文章编号:1007-5321(2017)04-0009-07 DOI:10.13190/j.jbupt.2017.04.002

面向阵列处理器的分布式共享存储结构设计

山 蕊[1], 沈绪榜[1], 蒋 林[2], 朱 筠[2], 宋 辉[2]
(1. 西安电子科技大学 微电子学院, 西安 710071; 2. 西安邮电大学 电子工程学院, 西安 710121)

摘要: 为了缓解随处理器核数增多而被激化的"存储墙"问题,提出了局部高速交叉互连、全局片上网络互连的两级混合互连网络结构,设计了支持统一编址方式的数据传送机制. 在现场可编程门阵列上实现了 2 种规模的存储结构,对面积、时序和功耗进行统计. 基于 SystemC 开发了混合仿真平台,仿真结果表明,所提结构具有较高的存储访问带宽和较低的局部数据访问延迟.

关 键 词: 阵列处理器; 存储结构; 片上网络; 分布式存储; 统一编址
中图分类号: TN492 **文献标志码**: A

Design of Distributed Shared Memory Structure for Array Processor

SHAN Rui[1], SHEN Xu-bang[1], JIANG Lin[2], ZHU Yun[2], SONG Hui[2]
(1. School of Microelectronics, Xidian University, Xi'an 710071, China;
2. School of Electronic Engineering, Xi'an University of Posts and Telecommunication, Xi'an 710121, China)

Abstract: With the increasing of number of processors, the problem of memory wall was more severely. In order to alleviate this problem, two-level mixed interconnection network was proposed: fast crossbar for local data transfer and network on chip for long distance data communication. Meanwhile data transfer mechanism was designed to support unified addressing. Two memory architecture sizes were implemented on field rpogrammable gate array, and area, frequency and power consumption were evaluated. A mixed simulation testbench based on SystemC language was developed. The simulation results show that the designed architecture has higher memory access bandwidth and lower local accessing latency.

Key words: array processor; memory structure; network on chip; distributed memory; unified addressing

随着集成电路技术的发展,仍然沿用传统的处理器性能提升方法[1]:不断提高主频和开发指令集并行性,存在设计复杂度急剧增长和功耗过大的问题,同时主频的不断提高使得"存储墙"问题日益突出[2]. 为了适应集成电路工艺的发展,维持摩尔定律,单片上集成数十个甚至上百个简单处理器核成为计算机体系结构发展的趋势. 以阵列方式组织众多计算核心,具有可扩展性好、实现代价低的优点,是众核处理器发展的重要方向[3]. 随着工艺的进步,片上集成的处理器核数越来越多,并行处理过程对数据传输带宽的要求也越来越高,"存储墙"问题变得越为突出,访问带宽受限、访问延迟过大是制约

收稿日期: 2016-10-18
基金项目: 国家自然科学基金项目(61272120,61634004,61602377); 陕西省科技统筹计划项目(2016KTZDGY02-04-02); 陕西省教育厅专项科研计划项目(17JK0689)
作者简介: 山 蕊(1986—), 女, 博士生, E-mail: shanrui0112@163.com; 沈绪榜(1933—), 男, 院士, 博士生导师.

阵列处理器性能提升的主要瓶颈.

现有解决“存储墙”的普遍方法是引入了多级cache技术[4-7]. 然而,随着集成电路工艺的进步,片上集成更多的处理核将成为可能,仍然采用cache技术存在两方面的问题:一方面,增加的cache数量和容量,使得cache占用芯片面积急剧增长,目前cache面积占用比例达到25% ~50%,而且这个比例还在不断上升[8];另一方面,对于多媒体应用,存在大量的数据级并行运算,cache针对此类应用效率较差,不满足实时性要求,同时急剧增加的电路复杂度及过高的功耗也变得不可容忍.

为了解决“存储墙”问题,笔者提出了一种统一编址方式下的分布式共享存储结构(UaDSMS, unified addressing distributed shared memory structure),其优势在于:1) 采用统一编址方式,从逻辑上提供了一个片上大容量存储资源. 同时,采用软硬件协同技术,让软件编程人员直接管理、分配存储资源,免去了cache中用于保证数据一致性的复杂电路,降低了电路复杂度及功耗,同时实现了片上存储资源的高效利用. 2) 物理实现上,为了满足高带宽、低延时的实时性需求,将片上存储资源进行分块,在进行局部数据访问时,通过高速交换结构实现,满足簇内4×4处理单元(PE, process unit)阵列16路数据并行读/写访问;远程数据通信采用片上网络(NoC, network on chip)互连结构实现,满足簇间的低延时、高速率数据传送.

1 UaDSMS

UaDSMS的总体结构如图1所示,4×4的PE、高速交换开关、存储单元构成一个簇,其中存储单元由4×4个存储块(bank)构成. 簇内数据访问通过高速交换开关实现,支持16路并行读/写访问. 簇间数据通信通过低延时虚通道路由器构成的NoC实现. 数据访问过程:当PE需要访问存储器时,首先判定是否访问簇内存储单元,如果是,通过高速交换开关实现数据的并行访问;否则,即访问其他簇的存储单元. 利用高效的数据通信机制,将访问信息按照特定的格式要求发送到NoC,实现远程数据通信. 为了降低功耗和设计复杂度,只允许4×4PE阵列最右下角的PE通过NoC进行数据传送.

1.1 高速交换单元设计

基于邻接互连的轻核阵列处理器与众核处理器结构相比,具有执行操作级并行和数据级并行的特点,在进行算法实现时,尽可能地开发操作级并行和数据级并行,因此需要多个PE协同或者并行完成算法的运算. 在处理一些算法时,如图像算法中的滤波、卷积、直方图,视频算法中的运动估计、帧内预测,需要相邻的一块或者全部数据参与运算,这样就涉及访问相邻的存储bank. 如果采用传统的NoC互连[9],一方面,数据访问延时较大,一般读/写延时高达数十个时钟周期,这与操作级或是数据级运算的1 ~2个时钟周期相比,延时太大,难以满足应用的实时性需求;另一方面,所设计的存储结构针对轻核阵列处理器,单个PE的电路规模大概在2 000个逻辑单元,而单个4通道的虚通道路由器占用资源大概是单个PE的6倍,导致通信资源远远大于计算资源,资源利用率低下.

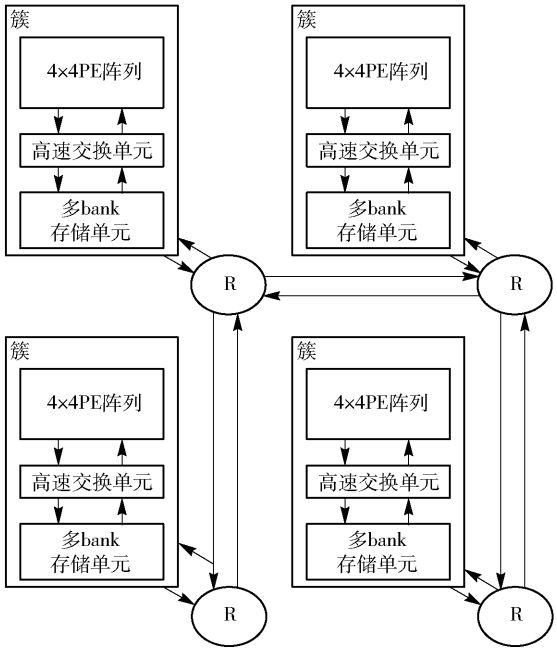

图1 UaDSMS总体结构

因此,提出了局部4×4 PE阵列内部数据访问通过高速交换单元实现,具体结构如图2所示.

每个PE与一个存储bank相对应,分别称为本地PE和本地存储bank,考虑到本地PE对本地存储bank的访问频率较高,因此在进行高速交换单元设计时,本地PE对本地存储bank的访问采用一条专用的硬件通道实现,进而实现本地PE对本地存储bank的无阻塞访问. 本地PE对簇内其他存储bank的访问则通过两级交换结构实现,如图3所示. 对所有位于同列的4个PE请求进行第1级仲裁,如果4个请求分别请求的是位于不同行的存储bank,则

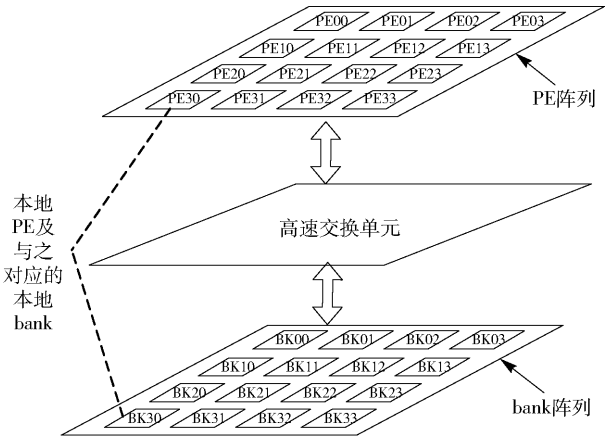

图 2　基于高速交换单元的簇内数据访问结构

同时输出 4 个请求给第 2 级仲裁器，如果存在多个请求同一行的存储 bank，则采用轮询策略进行仲裁. 通过第 1 级仲裁后，所有请求同一行的 4 个请求进行第 2 级仲裁，如果 4 个请求分别请求的是不同存储 bank，则同时输出 4 个请求给相应的存储 bank，如果多个请求同一存储 bank，则采用轮询策略进行仲裁.

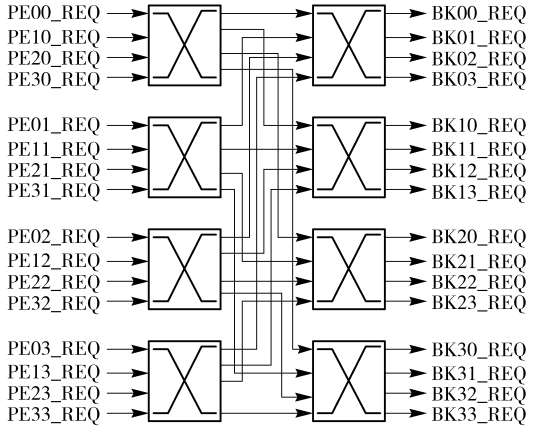

图 3　两级交换结构

1.2　低延时虚通道路由器设计

簇间数据通信若仍沿用高速交换单元实现，会导致电路延迟过大、设计复杂度急剧增长，且存在随着核数的增加不利于扩展的问题，因此，采用 NoC 结构实现簇间高速数据通信. NoC 的核心部件——NoC 路由器的性能将直接影响整个通信的效率. 为了降低通信延时，提出了一种低延时虚通道路由器结构，能够实现从输入到输出的单周期时延，设计结构如图 4 所示. 路由器由 5 个输入单元、交换分配模块、虚通道分配模块和输出交换开关组成. 每个输入单元包含 4 路虚通道，每个虚通道能够缓存 4 个片. 由于采用预先路由策略[10]，当头片到达本地路由器时，已经在上一级路由器中预先路由好了本级的输出端口，因此只需要在本地路由器中经过虚通道分配和交换分配即可路由输出. 为了降低分配带来的延时，采用快速分配策略，即当请求虚通道分配或交换分配时，根据上一周期预分配的结果，进行快速响应，产生响应信号，完成虚通道分配和交换分配，进而控制交换开关输出. 预分配结果将利用已经缓冲在虚通道中的数据片信息进行计算. 例如，进行虚通道分配时，根据当前的分配结果，以及已经缓冲在通路中的头片信息，计算下一时刻可能到达的头片分配请求，进行预分配，并将结果进行存储.

2　数据通信机制

进行存储访问时，由于采用统一编址方式，所以从用户编程角度看仍然采用 Load、Store 指令实现. 地址的编码格式从低位到高位依次为 12 位的存储 bank 地址，支持 16 KB 的块存储容量；4 位簇内地址，支持 4 × 4 存储 bank 阵列；8 位簇地址，支持 16 × 16 的阵列簇. 采用 Load、Store 指令访问簇内存储 bank 时，只能完成单个 32 位数据的读/写操作. 采用 Load、Store 指令访问簇外的存储 bank 时，为了提高数据的通信效率，设定每次的读/写操作完成 8 个 32 位的数据移动. Load 和 Store 指令的详细处理过程如图 5 所示.

在处理 Load 指令时，首先根据地址判断是否访问簇内存储 bank，如果是，经过高速交换结构完成数据读取；否则，需要通过 NoC 访问数据. 访问过程如下：

1）发起 LD 数据包，访问数据包格式如图 6(a)所示；

2）等待 RP 数据包的到来，RP 数据包格式如图 6(b)和(c)所示；

3）当 RP 数据包到达后，将访问数据存储到本地存储 bank 中.

在处理 Store 指令时，首先根据地址判断是否访问簇内存储 bank，如果是，经过高速交换结构完成数据写入；否则，通过 NoC 访问数据. 访问过程如下：

1）发起 ST 数据包，数据包格式如图 6(d)和(e)所示.

2）接收 PE 端接收到 ST 数据包后，根据地址信息，将数据存储到目标地址中.

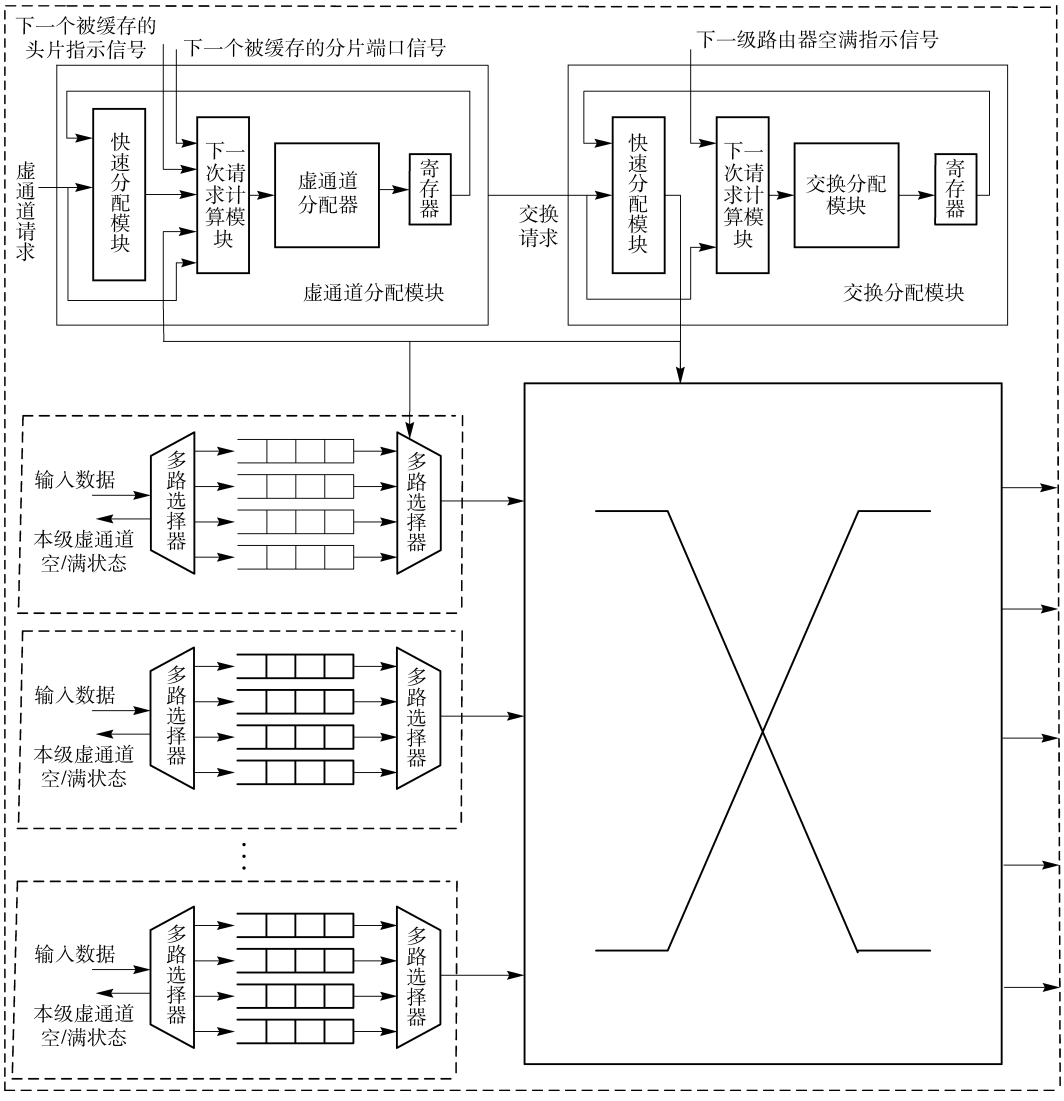

图4 低延时虚通道路由器结构

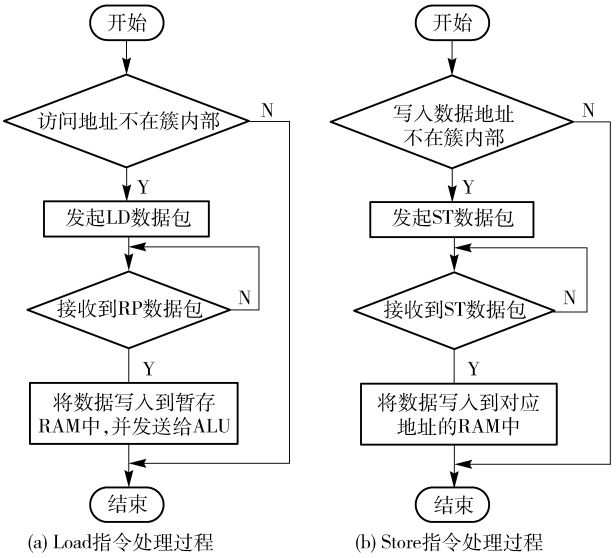

图5 Load 和 Store 指令处理过程

3 仿真与性能分析

为了验证 UaDSMS 功能的正确性及统计相关延迟信息,设计了阵列处理器存储器存取仿真平台(APMASB, array processor memory accessing simulation bench),其中包含一个简单的汇编器和仿真系统. 汇编器主要完成汇编语言指令到二进制的转换,产生文本文件,用于配置仿真系统. 仿真系统由 PE、UaDSMS、Xlinx 存储 IP 及统计计数器构成,其中 PE 采用 SystemC 语言完成行为级建模,UaDSMS 采用 Verilog HDL 语言完成电路设计. 使用仿真工具 Modelsim10.1d,并嵌入 C++调试器 gcc 进行 SystemC 和 Verilog 的混合仿真.

在 APMASB 中,分别对局部访问和远程访问的延迟信息进行了统计. 针对局部数据访问,统计了

输出端口号 3位	虚通道号 2位	包类型 2位	片类型 2位	源地址 8位	目标地址 8位	数据起始地址 12位	数据个数 3位

(a) LD数据包格式

输出端口号 3位	虚通道号 2位	包类型 2位	片类型 2位	源地址 8位	目标地址 8位	备用位 16位

(b) RP数据包格式——头片

输出端口号 3位	虚通道号 2位	包类型 2位	片类型 2位	访问数据 32位

(c) RP数据包格式——中间片/尾片

输出端口号 3位	虚通道号 2位	包类型 2位	片类型 2位	源地址 8位	目标地址 8位	数据起始地址 12位	数据个数 3位

(d) ST数据包格式——头片

输出端口号 3位	虚通道号 2位	包类型 2位	片类型 2位	访问数据 32位

(e) ST数据包格式——中间片/尾片

图 6　LD、RP、ST 数据包格式

不同冲突概率情况下的延迟信息及所对应的有效带宽,详见表 1. 有效带宽定义为单位时间传送的有效数据量,其计算方式为

$$\text{有效带宽} = \frac{\text{数据位宽} \times \text{时钟频率}}{\text{平均访问延迟}} \tag{1}$$

现场可编程门阵列(FPGA, field programmable gate array)综合结果显示,UaDSMS 工作频率达 100 MHz,预计 SMIC. 13 工艺下,时钟频率能够达到 200 MHz以上. 表 1 中有效带宽计算时,时钟频率初步采用 100 MHz. 由表 1 可以看出,UaDSMS 的局部数据访问最大带宽可以达到 51 Gbit/s,平均写延迟 1 个时钟周期,读延迟 2 个时钟周期,并且随着冲突概率的增加,平均访问延迟逐步增大,数据传送有效带宽随之下降. 但是,即使是在 100% 冲突的概率下,UaDSMS 的有效带宽也可以达到 5 Gbit/s.

表 1　UaDSMS 访问延迟

冲突概率/%	测试项		
	操作类型	平均访问延迟(时钟周期数)	有效带宽/(Gbit·s^{-1})
0	写操作	1	51
	读操作	2	25
25	写操作	1. 375	37
	读操作	2. 375	21
50	写操作	2. 750	18
	读操作	3. 750	13
75	写操作	5. 125	10
	读操作	6. 125	8
100	写操作	8. 500	6
	读操作	9. 500	5

针对远程数据访问,统计了规模为 4×4 个簇不同流量下的延迟信息. 图 7(a)(b)分别为无冲突情况下,针对不同源阵列簇(PEG, process element group)访问不同目的 PEG 的平均写/读延迟;图 7(c)(d)分别为不同冲突概率下,写/读不同目的 PEG 的平均延迟.

根据图 7 的统计结果,可以计算出不同冲突概率下的平均写/读访问延迟及有效带宽,如表 2 所示. 在无冲突访问下,UaDSMS 平均写延迟 11 个时钟周期,平均读延迟 20 个时钟周期,并且随着冲突概率的增加,写/读访问延迟也随之增加. 数据的有效传送带宽最高可以达到 1. 2 Gbit/s.

将表 2 中计算的结果与文献[5]和文献[11]进行对比,对比结果如表 3 所示. 局部数据访问方面,UaDSMS 在 100 MHz 的工作频率下,有效带宽达 6. 4 GB/s,高于文献[11]所提的基于 2×2 阵列 NoC 互连的三维堆叠结构,其在 1 GHz 工作频率下的带宽为 4. 53 GB/s. 局部数据读/写访问延迟方面,由于文献[5]内部采用了便签式存储,而 UaDSMS 中的本地存储请求无阻塞访问功能使得本地存储访问过程类似于便签式存储,因此 UaDSMS 和文献[5]都具有超低的读/写访问延迟,即读 2 个时钟周期,写 1 个时钟周期.

远程数据访问方面,UaDSMS 在 100 MHz 的工作频率下,有效带宽达 0. 15 Gbit/s. 未来采用先进工艺,UaDSMS 预期能够工作在 1 GHz,则有效带宽可达 1. 5 GB/s,略高于文献[11]的 0. 85 Gbit/s. 与文献[5]相比,UaDSMS 读/写延迟较接近文献[5],但是远程访问带宽远远低于文献[5]的 320 GB/s.

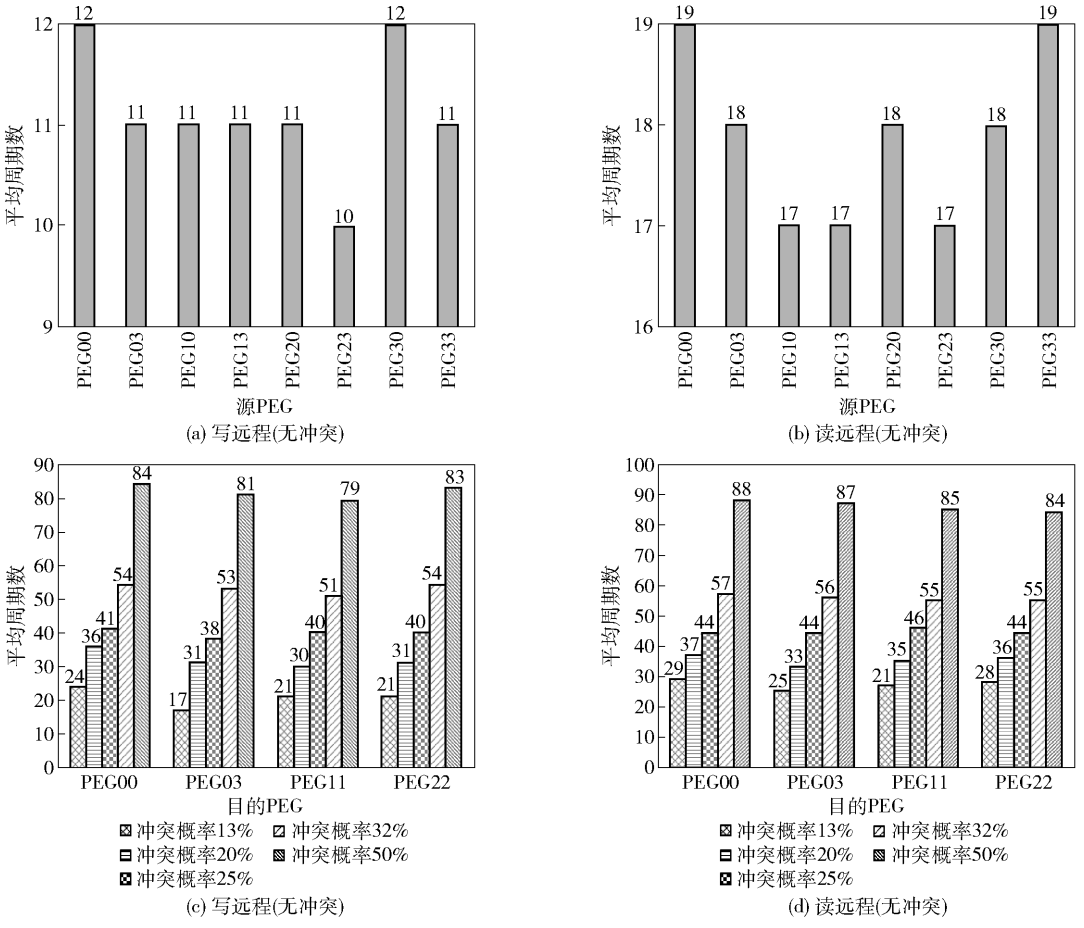

图7 远程数据访问的延迟信息

表2 不同概率冲突下的平均写/读访问延迟及带宽

远程访问	冲突率/%	延迟	带宽/(Gbit·s^{-1})
写	0	11	1.2
	13	21	0.6
	20	32	0.4
	25	40	0.3
	32	53	0.2
	50	82	0.1
读	0	18	0.7
	13	27	0.5
	20	35	0.4
	25	44	0.3
	32	56	0.2
	50	86	0.1

这主要是由于文献[5]中采用了一个96×96规模的全连接交换开关,有效地增加了带宽,并降低了延迟. 但是全连接交换开关随着核数的增加,硬件复杂度剧增,因此,UaDSMS在可扩展性方面具有一定的优势.

表3 UaDSMS与文献[5]和文献[11]对比

性能指标		工作频率/MHz	处理器核数	访问延迟(时钟周期数) 写	访问延迟(时钟周期数) 读	带宽/(GB·s^{-1})
UaDSMS	局部访问	100	256	1	2	6.40
	远程访问			11	18	0.15
文献[5]	SPM	100	80	1	2	—
	SRAM			10	20	320.00
文献[11]	直接访问	100		—	—	4.53
	远程访问			—	—	0.85

在Xilinx公司的V6550t FPGA开发板上实现了规模大小为4×4和8×8个簇的UaDSMS,两种规模的UaDSMS综合结果如表4所示. UaDSMS 4×4阵列簇的时钟频率为106 MHz,占用236 602逻辑单元;8×8阵列簇的时钟频率为111 MHz,占用508 656逻辑单元.

表 4　UaDSMS 综合结果

规模	逻辑单元	寄存器数量	时钟频率/MHz	功耗/W
4×4	236 602	101 815	106	6.419
8×8	508 656	121 150	111	–

4　结束语

笔者设计了一种 UaDSMS，该结构 4×4 簇内局部数据访问采用行列两级交换结构实现，支持本地 PE 对本地存储 bank 的无阻塞直接访问，支持簇内请求的无冲突并行访问；簇间远程数据通信采用 NoC 互连，利于扩展，适应集成电路工艺发展的趋势，同时设计了一种低延迟虚通道路由器，支持单周期延迟数据传送，以此降低全局数据通信延迟.

通过这种两级混合互连方式，将物理上分散的存储块高效互连起来，与大容量共享存储结构相比较，大幅度地提高了存储带宽；与 cache 结构相比，由于不需要硬件电路支持 cache 一致性，简化了电路结构，减小面积的同时，降低了功耗；与 NoC 互连结构相比，有效降低了局部数据访问延时，满足了应用的实时性要求.

参考文献：

[1] 魏少军，刘雷波，尹首一. 可重构计算处理器技术[J]. 中国科学：信息科学，2012(12)：1559-1576.
Wei Shaojun，Liu Leibo，Yin Shouyi. Key techniques of reconfigurable computing processor[J]. SCIENCE CHINA：Information Sciences，2012(12)：1559-1576.

[2] 李浩，谢伦国. 片上多处理器末级 Cache 优化技术研究[J]. 计算机研究与发展，2012，49(S1)：172-179.
Li Hao，Xie Lunguo. Research development of optimization technology on last level cache in chip multi-processors[J]. Journal of Computer Research and Development，2012，49(S1)：172-179.

[3] 石嵩，李宏亮，朱巍. 阵列众核处理器上的高效归并排序算法[J]. 计算机研究与发展，2016，53(2)：362-373.
Shi Song，Li Hongliang，Zhu Wei. Efficient merge sort algorithms on array-based manycore architectures[J]. Journal of Compute Research and Development，2016，53(2)：362-373.

[4] Berezecki M，Frachtenberg E，Paleczny M，et al. Power and performance evaluation of memcached on the TILEPro64 architecture[J]. Sustainable Computing Informatics & Systems，2012，2(2)：81-90.

[5] Hu Ziang，Cuvillo J D，Zhu Weirong，et al. Optimization of dense matrix multiplication on IBM cyclops-64：challenges and experiences[C]// Euro-Par 2006，Parallel Processing，12th International Euro-Par Conference. Dresden：[s. n.]，2006：134-144.

[6] 胡向东，杨剑新，朱英. 高性能多核处理器申威 1600[J]. 中国科学：信息科学，2015(4)：513-522.
Hu Xiangdong，Yang Jianxin，Zhu Ying. Shenwei-1600：a high-performance multi-core microprocessor[J]. SCIENCE CHINA：Information Sciences，2015(4)：513-522.

[7] 郑方，许勇，李宏亮，等. 一种面向高性能计算的自主众核处理器结构[J]. 中国科学：信息科学，2015(4)：523-534.
Zheng Fang，Xu Yong，Li Hongliang，et al. A homegrown many-core processor architecture for high-performance computing[J]. SCIENCE CHINA：Information Sciences，2015(4)：523-534.

[8] Banakar R，Steinke S，Lee B S，et al. Scratchpad memory：design alternative for cache on-chip memory in embedded systems[C]// Tenth International Symposium on Hardware/Software Codesign. Piscataway：IEEE，2002：73-78.

[9] 朱小虎，曹阳，王力纬. 多级拥塞控制的 NOC 路由算法[J]. 北京邮电大学学报，2007，30(5)：91-94.
Zhu Xiaohu，Cao Yang，Wang Liwei. A multilevel congestion control routing algorithm for network-on-chip[J]. Journal of Beijing University of Posts and Telecommunications，2007，30(5)：91-94.

[10] Mullins R，West A，Moore S. The design and implementation of a low-latency on-chip network[C]// 2006 Asia and South Pacific Conference on Design Automation. Piscataway：IEEE，2006：164-169.

[11] Loi I，Benini L. An efficient distributed memory interface for many-core platform with 3D stacked DRAM[C]// 2010 Design，Automation & Test in Europe Conference & Exhibition (DATE). Piscataway：IEEE，2010：99-104.