arXiv:2304.03320v1 [cs.AR] 6 Apr 2023

CamJ：实现面向传感器内视觉计算的系统级能耗建模与体系结构探索

Tianrui Ma*圣路易斯华盛顿大学 美国密苏里州圣路易斯
tianrui.ma@wustl.edu

Yu Feng*罗切斯特大学 美国纽约州罗切斯特
yfeng28@ur.rochester.edu

Xuan Zhang†圣路易斯华盛顿大学 美国密苏里州圣路易斯
xuan.zhang@wustl.edu

Yuhao Zhu†罗切斯特大学 美国纽约州罗切斯特
yzhu@rochester.edu

摘要

CMOS 图像传感器（CIS）是新兴视觉计算应用的基础。传统 CIS 纯粹是用于捕获图像的成像设备，而越来越多的 CIS 开始集成处理能力，例如深度神经网络（DNN）。计算型 CIS 拓展了架构设计空间，但迄今尚无全面的能耗模型。本文提出 CamJ，这是一个详细的能耗建模框架，可提供计算型 CIS 的组件级能耗细分，并通过九款近期 CIS 芯片进行验证。我们使用CamJ 展示了三个用例，以探索架构权衡，包括 CIS 内外计算、二维与三维堆叠式 CIS 设计，以及 CIS 内部的模拟与数字处理。CamJ 的代码可在以下地址获取：
https://github.com/horizon-research/CamJ.

CCS 概念

• **硬件** →**功率和能量**； • **计算机系统组- 织** → **模拟计算机**；*Sensors*和执行器； • **通用和参考** →估计。

关键词

传感器内计算、能耗建模、模拟建模

ACM 参考格式：
Tianrui Ma、Yu Feng、Xuan Zhang 和 Yuhao Zhu。2023年。CamJ：实现传感器内视觉计算的系统级能耗建模与体系结构探索。载于第*50*届年度国际计算机体系结构研讨会论文集（*ISCA '23*），*2023*年*6*月*17*日至*21*日，美国佛罗里达州奥兰多。ACM，美国纽约州纽约，14页。https://doi.org/10.1145/3579371.3589064

*两位作者对本文的贡献相同。
†两位作者均为通讯作者。

允许免费制作本作品的全部或部分内容的数字版或纸质版副本，用于个人或课堂用途，前提是这些副本不得用于营利或商业用途，也不得为此制作或分发，并且副本首页须保留本声明和完整引文。本作品中由作者以外的其他方拥有版权的组件，必须尊重其版权。注明出处后允许摘录。若要以其他方式复制、再版、发布到服务器或分发至列表，则须事先取得特定许可和/或支付费用。请通过permissions@acm.org 申请许可。*ISCA '23*，*2023*年*6*月*17*日至*21*日，美国佛罗里达州奥兰多©2023 版权所有，归所有者/作者所有。出版权授权予 ACM。ACM
ISBN 979-8-4007-0095-8/23/06...$15.00https://doi.org/10.1145/3579371.3589064

1 引言

自主机器、计算摄影和太空探索等前景广阔的视觉计算应用，都从根本上依赖图像感知。传统 CMOS 图像传感器（CIS）仅负责“成像”，即捕获像素；而如今，CIS 日益集成计算能力，涵盖从信号预处理 [12] 到深度神经网络（DNN） [6, 27]，并横跨模拟域和数字域。例如，尼康的一款 CIS [27] 集成了图像处理器，可对每个图块进行曝光控制；索尼 IMX 500 CIS [16] 则将深度神经网络加速器与像素阵列集成，用于边缘视觉处理。

计算型 CIS扩展了架构师的传统设计空间，为探索多样化的权衡提供了令人兴奋的试验场。例如，计算成像传感器内处理会原位 处理大量的像素数据，并减少数据传输开销；然而，在传感器内部进行计算的效率较低，因为与标准CMOS节点相比，CIS 往往采用更老的工艺节点制造（受光子感测灵敏度 [66]的限制），这抵消了降低通信成本所带来的增益。设计者在设计传感器内架构时也面临着大量选择。例如，虽然三维堆叠通过实现混合工艺集成 [69],来提高能效，但它也可能增加功率密度，从而导致热诱导噪声，因此需要在下游进行更多处理。

本文介绍了 CamJ，这是首创的能耗建模框架，可帮助设计人员探索广阔的算法与硬件协同设计空间。CamJ 能够在给定每秒帧数（FPS）目标的情况下，进行组件级能耗估算。为此，CamJ 对计算型 CIS 流水线中主要结构之间的交互进行建模：像素感测 → 模拟处理 → 数字处理。因此，CamJ 可支持对 CIS 架构进行端到端优化，涵盖从光子采集到语义结果的全过程。

设计。作为用于系统级探索的能耗建模工具，CamJ 有两个设计原则（第3）。首先，我们使用声明式接口来描述算法和硬件配置。这基于以下观察：图像处理算法具有规律性的计算与内存访问模式；因此，软件和硬件的声明式描述足以准确估算硬件访问次数，同时简化用户的工作。该接口还将算法描述与硬件描述解耦，以便进行迭代式架构探索。

其次，我们调整能耗建模方法以纳入模拟域和数字域的特性（第4节）。在数字域中，CamJ 直接要求用户提供处理单元（PE）的每次操作能耗和每次存取存储器能耗。工具 [3, 51] 和数据 [19, 24] 可广泛用于获取这些统计数据，而这些数据已常规用于当今数字加速器的仿真 [2, 18, 19, 24]。然而，模拟组件的能耗取决于许多底层电路细节（例如偏置电流、电容），设计人员可能无法（或无需）获取这些信息，且目前不存在成熟的工具。因此，CamJ 基于经典CIS中的设计，为常见模拟组件提供默认能耗模型 [30, 34, 54, 71, 72]，同时提供接口以接受专家用户的定制设计。

验证。 我们针对九款近期的CIS芯片进行了广泛验证（第5节）。这些芯片涵盖了多种工艺节点、像素类型、存储器容量和处理形式（例如模拟与数字、像素并行与列并行）。与这些芯片跨越数个数量级的实测绝对能耗相比，CamJ 实现了0.9999的皮尔逊相关系数和7.5%的平均绝对百分比误差。

用例。 我们使用三个用例来展示CamJ 支持的架构探索（第6节）。首先，我们展示了对于计算主导型算法，传统的二维CIS设计几乎无法带来能耗收益。其次，转向三维设计可以利用先进工艺节点进行计算，从而降低能耗——但代价是功率密度增加。第三，即使计算能耗的降低幅度有限，CIS中的模拟处理也可以显著降低存储器能耗。

总之，本文的贡献如下：

- 我们全面综述了 CIS 的设计与缩放趋势，并指出了架构探索的新机遇。
- 我们提出了首个面向 CIS 的组件级能耗建模工具，并通过实际硅片对其进行了验证。
- 我们使用 CamJ 展示了三个用例，说明了片内与片外 CIS、二维与三维以及模拟与数字处理之间的架构权衡。

2 动机：计算型CIS设计与缩放趋势

我们首先讨论 CIS 的主要设计趋势，这一趋势构成了本文的基础本文：CIS 正日益走向计算化（第2.1节）。接着，我们将说明此类计算型 CIS 的能效优势，这是推动这一设计趋势的主要动力（第2.2节）。最后，我们将讨论如何实现这些能效优势所面临的挑战，这也正是本研究的动机（第2.3节）。

2.1 设计趋势

CIS 入门。 从根本上说，CIS 由两个基本组件构成，如图所示。2a：一组对光敏感的光电二极管阵列，用于将光子转换为电荷；以及一个读出电路，通过模数转换器（ADC）将电荷转换为数字值（即原始像素）。传统上，原始像素会通过 MIPI CSI-2 接口传输到主机，例如智能手机中的片上系统（SoC） [22]。其中的图像信号处理器（ISP）

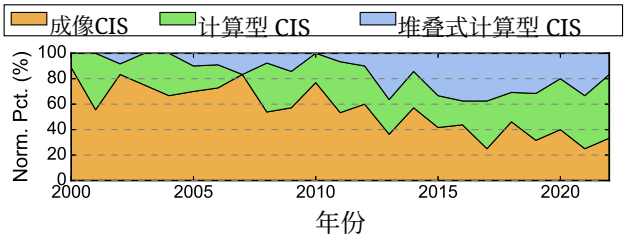
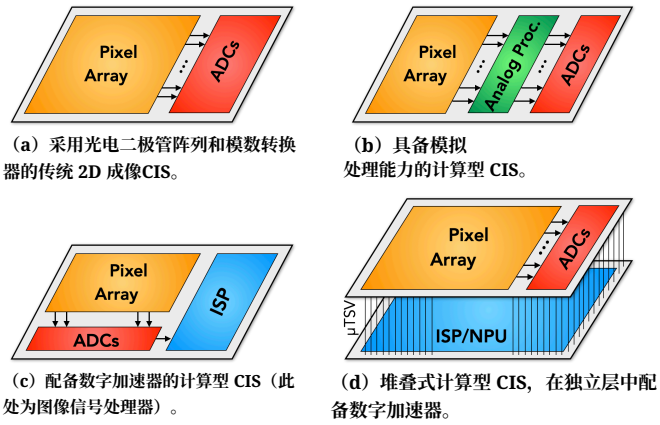


图 1：根据对 2000 年至 2022年发表的所有 ISSCC 和 IEDM 论文的综述，传统 CIS、计算型 CIS 和堆叠式计算型 CIS设计所占的比例。越来越多的 CIS设计采用计算型设计。



（a）采用光电二极管阵列和模数转换器的传统 2D 成像CIS。

（b）具备模拟处理能力的计算型 CIS。

（c）配备数字加速器的计算型 CIS（此处为图像信号处理器）。

（d）堆叠式计算型 CIS，在独立层中配备数字加速器。

图 2：CIS 架构演进。CIS 正逐渐摆脱纯成像器件 （a），以集成计算能力 （b） （c），有时采用三维堆叠方式（d）。

片上系统会去除感测伪影（例如降噪），并为计算机视觉任务和/或视觉显示准备像素。

CIS设计趋势。 CIS设计的一个明显趋势是转向在传感器中执行传统上在传感器外部完成的传感器计算，这促成了 计算型 *CIS*。图 1 展示了从2000 年到2022年期间，计算型 CIS 论文在 ISSCC 和 IEDM 中所占的百分比，该百分比是相对于同一时间范围内所有 CIS 论文而言的。越来越多的 CIS 设计集成了计算能力。

CIS 内部的计算可以在模拟和数字域中进行。图 2b 展示了将模拟计算集成到 CIS 芯片中的一个示例。模拟运算通常实现用于特征提取 [5, 6], 目标检测 [72], 和 DNN 推理 [30, 70]的基本操作。图 2c 展示了另一个集成数字处理（例如图像信号处理器 ISP）的示例 [52], 图像滤波 [35] 和 DNN [5]。

随着处理能力变得更加复杂，CIS 设计已采用 3D 堆叠技术，这一点从图 1 中堆叠式 CIS 数量的增加即可明显看出。图 2d 展示了一种典型的堆叠式设计，其中处理逻辑与像素阵列层分离并堆叠在一起。不同层通过混合键合或微型硅通孔进行通信

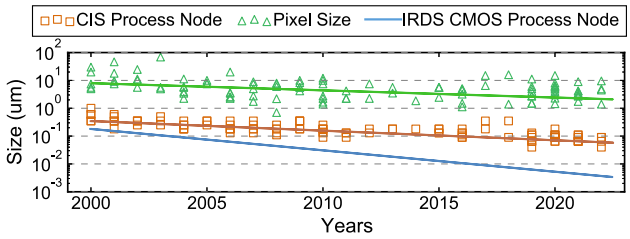


图 3：CIS 工艺节点始终落后于传统 CMOS 工艺节点。这是因为 CIS 节点缩放与像素尺寸缩放同步，而由于必须维持光子灵敏度这一根本需求，像素尺寸不会大幅缩小。

(μTSV） [49, 67]。处理层通常集成数字处理器；例如ISP [40], 图像信号处理器 [27, 38], 和 DNN 加速器 [16, 45]。三层堆叠式设计已被提出。Sony IMX 400 [25] 集成了像素阵列层、DRAM 层和带有 ISP 的数字层。Meta 构想了一种三层设计 [49]，包括像素阵列层、逐像素模数转换器层，以及集成 DNN 加速器的数字处理层。

2.2 计算型CIS的优势

当能效至关重要时，计算型 CIS 的出现并非巧合。从架构角度来看，计算型 CIS 能带来两大能耗优势。首先，将计算移入传感器内部，可以在更接近像素数据生成的位置使用这些数据。这样能够降低数据传输能耗，而这部分能耗可能在总能耗中占据主导地位。

具体而言，采用 μTSV 的 CIS 内部数据通信约消耗 1 pJ/B，而通过 MIPI CSI-2 接口将一个字节传输出 CIS，则需消耗约 100 pJ 的能量 [49]。例如，如果 CIS 能够直接执行目标检测深度神经网络，那么需要从传感器传输出去的数据量仅为几个字节（目标位置和标签），而非 1080p 图像所需的约 6 MB。

其次，计算型 CIS 也为模拟加速提供了天然平台，因为像素数据本身就源自模拟域，从而无需使用能耗密集型数模转换器；这类转换器往往是传统模拟加速器硬件开销的主要来源。与数字处理相比，模拟处理可最大限度地减少能耗密集型数据转换 [8, 50]，并降低计算和存储器能耗。

2.3 挑战与设计空间

然而，将计算移入 CIS 并非没有挑战。最重要的是，传感器内部的处理效率远低于传感器外部，这根本上是因为 CIS 工艺节点明显落后于传统 CMOS 的工艺节点。图3 展示了这一差异，其中方形标记表示 2000 年和 2022 年期间发表的所有 ISSCC 论文中 CIS 设计所采用的工艺节点，涵盖了不同时期业界领先的 CIS 设计。我们根据这些 CIS 设计拟合了一条趋势线，以便更清晰地展示缩放趋势。作为对比而言，底部的蓝线表示国际器件与系统路线图（IRDS）制定的传统 CMOS 工艺节点缩放路线 [1]。

大约在 2000年，CIS 工艺节点开始落后于传统 CMOS 节点，而且差距还在不断扩大。如今的 CIS设计通常采用65纳米及更早的工艺节点。这一差距并不是我们所选 CIS设计的偶然现象，而是根本性的：由于像素尺寸并未明显缩小，因此根本没有必要大幅缩小工艺节点。图中三角形 3 表示我们所综述的所有CIS设计的像素尺寸。CIS 工艺节点缩放的斜率几乎完全遵循像素尺寸缩放的斜率。像素尺寸之所以没有缩小，是为了确保光敏感度：较小的像素能够收集的光子数量减少，从而直接降低动态范围和信噪比（SNR） [4]。

通过 3D堆叠技术 [69], 可以缓解传感器内处理效率低下的问题，该技术支持异构集成：像素层和计算层可以分别采用各自最优的工艺节点。然而，堆叠可能会增加功率密度，尤其是在未来 CIS 集成更多处理能力的情况下。因此，要充分发挥（堆叠式）计算型 CIS 的能力，就需要探索广阔的设计空间并应对关键挑战，下面列出了其中一些。提供一种能够轻松探索设计空间的工具，正是我们 CamJ 框架的目标。

- 在CIS内还是CIS外进行计算，以及计算什么?
- 如何架构堆叠式CIS中的每一层，以在不增加功率密度的情况下实现能耗降低? • 在模拟域与数字域中分别计算什么?

3 CamJ 框架

迄今为止，还没有任何框架能够让设计人员在系统层面探索计算型 CIS复杂的设计空间。我们的 CamJ 框架旨在填补这一空白（第 3.1 节）。我们首先概述 CamJ 的设计原则，随后介绍 CamJ 的设计内部结构（第 3.2 节）。我们通过一个具体示例，从设计人员的角度展示如何使用 CamJ （第 3.3 节）。

3.1 CamJ 在设计周期的哪个阶段使用?

CamJ 旨在用于系统级探索，即在每个组件设计勾勒完成之后；一个类比是片上系统（SoC）与加速器设计。在进行系统级探索之前，团队通常已经掌握一系列组件级设计，这些设计可能是获得授权的知识产权（IP）模块、文献中的参考设计，或组织内其他团队较早完成的设计（例如使用综合流程或高级综合工具）；在所有情况下，组件级能耗行为都是已知的，或可以使用 Aladdin [63] 和 OpenRAM [23]进行建模。

CamJ 帮助设计人员在将各个（数字和模拟）组件组合成最优系统时做出设计决策。理想情况下，设计人员使用 CamJ，根据各个组件的初始设计估算系统能耗；利用该估算结果，设计人员可以迭代地 优化组件/系统设计。例如，CamJ 可以识别能量瓶颈，并指导对相应组件进行重新设计。此外，设计人员还可以使用 CamJ，探索算法的最优映射与划分，

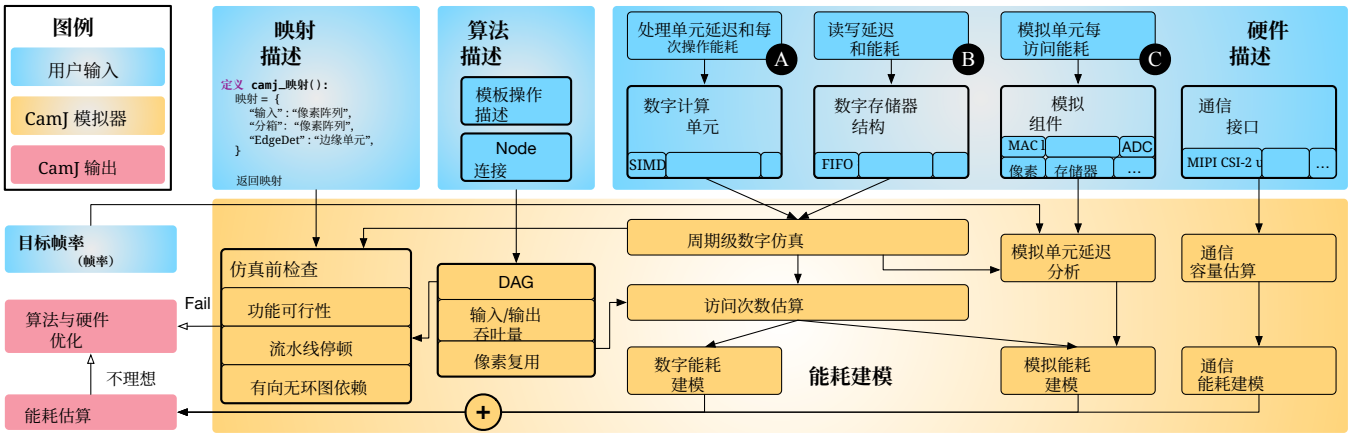


图 4：CamJ 框架概览，该框架在目标帧率（FPS）下提供组件级能耗估算。用户提供算法和硬件描述，以及二者之间的映射。 、 和 分别表示数字计算、数字存储器和模拟单元的每次访问能耗；前两者通过外部工具获取，最后一者由 CamJ提供。组件列表见表 1，其中列出了 CamJ所支持的数字计算/存储器和模拟组件。CamJ 随后估算每个硬件组件的访问次数，以获得总体能耗估算。

以便在模拟域与数字域之间，或在 CIS 内部与外部之间分配算法，从而在满足性能目标的前提下，最大限度地降低系统总能耗。

CamJ 不是一种综合工具；它不会生成（也不会估算）加速器的能耗。相反，CamJ 可以与高层次综合结合使用：可以先使用高层次综合生成一个加速器，然后使用 CamJ 在更大的系统中探索该加速器如何/是否适合计算型 CIS，以最大化端到端应用增益。

3.2 设计原则与概述

与任何能耗建模工具一样，[7, 33, 43], CIS 的总能耗是以下两项乘积之和：1）每个硬件单元的访问次数；2）每次访问能耗。因此，CamJ 的核心目标是开发一种建模方法，利用编程接口准确估计这两个统计量；更重要的是，该接口仅要求用户输入无法自动推断的信息。图 4 展示了 CamJ 框架的概览。第 3.3 节和第 4 节分别讨论编程接口和建模细节。下面将概述该接口和建模方法。

接口。 CamJ 注意到，CIS 处理基于模板的图像处理任务，这类任务具有规则的计算和存储器访问模式（即几乎没有或完全没有控制流），并且会静态映射到硬件单元。因此，无需了解实现细节，仅凭高级算法描述和硬件配置即可推断访问次数统计信息。基于此，CamJ 提供了一个声明式接口：使用有向无环图（DAG）描述算法，并指定每个节点的输入/输出/模板维度；同时，将硬件抽象为一组基本单元，每个单元执行特定的感测、计算或存储器操作。

CamJ 的接口还解耦了算法描述、底层硬件以及二者之间的映射。解耦的接口有助于迭代式系统设计流程，在此过程中，算法、硬件以及从算法到硬件的映射都可以独立变化。例如，可以评估通过重写算法描述来进行算法更改，而无需改动硬件设计；或者通过描述新的映射，探索不同的算法到硬件映射（例如，在模拟与数字之间，以及传感器内与传感器外之间进行拆分）。

内部建模。 CamJ 审慎地采用不同方法，分别获取数字域和模拟域中的每次访问能耗。对于数字结构，CamJ 直接要求用户提供计算处理单元（ ）的每周期能耗，以及存储器单元（ ）的每次访问能耗。这些统计数据通常通过 ASIC 综合流程或常用工具（例如 CACTI [3]和 OpenRAM [23]）获得，并且是当今数字加速器模拟器进行能耗估算时的常规输入 [2, 19, 37]。

有人可能会惊讶地发现，CamJ 会直接要求提供数字结构的每周期／每次访问能耗。这是因为 CamJ 的设计理念（第3.1节）：它并非用于生成数字加速器；相反，它帮助评估加速器如何融入整个计算型 CIS 系统。因此，CamJ 希望设计人员已有数字加速器的初步设计（无论是手动设计、由高层次综合生成，还是获得授权的 IP），此时便可提供每周期／每次访问能耗统计数据。

不同于数字结构，针对模拟结构的能耗建模工具很少，其能耗（ ）取决于许多底层电路细节（例如负载电容、增益、偏置电流），要求系统级设计人员提供这些信息既繁琐，或许也不合理。我们的设计决策是提供一个低层接口，以接受专家用户提供的这些参数，同时还基于模拟组件的经典实现和这些组件的延迟分析（第4.1节）提供默认能耗模型。

最后，CamJ 会在仿真前进行一系列设计检查，以确保算法与硬件的组合：1）在功能上可行（例如，模拟域与数字域之间必须存在模数转换器）；2）不存在流水线停顿（以避免累积过长的帧延迟）；3）算法有向无环图中的依赖关系定义良好（例如，不存在环）。我们会在检查时提供反馈。

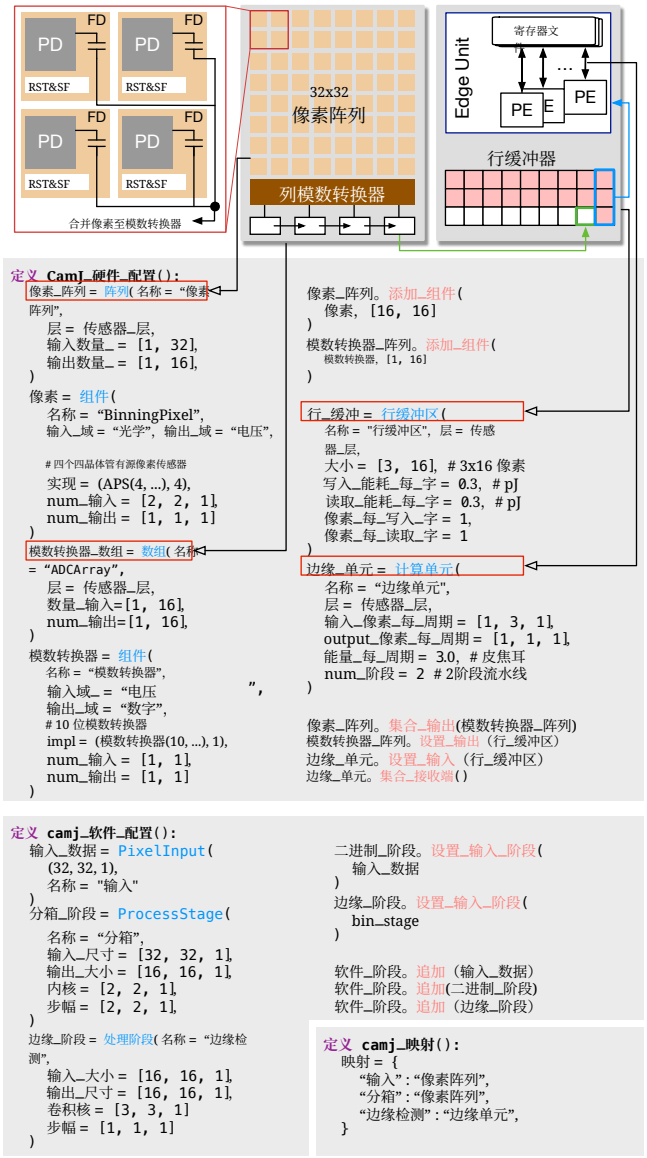


图 5：使用 Python 中的 CamJ 编程接口定义简单 CIS 的示例。顶部展示了模拟 CIS 的硬件架构。

失败情况和详细的能耗分解，这有助于设计人员迭代优化算法和/或硬件。

3.3 编程接口

我们在图5中给出了编程接口并说明主要设计决策。

一个示例。 图 5 展示了一个使用 CamJ 编程接口的具体示例。在这个概念性的 CIS 设计中，包含一个 32×32 像素阵列，每个 2×2 像素瓦片首先被平均（即"合并"）以生成一幅 16×16 图像。随后，传感器会对该图像执行数字边缘检测，然后通过 MIPI CSI 接口发送边缘数据。 `camj_hw_config`

表 1：CamJ 支持的硬件单元列表。APS：有源像素传感器；DPS：数字像素传感器；PWM：脉冲宽度调制；MAC：乘累加。

	模拟 (A-Component)	数字
存储器	无源/有源 ， 采样保持	FIFO, 行缓冲器 ， 双缓冲静态随机存取存储器
计算	像素（APS、DPS、PWM）， 模数转换器、乘累加器、最大值、缩放， 加法、对数、绝对值、比较器	脉动阵列，通用流水线加速器

function 和 camj_sw_config function 分别描述硬件组件和算法。camj_mapping 将每个算法阶段映射到硬件组件。接下来我们将解释各个部分。

算法描述。 camj_sw_config 中的代码描述了整个处理流水线的有向无环图，该流水线从像素阵列生成的原始像素（PixelInput）开始，依次经过两个处理阶段：用于像素合并的 bin_stage 和用于边缘检测的 edge_stage。set_input_stage 方法将各阶段连接起来，形成有向无环图。

请注意，算法描述并不要求提供实际的算术细节；我们观察到，图像处理算法可以抽象为模板运算，每次在像素的局部窗口上运行 [26, 58] ——卷积（或者按照传统图像处理术语所说的图像滤波）就是一个典型示例。自2000 年以来，我们综述的所有 ISSCC/IEDM 论文中都存在这一观察结果。不规则计算使硬件设计复杂化并增加能耗，从而违背了 CIS内计算 的初衷。

因此，用户只需指定输入/输出图像尺寸（input_size、output_size），以及模板窗口（内核）和步幅大小（stride）。鉴于模板运算具有规则的计算和数据访问模式，CamJ 能够准确估算不同硬件结构的访问次数，进而估算能耗。尽管如此，CamJ 也支持将离线采集的、不规则算法的存储器跟踪数据作为输入，并可将其与 DRAMPower [10] 等外部工具集成，以估算不规则算法的能耗。

硬件描述。 `camj_硬件_配置` 描述硬件架构，我们在图顶部展示了该架构。5。硬件描述包含两个组件：模拟处理单元和数字处理单元。

- 模拟单元。CIS硬件必然从模拟单元开始，从较高层次来看，这些模拟单元被描述为一组模拟功能阵列（AFA），而每个模拟功能阵列又由一组模拟功能组件（A-Component）组成。CIS 中最重要的模拟功能阵列是像素阵列（像素_阵列），其中每个 A-Component 都是一个像素，通过添加_组件方法将其添加到像素阵列中。在图 5的示例中，像素阵列之后是另一个模拟功能阵列，即模数转换器阵列（模数转换器_阵列），其中每个A-Component 都是一个模数转换器。

从用户的角度来看，每个A-Component都执行某种特定的（算术）运算。除了像素或模数转换器之外，CamJ 还提供 CIS中使用的其他常见 A-Component，例如 MAC 或对数运算。模拟组件的完整列表

A-Components 在表 1 中。每个 A-Component 的能耗由其电路级实现决定，并通过 `impl` 方法对用户隐藏。第 4.2 节将进一步介绍我们如何通过将每个 A-Component 映射到其模拟电路实现来建模其能耗。

不过，用户确实需要提供 AFA 输入和输出数据的信号维度（num_input 和 num_output）以及信号域（input_domain 和 output_domain ）。这些参数使 CamJ 能够检查模拟 CIS 在功能上是否可行。具体而言，消费者单元的 input_domain 必须与生产者单元的 output_domain 匹配。例如，如果生产者处于电荷域，而消费者处于电压域，CamJ 将要求设计人员插入一个电荷到电压转换组件[1]，这会影响能耗。同样，如果消费者单元的 num_input 与生产者单元的 num_output 不匹配，硬件就必须在两者之间加入一个模拟缓冲器，这同样可能影响能耗。

● 数字单元。硬件的数字部分通过指定一组经由存储结构进行通信的计算单元来描述。在此示例中，计算单元是边缘检测加速器（通过 ComputeUnit 实例化），它从行缓冲区（LineBuffer）读取数据。行缓冲区是一种预定义的存储结构，用于存储来自像素阵列的数据，而像素阵列是前文所述的一种模拟单元。

表 2 的第 2 列列出了 CamJ 中可用的存储结构和计算单元。我们支持图像/视觉处理中常见的三种存储结构：先进先出（`FIFO`）、行缓冲器 [26, 68]（`LineBuffer`）和双缓冲静态随机存取存储器（`DoubleBuffer`）。计算单元通过 `ComputeUnit` 接口抽象为流水线加速器。我们还提供了 `SystolicArray` 类，用于描述脉动阵列，因为脉动阵列对于执行深度神经网络至关重要。

借助通用的流水线加速器接口（ComputeUnit），用户可以对各种（图像处理）加速器进行建模。为了描述流水线加速器，CamJ 需要三个主要参数：每个周期读取的像素形状（input_pixel_per_cycle）、每个周期生成的像素形状（output_pixel_per_cycle）以及流水线深度（num_stages）。利用这些统计信息，CamJ 执行周期级仿真有两个目的。首先，CamJ 可以检查加速器是否会导致 CIS 流水线停滞；如果会，则要求重新设计加速器。其次，CamJ 可以估算数字域的总延迟，这对模拟能耗估算至关重要。正如我们将在第 4.1 节所讨论的，停滞检查和延迟估算对于模拟能耗估算至关重要。

映射。 camj_映射函数将每个算法阶段映射到一个硬件单元。代码含义一目了然。用户只需将算法重新映射到硬件，即可探索不同的系统设计。通过映射函数将算法与硬件描述解耦，也便于表达硬件复用——只需将不同的算法节点映射到同一个硬件组件即可。

[1]除非消费者的输出处于电压域中，此时消费者固有的电容会自然地充当模拟缓冲器。

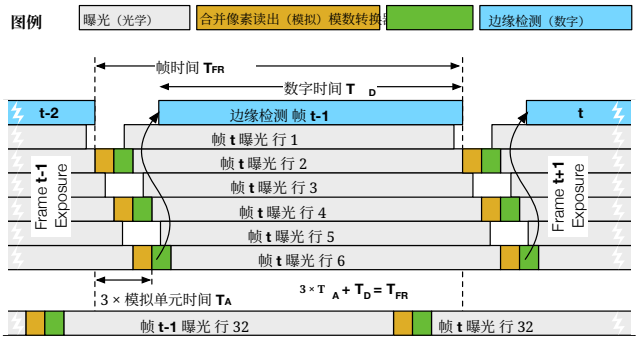


图 6：图 5 中示例的流水线示意图（非按比例绘制），5 此时没有发生流水线停顿。

4 能耗建模方法

CIS 传感器每帧的能耗等于模拟、数字和数据通信部分能耗之和：

$$E^{\text{frame}} = E_{\text{a}}^{\text{frame}} + E_{\text{d}}^{\text{frame}} + E_{\text{c}}^{\text{frame}} \tag{1}$$

在分别介绍如何对模拟组件 Eframea （第 4.2节）、数字组件 E framed （第 4.3节）和通信组件 Eframec （第 4.4节）进行建模之前，我们首先讨论能耗建模的一个前提：延迟估算（第 4.1节）。

4.1 延迟估算

模拟和数字能耗都与电路速度相关。例如，在模拟域中，响应速度更高的运算放大器（OpAmp）需要更大的偏置电流，从而增加能耗（假设 OpAmp 在固定时长内处于活动状态，例如用于模拟帧缓冲区时）。数字单元的延迟通过周期级仿真进行估算，具体方法如上一节所述。相比之下，模拟单元的延迟取决于完全设计好的电路所特有的许多参数。我们发现要求用户输入既繁琐又容易出错：用户往往需要不断调整底层参数，最终却得到一个无法达到目标帧率的设计。相反，CamJ 的核心洞见是，可以根据规定的帧率自动推断每个模拟单元的延迟。

具体而言，CamJ 所依赖的基本观察是：*CIS* 流水线经过设计，能够始终避免停顿。这是因为，当像素阵列以恒定速度曝光于光线下时，输入到流水线的数据也会以恒定速率生成。如果流水线在后续阶段发生停顿，帧延迟就会逐渐累积，从而导致响应延迟过长或帧丢失。因此，CIS 设计人员会确保硬件流水线始终不会停顿。在完全流水化的硬件中，每个流水线阶段都必须具有大致相同的延迟；这正是我们进行延迟估算的依据。

示例。图6 展示了图5中示例的流水线时序。帧时间 T_{FR} 为 1/FPS，其中 FPS 为目标帧率。在图中，帧时间是指当前帧的像素可供读出的时刻到当前帧的计算完成时刻之间的延迟。“合并像素读出”和“模数转换器”是两个模拟单元，它们共用相同的延迟 T_A（即平衡流水线），该延迟需要进行估算。“边缘检测”是数字单元，在第二行写入行缓冲器后便开始运行。

为了估算 T_A，我们首先模拟数字域，以估算整个数字域的延迟（此处为T_D）。给定帧时间 T_{FR}后，我们便可估算模拟单元还剩多少时间。在此示例中，$T_A = \frac{T_{FR}-T_D}{3}$。

CamJ 将分析硬件描述，并在检测到潜在停顿时要求用户重新设计硬件以避免停顿。具体而言，CamJ 会检查并避免以下三种情况：1）所需像素尚未由生产者生成；2）两个阶段之间的存储器已满；或 3）存储结构中的访问端口数量不足。

4.2 模拟能耗建模

每帧模拟能耗 $E_{\text{frame analog}}$是各A-Component 每次访问的能耗乘以该组件访问次数后所得的总和。支持的1 个A-Component见表CamJ 。

$$E_{\text{a}}^{\text{frame}} = \sum_i \left(E_{\text{a}}^{\text{component}[i]} \times \text{Num}_{\text{access}}^{\text{component}[i]}\right) \tag{2}$$

A-Component 访问次数建模。某个A-Component 的访问次数，是指每帧中该A-Component 被使用的次数。回顾第 3.3 节可知，每个A-Component 都是模拟功能阵列（AFA）的一部分。CamJ 利用了这一基本观察：图像处理中的模板运算具有规律的计算和存储器访问模式，因此，同一 AFA 中每个 A-Component 的访问次数相同。

因此，组件的访问次数 i 就是映射到包含该组件的 AFA 上的操作总数（$\text{NumAFA}^{[j]}\text{ops}$）与该 AFA 中组件数量（$\text{NumAFA}_{[j]}\,\text{component}_{[i]}$）的比值：

$$\text{Num}_{\text{access}}^{\text{component}[i]} = \frac{\text{Num}_{\text{ops}}^{\text{AFA}[j]}}{\text{Num}_{\text{component}[i]}^{\text{AFA}[j]}} \tag{3}$$

分子可根据模板操作的算法描述轻松得出（例如，计算卷积中的乘加运算次数）。分母是num_component AFA 的属性（参见图 5）。

对 A-Component的访问能耗进行建模。 在内部，每个 A-Component 由一组模拟单元元件构成，我们称之为 A-Cell。要对 A-Component 的每次访问能耗进行建模，需要了解其单元级实现。专家用户可以定义新的单元参数和/或 A- Component的单元级实现。若未进行此类定义，每个 A-Component 都采用从经典和最新 CIS 设计综述中得出的默认实现 [30, 34, 54, 71, 72] 4 。例如，T-APS 像素 A-Component 由一个光电二极管（PD）A-Cell、一个浮动扩散节点（FD）A-Cell和一个源极跟随器（SF）A-Cell组成；采用开关电容电荷再分配实现的乘法器 [42] 由一个电容阵列 A-Cell和一个运算放大器 A-Cell组成。

现在我们来介绍 A-Component 的能耗建模，但请注意，这些设计细节对于普通用户而言是抽象隐藏的。 E组件的能耗是组件中各个 A-Cell 的能耗与其 A-Cell 访问次数的加权总和：

$$E_{\text{a}}^{\text{component}[i]} = \sum_j E_{\text{a}}^{\text{cell}[j]} \times \text{Num}_{\text{access}}^{\text{cell}[j]} \tag{4}$$

尽管存在种类繁多的高层模拟电路，但用于传感器内模拟计算的 A-Cell 根据电路特性可分为三类：动态 A-Cell、静态偏置 A-Cell 和非线性 A-Cell。它们以不同的方式消耗能量。

● 动态 A-Cell。动态电路的能量来源于电路中总电容的充电和放电：

$$E_{\text{a}}^{\text{cell,d}} = \sum_i^{N_c} C[i] \times V_{\text{VS}}[i]^2 \tag{5}$$

其中，N_c 表示动态电路中的电容节点总数，$C[i]$和 $V\text{VS}[i]$分别表示 i^{th}电容节点处的电容和电压摆幅。典型的动态 A-Cell 包括电容式数模转换器（CDAC）和无源模拟存储器。

在公式 5中，$V\text{VS}$ 由模拟电源 $V\text{DDA}$ 以及模拟电源与地之间串联的晶体管数量决定。节点电容 C 由其热噪声和计算精度决定。为保证模拟计算的准确性，最大热噪声应保持在数据分辨率的12LSB 以下：

$$\sigma_{\text{thermal}} = \sqrt{\frac{kT}{C}}, \quad 3\sigma_{\text{thermal}} < \frac{1}{2}\text{LSB} \tag{6}$$

其中，LSB $= V\text{VS}/2$表示数据分辨率。数据分辨率取决于算法。例如，如果 $V\text{VS} = 1V$ 且所需分辨率为 8 位，则热噪声应小于 $\frac{1}{2^{8\text{-bit}}}1\text{V} = 2.6\text{mV}$，据此可得 C。

● 静态偏置 A-Cell。静态偏置电路的能耗来自模拟电源 V DDA下偏置电流在特定时间段内的积分：

$$E_{\text{a}}^{\text{cell,s}} = V_{\text{DDA}} \times I_{\text{bias}} \times t_{\text{static}} \tag{7}$$

其中，Ibias为偏置电流，$t$static为A单元处于静态偏置状态的时间。

我们根据电路细节提供两种估算 Istatic的方法。对于 I bias直接驱动负载电容的A单元（例如像素中的静态偏置 SF），Ibias由在给定时间内为负载充电的要求决定：

$$I_{\text{bias,1}} = \frac{C_{\text{load}} \times V_{\text{VS}}}{t_{\text{static}}} \tag{8}$$

其中，Cload为负载电容。能耗可简化为：

$$E_{\text{a}}^{\text{cell,s}} = C_{\text{load}} \times V_{\text{VS}} \times V_{\text{DDA}} \tag{9}$$

对于 A-Cell 不直接驱动负载电容的情形 I_{bias}（例如模拟存储器中的差分运算放大器或离散时间积分器），$I_{\text{偏置}}$ 可通过经典的 $\frac{g_{\text{m}}}{I_{\text{d}}}$方法 [32]确定：

$$I_{\text{bias,2}} = \frac{2\pi \times C_{\text{load}} \times \text{GBW}}{g_{\text{m}}/I_{\text{d}}} \tag{10}$$

其中，$\frac{g_m}{I_d}$ 是一个与工艺无关的因子，其取值范围为 10 到 20，具体取决于晶体管的反型程度；GBW 是增益 (G) 与带宽 (BW) 的乘积。

为了使用公式7 和公式10，CamJ必须估算 BW 和t静态，二者都取决于A-Cell延迟。具体而言，BW 是 A-Cell 延迟的倒数，而t静态为：

$$t_{\text{static}} = T_A - \sum_{i}^{K} t_i^{\text{cell}} \tag{11}$$

其中，T_A 是包含该 A-Cell 的A-Component 的延迟，并在第4.1 节中进行估算；K 是当前A-Cell之前位于A-Component 关键路径上的单元数量，t单元i 是一个 A-Cell的延迟。在用户未提供时序条件的情况下，我们根据模拟信号在所支持的 A-Co mponent中单向流动这一事实，将 A-Component的延迟平均分配给每个 A-Cell，因此所有 A-Cell 都位于关键路径上。

● 非线性 A-Cell。对于具有非线性传递函数的电路，例如模数转换器和比较器（本质上是 1位模数转换器），它们同时包含动态偏置/静态偏置电路单元和数字逻辑，因此很难通过解析公式估算能耗。相反，我们使用模数转换器的 Walden优值图 [53]（根据近期发表的CIS 论文进行综述），该图展示了模数转换器的每次转换能耗与其采样率之间的关系。具体而言，给定模数转换器采样率（A-Cell A-Cell 延迟的倒数），在缺少详细用户输入的情况下，我们使用该采样率下每次转换能耗的中位数作为估算值。非线性A-Cell 的总能耗因此通过其估计优值与所需转换次数的乘积获得：

$$E_{\text{a}}^{\text{cell,nl}} = FoM\ [\text{J/conversion}] \times \text{Num}_{\text{conversion}} \tag{12}$$

特定A-Cell 的访问次数，是该 A-Cell沿空间和时间尺度被使用以生成一个 A-Component 输出的次数：

$$\text{Num}_{\text{access}}^{\text{cell}[j]} = \text{Num}_{\text{spatial}}^{\text{cell}[j]} \times \text{Num}_{\text{temporal}}^{\text{cell}[j]} \tag{13}$$

例如，如果一个 A-Cell 表示像素中的静态偏置 SF，则 Numstatic-SFspatial 将是像素中的SF数量，而 Numstatic-SFtemporal 将是像素电荷被读出的次数（例如，如果使用相关双采样来降低噪声，则为 2 [9]）。A单元的访问次数信息针对每个 A- 组件进行硬编码，对普通用户进行了抽象隐藏，但可以针对定制设计进行更新。

4.3 数字能耗建模

帧 Eframed 的数字能耗是每个计算单元 Ecomp d 的计算能耗与每个存储结构 Ememd 的能耗之和：

$$E_{\text{d}}^{\text{frame}} = \sum_{i} E_{\text{d}}^{\text{comp}[i]} + \sum_{j} E_{\text{d}}^{\text{mem}[j]} \tag{14}$$

每个计算单元的能耗是每周期能耗 E周期 d与周期数 Numcycle 的乘积：

$$E_{\text{d}}^{\text{comp}[i]} = E_{\text{d}}^{\text{cycle}[i]} \times \text{Num}_{\text{cycle}} \tag{15}$$

我们依赖用户提供 E周期 d，该值通常通过 HLS/ASIC综合流程获得。相比之下，周期数由 CamJ 通过周期级仿真获得。

内存访问的能耗是漏电能耗与动态访问能耗之和；后者等于单次内存读取 Ereadd 或写入（Ewrited ）的能耗乘以内存读取（Numread）或写入（Numwrite）的总次数：

$$E_{\text{d}}^{\text{mem}[j]} = E_{\text{d}}^{\text{read}[j]} \times \text{Num}_{\text{read}}^{[j]} + E_{\text{d}}^{\text{write}[j]} \times \text{Num}_{\text{write}}^{[j]} + P_{\text{d}}^{\text{leakage}[j]} \times \frac{1}{\text{FR}} \times \alpha \tag{16}$$

漏电能量是漏电功耗 Pleakage d 与存储器活动时间（即未进行功耗门控）的乘积，而该活动时间是帧时间 ^{1}FR 的一部分 α。用户提供动态读写能量和漏电功耗；访问次数和活动时间则来自 CamJ 仿真。

4.4 通信能量建模

通信功耗主要由通过能耗较高的 MIPI CSI-2 接口将数据传输到传感器外部所需的能量决定；对于 3D 堆叠 CIS，还包括 μTSV 的能量。文献通常以每字节能耗的形式给出这两种接口的能耗。因此，通信能耗为：

$$E_{\text{c}}^{\text{frame}} = E_{\text{c}}^{\text{mipi}} \times \text{Num}_{\text{Bytes}}^{\text{mipi}} + E_{\text{c}}^{\text{tsv}} \times \text{Num}_{\text{Bytes}}^{\text{tsv}} \tag{17}$$

Emipic 和 Etsvc 由用户提供，相关表示数据已在文献 [48] 中报告。两个接口的数据量统计均由 CamJ 仿真生成（基于算法描述以及算法到硬件的映射）。

5 CAMJ 验证

在本节中，我们使用CamJ对来自九款近期CIS芯片的真实测量数据进行验证 [5, 13, 16, 29, 30, 54, 61, 70, 72]，如表2所示。这些设计涵盖了多种设计维度，包括二维和三维设计、不同的工艺节点、像素类型，以及模拟域和数字域中的处理单元设计与存储器容量。

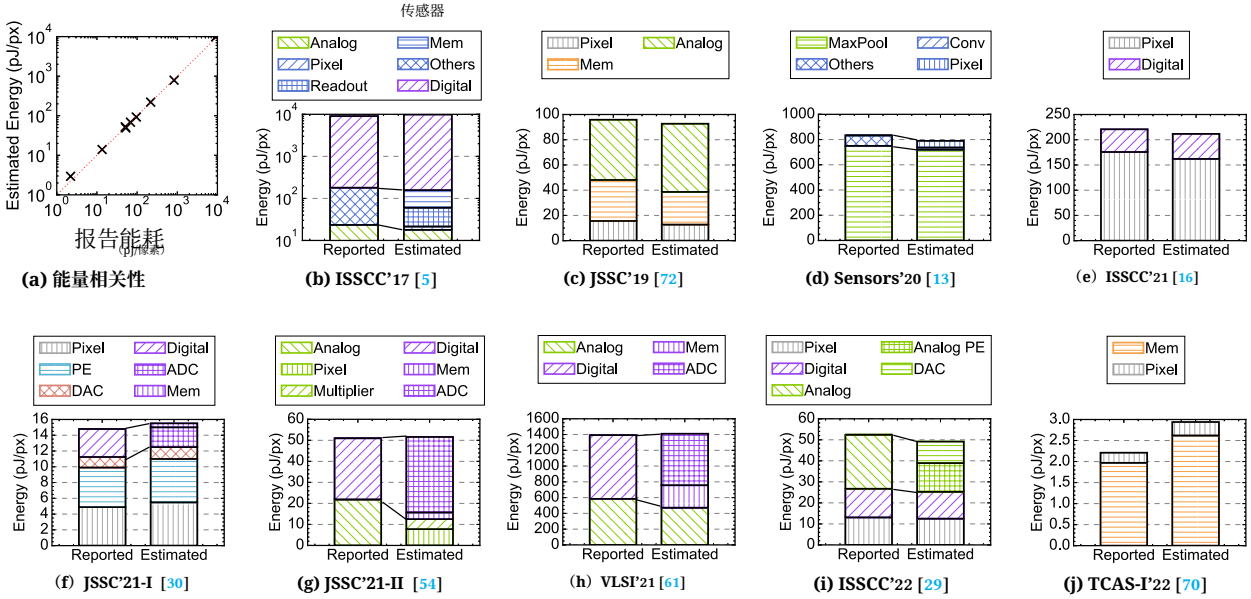
图 7 对比了原论文报告的每像素估算能耗与实际能耗。我们的估算与测量结果高度吻合，测量值跨越数个数量级，体现了 CIS 设计风格的多样性，以及 CamJ 能够灵活支持并准确建模的系统功耗/能耗范围之广。在所有设计中，CamJ 的平均绝对百分比误差为 7.5%，皮尔逊相关系数为 0.9999。

图7b – 图 7j 对比了九种设计的详细能耗分解。尽可能采用论文中报告的电路参数。对于 SRAM，我们使用 DESTINY [57] 来获取每次访问能耗。三篇进行数字计算的论文均执行深度神经网络，其中处理单元（PE）为 MAC 单元；我们采用 65 nm MAC 单元设计的综合结果作为每次 MAC 运算的能耗 [5]，，并基于经典 CMOS 缩放规律将其缩放至其他工艺节点 [60, 64]。

尽管总体而言，CamJ 能够准确地进行组件级和全系统能耗估算，但我们发现结果存在偏差主要有两个原因。首先，当 CamJ

表2：用于验证的CIS设计汇总，涵盖了广泛的设计变体。 *表示原始论文未报告的数据，数值基于我们的合理估算。模拟存储器大小的单位为"模拟值的数量"。

CIS	工艺节点	堆叠式	模拟 像素	模拟 存储器	模拟 处理单元运算	模拟 处理单元位置	模拟 工作领域	数字 存储器	数字 处理单元尺寸
ISSCC'17 [5]	65nm	No	三晶体管有源像素传感器	20 × 80	平均与累加	列与芯片	电荷与电压	160千字节	4 × 4 × 64
JSSC'19 [72]	130nm	No	4T有源像素传感器	4 × 240	对数子像素。	列	电压	-	-
传感器的20 [13]	110nm	No	4T有源像素传感器	No	MAC 和最大池化	列	电压	-	-
ISSCC'21 [16]	65纳米/22纳米	Yes	4T有源像素传感器 *	No	-	-	-	8MB	1 × 2304
JSSC'21-I [30]	180nm	No	PWM	No	MAC	列	时间与电流	-	-
JSSC'21-II [54]	110nm	No	4T有源像素传感器	No	MAC	列	电荷	-	-
VLSI'21 [61]	65 纳米/28 纳米	Yes	DPS	No	-	-	-	6MB	-
ISSCC'22 [29]	180nm	No	PWM	No	MAC	列	时间与电流	256B*	1
TCAS-I'22 [70]	180nm	No	三晶体管有源像素传感器	No	乘法与加法	像素&芯片	当前	-	-



(a) 能量相关性 (b) ISSCC'17 [5] (c) JSSC'19 [72] (d) Sensors'20 [13] (e) ISSCC'21 [16]

(f) JSSC'21-I [30] (g) JSSC'21-II [54] (h) VLSI'21 [61] (i) ISSCC'22 [29] (j) TCAS-I'22 [70]

图7：验证结果。CamJ达到了0.9999 的皮尔逊相关系数。多篇论文将不同组件归入粗粒度的"模拟"、"数字"或"其他"类别。我们展示了详细的分类结果，并指出在我们的估算中多个细粒度类别的总和何时与原论文中的某个粗粒度类别相对应。

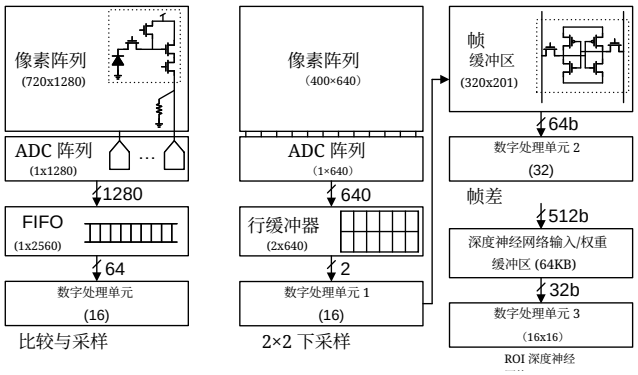
无法获取详细的设计参数时，其结果准确性会降低。例如，图7f、图7g和图7j中的像素估算分别显示出12.4%、38.9%和33.3%的绝对误差，这是由于像素斜坡发生器（图7f）、像素寄生电容（图7g）和光电二极管电压摆幅（图7j）的电路参数不足所致。类似地，图7f和图7b中的模拟处理单元分别由于采样电容（图7f）和感测放大器转换能耗（图7b）的电路参数不足，产生了9.3%和23.7%的绝对误差。相比之下，在图7c中，由于提供了模拟处理单元的详细设计参数，估算误差仅为0.4%。

另一个误差来源是实际电路设计与CamJ的默认电路模板之间存在差异。例如，图7g和图7h中的模数转换器分别存在31.7%和16%的绝对误差[2]；原始设计采用低功耗动态技术（图7g），而CamJ估算的是基于FOM调研得出的模数转换器能耗[53]。图7j中的存储器估算误差为33.0%，因为原始设计采用定制的8T SRAM，而CamJ使用来自DESTINY的标准6T SRAM[57],，导致漏电功耗更高。

6 体系结构探索

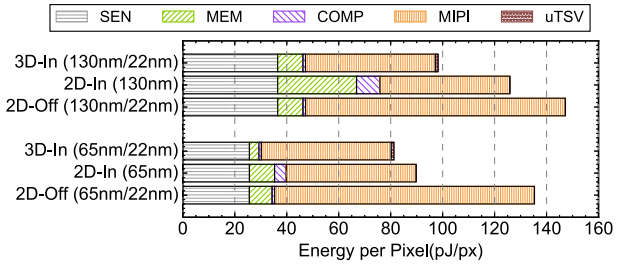
现在，我们展示三个互补的示例，说明如何使用CamJ探索架构权衡。这三个用例相互补充。首先，我们探讨将计算移入CIS内部何时能够带来能耗收益（第6.1节）。我们表明，在传统的二维设计中，能耗降低幅度很小。随后我们展示，引入三维堆叠后，由于能够实现混合集成，可以进一步提高能效，但代价是更高的功率密度（第6.2节）。最后，我们讨论将计算移入模拟域所带来的权衡（第6.3节）。

[2]两篇论文都将模数转换器视为数字单元，这也是我们在此采用的做法。

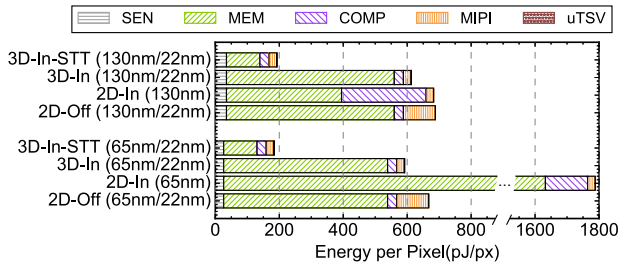


(a) Rhythmic。 **(b) Edgaze。**

图 8：以下内容的硬件设计 (a)Rhythmic Pixel Regions [37]，这是一种基于 ROI 的图像编码器；以及 (b) Ed-Gaze，这是一种视线追踪算法。在其原始设计中，模数转换器之后的所有操作都在 CIS 之外进行；我们评估了将整个执行过程移入 CIS 的优势。



(a) Rhythmic Pixel Regions 上的结果。



(b) Ed-Gaze 上的结果。

图 9：2D 传感器内（2D-In）、2D 传感器外（2D-Off）、3D 传感器内（3D-In）以及采用 STT-RAM 的 3D 传感器内（3D-In-STT）处理的能耗对比。SEN：截至并包含模数转换器的所有部分；MEM：存储器；COMP：计算；MIPI：MIPI CSI-2 通信；uTSV：层间 μTSV 通信。

6.1 传感器内计算与传感器外计算

在 CIS 内进行计算可通过在传感器内消耗像素数据来降低数据传输成本。为探索其优势，我们评估了近期的两篇论文 Rhythmic PixelRegions [37] 和 Ed-Gaze [17]，二者均生成一个小型感兴趣区域 (ROI) 这些算法基于原始全分辨率图像进行计算，因此，将 ROI 生成移至传感器内部，可能会从传感器内计算中受益。

图 8a 和图 8b 分别展示了两种算法的执行流程。在原始论文中，模数转换器之后的所有操作都在传感器外部执行；我们使用 CamJ 评估将整个执行过程移至传感器内部后的能耗。具体而言：

- **Rhythmic Pixel Regions**：1280×720 像素阵列由专用加速器（比较与采样）处理，以生成 ROI，平均可将图像尺寸缩小 50%。ROI 生成每帧大约执行 7.4×10^6 次算术运算。
- **Ed-Gaze**：640×400 像素阵列首先经 2×2 下采样，然后与前一帧进行逐像素减法运算，以生成事件图；随后由深度神经网络处理该图以生成 ROI。ROI 平均可将图像尺寸缩小 25%。深度神经网络占据主要计算量，每帧约执行 5.76×10^7 次乘加运算。

我们使用 CamJ 评估两种硬件配置：

- **2D-In (H)**：采用**H** 工艺节点制造的二维 CIS；整个执行过程均在 CIS 内完成。
- **2D-Off (H/L)**：采用**H** 工艺节点制造的二维 CIS；模数转换器之后的所有处理均在片上系统上进行，该片上系统采用**L** 工艺节点制造。我们评估两种 CIS 工艺节点：130 nm 和 65 nm，这两种节点在CIS设计中都很常见（表 2）。我们将片上系统的工艺节点设为 22 nm。

图 9a 展示了不同设计下 Rhythmic Pixel Region 的能耗。总体而言，**2D-In** 相比**2D-Off**降低了能耗。当 CIS 工艺节点为 130nm 时，**2D-In** 相比**2D-Off**节省了 14.5% 的能耗。当 CIS 工艺节点为 65 纳米时，这一节能效果提升至 33.4%。能耗降低的原因在于，需要通过 MIPI CIS-2 接口（图中的 MIPI）传输的数据量减少了（从全分辨率图像变为 ROI）。数据通信成本的降低是以计算（COMP）和内存访问（MEM）成本增加为代价的；由于 CIS 采用了更旧的工艺节点，这两项成本都会上升。

Ed-Gaze 的结果如图 9b所示，呈现出不同的情况。在 CIS 内部进行计算最终会比在片上系统上进行计算消耗多得多的能量。这是因为 Ed-Gaze 的通信成本较低：在传感器外系统中，通信成本起初仅占总能耗的 15.0%。因此，在传感器内部增加的计算和内存访问成本远远超过了通信成本的降低。有趣的是，65 纳米**2D-In** 比其 130nm 版本消耗更多能量，因为众所周知，65 纳米节点具有较高的漏电功耗 [20]。在 Ed-Gaze 中，由于必须始终保留帧以进行帧差，帧缓冲器必须在整个帧时间内保持开启，且不能使用电源门控。

发现 1： *CIS* 内的计算效率低于 *CIS* 外的计算。只有当 *CIS* 的工艺节点与片上系统的工艺节点差距不大，且应用以通信为主时，*CIS*内计算才能节省能耗。

表 3：功率密度。单位为 mW/mm²。

工艺节点	Rhythmic Pixel Regions			Ed-Gaze		
像素/计算	二维-离	二维-内	三维-内	二维-离	二维-内	3D-In
130纳米/22纳米	0.05	0.09	0.06	0.19	0.30	0.78
65纳米/22纳米	0.03	0.05	0.04	0.11	2.24	0.70

6.2 2D CIS 与 3D 堆叠式 CIS 的比较

我们使用 CamJ 来探索 3D 堆叠式 CIS 的优势。我们在 Rhythmic Pixel Regions和 Ed-Gaze 中使用相同的算法，并考虑另外两种 3D 配置:

- **3D-In（H/L）**：一种双层堆叠式 CIS，其中像素层采用 **H** 工艺节点制造，计算层采用先进工艺节点**L**制造。所有模数转换器后操作均在计算层中进行。
- **3D-In-STT（H/L）**：与**3D-In（H/L）**类似，但计算层中的 SRAM 被 STT-RAM 取代，我们使用 NVMExplore 对其进行建模 [55]。Rhythmic Pixel Regions 缺少 STT-RAM 的结果，因为它仅需要 2K 存储器，而 NVMExplore 不支持这种存储器。

比较 **3D-In** 和 **2D-In** 在图 9a 显示，对于RhythmicPixel Regions，3D 集成平均可将能耗降低 15.8%。这是因为数字处理单元和 SRAM 现在使用与片上系统相同的工艺节点；将计算移入传感器内部不再增加计算和存储器能耗，同时仍可享受显著的数据量减少。通过 μTSV移动数据所带来的额外成本微不足道，这是由于 μTSV 的能耗很低。

三维集成对 Ed-Gaze 的能耗降低幅度更高。对比 **3D-In** 和 **2D-In** 在图 9b中，三维堆叠平均可将能耗降低 38.5%，这是因为在 **2D-In**中，存储器能耗占总能耗的 71.3%，而使用 22nm 节点时，该能耗会显著降低。

尽管如此，**3D-In**中的存储器能耗仍占主导地位，因为如前所述，帧缓冲器无法进行电源门控，从而产生不可忽略的漏电功耗。为了进一步降低存储器能耗，我们探索了已知具有低漏电功耗的 STT-RAM。通过用 STT-RAM替代 SRAM，在 65 纳米/22 纳米和 130 纳米 /22 纳米组合下，**3D-In-STT** 相较于 **3D-In** 的整体能耗分别进一步降低了 69.1% 和 68.5%。

功率密度。虽然众所周知，三维堆叠会提高功率密度，但我们使用 CamJ 说明其对功率密度的影响取决于具体算法。我们采用保守的面积估算来获得
功率密度上限，同时将全面的面积建模留待未来工作。具体而言，我们使用像素阵列来估算模拟面积，并使用 SRAM 面积来估算数字面积。

表 3 比较了两种算法下三种传感器变体的功率密度。Rhythmic Pixel Regions 的功率密度在三种变体之间没有显著差异，因为 Rhythmic Pixel Regions 的功耗主要由通信决定，而通信受堆叠的影响不大。相比之下，在 130纳米/22纳米组合下，Ed-Gaze 通过三维堆叠使功率密度提高了一倍以上。这是因为 三**维堆叠式 In** 的面积小于 3×，比 二**维 In**。在 65纳米/22纳米组合下，

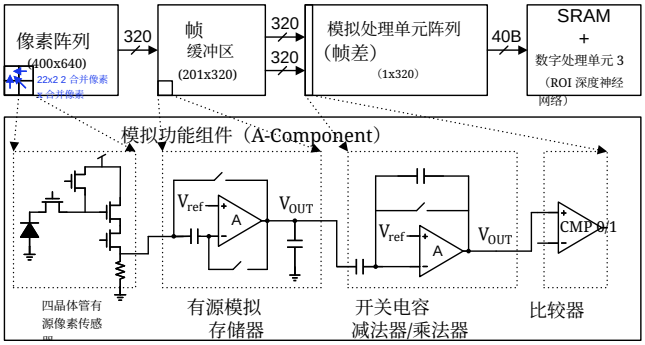


图 10：用于 Ed-Gaze 的混合信号CIS设计。我们将图 8b 移至模拟域。

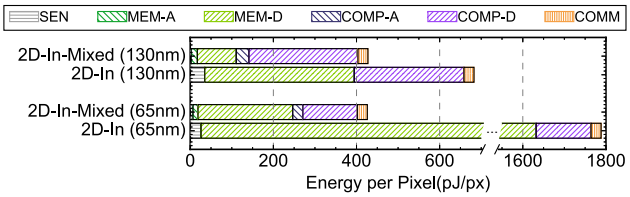


图 11：Ed-Gaze 上混合信号传感器内计算与全数字传感器内计算之间的能耗比较。COMP/MEM-D：数字计算和存储器；COMP/MEM-A：模拟计算和存储器。

二**维 In** 的功率密度更高，这是由于 65 纳米工艺的漏电功耗较高，而堆叠式设计避免了这一问题。

值得注意的是，绝对3D 堆叠式传感器的功率密度仍然很低，事实上比典型中央处理器的功率密度（最高可达 1W/mm² [14]）和图形处理器的功率密度（最高可达 0.3W/mm² [11]）低三到四个数量级。如此低的功率密度不太可能导致热热点，也不会带来散热挑战 [73]。然而，功率密度越高，热诱导噪声就越大，成像和计算质量也会越差 [36]。端到端应用优化必须考虑噪声影响，这项探索由CamJ 提供支持，我们将留待未来工作开展。

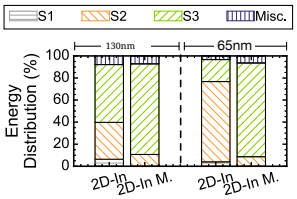
发现 2： *3D* 堆叠可节省能耗，但会增加功率密度——适用于计算密集型应用。绝对功率密度不足以造成热热点，但可能增加噪声，因此值得进一步研究。

6.3 模拟计算与数字计算的比较

我们使用 CamJ 探索传感器内模拟计算的优势。具体而言，我们以 Ed-Gaze 为案例研究，并考虑一种混合信号配置:

- **2D-In-Mixed**：一种 2D CIS，其中算法的前两个阶段（图 8b） 2×2 ——降采样和帧差分——采用模拟方式实现，而最后一个阶段（ROI 深度神经网络）则在数字域中实现。

图 10 展示了 Ed-Gaze如何映射到混合信号 CIS。在像素阵列内， 2×2 下采样通过



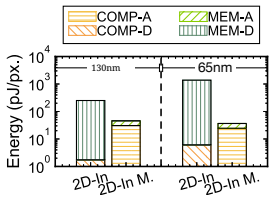


图 12：归一化能耗　图 13：前三个阶段的能耗分布（S1、S2、S3）。

像素合并（类似于图 5）。模拟帧缓冲区存储下采样后的模拟像素值，这些值由模拟处理单元阵列读取，用于帧差。每个模拟处理单元由一个开关电容减法器/乘法器和一个比较器组成，分别用于绝对值减法和帧差数字化。模拟处理单元阵列的输出进入 SRAM 阵列，此时硬件与 **2D-In** 中的相同。为进行公平比较并确保充分考虑面积开销，我们保守地将所有电容器均设为 100fF。尽管尺寸有所过大，模拟设计的面积仍比数字设计至少小 27%。

图 11比较了 **2D-In-Mixed** 和 **2D-In**。将 Ed-Gaze 算法的前几个阶段移至模拟域，可分别降低 38.8% 和 77.1% 的能耗。能耗降低来自两个方面：移除了模数转换器（SEN 降低所示），并在前两个阶段用模拟缓冲器替代 SRAM（MEM-D 降低所示）。对于 65 纳米节点，MEM-D 的降低尤为显著，因为 SRAM 的漏电功耗较高。为佐证这些结果，图 12 展示了三个阶段（S1、S2 和 S3）的归一化能耗分布。将前两个阶段移至模拟域后，S3（深度神经网络）成为主导阶段，体现了模拟处理的有效性。

有趣的是，尽管前两个阶段的计算能耗略有增加，整体能耗仍然降低了。图 13展示了前两个阶段的能耗分布。在混合信号模式下，存储器能耗有所降低，而计算能耗有所增加。这是因为，为了保持 8 位精度，运算放大器消耗了过多能量（公式 6 ）。需要注意的是，此处介绍的模拟设计采用了有源开关电容电路，是基于我们特定的实现方案。可以想见，采用不同设计会得到不同的能效结果。

发现 3： 众所周知，模拟计算能够降低模数转换器和计算能耗，但节能效果也归因于模拟存储器能耗较低，尤其是在存储器密集型应用中；许多 *CIS* 内计算用例都属于此类。

7 相关工作

第 2 节和第 5节讨论了新兴的 CIS设计。

功耗建模。 功耗/能耗建模是体系结构探索的基石。以往针对中央处理器 [7, 44,62], 图形处理器 [28, 33, 43], 和存储器 [3, 23, 55, 57] 的功耗模型推动了大量功耗/能耗优化。从根本上说，CamJ采用相同的自底向上建模方法，根据访问次数和每次访问能耗估算能耗。此外，CamJ 提供了简洁的编程接口，用于集成其他架构模拟器 [18, 19, 59, 63]和存储器建模工具 [55, 57] ，以对定制加速器和存储器进行建模。

以往的模拟功耗建模需要详细的晶体管级参数 [65]，或者基于特定模拟电路的统计模型 [41]。Lim等人 [47] 将混合信号电路分解为基本单元，并通过近似每个单元的传递函数来加速混合信号仿真。CamJ采用了类似的分解方法，但专门面向 CIS。

CIS 建模。 不存在全面的CIS 建模框架。Meta 最近的两篇论文使用一阶解析模型来估算其定制CIS 设计的能耗，即采用带有 DPS的三维堆叠 [21, 48]。它无法提供足够的灵活性来适应通用 CIS 设计和架构探索，而 CamJ 支持这些功能。

LiKamWa等人 [46] 使用空闲和活动周期/功耗提供了一个粗粒度计算成像传感器功耗模型，但未考虑硬件实现细节。相比之下，CamJ以更细的粒度对硬件进行建模，以实现更细粒度的体系结构探索。Kodukula等人 [36] 引用了典型CIS设计的粗粒度组件能耗，并构建了一个热模型。CamJ可以提供更准确的功耗/能耗建模，为这类热模型提供输入。

视觉计算优化。 近期研究探讨了在 CIS 内部进行处理以降低数据传输成本的可能性，例如 Ed-Gaze [17], Rhythmic Pixel Regions [37], Reid等人 [56], 和 SplitNets [15]。然而，所有这些工作都依赖一阶能耗模型。利用 CamJ，我们研究了 Ed-Gaze 和 Rhythmic Pixel Regions 中的算法，并量化了它们的收益（第 6节）。许多近期的视觉计算优化利用运动向量，这些运动向量可以在成像过程中自然生成，从而简化下游视觉处理 [18, 74]。探索如何利用 CamJ 将运动估计集成到 CIS 中很有意义。 CamJ 还可以与视觉计算基准测试集成 [31, 39] ， 以研究针对不同工作负载的 CIS内计算。

8 结论

CamJ 是一种经过硅验证的组件级能耗建模与架构探索框架，面向计算型CIS。它接- 受图像处理算法和硬件设计的高层描述，并支持探索架构权衡，例如 CIS内与 CIS外、二维与三维设计，以及模拟与数字处理之间的权衡。CamJ 是充满前景的传感器内视觉计算的第一步。

参考文献

[1] [未注明日期]。国际器件与系统路线图。https://irds.ieee.org/。[2] Vahideh Akhlaghi、Amir Yazdanbakhsh、Kambiz Samadi、RajeshKGupta 和 HadiEsmaeilzadeh。2018年。Snapea：用于减少深度卷积神经网络中计- 算量的预测性提前激活。载于*2018*年 *ACM/IEEE* 第*45*届年度国际计算机体系结构研讨会（*ISCA*）。IEEE，662–673。[3] RajeevBalasubramonian、Andrew B Kahng、Naveen Muralimanohar、AliShafiee和VaishnavSrinivas。2017年。CACTI 7：用于创新型片外存储器互连探索的新工具。*ACM*体系结构与代码优化汇刊（*TACO*）14，2（2017年），1–25。[4] M Bigas、Enric Cabruja、Josep Forest和Joaquim Salvi。2006年。CMOS图像传感器综述。微电子学杂志37，5（2006年），433–451。[5] Kyeongryeol Bong、Sungpill Choi、Changhyeon Kim、DonghyeonHan 和 Hoi-JunYoo。2017年。一种低功耗卷积神经网络人脸识别处理器及集成了常开人脸检测器的CIS。*IEEE*固态电路杂志53，1（2017年），115–123。

[6] Kyeongryeol Bong、Sungpill Choi、Changhyeon Kim、SanghoonKang、YouchangKim 和 Hoi-JunYoo。2017年。14.6一款0.62 mW 超低功耗卷积神经网络人脸识别处理器，以及集成始终开启式类Haar人脸检测器的CIS。在2017年 IEEE国际固态电路会议（ISSCC）上。IEEE，248–249。[7] David Brooks、VivekTiwari 和 Margaret Martonosi。2000。Wattch：用于体系结构级功耗分析和优化的框架。ACM SIGARCH《计算机体系结构新闻》28，2（2000年），83–94。[8] Weidong Cao、Yilong Zhao、Adith Boloor、YinheHan、Xuan Zhang 和Li Jiang。2022年。Neural-PIM：通过对外围电路进行神经近似实现高效的存内处理。IEEE汇刊：计算机。71，9（2022年），2142–2155。[9] Raffaele Capoccia、Assim Boukhayma 和Christian Enz。2020。模拟CMS对 CIS 读出噪声影响的实验验证。IEEE电路与系统汇刊I：常规论文 67，3（2020年），774–784。[10] Karthik Chandrasekar、Christian Weis、Yonghui Li、Benny Akesson、Norbert Wehn和 Kees Goossens。2012。DRAMPower：开源 DRAM 功耗与能耗估算工具。URL：http://www.drampower.info22（2012年）。[11] John YChen。2009年。GPU 技术趋势与未来需求。在2009年IEEE国际电子器件会议（IEDM）上。IEEE，1–6。[12] Chih-Chi Cheng、Chia-Hua Lin、Chung-Te Li 和 Liang-GeeChen。2008。iVisual：一种智能视觉传感器片上系统，配备 2790帧率 CMOS图像传感器和205 GOPS/W视觉处理器。IEEE固态电路杂志 44，1（2008），127–135。[13] Jaihyuk Choi、SungjaeLee、YoungdooSon 和 Soo YounKim。2020。使用模拟轻量级卷积神经网络设计始终开启式图像传感器。Sensors 20，11（2020年），3101。[14] Andrew Danowitz、Kyle Kelley、James Mao、JohnP Stevenson 和 Mark Horowitz。2012。CPU DB：记录微处理器历史。Commun. ACM55，4（2012年），55–63。[15] Xin Dong、Barbara De Salvo、Meng Li、Chiao Liu、Zhongnan Qu、HTKung和 Ziyun Li。2022年。SplitNets：面向头戴式系统高效分布式计算的神经网络架构设计。在 IEEE/CVF计算机视觉与模式识别会议论文集。12559–12569。[16] Ryoji Eki、Satoshi Yamada、Hiroyuki Ozawa、Hitoshi Kai、Kazuyuki Okuike、Hareesh Gowtham、HidetomoNakanishi、Edan Almog、YoelLivne、Gadi Yuval等。2021年。9.6 一款 1/2.3英寸、1230万像素、片上集成 4.97TOPS/WCNN 处理器的背照式堆叠式CMOS 图像传感器。在 2021年IEEE国际固态电路会议（ISSCC），第64 卷。IEEE，154–156。[17] Yu Feng、Nathan Goulding-Hotta、Asif Khan、Hans Reyserhove 和YuhaoZhu。2022年。利用事件驱动的眼睛分割实现实时注视跟踪。在 2022年IEEE虚拟现实与3D用户界面会议（VR）上。IEEE，399–408。[18] YuFeng、PaulWhatmough和 Yuhao Zhu。2019。Asv：加速立体视觉系统。在第 52届IEEE/ACM微体系结构国际研讨会论文集。643–656。[19] Mingyu Gao、Jing Pu、Xuan Yang、Mark Horowitz 和Christos Kozyrakis。2017年。Tetris：利用3D存储器实现可扩展且高效的神经网络加速。在第二十二届编程语言与操作系统体系结构支持国际会议论文集。751–764。[20] Georges Gielen 和 Wim Dehaene。2005年。65nmCMOS 中的模拟与数字电路设计：道路的尽头？在欧洲设计、自动化与测试会议上。IEEE，37–42。[21] Jorge Gomez、Saavan Patel、Syed Shakib Sarwar、Ziyun Li、RaffaeleCapoccia、Zhao Wang、Reid Pinkham、Andrew Berkovich、Tsung-HsunTsai、BarbaraDeSalvo 等。2022年。面向增强现实/虚拟现实设备的分布式传感器上计算系统：用于能耗估算的半解析仿真框架。arXiv预印本 arXiv:2203.07474（2022年）。[22] Camare Working Group。2021年。MIPI白皮书：MIPI汽车 SerDes 解决方案（MASS）入门指南。https://www.mipi.org/introductory- guide-to-mass[23] Matthew R Guthaus、JamesE Stine、SamiraAtaei、BrianChen、Bin Wu和MehediSarwar。2016年。OpenRAM：开源存储器编译器。在2016年 IEEE/ACM计算机辅助设计国际会议（ICCAD）上。IEEE，1–6。[24] Song Han、Xingyu Liu、Huizi Mao、Jing Pu、Ardavan Pedram、Mark AHorowitz和WilliamJDally。2016年。EIE：压缩深度神经网络上的高效推理引擎。ACM SIGARCH《计算机体系结构新闻》44，3（2016年），243–254。[25] Tsutomu Haruta、Tsutomu Nakajima、Jun Hashizume、Taku Umebayashi、Hiroshi Takahashi、Kazuo Taniguchi、Masami Kuroda、Hiroshi Sumihiro、KojiEnoki、Takatsugu Yamasaki等。2017年。4.6一款配备 DRAM的1/2.3 英寸 2000万像素三层堆叠式CMOS 图像传感器。在2017年IEEE国际固态电路会议（ISSCC）上。IEEE，76–77。[26] James Hegarty、John Brunhaver、Zachary DeVito、Jonathan Ragan-Kelley、Noy Cohen、Steven Bell、Artem Vasilyev、Mark Horowitz 和Pat Hanrahan。2014。Darkroom：将高级图像处理代码编译为硬件流水线。ACM图形学汇刊 33，4（2014年），144–1。

[27] Tomoki Hirata、Hironobu Murata、Hideaki Matsuda、Yojiro Tezuka 和 Shiro Tsunai。2021年。7.8 一款用于计算成像和自适应动态范围控制的 1 英寸 1700 万像素 1000 帧率块控制编码曝光背照式堆叠式 CMOS 图像传感器。载于*2021*年 *IEEE* 国际固态电路会议（*ISSCC*），第 64 卷。IEEE，120–122。[28] Sunpyo Hong 和 Hyesoon Kim。2010年。一种集成式 GPU 功耗与性能模型。载于第*37*届计算机体系结构年度国际研讨会论文集。280–289。[29] Tzu-Hsiang Hsu、Guan-Cheng Chen、Yi-Ren Chen、Chung-Chuan Lo、Ren-Shuo Liu、Meng-Fan Chang、Kea-Tiong Tang 和 Chih-Cheng Hsieh。2022年。一款采用微型卷积神经网络和可编程权重、基于混合模式传感器内处理技术的 0.8 V 智能视觉传感器，用于图像分类。载于*2022*年 *IEEE*国际固态电路会议（*ISSCC*），第 65 卷。IEEE，1–3。[30] Tzu-Hsiang Hsu、Yi-Ren Chen、Ren-Shuo Liu、Chung-Chuan Lo、Kea-Tiong Tang、Meng-Fan Chang 和 Chih-Cheng Hsieh。2020年。一种采用可编程内核进行特征提取的 0.5 V 实时计算型 CMOS 图像传感器。*IEEE*固态电路杂志 56, 5（2020年），1588–1596。[31] Muhammad Huzaifa、Rishi Desai、Samuel Grayson、Xutao Jiang、Ying Jing、Jae Lee、Fang Lu、Yihan Pang、Joseph Ravichandran、Finn Sinclair 等。2021年。ILLIXR：实现端到端扩展现实研究。载于*2021*年 *IEEE*工作负载表征国际研讨会（*IISWC*）。IEEE，24–38。[32] Paul G Jespers。2010年。*gm/ID* 方法论：低电压模拟 *CMOS* 电路的尺寸设计工具。Springer。[33] Vijay Kandiah、Scott Peverelle、Mahmoud Khairy、Junrui Pan、Amogh Manjunath、Timothy G Rogers、Tor M Aamodt 和 Nikos Hardavellas。2021年。AccelWattch：现代 GPU 的功耗建模框架。载于*MICRO-54*：第*54*届 *IEEE/ACM* 微体系结构年度国际研讨会。738–753。[34] Amandeep Kaur、Deepak Mishra、KM Amogh 和 Mukul Sarkar。2020年。利用累积空间梯度实现 CMOS 图像传感器阵列内的压缩采集。*IEEE* 视频技术电路与系统汇刊 31, 2（2020年），523–532。[35] Seong-Jin Kim、Kwang-Hyun Lee、Sang-Wook Han 和 Euisik Yoon。2005年。一款具有自适应列并行处理器的 200×160 像素 CMOS 指纹识别片上系统。载于*ISSCC*。*2005*年 *IEEE*国际技术论文集。*2005*年固态电路会议。IEEE，250–596。[36] Venkatesh Kodukula、Saad Katrawala、Britton Jones、Carole-Jean Wu 和 Robert LiKamWa。2021年。近传感器处理的动态温度管理，实现高能效高保真成像。*Sensors* 21, 3（2021年），926。[37] Venkatesh Kodukula、Alexander Shearer、Van Nguyen、Srinivas Lingutla、Yifei Liu 和 Robert LiKamWa。2021年。Rhythmic Pixel Regions：面向低功耗高精度视觉计算的多分辨率视觉感知系统。载于第*26*届 *ACM* 编程语言与操作系统体系结构支持国际会议论文集。573–586。[38] Oichi Kumagai、Atsumi Niwa、Katsuhiko Hanzawa、Hidetaka Kato、Shinichiro Futami、Toshio Ohyama、Tsutomu Imoto、Masahiko Nakamizo、Hirotaka Murakami、Tatsuki Nishino 等。2018年。一款 1/4 英寸、390 万像素、低功耗事件驱动背照式堆叠式 CMOS 图像传感器。载于*2018*年 *IEEE*国际固态电路会议（*ISSCC*）。IEEE，86–88。[39] Hyoukjun Kwon、Krishnakumar Nair、Jamin Seo、Jason Yik、Debabrata Mohapatra、Dongyuan Zhan、Jinook Song、Peter Capak、Peizhao Zhang、Peter Vajda 等。2022年。XRBench：面向元宇宙的扩展现实（XR）机器学习基准套件。*arXiv* 预印本 *arXiv:2211.08675*（2022年）。[40] Minho Kwon、Seunghyun Lim、Hyeokjong Lee、Il-Seon Ha、Moo-Young Kim、Il-Jin Seo、Suho Lee、Yongsuk Choi、Kyunghoon Kim、Hansoo Lee 等。2020年。一款低功耗 65/14 nm 堆叠式 CMOS 图像传感器。载于*2020*年 *IEEE*电路与系统国际研讨会（*ISCAS*）。IEEE，1–4。[41] E. Lauwers 和 G. Gielen。2002年。用于集成系统体系结构探索的模拟电路功耗估算方法。*IEEE*超大规模集成电路系统汇刊 10, 2（2002年），155–162。[42] Edward H. Lee 和 S. Simon Wong。2017年。面向近似计算的无源开关电容矩阵乘法器分析与设计。*IEEE*固态电路杂志 52, 1（2017年），261–271。[43] Jingwen Leng、Tayler Hetherington、Ahmed ElTantawy、Syed Gilani、Nam Sung Kim、Tor M Aamodt 和 Vijay Janapa Reddi。2013年。GPUWattch：实现通用 GPU 的能效优化。*ACM SIGARCH*《计算机体系结构新闻》41, 3（2013年），487–498。[44] Sheng Li、Jung Ho Ahn、Richard D Strong、Jay B Brockman、Dean M Tullsen 和 Norman P Jouppi。2009年。McPAT：面向多核与众核体系结构的集成式功耗、面积与时序建模框架。载于第*42*届 *IEEE/ACM* 微体系结构年度国际研讨会论文集。469–480。[45] Robert LiKamWa、Yunhui Hou、Julian Gao、Mia Polansky 和 Lin Zhong。2016年。Redeye：面向连续移动视觉的模拟卷积神经网络图像传感器架构。*ACM SIGARCH*《计算机体系结构新闻》44, 3（2016年），255–266。[46] Robert LiKamWa、Bodhi Priyantha、Matthai Philipose、Lin Zhong 和 Paramvir Bahl。2013年。面向连续移动视觉的图像感知能耗表征与优化。载于第*11*届移动系统、应用与服务年度国际会议论文集。69–82。

[47] Byong Chan Lim 和 Mark Horowitz。2019年。模拟模型模板库：简化芯片级混合信号设计验证。《IEEE超大规模集成电路系统汇刊》27，1（2019年），193–204。[48] Chiao Liu、Andrew Berkovich、Song Chen、Hans Reyserhove、Syed Shakib Sarwar 和 Tsung-Hsun Tsai。2019年。智能视觉系统——将人机界面引入增强现实/虚拟现实。在《2019年IEEE国际电子器件会议（IEDM）》。IEEE，10–5。[49] Chiao Liu、Song Chen、Tsung-Hsun Tsai、Barbara De Salvo 和 Jorge Gomez。2022年。增强现实——图像传感器与计算系统的新前沿。在《2022年IEEE国际固态电路会议（ISSCC）》，第65卷。IEEE，426–428。[50] Tianrui Ma、Weidong Cao、Fei Qiao、Ayan Chakrabarti 和 Xuan Zhang。2022年。HOGEye：基于RRAM的堆叠式图像传感器中HOG特征提取的神经近似。在《ACM/IEEE低功耗电子与设计国际研讨会论文集》。1–6。[51] Sparsh Mittal、Rujia Wang 和 Jeffrey Vetter。2017年。DESTINY：具备3D和多级单元存储器建模能力的综合工具。《低功耗电子与应用杂志》7，3（2017年），23。[52] Hirotaka Murakami、Eric Bohannon、John Childs、Grace Gui、Eric Moule、Katsuhiko Hanzawa、Tomofumi Koda、Chiaki Takano、Toshimasa Shimizu、Yuki Takizawa 等。2022年。一款用于计算机视觉的490万像素可编程分辨率多用途CMOS图像传感器。在《2022年IEEE国际固态电路会议（ISSCC）》，第65卷。IEEE，104–106。[53] B. Murmann。 [无日期]。ADC性能调查（1997—2022年）。http://web.stanford. edu/~murmann/adcsurvey.html。[54] Chanmin Park、Wenda Zhao、Injun Park、Nan Sun 和 Youngcheol Chae。2021年。一款51-pJ/像素、33.7-dB峰值信噪比的压缩式CMOS图像传感器，采用列并行单次压缩感测。《IEEE固态电路杂志》56，8 2021 2503 2515 ( )，–。[55] Lillian Pentecost、Alexander Hankin、Marco Donato、Mark Hempstead、Gu-Yeon Wei 和 David Brooks。2022年。NVMExplorer：嵌入式非易失性存储器跨堆栈比较框架。在《2022年IEEE高性能计算机体系结构国际研讨会（HPCA）》。IEEE，938–956。[56] Reid Pinkham、Tanner Schmidt 和 Andrew Berkovich。2020年。面向高速成像的算法感知神经网络图像压缩。在《2020年IEEE人工智能与虚拟现实国际会议（AIVR）》。IEEE，196–199。[57] Matt Poremba、Sparsh Mittal、Dong Li、Jeffrey S Vetter 和 Yuan Xie。2015年。Destiny：用于新兴3D非易失性存储器和eDRAM缓存建模的工具。在《2015年欧洲设计、自动化与测试会议及展览（DATE）》。IEEE，1543–1546。[58] Wajahat Qadeer、Rehan Hameed、Ofer Shacham、Preethi Venkatesan、Christos Kozyrakis 和 Mark A Horowitz。2013年。卷积引擎：平衡专用计算的效率与灵活性。在《第40届IEEE年度计算机体系结构国际研讨会论文集》。[59] Ananda Samajdar、Yuhao Zhu、Paul Whatmough、Matthew Mattina 和 Tushar Krishna。2018年。Scale-sim：脉动式CNN加速器模拟器。arXiv预印本 arXiv:1811.02883（2018年）。[60] Satyabrata Sarangi 和 Bevan Baas。2021 年。DeepScaleTool：用于准确估算深亚微米时代技术缩放的工具。在《2021年IEEE电路与系统国际研讨会（ISCAS）》。IEEE，1–5。

[61] Min-Woong Seo、Myunglae Chu、Hyun-Yong Jung、Suksan Kim、Jiyoun Song、Junan Lee、Sung-Yong Kim、Jongyeon Lee、Sung-Jae Byun、Daehee Bae、Minkyung Kim、Gwi-Deok Lee、Heesung Shim、Changyong Um、Changhwa Kim、In-Gyu Baek、Doowon Kwon、Hongki Kim、Hyuksoon Choi、Jonghyun Go、JungChak Ahn、Jaekyu Lee、Changrok Moon、Kyupil Lee 和 Hyoung-Sub Kim。2021年。一款具备像素级模数转换器和像素内存储器的 2.6 e-r ms 低随机噪声、116.2 mW 低功耗 2-Mp 全局快门CMOS图像传感器。载于*2021*年超大规模集成电路研讨会。1–2。[62] Yakun Sophia Shao 和 David Brooks。2013年。英特尔至强融核处理器的能耗表征与指令级能耗模型。载于国际低功耗电子与设计研讨会（*ISLPED*）。IEEE，389–394。[63] Yakun Sophia Shao、Brandon Reagen、Gu-Yeon Wei 和 David Brooks。2014年。Aladdin：一款RTL前功耗与性能加速器模拟器，可支持对定制化架构进行大规模设计空间探索。载于*2014*年第*41*届*ACM/IEEE*国际计算机体系结构研讨会（*ISCA*）。IEEE，97–108。[64] Aaron Stillmaker 和 Bevan Baas。2017年。用于准确预测 180 nm 至 7 nm CMOS器件性能的缩放方程。集成58（2017年），74–81。[65] Christer Svensson 和 J Jacob Wikner。2010年。模拟电路的功耗：教程。模拟集成电路与信号处理65，2（2010年），171–184。[66] Albert Theuwissen。2021年。1.4 画面中所见并非全部*，未来更是如此。载于*2021*年*IEEE*国际固态电路会议（*ISSCC*），第64卷。IEEE，30–35。[67] H. Tsugawa、H. Takahashi、R. Nakamura、T. Umebayashi、T. Ogita、H. Okano、K. Iwase、H. Kawashima、T. Yamasaki、D. Yoneyama、J. Hashizume、T. Nakajima、K. Murata、Y. Kanaishi、K. Ikeda、K. Tatani、T. Nagano、H. Nakayama、T. Haruta 和 T. Nomoto。2017年。像素/DRAM/逻辑三层堆叠CMOS图像传感器技术。载于*2017*年*IEEE*国际电子器件会议（*IEDM*）。3.2.1–3.2.4。[68] Paul N Whatmough、Chuteng Zhou、Patrick Hansen、Shreyas Kolala Venkatara- manaiah、Jae-sun Seo 和 Matthew Mattina。2019年。Fixynn：通过迁移学习实现移动计算机视觉的高效硬件。*arXiv*预印本 *arXiv:1902.11128*（2019年）。[69] Yuan Xie 和 Jishen Zhao。2015年。芯片堆叠架构。计算机体系结构综合讲座10，2（2015年），1–127。[70] Han Xu、Ningchao Lin、Li Luo、Qi Wei、Runsheng Wang、Cheng Zhuo、Xunzhao Yin、Fei Qiao 和 Huazhong Yang。2021年。Senputing：一款超低功耗常开视觉感知芯片，实现感知与计算深度融合。*IEEE*电路与系统汇刊*I*：常规论文69，1（2021年），232–243。[71] Minhao Yang、Shih-Chii Liu 和 Tobi Delbruck。2015年。一款动态视觉传感器，具有 1% 时间对比度灵敏度以及用于事件编码的像素内异步增量调制器。*IEEE*固态电路杂志50，9（2015年），2149–2160。[72] Christopher Young、Alex Omid-Zohoor、Pedram Lajevardi 和 Boris Murmann。2019年。一款用于常开式目标检测、具备多尺度读出的数据压缩型 1.5/2.75 位对数梯度 QVGA 图像传感器。*IEEE*固态电路杂志54，11（2019年），2932–2946。[73] Ying-Ju Yu 和 Carole-Jean Wu。2018年。设计温度模型以了解便携式计算平台面临的热挑战。载于*2018*年第*17*届*IEEE*电子系统热学与热机械现象跨学会会议（*ITherm*）。IEEE，992–999。[74] Yuhao Zhu、Anand Samajdar、Matthew Mattina 和 Paul Whatmough。2018年。Euphrates：面向低功耗移动连续视觉的算法与片上系统协同设计。载于*2018*年第*45*届*ACM/IEEE*年度国际计算机体系结构研讨会（*ISCA*）。IEEE计算机学会，547–560。