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：流水线 CPU 设计

学生姓名：刘良宇

学生学号：PB20000180

完成时间：2022. 5. 1

实验题目

流水线 CPU 设计

实验目标

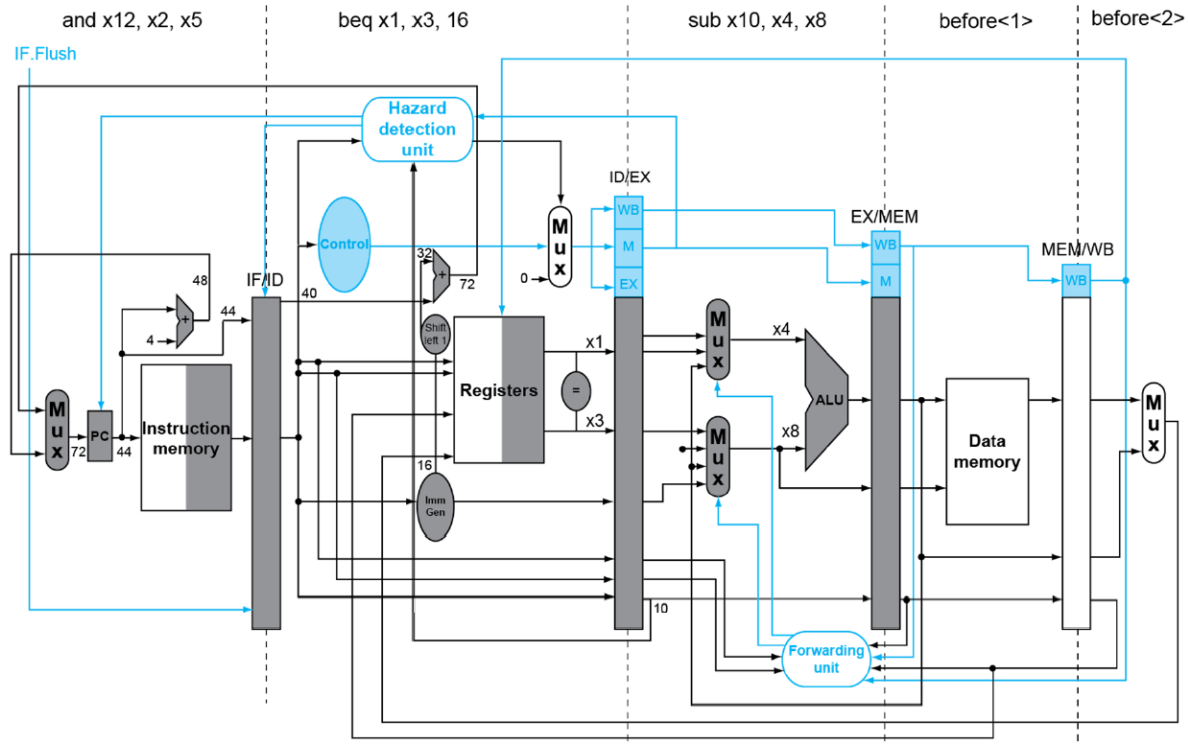
- 理解流水线 CPU 的结构和工作原理
- 掌握流水线 CPU 的设计和调试方法，特别是流水线中的数据相关和控制相关的处理
- 熟练掌握数据通路和控制器的设计和描述方法

实验环境

- Ubuntu 22.04
- Verilator 4.220
- Vlab Ubuntu 18.04
- Vivado 2019.1
- Nexys 4 DDR

实验练习

CPU 设计



主要参考该数据通路。梳理几个比较重要的流水段寄存器：

- IR, 在 ID 段译码后即可不用再传递，取而代之的是各种 ctrl 控制信号

- PC, 实际 PC 需要传递到 EX 段供给 auipc 等指令计算使用，而 IF 段计算出的 PC + 4 则一直传递（因为 jal 和 jalr 写回的是 PC + 4 ，这里相当于复用了这个 IF 段的加法器）

整体文件结构如下：

```
top_cpu.v
├── cpu.v
│   ├── hazard.v
│   ├── IF.v        // 含数据寄存器
│   ├── ID.v        // 控制单元，立即数拓展，寄存器
│   ├── EX.v        // ALU
│   ├── mem.v       // 含 mem_wrapper，是对数据寄存器和 mmio 的封装
│   ├── WB.v
└── pdu.v
```

因为具体模块细节与单周期类似，下面主要讲解不同 hazard 的处理

竞争处理

结构相关

- 哈佛结构，指令寄存器和数据寄存器分离
- 寄存器堆写优先

数据相关

数据相关主要发生在读取寄存器值时，也就是 ID 段。此时，如果发生数据相关，当前希望读取的值可能来源：

- 之前的算数逻辑指令的写回结果
- lw 从内存中访存得到

而希望读取的数据可能来源于 ID/EX 段， EX/MEM 段和 MEM 段，即分别对应当前 ID 段指令的上一条和上上条算数逻辑指令（或者 jal 和 jalr 的 pc + 4），以及 load 指令

对于算数逻辑指令，可以分别读取寄存的控制信号判断是否发生相关。如果发生相关，优先处理上一条指令的相关，如果上一条指令没有发生相关，再检查上上条指令。这样可以在发生连续相关的情况下得到正确的结果（因为上一条指令已经处理了上上条指令的相关）

load-use 相关相对而言比较麻烦，因为访存在 MEM 段，所以需要插入一个周期的气泡。这里采用 IF/ID 段停顿，ID/MEM 段清除的方式实现，具体如下：

```
// 以 rd1 为例，说明 forward 机制
reg [31:0] rd1_forward;
always @(*) begin
    rd1_forward = rd1;
    // 上上条算数逻辑指令，或者 load 指令
    if (ctrl_reg_write_EX && reg_wb_addr_EX == rs1) begin
        if (ctrl_wb_reg_src_EX == 2'b00)        // 算数逻辑
            rd1_forward = alu_out_EX;
        else if (ctrl_wb_reg_src_EX == 2'b01)   // load
            rd1_forward = mdr;
        else if (ctrl_wb_reg_src_EX == 2'b10)   // 与 alu 平行的 PC+4
            rd1_forward = pc_4_EX;
    end
    // 上条指令
    if (ctrl_reg_write_ID && reg_wb_addr_ID == rs1) begin
        if (ctrl_wb_reg_src_ID == 2'b00)        // 算数逻辑
```

```
            rd1_forward = alu_out;
        else if (ctrl_wb_reg_src_ID == 2'b10)   // 与 alu 平行的 PC+4
            rd1_forward = pc_4_ID;
    end
end

// 气泡插入，主要观察 ctrl_mem_r_ID，其他信号是用于处理控制相关的
assign stall_IF = ctrl_mem_r_ID;
assign flush_ID = pc_branch_EX | pc_jump_EX | ctrl_mem_r_ID;
```

控制相关

```
assign flush_ID = pc_branch_EX | pc_jump_EX | ctrl_mem_r_ID;
```

当前指令如果是分支指令或者是跳转指令，则统一清除 ID/EX 段

注意此时可能发生跳转，也可能不发生跳转（分支失败）

这里采取的策略是统一清除，如果不发生跳转，下一条发射的指令是当前跳转指令的 PC+4，否则下一条发射的指令是跳转后的 PC

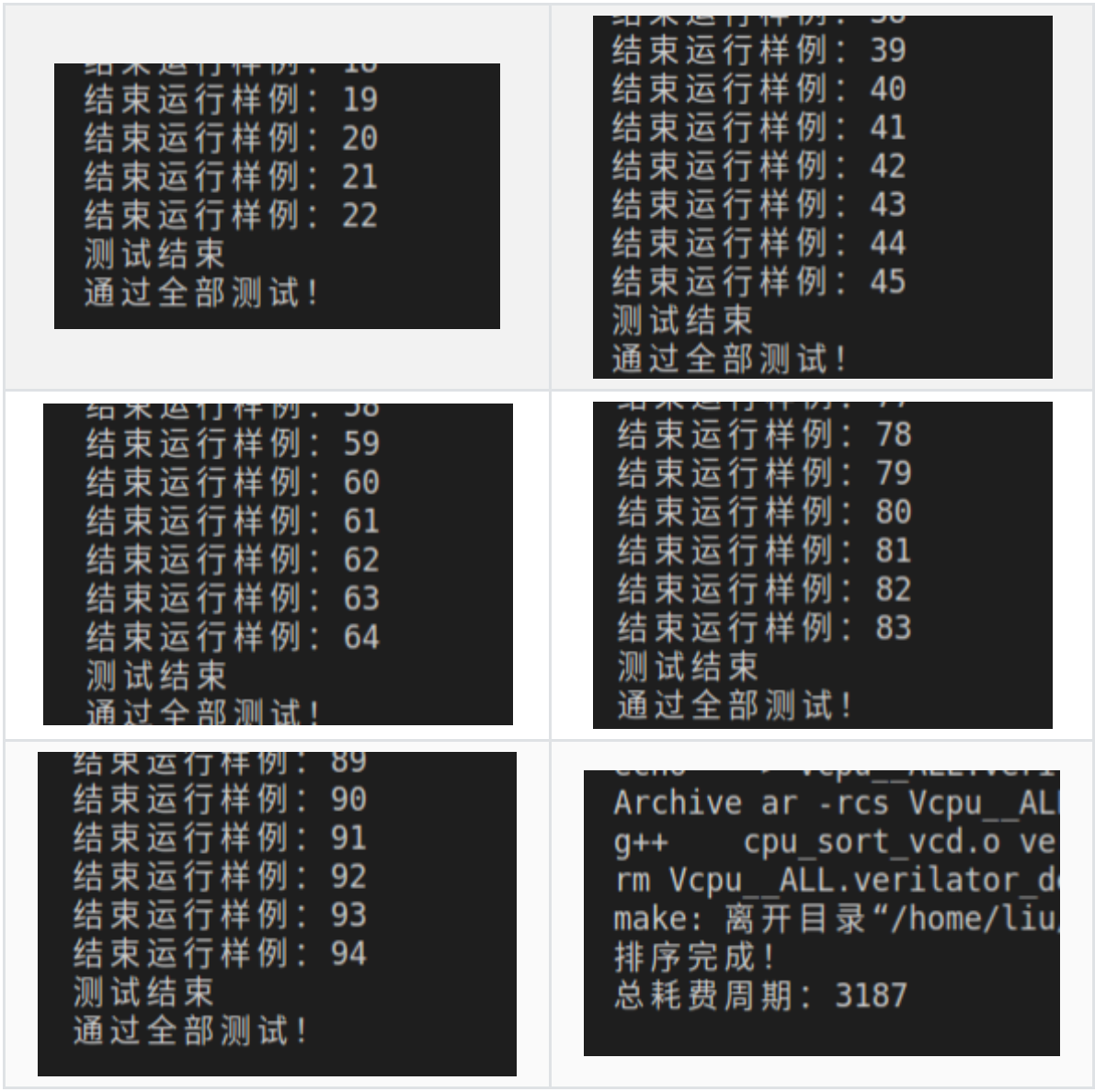
```
assign pc_branch_EX = ctrl_branch_ID[2];
assign pc_jump_EX   = ctrl_jump_ID;
wire branch_success = (ctrl_branch_ID[2] & (((ctrl_branch_ID[1] == 1)? alu_f[1] :
alu_f[0]) ^ ctrl_branch_ID[0]));
assign pc_nxt_EX    = (branch_success | pc_jump_EX) ? ((ctrl_pc_add_src_ID? rd1_ID:
pc_ID) + imm_ID) : pc_4_ID;
```

（无论是否跳转，pc_nxt_pc 总是下一条执行的指令地址）

仿真模拟

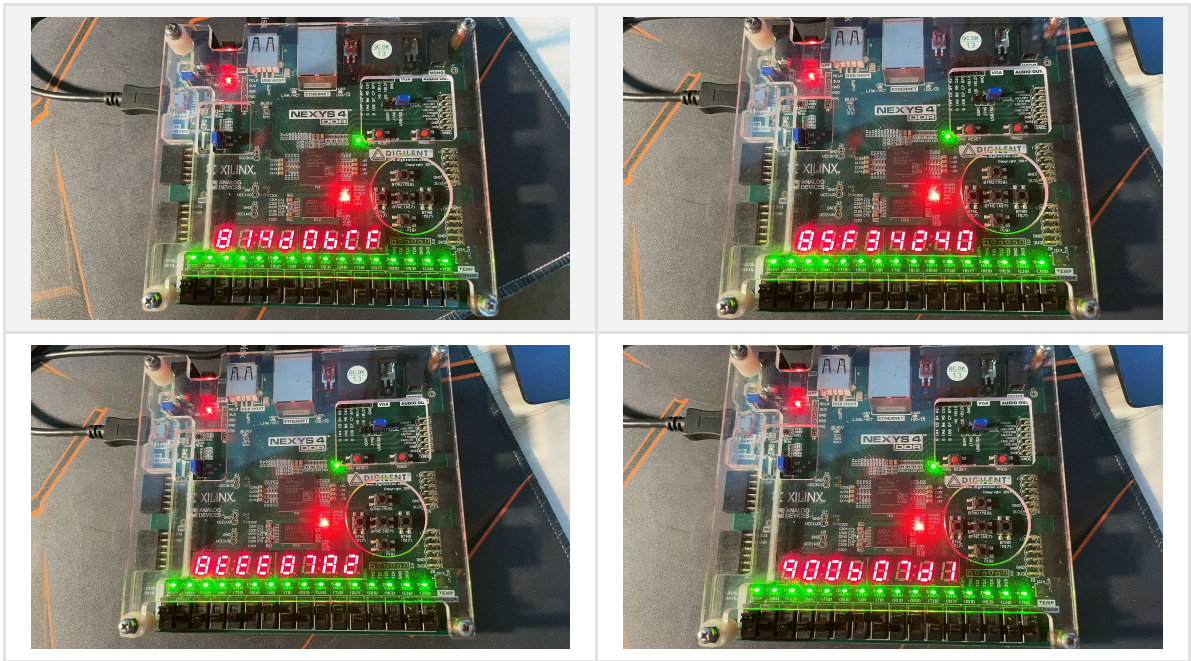
这里通过了完整的 `bypass` 测试，验证了本 CPU 对竞争处理的完备

也额外进行了排序测试（对比：单周期 cpu 需要耗费 2000 周期左右）



功能展示

这里我们同上次，展示查询式输出（MMIO）对排序程序进行演示



综合电路

资源使用

Name	Slice LUTs (63400)	Slice Registers (126800)	F7 Muxes (31700)	F8 Muxes (15850)	Bonded IOB (210)	BUFGCTRL (32)
top_cpu	1518	900	192	32	58	4
CPU (cpu)	1073	544	192	32	0	0
u_EX (EX)	114	105	0	0	0	0
alu32 (alu)	10	0	0	0	0	0
u_ID (ID)	417	175	0	0	0	0
register (register_file)	72	0	0	0	0	0
u_IF (IF)	178	96	64	32	0	0
ir_mem (dist_ir)	128	32	64	32	0	0
u_MEM (MEM)	364	168	128	0	0	0
PDU (pdu)	445	356	0	0	0	0

电路性能

Name	Slack	Levels	High Fanout	From	To	Total Delay	Logic Delay	Net Delay
Path 1	5.630	6	184	PDU/cnt_clk_r_reg[1]/C	PDU/cnt_clk_r_reg[17]/D	4.260	1.902	2.358
Path 2	5.651	6	184	PDU/cnt_clk_r_reg[1]/C	PDU/cnt_clk_r_reg[19]/D	4.239	1.881	2.358
Path 3	5.725	6	184	PDU/cnt_clk_r_reg[1]/C	PDU/cnt_clk_r_reg[18]/D	4.165	1.807	2.358
Path 4	5.741	6	184	PDU/cnt_clk_r_reg[1]/C	PDU/cnt_clk_r_reg[16]/D	4.149	1.791	2.358
Path 5	5.857	5	184	PDU/cnt_clk_r_reg[1]/C	PDU/cnt_clk_r_reg[13]/D	4.145	1.788	2.357
Path 6	5.878	5	184	PDU/cnt_clk_r_reg[1]/C	PDU/cnt_clk_r_reg[15]/D	4.124	1.767	2.357
Path 7	5.952	5	184	PDU/cnt_clk_r_reg[1]/C	PDU/cnt_clk_r_reg[14]/D	4.050	1.693	2.357
Path 8	5.968	5	184	PDU/cnt_clk_r_reg[1]/C	PDU/cnt_clk_r_reg[12]/D	4.034	1.677	2.357
Path 9	5.971	4	184	PDU/cnt_clk_r_reg[1]/C	PDU/cnt_clk_r_reg[9]/D	4.031	1.674	2.357
Path 10	5.992	4	184	PDU/cnt_clk_r_reg[1]/C	PDU/cnt_clk_r_reg[11]/D	4.010	1.653	2.357

总结与思考

- 本次实验使我理解了流水线 CPU 的设计，加深了对数据通路的理解
- 本次实验整体体验良好