# **2相关技术介绍**

本论文研发的基于LS1024A的海底油气勘探数据采集传输系统软件， 使用PCIe和DMA技术实现CPU和FPGA之间的数据传输，使用多核并行编程技术实现处理器性能的效率最大化，使用事件驱动编程的思想进行业务逻辑设计。本章将介绍本系统在整个研发过程当中所设计到的相关技术，包括嵌入式CPU，PCIe和DMA，多核并行编程技术和事件驱动编程技术。

## 2.1嵌入式处理器

嵌入式处理器是嵌入式系统的核心部件，掘不完全统计，目前全世界的嵌入式处理器品种已经有成千上万种之多。随着人工智能、物联网等行业的快速发展，人们对于嵌入式处理器的需求也不断增高，同时也对嵌入式处理器的运算能力、可扩充能力、系统稳定性、功耗和集成度等各方面提出了更高的要求，为了不断适应各方面需求，嵌入式处理器的体系结构也经历了指令集从复杂指令集运算（Complex Instruction Set Computing，CISC）、精简指令集运算（Reduced Instruction Set Computing，RISC）、显式并行指令集运算（Explicitly Parallel Instruction Computing，EPIC）到超长指令字指令集运算（Very Long Instruction Word，VLIW）；位宽从4位、8位、16位、32位到64位；寻址空间从64kB到256MB甚至更大；运算速度从0.l MIPS到5000 MIPS甚至更快；常用封装从8个引脚到725个引脚甚至更多。另外，处理器的集成度进一步提高，并且功耗也有了明显的降低；。

### 2.1.1 嵌入式处理器概述

嵌入式处理器可以分为以下几类：嵌入式微处理器（Embedded Microprocessor Unit，EMPU），嵌入式微控制器（Microcontroller Unit，MCU），DSP处理器（Embedded Digital Signal Processor，EDSP）以及嵌入式片上系统（System on Chip）。

嵌入式处理器的基础是通用计算机中的CPU，专门设计在指定装配的电路板上，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总线接口、ROM、RAM、以及各种外接设备等。目前主要的嵌入式微处理器有X86、DSP、PowerPC、MIPS、ARM系列等。

一般情况嵌入式微处理器具备以下4个特点：

1、实时的多任务处理能力。能够完成实时多任务，并且具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、强大的存储区保护能力。由于嵌入式系统的软件结构已模块化，为了避免各软件模块之间出现非预期的交叉作用，必须设计功能强大的存储区保护机制，同时也有利于软件诊断；

3、可扩展的处理器架构。能够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、功耗很低。嵌入式微处理器大都用在便携式场景，比如无线设备，移动通信设备等需要靠电池供电的嵌入式系统，其需要的功耗只有毫瓦甚至微瓦级；

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、Watch Dog、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SmartXA，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strongARM和XScale）、MIPS和SH三大系列。

### 2.1.2 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QorIQP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X：Layerscape系列处理器架构方框图

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PowerArchitecture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QorIQLS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（expresspacket）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、SerialRapidIO？、HiGig和PCIExpress？）。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，PCIExpress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。

### 2.1.3 Layerscape LS1024A处理器

LS1024A产品系列适用于从高端VoIP和视频家庭网关，中小型企业（SMB）高性能安全设备到以太网供电的802.11ac企业接入点和消费者网络存储产品等各种应用。

LS1024A系列处理器以现场强化的LS102MA为基础，大大提高了处理能力和VoIP密度，处理小数据包的线速，符合DRM标准的安全性以及企业级VPN和SSL吞吐量。新芯片集成了目前在CPE市场上出现的新功能，从而节省了大量的系统成本。

LS1024A利用ARM的高能效核心技术和飞思卡尔的低功耗设计流程，实现了同类产品中最低的功耗。此外，配套软件开发套件还提供了丰富的电源管理功能，以满足全球服务提供商和产品制造商的节能目标。

除了提供高吞吐量的IP Sec和SSLCPU卸载之外，LS1024A的板载安全引擎还包括一个功能强大的具有GZIP解压缩能力的深度包检测引擎。该设备的三个以太网接口允许DMZ配置为SOHO/SMB路由器和网关提供进一步的安全性。

LS1024AI/O接口与飞思卡尔创新的多层总线架构相结合，可跨所有数据接口实现无阻塞的并发事务处理，从而最大限度地减少片上数据包处理延迟。LS1024A的SATA-2接口以及强大的LRO/TSO和XOR引擎为网络附加存储应用提供了理想的解决方案。

为了提供性能可扩展性和最大的灵活性，LS1024A系列处理器包括从650MHz到1.2GHz的单个和双个ARM® Cortex® -A9内核器件，可提供高达6000DMIPS的性能。

LS1024A基于OpenWRT Linux的SDK针对单核和双核操作进行了优化。LS1024A双核设备交付的软件向后兼容LS102MA和100​​系列产品。

此外，飞思卡尔丰富的生态系统提供交钥匙解决方案，可缩短VPN/SSLSMB路由器，家庭网关，消费类NAS和企业接入点制造商的产品上市时间并降低开发成本。

下图给出LS1024A处理器的结构框图：

图X：ARMLS1024A处理器结构框图

LS1024A处理器的主要性能特征如下所示：

DDR3控制器-高达533MHz（DDR3-1066）的16/32位DDR3存储器接口，可选ECC支持。

•外围组件互连快速（PCIe）控制器-LS1024A装置包括符合PCIExpress基本规范修订版2.1和支持第一代（2.5Gbps的）和Gen2（5Gbps的）链路速率两个单道PCIe接口。

•通用串行总线（USB）2.0接口-与USB1.1向后兼容的USB高速（480Mbps）控制器和PHY。

•通用串行总线（USB）3.0接口-具有双总线架构的USB超高速控制器和PHY，支持并行USB2.0（高速，低速和全速）和USB3.0（超高速）操作。

•串行高级技术附件（SATA控制器）-两（2）个3GbpsSATA-2接口。

•时分复用（TDM）总线-全双工串行TDM总线，支持多达128个8。192MHz的时隙。

•扩展总线-提供用于连接系统外围设备的地址，数据和控制线。扩展总线为系统外设提供芯片选择，如闪存，引导ROM等。扩展总线支持5个片选：一个用于NAND，四个用于一般用途。

•IC间（I2C）总线-支持主机模式，从机模式或多主机模式。I2C总线可用于启动。

•高速串行外设接口（HS-SPI）-高达50MHz，2个从机选择。

•低速串行外设接口（LS-SPI）-高达16MHz，具有3个从机选择。LS-SPI接口可用于启动。

•通用输入输出（GPIO）接口-最多64个GPIO。八（8）GPIOS可以配置为接收中断。六（6）个GPIO可以进行脉宽调制。

•SerDes-三个可编程SerDes接口，最高可达5Gbps。

-SerDes＃0-PCIe0

-SerDes＃1-PCIe1或SATA0-通过引导进行选择

-SerDes＃2-SATA1或SGMII-通过引导选择

•Silicon Labs集成串行接口（ISI）-使用Silabs SLIC降低成本设计的3引脚接口。

•Zarlink串行接口（ZSI）-4引脚接口，使用Zarlink SLIC降低成本设计。

•双通用异步接收器/发送器（UART）-支持RS-232和流量控制。UART0支持蓝牙，频率高达3Mbps。

•IC间声音（I2S）接口-主机或从机，采样频率高达96kHz，每通道8/16或24位。

•联合测试行动组（JTAG）接口-JTAG接口提供对两个Cortex®-A9内核的访问。它支持IEEE1149。1和边界扫描制造和测试。

•参考时钟-使用48MHz或24MHz的单个振荡器/晶振。

## 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（Direct Memory Access，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（coherent memory）：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-coherent memory）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-Express Switch芯片进行系统级互联，IDT公司的89HPES12NTl2G2是一款高性能的PCIe Switch意片，专为PCI-Express Gen2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-transparent bridge）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多核并行编程技术

多核处理器发展的同时对软件行业带来巨大的影响。在多去单核时代，程序员基本上不用考虑并行编程的问题，所以传统的軟件大多都是基于串行单线程模式编写，将传统的应用程序直接运行在多核处理器上时，程序只能运行在单个核上，多核的运算能力无法被充分利用，不能发挥出多核处理器高性能的优势。在多核处理器的要件架构中，只有采用软件多进程、多线程技术方能充分发挥硬件的性能，线程分配到每一个核上，使得每一个线程都能得到一个核运行，多个线程并行运行。另外还要注意的是单核的多线程技术和多核多线程的区别，多核多线程技术是在多个物理核上的并行操作，真正意义上的并行执行，而单核上面的并行操作，实际上只是宏观上的一种并发运行，尽管核上有多个线程，但是同一时刻仅有一个线程在运行，多个线程按照时间先后依次运行，并非真正的并行运算。针对这样的问题，人们开始引入并行编程技术，从多个方面对并行编程进行设计和优化。首先针对多核处理器操作系统进行改进优化，比如多核间进程调度算法，核间负裁均衡算法等，以更好地支持多核处理器。其次在应用程序设计架构上，除了多线程编程技术得到更加广泛地应用，人们设计实现了一些并行编模型比如OpenMP和MPI。下面将分别介绍多线程编程技术和并行编程模

在Linux编程环境下，Linux系统提供了一套多线程编程接口，即Pthread线程库。该库符合POSIX标准，被各种UNIX操作系统所支持。相比于多进程编程，多线程编程有如下优点：创建线程的效率高，线程可使用其所属进程中的资源，因而创建线程的速度要比创建进程快；线程间IPC通信方式更加简单和高效，由于进程间的地址空间是相互独立的，进程间通信需要操作系统提供特定方式的支持，比如共享内存和域套接字（Domain Socket），但是线程之问是共享全局数据的，可以直接通过全局变量通信；线程间调度切換的代价也小于进程问的切換。而且在一些网络服务器上通常会频繁地响应请求，比如本文设计的视频监控服务器，需要频繁地响应客户端和前端设备的命令或是回应，所以其通常会在短时间内处理大量的务请求。在此环境下我们就可以利用多线程技术在初始化的时候创建一定数量线程的线程池，通常情况下池中线程处于阻塞状态，当服务器收到请求的时候，就从线程地中取出空闲的线程执行该请求，执行完后该线程重新变成阻塞状态，等待下一次请求的到来。当系统比较空闲时，大部分线程都一直处于阻塞状态，线程池可以自动销毁一部分线程以回收资源。相比于传统简单的多线程编程，即“即时创建，即时销毁”，也就是一旦收到一个请求后，创建一个新的线程，然后由该线程执行任务，执行完毕任务后，线程退出。尽管创建线程的时间与创建进程相比已经大大的缩短，但是如果每次线程执行任务的时间都非常短，而且服务器接收到请求的次数非常频繁，那么服务器将处于不断地创建和销毁线程的状态。如果我们将线程执行过程分为三个过程：T1、T2、T3。

T1：线程创建时间；

T2：线程执行时间，包括线程的同步等待的时间；

T3：线程销毁时间；

那么我们可以看出，线程产生和退出的开销所占的比例为(T1+T3)/(T1+T2+T3)。如果线程执行的时间T2很短的话，这比开销可能占到50%左右[]。如果任务执行次数非常频繁的话，这笔开销将是不可忽略的。因此线程地的出现正是着眼于减少线程创建和销毁带来的开销[]。线程地采用预创建的技术[]，基于这种预创建技术，线程池将创建和销毁线程所带来的开销分推到了各个具的任务上，执行次数越多，每个任务所分担到的线程本身开销则越小，提高了多核处理器的利用率，结短了系统的响应时间。

除了多线程编程技术外，还有一些并行编程模型，比如OpenMP。OpenMP是一种面向共享内存以及分布式共享内存的多处理器多线程并行编程语言。它以线程为基础，通过编译指导语句来显示地指导并行化。OpenMP的：执行模型采用Fork-Join的形式，Fork创建新线程或者唤醒已有线程，Join则是多线程汇合。其典型的编程模式如下：

Fork-Join在开始执行的时候，只有一个主线程在运行，当主线程在运行过程遇到需要进行并行计算的时候，这时候一般由OpenMP编译指导语言指示，会派生出子线程来执行并行的任务。在并行执行过程，主线程和派生线程共同工作。在并行代码结東后，派生线程退出或者挂起，不再工作，控制流程重新回到单独的主线程中[]。其工作流程如图2。3所示；主线程在适行过程中遇到并行指导语言，然后派生出4个子线程共同完成任务，其中第四个子线程在运行过中又嵌套地派生出4个子线程完成其任务，新产生的线程组并不会影响原进程的执行，最后子线程完成任务后退出，所有的线程汇合到主线程处[]。

## 2.4闪存技术

事件驱动编程是。。。

## 2.5本章小结

本章重点针对。。。