结果如下：

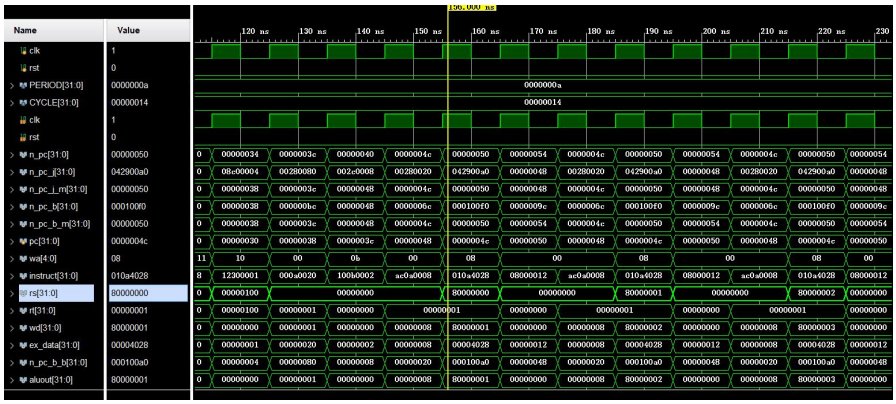

图 11

可见，进入循环后，$t0 寄存器每次加 1。

结果分析

CPU 及 DBU 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，并通过单周期 CPU 及 DBU 的设计，为后半学期的多周期，流水线 CPU 设计打下基础。

意见建议

无。

思考题

已在实验报告中体现。