中国集成电路 CIC
China Integrated Circuit

S698M SoC 芯片中 Cache 控制器的设计与实现

黄琳[1],陈第虎[1],梁宝玉[2],蒋晓华[2],颜军[2]
(1.中山大学理工学院,广东 广州 510275;
2.欧比特(珠海)软件工程有限公司,广东 珠海 519080)

摘要: 高速缓冲存储器 Cache 在微处理器中已经成为至关重要的一部分,它的使用能有效地缓和 CPU 和主存之间速度匹配的问题。本文以 32 位 S698M 微处理器的高速缓冲存储器 Cache 为例,分析了 Cache 的体系结构和关键技术,阐述了 S698M 中 Cache 的基本访存过程。该芯片已采用新加坡特许半导体 0.18 微米 CMOS 工艺流片成功。

关键词: 哈佛体系结构;直接映像;写直达;Cache 一致性;指令 Cache;数据 Cache

The Design and Realization of Cache Controller in S698M SoC Processor

HUANG Lin[1], CHEN Di-hu[1], LIANG Bao-yu[2], JIANG Xiao-hua[2], YAN Jun[2]
(1.School of Physics & Engineering, Sun Yat-Sen University, Guangzhou, 510275, China
2.Orbita Software Engineering Inc. Zhuhai, 519080,China)

Abstract: Cache is a significant part in modern microprocessor. It solves effectively the matching of speed between CPU and main memory. This paper analysis the architecture and key technology of Cache memory, describes basic access memory processing of Cache memory based on a 32-bit RISC microprocessor chip named "S698M". This SoC processor has been manufactured successfully using Chartered 0.18?mCMOS technology.

Keywords: Harvard architecture;Direct-Mapped;Write Through;Cache coherent;I-cache;D-cache

1 引言

随着微电子制造工艺技术的不断提高和微处理器体系结构的改进等多重技术作用的发展,使得 CPU 的速度快速增长。要使系统性能整体提高,在 CPU 速度增长的同时,必然要求系统的存取速度能够与 CPU 匹配。然而主存 DRAM 的速度提高远不能满足 CPU 速度的要求,据统计,CPU 和存储器 DRAM 之间的性能差异以每年 50%左右的速度增长,使存储器较长的访问时间成为系统整体性能提升的瓶颈。为了解决高速 CPU 和低速主存之间速度差异的矛盾,高速缓冲存储器 - Cache 设计思想应

运而生。本文以S698M微处理器的cache控制器为例，介绍了cache的基本原理和工作流程以及Cache在S698M微处理器中是如何设计和实现的。

2 Cache概述

高速缓冲存储器（Cache）位于CPU与主存之间，通常采用静态存储器（SRAM）构成，具有规模小，存取速度快等特点。其基本工作原理是用以保存CPU最常访问的部分主存数据，当CPU读取数据时，首先检查Cache中是否有该数据，如果有，称之为“命中”，直接从Cache中读取；如果没有，称为“不命中”，CPU在主存中寻找到该数据，然后通过数据总线传送给CPU，并且把该数据所在的块传送到Cache中。这种设计方式是以访存的局部性原理为理论基础的。即CPU对存储器的访问并不是随机的，而是存在着时间局限性和空间局限性，“在任何一段时间内，程序都趋于访问较小的一段地址空间，CPU存取指令或数据的操作在时间和空间上往往都集中在一定的范围内进行。”由于CPU反复访问的数据通常放在相邻的存储单元中，可以把这批可能被反复读写的数据预先写入Cache中，当CPU在一定的时间内读取数据时，就能避开低速的主存储器直接从高速的Cache中调用，这样可以有效地缩短CPU的等待时间，从而提高CPU的存取速度。由此可见，Cache能很大程度地提高CPU的整体性能，一个设计良好的高性能Cache在CPU设计中是至关重要的。

S698M微处理器是欧比特（珠海）软件工程有限公司自主研制的S698系列的第三代32位SOC芯片，专门针对恶劣环境所研制。S698M遵循SPARC V8标准，时钟频率最高可达166MHz，其内核电压为1.8V，I/O接口电压为3.3V，支持的操作系统：Rtems、ucLinux、Orbita EOS、Vxworks，已采用新加坡特许半导体（Chartered）0.18微米CMOS工艺流片成功，封装形式为QFP160，S698M的详细介绍见参考文献[1]。处理器内部Cache采用Harvard结构，支持4K字节的指令cache（I-Cache）和4K字节的数据cache（D-Cache），分别连接到独立的Cache控制器上，支持写直达（Write-Through）的写策略。

3 Cache的主要设计技术

在不同的设计要求和应用场合下，各种设计技术对Cache的命中率有很大影响。设计Cache时，主要对以下几个方面进行技术选择。

3.1 Cache结构选择

从结构上分析，主要分为两种结构，一种是将数据和指令都放在一个Cache中的普林斯顿（Princeton）结构，又称为统一Cache。统一Cache能自动协调分配指令Cache和数据Cache的空间；另一种是将指令和数据分别放在两个独立的cache中的哈佛（Harvard）结构，又称分立Cache。分立Cache的主要优点是解决了指令单元和执行单元之间的访问冲突问题，使得指令访问和数据访问能够同时进行。S698M采用哈佛结构的分立Cache，其中指令Cache只能取指令，提供给指令流水线，不能进行写操作；数据Cache用于存取数据，为整数和浮点数操作提供数据，既能读又能写。指令Cache和数据Cache能同时被访问，这种Cache结构使取指令和存取数据的同时进行成为可能，例如：在单周期内完成原本至少要两个机器周期才能完成的Load/Store指令。从而提高了处理器性能。

3.2 Cache容量选择

S698M的指令Cache和数据Cache的容量均为4Kb，也可以根据需求在1～64Kbyte内自行设置Cache的大小。如果CPU反复执行的指令并不是很多时，Cache的容量并不是越大越好，使用频率不高的指令和数据放在主存中即可，太大的Cache速度反而较慢，因为寻址的门数更多。另外，Cache的容量还受到芯片面积功耗以及SRAM的体积和价格的影响。当然cache也不应该太小，载入cache的数

据还未被使用就被置换了，命中率太低不能达到cache的作用。因此，Cache容量的大小取决于CPU读取数据的多少和常用数据的存取频率。经过研究证明，容量在1Kb～512Kb的Cache相对而言是最有效的。

3.3 Cache存储映像方式选择

通常主存的容量远远大于Cache，主存与Cache均被分割成大小相同的块，以块为单位进行数据的交换。当把主存的一个块调入Cache时，具体放在cache的什么位置才能保证准确迅速的查找，这就是存储映像要解决的问题。Cache中用到的存储映像主要有三种：全相联映像、组相联映像、直接映像。在S698M设计中，指令Cache和数据Cache都是采用直接映像方式。在直接映像中，主存中的每一个块在Cache中只有一个位置与之对应，只需比较一次地址就可找到主存的某块在Cache中的位置，查找起来非常方便，所以速度很快且硬件实现简单。具体实现是把主存以同等Cache大小分区，最多可分为220（S698M的TAG最多有20位）个区。主存的每一区又分为若干块，而4Kbyte的Cache也分为若干块（也称为Cache行，即Cache line），我们选择每行大小为16byte，则总共有256个Cache line。对于主存的第i块，映像到Cache的第j行，则有以下的对应关系：

$j = i \bmod (M)$

其中i为主存的块地址，j为Cache的行地址，M为Cache的行数。例如，主存的第0块，第2^a块，第2^{a+1}块……只能映像到Cache的第0行；主存的第1块，第2^{a+1}块，第$2^{a+1}+1$块……就只能映像到Cache的第1行；依此类推。图1是采用直接映像方式的Cache结构。

3.4 Cache替换算法选择

当CPU发出读操作命令时，根据产生的主存地址去Cache中查找，如果命中，即所需数据已经在Cache中，直接访问Cache，从对应的Cache line读取

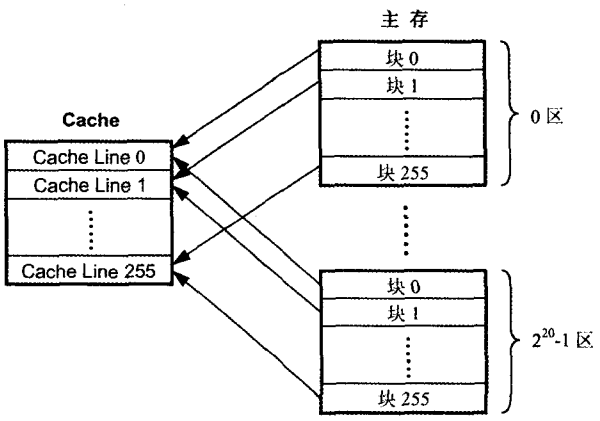

图1 直接映像的Cache结构

信息到数据总线；如果不命中，即所需数据并未装入Cache，此时CPU需要从主存中读取该信息，与此同时，Cache把该地址所在的主存块拷贝到Cache中，如果该主存块在Cache所对应的位置已被其他存储块占据，就必须去掉旧的存储块。采用何种替换方式，能使得访问cache的命中率最高，由替换算法来决定。常见的替换算法有：随机法（random）、先进先出（FIFO）、最近最少使用（LRU）等。由于S698M采用的是直接映像方式，在cache块失效时主存的一个块只能映像到Cache的一个固定对应的块，所以采用的是随机替换算法。随着相联度的增加，Cache的替换算法对Cache的性能有很大影响，应结合命中率，工作频率，硬件代价几个方面综合考虑选择比较好的替换算法。

3.5 Cache写策略选择

如果CPU往Cache的某个单元进行写操作，如何使Cache修改后和主存中对应的保持保持一致呢？在我们的设计中采用写直达（Write Through）的写策略。这种方法是指令或数据同时被写入Cache的同时，也要写入和主存相应的存储块。这种方法简单可靠，保证主存不会出现数据丢失的问题。

4 S698M中Cache的工作流程

下面介绍S698M中Cache的访存基本过程。根

据上述讨论，S698M 的指令和数据 Cache 大小分别定为 4KB，采用直接映像方式，前面已经提到 Cache 基本的存储单元是 Cache 行，在 S698M 的 Cache 中，每个 Cache 行大小为 16 字节。即每行包括 8 个连续的指令或数据字。总共分为 256 个 Cache 行。主存分为 220 个区，每个区的大小等同与 Cache 的大小，每个区又分为 256 块。系统初始化或系统复位时，所有的 Cache 行都为空是无效的，当指令或数据从主存中装入 Cache 后，该行含有了被缓存的指令或数据，称之为 Cache 行有效（Valid）。有效的 Cache 行包含若干个连续的主存的指令或数据字。而 S698M 的指令和数据 Cache 都是由 Tag 标志单元和数据单元组成，分别由两组 RAM 来实现。每一行 Cache Tag 的格式如图 2 所示。

图 2 Cache Line Tag 格式

其中：

Tag[31:12]：地址标志位。根据 Tag 地址标志位能够主存中的哪一块在 Cache 中唯一对应的行位置。

Lock：Cache Line 锁存位。该位使能后可以将 CPU 经常访问的 Cache 行进行锁存，不会被其他 Cache 行替换，提高了 CPU 的取指速度。

Valid[3:0]：有效位。表示相应的 Data RAM 中相应的指令或数据是否有效。在数据单元，每一个 Cache 块可以存放 4 条指令或数据。

I-Cache 和 D-Cache 的读操作类似，下面以 I-Cache 的读操作为例。当 CPU 发出指令访问存储器时，先算出存储器的地址，并在这个地址后附加一个 8 位的 ASI（Address Space Identifiers，地址空间标志符）来指明此次访问的是指令还是数据。S698M 处理器 CPU 内部指令实行单指令发射流水线，把指令执行的过程分成五级流水（Pipeline），分别是取指阶段（Instruction Fetch, FE）、译码阶段（Instruction Decode, DE）、执行阶段（Execute, EX）、存储阶段（Memory, ME）和回写阶段（Write, WR）。对 I-Cache 的访问发生在取指阶段 FE。处理器使用 PC 的内容访问 I-Cache 控制器，I-Cache 控制器接收到 IU 传送过来的 rpc 的地址和读信号后，将这个地址中的对应的块选择位取出，用作 Cache RAM Tag 和 Cache 体数据的译码地址。然后访问 I-Cache 的 tag，找到相应的行，并将该行的 Tag 及有效位取出。将从 RAM 中取回的 Tag 位与 fpc 的相应位(区号)进行比较，如果两者相等(同时有效位为 1)，则表明要访问的指令确实在 Data 中，并从 I-Data 中取出指令；如果不相等或有效位是 0，则为 miss，这时控制器会将地址传给总线，从主存中将指令取出来，传给 IU。同时由控制器写入 Data，在取指阶段结束时，把指令打入指令寄存器（IR）。同时，下一条指令的地址也在 FE 级算出，并把它送入地址寄存器（nPC）。常用的指令的放在一起的，pc 值连续加 1，一般情况下都是连续取指，加 1 的过程实际上就实现了 cache 的预取，一直接收完 4 条指令为止。32 位的指令 pc 值连续加 4，指向下一条指令。因为一条指令占 4 个字节。；当有跳转指令时，pc 值不再连续，一次只取一条指令，也不能进行预取。访问 Cache 的操作和预取判定操作可以看作是同时进行的。为了保证那些经常被访问的指令或数据不被频繁地置换，在设计时，可以把这些经常使用的指令或数据所在的行锁定，不被置换出去，如果这些常用指令或数据不再被使用，对其所在行解锁即可。如果 CPU 往 Cache 的某个单元进行写操作，为了保证 Cache 修改后与主存的数据保持一致，还涉及到回写的问题。此过程相对比较复杂在本文中不再做具体阐述，可参考文献[3]，[4]。其中 D-Cache 的控制器负责 D-Cache 的控制管理以及流程的正常运行。图 3 是 Cache 的查找过程。

5 结束语

由上述分析可以看到，高速缓存 Cache 有效地解决了 CPU 和主存之间速度匹配的问题，对微处理器性能有着至关重要的影响，因此高速缓存 Cache 技术在现今的处理器设计中是不可缺少的技术之

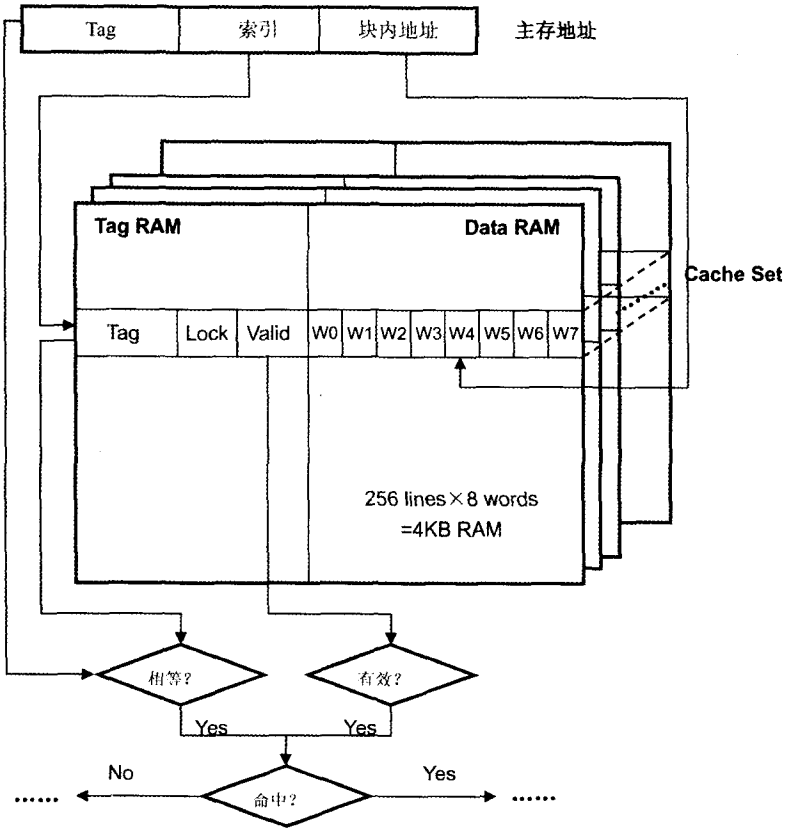

图 3 Cache 的查找过程

一。本文以欧比特公司研制的“S698M”微处理器为设计原型,分析了 Cache 的体系结构和工作原理,整个设计使用 Top-Down 流程,分别采用 ModelSim 和 Synopsys 工具进行仿真和综合,目前项目设计已经全部完成,并在 FPGA 验证平台通过操作系统和典型测试实际程序的仿真测试,结果表明 S698M 中的高速缓存 Cache 设计达到了预期的设计要求,包括逻辑功能、频率、功耗和面积等,采用 Chartered 0.18μm CMOS 工艺流片,频率达到 166 MHz。CIC

参考文献

[1] Orbita Software Engineering Inc. 32-bit SPARC V8 Embedded Processor Sailing S698M User's Manual, www.myorbita.net, 2004

[2] 王爱英. 计算机组成与结构. 清华大学出版社,2001

[3] 王健斌. RISC 处理器指令 Cache 设计及其优化. 硕士学位论文,2004

[4] 田芳芳. 多极系统下数据 Cache 的设计. 硕士学位论文,2006

[5] 陈章龙. 嵌入式处理器的 Cache 结构. 小型微型计算机系统,2004, 7: 1204-1206

[6] 谢学军,叶以正,王进祥,喻明艳. 哈佛体系结构的 Cache 控制器设计. 计算机工程,2004, 22: 38-39

[7] 魏素英,彭洪,林正浩. 高速缓冲存储器的设计与实现. 制造与设计,2005, 18: 86-88

上接第 60 页

[3] R. L. Hummel 编著, 朱莉, 张龙译. 80x86 处理器和 80x87 协处理器大全. 电子工业出版社, 1994 年 3 月

[4] i486(tm) Microprocessor Hardware Reference Manual. Intel Corp., 1990, Order No. 240552-001

[5] 387(tm)DX Math CoProcessor Data Sheet. Intel Corp., Sept. 1990, Order No. 240448-003

[6] S. F. Anderson, J. G. Earle, R. E. Goldschmidt, and D. M. Powers. The IBM System/360 Model 91: Floating-point execution unit. IBM J. Research and Development, 1967, 11(1): 34~53

[7] 387(tm) Numerics Coprocessor Extension Data Sheet. Intel Corp., Feb. 1989, Order No. 231920-005