单周期CPU设计说明文档

一、模块规格

1、IFU

（1）端口说明

|  |  |  |
| --- | --- | --- |
| 端口名 | 方向 | 描述 |
| CLK | I | 时钟信号 |
| RST | I | 异步复位信号 |
| NPCOp | I | 计算NPC功能选择 |
| IMM[15:0] | I | 16位立即数 |
| INSTR[31:0] | O | 输出32位指令 |

（2）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 复位 | 当RST为高电平时PC异步复位为0x0000\_0000 |
| 2 | 取出指令 | 根据PC的值从IM中取出指令 |
| 3 | 次地址计算 | NPCOp=0，计算顺序地址（PC+4）  NPCOp=1，计算beq地址 |

2、GRF

（1）端口说明

|  |  |  |
| --- | --- | --- |
| 端口名 | 方向 | 描述 |
| A1[4:0] | I | 第1个读出寄存器的编号 |
| A2[4:0] | I | 第2个读出寄存器的编号 |
| A3[4:0] | I | 写入寄存器的编号 |
| RD1[31:0] | O | A1指向寄存器的值 |
| RD2[31:0] | O | A2指向寄存器的值 |
| WD[31:0] | I | 写入寄存器的值 |
| WE | I | 写入使能 |
| CLK | I | 时钟信号 |
| RST | I | 异步复位信号 |

（2）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 复位 | RST=1时所有寄存器复位为0x0000\_0000 |
| 2 | 读出寄存器 | A1和A2对应的32位寄存器值分别通过RD1和RD2输出 |
| 3 | 写入寄存器 | WE=1时，WD的值写入A3所指的寄存器 |

3、ALU

（1）端口说明

|  |  |  |
| --- | --- | --- |
| 端口名 | 方向 | 描述 |
| A[31:0] | I | 第1个32位操作数 |
| B[31:0] | I | 第2个32位操作数 |
| C[31:0] | O | 32位计算结果 |
| ALUOp[1:0] | I | ALU功能选择 |
| Zero | O | A和B相等比较结果 |

（2）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | ALU数学逻辑计算 | ALUOp=00，C=A+B  ALUOp=01，C=A-B  ALUOp=10，C=A|B  ALUOp=11，保留 |
| 2 | 相等比较 | Zero=1，A=B  Zero=0，AB |

4、DM

（1）端口说明

|  |  |  |
| --- | --- | --- |
| 端口名 | 方向 | 描述 |
| Addr[31:0] | I | 地址输入 |
| WD[31:0] | I | 写入数据 |
| RD[31:0] | O | 读出数据 |
| WE | I | 写入使能 |
| CLK | I | 时钟信号 |
| RST | I | 异步复位信号 |

（2）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 复位 | RST=1时，RAM复位为零 |
| 2 | 读出数据 | 将Addr所表示的内存地址中的值读出 |
| 3 | 写入数据 | WE=1时，将WD的值写入Addr所表示的内存地址中 |

5、EXT

（1）端口说明

|  |  |  |
| --- | --- | --- |
| 端口名 | 方向 | 描述 |
| IMM | I | 16位立即数输入 |
| EXT32 | O | 32位扩展结果输出 |
| EXTOp[1:0] | I | 扩展功能信号 |

（2）功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 扩展立即数 | EXTOp=00，进行零扩展  EXTOp=01，进行符号扩展  EXTOp=10，进行LUI扩展  EXTOp=11，保留 |

6、Controller

（见第二部分：控制器设计）

二、控制器设计

1、端口定义及功能说明

|  |  |  |
| --- | --- | --- |
| 端口名 | 方向 | 描述 |
| opcode | I | Instr[31:26] |
| funct | I | Instr[5:0] |
| Zero | I | 相等比较信号 |
| DMWr | O | DM写使能信号 |
| ALUOp | O | ALU运算信号 |
| BSel | O | ALU的B端MUX选择信号：  0：寄存器RD2值  1：EXT扩展数据 |
| EXTOp | O | EXT扩展信号 |
| RFWr | O | GRF写使能信号 |
| WDSel | O | 寄存器的WD端MUX选择信号：  0：ALU运算结果  1：DM输出数据 |
| WRSel | O | 寄存器的A3端MUX选择信号：  0：Instr[20:16]  1：Instr[15:11] |
| NPCOp | O | IFU中NPC运算信号 |

2、真值表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Instr | addu | subu | ori | lw | sw | beq | lui |
| Op | 000000 | 000000 | 001101 | 100011 | 101011 | 000100 | 001111 |
| Func | 100001 | 100011 | NA | NA | NA | NA | NA |
| DMWr | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| ALUOp | 00 | 01 | 10 | 00 | 00 | 01 | X0 |
| BSel | 0 | 0 | 1 | 1 | 1 | X | 1 |
| EXTOp | XX | XX | 00 | 01 | 01 | XX | 10 |
| RFWr | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| WDSel | 0 | 0 | 0 | 1 | X | X | 0 |
| WRSel | 1 | 1 | 0 | 0 | X | X | 0 |
| NPCOp | 0 | 0 | 0 | 0 | 0 | Zero | 0 |

三、测试模块

1、测试A

测试程序：

ori $s0, $s0, 1

ori $a0, $a0, 2

ori $v0, $s0, 2

ori $t1, $0, 65535

期望结果：

$s0 = 0x0000\_0001

$a0 = 0x0000\_0002

$v0 = 0x0000\_0003

$t0 = 0x0000\_ffff

2、测试B

测试程序：

lui $t0, 65535

ori $t0, $t0, 65535

ori $t1, $t0, 65535

lui $t1, 65535

期望结果：

$t0 = 0xffff\_ffff

$t1 = 0xffff\_0000

3、测试C

测试程序：

lui $s0, 65535

ori $s0, 65535

ori $s1, $0, 1

addu $t0, $0, $s1

subu $t1, $0, $s1

addu $s0, $s0, $s1

subu $s1, $s1, $s1

期望结果：

$t0 = 0x0000\_0001

$t1 = 0xffff\_ffff

$s0 = 0x0000\_0000

$s1 = 0x0000\_0000

4、测试D

测试程序：

.data

number: .space 12

.text

ori $t0, $0, 1

ori $t1, $0, 2

ori $t2, $0, 3

ori $s0, $0, 0

sw $t0, 0($s0)

sw $t1, 4($s0)

sw $t2, 8($s0)

ori $s0, $0, 8

lw $t0, 0($s0)

lw $t1, -4($s0)

lw $t2, -8($s0)

期望结果：

$s0 = 0x0000\_0008

$t0 = 0x0000\_0003

$t1 = 0x0000\_0002

$t2 = 0x0000\_0001

4、测试E

测试程序：

ori $s0, $0, 1

ori $s1, $0, 1

nop

beq $s1, $s0, pass

nop

subu $s2, $s0, $s1

nop

pass:

nop

addu $s2, $s0, $s1

期望结果：

$s0 = 0x0000\_0001

$s1 = 0x0000\_0001

$s2 = 0x0000\_0002

四、思考题

1、若PC（程序计数器）位数为30位，试分析其与32位PC的优劣。

优点：30位的PC可以在处理数据位数后直接连接至指令存储器ROM的地址端口，无需同32位一样先对PC值左移2位后再进行处理。并且30位的PC在非跳转时只需自增为1，降低计数器设计复杂度

缺点：30位的PC与32位机器字长不搭配，不利于系统结构的设计统一。

并且当IM与DM合并后，若需要以字节访问主存，会增加设计复杂度

2、现在我们的模块中 IM使用ROM， DM使用RAM， GRF使用寄存器，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。

本次实验设计的是单周期CPU，因此将通用计算机中的主存分为IM和DM，有助于简化设计复杂度。IM使用ROM，有效的利用了ROM的只读性与非易失性的特点，让我们更好的利用IM导入和保存指令代码。DM使用RAM，使得我们可以使用lw、sw进行数据存储与读取。GRF使用寄存器，实现了两端口读加一端口写的功能，让设计变得更为合理。

3、结合上文给出的样例真值表，给出RegDst， ALUSrc， MemtoReg，RegWrite, nPC\_Sel, ExtOp与op和func有关的布尔表达式（表达式中只能使用“与、或、非”3 种基本逻辑运算。）

（1）逻辑与

rType = ~op[5] ~op[4] ~op[3] ~op[2] ~op[1] ~op[0]

addu = rtype f[5] ~f[4] ~f[3] ~f[2] ~f[1] f[0]

subu = rtype f[5] ~f[4] ~f[3] ~f[2] f[1] f[0]

ori = ~op[5] ~ op[4] op[3] op[2] ~op[1] op[0]

lw = op[5] ~ op[4] ~op[3] ~op[2] op[1] op[0]

sw = op[5] ~ op[4] op[3] ~op[2] op[1] op[0]

beq = ~op[5] ~ op[4] ~op[3] op[2] ~op[1] ~op[0]

lui = ~op[5] ~ op[4] op[3] op[2] op[1] op[0]

（2）逻辑或

DMWr = sw

ALUOp[0] = subu + beq

ALUOp[1] = ori

BSel = ori + sw + lw + lui

EXTOp[0] = sw + lw

EXTOp[1] = lui

RFWr = addu + subu + ori + lw + lui

WDSel = lw

WRSel = addu + subu

NPCOp = beq zero

4、充分利用真值表中的 X 可以将以上控制信号化简为最简单的表达式， 请给出化简后的形式。

上题中已是最简形式。

5、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空指令对于电路各个部件不触发有关写使能的控制信号，其他信号都可为X，故化简后其真值表可处处为零，因此不用对其进行逻辑运算

6、前文提到，“可能需要手工修改指令码中的数据偏移”，但实际上只需再增加一个 DM片选信号,就可以解决这个问题。请阅读相关资料并设计一个 DM 改造方案使得无需手工修改数据偏移。

由于本次实验DM的需求地址宽度为5bit，故可设计一片选信号，当输入32位地址后，若高27位为0x0000100，则令片选信号有效，即可解决偏移问题。

7、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，简要阐述相比与测试，形式验证的优劣。

优点：

(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：

(1)形式验证需要高额的成本，是一种需要极好技术水平的验证方法。

(2)形式验证技术只是数学上的证明，不能代替在真实环境中的测试验证。

(3)形式验证不能对过大型电路进行验证，它耗费的时间太多。