

Blackfin在线培训课程

课程单元：Blackfin®处理器性能优化

主讲人：Rick Gentile

第一章：导言

第1a节：概述

第1b节：背景信息

第2章：应用框架

第2a节：常见应用框架

第3章：性能优化

第3a节：Blackfin处理器特性概述

第3b节：高速缓存概述

第3c节：DMA概述

第3d节：高速缓存与DMA之比较

第3e节：Blackfin处理器存储资源概述

第3f节：主要存储性能基准测试

第4章：管理资源

第4a节：共享资源

第4b节：外接存储器

第4c节：DMA优先级

第4d节：中断处理

第5章：举例

第5a节：视频解码器

第6章：结束语

第6a章：总结

第一章：导言

第1a节：概述

大家好，我是模拟器件公司（ADI）的Blackfin处理器应用工程师，我的名字叫Rick Gentile。今天这个课程单元的主题是如何对Blackfin处理器进行性能优化。在开始之前，我首先要介绍一些关于本课程单元的背景信息，主要是让大家大致了解，有哪些技术可用于对Blackfin处理器家族进行系统性能优化。

参加本课程的用户最好具备Blackfin处理器架构方面及软件术语方面的基础知识，并且拥有一定的嵌入式系统开发经验。好吧，现在开始进入正题。

本单元的课程安排如下：首先，概述课程目的以及本课程对于嵌入式系统开发的重要意义。然后，简要介绍应用框架，以及我们的客户通常使用的应用框架的一些共同特征，以帮助用户在进行Blackfin处理器应用编程时轻松上手。我们还将讨论Blackfin处理器提供的存储器，以及如何实现最有效地利用其片上内存和外接存储器。接下来，我将简要介绍基准测试，以便大家初步理解如何实现最优性能，从而充分地利用系统。

之后，我将介绍如何管理共享资源，包括部分内部总线，以及与之相连的处理器各个组件。这一点非常重要，因为这有助于用户从更高的层面理解这些组件的相互作用及其工作方式。

中断管理是嵌入式处理的一个基本组成部分，我将介绍一些对于确保实现最优系统性能至关重要的最为基础的知识。最后，我将举例演示上述所有内容。

现在，正式开始。

第1b节：背景信息

现在，你大概已经参加了关于编译器优化的课程，那么，你也知道，编译器优化是进行C和C++编程优化的第一步。本课程其实就是在此基础上，进一步进行系统级优化，包括三个基本内容：存储管理、DMA管理和系统中断管理。

我们将深入讲解一些技术，通过充分利用这些技术，用户可以大幅提升系统性能。在任何视频系统中（其实非视频系统也一样），开发人员面临的一个挑战就是，要在系统中传输和处理大量数据。然而，由于可用缓冲区容量以及数据率的限制，开发人员最终需要综合利用处理器的各种存储资源，包括片上内存和外接存储器。

一般而言，大多数用户都是在PC或工作站上进行原型开发，然后再将原型导入嵌入式处理器，进行下一步开发。这往往会遇到类似的问题和挑战。这里我想说明的是，随着嵌入式处理器的功能越来越强大，将有更多的开发人员会将原本仅限于台式机处理系统的硬连线应用，搬到嵌入式系统中。因此，我们今天的课程就是为了帮助用户解决在这个过程中遇到的挑战。

对于许多应用而言，其开发过程中的各优化阶段都如下图所示。在整个开发流程中，不论是从编译器优化，还是从系统级优化的角度而言，一切都始于将在PC上完成开发的某种类型的C代码导入Blackfin处理器，然后，再对这些C代码继续进行下一步优化。

对于Blackfin处理器，其编译器是为了处理这种类型的应用而彻底从头设计的。因此，在对Blackfin处理器进行编程时，开发人员将体验到许多与众不同的经验。本

课程旨在介绍如何在Blackfin处理器上对C代码进行进一步优化，也就是说，如何通过有效利用存储资源和充分利用处理器总线，实现高效调用数据和指令。

第2章：应用框架

第2a节：常见应用框架

那么，现在就来介绍应用框架的概念。应用框架的概念非常重要，因为从我们的客户开发的许多应用来看，这些各不相同的应用分别可以归入几种不同类型的应用框架。下面，我将逐一介绍这几种应用框架，并归纳出其共同特征，以使用户在着手进行开发时能够很好地考虑这个因素。

首先，我要简单地给应用框架下个定义。从不同的角度，应用框架有不同的定义，而在本课程中，我们所说的应用框架其实就是在嵌入式系统中，调用代码和数据所使用的软件基础架构。在项目的最初阶段，设计好一个合理的应用框架将令开发人员在整个开发流程中获益匪浅。

在Blackfin处理器应用开发中，有三种最常见的应用框架类型，分别是“飞速处理”、“轻松编程比系统性能更重要”和“系统性能高于一切”。

采用“飞速处理”应用框架的情况其实有两种。一种情况是，不能长时间等待数据缓冲，例如视频应用，用户无法容忍对一个NTSC格式视频帧的缓冲时间超过33毫秒，因此，应用必须对视频信号进行逐行处理。另外一种情况就是，用户不愿使用外接存储器，只想利用“片上”内存，因此，其系统中没有可用于进行数据缓冲的资源。

因此，在这种应用框架中，整个处理流程其实就是接收数据、处理数据、做出决策、然后删除数据。举例说明，在车载道路偏离警告系统中，用户不希望等到对整个数据帧处理完毕后再做出决策。因此，应用可能只需要追踪每个帧中的一小部分数据，例如，道路一侧的分界线，应用只需要对视频信号中的这个部分进行处理。

在第二类常见的应用框架中，轻松编程最重要。有两种情况适于采用这种应用框架。一种为用户希望缩短产品上市时间，尽快推出产品。另一种情况是，用户在PC上开发原型，因此，用户需要采取特殊处理，将数据转换成某种格式，使得对该数据而言，处理器的存储架构与PC相类似。稍后我将详细解释这一点。

关键在于，使用这种应用框架的目的是简化应用开发，不论是新手还是专家都能利用这种应用框架，轻松开发需要的应用。

在这个例子中，处理器收到并处理视频信号。这可能需要执行大量的数据传输，但却能够按用户的需要，将数据转换成特定格式，从而为其提供与原型环境类似的系统环境。

第三种应用框架就是所谓的“性能至上规则”。也就是说，选择好所要使用的处理器之后，用户在编程中采用了大量的高级技术，使得应用能够充分利用处理器的处理能力。在这种框架下，应用可能要在系统中执行大量的数据调用，而与此同时，应用将在每一个周期对数据进行相应的处理。例如，视频编码解码应用。应用每接收到一行视频信号，就将其保存到片上内存中，进行行缓冲，然后再将其传输到外接存储器中。总之，应用将执行大量的数据传输和数据处理。

上面的这些介绍是为了说明，不论用户的应用适用于哪种类型的应用框架，所有这些应用框架都拥有一些共同特征。实际上，所有应用框架最大的共同点就是对系统中的指令和数据进行管理。在本课程结束时，大家将会理解，在项目的最初阶段，设计好一个合理的应用框架非常重要，这将有助于加快开发速度，减少不必要的麻烦。

现在，大家已经了解了本课程的重要意义，那么，接下来，我将介绍Blackfin处理器的一些固有特性。利用这些功能强大的特性，用户可以显著提升其系统性能。在这些特性中，有的可以直接利用，而有的则需要用户进行少量的设置和编程。下面，我将逐一介绍这些特性。

第3章：性能优化

第3a节：Blackfin处理器特性概述

这里列出了实现性能优化的几种策略，总的来讲，分为四大类。一类是利用系统中的高速缓存和DMA。我将介绍什么是高速缓存和DMA，以及如何实现最高效地综合利用这两种存储技术。第二类是存储管理。这个策略涉及一些高级技术，将会用到大家在前面的课程中学到的知识，例如LDF文件和如何将代码和数据映射到系统中的不同存储空间。第三类是DMA通道管理。DMA控制器是一个功能强大的特性，我将简要介绍如何对通道进行优先排序，以及如何最有效地利用这些通道。最后一类策略是中断管理。

那么，首先来比较一下高速缓存与DMA。这张示意图从全局的高度显示了高速缓存和DMA在整个应用编程世界中的位置。在应用编程中，开发人员有两个难以兼顾的目标，“提升性能”和“轻松编程”。例如，启用高速缓存非常容易，并且可以在一定程度上提升性能。但是，如果利用DMA，虽然需要进行少量的编程，但却可以实现大幅性能提升。待会我将举例说明这一点。

第3b节：高速缓存概述

首先介绍高速缓存，让大家了解一些关于Blackfin处理器以及一般的嵌入式系统中的高速缓存的工作方式的通用术语和基础知识。首先是指令高速缓存。顾名思义，高速缓存其实就是用于保存“东西”的存储空间。在Blackfin处理器上，这些“东西”

就是指令和数据。在一个嵌入式系统中，高速缓存的作用是提升系统性能。我们希望能够为处理器提供无限量的单周期调用内存空间，但是，如果我们或者任何其他制造商真的实现了这样的内存，那么，处理器的大小和成本将令人难以接受。为了实现片上内存和外接存储器之间的平衡，于是，我们利用高速缓存来弥补这种差距。

在这种情况下，使用高速缓存的目的实际上是将指令和数据保存到单周期调用片上内存空间中。也就是说，在理想的情况下，当应用需要调用这些指令和数据时，它们就在高速缓存里待命。这样一来，应用就可以随时利用高速缓存，执行单周期指令执行或单周期数据读取。

对于数据高速缓存和指令高速缓存，与Blackfin处理器内核之间的最高带宽通道是通过高速缓存线路占用机制实现的。另外，高速缓存还有一个很有用的特性，就是最常调用的指令和数据将实际上一直呆在高速缓存里。例如，如果应用不断地反复执行某个代码，那么，在Blackfin处理器的高速缓存架构中，这个最经常调用的代码将实际保存在高速缓存里，并且享有最高存储优先级。

关于指令高速缓存的介绍就是这些，下面，我将介绍数据高速缓存。数据可以来自外设，也可以静态生成或来自数据表。不论哪种情况，高速缓存将通过一条高性能宽带通道将数据存入单周期调用内存空间。

除了具备指令高速缓存的所有特性，数据高速缓存还具备其他一些特性，我们称之为“直写”和“回写”。“直写”是数据高速缓存的一个可配置选项，允许用户随时更新其源存储器中的数据。那么，什么是源存储器？

如果外接存储器中一个缓冲数据被映射至数据高速缓存，那么，这个外接存储器就是源存储器。假设，我从外接存储器读取了一个值，并且对其进行了上百万次修改。如果我在配置数据高速缓存时选中了“直写”选项，那么，我每一次修改这个值，其源存储器中的对应数据也会产生相应的变化。

此外，还有一个被称为“回写”的选项，可以进一步提高系统性能。在刚才的那个例子中，我从外接存储器读取了一段数据并对其进行了上百万次修改，与“直写”不同，如果我启用了“回写”选项，那么，只有当数据高速缓存中的这个数据被覆盖时，其源存储器中的对应数据才会产生相应的变化，从而实现大幅提升系统性能。

一般而言，在多种不同的应用中，最好首先采用“回写”特性。对于任何特定的算法，“回写”特性的处理效率比“直写”特性高约10-15%。当然，“直写”特性也有其自身的价值。如果应用要在多个资源之间共享数据，例如，ADSP-BF561双核处理器的两个内核之间，或者在单核处理器中，DMA控制器和处理器内核需要同时调用相同的数据，那么，就必须保持数据的一致性，这时最好选择“直写”选项。

必须指出的是，在进行C代码编程时，“回写”特性的作用可能并不明显。但是，当所有系统外设同时运行时，“回写”特性的优越性就会变得非常显著。在后面的课程中，我将介绍DMA控制器和内核在读写外接存储器时发生的相互作用，这时，你就会体验到“回写”特性的强大武力。但是，多个资源共享数据的情况则另当别论。

第3c节：DMA概述

好了，现在大家已经理解了高速缓存，接下来，我们要讨论DMA，以及如何综合利用这两种存储技术。当然，我不是要给大家一个统一的解决方案，我只能提供一些基本原则，以使用户根据自己的应用要求，做出最佳抉择。

DMA控制器是Blackfin处理器架构中的重要组件，它完全独立于内核。它不会进行周期挪用，完全无需占用处理器内核周期。在理想的应用配置中，内核只需要设置DMA控制器，并在数据调用过程中响应中断。这也是用户在应用中使用DMA控制器的终极目的。

通常，所有的高速外设以及大多数普通外设都具备DMA处理能力。也就是说，处理器可以利用DMA，直接从外设调用数据。此外，Blackfin处理器拥有一组适用于存储器之间的DMA控制器。利用这种存储器间DMA控制器，应用可以将数据从外接存储器映射至片上内存、从片上内存映射至外接存储器或者在某个存储空间内部进行映射。提供这种DMA控制器的目的其实就是在数据调用系统中，充分利用DMA控制器，因为DMA控制器每传输一段数据，对内核而言，其任务就减轻一分。

下面，我将简单演示Blackfin处理器的2-D DMA特性，这个特性对于提高系统性能非常重要。在本例中，左侧所示为红、绿、兰三色代表的三个缓冲区。在略靠中间位置，是采用1D DMA的应用中，由依次排列的红、绿、兰三个缓冲区构成的混合缓冲区。而在右侧，是实现了2D DMA的应用中，彼此独立的红、绿、兰三重缓冲区。当然，取决于数据处理的移动方向，这些缓冲区的排列位置也可以完全相反。在本幻灯片的左下角，列出了一个伪C代码。如果处理器正在运行这个代码，那么，它就不能同时执行其他任务。而如果设置了DMA控制器，由其来传输数据并在结束时生成一个中断，那么，核心就可以自由地执行其他的任务。通过这个简单的例子，希望大家能够了解2-D DMA特性的优越性。

DMA和高速缓存的另一个备受用户青睐的重要特性是，可以借助数据高速缓存来调用通过DMA输入的外设数据。也就是说，将高速外设在外接存储器中生成的缓冲数据映射到数据高速缓存中；通常采用乒乓缓冲，在填充一个缓冲区的同时，处理器对另一个缓冲区进行处理。我们发现，许多应用都同时利用了DMA控制器和高速缓存。

这里，值得注意的是，必须使高速缓存中的数据 and DMA控制器传输的数据保持步调一致。在本幻灯片的左下角，显示了这个数据流程。首先是外设生成一个新的缓

冲数据，并由此产生一个中断；中断通知内核，高速缓存中有了新的数据，等待处理；于是，内核开始处理这个缓冲数据，并在处理完毕后，将与该缓冲数据相关的高速缓存线路设置为无效，以确保新旧两个缓冲数据之间的一致性。因此，仅就内核而言，相比于只调用数据，这种处理方式提高了系统性能。然而，一个完善的DMA配置的优越性不只这些，也就是说，应用还可以利用DMA控制器，将缓冲数据直接映射到片上内存中。接下来，我将详细解释这一点。

第3d节：高速缓存与DMA之比较

先来说说指令分割。这个相当简单的流程图演示了应用将如何实现最优性能。首先，最简单的情况就是指令刚好可以映射到片上内存中，于是，只要将这个代码映射到片上内存中就大功告成，如你所愿实现了最优性能。因为所有指令都通过单周期调用执行完毕。然而，在采用Blackfin处理器的大多数应用中，往往要外接存储器中运行网络堆栈或大量代码。因此，在这种情况下，最好的办法就是启用指令高速缓存。这里，你会发现神奇的“2/8定律”，即，处理器有80%的时间是在运行20%的指令。从高速缓存的角度来讲，这是个好消息，因为高速缓存的容量通常比它所支持的源存储器小得多，所以说最常调用的代码只是一小部分对高速缓存而言是个好消息。因此，大部分时间，处理器都是在高速缓存上运行代码，可以说大多数应用都属于这种类型。

之前，我曾介绍了三种应用框架，其中一种是性能至上型。在这种应用框架下，会有许多代码无法保存到片上内存中。如果用户不想使用高速缓存，则可以采用所谓的“覆盖机制”，将代码保存到某个存储体中，同时在另一个存储体中执行代码，并以这种方式进行管理。下面的这条线表示，在这个图中，越往下走，需要进行的编程也越复杂。因此，Blackfin处理器架构的一个好处就是，用户可以进行选择，通过这一系列步骤满足所有这些要求。

数据DMA和高速缓存的情况也和指令高速缓存差不多。如果可以将缓冲数据映射到L1内存中，大多数应用都可以做到这一点；那么，就直接将数据映射到片上内存中。如果在应用编程中实现了DMA，那么，对外设而言，通过DMA输入数据最轻松。但问题是，如果外接存储器中的缓冲数据比较大，那么，该通过什么方法，将其映射到片上内存中并进行处理？在这种情况下，通常需要综合利用外设DMA和存储器间DMA配置，来按应用的需要传输数据，或者不仅要综合利用2D DMA，还要启用数据高速缓存，并在下一次调用该缓冲区之前，将其高速缓存线路设置为无效。

综上所述，在应用中，利用DMA和高速缓存的方法多种多样，通常是通过某种形式的综合利用，实现最优性能。不过在大多数应用中，最高效、最常见配置一般都启用了指令高速缓存。静态的表和数据将通过数据高速缓存进行处理。而动态的数据则最好综合利用外设DMA将其传输至外接存储器，再通过存储器间DMA，传输至片上内存。

第3e节：Blackfin处理器存储资源概述

现在，大家应该已经从宏观上理解了DMA与高速缓存在应用中发挥的作用，同样地，从宏观的高度理解存储架构也很重要。

当然，不是要求大家成为存储架构专家，而是希望你们能掌握一些可以在编程中用于大幅提升系统性能的存储架构知识。因此，下面我将简要介绍一些关于存储架构的基础知识。

Blackfin处理器内核以所谓的“核心频率”运行，通常是600 MHz。ADI提供了范围广泛的核心频率，不过，我在本例中选择采用600 MHz频率。此外，我们还提供了各种级别的存储器，以便用户平衡选择片上内存和片外存储器的最佳组合。一级存储器是指以核心频率运行的存储器，在本例中，即600 MHz。通过一级存储器，可以实现单周期调用指令和数据。通常，用户可以将一级存储器配置为SRAM或高速缓存。常用的一级存储器规格为几十Kb，最高不超过一百Kb。

离内核稍远一点，但仍然在芯片上的是所谓的“片上L2内存”，调用其中的指令和数据需要耗费数个周期。但是，这种内存的容量可以达到上百Kb，一般是128 Kb、256 Kb，或诸如此类的规格。然后，是片外存储器，其调用需要耗用数个系统周期（系统时钟频率通常为133 MHz），不过其容量也高得多，往往达到数百Mb。

在介绍存储体架构之前，在这里我想指出的是，在Blackfin处理器上，在一个内核时钟周期内（按600 MHz时钟频率计算），处理器可以执行一次指令读取和最多两次32位数据读取，或者一次32位数据读取和一次32位数据保存。所有这些读取操作只需要一个时钟周期。另外，前面我曾提过，DMA控制器也可以对类似存储体或同类存储体架构进行存取操作，而无需挪用任何内核周期。在许多情况下，调用子存储体不会产生任何存储体冲突，待会我会详细解释这一点。

接下来，我将介绍片上内存和外接存储器。先从片上内存开始，在Blackfin处理器上，所有片上内存存储体都由许多子存储体构成，这是为了支持内核在一个时钟周期内，同时调用多段数据，以及支持DMA控制器调用。

这张图概括地演示了如何有效利用片上内存的各个子存储体，存储体中的实线表示子存储体的界线。在本例中，所有的缓冲数据和系数都被映射到片上内存的两个子存储体中。这就使得当内核和DMA同时调用同一个子存储体时，会发生冲突，这显然是应当尽力避免的。

在图的右侧，我将这些缓冲数据分别映射到不同的子存储体中，这样，DMA和内核就能和谐地进行各自的存取操作。一会儿，我将详细解释这种映射和存取方式，不过，我首先要介绍片上内存的存储体架构。一般而言，Blackfin处理器拥有多个16 Kb可配置存储体。每个16 Kb存储体都由4个4 Kb子存储体构成。如我先前所说，这些子存储体可以配置为SRAM或高速缓存。

如果将这些子存储体配置为SRAM，那么，这个存储体将支持单周期调用，指令读取单元可以存取其中的数据，DMA控制器也可以在必要情况下向其中填充指令。此外，还可以将这些子存储体配置为高速缓存，作为四路联合高速缓存。前面我们已经介绍了高速缓存的作用。

下面来看看数据存储体。和指令存储体一样，每个16 Kb的数据存储体都由4个4 Kb子存储体构成。这种存储体架构旨在支持内核同时读取两个不同的子存储体，并支持DMA控制器调用第三个子存储体。另外需要指出的是，当内核同时对同一个子存储体执行两个读取操作时，如果数据的地址位并不相同，也就是说，一奇一偶，那么，也可以避免存储体冲突。理解存储体架构的工作原理非常重要，例如，如果我要处理4个数Kb的数据，那么，我可以让DMA控制器填充其中一个子存储体，同时让处理器内核读取其他子存储体中已经填充的数据。这样，存储体的运行方式就类似于某种交换机。

那么，关于片上内存的存储体我们就介绍到这里。接下来，我们要来讨论外接存储器的存储体。说到外接存储器，我们需要理解所谓的“SDRAM物理学”。基本上，任何时候调用SDRAM存储器中尚未激活的存储页都需要耗用好几个系统时钟周期。因此，在有些存储架构中，每一次调用超过1 Kb或2 Kb的一段数据或一个指令，都会遭遇时延。

Blackfin处理器的外部总线接口单元最多可以跟踪4个SDRAM内部存储体中的4个开放行。因此，连接至Blackfin处理器的任何SDRAM存储器，通常都会被划分为4个内部存储体。在本例中，连接至Blackfin处理器外部总线接口单元的SDRAM存储器包含4个16 Mb的存储体。虽然原理不同，但与片上内存的子存储体一样，合理利用SDRAM的内部存储体可以显著提升系统性能。

尤其是在外接存储器中，处理器通常会将所有的缓冲数据一股脑地全部塞进存储体。例如，应用可能会连续生成一个代码、一个视频缓冲帧、又一个视频缓冲帧，如果没有经过精心设计，处理器可能会默认将所有这些数据一个接一个保存到SDRAM存储器中，致使这些数据和代码全都挤在一个内部存储体中。

利用我们提供的LDF文件，用户可以轻松配置存放这些数据和代码的存储体。当然，这要取决于系统采用的SDRAM存储器的大小，因此用户必须根据其SDRAM存储器的规格，实现充分利用这几个内部存储体。在本例中，这种方法很管用。我将代码保存到一个存储体中，而将缓冲帧单独放到其他存储体中，这只需要进行简单的编程，但却可以大幅提高性能。

第3f节：主要存储性能基准测试

下面，我将介绍几种基准测试，我把这些基准测试归纳成一张表，以便大家今后参考。此外，所有硬件参考手册中关于芯片总线分级结构的部分都提供了一个表格，

其中包含了接入各种不同总线的所有基准测试。提供这张表是为了概括出最重要的内容，并再次强调，尽一切可能充分利用DMA控制器非常重要。

对SDRAM处理器而言，这一点也很重要。因为对外接存储器执行一次16位数据调用，Blackfin处理器需要耗用8个系统时钟周期。而执行一次32位数据调用，则需要耗用9个系统时钟周期。但如果利用DMA控制器，假设存储行已经激活，并且当前存储体处于开放状态，没有存储冲突，那么，读写数据都只需要一个系统时钟周期。所以说，关键在于借助DMA控制器，可以最高效地调用数据和指令。

第4章：管理资源

第4a节：共享资源

前面，我们介绍了高速缓存、DMA，我们还讨论了存储架构以及用户在构建系统时应当注意的关键点。接下来，我们将重点讨论管理共享资源，就我们今天举的例子而言，最重要的是管理外部总线。然后，我们还要讨论中断管理。

首先介绍共享资源。我想重点介绍外接存储器接口。在本例中，我将使用BF561处理器，这是一个双核处理器，它囊括了所有Blackfin处理器的全部组件。还有一点非常重要，在默认状态下，内核、外部总线和DMA控制器的优先级依次降低。下面，提醒大家注意几个要点。所谓内核调用是指由内核读取指令或读取数据，这也可能包括高速缓存线路填充，即内核实际是在读取高速缓存线路。

另外一个要点是，在默认状态下，内核的优先级高于DMA控制器，但DMA通道处于紧急状态的情况除外。稍后我将通过几张图表说明什么是紧急状态。关键在于，用户其实可以改变内核与DMA控制器之间的优先级，只需要将DMA通道设置为看起来像是处于紧急状态，那么，DMA控制器的优先级将高于内核。

在本幻灯片的底部，这里显示的是多个内核和DMA控制器。有一类处理器就属于这种情况。例如，BF561双核处理器就有两个内核。这里，我要特别指出的是，其实我们是借助一种EBIU寄存器，即，异步存储全局控制寄存器，来改变内核与DMA控制器之间的优先级的。也就是说，用户可以根据应用的需要，利用异步存储全局控制寄存器，改变内核与DMA控制器之间的优先级。

刚才我曾提到“紧急状态”，这是一个非常重要的概念。不过这也是个很容易理解的简单概念，在两种情况下，DMA通道可能处于紧急状态。一种情况是，外设正在发送数据，另一种情况是，外设正在接收数据。在外设发送数据的情况下，如果DMA FIFO为空，而外设又正在请求数据，那么，DMA通道就处于紧急状态。这就表示，有某种情况阻碍了DMA控制器向DMA FIFO填充数据，而最有可能的就是内核调用。这时，处理器要做的就是防止这种情况发展成外设运行不足，因此，虽然外设FIFO中还有数据，但是处理器却希望外设FIFO始终充满数据，以免发生运行不足。因此，处理器将生成一个“紧急状态”，提升DMA控制器的优先级。

同样地，在外设接收数据的情况下，处理器就要防止出现运行过度。在这种情况下，DMA FIFO不是为空，而是满载，当外设试图向一个满载的DMA FIFO填充数据时，就会出现紧急状态。在我们的系统中，紧急状态的作用其实就是防止外设实际发生下溢或溢出。究其根本，DMA FIFO之所以会为空或满载就是因为部分共享资源被处理器的其他组件占用。

这张幻灯片概括了在调用争夺战中的“赢家”，以帮助大家理解。以降序依次是，锁定内核调用，这种调用类似于测试设备发出的指令，这是一个小指令，一旦发出就要彻底执行。接下来依次是紧急状态DMA、高速缓存线路填充、内核调用和DMA调用。因此，如果系统中没有紧急状态DMA，并且用户并未进行相关设置，那么，内核的优先级总是高于DMA控制器。

此外，我们发现有许多应用都通过一个紧凑的轮询循环调用外接存储器，或者从外接存储器执行某种紧凑的指令循环。在这种情况下，就会需要进行仲裁，只有理解了这一点，才能充分了解系统的运行情况。对于L1内存调用，优先级为DMA调用高于内核调用L1内存。

第4b节：外接存储器

好了，接下来要讨论用于性能优化的另一项技术。之前，我们已经讨论了存储架构物理学：片上内存及其子存储体、外接存储器及其内部的多个存储体。在SDRAM物理学中，常常被忽视的一点是，在对SDRAM存储器相继执行读写操作时，会产生固有时延。

简单来讲，就是在同一个方向传输的调用效率优于不同方向的混合调用。理解这一点非常重要，因为我们可以按传输方向尽可能地将调用分组，从而减少外接存储器调用中固有的方向转换时延。为了支持用户自行设置这个特性，我们在Blackfin处理器上提供了一个“流量控制寄存器”，允许用户在有多个DMA同时运行时，有效提升系统性能。在一个系统中，经常会出现多个DMA同时运行的情况。

在这个函数中，用户可以设定不同的值，提高某个方向的流量，从而将在该方向传输的调用归为一组。这是一个方便实用的特性，用户编程的这个寄存器中包含了多个流量控制字段，包括“DMA调用总线”、“DMA内核总线”，以及对系统性能影响最大的“DMA外部总线”（简称DEB）。这里突出显示了这个DEB，因为用户首先要设置的就是这个值。利用DEB，SDRAM存储器可以实现最优性价比。

但愿可以通过某种灵验的公式，计算出百试不爽的最佳值，但实际上，对于不同的应用，所谓的最佳值也不尽相同。不过我们发现，如果系统中同时执行的调用不多，那么，一般而言，DEB值越高，系统性能也越高。目前，允许的最高值为15，那么，将DEB设置为15就可以了。然而，在大多数情况下，系统同时执行的调用总是超过3个，因此，最好将DEB设置为中值，也就是4至7之间。

第4c节：DMA优先级

接下来，我将介绍DMA控制通道的优先级。在这方面，用户只要有一些粗浅的认识，就能设计出更加高效的系统。前面我已经介绍了DMA控制器，DMA控制器与高速缓存在系统中的地位，以及如何综合利用DMA控制器和高速缓存，在系统中调用数据。现在，我将简要介绍DMA控制器，然后重点讨论DMA控制器内部通道的优先级。

首先必须知道，每个DMA控制器都拥有多个通道，当有多个DMA通道试图同时调用控制器或者任何特定总线时，具备最高优先级的通道将胜出。DMA通道的优先级是可以通过编程来设置的，因此，用户可以根据应用的需要，设置DMA通道的优先顺序。Blackfin处理器提供了默认配置，不过用户可以针对自己的应用，通过编程自定义DMA通道的优先顺序。

还要指出的是，如果系统中有多个DMA控制器，例如BF561处理器，那么，用户也可以通过编程，设置DMA控制器的仲裁优先级。充分利用DMA控制器对于提升系统性能非常重要，因此，我们在系统服务程序中提供了一个DMA管理器。DMA管理器其实是一套工具和API的组合，借助它，用户可以设置DMA控制器的所有函数。DMA管理器可以与范围广泛的操作系统和内核，以及基础数据流应用互通。

第4d节：中断处理

接下来，我们将讨论中断处理。不用我说你们也知道，在嵌入式系统中，中断处理极其重要。然而，从中断的角度而言，许多应用都会出现一个共同的错误，即，在执行一项任务时用太多的时间来执行中断服务程序（ISR），从而妨碍了处理器执行其他关键型代码。

从Blackfin处理器架构的角度来看，中断在执行完毕后就会自动禁用。对于每一个事件，我们都提供了一个返回寄存器，对它而言，一个中断也是一个事件。当应用将返回地址保存到堆栈中时，就会启用更高优先级的中断。也就是说，在执行中断服务程序时，如果将返回地址保存到堆栈中，就会启用更高优先级的中断。

应用花了多少时间来执行ISR非常关键。用户面临的问题是，如何嵌套中断，也就是说，用更高优先级的中断中断另一个中断。

与DMA控制器相似，最重要的中断通常具备最高优先级。我们提供了默认配置，不过，用户也可以根据应用的需要，通过编程自定义中断的优先级。用户可以通过嵌套来确保更高优先级的中断不会被优先级较低的事件封锁。类似于DMA控制器，我们强烈建议用户采用系统服务程序中提供的回调控制器，快速执行中断，并返回最低优先级的中断。这样就能确保更高优先级的中断永远不会被封锁。不过，

用户完全可以灵活控制系统中断的优先级，可以借助系统服务程序，配置中断优先级，从而实现最优系统性能。

第5章：举例

第5a节：视频解码器

到目前为止，我们已经介绍了许多不同的优化技术，下面，我将举例演示如何在实际应用中利用这些优化技术。在这个小节，我们其实要讨论如何在项目初期设置应用框架，一些关于存储器、高速缓存、DMA的基础知识，以及如何选择和使用这些组件。然后，我们将讨论如何以尽可能最简单的方法，有效利用处理器的各种资源。

在本例中，我将演示一个视频解码器。这个处理过程其实可以代表许多系统中的复杂数据流，处理器要调用大量数据，处理每一个像素都要执行许多任务。许多其他应用的处理过程都与此大同小异。在这个演示中，有两个要点。一是，从处理的角度，在开始构建系统之前，用户必须计算好可以向每个处理元素分配多少处理周期？

在视频应用中，处理的元素就是图形的像素。本例采用的是D1视频帧——帧速率为30帧每秒的720 X 480像素。我首先计算出每秒要处理的像素数量，然后我就可以知道，应用需要多少预算处理能力，应该怎样分配时间。

本例中，处理每一个像素需要50个周期。此外，还要确保留出一些处理能力，以便进行其他运算，例如，在本例中还要进行音频处理、系统层和运输层处理、一个小的内核或者操作系统等，在开始时要盘算好一切耗费。

与处理能力预算同样重要的是带宽预算。从宏观上看，整个视频解码过程就是，首先，接收从某个外部接口输入的编码位流，以及在外接存储器中创建好的参考帧，然后通过外接存储器和片上内存之间的互动，在系统中对这些数据进行一系列数据处理，最后生成输出至显示器的视频帧。

总而言之，从宏观上讲，用户首先要计算出所有这些处理需要的带宽，以确保系统带宽能够满足应用需求。在本例中，带宽要求是130 Mb/秒。当然，要考虑的不只这些。我们前面讨论的所有处理都会产生某种程度的互动，因此，还需要进行全盘筹划。稍后我将详细阐释这一点。

这张示意图简略地演示了视频解码过程，输入视频流进入系统，这是一系列编码数据包，应用将对其进行解码。另外，通常高端视频的解码速度都比较慢。接着，创建参考帧，并利用参考帧来创建显示帧，再将其输出至LCD或连接至显示器的视频解码器。

由于本例中的缓冲数据包速率为每秒几个Mb，所以我们可以轻松利用这个应用的外接存储器。显然，应当将比较大的缓冲数据，在本例中，也就是参考帧、显示帧以及缓冲数据包等，放到SDRAM外接存储器中。

当然，查找表中的任何静态数据或者任何最终缓冲数据都应当保存在L1内存中，因为需要对这些数据进行单周期调用。这个示意图显示了这个过程，图的左侧是这些比较大的YUV缓冲数据，在本例中，这些是视频的分量数据，我要利用2D DMA特性，将宏块保存在L1内存中，以便能够随时进行单周期调用。这样，在最后就生成了显示缓冲帧，通过视频端口，将其输出至LCD或者视频编码器。

最后，这里列出了各个传输数据流的数据率。在系统中有大量数据在不同的方向上传输，第一步是计算出总的带宽需求，并且从宏观上规划需要执行的处理。但这只是第一步，用户还要考虑总线之间的互动，以及外接存储器的运行方式及其物理特性，总线转换、同步活动等等。

第6章：结束语

第6a章：总结

好了，希望大家已经了解了可以利用哪些技术，提高基于Blackfin处理器的应用的系统性能。你们将会发现，Blackfin处理器提供了许多简便易用的特性。希望大家觉得今天的课程中介绍的几个要点深入浅出，有助于实践应用。例如，在系统中调用数据、使用DMA控制器和管理中断等。我们提供了一套工具和系统服务程序，帮助用户按需使用这些特性。

希望这个课程单元能对大家有所助益。

另外，我们还提供了一些帮助信息。我们为每一个型号的Blackfin处理器都提供了相应的硬件参考手册，并且提供了编程参考。还有一篇标题为‘*Embedded Media Processing*’（《嵌入式媒体处理》）的文章（D. Katz和R. Gentile合著），其中详细阐释了本课程单元讨论的一些概念。

如果还有其他疑问，请单击下面的“ask a question（我有疑问）”按钮。谢谢观看！