关于测试单周期 CPU 的简单方法

1. 测试程序段（汇编代码）

address	instruction	op	rs	rt	immediate	code	result
0x00000000	addiu $1,$0,8	001001	00000	00001	0000 0000 0000 1000	24010008	$1 = 8
0x00000004	ori $2,$0,2	001101	00000	00010	0000 0000 0000 0010	34020002	$2 = 2

address	instruction	op	rs	rt	rd	shamt	funct	code	result
0x00000008	add $3,$2,$1	000000	00010	00001	00011	00000	100000	00411820	$3 = 10
0x0000000C	sub $5,$3,$2	000000	00011	00010	00101	00000	100010	00622822	$5 = 8
0x00000010	and $4,$5,$2	000000	00101	00010	00100	00000	100100	00a22024	$4 = 0
0x00000014	or $8,$4,$2	000000	00100	00010	01000	00000	100101	00824025	$8 = 2
0x00000018	sll $8,$8,1	000000	00000	01000	01000	00001	000000	00084040	$8 = 4 $8 = 8

address	instruction	op	rs	rt	immediate	code	result
0x0000001C	bne $8,$1,-2 (≠,转 18)	000101	00001	01000	1111 1111 1111 1110	1501fffe	
0x00000020	slti $6,$2,4	001010	00010	00110	0000 0000 0000 0100	28460004	$6 = 1
0x00000024	sltiu $7,$6,0	001011	00110	00111	0000 0000 0000 0000	2cc70000	$7 = 0
0x00000028	addiu $7,$7,8	001000	00111	00111	0000 0000 0000 1000	24e70008	$7 = 8 $7 = 16
0x0000002C	beq $7,$1,-2 (=,转 28)	000100	00111	00001	1111 1111 1111 1110	10e1fffe	
0x00000030	sw $2,4($1)	101011	00001	00010	0000 0000 0000 0100	ac220004	mem[12:15]<=2
0x00000034	lw $9,4($1)	100011	00001	01001	0000 0000 0000 0100	8c290004	$9 = 2
0x00000038	addiu $10,$0,-2	001001	00000	01010	1111 1111 1111 1110	240afffe	$10 = -2
0x0000003C	addiu $10,$10,1	001001	01010	01010	0000 0000 0000 0001	254a0001	$10 = -1
0x00000040	andi $11,$2,2	001100	00010	01011	0000 0000 0000 0010	304b0002	$11 = 2

address	instruction	op	rs	rt	rd	shamt	funct	code	result
0x00000044	addu $10,$0,$2	000000	00000	00010	01010	00000	100001	00025021	$10 = 2
0x00000048	subu $10,$10,$2	000000	01010	00010	01010	00000	100011	01425023	$10 = 0
0x0000004C	xor $10,$8,$2	000000	01000	00010	01010	00000	100110	01025026	$10 =10
0x00000050	nor $10,$8,$2	000000	01000	00010	01010	00000	100111	01025027	$10 = fffffff5
0x00000054	sltu $10,$8,$2	000000	01000	00010	01010	00000	101011	0102502b	$10 = 0
0x00000058	srl $10,$8,1	000000	00000	01000	01010	00001	000010	00085042	$10 = 4
0x0000005C	sllv $10,$8,$6	000000	01000	00110	01010	00000	000100	00c85004	$10 =16
0x00000060	srlv $10,$8,$6	000000	01000	00110	01010	00000	000110	00c85006	$10 = 4

address	instruction	op	rs	rt	immediate	code	result
0x00000064	addi $10,$0,-1	001000	00000	01010	1111 1111 1111 1111	200affff	$10 = -1

address	instruction	op	rs	rt	rd	shamt	funct	code	result
0x00000068	sra $11,$10,1	000000	00000	01010	01011	00001	000011	000a5843	$11 = -1
0x0000006C	srav $11,$10,$6	000000	01010	00110	01011	00000	000111	00ca5807	$11 = -1

address	instruction	op	rs	rt	immediate	code	result
0x00000070	xori $10,$3,7	001110	00011	01010	0000 0000 0000 0111	386a0007	$10 = 13
0x00000074	lui $5, 10	001111	00000	00101	0000 0000 0000 1010	3c05000a	$5=0xA0000
0x00000078	sh $3, 8($1)	101001	00001	00011	0000 0000 0000 1000	a4230008	mem[16:17]=10
0x0000007C	sb $3, 10($1)	101000	00001	00011	0000 0000 0000 1010	a023000a	mem[18]=10
0x00000080	lh $10, 9($1)	100001	00001	01010	0000 0000 0000 1001	842a0009	$10=0x10101010
0x00000084	lb $10, 4($1)	100000	00001	01010	0000 0000 0000 0100	802a0004	$10 = 0

address	instruction	op	address	code	result
0x00000088	jal 0000008C	000011	0000008c	0c000023	

address	instruction	op	rs	rt	immediate	code	result
0x0000008C	ori $1,$0,0x00000098	001101	00000	00001	0000 0000 0110 0010	34010098	$1=0x00000098

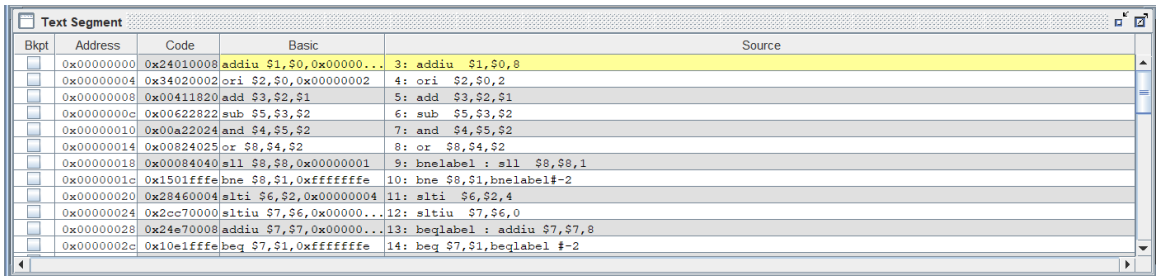
address	instruction	op	rs	rt	rd	shamt	funct	code	result
0x00000090	jr $1	000000	00001	00000	00000	00000	001000	00200008	
0x00000094	add $10,$1,$2	000000	00001	00010	01010	00000	000010	00225020	
0x00000098	slt $10,$5,$6	000000	00101	00110	01010	00000	101010	00a6502a	$10 = 0

address	instruction	op	rs	rt	rd	shamt	funct	code	result
0x0000009C	mult $9,$2	000000	01001	00010	00000	00000	011000	01220018	hi = 0 lo = 4
0x000000A0	mfhi $10	000000	00000	00000	01010	00000	010000	00005010	$10 = 0
0x000000A4	mflo $10	000000	00000	00000	01010	00000	010010	00005012	$10 = 4
0x000000A8	div $9, $2	000000	01001	00010	00000	00000	011010	0122001a	1
0x000000AC	mfhi $10	000000	00000	00000	01010	00000	010000	00005010	$10 = 1
0x000000B0	mflo $10	000000	00000	00000	01010	00000	010010	00005012	$10 = 0

address	instruction	op	address	code	result
0x000000B4	j 0x000000B8	000010	000000b8	0800002e	

address	instruction	op	rs	rt	immediate	code	result
0x000000B8	haul	111111	00000	00000	0000000000000000	fc000000	haul

2. 机器码



Text Segment

Bkpt	Address	Code	Basic	Source
	0x00000000	0x24010008	addiu $1,$0,0x00000...	3: addiu $1,$0,8
	0x00000004	0x34020002	ori $2,$0,0x00000002	4: ori $2,$0,2
	0x00000008	0x00411820	add $3,$2,$1	5: add $3,$2,$1
	0x0000000c	0x00622822	sub $5,$3,$2	6: sub $5,$3,$2
	0x00000010	0x00a22024	and $4,$5,$2	7: and $4,$5,$2
	0x00000014	0x00824025	or $8,$4,$2	8: or $8,$4,$2
	0x00000018	0x00084040	sll $8,$8,0x00000001	9: bnelabel : sll $8,$8,1
	0x0000001c	0x1501fffe	bne $8,$1,0xfffffffe	10: bne $8,$1,bnelabel#-2
	0x00000020	0x28460004	slti $6,$2,0x00000004	11: slti $6,$2,4
	0x00000024	0x2cc70000	sltiu $7,$6,0x00000...	12: sltiu $7,$6,0
	0x00000028	0x24e70008	addiu $7,$7,0x00000...	13: beqlabel : addiu $7,$7,8
	0x0000002c	0x10e1fffe	beq $7,$1,0xfffffffe	14: beq $7,$1,beqlabel #-2

由于寄存器数据宽度为 8 位，则 coe 文件的格式为：

MEMORY_INITIALIZATION_RADIX=16;

MEMORY_INITIALIZATION_VECTOR=

24,01,00,08,
34,02,00,02,
00,41,18,20,
00,62,28,22,
00,a2,20,24,
00,82,40,25,
00,08,40,40,
15,01,ff,fe,
28,46,00,04,
2c,c7,00,00,
24,e7,00,08,
10,e1,ff,fe,
ac,22,00,04,
8c,29,00,04,
24,0a,ff,fe,
25,4a,00,01,
30,4b,00,02,
00,02,50,21,
01,42,50,23,
01,02,50,26,
01,02,50,27,
01,02,50,2b,
00,08,50,42,
00,c8,50,04,
00,c8,50,06,
20,0a,ff,ff,
00,0a,58,43,
00,ca,58,07,
38,6a,00,07,
3c,05,00,0a,
a4,23,00,08,

a0,23,00,0a,
84,2a,00,09,
80,2a,00,04,
0c,00,00,23,
34,01,00,98,
00,20,00,08,
00,22,50,20,
00,a6,50,2a,
01,22,00,18,
00,00,50,10,
00,00,50,12,
01,22,00,1a,
00,00,50,10,
00,00,50,12,
08,00,00,2e,
fc,00,00,00;

3. 仿真模块

仿真代码及注释如下：

```
`timescale 1ns / 1ps

module top_sim;

// Inputs
reg clkin;
reg reset;

wire[31:0] inst, pc, pcAdd4;

// Instantiate the Unit Under Test (UUT)
top uut (
    .clkin(clkin),
    .reset(reset),
    .inst(inst),
    .pc(pc),
    .pcAdd4(pcAdd4)
);

initial
begin
    // Initialize Inputs
    clkin = 0;
    reset = 1;
    // Wait 20 ns for global reset to finish
    #20;
    reset = 0;
end

parameter PERIOD = 20;

always
```

```
begin
    clkin = 1'b0;
    #(PERIOD / 2);
    clkin = 1'b1;
    #(PERIOD / 2);
end
endmodule
```