五级流水线CPU设计文档

(一)CPU设计要求

32位五级流水线CPU

支持指令

```
add, sub, and, or, slt, sltu, lui
addi, andi, ori
lb, lh, lw, sb, sh, sw
mult, multu, div, divu, mfhi, mflo, mthi, mtlo
beq, bne, jal, jr
mtc0,mfc0,syscall,eret
```

(二)关键模块设计

mips_cpu

```
module mips_cpu(
    input         clk,
    input         reset,
    input   [5:0] HWInt,          // 外部中断
    input  [31:0] i_inst_rdata,   // F级的32bit指令
    input  [31:0] m_data_rdata,   // 从DM读取的值
    input         interrupt,      // 外部中断信号
    output [31:0] i_inst_addr,    // PC_F
    output [31:0] m_data_addr,    // DM 读写地址
    output [31:0] m_data_wdata,   // DM 待写入数据
    output [3 :0] m_data_byteen,  // DM 字节使能信号
    output [31:0] m_inst_addr,    // PC_M
    output        w_grf_we,       // GRF 写使能信号
    output  [4:0] w_grf_addr,     // GRF 待写入寄存器编号
    output [31:0] w_grf_wdata,    // GRF 待写入数据
    output [31:0] w_inst_addr,    // W 级 PC
    output [31:0] macroscopic_pc,// 宏观 PC
    output        Intrespon       // 中断相应
);
```

IFU

模块定义

信号名	方向	描述
clk	I	时钟信号
Reset	I	异步复位信号

信号名	方向	描述
NPCop[1:0]	I	两位控制信号，控制NPC的的值 00：PC←PC+4 01：PC←$PC_{31...28}$||instr_index||0_2 10：PC←PC + 4 + sign_extend(offset||0_2) 11：PC←GPR[rs]
Req	I	中断或异常信号
D_eret	I	D级是否是eret指令
EPC[31:0]	I	32位输入，异常处理返回地址
instr_index[26:0]	I	26位输入信号，用于PC的计算
offset[16:0]	I	16位输入信号，PC的偏移量
Reg[32:0]	I	32位输入信号，用于寄存器地址的跳转
Judge	I	一位输入，作为跳转的判断依据
PC_in[31:0]	I	32位输入，D级的PC地址
PCwrite	I	控制是否进行PC修改
Instr[31:0]	O	32为输出信号，输出当前要执行的指令
PC[31:0]	O	32位输出信号，当前PC的地址
BD	O	1位输出信号，标志是否是延迟槽指令

NPC

模块定义

信号名	方向	描述
PC[31:0]	I	32位输入信号，当前指令的地址
NPCop[2:0]	I	两位控制信号，控制NPC的的值 000：PC←PC+4 001：PC←$PC_{31...28}$||instr_index||0_2 010：PC←PC + 4 + sign_extend(offset||0_2) 011：PC←GPR[rs]
instr_index[26:0]	I	26位输入信号，用于PC的计算
offset[16:0]	I	16位输入信号，PC的偏移量
Reg[32:0]	I	32位输入信号，用于寄存器地址的跳转
Judge	I	一位输入，作为跳转的判断依据
Req	I	中断或异常信号

信号名	方向	描述
D_eret	I	D级是否是eret指令
EPC[31:0]	I	32位输入，异常处理返回地址
NPC[31:0]	O	32位输出，输出下一条指令的地址
BD	O	1位输出，是否是延迟槽指令

功能定义

序号	功能名称	描述
1	计算PC的下一个值	Req优先级最高，Req==1，进入异常处理程序 D_eret次高，异常处理指令结束之后 返回三位控制信号，控制NPC的的值 00：PC←PC+4 01：PC←$PC_{31...28}$||instr_index||0_2 10： 结合equal,n_equal,less,big,big or equal,less or equal判断是否需要跳转 PC←PC + 4 + sign_extend(offset||0_2) 11：PC←GPR[rs]
2	branch指令是否跳转判断	Judge=1，跳转 Judge=0，不跳转

IF_ID

模块定义

信号名	方向	描述
clk	I	时钟信号
reset	I	同步复位信号
Req	I	中断或异常请求
IF_ID_en	I	寄存器写入控制信号
PC_F[31:0]	I	32位输入信号，F级的PC
Instr_F[31:0]	I	32位输入信号，F级的指令
BD_F	I	1位输入，F级指令是否是延迟槽指令
Exc_Code_F[4:0]	I	5位输入，F级异常信号
PC_D[31:0]	O	32位输出信号，D级的PC
Instr_D[31:0]	O	32位输出，D级的指令

信号名	方向	描述
BD_D	O	1位输出，D级指令是否是延迟槽指令
Exc_Code_D[4:0]	O	5位输出，D级异常

CTRL

模块定义

信号名	方向	描述
OP[5:0]	I	6位输入信号，指令操作码
Func[6:0]	I	6位输入信号，指令的func段
RegDst[1:0]	O	GRFA3输入端控制信号 00 : A3←$Instr_{20...16}$ 01 : A3←$Instr_{15...11}$ 11 : A3←0x1f
RegWrite	O	寄存器写入控制信号 0 : 不能向GRF写入 1 : 可以向GRF写入
EXTop	O	功能选择信号 0：Imm无符号拓展到32位 1：Imm符号位拓展到32位
ALUsrc[1:0]	O	ALUSrcB输入控制信号 00 : SrcB←RD2 01 : SrcB←EXTImm 10 : SrcB←sll指令的s SrcA←RD2
ALUctrl[2:0]	O	3位输出信号，选择ALU的功能 000 : SrcA + SrcB 001 : SrcA - SrcB 010 : A |B 011 : A & B 100 : A>>B 101 : $signed(A)>>>B 110 : A<B 置1 111 : A<<B
Menwrite	O	内存写入控制信号 0 : 不能向DM写入 1 : 可以向DM写入

信号名	方向	描述
MemtoReg[1:0]	O	控制向寄存器的写入数据 00 : WD←ALUResult 01 : WD←RD 10 : WD←$NPC_{31...0}$ 11 : WD←[$ALUResult_{15...0}$||0_{16}]
NPCop[1:0]	O	两位控制信号，控制NPC的的值 00：PC←PC+4 01：PC←$PC_{31...28}$||instr_index||0_2 10：PC←PC + 4 + sign_extend(offset||0_2) 11：PC←GPR[rs]
CMPop[2:0]	O	用于指示进行何种跳转判断 000 : 判断equal 001 : 判断n_equal 010 : 判断less 011 : 判断big 100 : 判断less or equal 101 : 判断big or equal
DMop[1:0]	O	用于lb,sb,lh,sh等操作的拓展 00 : 正常读取，以字为单位 01 : 用于lh或sh Menwrite = 1--->sh Menwrite = 0--->lh 11 : 用于lb或sb Menwrite = 1--->sb Menwrite = 0--->lb
eret_D	O	1位输出，D级指令是否是eret
Exc_RI	O	1位输出，D级指令是否发生RI错误
ALU_Ov	O	1位输出，控制ALU算术溢出检测
ALU_DM_Ov	O	1位输出，控制ALU计算地址溢出检测
load	O	1位输出，当前指令是否是load指令
store	O	1位输出，当前指令是否是store指令
EXLClr	O	1位输出，将CP0的EXL置0
CP0_en	O	1位输出，CP0的写使能

CMP

信号名	方向	描述
RD1[31:0]	I	32位输入，作为要比较的值

信号名	方向	描述
RD2[31:0]	I	32位输入，作为要比较的值
CMPop[2:0]	I	用于指示进行何种跳转判断 000 : 判断equal 001 : 判断n_equal 010 : 判断less 011 : 判断big 100 : 判断less or equal 101 : 判断big or equal
Judge	O	一位输出，作为跳转的判断依据

GRF

模块定义

信号名	方向	描述
clk	I	时钟信号
reset	I	复位信号，将32个寄存器中的值全部清零 1：复位 0：无效
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
A1[4:0]	I	5位地址输入信号，指定32个寄存器中的一个，将其中存储的值读出到RD1
A2[4:0]	I	5位地址输入信号，指定32个寄存器中的一个，将其中存储的值读出到RD2
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写入的目标寄存器
WD[31:0]	I	32位数据输入信号
RD1[31:0]	O	输出A1指定的寄存器的32位数据
RD2[31:0]	O	输出A2指定寄存器中的32位数据

功能定义

序号	功能名称	描述
1	复位	reset信号有效是，所有寄存器存储的数值清零
2	读数据	读出A1，A2地址对应寄存器中所存储的数据到RD1，RD2
3	写数据	当WE有效且时钟上升沿来临时，将WD写入A3对应的寄存器中

EXT

信号名	方向	描述
Imm[15:0]	I	15位输入立即数
EXTop	I	功能选择信号 0：Imm无符号拓展到32位 1：Imm符号位拓展到32位
EXTImm[31:0]	O	32位输出信号，输出Imm拓展之后的数

ID_EX

模块定义

信号名	方向	描述
clk	I	时钟信号
reset	I	同步复位信号
ID_EX_clr	I	阻塞清零信号
PC_D[31:0]	I	32位输入，D级PC
A3_D[4:0]	I	5位输入，待写入寄存器编号
RD1_D[31:0]	I	32位输入，从GRF[A1]读出
RD2_D[31:0]	I	32位输入，从GRF[A2]读出
RD1_Sel_D[1:0]	I	RD1的转发控制信号
RD2_Sel_D[1:0]	I	RD2的转发控制信号
EXTImm_D[31:0]	I	拓展后的三十二位立即数
Instr_D[31:0]	I	32为输入，D级的指令
A2_D[4:0]	I	D级使用的寄存器编号
A1_D[4:0]	I	E级使用的寄存器编号
Req	I	中断或异常请求
BD_D	I	1位输入，D级指令是否是延迟槽指令
Exc_Code_D[4:0]	I	4位输入，D级指令错误
BD_E	O	1位输出，E级指令是否是延迟槽指令
Exc_Code_E[4:0]	O	4位输出，E级指令错误
A2_E[4:0]	O	D级使用的寄存器编号
A1_E[4:0]	O	E级使用的寄存器编号

信号名	方向	描述
Instr_E[31:0]	O	32为输出，E级的指令
A3_E[4:0]	O	5位输出，待写入寄存器编号
PC_E[31:0]	O	32位输出，E级PC
RD1_E[31:0]	O	32位输出，从GRF[A2]读出
RD2_E[31:0]	O	32位输出，从GRF[A2]读出
EXTImm_E[31:0]	O	拓展后的三十二位立即数
RD1_Sel_D_reg[1:0]	O	RD1的转发控制信号
RD2_Sel_D_reg[1:0]	O	RD2的转发控制信号

ALU

模块定义

信号名	方向	描述
SrcA[31:0]	I	32位输入信号，第一个操作数A
SrcB[31:0]	I	32位输入信号，第二个操作数B
s[4:0]	I	sll可能会用到的信号
ALUControl[2:0]	I	3位输入信号，选择ALU的功能 000 : SrcA + SrcB 001 : SrcA - SrcB 010 : A | B 011 : A & B 100 : 101 : 110 :
ALUResult[31:0]	O	32位输出信号，输出运算结果

MDU

模块定义

信号名	方向	描述
clk	I	时钟信号
reset	I	复位信号
SrcA[31:0]	I	32位输入信号，待操作数
SrcB[31:0]	I	32位输入信号，待操作数

信号名	方向	描述
Start	I	乘除指令开始信号
MDUclr	I	MDU清空信号
MDUop[2:0]	I	控制MDU的指令执行
Req	I	中断或异常请求
HI[31:0]	O	32位输出，高三十二位寄存器
LO[31:0]	O	32位输出，低三十二位寄存器
MDUout[31:0]	O	32位输出，MDU的输出
Busy	O	Busy信号，MDU正在工作

功能定义

序号	功能名称	描述
1	复位	reset信号有效时，将HI，LO，HI_tmp，LO_tmp全部清空
2	清空数据	MUDclr信号有效时，将HI，LO，HI_tmp，LO_tmp全部清空
3	乘除法相关指令	`define mult 3'b000 `define multu 3'b001 `define div 3'b010 `define divu 3'b011 `define mfhi 3'b100 `define mflo 3'b101 `define mthi 3'b110 `define mtlo 3'b111

EX_DM

信号名	方向	描述
clk	I	时钟信号
reset	I	同步复位信号
PC_E[31:0]	I	32位输入，E级PC
A3_E[4:0]	I	5位输入，待写入寄存器编号
Instr_E[31:0]	I	32位输入，E级的指令
RD2_E[31:0]	I	32位输入，从GRF[A2]读出
ALUresult_E[31:0]	I	32位输入，从ALU读出
A2_E[4:0]	I	5位输入，待写入WD的寄存器编号
Req	I	中断或异常请求

信号名	方向	描述
BD_E	I	1位输入，E级指令是否是延迟槽指令
Exc_Code_E[4:0]	I	4位输入，E级指令错误
BD_M	O	1位输出，M级指令是否是延迟槽指令
Exc_Code_M[4:0]	O	4位输出，M级指令错误
A2_M[4:0]	O	5位输出，待写入WD的寄存器编号
A3_M[4:0]	O	5位输出，待写入寄存器编号
PC_M[31:0]	O	32位输出，M级PC
Instr_M[31:0]	O	32位输出，M级指令
RD2_M[31:0]	O	32位输出，待写入DM的WD端
ALUresult_M[31:0]	O	32位输出

BE

模块定义

信号名	方向	描述
A[31:0]	I	32位输入，待写入的数据的地址
WD[31:0]	I	32位输入，待写入的数据
BEop[1:0]	I	2位控制信号
m_data_rdata[31:0]	I	32位输出信号，从地址中读出的数据
WE	I	写使能信号
store	I	是否是store指令
DM_Ov	I	是否计算地址溢出
Exc_AdES_M	O	1位输出信号，是否发生AdES错误
m_data_byteen[3:0]	O	4位输出信号，对外置存储器的使能信号
m_data_wdata[31:0]	O	32位输出信号，最终写入内存的数据

功能定义

序号	功能名称	描述

序号	功能名称	描述
1	控制写入数据种类	m_data_wdata = (BEop==2'b00) ? WD: (BEop==2'b01&&A[1]==1'b0) ? {RD24_31,RD16_23,WD8_15,WD0_7}: (BEop==2'b01&&A[1]==1'b1) ? {WD8_15,WD0_7,RD8_15,RD0_7}: (BEop==2'b10&&A[1:0]==2'b00)? {RD24_31,RD16_23,RD8_15,WD0_7}: (BEop==2'b10&&A[1:0]==2'b01)? {RD24_31,RD16_23,WD0_7,RD0_7}: (BEop==2'b10&&A[1:0]==2'b10)? {RD24_31,WD0_7,RD8_15,RD0_7}: (BEop==2'b10&&A[1:0]==2'b11)? {WD0_7,RD16_23,RD8_15,RD0_7} : 32'b0;
2	控制使能信号	m_data_byteen = (BEop==2'b00) ? 4'b1111: (BEop==2'b01&&A[1]==1'b0) ? 4'b0011: (BEop==2'b01&&A[1]==1'b1) ? 4'b1100: (BEop==2'b10&&A[1:0]==2'b00)? 4'b0001: (BEop==2'b10&&A[1:0]==2'b01)? 4'b0010: (BEop==2'b10&&A[1:0]==2'b10)? 4'b0100: (BEop==2'b10&&A[1:0]==2'b11)? 4'b1000: 4'b0000;
3	检测AdES异常	

DE

模块定义

信号名	方向	描述
A[31:0]	I	32位输入，待读出的数据的地址
Din[31:0]	I	32位输入，从地址中读出的数据
DEop[2:0]	I	3位控制信号
Dout[31:0]	I	32位输出信号，最终读取的数据
load	I	是否是load指令
DM_Ov	I	是否计算地址溢出
Exc_AdEL_M	O	1位输出信号，是否发生AdEL错误

功能定义

序号	功能名称	描述

序号	功能名称	描述
1	控制读取数据种类	Dout = (DEop == 3'b000)? Din : (DEop == 3'b001)? unsigned_byte: (DEop == 3'b010)? signed_byte : (DEop == 3'b011)? unsigned_half : (DEop == 3'b100)? signed_half : 32'b0;
2	检测AdEL异常	

CP0

模块定义

信号名	方向	描述
clk	I	时钟信号
reset	I	同步复位信号
en	I	CP0寄存器写使能信号
CP0Add[4:0]	I	CP0寄存器地址
CP0In[31:0]	I	CP0写入数据
VPC[31:0]	I	受害PC
BDIn	I	当期指令是否是延迟遭指令
ExcCodeIn[4:0]	I	当前指令的错误信息
HWInt[5:0]	I	外部中断情况
EXLClr	I	EXL复位信号
CP0Out[31:0]	O	CP0输出信号
EPCOut[31:0]	O	EPC的值
Req	O	中断或异常请求
Intrespon	O	中断响应信号

DM_WB

模块定义

信号名	方向	描述
clk	I	时钟信号
reset	I	同步复位信号
PC_M[31:0]	I	32位输入，M级PC

信号名	方向	描述
A3_M[4:0]	I	5位输入，待写入寄存器编号
Instr_M[31:0]	I	32位输入，M级的指令
RD_M[31:0]	I	32位输入，DM的输出端
PC_WB[31:0]	O	32位输出，WB级PC
ALUresult_M[31:0]	I	32位输入，从ALU读出
Req	I	中断或者异常请求
CP0out_M[31:0]	I	32位输入，CP0寄存器中读出的值
CP0out_WB[31:0]	O	32位输出，CP0寄存器中读出的值
A3_WB[4:0]	O	5位输出，待写入寄存器编号
RD_WB[31:0]	O	32位输出，DM的输出端的值
Instr_WB[31:0]	O	32位输出，WB级指令
ALUresult_WB[31:0]	O	32位输出

HAZARD

模块定义

信号名	方向	描述
clk	I	时钟信号
Instr_D[31:0]	I	D级指令
Instr_E[31:0]	I	E级指令
Regwrite_E	I	E级寄存器写使能
Regwrite_M	I	M级寄存器写使能
D_eret	I	D级是否是eret指令
PCwrite	O	PC写使能
IF_ID_en	O	IF_ID寄存器写使能
ID_EX_clr	O	ID_EX寄存器清零
Num_use_rs_D[4:0]	O	D指令待使用的rs
Num_use_rt_D[4:0]	O	D指令待使用的rt
Tnew_E_[1:0]	O	D指令待使用rs时间
Tnew_M_[1:0]	O	D指令待使用rt时间

信号名	方向	描述
MDUclr	O	MDU清空信号

HAZARD_E

模块定义

信号名	方向	描述
Instr_E[31:0]	I	E级指令
Tnew_E[1:0]	O	E级指令的Tnew
Num_new_E[4:0]	O	E级指令待修改寄存器编号

HAZARD_M

模块定义

信号名	方向	描述
Instr_E[31:0]	I	E级指令
Tnew_E[1:0]	I	E级指令的Tnew
Num_new_E[4:0]	I	E级指令待修改寄存器编号
Tnew_M	O	M级指令的Tnew
Num_new_M	O	M级指令待修改寄存器编号

TC

模块定义

信号名	方向	描述
clk	I	时钟信号
reset	I	同步复位信号
Addr[31:2]	I	地址信号
WE	I	写使能信号
Din[31:0]	I	写入信号
Dout[31:0]	O	TC输出信号
IRQ[31:0]	O	中断请求

Bridge

模块定义

信号名	方向	描述
interrupt	I	中断信号
Intrespon	I	中断响应信号
m_data_addr[31:0]	O	待写入外部地址
m_data_rdata[31:0]	I	从外部读进来的数据
m_data_wdata[31:0]	O	待写入外部数据
m_data_byteen[3:0]	O	写使能信号
m_data_addr_cpu[31:0]	I	CPU的地址
m_data_rdata_cpu[31:0]	O	送给CPU的数据
m_data_wdata_cpu[31:0]	I	CPU要写出的数据
m_data_byteen_cpu[3:0]	I	CPU的写使能
m_int_addr[31:0]	O	中断响应地址
m_int_byteen[3:0]	O	中断响应写使能
TC0_Dout[31:0]	I	计时器的输出
TC0_Addr[31:2]	O	计时器写入地址
TC0_WE	O	计时器写使能信号
TC0_Din[31:0]	O	计时器待写入数据
TC1_Dout[31:0]	I	计时器的输出
TC1_Addr[31:0]	O	计时器写入地址
TC1_WE	O	计时器写使能信号
TC1_Din[31:0]	O	计时器待写入数据

(三)思考题

1、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?

鼠标和键盘产生中断信号，进入中断处理区的对应位置，将输入信号从鼠标和键盘中读入寄存器。

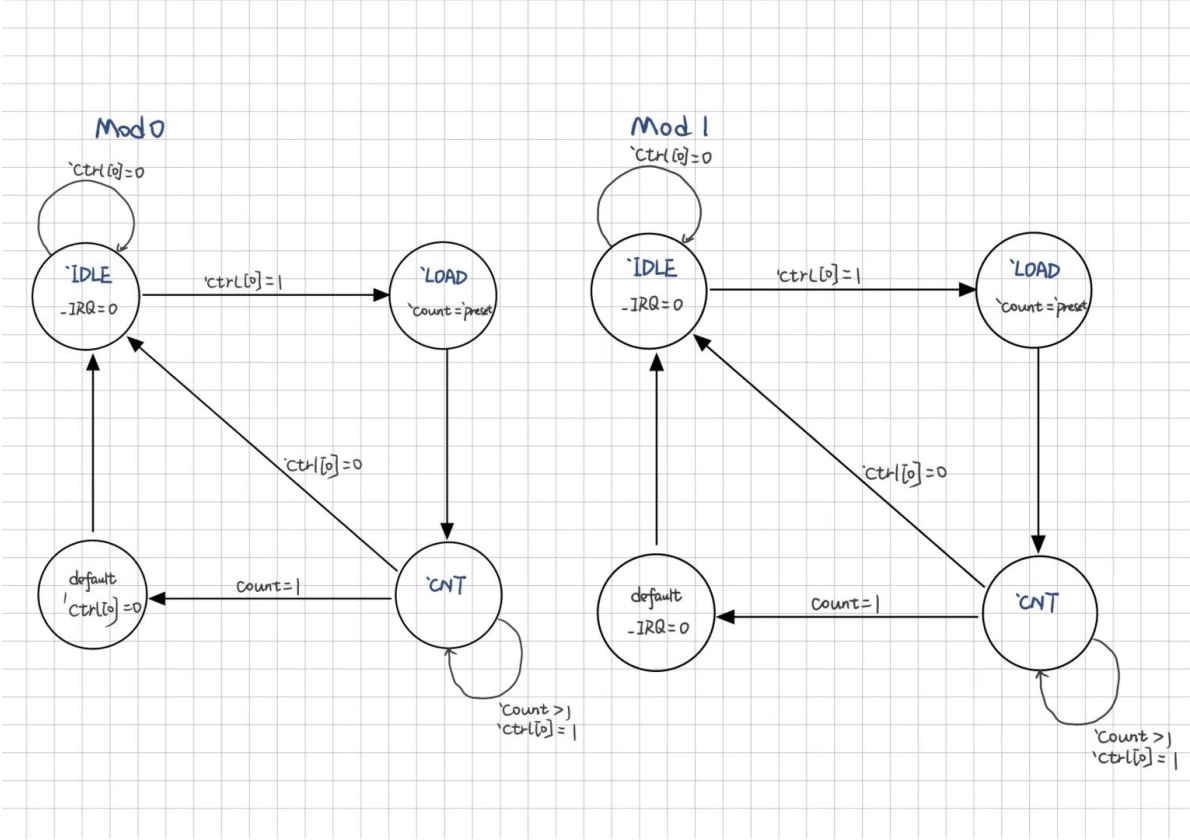
2、请思考为什么我们的 CPU 处理中断异常必须是已经指定好的地址？如果你的 CPU 支持用户自定义入口地址，即处理中断异常的程序由用户提供，其还能提供我们所希望的功能吗？如果可以，请说明这样可能会出现什么问题？否则举例说明。（假设用户提供的中断处理程序合法）

因为中断异常处理程序是在程序所有程序运行前就已经写好了，设置固定的地址，就能保证异常或者中断来临的时候呢能够跳转到正确的地址进行处理。如果由用户提供中断异常处理程序的话，它跳转的地址也是计算出来的，但是如果在算跳转地址的时候出现了错误，那就无法正常进行异常处理行为。

3、为何与外设通信需要 Bridge?

外设的种类是无穷无尽的，而 CPU 的指令集却是有限的。我们并不能总是因为新加入了一个外设，就专门为这个外设增加新的 CPU 指令。所以需要一个系统桥去连接CPU和外设，对外只留出简洁的接口，不显示其他细节，让CPU访问外设只需通过地址就可以，这样也是体现了"高内聚，低耦合"的原则。

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移图。



5、倘若中断信号流入的时候，在检测宏观 PC 的一级如果是一条空泡（你的 CPU 该级所有信息均为空）指令，此时会发生什么问题？在此例基础上请思考：在 P7 中，清空流水线产生的空泡指令应该保留原指令的哪些信息？

可能会出现宏观PC的错误，而且延迟槽标记信号也会出现问题。如果是中断或者异常指令造成的流水线清空，应该保持原有的PC值，以保证宏观PC的正确。如果是阻塞造成的DE级流水线清除，应该要保持原有的PC并且保持原有的BD标志信号。

6、为什么 `jalr` 指令为什么不能写成 `jalr $31, $31`?

```
Register specifiers rs and rd must not be equal, because such an instruction does
not have the same effect when reexecuted. The result of executing such an
instruction is UNPREDICTABLE. This restriction permits an exception handler to
resume execution by re-executing the branch when an exception occurs in the
branch delay slot.
```

指令集要求。寄存器说明符 rs 和 rd 不得相等，因为此类指令在重新执行时不具有相同的效果。执行此类指令的结果是不可预测的。此限制允许异常处理程序在分支延迟槽中发生异常时通过重新执行分支来恢复执行。