简单的单周期 32 位 CPU

一、CPU 设计方案综述

（一）总体设计概述

本 CPU 为 Logism 实现的单周期 CPU，支持的指令集包含{addu、subu、ori、lw、sw、beq、lui、jal、jr、nop}。为了实现这些功能，CPU 主要包含了 IFU、GRF、ALU、DM、EXT、Controller。

（二）关键模块定义

1. IFU

信号名	方向	描述
clk	I	时钟
reset	I	重置
pcIn	I	32 位下一个指令的地址
pcAdd4	O	32 位当前指令地址+4
pc	O	32 位当前指令地址
instr	O	32 位当前指令值

使用 ROM 存储指令，每次接受一个指令的地址，输出指令值和 pc+4。

2. GRF

信号名	方向	描述
clk	I	时钟
reset	I	重置
WE	I	写入使能
RA1	I	5 位读数地址 1
RA2	I	5 位读数地址 2
WA	I	5 位写入的地址

WD	I	32 位写入的数据
RD1	O	32 位地址 1 的数据
RD2	O	32 位地址 2 的数据

用 32 个寄存器实现，用 wd 译码选择某个寄存器输入，ra1、ra2 多路选择，选择 2 个寄存器输出。

3. ALU

信号名	**方向**	**描述**
A	I	32 位输入数据 1
B	I	32 位输入数据 2
ctrl	I	5 位控制信号
out	O	32 位输出数据
equal	O	有效则 A、B 相等
bigger	O	有效则 A 大于 B
smaller	O	有效则 A 小于 B

aluCtrl：0:A+B, 1:A-B, 4:A&B, 5:A|B, 6:B 加载到高位, 7: A+(B 符号扩展)

4. DM

信号名	**方向**	**描述**
clk	I	时钟
addr	I	5 位读写地址
wd	I	32 位要写入的数据
rd	O	32 位要读出的数据
re	I	读出使能
we	I	写入使能

使用读写分开的 RAM 实现

5.EXT

使用内置 16-32 位扩展器

6.Controller

信号名	方向	描述
Low6	I	指令最低 6 位
Op	I	指令最高 6 位
ifWrGrf	O	Grf 使能
ifWrRt	O	有效时 grf 写入地址为 rt，否则为 rd
ifImmExt	O	是否需要 0 扩展立即数
ifReDm	O	Dm 读数使能
ifWrDm	O	Dm 写入使能
ifBeq	O	是否 beq
ifJal	O	是否 jal
ifJr	O	是否 jr

用最高和最低 6 位控制各个指令的实现，再根据指令选择各个控制信号

（三）重要机制实现方法

1.GrfWa 的选择

```
out = (ifJal)? 5'b11111:
      (ifWrRt)? rt:
      rd;
```

2.GrfWd 的选择

```
out = (ifJal)? pcAdd4:
      (ifReDm)? dmOut:
      aluOut;
```

3.AluB 的选择

```
out = (ifImmExt)? immExt:
```

grfRD2;

4.IfuIn 的选择

out = (ifBeq && ifEqual)? (pcAdd4 + (((immExt[15] == 0)? (immExt) :

(immExt + 32'hffff0000)) << 2)):

(ifJal)? ({pc[31:28], jalTo, 2'b00}):

(ifJr)? (grfRD1):

pcAdd4;

二、测试方案

（一）典型测试样例

1. ALU 功能测试

A=B 时，测试 equal 是否有效

A=13，B=7， 测试 alu_ctrl 为 0,1,4,5,6,7 的输出

2. Controller 功能测试

测试 Controller 表格中的 low6 和 op 的所有信号，检查输出信号

（二）自动测试工具

利用 mars 写测试代码

3401000d

34020006

00221821

00222023

3c050007

ac050064

8c060064

10a6fff8

mars 程序如下：

start:

ori $1, $0, 13

```
ori $2, $0, 6
addu $3, $1, $2
subu $4, $1, $2
lui $5, 7
sw $5, 100
lw $6, 100
beq $5, $6, start
```

三、思考题

（一）

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。

目前比较合理，IM 保存程序，DM 模拟.data 区，一次只操作一个数，而 GRF 模拟 32 个寄存器，涉及两个数的读和一个数的写。但是因为 IM 是 ROM，不方便改变指令，可以对外提供接口增删改指令。

（二）

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。

nop 不控制任何部件，对外表现仅仅是 0 号寄存器或者运算数字 0，也不存取数据，没有影响。

（三）

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。

我的想法是把 mars 的指令地址减去 0x3000 作为新的地址输入到 ROM 中，相当于平移了 0x3000。

（四）

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。

形式验证不需要测试激励，只需要在数学上验证即可。形式验证可以对所有的情况进行验证。但是对于很大规模的电路，形式验证的逻辑可能很复杂，难以得出，需用仿真来补充。