高速缓冲存储器在Blackfin®处理器中的应用

撰稿人 Kunal Singh

Rev 1 – June 13, 2005

引言

本应用笔记讨论Blackfin®处理器中高速缓冲存储器的管理，本文首先介绍了受欢迎的高速缓存方案，然后详细介绍Blackfin的指令高速缓存和高速数据缓存，并提供ADSP-BF533 Blackfin处理器的高速缓冲存储器进行管理的程序代码实例。所有Blackfin处理器都具有这些特性，本文假定读者都熟悉高速缓存方面的基本术语。

高速缓冲存储器的概念

本节讨论通用高速缓冲存储器模型， Blackfin存储器的实际模型在下一节讨论。

存储器配置

需要大量存储器的系统通常采用具有不同级别的存储器配置，最高级的存储器（*L1存储器*）拥有最好的性能和最高的成本，低级别的存储器需要更多的访问周期，但是成本低廉很多。

用户不能直接访问高速缓存，但在高速缓存控制器控制下可用作高级存储器。在低级别存储器中存在较大的指令和数据段时，高速缓存控制器允许频繁使用的程序代码和数据导入到高速缓存（高速缓存控制器控制下），因而可以单周期访问，就像在L1存储器一样。高速缓存的结构是基于处理器存储空间已划分为多个有固定大小的数据块（称为cache-lines）这一事实。数据块可认为是存储器中最小的存储单元，当高速缓冲未命中时可从外部存储器传输到高速缓冲存储器。

任何对存储器的引用都可确定为是对某一特定存储器块的引用。图1描述了一个存储器配置，其中外部存储器空间分成24个存储器块，高速缓冲存储器分为6个块（这是一种通用的存储器配置，存储器块的数量随Blackfin处理器模型的不同也不同）。外部存储器和高速缓冲存储器的块大小相同。由于高速缓冲存储器有6个块的大小（在本特殊实例中），因此，任何时候在高速缓冲存储器中可用的主要存储器最多有6个数据块。

块布局

当处理器引用已在高速缓存上可用的存储器块时（在主存储器当中），处理器从高速缓冲存储器访问数据，这就是我们所说的*高速缓存命中*。

当处理器引用高速缓存中不可用的存储器块时，就称为*高速缓存未命中*。高速缓存未命中时，高速缓存控制器将要引用的存储器块从低级别存储器移动到高速缓冲存储器中。

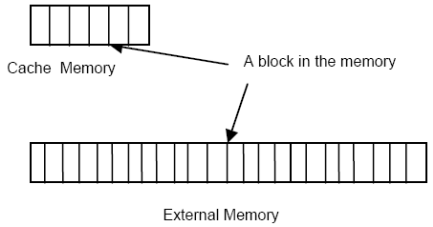

图1 块大小固定的存储器布局

如何确定输入块在高速缓冲存储器的位置，通常有三种方案可用。

直接映射的高速缓存

低级别存储器中的每一个块在高速缓存中只有一个固定的目标块，因此从低级别存储器到高速缓冲存储器是一对一的映射，而这种映射基于低级别存储器中块的地址。

全关联高速缓冲存储器

主存储器中的块可以替换高速缓冲存储器中的任何块，这种映射完全是随机的。

组关联高速缓冲存储器

高速缓冲存储器排列分为多个组，一组又由多个块构成。低级别存储器中的任何块在高速缓存中都有固定的目标组（可放置在目标组中），进入的块可替换该固定组中的任意一个块，若该组中有m个块，则这种高速缓存配置就称为m路组关联。图2描述了一个双组关联存储器。

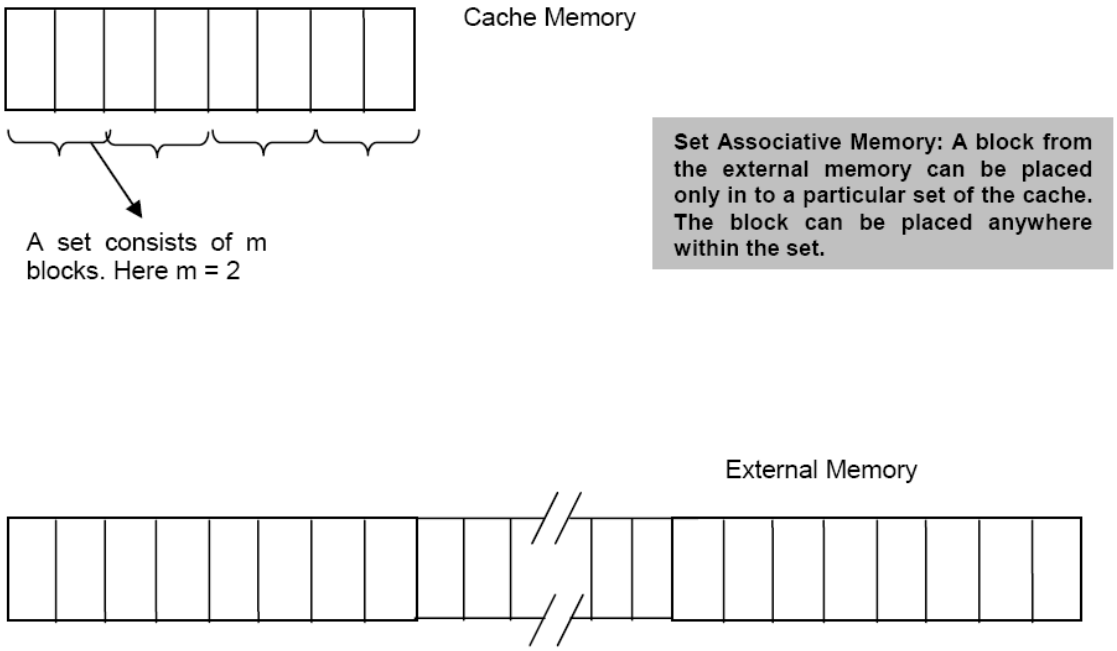

图2 双集关联高速缓冲存储器配置

整块替换

在直接映射的高速缓冲存储器中，来自低级别存储器的块总放置在固定位置上。然而，使用全关联或组关联放置，选择的很多块可能是高速缓存未命中。这些有可能被替换的块称为*共享块*。以下是选择将被替换块的一些主要策略。

(a) 随机替换：目标块从分享块中随机选择，使用伪随机数据产生器选择目标块，这是最简单效率最低的方式。

(b) 先进先出（FIFO）替换：进入的块替换最旧的分享块。

(c) 最近最少使用（LRU）替换：在该方案下，将记录对所有块的访问，被替换的块将是最长时间没有使用过的块。LRU 依赖此现场推论：如果最近使用的块可能还会再次使用，则替换的块最好选择最近最少使用的块。

(d) 改进的 LRU 替换：在该方案下，每个块都分配一个低优先级或高优先级。

如果进入的块为低优先级，则只有低优先级的块才可参与高速缓存替换。

如果进入的块为高优先级块，则所有低优先级块都参与高速缓存替换。如果存储器中没有低优先级块，则高优先级块参与替换。

LRU 策略用于在共享块中选择必须要放弃的块。

块标识

并非所有的高速缓存块都有有效的信息内容，例如：系统启动时，高速缓冲存储器中就不包含有效数据。当发生高速缓存未命中时，高速缓存块就被填充。因此，必须有机制来识别高速缓存块中是否有有效数据。要识别这种有效性，每个高速缓存块中都关联了一个有效位，当高速缓存块中填充有效数据时，相应的有效位将被置位。

除有效位之外，高速缓存中的每个数据块还有与之关联的地址标签。地址标签提供了高速缓存的外部存储器块物理地址，当引用外部存储器块时，高速缓存控制器就将外部地址和高速缓存地址标签比较（与有效位比较），确定组中是否包含该数据块。如果块地址与高速缓存地址标签中的一个匹配，则为高速缓存命中。

在组关联高速缓冲存储器配置下，每个存储器地址都可看作是三个位段的组合，这些位段如表1所示。第一级划分介于块地址和块偏移量之间，块（帧）地址还可进一步分为标签位段和索引字段。

块地址		数据块偏移量
标签位段	索引位段	
高速缓存控制器控制用于检测高速缓存命中或未命中	用于将给定的块映射到特定的组	用于选择给定的数据块当中的字

表 1 地址分区

块偏移量位段用于从块中选择想要得到的数据，索引位段用于选择集合，而标签位段用于比较高速缓存是否命中。

写策略

以下部分讨论与高速缓存相关的存储器写操作的两个不同问题。

高速缓存命中时的写操作

写高速缓存时有两种基本方式：

- Write-Through（直写）：信息写入到高速缓存中的块和源存储器中的块。
- Write-Back（回写）：信息只写入到高速缓存中的块。修改后的高速缓存块在被替换时再写入到主存储器。

为了减少替换时回写块的频率，通常使用称为页面重写标志位（dirty bit）的功能，该状态位用于标志块是否是已重写（在高速缓存中已修改）或未重写（未修改）。如果块未重写，则在替换时不会回写。尽管依赖于应用方式，但回写模式比直写模式仍可提高 10-15%的效率，但在多个源（如 DMA 控制器和处理器）之间必须保持一致性时，直写模式仍是最好的选择。

高速缓存未命中时的写操作

由于在写操作时数据并非必须，因此，在高速缓存未命中时有两种方式：

- Write Allocate（写分配）：在写未命中时分配数据块，其后紧随以上描述的写命中操作。在该方案中，写未命中与读未命中的行为类似。
- No-Write Allocate（非写分配）：在该选择下，写未命中不会影响高速缓存，只有低级别存储器中的被修改，且不会进行高速缓存操作。

Blackfin 高速缓冲存储器模型

Blackfin 处理器有三级存储器，提供容量与性能间很好的平衡。Level 1 存储器（即已知的 L1 存储器）提供最高的性能，但容量最小，Level 2（L2）和 Level 3（L3）存储器有更大的存储器容量，但通常需要多周期访问时间。

Blackfin 处理器有片上数据和指令存储器块，并可独立配置成 SRAM 和高速缓存。当存储器配置为高速缓存时，DMA 控制器不能对其进行访问。所有 Blackfin 处理器都有相似的高速缓存配置，以下是基于 ADPS-BF533 处理器的，作为参考。

Blackfin 指令高速缓冲配置

高速缓存组织

ADSP-BF533 处理器有 80K字节的片上指令存储器，其中 64K字节可作为指令SRAM，其余 16K字节可配置为指令高速缓存或指令SRAM。图 3说明了ADSP-BF533 处理器的存储器映射，并描述了可配置为指令高速缓存/SRAM的指令块。

当使能为高速缓存时，指令存储器作为 4 路组关联存储器工作，每一路都可以独立锁定。对于块替换，指令高速缓存控制器可配置为采用改进的 LRU 策略或者 LRU 策略。

16K 字节的高速缓存组织为四个 4K 字节的子阵，每个子阵由 32 个组构成（每个集合中有 4 个块），块大小（高速缓存未中时可读）为 32 字节。外部数据读取端口（用于高速缓存控制器）为 64 位宽（8 字节）。因此，读取一个完整的块是四个 8 字节宽的数据突发传输。

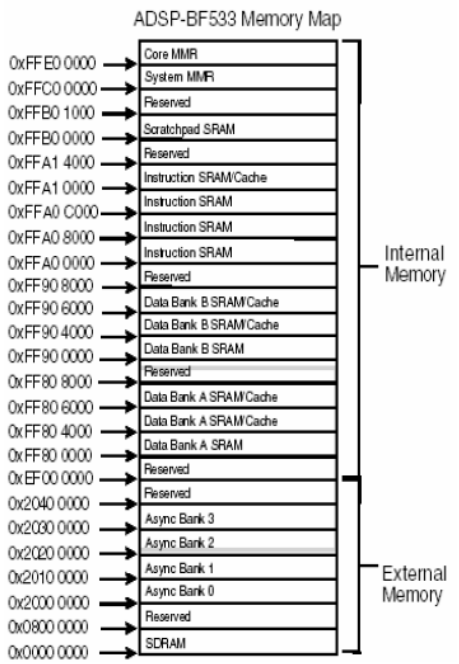

图3 ADSP-BF533 存储器映射

每个块都有一个与之关联的标志部分，标志由 4 部分构成—地址标志、LRU状态、LRU优先级和有效位。图 4说明了高速缓存块中标志部分的位段。在以下讨论中，*cache-block（高速缓存块）*和*cache-line（高速缓存行）*可以交换使用。

图 5说明了如何将 32 位地址空间映射到高速缓冲存储器空间中。

图 6说明了 4K字节的子块是如何在指令高速缓存中组织安排的。

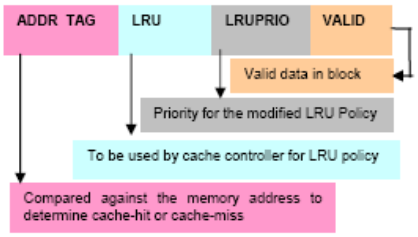

图4 指令高速缓存块中的标志部分

高速缓冲存命中/未命中与行替换

通过将取指令地址中的高 18 位以及位 11 和位 10 与当前存储在高速缓存组中有效行的地址标志进行比较，即可确定高速缓存是否命中，若上述地址标志比较相匹配，即命中，否则为未命中。

如果考虑访问地址 0x10374956，该地址映射到子块 0 中的组 10，该地址的高 18 位以及位 11-10（这些位构成标志字）将不断与组 10 的所有有效标志比较。

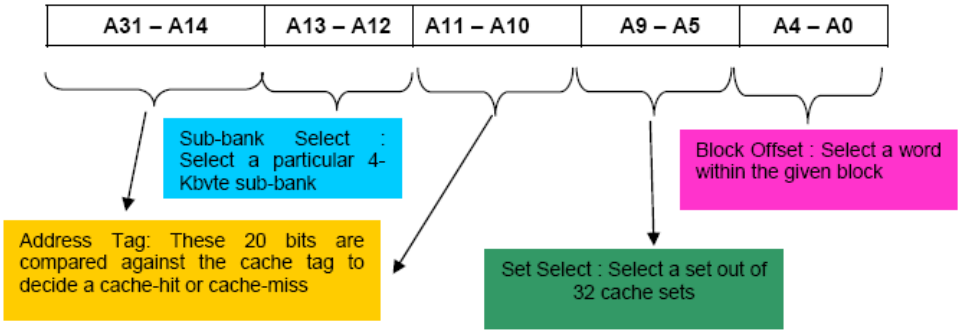

图5 将外部存储器地址空间映射到指令高速缓冲存储器空间

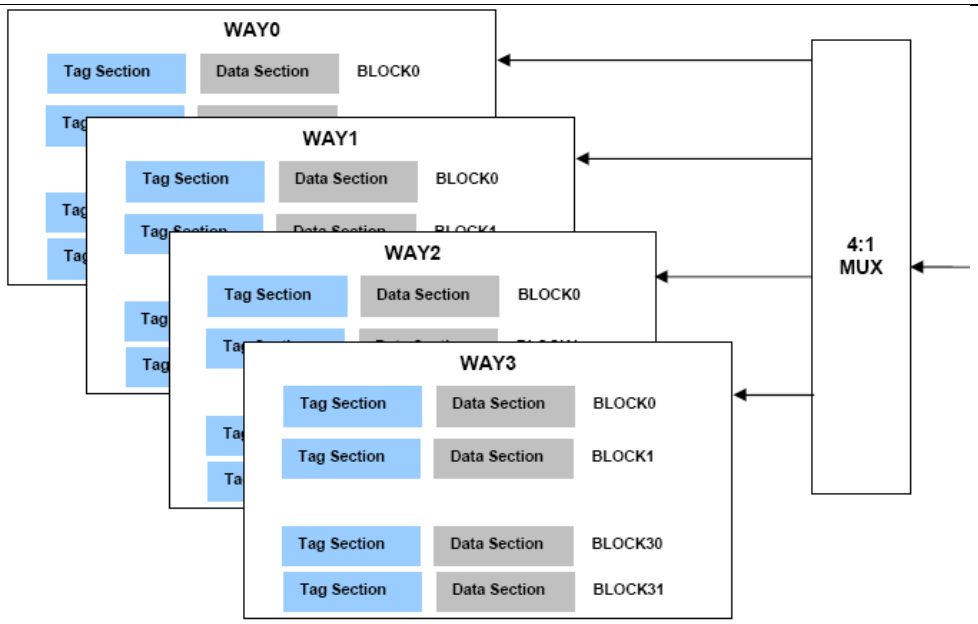

图 6 Blackfin 指令高速缓冲存储器配置 – 4 路组关联

当高速缓存未命中时，指令存储器单元会生成一个高速缓存行填充访问，从外部存储器中恢复未命中的高速缓存块。同时，在目标指令字从外部存储器返回前，处理器核将一直处于挂起状态。

高速缓存行替换单元使用Valid位和LRU位（未锁定通道的）来确定究竟使用哪个块用于新高速缓存行。图 7说明了如何选择高速缓存行替换策略。

Only One Invalid Way in the Set	Incoming block would replace this block
More than one Invalid Way in the Cache	The ways would be replaced in the following order: Way0 first, then Way1, then Way2, Way3
No Invalid Ways in the Cache	Least Recently Used (LRU) way would be replaced. **For Modified LRU policy** : Way with high priority would not be replaced if a low-priority way exists in the given set. A low-priority block cannot replace a high-priority block. If all ways are high priority, the low-priority way cannot be cached.

图 7 Blackfin 指令高速缓存的高速缓存行替换方式

高速缓存行填充访问由从存储器中取 32 字节(一个块）构成，读传输操作的地址就是目标指令字的地址。当指令存储器单元响应行读取请求时，外部存储器首先返回一个目标指令字，接下来以顺序地址读取三个字，如表2所示。

目标字	接下来三个字的取指顺序
WD0	WD0，WD1，WD2，WD3
WD1	WD1，WD2，WD3，WD0
WD2	WD2，WD3，WD0，WD1
WD3	WD3，WD0，WD1，WD2

表 2 高速缓存行字的取指顺序

当高速缓存块从外部存储器恢复后，每个 64 位的字在写入 4K 字节存储器块前都被缓冲到一个有 4 入口的行填充缓冲器中。行填充缓冲器允许处理器核从新的由外部存储器恢复的高速缓存行访问数据，而不必等待到这些行写入高速缓存后。

指令高速缓冲存储器的通路锁定

指令高速缓存有四个独立的锁定位（这些位在指令存储控制寄存器中可用），独立锁定四路中的任意一个。

当某路被锁定后（将指令存储控制寄存器中对应位置位），该路就不会进行块替换，只有 IFLUSH 指令才能从锁定的通路中去除高速缓存的指令。

Code-2 实例（参考相关的 ZIP 文件）说明如何将更为频繁使用的函数（外部存储器中）缓存和锁定，使之不被替换。方案包括锁定 Way1，Way2，Way3（Way0 不锁定）通路，并对感兴趣的函数进行虚拟访问，将这些函数缓存到 Way0 中（因为其他通路都被锁定），此时就可以锁定 Way0 了（Way1，Way2，Way3 不锁定），随后的任何高速缓存未命中都只会替换 Way1-3 中的块。

Blackfin 数据高速缓冲存储器配置

高速缓存组织

ADSP-BF533 处理器有 64K 字节的片上数据存储器，其中 32K 字节可用作数据 SRAM，其余 32K 字节可配置为两个独立的 16K 字节的存储器块，这两个块可配置为数据高速缓存或数据 SRAM。

当数据高速缓存使能时，它可用作一个双组关联存储器。数据高速缓存控制器为块替换使用 LRU 策略（但控制器不能像指令高速缓存一样运用改进的 LRU 策略）。每个 16K 字节的高速缓存块也被分为四个 4K 字节的子阵。每个子阵由 64 个组构成（每个组有 4 个块，块大小为 32 字节）。取决于可配置为高速缓存的存储器块数量，数据高速缓存可是 16K 字节或 32K 字节。

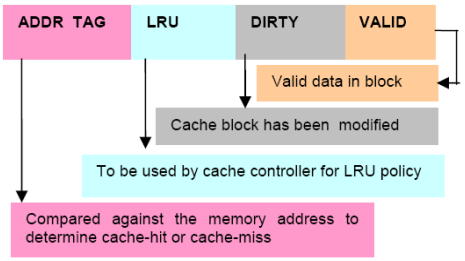

图 8 数据高速缓存块的标志部分

与指令高速缓存类似，每个块都有一个与之关联的标志部分，图 8说明了数据高速缓存块中的标志部分，图 9说明了 4K字节的子块是如何在数据高速缓存中组织安排的。

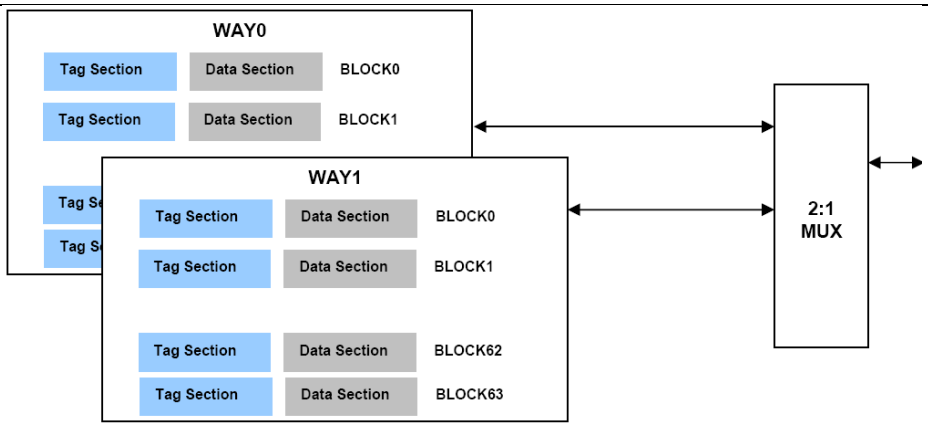

图 9 Blackfin 数据高速缓存的配置 – 双路组关联

图 10说明了 32 位的地址空间是如何映射到高速缓冲存储器空间。当两个数据块都使能为高速缓存时，根据DCBS位的状态不同，地址空间中的位 14 或位 23 用于选择任意一个 16K字节的数据块。

高速缓存的写方法

外部存储器分为不同的页（由数据高速缓冲存储旁视缓冲器—Data Cache Protection Lookaside Buffers – DCPLB 寄存器定义），每一页的属性都可以独立配置。和以下部分的讨论一样，存储器页可配置为：

- 要么在回写模式，或者在直写模式配置。
- 配置为只在读或者读写操作时分配高速缓存行

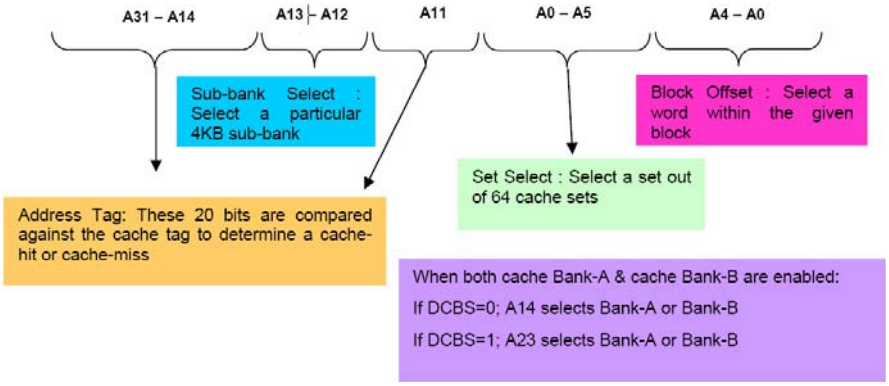

图 10 外部存储器地址空间在数据高速缓冲存储器空间上的映射

存储器管理单元（MMU）

Blackfin 处理器包括一个基于页面的 MMU，该单元在页层面控制存储器范围内的缓存能力和保护属性的管理。MMU 可实现两个 16 入口的内容可寻址的存储器（CAM）块，每个入口称为缓存能力旁视缓冲器（CPLB）描述符。

缓存能力保护监视缓冲器（CPLB）

CPLB 描述符的每个入口定义了给定存储器页面的缓存能力和保护属性。CPLB 入口可分为指令和数据 CPLB，其中 16 个 CPLB（称为 ICPLB）入口用于取指令请求，另外 16 个 CPLB（称为 DCPLB）入口用于数据传输。设置指令存储器控制寄存器（IMEM_CONTROL）和数据存储器控制寄存器（DMEM_CONTROL）中的相应位可以使能 ICPLB 和 DCPLB。每个 CPLB 入口都由一对 32 位数构成。

取指令

ICPLB_ADDR[n] 定义了 CPLB 描述符描述页面的起始地址。

ICPLB_DATA[n] 定义了CPLB描述符描述的页面属性。图 11说明了ICPLB_DATA寄存器中的各个位段及其功能。

数据操作

DCPLB_ADDR[m] 定义了 CPLB 描述符描述页面的起始地址。

DCPLB_DATA[m] 定义了CPLB描述符描述的页面属性。图 12说明了DCPLB_DATA寄存器中的各个位段及其功能。

存储器页面及页面属性

每个 CPLB 入口都与一个有效的存储器页面相对应，页面中每个地址均共享该页面定义的属性。地址描述符 xCPLB_ADDR[n]提供了存储器中页面的基地址，属性描述符字 xCPLB_DATA[n]指定页面的大小和属性。

页面大小

Blackfin 存储器结构支持四种不同的页面大小—1K 字节，4K 字节，1M 字节或 4M 字节。页面必须在页边界对齐，且页边界必须位于页面大小的整数倍处。

可高速缓存/不可高速缓存标志集合

如果一个页面定义为不可高速缓存，则高速缓存会旁路对该页面的访问。对以下情况，存储器页面需定义为可高速缓存：

- 一个映射到存储器地址的 I/O 设备
- 页面中驻留的程序代码很少被调用（用户也许不希望高速缓存该程序代码）

直写/回写标志

只存在于数据存储器，该属性为数据高速缓存定义为回写模式。

页面重写/修改标志位

只存在于数据存储器，该属性只有在回写模式中页面定义为可高速缓存时才有效，在访问页面之前，优先设置该位。

写访问允许标志

数据存储器 CPLB 有两个标志位分别用于管理模式和用户模式下使能/禁止对相应页面的写访问。

用户读访问允许标志位

在用户模式下该属性使能/禁止页面的读操作。

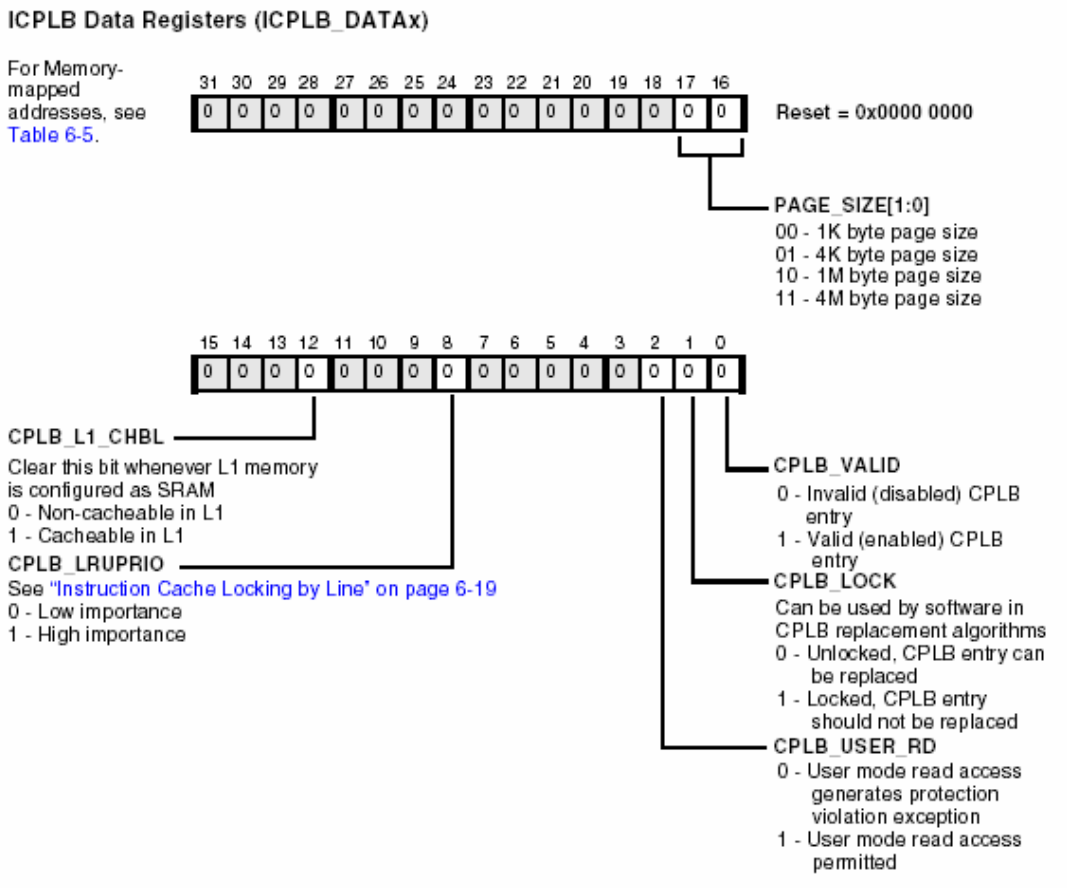

图 11 ICPLB_DATA 寄存器位段及其功能

锁定标志位

锁定 CPLB 入口，该属性对动态存储器管理十分有用。当锁定 CPLB 入口时， CPLB 未命中的异常处理程序不会将它作为替换处理。

LRU 优先权

该属性只在指令存储器 CPLB 上可用，为给定的页面定义 LRU 优先权（高/低），并在修改 LRU 策略中使用。

与“读/写允许”相关的页面属性处理存储器保护，对于在 OS 代码和用户代码间进行分区操作的实时应用，或许需要该功能。用户代码有诸多不同的线程，每个线程又有自己的存储器源，而这些存储器源不能被其他线程访问。但是 OS 内核可以访问所有存储器源，在不同线程中配置不同的 CPLB 配置即可实现该任务。

当使能 CPLB 时，对于将要访问的每个地址，在 CPLB 表中都存在一个有效的 CPLB 入口，如果该引用的地址在表中没有有效的 CPLB 入口，就会产生 CPLB 异常。

页面描述符表

通常，基于存储器的数据结构（称为页面描述符表）用于 CPLB 管理，所有潜在需要的 CPLB 入口都保存在页面描述符表中（通常位于内部 SRAM 中），当需要配置 CPLB 时，应用程序代码就从页面描述符表中取出 CPLB 入口，并填入 CPLB 描述符中。

对于小型/简单存储器模型，定义一组与 32 个 CPLB 入口相符合的 CPLB 描述符集（16 个 ICPLB 和 16 个 DCPLB），这种定义称为*静态储存器管理模型*。实例 1 就使用了这种静态储存器管理模型。

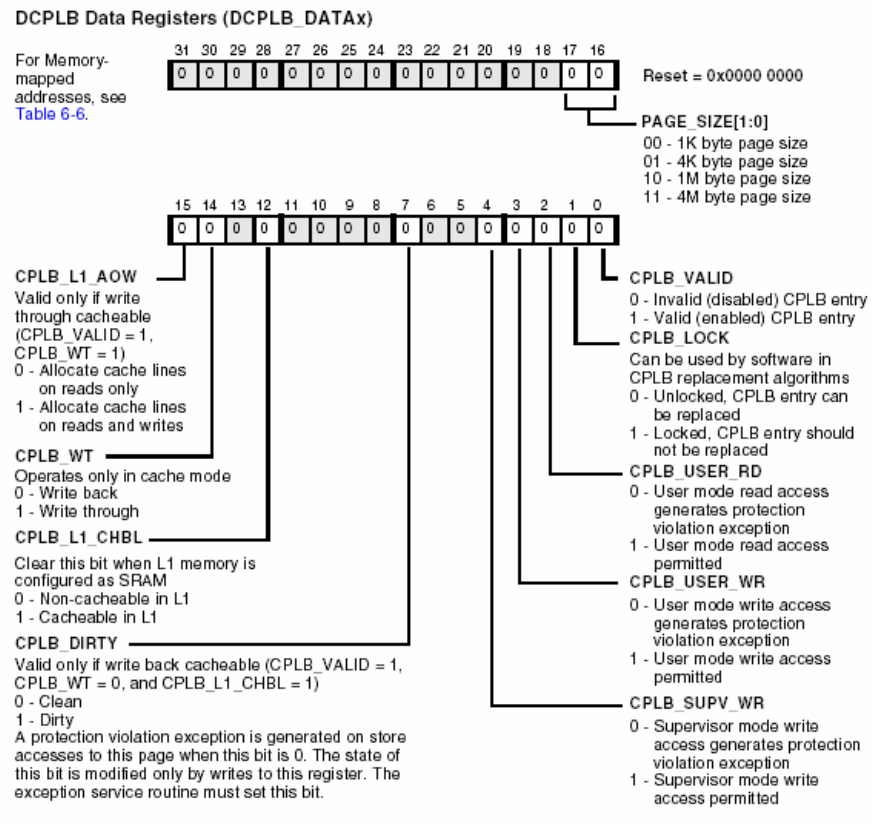

图 12 ICPLB_DATA 寄存器的位段及其功能

对于复杂的存储器模型，页面描述符表可能有 CPLB 入口，与 16 个可用的 CPLB 寄存器不太适合。在这种情况下，CPLB 可首先配置为有任意 16 个入口，当处理器引用了一个 CPLB 中未定义的存储单元时，会产生异常，并且出错存储单元的地址会存入到错误地址寄存器（xCPLB_FAULT_ADDR）中。异常处理程序通过错误地址查找 CPLB 表中所需要的入口，这样存在的CPLB入口就可以用新的CPLB入口代替。

CPLB 替换策略既可以简单，也可以很复杂，由系统要求而定。但也有可能会有多个引用存储器的地址在 CPLB 描述符中无效入口，在这种情况下，异常的优先级和服务顺序如下：

- 指令页面未命中
- DAG0 页面未命中
- DAG1 页面未命中

实例 3 中的程序提供了 DCPLB 未命中的异常处理程序，DCPLB 替换采用了累计补偿型轮循调度方法。

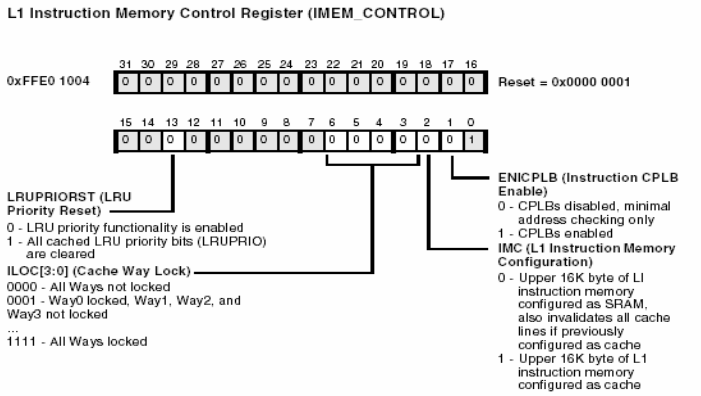

图 13 IMEM_CONTROL 寄存器的位段及其功能

指令和数据高速缓存处理

使能高速缓存

分别适当配置 IMEM_CONTROL 寄存器和 DMEM_CONTROL 寄存器，可以独立使能/禁止指令和数据高速缓存。实例说明如何使能指令/数据高速缓存。

- 在使能高速缓存之前，必须配置和使能有效的 CPLB 描述符。
- 当存储器设置为高速缓存时，不能直接访问（既不能通过处理器内核，也不能通过 DMA）。

指令存储器控制寄存器（IMEM_CONTROL）

图 13说明了IMEM_CONTROL寄存器中各个位段及其功能。

数据存储器控制寄存器（DMEM_CONTROL）

图 14说明了DMEM_CONTROL寄存器中不同的位段及其功能。

使指令高速缓存无效

使指令高速缓存无效的方法有三种：

(a). IFLUSH 指令可用于使存储器映射中的某个特定地址无效。当执行指令 IFLUSH[p2]时，如果 p2 指向的存储器地址已经移入高速缓存，则执行完上述指令后，相应的高速缓存行就会无效。当指令为 IFLUSH[p2++]形式时，指针的增量为高速缓存行的大小（例如 32 位）。

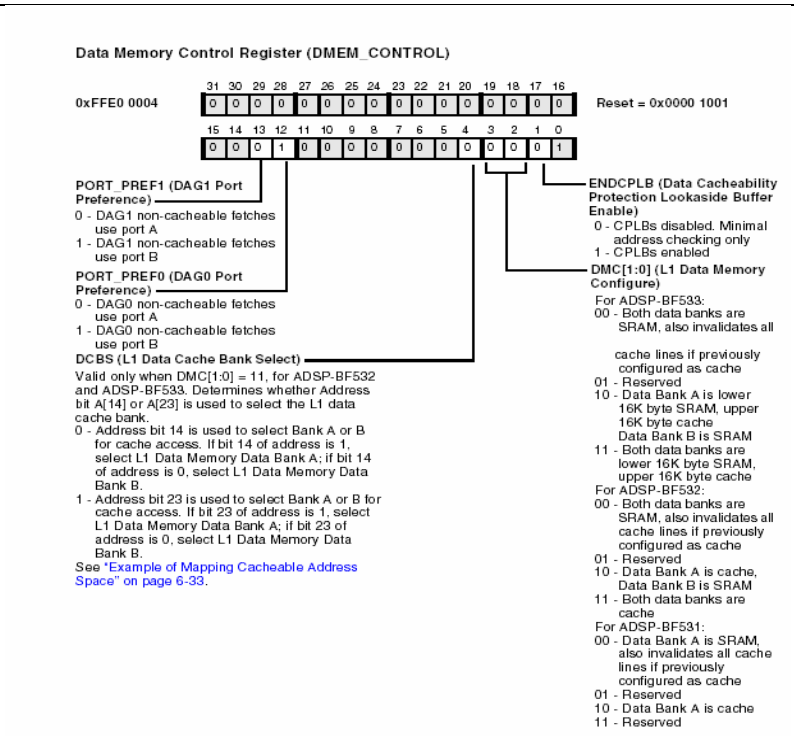

图 14 DMEM_CONTROL 寄存器的位段及其功能

(b). 将高速缓存中任意块的标志部分写 1，可明确清除标志部分的 valid 位。使用 ITEST_COMMAND 寄存器可以向标志部分写入数值。下节将进行详细讨论。

(c). 要使整个高速缓存无效，将 IMEM_CONTROL 寄存器中的 IME 位清零即可，该操作将清除高速缓存中所有标志部分的有效位。对 IMC 位置位又可再次使能高速缓存。

数据高速缓存控制指令

(a). PREFETCH 指令可用于将一个行分配到 L1 高速缓存中。

(b). FLUSH 指令将使数据高速缓存与外部存储器指定的高速缓存行同步。如果缓存的数据行被重写，则指令写完行后，会同时在数据高速缓存中标记该行已被清除。

(c). FLUSHINV 指令会使数据高速缓存执行与 FLUSH 指令相同的功能，并使高速缓存中指定的行无效。如果地址未被重写，则不会产生清除操作。在这种情况下，只会产生一无效步骤。

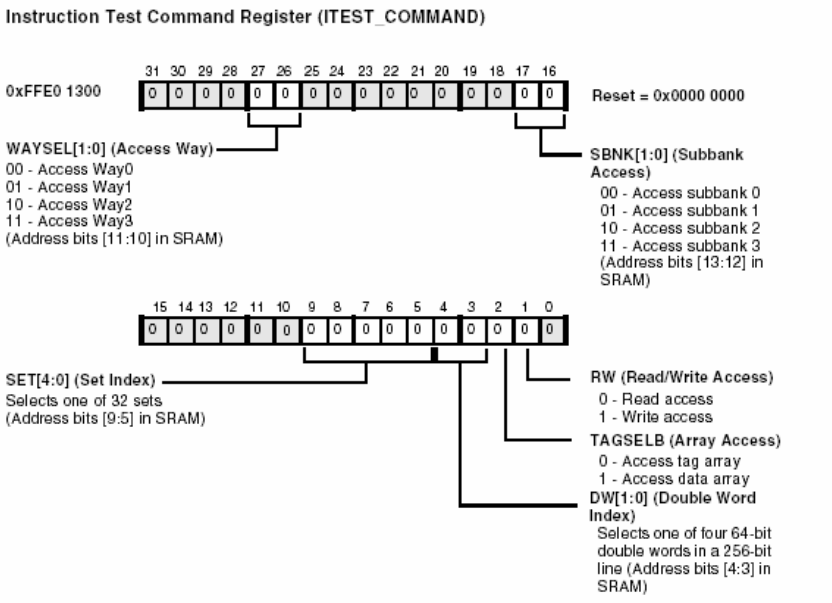

图 15 ITEST_COMMAND 寄存器的位段及其功能

访问高速缓冲存储器

当L1 存储器块配置为高速缓存时，处理器核/DMA就不能直接访问，但用ITEST_COMMAND寄存器和DTEST_CMOMMAND寄存器可以在高速缓存空间上进行读/写操作。DTEST_COMMAND寄存器也可以用于访问指令SRAM块。图 15说明了ITEST_COMMAND寄存器的位段，图 16说明了DTEST_COMMAND寄存器的位段。

访问指令高速缓存

ITEST_COMMAND 寄存器可用于访问指令高速缓存块的数据或者标志部分。

高速缓存块可以分为 4 个 64 位字，可以选择访问其中任意一个字。当读高速缓存时，数据将会读入到 ITEST_DATA[1:0]寄存器组中，当写高速缓存时，ITEST_DATA[1:0]寄存器组的值将写入高速缓存中。

当访问标志部分时，32 位的标志位值将传入或者传出 ITEST_DATA0 寄存器。

例如要将值 0x0C010360 写入 ITEST_COMMAND 寄存器，该指令将从子阵 1 中的块 27 的 way 3 读取标志，并传输给 ITEST_DATA0 寄存器。同样，写入值 0x0C010362，则将 ITEST_DATA0 寄存器中的内容传输给子阵 1 中的块 27 的 way 3 所定义的块标志部分。当访问标志部分（读或者写）时，ITEST_COMMAND 寄存器中的位 3 和位 4 将保留。

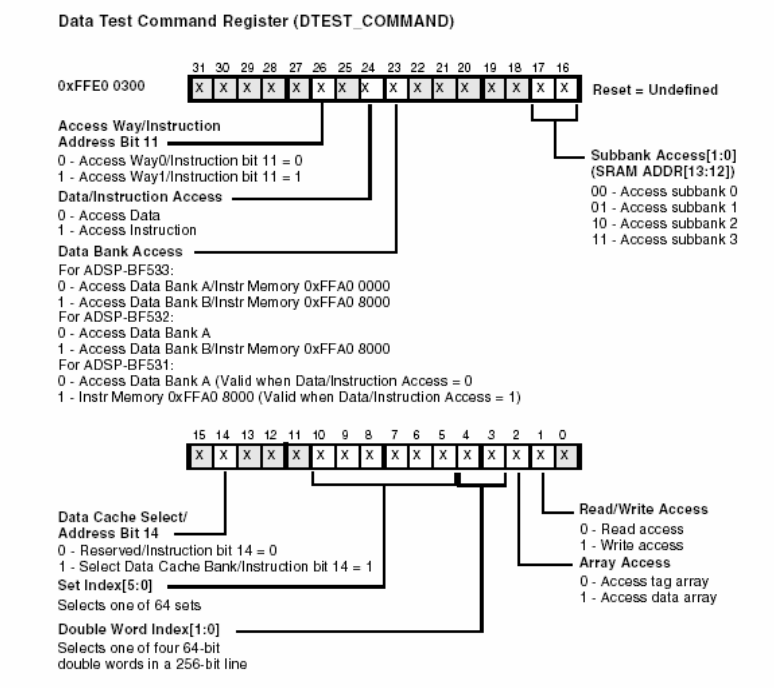

图 16 DTEST__COMMAND 寄存器的位段及其功能

写入值 0x0C010374 将读取子阵 1 中块 27 的 way 3 中高速缓存块的第二个字，并将其传输到 ITEST_DATA[1:0]寄存器组。写入值 0x0C010366 到 ITEST_COMMAND 寄存器，则将 ITEST_DATA[1:0]寄存器组中的值传输到位于指令高速缓存子阵 1 中块 27 的 way 3 的高速缓存块的字 0。在写高速缓存时，必须在写 ITEST_COMMAND 寄存器之前必须载入 ITEST_DATA[1:0]寄存器测试。

访问数据高速缓存

当 DTEST_COMMAND 寄存器中的位 24 清零时，DTEST_COMMAND 寄存器就可用于访问数据高速缓存块中的数据或标志部分。高速缓存块中的数据部分的字可写入 DTEST_DATA[1:0]寄存器组，或从其读出。

当访问标志部分时，32 位标志值可写入 DTEST_DATA0 寄存器或者从其读出。

考虑这样的情况，DTEST_DATA[1:0]寄存器组中的值需要写入数据高速缓存块 A 的子阵 0, 块 39,way1 中的数据块的字 0，向 DTEST_COMMAND 寄存器写入值 0x040004E5 即可，但在写该寄存器前，DTEST_DATA[1:0]寄存器组必须加载值。在访问数据高速缓存空间（bit24=1）时，DTEST_COMMAND 寄存器的位 14 保留未用。

访问指令 SRAM

当 DTEST_COMMAND 寄存器的位 24 置位时，该寄存器可用于访问指令 SRAM。64 位字可以从指令 SRAM 传输给 DTEST_DATA[1:0]寄存器组，或从 DTEST_DATA[1:0]寄存器组传输到指令 SRAM。因此，可同时访问存储器中的 8 个字节。

工作于该模式时，必须将 DTEST_COMMAND 寄存器中的位 2 置位，位 3-10 必须分配为被访问地址的位 3-10。考虑这种情况，假设从指令存储器读取地址 0xFFA07935 中的字节内容，该地址位于子阵 1，访问上述字节时，将访问由地址（0xFFA07930-0xFFA07937）寻址的整个块，因此必须将控制值 0x05034134 加载到 DTEST_COMMAND 寄存器。

VisualDSP++汇编支持

VisualDSP++工具支持高速缓冲存储器管理，以上讨论的某些特性的详细信息可在 *VisualDSP++ Complier Manual for Blackfin Processor* 中找到。指令和数据高速缓存可以单独或者同时使能，其缓冲的存储器空间也可以独立配置。

CPLB 控制

通过全局整数变量__cplb_ctrl 可控制对 CPLB 的支持，该变量的 C 名称前有两个下划线字符，而其汇编名称前有三个。该变量的值确定启动程序代码是否使能了 CPLB 系统，默认情况下，该变量值为 0，表征未使能 CPLB。编译指示 retain_name 应与__cplb_ctrl 一起使用，这样，在使能汇编优化时，编译器就不会删除该变量。

通过以下几种方法可以改变__cplb_ctrl 的值：

- 可以将该变量定义为有初始值的全局变量，这样就可以替换库中的定义。
- 载入应用程序后但在运行之前，可在调试器 debugger 中改变该变量的链接（linked-in）版本，这样，启动程序代码就采用了该变量的不同的值。

CPLB 安装

当__cplb_ctrl 使能 CPLB 时，启动程序代码就调用子程序__cplb_init。该子程序在一个表中建立指令和数据 CPLB，并使能存储器保护硬件。默认的配置表由文件 cplbtab*n*.s 定义，该文件位于 VisualDSP++\Blackfin\lib\src\ libc\crt，其中 n 为 Blackfin 处理器的部分编号。

当使能高速缓存时，将安装上述文件定义的默认 CPLB 配置。然而，用户也可以通过修改上述文件来定义用户自己的 CPLB 配置，但修改的文件必须包含在工程文件夹中，以便更有效的修改。工程“example5”说明了如何修改 CPLB 配置表。

异常处理机制

如以前讨论的一样，在复杂的存储器模型中，可能一次需要激活多个 CPLB。在这样的系统中，当应用程序试图访问激活的 CPLB 没有覆盖的存储器时，最终可能需要一段时间，这将增加 CPLB 未命中异常。

VisualDSP++库包括了用于这种情况的 CPLB 管理子程序，称为_cplb_mgr。当异常处理程序确定出现 CPLB 未命中（无论是数据未命中或指令未命中）时将调用该子程序。_cplb_mgr 可识别需要安装的未激活的 CPLB 来解析访问操作，并用其替换其中一个激活的 CPLB。如果将使能 CPLB，则默认启动程序代码将安装名为_cplb_hdr 的默认异常处理程序，该程序仅用于测试 CPLB 丢失异常，并将异常传递给_cplb_mgr。也期望用户使用自己的异常处理程序来处理这些额外事件。

MMR 概述

以下存储器映射寄存器用于 ADSP-BF533 存储器管理：

IMEM_CONTROL	DMEM_CONTROL
ITEST_COMMAND	DTEST_COMMAND
ITEST_DATA[1:0]	DTEST_DATA[1:0]
ICPLB_DATA[15:0]	DCPLB_DATA[15:0]
ICPLB_ADDR[15:0]	DCPLB_ADDR[15:0]
ICPLB_STATUS	DCPLB_STATUS
ICPLB_FAULT_ADDR	DCPLB_FAULT_ADDR

表 3 CPLB 存储器映射寄存器

Blackfin 衍生产品的高速缓冲存储器配置

以上章节讨论的是 ADSP-BF533 处理器的高速缓冲存储器配置和高速缓冲存储器控制，这些内容同样适用于 ADSP-BF531 和 ADSP-BF532 处理器。本应用文档提供的实例程序代码也可以用于 ADSP-BF531 和 ADSP-BF532 处理器。

ADSP-BF535 处理器中可配置为高速缓存的数据和指令存储器数量与 ADSP-BF533 相同，然而，ADSP- BF561 处理器中可配置为高速缓存的数据和指令存储器数量为 ADSP- BF533 处理器的两倍。ADSP-BFF535 和 ADSP-BF561 处理器的指令/数据高速缓冲存储器配置和控制与 ADSP-BF533 一样，但是某些 MMR 的地址和位段定义有差异，在配置高速缓存时必须予以考虑。

结论

本文讨论了 Blackfin 处理器上的的指令和数据高速缓冲存储器配置，还讨论了高速缓存块的地址映射。应用文档中提供的实例程序代码说明如何安装指令存储器和数据存储器的 CPLB 描述符，如何使能/禁止指令/数据高速缓冲存储器和如何处理 CPLB 异常，以及指令高速缓冲存储器的通道锁定，同时还讨论了处理器核通过 xTEST_COMMAND 访问指令/数据寄存器。

附录

本文档相关的 ZIP 文件包括下列程序代码：

(a) 用于配置 CPLB 描述符和指令/数据高速缓冲存储器的程序代码实例
(b) 用于 CPLB 异常处理的程序代码实例
(c) 用于指令高速缓冲存储器通道锁定的程序代码实例
(d) 说明数据高速缓冲存储器控制指令的程序代码实例
(e) 用于 Blackfin 高速缓冲存储器的 VisualDSP++汇编支持的程序代码实例

参考文献

[1] *ADSP-BF533 Blackfin Processor Hardware Reference*. Rev 3.1, May 2005. Analog Device, Inc.
[2] *Computer Architecture A Quantitative Approach.* Second Edition, 2000 David A. Patterson and John L. Hennessy.

文档记录

Revision	Description
Rev 1 – June 13, 2005 *by Kunal Singh*	Initial Release