5 测试及结果分析

5.1 仿真代码及分析

5.1.1 双重冒险测试

```
/* double hazard test */
addi x1, x0, 0x123
addi x2, x0, 0x345
addi x3, x0, 0xAB
addi x4, x0, 0
addi x5, x0, 0
# double hazard
add x1, x2, x2 # x1 = 0x68a  IF ID EXE MEM WB
add x1, x1, x2 # x1 = 0x9cf     IF ID  EXE MEM WB
sub x1, x1, x3 # x1 = 0x924        IF  ID  EXE MEM WB
```

sub x1, x1, x3 指令在 ID 阶段想要使用的是 add x1, x1, x2 在 EXE 阶段计算得到的 x1 的值，而非 MEM 阶段 x1 的值。

5.1.2 load 和 store 指令测试

主要测试 load 和 store 指令是否正常。

```
//load and store
addi x1, x0, 0x123
addi x2, x0, 0x4
addi x3, x0, 0x789
slli x1, x1, 0x14
addi x4, x0, 0x222
slli x3, x3, 0x8
addi x5, x0, 0x88
addi x6, x0, 0x99
addi x10, x0, 0x23
add x1, x1, x5
addi x11, x0, 0x90
addi x12, x0, 0x43
add x1, x1, x3
sb  x1, 0(x2)
lw x1, 0(x2)
```

5.1.3 分支跳转测试

主要测试分支跳转指令是否正常，以及能否在发生数据冒险时正常工作。

```
# 分支跳转测试，带冒险
addi x7, x0, 0x7 #x7 = 0x7
srli x3, x7, 0x1 # x3 = 0x3
addi x1, x0, 0x123 #x1 = 0x123
addi x2, x0,0x4 # x2 = 0x4
sw x1, 0x4(x2) # dmem[2] = 0x123
lw x5, 0x4(x2) # x5 = 0x123
add x6, x5, x7 # x6 = 0x12a
sub x1, x2, x3 # x1 = 0x1
or x7, x6, x7 # x7 = 0x12f
and x9, x7, x6 # x9 = 0x12a
add x7, x1, x7 # x7 = 0x130
sub x9, x9, x1 # x9 = 0x129
bge x7, x9, _L1
add x1, x7, x1 # x1 = 0x25a
add x3, x2, x6
_L1:
    add x1, x9, x1 # x1 = 0x12a
```

5. 2 仿真测试结果

指令执行结果用 verilog 系统函数$display 输出在 ModelSim 终端界面上。

5.2.1 数据冒险处理

数据冒险可以分为四类：

1. ID 阶段与 EXE 阶段数据冒险
2. ID 阶段与 MEM 阶段数据冒险
3. ID 阶段同时与 EXE 阶段和 MEM 阶段数据冒险
4. ID 阶段与 WB 阶段数据冒险

前两者通过前递解决，如果指令 1 和指令 2 发生 ID 与 EXE 数据冒险，指令 2 会在指令 1EXE 阶段将 EXE_MEM 寄存器里的 ALUResult 值前递给指令 1EXE 阶段的前递数据选择器。如果指令 1 和指令 2 发生 ID 与 MEM 数据冒险，指令 2 会在指令 1EXE 阶段将指令 2WB 阶段要写回的数据前递给指令 1EXE 阶段的前递数据选择器。当前两者同时发生时，CPU 会优先处理 ID 与 EXE 的数据冒险。对于第四种数据冒险，在设计流水线寄存器时，所有流水线寄存器写入模式指定为上升沿写入，而 RF 设置为下降沿写入，这样 WB 会在两个时钟周期的间隙向 RF 写入数据，等下一个时钟周期到来时，ID_EXE 寄存器中存取的已经是新写入的值。

对于 Load-Use 相关，不能采用前递方式解决数据冲突，必须在相邻两条指令之间强制插入一个气泡以消除数据冒险，这里采用的方法是让 PC 寄存器和 IF_ID 寄存器暂停写入一个中期，同时清空 ID_EXE 寄存器。

5.2.2 控制冒险处理

对于控制冒险，在发生分支跳转时，EXE 阶段检测到分支跳转，冒险控制单元生成清空信号，清空 IF_ID 和 ID_EXE 寄存器。

5. 3 下载测试代码及分析

下载测试代码即是 SoC 顶层文件。代码如下:

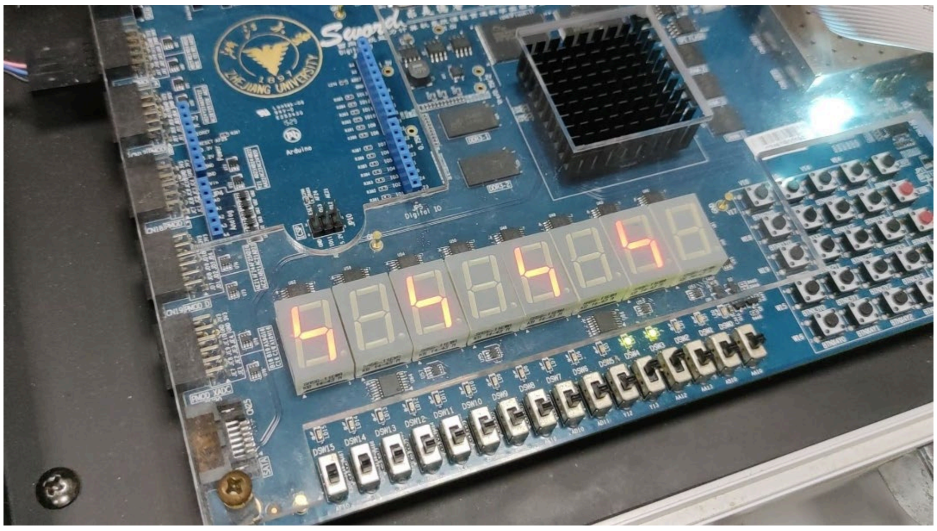
```
`timescale 1ns / 1ps
////////////////////////////////////////////////////////////////////////////
///////
// Company:
// Engineer:
//
// Create Date:    16:01:47 06/21/2022
// Design Name:
// Module Name:    IP2SOC_Top
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
////////////////////////////////////
module IP2SOC_Top(
input RSTN,
input [3:0] BTN_y,
input [15:0] SW,
input clk_100mhz,
output CR,
output seg_clk,
output seg_sout,
output SEG_PEN,
output seg_clrn,
output led_clk,
output led_sout,
output LED_PEN,
output led_clrn,
output RDY,
output readn,
output [4:0] BTN_x
    );

//name regulation: the wire is named by the port which the data output

//wire for U9_out
wire [4:0] Key_out;
wire [3:0] Pulse;
wire [3:0] BTN_OK;
wire [15:0] SW_OK;
wire rst;
SAnti_jitter U9(
.clk(clk_100mhz),
.RSTN(RSTN),
.readn(readn),
.Key_y(BTN_y[3:0]),
.Key_x(BTN_x[4:0]),
.SW(SW),
.Key_out(Key_out),
.Key_ready(RDY),
.pulse_out(Pulse),
.BTN_OK(BTN_OK),
.SW_OK(SW_OK),
.CR(CR),
.rst(rst)
);

//wire for U8_out
wire [31:0] clkdiv;
wire Clk_CPU;
clk_div U8(
.clk(clk_100mhz),
.rst(rst),
.SW2(SW_OK[2]),
.clkdiv(clkdiv),
.Clk_CPU(Clk_CPU)
);
```

```verilog
//wire for U1
wire [31:0] spo;
wire [31:0] PC_out;
wire [31:0] Data_in;
wire mem_w;
wire [3:0] wea;
wire [31:0] Addr_out;
wire [31:0] Data_out;
wire CPU_MIO;
wire INT;
wire counter0_out;
wire [31:0] ram_data_out;
PipelineCPU_top U1(
.Clk_CPU(Clk_CPU),
.rst(rst),
//.clk_100mhz(~clk_100mhz),
.MIO_ready(),
.inst_in(spo),
.Data_in(Data_in),
.mem_w(mem_w),
.PC_out(PC_out),
.Addr_out(Addr_out),
.Data_out(Data_out),
.CPU_MIO(CPU_MIO),
.wea(wea),
.INT(counter0_out)
);

//U3
wire [9:0] ram_addr;
wire [31:0] ram_data_in;
D_mem U3(
.ADDRA(ram_addr),
.DINA(ram_data_in),
.WEA(wea),
.CLKA(~clk_100mhz),
.DOUTA(ram_data_out)
);

//U2
IP1 U2(
.a(PC_out[11:2]),
.spo(spo)
);
```

```
//M4
wire [7:0] blink;
SEnter_2_32 M4(
.clk(clk_100mhz),
.BTN(BTN_OK[2:0]),
.Ctrl({SW_OK[7:5],SW_OK[15],SW_OK[0]}),
.D_ready(RDY),
.Din(Key_out),
.readn(readn),
.Ai(),
.Bi(),
.blink(blink)
);

//U4
wire [15:0] led_out;
wire [31:0] counter_out;
wire counter2_out;
wire counter1_out;
wire counter_we;
wire [31:0] Peripheral_in;
wire GPIOf0000000_we;
wire GPIOe0000000_we;
MIO_BUS U4(
.clk(clk_100mhz),
.rst(rst),
.BTN(BTN_OK),
.SW(SW_OK),
.mem_w(mem_w),
.Cpu_data2bus(Data_out),
.addr_bus(Addr_out),
.ram_data_out(ram_data_out),
.led_out(led_out),
.counter_out(counter_out),
.counter0_out(counter0_out),
.counter1_out(counter1_out),
.counter2_out(counter2_out),

.Cpu_data4bus(Data_in),
.ram_data_in(ram_data_in),
.ram_addr(ram_addr),
.data_ram_we(),
.GPIOf0000000_we(GPIOf0000000_we),
.GPIOe0000000_we(GPIOe0000000_we),
.counter_we(counter_we),
.Peripheral_in(Peripheral_in)
);
//U10
wire [1:0] counter_set;
Counter_x U10(
.clk(~Clk_CPU),
.rst(rst),
.clk0(clkdiv[6]),
.clk1(clkdiv[9]),
.clk2(clkdiv[11]),
.counter_we(counter_we),
.counter_val(Peripheral_in),
.counter_ch(counter_set),
.counter0_OUT(counter0_out),
.counter1_OUT(counter1_out),
.counter2_OUT(counter2_out),
.counter_out(counter_out)
);

//U6
wire [31:0] disp_num;
wire [7:0] point_out;
wire [7:0] LE_out;
SSeg7_Dev U6(
.clk(clk_100mhz),
.rst(rst),
.Start(clkdiv[20]),
.SW0(SW_OK[0]),
.flash(clkdiv[25]),
.Hexs(disp_num[31:0]),
.point(point_out),
.LES(LE_out),
.seg_clk(seg_clk),
.seg_sout(seg_sout),
.SEG_PEN(SEG_PEN),
.seg_clrn(seg_clrn)
);

//U5
Multi_8CH32 U5(
.clk(~Clk_CPU),
.rst(rst),
.EN(GPIOe0000000_we),
.Test(SW_OK[7:5]),
.point_in({clkdiv[31:0],clkdiv[31:0]}),
.LES({64'b0}),
.Data0(Peripheral_in),
.data1({1'b0,1'b0,PC_out[31:2]}),
.data2(spo[31:0]),
.data3(counter_out[31:0]),
.data4(Addr_out[31:0]),
.data5(Data_out[31:0]),
.data6(Data_in[31:0]),
.data7(PC_out[31:0]),
.point_out(point_out),
.LE_out(LE_out),
.Disp_num(disp_num)
);


//U7
SPIO U7(
.clk(~Clk_CPU),
.rst(rst),
.Start(clkdiv[20]),
.EN(GPIOf0000000_we),
.P_Data(Peripheral_in),
.counter_set(counter_set),
.LED_out(led_out),
.led_clk(led_clk),
.led_sout(led_sout),
.led_clrn(led_clrn),
.LED_PEN(LED_PEN),
.GPIOf0()
);
endmodule
```

5.4 下载测试结果

走马灯程序：



数据存储器写入：