**Layerscape架构：深入理解下一代QorIQ LS系列SoC**

作者：时间：2017-06-04来源：网络

本文引用地址：<http://www.eepw.com.cn/article/201706/348146.htm>

18世纪的发明家查尔斯。巴贝奇（Charles Babbage）是开路先锋。尽管他发明的“差分机”重达五吨，拥有20，000多个运动零件，这台机器仍然是卓越非凡的发明物，因为它是可以重复编程的。对于所有硬件而言，它便是从中脱颖而出的神奇软件。  
  
在现代，全球工程设计团队的调查显示，他们编写软件的时间要比创造硬件的时间更多。工程设计公司雇佣的程序员也比工程师数量更多，而且开发人员在选择MPU芯片之前，绝大多数都会首先选择运行芯片的软件环境。他们营销部门的同事深知，软件可以提供令产品卓然不群最为直接的捷径。原始代码可以提供“附加值”：独一无二的要素可以令一种产品在鱼龙混杂和快速发展的市场中脱颖而出。  
  
这不是说硬件并不重要：恰恰相反。硬件设计的重要性实际上已经变得显而易见。硬件需要快速推出，性能经过优化可以实现开箱即用，并且具备软件意识。  
  
接下来的步骤  
  
如果MPU只是达到目的的手段——即释放现有软件基础能力的途径或者钥匙——接下来重要的一步便是选择正确的钥匙。十分普及的处理器架构（例如Intel？ x86）可以提供丰富的x86软件配置，其中大多数都为个人计算机市场开发。同样，Power Architecture？、ARM？或者MIPS架构都配置包含了各自软件“俱乐部”的成员。如果目的是为了运行个人计算机软件，x86处理器属于必备的选择。对于嵌入式或者工业市场而言，开发人员的选择方案则更加灵活。  
  
当大多数程序仍然采用汇编语言编写的时代，处理器的选择也决定了程序员的选择。开发人员专门致力于各种不同的CPU指令集，勤奋钻研他们各自首选的ISA.代码本身与编写代码的人才相比同样一样不易转换移植；为一个处理器系列编写的软件对于另一个处理器系列完全无用，除非作为下一次的实例解释如何构建代码。编码的捍卫者会为汇编语言辩护，声称它的高效与快速，而他们的业务经理通常会嘲笑它开发缓慢、过度臃肿、只能针对于一种硬件平台、难以移植，对于没有深入原始开始工作的人员而言太过神秘莫测。  
  
理想情况下，软件可以实现快速和高效性，同时可以快速轻松地进行开发与维护。没有技术已经实现这种理想的情况，尽管已经实现了汇编水平编程的过渡，向更高水平的语言（例如C语言）迈进，这提供了极大程度的帮助。如今，不到10%的嵌入式开发人员还在普遍使用汇编语言；超过70%的开发人员几乎只在使用C或C++语言。可移植操作系统对此也有帮助。特别是Linux？适用于几乎所有处理器架构和所有硬件配置，使它成为了嵌入式开发人员近乎通用的平台。  
  
在可移植编程语言和通用平台之间，这个行业比以往任何时候都更有可能实现代码效率（从运行时性能角度考虑）和开发效率（从成本、上市时间和可维护性角度考虑）之间的理想平衡。性能与灵活性和易用性之间的平衡成为了嵌入式处理器供应商之间争相竞逐的领域。这便是软件感知架构变得至关重要的原因所在。软件感知架构是帮助客户通过性能优化的代码库和（便于实施的）软件，可以充分利用和轻松挖掘架构能力和功能的一种平台。  
  
[*Layerscape*](http://www.eepw.com.cn/news/listbylabel/label/Layerscape)架构介绍  
  
[*Layerscape*](http://www.eepw.com.cn/news/listbylabel/label/Layerscape)架构是下一代QorIQ LS系列片上系统（[*SoC*](http://www.eepw.com.cn/news/listbylabel/label/SoC)）的底层系统架构。从一开始便旨在充分利用新的开发、提取和效率现实条件（从字面的双重含义理解），[*Layerscape*](http://www.eepw.com.cn/news/listbylabel/label/Layerscape)架构的创建是为了让程序员找到极为轻松的方式“释放”每一块芯片的性能。该架构延伸了当前向多核芯片设计发展的趋势（包含同构和异构），以获得性能最大化，同时也可以提取足够的复杂硬件，以便让软件开发变得高效、可维护、灵巧、快速和相对简洁。简而言之，Layerscape架构可以实现手写汇编语言代码的性能和效率与高等级语言易用性和现代代码可维护性之间的平衡。  
  
Layerscape架构可以视为QorIQ P和T系列内[*数据路径加速*](http://www.eepw.com.cn/news/listbylabel/label/%E6%95%B0%E6%8D%AE%E8%B7%AF%E5%BE%84%E5%8A%A0%E9%80%9F)架构（[*DPAA*](http://www.eepw.com.cn/news/listbylabel/label/DPAA)）的演进——这是一种由多个CPU内核与加速器本身支持网络接口和加速器的基础设施。  
  
Layerscape架构方框图

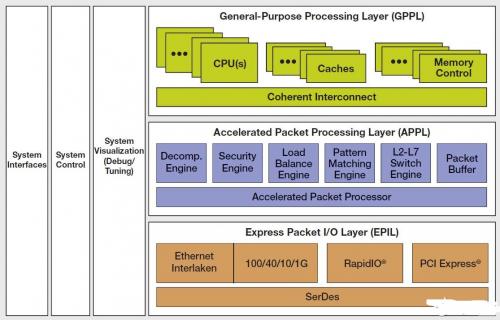


图1.所有LS系列芯片都采用逻辑方式（尽管并非总是从物理角度）被划分为三个层次。最高层包括任意类型的处理器，例如基于Power Architecture或者ARM技术的处理器。向下一层可以通过缓冲区、队列和API访问，能够提取实施项目的详细信息。  
  
Layerscape架构可以扩充[*DPAA*](http://www.eepw.com.cn/news/listbylabel/label/DPAA)，在ISO网络模型的独立层中单独地适当加速，这取决特定的芯片。有些芯片（例如）可以处理软件压缩，而其他芯片拥有专用的硬件加速器。无论采用哪种方式，功能对程序员都是透明的，这让一个芯片的实施切换到另一个芯片的实施变得简单直接，无需修改代码。结构化编程接口会包含压缩（在这个例子中），这样代码既不会调用它，也不会被它所调用，需要了解压缩实际上是如何应用的。再次重申，提取可以保存效率与性能，也可以保持开发人员的理智。  
  
在QorIQ LS系列中，每一个通信处理器按照逻辑方式划分为三层，如图1所示。通用处理层（GPPL）、加速包处理层（APPL）和快递包（express packet）输入/输出层（EPIL）分别粗略地代表了标准ISO模型的高、中和低层。无论芯片是否采用物理方式进行如此划分无关紧要；程序员认同这种方式，这与单个芯片如何进行分配无关。  
  
在最底层，快递包I/O层（图1，褐色部分）提供支持L2+转换功能的所有网络接口之间真正具有决定性的线速性能，并且包含芯片的网络数据报接口（例如Ethernet、Interlaken、Serial RapidIO？、HiGig和PCI Express？）。重要但不相关的接口（例如USB或者SATA）将不会属于这个接口层的组成部分，但会成为芯片“系统接口”模块的一部分，如图左侧所示（图1）。尽管严格来说，PCI Express并不是网络接口，它通常用作为堆栈中刀片之间的接口，因此也包括在此处。  
  
中间层（图1蓝色显示区块）包含芯片的包处理元素，它们或者是硬接线加速器、可编程引擎或者二者的组合。APPL可以通过传统顺序、同步、完整运行的模型提供客户定义的、自主和附加值功能，通过嵌入式C语言结构化编程实现完全可编程。再次重申，这些元素可以通过定义明确的接口与通用处理器进行通信，采用保留极具价值的开发人员代码的方式，提取它们（和处理器的）执行的详细信息。  
  
通用式处理器（图1，绿色显示区块）显而易见是属于通用性质，并且面向用户/开发人员免费提供，用于他们的操作系统、应用、高水平代码和其他附加值、功能。与Layerscape架构的提取、效率和硬件独立性的价值保持一致，这一层可以同时支持Power Architecture和ARM内核。一个人人皆知的事实是，Power Architecture技术通常采用二进位字节顺序，而ARM技术通常属于小端字节顺序，然而Layerscape架构可以轻松地支持这两种技术。  
  
显而易见，模块化硬件架构适用于众多不同的芯片配置，而且是横跨平台配置一致化软件的单一架构。模块化和灵活的硬件框架包括独立可扩展的层次，可以实现QorIQ产品组合的性能和电源效率最大化。如上所述，这些配置甚至包括不同指令集系列的通用型处理器，因此允许开发人员充分利用不同的代码库。Layerscape架构的模块性向上升级和向下降级——有时是在相同的物理套接字内——同时可以保留客户的代码。  
  
例如，一次极为基础的芯片实现可能只包括低水平接口（例如以太网）和高水平通用式处理器（即ARM或者Power Architecture内核），之间没有任何中介加速器。在这种情况下，EPIL层会对帧队列执行包解析、分类和分配（不会显示）。然后通用式CPU（或者可能为多个CPU）会消耗队列的这些数据包。

在多个以太网端口内扩展这一理念，利用Layerscape架构内置“链路聚合”（link aggregation）功能，同样的芯片可以作为第2层交换机。预分配容量更加充分的芯片可能在中间APPL包括硬件，以便实现颗粒度级的数据包分类、IPsec、SSL、LRO/TSO和其他高级联机卸载。与此类似，低水平EPIL可能会识别确定的数据包类型，并且直接将它们传送至中间APPL的相关加速器，完全绕开通用式处理器。  
  
该解决方案的基础在于其软件允许程序员快速且轻松地利用架构的能力。这款解决方案首先从优化的网络库入手，实现硬件加速功能（例如IPSec、深度包检测、IP转发、NAT/FW等），允许嵌入式开发人员专注于增值软件的开发，而无需进行性能调优。定义清晰的数据路径和控制API都是许多网络应用的标准配置，可以采用命令性C语言编程模型针对定制化应用轻松实现扩展。除此之外，软件框架可以提供标准服务（例如调试和配置、资源管理、虚拟化和初始化），以便确保易用性。最后，可以提供关键应用（例如软件定义网络、有线传输和回程、TCP终止和路由选择）的参考实现，这不仅可以降低您的研发投资成本，而且还能够加速上市时间。  
  
总结  
  
Layerscape架构将如今性能最强的通信处理器与全世界都在采用的相似的模块化、高水平编程模型相互结合。这无需硬件工程设计的高等级别，便可轻松获得高级通信引擎。更加重要的是，它不需要重新学习每个芯片实现的详细细节，可以作一代QorIQ LS系列器件由其后继产品直接取代。界限分明和定义清晰的编程模型可以在芯片之间、代代之间保存下来，这是基于开发人员的工作构建形成，而非将其视为硬件实现变更任务弃之不用。简而言之，Layerscape架构将开发团队最为重要和最具价值的方面保留了下来：即它独具特色的软件。再次重申，适当的硬件才是释放软件潜力的关键所在。