中国科学院大学计算机组成原理实验课

实 验 报 告

学号：2016K8009915009　姓名：钟赟　专业：计算机科学与技术

实验序号：__ 实验名称：内存及外设通路设计

一、 逻辑电路结构与仿真波形的截图及说明

1. 关键代码段

(1) 下面为 mips_cpu 多周期部分修改的代码

```
//different states
parameter IF = 3'd0,
                    IW = 3'd1,
                    ID_EX = 3'd2,
                    LD = 3'd3,
                    RDW = 3'd4,
                    WB = 3'd5,
                    ST = 3'd6;

//skip of state
always@(posedge clk)    begin
    if(rst)
        state <= IF;
    else
        state <= next_state;
end

//get next state depending on current state
always @(*) begin
    case(state)
        IF: next_state = (Inst_Req_Ack & Inst_Req_Valid) ? IW : IF;
        IW: next_state = (Inst_Valid & Inst_Ack) ? ID_EX : IW;
        ID_EX: begin
            if(store)
                next_state = ST;
            else if(load)
                next_state = LD;
            else if(PCsrc | jump | jal | jalr | jr)
                next_state = IF;
            else
                next_state = WB;
        end
        LD: next_state = (Mem_Req_Ack & MemRead) ? RDW : LD;
        RDW: next_state = (Read_data_Valid & Read_data_Ack) ? WB : RDW;
        WB: next_state = IF;
        ST: next_state = (Mem_Req_Ack & MemWrite) ? IF : ST;
    endcase
end
```

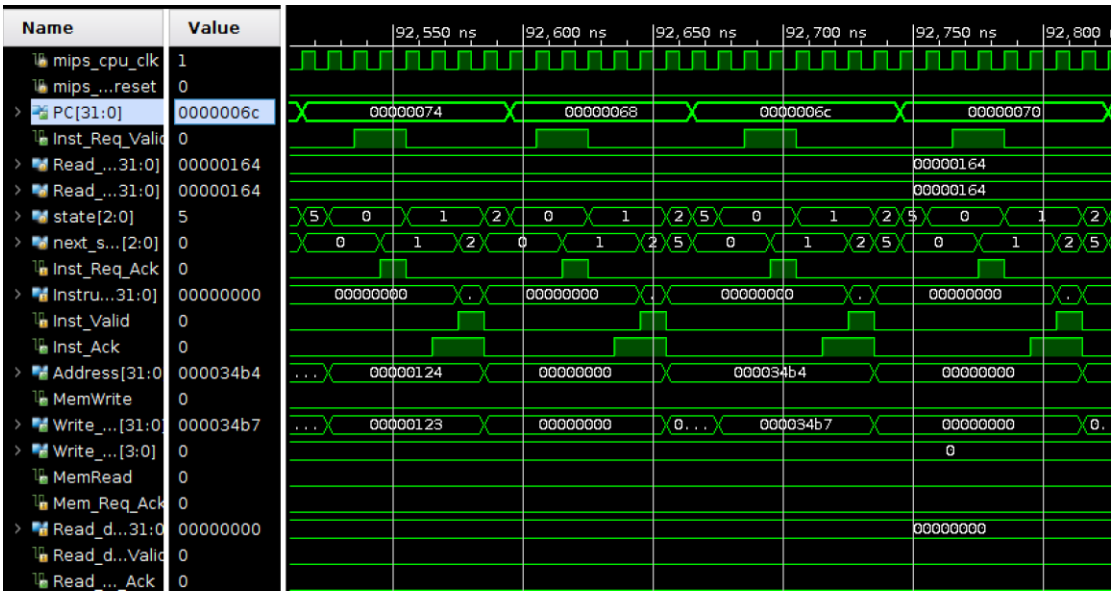
```
    //depending on current state, output different registers' signals
    always @(posedge clk) begin
        if(rst) begin
            PC <= 32'd0;
            Inst_Req_Valid <= 0;
            Inst_Ack <= 1;
            MemRead <= 0;
            MemWrite <= 0;
            Read_data_Ack <= 0;
        end
        else begin
        case(state)
            IF: Inst_Req_Valid <= Inst_Req_Ack ? 1'b0 : 1'b1;
            IW: Inst_Ack <= Inst_Valid ? 1'b0 : 1'b1;
            ID_EX: begin
                if(jr | jalr)
                    PC[31:2] <= rdata1[31:2];
                else if(jump | jal)
                    PC[31:2] <= {PC[31:28], Instruction1
[25:0]};
                else
                    PC <= (PCsrc) ? ( PC + 4 + immediate_32*4) : (PC + 4);
            end
            ST: MemWrite <= Mem_Req_Ack ? 1'b0 : 1'b1;
            LD: MemRead <= Mem_Req_Ack ? 1'b0 : 1'b1;
            RDW: Read_data_Ack <= Read_data_Valid ? 1'b0 : 1'b1;
        endcase
        end
    end

    //store current instruction
    always @(posedge clk) begin
        if(Inst_Valid && Inst_Ack)
            Instruction1 <= Instruction;
        else
            Instruction1 <= Instruction1;
    end
```

control_unit 部分添加了 bgtz 功能，alu, reg_file 代码与 prj2 相同，故不再赘述。

(2) printf.c 中 puts 函数代码：

```
puts(const char *s)
{
        int i = 0;
        while(s[i] != 0)
        {
                while((*(uart + UART_STATUS/4) & UART_TX_FIFO_FULL) != 0);
                *(uart + UART_TX_FIFO/4) = s[i];
                i++;
        }
        return i;
}
```

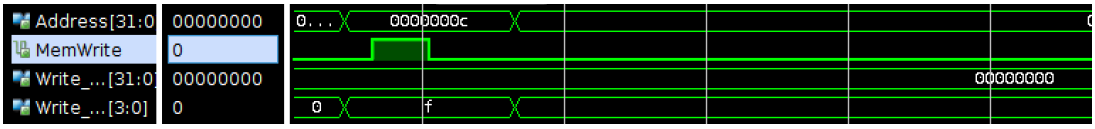
2. 仿真波形及上板



从上图可以看出 PC=0x74 时 ,state 从 0(IF)跳转到 1(ID_EX) ,并在 state = 1 时，

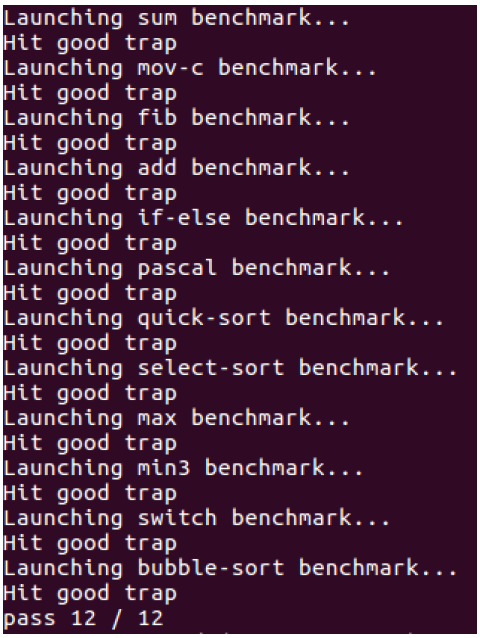
Inst_Valid 和 Inst_Ack 同时拉高，读取指令 Instruction，进行操作。

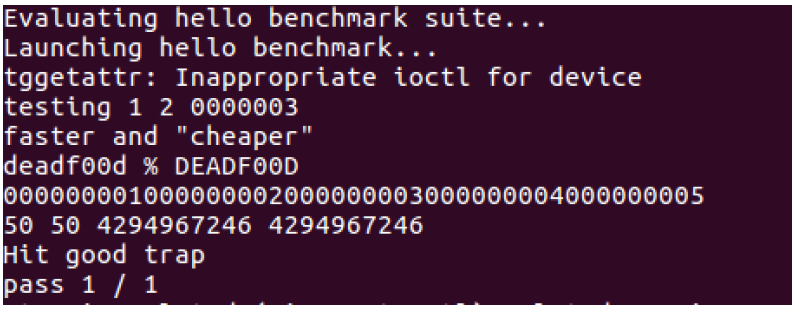
下图为行为仿真正确结束的标志：



下面为 benchmark 和 hello 上板测试的结果：

```
Launching memcpy benchmark...
Hit good trap
pass 1 / 1
```





二、 实验过程中遇到的问题、对问题的思考过程及解决方法

1. 刚开始时部分 benchmark 的仿真波形是正确的，但是上板没有通过，原因在于 Read_data 和 Instruction 信号在用寄存器存储时赋值的 always 语句应该使用 clk 触发，以及在复位信号有效时将所有应答和接收信号都赋初值。
2. 后来只有 advanced:04 上板不通过，debug 的结果是：在处理器向下一个状态跳转的组合逻辑里，ID_EX 状态遇到跳转指令时下一状态为 IF，此处的跳转指令是指指令为跳转指令且跳转条件满足，当条件不满足时状态不跳转。
3. 在写 puts 函数时，打印出现错误，是由于没有将地址偏移量除以 4。每个寄存器是 32bit，因此相邻寄存器的地址偏移值为 4。

三、 对讲义中思考题（如有）的理解和回答

思考题：UART 控制器基地址指针初始化中 volatile 关键字的作用是什么？如果去掉会出现什么后果？

答：volatile 表明 uart 指针是随时可能发生变化的，故每次使用时必须从地址中读取。加了 volatile 关键字的变量有关的运算，不进行编译优化，可以保证对地址的稳定访问。而且在打印程序中 uart 的访问非常频繁，如果不加 volatile 关键字，很容易导致取值不稳定。

另外，去掉 volatile 关键字打印是否出错似乎与虚拟机的环境有关，在我的虚拟机上没有出错。

四、 对于此次实验的心得、感受和建议

我认为这次实验的难度和它所给的时限不是很相称，存在时间很紧迫的情况。总体需要完成的任务量不大，但是 debug 过程比较复杂，只能与 project2 相应的 benchmark 波形逐一对比，不太易发现问题。

在此感谢张林隽同学在 debug 过程中进行的共同讨论和帮助。