Xel-FPGAs：面向基于现场可编程门阵列的系统中近似加速器的端到端自动化探索框架

Bharath Srinivas Prabakaran*, §、Vojtech Mrazek†, §、Zdenek Vasicek †、Lukas Sekanina†、Muhammad Shafique‡
*计算机工程研究所，维也纳工业大学（TU Wien），奥地利
bharath.prabakaran@tuwien.ac.at
†信息技术学院，IT4Innovations卓越中心，布尔诺理工大学，捷克共和国{mrazek, vasicek, sekanina}@fit.vutbr.cz
‡eBrain实验室，纽约大学阿布扎比分校（NYUAD）工程系，阿拉伯联合酋长国（UAE）
muhammad.shafique@nyu.edu

arXiv:2303.04734v3 [cs.AR] 8 Aug 2023

摘要—近似电路和加速器的生成与探索一直是实现能效和/或性能提升的重要研究领域。这项研究主要聚焦于专用集成电路，但由于两者固有的架构差异，在部署到基于FPGA的加速器系统时未能取得类似的收益。在本研究中，我们提出了一种新型框架Xel-FPGAs，该框架利用统计模型或机器学习模型，根据目标应用的简单 RTL 描述，有效探索最先进的基于ASIC的近似电路的架构空间，使其适用于基于FPGA的系统。我们还采用分层搜索策略，在一个多阶段应用中评估了该框架的可扩展性。与默认的综合、布局和布线方法相比，Xel-FPGAs 框架可将探索时间最多缩短 95%，同时针对给定应用识别出一组优于当前最先进水平的帕累托最优设计。完整框架已开源，可在线获取：https://github.com/ehw-fit/xel-fpgas。

索引词—**近似计算、加速器、现场可编程门阵列、专用集成电路、算术单元、回归、模型、综合。**

I. 引言

现场可编程门阵列（现场可编程门阵列）是一种颇具吸引力的计算平台，适用于各种应用，尤其是那些需要随时更新和/或修改的应用，这得益于其部分运行时重配置能力，即现场可编程性。自 1984 [1],以来，与专用集成电路相比，现场可编程门阵列凭借更短的上市时间和更低的原型成本，获得了广泛应用。除了部署于嵌入式应用场景和信息物理系统中，现场可编程门阵列也部署在服务器和高性能计算集群中，为复杂的计算密集型算法提供按需性能加速 [2]。这些可重构平台包括用于低功耗ARM处理器内核的硬核IP（集成电路实现）和视频编解码器等常用硬件加速器，以及用于实现定制加速器 [3]。凭借这些广泛的能力，现场可编程门阵列可作为嵌入式应用场景的可编程片上系统（可编程片上系统），也可作为按需计算加速服务。然而，与专用集成电路相比，现场可编程门阵列往往功耗更高、吞吐量更低。

除了采用动态电压与频率调节（DVFS） [4] 和时钟门控等传统的降功耗方法外， [5], 近似计算似乎也非常适合提升系统的能效。对系统计算单元进行近似，其核心理念源于这样一个事实：这些系统及其算法本质上具有容错性，即不会导致输出质量出现“显著”下降 [6]。这种固有的容错特性体现在涵盖多个领域的应用中，例如识别、挖掘和综合，以及数据分析 [7], 音频、图像或视频处理 [8], 语音和图像识别 [9], 等。这些应用所表现出的容错性可归因于四个关键因素，即：(1) 冗余输入数据；(2) 所用算法能够隐式地衰减误差；(3) 输出质量的差异在用户层面难以察觉；以及 (4) 不存在唯一的标准输出。学术界和工业界的众多研究广泛利用了这些因素，针对硬件和软件开展工作，以获得功耗、延迟或能耗方面的收益 [10]–[22]。

最先进水平的研究主要倾向于聚焦于生成近似电路（AC）和/或加速器，以便为基于专用集成电路的系统带来功耗、延迟和/或能耗方面的优势。过去的多项研究表明，为基于专用集成电路的系统开发的近似计算原理和技术，在应用于基于FPGA的系统时往往会带来不同的优势 [12]–[20]。例如，为专用集成电路开发的近似算术单元在专用集成电路平台上综合时，能耗最多可降低 70%；而同样的设计在现场可编程门阵列平台上综合时，节省的能耗微乎其微，有时甚至会增加 [12]。Ullah 等人 [14] 提出了一种

§这两位作者对本项工作作出了同等贡献。

系统化生成多种近似乘法器架构的方法，这些架构部署在现场可编程门阵列平台上时可带来类似的优势。该方法还包括一个小规模设计空间探索阶段，用于识别适用于该应用的近似加速器设计。另一方面，Prabakaran 等人 [15] 展示了探索基于专用集成电路的近似算术电路的庞大架构空间所带来的优势，从中识别出一组适用于目标现场可编程门阵列平台的帕累托最优[1] 设计。随后，他们利用大规模自动化近似加速器探索策略，为某个应用识别近乎最优的近似加速器。然而，这些研究并未全面探索利用多个算术近似电路构建近似加速器所涉及的庞大架构空间，因此探索过程中可能会忽略一些明显位于该应用帕累托前沿上的权衡方案。

A. 动机分析

为进一步说明专用集成电路与现场可编程门阵列之间的差异，我们探究了高斯滤波器的近似加速器变体。该滤波器由九个 8 位乘法器和八个 16 位加法器组成。这些近似加速器变体采用进化近似算术电路库 [23]中的帕累托最优近似电路，以及针对现场可编程门阵列识别出的帕累托最优近似电路 [15]构建。通过穷举这些设计的各种组合得到的近似加速器，使用 Xilinx xc7vx485tffg1157-1 现场可编程门阵列的 Vivado 2017。2 工具链进行综合。我们还禁用了现场可编程门阵列上的数字信号处理模块，以确保加速器映射到可重构逻辑上。我们使用这些变体的软件模型，对一组输入信号样本进行分析，以平均峰值信噪比（PSNR）衡量其质量。图 1 展示了这些实验结果，据此我们得出以下关键观察：

(1) 对于基于专用集成电路的平台而言，帕累托最优的近似加速器，在现场可编程门阵列上综合后未必仍是帕累托最优，尤其是在考虑功耗和延迟等不同硬件参数时。(2) 综合一组由 1000 个随机设计组成的近似加速器子集需要 2.8 天。这一开销主要归因于

Vivado 工具链的高层映射算法。(3) 尽管已有不少研究工作致力于为现场可编程门阵列设计定制的近似电路和加速器 [13], [18], 但与近似电路的自动生成和探索相比，它们在实现类似硬件成本改进方面的效果并不显著，正如 [15], [23]所进一步证实。

基于这些观察，我们确定了以下研究挑战：

[1]帕累托最优是指通过给定方法获得的最佳解决方案。

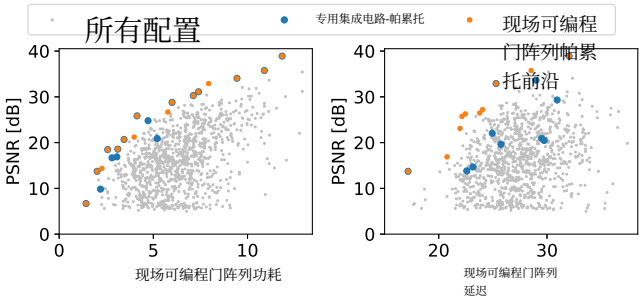


图 1：帕累托最优近似加速器分析。

•鉴于探索简单高斯滤波器的大量近似加速器变体所需的时间，对于更大型应用的探索可能需要数月甚至数年。– 我们如何利用统计模型或机器学习模型，高效缩短基于现场可编程门阵列的近似加速器的探索时间？– 除了从现场可编程门阵列综合工具中获得的传统硬件需求外，是否可以使用非常规特征和工具来缩短探索时间？

•目前尚无一种工具能够针对给定应用识别适用于现场可编程门阵列的近似加速器变体。– 我们能否构建一个端到端框架，针对应用的硬件和软件模型，为现场可编程门阵列生成并探索近似加速器变体？为应对这些研究挑战，我们提出以下创新贡献：

•一种端到端自动化框架 *Xel-FPGAs*，用于针对目标应用，系统地探索基于现场可编程门阵列的近似加速器变体的架构空间。

•我们的框架采用统计学习模型，在无需综合每个单独设计的情况下遍历适用于现场可编程门阵列的近似加速器架构空间。

• 为了进一步缩短近似加速器架构空间的探索时间，我们使用 ABC工具 [24] 来估算该加速器在目标现场可编程门阵列上的查找表数量、功耗和延迟。

•完整的 *Xel-FPGAs* 框架（包括本文讨论的近似加速器）均为开源项目，可通过以下网址在线获取，https://github.com/ehw-fit/xel-fpgas以促进该领域的广泛应用和研究。

II. 自动化框架

图 2 展示了所提出的 Xel-FPGAs 框架概览。该框架包含三个关键阶段，即模型训练、架构探索和最终评估。在第一阶段，我们训练并测试各种统计学习模型，这些模型可用于估算给定硬件加速器模块的参数。接下来，利用这些模型估算架构空间中所有近似加速器的潜在输出质量和硬件需求。最后，选定的加速器设计将

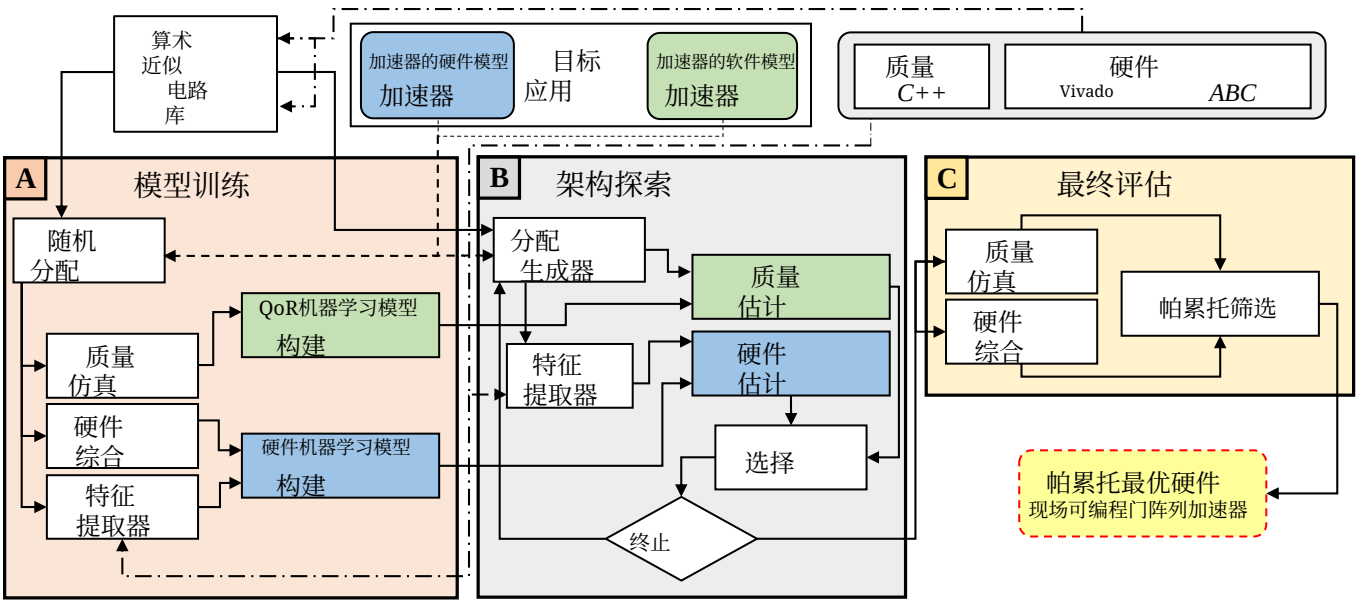


图 2：Xel-FPGAs 框架概览。

分别通过仿真和综合其软件模型与硬件模型来重新评估。

A. 模型训练

我们从一组可替代目标算法中算术单元的近似电路开始。为评估我们框架的有效性，我们采用广范围进化近似组件库 [23],，其中包含加法器和乘法器。使用其他加法器和乘法器库 [14] 与使用该库相互独立，可轻松集成到 *Xel-FPGAs*中。

当前的架构空间探索流程要求系统设计人员生成所有可能的近似加速器变体，并使用现场可编程门阵列综合工具对其进行综合，以确定其硬件需求。为了确定这些变体能够达到的结果质量（QoR），还需要使用软件模型开展类似的探索策略。如前所述，这可能导致一个穷尽式的架构探索阶段，持续数月甚至数年。为绕过这一瓶颈，我们提出训练并部署统计学习模型，用于估算加速器变体的硬件需求和结果质量能力，从而大幅缩短综合和评估每个独立近似加速器所需的时间。先前的研究 [15]评估了用于近似电路的类似模型，并发现与随机梯度下降、自适应提升甚至多层感知机（MLP）等其他方法相比，统计回归模型的相关性最高。我们主要关注轻量级的统计回归 模型，而非较大的机器学习模型，以减少模型训练和架构空间探索所需的总时间。

图 3 展示了我们在工作中探索的模型训练流程，旨在确定能够快速、高效地探索近似加速器架构空间的最佳方案。(A) 表示前文讨论的默认穷举探索方法，该方法要求使用我们的 FPGA工具链对每个近似加速器进行综合。由于这种方法并不可行，我们提出通过综合库中的算术近似电路来生成特征集（由功耗、查找表和延迟组成）。然后利用该特征集估算 1000 个随机近似加速器的硬件参数。随后需要使用 FPGA工具链综合这些随机近似加速器，以便模型学习特征与其期望值之间的关系。该流程如 (B) 所示。然而，使用 Vivado 工具链综合完整的近似电路库非常耗时（见第 IV 节）。因此，我们提出用 ABC 工具 [24] 取代 Vivado 工具链，以估算模型学习所需的特征。该流程如 (C) 所示。对此版本的另一种改进是加入更多特征，以期提升统计模型的准确性。如 (D) 所示，该变体增加了一个步骤：使用 ABC 提取近似加速器变体的特征，并将其与近似加速器的 Vivado 综合值一起输入模型训练阶段。最后，为了评估所有可能的组合，我们还测试了另外两种情形：其一 (i)，算术近似电路仍采用原有的特征提取流程和 Vivado 工具链，而使用 ABC 工具提取近似加速器的特征（如 (E) 所示）；其二 (ii)，不估算算术近似电路的特征，仅使用 ABC 工具提取的近似加速器特征来训练统计模型（如 (F) 所示）。我们比较了这些流程的实验结果

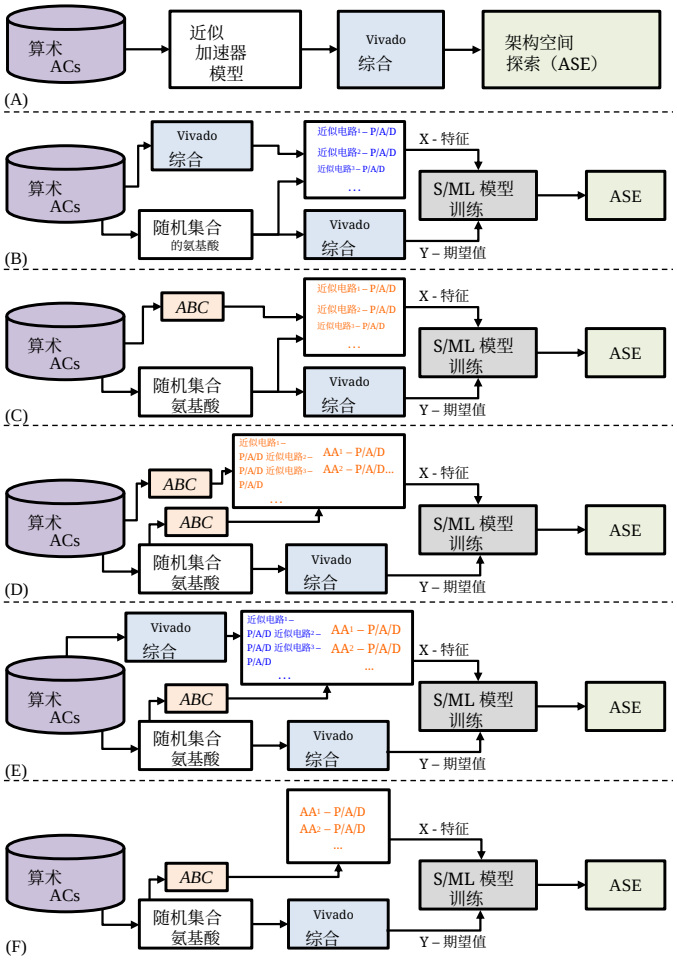


Fig. 3：可能的模型训练流程概览

并对其进行了分析，确定配置 (D) 最适合我们的框架（见第 IV 节）。

B. 架构探索

现在我们已经训练好了用于估算应用级 QoR 及其硬件需求的统计模型，因此可以高效地探索近似加速器变体的架构空间。尽管部署统计模型可以减少开销时间，我们仍研究了采用若干最先进的进化算法，通过选择需要评估的特定模型，进一步缩短架构空间探索所需的时间。例如，我们可以从一组 1000 个随机近似加速器变体开始，将其送入 ABC 工具以确定其特征。利用这些信息，我们可以估算近似加速器的硬件需求及其 QoR。随后，将 200 个表现最佳的帕累托最优及近似最优设计反馈给进化算法，作为父代集合。利用该父代集合，我们得到下一代 1000 个待评估的近似加速器，并从中选出 200 个最佳设计，作为再下一代的父代集合。重复这一过程，直到产生 1000 代近似加速器变体。我们将最后一代中得到的 200 个最佳近似加速器

线。

变体视为最终的帕累托最优设计集合，并观察到它们能够实现最优结果。与评估这些设计所需的时间相比，继续生成和探索近似加速器变体只能取得次优结果或微不足道的改进。因此，我们结束探索阶段，进入框架的最后阶段，在此阶段评估其效能。

进化算法：除了研究常用于探索设计空间的 NSGA-II [25] 遗传算法外，我们还决定评估另外两种基于局部搜索的方法，以考察它们在我们的设计空间中的适用性。受 [22],中展示的结果启发，我们选择实现并评估爬山算法 [26]，，但最终发现该算法在探索我们的设计空间时效率极低。该算法在每个迭代阶段只能识别出 1 个候选解，然后将其输入我们的统计模型。然而，这些模型的构建方式大量利用了向量化和并行性，而逐特征进行评估的速度较慢，这与我们的框架旨在减少探索时间这一最终目标相悖，即。因此，我们选择探索进化策略（ES）$\mu+\lambda$，在该策略中，λ 新的候选设计由一组最佳个体生成（μ）。与依赖“遗传机制”和复杂“交叉”机制来确定下一代设计的 NSGA-II 不同，ES 仅依赖于对最佳个体进行小幅随机变化。我们在时间限制内迭代遍历给定帕累托前沿中的每个设计，以生成可用于提取新帕累托前沿的设计。这个新前沿中的设计将作为下一代设计的基础。

我们的多目标进化算法如算法1所述。该算法以一组随机父代（μ）为起点，并通过对每个父代进行随机变异来生成 λ 个新候选解。我们利用统计模型并行估计新候选解的功耗和结果质量，以确定该设计是否更新了帕累托前沿 P。如果有新设计加入 P，该设计就会成为新的父代，随后可用于生成其他新候选解。否则，我们会迭代地从现有帕累托前沿 P 中选择一个新候选解，以避免陷入停滞。

C. 最终评估

获得最终的近似加速器变体集合后，我们对其进行综合和评估，以确定其准确的硬件需求和结果质量（QoR）值，而非采用模型得出的估算值。随后，我们利用这些数值构建帕累托前沿，作为目标应用的最终帕累托最优变体集合。系统设计人员随后可以分析这些设计在质量与硬件方面的权衡，并根据系统规格，为该应用部署合适的近似加速器变体。

算法1 我们的进化多目标优化

输入： RL – 库集合，$RL = \{RL_1, RL_2, \cdots, RL_n\}$, M_{HW} – 硬件成本模型，M_{QoR} – 质量模型
输出： 帕累托集 $P \subseteq RL_1 \times RL_2 \times \cdots \times RL_n$
函数 EVOLUTIONARYSTRATEGYMOO(RL, M_{QoR}, M_C)
  $ParentSet \leftarrow$
  P选R择随机($RL_1 \times RL_2 \times \cdots \times RL_n$) ▷ μ父代
  $P \leftarrow \emptyset$
  当 $\neg TerminationCondition$ **时执行**
    对所有 $Parent \in ParentSet$ **执行**
      $C \leftarrow$ {变异($Parent$) $\cdots$} ▷ λ 新候选解
      $e_{QoR} \leftarrow M_{QoR}(C)$ ▷ 估算 C_i的质量
      $e_{HW} \leftarrow M_{HW}(C)$ ▷ 估算 C_i的硬件成本
      对于所有 $c_i \in C$, **执行**
        如果 帕累托插入(P,(e_{QoR}, e_{HW}) C) **则**
          $Parent \leftarrow C$ ▷替换当前父代
        结束条件判断
      结束循环
      如果 $Parent \notin C$ **则**
        $Parent \leftarrow$ 从(P)中随机挑选 ▷防止停滞
      结束条件判断
    结束 for 循环
  结束 while 循环
  返回 P
结束函数

III. 实验设置

本研究所用的近似算术电路库为开源库，可立即获取[2,3]；其中包含近似电路（AC）的硬件模型（Verilog）和软件模型（C++）。这些模型可组合部署于目标应用中，从而生成多种可能的近似加速器变体。在不失一般性的前提下，本研究选取 1000 个随机近似加速器变体作为样本集，用于训练我们的统计回归模型。由于我们的框架遵循图 3 中的流水线 (D)，我们从寄存器传输级模型中提取组件级和加速器级特征，同时使用 Vivado Design Suite 2017.2，针对赛灵思 FPGA xc7vx485tffg1157-1 对设计进行综合和映射，以获取相关硬件参数，例如功耗、查找表数量以及近似加速器的延迟。我们还使用 C++ 中加速器的行为模型对应用进行仿真，并采用 [27]，中讨论的基准测试序列来确定应用级输出质量。利用这些数值，我们训练两个不同的回归模型：一个用于估算硬件参数，另一个用于估算近似加速器的结果质量（QoR）。硬件估算器以近似电路和近似加速器的特征为输入，用于预测各变体的面积、功耗或延迟；软件估算器则需要应用中所部署近似电路的平均值和平均误差，以估算其结果质量（QoR）。模型训练完成后，我们使用 NSGA-II 和进化策略（ES）进化算法，有选择地、迭代地探索加速器的架构空间，

μ后者设为 1。用于生成近似加速器变体集合、执行遗传算法以及训练统计学习模型的脚本均

[2]https://github.com/ehw-fit/evoapproxlib [3] https://github.com/ehw-fit/approx-fpgas

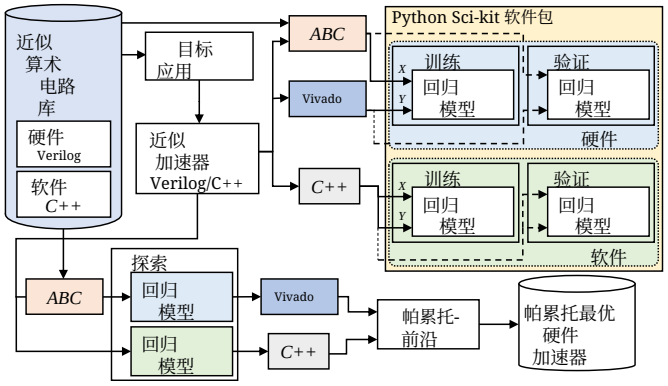


图 4：我们的实验设置和工具流程示意图。

使用 Python 编写。通过这种方式获得的最终一代加速器会经过综合和仿真，以确定其准确的硬件需求和结果质量，随后据此为目标应用构建帕累托最优硬件加速器集合。图 4 展示了这一实验设置的概览。

IV. 结果与讨论

我们使用高效视频编码应用展示了*Xel-FPGAs* 框架的有效性。研究表明，HEVC 由于采用离散余弦变换（DCT）函数，具有较强的抗错误能力。DCT 模块由 8 至 14 个加法器和乘法器组成，具体数量取决于所采用的无乘法器多常数乘法器（MCM）架构类型。这些算术单元可以进行近似处理，以提高系统的能效和/或性能。有关 HEVC 加速器模块的更多信息，请参见 [27]。

A. 流水线分析

首先，我们评估图 3 所示流水线的有效性和时间需求。具体而言，我们通过估算流水线所建立的特征集与期望硬件参数值（面积、功耗或延迟）之间的相关性来进行评估，该参数值通过 Vivado 综合获得。我们还评估每条流水线探索架构空间所需的时间，以便为目标应用获取一组帕累托最优近似加速器变体。尽管流水线（A）、（B）和（C）不依赖搜索算法，但与进化算法相关的选择会产生非零评估开销：使用 ABC 工具探索每个近似加速器时，开销约为 30ms。图 5 展示了这些实验结果。

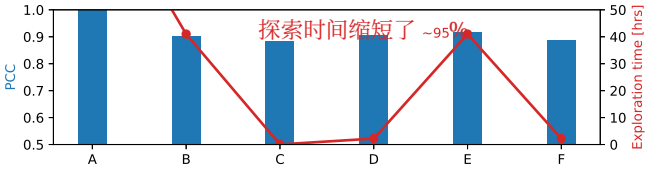


图 5：图 3 所述流水线的皮尔逊相关系数（PCC）和探索时间分析。

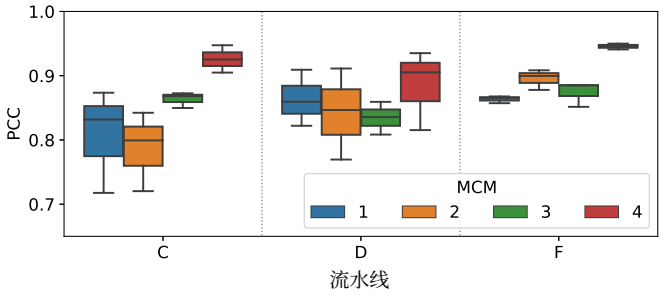


图 6：分析 *Xel-FPGAs* 的 3 条最快流水线（C、D 和 F）的 PCC——估算部署于 DCT 模块中的各种 MCM 模块的 HEVC 加速器功耗。

正如预期，配置（B）成功地将探索时间从数千小时（配置（A））缩短至仅 41 小时，但这往往会使皮尔逊相关系数（PCC）降至 0.9。通过在（B）中用 ABC 替代 Vivado，得到配置（C），我们可以进一步将时间缩短至仅 2 分钟，同时使 PCC 再降低 2%。这看似影响不大，但当考虑到规模非常大的架构空间（例如 HEVC 应用）时，模型估计值之间的差异可能相当大，从而导致算法探索非最优设计。因此，最好采用一种能够使 PCC 保持接近 0.9、同时进一步缩短探索时间的方法。流水线（D）融合了来自近似加速器变体的特征，使 PCC 达到 0.9 以上，同时将探索 1,000,000 个近似加速器变体的开销降低了近 95%。虽然配置（E）实现了更高的 PCC，但由于涉及 Vivado 工具链，探索时间又增加至 41 小时；而在流水线（F）中，PCC 降低，但探索时间也随之缩短。因此，流水线（D）是*Xel-FPGAs*的理想配置。

此外，除了缩短探索时间，我们还进行了实验，以分析三条最快流水线在部署四种不同类型的 MCM 组件时，估算 HEVC 加速器模块功耗的效果。实验结果如图 6 所示。尽管流水线（F）似乎能获得更好的解，但我们更关注平均而言能得到最佳 PCC 估算结果的流水线配置。对于 HEVC 用例，流水线（D）实现了优于其他流水线的权衡。

B. 消融研究

接下来，我们在 HEVC 用例中部署三种不同的 统计回归模型，以分析它们的能力，分别是：（1）随机森林，（2）贝叶斯回归，以及（3）支持向量机。按照前文所述，我们使用 PCC 指标和流水线（D），探究这些模型准确估算近似加速器变体的结果质量和功耗的效果。图 7 展示了这些模型针对四种不同 MCM

部署在应用中的加速器组件。从中可以看出，不同的回归 模型取得了不同的

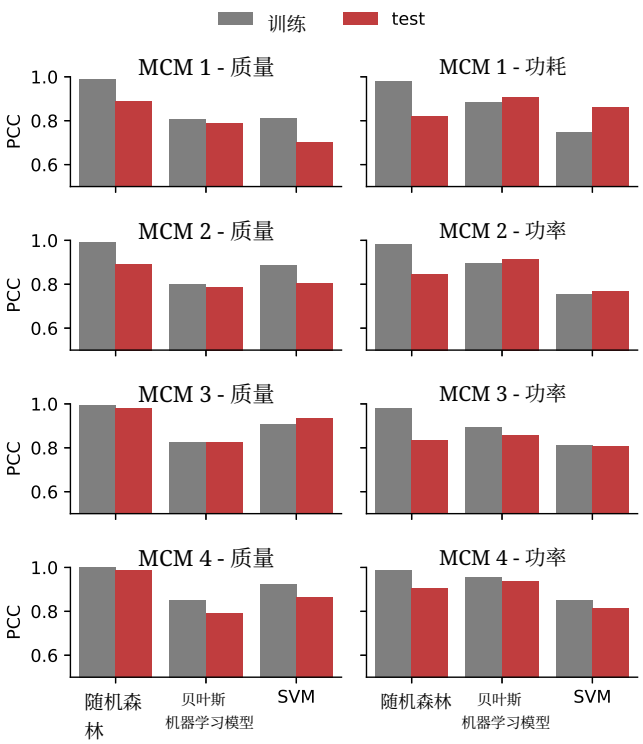


图 7：用于估算 HEVC 加速器硬件和质量参数的三种不同统计回归模型的皮尔逊相关系数（PCC）分析。

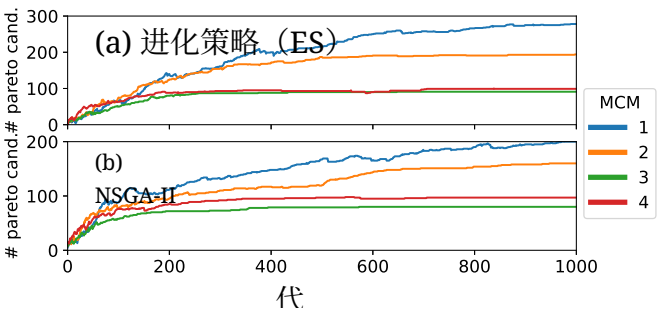


图 8：分析 ES 和 NSGA-II 的收敛情况。

相关性，具体取决于所估算的参数。例如，在估算所有相关加速器（即 MCM1 至 MCM4）的结果质量时，随机森林 的表现最佳；而在估算所有加速器的功耗时，贝叶斯回归的表现最佳。因此，在探索架构空间时，我们使用这两种模型来估算相关参数；它们是*Xel-FPGAs* 框架中预先提供的近 20 种模型中的两种。

C. 架构探索

既然我们已经确定了使用*Xel-FPGAs*这一用例时的最佳回归模型及其相应配置，现在就来分析该框架的模型生成和探索能力。我们的假设是，每一代新探索的模型相较于上一代，往往会改善各项权衡，从而在搜索结束时得到一组帕累托最优硬件

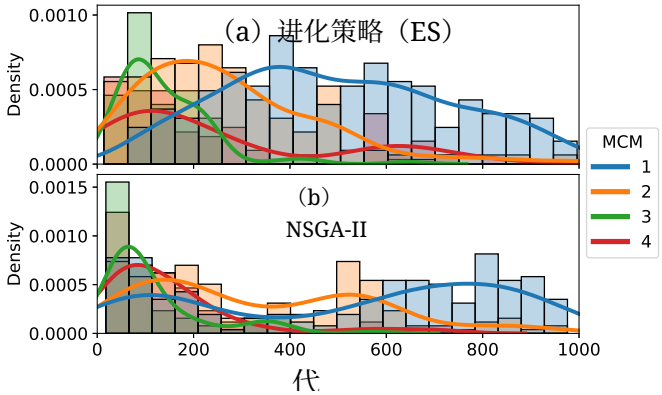


图 9：分析 ES 和 NSGA-II 在每一代中生成的帕累托最优设计所占的百分比（密度）。

加速器。我们首先分析框架中所用进化算法的能力。除对我们的用例帮助不大的爬山算法外，我们还按照第 II-B 节所述，在 HEVC 应用上评估了 NSGA-II 和进化策略（ES）算法。这些算法采用相似的种群设置，并且设计时间开销相同。图 8 展示了 ES 和 NSGA-II 算法在每一代中获得的帕累托最优点数量。另一方面，图 9 展示了算法执行每次迭代中获得的帕累托最优设计占总数的百分比，称为密度。从前者可以看出，与较复杂的 MCM2 和 MCM1 模块相比，较简单的 MCM3 和 MCM4 模块收敛更快，也就是说，它们更早达到所获帕累托最优设计数量的峰值。自第 300 代以来，这些较简单的模块几乎没有找到更好的解；这一点在 ES 中更为明显。另一方面，与 ES 相比，NSGA-II 需要多搜索几代，才能获得相近数量的帕累托最优设计。造成这一现象的原因分析图 9 时，这一点更加明显。NSGA-II 算法获得的大部分帕累托最优设计是在前 300 次迭代中得到的，尤其是对于更复杂的模块。因此，在 MCM1 和 MCM2 模块中可以观察到第二个峰值，即又有很大比例的帕累托最优设计在此阶段被识别出来。由于 NSGA-II 算法以较大的种群集合为起点，因此能够在搜索的早期代中识别出相当比例的帕累托最优设计。进化策略算法则并非如此，它在整个搜索迭代过程中识别帕累托最优设计的分布更加均匀。因此，进化策略可能更适合为目标应用识别大量设计权衡，而 NSGA-II 可能更适合快速获得高质量解，这与我们最初的假设相矛盾。

接下来，我们转向框架的主要目标。图 10 展示了使用进化策略和 NSGA-II 算法生成硬件加速器的结果，其中

我们描绘了每一代所搜索到的近似加速器变体的估计峰值信噪比和功耗值。尽管第 *0* 代标识出的点呈现随机分布，但正如预期的那样，只有第 *1000* 代的结果清晰可见，而其他结果似乎都不见了。这是因为第 *1000* 代的结果遮蔽了其他代的结果；也就是说，较新一代的结果遮蔽了较早一代的结果，因为它们与前几代的结果相似。这为将探索时间进一步缩短 10 倍提供了机会，从而将总探索时间缩短 99%。此外，正如预期的那样，与 NSGA-II 搜索相比，进化策略算法获得的帕累托最优点要多得多。

D. 最终评估

在完成所有中间评估后，我们继续评估该框架的整体有效性，并

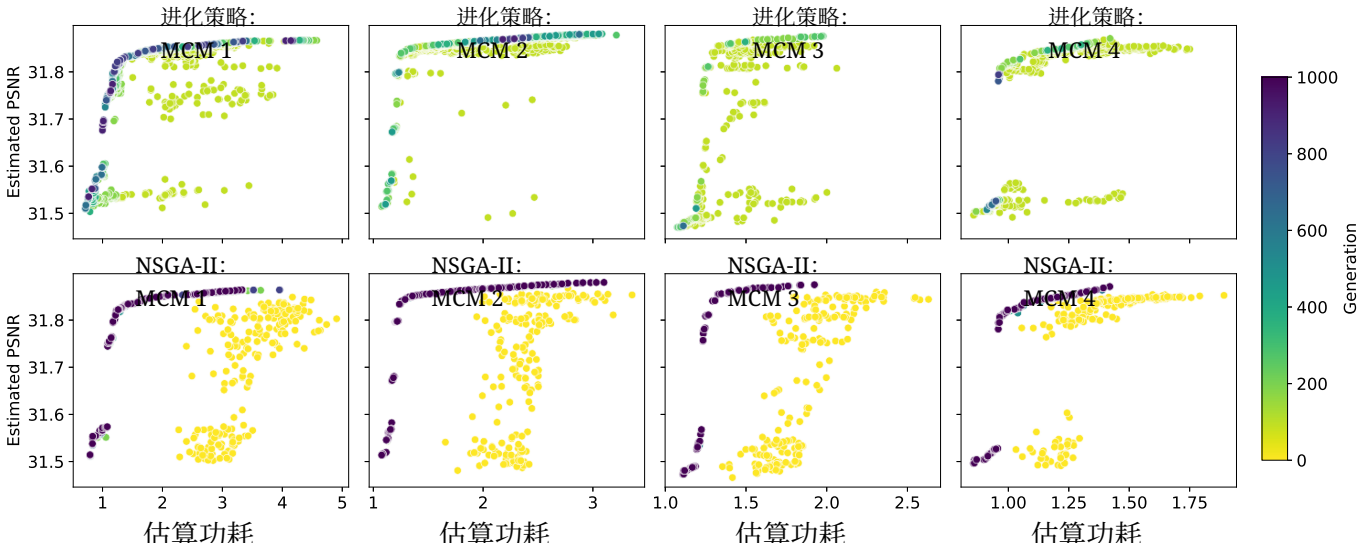


图 10：对进化策略（上）和 NSGA-II（下）每一代的帕累托最优 HEVC 近似加速器进行分析。

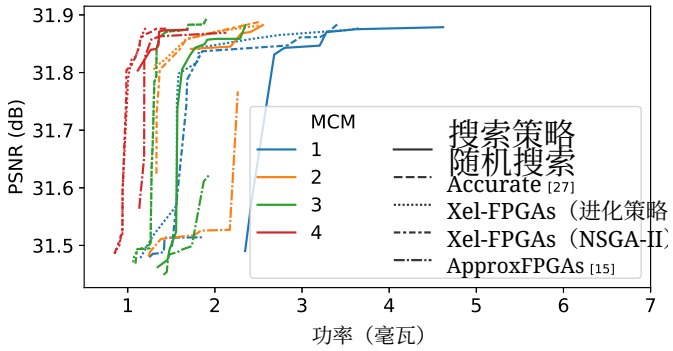


图 11：我们的*Xel-FPGAs*框架与最先进水平 [15] 所获得的近似加速器比较。

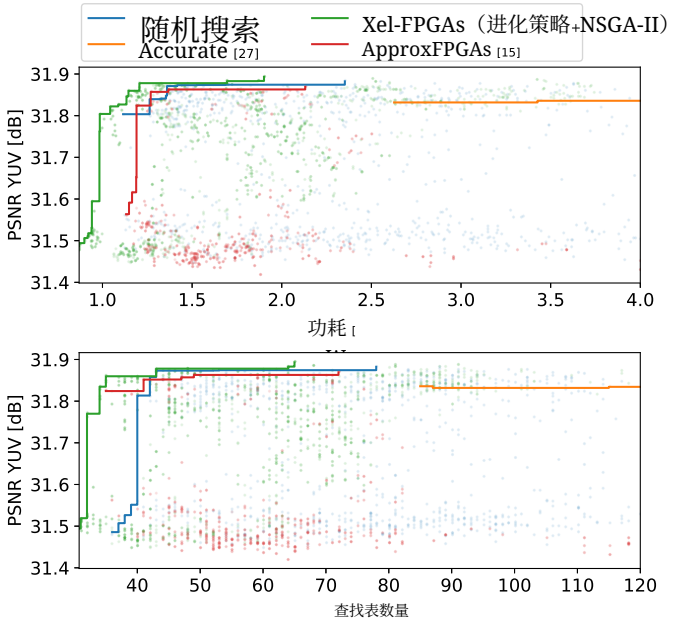


图 12：*Xel-FPGAs* 与*SoA* [15]的最终比较。

将其与当前最先进的近似加速器探索方法进行比较。我们重新实现了 ApproxFPGAs [15],中讨论的技术，为我们的 HEVC 用例生成近似加速器变体，并获得一组帕累托最优的基于FPGA的近似计算电路。图 11 展示了我们的*Xel-FPGAs*框架获得的帕累托最优近似加速器变体，与最先进的 ApproxFPGAs 方法在加速器级别上的比较。可以看出，对于四种加速器类型（MCM1 至 MCM4），我们的框架在获得更优的 HEVC 应用帕累托最优近似加速器变体集合方面，优于最先进的方法和随机搜索。采用进化策略可增加所获得的帕累托最优设计数量，即，通过找到位于帕累托前沿上的候选设计，进一步拓展帕累托前沿。同样，在对所获得的帕累托最优近似加速器进行应用级分析时，如图所示，*Xel-FPGAs* 框架中的进化算法组合明显比 ApproxFPGAs 能够识别出更优的设计权衡，

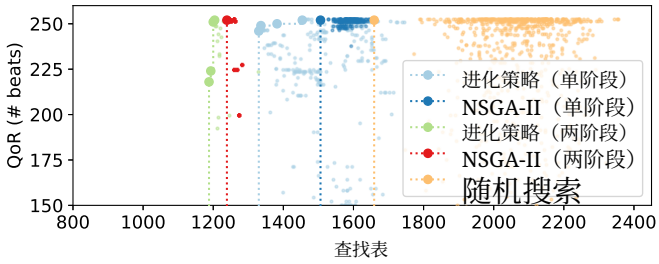


图 13：分析*Xel-FPGAs*的可扩展性。

图 12 中的帕累托前沿。此外，我们还成功找到了能够降低应用面积需求的设计点，具体而言，是减少所需查找表的数量；这也是使用我们的*Xel-FPGAs* 框架的另一项优势。

E. 可扩展性

最后，为了评估我们的框架扩展到多阶段应用的能力，我们考虑了 [28]中讨论的生物信号处理 Pan-Tompkins 算法的一个近似版本。该算法由五个应用阶段组成，每个阶段都是由不同组合的加法器和乘法器构成的滤波器，总计包含 73 个算术单元。因此，本实验的主要目标是评估：将探索分解为多个阶段（n 阶段）是否更适合该框架，即，先分别搜索每个应用阶段，再结合各阶段的帕累托最优设计进行搜索；还是一次性设计所有内容（1阶段）。正如文章 [28]，所强调的，我们使用应用级质量指标——心跳次数——来评估应用阶段的质量，方法是将近似变体的行为模型嵌入应用中。图 13 概述了采用 1阶段和 2阶段（此处为 n=2）探索策略时获得的结果。我们可以看到，通过进化策略和 NSGA-II 获得的 2阶段 策略设计，表现均显著优于 1阶段设计。因此，我们可以得出结论：分层逐步逼近策略（n阶段）更擅长识别改进后的帕累托最优设计集，而这些设计更适合该应用。

V. 结论

我们介绍了*Xel-FPGAs*，这是一个端到端自动化框架，用于针对计划部署在现场可编程门阵列中的目标应用开展近似加速器的架构空间探索。我们训练并使用统计回归模型来估计应用的硬件需求和输出质量，从而有效缩短架构空间探索所需的时间。我们还提出使用 ABC 工具作为近似电路和近似加速器的特征提取器，在不降低估计质量的前提下进一步缩短框架的探索时间。利用这些特征，我们通过迭代运行 NSGA-II 算法，高效探索近似加速器的架构空间，并最终确定一组包含 200 个帕累托最优近似加速器的方案

共 1000 代。这些设计经过重新综合和仿真，以计算其准确的硬件需求和质量，从而确定目标应用最终的帕累托最优近似加速器变体集合。系统设计人员可以分析这些近似加速器变体之间的权衡，选择满足系统规格和应用质量要求的方案。Xel-FPGAs 框架在识别更优的帕累托最优近似加速器变体集合方面优于最先进水平，同时还将探索时间缩短了近 95%。我们还通过一个多阶段应用展示了框架的可扩展性，并讨论采用分层搜索策略的优势。我们的 *Xel-FPGAs* 框架为开源框架，可通过以下网址在线访问：https://github.com/ehw-fit/xel-fpgas。

致谢

本研究部分得到弹性嵌入式系统博士学院的支持。该博士学院由 TU Wien 信息学学院与维也纳应用科技大学联合运营。本研究还部分得到捷克科学基金会项目 21-13001S 的支持。此外，本研究通过 ASPIRE 卓越研究奖获得 ASPIRE 的支持。ASPIRE 是阿布扎比先进技术研究委员会（ATRC）的技术项目管理部门。

参考文献

[1] S. M. S. Trimberger，“FPGA发展的三个时代：回顾FPGA技术的前三十年：本文探讨了摩尔定律如何推动FPGA设计经历三个时期：发明时代、扩展时代和积累时代”，*IEEE* 固态电路杂志，第10卷，第2期，第16–29页，2018年。[2] R. Watanabe、S. Ura、Q. Zhao 和 T. Yoshida，“将FPGA构建平台作为云服务实现”，载于高效加速器与可重构技术国际研讨会第*10*届会议论文集，2019年，第1–6页。[3] L. H. Crockett、R. A. Elliot、M. A. Enderwitz 和 R. W. Stewart，《*Zynq* 指南：使用*ARM Cortex-A9*在*Xilinx Zynq-7000*全可编程片上系统上进行嵌入式处理》。Strathclyde Academic Media，2014年。[4] P. Mantovani、E. G. Cota、K. Tien、C. Pilato、G. Di Guglielmo、K. Shepard 和 L. P. Carloni，“用于高性能嵌入式系统细粒度动态电压与频率调节分析的基于FPGA的基础设施”，载于第*53*届年度设计自动化会议论文集，2016年，第1–6页。[5] S. Huda、M. Mallick 和 J. H. Anderson，“用于降低FPGA功耗的时钟门控架构”，载于*2009*年现场可编程逻辑与应用国际会议。IEEE，2009年，第112–118页。[6] V. K. Chippa、S. T. Chakradhar、K. Roy 和 A. Raghunathan，“近似计算中固有应用韧性的分析与表征”，载于第*50*届年度设计自动化会议论文集，2013年，第1–9页。[7] M. Samadi、D. A. Jamshidi、J. Lee 和 S. Mahlke，“Paraprox：面向数据并行应用的基于模式的近似”，载于第*19*届编程语言与操作系统架构支持国际会议论文集，2014年，第35–50页。[8] W. El-Harouni、S. Rehman、B. S. Prabakaran、A. Kumar、R. Hafiz 和 M. Shafique，“在高效视频编码中采用近似计算实现节能运动估计”，载于欧洲设计、自动化与测试会议及展览（*DATE*），*2017*年。IEEE，2017年，第1384–1389页。[9] V. K. Chippa、D. Mohapatra、K. Roy、S. T. Chakradhar 和 A. Raghunathan，“可扩展努力型硬件设计”，*IEEE*超大规模集成电路系统汇刊，第22卷，第9期，第2004–2016页，2014年。

[10] H. Saadat、H. Javaid 和 S. Parameswaran，“误差偏差近零的近似整数与浮点除法器”，载于第56届年度设计自动化会议论文集，2019年，第1–6页。[11] S. Hashemi、R. I. Bahar 和 S. Reda，“Drum：面向近似应用的动态范围无偏乘法器”，载于2015年IEEE/ACM国际计算机辅助设计会议（ICCAD）。IEEE，2015年，第418–425页。[12] S. Ullah、S. Rehman、B. S. Prabakaran、F. Kriebel、M. A. Hanif、M. Shafique 和 A. Kumar，“面向基于FPGA的硬件加速器的面积优化低延迟近似乘法器”，载于第55届年度设计自动化会议论文集，2018年，第1–6页。[13] B. S. Prabakaran、S. Rehman、M. A. Hanif、S. Ullah、G. Mazaheri、A. Kumar 和 M. Shafique，“Demas：用于为基于FPGA的系统构建近似加法器的高效设计方法”，载于2018年欧洲设计、自动化与测试会议及展览（DATE）。IEEE，2018年，第917–920页。[14] S. Ullah、S. S. Sahoo、N. Ahmed、D. Chaudhury 和 A. Kumar，“Appaxo：为基于FPGA的嵌入式系统设计应用专用近似运算符”，《ACM嵌入式计算系统汇刊》（TECS），第21卷，第3期，第1–31页，2022年。[15] B. S. Prabakaran、V. Mrazek、Z. Vasicek、L. Sekanina 和 M. Shafique，“ApproxFPGAs：采用基于ASIC的近似算术组件构建基于FPGA的系统”，载于2020年第57届ACM/IEEE设计自动化会议（DAC）。IEEE，2020年，第1–6页。[16] J. Echavarria、S. Wildermann、A. Becher、J. Teich 和 D. Ziener，“Fau：基于LUT的FPGA上的快速误差优化近似加法单元”，载于2016年国际现场可编程技术会议（FPT）。IEEE，2016年，第213–216页。[17] S. Ullah、S. S. Sahoo 和 A. Kumar，“Clapped：用于在基于FPGA的嵌入式系统中实现跨层近似的设计框架”，载于2021年第58届ACM/IEEE设计自动化会议（DAC）。IEEE，2021年，第475–480页。[18] S. Ullah、S. Reh man、M. Shafique 和 A. Kumar，“面向基于FPGA的硬件加速器的高性能精确与近似乘法器”，《IEEE集成电路与系统计算机辅助设计汇刊》，第41卷，第2期，第211–224页，2021年。[19] S. Ullah、S. S. Murthy 和 A. Kumar，“Smapproxlib：基于FPGA的近似乘法器库”，载于第55届年度设计自动化会议论文集，2018年，第1–6页。[20] Z. Vasicek 和 L. Sekanina，“在FPGA上实现近似电路的基于搜索的综合”，载于2016年第26届现场可编程逻辑与应用国际会议（FPL）。IEEE，2016年，第1–4页。[21] S. Venkataramani、V. K. Chippa、S. T. Chakradhar、K. Roy 和 A. Raghunathan，“面向近似计算的质量可编程向量处理器”，载于第46届年度IEEE/ACM国际微体系结构研讨会论文集，2013年，第1–12页。[22] V. Mrazek、M. A. Hanif、Z. Vasicek、L. Sekanina 和 M. Shafique，“autoax：利用近似组件库进行自动设计空间探索与电路构建的方法”，载于第56届年度设计自动化会议论文集，2019年，第1–6页。[23] V. Mrazek、R. Hrbacek、Z. Vasicek 和 L. Sekanina，“Evoapprox8b：用于电路设计及近似方法基准测试的近似加法器与乘法器库”，载于欧洲设计、自动化与测试会议及展览（DATE），2017年。IEEE，2017年，第258–261页。[24] A. Mishchenko 等，“ABC：用于时序综合与验证的系统”，网址：http://www. eecs. berkeley. edu/alanmi/abc，第17卷，2007年。[25] K. Deb、A. Pratap、S. Agarwal 和 T. Meyarivan，“快速精英多目标遗传算法：NSGA-II”，《IEEE进化计算汇刊》，第6卷，第2期，第182–197页，2002年。[26] S. H. Jacobson 和 E. Y¨ucesan，“广义爬山算法性能分析”，《启发式学报》，第10卷，第387–405页，2004年。[27] Z. Vasicek、V. Mrazek 和 L. Sekanina，“迈向面向HEVC标准的低功耗近似DCT架构”，载于欧洲设计、自动化与测试会议及展览（DATE），2017年。IEEE，2017年，第1576–1581页。[28] B. S. Prabakaran、S. Rehman 和 M. Shafique，“Xbiosip：一种用于边缘近似生物信号处理的方法”，载于第56届年度设计自动化会议论文集，2019年，第1–6页。