一种多处理器原型及其系统芯片设计方法

黄 凯[1],殷 燎[1],林锋毅[1],葛海通[2],严晓浪[1]
(1. 浙江大学超大规模集成电路研究所,浙江杭州 310027;2. 杭州中天微系统有限公司,浙江杭州 310012)

摘 要: 随着嵌入式应用快速发展,系统芯片(SoC)设计日趋复杂.高效可靠的设计多处理器系统芯片逐渐成为一个巨大挑战.本文提出一种多处理器原型及其 SoC 设计方法,将多处理器及其通信统一建模于一个多层次、灵活和可配的软硬件原型中,通过分层次、从高层抽象到底层实现逐步深入的方法解决软硬件接口验证问题和完善软硬件架构.H.264 解码实验证明多处理器原型功能可行性和物理可实现性.基于该原型的多层次细化方法可有效确保 SoC 软硬件设计的正确性,并有助于软硬件结构协同设计优化.

关键词: 多处理器原型; 系统芯片; 软硬件协同设计
中图分类号: TP368 **文献标识码**: A **文章编号**: 0372-2112 (2009) 02-0305-07

A Multiprocessor Prototype and Its SoC Design Methodology

HUANG Kai[1], YIN Liao[1], LIN Feng-yi[1], GE Hai-tong[2], YAN Xiao-lang[1]
(1. *Institute of VLSI design, Zhejiang University, Hangzhou, Zhejiang* 310027, *China*;
2. *C-Sky Microsystems Company, Hangzhou, Zhejiang* 310012, *China*)

Abstract: Fast development of embedded application drives the SoC design more complex. How to design multiprocessor SoC efficiently and reliably is becoming a challenge to the designers. To address this challenge, a new multiprocessor prototype and its SoC design methodology are proposed in this paper. It combines multi processors and their communication into one software-hardware prototype in different abstraction levels. The method of seamless refinement from high level abstraction to low level VLSI implementation can design and verify the software/ hardware interface and improve designing software/ hardware architecture efficiently. The experiment of H.264 decoder shows the feasibility of multiprocessor prototype in both function and physical implementation. The seamless refinement method based on this prototype can ensure the correctness of SoC design and be helpful for its software/ hardware architecture optimization.

Key words: multi-processor prototype; System on Chip (SoC); software-hardware co-design

1 引言

近年来,基于嵌入式处理器的系统芯片(SoC)日益成为 IC 设计的主流.面对日益复杂的嵌入式应用,多处理器系统芯片(MPSoC)凭借其高性能、并行处理能力和灵活的可配置性逐渐成为一个吸引人的解决方法[1].然而,高效可靠的 MPSoC 芯片设计仍面临着三方面的挑战:(1)软硬件编程模型;(2)软硬件协同设计和调试;(3)快速有效的系统集成和生成方法[2].因此,新的系统级多层次抽象建模和快速高效软硬件协同设计方法成为 MPSoC 芯片设计的关键,通过合理的处理器系统架构设计和软件多线程划分可有效的提高 MPSoC 芯片性能[3].

本文提出一种多处理器原型及其 SoC 设计方法:在不同抽象层次的软硬件建模和仿真基础上,实现系统软硬件设计、功能验证和性能评估,通过调配多处理器原型来完善处理器硬件体系结构及相应软件.

2 研究背景

当前,常用的 SoC 的原型设计方法主要分为两类:虚拟原型(Virtual Prototype)和快速原型(Fast Prototype).虚拟原型通过采用高层抽象语言(如 SpecC、System C 等)来描述 SoC 的硬件,从而实现虚拟抽象建模和高层软硬件协同仿真平台,虚拟原型通过提高抽象层来有效用于早期 SoC 体系结构探索和软件的初期验证[4].SpecC 设计方法[5]采用自顶向下逐层细化方法,通过设计规格模型、体系结构模型、通信模型和具体实现模型四层抽象虚拟模型来实现系统芯片的软硬件设计.CoWare ConvergenSC[6],Mentor Seamless CVE[7]等商用工具采用基于 SystemC 的虚拟原型方法,通过抽象硬件的操作行为,有效提高设计灵活度和仿真速度,但相应降

收稿日期:2007-11-30;修回日期:2008-10-20

低了仿真精度.当前虚拟原型大多建立在硬件基础上,软硬件原型的分离使得其协同设计效率降低.此外,由于应用的多样复杂化,构建一个包括处理器、通信和特殊功能 IP 的虚拟原型开发平台需要较多的时间和人力,而且面向复杂应用的高精度仿真效率仍然是一个瓶颈.快速原型通过采用硬件实体(FPGA 或处理器硬件原型芯片)来模拟或实现 SoC 的硬件,从而实现真实的软硬件协同仿真平台. Xilinx DSPTM System Generator[8]和 Altera DSP BuilderTM[9]等工具是从 Simulink 开始设计生成硬件体系结构,并将其实现入 FPGA.该方法真实的模拟了硬件的操作行为,通过实时仿真保证仿真精度.但是由于 FPGA 的局限性,这些只限于特定的处理器,而且现有 FPGA 工具也不能提供软硬件的协同生成,这也同样降低了软硬件协同设计效率.快速原型的另一种方法是采用处理器硬件原型芯片,例如 ARM 开发板提供 FPGA 芯片给用户去扩充 SoC 其它部分.但处理器硬件原型芯片通常无法配置,因此 SoC 的体系设计范围也相应受限.

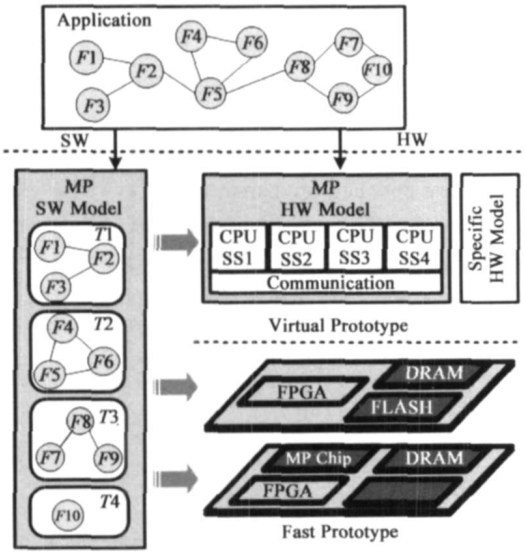


图1 基于多处理器原型的SoC设计框图

如图 1 所示,基于多处理器原型的 SoC 设计方法提出一种灵活且可配置的多处理器软硬件原型.与传统方法相比,它具有如下优点:(1)将 SoC 的处理器及其通信统一建模于一个多层次的软硬件原型中,利用原型的可重用性来加快平台开发周期;(2)提供软硬件协同建模和代码自动生成,利用设计自动化来提高软硬件协同设计效率;(3)从高层抽象虚拟建模(虚拟原型)到低层物理真实仿真(快速原型)实现不同层次软硬件协同设计.通过分层次、逐步深入的方法解决了软硬件接口难以验证的问题.

3 多处理器原型及 SoC 设计方法

3.1 多处理器原型

多处理器原型作为可重用的处理器架构原型,具备灵活可配的软硬件架构和接口,可根据应用的需求和软硬件协同仿真结果,相应调节多处理器原型的软硬件架构,从而实现系统最佳性能.为适应不同的需求,原型具有不同层次的模型:算法层(Simulink)、高层抽象(System C TLM)、RTL 层(Verilog)、FPGA(Xilinx)和硬件芯片.

3.1.1 硬件架构

多处理器原型的硬件架构如图 2 所示:四个处理器子系统(CPU SS)和存储器子系统(Memory SS)组成一个共享存储的多核架构.处理器子系统是多处理器原型的核心,承载着控制和数据处理的任务.存储器子系统可为处理器子系统提供灵活的片上数据存储和交互.此外,多核硬件调试控制器(Multi-Core HAD Controller)、性能监视控制器(Performance Monitor Controller)、高速缓存操作监视器(Cache OP. Snooper)和功耗管理器(Power Management)作为多处理器原型的辅助功能模块,可有效完善原型的硬件调试、性能监视、高速缓存同步和功耗管理功能.从整体硬件架构配置来说,为满足不同应用要求,处理器子系统的个数是可以配置的,多个处理器子系统可同构的并行工作,也可异构协同工作;对于处理器不使用高速缓存的应用,原型硬件架构可不包括高速缓存操作监视器;同样,存储器子系统和性能监视控制器也是可供选择的功能模块.

如图 2(*a*)所示,处理器子系统是一个基于处理器的最小系统(包括中断控制器、定时器等基本功能),所有子系统中的功能模块都是子系统私有的,对其它子系统均不可见.信箱(Mail Box)主要用于不同处理器子系统之间的数据通信同步.如图 2(*b*)所示,四个处理器的信箱与共享存储构成多处理器原型的通信架构.处理器子系统接口(CPU SS Interface)功能模块提供两组对外 AHB 接口分别连接到存储器子系统的存储器接口和其 AHB 总线上,从而可直接访问存储器子系统中的片上存储器资源及其总线上的 DMA 或通过外部接口访问多处理器原型外的资源.高速缓存同步模块(Cache Sync)通过处理高速缓存操作监视器捕捉的高缓数据操作来确保子系统内部处理器高速缓存与其他处理器同步.子系统监视器(SS Monitor)将按照性能监视控制器的要求,捕捉子系统内部各操作,并相应处理后返回给性能监视控制器.

存储器子系统由一块共享存储器和 DMA 控制器组成.从图 2(*b*)可以看出,存储器接口负责处理来自四个处理器子系统和 DMA 控制器的访问请求,并行分发给四个子存储器,从而有效提高存储器性能.该接口支持原子操作来消除共享存储器一致性问题.此外,各处理器还可通过存储器子系统 AHB 总线对系统内外部设备

访问配置.在通信子系统中,子存储器和 DMA 控制器通道的个数及大小都可以根据处理器个数或应用需求配置.子系统 AHB 总线位宽可以支持 32 位、64 位和 128 位,从而满足不同应用对总线带宽的要求.

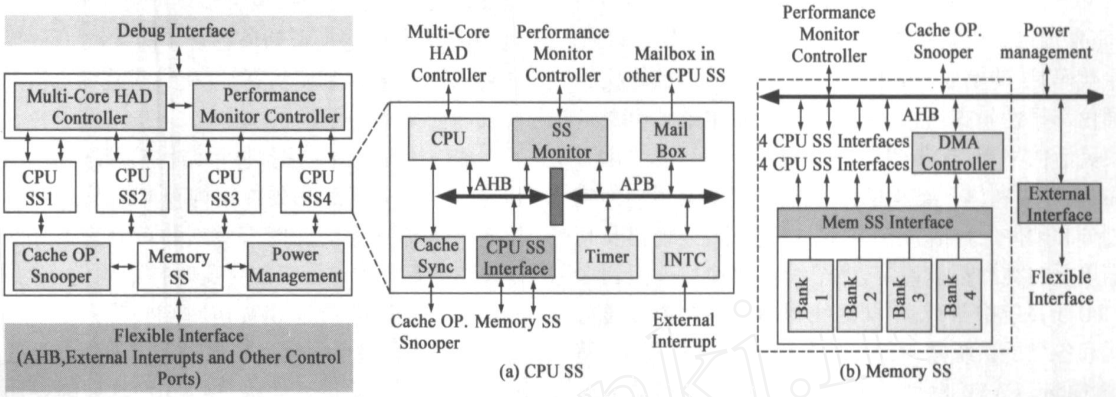


图2　多处理器原型的硬件架构

多处理器硬件调试控制器主要包括 TAP 状态机、多调试接口控制和状态寄存器.控制和状态寄存器直接决定着哪个处理器调试接口与外部调试接口相连接.在不改变单个处理器原有硬件调试接口的同时,控制器可实现外部硬件调试接口到内部多个处理器调试信号的控制.高速缓存控制器与各处理器子系统中的高缓同步模块协同工作,解决多处理器高速缓存的一致性问题.高速缓存操作监视器通过侦听总线(AHB 的 HPROT 等信号).当发现高缓操作时,将操作的地址和数据发给在各个处理器子系统中的高缓同步模块.各高缓同步模块(除这次操作发起者)将检查本地高缓中是否有相同地址,随后更新高缓,最后返回确认信号给高速缓存操作监视器.低功耗管理模块可通过对系统时钟的管理来实现多处理器功耗的管理.系统低功耗软件可根据应用的需求来控制处理器子系统的运行模式或调节系统时钟的频率去取得较好的能效比.

3.1.2　软件架构

多处理器原型的软件架构如图 3(*a*)所示,应用软件被划分到四个处理器子系统中,通过处理器子系统的软硬件接口,运行在多处理器原型的硬件平台上.如图 3(*b*)所示,处理器的软件堆栈由三部分组成:应用软件、与硬件相关的软件(Hardware Dependent Software, HDS)和硬件抽象层软件(Hardware Abstraction Layer, HAL).作为软件堆栈的最高层,应用软件包含应用的单线程或多线程代码,利用高层的 HDS API(如图 3(*c*)例中的 recv_data 和 send_data)来调用低层的与硬件平台相关软件.因此,上层用户开发的多线程软件代码具有较强的灵活性和可移植性.第二层是 HDS,主要包括通信库和操作系统,通信库为多种不同类型的处理器通信方式(GFIFO 和 HWFIFO)提供库函数.操作系统可是非常精简的线程库或复杂的 ucLinux 或 Linux.HDS 只关注硬件资源是否可获得等较高层次的硬件结构功能方面的问题,而底层的 HAL 则是包括如何获取或操作硬件资源的固件(Firmware)等其它基于硬件体系结构的软件,包括进程切换、中断服务程序、特殊硬件驱动或 IO 控制等.

多处理器原型软件堆栈的三层结构划分能够有效增加多线程软件设计开发的灵活度.软件设计师可以根据硬件需求,在三个不同层次上优化软件,从而提高软硬件协同设计性能.

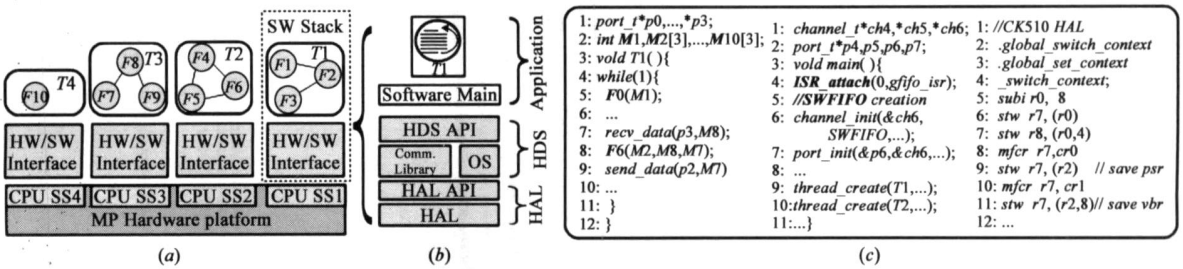


图3　多处理器软件原型及其处理器子系统软件堆栈

3.2　基于多处理器原型的多层次软硬件协同 SoC 设计

多处理器软硬原型的可配性和灵活性背后是复杂和困难的多处理器软硬件设计问题.基于多处理器原型的 SoC 设计方法是在基于 Simulink 模型的多核 SoC 设计方法[11]基础上优化而来,其基本流程如图 4 所示.

3.2.1　高层抽象建模(步骤 1 和 2)

Simulink 模型是对目标应用软件算法细粒度的功

能行为描述. Simulink 模型通常由三部分组成:Simulink 模块和 Simulink 互连. Simulink 模块是软件任务划分的基本单元,包括用户定义功能模块(S-function)和 Simulink 预定义功能模块. 在步骤 1 中,应用软件(C/C++代码)被转换成 Simulink 模型,将应用的运算核心和通信都显性暴露出来,为任务划分和体系结构探索提供灵活的平台. 在步骤 2 后,Simulink 模型被细化为 Simulink 算法和体系结构的结合模型(CAAM),包含软件划分和硬件配置信息. 在 CAAM 模型中,处理器及其通信单元的类型都被确定. 多个线程软件被分配到多个 CPU 子系统中执行,不同的线程通过系统内部通信单元和各个子系统之间的通信单元进行数据通信. 随后,Simulink CAAM 模型转换为基于 XML 描述的 Colif CAAM 模型. Colif 模型具有良好的数据结构,其通过模块、通道和端口去描述一个通用系统,可有效提高 CAAM 到低层建模自动生成效率.

3.2.2 中层抽象建模(步骤 3 和 4)

Colif CAAM 到低层硬件实现是一个复杂而繁琐的过程. 中间抽象模型通过分层细化的方法,逐步完成多处理器软硬件设计和验证. 中层抽象模型包括三个不同抽象层次:虚拟体系结构(Virtual Architecture,VA)层、传输精确(Transaction-Accurate,TA)层和虚拟原型(Virtual Prototype,VP)层. 多处理器软硬件模型从硬件到软件可分为三层:原型硬件系统、CPU 子系统和线程软件子系统.

虚拟体系结构模型是最为抽象的中层建模,它主要用于多线程应用软件的早期设计和验证. 原型硬件系统中包括了 CPU 子系统、存储器子系统和子系统间通信. 所有硬件子系统及其通信通道都是抽象模拟的,而线程软件子系统中仅包含多线程应用软件程序. 因此,虚拟体系结构模型抽象隐含了除线程子系统外的其它软硬件,从而加速仿真,可在早期避免死锁等多线程软件资源冲突问题.

传输精确模型是虚拟体系结构模型细化后的中层建模,它主要用于通信硬件架构和 HDS 软件的设计和验证. 存储器子系统、CPU 子系统中除处理器外的其它硬件和子系统的通信都细化成时钟精确的模型. 处理器仍是抽象模拟,通过 BFM 访问 CPU 子系统中的内部总线. 因此,传输精确模型抽象隐含了处理器及其 HAL 软件,着重于通信硬件架构及其 HDS 软件的设计和验证.

虚拟原型模型是最接近于底层实现的中层建模,它主要用于开发和验证整个软硬件系统和精确评估系统性能. 处理器已经细化成时钟精确的指令集仿真器(ISS),而基于处理器的 HAL 的设计使得软件堆栈变得完备.

如图 4 所示,硬件体系结构生成器(Hardware architecture Gen)和多线程代码生成器(Multithread code Gen)能从 Colif CAAM 自动生成三个中层建模的硬软件模型. 在硬件上,首先生成多处理器原型的系统;随后根据 Colif CAAM 中的配置,生成相应的 CPU 子系统、存储器子系统及其通信通道. 在软件上,首先生成各处理器的 Main 函数代码及 Makefile 等文件;随后生成相应的线程代码;最后通过链接 HDS 库,编译成可执行文件.

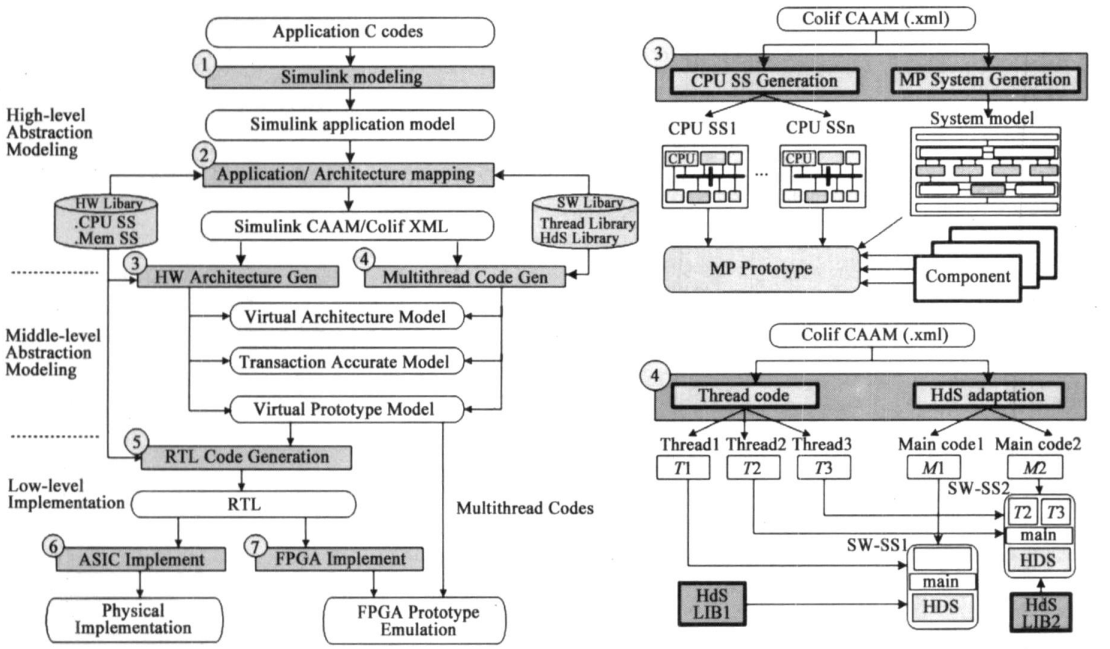


图4 基于多处理器原型的SoC设计方法

3.2.3　低层软硬实现(步骤 5、6 和 7)

通过中层建模,系统硬件架构和多线程软件划分基本确定.随后,多处理器原型可进一步细化至 RTL 实现.多处理器原型 SoC 生成器可根据 Colif CAAM 和用户配置生成 RTL 代码及其测试平台,从而有效提高 RTL 的设计和验证的效率. SoC 生成器的配置剖析器(Config Parser)从 Colif CAAM 和芯片配置文件中读取并分析体系结构配置和硬件工艺信息,将 IP 及其互连信息送给 IP 集成器(IP Integrator)和平台生成器(Platform Gen). IP 集成器根据配置信息,从 IP 库中选择需要的处理器、外设和存储器等,并相应生成系统和子系统的顶层文件.平台生成器可为整个设计和各 IP 提供测试向量,同时一些必要的工具和脚本也被自动生成,为整个平台提供反复回归(Regress)验证.最后,RTL 代码被用于 FPGA 原型验证或后端设计,细化至物理实现.

4　实验和流片

实验采用 32 位 CKCore 处理器作为多处理器原型的处理器核心,包括四个不同类型处理器:CK510、CK520(信息安全加密指令)、CK560(MMU)和 CK510E(DSP 扩展指令)[11].实验通过对 H.264 解码器的实现和分析,验证了基于多处理器原型及其 SoC 设计方法的有效性和可行性.

4.1　实验结果

实验中 H.264 基本档次解码器软件采用 JM7.3 优化代码,仿真实验是基于 30 帧 QCIF 的 Foreman 测试序列进行的. H.264 解码应用的 Simulink 高层建模由 83 个用户定义功能模块、24 个延时模块、310 数据连线、43 个 IF 操作子系统,5 个 FOR 循环子系统和 101 个预定义的 Simulink 模块组成.三个中层抽象建模(基于服务器:8 个 3.16GHz 的 Intel Xeon)和 FPGA 原型(基于 Virtex-4 XC4VLX160-FF15)仿真时间和精度如表 2 所示,VA 层仿真速度最快(18 秒),因此软件程序员可以利用它进行早期的多线程软件划分和并行化探索. TA 层通过细化通信硬件部分和软件操作系统,提高了时序精确度到 30%,但仿真速度相对减慢.如果在 TA 层仿真中使用时序反标技术[13],可使其仿真时序精度进一步提高到 80%,这可为多处理器原型体系结构探索提供高速有效的平台. VP 层仿真由于使用时钟精确的处理器 ISS,提供 90%以上的仿真时序精度,是精确系统性能评估的主要平台,但与其它层次仿真相比,其速度明显较慢,无法进行有效的多处理器原型体系结构探索.

表 1　各抽象层仿真速度和精度

Application	VA	TA	VP	FPGA
H.264 30-frame QCIF Foreman	18s	380s	4260s	～4s (50M Hz)
Timing Accuracy	0%	30～80%	～90%	100%

4.1.1　面向 H.264 应用的多处理器软件验证

面向 H.264 解码应用的多处理器软件中经 GNU GCC 编译后,各部分代码和数据大小如图 5(*a*)到(*d*)

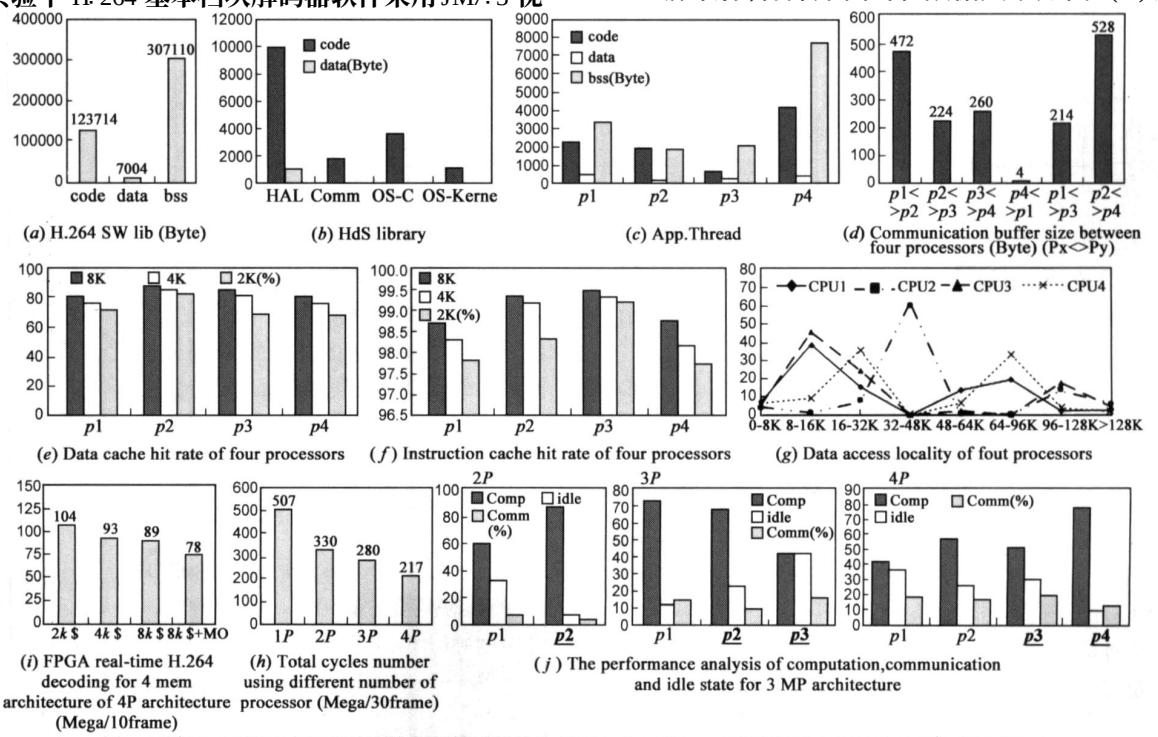


图 5　面向 H.264 解码应用多处理器软硬模型的软件代码大小(*a* 到 *d*)、存储器评估(*e* 到 *g*)和性能分析(*h* 到 *j*)

所示. H.264 应用库的代码段大小约为 120K,未初始化数据段大小高达 300K,这主要用于存放当前帧和参考帧. 在 VA 层仿真中,我们验证了基于该 H.264 库的四个线程并行应用代码的正确性. 处理器 2 负责总体的控制和一些预处理工作,而处理器 1,3 和 4 分别用于色度解码、亮度 Deblocking 和亮度补偿量化转换工作. 各处理器间的通信缓存大小如图 5(*d*)图所示,负责全局控制的 *p*2 与亮度处理器 *p*4 和色度处理器 *p*1 的通信量较大,每一个 MB 处理需有 0.5K byte 左右的数据通信. TA 和 VP 层仿真对 HdS 库代码进行验证,整个 HdS 的代码大小约为 17K:其中通信代码、OS kernel 和 OS C 代码大小分别为 1.8K、1.2K 和 3.8K,而 HAL 较大的原因是包括了存储器子系统中 DMA 控制器和外部 LCD 控制器等驱动程序.

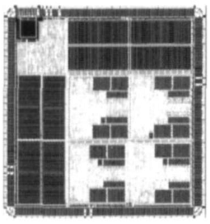
4.1.2 面向 H.264 应用的多处理器硬件性能分析

多处理器硬件存储器架构性能评估如图 5(*i*)到(*j*)所示,四个处理器在指令数据 Cache 分别都为 2K、4K 和 8K 时,H.264 应用仿真中 Cache 命中率呈递增趋势,8K 指令 Cache 的命中率平均都在 98.5%,因此由 Cache 未命中(miss)带来的填补(Refill)操作和外部存储器访问对系统性能影响几乎很小,但是,数据 Cache 即使在 8K 大小下的 Cache 命中率均低于 90%. 为提高处理器性能,如图 5(*g*)所示,VP 层仿真可得到对于不同地址段,不可 Cache 及 Cache 未命中数据访问次数分布情况. 通过分析可发现各处理器数据的方位性,将访问较多的地址块空间配置到存储子系统中片上存储器中,进一步减少片外存储访问来提高性能. 比如,对于 CPU2 处理器来说,近 70%的数据访问都在 16K 至 48K 的 32K 空间里,所以存储器子系统能提供一块子存储器将这块地址空间映射给 CPU2,可大幅降低片外存储器访问次数. 从 FPGA 原型(集成 8 块子存储器共 256KB,采用 DDR 作为外部主存储)性能评估的结果图 5(*i*)可以发现 Cache 命中率的下降明显降低了处理器性能,通过采用内部存储器地址空间优化(MO)可相应提高性能(约 12%).

为分析和验证多处理器模型在不同并行任务划分情况下性能,我们凭经验对 H.264 进行了三种任务划分,依次实现三种多处理器架构. 如图 5(*h*)所示(*x*P 表示共 *x* 个处理器被使用),随着处理器数目增加,处理器性能相应提高. 从图 5(*j*)可详细分析各处理器在不同架构下的性能,每个处理器的所有工作被分成三部分:运算(comp,解码工作),空闲(idle,处理器等待数据同步)和通信(comm,处理器间的数据通信). 在 2*P* 架构下,处理器 1(*p*1)负责宏块解码控制、预解码和色度解码工作,而处理器 2(*p*2)负责亮度解码工作. *p*2 处理器负载相对较高,因此我们将亮度解码工作细分给两个处理器:亮度 Deblocking 处理器(*p*2,4*P* 架构下的 *p*3)和亮度补偿量化转换处理器(*p*3,4*P* 架构下的 *p*4). 但此时 *p*1 处理器负载过高,我们继续细分其工作给两个处理器:宏块控制及预解码处理器(*p*1)和色度解码处理器(*p*2). 4*P* 架构比 1*P* 架构性能显著提高.

4.2 流片

多处理器原型的硬件芯片已在 SMIC 0.13μm 工艺下流片,其芯片样图及其参数如图 6 所示. 芯片共有 208 引脚,可向外扩展 AHB 总线. 芯片面积是 24.41 平方毫米,其中四个 CKCore 子系统和存储器子系统共占用了近 85%面积. 芯片在最坏情况(Worst Case)下,处理器可工作在 240MHz,而其它系统可工作在 120MHz,这可确保多处理器原型在两核和四核工作状态下,分别实现 20 帧和 30 帧 QCIF 的 H.264 实时解码.



SMIC 0.13μm	Worst Case: 240MHz (CPU) 120MHz (Bus)		
Area/ PAD	4.941 x 4.941 mm² / 208 PADs		
CK510	8K I$ 8K D$	476655 Gates	1.426x1.406 mm²
CK520	8K I$ 8K D$	479467 Gates	1.450x1.450 mm²
CK560	8K I$ 8K D$	521937 Gates	1.500x1.500 mm²
CK510E	8K I$ 8K D$	525505 Gates	1.503x1.484 mm²
On-chip Memory	256 KB (Totally 8 banks, 32 KB per bank) (0.833x1.764 mm²) x 8		

图6 多处理器原型的芯片样图及其参数

5 结论

为解决 SoC 软硬协同仿真设计的复杂度,本文提出一种多处理器原型及其 SoC 设计方法. 该处理器原型将软硬件模型结合于一体,提供灵活可配的架构,从而有效的提高设计效率. 基于该原型的 SoC 设计方法,通过分层细化,逐步完善软硬件接口及协同设计平台. H.264 解码实验证明了多处理器原型在功能可行性和物理可实现性. 基于多处理器原型的多层次仿真验证可有效确保 SoC 软硬件设计的正确性,并相应为软硬件结构优化提供有效帮助. 在随后的工作中,我们将开展基于该原型和方法的并行任务划分调度研究,从而完善该多处理器原型的系统级设计流程.

参考文献:

[1] A A Jerraya, et al. Special Issue on MPSoC[J]. IEEE Computer, 2005, 38(7): 36 - 40.

[2] Grant Martin. Overview of the MPSoC Design Challenge[A]. 43th DAC[C]. New York: ACM, 2006. 274 - 279.

[3] Ahmed Jerraya, Wayne Wolf. Multiprocessor Systems-on-Chip [M]. San Francisco: Elsevier Morgan Kaufmann, 2005. 11 - 13.

[4] Keutzer K et al. System-level design: Orthogonalization of con-

cerns and platform-based design[J]. IEEE Transaction On CAD of Integrated Circuits and Systems,2000,19(12):1523 - 1543.

[5] D D Gajski, J Zhu, R Dömer, A Gerstlauer, S Zhao. SpecC: Specification Language and Methodology[M]. Boston: Kluwer Academic Publishers,2000. 25 - 78.

[6] CoWare, SoC platform-based design using Convergen SC/SystemC[EB/OL],2002, http://www.coware.com.

[7] Mentor Graphics, Seamless CVE[EB/OL], http://www.mentor.com/products/fv/hwsw_coverification/seamless/.

[8] Xilinx, System Generator for DSP Performing Hardware-in-the-Loop With the Spartan?-3E Starter Kit [DB/OL]. www.xilinx.com/products/boards/s3estarter/files/s3esk_sysgen_hw_in_loop.pdf.

[9] Altera, DSP Builder[DB/OL]. www.altera.com.cn/products/software/products/dsp/dsp-builder.html.

[10] ARM, ARM Integrator CP Baseboard[DB/OL]. http://www.arm.com/documentation.

[11] 杭州中天微系统有限公司, 32 位高性能嵌入式 CPU 核 CK-CORE[EB/OL]. http://www.c-sky.com/product.php?id=5.

[12] Kai Huang, et al. Simulink Based MPSoC Design Flow: Cases study of Motion JPEG and H.264[A]. 44th DAC[C]. New York: ACM Press,2007. 39 - 42.

[13] Yi, Y Kim, D, Ha, S. Virtual synchronization technique with OS modeling for fast and time-accurate cosimulation[A]. In Proceedings of the Design Automation and Test in Europe[C]. New York: ACM Press,2003. 1 - 6.

作者简介:

黄　凯　男,1980 年 11 月出生于江西上饶,现为浙江大学超大规模集成电路设计研究所博士研究生,研究方向:MPSoC 设计、视频编解码和集成电路设计.

E-mail: huangk@vlsi.zju.edu.cn

殷　燎　男,现为浙江大学超大规模集成电路设计研究所博士研究生,研究方向:MPSoC 系统建模和性能估计.