

实验 6：完整流水线 CPU

张子康 PB22020660

2024 年 5 月 11 日

1 实验目的与内容

1.1 实验目的

将结合前递模块与段间寄存器控制模块，得到一个能正常运行的完整流水线 CPU。

1.2 实验内容

1.2.1 任务 1：前递模块

根据实验文档的内容，根据传入的信号与优先级的判断，正确进行前递模块的设计。

1.2.2 任务 2：加入前递的流水线

将前递模块正确接入 CPU，形成加入前递的流水线，并通过对应仿真测试。

1.2.3 任务 3：段间寄存器控制模块

根据实验文档的内容，根据传入的信号，正确根据输入信号进行段间寄存器的控制模块设计。

1.2.4 任务 4：完整流水线 CPU

将段间寄存器控制模块正确接入 CPU，形成最终的流水线 CPU，并通过仿真、上板测试。

2 逻辑设计

2.1 任务 1：前递模块

```
1 module FORWARDING (  
2     input [0:0] rf_we_mem,
```

```
3     input [0:0] rf_we_wb,
4     input [4:0] rf_wa_mem,
5     input [4:0] rf_wa_wb,
6     input [31:0] rf_wd_mem,
7     input [31:0] rf_wd_wb,
8     input [4:0] rf_ra0_ex,
9     input [4:0] rf_ra1_ex,
10    output [0:0] rf_rd0_fe,
11    output [0:0] rf_rd1_fe,
12    output [31:0] rf_rd0_fd,
13    output [31:0] rf_rd1_fd
14 );
15 reg [31:0] rd0_fd,rd1_fd;
16 reg [0:0] rd0_fe,rd1_fe;
17 initial begin
18     rd0_fd=0;
19     rd0_fe=0;
20     rd1_fd=0;
21     rd1_fe=0;
22 end
23
24 assign rf_rd0_fe=rd0_fe;
25 assign rf_rd1_fe=rd1_fe;
26 assign rf_rd0_fd=rd0_fd;
27 assign rf_rd1_fd=rd1_fd;
28
29 always @(*) begin
30     // 默认不前递
31     rd0_fd=0;
32     rd1_fd=0;
33     rd0_fe=0;
34     rd1_fe=0;
35     // 如果发现写入的地址与读取的地址相同就前递
```

```

36         if (rf_we_wb && rf_wa_wb != 5'b00000 && rf_wa_wb ==
           rf_ra0_ex) begin
37             rd0_fd = rf_wd_wb;
38             rd0_fe = 1;
39         end
40         if (rf_we_wb && rf_wa_wb != 5'b00000 && rf_wa_wb ==
           rf_ra1_ex) begin
41             rd1_fd = rf_wd_wb;
42             rd1_fe = 1;
43         end
44         if (rf_we_mem && rf_wa_mem != 5'b00000 && rf_wa_mem ==
           rf_ra0_ex) begin
45             rd0_fd = rf_wd_mem;
46             rd0_fe = 1;
47         end
48         if (rf_we_mem && rf_wa_mem != 5'b00000 && rf_wa_mem ==
           rf_ra1_ex) begin
49             rd1_fd = rf_wd_mem;
50             rd1_fe = 1;
51         end
52     end
53 endmodule

```

2.2 任务 2：加入前递的流水线

```

1 // rf_rd0_ex_和rf_rd1_ex_为前递选择后的数据
2 FORWARDING forwarding(
3     .rf_we_mem(rf_we_mem),
4     .rf_we_wb(rf_we_wb),
5     .rf_wa_mem(rf_wa_mem),
6     .rf_wa_wb(rf_wa_wb),
7     .rf_wd_mem(alu_res_mem),
8     .rf_wd_wb(alu_res_wb),

```

```
9         .rf_ra0_ex(rf_ra0_ex),
10         .rf_ra1_ex(rf_ra1_ex),
11         .rf_rd0_fe(rf_rd0_fe),
12         .rf_rd1_fe(rf_rd1_fe),
13         .rf_rd0_fd(rf_rd0_fd),
14         .rf_rd1_fd(rf_rd1_fd)
15     );
16
17     MUX1 mux3(
18         .src1(rf_rd0_fd),
19         .src0(rf_rd0_ex),
20         .sel(rf_rd0_fe),
21         .res(rf_rd0_ex_)
22     );
23
24     MUX1 mux4(
25         .src1(rf_rd1_fd),
26         .src0(rf_rd1_ex),
27         .sel(rf_rd1_fe),
28         .res(rf_rd1_ex_)
29     );
```

将上述代码加入 CPU 模块，同时作出如下修改：

```
1     MUX1 mux1(
2         .src0(rf_rd0_ex_),
3         .src1(pc_ex),
4         .sel(alu_src0_sel_ex),
5         .res(alu_src0_ex)
6     );
7
8     MUX1 mux2(
9         .src0(rf_rd1_ex_),
10        .src1(imm_ex),
11        .sel(alu_src1_sel_ex),
```

```

12         .res(alu_src1_ex)
13     );
14
15     EX_MEM ex_mem(
16         .clk(clk),
17         .rst(rst),
18         .flush(flush_ex_mem),
19         .stall(stall),
20         .en(en),
21         //PC
22         .pc_add4_in(pcadd4_ex),
23         .pc_in(pc_ex),
24         .pc_add4_out(pcadd4_mem),
25         .pc_out(pc_mem),
26         //INST
27         .inst_in(inst_ex),
28         .inst_out(inst_mem),
29         //DECODER
30         .alu_op_in(alu_op_ex),
31         .dmem_access_in(dmem_access_ex),
32         .imm_in(imm_ex),
33         .rf_ra0_in(rf_ra0_ex),
34         .rf_ra1_in(rf_ra1_ex),
35         .rf_wa_in(rf_wa_ex),
36         .rf_we_in(rf_we_ex),
37         .rf_wd_sel_in(rf_wd_sel_ex),
38         .alu_src0_sel_in(alu_src0_sel_ex),
39         .alu_src1_sel_in(alu_src1_sel_ex),
40         .br_type_in(br_type_ex),
41         .dmem_we_in(dmem_we_ex),
42         .alu_op_out(alu_op_mem),
43         .dmem_access_out(dmem_access_mem),
44         .imm_out(imm_mem),

```

```

45     .rf_ra0_out(rf_ra0_mem),
46     .rf_ra1_out(rf_ra1_mem),
47     .rf_wa_out(rf_wa_mem),
48     .rf_we_out(rf_we_mem),
49     .rf_wd_sel_out(rf_wd_sel_mem),
50     .alu_src0_sel_out(alu_src0_sel_mem),
51     .alu_src1_sel_out(alu_src1_sel_mem),
52     .br_type_out(br_type_mem),
53     .dmem_we_out(dmem_we_mem),
54     //REG_FILE
55     .rf_rd0_in(rf_rd0_ex_),
56     .rf_rd1_in(rf_rd1_ex_),
57     .rf_rd0_out(rf_rd0_mem),
58     .rf_rd1_out(rf_rd1_mem),
59     //MUX1
60     .alu_src0_in(alu_src0_ex),
61     .alu_src1_in(alu_src1_ex),
62     .alu_src0_out(alu_src0_mem),
63     .alu_src1_out(alu_src1_mem),
64     //NPC
65     .npc_in(npc_ex),
66     .npc_out(npc_mem),
67     //BRANCH
68     .npc_sel_in(npc_sel_ex),
69     .npc_sel_out(npc_sel_mem),
70     //ALU
71     .alu_res_in(alu_res_ex),
72     .alu_res_out(alu_res_mem),
73     //SLU
74     .dmem_rd_out_in(0),
75     .dmem_wdata_mem_in(0),
76     .dmem_rd_out_out(),
77     .dmem_wdata_mem_out(),

```

```
78         //DM
79         .dmem_rdata_mem_in(0),
80         .dmem_rdata_mem_out(),
81         .commit_in(commit_ex),
82         .commit_out(commit_mem)
83     );
```

2.3 任务 3：段间寄存器控制模块

```
1  `define DMEM_RDATA 2'b10
2  module SEG_CTRL (
3      input [0:0] rf_we_ex,
4      input [1:0] rf_wd_sel_ex,
5      input [4:0] rf_wa_ex,
6      input [4:0] rf_ra0_id,
7      input [4:0] rf_ra1_id,
8      input [1:0] npc_sel_ex,
9      output reg [0:0] stall_pc,
10     output reg [0:0] stall_if_id,
11     output reg [0:0] flush_if_id,
12     output reg [0:0] flush_id_ex
13 );
14     always @(*) begin
15         // 默认不阻塞或冲刷流水线
16         stall_pc=0;
17         stall_if_id=0;
18         flush_id_ex=0;
19         flush_if_id=0;
20         // 对于load-use, 若写入地址和读取地址相同
21         // 则阻塞流水线2周期
22         if(rf_we_ex && rf_wd_sel_ex==`DMEM_RDATA && (
            rf_wa_ex==rf_ra0_id || rf_wa_ex==rf_ra1_id))
            begin
```



```

23         flush_id_ex=1'b1;
24         stall_pc=1'b1;
25         stall_if_id=1'b1;
26     end
27     // 对于跳转指令，直接冲刷流水线
28     if(npc_sel_ex!=2'b00) begin
29         flush_id_ex=1'b1;
30         flush_if_id=1'b1;
31     end
32 end
33 endmodule

```

2.4 任务 4：完整流水线 CPU

```

1  `include "../include/config.v"
2
3  module CPU (input [0 : 0] clk,
4              input [0 : 0] rst,
5              input [0 : 0] global_en,
6              output [31 : 0] imem_raddr,
7              input [31 : 0] imem_rdata,
8              input [31 : 0] dmem_rdata,        // Unused
9              output [0 : 0] dmem_we,          // Unused
10             output [31 : 0] dmem_addr,        // Unused
11             output [31 : 0] dmem_wdata,       // Unused
12             output [0 : 0] commit,
13             output [31 : 0] commit_pc,
14             output [31 : 0] commit_inst,
15             output [0 : 0] commit_halt,
16             output [0 : 0] commit_reg_we,
17             output [4 : 0] commit_reg_wa,
18             output [31 : 0] commit_reg_wd,
19             output [0 : 0] commit_dmem_we,

```

```
20         output [31 : 0] commit_dmem_wa,
21         output [31 : 0] commit_dmem_wd,
22         input [4 : 0] debug_reg_ra,
23         output [31 : 0] debug_reg_rd);
24
25
26     // TODO
27     wire [0:0] commit_if;
28     wire [31:0] pc_if;
29     wire [31:0] pcadd4_if;
30     wire [31:0] inst_if;
31
32     wire [0:0] commit_id;
33     wire [31:0] pc_id;
34     wire [31:0] pcadd4_id;
35     wire [31:0] inst_id;
36     wire [4:0] alu_op_id;
37     wire [3:0] dmem_access_id;
38     wire [31:0] imm_id;
39     wire [4:0] rf_ra0_id;
40     wire [4:0] rf_ra1_id;
41     wire [5:0] rf_wa_id;
42     wire [0:0] rf_we_id;
43     wire [1:0] rf_wd_sel_id;
44     wire [0:0] alu_src0_sel_id;
45     wire [0:0] alu_src1_sel_id;
46     wire [3:0] br_type_id;
47     wire [0:0] dmem_we_id;
48     wire [31:0] rf_rd0_id;
49     wire [31:0] rf_rd1_id;
50
51     wire [0:0] commit_ex;
52     wire [31:0] pc_ex;
```

```
53     wire [31:0] pcadd4_ex;
54     wire [31:0] inst_ex;
55     wire [4:0] alu_op_ex;
56     wire [3:0] dmem_access_ex;
57     wire [31:0] imm_ex;
58     wire [4:0] rf_ra0_ex;
59     wire [4:0] rf_ra1_ex;
60     wire [5:0] rf_wa_ex;
61     wire [0:0] rf_we_ex;
62     wire [1:0] rf_wd_sel_ex;
63     wire [0:0] alu_src0_sel_ex;
64     wire [0:0] alu_src1_sel_ex;
65     wire [3:0] br_type_ex;
66     wire [0:0] dmem_we_ex;
67     wire [31:0] rf_rd0_ex;
68     wire [31:0] rf_rd1_ex;
69     wire [31:0] npc_ex;
70     wire [31:0] pc_j_ex;
71     wire [31:0] alu_src0_ex;
72     wire [31:0] alu_src1_ex;
73     wire [31:0] alu_res_ex;
74     wire [1:0] npc_sel_ex;
75
76     wire [0:0] commit_mem;
77     wire [31:0] pc_mem;
78     wire [31:0] pcadd4_mem;
79     wire [31:0] inst_mem;
80     wire [4:0] alu_op_mem;
81     wire [3:0] dmem_access_mem;
82     wire [31:0] imm_mem;
83     wire [4:0] rf_ra0_mem;
84     wire [4:0] rf_ra1_mem;
85     wire [5:0] rf_wa_mem;
```

```
86     wire [0:0] rf_we_mem;
87     wire [1:0] rf_wd_sel_mem;
88     wire [0:0] alu_src0_sel_mem;
89     wire [0:0] alu_src1_sel_mem;
90     wire [3:0] br_type_mem;
91     wire [0:0] dmem_we_mem;
92     wire [31:0] rf_rd0_mem;
93     wire [31:0] rf_rd1_mem;
94     wire [31:0] npc_mem;
95     wire [31:0] pc_j_mem;
96     wire [31:0] alu_src0_mem;
97     wire [31:0] alu_src1_mem;
98     wire [31:0] alu_res_mem;
99     wire [31:0] dmem_rd_out_mem;
100    wire [31:0] dmem_wdata_mem;
101    wire [31:0] dmem_rdata_mem;
102    wire [1:0] npc_sel_mem;
103
104    wire [0:0] commit_wb;
105    wire [31:0] pc_wb;
106    wire [31:0] pcadd4_wb;
107    wire [31:0] inst_wb;
108    wire [4:0] alu_op_wb;
109    wire [3:0] dmem_access_wb;
110    wire [31:0] imm_wb;
111    wire [4:0] rf_ra0_wb;
112    wire [4:0] rf_ra1_wb;
113    wire [5:0] rf_wa_wb;
114    wire [0:0] rf_we_wb;
115    wire [1:0] rf_wd_sel_wb;
116    wire [0:0] alu_src0_sel_wb;
117    wire [0:0] alu_src1_sel_wb;
118    wire [3:0] br_type_wb;
```

```

119     wire [0:0] dmem_we_wb;
120     wire [31:0] rf_rd0_wb;
121     wire [31:0] rf_rd1_wb;
122     wire [31:0] npc_wb;
123     wire [31:0] pc_j_wb;
124     wire [31:0] alu_src0_wb;
125     wire [31:0] alu_src1_wb;
126     wire [31:0] alu_res_wb;
127     wire [31:0] dmem_rd_out_wb;
128     wire [31:0] dmem_wdata_wb;
129     wire [31:0] dmem_rdata_wb;
130     wire [31:0] rf_wd_wb;
131     wire [1:0] npc_sel_wb;
132
133     wire [0:0] rf_rd0_fe,rf_rd1_fe;
134     wire [31:0] rf_rd0_fd,rf_rd1_fd;
135     wire [31:0] rf_rd0_ex_,rf_rd1_ex_;
136
137     wire flush,stall,stall_pc,stall_if_id,en;
138     wire flush_if_id,flush_id_ex,flush_ex_mem,flush_mem_wb;
139
140     assign commit_if = 1;
141     assign stall = 0;
142     assign en = global_en;
143     assign global_en = !(inst_wb == 32'H00100073);
144     assign imem_raddr = (pc_if-32'h00400000)/'d4;
145     assign inst_if = imem_rdata;
146     assign pc_j_ex = alu_res_ex&~1;
147     assign dmem_wd_in = rf_rd1_mem;
148     assign dmem_we = dmem_we_mem;
149     assign dmem_addr = (alu_res_mem-32'h10010000)/'d4;
150     assign dmem_wdata = dmem_wdata_mem;
151     assign dmem_rdata_mem = dmem_rdata;

```

```

152    //assign flush_if_id=inst_if==32'h00000013 && inst_id
        ==32'h00000013;
153    //assign flush_id_ex=inst_if==32'h00000013 && inst_id
        ==32'h00000013;
154    assign flush_ex_mem=0;
155    assign flush_mem_wb=0;
156
157    // 前递模块
158    FORWARDING forwarding(
159        .rf_we_mem(rf_we_mem),
160        .rf_we_wb(rf_we_wb),
161        .rf_wa_mem(rf_wa_mem),
162        .rf_wa_wb(rf_wa_wb),
163        .rf_wd_mem(alu_res_mem),
164        .rf_wd_wb(alu_res_wb),
165        .rf_ra0_ex(rf_ra0_ex),
166        .rf_ra1_ex(rf_ra1_ex),
167        .rf_rd0_fe(rf_rd0_fe),
168        .rf_rd1_fe(rf_rd1_fe),
169        .rf_rd0_fd(rf_rd0_fd),
170        .rf_rd1_fd(rf_rd1_fd)
171    );
172
173    MUX1 mux3(
174        .src1(rf_rd0_fd),
175        .src0(rf_rd0_ex),
176        .sel(rf_rd0_fe),
177        .res(rf_rd0_ex_)
178    );
179
180    MUX1 mux4(
181        .src1(rf_rd1_fd),
182        .src0(rf_rd1_ex),

```

```

183         .sel(rf_rd1_fe),
184         .res(rf_rd1_ex_)
185     );
186
187     // 段间寄存器控制器
188     SEG_CTRL seg_ctrl(
189         .rf_we_ex(rf_we_ex),
190         .rf_wd_sel_ex(rf_wd_sel_ex),
191         .rf_wa_ex(rf_wa_ex),
192         .rf_ra0_id(rf_ra0_id),
193         .rf_ra1_id(rf_ra1_id),
194         .npc_sel_ex(npc_sel_ex),
195         .stall_pc(stall_pc),
196         .stall_if_id(stall_if_id),
197         .flush_if_id(flush_if_id),
198         .flush_id_ex(flush_id_ex)
199     );
200
201
202     PC_PLUS4 pc_plus(
203         .pc(pc_if),
204         .pc_plus4(pcadd4_if)
205     );
206
207     PC pc(
208         .clk      (clk),
209         .rst      (rst),
210         .en       (global_en),    // 当 global_en 为高电平
                                   时, PC 才会更新, CPU 才会执行指令。
211         .npc      (npc_ex),
212         .pc       (pc_if),
213         .stall    (stall_pc),
214         .flush    (flush)

```

```
215     );
216
217     IF_ID if_id(
218         .clk(clk),
219         .rst(rst),
220         .flush(flush_if_id),
221         .stall(stall_if_id),
222         .en(en),
223         //PC
224         .pc_add4_in(pcadd4_if),
225         .pc_in(pc_if),
226         .pc_add4_out(pcadd4_id),
227         .pc_out(pc_id),
228         //INST
229         .inst_in(inst_if),
230         .inst_out(inst_id),
231         //DECODER
232         .alu_op_in(0),
233         .dmem_access_in(0),
234         .imm_in(0),
235         .rf_ra0_in(0),
236         .rf_ra1_in(0),
237         .rf_wa_in(0),
238         .rf_we_in(0),
239         .rf_wd_sel_in(0),
240         .alu_src0_sel_in(0),
241         .alu_src1_sel_in(0),
242         .br_type_in(0),
243         .dmem_we_in(0),
244         .alu_op_out(),
245         .dmem_access_out(),
246         .imm_out(),
247         .rf_ra0_out(),
```



```
248         .rf_ra1_out(),
249         .rf_wa_out(),
250         .rf_we_out(),
251         .rf_wd_sel_out(),
252         .alu_src0_sel_out(),
253         .alu_src1_sel_out(),
254         .br_type_out(),
255         .dmem_we_out(),
256         //REG_FILE
257         .rf_rd0_in(0),
258         .rf_rd1_in(0),
259         .rf_rd0_out(),
260         .rf_rd1_out(),
261         //MUX1
262         .alu_src0_in(0),
263         .alu_src1_in(0),
264         .alu_src0_out(),
265         .alu_src1_out(),
266         //NPC
267         .npc_in(0),
268         .npc_out(),
269         //BRANCH
270         .npc_sel_in(0),
271         .npc_sel_out(),
272         //ALU
273         .alu_res_in(0),
274         .alu_res_out(),
275         //SLU
276         .dmem_rd_out_in(0),
277         .dmem_wdata_mem_in(0),
278         .dmem_rd_out_out(),
279         .dmem_wdata_mem_out(),
280         //DM
```

```

281         .dmem_rdata_mem_in(0),
282         .dmem_rdata_mem_out(),
283         .commit_in(commit_if),
284         .commit_out(commit_id)
285     );
286
287     DECODER decoder(
288         .inst(inst_id),
289         .alu_op(alu_op_id),
290         .imm(imm_id),
291         .rf_ra0(rf_ra0_id),
292         .rf_ra1(rf_ra1_id),
293         .rf_wa(rf_wa_id),
294         .rf_we(rf_we_id),
295         .alu_src0_sel(alu_src0_sel_id),
296         .alu_src1_sel(alu_src1_sel_id),
297         .dmem_access(dmem_access_id),
298         .rf_wd_sel(rf_wd_sel_id),
299         .br_type(br_type_id),
300         .dmem_we(dmem_we_id)
301     );
302
303     REG_FILE reg_file(
304         .clk(clk),
305         .rf_ra0(rf_ra0_id),
306         .rf_ra1(rf_ra1_id),
307         .rf_wa(rf_wa_wb),
308         .rf_we(rf_we_wb),
309         .rf_wd(rf_wd_wb),
310         .rf_rd0(rf_rd0_id),
311         .rf_rd1(rf_rd1_id),
312         .debug_reg_rd(debug_reg_rd),
313         .debug_reg_ra(debug_reg_ra)

```

```

314     );
315
316     ID_EX id_ex(
317         .clk(clk),
318         .rst(rst),
319         .flush(flush_id_ex),
320         .stall(stall),
321         .en(en),
322         //PC
323         .pc_add4_in(pcadd4_id),
324         .pc_in(pc_id),
325         .pc_add4_out(pcadd4_ex),
326         .pc_out(pc_ex),
327         //INST
328         .inst_in(inst_id),
329         .inst_out(inst_ex),
330         //DECODER
331         .alu_op_in(alu_op_id),
332         .dmem_access_in(dmem_access_id),
333         .imm_in(imm_id),
334         .rf_ra0_in(rf_ra0_id),
335         .rf_ra1_in(rf_ra1_id),
336         .rf_wa_in(rf_wa_id),
337         .rf_we_in(rf_we_id),
338         .rf_wd_sel_in(rf_wd_sel_id),
339         .alu_src0_sel_in(alu_src0_sel_id),
340         .alu_src1_sel_in(alu_src1_sel_id),
341         .br_type_in(br_type_id),
342         .dmem_we_in(dmem_we_id),
343         .alu_op_out(alu_op_ex),
344         .dmem_access_out(dmem_access_ex),
345         .imm_out(imm_ex),
346         .rf_ra0_out(rf_ra0_ex),

```

```

347         .rf_ra1_out(rf_ra1_ex),
348         .rf_wa_out(rf_wa_ex),
349         .rf_we_out(rf_we_ex),
350         .rf_wd_sel_out(rf_wd_sel_ex),
351         .alu_src0_sel_out(alu_src0_sel_ex),
352         .alu_src1_sel_out(alu_src1_sel_ex),
353         .br_type_out(br_type_ex),
354         .dmem_we_out(dmem_we_ex),
355         //REG_FILE
356         .rf_rd0_in(rf_rd0_id),
357         .rf_rd1_in(rf_rd1_id),
358         .rf_rd0_out(rf_rd0_ex),
359         .rf_rd1_out(rf_rd1_ex),
360         //MUX1
361         .alu_src0_in(0),
362         .alu_src1_in(0),
363         .alu_src0_out(),
364         .alu_src1_out(),
365         //NPC
366         .npc_in(0),
367         .npc_out(),
368         //BRANCH
369         .npc_sel_in(0),
370         .npc_sel_out(),
371         //ALU
372         .alu_res_in(0),
373         .alu_res_out(),
374         //SLU
375         .dmem_rd_out_in(0),
376         .dmem_wdata_mem_in(0),
377         .dmem_rd_out_out(),
378         .dmem_wdata_mem_out(),
379         //DM

```

```
380         .dmem_rdata_mem_in(0),
381         .dmem_rdata_mem_out(),
382         .commit_in(commit_id),
383         .commit_out(commit_ex)
384     );
385
386     MUX1 mux1(
387         .src0(rf_rd0_ex_),
388         .src1(pc_ex),
389         .sel(alu_src0_sel_ex),
390         .res(alu_src0_ex)
391     );
392
393     MUX1 mux2(
394         .src0(rf_rd1_ex_),
395         .src1(imm_ex),
396         .sel(alu_src1_sel_ex),
397         .res(alu_src1_ex)
398     );
399
400     ALU alu(
401         .alu_src0(alu_src0_ex),
402         .alu_src1(alu_src1_ex),
403         .alu_op(alu_op_ex),
404         .alu_res(alu_res_ex)
405     );
406
407     BRANCH branch(
408         .br_type(br_type_ex),
409         .br_src0(rf_rd0_ex_),
410         .br_src1(rf_rd1_ex_),
411         .npc_sel(npc_sel_ex)
412     );
```

```

413     NPC npc(
414         .pc_offset(alu_res_ex),
415         .pc_add4(pcadd4_if),
416         .pc_j(pc_j_ex),
417         .npc_sel(npc_sel_ex),
418         .npc(npc_ex)
419     );
420
421     EX_MEM ex_mem(
422         .clk(clk),
423         .rst(rst),
424         .flush(flush_ex_mem),
425         .stall(stall),
426         .en(en),
427         //PC
428         .pc_add4_in(pcadd4_ex),
429         .pc_in(pc_ex),
430         .pc_add4_out(pcadd4_mem),
431         .pc_out(pc_mem),
432         //INST
433         .inst_in(inst_ex),
434         .inst_out(inst_mem),
435         //DECODER
436         .alu_op_in(alu_op_ex),
437         .dmem_access_in(dmem_access_ex),
438         .imm_in(imm_ex),
439         .rf_ra0_in(rf_ra0_ex),
440         .rf_ra1_in(rf_ra1_ex),
441         .rf_wa_in(rf_wa_ex),
442         .rf_we_in(rf_we_ex),
443         .rf_wd_sel_in(rf_wd_sel_ex),
444         .alu_src0_sel_in(alu_src0_sel_ex),
445         .alu_src1_sel_in(alu_src1_sel_ex),

```

```
446         .br_type_in(br_type_ex),
447         .dmem_we_in(dmem_we_ex),
448         .alu_op_out(alu_op_mem),
449         .dmem_access_out(dmem_access_mem),
450         .imm_out(imm_mem),
451         .rf_ra0_out(rf_ra0_mem),
452         .rf_ra1_out(rf_ra1_mem),
453         .rf_wa_out(rf_wa_mem),
454         .rf_we_out(rf_we_mem),
455         .rf_wd_sel_out(rf_wd_sel_mem),
456         .alu_src0_sel_out(alu_src0_sel_mem),
457         .alu_src1_sel_out(alu_src1_sel_mem),
458         .br_type_out(br_type_mem),
459         .dmem_we_out(dmem_we_mem),
460         //REG_FILE
461         .rf_rd0_in(rf_rd0_ex_),
462         .rf_rd1_in(rf_rd1_ex_),
463         .rf_rd0_out(rf_rd0_mem),
464         .rf_rd1_out(rf_rd1_mem),
465         //MUX1
466         .alu_src0_in(alu_src0_ex),
467         .alu_src1_in(alu_src1_ex),
468         .alu_src0_out(alu_src0_mem),
469         .alu_src1_out(alu_src1_mem),
470         //NPC
471         .npc_in(npc_ex),
472         .npc_out(npc_mem),
473         //BRANCH
474         .npc_sel_in(npc_sel_ex),
475         .npc_sel_out(npc_sel_mem),
476         //ALU
477         .alu_res_in(alu_res_ex),
478         .alu_res_out(alu_res_mem),
```

```

479         //SLU
480         .dmem_rd_out_in(0),
481         .dmem_wdata_mem_in(0),
482         .dmem_rd_out_out(),
483         .dmem_wdata_mem_out(),
484         //DM
485         .dmem_rdata_mem_in(0),
486         .dmem_rdata_mem_out(),
487         .commit_in(commit_ex),
488         .commit_out(commit_mem)
489     );
490
491     SLU slu(
492         .addr(alu_res_mem),
493         .dmem_access(dmem_access_mem),
494         .rd_in(dmem_rdata_mem),
495         .wd_in(rf_rd1_mem),
496         .rd_out(dmem_rd_out_mem),
497         .wd_out(dmem_wdata_mem)
498     );
499
500     MEM_WB mem_wb(
501         .clk(clk),
502         .rst(rst),
503         .flush(flush_mem_wb),
504         .stall(stall),
505         .en(en),
506         //PC
507         .pc_add4_in(pcadd4_mem),
508         .pc_in(pc_mem),
509         .pc_add4_out(pcadd4_wb),
510         .pc_out(pc_wb),
511         //INST

```



```

512         .inst_in(inst_mem),
513         .inst_out(inst_wb),
514     //DECODER
515     .alu_op_in(alu_op_mem),
516     .dmem_access_in(dmem_access_mem),
517     .imm_in(imm_mem),
518     .rf_ra0_in(rf_ra0_mem),
519     .rf_ra1_in(rf_ra1_mem),
520     .rf_wa_in(rf_wa_mem),
521     .rf_we_in(rf_we_mem),
522     .rf_wd_sel_in(rf_wd_sel_mem),
523     .alu_src0_sel_in(alu_src0_sel_mem),
524     .alu_src1_sel_in(alu_src1_sel_mem),
525     .br_type_in(br_type_mem),
526     .dmem_we_in(dmem_we_mem),
527     .alu_op_out(alu_op_wb),
528     .dmem_access_out(dmem_access_wb),
529     .imm_out(imm_wb),
530     .rf_ra0_out(rf_ra0_wb),
531     .rf_ra1_out(rf_ra1_wb),
532     .rf_wa_out(rf_wa_wb),
533     .rf_we_out(rf_we_wb),
534     .rf_wd_sel_out(rf_wd_sel_wb),
535     .alu_src0_sel_out(alu_src0_sel_wb),
536     .alu_src1_sel_out(alu_src1_sel_wb),
537     .br_type_out(br_type_wb),
538     .dmem_we_out(dmem_we_wb),
539     //REG_FILE
540     .rf_rd0_in(rf_rd0_mem),
541     .rf_rd1_in(rf_rd1_mem),
542     .rf_rd0_out(rf_rd0_wb),
543     .rf_rd1_out(rf_rd1_wb),
544     //MUX1

```

```

545         .alu_src0_in(alu_src0_mem),
546         .alu_src1_in(alu_src1_mem),
547         .alu_src0_out(alu_src0_wb),
548         .alu_src1_out(alu_src1_wb),
549         //NPC
550         .npc_in(npc_mem),
551         .npc_out(npc_wb),
552         //BRANCH
553         .npc_sel_in(npc_sel_mem),
554         .npc_sel_out(npc_sel_wb),
555         //ALU
556         .alu_res_in(alu_res_mem),
557         .alu_res_out(alu_res_wb),
558         //SLU
559         .dmem_rd_out_in(dmem_rd_out_mem),
560         .dmem_wdata_mem_in(dmem_wdata_mem),
561         .dmem_rd_out_out(dmem_rd_out_wb),
562         .dmem_wdata_mem_out(dmem_wdata_wb),
563         //DM
564         .dmem_rdata_mem_in(dmem_rdata_mem),
565         .dmem_rdata_mem_out(dmem_rdata_wb),
566         .commit_in(commit_mem),
567         .commit_out(commit_wb)
568     );
569
570     MUX2 RF_MUX(
571         .src0(pcadd4_wb),
572         .src1(alu_res_wb),
573         .src2(dmem_rd_out_wb),
574         .src3(0),
575         .sel(rf_wd_sel_wb),
576         .res(rf_wd_wb)
577     );

```

```

578
579  /*
-----
    */
580  /*                                Commit
                                */
581  /*
-----
    */

582
583  // wire [0 : 0] commit_if      ;
584  // assign commit_if = 1'H1;

585
586  reg  [0 : 0]    commit_reg      ;
587  reg  [31 : 0]   commit_pc_reg   ;
588  reg  [31 : 0]   commit_inst_reg ;
589  reg  [0 : 0]    commit_halt_reg ;
590  reg  [0 : 0]    commit_reg_we_reg ;
591  reg  [4 : 0]    commit_reg_wa_reg ;
592  reg  [31 : 0]   commit_reg_wd_reg ;
593  reg  [0 : 0]    commit_dmem_we_reg ;
594  reg  [31 : 0]   commit_dmem_wa_reg ;
595  reg  [31 : 0]   commit_dmem_wd_reg ;

596
597  always @(posedge clk) begin
598      if (rst) begin
599          commit_reg      <= 1'H0;
600          commit_pc_reg   <= 32'H0;
601          commit_inst_reg <= 32'H0;
602          commit_halt_reg <= 1'H0;
603          commit_reg_we_reg <= 1'H0;
604          commit_reg_wa_reg <= 5'H0;
605          commit_reg_wd_reg <= 32'H0;

```

```
606         commit_dmem_we_reg <= 1'H0;
607         commit_dmem_wa_reg <= 32'H0;
608         commit_dmem_wd_reg <= 32'H0;
609     end
610     else if (global_en) begin
611         commit_reg          <= commit_wb;
612         commit_pc_reg       <= pc_wb;
613         commit_inst_reg     <= inst_wb;
614         commit_halt_reg     <= inst_wb == `HALT_INST;
615         commit_reg_we_reg   <= rf_we_wb;
616         commit_reg_wa_reg   <= rf_wa_wb;
617         commit_reg_wd_reg   <= rf_wd_wb;
618         commit_dmem_we_reg  <= dmem_we_wb;
619         commit_dmem_wa_reg  <= alu_res_wb;
620         commit_dmem_wd_reg  <= dmem_wdata_wb;
621     end
622 end
623
624 assign commit          = commit_reg;
625 assign commit_pc      = commit_pc_reg;
626 assign commit_inst    = commit_inst_reg;
627 assign commit_halt    = commit_halt_reg;
628 assign commit_reg_we  = commit_reg_we_reg;
629 assign commit_reg_wa  = commit_reg_wa_reg;
630 assign commit_reg_wd  = commit_reg_wd_reg;
631 assign commit_dmem_we = commit_dmem_we_reg;
632 assign commit_dmem_wa = commit_dmem_wa_reg;
633 assign commit_dmem_wd = commit_dmem_wd_reg;
634 endmodule
```

3 仿真结果与分析

3.1 Test1



图 1: 仿真结果

Name	Number	Value
zero	0	0x00000000
ra	1	0x00000001
sp	2	0x000006e3
gp	3	0xf4b2e614
tp	4	0x00000000
t0	5	0x00000000
t1	6	0x00000004
t2	7	0x7f77ffff
s0	8	0xdffcffd5
s1	9	0x60cbe000
a0	10	0x006ffeff
a1	11	0xf1165000
a2	12	0xf4b2e614
a3	13	0x7594d000
a4	14	0x00000000
a5	15	0x0a438000
a6	16	0x0a438000
a7	17	0x60cbe000
s2	18	0x00000001
s3	19	0x006ffeff
s4	20	0xb474b650
s5	21	0x00000000
s6	22	0xc2a83000
s7	23	0x9b4ad664
s8	24	0x00000000
s9	25	0xb474b650
s10	26	0xa08b002a
s11	27	0x00000004
t3	28	0x97f8e63c
t4	29	0xb006f628
t5	30	0x9b4ad664
t6	31	0xb006f628
pc		0x00400684

图 2: 运行结果

可以看到，仿真结果和运行结果吻合的很好。

3.2 Test2



图 3: 仿真结果

Name	Number	Value
zero	0	0x00000000
ra	1	0x1002017c
sp	2	0x00000000
gp	3	0x00000000
tp	4	0x00000000
t0	5	0x10020b53
t1	6	0x10015b8d
t2	7	0x00000000
s0	8	0x00000000
s1	9	0x00000000
a0	10	0x10020277
a1	11	0x1001c80d
a2	12	0x00000000
a3	13	0x0000003e
a4	14	0x10026042
a5	15	0x00000000
a6	16	0x00000000
a7	17	0x00000000
s2	18	0x00000000
s3	19	0x00000000
s4	20	0x1001edad
s5	21	0x00000000
s6	22	0x00000000
s7	23	0x0000003e
s8	24	0x10035516
s9	25	0x10035cac
s10	26	0x0000007d
s11	27	0x1003b6b4
t3	28	0x00000000
t4	29	0x00000000
t5	30	0x10011ce8
t6	31	0x0000007d
pc		0x00400754

图 4: 运行结果

可以看到，仿真结果和运行结果吻合的很好。

4 电路设计与分析

4.1 前递模块

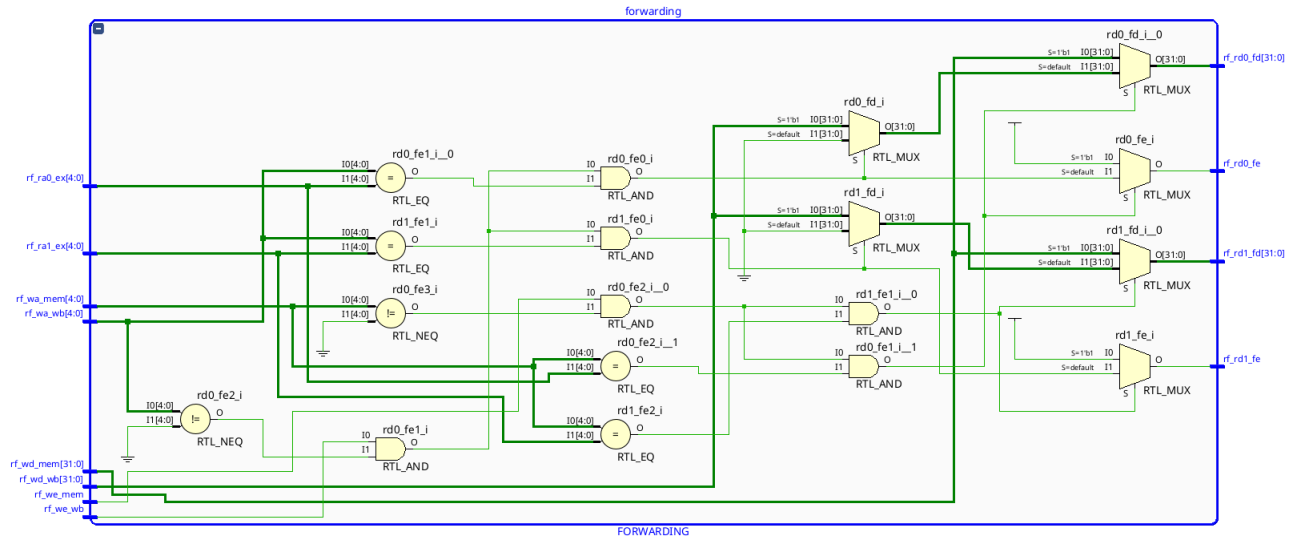


图 5: 前递模块

4.2 段间寄存器控制模块

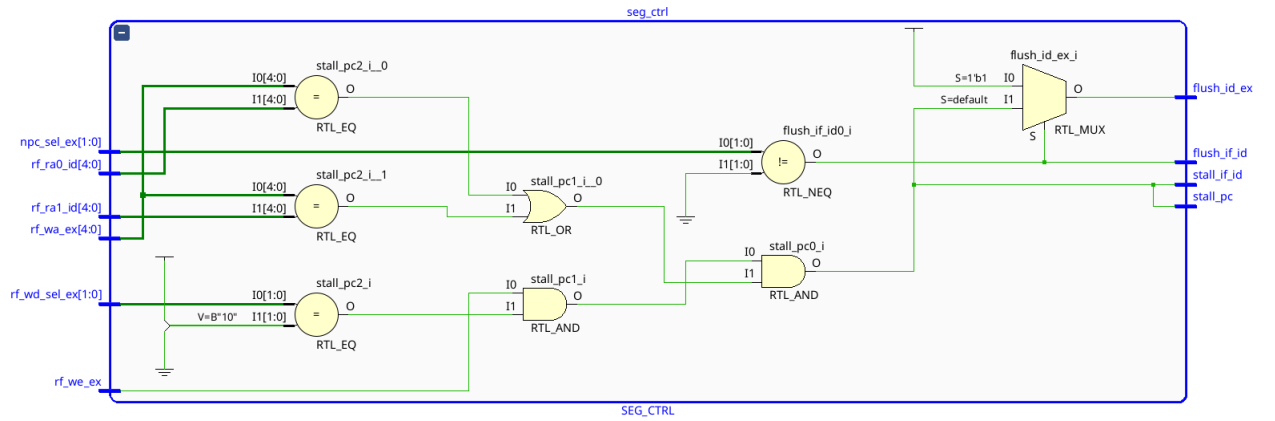


图 6: 段间寄存器控制模块

4.3 CPU 模块

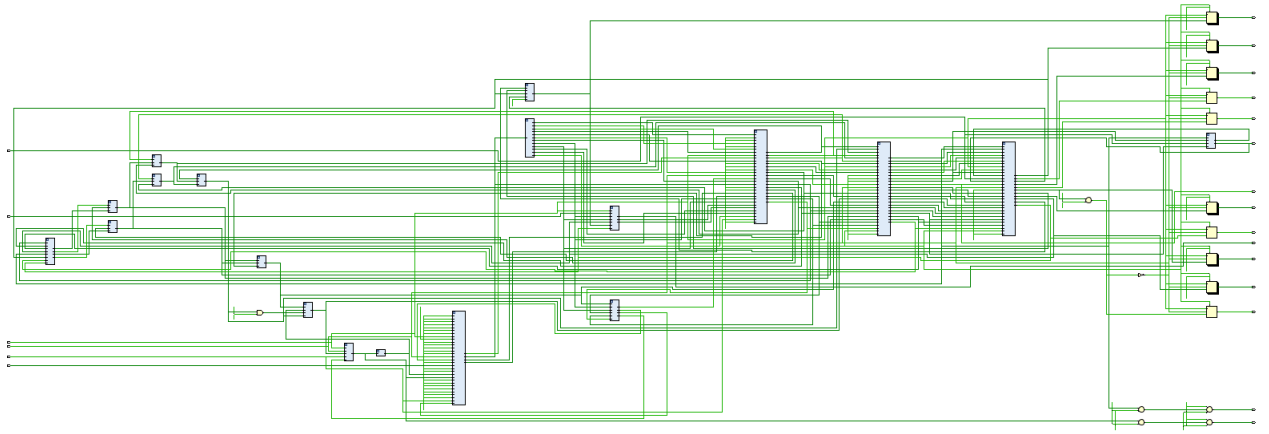


图 7: CPU 模块

5 测试结果与分析

5.1 Test1

```
FPGAOL UART xterm.js 1.1
R;
----- CPU -----

User: RR 0 16;
00000000 00000001 000006E3 F4B2E614
00000000 00000000 00000004 7F77FFFF
DFFCFD5 60CBE000 006FFEFF F1165000
F4B2E614 7594D000 00000000 0A438000

User: RR 10 16;
0A438000 60CBE000 00000001 006FFEFF
B474B650 00000000 C2A83000 9B4AD664
00000000 B474B650 A08B002A 00000004
97F8E63C B006F628 9B4AD664 B006F628

User:
```

图 8: 上板运行结果

可以看到，上板结果和运行结果吻合的很好。

5.2 Test2

```
FPGAOL UART xterm.js 1.1
R;
----- CPU -----

User: RR 0 16;
00000000 1002017C 00000000 00000000
00000000 10020B53 10015B8D 00000000
00000000 00000000 10020277 1001C80D
00000000 0000003E 10026042 00000000

User: RR 10 16;
00000000 00000000 00000000 00000000
1001EDAD 00000000 00000000 0000003E
10035516 10035CAC 0000007D 1003B6B4
00000000 00000000 10011CE8 0000007D

User:
```

图 9: 上板运行结果

可以看到，上板结果和运行结果吻合的很好。

6 总结

本次实验实现了前递模块与段间寄存器控制模块，并将其加入 CPU 模块，从而实现了一个完整的五级流水线 CPU。