# **2相关技术介绍**

本论文研发的。。。

## 2.1嵌入式CPU

嵌入式CPU是嵌入式系统的核心部件，掘不完全统计，全世界嵌入式处理器的品种已有上千种之多。随着工业、医疗卫生、国防等各部门对智能控制需求的不断增长，同时也对嵌入式处理器的运算速度、可扩充能力、系统可靠性、功耗和集成度等方面提出了更高的要求，为了适应各方面的需求，嵌入式微处理器体系结构也经历了一个从CISC到RISC和Compact RISC：从4位、8位、16位、32位到64位；寻址空间从64kB到16MB甚至更大；处理速度从0.lMIPS到2000MIPS；常用封装从8个引脚到144个引脚的过程。处理器的功耗也有了明显降低；集成度进一步提高。

### 2.1.l嵌入式处理器概述

嵌入式系统的处理器可以分为下面几类:嵌入式微处理器（Embedded Microprocessor Unit，EMPU），嵌入式微控制器（Microcontroller Unit，MCU）式DSP处理器（Embedded Digital Signal Processor, EDSP），嵌入式片上系统（System on Chip）。

嵌入式微处理器的基础是通用计算机中的CPU，它一般装配在专门设计的电路板上，只保留与嵌入式应用有关的母板功能，但是电路板上必须包括ROM、RAM、总线接口、各种外设等器件。嵌入式微处理器目前主要有Am186/88、PowerPC、MIPS、ARM系列等。

嵌入式微处理器一般具备一下4个特点：

1、对实时多任务有很强的支持能力，能完成多任务并且有较短的中断响应时间，从而使内部的代码和实时内核的执行时间减少到最低限度；

2、具有功能很强的存储区保护功能。这是由于嵌入式系统的软件结构已模块化，而为了避免在软件模块之间出现错误的交叉作用，需要设计强大的存储区保护功能，同时也有利于软件诊断；

3、可扩展的处理器结的，以便能最迅速地开发出满足应用的最高性能的嵌入式微处理器；

4、嵌入式微处理器必须功耗很低，尤其是用于便携式的无线及移动的计算和通信设备中靠电池供电的嵌入式系统更是如此，如需要功耗只有mW甚至uW级;

嵌入式微控制器又称単片机，也就是在一块芯片中集成了整个计算机系统。嵌入式微控制器一般以某种微控制器内核作为核心，芯片内部集成ROM/EPROM、EEPROM、Flash、RAM、总线、总线逻辑、定时/计数器、WatchDog、I/O口、脉宽调制输出、A/D、D/A等各种必要功能和外设。微控制器出于比微处理器体积小、功耗和成本低、可靠性高，因而是目前嵌入式工业的主流，品种和数量都很多。其中，比较有代表性的通用系列有805l，P51XA，MCS-251，，MCS-96/196/296，MC68HC05/11/12/16，C166/167等。另外还有半通用系列如：支持USB接口的MCU8XC930/931，CS40，C541；支持CAN-Bus、LCD的众多专用MCU和兼容系列。

DSP处理器对系统结构和指令进行了特殊设计，使其适合于执行DSP算法，编译效率较高，指令执行速度也快。DSP应用正由在通用单片机中以普通指令实现DSP功能，发展到采用嵌入式DSP处理器。嵌入式DSP处理器的长处在于能够进行向量运算、指针线形寻址等运算量较大的数掘处理。比较有代表性的产品是Motorola的DSP56000系列，Texas Instruments的TMS320系列，以及Philips的基于可重置嵌入式DSP结构制造的低成本、低功耗的R.E.A.LDSP处理器。

而所谓的片上系统（SoC）则是在一个硅片上实现一个更为复杂的系统。各种处理器内核将作为SoC设计公司的标准库，成为VLSI设计中一种标准的器件，用标准的VHDL语言描述，存储在器件库中。SoC可以分为通用和专用两类。通用系列包括Siemens的TriCore，Motorola的M-Core，某些ARM系列器件等。而专用的SoC专用于某个或者果类系统中，不为一般用户所知。比如Philips的Smart XA，它将XA単片机内核和支持超过2048位复杂RSA算法的CCU単元制作在一块硅片上，形成一个可以加载JAVA或C语言的专用的片上系统。

当前，嵌入式系统处理器的发展趋势主要采用32位嵌入式CPU其主流系统有ARM（包括Intel公司的strong ARM和XScale）、MIPS和SH三大系列。

### 2.1.2 ARM QorIQ Layerscape 系列处理器

Layerscape架构是QorIQ LS系列片上系统（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，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架构（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架构方框图如下所示：

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，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。

## 2.2 PCIe和DMA

PCI-Express（官方结写为PCIe），是一个高速串行计算机总线标准，是Intel公司在2001年为了替代老的PCI和AGP标准而提出的。在2002年，PCI特殊兴趣小组拟定并推出了PCI-Express1.0标准。在2003年，PCI-Express1.1标准推出，对一些规范进行了进一步的声明和定义。在2007年，PC1-Express2.0标准推出，X1棋式的数据传输速率达到了500MB/s，与PCI-Express1.1标准相比提升了一倍。更进一步的，X4模式能够达到2GB/s的数据传输速率，而最高的X32模式能够达到l6GB/s的数据传輪速率。

既然PCI-Express总线支持这么高的数据传输带宽，那么如何才能充分利用这些带宽呢？如果使用CPU来直接负责整个数据传输过程，由于一般的嵌入式CPU并不具备非常高的主频，那么必将会拖慢整个数据传输任务的整体带宽；同时，在进行大数据传输时，这也会消耗过多的CPU资源，使得CPU无法去执行系统中的其它任务。在这种情况下，通常会选择使用PCI-Express专用的DMA控制器来进行实际的数据传输。

DMA（DirectMemoryAccess，直接内存访问）是一种允许外设在不使用系统处理器的情况下访问系统主内存的硬件机制。CPU只需要对DMA控制器进行相应的配置，后续的数据传输任务由DMA控制器来完成，从而大大减轻了CPU的负担，提高了系统的处理能力。

但DMA通常会导致缓存一致性问題。因为现代的CPU都是带有最存（cache）的，CPU直接读写的是缓存中的数据，而DMA访问的是外部内存中的数据。那么就会出现以下两种情况：

1．CPU更新完缓存中的数据后，没有将相应的数据更新到外部内存中，导致DMA访问到的是外部内存中旧的数据。

2．DMA更新外部内存中的数据后，缓存没有进行相应的数据更新，导致CPU访问到的是缓存中旧的数据。

对于这个问題，一般有两种解决方法：

1．使用一致性内存（coherentmemory）：由硬件来确保缓存和外部内存的数据一致性。当DMA写外部内存时通知缓存控制器更新相应的数据，当DMA读外部缓存时将缓存中的数据全部清空到外部内存中。

2．对子使用非一致性内存（non-coherentmemory）的系统，使用软件的方式来完成。当CPU读缓存中的数据时，由軟件来确保缓冲中的数据是有效的；当DMA读外部内存中的数据时，由软件来确保外部内存中的数据都是有效。

当包含PCI-Express总线接口的系统之间需要进行高速数据传输时，通常会使用PCI-ExpressSwitch芯片进行系统级互联，IDT公司的89HPES12NTl2G2是一款高性能的PCIeSwitch意片，专为PCI-ExpressGen2包交换而优化，包含12个通道和12个端口，支持多个同时进行的点到点数据流，支持PCI-ExpressGen1和PCI-ExpressGen2包交换。此外，89HPES12NT12G2支持非透明桥（non-transparentbridge）功能，可以初始化和翻译地址和设备ID，实现跨PCI-Express域的数据交互；集成DMA控制器，可以在不加重CPU负载的情况下进行高速的数据传输。

## 2.3高速差分传输

这一块目前还没想好，是只讲一下高速查分的大致原理，例举一下当前比较热门的几种差分传输技术CML，LVDS，PECL，还是重点介绍下CML运用于本文的，还是可以不写？

## 2.4事件驱动编程

事件驱动编程是。。。

## 2.5本章小结

本章重点针对。。。