Verilog 多周期 CPU 设计文档

作者：李健健

一、 CPU 设计方案综述

（一）总体设计概述

使用 Verilog 开发一个简单的多周期 CPU，总体概述如下：

1. 此 CPU 为 32 位 CPU
2. 此 CPU 为多周期设计
3. 此 CPU 支持的指令集为：

   {addu, subu, ori, lw, sw, beq, lui, jal, jr,nop}
4. nop 机器码为 0x00000000
5. addu, subu 不支持溢出

（二）关键模块定义

1. PC

（1） 端口说明

表 1-PC 端口说明

序号	信号名	方向	描述
1	clk	I	时钟信号
2	reset	I	复位信号
3	NPC	I	下一条指令所在 IM 地址
4	PC	O	当前指令所在 IM 地址

（2） 功能定义

表 2-PC 功能定义

序号	功能	描述
1	存储指令的地址	保存当前执行指令在 IM 中的地址

2. IM

（1） 端口说明

表 3-IM 端口说明

序号	信号名	方向	描述
1	PC[31:0]	I	时钟信号
2	instr[31:0]	O	指令

（2） 功能定义

表 4-IM 功能定义

序号	功能	描述
1	取指令	就是取指令

3. IFID

（1） 端口说明

表 5-IDIF 功能定义

序号	信号名	方向	描述
1	clk	I	时钟信号
2	en	I	使能信号
3	reset	I	同步复位信号
4	PCF[31:0]	I	PC 在 F 级的值
5	InstrF[31:0]	I	instr 在 F 级的值
6	PCD[31:0]	O	PC 在 D 级的值
7	InstrD[31:0]	O	instr 在 D 级的值

（2） 功能定义

表 6-IFID 功能定义

序号	功能	描述
1	存储流水线值	存储流水线值

4. NPC

（1） 端口说明

表 7-NPC 端口说明

序号	信号名	方向	描述
1	branch	I	分支信号
2	JType	I	跳转信号
3	JReg	I	判断指令是否需要跳转寄存器
4	PCF[31:0]	I	F 级 PC 值
5	PCD[31:0]	I	D 级 PC 值
6	RegJump[31:0]	I	跳转寄存器中地址值
7	imm26D[25:0]	I	D 级的 26 位立即数
8	NPC[31:0]	O	根据各种指令计算出的下一个 PC 值

（2） 功能定义

表 8-NPC 功能定义

序号	功能	描述
1	计算下一个 PC 的值	

5. GRF

（1） 端口说明

表 9-GRF 端口说明

序号	信号名	方向	描述
1	clk	I	时钟信号
2	reset	I	同步复位信号，将 32 个寄存器中全部清零 1：清零 0：无效
3	WE	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
4	A1[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 RD1
5	A2[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 RD2
6	A3[4:0]	I	5 位地址输入信号，指定 32 个寄存器中的一个，作为 RD 的写入地址
7	WD[31:0]	I	32 位写入数据
8	RD1[31:0]	O	输出 A1 指定的寄存器的 32 位数据
9	RD2[31:0]	O	输出 A2 指定的寄存器的 32 位数据

（2） 功能定义

表 10-GRF 功能定义

序号	功能	描述
1	异步复位	reset 为 1 时，将所有寄存器清零
2	读数据	将 A1 和 A2 地址对应的寄存器的值分别通过 RD1 和 RD2 读出
3	写数据	当 WE 为 1 且时钟上升沿来临时，将 WD 写入到 A3 对应的寄存器内部

6. CMP

（1） 端口说明

表 11-CMP 端口说明

序号	信号名	方向	描述
1	A[31:0]	I	操作数 A
2	B[31:0]	I	操作数 B
3	eq	O	A==B?
4	eqz	O	A==0?
5	ltz	O	A<0?

（2） 功能描述

表 12-CMP 功能定义

序号	功能	描述

1	判断 A 和 B 是否相等	若 A 等于 B，eq 置一，否则置零
2	判断 A 是否等于 0	若 A 等于 0，eqz 置一，否则置零
3	判断 A 是否小于 0	若 A 小于 0，ltz 置一，否则置零

7. EXT

(1)端口说明

表 13-EXT 端口说明

序号	信号名	方向	描述
1	imm16[15:0]	I	代扩展的 16 位信号
2	sign	I	无符号或符号扩展选择信号 0：无符号扩展 1：符号扩展
3	imm32[31:0]	O	扩展后的 32 位的信号

(2)功能定义

表 14-EXT 功能定义

序号	功能	描述
1	无符号扩展	当 sign 为 0 时，将 imm16 无符号扩展输出
2	符号扩展	当 sign 为 1 时，将 imm16 符号扩展输出

8. IDEX

(1)功能定义

表 15-IDEX 功能定义

序号	功能	描述
1	流水线寄存器	保留 ID/EX 级流水线信息

9. ALU

(1)端口说明

表 16-ALU 端口说明

序号	信号名	方向	描述
1	A[31:0]	I	参与运算的第一个数
2	B[31:0]	I	参与运算的第二个数
3	ALUOp[2:0]	I	决定 ALU 做何种操作 0000：无符号加 0001：无符号减 0010：与 0011：或 0100：将 B[15:0]做为 res[31:16],res[15:0]=0
5	res[31:0]	O	A 与 B 做运算后的结果

(2)功能定义

表 17-ALU 功能定义

序号	功能	描述
1	加运算	res = A + B
2	减运算	res = A - B

3	与运算	res = A & B
4	或运算	res = A | B
5	加载高位运算	res = {B[15:0], 16'h0}

10. EXMEM

（1） 功能定义

表 18-EXMEM 功能定义

序号	功能	描述
1	流水线寄存器	保留 EX/MEM 级流水线信息

11. DM

(1)端口说明

表 11-DM 端口说明

序号	信号名	方向	描述
1	clk	I	时钟信号
2	reset	I	异步复位信号 0：无效 1：内存值全部清零
3	WE	I	写使能信号 0：禁止写入 1：允许写入
4	witdth	I	读写位宽
5	LoadSign	I	读写时是否带符号
6	addr[31:0]	I	读取或写入信号地址
7	WD[31:0]	I	写入 Mem 的值
6	RD[31:0]	O	32 位读出数据

(2)功能定义

表 12-DM 功能定义

序号	功能	描述
1	异步复位	当 reset 为 1 时，DM 中所有数据清零
2	写入数据	当 WE 有效时，时钟上升沿来临时，WD 中数据写入 A 对应的 DM 地址中
3	读出数据	RD 永远读出 A 对应的 DM 地址中的值

12. Controller

(1)端口说明

表 15-Controller 端口说明

序号	信号名	方向	描述
1	instr[31:0]	I	instr[31:26],6 位控制信号
2	eq	I	RegRead1 和 RegRead2 是否相等

3	sign	O	Ext 是否进行符号扩展
4	branch	O	是否进行分支
5	JType	O	J 型信号
6	JReg	O	是否读取寄存器值作为 NPC
7	WDSelD	O	若 D 级生成写入寄存器的值的选择信号。
8	A3DE	O	写入寄存器的地址
9	ALUOp	O	ALU 怎么算
10	ALUASel	O	ALU 第一个操作数的选择信号
11	ALUBSel	O	ALU 第二个操作数的选择信号
12	WDSelE	O	E 级写入寄存器的值的选择信号
13	WEMem	O	DM 写使能信号 0：禁止写入 1：允许写入
14	width	O	写入 Mem 的位宽
15	LoadSign	O	是否 load 带符号的值
16	WDSelM	O	M 级生成写入寄存器的值的选择信号
17	D1Use	O	D 级是否读取 rs
18	D2Use	O	D 级是否读取 rt
19	E1Use	O	E 级是否读取 rs
20	D2Use	O	E 级是否读取 rt
21	M2Use	O	M 级是否读取 rt

(2)真值表

表 16-Controller 内部真值对应

端口	addu	subu	ori	lw	sw	lui	beq	jal	jr
op	000000	000000	001101	100011	101011	001111	000100	000011	000000
func	100001	100011							001000
WEMem	0	0	0	0	1	0	0	0	0
ALUBSel	0	0	1	1	1	1	0	0	0
ALUOp	000	001	011	000	000	100	000	000	000
sign	0	0	0	1	1	0	1	0	0
branch	0	0	0	0	0	0	1	0	0

JType	0	0	0	0	0	0	0	1	0
JReg	0	0	0	0	0	0	0	0	1

二、 转发暂停控制

不同于教程所采用的方法，我使用的是一种方法。面对不同的情况，分别将指令暂停在 D 级或 E 级。
转发和暂停之间有优先级，
暂停在 D 级和 E 级有优先级。
每一级都将要写入到寄存器的数据流水，当一级要使用的数据冲突时，发现要写入的数据为高阻态，则暂停。

序号	信号名	方向	描述
1	clk	I	时钟信号
2	reset	I	重置信号
3	A1D[4:0]	I	D 级读取 rs 的序号
4	A2D[4:0]	I	D 级读取 rt 的序号
5	RD1D[31:0]	I	D 级读取 rs 的值
6	RD2D[31:0]	I	D 级读取 rt 的值
7	D1Use	I	D 级是否在使用 rs
8	D2Use	I	D 级是否在使用 rt
9	A1E[4:0]	I	E 级读取 rs 的序号
10	A2E[4:0]	I	E 级读取 rt 的序号
11	RD1E[31:0]	I	E 级读取 rs 的值
12	RD2E[31:0]	I	E 级读取 rt 的值
13	E1Use	I	E 级是否使用 rs
14	E2Use	I	E 级是否使用 rt
15	A3E[4:0]	I	写入寄存器的序号
16	WDE[31:0]	I	E 级写入寄存器的值
17	A2M[4:0]	I	M 级读取 rt 的序号
18	RD2M[31:0]	I	M 级读取到寄存器的值
19	M2Use	I	M 级是否在使用 rt
20	A3M[4:0]	I	M 级写入寄存器的信号
21	WDM[31:0]	I	M 级写入寄存器的值
22	A3W[4:0]	I	W 级写入寄存器的序号
23	WDW[31:0]	I	W 级写入寄存器的值
24	FwdD1[31:0]	O	转发给 D 级的值
25	FwdD2[31:0]	O	转发给 D 级的值
26	FwdE1[31:0]	O	转发给 E 级的值
27	FwdE2[31:0]	O	转发给 E 级的值
28	FwdM2[31:0]	O	转发给 M 级的值
29	EnPC	O	PC 的使能信号
30	EnIFID	O	IFID 的使能信号
31	EnIDEX	O	IDEX 的使能信号
32	FlushIDEX	O	IDEX 的清零信号

33	FlushEXMEM	O	EXMEM 的清零信号

三、 测试方案

（1）测试代码：

```
.text
   addi      $t1, $0, 4
   addi      $t0, $0, 4
   sw     $t0, 0($0)
   addi      $t0, $0, 8
   sw     $t0, 4($0)
   addi      $t0, $0, 12
   sw     $t0, 8($0)

   lw     $t0, 4($0)
   addu      $t0, $t0, $t1

   lw     $t0, 4($t0)
   ori    $t0, 8

   lw     $t0, 12($v0)
   sw     $t0, 4($0)

   lw     $t0, 12($v0)
   sw     $t0, 0($t0)

   addi      $t0, $v0, 4
   beq    $t0, $t0, Tag1
   addi      $s1, $v0, 256


Tag1:
   lui    $t0, 100
   beq    $t0, $t0, Tag2
   addi      $s1, $v0, 256

Tag2:
   ori    $t0, 100
   beq    $t0, $t0, Tag3
   addi      $s1, $v0, 256

Tag3:
   lw     $t0, 4($v0)
```

```
    beq    $t0, $t0, Tag4
    addi       $s1, $v0, 256

Tag4:
    jal    Tag5

Tag5:
    addi       $31, $0, 4
    jr     $31
```

（2）MARS 中运行结果

```
@00003000: $ 9 <= 00000004
@00003004: $ 8 <= 00000004
@00003008: *00000000 <= 00000004
@0000300c: $ 8 <= 00000008
@00003010: *00000004 <= 00000008
@00003014: $ 8 <= 0000000c
@00003018: *00000008 <= 0000000c
@0000301c: $ 8 <= 00000008
@00003020: $ 8 <= 0000000c
@00003024: $ 8 <= 00000000
@00003028: $ 8 <= 00000008
@0000302c: $ 8 <= 00000000
@00003030: *00000004 <= 00000000
@00003034: $ 8 <= 00000000
@00003038: *00000000 <= 00000000
@0000303c: $ 8 <= 00000004
@00003044: $17 <= 00000100
@00003048: $ 8 <= 00640000
@00003050: $17 <= 00000100
@00003054: $ 8 <= 00640064
@0000305c: $17 <= 00000100
@00003060: $ 8 <= 00000000
@00003068: $17 <= 00000100
@0000306c: $31 <= 00003074
@00003070: $31 <= 00000004
@00003070: $31 <= 00000004
```

（3）该 CPU 运行输出结果

55@00003000: $ 9 <= 00000004
65@00003004: $ 8 <= 00000004
65@00003008: *00000000 <= 00000004
85@0000300c: $ 8 <= 00000008

85@00003010: *00000004 <= 00000008
105@00003014: $ 8 <= 0000000c
105@00003018: *00000008 <= 0000000c
125@0000301c: $ 8 <= 00000008
145@00003020: $ 8 <= 0000000c
155@00003024: $ 8 <= 00000000
175@00003028: $ 8 <= 00000008
185@0000302c: $ 8 <= 00000000
185@00003030: *00000004 <= 00000000
205@00003034: $ 8 <= 00000000
215@00003038: *00000000 <= 00000000
235@0000303c: $ 8 <= 00000004
265@00003044: $17 <= 00000100
275@00003048: $ 8 <= 00640000
305@00003050: $17 <= 00000100
315@00003054: $ 8 <= 00640064
345@0000305c: $17 <= 00000100
355@00003060: $ 8 <= 00000000
395@00003068: $17 <= 00000100
405@0000306c: $31 <= 00003074
415@00003070: $31 <= 00000004
425@00003070: $31 <= 00000004

四、 思考题

（一） 流水线冒险

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。

```
module NPC(
    input branch,
    input JType,
    input JReg,
    input [31:0] PCF,
    input [31:0] PCD,
    input [25:0] imm26D,
    input [31:0] RegJump,
    output [31:0] NPC
    );
    wire [31:0] PCD4;
    assign PCD4 = PCD + 4;
    assign NPC =    (branch)? PCD + 4 + {{14{imm26D[15]}}, imm26D[15:0], 2'b0}:
                    (JType)? {PCD4[31:28], imm26D, 2'b0}:
                    (JReg)? RegJump:
                    PCF + 4;        // default the PC + 4


endmodule
```

数据通路：

PC：PC+4、branch 型指令、JumpRegister 指令和 J 型指令。

控制信号：

controller 解码出应当选择的 NPC 值即可。

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？

含有延迟槽，后面有 nop。

（二） 数据冒险的分析

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？

转发的目的：若算出来，才转发，否则暂停。这是使用“暂停-转发”策略下的核心思想。

直接转发的话，“Brutal-Forward”，也不是不行，但是一来课程组禁止这样，二来转发控制会变得复杂。

（三） AT 法处理流水线数据冒险

“转发（旁路）机制的构造”中的 Thinking 1-4

就是转发的优先级。以 jr 为例，在 D 级就要产生写入的 NPC 数据，此时若 E 级和 M 级要求写入同一个寄存器，导致数据冲突，则需要选择 E 级流水线的数据。

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？

如果不需要写寄存器，直接将 A 译码为为 0，这样甚至可以直接省略 we。

（四） 在线测试相关说明

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？

见测试样例和暂停转发表

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。

1. 将指令分类
2. 对分类后的指令进行枚举
3. 使用同学测评机