单周期 CPU 设计文档(P4)

17373252 丁禹衡

一、模块规格

1. PC（程序计数器）

端口说明：

表 1 PC 端口说明

信号名	方向	描述
clk	I	时钟信号
reset	I	复位信号，将 PC 置为 0x00003000 1：复位 0：无效
nPC[31:0]	I	下一条指令地址
PC[31:0]	O	当前指令地址

功能定义：

表 2 PC 功能定义

序号	功能名称	功能描述
1	复位	当复位信号有效时，PC 被设置为 0x00003000
2	计数	当时钟上升沿到来时，PC 更新为 nPC 的输入值

2. IM（指令存储器）

端口说明：

表 3 IM 端口说明

信号名	方向	描述
Addr[31:0]	I	当前指令地址
Instr[31:0]	O	Addr 指定的当前指令

功能定义：

表 4 IM 功能定义

序号	功能名称	功能描述
1	取指令	输出 Addr 指定的当前指令

3. GRF（通用寄存器文件）

端口说明：

表 5 GRF 端口说明

信号名	方向	描述
clk	I	时钟信号
reset	I	复位信号，将 32 个寄存器中的值全部清零 1：复位

		0：无效
WriteEn	I	写使能信号 1：可向 GRF 中写入数据 0：不能向 GRF 中写入数据
Addr1[4:0]	I	5 位地址输入信号，指定 32 个寄存器中的一个，将其中存储的数据读出到 ReadData1
Addr2[4:0]	I	5 位地址输入信号，指定 32 个寄存器中的一个，将其中存储的数据读出到 ReadData2
Addr3[4:0]	I	5 位地址输入信号，指定 32 个寄存器中的一个，作为写入的目标寄存器
WriteData[31:0]	I	32 位数据输入信号
ReadData1[31:0]	O	输出 Addr1 指定的寄存器中的 32 位数据
ReadData2[31:0]	O	输出 Addr2 指定的寄存器中的 32 位数据

功能定义：

表 6 GRF 功能定义

序号	功能名称	功能描述
1	复位	reset 信号有效时，所有寄存器存储的数值清零
2	读数据	读出 Addr1,Addr2 地址对应寄存器中所存储的数据到 ReadData1,ReadData2
3	写数据	当 WriteEn 有效且时钟上升沿来临时，将 WriteData 写入 Addr3 所对应的寄存器中

4. ALU（算术逻辑单元）

端口说明：

表 7 ALU 端口说明

信号名	方向	描述
A[31:0]	I	参与 ALU 计算的第一个值
B[31:0]	I	参与 ALU 计算的第二个值
ALUOp[2:0]	I	ALU 功能的选择信号 0：ALUResult = A + B 1：ALUResult = A - B 2：ALUResult = A & B 3：ALUResult = A | B 4：未使用 5：未使用 6：未使用 7：未使用
ALUResult[31:0]	O	ALU 的计算结果
Zero	O	零标志位 1：ALUResult 为 0 0：ALUResult 不为 0

功能定义：

表 8 ALU 功能定义

序号	功能名称	功能描述
1	加运算	ALUResult = A + B
2	减运算	ALUResult = A - B
3	与运算	ALUResult = A & B
4	或运算	ALUResult = A | B
5	判断运算结果是否为零	Zero = (ALUResult == 0) ? 1 : 0

5. DM（数据存储器）

端口说明：

表 9 DM 端口说明

信号名	方向	描述
clk	I	时钟信号
reset	I	复位信号，将 DM 清零 1：复位 0：无效
WriteEn	I	写使能信号 1：可向 DM 中写入数据 0：不能向 DM 中写入数据
Addr[4:0]	I	5 位地址输入信号，指定读出或写入数据的地址
WriteData[31:0]	I	32 位数据输入信号
ReadData[31:0]	O	输出 Addr 指定的 32 位数据

功能定义：

表 10 DM 功能定义

序号	功能名称	功能描述
1	复位	当复位信号有效时，将 DM 中数据清零
2	读数据	读出 Addr 所指定的数据到 ReadData
3	写数据	当 WriteEn 有效且时钟上升沿到来时，将输入数据 WriteData 写入 Addr 所指定的地址

6. Ext（位扩展单元）

端口说明：

表 11 Ext 端口说明

信号名	方向	描述
Src[15:0]	I	16 位输入数据
ExtOp[1:0]	I	位扩展方式选择信号 0：符号扩展 1：零扩展 2：加载至高位
Dst[31:0]	O	扩展后的 32 位输出数据

功能定义：

表 12 Ext 功能定义

序号	功能名称	功能描述
1	零扩展	将 16 位输入数据进行零扩展，输出 32 位数据
2	符号扩展	将 16 位输入数据进行符号扩展，输出 32 位数据
3	加载至高位	将 16 位输入数据加载至高 16 位，并将低 16 位置 0

二、控制器设计

端口说明：

表 13 Ctrl 端口说明

信号名	方向	描述
Op[5:0]	I	输入指令的 Op 字段
Funct[5:0]	I	输入指令的 Funct 字段
nPCSrc[1:0]	O	IFU 的 nPC 端口数据源选择信号 0：来源为 PC + 4 1：来源为 PC + 4 或 PC + 4 + sign_extend(offset || 00) 2：来源为 PC[31:28] || instr_index || 00 3：来源为 GRF 的 ReadData1 端口输出
RegSrc[1:0]	O	GRF 的 WriteData 端口数据源选择信号 0：来源为 ALU 的计算结果 1：来源为 DM 的输出数据 2：来源为加载至高位的立即数 3：来源为 PC + 4
MemWrite	O	DM 写使能信号 1：可向 DM 中写入数据 0：不能向 DM 中写入数据
ALUOp[2:0]	O	ALU 功能选择信号（详见 ALU 端口说明）
ALUSrc	O	ALU 的 B 端口数据源选择信号 1：来源为立即数 0：来源为 GRF 的输出数据
ExtOp	O	位扩展方式选择信号 1：零扩展 0：符号扩展
RegDst[1:0]	O	GRF 的 Addr3 端口数据源选择信号 0：来源为 Instr[20:16] 1：来源为 Instr[15:11] 2：来源为常量 31（$ra 寄存器地址）
RegWrite	O	GRF 写使能信号 1：可向 GRF 中写入数据 0：不能向 GRF 中写入数据

功能定义：

表 14 Ctrl 真值表（1）

Funct	100001	100011							001000
Op	000000	000000	001101	100011	101011	000100	001111	000011	000000
	addu	subu	ori	lw	sw	beq	lui	jal	jr
nPCSrc[1:0]	0	0	0	0	0	1	0	2	3
RegSrc[1:0]	0	0	0	1	X	X	2	3	X
MemWrite	0	0	0	0	1	0	0	0	0
ALUOp[2:0]	0	1	3	0	0	1	X	X	X
ALUSrc	0	0	1	1	1	0	X	X	X
ExtOp[1:0]	X	X	1	0	0	0	2	X	X
RegDst[1:0]	1	1	0	0	X	X	X	2	X
RegWrite	1	1	1	1	0	0	1	1	0

表 14 Ctrl 真值表（2）

Funct	000000								
Op	000000								
	nop								
nPCSrc[1:0]	0								
RegSrc[1:0]	X								
MemWrite	0								
ALUOp[2:0]	X								
ALUSrc	X								
ExtOp	X								
RegDst[1:0]	X								
RegWrite	0								

三、数据通路

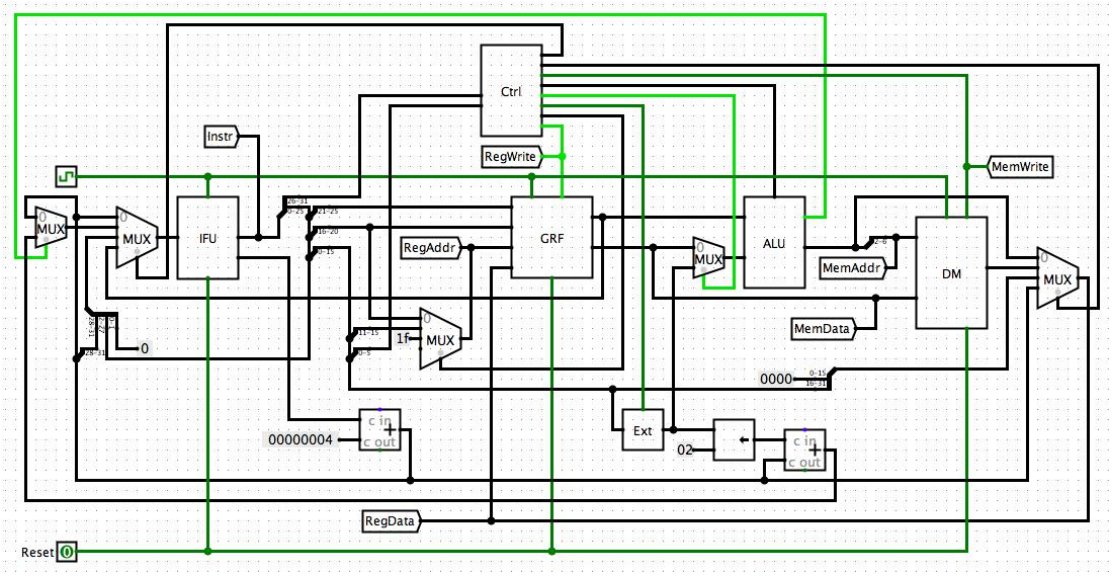

图 1 数据通路

四、测试程序

先将数据存储器中地址为 0x00000000、0x00000004、0x00000008、0x0000000c 的值分别设置为 1、2、3、4。

MIPS 汇编代码:

```
.data
one: .word 1
two: .word 2
three: .word 3
four: .word 4

.text
lw $t0, 0($zero)
lw $t1, 4($zero)
lw $t2, 8($zero)
lw $t3, 12($zero)

jal function
sw $s0, 16($zero)
sw $s1, 20($zero)
sw $s2, 24($zero)
sw $s3, 28($zero)
jal end

function:
addu $s0, $t0, $t1
subu $s1, $t3, $t0
beq $s0, $s1, label
ori $s2, $s0, 123

label:
lui $s2, 123
ori $s3, $s2, 321
nop
jr $ra

end:
```

测试期望:

GRF 中:

```
$8 = 0x00000001
$9 = 0x00000002
$10 = 0x00000003
```

```
$11 = 0x00000004
$16 = 0x00000003
$17 = 0x00000003
$18 = 0x007b0000
$19 = 0x007b0141
$31 = 0x00003028
```

DM 中：

```
Mem[0x00000000] = 0x00000001
Mem[0x00000004] = 0x00000002
Mem[0x00000008] = 0x00000003
Mem[0x0000000c] = 0x00000004
Mem[0x00000010] = 0x00000003
Mem[0x00000014] = 0x00000003
Mem[0x00000018] = 0x007b0000
Mem[0x0000001c] = 0x007b0141
```

思考题

1. 根据你的理解，在下面给出的 DM 的输入示例中，地址信号 addr 位数为什么是[11:2]而不是[9:0]？这个 addr 信号又是从哪里来的？

因为地址每次增加 4，而 DM 存储器组的下标每次增加 1，所以要舍掉地址的低两位，相当于将地址除以 4，以匹配存储器的下标。addr 信号来自 ALU 的计算结果。

2. 在相应的部件中，reset 的优先级比其他控制信号（不包括 clk 信号）都要高，且相应的设计都是同步复位。清零信号 reset 是针对哪些部件进行清零复位操作？这些部件为什么需要清零？

PC、GRF、DM。因为这些部件不是组合逻辑，具有存储记忆功能。若要 CPU 回到初始状态，则必须将这些部件中的数据清零。

3. 列举出用 Verilog 语言设计控制器的几种编码方式（至少三种），并给出代码示例。

（1）二进制码

```
always @(*) begin
   case (op)
      2'b00: out = in0;
      2'b01: out = in1;
      2'b10: out = in2;
      2'b11: out = in3;
   endcase
end
```

（2）格雷码

```
always @(*) begin
   case (op)
      2'b00: out = in0;
      2'b01: out = in1;
      2'b11: out = in2;
      2'b10: out = in3;
   endcase
end
```

（3）独热码

```
always @(*) begin
   case (op)
      4'b0001: out = in0;
      4'b0010: out = in1;
      4'b0100: out = in2;
      4'b1000: out = in3;
   endcase
end
```

4. 根据你所列举的编码方式，说明他们的优缺点。

二进制码比较直观，编码值本身就代表状态的序号。格雷码相邻状态转换时只有一个状态位发生翻转，能避免毛刺的产生，又可以降低功耗。独热码的优点在于状态比较时仅仅需要比较一位，简化了译码逻辑，因而速度较快，同时易于修改。二进制码和格雷码编码使用最少的触发器，较多的组合逻辑，而独热码反之。

5. C 语言是一种弱类型程序设计语言。C 语言中不对计算结果溢出进行处理，这意味着 C 语言要求程序员必须很清楚计算结果是否会导致溢出。因此，如果仅仅支持 C 语言，MIPS 指令的所有计算指令均可以忽略溢出。 请说明为什么在忽略溢出的前提下，addi 与 addiu 是等价的，add 与 addu 是等价的。提示：阅读《MIPS32® Architecture For Programmers Volume II: The MIPS32® Instruction Set》中相关指令的 Operation 部分。

addu 简单将两个操作数相加的结果赋给目的寄存器，而 add 先将两个操作数符号扩展一位，通过比较结果的最高两位是否相同判断溢出。由于 addu 和 add 都是将结果的 0 到 31 位赋给目的寄存器，若忽略溢出，add 的扩展操作并不影响最后赋给寄存器的值，且系统不会报出异常，所以两者是等价的。同理，addi 与 addiu 在忽略溢出的前提下也是等价的。

6. 根据自己的设计说明单周期处理器的优缺点。

优点：结构简单、清晰、直观，所需寄存器、复用器数量较少；一个周期执行一条指令，不需要考虑冲突，易维护和扩展。

缺点：部件闲置，效率低下，时钟频率受制于最慢的指令。

7. 简要说明 jal、jr 和堆栈的关系。

执行递归程序时，先将栈指针减 4 压栈，然后执行 jal，调用子程序，执行 jr 返回后栈指针加 4 弹栈。因此 jal、jr 需要和堆栈结构配合使用，相辅相成。