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：lab4_多周期 CPU

学生姓名：胡毅翔

学生学号：PB18000290

完成日期：2020 年 5 月 26 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

多周期 CPU

设计目标

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

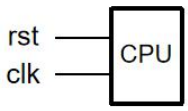

图 1

指令功能与格式如下所示：

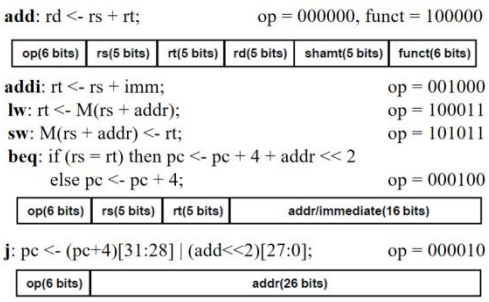

图 2

端口声明

端口声明如下：

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

设计实现

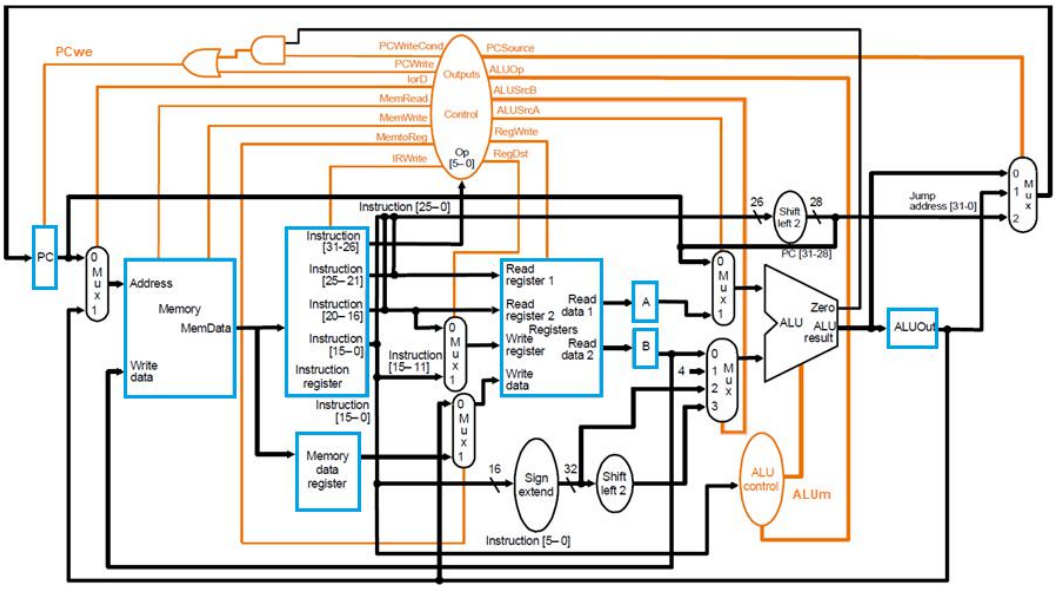

图 3

数据通路如上图所示。

而控制信号的输出，则根据状态进行控制，状态图如下：

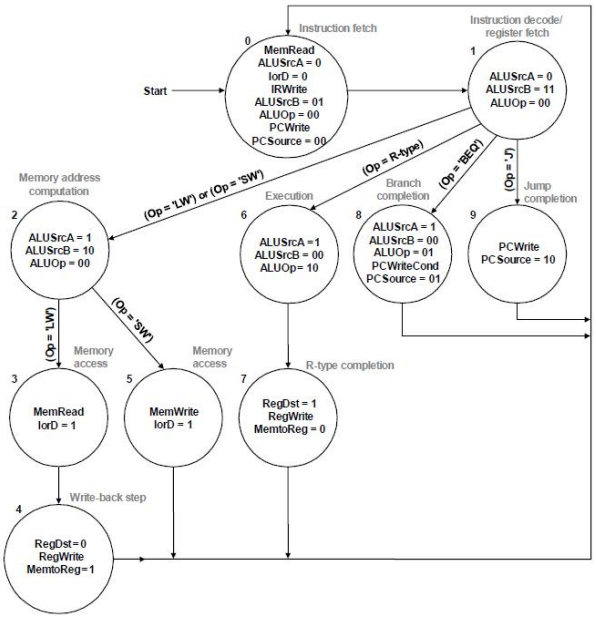

图 4

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：

```
//control logic
always @( * )
  begin
    { PCWriteCond, PCWrite, IorD, MemRead, MemWrite, MemtoReg, IRWrite, PCSource, ALUOp, ALUSrcB, ALUSrcA, RegWrite, RegDst, r_ao, r_rf0, r_rf1, r_data } = 20'd0;
    if ( rst )
      begin
        { PCWriteCond, PCWrite, IorD, MemRead, MemWrite, MemtoReg, IRWrite, PCSource, ALUOp, ALUSrcB, ALUSrcA, RegWrite, RegDst, r_ao, r_rf0, r_rf1, r_data } = 20'd0;
      end
    else
      begin
        case ( status )
          4'd0:
            begin
              MemRead = 1'b1;
              IRWrite = 1'b1;
              ALUSrcB = 2'b1;
              PCWrite = 1'b1;
            end
          4'd1:
            begin
              r_ao = 1'b1;
              r_rf0 = 1'b1;
              r_rf1 = 1'b1;
              ALUSrcB = 2'b11;
            end
          4'd2:
            begin
              r_ao = 1'b1;
              ALUSrcA = 1'b1;
              ALUSrcB = 2'b10;
            end
          4'd3:
            begin
              r_data = 1'b1;
              MemRead = 1'b1;
              IorD = 1'b1;
            end
          4'd4:
            begin
```

```
            RegWrite = 1'b1;
            MemtoReg = 1'b1;
          end
        4'd5:
          begin
            MemWrite = 1'b1;
            IorD = 1'b1;
          end
        4'd6:
          begin
            if ( instr[ 29 ] )
              begin
                r_ao = 1'b1;
                ALUSrcA = 1'b1;
                ALUSrcB = 2'b10;
                ALUOp = 2'b10;
              end
            else
              begin
                r_ao = 1'b1;
                ALUSrcA = 1'b1;
                ALUOp = instr[1]?2'b01:2'b10;
              end
          end
        4'd7:
          begin
            if ( instr[ 29 ] )
              begin
                RegDst = 1'b0;
                RegWrite = 1'b1;
              end
            else
              begin
                RegDst = 1'b1;
                RegWrite = 1'b1;
              end
          end
        4'd8:
          begin
            r_ao = 1'b0;
            ALUSrcA = 1'b1;
            ALUOp = 2'b01;
            PCWriteCond = 1'b1;
            PCSource = 2'b01;
```

```
            end
          4'd9:
            begin
              PCWrite = 1'b1;
              PCSource = 2'b10;
            end
        endcase
      end
  end
```

仿真结果

仿真结果如下，开始时执行 4 条 4 个周期的 addi 指令，1 条 4 周期的 add 指令，及 1 条 5 周期的 lw 指令，状态转移及当前指令，可由图 5 中 status 及 instr 看出：

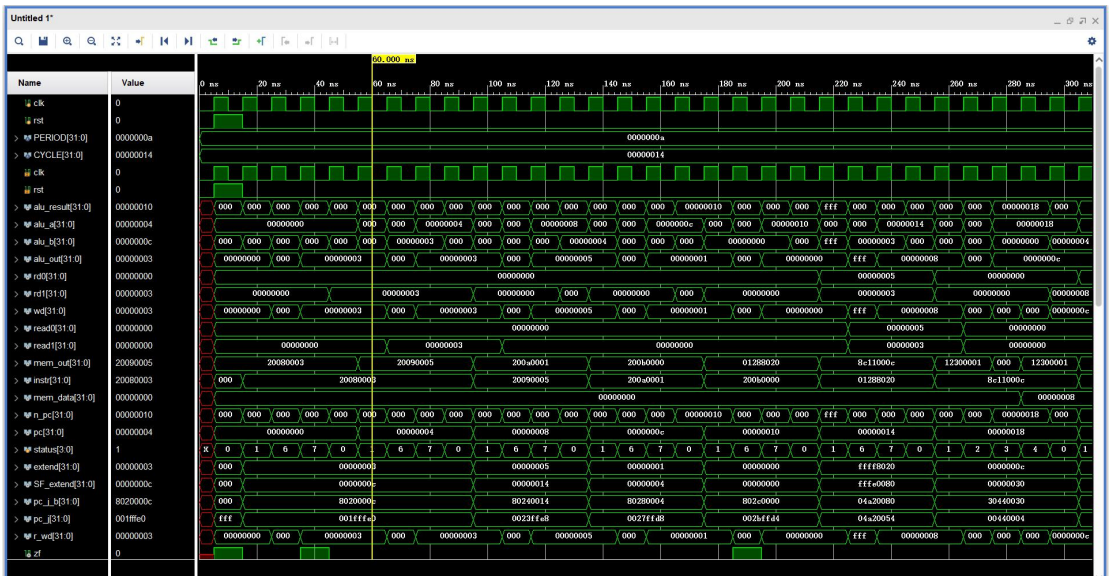

图 5

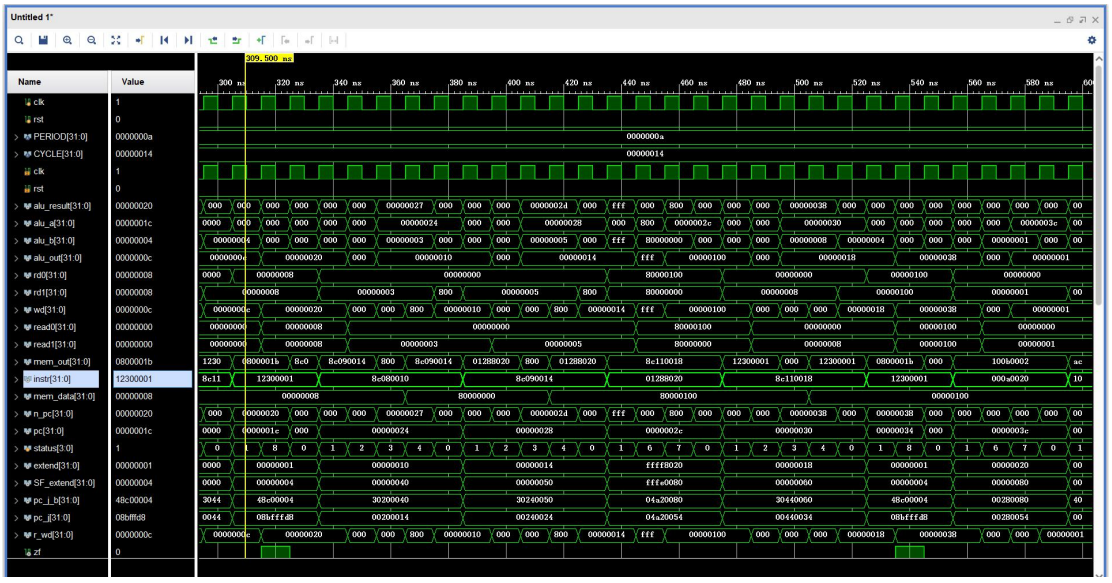

图 6

之后，执行 3 周期的 beq 指令，可见 pc 正确跳转。

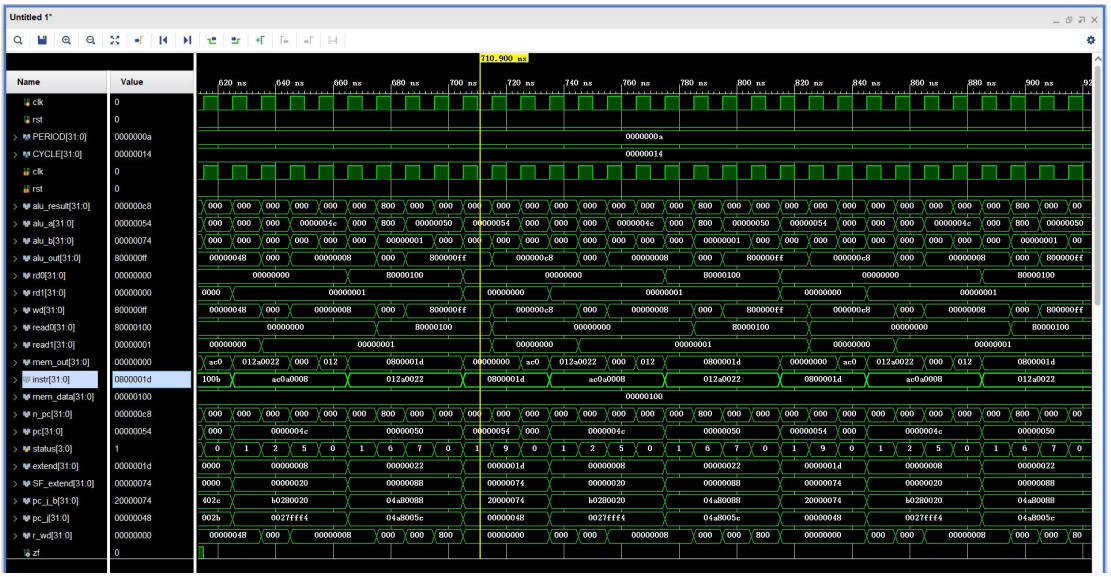

图 7

最后，指令进入_success 循环，在此处在 sw 和 j 指令之间，增加了 1 条 sub $0,$t1,$t2 指令，用于测试 sub 指令。根据 alu 计算结果，及输入寄存器堆的内容，可见，sub 指令正确执行。

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。

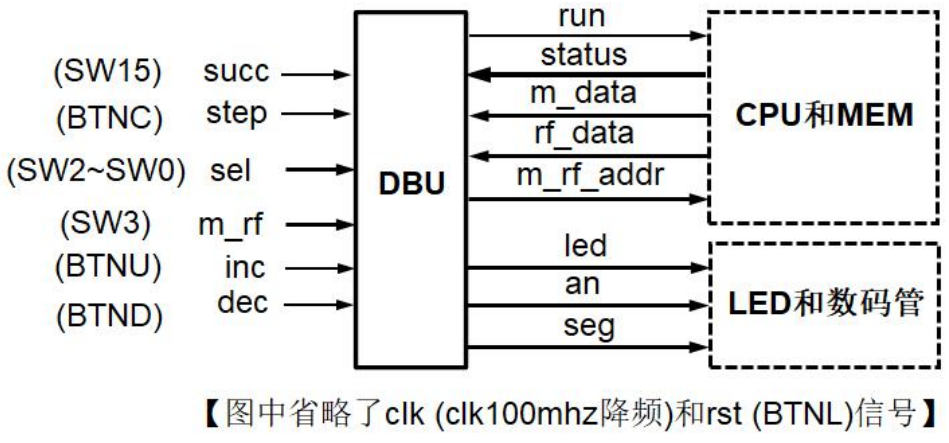

图 8

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(链接完整代码)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，并通过多周期 CPU 及 DBU 的设计，为接下来的流水线 CPU 设计打下基础。

意见建议

无。

思考题

增加了 sub 指令，已在实验报告中体现。