浙江大学

本科实验报告

课程名称：计算机体系结构

姓　　名：郑乔尹 董佳鑫

学　　院：计算机科学与技术学院

　　系：计算机科学与技术系

专　　业：计算机科学与技术

学　　号：3210104169 3210102181

指导教师：姜晓红

2023 年 11 月 30 日

浙江大学实验报告

课程名称：计算机体系结构 实验类型：综合

实验项目名称：支持多周期乱序执行的流水线 CPU

学生姓名：郑乔尹 专业：计算机科学与技术 学号：3210104169

同组学生姓名：董佳鑫 指导老师：姜晓红

实验地点：曹西 301 实验日期：2023 年 11 月 30 日

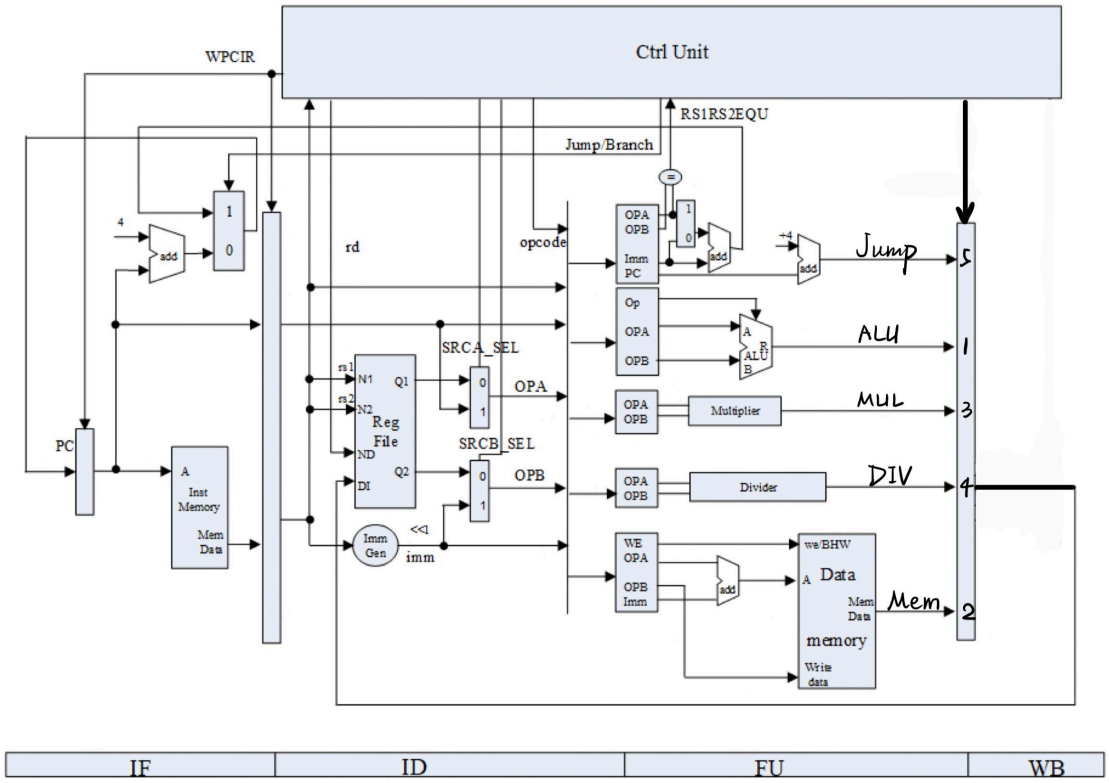
一、 实验目的和要求

Task 1: 写出本次实验的目的与要求 (5 points)

1. 理解多周期流水线 CPU 的原理
2. 掌握支持多周期乱序执行的流水线 CPU 的设计方法
3. 掌握多周期流水线 CPU 的验证方法

二、 实验内容和原理

Task 2: 简要画出本次实验实现的 core 的电路图。（可以使用 PPT 上的线路图进行修改，但是必须和自己的实现保持一致。如果不一致，本题将不给分）(10 points)



Task 3: 请给出四种 hazard 的处理逻辑的代码并加以解释 (20 points)

写后写：

```
reg[5:0] f;
wire WAW = rd && ((rd == FU_write_to[1] && !f[1]) ||
                  (rd == FU_write_to[2] && !f[2]) ||
                  (rd == FU_write_to[3] && !f[3]) ||
                  (rd == FU_write_to[4] && !f[4]) ||
                  (rd == FU_write_to[5] && !f[5]));
// read after write
```

通过比较 rd 与五个 FU_write_to 中的内容，如果 FU_write_to 中有相同的写目标寄存器，说明当前有 FU 正在准备写入该寄存器，发生 WAW hazard，但是需要考虑***当倒计时小于等于当前模块的执行需要延迟时***，WAW 消除的情况，因为此时发射当前指令，可以保证在写回时按照正确的写回顺序，所以我添加了一个标志寄存器 f，当 WAW 依赖的前一条指令执行到后一条指令的倒计时以下时，f[i]置 1，WAW 可以解除。

写后读：

```
wire RAW_rs1 = rs1 && ((rs1 == FU_write_to[1]) ||
                       (rs1 == FU_write_to[2]) ||
                       (rs1 == FU_write_to[3]) ||
                       (rs1 == FU_write_to[4]) ||
                       (rs1 == FU_write_to[5]));
wire RAW_rs2 = rs2 && ((rs2 == FU_write_to[1]) ||
```

```
                            (rs2 == FU_write_to[2]) ||
                            (rs2 == FU_write_to[3]) ||
                            (rs2 == FU_write_to[4]) ||
                            (rs2 == FU_write_to[5]));
```

通过比较 rs1/rs2 与五个 FU 中此时正在写入的寄存器，检查是否发生 RAW。

预约站预约冲突、FU busy：(但是实际上我把所有需要 stall 的 hazard 都放在这里)

```
    wire FU_hazard = (FU_status[use_FU] == 1 && reservation_reg[0] != use_FU)
                        || WAW
                        || RAW_rs1
                        || RAW_rs2
                        || reservation_reg[FU_delay_cycles[use_FU]] !=
0;
```

如果当前使用的FU的FU_status为1，代表该模块正被前面发射的指令占用，但是要注意检测前一条指令是否已经执行到最后，即如果其在预约站中已经倒计时到 reservation_reg[0]，则已经可以进行发射，因为寄存器的更新总是晚一个周期。

如果当前指令可以被预约，但是要预约进预约站的位置有另一条指令刚好占用(比如 div 指令倒计时到 6，而当前指令是 mul，这就导致它无法直接预约进 reservation_reg[6])，就需要 stall 一个周期，再进行预约。

Task 4: 请给出发射逻辑的代码并加以解释 (10 points)

```
if(valid_ID) begin
    if (use_FU == 0) begin //  check whether FU is used
        //! to fill sth.in
        for (i = 0; i < 31; i=i+1) begin
            reservation_reg[i] <= reservation_reg[i + 1];
        end
        reservation_reg[31] <= 0;
    end
    else if (FU_hazard | reg_ID_flush | reg_ID_flush_next) begin   // stall
        //! to fill sth.in
        // FU_writeback_en[use_FU] <= 0;
        B_in_FU <= 0;
        J_in_FU <= 0;
        for (i = 0; i < 31; i=i+1) begin
            if (reservation_reg[FU_delay_cycles[use_FU]]) begin
                f[reservation_reg[FU_delay_cycles[use_FU]]] <= 1'b1;
            end
            reservation_reg[i] <= reservation_reg[i + 1];
        end
```

```
        reservation_reg[31] <= 0;
    end
    else begin  // regist FU operation
        //! to fill sth.in
        for(i = 0; i < 6; i = i + 1) begin
            f[i] <= 0;
        end
        FU_status[use_FU] <= 1;
        FU_writeback_en[use_FU] <= 1'b1 & rd_used;
        FU_write_to[use_FU] <= rd & {5{rd_used}};
        reservation_reg[FU_delay_cycles[use_FU] - 1] <= use_FU;
        for (i = 0; i < 31; i=i+1) begin
            if(i != FU_delay_cycles[use_FU] - 1) begin
                reservation_reg[i] <= reservation_reg[i + 1];
            end
        end
        reservation_reg[31] <= 0;
        B_in_FU <= B_valid;
        J_in_FU <= JAL | JALR;
    end
end
```

首先检查 valid_ID，确认当前 ID 阶段的指令是否为有效指令(排除第一个时钟周期的 ID)。ID 阶段指令有效时，假如当前指令没有用到 FU，预约站进行左移，倒计时继续。

如果发生跳转，reg_ID_flush 被拉高，或者发生了 ***WAW,RAW,FU busy Hazard*** 或者 ***预约站预约冲突***，需要进行 stall，等待 hazard 结束，但是此时已经发射的指令仍然需要继续执行，预约站持续左移即可。

若需要使用 FU，且未发生 hazard 也并未发生跳转，则发射当前指令（f[1]-f[5]是用于检测 WAW_hazard 的结束标志）。将当前使用到的 FU 的 FU_status 置 1，代表正在占用，更新对应 FU 的 FU_write_to 和 FU_writeback_en(需要检测是否有目标寄存器 rd，故和 rd_used 按位与)，为预约站中写入当前 FU 的序号，注意需要往左写一个位置，代表当前已经经过一个时钟周期，然后将剩下的部分左移，避开写入的那一位，防止数据被覆盖，同时更新 B_in_FU 和 J_in_FU（供跳转指令使用）。

Task 5: 请给出 branch 处理的逻辑、代码以及解释 (5 points)

```
    assign reg_IF_en = ~FU_hazard | branch_ctrl;
    assign reg_ID_en = reg_IF_en;
```

```
assign branch_ctrl = (B_in_FU & cmp_res_FU) |  J_in_FU;
always @ (posedge clk or posedge rst) begin
    if (rst) begin
        reg_ID_flush_next <= 0;
    end
    else begin
        reg_ID_flush_next <= branch_ctrl;
    end
end
assign reg_ID_flush = branch_ctrl;
```

本实验采取 predict-not-taken 策略，如果正确跳转(branch_ctrl==1)应当将错误译码的指令 flush。但是 reg_ID_flush 只能 flush 当前周期内 ID 阶段的指令，故添加 reg_flush_next，其作用是 flush 下一个周期的指令，因为当前周期 IF 阶段的指令在下一周期会进入 ID 阶段。

Task 6: 为什么本实验不需要处理 WAR hazard？(5 points)

因为本实验的 CPU 是按序发射的，寄存器操作数在 ID 阶段取到，下一条指令即使再快也需要额外的两个周期才能写回，并不能影响到上一条指令读取的操作数。

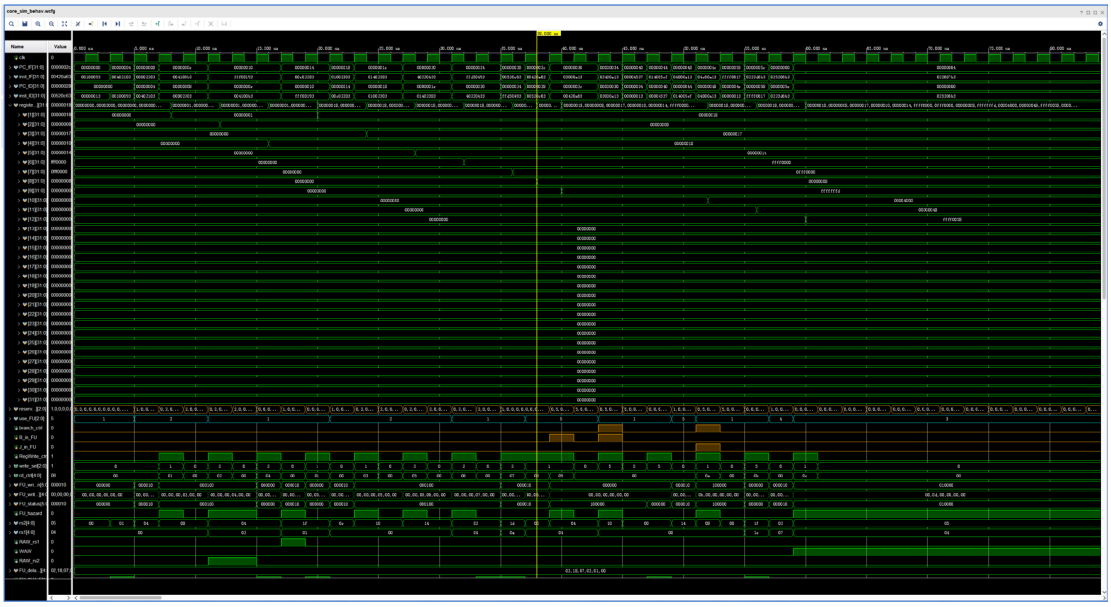
三、 实验过程和数据记录及结果分析

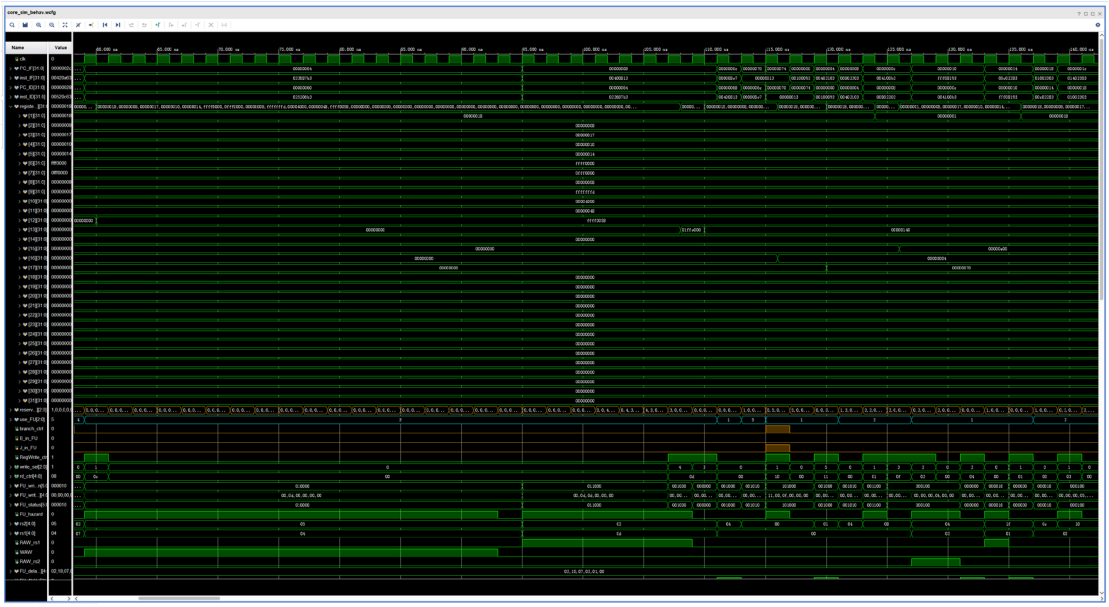
仿真图片应完整包含时间信息和信号名称。

对仿真的解释示例：XXXns，X 信号变为 X，由于 XXX，导致 X 信号变为 XXX，……，我们发现 X 被 forward 到了 X。

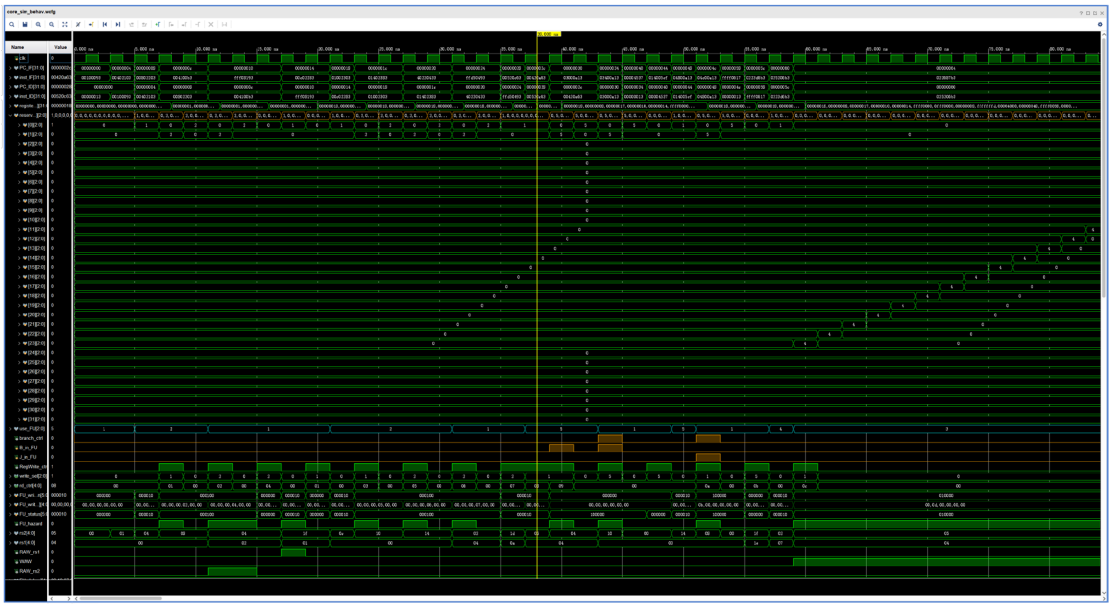
Task 7: 请给出本次实验仿真的完整截图 (5 points)

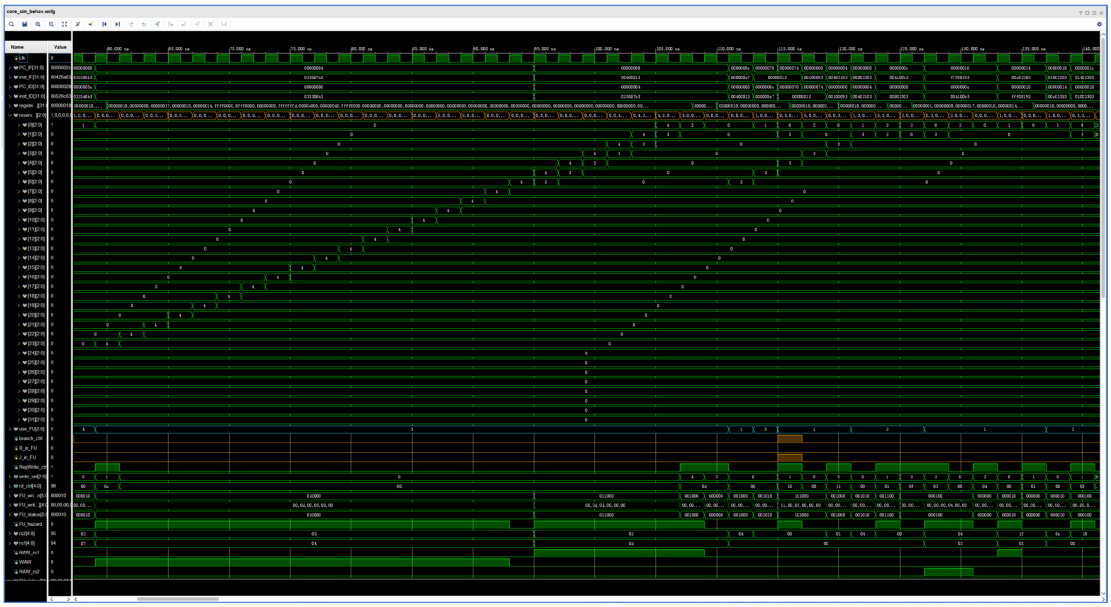
寄存器：



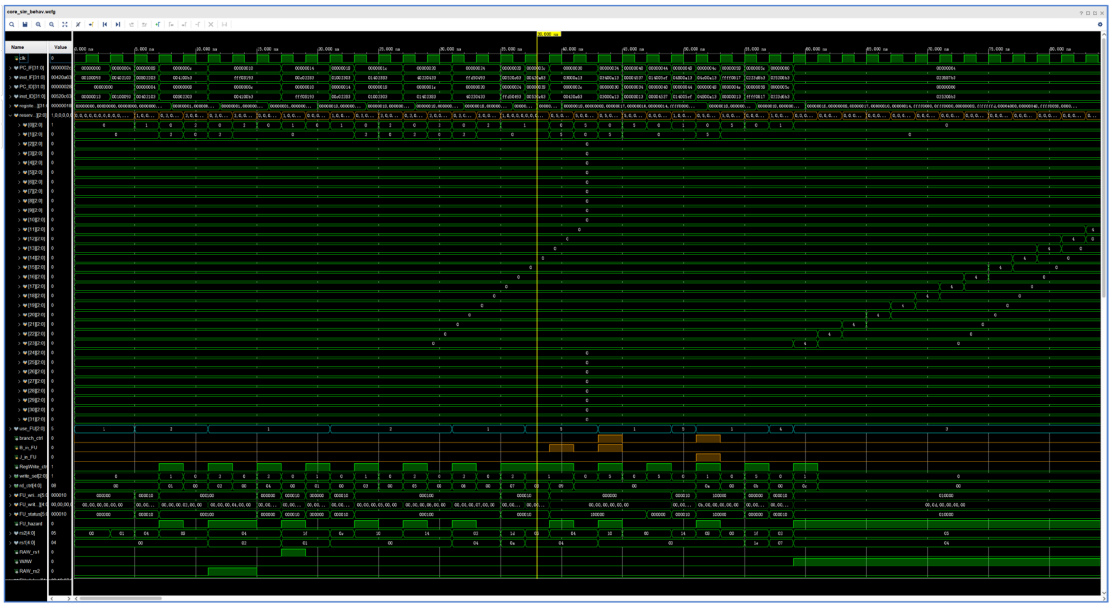


预约站：



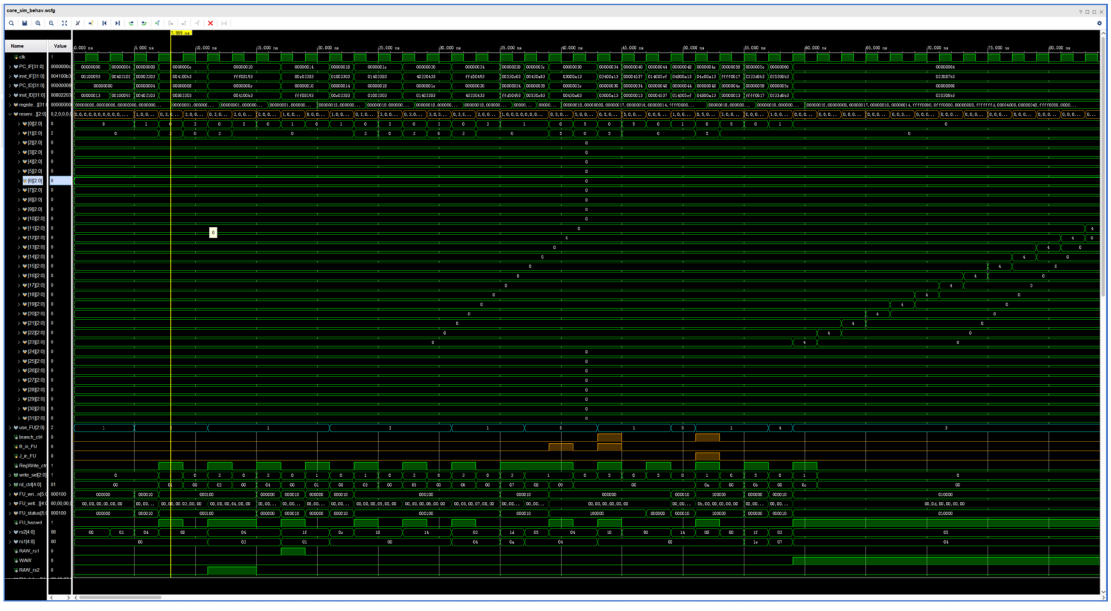


Task 8: 请给出一个某条指令发射时产生 Writing Competition hazard 部分仿真的高清图片，并对涉及到的信号加以详细解释 (10 points)



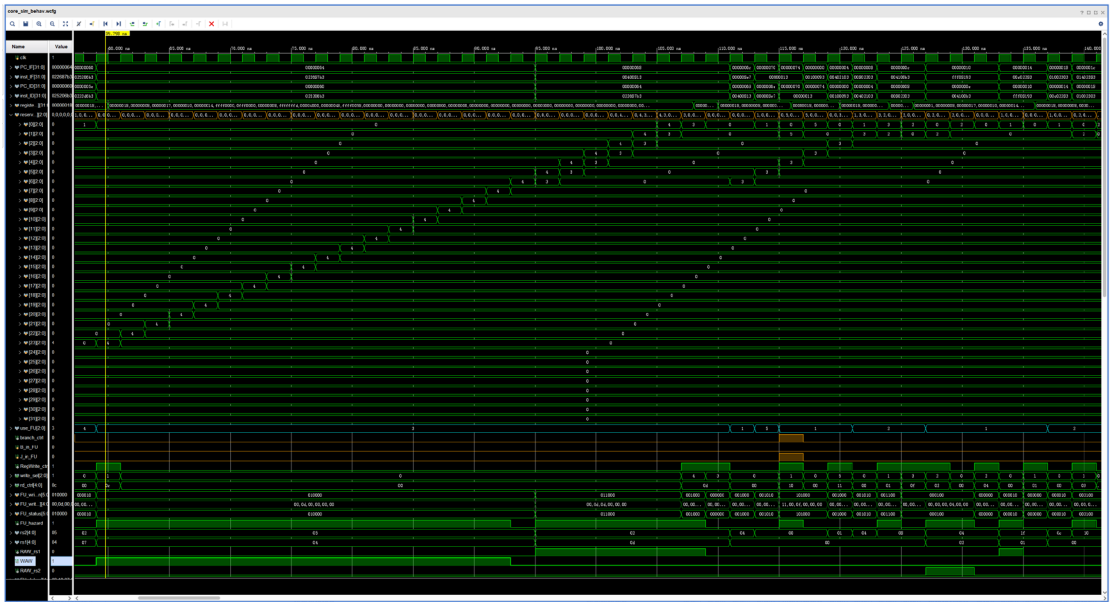
31000ns 时，PC=0x20 指令(sub)进入 ID 阶段，通过 RAW_rs1，RAW_rs2，WAW 可以看出当前并无此类 hazard，且 FU_status[1]=0(ALU 功能模块并未被占用)，故当前 sub 指令理论上已经可以被发射。然而此时预约站中的指令倒计时刚好来到了 reservation_reg[1]，下一个周期即将进入 reservation_reg[0]，假如此时预约 sub 指令，那么下一个周期 sub 指令的 FU 号将会被预约在 reservation_reg[0]，导致二者发生冲突，这里可以看到，CtrlUnit 正确检测了这一情况，将 FU_hazard 置 1 并成功 stall 了一个周期再发射 sub 指令。

Task 9: 请给出一个某条指令发射时产生 Structure Hazard(FU unit busy)部分仿真的高清图片，并对涉及到的信号加以详细解释 (10 points)



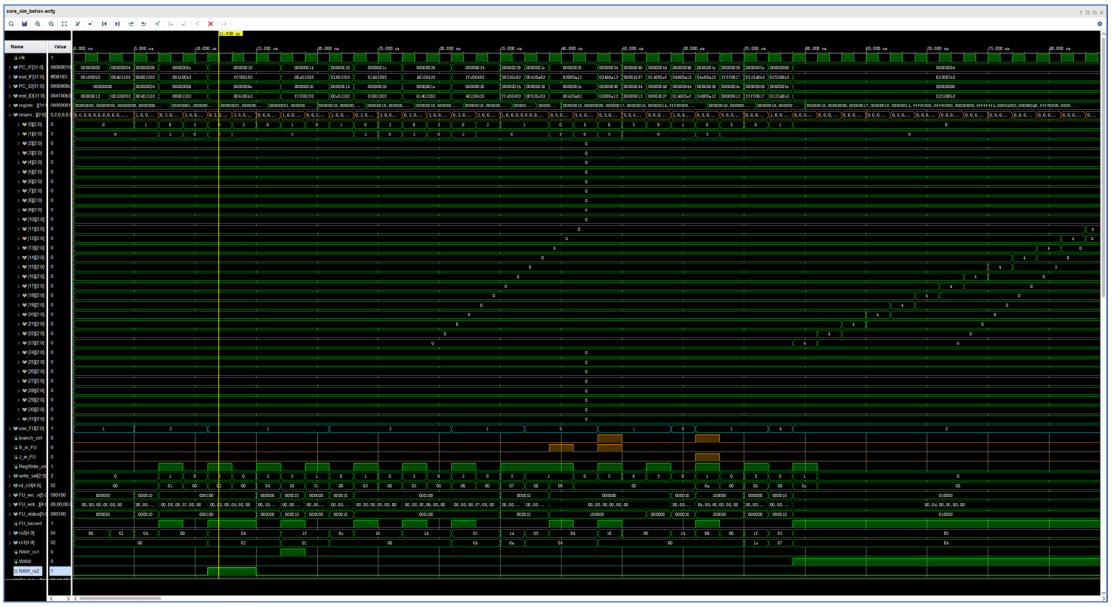
7000ns 时，PC=0x08 的 lw 指令进入 ID 阶段，RAW_rs1，RAW_rs2，WAW 信号均为 0，然而 FU_status[2]=1，代表 MEM 操作对应的 FU 正在使用，FU busy，需要 stall 直到 FU 解除占用，可以看到当前一条指令在预约站中倒计时结束时(11000ns 时)，stall 也结束，PC=0x0c 的 lw 指令成功发射。

Task 10: 请给出一个某条指令发射时产生 WAW hazard 部分仿真的高清图片，并对涉及到的信号加以详细解释 (10 points)



59000ns 时，PC=0x60 的 mul x13,x4,x5 指令进入 ID 阶段然而上一条的 divu x13,x7,x2 指令还未将 x13 写回，导致发生 WAW hazard，可以看到 WAW 信号置 1，开始 stall。等到 93000ns 时，DIV 模块的倒计时来到 reservation_reg[6]，而 MUL 模块的 delay 仅为 7，故下一周期发射，可以使得 mul 恰好在 divu 指令的下一周期写回，WAW hazard 此时不再存在，下一周期(95000ns)，mul 指令成功发射。

Task 11: 请给出一个某条指令发射时产生 RAW hazard 部分仿真的高清图片，并对涉及到的信号加以详细解释 (10 points)



11000ns 时，PC=0xc 的 add x1,x2,x4 指令进入 ID 阶段，而前一条指令为 lw x4,8(x0)，此时还未写回（FU_write_to[2]==4），发生 RAW hazard，需要进行 stall，等待 x4 写回。可以看到，在 x4 的写回周期前（15000ns），FU_write_to[2]已经被清空，RAW hazard 结束；在 x4 的写回结束后（16000-17000ns），add 指令成功被发射（17000ns）。

四、 讨论与心得

Task 12: 请写出对本次实验内容的深入讨论，或者本次实验的心得体会。例如遇到的难题等等。请认真填写本模块，若不填写或胡乱填写将酌情扣分，写明白真实情况即可。 (+10 points)

1. 在处理预约站的倒计时时，对于倒计时开始的操作一开始出现了问题，在发射时忘记直接-1，导致执行慢了一个周期
2. 由于寄存器的更新总是在下一个周期，导致如果 FU_mem 的使能信号直接作为 RAM 的使能信号会导致写回时无法读取到正确的 data，通过手动在 MEM 中添加了一个新的使能寄存器，将 RAM 的读取延迟一个周期，解决了这一问题。