

# 架构实验5

# 使用计分板的动态调度流水线

# 任务

控制单元 控制...

寄存器模块(.clk(debug\_clk),.rst(rst),

ImmGen模块(.ImmSel(ImmSel\_ctrl),.inst\_field(inst\_ID),.Imm\_out(Imm\_out\_ID));

.R\_addr\_A (inst\_ID[19:15]), .rdata\_A (rs1\_data\_ID),

- •重新设计具有IF/ID/FU/WB阶段且FU阶段支持多周期的流水线操作。
- •重新设计CPU控制器。

缪晨璐 2022年11月

概述

•控制单元

•功能单元

•架构概述

•解决数据冒险的流水线

•解决控制冒险的流水线

#### 架构概述 - 指令获取阶段

REG\_PC(.clk(调试时钟),.rst(复位信号),.CE(寄存器指令获取阶段使能信号),.D(下一指令获取阶段程序计数器值),.Q(指令获取阶段程序计数器值));

32位加法器add\_IF(.a(指令获取阶段程序计数器值),.b(32位十进制数4),.c(指令获取阶段程序计数器值加4));

32位二路选择器

选择器IF(.I0(指令获取阶段程序计数器值加4),.I1(功能单元跳转程序计数器值),.s(分支控制信号),.o(下一指令获取阶段程序计数器值));

指令只读存储器inst\_rom(.a(指令获取阶段程序计数器值[8:2]),.spo(指令获取阶段指令));

#### 架构概述 - 功能单元

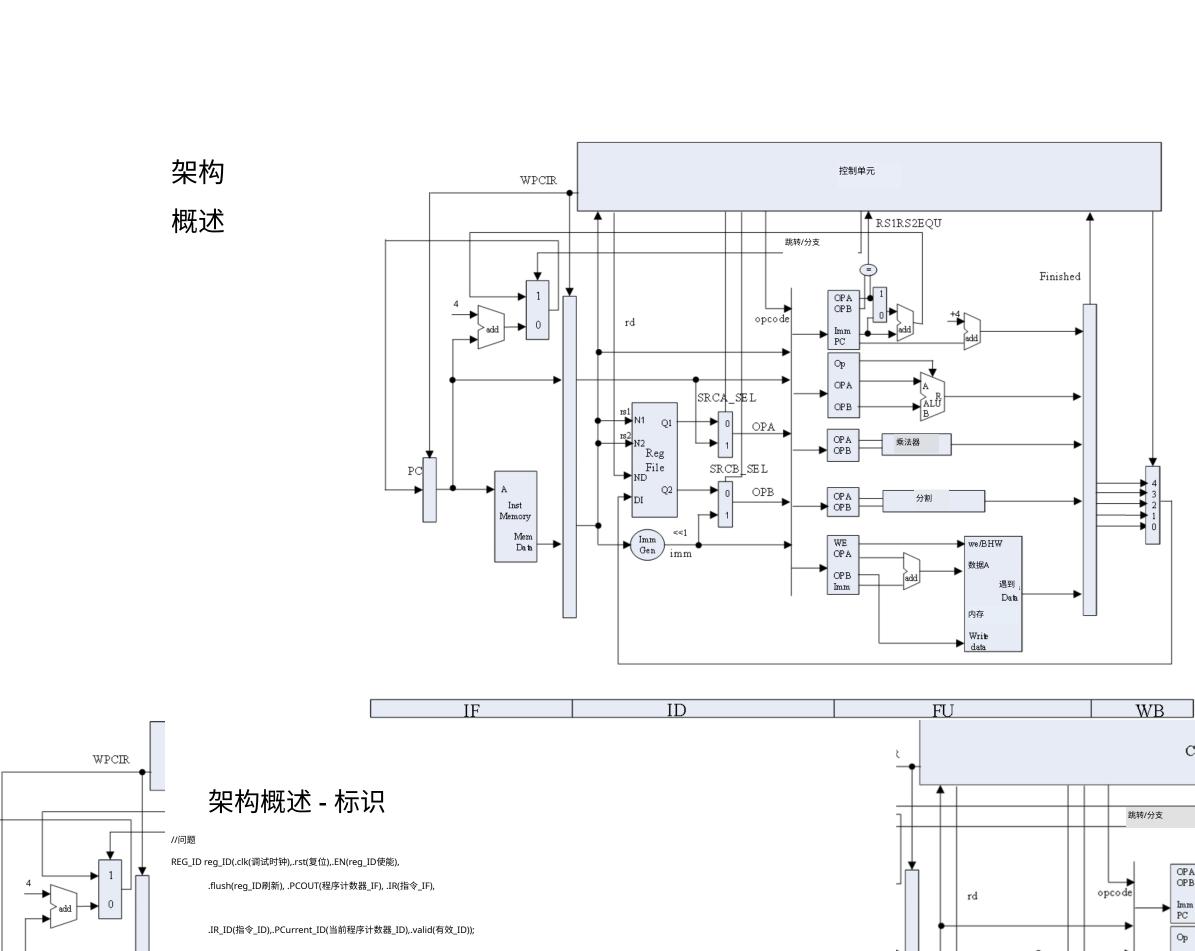
// FU 功能单元算术逻辑单元 alu(...);

功能单元内存单元 mem(...);

功能单元乘法器 mu(...);

除法功能单元 du(...);

跳转功能单元 ju(...);



源A选择

SRCB SEL



```
输出寄存器IF使能信号,分支控制信号
                                                                                     功能单元 - 算术逻辑单元
 控制单元
                                // ID
                                                                                    模块 FU_ALU(
                                输出寄存器ID使能信号,寄存器ID刷新信号
                                                                                                                            寄存器[3:0] 控制;
                                                                                        输入时钟信号,使能信号
 模块CtrlUnit(
                                输出[2:0]立即数选择信号
                                                                                                                            寄存器[31:0] A, B;
                                输出算术逻辑单元使能信号、存储器使能信号、乘法器使能信号、除法器使能信号、跳转使的各号:0]位的算术逻辑单元控制信号
     输入时钟信号
                                                                                        输入[31:0]位的算术逻辑单元操作数A,操作数B
     输入复位信号
                                                                                                                            始终在时钟信号的上升沿开始
                                // FU
                                                                                        输出[31:0]位的结果
                                输出[3:0]位跳转操作码
     输入[31:0]位指令
                                                                                                                               如果(使能信号 & 非状态信号) 开始 // 状态 == 0
                                                                                        输出零标志位,溢出标志位
                                输出[3:0]位算术逻辑单元操作码
     输入指令译码有效信号
                                                                                                                                    A <= ...;
                                                                                        输出完成标志位
                                输出ALU源A
                                                                                                                                    B <=...;
                                                                                   );
                                输出ALU源B
     输入ALU完成信号
                                输出存储器写使能信号
     输入存储器完成信号
     输入乘法器完成信号
                                                                                        寄存器状态;
                                                                                                                               结束
                                // WB
     输入DIV完成信号
                                                                                        赋值 finish = 状态 == 1'b1;
                                输出3位寄存器写选择信号
                                                                                                                                否则状态 <= 0;
     输入JUMP完成信号
                                                                                        初始开始
                                输出5位读控制信号
     输入功能单元比较结果
                                                                                                                            结束
                                                                                           状态 = 0;
                                输出寄存器写信号
                                                                                        结束
                                          始终@(时钟信号的上升沿) 开始
                                              如果(使能信号 & 状态信号全为低电平) 开始
  功能单元 - 乘法器
                                                                                     功能单元 - 除法器
                                                                                                                                    A有效 ...
                                                 A寄存器 ...
                                                                                                                                    B有效 ...
                                                                                     模块 FU div(
模块 FU_mul(
                                                 B寄存器 ...
                                                                                                                                    状态 ...
                                                                                        输入 时钟信号,使能信号,
                                                                                                                                 结束
   输入时钟信号、使能信号
                                                 状态...
                                                                                        输入 [31:0] A, B,
                                              结束
                                                                                        输出 [31:0] 结果,
   输入 [31:0] 位的 A、B
                                                                                        输出 完成信号
                                              否则状态 <= {1 位二进制 0, 状态 [6:1]}
   输出 [31:0] 位的结果
                                                                                                                                    状态...
   输出完成
                                          结束
                                                                                                                                 结束
                                                                                        线网 结果有效信号;
                                                                                                                              结束
                                                                                        63位至0位的线网divres;
                                          位宽为64位(63:0)的导线mulres;
                                                                                        寄存器state;
   位宽为7位(6:0)的寄存器state;
                                                                                        将finish赋值为res_valid与state的逻辑与;
   将finish赋值为state的第0位等于1;
                                          乘法器
                                                                                        初始块开始
                                                                                           state赋值为0;
                                      mul(.CLK(时钟信号), .A(A寄存器), .B(B寄存器), .P(乘法结果));
   初始开始
                                                                                        结束
       状态 = 0;
                                                                                        寄存器 A valid、B valid;
                                          赋值 结果 = 乘法结果[31:0];
   结束
                                                                                        32位寄存器 A_reg、B_reg;
   寄存器[31:0] A寄存器, B寄存器;
                                      模块结束
                                                                                                                          模块结束
 功能单元 - 跳转
                                                                                     功能单元 - 内存
                                            始终在时钟上升沿开始
                                               如果(使能信号 & <状态) 开始 // 状态 == 0
 模块 FU_jump(
                                                  JALR寄存器 ...
                                                                                     模块 FU mem(
                                                   比较控制寄存器 ...
     输入时钟信号,使能信号,JALR信号,
                                                                                        输入 时钟信号、使能信号、内存写信号,
                                                  rs1数据寄存器...
                                                                                                                                      立即数寄存器...
     输入[2:0] 比较控制信号,
                                                                                        3位输入 bhw,
                                                  rs2数据寄存器...
                                                                                                                                      状态...
     输入[31:0] rs1数据,rs2数据,立即数,
                                                                                        输入[31:0] rs1数据、rs2数据、立即数
                                                   立即数寄存器...
                                                                                                                                   end
                                                                                        输出[31:0] 内存数据
     输出[31:0] 程序计数器跳转值,程序计数器写回值
                                                  程序计数器寄存器...
                                                                                                                                    否则状态...
                                                                                        输出完成
                                                                                                                               结束
                                                   状态…
```

```
寄存器[1:0] 状态;
    赋值 完成 = 状态[0] == 1'b1;
    初始开始
        状态 = 0;
    结束
    寄存器 mem_w_reg;
    寄存器[2:0] bhw_reg;
    寄存器[31:0] rs1_data_reg, rs2_data_reg,
立即数寄存器;
```

```
控制 <=...;
       状态 <= 1;
始终在(时钟信号的上升沿)开始
    如果(使能信号 & <状态) 开始 // 状态 == 0
       A寄存器 ...
       B寄存器 ...
    否则如果(结果有效)则开始
       一个有效的...
       一个有效的...
分隔器 div(.aclk(clk),
    .s_axis_dividend_tvalid(A有效),
    .s_axis_dividend_tdata(A寄存器),
    .s_axis_divisor_tvalid(B有效)
    .s_axis_divisor_tdata(B寄存器),
    .m_axis_dout_tvalid(结果有效),
    .m_axis_dout_tdata(除法结果)
将 res 赋值为 divres[63:32];
  始终在时钟上升沿开始执行:
       如果 (EN 且 state 各位均为 0) 开始执行:
          内存写寄存器 ...
          字节半字寄存器 ...
         rs1数据寄存器...
          rs2数据寄存器...
```

### 解决数据冒险的流水线

寄存器[31:0] rs1数据寄存器,rs2数据寄存器,

输出 比较结果,结束信号

赋值 结束信号 = 状态 == 1'b1;

寄存器 状态;

初始块开始

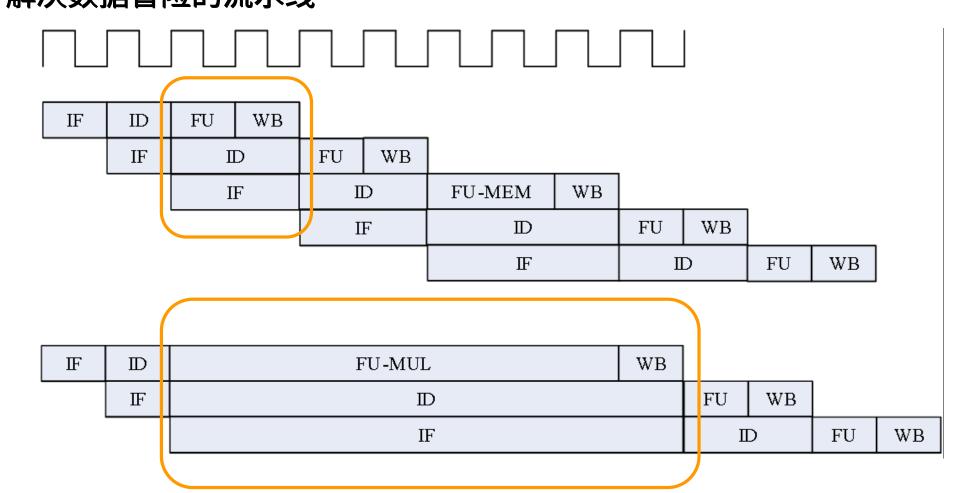
结束

状态 = 0;

寄存器 JALR寄存器;

立即数寄存器,程序计数器寄存器;

寄存器[2:0] 比较控制寄存器;



结束

32位比较器 比较 ...

32位加法 a...

32位加法 b...

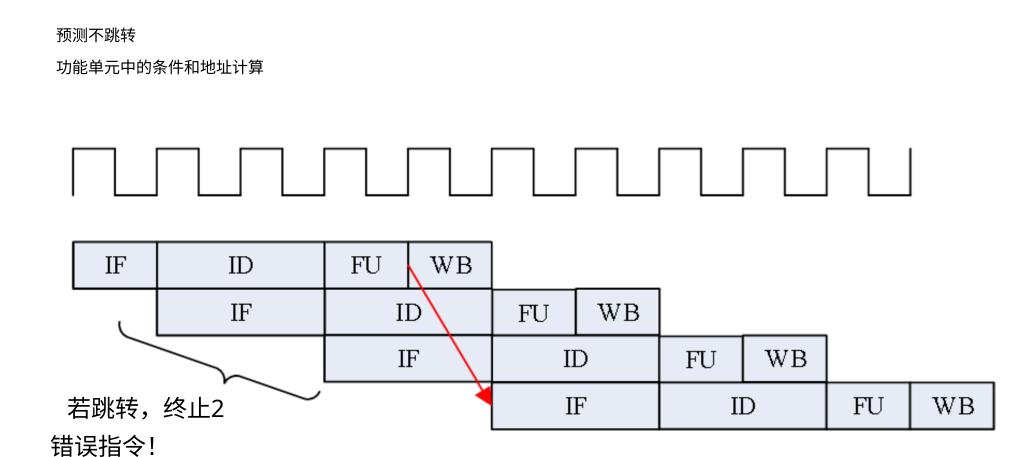
结束

模块结束

否则状态...

// IF

## 解决控制冒险的流水线



线网[31:0] 地址;

32位加法器 加...

.写使能a(寄存器内存写信号),

随机存取存储器(.时钟信号a(时钟),.地址a(地址),.数据输入a(寄存器rs2

.数据输出a(内存数据),.内存无符号字节/半字/字(寄存器b

RAM B

模块结束

NO.	指令	地址	 「标签	ASM	
0	00000013	0	开始:	将立即数0加到x0寄存器,结果存于x0	
1	00402103	4		将内存地址为x0 + 4处的数据加载到寄存器x2	
2	00802203	8		将内存地址为x0 + 8处的数据加载到寄存器x4	
3	004100b3	С		将寄存器x2和x4中的值相加,结果存入寄存器x1	
4	fff08093	10		将寄存器x1的值减1,结果存入寄存器x1	
5	00c02283	14		将内存地址为x0 + 12处的数据加载到寄存器x5	
6	01002303	18		从地址(x0 + 16)处加载字到x6寄存器	
7	01402383	1C		从地址(x0 + 20)处加载字到x7寄存器	
8	402200b3	20		用x4寄存器的值减去x2寄存器的值,结果存入x1寄存器	
9	ffd50093	24		将x10寄存器的值减3后存入x1寄存器	
10	00520c63	28		若x4寄存器的值等于x5寄存器的值,则跳转到label0处	
11	00420a63	2C		若x4等于x4,则跳转到标签label0	
12	00000013	30		将x0的值加0后存回x0	
13	00000013	34		将x0的值加0后存回x0	
14	00000013	38		将x0的值加0后存回x0	

指令	地址	标签	ASM	注释
00000013	3C		addi x0,x0,0	
000040b7	40	标签0:	lui x1,4	
00c000ef	44		跳转到地址12并链接到x1寄存器	
00000013	48		将x0寄存器的值加0后存回x0寄存器	
00000013	4C		将x0寄存器的值加0后存回x0寄存器	
00000013	50		将x0寄存器的值加0后存回x0寄存器	
00000013	54		将x0寄存器的值加0后存回x0寄存器	
ffff0097	58		将0xffff0与程序计数器相加并将结果存入x1寄存器	
0223c433	5C		将x7寄存器的值除以x2寄存器的值,结果存入x8寄存器	
025204b3	60		将x4寄存器的值乘以x5寄存器的值,结果存入x9寄存器	
022404b3	64		将x8寄存器的值乘以x2寄存器的值,结果存入x9寄存器	
00400113	68		将立即数4加到x0寄存器的值上,结果存入x2寄存器	
000000e7	6C		跳转到x0寄存器的值加上偏移量0的地址,并将返回地址存入x1寄存器	
	00000013 000040b7 00c000ef 00000013 00000013 00000013 ffff0097 0223c433 025204b3 022404b3 00400113	00000013       3C         000040b7       40         00c000ef       44         00000013       48         00000013       4C         00000013       50         00000013       54         ffff0097       58         0223c433       5C         025204b3       60         022404b3       64         00400113       68	000000013   3C   标签0:     1	00000013   3C   addi x0,x0,0

R M

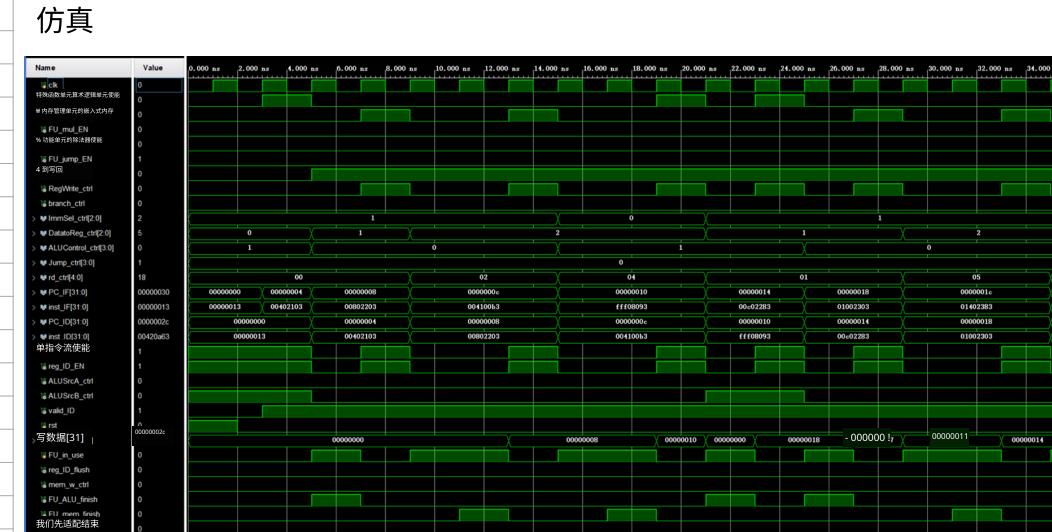
 ${f R}$ 

0

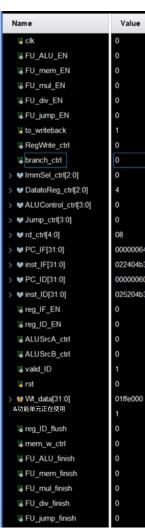
M

NO.	数据	地址
0	000080BF	0
1	00000008	4
2	00000010	8
3	00000014	С
4	FFFF0000	10
5	0FFF0000	14
6	FF000F0F	18
7	F0F0F0F0	1C
8	0000000	20
9	0000000	24
10	0000000	28
11	0000000	2C
12	0000000	30
13	0000000	34
14	0000000	38
15	0000000	3C

NO.	指令	地址
16	0000000	40
17	0000000	44
18	0000000	48
19	0000000	4C
20	A3000000	50
21	27000000	54
22	79000000	58
23	15100000	5C
24	0000000	60
25	0000000	64
26	0000000	68
27	0000000	6C
28	0000000	70
29	0000000	74
30	0000000	78
31	0000000	7C



仿真

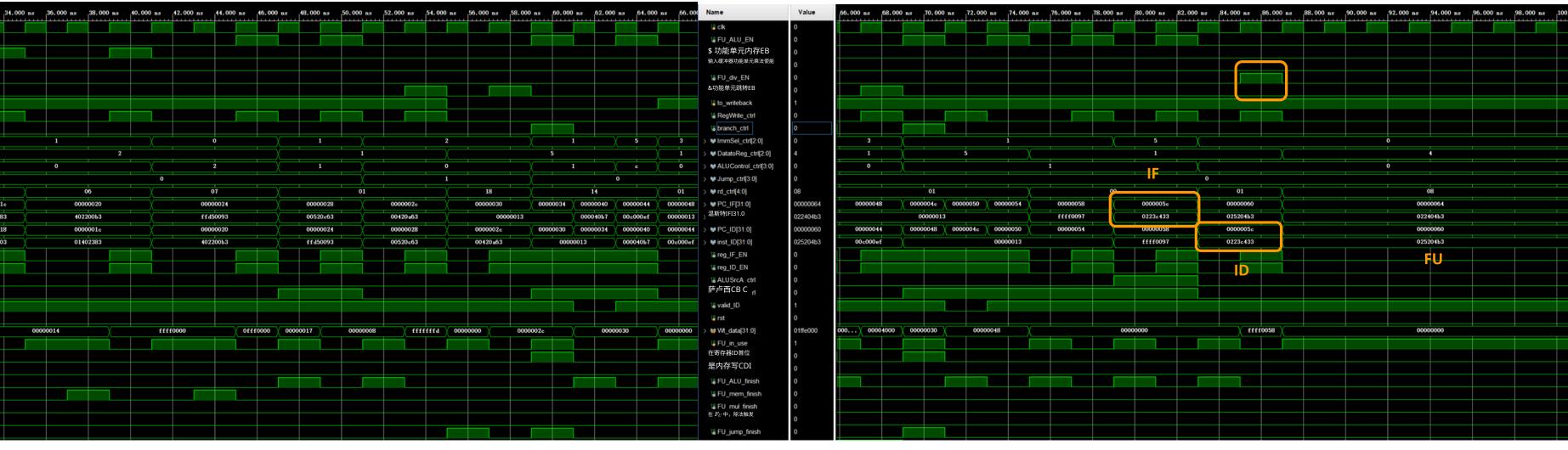


01402383

00000018

01002303





仿真

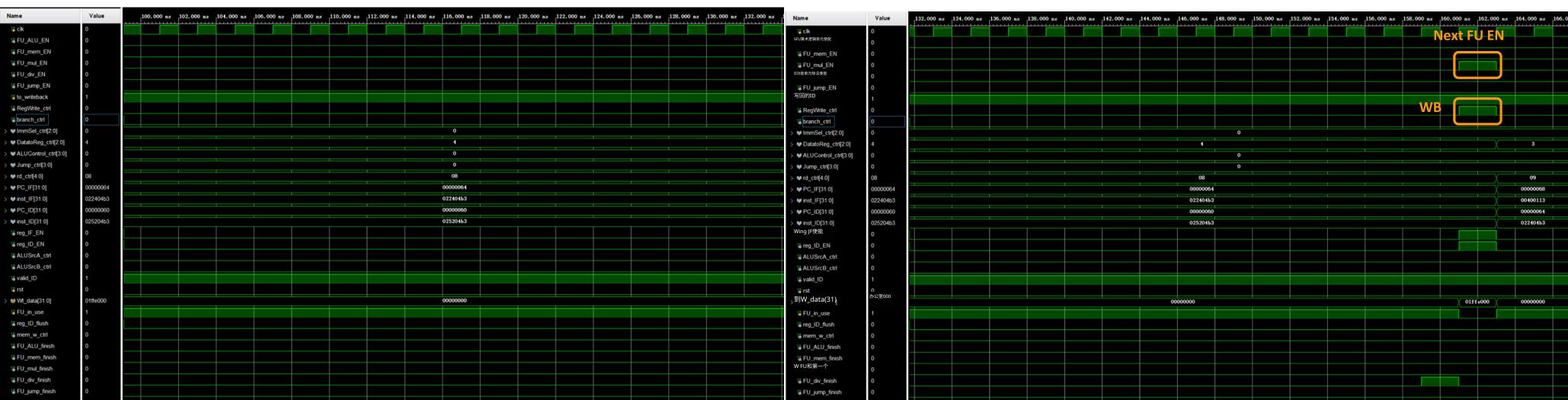
仿真

♣ FU\_div\_finish
% 首次跳转功能单元

仿真

R

M



模拟

