实验 5：无冒险流水线 CPU

张子康 PB22020660

2024 年 4 月 29 日

1 实验目的与内容

1.1 实验目的

将上一次实验设计的 CPU 流水化，形成一个不考虑冒险的流水线 CPU。

1.2 实验内容

1.2.1 任务 1：写优先的寄存器堆

根据实验文档中的介绍将寄存器堆改为写优先的模式，并仿真测试正确性。

1.2.2 任务 2：无冒险流水线

正确设计并例化四个段间寄存器，连线以实现无冒险流水线 CPU。最终，你需要在 FPGAOL 上上板运行，并通过我们给出的测试程序。

2 逻辑设计

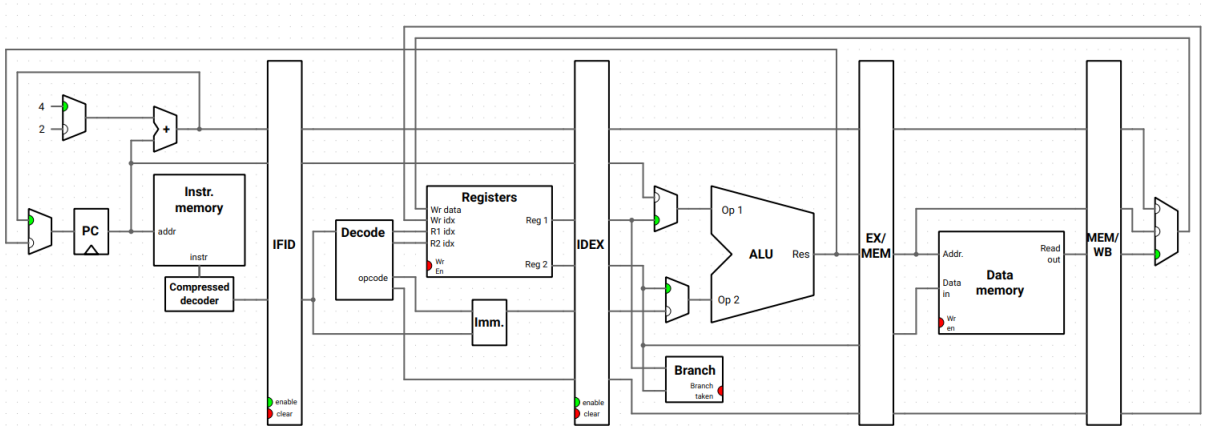

图 1: 无冒险五级流水线数据通路

2.1 任务 1：写优先的寄存器堆

写优先寄存器堆的实现如下：

```
module REG_FILE (input [0 : 0] clk,
                 input [4 : 0] rf_ra0,
                 input [4 : 0] rf_ra1,
                 input [4 : 0] rf_wa,
                 input [0 : 0] rf_we,
                 input [31 : 0] rf_wd,
                 output [31 : 0] rf_rd0,
                 output [31 : 0] rf_rd1,
                 input [4:0] debug_reg_ra,
                 output [31:0] debug_reg_rd);

reg [31 : 0] reg_file [0 : 31];

// 用于初始化寄存器
integer i;
initial begin
    for (i = 0; i < 32; i = i + 1)
        reg_file[i] = 0;
end

// 写优先
assign rf_rd0 = (rf_ra0==rf_wa && rf_ra0!=5'b00000 && rf_we
   )?rf_wd : reg_file[rf_ra0];
assign rf_rd1 = (rf_ra1==rf_wa && rf_ra1!=5'b00000 && rf_we
   )?rf_wd : reg_file[rf_ra1];

assign debug_reg_rd = /*(debug_reg_ra==rf_wa &&
   debug_reg_ra!=5'b00000 && rf_we)?rf_wd : */reg_file[
   debug_reg_ra];

always @(posedge clk) begin
```

```
28      if (rf_we && rf_wa != 5'd00000)
29          reg_file[rf_wa] <= rf_wd;
30      else
31          reg_file[rf_wa] <= reg_file[rf_wa];
32  end
33
34  endmodule
```

若没有使用写优先寄存器堆,给定程序结果如下:x4=32'H0000000c,x5=32'H00000017。

2.2 任务 2：无冒险流水线

段间寄存器实现如下（以 IF_ID 为例，其他段间寄存器实现相同，仅模块命名有区别）:

```
1  // 含_in后缀的为输入信号，含_out后缀的为输出信号
2  // 信号一旦产生就向后传输
3  module ID_EX(
4      input clk,
5      input rst,
6      input flush,
7      input stall,
8      input en,
9      // PC
10     input [31:0] pc_add4_in,
11     input [31:0] pc_in,
12
13     output [31:0] pc_add4_out,
14     output [31:0] pc_out,
15
16     // INST
17     input [31:0] inst_in,
18
19     output [31:0] inst_out,
20
```

```
21      // DECODER
22      input [4:0] alu_op_in,
23      input [3:0] dmem_access_in,
24      input [31:0] imm_in,
25      input [4:0] rf_ra0_in,
26      input [4:0] rf_ra1_in,
27      input [4:0] rf_wa_in,
28      input [0:0] rf_we_in,
29      input [1:0] rf_wd_sel_in,
30      input [0:0] alu_src0_sel_in,
31      input [0:0] alu_src1_sel_in,
32      input [3:0] br_type_in,
33      input [0:0] dmem_we_in,
34
35      output [4:0] alu_op_out,
36      output [3:0] dmem_access_out,
37      output [31:0] imm_out,
38      output [4:0] rf_ra0_out,
39      output [4:0] rf_ra1_out,
40      output [4:0] rf_wa_out,
41      output [0:0] rf_we_out,
42      output [1:0] rf_wd_sel_out,
43      output [0:0] alu_src0_sel_out,
44      output [0:0] alu_src1_sel_out,
45      output [3:0] br_type_out,
46      output [0:0] dmem_we_out,
47
48      // REG_FILE
49      input [31:0] rf_rd0_in,
50      input [31:0] rf_rd1_in,
51
52      output [31:0] rf_rd0_out,
53      output [31:0] rf_rd1_out,
```

```
54
55      // MUX1
56      input [31:0] alu_src0_in,
57      input [31:0] alu_src1_in,
58
59      output [31:0] alu_src0_out,
60      output [31:0] alu_src1_out,
61
62      // NPC
63      input [31:0] npc_in,
64
65      output [31:0] npc_out,
66      // BRANCH
67      input [1 : 0] npc_sel_in,
68
69      output [1 : 0] npc_sel_out,
70      // ALU
71      input [31:0] alu_res_in,
72
73      output [31:0] alu_res_out,
74
75      // SLU
76      input [31:0] dmem_rd_out_in,
77      input [31:0] dmem_wdata_mem_in,
78
79      output [31:0] dmem_rd_out_out,
80      output [31:0] dmem_wdata_mem_out,
81
82      // DM
83      input [31:0] dmem_rdata_mem_in,
84
85      output [31:0] dmem_rdata_mem_out,
86
```

```
87      // COMMIT
88      input [0:0] commit_in,
89
90      output [0:0] commit_out
91  );
92      reg [31:0] pc_add4;
93      reg [31:0] pc;
94      reg [31:0] inst;
95
96      reg [4:0] alu_op;
97      reg [3:0] dmem_access;
98      reg [31:0] imm;
99      reg [4:0] rf_ra0;
100     reg [4:0] rf_ra1;
101     reg [4:0] rf_wa;
102     reg [0:0] rf_we;
103     reg [1:0] rf_wd_sel;
104     reg [0:0] alu_src0_sel;
105     reg [0:0] alu_src1_sel;
106     reg [3:0] br_type;
107     reg [0:0] dmem_we;
108     reg [31:0] rf_rd0;
109     reg [31:0] rf_rd1;
110     reg [0:0] commit;
111
112     reg [31:0] alu_src0;
113     reg [31:0] alu_src1;
114     reg [31:0] npc;
115     reg [1:0] npc_sel;
116     reg [31:0] alu_res;
117
118     reg [31:0] dmem_rd_out;
119     reg [31:0] dmem_wdata_mem;
```

```
120     reg [31:0] dmem_rdata_mem;
121 
122     initial begin
123         commit=0;
124         pc_add4=0;
125         pc=32'h00400000;
126         inst=0;
127 
128         alu_op=5'b11111;
129         dmem_access=0;
130         imm=0;
131         rf_ra0=0;
132         rf_ra1=0;
133         rf_wa=0;
134         rf_we=0;
135         rf_wd_sel=0;
136         alu_src0_sel=0;
137         alu_src1_sel=0;
138         br_type=4'b1111;
139         dmem_we=0;
140         rf_rd0=0;
141         rf_rd1=0;
142 
143         alu_src0=0;
144         alu_src1=0;
145         npc=0;
146         npc_sel=0;
147         alu_res=0;
148 
149         dmem_rd_out=0;
150         dmem_wdata_mem=0;
151         dmem_rdata_mem=0;
152     end
```

```
153
154     assign commit_out=commit;
155     assign pc_add4_out=pc_add4;
156     assign pc_out=pc;
157     assign inst_out=inst;
158
159     assign alu_op_out=alu_op;
160     assign dmem_access_out=dmem_access;
161     assign imm_out=imm;
162     assign rf_ra0_out=rf_ra0;
163     assign rf_ra1_out=rf_ra1;
164     assign rf_wa_out=rf_wa;
165     assign rf_we_out=rf_we;
166     assign rf_wd_sel_out=rf_wd_sel;
167     assign alu_src0_sel_out=alu_src0_sel;
168     assign alu_src1_sel_out=alu_src1_sel;
169     assign br_type_out=br_type;
170     assign dmem_we_out=dmem_we;
171     assign rf_rd0_out=rf_rd0;
172     assign rf_rd1_out=rf_rd1;
173
174     assign alu_src0_out=alu_src0;
175     assign alu_src1_out=alu_src1;
176     assign npc_out=npc;
177     assign npc_sel_out=npc_sel;
178     assign alu_res_out=alu_res;
179     assign dmem_rd_out_out=dmem_rd_out;
180     assign dmem_wdata_mem_out=dmem_wdata_mem;
181     assign dmem_rdata_mem_out=dmem_rdata_mem;
182
183     always @(posedge clk) begin
184         // 复位信号
185         if(rst)begin
```

```
186             commit<=0;
187             pc_add4<=0;
188             pc<=32'h00400000;
189             inst<=0;
190 
191             alu_op<=5'b11111;
192             dmem_access<=0;
193             imm<=0;
194             rf_ra0<=0;
195             rf_ra1<=0;
196             rf_wa<=0;
197             rf_we<=0;
198             rf_wd_sel<=0;
199             alu_src0_sel<=0;
200             alu_src1_sel<=0;
201             br_type<=4'b1111;
202             dmem_we<=0;
203             rf_rd0=0;
204             rf_rd1=0;
205 
206             alu_src0<=0;
207             alu_src1<=0;
208             npc<=0;
209             npc_sel<=0;
210             alu_res<=0;
211 
212             dmem_rd_out<=0;
213             dmem_wdata_mem<=0;
214             dmem_rdata_mem<=0;
215         end
216         else if(en)begin
217             // flush信号
218             if(flush)begin
```

```
219                 commit<=0;
220                 pc_add4<=0;
221                 pc<=32'h00400000;
222                 inst<=0;
223 
224                 alu_op<=5'b11111;
225                 dmem_access<=0;
226                 imm<=0;
227                 rf_ra0<=0;
228                 rf_ra1<=0;
229                 rf_wa<=0;
230                 rf_we<=0;
231                 rf_wd_sel<=0;
232                 alu_src0_sel<=0;
233                 alu_src1_sel<=0;
234                 br_type<=4'b1111;
235                 dmem_we<=0;
236                 rf_rd0=0;
237                 rf_rd1=0;
238 
239                 alu_src0<=0;
240                 alu_src1<=0;
241                 npc<=0;
242                 npc_sel<=0;
243                 alu_res<=0;
244 
245                 dmem_rd_out<=0;
246                 dmem_wdata_mem<=0;
247                 dmem_rdata_mem<=0;
248             end
249             // stall信号
250             else if(stall)begin
251                 commit<=commit;
```

```
252
253                 pc_add4<=pc_add4;
254                 pc<=pc;
255                 inst<=inst;
256
257                 alu_op<=alu_op;
258                 dmem_access<=dmem_access;
259                 imm<=imm;
260                 rf_ra0<=rf_ra0;
261                 rf_ra1<=rf_ra1;
262                 rf_wa<=rf_wa;
263                 rf_we<=rf_we;
264                 rf_wd_sel<=rf_wd_sel;
265                 alu_src0_sel<=alu_src0_sel;
266                 alu_src1_sel<=alu_src1_sel;
267                 br_type<=br_type;
268                 dmem_we<=dmem_we;
269                 rf_rd0<=rf_rd0;
270                 rf_rd1<=rf_rd1;
271
272                 alu_src0<=alu_src0;
273                 alu_src1<=alu_src1;
274                 npc<=npc;
275                 npc_sel<=npc_sel;
276                 alu_res<=alu_res;
277
278                 dmem_rd_out<=dmem_rd_out;
279                 dmem_wdata_mem<=dmem_wdata_mem;
280                 dmem_rdata_mem<=dmem_rdata_mem;
281             end
282             // 在写使能且不进行复位，flush，stall情况下，
283             // 更新寄存器的值
284             else begin
```

```
285                 commit<=commit_in;
286
287                 pc_add4<=pc_add4_in;
288                 pc<=pc_in;
289                 inst<=inst_in;
290
291                 alu_op<=alu_op_in;
292                 dmem_access<=dmem_access_in;
293                 imm<=imm_in;
294                 rf_ra0<=rf_ra0_in;
295                 rf_ra1<=rf_ra1_in;
296                 rf_wa<=rf_wa_in;
297                 rf_we<=rf_we_in;
298                 rf_wd_sel<=rf_wd_sel_in;
299                 alu_src0_sel<=alu_src0_sel_in;
300                 alu_src1_sel<=alu_src1_sel_in;
301                 br_type<=br_type_in;
302                 dmem_we<=dmem_we_in;
303                 rf_rd0=rf_rd0_in;
304                 rf_rd1=rf_rd1_in;
305
306                 alu_src0<=alu_src0_in;
307                 alu_src1<=alu_src1_in;
308                 npc<=npc_in;
309                 npc_sel<=npc_sel_in;
310                 alu_res<=alu_res_in;
311
312                 dmem_rd_out<=dmem_rd_out_in;
313                 dmem_wdata_mem<=dmem_wdata_mem_in;
314                 dmem_rdata_mem<=dmem_rdata_mem_in;
315             end
316         end
317         else begin
```

```
318             commit<=commit;
319 
320             pc_add4<=pc_add4;
321             pc<=pc;
322             inst<=inst;
323 
324             alu_op<=alu_op;
325             dmem_access<=dmem_access;
326             imm<=imm;
327             rf_ra0<=rf_ra0;
328             rf_ra1<=rf_ra1;
329             rf_wa<=rf_wa;
330             rf_we<=rf_we;
331             rf_wd_sel<=rf_wd_sel;
332             alu_src0_sel<=alu_src0_sel;
333             alu_src1_sel<=alu_src1_sel;
334             br_type<=br_type;
335             dmem_we<=dmem_we;
336             rf_rd0=rf_rd0;
337             rf_rd1=rf_rd1;
338 
339             alu_src0<=alu_src0;
340             alu_src1<=alu_src1;
341             npc<=npc;
342             npc_sel<=npc_sel;
343             alu_res<=alu_res;
344 
345             dmem_rd_out<=dmem_rd_out;
346             dmem_wdata_mem<=dmem_wdata_mem;
347             dmem_rdata_mem<=dmem_rdata_mem;
348         end
349     end
350 endmodule
```

对于无用的输入，在例化时将其端口置 0。

CPU 模块如下：

```
`include "./include/config.v"

module CPU (input [0 : 0] clk,
            input [0 : 0] rst,
            input [0 : 0] global_en,
            output [31 : 0] imem_raddr,
            input [31 : 0] imem_rdata,
            input [31 : 0] dmem_rdata,      // Unused
            output [0 : 0] dmem_we,        // Unused
            output [31 : 0] dmem_addr,      // Unused
            output [31 : 0] dmem_wdata,     // Unused
            output [0 : 0] commit,
            output [31 : 0] commit_pc,
            output [31 : 0] commit_inst,
            output [0 : 0] commit_halt,
            output [0 : 0] commit_reg_we,
            output [4 : 0] commit_reg_wa,
            output [31 : 0] commit_reg_wd,
            output [0 : 0] commit_dmem_we,
            output [31 : 0] commit_dmem_wa,
            output [31 : 0] commit_dmem_wd,
            input [4 : 0] debug_reg_ra,
            output [31 : 0] debug_reg_rd);


    // 定义所需的网线
    wire [0:0] commit_if;
    wire [31:0] pc_if;
    wire [31:0] pcadd4_if;
    wire [31:0] inst_if;

```

```
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 flush,stall,en;
134     wire flush_if_id,flush_id_ex,flush_ex_mem,flush_mem_wb;
135 
136     assign commit_if = 1;
137     assign stall = 0;
138     assign en = global_en;
139     // assign global_en  = !(inst_if == 32'H00100073);
140     assign imem_raddr = (pc_if-32'h00400000)/'d4;
141     assign inst_if   = imem_rdata;
142     assign pc_j_ex       = alu_res_ex&~1;
143     assign dmem_wd_in = rf_rd1_mem;
144     assign dmem_we    = dmem_we_mem;
145     assign dmem_addr  = (alu_res_mem-32'h10010000)/'d4;
146     assign dmem_wdata = dmem_wdata_mem;
147     assign dmem_rdata_mem = dmem_rdata;
148 
149     // 控制各个段间寄存器的flush信号
150     assign flush_if_id=inst_if==32'h00000013 && inst_id
        ==32'h00000013;
151     assign flush_id_ex=inst_if==32'h00000013 && inst_id
        ==32'h00000013;
152     assign flush_ex_mem=0;
153     assign flush_mem_wb=0;
154 
155     // 例化各个模块
156     PC_PLUS4 pc_plus(
157         .pc(pc_if),
158         .pc_plus4(pcadd4_if)
159     );
160 
161     PC pc(
```

```
162         .clk    (clk),
163         .rst    (rst),
164         .en     (global_en),    // 当 global_en 为高电平
                时，PC 才会更新，CPU 才会执行指令。
165         .npc    (npc_ex),
166         .pc     (pc_if)
167     );
168
169     IF_ID if_id(
170         .clk(clk),
171         .rst(rst),
172         .flush(flush_if_id),
173         .stall(stall),
174         .en(en),
175         //PC
176         .pc_add4_in(pcadd4_if),
177         .pc_in(pc_if),
178         .pc_add4_out(pcadd4_id),
179         .pc_out(pc_id),
180         //INST
181         .inst_in(inst_if),
182         .inst_out(inst_id),
183         //DECODER
184         .alu_op_in(0),
185         .dmem_access_in(0),
186         .imm_in(0),
187         .rf_ra0_in(0),
188         .rf_ra1_in(0),
189         .rf_wa_in(0),
190         .rf_we_in(0),
191         .rf_wd_sel_in(0),
192         .alu_src0_sel_in(0),
193         .alu_src1_sel_in(0),
```

```
194         .br_type_in(0),
195         .dmem_we_in(0),
196         .alu_op_out(),
197         .dmem_access_out(),
198         .imm_out(),
199         .rf_ra0_out(),
200         .rf_ra1_out(),
201         .rf_wa_out(),
202         .rf_we_out(),
203         .rf_wd_sel_out(),
204         .alu_src0_sel_out(),
205         .alu_src1_sel_out(),
206         .br_type_out(),
207         .dmem_we_out(),
208         //REG_FILE
209         .rf_rd0_in(0),
210         .rf_rd1_in(0),
211         .rf_rd0_out(),
212         .rf_rd1_out(),
213         //MUX1
214         .alu_src0_in(0),
215         .alu_src1_in(0),
216         .alu_src0_out(),
217         .alu_src1_out(),
218         //NPC
219         .npc_in(0),
220         .npc_out(),
221         //BRANCH
222         .npc_sel_in(0),
223         .npc_sel_out(),
224         //ALU
225         .alu_res_in(0),
226         .alu_res_out(),
```

```
227         //SLU
228         .dmem_rd_out_in(0),
229         .dmem_wdata_mem_in(0),
230         .dmem_rd_out_out(),
231         .dmem_wdata_mem_out(),
232         //DM
233         .dmem_rdata_mem_in(0),
234         .dmem_rdata_mem_out(),
235         .commit_in(commit_if),
236         .commit_out(commit_id)
237     );
238 
239     DECODER decoder(
240         .inst(inst_id),
241         .alu_op(alu_op_id),
242         .imm(imm_id),
243         .rf_ra0(rf_ra0_id),
244         .rf_ra1(rf_ra1_id),
245         .rf_wa(rf_wa_id),
246         .rf_we(rf_we_id),
247         .alu_src0_sel(alu_src0_sel_id),
248         .alu_src1_sel(alu_src1_sel_id),
249         .dmem_access(dmem_access_id),
250         .rf_wd_sel(rf_wd_sel_id),
251         .br_type(br_type_id),
252         .dmem_we(dmem_we_id)
253     );
254 
255     REG_FILE reg_file(
256         .clk(clk),
257         .rf_ra0(rf_ra0_id),
258         .rf_ra1(rf_ra1_id),
259         .rf_wa(rf_wa_wb),
```

```
260         .rf_we(rf_we_wb),
261         .rf_wd(rf_wd_wb),
262         .rf_rd0(rf_rd0_id),
263         .rf_rd1(rf_rd1_id),
264         .debug_reg_rd(debug_reg_rd),
265         .debug_reg_ra(debug_reg_ra)
266     );
267 
268     ID_EX id_ex(
269         .clk(clk),
270         .rst(rst),
271         .flush(flush_id_ex),
272         .stall(stall),
273         .en(en),
274         //PC
275         .pc_add4_in(pcadd4_id),
276         .pc_in(pc_id),
277         .pc_add4_out(pcadd4_ex),
278         .pc_out(pc_ex),
279         //INST
280         .inst_in(inst_id),
281         .inst_out(inst_ex),
282         //DECODER
283         .alu_op_in(alu_op_id),
284         .dmem_access_in(dmem_access_id),
285         .imm_in(imm_id),
286         .rf_ra0_in(rf_ra0_id),
287         .rf_ra1_in(rf_ra1_id),
288         .rf_wa_in(rf_wa_id),
289         .rf_we_in(rf_we_id),
290         .rf_wd_sel_in(rf_wd_sel_id),
291         .alu_src0_sel_in(alu_src0_sel_id),
292         .alu_src1_sel_in(alu_src1_sel_id),
```

```
293         .br_type_in(br_type_id),
294         .dmem_we_in(dmem_we_id),
295         .alu_op_out(alu_op_ex),
296         .dmem_access_out(dmem_access_ex),
297         .imm_out(imm_ex),
298         .rf_ra0_out(rf_ra0_ex),
299         .rf_ra1_out(rf_ra1_ex),
300         .rf_wa_out(rf_wa_ex),
301         .rf_we_out(rf_we_ex),
302         .rf_wd_sel_out(rf_wd_sel_ex),
303         .alu_src0_sel_out(alu_src0_sel_ex),
304         .alu_src1_sel_out(alu_src1_sel_ex),
305         .br_type_out(br_type_ex),
306         .dmem_we_out(dmem_we_ex),
307         //REG_FILE
308         .rf_rd0_in(rf_rd0_id),
309         .rf_rd1_in(rf_rd1_id),
310         .rf_rd0_out(rf_rd0_ex),
311         .rf_rd1_out(rf_rd1_ex),
312         //MUX1
313         .alu_src0_in(0),
314         .alu_src1_in(0),
315         .alu_src0_out(),
316         .alu_src1_out(),
317         //NPC
318         .npc_in(0),
319         .npc_out(),
320         //BRANCH
321         .npc_sel_in(0),
322         .npc_sel_out(),
323         //ALU
324         .alu_res_in(0),
325         .alu_res_out(),
```

```
326         //SLU
327         .dmem_rd_out_in(0),
328         .dmem_wdata_mem_in(0),
329         .dmem_rd_out_out(),
330         .dmem_wdata_mem_out(),
331         //DM
332         .dmem_rdata_mem_in(0),
333         .dmem_rdata_mem_out(),
334         .commit_in(commit_id),
335         .commit_out(commit_ex)
336     );
337 
338     MUX1 mux1(
339         .src0(rf_rd0_ex),
340         .src1(pc_ex),
341         .sel(alu_src0_sel_ex),
342         .res(alu_src0_ex)
343     );
344 
345     MUX1 mux2(
346         .src0(rf_rd1_ex),
347         .src1(imm_ex),
348         .sel(alu_src1_sel_ex),
349         .res(alu_src1_ex)
350     );
351 
352     ALU alu(
353         .alu_src0(alu_src0_ex),
354         .alu_src1(alu_src1_ex),
355         .alu_op(alu_op_ex),
356         .alu_res(alu_res_ex)
357     );
358 
```

```
359     BRANCH branch(
360         .br_type(br_type_ex),
361         .br_src0(rf_rd0_ex),
362         .br_src1(rf_rd1_ex),
363         .npc_sel(npc_sel_ex)
364     );
365     NPC npc(
366         .pc_offset(alu_res_ex),
367         .pc_add4(pcadd4_if),
368         .pc_j(pc_j_ex),
369         .npc_sel(npc_sel_ex),
370         .npc(npc_ex)
371     );
372 
373     EX_MEM ex_mem(
374         .clk(clk),
375         .rst(rst),
376         .flush(flush_ex_mem),
377         .stall(stall),
378         .en(en),
379         //PC
380         .pc_add4_in(pcadd4_ex),
381         .pc_in(pc_ex),
382         .pc_add4_out(pcadd4_mem),
383         .pc_out(pc_mem),
384         //INST
385         .inst_in(inst_ex),
386         .inst_out(inst_mem),
387         //DECODER
388         .alu_op_in(alu_op_ex),
389         .dmem_access_in(dmem_access_ex),
390         .imm_in(imm_ex),
391         .rf_ra0_in(rf_ra0_ex),
```

```
392         .rf_ra1_in(rf_ra1_ex),
393         .rf_wa_in(rf_wa_ex),
394         .rf_we_in(rf_we_ex),
395         .rf_wd_sel_in(rf_wd_sel_ex),
396         .alu_src0_sel_in(alu_src0_sel_ex),
397         .alu_src1_sel_in(alu_src1_sel_ex),
398         .br_type_in(br_type_ex),
399         .dmem_we_in(dmem_we_ex),
400         .alu_op_out(alu_op_mem),
401         .dmem_access_out(dmem_access_mem),
402         .imm_out(imm_mem),
403         .rf_ra0_out(rf_ra0_mem),
404         .rf_ra1_out(rf_ra1_mem),
405         .rf_wa_out(rf_wa_mem),
406         .rf_we_out(rf_we_mem),
407         .rf_wd_sel_out(rf_wd_sel_mem),
408         .alu_src0_sel_out(alu_src0_sel_mem),
409         .alu_src1_sel_out(alu_src1_sel_mem),
410         .br_type_out(br_type_mem),
411         .dmem_we_out(dmem_we_mem),
412         //REG_FILE
413         .rf_rd0_in(rf_rd0_ex),
414         .rf_rd1_in(rf_rd1_ex),
415         .rf_rd0_out(rf_rd0_mem),
416         .rf_rd1_out(rf_rd1_mem),
417         //MUX1
418         .alu_src0_in(alu_src0_ex),
419         .alu_src1_in(alu_src1_ex),
420         .alu_src0_out(alu_src0_mem),
421         .alu_src1_out(alu_src1_mem),
422         //NPC
423         .npc_in(npc_ex),
424         .npc_out(npc_mem),
```

```
425         //BRANCH
426         .npc_sel_in(npc_sel_ex),
427         .npc_sel_out(npc_sel_mem),
428         //ALU
429         .alu_res_in(alu_res_ex),
430         .alu_res_out(alu_res_mem),
431         //SLU
432         .dmem_rd_out_in(0),
433         .dmem_wdata_mem_in(0),
434         .dmem_rd_out_out(),
435         .dmem_wdata_mem_out(),
436         //DM
437         .dmem_rdata_mem_in(0),
438         .dmem_rdata_mem_out(),
439         .commit_in(commit_ex),
440         .commit_out(commit_mem)
441     );
442 
443     SLU slu(
444         .addr(alu_res_mem),
445         .dmem_access(dmem_access_mem),
446         .rd_in(dmem_rdata_mem),
447         .wd_in(rf_rd1_mem),
448         .rd_out(dmem_rd_out_mem),
449         .wd_out(dmem_wdata_mem)
450     );
451 
452     MEM_WB mem_wb(
453         .clk(clk),
454         .rst(rst),
455         .flush(flush_mem_wb),
456         .stall(stall),
457         .en(en),
```

```
458         //PC
459         .pc_add4_in(pcadd4_mem),
460         .pc_in(pc_mem),
461         .pc_add4_out(pcadd4_wb),
462         .pc_out(pc_wb),
463         //INST
464         .inst_in(inst_mem),
465         .inst_out(inst_wb),
466         //DECODER
467         .alu_op_in(alu_op_mem),
468         .dmem_access_in(dmem_access_mem),
469         .imm_in(imm_mem),
470         .rf_ra0_in(rf_ra0_mem),
471         .rf_ra1_in(rf_ra1_mem),
472         .rf_wa_in(rf_wa_mem),
473         .rf_we_in(rf_we_mem),
474         .rf_wd_sel_in(rf_wd_sel_mem),
475         .alu_src0_sel_in(alu_src0_sel_mem),
476         .alu_src1_sel_in(alu_src1_sel_mem),
477         .br_type_in(br_type_mem),
478         .dmem_we_in(dmem_we_mem),
479         .alu_op_out(alu_op_wb),
480         .dmem_access_out(dmem_access_wb),
481         .imm_out(imm_wb),
482         .rf_ra0_out(rf_ra0_wb),
483         .rf_ra1_out(rf_ra1_wb),
484         .rf_wa_out(rf_wa_wb),
485         .rf_we_out(rf_we_wb),
486         .rf_wd_sel_out(rf_wd_sel_wb),
487         .alu_src0_sel_out(alu_src0_sel_wb),
488         .alu_src1_sel_out(alu_src1_sel_wb),
489         .br_type_out(br_type_wb),
490         .dmem_we_out(dmem_we_wb),
```

```
491         //REG_FILE
492         .rf_rd0_in(rf_rd0_mem),
493         .rf_rd1_in(rf_rd1_mem),
494         .rf_rd0_out(rf_rd0_wb),
495         .rf_rd1_out(rf_rd1_wb),
496         //MUX1
497         .alu_src0_in(alu_src0_mem),
498         .alu_src1_in(alu_src1_mem),
499         .alu_src0_out(alu_src0_wb),
500         .alu_src1_out(alu_src1_wb),
501         //NPC
502         .npc_in(npc_mem),
503         .npc_out(npc_wb),
504         //BRANCH
505         .npc_sel_in(npc_sel_mem),
506         .npc_sel_out(npc_sel_wb),
507         //ALU
508         .alu_res_in(alu_res_mem),
509         .alu_res_out(alu_res_wb),
510         //SLU
511         .dmem_rd_out_in(dmem_rd_out_mem),
512         .dmem_wdata_mem_in(dmem_wdata_mem),
513         .dmem_rd_out_out(dmem_rd_out_wb),
514         .dmem_wdata_mem_out(dmem_wdata_wb),
515         //DM
516         .dmem_rdata_mem_in(dmem_rdata_mem),
517         .dmem_rdata_mem_out(dmem_rdata_wb),
518         .commit_in(commit_mem),
519         .commit_out(commit_wb)
520     );
521 
522     MUX2 RF_MUX(
523         .src0(pcadd4_wb),
```

```
524         .src1(alu_res_wb),
525         .src2(dmem_rd_out_wb),
526         .src3(0),
527         .sel(rf_wd_sel_wb),
528         .res(rf_wd_wb)
529     );
530 
531     /* ---------------------------- */
532     /*           Commit             */
533     /* ---------------------------- */
534 
535     // wire [0 : 0] commit_if     ;
536     // assign commit_if = 1'H1;
537 
538     reg  [0 : 0]   commit_reg          ;
539     reg  [31 : 0]   commit_pc_reg       ;
540     reg  [31 : 0]   commit_inst_reg     ;
541     reg  [0 : 0]   commit_halt_reg     ;
542     reg  [0 : 0]   commit_reg_we_reg   ;
543     reg  [4 : 0]   commit_reg_wa_reg   ;
544     reg  [31 : 0]   commit_reg_wd_reg   ;
545     reg  [0 : 0]   commit_dmem_we_reg  ;
546     reg  [31 : 0]   commit_dmem_wa_reg  ;
547     reg  [31 : 0]   commit_dmem_wd_reg  ;
548 
549     always @(posedge clk) begin
550         if (rst) begin
551             commit_reg         <= 1'H0;
552             commit_pc_reg      <= 32'H0;
553             commit_inst_reg    <= 32'H0;
554             commit_halt_reg    <= 1'H0;
555             commit_reg_we_reg  <= 1'H0;
556             commit_reg_wa_reg  <= 5'H0;
```

```
557             commit_reg_wd_reg  <= 32'H0;
558             commit_dmem_we_reg <= 1'H0;
559             commit_dmem_wa_reg <= 32'H0;
560             commit_dmem_wd_reg <= 32'H0;
561         end
562         else if (global_en) begin
563             commit_reg          <= commit_wb;
564             commit_pc_reg       <= pc_wb;
565             commit_inst_reg     <= inst_wb;
566             commit_halt_reg     <= inst_wb == `HALT_INST;
567             commit_reg_we_reg   <= rf_we_wb;
568             commit_reg_wa_reg   <= rf_wa_wb;
569             commit_reg_wd_reg   <= rf_wd_wb;
570             commit_dmem_we_reg  <= dmem_we_wb;
571             commit_dmem_wa_reg  <= alu_res_wb;
572             commit_dmem_wd_reg  <= dmem_wdata_wb;
573         end
574             end
575
576             assign commit          = commit_reg;
577             assign commit_pc       = commit_pc_reg;
578             assign commit_inst     = commit_inst_reg;
579             assign commit_halt     = commit_halt_reg;
580             assign commit_reg_we   = commit_reg_we_reg;
581             assign commit_reg_wa   = commit_reg_wa_reg;
582             assign commit_reg_wd   = commit_reg_wd_reg;
583             assign commit_dmem_we  = commit_dmem_we_reg;
584             assign commit_dmem_wa  = commit_dmem_wa_reg;
585             assign commit_dmem_wd  = commit_dmem_wd_reg;
586 endmodule
```

3 仿真结果与分析

Testbench 如下：

```
`timescale 1ns / 1ps
module TB();
reg clk;
reg rst;
wire [31:0] imem_raddr;
wire dmem_we;
wire [31 : 0]            dmem_addr;
wire [31:0] imem_rdata;
wire [31 : 0]            dmem_wdata;
wire [ 0 : 0]            commit;
wire [31 : 0]            commit_pc;
wire [31 : 0]            commit_inst;
wire [ 0 : 0]            commit_halt;
wire [ 0 : 0]            commit_reg_we;
wire [ 4 : 0]            commit_reg_wa;
wire [31 : 0]            commit_reg_wd;
wire [ 0 : 0]            commit_dmem_we;
wire [31 : 0]            commit_dmem_wa;
wire [31 : 0]            commit_dmem_wd;
wire [31 : 0]            debug_reg_rd;
wire [31:0] dmem_rdata;
CPU cpu(
    .rst (rst),
    .clk (clk),
    .global_en(1'b1),
    .debug_reg_ra(0),
    .imem_raddr(imem_raddr),
    .imem_rdata(imem_rdata),
    .dmem_we(dmem_we),
    .dmem_addr(dmem_addr),
```

```
31      .dmem_wdata(dmem_wdata),
32      .commit(commit),
33      .commit_pc(commit_pc),
34      .commit_inst(commit_inst),
35      .commit_halt(commit_halt),
36      .commit_reg_we(commit_reg_we),
37      .commit_reg_wa(commit_reg_wa),
38      .commit_reg_wd(commit_reg_wd),
39      .commit_dmem_we(commit_dmem_we),
40      .commit_dmem_wa(commit_dmem_wa),
41      .commit_dmem_wd(commit_dmem_wd),
42      .debug_reg_rd(debug_reg_rd),
43      .dmem_rdata(dmem_rdata)
44  );
45
46  INST_MEM inst_mem (
47    .a(imem_raddr),      // input wire [8 : 0] a
48    .d(d),       // input wire [31 : 0] d
49    .clk(clk),  // input wire clk
50    .we(0),    // input wire we
51    .spo(imem_rdata)  // output wire [31 : 0] spo
52  );
53  DATA_MEM data_mem (
54    .a(dmem_addr),      // input wire [8 : 0] a
55    .d(dmem_wdata),      // input wire [31 : 0] d
56    .clk(clk),  // input wire clk
57    .we(dmem_we),    // input wire we
58    .spo(dmem_rdata)  // output wire [31 : 0] spo
59  );
60  localparam CLK_PERIOD = 10;
61  always #(CLK_PERIOD/2) clk=~clk;
62
63  initial begin
```

```
64      clk=0;
65      rst=0;
66      #10000
67      $finish;
68  end
69
70  endmodule
```

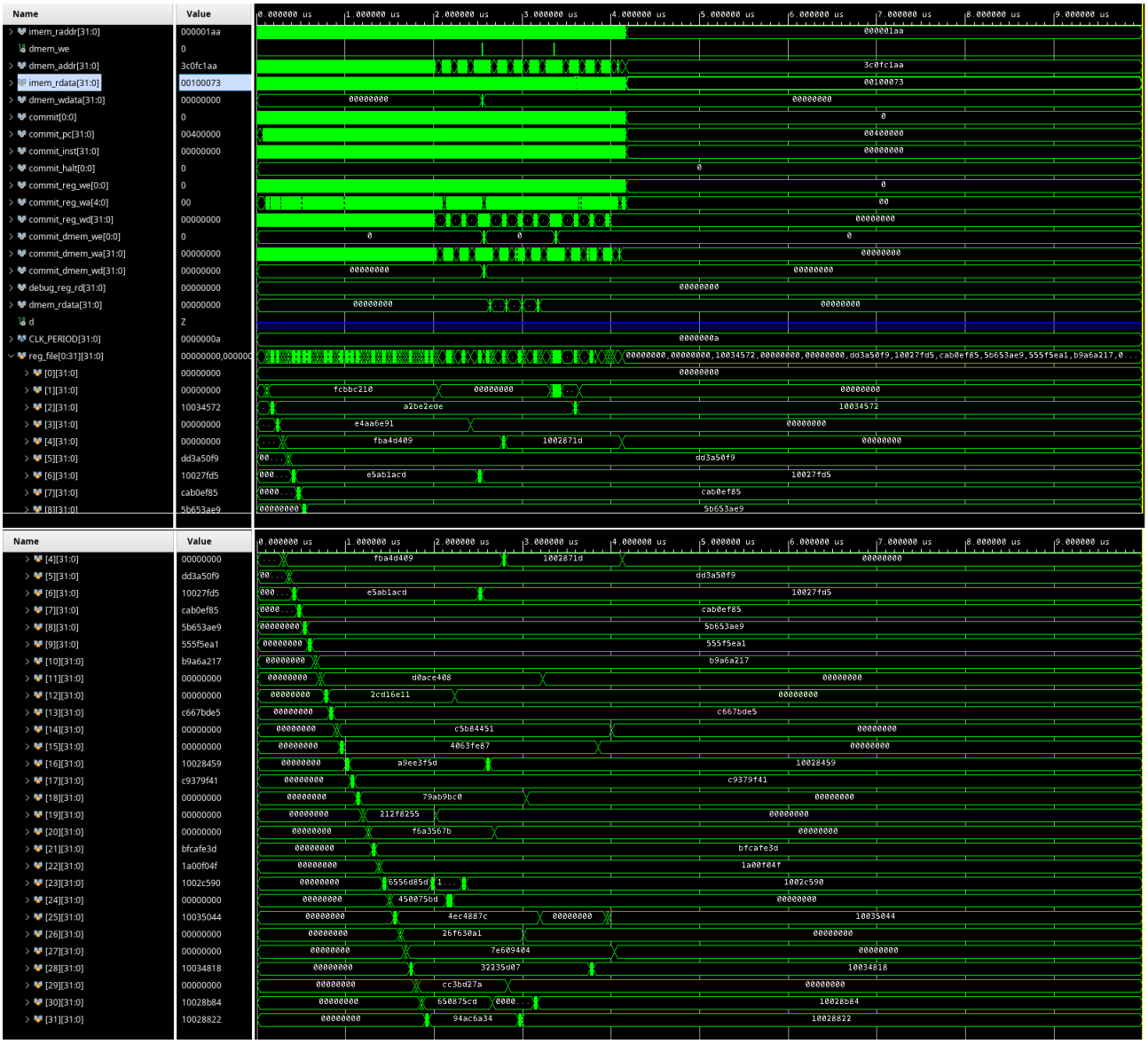

图 2: 测试程序 1 的测试结果

图 3: 测试程序 2 的测试结果

仿真结果与正确结果相吻合。

4 电路设计与分析

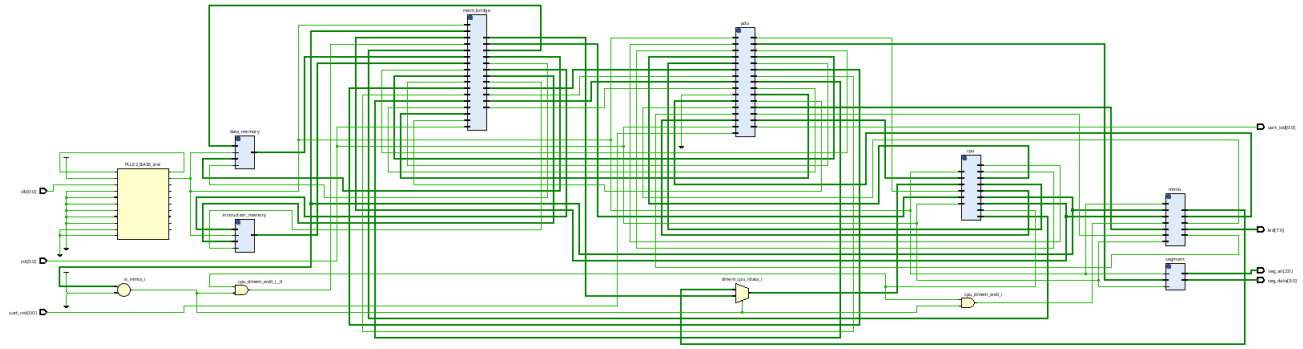

图 4: 整体的电路

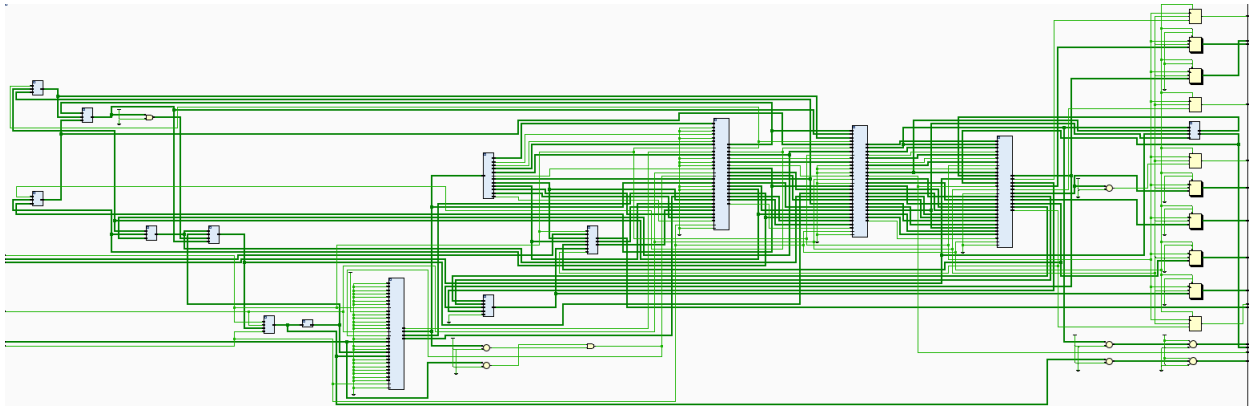

图 5: cpu 的电路

5 测试结果与分析

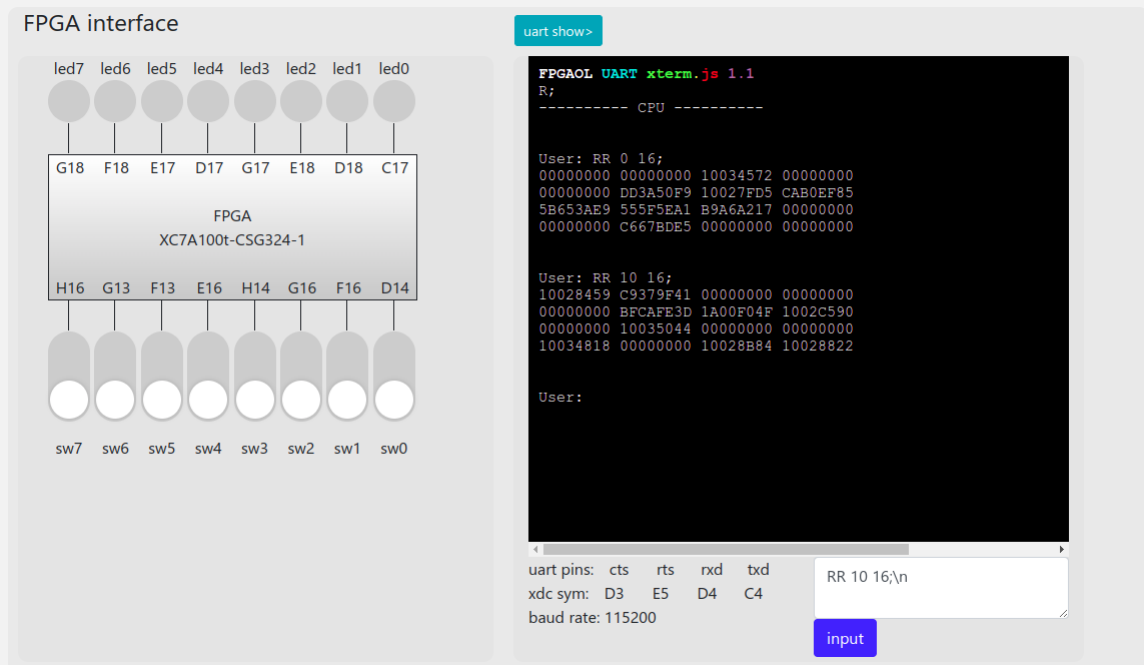

图 6: 测试程序 1 上板结果

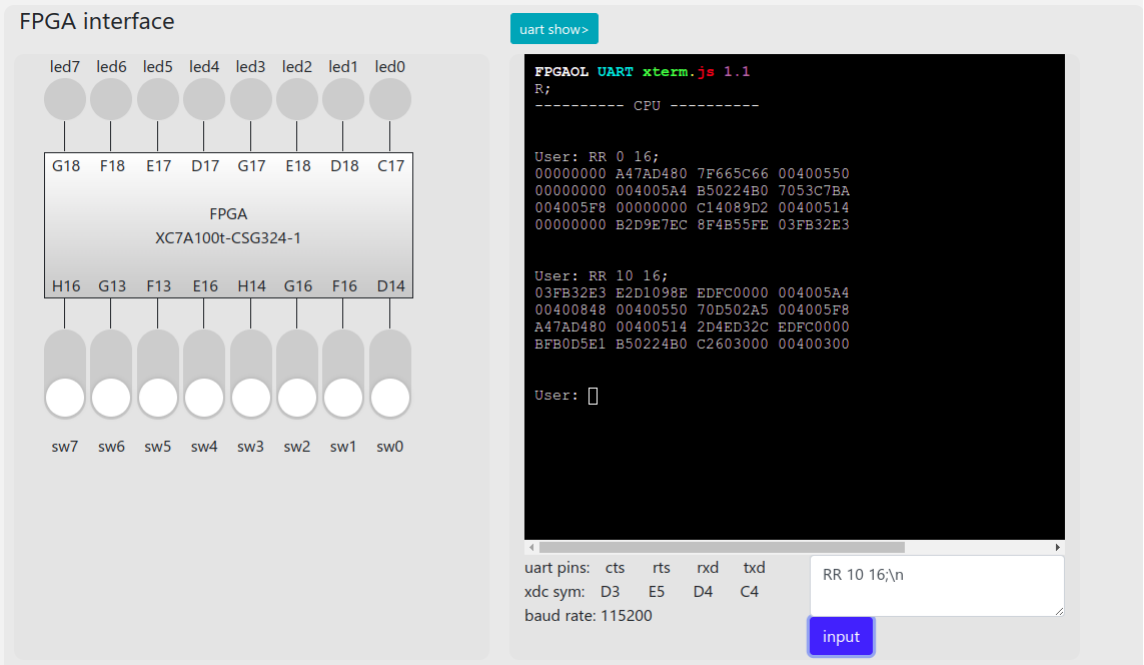

图 7: 测试程序 2 上板结果

上板结果与正确结果相吻合。

6 总结

本次实验设计了一个不考虑冒险的流水线 CPU，实现了写优先的寄存器堆和段间寄存器。