

Lab5 report

实验目的和内容

理解流水线CPU的结构和工作原理

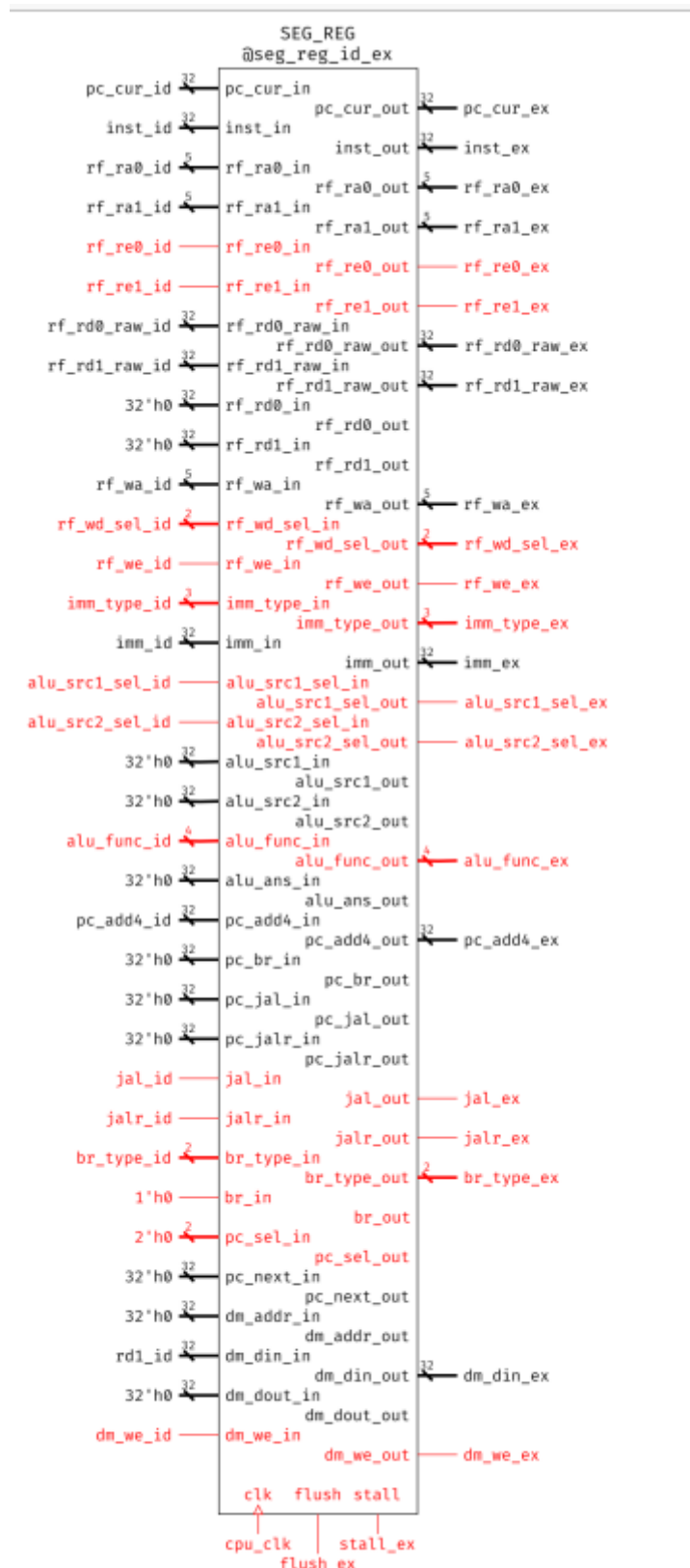
掌握流水线CPU的设计和调试方法，特别是流水线中数据相关和控制相关的处理

熟练掌握数据通路和控制器的设计和描述方法

逻辑设计

段间寄存器

CPU设计分为五个阶段if,id,ex,mem,wb.每个阶段之间有段间寄存器保存每个阶段的数据和控制信号.每个段间寄存器都是相同的.



Hazard模块

当以下四个条件满足时,前递信号生效:

1. MEM/WB 段写使能为 1;
2. EX 段某寄存器读使能非零;
3. 上述寄存器读地址等于 MEM/WB 段的写地址;
4. 若为 MEM 段, 写回的数并非数据存储器读取结果。

如果mem端的前递信号生效,就根据rf_wd_sel_mem确定前递数据为类型,如果wb前递信号生效,前递数据为rf_wd_wb.

如果前面1,2,3满足,4不满足说明是load-use情况,需要stall一周期.

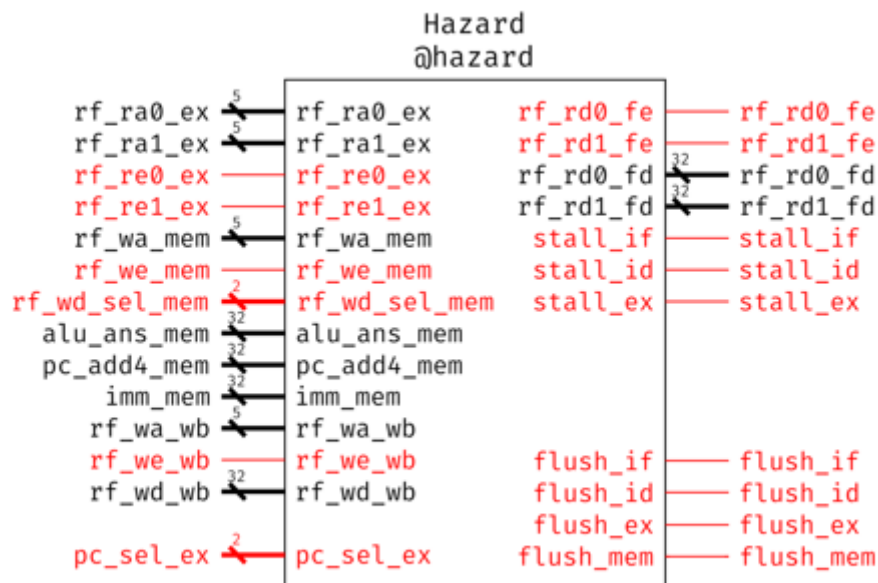
如果pc写回内容不是pc+4说明发生跳转,id,ex端清空.

```
1  module Hazard (
2      input [4:0] rf_ra0_ex,
3      input [4:0] rf_ra1_ex,
4      input rf_re0_ex,
5      input rf_re1_ex,
6      input [4:0] rf_wa_mem,
7      input rf_we_mem,
8      input [1:0] rf_wd_sel_mem,
9      input [31:0] alu_ans_mem,
10     input [31:0] pc_add4_mem,
11     input [31:0] imm_mem,
12     input [4:0] rf_wa_wb,
13     input rf_we_wb,
14     input [31:0] rf_wd_wb,
15     input [1:0] pc_sel_ex,
16     output reg rf_rd0_fe,
17     output reg rf_rd1_fe,
18     output reg [31:0] rf_rd0_fd,
19     output reg [31:0] rf_rd1_fd,
20     output reg stall_if,
21     output reg stall_id,
22     output reg stall_ex,
23     output reg flush_id,
24     output reg flush_ex,
25     output reg flush_mem
26 );
27     always @(*) begin
28         if(rf_re0_ex==1&&
29 ((rf_ra0_ex==rf_wa_mem&&rf_we_mem==1&&rf_wd_sel_mem!=2'b10)||
30 (rf_ra0_ex==rf_wa_wb&&rf_we_wb==1)))
31             rf_rd0_fe=1;
32         else
33             rf_rd0_fe=0;
34         if(rf_re1_ex==1&&
35 ((rf_ra1_ex==rf_wa_mem&&rf_we_mem==1&&rf_wd_sel_mem!=2'b10)||
36 (rf_ra1_ex==rf_wa_wb&&rf_we_wb==1)))
37             rf_rd1_fe=1;
38         else
39             rf_rd1_fe=0;
40         if(rf_re0_ex==1&&
41 (rf_ra0_ex==rf_wa_mem&&rf_we_mem==1&&rf_wd_sel_mem!=2'b10))
42             begin
43                 case(rf_wd_sel_mem)
44                     2'b00: rf_rd0_fd=alu_ans_mem;
45                     2'b01: rf_rd0_fd=pc_add4_mem;
46                     2'b11: rf_rd0_fd=imm_mem;
47                 endcase
48             end
49         else if(rf_we_wb)
50             begin
51                 rf_rd0_fd=rf_wd_wb;
```

```

47         end
48         if(rf_re1_ex==1&&
(rf_ra1_ex==rf_wa_mem&&rf_we_mem==1&&rf_wd_sel_mem!=2'b10))
49             begin
50                 case(rf_wd_sel_mem)
51                     2'b00:rf_rd1_fd=alu_ans_mem;
52                     2'b01:rf_rd1_fd=pc_add4_mem;
53                     2'b11:rf_rd1_fd=imm_mem;
54                 endcase
55             end
56
57         if((rf_re0_ex==1&&rf_ra0_ex==rf_wa_mem&&rf_we_mem==1&&rf_wd_sel_mem==2'b10)
|| (rf_re1_ex==1&&rf_ra1_ex==rf_wa_mem&&rf_we_mem==1&&rf_wd_sel_mem==2'b10))
58             begin
59                 stall_ex=1;
60                 stall_id=1;
61                 stall_if=1;
62                 flush_mem=1;
63             end
64         else
65             begin
66                 stall_if=0;
67                 stall_id=0;
68                 stall_ex=0;
69                 flush_mem=0;
70             end
71         if(pc_sel_ex!=2'b00)
72             begin
73                 flush_id=1;
74                 flush_ex=1;
75             end
76         else
77             begin
78                 flush_id=0;
79                 flush_ex=0;
80             end
81
82
83
84     end
85 endmodule
86

```



寄存器模块

寄存器堆模块遵循写优先原则,如果读寄存器和写寄存器相同,则读出数据与写数据相同.

```

1  module RF (
2      input clk,
3      input we,
4      input [4:0]ra0,
5      input [4:0]ra1,
6      input [4:0]wa,
7      input [31:0]wd,
8      input [4:0]ra_dbg,
9      output reg[31:0]rd0,
10     output reg[31:0]rd1,
11     output reg[31:0]rd_dbg
12 );
13 reg [31:0] regfile[31:0];
14 integer i;
15 initial begin
16     for(i=0;i<32;i++)    regfile[i]=0;
17 end
18 always@(posedge clk)
19 begin
20     if(we&&wa!=0)
21         regfile[wa]<=wd;
22     else
23         regfile[wa]<=regfile[wa];
24 end
25 always @(*) begin
26     if(we&&wa==rd0)
27         rd0=wd;
28     else
29         rd0=regfile[ra0];
30     if(we&&wa==rd1)

```

```

31     rd1=wd;
32     else
33         rd1=regfile[ra1];
34         if(we&&wa==rd0)
35             rd0=wd;
36         else
37             rd0=regfile[ra0];
38
39     end
40     assign rd0=(we&&wa==rd0)?wd:regfile[ra0];
41     assign rd1=(we&&wa==rd1)?wd:regfile[ra1];
42     assign rd_dbg=(we&&wa==rd_dbg)?wd:regfile[ra_dbg];
43 endmodule

```

ctrl模块

ctrl模块与lab4的ctrl模块基本相同,只是多了rf_re信号,当需要读寄存器时, 读使能有效.

```

1  module CTRL(
2      input  [31:0] inst,
3      output reg rf_re0,
4      output reg rf_re1,
5      output reg jal,
6      output reg jalr,
7      output reg [1:0] br_type,
8      output reg wb_en,
9      output reg [1:0] wb_sel,
10     output reg alu_op1_sel,
11     output reg alu_op2_sel,
12     output reg [3:0] alu_ctrl,
13     output reg [2:0] imm_type,
14     output reg mem_we
15 );
16 wire [6:0] opcode;
17 reg [5:0] type;
18 wire [2:0] func;
19 assign func=inst[14:12];
20 assign opcode=inst[6:0];
21 always@(*)
22 begin
23     case(type)
24         6'b000001,
25         6'b000010,
26         6'b000100,
27         6'b001000:
28         begin
29             if(inst[19:15] != 5'h0)
30                 rf_re0=1;
31             else
32                 rf_re0=0;
33         end
34         default: rf_re0=0;
35     endcase
36 end
37 always @(*) begin

```

```

38     case(type)
39     6'b000001,
40     6'b000100,
41     6'b001000:
42     begin
43         if(inst[24:20] != 5'h0)
44             rf_re1=1;
45         else
46             rf_re1=0;
47     end
48     default: rf_re1=0;
49     endcase
50 end
51 always @(*) begin
52     case (opcode)
53     7'b0110011: type = 6'b000001; // R-type
54     7'b1100111,
55     7'b0000011,
56     7'b0010011: type = 6'b000010; // I-type
57     7'b1100011: type = 6'b000100; // B-type
58     7'b0100011: type = 6'b001000; // S-type
59     7'b1101111: type = 6'b010000; // J-type
60     7'b0110111,
61     7'b0010111: type = 6'b100000; // U-type
62     default:   type = 6'b000000;
63     endcase
64     end
65     always @(*) begin
66         case (type)
67         6'b000010: imm_type = 3'b001; // I-type
68         6'b000100: imm_type = 3'b010; // B-type
69         6'b001000: imm_type = 3'b011; // S-type
70         6'b010000: imm_type = 3'b100; // J-type
71         6'b100000: imm_type = 3'b101; // U-type
72         default:  imm_type = 3'b000;
73         endcase
74     end
75
76     always @(*) begin
77         if(type==6'b000100)
78             case(inst[14:12])
79             3'b000: br_type=2'b01; //beq
80             3'b100: br_type=2'b10; //b1t
81             default: br_type=2'b00;
82             endcase
83         else
84             br_type=2'b00;
85     end
86
87     always @(*) begin
88         case(opcode)
89         7'b0110011,
90         7'b0010011,
91         7'b0010111: wb_sel=2'b00;
92         7'b1101111,
93         7'b1100111: wb_sel=2'b01;
94         7'b0000011: wb_sel=2'b10;
95         7'b0110111: wb_sel=2'b11;

```

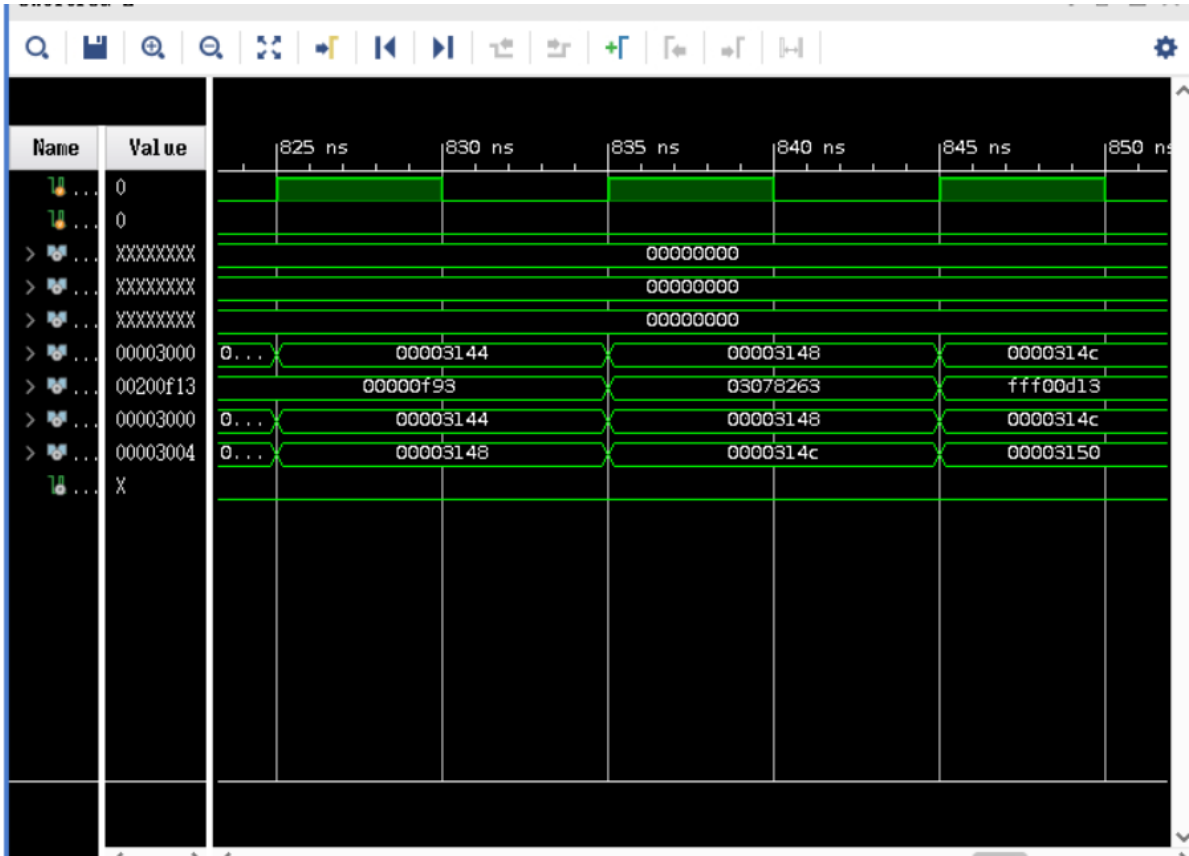
```

96     default:wb_sel=2'b00;
97     endcase
98 end
99 always @(*)
100 begin
101     if(opcode==7'b1101111)
102         jal=1;
103     else
104         jal=0;
105     if(opcode==7'b1100111)
106         jalr=1;
107     else
108         jalr=0;
109 end
110 always@(*)
111 begin
112     case(type)
113         6'b000001,
114         6'b000010,
115         6'b100000,
116         6'b010000:
117         begin
118             if(inst[11:7] != 5'h0)
119                 wb_en=1;
120             else
121                 wb_en=0;
122             end
123         default:wb_en=0;
124         endcase
125     end
126     always @(*) begin
127         case(type)
128             6'b000001,
129             6'b000010,
130             6'b001000:alu_op1_sel=0;
131             default:alu_op1_sel=1;
132         endcase
133     end
134     always @(*) begin
135         case(type)
136             6'b000001:alu_op2_sel=0;
137             default:alu_op2_sel=1;
138         endcase
139     end
140     always @(*) begin
141         case(type)
142             6'b001000:mem_we=1;
143             default:mem_we=0;
144         endcase
145     end
146     always @(*) begin
147         alu_ctrl=4'h0;
148     end
149 endmodule

```

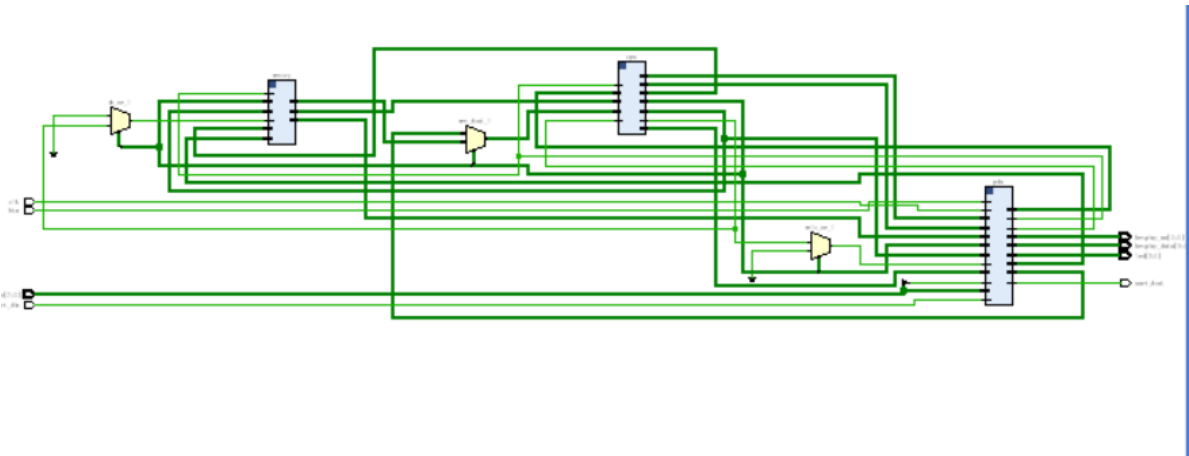

仿真结果与分析

可以看到仿真程序可以正常运行

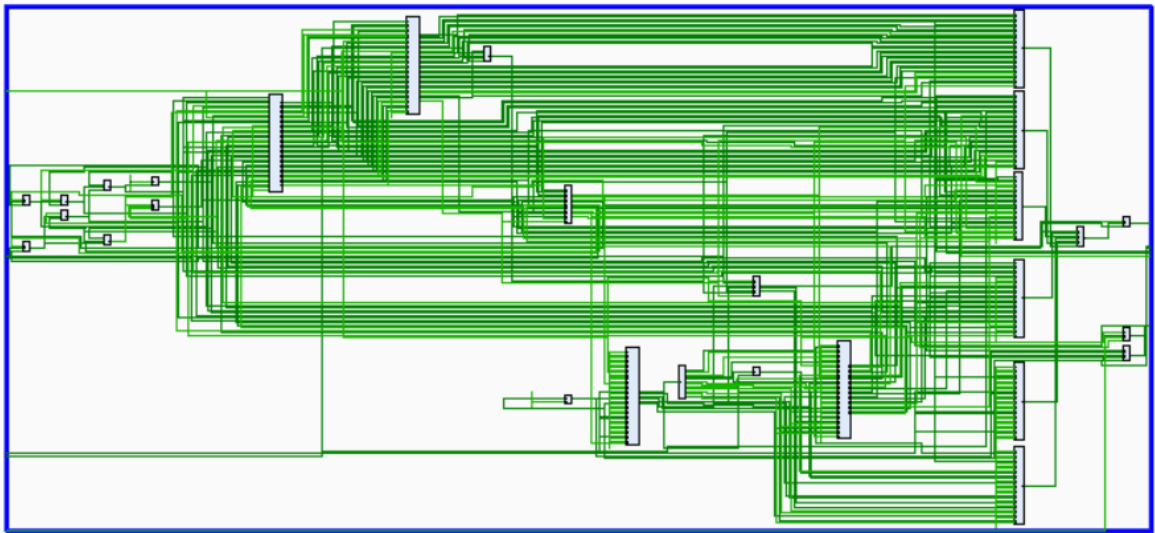


电路设计与分析

RTL总数据通路

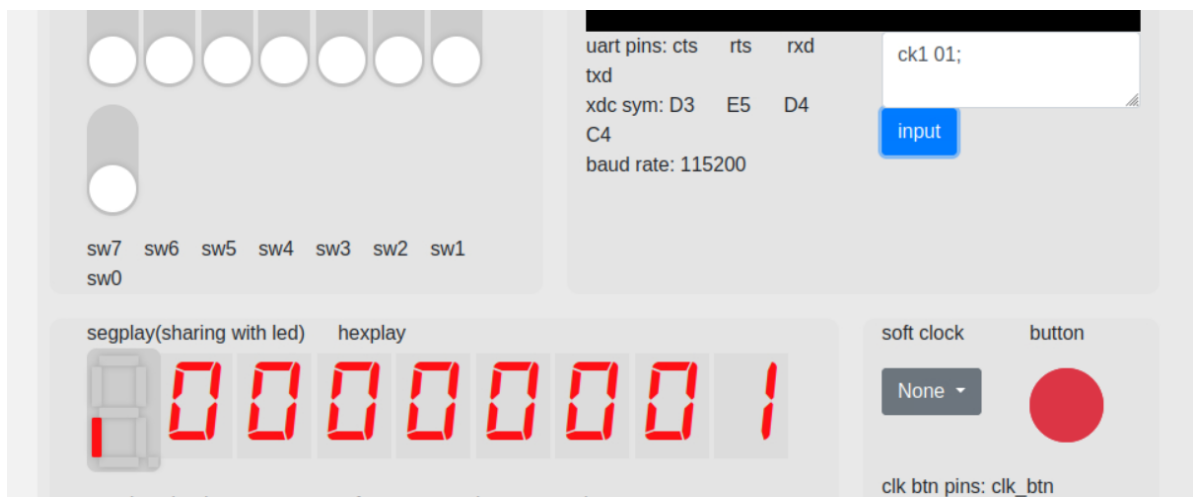


RTL cpu数据通路



测试结果与分析

先将bp点设为31f4进行第一部分测试,然后依次查看对应寄存器堆



sw7 sw6 sw5 sw4 sw3 sw2 sw1 sw0

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

ck1 03;
input

segplay(sharing with led) hexplay

0

0

0

0

0

0

0

3


segplay pin: dot seg_g seg_f seg_e seg_d seg_c seg_b seg_a
xdc.ucf sym: G18 F18 E17 D17 G17 E18 D18 C17

sw7 sw6 sw5 sw4 sw3 sw2 sw1 sw0

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

ck1 04;
input

soft clock button

None ▾ 

clk btn pins: clk_btn
xdc,ucf sym: B18

segplay(sharing with led) hexplay

0

0

0

0

0

0

0

4

segplay pin: dot seg_g seg_f seg_e seg_d seg_c seg_b seg_a

sw7 sw6 sw5 sw4 sw3 sw2 sw1 sw0

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

ck1 05;
input

soft clock button

None ▾ 

clk btn pins: clk_btn
xdc,ucf sym: B18

segplay(sharing with led) hexplay

0

0

0

0

0

0

0

5

segplay pin: dot seg_g seg_f seg_e seg_d seg_c seg_b seg_a
xdc.ucf sym: G18 F18 E17 D17 G17 E18 D18 C17

sw7 sw6 sw5 sw4 sw3 sw2 sw1 sw0

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

ck1 05;
input

soft clock button

None ▾ 

clk btn pins: clk_btn
xdc,ucf sym: B18

sw7 sw6 sw5 sw4 sw3 sw2 sw1 sw0

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

ck1 06;

input

segplay(sharing with led) hexplay

00000006

segplay pin: dot seg_g seg_f seg_e seg_d seg_c seg_b seg_a xdc,ucf sym: G18 F18 E17 D17 G17 E18 D18 C17 hexplay pin: an2 an1 an0 d3 d2 d1 d0 xdc,ucf sym: A18 B16 B17 A15 A16 A13 A14

sw7 sw6 sw5 sw4 sw3 sw2 sw1 sw0

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

ck1 07;

input

soft clock button

None

clk btn pins: clk_btn xdc,ucf sym: B18

segplay(sharing with led) hexplay

00000007

segplay pin: dot seg_g seg_f seg_e seg_d seg_c seg_b seg_a

sw7 sw6 sw5 sw4 sw3 sw2 sw1 sw0

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

ck1 08;

input

soft clock button

None

clk btn pins: clk_btn xdc,ucf sym: B18

segplay(sharing with led) hexplay

00000008

segplay pin: dot seg_g seg_f seg_e seg_d seg_c seg_b seg_a

sw7 sw6 sw5 sw4 sw3 sw2 sw1 sw0

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

ck1 09;

input

soft clock button

None

clk btn pins: clk_btn xdc,ucf sym: B18

segplay(sharing with led) hexplay

00000009

segplay pin: dot seg_g seg_f seg_e seg_d seg_c seg_b seg_a

sw7 sw6 sw5 sw4 sw3 sw2 sw1 sw0

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

ck1 09;

input

soft clock button

None

clk btn pins: clk_btn xdc,ucf sym: B18

sw7 sw6 sw5 sw4 sw3 sw2 sw1 sw0

segplay pin: dot seg_g seg_f seg_e seg_d seg_c seg_b seg_a
xdc,ucf sym: G18 F18 E17 D17 G17 E18 D18 C17
hexplay pin: an2 an1 an0 d3 d2 d1 d0
clk btn pins: clk_btn
xdc,ucf sym: B18

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

ck1 0a;
input

sw7 sw6 sw5 sw4 sw3 sw2 sw1 sw0

segplay(sharing with led) hexplay

segplay pin: dot seg_g seg_f seg_e seg_d seg_c seg_b seg_a
xdc,ucf sym: G18 F18 E17 D17 G17 E18 D18 C17
hexplay pin: an2 an1 an0 d3 d2 d1 d0
clk btn pins: clk_btn
xdc,ucf sym: B18

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

ck1 0b;
input

soft clock button

None

clk btn pins: clk_btn
xdc,ucf sym: B18

sw7 sw6 sw5 sw4 sw3 sw2 sw1 sw0

segplay(sharing with led) hexplay

segplay pin: dot seg_g seg_f seg_e seg_d seg_c seg_b seg_a
xdc,ucf sym: G18 F18 E17 D17 G17 E18 D18 C17
hexplay pin: an2 an1 an0 d3 d2 d1 d0
clk btn pins: clk_btn
xdc,ucf sym: B18

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

ck1 0c;
input

soft clock button

None

clk btn pins: clk_btn
xdc,ucf sym: B18

sw7 sw6 sw5 sw4 sw3 sw2 sw1 sw0

segplay(sharing with led) hexplay

segplay pin: dot seg_g seg_f seg_e seg_d seg_c seg_b seg_a
xdc,ucf sym: G18 F18 E17 D17 G17 E18 D18 C17
hexplay pin: an2 an1 an0 d3 d2 d1 d0
clk btn pins: clk_btn
xdc,ucf sym: B18

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

ck1 0d;
input

soft clock button

None

clk btn pins: clk_btn
xdc,ucf sym: B18

sw7

sw6

sw5

sw4

sw3

sw2

sw1

sw0

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

ck1 0d;

input

segplay(sharing with led) hexplay

0

0

0

0

0

0

0

d

segplay pin: dot seg_g seg_f seg_e seg_d seg_c seg_b seg_a
xdc,ucf sym: G18 F18 E17 D17 G17 E18 D18 C17
hexplay pin: an2 an1 an0 d3 d2 d1 d0

seg_g seg_f seg_e seg_d seg_c seg_b seg_a

soft clock button

None

clk btn pins: clk_btn
xdc,ucf sym: B18

sw7

sw6

sw5

sw4

sw3

sw2

sw1

sw0

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

ck1 0e;

input

segplay(sharing with led) hexplay

0

0

0

0

0

0

0

E

segplay pin: dot seg_g seg_f seg_e seg_d seg_c seg_b seg_a
xdc,ucf sym: G18 F18 E17 D17 G17 E18 D18 C17
hexplay pin: an2 an1 an0 d3 d2 d1 d0

seg_g seg_f seg_e seg_d seg_c seg_b seg_a

soft clock button

None

clk btn pins: clk_btn
xdc,ucf sym: B18

sw7

sw6

sw5

sw4

sw3

sw2

sw1

sw0

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

ck1 0f;

input

segplay(sharing with led) hexplay

0

0

0

0

0

0

0

F

segplay pin: dot seg_g seg_f seg_e seg_d seg_c seg_b seg_a
xdc,ucf sym: G18 F18 E17 D17 G17 E18 D18 C17
hexplay pin: an2 an1 an0 d3 d2 d1 d0

seg_g seg_f seg_e seg_d seg_c seg_b seg_a

soft clock button

None

clk btn pins: clk_btn
xdc,ucf sym: B18

sw7

sw6

sw5

sw4

sw3

sw2

sw1

sw0

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

ck1 10;

input

segplay(sharing with led) hexplay

0

0

0

0

0

0

0

10

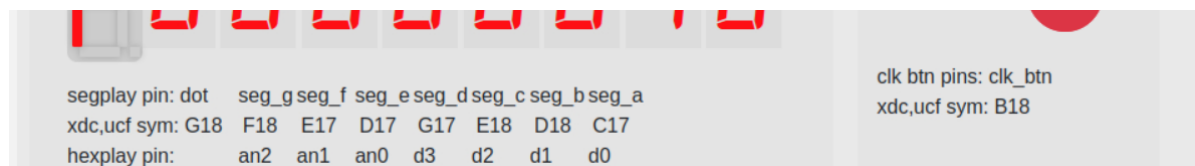
segplay pin: dot seg_g seg_f seg_e seg_d seg_c seg_b seg_a
xdc,ucf sym: G18 F18 E17 D17 G17 E18 D18 C17
hexplay pin: an2 an1 an0 d3 d2 d1 d0

seg_g seg_f seg_e seg_d seg_c seg_b seg_a

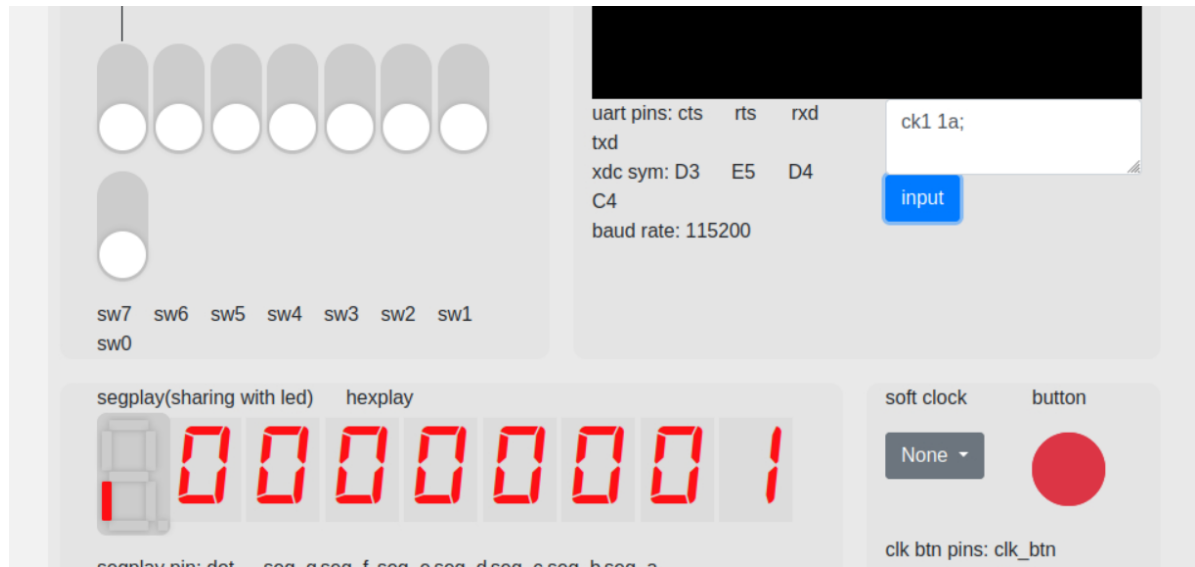
soft clock button

None

clk btn pins: clk_btn
xdc,ucf sym: B18



然后运行到程序结尾测试通过



总结

假设单周期cpu每clk需要1000ns,多周期cpu每clk需要200ns,多周期cpu每一时刻执行5条指令,所以多周期cpu平均每条指令需要200ns,而单周期cpu需要1000ns,所以性能提高5倍.