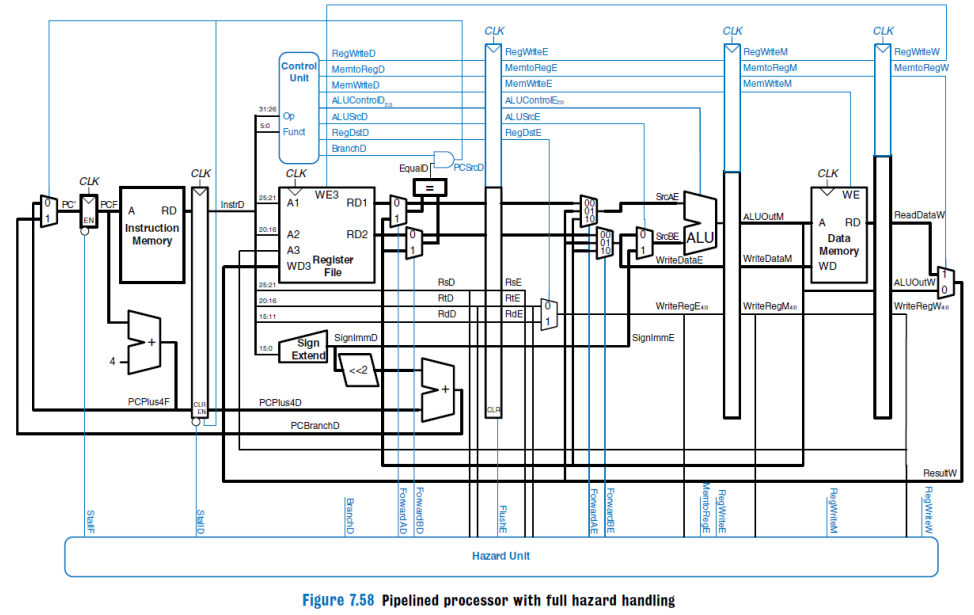
利用Verilog开发MIPS流水线处理器

一．整体结构：

流水线处理器包括流水寄存器、各级组合逻辑、各级控制器和冲突处理模块四大部分

处理器为32位处理器，支持的指令集为：**addu,subu, ori, lw, sw, beq, lui, j,jal, jr,nop**

As

数据通路、各部件的连接情况，见附件中的excel表格

二、模块设计

1. PC（程序计数器）

用于输出当前指令的PC值，和保存下一条指令的PC值。该模块由一个32位寄存器构成。

模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| NextPC[31:0] | I | 下一条指令的PC值 |
| clk | I | 时钟信号 |
| reset | I | 复位信号  1：有效  0：无效 |
| PC[31:0] | O | 当前指令PC值 |

功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 输出PC | 在clk上跳沿输出当前指令的PC值 |
| 2 | 复位 | 当复位信号有效时，PC被设置为起始地址0x00003000 |

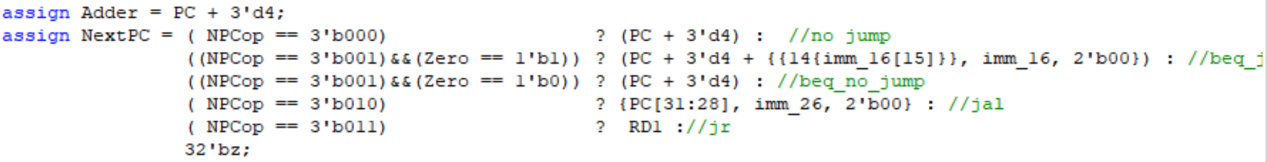
1. NextPC

用于计算下一条指令的PC值

模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| PC [31:0] | I | 当前指令PC值 |
| Imm\_16[15:0] | I | 16位立即数 |
| Imm\_26[25:0] | I | 26位立即数 |
| NPCop [2:0] | I | NPC操作指令 |
| RD1 [31:0] | I | 当前RD1 |
| zero | I | ALU计算结果为0标志  1：ALU的两个输入相等  0：ALU的两个输入不等 |
| Adder[31:0] | O | PC + 4 |
| NextPC | O | 下一条指令的PC值 |

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 输出PC | 输出下一条指令的PC值 |
| 2 | PC + 4 | 输出当前指令的PC + 4 |



1. IM（指令存储器）

IM容量为32bit×1024，实际地址宽度为10位，从而将地址的低10位（2~11位）连接到IM选择地址端口。

模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| addr[11:2] | I | 当前指令的PC |
| Opcode [5:0] | O | 6位opcode信号 |
| Func [5:0] | O | 6位Func信号 |
| rs [5:0] | O | rs寄存器编号 |
| rt [5:0] | O | rt寄存器编号 |
| rd [5:0] | O | rd寄存器编号 |
| imm\_16[15:0] | O | 16位立即数 |
| Imm\_26[25:0] | O | 26位立即数 |

功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 输出指令 | 根据当前PC值输出所对应的指令 |

1. GRF（通用寄存器组）

由32个32位寄存器构成，其中$0始终保持为0

模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| reset | I | 复位信号  1：有效  0：无效 |
| PC [31:0] | I | 当前指令的PC值 |
| RegWrite | I | 读写控制信号  1：写操作  0：读操作 |
| RA1 [4:0] | I | 读寄存器1的地址 |
| RA2 [4:0] | I | 读寄存器2的地址 |
| WA [4:0] | I | 写寄存器的地址 |
| WD [31:0] | I | 向写寄存器中写入的值 |
| RD1 [31:0] | O | RA1所对应的寄存器的值 |
| RD2 [31:0] | O | RA2所对应的寄存器的值 |

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，所有寄存器的值被设置为0 |
| 2 | 写寄存器 | 根据输入的写寄存器地址，把输入的数据写入写寄存器中 |
| 3 | 读寄存器 | 根据输入的读寄存器地址，将数据读出 |

1. ALU（算术逻辑单元）

提供32位加、减、或运算

可以不支持溢出

模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| A [31:0] | I | ALU32位输入数据A |
| B [31:0] | I | ALU32位输入数据B |
| ALUop[1:0] | I | ALU功能选择信号  00:加法  01:减法  10:或运算 |
| Result[31:0] | O | 32位计算结果 |
| Zero | O | A == B |

功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 加 | A + B |
| 2 | 减 | A - B |
| 3 | 或 | A | B |

1. DM（数据存储器）

 DM容量为32bit×1024，其起始地址为0x00003000

模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| reset | I | 复位信号  1：有效  0：无效 |
| MemWrite | I | 读写控制信号  1：写操作 |
| PC | I | 对应指令的PC值 |
| Address [31:0] | I | 所要进行操作的地址 |
| WD [31:0] | I | 写入数据的输入 |
| RD [31:0] | O | 读取数据的输出 |

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，所有数据被设置为0x00000000 |
| 2 | 写操作 | 根据输入address，把输入的数据写入 |
| 3 | 读操作 | 根据输入address，将其中的数据读出 |

1. EXT（位扩展器）

模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| Imm\_16 [15:0] | I | 16位立即数 |
| Extop[1:0] | I | 位扩展选择信号  00：无符号扩展  01：有符号扩展  10：扩展至[31:16]位，低16位补0 |
| out [31:0] | O | 位扩展后的32位输出 |

功能定义

|  |  |  |
| --- | --- | --- |
| 序号 | 功能名称 | 功能描述 |
| 1 | 高位补0 | 高16位补0 |
| 2 | 低位补0 | 低16位补0 |
| 3 | 符号扩展 | 若符号位为0，则高位补0  若符号位为1，则高位补1 |

1. 控制器（Controller）

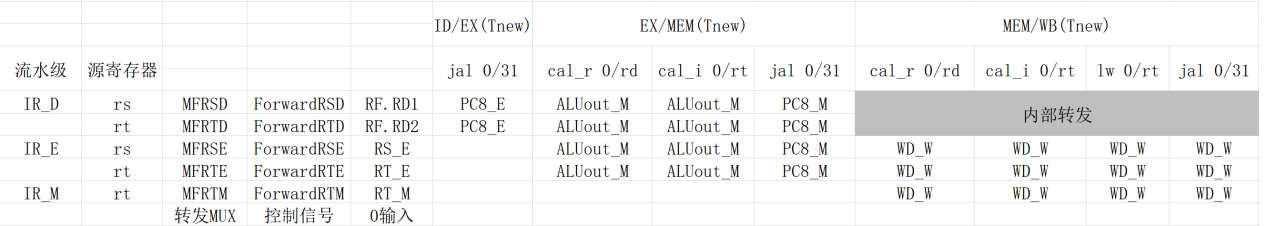
模块接口

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| Opcode[5:0] | I | Opcode |
| Func[5:0] | I | Func |
| RegDst[2:0] | O | WA输入的选择信号 |
| NPCop [2:0] | O | NPC模块的行为控制信号 |
| MemToReg[2:0] | O | WD输入的选择信号 |
| RegWrite | O | GRF写使能 |
| MemWrite | O | DM写使能 |
| ALUSrc [2:0] | O | ALU的B输入的控制信号 |
| Extop[1:0] | O | 位扩展控制信号 |
| ALUop[1:0] | O | ALU行为控制信号 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | RegDst | NPCop | MemToReg | RegWrite | MemWrite | ALUSrc | Extop | ALUop |
| addu | 000 | 000 | 000 | 1 | 0 | 000 | X | 00 |
| subu | 000 | 000 | 000 | 1 | 0 | 000 | X | 01 |
| ori | 001 | 000 | 000 | 1 | 0 | 001 | 00 | 10 |
| lw | 001 | 000 | 001 | 1 | 0 | 001 | 01 | 00 |
| sw | X | 000 | X | 0 | 1 | 001 | 01 | 00 |
| beq | X | 001 | X | 0 | 0 | 000 | 01 | X |
| lui | 001 | 000 | 000 | 1 | 0 | 001 | 10 | 00 |
| jal | 010 | 010 | 010 | 1 | 0 | X | X | X |
| jr | X | 011 | X | 0 | 0 | X | X | X |
| nop | / | / | / | / | / | / | / | / |

**三、冲突**





1. beq - lw

.space 4

.text

li $t0, 0x12345678

sw $t0, 0

lw $s0, 0 #beq与lw的冲突

beq $t0, $s0, skiplw

ori $a0, $zero, 1

ori $v0, $zero, 1

skiplw:

ori $a0, $zero, 2

lui $t1, 0x1234

addu $t2, $t1, $zero

beq $t1, $t2, skipr

ori $a1, $zero, 1

ori $v1, $zero, 1

skipr:

ori $a1, $zero, 2

@00003000: $ 1 <= 12340000

@00003004: $ 8 <= 12345678

@00003008: \*00000000 <= 12345678

@0000300c: $16 <= 12345678

@00003014: $ 4 <= 00000001

@0000301c: $ 4 <= 00000002

@00003020: $ 9 <= 12340000

@00003024: $10 <= 12340000

@0000302c: $ 5 <= 00000001

@00003034: $ 5 <= 00000002

结果与预期完全相同

1. beq - lw

li $t0, 0x12345678

sw $t0, 0

lw $zero, 0

beq $zero, $zero, skiplw

ori $a0, $zero, 1

ori $v0, $zero, 1

skiplw:

ori $a0, $zero, 2

lui $zero, 0x1234

addu $zero, $zero, $zero

beq $zero, $zero, skipr

ori $a1, $zero, 1

ori $v1, $zero, 1

skipr:

ori $a1, $zero, 2

@00003000: $ 1 <= 12340000

@00003004: $ 8 <= 12345678

@00003008: \*00000000 <= 12345678

@00003014: $ 4 <= 00000001

@0000301c: $ 4 <= 00000002

@0000302c: $ 5 <= 00000001

@00003034: $ 5 <= 00000002

1. sw - lw

ori $t4, $zero, 3

sw $t4, 0

ori $t5, $zero, 2

sw $t5, 4

lw $t2, 4

lw $t1, 0

addu $t3, $t1, $t2

@00003000: $12 <= 00000003

@00003004: \*00000000 <= 00000003

@00003008: $13 <= 00000002

@0000300c: \*00000004 <= 00000002

@00003010: $10 <= 00000002

@00003014: $ 9 <= 00000003

@00003018: $11 <= 00000005

1. sw - lw （不冲突）

li $t0, 0x12345678

sw $t0, 0

lw $zero, 0

ori $s1, $zero, 0xabcd

@00003000: $ 1 <= 12340000

@00003004: $ 8 <= 12345678

@00003008: \*00000000 <= 12345678

@00003010: $17 <= 0000abcd

1. Cali - calr （不暂停）

ori $zero, $zero, 1

addu $t1, $zero, $zero

550@00003004: $ 9 <= 00000000

1. 全覆盖

ori $2, 0x5678

lui $3, 0x1234

addu $4, $2, $3

subu $5, $4, $3 #rs\_E-cal\_r\_M

ori $11, 0x0004

lw $5, ($11) #rs\_E-cal\_i\_M

subu $2, $4, $4

nop

sw $4, ($2) #rs\_E-cal\_r\_W

lui $11, 0x0000

nop

sw $4, 4($11) #rs\_E-cal\_i\_W

lw $5, 8($0)

nop

lw $5, ($5) #rs\_E-ld\_W

ori $2, 0x5678

addu $4, $2, $3

addu $5, $3, $4 #rt\_E-cal\_r\_M

addu $5, $5, $2

nop

subu $6, $0, $5 #rt\_E-cal\_r\_W

ori $7, 0x117a

addu $8, $0, $7 #rt\_E-cal\_i\_M

lui $7, 0x6324

nop

addu $8, $2, $7 #rt\_E-cal\_i\_W

ld $9, 4($0)

nop

subu $10, $6 ,$9 #rt\_E-ld\_W

addu $11, $9, $10

sw $11, 8($0) #rt\_M-cal\_r\_W

ori $12, 0xffed

sw $12, 12($0) # rt\_M-cal\_i\_W

lw $13, 12($0)

sw $13, 16($0) #rt\_M-ld\_W

subu $13, $13, $13

nop

beq $13, $0, a #rs\_D-cal\_r\_M

nop

j end

nop

a:

lui $13, 0x0045

nop

beq $13, $0, end #rs\_D-cal\_i\_M

ori $14, 0x0004

nop

jal b

b: nop #delay branching

jr $ra

addu $ra, $ra, $14 #rs\_D-jal\_M

subu $13, $13, $13

nop

beq $0, $13, c #rt\_D-cal\_r\_M

nop

j end

nop

c:

lui $13, 0x0045

nop

beq $0, $13, end #rt\_D-cal\_i\_M

nop

lui $ra, 0x0000

jal d

nop

d: beq $ra, $0, end #rt\_D-jal\_M

nop

jal e

addu $13, $13, $2 #rs\_E-jal\_M

e:

jal f

subu $13, $4, $13 #rt\_E-jal\_M

f:

jal g

sw $ra, 16($0) #rt\_M-jal\_W

g:

jal h

nop

h: addu $13, $13, $2 #rs\_E-jal\_W

jal i

nop

i: subu $13, $4, $13 #rt\_E-jal\_W

ori $16, 0x0008

jal j

addu $ra, $ra, $16

j:nop

jr $ra #rs\_D-jal\_W / rs\_D-cal\_r\_M

nop

addu $17, $16, $16

nop

nop

beq $17, $0, end #rs\_D-cal\_r\_W

nop

lui $18, 0x1234

nop

nop

beq $18, $0, end #rs\_D-cal\_i\_W

nop

lw $19, 4($0)

nop

nop

beq $19, $0, end #rs\_D-id\_W

nop

addu $20, $17, $17

nop

nop

beq $0, $20, end #rt\_D-cal\_r\_W

nop

lui $21, 0x1234

nop

nop

beq $0, $21, end #rt\_D-cal\_i\_W

nop

lw $22, 4($0)

nop

nop

beq $0, $22, end #rt\_D-id\_W

nop

jal k

nop

k:nop

beq $0, $ra, k

end:nop

