高性能同构多核媒体处理器

奚 杰[1], 朱 玥[1,2], 陈 杰[1]

(1. 中国科学院 微电子研究所, 100029 北京, xishilong@hotmail.com; 2. 中国科学技术大学 物理学院, 230026 合肥)

摘 要: 为了大幅提高处理器的处理能力,设计了一款5核结构的同构多核处理器并实现了H.264在多核处理器上的并行解码.该多核处理器采用1个CPU作为主控处理器,另外的4个CPU作为受控处理器被调用,5个CPU可以同时访问1块32 KWord的共享存储器,任意2个CPU之间可以通过邮箱、信号量、硬件锁实现点对点的通讯.其中,主控处理器负责码流解析、帧内预测模式、运动向量、边界强度的计算,并把中间数据存储到共享存储器中;4个受控处理器每次取1行宏块数据处理.试验结果表明,采用行并行的解码方式,5核系统较传统的双核系统达到约3.6倍的加速比.该并行解码方法同样适用于其他的视频协议,具有广泛的通用性.

关键词: 多核;邮箱;H.264;信号量

中图分类号: TN47 **文献标志码:** A **文章编号:** 0367-6234(2011)05-0094-05

High-performance homogeneous media mutli-processor

XI Jie[1], ZHU Yue[1,2], CHEN Jie[1]

(1. Institute of Microelectronics, Chinese Academy of Sciences, 100029 Beijing, China, xishilong@hotmail.com;
2. School of Physical Sciences, University of Science and Technology of China, 230026 Hefei, China)

Abstract: To improve processor's computation capability heavily, one multi-core system with 5 CPUs is designed and H.264 is decoded on it parallely. One CPU is used to lead the other four CPUs. Five CPUs share one 32 KWord SRAM. Any two CPUs can communicate point to point through mailbox, semaphore, hardware lock. When decoding H.264, the leading CPU is used to parse the bit-stream and calculate the intra prediction mode, motion vector, boundary strength. Then, the decoded data is saved to shared SRAM. Other four CPUs fetch and solve one macro-block line each time. The simulation result shows that using the line parallel decoding algorithm, the five-cores system can achieve about 3.6 times acceleration than two-cores system. The line parallel decoding algorithm can also be used on other video protocol and it is universal.

Key words: multi-core; mailbox; H.264; semaphore

随着多媒体技术的发展,各种音视频标准如常见的H.264,MPEG4,AVS,MP3等越来越丰富,且编解码越来越复杂.另一方面,由于工艺极限的限制,单纯依靠提升频率来提升处理器性能的方法受到了限制[1].并且,不断增加的功耗也使得散热成为了一个难以逾越的障碍.在这种情况下,AMD正式推出了多核产品,自此以后,多核处理器[2-4]受到了越来越广泛的关注.

多核处理器可以分为同构[5]和异构[6]结构.异构结构采用不同的处理器组成多核处理器,从而可以根据任务的特性分配给合适的处理器处理,但是该结构需要对不同的处理器进行编程,其编译、调试环境都比较复杂;而同构多核处理器采用相同的处理器组成多核处理器,克服了编译、调试环境带来的挑战,但采用同样的处理器决定了同构多核处理器只适合处理某一特定的任务.为解决这个问题,一般都需要对处理器进行指令集的扩展,使得单核处理器可以处理更加广泛的任务,如Intel的MMX,AMD的3D NOW等[7-8].同构多核处理器有效增强了处理器的处理能力,但

收稿日期: 2010-02-01.
基金项目: 国家高技术研究发展资助项目(2009AA011700).
作者简介: 奚 杰(1982—),男,博士研究生;
陈 杰(1963—),男,研究员,博士生导师.

同时也对开发能够充分利用多核结构的应用软件带来了挑战. 目前需要将编译技术和开发工具更多的结合才能使多核软件获得成功. 新近出现的软件框架[9]为多核软件的开发提供了良好的起点，软件框架为多核软件定义了一套固定的框架，程序员只需要在框架内按照一定的规则编写代码就可以充分利用多核处理器.

1　同构多核媒体处理器

1.1　总体架构

本文实现了如图 1 所示的同构多核处理器. 它由 5 个 32 位的 CPU 构成. CPU 为自主研发的兼容 MIPS 32 指令集的 32 位通用处理器内核，可以同时接收 12 个中断源. 处理器采用哈佛结构，程序空间和数据空间分开，两者容量均为 8 KWord. CPU 内部采用 5 级流水线，包括取指(IF)、译码(ID)、执行(EX)、仿存(MEM)、写回(WB). 理想情跨下，每个时钟周期发送 1 条指令，并有 1 条指令执行结果写回寄存器堆. CPU 通过总线访问共享存储器(Share DM)、邮箱(Mailbox)、信号量(Semaphore)、硬件锁(Hardware Lock)、SDRAM 控制器等外设. 5 个 CPU 可以同时访问 1 块 32 KWord 的共享存储器，当访存冲突时候，首先允许优先级高的处理器访问. 其中 CPU_0 为主控处理器并负责系统的初始化、配置、任务的分配等操作. 任意 2 个 CPU 之间都可以通过双向邮箱通讯，每个邮箱拥有 4 个寄存器，分别对应为：数据寄存器、满标志寄存器、空标志寄存器及有效数据个数寄存器. 每个邮箱的深度为16 个字，每个字的位宽为 32 bit.

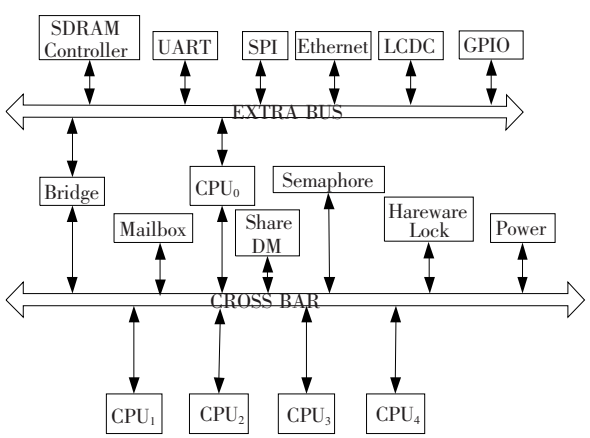

图 1　同构多核处理器

同时为了简化软件设计，本文设计了 16 个硬件信号量供各 CPU 使用，当系统复位时，硬件信号量初始值为 0，表示当前没有资源. 在多处理器中，硬件锁由于其高效、节约资源的特点而得到了广泛的应用，系统中一共设计了 64 个硬件锁供各内核使用. 另外，本设计中还有 1 个 Power 模块，Power 通过监测邮箱的状态，可以统计出运算当前任务所需的周期数. CPU_0 可以根据这个参数打开或者关闭其余的 CPU，从而实现功耗控制.

1.2　消息传递机制和共享地址空间的实现

消息传递机制通过底层硬件的支持，实现发送者和接收者的通讯. 本文分别采用邮箱、信号量、硬件锁实现了消息传递. 任意 2 个处理器之间都有双向的邮箱实现通讯. 当邮箱满时，写入操作会被阻塞；当邮箱空时，读取操作会被阻塞. 信号量是 1 个非负的整数，主要支持 2 种操作：push 和 pop. 其中：push 操作为当前有资源来到；pop 操作为消耗掉 1 个资源. 信号量对于操作系统中常见的临界区非常有用. 简化的二元信号量就是硬件锁，由于其占用资源非常少且实现简单，在多核处理器中得到了广泛的使用.

在多核处理器中，单一芯片上集成多个内核，并且片上存储器的资源非常有限，合理利用片上存储器将变的非常重要. 共享地址空间提供了一种合理、高效地利用片上存储器的机制. 考虑到芯片的面积、功耗，本文的共享存储器由 8 片 SRAM 构成，每片 4 KWord. 当不同的处理器访问不同的片时，不会引起访存冲突. 另外，当某一个 CPU 本地的指令存储器不够的时候，CPU_0 可以把共享存储器的一部分配置成 CPU 的本地指令存储器，而另外的部分仍作为共享数据存储器使用.

1.3　自适应功耗管理模块的设计

通过监控邮箱状态来统计处理器完成当前任务所需要的周期数，并用这个参数来控制处理器的关闭或开启状态. 在功耗控制单元中，共有 5 个初始值为 0 的 32 bit 计数器，分别对应 CPU_n(n = 0,1,2,3,4). CPU_0 通过读取计数器的值来控制 CPU_n 的开启和关闭. 这 5 个计数器一直在监控邮箱的状态，当接收邮箱里的信息表示启动任务，则计数器开始计数. 当发送邮箱里的信息表示任务结束，则计数器结束计数. 比如，CPU_0 完成码流解析后，就通过邮箱向 CPU_1 发送解码结束状态并等待，直到 CPU_1 完成解码并通过邮箱返回解码结束标志. 此时可以通过监控 CPU_1 的邮箱状态，统计出 CPU_1 完成解码的周期数. 如果这个周期数比实际要求的低很多，则可以关闭几个处理器，以实现功耗的控制.

2　多媒体指令扩展

MIPS32 指令集包括了 MIPSI，MIPSII 几乎全

部指令,同时也包括 MIPSIII、MIPSIV、MIPSV 指令集中的部分指令,共有 226 条. 根据功能,MIPS32 指令集可以分为 5 组,分别为:加载和存储指令、运算指令、跳转和分支指令、混杂指令和协处理器指令. 对于数据密集型运算、码流解析等任务,这些指令不太适合处理. 因此,本文采用向特定外部地址写数据的方法,实现了指令集的扩展,显著增强了单核处理器的处理能力并使之适合多媒体处理. 扩展指令集分为 2 类:1)码流解析类;2)数据运算类. 码流解析主要扩展了哥伦布解码、读取指定的位数等操作. 如表 1 所示.

表 1 码流解析指令

扩展指令(码流解析)	描述
golomb _ decode	哥伦布码解析
leading _ one	寻找最前面一个 1 的位置
prefix _ read	前缀 0 的个数
suffix _ read	读取后缀
read _ n _ bits	读取 n 位的数据

多媒体数据处理器中,像素都是 8 位的数据,通过把多个像素打包成 1 个向量,可以实现数据级的并行处理,这就是 SIMD(单指令流多数据流)的含义. 处理器把 4 个 8 位的数据打包成 1 个 32 位的数据,然后采用多媒体扩展指令处理,就可以显著提高单核处理器的处理能力. 具体的数据运算类扩展指令如表 2 所示.

表 2 数据运算类指令

扩展指令(数据运算)	描述
vec _ abs	向量取绝对值
vec _ add _ saturation	向量饱和加
vec _ min	向量取最小
vec _ mid	向量取中值
vec _ max	向量取最大
vec _ and	向量与
vec _ or	向量或
vec _ saturation	向量饱和
vec _ muliple	向量相乘
vec _ multiple _ para	向量带参数相乘

3 多核软件框架

3.1 H.264 分割模式

从并行任务处理数据的角度划分,并行方式主要分为功能并行和数据并行[10]. 功能并行是把不同的任务分配到各个处理器中,其所需的通讯资源较多,而且不易平衡负载,在多核处理器中较少使用;而数据并行是把相同的任务分配到各个处理器中. 比如每个处理器都处理 1 个宏块数据. 数据并行任务分配简单,处理器负载平衡易于实现,得到了较广泛的使用. 在多媒体解码中,常用的数据并行方法有 3 种,分别为以 SLICE 为单位并行、以宏块为单位并行和以行为单位并行. 在 H.264 中,SLICE[11]可以是一整帧,也可以是几个宏块,如果以 SLICE 为单位并行,则会导致整个处理器的负载难以平衡,同时过多的码流解析后的中间数据难以存放到共享存储器中. 现在较多的文献[12-14]是以宏块为分割单位,在处理器数目较少的情况下,这会取得较高的性能加速. 但由于宏块间的相互依赖性,在处理器数目较多的情况下会使得性能急剧恶化. 如图 2 所示,若假设当前 CPU 解码的是宏块 0,1,2,3,由于 4 个 CPU 依次得到数据,所以是依次启动. 而当 CPU_0 处理器完毕宏块 0,要开始处理宏块 4 的时候,宏块 4 需要宏块 3 的信息,而 CPU_3 可能刚开始处理宏块 3 的数据,这会导致 CPU_0 长时间的等待.

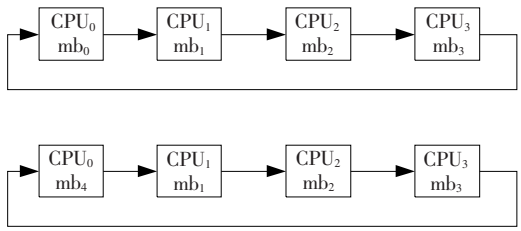

图 2 宏块并行

本文提出的以行为单位并行解码则可以解决这个问题,如图 3 所示. 由于各个 CPU 是依次启动,所以,各个 CPU 总是可以得到相邻的左面和上面宏块的信息. 当 CPU_1 解码完毕 1 行宏块后,CPU_1 开始解码新的 1 行宏块,此时 CPU_4 已经把上面的宏块解码完毕,而由于视频编码协议的特性,此时左面宏块不存在. 所以,CPU_1 可以直接解码新的宏块行.

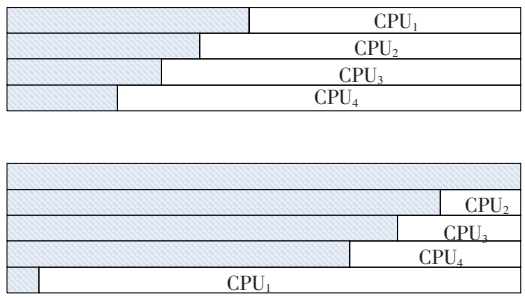

图 3 行并行

3.2 H.264 并行处理

本文在多核平台上实现了 H.264 的并行解码,以行为单位分割图像,采用 CPU_0 负责所有码流的解析和帧内预测模式、运动向量、边界强度的

计算，CPU_n（n = 1,2,3,4）接收解码后的数据，并且计算每 1 行的宏块数据. CPU_0 解码 4 行数据后，通过邮箱通知 CPU_n 数据准备好，CPU_n 接收到信息后，立即接收存放在共享存储器中的数据，并且开始解码. 为了实现流水处理宏块数据，共享存储器被划分成 2 部分. 负责码流解析的 CPU_0 和负责宏块数据处理的 CPU_n 在同一时刻只能工作在不同的片中，从而实现乒乓操作，任务分配如图 4 所示.

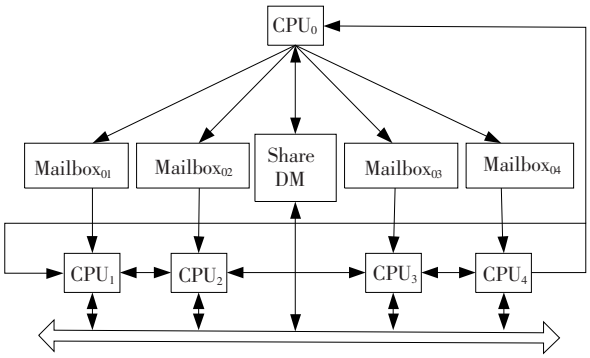

图 4　任务分配图

由于宏块间的相互依赖性，如帧内预测，环路滤波等，所以宏块间需要传递数据. CPU_n 之间通过共享存储器和硬件锁实现了数据之间的高效传递，如图 5 所示.

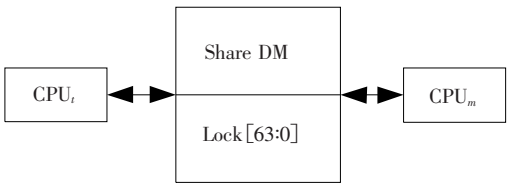

图 5　处理器间传递数据

CPU_t 完成宏块 a 后，把解码后的数据发送到共享存储器中，并且置相应的硬件锁为 1，表示宏块 a 的数据已经准备好. 当 CPU_m 解码到下一行相应的宏块 a 后，读取硬件锁的状态，如果为 1，则读取所需的数据，如果为 0，则不停地读取锁状态，直到锁状态变为 1.

4　实验结果

本文分别在双核至 5 核系统上实现了 H. 264 的解码. 其中：CPU_0 负责码流的解析，其余 CPU 负责宏块数据的解码. 多核硬件平台用 Verilog HDL 语言实现，用 Modelsim 进行功能仿真，并选用器件 Altera Stratix II EP2S180f1020 进行 FPGA 验证，如图 6 所示. 仿真和验证结果表明，系统运行在大约 364 MHz 时，可以实现 H. 264 标清（720 ×480@30）尺寸图像实时解码. 如表 3、4 中分别表示双核，3 核，4 核，5 核的情况下每个宏块的解码周期，由表 3、4 中可以看出，随着处理器数量的增加，每个宏块的解码周期在不断地减小. 其中：表 3 为没有使用多媒体指令扩展的仿真结果；表 4 为使用了多媒体指令扩展的仿真结果.

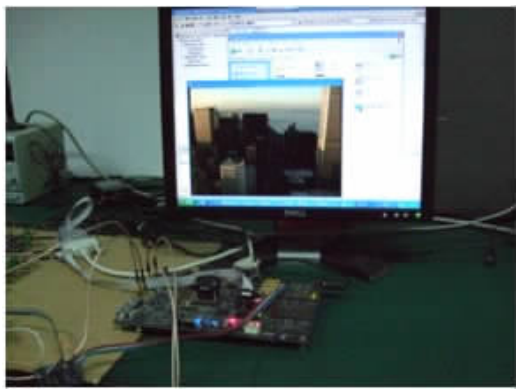

图 6　FPGA 验证图

表 3　无多媒体指令扩展仿真结果

序列	双核	3 核	4 核	5 核
City	50 533	26 811	18 325	12 652
Bus	51 102	27 342	19 034	13 908
Coastguard	52 943	23 095	17 043	13 801
Football	50 231	24 827	19 501	12 902

表 4　多媒体指令扩展仿真结果

序列	双核	3 核	4 核	5 核
City	33 023	20 752	13 893	8 532
Bus	33 871	22 182	14 760	9 903
Coastguard	34 923	18 754	13 645	10 642
Football	31 967	19 234	15 434	9 012

对于标清尺寸的视频，总共有宏块为

$$MB\_NUMBER = \frac{720 \times 480}{16 \times 16} = 1\,350. \quad (1)$$

根据仿真结果，若不使用多媒体指令加速，每个宏块所需的周期大约在 13 000 个左右. 假设处理器运行的频率为 f，处理器 30 帧/s 数据，则：

$$\frac{f}{1\,350 \times 30} = 13\,000. \quad (2)$$

$$f = 1\,350 \times 30 \times 13\,000 = 526. \quad (3)$$

而使用多媒体指令加速后，每个宏块处理的周期大约在 9 000 个，则：

$$\frac{f}{1350 \times 30} = 9\,000. \quad (4)$$

$$f = 1\,350 \times 30 \times 9\,000 = 364. \quad (5)$$

$$acceleration = \frac{526 - 364}{364} \times 100\% = 44\%. \quad (6)$$

由式（4）～式（6）可见，若不采用多媒体指令，则需要约 526 MHz 才可以实现 H. 264 标清图像的实时解码. 而采用多媒体指令加速后，只需要

364 MHz 就可以实现实时解码,可以使解码速度提高大约44%.

同时由表3、4可见,5核系统相对于双核系统可以得到约3.6的加速比,4核系统相对于双核系统可以得到约2.7的加速比,3核系统相对于双核系统可以得到约1.8的加速比.可以看出解码器的解码能力几乎随着处理器的个数实现线性的增加,这主要是由于采用以1行宏块为解码单位的软件框架,可以充分发挥多核处理器的性能.

与文献[12]相比,本文避免了适用硬件单元,具有良好的通用性,同时本文通过邮箱和信号量通讯,避免了适用复杂的微任务队列管理器.文献[18]在ARM11MPcore上实现了软件多核解码器,共有4个内核,加速比为2.3.而本文在4个内核的情况下,可以使加速比达到约2.7.

5 结 论

1)本文设计了一个能够并行解码H.264的同构多核处理器.

2)该多核处理器包含5个CPU,各个CPU之间共享32 Kword的存储器并且任意2个CPU之间可以通过邮箱、信号量、硬件锁实现点对点的通讯.

3)设计了可以并行解码H.264的多核软件框架,使得本文的多核处理器相对于传统的双核处理器,可以取得3.6倍的加速比.

4)该软件框架对于其他不同的视频协议同样适用.

参考文献:

[1] ROSS P E. Why CPU frequency stalled[J]. IEEE trans on Spectrum, 2008, 45(4): 72-72.

[2] XU Jiang, WOLF W, HENKEL J, *et al.* A methodology for design, modeling, and analysis of networks-on-chip [C]//Proceeding of 2005 IEEE International Symposium on Circuits and Systems. Kobe, Japan: IEEE press, 2005: 1778-1781.

[3] GORDER P F. Multicore processors for science and engineering[J]. IEEE Trans on Computing in Science & Engineering, 2007, 9(2): 3-7.

[4] BENINI L, De MICHELI G. Networks on chips: A new SoC paradigm[J]. IEEE trans on Digital Object Identifier, 2002, 35(1): 70-78.

[5] 于俊清, 李 江, 魏海涛. 基于同构多核处理器的H.264多粒度并行编码器[J]. 计算机学报, 2009, 32(6): 1100-1109.

[6] 李春江, 杨学军. 主从式单边异构多核处理器编程模型和编译架构[J]. 计算机工程与技术, 2009, 31(8): 66-68.

[7] BLINN J F. Fugue for MMX[J]. IEEE trans on Computer Graphics and Applications, 1997, 17(2): 88-93.

[8] OBERMAN S, FAVOR G, WEBER F. AMD 3DNow! technology: Architecture and implementations [J]. IEEE trans on Digital Object Identifier, 1999, 19(2): 37-48.

[9] SANGHAI K, GENTILE R, KATZ D, *et al.* 嵌入式多媒体应用的多处理器核软件设计框架[J]. EDN电子设计技术, 2008, 12(1): 102-105.

[10] BAIK H, SIHN K, KIM Y, *et al.* Analysis and parallelization of H.264 decoder on Cell Broadband Engine Architecture[C]//Proceeding of 2007 IEEE International Symposium on Signal Processing and Information Technology. Egypt: IEEE press, 2007: 791-795.

[11] 毕厚杰. 新一代视频压缩编码标准-H.264/AVC.[M]. 第5版. 北京: 人民邮电出版社, 2007: 138-144.

[12] 陈建文, 余 荣, 梅顺良. 视频多核处理器[J]. 清华大学学报, 2008, 48(1): 70-73.

[13] CHONG J, SATISH N, CATANZARO B, *et al.* Efficient paralleliazation of H.264 decoding with macro block level scheduling[C]//Proceeding of 2007 IEEE International Conference on Multimedia and Expo. Beijing, China: IEEE press, 2007: 1874-1877.

[14] ERIK B, EGBERT G T, ROB H. Mapping of H.264 decoding on a multiprocessor architecture[C]//Processings of SPIE Conference on Image and Video Comunications. CA, USA: SPIE press, 2003: 707-718.

[15] BENINI L, De MICHELI G. Networks on chips: A new SoC paradigm[J]. IEEE trans on Digital Object Identifier, 2002, 35(1): 70-78.

[16] LI-SHIUAN P, DALLY W J. A delay model for router microarchitectures[J]. IEEE trans on Digital Object Identifier, 2001, 21(1): 26-34.

[17] 侯 宁, 高明伦, 杜高明, 等. 二维网格NoC资源-网络接口[J]. 合肥工业大学学报, 2008, 48(1): 70-73.

[18] NISHIHARA K, HATABU A, MORIYOSHI T. Parallelization of H.264 video decoder for embedded multicore processor[C]//Proceedings of 2008 IEEE International Conference on Multimedia and Expo. Hannover, Germany: IEEE press, 2008: 329-332.

(编辑 张 红)