# 利用冲突预测方法的高缓组织方案

李晓明[1],鲍东星[2],喻明艳[1],叶以正[1]
(1. 哈尔滨工业大学微电子中心,黑龙江哈尔滨 150001;2. 黑龙江大学电子工程学院,黑龙江哈尔滨 150080)

**摘 要**: 本文提出了一种直接映象式高速缓存块冲突预测方法,即借助于高缓块的最近替换行为动态预测冲突发生. 基于该方法,我们设计了一种高缓结构-冲突预测高缓,主体为一个直接映象式高缓和一个较小的全相联高缓,利用冲突预测表实行高缓块的动态分配. 应用于片上数据高缓的 SPEC95 仿真结果表明,与 16kB 直接映象式高缓相比,(8 + 1) kB 冲突预测高缓命中率平均可提高 12.2 %,与类似结构的高缓(如 NTS 高缓、PCS 高缓等)相比降低了硬件开销,简化了控制机构,易于实现,并且在命中率和总线交通量等性能方面都有所提高.

**关键词**: 高缓性能; 命中率; 冲突预测
**中图分类号**: TN72 **文献标识码**: A **文章编号**: 0372-2112 (2003) 05-0724-04

## Cache Management by Using Conflict Prediction Method

LI Xiao-ming[1], BAO Dong-xing[2], YU Ming-yan[1], YE Yi-zheng[1]
(1. *Microelectronics Center, Harbin Institute of Technology, Harbin, Heilongjiang* 150001, *China*;
2. *Institute of Electronics Engineering, Heilongjiang University, Harbin, Heilongjiang* 150080, *China*)

Abstract: A method to predict block conflicts in direct-mapped caches is proposed, which is based on the block replacement behavior. Accordingly, a cache scheme named Conflict Prediction (CP) Cache is presented. Its architecture contains a direct-mapped cache and a small fully-associative cache, and a conflict prediction table (CPT) which dynamically decides the allocation of fetched blocks from next memory hierarchy. SPEC95 Simulation results show that the performance of CP cache is always better than the traditional direct-mapped cache with twice the size of CP cache. With the same block size of 32 bytes, the average improvement of miss ratio of (8 + 1) kB CP cache with 8-entry CPT is about 12.2 % over the traditional 16kB direct-mapped cache. By comparison with other similar architectures such as NTS cache and PCS cache, CP cache not only requires less hardware tradeoff and simpler control but also improves hit ratio and bus traffic.

Key words: cache performance; hit ratio; conflict prediction

## 1 引言

随着深亚微米技术和计算机辅助设计技术的发展,CPU 的性能得到很大提高,与此同时虽然存储器取数时间及带宽也得到改进,但与 CPU 相比要慢得多. 高速缓冲存储器(简称高缓)是弥合片外存储器和 CPU 速度差距最有效的方法. 目前一些处理器采用直接映象式高缓(简称 DM 高缓)做为其 $L1$ 高缓组织形式,因为在容量相同的条件下,DM 高缓比其他映象形式的高缓结构的取数延迟小[1],并且结构简单,数据访问与标签比较可以同时进行,且每次高缓访问只需进行一次标签比较. 但其缺点也很明显:命中率较其它组织形式高缓低. 解决这一问题最常提及的便是牺牲高缓(victim cache)[2],它将 $L1$ 分成容量相差较大的两个存储器,大的组织形式为 DM 式(主高缓),小的一般为全相联式(辅高缓),二者之间存在一条双向数据传输通道. 相似结构的高缓统称为扩充高缓(augmented cache)[3],其中比较著名的包括 PA7200 辅助高缓(assist cache)[4],非时间流高缓(NTS cache)[5,6],程序指针流高缓(PCS cache)[5,6],存储器地址表高缓(MAT cache)[7],选择性高缓[8]等. 上述诸方案可进一步划分为有通道和无通道(NTS, PCS, MAT 等)两种,即是否在主高缓和辅高缓之间有直接片上数据通道. 一般来说,有通道扩充高缓能更有效地增加命中率,因为相关块并未被从 $L1$ 中排除,而是依据一定的方案在主辅高缓之间进行分配. 但是此类方案需要复杂的通道控制逻辑,硬件开销很大,并且由于片上高缓之间频繁的数据块交换会产生大量的功耗. 因此,近年来无通道扩充高缓的研究受到人们的重视.

无通道扩充高缓通过增加某种硬件机构,动态监测 $L1$ 中块的复用性特征,例如时间局部性和空间局部性,然后根据硬件预测机构,有选择地进行块的分配及替换. 无论如何,无通道扩充高缓的命中率一般不如有通道扩充高缓. 这是因为一方面硬件预测精度较低,另一方面块的 $L1$ 保留期不如有通道扩充高缓长,难以保证块的充分有效的复用.

收稿日期:2002-05-27;修回日期:2002-09-04

## 2　直接映象高缓中块访问冲突行为

Gary Tyson[9]等人通过研究高缓失效的原因后发现,在运行 SPEC92 系列标准测试程序后,程序运行时间的 90 %集中在程序代码的 10 %的区域内,对大部分程序而言,少于 5 %的数据装载指令导致多于 99 %的高缓失效,此结果表明数据高缓的访存性能是影响系统性能的关键. 因此本文着重研究片上数据高缓的性能改进方法.

高缓失效的原因有三:强制型失效,空间型失效以及冲突型失效. 冲突型失效产生的原因可分为 T 型冲突和 NT 型冲突. 当多于一个时间局部性块映射到高缓同一组,并在一定时段内被反复访问所造成的失效称为 T 型冲突;映射到高缓同一组的数据块中至少有一个为非时间局部性块,由此块而来的访问冲突称为 NT 型冲突. 此二类冲突是造成冲突型失效及一部分空间型失效的原因,绝大部分冲突都发生在一段有限的时间内,因此我们希望有效的 T 型数据块尽可能久留高缓中,以保持当前工作集有效利用.

在 DM 高缓中,由于块冲突引发的失效率约占总数的 30 %(128kB)～50 %(1kB)[1]. 因此,寻找一种方法充分减少冲突型失效的发生,必将有助于高缓的性能优化.

下面来看一个常见的 DM 高缓中数据块冲突的例子.

```
for(i=1;i<=m;i++){
    访问数据 a 的指令;// a 属于块 A,映射于高缓中组 i
    ……
    for(j=1;j<=n;j++)
    访问数据 b 的指令;// b 属于块 B,映射于高缓中组 i
    ……
    访问数据 c 的指令;// c 属于块 C,映射于高缓中组 i
}
```

从例子中可以看出,数据 $a$, $b$ 和 $c$ 分别属于块 $A$, $B$ 和 $C$,且映射于高缓中同一组 $i$,互相之间将频繁地替换,因此会引发大量的冲突失效. 将块数据流表示为

$$(A_mB_m(B_h)^{n-1}C_m)^m$$

其中下标 $m$ 和 $h$ 分别代表数据块访问的失效和命中,上标 $n$ 和 $m$ 代表程序循环次数. 由此可知 DM 高缓相应的失效率为 $3/(n+2)$. 通过观察我们知道,在第一次外层循环后,块 $A$ 将替换上一次占据组 $i$ 的块 $C$. 此时如果预知冲突可能发生且将其装入另外一个缓冲器中,那么其后的冲突(或部分冲突)将会避免. 那么如何确认冲突块呢?简单地说,如果同一数据块在有限的时间内,被程序多次访问并且被多于一次地替换,那么此块即为冲突块,其主要前提为反复替换行为的发生. 那么经过第一次替换后,如果在有限时域内同一块被访问,便可以判定它很有可能为冲突块. 这种方法称之为冲突预测方法. 如果把被预测为将发生冲突的块装入一个并行的缓冲区中,用此方法,上述例子的块数据流则为

$$A_mB_m(B_h)^{n-1}C_mA_mB_m(B_h)^{n-1}C_h(A_h(B_h)^nC_h)^{m-2}$$

相应的失效率则为 $5/m(n+2)$. 显然,当循环次数 $m$ 和 $n$ 足够大时,命中率能够得到很大提高.

## 3　冲突预测高缓

基于上一节提出的方法,我们将其应用于 $L1$ 数据高缓的设计,便得到一种新的无通道式扩充高缓设计方案.

### 3.1　冲突预测表设计

冲突预测硬件设计的原则为面积小,结构简单且不增加额外的时序负担. 我们采用一个独立的历史表的结构,硬件采用全相联的组织形式的缓冲器,其中存储每次从 $L1$ 中替换掉的高缓块的标签. 我们称此表为冲突预测表,简称 CPT (Conflict Prediction Table). 为了研究的方便,其替换算法采用最近最少使用(LRU)算法. 通过试验,我们发现标签来源直接影响命中率等性能. 当 CPT 只接受来自主高缓的标签时,系统性能提高很大;但当 CPT 既接受来自主高缓的标签又接受来自辅高缓的标签时,系统性能提高很小. 这是由于冲突一般发生在循环体内,表现为一定的时间性,在此期间冲突预测相对比较准确. 当超过此一时段,即根据辅高缓替换的块标签作预测准确率降低,则有可能使得非冲突块大量涌入辅高缓,从而降低主高缓的利用率,造成辅高缓利用过度,降低命中率. 另一方面,复用性较差的高缓块过多占据辅高缓造成高缓污染(cache pollution)问题加剧,也降低扩充高缓的性能. CPT 的大小能够在一定程度上影响命中率.

### 3.2　冲突预测高缓结构

本文提出的冲突预测高缓结构,以下简称 CP(Conflict Prediction)高缓,如图 1 所示. 以 $L1$ 高缓为例,它的主体结构包括:一个容量大、DM 数据主高缓,另一个为容量相对很小的全相联辅高缓,以及一个 CPT. 在每一次执行存储器访问操作时,主高缓和辅高缓并行查询,这包括以下三种情况:

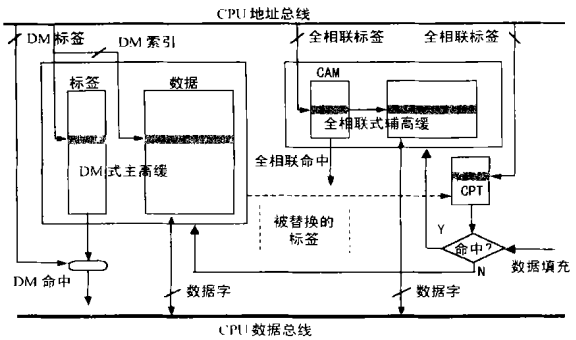



图 1　冲突预测高缓结构示意图

(1) 主高缓命中:如果需要的数据字在 DM 高缓中找到,那么它直接被 CPU 使用. (2) 辅高缓命中:如果需要的数据字在全相联高缓中找到,那么它直接被 CPU 使用. (3) 主高缓及辅高缓均失效:如果需要的数据字在 $L1$ 高缓中未找到,那么必须从下一级存储器(L2 高缓或主存)中将相应的高缓块取回. 此时待取块的标签在 CPT 中进行比较查询,如果找到相等的标签,即预测表命中,那么取回的块将向辅高缓填充;如果该标签未在 CPT 中找到,那么取回的块将向主高缓填充. 填充的同时关键字送 CPU 使用.

当高缓块填充主高缓时,原来占据该组的高缓块的标签将记录在 CPT 中. 如果 CPT 中有空位置,则只需填充之;否则,

由替换算法决定替换其中一条记录项.

## 4 实验仿真

### 4.1 仿真环境

我们采用了 SimpleScalar 工具集[12]中的 sim-outorder 仿真器做为仿真平台,其中采用 mlcache 高缓仿真器[13]来取代 sim-outorder 中相关的高缓部件.由于本实验只检验 $L1$ 数据高缓的性能,因此指令高缓为理想化的. $L2$ 及总线假设为理想结构.仿真器和存储器系统的参数见表 1.

表 1 处理器及存储器参数表

| | |
|---|---|
| 取指机构 | 每周期按程序顺序可取多达 4 条指令 |
| 转移预测 | 2084-entry Bimodal 预测 |
| 发射机构 | 每周期无序发射达 4 条操作,16-entry 重排序缓冲器,8-entry 装载/存储队列 |
| 功能部件 | 4 个整数 ALU,4 个浮点 ALU,1 个整数乘/除部件,1 个浮点乘/除部件 |
| 数据高缓 | 回写式,写分配,每块为 32 个字节,4 读/写端口,无阻塞式访问 |
| $L2$ 高缓 | $L1$-$L2$ 256 位总线,数据带宽 32 字节/周期,$L2$ 高缓设为无穷大,访问延迟为 18 周期,$L1$ 到 $L2$ 的访问为充分流水线式访问 |

### 4.2 标准测试程序(Benchmark)

我们从 SPEC95 标准测试程序集中挑选了六个典型的程序来进行仿真,其中除了 compress 程序采用训练数据集(train data set)做为仿真输入外,其他程序的输入文件均来自测试数据集(test data set).所有的仿真程序都运行到结束,运行后的程序指令数和存储器访问数统计见表 2.

表 2 六个 SPEC95 程序存储器访问参数表

| 程序 | 指令数(百万条) | 存储器访问数(百万条) | |
|---|---|---|---|
| | | 装截 | 存储 |
| SPEC95 整数程序 | | | |
| compress | 35.83 | 7.40 | 6.00 |
| gcc | 291.35 | 68.17 | 38.39 |
| li | 956.87 | 286.32 | 168.85 |
| SPEC95 浮点程序 | | | |
| hydro2d | 967.24 | 191.86 | 58.69 |
| su2cor | 1034.45 | 250.74 | 79.99 |
| swim | 796.56 | 183.24 | 51.60 |

### 4.3 性能标准

对于给定结构的高缓,评价其性能的首选标准为其存储器等效访问周期或存储器访问周期总数.我们定义存储器等效访问周期为

命中率 ×命中周期数 + (1 - 命中率) ×等效失效损耗周期数 (2)

这里我们假设 DM 主高缓和全相联辅高缓命中周期同为 1 个时钟周期,而等效失效损耗为 18 个时钟周期.对于本实验,失效率为

$L1$ 数据高缓失效数/$L1$ 数据高缓访问总数 (3)

为了便于比较,我们定义了加速比参量

目标结构存储器等效访问周期/基准结构存储器等效访问周期 (4)

此外,我们也给出了不同高缓结构的失效率.总线交通量(bus traffic) ——即 $L1$ 与下一级存储器通过总线交换的数据量 ——是另一个重要的性能参数,对于整个处理器系统而言,总线交通量越小,系统的速度也会越快.本文将由数据高缓申请的通过总线的数据总量(以百万字为单位)做为总线交通量的衡量标准.为便于比较,我们定义相对总线交通量为

基准结构的总线交通量 - 目标结构的总线交通量 (5)

相对总线交通量为正数值,说明目标结构的总线交通量比基准结构的小.正值越大,目标结构性能越好.

### 4.4 用于比较的高缓结构

限定所有参与比较的扩充高缓容量固定为(8 + 1) kB,其中主高缓为 8kB,辅高缓为 1kB.做为基准结构的 DM 高缓则分别为 8kB 和 16kB,此外还对 8kB 2 路组相联高缓的性能进行了仿真以便于对比.参与比较的无通道扩充高缓为 NTS 高缓[5,6]和 PCS 高缓[5,6].为便于对比,也将两种有通道扩充高缓 ——牺牲高缓[2]和 HP 7200 辅助高缓[4]的仿真结果给出.所有的全相联结构的硬件组织都应用 LRU 替换算法.

### 4.5 仿真结果

**4.5.1 失效率** 表 3 给出了不同组织形式的 $L1$ 数据高缓结构在运行 6 个 SPEC95 程序后得到的失效率的数值结果.表中给出 CP,NTS 和 PCS 高缓中监测表的大小,如 NTS:16-DU 表示 DU 为 16 条记录项,CP:8-CPT 表示其 CPT 为 8 条记录项.从结果可以看出,对于不同程序,随着 CPT 大小的变化,CP 高缓失效率是变化的,平均来看,当 CPT 记录项为 8 条左右时,其平均失效率为最低,与 16kB DM 高缓相比平均减少约 12.2 %,与 8kB 2-路组相联高缓相比减少 16.2 %.同时从表中亦可看出,8-CPT CP 高缓失效率总是低于 NTS 高缓,分别平均减少 6.5 %(16-DU NTS)及 7.3 %(8-DU NTS).8-CPT CP 高缓失效率也低于 PCS 高缓(除了在 swim 中失效律高于 16-DU PCS 高缓),分别平均减少 1.1 %(16-DU PCS)及 11.7 %(8-DU PCS).

**4.5.2 加速比** 上述部分结构相对做为基准结构的 8kB DM 高缓的加速比如图 2 所示.柱状图中相应矩形块越低,代表该结构比 8kB DM 高缓的等效存储器访问周期速度越快.在运行的 6 个 SPEC95 程序中,综合来看 CP 高缓的加速比性能要好于 NTS,PCS,传统的 16kB DM 高缓以及 8kB 2 路组相联高缓,但是一般要比牺牲高缓差.在其中 4 个程序(compress,gcc,li 及 hydro2d)中 CP 高缓略好于辅助高缓.

表 3 不同组织形式的 $L1$ 数据高缓结构的失效率

| | compress | gcc | li | hydro2d | su2cor | swim |
|---|---|---|---|---|---|---|
| DM:8kB | 0.0673 | 0.0462 | 0.0231 | 0.1195 | 0.0891 | 0.4068 |
| DM:16kB | 0.0557 | 0.0288 | 0.0178 | 0.1063 | 0.0796 | 0.1424 |
| 2-路:8kB | 0.0563 | 0.0300 | 0.0148 | 0.1112 | 0.0768 | 0.3904 |
| 牺牲高缓 | 0.0553 | 0.0259 | 0.0141 | 0.1094 | 0.0709 | 0.0473 |
| 辅助高缓 | 0.0562 | 0.0287 | 0.0148 | 0.1095 | 0.0722 | 0.0472 |
| NTS:16-DU | 0.0564 | 0.0300 | 0.0150 | 0.1099 | 0.0749 | 0.1005 |
| NTS:8-DU | 0.0564 | 0.0306 | 0.0151 | 0.1100 | 0.0754 | 0.1033 |
| PCS:16-DU | 0.0592 | 0.0320 | 0.0161 | 0.1138 | 0.0730 | 0.0607 |
| PCS:8-DU | 0.0590 | 0.0327 | 0.0164 | 0.1143 | 0.0749 | 0.1113 |
| CP:16-CPT | 0.0550 | 0.0284 | 0.0143 | 0.1099 | 0.0736 | 0.0752 |
| CP:8-CPT | 0.0549 | 0.0285 | 0.0144 | 0.1094 | 0.0729 | 0.0766 |
| CP:4-CPT | 0.0551 | 0.0290 | 0.0146 | 0.1094 | 0.0733 | 0.0766 |

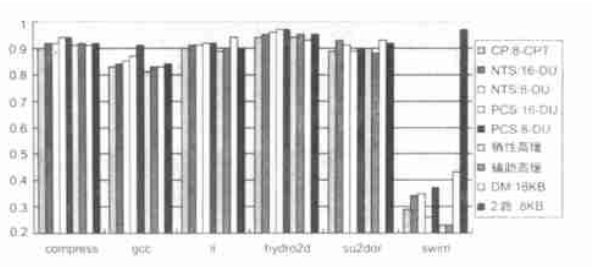



图 2　九个高缓结构相对于 8kB 直接映象高缓结构的加速比

**4.5.3　总线交通量**　将 16kB DM 高缓与下一级存储器总线上数据交换总数为基准,和上节提到的 CP 高缓,NTS 高缓,PCS 高缓以及 2 路组相联高缓的相应值按式(5)进行比较,便得到表 4. 从表中可以看出,在总线读写次数上,CP 高缓几乎总是最少的. 总线交通量的减少,意味着占用较少的总线周期,这有利于系统速度的提高和系统动态功耗的降低.

表 4　8 个高缓结构与 16kB DM 高缓的相对总线交通量数据表(单位:百万字)

| | compress | gcc | li | hydro2d | su2cor | swim |
|---|---|---|---|---|---|---|
| CP:8-CPT | 0.3 | 1.2 | 16.7 | - 5.4 | 23.4 | 166.7 |
| NTS:16-DU | - 0.007 | - 0.7 | 13.3 | - 6.6 | 16.5 | 104.1 |
| NTS:8-DU | - 0.002 | - 1.5 | 13.0 | - 6.7 | 15.1 | 96.2 |
| PCS:16-DU | - 0.3 | - 3.8 | 7.6 | - 15.4 | 22.8 | 187.5 |
| PCS:8 - DU | - 0.2 | - 4.8 | 5.7 | - 16.4 | 17.0 | 73.1 |
| 牺牲高缓 | - 11.1 | - 64.4 | - 122.9 | - 319.8 | - 309.1 | - 449.5 |
| 辅助高缓 | - 8.2 | - 30.1 | - 57.9 | - 274.9 | - 212.9 | 124.8 |
| 2 - 路:8kB | 0.1 | - 0.4 | 16.4 | - 8.8 | 11.4 | - 144.0 |

**4.5.4　其他性能比较**　CP 高缓与 NTS 以及 PCS 高缓方案相比,由于不需要与每一高缓块相连的局部性监测单元 NTDU,并且实行一种简单的预测表应用方式,面积大幅度减少,控制机构简单,硬件实现容易. 例如同为(8 + 1) kB,每块 32 字节的配置,假设全相联高缓 CAM 中高缓标签为 27 位,那么 8-CPT CP 高缓增加的存储单元为 8 ×27 = 216 位;8-DU NTS 高缓增加的存储单元为两部分:DU 为 8 ×(27 + 1) = 224 位;NTDU 存储单元面积计算如下:

主高缓的 NTDU:8 ×1024/ 4 + 8 ×1024/ 32 = 2048 + 256 = 2304 位

辅高缓的 NTDU:1 ×1024/ 4 + 1 ×1024/ 32 = 256 + 32 = 288 位

总的 NTDU 存储单元面积 = 2304 + 288 = 2592 位

那么,8-DU NTS 高缓总的新增存储单元 = 2592 + 224 = 2816 位,远远多于 CP 高缓. 而 PCS 高缓增加的存储单元面积会更大,这是因为其 NTDU 中每一项需要增加相应的 PC 指针,所以不太适合硬件实现. 总之,CP 高缓比 NTS 和 PCS 高缓新增硬件开销要低一个数量级以上.

另一方面,与牺牲高缓以及 HP PA7200 辅助高缓等有通道扩充高缓相比,虽然命中率有所下降,但控制机构要简单得多,无需软件编译器方面的支持. 由于切断了片上的数据通道,彻底清除了主辅高缓之间大量数据交换引发的功耗,这对于微处理器及 SoC 设计中高缓低功耗设计具有一定的意义.

## 5　总结

本文提出一种简单的冲突型失效预测方法,由此提出的冲突预测高缓方案,与类似结构的其他方案相比硬件开销小,同时提高了存储器等效访问周期速度. 由于只增加了一个很小的全相联历史表以用来对数据块进行选择性分配,因此控制机构简单. SPEC95 仿真试验结果证明该结构的性能指标,均超过二倍容量的直接映象高缓. 与其他典型无通道扩充高缓的 NTS 和 PCS 高缓方案相比,也显示出它的优势.

**参考文献**:

[1] D A Patterson, J L Hennessy. Computer Architecture a Quantitative Approach [M]. 北京:机械工业出版社,1999. 396 - 398.

[2] N P Jouppi. Improving direct mapping cache performance by the addition of a small full associative cache and prefetch buffers [A]. Proc. of ISCA - 17 [C]. Seattle, USA, 1990. 364 - 373.

[3] J-K Peir, et al. Functional implementation techniques for CPU cache memories [J]. IEEE Trans on Computers, 1999, 48(2): 100 - 110.

[4] G Kurpanek, et al. PA7200: A PA-RISC Processor with Integrated High Performance MP Bus Interface [R]. COMPCON Digest of Papers, San Francisco, USA, 1994. 375 - 382.

[5] E S Tam, et al. Active management of data caches by exploiting reuse information [J]. IEEE Trans on Computers, 1999, 48(11): 1244 - 1258.

[6] J A Rivers, et al. Evaluating the performance of active cache management schemes [A]. Proc. of ICCD '98 [C]. Austin, USA, 1998: 368-375.

[7] T Johnson, W W Hwu. Run-time adaptive cache hierarchy management via reference analysis [A]. Proc. of ISCA - 24 [C]. Boulder, USA, 1997. 315 - 326.

[8] A Gonzalez, et al. Data cache with multiple caching strategies tuned to different types of locality [A]. Supercomputing '95 [C], San Diego, USA, 1995. 338 - 347.

[9] G Tyson, et al. A modified approach to data cache management [A]. Proc. of MICRO-28 [C]. Arm Arbor, USA, 1995. 93 - 103.

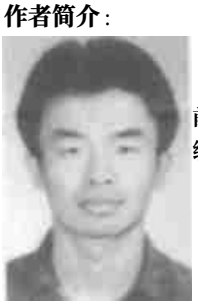
[10] L K John, A Subramanian. Design and performance of a cache assist to implement selective caching [A]. Proc. of ICCD '97 [C]. Austin, USA, 1997. 510 - 518.

[11] S McFarling. Cache replacement with dynamic exclusion [A]. Proc. of Annual Symposium on Computer Architecture [C]. Gold Coast, Australia, 1992. 191 - 200.

[12] D Burger, T MAustin. Evaluating future processors: The SimpleScalar Tool Set. Technical Report # 1342 [R]. University of Wisconsin, 1997.

[13] E S Tam, et al. mlcache: A flexible multilateral cache simulater [A]. Proc. MASCOTS '98 [C]. Motreal, Canada, 1995. 19 - 26.

**作者简介**:



**李晓明**　男,1969 年出生于大庆,博士生,目前主要从事 SoC 平台,微处理器设计以及存储系统管理等研究工作.