

第五届全国大学生计算机系统能力培养大赛（龙芯杯）

SuboltMIPS CPU设计报告

王翰墨 王玉佳 吴奇 杨士欣

西北工业大学1队

目录

第一部分 概述

1.1 设计框架

1.2 系统概述

第二部分 CPU

2.1 流水线结构

2.1.1 总体架构

2.1.2 Pre-IF级

2.1.3 IF级

2.1.4 ID级

2.1.5 EX级

2.1.6 MEM1级

2.1.7 MEM2级

2.1.8 WB级

2.2 分支预测

2.2.1 设计思路

2.2.2 分支方向预测

2.2.3 分支目标地址预测

2.3 高速缓存设计

2.4 内存管理单元

2.5 AXI总线桥

2.6 指令集

2.7 CP0

2.8 中断和异常

2.8.1 中断

2.8.2 异常

2.8.3 例外处理

第三部分 系统与外设

3.1 概述

3.2 Pmon

3.3 Ucore

3.4 Linux

附录A 参考文献

第一部分 概述

本项目是在“龙芯杯”大赛方提供的 FPGA 实验平台上实现一个片上系统 (SOC)。其中，CPU 基于 MIPS 32 Rev 1 指令集架构，包含指令缓存和数据缓存；外设包含 SPI Flash,GPIO,MAC,DDR3,NAND,串口；能够启动并使用 PMON、uCore 操作系统。功能方面能够通过大赛方提供的功能测试、性能测试、系统测试；性能方面频率达 133MHz，每时钟周期指令数为龙芯 GS132 的 26.151 倍。

1.1 设计框架

我们的CPU采用顺序单发射七级流水线架构，实现了94条指令，12种例外，18个CP0寄存器，4项TLB，固定页大小为4KB。采用AXI3接口与外界交互。CPU 使用 2路组相联 8KB icache 及 2 路组相联 8KB dcache，采用写回，按写分配的设计。

1.2 系统概述

完整支持 PMON 和 Ucore 启动运行所需要的指令集，并且支持这两个系统的全部异常处理。可以完全的运行这两个系统，并执行应用程序。可以调用开发板上4×4按键，LED灯列，7段数码管等外设。

第二部分 CPU

2.1 流水线结构

2.1.1 总体架构

我们的CPU是一个7级流水线CPU，分别是PF（预取指）、IF（取指）、ID（译码）、EX（执行）、MEM1（访存第一阶段）、MEM2（访存第二阶段）、WB（写回阶段）。

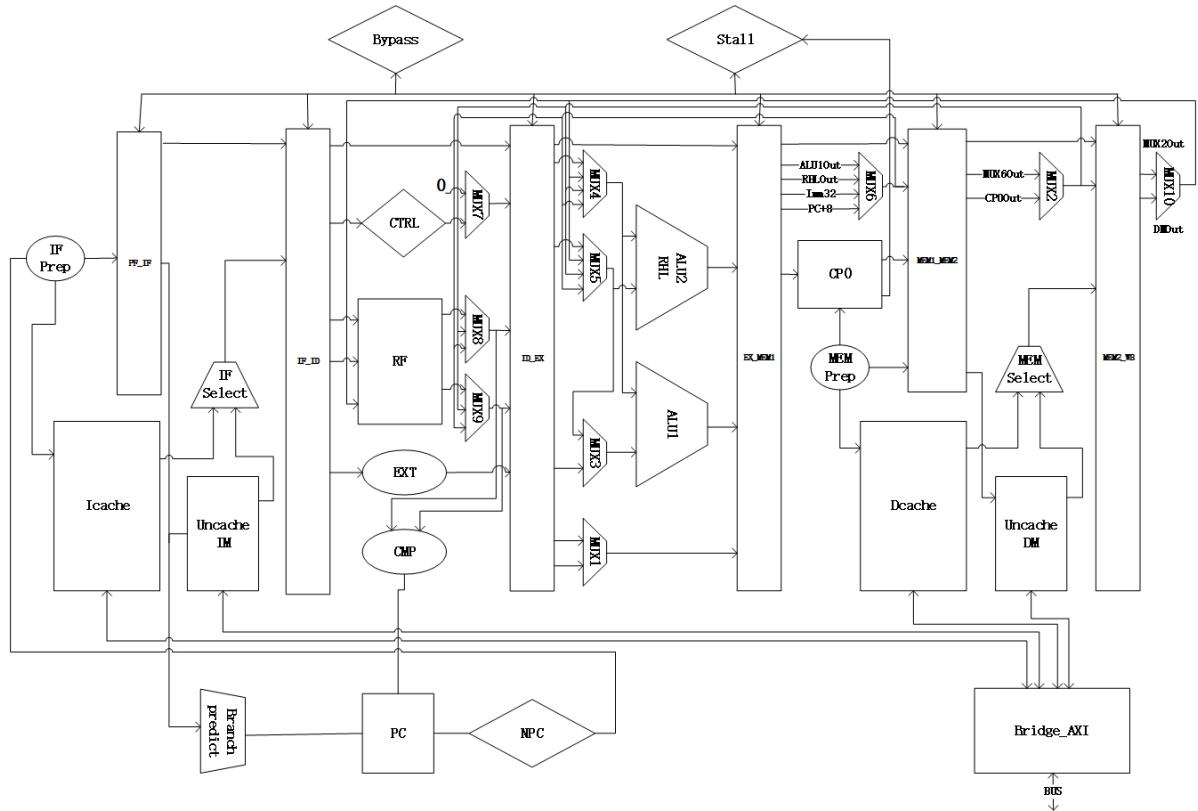


图2.1 流水线结构

2.1.2 Pre-IF级

计算出NPC的值，将NPC送入TLB完成地址转换，同时将虚地址送入Icache。

2.1.3 IF级

据Pre-IF送入的地址取出对应的指令（针对Icache）；将IF_PC送入分支预测模块，得到分支信号以及分支地址并送入PC。

2.1.4 ID级

完成对指令的译码，并生成各种控制信号往后传输；完成对分支指令的源操作数的比较；完成对16位立即数的扩展。将NPC生成需要的信息送给PC寄存器。

2.1.5 EX级

完成加减乘除移位等指令的运算。对分支模块中的PHT和BTB进行更新。

2.1.6 MEM1级

对CP0寄存器进行访存，报出例外；根据ALU算出的地址送入TLB，同时将虚地址送入Dcache。

2.1.7 MEM2级

根据MEM1送入的地址取出对应的数据（针对Dcache）；对访存地址进行判断，根据对应的结果决定访存是走Dcache模块还是Uncache_DM模块（访问外设）。

2.1.8 WB级

将要写回寄存器的数据写回寄存器堆。

2.2 分支预测

2.2.1 设计思路

在没有分支预测的情况下，分支指令在ID级才能确定分支方向和分支目标地址，此时IF级是一条分支延迟槽指令，而PF级就很有可能是一条要被刷新的指令。最初我们是预测分支总是不发生，故若分支发生，那么将会损失一个周期的代价。理想情况是当分支指令在IF级时就可以知道它的分支方向和目标地址，因此我们将分支预测单元设置在了IF级，包括分支方向预测和分支目标地址预测。

2.2.2 分支方向预测

采用两位饱和计数器和PHT结合的方法进行分支预测，用PC中的8位对PHT进行寻址，因此PHT中一共有256个饱和计数器，每一个PC都有对应的饱和计数器。对于饱和计数器，设置一个有限状态机，初始状态为**strongly not taken**，由状态机示意图可以看出，当某条分支指令连续两次发生跳转，那么下一次必定预测它跳转，反之一样。

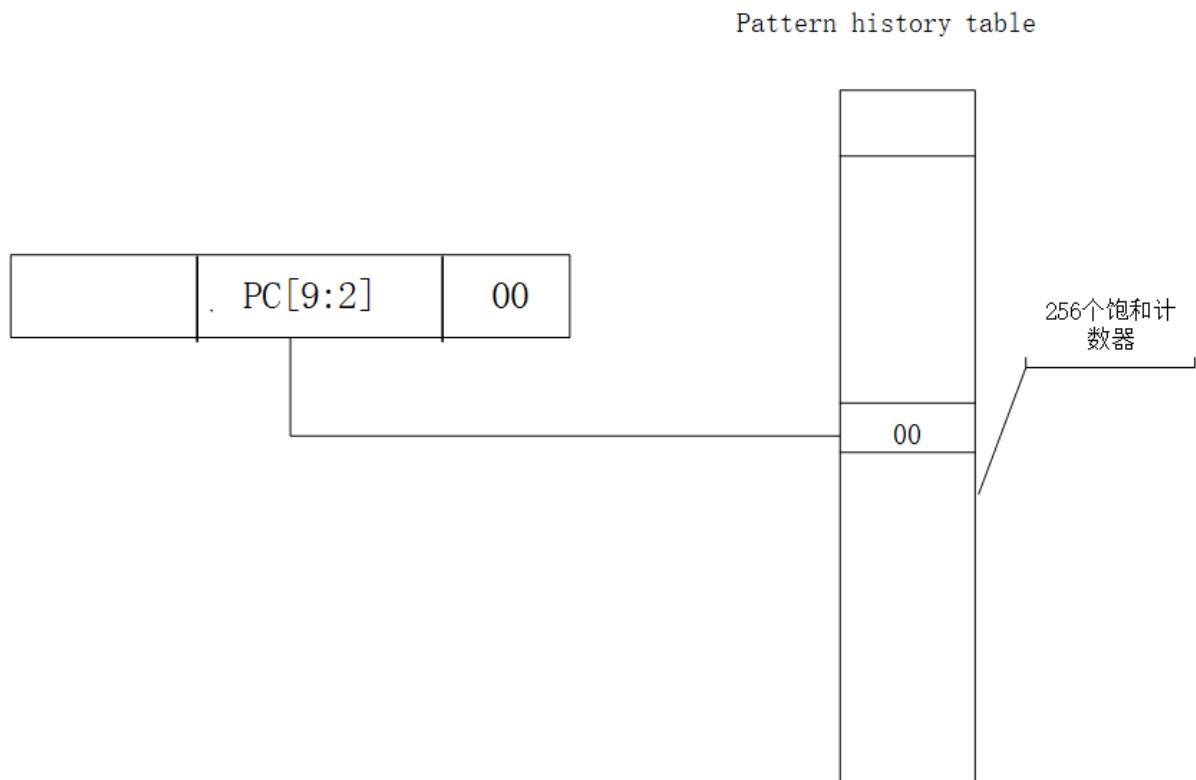


图2.2 PC寻址PHT中的饱和计数器

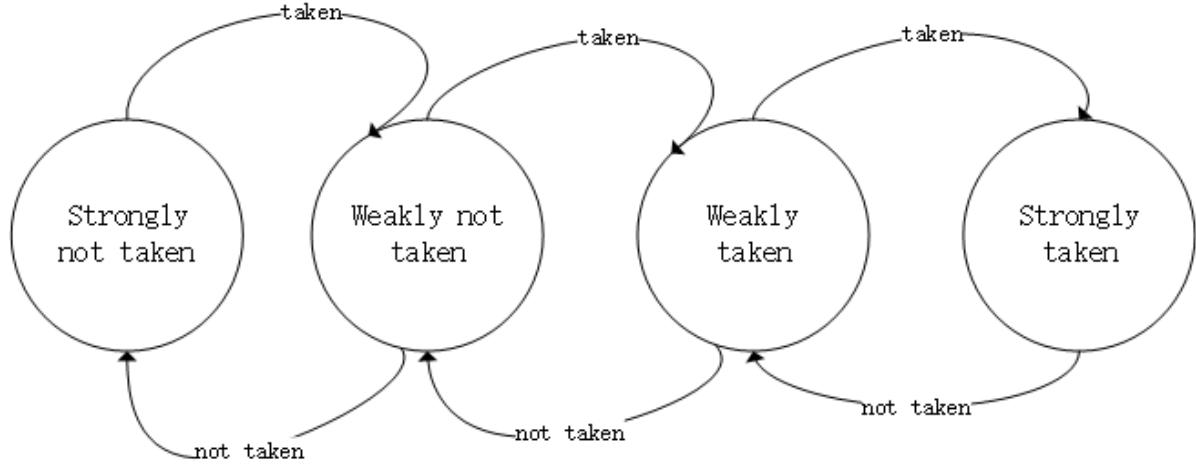


图2.3 方向预测单元状态机

我们尝试过加入全局预测位和局部预测位，加入全局预测位效果较差，原因可能是性能测试基准程序的分支指令的全局依赖性都较弱，加入局部预测位的效果较好，每一个性能测试基准程序的分支预测成功率都有所上升，但由于增加位数后对时序影响较大，故没有采用这种方法。

2.2.3 分支目标地址预测

采用BHT和RAS结合对目标地址进行预测。对于直接跳转指令和除了return指令外的间接跳转指令，将他们的跳转地址存放在BHT中，对于return指令 (JR \$31)，每当遇到一条call指令 (jal、Bgezal等)，就将call指令所在地址的PC+8存放到RAS中，并将栈指针加1，每当遇到一条return指令，就从RAS的栈顶取出地址，并将栈指针减1，从而完成对return指令的预测。

2.3 高速缓存设计

在流水线中我们设计了两个一级高速缓存，分别是指令CACHE和数据CACHE，大小均为8KB，采用二路组相联，组大小64，Cache Line大小为64B。这两个模块分别为ICACHE和DCACHE，其中ICACHE是DCACHE的缩减版，删去了写的相关功能。下面主要介绍DCACHE的完整功能。

DCACHE需要实现四个方向的数据交互：①从流水线到cache；②从cache到流水线；③从总线到cache；④从cache到总线。由于cache是使用LUTRAM和寄存器结合来实现的，以求得良好的时序，因而一条访存指令需要两拍才能完成（在不缺失的情况下），因而可以将第一拍作为发起请求，在MEM1级执行；第二拍作为返回结果/完成操作，在MEM2级执行。在一开始，cache可以处于IDLE空闲状态，表示不在工作，而当有连续的访存指令时，前一拍指令在MEM2级，对应LOOKUP状态，如果不缺失，那么下一拍继续就保持在LOOKUP，下一拍会将当前MEM1级的后序请求送过来，实现流起水来的效果。

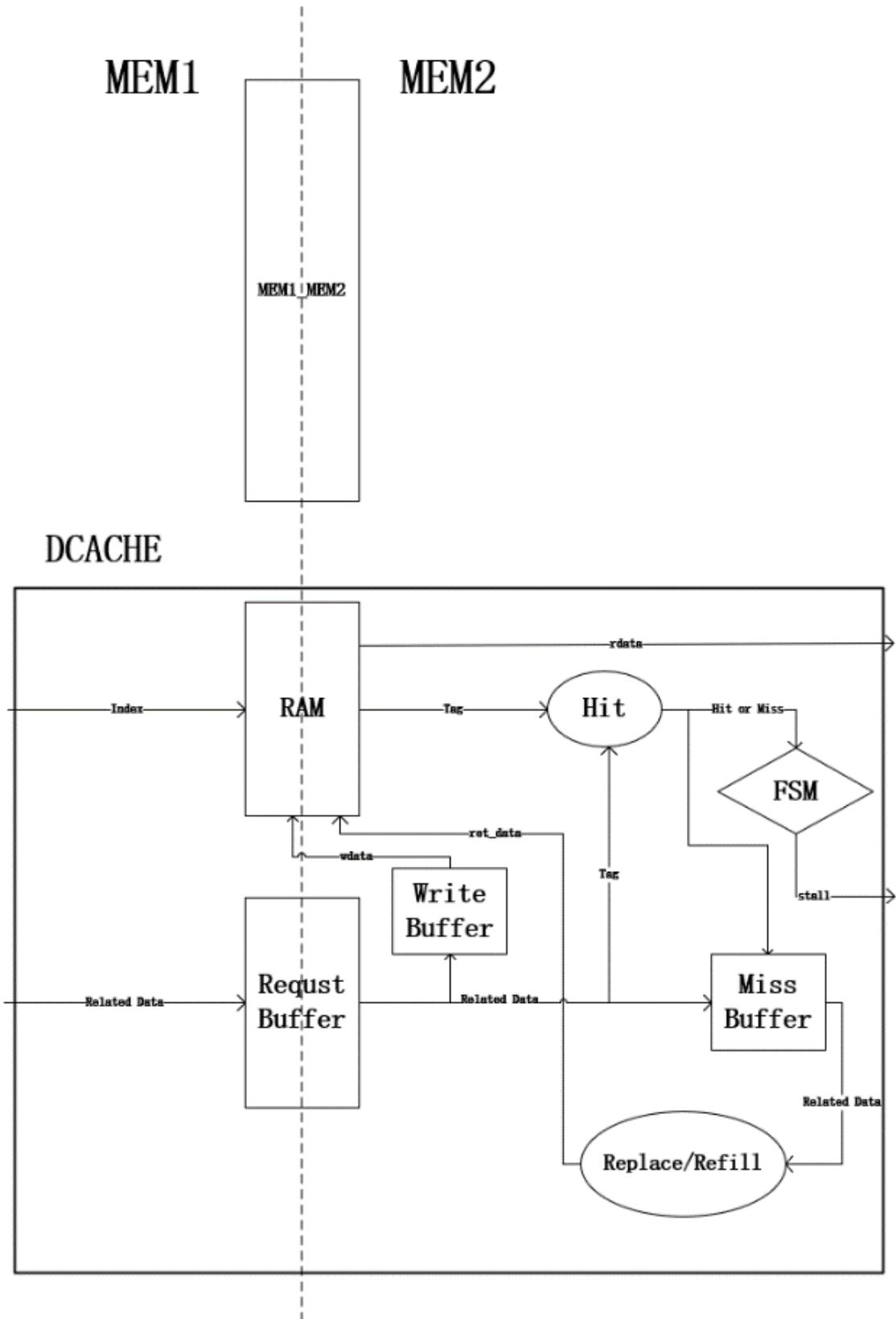


图2.4 DCache内部数据通路示意图

从上图中可以看出，Request Buffer起到了和MEM1_MEM2相同的作用，将两个流水级清晰地分隔开，方便进行访存第一步和第二步的相应操作，而Miss Buffer则是在cache未命中时才启用的，专门用于脏块替换和处理缺失的。如此，若发生cache缺失，则将对应的访存指令阻塞在MEM2级，等到走完MISS->REPLACE->REFILL后重新回到LOOKUP，此时cache应当命中，并进行相应的操作。下图是DCACHE内部的状态机。

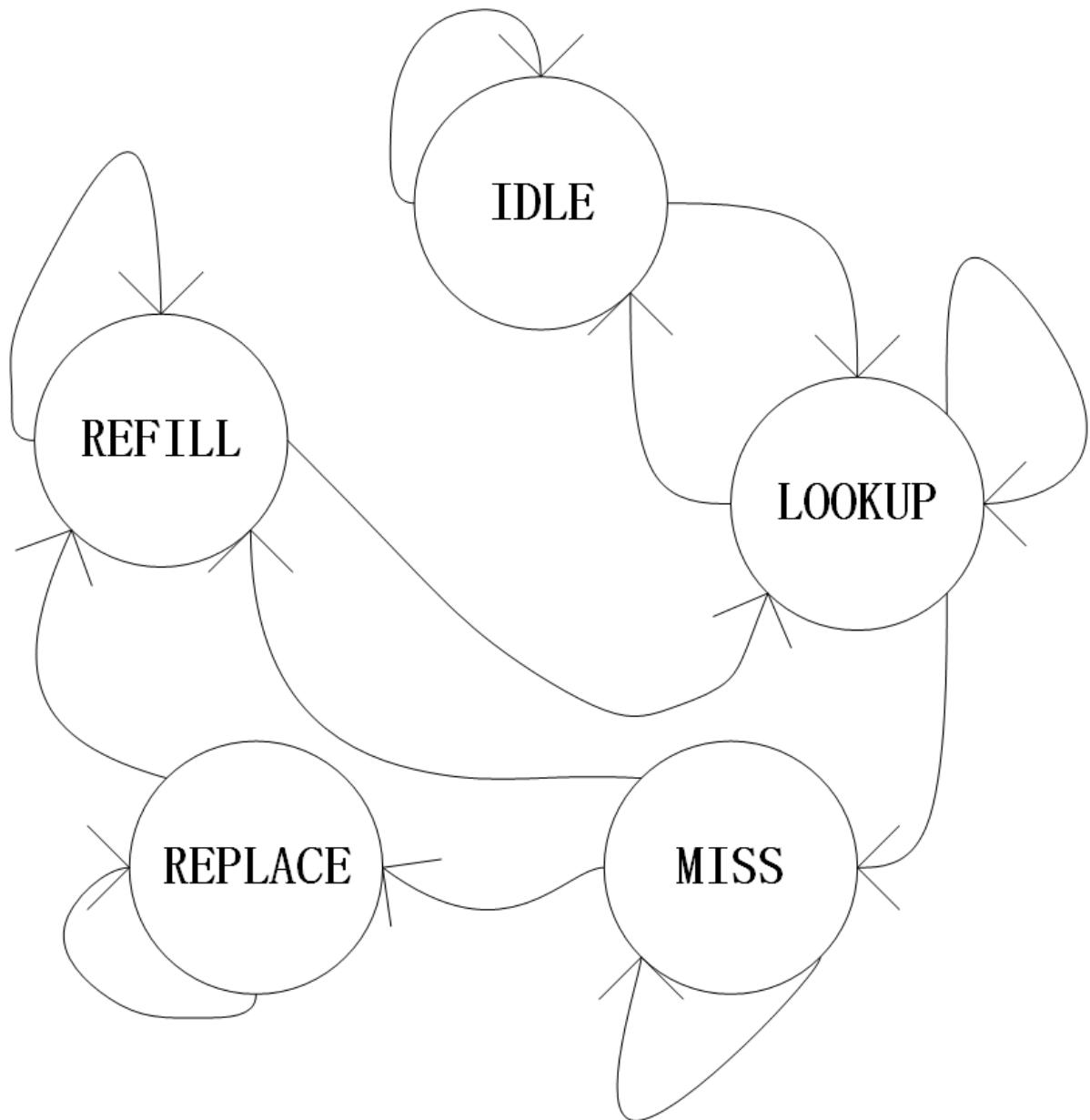


图2.5 Cache有限状态机示意图 为了支持操作系统，需要实现5条CACHE指令，以保证指令Cache和数据Cache的一致性。实现CACHE指令大部分复用正常使用Cache时的数据通路，主要进行三个操作：判别需要操作的Cache Line、进行脏块写回（如果需要）以及对Valid位进行清除（如果需要）。判别需要操作的Cache Line复用访存时的地址处理和TLB，脏块写回用Cache替换的通路，而对Valid位清除则设置单独的控制信号，配合状态机新增的状态实现。

2.4 内存管理单元

MIPS 为操作系统的内存管理提供了较为简单的支持，虚拟地址通过 MMU 转换为物理地址，具体的地址映射关系如下：

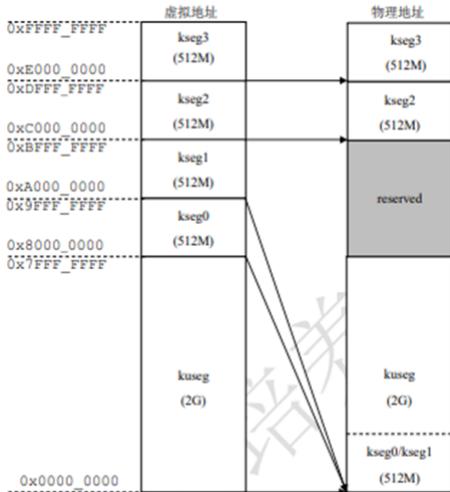


图 4-1 虚实地址映射关系图

内存段	映射关系	功能
kuseg	映射	存放用户程序、动态链接库等
kuseg0	未映射	存放内核代码、异常程序入口等
kuseg1	未映射	存放启动ROM、IO寄存器等
kuseg2, kuseg3	映射	存放动态分配的内核数据等

具体的地址转换由 TLB 来完成，TLB 可以认为是在 CPU 内部的地址转换表的高速缓存。具体的内容需要由操作系统来进行填充。如果在 TLB 中没有找到对应虚拟地址则会触发一个 TLB miss 异常，操作系统对该异常处理，并且将对应转换表填入 TLB 中的某一项来完成对地址的转换。为了降低 TLB 查找处理器整体频率的影响，我们一共只实现了 4 个 TLB 项，4 项 TLB 使用全相连的组织方式，同时实现了 tlbr, tlbp, tlbw, tlbrw 指令。

2.5 AXI总线桥

该模块负责将CPU封装为axi接口，负责处理CPU与外部设备的通信。我们使用四个状态机分别控制取指的cache/uncache，访存的cache/uncache请求，通过调用Xilinx的 axi_crossbar IP核来进行四路仲裁。同时该模块还负责处理访存cache段以块为单位写返回的问题。

2.6 指令集

我们实现了大赛要求的所有指令，在此基础上增加了一些额外的指令。下按功能分类列出。

- **逻辑运算指令** OR, AND, XOR, NOR, ORI, ANDI, XORI, LUI
- **移位指令** SLL, SRL, SRA, SLLV, SRLV, SRAV
- **数据移动指令** MFHI, MFLO, MTHI, MTLO, MOVN, MOVZ
- **算术运算指令** ADD, ADDU, SUB, SUBU, SLT, SLTU, ADDI, ADDIU, SLTI, SLTIU, MULT, MULTU, MUL, DIV, DIVU
- **跳转/分支指令** J, JAL, JR, JALR, BEQ, BNE, BGTZ, BLEZ, BGEZ, BGEZAL, BLTZ, BLTZAL
- **加载/存储指令** LB, LBU, LH, LHU, LW, SB, SH, SW, LWL, LWR, SWL, SWR, LL, SC

- **特权指令** MFC0, MTC0, ERET
- **自陷指令** BREAK, SYSCALL, TEQ, TNE, TGE, TGEU, TLT, TLTU, TEQI, TNEI, TGEI, TGEIU, TLTI, TLTIU, CACHE, TLBR, TLBWR, TLBWI, TLBP, SYNC, WAIT

2.7 CP0

寄存器名称	功能简述
Count	处理器时钟计数器
Compare	计时器中断控制
Status	处理器状态与控制寄存器
Cause	存放上一次例外原因
EPC	存放上一次发生例外的PC
Index	TLB的入口索引
Random	产生TLB的随机入口索引
EntryLo0	偶数虚拟页的入口地址的低位部分
EntryLo1	奇数虚拟页的入口地址的低位部分
Context	指向内存虚拟页表入口地址的指针
PageMask	控制TLB入口中可变页面的大小
Wired	控制固定的TLB入口的数目
BadVAddr	记录最近一次地址相关异常的地址
EntryHi	TLB入口地址的高位部分
PRId	处理器标志和版本
Config	配置寄存器，用来设置CPU的参数
TagLo	Cache中Tag接口的低位部分
Ebase	产生例外入口

表2.1 实现的CP0寄存器

2.8 中断和异常

2.8.1 中断

CPU 中支持 2 个软件中断 (SW0 SW1)、6 个硬件中断 (HW0 HW5) 和 1 个计时器中断。其中计时器中断复用 HW5 硬件中断。

2.8.2 异常

- 读数据或取指令的地址错例外
- 写数据的地址错例外
- TLB重填例外
- TLB无效例外
- TLB修改例外
- 算术溢出
- 系统调用
- 断点例外
- 自陷
- 保留指令例外
- 协处理器异常

2.8.3 例外处理

- 在PF级检测取指令的地址错例外、TLB相关例外；在ID（译码）级检测系统调用例外、断点例外、保留指令例外、协处理器异常；在EX级检测出trap；在MEM1级检测算术溢出例外、读数据和写数据的地址错例外、TLB相关例外。
- 将中断视作一种特殊的例外，检测到中断时将这种例外标记在MEM1级的指令上。
- 所有例外（包括中断）统一在MEM1级报出，实现精确例外

第三部分 系统与外设

3.1 概述

我们加入了功能完备的TLB，使其能够支持Pmon和Ucore的正常运行。并在Ucore的基础上编写了用户程序调用开发板上4×4按键、LED灯，七段数码管。

3.2 Pmon

成功启动了Pmon

```

PMON2000 MIPS Initializing. Standby...
ERRORPC=00000000 CONFIG=80000083
PRID=00004220
Init caches... do noting...
godson1 caches found
Init caches done, cfg = 80000083

Copy PMON to execute location...
start = 0x87900000
s0 = 0x38300000
_edata = 0x8794e010
_end = 0x8794f028
a7940000
copy text section done.
Copy PMON to execute location done.
sp=878fc000
Uncompressing Bios
OK,Booting Bios
FREQ
RTC time invalid, reset to epoch.
FREI
DONE
DEVI
ENVI
MAPV
in envinit
nvrarn=bfc00000
NVRAM is invalid!
NVRAM@bfc00000
STDV
80100000: memory between 82fff800-83000000 is already been allocated,heap is already above this point
SBDD
DINI
==gpio: gpio status 0
NETI
RTCL
====before configure
in configure
mainbus0 (root)
localbus0 at mainbus0
dmfe0 at localbus0: address 00:98:76:64:32:19
in if attach
loopdev0 at mainbus0out configure
====before init ps/2 kbd
devconfig done.
ifinit done.
domaininit done.
init_proc....
HSTI
SYMI
SBDE

* PMON2000 Professional *
Configuration [FCR,EL,NET]
Version: PMON2000 2.1 ([s1b] #367; 2021年08月11日星期二 21:48:14 CST).
Supported loaders [srec, elf, bin]
Supported filesystems [mtd, net, fs/yaffs2, fat, fs, disk, socket, tty, ram]
This software may be redistributed under the BSD copyright.
Copyright 2000-2002, opsycon AB, Sweden.
Copyright 2005, ICT CAS.
CPU GODSON1 @ 99.99 MHz / Bus @ 99.99 MHz
Memory size 128 MB (128 MB Low memory, 0 MB High memory) .
Primary Instruction cache size 8kb (64 line, 2 way)
Primary Data cache size 8kb (64 line, 2 way)

BEV1
BEV2
BEV3
BEVO
BEV in SR set to zero.

NAND DETE
NAND device: Manufacturer ID: 0xec, Chip ID: 0xf1 (Samsung NAND 128MiB 3,3V 8-bit)
NAND_ECC_NONE selected by board driver. This is not recommended !!
Scanning device for bad blocks
Bad eraseblock 515 at 0x04060000
Bad eraseblock 1023 at 0x07fe0000
NANDFlash info:
erasesize      131072 B
writesize      2048 B
oobsize        64 B
| PMON>

```

图3.1 Pmon启动示意图

3.3 Ucore

在顺利跑通Ucore的基础上，我们添加了一组系统调用用于用户程序对指定地址的读写操作，通过对GPIO地址的读写，调用了4×4按键，LED灯，七段数码管等显示。

使用Ucore控制外设的方法为，运行 showNumber 用户程序，然后按4×4键盘的下面两行可以控制7段数码管显示的数字，按键盘上面两行可以控制7段数码管上方LED灯列的亮灯。

```

PMON> g console=ttyS0,115200 rdinit=sbin/init
      zero      at      v0      v1      a0      a1      a2      a3
00000000 00000000 00000000 00000000 00000003 a7dfffd78 a7dfffd88 870af350
      t0      t1      t2      t3      t4      t5      t6      t7
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
      s0      s1      s2      s3      s4      s5      s6      s7
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
      t8      t9      k0      k1      gp      sp      s8      ra
00000000 00000000 00000000 00000000 00000000 a7dfffd58 00000000 8707abb8
++setup timer interrupts
Initrd: 0x8005e310 - 0x8025230f, size: 0x001f4000, magic: 0x2f8dbe2a
(THU.CST) os is loading ...
Special kernel symbols:
  entry 0x80000128 (phys)
  etext 0x80028400 (phys)
  edata 0x80252310 (phys)
  end   0x80255620 (phys)
Kernel executable memory footprint: 2217KB
memory management: buddy_pmm_manager
memory map:
[80000000, 82000000]

freemem start at: 80296000
free pages: 00001D6A
## 00000020
check_alloc_page() succeeded!
check_pkdir() succeeded!
check_boot_pkdir() succeeded!
----- BEGIN -----
----- END -----
check_slab() succeeded!
kmalloc_init() succeeded!
check_vma_struct() succeeded!
check_pgfault() succeeded!
check_vmm() succeeded.
sched class: RR_scheduler
ramdisk_init(): initrd found, magic: 0x2f8dbe2a, 0x00000fa0 secs
sfs: mount: 'simple file system' (129/371/500)
vfs: mount disk0.
kernel_execve: pid = 2, name = "sh".
user sh is running!!!
argc = 1
$ ls
@ is [directory] 2(hlinks) 8(blocks) 2048(bytes) : @'.'.
[d] 2(h)     8(b)    2048(s) .
[d] 2(h)     8(b)    2048(s) ..
[-] 1(h)    21(b)   85184(s) showNumber
[-] 1(h)    21(b)   85177(s) pwd
[-] 1(h)    21(b)   85323(s) ls
[-] 1(h)     1(b)    21(s) test.txt
[-] 1(h)    21(b)   85197(s) cat
[-] 1(h)    22(b)   89464(s) sh
lsdir: step 4
$ cat test.txt
hello world! Haha...
$ pwd
disk0:/
$ showNumber
$ 

```

图3.2 Ucore启动示意图

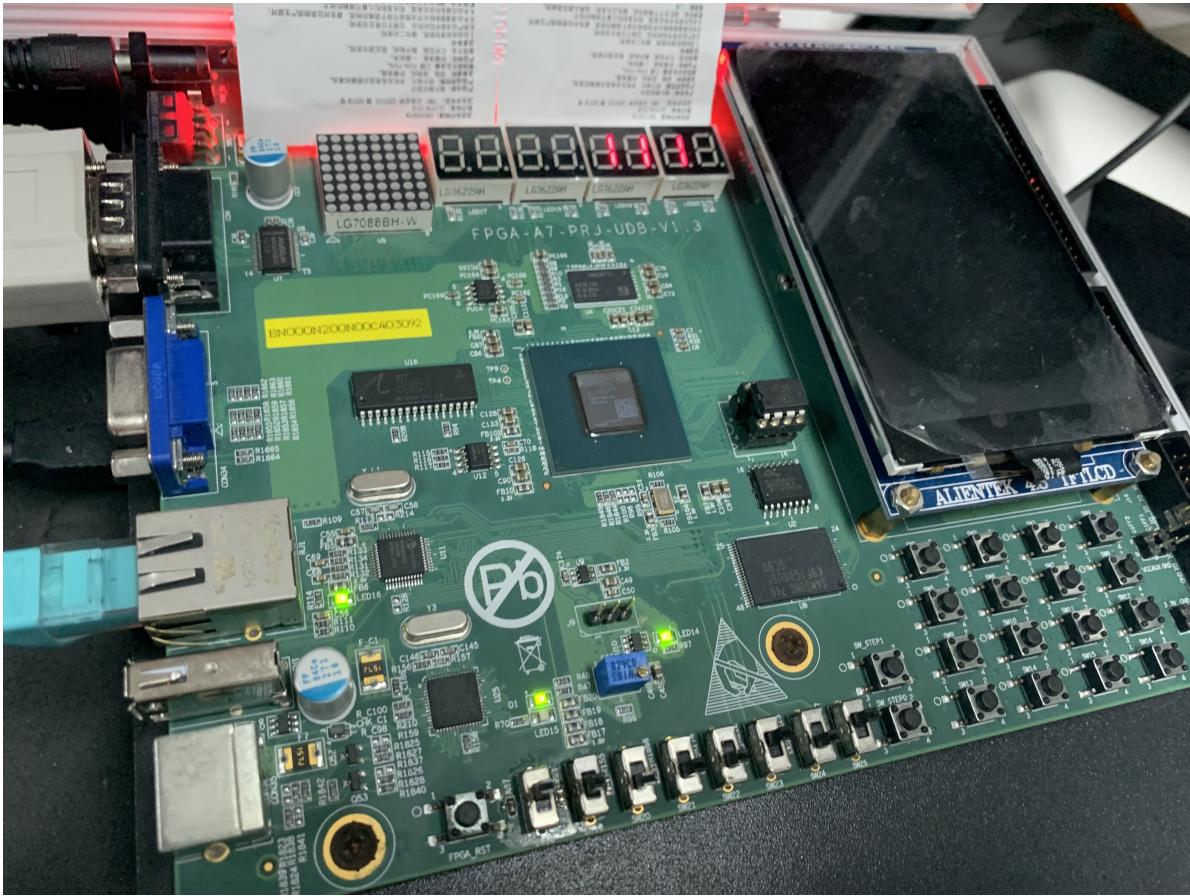


图3.3 通过Ucore控制7段数码管显示数字

3.4 Linux

由于时间原因，很遗憾没有完全启动Linux。

我们发现并解决了以下BUG，供以后参赛的队伍参考

- 新加指令的数据相关没有经过充分验证。例如'movn、movz'在ID级就要用到通用寄存器中的值，故需要阻塞。
- 阻塞往后续流水级插入空泡时，ALU1Op不可随便插入。例如我们插入空泡时默认ALU进行加法，可能会错误报出溢出例外。
- 中断不能标记在被刷掉的指令上。例如预测错误的PC被标记中断，导致eret返回后执行流错误。
- 例外报出优先级、例外入口选择的优先级。例如虽然报出了正确的例外，但是选错了例外入口。

附录A 参考文献

- David A. Patterson, John L. Hennessy. 计算机组装与设计：硬件/软件接口（第5版）. 王党辉等译. 机械工业出版社
- 姚永斌. 超标量处理器设计. 清华大学出版社
- 汪文祥, 邢金璋. CPU设计实战. 机械工业出版社

