实验5 处理器实验

计23 万振南

一、仿真

二、实验数据

1. CPU 的结构图

2. 信号表和状态转移表

LUI 指令

信号表：

状态	wb_addr	wb_cyc	rf_raddr_a	rf_raddr_b	imm_gen_type	alu_operand1	alu_operand2	alu_op	rf_wen	rf_
STATE_IF	pc	1	x	x	x	pc	32'h4	ADD	0	x
STATE_ID	x	0	x	x	U	x	x	x	0	x
STATE_EXE	x	0	x	x	x	32'b0	imm_U	ADD	0	x
STATE_WB	x	0	x	x	x	x	x	x	1	alu_

状态转移和时序：

原状态	新状态	条件	时序操作
STATE_IF	STATE_IF	ack == 0	
STATE_IF	STATE_ID	ack == 1	inst <= wb_data_i; pc_now <= pc; pc <= pc + 4
STATE_ID	STATE_EXE	TRUE	
STATE_EXE	STATE_WB	TRUE	rf_writeback_data <= alu_result
STATE_WB	STATE_IF	TRUE	

BEQ 指令

信号表：

状态	wb_addr	wb_cyc	rf_raddr_a	rf_raddr_b	imm_gen_type	alu_operand1	alu_operand2	alu_op	rf_wen	rf_
STATE_IF	pc	1	x	x	x	pc	32'h4	ADD	0	x
STATE_ID	x	0	rs1	rs2	B	x	x	x	0	x
STATE_EXE	x	0	x	x	x	operand1	operand2	SUB	0	x
STATE_WB	x	0	x	x	x	x	x	x	0	x

状态转移和时序：

原状态	新状态	条件	时序操作
STATE_IF	STATE_IF	ack == 0	
STATE_IF	STATE_ID	ack == 1	inst <= wb_data_i; pc_now <= pc; pc <= pc + 4
STATE_ID	STATE_EXE	TRUE	operand1 <= rf_rdata_a; operand2 <= rf_rdata_b

原状态	新状态	条件	时序操作
STATE_EXE	STATE_WB	TRUE	rf_writeback_data <= alu_result
STATE_WB	STATE_IF	TRUE	if (alu_result == 0) pc <= pc_now + imm_gen_imm

LB 指令

信号表：

状态	wb_addr	wb_cyc	rf_raddr_a	rf_raddr_b	imm_gen_type	alu_operand1	alu_operand2	alu_op	rf_wen	rf_
STATE_IF	pc	1	x	x	x	pc	32'h4	ADD	0	x
STATE_ID	x	0	rs1	x	I	x	x	x	0	x
STATE_EXE	x	0	x	x	x	operand1	imm_I	ADD	0	x
STATE_MEM	alu_result	1	x	x	x	x	x	x	0	x
STATE_WB	x	0	x	x	x	x	x	x	1	wb

状态转移和时序：

原状态	新状态	条件	时序操作
STATE_IF	STATE_IF	ack == 0	
STATE_IF	STATE_ID	ack == 1	inst <= wb_data_i; pc_now <= pc; pc <= pc + 4
STATE_ID	STATE_EXE	TRUE	operand1 <= rf_rdata_a
STATE_EXE	STATE_MEM	TRUE	mem_addr <= alu_result
STATE_MEM	STATE_MEM	wb_ack_i == 0	
STATE_MEM	STATE_WB	wb_ack_i == 1	rf_writeback_data <= wb_dat_i
STATE_WB	STATE_IF	TRUE	

SB 指令

信号表：

状态	wb_addr	wb_cyc	rf_raddr_a	rf_raddr_b	imm_gen_type	alu_operand1	alu_operand2	alu_op	rf_wen	rf_
STATE_IF	pc	1	x	x	x	pc	32'h4	ADD	0	x
STATE_ID	x	0	rs1	rs2	S	x	x	x	0	x
STATE_EXE	x	0	x	x	x	operand1	imm_S	ADD	0	x
STATE_MEM	alu_result	1	x	x	x	x	x	x	0	x
STATE_WB	x	0	x	x	x	x	x	x	0	x

状态转移和时序：

原状态	新状态	条件	时序操作
STATE_IF	STATE_IF	ack == 0	
STATE_IF	STATE_ID	ack == 1	inst <= wb_data_i; pc_now <= pc; pc <= pc + 4
STATE_ID	STATE_EXE	TRUE	operand1 <= rf_rdata_a; operand2 <= rf_rdata_b
STATE_EXE	STATE_MEM	TRUE	mem_addr <= alu_result
STATE_MEM	STATE_MEM	wb_ack_i == 0	
STATE_MEM	STATE_WB	wb_ack_i == 1	

原状态	新状态	条件	时序操作
STATE_WB	STATE_IF	TRUE	

SW 指令

信号表：

状态	wb_addr	wb_cyc	rf_raddr_a	rf_raddr_b	imm_gen_type	alu_operand1	alu_operand2	alu_op	rf_wen	rf_
STATE_IF	pc	1	x	x	x	pc	32'h4	ADD	0	x
STATE_ID	x	0	rs1	rs2	S	x	x	x	0	x
STATE_EXE	x	0	x	x	x	operand1	imm_S	ADD	0	x
STATE_MEM	alu_result	1	x	x	x	x	x	x	0	x
STATE_WB	x	0	x	x	x	x	x	x	0	x

状态转移和时序：

原状态	新状态	条件	时序操作
STATE_IF	STATE_IF	ack == 0	
STATE_IF	STATE_ID	ack == 1	inst <= wb_data_i; pc_now <= pc; pc <= pc + 4
STATE_ID	STATE_EXE	TRUE	operand1 <= rf_rdata_a; operand2 <= rf_rdata_b
STATE_EXE	STATE_MEM	TRUE	alu_result <= operand1 + imm_gen_imm
STATE_MEM	STATE_MEM	wb_ack_i == 0	
STATE_MEM	STATE_WB	wb_ack_i == 1	
STATE_WB	STATE_IF	TRUE	

ADDI 指令

信号表：

状态	wb_addr	wb_cyc	rf_raddr_a	rf_raddr_b	imm_gen_type	alu_operand1	alu_operand2	alu_op	rf_wen	rf_
STATE_IF	pc	1	x	x	x	pc	32'h4	ADD	0	x
STATE_ID	x	0	rs1	x	I	x	x	x	0	x
STATE_EXE	x	0	x	x	x	operand1	imm_I	ADD	0	x
STATE_WB	x	0	x	x	x	x	x	x	1	alu_

状态转移和时序：

原状态	新状态	条件	时序操作
STATE_IF	STATE_IF	ack == 0	
STATE_IF	STATE_ID	ack == 1	inst <= wb_data_i; pc_now <= pc; pc <= pc + 4
STATE_ID	STATE_EXE	TRUE	operand1 <= rf_rdata_a
STATE_EXE	STATE_WB	TRUE	rf_writeback_data <= alu_result
STATE_WB	STATE_IF	TRUE	

ANDI 指令

信号表：

状态	wb_addr	wb_cyc	rf_raddr_a	rf_raddr_b	imm_gen_type	alu_operand1	alu_operand2	alu_op	rf_wen	rf_
STATE_IF	pc	1	x	x	x	pc	32'h4	ADD	0	x
STATE_ID	x	0	rs1	x	I	x	x	x	0	x
STATE_EXE	x	0	x	x	x	operand1	imm_I	AND	0	x
STATE_WB	x	0	x	x	x	x	x	x	1	alu_

状态转移和时序：

原状态	新状态	条件	时序操作
STATE_IF	STATE_IF	ack == 0	
STATE_IF	STATE_ID	ack == 1	inst <= wb_data_i; pc_now <= pc; pc <= pc + 4
STATE_ID	STATE_EXE	TRUE	operand1 <= rf_rdata_a
STATE_EXE	STATE_WB	TRUE	rf_writeback_data <= alu_result
STATE_WB	STATE_IF	TRUE	

ADD 指令

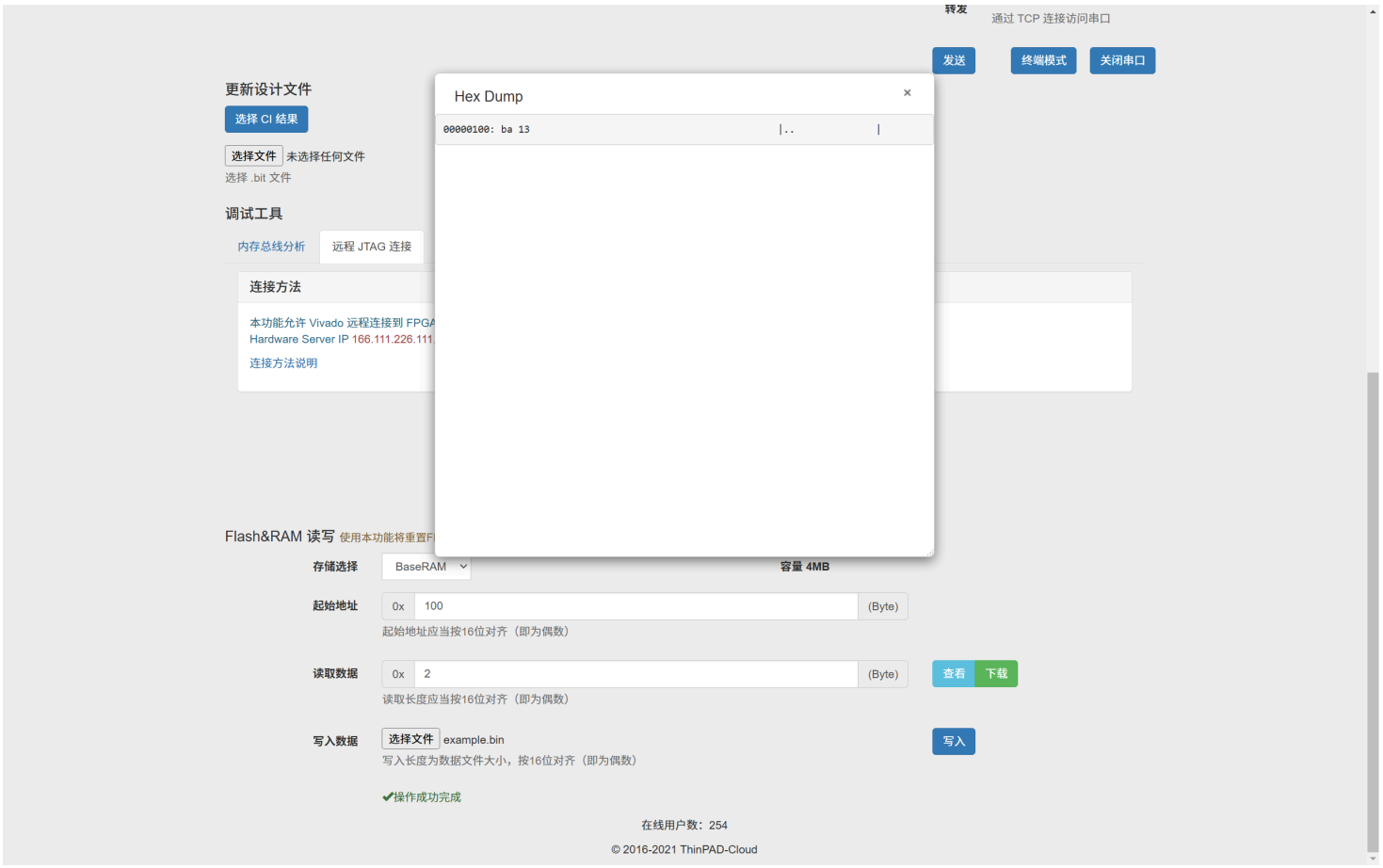
信号表：

状态	wb_addr	wb_cyc	rf_raddr_a	rf_raddr_b	imm_gen_type	alu_operand1	alu_operand2	alu_op	rf_wen	rf_
STATE_IF	pc	1	x	x	x	pc	32'h4	ADD	0	x
STATE_ID	x	0	rs1	rs2	x	x	x	x	0	x
STATE_EXE	x	0	x	x	x	operand1	operand2	ADD	0	x
STATE_WB	x	0	x	x	x	x	x	x	1	alu_

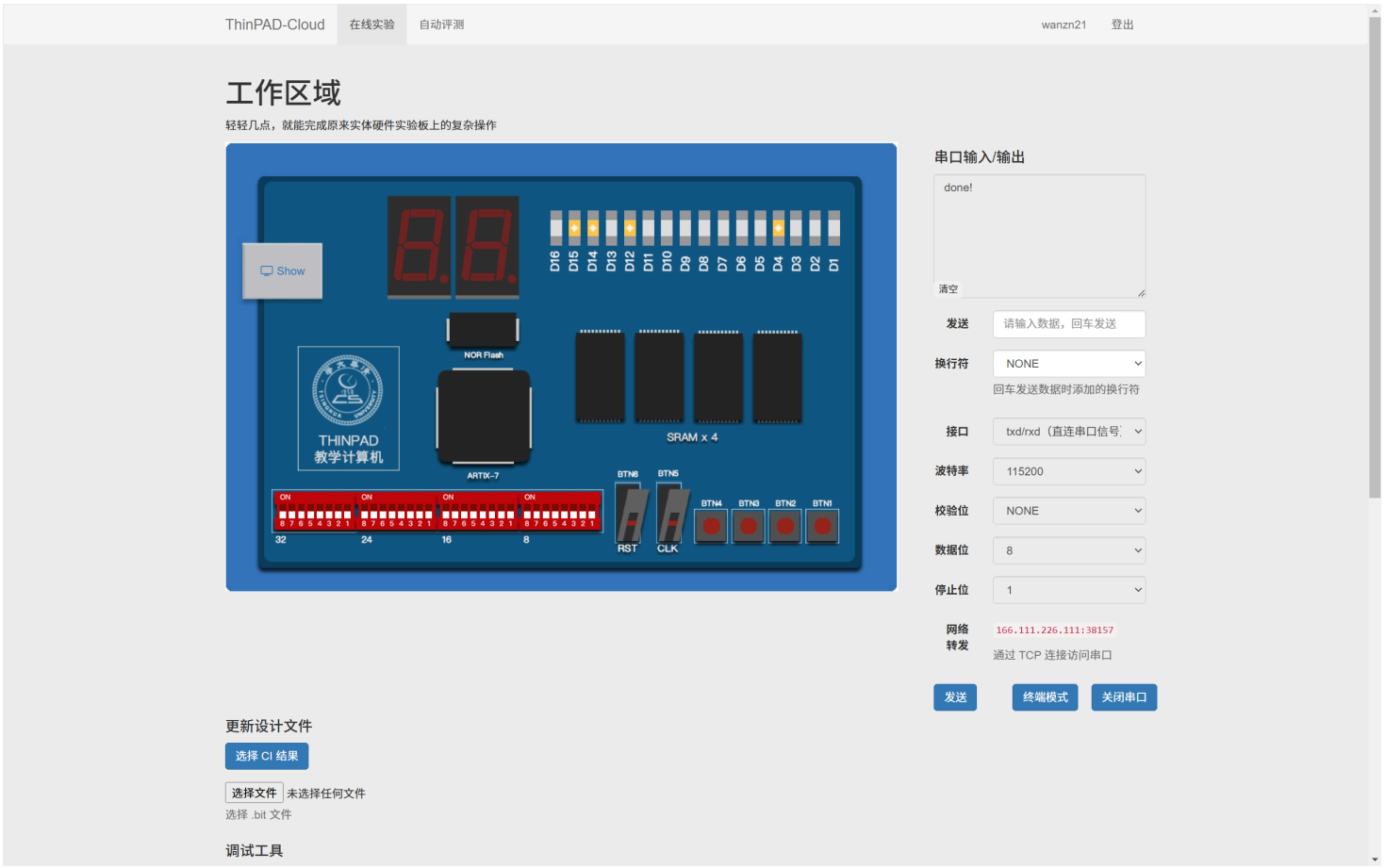
状态转移和时序：

原状态	新状态	条件	时序操作
STATE_IF	STATE_IF	ack == 0	
STATE_IF	STATE_ID	ack == 1	inst <= wb_data_i; pc_now <= pc; pc <= pc + 4
STATE_ID	STATE_EXE	TRUE	operand1 <= rf_rdata_a; operand2 <= rf_rdata_b
STATE_EXE	STATE_WB	TRUE	rf_writeback_data <= alu_result
STATE_WB	STATE_IF	TRUE	

3. 内存数据截图


转发
通过 TCP 连接访问串口
发送
终端模式
关闭串口
更新设计文件
选择 CI 结果
选择文件 未选择任何文件
选择 .bit 文件
调试工具
内存总线分析
远程 JTAG 连接
连接方法
本功能允许 Vivado 远程连接到 FPGA
Hardware Server IP 166.111.226.111
连接方法说明
Hex Dump
00000100: ba 13 |..|
Flash&RAM 读写 使用本功能将重置F
存储选择
BaseRAM
容量 4MB
起始地址
0x 100 (Byte)
起始地址应当按16位对齐（即为偶数）
读取数据
0x 2 (Byte)
查看
下载
读取长度应当按16位对齐（即为偶数）
写入数据
选择文件 example.bin
写入
写入长度为数据文件大小，按16位对齐（即为偶数）
✔操作成功完成
在线用户数：254
© 2016-2021 ThinPAD-Cloud

4. 串口输出截图


ThinPAD-Cloud
在线实验
自动评测
wanzn21
登出
工作区域
轻轻几点，就能完成原来实体硬件实验板上的复杂操作
Show
D16 D15 D14 D13 D12 D11 D10 D9 D8 D7 D6 D5 D4 D3 D2 D1
NOR Flash
SRAM x 4
THINPAD
教学计算机
ARTIX-7
BTN6 BTN5
BTN4 BTN3 BTN2 BTN1
RST CLK
串口输入/输出
done!
清空
发送
请输入数据，回车发送
换行符
NONE
回车发送数据时添加的换行符
接口
txd/rxd（直连串口信号
波特率
115200
校验位
NONE
数据位
8
停止位
1
网络转发
166.111.226.111:38157
通过 TCP 连接访问串口
发送
终端模式
关闭串口
更新设计文件
选择 CI 结果
选择文件 未选择任何文件
选择 .bit 文件
调试工具

三、思考题

1. 流水线 CPU 中，用于 branch 指令的比较器既可以放在 ID 阶段，也可以放在 EXE 阶段。放在这两个阶段分别有什么优缺点？