实验报告

基本单周期 CPU 设计

王 谱越 516030910462

2018-11-26

目录

一、 实验目的 1
二、 实验内容： 1
三、 预习内容： 2
四、 实验器材: 2
五、 实验结果： 2
(一) 实验电路图： 2
(二) 实验工作： 2
(三) 功能验证代码： 3
六、 实验心得： 3
(一) 遇到的问题 4
(二) 实验总结 4

一、 实验目的

1. 理解计算机 5 大组成部分的协调工作原理，理解存储程序自动执行的原理。
2. 掌握运算器、 存储器、控制器的设计和实现原理。重点掌握控制器设计原理
和实现方法。
3. 掌握 I/O 端口的设计方法，理解 I/O 地址空间的设计方法。
4. 会通过设计 I/O 端口与外部设备进行信息交互。

二、 实验内容：

1. 采用 Verilog HDL 在 quartus II 中实现基本的具有 20 条 MIPS 指令的单周期
CPU 设计。
2. 利用实验提供的标准测试程序代码，完成仿真测试。
3. 采用 I/O 统一编址方式，即将输入输出的 I/O 地址空间，作为数据存取空间
的一部分，实现 CPU 与外部设备的输入输出端口设计。实验中可采用高端地址。
4. 利用设计的 I/O 端口，通过 lw 指令，输入 DE2 实验板上的按键等输入设备
信息。即将外部设备状态，读到 CPU 内部寄存器。
5. 利用设计的 I/O 端口，通过 sw 指令，输出对 DE2 实验板上的 LED 灯等输出
设备的控制信号（或数据信息）。即将对外部设备的控制数据，从 CPU 内部的寄存器，写入到外部设备的相应控制寄存器（或可直接连接至外部设备的控制输入信号）。
6. 利用自己编写的程序代码，在自己设计的 CPU 上，实现对板载输入开关或

按键的状态输入，并将判别或处理结果，利用板载 LED 灯或 7 段 LED 数码
管显示出来。
7. 例如，将一路 4bit 二进制输入与另一路 4bit 二进制输入相加，利用两组分别
2 个 LED 数码管以 10 进制形式显示“被加数”和“加数”，另外一组 LED
数码管以 10 进制形式显示“和”等。（具体任务形式不做严格规定，同学可
自由创意）。
8. 在实验报告中，汇报自己的设计思想和方法；并以汇编语言的形式，提供指令集（MIPS）的应用功能的程序设计代码，并提供程序主要流程图。

三、 预习内容：

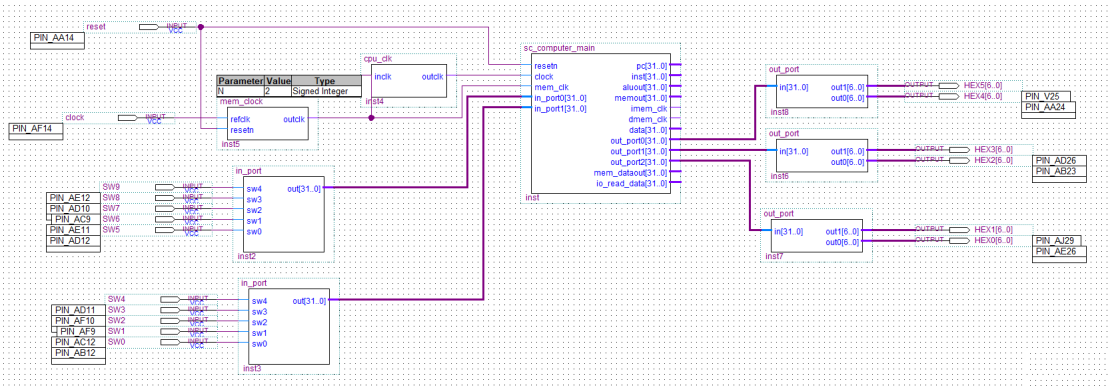
1. 实验前仔细阅读 Altera DE1-SOC User Manual 及相关用户应用数据手册，学习并掌握其板载相关资源的工作原理、连接方式、 和应用注意事项。
2. 根据课程所讲单周期 CPU 设计原理，提前设计并仿真实现相关设计代码。

四、 实验器材:

- 硬件: DE1-SoC 实验板
- 软件: Altera Quartus II 13.1、 Altera ModelSim 10.1d

五、 实验结果：

(一) 实验电路图：



(二) 实验工作：

1. 根据指令真值表, 补全了 sc_cu 模块和 alu 模块, 完成了 cpu 的基本功能。

指令	指令格式	op	rs	rt	rd	sa	func	z	pcsource [1..0]	aluc [3..0]	shift	aluimm	sext	wmem	wreg	m2reg	regrt	call jal
add	add rd, rs, rt	000000	rs	rt	rd	00000	100000	x	0 0	x 0 0 0	0	0	x	0	1	0	0	0
sub	sub rd, rs, rt	000000	rs	rt	rd	00000	100010	x	0 0	x 1 0 0	0	0	x	0	1	0	0	0
and	and rd, rs, rt	000000	rs	rt	rd	00000	100100	x	0 0	x 0 0 1	0	0	x	0	1	0	0	0
or	or rd, rs, rt	000000	rs	rt	rd	00000	100101	x	0 0	x 1 0 1	0	0	x	0	1	0	0	0
xor	xor rd, rs, rt	000000	rs	rt	rd	00000	100110	x	0 0	x 0 1 0	0	0	x	0	1	0	0	0
sll	sll rd, rt, sa	000000	00000	rt	rd	sa	000000	x	0 0	0 0 1 1	1	0	x	0	1	0	0	0
srl	srl rd, rt, sa	000000	00000	rt	rd	sa	000010	x	0 0	0 1 1 1	1	0	x	0	1	0	0	0
sra	sra rd, rt, sa	000000	00000	rt	rd	sa	000011	x	0 0	1 1 1 1	1	0	x	0	1	0	0	0
jr	jr rs	000000	rs	00000	00000	00000	001000	x	1 0	x x x x	x	x	x	0	0	x	x	x
指令	指令格式	op	rs	rt	rd	sa	func		pcsource [1..0]	aluc [3..0]	shift	aluimm	sext	wmem	wreg	m2reg	regrt	call jal
addi	addi rt, rs, imm	001000	rs	rt	imm			x	0 0	x 0 0 0	0	1	1	0	1	0	1	0
andi	andi rt, rs, imm	001100	rs	rt	imm			x	0 0	x 0 0 1	0	1	0	0	1	0	1	0
ori	ori rt, rs, imm	001101	rs	rt	imm			x	0 0	x 1 0 1	0	1	0	0	1	0	1	0
xori	xori rt, rs, imm	001110	rs	rt	imm			x	0 0	x 0 1 0	0	1	0	0	1	0	1	0
lw	lw rt, imm(rs)	100011	rs	rt	imm			x	0 0	x 0 0 0	0	1	1	0	1	1	1	0
sw	sw rt, imm(rs)	101011	rs	rt	imm			x	0 0	x 0 0 0	0	1	1	1	0	x	x	x
beq	beq rs, rt, imm	000100	rs	rt	imm			0	0 0	x 1 0 0	0	0	1	0	0	x	x	x
								1	0 1									
bne	bne rs, rt, imm	000101	rs	rt	imm			0	0 1	x 1 0 0	0	0	1	0	0	x	x	x
								1	0 0									
lui	lui rt, imm	001111	00000	rt	imm			x	0 0	x 1 1 0	x	1	x	0	1	0	1	0
j	j addr	000010	addr					x	1 1	x x x x	x	x	x	0	0	x	x	x
jal	jal addr	000011	addr					x	1 1	x x x x	x	x	x	0	1	x	x	1

2. 根据实验手册，完成了 IO 部分的功能，将 datamem 中 1 开头的地址作为 io 操作的地址。加入了 2 个输入端口 input_port，3 个输出端口 out_port。其中每个输入端口接收 5 个拨动开关的输入，组合为一个长度为 5bit 的数，三个输出端口将 2 进制数转为 10 进制数。其中 2 个负责将输入显示在 LED 管上，另外一个负责展示 2 个输入相加的结果。
3. 编写汇编代码将两个输入相加并输出到相应的输出端口。在实验板上运行并验证正确性。

（三） 功能验证代码：

```
CONTENT
BEGIN
[0..F] : 00000000;   % Range--Every address from 0 to 1F = 00000000 %

0 : 20010080;        % (00) main: addi $1, $0, 128 # outport0                    %
1 : 20020084;        % (04)       addi $2, $0, 132 # outport1                    %
2 : 20030088;        % (08)       addi $3, $0, 136 # outport2                    %
3 : 200400c0;        % (0c)       addi $4, $0, 192 # inport0                     %
4 : 200500c4;        % (10)       addi $5, $0, 196 #inport1                      %
5 : 8c860000;        % (14) loop: lw   $6, 0($4)   # input inport0 to $4         %
6 : 8ca70000;        % (18)       lw   $7, 0($5)   # input inport1 to $5         %
7 : 00c74020;        % (1c)       add  $8, $6, $7  # add inport0 with inport1 to $6 %
8 : ac260000;        % (20)       sw   $6, 0($1)   # output inport0 to outport0    %
9 : ac470000;        % (24)       sw   $7, 0($2)   # output inport1 to outport1    %
A : ac680000;        % (28)       sw   $8, 0($3)   # output result to outport2     %
B : 08000005;        % (2c)       j loop           #                              %
END ;
```

128-136 的 2 进制地址对应 3 个输出端口，192-196 的 2 进制地址对应 2 个输入端口。

之后是一个不断的循环，将 input 端口中的数据利用 lw 读出到 6,7 号寄存器中相加，将结果保持到 8 号寄存器，之后用 sw 输出到相应的输出端口。

六、 实验心得：

(一) 遇到的问题

1. ModelSim 的使用

在 ModelSim 的使用过程中，一开始总无法加载所有模块。检查后发现部分模块有 BUG，所以无法编译，修正后可以正常使用。

之后，发现 ModelSim 中的输入一直为零，无法得到和指导手册上相同的波形结果。之后，更新了 testbench 加入了 input 相关的模拟输入，最终获得了与实验指导手册上相近的结果。

2. ROM 和 RAM 模块的使用：

第一次使用时，不了解如何制定 mif 文件，因此无法把自己写好的指令加载到 CPU 中，之后在模块文件中找到了 mif 的设置项，成功读入自己写好的指令完成测试 •。

3. 输入输出的转化：

由于实验板上的输入位数有限，而数据储存都是 32 位的，因此需要一些单独的组件，将多个拨动开关的输入转化为 32 位数据，将 2 进制输出结果转化为十进制，并通过数码管显示。

(二) 实验总结

本次实验，我根据老师提供的代码完成了一个单周期 CPU，并利用汇编代码成功编写了程序验证了 CPU 实现。实践了 ModelSim 在 CPU 设计中的使用，熟悉了如何定制自己需要的 testbench.同时，学习到了不同的 IO 交互模式，并实践了 IO 与 CPU 交互的实现。通过这次实验，我了解了 CPU 设计中从设计到实现再到验证的整个流程的大概方法。