32 位 CPU 计软联合实验组

设计文档

Totoro 组

杨乐 琚锡廷 董原良 李林涵

2019 年 1 月

1 文档说明

本文档是 Totoro 组 32 位 MIPS CPU 的设计文档，首先先介绍 CPU 总体的模块划分，再分别介绍各个模块的大体实现和部分设计细节。每个模块在介绍时均将遵守以下顺序：

1. **简介**：简要介绍模块功能；
2. **接口定义**：介绍模块的接口；
3. **实现细节**：介绍模块具体相关的实现细节。

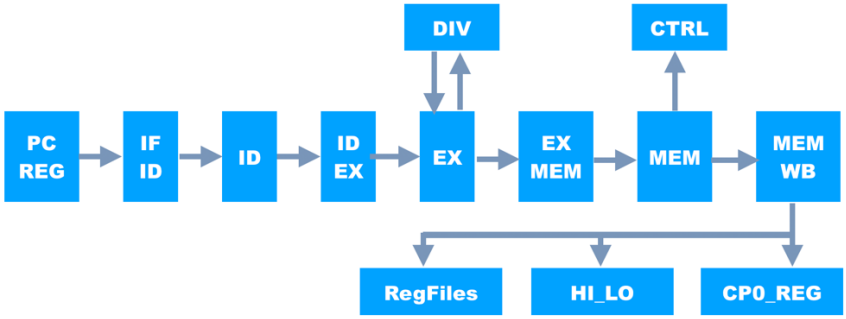
2 总览

我们小组的 CPU 整体设计如下：整体分为五级流水线、访存、与外设管理三部分。

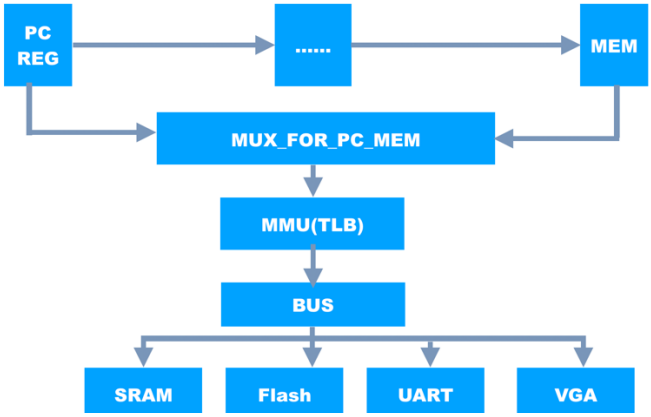
流水线部分的设计与课程讲的基本相似，这里我们由于需要用到除法，故在 EX 模块外接了一个 Div 模块单独处理除法；并且在 MEM 阶段外界了一个 Ctrl 模块用于处理异常与中断，可以控制流水线的暂停。

流水线与访存通过 MUX_FOR_PC_MEM 和 MemControl 来连接，流水线向其传送 CPU 对于内存的访问信息，访存部分则返回从内存中取出的数据信息。

访存部分与外设部分会在下面分别仔细叙述；值得一提的是我们加入了总线(BUS)，可以控制四种外设设备的输入。



（流水线部分）



（访存和外设部分）

3 PC_Reg

3.1 简介

PC_Reg，PC 寄存器，用于存放当前指令执行地址的寄存器，同时用于对于下一条执行的指令地址的选择，在时钟上升沿跟据控制信息变更 PC 的值，属于时序电路。

3.2 接口定义

信号名称	信号规格	输入/输出	信号来源/去处	详细描述
clk	wire	input	电路板	时钟信号
rst	wire	input	电路板	复位信号
stall	wire[5:0]	input	Ctrl	流水线暂停的信号
flush	wire	input	Ctrl	由异常引起的流水线清洗的信号
new_pc	wire[31:0]	input	Ctrl	异常发生时 pc 需要跳转到的位置，flush 为 1 时用
jr_o	wire	input	ID	下一条指令是否是跳转指令
jr_addr	wire[31:0]	input	ID	跳转指令的目标地址
pc	reg[31:0]	output	mux_for_pc_mem	下一条指令地址，用于取指
ce	reg	output	mux_for_pc_mem	IF 段使能信号

3.3 设计细节

PC_Reg 即五级流水线中的取指段，负责向 MMU 模块给出下一条取指的地址。下一条指令 PC 的变化主要有以下几种情况：

1. 正常+4
2. 跳转到跳转指令的目标地址(jr_o 信号为 1)
3. 流水线暂停，PC 不变(stall 信号的第 0 位为 1)
4. 有异常发生，需要清空其他控制信息，同时 PC 跳转到异常处理例程入口处(flush 信号为 1)

判断的优先级为：4 > 3 > 2 > 1

将按照优先级依次判断，并向 Mux 模块传递下一条需要取指的指令地址。

4 IF_ID

4.1 简介

五级流水线中 IF 段和 ID 段间的触发器，在时钟上升沿跟据控制信息将 IF 段的信息传至 ID 段，属于时序逻辑电路。

4.2 接口定义

信号名称	信号规格	输入/输出	信号来源/去处	详细描述
clk	wire	input	电路板	时钟信号
rst	wire	input	电路板	复位信号
stall	wire[5:0]	input	Ctrl	流水线暂停的信号
flush	wire	input	Ctrl	由异常引起的流水线清洗的信号
if_pc	wire[31:0]	input	pc_reg	当前指令地址
if_inst	wire[31:0]	input	mux_for_pc_mem	当前指令
if_int	wire[5:0]	input	外层模块	是否有中断信息
if_instmiss	wire	input	mux_for_pc_mem	取指是否产生 miss 错误
if_instinvalid	wire	input	mux_for_pc_mem	取指是否产生 invalid 错误
if_instpermissionDenied	wire	input	mux_for_pc_mem	取指是否产生 permissionDenied 错误
id_pc	reg[31:0]	output	ID	指令地址
id_inst	reg[31:0]	output	ID	指令地址
id_int	reg[5:0]	output	ID	是否有中断信息
id_instmiss	reg	output	ID	取指是否产生 miss 错误
id_instinvalid	reg	output	ID	取指是否产生 invalid 错误
id_instpermissionDenied	reg	output	ID	取指是否产生 permissionDenied 错误
id_flush	reg	output	ID	流水线此时是否被清空

4.3 设计细节

IF_ID，位于五级流水线 IF（取指）与 ID（译码）段间的触发器，根据控制信息(flash 和 stall)来决定向译码段传入什么信息，有四种情况：

1.流水线正常运转

2.流水线译码段暂停(stall[1]和 stall[2]均为 1)

3.流水线暂停，向译码段插入气泡(stall[1]为 1，stall[2]为 0)

4.异常处理，流水线清空(flash 信号为 1)

优先级为：4 > 3 > 2 > 1

按照优先级判断向 ID（译码）段传递相关信息

instmiss，instinvalid，instpermissionDenied，int_i，id_flush 与异常处理有关，沿流水线传至 mem 段处理

5 ID

5.1 简介

译码段，对指令进行译码，并向 ID_EX 触发器传递译码结果，组合逻辑电路。

5.2 接口定义

信号名称	信号规格	输入/输出	信号来源/去处	详细描述
rst	wire	input	电路板	复位信号
pc_i	wire[31:0]	input	IF_ID	当前指令地址
inst_i	wire[31:0]	input	IF_ID	当前指令
int_i	wire[5:0]	input	IF_ID	中断信息
instmiss_i	wire	input	IF_ID	异常处理相关
instinvalid_i	wire	input	IF_ID	异常处理相关
Instpermission Denied_i	wire	input	IF_ID	异常处理相关
flush_i	wire	input	IF_ID	异常处理相关
is_in_delayslot_i	wire	input	ID_EX	当前指令是否在延迟槽内
reg1_data_i	wire[31:0]	input	Regfiles	当前指令需要用的寄存器 1 的值
reg2_data_i	wire[31:0]	input	Regfiles	当前指令需要用的寄存器 2 的值
ex_wreg_i	wire	input	EX	EX 段的数据旁路
ex_wd_i	wire[4:0]	input	EX	EX 段的数据旁路
ex_wdata_i	wire[31:0]	input	EX	EX 段的数据旁路
mem_wreg_i	wire	input	MEM	MEM 段的数据旁路
mem_wd_i	wire[4:0]	input	MEM	MEM 段的数据旁路
mem_wdata_i	wire[31:0]	input	MEM	MEM 段的数据旁路
int_o	wire[5:0]	output	ID_EX	异常处理相关
instmiss_o	wire	output	ID_EX	异常处理相关
instinvalid_o	wire	output	ID_EX	异常处理相关
Instpermission Denied_o	wire	output	ID_EX	异常处理相关
flush_o	wire	output	ID_EX	异常处理相关
reg1_re_o	reg	output	Regfiles	当前指令是否需要读取寄存器 1
reg1_addr_o	reg[4:0]	output	Regfiles	当前指令需要读取的寄存器 1 的地址
reg2_re_o	reg	output	Regfiles	当前指令是否需要读取寄存器 2

reg2_addr_o	reg[4:0]	output	Regfiles	当前指令需要读取的寄存器 2 的地址
reg1_o	reg[31:0]	output	ID_EX	当前指令读取的寄存器 1 数据
reg2_o	reg[31:0]	output	ID_EX	当前指令读取的寄存器 2 数据
jr_o	reg	output	pc_reg	下个周期是否需要跳转
jr_addr	reg[31:0]	output	pc_reg	下个周期的跳转目标地址
next_in_delayslot	reg	output	ID_EX	当前指令的下一条指令是否在延迟槽内
is_in_delayslot_o	reg	output	ID_EX	当前指令是否在延迟槽内
ret_addr	reg[31:0]	output	ID_EX	JAL 和 JALR 指令要求的存入寄存器的地址值
pc_o	reg[31:0]	output	ID_EX	当前指令地址
inst_o	reg[31:0]	output	ID_EX	当前指令
wreg_o	reg	output	ID_EX	当前指令是否需要写寄存器
wd_o	reg[4:0]	output	ID_EX	写寄存器的地址
excepttype_o	wire[31:0]	output	ID_EX	异常处理相关
alu_op_o	reg[7:0]	output	ID_EX	ALU 的具体操作码
alu_sel_o	reg[2:0]	output	ID_EX	ALU 的操作类型

5.3 设计细节

译码阶段主要对当前指令进行译码，在这一阶段解析指令的类型，给出 ALU 的操作类型和具体操作码，从寄存器堆读出指令需要读的寄存器值，并将这些信息沿流水器传给 EX 段。

5.3.1 分支跳转

当有分支跳转指令时会发生控制冒险，解决冲突正常需要暂停流水线或进行分支预测，这里并没有采用那些方法，而是仅仅记录下在延迟槽中的指令，要求软件遍程时实现延迟中指令为空的要求。

需要尽早判断跳转，所以在译码阶段即判断，这样延迟槽中就仅有一条指令。若当前指令是跳转指令，则传递一个 next_in_delayslot 给 ID_EX 段，并在 ID_EX 段回传到下一条进入 ID 段的指令，同时向 pc_reg 段传递跳转信息及目标地址。即仅对当前指令是否在延迟槽中作记录，在异常处理时会用到这个信息。

5.3.2 数据旁路

正常情况下，指令的执行结果在 WB 阶段才会写回到寄存器中，所以如果有连续两条指令影响了同一个寄存器，不做任何处理会发生数据冒险，使得后一条指令无法取到正常指令的值。在这里采用的处理是使用数据旁路，将流水线上处于 EX 和 MEM 段的处理结果回发

到 ID 阶段，使得 ID 段能取到正确的值。

仅采用数据旁路时，仍有一种数据冒险无法解决：load-使用型冒险，前一条是 load 指令，后一条指令要用到 load 指令存入的寄存器。load 指令的结果值在 MEM 段才能得到，这时下一条指令已经进入 EX 段了，所以正常旁路无法处理。需要暂停流水线，具体在 EX 段介绍。

5.3.3 异常判断

在 ID 段，仅判断四种类型的异常：

1.无效指令

2.当前指令是否是 syscall 指令

3.当前指令是否是 break 指令

4.当前指令是否是 eret 指令

这些信息随 excepttype_o 传出，到 MEM 段处理。

6 ID_EX

6.1 简介

五级流水线中 ID 段和 EX 段间的触发器，在时钟上升沿跟据控制信息将 ID 段的信息传至 EX 段，属于时序逻辑电路。

6.2 接口定义

信 号 名 称	信号规格	输入/输出	信号来源/去处	详细描述
clk	wire	input	电路板	时钟信号
rst	wire	input	电路板	复位信号
stall	wire[5:0]	input	Ctrl	流水线暂停的信号
flush	wire	input	Ctrl	由异常引起的流水线清洗的信号
id_instmiss	wire	input	ID	异常处理相关
id_instinvalid	wire	input	ID	异常处理相关
id_instpermissionDenied	wire	input	ID	异常处理相关
id_int	wire[5:0]	input	ID	异常处理相关
id_flush	wire	input	ID	异常处理相关
id_excepttype	wire[31:0]	input	ID	ID 段发生异常情况
id_pc	wire[31:0]	input	ID	当前指令地址
id_inst	wire[31:0]	input	ID	当前指令
id_reg1	wire[31:0]	input	ID	当前指令需用的寄存器 1 的值
id_reg2	wire[31:0]	input	ID	当前指令需用的寄存器 2 的值
id_wd	wire[4:0]	input	ID	当前指令所写寄存器地址
id_wreg	wire	input	ID	当前指令的寄存器写使能
id_is_in_delayslot	wire	input	ID	当前指令是否在延迟槽内
next_in_delayslot_i	wire	input	ID	当前指令的下一条指令是否在延迟槽内
id_ret_addr	wire[31:0]	input	ID	JAL 或 JALR 指令需要存入寄存器的地址值
id_aluop	wire[7:0]	input	ID	当前指令的 ALU 操作码
id_alusel	wire[2:0]	input	ID	当前指令的 ALU 操作类型

ex_instmiss	reg	output	EX	异常处理相关
ex_instinvalid	reg	output	EX	异常处理相关
ex_instpermissionDenied	reg	output	EX	异常处理相关
ex_int	reg[5:0]	output	EX	异常处理相关
ex_flush	reg	output	EX	异常处理相关
ex_excepttype	reg[31:0]	output	EX	ID 段发生异常情况
ex_pc	reg[31:0]	output	EX	当前指令地址
ex_inst	reg[31:0]	output	EX	当前指令
ex_reg1	reg[31:0]	output	EX	当前指令需用的寄存器 1
ex_reg2	reg[31:0]	output	EX	当前指令需用的寄存器 2
ex_wreg	reg	output	EX	当前指令的寄存器写使能
ex_wd	reg[4:0]	output	EX	当前指令所写寄存器地址
ex_is_in_delayslot	reg	output	EX	当前指令是否在延迟槽内
is_in_delayslot_o	reg	output	ID	当前指令的下一条指令是否在延迟槽内
ex_ret_addr	reg[31:0]	output	EX	JAL 或 JALR 指令需要存入寄存器的地址值
ex_aluop	reg[7:0]	output	EX	当前指令的 ALU 操作码
ex_alusel	reg[2:0]	output	EX	当前指令的 ALU 操作类型

6.3 设计细节

ID_EX，位于五级流水线 ID 与 EX 段间的触发器，根据控制信息(flash 和 stall)来决定向 EX 段传入什么信息，有四种情况：

1. 流水线正常运转
2. 流水线译码段暂停(stall[2]和 stall[3]均为 1)
3. 流水线暂停，向译码段插入气泡(stall[2]为 1，stall[3]为 0)
4. 异常处理，流水线清空(flash 信号为 1)

优先级为：4 > 3 > 2 > 1

按照优先级判断向 EX 段传递相关信息

instmiss，instinvalid，instpermissionDenied，int_i，id_flush 与异常处理有关，沿流水线传至 mem 段处理

7 EX

7.1 简介

EX 段，主要是 ALU，按照指令类型进行运算，给出运算结果并传给 mem 段，属于组合逻辑电路。

7.2 接口定义

信号名称	信号规格	输入/输出	信号来源/去处	详细描述
rst	wire	input	电路板	复位信号
instmiss_i	wire	input	ID_EX	异常处理相关
instinvalid_i	wire	input	ID_EX	异常处理相关
Instper Mission Denied_i	wire	input	ID_EX	异常处理相关
int_i	wire[5:0]	input	ID_EX	异常处理相关
flush_i	wire	input	ID_EX	异常处理相关
wreg_i	wire	input	ID_EX	当前指令的寄存器写使能
wd_i	wire[4:0]	input	ID_EX	当前指令的寄存器写地址
aluop_i	wire[7:0]	input	ID_EX	当前指令的 ALU 操作码
alusel_i	wire[2:0]	input	ID_EX	当前指令的 ALU 操作类型
reg1_i	wire[31:0]	input	ID_EX	当前指令的寄存器 1 的值
reg2_i	wire[31:0]	input	ID_EX	当前指令的寄存器2的值
hi_i	wire[31:0]	input	hilo_reg	HI 寄存器值
lo_i	wire[31:0]	input	hilo_reg	LO 寄存器值
mem_whio	wire	input	MEM	MEM 段旁路
mem_hi_i	wire[31:0]	input	MEM	MEM 段旁路
mem_lo_i	wire[31:0]	input	MEM	MEM 段旁路
wb_whio	wire	input	MEM_WB	WB 段旁路
wb_hi_i	wire[31:0]	input	MEM_WB	WB 段旁路
wb_lo_i	wire[31:0]	input	MEM_WB	WB 段旁路
cp0_data_i	wire[31:0]	input	cp0_reg	从 cp0 读取的寄存器值
mem_cp0_we	wire	input	MEM	MEM 段旁路
mem_cp0_addr	wire[4:0]	input	MEM	MEM 段旁路
mem_cp0_data	wire[31:0]	input	MEM	MEM 段旁路
wb_cp0_we	wire	input	MEM_WB	WB 段旁路

wb_cp0_addr	wire[4:0]	input	MEM_WB	WB 段旁路
wb_cp0_data	wire[31:0]	input	MEM_WB	WB 段旁路
pc_i	wire[31:0]	input	ID_EX	当前指令地址
inst_i	wire[31:0]	input	ID_EX	当前指令
ret_addr_i	wire[31:0]	input	ID_EX	JAL 或 JALR 指令要存入寄存器的地址值
is_in_delayslot_i	wire	input	ID_EX	当前指令是否在延迟槽内
excepttype_i	wire[31:0]	input	ID_EX	ID 段的异常情况
div_end	wire	input	Div	除法是否结束
divres	wire[63:0]	input	Div	除法结果
instmiss_o	wire	output	EX_MEM	异常处理相关
instinvalid_o	wire	output	EX_MEM	异常处理相关
Instpermission Denied_o	wire	output	EX_MEM	异常处理相关
int_o	wire[5:0]	output	EX_MEM	异常处理相关
flush_o	wire	output	EX_MEM	异常处理相关
wreg_o	reg	output	EX_MEM	寄存器写使能
wd_o	reg[4:0]	output	EX_MEM	寄存器写地址
wdata_o	reg[31:0]	output	EX_MEM	写入寄存器的值
whio	reg	output	EX_MEM	HI/LO 写使能
hi_o	reg[31:0]	output	EX_MEM	写入 HI 的值
lo_o	reg[31:0]	output	EX_MEM	写入 LO 的值
cp0_we_o	reg	output	EX_MEM	cp0 写使能
cp0_waddr_o	reg[4:0]	output	EX_MEM	cp0 写地址
cp0_wdata_o	reg[31:0]	output	EX_MEM	cp0 写数据
cp0_rsel_o	reg	output	cp0_reg	cp0 读使能
cp0_raddr_o	reg[4:0]	output	cp0_reg	cp0 读地址
div_start	reg	output	Div	是否除法
div_data1	reg[31:0]	output	Div	被除数
div_data2	reg[31:0]	output	Div	除数
mem_ce_o	reg	output	EX_MEM	访存使能信号
mem_op_o	reg[4:0]	output	EX_MEM	访存操作码
mem_addr_o	reg[31:0]	output	EX_MEM	访存地址
mem_data_o	reg[31:0]	output	EX_MEM	写入内存数据
is_in_delayslot_o	reg	output	EX_MEM	当前指令是否在延迟槽中
pc_o	reg[31:0]	output	EX_MEM	当前指令地址
excepttype_o	reg[31:0]	output	EX_MEM	异常发生情况
stall	reg	output	Ctrl	除法暂停流水线信号
idstall	reg	output	Ctrl	访存暂停流水线信号

7.3 设计细节

EX 段主要负责 ALU 的数据运算以及将数据沿流水线继续往下传递，除法指令和访存指令引起的流水线暂停信号也由 EX 段发出。

7.3.1 ALU

ALU 包含的运算类型和具体运算码如下表所示：

运算类型	具体操作码	功能
逻辑运算	ALU_AND_OP	与
	ALU_OR_OP	或
	ALU_XOR_OP	异或
	ALU_LUI_OP	存入高 16 位
	ALU_NOR_OP	或非
移位运算	ALU_SLL_OP	左移
	ALU_SRL_OP	逻辑右移
	ALU_SRA_OP	算术右移
算术运算	ALU_ADD_OP	有符号数加（存在溢出）
	ALU_ADDU_OP	无符号数加
	ALU_SUBU_OP	无符号数减
	ALU_SLT_OP	有符号数比较
	ALU_SLTU_OP	无符号数比较
	ALU_MULT_OP	有符号数乘，结果存于 HI/LO
	ALU_DIVU_OP	无符号数除，结果存于 HI/LO
传递运算	ALU_MFC0_OP	传递 cp0 数据
	ALU_MFHI_OP	传递 HI 数据
	ALU_MFLO_OP	传递 LO 数据
跳转指令相关		JAL,JALR 要求存入的地址值

7.3.2 异常检测

在 EX 段仅检测算术溢出异常，传递至 MEM 段处理

7.3.3 暂停流水线信号

除法和访存指令需要暂停流水线，暂停信号均由 EX 段发出，传至 Ctrl 段，与 flush 作优先级比较后再由 Ctrl 传递至各触发器。

访存指令，存在此前 ID 段提到过的，数据旁路无法解决的数据冒险，还存在与 pc_reg 之间的结构冒险（均需要访存），这两种冒险插入一个暂停信号可同时解决。

解决方法为：当 EX 段检测出当前指令为访存指令时，在下一个时钟周期向 EX 段插入一个气泡，IF 和 ID 段暂停。这样操作，下一个周期，该指令进入 MEM 段，pc 传出的值不变，无取指操作，向内存传递 MEM 段信号访存即可，控制冒险解决；该指令的下一条指令仍停留在 ID 段，而该指令已到 MEM 段，可获得该指令结果，由旁路正常回送给 ID 段的下一条指令，数据冒险解决。

除法指令，由于除法指令需要运行不止一个周期，需要需要暂停流水线。在 EX 段检测出当前指令为除法时，将相关数值传给 Div 模块，同时传递流水线暂停信号，从下一个周期

开始向 MEM 段插入气泡，前三个阶段暂停等待除法的结果返回，当收到 Div 模块的除法结束信号时，取消暂停并将结果沿流水线向下传递到 WB 阶段回写到 HI/LO 寄存器。

8 Div

8.1 简介

除法器，采用试商法，为一个状态机，时序逻辑电路。

8.2 接口定义

信号名称	信号规格	输入/输出	信号来源/去处	详细描述
clk	wire	input	电路板	时钟信号
rst	wire	input	电路板	复位信号
div	wire	input	EX	除法开始信号
divinterrupt	wire	input	Ctrl	异常处理，除法中断信号
data1	wire[31:0]	input	EX	被除数
data2	wire[31:0]	input	EX	除数
result	reg[63:0]	output	EX	运算结果
fin	reg	output	EX	运算结束信号

8.3 设计细节

Div 模块与 EX 模块进行信息交互，进行无符号数的除法运算，接收 EX 段的除法开始信号和数据，发送结果信号与结果到 EX 段。

内部实现采用试商法，正常运行需要 32 个周期，使用以下状态机。

有 4 个状态：

DivFree：当前无除法操作，处于空闲状态

DivByZero：收到除法操作但除数为 0，将运算机结果为 0

DivOn：正在进行除法操作

DivEnd：除法操作结束，将 fin 置为 1，将结果返还给 EX 段。

4 个状态间的转移情况如下：

DivFree -> DivByZero：收到除法运算，但除数为 0

DivFree -> DivOn：收到除法运算且除数不为 0，初始化部分积和运算数

DivOn -> DivOn：除法正常进行

DivOn -> DivFree：divinterrupt 为 1，发生异常，流水线清空，除法运算直接结束

DivOn -> DivEnd：除法运算正常结束

DivEnd -> DivFree：将 fin 置为 1，返回结果和 fin 到 EX 段

除法采用试商法，过程如下：

部分积 tempres 规模为 reg[65:0]，初始化将 32 位被除数置为 tempres[32:1]位，其余位为 0。

每个周期采用试商，用 tempres[63:32]位减去除数，若为负，部分积[63:32]位不变，左移 1 位，多出最后一位商 0 ；若为非负，部分积[63:32]置为减除数后的值，左移 1 位，多出最后一位商 1。如此进行 32 个周期，最后问部分积[64:33]为余数，[31:0]为商。

9 EX_MEM

9.1 简介

五级流水线中 EX 段和 MEM 段间的触发器，在时钟上升沿跟据控制信息将 EX 段的信息传至 MEM 段，属于时序逻辑电路。

9.2 接口定义

信号名称	信号规格	输入/输出	信号来源/去处	详细描述
clk	wire	input	电路板	时钟信号
rst	wire	input	电路板	复位信号
stall	wire[5:0]	input	Ctrl	流水线暂停的信号
flush	wire	input	Ctrl	由异常引起的流水线清洗的信号
ex_instmiss	wire	input	EX	异常处理相关
ex_instinvalid	wire	input	EX	异常处理相关
ex_instpermissionDenied	wire	input	EX	异常处理相关
ex_int	wire[5:0]	input	EX	异常处理相关
ex_flush	wire	input	EX	异常处理相关
ex_wreg	wire	input	EX	寄存器写使能
ex_wd	wire[4:0]	input	EX	寄存器写地址
ex_wdata	wire[31:0]	input	EX	寄存器写数据
ex_whio	wire	input	EX	HI/LO 写使能
ex_hi	wire[31:0]	input	EX	HI 写数据
ex_lo	wire[31:0]	input	EX	LO 写数据
ex_cp0_we	wire	input	EX	cp0 写使能
ex_cp0_waddr	wire[4:0]	input	EX	cp0 写地址
ex_cp0_wdata	wire[31:0]	input	EX	cp0 写数据
ex_memce	wire	input	EX	访存使能
ex_memop	wire[4:0]	input	EX	访存操作码
ex_memaddr	wire[31:0]	input	EX	访存地址
ex_memdata	wire[31:0]	input	EX	访存写数据
ex_is_in_delayslot	wire	input	EX	当前指令是否在延迟槽内
ex_pc	wire[31:0]	input	EX	当前指令地址
ex_excepttype	wire[31:0]	input	EX	异常发生情况
mem_instmiss	reg	output	MEM	异常处理相关
mem_instinvalid	reg	output	MEM	异常处理相关
mem_instpermissionDenied	reg	output	MEM	异常处理相关
mem_int	reg[5:0]	output	MEM	异常处理相关

mem_flush	reg	output	MEM	异常处理相关
mem_wreg	reg	output	MEM	寄存器写使能
mem_wd	reg[4:0]	output	MEM	寄存器写地址
mem_wdata	reg[31:0]	output	MEM	寄存器写数据
mem_whio	reg	output	MEM	HI/LO 写使能
mem_hi	reg[31:0]	output	MEM	HI 写数据
mem_lo	reg[31:0]	output	MEM	LO 写数据
mem_cp0_we	reg	output	MEM	cp0 写使能
mem_cp0_waddr	reg[4:0]	output	MEM	cp0 写地址
mem_cp0_wdata	reg[31:0]	output	MEM	cp0 写数据
mem_ce	reg	output	MEM	访存使能
mem_op	reg[4:0]	output	MEM	访存操作码
mem_addr	reg[31:0]	output	MEM	访存地址
mem_data	reg[31:0]	output	MEM	访存写数据
mem_is_ in_delayslot	reg	output	MEM	当前指令是否在延迟槽内
mem_pc	reg[31:0]	output	MEM	当前指令地址
mem_excepttype	reg[31:0]	output	MEM	异常发生情况

9.3 设计细节

EX_MEM，位于五级流水线 EX 与 MEM 段间的触发器，根据控制信息(flash 和 stall)来决定向 MEM 段传入什么信息，有四种情况：

1. 流水线正常运转
2. 流水线译码段暂停(stall[3]和 stall[4]均为 1)
3. 流水线暂停，向译码段插入气泡(stall[3]为 1，stall[4]为 0)
4. 异常处理，流水线清空(flash 信号为 1)

优先级为：4 > 3 > 2 > 1

按照优先级判断向 MEM 段传递相关信息

instmiss，instinvalid，instpermissionDenied，int_i，id_flush 与异常处理有关，沿流水线传至 mem 段处理

10 MEM

10.1 简介

五级流水线中的 MEM 段，执行访存操作和异常处理，将异常处理相关信息传给 Ctrl 和 cp0 模块，属于组合逻辑电路。

10.2 接口定义

信号名称	信号规格	输入/输出	信号来源/去处	详细描述
rst	wire	input	电路板	复位信号
wreg_i	wire	input	EX_MEM	寄存器写使能
wd_i	wire[4:0]	input	EX_MEM	寄存器写地址
wdata_i	wire[31:0]	input	EX_MEM	寄存器写数据
whio_i	wire	input	EX_MEM	HI/LO 写使能
hi_i	wire	input	EX_MEM	HI 写数据
lo_i	wire	input	EX_MEM	lo 写数据
cp0_we_i	wire	input	EX_MEM	cp0 写使能
cp0_waddr_i	wire[4:0]	input	EX_MEM	cp0 写地址
cp0_wdata_i	wire[31:0]	input	EX_MEM	cp0 写数据
instmiss	wire	input	EX_MEM	当前指令的取指 miss 情况
instinvalid	wire	input	EX_MEM	当前指令的取指 invalid 情况
Instpermissi onDenied	wire	input	EX_MEM	当前指令取指的 permisionDenied 情况
tlbmiss	wire	input	mux_for_pc_mem	当前指令访存 miss 情况
tlbinvalid	wire	input	mux_for_pc_mem	当前指令访存 invalid 情况
Permission Denied	wire	input	mux_for_pc_mem	当前指令访存的 permissionDenied 情况
pcreset_i	wire	input	EX_MEM	当前是否是插入的气泡
int_i	wire	input	EX_MEM	中断信息
ram_ce_i	wire	input	EX_MEM	访存使能
ram_op_i	wire[4:0]	input	EX_MEM	访存操作码
ram_data_i	wire[31:0]	input	EX_MEM	访存写数据
ram_addr_i	wire[31:0]	input	EX_MEM	访存读/写地址
ld_data_i	wire[31:0]	input	mux_for_pc_mem	访存读出的数据
wb_cp0_we	wire	input	MEM_WB	WB 段 cp0 旁路
wb_cp0_waddr	wire[4:0]	input	MEM_WB	WB 段 cp0 旁路
wb_cp0_wdata	wire[31:0]	input	MEM_WB	WB 段 cp0 旁路
is_in_delayslot_i	wire	input	EX_MEM	当前指令是否在延迟槽内

pc_i	wire[31:0]	input	EX_MEM	当前指令地址
excepttype_i	wire[31:0]	input	EX_MEM	异常发生情况
cp0_entrylo0_i	wire[31:0]	input	cp0_reg	目前 cp0 内 entrylo0 的值
cp0_entrylo1_i	wire[31:0]	input	cp0_reg	目前 cp0 内 entrylo1 的值
cp0_entryhi_i	wire[31:0]	input	cp0_reg	目前 cp0 内 entryhi 的值
cp0_index_i	wire[31:0]	input	cp0_reg	目前 cp0 内 index 的值
cp0_cause_i	wire[31:0]	input	cp0_reg	目前 cp0_内 cause 的值
cp0_status_i	wire[31:0]	input	cp0_reg	目前 cp0 内 status 的值
cp0_epc_i	wire[31:0]	input	cp0_reg	目前 cp0 内 epc 的值
cp0_ebase_i	wire[31:0]	input	cp0_reg	目前 cp0 内 ebase 的值
wreg_o	reg	output	MEM_WB	寄存器写使能
wd_o	reg[4:0]	output	MEM_WB	寄存器写地址
wdata_o	reg[31:0]	output	MEM_WB	寄存器写数据
whio_o	reg	output	MEM_WB	HI/LO 写使能
hi_o	reg[31:0]	output	MEM_WB	HI 写数据
lo_o	reg[31:0]	output	MEM_WB	LO 写数据
cp0_we_o	reg	output	MEM_WB	cp0 写使能
cp0_waddr_o	reg[4:0]	output	MEM_WB	cp0 写地址
cp0_wdata_o	reg[31:0]	output	MEM_WB	cp0 写数据
ram_ce_o	reg	output	mux_for_pc_mem	访存使能
MasterRd	reg	output	mux_for_pc_mem	访存读使能
MasterWr	reg	output	mux_for_pc_mem	访存写使能
MasterByte Enable	reg[3:0]	output	mux_for_pc_mem	访存读/写的字节使能
virtAddr	reg[31:0]	output	mux_for_pc_mem	访存地址
ram_data_o	reg[31:0]	output	mux_for_pc_mem	访存写数据
userMode	reg	output	mux_for_pc_mem	当前 cpu 运行模式
tlbwi	reg	output	mux_for_pc_mem	当前指令是否是 tlbwi 指令
refillMessage	reg[66:0]	output	mux_for_pc_mem	tlbrefillmessage
pc_o	reg[31:0]	output	cp0_reg	当前指令地址
is_in_delayslot_o	reg	output	cp0_reg	当前指令是否在延迟槽内
pcreset_o	reg	output	cp0_reg	当前指令是否是插入气泡
bad_mmu_addr	reg[31:0]	output	cp0_reg	访存失败的访存地址
int_o	reg[5:0]	output	cp0_reg	中断信息
cp0_epc_o	reg[31:0]	output	ctrl	最新 epc 值
cp0_ebase_o	reg[31:0]	output	ctrl	最新 ebase 值
excepttype_o	reg[4:0]	output	ctrl,cp0_reg	最终需要处理的异常类型

10.3 设计细节

MEM 段，五级流水线的访存阶段，在这个阶段进行访存操作（包含写 TLB）以及异常处理。

10.3.1 访存操作

访存操作由 MEM，mux_for_pc_mem 以及 MMU 相关模块协助完成，由于 IF 和 MEM 段均需要访存，所以将两者的信号汇总到 mux_for_pc_mem，再由 mux_for_pc_mem 模块根据控制信号将相关信号传到 MMU 模块访存，具体将在 mux_for_pc_mem 中介绍。

MEM 段仅负责给出访存信号，操作码以及相关，将由 MMU 模块在一个 CPU 时钟周期内完成访存操作并将结果传入。MEM 访存操作码如下：

类别	mem_op	功能
读操作	MEM_LB_OP	读取字节，符号扩展
	MEM_LBU_OP	读取字节，无符号扩展
	MEM_LW_OP	读取字
写操作	MEM_SB_OP	写字节
	MEM_SW_OP	写字
TLB 操作	MEM_TLBWI_OP	写 TLB

10.3.2 异常处理以及其与访存的交互

本 CPU 采取的异常处理方式为：统一将异常/中断传到 MEM 段进行处理。处理的异常类型如下：

异常发生阶段	异常码	异常类型
外界由 IF 段传入	00000	中断
MMU 传回	00010	TLB Load 异常
MMU 传回	00011	TLB Store 异常
访存前判断或 MMU 传回	00100	Load 异常
访存前判断或 MMU 传回	00101	Store 异常
ID 段判断	01000	syscall 指令
ID 段判断	01010	无效指令
ID 段判断	01001	break 指令
ID 段判断	01110	eret 指令
EX 段判断	01100	算术溢出

异常判断与访存的时序如下：

优先判断是否有中断和沿 ID 和 EX 段传来的异常，若有，则不进行访存操作；

再判断访存地址是否有对齐，若有，亦不进行访存操作；

若都无，进行访存，再判断是否有 MMU 部件返回的异常。

由此流程下来处理，若存在异常，则进入异常处理流程。

异常处理流程如下：

采用精确异常处理，所有异常（包括中断）都沿流水线传递至 MEM 处理（中断由 IF 段传入），优先处理最先达到的异常，若在同一个周期内有异常和中断，则优先处理中断。

在 MEM 段，若判断有异常，则：

向 Ctrl 发出清空流水线的 flush 信号，将在下一个信号清空流水线，并将 pc 跳至异常处理例程入口（此处为 cp0_ebase 寄存器），同时向 cp0 的 cause，status 和 Badvaddr 寄存器写入异常相关信息。

对于 eret 指令，流程大致相同，区别仅是将 pc 跳至 cp0_epc 寄存器。

11 mux_for_pc_mem

11.1 简介

用于处理 IF 段和 MEM 段访存冲突的模块，接受两者信息进行选择后传给 MMU 模块，并将 MMU 返回的信息进行选择传递给 IF 或是 MEM 段，属于组合逻辑电路。

11.2 接口定义

信号名称	信号规格	输入/输出	信号来源/去处	详细描述
rst	wire	input	电路板	复位信号
pc_e	wire	input	pc_reg	IF 段使能信号
mem_ce	wire	input	MEM	MEM 段使能信号
pc_addr	wire[31:0]	input	pc_reg	取指段指令地段
mem_addr	wire[31:0]	input	MEM	MEM 段访存地址
mem_MasterRd	wire	input	MEM	MEM 段读使能
mem_MasterWr	wire	input	MEM	MEM 段写使能
mem_Master ByteEnable	wire[3:0]	input	MEM	MEM 段字节使能
mem_data_i	wire[31:0]	input	MEM	MEM 段写数据
mem_userMode	wire	input	MEM	MEM 段用户模式
mem_tlbwi	wire	input	MEM	MEM 段是否是 tlbwi 指令
mem_refillMessage	wire[66:0]	input	MEM	MEM 段 TLB Refill 信息
mmu_data_i	wire	input	MMU 模块	MMU 段传回访存结果
mmu_miss	wire	input	MMU 模块	访存是否 miss
mmu_invalid	wire	input	MMU 模块	访存是否 invalid
mmu_permission Denied	wire	input	MMU 模块	访存是否 permissionDenied
mmu_addr	reg[31:0]	output	MMU 模块	传给 MMU 的访存地址
mmu_MasterRd	reg	output	MMU 模块	传给 MMU 的读使能
mmu_MasterWr	reg	output	MMU 模块	传给 MMU 的写使能
mmu_Master ByteEnable	reg[3:0]	output	MMU 模块	传给 MMU 的字节使能
mmu_data_o	reg[31:0]	output	MMU 模块	传给 MMU 的写数据

userMode	reg	output	MMU 模块	传给 MMU 的用户模式
tlbwi	reg	output	MMU 模块	传给 MMU 的 tlbwi 使能
refillMessage	reg[66:0]	output	MMU 模式	传给 MMU 的 TLB Refill 信息
pc_inst	reg[31:0]	output	IF_ID	IF 段取到的指令
pc_miss	reg	output	IF_ID	IF 段访存是否出现 miss
pc_invalid	reg	output	IF_ID	IF 段访存是否出现 invalid
pc_permission Denied	reg	output	IF_ID	IF 段访存是否 permissionDenied
mem_data_o	reg[31:0]	output	MEM	MEM 段访存结果
mem_miss	reg	output	MEM	MEM 段访存是否出现 miss
mem_invalid	reg	output	MEM	MEM 段访存是否出现 invalid
mem_permission Denied	reg	output	MEM	MEM 段访存是否 permissionDenied

11.3 设计细节

先介绍 CPU 与 MMU 的交互，CPU 访问 MMU 模块时，需要提供读写使能和字节使能，访存地址，可能需要写入的数据，运行模式，tlbwi 以及 TLB Refill 信息，而 MMU 返回访存结果和是否在访存时出现 miss, invalid, permissionDenied 三种错误。而 CPU 的 IF 段和 MEM 段都需要访存，所以设此模块在 IF 和 MEM 段信息中作选择传给 MMU 模块，在一个时钟周期内 MMU 返回结果，此模块同样选择把 MMU 的访存结果传到 IF 或 MEM 模块。

IF 和 MEM 段均有使能信号，而由于 MEM 段访存指令在 EX 段时会暂停流水线，可以保证在 MEM 段需要访存时 IF 段不需要，所以此处判断以 MEM 段优先，若 MEM 段有使能，则向 MMU 传递 MEM 段的信号，否则传递 IF 段的信号。

MMU 段返回结果，此处的选择同是 MEM 段优先，若有 MEM 使能，同样优先传信号给 MEM 段，同时保持传给 IF 段的信号不变，反之同样。

若 MEM 段访存出现错误，直接回传到 MEM 段处理；若 IF 段访存出错，则传错误信息到 IF 段，并传空指令，沿流水线将错误信息传至 MEM 段再处理。

12 MEM_WB

12.1 简介

五级流水线中 MEM 段和 WB 段间的触发器，在时钟上升沿跟据控制信息将 MEM 段的信息传至 WB 段，属于时序逻辑电路。

12.2 接口定义

信号名称	信号规格	输入/输出	信号来源/去处	详细描述
clk	wire	input	电路板	时钟信号
rst	wire	input	电路板	复位信号
stall	wire[5:0]	input	Ctrl	流水线暂停的信号
flush	wire	input	Ctrl	由异常引起的流水线清洗的信号
mem_wreg	wire	input	MEM	寄存器写使能
mem_wd	wire[4:0]	input	MEM	寄存器写地址
mem_wdata	wire[31:0]	input	MEM	寄存器写数据
mem_whio	wire	input	MEM	HI/LO 写使能
mem_hi	wire[31:0]	input	MEM	HI 写数据
mem_lo	wire[31:0]	input	MEM	LO 写数据
mem_cp0_we	wire	input	MEM	cp0 写使能
mem_cp0_waddr	wire[4:0]	input	MEM	cp0 写地址
mem_cp0_wdata	wire[31:0]	input	MEM	cp0 写数据
wb_wreg	reg	output	regfile	寄存器写使能
wb_wd	reg[4:0]	output	regfile	寄存器写地址
wb_wdata	reg[31:0]	output	regfile	寄存器写数据
wb_whio	reg	output	hilo_reg	HI/LO 写使能
wb_hi	reg[31:0]	output	hilo_reg	HI 写数据
wb_lo	reg[31:0]	output	hilo_reg	LO 写数据
wb_cp0_we	reg	output	cp0_reg	cp0 写使能
wb_cp0_waddr	reg[4:0]	output	cp0_reg	cp0 写地址
wb_cp0_wdata	reg[31:0]	output	cp0_reg	cp0 写数据

12.3 设计细节

MEM_WB，位于五级流水线 MEM 与 WB 段间的触发器，根据控制信息(flash 和 stall)来决定向 WB 段传入什么信息，有四种情况：

1. 流水线正常运转
2. 流水线译码段暂停(stall[3]和 stall[4]均为 1)
3. 流水线暂停，向译码段插入气泡(stall[3]为 1，stall[4]为 0)
4. 异常处理，流水线清空(flash 信号为 1)

优先级为：4 > 3 > 2 > 1

按照优先级判断向 WB 段传递相关信息，分别将信息传至寄存器堆，HI/LO 寄存器和 CP0 寄存器堆写回。

13 regfile

13.1 简介

regfile 模块实现 32 个通用寄存器，提供两个读端口和一个写端口，随时可读，在每个时钟上升沿写数据。

13.2 接口简介

信号名称	信号规格	输入/输出	信号来源/去处	详细描述
clk	wire	input	电路板	时钟信号
rst	wire	input	电路板	复位信号
re1	wire	input	ID	读使能 1
raddr1	wire[4:0]	input	ID	读地址 1
re2	wire	input	ID	读使能 2
raddr2	wire[4:0]	input	ID	读地址 2
we	wire	input	ID	写使能
waddr	wire[4:0]	input	ID	写地址
wdata	wire[31:0]	input	ID	写数据
rdata1	reg[31:0]	output	ID	读数据 1
rdata2	reg[31:0]	output	ID	读数据 2

13.3 设计细节

设置两个读端口和一个写端口。

读端口为组合逻辑，任何时刻可读，WB 段的数据旁路在此实现，即若要读的地址是即将写入的地址，直接返回要写入的数据，否则从寄存器读数据。

写端口为时序逻辑，在每个时钟上升沿写入数据。

对 0 号寄存器的读写进行特判，永远读出 0，无法写入数据。

14 hilo_reg

14.1 简介

实现存储 HILO 寄存器的模块，提供写端口，每个周期都将当前数据读出到 EX 段。

14.2 接口定义

信号名称	信号规格	输出/输出	信号来源/去处	详细描述
clk	wire	input	电路板	时钟信号
rst	wire	input	电路板	复位信号
we	wire	input	MEM_WB	写使能
hi_i	wire[31:0]	input	MEM_WB	HI 写数据
lo_i	wire[31:0]	input	MEM_WB	LO 写数据
hi_o	reg[31:0]	output	EX	输出数据
lo_o	reg[31:0]	output	EX	输出数据

14.3 设计细节

HILO 寄存器与寄存器堆类似，在每个时钟上升沿写入，每个周期都读出数据，不再另设读端口。

15 cp0_reg

15.1 简介

实现 cp0 里各寄存器的读写。

15.2 接口定义

信号名称	信号规格	输入/输出	信号来源/去处	详细描述
clk	wire	input	电路板	时钟信号
rst	wire	input	电路板	复位信号
we_i	wire	input	MEM_WB	写使能
waddr_i	wire[4:0]	input	MEM_WB	写地址
wdata_i	wire[31:0]	input	MEM_WB	写数据
rsel_i	wire	input	EX	读使能
raddr_i	wire[4:0]	input	EX	读地址
pc_i	wire[31:0]	input	MEM	异常指令地址
pcreset_i	wire	input	MEM	发生异常的指令是否是气泡
is_in_delayslot_i	wire	input	MEM	异常指令是否在延迟槽内
excepttype_i	wire[4:0]	input	MEM	异常指令类型
badaddr_i	wire[31:0]	input	MEM	访存错误的地址
int_i	wire[5:0]	input	MEM	中断信息
data_o	reg[31:0]	output	EX	读出的数据
timer_int_o	reg	output	外部模块	是否发生时钟中断
status_o	reg[31:0]	output	MEM	status 寄存器值
cause_o	reg[31:0]	output	MEM	cause 寄存器值
epc_o	reg[31:0]	output	MEM	epc 寄存器值
ebase_o	reg[31:0]	output	MEM	ebase 寄存器值
index_o	reg[31:0]	output	MEM	index 寄存器值
entrylo0_o	reg[31:0]	output	MEM	entrylo0 寄存器值
entrylo1_o	reg[31:0]	output	MEM	entrylo1 寄存器值
entryhi_o	reg[31:0]	output	MEM	entryhi 寄存器值

15.3 设计细节

实现正常的 CP0 寄存器读写，在每个时钟周期的上升沿写入，任意时刻均可读。

内部实现 compare 和 count 寄存器，当它们值相等时产生时钟中断，传至外部模块和串口中断一同从 IF 段传入流水线，传至 MEM 段处理，在 compare 被写入新值前时钟中断信号不会消失。

异常处理流程在 MEM 段已有介绍，关于 cp0 的部分即，向 cause 和 status 寄存器中写入相关值，根据异常类型和异常指令是否在延迟槽内更新 badvaddr 寄存器和 epc 寄存器。

16 ctrl

16.1 简介

Ctrl 模块负责接收流水线各个阶段的暂停和异常信号，然后根据优先级向各触发器发出清洗，暂停或插入气泡指令。

16.2 接口定义

信号名称	信号规格	输入/输出	信号来源/去处	详细描述
rst	wire	input	电路板	复位信号
ex_stall	wire	inpue	EX	除法指令暂停信号
ex_idstall	wire	input	EX	访存指令暂停信号
flash_stall	wire	input	MMU 模块	读 flash 暂停信号
excepttype_i	wire[4:0]	input	MEM	异常类型
cp0_epc_i	wire[31:0]	input	MEM	异常发生时 epc 值
cp0_ebase_i	wire[31:0]	input	MEM	异常发生时 ebase 值
flush	reg	output	各触发器	异常处理流水线清空信号
stall	reg[5:0]	output	各触发器	流水线暂停信号
new_pc	reg[31:0]	output	pc_reg	异常处理跳转地址

16.3 设计细节

ctrl 模块主要接收各个模块发出的流水线暂停清求以及异常处理时的流水线清洗请求。

可能收到的请求如下：

异常处理的流水线清洗信号

读 flash 时的流水线暂停信号

除法指令的流水线暂停信号

访存指令的流水线暂停信号

其中 flash_stall 由外部 flash 模块传入，异常信号由 MEM 段传入，除法和访存暂停均由 EX 段传入。优先级顺序为：异常信号 > flash_stall > 除法/访存指令的暂停

异常发生时，将清空流水线，跳至异常处理例程，处理完毕后再重新回到异常发生处执行，所以在异常发生时的其他暂停请求在异常处理完毕之后还会照常出现，故异常处理的优先级最高。

flash_stall 为访问 flash 读时发生，flash 访问为访存指令，所以当该指令进入 MEM 段时 EX 段应为插入的气泡，不会有其他 stall 请求，所以次之。

除法和访存的 stall 均由 EX 段发出，故只会发出一个，不必考虑优先级问题。

各信号的处理方法如下：

对于异常信号，向各触发器发出清空流水线的信息，同时向 pc_reg 段发出异常处理将要跳转的地址

对于 flash_stall，向 WB 段插入气泡，暂停之前的四个阶段。

对于除法，向 MEM 段插入气泡，暂停之前的三个阶段

对于访存，向 EX 段插入气泡，暂停之前的两个阶段。

17 MMU

17.1 模块介绍

MMU 模块负责将虚拟地址转为物理地址，以及 tlb 的查询，重填等操作。

17.2 接口定义

信号名称	规格	来源	功能描述
clk	wire	pll 时钟发生器	cpu 时钟
rst	wire	pll 时钟发生器	复位信号
en	wire	CPU	访存使能信号
userMode	wire	CPU	当前运行级别：0 内核态，1 用户态
virtAddr	wire[31:0]	CPU	访存的虚拟地址
refillMessage	wire[66:0]	CPU	重填 TLB 表项
tlbwi	wire	CPU	TLB 重填特权指令
MasterRd	wire	CPU	访存读使能信号
MasterWr	wire	CPU	访存写使能信号
MasterByteEnable	wire	CPU	访存字节使能信号

（输入信号）

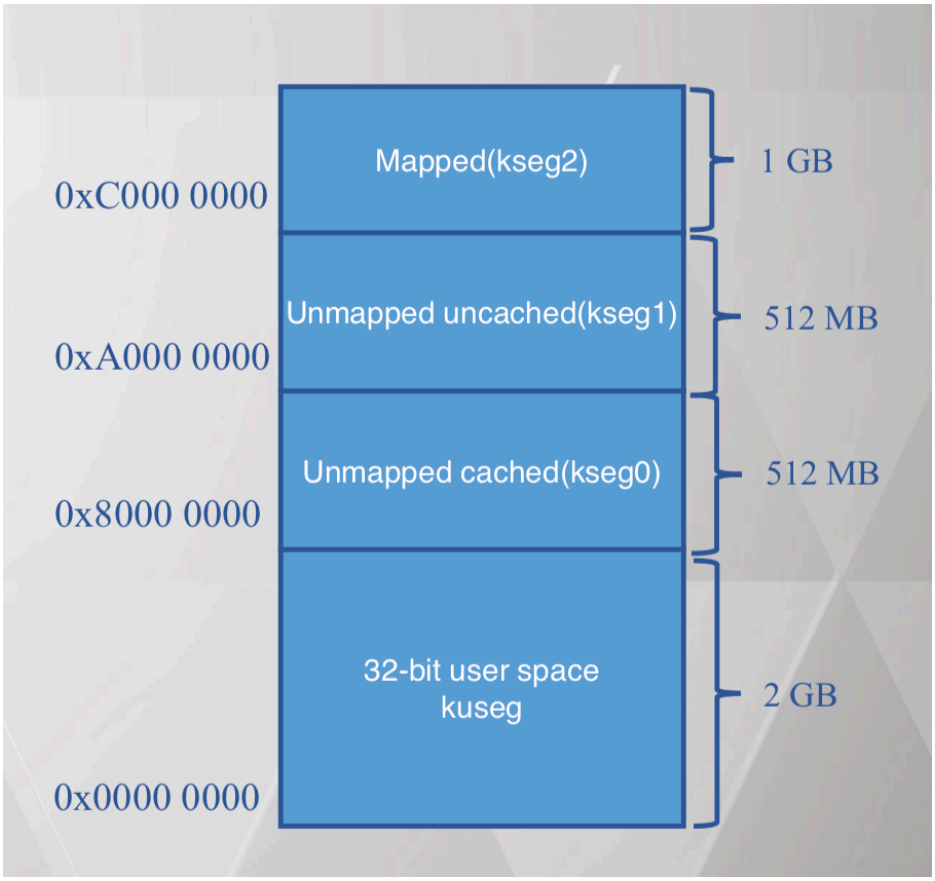
信号名称	规格	去向	功能描述
physAddr	wire[31:0]	CPU	访存的物理地址
permissionDenied	wire	CPU	权限不足异常
miss	wire	CPU	tlb 缺失异常
dirty	wire	CPU	tlb 条目为脏（未实现）
invalid	wire	CPU	tlb 条目不合法
BusRd	wire	Bus	总线读使能信号
BusWr	wire	Bus	总线写使能信号
BusByteEnable	wire[3:0]	Bus	总线字节使能信号

（输出信号）

17.3 设计思路

17.3.1 虚拟地址管理

ucore 对于虚拟地址空间的划分见下图，其中 kuseg 和 kseg2 需要使用页表查找物理地址，而 kseg0 和 kseg1 区域直接将虚拟地址的最高三位清零就是对应的物理地址。当运行状态为用户态时，只能访问 kuseg 内部的地址，否则会触发权限不足异常（按照地址未对齐异常处理）：

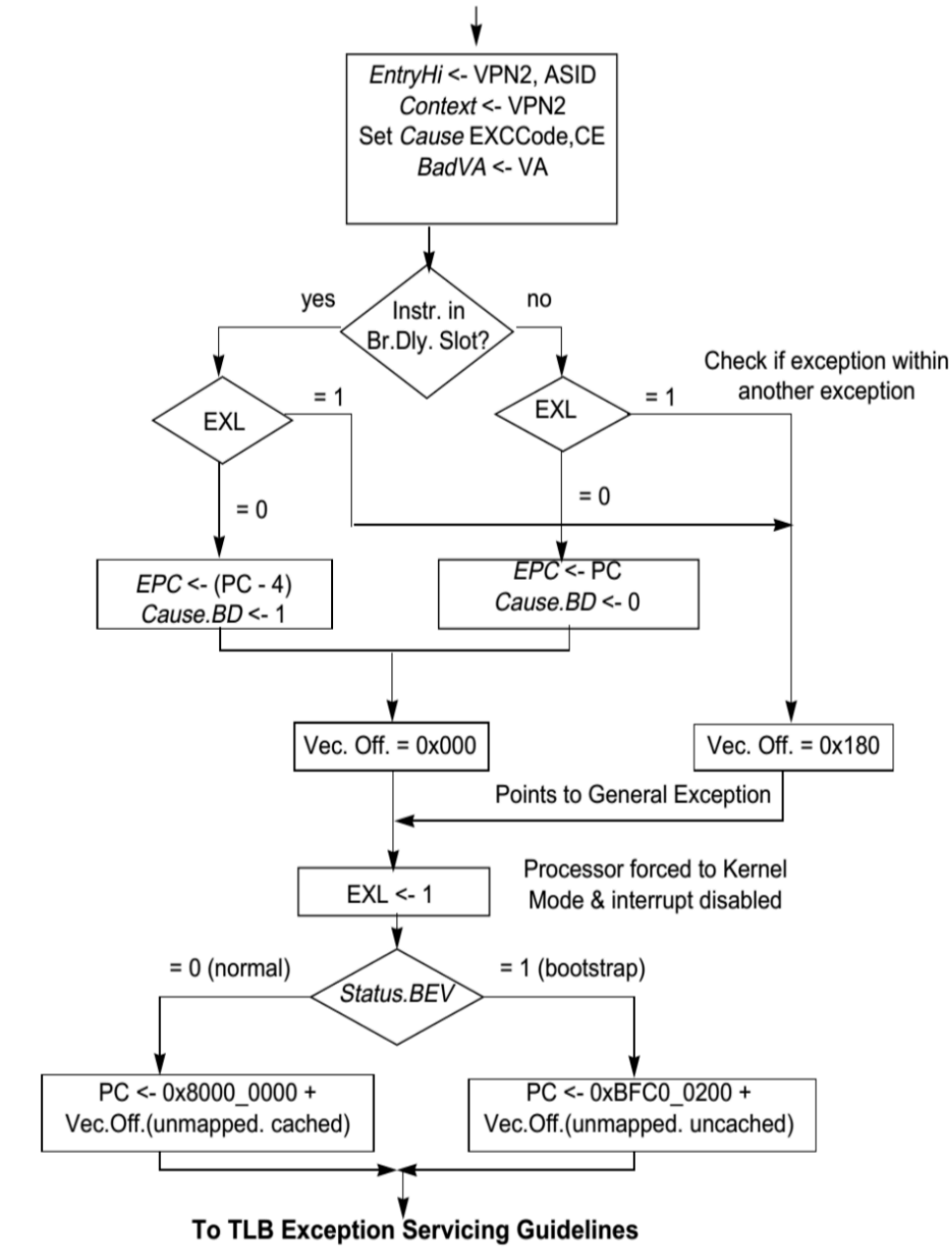


17.3.2 TLB

TLB 是页表的缓存，承担着一部分的将虚拟地址映射为物理地址的功能。在 Totoro CPU 中，页面大小为 4KB，TLB 共有 16 项，采用全相连结构，TLB 的装填由特权指令 tlbwi 完成，由软件负责实现 TLB 的装填策略。

62~44	43~24	23	22	21~2	1	0
Virtual Page Number	Page Frame Number0	D0	V0	Page Frame Number1	D1	V1

TLB 模块在查询的过程中可能产生的异常有：TLB Miss, TLB Invalid, TLB Modified, TLB Refill 等（对于本项目实际上只会出现前两种）异常处理时，首先要关闭总线使能，然后按照以下流程处理：



18 总线

18.1 模块介绍

Totoro 的总线模块是一个纯组合逻辑模块，负责解析物理地址并映射到对应的外设的访问地址。

18.2 接口定义

信号名称	规格	来源	功能描述
masterAddr_i	wire[31:0]	MMU	准备访存的地址
masterRd_i	wire	MMU	总线读使能信号
masterWr_i	wire	MMU	总线写使能信号
masterByteEnable_i	wire[31:0]	MMU	总线字节使能信号
masterData_i	wire[31:0]	CPU	准备写的数据
sramData_i	wire[31:0]	SramController	sram 读出的数据
romData_i	wire[31:0]	BootRom	rom 中读出的指令
flashData_i	wire[31:0]	FlashController	flash 读出的数据
uartData_i	wire[31:0]	UartController	串口收到的数据/串口读写状态

（输入信号）

信号名称	规格	去向	功能描述
masterData_o	reg[31:0]	CPU	读外设得到的数据
sramRd_o	reg	SramController	sram 读使能信号
sramWr_o	reg	SramController	sram 写使能信号
sramByteEnable_o	reg[3:0]	SramController	sram 字节使能信号
sramData_o	reg[31:0]	SramController	准备写入 sram 的数据
sramAddr_o	reg[21:0]	SramController	读写 sram 地址
romAddr_o	reg[11:0]	BootRom	读 rom 地址
flashRd_o	reg	FlashController	flash 读使能信号
flashWr_o	reg	FlashController	flash 写使能信号
flashByteEnable_o	reg[3:0]	FlashController	flash 字节使能信号
flashData_o	reg[31:0]	FlashController	准备写入 flash 的数据
flashAddr_o	reg[23:0]	FlashController	读 flash 地址
uartRd_o	reg	UartController	串口读使能信号
uartWr_o	reg	UartController	串口写使能信号
uartData_o	reg[31:0]	UartController	准备发送串口的数据
uartAddr_o	reg[3:0]	UartController	读写串口地址
vgaRd_o	reg	VGAController	显存读使能信号
vgaWr_o	reg	VGAController	显存写使能信号
vgaData_o	reg[31:0]	VGAController	准备写入显存的数据
vgaAddr_o	reg[18:0]	VGAController	写显存地址

（输出信号）

18.3 映射地址

外设	起始物理地址	终止物理地址
主存	0x00000000	0x003fffff
bootRom	0x1fc00000	0x1fcfffff
Flash	0x1e000000	0x1effffff
串口	0x1fd003f0	0x1fd003ff
显存	0x1b000000	0x1b07ffff

19 SramController

19.1 模块介绍

在 thinpad 上有两块 sram 可以作为 Totoro 的内存，SramController 位于总线和内存之间，负责解析总线传递过来的访存信号，处理 cpu 和内存的时序关系（在一个 cpu 周期内完成 sram 的访存操作）

19.2 接口定义

信号名称	规格	来源	功能描述
clk2x	wire	pll 时钟发生器	cpu 时钟倍频+适当延时
rst	wire	bus	复位信号
masterRd_i	wire	bus	读使能信号
masterWr_i	wire	bus	写使能信号
byteEnable_i	wire[3:0]	bus	字节使能信号（低有效）
writeData_i	wire[31:0]	bus	要写到 sram 里的数据
ramAddr_i	wire[21:0]	bus	读写 sram 的地址（字节编址）

（输入信号）

信号名称	规格	来源/去向	功能描述
ramData_bus	wire[31:0]	板上 sram 芯片	板上 sram 的数据总线

（输入/输出信号）

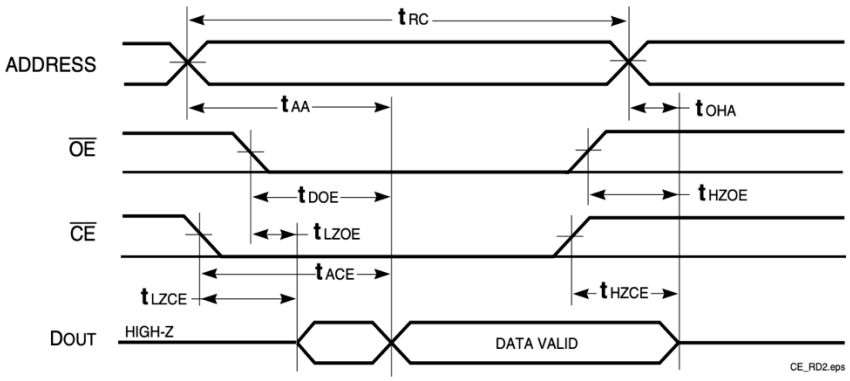
信号名称	规格	去向	功能描述
ramAddr_o	reg[19:0]	板上 sram 芯片	读写 sram 的地址（字编址）
sramEnable_n_o	reg	板上 sram 芯片	sram 使能信号（低有效）
writeEnable_n_o	reg	板上 sram 芯片	sram 写使能信号（低有效）
readEnable_n_o	reg	板上 sram 芯片	sram 读使能信号（低有效）
byteEnable_o	reg[3:0]	板上 sram 芯片	sram 字节使能信号（低有效）
ramData_o	reg[31:0]	bus	从 sram 中读出的数据

（输出信号）

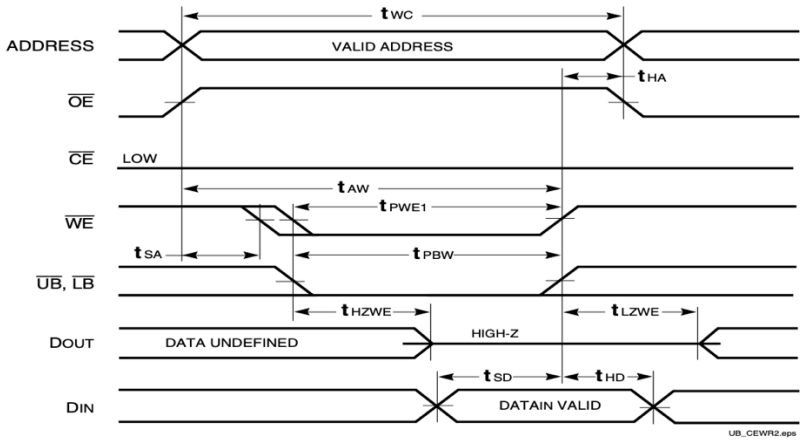
19.3 设计思路

19.3.1 芯片规格与时序

* 型号 IS61WV102416ALL
* 大小 2^{20} * 16bit * 2，共 4MB（选取 extRam 作为主存），地址空间 20 位（字编址）
* 读周期：8ns-10ns
* 写周期：8ns-10ns



（读时序）



（写时序）

由时序关系图可看出，读操作一个周期即可完成，但写操作需要考虑写保持时间，即在 WE 信号拉高后，写数据仍需保持一段时间。基于以上特性，我们将 SramController 的周期设置为 cpu 周期的两倍，在处理写操作时，第一个周期将 WE 信号拉低，第二个周期将 WE 信号拉高，并且缓存要写到 sram 中的地址与数据。

19.3.2 数据通路延迟问题

由于 MEM 段得到的访存操作使能信号需要经过 MMU，TLB，bus 才能最终到达 SramController，实际延迟大概在 18ns 左右，若采用 25MHz 的 cpu 时钟，则需要将 clk2x 的相位设置为 330°。

20 BootRom

20.1 模块介绍

ucore 在启动之前，需要先将存储在 flash 上的代码转移到主存当中，这部分代码是只读的，而且断电后不能消失，所以需要存储在 rom 中，但 thinpad 实验板并未提供 rom 芯片，所以需要在 FPGA 当中模拟一个 rom.

vivado 提供了 Distributed Memory Generator IP 核，用来在 FPGA 中模拟比较小的存储空间，只需要将 ucore 的 bootloader 汇编为机器码，再转化为 IP 核需要的 coe 文件导入进去即可。

由于 rom 功能比较简单，不需要时序控制和使能信号，实际上相当于组合逻辑电路。

大小：512B，地址空间 7 位（字编址）

20.2 接口定义

信号名称	规格	来源	功能描述
a	wire[6:0]	bus	读 rom 地址（字编址）

（输入信号）

信号名称	规格	去向	功能描述
spo	wire[31:0]	bus	从 rom 中读出的指令

（输出信号）

21 FlashController

21.1 模块介绍

FlashController 模块位于总线和板上 flash 芯片之间，负责解析总线传递过来的读 flash 的信号，完成 cpu 编址方式到 flash 编址方式的转化（转化过程对总线透明）以及处理访问 flash 的时序关系，控制流水线的暂停与继续。

21.2 接口定义

信号名称	规格	来源	功能描述
clk	wire	pll 时钟发生器	cpu 时钟反相
rst	wire	pll 时钟发生器	复位信号
masterRd_i	wire	bus	读使能信号
masterWr_i	wire	bus	写使能信号
byteEnable_i	wire[3:0]	bus	字节使能信号（低有效）（仅低两位有效）
writeData_i	wire[31:0]	bus	要写到 flash 里的数据（未处理）
flashAddr_i	wire[23:0]	bus	读 flash 的地址（字节编址）

（输入信号）

信号名称	规格	来源/去向	功能描述
flashData_bus	wire[15:0]	板上 flash 芯片	板上 flash 的数据总线

（输入/输出信号）

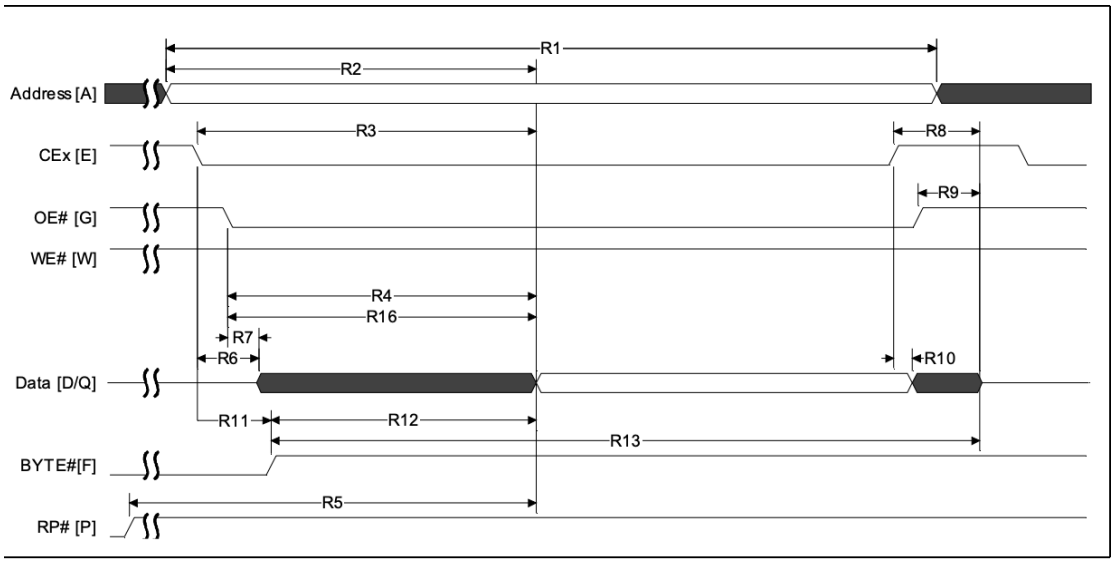
信号名称	规格	去向	功能描述
flashAddr_o	reg[22:0]	板上 flash 芯片	读写 flash 的地址（字节编址）
flashEnable_n_o	reg	板上 flash 芯片	flash 使能信号（低有效）
writeEnable_n_o	reg	板上 flash 芯片	flash 写使能信号（低有效）
readEnable_n_o	reg	板上 flash 芯片	flash 读使能信号（低有效）
lowBitMode_n_o	reg	板上 flash 芯片	flash 8bit 模式选择信号（低有效）
flashData_o	reg[31:0]	Bus	从 flash 中读出的数据（仅低 16 位有效，高 16 位为 0）
flashStall_o	reg	cpu	流水线暂停信号

（输出信号）

21.3 设计思路

21.3.1 芯片规格与时序

* 型号 JS28F640J3D75
* 大小 2^{22} * 16bit，共 8MB，地址空间 23 位（字节编址，但最低位只有开启 8bit 模式才有效）
* 读周期：75ns-95ns
* 写周期：较长，故不实现写操作



（读时序）

由时序图可知，flash 读一次的时间大约需要三个 cpu 周期，于是在读 flash 的时候流水线必须暂停。并且，在读 flash 之前，需要先用两个 cpu 周期向 flash 中写入 0xFF 的数据以切换内部状态机的状态。由于数据通路存在延迟，还需要再加一个时钟周期来等待流水线恢复工作，这期间需要缓存读到的数据。所以，读一次 flash 的时间为 6 个 cpu 时钟周期。为了方便控制，flash 的时钟频率和 cpu 时钟频率相同。

21.3.2 编址方式转换

flash 内部为字节编址，每次最多读出 16bit 数据。在软件层面上看来，flash 的空间有 16MB，但每个字的高 16bit 都是 0。所以需要将总线传过来的物理地址除以 4 以后作为传给 flash 芯片的地址的高 22 位。传给 flash 芯片的地址的最低位以及是否开启 8bit 模式由总线传过来的字节使能的低两位确定。

21.3.3 数据通路延迟问题

同样地，cpu 对 flash 的控制信号到达 FlashController 时也存在一定延迟，时长也在 18ns 左右，对此，在 cpu 时钟频率为 25MHz 时，我们需要将 flash 时钟反相，提供充足的数据准备时间。

22 UartController

22.1 模块介绍

Totoro CPU 和外界的通信通过串口进行，而 UartController 位于总线和串口之间，负责解析 cpu 和串口相互传递的信号，控制串口的收发时序。

22.2 接口定义

信号名称	规格	来源	功能描述
clk_cpu	wire	pll 时钟发生器	cpu 时钟
masterRd_i	wire	bus	读使能信号
masterWr_i	wire	bus	写使能信号
writeData_i	wire[31:0]	bus	准备发送的数据（仅低 8 位有效）
uartAddr_i	wire[3:0]	bus	读写串口的地址
rxd	wire	电路板	串口数据接收线

（输入信号）

信号名称	规格	去向	功能描述
uartData_o	reg[31:0]	bus	接收到的串口的数据（仅低 8 位有效）/ 串口读写状态(仅低 2 位有效)
interrupt	wire	CPU	串口中断信号
txd	wire	电路板	串口数据发送线

（输出信号）

22.3 设计思路

mips 串口遵循的标准如下：

* 起始位 1bit
* 数据位 8bit
* 终止位 1bit
* 校验位 无

模板工程中提供了控制串口收发的模块 async_receiver 和 async_transmitter, 可以通过对 cpu 时钟的分频实现可变波特率的异步收发，例化时将 cpu 时钟频率和波特率作为参数传递进去即可。UartController 模块只需实现信号的传递与解析功能即可，是组合逻辑电路。

ucore 读写串口的地址有两个：0x8 和 0xC，前者为数据地址，存储发送或者接收的数据；后者为只读的状态地址，存储当前串口的读写状态，最低位表示当前是否可以发送数据，次低位表示当前是否收到了来自外界的数据。

23 VGAController

23.1 模块介绍

VGAController 模块一方面接收来自总线的写显存的指令，另一方面每个周期循环地将显存中的颜色输出到 vga 接口上，最终在屏幕上显示。

23.2 接口定义

信号名称	规格	来源	功能描述
clk_cpu	wire	pll 时钟发生器	cpu 时钟反相
clk_vga	wire	pll 时钟发生器	50MHz
rst	wire	pll 时钟发生器	复位信号
vgaAddr_i	wire[18:0]	bus	写显存地址
writeData_i	wire[31:0]	bus	写显存数据
masterWr_i	wire	bus	写使能信号

（输入信号）

信号名称	规格	去向	功能描述
hsync	wire	vga	行同步信号
vsync	wire	vga	列同步信号
data_enable	wire	vga	vga 数据使能信号
vgaData_o	reg[7:0]	vga	写入 vga 的颜色值

（输出信号）

23.3 设计思路

实验中使用的 VGA 参数如下：

* 分辨率 800 * 600
* 刷新率 75Hz
* 像素时钟 50MHz
* 单像素大小 8bit (R: 3bit, G: 3bit, B: 2bit)

VGA 使用的显存需要支持同时读写操作，这里我们使用了 vivado 的 Block Memory Generator，在 FPGA 中开辟了一块 480000 字节的存储空间，类型选择 Simple Dual Port RAM，若同时读写，则写优先级更高，读取结果为正在写入的数据。读时钟选择 vga 的 50MHz 时钟，写时钟选择 cpu 的时钟（同样考虑到数据通路延迟，这里实际上选择的是 flash 的时钟）

由于 Block Memory Generator 在读取时存在一个周期的延迟，所以每次刷新的时候要读取下一个地址，这样在屏幕上显示的才是正确的地址的像素值。