文章编号: 1007 - 4252(2021)06 - 0006 - 519

基于相变存储查找表的低功耗存算一体计算方案

阿西木·约麦尔[1,2],陈小刚[1,*],陆俊杰[1],钱星宇[1],李顺芬[1],宋志棠[1]
(1. 中国科学院上海微系统与信息技术研究所 信息功能材料国家重点实验室,上海 200050; 2. 上海科技大学 物质学院,上海 201210)

摘要: 本文介绍一种在相变存储器(PCM)内基于查表方式实现的存算一体计算方案。该方案硬件系统主要由传统的控制和计算模块、连接模块以及 PCM 模块组成。在同一 PCM 阵列中,映射有运算算子结果的查找表(LUT)和 LUT 的地址索引信息。并且用此方案在快速傅里叶变换(FFT)的计算上验证及评估了方案的可行性。与传统的计算架构相比,在计算相同量级的 FFT 运算时,新提出的计算架构在功耗上面具有很大的优势,并且需要的硬件资源也较少。

关键词: 存算一体; 查找表; 相变存储器; FFT
中图分类号: TP333 **文献标志码:** A

A low power In-memory computing scheme based on lookup table with PCM

Aximu Yuemaier[1,2],CHEN Xiao-gang[1,*],LU Jun-jie[1],QIAN Xin-yu[1],LI Shun-feng[1],SONG Zhi-tang[1]
(1. State Key Laboratory of Functional Materials for Informatics, Shanghai Institute of Micro-system and Information Technology, Chinese Academy of Sciences, Shanghai 200050, China; 2. School of Physical Science and Technology, ShanghaiTech University, Shanghai 201210, China)

Abstract: This paper introduces an In-memory computing scheme based on table lookup in PCM. The hardware system of this scheme consists of control module, connection module and PCM module. In the same PCM array, the lookup table (LUT) with operator results and the address index information of the LUT are mapped. With this scheme, the feasibility of the scheme is verified and evaluated by the calculation of Fast Fourier transform (FFT). Compared with traditional computing architectures, the new computing architecture has great advantages in power consumption and requires less hardware resources when calculating operations of the same order of magnitude as FFT.

Key words: In-Memory computing; Lookup table; Phase change memory; FFT;

收稿日期: 2021-11-24; **修订日期:** 2021-12-08
基金项目: 国家重点研发计划项目(No. 2017YFA0206101; No. 2018YFB0407500).
作者简介: 阿西木·约麦尔(1996—),男,硕士研究生,主要研究方向为相变存储器应用(E-mail: aximuyme@ shanghaitech.edu.cn).
通信作者: 陈小刚(1976—),男,博士,副研究员,主要研究方向为新存储架构、内外存融合、存储内计算和复杂系统等(E-mail: chenxg@ mail.sim.ac.cn).

0 引言

近年来，随着应用数据处理需求的激增，内存与处理器之间的传输带宽限制成为数据密集型应用的主要瓶颈。冯诺依曼架构的计算和存储分离，成为了制约系统能效和计算延时的一大阻碍[1]。由此发展出了存储单元和逻辑单元有机整合的存算一体化架构。其中比较流行的方案就是基于新型非易失存储器的存内计算(In-memory-computing，IMC)，也成为了解决存储墙问题最理想的方案[2]。

在基于非易失内存的存算一体方案中，内存单元直接参与计算，主要是利用电阻和电流电压的物理关系来表达运算过程。对于这类存算一体化的研究主要分为三种：一种是在内存阵列内用基尔霍夫定律实现向量乘矩阵操作[3]，另一种是在内存内实现逻辑操作的内存计算[4]，还有一种是基于搜索查表操作的存内计算[5]。这些 IMC 方案相较于传统计算架构都表现出了很高的性能和较低的能耗优势。但是在前两种方案中用存储器的物理结构和特性来支持的算子较少，设计不灵活，应用场景相对单一。而查表方案的应用相对自由，可扩展性好。

查找表(LUT)是用简单的查询操作替换运行时计算的数组。LUT 广泛应用于 FPGA 等可编程器件上，满足了硬件电路反复可配置的要求。若用查表替代算术计算，最大的缺陷就是 LUT 的大小会指数倍地增长。对此，有的方案就用 LUT 缓存出现频率高的计算值，达到降低功耗减少频繁计算开销的效果。比如像 Imani 等[6-7]的研究就是通过使用由关联内存组成的 LUT 来节省能源。关联内存在表中搜索、输入最佳匹配的值，并把查表结果作为给定输入的值返回[8]。在这种方案中，提高搜索命中率是需要解决的问题，而且该方案需要多次写入到关联内存中，这对于擦写寿命有限的非易失内存是不可接受的。

本文探讨基于相变存储搜索查表方式的低功耗存算一体计算方案。采用降低数据精度的方法，使得 LUT 的大小尽可能地减小，并且在应用精度允许的条件下，用查表的方式实现计算操作。由于计算数据跟地址线有很强的映射关系，LUT 里存了几乎所有的真值表内容，因此不需要搜索这一步，这样避免了对 PCM 的反复读写以及搜索的开销。对于精度不敏感、数据局部性高的应用，在功耗上面有很高的优势。本文第一节介绍新型基于查表方案的 IMC 方案，并且介绍快速傅里叶变换(FFT)的实现方式；第二节介绍方案中实验环境的搭建和实验评估结果；最后一节总结了全文并得出结论。

1 基于 LUT 的内存计算

以下是基于 PCM 的存内计算方案的机理介绍。首先对架构进行了概述，之后介绍了降低浮点数精度的方法，最后在此架构的基础上展示了 FFT 算法的计算。

1.1 基于 LUT 的 IMC 概述

本文提出的 IMC 架构主要由四部分组成，分别是控制模块(CE)、计算模块、连接模块和 PCM 阵列，如图 1 所示。CE 的作用是对数据进行调度和分发，PCM 通过连接模块与控制中心传输数据。在 PCM 里主要划分了两块区域，分别是算子的低精度 LUT 段和 LUT 辅助查找的索引信息 TI。其中 LUT 和 TI 都是由 CE 进行初始化，在计算阶段又由 CE 依次读取 TI 和 LUT 的内容来得到相应的计算结果。

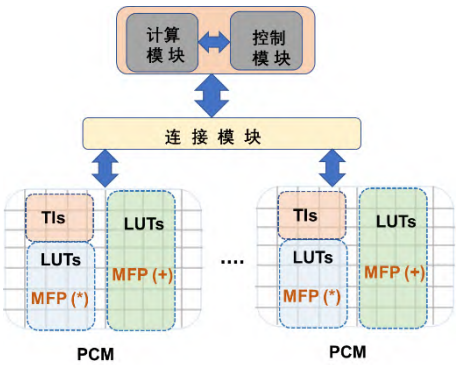

图 1 基于 PCM 查找表的存算一体架构

Fig. 1 Proposed IMC architecture with PCM crossbar

方案充分利用了 PCM 非易失、高存储容量的特点，只通过读操作实现计算，避免了 PCM 写入次数有限的局限。但是对于高位宽的数字计算，所需要的 LUT 大小较大。对于 N 比特的算术操作，所需要的表格大小为 2^N。因此，减小 LUT 的大小成了该架构中需首要解决的问题。

1.2 低精度浮点数计算

通过调整浮点数位宽来降低计算精度，不光可

以达到减小 LUT 大小的目的,还能够有效减小能耗[9]。在传统的计算架构中,实数用 IEEE-754 浮点数表示方法来表示。32 位浮点数或 64 位双精度浮点数是常用的数据类型。在应用精度可接受的范围内,减小浮点数位宽到 16 位甚至 8 位左右,可以有效减小 LUT 的大小。但是低位宽 IEEE-754 浮点数表示方法容易出现溢出,且能表示的数据范围有限。比如如下图 2 所示,在 8 位宽浮点数表示方法中,数值能表示的范围为(-15.5, +15.5),因此对应用的计算类型有一定的限制。还可以用 Gustafson[10] 等人提出的 Posit 数据表示方法来替换 IEEE-754 浮点数。这种数据表示方法不仅可以避免溢出出现,而且还对硬件友好。两种浮点数表示方法的 8 位宽表示形式如下图 2 所示。

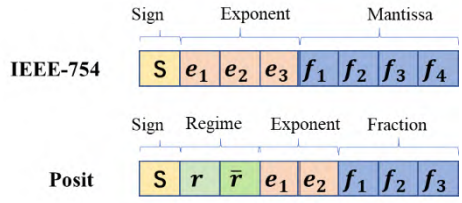

图 2　IEEE-754 和 Posit 类型的 8 位宽浮点数表示形式

Fig. 2 8-bit binary format for the IEEE 754 standard and Posit number system

寻找合适的数据映射方式,对计算的效率及可靠性尤其重要。因此,在下面一个小节将讨论数据的映射方式以及具体的查表方案。并且用 FFT 来验证该基于近似查表的计算方案。

1.3　基于查表的 FFT 计算

FFT 是计算离散傅里叶变换(DFT)的有效算法,也是数字信号处理(Digital Signal Processing, DSP)系统中基本的计算方法。其中基-2 Cooley-Tukey 算法是经典且最简单的 FFT 算法[11]。在本文中我们用查表方案替换传统 FFT 计算方式,使得浮点数的乘加运算不需要复杂计算过程就能得到结果。

长度为 N 的 x(n) 序列 DFT 变换为:

$$X(k)=\sum_{n=0}^{N-1}x(n)\,Wnk_N,\quad n=0,1,\cdots,N-1$$

其中 $X(k)$ 是 $x(n)$ 的频域表示,$k=0,1,\cdots,N-1$。$Wnk_N=\exp(-j2\pi k/N)$ 是旋转因子。

1.3.1 FFT 计算查表流程

对于 FFT 的查表计算过程主要由两个部分组成,分别是计算前的预处理部分和查表计算的部分。在计算之前的预处理过程中,首先搜索读取 1.1 节中提到的 TI 的内容,获取 LUT 的分布地址给后续查表计算过程使用; 其次在此阶段会根据计算算子需求更新 LUT 和 TI 里的内容。在查表计算过程当中,如图 3 所示为串行计算流程,首先根据基-2 Cooley-Tukey 算法进行数据的调度,然后根据预处理阶段获取的 LUT 地址信息来进行 PCM 访问地址的映射。最后判断是否完成所有的输入点数,若没有完成则返回到数据调度部分计算。

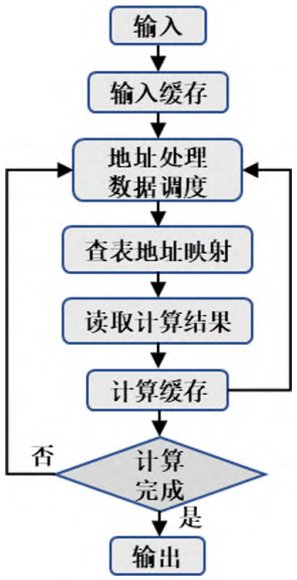

图 3　FFT 查表计算流程

Fig. 3 LUT based FFT computing work flow

蝶形运算是 FFT 每一级运算的基本组成部分。图 4(a)表示了由两个输入、两个输出和一个旋转因子组成的蝶形单元,其中所有数字都是复数。在蝶形运算中,有三种算术操作,分别是乘和加减运算。在查表计算中,CE 接收数据后会根据算法去调度数据进行搜索操作。比如如下图 4(b)展示的计算中,首先查表得到 b 和旋转因子 w 的乘积,然后再一次查表获得加和减的结果。在 PCM 中,会在预处理阶段初始化存储蝶形运算所需的算术运算的查找表。

1.3.2 数据映射方法

在算术运算中,一般都有数据的输入、处理以及输出三个步骤。在查表方案中,这三部分需要映射到基于查表的 IMC 架构中。其中输入的数据映射到地址线中,如下图 5 所示为两输入算术运算在 22

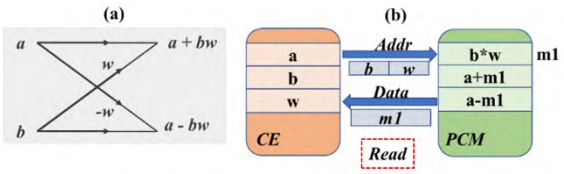

图 4 (a) 蝶形运算;(b) 蝶形运算的查表方式

Fig.4 (a) Simple butterfly operation; (b) LUT searching method for butterfly operation.

位地址线上的映射。其中操作数 A 和 B 分别是需要计算的两个 8 位宽的数据,还有 4 位是 LUT 的基址(BA)信息部分。对于数据的处理部分就是在 PCM 中的查表读取操作。比如在 FFT 运算中的乘和加减运算的结果会存储在数据输入映射到的地址里面。相应的算术运算的输出,则对应着数据线里读出的内容。

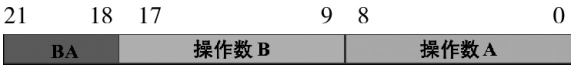

图 5 两输入算术运算映射到 22 位宽的 PCM 地址线

Fig.5 Two-input arithmetic operations map to a 22-bit width PCM address line.

在查表计算时,PCM 内 LUT 的布局索引信息会提前存在图 1 里展示的 TI 段里。因此应先读取 TI 里面的内容,以便获取不同算术运算 LUT 的基址 BA。CE 模块不仅要确保生成合适的浮点数运算位宽,也要保证在计算过程中数据位宽的一致性。选取合适的位宽,不仅影响 LUT 的大小,还会影响计算结果的准确度。

1.3.3 LUT 大小的缩减

查表计算必须要解决的问题是 LUT 的大小。如下表 1 所示为不同浮点数位宽时单个算子的 LUT 资源的大小。数据位宽越小,LUT 的资源相应的指数次越低。在满足应用的精度需求时数据位宽越小越好。

表 1 单个算子在不同浮点数位宽下的 LUT 资源

Table 1 LUT resources of a single operator

浮点位宽	16	12	10	8	6
LUT(KB)	10E+5	1000	64	16	1

在生成表格时可以用一些技巧使得 LUT 的大小进一步缩减。比如算术运算的两个符号位缩减成一个判断位。在乘法运算时,结果的符号位就可用异或逻辑来得到。这时数据映射到地址线时就可以减少一位,相应地生成表格时也会减少一个指数量级。

2 实验与评估

为了验证基于 LUT 的 IMC 的可行性并获取性能评估,我们构建了硬件验证系统板。如下图 6 所示,硬件验证系统板由 Xilinx Zynq-7020 开发板和 PCM 芯片扩展板组成。实验所用的 PCM 芯片为课题组自研的 64 M 容量的 PCM 芯片,接口为 SRAM 类型,有 16 位数据线和 22 位地址线。其中 FPGA 提供数据流控制和计算结果的读取和 LUT 的写入。而 PCM 作为查表方案的主体,用于查表方式的映射和 LUT 及 TI 的存储及更新。下面用 FFT 计算来验证及评估近似查找表计算的性能与功耗。

图 6 基于 LUT 的 IMC 验证硬件系统板

Fig.6 Hardware verification system board

为了验证不同精度对结果的影响,FFT 的输入采用基准 AxBench [12] 的数据集。用 IEEE-754 和 Posit 数字表示方法,仿真了从 4 位宽到 16 位宽浮点数的 FFT 计算。此外,分别计算信噪比(SNR)和峰值信噪比(PSNR)作为本仿真的 QoR。为了对比,我们以 IEEE-754 形式的 32 位浮点数的 FFT 作为标准准确值。图 7 所示为以不同位宽度显示 2048 点 FFT 的 SNR 和 PSNR。从图中可以看出在 8 位宽的浮点数计算 SNR 可达到 60 dB,可以符合高容错率应用的精度要求。

最后,我们在表 2 中列出了用专用 FFT 处理器和 Zynq 7020 计算 FFT 跟本文提出的方案计算 FFT 的性能对比。数据类型都采用浮点数,其中查表方案为 8 位浮点数,其他两种都是 32 位浮点数。硬件资源方面,由于 FFT 中的主要算术操作是乘法和加

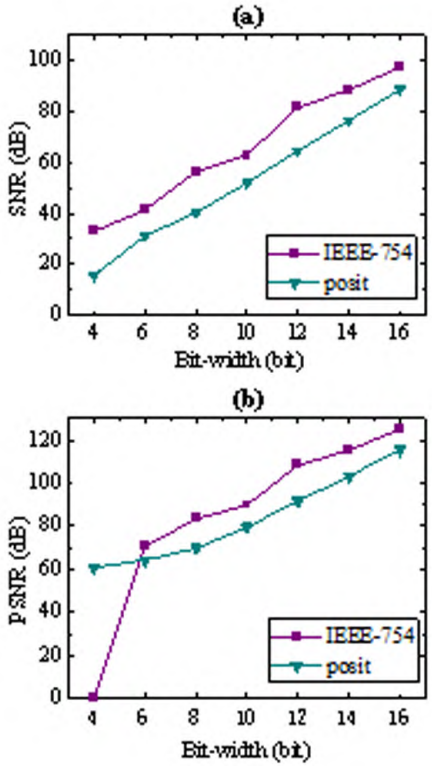

图 7　IEEE-754 和 Posit 形式的 2048 点 FFT,(a) SNR vs.数据位宽; (b) PSNR vs.数据位宽.

Fig.7 2048-point FFT for IEEE-754 and Posit floating point number system, (a) SNR vs. data bit width; (b) PSNR vs data bit width.

减,1 024 点 FFT 的 LUT 大小为 33 KB。在 FPGA 计算中,FFT 计算需要 8 块 DSP48 资源和 10 块 BRAM。吞吐率方面,查表方案中每秒能处理的数据量为 1.25 M,相较于 Paul [13] 的研究有 6 倍左右的优势,但不及并行运算能力强的 FPGA 的吞吐率。对于 PCM 的功耗,表 3 为 PCM 读数据时的电学参数。在 3.3 V 读取电压下,功耗可达到 0.66 nJ/KB。基于 LUT 的计算,主要能耗为内存读数,功耗为 0.528 mW。但是,在基于 LUT 的计算中,当 LUT 只使用一个 PCM 模块时,会比同一频率的 FPGA 计算有更长的延迟。为了提高并行性,可同时从多个 PCM 里并行读取。例如,在 FFT 蝶形运算中的复数乘法中,可以并行从两个 PCM 中读取实部和虚部的乘法,从而将计算时间缩减到一半。

表 2　提出的方案和其他设计的性能比较

Table 2 Performance comparisons between the proposed architecture and previous designs

	LUT	Paul's [13]	Zynq 7020
FFT 点数	1024	1024	1024
数据类型	8/Float	32/Float	32/Float
硬件资源	PCM: 33 KB	Gate: 307 KB	DSP48: 8 BRAM: 10/36 KB
频率/MHz	50	50	50
吞吐率 MS/s	1.25	0.202	6.11
功率/mW	0.528	68	—

表 3　PCM 电学参数

Table 3 Electrical parameters for PCM data reading

T_{read}	I_{read}	U_{read}	E_{read}
20 ns	10 μA	3.3 V	0.66 nJ/KB

3　结论

基于相变存储查找表的低功耗存算一体计算方案充分利用了 PCM 高的读性能和非易事性等优点。为了降低 LUT 的大小,采用降低浮点数位宽的方法。FFT 计算在 8 位浮点数左右都能得到预期的精度,而且单个算子 LUT 的大小可以降到 16 KB。并且在相同的频率下,查表方式的功耗比用专用处理器的功耗小 3 个数量级,适合用于低功耗需求的计算设备。该方案在计算精度不敏感、算术计算较为复杂的近似计算系统中具有很大的优势。

参考文献:

[1] Mutlu O, Ghose S, Gómez-Luna J, et al. Processing data where it makes sense: Enabling in-memory computation [J]. Microprocessors and Microsystems, 2019, 67: 28-41.

[2] Zidan M A, Strachan J P, Lu W D. The future of electronics based on memristive systems [J]. Nature Electronics, 2018, 1(1): 22-29.

[3] Shafiee A, Nag A, Muralimanohar N, et al. ISAAC: A Convolutional Neural Network Accelerator with In-Situ Analog Arithmetic in Crossbars [C]. //43rd ACM/IEEE Annual International Symposium on Computer Architecture (ISCA). Seoul, 2016: 14-26.

[4] Imani M, Kim Y, Rosing T, et al. MPIM: Multi-Purpose In-Memory Processing Using Configurable Resistive Memory [C]. //2017 22nd Asia and South Pacific Design Automation Conference. Tokyo, Japan, 2017: 757-763.

[5] Imani M, Gupta S, Arredondo A, et al. Efficient Query Processing in Crossbar Memory [M]. 2017.

[6] Imani M, Gupta S, Sharma S, et al. NVQuery: Efficient Query Processing in Nonvolatile Memory [J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2019, 38(4): 628-639.

[7] Imani M, Kim Y, Rahimi A, et al. Acam [C]. //Proceedings of the 2016 International Symposium on Low Power Electronics and Design. 2016: 162-167.

[8] Imani M, Peroni D, Rosing T. Program Acceleration Using Nearest Distance Associative Search [C]. //2018 19th International Symposium on Quality Electronic Design (Isqed). Santa Clara, CA, 2018: 43-48.

[9] Fuketa H, Uchiyama K. Edge Artificial Intelligence Chips for the Cyberphysical Systems Era [J]. Computer, 2021, 54(1): 84-88.

[10] Gustafson J L, Yonemoto I. Beating Floating Point at its Own Game: Posit Arithmetic [J]. Supercomputing Frontiers and Innovations, 2017.

[11] Cooley J W, Tukey J W. An Algorithm For Machine Calculation Of Complex Fourier Series [J]. Mathematics of Computation, 1965, 19(90): 297-&.

[12] Yazdanbakhsh A, Mahajan D, Esmaeilzadeh H, et al. AxBench: A Multiplatform Benchmark Suite for Approximate Computing [J]. IEEE Design & Test, 2017, 34(2): 60-68.

[13] Paul A S B, Raju S, Janakiraman R. Low power reconfigurable FP-FFT core with an array of folded DA butterflies [J]. EURASIP Journal on Advances in Signal Processing, 2014: 144.