·体系结构与软件技术· 文章编号：1000-3428(2017)02-0105-06 文献标志码：A 中图分类号：TN47

基于 2.5D 封装系统的存储型计算研究

尹颖颖[1]，虞志益[2,3]

（1. 复旦大学 专用集成电路与系统国家重点实验室，上海 201203；
2. 中山大学 中山大学-卡内基梅隆大学联合工程学院，广州 510006；
3. 广东顺德中山大学-卡内基梅隆大学国际联合研究院，广东 顺德 528300）

摘 要： 对于数据密集型应用，大量能量和延时消耗在计算和存储单元之间的数据传输上，造成冯·诺依曼瓶颈。在采用 2.5D 封装集成的系统中，这一问题依然存在。为此，提出一种新型的硬件加速方案。引入存储型计算到 2.5D系统中，使片外存储具备运算的能力。将存储器划分为若干个 bank，支持 bank 间并行访问，并在存储阵列中设计可配置的加速单元，充分利用存储阵列的带宽进行并行计算，降低数据传输的延时和能耗。以H.264解码中的反量化反变换为例对该结构进行实现，仿真结果显示，相较于传统软件实现方法，该方案可获得 7.1 倍的性能提升，节省 80.5% 的能量，并且只增加 2% 的面积开销。

关键词： 存储型计算；冯·诺依曼瓶颈；2.5D 封装系统；H.264 解码；数据密集型应用

中文引用格式： 尹颖颖，虞志益. 基于 2.5D 封装系统的存储型计算研究[J]. 计算机工程，2017，43(2)：105-110，119.

英文引用格式： Yin Yingying，Yu Zhiyi. Research on In-memory Computing Based on 2.5D Integrated System[J]. Computer Engineering，2017，43(2)：105-110，119.

Research on In-memory Computing Based on 2.5D Integrated System

YIN Yingying[1]，YU Zhiyi[2,3]

（1. State Key Laboratory of ASIC and System，Fudan University，Shanghai 201203，China；
2. SYSU-CMU Joint Institute of Engineering，Sun Yat-sen University，Guangzhou 510006，China；
3. SYSU-CMU Shunde International Joint Research Institute，Shunde，Guangdong 528300，China）

【Abstract】 For data-intensive applications，the large amounts of energy and latency spent in transporting data between off-chip memory and on-chip computing elements cause a limitation referred to as the von Neumann bottleneck. Even in the 2.5D integrated system，the bottleneck also exists. Aiming at this problem，this paper proposes a novel hardware acceleration framework that enables computing in off-chip memory array for 2.5D system. It divides the memory into multiple banks and puts an accelerator designed for H.264 decoder in the memory to utilize the high bandwidth provided by the memory array. Simluation result shows that，compared with traditional software implementation method，this framework achieves 7.1X improvement in performance and 80.5% reduction in energy consumption，and it only increases 2% accelerator area.

【Key words】 in-memory computing；von Neumann bottleneck；2.5D integrated system；H.264 decoding；data-intensive application

DOI： 10.3969/j.issn.1000-3428.2017.02.018

0 概述

性能提升一直是集成电路设计的优化目标，而消费类电子产品、可穿戴设备等市场的繁荣发展也使得功耗成为纳米时代电路设计的严峻挑战。如何在获得高性能的同时降低电路的功耗成为电路设计主要考虑的问题。计算单元和存储单元之间的数据传输是限制系统性能和能效提升的主要瓶颈。在传统的冯·诺依曼结构下，数据需要在计算单元和存储单元之间反复传输，而处理器带宽需求的增长速度高于存储器性能的提升，两者之间日益加大的差异造成了存储墙的问题[1]。层次性存储结构能够部分缓解 I/O 的瓶颈，但片上缓存在隐藏访存延时上已经达到了极限。

基金项目： 广东顺德中山大学-卡内基梅隆大学国际联合研究院项目(20150303)；三星电子横向课题(SLSI-201403DD013)。

作者简介： 尹颖颖(1990—)，女，硕士研究生，主研方向为多核处理器设计；虞志益，副教授、博士、博士生导师。

收稿日期： 2016-01-06 **修回日期：** 2016-03-10 **E-mail：** 13212020034@fudan.edu.cn

在系统集成策略上,片上系统(System on Chip,SoC)无法集成不同的工艺,并且系统灵活性较差,而采用传统封装技术的系统其性能极大地受制于片间互连[2]。先进的 3D 和 2.5D 封装工艺有望实现灵活性及高性能,成为未来系统集成的主流技术。3D 封装[3-4]将芯片在竖直方向上进行堆叠,能够实现不同工艺的集成,使芯片更紧凑、带宽更高、延时更短、功耗更低[5],但由于 3D 封装散热困难、设计成本高、耗时长,其测试难度大,整体风险很大,目前被较多地应用在存储器设计中,如三星和美光的混合存储立方。2.5D 封装[6-7]在硅中介上实现芯片在水平面上的堆叠,同样可以集成不同工艺,提高芯片间的通信速度,降低系统功耗,几乎拥有纯 3D 封装的所有优点,只是程度有所不及。此外,2.5D 封装设计复杂度低于 3D 封装,稳定性更高,在硅中介上实现裸片的堆叠,设计更灵活,提供了风险更低的方案,应用范围更加广。

然而在采用 2.5D 封装的系统中,冯·诺依曼瓶颈依然存在。为解决这一问题,本文将存储型计算技术引入到 2.5D 系统中,使片外存储具备运算的能力。同时利用 2.5D 系统封装灵活的特点,针对数据密集型特定应用进行加速,简化处理器与存储器之间的数据传输与同步,以最大化加速比,从而提升能效。

1 研究现状

存储型计算的概念可以追溯到 20 世纪 90 年代[8-9]。1995 年至 1996 年,圣母大学在 NASA 的资助下开展了一系列研究,试图以存储处理器(Processor in Memory,PIM)结构的大规模并行处理器(Massively Parallel Processor,MPP)实现高性能、低功耗的 Petaflops,无奈受制于当时的 CMOS 工艺水平以及存储器工艺,而且超标量、超流水等传统冯·诺依曼结构是主流趋势,最终并未成功。

然而,CMOS 器件尺寸的缩小使问题日益严重,加之新型存储器件[10-11]的出现,使得存储型计算再度吸引了研究者的注意力。文献[12-13]将存储型计算融入到传统通用处理器的结构中,形成混合式系统。文献[12]用存储型计算实现 GP-SIMD(General-Purpose SIMD Computer Architecture)架构中的数据同步,并把数据存储和大规模并行计算结合起来。该方案用片上存储实现向量的加减、移位等 ALU 操作,计算粒度较细,与处理器耦合紧密、通信频繁,并不适用于访存延时较大的片外存储。文献[13]将存储型计算引入到通用处理器的最后一级内存(Last Level Memory,LLM)中,在常规存储功能外,基于查找表赋予存储阵列计算功能并与定制的数据通路一起利用可编程的互连构成计算网络,在片外存储器中完成数据密集型应用的运算处理。该方案的优点在于存储器中的计算功能可灵活配置,但其可配置是以软件开销和硬件开销为代价的。

本文将存储型计算技术引入到 2.5D 系统的片外存储中,并利用 2.5D 中硅中介堆叠带来的灵活性,提出专用加速的设计思想,优化处理器与片外加速之间的通信,以最小的硬件开销实现加速器的调用和工作模式配置。同时选取 H.264 解码中的关键运算,设计加速单元,以较低的功耗获得较好的性能。

2 2.5D 多核系统

2.5D 系统的封装方式如图 1 所示。裸片倒装,用硅衬底之间的重布线层(Redistribution Layer,RDL)实现裸片之间的互连通信,用 TSV 通道引出系统的输入输出信号。2.5D封装拉近了芯片间的距离,尽管通信带宽和速度远高于传统印制电路板(Printed Circuit Board,PCB),但尚无法与片上互连匹敌,片间通信仍旧是影响系统性能和能效的关键因素。

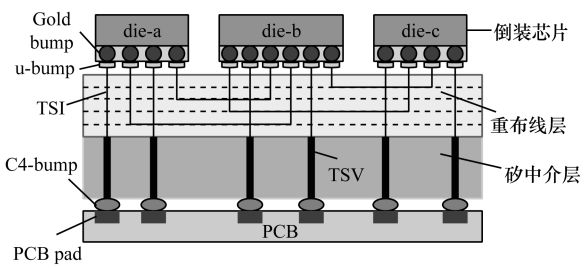

图 1 2.5D 系统封装

传统的 2.5D 多核系统[14]将多核处理器芯片、存储器芯片和加速器芯片封装在一起,如图 2 所示。处理器纵向扩展存储器,横向扩展加速器。其中多核处理器为 4×2 的 MIPS 核阵列,4 个核组成 1 簇,采用簇内分布式共享的存储体系[15]。片外的存储器可看作片上存储空间的扩展,处理器可以直接访问,也可以通过 DMA 访问。扩展的加速器芯片为加速阵列。

在该 2.5D 系统中,加速器和存储器之间传输数据需要跨过处理器芯片走 2 次 TSI 通道,严重增加了片间通信的负担,妨碍了性能和能效的提升[16]。将存储型计算引入到该系统中,在存储阵列中实现加速计算,可以降低 TSI 通道的通信压力,提升系统的性能。芯片间的通信延迟对计算的粒度及其与处理器的通信也提出了要求,其粒度要足够大才能保证加速效果不被通信延迟抵消,与处理器的通信要高效精简以减弱片间延时的影响。

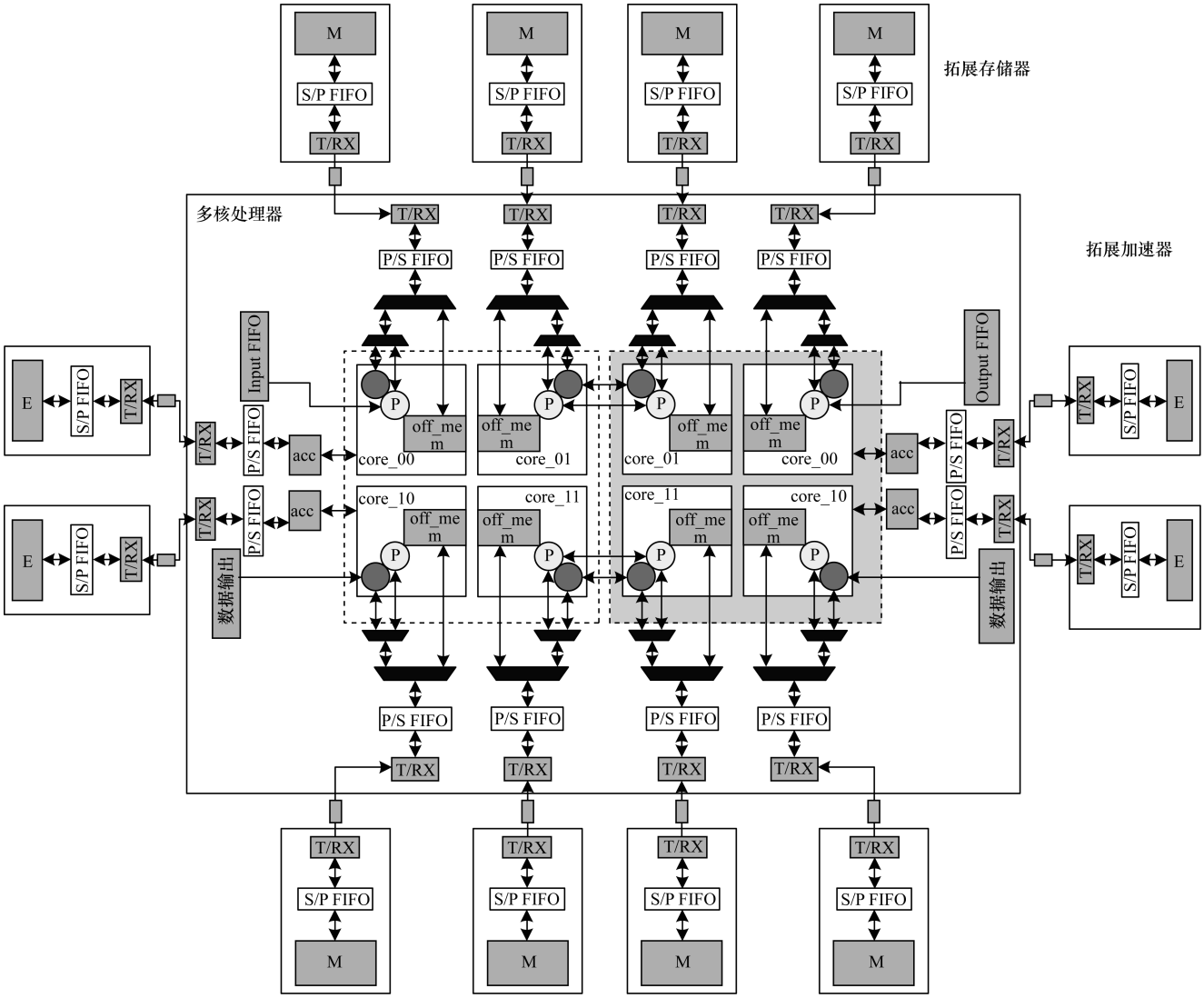

图 2　2.5D 多核处理器架构

2.5D 封装用硅中介实现裸片的堆叠,可以使用裸片级的 IP,设计复用度高,能够大幅降低优化升级系统的设计成本和时间,即可以在不改变整体设计的情况下,优化升级其中一个裸片来对整个系统进行升级。利用 2.5D 封装本身具备的灵活性,本文针对特定应用进行存储型计算设计,从而在保持系统灵活性和可扩展性的同时,最大程度优化加速器设计,提高性能,降低功耗。

基于以上的考虑,本文选取 H.264 解码过程中运算复杂、功能相对独立完整的反量化反变换模块进行加速。

3　基于 2.5D 系统的存储型计算结构

为减少系统数据传输上的性能和能耗,本文研究并设计了存储型计算进行片外存储及加速的扩展。本节将详细描述基于 2.5D 系统的可配置专用型存储型计算的硬件结构、调用方式,并对其优势进行分析。

3.1　硬件结构

工作模式可配置的专用型存储型计算将定制的加速器集成在片外存储器中,使得片外存储器既可以作为普通的存储器访问,也可以作为加速器使用,其硬件结构如图 3 所示。加速器针对 H264 解码中的反量化反变换进行加速,工作模式可配置。

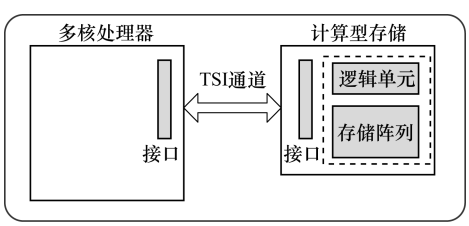

图 3　存储型计算的总体结构

3.1.1　接口电路

接口电路包含数字电路和模拟电路。定制设计的 TRx 部分为模拟电路,负责接收和发送数据,其中

发射器和接收器的功率分别为 15.2 mW 和 7.1 mW，延时分别为 1.16 ns 和 2.67 ns，数据传输速率为 8 Gb/s。数字电路部分通过纠错电路、串并转换电路将接收到的串行数据转换为 32 位数据，存储在异步 FIFO 中。整个接口电路可抽象为一个 32 位的异步 FIFO[16]。

3.1.2 存储型计算

存储型计算由控制器、存储阵列和加速器构成，如图 4 所示。控制器从接口电路读取数据，解析包头文件，控制访问存储器或者调用加速器。存储器被划分为若干小块，小块之间的访问相互独立，并采用交叉编码的方式编码地址，以支持加速器与存储器之间的多路数据传输。本文中的加速器可完成 H.264 解码器中并行度较高的反量化反变换操作。

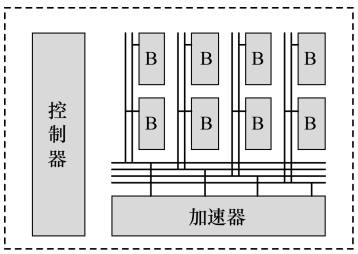

图 4 存储型计算的硬件结构

控制器支持处理器对存储器的多种访问请求，支持单字访问以及 DMA 传输，也支持处理器对加速器的调用。在调用加速器时，控制器控制加速器和存储器之间的数据传输以及加速器工作模式的配置。加速器从存储器指定地址获取输入数据，完成运算后将结果写回存储器的指定地址。加速器的工作状态也存储在存储器中，以供处理器查询。

3.1.3 可配置加速器

H.264 解码器的解码过程如图 5 所示，其中反量化反变换过程主要是矩阵运算，本身具备较高的并行度，非常适合存储型计算，因而本文针对该模块设计了工作模式可配置的加速器，另外加速器同时完成了与反量化反变换模块联系紧密的重排序模块。重排序过程完成逆 ZIGZAG 扫描（iScan），其扫描方式如图 6 所示。解码不同类型的残差宏块需要执行不同的操作，根据解码的宏块类型及图像参数 CBPL，共有 4 种不同的操作流程，因而加速器支持4 种反量化反变换工作模式，如表 1 所示。

反量化反变换操作的流程[17]如图 7 所示，主要的操作有反量化（Inverse Quantization，IQ）、逆整数离散余弦变换（Inverse Discrete Cosine Transform，IDCT）和逆离散 Hadamard 变换（Inverse Discrete Hadamared Transform，IDHT）。根据反量化反变换运算的特点，本文在以下方面对加速器进行了优化设计：反量化反变换运算中处理 4×4 的宏块，先对残差块的每行进行一维 IDCT 或 IDHT 变换，再对每列进行变换以完成反变换。在运算中，每次变换同时需要4 个数据。因此，将加速器与存储器传输带宽设为4×32 位，每个时钟周期读进 4 个数据，输出亦然，以减少加速器的等待时间。同时，设计状态机以流水线的方式完成数据的输入、行变换、列变换以及输出过程，以最少的时钟周期完成反量化反变换。在反量化反变换过程中，特别是反量化过程，存在大量常系数乘法运算，因此，选择采用移位加的方式实现乘法运算，在减少硬件开销的同时加快运算速度。

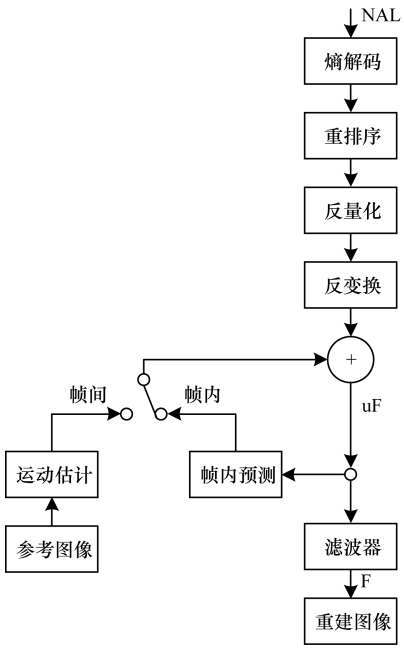

图 5 H.264 解码器流程

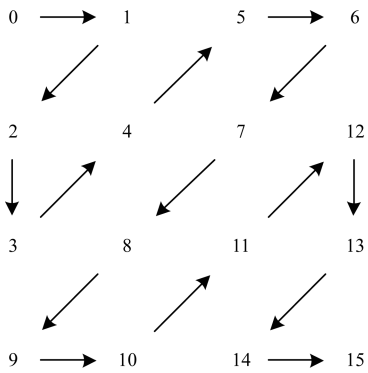

图 6 逆 ZIGZAG 扫描方式

表 1 反量化反变化加速器的 4 种工作模式

宏块类型	CBPL	分量	解码方式
4×4	1	ALL	iScan→IDCT→IQ→OUTPU
16×16	0	DC	iScan→IDCT→IQ→OUTPUT
16×16	1	DC	iScan→IDCT→IQ→STORE
16×16	1	AC	iScan→IQ→IDCT→OUTPUT

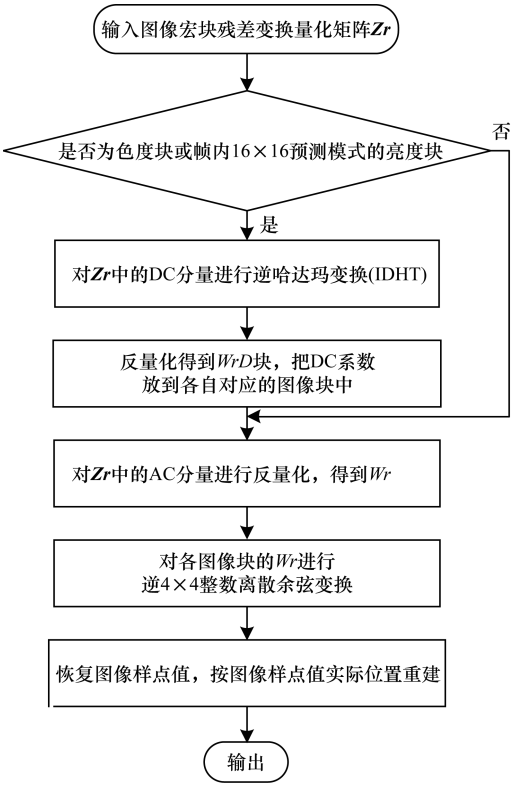

图 7　反量化反变换流程

3.1.4　访存及加速器调用方式

电路以极小的硬件开销实现了加速器的调用和工作模式配置。利用原有指令集的可扩展性,在不改动处理器设计的情况下,修改片外存储器的控制器,并在加速器中添加简单的译码电路实现加速器的调用和模式配置。2.5D 多核处理器[16]采用簇内分布式共享[18]的存储体系,同时支持丰富的直接内存访问机制。该存储体系既支持处理器以 load 和 store 指令对片外存储器进行单字访问,也支持以直接内存存取(Direct Memory Access,DMA)方式访问片外存储器。不管是以何种方式访问片外存储器,访问请求都以包头加数据包的形式通过 TSI 通道到达片外存储器。本文的可配置专用型存储型计算,在不改动原有指令集的基础上利用包头中的保留位实现加速器的调用和工作模式配置,利用 load 和 store 指令,通过单字访问片外存储器完成加速器的调用。写固定地址以调用加速器,读固定地址以查询加速器执行状态;加速器的配置信息以包头的形式写入固定地址,以配置加速器的数据流向和工作模式。

3.2　优势分析

本文设计的存储型计算可大幅减少数据在处理器和存储器间的反复传输,缓解了冯·诺依曼瓶颈,降低了在数据传输上消耗的功耗和时间,提高了系统性能和能效。充分利用存储阵列天然具备的丰富并行资源和高数据带宽对应用进行并行加速,性能优良。加速器不通过 CPU,可以直接访问存储器,减轻了处理器的负担,同时也降低了处理器和片外存储之间的通信压力。

加速器的工作模式可配置,可应对多种应用需求。虽然是针对特定应用进行加速,损失了一定的通用性,但是可重构可配置的加速器设计方式具有较好的灵活性。并且,作为片外加速,对加速器的改动并不影响整体的架构,不需要重新设计整个电路,改进加速器并不会带来庞大的设计成本。

4　仿真与结果分析

4.1　电路实现

电路中的逻辑单元采用 Globle Foundry 65 nm LPE 工艺库中的标准单元以最小时序约束进行综合,电路中的存储器是用 memory compiler 产生的单端口 SRAM,大小为 16 KB,总面积为 1.8×10^5 μm^2。在 1.2 V电压下存储型计算电路的工作频率为500 MHz,Prime Time PX 功耗分析结果显示完成一次反量化反变换操作的平均功耗为 4.72 mW,其面积和功耗的相对分布如图 8 所示。

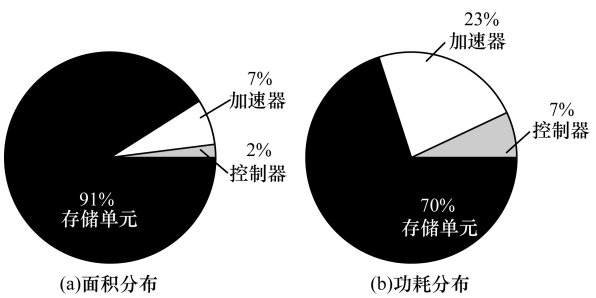

图 8　存储型计算的面积和功耗分布

4.2　性能和能效分析

在性能方面,与传统软件实现相比,存储型计算获得了7.1 倍的性能提升。将以存储型计算方式实现的反量化反变换加速器应用到 2.5D 多核系统上,在周期精确的 RTL 平台上(NCverilog + Verdi)进行仿真;作为对照,在同样的平台上以纯软件的方式在单核上实现反量化反变换功能,对比结果如表 2 所示。可以看出,存储型计算方案对存储器的访问从 793 次减少到了 8 次,访存量减少了 99% ,将时间和电路主要用于运算,而不是数据搬移上(不需要读取指令,也无中间结果的暂存),最终带来了 7.1 倍的性能提升,相当于在每个时钟周期平均完成 6.2 条指令。

表 2　存储型计算与纯软件实现的比较

实现方式	每个时钟周期完成的指令数	延迟/ns	存储器访问次数
纯软件方式	598.0	1 358	793
存储型计算	6.2	192	8

采用存储型计算完成一次完整加速器的调用，需要流水线执行访存指令，经片间接口电路传送请求给片外加速器、加速器执行操作。对上述过程的延时分别进行统计，得到的延时分布如表 3 所示，其中加速器执行反量化反变换操作只消耗了 42 ns，即 21 个时钟周期，运算速度得到了大幅的提升，是性能提升的主要原因。

表 3 存储型计算的延时分布

模块	延时/ns	占总延时比例/%
流水线	38	19.8
片间接口电路	112	58.3
加速器	42	21.9
总计	192	100.0

上述结果一方面得益于存储型计算在数据带宽方面的优势，加速器和存储阵列之间 4×32 位传输数据，大幅减少了加速器等待数据输入、输出的时间；另一方面得益于加速器的专用性，可针对特定的运算进行优化设计，反量化反变换过程中天然具备大量并行运算，加速器利用这一特点对运算进行并行加速，并对其中的乘法运算进行优化，用移位加实现常系数乘法，加快了运算速度。

需要指出的是，虽然片间接口电路的延时占到了总延时的 58.3%，这也已经是简化了处理器和加速器之间的通信后的结果，否则，两者之间的通信将严重阻碍性能的提升。

电路结构和算法上的优化同样带来了能效上的提升。用 Prime Time PX 分别对传统软件和存储型计算完成反量化反变换功能的功率进行统计，各自的平均功耗和消耗的总能量如表 4 所示，可见存储型计算能够节省 80.5% 的能量，功耗降低 59.7%。

表 4 存储型计算的能效提升

实现方式	平均功耗/mW	总能量/nJ
纯软件方式	11.70	15.90
存储型计算	4.72	3.10
减少比例/%	59.70	80.50

此处存储型计算所消耗的总能量包含流水线、片间接口电路以及加速器各自消耗的能量，它们所消耗能量的比重如图 9 所示，加速器完成运算消耗的电量仅占总消耗能量的 6%。

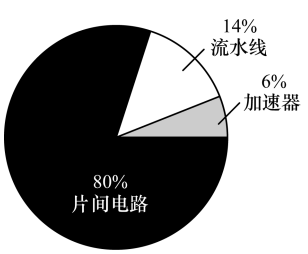

图 9 存储型计算的能量分布

此外，需要注意的是，用于数据传输与通信的片间接口电路的延时为总延时的 58.3%，且这一过程消耗的能量找到了消耗总能量的 80%，是性能和能效进一步提升的关键。

5 结束语

本文提出了一种存储型计算的结构，并将其应用到采用 2.5D 封装的多核系统中。将工作模式可配置的硬件加速电路集成到片外存储器中，在减少冯·诺依曼结构下数据传输带来的性能和能量损耗的同时，充分利用存储阵列提供的高带宽，最大程度地实现并行计算，进一步提升性能。以 H.264 解码过程中的反量化反变换为例对该结构进行实现，并与纯软件实现方式进行比较，结果表明，存储型计算以少量的芯片面积为代价，能够带来 7.1 倍的性能提升并减少平均 80.5% 的能量消耗。

考虑到与 CMOS 工艺的兼容性，本文选用了 SRAM 作为存储器进行存储型计算的研究，将 SRAM 划分为若干 bank 来提高数据带宽，但 SRAM 并非大容量片外存储的首选，因此，下一步可以将存储型计算应用到 DRAM 或 FLASH 等存储器中，以提供更大的数据容量。

参考文献

[1] Murphy R, Rodrigues A, Kogge P, et al. The Implications of Working Set Analysis on Supercomputing Memory Hierarchy Design[C]//Proceedings of ACM International Conference on Supercomputer. New York, USA: ACM Press, 2005: 332-340.

[2] Banerjee K, Souri S J, Kapur P, et al. 3-D ICs: A Novel Chip Design for Improving Deep-submicrometer Interconnect Performance and Systems-on-chip Integration[J]. Proceedings of the IEEE, 2001, 89(5): 602-633.

[3] Luo Guojie, Shi Yiyu, Cong J. An Analytical Placement Framework for 3D ICs and Its Extension on Thermal-awareness[J]. IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems, 2013, 32(4): 510-523.

[4] Lung Chiao-ling, Su Yu-shih, Huang Hsih-hsiu, et al. Through-silicon via Fault-tolerant Clock Networks for 3D ICs[J]. IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems, 2013, 32(7): 1100-1109.

[5] 吴 际，谢冬青. 三维集成技术的现状和发展趋势[J]. 现代电子技术，2014，37(6)：104-107.

[6] Sturcken N, O'Sullivan E, Wang Naigang, et al. A 2.5D Integrated Voltage Regulator Using Coupled-magnetic-core Inductors on Silicon Interposer Delivering 10.8A/mm^2[J]. IEEE ISSCC Journal of Solid-State Circuits, 2012, 48(1): 244-254.

[7] Huang Po-tsang, Chou Lei-chun, Huang Teng-chieh, et al. 2.5 D Heterogeneously Integrated Bio-sensing Micro-system for Multi-channel Neural-sensing Applications[C]//Proceedings of 2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers. Washington D. C., USA: IEEE Press, 2014: 320-321.

（下转第 119 页）

参考文献

[1] 王　泉,张学宏,周敏刚,等. 无人机飞控软件测试方法研究[J]. 航空计算技术,2008,38(2):78-81.

[2] Korel B, Laski J. Dynamic Program Slicing[J]. Information Processing Letters, 1988, 29(3): 155-163.

[3] Wong E, Qi Yu. Effective Program Debugging Based on Execution Slices[J]. Journal of Systems and Software, 2006, 79(7): 891-903.

[4] Wong E, Debroy V, Choi B. A Family of Code Coverage-based Heuristics for Effective Fault Localization[J]. Journal of Systems and Software, 2010, 83(2): 188-208.

[5] Naish L, Lee H J, Ramamohanarao K. A Model for Spectra-based Software Diagnosis[J]. ACM Transactions on Software Engineering and Methodology, 2011, 20(3): 563-574.

[6] Jeffrey D, Gupta N, Gupta R. Fault Localization Using Value Replacement[C]//Proceedings of 2008 International Symposium on Software Testing and Analysis. New York, USA: ACM Press, 2008: 167-178.

[7] Zeller A. Isolating Cause-effect Chains from Computer Programs[J]. ACM SIGSOFT Software Engineering Notes, 2002, 27(6): 1-10.

[8] Wong E, Debroy V, Golden R, et al. Effective Software Fault Localization Using an RBF Neural Network[J]. IEEE Transactions on Reliability, 2012, 61(1): 149-169.

[9] Brun Y, Ernst M D. Finding Latent Code Errors via Machine Learning over Program Executions[C]//Proceedings of the 26th International Conference on Software Engineering. Washington D. C., USA: IEEE Computer Society, 2004: 480-490.

[10] Nessa S, Abedin M, Wong E, et al. Software Fault Localization Using N-gram Analysis[M]//Li Yingshu, Huynh D T, Das S K, et al. Wireless Algorithms, Systems, and Applications. Berlin, Germany: Springer, 2008: 548-559.

[11] Jones J A, Harrold M J. Empirical Evaluation of the Tarantula Automatic Fault-localization Technique[C]//Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering. New York, USA: ACM Press, 2005: 273-282.

[12] Liu Chao, Fei Long, Yan Xifeng, et al. Statistical Debugging: A Hypothesis Testing-based Approach[J]. IEEE Transactions on Software Engineering, 2006, 32(10): 831-848.

[13] 周保宇,田　力. 高实时性的无人机飞控软件测试系统设计[J]. 计算机测量与控制,2012,20(9):2384-2385.

[14] Abreu R, Zoeteweij P, Golsteijn R, et al. A Practical Evaluation of Spectrum-based Fault Localization[J]. Journal of Systems and Software, 2009, 82(11): 1780-1792.

[15] Jones J A, Harrold M J, Stasko J. Visualization of Test Information to Assist Fault Localization[C]//Proceedings of the 24th International Conference on Software Engineering. New York, USA: ACM Press, 2002: 467-477.

编辑　金胡考

(上接第 110 页)

[8] Kogge P M. Final Report: Processor-in-Memory (PIM) Based Architectures for PetaFlops Potential Massively Parallel Processing[EB/OL]. (1996-07-15). http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19970001424.pdf.

[9] Elliott D G, Stumm M, Snelgrove W M, et al. Computational RAM: Implementing Processors in Memory[J]. IEEE Design & Test of Computers, 1999, 16(1): 32-41.

[10] Wang Yuhao, Zhang Chun, Yu Hao, et al. Design of Low Power 3D Hybrid Memory by Non-volatile CBRAM-crossbar with Block-level Data-retention[C]//Proceedings of ACM/IEEE International Symposium on Low Power Electronics and Dsign. New York, USA: ACM Press, 2012: 197-202.

[11] Wang Liyun, Zhang Chun, Chen Liguang, et al. A Novel Memristor-based rSRAM Structure for Multiple-bit Upsets Immunity[J]. IEICE Electronics Express, 2012, 9(9): 861-867.

[12] Morad A, Yavits L, Ginosar R. GP-SIMD Processing-in-Memory[J]. ACM Transactions on Architecture and Code Optimization, 2015, 11(4): 1-26.

[13] Paul S, Krishna A, Qian Wenchao, et al. MAHA: An Energy-efficient Malleable Hardware Accelerator for Data-intensive Applications[J]. IEEE Transactions on Very Large Scale Integration Systems, 2015, 23(6): 1005-1016.

[14] Lin Jie, Zhu Shikai, Yu Zhiyi, et al. A Scalable and Reconfigurable 2.5D Integrated Multicore Processor on Silicon Interposer[C]//Proceedings of 2015 IEEE Custom Integrated Circuits Conference. Washington D. C, USA: IEEE Press, 2015: 1-4.

[15] Ou Peng, Zhang Jiajie, Quan Heng, et al. A 65 nm 39GOPS/W 24-core Processor with 11Tb/s/W Packet Controlled Circuit-switched Double-layer Network-on-Chip and Heterogeneous Execution Array[C]//Proceedings of 2013 IEEE International Solid-State Circuits Conference Digest of Technical Papers. Washington D. C., USA: IEEE Press, 2013: 56-57.

[16] 林　杰. 2.5D 多核处理器关键技术研究——存储体系、片间接口及芯片实现[D]. 上海:复旦大学,2015.

[17] 毕厚杰,王　建. 新一代视频压缩编码标准——H.264/AVC[M]. 2 版. 北京:人民邮电出版社,2009.

[18] 欧　鹏. 面向通信和多媒体的多核处理器关键技术研究[D]. 上海:复旦大学,2013.

编辑　金胡考