Verilog流水线CPU设计文档

1. CPU设计方案综述

本CPU为Verilog实现的流水线CPU（32位），支持的指令集包含**{addu, subu, ori, lw, sw, beq, lui, nop，j,jal，jr}，并进行了适当的指令扩展。**

**该流水线CPU采用五级流水结构，包括流水级寄存器、主要功能部件、功能控制器、冒险控制器等，采用分布式译码方式，对指令进行流水，并在各级进行译码处理。CPU支持转发和必要的暂停，处理器顶层包含两个输入端口时钟信号clk和复位信号reset。**

1. 关键模块定义
2. stageF

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| reset | I | 复位信号 |
| NPC[31:0] | I | 输入跳转、分支指令下的下一指令地址 |
| PC\_en | I | PC使能端，高电平有效 |
| instr\_F[31:0] | O | 根据地址取到的当前指令 |
| PC\_F[31:0] | O | F级当前指令地址 |
| PC8\_F[31:0] | O | F级当前指令地址+8 |

1. regD

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| reset | I | 复位信号 |
| D\_en | I | D级流水线寄存器使能信号 |
| instr\_F[31:0] | I | F级指令 |
| PC\_F[31:0] | I | F级当前指令地址 |
| PC8\_F[31:0] | I | 当前指令地址+8 |
| instr\_D[31:0] | O | D级指令 |
| PC\_D[31:0] | O | D级当前指令地址 |
| PC8\_D[31:0] | O | D级当前指令地址+8 |

1. stageD

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| reset | I | GRF复位信号 |
| A1\_D[4:0] | I | 地址输入信号，将对应地址寄存器的值输出至RD1 |
| A2\_D[4:0] | I | 地址输入信号，将对应地址寄存器的值输出至RD2 |
| A3\_W[4:0] | I | 地址输入信号，指定要进行写入的寄存器 |
| PC\_D[31:0] | I | D级当前指令地址 |
| PC\_W[31:0] | I | W级当前指令地址 |
| imm16[15:0] | I | 16位立即数 |
| addr26[25:0] | I | 26位地址 |
| EXTOp | I | 扩展的方式 |
| RFWr | I | 写使能信号 |
| RFWD\_W[31:0] | I | 要写入寄存器的值 |
| NPCOp[1:0] | I | 控制NPC进行相应的操作：  00：当前为顺序执行指令，NPC输出PC+4  01：当前指令为beq，作为决定是否跳转的条件之一  10：当前指令为jal，NPC输出PC31..28||instr\_index||02  11：当前指令为jr，NPC输出GRF[rs] |
| MF\_RD1\_Sel[1:0] | I | RD1端口转发MUX信号 |
| MF\_RD2\_Sel[1:0] | I | RD2端口转发MUX信号 |
| RD1\_D[31:0] | O | 转发后的输出，输出A1地址对应的寄存器的值 |
| RD2\_D[31:0] | O | 转发后的输出，输出A2地址对应的寄存器的值 |
| imm32\_D[31:0] | O | 数据输出信号，输出EXT扩展后的32位立即数 |
| Next\_PC[31:0] | O | 下一指令地址 |

1. regE

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| reset | I | 复位信号 |
| E\_clr | I | E级单独的清空信号，用于暂停清空 |
| instr\_D[31:0] | I | D级指令 |
| PC\_D[31:0] | I | D级当前指令地址 |
| PC8\_D[31:0] | I | D级当前指令地址+8 |
| RD1\_D[31:0] | I | 数据输入信号，输出A1地址对应的寄存器的值 |
| RD2\_D[31:0] | I | 数据输入信号，输出A2地址对应的寄存器的值 |
| imm32\_D[31:0] | I | 数据输入信号，输出EXT扩展后的32位立即数 |
| RD1\_E[31:0] | O | 数据输出信号，输出A1地址对应的寄存器的值 |
| RD2\_E[31:0] | O | 数据输出信号，输出A2地址对应的寄存器的值 |
| imm32\_E[31:0] | O | 数据输出信号，输出EXT扩展后的32位立即数 |
| instr\_E[31 | O | E级指令 |
| PC\_E[31:0] | O | E级当前指令地址 |
| PC8\_E[31:0] | O | E级当前指令地址+8 |

1. stageE

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| RD1\_E[31:0] | I | 数据输入信号，输出A1地址对应的寄存器的值 |
| RD2\_E[31:0] | I | 数据输入信号，输出A2地址对应的寄存器的值 |
| imm32\_E[31:0] | I | 数据输入信号，输出EXT扩展后的32位立即数 |
| RFWD\_M[31:0] | I | M级转发来源 |
| RFWD\_W[31:0] | I | W级转发来源 |
| MF\_ALUA\_Sel[31:0] | I | ALU.A转发MUX控制信号 |
| MF\_ALUB\_Sel[31:0] | I | ALU.B转发MUX控制信号 |
| ALUOp[1:0] | I | ALU控制信号  00：A+B  01：A-B  10：A|B  11：B |
| Bsel | I | ALUB端口功能MUX控制信号 |
| C\_E[31:0] | O | E级ALU计算结果 |

1. regM

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| reset | I | 复位信号 |
| instr\_E[31:0] | I | E级指令 |
| PC\_E[31:0] | I | E级当前指令地址 |
| PC8\_E[31:0] | I | E级PC+8 |
| C\_E[31:0] | I | E级ALU计算结果 |
| RD2\_E[31:0] | I | E级rt寄存器的值 |
| C\_M[31:0] | O | M级ALU计算结果 |
| RD2\_M[31:0] | O | M级rt寄存器的值 |
| instr\_M[31:0] | O | M级指令 |
| PC\_M[31:0] | O | M级当前指令地址 |
| PC8\_M[31:0] | O | M级PC+8 |

1. stageM

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| reset | I | 复位信号 |
| PC\_M[31:0] | I | M级当前指令地址 |
| DMWr | I | DM写使能 |
| A\_M[31:0] | I | DM写入地址 |
| WD\_M[31:0] | I | DM写入值 |
| RFWD\_W[31:0] | I | W级转发来源 |
| MF\_DMWD\_Sel | I | DM.WD端口转发MUX控制信号 |
| WDSel\_M[1:0] | I | M级写入值的控制信号 |
| C\_M[31:0] | I | M级ALU计算结果 |
| PC8\_M[31:0] | I | M级PC+8 |
| D\_M[31:0] | O | M级DM读出值 |
| RFWD\_M[31:0] | O | M级写入值 |

1. regW

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| clk | I | 时钟信号 |
| reset | I | 复位信号 |
| instr\_M[31:0] | I | M级指令 |
| PC8\_M[31:0] | I | M级PC+8 |
| D\_M[31:0] | I | M级DM读出值 |
| C\_M[31:0] | I | M级ALU计算结果 |
| instr\_W[31:0] | O | W级指令 |
| D\_W[31:0] | O | W级DM读出值 |
| C\_W[31:0] | O | W级ALU计算结果 |
| PC8\_W[31:0] | O | W级PC+8 |

9、stageW

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| D\_W[31:0] | I | W级DM读出值 |
| C\_W[31:0] | I | W级ALU计算结果 |
| PC8\_W[31:0] | I | W级PC+8 |
| WDSel\_W[1:0] | I | W级写入值选择信号 |
| RFWD\_W[31:0] | O | W级写入值 |

10、controller

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| instr[31:0] | I | 指令 |
| NPCOp[1:0] | O | NPC控制信号 |
| RFWr | O | GRF使能信号 |
| EXTOp | O | EXT控制信号 |
| ALUOp[1:0] | O | ALU控制信号 |
| DMWr | O | DM使能信号 |
| A3Sel[1:0] | O | A3选择信号 |
| WDSel[1:0] | O | WD端口选择信号 |
| BSel | O | B选择信号 |
| imm16[15:0] | O | 16位立即数 |
| addr26[25:0] | O | 26位立即数 |
| rs[4:0] | O | rs寄存器地址 |
| rt[4:0] | O | rt寄存器地址 |
| rd[4:0] | O | rd寄存器地址 |
| A3[4:0] | O | 当前GRF写入地址 |
| Tuse\_rs[1:0] | O | D级rs的Tuse |
| Tuse\_rt[1:0] | O | D级rt的Tuse |
| Tnew\_E[1:0] | O | rs或rt的Tnew\_E |
| Tnew\_M[1:0] | O | rs或rt的Tnew\_M |

11、ATcontroller

|  |  |  |
| --- | --- | --- |
| 信号名 | 方向 | 描述 |
| rs\_D[4:0] | I | D级rs地址 |
| rt\_D[4:0] | I | D级rt地址 |
| rs\_E[4:0] | I | E级rs地址 |
| rt\_E[4:0] | I | E级rt地址 |
| rt\_M[4:0] | I | M级rt地址 |
| A3\_E[4:0] | I | E级A3地址 |
| A3\_M[4:0] | I | M级A3地址 |
| A3\_W[4:0] | I | M级A3地址 |
| Tuse\_rs[1:0] | I | D级rs的Tuse |
| Tuse\_rt[1:0] | I | D级rt的Tuse |
| Tnew\_E[1:0] | I | rs或rtE级Tnew |
| Tnew\_M[1:0] | I | rs或rt的Tnew |
| MF\_RD1\_Sel[1:0] | O | CMPA端口的转发MUX的选择信号 |
| MF\_RD2\_Sel[1:0] | O | CMPB端口的转发MUX的选择信号 |
| MF\_ALUA\_Sel[1:0] | O | ALUA端口的转发MUX的选择信号 |
| MF\_ALUB\_Sel[1:0] | O | ALUB端口的转发MUX的选择信号 |
| MF\_DMWD\_Sel | O | DMWD端口的转发MUX的选择信号 |
| PC\_en | O | PC使能信号 |
| D\_en | O | D级使能 |
| E\_clr | O | E级流水线寄存器清除信号 |

各级流水寄存器储存值如下

|  |  |  |  |
| --- | --- | --- | --- |
| D | E | M | W |
| instr\_D | instr\_E | instr\_M | instr\_W |
|  | RD1\_D | C\_M | C\_W |
|  | RD2\_D | RD2\_M | D\_W |
|  | imm32\_D |  |  |
|  |  |  |  |
| PC\_D | PC\_E | PC\_M |  |
| PC8\_D | PC8\_E | PC8\_M | PC8\_W |
|  | (A3\_E) | (A3\_M) | (A3\_W) |
|  | (WDSel\_E) | (WDSel\_M) | (WDSel\_W) |

1. 转发与暂停处理

转发

转发点共有五个：CMP.A,CMP.B,ALU.A,ALU.B,DM.WD

最后一级W级对GRF采用内部转发

对其他为RFWD\_W:C\_W,D\_W,PC8\_W

M级为RFWD\_M：PC8\_M,C\_E

E级为PC8\_E

|  |  |  |  |
| --- | --- | --- | --- |
| 转发点 | 0 | 1 | 2 |
| GRF.RD1 | RD1\_D\_raw | PC8\_E | RFWD\_M |
| GRF.RD2 | RD2\_D\_raw | PC8\_E | RFWD\_M |
| ALU.A | RD1\_E | RFWD\_M | RFWD\_W |
| ALU.B | RD2\_E | RFWD\_M | RFWD\_W |
| DM.WD | RD2\_M | RFWD\_W |  |

转发的判断条件是：当前使用的寄存器地址与转发来源寄存器地址相同，且不为0，则按照优先等级进行转发。

如果转发来源不进行写入的话，令转发来源寄存器地址为0（从而实现不转发）。

暂停

在D级进行Tuse和各级Tnew的比较如果Tuse<Tnew，则进行暂停：

PC\_en无效，保持原值不变，D级流水线寄存器使能信号无效，保持原值不变。E\_clr有效，清除原流水线寄存器的值，实现暂停。

T矩阵：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 指令 | Tuse\_rs | Tuse\_rt | Tnew\_E | Tnew\_M |
| addu | 01 | 01 | 01 | 00 |
| subu | 01 | 01 | 01 | 00 |
| ori | 01 | 11 | 01 | 00 |
| lui | 11 | 11 | 01 | 00 |
| lw | 01 | 11 | 10 | 01 |
| sw | 01 | 10 | 00 | 00 |
| beq | 00 | 00 | 00 | 00 |
| jal | 11 | 11 | 00 | 00 |
| j | 11 | 11 | 00 | 00 |
| jr | 00 | 11 | 00 | 00 |
| nop | 不处理即可 |  |  |  |

1. 控制信号取值表

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 指令 | NPCOp[1:0] | RFWr | EXTOp | ALUOp[1:0] | DMWr | A3Sel[1:0] | WDSel[1:0] | BSel |
| addu  (000000/100001) | 00 | 1 | x | 00 | 0 | 00 | 00 | 0 |
| subu  (000000/100011) | 00 | 1 | x | 01 | 0 | 00 | 00 | 0 |
| ori  (001101) | 00 | 1 | 0 | 10 | 0 | 01 | 00 | 1 |
| lui  (001111) | 00 | 1 | 0 | 11 | 0 | 01 | 00 | 1 |
| lw  (100011) | 00 | 1 | 1 | 00 | 0 | 01 | 01 | 1 |
| sw  (101011) | 00 | 0 | 1 | 00 | 1 | xx | xx | 1 |
| beq  (000100) | 01 | 0 | x | 01 | 0 | xx | xx | 0 |
| jal  (000011) | 10 | 1 | x | xx | 0 | 10 | 10 | x |
| j  （000010） | 10 | 0 | x | xx | 0 | xx | xx | x |
| jr  (000000/001000) | 11 | 0 | x | xx | 0 | xx | xx | x |

1. 测试方案

ori $1,$1,1

ori $2,$2,4

subu $3,$1,$2

addu $1,$3,$2

lui $1,1234

sw $1,0($zero)

lw $4,0($zero)

addu $4,$4,$1

ori $5,$5,5

ori $6,$6,5

beq $5,$6,beqt1

addu $2,$2,$1

subu $2,$2,$1

beqt1:

addu $5,$5,$1

beq $5,$6,beqt2

addu $2,$2,$1

subu $2,$2,$1

beqt2:

j jt

addu $5,$5,$1

subu $5,$5,$1

jt:

jal jalt

addu $5,$5,$1

subu $5,$5,$1

addu $5,$5,$1

subu $5,$5,$1

jalt:

ori $7,$7,4

addu $31,$31,$7

sw $31,0($zero)

lw $31,0($zero)

jr $ra

addu $1,$1,$1

预期结果见mars：

结果：

55@00003000: $ 1 <= 00000001

65@00003004: $ 2 <= 00000004

75@00003008: $ 3 <= fffffffd

85@0000300c: $ 1 <= 00000001

95@00003010: $ 1 <= 04d20000

95@00003014: \*00000000 <= 04d20000

115@00003018: $ 4 <= 04d20000

135@0000301c: $ 4 <= 09a40000

145@00003020: $ 5 <= 00000005

155@00003024: $ 6 <= 00000005

185@0000302c: $ 2 <= 04d20004

195@00003034: $ 5 <= 04d20005

225@0000303c: $ 2 <= 09a40004

235@00003040: $ 2 <= 04d20004

255@00003048: $ 5 <= 09a40005

265@00003050: $31 <= 00003058

275@00003054: $ 5 <= 0e760005

285@00003064: $ 7 <= 00000004

295@00003068: $31 <= 0000305c

295@0000306c: \*00000000 <= 0000305c

315@00003070: $31 <= 0000305c

355@00003078: $ 1 <= 09a40000

365@0000305c: $ 5 <= 181a0005

375@00003060: $ 5 <= 0e760005

385@00003064: $ 7 <= 00000004

395@00003068: $31 <= 00003060

395@0000306c: \*00000000 <= 00003060

415@00003070: $31 <= 00003060

455@00003078: $ 1 <= 13480000

465@00003060: $ 5 <= fb2e0005

475@00003064: $ 7 <= 00000004

485@00003068: $31 <= 00003064

485@0000306c: \*00000000 <= 00003064

505@00003070: $31 <= 00003064

545@00003078: $ 1 <= 26900000

555@00003064: $ 7 <= 00000004

565@00003068: $31 <= 00003068

565@0000306c: \*00000000 <= 00003068

585@00003070: $31 <= 00003068

625@00003078: $ 1 <= 4d200000

635@00003068: $31 <= 0000306c

635@0000306c: \*00000000 <= 0000306c

655@00003070: $31 <= 0000306c

695@00003078: $ 1 <= 9a400000

695@0000306c: \*00000000 <= 0000306c

715@00003070: $31 <= 0000306c

755@00003078: $ 1 <= 34800000

755@0000306c: \*00000000 <= 0000306c

775@00003070: $31 <= 0000306c

815@00003078: $ 1 <= 69000000

815@0000306c: \*00000000 <= 0000306c

835@00003070: $31 <= 0000306c

875@00003078: $ 1 <= d2000000

875@0000306c: \*00000000 <= 0000306c

895@00003070: $31 <= 0000306c

935@00003078: $ 1 <= a4000000

935@0000306c: \*00000000 <= 0000306c

955@00003070: $31 <= 0000306c

995@00003078: $ 1 <= 48000000

995@0000306c: \*00000000 <= 0000306c

1. 思考题
2. 在采用本节所述的控制冒险处理方式下，PC的值应当如何被更新？请从数据通路和控制信号两方面进行说明。

PC根据NPCOp进行更新，该控制信号来源于D级。

顺序执行的指令PC更新为PC\_F+4；分支类需要进行判断，如果条件满足更新为PC\_D+4+offset，否则更新为PC\_F+4；对于j/jal 按照绝对地址进行更新；对于jr按照rs寄存器中的值进行更新。

1. 对于jal等需要将指令地址写入寄存器的指令，为什么需要回写PC+8？

实验支持延迟槽，会执行jal的下一条指令，应该将下下条指令的地址存入rs寄存器。

1. 为什么所有的供给者都是存储了上一级传来的各种数据的**流水级寄存器**，而不是由ALU或者DM等部件来提供数据？

如果由ALU或者DM等部件提供数据进行转发，相比流水级寄存器提供数据进行转发，要经过的组合逻辑更多，需要的时间更多。

1. **Thinking 1**：如果不采用已经转发过的数据，而采用上一级中的原始数据，会出现怎样的问题？试列举指令序列说明这个问题。

会产生数据冒险。

addu $1,$1,$2

subu $3,$1,$2

addu到M级的时候，subu在E级，但此时addu还没有对$1进行写入操作，如果采用原始数据，$1就不是最新值，于是产生了数据冒险。

**Thinking 2**：我们为什么要对GPR采用内部转发机制？如果不采用内部转发机制，我们要怎样才能解决这种情况下的转发需求呢？

确保读出的数是W级最新写入的数，避免数据冒险。

把要写入的数据与读出的数据接入MUX，根据两寄存器的号数关系进行选择（0号进行特判），相等就选择写入值，反之原值。

**Thinking 3**：为什么0号寄存器需要特殊处理？

如果当前读0号寄存器，之前写入0号寄存器，不对0号进行特判就会产生从0号寄存器中读出非零数的现象。

**Thinking 4**：什么是“最新产生的数据”？

距离该条指令最近的指令对GRF写入的数据为最新产生的数据。

1. 在AT方法讨论转发条件的时候，只提到了“供给者需求者的A相同，且不为0”，但在CPU写入GRF的时候，是有一个we信号来控制是否要写入的。为何在AT方法中不需要特判we呢？为了**用且仅用**A和T完成转发，在翻译出A的时候，要结合we做什么操作呢？

如果WE无效，则将A置为0，从而转发MUX中选择前一级流水线寄存器中的值，完成了WE无效的情况下不转发的行为。

1. 在本实验中你遇到了哪些不同指令类型组合产生的冲突？你又是如何解决的？相应的测试样例是什么样的？

写寄存器的指令：addu，subu,ori,lui,lw,jal

读寄存器的指令:addu,subu,ori,lw,sw,beq,jr

以上两种指令类型相互间会发生冲突，通过暂停和转发解决

在D级比较Tuse和Tnew的关系，如果Tuse<Tnew\_? 且读出寄存器地址和写入寄存器地址相同且不为0则暂停，PC\_en,D\_en无效,E\_clr有效。（如果不写入则将写入地址置为0，转发MUX选择为原值，相当于没转发）

否则可以通过转发解决，如果读取地址和写入地址相同，则按“近”转发，使用、传递最新值。

测试样例见上面代码。

1. 如果你是手动构造的样例，请说明构造策略，说明你的测试程序如何保证**覆盖**了所有需要测试的情况；如果你是**完全随机**生成的测试样例，请思考完全随机的测试程序有何不足之处；如果你在生成测试样例时采用了**特殊的策略，**比如构造连续数据冒险序列，请你描述一下你使用的策略如何**结合了随机性**达到强测的效果。

手动测试样例。

写寄存器的指令：addu，subu,ori,lui,lw,jal

读寄存器的指令:addu,subu,ori,lw,sw,beq,jr

不读不写的指令：nop（不用管）

只有可能在这两类之间发生冲突，并且要求读取和写入寄存器的地址相同，共有6\*7 种组合，即可覆盖。