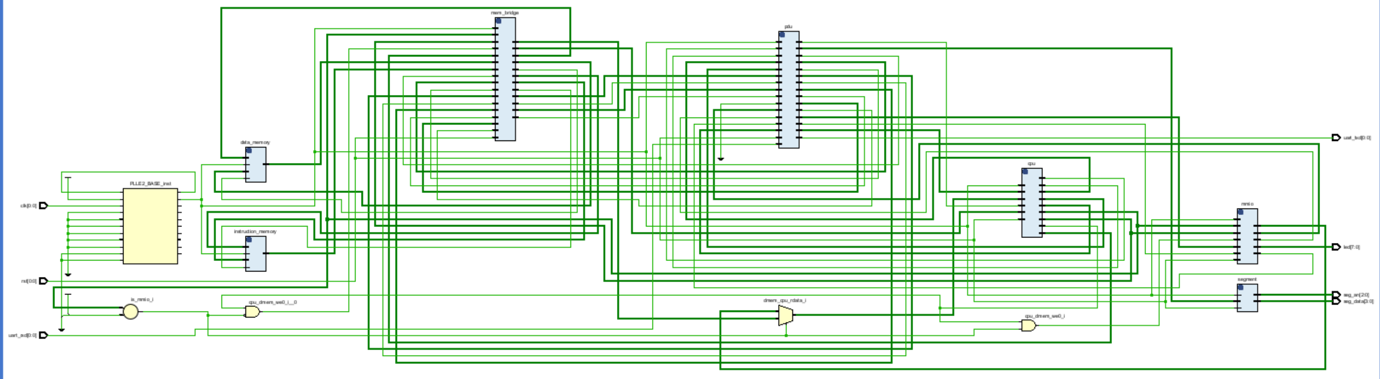
Lab 3 report

实验目的与内容

1. **简述本次实验做的内容以及本次实验的目的、逻辑设计**
    - 通过 Verilog 硬件描述语言实现一个可用的简单单周期 CPU。实验三进行单周期CPU底层与寄存器指令相关的功能部件的搭建。
    - 此次实验在实验二的基础上进行R型指令的执行，实现CPU的各子模块的设计，包括实现PC计数器、多路选择器等模块，最后需要在龙芯框架中运行仿真测试并上板验证结果。
2. 请参照PPT画出你设计的各模块的框图和数据通路，



3. 如果存在状态机，请绘制出状态机的状态转换图；

> 无。

4. 请贴出较为核心的代码设计代码，并加以解释说明。
    见第二部分。

具体实验任务

PC 寄存器

任务：设计符合要求的 PC 寄存器，满足计数要求并有输出以供下一步在指令存储器中取出指令。

```
module PC (
    input                   [ 0 : 0]            clk,
    input                   [ 0 : 0]            rst,
    input                   [ 0 : 0]            en,
    input                   [31 : 0]            npc,

    output      reg         [31 : 0]            pc
);

always @(posedge clk or posedge rst) begin
    if(rst)begin
        pc <= 32'h1C000000;
    end
    else if(en)begin
        pc <= npc;
    end
end

endmodule
```

- 此外，按照cpu数据通路的要求， `pc` 自增的操作在cpu模块中完成而不在PC 寄存器中完成。

```
wire    [31 : 0]    cur_pc,cur_npc,cur_inst;
assign cur_npc = cur_pc + 32'h4;//CPU模块中的wire操作
```

- 多路选择器和寄存器堆已经在 `lab2` 中实现，此处不再赘述。

译码器

任务：负责根据输入的指令生成相应的控制与数据信号。本次实验需要实现所有寄存器指令的译码。根据龙芯的编码规则需要分别取出指令中源寄存器，目的寄存器，操作码以及生成寄存器堆的写使能信号。

具体代码如下：

> 其中还需要产生最后输入ALU的源操作数的来源，因此译码器还需要产生两个选择信号，alu_src0来自pc还是rf_rd0，alu_src1来自rf_rd1还是imm。

```verilog
always @(*) begin
    if(inst[31:15] == 17'b0000_0000_0001_00000)begin//add.w rd,rj,rk
        alu_op = 5'b00000;
        imm = 32'b0;
        rf_ra0 = inst[9 : 5];           //ra0 = rj
        rf_ra1 = inst[14:10];           //ra1 = rk
        rf_wa  = inst[4 : 0];           //wa  = rd
        rf_we  = 1'b1;
        alu_src0_sel = 1'b1;
        alu_src1_sel = 1'b0;
    end

    else if(inst[31:22] == 10'b00000_01010)begin//addi.w rd,rj,si12
        alu_op = 5'b00000;
        imm = {{20{inst[21]}},inst[21:10]};
        rf_ra0 = inst[9 : 5];           //ra0 = rj
        rf_ra1 = inst[14:10];           //ra1 = rk
        rf_wa  = inst[4 : 0];           //wa  = rd
        rf_we  = 1'b1;
        alu_src0_sel = 1'b1;
        alu_src1_sel = 1'b1;
    end

    else if(inst[31:15] == 17'b0000_0000_0001_00010)begin//sub.w rd,rj,rk
        alu_op = 5'b00010;
        imm = 32'b0;
        rf_ra0 = inst[9 : 5];           //ra0 = rj
        rf_ra1 = inst[14:10];           //ra1 = rk
        rf_wa  = inst[4 : 0];           //wa  = rd
        rf_we  = 1'b1;
        alu_src0_sel = 1'b1;
        alu_src1_sel = 1'b0;
    end

    else if(inst[31:15] == 17'b0000_0000_0001_00100)begin//slt rd,rj,rk
        alu_op = 5'b00100;
        imm = 32'b0;
        rf_ra0 = inst[9 : 5];           //ra0 = rj
        rf_ra1 = inst[14:10];           //ra1 = rk
        rf_wa  = inst[4 : 0];           //wa  = rd
        rf_we  = 1'b1;
        alu_src0_sel = 1'b1;
        alu_src1_sel = 1'b0;
```

```
end

else if(inst[31:22] == 10'b00000_01000)begin//slti.w rd,rj,si12
    alu_op = 5'b00100;
    imm = {{20{inst[21]}},inst[21:10]};
    rf_ra0 = inst[9 : 5];           //ra0 = rj
    rf_ra1 = inst[14:10];           //ra1 = rk
    rf_wa  = inst[4 : 0];           //wa  = rd
    rf_we  = 1'b1;
    alu_src0_sel = 1'b1;
    alu_src1_sel = 1'b1;
end

else if(inst[31:15] == 17'b0000_0000_0001_00101)begin//sltu rd,rj,rk
    alu_op = 5'b00101;
    imm = 32'b0;
    rf_ra0 = inst[9 : 5];           //ra0 = rj
    rf_ra1 = inst[14:10];           //ra1 = rk
    rf_wa  = inst[4 : 0];           //wa  = rd
    rf_we  = 1'b1;
    alu_src0_sel = 1'b1;
    alu_src1_sel = 1'b0;
end

else if(inst[31:22] == 10'b00000_01001)begin//sltui.w rd,rj,si12
    alu_op = 5'b00101;
    imm = {{20{inst[21]}},inst[21:10]};
    rf_ra0 = inst[9 : 5];           //ra0 = rj
    rf_ra1 = inst[14:10];           //ra1 = rk
    rf_wa  = inst[4 : 0];           //wa  = rd
    rf_we  = 1'b1;
    alu_src0_sel = 1'b1;
    alu_src1_sel = 1'b1;
end

else if(inst[31:15] == 17'b0000_0000_0001_01001)begin//and rd,rj,rk
    alu_op = 5'b01001;
    imm = 32'b0;
    rf_ra0 = inst[9 : 5];           //ra0 = rj
    rf_ra1 = inst[14:10];           //ra1 = rk
    rf_wa  = inst[4 : 0];           //wa  = rd
    rf_we  = 1'b1;
    alu_src0_sel = 1'b1;
```

```
        alu_src1_sel = 1'b0;
    end

    else if(inst[31:22] == 10'b00000_01101)begin//andi.w rd,rj,usi12
        alu_op = 5'b01001;
        imm = {{20{1'b0}},inst[21:10]};
        rf_ra0 = inst[9 : 5];           //ra0 = rj
        rf_ra1 = inst[14:10];           //ra1 = rk
        rf_wa  = inst[4 : 0];           //wa  = rd
        rf_we  = 1'b1;
        alu_src0_sel = 1'b1;
        alu_src1_sel = 1'b1;
    end

    else if(inst[31:15] == 17'b0000_0000_0001_01010)begin//or rd,rj,rk
        alu_op = 5'b01010;
        imm = 32'b0;
        rf_ra0 = inst[9 : 5];           //ra0 = rj
        rf_ra1 = inst[14:10];           //ra1 = rk
        rf_wa  = inst[4 : 0];           //wa  = rd
        rf_we  = 1'b1;
        alu_src0_sel = 1'b1;
        alu_src1_sel = 1'b0;
    end

    else if(inst[31:22] == 10'b00000_01110)begin//ori.w rd,rj,si12
        alu_op = 5'b01010;
        imm = {{20{1'b0}},inst[21:10]};
        rf_ra0 = inst[9 : 5];           //ra0 = rj
        rf_ra1 = inst[14:10];           //ra1 = rk
        rf_wa  = inst[4 : 0];           //wa  = rd
        rf_we  = 1'b1;
        alu_src0_sel = 1'b1;
        alu_src1_sel = 1'b1;
    end

    else if(inst[31:15] == 17'b0000_0000_0001_01011)begin//xor rd,rj,rk
        alu_op = 5'b01011;
        imm = 32'b0;
        rf_ra0 = inst[9 : 5];           //ra0 = rj
        rf_ra1 = inst[14:10];           //ra1 = rk
        rf_wa  = inst[4 : 0];           //wa  = rd
        rf_we  = 1'b1;
        alu_src0_sel = 1'b1;
        alu_src1_sel = 1'b0;
    end

    else if(inst[31:22] == 10'b00000_01111)begin//xori.w rd,rj,si12
        alu_op = 5'b01011;
        imm = {{20{1'b0}},inst[21:10]};
        rf_ra0 = inst[9 : 5];           //ra0 = rj
        rf_ra1 = inst[14:10];           //ra1 = rk
        rf_wa  = inst[4 : 0];           //wa  = rd
        rf_we  = 1'b1;
        alu_src0_sel = 1'b1;
        alu_src1_sel = 1'b1;
    end

    else if(inst[31:15] == 17'b0000_0000_0001_01110)begin//sll.w rd,rj,rk
        alu_op = 5'b01110;
        imm = 32'b0;
        rf_ra0 = inst[9 : 5];           //ra0 = rj
        rf_ra1 = inst[14:10];           //ra1 = rk
        rf_wa  = inst[4 : 0];           //wa  = rd
        rf_we  = 1'b1;
        alu_src0_sel = 1'b1;
        alu_src1_sel = 1'b0;
    end

    else if(inst[31:15] == 17'b0000_0000_0100_00001)begin//slli.w rd,rj,si12
        alu_op = 5'b01110;
        imm = {{27{1'b0}},inst[14:10]};
        rf_ra0 = inst[9 : 5];           //ra0 = rj
        rf_ra1 = inst[14:10];           //ra1 = rk
        rf_wa  = inst[4 : 0];           //wa  = rd
        rf_we  = 1'b1;
        alu_src0_sel = 1'b1;
        alu_src1_sel = 1'b1;
    end

    else if(inst[31:15] == 17'b0000_0000_0001_01111)begin//srl.w rd,rj,rk
        alu_op = 5'b01111;
        imm = 32'b0;
        rf_ra0 = inst[9 : 5];           //ra0 = rj
        rf_ra1 = inst[14:10];           //ra1 = rk
        rf_wa  = inst[4 : 0];           //wa  = rd
```

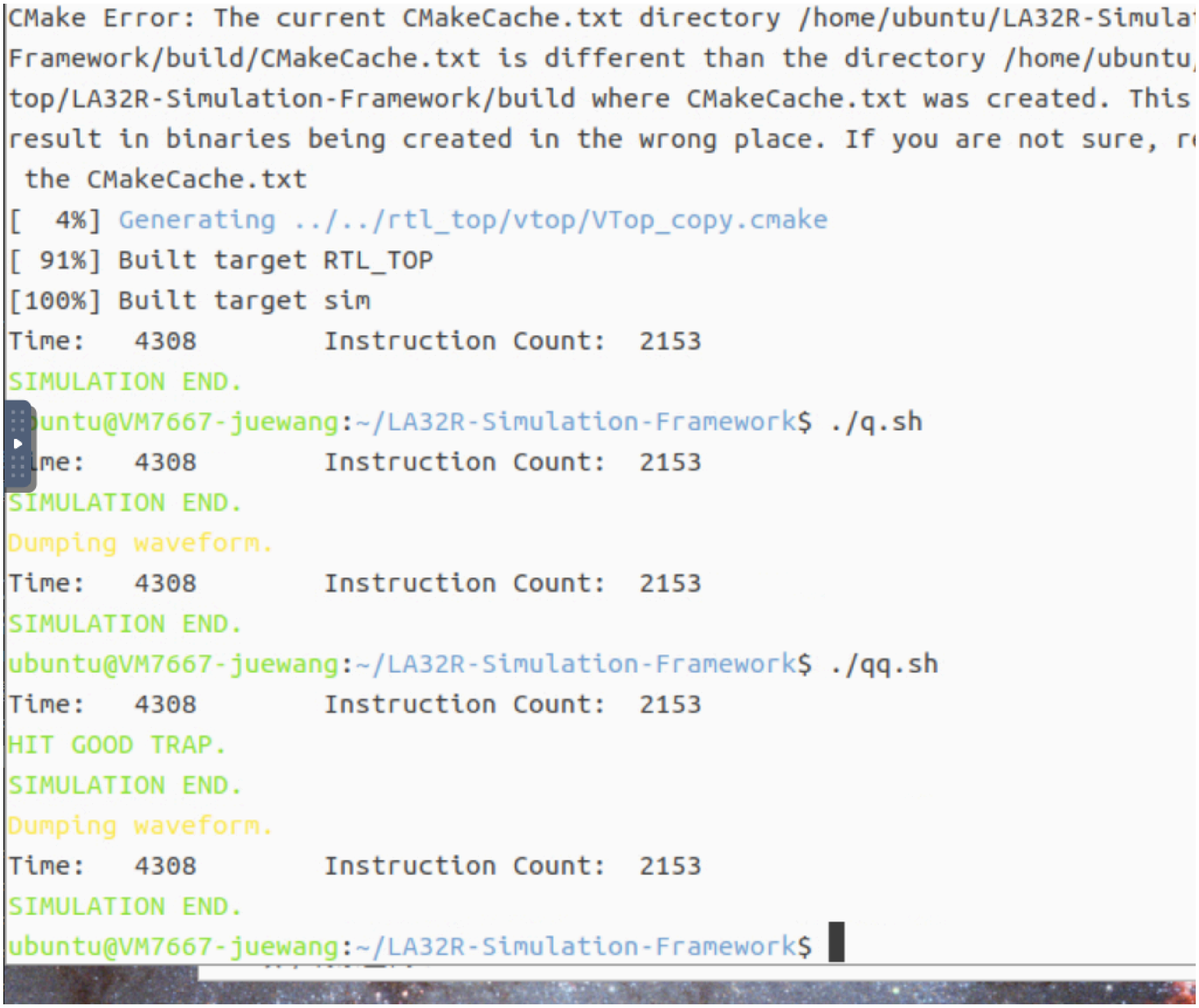
```verilog
    rf_we  = 1'b1;
    alu_src0_sel = 1'b1;
    alu_src1_sel = 1'b0;
end

else if(inst[31:15] == 17'b0000_0000_0100_01001)begin//srli.w rd,rj,si12
    alu_op = 5'b01111;
    imm = {{27{1'b0}},inst[14:10]};
    rf_ra0 = inst[9 : 5];           //ra0 = rj
    rf_ra1 = inst[14:10];           //ra1 = rk
    rf_wa  = inst[4 : 0];           //wa  = rd
    rf_we  = 1'b1;
    alu_src0_sel = 1'b1;
    alu_src1_sel = 1'b1;
end

else if(inst[31:15] == 17'b0000_0000_0001_10000)begin//sra.w rd,rj,rk
    alu_op = 5'b10000;
    imm = 32'b0;
    rf_ra0 = inst[9 : 5];           //ra0 = rj
    rf_ra1 = inst[14:10];           //ra1 = rk
    rf_wa  = inst[4 : 0];           //wa  = rd
    rf_we  = 1'b1;
    alu_src0_sel = 1'b1;
    alu_src1_sel = 1'b0;
end

else if(inst[31:15] == 17'b0000_0000_0100_10001)begin//srai.w rd,rj,si12
    alu_op = 5'b10000;
    imm = {{27{1'b0}},inst[14:10]};
    rf_ra0 = inst[9 : 5];           //ra0 = rj
    rf_ra1 = inst[14:10];           //ra1 = rk
    rf_wa  = inst[4 : 0];           //wa  = rd
    rf_we  = 1'b1;
    alu_src0_sel = 1'b1;
    alu_src1_sel = 1'b1;
end

else if(inst[31:25] == 7'b000_1010)begin//lu12i.w rd,si20
    alu_op = 5'b10011;
    imm = {inst[24:5],{12{1'b0}}};
    rf_ra0 = inst[9 : 5];           //ra0 = rj
    rf_ra1 = inst[14:10];           //ra1 = rk
```

```
        rf_wa  = inst[4 : 0];           //wa  = rd
        rf_we  = 1'b1;
        alu_src0_sel = 1'b1;
        alu_src1_sel = 1'b1;
    end

    else if(inst[31:25] == 7'b000_1110)begin//pcaddu12i rd,si20
        alu_op = 5'b10111;
        imm = {inst[24:5],{12{1'b0}}};
        rf_ra0 = inst[9 : 5];           //ra0 = rj
        rf_ra1 = inst[14:10];           //ra1 = rk
        rf_wa  = inst[4 : 0];           //wa  = rd
        rf_we  = 1'b1;
        alu_src0_sel = 1'b0;
        alu_src1_sel = 1'b1;
    end
    else begin//此处为了避免锁存器进行异常情况处理
        alu_op = 5'b0;
        imm = 32'b0;
        rf_ra0 = 5'b0;
        rf_ra1 = 5'b0;
        rf_wa = 5'b0;
        rf_we = 1'b0;
        alu_src0_sel = 1'b1;
        alu_src1_sel = 1'b0;
    end
end
```

CPU设计

```
//PC instantiation
PC my_pc (
    .clk    (clk        ),
    .rst    (rst        ),
    .en     (global_en  ),    // 当 global_en 为高电平时，PC 才会更新，CPU 才会执行指令。
    .npc    (cur_npc    ),
    .pc     (cur_pc     )
);

assign imem_raddr = cur_pc;

wire    [ 4 : 0]    alu_op;
wire    [ 31: 0]    imm;
wire    [ 4 : 0]    rf_ra0;
wire    [ 4 : 0]    rf_ra1;
wire    [ 4 : 0]    rf_wa;
wire    [ 0 : 0]    rf_we;
wire    [ 31: 0]    rf_wd;
wire    [ 0 : 0]    alu_src0_sel;
wire    [ 0 : 0]    alu_src1_sel;

wire    [ 31: 0]    rf_rd0,rf_rd1;
wire    [ 31: 0]    alu_res;
wire    [ 31: 0]    alu_src0,alu_src1;

DECODE deco(
    .inst(imem_rdata),
    .alu_op(alu_op),
    .imm(imm),
    .rf_ra0(rf_ra0),
    .rf_ra1(rf_ra1),
    .rf_wa(rf_wa),
    .rf_we(rf_we),
    .alu_src0_sel(alu_src0_sel),
    .alu_src1_sel(alu_src1_sel)
);

assign cur_inst = imem_rdata;

REG_FILE reg_f(
    .clk(clk),
    .rf_ra0(rf_ra0),
    .rf_ra1(rf_ra1),
    .rf_wa(rf_wa),
    .rf_we(rf_we),
    .rf_wd(rf_wd),
    .rf_rd0(rf_rd0),
    .rf_rd1(rf_rd1),
    .dbg_reg_ra(debug_reg_ra),
    .dbg_reg_rd(debug_reg_rd)
);
//以下两个选择器选择源操作数的来源
MUX #(.WIDTH(32)) mux0 (//res = sel ? src1 : src0;
    .src0(cur_pc),
    .src1(rf_rd0),
    .sel(alu_src0_sel),
    .res(alu_src0)
);

MUX #(.WIDTH(32)) mux1 (//res = sel ? src1 : src0;
    .src0(rf_rd1),
    .src1(imm),
    .sel(alu_src1_sel),
    .res(alu_src1)
);

ALU alu(
    .alu_src0(alu_src0),
    .alu_src1(alu_src1),
    .alu_op(alu_op),
    .alu_res(rf_wd)
);
```

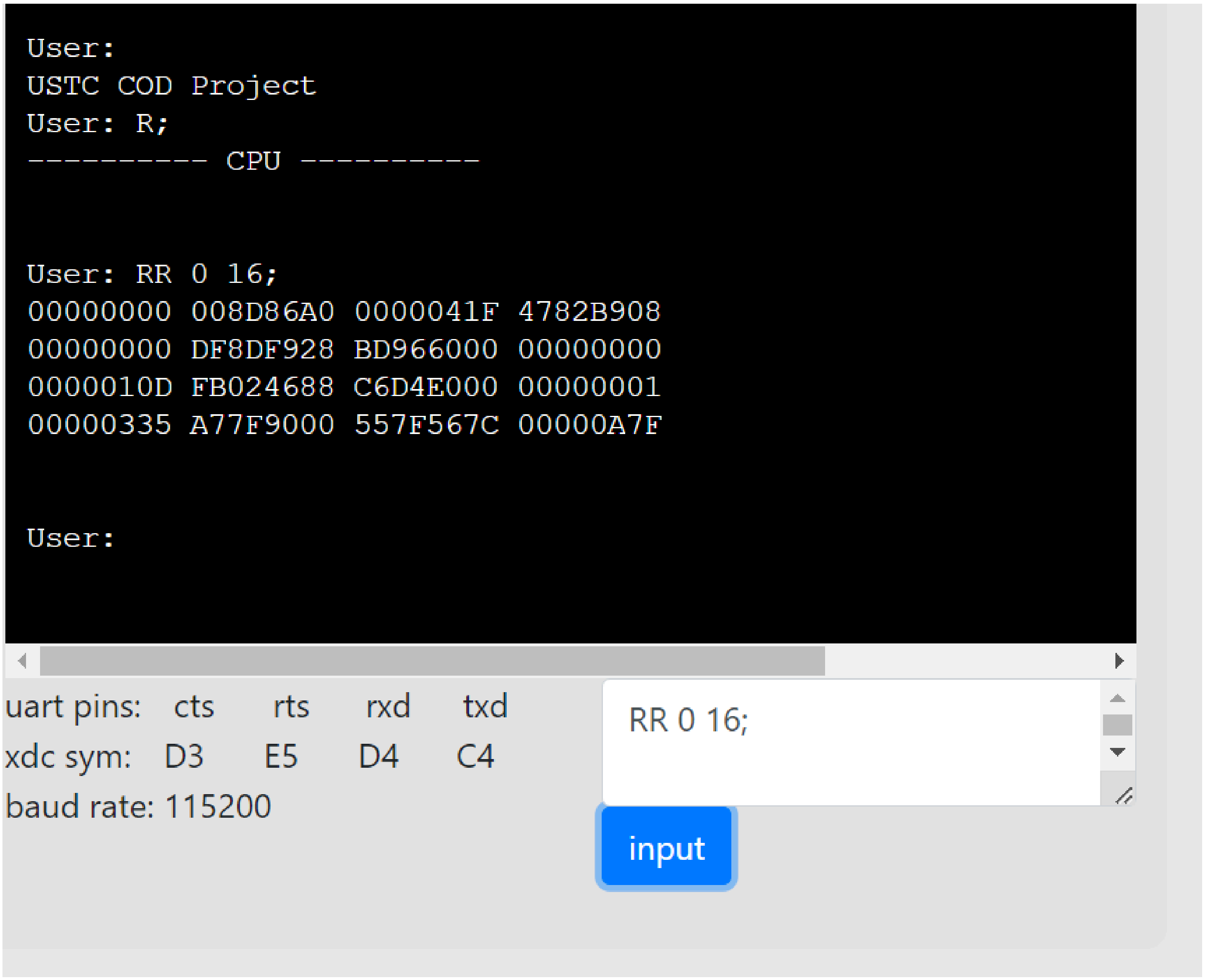
仿真结果



```
CMake Error: The current CMakeCache.txt directory /home/ubuntu/LA32R-Simula
Framework/build/CMakeCache.txt is different than the directory /home/ubuntu
top/LA32R-Simulation-Framework/build where CMakeCache.txt was created. This
result in binaries being created in the wrong place. If you are not sure, r
 the CMakeCache.txt
[  4%] Generating ../../rtl_top/vtop/VTop_copy.cmake
[ 91%] Built target RTL_TOP
[100%] Built target sim
Time:   4308        Instruction Count:  2153
SIMULATION END.
ubuntu@VM7667-juewang:~/LA32R-Simulation-Framework$ ./q.sh
Time:   4308        Instruction Count:  2153
SIMULATION END.
Dumping waveform.
Time:   4308        Instruction Count:  2153
SIMULATION END.
ubuntu@VM7667-juewang:~/LA32R-Simulation-Framework$ ./qq.sh
Time:   4308        Instruction Count:  2153
HIT GOOD TRAP.
SIMULATION END.
Dumping waveform.
Time:   4308        Instruction Count:  2153
SIMULATION END.
ubuntu@VM7667-juewang:~/LA32R-Simulation-Framework$ 
```

在龙芯仿真框架下进行仿真验证，结果正确符合预期。

上板结果



```
User:
USTC COD Project
User: R;
---------- CPU ----------


User: RR 0 16;
00000000 008D86A0 0000041F 4782B908
00000000 DF8DF928 BD966000 00000000
0000010D FB024688 C6D4E000 00000001
00000335 A77F9000 557F567C 00000A7F


User:
```

uart pins: cts rts rxd txd
xdc sym: D3 E5 D4 C4
baud rate: 115200

RR 0 16;

上板结果前16个寄存器的值与 LARS 运行结果一致。

总结

1. 本次实验的 CPU 中，哪些模块用到了时钟信号？
   - pc寄存器，寄存器堆，cpu模块用到时钟信号。
2. 请分别给出一条指令，以符合下面的描述：
   - alu_src0 选择 pc： b指令无条件跳转
   - alu_src0 选择 rf_rd0； add指令
   - alu_src1 选择 rf_rd1； sub指令
   - alu_src1 选择 imm； addi指令

3. 请指出本次实验的 CPU 中可能的关键路径；如果这条路径的延迟大于一个时钟周期，可能会带来什么影响？
    - 关键路径从 `pc` 计数开始，随后进入指令存储器取指，然后进行译码，进入寄存器堆取出操作数，经过多路选择器后进入 `alu` 计算，计算结果通过选择器写回寄存器堆，这个是一个可能的关键路径。
    - 如果延迟大于一个周期，后续指令可能会难以取出**准确的寄存器堆中需要的但是被当前指令修改过的**寄存器值，导致类似流水线的数据冲突和结构冒险。