计算机组成原理实验报告

19373106 裴宝琦

一、CPU 设计方案综述

（一）总体设计概述

本 CPU 为 Logisim 实现的单周期 MIPS - CPU，支持的指令集包含{addu,subu,ori,lui,beq,nop,lw,sw,j}。为了实现这些功能，CPU 主要包含了 IFU、GRF,DM,IM,ALU,control 等模块，有些模块的下层还包含一些用来计算的子模块电路。

（二）关键模块定义

1. GRF

信号名	方向	描述
RESET	I	异步复位信号
WE	I	为 1 时，时钟上沿将 WD 写入对应寄存器
WD	I	回写寄存器值
A1	I	第一个寄存器编号
A2	I	第二个寄存器编号
A3	I	回写寄存器编号
CLK	I	时钟信号
RD1	O	第一个寄存器读出的值
RD2	O	第二个寄存器读出的值

2. DM

信号名	方向	描述
Address	I	地址输入
WD	I	数据写入
WE	I	为 1 时，时钟上沿将 WD 写入对应地址单元
CLK	I	时钟信号
RE	I	异步复位信号
LOAD	I	为 1 时，时钟上沿读出数据
OUTPUT	O	读出地址的值
RD	O	读出数据

3. ALU

信号名	方向	描述
In1	I	第一个操作数
In2	I	第二个操作数
ALUop	I	控制信号
ZERO	O	判断两数是否相等
Result	O	输出操作结果

4.IFU

这个模块里包括了 PC,IM 等模块以及相关逻辑，该模块的目的是在每个周期都输出正确的指令。

信号名	方向	描述
Imm_exp_beq	I	计算出处在 beq 指令时需要加的值
J_addr_exp	I	计算出 j 指令时需要转到的地址
CLK	I	时钟信号

Branch	I	判断是否转到 beq 所指向的地址
PCJump	I	判断是否转到 j 所指向的地址
Reset	I	异步复位信号
Instruction	O	输出指令机器码

5.IFU_splitter

信号名	方向	描述
Instruction	I	输入指令机器码
J_addr	O	J 指令对应的地址
Imm	O	立即数
func	O	对应的功能位
rd	O	寄存器
rt	O	寄存器
Opcode	O	操作码
rs	O	寄存器

6.control

信号名	方向	描述
Opcode	I	操作码
func	I	对应的功能位
RegDst	O	失效选 rt，有效写 rd
RegWrite	O	有效时把数据写入对应寄存器
ALUSrc	O	无效输入 R[rt]，有效选择 SignExt 输出
Branch	O	判断是否是 beq
Memread	O	从 DM 中读数据
MemWrite	O	往 DM 中写数据
MemtoReg	O	无效取 ALU 的输出，有效取 DM 的输出

ALUOp	O	ALU 控制电路
Exp	O	扩展位方式
PCJump	O	判断是否是 j

（三）重要机制实现方法

1. PC 值的确定

通过两个多路选择器确定下一个周期的 PC 值，一个选择器选择 PC+4 和 PC+4+imm_exp，一个选择器选择是否执行 j 通路。

2. RAM 和 ROM 的取值问题

由于取出的指令以及操作 Data 都是按字读取的，所以 RAM 和 ROM 的 address 接口和数据的 7-2 位相连。

3. 控制器真值表及一些说明

	addu	subu	ori	lw	sw	beq	lui	nop	j
opcode	000000	000000	001101	100011	101011	000100	001111	000000	000010
func	100001	100011							
RegDst	1	1	0	0	0	0	0	x	x
RegWrite	1	1	1	1	0	0	1	0	x
ALUSrc	0	0	1	1	1	0	1	x	x
PCSrc	0	0	0	0	0	1	0	0	0
PCJump	0	0	0	0	0	0	0	0	1
MemRead	0	0	0	1	0	0	0	0	0
MemWrite	0	0	0	0	1	0	0	0	0
MemtoReg	0	0	0	1	0	0	0	0	0
ALUctrl	+	-	or	+	+	-	+	x	
	0	1	000002	000000	000000	000001	000000	000000	
EXP	000000	000000	000000	000001	000001	000001	000002	000000	

控制器作为 CPU 的核心部件，通过读取的指令来决定各个模块控制信号的值。控制器采用 and 逻辑和 or 逻辑两个模块，第一个模块得到对应的指令，第二个模块通过相应的指令来决定各个信号的真值。这么做可以在增加指令时可以比较清晰的完成数据通路的建模。

二、测试方案

```
#ori
ori $a0,$zero,0x1000
ori $a1,$a0,0x1234
#addu & subu
j_1:
addu $s0,$a0,$zero
subu $s1,$a1,$a0
#beq
beq $a0,$a1,j_1
beq $a0,$s0,loop1
end:
lui $s5,0x5555
li $v0,10   #在 logisim 中的测试到这里就完成了
syscall
#lw #sw
loop1:
sw $a0,0($zero)
sw $a1,4($zero)
sw $s0,8($zero)
lw $s2,4($zero)
#lui
lui $t7,0x2345
lui $t8,0x3456
j end
```

三、思考题

1. 现在我们的模块中 IM 使用 ROM， DM 使用 RAM， GRF 使用 Register，这种做法合理吗？ 请给出分析，若有改进意见也请一并给出。

合理。因为 ROM 只读不写，而指令也不需要重写，符合指令集的要求。而 RAM 能读也能写，符合对数据写入和读出的要求。而 GRF 就是 32 个有写使能的寄存器构成，和 Register 元件用法相同。

2.事实上，实现 nop 空指令，我们并不需要将它加入控制信号真值表，为什么？请给出你的理由。

在 control 电路的和或逻辑中，nop 指令为全 0，所以经过组合逻辑后的每一个信号值都为 0，此时 GRF、DM 等模块不会进行读写操作，即不需要加入真值表就可以实现 nop 指令。

3. 上文提到，MARS 不能导出 PC 与 DM 起始地址均为 0 的机器码。实际上，可以通过为 DM 增添片选信号，来避免手工修改的麻烦，请查阅相关资料进行了解，并阐释为了解决这个问题，你最终采用的方法。

虽然 MARS 不能导出 PC 与 DM 起始地址均为 0 的机器码，但 logisim 里可以通过设置 RAM 和 ROM 使起始地址都为 0。对于我们目前设计的 CPU，DM 只取了地址位的 2~6 位，所以不增添片选信号并不影响正确运行。

当 DM 位数较多（比如为 32 位时），假设取值范围在 0x30000000~0x3FFFFFFF 区间，这个时候只需要增加片选信号忽略高 4 位地址即可得到正确结果。

2.除了编写程序进行测试外，还有一种验证 CPU 设计正确性的办法——形式验证。 **形式验证**的含义是根据某个或某些形式规范或属性，使用数学的方法证明其正确性或非正确性。请搜索“形式验证（Formal Verification)”了解相关内容后，简要阐述相比于测试，形式验证的优劣之处。

优点：1.对电路中可能出现的所有可能的情况进行验证，更加严密 2. 从数学上完备地证明了电路的实现方案是否确实实现了电路设计所描述的功能，不需要开发测试，操作上比较简单。

缺点： 1. 通过数学方法验证比较繁琐，推导过程容易出现错误。2.随着 CPU 结构复杂度的增加，运算量迅速增长。 3.形式验证只能验证正确性，对性能等其他性质无法检验。