MIPS单周期处理器设计文档

MIPS单周期处理器设计文档由顶层结构、各模块规格、控制器设计、测试程序组成。

1. 顶层结构

从顶层视图来看，MIPS单周期处理器模块主要分为IFU（取指令单元）、GRF（通用寄存器组）、ALU（算术逻辑单元）、DM（数据存储器）、EXT（位扩展器）、Controller（控制器）以及shift left 16（左移十六位）。除此之外，还有数个多路开关、逻辑门、Splitter、时钟信号、reset信号。

IFU、GRF、DM、共用一个时钟信号。

reset信号为顶层唯一有效驱动源。

顶层信号端口

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 信号 | 位数 | 功能描述 |
| 1 | reset | 1 | 当reset=1时，PC被设置为0x00000000，GPR所有寄存器置零，DM所有数据置零 |
| 2 | clk | 1 | IFU、GRF、DM的时钟信号 |

1. 各模块规格（除控制器）
2. IFU

IFU信号端口

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 信号 | 位数 | 方向 | 描述 |
| 1 | reset | 1 | I | 复位信号 |
| 2 | clk | 1 | I | 时钟信号 |
| 3 | if beq zero | 1 | I | 信号为1时执行beq指令，修改PC值，否则PC=PC+4 |
| 4 | Instr | 32 | O | 输出32位指令 |

IFU模块功能

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 复位 | reset=1时，PC被置为0x00000000 |
| 2 | 取指令 | 以PC为地址从IM中取出32位指令 |
| 3 | 修改PC值 | 在时钟上升沿时，当if beq zero=1时，按beq指令修改PC，否则PC=PC+4 |
| 4 | 输出 | Instr输出取出的32位指令 |

1. GRF

GRF信号端口

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 信号 | 位数 | 方向 | 描述 |
| 1 | RA1 | 5 | I | 读寄存器文件时第一个寄存器的下标 |
| 2 | RA2 | 5 | I | 读寄存器文件时第二个寄存器的下标 |
| 3 | WA | 5 | I | 写入文件的寄存器下标 |
| 4 | WD | 32 | I | 寄存器文件写入数据 |
| 5 | clk | 1 | I | 时钟信号 |
| 6 | reset | 1 | I | 复位信号 |
| 7 | RegWrite | 1 | I | 寄存器文件写使能 |
| 8 | RD1 | 32 | O | 读寄存器文件时第一个寄存器的输出 |
| 9 | RD2 | 32 | O | 读寄存器文件时第二个寄存器的输出 |

GRF模块功能

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 读寄存器 | RD1输出RA1所寻址的寄存器的文件数据，RD2输出RA2所寻址的寄存器的文件数据 |
| 2 | 写寄存器 | 当时钟上升沿到来时，并且RegWrite有效时，WD被写入WA所寻址的寄存器 |
| 3 | 复位 | GRF所有寄存器置零 |

1. ALU

ALU信号端口

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 信号 | 位数 | 方向 | 描述 |
| 1 | A | 32 | I | ALU的第一个操作数 |
| 2 | B | 32 | I | ALU的第二个操作数 |
| 3 | ALUOP | 4 | I | 选择ALU运算类型的信号 |
| 4 | C | 32 | O | ALU的运算结果 |
| 5 | If = | 1 | O | 判断A与B是否相等，若相等输出1，否则输出0 |

ALU模块功能

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 加法 | 输出端C=A+B |
| 2 | 减法 | 输出端C=A-B |
| 3 | 或 | 输出端C=A|B |
| 4 | 判断相等 | 若A=B，if =信号输出1，否则输出0 |

1. DM

DM信号端口

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 信号 | 位数 | 方向 | 描述 |
| 1 | MemAddr | 5 | I | 写入数据储存器的数据的地址/读取数据储存器的数据的地址 |
| 2 | MemData | 32 | I | 写入数据储存器的数据 |
| 3 | MemWrite | 1 | I | 写入控制信号 |
| 4 | clk | 1 | I | 时钟信号 |
| 5 | reseet | 1 | I | 复位信号 |
| 6 | Data | 32 | O | 从数据存储器读取的数据 |

DM模块功能

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 读数据储存器 | Data输出数据寄存器在MemAddr地址的数据 |
| 2 | 写数据储存器 | 当MemWrite有效并且时钟为上升沿时，MemData被写入数据存储器地址为MemAddr的区域 |
| 3 | 复位 | DM所有数据置零 |

1. EXT

EXT信号端口

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 信号 | 位数 | 方向 | 描述 |
| 1 | A | 16 | I | 位扩展输入信号 |
| 2 | 0 or sign | 1 | I | 信号为1时，进行符号扩展，信号为0时，进行0扩展 |
| 3 | B | 32 | O | 位扩展输出信号 |

EXT模块功能

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 符号扩展 | 当0 or sign信号为1时，A符号扩展后由B输出 |
| 2 | 0扩展 | 当0 or sign信号为0时，在A前置16个0后由B输出 |

1. shift left 16

Shift left 16信号端口

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 信号 | 位数 | 方向 | 描述 |
| 1 | A | 16 | I | 左移输入 |
| 2 | B | 32 | O | 左移后输出信号 |

Shift left 16模块功能

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能名称 |
| 1 | 左移16位 | 将A左移16位后由B输出 |

1. shift left 2

Shift left 2信号端口

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 信号 | 位数 | 方向 | 描述 |
| 1 | A | 32 | I | 左移输入 |
| 2 | B | 32 | O | 左移后输出信号 |

Shift left 2模块功能

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能名称 |
| 1 | 左移2位 | 将A左移2位后由B输出 |

1. 控制器设计

以下为控制信号产生的真值表：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| func | 100001 | 100011 | 无 | | | | |  |
| opcode | 000000 | 000000 | 001101 | 100011 | 101011 | 000100 | 001111 |  |
| 控制信号 | addu | subu | ori | lw | sw | beq | lui | nop |
| if beq | 0 | 0 | 0 | 0 | 0 | 1 | 0 | x |
| Ext | x | x | 0 | 1 | 1 | x | 0 | x |
| RegDst | 0 | 0 | 1 | 1 | x | x | 1 | x |
| RegWrite | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 |
| RegWData[1:0] | 00 | 00 | 00 | 01 | x | x | 10 | x |
| ALUSrc | 0 | 0 | 1 | 1 | 1 | x | x | x |
| ALUOP[3:0] | 0000 | 0001 | 0010 | 0000 | 0000 | x | x | x |
| MemWrite | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

以下为各控制信号功能：

if beq与ALU的if=信号同为1时，if beq zero=1，PC执行beq跳转。

Ext信号为0时，执行0扩展；为1时，执行符号扩展。

RegDst为0时，GRF的WA端口输入为rd，为1时，GRF的WA端口输入为rt.

RegWrite为0时，GRF写使能端无效，为1时，GRF写使能端有效。

RegWData[1:0]为00时，GRF的WD端口输入为ALU的C端口，为01时，GRF的WD端口输入为DM的Data端口，为10时GRF的WD端口输入为shift left 16输出端B。

ALUSrc为0时，ALU的B端口输入为GRF的RD2端口，为1时，ALU的B端口输入为EXT的B端口。

ALUOP[3:0]为0000时，ALU执行加法，为0001时，ALU执行减法，为0010时，ALU执行或操作。

MemWrite为0时，DM写入控制信号无效，为1时，DM写入控制信号有效。

1. 测试程序

汇编语言：

nop

lui $2,0x0101

sw $2,4($0)

nop

lw $3,4($0)

addu $2,$2,$3

subu $4,$2,$4

beq $4,$2,label

lw $1,4($0)

label:

ori $5,$2,1

nop

sw $5,8($0)

机器码：

00000000 3c020101 ac020004 00000000 8c030004 00431021 00442023 108200001 8c010004 34450001 00000000 ac050008

测试期望：指令结束后，1号寄存器值为0x00000000,2、4号寄存器值为0x02020000,3号寄存器值为0x01010000,5号寄存器值为0x02020001,DM的地址为4处所存值为0x01010000,DM的地址为8处所存值为0x02020001。

思考题：

1. 在上个学年的计组课程中，PC（程序计数器）位数被规定为30位，试分析其与32位PC的优劣。

PC使用30位时，忽略32位最低的两位，32位PC加4实际上是30位PC加1，而ROM两个相邻储存单元地址相差1，用30位符合logism的情况，实际上ROM两个相邻储存单元地址相差4，使用32位符合实际。

1. 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，指令执行中，IM不会被写入数据，只会被读取指令，使用ROM合适。指令执行中，DM有数据读写，因此用RAM合适。GRF使用寄存器对应在MIPS指令中的32个通用寄存器，合理。

1. 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。）

RegDst=~op0~op1~op2~op3~op4~op5

ALUSrc=op0~op1op2op3~op4~op5+op0op1~op2~op3~op4op5+op0op1~op2op3~op4op5

MemtoReg=op0op1~op2~op3~op4op5

RegWrite=~op0~op1~op2~op3~op4~op5+op0~op1op2op3~op4~op5+op0op1~op2~op3~op4op5

nPC\_Sel=~op0~op1op2~op3~op4~op5

ExtOp=op0op1~op2~op3~op4op5+op0op1~op2op3~op4op5

令ALUctr[2:0]:

00 Add

01 Subtract

10 Or

AlUctr0=~op0~op1~op2~op3~op4~op5~func0func1~func2~func3~func4func5+~op0~op1op2~op3~op4~op5

AlUctr1=op0~op1op2op3~op4~op5

1. 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。

RegDst=~op0~op1~op2~op3~op4~op5

ALUSrc=op0~op1op2op3~op4~op5+op0op1~op2~op3~op4op5+op0op1~op2op3~op4op5

MemtoReg=op0op1~op2~op3~op4op5

RegWrite=~op0~op1~op2~op3~op4~op5+op0~op1op2op3~op4~op5+op0op1~op2~op3~op4op5

nPC\_Sel=~op0~op1op2~op3~op4~op5

ExtOp=op0op1~op2~op3~op4op5+op0op1~op2op3~op4op5

令ALUctr[2:0]:

00 Add

01 Subtract

10 Or

AlUctr0=~op0~op1~op2~op3~op4~op5~func0func1~func2~func3~func4func5+~op0~op1op2~op3~op4~op5

AlUctr1=op0~op1op2op3~op4~op5

1. 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指令为0x00000000,如果不加入控制信号真值表，当IFU输出nop时，寄存器写使能信号RegWrite、数据储存器写使能信号MemWrite、npc\_sel都为无效，nop指令实际上无法改变GRF、DM、PC中的数据，所以不用考虑。

1. 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DM、IM仅仅取了5位地址，当数据偏移量过大，可能出现2个32位地址所取5位地址相同，但实际上两个地址不同的情况。可将除5位地址外的剩余地址作为片选信号，同时增加IM、DM片数。

1. 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。

形式验证优点：运用数学理论方法，对所有可能进行验证，验证时间短。

形式验证缺点：不能有效的验证电路的实际性能，如电路的时延和功耗等。