P8　MIPS 微系统设计文档

一、　整体结构

本文档所描述的微系统包括处理器，系统桥和外设部分。处理器为 32 位五级流水线处理器，采用 Verilog HDL 实现。该处理器支持的指令集为 MIPS-C5={LB、LBU、LH、LHU、LW、SB、SH、SW、ADD、ADDU、SUB、SUBU、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、ERET、MFC0、MTC0}，且支持异常、中断和延迟槽。

处理器采用模块化和层次化设计，包含 Controller（控制器）、PC（程序计数器）、IM（指令存储器）、Adder（加法器）、NPC（下一条指令地址计算单元）、GRF（通用寄存器组）、CMP（比较单元）、ALU（算术逻辑单元）、DM（数据存储器）、EXT（位扩展器）等基本部件。

二、　基本模块规格

1. PC（程序计数器）

1) 基本描述

PC 存储当前指令地址，并在时钟上升沿更新值。

2) 端口说明

表 1 PC 端口说明

信号名	方向	描述
Clk	I	时钟信号。
Reset	I	复位信号（高电平有效）。
En	I	写使能信号（高电平有效）。
In[31:0]	I	输入下一个时钟上升沿 PC 要写入的值。
Out[31:0]	O	输出当前 PC 的值。

3) 功能定义

表 2 PC 功能定义

序号	功能名称	功能描述
1	同步复位	当时钟上升沿到来时，若复位信号有效，PC 被设置为 0x00003000。
2	输出当前指令地址	存储并由 Out 输出当前指令地址。
3	更新 PC	当时钟上升沿到来时，若写使能信号有效且复位信号无效，则更新 PC 为 In 的值。

2. IM（指令存储器）

1) 基本描述

IM 存储 CPU 要执行的指令，并输出输入地址所对应的指令。IM 的容量为 32bit*2048。

2) 端口说明

表 3 IM 端口说明

信号名	方向	描述
Addr[31:0]	I	输入指令地址。

Instr[31:0]	O	输出 Addr 所对应的指令。

3) 功能定义

表 4 IM 功能定义

序号	功能名称	功能描述
1	输出当前指令	由 Instr 输出 Addr 所对应的指令。

3. Adder（加法器）

1) 基本描述

Adder 输入当前 PC 的值，输出 PC+4 的值。

2) 端口说明

表 5 Adder 端口说明

信号名	方向	描述
In[31:0]	I	数据输入信号，输入当前 PC 的值。
Out[31:0]	O	数据输出信号，输出 PC+4 的值。

3) 功能定义

表 6 Adder 功能定义

序号	功能名称	功能描述
1	PC+4	由 Out 输出 PC+4 的值。

4. NPC（下一条指令地址计算单元）

1) 基本描述

NPC 根据当前指令地址和相应的控制信号计算出下一条指令地址，并输出 PC+8 的值。

2) 端口说明

表 7 NPC 端口说明

信号名	方向	描述
NPCOp	I	指定 NPC 要执行的操作： 0：B 型指令； 1：J 型指令。
PC4[31:0]	I	输入 PC+4。
Imm26[25:0]	I	输入 26 位立即数，用于计算分支或跳转后 PC 的值。
Out[31:0]	O	输出计算出的下一条指令地址。
PC8[31:0]	O	输出 PC+8。

3) 功能定义

表 8 NPC 功能定义

序号	功能名称	功能描述
1	计算下一条指令地址	当 NPCOp 为 0 时，Out 输出 PC4+SignExt(Imm26[15:0]||0^2); 当 NPCOp 为 1 时，Out 输出(PC4)[31:28]||Imm26||0^2。
2	输出 PC+8	由 PC8 输出 PC+8 的值。

5. GRF（通用寄存器组）

1) 基本描述

GRF 内部包括 32 个具有复位功能的寄存器。其中，0 号寄存器的值始终保持为 0，其他寄存器初始值均为 0。GRF 提供同时读取 2 个寄存器和写入 1 个寄存器的功能，并且支持内部转发。

2) 端口说明

表 9 GRF 端口说明

信号名	方向	描述
Clk	I	时钟信号。
Reset	I	复位信号（高电平有效）。
A1[4:0]	I	地址输入信号 1，将其对应寄存器中存储的数据输出至 RD1。
A2[4:0]	I	地址输入信号 2，将其对应寄存器中存储的数据输出至 RD2。
A3[4:0]	I	地址输入信号 3，指定写入操作所对应的寄存器。
WD[31:0]	I	数据输入信号，即要写入寄存器中的数据。
RD1[31:0]	O	数据输出信号，输出 A1 对应寄存器中的 32 位数据。
RD2[31:0]	O	数据输出信号，输出 A2 对应寄存器中的 32 位数据。

注：在 Verilog 实现中增加了 WPC[31:0]输入，用于在线测试时输出 PC 的值。

3) 功能定义

表 10 GRF 功能定义

序号	功能名称	功能描述
1	同步复位	当时钟上升沿到来时，若复位信号有效，GRF 中的每一个寄存器都被设置为 0x00000000。
2	读取数据	读取 A1 和 A2 所对应寄存器中的数据至 RD1 和 RD2（当同一个寄存器同时被写入和读取时，读取的值为写入的值）。
3	写入数据	当时钟上升沿到来时，如果复位信号无效，就将 WD 写入 A3 所对应的寄存器中。

6. CMP（比较单元）

1) 基本描述

CMP 对输入的两个操作数提供相等比较功能，并对输入的第一个操作数提供零比较功能，输出比较结果。

2) 端口说明

表 11 CMP 端口说明

信号名	方向	描述
A[31:0]	I	数据输入信号，输入 CMP 的第一个操作数。
B[31:0]	I	数据输入信号，输入 CMP 的第二个操作数。
Equal	O	相等标志信号（高电平有效），标志两操作数是否相等。
LTZ	O	小于 0 标志信号（高电平有效），标志 A 是否小于 0。
EQZ	O	等于 0 标志信号（高电平有效），标志 A 是否等于 0。

3) 功能定义

表 12 CMP 功能定义

序号	功能名称	功能描述
1	相等比较运算	若 A=B，则 Equal 信号有效；否则无效。

2	零比较运算	若 A<0，则 LTZ 信号有效；否则无效。 若 A=0，则 EQZ 信号有效；否则无效。

7. ALU（算术逻辑单元）

1) 基本描述

ALU 对输入的两个操作数提供 32 位加、减、或、与、或非、异或和移位运算以及小于置位功能，输出运算结果。

2) 端口说明

表 13 ALU 端口说明

信号名	方向	描述
A[31:0]	I	数据输入信号，输入 ALU 的第一个操作数。
B[31:0]	I	数据输入信号，输入 ALU 的第二个操作数。
ALUOp[3:0]	I	指定 ALU 所要进行的操作： 0000：A+B； 0001：A-B； 0010：A|B； 0011：A&B； 0100：~(A|B)； 0101：A^B； 0110：B<<A[4:0]； 0111：B>>A[4:0]； 1000：B>>>A[4:0]； 1001：(A<B)?1:0； 1010：((0|A)<(0|B))?1:0。
Result[31:0]	O	数据输出信号，输出 ALU 的计算结果。

3) 功能定义

表 14 ALU 功能定义

序号	功能名称	功能描述
1	加法	当 ALUOp 为 0000 时，Result 输出 A+B 的值。
2	减法	当 ALUOp 为 0001 时，Result 输出 A-B 的值。
3	或运算	当 ALUOp 为 0010 时，Result 输出 A|B 的值。
4	与运算	当 ALUOp 为 0011 时，Result 输出 A&B 的值。
5	或非运算	当 ALUOp 为 0100 时，Result 输出~(A|B)的值。
6	异或运算	当 ALUOp 为 0101 时，Result 输出 A^B 的值。
7	逻辑左移运算	当 ALUOp 为 0110 时，Result 输出 B<<A[4:0]的值。
8	逻辑右移运算	当 ALUOp 为 0111 时，Result 输出 B>>A[4:0]的值。
9	算术右移运算	当 ALUOp 为 1000 时，Result 输出 B>>>A[4:0]的值。
10	小于比较运算	当 ALUOp 为 1001 时，若 A<B，则 Result 输出 1；否则 Result 输出 0。
11	无符号小于比较运算	当 ALUOp 为 1010 时，若(0|A)<(0|B)，则 Result 输出 1；否则 Result 输出 0。

8. DM（数据存储器）

1) 基本描述

DM 用于存储数据，其容量为 32bit*2048，起始地址为 0x00000000。DM 支持同步复位功能，并且数据读取和写入端口分离。

2) 端口说明

表 15 DM 端口说明

信号名	方向	描述
Clk	I	时钟信号。
Reset	I	复位信号（高电平有效）。
Addr[31:0]	I	地址信号，指定要操作的存储单元的地址。
WD[31:0]	I	数据输入信号，输入要写入到 Addr 所对应的存储单元的数据。
MemWrite	I	写使能信号（高电平有效）。
OpWidth[1:0]	I	指定操作位宽： 00：Word； 01：Half； 10：Byte。
LoadSigned	I	指定是否进行有符号读取（高电平有效）。
RD[31:0]	O	数据输出信号，输出 Addr 所对应的存储单元的数据。

注：在 Verilog 实现中增加了 WPC[31:0]输入，用于在线测试时输出 PC 的值。

3) 功能定义

表 16 DM 功能定义

序号	功能名称	功能描述
1	同步复位	当时钟上升沿到来时，若复位信号有效，DM 中的每一个存储单元都被设置为 0x00000000。
2	读取	RD 根据 OpWidth 和 LoadSigned 信号输出 Addr 所对应的存储单元的数据。
3	写入	当时钟上升沿到来时，如果 MemWrite 有效且复位信号无效，就根据 OpWidth 信号将 WD 写入 Addr 所对应的存储单元中。

9. EXT（位扩展器）

1) 基本描述

EXT 用于将输入的 16 位立即数根据操作信号扩展成 32 位并输出。

2) 端口说明

表 17 EXT 端口说明

信号名	方向	描述
Imm16[15:0]	I	数据输入信号，输入要进行扩展的数据。
ExtOp[1:0]	I	符号扩展信号（高电平有效）。
Imm32[31:0]	O	数据输出信号，输出扩展后的数据。

3) 功能定义

表 18 EXT 功能定义

序号	功能名称	功能描述
1	无符号扩展	当 ExtOp 为 00 时，将 Imm16 无符号扩展至 32 位并输出至 Imm32。
2	符号扩展	当 ExtOp 为 01 时，将 Imm16 符号扩展至 32 位并输出至 Imm32。
3	左移 16 位	当 ExtOp 为 10 时，将 Imm16 左移 16 位并输出至 Imm32。

三、 数据通路设计

见 Excel 表格。

四、 数据通路参考示意图

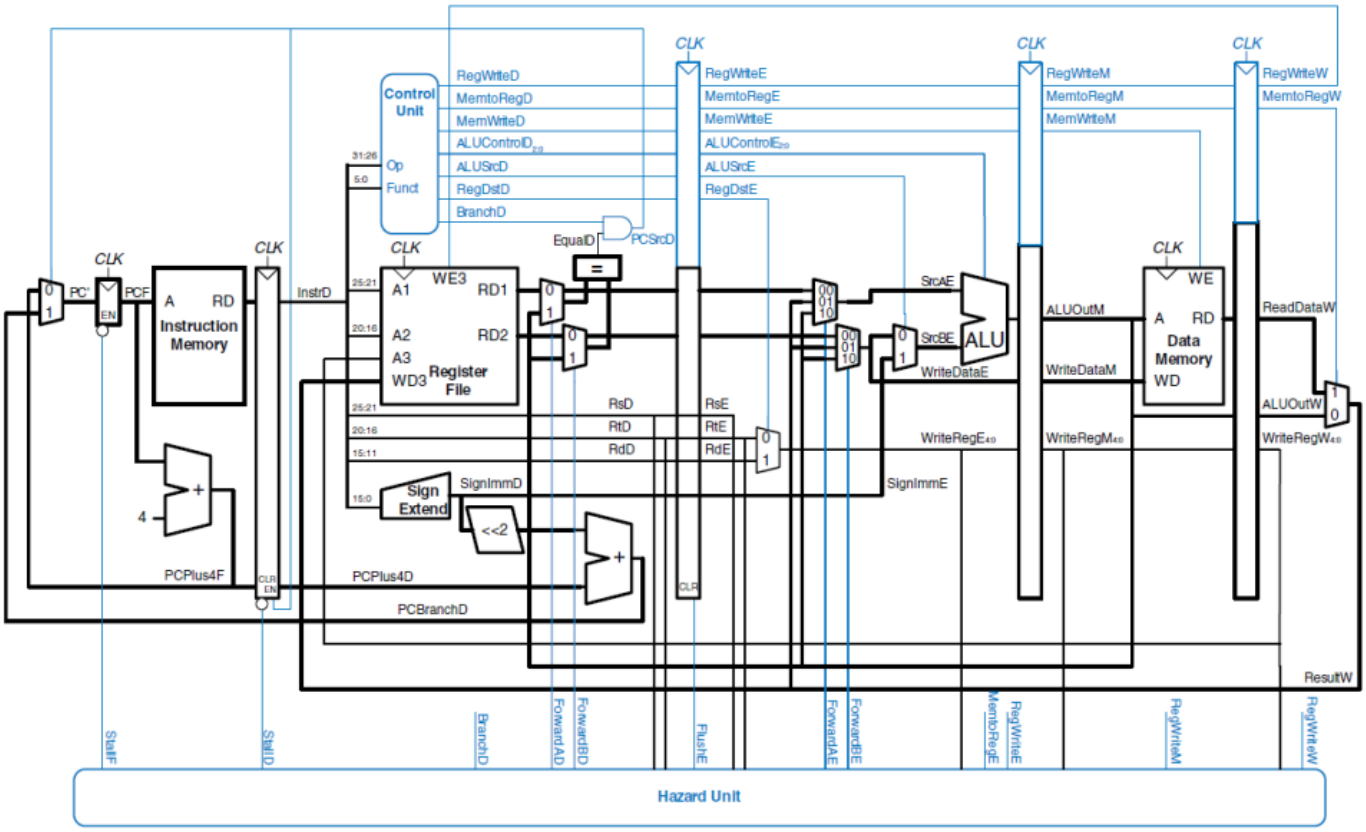

Figure 7.58 Pipelined processor with full hazard handling

五、 控制器（Controller）设计

1. 基本描述

控制器分为主控制器和冒险处理单元。主控制器通过输入的 Op 和 Funct 信号以及 CMP 产生的比较信号产生数据通路所需的控制信号，采用分布式译码，实例化 3 个；冒险处理单元负责为几个转发位点提供数据转发并通过检测无法由转发解决的数据冒险来插入暂停。

2. D 级主控制器真值表

表 19 D 级主控制器真值表

指令(Op/Funct)	NPC Op	ExtOp[1:0]	PCSrc[1:0]	A3Sel[1:0]	GenD	MD	D1Use	D2Use	BD
addu (000000/100001)	x	xx	00	10	0	0	0	0	0
subu (000000/100011)	x	xx	00	10	0	0	0	0	0
ori (001101)	x	00	00	11	0	0	0	0	0
lw (100011)	x	01	00	11	0	0	0	0	0
sw (101011)	x	01	00	00	0	0	0	0	0
beq	0	01	CMP.Equal?01:00	00	0	0	1	1	1

(000100)									
lui (001111)	x	10	00	11	0	0	0	0	0
j (000010)	1	xx	01	00	0	0	0	0	1
jal (000011)	1	xx	01	01	1	0	0	0	1
jr (000000/001000)	x	xx	10	00	0	0	1	0	1
lb (100000)	x	01	00	11	0	0	0	0	0
lbu (100100)	x	01	00	11	0	0	0	0	0
lh (100001)	x	01	00	11	0	0	0	0	0
lhu (100101)	x	01	00	11	0	0	0	0	0
sb (101000)	x	01	00	00	0	0	0	0	0
sh (101001)	x	01	00	00	0	0	0	0	0
add (000000/100000)	x	xx	00	10	0	0	0	0	0
sub (000000/100010)	x	xx	00	10	0	0	0	0	0
sll (000000/000000)	x	xx	00	10	0	0	0	0	0
srl (000000/000010)	x	xx	00	10	0	0	0	0	0
sra (000000/000011)	x	xx	00	10	0	0	0	0	0
sllv (000000/000100)	x	xx	00	10	0	0	0	0	0
srlv (000000/000110)	x	xx	00	10	0	0	0	0	0
srav (000000/000111)	x	xx	00	10	0	0	0	0	0
and (000000/100100)	x	xx	00	10	0	0	0	0	0
or (000000/100101)	x	xx	00	10	0	0	0	0	0
xor (000000/100110)	x	xx	00	10	0	0	0	0	0
nor (000000/100111)	x	xx	00	10	0	0	0	0	0
addi (001000)	x	01	00	11	0	0	0	0	0
addiu	x	01	00	11	0	0	0	0	0

(001001)									
andi (001100)	x	00	00	11	0	0	0	0	0
xori (001110)	x	00	00	11	0	0	0	0	0
slt (000000/101010)	x	xx	00	10	0	0	0	0	0
slti (001010)	x	01	00	11	0	0	0	0	0
sltiu (001011)	x	01	00	11	0	0	0	0	0
sltu (000000/101011)	x	xx	00	10	0	0	0	0	0
bne (000101)	0	01	!CMP.Equal?01:00	00	0	0	1	1	1
blez (000110)	0	01	(CMP.LTZ||CMP.EQZ)?01:00	00	0	0	1	0	1
bgtz (000111)	0	01	(!CMP.LTZ&&!CMP.EQZ)?01:00	00	0	0	1	0	1
bltz (000001/Instr[20:16]=00000)	0	01	CMP.LTZ?01:00	00	0	0	1	0	1
bgez (000001/Instr[20:16]=00001)	0	01	!CMP.LTZ?01:00	00	0	0	1	0	1
jalr (000000/001001)	x	xx	10	10	1	0	1	0	1
eret (010000/011000)	x	xx	xx	00	0	0	0	0	0
mfc0 (010000/Instr[25:21]=00000)	x	xx	00	11	0	0	0	0	0
mtc0 (010000/Instr[25:21]=00100)	x	xx	00	00	0	0	0	0	0

3. E 级主控制器真值表

表 20 E 级主控制器真值表

指令(Op/Funct)	ALUOp[3:0]	ALUSrcA	ALUSrcB	Start	MDUOp[1:0]	HIWrite	LOWrite	GenE[1:0]	E1Use	E2Use
addu (000000/100001)	0000	0	0	0	xx	0	0	01	1	1
subu (000000/100011)	0001	0	0	0	xx	0	0	01	1	1
ori (001101)	0010	0	1	0	xx	0	0	01	1	0
lw (100011)	0000	0	1	0	xx	0	0	00	1	0

sw (101011)	0000	0	1	0	xx	0	0	00	1	0
beq (000100)	xxxx	x	x	0	xx	0	0	00	0	0
lui (001111)	0000	0	1	0	xx	0	0	01	0	0
j (000010)	xxxx	x	x	0	xx	0	0	00	0	0
jal (000011)	xxxx	x	x	0	xx	0	0	00	0	0
jr (000000/001000)	xxxx	x	x	0	xx	0	0	00	0	0
lb (100000)	0000	0	1	0	xx	0	0	00	1	0
lbu (100100)	0000	0	1	0	xx	0	0	00	1	0
lh (100001)	0000	0	1	0	xx	0	0	00	1	0
lhu (100101)	0000	0	1	0	xx	0	0	00	1	0
sb (101000)	0000	0	1	0	xx	0	0	00	1	0
sh (101001)	0000	0	1	0	xx	0	0	00	1	0
add (000000/100000)	0000	0	0	0	xx	0	0	01	1	1
sub (000000/100010)	0001	0	0	0	xx	0	0	01	1	1
sll (000000/000000)	0110	1	0	0	xx	0	0	01	0	1
srl (000000/000010)	0111	1	0	0	xx	0	0	01	0	1
sra (000000/000011)	1000	1	0	0	xx	0	0	01	0	1
sllv (000000/000100)	0110	0	0	0	xx	0	0	01	1	1
srlv (000000/000110)	0111	0	0	0	xx	0	0	01	1	1
srav (000000/000111)	1000	0	0	0	xx	0	0	01	1	1
and (000000/100100)	0011	0	0	0	xx	0	0	01	1	1
or (000000/100101)	0010	0	0	0	xx	0	0	01	1	1
xor (000000/100110)	0101	0	0	0	xx	0	0	01	1	1
nor (000000/100111)	0100	0	0	0	xx	0	0	01	1	1

addi (001000)	0000	0	1	0	xx	0	0	01	1	0
addiu (001001)	0000	0	1	0	xx	0	0	01	1	0
andi (001100)	0011	0	1	0	xx	0	0	01	1	0
xori (001110)	0101	0	1	0	xx	0	0	01	1	0
slt (000000/101010)	1001	0	0	0	xx	0	0	01	1	1
slti (001010)	1001	0	1	0	xx	0	0	01	1	0
sltiu (001011)	1010	0	1	0	xx	0	0	01	1	0
sltu (000000/101011)	1010	0	0	0	xx	0	0	01	1	1
bne (000101)	xxxx	x	x	0	xx	0	0	00	0	0
blez (000110)	xxxx	x	x	0	xx	0	0	00	0	0
bgtz (000111)	xxxx	x	x	0	xx	0	0	00	0	0
bltz (000001/Instr[20:16]= 00000)	xxxx	x	x	0	xx	0	0	00	0	0
bgez (000001/Instr[20:16]= 00001)	xxxx	x	x	0	xx	0	0	00	0	0
jalr (000000/001001)	xxxx	x	x	0	xx	0	0	00	0	0
eret (010000/011000)	xxxx	x	x	0	xx	0	0	00	0	0
mfc0 (010000/Instr[25:21]= 00000)	xxxx	x	x	0	xx	0	0	00	0	0
mtc0 (010000/Instr[25:21]= 00100)	xxxx	x	x	0	xx	0	0	00	0	0

4. M 级主控制器真值表

表 21 M 级主控制器真值表

指令(Op/Funct)	MemWrite	OpWidth[1:0]	LoadSigned	GenM[1:0]	CP0Write
addu (000000/100001)	0	xx	x	00	0
subu (000000/100011)	0	xx	x	00	0
ori (001101)	0	xx	x	00	0

lw (100011)	0	00	x	01	0
sw (101011)	1	00	x	00	0
beq (000100)	0	xx	x	00	0
lui (001111)	0	xx	x	00	0
j (000010)	0	xx	x	00	0
jal (000011)	0	xx	x	00	0
jr (000000/001000)	0	xx	x	00	0
lb (100000)	0	10	1	01	0
lbu (100100)	0	10	0	01	0
lh (100001)	0	01	1	01	0
lhu (100101)	0	01	0	01	0
sb (101000)	1	10	x	00	0
sh (101001)	1	01	x	00	0
add (000000/100000)	0	xx	x	00	0
sub (000000/100010)	0	xx	x	00	0
sll (000000/000000)	0	xx	x	00	0
srl (000000/000010)	0	xx	x	00	0
sra (000000/000011)	0	xx	x	00	0
sllv (000000/000100)	0	xx	x	00	0
srlv (000000/000110)	0	xx	x	00	0
srav (000000/000111)	0	xx	x	00	0
and (000000/100100)	0	xx	x	00	0
or (000000/100101)	0	xx	x	00	0
xor (000000/100110)	0	xx	x	00	0

nor (000000/100111)	0	xx	x	00	0
addi (001000)	0	xx	x	00	0
addiu (001001)	0	xx	x	00	0
andi (001100)	0	xx	x	00	0
xori (001110)	0	xx	x	00	0
slt (000000/101010)	0	xx	x	00	0
slti (001010)	0	xx	x	00	0
sltiu (001011)	0	xx	x	00	0
sltu (000000/101011)	0	xx	x	00	0
bne (000101)	0	xx	x	00	0
blez (000110)	0	xx	x	00	0
bgtz (000111)	0	xx	x	00	0
bltz (000001/Instr[20:16]=00000)	0	xx	x	00	0
bgez (000001/Instr[20:16]=00001)	0	xx	x	00	0
jalr (000000/001001)	0	xx	x	00	0
eret (010000/011000)	0	xx	x	00	0
mfc0 (010000/Instr[25:21]=00000)	0	xx	x	10	0
mtc0 (010000/Instr[25:21]=00100)	0	xx	x	00	1

5. 暂停策略

采用标记转发法，当需求寄存器的值尚未算出时暂停，具体策略如下：

```
assign StallD = ((D1Use && A1D == A3E && A3E != 0 && WDE === 32'bz) ||
                 (D1Use && A1D == A3M && A3M != 0 && WDM === 32'bz && !(A1D
== A3E && A3E != 0 && WDE !== 32'bz)) ||
                 (D2Use && A2D == A3E && A3E != 0 && WDE === 32'bz) ||
                 (D2Use && A2D == A3M && A3M != 0 && WDM === 32'bz && !(A2D
== A3E && A3E != 0 && WDE !== 32'bz)))
```

```
                    && !StallE;
assign StallE = (E1Use && A1E == A3M && A3M != 0 && WDM === 32'bz) ||
                    (E2Use && A2E == A3M && A3M != 0 && WDM === 32'bz);

   assign PCEn = ~(StallD || StallE);
   assign DRegEn = ~(StallD || StallE);
   assign ERegEn = ~StallE;
   assign ERegFlush = StallD;
   assign MRegFlush = StallE;
```

6. 转发策略

采用标记转发法，为每一个需求者增加转发，具体策略如下：

```
assign ForwardD1 = (A1D == A3E && A3E != 0) ? WDE :
                   (A1D == A3M && A3M != 0) ? WDM :
                   RD1D;
   assign ForwardD2 = (A2D == A3E && A3E != 0) ? WDE :
                   (A2D == A3M && A3M != 0) ? WDM :
                   RD2D;
   assign ForwardE1 = (A1E == A3M && A3M != 0) ? WDM :
                   (A1E == A3W && A3W != 0) ? WDW :
                   (A1E == A3T && A3T != 0) ? WDT :
                   RD1E;
   assign ForwardE2 = (A2E == A3M && A3M != 0) ? WDM :
                   (A2E == A3W && A3W != 0) ? WDW :
                   (A2E == A3T && A3T != 0) ? WDT :
                   RD2E;
   assign ForwardM2 = (A2M == A3W && A3W != 0) ? WDW :
                   RD2M;
```

六、 CP0 设计

1. 基本描述

CP0 内部包含 SR 寄存器的部分位、CAUSE 寄存器的部分位、EPC 寄存器、PrID 寄存器以及产生中断信号的组合逻辑，用于处理 CPU 的异常与中断状态。

2. 端口说明

表 22 CP0 端口说明

信号名	方向	描述
Clk	I	时钟信号。
Reset	I	复位信号（高电平有效）。
Addr[4:0]	I	地址信号，用于指定操作 CP0 寄存器的地址。
DIn[31:0]	I	数据输入信号，用于指定向 CP0 寄存器中写入的值。
PC[31:2]	I	输入当前流水级 PC 相应位的值。
ExcCode[6:2]	I	输入异常代码。
HWInt[7:2]	I	输入外部中断信号。
WE	I	寄存器写使能（高电平有效）。
ExlSet	I	异常进入信号。
ExlClr	I	异常返回信号。
BD	I	延迟槽信号，用于指定当前指令是否为延迟槽指令。
IntReq	O	中断请求指令（高电平有效）。
EPC[31:0]	O	输出 EPC 寄存器的值。
DOut[31:0]	O	数据输出信号，用于输出指定寄存器的值。

3. 功能定义

表 23 CP0 功能定义

序号	功能名称	功能描述
1	同步复位	当时钟上升沿到来时，若复位信号有效，CP0 中的每一个寄存器都被设置为 0x00000000（PrID 寄存器除外）。
2	读取寄存器	DOut 输出 Addr 所对应的寄存器的数据（未定义寄存器及未定义位输出 0）。
3	写入寄存器	当时钟上升沿到来时，如果 WE 信号有效且复位信号无效，就将 DIn 写入 Addr 所对应的寄存器中。
4	输出 EPC	输出 EPC 寄存器的值。
5	中断请求	如果 EXLSet 信号或中断条件有效且 EXL 信号无效且复位信号无效，IntReq 信号就有效，否则无效。
6	中断异常处理	当时钟上升沿到来时，如果 EXLSet 信号或中断条件有效且 EXL 信号无效且复位信号无效，就将 SR 寄存器的 EXL 位置 1，将 CAUSE 寄存器的 BD 位和 ExcCode 位设置为相应的输入，将 EPC 寄存器设置为相应的输入。
7	中断异常返回	当时钟上升沿到来时，如果 EXLClr 信号有效且复位信号无效，就将 SR 寄存器的 EXL 位置 0。
8	更新 IP	当时钟上升沿到来时，如果复位信号无效，就将 HWInt 存入 CAUSE 寄存器的 IP 位中。

七、 桥与 IO 设计

1. 基本描述

微系统通过桥实现 IO，连接 6 个外设。

2. 端口说明

表 24 Bridge 端口说明

信号名	方向	描述
PrAddr[31:2]	I	输入处理器操作地址。
PrWD[31:0]	I	输入处理器要向外设中写入的数据。
DEVRD0[31:0]	I	输入外设 0 读出的数据。
DEVRD1[31:0]	I	输入外设 1 读出的数据。
DEVRD2[31:0]	I	输入外设 2 读出的数据。
DEVRD3[31:0]	I	输入外设 3 读出的数据。
DEVRD4[31:0]	I	输入外设 4 读出的数据。
DEVRD5[31:0]	I	输入外设 5 读出的数据。
PrWE	I	处理器写使能信号（高电平有效）。
PrRD[31:0]	O	输出处理器从外设中读取的数据。
DEVAddr[31:2]	O	输出对外设操作的地址。
DEVWD[31:0]	O	输出将要写入外设的数据。
DEVWE0	O	设备 0 写使能信号（高电平有效）。
DEVWE1	O	设备 1 写使能信号（高电平有效）。
DEVWE3	O	设备 3 写使能信号（高电平有效）。
DEVWE4	O	设备 4 写使能信号（高电平有效）。

3. 功能定义

表 25 Bridge 功能定义

序号	功能名称	功能描述
1	外设读取	根据 PrAddr 从对应外设中读取数据并驱动 PrRD。
2	外设写入	当 PrWE 信号有效时，根据 PrAddr 将 PrWD 写入对应外设中。

八、 测试软件

见 asm 文件。

思考题

1. 请查阅相关资料，说一说什么是「FPGA 技术」？它有哪些好处和缺陷？

   FPGA 技术即“现场可编程逻辑门阵列”,它是一种半定制电路,可根据不同的需要进行编程。它功能丰富，设计方式灵活，可以无限地重新编程，并且可以自定义时钟频率，速度较快；但是，由于 FPGA 的所有功能均依靠硬件实现，它无法实现分支条件跳转等操作，且只能实现定点运算，灵活度有限。

2. 在上述步骤中，同学们可能会出现各种各样的问题，例如综合失败、无法布局布线等，或者也有同学会尝试消除所有的 Warning。无论是何种情况，希望同学们能记录下自己的问题和解决的过程，并体现自己对实验的理解（例如对 FPGA 的理解，对 Verilog 语法可综合性的理解等）。

   问题：由于时钟频率设置过快导致无法实现设计。

   解决方法：降低时钟 IP 核输出的时钟频率。

   理解：可能是由于设计中的关键路径过长，导致无法满足时钟约束。

3. 简述你的中断实现方案。

   为 UART 模块增加 INT 输出，并且连接到 rs 信号。