P4 单周期处理器（Verilog）设计文档

一、 数据通路设计

表 1 数据通路模块定义

文件	模块接口定义
datapath.v	module datapath(input Clk, input Reset, input [1:0] NPC_ctrl, input RegDst, input RegDst_2, input [1:0]grf_WDsrc, input RegWr, input EXT_ZS, input ALUsrc, input [1:0] ALUOp, input MemWr, output [5:0] Funct, output [5:0] Op, output Eq);

a) IFU：内部包括 PC，IM 和 NPC 共三个子模块。

表 2 IFU 端口定义

信号	方向	描述
Clk	I	全局时钟信号（由顶层统一接入时钟）
Reset	I	复位有效时，将 PC 清零
BrOffset[15:0]	I	Beq 指令中的偏移量
JrOffset[31:0]	I	Jr 指令偏移量
JalOffset[25:0]	I	Jal 指令偏移量
NPC_ctrl[1:0]	I	控制 NPC 模块输出下一条指令在 IM 中的地址
PC4	O	输出 PC+4 的值
ins[31:0]	O	输出当前指令

表 3 IFU 功能定义

序号	功能定义	功能描述
1	加载指令	初始化时通过加载 raw 文件将指令读取到 IM 中
2	读出指令	通过 PC 输出的地址输出 IM 中相应指令
3	计算 NPC（下一条指令地址）	用当前地址 PC 和相关信号计算 NPC

注：PC 起始地址为 0x00003000；

IM 容量为 1024 字且 ROM 寻址方式为字地址故而取 PC[11:2]

表 4 IFU 子模块接口及定义

文件	模块接口定义
pc.v	module pc(

	input Clk, //时钟信号 input Reset, //复位信号 input [31:0] Addr, //输入的下一条指令地址 output reg [31:0] Out //输出当前指令地址);
npc.v	module npc(input [31:0] PC, //当前指令地址 input [15:0] BrOffset, input [31:0] JrOffset, input [25:0] JalOffset, output [31:0] NPC, //下一条指令地址 output [31:0] PC4, input [1:0] NPC_ctrl);
im.v	module im(input [11:2] Addr, //当前指令地址 input Clk, output [31:0] Out //当前指令);

b) GRF（通用寄存器组）

表 5 GRF 模块定义

文件	模块接口定义
grf.v	module grf(input [4:0] A1, input [4:0] A2, input [4:0] A3, input [31:0] WD, output [31:0] RD1, output [31:0] RD2, input Clk, input RegWr, input Reset);

表 6 GRF 端口定义

信号	方向	描述
A1[4:0]	I	准备进行读取的第一个寄存器编号
A2[4:0]	I	准备进行读取的第二个寄存器编号
A3[4:0]	I	准备写入的寄存器编号
WD[31:0]	I	准备写入寄存器的 32 位数据
RD1[31:0]	O	读取的第一个寄存器中的数据

RD2[31:0]	O	读取的第二个寄存器中的数据
RegWr	I	寄存器写入使能
Clk	I	全局时钟信号
Reset	I	复位有效时，将所有寄存器清零

表 7 GRF 功能定义

序号	功能定义	功能描述
1	读取数据	输出指定编号（A1,A2）寄存器的内容
2	写入数据	将来自于 WD 的数据写入指定编号（A3）寄存器
3	存储数据	将数据保存在寄存器当中即使断电也不会清零

注：寄存器总数为 32 个；

0 号寄存器的值保持为 0。

c) ALU（算术逻辑单元）：

表 8 ALU 模块接口定义

文件	模块接口定义
alu.v	module alu(input [31:0] A, input [31:0] B, output [31:0] Out, output Eq, input [1:0] ALUOp);

表 9 ALU 端口定义

信号	方向	描述
A[31:0]	I	ALU 进行运算的第一个操作数
B[31:0]	I	ALU 进行运算的第二个操作数
Eq	O	判断两个输入是否相等，相等则输出 1
ALUop[1:0]	I	ALU 功能选择信号，决定输出哪一个运算结果
Out[31:0]	O	运算结果输出

表 10 ALUop 信号定义

ALUop	Func
00	加法
01	减法
10	比较大小，当且仅当 A>B 时输出 1
11	按位或运算

表 11 ALU 功能定义

序号	功能定义	功能描述
1	加	执行 A+B
2	减	执行 A-B
3	OR	按位计算 A or B
4	比较大小	当且仅当 A > B 输出 1

5	判断相等关系	当且仅当 A = B 输出 1

d) DM（数据存储器）

表 12 DM 模块接口定义

文件	模块接口定义
dm.v	module dm(input Clk, input MemWr, input [31:0] WD, input [31:0] Addr, input Reset, output [31:0] RD);

表 13 DM 端口定义

信号	方向	描述
WD[31:0]	I	将写入内存的数据
Addr[31:0]	I	准备进行读写的内存地址
MemWr	I	内存写使能
RD[31:0]	O	输出指定地址中存放的数据
Clk	I	全局时钟信号
Reset	I	复位有效时，将 Mem 清零

表 14 DM 功能定义

序号	功能定义	功能描述
1	读取	按照指定地址输出存储的数据
2	写入	将输入数据写入指定地址内存中

注：内存起始地址：0x00000000。

RAM 大小共计 1024 字

e) EXT：

表 15 EXT 模块接口定义

文件	模块接口定义
ext.v	module ext(input ExtOp, input [15:0] In, output [31:0] Out);

表 16 端口定义

信号	方向	描述
In[15:0]	I	输入 16 位带扩展数字
ExtOp	I	扩展功能选择
Out[31:0]	O	输出扩展结果

表 17 ExtOp 信号定义

ExtOp	Func
0	零扩展

1	符号扩展

表 18 EXT 功能定义

序号	功能定义	功能描述
1	零扩展	将输入数据扩展为 32 位并用 0 填充高 16 位
2	符号扩展	将输入数据扩展为 32 位并用原数据符号位填充高 16 位

f) SH16（16 位移位器）

表 19 SH16 模块接口定义

文件	模块接口定义
sl16.v	module sl16(input [15:0] in, output [31:0] out);

表 20 SH16 端口定义

信号	方向	描述
In[15:0]	I	输入十六位待移位数据
SH[31:0]	O	移位结果

表 21 SH16 功能定义

序号	功能定义	功能描述
1	16 位左移	将输入数据零扩展为 32 位后左移指定位数

g) MUX（多路选择器）

表 22 MUX 模块接口定义

文件	模块	模块接口定义
mux.v	mux2_5	module mux2_5(//5 位 2 选 1 input [4:0] d1, //输入数据 input [4:0] d2, input src, //输出选择 output [4:0] out //输出);
	mux4_32	module mux4_32(//32 位 4 选 1 input [1:0] src, input [31:0] d1, input [31:0] d2, input [31:0] d3, input [31:0] d4, output reg [31:0] out);
	mux2_32	module mux2_32(//32 位 2 选 1 input src, input [31:0] d1, input [31:0] d2,

		output [31:0] out);

二、 控制器设计

表 23 Controler 模块接口定义

文件	模块端口定义
control.v	module control(input [5:0] Funct, input [5:0] Op, input Eq, output reg [1:0] NPC_ctrl, output RegDst, output RegDst_2, output [1:0] grf_WDsrc, output RegWr, output EXT_ZS, output ALUsrc, output [1:0] ALUOp, output MemWr);

a) 为具有多个输入的模块添加选择信号确定输入来源，并根据指令做出真值表确定其他命令信号

表 24 控制信号

FUNC	100001	001000	100011	N/A				
OP	000000	000000	000000	001101	100011	101011	000100	001111	000011
控制信号	addu	jr	subu	ori	lw	sw	beq	lui	jal
RegDst	0	0	0	1	1	X	X	1	0
RegDst_2	0	0	0	0	0	0	0	0	1
grf_WDsrc	00	X	00	00	01	X	X	10	11
RegWr	1	0	1	1	1	0	0	1	1
EXT_ZS	X	X	X	0	1	1	1	X	X
ALUsrc	0	0	0	1	1	1	0	X	X
ALUOp	00	X	01	11	00	00	10	X	X
MemWr	0	0	0	0	0	1	0	0	0
NPC_ctrl	00	10	00	00	00	00	01	00	11

b) 具体参数

表 25 控制器端口定义

信号	方向	描述
Funct[5:0]	I	当前指令中的 Func 字段
Op[5:0]	I	当前指令中的操作码字段

RegDst	O	控制数据存储在 GRF 中的位置，0 取 rd 字段作为地址，1 取 rt 字段
RegDst_2	O	控制数据存储在 GRF 中的位置，0 取 RegDst 对应输出，1 取$31 输出
WDsrc[1:0]	O	GRF 写入数据来源
ALUsrc	O	ALU 操作数来源，0 为 RD2，1 为 EXT
EXT_ZS	O	选择扩展器进行零扩展（0）或者符号扩展（1）
RegWr	O	寄存器堆写使能
MemWr	O	内存写使能
ALUop[1:0]	O	ALU 功能选择信号
NPC_ctrl[1:0]	O	控制下一条指令地址

表 26 GRF 写入数据来源控制信号

WDsrc	Src
00	ALU 计算结果
01	Mem（内存）
10	SH（移位器）
11	PC+4

表 27 ALU 控制信号

ALUop	Func
00	加法

01	减法
10	比较大小，当且仅当 A>B 时输出 1
11	按位或运算

表 28 NPC 控制信号

NPC_ctrl	Src
00	PC+4
01	输出 beq 对应地址
10	Jr 地址
11	Jal 地址

三、 测试程序

```
ori     $t0, $t0, 0x1
addu    $t1, $t0, $t0
ori     $t2, $t2, 0x2
addu    $t3, $t2, $t2
lui     $t4, 0x2333
lui     $t5, 0x2333
addu    $t0, $t0, $t2
addu    $t0, $t0, $t4
subu    $t5, $t5, $t3
lui     $t5, 0x2333
sw      $t4, 0x4($t3)
lw      $t5, 0x4($t3)
```

```
lw      $t1, 0x4($t3)
jal     jal_0
subu    $t1, $t1, $t5
beq     $t1, $zero, beq_1
jal_0:
ori     $t1, $t1, 0x3333
ori     $t5, $t5, 0x3333
jr      $ra
beq_1:
nop
nop
beq     $t1, $zero, beq_1
```

测试程序期望结果是：

（1） 此程序最后进入死循环

（2） $ra = 0x00003038

（3） $12 = 0x23330000

（4） $13 = 0x23333333

（5） $10 = 0x00000002

（6） $11 = 0x00000004

（7） $8 = 0x23330003