ARM926EJ-S™

版本：**r0p5**

技术参考手册

ARM®

ARM926EJ-S

技术参考手册

发行信息

变更历史

日期	发行	机密性	变更
2001-09-26	A	公开	第一次发行
2002-01-29	B	公开	第二次发行
2003-12-05	C	公开	第三次发行。包含 r0p5 变更，瑕疵更正。
2004-01-26	D	公开	第四次发行。包含 r0p4。技术上与之前版本统一。
2008-06-16	E	公开	修改发行的 r0p5 版本，以及其他强化。

专利告示

机密状态

产品状态

本文中的信息是最终版本，也就是成熟产品。

网站地址

http://www.arm.com

目 录

表目录

图目录

前言

本前言介绍了 ARM926EJ-S 版本 r0p5 技术参考手册（Technical Reference Manual，TRM）。它包含以下两个部分：

- 关于本文；
- 问题反馈。

关于本文

本文是关于 ARM926EJ-S 处理器的技术参考手册。

产品版本状态

rnpn 标示符表示本手册中描述的产品版本状态，其中：

rn　　表示该产品的主要版本。

pn　　表示该产品的次要版本或修订情况。

面向的读者

本手册适用于系统设计师、系统综合师以及使用 ARM926EJ-S 处理器设计或规划片上系统（System-on-Chip，SoC）的程序员。

使用本手册

本文由下面的章节组成：

第一章　引言

这一章给出了 ARM926EJ-S 处理器的概述。

第二章　编程模型

这一章描述了编程模型和 ARM926EJ-S 寄存器的细节。

第三章　存储器管理单元

这一章介绍了存储器管理单元（Memory Management Unit，MMU）和地址映射处理，以及怎样使用 CP15 寄存器来使能或禁能 MMU。

第四章　Cache和写缓冲

这一章介绍了指令 cache、数据 cache、写缓冲和物理地址标记 RAM。

第五章　紧耦合存储器接口

这一章介绍了紧耦合存储器接口（Tightly Coupled Memory，TCM）和怎样使用 CP15 区域寄存器来使能和禁能 cache。它包含了一些关于怎样连接不同类型的 RAM 的例子。

第六章　总线接口单元

这一章介绍了到 AMBA 的总线接口单元（Bus Interface Unit，BIU）。

第七章　非高速缓存的指令

这一章介绍了 ARM926EJ-S 处理器是如何推测非高速缓存指令的预取以达到提高性能的目的。

第八章　协处理器接口

这一章介绍了协处理器接口。本章包含了协处理器操作的时序图。

第九章　指令内存栅

这一章介绍了指令存内存栅（Instruction Memory Barrier，IMB）以及 IMB

操作如何被用来确保 ARM926EJ-S 处理器处理的数据和指令流的一致性。

第十章 嵌入式跟踪宏单元的支持

这一章介绍了 ARM926EJ-S 处理器是如何支持嵌入式跟踪宏单元的（Embedded Trace Macrocell，ETM）。

第十一章 调试支持

这一章介绍了调试接口和 EmbeddedICE-RT。

第十二章 电源管理

这一章介绍了由 ARM926EJ-S 处理器提供的电源管理能力。

附录A 信号描述

以官能团的方式介绍了 ARM926EJ-S 处理器的信号。

附录B CP15 测试和调试寄存器

介绍了用作测试和调试的寄存器的细节。

附录C 术语表

介绍了本书中使用的术语定义。

惯例

本手册的使用惯例描述如下：

- 印刷习惯；
- 时序图惯例；
- 信号；
- 编码规则。

印刷习惯

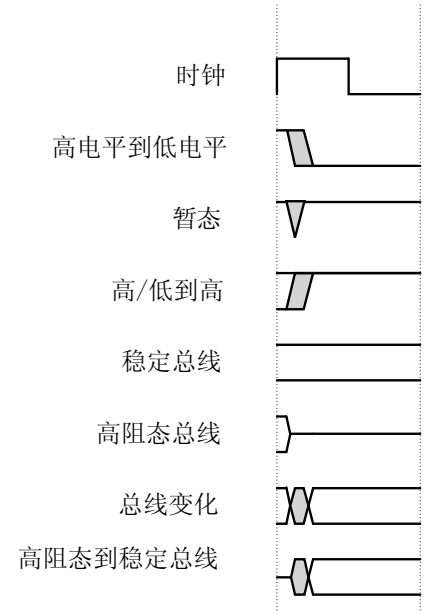
印刷惯例如下：

italic	高亮的重要提示，包括特殊术语、交叉引用的符号以及引文。
bold	高亮的接口元素，比如条目名称。信号名称的符号。也适当的用于描述列表中的条款。
`monospace`	表示键盘输入的文本，比如命令、文件名、程序名以及源代码。
`mono`space	表示一个允许的命令或者选项的缩写。用户可以输入带下划线的文本以替代完整命令或者选项名称。
`monospace italic`	表示键盘输入文本的参数，其中这些参数可以用特定值来替代。
`monospace bold`	表示代码例程中程序语言的关键字。
<and>	在汇编语法中装入的可替代的条目，这些条目可能以代码或代码片段的方式出现。例如： `MRC p15, 0 <Rd>, <CRn>, <CRm>, <Opcode_2>`

时序图惯例

图片时序图惯例解释了时序图中使用的元素。下面的样例给出了时序图中的成分。时序图中发生任何变化都有明显的标记。用户不能假定任何没有在该图中呈现的时序信息。

总线和信号中的阴影部分为未定义，所以总线和信号在阴影区域时间段内可以是任何值。这时真实值并不重要同时也不影响正常操作。



序图 1　时序图惯例

时序图惯例中表示的单一比特（single-bit）信号在某些时候表现为高电平而同时又是低电平，而这与总线状态改变比较类似。如果一个单一比特信号像这样表示出来，那么它的值并不影响附随的描述。

信号

信号惯例如下：

Signal level	一个断言信号的电平取决于该信号是高电平有效还是低电平信号。断言的意思是： 对高电平有效信号为 HIGH； 对低电平有效信号为 LOW。
Lower-case n	在信号名称开头或结尾的小写 n 表示一个低电平有效信号。
Prefix A	前缀 A 表示全局的高级可扩展接口（Advanced eXtensible Interface，AXI）信号。
Prefix AR	前缀 AR 表示 AXI 读地址通道信号。
Prefix AW	前缀 AW 表示 AXI 写地址通道信号。
Prefix B	前缀 B 表示写响应通道信号。
Prefix C	前缀 C 表示 AXI 低功耗接口信号。
Prefix H	前缀 H 表示高级高性能总线（Advanced High-performance Bus，AHB）信号。
Prefix P	前缀 P 表示高级外设总线（Advanced Peripheral Bus，APB）信号。
Prefix R	前缀 R 表示 AXI 读数据通道信号。
Prefix W	前缀 W 表示 AXI 写数据通道信号。

编码规则

编码惯例如下：

```
<size in bits>'<base><number>
```

这是一个 Verilog 方式的数字常量缩写。例如：

- ‘h7B4 是一个无尺寸的十六进制数；
- ‘o7654 是一个无尺寸的八进制数；
- 8’d9 是一个 8 位宽的十进制数，值为 9；
- 8’h3F 是一个 8 位宽的十六进制数，值为 0x3F。这等同于 b0011 1111；
- 8’b1111 是一个 8 位宽的二进制数，值为 b0000 1111。

推荐阅读

这部分列出了由 ARM 和第三方的出版物。

ARM对这些文档提供更新和错误纠正。参考http://www.arm.com以获取当前刊误手册、附录以及常见问答清单。

ARM 出版物

本手册中包含的信息指定为 ARM926EJ-S 处理器的缩写器件刊名。可参考下面的文档以获取其他相关信息：

- ARM Architecture Reference Manual (ARM DDI 0100)
- ARM AMBA Specification (Rev 2.0) (ARM IHI 0001)
- ARM926EJ-S Implementation Guide (ARM DII 0015)
- ARM926EJ-S Test Chip Implementation Guide (ARM DXI 0131)
- ARM9EJ-S Technical Reference Manual (ARM DDI 0222)
- Multi-layer AHB Overview (ARM DVI 0045)
- ETM9 Technical Reference Manual (ARM DDI 0157).

问题反馈

ARM 有限公司欢迎关于 ARM926EJ-S 处理器和它的文档的反馈。

关于本产品的反馈

如果用户有任何关于本产品的评论或意见，联系用户的供应商并给出：

- 产品名称；
- 简明的解释。

关于本手册的反馈

如果用户对本手册有任何评论，请发邮件给errata@arm.com，并给出下列信息：

- 文档标题；
- 文档号；
- 你的评论中引用的页码；
- 对你评论的简单解释。

ARM 同时也欢迎常规的建议和改进方法。

第1章 引言

本章介绍了 ARM926EJ-S 处理器和它的特征。包含以下内容：

- 关于ARM926EJ-S处理器。

1.1 关于ARM926EJ-S处理器

ARM926EJ-S 处理器是通用微处理器 ARM9 家族中的一员。ARM926EJ-S 处理器面向有完整的存储器管理、高性能、低晶粒尺寸以及低功耗等重要要求的多任务处理应用。

ARM926EJ-S 处理器支持 32 位的 ARM 和 16 位的 Thumb 指令集，让用户能够在高性能和高代码密度之间自由权衡。ARM926EJ-S 处理器包括高效执行 Java 字节代码、提供和 JIT 相比拟的 Java 性能而又没有附加的代码负荷的特征。

ARM926EJ-S 处理器支持 ARM 调试体系并包括支持硬件和软件调试的逻辑。ARM926EJ-S 处理器拥有一条带高速缓存（cached）的哈佛总线体系并提供一个完整的高性能处理器子系统，包括：

- 一个 ARM926EJ-S 定点数（integer）内核；
- 一个存储器管理单元（MMU）；
- 单独的指令和数据 AMBA AHB 总线接口；
- 单独的指令和数据 TCM 接口。

ARM926EJ-S 处理器为外部协处理器提供支持，以便能够处理浮点数或增加其他特定应用的硬件加速（模块）。ARM926EJ-S 处理器的 ARM 体系结构实现是 v5TEJ。

ARM926EJ-S 处理器是一个可综合的宏单元。这意味着用户可以针对特定的目标库进行优化，并且用户可以配置存储系统以适合用户的目标应用。用户可以在 4KB 到 128KB 之间以任何 2 的次幂单独配置两个 cache 大小。

紧耦合的指令和数据内存在外部实例化为 ARM926EJ-S 的宏单元，给用户提供了存储子系统在性能、功耗和特定 RAM 类型上优化的灵活性。TCM 接口使得非零等待状态存储器能够被连接，除此之外还提供了支持 DMA 的机制。

图 1.1表示了ARM926EJ-S处理器的主要模块。

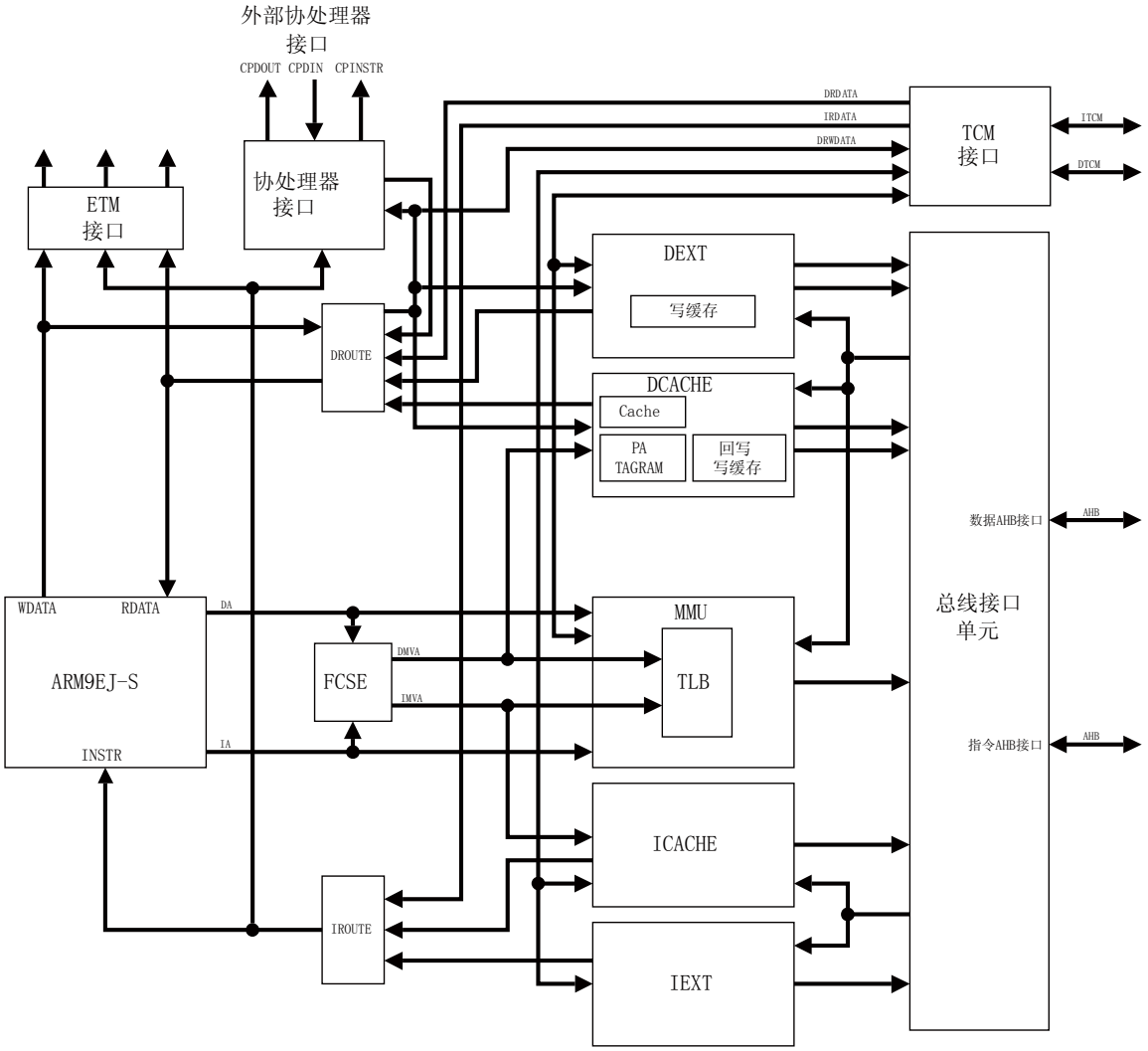


图 1.1　ARM926EJ-S 框图

图 1.2和图 1.3表示了ARM926EJ-S的接口。

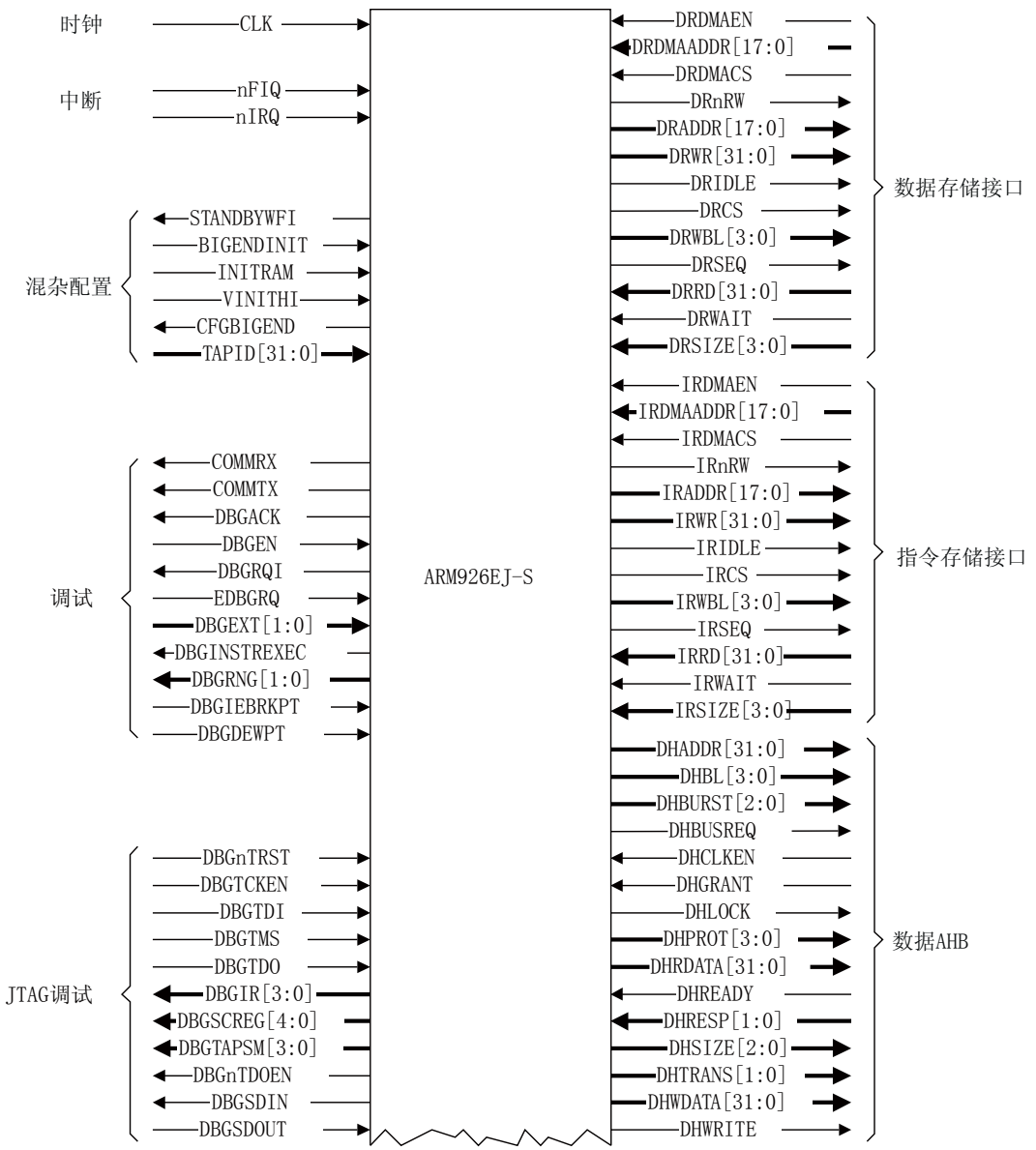


图 1.2　ARM926EJ-S 接口框图（第一部分）

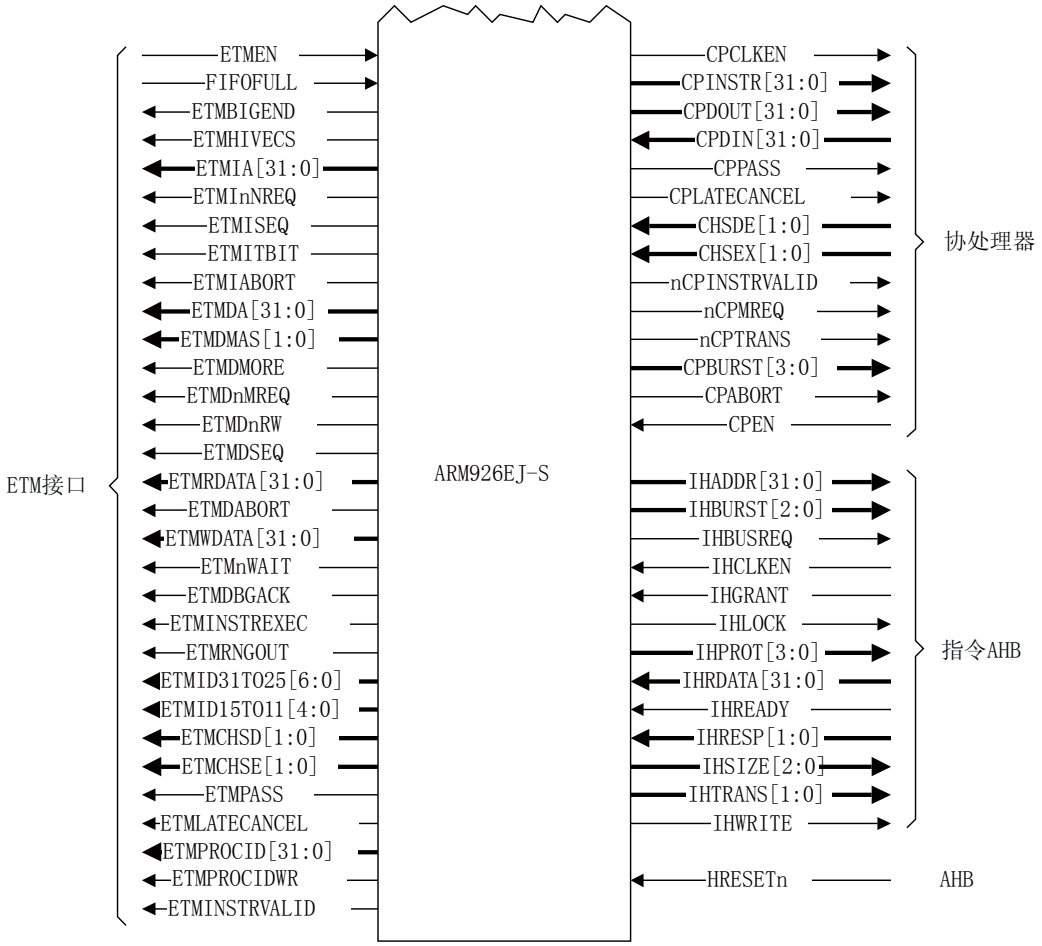


图 1.3　ARM926EJ-S 接口框图（第二部分）

第2章 编程模型

本章描述了 ARM926EJ-S 在系统控制协处理器 CP15 中的寄存器，并提供了编程微处理器的信息。包含如下内容：

- 关于编程模型；
- ARM926EJ-S系统控制协处理器CP15 寄存器汇总；
- 寄存器描述。

2.1 关于编程模型

系统控制协处理器（CP15）被用来配置以及控制 ARM926EJ-S 处理器。Cache、紧耦合存储器（TCM）、存储器管理单元（MMU）、以及大部分其他系统选项都通过使用 CP15 的寄存器来控制。用户只能在特权模式下通过 MRC 和 MCR 指令来访问 CP15 的寄存器。使用 CDP、LDC、STC、MCRR 和 MRRC 指令、以及非特权模式下的 MRC 或 MCR 指令访问 CP15 会引起未定义指令异常发生。

2.2 ARM926EJ-S系统控制协处理器CP15 寄存器汇总

CP15 定义了 16 个寄存器。表 2.1表示了这些寄存器的读写功能。

表 2.1 CP15 寄存器汇总

寄存器	读	写
0	ID 编码[1]	不可预测
0	Cache 类型[1]	不可预测
0	TCM 状态[1]	不可预测
1	控制	控制
2	转换表基址	转换表基址
3	域访问控制	域访问控制
4	保留	保留
5	数据错误状态[1]	数据错误状态[1]
5	指令错误状态[1]	指令错误状态[1]
6	错误地址	错误地址
7	Cache 操作	Cache 操作
8	不可预测	TLB 操作
9	Cache 关闭[2]	Cache 关闭
9	TCM 区域	TCM 区域
10	TLB 关闭	TLB 关闭
11 和 12	保留	保留
13	FCSE PID[1]	FCSE PID[1]
13	上下文 ID[1]	上下文 ID[1]
14	保留	保留
15	测试配置	测试配置

[1]、寄存器地址 0、5 和 13 都提供超过一个寄存器的访问。实际访问的寄存器取决于 Opcode_2 域的值。

[2]、寄存器地址 9 提供超过一个寄存器的访问。实际访问的寄存器取决于CRm域的值。参看寄存器描述以获取更多细节。

所有的 CP15 寄存器的比特定义和包含的状态在复位时都被设置为 0，以下情况除外：

- 如果 VINITHI 信号在复位时为低，则位 V 被设置为 0，如果 VINITHI 信号为高则设置为 1；
- 如果 BIGENDINIT 信号在复位时为低，则位 B 被设置为 0，如果 BIGENDINIT 信号为高则设置为 1；
- 如果 INITRAM 引脚在复位时为高则指令 TCM 被使能。这允许从指令 TCM 的启动并设置 ITCM 区域寄存器中的位 ITCM 为 1。

2.2.1 ARM926EJ-S系统的地址

在ARM926EJ-S系统中存在三种完全不同的地址类型。表 2.2表示了ARM926EJ-S处理器中的地址类型。

表 2.2　ARM926EJ-S 中的地址类型

域	ARM9EJ-S	Cache 和 MMU	TCM 和 AMBA 总线
地址类型	虚拟地址（VA）	修改过的虚拟地址（MVA）	物理地址（PA）

下面是当 ARM9EJ-S 内核请求一条指令时地址操作的一个例子：

1. ARM9EJ-S 内核发出指令的虚拟地址（Virtual Address）；
2. 使用FCSE PID的值将虚拟地址转换为修改过的虚拟地址（Modified Virtual Address）。指令Cache（ICache）和存储器管理单元（MMU）检测MVA，参考处理器ID寄存器c13；
3. 如果由 MMU 对于 MVA 执行的保护检查没有中止并且 MVA 标签在 ICache 中，则指令数据被送回 ARM9EJ-S 内核；
4. 如果由 MMU 对于 MVA 执行的保护检查没有中止，并且由于 MVA 标签并不存在 cache 中而导致 cache 未命中，那么 MMU 将 MVA 转换为物理地址 PA。该地址被传送给 AMBA 总线接口以执行一个外部访问。

2.2.2 访问CP15 寄存器

用户仅能在特权模式下通过MRC和MCR指令来访问CP15 寄存器。MCR和MRC指令的指令位样式如图 2.1所示。

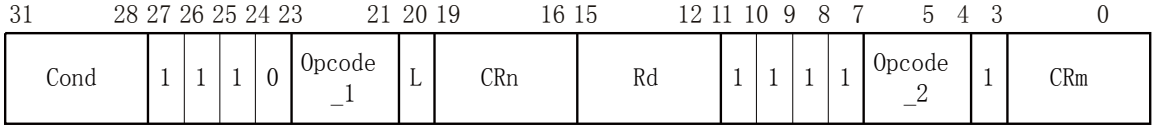


图 2.1　MRC 和 MCR 位样式

这些指令的记忆方法如下：

```
MCR{cond}   p15,<Opcode_1>,<Rd>,<CRn>,<CRm>,<Opcode_2>
MRC{cond}   p15,<Opcode_1>,<Rd>,<CRn>,<CRm>,<Opcode_2>
```

尝试读一个只写寄存器，或写一个只读寄存器将引起不可预测的结果。在所有访问 CP15 的指令中：

- Opcode_1 域应该为 0，除非指定的值被用来选择规定的操作。使用其他值将导致不可预测的行为；
- Opcode_2 和 CRm 域应该为 0，除非指定的值被用来选择规定的操作。使用其他值将导致不可预测的行为。

表 2.3表示了本章中使用的术语和缩写。

表 2.3 CP15 缩写词

术语	缩写	描述
不可预测（Unpredictable）	UNP	对于读：当读该地址时返回的数据是不可预测的。可以是任何值。 对于写：写该地址将导致不可预测的行为，或者器件配置中一个不可预测的改变。
未定义（Undefined）	UND	以会导致未定义指令异常的方式访问 CP15 的指令。
应该为 0（Should Be Zero）	SBZ	当写该地址，这个区域的所有比特都应该为 0。
应该为 1（Should Be One）	SBO	当写该地址，这个区域的所有比特都应该为 1。
应该为 0 或保持（Should Be Zero or Preserved）	SBZP	当写该地址，这个区域的所有比特都应该为 0，或通过将之前从该区域读到的值写入相同区域来维持原状。

在所有情况下，读或写任何数据值到任何 CP15 的寄存器中，包括指定为不可预测、应该为 1 或应该为 0 的这些区域并不引起任何芯片的物理损坏。

2.3 寄存器描述

本节描述了以下寄存器：

- ID编码、Cache类型和TCM状态寄存器c0；
- 控制寄存器c1；
- 转换表基址寄存器c2；
- 域访问控制寄存器c3；
- 寄存器c4；
- 错误状态寄存器c5；
- 错误地址寄存器c6；
- Cache操作寄存器c7；
- TLB操作寄存器c8；
- Cache锁定和TCM区域寄存器c9；
- TLB锁定寄存器c10；
- 寄存器c11 和c12；
- 处理器ID寄存器c13；
- 寄存器c14；
- 测试和调试寄存器c15；

2.3.1 ID编码、Cache类型和TCM状态寄存器c0

寄存器 c0 访问 ID 寄存器、Cache 类型寄存器和 TCM 状态寄存器。读该寄存器返回设备 ID、cache 类型或 TCM 状态。这取决于使用的 Opcode_2 的值：

Opcode_2 = 0　　ID 值；

Opcode_2 = 1　　指令和数据 cache 类型；

Opcode_2 = 2　　TCM 状态。

当读取这些寄存器时CRm域应该为 0。表 2.4表示了用户可以用来读取c0 的指令。

表 2.4　读寄存器 c0

功能	指令	
读 ID 编码	MRC	p15,0,<Rd>,c0,c0,{0, 3-7}
读 cache 类型	MRC	p15,0,<Rd>,c0,c0,1
读 TCM 状态	MRC	p15,0,<Rd>,c0,c0,2

写寄存器 c0 将产生不可预测的结果。

ID编码寄存器c0

这是一个只读寄存器，返回一个 32 位的器件 ID 编码。

用户可以通过读 CP15 寄存器 c0 来访问 ID 编码寄存器，其中 Opcode_2 域需要设置为除了 1 和 2 之外的其他值。例如：

```
MRC     p15, 0, <Rd>, c0, c0, {0, 3-7}     ;returns ID
```

ID编码寄存器的内容见表 2.5。

表 2.5　寄存器 0，ID 编码

寄存器位	功能	值
[31：24]	制造者商标的 ASCII 编码	0x41
[23：20]	变种	0x0
[19：16]	体系（ARMv5TEJ）	0x06
[15：4]	部件号	0x926
[3：0]	版本	0x05[1]

[1]、版本的值可以再范围 0x0 到 0x5 之间，取决于用户使用的版图版本。

Cache类型寄存器c0

这个只读寄存器包含了指令 Cache（ICache）和数据 Cache（DCache）的大小和体系信息，让操作系统能确定怎样执行诸如 cache 清楚和锁定的操作。

用户可以通过读 CP15 寄存器 0 来访问 cache 类型寄存器，其中 Opcode_2 域需要设置为 1。例如：

```
MRC     p15, 0, <Rd>, c0, c0, 1         ; returns cache details
```

Cache类型寄存器的格式见图 2.2。

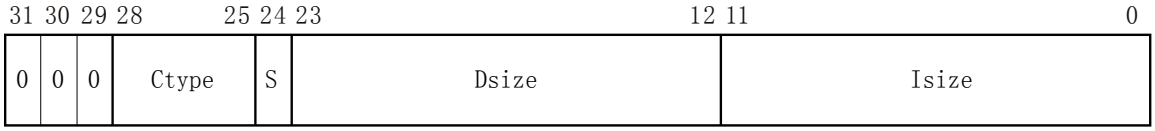


图 2.2　Cache 类型寄存器格式

Ctype　Ctype决定了cache的类型。见表 2.6。

S 位　S=0，指定 cache 为统一的 cache；或 S=1，ICache 和 DCache 是独立的。如果 S=0，Isize 和 Dsize 域均描述了统一 cache 并且两者必须是相同的。在 ARM926EJ-S 处理器中，该位被设置为 1 来指示独立的 cache。

Dsize　指定 DCache 或统一 cache（S 位为 0 时）的大小、行长度和相连性。

Isize 指定 ICache 或统一 cache（S 位为 0 时）的大小、长度和相连性。

Ctype域指定了cache是否支持锁定，以及cache如何清理。其编码见表 2.6。所有未用的值均保留。

表 2.6 Ctype 编码

值	方法	Cache 清理	Cache 锁定
b1110	写回（Write-back）	寄存器 7 操作	格式 C[1]

[1]、参考Cache锁定寄存器c9以获取更多关于cache锁定格式C的细节。

Cache类型寄存器中Dsize和Isize域格式相同。见图 2.3。

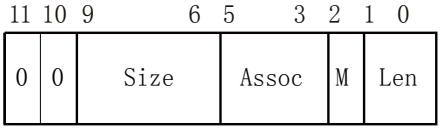


图 2.3 Dsize 和 Isize 域格式

Size Size 域和 M 位一起确定了 cache 的大小。

Assoc Assoc 域和 M 位一起确定了 cache 的相连性。

M 位 乘数（M）位和 Size 及 Assoc 域一起确定了 Cache 大小和相连性。如果 cache 存在，M 位必须被设为 0。如果 cache 不存在，M 位必须设为 1.对于 ARM926EJ-S 处理器，M 位总是设为 0。

Len Len 域确定了 cache 的行长度。

cache的大小由Size域和M位来确定。对于DCache和ICache，M位为 0。DCache的Size域是位[21：18]，ICache则是位[9：6]。每个cache的最小大小是 4KB，而最大大小是 128KB。表 2.7 表示了cache大小的编码。

表 2.7 Cache 大小的编码（M=0）

Size 域	Cache 大小
b0011	4KB
b0100	8KB
b0101	16KB
b0110	32KB
b0111	64KB
b1000	128KB

cache的相连性由Assoc和M位决定。对DCache和ICache，M位为 0。DCache的Assoc域是位[17：15]，ICache则是位[5：3]。表 2.8表示了cache相连性的编码。

表 2.8 Cache 相连性的编码（M=0）

Assoc 域	相连性
b010	4 路
其他值	保留

cache的行长度由Len域决定。DCache的Len域是位[13：12]，ICache则是位[1：0]。表 2.9表示了行长度的编码。

表 2.9　行长度的编码

Len 域	Cache 行长度
b10	8 字（32 字节）
其他值	保留

表 2.10中的配置给出了一个ARM926EJ-S处理器cache类型寄存器的值（的例子）：

- 独立的指令和数据 cache；
- DCache 大小=8KB，ICache 大小=16KB；
- 相连性=4 路；
- 行长度=8 字；
- cache 使用写回，寄存器 7 用于 cache 清理，而格式 C 用于 cache 锁定。

参考Cache锁定寄存器c9以获取关于cache锁定格式C的细节。

表 2.10　Cache 类型寄存器格式的样例

功能		寄存器位	值
保留		[31：29]	b000
Ctype		[28：25]	b1110
S		[24]	b1=哈佛 cache
Dsize	保留	[23：22]	b00
	Size	[21：18]	b0100=8KB
	Assoc	[17：15]	b010=4 路
	M	[14]	b0
	Len	[13：12]	b10=8 字每行（32 字节）
Isize	保留	[11：10]	b00
	Size	[9：6]	b0101=16KB
	Assoc	[5：3]	b010=4 路
	M	[2]	b0
	Len	[1：0]	b10=8 字每行（32 字节）

TCM状态寄存器c0

这个只读寄存器让操作系统能够确认TCM存储器是否存在。详情参看TCM区域寄存器c9。

用户可以通过读 CP15 寄存器 0 访问 TCM 状态寄存器，其中 Opcode_2 域需要设置为 2。例如：

```
MRC     p15,0,<Rd>,c0,c0,2              ;returns TCM details
```

TCM状态寄存器的格式见图 2.4。

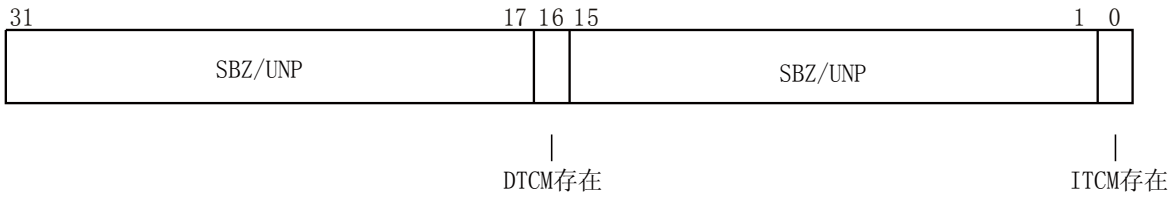


图 2.4　TCM 状态寄存器格式

2.3.2 控制寄存器c1

寄存器 c1 是 ARM926EJ-S 处理器的控制寄存器。该寄存器指定了用来使能或禁能 cache 和 MMU 的配置。建议用户使用读-修改-写的顺序来访问该寄存器。

对于读和写而言，CRm 和 Opcode_2 域都应该为 0。使用下面的指令来读写该寄存器：

```
MRC     p15, 0, <Rd>, c1, c0, 0         ;read control register
MCR     p15, 0, <Rd>, c1, c0, 0         ;write control register
```

除了 V 位和 B 位之外在复位时所有定义的控制位都设为 0。如果 VINITHI 信号为低，V 位在复位时被设为 0，如果 VINITHI 信号为高则设置为 1。如果 BIGENDINIT 信号为低，B 位在复位时被设为 0，如果 BIGENDINIT 信号为高则设置为 1。

图 2.5表示了控制寄存器的格式。

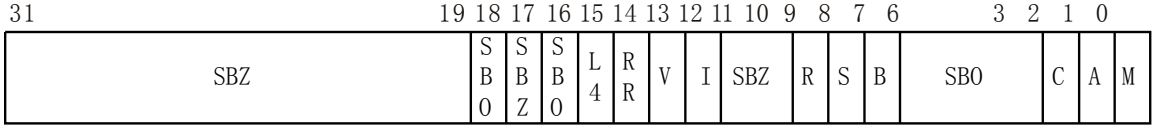


图 2.5　控制寄存器格式

表 2.11描述了控制寄存器位的功能。

表 2.11　控制位功能寄存器 c1

位	名称	功能
[31：19]	-	保留。读该区域返回一个不可预测的值。写入该区域的值应该为 0，或者是从处理器上位[31：19]中读回的相同的值。当修改该寄存器时使用读-修改-写的顺序将提供最好的兼容性。
[18]	-	保留，SBO。读=1，写=1。
[17]	-	保留，SBZ。读=0，写=0。
[16]	-	保留，SBO。读=1，写=1。
[15]	L4 位	当加载指令改变了 PC：0=加载到 PC 设置 T 位；1=加载到 PC 不设置 T 位时确定是否设置 T 位（ARMv4 的行为）。更多细节请参考《ARM 体系结构参考手册》。
[14]	RR 位	ICache 和 Dcache 的替换策略：0=随机替换，1=循环替换
[13]	V 位	异常向量表的位置： 0=选择普通异常向量表，地址范围=0x0000 0000 到 0x0000 001C； 1=选择高地址异常向量表，地址范围=0xFFFF 00001 到 0xFFFF 001C。在复位时设置 VINITHI 的值。
[12]	I 位	Icache 使能/禁能：0=Icache 禁能，1=Icache 使能。
[11：10]	-	SBZ

续上表

位	名称	功能
[9]	R 位	ROM 保护。 该位可以修改ROM保护系统。参考域访问控制寄存器c3。
[8]	S 位	系统保护。 该位可以修改MMU保护系统。参考域访问控制寄存器c3。
[7]	B 位	端结构：0=小端操作，1=大端操作。在复位时设置 BIGENDINIT 的值。
[6：3]	-	保留。SBO。
[2]	C 位	Dcache 使能/禁能：0=Cache 禁能，1=Cache 使能。
[1]	A 位	对齐错误使能/禁能：0=数据地址对齐错误检查禁能，1=数据地址对齐错误检查使能。
[0]	M 位	MMU 使能/禁能：0=禁能，1=使能。

控制寄存器对cache的影响

控制寄存器中直接影响 ICache 和 DCache 行为的位如下：

- M 位；
- C 位；
- I 位；
- RR 位。

假定TCM区域被禁能，则cache行为见表 2.12。

表 2.12　控制寄存器对 cache 的影响

Cache	MMU	行为
ICache 禁能	使能或禁能	所有指令读取都来自外部存储器（AHB）。
ICache 使能	禁能	所有指令读取都是高速缓存的，没有保护检查。所有地址都是平面映射的。也就是 VA=MVA=PA（虚拟地址=映射虚拟地址=物理地址）。
ICache 使能	使能	指令读取是高速缓存或非缓存的，并且执行保护检查。所有地址都从 VA 重映射到 PA，取决于 MMU 页表条目。也就是，VA 转换为 MVA，MVA 重映射到 PA。
DCache 禁能	使能或禁能	所有数据访问都到外部存储器（AHB）
DCache 使能	禁能	所有数据访问均非高速缓存且未缓冲。所有地址都是平面映射的。也就是 VA=MVA=PA。
DCache 使能	使能	所有数据访问是高速缓存或非缓存的，并且执行保护检查。所有地址都从 VA 重映射到 PA，取决于 MMU 页表条目。也就是 VA 转换为 MVA，MVA 重映射到 PA。

如果 DCache 或 ICache 之一禁能，那么该 cache 的内容将不再访问。如果 cache 随后被重新使能，则 cache 的内容没有改变。为了保证内存一致性，DCache 必须在禁能前清理脏的数据（数据被修改过）。

控制寄存器对TCM接口的影响

控制寄存器中的M位，与各个TCM区域寄存器c9 中的En位一起，直接影响TCM接口的行为，如表 2.13所示。

表 2.13　控制寄存器对 TCM 接口的影响

TCM	MMU	Cache	行为
指令 TCM 禁能	禁能	ICache 禁能	所有指令读取都来自外部存储器（AHB）。
指令 TCM 使能	禁能	ICache 禁能	所有指令读取都来自 TCM 接口，或来自外部存储器（AHB），取决于指令 TCM 区域寄存器中设置的基地址。不执行保护检查。所有地址都平面映射，也就是 VA=MVA=PA。
指令 TCM 使能	禁能	ICache 使能	所有指令读取来自 TCM 接口，或来自 ICache，取决于指令 TCM 区域寄存器中设置的基地址。不执行保护检查。所有地址都平面映射，也就是 VA=MVA=PA。
指令 TCM 使能	使能	ICache 使能	所有指令读取都来自 TCM 接口，或来之 ICache/AHB 接口，取决于指令 TCM 区域寄存器中设置的基地址。执行保护检查。所有地址都从 VA 重映射到 PA，取决于页条目。也就是 VA 转换为 MVA，而 MVA 被重映射到 PA。
数据 TCM 禁能	禁能	DCache 禁能	所有数据访问都到外部存储器（AHB）。
数据 TCM 使能	禁能	DCache 禁能	所有数据访问都到 TCM 接口，或到外部存储器，取决于数据 TCM 区域寄存器中设置的基地址。不执行保护检查。所有地址都平面映射。也就是 VA=MVA=PA。
数据 TCM 使能	禁能	DCache 使能	所有数据访问都到 TCM 接口或外部存储器，取决于数据 TCM 区域寄存器中设置的基地址。所有地址都平面映射。也就是 VA=MVA=PA。
数据 TCM 使能	使能	DCache 使能	所有数据访问来自 TCM 接口，或者来自 DCache/AHB 接口，取决于数据 TCM 区域寄存器中设置的基地址。执行保护检查。所有地址都从 VA 重映射到 PA，取决于页条目。也就是 VA 转换为 MVA，而 MVA 被重映射到 PA。

注：当 ARM926EJ-S 处理器存储器访问中止时并不阻止对 TCM 接口的读访问。所有 TCM 接口上的读操作必须被视为推测性的。ARM926EJ-S 处理器的写访问中止并不发生在 TCM 接口上。

2.3.3 转换表基址寄存器c2

寄存器 c2 是转换表基址寄存器（Translation Table Base Register，TTBR），用于一级转换表的基地址。

读 c2 返回的位[31：14]为指向当前有效一级转换表的指针，位[13：0]中的值不可预测。

写寄存器 c2 将以写入值中位[31：14]的值更新指向一级转换表的指针。位[13：0]中的值应该为 0。

用户可以使用下面的指令来访问 TTBR：

```
MRC     p15, 0, <Rd>, c2, c0, 0         ;read TTBR
MCR     p15, 0, <Rd>, c2, c0, 0         ;write TTBR
```

当写 c2 寄存器时 CRm 和 Opcode_2 域应该为 0。

图 2.6表示了转换表基址寄存器的格式。

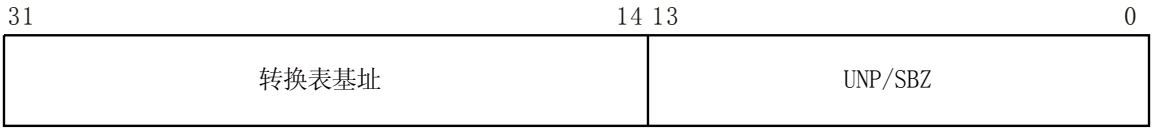


图 2.6　TTBR 格式

2.3.4 域访问控制寄存器c3

寄存器c3 是域访问控制寄存器，由 16 个两位组组成，见图 2.7。

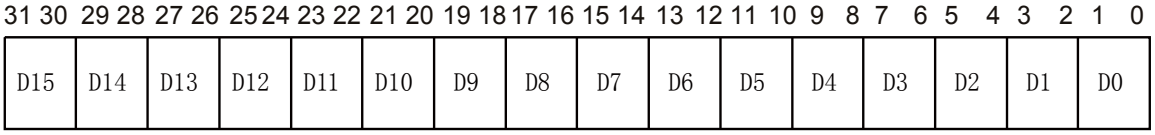


图 2.7　寄存器 c3 格式

每个两位组定义了 16 个域（D15 – D0）之一的访问许可。见表 2.14。

读寄存器 c3 返回域访问控制寄存器的值。

写寄存器 c3 将值写入域访问控制寄存器。

表 2.14　域访问控制定义

值	含义	描述
00	无访问	任何访问都产生一个域错误。
01	客户	对比节或页描述符中的访问许可位进行访问检查。
10	保留	保留。当前行为与无访问模式类似。
11	管理员	访问并不对比访问许可位做检查，因此不会产生许可错误。

用户可以使用下面的指令来访问域访问控制寄存器：

```
MRC     p15, 0, <Rd>, c3, c0, 0         ;read domain access permissions
MCR     p15, 0, <Rd>, c3, c0, 0         ;write domain access permissions
```

2.3.5 寄存器c4

访问、读或写该寄存器将导致不可预测的行为。

2.3.6 错误状态寄存器c5

寄存器 c5 访问错误状态寄存器（Fault Status Register，FSRs）。FSRs 包含上一次指令或数据错误的源。指令方面的 FSR 仅用作调试目的。FSR 在 MMU 禁能时因对齐错误和外部中止而更新。

FSR 的访问由 Opcode_2 域的值决定：

Opcode_2 = 0　　数据错误状态寄存器（DFSR）；

Opcode_2 = 1　　指令错误状态寄存器（IFSR）；

错误类型编码在表 3.9中列出。

用户可以使用下面的指令访问 FSRs：

```
MRC     p15, 0, <Rd>, c5, c0, 0             ;read DFSR
MCR     p15, 0, <Rd>, c5, c0, 0             ;write DFSR
```

```
MRC     p15, 0, <Rd>, c5, c0, 1          ;read IFSR
MCR     p15, 0, <Rd>, c5, c0, 1          ;write IFSR
```

错误状态寄存器的格式见图 2.8。

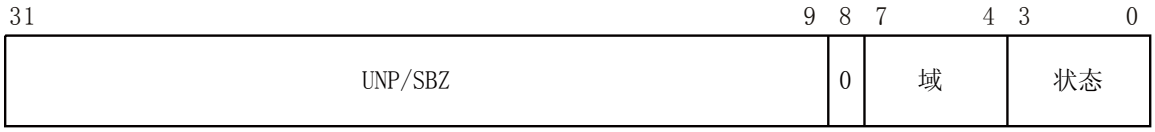


图 2.8　FSR 格式

表 2.15表示了FSR的位域描述。

表 2.15　FSR 位域描述

位	描述
[31：9]	UNP/SBZP。
[8]	读总是为 0，写被忽略。
[7：4]	指定了当发生数据错误时 16 个域（D15 – D0）中哪个域正被访问。
[3：0]	产生的错误类型。见表 2.16。

表 2.16表示了FSR中状态域使用的编码，以及域字段中是否包含有效信息。参看错误地址和错误状态寄存器以获取MMU中止的细节。

表 2.16　FSR 状态域编码

优先级	源	大小	状态	域
最高	对齐	-	b00x1	有效
	转换中的外部中止	一级	b1100	无效
		二级	b1110	有效
	转换	节和页	b0101	无效
			b0111	有效
	域	节和页	b1001	有效
			b1011	有效
	许可	节和页	b1101	有效
			b1111	有效
最低	外部中止	节或页	b10x0	无效

2.3.7 错误地址寄存器c6

寄存器 c6 访问错误地址寄存器（Fault Address Register，FAR）。FAR 包含当数据中止发生时尝试访问的修改过的虚拟地址（MVA）。FAR 仅因数据中止而更新，不因预取指中止而更新。FAR 在 MMU 禁能时由于发生对齐错误和外部中止而更新。

用户可以使用下面的指令来访问 FAR：

```
MRC     p15, 0, <Rd>, c6, c0, 0          ; read FAR
MCR     p15, 0, <Rd>, c6, c0, 0          ; write FAR
```

写寄存器 c6 将 FAR 设置为写入的数据值。这对于调试器将 FAR 的值重置为前一个状态显得有用。

当读写 CP15 的 c6 时 CRm 和 Opcode_2 域都应该为 0。

2.3.8 Cache操作寄存器c7

寄存器 c7 控制着 cache 和写缓冲。每个 cache 操作的功能由用来写 CP15 的 c7 的 MCR 指令中 Opcode_2 和 CRm 域来选择。写其他的 Opcode_2 或 CRm 值结果将不可预测。

读CP15 的c7 是不可预测的，并伴随两个测试和清理操作的异常。见表 2.18和测试和清理操作。

用户可以使用下面的指令来写 c7：

```
MCR     p15, <Opcode_1>, <Rd>, <CRn>, <CRm>, <Opcode_2>
```

表 2.17列出了cache功能，和该寄存器提供的每个功能的描述。

表 2.17　寄存器 c7 功能描述

功能	描述
清除 cache	清除所有 cache 数据，包括任何脏的数据。
使用索引或修改过的虚拟地址来清除单个条目	清除单个 cache 行，丢弃任何脏的数据。
使用索引或修改过的虚拟地址来清理单个数据条目	如果指定的 DCache 行被标记为有效和脏，则将该行写到主存储器。之后该行被标记为非脏。有效位并不改变。
使用索引或修改过的虚拟地址来清理和清除单个数据条目	如果指定的 DCache 行被标识为有效和脏，则将该行写到主存储器。之后该行被标记为失效。
测试和清理 DCache	测试多个cache行，如果任意一行为脏则清理该行。在位 30 中返回总的cache脏状态。参看测试和清理操作。
测试、清理和清除 DCache	用作测试和清理，除了当整个cache被测试和清理时，DCache被清除。参看测试和清理操作。
预取 ICache 行	对指定修改过的虚拟地址执行一个 ICache 查询。如果 cache 未命中，则该区域是高速缓存的，并执行一个行填充。
排干写缓冲	该指令直接起内存栅的作用。它排干了在本指令完成前以程序顺序事件存放在内存中的写缓冲中所有内容。在它完成（排干）之前没有以程序顺序的指令事件在这条指令之后被执行。这可以用在到二级存储系统的具体存储时间需要被控制的场合，比如到一个中断应答位置的存储必须在中断被使能之前完成。
等待中断	该指令排干写缓冲中的内容，并使处理器进入一个低功耗状态，并且在中断或调试请求发生之前阻止处理器执行下一步的指令。当一个中断确实发生时，本 MCR 指令完成，并且正常进入 IRQ 或 FIQ 处理程序。在 R14_irq 或 R14_fiq 中的返回链接包含着本 MCR 指令地址加上 8 的地址，因此用于中断返回的典型指令（SUBS　PC, R14, #4）返回到本 MCR 之后的指令。

表 2.18列出了用于c7 的cache操作功能和相关的数据、指令格式。

表 2.18　c7 的 Cache 操作

功能/操作	数据格式	指令	
清除 ICache 和 DCache	SBZ	MCR	p15, 0, <Rd>, c7, c7, 0
清除 ICache	SBZ	MCR	p15, 0, <Rd>, c7, c5, 0
清除 ICache 单个条目（MVA）	MVA	MCR	p15, 0, <Rd>, c7, c5, 1
清除 ICache 单个条目（Set/Way）	Set/Way	MCR	p15, 0, <Rd>, c7, c5, 2

续上表

功能/操作	数据格式	指令	
预取 ICache 行（MVA）	MVA	MCR	p15, 0, <Rd>, c7, c13, 1
清除 DCache	SBZ	MCR	p15, 0, <Rd>, c7, c6, 0
清除 DCache 单个条目（MVA）	MVA	MCR	p15, 0, <Rd>, c7, c6, 1
清除 DCache 单个条目（Set/Way）	Set/Way	MCR	p15, 0, <Rd>, c7, c6, 2
清理 DCache 单个条目（MVA）	MVA	MCR	p15, 0, <Rd>, c7, c10, 1
清理 DCache 单个条目（Set/Way）	Set/Way	MCR	p15, 0, <Rd>, c7, c10, 2
测试和清理 DCache	-	MRC	p15, 0, <Rd>, c7, c10, 3
清理和清除 DCache 条目（MVA）	MVA	MCR	p15, 0, <Rd>, c7, c14, 1
清理和清除 DCache 条目（Set/Way）	Set/Way	MCR	p15, 0, <Rd>, c7, c14, 2
测试、清理和清除 DCache	-	MRC	p15, 0, <Rd>, c7, c14, 3
排干写缓冲	SBZ	MCR	p15, 0, <Rd>, c7, c10, 4
等待中断	SBZ	MCR	p15, 0, <Rd>, c7, c0, 4

CP15 C7 的MCR操作中MVA的格式见图 2.9。Tag、Set和Word域定义了MVA。对于所有的cache操作，Word应该为 0。

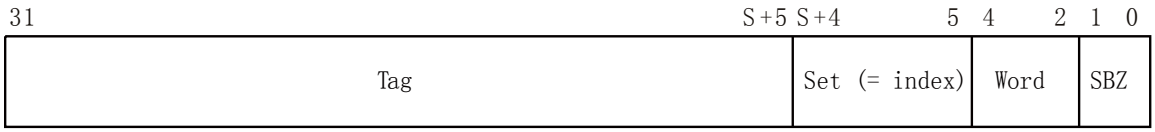


图 2.9　寄存器 c7 的 MVA 格式

CP15 c7 MCR操作的Rd中Set/Way的格式见图 2.10，其中A和S是相连性（associativity）和组数目以 2 为底的对数。组（Set）、路（Way）和字（Word）域定义了格式。对于cache的所有操作，字（Word）应该为 0。

对于 16KB 的 cache，4 路一组相连，每行 8 个字，那么：

- A=$\log_2$associativity=$\log_2$4=2；
- S=$\log_2$NSETS，其中：

  NSETS=cache size in bytes / associativity / line length in bytes:：

  NSETS=16384 / 4 / 32 = 128

  因此：

  S=$\log_2$128=7

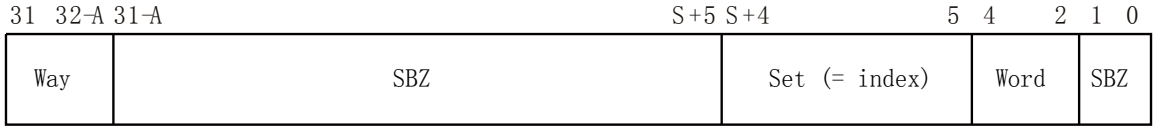


图 2.10　寄存器 c7 的 Set / Way 格式

测试和清理操作

测试和清理 DCache 指令提供了一个使用简单循环来清理整个 DCache 的有效方式。测试和清理 DCache 指令测试 DCache 中的一系列行来检测这些行是否为脏。如果找到任何脏的行，那么这些脏的行之一将被清理。测试和清理 DCache 指令也在位 30 中返回整个 DCache 的状态。

注：测试和清理 DCache 指令 MRC　p15, 0, r15, c7, c10, 3 是一个特殊编码的指令，它使用 r15 作为一个目的操作数。然而，使用该指令并不改变 PC。这个 MRC 指令也设置条件编码标志。

如果 cache 包含任何脏的行，位 30 被设置为 0。如果 cache 不包含脏的行，位 30 被设为 1。这意味着用户可以使用下面的循环来清理整个 DCache：

```
tc_loop:    MRC     p15, 0, r15, c7, c10, 3     ; test and clean
            BNE     tc_loop
```

测试、清理和清除 DCache 指令与测试和清理 DCache 指令类似，除了当整个 cache 被清理之后，DCache 被清除。这意味着用户可以使用下面的循环来清理和清除整个 DCache：

```
tci_loop:   MRC     p15, 0, r15, c7, c14, 3     ; test clean and invalidate
            BNE     tci_loop
```

2.3.9 TLB操作寄存器c8

这是一个只写寄存器，用来控制转换后备缓冲（Translation Lookaside Buffer，TLB）。只有一个 TLB，用来保持数据和指令的条目。TLB 被分为两个部分：

- 一个组相连（set-associative）部分；
- 一个全相连（fully-associative）部分。

全相连部分，也被提及为 TLB 的锁定部分，被用来存储将被锁定的条目。TLB 锁定部分中保持的条目在清除 TLB 操作期间被保持。使用清除 TLB 单个条目操作可以将条目从锁定 TLB 中移除。

定义了 6 种 TLB 操作，要执行的功能由用来写 CP15 c8 的 MCR 指令中 Opcode_2 和 CRm 域来选择。写其他的 Opcode_2 或 CRm 值将导致不可预测的结果。读该寄存器也将导致不可预测的结果。

用户可以使用表 2.19中列出的指令来执行TLB操作。

表 2.19　寄存器 c8 的 TLB 操作

ARMv4/ARMv5 操作	ARM926EJ-S 操作	数据	指令	
清除 TLB	清除组相连 TLB	SBZ	MCR	p15, 0, <Rd>, c8, c7, 0
清除 TLB 单个条目（MVA）	清除单个条目	MVA	MCR	p15, 0, <Rd>, c8, c7, 1
清除指令 TLB	清除组相连 TLB	SBZ	MCR	p15, 0, <Rd>, c8, c5, 0
清除指令 TLB 单个条目（MVA）	清除单个条目	MVA	MCR	p15, 0, <Rd>, c8, c5, 1
清除数据 TLB	清除组相连 TLB	SBZ	MCR	p15, 0, <Rd>, c8, c6, 0
清除数据 TLB 单个条目（MVA）	清除单个条目	MVA	MCR	p15, 0, <Rd>, c8, c6, 1

这些指令是打算用于双 TLB 的实现中，比如 ARM920T 内核或 ARM1020T 内核，可应用于任何条目，而不必考虑将条目加载到 TLB 中的访问类型。详情参考《ARM Architecture Reference Manual》。

清除TLB操作清除了所有TLB中未保持的条目。清除TLB单个条目操作清除与Rd中给出的修改过的虚拟地址相对应的任何TLB条目，而忽略条目的保持状态。参见TLB锁定寄存器c10以获取如何保持TLB中条目的描述。

图 2.11表示了用在TLB单个条目操作中修改过的虚拟地址格式。

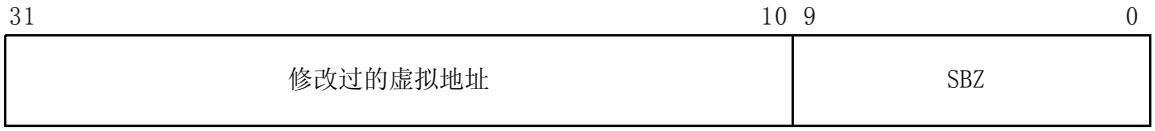


图 2.11　寄存器 c8 的 MVA 格式

注意：如果使用较小或较大的页，并且这些页包含不同的子页访问许可，那么用户必须使用四个清除 TLB 单个条目操作，MVA 设为各个子页，来清除 TLB 中保持的与该页相关的所有信息。

2.3.10　Cache锁定和TCM区域寄存器c9

寄存器 c9 访问 Cache 锁定和 TCM 区域寄存器。访问的寄存器由 CRm 域的值决定：

CRm = c0　　选择 Cache 锁定寄存器；

CRm = c1　　选择 TCM 区域寄存器。

其他 CRm 值保留。

Cache锁定寄存器c9

Cache 锁定寄存器使用基于路的高速缓存（cache-way-based）锁定方案，格式 C，这使得用户可以独立控制每个 cache 路。

这些寄存器让用户能控制四路 cache 中的哪一路 cache 被用作行填充时的分配空间。当这些寄存器被定义时，随后的行填充仅被放入到指定的目标 cache 路。这在由特定应用引起的 cache 污染方面给了用户一些控制机制，并提供一个传统的锁定操作，用于将关键代码锁到 cache 中。

对于每个cache路有一个锁定位来确定是否普通的cache分配被允许访问该路cache。见表 2.21。

四路相连的 cache 中最大有三路 cache 能被锁定，以确保普通 cache 行替换能被执行。

注意：如果没有 cache 路的 L 位被设为 0，那么 cache 路 3 被用作所有的行填充。

该寄存器（c9）的前四位确定了相连 cache 路的 L 位。MRC 或 MCR 指令中的 Opcode_2 域确定访问的是指令或数据锁定寄存器：

Opcode_2 = 0　　选择 DCache 锁定寄存器；

Opcode_2 = 1　　选择 ICache 锁定寄存器。

用户可以使用表 2.20中列出的指令来访问Cache锁定寄存器。

表 2.20　Cache 锁定寄存器指令

功能	数据	指令	
读 DCache 锁定寄存器	L 位	MRC	p15,0,<Rd>,c9,c0,0
写 DCache 锁定寄存器	L 位	MCR	p15,0,<Rd>,c9,c0,0
读 ICache 锁定寄存器	L 位	MRC	p15,0,<Rd>,c9,c0,1
写 ICache 锁定寄存器	L 位	MCR	p15,0,<Rd>,c9,c0,1

用户必须使用读-修改-写的顺序来修改 Cache 锁定寄存器。例如：

```
MRC     p15, 0, <Rn>, c9, c0, 1 ;
ORR     <Rn>, <Rn>, 0x01 ;
MCR     p15, 0, <Rn>, c9, c0, 1 ;
```

这个操作序列设置ICache路 0 的L位为 1。cache锁定寄存器c9 的格式见图 2.12。

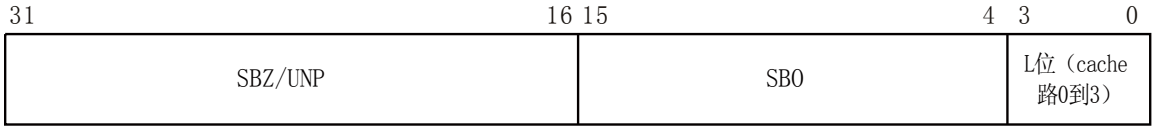


图 2.12　cache 锁定寄存器 c9 格式

Cache锁定寄存器L位的格式见表 2.21。所有的cache路从复位起都可用作分配。

表 2.21　Cache 锁定寄存器 L 位格式

位	四路相连性	说明
[31：16]	UNP/SBZP	保留
[15：4]	0xFFF	SBO
3	路 3 的 L 位	位[3：0]为每个 cache 路的 L 位： 0 = 到该路 cache 的分配由标准替换算法决定（复位状态）； 1 = 该 cache 路不参与分配。
2	路 2 的 L 位	
1	路 1 的 L 位	
0	路 0 的 L 位	

用户可以使用下面描述的 cache 锁定和 cache 解锁步骤：

- 指定地址加载到一个cache路；
- Cache解锁步骤。

指定地址加载到一个cache路

使用格式 C 来锁定代码和数据到 N 路 cache 中第 i 路 cache 的步骤，涉及到使除目标 cache 路之外能分配任何 cache 路：

1. 确保执行本步骤时没有处理器异常发生，例如通过禁止中断的方式。如果这不能做到，任何异常处理程序用到的代码和数据必须被视作步骤 2 和 3 中的代码和数据；
2. 如果 ICache 路正被锁定，确保锁定程序执行的所有代码位于内存中一个非高速缓存的区域，包括 TCM，或在已经锁定的 cache 路中；
3. 如果 DCache 正被锁定，确保锁定程序使用的所有数据位于内存中一个非高速缓存的区域，包括 TCM，或在已经锁定的 cache 路中；
4. 确保将被锁定的指令/数据位于内存中一个可高速缓存的区域；
5. 确保将被锁定的指令/数据并不已经位于（将被锁定的）cache 中。使用寄存器 c7 清理和/或清除操作来确保这一点；
6. 将 CRm == 0，设置位 i 的 L == 0 而其它路的 L == 1 写入到寄存器 c9 中。这使能了目标 cache 路的分配；
7. cache 路 i 中对于每个将被锁定的 cache 行：
   - 如果 DCache 将被锁定，使用 LDR 指令从存储器 cache 行中加载一个字来确保存储器 cache 行被加载到 cache 中；
   - 如果 ICache 将被锁定，使用寄存器 c7 的 MCR 预取 ICache 行（CRm == c13，Opcode_2 == 1）来取存储器 cache 行到 cache 中。
8. 将 CRm == 0，设置位 i 的 L == 1 写入寄存器 c9，并重置所有其它的位为锁定程序开始之前的值。

Cache解锁步骤

要解锁 cache 中锁定的区域，对合适的位写寄存器 c9 并设置 L==0。例如，下面的序列将 ICache 的路 0 的 L 位设置为 0，解锁路 0：

```
MRC     p15, 0, <Rn>, c9, c0, 1;
BIC     <Rn>, <Rn>, 0x01 ;
MCR     p15, 0, <Rn>, c9, c0, 1;
```

TCM区域寄存器c9

ARM926EJ-S处理器支持物理索引（physically-indexed）、物理标记（physically-tagged）的TCM。TCM区域寄存器支持一个指令TCM区域和一个数据TCM区域。能够支持的最小的TCM区域大小是 4KB。TCM状态寄存器指示TCM存储器是否被附属上。详情参看TCM状态寄存器c0。每个TCM区域的大小由DRSIZE和IRSIZE输入引脚定义。

数据TCM在复位时总是被禁能的。在复位时如果INITRAM引脚为高则指令TCM使能。这使能了从指令TCM的启动并且设置了在ITCM区域寄存器中的ITCM使能位。用户可以使用表2.22中列出的TCM区域寄存器指令。

表 2.22　TCM 区域寄存器指令

功能	数据	指令	
读数据 TCM 区域寄存器	基地址	MRC	p15,0,<Rd>,c9,c1,0
写数据 TCM 区域寄存器	基地址	MCR	p15,0,<Rd>,c9,c1,0
读指令 TCM 区域寄存器	基地址	MRC	p15,0,<Rd>,c9,c1,1
写指令 TCM 区域寄存器	基地址	MCR	p15,0,<Rd>,c9,c1,1

TCM区域寄存器格式见图 2.13。

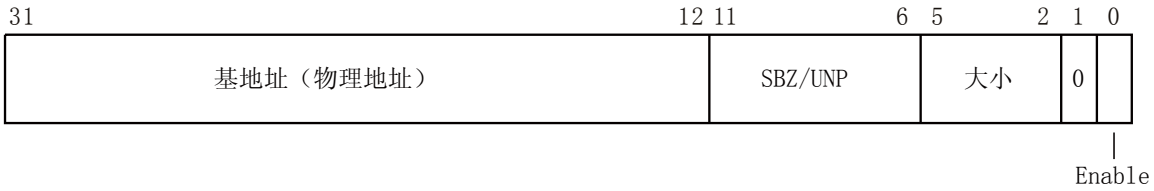


图 2.13　TCM 区域寄存器 c9 格式

表 2.23表示了TCM区域寄存器的位分配。

表 2.23　TCM 区域寄存器 c9

位	功能
[31：12]	基地址（物理地址）
[11：6]	SBZ/UNP
[5：2]	大小。大小域反映了IRSIZE/DRSIZE宏单元输入的值。大小域的编码见表 2.24。
[1]	SBZ/UNP
[0]	使能位： 0=禁能； 1=使能。

表 2.24　TCM 大小域编码

存储器大小	值
0KB/不存在	b0000
保留	b0001，b0010
4KB	b0011
8KB	b0100
16KB	b0101
32KB	b0110
64KB	b0111
128KB	b1000
256KB	b1001
512KB	b1010
1MB	b1011
保留	b1100，b1101，b1110，b1111

如果数据或指令 TCM 被禁能，那么相应 TCM 的内容则不被访问。如果 TCM 随后被重新使能，则 TCM 中的内容没有被 ARM926EJ-S 处理器修改。

对于哈佛体系而言，指令端的TCM必须在正常操作、加载代码或调试活动中可读写访问。这使能了到文字池的访问，未定义指令的仿真和SWI操作的参数传递。用户必须在到指令TCM的写和从指令TCM的读之间插入一个指令内存栅（Instruction Memory Barrier，IMB）。详情参考第九章指令内存栅。

注意：从数据 TCM 获取指令是不可能的。尝试从一个在数据 TCM 空间的地址获取指令并不导致一个到数据 TCM 的访问，而指令实际是从主存储器获取的。这些访问会导致外部中止，因为地址范围可能不被主存储器支持。

指令 TCM 禁止被编程为和数据 TCM 一样的基地址。如果这两个 TCM 是不同大小的，那么物理存储器中的区域不能重叠。如果它们确实重叠，则不能预测被访问的是哪个存储器。

注意：基地址值的设置必须和 TCM 大小对齐。

2.3.11　TLB锁定寄存器c10

TLB锁定寄存器控制着硬件页表搜索（walk）在何处放置到TLB条目，是在TLB的组相连区域还是锁定区域，以及如果是在锁定区域，写入哪个条目。TLB锁定区域包含着 8 个条目。参见TLB结构以获取TLB结构体的描述。

写 TLB 锁定寄存器并将保护位（P 位）设为：

1　　意味着随后的硬件页表搜索放置在锁定区域中的 TLB 条目，被替换的条目由 victim（替换者）指定，范围是 0 到 7；

2　　意味着随后的硬件页表搜索放置在 TLB 的组相连区域中的 TLB 条目。

在锁定区域中的TLB条目是受保护的，因此清除TLB操作仅清除TLB中未保护的条目。也就是说，这些（未保护的）条目在组相连的区域中。清除TLB单个条目操作清除由Rd中给出的修改过的虚拟地址对应的任何TLB条目，而不管它们的保护状态。也就是说，（被清除的单个）条目是否在TLB的锁定或组相连区域。参看TLB操作寄存器c8以获取TLB清除操作的描述。

表 2.25列出了用户可以用来编程TLB锁定寄存器的指令。

表 2.25　编程 TLB 锁定寄存器

功能	指令	
读数据 TLB 锁定 victim	MRC	p15,0,<Rd>,c10,c0,0
写数据 TLB 锁定 victim	MCR	p15,0,<Rd>,c10,c0,0

图 2.14表示了TLB锁定寄存器的格式。

图 2.14　TLB 锁定寄存器格式

victim（替换者）在导致一个条目写入 TLB 的锁定部分的任何硬件页表搜索之后自动增加。

注意：对一个锁定的条目不可能完整映射到较小或较大的页，除非所有的子页访问许可都相同。条目仍可以被写入到锁定区域，但是映射的地址范围仅覆盖了用来执行页表搜索的地址对应的子页。

程序清单 2.1是将一个条目锁定到当前victim的代码序列。

程序清单 2.1　锁定一个条目到当前 victim

```
ADR     r1,LockAddr             ;设置 r1 为将要锁定的地址值
MCR     p15,0,r1,c8,c7,1        ;清除 TLB 单个条目以确保 LockAddr 不在 TLB 中
MRC     p15,0,r0,c10,c0,0       ;读锁定寄存器
ORR     r0,r0,#1                ;设置保护位
MCR     p15,0,r0,c10,c0,0       ;写锁定寄存器
LDR     r1,[r1]                 ;TLB 将丢失，而条目将被加载
MRC     p15,0,r0,c10,c0,0       ;读锁定寄存器(victim 将自增)
BIC     r0,r0,#1                ;清除保护位
MCR     p15,0,r0,c10,c0,0       ;写锁定寄存器
```

2.3.12　寄存器c11 和c12

访问、读或写这些寄存器将引起不可预测的行为。

2.3.13　处理器ID寄存器c13

寄存器 c13 访问处理器标示符寄存器。访问的寄存器取决于 Opcode_2 域的值：

Opcode_2 = 0　选择快速上下文切换扩展（Fast Contex Switch Extension，FCSE）进程标示符（Process Identifier，PID）寄存器；

Opcode_2 = 1　选择上下文 ID 寄存器。

用户可以使用进程ID寄存器来确定当前正在运行的进程。进程标示符在复位时被设置为0。

FCSE PID寄存器

由 ARM9EJ-S 内核发出在范围 0 到 32MB 内的地址将根据该寄存器内包含的值进行转化。地址 A 变成 A+（FCSE PID×32MB）。这就是 cache、MMU 和 TCM 接口看到的修改过的地址。大于 32MB 的地址不被修改。FCSE PID 是一个 7 位的域，能够映射 128 个×32MB 的进程。

如果 FCSE PID 为 0，在由 ARM9EJ-S 内核输出的虚拟地址和由 cache、MMU 与 TCM 接口使用的修改过的虚拟地址之间是平面映射。FCSE PID 在系统复位时被设为 0。

如果 MMU 被禁能，那么没有 FCSE 地址转化发生。

FCSE 转化没有应用于基于条目的 cache 或 TLB 维护操作的地址。对这些操作，VA=MVA。

表 2.26表示了能够用来访问FCSE PID寄存器的ARM指令。

表 2.26　FCSE PID 寄存器操作

功能	数据	ARM 指令	
读 FCSE PID	FCSE PID	MRC	p15,0,<Rd>,c13,c0, 0
写 FCSE PID	FCSE PID	MCR	p15,0,<Rd>,c13,c0, 0

图 2.15表示了FCSE PID寄存器的格式。

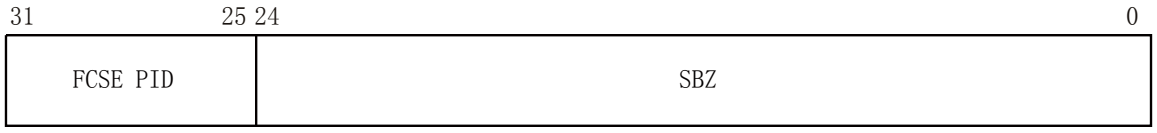


图 2.15　进程 ID 寄存器格式

执行一个快速上下文切换

用户可以通过写 CP15 寄存器 c13 且 Opcode_2=0 的方式来执行一个快速上下文切换。cache 和 TLB 的内容没必要在快速上下文切换之后清空（flush），因为它们仍然保持有效的地址标签。在 FCSE PID 被写入之后的两条指令是以旧的 FCSE PID 预取的[1]，如下面的例程所示：

```
{FCSE PID = 0}
    MOV     r0, #1:SHL:25                       ;Fetched with FCSE PID = 0
    MCR     p15,0,r0,c13,c0,0                   ;Fetched with FCSE PID = 0
    A1                                          ;Fetched with FCSE PID = 0
    A2                                          ;Fetched with FCSE PID = 0
    A3                                          ;Fetched with FCSE PID = 1
```

[1]: 这是由于 ARM 的流水线特性造成的。

其中 A1、A2 和 A3 是跟随在快速上下文切换之后的三条指令。

上下文ID寄存器

上下文 ID 寄存器提供一种在多任务环境中让实时跟踪工具能够辨别当前正在执行的进程的机制。

该寄存器中的内容被复制到 ARM926EJ-S 处理器的 ETMPROCID 引脚上。当写上下文 ID 寄存器时 ETMPROCIDWR 引脚上产生一个脉冲。

表 2.27表示了用户能够用来访问上下文ID寄存器的ARM指令。

表 2.27　上下文 ID 寄存器操作

功能	数据	ARM 指令	
读上下文 ID	上下文 ID	MRC	p15,0,<Rd>,c13,c0, 1
写上下文 ID	上下文 ID	MCR	p15,0,<Rd>,c13,c0, 1

在该操作中转移的上下文ID寄存器，Rd的格式见图 2.16。

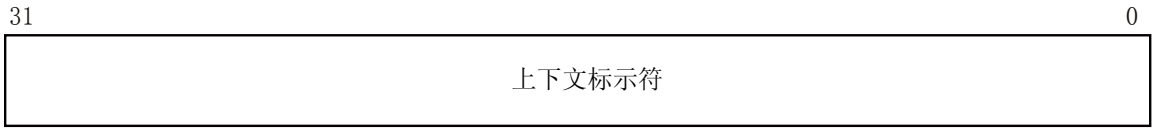


图 2.16　上下文 ID 寄存器格式

2.3.14　寄存器c14

访问、读或写该寄存器被保留。

2.3.15　测试和调试寄存器c15

用户可以使用寄存器c15 来在ARM926EJ-S处理器上提供器件特定（device-specific）的测试和调试操作。附录B CP15 测试和调试寄存器描述了该寄存器和使用CP15 c15 可用的功能。该寄存器在ARM体系结构参考手册中被定义为保留给实现定义（implementation-defined）的目的。如果用户编写的软件使用了由c15 提供的器件特定的功能，那么这个软件未必能够后向或前向兼容。

第3章 存储器管理单元

本章描述了存储器管理单元（Memory Management Unit，MMU）。包含以下部分：

- 关于MMU；
- 地址转换；
- MMU错误和CPU中止；
- 域访问控制；
- 错误检查顺序；
- 外部中止；
- TLB结构。

3.1 关于MMU

ARM926EJ-S MMU 是 ARM 体系 v5 的 MMU。它提供了由诸如在 Symbian OS、WindowsCE 和 Linux 这类平台上系统操作所要求的虚拟内存特征。存储在主存中的一个单组二级页表被用来控制数据和指令访问的地址转换、许可检查和内存区域属性。

MMU 使用一个统一的转换后备缓冲（TLB）来高速缓存页表中保持的信息。

为了支持节（section）和页（page），使用两级地址转换。MMU 将转换过的物理地址放入 MMU 转换后备缓冲 TLB。

MMU TLB 有两部分：

- 主 TLB；
- 锁定 TLB。

主TLB是用于页表信息的两路、组相连的cache。主TLB每路有 32 个条目，总计 64 个条目。锁定TLB是包含锁定的TLB条目的一个 8 条目全相连的cache。锁定TLB的条目能够确定到给定区域的内存访问绝不招致页表搜索的缺陷。更多细节请参考TLB结构。

MMU 的特征如下：

- 标准 ARM 体系 v4 和 v5 MMU 映射大小、域和访问控制方案；
- 映射大小为 1MB（节）、64KB（大页）、4KB（小页）和 1KB（微页）；
- 大页和小页的访问许可能单独指定，适用于每个页的四分之一（子页许可）；
- 硬件页表搜索；
- 使用 CP15 c8 可清除整个 TLB；
- 使用 CP15 c8 可由 MVA 来选择清除 TLB 条目；
- 使用 CP15 c10 来锁定 TLB 条目。

下面的小节是：

- 访问许可和域；
- 转换条目；
- MMU编程可访问的寄存器。

3.1.1 访问许可和域

对于大页和小页，访问许可定义针对每个子页，小页为 1KB，大页为 16KB。节和微页有单独的访问许可设置。

所有内存区域都有一个关联的域（domain）。对于一个内存区域而言域是初级的访问控制机制。它定义了一个访问得以继续进行的必要条件。域确定了是否：

- 访问许可是否被用来限定访问；
- 访问是无条件的允许进行；
- 访问是无条件的中止。

在后两种情况下，访问许可属性被忽略。

总共有 16 个域。使用域访问控制寄存器可以配置它们。参见域访问控制寄存器c3。

3.1.2 转换条目

主 TLB 高速缓存 64 个转换过的条目。如果在一个内存访问期间，主 TLB 包含一个 MVA 的转换条目，那么 MMU 读取保护数据以确定访问是否被许可：

- 如果访问被许可并且要求的是一个片外（存储器）访问，MMU 输出和 MVA 对应的物理地址；
- 如果访问被许可并且要求的不是一个片外（存储器）访问，那么 cache 或 TCM 负责服务这个访问；
- 如果访问不被许可，MMU 向 CPU 内核发送中止信号。

如果 TLB 因为它未包含 MVA 的条目而没命中，则转换表搜索硬件被调用，从一个位于物理存储器中的转换表来检索转换信息。当检索到时，转换信息被写入到 TLB，有可能覆盖一个已经存在的值。

为了使用 TLB 锁定特征，要被写入的位置可以用 CP15 c10 TLB 锁定寄存器来指定。

在复位时 MMU 是关闭的，没有发生地址映射，并且所有区域都被标记为非高速缓存（noncacheable）和非缓冲的（nonbufferable）。

3.1.3 MMU编程可访问的寄存器

表 3.1列出了用于和存储在主存中以确定MMU操作的页表描述符相配合的CP15 寄存器。

表 3.1　MMU 编程可访问的 CP15 寄存器

寄存器	位	寄存器描述
控制寄存器 c1	M，A，S，R	包含使能 MMU 的位 M，使能数据地址对齐检查的位 A，和控制访问保护方案的 S 与 R 位。
转换表基址寄存器 c2	[31：14]	保持着维持在主存中的转换表基址的物理地址。基地址必须位于 16KB 的边界。
域访问控制寄存器 c3	[31：0]	由 16 个两位域构成。每个域定义了 16 个域（D15 到 D0）之一的访问控制属性
错误状态寄存器，IFSR 和 DFSR，c5	[7：0]	指示数据或预取指中止的原因，以及当中止发生时引起中止的访问的域号。位[7：4]指定了当一个错误发生时 16 个域（D15 到D0）中的哪一个正被访问。所有其它位的值是不可预测的。这些位的编码见表 3.9。
错误地址寄存器 c6	[31：0]	保持着引起数据中止相关的MVA。见表 3.9以获取每种错误类型存储的地址的细节。ARM9EJ-S寄存器R14_abt保持着预取指中止相关的VA。
TLB 操作寄存器 c8	[31：0]	该寄存器用于执行 TLB 维护操作。可以清除 TLB 中所有的（未保护的）条目，或清除一个指定的条目。

续上表

寄存器	位	寄存器描述
TLB 锁定寄存器 c10	[28：26]和[0]	使能指定的页表条目锁定到 TLB 中。在 TLB 中锁定的条目保证了到锁定页或节的访问能够进行而不招致在 TLB 未命中时的时间损失。这使得诸如中断处理程序等时间关键的代码段的执行时间能够最小化。

所有的 CP15 MMU 寄存器，除了 c8 之外，包含着可以用 MRC 指令读取的状态，并且可用 MCR 指令写入。寄存器 c5 和 c6 在一个中止期间也由 MMU 写入（相关信息）。写 c8 会引起 MMU 执行一个 TLB 操作，来控制 TLB 条目。这个寄存器是只写的。

第二章编程模型中描述了CP15 寄存器。

3.2 地址转换

由 CPU 内核产生的 VA 被 FCSE 使用 CP15 c13 中保持的值转换成一个修改过的虚拟地址（Modified Virtual Address，MVA）。MMU 将 MVA 转换成物理地址以访问外部存储器，同时执行访问许可检查。

MMU 表搜索硬件被用来添加条目到 TLB 中。位于物理存储器里的转换表中包含着由地址和转换数据组成的转换信息与访问许可数据。MMU 提供用于转换表的自动搬移和加载条目到 TLB 中的逻辑电路。

硬件页表搜索和许可检查处理的级数是一级或二级，取决于地址是被标记为一个节映射（section-mapped）的地址还是一个页映射（page-mapped）的地址。

有三种尺寸的页映射访问和一种尺寸的节映射访问。页映射访问用于：

- 大页（large pages）；
- 小页（small pages）；
- 微页（tiny pages）。

转换处理总是以相同的方式（一个一级读取）开始。节映射的访问仅要求一个一级读取，但页映射的访问要求一个额外的二级读取。

下面的小节是：

- 转换表基址；
- 一级读取；
- 一级描述符；
- 节描述符；
- 粗页表描述符；
- 细页表描述符；
- 转换节参考；
- 二级描述符；
- 转换大页参考；
- 转换小页参考；
- 转换微页参考。

3.2.1 转换表基址

硬件转换处理在TLB未包含要求的MVA的转换结果时初始化。转换表基址寄存器（Translation Table Base Register，TTBR），CP15 寄存器c2，指向物理存储器中包含节和/或页

描述符的转换表基地址。TTBR中的 14 个低阶位[13：0]在读取时不可预测，并且转换表必须位于 16KB的地址边界。图 3.1表示了TTBR的格式。

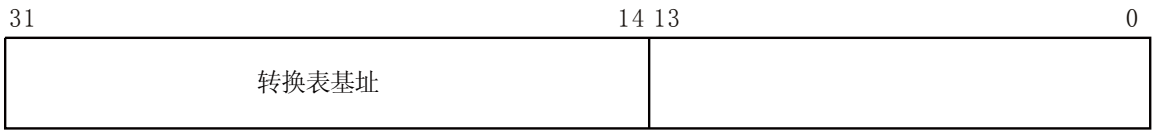


图 3.1　转换表基址寄存器

转换表有可高达 4096×32 位的条目，每个条目描述了 1MB 的虚拟内存。这能够寻址高达 4GB 的虚拟内存。

图 3.2表示了转换表搜索的过程。

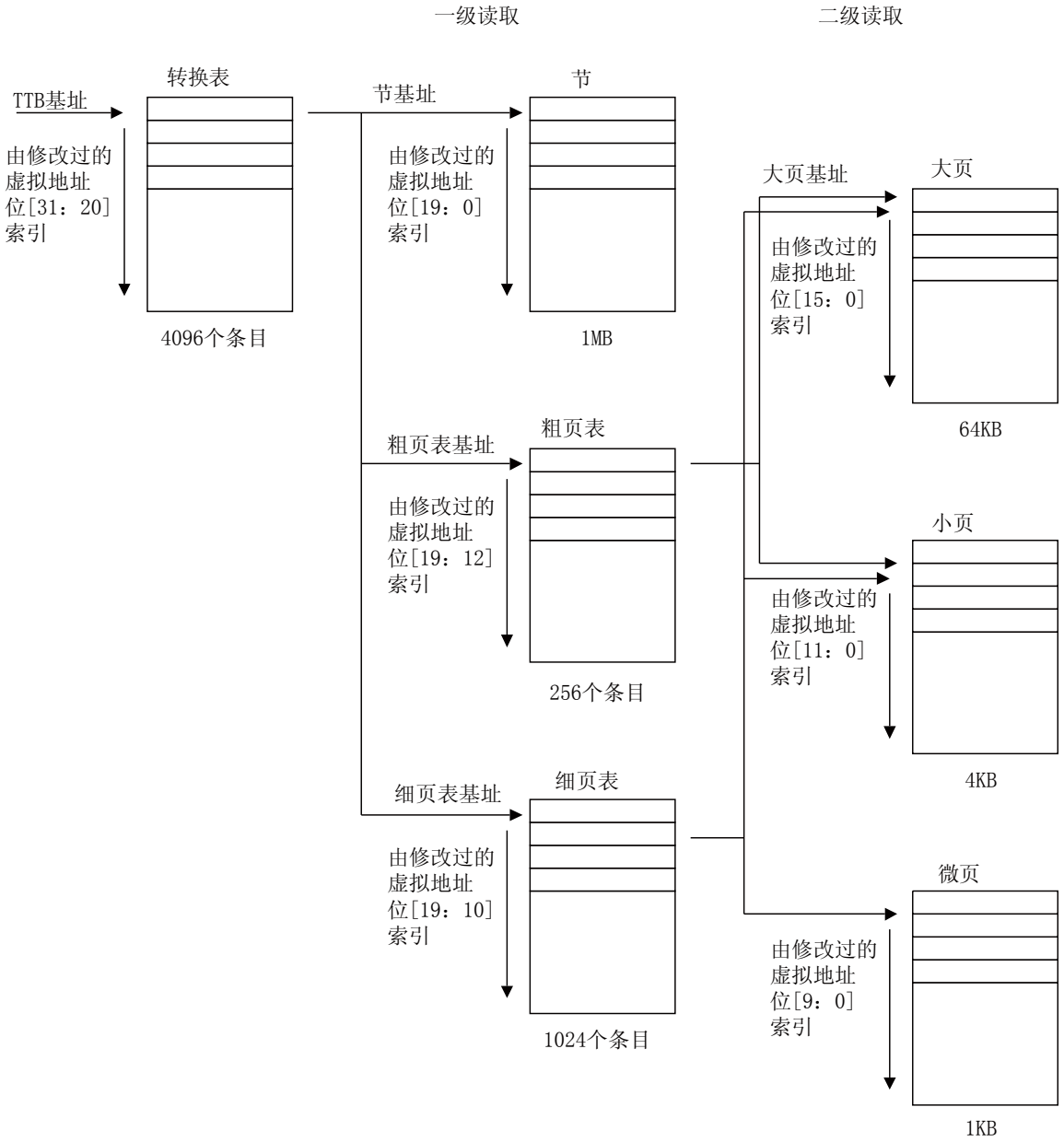


图 3.2　转换页表

3.2.2 一级读取

TTBR的位[31：14]与MVA的位[31：20]连接在一起产生一个 30 位的地址，见图 3.3。

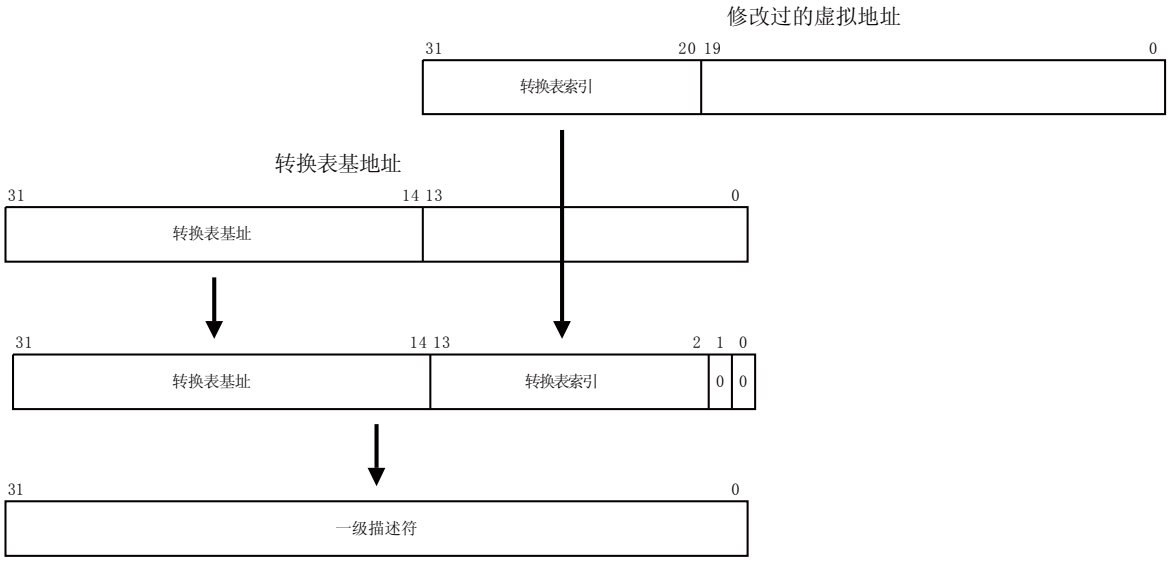


图 3.3 访问转换表一级描述符

这个地址选择一个 4 字节的转换表条目。这是用于节或页表的一级描述符。

3.2.3 一级描述符

一级描述符返回一个节描述符、粗页表描述符或细页表描述符、或无效值。图 3.4表示了一级描述符的格式。

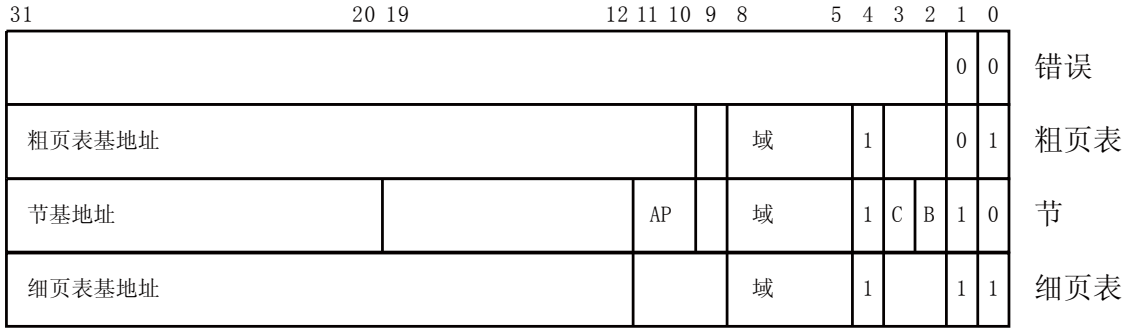


图 3.4 一级描述符

一个节描述符提供了一个 1MB 内存块的基地址。

页表描述符提供了包含着二级描述符的页表的基地址。有两种尺寸的页表：

- 粗页表有 256 个条目，将（粗页）表描述的 1MB 空间划分为 4KB 的块；
- 细页表有 1024 个条目，将（细页）表描述的 1MB 空间划分为 1KB 的块。

一级描述符的位分配见表 3.2。

表 3.2 一级描述符位分配

位			描述
节	粗页	细页	
[31：20]	[31：10]	[31：12]	这些位是来自物理地址相应的位。
[19：0]	-	-	应该为 0。
[11：10]	-	-	访问许可位。访问许可和域以及错误地址和错误状态寄存器表示了如何解释访问许可位。
[9]	[9]	[11：9]	应该为 0。
[8：5]	[8：5]	[8：5]	域控制位。
[4]	[4]	[4]	必须为 1。
[3：2]	-	-	位 C 和 B 表示由该页映射的内存区域被视为是写回（write-back）高速缓存的、写通（write-through）高速缓存的、非高速缓存的缓冲的还是非高速缓存非缓冲的。
-	[3：2]	[3：2]	应该为 0。
[1：0]	[1：0]	[1：0]	这些位表示页大小和有效性，并在表 3.3中作了说明。

一级描述符的两个最低有效位表示描述符类型，见表 3.3。

表 3.3 一级描述符位[1：0]的说明

值	含义	描述
00	无效	产生一个节转换错误。
01	粗页表	表示这是一个粗页表描述符。
10	节	表示这是一个节描述符。
11	细页表	表示这是一个细页表描述符。

3.2.4 节描述符

节描述符提供了一个 1MB内存块的基地址。图 3.5表示了节描述符的格式。

图 3.5 节描述符

表 3.4描述了节描述符的位分配。

表 3.4 节描述符位分配

位	描述
[31：20]	来自一个节的物理地址对应的位
[19：12]	总是写入 0
[11：10]	AP 位指定了该节的访问许可
[9]	总是写入 0
[8：5]	指定 16 个可能的域之一，（域）保持在域访问控制寄存器中，里面包含着主要的访问控制

续上表

位	描述
[4]	为了后向兼容，应该被写为 1。
[3：2]	这些位（C 和 B）表示被该节映射的内存区域是被视为写回高速缓存的、写通高速缓存的、非高速缓存的缓冲的还是非高速缓存非缓冲的
[1：0]	这些位必须为 10 以表示一个节描述符

3.2.5 粗页表描述符

粗页表描述符提供了包含着用于大页或小页访问的二级描述符的页表的基地址。粗页表有 256 个条目，将该表描述的 1MB空间划分为 4KB的块。图 3.6表示了粗页表描述符的格式。

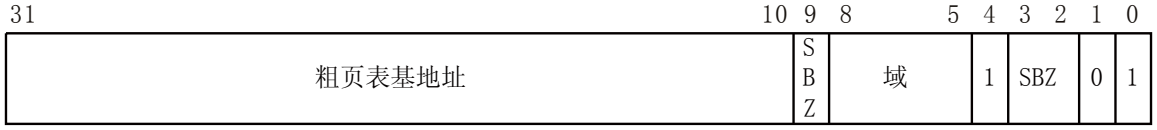


图 3.6　粗页表描述符

注意：如果从一级读取中返回的是一个粗页表描述符，则一个二级读取被初始化。

表 3.5描述了粗页表描述符的位分配。

表 3.5　粗页表描述符位分配

位	描述
[31：10]	这些位来自引用的二级描述符的基址（粗页表索引由 MVA 得到的条目）
[9]	总是写入 0
[8：5]	这些位指定 16 个可能的域之一，（域）保持在域访问控制寄存器中，里面包含着主要的访问控制
[4]	总是写入 1
[3：2]	总是写入 0
[1：0]	这些位必须为 01 以表示一个粗页表描述符

3.2.6 细页表描述符

细页表描述符提供了包含着用于大页、小页或微页访问的二级描述符的页表的基地址。细页表有 1024 个条目，将该表描述的 1MB空间划分为 1KB的块。图 3.7表示了细页表描述符的格式。

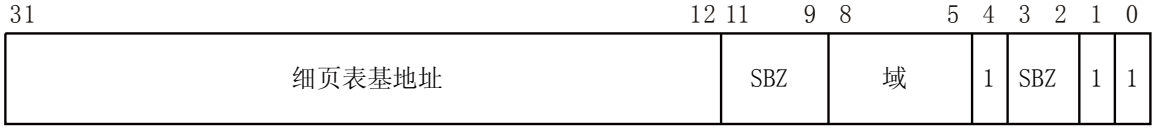


图 3.7　细页表描述符

注意：如果从一级读取中返回的是一个细页表描述符，则一个二级读取被初始化。

表 3.6表示了细页表描述符的位分配。

表 3.6　细页表描述符位分配

位	描述
[31：12]	这些位来自引用的二级描述符的基址（细页表索引由 MVA 得到的条目）
[11：9]	总是写入 0
[8：5]	这些位指定 16 个可能的域之一，（域）保持在域访问控制寄存器中，里面包含着主要的访问控制
[4]	总是写入 1
[3：2]	总是写入 0
[1：0]	这些位必须为 11 以表示一个细页表描述符

3.2.7 转换节参考

图 3.8表示了完整的节转换顺序。

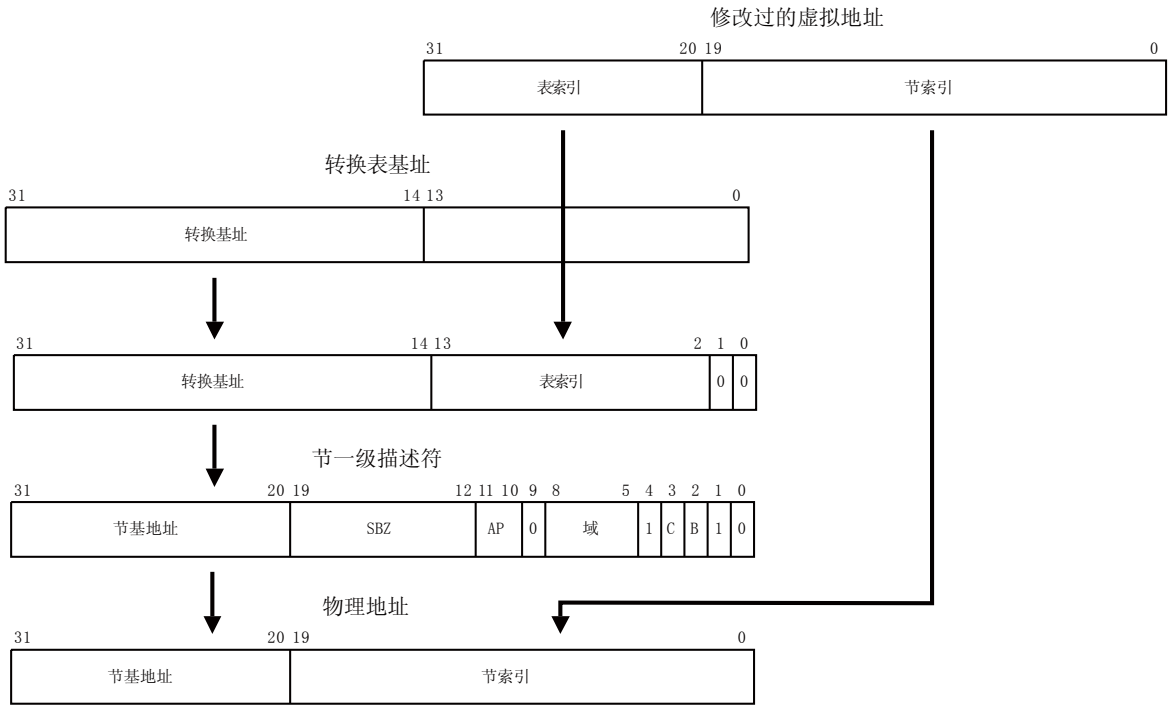


图 3.8　节转换

3.2.8 二级描述符

如果一级读取返回一个粗页表描述符或一个细页表描述符，这提供了待用页表的基地址。然后访问该页表并返回一个二级描述符。图 3.9表示了二级页表描述符的格式。

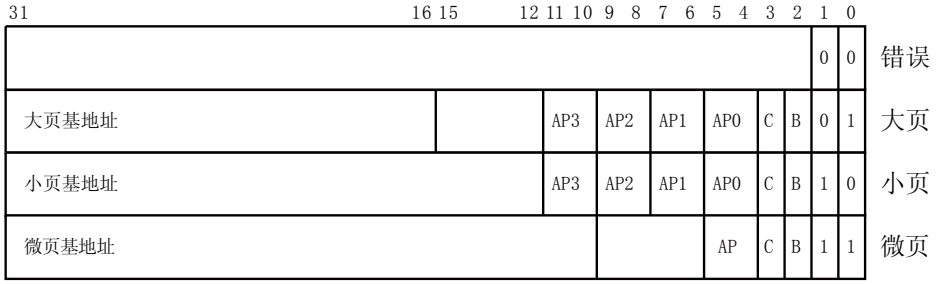


图 3.9 二级描述符

二级描述符定义了一个微页、小页或大页描述符，或者是无效的：

- 大页描述符提供了一个 64KB 内存块的基地址；
- 小页描述符提供了一个 4KB 内存块的基地址；
- 微页描述符提供了一个 1KB 内存块的基地址。

粗页表为大页或小页提供了基地址。大页描述符必须在 16 个连续的条目中重复。小页描述符必须在各自连续的条目中重复。

细页表为大页、小页或微页提供了基地址。大页描述符在 64 个连续的条目中重复。小页描述符必须在 4 个连续的条目中重复而微页描述符必须在各自连续的条目中重复。

表 3.7描述了二级描述符的位分配。

表 3.7 二级描述符位分配

位			描述
大页	小页	微页	
[31：16]	[31：12]	[31：10]	这些位来自物理地址中对应的位
[15：12]	-	[9：6]	应该为 0
[11：4]	[11：4]	[5：4]	访问许可位。域访问控制和错误检查顺序表示了如何解释访问许可位
[3：2]	[3：2]	[3：2]	位 C 和 B，表示由该页映射的内存空间是被视为写回高速缓存的、写通高速缓存的、非高速缓存的缓冲的还是非高速缓存的非缓冲的
[1：0]	[1：0]	[1：0]	这些位表示也大小和有效性，并在表 3.8中做了说明。

二级描述符的两个最低有效位表示描述符类型，见表 3.8。

表 3.8 页表条目位[1：0]说明

值	含义	描述
00	无效的	产生一个页转换错误
01	大页	表示这是一个 64KB 的页
10	小页	表示这是一个 4KB 的页
11	微页	表示这是一个 1KB 的页

注意：微页并不支持子页许可因此只有一个访问许可位的设置。

3.2.9 转换大页参考

图 3.10表示了一个 64KB的大页完整的转换顺序。

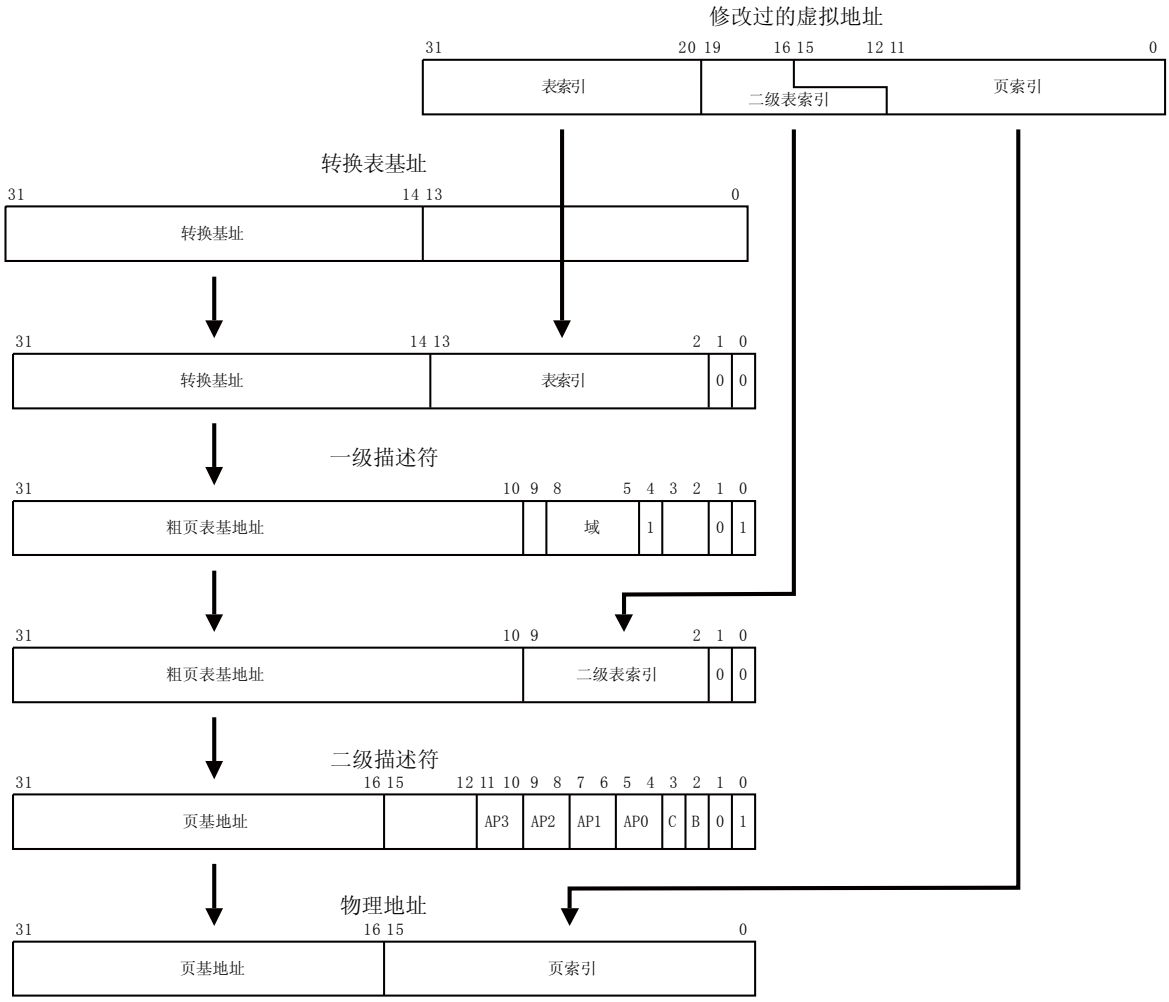


图 3.10　从粗页表进行大页转换

因为页索引的高四位和粗页表索引的低四位交叠，每个用于大页的粗页表条目必须复制 16 次，在粗页表的连续内存空间中。

如果大页描述符被包含在一个细页表中，则页索引的高六位和细页表索引的低六位交叠。因此每个用于大页的细页表条目被复制 64 次。

3.2.10　转换小页参考

图 3.11表示了一个 4KB的小页完整的转换顺序。

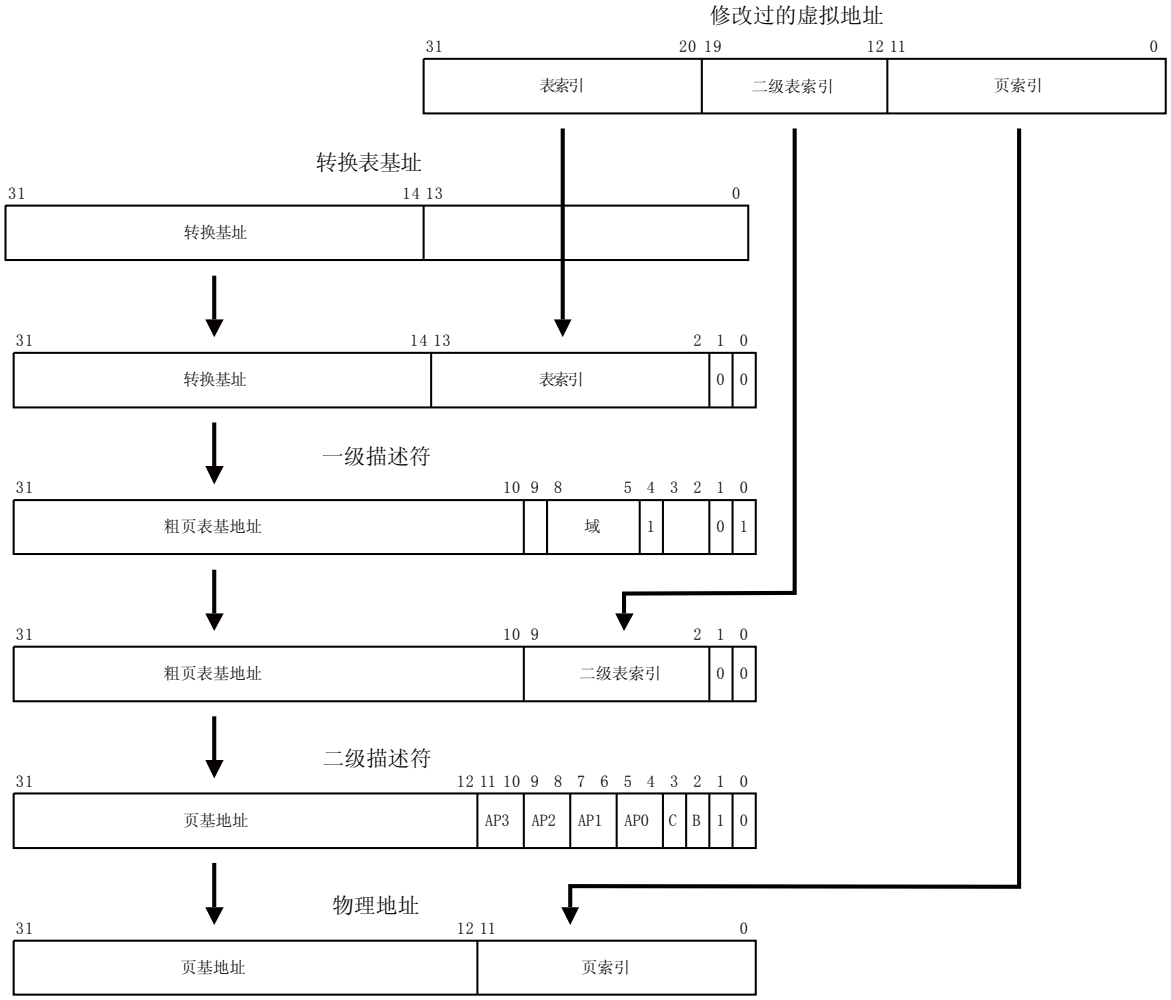


图 3.11　从粗页表进行小页转换

如果小页描述符被包含在一个细页表中，则页索引的高两位和细页表索引的低两位交叠。因此每个用于小页的细页表条目必须被复制 4 次。

3.2.11　转换微页参考

图 3.12表示了一个 1KB的微页完整的转换顺序。

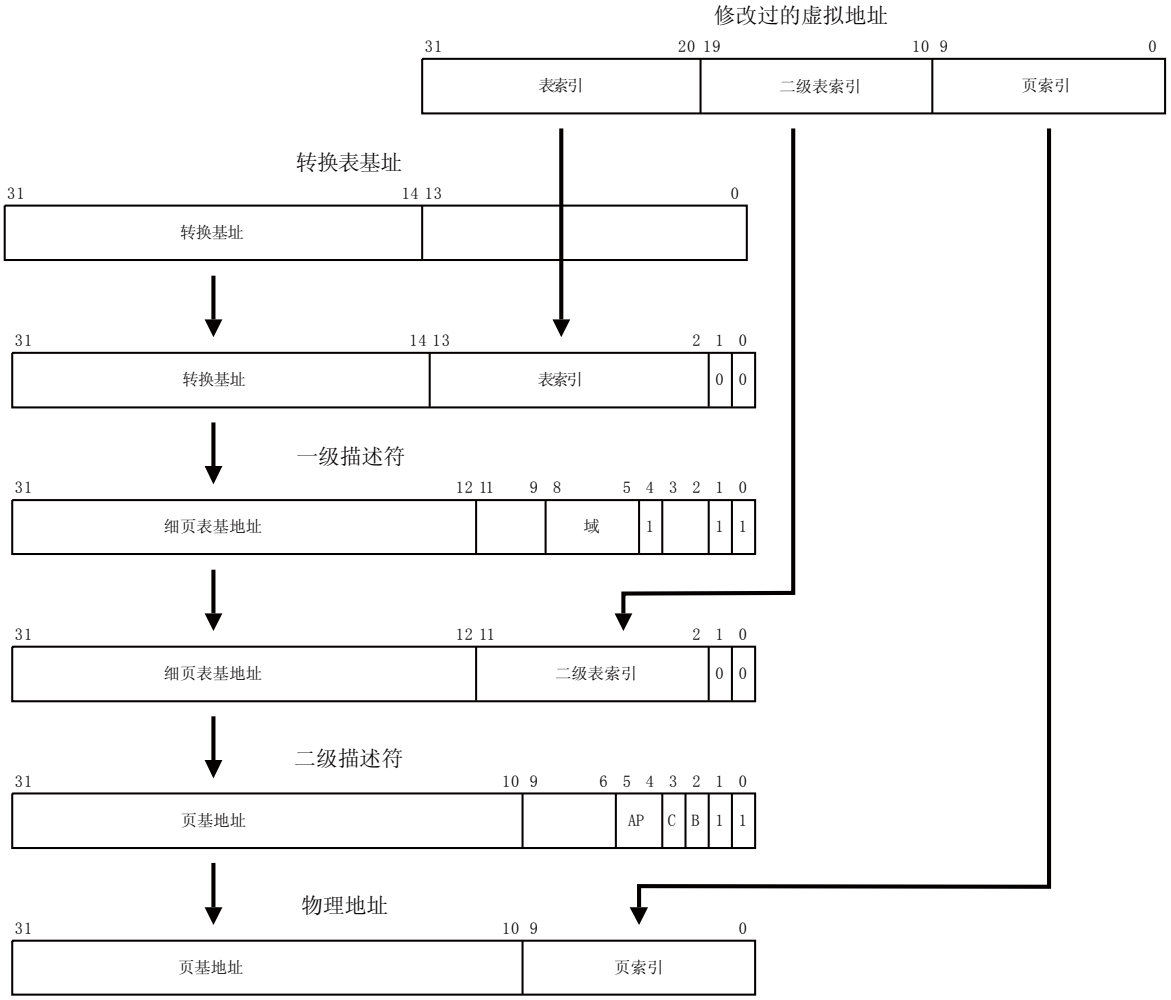


图 3.12　从细页表进行微页转换

页转换包含一个在节转换之上的附加步骤。一级描述符是细页表描述符且这被用来指向一级描述符。

注意：在一级描述中指定的域和在一级描述中指定的访问许可一起确定访问是否有许可需要继续进行。详见域访问控制这一节。

子页

用户可以定义小页和大页的子页访问许可。如果在页表搜索期间，一个小页或大页有不同的子页许可，只有正被访问的子页被写入 TLB。例如，如果子页许可不一致则一个 16KB（大页）的子页条目被写入 TLB，而如果所有子页许可都一样则 64KB 的条目被放入 TLB。

当用户使用子页许可时，而页条目之后变为无效，用户必须分别清除所有的四个子页。

3.3　MMU错误和CPU中止

MMU 在发生下列类型的错误时产生一个中止：

- 对齐错误（仅数据访问）；
- 转换错误；
- 域错误；
- 许可错误。

另外，外部中止也有可能由外部系统引起。这仅发生在将内核和外部系统同步的访问类型：

- 页搜索；
- 非高速缓存的读；
- 非缓冲的写；
- 非高速缓存的读锁写（read-lock-write）序列（SWP）。

对齐错误检查通过 CP15 c1 的 A 位使能。对齐错误检查不受 MMU 是否使能的影响。转换、域和许可错误只在 MMU 使能后才产生。

MMU的访问控制机制检测产生这些错误的条件。如果由于一个存储器访问的错误被检测到，MMU在数据错误状态寄存器和错误地址寄存器中保持由数据访问产生的错误状态和地址信息。参见错误地址和错误状态寄存器。

MMU 也在指令错误状态寄存器中保持由指令读取产生的错误状态。

注意：对于指令端中止的地址信息包含在内核链接寄存器 r14_abt 中。

与给定存储器访问不符的访问阻止了对应的到 AHB 接口上的外部访问，而以一个中止返回给 CPU 内核。

3.3.1 错误地址和错误状态寄存器

在数据中止时，MMU放置一个 4 位的编码值：错误状态，和 4 位编码的域号到数据FSR中。类似的，在预取指中止时，（编码值）也被放入指令FSR中，这仅用作调试目的。另外，与数据中止相应的MVA被锁存到FAR。如果一个访问违规同时产生多于一个的中止源，那么它们（中止源）以优先级编码并在表 3.9中给出。FAR不被由指令预取引起的错误更新。

错误状态寄存器（FSR）

表 3.9表示了由数据MMU支持的多种访问许可和控制，以及这些许可和控制如何说明了产生的错误。

表 3.9　错误状态的优先级编码

优先级	源	大小	状态	域
最高	对齐	-	b00x1	有效
	转换中的外部中止	一级	b1100	无效
		二级	b1110	有效
	转换	节和页	b0101	无效
			b0111	有效
	域	节和页	b1001	有效
			b1011	有效
	许可	节和页	b1101	有效
			b1111	有效
最低	外部中止	节或页	b10x0	无效

注意：对齐错误可以写 b0001 或 b0011 到 FSR 的位[3：0]中。

无效值可以出现在状态位中为域错误编码。当错误是在一个有效的域被从页表描述符中读出前引起时发生。

被较高优先级中止屏蔽的中止可以通过确定较高优先级中止的起因而重新产生，并且重复访问。

对齐错误不可能发生在指令读取中。

指令 FSR 也可以由于指令预取操作而更新：

MCR　　p15, 0, <Rd>, c5, c0, 1

错误地址寄存器（FAR）

对于可以包含超过一个字地传输的加载和存储指令（LDM/STM、LDRD、STDRD和STC/LDC），写入到FAR寄存器中的值取决于访问的类型，且对于外部中止，取决于访问是否越过了 1KB的边界。表 3.10表示了多字传输的FAR值。

表 3.10　多字传输的 FAR 值

源	FAR
对齐	传输中首个中止地址的 MVA
转换中的外部中止	传输中首个中止地址的 MVA
转换	传输中首个中止地址的 MVA
域	传输中首个中止地址的 MVA
许可	传输中首个中止地址的 MVA
非高速缓存的读或非高速缓存的写的外部中止	在 1KB 边界前最后一个地址的 MVA，如果传输中的任何字在 1KB 边界前发生了外部中止。 传输中最后地址的 MVA，如果第一个外部中止的字（出现）在 1KB 边界之后。

兼容性发布

为了让代码能够容易的移植到 ARM 体系 v4 或 v5 的 MMU，或者是将来的体系，建议在外部中止的行为上不要做出信任。

指令 FSR 仅打算用于调试目的。用于移植到其他 ARM 体系 v4 或 v5 MMU 上的代码不能使用指令 FSR。

3.4　域访问控制

MMU 访问主要通过使用域（domain）来控制。有 16 个域且每个域有一个两位段来定义它的访问。支持两种类型的用户：

- 客户；
- 管理员。

域在域访问控制寄存器CP15 c3 中定义。图 2.7表示了这个 32 位的寄存器是如何分配以定义 16 个两位的域。

表 3.11定义了在各自的域内这些位是如何解释以指定访问许可的。

表 3.11　域访问控制寄存器，访问控制位

值	含义	描述
00	无访问	任何访问都产生一个域错误
01	客户	访问根据节或页描述符中的访问许可位进行检查
10	保留	保留。当前行为类似无访问模式
11	管理员	访问不根据访问许可位进行检查，因此不可能产生许可错误

表 3.12表示了如何解释访问许可位（Access Permission，AP）以及它们的解释是如何依赖于控制寄存器c1 的位[9：8]，R和S位。

表 3.12　访问许可（AP）位说明

AP	S	R	特权许可	用户许可
00	0	0	无访问	无访问
00	1	0	只读	无访问
00	0	1	只读	只读
00	1	1	不可预测的	不可预测的
01	x	x	读/写	无访问
10	x	x	读/写	只读
11	x	x	读/写	读/写

3.5　错误检查顺序

MMU用来检查访问错误的顺序对于节和页而言是不同的。两种访问类型的（检查）顺序见图 3.13。

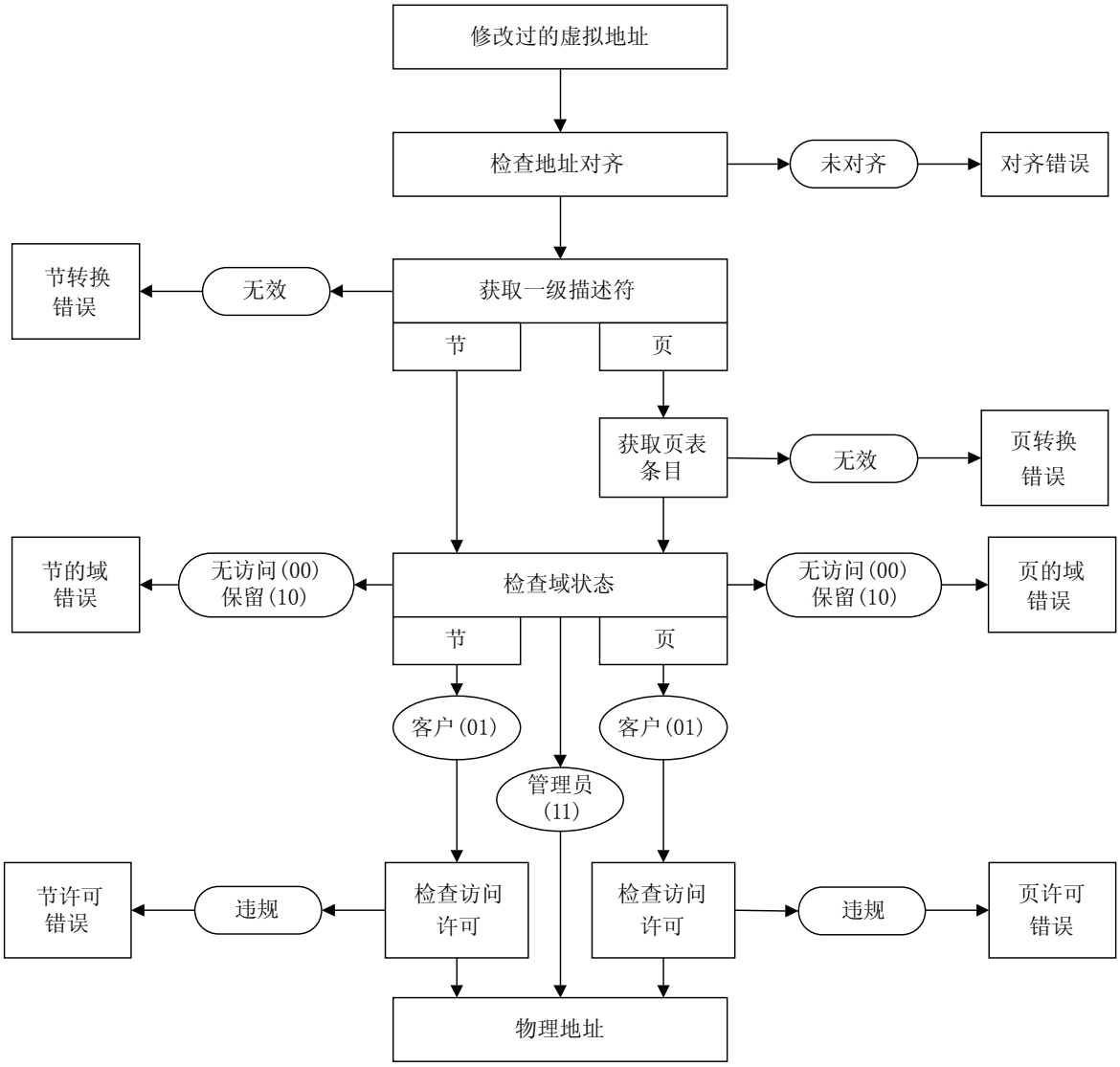


图 3.13　错误检查顺序

产生各种错误的条件描述如下：

- 对齐错误；
- 转换错误；
- 域错误；
- 许可错误。

3.5.1 对齐错误

如果对齐错误检查被使能（CP15 c1 的 A 位置位），MMU 在任何数据字访问的地址不是字对齐，或任何半字访问的地址不是半字对齐时产生一个对齐错误，而不论 MMU 是否被使能。对齐错误不在任何指令读取或字节访问中产生。

注意：如果访问产生了一个对齐错误，访问序列中止而不涉及其他许可检查。

3.5.2 转换错误

有两种类型的转换错误：

节 节转换错误在一级描述符被标记为无效时产生。这在描述符的位[1：0]都为 0 时发生。

页 页转换错误在一级描述符被标记为无效时产生。这在描述符的位[1：0]都为 0 时发生。

3.5.3 域错误

有两种类型的域错误：

节 一级描述符保持四位的域段，这个四位域段选择了域访问控制寄存器的 16 个两位域中的一个。指定域的两个位之后被用作访问许可检查，如表 3.12中描述的一样。域在一级描述符返回时被检查。

页 一级描述符保持四位的域段，这个四位域段选择了域访问控制寄存器的 16 个两位域中的一个。指定域的两个位之后被用作访问许可检查，如表 3.12中描述的一样。域在一级描述符返回时被检查。

如果指定的访问是无访问（00）或保留（10），那么将发生一个节的域错误或页的域错误。

3.5.4 许可错误

如果两位的域段返回 01（客户），那么做如下的访问许可检查：

节 如果一级描述符定义了一个节映射的访问，描述符的AP位根据表 3.12定义了访问是否被允许。它们的解释依赖于 S 和 R 位，即 CP15 c1 的位 8 和 9。如果访问不被允许，则产生一个节许可错误。

大页或小页

如果一级描述符定义了一个页映射的访问且二级描述符用于大页或小页，则指定了四个访问许可域：ap3 到 ap0，每一个访问许可域对应于页的四分之一。对于小页 ap3 被选作页的顶部 1KB（访问许可）而 ap0 被选作页的底部 1KB（访问许可）。对于大页，ap3 被选作页的顶部 16KB（访问许可）而 ap0 被选作页的底部 16KB（访问许可）。被选中的AP位之后以和节严格相同的方式解释。见表 3.12。唯一的区别就是产生的错误是一个页许可错误。

微页 如果一级描述符定义了一个页映射的访问，且二级描述符用于微页，则一级描述

符的 AP 位以和节一样的方式定义了访问是否被允许。产生的错误是一个页许可错误。

3.6 外部中止

MMU除了产生中止之外，还可以产生包含在AHB总线上传输的特定类型访问的外部中止。这可以用来标记在外部存储器访问上的错误。然而，并不是所有的访问都能用这种方式中止。

下面的访问可以是外部中止：

- 页搜索；
- 非高速缓存的读；
- 非高速缓存的写；
- 非高速缓存的读锁写（SWP）顺序。

对于读锁写（SWP）顺序，如果读（产生了）外部中止，写总是会（继续）尝试。

到一个 NCB 区域的交换被强制要求有和到一个 NCNB 区域交换精确一样的行为。这意味着到一个 NCB 区域的写的部分可以外部中止。

3.6.1 使能MMU

在使用 CP15 c1 使能 MMU 之前用户必须：

1. 编程 TTB 寄存器（CP15 c2）和域访问控制寄存器（CP15 c3）；
2. 编程所要求的一级和二级页表，确保有效的转换表被放置在由 TTB 寄存器所指定的内存位置中。

当这些步骤被执行后，用户可以通过设置 CP15 c1 的位 0 为高来使能 MMU。

如果转换的地址不同于未转换的地址时必须注意，因为跟在使能 MMU 之后的一些指令可能在 MMU 关闭（VA=MVA=PA）时被预取。

在这种情况下，使能 MMU 可以被认为是带有延时执行的分支语句。当 MMU 被禁能时也会发生相似的情况。可以考虑下面的代码序列：

```
MRC     p15, 0, R1, c1, C0, 0       ; Read control register
ORR     R1, #0x1                    ; Set M bit
MCR     p15, 0,R1,C1, C0,0          ; Write control register and enable MMU
Fetch Flat
Fetch Flat
Fetch Translated
```

注意：因为同一个寄存器 CP15 c1 控制着 ICache、DCache 和 MMU 的使能，所有这三个可以通过一个 MCR 指令来使能。

3.6.2 禁能MMU

要禁能 MMU，清除 CP15 c1 的位 0。

注意：如果MMU是使能的，之后禁能，随后又重新使能，TLB中的内容是被保护的。如果这些TLB现在失效了，那么这些TLB必须在重新使能MMU之前被清除。参见TLB操作寄存器c8。

3.7 TLB结构

MMU 包含一个统一的 TLB，用于数据访问和指令读取。TLB 被分为两个部分：

- 一个八条目全相连部分，专用于保持锁定的 TLB 条目；
- 一个组相连的部分用于所有其他的条目：2 路×32 个条目。

当一个条目被写入TLB时，条目被放置到TLB的组相连还是锁定部分取决于TLB锁定寄存器的状态。参见TLB锁定寄存器c10。

当一个条目被写入到 TLB 的锁定部分，它只能通过直接地覆盖写来移除，或通过一个基于 MVA 的 TLB 清除操作来移除，其中 MVA 地址和锁定条目匹配。

TLB 的组相连部分的结构并不组成 ARM926EJ-S 处理器编程模型的部分。不能做出组相连部分的结构、替代算法或条目的持续性的假定。特别地：

- 任何写入到 TLB 组相连部分的条目可以在任何时候被移除。TLB 的组相连部分必须被认为是一个转换/页表信息的临时 cache。必须在条目上放置无依赖性，不管条目位于或不在 TLB 的组相连部分，除非该条目已经存在于锁定 TLB 中。TLB 的组相连部分可以包含在页表中定义但与 TLB 失效期间被访问的地址值无关的条目；
- TLB 的组相连部分必须被认为是下层页表的一个 cache，其中内存一致性必须一直保持。如果一个一级描述符在主存中被修改，那么为了保证一致性必须使用一个清除 TLB 或清除 TLB 条目的操作来移除任何缓存的一级描述符的拷贝。不论一级描述符的类型（节、二级页表引用或错误）如何这是必须要求的；
- 如果对于一个给定的页任一子页许可不同，那么每个子页都分别对待。为了清除带子页许可的页相关的所有条目那么需要四个基于 MVA 的清除操作，每个子页一个。

第4章 Cache和写缓冲

本章描述了指令 Cache（Instruction Cache，ICache）、数据 Cache（Data Cache，DCache），和写缓冲（write buffer）。包含以下小节：

- 关于cache和写缓冲；
- 写缓冲；
- 使能cache；
- TCM和cache访问优先级；
- Cache MVA和组/路格式。

4.1 关于cache和写缓冲

ARM926EJ-S 处理器包括：

- 一个指令 Cache（ICache）；
- 一个数据 Cache（DCache）；
- 一个写缓冲。

cache 的大小可以从 4KB 到 128KB，以二的指数幂增加。

cache 有以下的特征：

- cache 是虚拟索引、虚拟标签，由修改过的虚拟地址（MVA）来寻址。这能够避免上下文切换时的 cache 清理和/或清除；
- cache 是四路（four-way）组相连的，每个 cache 行长度为 8 个字，即 32 字节每行，且 DCache 中有两个脏位；
- DCache 支持写通（write-through）和写回（write-back），或拷回（copyback），以及 cache 操作，由 MMU 转换表的 C 和 B 位确定的内存区域来选择；
- 支持在读未命中时的 cache 分配。cache 执行关键字（critical-word）优先的 cache 重填充；
- 伪随机和循环优先级（round-robin）替换，由 CP15 c1 的 RR 位来选择；
- cache 锁定寄存器能够控制哪一个 cache 总是行填充时的分配，提供一个用于锁定和控制 cache 污染的机制；
- DCache 存储的物理地址（PA）标签对应于标签 RAM 中的每个 DCache 条目，标签 RAM 在 cache 行写回期间使用，除此之外标签 RAM 中还存储着虚拟地址标签。这意味着 MMU 并不包含在 DCache 写回操作中，去除了与写回地址相关的 TLB 丢失的可能性；
- PLD 数据预加载指令并不引起数据 cache 的行填充。它被当做一个 NOP 指令；
- Cache 维护操作提供了高效的清除：
  - 整个 DCache 或 ICache；
  - DCache 或 ICache 区域；
  - 虚拟内存区域。

  Cache 的维护操作也提供了高效的清理和清除操作：
  - 整个 DCache；
  - DCache 区域；
  - 虚拟内存区域。

当小的代码变更发生时后者能够高效的维持 DCache 的一致性，例如用于自修改的代码和异常向量。

4.2 写缓冲

写缓冲用于所有到非高速缓存的、可缓冲的区域、写通区域和到写回区域的写未命中的写操作。DCache 中包含一个独立的缓冲，用于保持 cache 行回收或清理脏的 cache 行时写回的数据。

主写缓冲有一个 16 字的数据缓冲和 4 个地址缓冲。

DCache 写回缓冲有 8 个数据字条目和一个地址条目。

MCR 排空写缓冲指令能够让两个写缓冲在软件控制下都被排空。

MCR 等待中断导致两个写缓冲都被排空并且 ARM926EJ-S 处理器被置于一个低功耗状态直到中断发生。

表 4.4描述了写缓冲的行为。

对于在写缓冲中有相应待处理的写的读访问不会推进（读的）发生。对这类的访问，写缓冲被排空并且值从外部存储器读取。

4.3 使能cache

在复位时，ICache和DCache的条目均无效并且cache被禁能。读和写操作并不访问cache。使用CP15 c1 中的I、C和M位来使能cache，并且两者能独立与另一个被使能。表 4.1给出了对于ICache的I和M位设置，以及相关的行为。TCM和cache行为的优先级在TCM和cache访问优先级中描述。

表 4.1　CP15 c1 的 I 和 M 位关于 ICache 的设置

I 位	M 位	AMR926EJ-S 行为
0	-	ICache 禁能。所有指令读取都取自外部存储器（AHB）。
1	0	ICache 使能，MMU 禁能。所有指令读取都是高速缓存的，没有保护检查。所有地址都平面映射，也就是 VA=MVA=PA。
1	1	ICache使能，MMU使能。指令读取是高速缓存或非高速缓存的，取决于页描述符的C位（见表 4.2），且执行保护检查。所有地址从VA重映射到PA，取决于页条目，也就是VA被转换位MVA，而MVA被重映射到PA。

表 4.2给出了用于ICache的页表C位设置（CP15 c1 I位=M位=1）。

表 4.2　用于 ICache 的页表 C 位设置

C 位	描述	AMR926EJ-S 行为	
0	非高速缓存的	ICache 禁能。所有指令读取都取自外部存储器。	
1	高速缓存的	**Cache 命中**	从 ICache 中读取
		Cache 未命中	从外部存储器填充 cache 行

表 4.3给出了CP15 c1 的C和M位关于DCache的设置，以及相应的行为。

表 4.3　CP15 c1 的 C 和 M 位关于 DCache 的设置

C 位	M 位	ARM926EJ-S 行为
0	0	DCache 禁能。所有数据访问都到外部存储器。
1	0	DCache 使能，MMU 禁能。M 位的设置优于 C 位，因此 DCache 实际上被禁能。所有数据访问都是非高速缓存的、非缓冲的，且没有保护检查。所有地址都平面映射，也就是 VA=MVA=PA。
1	1	DCache使能，MMU使能。所有数据访问是高速缓存的或非高速缓存的，取决于页描述符的C和B位（见表 4.4），执行保护检查。所有地址都从VA重映射到PA，取决于MMU的页表条目，也就是VA被转换为MVA，而MVA被重映射到PA。

表 4.4给出了页表C和B位关于DCache的设置（CP15 c1 C位=M位=1），以及相关的行为。

表 4.4　页表 C 和 B 位关于 DCache 的设置

C 位	B 位	描述	ARM926EJ-S 行为
0	0	非高速缓存的、非缓冲的	DCache 禁能。读数据来自外部存储器。写数据作为一个到外部存储器的非缓冲的存储。DCache 不被更新。
0	1	非高速缓存的、可缓冲的	DCache 禁能。读数据来自外部存储器。写数据作为一个到外部存储器的缓冲的存储。DCache 不被更新。
1	0	写通	DCache 使能： **读命中** 从 DCache 读取数据 **读未命中** （cache）行填充 **写命中** 写数据到 DCache，并缓冲到外部存储器的存储 **写未命中** 缓冲到外部存储器的存储
1	1	写回	DCache 使能： **读命中** 从 DCache 读取数据 **读未命中** （cache）行填充 **写命中** 仅写数据到 DCache **写未命中** 缓冲到外部存储器的存储

4.4　TCM和cache访问优先级

应用于ARM926EJ-S处理器指令访问的优先级见表 4.5。ARM926EJ-S处理器赋予在指令TCM区域中的地址以最高的优先级。

表 4.5　到 TCM 和 cache 的指令访问优先级

地址在 ITCM 区域中	地址在 DTCM 区域中	页描述符中可高速缓存	ARM926EJ-S 行为
是	是	无关	访问 ITCM
是	否	可高速缓存的	
是	否	不可高速缓存的	
否	无关	可高速缓存的	访问 ICache
否	无关	不可高速缓存的	访问外部存储器

应用于ARM926EJ-S处理器数据访问的优先级见表4.6。TCM的哈佛体系与数据读写的cache需求能够以读和写操作访问指令TCM。表 4.6的列顺序有意和表 4.5中的指令访问一致。

表 4.6　到 TCM 和 cache 的数据访问优先级

地址在 ITCM 区域中	地址在 DTCM 区域中	页描述符中可高速缓存	ARM926EJ-S 行为
是	是	无关	访问 DTCM
否	是	可高速缓存的	
否	是	不可高速缓存的	
是	否	可高速缓存的	访问 ITCM
是	否	不可高速缓存的	
否	否	可高速缓存的	访问 DCache
否	否	不可高速缓存的	访问外部存储器

4.5　Cache MVA和组/路格式

本节展示了 ARM926EJ-S cache 的 MVA 和组/路格式是如何映射到一个通用虚拟索引、虚拟寻址的 cache。

图 4.1表示了一个通用的、虚拟索引的、虚拟寻址的cache。

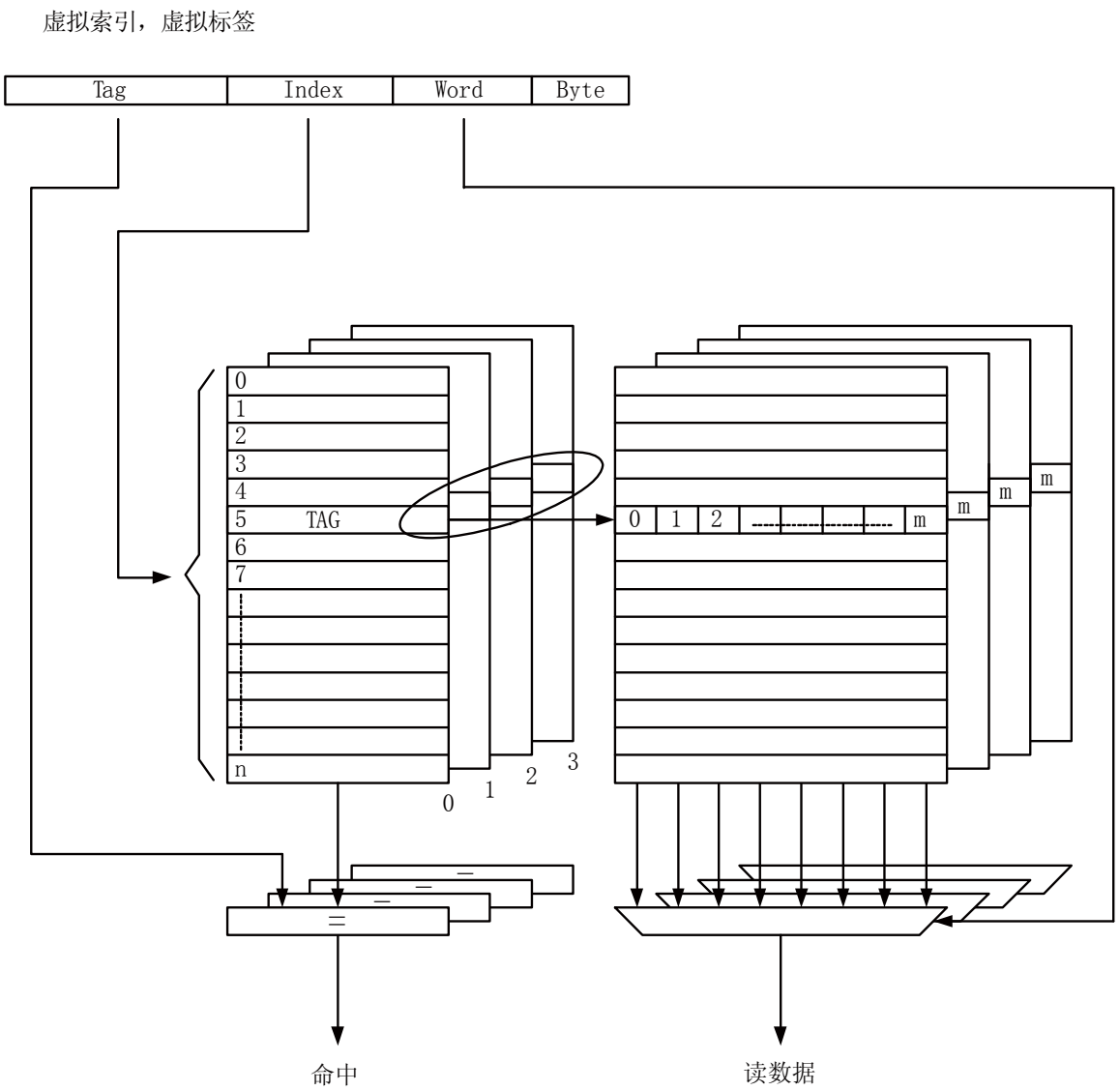


图 4.1　通用虚拟索引虚拟寻址 cache

图 4.2表示了ARM926EJ-S cache的格式。

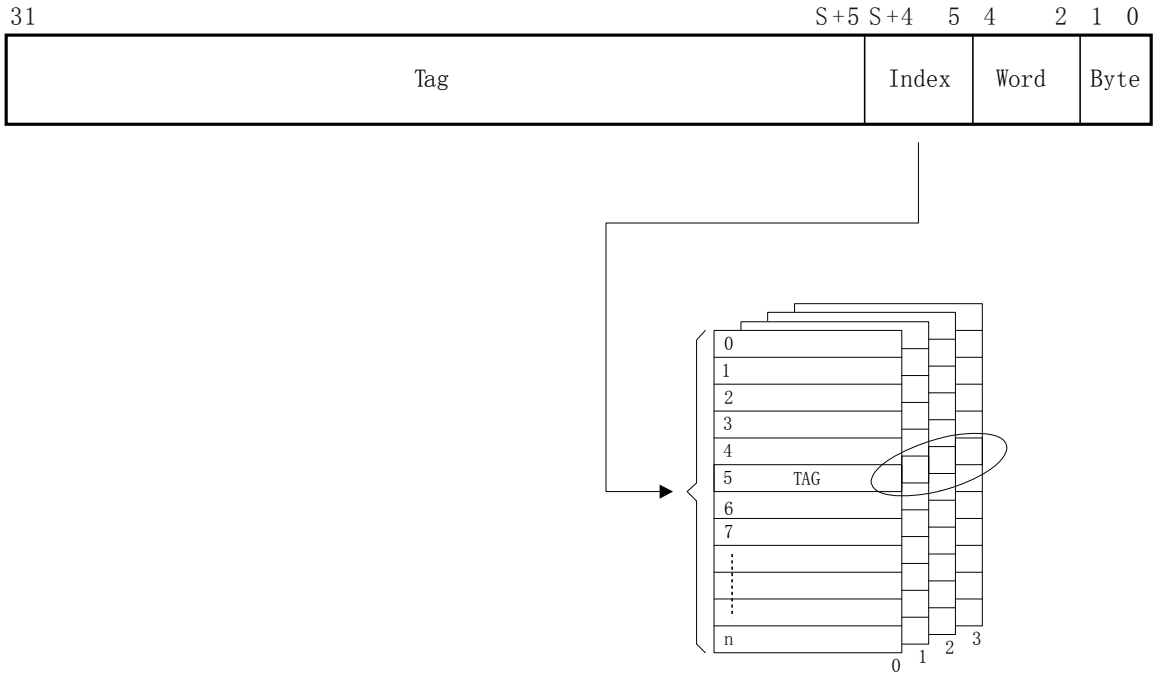


图 4.2　ARM926EJ-S cache 相连性

表 4.7表示了ARM926EJ-S cache的S和NSETS的值。

表 4.7　S 和 NSETS 的值

Cache 大小	S	NSETS
4KB	5	32
8KB	6	64
16KB	7	128
32KB	8	256
64KB	9	512
128KB	10	1024

图 4.2表示了ARM926EJ-S cache的相连性。在图 4.2中，应用了以下要点：

- 相同索引（Index）的标签集合定义了一个组（Set）；
- 在一个组中的标签数目即为相连性（Associativity）；
- ARM926EJ-S cache 是四路（four-way）相连的；
- 由索引寻址的标签的范围定义了一个路（Way）；
- 在一个路中标签的数目即为组的个数，NSET。

ARM926EJ-S cache的组/路/字格式见图 4.3。

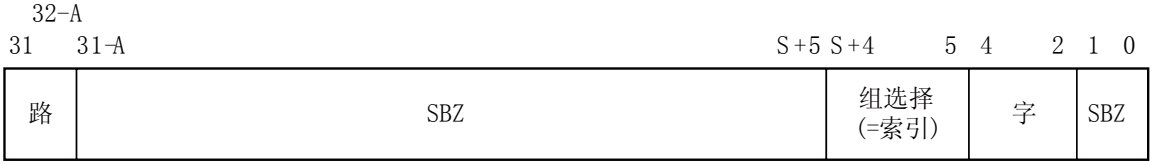


图 4.3　ARM926EJ-S cache 组/路/字格式

在图 4.3中：

A = $\log_2$Associativity。

例如，对于一个四路的 cache，A=2。

S = $\log_2$NSETS。

第5章 紧耦合存储器接口

本章描述了 ARM926EJ-S 紧耦合存储器（Tightly-Coupled Memory，TCM）接口。包含以下部分：

- 关于紧耦合存储器接口；
- TCM接口信号；
- TCM接口总线周期类型和时序；
- TCM编程模型；
- TCM接口示例；
- TCM访问缺陷；
- TCM写缓冲；
- 使用同步SRAM作为TCM存储器；
- TCM时钟门控。

5.1 关于紧耦合存储器接口

ARM926EJ-S 处理器能够使用紧耦合存储器（TCM）接口以低延时访问外部存储器。术语紧耦合存储器针对和 ARM9EJ-S CPU 内核之间的关系，以及存储器的操作，其中在 ARM9EJ-S 的指令和数据访问活动性与到外部存储器的访问模式之间有较强的关联。这是对比于到 AHB 接口访问的模式，到 AHB 接口的访问相应地从 ARM9EJ-S 内核去耦。

TCM是用于存储特定类型的关键代码或数据，其中要求低延时、确定的访问。TCM对于各种类型的代码或数据并不是必须的最佳选择，如果代码或数据表现出高度的空间性或临时局部性，通过使用cache存储器可以获取更好的性能。见第四章Cache和写缓冲。

ARM926EJ-S 处理器支持两个 TCM 区域，一个用于指令（ITCM）而另一个用于数据（DTCM）。ITCM 接口也可被 ARM9EJ-S 内核的数据端访问。这在代码被加载到 ITCM 中，SWI 和仿真指令处理程序，以及访问 PC 相关的文字池时是必须的。

TCM地址空间是物理寻址的，TCM区域在物理地址空间中的位置由TCM区域寄存器控制。见TCM区域寄存器c9。TCM区域的物理大小由外部输入（IRSIZE、DRSIZE）定义，范围从 4KB 到 1MB。这些引脚的编码见表 2.24。TCM区域在物理地址映射中可以被放置到任何位置，带有的限制是TCM的基址必须和TCM大小对齐，并且指令和数据TCM区域不能交叠。TCM区域大小可以通过用软件读TCM状态寄存器得到。见TCM状态寄存器c0。

INITRAM 引脚让 ARM926EJ-S 处理器能够在系统复位后从指令 TCM 空间启动。如果 INITRAM 在系统复位期间有效并且 VINITHI 引脚失效，那么 ARM926EJ-S 处理器从指令 TCM 接口的地址 0x0000 0000 处读取指令。如果 INITRAM 和 VINITHI 都有效，在复位之后第一条读取的指令是在 AHB 上的 0xFFFF 0000。

TCM 接口支持零或更多等待状态的存储器访问。支持零等待状态访问的要求对于 TCM 子系统设置强加了许多限制，这些限制在以普通总线接口（比如 AHB）连接存储器时并不应用。

由于时序的限制，在 TCM 接口上出现的读访问没有被 MMU 做优先级限定。这意味着在 TCM 接口上的所有读操作必须被视为推测性的，因而需阻止使用读敏感的存储器。TCM 接口包含两个条目写缓冲以避免等待（stall）周期的要求，这是因为（TCM 接口）和 ARM9EJ-S 原本的存储器接口不匹配，以及标准 SRAM 的要求。

TCM 访问可以通过使用 IRWAIT/DRWAIT 输入而延长以产生等待状态。然而，这两个信号和其他信号的时序表示能够实现的存储器子系统的类型是受限的。例如，要求一个地址译码来确定是否需要插入一个等待状态的方案如果在以最高频率运行时是不可能的。

DMA 访问可以通过使用 IRWAIT/DRWAIT 信号之一以在 DMA 访问期间插入等待状态来执行，或者是使用能够避免外部多样的关键接口信号需求的专用 DMA 接口，当使用单周期访问的存储器时。

5.2 TCM接口信号

TCM 接口被设计为兼容标准 ASIC SRAM 元件的时序，能够以最低的接口逻辑要求来连接单周期 SRAM。对于标准的 SRAM，片选、地址和写数据/控制信号在一个周期内建立，而读或写操作在下一个周期发生。

5.2.1 数据接口信号

DTCM 接口中的信号可以由功能分组为四类：

- 控制信号
  - DRCS
  - DRWAIT
  - DRIDLE
- 地址和属性信号
  - DRSEQ
  - DRADDR[17：0]
  - DRWBL[3：0]
  - DRnRW
- 数据信号
  - DRRD[31：0]
  - DRWD[31：0]
- DMA 信号
  - DRDMAEN
  - DRDMACS
  - DRDMAADDR[17：0]

控制信号

数据接口的控制信号是：

DRCS

DRCS 被用来表示访问将在接下来的周期开始。对于简单的零等待状态 TCM 系统而言 DRCS 信号直接对应为存储器的片选信号。对于较复杂的系统 DRCS 对应到一个存储器请求信号。

DRWAIT

DRWAIT 被用于通过插入等待状态来扩展 TCM 传输。DRWAIT 信号的时序超前数据传输发生的周期一个周期。这表示如果一个访问要被等待，DRWAIT 必须和 DRCS 生效相同的周期，且在数据传输发生前一个周期失效。

DRIDLE

DRIDLE 信号提供一个较早的表示：在当前周期没有 TCM 访问发生。

地址和属性信号

当 DRCS 断言和有效期间所有的地址和属性信号都有效，而 DRSEQ 例外，它在等待状态期间也有一个定义的值，这期间 DRCS 无效。

DRSEQ

当 DRCS 被断言且有效时，DRSEQ 表示用于当前 TCM 访问的地址是前一个访问的接续。在等待状态期间 DRSEQ 被强制位高电平。

DRADDR[17：0]

DRADDR 是用于传输的字（32 位）地址。

DRnWR

DRnWR 表示访问是读或写。

DRWBL[3：0]

DRWBL 是用来表示写访问时一个地址中的哪个字节必须被更新。这独立于地址，传输的大小一级当前的端结构设置。DRWBL 在读时为 b0000。

数据信号

数据信号是：

DDRD[31：0]

DDRD 是 TCM 返回的读数据。对零等待状态系统，DDRD 在 DRCS 之后的周期有效。对于有等待状态的系统，DDRD 在 DRWAIT 失效之后的周期有效。

DDWD[31：0]

DDWD 是写入到 TCM 中的写数据。它在和 DRCS 相同的周期内有效。

DMA 信号

DMA 接口让用户能够从一个到 ARM926EJ-S 处理器的外部源产生 DRADDR 和 DRCS 的值。

DRDMAEN

DRDMAEN 是 DMA 使能信号。当断言它时表示 DMA 值必须被用来产生 DRCS 和 DRADDR 而不是那些来自 ARM926EJ-S TCM 控制器内部的（信号）。

DRDMACS

DRDMACS被用来在DRDMAEN被断言时产生DRCS。由于DRDMACS信号的线路是与ARM926EJ-S内部的TCM控制器连接的，在DRDMACS没有断言的情况下不能有效产生DRDMAEN，除非内部TCM控制器为空闲状态（DRIDLE被断言）。这些信号的关系见表 5.1。

表 5.1　DRDMAEN、DRDMACS 和 DRIDLE 之间的关系

DRDMAEN	DRDMACS	DRIDLE	DRCS
1	1	0	1
1	0	0	未知
1	1	1	1
1	0	1	0

DRDMAADDR[17：0]

DRDMAADDR 被用作 DRADDR 的源，无论 DRDMAEN 是否被断言。

5.2.2 指令TCM信号

指令端 TCM 信号几乎和 DTCM 信号一样。DTCM 上所有的信号都在指令端有一个等价的信号。

- 控制信号
  - IRCS
  - IRWAIT
  - IRIDLE
- 地址和属性信号
  - IRSEQ
  - IRADDR[17：0]
  - IRWBL[3：0]
  - IRnWR
- 数据信号
  - IRRD[31：0]
  - IRWD[31：0]
- DMA 信号
  - IRDMAEN
  - IRDMACS
  - IRDMAADDR[17：0]

5.2.3 DTCM和ITCM的差异

DTCM 和 ITCM 接口之间存在差异：

- 禁止发生到 ITCM 的 DMA，除非 IRIDLE 被断言；
- 在 DTCM 上只有背靠背（back-to-back）的传输才能被标记为顺序的。在 ITCM 上，空闲周期可以在标记为顺序的请求之前出现；
- 在 ITCM 上不会出现顺序的写传输；
- ARM926EJ-S 处理器不支持通过 ITCM 和 DMA 的并发访问。DMA 访问必须在当用户明确 ARM926EJ-S 处理器不能从 ITCM 执行代码时才能发生。TCM 接口通过 IRIDLE 或 STANDBYWFI 信号来表示该事件。

5.3 TCM接口总线周期类型和时序

TCM总线接口是流水线的，能够背靠背地以零等待状态访问TCM存储器。对于每个TCM访问只有一个请求周期和一个或多个数据周期。图 5.1表示了一个多周期数据端TCM访问。

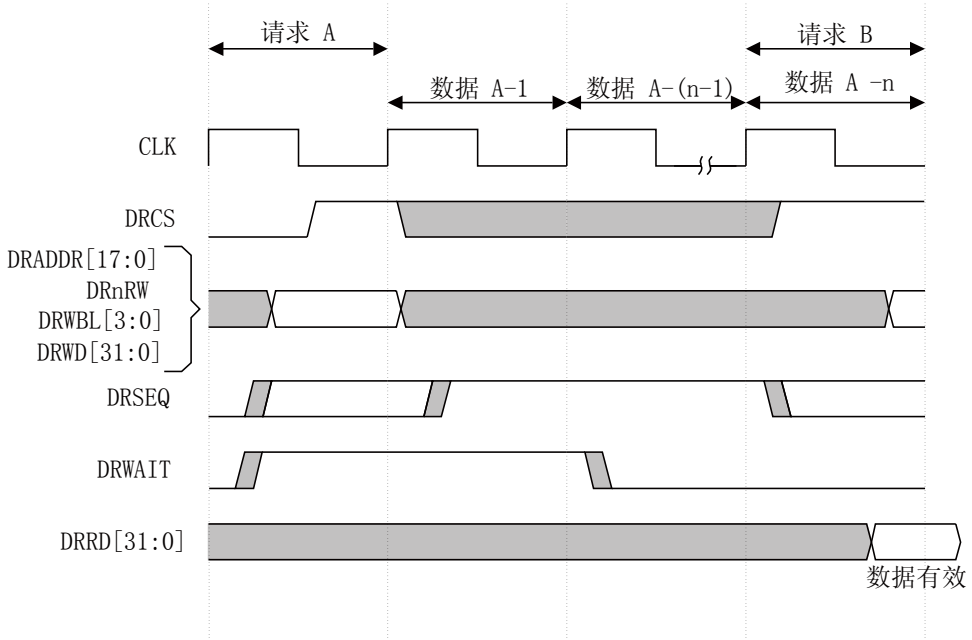


图 5.1　多周期数据端 TCM 访问

第一个周期是一个请求周期，即请求 A，这里所有的 TCM 接口输出信号都有效。TCM 子系统在 DRWAIT 上响应，表示访问将不会在接下来的周期内完成。跟在请求周期之后的周期，即数据 A-1，是第一个等待数据周期。在这个周期内 DRADDR、DRnWR、DRWBL 和 DRWD 的值不在有效并且它们的值是不确定的，而 DRSEQ 是断言的。由于在请求周期内 DRWAIT 表示访问是否会在接下来的周期内完成。在倒数第二个数据周期，即数据 A-n-1，DRWAIT 是失效的，表示访问将在下一个周期完成。如果访问的最后一个数据周期，即数据 A-n，是一个读操作则 DRRD 包含有效的读数据。由于接口的流水线特征，一个访问最后的数据周期可以和下一个访问的请求周期交叠。

5.3.1 零等待状态时序

对于零等待状态访问，TCM接口的时序符合一个标准SRAM元件的时序，而仅要求最少的接口逻辑。图 5.2表示了一个在ITCM接口上相应的指令读取操作的零等待状态访问的例子。所有的访问都是读操作。

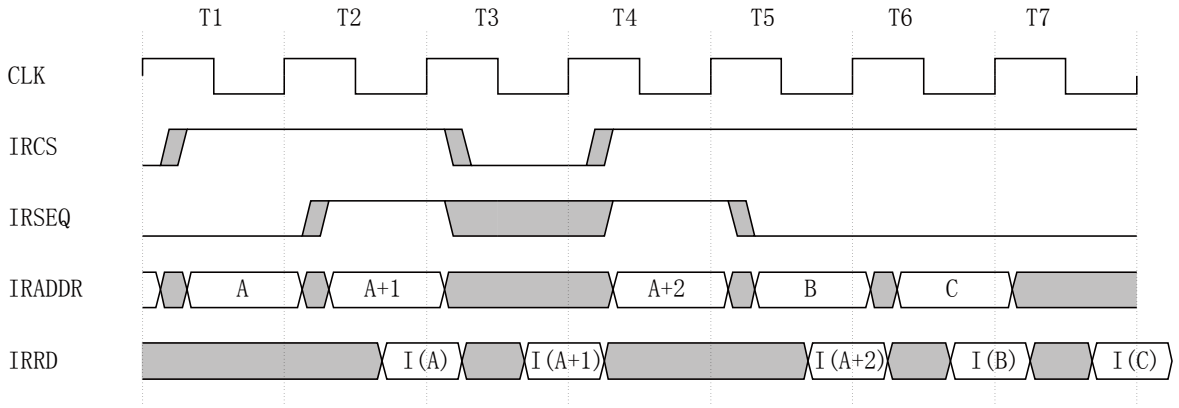


图 5.2　指令端零等待状态访问

在周期 T1 内，发出一个到地址 A 的非顺序请求。

在周期 T2 内，发出一个到地址 A+1 的顺序请求且到 A 访问的数据被返回。

在周期 T3 内，没有发出请求且到 A+1 访问的数据被返回。

在周期 T4 内，发出一个到 A+2 的顺序请求。

在周期 T5 内，发出一个到地址 B 的非顺序请求且到 A+2 访问的数据被返回。

在周期 T6 内，发出一个到地址 C 的非顺序请求且到 B 访问的数据被返回。

注意: 一个顺序请求周期的各个周期对于 ITCM 接口不必以连续的总线周期出现。在两个请求之间可以出现任意数目的空闲请求周期，只要（空闲请求之后的）第二个请求被标记为顺序的。DTCM 接口仅在连续的总线周期内产生顺序访问。

图 5.3表示了数据端零等待状态访问的例子。

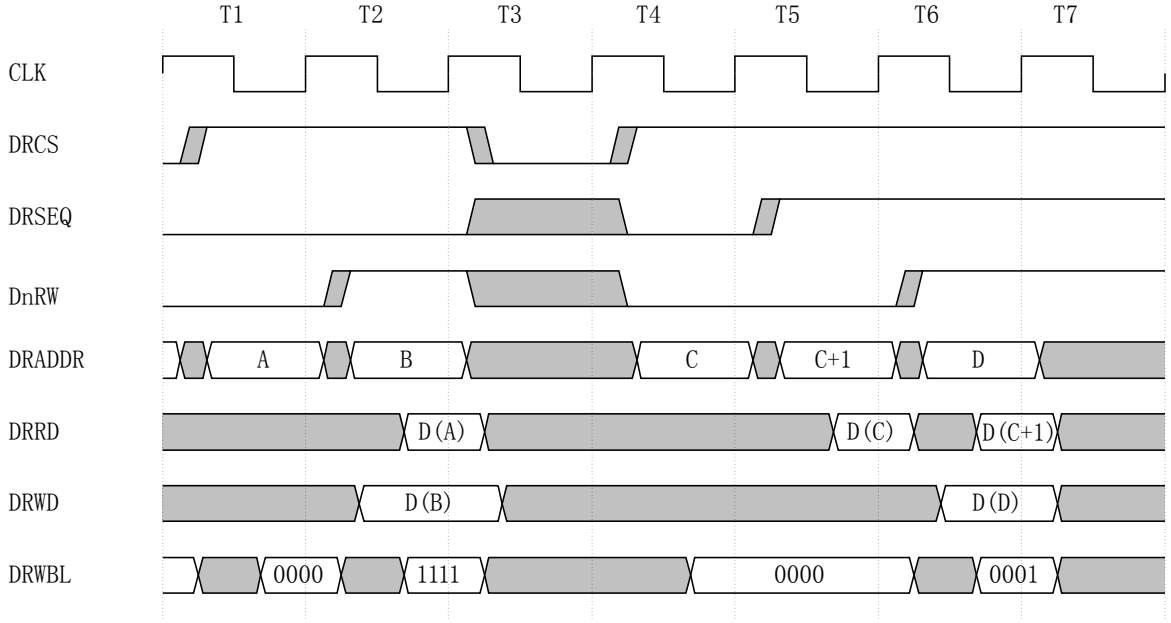


图 5.3　数据端零等待状态访问

在周期 T1 内，发出一个到地址 A 的非顺序读请求。

在周期 T2 内，发出一个到地址 B 的非顺序字写请求且到 A 访问的数据被返回。

在周期 T3 内，没有发出请求。

在周期 T4 内，发出一个到地址 C 的非顺序读请求。

在周期 T5 内，发出一个到地址 C+1 的顺序读请求且到 C 访问的数据被返回。

在周期 T6 内，发出一个到地址 D 的非顺序字节写请求。

5.3.2 到零等待状态TCM的DMA访问

对于到零等待状态存储器的DMA访问，可以使用TCM DMA接口。这能让另外一个地址源和片选传输到TCM存储器而不影响时序。图 5.4表示了DRDMAEN、DRDMACS、DRDMAADDR、DRADDR和DRCS之间的关系。

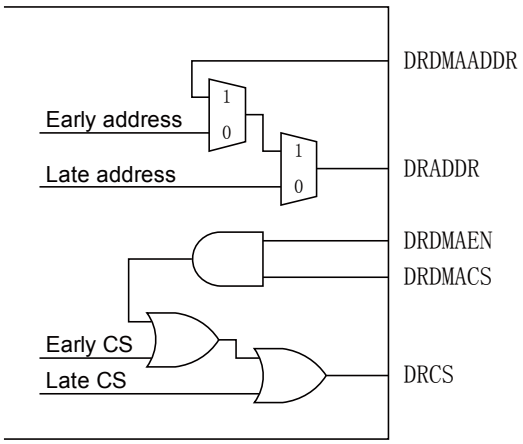


图 5.4　DRDMAEN、DRDMACS、DRDMAADDR、DRADDR 和 DRCS 之间的关系

在 ARM926EJ-S 处理器内部有多个地址和片选输出的源。TCM 接口的地址和片选输出是时序严格的（critical），然而并不是所有的内部源都是时序严格。通过将 DMA 输入和非严格的地址和片选信号相结合，可以完成 DMA 而不影响这些输出的时序。所有其他的 TCM 接口输出都不是时序严格的，可以多路复用到外部。

用于连接 DMA 片选和内部片选信号的逻辑被设计为如果 DMA 输入被选择，那么 DMA 片选也被断言。如果不是这种情况那么片选输出值是不确定的，除非已知 TCM 接口处于空闲状态，由 DRIDLE 和 STANDBYWFI 来表示。

图 5.5表示了DMA访问是如何与普通DTCM访问相结合的例子。

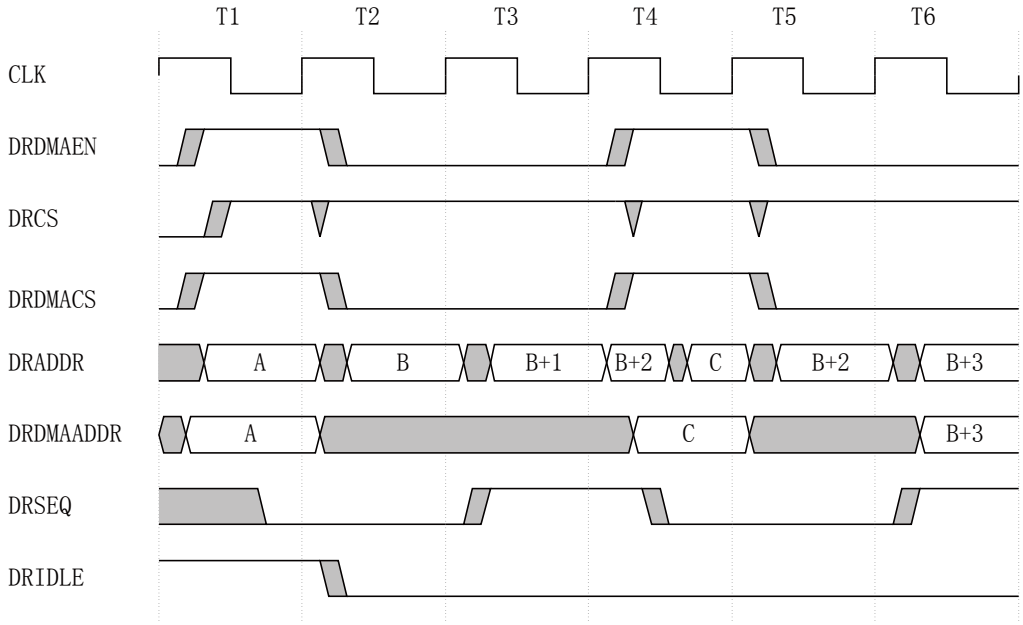


图 5.5　DMA 访问与普通 DTCM 访问相结合

在周期 T1 内，ARM926EJ-S 内部 TCM 控制器是空闲状态且 DRIDLE 被断言。DRDMAEN 也被断言，因此 DRDMAADDR 的值被传输到 DRADDR，并且 DRCS 被断言（DRDMACS=1）。DRSEQ 被强制为低电平。

在周期 T2 内，ARM926EJ-S 内部 TCM 控制器不再是空闲的，且 DRIDLE 被断言。发出一个到地址 B 的非顺序请求。

在周期 T3 内，发出一个到地址 B+1 的顺序请求且 DRSEQ 被断言。

在周期 T4 内，ARM926EJ-S 内部 TCM 控制器尝试输出对应到地址 B+2 顺序请求的值。DRDMAEN 被断言，且 DRADDR 和 DRSEQ 的值相应更改。ARM926EJ-S TCM 控制器被暂停。

在周期 T5 内，DRDMAEN 失效且 ARM926EJ-S TCM 控制器重新发出到地址 B+2 的请求。由于 DMA 访问的插入，DRSEQ 因为重复的请求而失效。

在周期 T6 内，发出到地址 B+3 的顺序请求且 DRSEQ 被重新断言。

使用 IRDMAEN、IRDMACS 和 IRDMAADDR 信号 DMA 访问可以用到 ITCM 中，但是和 DTCM 不一样，不支持 ARM926EJ-S 和 DMA 同时访问。这意味着 ITCM DMA 在从 ITCM 中执行代码时不能发生。

5.3.3 多周期访问时序

如果非零等待状态存储器用于 TCM，那么使用 DRWAIT/IRWAIT 信号来让 ARM926EJ-S 等待。数据周期的等待信息是流水化的因此 DRWAIT/IRWAIT 的值属于接下来的数据周期。这和第一个数据周期的的请求周期对应。如果没有有效的 TCM 访问那么 DRWAIT/IRWAIT 上的值被忽略。这让用户能够推测产生等待信号。

图 5.6表示了IRWAIT的推测产生是怎样用来对每个ITCM访问发出一个等待状态的。

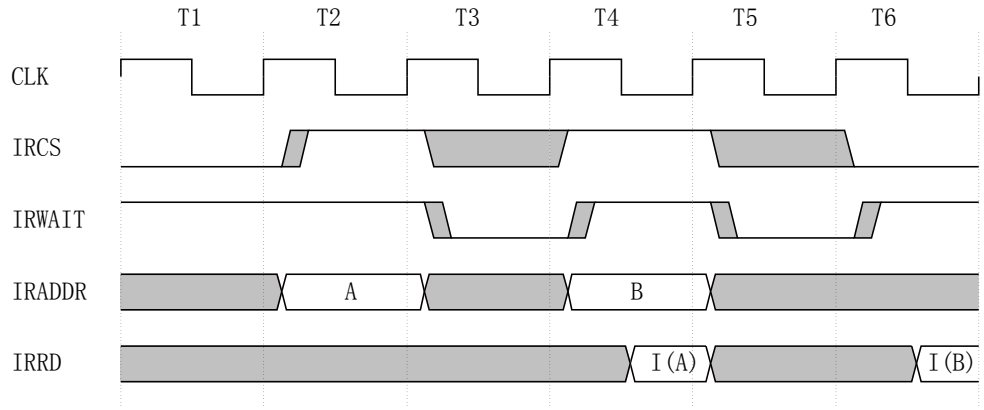


图 5.6　使用 IRWAIT 对 ITCM 访问产生一个等待状态

在周期 T1 内，IRWAIT 被断言但没有发出请求。

在周期 T2 内，IRWAIT 被断言且发出一个请求。

在周期 T3 内，IRWAIT 失效以表示到 A 的访问将在下一个周期完成。

在周期 T4 内，IRWAIT 被断言并发出一个请求。到 A 的访问完成。

在周期 T5 内，IRWAIT 失效以表示到 B 的访问将在下一个周期内完成。

在周期 T6 内，IRWAIT 被断言。没有发出请求，到 B 的访问完成。

在图 5.6中表示的例子所需要的逻辑对应为两状态的状态机，见图 5.7。

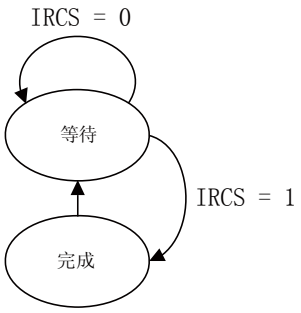


图 5.7　产生一个等待状态的状态机

在等待状态中时 IRWAIT 被断言。在完成状态时 IRWAIT 失效。

某些特定类型的存储器有不同的访问缺陷，取决于访问是顺序的还是非顺序的。IRSEQ/DRSEQ信号表示在访问的请求周期内该访问是否是顺序的，并且在等待周期内被保持为高电平。这个行为能够形成回环安排，其中SEQ输出可以通过一个反相器直接反馈到WAIT输入，以对非连续的访问产生一个单周期等待状态，见图 5.8。

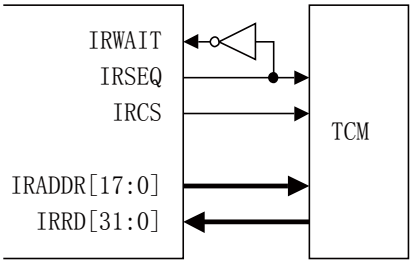


图 5.8　SEQ 的回环以产生一个单周期的等待状态

图 5.8中表示的电路的周期时序见图 5.9。

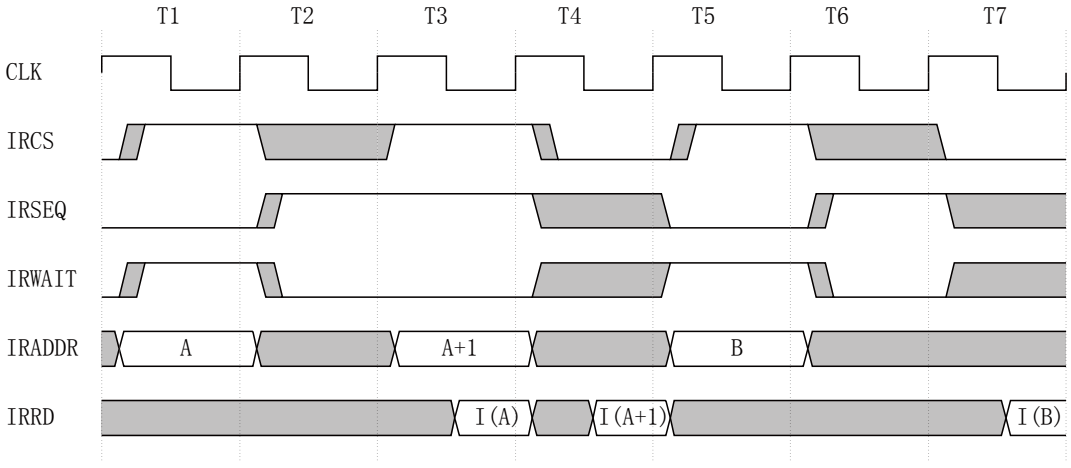


图 5.9　回环电路的周期时序

在周期 T1 内，发出一个到地址 A 的非顺序请求且 IRWAIT 被断言。

在周期 T2 内，由于等待状态 IRSEQ 被断言。IRWAIT 失效。IRCS 是未知的。

在周期 T3 内，到 A 的访问完成且发出一个到 A+1 的顺序请求。IRSEQ 为高电平而 IRWAIT 为低电平。

在周期 T4 内，到 A+1 的访问完成。没有发出新的请求。IRSEQ 和 IRWAIT 的值是未知的。

在周期 T5 内，发出一个到地址 B 的非顺序请求且 IRWAIT 被断言。

在周期 T6 内，由于等待状态 IRSEQ 被断言。IRWAIT 失效，IRCS 是未知的。

在周期 T7 内，到 B 的访问完成。

对于也要求到零等待状态存储器的 DMA 访问的系统，WAIT 信号被用来在等待状态和 DMA 仲裁上暂停 ARM926EJ-S 处理器。执行一个访问所要求的信息仅在该访问的请求周期内有效。如果一个 TCM 访问由于 DMA 而被延期，则访问请求信息必须在请求周期的末尾被捕获。

图 5.10表示了一个要求到存储器的DMA访问的系统的例子，该存储器对于非顺序访问有一个等待状态。

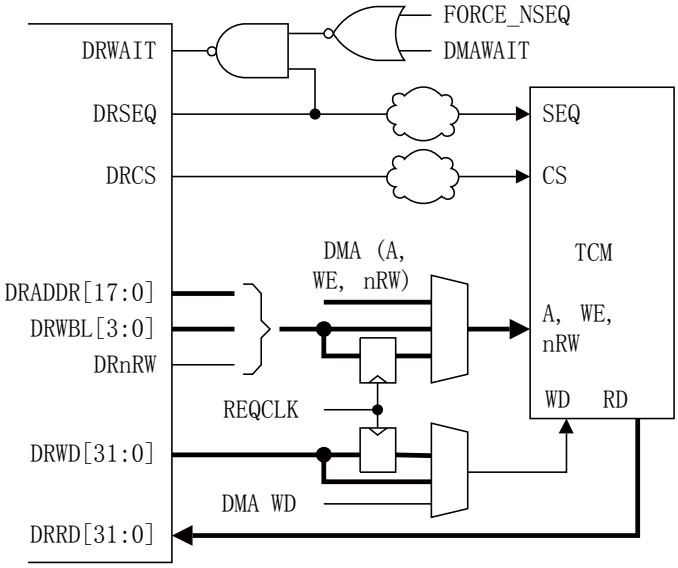


图 5.10 对非顺序访问有单等待状态的 DMA

用于产生 DRWAIT 的逻辑使用了用 DRSEQ 对非顺序请求插入一个等待状态的回环方案，以及一个附加的信号 DMAWAIT，用来在 DMA 访问期间暂停（CPU）。FORCE_NSEQ 信号是一个强迫信号，用来迫使 ARM926EJ-S 的访问由于 DMA 访问的插入而被视作非顺序的。

输入到 TCM 的 A、WE、nWR 和 WD，是直接源自 ARM926EJ-S TCM 接口、DMA 控制器或捕获寄存器，（它们）由 REQCLK 提供时钟，只要 ARM926EJ-S 访问由于 DMA 的有效性而被延期。

图 5.10中表示的电路的周期时序见图 5.11。

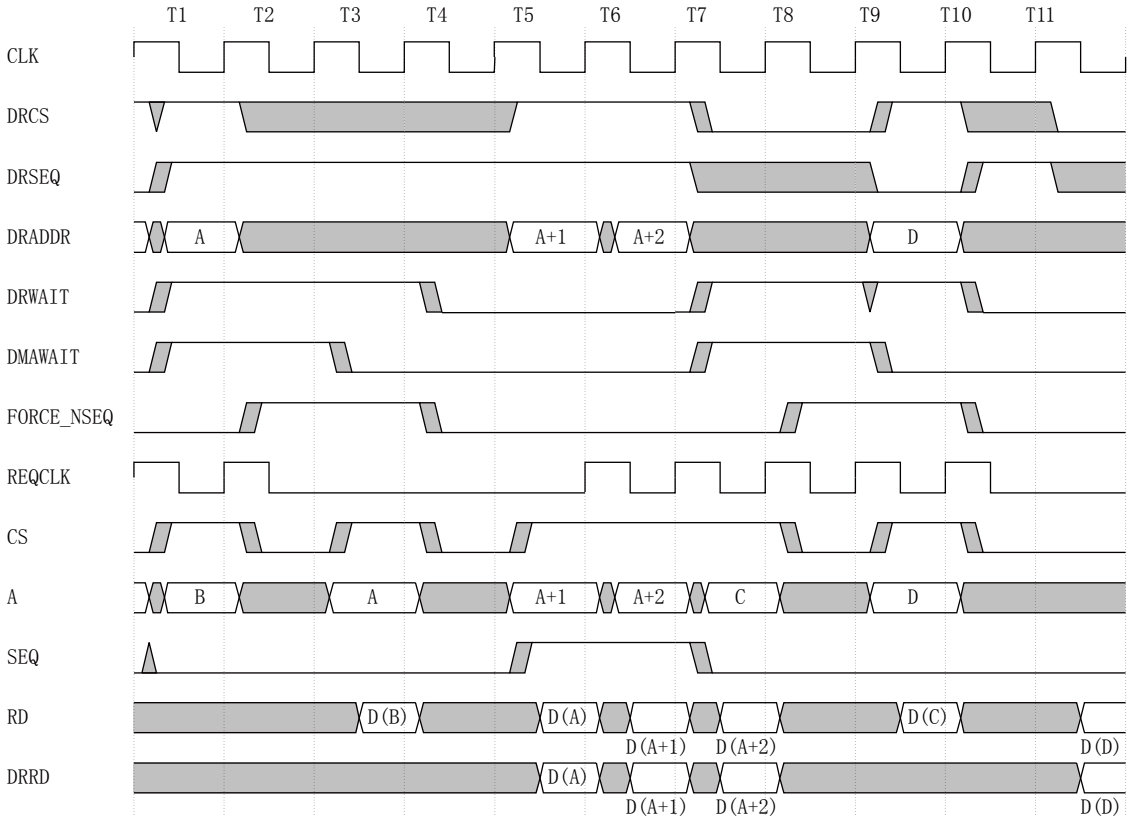


图 5.11　单等待状态的非顺序访问和 DMA 的电路的周期时序

在周期 T1 内，ARM926EJ-S 初始化一个到地址 A 的顺序请求，而 DMA 获取了 TCM 的占有权。DRWAIT 因 DMAWAIT 而被断言。用于 TCM 的 CS、A、WE 信号源自 DMA。DRADDR、DRWBL 和 DnWR 的值被存入寄存器。

在周期 T2 内，DMA 访问仍然有效（两周期的非顺序访问）。DRWAIT 因 DMAWAIT 而保持为高电平。

在周期 T3 内，DMA 访问完成且 DMAWAIT 失效。在 T1 末尾被捕获到的访问属性被用来产生用于 TCM 的 CS、A 和 WE 信号。DRWAIT 因 FORCE_NSEQ 而被断言。

在周期 T4 内，FORCE_NSEQ 失效导致 DRWAIT 失效，表示访问将在下一个周期完成。

在周期 T5 内，到 A 的访问完成。发出一个到 A+1 的顺序请求。没有 DMA 的活动。

在周期 T6 内，到 A+1 的访问完成。发出一个到 A+2 的顺序请求。没有 DMA 的活动。

在周期 T7 内，到 A+2 的访问完成。没有发出请求且 DRCS 失效。启动一个到地址 C 的 DMA 访问且用 DMAWAIT 使 DRWAIT 被断言。

在周期 T8 内，DRWAIT 由于 DMA 访问保持高电平。没有发出请求，DRCS 保持为低电平。

在周期 T9 内，到 C 的 DMA 访问完成。发出一个到地址 D 的非顺序请求。

5.4　TCM编程模型

在复位之后，TCM 的行为由 TCM 区域寄存器 CP15 c9 的状态控制。

5.4.1 使能ITCM

ITCM 可以在复位后使用 INITRAM 引脚自动使能。如果 INITRAM 在系统复位期间保持为高电平，且 VINITHI 引脚是无效的，则 ITCM 被使能且 ITCM 区域基址设为 0x0。这让用户能够从 ITCM 运行引导代码。用于这个目的时引导代码必须被预加载到 TCM 内。

如果INITRAM在系统复位期间为低电平则ITCM被禁能，ITCM可以通过写ITCM区域寄存器来使能。见TCM区域寄存器c9。

注意：如果 INITRAM = 1 且 VINITHI = 1，ITCM 在系统复位时被使能但是 ARM926EJ-S 处理器从 0xFFFF0000 开始引导。

5.4.2 使能DTCM

和ITCM不同，在复位时没有方法来自动使能DTCM。DTCM只能通过写DTCM区域寄存器来使能。见TCM区域寄存器c9。

5.4.3 禁能ITCM

通过清零 ITCM 区域寄存器，CP15 c9 的位 0 来禁能 ITCM。该寄存器必须通过一个读修改写操作来写入。

5.4.4 禁能DTCM

通过清零 DTCM 区域寄存器，CP15 c9 的位 0 来禁能 DTCM。该寄存器必须通过一个读修改写操作来写入。

5.4.5 可高速缓存和可缓冲的属性

用来映射 TCM 地址空间的所有 MMU 页表条目必须被标记为非高速缓存的。这是前向兼容性的要求。

注意:

- 如果 HRESETn 在一个热复位期间被异步断言，用户必须确保所有到 TCM 的写操作都已经完成且 TCM 接口在 HRESETn 被断言时是空闲的。这确保了写缓冲中没有待处理的写（写缓冲中的数据在热复位事件中可能会丢失），且阻止了由 TCM 接口上正存在的写而导致的丢失（数据）或 TCM 内容的破坏（corruption）。
- 为达到上述要求，用户必须:
  1. 执行一个排空写缓冲指令
  2. 进入待命（standby）WFI 模式
  3. 断言 HRESETn
- 联系 ARM 有限公司以获取更多细节。

5.5 TCM接口示例

这一节包含下面的示例：

- 零等待状态RAM示例；
- 用字可写的RAM生成字节可写的存储器；
- 多组RAM示例。

注意：本节中的大多数示例是用于 DTCM 接口的。这些示例也可应用于 ITCM 接口。

在本节中实现这些示例所需要的附加逻辑电路由实现者负责。

5.5.1 零等待状态RAM示例

图 5.12表示了最简单的RAM接口，其中RAM模块由单个字宽且有字节写控制的RAM构成。TCM接口可以直接连接到RAM模块。这是一个零等待状态的存储器因此DRWAIT被连接到低电平。

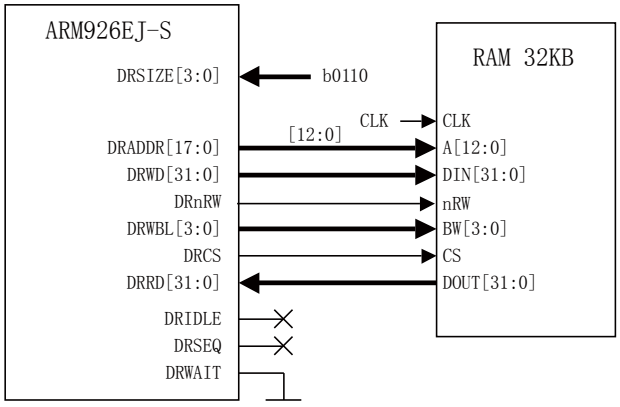


图 5.12　零等待状态 RAM 示例

5.5.2 用字可写的RAM生成字节可写的存储器

如果没有字节可写的RAM，必须使用如图 5.13所示的四组字节宽度的RAM。

连接这四个 RAM 模块的规则如下：

- 每个字节宽度的 RAM 有和字宽度 RAM 相同的地址和片选控制。
- 必须使用下面的连接：
  - DRWBL[0]，DRWD[7：0]，和 DRRD[7：0]连接到 RAM 的字节 0；
  - DRWBL[1]，DRWD[15：8]，和 DRRD[15：8]连接到 RAM 的字节 1；
  - DRWBL[2]，DRWD[23：16]，和 DRRD[23：16]连接到 RAM 的字节 2；
  - DRWBL[1]，DRWD[31：24]，和 DRRD[31：24]连接到 RAM 的字节 3；

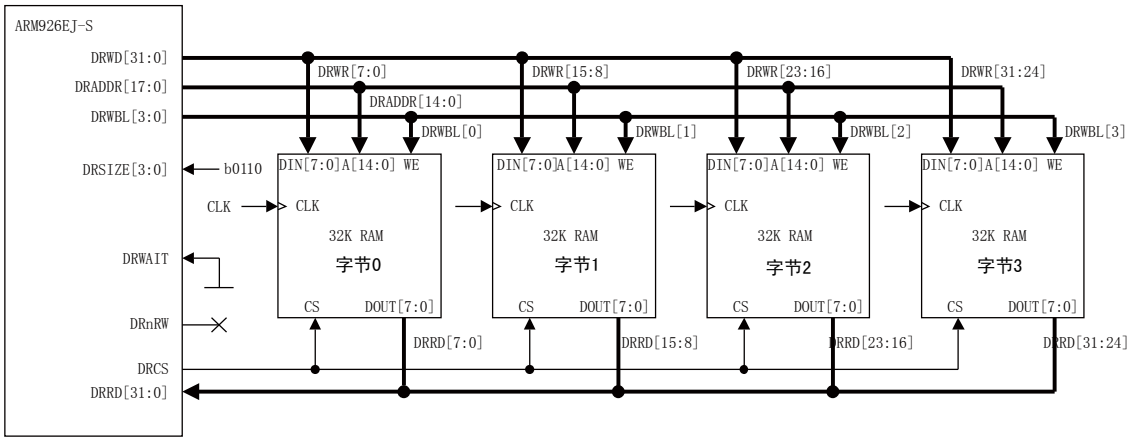


图 5.13　字节组的 RAM 示例

注意：在小端模式，DRWBL[0]表示一个字的 LSB 而 DRWBL[3]表示 MSB。在大端模式，DRWBL[3]表示字的 LSB 而 DRWBL[0]表示 MSB。

5.5.3 多组RAM示例

如果用户必须用小的 RAM 模块来创建一个大的存储器，有两种方法来实现：

- 如果最低功率消耗比一个快速的设计更重要，用户必须使用功率优化的示例；
- 如果一个快速的设计比最低功率消耗更重要，用户必须使用速度优化的示例。

由小的 RAM 模块生成存储器的规则是：

- RAM 模块数 b 必须是偶数的，例如 b = 2，4，8；
- 每个 RAM 模块必须是相同大小的；
- 如果需求的存储器的地址宽度大小是 n 位，则小的 RAM 模块的地址端口是 m = n – (logb/log2)位宽的；
- 地址位[m-1：0]被提供到所有的 RAM 模块；
- 地址位[n-1：m]在功率优化的解决方案中由 DRCS 来门控，或在速度优化的解决方案中由 IRnRW 来门控；
- 流水化的地址位[n-1：m]被用来选择正确的 RAM 读数据。

功率优化

图 5.14表示了如果用户要优化功率时一个大的存储器是如何从两个小的RAM模块生成的。对每个RAM模块需要独立的片选控制：

CS_bank0 = ~DRADDR[14] & DRCS

CS_bank1 = DRADDR[14] & DRCS

这确保了只有正被访问的 RAM 是使能的，并最小化功率消耗。

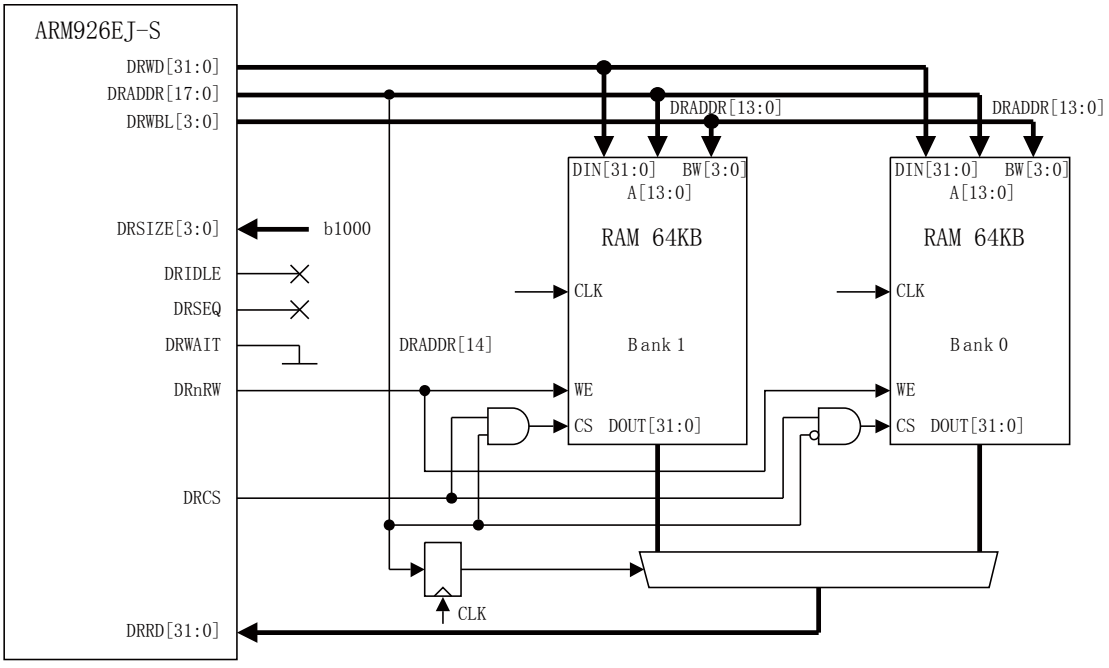


图 5.14 功率优化

速度优化

图 5.15表示了如果用户要优化速度时一个大的存储器是如何从两个小的RAM模块生成的。对每个RAM模块需要独立的写使能控制：

WE_bank0 = ~DRADDR[14] & DRnRW

WE_bank1 = DRADDR[14] & DRnRW

没有添加逻辑到关键的 DRCS 通路，但只要 DRCS 被断言则两个 RAM 都被使能，这导致较高的功率消耗。

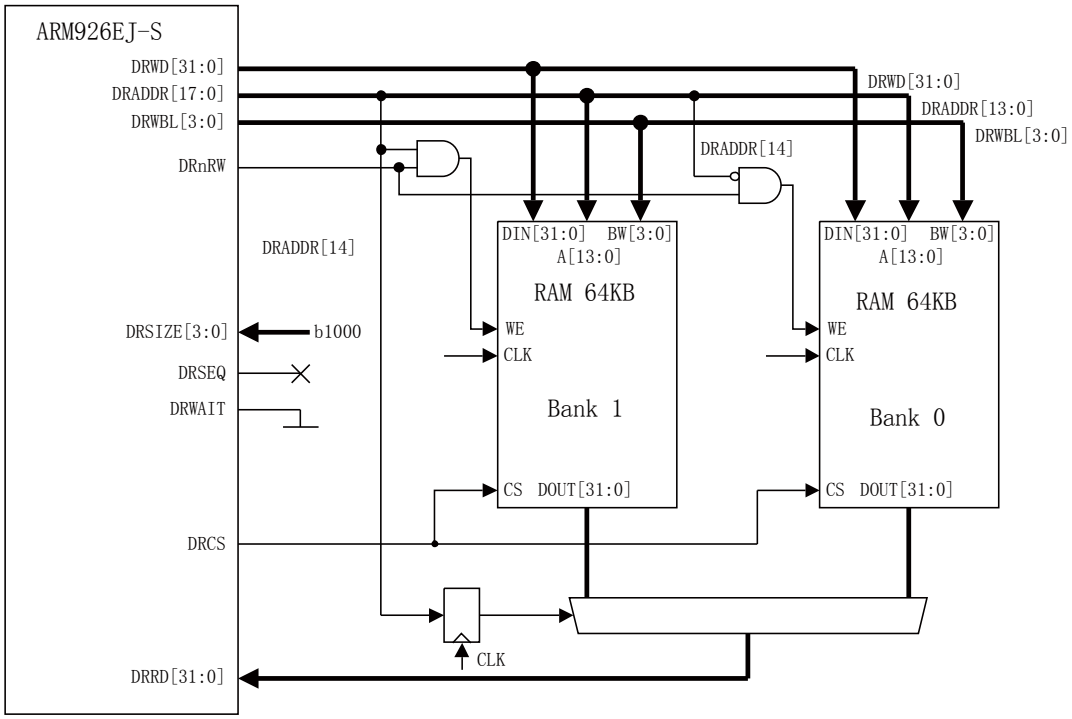


图 5.15　速度优化

5.5.4 顺序ROM示例

图 5.16表示了一个对非顺序访问使用等待状态的TCM子系统的例子。ROM中用来保持指令（的逻辑）可以和与之接口的ARM926EJ-S处理器以相同的频率周期运行。然而，ROM的存储器访问时间，从片选/地址到数据输出的时间，比直接与它接口的ARM926EJ-S处理器是不够快的。

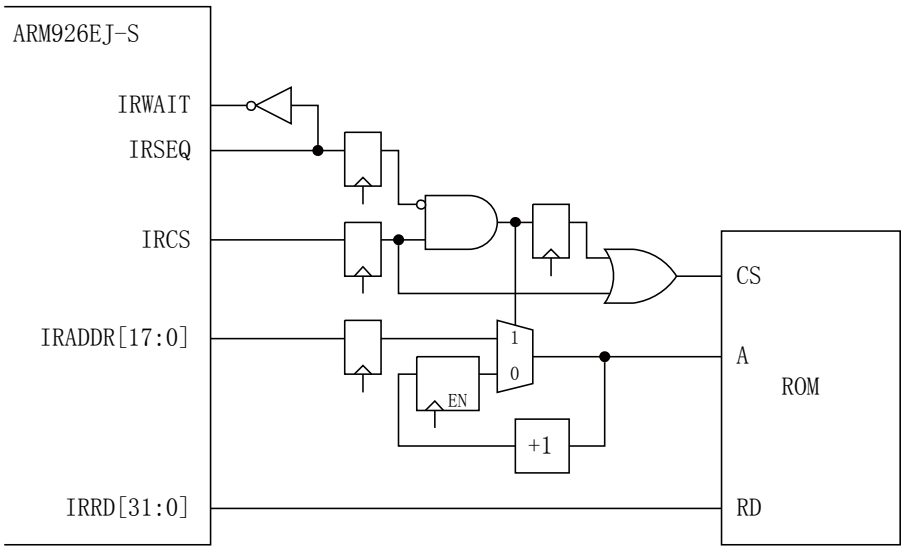


图 5.16　对非顺序访问使用等待状态的 TCM 子系统

输入到 ROM 的地址和片选是流水化的并遵守 ARM926EJ-S TCM 接口的输出。使用一个地址增量器来产生顺序的地址。当 ROM CS 片选有效时增量器的输出在每个周期的末尾被捕获。ROM 的地址源在一个非顺序的访问发生时切换到已寄存的 IRADDR 版本上。

图 5.17表示了与ARM926EJ-S TCM接口信号相关的ROM地址、片选和读数据的时序。提供给ROM的地址可以滞后、同步或超前于IRADDR，取决于存储器访问的类型和空闲周期的存在。

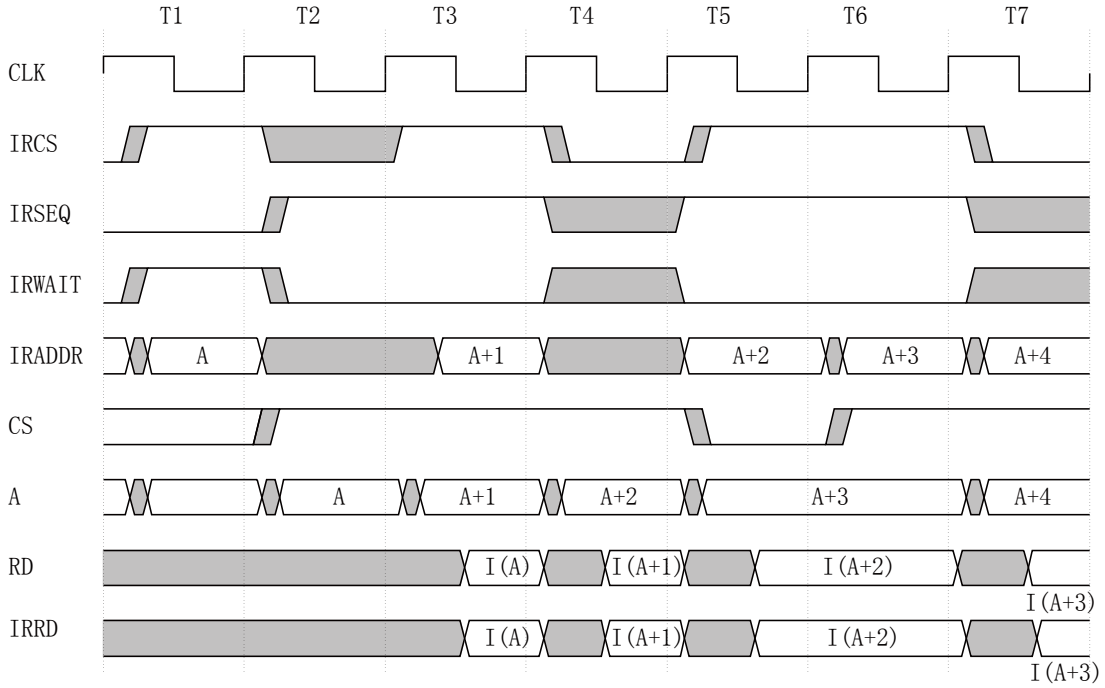


图 5.17　用于非顺序访问等待状态电路的周期时序

5.5.5 DMA接口示例

图 5.18使用DMA接口的TCM子系统的示例。驱动DRDMAEN的信号被连接到DRDMAEN和DRDMACS输入上。它也被用于控制时序不关键信号（WBL，nRW和WD）的多路复用，尽管这没有清楚表示出来。

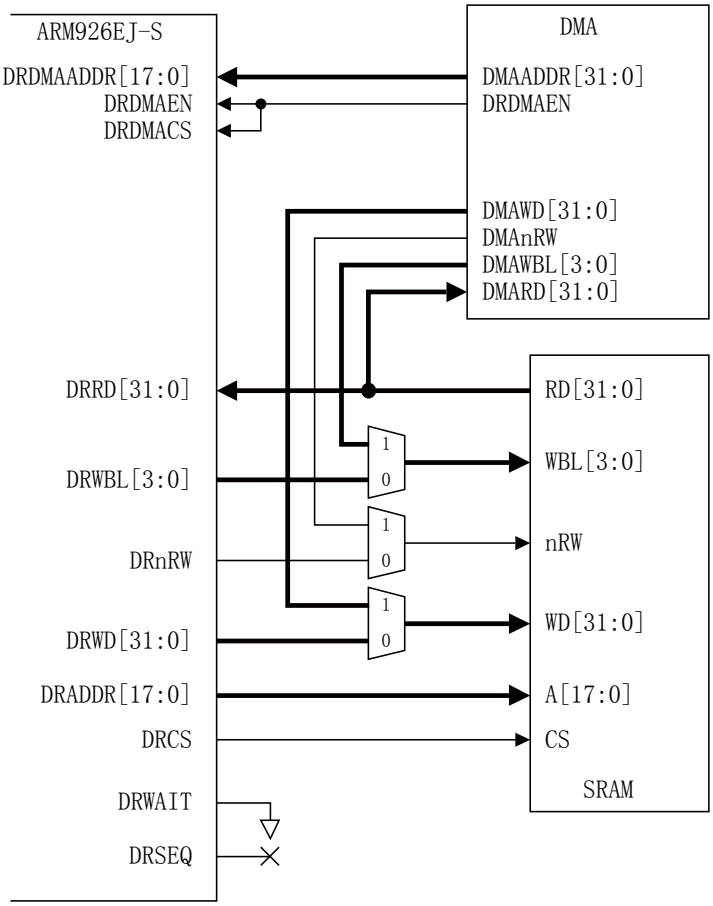


图 5.18　使用 DMA 接口的 TCM 子系统

5.5.6 集成RAM测试逻辑

用于实现TCM的存储器可能需要一些形式的测试访问，典型的是通过一个BIST控制器。通常通过在存储器的输入周围添加一圈的多路复用器。然而，这种方式对片选和地址信号引入了不需要的延时。这可以通过使用DMA接口来执行地址和片选值的复用的方式来避免。这表示在图 5.19中。

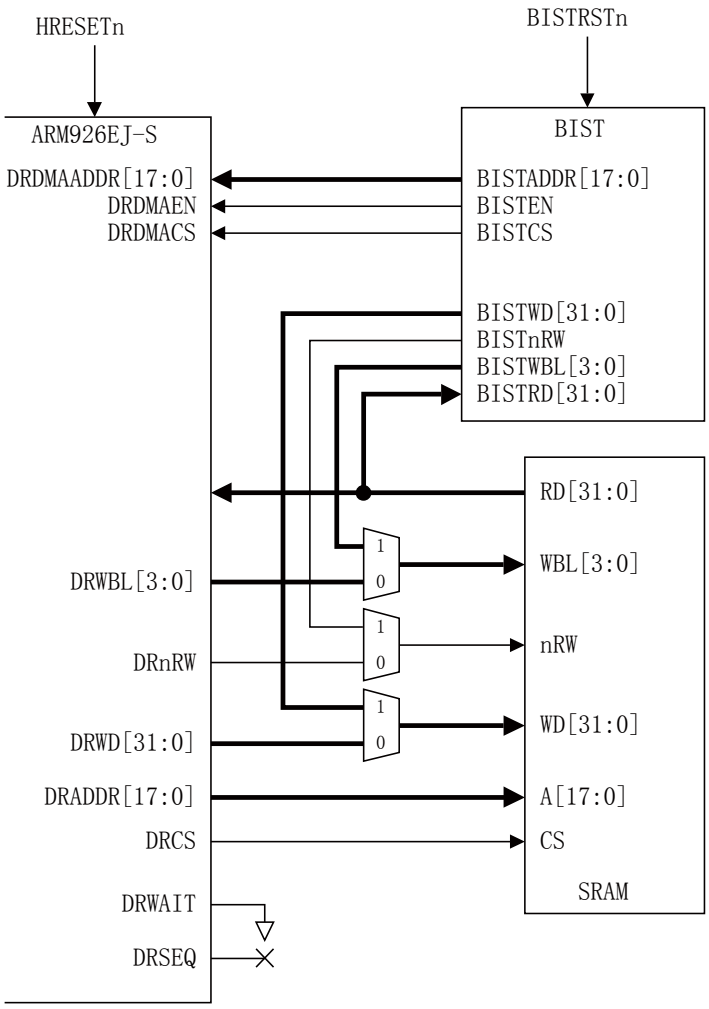


图 5.19　使用 BIST 的 TCM 测试访问

这类似于前面的 DMA 示例。然而，对 BIST 测试而言 BIST 控制器必须能够强制存储器片选为高电平和低电平。这种要求意味着必须保持 ARM926EJ-S 内核处于内部的片选值保证为低电平的状态。这可以通过保持 ARM926EJ-S 处于复位中来完成，HRESETn 在 TCM 存储器的 BIST 测试中为低电平。

注意：HRESETn 不能也用作 BIST 控制器的复位控制。

5.6　TCM访问缺陷

ARM926EJ-S 内核的数据端能够访问 ITCM。为了最大化 ITCM 的性能，到 ITCM 的数据读访问是流水化的。ARM926EJ-S 内核被暂停两个周期以让流水线完成读操作。这是 ARM926EJ-S TCM 接口唯一的暂停情景。TCM 控制器内的写缓冲排除了所有其他的对零等待状态 TCM 的潜在暂停源。

5.7　TCM写缓冲

每个 TCM 接口有一个两字条目的写缓冲。这是去除流水化的（de-pipeline）由 ARM9EJ-S 内核产生的地址和数据值所要求的，因此可对具有 SRAM 特征的存储器发出非推测性的（non-speculative）写操作，执行（写）不会引入暂停周期。

ARM9EJ-S 内核读请求占据高于写的优先级，因此 TCM 事务会因指令的执行而次序颠倒。如果读访问出现在在写缓冲内有一个对应条目的位置，那么数据从写缓冲中传回。如果确保在 TCM 接口上所有的未完成的写是必须的，那么可以使用 CP15 排空写缓冲指令（MCR p15, 0, Rd, c7, c10, 4）。这条指令在所有待处理的缓冲写、TCM 和 AHB 完成之前是不会完成执行的。

要保证 TCM 写缓冲被排空且在 TCM 接口上所有待处理的请求已完成，必须在禁止任何一个 TCM 区域之前使用一个排空写缓冲的指令。

5.8 使用同步SRAM作为TCM存储器

如果用户使用 SRAM 来实现 TCM 存储器，那么用户的 RAM 库必须满足下面的要求：

- RAM 必须是同步的。所有的时序必须关联到时钟上升沿；
- RAM 必须有一个片选，RAM 使能；
- RAM 输出必须总是有效。输出不能为三态；
- 要求字节写控制；
- RAM（信号）建立时间必须小于 10 – 15%且访问时间必须小于目标周期时间的 40 – 50%。违反这些要求将导致低速的设计。建立和访问时间可以通过滞后到 RAM 的时钟来平衡。

理想情况下每个 TCM 可以从单个 RAM 模块创建。然而，由于下面的原因这并不总是可能的：

- 如果用户的RAM没有字节写控制，用户必须用四个字节宽的RAM构建字宽的RAM。见用字可写的RAM生成字节可写的存储器；
- 如果用户的编译器不能生成所需大小的单个RAM模块，或单个RAM模块不满足时序要求。在这些情况下，用户必须由两个或更多的小RAM模块来生成RAM。见多组RAM示例。

理想情况下，用户的 RAM 模块可以直接连到 ARM926EJ-S TCM 接口。然而，这不总是可能的，并且在下面的情况下需要附加的逻辑：

- 所有 TCM 信号都被驱动为高电平有效。如果用户的 RAM 要求低电平信号，用户必须添加反相器以创建低电平有效的信号；
- 如果需要功率控制逻辑；
- 如果 RAM 是非单周期的，或需要硬件 DMA 仲裁，需要逻辑来驱动相应的等待信号。

注意：DRADDR 总是一个字地址。DRWBL 被用作一个字节通路门控来选择在写操作时被寻址的字上对应的字节。如总是字宽度的。

5.9 TCM时钟门控

如果 ARM926EJ-S 处理器当前并不从一个 TCM 区域运行代码，该 TCM 的空闲信号（对 DTCM 为 DRIDLE，对 ITCM 为 IRIDLE）被断言。这表示在该周期不执行 TCM 访问，让用户能够停止 TCM 时钟。如果没有时钟停止的要求，用户可以忽略空闲信号。

如果用户要求更严格的功率控制，用户也可使用空闲信号来禁能对 RAM 的供电。移除 RAM 的电源清除了 RAM 的内容，因此用户必须只能在 TCM 将不被使用且不包含有效数据的情况下移除 RAM 电源。

第6章 总线接口单元

本章描述了 ARM926EJ-S 总线接口单元（Bus Interface Unit，BIU）。包含以下小节：

- 关于总线接口单元；
- 支持的AHB传输。

6.1 关于总线接口单元

ARM926EJ-S 总线接口单元（BIU）仲裁和调度 AHB 请求。BIU 包含独立的用于指令和数据访问的主机，实现了 AHB 系统完整的灵活性。独立的主机能够实现多层 AHB（见《Multi-layer AHB Overview》）和多 AHB（multi-AHB）系统，带来了总体总线带宽增加和更灵活的系统体系的好处。每个主机完全兼容在 AMBA 规范（版本 2.0）中定义的 AHB 总线主机并实现了主机功能。

为增加系统性能，使用写缓冲来阻止AHB写操作暂停ARM926EJ-S系统。详细细节，参见第四章Cache和写。

数据 BIU AHB 信号以 D 为前缀，指令 BIU 信号以 I 为前缀。

6.2 支持的AHB传输

ARM926EJ-S 处理器支持 AHB 传输的子集。允许的 AHB 传输在下面的小节中描述：

- 存储器映射；
- 传输大小；
- 一级和二级AHB属性的映射；
- 字节和半字访问；
- AHB系统考虑因素；
- AHB时钟。

6.2.1 存储器映射

ARM926EJ-S 处理器是一个带两个 AHB 接口的掩埋（cached）处理器。关键的系统设计问题是 D 端必须能和 I 端一样以相同的存储器映射，访问相同的存储器。这不仅是加载代码的要求，也是能访问 PC 相关的文字池，以及 SWI 和仿真指令处理程序正常工作的要求。

注意：这和一些 I 总线可以连接到 ROM 而 D 总线仅连接到 RAM/外设的哈佛安排不同。

6.2.2 传输大小

ARM926EJ-S 处理器能以单字节、半字、字、四字突发或八字突发的方式执行所有的 AHB 访问。任何不是 1、4 或 8 字大小的 ARM9EJ-S 内核请求都被分块为这些大小的数据包。例如，一个 12 字的 STM 指令在 AHB 上是以一个 8 字突发之后紧跟一个 4 字突发的方式执行。如果突发由于分块或重试响应或由于 HGRANT 的移除而打断，那么该突发作为单次传输而完成。因此 ARM926EJ-S 处理器仅使用 HBURST 和 HSIZE 可能的编码的一个子集。

表 6.1表示了ARM926EJ-S处理器使用的HBURST编码，以及执行每个突发大小时的操作。

表 6.1 支持的 HBURST 编码

HBURST[2：0]	描述	操作
Single	单次传输	单次传输字、半字或字节： ● 数据写（DCache 中未命中的 NCNB、NCB、WT 或 WB）； ● 数据读（NCNB 或 NCB）； ● NC 指令读取（预取和非预取）； ● 页表搜索读； ● 失去授予（总线）或接收到分块/重试响应的突发的继续。
Incr4	四字增量突发	半行 cache 写回。指令预取，如果使能了的话。四字突发 NCNB、NCB、WT 或 WB 的写。
Incr8	八字增量突发	全行 cache 写回。八字突发 NCNB、NCB、WT 或 WB 的写。
Wrap8	八字回环突发	Cache 行填充。

注意：Incr4 和Incr8 突发可以对齐到任何字边界。ARM926EJ-S处理器以字宽的传输在AHB上执行所有的Thumb指令读取。见一级和二级AHB属性的映射。

所有突发读和写操作被 ARM926EJ-S 处理器以字宽（HSIZE[2：0] = 010）传输来执行。单次读和写操作以字节（HSIZE[2：0] = 000）、半字（HSIZE[2：0] = 001）或字（HSIZE[2：0] = 010）宽的传输来执行。

注意：ARM926EJ-S 处理器并不在 DHTRANS 或 IHTRANS 上产生 BUSY 传输。

6.2.3 一级和二级AHB属性的映射

表 6.2表示了IHPROT[3：0]和DHPROT[3：0]对存储器操作的映射。

表 6.2 IHPROT[3：0]和 DHPROT[3：0]属性

操作		HPROT[3：0][1]	描述
DCache 行填充		11P1[2]	CB，数据访问
ICache 行填充		11P0[2]	CB，操作代码读取
页表搜索（数据）		1111	页表搜索由数据访问的 TLB 未命中引起
页表搜索（指令）		1110	页表搜索由指令读取的 TLB 未命中引起
指令读取		00P0[2]	NCNB 操作代码读取
数据访问	LDR/STR	00P1[2]	NCNB
		01P1[2]	NCB
	STR	11P1[2]	WT/WB
DCache 写回		1111	-

[1]、为 IHPROT[3：0]或 DHPROT[3：0];

[2]、如果访问是由内核的特权访问方式引起的则 P 为 1，如果是由用户方式访问引起的则 P 为 0。

由于数据和指令的 TLB 未命中而出现的表搜索读是使用数据端总线主机来执行的。DHPROT[0]的状态可以用来识别表搜索是否由 TLB 中的指令读取未命中引起的：

DHPROT[0] = 0 表示指令读取未命中导致页表搜索。

DHPROT[0] = 1 表示数据访问未命中导致页表搜索。

指定到 LDR 指令上的属性也应用到 LDM、LDRD 和 LDC 操作。类似地 STR 的这些属性也应用到 STM、STRD 和 STC 操作。

DCache 的写回可以由行填充期间的回收或直接的 cache 清理操作引起。

6.2.4 字节和半字访问

本节描述了字节和半字访问的：

- 地址对齐；
- Thumb指令读取；
- 端结构和字节通路指示。

地址对齐

ARM926EJ-S BIU 执行地址对齐检查并对齐 AHB 地址到必需的边界。16 位的访问被对齐到半字边界，而 32 位的访问（被对齐）到字边界。

Thumb指令读取

所有指令读取都被当做 AHB 上的 32 位访问，无关 ARM9EJ-S 内核的状态。如果 ARM9EJ-S 内核处于 Thumb 状态，那么一次可以读取两条指令。

端结构和字节通路指示

AMBA 规范（版本 2.0）并没有指定任何端结构的直接支持。ARM926EJ-S 处理器提供一个增补信号，DHBL，来表示写传输时哪个字节将被更新和读操作时哪个字节必须包含有效数据。这是使用地址和访问的端结构创建的。

CFGBIGEND信号表示了ARM9EJ-S内核使用的当前端结构设置，并反映在CP15 c1 中保持的值里。参见控制寄存器c1。

由于写操作被缓冲，如果写缓冲在控制寄存器里改变端结构的设置之前没有被排干，CFGBIGEND 信号的值可能和 DHBL 不一致。

DHBL 是以小端格式编码的。例如，值 b0001 表示小端模式中的字节 0，而在大端模式中为字节 3。

6.2.5 AHB系统考虑因素

这一节描述了 AHB 以下方面的考虑因素：

- 单层AHB系统；
- 多层AHB系统；
- 多AHB系统；
- 存储器一致性。

单层AHB系统

如果 ARM926EJ-S 处理器将被用在一个单层 AHB 系统中，则各个 BIU 主机必须被视为唯一的。

集成 ARM926EJ-S 两个总线主机到一个单层 AHB 系统中最简单的方式是将每个主机作为独立的到 AHB 仲裁器的请求者，和任何多主机系统一样。数据主机通常有高于指令主机的仲裁优先权。

注意：ARM926EJ-S 指令 AHB 接口并不执行锁定的传输因此 IHLOCK 总是被驱动为低电平。

DHCLKEN和IHCLKEN必须被连接到一起，如AHB时钟部分的描述一样。如果HCLK和CLK是相同的频率，DHCLKEN和IHCLKEN必须被连接到高电平。

由于当总线转移占有权时的移交周期，如果总线当前是空闲的则一个未被授权的总线主机将在总线上招致一个额外的延时周期。这意味着如果数据 BIU 是默认的总线主机，那么它能比指令 BIU 早一个周期发起 AHB 传输，指令 BIU 必须等待总线占有权被移交。

这个周期的延时仅在第一次（总线周转）处理时存在。如果指令 BIU 正在预取指令，例如，它能执行背靠背请求且维持总线占有权，直到一个更高优先级总线主机被授权。

多层AHB系统

图 6.1表示了一个多层AHB系统的示例。在这个示例中I接口被标签为I-side，而D接口被标签为D-side，被配置为通过一个互联矩阵来访问四个从机端口。如果这两个AHB接口，即所谓的层（layer），在同一时刻需要访问相同的从机，那么在互联矩阵内的仲裁处理将确定有最高优先级的层。在这个系统中D-side可以访问任何I-side当前未使用的从机端口。这增加了总体可用的总线带宽。

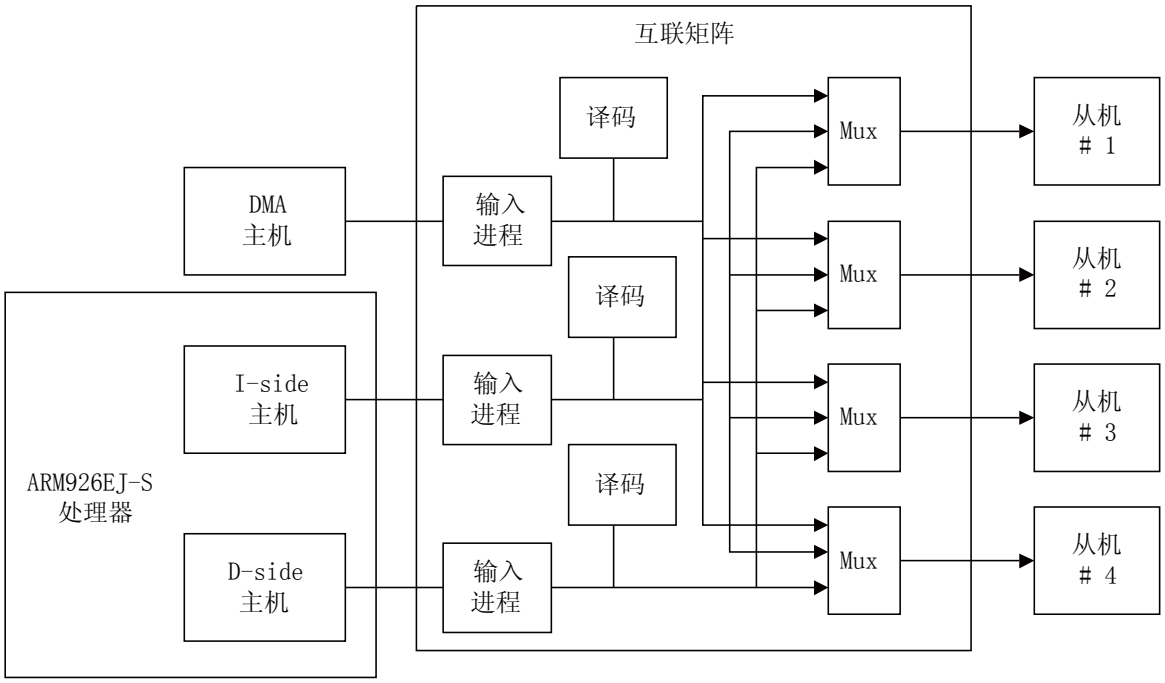


图 6.1　多层 AHB 系统示例

多层 AHB 的更多细节见《Multi-layer AHB Overview》的描述。

多AHB系统

ARM926EJ-S 指令和数据 AHB 接口可以连接到分离的 AHB 系统上是可能的，尽管必须有一个机制来支持数据端访问指令端存储器。每个 AHB 系统可以运行在不同的频率。
ARM926EJ-S 处理器能够通过提供两个 HCLKEN 输入来处理这个问题：

- DHCLKEN 被用来指定数据 BIU 是主机的系统的 HCLK 上升沿；
- IHCLKEN 被用来指定指令 BIU 是主机的系统的 HCLK 上升沿。

图 6.2表示了一个多AHB系统的示例。

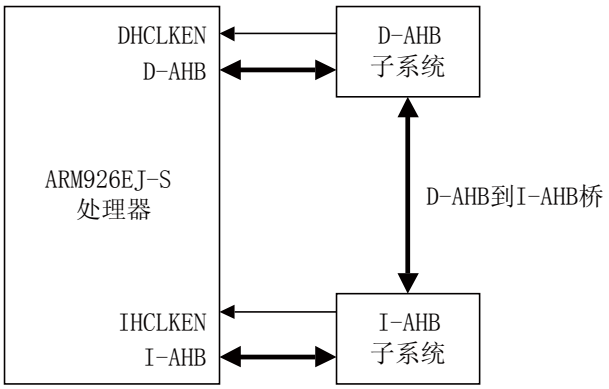


图 6.2　多 AHB 系统示例

如果AHB系统都运行在相同的频率，DHCLKEN和IHCLKEN必须连接到一起。更多细节见AHB时钟。

每个系统的 AHB 时钟，HCLK1 和 HCLK2 必须和 ARM926EJ-S 时钟信号 CLK 同步。

存储器一致性

由于 ARM926EJ-S 处理器哈佛体系的本质，指令和数据流顺序不能被保证，而这两个主机的仲裁顺序可以任意考虑。

对单个和多层 AHB 系统：

- 如果两个主机都同时发出请求，则两个主机的仲裁优先级确定了哪个主机被授权总线；
- 如果被授权的主机接收到一个分块或重试响应，另外一个主机可以被授权总线并在被分块主机完成之前完成它的传输。

对多 AHB 系统：

- 两个系统可以运行在不同的频率；
- 存储器从机可以插入等待和/或发出分块或重试响应。

如果（指令和数据）流的顺序是关键的，以自修改代码为例，一个指令内存栅（Instruction Memory Barrier，IMB）必须被用来强制一致性。更多细节见第九章指令内存栅。

6.2.6 AHB时钟

ARM926EJ-S 的设置使用单一的时钟，CLK。为使 ARM926EJ-S 处理器比 AHB 系统总线运行在更高的频率，对每个总线主机要求分离的 AHB 时钟使能。在一个多 AHB 系统中每个 AHB 系统可以运行在不同的频率：

DHCLKEN　　被用来对系统数据 BIU 总线主机表示 HCLK 上升沿。

IHCLKEN　　被用来对系统指令 BIU 总线主机表示 HCLK 上升沿。

图 6.3表示了CLK、HCLK、DHCLKEN和IHCLKEN之间的关系。

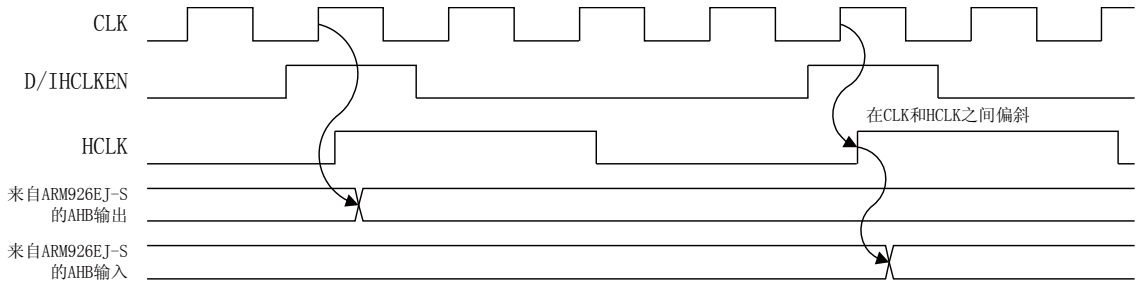


图 6.3　AHB 时钟的关系

对单个和多层 AHB 系统，DHCLKEN 和 IHCLKEN 必须被连接到一起。如果 HCLK 和 CLK 是相同频率的，那么相关的 HCLKEN 输入，必须被连接到高电平。

CLK 和 HCLK 必须被同步。CLK 和 HCLK 之间的偏斜必须最小化。

6.2.7 外部中止的限制

如果从 AHB 传输返回一个错误响应，只有特定类型的访问才引起外部中止。它们是：

- 页表搜索；
- 非高速缓存的读；
- 非缓冲的写；
- 非高速缓存的读锁写（SWP）。

对所有其它类型的访问（cache 行填充、写回排出（eviction）、缓冲的写），错误响应被忽略。

如果 ARM926EJ-S 处理器被用在必须容忍在外部存储器中的软件错误的系统中，那么当发出 AHB 传输时软件错误检测和修正必须都在硬件中完成。DHREADY 和 IHREADY 信号可以被用来延长传输，直到正确的数据可用时。

第7章 非高速缓存的指令读取

本章描述了在 ARM926EJ-S 处理器上非高速缓存的指令读取。包含以下小节：

- 关于非高速缓存的指令读取。

7.1 关于非高速缓存的指令读取

ARM926EJ-S处理器执行推测性的指令读取以提高性能。推测的指令读取在复位时被时能。这可以通过使用调试状态寄存器CP15 c15 的 16 位来禁止。见测试和调试寄存器c15。如果预取被禁能则只有直接由ARM9EJ-S内核发出的指令读取能够导致在AHB接口上的指令读取。

下面的部分被分为：

- 使用非高速缓存的代码；
- 自修改代码；
- AHB的行为。

7.1.1 使用非高速缓存的代码

和其他 ARM9 家族的掩埋内核相比尽管非高速缓存的代码性能已经被提升，但仍然建议实际使用中优先使用 ICache。

非高速缓存的代码以前被用作操作系统的启动引导和阻止cache污染。使能ICache而没使能MMU是没有任何意义的，见第四章Cache和写，而cache污染可以用cache锁定寄存器来控制，见Cache锁定寄存器c9。

7.1.2 自修改代码

使用四字缓冲来保存推测性的读取指令。只有顺序的指令是推测性读取的，而在 ARM9EJ-S 内核发出一个非顺序指令读取的事件中，缓冲的内容被丢弃（清除）。在预取缓冲中的内容可能在由 ARM9EJ-S 内核发出的顺序指令预取序列期间失效的情况下，例如打开或关闭 MMU，或打开 ICache，预取指缓冲也被清除。这避免了执行一个直接的指令内存栅（IMB）操作的要求，除非当自修改代码被使用时。由于预取指缓冲区在 ARM9EJ-S 内核发出一个非顺序的指令读取、分支指令或等效的操作时被清除，（这个现象）可以被用来实现要求的 IMB 行为。这在下面的代码序列中举例说明：

```
    LDMIA       R0,{R1-R5}          ; load code sequence into R1-R5
    ADR         R0,self_mod_code
    STMIA       R0,{R1-R5}          ; store code sequence (nonbuffered region)
    B           self_mod_code       ; branch to modified code
self_mod_code:
```

这种IMB实现仅应用到ARM926EJ-S处理器从一个非高速缓存的存储器区域运行代码时。如果代码是从一个可高速缓存的存储器区域运行，或是使用了不同的器件，那么要求不同的IMB实现。IMB在第九章指令内存栅中描述。

7.1.3 AHB的行为

如果指令预取被禁能，所有指令读取以单个，非顺序读取的方式出现在 AHB 接口上。

如果预取被使能那么指令读取以四个指令突发，或单个，非顺序读取的方式出现。跨越 1KB 边界时没有执行推测性的指令读取。

所有的指令读取，包括那些在 Thumb 状态下完成的，都是字（32 位）传输。在 Thumb 状态下一个单字指令读取读到两条 Thumb 指令，而一个四字突发读到 8 条指令。

第8章 协处理器接口

本章描述了 ARM926EJ-S 协处理器接口。包含下面的小节：

- 关于ARM926EJ-S外部协处理器接口；
- LDC/STC；
- MCR/MRC；
- CDP；
- 特权指令；
- 忙等待和中断；
- CPBURST；
- CPABORT；
- nCPINSTRVALID；
- 连接多个外部协处理器

8.1 关于ARM926EJ-S外部协处理器接口

ARM926EJ-S 支持通过一个外部协处理器接口从片上协处理器到 ARM9EJ-S 内核的连接。支持协处理器指令的所有类型。

协处理器通过使用协处理器内的流水线跟踪器（pipeline follower）来确定它们必须执行的指令。因为每个指令从存储器送来，指令进入了 ARM9EJ-S 流水线和协处理器流水线。为避免指令被协处理器锁存的临界通道，协处理器流水线必须运行在 ARM9EJ-S 内核流水线一个时钟周期之后。

这两个流水线通过在第一个执行周期内暂停 ARM9EJ-S 内核流水线来同步，无论外部协处理器指令何时从译码移动到执行状态。

为能让协处理器在ARM9EJ-S内核流水线被暂停时继续执行协处理器数据操作，例如正在等待cache行填充发生时，协处理器接收时钟CLK，和时钟使能信号CPCLKEN。用户可以使用这些信号来产生一个门控的协处理器时钟，电路见图 8.1。

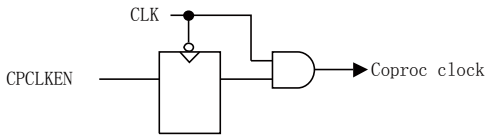


图 8.1 产生协处理器时钟

图 8.2表示了这些信号的时序以及协处理器在什么时候必须推进（advance）它的状态。

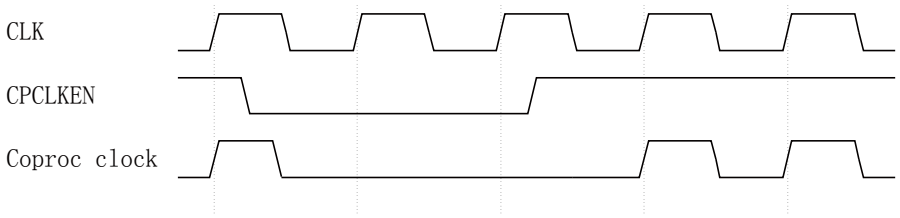


图 8.2 协处理器时钟

这是一种产生反映 ARM9EJ-S 内核流水线推进的时钟的技术。如果 CPCLKEN 在 CPCLK 上升沿时为低电平那么 ARM9EJ-S 内核流水线被暂停并且协处理器流水线不能推进。

8.1.1 协处理器指令

有三种类型的协处理器指令：

LDC 或 STC 从存储器加载协处理器寄存器或存储协处理器寄存器到存储器。

MCR/MCRR 或 MRC/MRRC

在协处理器和 ARM 处理器内核之间的寄存器传输。

CDP 协处理器数据操作。

以下部分给出了协处理器怎么必须执行这些指令集的示例：

- LDC/STC；
- MCR/MRC；
- CDP。

8.2 LDC/STC

图 8.3表示了该操作的周期时序。

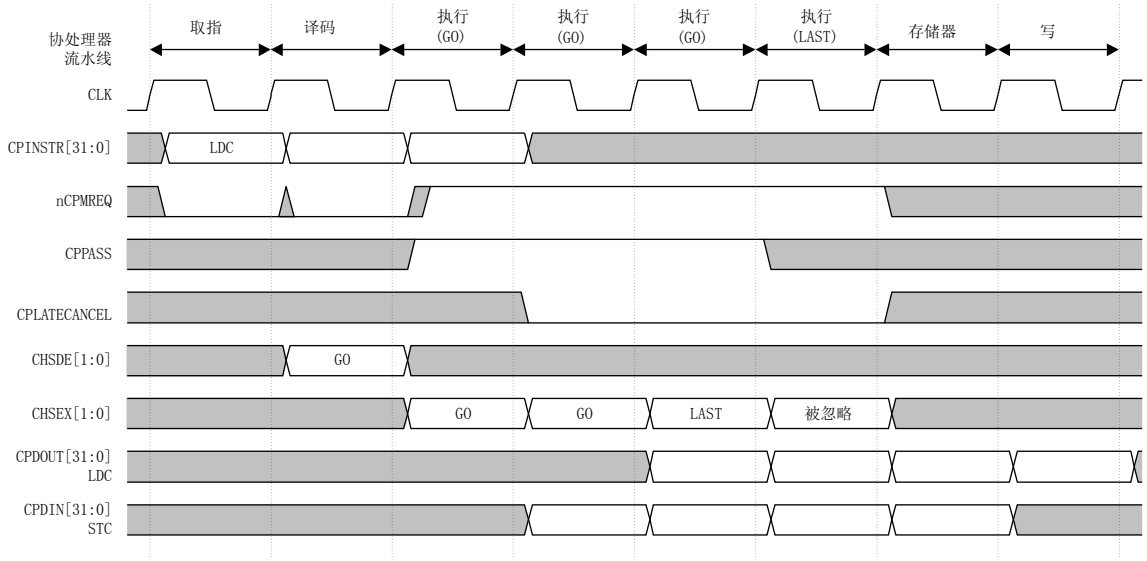


图 8.3 LDC/STC 周期时序

图 8.3中传输了四个字的数据。传输的字数是由协处理器如何驱动CHSDE[1：0]和CHSEX[1：0]总线来确定。

和所以其他指令一样，ARM9EJ-S 内核在译码状态期间的时钟上升沿之后执行主体的译码。从这之后，内核提交到执行指令（阶段）并因此执行一个指令读取。协处理器指令流水线通过监测 nCPMREQ 保持和 ARM9EJ-S 内核步调一致。nCPMREQ 是一个低电平有效信号，表示 ARM9EJ-S 流水线是否已经推进。CPINSTR 在下一个周期跟随着读取的指令而更新。这意味着当前在 CPINSTR 上的指令必须进入协处理器流水线的译码状态，并且处在协处理器流水线译码状态的指令必须进入它的执行状态。

在执行状态期间，条件码和标志位一起来确定该指令是否执行。CPPASS 的输出被断言为高电平，如果处在协处理器流水线上的指令：

- 是一个协处理器指令；
- 通过了它的条件码。

如果一个协处理器指令处于忙等待（busy-wait）状态那么 CPPASS 在每个周期都被断言，直到该协处理器指令被执行完毕。如果在忙等待期间出现一个中断，那么 CPPASS 被驱动为低电平且协处理器必须停止协处理器指令的执行。

另外一个输出，当指令进行中导致一个数据中止时 CPLATECANCEL 被用来撤销协处理器指令。该信号在协处理器指令的第一个协处理器执行周期之后的周期上的 CLK 上升沿有效。

在时钟的上升沿 ARM9EJ-S 内核检查协处理器握手信号 CHSDE[1：0]和 CHSEX[1：0]：

- 如果一个新的指令正进入在下一个周期内的执行状态，那么它检查 CHSDE[1：0]；
- 如果当前在执行的协处理器指令需要另外一个执行周期，那么它检查 CHSEX[1：0]。

这些握手信号编码了四个状态，见表 8.1。

表 8.1　握手信号编码

状态	值	描述
WAIT	00	如果有附属的协处理器能够处理该指令，但不能立即处理，那么协处理器握手信号被驱动来表示 ARM9EJ-S 内核已暂停。这既是所谓的忙等待条件。在忙等待条件下，ARM9EJ-S 内核在一个空闲状态里循环，等待 CHSEX[1：0]被驱动为另一个状态，或等待一个中断出现。如果 CHSEX[1：0]变为 ABSENT 那么捕获到未定义指令陷阱。如果 CHSEX[1：0]变为 GO 或 LAST 那么该指令正在进行，如在 GO 中描述的一样。如果出现了一个中断那么 ARM9EJ-S 内核被强制退出忙等待状态。这通过 CPPASS 信号变低电平来表示给协处理器。当指令重新启动时协处理器禁止提交到该指令，也就是改变任何的协处理器状态，直到协处理器在握手信号表示 GO 或 LAST 条件时发现 CPPASS 为高电平。
GO	01	GO 状态表示协处理器可以立即执行官该指令，而这需要另外一个执行周期。ARM9EJ-S 内核和协处理器在提交到该指令之前必须考虑 CPPASS 信号的状态。对一个 LDC 或 STC 指令，那么协处理器指令在有两个或更多的字仍需要传输时以 GO 状态驱动握手信号。当只要求一个附加的字时协处理器以 LAST 驱动握手信号。
ABSENT	10	如果没有附属的协处理器能够执行协处理器指令，那么握手信号表示为 ABSENT 且 ARM9EJ-S 内核置于未定义指令陷阱。
LAST	11	一个 LDC 或 STC 指令可能传输多于一个字的数据。如果是这种情况，那么可能在忙等待之后，协处理器以一系列的 GO 状态驱动协处理器握手信号，（最后）跟着一个 LAST 周期。LAST 状态表示下一个传输是最后一个。如果仅有一个传输那么状态序列应该是[WAIT，[WAIT，…]]，LAST。

8.3 MCR/MRC

这些周期看起来很像STC/LDC。图 8.4表示了一个带忙等待状态的示例。

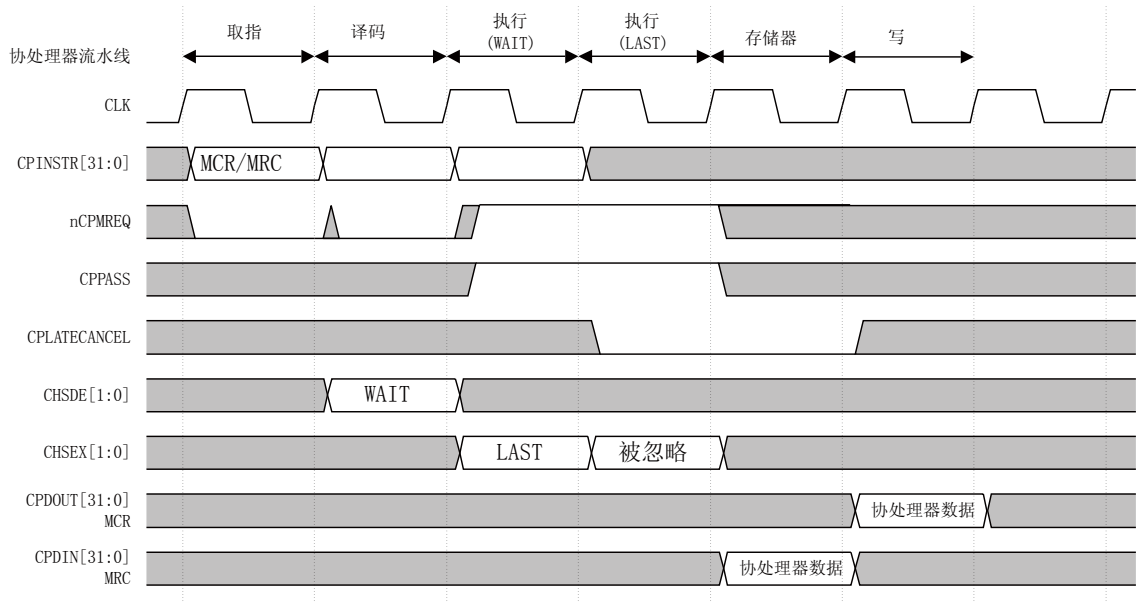


图 8.4　MCR/MRC 周期时序

首先，nCPMREQ 被驱动为低电平以表示在 CPINSTR 上的指令正进入流水线的译码状态。该协处理器译码新到的指令并按需要驱动 CHSDE[1：0]。

在下一个周期，nCPMREQ 被驱动为低电平以表示该指令现在被发布到执行状态。如果条件码通过且该指令将被执行，CPPASS 信号被驱动为高电平且 CHSDE[1：0]握手总线被检查。在所有其他情况下它被忽略。

对任何连续的执行周期而言检查 CHSEX[1：0]握手总线。当观察到 LAST 条件时，该指令被提交。在 MCR 指令的情况下，CPDOUT[31：0]总线在协处理器写状态期间被驱动为寄存器数据。在 MRC 指令情况下，CPDIN[31：0]在 ARM9EJ-S 存储器状态末尾被采样并在下一个周期内写入到目的寄存器。

8.3.1 互锁MCR

如果 MCR 操作的数据在第一个译码周期期间在 ARM9EJ-S 内核流水线内部是不可用的，那么 ARM9EJ-S 内核流水线互锁一个或多个周期直到数据可用为止。这种情况的一个例子就是将被传输的寄存器是上一个 LDR 指令的目的寄存器。在这种情况下 MCR 指令进入协处理器流水线的译码状态，并在进入执行周期之前在那里保持一定的周期数。

图 8.5表示了互锁MCR的示例。

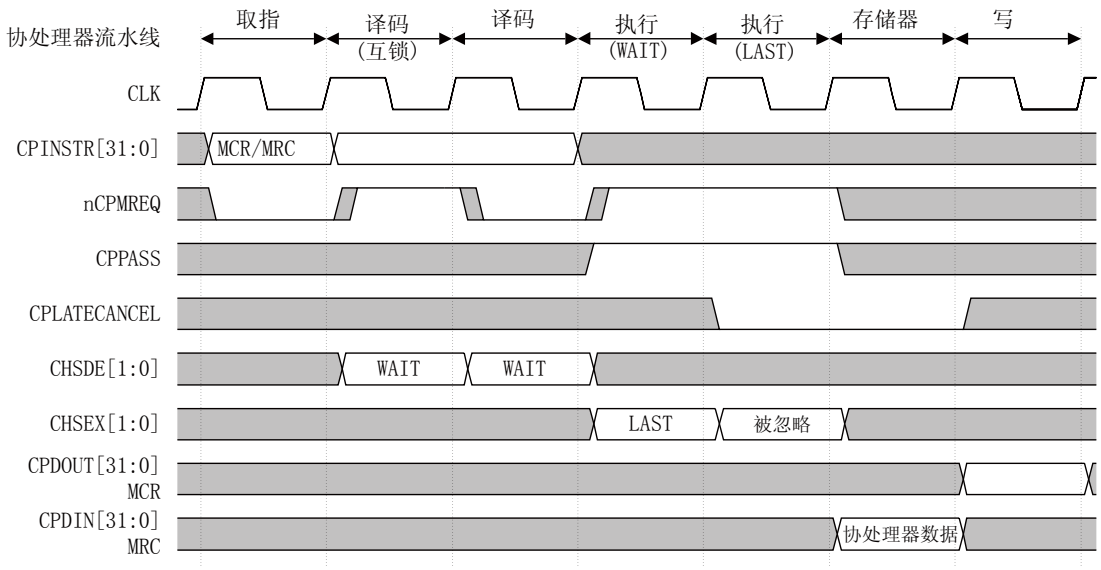


图 8.5　互锁 MCR

8.4　CDP

CDP指令通常在单周期内执行。和所有的前期周期一样，nCPMREQ被驱动为低电平以表示一个指令何时进入流水线的译码和之后的执行状态。如果指令将被执行那么CPPASS信号在执行期间被驱动为高电平。如果协处理器能够执行该指令则它立即以LAST驱动CHSDE[1：0]。如果指令需要一个忙等待周期，那么协处理器以WAIT驱动CHSDE[1：0]并在之后以LAST驱动CHSEX[1：0]。图 8.6表示了一个由于前一个指令引起了一个数据中止而被取消的CDP。

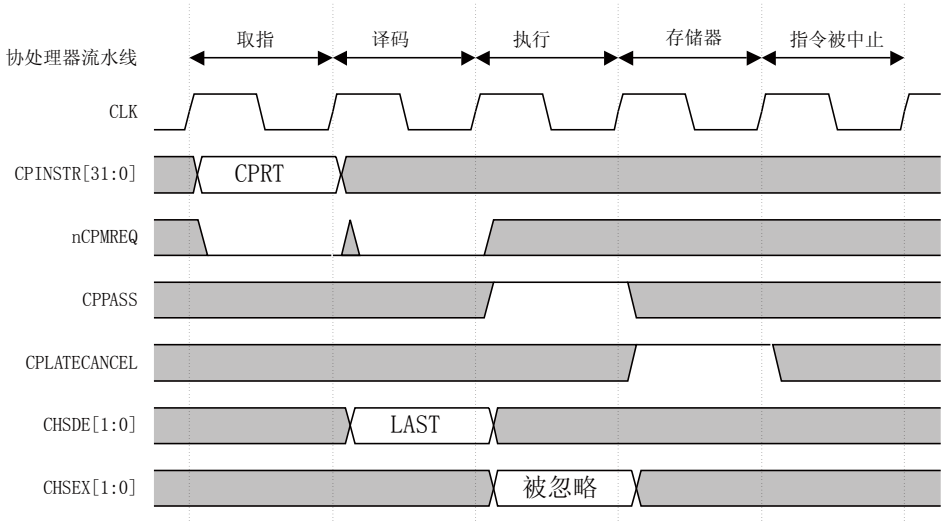


图 8.6　迟后取消的 CDP

CDP 指令进入流水线的指令状态并被 CPPASS 信号告知可以执行。在接下来的相位 CPLATECANCEL 被断言。这导致协处理器中止 CDP 指令的执行而这不会导致协处理器的状态改变。

注意：CPLATECANCEL 可以在存储器周期或执行周期断言。协处理器必须能够在这两个状态期间处理指令中止。

8.5 特权指令

协处理器可能仅在特权模式下限制特定指令的使用。为实现这个特点，协处理器必须跟随 nCPTRANS 的输出。

图 8.7表示了nCPTRANS如何在模式改变之后改变的。

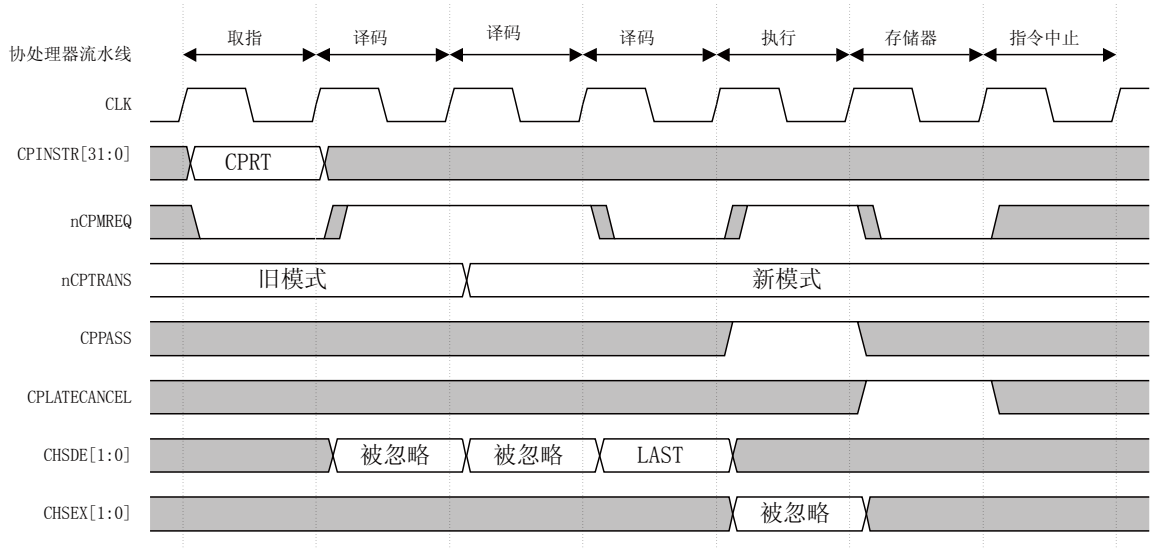


图 8.7 特权指令

8.6 忙等待和中断

协处理器在协处理器指令的执行期间被允许暂停（busy-wait）处理器，如果（例如）它仍忙于处理一个早期的协处理器指令。要完成这个，协处理器结合指令译码阶段在 CHSDE[1：0]上驱动 WAIT。当相关的指令进入流水线执行阶段时，协处理器可以在 CHSEX[1：0]上驱动 WAIT 足够多的周期以保持指令处于忙等待循环。

由于中断延时的原因协处理器可能在忙等待期间被中断，导致指令通过使用 CPPASS 而被抛弃。协处理器必须在每个忙等待周期监视 CPPASS 的状态。如果它为高电平则指令必须被执行。如果它为低电平则指令必须被抛弃。

图 8.8表示了一个忙等待协处理器指令由于一个中断被抛弃的示例。

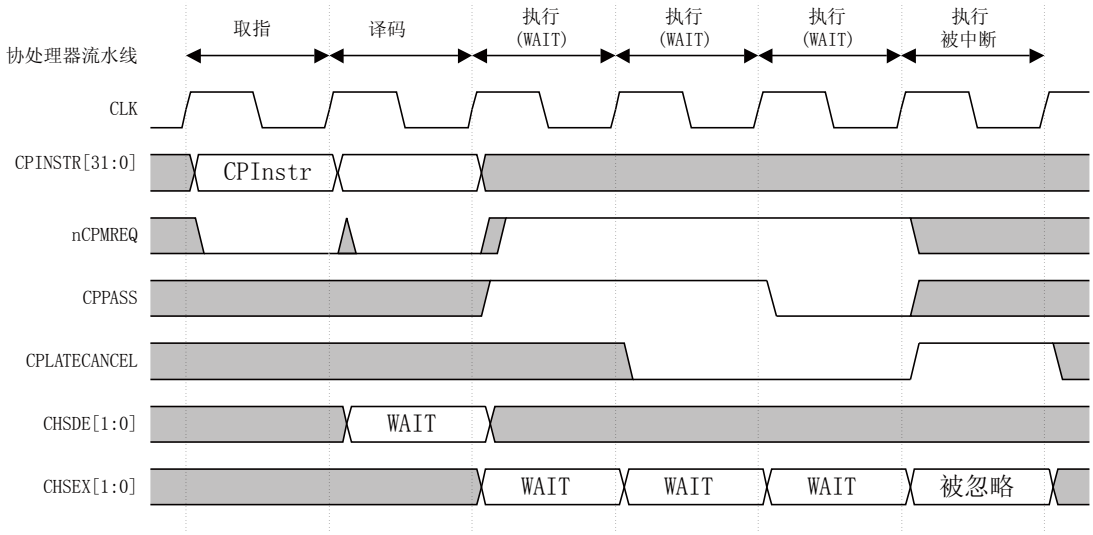


图 8.8　忙等待和中断

图 8.8中，CPLATECANCEL也由于中断的执行而被断言。

8.7　CPBURST

CPBURST信号被外部协处理器用来表示在一个LDC或STC操作中需要传输的字的数目。CPBURST被ARM926EJ-S存储器系统用来优化访问非高速缓存的或非缓冲的存储器区域的LDC/STC指令。CPBURST的编码见表 8.2。

表 8.2　CPBURST 编码

CPBURST[3：0]	需要传输的字数
b0000	1 个字或未知
b0001	2 个字
b0010	3 个字
…	…
b1110	15 个字
b1111	16 个字

对单个字传输和未知数目的传输的编码是一样的。对于一个 STC 或 LDC 操作如果 CPBURST 被设为 b0000，且这导致到一个非高速缓存的或非缓冲的存储器区域的访问，那么任何作为结果的 AHB 总线传输都被当做单独的非顺序的访问而执行。

CPBURST以和CHSDE响应相同的周期被外部协处理器驱动。该信号必须在所有其他时间被驱动为b0000。图 8.9表示了一个使用CPBURST的传输的示例。

8.8　CPABORT

CPABORT信号被断言为高电平，表示一个LDC/STC指令被中止。CPABORT在LDC/STC指令的存储器阶段之后的周期被断言。见图 8.9。

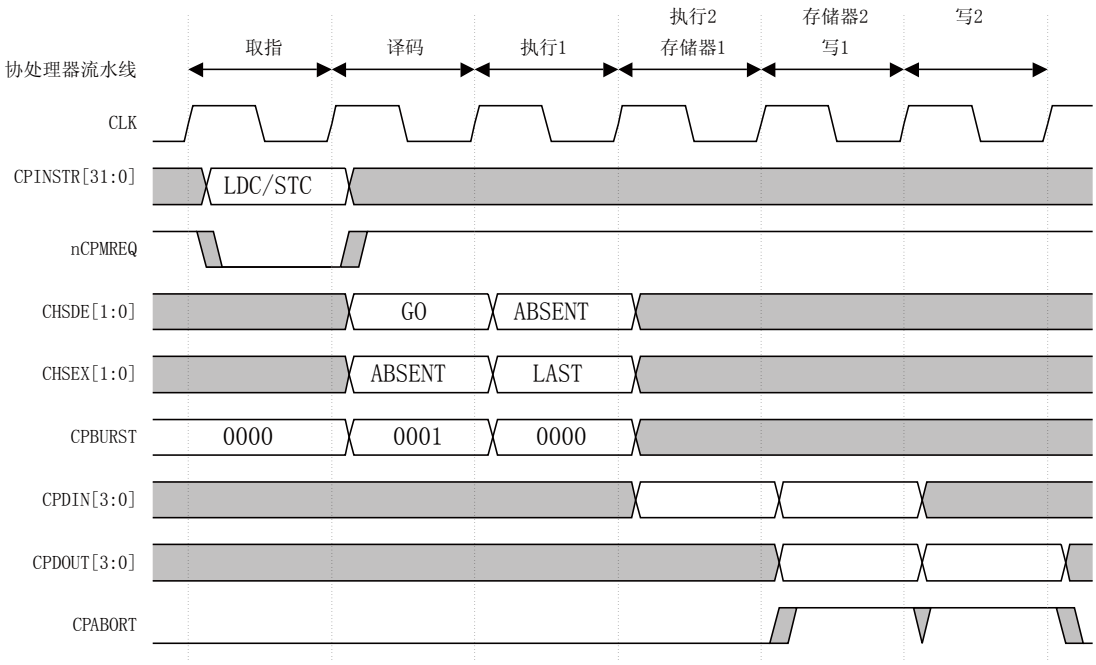


图 8.9　CPBURST 和 CPABORT 时序

8.9　nCPINSTRVALID

nCPINSTRVALID 信号表示当前在 CPINSTR 总线上的指令是否有效，并且必须被协处理器译码。如果 nCPINSTRVALID 为 1，那么该指令不能被协处理器译码且对该指令的所有相应的译码周期必须发出一个 ABSENT 响应。

nCPINSTRVALID 是在 ARM946E-S 和 ARM966E-S 处理器中 CPTBIT 信号的等价信号。

8.10 连接多个外部协处理器

如果多个协处理器连接到ARM926EJ-S处理器，那么多个协处理器的输出必须被组合以构成一个单组的协处理器输入。协处理器握手信号通过逻辑与（AND）高位和逻辑或（OR）低位而组合到一起。这让一个协处理器能够在它不活动时产生一个固定的b10（Absent）响应。其它的外部协处理器输入，CPDIN和CPBURST，通过逻辑或结合。见图 8.10。

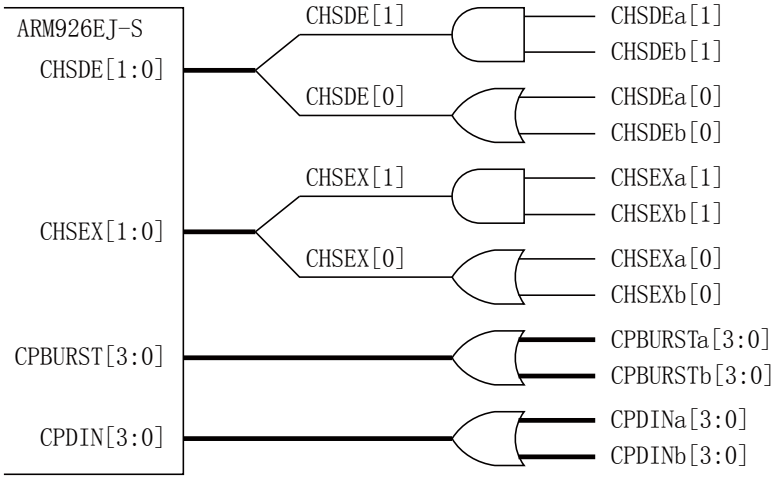


图 8.10　连接两个协处理器的排列

CPBURST 和 CPDIN 的逻辑或排列意味着协处理器必须在不活动时驱动 0 值到它们的 CPBURST 和 CPDIN 输出上，或不占有与这些信号有关的协处理器流水线阶段。

第9章 指令内存栅

本章描述了 ARM926EJ-S 指令内存栅（Instruction Memory Barrier，IMB）操作。包含以下小节：

- 关于指令内存栅操作；
- IMB操作；
- IMB序列示例。

9.1 关于指令内存栅操作

只要代码被视作数据，例如自修改（self-modifying）代码，或加载代码到存储器中，那么一个被称为指令内存栅（IMB）操作的指令序列必须被用来确保 ARM926EJ-S 处理器处理的数据和指令流之间的一致性。

通常指令和数据流被 ARM926EJ-S 处理器的存储器系统看作是完全独立的，并且任何数据端的改变不直接反映到指令端。例如，如果代码在主存中被修改那么 ICache 可能包含陈旧的条目。为移除这些陈旧的条目，部分或全部的 ICache 必须被清除。

9.2 IMB操作

为确保数据和指令端的一致性，用户必须采用下面的步骤：

1. 清理DCache；
2. 排干写缓冲；
3. 在二级AHB子系统中同步数据和指令流；
4. 清除ICache；
5. 清空预取值缓冲。

9.2.1 清理DCache

如果cache包含对应到内存的写回区域的cache行，那么它可能包含脏的条目。这些条目必须被清理以使外部存储器和DCache一致。如果仅有cache的一小部分必须被清理，那么这可以通过使用一个清理DCache单个条目指令的序列来完成，或如果整个cache必须被清理，那么这可以通过使用测试和清理指令来高效的完成。参见Cache操作寄存器c7以获取cache维护操作的细节。

9.2.2 排干写缓冲

执行一个排干写缓冲指令导致 ARM9EJ-S 内核进入等待，直到未完成的缓冲的写完全呈现到 AHB 接口上。这包含由于（cache）清理操作而导致地数据被写回到主存而出现的写操作，以及用于存储指令的数据。

9.2.3 在二级AHB子系统中同步数据和指令流

二级 AHB 子系统可能也要求在数据和指令端的直接同步。对数据和指令 AHB 主机而言被连接到不同的 AHB 子系统是可能的。即使两个主机都出现在相同的总线上，一些分离 ICache 的形式也可能由于性能原因而存在，而这必须被清除以确保一致性。

在二级存储器中同步指令和数据的处理必须通过使用一些完整的模块操作的形式而被调用。这是为了确保操作的结束可以用软件来决定。建议使用非缓冲的存储（STR）或非高速缓存的加载（LDR）之一来触发外部同步。

9.2.4 清除ICache

ICache 必须被清除以移除不再有效的指令的任何陈旧拷贝。如果 ICache 将不被使用，或被修改的区域不处在可内存的高速缓存区域，那么这（清除 ICache）可能不被要求。

9.2.5 清空预取值缓冲

为确保一致性，预取指缓冲必须在自修改的代码执行前被清空（flush）。参见自修改代码。

9.3 IMB序列示例

下面的代码序列对应 IMB 操作中的步骤 1–4：

```
clean_loop
    MRC     p15, 0, r15, c7, c10, 3    ;clean entire dcache using test and clean
    BNE     clean_loop
    MCR     p15, 0, r0, c7, c10, 4     ;drain write buffer
    STR     rx,[ry]                    ;nonbuffered store to signal L2 world to
                                       ;synchronize
    MCR     p15, 0, r0, c7, c5, 0      ;invalidate icache
```

下面的代码序列表示了在修改了单个指令之后使用的一个 IMB 序列，例如设置一个软件断点，没有外部同步的要求：

```
    STR     rx,[ry]                    ;store that modifies instruction at
                                       ;address ry
    MCR     p15, 0, ry, c7, c10, 1     ;clean dcache single entry (MVA)
    MCR     p15, 0, r0, c7, c10, 4     ;drain write buffer
    MCR     p15, 0, ry, c7, c5, 1      ;invalidate icache single entry (MVA)
```

第10章 嵌入式跟踪宏单元的支持

本章描述了对 ARM926EJ-S 处理器的嵌入式跟踪宏单元（Embedded Trace Macrocell，ETM）的支持。包含下面的小节：

- 关于嵌入式跟踪宏单元的支持。

10.1 关于嵌入式跟踪宏单元的支持

为支持实时跟踪，ARM926EJ-S 处理器提供了一个能够连接到嵌入式跟踪宏单元（ETM）的接口。关于 ETM 的更多信息，参见《ETM9 Technical Reference Manual》。

ETM 包含两个部分：

跟踪端口 开发了一个跟踪协议对在大型 ASIC 设计中深入嵌入的处理器内核提供实时跟踪能力。由于 ASIC 通常包含大量的片上存储器，不可能通过仅观察 ASIC 的引脚来确定处理器是如何工作的。要求一个跟踪端口来推定处理器的操作。

触发工具 由于可扩展规范的存在，让用户能够指定触发源的精确设置是实际应用所要求的。触发源包括地址和数据比较器，计数器和序列器。

ETM 被用来压缩跟踪信息并通过一个窄的跟踪端口输出。使用一个外部的跟踪端口分析仪（Trace Port Analyzer，TPA）来捕获跟踪信息。

ARM926EJ-S ETM 接口输出 ETM 所需要的信号来执行跟踪。接口通过 ETMEN 输入信号来使能和禁能。当不需要一个 ETM 模块时，ETMEN 输入可以被连接到低电平以禁能跟踪输出并节省功耗。

10.1.1 FIFOFULL

只要 ETM FIFO 填满了，EMT 就断言它的 FIFOFULL 信号。为防止跟踪范围内的数据丢失，ARM926EJ-S 处理器暂停直到 FIFOFULL 失效为止。

ARM926EJ-S 处理器仅在指令边界暂停，以运行任何 AHB 传输完成。ETM FIFO 水位线的编程必须考虑到这个条件。如果当前指令是 LDM 或 STM，那么 FIFO 可能在 FIFOFULL 被断言后不得不接收多达 16 个字。

中断（FIQ或IRQ）在FIFOFULL被断言时阻止了ARM926EJ-S处理器的暂停，除非中断被屏蔽。见测试和调试寄存器c15以获取中断如何在跟踪期间被屏蔽的细节。

注意：用 FIFOFULL 暂停内核影响了实时操作的性能。如果连接了 ETM，则 ETM 必须在正常的 ARM926EJ-S 处理器操作中被禁能，以阻止 FIFOFULL 负面影响 ARM926EJ-S 处理器的性能。

第11章 调试支持

本章描述了 ARM926EJ-S 处理器的调试支持。包含以下小节：

- 关于调试支持。

11.1 关于调试支持

使用 ARM9EJ-S 内核嵌入到 ARM926EJ-S 处理器中实现了调试支持。由 ARM9EJ-S 内核提供的调试支持的全部细节在《ARM9EJ-S Technical Reference Manual》中描述。

ARM926EJ-S 存储器系统的调试支持是通过扩展调试工具来实现的，使用 ARM9EJ-S 外部扫描链（扫描链 15）提供到 CP15 的访问。该扫描链在 ARM9EJ-S 内核外部但在 ARM926EJ-S 处理器内部。

11.1.1 调试时钟

系统和测试时钟必须外部同步到 ARM926EJ-S 宏单元。为与 ARM926EJ-S 宏单元同步片外调试时钟，需要一个三态同步器。这在《ARM9EJ-S Technical Reference Manual》的调试章节中描述。

11.1.2 扫描链 15

扫描链 15 使能了到CP15 寄存器的访问。扫描链 15 是 48 位长度。表 11.1表示了扫描链 15 的位分配。

表 11.1 扫描链 15 格式

位	功能
[47]	写，非读（W/R）
[46：33]	寄存器地址
[32]	初始化访问/访问完成 当写入： 0=NOP 1=初始化新的访问 当读出： 0=访问未完成 1=访问完成
[31：0]	数据值

当选择了扫描链 15 后，TDI 被连接到位 47 而 TDO 被连接到位 0。

要使用扫描链 15 来执行一个访问，用户必须：

1. 在 TAP 状态机的 SHIFT-DR 状态期间，移入读/写位，寄存器地址，和写入的寄存器数据，且位 32 设置为 1。对读操作数据值区域没有必要写入；
2. 移动到 UPDATE-DR 状态。由寄存器地址和写非读位指定的操作并未开始；
3. 返回到 SHIFT-DR 状态并执行一个移位操作因此位 32，和[31：0]为读，而一个 NOP 指令（位 32=0）被移入；
4. 移动到 UPDATE-DR 状态。没有操作被执行，因为位 32 为 0；
5. 检查移出的访问完成值。如果它为 1，则操作完成且位[31：0]包含读取的有效数据。如果它为 0，访问没有完成且用户必须返回到步骤 3。

注意：如果使用了 Multi-ICE，那么这有一个限制是一次只能写入任何扫描链的最多 40 个位。由于扫描链 15 是 48 位长度，CP15 寄存器的写操作要求两次操作以写入所有需要的位，并初始化访问。这可以通过第一次写位[31：0]为所要求的数据值，且位 32 为 0 来完成。这有为下一个操作预设值数据值域的效果。第二次操作设置位[47：33]为所需要的值，而位 32 为 1 以初始化访问。这依赖于扫描链 15 的特定行为，即如果一个值以位 32 被设为 0 的方式扫描则数据能够回流，并且没有在等待的访问。在这种情况下从 UPDATE-DR 的转变并不修改扫描链的内容，且写入的值可以在后继的 CAPTURE-DR，SHIFT-DR 序列中安全的被读回。

扫描链 15 到 CP15 寄存器的映射以和 CP15 MRC/MCR 操作相同的方式完成。扫描链的位[46：33]被映射到 Opcode_1，Opcode_2，CRn 和 CRm。

表 11.2表示了寄存器地址域到CP15 寄存器的映射。

表 11.2　扫描链 15 映射到 CP15 寄存器

MRC/MCR 指令域	扫描链 15 映射
Opcode_1	[46：44]
Opcode_2	[43：41]
CRn	[40：37]
CRm	[36：33]

到 cache 操作寄存器（CRn=c7）或 TLB 操作寄存器（CRn=c8）的写，需要一个地址的格式以选择要操作的条目，使用扫描链的数据值部分来提供地址信息。地址域的格式是和 Rd 中使用的值相同的，对于等价的 MCR 指令而言。

存储器系统调试操作（CRn=c15），需要一个地址用来选择一个条目，使用保持在调试地址寄存器中的值。见调试和测试地址寄存器。地址域的格式是和Rd中使用的值相同的，对于等价的MCR指令而言。

如果一个无效的执行被扫描到扫描链 15，则它被转换为 ID 寄存器的一个读操作。这意味着用户可以对比 ID 寄存器的读出值检查输出数据，以指示一个无效的指令被扫描入（扫描链）。

第12章 电源管理

本章描述了由 ARM926EJ-S 处理器提供的电源管理工具。包含以下小节：

- 关于电源管理。

12.1 关于电源管理

由 ARM926EJ-S 处理器提供的电源管理工具是：

- 动态电源管理（等待中断模式）；
- 静态电源管理（漏电流控制）。

12.1.1 动态电源管理（等待中断模式）

ARM926EJ-S 处理器可以通过等待中断指令进入低功耗状态：

```
MCR     p15,0,<Rd>,c7,c0,4
```

这条指令切换 ARM926EJ-S 处理器到一个低功耗状态直到中断（IRQ 或 FIQ）或调试请求的发生。调试请求既可以是一个外部调试请求 EDBGRQ，也可以是调试器使用扫描链 2 写 ARM9EJ-S 调试控制寄存器的 DBGRQ 位而发出的调试请求。

在等待中断模式下，所有的内部 ARM926EJ-S 时钟都可以停止。切换到低功耗状态被延时到所有写缓冲都被排干，且 ARM926EJ-S 存储器系统处于静态时为止。

到低功耗状态的切换由 STANDBYWFI 信号的断言来表示。如果 STANDBYWFI 被断言那么它保证了所有的 ARM926EJ-S 外部接口（AHB，TCM 和外部协处理器）都处于空闲状态。STANDBYWFI 信号被打算用来关闭到系统其它部分的时钟，比如外部协处理器，如果 ARM926EJ-S 处理器是空闲的则没有必要供给外部协处理器时钟。

STANDBYWFI信号在一个中断或调试请求之后的第二个周期失效。它保证了在任何外部接口上没有访问的形式被启动，直到STANDBYWFI失效之后的周期为止。图 12.1表示了一个IRQ中断之后STANDBYWFI信号的失效。

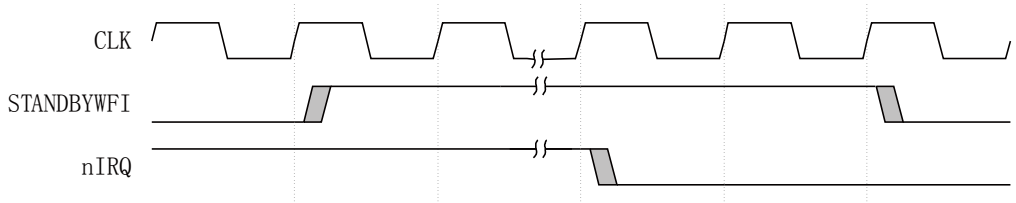
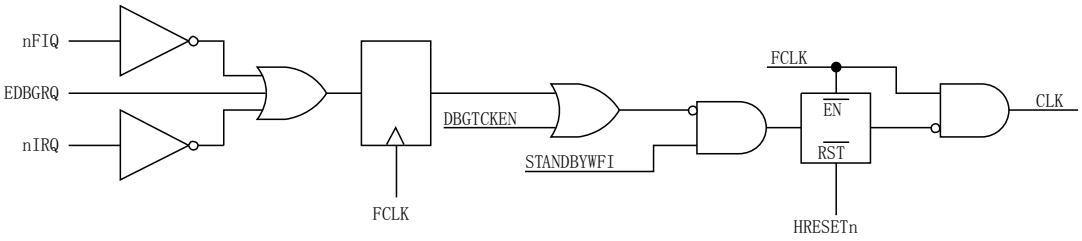


图 12.1 在 IRQ 中断之后 STANDBYWFI 的失效

当 ARM926EJ-S 已经进入到低功耗状态时，所有主要的内部时钟都被停止，包括 ARM9EJ-S 内核的时钟。然而，如果 DBGTCKEN 被断言则 ARM9EJ-S 被激活。这能够把数据写入到 ARM9EJ-S 的调试控制寄存器因此调试器能够强制（ARM9EJ-S）从等待中断模式中退出。这意味着如果 STANDBYWFI 为高电平且 DBGTCKEN 为低电平，则用户可以安全的停止 ARM926EJ-S 的 CLK。

图 12.2 表示了在等待中断期间停止主要的 ARM926EJ-S 时钟的推荐逻辑。



FCLK = 自由运行时钟

CLK = 供给ARM926EJ-S宏单元的时钟

图 12.2　在等待中断期间停止 ARM926EJ-S 时钟的逻辑

nFIQ，nIRQ 和 EDBGRQ 信号的本质让它们在门控逻辑中被使用前先被寄存。DBGTCKEN 必须联合使用以维持 ARM926EJ-S JTAG 逻辑和调试器使用的 RTCK 信号之间的关系。见《ARM9EJ-S Technical Reference Manual》以获取 DBGTCKEN 是如何被产生和使用的细节。

注意：如果 ARM926EJ-S 正处于低功耗 WFI 模式，STANDBYWFI 为高电平且 BLOCK_LEVEL_CLOCK_GATING 未使能，那么外部接口事务，例如 TCM 处理，将在外部接口非空闲时导致 STANDBYWFI 信号变为低电平。

这可能是系统中一个不需要的负面效果，引发外部逻辑脱离了 STANDBYWFI 信号的下降或上升沿。

如果 BLOCK_LEVEL_CLOCK_GATING 使能则这种效果不会发生，如果有外部接口活动 STANDBYWFI 仍保持为高电平。

12.1.2　静态电源管理（漏电流控制）

ARM926EJ-S 的设计是分区的，因此用于 cache 和 MMU 的 SRAM 模块可以在特定条件下掉电。

Cache 的 RAM

任一 Cache 的 RAM 可以被安全的掉电，如果使用 CP15 控制寄存器 c1 禁能各自的 cache 且它不包含有效的条目。当一个 cache 被禁能，仅有直接的 CP15 操作能引起 cache 的 RAM 被访问（c7 cache 维护操作）。这些指令在任一 cache 的 RAM 处于掉电时禁止执行。如果一个 cache 的任何 RAM 已经掉电，那么它们必须在重新使能相关 cache 之前被上电。

MMU 的 RAM

用于实现 MMU 的 RAM 可以被安全的掉电，如果使用 CP15 控制寄存器 c1 禁能 MMU 且它不包含有效的条目。当 MMU 被禁能，仅有直接的 CP15 操作能偶引起 MMU RAM 被访问（c8 TLB 维护操作和 c15 MMU 测试/调试操作）。这些指令在 MMU 的 RAM 处于掉电时禁止执行。MMU 的 RAM 必须在重新使能 MMU 之前被上电。

附录A 信号描述

本附录描述了 ARM926EJ-S 处理器的输入和输出信号。包含下面的小节：

- 信号属性和要求；
- AHB相关的信号；
- 协处理器接口信号；
- 调试信号；
- JTAG信号；
- 混杂信号；
- ETM接口信号；
- TCM接口信号。

A.1 信号属性和要求

为确保减轻 ARM926EJ-S 处理器集成到嵌入式应用中，以及简化综合流程，使用了下面的设计技术：

- 单个上升沿时钟定时（time）所有行为；
- 所有信号和总线都是单向的；
- 所有输入都要求和单个时钟同步。

这些技术指定了 ARM926EJ-S 处理器信号的顶层定义，因为所有输出从时钟上升沿之后改变而所有输入都在时钟上升沿被采样。另外，所有信号仅或为输入或为输出。没有使用双向信号。

注意：用户必须使用外部逻辑来同步异步的信号，例如中断源，在把它们应用到 ARM926EJ-S 处理器之前。

A.2 AHB 相关的信号

附表A.1描述了ARM926EJ-S处理器AHB相关的信号。

附表 A.1 AHB 相关的信号

信号名称	方向	描述
DHADDR[31：0]	输出	AHB 地址（数据）
DHBL[3：0]	输出	当前传输的字节通路指示
DHBURST[2：0]	输出	AHB 突发大小（数据）
DHBUSREQ	输出	AHB 总线请求（数据）
DHCLKEN	输入	表示数据 AHB 的 HCLK 的上升沿。如果 CLK 和 HCLK 是同频率的，DHCLKEN 必须被连接到高电平
DHGRANT	输入	AHB 总线授权信号（数据）
DHLOCK	输出	AHB 总线锁定信号（数据）
DHPROT[3：0]	输出	AHB 总线访问信息（数据）
DHRDATA[31：0]	输入	AHB 读数据（数据）
DHREADY	输入	AHB 传输完成信号（数据）
DHRESP[1：0]	输入	AHB 传输响应（数据）
DHSIZE[2：0]	输出	AHB 传输大小（数据），表示字节、半字或字。DHSIZE[2]被连接到低电平
DHTRANS[1：0]	输出	AHB 传输类型（数据）

续上表

信号名称	方向	描述
DHWDATA[31：0]	输出	AHB 写数据（数据）
DHWRITE	输出	AHB 传输方向（数据）
HRESETn	输入	AHB 复位信号
IHADDR[31：0]	输出	AHB 地址（指令）
IHBURST[2：0]	输出	AHB 突发大小（指令）
IHBUSREQ	输出	AHB 总线请求（指令）
IHCLKEN	输入	表示指令 AHB 的 HCLK 上升沿。如果 CLK 和 HCLK 是同频的，IHCLKEN 必须被连接到高电平
IHGRANT	输入	AHB 总线授权信号（指令）
IHLOCK	输出	AHB 总线锁定信号（指令）
IHPROT[3：0]	输出	AHB 总线访问信息（指令）
IHREADY	输入	AHB 传输完成信号（指令）
IHRDATA[31：0]	输入	AHB 读数据（指令）
IHRESP[1：0]	输入	AHB 传输响应（指令）
IHSIZE[2：0]	输出	AHB 传输大小（指令），表示字节、半字或字。IHSIZE[2]被连接到低电平
IHTRANS[1：0]	输出	AHB 传输类型（指令）
IHWRITE	输出	AHB 传输方向（指令）

A.3 协处理器接口信号

附表A.2描述了ARM926EJ-S处理器的协处理器接口信号。

附表 A.2 协处理器接口信号

信号名称	方向	描述
CPABORT	输出	表示 STC/LDC 操作被中止。在协处理器流水线的 WB 阶段被断言
CPBURST[3：0]	输入	表示 LDC/STC 操作传输的字数。如果没有连接外部协处理器，该信号必须被连接到 b0000
CPCLKEN 协处理器时钟使能	输出	协处理器时钟使能。当在 CLK 上升沿时为高电平则流水线跟随器逻辑可以推进
CPDIN[31：0] 协处理器写数据	输入	传输数据来自协处理器的协处理器数据总线
CPDOUT[31：0] 协处理器读数据	输出	传输数据到协处理器的协处理器数据总线
CPEN 协处理器使能	输入	当为低电平则禁能外部协处理器接口。如果 CPEN 为低电平那么 CHSDE 和 CHSEX 必须都被驱动为 b10（ABSENT 响应）
CPINSTR[31：0] 协处理器指令数据	输出	协处理器指令总线，指令被传输到协处理器内的流水线跟随器
CPPASS	输出	表示有一个协处理器指令正处于流水线的执行阶段，该指令必须被执行
CPLATECANCEL	输出	如果在协处理器指令的第一个存储器周期期间为高电平，那么协处理器必须取消该指令而不用改变任何内部状态
CHSDE[1：0] 协处理器握手译码	输入	来自协处理器流水线跟随器译码阶段的握手信号。表示 ABSENT（b10），WAIT（b00），GO（b01）或 LAST（b11）。如果没有连接外部协处理器则该信号必须被连接到 b10（ABSENT 响应）

续上表

信号名称	方向	描述
CHSEX[1：0] 协处理器握手译码	输入	来自协处理器流水线跟随器执行阶段的握手信号。表示 ABSENT（b10），WAIT（b00），GO（b01）或 LAST（b11）。如果没有连接外部协处理器则该信号必须被连接到 b10（ABSENT 响应）
nCPINSTRVALID 协处理器指令有效	输出	CPINSTR 的有效指令指示（替代 CPTBIT）
nCPMREQ 非协处理器指令请求[1]	输出	如果在 CLK 上升沿时该信号为低电平且 CPCLKEN 为高电平，则在 CPINSTR 上的指令必须进入协处理器流水线
nCPTRANS 非协处理器存储器转换	输出	当为低电平则协处理器接口位于非特权状态。 当为高电平则协处理器接口位于特权状态。

注：这里的非表示逻辑“非”，下同。

A.4 调试信号

附表A.3描述了ARM926EJ-S处理器调试信号。

附表 A.3 调试信号

信号名称	方向	描述
COMMRX 通信通道接收	输出	当为高电平，该信号表示通信通道接收缓冲包含有效数据，等待被读取
COMMTX 通信通道发送	输出	当为高电平，该信号表示通信通道发送缓冲为空
DBGACK 调试应答	输出	当为高电平表示处理器处于调试状态
DBGDEWPT 数据观察点	输入	由外部硬件断言以停止处理器的执行，用于调试目的。如果在一个数据存储器请求周期的末尾为高电平，它导致 ARM926EJ-S 处理器进入调试状态
DBGEN 调试使能	输入	使能处理器的调试特征。如果不要求调试则该信号必须被连接到低电平
DBGEXT[1：0] EmbeddedICE-RT 外部输入	输入	输入到 EmbeddedICE-RT 逻辑，让断点或观察点能独立于外部条件
DBGIEBKPT 指令断点	输入	由外部硬件断言以停止处理器的执行，用于调试目的。如果在一个指令读取的末尾为高电平，它在指令到达处理器流水线的执行阶段时导致 ARM926EJ-S 处理器进入调试状态
DBGINSTREXEC 指令执行	输出	表示处在处理器流水线执行阶段的指令已被执行
DBGRNG[1：0] EmbeddedICE-RT 范围输出	输出	表示 EmbeddedICE-RT 中相应的观察点寄存器匹配了当前出现在地址、数据和控制总线上的条件。该信号独立于观察点使能控制位的状态
DBGRQI 内部调试请求	输出	表示出现在内核调试逻辑上的调试请求信号。这是 EDBGRQ 和调试控制寄存器的位 1 的组合
EDBGRQ 外部调试请求	输入	外部调试器可以通过断言该信号来迫使处理器进入调试状态

A.5 JTAG 信号

附表A.4描述了ARM926EJ-S处理器的JTAG信号。

附表 A.4 JTAG 信号

信号名称	方向	描述
DBGIR[3：0] TAP 控制器指令寄存器	输出	这四个位反映当前加载到 TAP 控制器指令寄存器内的指令。这些位在 TAP 控制器处于 UPDATE-IR 状态时发生改变。
DBGnTRST 非测试复位	输入	这是 EmbeddedICE-RT 内部状态的低电平有效复位信号。该信号是一个电平敏感的异步复位信号
DBGnTDOEN 非 DBGTDO 使能	输出	当为低电平，表示串行数据将被驱动到 DBGTDO 输出上。通常在封装部分被用作 DBGTDO 引脚的输出使能
DBGSCREG[4：0]	输出	这五个位反映当前被 TAP 控制器选择的扫描链的 ID 号。这些位在 TAP 控制器处于 UPDATE-DR 状态时发生改变
DBGSDIN 外部扫描链串行输入数据	输出	包含将被应用到一个外部扫描链上的串行数据
DBGSDOUT 外部扫描链串行输出数据	输入	包含一个外部扫描链的串行数据输出。当没有连接外部扫描链时，该信号必须被连接到低电平
DBGTAPSM[3：0] TAP 控制器状态机	输出	该总线反应了 TAP 控制器状态机当前的状态
DBGTCKEN	输入	同步测试时钟使能
DBGTDI	输入	调试逻辑的测试数据输入
DBGTDO	输出	来自调试逻辑的测试数据输出
DBGTMS	输入	TAP 控制器的测试模式选择

A.6 混杂信号

附表A.5描述了ARM926EJ-S处理器上的混杂信号。

附表 A.5 混杂信号

信号名称	方向	描述
BIGENDINIT	输入	确定在系统复位之后 CP15 c1 的 B 位的设置。当为高电平则 B 位的复位状态是 1（大端）。当为低电平则 B 位的复位状态是 0（小端）
CLK	输入	该时钟定时所有 ARM926EJ-S 设计的操作。所有输出在上升沿之后改变而所有输入在上升沿被采样。该时钟可以在任一相位被延长。通过 DHCLKEN 和 IHCLKEN 信号，该时钟也定时 AHB 操作。通过使用 DBGTCKEN 信号，该时钟也控制 JTAG 和调试操作
CFGBIGEND ARM9EJ-S 内核端结构配置	输出	该信号反映了 CP15 c1 的 B 位的设置。当为高电平，处理器将存储器中的字节视为大端格式。当为低电平，存储器被视为小端
EXTEST	输入	EXTEST 模式测试信号。该信号必须在正常操作期间为低电平
INTEST	输入	INTEST 模式测试信号。该信号必须在正常操作期间为低电平

续上表

信号名称	方向	描述
nFIQ 非快速中断请求	输入	这是快速中断请求信号。该信号必须被同步到 CLK
nIRQ 非中断请求	输入	这是中断请求信号。该信号必须被同步到 CLK
SCANENABLE	输入	扫描使能测试信号。该信号必须在正常操作期间为低电平
STANDBYWFI	输出	当为高电平表示 ARM926EJ-S 处理器正处于等待中断模式
TAPID[31：0]	输入	这是 ARM926EJ-S 设备标识（ID）编码测试数据寄存器，可从扫描链访问。对于 ARM926EJ-S 处理器在设备被初始化时它必须被连接到 0x07926F0F
TESTMODE	输入	测试模式测试信号。该信号必须在正常操作期间为低电平
VINITHI 复位时异常向量的位置	输入	确定异常向量的复位位置。当为低电平，异常向量被放置在 0x0000 0000。当为高电平，异常向量被放置在 0xFFFF 0000

A.7 ETM 接口信号

附表A.6描述了ARM926EJ-S处理器ETM接口信号。

附表 A.6 ETM 接口信号

信号名称	方向	描述
ETMBIGEND	输出	ETM 大端配置指示
ETMCHSD[1：0]	输出	ETM 协处理器译码握手信号
ETMCHSE[1：0]	输出	ETM 协处理器执行握手信号
ETMDA[31：0]	输出	ETM 数据端的地址
ETMDABORT	输出	ETM 数据端的中止
ETMDBGACK	输出	ETM 调试模式指示
ETMDMAS[1：0]	输出	ETM 数据大小指示
ETMDMORE	输出	ETM 更多顺序数据指示
ETMDnMREQ	输出	ETM 数据端的存储器请求
ETMDnRW	输出	ETM 数据非读/写
ETMDSEQ	输出	ETM 顺序数据指示
ETMEN	输入	同步的 ETM 接口使能。如果没有使用 ETM 则该信号必须被连接到低电平
ETMHIVECS	输出	ETM 异常向量配置
ETMIA[31：0]	输出	ETM 指令端的地址
ETMIABORT	输出	ETM 指令端的中止
ETMID15TO11[15：11]	输出	ETM 指令端的数据域位[15：11]
ETMID31TO25[31：25]	输出	ETM 指令端的数据域位[31：25]
ETMIJBIT	输出	ETM Jazelle 状态指示
ETMInMREQ	输出	ETM 指令端的存储器请求
ETMINSTREXEC	输出	ETM 指令执行指示
ETMINSTRVALID	输出	ETM 指令有效指示
ETMISEQ	输出	ETM 顺序的指令访问
ETMITBIT	输出	ETM Thumb 状态指示
ETMLATECANCEL	输出	ETM 协处理器延后取消指示

续上表

信号名称	方向	描述
ETMnWAIT	输出	ETM 时钟暂停信号
ETMPASS	输出	ETM 协处理器指令执行指示
ETMPROCID[31：0]	输出	ETM 进程标识符
ETMPROCIDWR	输出	ETMPROCID 写选通
ETMRDATA[31：0]	输出	ETM 读数据
ETMRNGOUT[1：0]	输出	ETM 观察点寄存器匹配指示
ETMWDATA[31：0]	输出	ETM 写数据
ETMZIFIRST	输出	对于当前 Java 指令表示当前的译码周期是第一个被跟踪到的
ETMZILAST	输出	对于当前 Java 指令表示当前的译码周期是最后一个被跟踪到的
FIFOFULL	输入	ETM FIFO 满。如果没有使用 ETM 则该信号必须被连接到低电平

A.8 TCM 接口信号

附表A.7描述了ARM926EJ-STCM接口信号。

附表 A.7 TCM 接口信号

信号名称	方向	描述
DRADDR[17：0]	输出	数据 TCM 地址。这是访问的字地址。在请求周期有效
DRCS	输出	片选。表示在接下来的周期是否会发生一个 访问。在等待周期无效
DRDMAADDR[17：0]	输入	DTCM 存储器的直接存储器访问（DMA）地址。如果 DRDMAEN 被设为 1，那么 DRDMAADDR 的值直接被选通到 DRADDR
DRDMAEN	输入	DMA 访问周期。如果断言，DRADDR 是直接源自 DRDMAADDR，且 DRCS 是 DRDMACS 和当前 TCM 访问片选值逻辑或的结果
DRDMACS	输入	DTCM 的直接存储器访问的片选
DRIDLE	输出	数据 TCM 接口空闲： 0=TCM 访问 1=当前周期将不会发生访问或 TCM 被禁能。 （该信号）对 DMA 访问无效
DRnRW	输出	数据 TCM 的读非写： 0=读 1=写 表示访问是一个读还是写。在请求周期内有效
DRRD[31：0]	输入	数据 TCM 的读数据。在非等待的数据周期内有效
DRSEQ	输出	顺序请求。在请求周期内有效，在等待周期内断言。表示当前周期内的地址是顺序于前一个请求周期内使用的地址
DRSIZE[3:0]	输入	数据 TCM 大小。指定了连接的 TCM 存储器物理尺寸的静态配置输入。 0000=无 0011=4KB … 1010=512KB 1011=1MB 值 0001、0010 和 1100 到 1111 是保留的

续上表

信号名称	方向	描述
DRWAIT	输入	数据 TCM 等待状态输入。如果为高电平，则 DTCM 不能在该周期内服务（内核的）请求。在请求周期和顺序等待周期内有效。如果不是请求或等待周期则被忽略
DRWBL[3：0]	输出	数据 TCM 的写数据字节通路指示。在请求周期有效。对于读，设置为 b0000，对于写则表示哪些字节被写入，依赖于地址和访问的大小（字、半字或字节）。DRWBL 的位仅在写操作正发生时被设置，因此当 DRnRW 未被设置则 DRWBL 的所有位也未被设置
DRWD[31：0]	输出	数据 TCM 的写数据。在请求周期内当 DRnRW 为 0 时有效。在等待写周期内有效。
INITRAM	输入	在系统复位时使能指令 TCM。如果 VINITHI 为低电平则使能从指令 TCM 启动
IRADDR[17：0]	输出	指令 TCM 的地址。这是用于访问的字地址。在请求周期内有效
IRCS	输出	片选。表示在接下来的周期是否会发生一个 访问。在等待周期无效
IRDMAADDR[17：0]	输入	DMA 访问周期。如果断言，IRADDR 是直接源自 IRDMAADDR，且 IRCS 是 IRDMACS 和当前 TCM 访问片选值逻辑或的结果
IRDMAEN	输入	使能了用 IRDMAADDR 和 IRDMACS 输入到 ITCM 存储器的直接存储器访问
IRDMACS	输入	ITCM 的直接存储器访问的片选
IRIDLE	输出	指令 TCM 接口空闲： 0=TCM 访问 1=当前周期将不会发生访问或 TCM 被禁能。（该信号）对 DMA 访问无效
IRnRW	输出	指令 TCM 的读非写： 0=读 1=写 表示访问是一个读还是写。在请求周期内有效
IRRD[31：0]	输入	指令 TCM 的读数据。在非等待的数据周期内有效
IRSEQ	输出	顺序请求。在请求周期内有效，在等待周期内断言。表示当前周期内的地址是顺序于前一个请求周期内使用的地址。在 ITCM DMA 访问之后的 IRSEQ 无效
IRSIZE[3:0]	输入	指令 TCM 大小。指定了连接的 TCM 存储器物理尺寸的静态配置输入。 0000=无 0011=4KB … 1010=512KB 1011=1MB 值 0001、0010 和 1100 到 1111 是保留的
IRWAIT	输入	指令 TCM 等待状态输入。如果为高电平，则 ITCM 不能在该周期内服务（内核的）请求。在请求周期和顺序等待周期内有效。如果不是请求或等待周期则被忽略

续上表

信号名称	方向	描述
IRWBL[3：0]	输出	指令 TCM 的写数据字节通路指示。在请求周期有效。对于读，设置为 b0000，对于写则表示哪些字节被写入，依赖于地址和访问的大小（字、半字或字节）。IRWBL 的位仅在写操作正发生时被设置，因此当 IRnRW 未被设置则 IRWBL 的所有位也未被设置
IRWD[31：0]	输出	指令 TCM 的写数据。在请求周期内当 IRnRW 为 0 时有效。在等待写周期内有效。

附录B CP15 测试和调试寄存器

本附录描述了 ARM926EJ-S CP15 测试和调试寄存器。包含下面的小节：

- 关于测试和调试寄存器。

B.1 关于测试和调试寄存器

ARM926EJ-S 测试和调试寄存器，CP15 c15，提供附加的设备指定（device-specific）的测试操作。用户可以使用该寄存器访问和控制以下的：

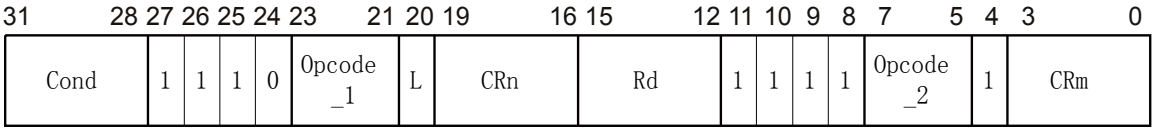
- 调试代理寄存器；
- 调试和测试地址寄存器；
- 跟踪控制寄存器；
- MMU测试操作；
- Cache调试控制寄存器；
- MMU调试控制寄存器；
- 存储器区域重映射寄存器。

用户仅能将这些操作用于测试。《ARM Architecture Reference Manual》将该寄存器描述为实现定义的（implementation-defined）。

CP15 测试和调试操作的格式是：

```
MCR/MRC     p15, <Opcode_1>, <Rd>, c15, <CRm>, <Opcode_2>
```

MRC和MCR位格式见附图B.1。



附图 B.1 CP15 MRC 和 MCR 位格式

L 位区分了 MCR(L=1)和 MRC（L=0）。

B.1.1 调试代理寄存器

用户可以使用调试代理（override）寄存器从默认行为来修改 ARM926EJ-S 内核的行为。

附表B.1表示了每个ARM926EJ-S调试代理寄存器位的功能。

调试代理寄存器可以使用下面的指令来访问：

```
MRC{cond}   p15,0,<Rd>,c15,c0,0            ;Read Debug Override Register
MCR{cond}   p15,0,<Rd>,c15,c0,0            ;Write Debug Override Register
```

调试代理寄存器的复位状态是 0x0。

附表 B.1　调试代理寄存器

位	功能或名称	描述
[31：20]	保留	读=不可预测的 写=应该为 0（SBZ）
[19]	测试和清理全部	0=测试和清理指令的默认行为 1=修改测试和清理，测试，清理和清除指令的行为以便它们能够对整个 cache 起作用
[18]	中止数据 TLB 未命中	0=不中止 DTLB 未命中 1=中止 DTLB 未命中
[17]	中止指令 TLB 未命中	0=不中止 ITLB 未命中 1=中止 ITLB 未命中
[16]	禁能 NC 指令预取	0=使能预取 1=禁能预取
[15]	禁能模块级时钟门控	0=使能模块级（block-level）时钟门控 1=禁能模块级时钟门控
[14]	禁能 NCB 存储（强制 NCNB）	0=使能 NCB 存储 1=禁能 NCB 存储（强制 NCNB）
[13]	MMU 禁能，DCache 使能行为	0=如果 MMU 禁能，则一级访问为 NCNB 1=如果 MMU 禁能且 DCache 被使能，一级访问为 WT
[12：0]	保留	读=不可预测的 写=应该为 0

位 13，MMU 禁能，DCache 使能行为

该位改变了当 MMU 禁能时的行为，但 DCache 被使能。在正常操作时，如果 MMU 被禁能，所有数据访问被视为 NCNB。如果位 13 被置位且 MMU 被禁能，而 DCache 被使能，所有数据访问被视为 WT。

注意：当使用存储器区域寄存器时该行为可以被忽视。

位 14，禁止 NCB 存储（强制 NCNB）

用户可以使用该位来强制所有一级 NCB 存储被视为 NCNB 存储。该位凌驾于 MMU 页表和存储器区域重映射寄存器中的设置。

位 15，禁止模块级时钟门控

用户可以使用该位来禁能 ARM926EJ-S 处理器的模块级时钟门控。该位并不影响 ARM926EJ-S 处理器的功能。它允许了模块级时钟门控评估的好处，而不需要建立两个不同的 ARM926EJ-S 宏单元的实现，一个有模块级时钟门控，而一个没有。

位 16，禁能 NC 指令预取

用户可以使用该位来禁能在存储器非高速缓存区域中的指令推测性的预取。ARM926EJ-S 处理器的默认行为是在 AHB 接口上执行推测性的顺序指令读取。禁能预取阻止了 ARM926EJ-S 存储器系统的任何推测性的非高速缓存的指令预取，且只有 ARM9EJ-S 内核发出的指令请求才导致 AHB 接口上的指令读取。

位 17&18，中止指令 TLB 未命中

用户可以使用中止数据 TLB 未命中和中止指令 TLB 未命中位来阻止由于 TLB 未命中导致的页表搜索的发生。当置位时，TLB 的未命中导致了访问将被中止，就像访问导致了一个转换错误，且值 0000 被写入到对应的 FSR 状态域中一样。

位 19，测试和清理全部

用户可以使用测试和清理全部（test-and-clean-all）位来修改测试和清理，测试清理和清除指令的行为以便单个指令可以被用来清理或清理并清除整个 cache。这仅打算供调试器使用，以提供使用扫描链 15 来高效清理数据 cache 的方式。

B.1.2 调试和测试地址寄存器

该寄存器定义了用于调试和测试操作的地址，以及用 MMU 测试寄存器进行 MMU 测试操作的地址。

用户可以使用下面的指令访问调试和测试地址寄存器：

```
MRC{cond}   p15,0,<Rd>,c15,c1,0         ;Read Debug and Test Address Register
MCR{cond}   p15,0,<Rd>,c15,c1,0         ;Write Debug and Test Address Register
```

B.1.3 跟踪控制寄存器

用户可以使用下面的指令访问跟踪控制寄存器：

```
MCR     p15, 1, <Rd>, c15, c1, 0        ;Write Trace Control Register
MRC     p15, 1, <Rd>, c15, c1, 0        ;Read Trace Control Register
```

用户可以使用跟踪控制寄存器来确定当 FIFOFULL 信号被断言时在哪种条件下 ARM9EJ-S 内核被暂停。

通常，非入侵的实时跟踪需要 nFIQ 或 nIRQ 中断的出现来阻止 ARM9EJ-S 内核在 FIFOFULL 被断言时被暂停。

跟踪控制寄存器让用户能够修改这个行为，因此中断的出现并不阻止 ARM9EJ-S 内核在 FIFOFULL 被断言时被暂停。

附表B.2表示了跟踪控制寄存器的位分配。该寄存器的位[2：1]被复位为 0。

附表 B.2　跟踪控制寄存器位分配

位	内容
[31：3]	保留（应该为 0）
[2]	1=FIQ 中断并不阻止 FIFOFULL 暂停 ARM9EJ-S 内核 0=FIQ 中断阻止 FIFOFULL 暂停 ARM9EJ-S 内核
[1]	1=IRQ 中断并不阻止 FIFOFULL 暂停 ARM9EJ-S 内核 0=IRQ 中断阻止 FIFOFULL 暂停 ARM9EJ-S 内核
[0]	保留（应该为 0）

B.1.4 MMU 测试操作

MMU 测试操作支持访问 MMU 内的 TLB 结构体，并且和调试和测试地址寄存器联合使用。

用户可以使用附表B.3中的指令访问MMU测试操作。

附表 B.3　MMU 测试操作指令

指令		操作
MRC	p15, 4/5, <Rd>, c15, c2, 0	读主 TLB 条目中的标签
MCR	p15, 4/5, <Rd>, c15, c3, 0	写主 TLB 条目中的标签
MRC	p15, 4/5, <Rd>, c15, c4, 0	读主 TLB 条目中的 PA 和访问许可数据
MCR	p15, 4/5, <Rd>, c15, c5, 0	写主 TLB 条目中的 PA 和访问许可数据
MCR	p15, 4/5, <Rd>, c15, c7, 0	转换主 TLB 条目到 RAM 中
MRC	P15, 4/5, <Rd>, c15, c2, 1	读锁定 TLB 条目中的标签
MCR	P15, 4/5, <Rd>, c15, c3, 1	写锁定 TLB 条目中的标签
MRC	P15, 4/5, <Rd>, c15, c4, 1	读锁定 TLB 条目中的 PA 和访问许可数据
MCR	P15, 4/5, <Rd>, c15, c5, 1	写锁定 TLB 条目中的 PA 和访问许可数据
MCR	P15, 4/5, <Rd>, c15, c7, 1	转换锁定 TLB 条目到 RAM 中

在主 TLB 中插入或读取条目

使用这个步骤来访问在主 TLB 中的条目：

1. 使用下面的调试和测试地址寄存器指令来访问一个主 TLB 条目：

```
MCR     p15, 0, <Rd>, c15, c1, 0        ; select TLB entry
```

Rd寄存器选择了主TLB条目，如附图B.2所示：

附图 B.2　选择主 TLB 条目的 Rd 格式

附表B.4描述了Rd寄存器的条目选择位域。

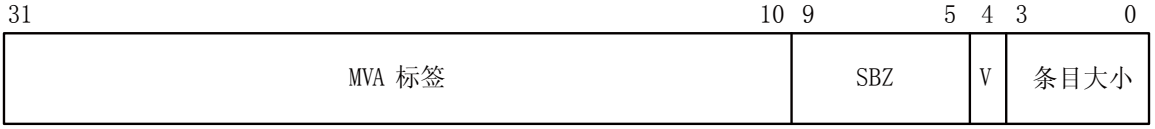
附表 B.4　主 TLB 条目选择位域的编码

位	名称	定义
[31]	路	路选择： 1=路 1 0=路 0
[30：15]	-	应该为 0
[14：10]	被索引的条目	在主 TLB 中被索引的条目
[9：0]	-	应该为 0

2. 使用下面的 MMU 测试操作指令来访问 MVA 标签：

```
MRC     p15, 4/5, <Rd>, c15, c2, 0      ;read tag in main TLB
MCR     p15, 4/5, <Rd>, c15, c3, 0      ;write tag in main TLB
```

Rd寄存器包含着如附图B.3所示的读或写数据。



附图 B.3　访问主或锁定 TLB 条目的 MVA 标签的 Rd 格式

附表B.5描述了Rd寄存器中的MVA标签位域。

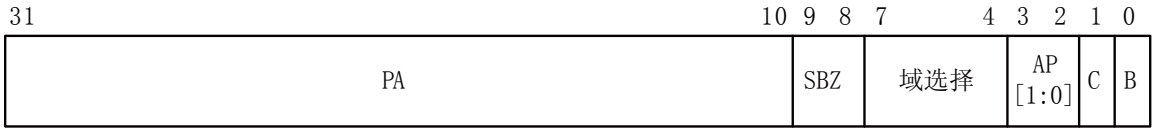
附表 B.5　TLB MVA 标签位域的编码

位	名称	定义
[31：10]	MVA 标签	修改过的虚拟地址
[9：5]	-	应该为 0
[4]	V	有效位
[3：0]	条目大小	条目的大小： b0001=1KB 页或 4KB 页的 1KB 子页 b0011=4KB 页 b0101=64KB 页的 16KB 子页 b0111=64KB 页 b1011=1MB 节

3. 使用下面的 MMU 测试寄存器指令来访问 PA 和访问许可数据：

```
MRC     p15, 4/5, <Rd>, c15, c4, 0 ;read PA and access permission data
MCR     p15, 4/5, <Rd>, c15, c5, 0 ;write PA and access permission data
```

Rd寄存器包含着如附图B.4所示的读或写数据。



附图 B.4　访问主或锁定 TLB 条目的 PA 和 AP 数据的 Rd 格式

附表B.6描述了在Rd寄存器中PA和访问许可位域。

附表 B.6　TLB MVA 标签位域的编码

位	名称	定义
[31：10]	PA	物理地址
[9：8]	-	应该为 0
[7：4]	域选择	域选择： b0000=D0 b0001=D1 … b1110=D14 b1111=D15

续上表

位	名称	定义
[3：2]	AP	访问许可： b00=无访问 b01=特权模式，读/写。用户模式，无访问 b10=特权模式，读/写。用户模式，只读 b11=特权模式，读/写。用户模式，读/写
[1]	C	可高速缓存的位
[0]	B	可缓冲的位

4. 使用下面的指令来完成写一个条目：

```
MCR     p15, 4/5, Rd, c15, c7, 0        ;transfer main storage into RAM
```

要写一个条目到 2 路的主 TLB 中，因而完整的序列是：

```
MCR     p15, 4/5, <Rd>, c15, c3, 0      ;write tag main TLB storage reg
MCR     p15, 4/5, <Rd>, c15, c5, 0      ;write PA/PROT main TLB storage reg
MCR     p15, 4/5, <Rd>, c15, c7, 0      ;transfer main storage into RAM
```

要从 2 路主 TLB 中读取一个条目，该条目必须首先被写入。然后该条目可以使用下面的指令读出：

```
MRC     p15, 4/5, <Rd>, c15, c2, 0      ;read tag main TLB
MRC     p15, 4/5, <Rd>, c15, c4, 0      ;read PA/PROT main TLB
```

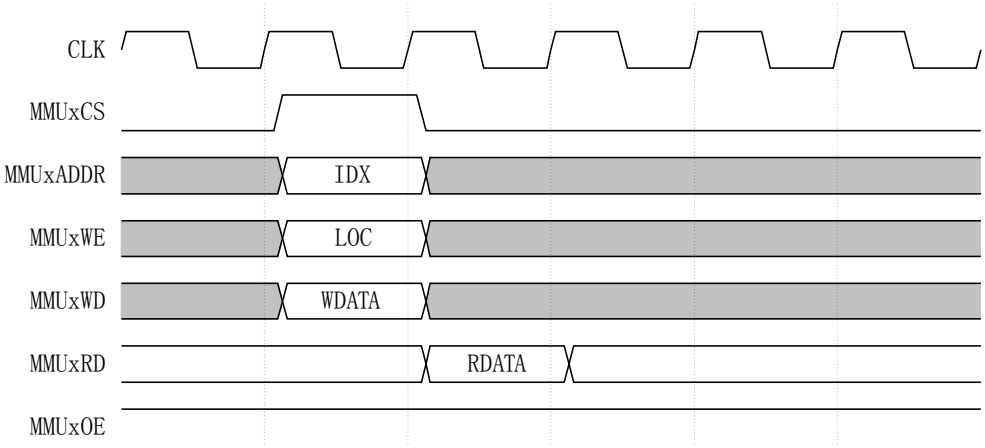
连接到主 MMU 的数据 RAM 是 112 位宽的。对主 TLB 用于 TAG 的写到数据 RAM 的映射见附表B.7并会如附表B.7所示一样出现在MMUxWD[111：0]上。

附表 B.7　主 TLB 映射到 MMUxWD

路	MMUxWD 位	描述
1	[111：90]	TAG[31：10]
	[89：86]	条目大小
	[85：64]	PA[31：10]
	[63：60]	域选择[3：0]
	[59：58]	AP[1：0]
	[57]	可高速缓存的位
	[56]	可缓冲的位
0	[55：34]	TAG[31：10]
	[33：30]	条目大小
	[29：8]	PA[31：10]
	[7：4]	域选择[3：0]
	[3：2]	AP[1：0]
	[1]	可高速缓存的位
	[0]	可缓冲的位

在写操作期间，数据被复制因此每个路接收到相同的数据拷贝。被写入的准确的路和该路准确的索引在测试和调试地址寄存器中指定。

附图B.5表示了到连接着主MMU的数据RAM的写操作期间所发生的事件。



附图 B.5　到数据 RAM 的写

注意：当 MMUxCS=1 时在时钟的上升沿，在 MMUxWD 上的数据被写入到数据 RAM。准确的索引在 MMUxADDR 上，由测试和调试地址寄存器指定。写入的通路由 MMUxWE[3：0]引脚控制。映射如下：

MMUxWE[0]：0=读，1=写 MMUxWD[29：0]到 RAM；MMUxWE[1]：0=读，1=写 MMUxWD[55：30]到 RAM；MMUxWE[2]：0=读，1=写 MMUxWD[85：57]到 RAM；MMUxWE[3]：0=读，1=写 MMUxWD[111：86]到 RAM。在主 MMU 条件下，输出使能 MMUxOE 一直被驱动。MMUxRD 必须被一直强驱动。当执行一个读操作时控制器从 MMUxRD 数据总线采样数据。

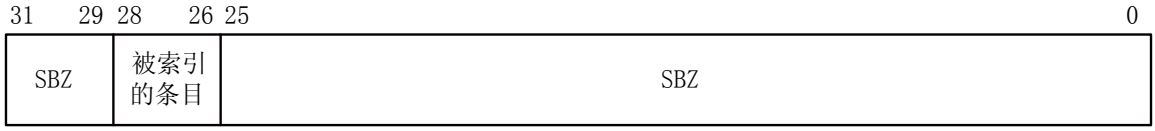
在锁定 TLB 中插入或读取条目

使用这个步骤来访问在锁定 TLB 中的条目：

1. 使用下面的调试和测试地址寄存器指令来访问一个锁定的 TLB 条目：

```
MCR     p15, 0, <Rd>, c15, c1, 0
```

Rd寄存器选择了如附图B.6所示的锁定TLB条目。



附图 B.6　选择锁定 TLB 条目的 Rd 格式

附表 B.8 描述了 Rd 寄存器中的条目选择位域

附表 B.8　锁定 TLB 条目选择位域的编码

位	名称	定义
[31：29]	-	应该为 0
[28：26]	被索引的条目	在锁定 TLB 中被索引的条目
[25：0]	-	应该为 0

2. 使用下面的 MMU 测试寄存器指令来访问 MVA 标签：

```
MRC     p15, 4, <Rd>, c15, c2, 1        ;read lockdown TLB
MCR     p15, 4, <Rd>, c15, c3, 1        ;write lockdown TLB
```

Rd寄存器中的读或写数据见附图B.3。

3. 使用下面的 MMU 测试寄存器指令来读或写 PA 和访问许可数据：

```
MRC     p15, 4, <Rd>, c15, c4, 1    ;read PA and access permission data
MCR     p15, 4, <Rd>, c15, c5, 1    ;write PA and access permission data
```

Rd寄存器中的读或写数据见附图B.4。

4. 使用下面的指令来完成到一个条目的写：

```
MCR     p15, 4, <Rd>, c15, c7, 1    ;transfer lockdown storage into RAM
```

要写一个条目到锁定 TLB 中，因而完整的序列是：

```
MCR     p15, 4/5, <Rd>, c15, c3, 1 ; write tag lockdown TLB storage reg
MCR     p15, 4/5, <Rd>, c15, c5, 1 ; write PA/PROT lockdown TLB storage reg
MCR     p15, 4/5, <Rd>, c15, c7, 1 ; transfer lockdown storage into RAM
```

要从锁定 TLB 中读一个条目，该条目必须首先被写入。然后该条目可以使用下面的指令读出：

```
MRC     p15, 4/5, <Rd>, c15, c2, 1 ;read tag lockdown TLB
MRC     p15, 4/5, <Rd>, c15, c4, 1 ;read PA/PROT lockdown TLB
```

要写入或读出的数据被放在ARM寄存器Rd中，格式见附图B.4。

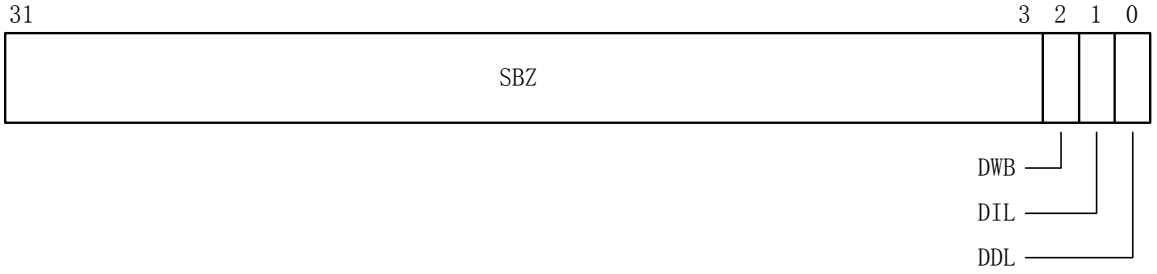
B.1.5 Cache 调试控制寄存器

Cache 调试控制寄存器被用来强加调试所需要的指定 cache 行为。

下面的指令可以用来访问 Cache 调试控制寄存器：

```
MRC{cond}   p15,7,<Rd>,c15,c0,0         ;read cache debug control register
MCR{cond}   p15,7,<Rd>,c15,c0,0         ;write cache debug control register
```

附图B.7表示了Cache调试控制寄存器的格式。



附图 B.7　Cache 调试控制寄存器格式

附表B.9表示了Cache调试控制寄存器的位分配。Cache调试控制寄存器的复位值是 0x0。

附表 B.9　Cache 调试控制寄存器位分配

位	名称	功能	描述
[31：3]	-	保留	读=不可预测的 写=应该为 0
[2]	DWB	禁能写回（强制 WT）	0=使能写回行为 1=强制写通行为
[1]	DIL	禁能 ICache 行填充	0=使能 ICache 行填充 1=禁能 ICache 行填充
[0]	DDL	禁能 DCache 行填充	0=使能 DCache 行填充 1=禁能 DCache 行填充

强制写通（write-through）行为

设置 DWB 位为 1 将强制 DCache 把所有可高速缓存的访问都视作它们在一个内存的写通区域一样。DWB 位的设置凌驾于任何在 MMU 页表或存储器区域重映射寄存器中指定的设置。

如果 cache 包含脏的 cache 行，这些行在 DWB 位被设置时仍保持为脏，直到由于一个行填充之后的写回排出（eviction），或一个直接的清理操作它们才被写回。

如果干净的行在 DWB 位被设置的同时被更新则它们不被标记为脏的。这个功能让一个调试器能下载代码或数据到外部存储器，而没有清理部分或全部 DCache 以确保下载的代码或数据已经被写入到外部存储器的需求。

注意：如果 DWB 位被设置，且发出了一个到脏的 cache 行的写，那么该 cache 行和外部存储器都被写入的数据更新。cache 行内的其他条目仍必须被写回到主存中以实现一致性。

禁能 cache 行填充

设置 DDL 和 DIL 位将在（cache）未命中而执行一个行填充时阻止相关的 cache 更新。当置位后，在 cache 未命中时将执行一个行填充，从外部存储器读取八个字，但是 cache 不会被行填充的数据更新。存储器区域映射不变。这种操作模式是调试所需要的，以便由 ARM9EJ-S 内核所看到的存储器镜像，可以用一种非入侵的方式被检查。来自一个可高速缓存的区域的 Cache 命中将从 cache 中读取数据字，而来自一个可高速缓存的区域的 cache 未命中将直接从存储器读取字。

B.1.6 MMU 调试控制寄存器

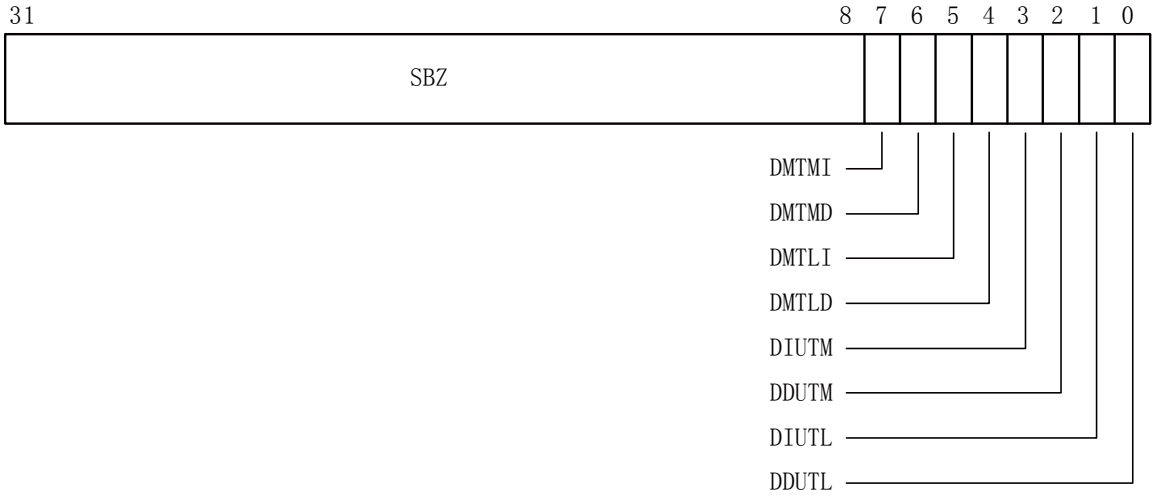
用户可以使用 MMU 调试控制寄存器来在调试期间保护 TLB 和微 TLB 条目。对于非入侵的调试，位[5：0]必须在改变任何其它的 CP15 寄存器或发出任何系统速度的加载或存储操作之前被设置为 b11111。如果主 TLB 加载被禁能，页表搜索仍会发生，但作为结果的数据将被转送到 TLB 周围。

有时可能需要临时改变一个页表条目的内容来促进调试操作。使用位 6 或 7 禁能主 TLB 匹配能够让修改过页表内容被用于访问而不必清除主 TLB 中的任何条目。

用户可以使用下面的指令访问 MMU 调试控制寄存器：

```
MRC{cond}   p15,7,<Rd>,c15,c1,0          ;read MMU debug control register
MCR{cond}   p15,7,<Rd>,c15,c1,0          ;write MMU debug control register
```

附图B.8表示了MMU调试控制寄存器的格式。



附图 B.8　MMU 调试控制寄存器格式

附表B.10表示了MMU调试控制寄存器的位分配。MMU调试控制寄存器的复位值是 0x0。

附表 B.10　MMU 调试控制寄存器位分配

位	名称	功能	描述
[31：8]	-	保留	读=不可预测的 写=应该为 0
[7]	DMTMI	禁能主 TLB 的指令读取匹配	0=使能匹配 1=禁能匹配
[6]	DMTMD	禁能主 TLB 的数据访问匹配	0=使能匹配 1=禁能匹配
[5]	DMTLI	禁能由于指令读取未命中导致的主 TLB 加载	0=使能加载 1=禁能加载
[4]	DMTLD	禁能由于数据访问未命中导致的主 TLB 加载	0=使能加载 1=禁能加载
[3]	DIUTM	禁能指令微（micro）TLB 匹配	0=使能 I-micro TLB 匹配 1=禁能 I-micro TLB 匹配
[2]	DDUTM	禁能数据微 TLB 匹配	0=使能 D-micro TLB 匹配 1=禁能 D-micro TLB 匹配
[1]	DIUTL	禁能指令微 TLB 加载	0=使能 I-micro TLB 匹配 1=禁能 I-micro TLB 匹配
[0]	DDUTL	禁能数据微 TLB 加载	0=使能 D-micro TLB 匹配 1=禁能 D-micro TLB 匹配

B.1.7 存储器区域重映射寄存器

可读/写的存储器区域重映射寄存器凌驾于在 MMU 页表中指定的设置，和在 MMU 被禁能时的默认行为。

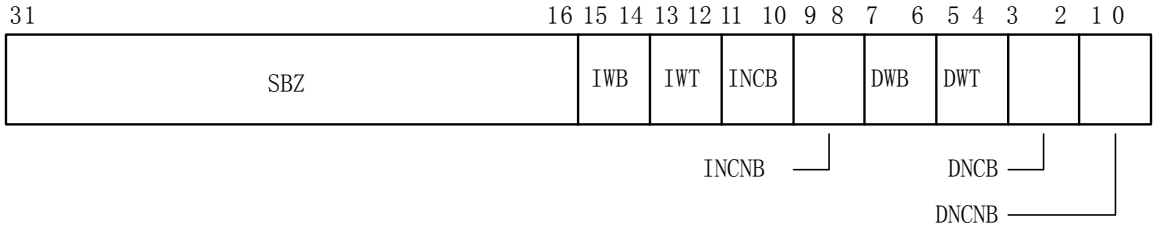
存储器区域寄存器有四个用于重映射指令端存储器区域的域和四个用于重映射数据端存储器区域的域。

用户可以根据附表B.11中的指令访问存储器区域重映射寄存器。

附表 B.11　存储器区域重映射寄存器指令

指令		操作
MRC	p15, 0, Rd, c15, c2, 0	读存储器区域重映射寄存器
MCR	p15, 0, Rd, c15, c2, 0	写存储器区域重映射寄存器

附图B.9表示了存储器区域重映射寄存器的位域。



附图 B.9　存储器区域重映射寄存器格式

附表B.12描述了存储器区域重映射寄存器的位域。

附表 B.12　存储器区域重映射寄存器的编码

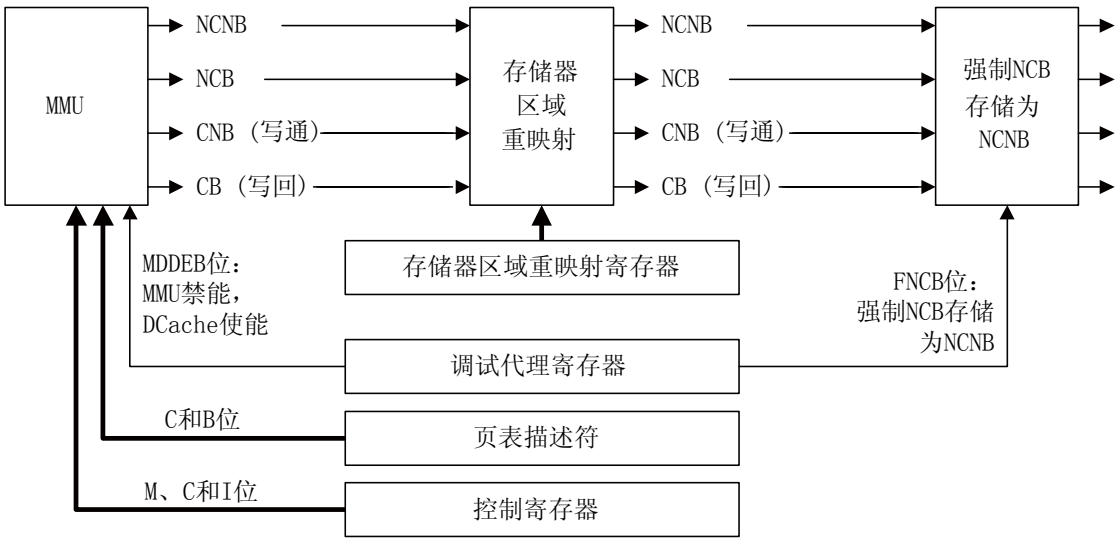
位	名称	描述	复位值
[31：16]	-	应该为 0	0x0000
[15：14]	IWB	指令端写回区域的重映射选择位	b11
[13：12]	IWT	指令端写通区域的重映射选择位	b10
[11：10]	INCB	指令端非高速缓存的可缓冲的区域的重映射选择位	b01
[9：8]	INCNB	指令端非高速缓存的非缓冲的区域的重映射选择位	b00
[7：6]	DWB	数据端写回区域的重映射选择位	b11
[5：4]	DWT	数据端写通区域的重映射选择位	b10
[3：2]	DNCB	数据端非高速缓存的可缓冲的区域的重映射选择位	b01
[1：0]	DNCNB	数据端非高速缓存的非缓冲的区域的重映射选择位	b00

附表B.13表示了每个重映射域的编码。

附表 B.13　重映射域的编码

编码	定义
b00	非高速缓存的非缓冲的
b01	非高速缓存的可缓冲的
b10	写通
b11	写回

附图B.10表示了在分解一个存储器引用的可高速缓存和可缓冲属性时的流程和CP15 c15 的优先权。



附图 B.10　存储器区域属性分解

附录C 术语表

本术语表描述了在本手册中使用的一些术语。其中术语可以有多个含义，在这里出现的含义即为（术语）所打算的。

Abort 中止。一种提示内核必须停止所尝试地非法的存储器访问地执行的机制。中止可以由外部或内部的存储器系统引起，作为尝试访问无效的指令或数据存储器的结果。中止被分类为预取指或数据中止，以及内部或外部中止。

参见 Data Abort，External Abort 和 Prefetch Abort。

Abort model 中止模型。中止模型是 ARM 处理器在响应数据中止异常中的定义行为。不同的中止模型行为不同，反映在加载和存储的指定基址寄存器写回的指令。

Access permission 访问许可。控制着一个任务或进程是否被允许访问存储器的节或页的机制。如果一个访问被尝试用到没有所要求许可的存储器区域，则出现一个许可错误。

Addressing modes 寻址模式。一种机制，被许多不同的指令共享，用于产生指令所使用的值。对于 ARM 的四种寻址模式，产生的值是存储器地址（这是寻址模式的传统角色）。第五种寻址模式产生的值被用作数据处理指令的操作数。

Advanced High-performance Bus（AHB）

高级高性能总线（AHB）。AMBA 高级高性能总线系统将诸如 ARM 内核的嵌入式处理器连接到高性能的外设，DMA 控制器，片上存储器和接口。它是一个高速，高带宽总线，支持多主机总线管理以优化系统系能。

参见 Advanced Microcontroller Bus Architecture 和 AHB-Lite。

Advanced Microcontroller Bus Architecture（AMBA）

高级微控制器总线体系（AMBA）。AMBA 是 ARM 的开放标准，用于多主机片上总线，能够在多主机和从机的条件下运行。它是一个片上总线规范，描述了用于互联和管理组成一个片上系统（SoC）的功能模块的策略。它有助于拥有一个或多个 CPU 或信号处理器和多个外设的嵌入式处理器的开发。通过定义 SoC 模型的通用骨架 AMBA 满足重复使用的设计方法。AHB 遵从该标准。

Advanced Peripheral Bus（APB）

高级外设总线（APB）。AMBA 高级外设总线对比于 AHB 是简单的总线协议。它被设计用于辅助或通用的外设，比如定时器，中断控制器，UART 和 I/O 端口。通过一个系统到外设总线桥接器连接到主系统总线，有助于降低系统功耗。

参见 Advanced High-perfmance Bus。

AHB 见 Advanced High-perfmance Bus。

Aligned 对齐。对齐的数据条目被存储以至于它们的地址是可被 2 的最高乘幂整除的，数 2 划分了它们的大小。对齐的字和半字有可以被 4 和 2 分别整除的地址。

因此术语字对齐和半字对齐规定地址是可被 4 和 2 各自整除的。其它相关的术语以相似的方式被定义。

AMBA 见 Advanced Microcontroller Bus Architecture。

AP 见 Access permission。

APB 见 Advanced Peripheral Bus。

Application Specfic Integrated Circuit（ASIC）

应用特定的集成电路（ASIC）。一个集成电路，被设计为执行特定的应用功能。可以是定制的或大规模量产的。

Application Specfic Standard Part/Product（ASSP）

应用特定的标准部件/产品（ASSP）。一个集成电路，被设计为执行特定的应用功能。通常由两个或多个独立的电路功能组合为构建的模块，适合用在一个或多个特定应用市场的不同种类的产品中。

Architecture 体系。硬件和/或软件的组织，以处理器和它附属的部件为特征，并且在描述它们的行为时能够让有类似特征的设备被分组到一起，例如，哈佛体系，指令集体系，ARMv6 体系。

ARM instruction ARM 指令。是一个指定了 ARM 处理器将要执行的操作的字。ARM 指令必须是字对齐的。

ARM state ARM 状态。一个正在执行 ARM（32 位）字对齐的指令的处理器即为运行在 ARM 状态。

ASIC 见 Application Specific Integrated Circuit。

ASSP 见 Application Specific Standard Part/Product。

ATPG 见 Automatic Test Pattern Generation。

Automatic Test Pattern Generation（ATPG）

自动测试模式生成。使用一个特殊的软件工具，自动产生用于 ASIC 设计的制造业的测试向量的过程。

Back-annotation 逆向注解。从实现处理中应用时序特征到一个模型上的过程。

Banked registers 分组的寄存器。由当前处理器模式定义使用的那些物理寄存器。分组的寄存器是 r8 到 r14。

Base register 基址寄存器。由加载或存储指令指定的寄存器，用来保持指令的地址计算中的基地址。取决于指令和它的寻址模式，基址寄存器的值可以加上或减去一个偏移量以形成被送到存储器的虚拟地址。

Base register write-back

基址寄存器写回。更新用于指令目标地址计算的基址寄存器的内容，因此修改过的地址被改变到存储器中稍高或稍低的顺序地址。这意味着对于连续的指令传输和使能到顺序存储器的快速突发访问时没有必要读取目标地址。

Beat 拍。突发中单个传输的字。例如，一个 INCR4 突发由四个拍组成。

参见 Burst。

Big-endian 大端。字节顺序方案，一个数据字中降权重的字节被存储在存储器的递增地址中。

参见 Little-endian 和 Endianness。

Big-endian memory

大端存储器。存储器：在一个字对齐的地址处的字节或半字是在该地址处的字内的最高权位的字节或半字；在一个半字对齐的地址处的字节是在该地址处的半字内的最高权位字节。

参见 Little-endian memory。

Block address 块地址。由一个标签，一个索引和一个字域构成的地址。标签位识别包含着 cache 命中时匹配的 cache 条目的路。索引位识别被寻址的组。字域包含着用来识别 cache 条目内的字、半字或字节的字地址。

参见本术语表最后一页的 Cache 术语学图。

Boundary scan chain

边界扫描链。边界扫描链由串行连接的设备组成，这些设备实现了使用标准 JTAG TAP 接口的边界扫描技术。每个设备包含至少一个 TAP 控制器，TAP 控制器包含构成了连接在 TDI 和 TDO 之间的链的移位寄存器，测试数据通过移位寄存器被移动。处理器可以包含多个移位寄存器，让用户能够访问设备中被选择的部分。

Breakpoint 断点。断点是一种由调试器提供的识别在程序的执行将被停止处的指令的机制。断点被程序员插入以使能在程序执行的固定位置中寄存器内容、存储单元、变量值的检查，用来测试程序是否运行正确。断点在程序测试成功之后被移除。

参见 Watchpoint

Burst 突发。到连续地址的一组传输。由于地址是连续的，没有必要为第一个传输之后的任一传输提供地址。这增加了在组群传输发生时的速度。在 AHB 总线上的突发使用 HBURST 信号控制，以指定传输是单个、四拍、八拍或 16 拍的突发，以及指定地址是如何增加的。

参见 Beat。

Bus Interface Unit 总线接口单元。总线接口单元（BIU）控制着通过 AHB 的所有数据访问。它仲裁和调度 AHB 请求。

Byte 字节。一个 8 位数据条目。

Cache 高速缓存。一块片上或片外快速访问的存储器区域，位于处理器和主存之间，被用来存储和重新获取经常使用的指令和/或数据的拷贝。这用来显著降低存储器访问的平均速度和提升处理器性能。

参见本术语表最后一页的 Cache 术语图。

Cache contention Cacne 竞争。当经常使用的存储器 cache 行（这些 cache 行使用特定的 cache 组）数超出了 cache 的组相连。在这种情况下，主存活动性增加且性能降低。

Cache hit Cache 命中。由于指令或数据的地址已经保持在 cache 中，所以存储器访问可以被高速的处理。

Cache line Cache 行。cache 内的基本存储单元。在大小上总是两个字的乘幂（通常是四或八个字），并且需要被对齐到一个合适的存储器边界。
参见本术语表最后一页的 Cache 术语图。

Cache line index Cache 行索引。在一个 cache 路中与每个 cache 行相关的数字。在每个 cache 路内，cache 行从 0 到 set associativty（组相连）－1 编号。
参见本术语表最后一页的 Cache 术语图。

Cache lockdown Cache 锁定。为固定 cache 存储器中的一行以便让它不能被替换。Cache 锁定让关键的指令和/或数据能够被加载到 cache 中，以便包含它们的 cache 行之后不会被重新分配。这确保了所有后继到相关的指令/数据的访问都是 cache 命中，因此能够尽快的完成。

Cache miss Cache 未命中。不能以最快速度处理的一次存储器访问，因为访问所寻址的指令/数据不在 cache 中，并且需要一次主存访问。

Cache set Cache 组。cache 组是一组 cache 行（或块）。一个组包含可用相同索引寻址的所有路。cache 组的数目总是 2 的乘幂。
参见本术语表最后一页的 Cache 术语图。

Cache way Cache 路。一组 cache 行（或块）。它的值是 2 为基数，大小域中索引位的数目为指数的乘幂。
参见本术语表最后一页的 Cache 术语图。

CAM 见 Content Addressable Memory。

Cast Out 见 Victim。

Clean 清理。一个位于 cache 中时没有被修改过的 cache 行被认为是干净的。要清理一个 cache 既是将脏的 cache 条目写入主存中。如果一个 cache 行是干净的，在 cache 未命中时它不被写入，因为下一级的存储器中包含着和该 cache 内一样的数据。
参见 Dirty。

Clock gating 时钟门控。用一个控制信号和更改过的时钟来选通一个宏单元的时钟信号，作为结果的时钟控制着宏单元的运行。

Clocks Per Instruction
见 Cycles Per Instruction。

Coherency 见 Memory coherency。

Cold reset 冷复位。也被称为上电复位。通过打开电源来启动处理器。关闭电源然后再上电来清理主存和许多的内部设置。一些程序失效能够锁定处理器并要求一个冷复位以重新使用系统。在其它情况下，仅需要一个热复位。
参见 Warm reset。

Communications channel

通信通道。用于运行在处理器上的软件与一个外部主机之间通信的硬件，使用调试接口。当该通信是用于调试目的，它被称为调试通信通道。在一个 ARMv6 兼容的内核中，通信通道包括数据传输寄存器，一些数据状态和控制寄存器的位，以及外部调试接口控制器，比如在 JTAG 接口条件下的 DBGTAP 控制器。

Condensed Reference Format（CRF）

浓缩基准格式。一种 ARM 私有的指定测试向量的文件格式。

Condition field 条件域。指令中的一个 4bit 域，用来指定指令可以执行的条件。

Conditional execution

条件执行。如果在指令开始执行时条件码标志表示相应的条件为真，那么指令正常执行。否则，该指令不做任何事情。

Content Addressable Memory（CAM）

内容寻址的存储器。通过内容而被识别的存储器。内容寻址的存储器被用在 CAM-RAM 体系的 cache 中，存储 cache 条目的标签。

CAM 包含和每个存储的位的比较逻辑。数据值被广播到所有存储的字上并和其中的值比较。匹配的字用某种方式标志出来。后继的操作可以在被标记的字上进行。一次读出一个被标记的字或写入特定位置的位到所有的字是可能的。

Contex 上下文。多任务操作系统中每个进程运行中的环境。在 ARM 处理器中，上下文被限制，以平均它能在存储器中寻址的物理地址范围和相关的存储器访问许可。

参见 Fast context switch。

Control bits 控制位。程序状态寄存器（PSR）的低 8 位。当异常发生时控制位会改变，并且仅当处理器处于特权模式时可被修改。

Coprocessor 协处理器。补充主处理器的一个处理器。它执行主处理器不能执行的附加功能。通常用于浮点数学运算，信号处理或存储器管理。

Copy back 见 Write-back。

Core 内核。内核是处理器的一部分，包含 ALU，数据通道，通用寄存器，程序计数器，指令译码和控制电路。

Core module 内核模块。在 ARM 综合器的环境中，内核模块是一个另外加入的包含一个 ARM 处理器和局部存储器的开发板。内核模块可以独立运行，或被堆叠到综合器母版中。

Core reset 见 Warm reset。

CPI 见 Cycles per instruction。

CPSR 见 Current Program Status Register。

CRF 见 Condensed Reference Format。

Current Program Status Register（CPSR）

当前程序状态寄存器。保持着当前运行的处理器状态的寄存器。

Cycles Per instruction（CPI）

每指令周期数。每指令周期数（或每指令时钟数）是能在一个时钟周期内被执行的计算机指令数的测量方法。这个性能指标能被用来比较不同 CPU 之间的性能。该值越低，性能越高。

Data Abort 数据中止。从存储器系统到内核的指示，内核必须停止所尝试的非法的存储器访问的执行。数据中止是尝试访问无效的数据存储器（的结果）。

参见 Abort，External Abort，和 Prefetch Abort。

Data cache 数据 cache。一块片上可快速访问的存储器区域，位于处理器和主存之间，被用来存放和重新获取经常使用的数据的拷贝。这用来显著降低存储器访问的平均速度和提升处理器性能。

DBGTAP 见 Debug Test Access Port。

DCache 数据 cache。一块片上可快速访问的存储器区域，位于处理器和主存之间，被用来存放和重新获取经常使用的数据的拷贝。这用来显著降低存储器访问的平均速度和提升处理器性能。

Debugger 调试器。调试系统，包括一个用来检测，定位和修正软件错误的程序，和支持软件调试的定制硬件。

Debug Test Access Port（DBGTAP）

调试测试访问端口。四个代理和一个可选的终端的集合，构成了到 JTAG 边界扫描体系的输入/输出和控制接口。代理终端是 DBGTDI，DBGTDO，DBGTMS 和 TCK。可选的终端是 TRST（DBGnTRST）。该信号是 ARM 内核的代理，因为它被用来复位调试逻辑。

Direct Memory Access（DMA）

直接存储器访问。一种直接访问主存的操作，而不用处理器执行任何到相关数据的访问。

Dirty 脏的。在写回 cache 中的一个 cache 行，当它处于 cache 中时是已经被修改过的，则被认为是脏的。一个 cache 行通过设置脏位而被标记为脏的。如果一个 cache 行是脏的，那么它必须在 cache 未命中时被写入到存储器中，因为下一级存储器包含的数据没有被更新。将脏的数据写回到主存的操作被称为 cache 清理。

参见 Clean。

DMA 见 Direct Memory Access。

DNM 见 Do Not Modify。

Domain 域。节，大页和小页存储器的集合，它们的访问许可通过写域访问控制寄存器（CP15 寄存器 c3）来迅速切换。

Do Not Modify（DNM）

禁止修改。在禁止修改域，数据值禁止通过软件修改。DNM 域读被看做不

可预知的值，并且必须只能写入从相同处理器上相同域中读出的相同值。贯穿本手册，DNM 域后有时会跟着圆括弧内的 RAZ 或 RAO 以表示出于将来的兼容性，这些位应该用哪种方式被读取，但是程序员不能依赖这种行为。

Doubleword 双字。一个 64 位的数据条目。除非作其他声明，数据的容将被视作无符号整数。

Doubleword-aligned

双字对齐。拥有能被 8 整除的存储器地址的数据条目。

EmbeddedICE logic

嵌入式 ICE 逻辑。一个片上逻辑模块，对 ARM 处理器内核提供基于 TAP 的调试支持。使用 JTAG 接口通过在 ARM 内核上的 TAP 控制器访问它。

EmbeddedICE-RT

嵌入式 ICE-RT。由可调试的 ARM 处理器提供的基于 JTAG 的硬件，用于协助实时调试。

Embedded Trace Buffer

嵌入式跟踪缓冲。ETB 使用可配置大小的 RAM 提供跟踪数据的片上存储。

Embedded Trace Macrocell（ETM）

嵌入式跟踪宏单元。一个硬件宏单元，当连接到一个处理器内核时，在一个跟踪端口上输出指令和数据跟踪信息。ETM 提供处理器通过兼容 AHB 协议的跟踪端口来驱动跟踪。

Endianness 端结构。字节顺序。确定一个数据字中连续的字节以何种方式存储在存储器中的方案。（是）系统存储器映射的一个方面。

参见 Little-endian 和 Big-endian。

ETM 见 Embedded Trace Macrocell。

Event 1（简单）一个可观察到的条件，能被 ETM 用来控制跟踪的一个方面。

2（复杂）简单事件的布尔组合，被 ETM 用来控制跟踪的多个方面。

Exception 异常。被认为足够严重而要求程序的执行被中断的一个失效或错误事件。例子包括尝试执行一个无效的存储器访问，外部中断，以及未定义的指令。当一个异常发生时，正常的程序流被中断并且（程序的）执行在对应的异常向量处被重新恢复。异常向量包含了中断处理程序处理异常时的第一条指令。

Exception service routine

见 Interrupt handler。

Exception vector 见 Interrupt vector。

External Abort 外部中止。从外部存储器系统到内核的一个指示，内核必须停止所尝试的非法的存储器访问。外部中止是作为尝试访问无效存储器的结果而被外部存储器系统引起。

参见 Abort，Data Abort 和 Prefetch Abort。

Fast contex switch 快速上下文切换。在多任务系统中，是分配给一个进程的时间片停止而另一

个进程的时间片启动时的时刻。如果进程之间被足够频繁的切换，那么它们能够对用户表现出像在并行运行一样，除此之外能够对可能影响进程的外部事件尽量响应。

在 ARM 处理器中，快速上下文切换由选择一个非零的 PID 值以切换上下文到下一个进程而引起的。快速上下文切换导致由 ARM 处理器产生的、用于存储器访问的每个虚拟地址，生成为一个修改过的虚拟地址，修改过的虚拟地址被送到其余的存储器系统，以替代正常的虚拟地址。对一些 cache 控制操作而言虚拟地址被当做数据被送往存储器系统。在这种情况下不会发生地址修改。

参见 Fast Contex Switch Extension。

Fast Contex Switch Extension（FSCE）

快速上下文切换扩展。ARM 体系的一个扩展，让带 MMU 的有高速缓存的处理器能在不同的软件进程中对其余的存储器系统呈现不同的地址，即使这些进程使用相同的地址。

FCSE 见 Fast Context Switch Extension。

Flat address mapping

平面地址映射。在内存空间中以每个物理地址包含的（内容）和对应的虚拟地址是相同的这样的方式来组织存储器的一个系统。

Fully-associative cache

全相连的 cache。只有一个 cache 组构成整个 cache 的 cache。cache 条目的数目等同于 cache 路的数目。

参见 Direct-mapped cache。

Half-rate clocking（ETM）

半速率时钟（ETM）。跟踪时钟的两分频，因此 TPA 可以在跟踪时钟的上升沿和下降沿采样跟踪数据。半速率时钟的最初目的是对于非常高速的系统降低 ASIC 的跟踪时钟的信号转变速率。

Halfword 半字。一个 16 位的数据条目。

Halt mode 停止模式。两种互斥的调试模式之一。在停止模式中当遇到一个断点或观察点时所有的处理器执行都停止。所有的处理器状态，协处理器状态，存储器和输入输出区域都可以通过 JTAG 接口检查和修改。

参见 Monitor debug-mode。

High vectors 高向量。异常向量的可选位置。高向量地址范围是靠近地址空间的顶部，而不是在底部。

Host 主机。对另外的计算机提供数据和其他服务的一个计算机。特别的，（是指）对一个待调试的目标板提供调试服务的计算机。

ICache 指令 Cache。一块片上可快速访问的存储器区域，位于处理器和主存之间，被用来存放和重新获取经常使用的指令的拷贝。这用来显著降低存储器访问

的平均速度和提升处理器性能。

IGN 见 Ignore。

Ignore（IGN） 忽略。必须忽略存储器写操作。

Illegal instruction 无效指令。体系上未定义的一个指令。

IMB 见 Instruction Memory Barrier。

Implementation-defined

实现定义的。意思是行为没有在体系上定义，但是应该被各自的实现所定义和证实。

Implementation-specfic

实现指定的。意思是行为没有在体系上地你故意，并且不是必须被各自的实现所证实。在有许多实现选项可用并且这些选项的选择并不影响软件兼容性时使用。

Index 见 Cache index。

Index register 索引寄存器。在一些加载或存储指令中指定的寄存器。该寄存器的值被用作一个偏移量而被基址寄存器的值加上或减去，以构成虚拟地址，虚拟地址被送到存储器。一些寻址模式可选择让索引寄存器的值在被加上或减去之前移位。

Instruction cache 指令 cache。一块片上可快速访问的存储器区域，位于处理器和主存之间，被用来存放和重新获取经常使用的指令的拷贝。这用来显著降低存储器访问的平均速度和提升处理器性能。

Instruction cycle count

指令周期计数。一条指令占据流水线的执行状态的周期数。

Instruction Memory Barrier（IMB）

指令内存栅。用来确保预取指缓冲清空所有过时的指令的一个操作。

Internal scan chain

内部扫描链。一连串的寄存器连接在一起以形成贯通器件的一个通路，在生产测试期间被用来导入测试模式到器件的内部节点并导出结果值。

Interrupt handler 中断处理程序。当中断发生时处理器的控制权被送给的目的程序。

Interrupt vector 中断向量。许多固定的地址之一，位于存储器的底部或当高向量被配置时则位于存储器顶部，包含对应的中断处理程序得第一条指令。

Invalidate 清除。通过清除有效位来标记一个 cache 行为无效。这必须在（cache）行不包含有效的 cache 条目时完成。例如，在一个 cache 清空后所有的行都是无效的。

Joint Test Action Group（JTAG）

联合测试行动小组（JTAG）。开发了 IEEE1149.1 标准的组织名称。这个标准定义了用在集成电路器件在线测试中的边界扫描体系。它通常由于词首字母 JTAG 而被熟知。

JTAG 见 Joint Test Action Group。

Line 见 Cache line。

Little-endian 小端。字节顺序方案，一个数据字中升权重的字节被存储在存储器的递增地址中。

参见 Big-endian 和 Endianness。

Little-endian memory

小端存储器。存储器：在一个字对齐的地址处的字节或半字是在该地址处的字内的最小权位的字节或半字；在一个半字对齐的地址处的字节是在该地址处的半字内的最低权位字节。

参见 Big-endian memory。

Load/store architecture

加载/存储体系。一个处理器体系，其中数据处理操作仅操作寄存器上的内容，不直接操作存储器上的内容。

Load Store Unit（LSU）

存储加载单元（LSU）。处理器中处理加载和存储传输的部分。

LSU 见 Load Store Unit。

Macrocell 宏单元。有定义的接口和行为的复杂逻辑模块。一个典型的 VLSI 系统由一些宏单元（诸如一个处理器，EMT 和存储器模块）加上应用指定的逻辑组成。

Memory bank 存储器组。两个或多个交叉存取的存储器分区之一，通常一个字宽度，相比单个字而言能够一次读和写多个字。所有的存储器组都同时寻址并且一个组使能或片选信号确定每个传输中哪个组被访问。到顺序字地址的访问导致了到顺序组的访问。这让访问一个组的相关延时能够在访问它邻近的组时发生，加速了存储器传输。

Memory coherency

存储器一致性。如果通过一个数据读或指令读取而读到的值就是最新近写入到该位置的值时，存储器是一致的。存储器一致性在相关的（地址）有多个可能的物理位置时变得复杂，比如有主存，写缓冲和 cache 的系统。

Memory Management Unit（MMU）

存储器管理单元（MMU）。控制着 cache 和到存储区块的访问许可，并转换虚拟地址为物理地址的硬件。

Memory Protection Unit（MPU）

存储器保护单元（MPU）。控制着到存储区块的访问许可的硬件。和 MMU 不同，MPU 并不转换虚拟地址为物理地址。

Micorprocessor 见 Processor。

Miss 见 Cache miss。

MMU 见 Memory Management Unit。

Modified Virtual Address（MVA）

修改过的虚拟地址（MVA）。由 ARM 处理器产生的虚拟地址可以通过当前进程 ID 被修改以提供一个用于 MMU 和 cache 的修改过的虚拟地址（MVA）。参见 Fast Contex Switch Extension。

Monitor debug-mode

监控调试模式。两种互斥的调试模式之一。在监控调试模式下处理器使能一个由调试监控器或操作系统的调试任务提供的软件中止处理程序。当遇到一个断点或观察点时，这让至关重要的系统中断能继续被服务而普通程序的执行被暂停。

参见 Halt mode。

MPU 见 Memory Protection Unit。

Multi-ICE 用于调试嵌入式系统的一个基于 JTAG 的工具。

MVA 见 Modified Virtual Address。

NCB 见 Noncacheable Buffered。

NCNB 见 Noncacheable Nonbufferable。

Noncacheable Bufferd

非高速缓存的缓冲的。是一个读操作从主存中被执行而不被分配到 cache 中的存储器区域。写操作通过一个写缓冲执行到主存，因此处理器内核的执行可以在写操作完成到主存的同时继续进行。

Noncacheable Nobufferable

非高速缓存的非缓冲的。是一个读操作从主存中被执行而不被分配到 cache 中的存储器区域。写操作被执行到主存中而没有缓冲，因此处理器内核的执行在写操作完成期间被暂停。

PA 见 Physical Address。

Penalty 缺陷。发生在无用的执行阶段流水线活动中的周期数，因为指令流和假想或预测的不同。

Power-on reset 见 Cold reset。

Prefetching 预取指。在流水化的处理器中，是在指令完成执行前从存储器中读取指令来填充流水线的过程。预取一个指令并不意味着该指令必须被执行。

Prefetch Abort 预取指中止。从存储器系统到内核的一个指示，内核必须停止所尝试的非法的存储器访问。预取指中止由于尝试访问无效的指令存储器的结果而被外部或内部存储系统引起。

参见 Data Abort，External Abort 和 Abort。

Processor 处理器。处理器是计算机系统中所要求的用计算机指令来处理数据的电路。它是微处理器的缩写。也需要时钟源，电源供应和主存储器以构建一个最小的完整的可工作计算机系统。

Physical Address（PA）

物理地址（PA）。MMU 对修改过的虚拟地址（MVA）执行一个转换以产生

物理地址，物理地址送给 AHB 以执行一个外部访问。PA 也被存储在数据 cache 中以避免数据被排出 cache 时所必需的地址转换。

参见 Fast Contex Switch Extension。

Read 读。读被定义为有加载的语义的存储器操作。也就是，ARM 指令 LDM，LDRD，LCD，LDR，LDRT，LDRSH，LDRH，LDRSB，LDRB，LDRBT，LDREX，RFE，STREX，SWP 和 SWPB，和 Thumb 指令 LDM，LDR，LDRSH，LDRH，LDRSB，LDRB 和 POP。由硬件加速的 Java 指令能导致许多的读发生，根据 Java 堆栈的状态和 Java 硬件加速的实现。

RealView ICE 使用 JTAG 接口调试嵌入式处理器的一个系统。

Region 区域。指令或数据存储空间的一个分区。

Remapping 重映射。在应用程序开始执行之后改变物理存储器或器件的地址。典型的当初始化完成后这被用来允许 RAM 替代 ROM。

Reserved 保留。控制寄存器或指令格式中如果一些域是由实现定义的，或如果域的内容不为零将产生不可预知的结果，则这些域是被保留的。这些域被保留做将来体系的扩展使用或由实现指定。所有实现中未用的保留位必须写作 0 且读作 0。

Saved Program Status Register（SPSR）

备份的程序状态寄存器。在导致切换到当前模式的异常发生之前立刻保持着任务的 CPSR 的寄存器。

SBO 见 Should Be One。

SBZ 见 Should Be Zero。

SBZP 见 Should Be Zero or Preserved。

Scan chain 扫描链。扫描链由使用标准 JTAG TAP 接口实现边界扫描技术的串联器件组成。每个器件至少包含一个 TAP 控制器，TAP 控制器包含构成连接在 TDI 和 TDO 之间的链的移位寄存器，数据通过它移动。处理器可以包含多个移位寄存器，让用户能够访问被选中的器件部分。

SCREG 在 ARM TAP 控制器中当前被选择的扫描链号。

Set 见 Cache set。

Set-associative cahce

组相连 cache。在一个组相连 cache 中，行仅能被放置在 cache 中对应于存储器地址被组的数目取模的位置。如果一个 cache 有 n 路，则 cache 被称作 n 路组相连。组相连可以是任何大于或等于 1 的数且并不严格限制为 2 的乘幂。

Short vector operation

小向量操作。在产生每个目的的结果中包含多于一个目的寄存器和可能多于一个源寄存器的操作。

Should Be One（SBO）

应该为 1（SBO）。应该被软件写作 1（位域中所有位都为 1）。写 0 将产生不

可预知的结果。

Should Be Zero（SBZ）

应该为 0（SBZ）。应该被软件写作 0（位域中所有位都为 0）。写 1 将产生不可预知的结果。

Should Be Zero or Preserved（SBZP）

应该为 0 或保留（SBZP）。应该被软件写作 0（位域中所有位都为 0），或在相同的处理器上把之前从该域读出的值写回到相同的域来保留。

SPICE 强化集成电路的仿真程序。一个精确的晶体管级电子电路仿真工具，对给定的电路条件可被用来预测等价的实际电路将如何行为。

SPSR 见 Saved Program Status Register。

Tag 标签。一个块地址的较高部分，被用来在 cache 中识别一个 cache 行。来自 CPU 的块地址和一个组中的每个标签并行地作比较以确定对应的行是否在 cache 中。如果存在，则被称为一次 cache 命中并且该行可以从 cache 读取。如果块地址并不对应于任何标签，则被称为一次 cache 未命中而 cache 行必须从下一级存储器读取。

参见本术语表最后一页的 Cache 术语图。

TAP 见 Test access port。

TCM 见 Tightly coupled memory。

Test Access Port（TAP）

测试访问端口（TAP）。四个代理和一个可选终端的集合，它们构成了 JTAG 边界扫描体系的输入输出和控制接口。代理终端是 TDI，TDO，TMS 和 TCK。可选的终端是 TRST。该信号是 ARM 内核的代理，因为它被用来复位调试逻辑。

Thumb instruction

Thumb 指令。指定处于 Thumb 状态下的 ARM 处理器执行的操作的一个半字。Thumb 指令必须是半字对齐的。

Thumb state Thumb 状态。执行 Thumb（16 位）半字对齐指令的处理器是运行在 Thumb 状态下的。

Tightly coupled memory（TCM）

紧耦合存储器（TCM）。低延时的存储器区域，提供可预测的指令执行或数据加载时序，在要求确定的执行的情况下。TCM 适用于保持：

- 关键的程序，比如中断处理程序；
- 中间结果的数据；
- 存放位置不适合 cache 的数据类型；
- 关键的数据结构，比如中断堆栈。

TLB 见 Translation Look-aside Buffer。

Translation Lookaside Buffer（TLB）

转换后备缓冲。包含最近使用的页表条目的一个 cache，避免每次存储器访

问时过多的页表搜索。是存储器管理单元的一部分。

Translation table 转换表。保存在内存中的一个表，包含着定义了多种固定大小的存储区域属性的数据。

Translation table walk

转换表搜索。做一个完整的转换表查询的过程。它是由硬件自动执行的。

Undefined 未定义。表示产生了未定义指令陷阱的一个指令。关于 ARM 异常的更多细节参见《ARM Architecture Reference Manual》。

Unpredictable 不可预测的。意思是 ETM 的行为不能被信任。这种条件没有生效。当应用到事件资源的编程时，只有该事件资源的输出是不可预测的。不可预测的行为能够影响整个系统的行为，因为 ETM 能够导致内核进入调试状态，且外部输出可能被用作其它目的。

Unpredictable 不可预测的。对于读，读该区域时返回的数据是不可预测的。它可以是任何值。对于写，写该区域导致不可预测的行为，或不可预测的器件配置的改变。不可预测的指令不能停止或挂起处理器，或系统的任何部分。

VA 见 Virtual Address。

Victime 替代者。一个 cache 行，被选中并丢弃来为 cache 未命中时需要的 cache 行替换制造空间。被选中用于排出的替换者中的路由处理器指定。替换者也被称为逐出。

Virtual Address（VA）

虚拟地址。MMU 使用它的页表来转换虚拟地址为物理地址。处理器在虚拟地址执行代码，虚拟地址可放在物理存储器的任何位置。

参见 Fast Contex Switch Extension，Modified Virtual Address，和 Physical Address。

Warm reset 热复位。也被称为内核复位。初始化处理器的主体，除了调试控制器和调试逻辑。这种类型的复位在用户使用处理器的调试特征时很有用。

Watchpoint 观察点。观察点是由调试器提供的机制，当一个特定的存储器地址中包含的数据改变时用来停止程序的执行。观察点由编程器插入，当存储器被写入以测试程序运行是否正确时允许检查寄存器内容，存储器区域，和变量值。观察点在程序被成功测试之后被移除。

参见 Breakpoint。

Way 见 Cache way。

WB 见 Write-back。

Word 字。一个 32 位的数据条目。

Write 写。写被定义为有存储的语义的操作。也就是，ARM 指令 SRS，STM，STRD，STC，STRT，STRB，STREX，SWP 和 SWPB，以及 Thumb 指令 STM，STR，STRH，STRB 和 PUSH。由硬件加速的 Java 指令能够导致许多写的发生，根据 Java 堆栈的状态和 Java 硬件加速的实现。

Write-back（WB）

写回。在一个写回 cache 中，数据仅在 cache 未命中之后的行替换被强制逐出 cache 时才被写入到主存。否则，处理器的写仅更新 cache。（也被称作拷贝回）。

Write buffer 写缓冲。一块高速存储器，被组织为 FIFO 缓冲，在数据 cache 和主存之间，其目的是优化到主存的存储。

Write completion 写完成。存储系统指示处理器写已经完成，在存储系统能够保证写的效果能对系统中所有处理器可见的传输点时。这不是写和存储器同步简单的相关，或写到一个器件或顺序要求严格的区域的情况。在这些情况中存储系统可能仅在访问影响了目标的状态时指示写的完成，除非不能区分写可见的效果和目标状态被更新。

对某些类型的存储器这个严格的要求确保了存储器访问的任何负面效果被处理器保证不被发生。用户能够使用这个特点来阻止以程序顺序的后继操作的启动，直到负面效果可见为止。

Write-through（WT）

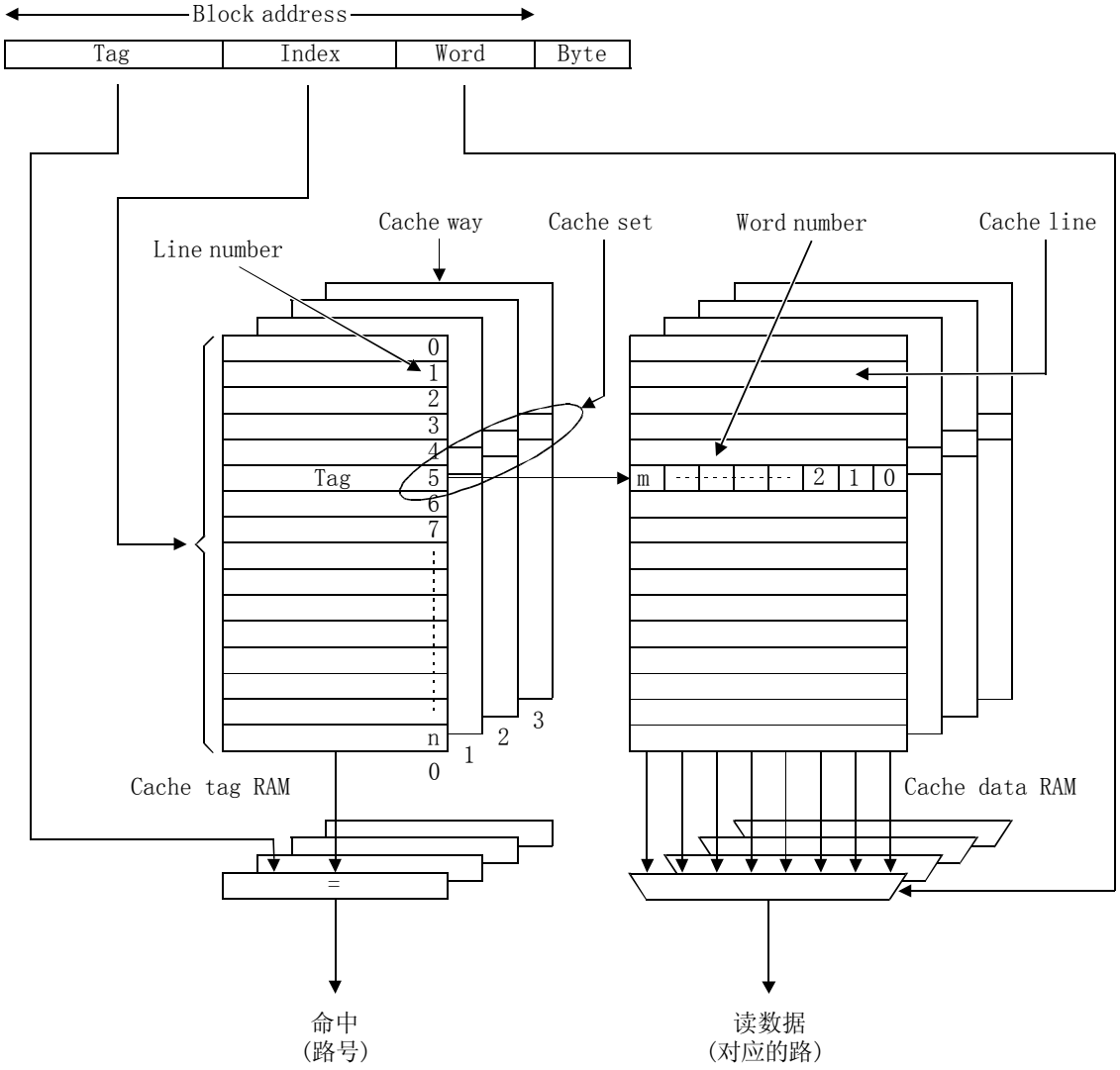
写通。在一个写通 cache 中，数据在 cache 被更新的同时被写入到主存中。

WT 见 Write-through。

Cache 术语图

下面的表示了以下的 cache 术语：

- block address；
- cache line；
- cache set；
- cache way；
- index；
- tag。



后记

这是 kongsuo 个人的第三部外文技术资料翻译作品。前两部是《AMBA™ 总线规范 V2.0》和《ISO11172-3 运动图像及其音频的编码》。它们的下载网址是：

http://download.csdn.net/source/700594和http://download.csdn.net/source/1397785

《AMBA™ 总线规范 V2.0》定义了设计高性能嵌入式微控制器时的一种片上通信标准。详细描述了 AHB、ASB、APB 总线的信号、时序、通信过程的细节。这篇文档有助于硬件工程师设计出符合 AMBA 规范的模块。

《ISO11172-3 运动图像及其音频的编码》详细描述了现下流行的 MP3 的码流格式、编解码流程以及附表中的标准数据。这篇文档有助于软件或硬件工程师编写符合 ISO11172-3 标准的 MP3 码流编解码程序或芯片。

关于本文英文原文，可在 ARM 公司的 Information Center 里下载，链接如下。ARM 公司的 Information Center 里面还包含了很多 ARM 公司产品的技术手册和开发工具的资料，有兴趣的可以去里面淘宝。

http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.dui0155i/index.html

翻译这些外文技术资料的本意有两个：一是想提高自己的英语水平；另外就是不想荒废业余的时光，想给由学生到 IT 民工转变过程中迷茫的自己找点有意义的事情做。

不过，第一个目的可耻的失败了。因为我个人的生活和工作环境中，除了看数据手册之外，几乎没有张嘴说英语的条件。所以这三篇文档的翻译都是靠着威武的金山词霸和 Google 才完成的。对个人而言唯一的收获就是多认识了几个出现频度较高的专业术语。看来要学好英语，只能另找方法了。

好在另外一个本意倒是达到了。翻译是个脑力活，也是一个体力活，对于动则几百页的外文技术资料来说。因为白天要上班，所以只有晚上回家后才开始翻译。每天付出的工作量也不大，8 点多回到住所，休闲一会后就开始翻译，翻译到 12 点左右就停工了，因为第二天还得上班，不敢开夜车开的太晚。《AMBA™ 总线规范 V2.0》的翻译耗时两个月，第一次做这种翻译，毫无经验，好几次差点放弃；《ISO11172-3 运动图像及其音频的编码》的翻译拖沓了四个月的时间才完成，因为中间有朋友来访，在我家里住了一段时间，打乱了我的计划，导致这个工程差点烂尾，还好后面顶着种种放弃的理由和游戏对我的诱惑坚持完成了。而本文档由于有前两部的经验，过程上要顺利很多，不过也花了一个半月的时间。坚持到最后的疲倦，都被作品的完成以及网上大家对我的肯定评论所带来的成就感一扫而空。

其他的收获，就是 Word 排版方面的进步了。《AMBA™ 总线规范 V2.0》和本文档都是 ARM 公司的文档。ARM 公司取得如今这般举世瞩目的成就，我想它的文档也应该功不可没。文档里面的内容描述的很细致，通俗易懂。排版很规范：关键字粗体、斜体高亮显示，交叉引用的广泛使用，极大的方便了阅读。而且文档里的所有图形都是矢量图，避免了图像缩放所带来的模糊和失真。所以本文在编写的时候，都尽量山寨 ARM 公司文档里面这些成功的元素。不过最后不得不鄙视一下 Word2007，徒有华丽的界面，但排版方面的很多操作都不如 Word2003 方便，而且还在把这篇文档转成 PDF 格式时给我带来了不少麻烦。

到如今 kongsuo 已经工作了快一年半了，渡过了才入职时的迷茫期，逐渐找到了自己应该发展的目标。希望，我自己这点微不足道的经历，能对许多曾经和我一样迷茫过的人有所帮助，在迷茫的时候，学会坚韧，学会给自己找有意义的事情做。也希望这篇文档，能帮助到那些和我一样热衷于嵌入式，喜欢研究 ARM 的人。

声明

由于译者水平有限，加上时间匆忙，本文档中难免有错误和不足之处，由此带来的任何负面效果，本人概不负责。但欢迎感兴趣的同志批评和指正。请把您的意见和建议发送到 kongsuozt@126.com，我会及时更新大家的意见和建议，争取把这篇文档做得更好。

本文档仅供从事相关行业的人员作学习和交流之用，不得用于出版、发行或者其他商业目的。