Verilog 流水线 CPU 设计文档

一、 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

{LB、LBU、LH、LHU、LW、SB、SH、SW、ADD、ADDU、SUB、SUBU、MULT、MULTU、DIV、DIVU、SLL、SRL、SRA、SLLV、SRLV、SRAV、AND、OR、XOR、NOR、ADDI、ADDIU、ANDI、ORI、XORI、LUI、SLT、SLTI、SLTIU、SLTU、BEQ、BNE、BLEZ、BGTZ、BLTZ、BGEZ、J、JAL、JALR、JR、MFHI、MFLO、MTHI、MTLO、MFC0、MTC0, ERET}。

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，处理器顶层包含三个输入端口时钟信号 clk 和复位信号 reset，和外部中断信号 interrupt，和一个输出端口宏观 PCaddr。

内部包含 CPU 模块 Bridge 模块和 Timer0 和 Timer1 模块。CPU 内部增加了 CP0 协处理器。支持异常和中断的处理。

二、 关键模块定义

①Timer

信号名	方向	描述
clk	I	时钟信号
reset	I	复位信号
Addr[31:2]	I	写入/读取寄存器的地址
WE	I	写使能
Din[31:0]	I	写入数据
Dout[31:0]	O	输出数据
IRQ	O	中断信号

②Bridge

信号名	方向	描述
PrAddr[31:2]	I	CPU 向 DEV 读取或写入的地址
PrWD[31:0]	I	CPU 向 DEV 写入的数据
PrWe	I	CPU 对 DEV 的写使能
DEV0_RD[31:0]	I	从 DEV0 读出的数据
DEV1_RD[31:0]	I	从 DEV1 读出的数据
DEV_Addr[31:2]	O	CPU 向 DEV 读取或写入的地址

DEV_WD[31:0]	O	CPU 向 DEV 写入的数据
DEV0_WE	O	DEV0 写使能
DEV1_WE	O	DEV1 写使能
PrRD[31:0]	O	DEV 向 CPU 的写入数据

③mips

信号名	方向	描述
clk	I	时钟信号
reset	I	复位信号
interrupt	I	外部中断信号
addr	O	宏观 PC

④CP0 协处理器接口：

信号名	方向	描述
clk	I	时钟信号
reset	I	复位信号
A[4:0]	I	读\写 CP0 寄存器的地址
Din[31:0]	I	写入 CP0 寄存器的数据

PC_Int[31:2]	I	中断或异常发生级的 PC 值
ExcCode[6:2]	I	内部异常信号
HWInt[15:10]	I	外部中断信号
We	I	CP0 写使能
eret	I	当前指令为 eret
BD	I	跳转分支指令后紧接指令异常中断标记位
Dout[31:0]	O	读 CP0 寄存器的数据
EPC[31:0]	O	eret 返回的 PC 地址
IntReq	O	中断（包括异常）信号

⑤CPU

信号名	方向	描述
clk	I	时钟信号
reset	I	复位信号
PrRD[31:0]	I	写入 GRF 的数据
HWInt[15:10]	I	中断位
PrWE	O	向 timer 的写使能
PrWD[31:0]	O	写入 timer 的数据

PrAddr[31:2]	0	写入 timer 的地址
addr[31:0]	0	宏观 PC

三、 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(不用为 3)	Tnew_E（不用为 0)	Tnew_M
addu	01	01	01	00
add	01	01	01	00
subu	01	01	01	00
sub	01	01	01	00
and	01	01	01	00
or	01	01	01	00
xor	01	01	01	00

nor	01	01	01	00
slt	01	01	01	00
sltu	01	01	01	00
sll	11	01	01	00
sllv	01	01	01	00
srl	11	01	01	00
srlv	01	01	01	00
sra	11	01	01	00
srav	01	01	01	00
mfhi	11	11	01	00
mflo	11	11	01	00
mthi	01	01	00	00
mtlo	01	01	00	00
mult	01	01	00	00
multu	01	01	00	00
div	01	01	00	00
divu	01	01	00	00

addi	01	11	01	00
addiu	01	11	01	00
ori	01	11	01	00
andi	01	11	01	00
xori	01	11	01	00
slti	01	11	01	00
sltiu	01	11	01	00
lui	11	11	01	00
lw	01	11	10	01
lh	01	11	10	01
lhu	01	11	10	01
lb	01	11	10	01
lbu	01	11	10	01
sw	01	10	00	00
sh	01	10	00	00
sb	01	10	00	00
beq	00	00	00	00

bne	00	00	00	00
blez	00	00	00	00
bltz	00	00	00	00
bgez	00	00	00	00
bgtz	00	00	00	00
jal	11	11	00	00
jalr	00	11	00	00
j	11	11	00	00
jr	00	11	00	00
mfc0	11	11	10	01
mtc0	11	10	00	00
eret	11	11	00	00
nop	不处理即可			

四、 控制信号取值表

溢出指令类型	CalType[1:0]

add,sub,addi	01
lw,lh,lhu,lb,lbu	10
sw,sh,sb	11

加载指令类型	LoadType[1:0]
lw	01
lh,lhu	10
lb,lbu	11
存储指令类型	LoadType[1:0]
sw	01
sh	10
sb	11

指令	NPCOp[1:0]	RFWr	EXTOp	ALUOp[3:0]	DMWr	A3Sel[1:0]	WDSel[1:0]	BSel	VSel	DAOp[1:0]	SSel	MDOp[3:0]	start	CSel	MD_instr

addu (000000/100001)	00	1	x	0000	0	00	00	0	x	xx	x	xxxx	0	0	0
add (000000/100000)	00	1	x	0000	0	00	00	0	x	xx	x	xxxx	0	0	0
subu (000000/100011)	00	1	x	0001	0	00	00	0	x	xx	x	xxxx	0	0	0
sub (000000/100010)	00	1	x	0001	0	00	00	0	x	xx	x	xxxx	0	0	0
and (000000/100100)	00	1	x	0010	0	00	00	0	x	xx	x	xxxx	0	0	0
or (000000/100101)	00	1	x	0011	0	00	00	0	x	xx	x	xxxx	0	0	0
xor	00	1	x	0100	0	00	00	0	x	xx	x	xxxx	0	0	0

(000000/100110)															
nor (000000/100111)	00	1	x	0101	0	00	00	0	x	xx	x	xxxx	0	0	0
slt (000000/101010)	00	1	x	0111	0	00	00	0	x	xx	x	xxxx	0	0	0
sltu (000000/101011)	00	1	x	1000	0	00	00	0	x	xx	x	xxxx	0	0	0
sll (000000/000000)	00	1	x	1001	0	00	00	0	0	xx	x	xxxx	0	0	0
sllv (000000/000100)	00	1	x	1001	0	00	00	0	1	xx	x	xxxx	0	0	0
srl (000000/000010)	00	1	x	1010	0	00	00	0	0	xx	x	xxxx	0	0	0

srlv (000000/000110)	00	1	x	1010	0	00	00	0	1	xx	x	xxxx	0	0	0
sra (000000/000011)	00	1	x	1011	0	00	00	0	0	xx	x	xxxx	0		0
srav (000000/000111)	00	1	x	1011	0	00	00	0	1	xx	x	xxxx	0	x0	0
mfhi (000000/010000)	00	1	x	xxxx	0	00	00	x	x	xx	x	0000	0	1	1
mflo (000000/010010)	00	1	x	xxxx	0	00	00	x	x	xx	x	0001	0	1	1
mthi (000000/010001)	00	0	x	xxxx	0	xx	xx	x	x	xx	x	0010	0	x	1
mtlo	00	0	x	xxxx	0	xx	xx	x	x	xx	x	0011	0	x	1

(000000/010011)															
mult (000000/011000)	00	0	x	xxxx	0	xx	xx	x	x	xx	x	0100	1	x	1
multu (000000/011001)	00	0	x	xxxx	0	xx	xx	x	x	xx	x	0101	1	x	1
div (000000/011010)	00	0	x	xxxx	0	xx	xx	x	x	xx	x	0110	1	x	1
divu (000000/011011)	00	0	x	xxxx	0	xx	xx	x	x	xx	x	0111	1	x	1
addi (001000)	00	1	1	0000	0	00	01	1	x	xx	x	xxxx	0	0	0
addiu (001001)	00	1	1	0000	0	00	01	1	x	xx	x	xxxx	0	0	0
ori	00	1	0	0011	0	00	01	1	x	xx	x	xxxx	0	0	0

(001101)															
andi (001100)	00	1	0	0010	0	00	01	1	x	xx	x	xxxx	0	0	0
xori (001110)	00	1	0	0100	0	00	01	1	x	xx	x	xxxx	0	0	0
slti (001010)	00	1	1	0111	0	00	01	1	x	xx	x	xxxx	0	0	0
sltiu (001011)	00	1	1	1000	0	00	01	1	x	xx	x	xxxx	0	0	0
lui (001111)	00	1	0	0110	0	01	00	1	x	xx	x	xxxx	0	0	0
lw (100011)	00	1	1	0000	0	01	01	1	x	00	x	xxxx	0	0	0
lh (100001)	00	1	1	0000	0	01	01	1	x	01	1	xxxx	0	0	0
lhu (100101)	00	1	1	0000	0	01	01	1	x	01	0	xxxx	0	0	0

lb (100000)	00	1	1	0000	0	01	01	1	x	10	1	xxxx	0	0	0
lbu (100100)	00	1	1	0000	0	01	01	1	x	10	0	xxxx	0	0	0
sw (101011)	00	0	1	0000	1	xx	xx	1	x	00	x	xxxx	0	0	0
sh (101001)	00	0	1	0000	1	xx	xx	1	x	01	x	xxxx	0	0	0
sb (101000)	00	0	1	0000	1	xx	xx	1	x	10	x	xxxx	0	0	0
beq (000100)	01	0	x	xxxx	0	xx	xx	x	x	xx	x	xxxx	0	x	0
bne (000101)	01	0	x	xxxx	0	xx	xx	x	x	xx	x	xxxx	0	x	0
blez (000110)	01	0	x	xxxx	0	xx	xx	x	x	xx	x	xxxx	0	x	0
bltz	01	0	x	xxxx	0	xx	xx	x	x	xx	x	xxxx	0	x	0

(000001)															
bgez (000001/00001)	01	0	x	xxxx	0	xx	xx	x	x	xx	x	xxxx	0	x	0
bgtz (000111)	01	0	x	xxxx	0	xx	xx	x	x	xx	x	xxxx	0	x	0
jal (000011)	10	1	x	xx	0	10	10	x	x	xx	x	xxxx	0	x	0
Jalr (000000/001001)	11	1	x	xx	0	00	10	x	x	xx	x	xxxx	0	x	0
j （000010）	10	0	x	xx	0	xx	xx	x	x	xx	x	xxxx	0	x	0
jr (000000/001000)	11	0	x	xx	0	xx	xx	x	x	xx	x	xxxx	0	x	0
mfc0	00	1	x	xxxx	0	01	01	x	x	xx	x	xxxx	0	x	0
mtc0	00	0	x	xxxx	0	xx	xx	x	x	xx	x	xxxx	0	x	0
eret	xx	x	x	xxxx	0	xx	xx	x	x	xx	x	xxxx	0	x	0

五、 测试方案

①测试新增指令的功能正确性

②测试新增指令的冒险处理正确性

③测试桥和 IO 的正确性

④测试各种异常是否能正确报出

构造表格中的异常 看是否能进入 4180

⑤测试各种情况下的中断能否正确处理

.普通指令中断，注意如 lw, mtc0, sw时候，下中断的情况，这时写使能应该关闭

普通气泡的中断，如执行 lw ,add时， 系统会增加气泡， lw nop add，在nop下中端

延时槽中断， beq add, 在add下中断，同时应该考虑 add发生异常且被下中断的情况

mult mflo 之间的延时槽下中断，根据是否会滚要求，具体判断

. 延时槽中气泡下中断，考虑情况mult beq mflo 其中mflo在延时槽里，但是实际情况会是， mult beq nop nop nop mflo,这时，在nop下中断

eret返回后，会系统增加nop ，即 eret, nop, xxx， 在eret返回后的nop中下中断

六、思考题

1、我们计组课程一本参考书目标题中有“硬件/软件接口”接口字样，那么到底什么是“硬件/软件接口”？（Tips：什么是接口？和我们到现在为止所学的有什么联系？）

是硬件与软件的交互方式，是联系硬件与软件的界面。硬件提供给软件一套操作硬件的方法（指令集），而软件根据硬件的结构与功能做出相应的调度。

2、在我们设计的流水线中，DM 处于 CPU 内部，请你考虑现代计算机中它的位置应该在何处。

在 CPU 外部，通过总线与 CPU 连接。

3、BE 部件对所有的外设都是必要的吗？

不是。timer 只支持整字读写，所以不需要 BE 信号。

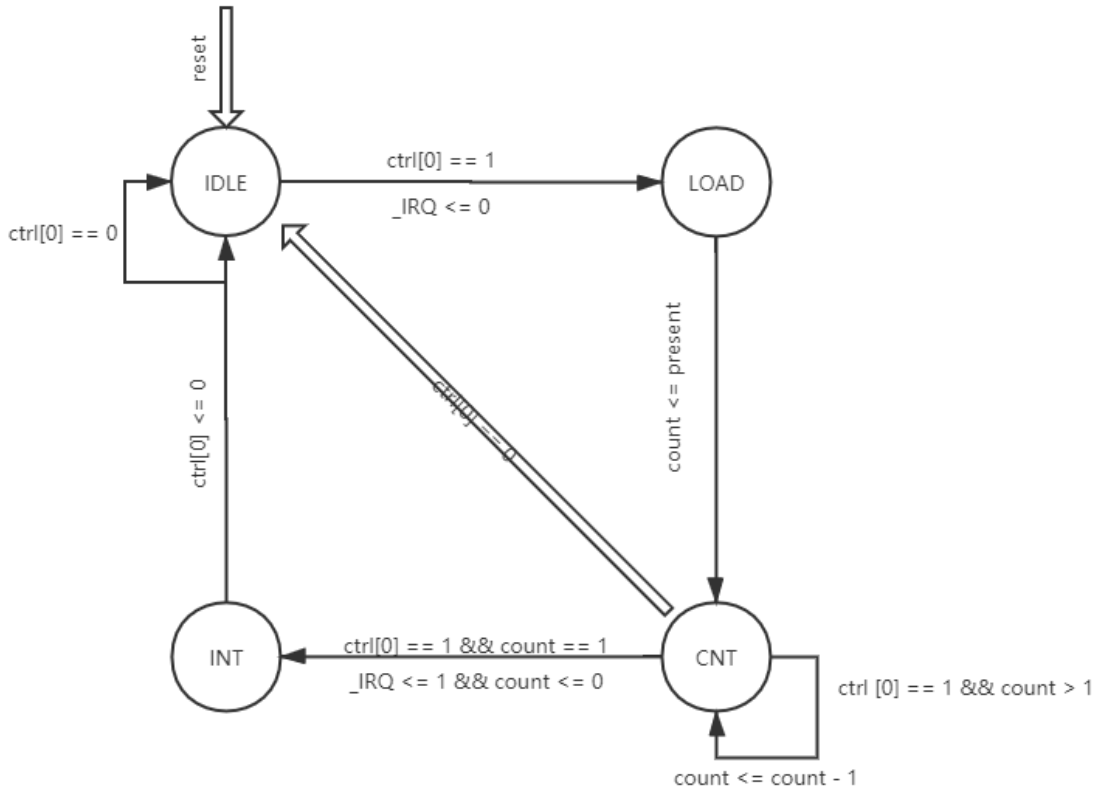
4、请阅读官方提供的定时器源代码，阐述两种中断模式的异同，并分别针对每一种模式绘制状态转移图。

两种中断模式基本相同，仅在从 INT 状态返回 IDLE 状态时行为有所不同。

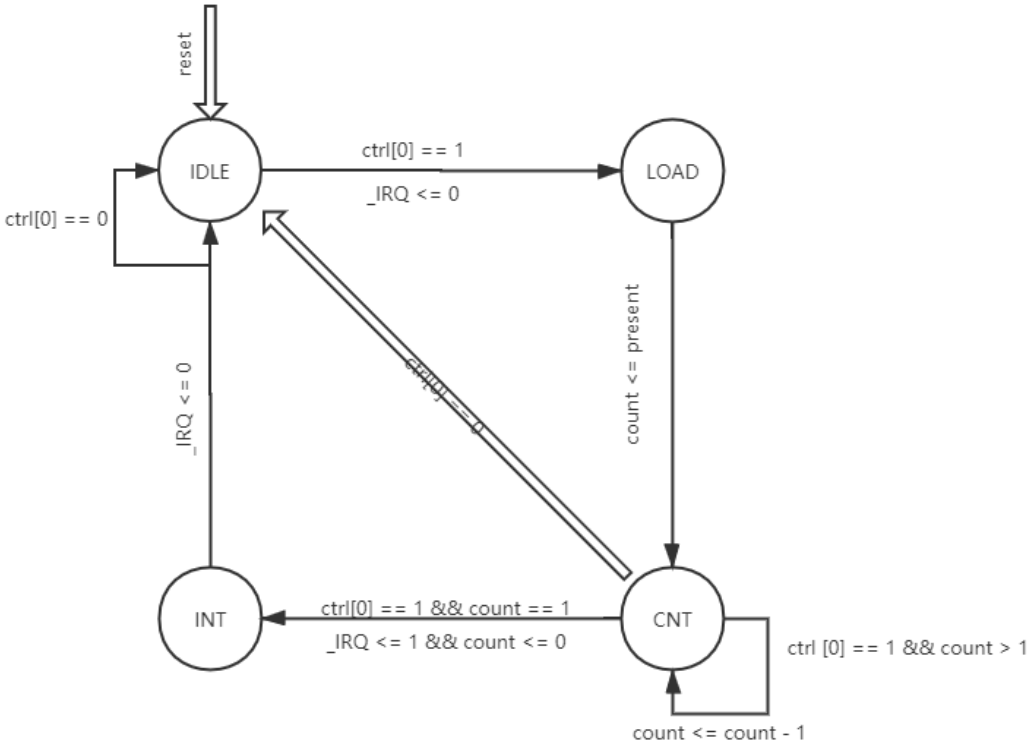
在模式 0 下，当计数器计数变为 0 后，ctrl[0]变为 0，计数器停止计数，此时中断信号将持续有效，直到控制寄存器中的中断屏蔽位被置为 0。

在模式 1 下，当计数器计数变为 0 后，初值寄存器中的值将被自动加载到计数器中，计数器继续倒数。不同于模式 0，模式 1 下计数器每次技术循环只产生一个周期的中断信号。

模式 0：



模式 1：



5、请开发一个主程序以及定时器的 exception handler。整个系统完成如下功能：

1. 定时器在主程序中被初始化为模式 0；
2. 定时器倒计数至 0 产生中断；
3. handler 设置使能 Enable 为 1 从而再次启动定时器的计数器。2 及 3 被无限重复。
4. 主程序在初始化时将定时器初始化为模式 0，设定初值寄存器的初值为某个值，如 100 或 1000。（注意，主程序可能需要涉及对 CP0.SR 的编程，推荐阅读过后文后再进行。）

```
.text

ori $1,$1,9

ori $2,$2,100
```

```
ori $3,$3,0x7f00

sw $1,0($3)

sw $2,4($3)

waiting:

beq $0,$0,waiting

nop



.ktext

ori $1,$1,9

ori $3,$3,0x7f00

sw $1,0($3)

eret
```

6、请查阅相关资料，说明鼠标和键盘的输入信号是如何被 CPU 知晓的？

鼠标移动或者键盘按下都将产生相应的中断信号，从而使 CPU 进入中断处理程序。