计算机体系结构实验

MIPS PIPELINE

流水线 cpu 设计实验

测

试

手

册

实验组成员: 张瑞祎,徐栋,李乾科

手册编者: 李乾科

2014-6-10

目 录

首页 .		•	•	•		•	•	•	•	•	•	•	•	•	•	•	•	•	•	1
目录 .			•				•	•			•	•	•			•	•		•	2
单条指	令测	则词	ζ.	•		•			•					•			•	•		3
R-type	指令	>	•				•	•		•	•									3
I-type	指令	À						•	•			•	•	•	•	•	•	•		6
lw,sw	及其	相	关扌	旨令	>测	试		•	•	•	•	•		•	•	•	•	•		11
跳转力	及分え	支指	† 令	测计	试.			•		•	•	•	•		•	•	•			14
冒险测	试	•	•				•	•	•	•	•	•	•		•	•				17
test-1			•				•													17
test-2		•	•			•	•		•					•				•		20
test-3		•			•	•	•	•												25
程序测	试		•				•													33
计数器	i r						•		•	•		•		•	•		•			33
冒泡排	序					•										•	•	•		34

一、 单条指令测试

State:

单条指令测试分为四个部分:

R-type 指令测试、I-type 指令测试、sw 与 lw 及其相关指令测试、跳转及分支指令测试

其中 R-type 指令与 I-type 指令共 15 种指令,在一个测试文件中测试,测试文件共 18 条指令,其中多出的三条为 add,addi,sub 溢出的情况测试。在这 15 张截图中,为了方便观察验证,每条指令只截取了从取指令开始的四个周期。图中显示包括: IR,ALU、SHIFT的输入输出,ID 段中有关寄存器的输入输出,Data 输出,跳转控制信号,PC 的输入输出等信号。

sw 与 lw 及其相关指令测试中我们测试了 sw,swl,swr,lw,lwl,lwr 六种指令。为了更好的显示测试,我们显示了更多的信号,包括: IR, ALU、SHIFT 的输入输出, ID 段中有关寄存器的输入输出, Data 输出, 跳转控制信号, PC 的输入输出, 数据内存相关的寄存器值, 立即数的值以及是否冲刷的控制信号等。

跳转及分支指令测试,共有 begz,begzal,beq,j 四条指令,信号输出同 sw、lw 指令,其他测试说明在测试图后会说明

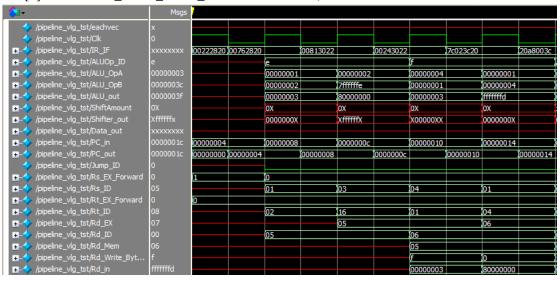
1、R-type 指令

initial

r1 = 1; r2 = 2; r3 = 3; r20 = 0x7ffffffc; r21 = 0x7ffffffd; r22 = 0x7ffffffe;

add

ram[0]=32'b0000000001000100010100000100000; //add r1 r2 = 3 -> r5 不溢出 ram[1]=32'b000000_00011_10110_0010100000100000; //add r3 r22 -> r5 溢出



sub

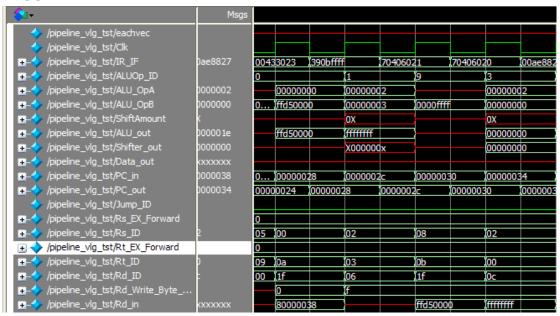
ram[2]=32'b00000000100000010011000000100010; //sub r4 r1 = 3 -> r6 不溢出

ram[3]=32'b000000_00001_00100_0011000000100010; //sub r1 r4 = -3 -> r6 溢出

/pipeline_vlg_tst/eachvec	·										
/pipeline_vig_tst/eactivec /pipeline_vig_tst/Clk	1										
pipeline_vlg_tst/IR_IF	xxxxxxxx	00813022	01043022		7c023c20		20a8003c		2288003c		
/pipeline_vlg_tst/ALUOp_ID	e	e		f				a		le le	
-/-/ /pipeline_vlg_tst/ALU_OpA	7ffffffc	00000002		00000004		80000001		00000000		00000003	
-/-/ /pipeline_vlg_tst/ALU_OpB	0000003c	7ffffffe		00000001		00000004		00000002		0000003c	
	80000038	80000000		00000003		7ffffffd		00000002		0000003f	
-/-/ /pipeline_vlg_tst/ShiftAmount	хX	0X		0X		0X		x0		0X	
/pipeline_vlg_tst/Shifter_out	xxxxxxX	XffffffX		X00000XX		0000000X		000X000X		x000000x	
Ⅲ - <pre>/pipeline_vlg_tst/Data_out</pre>	xxxxxxxx										
Ⅲ — /pipeline_vlg_tst/PC_in		000000c		00000010		00000014		00000018		0000001c	
	0000001c	80000000	0000000c		00000010		00000014		00000018		0000001c
<pre>/pipeline_vlg_tst/Jump_ID</pre>	0										
	0	0									
	14	03		04		08		00		05	
pipeline_vlg_tst/Rt_EX_Forward	0	0									
/pipeline_vlg_tst/Rt_ID	1	16		01		04		02		08	
// /pipeline_vlg_tst/Rd_EX	l i	05 05		Vac.		06		Vo.7		07	
pipeline_vlg_tst/Rd_ID // pipeline_vlg_tst/Rd_Mem	07	05		06 05				07 106		00	
*-// /pipeline_vig_tst/Rd_Write_Byte_en_Mem	f			U3 F		10		100 F		Ĭo.	
#- /pipeline_vig_tst/Rd_in	00000002			0000003		80000000		100000003		7fffffd	

subu

ram[9]=32'b00000000010000110011000000100011; //subu r2 r3 = -1 -> r6



nor

ram[13]=32'b000000010101110100010000100111; //nor r5 r14 -> r17

≨ 1+	Msgs									
/pipeline_vlg_tst/eachvec		-								
/pipeline_vlg_tst/Clk										
#-/ /pipeline_vlg_tst/IR_IF	9707fff	00ae882	7	002b680	7	002a71	2	00c7782	b	29707f
∓ - / /pipeline_vlg_tst/ALUOp_ID		3	2							
 /pipeline_vlg_tst/ALU_OpA	0000001	00000002	2				0000000	1		
≖ - <pre>/pipeline_vlg_tst/ALU_OpB</pre>	oxxxxxx	00000000)							
pipeline_vlg_tst/ShiftAmount	5	0X					01		05	
#/> /pipeline_vlg_tst/ALU_out	oxxxxxx	00000	0000001	e	0000002	0				
∓ - ∜ /pipeline_vlg_tst/Shifter_out	oxxxxxx	00000000)							
- - - - / pipeline_vlg_tst/Data_out	oxxxxxx	-								
II — ∜ /pipeline_vlg_tst/PC_in	00000 44	00000	0000003	8	0000003	c	0000004	0	0000004	4
∓ - ∜ /pipeline_vlg_tst/PC_out	0000044	0000003	4	0000003	8	0000003	c	0000004	0	000000
/pipeline_vlg_tst/Jump_ID										
pipeline_vlg_tst/Rs_EX_Forward		0								
∓ - ∜ /pipeline_vlg_tst/Rs_ID	ļ.	02			05		01			
pipeline_vlg_tst/Rt_EX_Forward		0								
#-// /pipeline_vlg_tst/Rt_ID	a	00			0e		0b		0a	
 /pipeline_vlg_tst/Rd_ID	2	0c			11		0d		0e	
+-/ /pipeline_vlg_tst/Rd_Write_Byte		f								
#/> /pipeline_vlg_tst/Rd_in	0000020	fffffff			0000000	0	0000001	e	0000002	20

€1 +	Msgs									
/pipeline_vlg_tst/eachvec										
/pipeline_vlg_tst/Clk										
-/	oxxxxxx	002b	6807	002a71	42	00c7782	b	29707ff		
		2			}				7	
	oxxxxxx	0	-		0000000	1				
∓ - ∜ /pipeline_vlg_tst/ALU_OpB	oxxxxxx	0	-							
	¢	0X			01		05			
 /pipeline_vlg_tst/ALU_out	oxxxxxx	0	000000	20	}					
 /pipeline_vlg_tst/Shifter_out	oxxxxxx	0	}—							
+ /pipeline_vlg_tst/Data_out	oxxxxxx									
+ /pipeline_vlg_tst/PC_in	0000048		000000		000000		000000		0000004	
+	0000048	0000	0038	000000	3c	000000	10	0000004	4	000000
<pre>/pipeline_vlg_tst/Jump_ID</pre>										
+- /pipeline_vlg_tst/Rs_EX_Forward		0								
+	Б		05		01				06	
+- /pipeline_vlg_tst/Rt_EX_Forward		0								
+ /pipeline_vlg_tst/Rt_ID	7		0e		0b		0a		07	
/pipeline_vlg_tst/Rd_ID			11		0d		0e		0f	
// /pipeline_vlg_tst/Rd_Write_Byte		f								
∓- ∜ /pipeline_vlg_tst/Rd_in	CXXXXXXX		(000000	UU	(000000)	le	(000000)	40		

rotr

 $ram[15] = 32'b00000000001010100111000101000010; //rotr\ r10\ 5 -> r14$

\$ 1+	Msgs								
/pipeline_vlg_tst/eachvec									
/pipeline_vlg_tst/Clk									
+	cxxxxxx	002a7	00c7782	b	29707ff				
+-/> /pipeline_vlg_tst/ALUOp_ID		<u> </u>				7		5	
+-/> /pipeline_vlg_tst/ALU_OpA	oxxxxxx	0000000	1						
+-/> /pipeline_vlg_tst/ALU_OpB	oxxxxx							00007ffi	
-//pipeline_vlg_tst/ShiftAmount	K	01		05					
-/	oxxxxxx								
-/	oxxxxxx	_							
-/	cxxxxxx	_							
-/	0000050	0000004	Ю	0000004	4	0000004	8	0000004	c
-/-/pipeline_vlg_tst/PC_out	000004c	00000	0000004	0	0000004	4	0000004	8	0000004
/pipeline_vlg_tst/Jump_ID									
		0							
	C	01				06		0b	
		0							
→ /pipeline_vlg_tst/Rt_ID	C	0b		0a		07		10	
	C	0d		0e		Of			
		f							
	OXXXXXX	0000001	e	0000002	0				

sltu ram[16] = 32'b0000000110001110111100000101011; //sltu r12 r13 -> r15

≨ 1+	Msgs						
/pipeline_vlg_tst/eachvec							
/pipeline_vlg_tst/Clk							
-/-/ /pipeline_vlg_tst/IR_IF)c7782b	00c7 2	9707fff				
-/-/ /pipeline_vlg_tst/ALUOp_ID				7		5	
-/-/ /pipeline_vlg_tst/ALU_OpA	0000001	00000001					
	CXXXXXX					00007fff	
→ /pipeline_vlg_tst/ShiftAmount	5	05					
→ /pipeline_vlg_tst/ALU_out	OXXXXXX						
	OXXXXXX						
- → /pipeline_vlg_tst/Data_out	OXXXXXX						
	0000044	00000044	1	0000004	8	0000004	c l
+	0000040	00000	000004	4	0000004	8	0000004
/pipeline_vlg_tst/Jump_ID							
		0					
+	1	01		06		0b	
+- /pipeline_vlg_tst/Rt_EX_Forward		0					
+	3	0a		07		10	
+ /pipeline_vlg_tst/Rd_ID	2	0e		0f			
 /pipeline_vlg_tst/Rd_Write_Byte		f					
+	0000020	00000020)				

2、I-type 指令

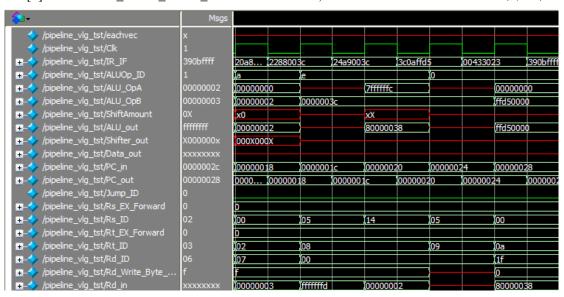
seb

 $ram[4] = 32'b01111100000000100011110000100000; \quad //seb \quad r2 \\ = 0 -> r7$

≨ 1+	Msgs									
/pipeline_vlg_tst/eachvec	х									
/pipeline_vlg_tst/Clk	0									
 /pipeline_vlg_tst/IR_IF	390bffff	7c02	23c20	20a8003	c	2288003	c	24a9003	c	3c0affd5
- pipeline_vlg_tst/ALUOp_ID	0	f			a		e			
- → /pipeline_vlg_tst/ALU_OpA	00000000	0	0000000	1	0000000	0			7ffffffc	
- → /pipeline_vlg_tst/ALU_OpB	ffd50000	0	0000000	4	0000000	2	0000003	c		
∓ - ∜ /pipeline_vlg_tst/ShiftAmount	xx	0X	0X		x0				хX	
 	ffd50000	0	fffffffd		0000000	2			8000003	8
∓ - ∜ /pipeline_vlg_tst/Shifter_out	XXXXXXXX	х	0000000	Х	000X000	Х				
∓ - ∜ /pipeline_vlg_tst/Data_out	xxxxxxxx									
∓ - ∜ /pipeline_vlg_tst/PC_in	00000028	0	0000001	4	0000001	8	0000001	c	0000002	0
∓ - ∜ /pipeline_vlg_tst/PC_out	00000028	0000	00010	000000	4	0000001	8	0000001	.c	0000002
<pre>/pipeline_vlg_tst/Jump_ID</pre>	0									
II — / /pipeline_vlg_tst/Rs_EX_Forward	0	0								
∓- <pre> /pipeline_vlg_tst/Rs_ID</pre>	00	04	01		00		05		14	
∓ - ∜ /pipeline_vlg_tst/Rt_EX_Forward	0	0								
∓ - ∜ /pipeline_vlg_tst/Rt_ID	0a	01	04		02		08			
→ /pipeline_vlg_tst/Rd_ID	1f	06			07		00			
II — ∜ /pipeline_vlg_tst/Rd_Write_Byte	0	f								
	80000038	0	1000000	1	0000000	3	fffffffd		0000000	2

addi

ram[5]=32'b00100000101010000000000000111100; //addi r5 im = 111111 -> r8 溢出 ram[6]=32'b001000_10100_01000_0000000000111100; //addi r20 im = 111111 -> r8 不溢出



lui

ram[8]=32'b0011110000001010111111111111010101; //lui 11111111111010101 -> r10

∻ 1+	Msgs								
/pipeline_vlg_tst/eachvec									
<pre>/pipeline_vlg_tst/Clk</pre>									
<pre>#</pre> /pipeline_vlg_tst/IR_IF	0406020	3c0	0043302	3	390bffff		7040602	1	7040602
<pre>#</pre> /pipeline_vlg_tst/ALUOp_ID		0				1		9	
<pre>#</pre> /pipeline_vlg_tst/ALU_OpA	0000002	_		0000000	0	0000000	2		
∓ - ∜ /pipeline_vlg_tst/ALU_OpB	0000000	000000)3c	ffd5000	0	0000000	3	0000ffff	
<pre></pre>	(0X			
<pre>#</pre> /pipeline_vlg_tst/ALU_out	0000000			ffd5000	0	ffffffff			
	0000000	_				X000000	x		
	oxxxxxx	_							
	0000034	000000	24	0000002	28	0000002	c	0000003	0
-/	0000030	000	0000002	4	0000002	8	0000002	c .	0000003
<pre>/pipeline_vlg_tst/Jump_ID</pre>									
IIIIIIIIIIIII		0							
 /pipeline_vlg_tst/Rs_ID	2	05		00		02		08	
/pipeline_vlg_tst/Rt_EX_Forward		0							
∓ - ∜ /pipeline_vlg_tst/Rt_ID		09		0a		03		0b	
∓ - ∜ /pipeline_vlg_tst/Rd_ID	:	00		1f		06		1f	
∓ - ∜ /pipeline_vlg_tst/Rd_Write_Byte				0		f			
+	rrrrr			8000003	8			ffd50000)

xoriram[10]=32'b00111001000010111111111111111111; //xori 1111111111110111110 -> r11

% +	Msgs								
/pipeline_vlg_tst/eachvec									
/pipeline_vlg_tst/Clk									
 /pipeline_vlg_tst/IR_IF	02b6807	390bffff	7040602	1	7040602	0	00ae882	27	002b68
≖ - <pre>/pipeline_vlg_tst/ALUOp_ID</pre>		1		9		3		2	
 /pipeline_vlg_tst/ALU_OpA	0000002	0000000	2			0000000	2		
 /pipeline_vlg_tst/ALU_OpB	0000000	0000000	3	0000ffff		0000000	0		
 /pipeline_vlg_tst/ShiftAmount	C	0X				0X			
 /pipeline_vlg_tst/ALU_out	000001e	ffffffff				0000000	0	000000	le
- - / pipeline_vlg_tst/Shifter_out	0000000	X000000	x			0000000	0		
- - / pipeline_vlg_tst/Data_out	oxxxxxx								
 /pipeline_vlg_tst/PC_in	0000038	0000002	c	0000003	0	0000003	4	0000003	8
 /pipeline_vlg_tst/PC_out	0000038	00000	0000002	c	0000003	0	0000003	4	000000
/pipeline_vlg_tst/Jump_ID									
 /pipeline_vlg_tst/Rs_EX_Forward		0							
 /pipeline_vlg_tst/Rs_ID	2	02		08		02			
 /pipeline_vlg_tst/Rt_EX_Forward		0							
	D	03		0b		00			
+	:	06		1f		0c			
 /pipeline_vlg_tst/Rd_Write_Byte		χf							
II —◆ /pipeline_vlg_tst/Rd_in	CXXXXXX			ffd5000)	ffffffff			

clo

 $ram[11] = 32'b011100_00010_00000_0110000000100001; \quad //clo \ r2 = 0 \ -> r12$

≨ 1+	Msgs									
/pipeline_vlg_tst/eachvec										
/pipeline_vlg_tst/Clk										
+-/ /pipeline_vlg_tst/IR_IF	02a7142	70406	021	7040602	0	00ae88	27	002b680	7	002a71
-/-/ /pipeline_vlg_tst/ALUOp_ID		1	9		3		2			
+-/> /pipeline_vlg_tst/ALU_OpA	0000001	000	}		0000000	2			-	
+-/- /pipeline_vlg_tst/ALU_OpB	oxxxxxx	000	0000ffff		0000000	0				
-//pipeline_vlg_tst/ShiftAmount	1	0X			0X					
+-/	xxxxxx	fffffff	-		0000000	0	0000001	e	0000002	.0
//pipeline_vlg_tst/Shifter_out	oxxxxxx	X00			0000000	0				
//pipeline_vlg_tst/Data_out	cxxxxxx		-							
-//pipeline_vlg_tst/PC_in	0000040	000	0000003	0	0000003	4	0000003	8	0000003	c
+-/pipeline_vlg_tst/PC_out	000003c	00000	02c	0000003	0	000000:	4	0000003	8	000000
/pipeline_vlg_tst/Jump_ID										
/pipeline_vlg_tst/Rs_EX_Forward		0								
<pre>#</pre>	1	02	08		02				05	
/pipeline_vlg_tst/Rt_EX_Forward		0								
+-/	þ	03	0b		00				0e	
+-/	H	06	1f		0c				11	
≖ - <pre>/pipeline_vlg_tst/Rd_Write_Byte</pre>		f								
+-/	000001e		ffd5000		ffffffff				0000000	0

 $\label{eq:clz} \mbox{ram} \mbox{[12]=32'b011100_00010_00000_01100_00000100000;} \quad \mbox{//clz r2 = 30 -> r12}$

€ 1+	Msgs									
/pipeline_vlg_tst/eachvec										
/pipeline_vlg_tst/Clk		Ш								
 /pipeline_vlg_tst/IR_IF	0c7782b	7040	6020	00ae882	7	002b680	7	002a714	2	00c7782
 - pipeline_vlg_tst/ALUOp_ID		9 3			2					
 /pipeline_vlg_tst/ALU_OpA	0000001	0	000000	2					0000000	1
 /pipeline_vlg_tst/ALU_OpB	oxxxxxx	0	000000	0						
- pipeline_vlg_tst/ShiftAmount	5	0	X						01	
 /pipeline_vlg_tst/ALU_out	oxxxxxx	0	000000	0	0000001	e	000000	20		
 /pipeline_vlg_tst/Shifter_out	oxxxxxx	0	000000	0						
pipeline_vlg_tst/Data_out	oxxxxxx	\vdash								
pipeline_vlg_tst/PC_in	00000 44	0	000003	4	0000003	8	0000003	c	0000004	10
//pipeline_vlg_tst/PC_out	0000040	0000	0030	0000003	4	0000003	8	0000003	c	000000
/pipeline_vlg_tst/Jump_ID										
//pipeline_vlg_tst/Rs_EX_Forward		0								
/pipeline_vlg_tst/Rs_ID	ļ.	08 0	2				05		01	
#————————————————————————————————————		0								
/pipeline_vlg_tst/Rt_ID	а	0b 0	0				0e		0b	
 /pipeline_vlg_tst/Rd_ID	1	1f 0	С				11		0d	
 /pipeline_vlg_tst/Rd_Write_Byte		f								
	0000020	ff	ffffff				0000000	0	000000	le

slti

ram[17] = 32'b001010010111000001111111111111111; //slti r11 0x7fff -> r16

<u>\$</u> 1+	Msgs						
/pipeline_vlg_tst/eachvec							
/pipeline_vlg_tst/Clk							
-/)c7782b	00c7	29707ff				
-/				7		5	
	0000001	0000000	1				
∓ - ∜ /pipeline_vlg_tst/ALU_OpB	oxxxxxx					00007fff	
	5	05					
	oxxxxxx						
	oxxxxxx						
	oxxxxxx						
→ /pipeline_vlg_tst/PC_in)000044	0000004	14	0000004	8	0000004	c ,
	0000040	0000	0000004	4	0000004	8	0000004
<pre>/pipeline_vlg_tst/Jump_ID</pre>							
		0					
	1	01		06		0b	
/pipeline_vlg_tst/Rt_EX_Forward		0					
	a	0a		07		10	
	=	0e		Of			
- - - - - - - - - -		f					
	0000020	0000002	20				

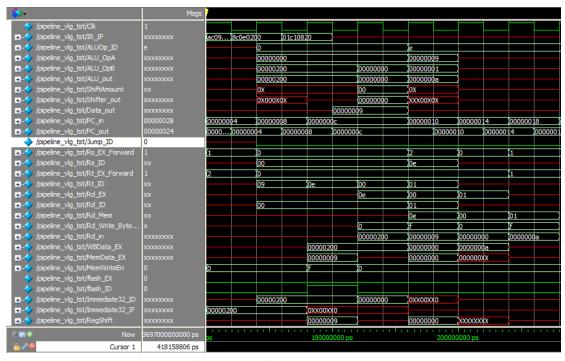
3、sw 、 lw 及其相关指令测试

initial

register[0] = 0;

```
register[1] = 1;
register[2] = 5;
register[3] = 4;
register[4] = 3;
register[5] = 2;
register[8] = 32'hffffffe;
register[9] = 9;
register[20] = 32'h7ffffffc;
register[21] = 32'h7ffffffd;
register[22] = 32'hffffffe;

ram[0] = {16'b101011_00000_01001,16'd512};//sw reg[9] -> IM[512]
ram[1] = {16'b100011_00000_01110,16'd512};//lw IM[512] -> reg[14]
ram[2] = 32'b0000000_01110_00001_00001_000000; add r14 + r1 ->r1;
```



r14 并未有初值,其值是从 r9 转入内存再转入 r14,因此,add 的正常执行表示 sw 与 lw 都 正常执行了。

lw,sw 及其相关指令的综合测试

state:

本项测试将 lwl,lwr,swl,swr 中每一种转存方式都测试了一次,详情请看以下测试

initial

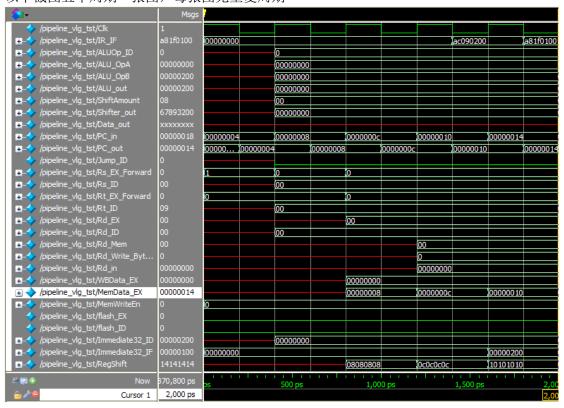
ram[2] = 32'b0; ram[3] = 32'b0;

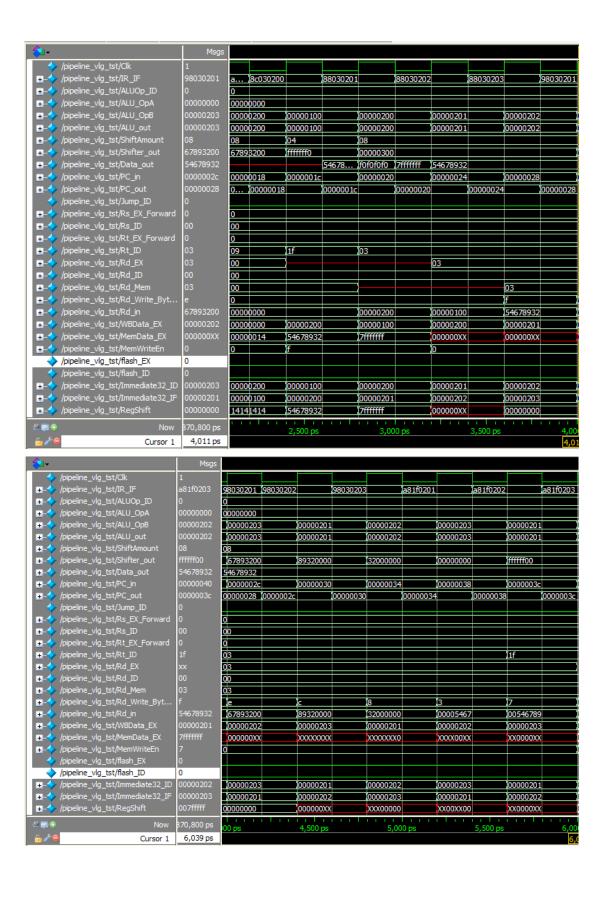
```
register[0] = 32'h0;
    register[1] = 32'h11112345;
    register[2] = 32'h2;
    register[3] = 32'h3;
    register[4] = 32'h4;
    register[5] = 32'h55556789;
    register[8] = 32'h88;
    register[9] = 32'h5467_8932;
    register[10] = 32'h3476_8906;
    register[11] = 32'hfffa_bcde;
    register[12] = 32'h6789_3954;
    register[13] = 32'h88;
    register[30] = 32'hffff_ffff;
    register[31] = 32'h7fff_ffff;
二进制代码:
ram[0] = 32'b0;
ram[1] = 32'b0;
```

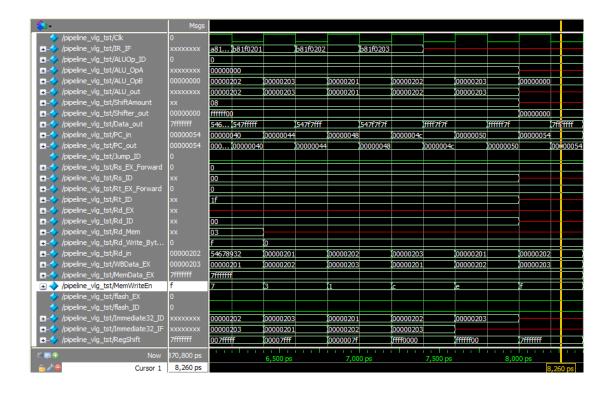
```
ram[4] = \{16b101011\_00000\_01001,16d512\};//sw
                                                  Reg[9] -> DM[512]
ram[5] = \{16b101010\_00000\_11111,16d256\};//swl
                                                  reg[31] -> DM[256]
ram[6] = \{16b100011\_00000\_00011,16d512\};//lw
                                                  ID[512] -> reg[3]
ram[7] = \{16b100010\_00000\_00011,16d513\}; //lwl
                                                   DM[513] -> reg[3]
ram[8] = \{16b100010\ 00000\ 00011, 16d514\}; //lwl
                                                   DM[514] -> reg[3]
ram[9] = \{16'b100010\_00000\_00011,16'd515\}; //lwl
                                                   DM[515] -> reg[3]
ram[10] = \{16b100110\ 00000\ 00011, 16d513\}; //lwr
                                                    DM[513] -> reg[3]
ram[11] = \{16b100110\_00000\_00011,16d514\}; //lwr
                                                    DM[514] -> reg[3]
ram[12] = {16b100110\_00000\_00011,16d515};//lwr
                                                    DM[515] -> reg[3]
ram[13] = \{16'b101010\ 00000\ 11111,16'd513\}; //swl
                                                    reg[31] -> DM[513]
ram[14] = \{16b101010\_00000\_11111,16b314\}; //sw1
                                                    reg[31] -> DM[514]
ram[15] = \{16b101010\_00000\_11111,16d515\};//swl
                                                    reg[31] -> DM[515]
ram[16] = \{16b101110_{00000_{11111,16}d513}\}; //swr
                                                    reg[31] -> DM[513]
ram[17] = \{16'b101110\ 00000\ 11111,16'd514\};//swr
                                                    reg[31] -> DM[514]
ram[18] = \{16b101110\_00000\_11111,16b15\}; //swr reg[31] -> DM[515]
```

仿真截图:

以下截图五个周期一张图,每张图无重复周期







4、跳转及分支指令测试

initial:

register[0] = 0;

register[1] = 1;

register[2] = 5;

register[3] = 4;

register[4] = 3;

register[5] = 2;

register[8] = 32'hfffffffe;

register[9] = 9;

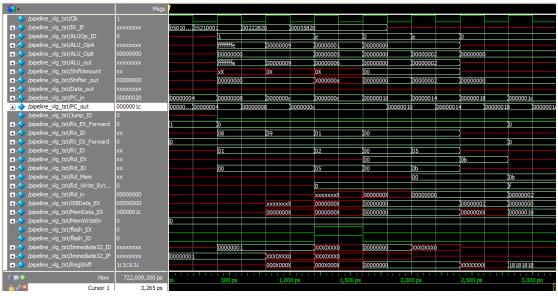
register[20] = 32'h7ffffffc;

register[21] = 32'h7ffffffd;

register[22] = 32'hffffffe;

begz

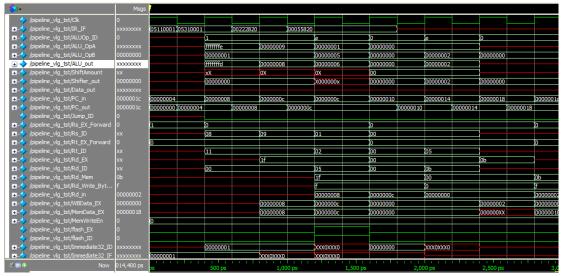
```
ram[0] = 32'b000001_01000_00001_00000000000001; //bgez r8 -> 1 跳转失败 ram[1] = 32'b000001_01001_00001_00000000000001; //bgez r9 -> 1 跳转成功 ram[2] = 32'b000000_00001_00010_0010100000100000; //add r1 r2 = 6 -> r5 ram[3] = 32'b000000_00000_00101_0101100000100000; //add r0 r5 = ? -> r11
```



从图中可以看出,r0+r5的值为2,意味着跳过了r5加为6的指令操作,跳转有效

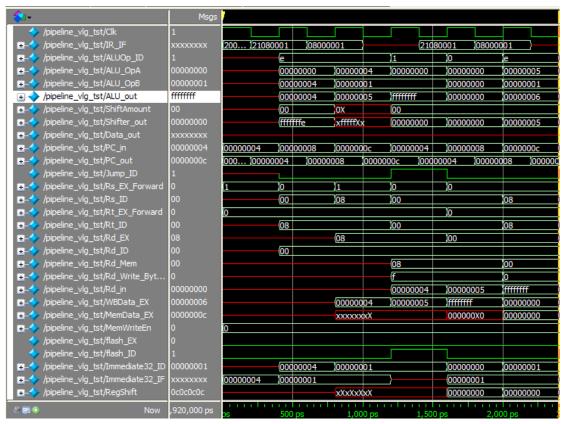
begzal

```
ram[0]=32'b000001\_01000\_10001\_000000000000001; //bgez r8 -> 1 跳转失败 ram[1]=32'b000001\_01001\_10001\_000000000000001; //bgez r9 -> 1 跳转成功 ram[2]=32'b000000\_00001\_00010\_0010100000100000; //add r1 r2 = 6 -> r5 ram[3]=32'b000000\_00000\_00101\_0101100000100000; //add r0 r5 = ? -> r11
```



原理同 begz, r0+r5 的值为 2, begzal 有效

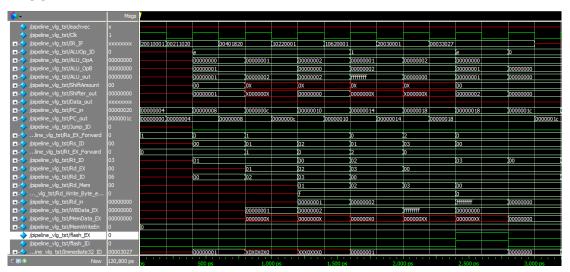
```
j
```



从图中可以看出,r8在不断的加一,因此j指令生效

beq:

```
ram[0] = 32'b001000\_00000\_00001\_0000000000000001; //addi r0 im(1) -> r1 ram[1] = 32'b000000\_00001\_00001\_0001000001; //add r1 r1 -> r2 ram[2] = 32'b000000\_00010\_00000\_0001100000100000; //add r2 r0 -> r3 ram[3] = 32'b000100\_00001\_00010\_000000000000001; //beq (r1 == r2)? ->1 跳转失败 ram[4] = 32'b000100\_00011\_00010\_00000000000001; //beq (r3 == r2)? ->1 跳转成功 ram[5] = 32'b001000\_00000_00011\_000000000000001; //addi r0 im(1) -> r3 ram[6] = 32'b000000\_00000_00011_00110\_00000100111; //nor r0 r3 -> r6
```



conclusion:

经以上测试, 单条指令的功能都已实现

二、冒险测试

test-1:

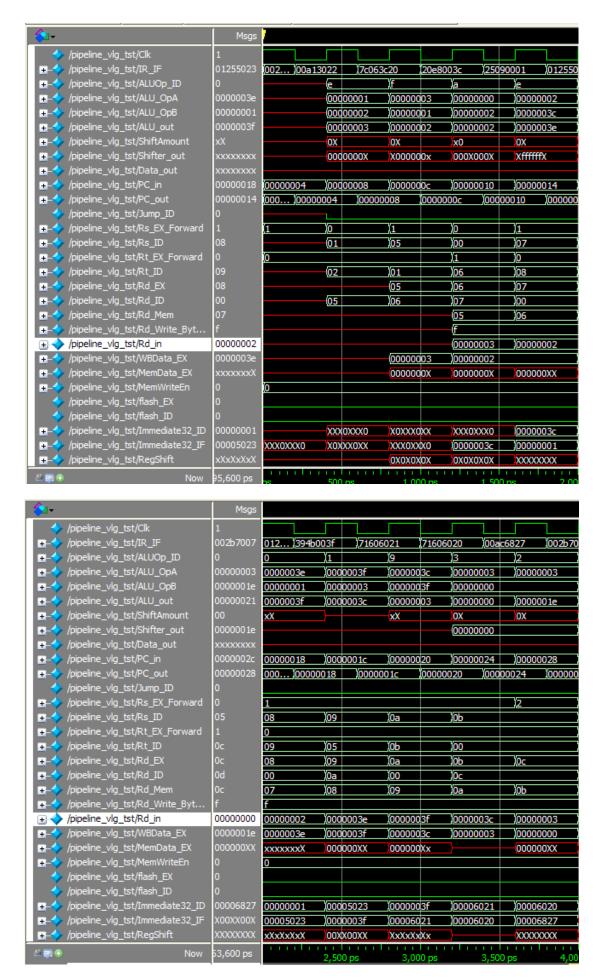
test-1 测试数据冒险,将 I-type R-type 指令的数据冒险均测试了一遍每条指令中用的寄存器都是上一条指令更新的寄存器test-1 的截图每张截图 5 个周期,不同截图的周期不重复

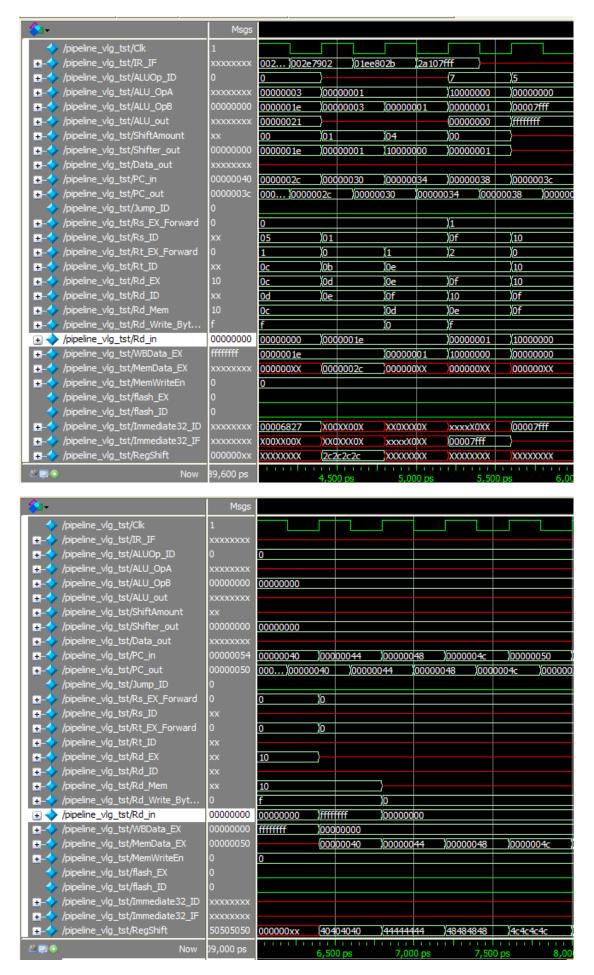
initial

register[0] = 0; register[1] = 1; register[2] = 2;

二进制代码:

仿真截图:





test-1-conclusion:

经以上测试, I-type R-type 指令的数据冒险都无误

test-2:

本次测试进行利用多次跳转及分支完成了一个 for 循环,可完成连续跳转的冒险测试,以及数据冒险和控制冒险夹杂并且同时发成的测试。

汇编代码的意义是:将 r1 加为 1, r2 加为 2, r3 加为 3, r5 加为 0, 然后进入循环。

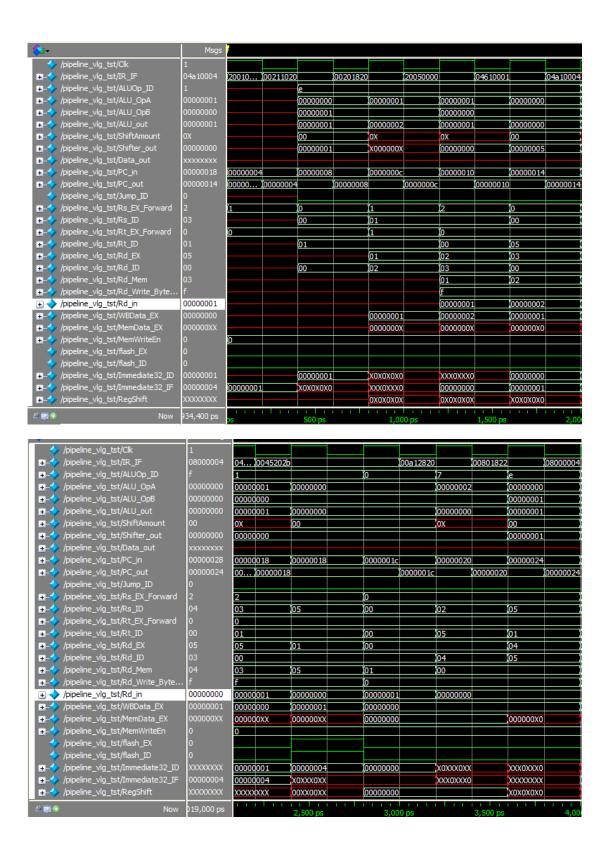
循环做的是: begzal 跳过 begz,然后下面每次使 r5 加一,直到 r5 大于等于 r2,就使得 begzal 不执行,然后执行 begz,跳过 jump,结束循环。最后五个 add 是希望得到 r1~r5 的值,来看是否和预期值相同。

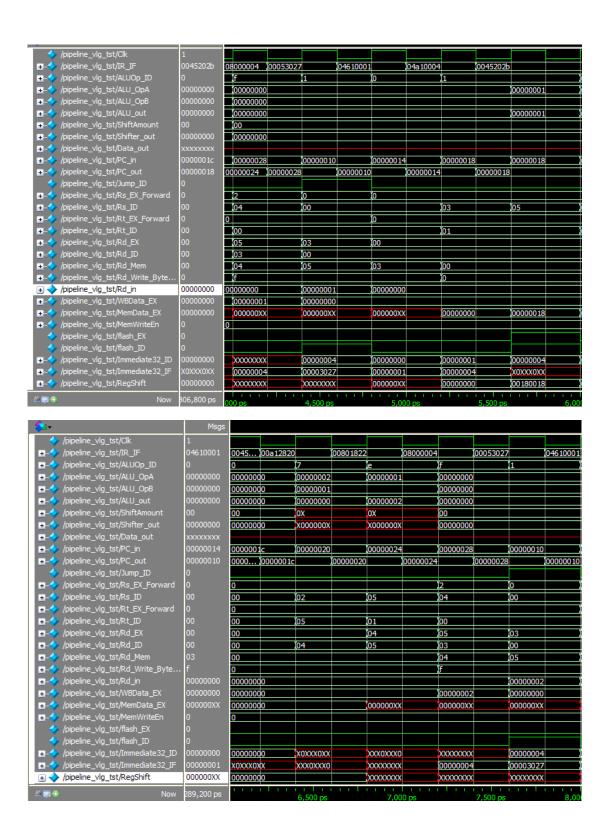
initial:

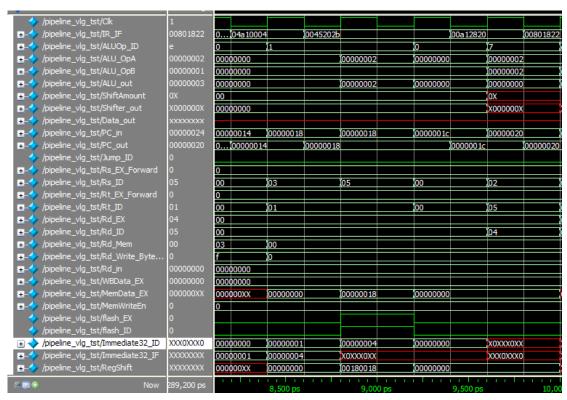
本次测试寄存器全无初始化, 仅用到 r0 = 0;

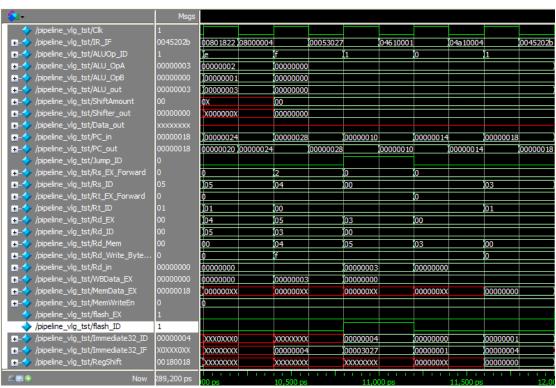
二进制代码:

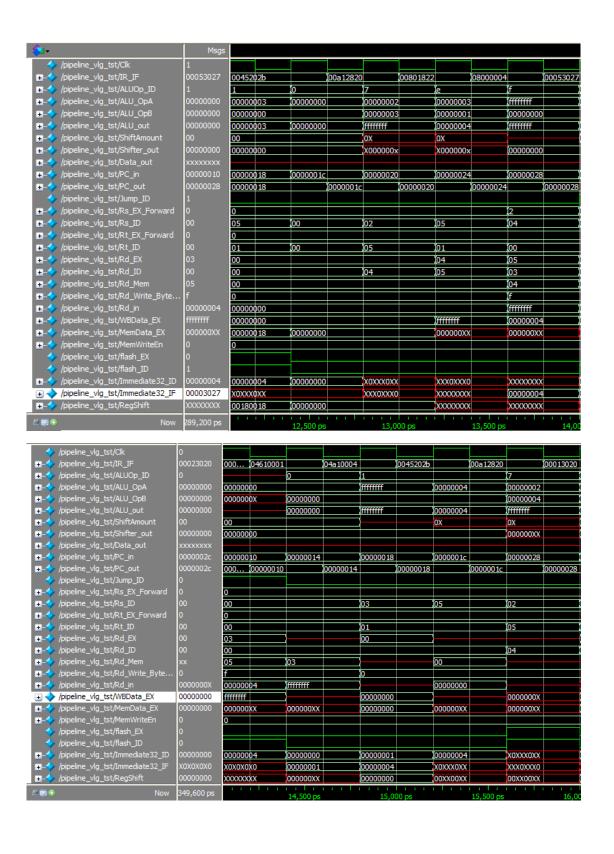
```
ram[0] = 32'b00100000000000010000000000000001; //addi r0 im(1) -> r1
ram[1] = 32'b000000000010001000100000100000; //add
                                                   r1 r1 -> r2
ram[2] = 32'b0000000000100000001100000100000; //add
                                                   r1 r0 -> r3
//addi r0 \text{ im}(0) -> r5
ram[4] = 32'b00000100011100010000000000000001;
                                            //begzal r3 -> 1
ram[5] = 32'b00000100101000010000000000000100; //begz
                                                    r5 -> 4
ram[6] = 32'b00000000101000100010000000101011; //sltu (r5 r2) r4
ram[7] = 32'b00000000101000010010100000100000; //add r5 r1 -> r5
ram[8] = 32'b00000000100000010010100000100010; //sub r4 r1 -> r3
// r1=1 r2=2 r3=-1 r4=-1 r5=4
ram[10] = 32'b000000\_00000\_00001\_00110\_00000100000; //add r0 r1 -> r6
ram[11] = 32'b000000 00000 00010 00110 00000100000; //add r0 r2 -> r6
ram[12] = 32'b000000\_00000\_00011\_00110\_00000100000; //add r0 r3 -> r6
ram[13] = 32'b000000\_00000\_00100\_00110\_00000100000; //add r0 r4 -> r6
ram[14] = 32'b000000\_00000\_00101\_00110\_00000100000; //add r0 r5 -> r6
仿真截图:
```

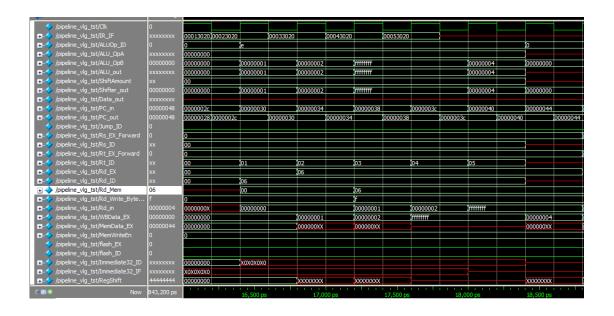












conclusion:

图中可看出,最后的五个 add 得到的寄存器 r1~r5 的值都与预期值相同,连续执行跳转指令以及数据冒险和控制冒险同时发生的情况,该流水线 CPU 都能很好的解决,并无差错。

test-3:

本次测试在 test-2 的基础上加入了 sw, lw 指令,用于测试内存存取与其他类型指令冒险的结合性

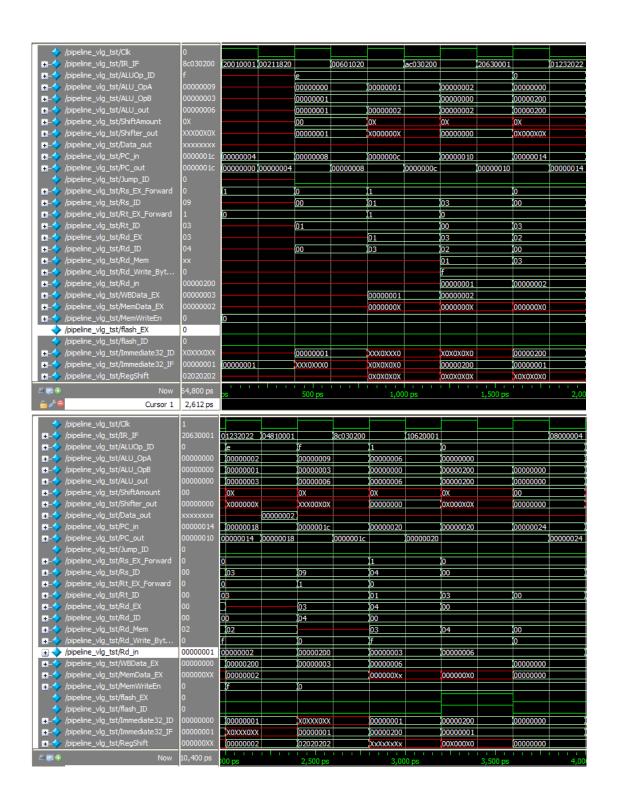
r10为9;

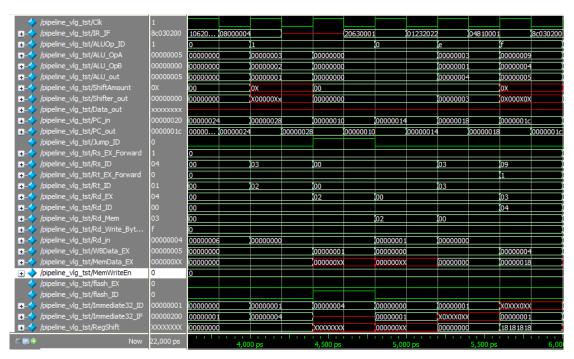
汇编代码的意义是:将 r1 加为 1, r2 加为 2, r3 加为 2, 然后将 r3 存入内存的 512, 然后进入循环,循环里做的是:r3 每回合+1,直到 r3 等于 r10,则 begz 不执行,就不跳过 lw,然后将 r3 从内存 512 中取出前面存入的 2,则执行 beg,跳过 jump,结束循环。

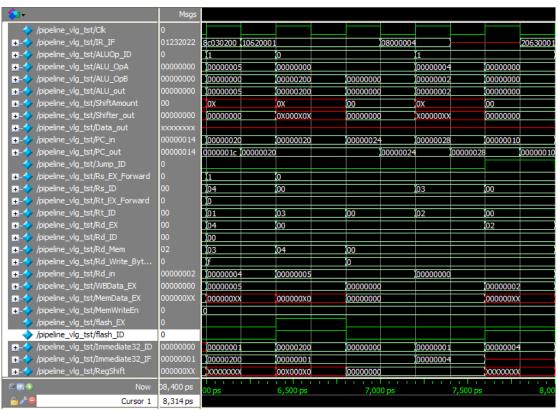
```
ram[0] = 32'b001000\_00000\_00001\_000000000000001; //addi r0 im(1) -> r1
ram[1] = 32'b000000 00001 00001 0001100000100000;
                                                 //add
                                                         r1 r1 -> r3
ram[2] = 32'b000000_00011_00000_000100000100000; //add
                                                         r3 r0 -> r2
ram[3] = \{16b101011\_00000\_00011,16d512\};
                                                  // \text{ sw } \text{ reg}[3] -> DM[512]
ram[4] = 32'b001000\_00011\_00011\_000000000000001; //addi r3 im(1) -> r3
ram[5] = 32'b000000\_01001\_00011\_0010000000100010;
                                                 //sub r10 r3 ->r4
ram[6] = 32'b000001\_00100\_00001\_0000000000000001; //begz r4 -> 1
ram[7] = \{16b100011\_00000\_00011,16d512\};
                                                  //lw DM[512] -> reg[3]
ram[8] = 32'b000100\_00011\_00010\_000000000000001; //beq (r3 == r2)? ->1
```

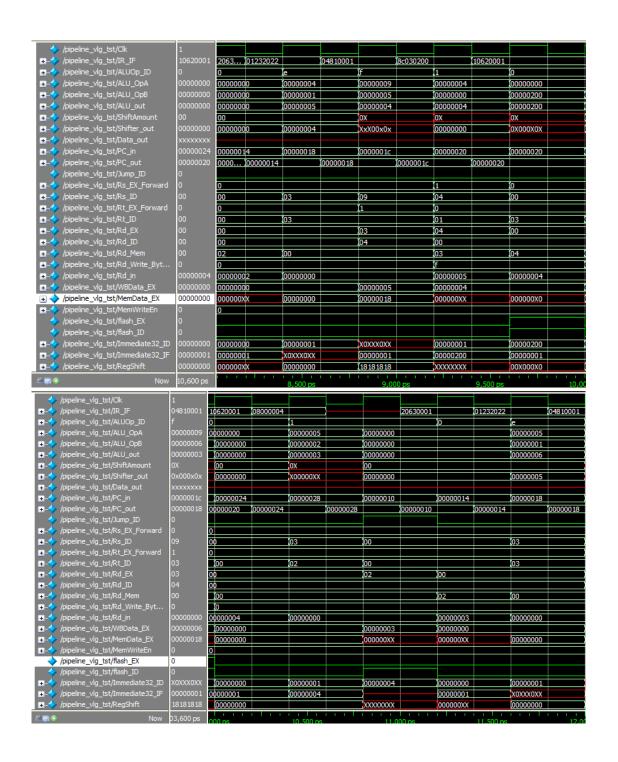
仿真截图:

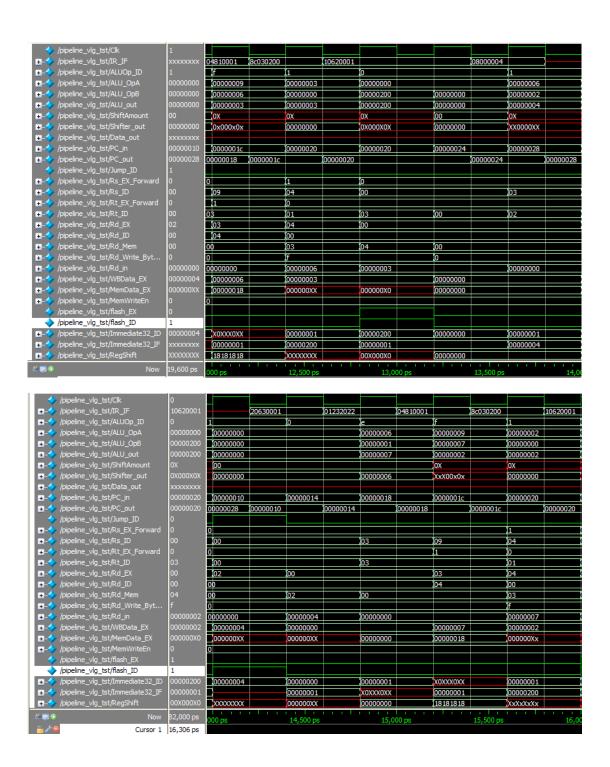
一下截图每张图五个周期,每张图五重复周期



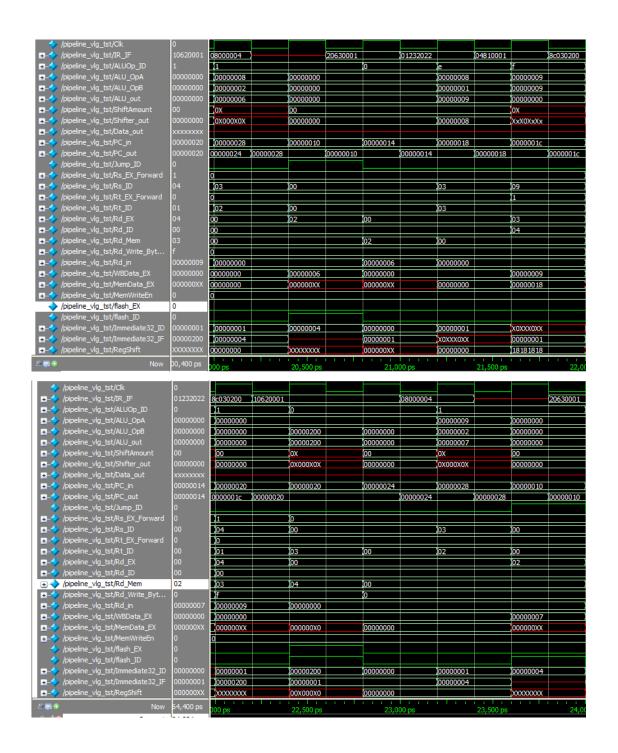




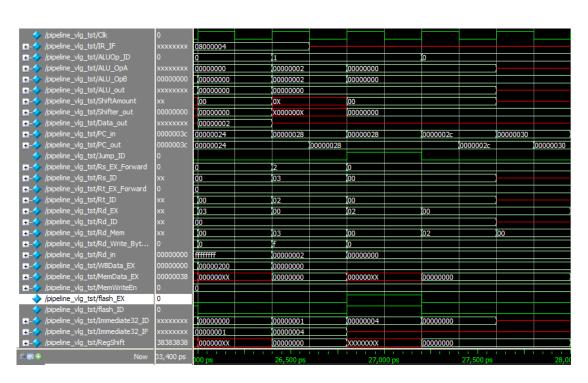




/pipeline_vlg_tst/Clk	lo l									
/pipeline_vlg_tst/IR_IF	04810001	10620001		08000004				20630001		0123202
/pipeline_vlg_tst/ALUOp_ID	e	10		,0000000 T	1			20030001	70	0123202
/pipeline_vlg_tst/ALU_OpA	00000007	100000000			00000007		00000000			
/pipeline_vlg_tst/ALU_OpB	00000001	00000000	200000000		00000002		00000000			
/pipeline_vlg_tst/ALU_out	80000000	00000200	00000000		00000005		00000000			
/pipeline_vlg_tst/ShiftAmount	00	0X	00		0X		00			
/pipeline vlg tst/Shifter out	00000007	0X000X0X	(00000000		xX000Xxx		00000000			
/pipeline_vlg_tst/Data_out	xxxxxxxxx	GAGGGAGA	5555555		XXXXX		00000000			
/pipeline_vlg_tst/PC_in	00000018	200000020	00000024		00000028		00000010		00000014	
/pipeline_vlg_tst/PC_out	00000018	00000020	,0000002	00000024	1	00000028		00000010	1	000000
/pipeline_vlg_tst/Jump_ID	0	00000020		100000021		00000020		200000010		000000
/pipeline_vlg_tst/Rs_EX_Forward	o	70								
/pipeline_vlg_tst/Rs_ID	03	200			03		100			
/pipeline_vlg_tst/Rt_EX_Forward	0	0			103		,00			
/pipeline_vlg_tst/Rt_ID	03	03	200		102		100			
/pipeline_vlg_tst/Rd_EX	00	00	,00		02		02		100	
/pipeline_vlg_tst/Rd_ID	00						02		700	
/ /pipeline_vig_tst/Rd_1D // /pipeline_vlg_tst/Rd_Mem	00	00 104	200						02	
/pipeline_vlg_tst/Rd_Write_Byt		f	100						02	
// /pipeline_vig_tst/Rd_in	00000000	00000002	U		200000000				100000005	
/pipeline_vlg_tst/WBData_EX	00000000		***************************************		,00000000		00000005			
	00000000	00000002 000000X0	00000000				0000000XX		0000000XX	
/pipeline_vlg_tst/MemData_EX	00000000	000000000	00000000				UUUUUUXX		UUUUUUXX	
/pipeline_vlg_tst/MemWriteEn	o o	U								
/pipeline_vlg_tst/flash_EX	0									
/pipeline_vlg_tst/flash_ID	_	V00000000	***********		00000001		00000000		00000000	
/pipeline_vlg_tst/Immediate32_ID		00000200	00000000		00000001		00000004	_	00000000	
/pipeline_vlg_tst/Immediate32_IF	X0XXX0XX 00000000	00000001 00X000X0	222222		00000004				00000001 000000XX	
/pipeline_vlg_tst/RegShift										
I ◆ Now	44,800 ps	000 ps	(000000000 	1 1 1	17,0	00 ps)00000000X	17,500 ps	1 1 1	1
Now /pipeline_vlg_tst/Clk		000 ps	16,500 p	S	17,0	00 ps	XXXXXXXX	17,500 ps		1
	14,800 ps	0000 ps	16,500 p	8c030200		00 ps	, , , ,	17,500 ps		1
/ /pipeline_vlg_tst/Clk / /pipeline_vlg_tst/IR_IF	14,800 ps	000 ps	16,500 p	8c030200		10620001	,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,	17,500 ps		1
/pipeline_vlg_tst/Clk /pipeline_vlg_tst/IR_IF	14,800 ps	000 ps	16,500 p	8c030200		10620001		17,500 ps		1
/pipeline_vlg_tst/Clk /pipeline_vlg_tst/IR_IF /pipeline_vlg_tst/ALUOp_ID	1 xxxxxxxx 1	000 ps	16,500 p	8c030200	1	10620001	0	17,500 ps		1
/pipeline_vig_tst/Clk /pipeline_vig_tst/IR_IF /pipeline_vig_tst/ALUOp_ID /pipeline_vig_tst/ALUOpA	1 xxxxxxxx 1 00000000	000 ps 01232022 (c) (e)	16,500 p	8c030200	1	10620001	0	17,500 ps		0800000
/pipeline_vlg_tst/Clk /pipeline_vlg_tst/R_IF /pipeline_vlg_tst/ALUOp_ID /pipeline_vlg_tst/ALU_OpA /pipeline_vlg_tst/ALU_OpB	1 XXXXXXXXX 1 00000000 00000000 00000000	000 ps 01232022)(e 00000007 00000001	16,500 p	8c030200	1 000000001 00000000	10620001	0 00000000 00000200	17,500 ps	00000000	0800000
/pipeline_vlg_tst/Clk /pipeline_vlg_tst/R. JF /pipeline_vlg_tst/ALUOp_ID /pipeline_vlg_tst/ALU_OpA /pipeline_vlg_tst/ALU_OpB /pipeline_vlg_tst/ALU_Out	1 xxxxxxxxx 1 00000000 00000000 00000000	000 ps 01232022	16,500 p	8c030200	1 00000001 00000000 00000001	10620001	0 00000000 00000200 00000200	17,500 ps	00000000	0800000
/pipeline_vlg_tst/Clk /pipeline_vlg_tst/IR_IF /pipeline_vlg_tst/ALUOp_ID /pipeline_vlg_tst/ALU_OpA /pipeline_vlg_tst/ALU_OpB /pipeline_vlg_tst/ALU_out /pipeline_vlg_tst/ALU_out	1 xxxxxxxx 1 00000000 00000000 00000000 00	01232022 (e 00000007 (00000001 (00000008 00)	16,500 p 16,500 p 16,500 p 16,500 p 10,0000000 10,000000000000000000000000	8c030200	1 00000001 00000000 00000001 0X	10620001	0 00000000 00000200 00000200 0X	17,500 ps	00000000	0800000
/pipeline_vlg_tst/Clk /pipeline_vlg_tst/IR_IF /pipeline_vlg_tst/ALUOp_ID /pipeline_vlg_tst/ALU_OpA /pipeline_vlg_tst/ALU_OpA /pipeline_vlg_tst/ALU_out /pipeline_vlg_tst/ShiftAmount /pipeline_vlg_tst/Shifter_out	1 xxxxxxxx 1 00000000 00000000 00000000 00000000	01232022 (e 00000007 (00000001 (00000008 00)	16,500 p 16,500 p 16,500 p 16,500 p 10,0000000 10,000000000000000000000000	8c030200	1 00000001 00000000 00000001 0X	10620001	0 00000000 00000200 00000200 0X	17,500 ps	00000000	080000
/pipeline_vlg_tst/Clk /pipeline_vlg_tst/IR_IF /pipeline_vlg_tst/ALUOp_ID /pipeline_vlg_tst/ALU_OpA /pipeline_vlg_tst/ALU_out /pipeline_vlg_tst/ALU_out /pipeline_vlg_tst/ShiftAmount /pipeline_vlg_tst/Shifter_out /pipeline_vlg_tst/Shifter_out	1 XXXXXXXXX 1 00000000 00000000 00000000 XXXXXXXX	01232022 % ke 00000007 00000001 00000008 00 0000007	04810001 If 00000009 00000001 00000001	8c030200	1 00000001 00000000 00000001 0X 00000000	10620001	0 00000000 00000200 00000200 0X 0X000X0X	17,500 ps	00000000 00000000 00 00000000	0800000
/pipeline_vlg_tst/Clk /pipeline_vlg_tst/R_IF /pipeline_vlg_tst/ALUOp_ID /pipeline_vlg_tst/ALU_OpA /pipeline_vlg_tst/ALU_OpB /pipeline_vlg_tst/ALU_out /pipeline_vlg_tst/ShiftAmount /pipeline_vlg_tst/ShiftEr_out /pipeline_vlg_tst/Data_out /pipeline_vlg_tst/Pot_in	1 XXXXXXXXX 1 00000000 00000000 00000000 XXXXXXXX	01232022 (c) le (0000007 (0000008 0) 00000007 (0000007 (00000018 0)	04810001 If 00000009 00000001 00000001	8c030200	1 00000001 00000000 00000001 0X 00000000	10620001	0 00000000 00000200 00000200 0X 0X000X0X	17,500 ps	00000000 00000000 00 00000000	0800000
/pipeline_vlg_tst/Clk /pipeline_vlg_tst/R_JF /pipeline_vlg_tst/ALUOp_ID /pipeline_vlg_tst/ALU_OpA /pipeline_vlg_tst/ALU_OpA /pipeline_vlg_tst/ALU_OpB /pipeline_vlg_tst/ALU_out /pipeline_vlg_tst/Shifter_out /pipeline_vlg_tst/Shifter_out /pipeline_vlg_tst/PC_in /pipeline_vlg_tst/PC_in /pipeline_vlg_tst/PC_out /pipeline_vlg_tst/PC_out /pipeline_vlg_tst/PC_out	1 XXXXXXXXX 1 00000000 00000000 00000000 XXXXXXXX	01232022 (c) le (0000007 (0000008 0) 00000007 (0000007 (00000018 0)	04810001 If 00000009 00000001 00000001	8c030200	1 00000001 00000000 00000001 0X 00000000	[10620001 10620001	0 00000000 00000200 00000200 0X 0X000X0X	17,500 ps	00000000 00000000 00 00000000	080000
/pipeline_vlg_tst/Clk /pipeline_vlg_tst/R_IF /pipeline_vlg_tst/ALUOp_ID /pipeline_vlg_tst/ALU_OpA /pipeline_vlg_tst/ALU_OpA /pipeline_vlg_tst/ALU_out /pipeline_vlg_tst/ShiftAmount /pipeline_vlg_tst/ShiftAmount /pipeline_vlg_tst/Data_out /pipeline_vlg_tst/PC_in /pipeline_vlg_tst/PC_out	1 XXXXXXXX 1 00000000 00000000 00000000 0XXXXXXXX	01232022 (c) le (0000007 (0000008 0) 00000007 (0000007 (00000018 0)	04810001 16,500 p	\$c030200	00000001 00000000 00000000 00000000 000000	(10620001 (10620001	0 00000000 00000200 00000200 0X 0X000X0X	17,500 ps	00000000 00000000 00 00000000	080000
/pipeline_vig_tst/Cilk /pipeline_vig_tst/IR_IF /pipeline_vig_tst/ALUOp_ID /pipeline_vig_tst/ALU_OpA /pipeline_vig_tst/ALU_OpA /pipeline_vig_tst/ALU_out /pipeline_vig_tst/ShiftAmount /pipeline_vig_tst/ShiftAmount /pipeline_vig_tst/Data_out /pipeline_vig_tst/PC_in /pipeline_vig_tst/PC_in /pipeline_vig_tst/PC_out /pipeline_vig_tst/Sump_ID /pipeline_vig_tst/Rs_IT /pipeline_vig_tst/Rs_IT	1 XXXXXXXXX 1 00000000 00000000 00000000 0XXXXXXXX	01232022	04810001 If 00000009 00000001 00000001	(Sc030200	11 00000001 00000000 00000001 0X 00000000	(10620001 (10620001	0 00000000 00000200 00000200 0X 0X0000X0X	17,500 ps	00000000 00000000 00 00000000	080000
/pipeline_vlg_tst/Clk /pipeline_vlg_tst/R_IF /pipeline_vlg_tst/ALUOp_ID /pipeline_vlg_tst/ALU_OpA /pipeline_vlg_tst/ALU_OpA /pipeline_vlg_tst/ALU_out /pipeline_vlg_tst/ShiftAmount /pipeline_vlg_tst/ShiftAmount /pipeline_vlg_tst/Po_tout /pipeline_vlg_tst/PC_in /pipeline_vlg_tst/PC_in /pipeline_vlg_tst/PC_out /pipeline_vlg_tst/RC_in /pipeline_vlg_tst/RS_ID /pipeline_vlg_tst/Rs_ID /pipeline_vlg_tst/Rt_EX_Forward /pipeline_vlg_tst/Rt_EX_Forward	1 XXXXXXXXX 1 00000000 00000000 00000000 0XXXXXXXX	01232022	04810001	38c030200	11 00000001 00000000 00000001 0X 00000000	(10620001	0 00000000 00000200 00000200 0X 0X0000X0X	17,500 ps	00000000 00000000 00 00000000 000000024	080000
/pipeline_vig_tst/Cilk /pipeline_vig_tst/IR_IF /pipeline_vig_tst/ALUOp_ID /pipeline_vig_tst/ALU_OpA /pipeline_vig_tst/ALU_OpA /pipeline_vig_tst/ALU_out /pipeline_vig_tst/ShiftAmount /pipeline_vig_tst/ShiftAmount /pipeline_vig_tst/Data_out /pipeline_vig_tst/PC_in /pipeline_vig_tst/PC_in /pipeline_vig_tst/PC_out /pipeline_vig_tst/Sump_ID /pipeline_vig_tst/Rs_IT /pipeline_vig_tst/Rs_IT	1 XXXXXXXX 1 00000000 00000000 00000000 00000000	01232022	04810001	(\$c030200 (\$0000001c	1 00000000 00000000 00000001 0X 00000000	(10620001	0 00000000 00000200 00000200 0X 0X000X0X	17,500 ps	00000000 00000000 00 00000000	080000
/pipeline_vig_tst/Clk /pipeline_vig_tst/R, IF /pipeline_vig_tst/ALUOp_ID /pipeline_vig_tst/ALU_OpA /pipeline_vig_tst/ALU_OpA /pipeline_vig_tst/ALU_OpB /pipeline_vig_tst/ALU_out /pipeline_vig_tst/Shifter_out /pipeline_vig_tst/Shifter_out /pipeline_vig_tst/Pata_out /pipeline_vig_tst/PC_out /pipeline_vig_tst/PC_out /pipeline_vig_tst/Rs_EX_Forward /pipeline_vig_tst/Rs_ID /pipeline_vig_tst/Rt_EX_Forward /pipeline_vig_tst/Rt_EX_Forward /pipeline_vig_tst/Rt_EX_Forward /pipeline_vig_tst/Rt_EX_Forward /pipeline_vig_tst/Rt_EX_Forward /pipeline_vig_tst/Rt_EX_Forward /pipeline_vig_tst/Rt_EX_Forward /pipeline_vig_tst/Rt_EX_Forward /pipeline_vig_tst/Rt_EX_Forward	1 XXXXXXXX 1 00000000 00000000 00000000 0XXXXXXXX	01232022	04810001 f 00000008 00000001 00000018 00000018 00000018 000000018 000000018 000000018 000000018 000000018 000000018 000000018 0000000018 0000000000	38c030200	1 000000001 000000000 00X 000000000 00000000	(10620001	00 00000000 00000200 00000200 0X 0X 0X0000X0X	17,500 ps	00000000 00000000 00 00000000 000000024	080000
/pipeline_v/g_tst/Clk /pipeline_v/g_tst/R_JF /pipeline_v/g_tst/RLUOp_ID /pipeline_v/g_tst/ALU_OpA /pipeline_v/g_tst/ALU_OpA /pipeline_v/g_tst/ALU_OpA /pipeline_v/g_tst/ALU_out /pipeline_v/g_tst/ShiftAmount /pipeline_v/g_tst/Fota_out /pipeline_v/g_tst/PC_out /pipeline_v/g_tst/PC_out /pipeline_v/g_tst/RS_ID /pipeline_v/g_tst/RS_ID /pipeline_v/g_tst/RS_ID /pipeline_v/g_tst/RS_ID /pipeline_v/g_tst/RS_ID /pipeline_v/g_tst/RS_ID	1 XXXXXXXX 1 00000000 00000000 00000000 0XXXXXXXX	01232022	04810001	\$c030200	11 00000001 00000000 00000001 0X 00000000	(10620001 (10620001	00 00000000 00000200 00000200 0X 0X 0X0000X0X	17,500 ps	00000000 00000000 00 00000000 000000024	080000
/pipeline_vig_tst/Cilk /pipeline_vig_tst/R_IF /pipeline_vig_tst/RLUOp_ID /pipeline_vig_tst/RLU_OpA /pipeline_vig_tst/ALU_OpA /pipeline_vig_tst/ALU_OpA /pipeline_vig_tst/ALU_out /pipeline_vig_tst/ShiftAmount /pipeline_vig_tst/ShiftAmount /pipeline_vig_tst/PC_in /pipeline_vig_tst/PC_in /pipeline_vig_tst/PC_in /pipeline_vig_tst/RC_in /pipeline_vig_tst/RS_ID /pipeline_vig_tst/RS_ID /pipeline_vig_tst/Rt_EX_Forward /pipeline_vig_tst/Rt_EX_Forward /pipeline_vig_tst/Rt_EX_Forward /pipeline_vig_tst/Rd_EX /pipeline_vig_tst/Rd_EX /pipeline_vig_tst/Rd_EX /pipeline_vig_tst/Rd_ID /pipeline_vig_tst/Rd_Mem	1 XXXXXXXX 1 00000000 00000000 00000000 00000000	01232022 % %e	04810001 f 00000008 00000001 00000018 00000018 00000018 000000018 000000018 000000018 000000018 000000018 000000018 000000018 0000000018 0000000000	(\$c030200	11 000000001 000000000 000000000 0000000	(10620001 (10620001	0 00000000 00000200 00000200 0X 0X 0X 0X 0X 0X 0X 0X 0X 0X 0X 0X 0	17,500 ps	(20000000 (200000000 (200000000 (200000000	080000
/pipeline_vlg_tst/Clk /pipeline_vlg_tst/R. JF /pipeline_vlg_tst/RLUOp_ID /pipeline_vlg_tst/ALU_OpA /pipeline_vlg_tst/ALU_OpA /pipeline_vlg_tst/ALU_OpA /pipeline_vlg_tst/ALU_out /pipeline_vlg_tst/ShiftAmount /pipeline_vlg_tst/ShiftAmount /pipeline_vlg_tst/PC_in /pipeline_vlg_tst/PC_in /pipeline_vlg_tst/PC_out /pipeline_vlg_tst/PC_out /pipeline_vlg_tst/RS_EX_Forward /pipeline_vlg_tst/RS_EX_Forward /pipeline_vlg_tst/Rt_ID /pipeline_vlg_tst/Rt_ID /pipeline_vlg_tst/Rd_EX /pipeline_vlg_tst/Rd_ID /pipeline_vlg_tst/Rd_ID /pipeline_vlg_tst/Rd_ID /pipeline_vlg_tst/Rd_ID /pipeline_vlg_tst/Rd_ID /pipeline_vlg_tst/Rd_ID /pipeline_vlg_tst/Rd_Whem /pipeline_vlg_tst/Rd_Write_Byt	1 XXXXXXXX 1 00000000 00000000 00000000 00000000	01232022	04810001 f 00000008 00000001 00000018 00000018 00000018 000000018 000000018 000000018 000000018 000000018 000000018 000000018 0000000018 0000000000	(0000001c	11 00000001 00000000 00000000 00000000 000000	(10620001	00 00000000 00000200 00000200 0X 0X000X0X 000000020 00 00	17,500 ps	00000000 00000000 00000000 000000024	080000
/pipeline_vlg_tst/Clk /pipeline_vlg_tst/RIR_JF /pipeline_vlg_tst/ALUOp_ID /pipeline_vlg_tst/ALU_OpA /pipeline_vlg_tst/ALU_OpA /pipeline_vlg_tst/ALU_OpA /pipeline_vlg_tst/ALU_OpA /pipeline_vlg_tst/ALU_opt /pipeline_vlg_tst/Folat_out /pipeline_vlg_tst/PC_out /pipeline_vlg_tst/PC_out /pipeline_vlg_tst/PC_out /pipeline_vlg_tst/RS_ID /pipeline_vlg_tst/RS_ID /pipeline_vlg_tst/RS_ID /pipeline_vlg_tst/Rt_ID /pipeline_vlg_tst/Rt_ID /pipeline_vlg_tst/Rt_ID /pipeline_vlg_tst/Rd_IX	1 XXXXXXXXX 1 00000000 00000000 00000000 00000000	000 ps 01232022	04810001	(\$c030200 (\$0000001c	11 00000001 00000000 00000000 00000000 000000	(10620001	0 00000000 00000200 00000200 0X 0X 0X 0X 0X 0X 0X 0X 0X 0X 0X 0X 0	17,500 ps	00000000 00000000 00 00000000 000000024	0800000
/pipeline_vlg_tst/Clk /pipeline_vlg_tst/IR_IF /pipeline_vlg_tst/IR_IF /pipeline_vlg_tst/ALU_OpA /pipeline_vlg_tst/ALU_OpB /pipeline_vlg_tst/ALU_OpB /pipeline_vlg_tst/ShiftAmount /pipeline_vlg_tst/ShiftAmount /pipeline_vlg_tst/ShiftAmount /pipeline_vlg_tst/PC_in /pipeline_vlg_tst/PC_out /pipeline_vlg_tst/PC_out /pipeline_vlg_tst/PC_EX_Forward /pipeline_vlg_tst/Rs_EX_Forward /pipeline_vlg_tst/Rs_ID /pipeline_vlg_tst/Rt_EX_Forward /pipeline_vlg_tst/Rd_ID /pipeline_vlg_tst/Rd_ID /pipeline_vlg_tst/Rd_In /pipeline_vlg_tst/Rd_In /pipeline_vlg_tst/Rd_In /pipeline_vlg_tst/WBData_EX	1 XXXXXXXXX 1 00000000 00000000 00000000 0XXXXXXX 00000028 1 0 00 00 00 00 00 00 00 00	000 ps 01232022	04810001 If 00000000 JON 00000001 00000018 00000018 000000018	(\$c030200 (\$0000001c	11 000000001 000000001 00X 000000001 000000020 11 004 00 00 003 if f	(10620001	00 000000000 00000200 00000200 0X 0X 0X0000X0X 00000020 00 03 00 04	17,500 ps	(00000000 (00000000 (00000000 (000000024	0800000
/pipeline_vlg_tst/Clk /pipeline_vlg_tst/IR_IF /pipeline_vlg_tst/ALUOp_ID /pipeline_vlg_tst/ALU_OpA /pipeline_vlg_tst/ALU_OpA /pipeline_vlg_tst/ALU_OpA /pipeline_vlg_tst/ALU_out /pipeline_vlg_tst/Shifter_out /pipeline_vlg_tst/Shifter_out /pipeline_vlg_tst/Data_out /pipeline_vlg_tst/Data_out /pipeline_vlg_tst/PC_in /pipeline_vlg_tst/PC_out /pipeline_vlg_tst/PC_out /pipeline_vlg_tst/RS_EX_Forward /pipeline_vlg_tst/RS_EX_Forward /pipeline_vlg_tst/Rt_EX_Forward /pipeline_vlg_tst/Rd_EX /pipeline_vlg_tst/Rd_EX /pipeline_vlg_tst/Rd_Mem /pipeline_vlg_tst/Rd_Mitte_Byt /pipeline_vlg_tst/Rd_Data_EX /pipeline_vlg_tst/MEData_EX /pipeline_vlg_tst/MemData_EX	1 XXXXXXXXX 1 00000000 00000000 00000000 XXXXXXXX	01232022 % %e	04810001	(\$c030200 (\$0000001c	11 00000001 00000000 00000000 00000000 000000	(10620001	00 00000000 00000200 00000200 0X 0X000X0X 000000020 00 00	17,500 ps	00000000 00000000 00 00000000 000000024	0800000
/pipeline_vlg_tst/Clk /pipeline_vlg_tst/IR_IF /pipeline_vlg_tst/IR_IF /pipeline_vlg_tst/ALUOp_ID /pipeline_vlg_tst/ALU_OpA /pipeline_vlg_tst/ALU_OpA /pipeline_vlg_tst/ALU_OpA /pipeline_vlg_tst/ShiftAmount /pipeline_vlg_tst/ShiftAmount /pipeline_vlg_tst/ShiftAmount /pipeline_vlg_tst/ShiftAmount /pipeline_vlg_tst/PC_in /pipeline_vlg_tst/PC_out /pipeline_vlg_tst/PC_out /pipeline_vlg_tst/RS_EX_Forward /pipeline_vlg_tst/RS_EX_Forward /pipeline_vlg_tst/RS_EX_Forward /pipeline_vlg_tst/Rd_ID	1 XXXXXXXXX 1 00000000 00000000 00000000 XXXXXXXX	000 ps 01232022	04810001 If 00000000 JON 00000001 00000018 00000018 000000018	(\$c030200 (\$0000001c	11 000000001 000000001 00X 000000001 000000020 11 004 00 00 003 if f	(10620001	00 000000000 00000200 00000200 0X 0X 0X0000X0X 00000020 00 03 00 04	17,500 ps	(00000000 (00000000 (00000000 (000000024	0800000
/pipeline_vlg_tst/Clk /pipeline_vlg_tst/R. JF /pipeline_vlg_tst/ALUOp_ID /pipeline_vlg_tst/ALU_OpA /pipeline_vlg_tst/ALU_OpA /pipeline_vlg_tst/ALU_OpA /pipeline_vlg_tst/ALU_OpA /pipeline_vlg_tst/ALU_out /pipeline_vlg_tst/ShiftAmount /pipeline_vlg_tst/PC_in /pipeline_vlg_tst/PC_in /pipeline_vlg_tst/PC_out /pipeline_vlg_tst/PC_out /pipeline_vlg_tst/RS_EX_Forward /pipeline_vlg_tst/RS_EX_Forward /pipeline_vlg_tst/Rt_ID /pipeline_vlg_tst/Rt_ID /pipeline_vlg_tst/Rd_ID /pipeline_vlg_tst/Rd_ID /pipeline_vlg_tst/Rd_ID /pipeline_vlg_tst/Rd_ID /pipeline_vlg_tst/Rd_In /pipeline_vlg_tst/Rd_Write_Byt /pipeline_vlg_tst/WBData_EX /pipeline_vlg_tst/WBData_EX /pipeline_vlg_tst/MemData_EX /pipeline_vlg_tst/MemData_EX /pipeline_vlg_tst/Hash_EX	1 XXXXXXXXX 1 00000000 00000000 00000000 XXXXXXXX	01232022 % %e	04810001 If 00000000 JON 00000001 00000018 00000018 000000018	(\$c030200 (\$0000001c	11 000000001 000000001 00X 000000001 000000020 11 004 00 00 003 if f	(10620001	00 000000000 00000200 00000200 0X 0X 0X0000X0X 00000020 00 03 00 04	17,500 ps	(00000000 (00000000 (00000000 (000000024	0800000
/pipeline_vlg_tst/Clk /pipeline_vlg_tst/R_IF /pipeline_vlg_tst/RLUOp_ID /pipeline_vlg_tst/ALUOp_ID /pipeline_vlg_tst/ALU_OpA /pipeline_vlg_tst/ALU_OpA /pipeline_vlg_tst/ALU_OpA /pipeline_vlg_tst/ShiftAmount /pipeline_vlg_tst/ShiftAmount /pipeline_vlg_tst/ShiftAmount /pipeline_vlg_tst/PC_out /pipeline_vlg_tst/PC_out /pipeline_vlg_tst/PC_out /pipeline_vlg_tst/RS_EX_Forward /pipeline_vlg_tst/RS_ID /pipeline_vlg_tst/Rd_ID /pipeline_vlg_tst/Rd_ID /pipeline_vlg_tst/Rd_ID /pipeline_vlg_tst/Rd_ID /pipeline_vlg_tst/Rd_ID /pipeline_vlg_tst/Rd_in /pipeline_vlg_tst/Rd_in /pipeline_vlg_tst/Rd_in /pipeline_vlg_tst/Rd_in /pipeline_vlg_tst/Rd_in /pipeline_vlg_tst/Rd_in /pipeline_vlg_tst/RemData_EX /pipeline_vlg_tst/RemWriteEn /pipeline_vlg_tst/flash_ID	14,800 ps 1	01232022	000000018	(30000001c	11 000000001 000000000 000000000 0000000	00000020	00 00000000 00000200 00000200 0X 0X 0X 0X 0X 0X 0X 0X 0X 0X 0X 0X 0	17,500 ps	00000000 00000000 00000000 000000024	0800000
/pipeline_vlg_tst/Clk /pipeline_vlg_tst/ALUOp_ID /pipeline_vlg_tst/ALUOpA /pipeline_vlg_tst/ALU_OpA /pipeline_vlg_tst/ALU_OpA /pipeline_vlg_tst/ALU_OpA /pipeline_vlg_tst/ALU_OpA /pipeline_vlg_tst/ShiftAmount /pipeline_vlg_tst/ShiftAmount /pipeline_vlg_tst/PC_in /pipeline_vlg_tst/PC_out /pipeline_vlg_tst/PC_out /pipeline_vlg_tst/PC_out /pipeline_vlg_tst/PS_ID /pipeline_vlg_tst/RS_ID /pipeline_vlg_tst/RS_ID /pipeline_vlg_tst/Rt_EX_Forward /pipeline_vlg_tst/Rd_EX /pipeline_vlg_tst/Rd_Mem /pipeline_vlg_tst/Rd_Mem /pipeline_vlg_tst/Rd_Mem /pipeline_vlg_tst/WBData_EX /pipeline_vlg_tst/MemData_EX /pipeline_vlg_tst/MemData_EX /pipeline_vlg_tst/flash_EX	1 XXXXXXXXX 1 00000000 00000000 00000000 XXXXXXXX	01232022 % %e	04810001 If 00000000 JON 00000001 00000018 00000018 000000018	(\$c030200	11 000000001 000000001 00X 000000001 000000020 11 004 00 00 003 if f	00000020	00 000000000 00000200 00000200 0X 0X 0X0000X0X 00000020 00 03 00 04	17,500 ps	(00000000 (00000000 (00000000 (000000024	0800000



<pre>/pipeline_vlg_tst/Clk</pre>	1										
- → /pipeline_vlg_tst/IR_IF	08000004	20630001	01232022		04810001		8c030200		10620001		08000004
- → /pipeline_vlg_tst/ALUOp_ID	0	0		e		(f		1		0	
+ /pipeline_vlg_tst/ALU_OpA	00000000	00000000		00000009				rrrrrrr		00000000	
+- /pipeline_vlg_tst/ALU_OpB	00000000	00000000		00000001		0000000a		00000000		00000200	
- - / pipeline_vlg_tst/ALU_out	00000000	00000000		0000000a		rrrrrrr		rrrrrrr		00000200	
- / pipeline_vlg_tst/ShiftAmount	00	00				0X				0X	
- pipeline_vlg_tst/Shifter_out	00000000	00000000		00000009		Xx00XxXx		00000000		0X000X0X	
Ⅲ - / /pipeline_vlg_tst/Data_out	00000002										
⊞ – ∜ /pipeline_vlg_tst/PC_in	00000024	00000014		00000018		0000001c		00000020		00000024	
	00000024	00000010	00000014		00000018		0000001c		00000020		00000024
<pre>/pipeline_vlg_tst/Jump_ID</pre>	0										
- / pipeline_vlg_tst/Rs_EX_Forward	0	0						1		0	
 /pipeline_vlg_tst/Rs_ID	00	00		03		09		04		00	
- - / pipeline_vlg_tst/Rt_EX_Forward	0	0				1		0			
- - / pipeline_vlg_tst/Rt_ID	00	00		03				01		03	
 /pipeline_vlg_tst/Rd_EX	03	00				03		04		00	
 /pipeline_vlg_tst/Rd_ID	00	00				04		00			
/pipeline_vlg_tst/Rd_Mem	00	02		00				03		04	
≖ - / /pipeline_vlg_tst/Rd_Write_Byt	0	0						f			
≖– <pre>/pipeline_vlg_tst/Rd_in</pre>	HHHHH	00000007		00000000				0000000a		rrrrrrr	
 /pipeline_vlg_tst/WBData_EX	00000200	00000000				0000000a		fffffff			
≖ – ∜ /pipeline_vlg_tst/MemData_EX	000000XX	000000XX		00000000		00000018		000000XX		000000X0	
≖ - <pre>/pipeline_vlg_tst/MemWriteEn</pre>	0	0									
/pipeline_vlg_tst/flash_EX	0										
<pre>/pipeline_vlg_tst/flash_ID</pre>	0										
≖ - <pre>/pipeline_vlg_tst/Immediate32_ID</pre>	00000000	00000000		00000001		XOXXXOXX		00000001		00000200	
/pipeline_vlg_tst/Immediate32_IF	00000001	00000001		XOXXXOXX		00000001		00000200		00000001	
 /pipeline_vlg_tst/RegShift	000000XX	000000XX		00000000		18181818		XXXXXXXXX		00X000X0	
△ ■ • Now	35,600 ps	000 ps		24,500 ps		25,0	00 ps		25,500 ps		26,0



conclusion:

图中可看出,连续执行跳转指令以及数据冒险和控制冒险同时发生的情况,该流水线 CPU 都能很好的解决,并无差错,在其中插入从内存中存取也没什么问题。

三、程序性检测:

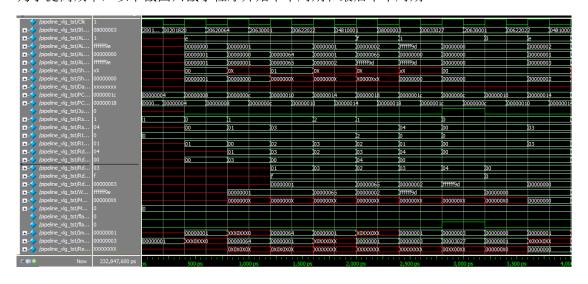
计数器:

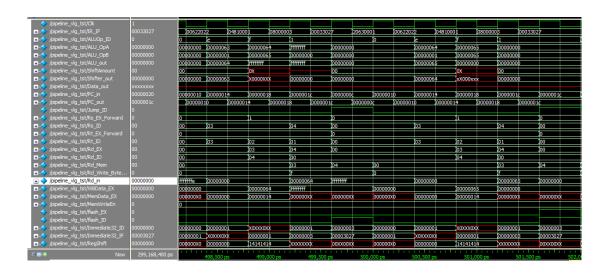
程序把 r3 从 1 加到 100

二进制代码:

仿真截图:

为了提高效率,以下截图只截了程序开始十个周期和最后十个周期





冒泡排序:

寄存器 r1~r5 五个数据进行冒泡排序

初始寄存器的值为

```
register[0] = 0;
register[1] = 1;
register[2] = 5;
register[3] = 4;
register[4] = 3;
register[5] = 2;
```

然后进行冒泡排序

即进行两两比较,若后者小,就交换,反之不交换,如此进行四次。

若后者大,采用 begz 指令来跳过交换过程。最后采用 j 指令来实现循环执行,循环四次之后,采用一个 begz 指令过 j 指令,完成排序。

二进制代码:

```
ram[0] = 32'b001000_00000_00111_0000000000000100; //addi r0 im(4) -> r7
ram[1] = 32'b000000_00000_00111_0100000000100010; //sub r0 r7 -> r8
ram[2] = 32'b001000_01000_01000_000000000000001; //addi r8 im(1) -> r8

ram[3] = 32'b000000_00010_00001_0101000000100010; //sub r2 r1 -> r10
ram[4] = 32'b000001_01010_00001_0000000000000011; //bgez r10 -> 3
ram[5] = 32'b000000_00000_00001_0100100000100000; //add r0 r1 -> r9
ram[6] = 32'b000000_00000_00010_00001000001; //add r0 r2 -> r1
ram[7] = 32'b000000_00000_01001_000100000100000; //add r0 r9 -> r2

ram[8] = 32'b000000_00011_00010_010100000100000; //sub r3 r2 -> r10
ram[9] = 32'b000001_01010_00001_000000000000001; //sub r3 r2 -> r10
ram[10] = 32'b000000_00000_00010_0100100000100000; //add r0 r2 -> r9
ram[10] = 32'b000000_00000_00010_0100100000100000; //add r0 r2 -> r9
ram[11] = 32'b000000_00000_00011_000100000100000; //add r0 r3 -> r2
```

```
ram[12] = 32'b000000\_00000\_01001\_0001100000100000; //add r0 r9 -> r3
ram[13] = 32'b000000 00100 00011 0101000000100010; //sub r4 r3 -> r10
ram[14] = 32'b000001\_01010\_00001\_000000000000011; //bgez r10 -> 3
ram[15] = 32'b000000 00000 00011 0100100000100000; //add r0 r3 -> r9
ram[16] = 32'b000000_00000_00100_0001100000100000; //add r0 r4 -> r3
ram[17] = 32'b000000 00000 01001 0010000000100000; //add r0 r9 -> r4
ram[18] = 32'b000000 00101 00100 0101000000100010; //sub r5 r4 -> r10
ram[20] = 32'b000000\_00000\_00100\_0100100000100000; //add r0 r4 -> r9
ram[21] = 32'b000000\_00000\_00101\_001000000100000; //add r0 r5 -> r4
ram[22] = 32'b000000\_00000\_01001\_0010100000100000; //add r0 r9 -> r5
ram[24] = 32'b000010\_00000\_000000000000000000010; //j
                                                jump to 2
ram[25] = 32'b001000_00000_11110_0000000000001111; //addi r0 im(15) -> r30
ram[26] = 32'b001000\_00001\_00111\_000000000000000000;
                                            //addi r1 im(0) -> r7
//addi \ r2 \ im(0) -> r7
ram[28] = 32'b001000\_00011\_00111\_000000000000000000;
                                            //addi r3 im(0) -> r7
//addi r4 im(0) -> r7
//addi r5 im(0) -> r7
```

最后这五行 addi 是后来为了方便看出排序的正确性加上的,在最后一张截图可以看到 r1 到 r5 的值是按顺序排好的。

仿真截图:

每张图有十个周期,每张图的周期都不重复。

<pre>/pipeline_vlg_tst/Clk</pre>	1																				
-//////////////////////////////////	00031027	20070	0007402	2	21080001		0041502	2	05410003		00014827	7	00020827		00625022		05410003		00024827	7	0003
⊢ ∜ /pipeline_vlg_tst/ALUOp_ID	1			e		f		e		f		1				0		f		1	
⊢/pipeline_vlg_tst/ALU_OpA	00000000		-	00000000)			ffffffc		00000005	j	00000004	4	00000000)			00000004		fffffff	
⊢/pipeline_vlg_tst/ALU_OpB	00000005			00000004	•	0000000	4	0000000	1	00000001		00000000)	0000000		00000000		00000005	,	00000000	0
🛶 /pipeline_vlg_tst/ALU_out	ШШ			00000004		fffffffc		fffffffd		00000004		00000004	4	ffffffff		00000000		ffffffff		fffffff	
⊢/pipeline_vlg_tst/ShiftAmount	00		-	00				хX		0X		0X		00				0X			
⊢🧇 /pipeline_vlg_tst/Shifter_out	00000005					0000000	4			XX0000XX		00000000)	0000000		00000000		X00000XX		00000000	0
⊢ ∜ /pipeline_vlg_tst/Data_out	xxxxxxxx																				
⊢ ∜ /pipeline_vlg_tst/PC_in	00000030	0000000	4	00000008	3	0000000	c	0000001	0	00000014		00000018	3	00000020		00000024		00000028		00000020	d
⊢ ∜ /pipeline_vlg_tst/PC_out	0000002c	00000	0000000	4	00000008	3	0000000		00000010		00000014		00000018		00000020		00000024		00000028		000
<pre>/pipeline_vlg_tst/Jump_ID</pre>	0																				
⊢ ∜ /pipeline_vlg_tst/Rs_EX_Forward	0	1		0				1		0		1		0						1	
/ /pipeline_vlg_tst/Rs_ID	00		-	00				08		02		0a		00				03		0a	
⊢/pipeline_vlg_tst/Rt_EX_Forward	0	2		Ю		1		0													
⊢ ∜ /pipeline_vlg_tst/Rt_ID	02			07				08		01						00		02		01	
	00					07		08				0a		00						0a	
	09			00		08		00		0a		00		09		00		0a		00	
	0a							07		08				0a		00					
⊢- <pre>/pipeline_vlg_tst/Rd_Write_Byt</pre>	f							f								0					
⊢- <pre>/pipeline_vlg_tst/Rd_in</pre>	ffffffff							0000000	4	ffffffc		fffffffd		0000000	•			00000000			
⊢ ∜ /pipeline_vlg_tst/WBData_EX	ffffffff					0000000	4	ffffffc		fffffffd		00000004	4			00000000				fffffff	
	000000XX							0000000	X			000000XX	X .	000000XX		00000000				000000XX	X
⊢ ∜ /pipeline_vlg_tst/MemWriteEn	0	0																			
<pre>/pipeline_vlg_tst/flash_EX</pre>	0																				
<pre>/pipeline_vlg_tst/flash_ID</pre>	0																				
⊢🦴 /pipeline_vlg_tst/Immediate32_ID	00004827			00000004		XOXXXOX	X	0000000		XOXXXOXX		00000003	3	0000482	7	00000000		XOXXXOXX		00000003	3
├�️ /pipeline_vlg_tst/Immediate32_IF	00001027	0000000	4	xoxxxox		0000000	i	XOXXXOX	X	00000003	3	00004827	7	0000082	7	XOXXXOX		00000003		0000482	7
								OXOXOXO				xxxxxxxx		00XX00X		00000000		_		xxxxxxxx	

A (-ilil- +-+(-)).	Msgs																				_
/ /pipeline_vlg_tst/Clk	00052027	00004	0000100		0000500		0544000		00000400		0004400		0000000		00-45000		05440000		2004402		10001
├── /pipeline_vlg_tst/IR_IF ├── /pipeline_vlg_tst/ALUOp_ID	1	00031	0009182		0083502	/	0541000	3 VC	00034827		00041827		00092027		00a45022		05410003		00044827	_	0005
// /pipeline_vlg_tst/ALUOp_ID // /pipeline_vlg_tst/ALU_OpA	00000000	00000000						100000003		fffffffe		00000000						00000002	==	ffffffd	
// /pipeline_vig_tst/ALU_OpB		0000000		00000004		ffffffb		100000003		00000000	_	00000000		0000003		ffffffb	_	000000005		00000000	
⊢/pipeline_vig_tst/ALO_opb ⊢ /pipeline vig tst/ALU out		ffffffb	,	fffffffc		0000000		Iffffffe		fffffffe		ffffffb		ffffffd		0000000		fffffffd		fffffffd	
	00	00		пппс		,0000000:		ox		xX		00	, i	ШПП		UUUUUUU:		OX		хХ	
// /pipeline_vlg_tst/ShiftAmount				0000000		eccecco.								2000000		rrrrra					
// /pipeline_vlg_tst/Shifter_out		0000000	•	00000004	,	ffffffb		X00000X	•	00000000	,	00000005	,	0000003	;	ffffffb		X00000XX		00000000	
// /pipeline_vlg_tst/Data_out	00000058	0000000				*********		*******		202222		0000004		2222224		2222224		222225		222222	
// /pipeline_vlg_tst/PC_in		00000030		00000034		00000038		00000030		0000004		00000044		0000048		0000004		00000050		00000054	
/pipeline_vlg_tst/PC_out	00000054	00000	00000030)	00000034	•	0000003	8	0000003c		00000040		00000044		00000048		0000004c		00000050	_	000
/pipeline_vlg_tst/Jump_ID	0																				
/pipeline_vlg_tst/Rs_EX_Forward	0	0								1		0								1	
/pipeline_vlg_tst/Rs_ID	00	00						04		0a		00						05		0a	
/pipeline_vlg_tst/Rt_EX_Forward	0	0				2		11		0						2		1		0	
	04	02		03		09		03		01		03)4		09		04		01	
/pipeline_vlg_tst/Rd_EX		00		09		02		03		0a		00		09	_	03		04		0a	
/pipeline_vlg_tst/Rd_ID	09	09		02		03		0a		00		09)3	_	04		0a		00	
/pipeline_vlg_tst/Rd_Mem	0a	0a		00		09		02		03		0a		00		09		03		04	
	f	f		0		f)		f					
-🥎 /pipeline_vlg_tst/Rd_in		ffffffff				ffffffb		fffffffc		0000000	j	fffffffe				ffffffb		fffffffd		00000005	5
/pipeline_vlg_tst/WBData_EX	fffffffd	ffffffff		ffffffb		fffffffc		0000000	5	fffffffe);	fffffb		fffffffd		00000005		fffffffd	
/pipeline_vlg_tst/MemData_EX	000000XX	000000XX	(000000XX		000000XX		xxxxxxxx		000000XX		000000X0		00000XX		000000X				000000XX	ķ
/pipeline_vlg_tst/MemWriteEn	0	0																			
/pipeline_vlg_tst/flash_EX	0																				
/pipeline_vlg_tst/flash_ID	0																				
/pipeline_vlg_tst/Immediate32_ID	00004827	0000482	7	00001027		0000182	,	XOXXXOX		0000000	3	00004827	, J	00001827		0000202	,	XOXXXOXX		00000003	3
// /pipeline_vlg_tst/Immediate32_IF	00002027	0000102	7	00001827		XOXXXOXX		00000003	3	0000482	,	00001827	į	0002027		XOXXXOX		00000003		00004827	,
<pre>// /pipeline_vlg_tst/RegShift</pre>		00XX00X		00000000		xxxxxxxx		xXxXxXx)		XXXXXXXX		00X000X0		OOXXXXXX		XXXXXXXX				xxxxxxxx	
Now	60000000 ps					111				111						1 1 1		==			=

å+	Msgs																				
<pre>/pipeline_vlg_tst/Clk</pre>	1																				
	00091027	0005	00092827	7	0501000		0800000	2	201e000f		21080001		00415022		0541000	3	0001482	,	00020827		0009
-/	1	1										0		e		f		1			
-/-/pipeline_vlg_tst/ALU_OpA		0000000	0					fffffffd		00000000)			fffffffd		fffffffc		ffffffb		00000000	o
-/-/pipeline_vlg_tst/ALU_OpB	fffffffc	0000000	5	00000002	2	ffffffb		00000000)	00000000)			0000000		0000000		00000000		0000000	
	00000004	ffffffb		fffffffe		0000000	5	fffffffd		00000000)			fffffffe		ffffffb		ffffffb		fffffff	
🙌 /pipeline_vlg_tst/ShiftAmount	00	00						хX		00				хX		хX		хX		00	
⊢ ∜ /pipeline_vlg_tst/Shifter_out	fffffffc	0000000	5	00000002	2	ffffffb		00000000)	00000000)					XXXXXXXX	X.	00000000		0000000	
-🔷 /pipeline_vlg_tst/Data_out	xxxxxxxx																				
-👉 /pipeline_vlg_tst/PC_in	00000020	0000005	8	00000050		00000060	0	0000006	•	00000008	3	00000000		00000010		0000001	4	00000018		0000001	ď
	0000001c	0000	00000058	3	00000050		0000006	0	00000064		00000008		00000000		00000010)	0000001	,	00000018	3	000
/pipeline_vlg_tst/Jump_ID	0																				
/pipeline_vlg_tst/Rs_EX_Forward	0	0								0								1		0	
-👉 /pipeline_vlg_tst/Rs_ID	00	00						08		00				08		02		0a		00	
/pipeline_vlg_tst/Rt_EX_Forward	0	0				2		0		0											
-🔷 /pipeline_vlg_tst/Rt_ID	02	04		05		09		01		00				08		01					
-🔷 /pipeline_vlg_tst/Rd_EX	09	00		09		04		05		00						08		0a		00	
-👉 /pipeline_vlg_tst/Rd_ID	01	09		04		05		00								0a		00		09	
-👉 /pipeline_vlg_tst/Rd_Mem	00	0a		00		09		04		05		00						08		0a	
-👉 /pipeline_vlg_tst/Rd_Write_Byt	0	f		0		f						0						f			
-🔷 /pipeline_vlg_tst/Rd_in	ffffffb	fffffffd				ffffffb		fffffffe		0000000	,	ffffffd		00000000)			fffffffe		ffffffb	
-👉 /pipeline_vlg_tst/WBData_EX	ffffffff	fffffffd		ffffffb		fffffffe		0000000	,	fffffffd		00000000				fffffffe		ffffffb			
/pipeline_vlg_tst/MemData_EX	000000XX	000000X	K	000000XX	(000000X	×	xxxxxxxx	(000000XX		000000XX		00000000		xxxxxxxxx	×.	000000XX		000000XX	X
-👉 /pipeline_vlg_tst/MemWriteEn	0	0																			
<pre>/pipeline_vlg_tst/flash_EX</pre>	0																				
<pre>/pipeline_vlg_tst/flash_ID</pre>	0																				
/pipeline_vlg_tst/Immediate32_ID	00000827	0000482	7	00002027	7	0000282	7	0000000		0000000	2	00000000)	0000000		xoxxxox	×	00000003		0000482	7
/pipeline_vlg_tst/Immediate32_IF	00001027	0000202	7	00002827	7	0000000	1	0000000	2	0000000f		00000001		XOXXXOXX		0000000	3	00004827	,	0000082	7
// /pipeline_vlg_tst/RegShift	xxxxxxxx	00XX00X	x	xxxxxxx		xxxxxxx	x	xXxXxXx	(00XX00X		000000XX		00000000		xXxXxXx	×	XXXXXXXX		00XX00X	X
	80000000 ps	1.1.1	1111	1111	111		17.7	1111									111	$\overline{}$	1 1 1	111	声

*	Msgs																				_
<pre>/pipeline_vlg_tst/Clk</pre>	1																				
V	00092027	0009	0062502	2	05410003	3	0002482	7	00031027		00091827		00835022		05410003	3	0003482	7	0004182	7	000
/pipeline_vlg_tst/ALUOp_ID	1	1				f		1								f		1			
V		0000000	0			fffffffd		fffffffc	_	00000000						fffffffe		ffffffd		00000000	
· · · · · · · · · · · · · · · · · · ·		fffffffc		fffffff		0000000		00000004		00000001		fffffffd		fffffff		0000000		00000000		00000001	1
<pre>/pipeline_vlg_tst/ALU_out</pre>	00000002	0000000	4	00000001		fffffffc		fffffff8		fffffff		00000003		00000001		fffffffd		fffffffd		fffffff	
V		00				хX		хX		00						xΧ		хX		00	
/pipeline_vlg_tst/Shifter_out	ffffffe	fffffffc		ffffffff		XXXXXXXX	(XXXXXXXX		00000001		ffffffd		ffffffff		XXXXXXXX	(00000000		00000001	
/pipeline_vlg_tst/Data_out	XXXXXXX																				
<pre>/pipeline_vlg_tst/PC_in</pre>	00000048	0000002	0	00000024		00000028	3	00000020		00000030		00000034		00000038		0000003		00000040		00000044	4
<pre>/pipeline_vlg_tst/PC_out</pre>	00000044	0000	00000020)	00000024		00000028	3	00000020		00000030		00000034		00000038	3	00000030		00000040)	00
<pre>/pipeline_vlg_tst/Jump_ID (</pre>	0																				
/pipeline_vlg_tst/Rs_EX_Forward (0	0						1		0								1		0	
/pipeline_vlg_tst/Rs_ID (00	00				03		0a		00						04		0a		00	
/pipeline_vlg_tst/Rt_EX_Forward (0	0		2		1		0						2		1		0			
/pipeline_vlg_tst/Rt_ID	04	02		09		02		01		02		03		09		03		01		03	
/pipeline_vlg_tst/Rd_EX	09	09		01		02		0a		00		09		02		03		0a		00	
/pipeline_vlg_tst/Rd_ID	03	01		02		0a		00		09		02		03		0a		00		09	
/pipeline_vlg_tst/Rd_Mem (00	00		09		01		02		0a		00		09		02		03		0a	
/pipeline_vlg_tst/Rd_Write_Byt (0	0		f								0		f							
<pre>/pipeline_vlg_tst/Rd_in</pre>	ffffffd	ffffffb		ffffffff		00000004	,	00000001		ffffffc		fffffff8		fffffff		0000000	3	00000001		fffffffd	
/pipeline_vlg_tst/WBData_EX	rrrrrrrr	fffffff		00000004		0000000		ffffffc		fffffff8		(fffffff)		00000003		0000000		ffffffd			
/pipeline_vlg_tst/MemData_EX	000000XX	000000X	x	xxxxxxxx				000000XX		000000XX		000000XX		xxxxxxxX				000000XX		000000X0	ò
/pipeline_vlg_tst/MemWriteEn (0	0																			
<pre>/pipeline_vlg_tst/flash_EX</pre>	0																				
<pre>/pipeline_vlg_tst/flash_ID (</pre>	0																				
/pipeline_vlg_tst/Immediate32_ID (00001827	0000082	7	00001027		XOXXXOXX		00000003	3	00004827		00001027		00001827		XOXXXOX		00000003		00004827	7
7		0000102		XOXXXOXX		0000000		00004827		00001027	_	00001827		XOXXXOXX		0000000		00004827		00001827	
7		XXXXXXXX		xXxXxXxX				xxxxxxxx		00XX00XX		XXXXXXXXXX		xXxXxXxX				XXXXXXXXXX		00X000X0	

\$ 1+	Msgs																		
/pipeline_vlg_tst/Clk	1																		
+ /pipeline_vlg_tst/IR_IF	05410003	0009 00a450	22 105	410003	0004482	7 /	00052027		0501000		08000002		201e000f		21080001		00415022		0541000
/pipeline_vlg_tst/ALUOp_ID	†	1)f		.1				0		1				0		e	
/pipeline_vlg_tst/ALU_OpA		00000000		00000		00000004		0000000				fffffffe		00000000				ffffffe	
// /pipeline_vlg_tst/ALU_OpB		fffffffe	rrrrrrr	00000		00000000		0000001		00000000								00000001	
/pipeline_vlg_tst/ALU_out		00000002	00000001	00000	J04	00000004		ffffff		00000000		fffffffe		00000000				fffffff	
/pipeline_vlg_tst/ShiftAmount		00	Veccente	0X	220	0X	0					xΧ		00			==	xΧ	
/pipeline_vlg_tst/Shifter_out		fffffffe	rtttttf	XX000	JXX	00000000	100	0000001		00000000									
/pipeline_vlg_tst/Data_out	00000014		222222	******															
/pipeline_vlg_tst/PC_in		00000048	00000044	00000		00000054		000005c		00000060		0000006		00000008		0000000c		00000010	
/pipeline_vlg_tst/PC_out	00000010	0000 000000	18 ,00	00004c	0000005	4	00000054		00000050		00000060		00000064		30000000		0000000c		000000
/ /pipeline_vlg_tst/Jump_ID	0						10												
/pipeline_vlg_tst/Rs_EX_Forward	02	0		205		1	100	0				08		00			 ,	00	
/pipeline_vlg_tst/Rs_ID	02	00	12	05		0a	,01	U				08		00				08	
pipeline_vlg_tst/Rt_EX_Forward	01	0	109	04		101	10-			00		0.4		00				00	
+- /pipeline_vlg_tst/Rt_ID /pipeline_vlg_tst/Rd_EX		04	03	04			10.			00		01		00				08	
+- /pipeline_vig_tst/Rd_EX		03	03	.04 .0a		0a 100	10			00									
/pipeline_vlg_tst/Rd_Mem		00	109	.0a		00	10:			00									
+- /pipeline_vig_tst/Rd_Write_Byt	0.0	00	109	103		JU4	,00	d		00									
+- /pipeline_vig_tst/Rd_in	00000000	ffffffd	recent	200000	202	00000001		0000004		U		0000000				ffffffe		00000000	
+- /pipeline_vig_tst/WBData_EX		fffffff	100000002	200000		100000001		JUUUUU-		00000000		0000000		ffffffe		00000000		00000000	
pipeline_vig_tst/WebData_EX pipeline_vig_tst/MemData_EX		000000XX	xxxxxxxX	100000	JU I	00000004		00000XX		00000000				000000XX		0000000XX		00000000	
+- /pipeline_vig_tst/MemWriteEn	0	00000000	XXXXXXXX			OOOOOOXX	U	σουσαλ		JUUUUUUU				OOUUUXX		σσσσσσαχ		0000000	
<pre>/pipeline_vig_tst/flash_EX</pre>	0	•																	
/pipeline_vig_tst/flash_EX /pipeline_vig_tst/flash_ID	0																		
pipeline_vig_tst/limmediate32_ID pipeline_vlg_tst/Immediate32_ID	XOXXXOXX	00001827	00002027	X0XXX	nyy	00000003	0	0004827		00000000		0000000		00000002		00000000		00000001	
+- /pipeline_vig_tst/Immediate32_IF		00001827	XOXXXOXX	00000		00000003		0004827		00000001		0000000		00000002		00000000		XOXXXOX)	
+- /pipeline_vig_tst/RegShift		XXXXXXXXX	xXxXxXxX	00000	303	XXXXXXXXXX		0XX00XX		00000000		0000000		OOXXOOXX		00000001		00000000	
- ,		******	XXXXXXX			NANAXA,	0	JANUUAA		00000000				OUXXUUXX		0000000		0000000	
≜. provide the second of the	10000000 ps			1700	000000 ps				180000	0000 ps				1900000	0000 ps				20000

/pipeline_vlg_tst/Clk	Msgs 1																				
/ /pipeline_vlg_tst/IR_IF	00034827	0541	0001482	7	0002082	7	00091027	7	00625022		05410003	3	00024827		00031027		00835022		05410003		0003
/pipeline_vlg_tst/ALUOp_ID	1	f		11								f		1				0		f	
/pipeline_vlg_tst/ALU_OpA	rrrrrrrr	0000000	3	ffffffff		00000000)					00000002		fffffffe		00000000				00000001	
/pipeline_vlg_tst/ALU_OpB	00000000	0000000	4	00000000		00000004		00000003		ffffffc		00000004		ffffffd		00000004		00000000		00000002	_
/pipeline_vlg_tst/ALU_out	ffffffff	ffffffff		ffffffff		fffffffc		fffffffd		00000004		fffffffe	X	00000001		fffffffc		00000000		fffffff	
/pipeline_vlg_tst/ShiftAmount	xx	0X				00						0X		xΧ		00				0X	
/pipeline_vlg_tst/Shifter_out	00000000	X00000X	х	00000000		00000004		00000003		ffffffc		000000XX				00000004	,	00000000		0000000X	
/pipeline_vlg_tst/Data_out	xxxxxxxx																				
/pipeline_vlg_tst/PC_in	00000040	0000001	4	00000018	3	00000010		00000020		00000024		00000028)	00000020		00000034	,	00000038		0000003c	
/pipeline_vlg_tst/PC_out	0000003c	0000	0000001	4	0000001	3	00000010		00000020		00000024		00000028		00000020		00000034		00000038		000
/pipeline_vlg_tst/Jump_ID	0																				
/pipeline_vlg_tst/Rs_EX_Forward	1	0		1		0								1		0					
<pre>/pipeline_vlg_tst/Rs_ID</pre>	0a	02		0a		00						03)	0a		00				04	
/pipeline_vlg_tst/Rt_EX_Forward	0	0								2		1	,	0							
<pre>/pipeline_vlg_tst/Rt_ID</pre>	01	01						02		09		02	X	01		02		00		03	
/pipeline_vlg_tst/Rd_EX	0a	08		0a		00		09		01		02	X	0a		00					
/pipeline_vlg_tst/Rd_ID		0a		00		09		01		02		0a		00		09		00		0a	
<pre>/pipeline_vlg_tst/Rd_Mem</pre>	00	00		08		0a		00		09		01	X	02		0a		00			
/>/pipeline_vlg_tst/Rd_Write_Byt	0	0		f				0		f								0			
<pre>/pipeline_vlg_tst/Rd_in</pre>	00000000	0000000	0	rrrrrrr						ffffffc		fffffffd		00000004		fffffffe		00000001		00000000	
<pre>/pipeline_vlg_tst/WBData_EX</pre>	mmm	ffffffff						fffffffc		ffffffd		00000004)	ffffffe		0000000		00000000			
<pre>/pipeline_vlg_tst/MemData_EX</pre>	000000XX	XXXXXXX	X	000000XX		000000XX		000000XX		000000XX		xxxxxxxx)		000000XX		xxxxxxx)		00000000			
/pipeline_vlg_tst/MemWriteEn	0	0																			
/pipeline_vlg_tst/flash_EX	0																				
/pipeline_vlg_tst/flash_ID	0																				
/pipeline_vlg_tst/Immediate32_ID	00000003	X0XXX0X		00000003		00004827		00000827		00001027		XOXXXOXX		00000003		00004827		00000000		XOXXXOXX	
/pipeline_vlg_tst/Immediate32_IF	00004827	0000000	3	00004827		00000827		00001027		XOXXXOXX		00000003	;	00004827		0000102		XOXXXOXX		00000003	3
/pipeline_vlg_tst/RegShift	XXXXXXXXX	xXxXxXx	X	XXXXXXXX		00XX00XX		XXXXXXXX		XXXXXXXX		xXxXxXx)		XXXXXXXX		xxxXxxx)		00000000			

≨ 1•	Msgs																	
/pipeline_vlg_tst/Clk	1																	
-/	21080001	0003	141827	00092027	00a45022	0541000	3	000 44 827		00052027	,	0501000		0800000	2	201e000f		2108000
-/	0	1					f		1				0		1			
		ffffffff	00000000				0000000		0000000		0000000)			ffffffff		0000000)
+		0000000	00000002		ffffff	e	0000000		0000000		0000000	2	00000000					
- → /pipeline_vlg_tst/ALU_out	00000000	fffffff	fffffffe	(fffffff	00000	002	0000000	;	0000000	3	fffffffe		00000000)	ffffffff		0000000)
± - /pipeline_vlg_tst/ShiftAmount	00		00				OX		0X		00						00	
+-/	00000000	0000000	00000002	0000000	ffffff	e	X00000X		0000000)	0000000	2	00000000)				
	xxxxxxx																	
+		00000040	00000044	0000004	3 (00000	04c	00000050		0000005	•	0000005		00000060)	0000006		0000000	3
	80000000	0000 0000	000040	00000044	00000048	0000004	c	00000050		00000054		00000050		00000060)	00000064	1	000000
<pre>/pipeline_vlg_tst/Jump_ID</pre>	0																	
/pipeline_vlg_tst/Rs_EX_Forward	0	1	Ю						1		0							
≖ /pipeline_vlg_tst/Rs_ID	00	0a	00				05		0a		00				08		00	
+ /pipeline_vlg_tst/Rt_EX_Forward	0	0			2		1		0									
-	00	01	03	04	09		04		01		04		00		01		00	
+	00	0a	00	09	03		04		0a		00							
- /pipeline_vlg_tst/Rd_ID	00	00	09	03	04		0a		00		09		00					
■ - / /pipeline_vlg_tst/Rd_Mem	00	00	0a	00	09		03		04		0a		00					
≖ - <pre>/pipeline_vlg_tst/Rd_Write_Byt</pre>	0	0	f	Ю	f								0					
≖ /pipeline_vlg_tst/Rd_in		0000000	recent		ffffff		fffffff		0000000		0000000				0000000)		
- - / pipeline_vlg_tst/WBData_EX		fffffff		fffffffe	,ffffff		0000000	2	0000000	3			00000000)			ffffffff	
- → /pipeline_vlg_tst/MemData_EX	000000XX	000000XX	000000X0	000000X	00000	0XX	xxxxxxxx		000000X		000000X	(00000000)			000000X	K.
	0	0																
<pre>/pipeline_vlg_tst/flash_EX</pre>	0																	
<pre>/pipeline_vlg_tst/flash_ID</pre>	0																	
■ - / /pipeline_vlg_tst/Immediate32_ID		00000003	00004827			027	XOXXXOX		0000000		0000482		00000000		0000000		0000000	
∓ - / /pipeline_vlg_tst/Immediate32_IF		00004827	00001827	0000202	7 XOXXX	0XX	0000000	;	0000482	7	0000202		0000000		0000000	2	00000001	
■ - / /pipeline_vlg_tst/RegShift	000000XX	XXXXXXXX	00X000X0	XXXXXXXX	XXXXX	XXX	xXxXxXx		XXXXXXXX	(00XX00X	(00000000)			00XX00X	4
[△] 👺 🏵 Now	00000000 ps	11111		250000	0000 ps		1 1 1 1	2600000					270000					28000
Gursor 1)1861083 ps																	2801

ù +	Msgs																				
<pre>/pipeline_vlg_tst/Clk</pre>	1																				
+	00835022	2108	0041502	2	05410003		0001482	7	00020827		00625022		05410003		0002482		0003102		00091827		0083
+	1	0		e		f		1				0		f		1					
+	00000000	0000000		ffffffff		0000000	4	0000000		00000000				rrrrrrr		ffffffb		00000000			
+	fffffffc	0000000		00000001		fffffffd		0000000		fffffffd		00000000		00000004		00000000		00000004		ffffffff	
	00000004	0000000	0			0000000	7	0000000	7	0000000	;	00000000		ffffffb		ffffffb		fffffffc		0000000	i e
₽-	00	00				0X		0X		00						xΧ		00			
-/	fffffffc	0000000	0			xfffffXx		0000000	0	fffffffd		00000000				00000000		00000004		ffffffff	
/> /pipeline_vlg_tst/Data_out	xxxxxxx																				
	00000038	0000000	C	00000010		0000001	4	0000001	3	00000020		00000024		00000028	3	0000002		00000030		0000003	4
	00000034	0000	0000000		00000010)	0000001	4	00000018		00000020		00000024		00000028	3	00000020		00000030		000
<pre>/pipeline_vlg_tst/Jump_ID</pre>	0																				
-/	0	0						1		0						1		0			
├ <mark>-</mark>	00	00		08		02		0a		00				03		0a		00			
	2	0																			
	09	00		08		01						00		02		01		02		03	
├-{> /pipeline_vlg_tst/Rd_EX	02	00				08		0a		00						0a		00		09	
├�️/pipeline_vlg_tst/Rd_ID	03	00				0a		00		09		00		0a		00		09		02	
	09	00						08		0a		00						0a		00	
🛶 /pipeline_vlg_tst/Rd_Write_Byt	f	0						f				0						f		0	
🛶 /pipeline_vlg_tst/Rd_in	fffffffc	ffffffff		00000000						0000000	7			00000000)			ffffffb			
├� /pipeline_vlg_tst/WBData_EX	00000001	0000000	0					0000000	,			00000000				ffffffb				ffffffc	
├- ∜ /pipeline_vlg_tst/MemData_EX	xxxxxxxx	0000000	X	00000000				xxxxxxx	X .	000000XX		00000000				000000X				000000X	X
/pipeline_vlg_tst/MemWriteEn	0	0																			
<pre>/pipeline_vlg_tst/flash_EX</pre>	0																				
<pre>/pipeline_vlg_tst/flash_ID</pre>	0																				
-/-/ /pipeline_vlg_tst/Immediate32_ID	00001827	0000000	0	00000001		XOXXXOX	X	0000000	3	0000482	,	00000000		X0XXX0XX		0000000	;	00004827	,	0000102	,
	XOXXXOXX	0000000		XOXXXOXX		0000000	3	0000482		0000082		XOXXXOX		0000000		0000482		00001027		0000182	
	xxxxxxxx	000000>	x	00000000				xXxXxXx	X	00XX00X		00000000				xxxxxxx		00XX00X		XXXXXXX	X
	80000000 ps	1 1 1			1.1.1			1 1 1 1	intro	1 1 1			1.1.1		1.1.1				1111	1 1 1	
✓ Cursor 1)1245447 ps					290000	00000 ps				300000	JUUU ps				310000	0000 ps				32

<pre>/pipeline_vlg_tst/Clk</pre>	1																		
/pipeline_vlg_tst/IR_IF	08000002	0083 0541000)3 (0	0034827	00041	827	00092027		00a4502	2	05410003		0004482		0005202	,	05010001		08000
► / pipeline_vlg_tst/ALUOp_ID	1	1	f	1							f			1				0	
├ <mark>-</mark> /pipeline_vlg_tst/ALU_OpA	00000000	0000000	00000002	fffff	fe	0000000)				00	0000005		0000000		00000000			
⊢🍫 /pipeline_vlg_tst/ALU_OpB	00000000	ffffffc	00000004	0000	0000	0000000	,	0000000		ffffffc	00	0000004		00000000)	00000004		00000000	
-👉 /pipeline_vlg_tst/ALU_out	00000000	00000004	fffffffe	fffff	fe	fffffffc		fffffffe		00000004	1 00	0000001		0000000		ffffffc		00000000	
-👉 /pipeline_vlg_tst/ShiftAmount	00	00	OX	хX		00					0)	X .		0X		00			
/pipeline_vlg_tst/Shifter_out	00000000	ffffffc	000000XX	0000	0000	0000000	4	0000000	1	ffffffc	X	00000XX		00000000)	00000004		00000000	
-🔷 /pipeline_vlg_tst/Data_out	xxxxxxxx																		
-🧇 /pipeline_vlg_tst/PC_in	00000064	00000038	0000003c	0000	0040	0000004	4	0000004		0000004	00	0000050		00000054	1	0000005c		00000060	
/pipeline_vlg_tst/PC_out	00000060	0000 0000003	38 (0	000003c	00000	040	00000044		00000048		0000004c		00000050		0000005)	0000005		000
<pre>/pipeline_vlg_tst/Jump_ID</pre>	0																		
/pipeline_vlg_tst/Rs_EX_Forward	0	0		1		0								1		0			
<pre>/pipeline_vlg_tst/Rs_ID</pre>	08	00	04	0a		00					05	5		0a		00			
/pipeline_vlg_tst/Rt_EX_Forward	0	2	1	0						2	1			0					
/pipeline_vlg_tst/Rt_ID	01	09	03	01		03		04		09	04	4		01		04		00	
-🧇 /pipeline_vlg_tst/Rd_EX	00	02	03	0a		00		09		03	04	4		0a		00			
	00	03	0a	00		09		03		04	Oa			00		09		00	
/pipeline_vlg_tst/Rd_Mem	00	09	02	03		0a		00		09	03	3		04		0a		00	
/pipeline_vlg_tst/Rd_Write_Byt	0	f						0		f								0	
/pipeline_vlg_tst/Rd_in	00000000	ffffffc	00000001		0004	fffffffe				fffffffc		fffffe		0000000		00000001			
- /pipeline_vlg_tst/WBData_EX		00000001	00000004	Xfffff				fffffffc		fffffffe		0000004		0000000				00000000	
/pipeline_vlg_tst/MemData_EX	00000000		xxxxxxxxX	0000	00XX	000000X)	000000X		000000XX	(XX	xxxxxxx		000000XX				00000000	
/pipeline_vlg_tst/MemWriteEn	0	0	++																
/pipeline_vlg_tst/flash_EX	0		+																
/pipeline_vlg_tst/flash_ID	0																		
/pipeline_vlg_tst/Immediate32_ID		00001827	XOXXXOXX		0003	0000482		0000182		0000202		OXXXOXX		0000000		00004827		00000000	
/pipeline_vlg_tst/Immediate32_IF	00000002	X0XXX0XX	00000003		4827	0000182		0000202		X0XXX0X)	_	0000003		0000482		00002027		00000001	
/pipeline_vlg_tst/RegShift	00000000		xXxXxXxX	XXXX	xxxx	00X000X	,	XXXXXXXX		XXXXXXXXX	(X)	XXXXXX		XXXXXXXXX		00XX00XX		00000000	
! ● Now	0000000 ps			33	00000000	I I I I I			340000	0000 ps				350000	0000 ps				36

/pipeline_vlg_tst/Clk	1										
	xxxxxxxxx	20470000	20670000		20870000		20a70000				
+-/-/pipeline_vlg_tst/ALUOp_ID	0	e									
+-/-/pipeline_vlg_tst/ALU_OpA	xxxxxxxxx	00000001		00000002		00000003		00000004		00000005	i e
-/	00000000	00000000									
+	xxxxxxxxx	00000001		00000002		00000003		00000004		00000005	i
-/	xx	OX		0X		0X		OX		OX	
+-/-/pipeline_vlg_tst/Shifter_out	00000000	0000000X		000000XX		X00000Xx		X00000XX		X00000XX	
+-/-/pipeline_vlg_tst/Data_out	xxxxxxxx										
+	00000088	00000070		00000074		00000078		0000007c		00000080	
+	00000084	0000006c	00000070		00000074		00000078		0000007c		0000008
/pipeline_vlg_tst/Jump_ID	0										
	0	0									
+-/-/pipeline_vlg_tst/Rs_ID	xx	01		02		03		04		05	
+-/- /pipeline_vlg_tst/Rt_EX_Forward	0	0									
≖ - / /pipeline_vlg_tst/Rt_ID	xx	07									
≖ - ∜ /pipeline_vlg_tst/Rd_EX	xx	1e		07							
-/	xx	00									
	07	00		1e		07					
≖ - / /pipeline_vlg_tst/Rd_Write_Byte	f	0		f							
+	00000005	00000000		0000000f		00000001		00000002		00000003	3
+	00000000	0000000f		00000001		00000002		00000003		00000004	
+	00000084			000000XX				000000XX		000000XX	
→ /pipeline_vlg_tst/MemWriteEn	0	0									
<pre>/pipeline_vlg_tst/flash_EX</pre>	0										
/pipeline_vlg_tst/flash_ID	0										
+- /pipeline_vlg_tst/Immediate32_ID	xxxxxxxx	00000000									
	xxxxxxxx	00000000								1	
+	84848484			XXXXXXXXX				xxxxxxxx		xxxxxxxx	
Now	7,342,400 ps			1 1 1		1 1 1 1		1 1 1		1 1 1	

conclusion:

最后一张图看出,按顺序输出 r1~r5 的值为 1,2,3,4,5,所以,排序起效;