“翱翔之光”CPU 设计报告

西北工业大学 1 队
蔡金洋、杨赟、董奕兰、王杰

一、设计简介

本设计实现了一个基于 MIPS 指令集系统的五级流水线 CPU. 本 CPU 支持“龙芯杯”第二届全国大学生计算机系统能力培养大赛所要求的全部 57 条指令, 支持中断与例外处理, 支持系统控制寄存器, 支持 AXI 协议的基本功能, 实现了一级指令 cache 和一级数据 cache, 并能成功通过大赛所规定的 soc_axi_func、memory_game 和 soc_axi_perf。 我们采用 verilog 语言来描述这一寄存器传输级(RTL)电路设计. 最高频率 125MHZ, 性能测试得分 70.773 分.

二、设计方案

（一）总体设计思路

我们的 CPU 分为 5 级, 分别是取指阶段 IF、译码阶段 DEC、执行阶段 EXE、访存阶段 MEM、寄存器写回阶段 WB．各个阶段执行完成后产生的结果将锁存在相应的流水线寄存器中以供下一阶段使用．CPU 控制信号在译码阶段产生.

工作机制：整个 CPU 只受同一个时钟驱动，并且时序逻辑只在时钟上升沿进行触发，通过流水线寄存器将每条指令的执行过程分为五步来进行，从而实现五级流水。

指令执行过程简要说明如下：

IF：程序计数器（PC）通过访问 Icache 或者通过 AXI 接口直接访问存储器来获取所需执行的指令，在时钟上升沿到来后将指令存入 IF/DEC 流水线寄存器中，从而进入 DEC 级。

DEC：从 IF/DEC 流水线寄存器中读取指令进行译码， 以获得控制信号，并从寄存器堆读出所需数据，在时钟上升沿到来后将信息存入 DEC/EXE 流水线寄存器中，从而进入 EXE 级。

EXE：从 DEC/EXE 流水线寄存器中获取所需数据和控制信号，在 EXE 级进行相应的算术逻辑运算，根据控制信号获得运算结果并在时钟上升沿到来时将结果存入 EXE/MEM 流水线寄存器中。

MEM：从 EXE/MEM 流水线寄存器获取相应的访存地址，通过访问 Dcache 或直接访问存储器实现数据传送指令。

WB：将所获得的数据写回相应的寄存器。

关于异常处理：对于一条指令来说，在 IF、DEC、EXE、MEM 级进行异常信息的收集，并在 MEM 级进行简单的异常处理，异常信息写入 CP0 寄存器的过程放在 WB 级进行。

为了使流水线更加规整，根据所要写回的寄存器，将数据通路实际分为 3 个通道，分别为 RegFile 通道，HiLo 通道和 CP0 通道。

(二) if_stage 模块设计

1、主要功能：

（1）更新程序计数器（PC），根据 PC 值从 ICache 或存储器中读取指令，并保存在流水线寄存器中。

（2）检测当前指令地址错例外（取指 PC 不对齐于字边界），并将异常信息保存在流水线寄存器中。

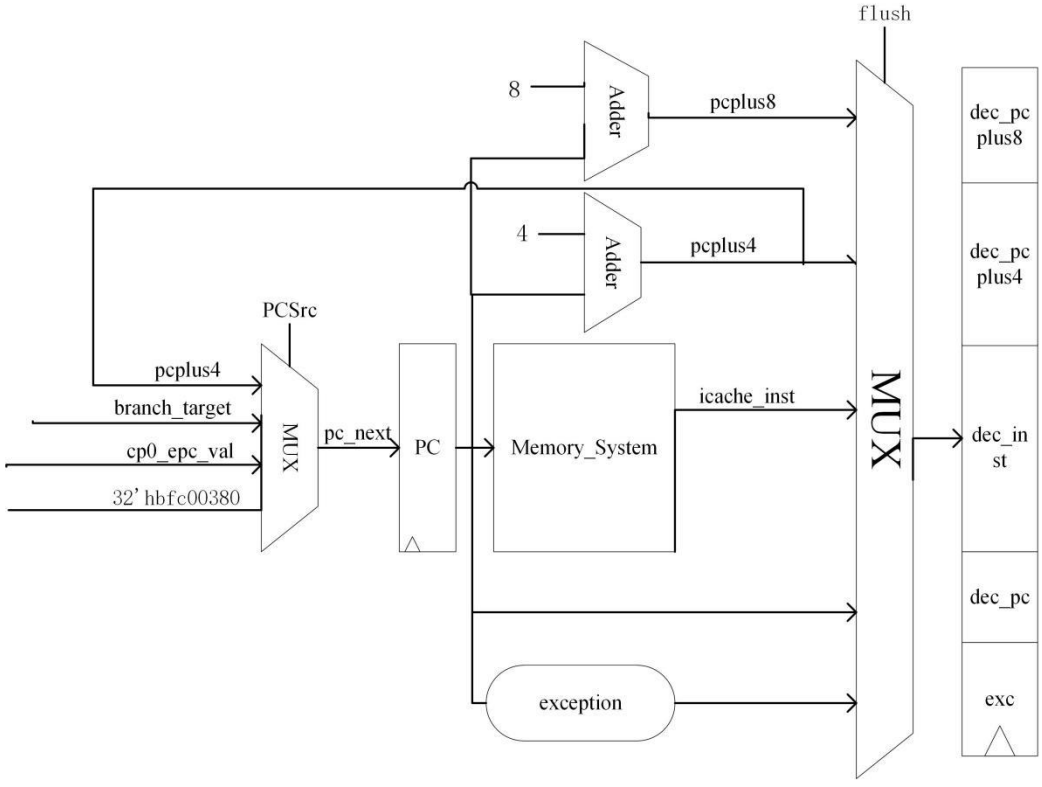


图 1　if_stage 结构简图

2、输入输出：

表 1　if_stage 输入

名称	类型	产生模块	说明
clk	wire		时钟信号，驱动时序逻辑
reset	wire		复位信号，进行初始化
PCSrc	wire[1:0]	branch	更新 pc 的选择信号
branch_target	wire[31:0]	branch	更新 pc 对应的分支跳转地址
cp0_epc_val	wire[31:0]	CP0	更新 pc 对应的中断、例外返回地址
pc_wren	wire	dec_stage	pc 写使能信号，当发生 lw-use 型冒险时为 0
dec_wren	wire	dec_stage	IF/DEC 流水线寄存器写使能信号，当发生 lw-use 型冒险时为 0
ready	wire	Memory_System	各流水线寄存器及 pc 的写使能信号，当访问存储器（取指/取数）时为 0
complete	wire	exe_stage	各流水线寄存器及 pc 的写使能信号，当进行乘除法时为 0
exception_flush	wire	mem_stage	流水线寄存器输入端输入数据的选择信号，当发生异常时为 1，将流水线寄存器中的数据清 0
eret_flush	wire	dec_stage	IF/DEC 流水线寄存器输入端输入数据的选择信号，当 dec 级为 eret 指令时为 1，将 IF/DEC 流水线寄存器数据清 0
bd	wire	dec_stage	表示当前指令是否为延迟槽指令，若是，则为 1
icache_inst	wire[31:0]	Memory_System	表示从 icache 或存储器中取回的指令

表 2　if_stage 输出

名称	类型	目的模块	说明
pc	reg[31:0]	Memory_System	pc 寄存器，用于取指
dec_inst	reg[31:0]	dec_stage	IF/DEC 流水线寄存器，传递信息
dec_pcplus4	reg[31:0]	dec_stage	IF/DEC 流水线寄存器，传递信息
dec_pcplus8	reg[31:0]	dec_stage	IF/DEC 流水线寄存器，传递信息
dec_pc	reg[31:0]	dec_stage	IF/DEC 流水线寄存器，传递信息
dec_exception_if_exchappen	reg	dec_stage	IF/DEC 流水线寄存器，传递信息
dec_exception_if_epc	reg[31:0]	dec_stage	IF/DEC 流水线寄存器，传递信息
dec_exception_if_bd	reg	dec_stage	IF/DEC 流水线寄存器，传递信息
dec_exception_if_badvaddr	reg[31:0]	dec_stage	IF/DEC 流水线寄存器，传递信息

dec_exception_if_excode	reg[4:0]	dec_stage	IF/DEC 流水线寄存器，传递信息

(三) dec_stage 模块设计

1、主要功能：

（1）根据从 IF/DEC 流水线寄存器中读取的指令进行译码，产生所需要的控制信号。

（2）分别从 CP0 寄存器、32 个通用寄存器堆、HiLo 寄存器中读取所需要的数据（在各寄存器模块内部已经经过旁路）。

（3）对 16 位立即数分别进行有符号扩展和无符号扩展。

（4）为了达到性能优化，在此处进行了一小部分 ALU 运算以减轻 exe_stage 负担。

（5）检测 break、syscall、和保留指令异常。

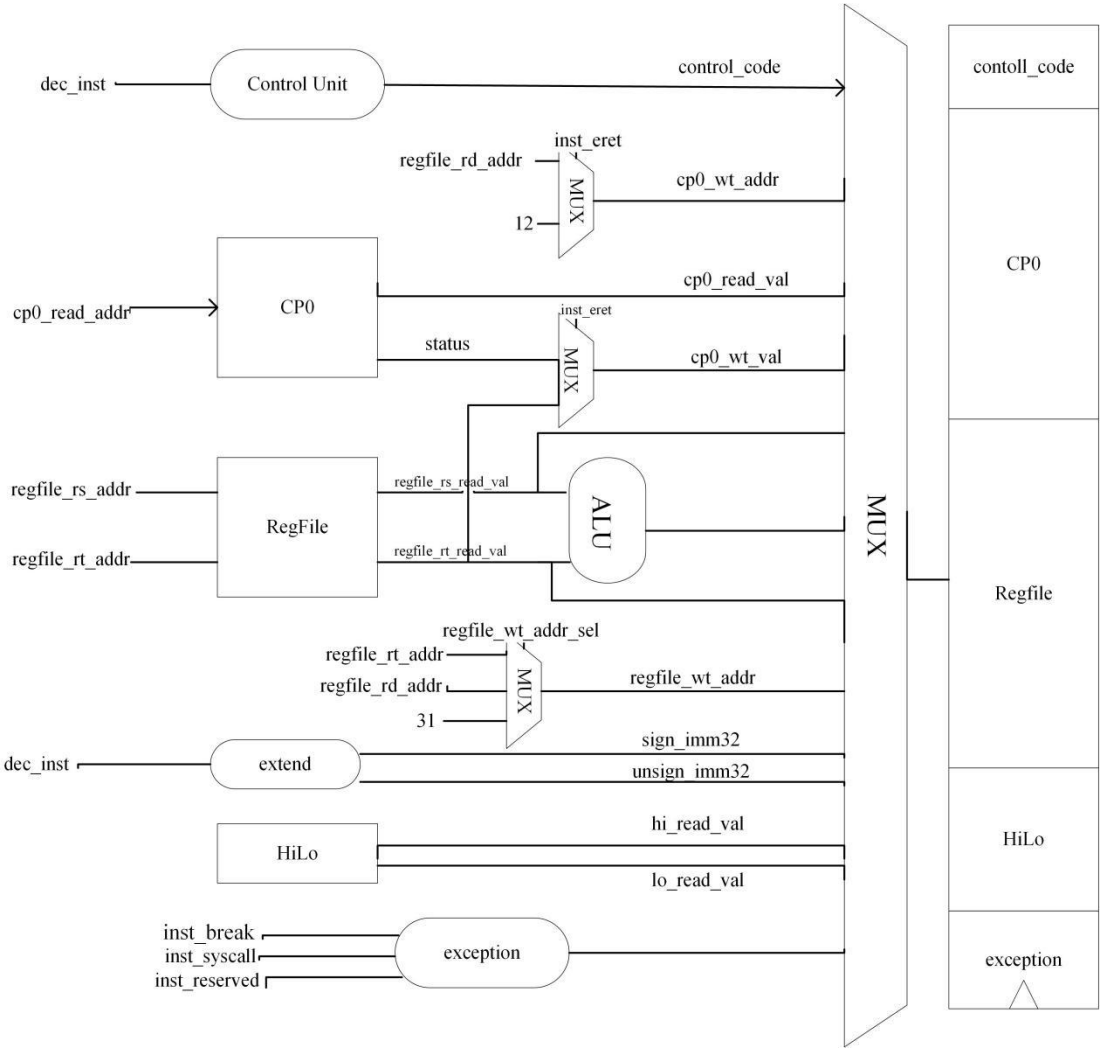


图 2　dec_stage 结构简图

（6）在 DEC 级进行了冒险检测，当 EXE 级或者 MEM 级为 lw 型指令，且写回的寄存器号

与 DEC 级此时读取的寄存器号相同时，进行阻塞。

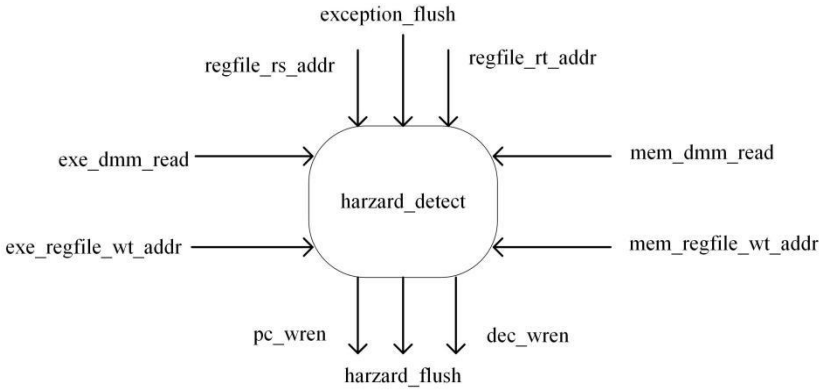


图 3　dec_stage 冒险检测单元

2、输入输出：

表 3　dec_stage 输入

名称	类型	产生模块	说明
clk	wire		时钟信号，驱动时序逻辑
reset	wire		复位信号，进行初始化
dec_inst	wire[31:0]	if_stage	指令，从 IF/DEC 流水线寄存器中读取
dec_pcplus4	wire[31:0]	if_stage	pc+4，从 IF/DEC 流水线寄存器中读取，用于计算分支跳转地址
dec_pcplus8	wire[31:0]	if_stage	pc+8，从 IF/DEC 流水线寄存器中读取，用于部分分支跳转指令保存到 31 号的地址
dec_pc	wire[31:0]	if_stage	当前指令的 pc 值，从 IF/DEC 流水线寄存器中读取，用于 trace 对比
dec_exception_if_exc happen	wire	if_stage	当前指令在 IF 级的异常信息，表示是否发生异常，从 IF/DEC 流水线寄存器中读取
dec_exception_if_epc	wire[31:0]	if_stage	当前指令在 IF 级的异常信息，表示存入 cp0 寄存器中 epc 的值，从 IF/DEC 流水线寄存器中读取
dec_exception_if_bd	wire	if_stage	当前指令在 IF 级的异常信息，表示是否为延迟槽指令，从 IF/DEC 流水线寄存器中读取
dec_exception_if_bad vaddr	wire[31:0]	if_stage	当前指令在 IF 级的异常信息，表示地址相关例外的出错地址，从 IF/DEC 流水线

			寄存器中读取
dec_exception_if_exc code	wire[4:0]	if_stage	当前指令在 IF 级的异常信息，表示异常类型，从 IF/DEC 流水线寄存器中读取
ready	wire	Memory_System	各流水线寄存器及 pc 的写使能信号，当访问存储器（取指/取数）时为 0
complete	wire	exe_stage	各流水线寄存器及 pc 的写使能信号，当进行乘除法时为 0
exception_flush	wire	mem_stage	异常发生时的各级水线寄存器的 flush 信号
regfile_rs_read_val	wire[31:0]	RegFile	经过旁路后，从寄存器堆中读取的数据
regfile_rt_read_val	wire[31:0]	RegFile	经过旁路后，从寄存器堆中读取的数据
hi_read_val	wire[31:0]	HiLo	经过旁路后，从 HiLo 寄存器中读取的数据
lo_read_val	wire[31:0]	HiLo	经过旁路后，从 HiLo 寄存器中读取的数据
cp0_read_val	wire[31:0]	CP0	经过旁路后，从 cp0 寄存器中读取的数据
cp0_status_val	wire[31:0]	CP0	经过旁路后，从 cp0 寄存器中读取 status 寄存器的值，用于 eret 清除 status 寄存器的 exl 位
mem_dmm_read	wire	exe_stage	MEM 级的控制信号，当 MEM 级为 lw 指令时取 1，用于 lw-use 型冒险检测
mem_regfile_wt_addr	wire[4:0]	exe_stage	MEM 级指令对应的目的寄存器号，用于 lw-use 型冒险检测

表 4　dec_stage 输出

名称	类型	目的模块	说明
eret_flush	wire	if_stage	当 DEC 为 eret 指令时，用于刷掉 IF/DEC 流水线寄存器中数据
bd	wire	if_stage	用于表明 DEC 级为分支跳转指令，IF 级为延迟槽指令
pc_wren	wire	if_stage	pc 寄存器的写使能信号，当发生 lw-use 型冒险时为 1
dec_wren	wire	if_stage	IF/DEC 流水线寄存器的写使能信号，当发生 lw-use 型冒险时为 1
inst_jr	wire	branch	表明是否为 JR 指令，从而确定跳转地

			址
inst_eret	wire	branch	表明是否是 eret 指令
sign_imm32	wire[31:0]	branch	32 位有符号立即数
imm26	wire[25:0]	branch	26 位立即数
branch	wire[2:0]	branch	分支指令类型
regfile_rs_addr	wire[4:0]	branch, Regfile	寄存器堆源地址
regfile_rt_addr	wire[4:0]	branch, Regfile	寄存器堆源地址
cp0_read_addr	wire[4:0]	CP0	CP0 读地址
exe_pcplus4	reg [31:0]	exe_stage	Exe 级指令的 pc+4
exe_pcplus8	reg [31:0]	exe_stage	Exe 级指令的 pc+8
exe_pc	reg [31:0]	exe_stage	Exe 级指令的 pc
exe_regfile_wren	reg	exe_stage	寄存器堆的写使能信号
exe_regfile_wt_addr	reg [4:0]	exe_stage	寄存器堆的写回地址
exe_regfile_mem2reg	reg	exe_stage	选择写回寄存堆的数据来源，为 0 选 ALU 的计算结果，为 1 选 load 取出的数据
exe_regfile_rs_read_val	reg [31:0]	exe_stage	寄存器堆源操作数
exe_regfile_rt_read_val	reg [31:0]	exe_stage	寄存器堆源操作数
exe_regfile_aluctr	reg [4:0]	exe_stage	寄存器堆通道的 alu 控制信号
exe_sra_left	reg [31:0]	exe_stage	指令 sra 的部分计算结果
exe_sra_right	reg [31:0]	exe_stage	指令 sra 的部分计算结果

exe_srav_left	reg [31:0]	exe_stage	指令 srav 的部分计算结果
exe_srav_right	reg [31:0]	exe_stage	指令 srav 的部分计算结果
exe_add_val	reg [31:0]	exe_stage	指令 add 的部分计算结果
exe_addi_val	reg [31:0]	exe_stage	指令 addi 的部分计算结果
exe_sub_val	reg [31:0]	exe_stage	指令 sub 的部分计算结果
exe_subi_val	reg [31:0]	exe_stage	指令 subi 的部分计算结果
exe_hi_wren	reg	exe_stage	Hi 寄存器的写使能信号
exe_hi_read_val	reg [31:0]	exe_stage	Hi 寄存器的值
exe_lo_wren	reg	exe_stage	Lo 寄存器的值
exe_lo_read_val	reg [31:0]	exe_stage	Lo 寄存器的值
exe_hilo_aluctr	reg [2:0]	exe_stage	HiLo 寄存器的 alu 控制信号
exe_cp0_wren	reg	exe_stage	cp0 寄存器的写使能信号
exe_cp0_wt_addr	reg [31:0]	exe_stage	cp0 寄存器的写回地址
exe_cp0_wt_val	reg [31:0]	exe_stage	cp0 寄存器的写回值
exe_cp0_read_val	reg [31:0]	exe_stage	从 cp0 寄存器读的值
exe_sign_imm32	reg [31:0]	exe_stage	32 位有符号立即数
exe_unsign_imm32	reg [31:0]	exe_stage	32 位无符号立即数
exe_branch	reg [2:0]	exe_stage	分支指令类型
exe_start_div	reg	exe_stage	标识除法指令
exe_start_mult	reg	exe_stage	标识乘法指令
exe_lw_sw_type	reg [2:0]	exe_stage	取数存数的指令类型

exe_dmm_read	reg	exe_stage	标识取数指令
exe_dmm_write	reg	exe_stage	标识存数指令
exe_exception_if_exc happen	reg	exe_stage	标识 dec 级指令在 if 级时是否发生异常
exe_exception_if_epc	reg [31:0]	exe_stage	当 dec 级指令在 if 级发生异常时需要保存的 epc
exe_exception_if_bd	reg	exe_stage	标识 dec 级指令是否是延迟槽指令
exe_exception_if_bad vaddr	reg [31:0]	exe_stage	当 dec 级指令在 if 级发生地址错例外时需要保存的 badvaddr
exe_exception_if_exc code	reg [4:0]	exe_stage	当 dec 级指令在 if 级发生例外时需要保存的 exception code
exe_exception_dec_e xchappen	reg	exe_stage	标识 dec 级指令是否在 dec 发生例外
exe_exception_dec_e xccode	reg [4:0]	exe_stage	当 dec 级指令在 dec 级发生例外时需要保存的 exception code

(四) exe_stage 模块设计

1、主要功能：

（1）将流水线根据写回的寄存器类型不同,分为 CP0 通道、RegFile 通道和 HiLo 通道。

（2）对于 CP0 通道，只需要将所需数据及控制信号传递到下一级。

（3）对于 RegFile 通道，进行 ALU 运算，并根据 exe_regfile_aluctr 选择出所需要的运算结果，保存到流水线寄存器中。

（4）对于 HiLo 通道，进行*乘除法*[2]法运算，并根据 exe_hilo_aluctr 选择出所需要的运算结果，保存到流水线寄存器中。

（5）检测算术溢出异常。

（6）根据运算结果和控制信号 lw_sw_type 来 lh 产生 byte_enable 信号，从而做好对 lb、、sb、sh 等指令的数据处理。

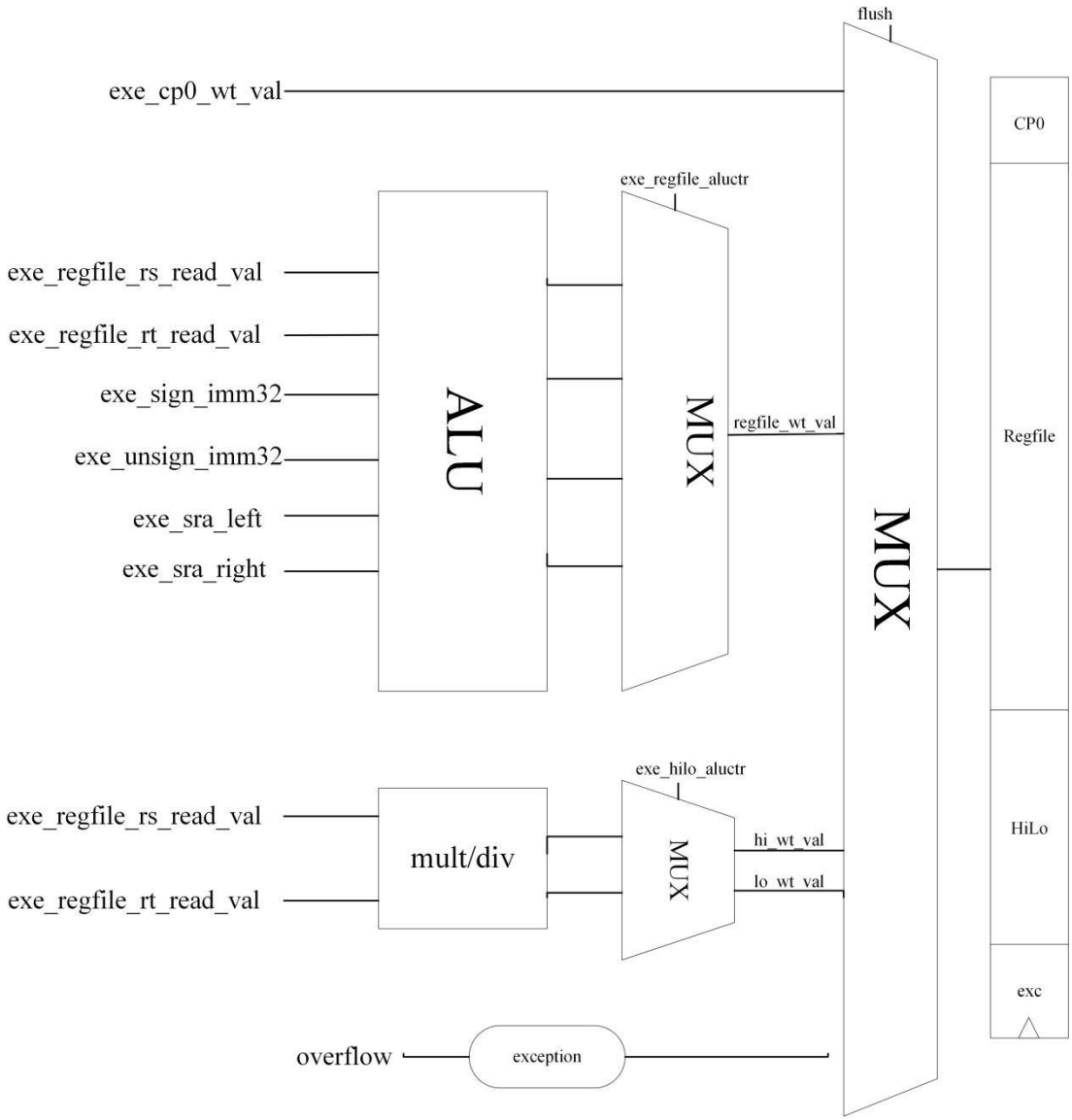


图 4　exe_stage 结构简图

2、输入输出

表 5　exe_stage 输入

名称	类型	产生模块	说明
clk	wire		时钟信号，驱动时序逻辑
reset	wire		复位信号，进行初始化
exe_pcplus4	reg [31:0]	exe_stage	Exe 级指令的 pc+4
exe_pcplus8	reg [31:0]	exe_stage	Exe 级指令的 pc+8
exe_pc	reg [31:0]	exe_stage	Exe 级指令的 pc

exe_regfile_wren	reg	exe_stage	寄存器堆的写使能信号
exe_regfile_wt_addr	reg [4:0]	exe_stage	寄存器堆的写回地址
exe_regfile_mem2reg	reg	exe_stage	选择写回寄存堆的数据来源，为 0 选 ALU 的计算结果，为 1 选 load 取出的数据
exe_regfile_rs_read_val	reg [31:0]	exe_stage	寄存器堆源操作数
exe_regfile_rt_read_val	reg [31:0]	exe_stage	寄存器堆源操作数
exe_regfile_aluctr	reg [4:0]	exe_stage	寄存器堆通道的 alu 控制信号
exe_sra_left	reg [31:0]	exe_stage	指令 sra 的部分计算结果
exe_sra_right	reg [31:0]	exe_stage	指令 sra 的部分计算结果
exe_srav_left	reg [31:0]	exe_stage	指令 srav 的部分计算结果
exe_srav_right	reg [31:0]	exe_stage	指令 srav 的部分计算结果
exe_add_val	reg [31:0]	exe_stage	指令 add 的部分计算结果
exe_addi_val	reg [31:0]	exe_stage	指令 addi 的部分计算结果
exe_sub_val	reg [31:0]	exe_stage	指令 sub 的部分计算结果
exe_subi_val	reg [31:0]	exe_stage	指令 subi 的部分计算结果
exe_hi_wren	reg	exe_stage	Hi 寄存器的写使能信号

exe_hi_read_val	reg [31:0]	exe_stage	Hi 寄存器的值
exe_lo_wren	reg	exe_stage	Lo 寄存器的值
exe_lo_read_val	reg [31:0]	exe_stage	Lo 寄存器的值
exe_hilo_aluctr	reg [2:0]	exe_stage	HiLo 寄存器的 alu 控制信号
exe_cp0_wren	reg	exe_stage	cp0 寄存器的写使能信号
exe_cp0_wt_addr	reg [31:0]	exe_stage	cp0 寄存器的写回地址
exe_cp0_wt_val	reg [31:0]	exe_stage	cp0 寄存器的写回值
exe_cp0_read_val	reg [31:0]	exe_stage	从 cp0 寄存器读的值
exe_sign_imm32	reg [31:0]	exe_stage	32 位有符号立即数
exe_unsign_imm32	reg [31:0]	exe_stage	32 位无符号立即数
exe_branch	reg [2:0]	exe_stage	分支指令类型
exe_start_div	reg	exe_stage	标识除法指令
exe_start_mult	reg	exe_stage	标识乘法指令
exe_lw_sw_type	reg [2:0]	exe_stage	取数存数的指令类型
exe_dmm_read	reg	exe_stage	标识取数指令
exe_dmm_write	reg	exe_stage	标识存数指令
exe_exception_if_exchappen	reg	exe_stage	标识 dec 级指令在 if 级时是否发生异常
exe_exception_if_epc	reg [31:0]	exe_stage	当 dec 级指令在 if 级发生异常时需要保存的 epc
exe_exception_if_bd	reg	exe_stage	标识 dec 级指令是否是延迟槽指令
exe_exception_if_badvaddr	reg [31:0]	exe_stage	当 dec 级指令在 if 级发生地址错例外时需要保存的 badvaddr
exe_exception_if_exccode	reg [4:0]	exe_stage	当 dec 级指令在 if 级发生例外时需要保存的 exception code
exe_exception_dec_exchappen	reg	exe_stage	标识 dec 级指令是否在 dec 发生例外
exe_exception_dec_exccode	reg [4:0]	exe_stage	当 dec 级指令在 dec 级发生例外时需要保存的 exception code
ready	wire	Memory_	各流水线寄存器及 pc 的写使能信

		System	号，当访问存储器（取指/取数）时为 0

表 6　exe_stage 输出

名称	类型	目的模块	说明
complete	wire	if, dec, exe, mem, wb	
regfile_wt_val	reg [31:0]	Regfile	regfile 通道的前递值
hi_wt_val	reg [31:0]	HiLo	hilo 通道的前递值
lo_wt_val	reg [31:0]	HiLo	hilo 通道的前递值
cp0_wt_val	wire [31:0]	CP0	cp0 通道的前递值
mem_pc	reg [31:0]	mem_stage	当前指令的 pc
mem_regfile_wren	reg	mem_stage	寄存器通道的写使能
mem_regfile_wt_addr	reg [4:0]	mem_stage	寄存器通道的写回地址
mem_regfile_mem2reg	reg	mem_stage	寄存器的写回值的数据来源
mem_regfile_wt_val	reg [31:0]	mem_stage	寄存器通道的写回值
mem_regfile_rt_read_val	reg [31:0]	mem_stage	存数指令的要存储的数据
mem_hi_wren	reg	mem_stage	hi 寄存器的写使能
mem_hi_wt_val	reg [31:0]	mem_stage	hi 寄存器通道的写回值
mem_lo_wren	reg	mem_stage	lo 寄存器的写使能
mem_lo_wt_val	reg [31:0]	mem_stage	lo 寄存器通道的写回值
mem_cp0_wren	reg	mem_stage	cp0 寄存器通道的写使能
mem_cp0_wt_addr	reg [4:0]	mem_stage	cp0 寄存器通道的写回地址
mem_cp0_wt_val	reg [31:0]	mem_stage	cp0 寄存器的写回值
mem_lw_sw_type	reg [2:0]	mem_stage	取数存数指令类型
mem_dmm_addr	reg [31:0]	mem_stage	取数存数指令的访存地址
mem_dmm_read	reg	mem_stage	标识取数指令
mem_dmm_write	reg	mem_stage	标识存数指令
mem_dmm_byte_enable	reg [3:0]	mem_stage	取数存数指令的字节使能信号
mem_exception_if_exchappen	reg	mem_stage	标识 exe 级指令在 if 级时是否发生异常

mem_exception_if_epc	reg [31:0]	mem_stage	当 exe 级指令在 if 级发生异常时需要保存的 epc
mem_exception_if_bd	reg	mem_stage	标识 exe 级指令是否是延迟槽指令
mem_exception_if_badvaddr	reg [31:0]	mem_stage	当 exe 级指令在 if 级发生地址错例外时需要保存的 badvaddr
mem_exception_exccode	reg [4:0]	mem_stage	当 exe 级指令在 if 级发生例外时需要保存的 exception code
mem_exception_dec_exchappen	reg	mem_stage	标识 exe 级指令在 dec 级时是否发生异常
mem_exception_dec_exccode	reg [4:0]	mem_stage	当 exe 级指令在 dec 级发生例外时需要保存的 exception code
mem_exception_exe_exchappen	reg	mem_stage	标识 exe 级指令在 exe 级时是否发生异常
mem_exception_exe_exccode	reg [4:0]	mem_stage	当 exe 级指令在 exe 级发生例外时需要保存的 exception code

(五) mem_stage 模块

1、主要功能：

（1）对于 CP0 通道，只需将所需要的数据以及控制信号传递到下一级流水线寄存器中。

（2）对于 RegFile 通道，根据控制信号以及地址段，与 DCache 或者存储器交互，进行数据的存取。

（3）对于 HiLo 通道，只需将所需要的数据以及控制信号传递到下一级流水线寄存器中。

（4）在 MEM 级收集异常信息，并进行异常信息的处理，产生 exception_flush 等信号，对于要写回 CP0 的异常信息，将其随流水线一起流到下一级。

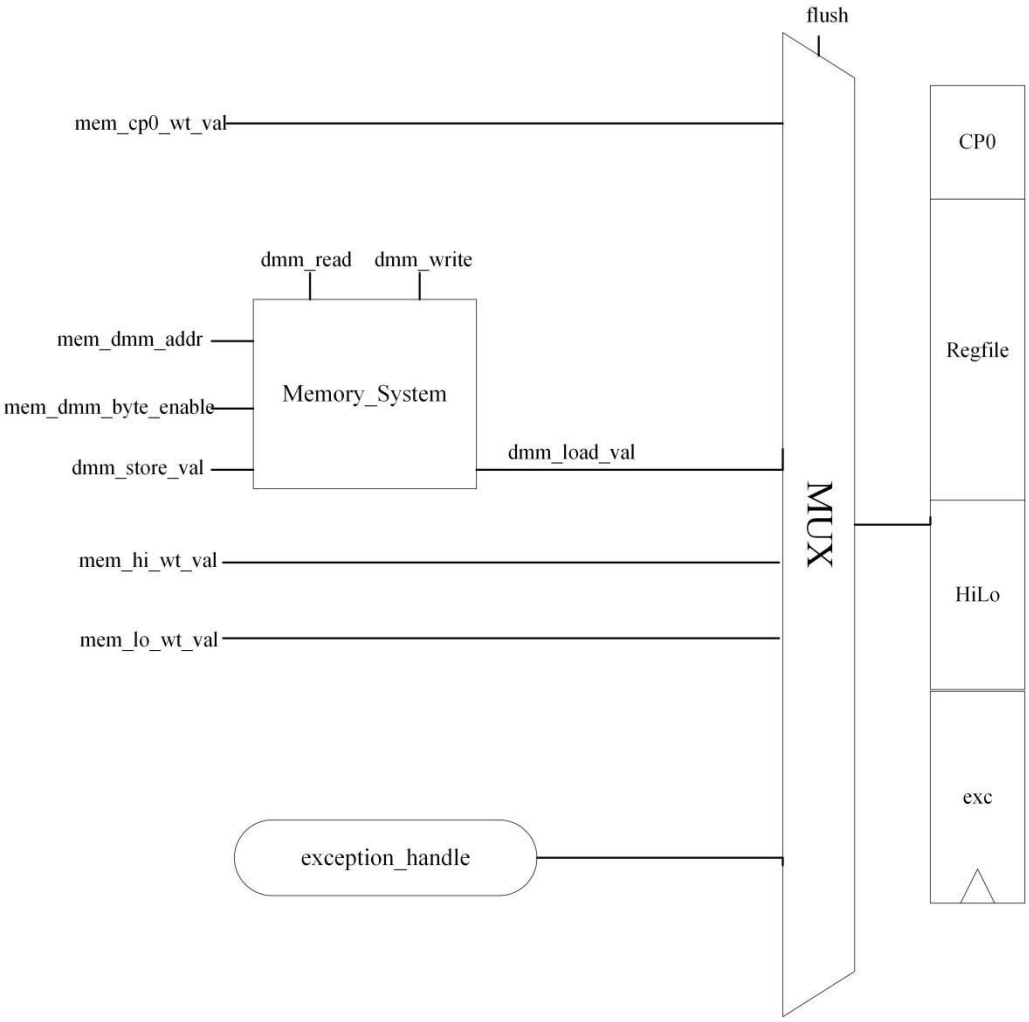


图 5　mem_stage 结构简图

2、输入输出:

表 7　mem_stage 输入

名称	类型	产生模块	说明
clk	wire		时钟信号，驱动时序逻辑
reset	wire		复位信号，进行初始化
mem_pc	reg [31:0]	mem_stage	当前指令的 pc
mem_regfile_wren	reg	mem_stage	寄存器通道的写使能
mem_regfile_wt_addr	reg [4:0]	mem_stage	寄存器通道的写回地址
mem_regfile_mem2reg	reg	mem_stage	寄存器的写回值的数据来源
mem_regfile_wt_val	reg [31:0]	mem_stage	寄存器通道的写回值
mem_regfile_rt_read_val	reg [31:0]	mem_stage	存数指令的要存储的数据

mem_hi_wren	reg	mem_stage	hi 寄存器的写使能
mem_hi_wt_val	reg [31:0]	mem_stage	hi 寄存器通道的写回值
mem_lo_wren	reg	mem_stage	lo 寄存器的写使能
mem_lo_wt_val	reg [31:0]	mem_stage	lo 寄存器通道的写回值
mem_cp0_wren	reg	mem_stage	cp0 寄存器通道的写使能
mem_cp0_wt_addr	reg [4:0]	mem_stage	cp0 寄存器通道的写回地址
mem_cp0_wt_val	reg [31:0]	mem_stage	cp0 寄存器的写回值
mem_lw_sw_type	reg [2:0]	mem_stage	取数存数指令类型
mem_dmm_addr	reg [31:0]	mem_stage	取数存数指令的访存地址
mem_dmm_read	reg	mem_stage	标识取数指令
mem_dmm_write	reg	mem_stage	标识存数指令
mem_dmm_byte_enable	reg [3:0]	mem_stage	取数存数指令的字节使能信号
mem_exception_if_exchappen	reg	mem_stage	标识 exe 级指令在 if 级时是否发生异常
mem_exception_if_epc	reg [31:0]	mem_stage	当 exe 级指令在 if 级发生异常时需要保存的 epc
mem_exception_if_bd	reg	mem_stage	标识 exe 级指令是否是延迟槽指令
mem_exception_if_badvaddr	reg [31:0]	mem_stage	当 exe 级指令在 if 级发生地址错例外时需要保存的 badvaddr
mem_exception_exccode	reg [4:0]	mem_stage	当 exe 级指令在 if 级发生例外时需要保存的 exception code
mem_exception_dec_exchappen	reg	mem_stage	标识 exe 级指令在 dec 级时是否发生异常
mem_exception_dec_exccode	reg [4:0]	mem_stage	当 exe 级指令在 dec 级发生例外时需要保存的 exception code
mem_exception_exe_exchappen	reg	mem_stage	标识 exe 级指令在 exe 级时是否发生异常
mem_exception_exe_exccode	reg [4:0]	mem_stage	当 exe 级指令在 exe 级发生例外时需要保存的 exception code

cp0_status_exl	wire	CP0	cp0 status 寄存器的 exl 位
cp0_status_ie	wire	CP0	cp0 status 寄存器的 ie 位
cp0_status_im0	wire	CP0	cp0 status 寄存器的 im0 位
cp0_status_im1	wire	CP0	cp0 status 寄存器的 im1 位
cp0_cause_ip0	wire	CP0	cp0 cause 寄存器的 ip0 位
cp0_cause_ip1	wire	CP0	cp0 cause 寄存器的 ip1 位
ready	wire	Memory_System	各流水线寄存器及 pc 的写使能信号，当访问存储器（取指/取数）时为 0
complete	wire	exe_stage	各流水线寄存器及 pc 的写使能信号，当进行乘除法时为 0
dmm_load_val	wire [31:0]	Dcache	取数指令从 cache 中取回的数据

表 8　mem_stage 输出

名称	类型	目的模块	说明
exception_inst_exchappen	wire	Dcache	标识 mem 级指令是否发生异常
exception_flush	wire	if, dec, exe, mem	异常发生时的各级流水线寄存器的 flush 信号
exception_inst_interrupt	wire	CP0	标识 mem 级指令是否发生中断
wb_exception_inst_exchappen	reg	wb_stage	标识 mem 级指令是否发生异常
wb_exception_inst_epc	reg [31:0]	wb_stage	当mem级指令发生异常时需要保存的epc
wb_exception_inst_bd	reg	wb_stage	当 mem 级指令发生异常时需要保存的 branch delay 位
wb_exception_inst_badvaddr	reg [31:0]	wb_stage	当 mem 级指令发生异常时需要保存的 badvaddr 的值
wb_exception_inst_badvaddr_wren,	reg [31:0]	wb_stage	当 mem 级指令发生异常时 badvaddr 寄存器的写使能信号
wb_exception_inst_excode	reg [4:0]	wb_stage	当 mem 级指令发生异常时需要保存的 exception code

wb_pc	reg [31:0]	wb_stage	当 mem 级指令发生异常时需要保存的 exception code
wb_regfile_wren	reg	wb_stage	寄存器通道的写使能信号
wb_regfile_wt_addr	reg [4:0]	wb_stage	寄存器通道的写回地址
wb_regfile_mem2reg	reg	wb_stage	寄存器通道的写回值的数据来源
wb_regfile_wt_val	reg [31:0]	wb_stage	寄存器通道的写回值
wb_dmm_load_val	reg [31:0]	wb_stage	取数指令 dcache 中取出的数据
wb_dmm_byte_enable	reg [3:0]	wb_stage	取数指令的字节使能信号, 用于拼接数据
wb_lw_sw_type	reg [2:0]	wb_stage	取数存数指令类型
wb_hi_wren	reg	wb_stage	hi 寄存器的写使能信号
wb_hi_wt_val	reg [31:0]	wb_stage	hi 寄存器的写回值
wb_lo_wren	reg	wb_stage	lo 寄存器的写使能信号
wb_lo_wt_val	reg [31:0]	wb_stage	lo 寄存器的写回值
wb_cp0_wren	reg	wb_stage	cp0 寄存器的写使能信号
wb_cp0_wt_addr	reg [4:0]	wb_stage	cp0 寄存器的写回地址
wb_cp0_wt_val	reg [31:0]	wb_stage	cp0 寄存器的写回值

(六) wb_stage 模块设计

1、主要功能：

（1）将要写回各个寄存器的数据传递给相应的寄存器堆。

（2）根据 wb_dmm_byte_enable 信号对 MEM 级取出的数据进行处理。

（3）生成 trace 对比所需要的信号。

2、输入输出:

表 9 wb_stage 输入

名称	类型	产生模块	说明
clk	wire		时钟信号，驱动时序逻辑
reset	wire		复位信号，进行初始化
wb_regfile_wren	wire	mem_stage	寄存器堆通道的写使能信号

wb_regile_wt_addr	wire [4:0]	mem_stage	寄存器堆通道的写回地址
wb_regfile_mem2reg	wire	mem_stage	寄存器堆写回值的数据来源
wb_regfile_wt_val	wire [31:0]	mem_stage	寄存器堆通道的写回值
wb_dmm_load_val	wire [31:0]	mem_stage	取数指令从 Dcache 中取回的数据
wb_dmm_byte_enable	wire [3:0]	mem_stage	取数指令的字节使能信号
wb_lw_sw_type	wire [2:0]	mem_stage	取数存数指令类型
wb_pc	wire [31:0]	mem_stage	wb 级指令的 pc
ready	wire	Memory_System	各流水线寄存器及 pc 的写使能信号，当访问存储器（取指/取数）时为 0
complete	wire	exe_stage	各流水线寄存器及 pc 的写使能信号，当进行乘除法时为 0

表 10　wb_stage 输出

名称	类型	目的模块	说明
wb_regfile_wt_val_mux	wire [31:0]	Regfile	寄存器通道的写回值
debug_wb_pc	wire [31:0]	tb_top	用于 trace 对比的调试信号
debug_wb_rf_wren	wire [3:0]	tb_top	用于 trace 对比的调试信号
debug_wb_rf_wnum	wire [4:0]	tb_top	用于 trace 对比的调试信号
debug_wb_rf_wdata	wire [31:0]	tb_top	用于 trace 对比的调试信号

(七) branch 模块设计

1、主要功能：

根据指令信息、异常信息等产生 PCSrc 信号及 branch_target,从而决定 PC 的取值。

2、输入输出：

表 11　branch 输入

名称	类型	产生模块	说明
dec_pcplus4	wire [31:0]	if_stage	dec 级指令的 pc+4
branch	wire [2:0]	dec_stage	分支指令的类型

inst_jr	wire	dec_stage	标识 dec 级指令是否是 jr 指令
inst_eret	wire	dec_stage	标识 dec 级指令是否是 eret 指令
exception_flush	wire	mem_stage	发生异常时的 flush 信号
sign_imm32	wire [31:0]	dec_stage	32 位有符号立即数
imm26	wire [25:0]	dec_stage	26 位立即数
regfile_rs_read_val	wire [31:0]	Regfile	寄存器堆源操作数
regfile_rt_read_val	wire [31:0]	Regfile	寄存器堆源操作数

表 12　branch 输出

名称	类型	目的模块	说明
PCSrc	reg [1:0]	if_stage	pc 寄存器的选择信号
branch_target	wire [31:0]	if_stage	分支指令的跳转地址

(八) Regfile 模块

1、主要功能：

（1）对于从寄存器堆中读取数据，通过从 EXE、MEM、WB 进行旁路，从而读取出正确的数据。

（2）承担写回寄存器堆，保存数据的功能。

2、输入输出：

表 13　Regfile 输入

名称	类型	产生模块	说明
clk	wire		时钟信号，驱动时序逻辑
reset	wire		复位信号，进行初始化
regfile_rs_addr	wire [4:0]	dec_stage	寄存器堆读地址
regfile_rt_addr	wire [4:0]	dec_stage	寄存器堆读地址
wb_regfile_wren	wire	wb_stage	wb 级指令寄存器堆的写使能信号
wb_regfile_wt_addr	wire [4:0]	wb_stage	wb 级指令寄存器堆的写回地址
wb_regfile_wt_val	wire [31:0]	wb_stage	wb 级指令寄存器堆通道的写回值

mem_regfile_wren	wire	mem_stage	mem 级指令寄存器堆的写使能
mem_regfile_wt_addr	wire [4:0]	mem_stage	mem 级指令寄存器堆的写回地址
mem_regfile_wt_val	wire [31:0]	mem_stage	mem 级指令的寄存器堆写回值
exe_regfile_wren	wire	exe_stage	exe 级指令的寄存器堆写使能信号
exe_regfile_wt_addr	wire [4:0]	exe_stage	exe 级指令的寄存器堆写使能信号
exe_regfile_wt_val	wire [31:0]	exe_stage	exe 级指令的寄存器堆写回值

表 14　Regfile 输出

名称	类型	目的模块	说明
regfile_rs_read_val	reg [31:0]	dec_stage	经过旁路选择后的寄存器堆源操作数
regfile_rt_read_val	reg [31:0]	dec_stage	经过旁路选择后的寄存器堆源操作数

(九) HiLo 模块设计

1、主要功能：

（1）对于读取数据，通过从 EXE、MEM、WB 级进行旁路从而读取出正确的数据。

（2）承担乘除法以及 MTHI、MTLO 等指令写回 HiLo 数据的保存。

2、输入输出：

表 15　HiLo 输入

名称	类型	产生模块	说明
clk	wire		时钟信号，驱动时序逻辑
reset	wire		复位信号，进行初始化
wb_hi_wren	wire	wb_stage	wb 级指令的 hi 寄存器写使能信号
wb_lo_wren	wire	wb_stage	wb 级指令的的 lo 寄存器写使能信号
wb_hi_wt_val	wire [31:0]	wb_stage	wb 级指令的 hi 寄存器写回值
wb_lo_wt_val	wire [31:0]	wb_stage	wb 级指令的 lo 寄存器写回值
mem_hi_wren	wire	mem_stage	mem 级指令的 hi 寄存器写回值
mem_lo_wren	wire	mem_stage	mem 级指令的 lo 寄存器写使能信号
mem_hi_wt_val	wire [31:0]]	mem_stage	mem 级指令的 hi 寄存器写回值
mem_lo_wt_val	wire [31:0]	mem_stage	mem 级指令的 lo 寄存器写回值

exe_hi_wren	wire	exe_stage	exe 级指令的 hi 寄存器写回值
exe_lo_wren	wire	exe_stage	exe 级指令的 lo 寄存器写回值
exe_hi_wt_val	wire [31:0]	exe_stage	exe 级指令的 hi 寄存器写回值
exe_lo_wt_val	wire [31:0]	exe_stage	exe 级指令的 lo 寄存器写回值

表 16　HiLo 输出

名称	类型	目的模块	说明
hi_read_val	wire [31:0]	dec_stage	经过旁路选择后的 hi 寄存器的值
lo_read_val	wire [31:0]	dec_stage	经过旁路选择后的 lo 寄存器的值

(十)　CP0 模块设计

1、主要功能：

（1）对于读取数据，通过从 EXE、MEM、WB 旁路从而得到正确的数据。

（2）包括中断、异常等信息的保存。

（3）实现 MTC0 的写回。

表 17　CP0 输入

名称	类型	产生模块	说明
clk	wire		时钟信号，驱动时序逻辑
reset	wire		复位信号，进行初始化
exception_inst_interrupt	wire	mem_stage	标识是否检测到中断
wb_exception_inst_exchappen	wire	wb_stage	标识是否检测到异常
wb_exception_inst_epc	wire [31:0]	wb_stage	异常和中断发生发生时需要保存的 epc
wb_exception_inst_bd	wire	wb_stage	标识异常指令是否是延迟槽指令
wb_exception_inst_exccode	wire [4:0]	wb_stage	异常发生时需要保存的 exception code
wb_exception_badvaddr	wire [31:0]	wb_stage	异常发生时需要保存的 badvaddr
wb_exception_badvaddr_	wire	wb_stage	发生异常时 badvaddr 寄存器的写使

wren			能信号
cp0_read_addr	wire [4:0]	dec_stage	cp0 寄存器的读地址
wb_cp0_wren	wire	wb_stage	wb 级 cp0 寄存器的写使能信号
wb_cp0_wt_addr	wire [4:0]	wb_stage	wb 级 cp0 寄存器的写回地址
wb_cp0_wt_val	wire [31:0]	wb_stage	wb 级 cp0 寄存器的写回值
mem_cp0_wren	wire	mem_stage	mem 级 cp0 寄存器的写使能信号
mem_cp0_wt_addr	wire	mem_stage	mem 级 cp0 寄存器的写使能信号
mem_cp0_wt_val	wire [31:0]	mem_stage	mem 级 cp0 寄存器的写回值
exe_cp0_wren	wire	exe_stage	exe 级 cp0 寄存器的写使能信号
exe_cp0_wt_addr	wire [4:0]	exe_stage	exe 级 cp0 寄存器的写回地址
exe_cp0_wt_val	wire [31:0]	exe_stage	exe 级 cp0 寄存器的写回值
ready	wire	Memory_System	各流水线寄存器及 pc 的写使能信号，当访问存储器（取指/取数）时为 0
complete	wire	exe_stage	各流水线寄存器及 pc 的写使能信号，当进行乘除法时为 0

表 18　CP0 输出

名称	类型	目的模块	说明
cp0_read_val	wire [31:0]	dec_stage	经过旁路选择后的 cp0 寄存器的值
cp0_epc_val	wire [31:0]	if_stage	epc 寄存器的值
cp0_status_val	wire [31:0]	mem_stage	status 寄存器的值
cp0_status_exl	wire	mem_stage	status 寄存器的 exl 位
cp0_status_im0	wire	mem_stage	status 寄存器的 im0 位
cp0_cause_ip0	wire	mem_stage	status 寄存器的 ip0 位
cp0_cause_ip1	wire	mem_stage	status 寄存器的 ip1 位

(十一) ICache 模块设计

1、主要功能：

（1）根据 PC 值从 ICache 读取出所需指令，或直接访问存储器读取指令。

（2）对于 Cached 访问，若 Cache 命中，则当拍给出所需指令，否则根据*状态机*[1]进行数据分配。

（3）对于 Uncached 访问，根据状态机，通过 AXI 接口与存储器进行直接交互，获取所需指令。

注：为了更加高效的利用资源，并且保证性能，使用 Xilinx 提供的 DRAM IP 核来作为 CacheLine 以及 Tag 域。

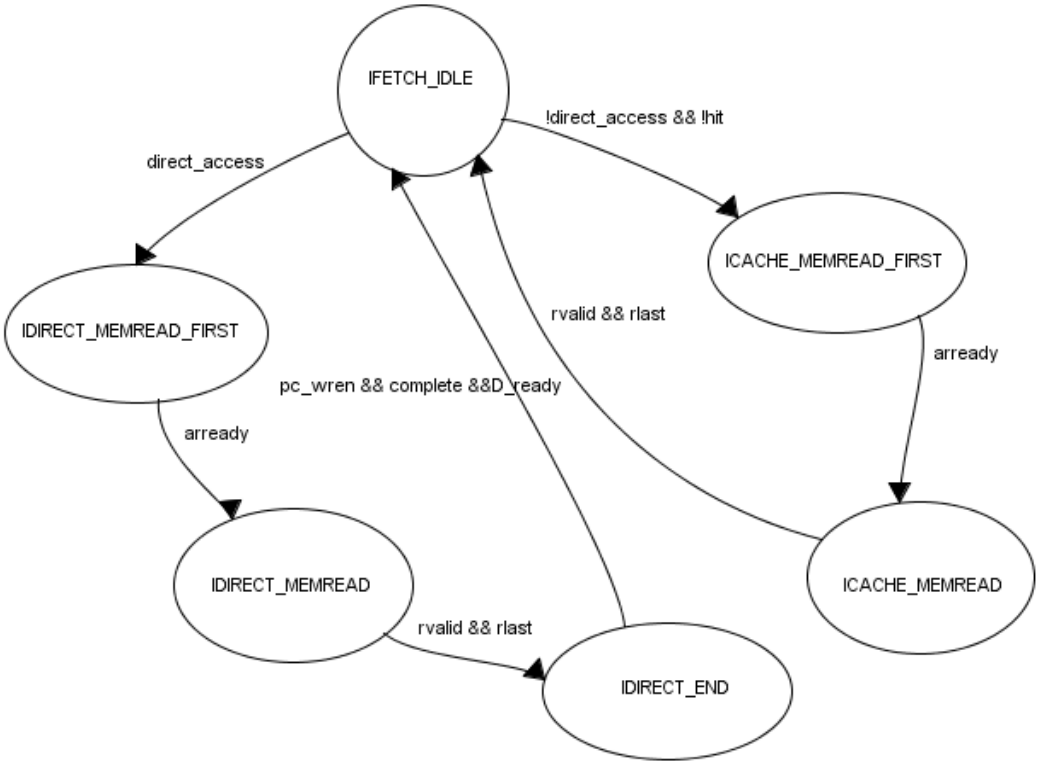


图 6　ICache 状态机

2、输入输出：

表 19　ICache 输入

名称	类型	产生模块	说明
clk	wire		时钟信号，驱动时序逻辑
reset	wire		复位信号，进行初始化
pc	wire [31:0]	if_stage	pc 寄存器的值
pc_wren	wire	if_stage	pc 寄存器的写使能信号
complete	wire	exe_stage	各流水线寄存器及 pc 的写使能信号，当进行乘除法时为 0

D_ready	wire	Dcache	Dcache 模块中的数据是否有效
arready	wire	AXI	AXI 接口信号
rvalid	wire	AXI	AXI 接口信号
rlast	wire	AXI	AXI 接口信号
rdata	wrie [31:0]	AXI	AXI 接口信号

表 20　ICache 输出

名称	类型	目的模块	说明
icache_inst	wire [31:0]	if_stage	从存储器或 cache 中读出的指令
ready	wire	if, dec, exe, mem, cp0	标识 icache 中的数据是否有效
araddr	reg [31:0]	AXI	AXI 接口信号
arlen	reg [3:0]	AXI	AXI 接口信号
arvalid	reg	AXI	AXI 接口信号
not_ifetch_idle	wire	Memory_System	标识 Icache 的状态机是否处于 ifetch_idle 状态
not_idirect_end	wire	Memory_System	标识 Icache 的状态是否处于 idirect_end 状态

(十二) Dcache 模块设计

1、主要功能：

（1）根据 mem_dmm_addr 对 DCache 进行数据的存取，或者直接与存储器通过 AXI 接口交互，进行数据的存取。

（2）对于 Cache 访问：

若命中，则当拍进行存取，但是对于 store 指令 sb、sh，为了缩短路径长度，根据状态机进行两个周期的处理，第一个周期读取出相应地址的数据，第二个周期首先进行拼接，再进行 store；

若未命中，则根据 CacheLine 的 dirty 位来确定是进入分配状态，还是写回状态：

若 dirty 位为 1，则进行写回，将 CacheLine 中的数据先写回存储器中，然后进行分配。若 dirty 位为 0，则直接进行分配。

（3）对于 UnCache 访问：

根据是 load 还是 store 分别进入相应的状态与存储器直接交互。

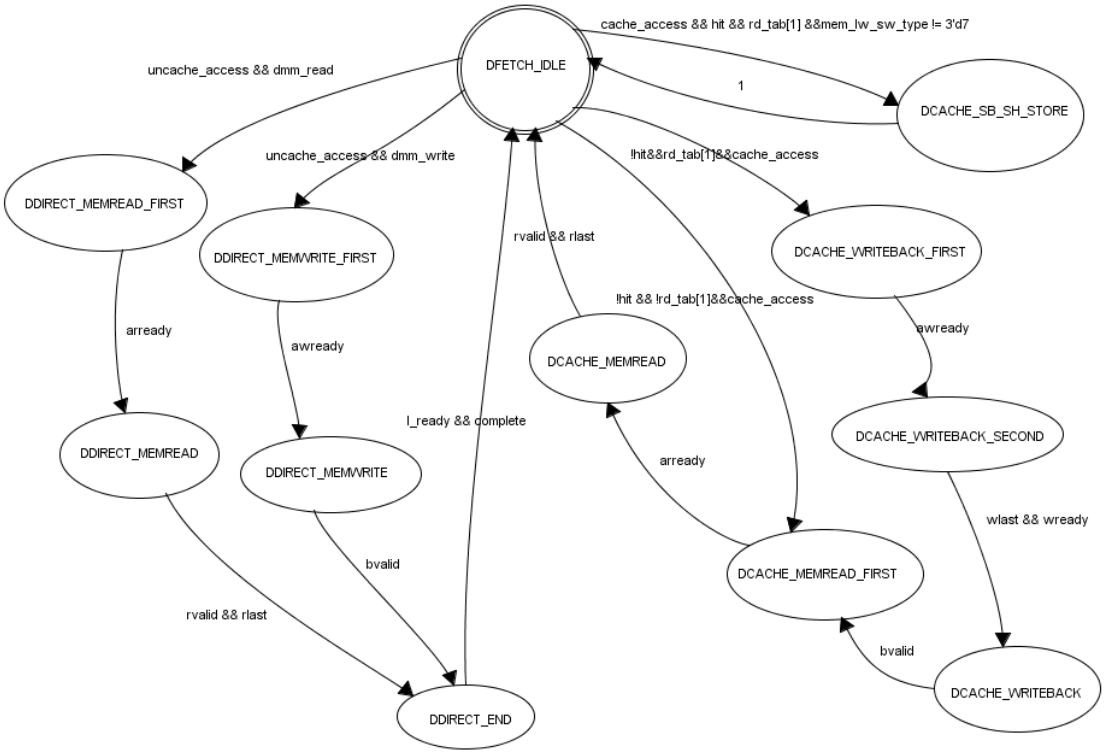


图 7　DCache 状态机

2、输入输出：

表 21　DCache 输入

名称	类型	产生模块	说明
clk	wire		时钟信号，驱动时序逻辑
reset	wire		复位信号，进行初始化
complete	wire	exe_stage	各流水线寄存器及 pc 的写使能信号，当进行乘除法时为 0
I_ready	wire	Icache	标识 Icache 模块中的数据是否有效
exception_inst_exchappen	wire	mem_stage	标识 mem 级指令是否发生异常
dmm_write	wire	mem_stage	标识 mem 级指令是否是存数指令
dmm_read	wire	mem_stage	标识 mem 级指令是否是取数指令

wen	wire [3:0]	mem_stage	取数存数指令的字节使能信号
mem_dmm_addr	wire [31:0]	mem_stage	访存地址
mem_lw_sw_type	wire [2:0]	mem_stage	取数存数指令类型
dmm_data	wire[31:0]	mem_stage	存数指令要存储的数据
arready	wire	AXI	AXI 接口信号
awready	wire	AXI	AXI 接口信号
wready	wire	AXI	AXI 接口信号
rvalid	wire	AXI	AXI 接口信号
rlast	wire	AXI	AXI 接口信号
rdata	wire [31:0]	AXI	AXI 接口信号
bvalid	wire	AXI	AXI 接口信号

表 22 DCache 输出

名称	类型	目的模块	说明
araddr	reg [31:0]	AXI	AXI 接口信号
arlen	reg [3:0]	AXI	AXI 接口信号
arvalid	reg	AXI	AXI 接口信号
awaddr	reg [31:0]	AXI	AXI 接口信号
awlen	reg [3:0]	AXI	AXI 接口信号
awvalid	reg	AXI	AXI 接口信号
wdata	reg [31:0]	AXI	AXI 接口信号
wlast	reg	AXI	AXI 接口信号
wvalid	reg	AXI	AXI 接口信号
ready	wire	Memory_System	标识 Dcache 模块中的数据是否有效
dcache_rdata	wire [31:0]	mem_stage	从 Dcache 或存储中读出的数据
not_dfetch_idle	wire	Memory_System	标识 Dcache 模块的状态机是否处于 dfetch_idle 信号

(十三) Memory_System 模块设计

1、主要功能：

（1）管理 DCache 和 ICache 模块，得到所需指令和数据。

（2）因对外只有一个 AXI 接口，所以在此模块中根据状态进行仲裁，从而确定与 AXI 接口进行交互的对象。

2、输入输出：

表 23　Memory_System 输入

名称	类型	产生模块	说明
clk	wire		时钟信号，驱动时序逻辑
reset	wire		复位信号，进行初始化
pc	wire [31:0]	if_stage	pc 寄存器的值
pc_wren	wire	if_stage	pc 寄存器的写使能信号
complete	wire	exe_stage	各流水线寄存器及 pc 的写使能信号，当进行乘除法时为 0
exception_inst_exchappen	wire	mem_stage	标识 mem 级指令是否发生异常
mem_dmm_addr	wire [31:0]	mem_stage	访存地址
mem_lw_sw_type	wire [2:0]	mem_stage	取数存数指令类型
dmm_read	wire	mem_stage	标识 mem 级指令是否是取数指令
dmm_write	wire	mem_stage	标识 mem 级指令是否是存数指令
wen	wire [3:0]	mem_stage	取数存数指令的字节使能信号
store_val	wire [31:0]	mem_stage	存数指令要存储的数据
arready	wire	AXI	AXI 接口信号
rid	wire [3:0]	AXI	AXI 接口信号
rdata	wire [31:0]	AXI	AXI 接口信号
rresp	wire [1:0]	AXI	AXI 接口信号
rlast	wire	AXI	AXI 接口信号
awready	wire	AXI	AXI 接口信号
wready	wire	AXI	AXI 接口信号

bid	wire [3:0]	AXI	AXI 接口信号
bresp	wire [1:0]	AXI	AXI 接口信号
bvalid	wire	AXI	AXI 接口信号

表 24　Memory_System 输出

名称	类型	目的模块	说明
arid	wire [3:0]	AXI	AXI 接口信号
araddr	wire [31:0]	AXI	AXI 接口信号
arlen	wire [3:0]	AXI	AXI 接口信号
arsize	wire [2:0]	AXI	AXI 接口信号
arburst	wire [1:0]	AXI	AXI 接口信号
arlock	wire [1:0]	AXI	AXI 接口信号
arcache	wire [3:0]	AXI	AXI 接口信号
arprot	wire [2:0]	AXI	AXI 接口信号
arvalid	wire	AXI	AXI 接口信号
rready	wire	AXI	AXI 接口信号
awid	wire [3:0]	AXI	AXI 接口信号
awaddr	wire [31:0]	AXI	AXI 接口信号
awlen	wir [3:0]	AXI	AXI 接口信号
awsize	wire [2:0]	AXI	AXI 接口信号
awburst	wire [1:0]	AXI	AXI 接口信号
awlock	wire [1:0]	AXI	AXI 接口信号
awcache	wire [3:0]	AXI	AXI 接口信号
awprot	wire [2:0]	AXI	AXI 接口信号
wid	wire [3:0]	AXI	AXI 接口信号
wdata	wire [31:0]	AXI	AXI 接口信号
wstrb	wire [3:0]	AXI	AXI 接口信号
wlast	wire	AXI	AXI 接口信号
wvalid	wire	AXI	AXI 接口信号
bready	wire	AXI	AXI 接口信号

ready	wire	if, dec, exe, mem, cp0	各流水线寄存器及 pc 的写使能信号，当访问存储器（取指/取数）时为 0
icache_inst	wire [31:0]	if_stage	从存储器或 cache 中读出的指令
dcache_rdata	wire [31:0]	mem_stage	从 Dcache 或存储中读出的数据

三、设计结果

（一）设计交付物说明

1、目录:

```
| -score.xls
| -design.pdf
| -readme.txt     运行仿真,综合,实现等的计算机配置信息
| -soc_axi_func /
|          | --rtl /
|          |          | --soc_axi_lite_top.v
|          |          | --myCPU /    包含自实现 cpu 源代码和 8 个 xilinx ip.
|          |          | --CONFREG /
|          |          | --BRIDGE /
|          |          | --xilinx_ip /
|          | --testbench /
|          |          | --mycpu_tb.v
|          | --run_vivado /
|          |          | --soc_lite.xdc
|          |          | --mycpu_prj1 /
|          |          |          | --mycpu.xpr
|          |          |          | --func.bit
|          |          |          | --memory.bit
| -soc_axi_perf /
|          | --rtl /
|          |          | --soc_axi_lite_top.v
|          |          | --myCPU /
|          |          | --CONFREG /
|          |          | --ram_wrap /
```

```
|       |       | --xilinx_ip /
|       | --tesetbench /
|       |       | --myCPU_tb.v
|       | --run_vivado /
|       |       | --soc_lite.xdc
|       |       | --mycpu_prj1 /
|       |       | --run_allbench.tcl
|       |       |       | --mycpu.xpr
|       |       |       | -perf.bit
| -soft /
|       | --func/
|       | --func /
|       | --memory_game /
|       | --perf_func /
```

2、操作提示:

提交工程中，功能测试 IP 核 axi_ram 的默认 coe 文件为 89 个功能测试点的 inst_ram.coe，若要生成 memory_game 的比特流文件，需将 axi_ram 重新定制，选择 memory_game 对应的 coe 文件；性能测试 IP 核 axi_ram 的默认 coe 文件为

allbench 对应的 coe 文件，若要对单个性能测试进行仿真、综合、实现，需重新定制 axi_ram，选择对应的 coe 文件。

对于相应的测试，只需点击对应测试目录下的 mycpu.xpr 文件，进入对应工程，直接点击 Run Simulation 进行仿真，点击 Run Synthesis 进行综合，点击 Run Implementation 进行实现，点击 Generate Bitstream 进行比特流的生成，注意根据需要更换 axi_ram 的 coe 文件。

（二）设计演示结果

myCPU 直接使用 AXI 接口：

功能测试频率为 50MHZ，成功通过 func 和 memory_game；

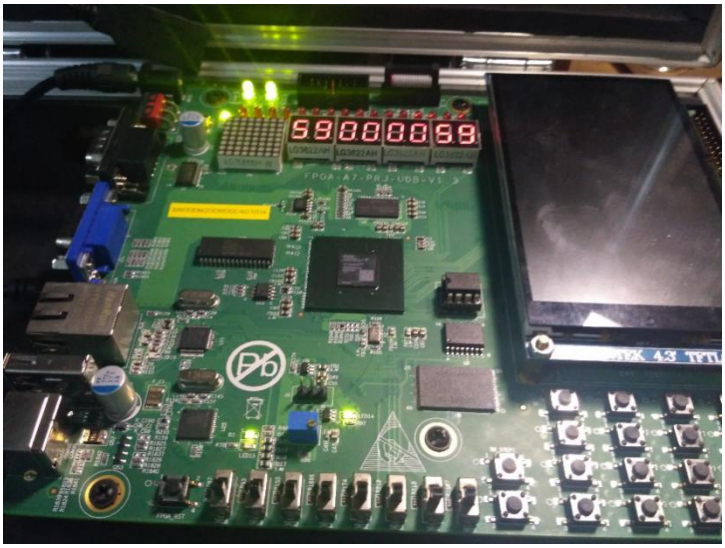
性能测试频率为 125MHZ，成功通过所有性能测试，最终得分 70.773 分。

1、功能测试仿真结果

```
        [18122000 ns] Test is running, debug_wb_pc = 0x00000000
------[18125875 ns] Number 8'd88 Functional Test Point PASS!!!
        [18132000 ns] Test is running, debug_wb_pc = 0x00000000
        [18142000 ns] Test is running, debug_wb_pc = 0x00000000
        [18152000 ns] Test is running, debug_wb_pc = 0x00000000
        [18162000 ns] Test is running, debug_wb_pc = 0x00000000
        [18172000 ns] Test is running, debug_wb_pc = 0x00000000
        [18182000 ns] Test is running, debug_wb_pc = 0xbfc00394
        [18192000 ns] Test is running, debug_wb_pc = 0x00000000
        [18202000 ns] Test is running, debug_wb_pc = 0xbfc003c0
        [18212000 ns] Test is running, debug_wb_pc = 0x00000000
------[18217395 ns] Number 8'd89 Functional Test Point PASS!!!
        [18222000 ns] Test is running, debug_wb_pc = 0x00000000
================================================================

Test end!
------PASS!!!
```

2、功能测试上板结果



3、记忆游戏上板结果

4、性能测试仿真结果

```
bitcount PASS!Bits: 811

bitcount: Total Count = 0x878d

=======================================
Test end!
----PASS!!!
```

图 1 bitcount

```
Test begin!
bubble sort test begin.

bubble sort PASS!

bubble sort: Total Count = 0x2f403

=======================================
Test end!
----PASS!!!
```

图 2 bubble_sort

```
coremark PASS!

coremark: Total Count = 0x8f67c

=================================
Test end!
----PASS!!!
```

图 3 coremark

```
crc32 PASS!

crc32: Total Count = 0x54561

=================================
Test end!
----PASS!!!
```

图 4 crc32

```
dhrystone PASS!

dhrystone: Total Count = 0x117b1

=======================================
Test end!
----PASS!!!
```

图 5 dhrystone

```
Test begin!
quick sort test begin.

quick sort PASS!

quick sort: Count = 0x331a2

=====================================
Test end!
----PASS!!!
```

图 6 quick_sort

```
Test begin!
select sort test begin.

select sort PASS!

select sort: Count = 0x30fd8

=====================================
Test end!
----PASS!!!
```

图 7 select_sort

```
sha PASS!

sha: Total Count = 0x3570a

=====================================
Test end!
----PASS!!!
```

图 8 sha

```
Test begin!
stream copy test begin.

stream copy PASS!

stream copy: Count = 0x4d0f

=====================================
Test end!
----PASS!!!
```

图 9 stream_copy

```
string search PASS!

string search: Total Count = 0x25dab

==========================================
Test end!
----PASS!!!
```

图 10 stringsearch

5、性能测试上板结果

图 1 bitcount

图 2 bubble_sort

图 3 coremark

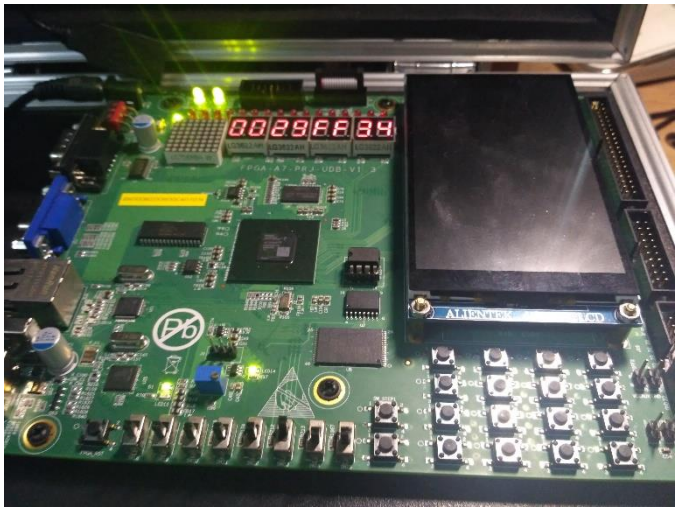


图 4 crc32

图 5 dhrystone

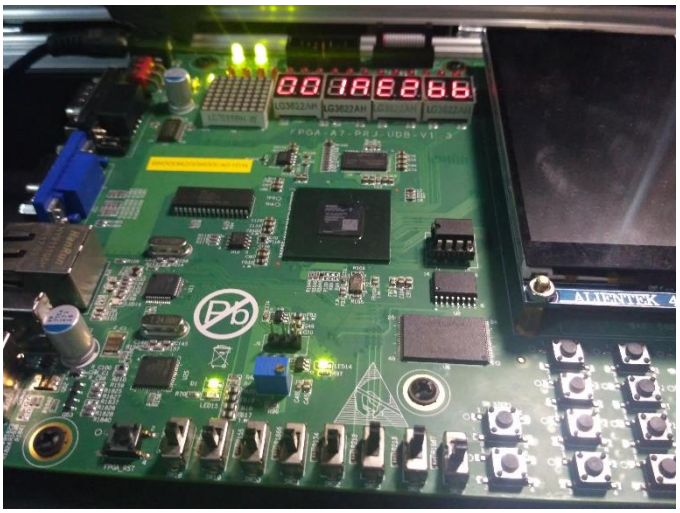


图 6 quick_sort

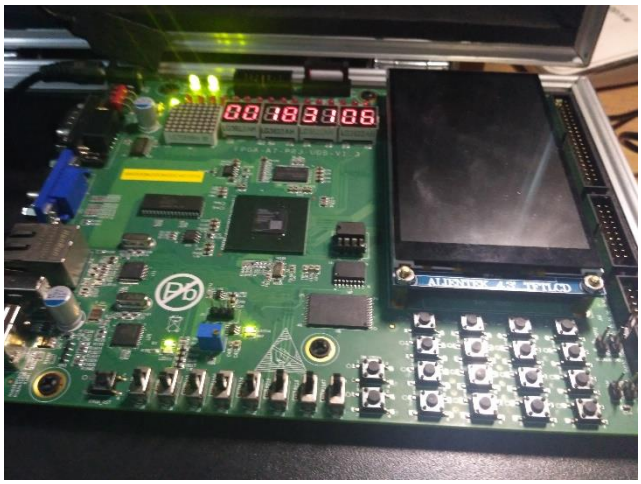


图 7 select_sort

图 8 sha

图 9 stream_copy

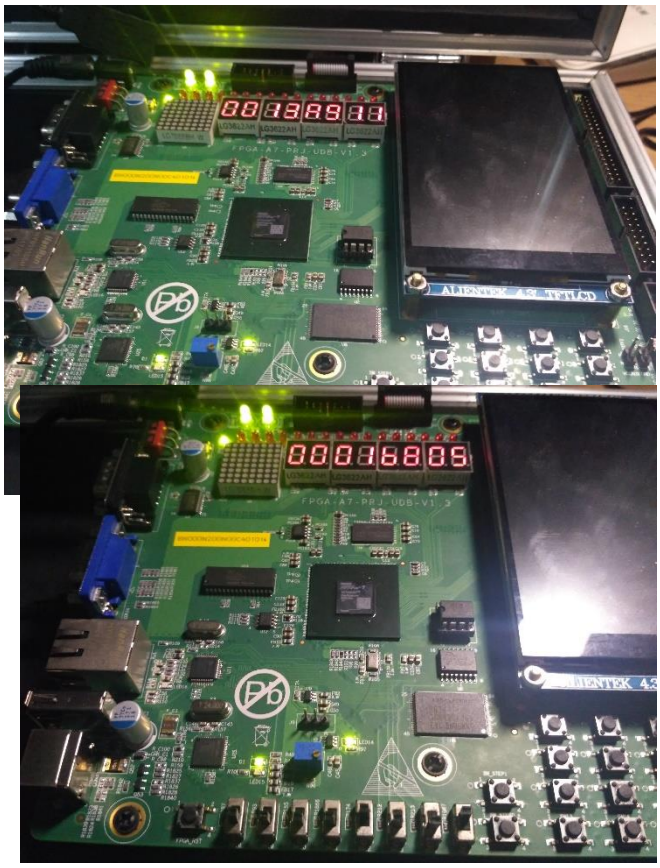


图 10 stringsearch

四、参考设计说明

为了充分利用资源，且发挥更好的性能，使用了 Xilinx IP：

乘法运算使用了 Xilinx IP Multiplier(12.0).(exe_stage)

除法运算使用了 Xilinx IP Divider Generator(5.1).(exe_stage)

cache 中使用了 Xilinx IP Distributed Memory Generator(8.0).(ICache、DCache)

五、参考文献

[1] 戴维 A.帕特森 (David A.Patterson),约翰 L.亨尼斯 (John L.Hennessy).计算机组成与设计(原书第 5 版)[M].机械工业出版社,2015.

[2] Stephen Brown,Zvonko Vranesic.数字逻辑与 Verilog 设计[M].清华大学出版社,2014.