**电 子 科 技 大 学**

UNIVERSITY OF ELECTRONIC SCIENCE AND TECHNOLOGY OF CHINA

**学士学位论文**

**BACHELOR THESIS**



论文题目

专 业

学 号

作者姓名

指导教师

**电 子 科 技 大 学**

**20 级本科毕业设计（论文）任务书**

**拟题单位** **审题人（签名）**

**题目及副标题**

**题目来源: 1.科研 2.生产 3.教学（含实验） 4.创新创业 （选择其中一种）**

**主要任务：**

**预期成果或目标：**

**预期成果形式： 1.硬件 2.硬件+软件 3.软件 4.理论研究 （选择其中一种）**

**指导教师签名:**

**起止时间： 年 月 日至 年 月 日**

**学生姓名** **专业**  **学号**

**指导单位**

**指导教师姓名、职称**

**设计地点**

**年 月 日**

**备注：**1.此任务书应由指导教师填写，签名处须由教师亲笔签名。

2.此任务书必须在学生毕业设计开始前下达给学生。

**电 子 科 技 大 学**

**20 级本科毕业设计（论文）开题报告表**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **学号：** | | | **姓名：** | **学院：** | **专业：** |
| **论文题目** |  | | | | |
| **题目来源： 1.科研　 2.生产　 3.教学（含实验） 4.创新创业**  **（在选项上打勾选择）** | | | | | |
| **成果形式： 1.硬件 2.硬件+软件 3.软件 4.理论研究**  **（在选项上打勾选择）** | | | | | |
| **学位论文**  **研究内容** | | **1500-2000字，主要从以下五个方面进行阐述（参考毕业论文正文格式：宋体小4号，英文字体为Times New Rome，行间距固定20磅，可另加页）：**  **1．研究现状及发展态势**  **2．选题依据及意义**  **3．课题研究内容**  **4. 拟解决的关键问题和最终目标，以及拟采取的主要理论、技术路线和实施方案等**  **5．论文特色或创新点** | | | |
| **导师审查**  **意见** | | **签名：**  **日期： 年 月 日** | | | |

**电 子 科 技 大 学**

**20 级本科毕业设计（论文）进度计划表**

**学院名称： 填表日期： 年 月 日**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **学生姓名** |  | | **论文题目 (含副标题)** |  | | | |
| **学 号** |  | |
| **周 次** | | **主要工作计划**  **（内容）** | | | **完成情况** | **指导教师签字** | **备 注** |
|  | |  | | |  |  |  |
|  | |  | | |  |  |  |
|  | |  | | |  |  |  |
|  | |  | | |  |  |  |
|  | |  | | |  |  |  |
|  | |  | | |  |  |  |
|  | |  | | |  |  |  |
|  | |  | | |  |  |  |
|  | |  | | |  |  |  |
|  | |  | | |  |  |  |
|  | |  | | |  |  |  |
|  | |  | | |  |  |  |
|  | |  | | |  |  |  |
|  | |  | | |  |  |  |

**说明:** 1.此表由指导教师填写，并与毕业设计任务书同时下达给学生；

2.该表作为专家组或指导教师对学生毕业设计进度检查的参考依据。

**电 子 科 技 大 学**

**20 级本科毕业设计（论文）初期检查表**

**学院名称： 填表日期： 年 月 日**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **以下内容由学生本人填写** | | | | | | | | | | | | | | | |
| **学生姓名** |  | | | **题目名称** |  | | | | | | | | | | |
| **学号** |  | | | **题目性质** | **科研 生产 教学（含实验） 创新创业** | | | | | | | | | | |
| **指导教师** |  | | | **工作地点** | **校内：** | | | | | | **校外：** | | | | |
| **设计时间** | **20 年 月 日至 20 年 月 日** | | | | | | | | | | | | | | |
| **以下内容由检查教师填写** | | | | | | | | | | | | | | | |
| **对课题的基本评价**（在备选项后面划勾） | | **课题工作量** | | | | **饱满** | |  | **适中** | | |  | **不够** | |  |
| **课题难度** | | | | **大** | |  | **适中** | | |  | **不够** | |  |
| **涉及知识点** | | | | **丰富** | |  | **比较丰富** | | |  | **较少** | |  |
| **课题价值** | | | | **很有价值** | |  | **价值一般** | | |  | **价值不大** | |  |
| **检查评语**  （学生毕业设计的准备工作及对设计任务的认识） | | **检查教师签名：** | | | | | | | | | | | | | |
| **当期完成情况成绩** | **优 秀** | | **良 好** | | | | **中 等** | | | **合 格** | | | | **不合格** | |
|  | |  | | | |  | | |  | | | |  | |

**说明：**1、本表内容应如实填写；

2、本表应妥善保管，以便装订在毕业论文中；

3、学院教务科对检查情况分类汇总后，报送教务处实践教学科备案。

**电 子 科 技 大 学**

**20 级本科毕业设计（论文）中期检查表**

**学院名称： 填表日期： 年 月 日**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **以下内容由学生本人填写** | | | | | | | | | | |
| **学生姓名** | |  | | | **题目名称** |  | | | | |
| **学号** | |  | | | **题目性质** | **科研 生产 教学（含实验） 创新创业** | | | | |
| **指导教师** | |  | | | **工作地点** | **校内：** | | | **校外：** | |
| **设计时间** | | **20 年 月 日至 20 年 月 日** | | | | | | | | |
| **以下内容由检查教师填写** | | | | | | | | | | |
| **课题核心** |  | | | | | | | | | |
| **课题进展情况** |  | | | | | | | | | |
| **存在困难** |  | | | | | | | | | |
| **解决办法**  **或**  **建议** | **检查教师签名：** | | | | | | | | | |
| **当期完成情况成绩** | | | **优 秀** | **良 好** | | | **中 等** | **合 格** | | **不合格** |
|  |  | | |  |  | |  |

**说明：**1、本表内容应如实填写；

2、本表应妥善保管，以便装订在毕业论文中；

3、学院教务科对检查情况分类汇总后，报送教务处实践教学科备案。

**电 子 科 技 大 学**

**20 级本科毕业设计（论文）答辩提问记录表**

**学生姓名** **学号**

**学 院** **专业**

**回答问题情况记录：**

**答辩组提问一：**

**学生回答情况(请打勾)： 1.好（ ） 2.较好（ ） 3.一般（ ） 4.差（ ）**

**答辩组提问二：**

**学生回答情况(请打勾)： 1.好（ ） 2.较好（ ） 3.一般（ ） 4.差（ ）**

**答辩组提问三：**

**学生回答情况(请打勾)： 1.好（ ） 2.较好（ ） 3.一般（ ） 4.差（ ）**

**答辩组提问四：**

**学生回答情况(请打勾)： 1.好（ ） 2.较好（ ） 3.一般（ ） 4.差（ ）**

**答辩组提问五：**

**学生回答情况(请打勾)： 1.好（ ） 2.较好（ ） 3.一般（ ） 4.差（ ）**

**答辩组组长签名：**\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**年 月 日**

**电 子 科 技 大 学**

**20 级本科毕业设计（论文）成绩考核表**

**学生姓名**   **专业**  **学号**

**题目全称**

**指导单位**

**指导教师**

**一、指导教师评语**

**1．课题工作量、难度及软硬件等方面能力锻炼**

**2．学生工作态度、进度执行及毕业设计任务完成等情况**

**3．论文中英文摘要、目录、正文、参考文献撰写及外文资料翻译情况**

**指导教师签名：\_\_\_\_\_\_\_\_\_\_\_\_\_**

**年 月 日**

**二、论文评阅教师评语：**

**1．论文中英文摘要、目录、正文、参考文献及译文等内容的文法及逻辑思路**

**2．论文内容所反映出的学生本人的工作量、难度及任务完成等情况**

**3．对论文全文的总体评价（注明是否达到答辩要求）**

**评阅教师签名：\_\_\_\_\_\_\_\_\_\_\_\_**

**年 月 日**

**三、毕业答辩专家组评语：**

**1．答辩报告的逻辑思路、语言表达、学生本人的工作量、难度及任务完成等情况**

**2．学生回答问题时所反映的逻辑思维、基本知识、基本技能和知识面等情况**

**答辩专家签名：\_\_\_\_\_\_\_\_\_\_\_\_\_**

**年 月 日**

**四、毕业设计（论文）成果形式认定：**

**1.硬件 2.硬件+软件 3.软件 4.理论研究 （请选择其中一种）**

**学院授权专家签名：\_\_\_\_\_\_\_\_\_\_\_\_\_**

**年 月 日**

**五、毕业设计（论文）成绩（总分100分）：**

|  |  |  |  |
| --- | --- | --- | --- |
| **设计过程** | **毕业论文** | **毕业答辩** | **总分** |
|  |  |  |  |

**年 月 日**

**注：成绩考核表要求如实填写，填写内容必须由教师亲笔手写。**

摘 要

虚拟存储技术和Cache是计算机存储系统的重要组成部分，它对操作系统的实现提供了硬件上的支持，近年来随着云计算技术的广泛应用，虚拟存储技术的研究日趋活跃。本文以虚拟存储系统的实现为课题，重点研究了虚拟存储器控制芯片以及Cache的Verilog实现，并实现了以组成完整计算机系统的MIPS架构的CPU，主要研究内容分为三部分。

**关键词：**虚拟存储器，高速缓存，Cache，虚拟存储器控制芯片，Verilog

ABSTRACT

Memory virtualization and cache are two crucial components of computer storage system, it provides hardware support to the operating systems.

**Keywords:**

目 录

[摘 要 I](#_Toc1445470025)

[ABSTRACT II](#_Toc1294057574)

[目 录 III](#_Toc1831000752)

[第一章 绪 论 1](#_Toc146570136)

[1.1 研究工作的背景与意义 1](#_Toc634055027)

[1.2 时域积分方程方法的国内外研究历史与现状 1](#_Toc1942883562)

[1.3 本文的主要贡献与创新 1](#_Toc1301323606)

[1.4 本论文的结构安排 1](#_Toc141960830)

[第二章 虚拟存储系统基本结构 2](#_Toc1259084018)

[2.1 内存管理单元 2](#_Toc230227678)

[2.1.1 地址翻译 3](#_Toc920693850)

[2.1.2 翻译缓存 4](#_Toc889504913)

[2.1.3 进程间的内存隔离 5](#_Toc1942636477)

[2.1.4 多级页表 5](#_Toc424837313)

[2.1.5 MIPS的虚拟存储系统 6](#_Toc2038581941)

[2.2 高速缓存 10](#_Toc1262909367)

[2.2.1 映射方法 10](#_Toc1452206142)

[2.2.2 替换算法和写策略 12](#_Toc731036366)

[2.2.3 虚拟存储器中的高速缓存 13](#_Toc1524812194)

[2.3 本章小结 14](#_Toc249916380)

[第三章 虚拟存储控制芯片的实现 15](#_Toc1737635933)

[3.1 系统整体结构及开发环境 15](#_Toc760357458)

[3.1.1 Verilog HDL 15](#_Toc1988182067)

[3.1.3 模块层次结构 16](#_Toc1886037553)

[3.3 MMU模块的实现 17](#_Toc942175947)

[3.3.1 TLB模块的设计 21](#_Toc22944863)

[3.4 Cache模块的设计 24](#_Toc1888095575)

[第三章 时域积分方程数值方法研究 26](#_Toc1462947218)

[3.1 时域积分方程时间步进算法的阻抗元素精确计算 26](#_Toc1996401455)

[3.2 时域积分方程时间步进算法阻抗矩阵的存储 26](#_Toc1603538963)

[3.2.1 时域积分方程时间步进算法产生的阻抗矩阵的特征 26](#_Toc1297258942)

[3.2.2 数值算例与分析 26](#_Toc1294387833)

[3.3 时域积分方程时间步进算法矩阵方程的求解 27](#_Toc750112889)

[3.4 本章小结 27](#_Toc980776046)

[第四章 全文总结与展望 28](#_Toc1440957969)

[4.1 全文总结 28](#_Toc1384167916)

[4.2 后续工作展望 28](#_Toc776175960)

[致 谢 29](#_Toc594797928)

[参考文献 30](#_Toc1526128746)

[外文资料原文 31](#_Toc2035259978)

[外文资料译文 32](#_Toc825025606)

第一章 绪 论

1.1 研究工作的背景与意义

……

1.2 时域积分方程方法的国内外研究历史与现状

1.3 本文的主要贡献与创新

1.4 本论文的结构安排

第二章 虚拟存储系统基本结构

为了提高执行的效率以及满足操作系统和应用程序的需求，现代计算机采用了虚拟存储器技术。处理器通过向存储器发出访存信号以及地址来完成指令的读取和数据的读写。存储器发出的地址称为虚拟地址，它要先经过内存管理单元（MMU）翻译成物理（或实际）地址，然后再传递给各个层级的存储器完成访存的操作，最终得到访存的结果。这样不仅可以将各个进程的内存区域隔离开，还可以使得虚拟的内存空间大于实际的内存大小。

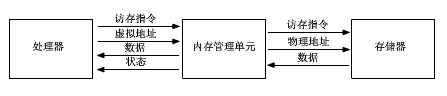


图2-1 虚拟存储器的系统结构

2.1 内存管理单元

从虚拟地址到实际地址的转化操作往往是使用内存管理单元来实现的。这样的地址翻译过程使得用户可以把包含主存储器和大容量存储器的装置感觉成单个大容量的空间。当内存管理单元收到地址翻译请求后就会根据翻译的结果作出不同的响应：输出实际地址或触发异常，使得CPU完成不同的操作。



图2-2 内存管理单元

在操作系统中往往会对不同进程所能访问的内存单元做不同的限制，如有些内存区域是只读的，有些区域只允许内核进程访问。为了实现这些需求，MMU同样实现了内存访问的权限管理，对非法的内存访问触发异常。

硬件上，MMU可以是一个单独的芯片，如68851，也可以与处理器集成在一起，使得处理器可以直接输出物理地址。

2.1.1 地址翻译

为了实现虚拟地址到实际地址的翻译，MMU一般使用分页存储管理方式，即将虚拟地址空间和实际空间分成若干个大小相等的块，称为“页（page）”，并以页为单位来完成地址的映射。这样，虚拟地址和物理地址就都分别分成了页号和页内地址两部分，其中虚拟地址的页号被称为虚拟页（virtual page number, VPN），物理地址的页号被称为物理页（physical page number, PPN）。并使用一个页表来记录虚拟页号相对应的物理页号。



图2-2 虚拟地址与物理地址的组成

通常，一个页表的表项包含虚拟页号、物理页号以及一些用于标记该项状态的服务位（Service bit）：

（1）有效位，用于标示是否可以用该项进行地址转换。在机器初始化的时候由于还没有分配页，所以应将所有的项都标记为无效。

（2）修改位，或脏位，用于指示该项对应的页是否有被修改过。对于使用了大容量存储器来扩大内存的操作系统，当一个被修改过的页被替换时就需要将该页的数据先写回到大容量存储器上。

（3）引用位。用于实现页替换的LRU算法。当页被访问时会被写为1，并且会被定期地写为0。

实际中页表往往是保存在具有缓存系统的存储器中，每个表项相对于页表起始地址的位置就是虚拟页号，这样使得给出一个虚拟地址立即就能知道其表项的地址。此外，每个表项还包含用于标明此项是否有效和用于控制内存访问的状态位。对于一个一般的MMU其地址翻译过程大概如下：

（1）根据虚拟页号找出与所给虚拟地址的页号相符项，检查有效位是否有效，如无效，则说明此虚拟页还没有与之相对应的物理页，所以触发缺页异常；

（2）根据匹配到的项中的服务位来检查此次访问是否合法，若非法，则触发异常；

（3）取出物理页号，将其与虚拟地址中的页内偏移连接得到转化后的物理地址。

必须指出，地址翻译过程对于不同架构其具体的过程是不同的，有些与以上的描述有所差异。

2.1.2 翻译缓存

为了提高地址转化的效率，MMU中保存了页表的一小段缓存，称为翻译后备缓冲区（Translation Lookaside Buffer, TLB）。对于有TLB的MMU其地址翻译过程应为先在TLB中查找，若找到则TLB命中，直接从TLB中取出表项做后续的处理，否则先从内存中的页表里取出相应的项填入TLB。TLB通常采用全相联的缓存，实际中可使用相联存储器（CAM）来实现。

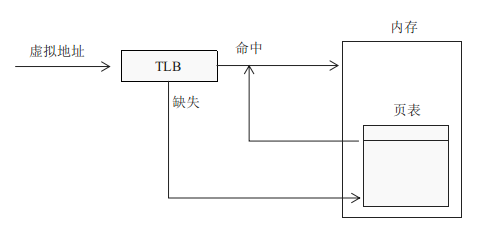


图2-3 翻译后备缓冲区

不同架构对于TLB的实现以及操作方法是有所差异的。在TLB中查找相匹配的项一般都是硬件完成，而TLB不命中时从内存中调出表项则不同：对于x86、PowerPC等架构这部分操作是由硬件来完成的，软件只需要维护位于内存中的页表以及处理页表项无效时的缺页中断，并且将页表的起始（物理）地址写入相应的寄存器即可。MIPS架构则是需要软件维护TLB，当TLB不命中时会触发TLB Miss中断，由软件完成页表项的加载。换句话说，硬件并不认为TLB是页表的缓存，相反，TLB被认为是整个“页表”。此时便不存在缺页中断了，当软件发现位于内存中的表项无效时直接调用缺页处理程序完成物理页框的分配以及将被替换的页写回大容量存储器。当软件在从内存中调入页表项的时候，如果页表的起始地址使用的也是虚地址此时就可能再次触发TLB Miss中断，而当此次中断处理完成后就会回到最初触发中断的地方，由于只更新了页表所在页的地址映射，这次访存还会触发TLB Miss异常。所以在最坏的情况下MIPS的一条访存指令会触发三次异常。但这样做的好处就是使得系统的编写更加灵活。由于加载表项到TLB中的过程是软件来完成的，所以操作系统实现的表项结构可以与TLB的表项有所不同。

当TLB不命中而TLB已满时就需要将其中已有的一个表项替换掉。常用的替换方式有随机替换、最近最少使用、先进先出等。

2.1.3 进程间的内存隔离

在操作系统中不同进程间的内存需要隔离开，即同一虚拟地址在不同的进程中会被翻译成不同的物理地址，不同进程的虚拟地址空间是不同的，以此来达到内存隔离的目的。实现的方法一般有两种：

（1）每个进程有自己单独的页表（对于二级或多级页表，这里应该是页目录），在进程切换时将TLB的所有表项标记为无效，并修改保存页表起始地址的寄存器；

（2）在页表的每个表项中引入进程号用于标记该表项所属于的进程，并通过一个寄存器记录当前进程的进程号。在进行地址翻译时增加表项的进程号与当前进程匹配的要求。

x86使用的是第一种方式，它的页表项中没有用于区分进程的字段，在进程切换时处于TLB中的表项均属于之前的进程，必须先无效化，以便从内存中读取出新进程的页表。TLB中的所有表项总是属于同一个进程。

MIPS采用的是第二种方式，其表项中有一个8位的ASID字段用于标明进程，同时当前进程的进程号位于Context寄存器中。

2.1.4 多级页表

单级页表占用的空间可能会很大，而在一般的应用中很少会将整个页表填满。为了节约页表所占的空间，可以采用多级页表。对于二级页表而言，第一级页表也叫页目录，用于保存第二级页表的起始地址，第二级页表则为页表，保存了相应的物理地址。这样虚拟地址被分成三个部分：页目录号、虚拟页号、页内偏移。在翻译地址的时候先根据页目录号查出页表的起始地址，然后在页表中找到物理页号。类似地，n级页表的前n - 1级保存的是下一级页表的起始地址，最后一级则是物理地址。



图2-4 二级分页的翻译过程

2.1.5 MIPS的虚拟存储系统

本课题是要实现MIPS32中的MMU，这里重点讨论一下MIPS的内存管理单元具体的工作原理。

2.1.5.1 虚拟地址空间与权限管理

MIPS32把4GB的虚拟内存空间划分为了几个段，如图2-5

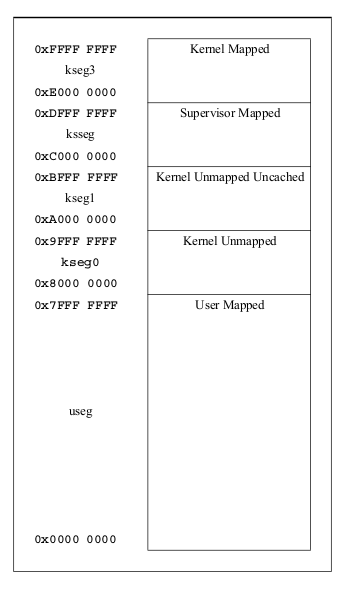


图2-5 虚拟地址空间的划分

对这些数据段[[1]](#footnote-0)的访问权限取决于处理器当前的运行模式。MIPS处理器存在用户模式（User Mode）、管理模式（Supervisor Mode）、内核模式（Kernel Mode）这三种不同的模式，其特权级别是递增的，高特权级可以访问相同或低特权级的段，非法的访问会触发地址错误异常（Address Error Exception）。

不同的数据段的内存访问方式也是不同的。useg、ksseg、kseg3的地址是经过虚拟地址映射的，对这三个段的访问的地址会经过MMU翻译。kseg0段是直接映射到物理地址0x000000-0x1FFFFFFF，在做地址翻译的时候只需将虚拟地址的最高位换成0即得到物理地址。这部分空间用于TLB初始化前的初始化操作，异常处理程序的入口地址也在这段中。

kseg1是用于IO操作的段，这部分地址所指向的通常不是存储器，而是外设，向这部分区域的读写操作就是在控制外设。由于在这段中向一个地址的读和写可能不是普通的内存读写，因此该段不会经过Cache缓存（Uncached）。

2.1.5.2 地址翻译过程

MIPS的地址翻译同样是基于TLB的，其TLB的条目包含EntryHi、EntryLo0、EntryLo1、PageMask四个部分，在MIPS32/64中，这四个部分都是32/64位的。其中虚拟页号位于EntryHi，一个TLB条目可同时记录两条地址映射信息于EntryLo0和EntryLo1中，分别对应于页内地址的最高位为0或1的两种情况。四个部分的具体字段及意义如下：

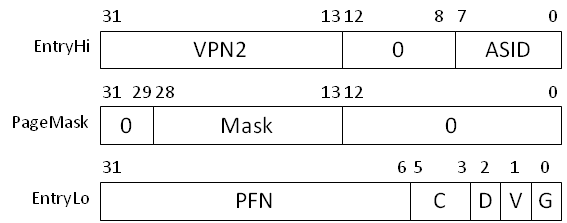


图2-5 TLB条目的组成

VPN2：虚拟页号。其中“2”表示一个条目中一个虚拟页号对应两个物理页号；

ASID：该条目属于的进程号。在地址翻译时必须和当前进程号相等才匹配成功；

Mask：虚拟页号掩码，用于实现可变大小的页；

FPN：物理页号；

C：用于控制Cache对该页的缓存方式，等于不同的值时的作用不一样。MIPS规定这个字段等于2和3时分别代表不允许和允许将该页放入Cache，而其他值不同的厂商可以定义不同的意义；

D：脏位。等于1时表示该页被修改过；

V：有效位；

G：全局位，为1表示该项是全局的，所有进程都可以用这项做地址翻译。如果一个表项的两个G 位都是1那么在查找TLB做地址翻译的时候就不需要匹配ASID。

一个表项总大小是16B，如果按所有页都是1KB来计算，页表总共包含0.5M条记录，总大小就是8MB，并且由于不同进程的表项可用ASID来区分，整个操作系统就只需要一张页表即可，因此页表大小在大多数情况下都是可接受的。

此处条目中的物理页号长度为26位，加上12位的页内地址，物理地址最长可以达到39位，也就是可支持0.5TB大小的内存。然而大多数情况下内存是小于这个值的，这时可直接将物理页号的高位置为零并在地址翻译的时候忽略。

此外，TLB表项中还包含可选字段RI和XI，当为1时就禁止对该页的读取和执行操作，否则会触发TLBRI和TLBXI异常。

TLB表项中的虚拟页号与实际的虚拟页号相比少了最后一位，这是因为TLB的一个表项有两个物理地址的记录，选择哪个记录由虚拟地址的最后一位决定，也就是说两个记录分别对应奇数页号与偶数页号。如对于1KB大小的页，其虚拟地址的12到31位是虚拟页号，0到11是块内地址，12位则是用于选择两个物理地址其中一个的。

页的大小可以通过Mask字段来控制，虚拟地址的虚拟页在与TLB的每个表项中的虚拟页做匹配前两者都要先与Mask的反码取与操作之后再比较，这就实现了可变长度的虚拟页号和页内地址。MIPS中，页的大小可以取4KB、16KB、64KB、256KB、1MB、4MB、16MB、64MB、256MB，相应的虚拟页号长度分别是20、18、16、14、12、10、8、6、4。

MIPS中与MMU有关的异常有：

（1）TLBS与TLBL：这两个异常都是当TLB中不存在与所给虚拟地址相匹配的项，或虽然存在匹配的项，但有效位为0时触发，具体触发哪个取决于当前的访存操作是读还是写。前一个条件触发的异常又称为TLB Miss（TLB Refill），后一个又称为TLB Invalid，具体触发的是哪个异常只能通过TLB查询指令来获取，一般情况下对这两个异常的处理方式都类似；

（2）TLB Modified：第一次修改页的时候触发。此时D位会被写为1,并且之后的写操作都不会再触发异常。这个异常用于向操作系统发出这个页已经被修改的警告；

（3）ADEL、ADES：这两个为地址错误，都在出现低特权级模式下访问高特权的内存段时触发，并根据访存操作是读取或写入分别选择触发前者还是后者；

（4）TLBRI和TLBXI：分别当RI或XI为1并且此次访存操作是读数据或读指令时触发。



图2-6 TLB表项的匹配过程

2.1.5.3 MMU操作

MIPS的MMU是与CPU集成在一起的，和MMU相关的寄存器都在协处理器0（Coprocessor 0）中，可通过两个协处理器的寄存器读写指令来对寄存器进行操作。其格式如下：

|  |
| --- |
| MTC0 $reg, $cp0reg  MFC0 $reg, $cp0reg |

协处理器0是一个逻辑处理器，并不是一个实体，可以通过它来完成包含内存管理控制的一些操作。协处理器0中与MMU有关的寄存器有：

Context：上下文寄存器，用于记录TLB异常所涉及到的TLB条目在内存中的地址，其结构如下图

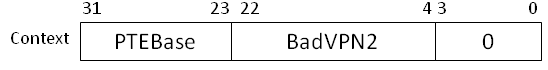


图2-7 Context寄存器

其中PTEBase是页表的起始地址，BadVPN2是触发异常的虚拟页号，当和TLB有关的异常被触发后这个字段就会被写为触发异常的虚拟页号。最低4位总是0,正好对应一个页表条目的大小16B。操作系统可通过修改PTEBase来将页表定位在其他以8MB为单位的地址上。

EntryHi：对应于TLB条目的EntryHi，当执行TLB写入指令时会将此寄存器的值写入相应条目中的EntryHi部分，执行TLB读取时相应条目中的EntryHi部分会被写到此寄存器中。此外，在其他情况下该寄存器的低8位用于记录当前进程的进程号。

EntryLo0、EntryLo1、PageMask：作用与EntryHi寄存器相同，只不过没有其他用处。

Index：TLB条目索引号，在执行TLB写入时用于指明写入的条目的索引号，在执行TLB查找的时候会将匹配的条目的索引写入该寄存器。

Random：随机数寄存器，是只读的。用于在执行随机写入时指明索引。

Wired：若大于0，则TLB条目索引为0到Wired-1之间的条目不能被随机写入。

MIPS提供了用于操作TLB的指令，可完成TLB条目的读取、写入以及查找操作：

TLBWI：按索引写入TLB条目，索引号由Index寄存器给出；

TLBWR：将数据写入一个随机的TLB条目，此时的索引由Random寄存器给出；

TLBR：读取索引为Index的TLB条目；

TLBP：找出与EntryHi寄存器中的虚拟地址和进程号相匹配的TLB条目，并将相匹配项的索引写入Index寄存器。若没有匹配的条目，写入的值就是最大的的TLB条目索引加一；

2.2高速缓存

高速缓存（Cache）是位于处理器和存储器之间的缓冲存储设备，其访存的速度是大于主存的。当处理器发出访存指令和地址时先会在Cache中查找该地址是否存在于缓存中，如果存在就立即访问并返回结果，否则再去访问主存，把主存中相应的数据块调入到Cache中。这个访问Cache的过程完全由硬件来完成，一般情况下对于用户（甚至处理器）来说是不可见的，当然也有例外，如MIPS就提供了操作Cache的指令。

Cache的结构一般是分为多个大小相等的数据块，内存中的块会通过一定的方式映射到Cache中的块，这个块称为行（line）。由于Cache不可能把整个内存都缓存下来，因此这个映射只能是内射（injection）。每个块都有相应的标记位，分别用于标明这个块是否有效、是否被修改过（脏位），以及用于匹配的标签位，因为一个Cache块可对应多个内存中的块。

Cache的依据是局部性原理，即相邻的几次访存操作的地址往往很接近。在这种情况下如果把内存中的一块数据加载到Cache中那么这几次的访存就都只需要访问缓存而不用访问主存，从而提高了程序执行的效率。

2.2.1 映射方法

常用的映射方法有全相联、直接映射和组相联。

2.2.1.1 全相联

全相联是指一个数据块可以放在Cache中的任何块上，并通过标签来区别不同的内存块。整个Cache是一个相联存储器（CAM）。对于全相联，内存地址被分成两部分：标签和块内偏移量，其中偏移量的长度由块的大小决定。在访存操作中，将所给地址的标签部分与Cache中所有的块的标签作比较，如果发现有匹配的块即说明Cache命中，此时只需要对该块做相应的操作。



图2-5 全相联Cache

由于数据块可以被放在Cache中的任何地方，因此全相联是自由度最高的Cache，随之带来的问题就是使用的用于判断标签是否匹配的比较器较多，会占用很多电路空间。

2.2.1.2 直接映射

直接映射的Cache将地址分为三部分：标签、索引、块内地址，其对应成数据块直接就是索引所指的块，也就是说内存中的一个块对应一个固定的Cache块，Cache块的个数即为行数。



图2-6 直接映射

直接映射的Cache电路设计简单，资源少，但最大的问题就是一个内存行只能对应一个Cache行，如果出现程序交替访问两个标签不同，而索引相同的块的情况Cache将一直不命中，并且会一直从内存中调取块，造成极大的性能损耗。

2.2.1.3 组相联

组相联结合了全相联与直接映射，其地址结构与直接映射相同，但Cache中每个索引有多个块，这些具有相同索引的行组成的一个行组被称为一排（Row），每排具有的块数称为相联性（Associativity）。也就是说，在内存块到Cache中的排时采用直接映射，在一排内采用的是全相联映射。



图2-7 组相联

2.2.2 替换算法和写策略

对Cache的读操作没有对内存产生影响，因此不用考虑，而写操作则可能有如下的情况：

（1）若Cache不命中，就先更新主存。此时可以将块加载到Cache中（写分配），也可以不加载。

（2）若命中，则可以采取以下的方式之一：

写直达（write through）：同时将数据写入缓存和主存中。

回写（write back）：只写缓存，并且当要替换该块时并且块被修改过才将该块写回到内存中。这种方法有效地克服了前一种方法的缺陷，如程序包含对同一个内存变量的多次修改时，第一种方法便体现不出Cache的优势。不过，这种方法存在内存和Cache数据一致性的问题，特别是在DMA中，外设需要访问内存来实现和软件的通信，而如果此内存区域还在Cache中内存中的数据就还是旧的值，外设就会读取到错误的数据。克服这种缺陷的有效方法是允许软件对Cache直接操作，当需要使用DMA等来通信时手动地让Cache将数据写回，或者软件规定一个内存块能否经过Cache缓存。

写后写（posted write）：将数据写入Cache，并在Cache能访问总线时将数据写回内存。这种方式是异步的读写，克服了回写可能造成的数据不一致的情况。

当需要将新的块加载进Cache，而所有的块都被占用时就需要将某个已有的块替换出去。一般有三种主要的策略，分别是：

（1）随机选择被替换的行（随机策略）；

（2）替换最先加载的行，这种情况下Cache相当于一个队列（先进先出策略，FIFO）；

（3）替换最近使用次数最少的块（LRU, Lease Recently Used）。这种策略最常用，而且需要在每行的记录中加入额外的字段。

2.2.3 虚拟存储器中的高速缓存

在使用虚拟存储器的系统中，由于存在虚拟地址和物理地址两种地址，Cache可以使用这两个地址中的一个来进行映射。根据映射使用的地址不同，Cache可以分成以下两种（图2-8）：

1. 物理地址Cache（physical address cache）：直接使用经MMU转化后的实际地址来映射。由于实际地址是唯一的，所对应的存储单元固定，所以标签也使用实际地址。在这种情况下虚拟存储机制对Cache来说是不可见的。

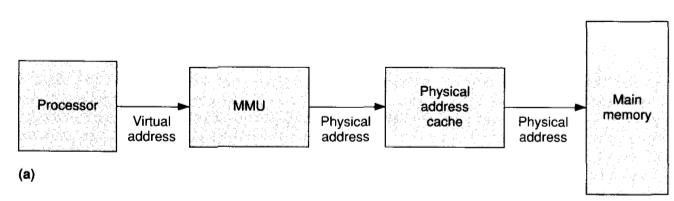


图2-8 物理地址Cache

1. 虚拟地址Cache（virtual address cache）：使用虚拟地址映射，实际地址作标签。由于同一个虚拟地址可以对应多个不同的物理地址，所以要使用物理地址作标签，在访问Cache时要将经MMU转化的物理地址与读出的标签相比较，匹配之后才能说明Cache命中。

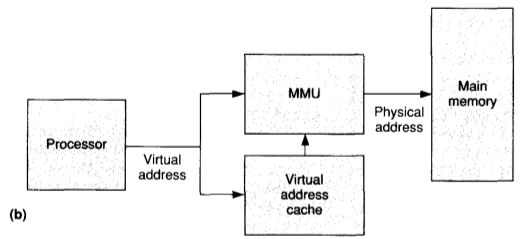


图2-9 虚拟地址Cache

虚拟地址Cache主要的优点就是检查是否命中和地址转换可以同时进行，缺点则是它必须处理虚拟地址重复的问题，因此在读取出块的标签后还需要和物理地址相比较。

2.3 本章小结

本章讨论了虚拟存储器系统和高速缓存的工作原理以及实现方法。对于虚拟存储器，列出了MMU中不同架构对于TLB实现方法的差异以及优缺点，TLB替换算法的基本原理。基本方法就是操作系统负责维护用于地址转换的页表，硬件读取页表并完成地址的转化或者根据情况触发不同的异常。操作系统需要在缺页异常发生时完成物理页的分配，并根据其修改位的值确定是否将该页写回到内存。

详细讨论了MIPS32中的虚拟地址空间划分和地址翻译过程。MIPS只允许访问与当前运行模式特权级相同或较低的特权级的段。地址的翻译过程是采用TLB，并使用ASID字段来区别不同进程的表项。通过页掩码来实现可变大小的页。当TLB不命中时会触发相应的异常，让操作系统从内存中取出页表项。

对于高速缓存，讨论了Cache常用的映射方法、替换算法和写策略以及其相应的优缺点。常用的Cache映射方法有直接映射、全相联和组相联，其中组相联可以看成是前面两者的结合，兼有两者的优点。写策略一般会使用回写或者写后写，因为写直达在程序可能有多次写操作时由于每次都要访存，会消耗大量的性能。

第三章 虚拟存储控制芯片的实现

3.1 系统整体结构及开发环境

本课题需要使用Verilog语言实现一个内存管理单元（MMU）,高速缓存，以及MIPS架构的CPU，并在FPGA芯片上运行。系统的整体结构如图3-1所示。



图3-1 整体结构

FPGA开发板采用ALINX系列开发板AX301，其使用的是ALTERA公司的Cyclone IV的FPGA芯片EP4CE6F17C8，其包含6272个逻辑单元、270KB内存，资源可以实现整个系统。开发软件是ALTERA公司配套的Quartus Prime 17.1.0精简版，此软件可以将Verilog代码以及引脚定义生成FPGA程序并下载到开发板上进行实际调试。

仿真测试使用的是Icarus Verilog，它可以直接运行输入的Verilog文件并生成波形图，可以使用gtkwave等软件查看生成的波形图。

3.1.1 Verilog HDL

整个系统采用Verilog语言实现。Verilog HDL，是一种硬件描述语言，用户可以使用类似于软件语言的方式定义各个信号间的逻辑和时序关系。一份Verilog设计包含一个模块的层次体系，一个模块在逻辑上相当于一个芯片，是一个封装好的具有一定功能的实体，并且包含若干个输入口、输出口以及双向口。在一个模块中可以实例化其他已经定义好的模块并使用导线，即wire类型的变量，将它们链接起来组成更大的模块。在设计中还应定义好并指明顶层模块。

Verilog的所有语句中包含可综合和不可综合的语句，可综合代表可以通过该代码生成实际的电路下载到FPGA中。设计的主要部分应采用可综合语句。不可综合的语句通常用于仿真测试，包括延时、系统函数、wait语句等。Verilog提供了一些在仿真中有用的系统函数，经常用到的包括打印函数$display等。

根据描述的方式可分为开关级、门级、数据流级和行为级，本课题主要采用的是后两种描述方式。在实现时序逻辑时采用寄存器传输级描述（Register Transfer Level, RTL），这是硬件描述语言的一种编码风格，可以很方便地对同步逻辑电路建模。在实际电路设计中，很多设计模型，如本课题多次用到的状态机模型，都属于同步逻辑电路。这是一种包含组合逻辑和触发器的电路，典型结构如图3-2所示

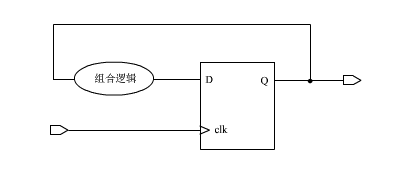


图3-2 同步逻辑电路

在RTL描述中使用数据流级来描述组合逻辑部分，使用行为级描述触发器。对于以上的电路，其RTL描述的Verilog代码大致如下

|  |
| --- |
| assign D = ......;  always @(posedge clk) begin  Q <= D;  end |

3.1.3 模块层次结构

图3-3给出了整个系统包含的主要模块以及层次结构。



图3-3 模块层次结构

3.3 MMU模块的实现

MMU是本次课题的重点之一。MMU需要实现的功能包括将虚拟地址通过TLB翻译成物理地址或触发异常，以及响应处理器的TLB读写指令。根据这些功能可定义模块的各输入信号及意义如下：

clk：时钟信号输入；

res：复位信号；

addrValid：地址翻译指令，为高时表示应开始执行地址翻译；

vAddr：32位虚地址输入；

mmu\_reg：4位的输入，执行MMU中的寄存器的读写操作时用于指明被操作的寄存器。其编码与MIPS标准一致；

mmu\_accessType：2位的输入，表示本次访存操作的操作类型（读数据、写数据、取指令）；

mmu\_dataIn：32位值输入，写入寄存器的数据；

mmu\_cmd：4位输入，不为0时表示要对MMU执行的操作类型。具体的值与操作的对应关系见表3-1；

以及各个输出信号：

pAddr：32位翻译后的物理地址输出；

db\_io：所给虚拟地址是否位于IO段中；

mmu\_dataOut：读取到的寄存器数据；

mmu\_exception：所发生的MMU异常，其编码与MIPS标准中的异常码一致。

其中输入信号mmu\_cmd用于对MMU完成不同的操作，其操作的编码如表3-1所示。

表3-1 MMU指令编码

|  |  |
| --- | --- |
| mmu\_cmd的值 | 对应的操作 |
| 0 | 无操作 |
| 1 | 读MMU的寄存器 |
| 2 | 写MMU寄存器 |
| 3 | 写TLB，以Index为索引 |
| 4 | 随机写入TLB |
| 5 | 查找与EntryHi相匹配的TLB项 |
| 7 | 读索引为Index的TLB表项 |

该设计中MMU模块包含子模块TLB以及随机数生成模块，TLB模块用于保存所有的TLB条目以及按照所给的虚拟地址找出匹配的TLB条目并输出翻译后的物理地址。地址翻译和条目的读取逻辑部分采用的是组合逻辑，其中地址翻译只要输入一改变立即就能得到物理地址输出以及用于检测异常的信号，读取时只需要将读数据使能信号置高并输入要读取的索引，数据就能立即输出。对TLB条目的写入是时序逻辑，在上一个周期将需要写的条目的索引以及写入的数据准备好后要在下一个周期才能得到输出或完成操作。TLB模块的主要输入与输出口有：

（1）与TLB条目有关的四个寄存器的值的输入和输出，即EntryHi、EntryLo0、EntryLo1、PageMask。这四个寄存器位于MMU模块中，TLB在执行条目的读写以及查询指令时需要读取到或更新这四个寄存器的值；

（2）TLB索引输入，用于在TLB读写时指明被操作的条目的索引号；

（3）与地址翻译和TLB查找有关的状态位输出，包括匹配到的条目的索引matchedIndex、说明是否存在匹配的条目的信号found，匹配到的条目的状态位C、D、V，信号名分别是bitC、bitD、bitV；

（4）虚拟地址输入和物理地址输出。输入的虚拟地址将直接通过一系列组合逻辑最终。为了节省FPGA上的逻辑单元，用于地址翻译和TLB条目查找的虚拟地址用的是同一个逻辑电路，通过不同的操作来输入不同的虚地址。显然，当用于TLB条目查找时物理地址输出是没有意义的；

（5）TLB条目的读取和写入的使能信号re和we；

（6）时钟信号和复位信号。

随机数模块结构比较简单，其端口只包含时钟、复位以及输出。用于生成Random寄存器的值和随机写入TLB时被写入项的索引生成，对Random寄存器的读操作就是对随机数模块的读。

下面讨论MMU模块内部的设计方法。对MMU的寄存器和TLB条目的读写操作都只需要一个时钟周期，在TLB中找出与虚拟地址或EntryHi相匹配的项并取出该项的值这部分使用组合逻辑电路，也只需要一个周期，但如果发生异常则还需要写Context寄存器，并且在TLBP指令中还需要把查找的结果写到Index寄存器中，因此所有的MMU操作中最多需要两个周期，这样MMU应是一个状态机。虽然在实现的时候并没有按状态机的标准编码方法，即定义一个变量来保存当前状态，而是只使用了几个变量来标示当前是否在查找TLB，但严格的来讲，由于MMU在某个周期的输出可能依赖于上一个周期的数据，所以此时实现的MMU仍是一个状态机。

寄存器通过reg类型的变量实现，均为32位的变量，对这些变量的读写使用非阻塞的赋值，这样综合工具就会生成触发器。代码如下：

|  |
| --- |
| reg [31:0] reg\_index;  reg [31:0] reg\_entryHi;  ... |

对于寄存器的读写只需在每个时钟的上升沿判断指令输入mmu\_cmd是否为1或2，然后将信号mmu\_dataIn上的数据写入由mmu\_reg指明的寄存器，或将相应寄存器的值写到mmu\_dataOut上。为了便于仿真测试，在对每个寄存器的读写操作后都加上了$display系统函数来打印读写的信息。

而对于可能需要两个时钟周期才能完成的地址翻译和TLB查找指令，第一个周期先将输入的虚拟地址锁存到虚拟地址锁存器，在地址翻译的第二个周期时判断是否发生了异常，如果有就将虚地址的虚拟页号写入Context寄存器的4到22位；在TLB查询的第二个周期将所得的索引值写入Index寄存器。这里便需要两个变量来分别保存上一个周期是否在做地址翻译或查找TLB条目：

|  |
| --- |
| reg prob, convert; |

在初始化（输入信号res为1）的时候这两个变量都会被写为0，之后在每个时钟信号的上升沿时，当信号addrValid有效时convert会变为1，mmu\_cmd代表的指令是TLB查找时prob变为1，这两个信号为1的状态都只会保持一个周期，下一周期来临时都会再次变为0。在每个周期中如果convert为1并且mmu\_exception的值说明有异常发生，就会将输入的虚拟地址中的虚拟页号写入Context寄存器；如果prob为1，就检查是否有匹配的条目被找到，并将匹配的条目的索引写入Index寄存器。之前提到过，这部分逻辑实际上描述的是一个状态机的状态转移表。这两个变量可能取的值有四种组合情况，因此MMU是一个有四个状态的状态机，根据以上变量的转化条件可以画出其状态转移图如图3-4。



图3-4 MMU的状态图

在进行状态转移的时候由于后续的处理也会依赖之前的输入信号，因此对于地址翻译和TLB查询操作应在第一个周期结束，改变状态的同时将输入信号使用锁存器保存下来。在地址翻译时需要保存虚拟地址、CPU运行状态和内存访问类型，其中后面两者会在下一个时钟周期用于检查和触发异常，这三个寄存器分别为vAddrLatch、cpuModeLatch和accessTypeLatch；TLB查找只需要保存虚拟地址，但此时用于匹配的虚拟页号是由EntryHi寄存器给出而非来自处理器的虚拟地址。

除了以上提到的与时序有关的信号外，其余的都是组合逻辑。这些组合逻辑包含异常信号mmu\_exception的生成、控制是否经过Cache的cachable信号的生成以及TLB的index端口的输入。 异常信号等于0时说明没有异常，其他可能产生的异常有TLBS、TLBL、TLB Modified、ADES、ADEL。对于最后两者，本课题只对用户模式和内核模式进行检查。其具体判断过程为：

（1）首先判断当前CPU的模式，如果为用户模式并且虚拟地址的最高位为1，则异常信号为ADES或ADEL，由访存类型决定，否则继续第（2）步；

（2）判断虚拟地址是否处在映射的区域，如果不是，则异常信号等于0，否则继续第（3）步；

（3）如果TLB缺失或者匹配到的条目被标记为无效，则由访存类型决定异常信号等于TLBL或TLBS，否则继续第（4）步；

（4）如果匹配的TLB条目的D位为1并且访存操作是写，则异常为TLB Mod，否则没有异常。

cachable信号的逻辑则相对简单，如果虚拟地址对应的内存区域是非映射区，则cachable直接等于db\_io取反，后者是虚拟地址处于IO段的标志。如果是映射区，则取决于相应TLB条目的C位：MIPS标准中当C位等于2时表示不经过Cache，等于3则表示经过，其他的值可以根据实现自己赋予含义。本课题的实现中不对其他值定义含义。相应的代码如下：

|  |
| --- |
| if(mapped)  case(cacheC)  3'd3: cachable = 1'b1;  3'd2: cachable = 1'b0;  default: cachable = 1'b0;  endcase  else  cachable = !db\_io; |

TLB的index端口的输入口为信号为tlbWriteIndex，其取值为一个多路选择器：当mmu\_cmd为TLB写入或读取时等于Index寄存器的值，为随机写入时等于Random寄存器的值。

综上，在一个时钟周期内MMU的执行流程图如下：



图3-5 一个周期内的流程图

3.3.1 TLB模块的设计

TLB是MMU很重要的一个子模块，它完成了MMU最主要的功能。模块的端口已在前一节给出。TLB模块还包含一个参数ENTRY\_ADDR\_WIDTH（以下简称W），用于指明其TLB条目索引的位数，这样总的条目数N就是1<<W。受使用的FPGA逻辑单元数的限制，在最终实际调试的时候这里的W最大只能取4，即TLB有16个条目。

之前已给出一个条目的组成，一个条目包含四个32位的字段。为了设计的方便，这里用四个寄存器变量来保存，如下：

|  |
| --- |
| reg [31:0] tlb\_entryHi[ENTRY\_COUNT - 1:0];  reg [31:0] tlb\_entryLo0[ENTRY\_COUNT - 1:0];  reg [31:0] tlb\_entryLo1[ENTRY\_COUNT - 1:0];  reg [31:0] tlb\_pageMask[ENTRY\_COUNT - 1:0]; |

TLB条目的读写操作方式与MMU中的寄存器的读写类似，都是使用同步时序电路，写数据使用的索引是index输入。唯一的区别在于读，在读取TLB指令和地址翻译时都需要对TLB进行读取，并且这两个操作不会同时进行。因此为了简化生成的电路，节省逻辑单元，这两个读操作采用同一个地址译码电路，电路的地址输入端用一个多路选择器，信号名为readIndex，通过读使能信号是否有效来确定选择输入的index或匹配到的TLB条目的索引：

|  |
| --- |
| assign readIndex = re ? index : matchedIndex; |

下面讨论TLB条目匹配的实现，匹配逻辑最终需要输出匹配到的条目的索引。由于TLB条目的匹配和一般的相联存储器（CAM）有区别，不是将输入的虚拟页号简单得与每个条目的页号相比较，而是要先与该项的页掩码的反码取与操作再比较，所以不能使用FPGA芯片中提供的CAM而需要手动实现，这使得TLB会使用大量的逻辑单元。条目匹配的电路结构如图3-6所示。



图3-6 匹配TLB条目的电路

所有的条目都需要与虚拟页号以及进程号作比较，每个条目都需要输出一个匹配是否成功的信号，然后将这些信号通过优先编码器转化成索引。在每个条目中，匹配成功需要满足一下两个条件之一：

（1）虚拟地址的页号和表项的页号都与该项的页掩码的反码取与操作后相等；

（2）该项的进程号与EntryHi中的进程号相等或两个G标志位都为1。

实现以上用于匹配TLB的组合逻辑电路使用的是generate语句，先定义一个保存每个条目是否匹配成功的宽度与TLB条目数相同的信号，然后使用该语句为这个信号的各个分量生成一个组合逻辑。这部分代码如下：

|  |
| --- |
| wire [ENTRY\_COUNT - 1:0] matched;  genvar i;  generate  for(i = 0; i < ENTRY\_COUNT; i = i + 1) begin  wire [31:0] entryHi = tlb\_entryHi[i];  wire [31:0] entryLo0 = tlb\_entryLo0[i];  wire [31:0] entryLo1 = tlb\_entryLo1[i];  wire [15:0] mask = tlb\_pageMask[i][28:13];  wire [18:0] tlb\_vpn2 = entryHi[31:13];  wire [18:0] vpn2 = vAddr[31:13];  wire g = entryLo0[0] & entryLo1[0];  assign matched[i] = ((tlb\_vpn2 & ~mask) == (vpn2 & ~mask)) && (g || entryHiIn[7:0] == entryHi[7:0]);  end  endgenerate |

之后，信号matched会被输入优先编码器，编码成匹配成功的条目的索引matchedIndex。另外，由于没有条目匹配和第1个条目匹配时编码器的输出都是0，所以还应定义一个额外的用于指示是否有条目匹配的信号：

|  |
| --- |
| assign found = |matched; |

找出匹配成功的条目的索引后该索引又会被通过readIndex读入从而得到匹配的条目数据，根据这个数据生成翻译出来的物理地址。这部分的逻辑并没有考虑匹配不成功的情况，也就是说当TLB缺失的时候得到的物理地址是没有意义的。为了组合出物理地址先要根据虚拟页号的奇偶性选择出两个物理页号记录中的一个，这依赖于页掩码的取值，因为不同的页大小其虚拟页号的长度不一样，奇偶位的位置也就不一样。同样，在进行地址拼接的时候也需要页掩码，因此先对页掩码进行分析。由于在不止一个地方需要依赖由页掩码确定的页大小，这里先将页的大小进行编码，根据掩码得到编码的值，即把1KB至256MB的页大小分别编码为0至8，编码过程本质上是一个优先编码器，通过一系列if语句实现，大致如下：

|  |
| --- |
| if(~|mask)  matchedPageMaskKind = 4'd0;  else if(~|mask[15:2] && &mask[1:0])  matchedPageMaskKind = 4'd1;  ...  else  matchedPageMaskKind = 4'dx; |

其中最后一项代表页掩码不合法。

将页大小进行编码之后就可以获取到虚拟地址中确定奇偶页号的位，即虚拟页号的最后一位。对于1KB的页，奇偶位位于12位，4KB位于14，以此类推。这里使用case语句实现一个多路复用器来选择出奇偶位。通过该奇偶位的值又可以从两个物理页号记录中选择出相应的那一个，得到selectedEntryLo。输出信号bitV、bitD、bitC直接分别取自该信号的1、2、5到3位。最后物理页号和块内地址根据页的大小拼接成物理地址，同样使用多路复用器来实现。由于这里采用的物理地址是32位的，所以需要忽略物理页号的高6位。1KB的页为将selectedEntryLo的6到25位与虚拟地址的0到11位连接；4KB为将8到25位与0到13位连接，以此类推。至此，所有信号都已生成完毕。

3.4 Cache模块的设计

本课题中的Cache采用虚拟地址Cache，即映射使用虚拟地址，物理地址作标签。由于在之后的测试中会向可执行的区域写数据并执行，所以这里没有将指令Cache和数据Cache分开设计，而采用一个统一的Cache。

具体的映射方法使用相联度为2的组相联，写策略使用回写，写操作缺失时采用写分配。选择组相联的原因是为了克服直接映射中出现需要不断交替访问两个标签不同但Cache索引相同的内存块时出现的连续缺失的情况。如果将指令和数据Cache分开，这种情况出现的可能性就很小，但在本课题中由于是统一的一个Cache，程序在读指令和读写数据时两者的地址所映射的索引就有可能会相同，从而使Cache不断出现缺失的情况。曾使用过的设计就是直接映射Cache，而测试的结果就是程序运行所需要的时间大大增加。而对于替换算法，采用如下的方式：如果一排中的两个块有无效的，则替换无效的，如果都有效，则替换脏位为1，即被修改过的，如果都被修改过，则随机替换。

Cache的主要端口及作用包含：

（1）时钟及复位信号，这是每个模块都有的；

（2）就续指示输出ready。在复位的时候Cache需要使用一个时序逻辑来将所有的块的标签以及服务位初始化，在初始化之前处理器应该等待，当该信号为1时才能开始运行；

（3）物理地址和虚拟地址的输入pAddr与vAddr；

（4）处理器访存接口，用于Cache与处理器之间的连接，接收来自处理器的访存请求并响应。这些信号包括来自处理器的数据db\_dataOut、返回给处理器的数据db\_dataIn、访存操作是否完成的指示信号db\_ready以及访存类型输入db\_accessType；

（5）Cache与内存（或下一级缓存）间的接口。这部分接口和处理器访存接口相似，区别在于信号名，都为dbOut\_\*而不是db\_\*，并且没有db\_accessType信号，还有额外的信号dbOut\_re和dbOut\_we，分别是读取和写入的使能，输出。

Cache模块还包含两个参数：块索引的位数和块内地址的位数。在最终的设计中这两个参数的和不能大于32，而且由于标签使用的是物理地址，所以一个块只能在一个页中，不能出现块的大小大于页的情况。

Cache中包含两个存储器，数据存储器和标签存储器。为了设计的方便，在设计中采用两组以上的存储器，分别对应于一排中的两个行，这样，同一索引对应的两个数据块以及标签和服务位分别来自这两个存储器。数据存储器中每个数据都是32位的类型。每个标签是由物理地址标签和服务位拼接而成。其中的物理地址标签由将物理地址去掉块内地址得到，这样才能保证同一个块中对应的物理地址标签都相同。服务位包含有效位和脏位，当一个内存块被加载到Cache中时相应的有效位就会被写为1。脏位的含义与TLB中的D位相同，即该块是否有被修改过，用于在要将该块替换掉时判断是否需要将该块写回到内存中。这样，标签存储器的地址宽度等于索引的宽度，地址的值等于索引；数据存储器的地址宽度为块索引宽度加上块内地址的宽度减2，其中减2是因为数据的地址是字节编址的，而这里的数据寄存器的数据宽度是32，包含4个字节，对应地址中的两位。数据存储器的输入地址应为块索引和块内地址拼接而成。

存储器使用的是同步读写的RAM，即在一个周期内读取或写入，上第一周期将数据准备好后下一个周期才能得到数据或完成写入。RAM包含数据的输入输出口、地址输入以及读取和写入的使能信号。RAM的实现大致如下：

|  |
| --- |
| reg [WIDTH - 1:0] data[SIZE - 1:0];  reg [ADDR\_WIDTH - 1:0] addrLatch;  assign dataOut = data[addrLatch];  always @(posedge clk) begin  if(we)  data[writeAddr] <= dataIn;  else if(re) begin  addrLatch <= readAddr;  end  end |

以上实现的RAM称为同步RAM（synchronous RAM），因为其读或写都是在一个周期内进行的，两者不会同时进行。之所以采用同步RAM来保存，是因为本课题使用的FPGA中有内建的同步RAM，Quartus软件在综合的时候会根据设计的逻辑来将其用内部资源实现。对于以上的存储器，该软件会生成使用内部RAM的电路（inferred RAM）而不会直接用逻辑单元来实现，后者的地址译码逻辑会占用大量的逻辑单元。

如果将上面代码中的

|  |
| --- |
| assign dataOut = data[addrLatch]; |

改为

|  |
| --- |
| assign dataOut = data[addr]; |

此时读取数据的输出与地址之间就是一个纯组合逻辑，读操作和写操作相互独立并可同时进行。此时这个存储器也叫异步RAM（asynchronous RAM）。由于选择的FPGA芯片中没有异步RAM资源，因此Quartus软件就会直接用逻辑单元来实现。

下面讨论Cache的实现方法。Cache需要完成的功能包括检查是否命中、对存在于缓存中的数据进行读写、通过访存将位于内存中的数据加载到Cache中以及将缓存中的数据块写回到内存中。

时序电路用于实现Cache的缓存读写以及缺失时候的访存操作。整个Cache的时序电路部分实现用的是有限状态机模型，状态的编码及意义如下：

（1）S\_RES：初始化状态，代表Cache正在完成初始化，即把所有的有效位都写为0。初始化需要一个时序操作，因为RAM是同步的，一次只能写入一个条目而不能一次将所有块都写入。当res信号为1时当前状态就会变为该状态；

（2）S\_IDLE：空闲状态，当初始化完成后就会进入该状态，并且会一直保持在该状态直到有访存请求；

（3）S\_CHK\_HIT：检查是否命中。这是访存请求到来之后到达的第一个状态。当转化到这个状态时标签以及数据应已经从存储器中读取出来了，并且这些信号已经通过一个组合逻辑产生了是否命中等信号。此时就会根据是否命中以及服务位来确定下一个状态；

（4）S\_LOAD\_BLOCK、S\_LOAD\_BLOCK\_WAIT：均表示从内存中加载块。加载时是以字为单位加载的，在加载一个字时需要等待dbOut\_ready为1时才能读取下一个。使用两个状态的原因是为了产生合适的访存信号dbOut\_re，因为在读取一个字时的第一个周期内读使能信号才为1，在等待访存的过程中读使能信号应为0。因此在第一个状态中若就续信号为0则应转入第二个状态；

（5）S\_WRITE\_BACK、S\_WRITE\_BACK\_WAIT：与上一个类似，表示将块内的数据写回到内存中。

（6）S\_READ\_FIRST\_W：从缓存中读取块内的第一个字。在写回中写第一个字时需要先将第一个字从数据存储器中读取出来，而之后写回的数据可以在上一次写的时候读；

（7）S\_WRITE\_LAST\_W、S\_WRITE\_LAST\_W\_WAIT：写回最后一个字。在写回的时候当地址为最后一个地址时还没有完成整个写回操作，还需要将最后一个字写回；

（8）S\_LOAD\_LAST\_W：加载最后一个字。在加载数据块时从内存读取到最后一个字后还需要一个周期将读取到的字写到数据存储器中；

（9）S\_WRITE、S\_WRITE\_WAIT：写数据回内存。当执行写操作时发生缺失，应先访问内存将数据写到内存中，然后再到内存中读取出整个块。

当前状态保存在state变量中，一个组合逻辑根据当前状态以及其他输入信号来得到下一个状态nextState，其他的一些信号也依赖于下一个状态的值，因为所有的信号需要在时钟信号上升沿前准备好。

Cache模块的组合逻辑电路包括判断是否命中并从索引对应的两个行中选择出匹配的行和数据，以及输出的访问内存成地址的生成和Cache内部存储器的地址及数据的生成。其中后面两者依赖于下一个状态的值。其中较为重要的信号包括块选择信号hitEntry、victim和whichEntry。hitEntry信号用于指示一排中命中的块，victim信号用于指示将要被替换掉的块，而whichEntry则用于在排中选择出需要用到的块，其取值在Cache命中时等于hitEntry，在缺失的时候等于victim。其余信号的具体依赖关系将在下面的时序逻辑中一并讨论。

下面讨论Cache模块在实现不同操作时的状态转移条件和路径。首先是初始化操作，与初始化有关的状态只有S\_RES，相关的变量是resetIndex，用于标签存储器的地址输入。有关的信号还包括两个标签存储器的写使能，都为1，以及标签存储器的输入数据，也为0，这里只要保证其有效位为0即可。状态转移逻辑较为简单：在每个时钟上升沿，如果resetIndex等于最后一个索引即块数减一，则转移到S\_IDLE状态，否则resetIndex加一。

在访存缺失时会执行块的加载和写回，这里先讨论这两种操作的时序。与块的加载有关的状态包括S\_LOAD\_BLOCK、S\_LOAD\_BLOCK\_WAIT以及S\_LOAD\_LAST\_W，相关的变量是inBlockAddr和addedInBlockAddr，前者是一个与resetIndex类似的变量，是一个寄存器，用于记录当前正在加载的字的地址，后者是下一个地址的值，由于访存的单位是字，因此该变量每次应该增加4，所以这两个变量的关系为

|  |
| --- |
| assign {memEnd, addedInBlockAddr} = inBlockAddr + 4; |

这里的memEnd变量用于指示是否是最后一个地址了，当表达式右侧的值溢出时memEnd就会等于1。相关的信号包括与内存的接口中的地址、读使能564

hkm

第三章 时域积分方程数值方法研究

3.1 时域积分方程时间步进算法的阻抗元素精确计算

时域积分方程时间步进算法的阻抗元素直接影响算法的后时稳定性，因此阻抗元素的计算是算法的关键之一，采用精度高效的方法计算时域阻抗元素是时域积分方程时间步进算法研究的重点之一。

……

3.2 时域积分方程时间步进算法阻抗矩阵的存储

时域阻抗元素的存储技术也是时间步进算法并行化的关键技术之一[14]，采用合适的阻抗元素存储方式可以很大的提高并行时间步进算法的计算效率。

3.2.1 时域积分方程时间步进算法产生的阻抗矩阵的特征

……

由于时域混合场积分方程是时域电场积分方程与时域磁场积分方程的线性组合，因此时域混合场积分方程时间步进算法的阻抗矩阵特征与时域电场积分方程时间步进算法的阻抗矩阵特征相同。

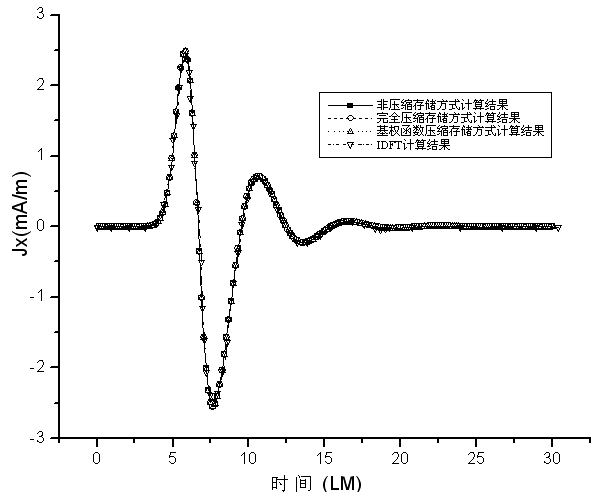
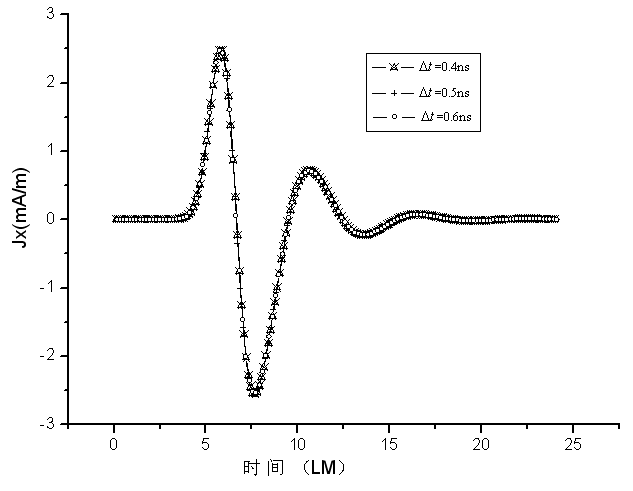
3.2.2 数值算例与分析

……。如表3-1所示给出了时间步长分别取0.4ns、0.5ns、0.6ns时的三种存储方式的存储量大小。……。

表3-1 计算理想导体平板时域感应电流采用的三种存储方式的存储量比较

|  |  |  |  |
| --- | --- | --- | --- |
| 存储方式  时间步长 | 非压缩存储方式 | 完全  压缩存储方式 | 基权函数  压缩存储方式 |
| 0.4ns | 11.96 MB | 5.59 MB | 6.78 MB |
| 0.5ns | 10.17 MB | 5.06 MB | 5.58 MB |
| 0.6ns | 8.38 MB | 4.65 MB | 4.98 MB |

如图3-1(a)所示给出了时间步长选取为0.5ns时采用三种不同存储方式计算的平板中心处方向的感应电流值与IDFT方法计算结果的比较，……。如图3-1(b)所示给出了存储方式为基权函数压缩存储方式，时间步长分别取0.4ns、0.5ns、0.6ns时平板中心处方向的感应电流计算结果，从图中可以看出不同时间步长的计算结果基本相同。

(a) (b)

图3-1 的理想导体平板中心处感应电流*x*分量随时间的变化关系。(a)不同存储方式的计算结果与IDFT方法的结果比较；(b)不同时间步长的计算结果比较

由于时域混合场积分方程是时域电场积分方程与时域磁场积分方程的线性组合，因此时域混合场积分方程时间步进算法的阻抗矩阵特征与时域电场积分方程时间步进算法的阻抗矩阵特征相同。

3.3 时域积分方程时间步进算法矩阵方程的求解

……

定理 3.1 如果时域混合场积分方程是时域电场积分方程与时域磁场积分方程的线性组合……

证明：

首先，由于……

……

根据……，结论得证

3.4 本章小结

本章首先研究了时域积分方程时间步进算法的阻抗元素精确计算技术，分别采用DUFFY变换法与卷积积分精度计算法计算时域阻抗元素，通过算例验证了计算方法的高精度。……

第四章 全文总结与展望

4.1 全文总结

本文以时域积分方程方法为研究背景，主要对求解时域积分方程的时间步进算法以及两层平面波快速算法进行了研究。

……

4.2 后续工作展望

时域积分方程方法的研究近几年发展迅速，在本文研究工作的基础上，仍有以下方向值得进一步研究：

……

致 谢

本论文的工作是在我的导师XX老师悉心指导下完成的，……

……

参考文献

1. W. C. Chew, J. M. Jin, E. Michielssen, et al. Fast and efficient algorithms in computational electromagnetics[M]. Boston: Artech House, 2000
2. 盛新庆.计算电磁学要论[M].北京:科学出版社, 2004
3. 王秉中.计算电磁学[M].北京:科学出版社, 2001
4. 吕英华.计算电磁学的数值方法[M].北京:清华大学出版社, 2006
5. 王长清.现代计算电磁学基础[M].北京:北京大学出版社, 2005
6. 潘小敏.计算电磁学中的并行技术及其应用[D].北京:中国科学院电子学研究所, 2006
7. 中华人民共和国国家技术监督局.GB3100-3102.中华人民共和国国家标准--量与单位[S]. 北京:中国标准出版社, 1994年11月1日
8. W. C. Gibson. The method of moments in electromagnetics[M]. New York: Chapman and Hall/CRC, 2008
9. 胡俊.复杂目标矢量电磁散射的高效算法——快速多极子方法及其应用[D].成都:电子科技大学, 2000
10. H. C. Martin, G. F. Carey. Introduction to finite element analysis: theory and application [M]. New York: McGraw Hill, 1973
11. 金建铭 (著), 王建国 (译).电磁场有限元方法[M].西安:西安电子科技大学出版社, 1998
12. M. Clerc. Discrete particle swarm optimization: a fuzzy combinatorial box[EB/OL]. http://clere.maurice.free.fr/pso/Fuzzy\_Discrere\_PSO/Fuzzy\_DPSO.htm, July 16, 2010
13. S. P. Walker, C. Y. Leung. Parallel computation of integral equation methods for three-dimensional transient wave propagation[J]. Communications in Numerical Methods in Engineering, 1997, 11(6): 515-524
14. 肖珍新.一种新型排渣阀调节降温装置[P].中国,实用新型专利,ZL201120085830.0, 2012年4月25日
15. X. F. Liu, B. Z. Wang, W. Shao. A marching-on-in-order scheme for exact attenuation constant extraction of lossy transmission lines[C]. China-Japan Joint Microwave Conference Proceedings, Chengdu, 2006, 527-529

外文资料原文



外文资料译文

基于多载波索引键控的正交多路复用系统的误码率上界

二．基于多载波索引键控的正交频分多路复用系统模型

我们考虑一个端到端的M-QAM，Nc子载波的基于多载波索引键控的正交频分多路复用系统有n个簇，每个簇有N个子载波（Nc=nN）。M-QAM的符号流经过串并转换之后每n个符号组成一个相量，是和传统正交频分多路复用一样是用来调制子载波的，但是不同的是只有这n个活跃子载波进行了调制。……

……

1. 这里的段不同于段式内存管理中的段。 [↑](#footnote-ref-0)