文章编号:1001-2486(2011)06-0031-05

# 具有高效混洗模式存储器的可编程混洗单元*

万江华,刘 胜,周 锋,王耀华,陈书明
(国防科技大学 计算机学院,湖南 长沙 410073)

**摘 要**:为向量DSP提出并设计了一种具有高效混洗模式存储器的可编程混洗单元,该结构能够使应用程序的混洗指令在执行时和通用寄存器或访存带宽等系统的关键资源分离,从而提高混洗单元的执行效率。采用区分不同混洗粒度和元素索引等方法对混洗模式所对应的开关矩阵进行了压缩,我们的存储效率高于当前其他方案。该混洗单元具有可编程的特点,应用程序所需要的混洗模式可以提前由程序员编写并通过DMA等途径加载。对该混洗单元进行了VLSI实现及性能评测,结果显示此方案能够在给系统带来0.6%的额外面积开销基础上使应用程序的性能提升7.4%~17.4%。

**关键词**:向量DSP;混洗单元;混洗模式存储器;混洗指令;存储效率
**中图分类号**:TP302 **文献标识码**:A

# A Programmable Shuffle Unit with the Efficient Shuffle Pattern Memory

*WAN Jiang-hua, LIU Sheng, ZHOU Feng, WANG Yao-hua, CHEN Shu-ming*
(College of Computer, National Univ. of Defense Technology, Changsha 410073, China)

**Abstract**: This paper presents a programmable shuffle unit with the efficient shuffle pattern memory for vector DSPs. The shuffle instructions can efficiently execute without occupying the system's key resource such as the general registers or the memory bandwidth. We compress the switch-matrix by differentiating the shuffle granularity and indexing the elements. The memory efficiency of our scheme is higher than the state-of-art methods. Programmers can design the shuffle patterns ahead of time and load them into the shuffle pattern memory by the DMA or other ways. Experimental results show that our scheme can reduce the execution cycles by 7.4% ~17.4% for the applications with the shuffle instruction requirement, at the cost of 0.6% additional chip area.

**Key words**: vector DSP; shuffle unit; shuffle pattern memory; shuffle instructions; memory efficiency

随着无线通讯、视频处理等算法的不断演进,向量技术在DSP中重新焕发了生机[1-4]。向量DSP采用多个并行的处理单元,也称向量处理单元(Vector Processing Elements, VPE)共用同一套取指、译码单元来高能效地开发嵌入式应用程序中的数据并行。VPE间寄存器级的数据交互(如交换、反序、广播等)一般由混洗单元负责,文献[5]通过在基于Altivec结构的处理器上映射GSM解码算法,发现混洗指令占到了30%以上。因而高效的混洗单元是影响向量处理器发挥其性能的重要因素之一。

传统支持多媒体扩展的处理器的指令集一般包含专用混洗指令(specific shuffle Ins.)和任意混洗指令(arbitrary shuffle Ins.)两类。其中专用混洗指令(如HP公司MAX-2指令集中的mix、check等指令[6];ARM公司NEON指令集中的VDUP、VSWP等指令[7])不需要明确指定其混洗模式(指令本身已经包含了具体的混洗模式),但使用范围有限,对应用中需要的特殊的混洗方式无能为力。任意混洗指令(如Intel公司IA-64中的mux指令[8]、Motorola公司Altivec指令集中的vperm指令[9]、ARM公司NEON指令集中的VTBL和VTBX指令等[7])通过混洗模式信息从源向量操作数选取不同的元素进行组合并形成新的向量可以进行任意混洗,但需要提前输入混洗模式信息至寄存器中[7-9]。由于任意混洗指令能够适应不同任务的混洗需求,甚至同一任务不同映射方式的混洗需求,因而是混洗单元的设计重

* 收稿日期:2011-09-20
基金项目:国家"核高基"重大专项资助(2009ZX01034-001-001-006);国家自然科学基金资助项目(61070036、61133007);国家863高技术资助项目(2009AA011704)
作者简介:万江华(1977—),男,助理研究员,博士。

点,本文也侧重于对任意混洗指令的高效设计、软硬件支持等问题的研究。

由于执行任意混洗指令时需要使用向量 load 指令提前从存储器中加载混洗模式信息至通用寄存器[7-9],这将会造成某些应用程序性能下降。如果某一个应用程序需要多种混洗模式,则需要花费多个通用寄存器来存放混洗模式信息,占用系统的通用寄存器资源;或者需要在程序的执行当中反复加载混洗模式,占用系统的访存带宽。在向量 DSP 中通用寄存器和访存带宽都是比较稀缺的资源,过度地占用将会影响程序的执行效率;程序员在预先 load 混洗模式时,必须考虑访存指令的节拍数提前插入 load 指令,增加了程序员的编程难度。文献[4]提出了一种 SRAM-based 交叉开关的方案,用 SRAM 单元取代了交叉开关结点的 flip-flop,可将混洗模式信息从寄存器和访存通路中释放出来,但是其混洗模式信息的位数和交叉开关的结点的数目是对应的,对于 $N\times N$ 的交叉开关,其混洗模式需要用 $N^2$ 位来表示,开销较大。目前向量 DSP 已经由传统的 8 路或 16 路逐渐向 32 路、64 路甚至更大的数据通路宽度扩展[10],在其上映射的程序的混洗需要更加复杂和多样,因而需要设计更加合理高效的混洗单元。

本文提出并设计了一种具有高效混洗模式存储器的可编程混洗单元,该结构使混洗指令在执行时和通用寄存器或访存带宽等系统的关键资源分离,从而提高混洗单元的执行效率。我们采用区分不同混洗粒度和元素索引等方法对混洗模式所对应的开关矩阵进行了压缩,与现有的方案相比,我们的存储效率最高。该混洗单元具有可编程的特点,应用程序所需要的混洗模式可以提前由程序员编写并通过 DMA 等途径加载。我们对该混洗单元进行了 VLSI 实现,并设计了与之对应的混洗指令。评测结果显示该方案花费的额外面积开销较小,能够方便快捷地执行混洗操作从而减少应用程序总的执行节拍数。

## 1　向量 DSP 中的数据混洗问题

混洗运算在数学上主要采用开关矩阵表示法[11],这种表示方法假设存在一个交叉开关网络,开关矩阵的内容与交叉开关网络结点的开启或关闭的状态对应。即开关矩阵 $\boldsymbol{P}$ 是一个 $M\times N$ 的方阵,矩阵的元素为 0 或 1,并且任意一行有且只有一个元素为 1,如下所示的 $\boldsymbol{P}_{4\times4}$ 是一个开关矩阵。

开关矩阵和列向量相乘就相当于对该列向量进行混洗,如下式所示。

$$\boldsymbol{P}_{4\times4}=\begin{bmatrix}0&0&1&0\\0&0&0&1\\1&0&0&0\\0&1&0&0\end{bmatrix}$$

$$\boldsymbol{P}_{4\times4}\cdot\boldsymbol{B}=\begin{bmatrix}0&0&1&0\\0&0&0&1\\1&0&0&0\\0&1&0&0\end{bmatrix}\cdot\begin{bmatrix}b_0\\b_1\\b_2\\b_3\end{bmatrix}=\begin{bmatrix}b_2\\b_3\\b_0\\b_1\end{bmatrix}$$

显然采用开关矩阵表示混洗模式,对于 $N\times N$ 的交叉开关需要 $N^2$ bit,这种表示方式的开销太大,在实现中一般采用元素索引法来进行压缩,对于 $\boldsymbol{P}_{4\times4}$ 可以压缩为“10,11,00,01”,每两位表示结果向量的每一个元素在源向量中的第几个位置,如“10”表示结果向量的第一个元素来自于源向量的第二个位置,采用元素索引法可以将开关矩阵压缩为 $N\times\log_2 N$ bits。

一些嵌入式应用程序的核心 kernel 如无线通信算法中 FFT/IFFT 变换、视频处理算法中的帧内预测等,在向量 DSP 运行时,混洗指令是制约其性能发挥的重要因素。我们通过这些算法进行分析,发现它们对混洗的需求具有如下的特性:①不同的应用需要预先确定的多个混洗模式;②不同的应用具有不同的混洗粒度需求;③一个应用的某一层循环往往对几个混洗模式反复使用;④某些应用不同循环层对混洗模式的使用具有相似性。这些特性是设计本文提出的混洗单元的主要依据。

## 2　具有高效混洗模式存储器的混洗单元

### 2.1　整体结构

本文提出的具有混洗模式存储器的混洗单元的整体结构如图 1 所示,主要由混洗模式存储器、交叉开关 crossbar 网络以及转换逻辑组成。其中混洗模式存储器存放应用程序需要的混洗模式,程序执行时,混洗指令中的立即数或混洗模式索引寄存器(Shuffle Pattern Index Register, SPIR)的内容将直接作为索引地址进行寻址读出本次混洗指令所需要的混洗信息,再经过转换逻辑处理之后驱动 crossbar 网络进行数据混洗。crossbar 网络采用基于选择器树的实现方式[12],具有两个向量数据输入端 Src1 和 Src2 和一个向量数据输出端 Dst。对于数据通路为 $N$(byte)的向量 DSP,该 crossbar 的规模为 $2N^2$,每一路的数据宽度为 8

bits。

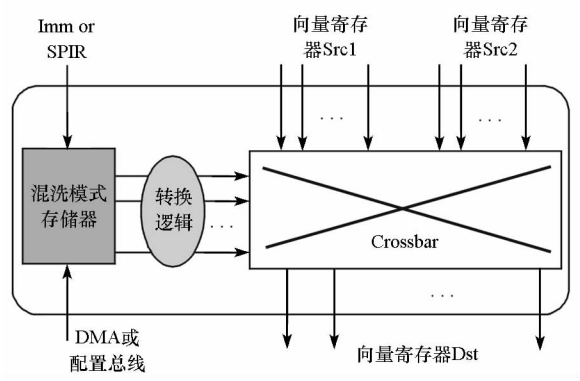


图 1　具有高效混洗模式存储器的混洗单元

Fig. 1　The shuffle unit with efficient shuffle pattern mode memory

## 2.2　混洗模式存储器的实现

混洗模式存储器由 4 个并行的 SRAM 组成,其深度由具体的应用需求决定,本文设置其为 32。对于混洗模式存储器的宽度我们进行了如下考虑:显然对于数据通路为 $N$(byte)的向量 DSP 其两输入一输出的混洗指令,按照元素索引法将会有效地压缩某次混洗所需要的信息位数,如表 1 所示。

**表 1　两种不同的混洗表示方法所需要的比特数**

Tab. 1　The needed bits of the two representation methods

| 混洗粒度 | 原始开关矩阵表示方法(bit) | 元素索引压缩法(bit) |
|---|---|---|
| 字节 | $2N^2$ | $N\cdot(\log_2 N+1)$ |
| 半字 | $N^2/2$ | $\dfrac{N\cdot\log_2 N}{2}$ |
| 字 | $N^2/4$ | $\dfrac{N\cdot(\log_2 N-1)}{4}$ |

在传统的采用通用寄存器保存混洗模式的方法中,由于受到通用向量寄存器编码的限制,即使是粒度较大的混洗模式也必须占用一个完整向量寄存器,这造成了存储效率的降低。在本文设计的混洗模式存储器中,地址编码会更加灵活,我们提出了区分不同混洗粒度的混洗模式压缩方法(在实际实现时区分了字节、半字和字三种粒度),考虑到这三种粒度的混洗模式能够共用同一个混洗模式存储器以及程序员配置的方便性,将构成混洗模式存储器的 SRAM 的宽度设置为 $N\cdot\lceil\dfrac{\log_2 N+1}{4}\rceil$,这样 4 个 SRAM 并行读出的数据能够进行一个字节粒度的混洗、2 个半字粒度的混洗或 4 个字粒度的混洗。图 2 为 $N=64$ 时混洗模式存储器的示意图,可以支持 128 种字粒度的混洗、64 种半字粒度的混洗或 32 种字节粒度的混洗。这种结构和 3.2 节的区分混洗粒度的混洗指令相对应,可以充分有效地节省混洗模式存储器的空间。

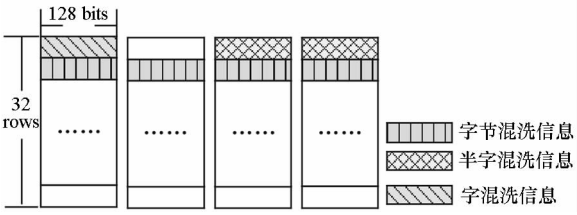


图 2　混洗模式存储器

Fig. 2　Shuffle pattern memory

混洗模式存储器采用两套不同的编址方式:全局编址和局部编址。其中全局编址采用字节编址方式,支持 DMA 或系统配置总线访问并提前进行配置,使我们的混洗单元具有可编程性,从而使应用程序执行当中可以直接执行混洗指令,不需要进行额外的配置。局部编址按照字粒度混洗模式的大小(128 bits)进行编址,供混洗指令中的立即数或 SPIR 进行索引,从而节省混洗指令的索引开销。

## 2.3　压缩混洗模式引入的时间开销分析

如果采用字节索引的方法可以直接驱动基于选择器树的 crossbar 网络,那么我们采用区分不同混洗粒度的混洗模式压缩方法的主要开销体现在字或半字粒度的混洗模式需要转换成字节混洗模式。转换逻辑主要负责这种转换。转换逻辑采用图 3 方法将字粒度混洗或半字粒度混洗的信息转换为字节粒度的选择信息,从而驱动 crossbar 网络进行数据重排。

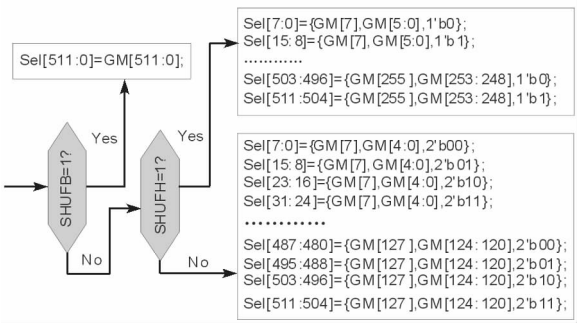


图 3　转换逻辑处理示意图

Fig. 3　The sketch map of the switch logic

图 3 中 GM 表示从混洗模式存储器中读出的数据,Sel 表示最终驱动 crossbar 的选择信息,GM 中每一个字节的最高位表示是否将目的寄存器的对应元素置为全零,因而在转换过程中保留。对于半字粒度的混洗信息扩展为字节粒度的混洗信息,只需在有效的选择字段后添加 1′b0 或 1′b1 即可。对于字粒度的混洗信息扩展为字节粒度的

混洗信息,只需在有效的选择字段后添加 2′b00, 2′b01, 2′b10 或 2′b11 即可。由于转换逻辑可以多个字节同时处理,每一个字节的转换开销比较小,因而对整个混洗单元的关键路径影响不大。

### 2.4 混洗指令

- SHUFBI imm, src0, src1, dst
- SHUFHI imm, src0, src1, dst
- SHUFWI imm, src0, src1, dst
- SHUFBR +/-imm, src0, src1, dst
- SHUFHR +/-imm, src0, src1, dst
- SHUFWR +/-imm, src0, src1, dst

我们提供上述 6 条混洗指令是与具有混洗模式存储器的混洗单元相配合。其中前 3 条指令采用立即数从混洗模式存储器选取不同混洗信息;后 3 条指令选用 SPIR 寄存器的内容从混洗模式存储器选取不同混洗信息,同时能够对 SPIR 寄存器进行小范围的立即数加减。这 6 条混洗指令均能够单周期完成,指令译码栈完成混洗信息的读取和转换,源操作数的读取;执行栈完成 crossbar 的选择和目的寄存器的写入。

显然,具有可编程性的混洗模式存储器能够满足应用对混洗需求多样性的需求;区分粒度的混洗指令满足了应用中不同混洗粒度的需求,同时能够节省混洗模式存储器的容量;支持立即数或寄存器寻址的混洗指令能够高效地解决应用中的某一层循环对几个混洗模式反复使用或不同循环层对混洗模式的使用具有相似性的问题。

## 3 分析与评测

### 3.1 硬件实现开销

采用硬件描述语言实现了一个规模为 128 × 64,数据通路宽度为 8 bits 的 crossbar 网络以及一个 4 路 32 深度,宽度为 128 bits 的混洗模式存储器。该混洗单元在国防科大自主知识产权的向量 DSP FT-Matrix 中得到了应用。

我们在 TSMC® 65nm 工艺下用 Synopsys® 公司的 Design Complier 对设计进行了综合,为了使混洗模式存储器的延时不对执行栈产生影响,没有采用通常的 Memory Compiler 工具产生存储体,而是采用多个寄存器进行组合。表 2 给出了综合的结果,其中混新模式存储器和转换逻辑占了混洗单元面积的 26.9%(等价于 32.8K gates),在全芯片布局布线之后发现其占了 FT-Matrix 全部面积(15 877 661 $\mu m^2$)的 0.6%,因而本文提出的方案引入的额外面积开销并不大。混洗模式存储器引入的额外路径延时主要存在于译码栈(而译码栈本身的路径延时比较宽松),并没有对执行栈(crossbar 选择)产生较大影响,目前关键路径延时为 1.59ns,满足目标芯片 500MHz 的频率需求。

表 2 混洗单元的综合结果

Tab. 2 The synthesis results of the proposed shuffle unit

| | 关键路径延时/ms | 面积/$\mu m^2$ | 百分比 |
|---|---|---|---|
| 交叉开关 | 1.59 | 257 697 | 73.1% |
| 混洗模式存储器及转换逻辑 | | 94 629 | 26.9% |
| total | | 352 326 | 100% |

### 3.2 混洗模式表示效率

我们将表示一种混洗模式按照元素索引方法理论上需要的位数和实际实现时表示一种混洗模式所花费的位数的比值定义为混洗模式表示效率,分别比较了 NEON 指令集中的 VTBL、VTBX 指令[7],Altivec 指令集中的 Vperm[9],AnySP 处理器中的混洗单元及指令[4]以及本文提出的混洗单元的混洗模式表示效率,如表 3 所示。本文提出的混洗单元在字节、半字、字粒度下混洗模式表示效率分别为 1.00、0.88 和 0.75,均为最高,这有利于高效地利用混洗模式存储器,方便程序员配置混洗指令。

表 3 不同指令集/方法的混洗模式表示效率

Tab. 3 The shuffle pattern representation efficiency

| 指令集/方法 | 混洗粒度 | 理论上需要的位数 | 实际使用位数 | 混洗模式表示效率 |
|---|---|---|---|---|
| NEON[7] | 字节 | 48 | 64 | 0.75 |
| | 半字 | 20 | 64 | 0.31 |
| | 字 | 16 | 64 | 0.25 |
| Altive[9] | 字节 | 96 | 128 | 0.75 |
| | 半字 | 40 | 128 | 0.31 |
| | 字 | 32 | 128 | 0.25 |
| Anysp[4] | 字节 | 128 | 2048 | 0.06 |
| | 半字 | 96 | 2048 | 0.05 |
| | 字 | 40 | 2048 | 0.02 |
| 本文 | 字节 | 512 | 512 | 1.00 |
| | 半字 | 224 | 256 | 0.88 |
| | 字 | 96 | 128 | 0.75 |

### 3.3 应用程序分析

我们在 FT-Matrix 的体系结构模拟器中构建了不具有混洗模式存储器的混洗单元(SU-NM)以及本文提出的混洗单元结构和指令(SU-M),选取 1024 点基 2/基 4 定点复数的 FFT,H.264 编码算法中的 Intra Prediction(按照 4 × 4 luma 块处理

9 种不同的模式)这 3 种应用程序进行对比模拟,结果如图 4 所示。我们发现采用本文提出的具有高效混洗模式存储器的混洗单元相对于传统的方法能够使上述 3 个程序的执行周期数降低 14.3%、17.4%和 7.4%.

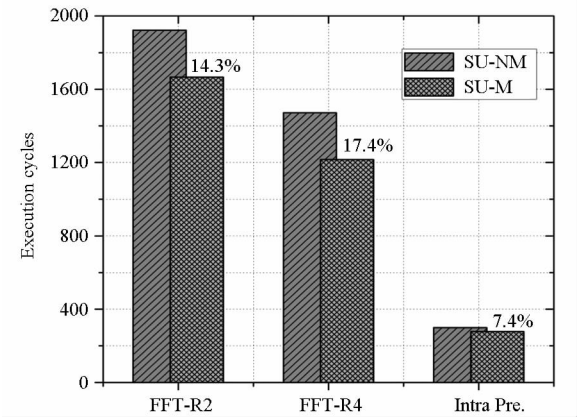


图 4　3 个应用程序在基于不同混洗单元的模拟器中的执行效果

Fig. 4　The effect of the 2 shuffle units under 3 applications

## 4　小 结

针对传统混洗单元中的任意混洗指令需要采用 load 指令加载混洗模式或采用通用寄存器保存混洗模式的弊端,本文提出了一种具有混洗模式存储器的混洗单元及其对应的混洗指令,执行混洗操作时可以直接读取混洗模式存储器的混洗模式信息并进行混洗,不需要占用通用寄存器和访存带宽等关键资源。分析和评测显示该混洗单元不会对关键路径延时产生影响,额外引入的面积在目标芯片中所占份额不大,但能够有效地减少应用程序总的执行周期数。

## 参 考 文 献:

[1] Van Berkel K, Heinle F, Meuwissen P P E et al. Vector Processing as an Enabler for Software Defined Radio in Handheld Devices [J]. EURASIP Journal on Applied Signal Processing, 2005 (16):2613 - 2625.

[2] Limberg T, Winter M, Bimberg M, et al. A Heterogeneous MPSoC with Hardware Supported Dynamic Task Scheduling for Software Defined Radio [C]//Proceedings of the Design Automation conference, 2009.

[3] Woh M, Lin Y, Seo S et al. From SODA to Scotch: The Evolution of a Wireless Baseband Processor [C]//Proceedings of 41st IEEE International Symposium on Microarchitecture, Lake Como, Italy, 2008: 8 - 12.

[4] Woh M, Seo S, Mahlkes, et al. AnySP: Anytime Anywhere Anyway Signal Processing [C] //Proceedings of ISCA, 2009: 128 - 139.

[5] Freescale Semiconductor. Altivec Engine Benchmarks [EB/OL]. [2011 - 09 - 20]. http://www.freescale.com/webapp/sps/site/overview.jsp? nodeId = 0162468rH3bTdGmKqW5Nf2, 2006.

[6] Lee R. Subword Parallelism with MAX-2 [J]. IEEE Micro, 1996,16(4):51 - 59.

[7] ARM. Cortex-A8 Technical Reference Manual [EB/OL]. [2011 - 09 - 20]. http://infocenter.arm.com 2008, ARM DDI 0344H.

[8] Intel Corporation, IA-64 Application Developer's Architecture Guide[EB/OL]. [2011 - 09 - 20]. http://developer.intel.com/design/ia64,1999.

[9] Motorola. AltiVec Extension to PowerPC Instruction Set Architecture Specification[EB/OL]. [2011 - 09 - 20]. http://www.motorola.com/AltiVec,1998.

[10] Woh M, Lin Y, Seo S, et al. Analyzing the Scalability of SIMD for the Next Generation Software Defined Radio [C]// Proceedings of the IEEE International Conference on Acoustics, Speech and Signal 2008:5388 - 5391.

[11] Moon T K, Stirling W C. Mathematical Methods and Algorithms for Signal Processing [J]. Upper Saddle River, NJ, U.S.A.: Prentice Hall, Inc., 2000.

[12] Cooperman M, Andrade P. CMOS Gigabit-per-second Switching [C]//Proceedings of the IEEE Journal of Solid-state Circuits, 1993, 28(6): 631 - 639.