

## 使用OCP？在多核设计中OCP扮演的角色

作者：Clive "Max" Maxfield

在时间的长河里，当我很久以前开始我的职业生涯的时候（“当恐龙统治地球的时候”），事情比现在简单得多了。举一个简单的例子，基本上微处理器可以被任何一个使用它们做试验的人所理解。啊，和8位数据总线，16位地址总线以及一组多用途的寄存器打交道是一件多么愉悦的事情！

比较起来，今天的高端片上系统（SoC）设计需要耗费大量的脑力，非言语所能形容。这些庞然大物包含了千万级甚至几个亿的逻辑门，它们的主要特征包括执行不同任务的多处理器核，拥有多级存储缓冲的层级存储器结构，以及多层总线结构（别再让我谈及超流水线和超标量了）。

问题是这些大量的零碎的模块中的绝大多数——像处理器核和其它功能模块，是以第三方IP模块的形式获得的。构建一个SoC的过程包括从广大的多途径来源中获得处理器、设备和存储器核。此外，还有一个时间的因素，因为这些核可能是在不同的时间被设计的。

一个核的功能规定了其与外界沟通的界面（以芯片的其它部分的形式）。我们不能期望于在不同的时间、不同的地点、不同的目的下的核设计者能够做出相同的决定。因此，一个现代SoC的设计者处于这样一种境地，他们使用不是他们设计的功能单元构建芯片，并且他们也不可能有相同程度的理解。（如果设计者确实需要完全理解内部的相互作用以

及每一个核，他们也就有从头到尾地设计它们的可能，这样将导致一次一个新的芯片的面世需要百年左右的时间 )

很显然，我们需要以某种途径去解决这些多核问题。但是被我的胆量吓着了 ( “没有什么可怕的东西，只是吓自己”，就像我亲爱的年老的父亲所经常说的 ) ，因为已经有一种以OCP的形式实现的解决方案。为了搞明白这个问题，我们需要问 ( 并且回答 ) 三个基本问题：

- 什么是多核设计？
- 什么是插槽 ( socket ) ？
- 什么是OCP？

好，首选哪一个？什么，你们需要我解释所有的？这有点过分夸赞我了。但是是什么使你们认为我知道所有的 ( 而不是像以前一样我止步于缺乏足够的知识 ) ？那好吧，如果你们坚持...

## 什么是多核设计？

我一向喜欢先确认事情在头脑中已经井井有条了，所以在这件事情上让我们一步步来，从多重处理器的概念开始。事实上，我们已经在“计算机空间” ( “在那里没有人能听到你的尖叫” ) 使用多处理器系统有一段时间了。我们以大约1995年的一台简单个人电脑为例，作为母版上的主微处理器的补充，在硬盘驱动器上可能会有一个小一点的处理器，甚至在键盘上也会有一个微小的处理器。

事实上，即使是今天最小的个人电脑也趋向于有多个处理器，像抛散的糖果一样分散在四周。然而，这种类型的多处理器执行相对简单，因为所有的处理器都是独立的。主要的、多用途的处理器处理大部分的工作，而小一些、单一用途的处理器处理有限数量的特定任务。此外，每个处理器在完全控制其自己的资源，如当地存储子系统期间，都是“其特定领域的主机”。

另外一种多处理器包括许多同样的处理器共享相同的资源；例如，一群处理器共享相同的存储器空间。作为例子，可以参照1970年代的"Big Iron"机器，例如IBM大型机和Crays。

所有的这些把我们引向了“多核”的概念，其主要指在同一块硅片上的多个处理器、设备和存储器核。其中的一个例子来自Intel和AMD的多核处理器，它们包括了两个、四个或者更多同样的（类似的）处理器核共享相同的资源。

但是现货供应的，不用定制的微处理器并不是我们这里感兴趣的。作为这篇文章的目的，我们思考高端SoC的设计。既然这样，使用相同核的多份拷贝实现相似的操作拥有更多用途并且更加复杂，在这种事实背景下，由多种用于完成特定任务的特定用途的不同种类的核组成的实现方式则更有效率。

并且，事实上，我们能将同类的和不同类的核混合在相同的芯片上。例如，ARM Cortex-A9可以由两个、三个或者四个同类的核获得（设计者也可以选择放弃在芯片上倍增这样的群，如果他们愿意）。然后这些核能够使用其他提供商提供的专门的核来进行扩张。

最后的结果就像是在一片芯片上拥有一个超级计算机，难以置信的复杂的处理器的结合分散在四周。挑战是如何处理所有这些的不同种类的特性...

## 什么是插槽？

每当被问到这个问题，我都不由自主地想起“Dr. Seuss解释为什么计算机有时会死机”这篇讽刺诗歌,其文如下：

*如果一个数据包准确击中插槽上的一个端口，  
且总线被中断作为最后的一个手段，  
且存储器的地址使得你的软盘意外中止，  
于是插槽数据包端口将会报告一个错误..... ( 等等 )*

但是我们离题了...

术语“插槽”最初来源于网络系统中上下端互联的构建。插槽定义了系统中的两个完全不同的器件边界处的功能。更具体的说，插槽定义了边界处的电气和逻辑功能特性，并且从协议的层次上看，它定义了与外界交流的通信行为。

现在，考虑这样的情形，即某人开始用完全不同的核来搭建一个多核系统。分析一个相对简单的情况，如图1所示，我们只用四个核来构造这个系统，它们是通过系统总线互联的。

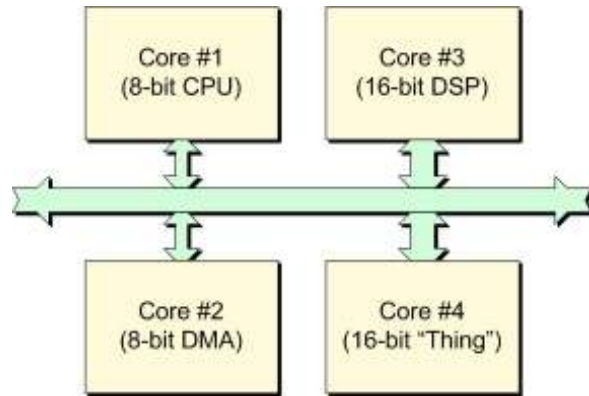


图1 一个非常简单的多核系统的例子

为了讨论的方便，我们假定核1是原系统已有的处理器，它含有8位的数据总线和一些控制信号；同样的，我们假定核2是原系统已有的DMA引擎，其数据总线为8位宽；而核3是新添加入系统的16位的DSP引擎。

现在，如果我们希望核1（处理器）使用核2（DMA），糟糕的是你会发现它们的控制信号并不匹配。另一种情况，如果我们我们希望系统使用DRAM，你会发现我们的8位处理器和DMA核在设计的时候并没有考虑到去支持DRAM的猝发传输模式；这种棘手的情况，你还可以举出许许多多例子来。

除了这些很明显的复杂性之外，这样的事实又使得事情变得更加复杂，即像我们先前讨论的那样，一个现代SoC的设计者只是芯片架构的设计者，他们使用一些并不是他们设计的功能单元来构造芯片，所以他们也不可能理解这些功能单元的所有细节。

解决方法是给各种核使用插槽，如图2所示，插槽包装了核的内部控制和数据信号，使其成为一个通用的区域。

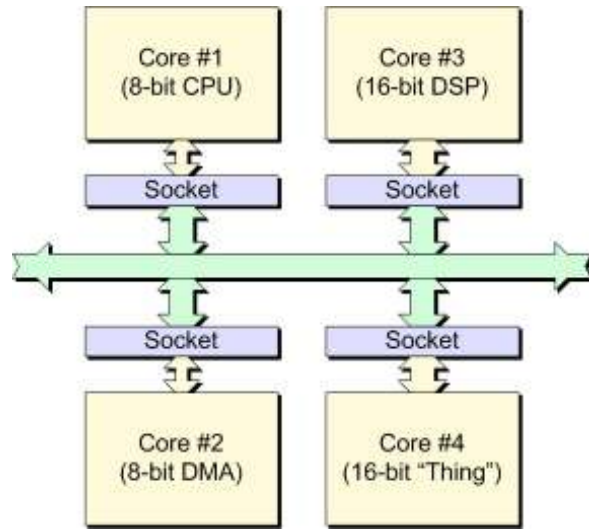


图2 使用插槽将核集成到系统之中

像许多事情一样，似乎这是显而易见的，只需快速的说出然后摆摆手就可以了，但实际情况却并非如此。对于不同的系统，如果我们一不小心，将会不得不为各种核构造不同的插槽，这样我们又重回老路了，使用插槽并没有使得系统更易于设计。

## 什么是OCP？

在2001年12月，开放式内核协议国际合作协会 ( the Open Core Protocol International Partnership Association ， OCP-IP ) 在喇叭嘹亮的吹奏声中跃出了舞台。OCP是第一个半导体知识产权 ( intellectual property ， IP ) 核领域的得到完全支持、使用许可公开、广泛兼容的界面插槽。同时，OCP-IP是为提供OCP的支持、改进和增强而专门成立的一个独立的、非盈利的半导体工业联盟。这个联盟已经发展到拥有超过200家会员公司，从一般意义上讲可以为SoC设计者营造一个更好的环境。例如：我听说大约有5亿的“小发明”其内部使用了OCP。

就像SoC的设计者都会意识到的那样，OCP-IP的创立者们充分意识到集成不同来源和不同时代的多核所遇到的问题。知道了这个，从最开始就牢记OCP开发并且维持是为多核设计的。OCP的设计者也明白试图强制人们在设计核的过程中完全精确地使用他们的界面是不现实的。很容易地，他们认识到解决方案应该以可配置的插槽的形式建立。

说了这么多，然而，重要的是要明白OCP本身并不是一个可配置的插槽；取而代之，它是一种可配置的插槽规范，为核内外的片上数据、测试以及控制流的各个方面提供了一个统一的架构。

事实上，OCP规范实际上是不同核的设计者希望他们所设计的核的功能相互之间进行通信的可能途径的一个超集。假设你已经设计了一个核，并且你希望设计一个OCP插槽。你看着你的核的控制、数据和测试信号，以及它必须具有的通信特性，例如必需的簇大小和绝对地址模式等，然后你仔细研读OCP规范，通过选择得到你用来设计相关的插槽的构造。

让我们假设你想让你的核与“Max的最高机密核”通信。我们可能已经分别设计好了我们的内部界面，但是有一点点差别，如果我还提供给你一个和你的插槽处理相同事务的OCP插槽，然后你确实就不需要知道（或者关心）我在干嘛或者在我的核的内部我是怎么操作的。

所以现在我们已经到了这样一个阶段，我们已经解决了语意上的挑战并且我们正在使用相同的词汇进行交流。使用OCP，知识产权核（IP）的设计者能够使得他们的核与特定的总线协议，进而与任何的特定的设计操作相独立。这使得在多重SoC设计中，重用兼

容OCP协议的核更容易，当然更不用说可以使你的当前SoC设计更便利了。同样，极致的OCP配置允许你实现已经设计好的核使用其它界面，并且使它们在不丢失任何性能的前提下，和其它实现方式相比能够更快地做到和OCP兼容。

我们也应当意识到OCP不仅仅包含了一个协议，同时还包含了控制协议的参数，这些参数可以在元数据文件中获得，它们描述了一个特定的插槽所选择的参数设置。这样，就可以使用自动化工具通过访问相互通信的不同的插槽的元数据文件来确保它们在使用相同的语言通信（并且同时也是相同的语调）。

最后，但也是很重要的，因为OCP规范包含了如此多的内容，向新的OCP使用者演示怎么开始就显得重要了。因此，OCP提供的内容包括指导方针和一系列的框架，这些框架是能够完成通常的任务的OCP架构；例如，有一个框架就是给DMA核的一些建议。

## OCP TNG

今天的SoC是功能点的集中，这些功能点曾经都是在不同的设备中实现的。这些功能覆盖了从简单的继承核到复杂的现代处理器。对于使用简单通信协议的基本核、用于高级通信需要的高端处理器以及复杂度介于它们之间的所有核，OCP都是很有效率的；在工业上，OCP在这个范围的独特性是有目共睹的。

今后是什么样的呢？好，当前的OCP版本是2.0，但是OCP 3.0——我更喜欢把它看作是OCP TNG（“The Next Generation”，下一代）——包括了各种各样的大量的改进。例如，高速缓冲存储器的一致性就是一个紧迫的主题。不

要吃惊，由特定目的的不同种类核组成的系统软件的设计要比由拥有多种用途的相似的组件组成的系统更困难。

多核处理器像来自ARM和MIPS的，当然，其内部高速缓冲存储器是一致的，但是如果你愿意，我们正在讨论的是在这些内部簇群之外的一致性——“超级高速缓冲存储器一致性”。为了解决这个事情，IP核的设计者们正在为系统范围的高速缓冲存储器一致性提供硬件支持。在这种情况下，OCP 3.0已经通过扩充，包括了高速缓冲存储器一致性信号。

功耗是另外一个热点话题（故意使用了双关语）。在不工作的时候降低其功耗，这样一种功能在如今已经是一种很常见的实践。为了解决这个事情，OCP 3.0已经着重补充了通过添加插槽信号来实现允许界面的一边被关掉的能力。这包括了额外的握手协议，它用来更容易地实现界面的一边和系统相断开，以及在安全意义上打开/关闭时钟和/或电源。

我告诉你，我能够在这个事情上唠叨几个小时，但是你应当去浏览OCP-IP的网站([www.ocpip.org](http://www.ocpip.org))，在上面听取专家的讲述并且了解他们所做的不同的事情（比如，我就从来没有提及过验证IP——Verification IP，VIP），这将胜于听我的讲述。希望你们一切都好！

**关于作者**

Clive (Max) Maxfield是TechBites

(www.TechBites.com)的总裁，一位专攻高技术的市场顾问。Max还是EE

Times子站，Programmable Logic DesignLine (www.PLDesignLine.com)的编辑。

在过去的岁月里，Max曾经设计了从硅片到印刷电路版之间的所有东西。Max的文章出现在世界范围内的技术杂志上，同时他还是包括《Bebop to the Boolean Boogie (An Unconventional Guide to Electronics)》、《The Design Warrior's Guide to FPGAs (Devices, Tools, and Flows)》以及《How Computers Do Math》等在内的一系列书的作者和合著者。