第 33 卷 第 2 期 计 算 机 工 程 2007 年 1 月
Vol.33 No.2 Computer Engineering January 2007
·开发研究与设计技术· 文章编号：1000—3428(2007)02—0252—03 文献标识码：A 中图分类号：TP393

网络处理器中处理单元的设计与实现

李 诚[1,2]，李华伟[1]
(1. 中国科学院计算技术研究所，北京 100080；2. 中国科学院研究生院，北京 100039)

摘 要：随着网络带宽的飞速增长和各种新的网络应用不断涌现，原有的基于通用处理器和 ASIC 的互联网架构已经不能满足新的需求。兼具强大处理能力和灵活可编程配置能力的网络处理器逐渐得到广泛的应用。高性能的网络处理器通常采用多个并发的处理单元进行数据平面的快速处理，这些处理单元在网络处理器中居于核心的地位。该文讨论了网络处理器中处理单元设计需要考虑的因素，设计了一种较为灵活有效的处理单元架构，并进行了 FPGA 原型验证，证实了该结构的可行性。
关键词：网络处理器；处理单元；并行处理

Design and Implementation of Processing Element in Network Processor

LI Cheng[1,2], LI Huawei[1]
(1. Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080;
2. Graduate School, Chinese Academy of Sciences, Beijing 100039)

【Abstract】 With the rapid increase in network bandwidth and emergence of various new network applications, original solutions based on general purpose processor(GPP) and application specific integrated circuit (ASIC) can not fulfill the requirements of high performance, flexibility and extensibility. Network processors are getting wide acceptance with its powerful processing ability and programmability. Almost all the network processors of high performance consist of multiple processing elements which are powerhouse for deep packet processing process packets concurrently. This paper discusses the design of processing elements, proposes a flexible yet efficient architecture for processing element, and demonstrates its feasibility through FPGA prototyping.
【Key words】 Network processor; Processing element; Parallel processing

1 概述

计算机网络可以看成是由通信子网和接入子网的主机组成的。而子网则可看作路由交换设备和传输媒介互联而成。随着光传输技术的高速发展，传输媒介能提供的带宽已经不再是制约网络速度和应用的因素。随着带宽的增长，核心网向高速的交换迁移，而转发智能则向网络的边沿迁移，通信子网中路由器和交换机的处理能力逐渐成为互联网的带宽瓶颈[1]。

各种新的应用和协议也对路由交换设备提出了更高的要求。互联网上的业务包括了 VoIP、VoD、P2P、VPN、在线游戏、网格计算等多种复杂的应用，网络已经从带宽驱动为主转向应用驱动为主，要求高带宽的数据网和要求低延迟的电信网的融合成为发展趋势。这些复杂的应用往往要求对分组进行深度处理，除了分组头部外还需要检查分组的负载，从而要求路由交换设备具有更高的性能以及高度的灵活性和可扩展性。

传统的基于 ASIC(Application Specific Integrated Circuits)和 GPP(General Purpose Processor)的解决方案不能同时满足高性能和高扩展性的要求。在这种情况下网络处理器应运而生，提供了较为理想的解决方案。

网络处理器是专门针对网络应用的可编程器件，它同时具有GPP的灵活性和ASIC的高性能[2]。网络处理器通常包含大量的可编程并发处理单元，通过多个单元并行的对分组进行处理来提供很高的吞吐率，而处理单元的可编程能力则提供了极大的灵活性和可扩展性。同时每个处理单元还会同时运行多个线程以便对访存、协处理操作等产生的延迟进行隐藏。

本文讨论了网络处理器中处理单元设计的细节问题，针对实际需求和实现的约束提出了一种简洁有效的处理单元结构，用 FPGA 原型对该结构进行了验证，最后采用针对网络处理器的基准测试程序进行了测试和评估以及性能分析。

2 网络处理器的结构

由于网络应用的范围及种类极其宽泛，从低速的电话线接入到高速的核心网流量汇聚，从以太网交换到安全防火墙，从而对应产生了各种各样的网络处理器结构，可以简单到微控制器，也可以复杂到千万门级的系统芯片(System-on-Chip，SoC)。

而高性能的网络处理器则普遍采用包含多个精简的处理单元进行并发的处理，应用于边沿网络进行流量汇聚，应用于核心网络进行路由交换，同时还包括各种用于流量管理、存储缓冲管理、队列管理、校验计算等的协处理器或专门部件，如IBM的PowerNP[1]、Intel的IXP2400/2800[3]、Freescale的C-Port[4]等。网络处理器的结构有很多种，但是从功能上基本上都可以分成可编程处理核心、专用协处理单元、存储层次、片上互连机制以及外围接口等 5 大部分，它们的组织如图 1 所示。

基金项目：国家“863”计划基金资助项目(2003AA115120)
作者简介：李 诚(1983－)，男，硕士生，主研方向：网络处理器设计；李华伟，博士、副研究员
收稿日期：2006-02-03 **E-mail：**li_cheng@ict.ac.cn

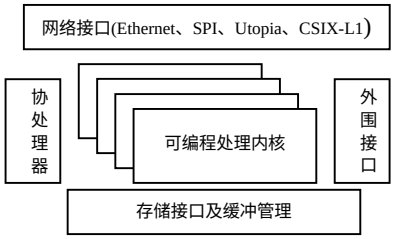
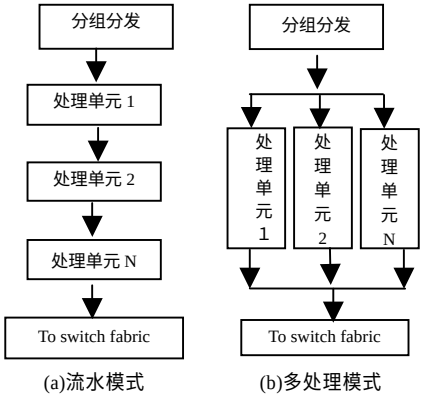


图 1 网络处理器的结构

其中的可编程处理核心为多个精简的RISC核或者VLIW核组成的阵列，它们为网络处理器提供了灵活的可编程能力。这些处理单元通常为高速并行的网络处理定制了专门的微结构和指令集，成为一种典型的专用指令集处理器(Application Specific Instruction Processor，ASIP)。这些处理单元的拓扑结构通常可以分成 2 种：一种为流水处理结构，较典型的代表为IXP2400/2800；另一种为多处理结构[5]，后者也叫RTC模式(Run-to-Complete Model)，较典型的如PowerNP，如图 2 所示。



(a)流水模式　　(b)多处理模式

图 2 处理单元的两种拓扑结构

3 处理单元设计

可编程的处理单元在网络处理器中居于核心的地位。由于网络处理器中包含大量的处理单元(16 到 64 个甚至更多)，因此处理单元的设计必须高效，才能满足网络处理器在性能、面积、功耗等方面的约束。处理单元的结构还必须具有足够的灵活性，方便编程配置以及对各种协处理器和接口进行管理。同时处理单元的结构还应该简洁规整，模块划分应该合理，从而方便设计正确性的验证。

由于网络处理器中的处理单元主要侧重于数据平面/快速通路的处理，运行的微码程序一般比较小，从而得以将指令存储全部放在片上，这可以极大地减轻处理单元取指部分的开销，在多个处理单元同时执行多个指令流的情况下更为显著，在片上通过采用多体交叉的指令存储形式可以满足多个处理单元需要的取指带宽。

另一方面由于延迟和面积的约束，片上的存储器不能做得很大，同时由于当前数据平面和控制平面具有融合的趋势导致了微程序的膨胀，因此用于网络处理器的指令集应该具有极高的代码密度，能以较少的指令完成复杂的功能。但是同时处理单元必须尽量精简，如果采用变长编码的指令集势必增加取指和译码的复杂度，因此在设计中采用了类似 MIPS 的 RISC 指令集，将复杂功能用协处理器来实现，处理单元通过简单的特殊指令来控制协处理器的工作。

鉴于面积的约束，处理单元应当简洁，采用单周期的设计可以节约硬件资源，但会严重降低处理器的运行频率和性能。即便单周期的设计的 IPC 可以达到 1，降低了运行频率之后总体性能仍然是下降的，因此简洁的流水结构的同时保持合理的 IPC 是很必要的。

采用流水结构会降低 IPC 的主要原因是数据/结构相关和控制相关(分支指令)造成的开销。适当的安排流水级和采用 Bypass 网络可以降低这些开销，但是某些技术有可能受限于实现相关的问题。例如，由于 Xilinx Virtex II Pro 系列 FPGA 上面能够大量提供的存储器资源为同步的，Virtual Silicon 针对 UMC 的.18μm 工艺提供的 Memory Compiler 也只提供同步的存储器，而由于采用触发器/锁存器搭建的寄存器文件占用太多 FPGA 资源，也采用同步存储器做寄存器文件，从而导致流水级的数目增加到 5 级，和经典的 MIPS R3000 流水线相似。而寄存器文件为同步读取则导致分支指令的延迟增加到 2 拍，在单个延迟槽的填充率都不甚理想(50%～60%)的情况下这种开销就可能会较大地影响某些程序的性能。

最终采用的处理单元的微结构如图 3 所示，其中，虚线部分为 Bypass 网络。处理单元共分成 5 级，包括 IF、ID/RF、ALU、MEM、WB。由于取指、访存和协处理器指令执行可能消耗多个时钟周期，如果此时停止整个流水线将导致流水线效率降低，同时为了方便扩展需要多周期完成的复杂运算以及消除 Load 指令的延迟槽，我们引入了流水线互锁结构。在前面流水级不能按时产生结果的时候向后面传递气泡，并且后续各级流水级寄存器在气泡传递过程中保持不变或者切换到空闲的状态，以便降低功耗，同时后面各级已有的指令继续向后流动。气泡在传递的过程当中如果碰到后续流水级暂停，则会被挤破(Squeeze)，仿真结果表明具有两拍延迟的分支指令在延迟槽之后插入的气泡会经常被挤破，从而一定程度上增加了流水线的效率。

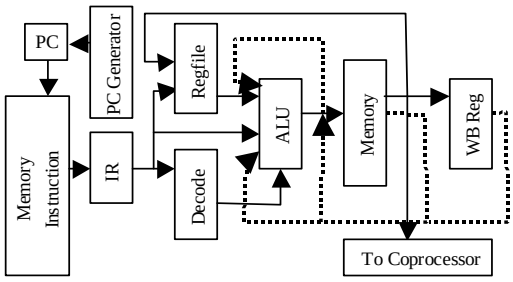


图 3 处理单元的流水结构

流水线的 Bypass 网络在一定程度上解除了数据相关，让操作数依赖于前面指令的后续指令得以尽早执行。除了通常会有的 ALU 到 ALU、MEM 到 ALU、WB 到 ALU 之外，还增加了 MEM 到 MEM 的 Bypass 路径，这有利于在内存中进行数据复制的操作，该操作在部分需要复制生成分组头部的应用中可能会很频繁。虽然更多的 Bypass 路径会导致延迟的增加，特别是在深亚微米工艺下可能需要插入 Repeater 等，但是对于我们的设计来说还是值得的。

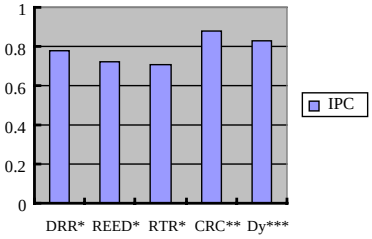
4 原型验证

我们对上面提出的结构用Verilog描述实现，并用 FPGA 进行了原型验证。在Xilinx Virtex II Pro 30(-6)平台上综合的结果表明单个处理单元运行的频率可达 80MHz，占用资源为 1 400 个 4 输入LUT，等价于 14 万系统门。采用Virtual Silicon 提供的UMC的.18μm的标准单元库进行综合的结果表明，单个处理单元可达到的频率为 380MHz，面积为 0.266 mm^2，大约 1.6 万门，功耗为 65mW，其中用作Regfile的双端口存储

器面积为 0.07 mm^2，功耗为 21mW。

当采用多个处理单元时，取指部分的 Crossbar 结构将占用较多的资源，成为频率的瓶颈。实验中我们在 Virtex II Pro 30(-6)平台上集成了 4 个并发处理的处理单元，加上指令存储、共享存储池、 Scratch Pad、MAC 控制器、UART 等的原型系统利用了 300 万系统门的 FPGA 上 90%的逻辑资源。FPGA 原型运行的频率达到了 50MHz，可提供 200MIPS 的处理能力，可以满足 4 个百兆以太网端口之间进行线速转发(按照每个分组花费 250 条指令进行计算)。

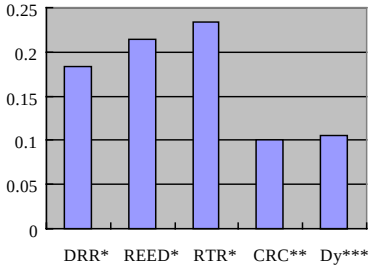
我们采用Dhrystone、Commbench[6]、NPBench[7]等测试基准集中的部分典型程序进行了测试(对测试程序针对我们的平台进行了修改)，结果IPC可以达到 0.78 左右(多个结果取几何平均值)，证明了设计的结构是可行的。部分结果如图 4 所示。



(*-Commbench，**-NPBench，***-Dhrystone)

图 4 测试程序运行结果

由于分支指令(包括条件跳转和寄存器间接跳转)的延迟较大(2 个时钟周期)，在无预测的情况下 IPC 受分支指令的数目和分支延迟槽的填充率影响较大，这里统计了程序运行中动态执行的分支指令所占的比例，如图 5 所示。



(*-Commbench，**-NPBench，***-Dhrystone)

图 5 动态运行分支指令所占的比例

可以看到，IPC 和分支指令的多少具有较强的相关性。可以改进的手段包括增加类似 MIPS II 当中的 branch likely 指令来提高延迟槽的填充率；提前计算分支条件来降低分支延迟；根据对应用程序进行 Profile 的结果进行简单的预测(如预测总是跳转或者总是不跳转)。这些措施不可避免地会增加硬件复杂度，从而增加面积和延迟，而这对于采用大量的处理单元的网络处理器来说是不可以接受的。

采用多个处理单元及 0.18μm 工艺实现的网络处理器将可以提供更高的吞吐率。在片上集成 16 个处理单元，运行频率为 380MHz，每个分组消耗 500 条指令的情况下可以提供的吞吐率为 3Gbps(最小分组为 40B)，可满足进行高速边沿汇聚的要求。

5 总结

本文详细讨论了网络处理器中的处理单元设计的相关问题，分析了流水线结构，并提出了一种简洁有效的处理单元实现方案。在我们的设计中处理单元采用具有互锁的 5 级流水结构，增加了 Bypass 路径，通过 Crossbar 共享片上指令存储，原型系统集成了 4 个处理单元、Scratch PAD、共享存储池、MAC 控制器、UART 等部件。我们通过部分网络处理器专用的基准测试程序的测试对原型系统进行了性能分析和验证，经过模拟仿真和 FPGA 原型验证的结果证实了该方案可行性，将该系统进行进一步的扩展、增加更多并发的处理单元即可满足边沿网络处理的需求。

参考文献

1 Allen J R. IBM PowerNP Network Processor: Hardware, Software, and Applications[J]. IBM J. Res. & Dev., 2003, 47(2/3): 177-193.

2 Shah N. Understanding Network Processors[C]. Proceedings of EECS'01, Berkeley: Univ. of California, 2000.

3 Intel IXP2800[EB/OL]. 2005-11. http://www.intel.com/design/network/products/npfamily/ixp2800.htm.

4 Freescale C-Port[EB/OL]. 2005-11. http://www.freescale.com/webapp/ sps/site/homepage.jsp?nodeId=02VS0lDFTQ3126.

5 Ning W, Wolf T. Pipelining vs. Multiprocessors: Choosing the Right Network Processor System Topology[C]. Proc. of Advanced Networking and Communications Hardware Workshop, Munich, Germany, 2004.

6 Wolf T, Franklin M. A Telecommunications Benchmark for Network Processors[C]. Proc. of IEEE International Symposium on Performance Analysis of Systems and Software, 2000.

7 Lee B K, John L K. NpBench: A Benchmark Suite for Control Plane and Data Plane Applications for Network Processors[C]. Proc. of the 21th International Conference on Computer Design, 2003.

(上接第 251 页)

2 Qiong Liu, Reihaneh S N, Nicholas P S. Digital Rights Management for Content Distribution[C]. Proc. of Australasian Information Security Workshop 2003: 49-58.

3 Bogdan C P, Bruno C, Andrew S T. A DRM Security Architecture for Home Networks[C]. Proc. of DRM'04, 2004.

4 Pestoni F, Lotspiech J B, Nusser S. xCP: Peer-to-Peer Content Protection[J]. Signal Processing Magazine, IEEE, 2004, 21(2): 71-81.

5 Andreaux J P, Durand A, Furon T, et al. Copy Protection System for Digital Home Networks[J]. Signal Processing Magazine, IEEE, 2004, 21(2): 100–108.

6 Reihaneh S N, Nicholas P S, Takeyuki U. Import/export in Digital Rights Management [C]. Proc. of DRM'04, 2004.

7 Richard G. Requirements for DRM Systems[C]. Proc. of Digital Rights Management, 2003: 16–25.