arXiv:2601.07172v1 [cs.ET] 12 Jan 2026

TranSC：使用随机逻辑的硬件感知超越函数设计

Mehran Moghadam，*IEEE*研究生会员，Sercan Aygun ，*IEEE*高级会员，以及 M. Hassan Najafi ，*IEEE*高级会员

摘要——**超越函数的硬件友好型实现一直是设计自动化领域长期存在的挑战。这些无法表示为代数运算有限组合的函数，给数字电路设计带来了显著的复杂性。本研究提出了一种新方法 TranSC，利用随机计算（SC）以轻量且高精度的方式实现超越函数。在成熟的随机计算技术基础上，我们的方法探索了替代随机源——具体而言，采用准随机范德科普特低差异度（LD）序列，而非传统的伪随机性。这一转变提升了基于随机计算的计算的精度与效率。我们通过对包括三角函数、双曲函数和激活函数在内的各种函数类型进行广泛实验，验证了该方法的有效性。与当前最先进的解决方案相比，所提出的设计方法将均方误差（MSE）最多降低了98%，同时分别将硬件面积、功耗和能耗降低了33%、72%和64%。**

索引词—**图像变换、伪随机性、准随机性、机器人机械臂机动、随机计算、超越函数、范德科普序列。**

I. 引言

这HE 深度神经网络的快速发展对计算硬件提出了前所未有的需求，尤其是对引入非线性的激活函数。Sigmoid 函数、双曲正切及其底层的指数和对数形式等函数，采用传统二进制计算实现时开销尤为昂贵。同样，正弦和余弦等三角函数对于分析数字信号处理（DSP）中的周期性行为不可或缺，并广泛应用于电机控制、通信和噪声滤波。这些领域通常需要满足严格的实时性和低功耗约束。在通用处理器上通过软件实现超越函数会造成显著的计算延迟，最终限制吞吐量和响应能力。为克服这一瓶颈，低延迟、高性价比的设计

面向嵌入式数字信号处理系统的硬件加速器已成为一项关键目标 [2]。

超越函数在计算机图形学和机器人学中同样至关重要，在坐标变换、光照计算和机器人

本研究部分得到美国国家科学基金会（NSF）项目资助，资助编号为 # 2019511、#2339701；还得到美国国家航空航天局项目资助，编号为 80NSSC25C0335，以及英伟达慷慨提供的捐赠。Mehran Moghadam 和 M. Hassan Najafi 任职于美国俄亥俄州克利夫兰市凯斯西储大学电气、计算机与系统工程系。电子邮箱：{moghadam, najafi}@case.edu。Sercan Aygun 任职于美国路易斯安那州拉法叶市路易斯安那大学拉法叶分校计算与信息学院。电子邮箱：sercan.aygun@louisiana.edu。本文的初步版本曾以 [1] 的形式发表。

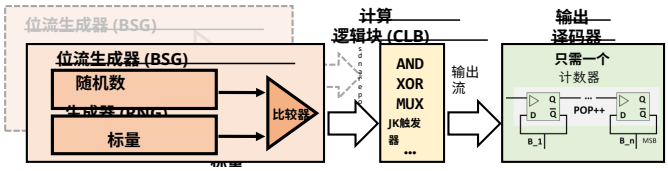

图 1. 一般随机计算系统架构，包含位流生成器、CLB 和输出解码器，后者将输出位流转换为标准基数表示。

运动学中发挥着核心作用。这些运算通常依赖正弦和余弦计算，其精度会直接影响最终输出，无论是渲染图像的视觉质量，还是机器人机械臂的定位精度。现代图形处理器中配备专用特殊功能单元（SFU）以加速这些函数的求值，这凸显了它们的重要性 [3]。然而，在移动机器人、自主无人机或增强现实/虚拟现实头显等资源受限的平台上，特殊功能单元的功率和面积需求可能难以承受。这一限制促使人们迫切寻求替代硬件解决方案：它们既要极其轻量、能效高，又要为性能关键型应用提供足够的精度。

随机计算（SC）是一种重新兴起的计算范式，能够以极低成本实现复杂的算术运算 [4], [5]。随机计算具有强大的数据处理能力，同时大幅降低硬件成本和功耗，因此适用于资源受限系统。随机计算系统通常由三个核心模块组成：一个位流生成器（*BSG*）、一个计算逻辑块（*CLB*）和一个输出解码器 [6]。在随机计算系统中，标量值会转换为随机二进制序列，即位流，其中每一位的权重相同。位流的随机性质量（由 BSG 决定，BSG 通常由伪随机数生成器（RNG）实现）对系统精度起着关键作用。图1 展示了一种由 BSG、CLB 和输出解码器组成的一般随机计算系统。CLB 单元使用简单的逻辑门（例如与、或、异或）或轻量级时序电路（如触发器（FF））处理位流。解码器将输出位流转换回标准二进制基数格式，通常通过统计位流中 1 的数量来实现 [7]。这种架构使随机计算能够用基本逻辑组件取代高成本算术电路，为实现复杂数学函数提供了一种高效的替代方案。

在随机计算框架中，已经开发出大量算术模块 [8]–[10]。特别是，许多研究致力于将随机计算扩展到超越函数和代数函数 [11]–[15]，通常通过采用中间

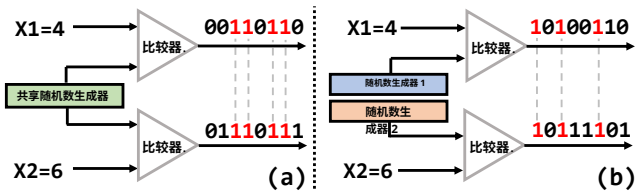

图 2. 使用随机数生成器和比较器生成位流：(a) 用于相关位流的共享随机数生成器。(b) 用于不相关位流的独立伪随机数生成器。

数学工具，例如伯恩斯坦多项式以及泰勒或麦克劳林级数展开。尽管以往的工作主要集中于设计低成本的CLB单元 [16]，本工作则将注意力转向BSG，将其作为提高精度、降低硬件开销和处理延迟的关键手段。具体而言，我们利用轻量级低差异（LD）序列生成器构成比特流生成单元，以实现高效的随机计算设计。与伪随机源相比，LD序列是一种能够更均匀地分布值的准随机数生成器，可实现更快的收敛速度并减少基于随机计算的设计中的误差 [17]。本研究的主要贡献如下：

❶ 使用轻量级准随机BSG单元探索基于随机计算的超越函数设计。

❷ 利用低差异比特流降低超越函数设计的延迟并提高硬件效率。❸ 缓解基于随机计算的超越函数设计中的计算误差和硬件成本。

❹ 展示随机计算的两种新颖应用：图像几何变换和机器人机械手定位。

本文其余部分安排如下。第II 节回顾随机计算的基础知识。第III节讨论实现超越函数的核心概念，并从传统随机计算方法过渡到相关实现。第IV节介绍所提出的框架，该框架采用新颖的随机数生成技术，并在最先进水平的设计基础上进行了改进。第V节给出详细的实验结果，包括精度和硬件成本比较。第VI 节对所提出的方法相较于最先进方法进行验证和论证。第VII节通过两个具有代表性的用例展示随机计算的应用。第VIII 节详细讨论并比较最先进的设计实现。最后，第IX 节总结本文的主要贡献。

II. 随机计算基础

随机计算（SC）通过摆脱以位置编码信息的传统二进制基数系统，提供了一种非传统的计算方法 [4]。不同于依赖需要复杂算术电路的定点或浮点表示，随机计算使用概率比特流上的简单位级逻辑来实现算术运算。例如，对独立位流执行位级与运算即可实现乘法。其他基本的随机计算算术电路包括用于缩放加法（add）的 [18],多路复用器，用于缩放减法（sub）的输入反相JK-FF，以及用于近似除法（div）的 [19]。

范围 $0 \leq X \leq N$ 内的任意标量值（或等价地，归一化到单位区间后的 $0 \leq \frac{X}{N} \leq 1$ ）都可以编码为 N位随机比特流，所用硬件配置由随机数生成器和比较器组成。对于每个位位置，随机数生成器会生成一个随机值（R），比较器则判断条件$X > R_i$是否成立。若成立，相应位位置将设为逻辑1；否则设为逻辑0。这一位流生成过程是随机计算系统中的基本步骤。

随机计算操作的性能直接取决于比特流生成单元。随机性的质量以及位流之间的相关程度，对计算的性能和精度起着关键作用。令 $X1$ 和$X2$ 表示编码为随机比特流的标量值。现有的位流生成方法主要采用以下两种方式之一：I）共享随机数生成器方法：将单个随机源与两个比较器共享，以生成两个相关的位流（图2（a））；II）独立随机数生成器方法：使用两个独立的随机源生成两个不相关的位流（图2（b））。

当前先进的随机计算设计中，有两种主要的比特流生成设置：i）线性反馈移位寄存器（LFSR）与比较器组合 [20] ；ii）Sobol 序列生成器与比较器组合 [21]。前者生成伪随机序列，后者生成准随机序列。伪随机源使用确定性数学算法生成序列，并且在使用相同种子初始化时会重复生成相同序列。准随机性引入有序的变化，生成看似随机、同时保持更均匀分布的序列。这些序列的特点是空间中的点分布均匀，因此具有更好的均匀性。我们将在后文讨论，对于本项工作，我们采用经济高效的范德科普特（VDC）序列生成器来利用准随机序列。这些生成器能够提供均匀分布，并具有良好的硬件特性。

CLB单元处理生成的随机位流。任何算术函数 f 都可以用与输入位流相关的概率 来表示。对于长度为 N 的位流，观测到逻辑1的概率记为 $P = \frac{\sum logic-1}{N}$ 。对于两个输入操作数$X1$ 和 $X2$，相应的概率分别记为P_{X1} 和 P_{X2}。随机函数 f 的输出则表示为 $f_{mul} = P_{X1} \times P_{X2}$、$f_{add} = P_{X1} + P_{X2}$、$f_{sub} = P_{X1} - P_{X2}$ 和 $f_{div} = \frac{P_{X1}}{P_{X2}}$。在实践中，随机计算系统的预期 输出与实际产生的 输出之间可能存在差异，主要有三个原因：① 输入位流之间的交叉相关性不当；② 位流中的随机波动，通常源自伪随机源；③ CLB引入的函数近似。

相关性由两个位流中“1”的对齐情况定义。确保输入操作数之间具有适当的互相关性，对于实现所需的随机函数至关重要。量化两个位流之间相关性的常用指标是随机交叉相关(SCC) [22]:

$$SCC = \begin{cases} \frac{ad-bc}{N\times min(a+b,a+c)-(a+b)\times(a+c)} & , if\ ad > bc \\ \frac{ad-bc}{(a+b)\times(a+c)-N\times max(a-d,0)} & , else \end{cases} \quad (1)$$

其中， a、 b、 c和 d 分别表示从两个位流的对应位位置取得的逻辑对 11、10、01 和 00。

相关性级别由所有比特位置上重叠的逻辑1的数量决定。当这种重叠达到最大时，位流表现出最大相关性($SCC = +1$)；当重叠最少时，它们处于最小相关性(SCC=−1)。一种特殊情况是位流具有零相关性 ($SCC = 0$), 这在某些随机计算操作中很理想，例如使用按位与运算进行乘法。设 $X1 = 4$, $X2 = 6$ 和 $N = 8$。输入概率值为 $P_{X1} = \frac{4}{8}$ 和 $P_{X2} = \frac{6}{8}$。位流的三种相关性度量示例如下:

① $X1 \to 0\,0\,0\,0\,1\,1\,1\,1$, $X2 \to 0\,0\,1\,1\,1\,1\,1\,1$ ➙ Maximum Correlation

② $X1 \to 1\,1\,1\,1\,0\,0\,0\,0$, $X2 \to 0\,0\,1\,1\,1\,1\,1\,1$ ➙ Minimum Correlation

③ $X1 \to 0\,1\,0\,1\,0\,1\,0\,1$, $X2 \to 0\,0\,1\,1\,1\,1\,1\,1$ ➙ Zero Correlation

红色标出的部分表示重叠的1比特。对于最大相关性 1，为清晰起见，重叠的“1”位位置在右侧对齐，尽管它们也可能随机出现。对于最小相关性 2，重叠的“1”数量最少，因此第一个和第二个操作数中的1彼此分开。互相关会影响CLB单元的行为，从而产生不同的功能结果 [23], [24]。例如，在零相关性 3 的情况下，对 $X1$ 和$X2$ 执行按位与运算，得到输出 $Y \to 0\,0\,0\,1\,0\,1\,0\,1$。输入位流不相关时，输入与门的预期 f为 $f_{mul} = P_{X1} \times P_{X2}$。输出 Y的概率为 3 8，验证了预期的运算（即乘法）：$f_{mul} = P_{X1} \times P_{X2} = \frac{4}{8} \times \frac{6}{8} = \frac{3}{8}$。然而，在 1 和 2 中，最大相关性和最小相关性经按位与运算后会产生不同的输出概率值： $Y1 = \frac{4}{8}$和 $Y2 = \frac{2}{8}$。按位与运算中的这些不同相关性揭示了不同的函数，例如 $f_{①} = \min(P_{X1}, P_{X2})$和$f_{②} = \max(0, P_{X1} + P_{X2} - 1)$。如上所示，相关性在定义CLB的行为方面起着关键作用，这凸显了比特流生成单元的重要性。在本研究中，我们通过重新审视比特流生成器单元中使用的伪随机数生成器，显著提升了基于随机计算的超越函数设计的性能和硬件效率。

III. 从传统超越函数到基于随机计算的超越函数

随机计算（SC）的一个主要优势是能够通过简单且低成本的设计实现复杂的算术函数。本研究提出了一种高效的超越函数实现方法，重点是在降低硬件成本的同时提高精度，并与最先进水平的随机计算实现进行比较。

表I 本研究所探讨的超越函数的截断麦克劳林级数展开

函数	多项式等价式
正弦(x)	$x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!}$
余弦(x)	$1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!}$
正切(x) = 正弦(x) 余弦(x)	$x + \frac{x^3}{3} + \frac{2x^5}{15} + \frac{17x^7}{315}$
双曲正切(x)	$x - \frac{x^3}{3} + \frac{2x^5}{15} - \frac{17x^7}{315}$
反正切(x)	$x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7}$
$sigmoid(x)$	$\frac{1}{2} + \frac{x}{4} - \frac{x^3}{48} + \frac{x^5}{480}$
$Sinc\ (x)$ = 正弦(x) x	$1 - \frac{x^2}{3!} + \frac{x^4}{5!} - \frac{x^6}{7!}$
e^{-x}	$1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} + \frac{x^4}{4!} - \frac{x^5}{5!}$
$\ln(1+x)$	$x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \frac{x^5}{5}$

设计。以往研究探索了多种基于随机计算的算术方法，包括分段线性近似 [25], 基于有限状态机的设计 [26],以及多项式因式分解技术 [27]。较新的研究则着重于在可配置逻辑块单元中实现多项式函数，这对于机器学习应用尤其有利 [13], [15]。本研究强调了轻量级、高质量比特流生成器单元的关键作用，它们会间接影响可配置逻辑块的性能。

超越函数在从通信系统到计算机视觉等多种应用中发挥着关键作用 [28]–[30]。当实现这些函数时，鲁棒性和硬件效率是首要设计目标，随机计算尤为重要。在最先进水平的研究中，Parhi和Liu [12] 利用截断麦克劳林级数展开，探索了基于随机计算的复杂非线性函数实现，包括三角函数、指数函数、对数函数、双曲正切和Sigmoid 函数。研究者还提出了使用切比雪夫多项式 [31],有限状态机 [32], 以及伯恩斯坦多项式 [33], [34]实现这些函数的其他随机计算方案。其他最先进水平的研究则将这一领域拓展至互补金属氧化物半导体技术以外的器件级设计。例如，Chen等人 [35] 将绝热量子磁通参量子（AQFP）超导技术与伯恩斯坦多项式结构结合，用于实现超越函数。据我们所知，我们的工作首次将准随机性引入随机计算设计中的比特流生成器单元，从而实现更高效的基于随机计算的超越函数实现。

在数学中，超越函数与代数函数不同，属于一个独特的类别，无法表示为有限个代数运算（如加法、减法、乘法、除法、乘方或开方）的组合。常见示例包括三角（例如，sin(x)）、指数（例如， a^x）、对数（例如，$\log_a x$）和双曲（例如，tanh(x)）函数。相比之下，诸如 $x^3 + 1$、 $\frac{2x-3}{x+1}$和 $\sqrt[5]{x}$ 的表达式被归类为代数函数。我们的方法显著提升了超越算术函数的随机计算设计，使其有别于现有的最先进水平解——

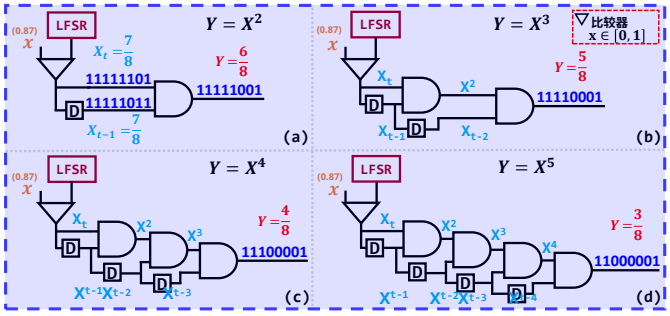

图3. 随机计算中简单多项式函数的传统设计：(a) 二次函数（x^2），(b) 三次函数（x^3），(c) 四次函数（x^4），以及 (d) 五次函数（x^5）。

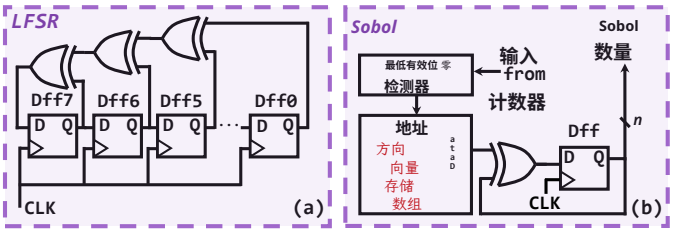

图 4. 最先进的随机数生成器的整体设计：(a) 线性反馈移位寄存器设计（8 位）；以多项式 $x^8+x^7+x^6+1$ 展示最大长度线性反馈移位寄存器的示例；(b) Sobol设计。

决方案。我们聚焦于关键函数的高硬件效率互补金属氧化物半导体实现，包括sin(x)、cos(x)、tan(x)、e^{-x}、tanh(x)、arctan(x)、sigmoid(x)、Sinc(x)和ln(1 +x)，并采用截断麦克劳林级数展开和霍纳法则 [12], [13]。函数 $f(x)$的麦克劳林级数是以零为中心的泰勒级数的特例，可表示为 $\sum_{n=0}^{\infty} \frac{f^{(n)}}{n!} x^n$，其中$f(n)$表示 $f(x)$的第 n阶导数。下文将介绍使用霍纳法则对上述函数进行近似的随机计算设计。表I列出了本研究所探讨的各函数的截断麦克劳林级数展开。

A. 随机计算中的多项式设计

为便于设计复杂的超越函数，我们首先实现基本的多项式函数，包括随机计算中的二次函数（x^2）、三次函数（x^3）、四次函数（x^4）、五次函数（x^5）以及高阶多项式。二次函数（x^2）可在随机计算中通过使用一个与门并结合一个去相关元件（例如，一个一位移位寄存器或一个D型触发器（D-FF））来实现。同样，x^3、x^4和 x^5 函数可分别通过两个、三个和四个与门，并分别配以两个、三个和四个延迟元件来实现，从而确保每个与门的输入端达到所需的独立性（见第II节）。图3（a）至（d）展示了这些多项式函数在比特流长度为 $N = 8$时的电路设计。

例如，要计算0.875在比特流长度为 8 时的多项式值，会生成相应的位流$X_t = 11111101$ 及其移位一位的版本 $X_{t-1} = 11111011$。所得位流（Y）表示$(0.87)^2$ 的预期结果，如图3(a) 所示。更高阶多项式的实现方式类似，只需逐步在中间电路中添加延迟元件，如图3(b)-(d) 所示。基于这些简单多项式的基础随机计算实现，我们将在下一节探讨更复杂超越函数的设计。为进行公平比较，每个函数的多项式阶数均选取为与传统方法所用阶数一致。

IV. 提出框架：TRAN SC

我们提出的框架 **TranSC** 改进了比特流生成单元，使可配置逻辑块能够进行修改，从而更高效地实现超越函数的随机计算。尽管近期最先进水平的研究日益强调设计可配置逻辑块以支持这些函数，主要针对函数模块 [12], [13], [15]，但比特流生成单元的作用往往被忽视。大多数现有的随机计算设计都假设比特流生成器单元依赖由线性反馈移位寄存器生成的伪随机序列（图4(a)）。然而，要用较短的位流实现高精度（这对低延迟、高能效处理至关重要），就需要创新的位流生成技术。Sobol 生成器等低差异序列生成器 [17], [36], [37], 通过生成快速收敛的准随机比特流提供了颇具前景的替代方案，尽管其硬件成本较高（图4(b)）。提出的**TranSC** 框架采用另一种低差异序列来优化随机数生成器模块，从而提高设计效率和精度。我们证明，恰当使用新型低差异序列能够显著提升硬件效率和计算精度。下一节将介绍范德科普特（*VDC*）序列，这是一种特定类型的低差异准随机序列，并介绍其适用于硬件的实现方法。

A. 范德科普特（*VDC*）序列

VDC序列是一类在数量理论和差异理论中得到广泛研究的随机序列，尤其因其均匀分布性质而闻名 [38]。它们使用基数反演函数生成，该函数对任意基数 $\mathcal{B} \geq 2$ [39]均有定义。VDC序列的一个显著特征是其LD性质，可确保单位区间得到均匀覆盖。因此，它们在数值积分、准蒙特卡罗方法，以及计算机图形学和计算数学等应用中尤为重要 [40]。构造VDC序列时，将自然数映射到单位区间 [0,1) 上，并采用基数 $\mathcal{B}$ 展开（记作VDC-$\mathcal{B}$）。具体而言，将以基数 $\mathcal{B}$表示的自然数的各位数字反转，再将反转后的数字解释为小数值，即可得到每一项。此过程保证了模一等分布性质。例如，十进制数77在五进制中表示为$(302)_5$。反转各位数字后得到$(0.203)_5 = 2 \times 5^{-1} + 0 \times 5^{-2} + 3 \times 5^{-3} = \frac{53}{125} = (0.424)_{10}$。

B. *VDC*-2^n：一种准确且硬件友好的情况

受 VDC 序列这一通用概念的启发，我们为比特流生成单元采用了一种硬件友好的变体，即使用 2 的幂基数的VDC-2^n。生成 VDC-2^n 序列只需一个简单的模 -m 加法计数器，其中 $m \geq n$。该加法计数器可以异步或同步模式运行。该计数器的二进制输出被划分

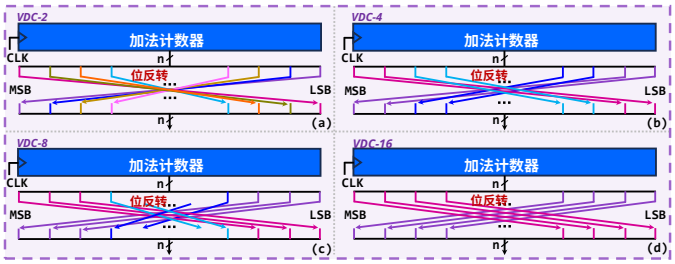

图 5. 使用二进制向上计数器的 VDC-2^n RNG 总体设计：(a) VDC-2，对计数器输出的每一位进行反转；(b) VDC-4，对计数器输出的每个 2 位子组进行反转；(c) VDC-8，对计数器输出的每个 3 位子组进行反转；(d) VDC-16，对计数器输出的每个 4 位子组进行反转。所有设计均按 n 位精度考虑——不同设计采用相同的电路，仅硬连线配置不同。

为若干组，每组包含 n 位；如果最后一组少于 n 位，则进行零填充，以确保其完整达到 n 位。随后通过硬连线对这些分组重新排序，使最低有效分组变为最高有效分组，反之亦然。接着，将所得的 m 位二进制数映射到 $[0, 1)$ 区间，从而生成所需的 VDC-2^n 序列。

将 VDC-2^n 集成到 **TranSC**中，可利用简单的 $\log_2 N$ 位加法计数器实现低成本位流生成，其中 N 表示比特流长度。计数器输出可灵活配置为所需的任意 2 的幂次基数。VDC-2^n 的一项关键优势是其灵活性：无需额外硬件，即可同时生成多达 $\log_2 N$ 个互不相同且相互独立的序列。图 5 (a)–(d) 分别展示了 VDC-2、VDC-4、VDC-8 和 VDC-16 的硬件设计。所有设计共用同一个计数器，但硬连线重排序逻辑各不相同。通过利用 VDC-2^n形式的低差异序列，**TranSC**实现了轻量且高效的位流生成器设计，为超越函数实现中主输入及中间阶段的标量系数提供了替代现有随机数生成器设计的有力方案。

Parhi 和 Liu [12]研究了利用多项式近似实现复杂超越函数的随机计算设计，其中每个多项式函数的正系数和负系数均按降序排列。Chu 等人 [13] 提出了一种基于相关性的随机计算实现方法，使用单极性比特流计算多项式函数。他们引入了一种 与非-与 结构，以更少的延迟元件实现三角函数。这两种方法都依赖霍纳法则进行多项式求值。

同样，我们提出的 **TranSC**将霍纳法则应用于目标函数的麦克劳林级数展开。在现有最先进设计中，使用两个独立的随机数生成器单元为输入和系数生成相互独立的位流。所有系数位流均由共享的
随机数生成器单元 [12]，[13]。为使中间位流去相关，这些工作采用了额外的延迟元件（例如 D触发器）。相比之下，**TranSC**通过利用 *VDC*-序列的 *LD* 和不同分布特性，消除了对此类延迟元件的需求。因此，输入和系数比特流均可由单个 VDC-2^n 比特流生成单元生成，从而显著降低硬件成本

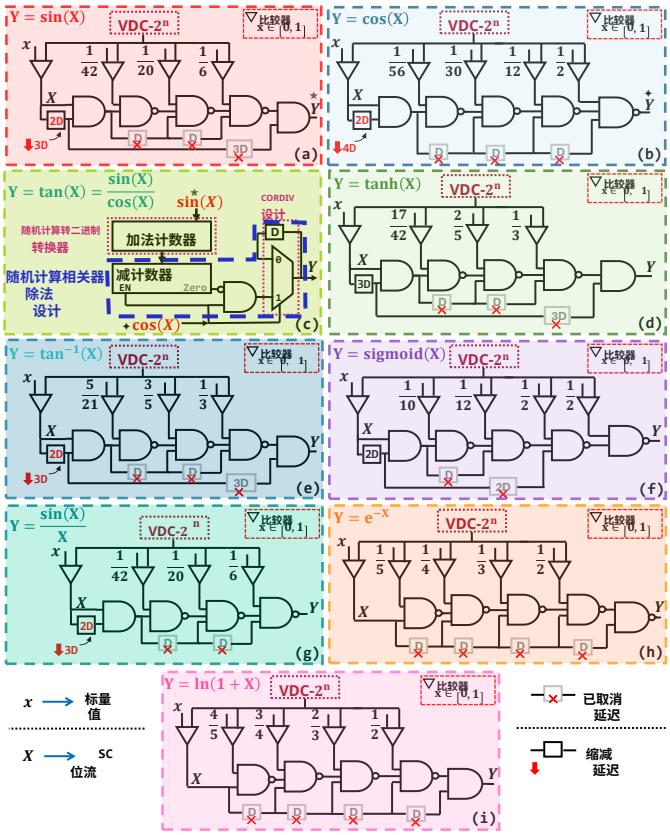

图 6. 所提出的 **TranSC** 超越函数框架：利用高效的 BSG 单元消除延迟元件。(a) **正弦**(x)，(b)**余弦**(x)，(c) **正切**(x)，(d) **双曲正切**(x)，(e) **反正切**(x)，(f) **Sigmoid 函数**(x)，(g) **Sinc函数**(x)，(h) e^{-x}，以及 (i) $\ln(1+x)$。

同时保持中间计算的精度和独立性。

图 6 展示了所提出的**TranSC** 框架中实现的超越函数集合。我们通过集成新开发的 BSG 输入，重新审视了以往最先进设计中的 CLB。在修改后的设计中，延迟模块（此前用于对位流进行去相关）要么被完全消除，要么显著减少。图6(a) 和 6(b) 分别展示了改进后的**正弦**(x) 和**余弦**(x) 函数设计。我们的设计与最先进方法的主要区别有两点：i）采用基于 VDC-2^n序列的多功能单源准随机 BSG 单元；ii）移除中间阶段的去相关模块（例如D-FF），从而降低硬件成本和计算延迟。

对于图 6(c) 所示的正切函数，我们采用一种高效的随机计算除法电路，根据正弦函数和余弦函数得出结果。该电路集成了一个相关器模块（由一个减计数器和一个与门组成）以及 CORDIV 架构 [41]。图 6(d)–(i) 展示了其他超越函数的 TranSC 设计，包括双曲正切函数、反正切函数、Sigmoid函数、Sinc函数、e^{-x} 和 $\ln(1+x)$。

V. 实验结果

基于 VDC-2^n 的 BSG 天生具有准随机性，能够在无需额外去相关组件的情况下，为中级电路元件提供所需的统计独立性，

表 II TranSC 与最先进方法实现超越函数的精度评估——第 1 部分（N：比特流长度）

函数	设计方法	N	随机数生成器来源		No. of 延迟元件				MSE ($\times 10^{-4}$)
			输入	系数	1st 阶段	2nd 阶段	3rd 阶段	4th 阶段	
正弦(x)	**TranSC***	1024	VDC-4	VDC-128 ,256,512	**2**	**0**	**0**	**0**	**0.523**
		512		VDC-128 ,256,512					0.582
		256		VDC-128					0.576
		128		VDC-2 ,8,128					0.967
		64		VDC-2 ,8,8					1.743
	TranSC ✦	1024	VDC-2	VDC-1024	1	**0**	**0**	**0**	0.888
		512		VDC-512					0.999
		256		VDC-256					1.221
		128		VDC-128					1.894
		64		VDC-64					5.373
	工作于 [12]	1024	Sobol-1	Sobol-2	3	1	1	3	3.133
	工作于 [13]	1024	Sobol-1	Sobol-2	1	1	1	1	0.917
	工作于 [12]	1024	LFSR1	LFSR2	3	1	1	3	2.256
	工作于 [13]	1024	LFSR1	LFSR2	1	1	1	1	5.036
余弦(x)	**TranSC** *	1024	VDC-8	VDC-8 ,4,16,256	**2**	**0**	**0**	**0**	**1.073**
		512		VDC-8 ,4,16,256					1.087
		256		VDC-8 ,4,16,256					1.096
		128		VDC-8 ,4,16,128					1.119
		64		VDC-8 ,4,16,64					3.815
	TranSC ✦	1024	VDC-4	VDC-16 ,4,8,32	1	**0**	**0**	**0**	1.342
		512							1.374
		256							1.352
		128							1.678
		64							1.484
	工作于 [12]	1024	Sobol-1	Sobol-2	4	1	1	1	42.243
	工作于 [13]	1024	Sobol-1	Sobol-2	1	1	1	1	1.343
	工作于 [12]	1024	LFSR1	LFSR2	4	1	1	1	2.518
	工作于 [13]	1024	LFSR1	LFSR2	1	1	1	1	4.934
正切(x)	**TranSC** *	1024	正弦(x):VDC-8 余弦(x):VDC-4	正弦(x): VDC-128,128,128 余弦(x): VDC-16 ,8,2,128	3 3	**0** **0**	**0** **0**	**1** 2	**0.721** 0.780 1.099 1.349
		512							
		256							
		128							
	TranSC ✦	1024	正弦(x):VDC-32 余弦(x):VDC-8	正弦(x): VDC-256,256,16 余弦(x): VDC-256,256,256,256	1 1	0 0	0 0	2 0	5.471 3.938 4.942
		512							
		256							
	工作于 [12]	1024	正弦(x):Sobol-1 余弦(x):Sobol-1	正弦(x):Sobol-2 余弦(x):Sobol-2	3 4	1 1	1 1	1 1	124.403
	工作于 [13]	1024	正弦(x):Sobol-1 余弦(x):Sobol-1	正弦(x):Sobol-2 余弦(x):Sobol-2	1	1	1	1	89.835
	工作于 [12]	1024	正弦(x):LFSR1 余弦(x):LFSR1	正弦(x):LFSR1 余弦(x):LFSR2	3 4	1 1	1 1	3 1	9.845
	在 [13] 1024中工作		LFSR1	LFSR2	1	1	1	1	4.564
双曲正切(x)	**TranSC** *	1024	VDC-16	VDC-32 ,16,2	3	**0**	**0**	**0**	**2.881**
		512							2.944
		256							3.273
		128							3.244
		64							5.800
	TranSC ✦	1024	VDC-2	VDC-4 ,64,64	1	**0**	**0**	**0**	3.164
		512							3.172
		256							3.190
		128							3.240
		64							3.392
	工作于 [12]	1024	Sobol-1	Sobol-2	3	1	1	3	7.731
	工作于 [13]	1024	Sobol-1	Sobol-2	1	1	1	1	2.390
	工作于 [12]	1024	LFSR1	LFSR2	3	1	1	3	6.579
	工作于 [13]	1024	LFSR1	LFSR2	1	1	1	1	12.548

*：将 **TranSC** 应用于 [12]的设计；将 **TranSC** 应用于 [13]的设计。不同的 VDC-2^n 基于简单的硬连线，均来自同一硬件源。

表 III **TranSC** 与最先进方法实现超越函数的精度评估——第 2 部分（N：比特流长度）

函数	设计方法	N	随机数生成器来源		No. of 延迟元件				MSE ($\times 10^{-4}$)
			输入	系数	1st 阶段	2nd 阶段	3rd 阶段	4th 阶段	
$sigmoid(x)$	**TranSC** *	1024	VDC-1024	VDC-2 ,4,32	2	**0**	**0**	-	**0.072**
		512							0.099
		256	VDC-512						0.418
		128	VDC-256						2.109
		64							9.648
	TranSC ✦	1024	VDC-128	VDC-4	1	**0**	**0**	-	0.151
		512							0.152
		256							0.153
		128							0.187
		64							0.547
	工作于 [12]	1024	Sobol-1	Sobol-2	2	1	2	-	189.310
	工作于 [13]	1024	Sobol-1	Sobol-2	1	1	1	-	282.834
	工作于 [12]	1024	LFSR1	LFSR2	2	1	2	-	2.903
	工作于 [13]	1024	LFSR1	LFSR2	1	1	1	-	0.383
e^{-x}	**TranSC** *	1024	VDC-128	VDC-16 ,1024,512,512	**0**	**0**	**0**	**0**	3.032
		512	VDC-128	VDC-128 ,512,512,512					3.517
		256	VDC-32	VDC-32 ,256,256,256					2.508
		128	VDC-8	VDC-64 ,128,128,128					3.738
		64	VDC-8	VDC-64 ,64,128,128					2.391
	TranSC ✦	1024	VDC-32	VDC-64	**0**	**0**	**0**	**2**	**1.730**
		512	VDC-32	VDC-64					2.647
		256	VDC-32	VDC-64					3.412
		128	VDC-32	VDC-4					2.996
		64	VDC-32	VDC-64					2.871
	工作于 [12]	1024	Sobol-1	Sobol-2	1	1	1	1	22.231
	工作于 [13]	1024	Sobol-1	Sobol-2	1	1	1	1	21.210
	工作于 [12]	1024	LFSR1	LFSR2	1	1	1	1	9.331
	工作于 [13]	1024	LFSR1	LFSR2	1	1	1	1	6.807
反正切(x)	**TranSC** *	1024	VDC-8	VDC-512,8,256	**2**	**0**	**0**	**0**	**0.835**
		512		VDC-512,8,256					0.854
		256		VDC-16,8,64					2.544
		128		VDC-128 ,8,64					2.600
		64		VDC-16,8,64					3.271
	TranSC ✦	1024	VDC-4	VDC-4,2,2	1	**0**	**0**	**0**	1.733
		512							1.853
		256							1.678
		128							2.336
		64							2.006
	工作于 [12]	1024	Sobol-1	Sobol-2	3	1	1	3	8.028
	工作于 [13]	1024	Sobol-1	Sobol-2	1	1	1	1	2.327
	工作于 [12]	1024	LFSR1	LFSR2	3	1	1	3	1.947
	工作于 [13]	1024	LFSR1	LFSR2	1	1	1	1	7.518
$Sinc(x)$	**TranSC** *	1024	VDC-8	VDC-256 ,32,1024	**2**	**0**	**0**	-	**0.124**
		512		VDC-256 ,32,512					0.186
		256		VDC-256 ,32,256					0.297
		128		VDC-4 ,64,128					0.386
		64		VDC-4 ,8,64					1.145
	TranSC ✦	1024	VDC-4	VDC-512	1	**0**	**0**	-	0.181
		512		VDC-256					0.258
		256		VDC-128					0.365
		128		VDC-128					0.416
		64		VDC-64					2.917
	工作于 [12]	1024	Sobol-1	Sobol-2	3	1	1	-	4.324
	工作于 [13]	1024	Sobol-1	Sobol-2	1	1	1	-	3.450
	工作于 [12]	1024	LFSR1	LFSR2	3	1	1	-	7.081
	工作于 [13]	1024	LFSR1	LFSR2	1	1	1	-	1.952
$\ln(1+x)$	**TranSC** *	1024	VDC-64	VDC-4 ,512,1024,512	**0**	**0**	**0**	**0**	0.996
		512	VDC-64	VDC-4,256,512,256					2.482
		256	VDC-16	VDC-64,128,2					1.367
		128	VDC-8	VDC-32 ,64,2					3.470
		64	VDC-4	VDC-8 ,32,2					4.973
	TranSC ✦	1024	VDC-16	VDC-32 ,64,256,256	**0**	**0**	**0**	**0**	**0.734**
		512	VDC-16	VDC-32 64 256 256					1.023
		256	VDC-16	„ , VDC-32,256,256					0.979
		128	VDC-16	VDC-32,8,128,128					2.277
		64	VDC-4	VDC-2 ,32,64,64					1.593
	工作于[12]	1024	Sobol-1	Sobol-2	1	1	1	1	15.262
	工作于[13]	1024	Sobol-1	Sobol-2	1	1	1	1	17.258
	工作于[12]	1024	LFSR1	LFSR2	1	1	1	1	4.131
	工作于[13]	1024	LFSR1	LFSR2	1	1	1	1	3.294

*：将**TranSC** 应用于 [12]、✦的设计；将**TranSC** 应用于 [13]的设计。不同的VDC-2^n 基序列可通过简单的硬连线由同一硬件源生成。

的去相关组件。我们的评估表明，**TranSC** 与最先进设计相比，精度显著更高。表格II 和III 展示了**TranSC**设计与最先进设计之间的精度对比分析。

对于最先进水平的基准方法，我们使用最大周期的线性反馈移位寄存器（由多项式 $x^{10}+x^8+x^6+1$和 $x^{10}+x^8+x^5+x^4$定义的两个线性反馈移位寄存器），并将每个操作执行 1，000 次，以确保结果具有统计显著性。此外，我们还使用 Sobol序列测试了最先进设计（通过 MATLAB 内置的 Sobol序列生成器生成），并将其分别应用于输入和系数 BSG。然而，将这些准随机序列纳入最先进设计会降低性能，因为延迟元件破坏了序列的独立性。我们的研究结果表明，当在随机计算实现中使用准随机数生成器时，在超越函数的实现中，必须尽量减少或消除延迟元件，以保持独立性。由于硬件成本高昂，**TranSC**未采用Sobol 序列。

TranSC **框架中的输入 BSG** 采用不同的 VDC-配置进行参数化，详见表 II和 III。系数 BSG 模块会生成位流，用于表示每个超越函数的函数专用系数

表 IV 超越函数实现的**TranSC** 与最先进设计的硬件成本比较

函数	N	TranSC ∗				TranSC ✦				工作于 [12]				工作于 [13]			
		Area (μm^2)	CPL (ns)	功率 (μW)	能量 (pJ)	Area (μm^2)	CPL (ns)	功率 (μW)	能量 (pJ)	Area (μm^2)	CPL (ns)	功率 (μW)	能量 (pJ)	Area (μm^2)	CPL (ns)	功率 (μW)	能量 (pJ)
正弦(x)	256	**435**	0.42	**706.5**	**76.0**	**430**	0.42	**666.0**	**71.6**	655	0.39	1965.3	196.2	618	0.39	1786.0	178.3
双曲正切(x)	512	**497**	0.43	**759.7**	**167.3**	**487**	0.43	**740.6**	**163.0**	734	0.40	2092.7	428.6	683	0.40	1914.5	392.1
反正切(x)✲	1024	**554**	0.44	**812.8**	**366.2**	**544**	0.44	**777.9**	**350.5**	801	0.42	2178.2	936.8	757	0.41	2042.6	857.6
余弦(x)	256	**489**	0.44	**761.7**	**85.8**	**479**	0.45	**705.5**	**81.3**	723	0.38	2159.3	210.0	692	0.41	1887.5	198.1
	512	**575**	0.46	**767.7**	**180.8**	**565**	0.46	**704.2**	**165.8**	797	0.39	2296.3	458.5	766	0.41	2041.6	402.3
	1024	**632**	0.44	**829.8**	**390.9**	**621**	0.46	**780.5**	**367.6**	883	0.42	2284.0	982.3	852	0.42	2180.8	937.9
正切(x)	256	**1286**	0.54	**1826.7**	**252.5**	**1273**	0.53	**1783.6**	**242.0**	1728	0.49	3600.4	451.6	1653	0.48	3594.7	441.7
	512	**1513**	0.54	**1925.7**	**532.4**	**1493**	0.55	**1860.8**	**524.0**	1924	0.49	4178.0	1048.2	1850	0.45	4326.1	996.7
	1024	**1630**	0.55	**1994.8**	**1123.5**	**1609**	0.53	**2040.9**	**1107.6**	2092	0.49	4473.5	2244.6	2018	0.47	4540.0	2185.0
$sigmoid(x)$	256	**490**	0.44	**739.8**	**83.3**	**480**	0.45	**671.0**	**90.7**	707	0.39	1969.6	222.5	687	0.38	1921.4	196.4
	512	**567**	0.45	**835.1**	**192.4**	**557**	0.45	**787.0**	**181.3**	785	0.40	2172.9	463.7	764	0.38	2267.3	463.6
	1024	**628**	0.47	**858.9**	**413.4**	**618**	0.46	**830.0**	**391.0**	852	0.41	2264.3	989.0	831	0.40	2355.8	989.0
e^{-x}	256	**481**	0.43	**653.7**	**72.0**	**481**	0.43	**653.7**	**72.0**	701	0.37	2064.0	195.5	701	0.38	1981.7	192.8
	512	**537**	0.50	**613.0**	**156.9**	**537**	0.50	**613.0**	**156.9**	778	0.39	2195.5	438.9	778	0.39	2198.0	438.9
	1024	**602**	0.46	**734.9**	**346.2**	**602**	0.46	**734.9**	**346.2**	845	0.40	2302.2	943.0	845	0.40	2399.9	983.0
$Sinc(x)$	256	**432**	0.41	**742.6**	**78.0**	**422**	0.41	**682.2**	**71.6**	619	0.38	1900.2	184.9	598	0.37	1842.5	174.5
	512	**494**	0.42	**764.8**	**164.5**	**483**	0.42	**712.8**	**153.3**	698	0.40	1978.8	405.3	678	0.37	2039.5	407.2
	1024	**554**	0.43	**809.7**	**338.0**	**544**	0.43	**767.7**	**338.0**	757	0.39	2288.4	958.0	736	0.39	2154.1	860.3
$\ln(1+x)$	256	**475**	0.45	**620.6**	**71.5**	**475**	0.45	**620.6**	**71.5**	699	0.38	1938.5	193	701	0.38	1983.3	193
	512	**551**	0.45	**736.0**	**169.6**	**551**	0.45	**736.0**	**169.6**	779	0.41	2105.2	450.6	779	0.39	2256.5	450.6
	1024	**608**	0.45	**744.6**	**343.1**	**608**	0.45	**744.6**	**343.1**	846	0.40	2373.4	972.1	846	0.40	2332.0	955.2

∗ 与 ✦ 分别是 Works 在 [12] 和 [13], 中的修改版本。✲：这些函数的硬件设计相同；系数不同。功耗是在最高频率下报告的。

。对于每个测试用例，我们评估 $[0, N]$ 区间内的所有可能输入值，对应于 $\log_2 N$ 位精度。我们针对输入位流和系数比特流探索多种 VDC-2^n 序列，并报告均方误差（MSE）以及电路任意阶段所用去相关延迟元件的总数量。

如图所示，**TranSC** 设计消除了所有中级延迟元件，这与以往的方法不同。我们的方法的一项关键优势在于，即使比特流长度显著缩短，也能保持较高精度。表II 和 III清楚地展示了这一点：**TranSC** 即使将比特流长度缩短至 $N = 64$，其性能仍优于最先进设计。此外，现有设计需要两个独立的 BSG（例如表II 和 III所示的 LFSR1/LFSR2 或 Sobol-1/Sobol-2），而**TranSC** 仅用一个比特流生成单元即可实现独立性，无需额外硬件。例如，在基于**TranSC**的方法设计正弦(x)函数时，使用 $_{1\,42}$、$_{1\,20}$和 $_{1\,6}$时，VDC-128、VDC-256 和 VDC-512 序列分别分配给这些系数 $N = 1024$, $\overline{512}$（表II 中正弦(x) 函数对应的第一行和第二行）。

我们将所提出的**TranSC** 框架与用于实现各种超越函数的最先进设计进行硬件成本比较。我们使用 Synopsys Design Compiler v2018.06 对所有设计进行综合，并采用 45nmFreePDK 门级库。表 IV汇总了各设计的综合结果，包括面积、关键路径延迟（CPL）、最高工作频率下的功耗以及能耗。如表所示，**TranSC**通过采用高效的 BSG 作为随机计算架构的核心，始终优于最先进方法。通过消除中间级的延迟元件，并将多个 BSG 集成到单个单元中，**TranSC**显著提升了硬件效率。具体而言，所提出的设计最多可将硬件面积、功耗和能耗分别降低 33%、72% 和 64%，分别，同时保持计算精度。

VI. 概念验证

A. 性能分析

通过采用各自独立的 BSG，所提出的**TranSC** 框架无需在中间级使用去相关器模块（即延迟元件），因此与最先进设计相比，硬件效率更高。图7 展示了所实现的精度提升（纵轴）
在各种超越函数（x轴）上，**TranSC**优于最先进水平的实现，且采用固定的比特流长度 $N = 1024$。

为进一步验证**TranSC** 设计的有效性，我们评估品质因数 *(FoM)*，这是一项综合指标，结合精度和硬件 (HW) 成本，以反映整体设计效率。品质因数 定义如下：

$$FoM = \frac{Accuracy}{HWCosts} = \frac{\frac{1}{MSE}}{Area \times Power \times Latency} \quad (2)$$

该指标综合考虑均方误差、硬件占用、功耗和延迟，为比较不同设计提供了统一依据。图 8展示了品质因数在各种设计中的评估结果，数据摘自表II、III，

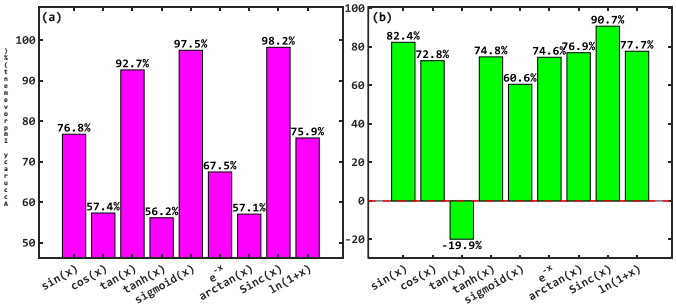

图 7. 精度提升概览：(a) **TranSC** 相较于 [12]中的工作，以及 (b)**TranSC** 相较于 [13]中的工作。

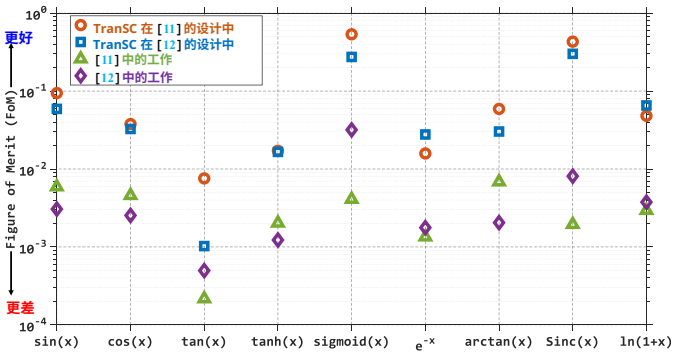

图 8. 不同设计方法的性能指标比较。

和IV。根据公式 2的定义，较高的品质因数值表示整体性能更优，这得益于更高的计算精度和更低的硬件成本。

B. 为什么 *VDC-*2^n 表现出色?

为进一步解释由 VDC-**序列驱动的** TranSC2^n 设计为何优于最先进方法，我们评估了 SCC （公式 1）和零相关误差（ZCE）这两项指标。 ZCE 用于量化 SC 比特流之间的独立程度；对于具有最大独立性的配置，该指标的值为零 [42]。其形式化定义如下:

$$ZCE = \Delta \cdot \left(1 - \frac{\Delta_0}{\Delta}\right) \tag{3}$$

其中， $\Delta = \frac{a}{N} - \frac{(a+b)(a+c)}{N^2}$，且 $\Delta_0 = \frac{\lfloor (a+b)(a+c) \rfloor}{N^2} - \frac{\lfloor \frac{a+c}{N} + \frac{1}{2} \rfloor}{N}$ $(a+b)(a+c)$ N^2。与 SCC指标类似， ZCE 公式中的值 a、 b、c 和 d分别表示从两个位流对应比特位置获得的逻辑对 11、10、01 和 00 的出现次数。我们使用 SCC 和 ZCE 这两种指标，通过一个代表性案例——正弦函数——评估设计性能。图 9(a) 和 (b) 对比了传统方法 [12] 与所提出的 TranSC 设计，并分别展示了各方法的中间计算项，如下所示:

$$\begin{aligned} i_1 &= X^2 \\ i_2 &= 1 - \frac{1}{42} i_1 = 1 - \frac{1}{42} X^2 \\ i_3 &= 1 - \frac{1}{20} i_1 i_2 = 1 - \frac{1}{20} X^2 (1 - \frac{1}{42} X^2) \\ i_4 &= 1 - \frac{1}{6} i_1 i_3 = 1 - \frac{1}{6} X^2 (1 - \frac{1}{20} X^2 (1 - \frac{1}{42} X^2)). \end{aligned} \tag{4}$$

图9(c) 展示了 SCC 每个中间位流的测量结果（以归一化概率表示）。为确保公平比较并与最先进设计保持一致，我们采用长度为 1024 的 SC 比特流，并遵循表 II 中指定的设计配置。这些测量结果是通过评估 $[0, 1]$ 区间内所有可能的 10 位精度输入值获得的。

在第一个中间级（i_1）中，传统（最先进水平）方法的零相关性（$SCC = 0$）至多为 20%，而**TranSC**设计在所有位流中始终实现最小相关性（$SCC = -1$）。在第二个

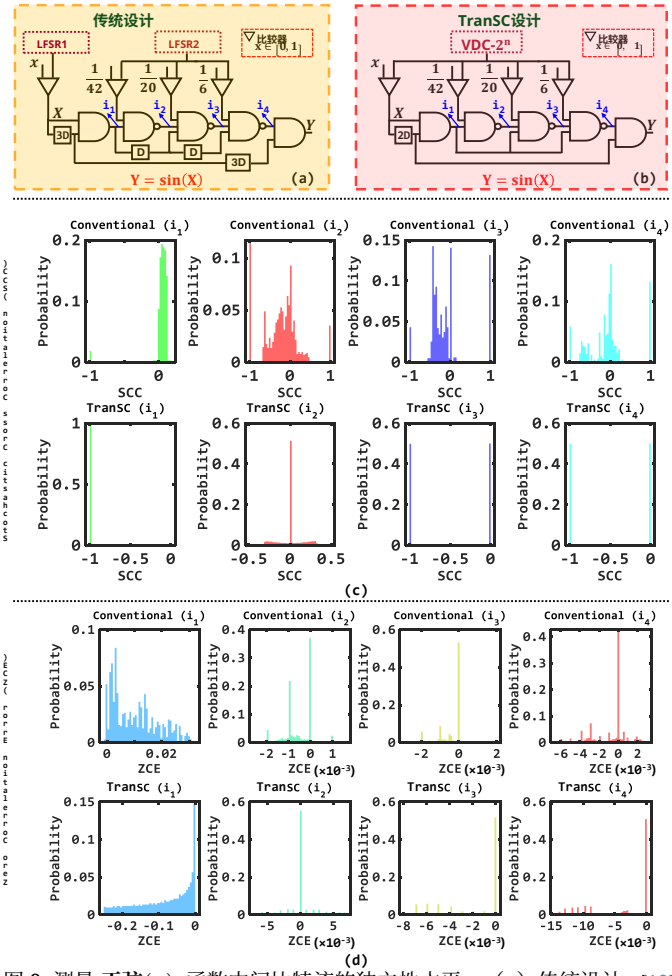

图 9. 测量 **正弦**(**x**) 函数中间比特流的独立性水平。（a）传统设计 [12]，（b）**TranSC** 设计，（c）SCC 测量，以及（d）ZCE 测量：针对 **TranSC** 和传统方法，对 i_1、 i_2、 i_3和 i_4。

中间级（i_2）中，传统设计的 SCC 值广泛分布在 -1 至 $+1$范围内，表明位流之间存在显著依赖性。相反，**TranSC** 设计的 SCC 值紧密聚集在 0 附近，离散程度极小，表明该阶段位流之间具有出色的去相关效果。对于第三和第四个中间级（i_3， i_4），传统设计的 SCC 分布仍然很宽——部分值聚集在 0 附近，但在整个（-1 至 $+1$）范围内都有显著波动。与此同时，**TranSC**设计在 i_3 和 i_4 阶段维持完全独立（$SCC = 0$）或最小相关性（$SCC = -1$）。值得注意的是，只有当传统方法明确使用去相关元件来确保独立性时，**TranSC**设计中观察到的 SCC 趋势才与传统方法一致。最先进水平设计的 SCC 图中的波动模式，归因于基于线性反馈移位寄存器的生成器固有的随机性和相关性漂移。相比之下，**TranSC**框架利用 VDC 序列的低差异特性，避免了此类不一致，并实现稳定的流间相关性。

图9（d）展示了 ZCE 图，作为评估sin(**x**）函数设计中间级 SC 比特流独立性的另一项指标。在第一个

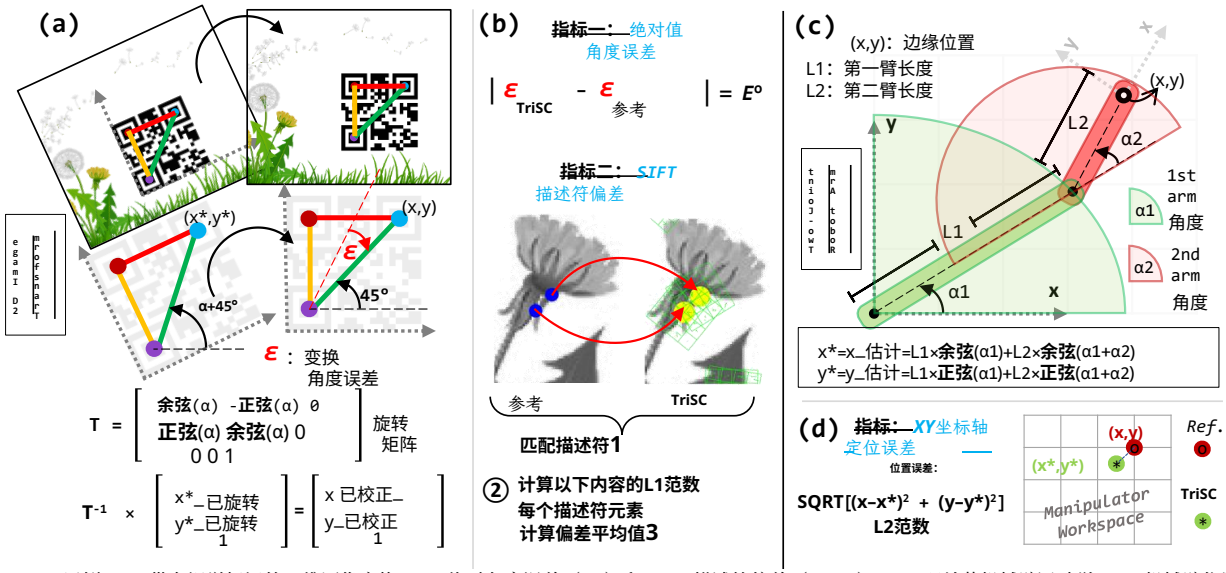

图10. TranSC 用例：(a) 带有视觉标记的二维图像变换，(b) 绝对角度误差（$E°$）和 SIFT 描述符偏差（SIFTσ），(c) 双关节机械臂运动学，(d) 机械臂位置估计误差（位置误差）。

中间级（i_1）中，传统设计的 ZCE 值峰值接近 0，但明显分散至较大的正误差。相比之下，**TranSC** 设计的 ZCE 值紧密聚集在 0 附近，变化极小，表明其独立性更强。在第二级（i_2）中，传统设计的 ZCE 值分布范围很广，峰值接近 0，但大量分散至正负两端。与此同时，**TranSC** 设计的 ZCE 值仍紧密集中在 0 附近，偏差极小。对于第三级（i_3），传统设计呈现出更广泛的 ZCE 分布，覆盖了很大的值域。相比之下，**TranSC** 设计的 ZCE 分布仍较为集中，聚集在 0 附近，表明其流间去相关效果更好。最后，在第四级（i_4）中，传统设计的 ZCE 值变化显著，既在 0 附近出现峰值，也存在正负方向上的明显离群值。相比之下，**TranSC**设计的 ZCE 值仍密集聚集在 0 附近，偏差极小，凸显了其在所有计算阶段持续保持流间独立性的能力。

这些分析共同验证了所提出的**TranSC**设计的有效性，证明它是一种轻量级且精度极高的解决方案，可利用随机计算范式高效实现超越函数。在此基础上，我们接下来将展示**TranSC** 在真实场景中的实际应用价值——首次将基于随机计算的设计引入图像变换和机器人学领域。

VII. 用例研究

本节评估所提出的**TranSC** 设计方法在两种新颖应用中的性能。这两种应用首次在随机计算文献中得到探索：①二维图像变换 和 ②机械臂运动学。

二维图像变换是快速响应（QR）码图像校正中广泛采用的一项基本操作，在移动机器人导航和相机姿态标定中发挥着重要作用。这些任务通常需要计算密集型三角运算，因此对低成本硬件实现提出了要求 [43]。图10（a）展示了静态图像中 QR 码的方形定位图案，这些图案用于确定参考旋转角。典型的 QR 码在左上角、右上角和左下角各设有一个定位图案。连接这些图案的对角线（以绿色表示）与 45° 轴形成夹角 x；任何偏离这一理想角度的情况 α 都表示图像发生了旋转。为了校正这种旋转，需要应用一个变换矩阵 T，该矩阵包含平移、缩放、错切、反射和旋转等常见图像操作。具体而言，T 中的旋转操作涉及正弦函数和余弦。逆变换^1_T 对于通过补偿任意旋转角来校正图像方向至关重要 α。传统的 QR 码处理同样采用T 的逆变换，以逆转图像采集过程中产生的任何旋转，从而将图像恢复到正确的对齐状态。图10（a）展示了旋转 T 矩阵，其中采用了正弦和余弦运算。逆运算用于确定校正后的像素位置$x\ corrected$ 和 $y\ corrected$。

图10(b) 展示了我们用于评估性能的指标。 绝对角度误差 根据变换角度误差，评估随机计算设计（本文提出的设计和最先进水平）相较于参考二进制模型的性能 ϵ。图像校正后， ϵ 表示与预期角度之间的偏差，如图 10(a) 所示。*SIFT*描述符偏差用于评估基于尺度不变特征变换（SIFT）的描述符的偏差。我们将变换后的输出图像与参考图像进行比较，并使用 $\mathcal{L}1$范数分析匹配描述符，以量化采用所实现设计进行图像变换时的平均变化。

我们还将基于随机计算的三角函数设计应用于机器人运动学应用中。图10(c) 展示了涉及机械手的双关节机械臂的操作

表 VTRANSC 在不同用例中的性能

设计方法	用例 I：图像定向			用例 II：机器人定位	
	角度误差 (E°) & SIFT 偏差 ($SIFT_\sigma$)			位置估计误差 (位置误差)	
	N=256	N=512	N=1024	N=512	N=1024
TranSC*	E°: **0.091** $SIFT_\sigma$: **6.368**	E°: **0.084** $SIFT_\sigma$: **6.367**	E°: **0.068** $SIFT_\sigma$: **6.366**	位置误差:**0.158**	位置误差:**0.157** 位置误差: **0.155**
TranSC ✦	E°: **0.177** $SIFT_\sigma$: **6.457**	E°: **0.164** $SIFT_\sigma$: **6.405**	E°: **0.163** $SIFT_\sigma$: **6.373**	位置误差: **0.176**	位置误差: **0.174** 位置误差: **0.173**
工作于 [12]	E°: 0.386 $SIFT_\sigma$: 6.594	E°: 0.207 $SIFT_\sigma$: 6.531	E°: 0.178 $SIFT_\sigma$: 6.448	位置误差: 0.546	位置误差: 0.542 位置误差: 0.541
在 [13]中的工作	E°: 0.858 $SIFT_\sigma$: 6.602	E°: 0.652 $SIFT_\sigma$: 6.597	E°: 0.493 $SIFT_\sigma$: 6.543	位置误差: 0.756	位置误差: 0.636 位置误差: 0.468

: TranSC 应用于 [12], ✦: TranSC 应用于 [13]。表 II 中性能最佳的 BSG 用于 TranSC 和 TranSC✦。

系统具有两根长度分别为 $L1$ 和 $L2$的连杆。 α_1和 α_2 角度定义了图中绿色和红色阴影区域所示的连杆运动范围，坐标为 xy。机械手边缘点$(x,\ y)$ 通过涉及正弦和余弦函数的正向运动学方程进行估算。使用随机计算操作计算边缘位置时，我们将估算点$(x^*,\ y^*)$ 与二进制计算的参考结果进行比较，以求得图位置误差（PErr），见图10(d)。

表 V 展示了两个已实现用例的性能结果。对于第一个用例，我们考察了一个在真实环境中拍摄的二维码图像数据集 [44]，并报告平均角度误差和 SIFT描述符偏差。对于第二个用例，我们分析了所有可能的笛卡尔坐标对的连杆导航范围 $(\alpha_1,\ \alpha_2)$，并报告平均位置误差。可以看出，在所有测试场景中，对于从 256 到 1024 的不同位流长度，我们提出的架构都优于最先进设计。

VIII. 讨论

A. 与最先进的随机数生成器的比较

Sobol序列以其出色的均匀性和在随机计算设计中的卓越性能而闻名。我们采用 VDC-2^n 序列，并非因为它们在所有情况下都更加均匀，而是因为它们更适合 **TranSC** 框架的架构需求。**TranSC** 的首要目标是消除中间阶段的去相关元件（D触发器），这些元件一直是以往最先进设计中硬件成本和延迟的主要来源。我们通过从单一共享硬件源生成多个彼此不同且统计独立的位流，分别用于函数输入和多项式系数，从而消除这些元件。VDC-2^n 生成器的硬件架构——一个带有可配置位反转硬连线的简单向上计数器——极其轻量。它几乎无需额外硬件成本，即可同时生成多达 $log_2 N$ 个彼此不同且独立的序列。相比之下，Sobol序列生成器复杂得多，需要地址生成器、用于存储方向向量的存储器以及异或逻辑网络 [17], [21]。若要生成多个相互独立的Sobol流，就必须复制

表 VI 考虑 8位精度随机数生成器来源时随机计算位流生成的硬件实现成本

随机数生成器类型	Area (μm^2)	最大频率下的功耗 (μW)	CPL (ns)
LFSR	233	949.2	0.37
Sobol有限状态机 [45]	281	720.4	0.36
VDC-2^n	201	544.4	0.34

最大 周期采用多项式 $x^8 + x^6 + x^3 + x^2$，用于 线性反馈移位寄存器。

这种复杂逻辑，或采用更加复杂的多维生成器；两种做法都会大幅增加硬件成本。相比之下，VDC-2^n 序列使 **TranSC** 框架能够以远低于基于 Sobol 的替代方案的硬件成本，获得高效的比特流生成单元。

表II和III中的结果表明，**TranSC** 设计具有更高的精度和更强的去相关性，这一点可从图9中较低的 SCC 和 ZCE值看出。这些结果证实，VDC-2^n 序列所提供的均匀性足以确保已实现设计所需的独立性水平。表VI比较了三种方法的位流生成硬件成本：基于线性反馈移位寄存器的生成器、最先进水平的基于有限状态机的 Sobol 生成器，以及基于 VDC-2^n的位流生成器。结果表明，与基于有限状态机的 Sobol [45]生成器和基于线性反馈移位寄存器的位流生成器相比，基于 VDC 的生成器面积分别减少约 28% 和 14%，功率分别降低 24% 和 43%。当需要多个相互独立的序列时，这一硬件优势会更加显著，因此，采用 VDC-2^n 作为轻量级**TranSC** 架构的核心赋能技术是合理的。

B. 与传统二进制设计的比较

*1) **TranSC** 与坐标旋转数字计算（CORDIC）比较：* 坐标旋转数字计算（CORDIC）算法 [46] 是一种经典且应用广泛的“移位加法”技术，用于计算超越函数，在没有专用乘法器的硬件上尤其高效，例如现场可编程门阵列或简单的微控制器。作为一种迭代方法，CORDIC 的延迟会随所需精度而变化。**TranSC**的精度和延迟由N决定，其硬件实现则由 BSG 和简单的组合逻辑构成。两者的主要区别在于目标应用：CORDIC 适用于要求高且精度有保证、并且可以接受迭代延迟的系统。相比之下，**TranSC** 针对以下场景进行了优化：超低功率、最小硬件面积和高容错性是首要设计要求，且能够接受随机计算固有的精度损失。文中给出的应用案例凸显了这种权衡特别有优势的领域。

*2) **TranSC** 与分段多项式插值（PPI）的比较：* PPI 方法是另一类主要的函数求值器。在这类技术中，输入范围被划分为较小的区间，并使用低次多项式（通常以系数形式存储在查找表中）来近似每个区间内的函数。通过增加区间数量或

表 VII 跨范式性能与资源基准测试：10 位二进制精度的正弦(x)函数

方法	Area (μm^2)	功耗@最大频率(μW)	CPL (ns)	能量 (pJ)	MSE $(\times 10^{-4})$
坐标旋转数字计算	6562	14500	0.60	8.7	$\ll 0.01$
PPI	4419	5500	0.90	4.9	$\ll 0.005$
TranSC	**554**↓	**812.8**↓	**0.44**↓	366.2↑	0.523↑

坐标旋转数字计算采用展开的 10 次迭代硬件实现。PPI 使用只读存储器和乘累加单元，专为二阶多项式设计。

多项式次数 [47]。与 **TranSC**的比较揭示了硬件设计中的一项根本性权衡。PPI 方法依赖传统的二进制算术单元（例如乘法器和加法器）以及存储块，其硬件成本会随着所需精度和多项式复杂度的提高而增加。相比之下，**TranSC**的硬件主要由 BSG 和简单的位逻辑构成，因此提供了一种轻量级替代方案。

表 VII 对比测试 **TranSC** 与坐标旋转数字计算和二阶 PPI 方法在 正弦(x)函数上的表现，所有设计的目标输入精度均相当于 10 位。虽然坐标旋转数字计算和 PPI 的精度显著更高（均方误差更低），但这需要付出硬件面积和功耗大幅增加的代价。 **TranSC** 的面积几乎 $12\times$ ，且 $8\times$更小，功耗比坐标旋转数字计算和 PPI 设计分别 $18\times$ ，并且 $7\times$ 更低。它唯一的缺点是能耗较高，这是随机计算设计的已知局限，源于位流的串行处理。不过，通过并行处理可以大幅缓解这种低效。总体而言，这一量化比较表明，**TranSC** 并非能够全面取代高精度传统二进制方法，而是适用于特定应用的一种有竞争力的解决方案——这类应用受到严格的资源限制，并且能够容忍适度的精度折衷，以换取效率和鲁棒性的显著提升。

IX. 结论

本文介绍了一种利用具有准随机范德科普特（VDC）序列的随机计算（SC）实现超越函数的硬件高效方法。所提出的**TranSC**框架无需使用去相关元件，因此与最先进的随机计算设计相比，既降低了硬件成本和延迟，又提高了精度。我们通过对多种函数进行广泛评估，展示了该方法的有效性，其中包括随机计算在机器人视觉和机动领域的两项新颖应用。结果表明，**TranSC**有望用于依赖三角函数和超越函数计算的轻量、高效系统。未来的工作将探索端到端随机计算架构的开发，尤其是在神经网络中的应用，并将我们针对 Sigmoid 激活函数和对数函数等非线性函数的设计纳入其中。这些进展或可进一步拓展随机计算在机器人学习和低功耗人工智能系统等新兴领域的适用性。

参考文献

[1] S. Aygun、M. S. Moghadam 和 M. H. Najafi，“最新成果：TriSC：低成本三角函数设计，采用《准随机计算》，载于《第61届 ACM/IEEE设计自动化会议论文集》，系列：DAC'24，2024年。 [在线]。网址：https://doi.org/10 1145 3649329 3663490。/。A. M. Dalloo、A. J. Humaidi、[2] A. K. A. Mhdawi 和 H. Al-Raweshidy，“面向嵌入式系统应用的双曲函数和指数函数近似低功耗、低延迟硬件实现”，IEEE Access，第12卷，第24 151–24 163页，2024年。[3] A. Li、S. L. Song、M. Wijtvliet、A. Kumar 和 H. Corporaal，“基于图形处理器流处理单元的透明近似加速”，载于《2016年国际超级计算会议论文集》，系列：ICS '16。美国纽约：计算机协会，2016年。[ ] 10 1145 2925426 2926255 在线。网址：[4] https://doi.org/。/。A. Alaghi 和 J. P. Hayes，“随机计算综述”，ACM Trans. Embed. Comput. Syst.，第12卷，第2s期，2013年5月。 [在线]。网址：https://doi.org/ 10 1145 2465787 2465794 。/。S. [5] Liu、J. L. Rossello、S. Liu、X. Tang、J. Font-Rossell´o、C. F. Frasser、W. Qian、J. Han、P. Reviriego 和 F. Lombardi，“从乘法器到积分器：随机计算原语综述”，IEEE TNANO.，第23卷，第238–249页，2024年。[6] S. Liu、W. J. Gross 和 J. Han，“动态随机计算简介”，IEEE电路与系统杂志，第20卷，第3期，第19–33页，2020年。[7] S. Aygun、E. O. Gunes 和 C. De Vleeschouwer，“二值化神经网络中的高效稳健位流处理”，电子学快报，第57卷，第5期，第219–222页，2021年。[8] H. Joe 和 Y. Kim，“面向高能效图像处理器的新型随机计算”，Electronics，第8卷，第6期，2019年。 [在线]。网址：https://www.mdpi.com/ 2079 9292 8 6 720 - / / / N. [9] Temenos 和 P. P. Sotiriadis，“随机计算非缩放加法器建模及其在图像锐化中的应用”，IEEE TCAS II，第69卷，第5期，第2543–2547页，2022年。[10] H. Zhou、S. P. Khatri、J. Hu 和 F. Liu，“用于高效随机计算的缩放总体算术”，载于2020年第25届亚洲及南太平洋设计自动化会议，2020年。[11] H. Li 和 Y. Chen，“混合随机数及其神经网络计算”，IEEE TVLSI，第32卷，第03期，第432–441页，2024年3月。[12] K. K. Parhi 和 Y. Liu，“利用级数展开，通过随机逻辑计算算术函数”，IEEE TETC，第7卷，第1期，第44–59页，2019年。[13] S.-I. Chu、C.-L. Wu、T. N. Nguyen 和 B.-H. Liu，“采用单极性随机逻辑和相关性技术进行多项式计算”，IEEE TC，第71卷，第6期，第1358–1373页，2022年。[14] A. Khataei、G. Singh 和 K. Bazargan，“Simbu：面向非线性函数的基于自相似性的混合二进制—一进制计算”，IEEE TC，第73卷，第9期，第2192–2205页，2024年。[15] S. Wang、G. Xie、X. Cheng 和 Y. Zhang，“利用相关单极性随机位流进行基于加权加法器的多项式计算”，IEEE TCAS II，第69卷，第11期，第4528–4532页，2022年。[16] Y. Liu 和 K. K. Parhi，“使用单极性随机逻辑计算多项式”，新兴计算技术与系统期刊，第13卷，第3期，2017年4月。2017 [ ] 10 1145 3007648 在线。网址：[17] https://doi.org/。/。S. Liu 和 J. Han，“利用并行Sobol序列实现节能型随机电路”，IEEE TVLSI，第26卷，1339，2018 7，1326 no. pp. –。[18] T. J. Baker 和 J. P. Hayes，“Cemux：最大限度提高随机多路复用器加法器的精度及其在滤波器设计中的应用”，ACM TO-DAES，第27卷，第3期，2022年。[19] S.-I. Chu，“随机计算的新型除法器设计”，IEEE TCAS II，第67卷，第1期，第147–151页，2020年。[20] J. H. Anderson、Y. Hara-Azumi 和 S. Yamashita，“线性反馈移位寄存器播种、扰码和反馈多项式对随机计算精度的影响”，载于2016年欧洲设计、自动化与测试会议暨展览。[21] M. H. Najafi、D. Jenson、D. J. Lilja 和 M. D. Riedel，“确定性地执行随机计算”，IEEE TVLSI，第27卷，第12期，第2925–2938页，2019年。[22] A. Alaghi 和 J. P. Hayes，“在随机电路设计中利用相关性”，载于国际计算机设计会议，美国北卡罗来纳州阿什维尔，2013年，第39–46页。[23] D. Wu、R. Yin 和 J. S. Miguel，“随机计算中基于流内相关性的除法和位插入平方根运算”，IEEE设计与测试，第38卷，第6期，第53–59页，2021年。[24] Q. Zhang、Y. Chen、S. Li、X. Zeng 和 K. K. Parhi，“通过相关性分析设计的高性能随机LDPC译码器架构”，IEEE TCAS I，第67卷，第12期，第5429–5442页，2020年。[25] T.-K. Luong、V.-T. Nguyen、A.-T. Nguyen 和 E. Popovici，“基于随机计算的算术函数近似的高效架构与实现”，载于IEEE应用专用处理器国际会议，2019年，第281–287页。[26] L. Huai、P. Li、G. E. Sobelman 和 D. J. Lilja，“三角函数和双曲函数的随机计算实现”，载于IEEE第12届ASICON，2017年，第553–556页。

[27] Y. Liu 和 K. K. Parhi，“利用因式分解在单极性随机逻辑中计算复函数”，系列：GLSVLSI ’16。美国纽约：计算机协会，2016年，第 109 112 [ ] 10 1145 2902961 2902999–页。[28] 在线。网址：https://doi.org/。/。R. Chen、L. Hei 和 Y. Lai，“基于深度学习的物联网光学成像目标检测”，PeerJ Comp. Sc.，第9卷，文章编号 e1718，2023年。[29] Z. Wang、Z. Xu、D. He 和 S. Chan，“用于互联网入侵检测的深度对数神经网络”，Soft Computing，第25卷，第15期，第10 129–10 152页，2021年8月。[30] S. Ankalaki 和 M. N. Thippeswamy，“用于一维卷积神经网络的新型优化参数化双曲正切 Swish 激活函数：基于传感器的人类活动识别与异常检测应用”，Multimedia Tools and Applications，2023年。[31] A. Kind，“利用随机计算进行一般函数近似”，Electronics，第14卷，第9期，2025年。 [在线]。网址：https://www.mdpi.com/ 2079 9292 14 9 1845 - / / / [32] C. Ma 和 D. J. Lilja，“ 用于降低随机计算延迟的有限状态机并行实现 ”，载于第19届 ISQED，2018年，第335–340页。[33] W. Qian、X. Li、M. D. Riedel、K. Bazargan 和 D. J. Lilja，“采用随机逻辑的容错计算架构”，Computers, IEEE Trans. on，第60卷，第1期，第93–105页，2011年1月。[34] V.-T. Nguyen、T.-K. Luong、E. Popovici、Q.-K. Trinh、R. Zhang 和 Y. Nakashima，“基于随机逻辑的高精度紧凑型双曲正切和 sigmoid 计算”，载于IEEE MWSCAS，2021年，第386–390页。[35] O. Chen、R. Zhang、W. Luo、Y. Wang 和 N. Yoshikawa，“采用随机超导器件的超高能效非线性函数近似框架”，IEEE Transactions on Emerging Topics in Computing，第12卷，第4期，第956–967页，2024年。[36] N. Temenos 和 P. P. Sotiriadis，“用于高效神经网络设计的随机计算西格玛-德尔塔加法器架构”，IEEE J. on Emerging and Sel. Topics in Circ. and Sys.，第13卷，第1期，第285–294页，2023年。[37] J. Wang、H. Chen、D. Wang、K. Mei、S. Zhang 和 X. Fan，“用于节能型深度神经网络启发式精度提升的噪声驱动异构随机计算乘法器”，IEEE TCAD，第42卷，第2期，第630–643页，2023年。[38] J. G. van der Corput，“分布函数 I”，Proc. Akad. Wet. Amsterdam，第38卷，第813–821页，1935年。[39] F. Pausinger 和 A. Topuzo˘glu，“关于两类置换范德科普序列的差异度”，Uniform Distribution Theory，第13卷，第1期，第47–64页，2018年。[40] V. Todorov、T. Ostromsky、I. Dimov 和 S. Fidanova，“基于低差异序列、用于空气污染建模敏感性分析的优化准蒙特卡洛方法”，2020年9月，第25–28页。[41] T.-H. Chen 和 J. P. Hayes，“随机计算除法电路设计”，载于 ISVLSI，2016年，第116–121页。[42] H. Hsiao、J. S. Miguel、Y. Hara-Azumi 和 J. Anderson，“零相关误差：随机计算中有限长度比特流独立性的度量”，载于第26届亚洲及南太平洋设计自动化会议，系列：ASPDAC ’21，2021年，第260–265页。[43] M. R. Pedersen、L. Nalpantidis、R. S. Andersen、C. Schou、S. Bøgh、V. Kr¨uger 和 O. Madsen，“制造业机器人技能：从概念到工业部署”，Robotics and Computer-Integrated Manufacturing，第37卷，第282–291页，2016年。[44] I. Szentandr´asi、A. Herout 和 M. Dubsk´a，“高分辨率图像中二维码的快速检测与识别”，载于计算机图形学会议，系列：SCCG’12，2012年。[45] S. Asadi、M. H. Najafi 和 M. Imani，“面向低差异随机计算的低成本基于有限状态机的位流生成器”，载于2021年欧洲设计、自动化与测试会议暨展览（DATE），2021年，第908–913页。[46] P. K. Meher、J. Valls、T.-B. Juang、K. Sridharan 和 K. Maharatna，“坐标旋转数字计算50年：算法、架构与应用”，IEEE Transactions on Circuits and Systems I: Regular Papers，第56卷，第9期，第1893–1907页，2009年。[47] D.-U. Lee、R. Cheung、W. Luk 和 J. Villasenor，“多项式近似与插值的硬件实现权衡”，IEEE Transactions on Computers，第57卷，第5期，第686–701页，2008年。



Mehran Moghadam (S’22)于2010年和2016年分别获得伊斯法罕大学（伊朗）的计算机工程学士学位和计算机系统结构 硕士学位。他在两个项目中均以优异成绩毕业。2022年，他开始在美国路易斯安那州拉法叶的路易斯安那大学拉法叶分校计算与信息学学院攻读计算机工程博士学位。2024年，他转入美国俄亥俄州克利夫兰的凯斯西储大学电气、计算机与系统工程系，继续攻读计算机工程博士学位。Mehran于2024年入围嵌入式系统周（ESWEEK）和国际计算机辅助设计会议（ICCAD）的ACM SIGBED学生研究竞赛（SRC）决赛，并获选为2024年设计自动化会议（DAC）青年学者。他的研究兴趣包括新兴的非常规计算范式，例如能效随机计算模型、实时高精度超维计算系统、位流处理、鲁棒的存内算术计算，以及面向边缘人工智能的低功耗片上/近传感器计算设计。



Sercan Aygun （S’09-M’22-SM’24）于2013年获土耳其埃斯基谢希尔奥斯曼加济大学电气与电子工程理学学士学位，并同时修读计算机工程双学位。他于2015年获伊斯坦布尔技术大学电子工程理学硕士学位，并于2016年获阿纳多卢大学计算机工程第二个理学硕士学位。Aygun博士于2022年获伊斯坦布尔技术大学电子工程博士学位。Aygun博士获得了ACM SIGBED学生研究竞赛（SRC）嵌入式系统周2022最佳科学研究奖、GLSVLSI’23最佳论文奖，以及GLSVLSI’24最佳海报奖。Aygun博士的博士研究获得土耳其电子制造商协会颁发的最佳科学应用博士奖，并在土耳其科学院评选的科学与工程博士论文奖中位列全国第一。他目前是美国路易斯安那州拉法叶市路易斯安那大学拉法叶分校计算与信息学院的助理教授。他从事微型机器学习和新兴计算研究，包括随机计算和超维计算。



M. Hassan Najafi （S’15-M’18-SM’23）于2011年、2014年和2018年分别获得伊朗伊斯法罕大学计算机工程学学士学位、伊朗德黑兰大学计算机体系结构学硕士学位，以及美国明尼苏达大学双城分校电气工程学博士学位。2018年至2024年，他担任美国路易斯安那州拉法叶的路易斯安那大学拉法叶分校计算与信息学院助理教授。目前，他是美国俄亥俄州克利夫兰凯斯西储大学电气、计算机与系统工程系副教授。他的研究兴趣包括随机计算与近似计算、一元处理、存内计算和超维计算。他已发表或合著了80多篇经过同行评审的论文，并获得了6项美国专利，另有更多专利正在申请中。Najafi博士于2024年获得NSF CAREER奖，并曾获得GLSVLSI’23和ICCD’17最佳论文奖、GLSVLSI’24最佳海报奖、2018年EDAA杰出学位论文奖，以及明尼苏达大学博士学位论文奖学金。Najafi博士曾担任IEEE Journal on Emerging and Selected Topics in Circuits and Systems编辑，以及多个EDA会议的技术程序委员会委员。