

Lab6_多周期 mips-cpu 设计

金泽文 PB15111604

实验目的：

设计多周期 mips-cpu。

实验内容：

- 设计 CPU ,完成以下程序代码的执行 ,其功能是起始数为 3 和 3 的斐波拉契数列的计算。只计算 20 个数。
- 实验设计中可以不使用给定的数据通路和状态机 , 但仅允许使用一个存储器。
- 对指令/数据存储器的附加要求：
 - 使用异步存储器 , 最高评分为√√
 - 使用同步存储器 , 最高评分为√√√ , 使用同步存储器时 , 需要对数据通路和状态机进行适当修改。

实验分析与设计：

我采用了同步存储器。针对 ppt 中给出的状态机，出于对时延的考虑，在 S3 与 S4 之间，S5 之后，S8 之后，分别加了一个等待状态。

Top 和 control 基本按照 ppt 中数据通路实现。需要指出的是，我省略了 2 位的 AluOp，直接使用了 3 位的 AluControl，除了 S8 对应的 beq 需要 “-” 运算，其他都是 “+” 运算。

对于 ALU 模块，不同于以往，需要设置 zero 输出变量。

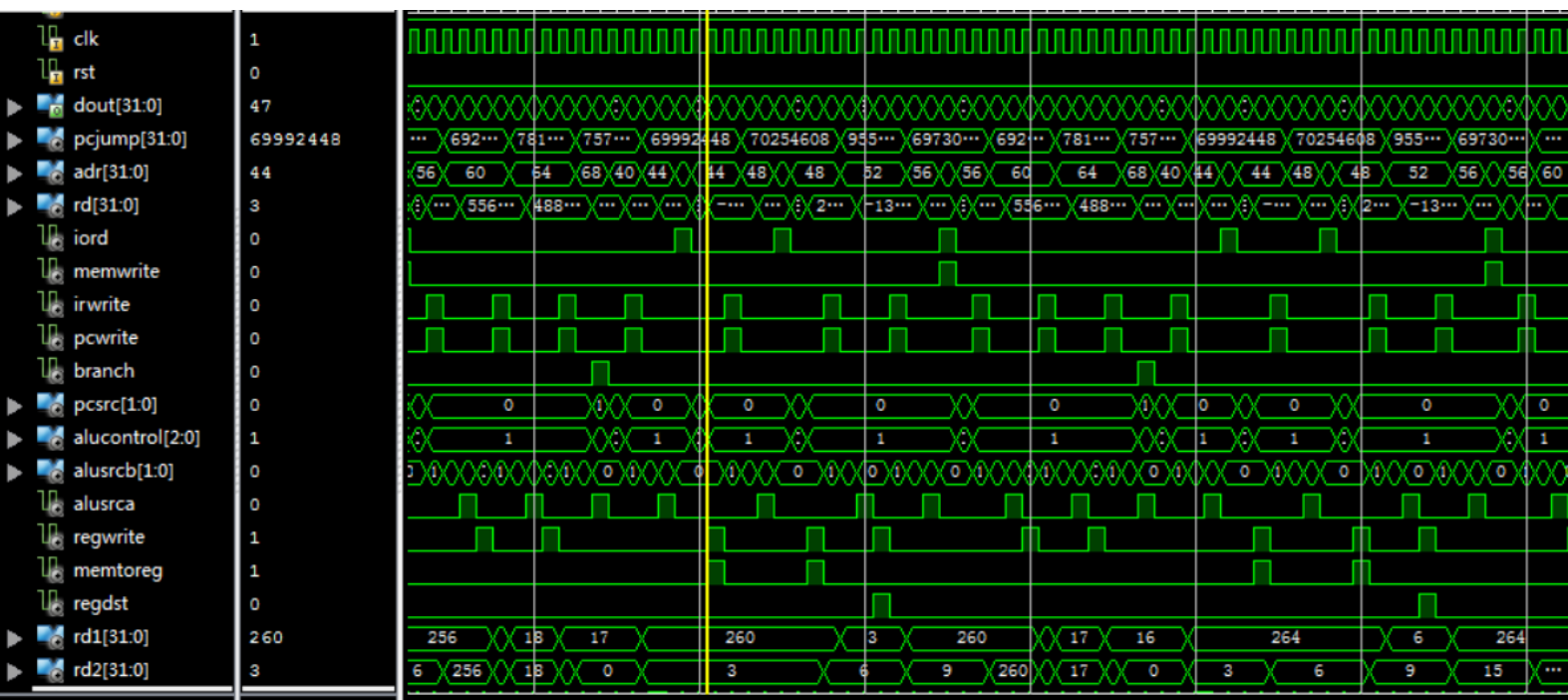
对于 regfile，沿用之前。

除此之外，需要注意的是，由于 mem 一个地址对应的是 4 个字节，所以在计算对应 mem 地址的时候需要考虑截断后两位。同时，为了方便仿真时的观察，将数据输出首地址改为 0x40，对应的 coe 需要修改四处：

20080100，200d0150，200b0154，200c0154

实验结果：

仿真波形：



内存：

0x30	0	0	0	0
0x34	0	0	0	0
0x38	0	0	0	0
0x3C	0	0	0	0
0x40	3	3	6	9
0x44	15	24	39	63
0x48	102	165	267	432
0x4C	699	1131	1830	2961
0x50	4791	7752	12543	20295
0x54	20	3	3	0
0x58	-	-	-	-

实验代码：

Top：

```
1 module top(
2     input      clk,
3     input      rst,
4     output [31:0] dout
5 );
6
7 reg [31:0] pc1;
8 reg [31:0] pc;
9 wire [31:0] pcjump;
10 wire [31:0] adr;
11 wire [31:0] rd;
12 reg [31:0] instr;
13 reg [31:0] data;
14 wire iord;
15 wire memwrite;
16 wire irwrite;
17 wire pcwrite;
18 wire branch;
19 wire [1:0] psrc;
20 wire [2:0] alucontrol;
21 wire [1:0] alusrcb;
22 wire alusrca;
23 wire regwrite;
24 wire memtoreg;
25 wire regdst;
26 wire [31:0] rd1;
27 wire [31:0] rd2;
28 reg [31:0] a;
29 reg [31:0] b;
30 wire [31:0] result;
31 wire [4:0] writereg;
32 reg [31:0] signimm;
33 wire [31:0] signimm12;
34 wire [31:0] srcb;
35 reg [31:0] srcb;
36 wire zero;
37 wire [31:0] aluresult;
38 reg [31:0] aluout;
39
40 always@(posedge clk or posedge rst)
41 begin
42     if(rst)
43         pc <= 32'b0;
44     else if(pcen)
45         pc <= pc1;
46     end
47
48     assign adr = iord ? aluout : pc;
49
50     always@(posedge clk)
51     begin
52         if(irwrite)
53             instr <= rd;
54         end
55
56     always@(posedge clk)
57     begin
58         data <= rd;
59     end
60
61     assign writereg = regdst ? instr[15:11] : instr[20:16];
62
63     assign result = memtoreg ? data :
64         aluout;
65
66     always@(*)
67     begin
68         if(instr[15])
69             signimm = 32'hffff0000 + instr
70                 [15:0];
71         else
72             signimm = 32'h00000000 + instr
73                 [15:0];
74         end
75
76     assign signimm12[31:2] = signimm[29:0];
77     assign signimm12[1:0] = 2'h0;
78
79     always@(posedge clk)
80     begin
81         a <= rd1;
82     end
83
84     always@(posedge clk)
85     begin
86         b <= rd2;
87     end
88
89     assign srcb = alusrcb ? a : pc;
90
91     always@(*)
92     begin
93         case(alusrcb)
94             2'b00: srcb = b;
95             2'b01: srcb = 32'h4;
96             2'b10: srcb = signimm;
97             2'b11: srcb = signimm12;
98         endcase
99     end
100
101     assign pcjump[31:28] = pc[31:28];
102     assign pcjump[27:2] = instr[25:0];
103     assign pcjump[1:0] = 2'h0;
104
105     assign pcen = pcwrite | (branch & zero
106         );
107
108     always@(posedge clk)
109     begin
110         aluout <= aluresult;
111     end
112
113     always@(*)
114     begin
115         case(psrc)
116             2'b00: pc1 = aluresult;
117             2'b01: pc1 = aluout;
118             2'b10: pc1 = pcjump;
119             default: pc1 = 32'h0;
120         endcase
121     end
122
123     assign dout = aluresult;
124
125     control u_control(
126         .opcode (instr[31:26]),
127         .funct (instr[5:0]),
128         .clk (clk)
129     );
130
131     .rst (rst)
132     .iord (iord)
133     .memwrite (memwrite)
134     .irwrite (irwrite)
135     .pcwrite (pcwrite)
136     .branch (branch)
137     .psrc (psrc)
138     .alucontrol (alucontrol)
139     .alusrcb (alusrcb)
140     .alusrca (alusrca)
141     .regwrite (regwrite)
142     .memtoreg (memtoreg)
143     .regdst (regdst)
144 );
145
146 mem u_mem(
147     .clk (clk)
148     .wea (memwrite)
149     .addra (adr[8:2])
150     .dina (b[31:0])
151     .douta (rd)
152 );
153
154 REG_FILE u_REG_FILE(
155     .clk (clk)
156     .r1_addr (instr[25:21])
157     .r2_addr (instr[20:16])
158     .r3_addr (writereg)
159     .r3_din (result)
160     .r3_wr (regwrite)
161     .r1_dout (rd1)
162     .r2_dout (rd2)
163 );
164
165 alu u_alu(
166     .alu_a (srcb)
167     .alu_b (srcb)
168     .alu_op (alucontrol)
169     .alu_out (aluresult)
170     .alu_zero (zero)
171 );
172 endmodule
```

Reg_file

```
1 module REG_FILE(  
2   input clk,  
3   input rst,  
4   input [4:0] r1_addr,  
5   input [4:0] r2_addr,  
6   input [4:0] r3_addr,  
7   input [31:0] r3_din,  
8   input r3_wr,  
9   output reg [31:0] r1_dout,  
10  output reg [31:0] r2_dout  
11 );  
12 reg [31:0] data [31:0];  
13 always@(posedge clk or  
14   posedge rst)  
15 begin  
16   if (rst) begin  
17     data[0] <= 32'b1;  
18     data[1] <= 32'b1;  
19     data[2] <= 32'b1;  
20     data[3] <= 32'b1;  
21     data[4] <= 32'b1;  
22     data[5] <= 32'b1;  
23     data[6] <= 32'b1;  
24     data[7] <= 32'b1;  
25     data[8] <= 32'b1;  
26     data[9] <= 32'b1;  
27     data[10] <= 32'b1;  
28     data[11] <= 32'b1;  
29     data[12] <= 32'b1;  
30     data[13] <= 32'b1;  
31     data[14] <= 32'b1;  
32     data[15] <= 32'b1;  
33     data[16] <= 32'b1;  
34     data[17] <= 32'b1;  
35     data[18] <= 32'b1;  
36     data[19] <= 32'b1;  
37     data[20] <= 32'b1;  
38     data[21] <= 32'b1;  
39     data[22] <= 32'b1;  
40     data[23] <= 32'b1;  
41     data[24] <= 32'b1;  
42     data[25] <= 32'b1;  
43     data[26] <= 32'b1;  
44     data[27] <= 32'b1;  
45     data[28] <= 32'b1;  
46     data[29] <= 32'b1;  
47     data[30] <= 32'b1;  
48     data[31] <= 32'b1;  
49   end  
50   else if(r3_wr)  
51     data[r3_addr] <= r3_din;  
52   end  
53   always@(*)  
54   begin  
55     if(r1_addr)  
56       r1_dout = data[r1_addr];  
57     else  
58       r1_dout = 32'h0;  
59   end  
60   always@(*)begin  
61     if(r2_addr)  
62       r2_dout = data[r2_addr];  
63     else  
64       r2_dout = 32'h0;  
65   end  
66 end  
67 endmodule  
68
```

Alu :

```
1  module alu(  
2  input  signed  [31:0]  alu_a,  
3  input  signed  [31:0]  alu_b,  
4  input          [2:0]   alu_op,  
5  output reg      [31:0]  alu_out,  
6  output reg      [31:0]  alu_zero  
7  );  
8  
9  always@(*)  
10 begin  
11     case(alu_op)  
12         3'h00: alu_out = 32'h0;  
13         3'h01: alu_out = alu_a + alu_b;  
14         3'h02: alu_out = alu_a - alu_b;  
15         3'h03: alu_out = alu_a & alu_b;  
16         3'h04: alu_out = alu_a | alu_b;  
17         3'h05: alu_out = alu_a ^ alu_b;  
18         3'h06: alu_out = ~(alu_a |  
19                 alu_b);  
19         default: alu_out = 32'h0;  
20     endcase  
21 end  
22  
23 always@(*)  
24 begin  
25     if(alu_out > 0)  
26         alu_zero = 1'h1;  
27     else  
28         alu_zero = 1'h0;  
29     end  
30  
31 endmodule  
32
```

control

```
1 module control(  
2     input [5:0] opcode,  
3     input [5:0] funct,  
4     input clk,  
5     input rst,  
6     output reg iord,  
7     output reg memwrite,  
8     output reg irwrite,  
9     output reg pcwrite,  
10    output reg branch,  
11    output reg [1:0] psrc,  
12    output reg [2:0] alucontrol,  
13    output reg [1:0] alusrcb,  
14    output reg alusrcra,  
15    output reg regwrite,  
16    output reg memtoreg,  
17    output reg regdst  
18 );  
19  
20 reg [3:0] curr_state;  
21 reg [3:0] next_state;  
22  
23 always@(posedge clk or posedge rst)  
24 begin  
25     if(rst)  
26         curr_state <= 4'h0;  
27     else  
28         curr_state <= next_state;  
29     end  
30  
31 always@(*)  
32 begin  
33     case(curr_state)  
34         4'd0:  
35             next_state = 4'd1;  
36         4'd1:  
37             begin  
38                 case(opcode)  
39                     6'b100011: next_state = 4'd2;  
40                     6'b101011: next_state = 4'd2;  
41                     6'b000000: next_state = 4'd6;  
42                     6'b000111: next_state = 4'd8;  
43                     6'b001000: next_state = 4'd9;  
44                     6'b000010: next_state = 4'd11;  
45                     default: next_state = 4'd0;  
46                 endcase  
47             end  
48         4'd2:  
49             begin  
50                 case(opcode)  
51                     6'b100011: next_state =  
52  
53                     6'b101011: next_state = 4'd5;  
54                     default: next_state = 4'd0;  
55                 endcase  
56             end  
57         4'd3: next_state = 4'd12;  
58         4'd4: next_state = 4'd0;  
59         4'd5: next_state = 4'd13;  
60         4'd6: next_state = 4'd7;  
61         4'd7: next_state = 4'd0;  
62         4'd8: next_state = 4'd15;  
63         4'd9: next_state = 4'd10;  
64         4'd10: next_state = 4'd0;  
65         4'd11: next_state = 4'd4;  
66         4'd12: next_state = 4'd4;  
67         4'd13: next_state = 4'd0;  
68         4'd14: next_state = 4'd0;  
69         4'd15: next_state = 4'd0;  
70         default: next_state = 4'd0;  
71     endcase  
72 end  
73  
74 always@(*)  
75 begin  
76     case(curr_state)  
77         4'd0: begin  
78             iord = 1'b0;  
79             memwrite = 1'b0;  
80             irwrite = 1'b1;  
81             pcwrite = 1'b1;  
82             branch = 1'b0;  
83             psrc = 2'b0;  
84             alucontrol = 3'b001;  
85             alusrcb = 2'b01;  
86             alusrcra = 1'b0;  
87             regwrite = 1'b0;  
88             memtoreg = 1'b0;  
89             regdst = 1'b0;  
90         end  
91         4'd1: begin  
92             iord = 1'b0;  
93             memwrite = 1'b0;  
94             irwrite = 1'b0;  
95             pcwrite = 1'b0;  
96             branch = 1'b0;  
97             psrc = 2'b0;  
98             alucontrol = 3'b001;  
99             alusrcb = 2'b11;  
100            alusrcra = 1'b0;  
101            regwrite = 1'b0;  
102            memtoreg = 1'b0;  
103            regdst = 1'b0;  
104        end  
105         4'd2: begin  
106             iord = 1'b0;  
107             memwrite = 1'b0;  
108             irwrite = 1'b0;  
109             pcwrite = 1'b0;  
110             branch = 1'b0;  
111             psrc = 2'b0;  
112             alucontrol = 3'b001;  
113             alusrcb = 2'b10;  
114             alusrcra = 1'b1;  
115             regwrite = 1'b0;  
116             memtoreg = 1'b0;  
117             regdst = 1'b0;  
118         end  
109         4'd3: begin  
110             iord = 1'b1;  
111             memwrite = 1'b0;  
112             irwrite = 1'b0;  
113             pcwrite = 1'b0;  
114             branch = 1'b0;  
115             psrc = 2'b0;  
116             alucontrol = 3'b001;  
117             alusrcb = 2'b0;  
118             alusrcra = 1'b0;  
119             regwrite = 1'b0;  
120             memtoreg = 1'b0;  
121             regdst = 1'b0;  
122         end  
123         4'd4: begin  
124             iord = 1'b0;  
125             memwrite = 1'b0;  
126             irwrite = 1'b0;  
127             pcwrite = 1'b0;  
128             branch = 1'b0;  
129             psrc = 2'b0;  
130             alucontrol = 3'b001;  
131             alusrcb = 2'b0;  
132             alusrcra = 1'b0;  
133             regwrite = 1'b0;  
134             memtoreg = 1'b0;  
135             regdst = 1'b0;  
136         end  
137         4'd5: begin  
138             iord = 1'b1;  
139             memwrite = 1'b1;  
140             irwrite = 1'b0;  
141             pcwrite = 1'b0;  
142             branch = 1'b0;  
143             psrc = 2'b0;  
144             alucontrol = 3'b001;  
145             alusrcb = 2'b0;  
146             alusrcra = 1'b0;  
147             regwrite = 1'b1;  
148             memtoreg = 1'b1;  
149             regdst = 1'b0;  
150         end  
151         4'd6: begin  
152             iord = 1'b1;  
153             memwrite = 1'b1;  
154             irwrite = 1'b0;  
155             pcwrite = 1'b0;  
156             branch = 1'b0;  
157             psrc = 2'b0;  
158             alucontrol = 3'b001;  
159             alusrcb = 2'b0;  
160             alusrcra = 1'b0;  
161             regwrite = 1'b0;  
162             memtoreg = 1'b0;  
163             regdst = 1'b0;  
164         end  
165         4'd7: begin  
166             iord = 1'b0;  
167             memwrite = 1'b0;  
168             irwrite = 1'b0;  
169             pcwrite = 1'b0;  
170             branch = 1'b0;  
171             psrc = 2'b0;  
172             alucontrol = 3'b001;  
173             alusrcb = 2'b0;  
174             alusrcra = 1'b1;  
175             regwrite = 1'b0;  
176             memtoreg = 1'b0;  
177             regdst = 1'b1;  
178         end  
179         4'd8: begin  
180             iord = 1'b0;  
181             memwrite = 1'b0;  
182             irwrite = 1'b0;  
183             pcwrite = 1'b0;  
184             branch = 1'b0;  
185             psrc = 2'b0;  
186             alucontrol = 3'b001;  
187             alusrcb = 2'b0;  
188             alusrcra = 1'b0;  
189             regwrite = 1'b0;  
190             memtoreg = 1'b0;  
191             regdst = 1'b0;  
192         end  
193         4'd9: begin  
194             iord = 1'b0;  
195             memwrite = 1'b0;  
196             irwrite = 1'b0;  
197             pcwrite = 1'b0;  
198             branch = 1'b0;  
199             psrc = 2'b0;  
200             alucontrol = 3'b001;  
201             alusrcb = 2'b0;  
202             alusrcra = 1'b0;  
203             regwrite = 1'b0;  
204             memtoreg = 1'b0;  
205             regdst = 1'b0;  
206         end  
207         4'd10: begin  
208             iord = 1'b0;  
209             memwrite = 1'b0;  
210             irwrite = 1'b0;  
211             pcwrite = 1'b0;  
212             branch = 1'b0;  
213             psrc = 2'b0;  
214             alucontrol = 3'b001;  
215             alusrcb = 2'b0;  
216             alusrcra = 1'b0;  
217             regwrite = 1'b0;  
218             memtoreg = 1'b0;  
219             regdst = 1'b0;  
220         end  
221         4'd11: begin  
222             iord = 1'b0;  
223             memwrite = 1'b0;  
224             irwrite = 1'b0;  
225             pcwrite = 1'b0;  
226             branch = 1'b0;  
227             psrc = 2'b0;  
228             alucontrol = 3'b001;  
229             alusrcb = 2'b0;  
230             alusrcra = 1'b0;  
231             regwrite = 1'b0;  
232             memtoreg = 1'b0;  
233             regdst = 1'b0;  
234         end  
235         4'd12: begin  
236             iord = 1'b0;  
237             memwrite = 1'b0;  
238             irwrite = 1'b0;  
239             pcwrite = 1'b0;  
240             branch = 1'b0;  
241             psrc = 2'b0;  
242             alucontrol = 3'b001;  
243             alusrcb = 2'b0;  
244             alusrcra = 1'b0;  
245             regwrite = 1'b0;  
246             memtoreg = 1'b0;  
247             regdst = 1'b0;  
248         end  
249         4'd13: begin  
250             iord = 1'b0;  
251             memwrite = 1'b0;  
252             irwrite = 1'b0;  
253             pcwrite = 1'b0;  
254             branch = 1'b0;  
255             psrc = 2'b0;  
256             alucontrol = 3'b001;  
257             alusrcb = 2'b0;  
258             alusrcra = 1'b0;  
259             regwrite = 1'b0;  
260             memtoreg = 1'b0;  
261             regdst = 1'b0;  
262         end  
263         4'd14: begin  
264             iord = 1'b0;  
265             memwrite = 1'b0;  
266             irwrite = 1'b0;  
267             pcwrite = 1'b0;  
268             branch = 1'b0;  
269             psrc = 2'b0;  
270             alucontrol = 3'b001;  
271             alusrcb = 2'b0;  
272             alusrcra = 1'b0;  
273             regwrite = 1'b0;  
274             memtoreg = 1'b0;  
275             regdst = 1'b0;  
276         end  
277         4'd15: begin  
278             iord = 1'b0;  
279             memwrite = 1'b0;  
280             irwrite = 1'b0;  
281             pcwrite = 1'b0;  
282             branch = 1'b0;  
283             psrc = 2'b0;  
284             alucontrol = 3'b001;  
285             alusrcb = 2'b0;  
286             alusrcra = 1'b0;  
287             regwrite = 1'b0;  
288             memtoreg = 1'b0;  
289             regdst = 1'b0;  
290         end  
291     endcase  
292 end  
293 endmodule
```