

Lab 3 report

PB22051022王嘉宁

实验目的与内容

搭建单周期简单CPU

逻辑设计

任务 2：搭建 CPU 关键代码段

CPU连接

```
`define    HALT_INST    32'h8000_0000
module CPU (//变量声明省略
);
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;
ADD4 adder(
    .a        (cur_pc    ),
    .b        (32'd4     ),
    .c        (cur_npc   )
);

assign cur_inst = imem_rdata;
DECODE decode(
    .inst(cur_inst),
    .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),
);
REGFILE regfile(
    .clk(clk),
    .rf_ra0(rf_ra0),
    .rf_ra1(rf_ra1),
```

```

        .dbg_reg_ra(debug_reg_ra),
        .rf_wa(rf_wa),
        .rf_we(rf_we),
        .rf_wd(rf_wd),
        .rf_rd0(rf_rd0),
        .rf_rd1(rf_rd1),
        .dbg_reg_rd(debug_reg_rd)
    );
    MUX mux0(
        .src0(rf_rd0),
        .src1(cur_pc),
        .sel(alu_src0_sel),
        .res(alu_src0)
    );
    MUX mux1(
        .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)
    );

endmodule

```

PC

```

module PC(//.....
);

    always @(posedge clk) begin
        if(rst)      pc <= 32'h1c00_0000;
        else begin
            if(en)    pc <= npc;
            else      pc <= 32'h1c00_0000;
        end
    end
endmodule

```

DECODE

```

module DECODE(//.....
);
always @(*) begin
    if(inst[31:21] == 11'h0)begin
        case(inst[20:15])
            6'b100000:begin rf_wa = inst[4:0]; rf_ra0 = inst[9:5]; rf_ra1 =
inst[14:10]; rf_we = 1; alu_op = 5'b00000; alu_src0_sel = 0; alu_src1_sel = 0; imm
= 0; end
            6'b100010:begin rf_wa = inst[4:0]; rf_ra0 = inst[9:5]; rf_ra1 =
inst[14:10]; rf_we = 1; alu_op = 5'b00010; alu_src0_sel = 0; alu_src1_sel = 0; imm
= 0; end
            6'b100100:begin rf_wa = inst[4:0]; rf_ra0 = inst[9:5]; rf_ra1 =
inst[14:10]; rf_we = 1; alu_op = 5'b00100; alu_src0_sel = 0; alu_src1_sel = 0; imm
= 0; end
            6'b100101:begin rf_wa = inst[4:0]; rf_ra0 = inst[9:5]; rf_ra1 =
inst[14:10]; rf_we = 1; alu_op = 5'b00101; alu_src0_sel = 0; alu_src1_sel = 0; imm
= 0; end
            6'b101001:begin rf_wa = inst[4:0]; rf_ra0 = inst[9:5]; rf_ra1 =
inst[14:10]; rf_we = 1; alu_op = 5'b01001; alu_src0_sel = 0; alu_src1_sel = 0; imm
= 0; end
            6'b101010:begin rf_wa = inst[4:0]; rf_ra0 = inst[9:5]; rf_ra1 =
inst[14:10]; rf_we = 1; alu_op = 5'b01010; alu_src0_sel = 0; alu_src1_sel = 0; imm
= 0; end
            6'b101011:begin rf_wa = inst[4:0]; rf_ra0 = inst[9:5]; rf_ra1 =
inst[14:10]; rf_we = 1; alu_op = 5'b01011; alu_src0_sel = 0; alu_src1_sel = 0; imm
= 0; end
            6'b101110:begin rf_wa = inst[4:0]; rf_ra0 = inst[9:5]; rf_ra1 =
inst[14:10]; rf_we = 1; alu_op = 5'b01110; alu_src0_sel = 0; alu_src1_sel = 0; imm
= 0; end
            6'b101111:begin rf_wa = inst[4:0]; rf_ra0 = inst[9:5]; rf_ra1 =
inst[14:10]; rf_we = 1; alu_op = 5'b01111; alu_src0_sel = 0; alu_src1_sel = 0; imm
= 0; end
            6'b110000:begin rf_wa = inst[4:0]; rf_ra0 = inst[9:5]; rf_ra1 =
inst[14:10]; rf_we = 1; alu_op = 5'b10000; alu_src0_sel = 0; alu_src1_sel = 0; imm
= 0; end
            default:begin rf_wa = 0; rf_ra0 = 0; rf_ra1 = 0; rf_we = 0; alu_op =
0; alu_src0_sel = 0; alu_src1_sel = 0; imm = 0;end
        endcase
    end
    else if(inst[31:23] == 9'h0)begin
        case(inst[22:18])
            5'b10000:begin rf_wa = inst[4:0]; rf_ra0 = inst[9:5]; rf_ra1 = 0;
rf_we = 1; alu_op = 5'b01110; alu_src0_sel = 0; alu_src1_sel = 1; imm =
{27'b0,inst[14:10]}; end
            5'b10001:begin rf_wa = inst[4:0]; rf_ra0 = inst[9:5]; rf_ra1 = 0;
rf_we = 1; alu_op = 5'b01111; alu_src0_sel = 0; alu_src1_sel = 1; imm =
{27'b0,inst[14:10]}; end
            5'b10010:begin rf_wa = inst[4:0]; rf_ra0 = inst[9:5]; rf_ra1 = 0;
rf_we = 1; alu_op = 5'b10000; alu_src0_sel = 0; alu_src1_sel = 1; imm =
{27'b0,inst[14:10]}; end

```

```

        default:begin rf_wa = 0; rf_ra0 = 0; rf_ra1 = 0; rf_we = 0; alu_op =
0; alu_src0_sel = 0; alu_src1_sel = 0; imm = 0;end
    endcase
end
else if(inst[31:26] == 6'h0)begin
    case(inst[25:22])
        4'b1010:begin rf_wa = inst[4:0]; rf_ra0 = inst[9:5]; rf_ra1 = 0; rf_we
= 1; alu_op = 5'b00000; alu_src0_sel = 0; alu_src1_sel = 1; imm =
{{20{inst[21]}},inst[21:10]}; end
        4'b1000:begin rf_wa = inst[4:0]; rf_ra0 = inst[9:5]; rf_ra1 = 0; rf_we
= 1; alu_op = 5'b00100; alu_src0_sel = 0; alu_src1_sel = 1; imm =
{{20{inst[21]}},inst[21:10]}; end
        4'b1001:begin rf_wa = inst[4:0]; rf_ra0 = inst[9:5]; rf_ra1 = 0; rf_we
= 1; alu_op = 5'b00101; alu_src0_sel = 0; alu_src1_sel = 1; imm =
{{20{inst[21]}},inst[21:10]}; end
        4'b1101:begin rf_wa = inst[4:0]; rf_ra0 = inst[9:5]; rf_ra1 = 0; rf_we
= 1; alu_op = 5'b01001; alu_src0_sel = 0; alu_src1_sel = 1; imm =
{20'b0,inst[21:10]}; end
        4'b1110:begin rf_wa = inst[4:0]; rf_ra0 = inst[9:5]; rf_ra1 = 0; rf_we
= 1; alu_op = 5'b01010; alu_src0_sel = 0; alu_src1_sel = 1; imm =
{20'b0,inst[21:10]}; end
        4'b1111:begin rf_wa = inst[4:0]; rf_ra0 = inst[9:5]; rf_ra1 = 0; rf_we
= 1; alu_op = 5'b01011; alu_src0_sel = 0; alu_src1_sel = 1; imm =
{20'b0,inst[21:10]}; end
        default:begin rf_wa = 0; rf_ra0 = 0; rf_ra1 = 0; rf_we = 0; alu_op =
0; alu_src0_sel = 0; alu_src1_sel = 0; imm = 0;end
    endcase
end
else if(inst[31:29] == 3'h0)begin
    case(inst[28:25])
        4'b1010:begin rf_wa = inst[4:0]; rf_ra0 = 0; rf_ra1 = 0; rf_we = 1;
alu_op = 5'b10010; alu_src0_sel = 0; alu_src1_sel = 1; imm =
{inst[24:5],12'b0};end
        4'b1110:begin rf_wa = inst[4:0]; rf_ra0 = 0; rf_ra1 = 0; rf_we = 1;
alu_op = 5'b00000; alu_src0_sel = 1; alu_src1_sel = 1; imm =
{inst[24:5],12'b0};end
        default:begin rf_wa = 0; rf_ra0 = 0; rf_ra1 = 0; rf_we = 0; alu_op =
0; alu_src0_sel = 0; alu_src1_sel = 0; imm = 0;end
    endcase
end
else begin
    rf_wa = 0; rf_ra0 = 0; rf_ra1 = 0; rf_we = 0; alu_op = 0; alu_src0_sel =
0; alu_src1_sel = 0; imm = 0;
end
end
endmodule

```

测试结果与分析

任务 2 : 搭建 CPU

uart show>

FPGAOL UART xterm.js 1.1

USTC COD Project

User: R;

----- CPU -----

```
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;

input

任务三

1.本次实验的 CPU 中, 哪些模块用到了时钟信号?

PC和寄存器模块。

2.请分别给出一条指令, 以符合下面的描述:

alu_src0 选择 pc; pcaddu

alu_src0 选择 rf_rd0 ; add.w

alu_src1 选择 rf_rd1 ; addi.w
alu_src1 选择 imm ; and

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？

PC-IM-DECODER-REGFILE-MUX-ALU-REGFILE
若关键路径大于一个周期可能导致时序错误。

总结

学会打简单周期简单CPU，学会使用仿真框架进行仿真测试。