arXiv:2601.14148v1 [cs.AR] 20 Jan 2026

可靠人工智能加速器的探索：跨层评估与设计优化

李萌[123*]，谢通[21],张作栋[4],和王润生[234*1]北京大学人工智能研究院 & [2]集成电路学院，中国北京[3]北京集成电路先进创新中心，中国北京[4]北京大学电子设计自动化研究院，中国无锡

摘要——随着 CMOS 技术迈向纳米尺度，老化效应和工艺波动日益显著，对人工智能加速器构成了严峻的可靠性挑战。传统的基于保护带的设计方法依赖悲观的时序裕量，牺牲了大量性能和计算效率，因此无法满足高性能人工智能计算的需求。当前，面向可靠性的人工智能加速器设计面临两项核心挑战：（1）缺乏系统性的跨层分析工具，无法捕捉器件、电路、架构和应用层之间的耦合可靠性效应；（2）传统可靠性优化与计算效率之间存在根本性的权衡。为应对这些挑战，本文系统地介绍了一系列面向可靠性的加速器设计，包括：（1）老化与波动感知动态时序分析器；（2）采用关键输入模式缩减的加速器数据流优化；（3）针对大语言模型（LLM）的韧性表征与新型架构设计。通过紧密结合跨层可靠性建模与人工智能工作负载特征，这些协同优化方法有效实现了可靠且高效的人工智能加速。

I. 引言

近年来，人工智能已成为一项颠覆性技术，并革新了许多不同的应用领域，包括自动驾驶等对可靠性要求极高的任务。人工智能工作负载，尤其是大语言模型（LLM），计算需求极高，已广泛部署在类 TPU 脉动阵列等定制加速器上。然而，随着摩尔定律推动技术迈向纳米尺度，这些加速器容易受到硬件故障的影响。这些故障包括时序错误，其成因包括工艺、电压、温度和老化（PVTA）变化等各种可靠性问题 [1]–[4]。因此，启用这两者至关重要可靠且高效的人工智能加速。

然而，确保人工智能加速器的可靠性面临重大挑战。首先，可靠性问题具有很强的跨层耦合特性，且缺乏准确的建模方法。例如，在器件级，电压应力会引起阈值电压漂移（ΔV_{th}）；在电路级，这些漂移会表现为特定的时序错误模式和时序错误率（TER）；在应用级，这些错误会降低人工智能模型精度（Acc）。这些关系极其复杂，难以进行精确建模。其次，现有的缓解方法无法兼顾可靠性与效率。采用最坏情况电压裕量可以消除时序错误，但会带来很大的开销；而缩小裕量则会提高比特错误率（BER），降低模型性能。Razor 触发器 [5]和基于算法的容错（ABFT） [6]等经典技术要么无法扩展到大型加速器，要么会导致频繁且不必要的恢复，因此不适用于计算密集型人工智能工作负载。为应对这些挑战，本文总结了我们近期在跨层评估和设计优化技术方面的工作 [7]–[9], 如图1所示。

本研究部分得到了国家自然科学基金委员会（62495102、92464104、62125401）、国家重点研发计划（2024YFB4505004）、北京市科技计划（Z241100004224015）、北京市杰出青年科学家计划（JWZQ20240101004）和111计划（B18001）的资助。*通讯作者：{meng.li,r.wang}@pku.edu.cn

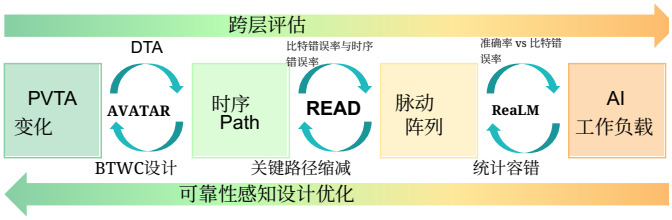


图 1. 我们近期工作的概述。

II. 老化与变化感知动态时序分析

A. 动机

随着技术缩放加剧时序保护裕量，优于最坏情况（BTWC）设计旨在利用动态时序裕量提升效率。传统动态时序分析（DTA）工具会加入最坏情况下的老化与变化保护带，从而扩大裕量并削弱 BTWC 带来的收益。AVATAR [7] 通过将晶体管老化和工艺变化直接纳入 DTA 来解决这一问题，消除不必要的保护带，同时保持性能与效率。*B. AVATAR*算法

AVATAR 的核心思想是将老化分析整合到 DTA 流程中，如图 2所示。该算法包含三个主要步骤：(1) 门级老化与变化模型表征，(2) 工作负载分析，以估算ΔV_{th} 每个晶体管的情况，以及 (3) 基于事件的DTA。

在步骤 1 中，老化模型以时序弧、输入转换时间、输出负载、每个晶体管的 ΔV_{th}、 V_{DD}以及温度作为输入，并使用一阶泰勒展开输出老化单元延迟和转换时间。变化模型基于采用 LVF数据的 POCV分析。在步骤 2 中，执行零延迟门级仿真，以获得每个内部网络的翻转率，随后应用老化模型

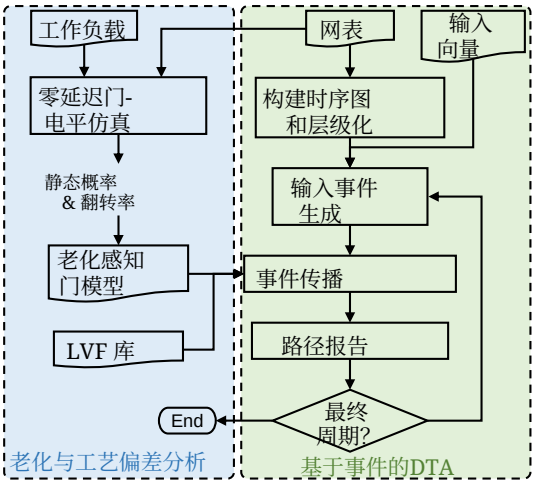


图2。AVATA 的老化与变化感知的 DTA 流程 R.

表I 基于基于工艺角的DTA和AVATAR的基于应用的DVFS带来的性能提升

基准测试	基于工艺角的DTA [10], [11]		AVATAR（本研究）	
	最大频率 (MHz)	提升 (相比 STA)	最大频率 (MHz)	提升 (相比 STA)
SHA	948	13.75%	1020	22.38%
AES CBC	883	5.99%	951	14.10%
FIR	915	9.82%	986	18.35%
冒泡排序	1290	55.38%	1380	65.36%
运动检测	958	15.00%	1030	23.97%
CNN	868	4.18%	936	12.30%
卷积	868	4.19%	936	12.28%
二维滤波器	936	12.33%	1050	26.37%
矩阵乘法	916	9.89%	989	18.63%
DCT	1170	40.77%	1270	52.15%

计算每个晶体管的 ΔV_{th} 。在步骤 3 中，事件被定义为引脚上的数字开关活动。基于事件的 DTA 首先构建用于事件传播的时序图，随后进行逐周期分析，包括输入事件生成、事件传播和路径报告，并记录所有时序端点的到达时间。

C. 实验

为了评估 AVATAR 的优势，我们通过两种方法确定特定应用的 $V_{\min}/f_{\max}$。 (1) 基于角落的 DTA，采用额外的保护带 [10], [11]。动态延迟计算如下 $delay \times (1+total\ \ guardband)$，其中假设老化保护带为 15%，标称条件下的随机变化保护带为 5% V_{DD}。保护带随V_{DD} 变化的趋势以 FO4 延迟作为代表性单元进行表征。(2) 基于 AVATAR 的分析。AVATAR 本身同时考虑老化和变化，无需额外的保护带。最终延迟计算如下 $\mu(delay)+3\sigma(delay)$。表 I 展示了在标称 0.8V 下应用最大频率所带来的性能提升。

III. 关键输入模式缩减

A. 计算序列和时序误差

READ [8] 提出了一种可靠性增强的加速器数据流优化技术，可有效减少

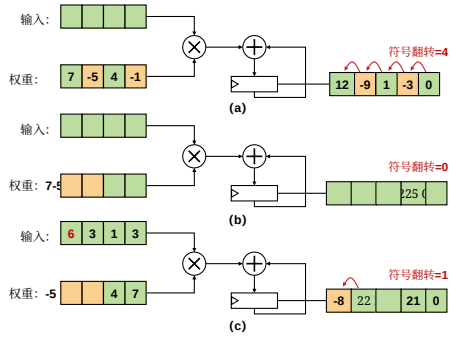


图 3. 以不同顺序计算 1×4 卷积。重排权重不会改变计算结果，但可以避免关键输入模式。

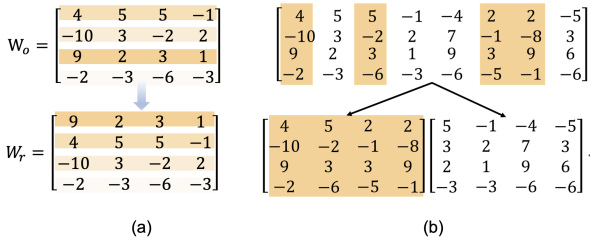


图 4. (a) 输入通道重排序。(b) 输出通道聚类。

时序错误。对于给定的周期，时序错误主要由两个因素决定：(1) 输入模式，它决定了激活路径；(2) 工作条件（例如温度、电压和老化），这些因素会影响这些路径的延迟。虽然工作条件通常无法控制，但通常只有少数路径（即关键路径）的延迟会超过时钟周期。因此，降低这些关键路径的激活率是降低时序错误率的关键。

本研究中的基本处理单元（PE）是一个乘累加器（MAC）单元。以 TPU MAC 的 8 位乘法器和 24 位累加器为例，对综合单元进行动态时序分析后发现，最常见的关键输入模式是使部分和（PSUM）的符号位翻转的模式。例如，计算$3 \times(-2) + 2 = -4$ 会产生二进制补码编码 111111111111111111111100，其中符号位翻转会在累加器中触发较长的进位链，从而激活关键路径。当 PSUM 从负数变为正数时，也会产生类似的效应。因此，降低 PSUM 符号翻转率是减少时序错误率的有效策略。

B. 输入通道重排序与输出通道聚类

鉴于 ReLU函数的输出大多为非负值，我们的启发式方法优先处理非负权重计算，如图4(a)所示。该方法通过按通道重排序算法实现，该算法根据同一通道内正权重所占比例对运算进行排序。

为在列数A_c 较大时提升效果，我们采用先聚类后重排序策略：先对输出通道进行聚类，再对每个簇内的输入通道进行重排序。如图4 (b) 所示，我们不直接对 W进行分段，而是先将输出通道聚类成组，再将 W 分成两个子矩阵。聚类-

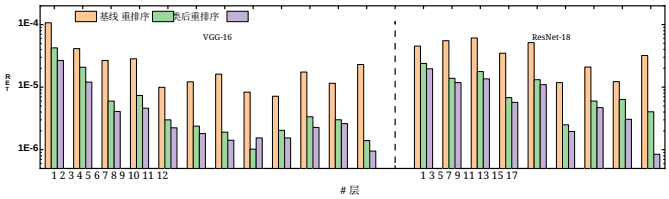


图 5. ResNet-18 和 VGG-16 上不同可靠性增强算法的时序错误率比较。

将符号模式相似的通道聚为一组，有助于在每个子矩阵内进行输入通道重排序。

形式化地，我们将两个n维向量**的符号差异（SD）**$\mathbf{x}$和$\mathbf{y}$ 定义为其符号向量之间的曼哈顿距离。在每个子矩阵内最小化 SD，可简化输入通道重排序并减少符号位翻转。因此，输出通道聚类问题可表述为最小化所有簇上的 SD(W_{T_i})；我们采用基于曼哈顿距离的权重符号矩阵平衡 KNN 来解决该问题。

C. 实验

如图5所示，直接重排序和先聚类后重排序算法的平均 TER 分别降低了 4.9×和 7.8×。先聚类后重排序方法在大多数层中都能持续实现更大幅度的降低，并且在输出通道数量较多的网络后续层中表现更优。

IV. 基于统计算法的容错

A. 大语言模型韧性表征

ReaLM [9] 刻画了大语言模型的错误韧性，并提出统计容错方法，以自适应地纠正关键错误。本节旨在回答六个问题。**Q1.1**：大语言模型不同层的韧性有何差异？**Q1.2**：大语言模型的位级韧性如何？**Q1.3**：在预填充阶段，大语言模型内部不同组件的韧性有何差异？**Q1.4**：故障幅度和频率对大语言模型性能的影响有何关联？**Q2.1**：大语言模型在预填充阶段和解码阶段的韧性有何差异？**Q2.2**：在解码阶段，大语言模型内部不同计算组件的韧性有何差异？

图 6 展示了针对这些问题的韧性表征。对于**Q1.1**，尽管各层表现出相近的韧性行为，但较早的层更容易受到错误影响。对于**Q1.2**，我们得出结论：低位错误对模型性能的影响可以忽略不计，而高位错误的影响会因重新量化而达到饱和。

对于**Q1.3**，我们发现网络组件后接归一化操作时，其韧性明显更差。另一方面，像 QKV 这样的组件具有更强的韧性。因此，我们可以将这些组件分为两类：敏感分量，例如 O 和 Down 投影；以及韧性组件，例如 QKV。

Q1.4 关注的是错误幅度和频率。如图 6 (g)(h) 所示，我们发现，在总和给定的情况下随着错误频率增加、错误幅度减小，韧性组件会表现出非单调的韧性行为。韧性组件既能容忍偶发的大错误，也能容忍频繁的小错误。但对于敏感分量，即使少数几个大错误也会严重降低性能。这一发现与我们之前的观察一致。

对于 **Q2.1** 和 **Q2.2**，图 6 （i）和（j）表明，预填充阶段对错误的敏感性高于解码阶段。此外，图 6 （k）和（l）进一步证实，韧性组件和敏感分量在两个阶段中保持一致。

B. 基于统计算法的容错

我们的统计型错误检测策略旨在仅纠正关键错误。图 7 重新阐释了错误幅度与频率之间的权衡关系。对于特定任务，我们首先定义一个可接受的性能下降阈值。该阈值界定了一个关键区域，在该区域内，计算错误会显著影响大语言模型性能。只有当观测到的错误落入关键区域时，才会触发重新计算过程，从而避免不必要的错误恢复过程。

图8 展示了我们的统计型 ABFT 电路设计。我们首先以权重驻留数据流为例。在右侧增加一列处理单元，用于存储计算所需的权重校验和 $e^{\mathrm{T}}WX$。与此同时，在底部增加一行加法器，用于累加输出校验和。此外，我们引入了一个统计单元，用于捕获错误统计数据，并据此确定是否需要恢复。一旦观测到的错误落入关键区域，就会触发重新计算过程。我们同样支持输出驻留数据流。在左侧增加一列加法器以计算权重校验和，并在底部增加一行处理单元以计算输出校验和。

C. 实验

在电路开销对比方面，与未防护的脉动阵列相比，我们的设计仅引入约 1.4% 的面积开销和 1.8% 的功耗开销，具有轻量化和实用性，适合实际部署。随后，我们在图 9中评估大语言模型性能和能耗节省。我们以 K在`OPT-1.3B`和 WikiText-2 上的表现，以及 V 投影在`LLaMA-3-8B`和 HellaSwag 上的表现为例。对于`OPT-1.3B`，我们的设计在 0.72 V 下达到最佳点，与现有技术方法相比节省 23% 的能耗。对于 `LLaMA-3-8B`，它在 0.70 V 下达到最佳点，并节省 24% 的能耗。

V. 结论

人工智能加速器的可靠性问题源于器件、电路、架构和应用层之间的跨层耦合。准确的可靠性建模可以显著减少过于保守的设计裕量，而跨层协同优化能够有效缓解可靠性约束。许多硬件故障可能被屏蔽，而且人工智能应用天生具有容错能力，因此无需针对最坏情况条件进行设计。允许有限的时序违例，由此为显著改善功耗、性能和面积创造了机会。

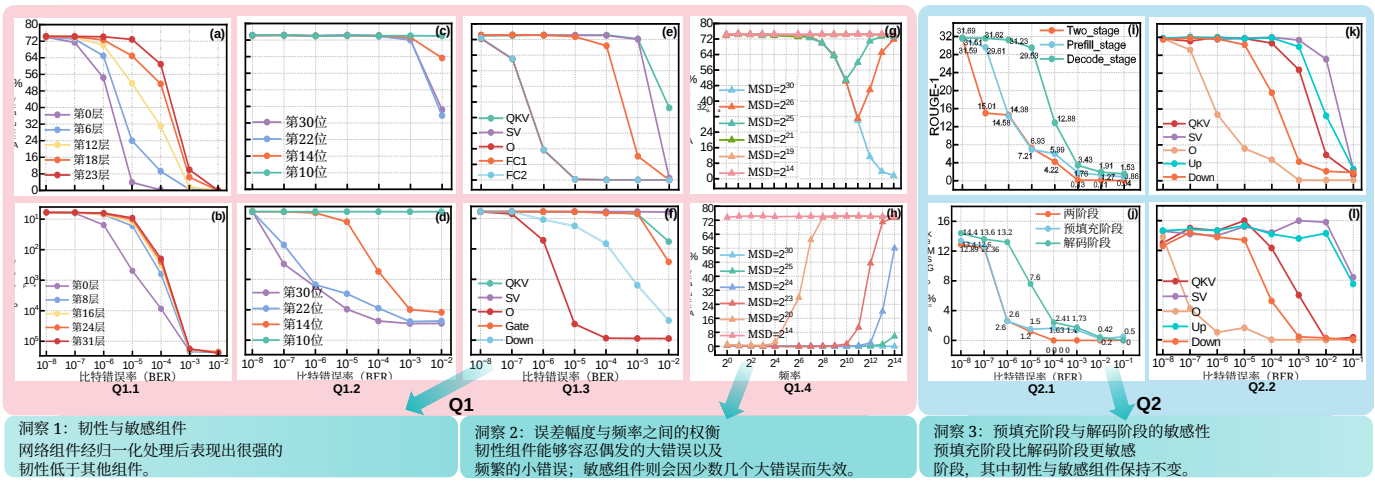
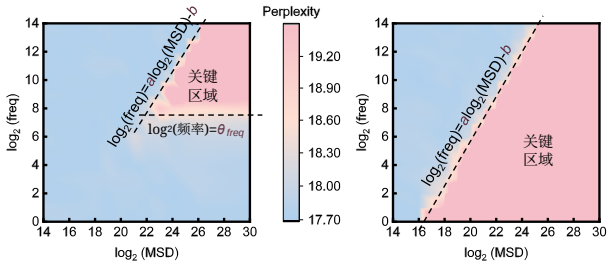


图 6. **Q1.1**：（a）（b）不同大语言模型在不同任务上的逐层容错能力。**Q1.2**：逐位错误容错能力。（c）对 K 进行错误注入。（d）对 O 进行错误注入。**Q1.3**：（e）（f）不同大语言模型组件的错误敏感性。**Q1.4**：错误频率与错误幅度之间的关系。（g）K 等韧性组件。（h）O 等敏感分量。给定 MSD，错误幅度随错误频率增加而降低。**Q2.1**：（i）（j）预填充阶段与解码阶段的比较。**Q2.2**：（k）（l）错误注入对不同网络组件的影响：O 和 Down 仍然高度敏感。（a）（c）（e）（g）（h）使用 OPT-1.3B 在 LAMDABA 上进行评估；（b）（d）（f）使用 LLaMA-2-7B 在 WikiText-2 上进行评估；（i）（k）使用 LLaMA-2-7B 在 X-Sum 上进行评估；（j）（l）使用 LLaMA-2-7B 在 GSM8K 上进行评估。



(a) 韧性组件　　(b) 敏感分量

图 7. 我们的统计型算法基础容错策略仅纠正落在关键区域内的错误。

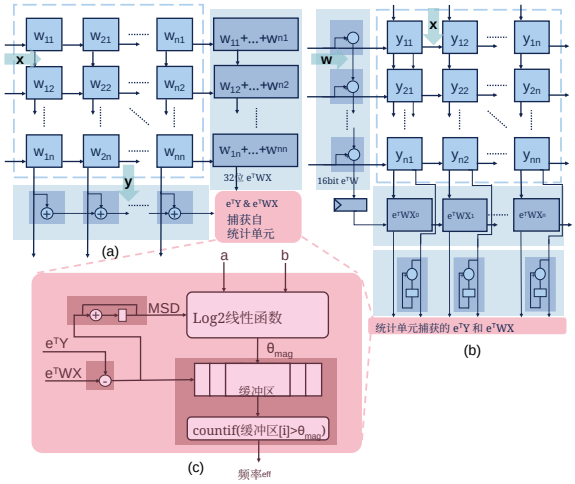


图 8. 脉动阵列上的统计型算法基础容错架构设计：(a) 权重驻留数据流的算法基础容错实现；(b) 输出驻留数据流的算法基础容错实现；(c) 定制统计单元。

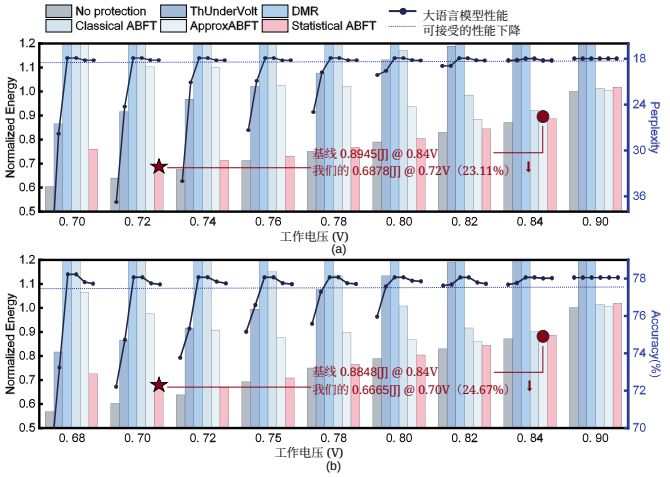


图 9. 大语言模型性能与总能耗节省对比。(a) WikiText-2 上的 OPT-1.3B。(b) HellaSwag 上的 LLaMA-3-8B。我们的方法以极低的保护开销实现了有竞争力的性能，与现有方法相比显著降低了成本，并分别在 0.72V 和 0.70 V 下实现了最大的能耗节省。虚线表示可接受的性能下降。

参考文献

[1] R. Huang 等，“面向16/14 nm及更先进技术的变异性与可靠性感知设计”，载于*IEDM*会议论文集，第12–4页，IEEE，2017年。[2] R. Wang 等，“新兴计算范式能否帮助提升技术路线图末期的可靠性？”，载于*IRPS*会议论文集，第1–7页，IEEE，2021年。[3] R. Wang 等，“先进技术节点的可靠性跨层设计：电子设计自动化视角”，载于*ICSICT*会议论文集，第1–4页，IEEE，2022年。

[4] Z. Sun 等，“先进 CMOS 技术中面向可靠性的设计挑战：从单模式到混合模式机制”，载于 *ICSICT* 会议录，第 1–1 页，IEEE，2024。[5] D. Ernst 等，“Razor：一种基于电路级时序推测的低功耗流水线”，载于 *MICRO* 会议录，第 7–18 页，IEEE，2003。[6] K.-H. Huang 等，“矩阵运算的基于算法的容错”，*IEEE Transactions on Computers*，第 518–528 页，1984。[7] Z. Zhang 等，“Avatar：面向应用的 DVAFS 老化与变异性感知动态时序分析器”，载于 *DAC* 会议录，第 841–846 页，2022。[8] Z. Zhang 等，“READ：利用关键输入模式缩减实现可靠性增强的加速器数据流优化”，载于 *ICCAD* 会议录，第 1–9 页，IEEE，2023。[9] T. Xie 等，“ReaLM：采用基于统计算法的容错实现可靠且高效的大语言模型推理”，载于 *DAC* 会议录，第 703–709 页，2025。[10] H. Cherupalli 等，“利用动态时序裕量提升超低功耗嵌入式系统的能效”，载于 *ISCA* 会议录，2016。[11] J. Constantin 等，“利用微处理器中的动态时序裕量，通过基于指令的时钟调节实现频率过度缩放”，载于 *DATE* 会议录，第 381–386 页，2015。