北航2021计组P3设计文档

黄雨石20376156

一、CPU设计方案

(一)总体设计概述

使用 `Logisim` 开发一个简单的单周期 `CPU`，总体概述如下：

1. 此 `CPU` 为 `32` 位 `CPU`
2. 此 `CPU` 为单周期设计
3. 此 `CPU` 支持的指令集为：`{addu, subu, ori, lw, sw, beq, lui, nop}`
4. `nop` 机器码为 `0x00000000`
5. `addu`，`subu` 不支持溢出

(二)关键模块定义

1. **ALU**

   ALU端口说明

端口名	方向	描述
SrcA[31:0]	I	数据1
SrcB[31:0]	I	数据2
Result[31:0]	O	输出的数据
ALUControl[3:0]	I	控制信号 000：加 001：减 010：或 011：比较是否相等 100：左移16位
ALUZero	O	判断A与B是否相等

   ALU功能描述

   1. 加：A+B
   2. 减：A-B
   3. 或：A|B
   4. 是否相等：A==B?
   5. 左移16位：A<<16

2. **IFU**

   一、PC

   PC端口说明

端口名	方向	描述
NPC[31:0]	I	32位输入地址
PC[31:0]	O	32位输出地址
clk	I	时钟信号
reset	I	复位信号

二、NPC

NPC端口说明

端口名	方向	描述
PC[31:0]	I	32位输入地址
im26[25:0]	I	26位立即数
im16[15:0]	I	16位立即数
NPC[31:0]	O	32位输出地址
ALUZero	I	判断A与B是否相等（判断beq）
NPCSel[2:0]	I	控制信号 00：加4 01：beq指令

三、IM

IM端口说明

端口名	方向	描述
Addr[31:0]	I	输入的5位地址信号
RD[31:0]	O	32位指令

IFU功能描述

1. 复位 复位信号有效时，PC被设置为0x00000000
2. 取指令 根据PC当前值从IM中取出指令。
3. 计算下一条指令地址。如果当前指令不是beq指令，PC=PC+4；如果当前指令是beq指令且zero=1，则PC=PC+4+sign_extend(offset||02)

3. **Ctrl**

Ctrl端口说明

端口名	方向	描述
RD[31:0]	I	32位指令信号
NPCSel[2:0]	O	NPC控制信号
ExtOp	O	Ext控制信号
GWDSel[1:0]	O	GRF写入数据控制信号
A3Sel[1:0]	O	GRF写入地址控制信号
GWE	O	GRF写入使能控制信号
SrcBSel[1:0]	O	ALU的SrcB控制信号
ALUControl[3:0]	O	ALU控制信号
DWE	O	DM写入使能控制信号

Ctrl功能描述

端口	addu	subu	ori	lw	sw	lui	beq
op	000000	000000	001101	100011	101011	00111	000100
func	100001	100011					
NPCSel[2:0]	000	000	000	000	000	000	001
ExtOp	0	0	0	1	1	0	0
GWDSel[1:0]	00	00	00	01	00	00	00
A3Sel[1:0]	00	00	01	01	00	01	00
GWE	1	1	1	1	0	1	0
SrcBSel	00	00	01	01	01	01	00
ALUControl[3:0]	0000	0001	0010	0000	0000	0100	0011
DWE	0	0	0	0	1	0	0

4. **Ext**

   Ext端口说明

端口名	方向	描述
im16[15:0]	I	代扩展的 16 位信号
ExtOut[31:0]	O	扩展后的 32 位的信号
Signed	I	无符号或符号扩展选择信号 0：无符号扩展 1：符号扩展

   Ext功能描述

   1. 当 Signed为 0 时，将 imm16 无符号扩展输出
   2. 当 Signed为 1 时，将 imm16 符号扩展输出

5. **GRF**

GRF端口说明

端口名	方向	描述
WD[31:0]	I	写入A3对应寄存器的数据
RD1[31:0]	O	A1地址对应寄存器数值
RD2[31:0]	O	A2地址对应寄存器数值
A1[4:0]	I	RD1中数据的寄存器地址
A2[4:0]	I	RD2中数据的寄存器地址
A3[4:0]	I	写入数据的寄存器地址
WE	I	写使能信号
clk	I	时钟信号
reset	I	复位信号

GRF功能描述

1. 复位。复位信号有效时所有寄存器的数值被设置为0x00000000
2. 读寄存器 根据当前输入的地址信号读出32位数据
3. 写寄存器。根据当前输入的地址信号，把输入的数据写入相应寄存器

6. **DM**

DM端口说明

端口名	方向	描述
Addr[4:0]	I	数据的地址信号
WD[31:0]	I	当WE为高电平时，写入地址A的数据
RD[31:0]	O	读出地址A的数据
WE	I	高电平时允许写入数据
clk	I	时钟信号
reset	I	复位信号

DM功能描述

1. 复位：复位信号有效时，所有数据被设置为0x00000000
2. 读：根据当前输入的寄存器地址读出数据
3. 写数据：根据输入地址，写入输入的数据

二、测试方案

(1)测试代码

mips

```
.text
ori $11,$9,25679
```

```
lui $23,63605
sw $3,24($9)
ori $16,$19,60938
subu $21,$25,$14
lui $0,21561
lui $12,43549
addu $0,$24,$18
ori $6,$26,33812
nop
subu $15,$21,$0
ori $5,$0,13021
lw $6,24($17)
lw $24,36($4)
lui $10,34999
ori $10,$4,2375
lw $7,4($9)
subu $26,$2,$0
ori $18,$8,62986
beq $5,$9,branch4
sw $26,24($22)
lui $5,5108
branch1:
ori $23,$5,7026
nop
subu $2,$18,$23
branch2:
lw $7,12($4)
nop
lui $11,29264
branch4:
subu $10,$26,$13
```

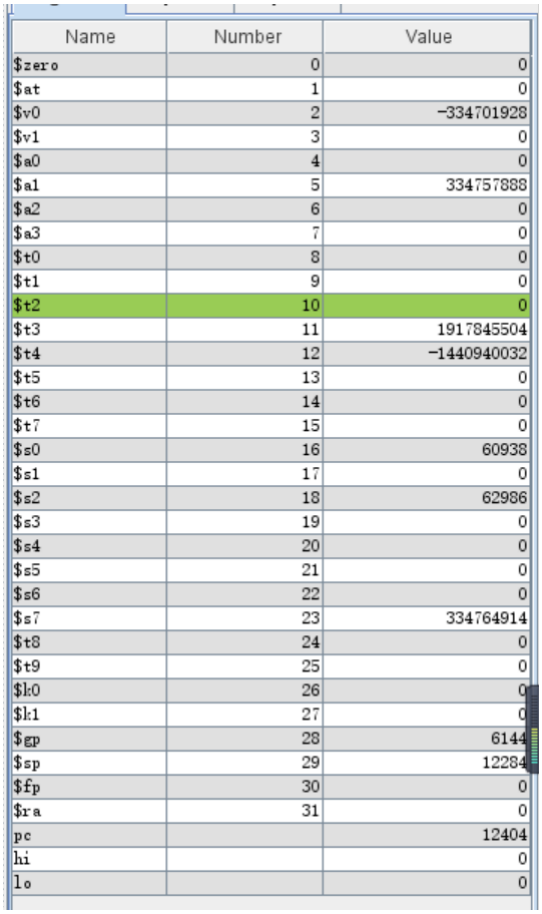
机器码

```
v2.0 raw
352b644f
3c17f875
ad230018
3670ee0a
032ea823
3c005439
3c0caa1d
03120021
37468414
00000000
02a07823
340532dd
8e260018
8c980024
3c0a88b7
348a0947
8d270004
0040d023
3512f60a
10a90008
aeda0018
3c0513f4
```

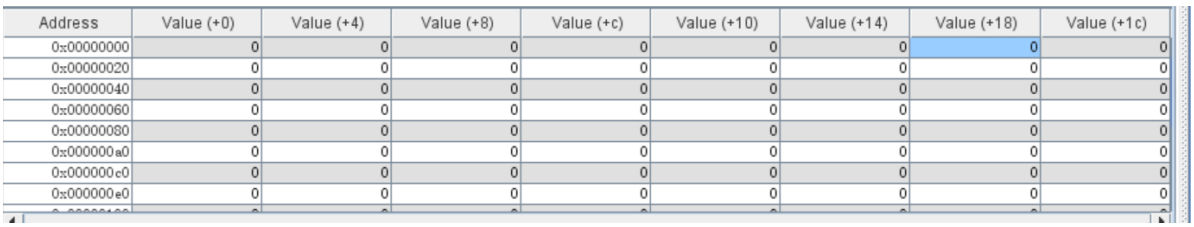
```
34b71b72
00000000
02571023
8c87000c
00000000
3c0b7250
034d5023
```

(2)模拟结果

mars

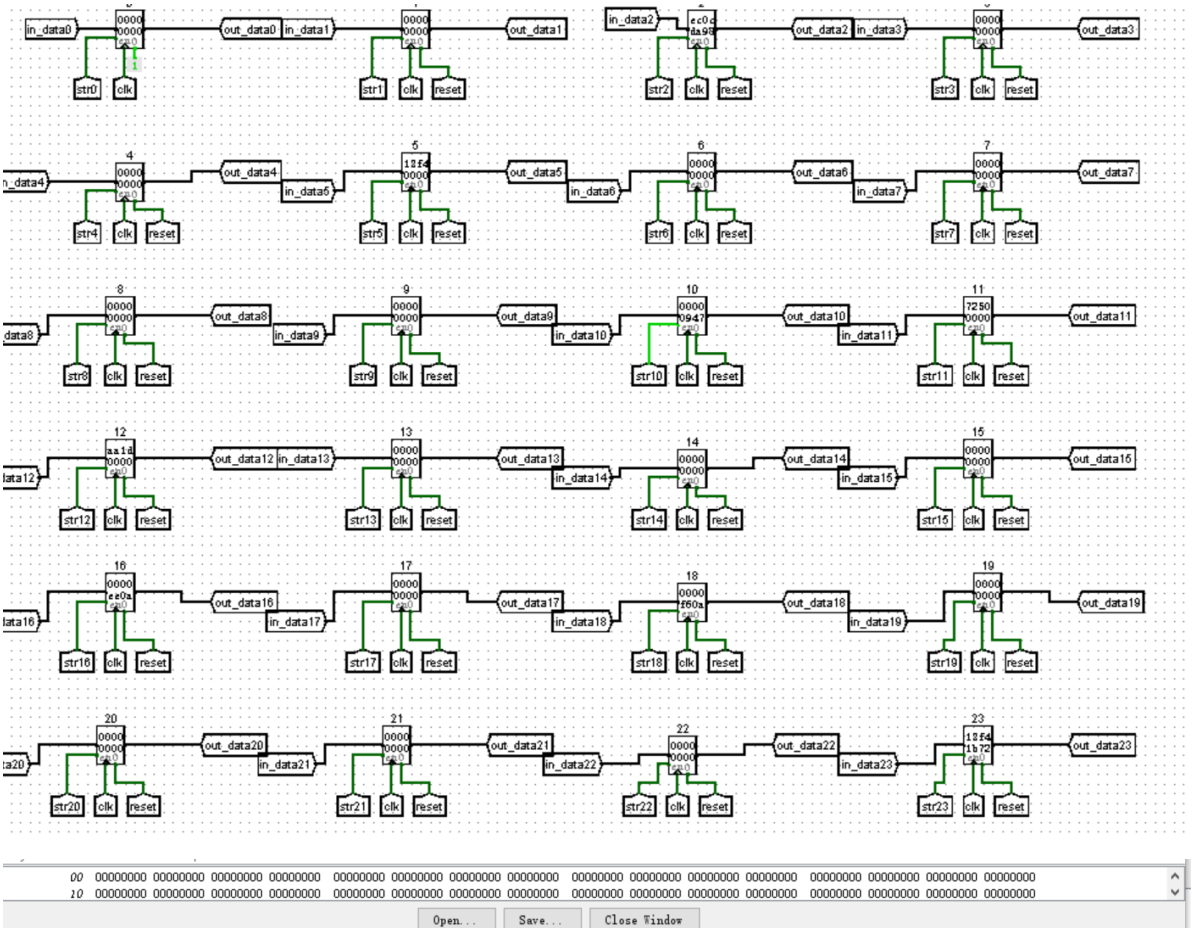


Name	Number	Value
$zero	0	0
$at	1	0
$v0	2	-334701928
$v1	3	0
$a0	4	0
$a1	5	334757888
$a2	6	0
$a3	7	0
$t0	8	0
$t1	9	0
$t2	10	0
$t3	11	1917845504
$t4	12	-1440940032
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	60938
$s1	17	0
$s2	18	62986
$s3	19	0
$s4	20	0
$s5	21	0
$s6	22	0
$s7	23	334764914
$t8	24	0
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	6144
$sp	29	12284
$fp	30	0
$ra	31	0
pc		12404
hi		0
lo		0



Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x00000000	0	0	0	0	0	0	0	0
0x00000020	0	0	0	0	0	0	0	0
0x00000040	0	0	0	0	0	0	0	0
0x00000060	0	0	0	0	0	0	0	0
0x00000080	0	0	0	0	0	0	0	0
0x000000a0	0	0	0	0	0	0	0	0
0x000000c0	0	0	0	0	0	0	0	0
0x000000e0	0	0	0	0	0	0	0	0

cpu



二者结果一致

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。

答：ROM是只读存储器，适合存储固定不变的数据，而指令数据在单周期cpu运行过程有限个周期中不变的，因此选用ROM是合理的。RAM是读写存储器，DM要求存储器可以读和写，且速度要求不高，因此选取速度不高但是功能齐全的RAM比较合理。GRF需要经常对数据进行读写且速度要求比较高，因此选取Register比较合理。

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。

答：nop指令数据是0x00000000，除了PC=PC+4之外，没有进行任何其他操作，因此没有对电路中的逻辑真值运算产生任何影响，存在与否对电路无影响。

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。

答：如果很不幸，寄存器中存储的 DM 的地址被映射在 0x3000_0000 到 0x3fff_ffff 间，而我们的 DM 起始地址是 0，那么，我们可以将输 入地址直接减去 0x3000_0000，再作为 DM 的地址输入。

假如我们不确定寄存器中的存储的 DM 地址的起始值，我们可以将 其与 0x3000_0000 比较，得到片选信号。

4.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。

答：所谓形式验证，是指从数学上完备地证明或验证电路的实现方案是否确实实现了电路设计所描述的功能。形式验证方法分为等价性验证、模型检验和定理证明等。

对组合逻辑来说，不存在状态寄存器，其输出值仅仅依赖于当前的输入值。这时只要对每个输入值组合证明其输出的数据组合相同即可。

对时序逻辑而言，可以把它看成一个有限状态机。电路功能的等价可以用有限状态机的等价来判断。假定有两个状态机A和B，对他们的验证可以转化为以下的方法，当A和B有相同的接口，而且从相同的初始状态出发，两者对有效输入值序列产生相同的输出值序列，则可以说A和B等价。

形式验证的优点是：(1)形式验证是对指定描述的所有可能的情况进行验证，覆盖率达到了100%。(2)形式验证技术是借用数学上的方法将待验证电路和功能描述或参考设计直接进行比较，不需要开发测试激励。(3)形式验证的验证时间短，可以很快发现和改正电路设计中的错误，可以缩短设计周期。

缺点是：验证方法复杂抽象，难以准确把握。而且形式验证是数学逻辑分析，而不是电路分析，不能有效的验证电路的性能，如电路的时延和功耗等