中国科学院大学计算机组成原理实验课

实 验 报 告

学号：2015K8009929014　姓名：李云志　专业：计算机科学与技术

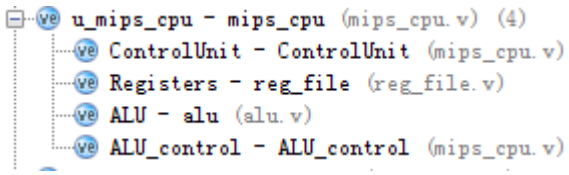
实验序号：3　实验名称：多周期 CPU

注 1：本实验报告请以 PDF 格式提交。文件命名规则：[学号]-PRJ[实验序号]-RPT.pdf，
其中文件名字母大写，后缀名小写。例如：[2014K8009959088]-PRJ[1]-RPT.pdf
注 2：实验报告模板以下部分的内容供参考，可包含但不限定如下条目内容。

一、 逻辑电路结构与仿真波形的截图及说明（比如关键 RTL 代码段{包含注释}及其对应的逻辑电路结构、相应信号的仿真波形和信号变化的说明等）

1、 多周期 CPU 设计

CPU 模块构成：根据多周期 CPU 的数据通路以及控制通路等概念，将 CPU 规划为 5 部分，分别为 mips_cpu, ControlUnit, Registers, ALU, ALU_control（如图）



2、 核心代码段及逻辑电路图：

```
PC：
always @(posedge clk)
    begin
        if(rst)
          begin
            PC_reg <= 32'd0;
            Instruction_reg=32'b0;
            Memory_data = 32'b0;
            A = 32'b0;
            B = 32'b0;
            ALUOut = 32'b0;
        end
```

```
    else if(PC_control)
      PC_reg <= PC_next;
      if(IRWrite)
            Instruction_reg = Instruction;
            Memory_data = Read_data;
            A = reg_out_rdata1;
            B = reg_out_rdata2;
            ALUOut = alu_out_result;
      end
```

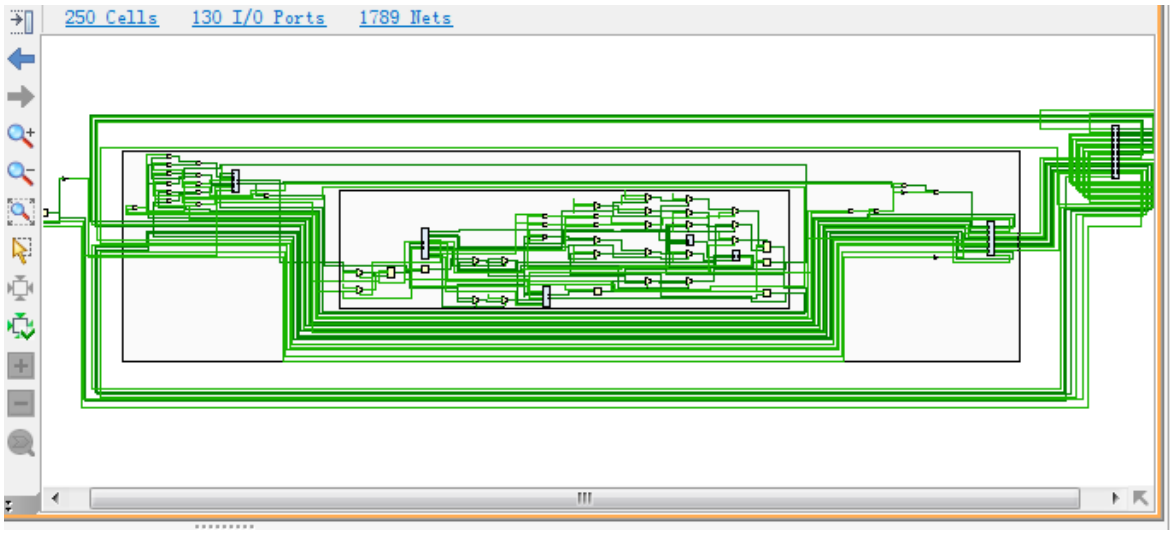
ALU_control：

```
module ALU_control(
input [5:0] funct,
input [5:0] opcode,
input [1:0] ALUOp,
output reg [2:0] ALUop
);
always@(*)
begin
   case(ALUOp)
   2'b00:
   ALUop=3'b010;
   2'b10:
   begin
      case(funct)
      6'b100001:ALUop=3'b010;//addu
      6'b100000:ALUop=3'b010;//add
      6'b100011:ALUop=3'b110;//subu
      6'b100010:ALUop=3'b110;//sub
      6'b001000:ALUop=3'b010;//JR
      6'b000000:ALUop=3'b101;//sll
      6'b100101:ALUop=3'b010;//or
      6'b100100:ALUop=3'b000;//and
      6'b101010:ALUop=3'b111;//slt
      default:ALUop=3'b010;
      endcase
      end
   2'b01:
   case(opcode)
   6'b000100:ALUop=3'b110;//beq,bne's sub
   6'b000101:ALUop=3'b110;//beq,bne
   6'b001111:ALUop=3'b011;//lui
   6'b001010:ALUop=3'b111;//slti
   6'b001011:ALUop=3'b100;//sltiu

   default:ALUop=3'b010;
   endcase
   default:ALUop=3'b000;
   endcase
end
endmodule
```

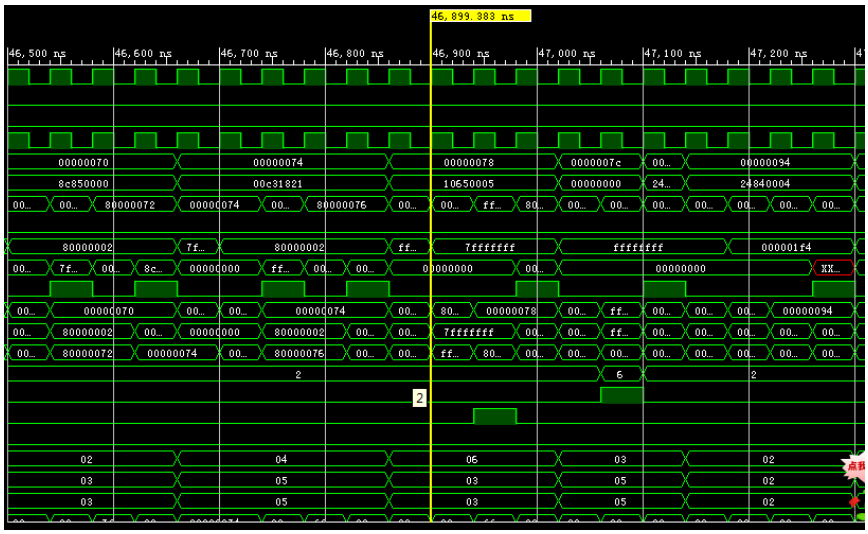
ControlUnit （状态机部分）：

```
always@(current_state or opcode)
    begin
      case(current_state)
      S0:next_state=S1;
      S1:
      begin
        case(opcode[5:0])
        6'b000000:begin
        if(funct==6'b000000)
        next_state=S15;
        else
        next_state=S6;
        end//r_type
        6'b000101:next_state=S8;//bne
        6'b000100:next_state=S8;//beq
        6'b100011:next_state=S2;//lw
        6'b101011:next_state=S2;//sw
        6'b000010:next_state=S9;//j
        6'b000011:next_state=S12;
        6'b001111:next_state=S13;//lui
       6'b001010:next_state=S13;//slti
       6'b001001:next_state=S10;//addiu
       6'b001011:next_state=S13;//sltiu
        default:next_state=S0;
        endcase
      end
      S2:
      begin
        case(opcode[3])
        1'b0:next_state=S3;
        1'b1:next_state=S5;
        default:next_state=S0;
        endcase
        end
      S3:next_state=S4;
      S4:next_state=S0;
      S5:next_state=S0;
      S6:begin
      case(funct)
      6'b001000:next_state=S14;
      default:next_state=S7;
      endcase
      end
      S7:next_state=S0;
      S8:next_state=S0;
      S9:next_state=S0;
      S10:next_state=S11;
      S11:next_state=S0;
      S12:next_state=S0;
      S13:next_state=S11;
      S14:next_state=S0;
      S15:next_state=S7;
      default:next_state=S0;
      endcase
    end
```



3、 SUM 行为仿真波形及结果

波形:



4、 后期调试

完成代码的编写后，进行单条指令测试，编写 Ideal_Mem

```
mem[0] = {`ADDIU_opcode, 5'b0, 5'b1, 16'b1};
mem[1] = {`ADDIU_opcode, 5'b0, 5'd2, 16'd2};
mem[2] = {`ADDIU_opcode, 5'b0, 5'd3, 16'hFFFF};//reg[3] = -1
mem[3] = {`SLL_func_code, 5'b0, 5'b1, 5'd4, 5'b1, `SLL_func_code};//reg[4]=1<<1;
mem[4] = {`BEQ_opcode, 5'd2, 5'd4, 16'd3 };//jump to mem[8]
mem[5] = {`SLTI_opcode, 5'd3, 5'd5, 16'd2}; //if(-1< 2) reg[5]=1
mem[6] = {`SLTIU_opcode, 5'd3, 5'd6, 16'd2};// if(0xffff<0x0002) reg[6]=1
mem[7] = {`SPECIAL_opcode, 5'b11111, 10'd0, 5'd0, `JR_func_code};// jump to (reg[31])
mem[8] = {`JAL_opcode, 26'd5}; //reg[31]=40; jump to mem[5]
mem[9] = 32'b0;
mem[10] = {`LUI_opcode, 5'd0, 5'd3, 16'hABCD};//reg[3] =0xabcd0000
mem[11] = {`SPECIAL_opcode, 5'd1, 5'd2, 5'd7,5'd0, `OR_func_code};//reg[7] = reg[1]|reg[2]
mem[12] = {`J_opcode,26'd15 };
mem[15] = {`SPECIAL_opcode, 5'd1, 5'd2,5'd8, 5'd0, `SLT_func_code};
```

5、 思考与总结

（1） 相较于单周期 CPU，多周期 CPU 在数据通路以及控制逻辑的设计复杂程度上都有了较大程度的提升，具体体现在状态机的设计，选择器的设计等等。而通过对多周期 CPU 的设计，不仅对多周期 CPU 结构有了更深入的认识，还提高了代码的编写以及测试能力。

（2） 关于代码测试：这次的实验相比于单周期 CPU 新加入了较多指令，所以在后期调试中，就更加需要精细认真。我选择了先测试单条指令，后综合的方法，在较快时间内达到了调试目的。

（3） 在控制逻辑的设计上，发现许多有限状态机的状态可以复用，从而简化代码与控制电路等，有利于提高处理器性能，同时便于调试。

（4） 更加了解 mips 指令集，深刻体会到 RISC 指令集的特点以及优势