中国科学院大学计算机组成原理实验课

实 验 报 告

学号：_2018K8009929030_　姓名：_热伊莱·图尔贡_　专业：_计算机科学与技术_

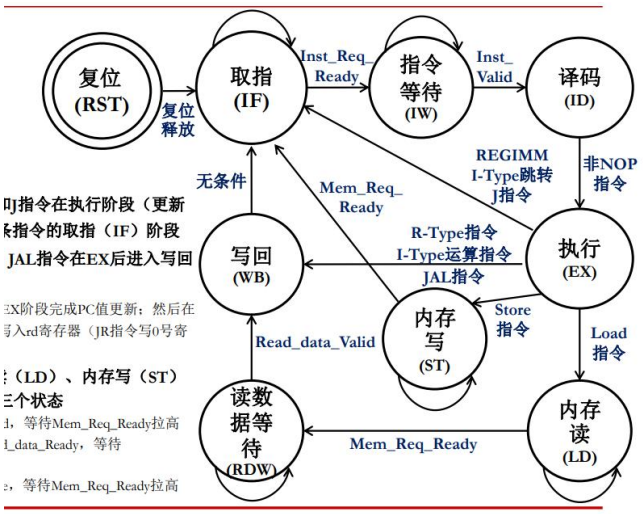
实验序号：_3_　实验名称：_内存及外设通路设计与处理器性能评估_

注 1：撰写此 Word 格式实验报告后以 PDF 格式保存在本地仓库的主目录下。文件命名规则：学号-prjN.pdf，其中学号中的字母“K”为大写，“-”为英文连字符，“prj”和后缀名“pdf”为小写，“N”为 1 至 4 的阿拉伯数字。例如：2019K8009929000-prj1.pdf。PDF 文件大小应控制在 5MB 以内。此外，实验项目 5 包含多个选做内容，每个选做实验应提交各自的实验报告文件，文件命名规则：学号-prj5-projectname.pdf，例如：2019K8009929000-prj5-dma.pdf。具体要求详见实验项目 5 讲义。 注 2：使用 git add 及 git commit 命令将实验报告 PDF 文件添加到本地仓库 master 分支，并通过 git push 推送到 GitLab 远程仓库 master 分支（具体命令详见实验报告）。 注 3：实验报告模板下列条目仅供参考，可包含但不限定如下内容。实验报告中无需重复描述讲义中的实验流程。

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. 状态机描述：

使用“三段式”状态机描述方法：



“第一段”描述状态寄存器的同步状态跳转：

```
//describe the current_state
always @ (posedge clk) begin
        if (rst) begin
            current_state <= RST;
        end
        else begin
            current_state <= next_state;
        end
end
```

“第二段”根据状态机当前状态和输入信号，描述下一状态的计算逻辑：

```
//describe the next_state
always @ (*) begin
       case (current_state)
           RST:begin
              next_state = IF;
           end
           IF:begin
             if(Inst_Req_Ready)
                next_state = IW;
             else
                next_state = IF;
           end
           IW:begin
             if(Inst_Valid)
                next_state = ID;
             else
                next_state = IW;
           end
           ID:begin
             if (NOP)
                next_state = IF;
             else
                next_state = EX;
           end
```

```
EX:begin
  if(Regimm || I_B || (Jump && !Jal))
     next_state = IF;
  else if(Store)
     next_state = ST;
  else if(Load)
     next_state = LD;
  else
     next_state = WB;
end
LD:begin
  if(Mem_Req_Ready)
     next_state = RDW;
  else
     next_state = LD;
end
ST:begin
  if(Mem_Req_Ready)
     next_state = IF;
  else
     next_state = ST;
end
    RDW:begin
      if(Read_data_Valid)
         next_state = WB;
      else
         next_state = RDW;
    end
    WB:begin
       next_state = IF;
    end
    default:
       next_state = current_state;

endcase
```

“第三段”根据状态机当前状态，描述不同输出寄存器的同步变化：

```
//describe synchronous changes in different registers
assign Inst_Ready     =((current_state == IW) || (current_state == RST))?1:0;
assign Inst_Req_Valid =(current_state == IF)?1:0;
assign Read_data_Ready=((current_state == RDW) || (current_state == RST))?1:0;
assign MemRead        =(current_state == LD)?1:0;
assign MemWrite       =(current_state == ST)?1:0;
```

2. PC，Instruction 和 Read_data 的更新

Read_data/Instruction/PC 三个值与单周期时不同，只有特定周期才更新，因此我分别设置了三个 reg 类型的变量用于这三个信号的更新：

```
//INSTRUCTION
always @(posedge clk) begin
       if(rst)
          Instruction_r <=32'b0;
       else if (current_state == IW && (Inst_Valid && Inst_Ready))
          Instruction_r <= Instruction;
       else
          Instruction_r <= Instruction_r;
end
```

```
//PC
always @(posedge clk) begin
       if(rst)
          PC_r <=32'b0;
       else if (current_state == EX || ((current_state == ID) && NOP))
          PC_r <=PC_next;
       else
          PC_r <= PC_r;
end
assign PC = (current_state == IF)?PC_r:PC;

//Read_data
always @(posedge clk) begin
       if(rst)
          Read_data_r <=32'b0;
       else if (current_state == RDW && (Read_data_Valid && Read_data_Ready))
          Read_data_r <= Read_data;
       else
          Read_data_r <= Read_data_r;
end
```

3. 外设控制器访问

由于 UART 是一个 unsigned int 类型的指针，UART_STATUS 和 UART_TX_FIFO 在 UART 的基础上分别偏离了 4 个和 8 个字节。因此可以通过 UART 的地址加上 UART_STATUS 的偏移地址/4 和 UART_TX_FIF 的偏移地址/8 来访问。

```
int
puts(const char *s)
{
        //TODO: Add your driver code here
int i =0;
while (s[i] != '\0')
{
  while((*(volatile unsigned int*)(uart + UART_STATUS/4))&UART_TX_FIFO_FULL)
    ;
  (*(volatile unsigned int*)(uart + UART_TX_FIFO/8)) = s[i];
  i++;
}
return i;
}
```

4. 周期计数器

```
//the number of clock cycle
reg [31:0] cycle_cnt;
always @(posedge clk) begin
        if(rst)
          cycle_cnt <=32'b0;
        else
          cycle_cnt <= cycle_cnt + 32'b1;
end
assign cpu_perf_cnt_0 = cycle_cnt;
```

二、 实验过程中遇到的问题、对问题的思考过程及解决方法（比如 RTL 代码中出现的逻辑 bug，仿真、云平台调试过程中的难点等）

1.问题：忘记 NOP 指令在 ID 状态需要返回 IF，导致 PC 值乱加。

解决方法：经刘士祺助教的提醒，补上了 NOP 指令。

三、 对讲义中思考题（如有）的理解和回答

volatile 的作用是作为指令关键字，确保本条指令不会因编译器的优化而省略，且要求每次直接读值。

四、 在课后，你花费了大约_36_小时完成此次实验。

五、 对于此次实验的心得、感受和建议（比如实验是否过于简单或复杂，是否缺少了某些你认为重要的信息或参考资料，对实验项目的建议，对提供帮助的同学的感谢，以及其他想与任课老师交流的内容等）

本次实验不是很复杂，所有信息和资料都在 ppt 上可以找到，但是由于对看波形 debug 不太熟悉以及平台仿真，看波形需要花费太多时间导致变得非常烦躁，debug 变得不细心，搞出了更多 bug。非常感谢刘士祺助教的帮助。