查看所有版本

Versal 自适应 SoC 系统和解决方案规划方法指南

UG1504 (v2023.2) 2023 年 11 月 15 日

目录

第 1 章

简介

关于 Versal 自适应 SoC 设计方法论

AMD Versal™ 自适应 SoC 设计方法论是一整套旨在帮助简化当今 Versal 器件设计进程的最佳实践。鉴于这些设计的规模与复杂性，因此必须通过执行特定步骤与设计任务才能确保设计每个阶段都能成功完成。建议您遵循这些步骤和最佳实践进行操作，这将有助于您以尽可能最快且最高效的方式实现期望的设计目标。

按设计进程浏览内容

AMD 自适应计算文档按一组标准设计进程进行组织，以便帮助您查找当前开发任务相关的内容。您可以在设计中心页面上访问 AMD Versal™ 自适应 SoC 设计流程。您还可以使用设计流程助手来更深入了解设计流程，并找到特定于预期设计需求的内容。

- 系统和解决方案规划：确认系统级别的组件、性能、I/O 和数据传输要求。包括解决方案到 PS、PL 和 AI 引擎的应用映射。

  注释：如需了解更多信息，请参阅《Versal 自适应 SoC 设计指南》(UG1273)。

如需获取更多方法论相关信息，请参阅以下文档：

- 嵌入式软件开发：基于硬件平台来创建软件平台，并使用嵌入式 CPU 开发应用代码。还涵盖 XRT 和计算图 API。请访问此链接以参阅《AI 引擎工具和流程用户指南》(UG1076) 中的相应内容。
- AI 引擎开发：创建 AI 引擎计算图及内核、库用法、仿真调试与剖析以及算法开发。还包含 PL 与 AI 引擎内核的集成。请参阅《AI 引擎工具和流程用户指南》(UG1076) 和《AI 引擎内核与计算图编程指南》(UG1079)。
- 硬件、IP 和平台开发：为硬件平台创建 PL IP 块、创建 PL 内核、功能仿真以及评估 AMD Vivado™ 时序收敛、资源使用情况和功耗收敛。还涉及为系统集成开发硬件平台。请参阅《Versal 自适应 SoC 硬件、IP 和平台开发方法指南》(UG1387)。
- 系统集成与确认：集成和确认系统功能性能，包括时序收敛、资源使用情况和功耗收敛。请参阅《Versal 自适应 SoC 系统集成和确认方法指南》(UG1388)。
- 开发板系统设计：通过板级原理图和开发板布局来设计 PCB。还包含功耗、散热以及信号完整性注意事项。请参阅《Versal 自适应 SoC 开发板系统设计方法指南》(UG1506)。

关于本指南

本指南包含对应如下主题的高层次信息、设计指南和设计决策利弊取舍：

- 第 2 章：系统设计类型：提供了每种 Versal 器件系列所支持的不同系统设计类型和设计流程的概述。
- 第 3 章：系统设计规划方法论流程：描述了进行系统设计规划时的方法论建议，包括基于系统设计类型的特殊注意事项。
- 第 4 章：功耗和散热规划：涵盖了有关功耗估算、散热设计、供电和去耦的系统设计注意事项。
- 第 5 章：系统调试规划：提供了有关 Versal 自适应 SoC 调试接口的信息以及有关调试 AI 引擎和 PS 的高层次信息。
- 第 6 章：系统验证规划：提供了有关 Versal 自适应 SoC 的高层次仿真建议。
- 第 7 章：系统确认规划：提供了有关基于系统设计类型制定系统确认规划时的主要聚焦领域的信息。

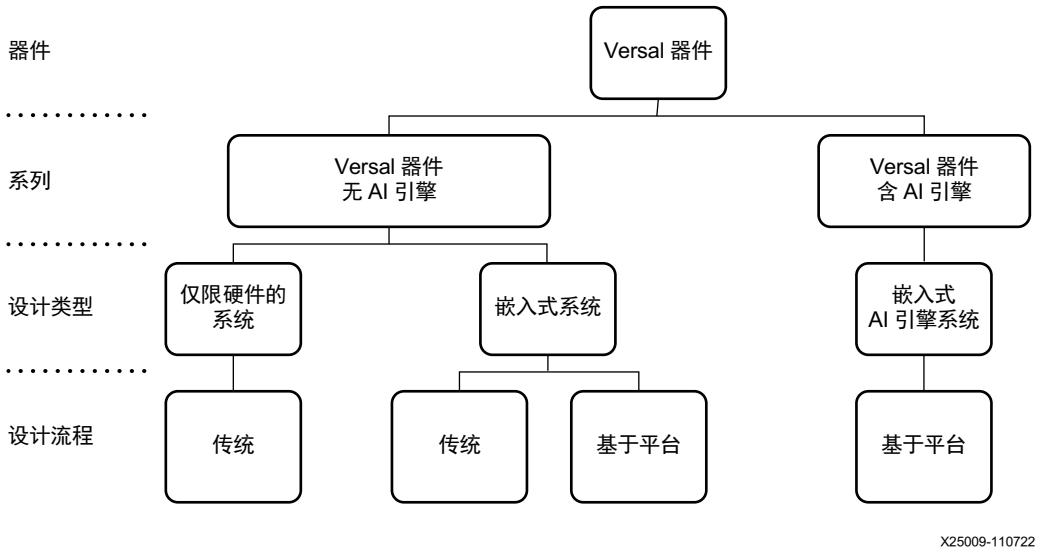
第 2 章

系统设计类型

AMD Versal™ 自适应 SoC 属于异构计算平台，具有多个计算引擎。在 Versal 自适应 SoC 上可映射各种应用，包括对无线系统、机器学习推断和视频处理算法进行信号处理。除了多个计算引擎外，Versal 自适应 SoC 还可使用高速串行 I/O、片上网络 (NoC)、DDR4/LPDDR4 存储器控制器、HBM 控制器和多重速率以太网媒体访问控制器 (MRMAC) 来提供超高系统带宽。Versal 器件分类为以下几个系列：Versal Prime、Premium、HBM、AI Core 和 AI Edge。下图显示了每种 Versal 器件系列所支持的不同系统设计类型和设计流程。

注释：Versal Prime 系列、Premium 系列和 HBM 系列的设计流程与 AMD FPGA 所使用的流程类似。Versal AI Core 系列、AI Edge 系列以及 Versal Premium VP2502 和 VP2802 器件的设计流程要求您面向异构计算平台进行设计，此平台具有特殊的硬件配置和软件支持要求。

图 1：**系统设计类型**



下表显示了每种 Versal 器件系列所支持的系统设计类型和设计流程。如该表中所示，大部分设计流程都以构建平台为基础。

表 1：**系统设计类型**

设计类型	器件系列	设计流程	平台源文件	GitHub 示例
仅限硬件的系统	Versal Prime 系列 Versal Premium 系列 Versal HBM 系列	传统	不适用	Versal 器件架构教程
嵌入式系统	Versal Prime 系列 Versal Premium 系列 Versal HBM 系列	传统	不适用	Versal 自适应 SoC 嵌入式设计教程
		基于平台	定制	Versal Prime 系列 VMK180 目标参考设计

表 1：**系统设计类型** (续)

设计类型	器件系列	设计流程	平台源文件	GitHub 示例
嵌入式 AI 引擎系统	Versal AI Core 系列 Versal AI Edge 系列 Versal Premium VP2502 器件和 VP2802 器件	基于平台	定制	AI 引擎开发设计教程 VCK190 基本 TRD AI 引擎机器学习教程

提示：请访问 GitHub 以获取更多示例，这些示例会定期更新。

以下提供了每种系统设计类型的汇总信息：

- 仅限硬件的系统：可编程逻辑设计。使用传统设计流程创建此系统。
- 嵌入式系统：嵌入式处理器系统，软件在 Arm® Cortex®-A72 或 Cortex-R5F 处理器上运行，硬件内容则位于 PL 内。使用传统设计流程或基于平台的设计流程创建此系统。
- 嵌入式 AI 引擎系统：嵌入式处理器系统，软件在 Arm Cortex-A72 或 Cortex-R5F 处理器上运行，硬件内容位于 PL 内，算法内容则位于 AI 引擎内。使用基于平台的设计流程创建此系统。

Versal 自适应 SoC 的设计流程如下所示：

- 传统设计流程：在传统设计流程中，系统的整个 PL 部分都是在单个 AMD Vivado™ 工程中定义的。该工程必须包括 Versal 基础硬件 IP 块（例如，Control, Interface, and Processing System (CIPS)、NoC、I/O 控制器）以及工程所需的任何其他定制 RTL 和 IP 块。设计源文件将添加到 Vivado 工具中，并通过 Vivado 实现流程进行编译。如果系统仅包含 PL 组件，那么可使用 Vivado 工具来生成可编程器件镜像 (PDI)，以便对 Versal 器件进行编程。如果系统还包含嵌入式软件内容，那么将在从 Vivado 工具导出的固定硬件设计上的 AMD Vitis™ 环境中开发软件应用。此流程类似于用于 AMD Zynq™ UltraScale+™ MPSoC 的传统流程。
- 基于平台的设计流程：在基于平台的设计流程中，硬件系统分为下列不同元素：可复用的基础平台以及基本硬件扩展，此平台是在 Vivado 中开发的，而扩展则是在 Vitis 中通过基础平台的可扩展区域内精确定义的一组连接接口来开发的。大部分硬件设计是在 Vivado 中开发的，但设计中以 C++ 而非硬件描述语言 (HDL) 指定的部分大多是在 Vitis 中自然开发并集成的。后者示例包括 AI 引擎计算图与内核以及以通过高层次综合 (HLS) 编译的 PL 作为目标的内核函数。

  您可根据自身工作效率来选择任一设计分区方式：基础平台或可扩展区域。在整个设计周期过程中，基本硬件和可扩展区域均可进化，精心设计的基础平台能为多种应用奠定基础，以便 Vitis 工具在其中对可扩展区域进行扩展。相应开发团队可通过合理的松散耦合与紧密耦合将设计内容从 Vivado 导出到 Vitis，反之亦然，这有助于促进组成异构系统的不同元素的并发开发和集成。

不含 AI 引擎的 Versal 器件的系统设计类型

不含 AI 引擎的 Versal 器件由以下部分组成：传统的可编程逻辑互连结构、存储器（块 RAM 和 UltraRAM）、片上网络 (NoC)、硬化的 DDR 存储器控制器、处理子系统以及多个硬化的 IP（例如，PL PCIe®、CPM、Multirate Ethernet MAC (MRMAC)、600G Channelized Multirate Ethernet Subsystem (DCMAC) 和 High-Speed Crypto Engine (HSC)）。

欲知详情，请参阅下列产品选型指南：

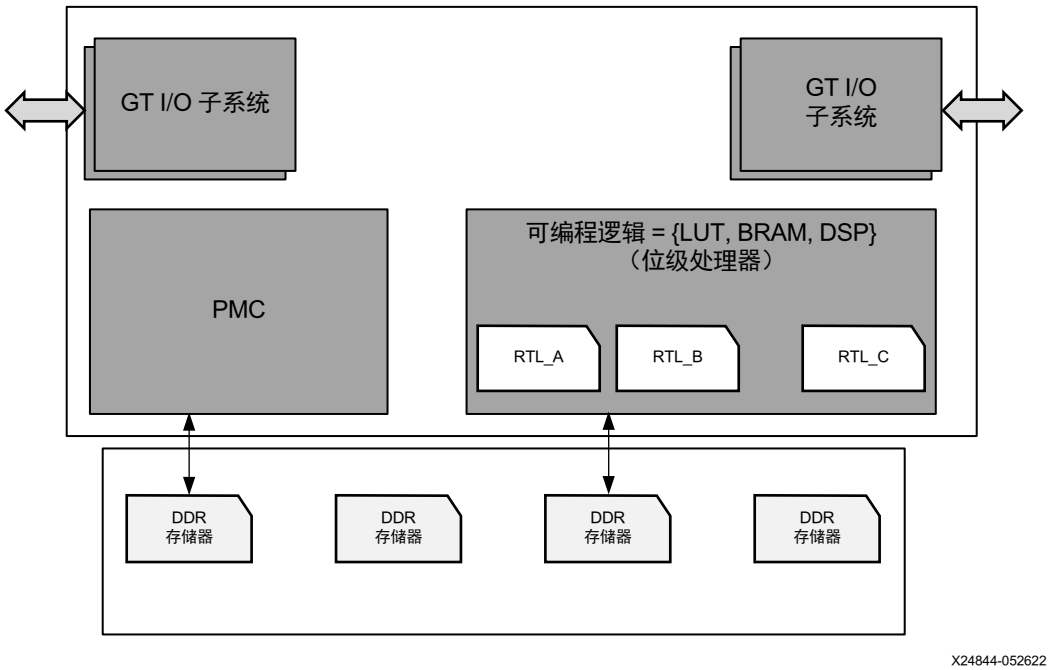
- 《Versal Prime 系列产品选型指南》(XMP453)
- 《Versal Premium 系列产品选型指南》(XMP463)

- 《Versal HBM 系列产品选型指南》(XMP465)

仅限硬件的系统

仅限硬件的系统由不含嵌入式软件栈或服务器软件栈（用于控制系统操作）的可编程逻辑实现组成。仅限硬件的系统采用自动运行，代表传统可编程逻辑实现。与 FPGA 设计相似，仅限硬件的系统通常由顶层 RTL 设计来驱动。器件编程由 Versal 自适应 SoC 平台管理控制器 (PMC) 负责，如下图所示。仅限硬件的系统设计示例包括联网线路卡、I/O 协议桥接器、仿真系统和无需嵌入式软件栈的仅限逻辑实现。在此系统中，整个器件中的 NoC 是高带宽互连。

图 2：**仅限硬件的系统**



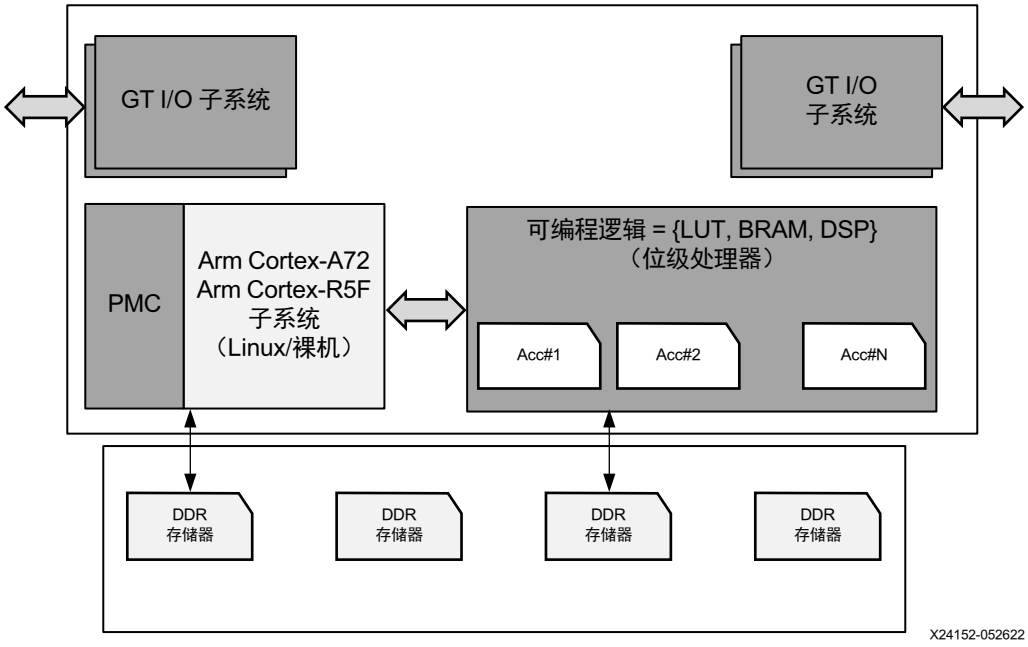
嵌入式系统

嵌入式系统由 Versal 器件中的嵌入式处理器与下列硬件组件中内置的加速逻辑组成：传统 PL（LUT、块 RAM、UltraRAM、DSP）、Versal 驱动程序、裸机/Linux BSP 和 PetaLinux/Yocto 工具。对于 Versal 器件，嵌入式计算系统包含 Arm Cortex-A72 处理器和 Cortex-R5F 处理器。对于此设计类型，使用模型覆盖范围广泛，从复杂的嵌入式软件栈到简单的裸机栈（仅用于支持硬件单元编程）都包含在内。

嵌入式系统设计可运行在内建嵌入式处理器上执行的软件栈，充当加速单元上运行的内核的整体控制层。嵌入式处理器与 Versal 器件之间的数据传输由 Xilinx Runtime (XRT) 应用编程接口 (API) 来管理。这些 API 还具有用于管理硬件单元的函数调用。如需了解更多信息，请参阅《XRT 文档》中的 XRT 和 Vitis 平台概述。

注释：XRT 是基于 Linux 系统软件之上的增值软件层，能提供特定的加速编程模型。但正如 Vivado，Vivado 基础之上的 Vitis 硬件工具同样能通过标准硬件交接和驱动程序来支持更低层次的传统嵌入式软件编程。

图 3：**嵌入式系统**



含 AI 引擎的 Versal 器件的系统设计类型

含 AI 引擎的 Versal 器件使用基于平台的设计流程来处理以下类型的系统设计。

欲知详情，请参阅下列产品选型指南：

- 《Versal AI Core 系列产品选型指南》(XMP452)
- 《Versal AI Edge 系列产品选型指南》(XMP464)
- 《Versal Prime 系列产品选型指南》(XMP453)

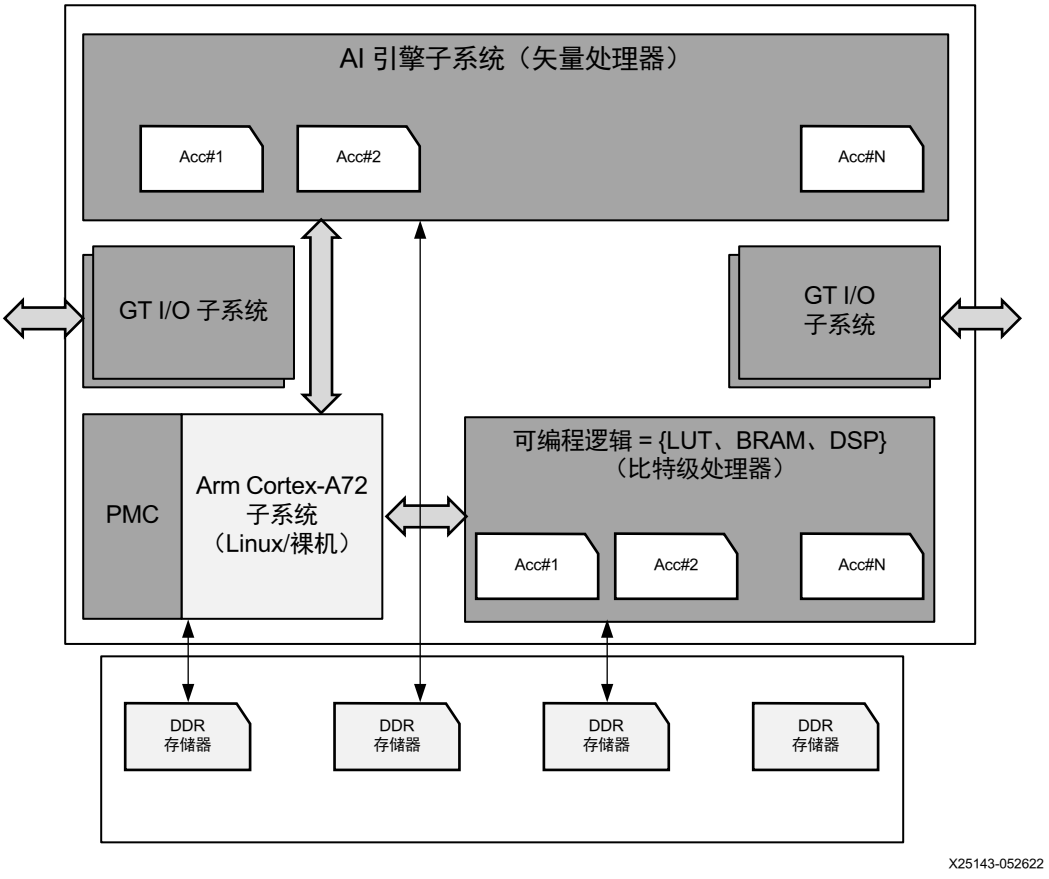
嵌入式 AI 引擎系统

嵌入式 AI 引擎系统包含 Versal 自适应 SoC 中的嵌入式处理器及加速逻辑，在以下 2 个主要的加速组件类别中均内置此加速逻辑：传统 PL（LUT、BRAM、URAM、DSP）和 AI 引擎。对于 Versal 自适应 SoC，嵌入式计算系统包含 Arm Cortex-A72 和 Cortex-R5F 处理器。对于此设计类型，使用模型覆盖范围广泛，从复杂的嵌入式软件栈到简单的裸机栈（仅用于支持加速单元编程）都包含在内。

嵌入式 AI 引擎系统设计可运行在内建嵌入式处理器上执行的软件栈，充当加速单元上运行的内核的整体控制层。嵌入式处理器与 Versal 自适应 SoC 之间的数据传输由 Xilinx Runtime (XRT) 应用编程接口 (API) 来管理。这些 API 还具有用于管理加速单元的函数调用。

虽然嵌入式 AI 引擎能完全通过硬件串流对接到 Versal 器件中的 PL，但其目标系统也包括利用嵌入式 Arm 处理子系统并使用 AI 引擎和 PL 的系统，如下图所示。

图 4：**嵌入式 AI 引擎系统**



第 3 章

系统设计规划方法论流程

系统设计规划方法论要求基于目标应用明确所有系统要求。其中包括识别具有正确特性（例如，DDR 存储器控制器 IP 数量、AI 引擎等）的相应 AMD Versal™ 器件。您还必须考量功耗和散热要求。选择相应的器件后，下一步即可着手系统设计，包括在器件上进行目标应用的软硬件协同设计、系统验证以及初始化和调试。

为确保充分利用 AMD Versal™ 自适应 SoC 中可用的多种多样的计算元件，并使用最高效的实现流程，AMD 建议遵循如下系统设计方法论流程进行操作：

- 应用映射：开发应用和算法到器件的映射。
- 性能建模和仿真：捕获 NoC 流量和数据流建模。运行系统仿真，从性能监控器收集统计数据并加以分析。

重要提示！预先分析您的安全保障要求（例如，隔离需求、安全启动等）。如需了解更多信息，请参阅《Versal 自适应 SoC 技术参考手册》(AM011)。如需了解有关安全性功能特性的详细信息，请参阅 AMD 网站上的设计安全性专区（需注册）中提供的《Versal 自适应 SoC 安全手册》(UG1508)。

应用映射和设计分区

应用映射是使用 Versal 器件执行系统设计规划的第一步。应用映射期间，会根据性能、时延和系统成本要求，将系统各部分映射到自适应 SoC 硬件中。

例如，嵌入式应用通常由以下几部分组成：执行计算密集型处理的加速器块、用于数据存储的 DRAM、用于将数据从 DRAM 移至加速器硬件的互连开关、用于加速中间访问的片上存储器和用于控制数据流并执行计算密集度较低的任务的嵌入式处理器。数据中心应用通常由以下几部分组成：基于 PCIe® 接口的接口（用于将数据从主机传输至卡 DRAM）、互连开关、用于处理计算密集型功能的加速器块、用于加速加速器访问的片上存储器，以及嵌入式处理器核（用于控制 Versal 器件中的数据移动、执行计算密集度较低的功能以及管理与主机处理器的通信）。

执行应用映射时，请考虑以下哪个 Versal 器件系列更适合您的系统应用：

- Versal Prime 系列、Premium 系列和 HBM 系列：Versal Prime 系列、Premium 系列和 HBM 系列包含数字信号处理器 (DSP) 引擎、可编程逻辑 (PL)、片上网络 (NoC)、PCIe 接口和处理子系统。
- Versal AI Core 系列、AI Edge 系列和 Premium 系列：Versal AI Core 系列、AI Edge 系列及 Versal Premium VP2502 和 VP2802 器件包含自适应 AI 引擎以及 DSP 引擎、PL、NoC、PCIe 接口和处理子系统。

例如，您可根据计算、功耗和时延要求将计算密集型加速器功能映射到 DSP 引擎或 AI 引擎。如果某个功能的计算量较大，需多重并行处理，则建议使用 AI 引擎。如果某个功能计算要求较低，但时延至关重要，则建议使用 DSP 引擎。

要识别适合您的设计的最佳计算映射，请参阅 GitHub 仓库中提供的教程：

- 使用 Vitis 加速库完成 Versal 2D-FFT 实现教程
- Versal AI 引擎/HLS FIR 滤波器教程

系统计算

DSP 引擎和 AI 引擎均可执行相似类型的计算操作。这两种引擎都是为了实现高效的乘积累加 (MAC) 函数（例如，FIR 滤波器）而设计的。进行设计分区时，重要的是了解两种块的不同功能及其与 PL 的交互方式。本节主要介绍块的计算功能以及不同数据类型映射到不同引擎的优劣比较结果。

Versal AI Core 系列、AI Edge 系列以及 Versal Premium VP2502 和 VP2802 中的器件包含 AI 引擎 tile 拼块阵列或 AI 引擎机器学习 (AIE-ML) tile 阵列。含 AIE-ML 阵列的器件可包含额外的 512 KB 存储器拼块行，专为机器学习推断加速等各种计算密集型应用经过最优化。

以下提供了部分函数的示例，这些函数可映射到 DSP 引擎，或者可通过矢量化来映射到 AI 引擎：

- 乘
- 乘积累加
- 快速傅里叶变换 (FFT)
- 有限脉冲响应 (FIR) 滤波器
- 矩阵相乘

DSP 和 PL 计算是基于样本来计算的。PL 擅长位操作函数和快速数据重排序，这些功能在为系统管理数据时尤为重要。

AI 引擎是 SIMD 矢量处理器，即易于矢量化的函数适合在 AI 引擎中实现。例如，线性代数函数本身就适合矢量化。

AI 引擎可执行基于样本和基于块的处理。对于基于样本的处理，AI 引擎随串流接口一起运行，且块与矢量处理和寄存器存储器对齐。这样即可实现低时延，尤其是可以支持高吞吐量设计以高采样率来运行（例如，超采样率处理）。

使用窗口接口时，AI 引擎已配置为基于块进行处理，即矢量化处理。根据窗口大小和采样率，时延和吞吐量可能会受到影响，如果窗口（块）大小较小，则尤其容易受到影响。如果同时使用 256 位存储器读取端口和 256 位写入端口，则此选项适用于高带宽。

由于 AI 引擎是矢量处理器，因此它在单一时钟周期内可执行的操作数量比 DSP 引擎更多，例如，在 INT8 内可执行 256 项操作，相比之下，DSP 引擎在单一时钟周期内只能执行 6 项操作。下表显示了 AI 引擎中的本机数据类型与使用 DSP 引擎时同等性能情况下的比较结果。在某些情况下，如果系统包含大量针对这些数据类型的线性代数计算，那么 AI 引擎可能是更合适的选择。但还有其他注意事项需要考量，详情参阅下文所述。

表 2：**每个 Versal 自适应 SoC 智能引擎的每个周期内的操作数**

数据类型	DSP 引擎	AI 引擎
INT8	6	256
INT16	2	64
INT24	2	16
INT32	不适用 [1]	16
FP32	2	16
Complex 16	2[2]	16
Complex 32	不适用 [1]	2

注释：

1. 无法在单一 DSP 引擎内实现，需要额外 PL 资源
2. 需要 2 个 DSP 引擎以实现最多 18 位复数乘法器或 MACC

在决定用于实现这些类型的函数的引擎之前，必须评估所需的计算量。如下表中的示例所示，小型 11 抽头 FIR 滤波器所需算力远小于大型 131 抽头 FIR。因此，您可选择使用 DSP 引擎和 PL 来实现 11 抽头 FIR。但 131 抽头 FIR 可能在 AI 引擎中更高效。

表 3：**不同 FIR 实现的计算要求示例**

11 抽头 FIR 滤波器	131 抽头 FIR 滤波器
16 位实数数据和 16 位实数系数 AI 引擎中 16 位的可用计算量 = 32 个 MAC	
所需计算量 = 11 个 MAC AI 引擎数 = 0.35 (1) 资源使用率 = 0.35 使用率低	所需计算量 = 131 个 MAC AI 引擎数 = 4.09 (5) 每个 tile 的资源使用率 = 0.82 使用率高

在进行系统分区时，还有其他因素可能影响您的决策。例如，在应用数据流中 11 抽头滤波器发生位置在哪里？它是否属于更大的滤波器链中的一部分？如果是这样，那么从架构角度而言，合理方法可能是在 AI 引擎中实现此小型 FIR 和滤波器链的其余部分，这样即可提升总体系统设计的效率。

对于 AI 引擎不提供原生支持的函数（例如，INT4 或 24 位复数），最好在 DSP58 和 PL 中实现这些函数，因为这样可以简化实现。或者，在 AI 引擎中也可能支持这些非原生运算符。如果您使用此方法（比如在 PL 内），则必须在矢量通道内谨慎管理这些数据，这需要额外的数据管理功能。

AI 引擎矢量处理器能够高效执行大量并发算术运算。如果应用需要此类运算，那么部分非线性代数函数也可以在 AI 引擎标量处理器中实现。此方法的优势在于它无需将数据移出 AI 引擎阵列再移入 PL 来执行这些函数。由于 PL 与 AI 引擎阵列之间存在的往返时延，在 PL 中对 DSP58 执行预处理和后处理尤为有效。

存储器和数据移动

了解 Versal 自适应 SoC 中的存储器层级的重要意义在于帮助您判定：

- 需解决的问题的范围。
- 不同引擎之间的数据通信方式以及可用带宽。
- 如何利用每个 AI 引擎中可用的原始计算来为应用提供最佳单位功耗性能（如适用）。

不同类型的应用根据其部署环境，有不同的存储器层级。例如，某些应用具有外部 DDR 存储器需求，其他应用使用诸如 JESD 等接口来从离散模数转换器 (ADC) 引入数据。无论数据源自何处，构建正确的存储器层级以满足系统需求都至关重要。

例如，在需要外部 DDR 存储器的系统中，只能以指定速率从 DDR 存储器获取数据。因此，最大带宽针对系统为固定值。例如，在 Versal 自适应 SoC 中，每个存储器控制器到 NoC 的最大 LPDDR 存储器带宽约为 34 GB/s。

来自 DDR 存储器的数据可通过 NoC 在整个自适应 SoC 内移动。对于 AI 引擎，可通过 AI 引擎阵列接口中的 NoC 接口拼块 (tile) 直接访问 AI 引擎阵列。但带宽相对较低，因此这可能并非将输入导入 AI 引擎拼块的最佳途径。

在大部分情况下，由于 Versal 器件中有大量片上存储器可用，因此数据应通过 NoC 导入 PL 中的多阶段存储器（例如，UltraRAM 和/或块 RAM）。然后，由于从 PL 进出 AI 引擎阵列的带宽要高得多，因此使用 PL 接口阵列拼块来将数据传入和传出 AI 引擎。

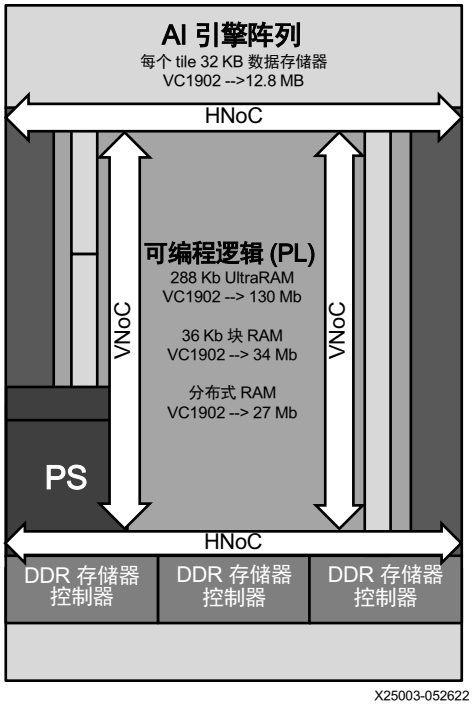
某些应用可能要求在 DDR 存储器到 NoC 再到 AI 引擎之间进行直接通信。此通信方式虽然可行，但可用总体带宽较低。因此，对于大部分应用，建议使用 NoC 通过 PS 或任何主控制器来对通信进行调试、追踪和控制。

包含 AI 引擎阵列的器件在每个 AI 引擎拼块内还包含本地数据存储器。每个拼块都有 8 个数据存储体，每个存储体均为 4 KB，总计每个拼块 32 KB。每个 AI 引擎核均可在本地直接访问相同 AI 引擎拼块上的数据存储器以及相邻拼块上的 3 个数据存储器（例如，北、南以及东或西侧）。这样每个拼块都有 128 KB 本地共享存储器。

含 AI 引擎机器学习阵列的器件可包含额外的行，其中含有 512 KB 存储器拼块，用于提供低时延的本地存储器存储空间。

下图显示了 VC1902 可用存储器总量。对于 Versal AI Core 器件，请参阅《Versal AI Core 系列产品选型指南》(XMP452)。

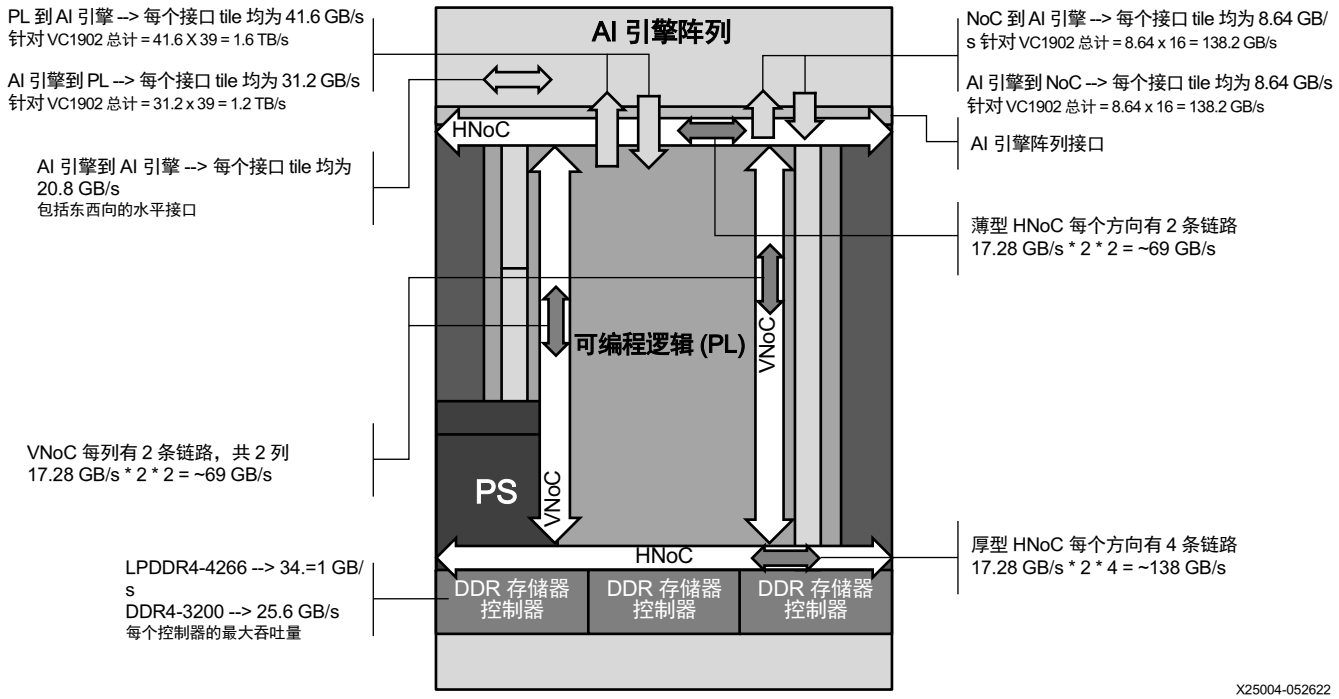
图 5：**Versal 自适应 SoC AI Core 系列中的片上存储器层级**



注释： AI 引擎阵列存储器总量为 32 KB 乘以器件上的 AI 引擎拼块数量。例如，VC1902 包含 400 个拼块。因此，AI 引擎阵列存储器总量（32 KB 乘以 400）等于 12.8 MB。

数据通信是实现高效设计的关键，使用 AI 引擎阵列时尤其如此。因此，必须了解往来 AI 引擎的数据带宽以及各 AI 引擎间内部数据带宽才能对设计进行高效分区。欲知详情，请访问此链接以参阅《Versal 自适应 SoC AI 引擎架构手册》(AM009) 中的相应内容。

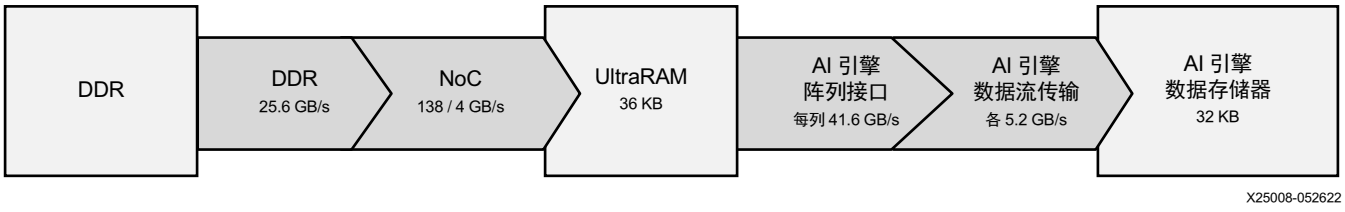
图 6：**通过 Versal 自适应 SoC 的通信带宽**



对称 FIR、卷积神经网络 (CNN) 或波束成形等功能之间有些数据复用（例如，系数和权重共享）。对于此类功能，您可以降低存储器带宽，并使用串流广播功能来向多个 AI 引擎拼块发送相同的权重或系数。具有大量数据复用的应用也适合在 AI 引擎中实现。在单一拼块上的大型滤波器实现中，窗口接口更适合数据复用。

下图显示了数据通过 Versal 自适应 SoC 进入 AI 引擎阵列的数据流示例，其中包含一些通过器件进行数据映射时需要考量的重要带宽数值。

图 7：**进入 AI 引擎的数据流示例**



使用此数据流示例时，根据您的应用详细信息，有些必须考量的注意事项。将数据从 DDR 存储器导入 PL 中的多阶段存储器（例如，UltraRAM）的方式取决于要传输的数据量、存储器控制器吞吐量、NoC 带宽以及存储器大小。

将数据载入 UltraRAM 后，PL 中可能需要数据排序或预处理阶段。假定无需这些阶段，那么数据需进入 AI 引擎以供处理。此数据流阶段内主要需要关注的是通过 AI 引擎阵列接口的带宽以及 AI 引擎阵列内用于将数据传输至必要拼块的带宽。

在 AI 引擎中要解决的问题范围取决于 AXI4-Stream 带宽和 AI 引擎拼块与阵列中的数据存储器大小。

吞吐量和时延

大多数应用都期望能实现高吞吐量和低时延。Versal 自适应 SoC 可帮助您实现这一目标，有了 AI 引擎的帮助则更是如此。但通常应用对这两者之一具有更严格的要求，因此可能需要在高吞吐量与低时延之间做出取舍以满足其要求。

AI 引擎吞吐量和时延

大多数应用都期望能实现高吞吐量和低时延。Versal 自适应 SoC 可帮助您实现这一目标，有了 AI 引擎的帮助则更是如此。但通常应用对这两者之一具有更严格的要求，因此可能需要在高吞吐量与低时延之间做出取舍以满足其要求。

例如，对于 FIR 滤波器，可在 AI 引擎拼块 (tile) 内使用级联串流来将计算操作分为多个 AI 引擎拼块从而提升总体吞吐量。但这样可能就会导致 FIR 总体时延增大。如需了解更多信息，请参阅 滤波器链简单示例。

出入 AI 引擎阵列以及围绕其移动的数据会直接造成系统时延增大，例如，为内核使用窗口或串流接口。根据窗口大小，在加载输入时，这可能导致额外延迟，因为 AI 引擎会等待整个窗口变为可用后再开始计算。如果采用串流解决方案，那么在传递数据时就会开始计算。串流接口对于高数据速率设计而言是公用的。但这可能导致需要借助额外的 PL 功能来对数据进行排序和对齐以便执行串流。如果采用窗口化解决方案，那么乒乓缓冲可帮助缓解时延增加的问题。

在 PL 与 AI 引擎阵列之间进行通信时，如果 PL 运行速度比 AI 引擎慢得多，则必须留意系统中是否存在额外时延。在此情况下，必须使用更宽的总线（32、64 或 128 位）来将输入传输到阵列中。这样就会造成整体系统时延增加，因为 AXI4-Stream 与 AI 引擎存储器与核之间的互连为 32 位，这导致增加 4 个周期并占用 128 位总线。

提示：在 -1 器件内馈送 AI 引擎拼块并且此 AI 引擎运行速率为 1 GHz 时，使用 64 位 PLIO (500 MHz) 即可充分利用带宽。

为了在 PL 中进行数据处理，针对具有低时延要求的关键功能考虑采用 DSP 引擎和 PL 尤显重要。这样即可避免数据出入 AI 引擎阵列，因为当存在低时延要求时，此类数据可能占用大量资源。

另一个注意事项是，您希望采用何种方式来控制 AI 引擎阵列中的功能或应用。您是否会在运行时使用 PS 或 PL 来控制内核功能？PS 能否满足时延要求，或者您是否需要 PL 控制器？这些都是需要针对特定应用进行评估的重要因素。

欲知详情，请参阅以下文档：

- 《Versal 自适应 SoC AI 引擎架构手册》(AM009)
- 《Versal 自适应 SoC AIE-ML 架构手册》(AM020)
- 《AI 引擎工具和流程用户指南》(UG1076)
- 《AI 引擎内核与计算图编程指南》(UG1079)
- 《AI 引擎机器学习内核与计算图编程指南》(UG1603)
- Vitis 教程：AI 引擎开发

注释：如需了解有关 NoC 时延的信息，请访问此链接以参阅《Versal Adaptive SoC Programmable Network on Chip and Integrated Memory Controller LogiCORE IP 产品指南》(PG313) 中的相应内容。

滤波器链简单示例

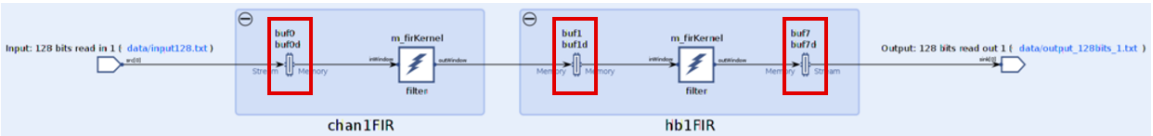
为了演示处理 FIR 滤波器时的吞吐量和时延注意事项，以下显示了 1 个通道滤波器后接 1 个内插半带滤波器的示例。

图 8：**滤波器链简单示例**

- 实现 1：通道滤波器和内插滤波器内核分别布局到其各自的独立 AI 引擎 tile 中。

  乒乓缓冲器深度仅为 1，用于管理这整个简单示例内的数据。

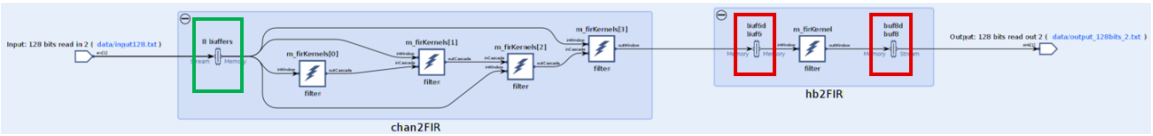
图 9：**Vitis 分析器的 graph 视图**



- 实现 2：通道滤波器分跨 4 个 AI 引擎，以改善吞吐量。

  请注意，输入上的数据需要额外数量的缓冲（包括绿框内的 4 对乒乓缓冲），用于维护这 4 个 AI 引擎内核。但这种方式可以提升总体吞吐量，并且代价是需要牺牲 AI 引擎资源使用率。

图 10：**此 graph 视图显示了分跨 4 个 AI 引擎 tile 的通道滤波器**



NoC 吞吐量和时延

NoC 可提供相应机制以在系统内的不同流量源之间分配吞吐量。此外还提供了各种服务质量 (QoS) 选项用于读写流量。如果吞吐量低于预期（通常是由 DRAM 效率所致），那么您可调整 DRAM 地址映射，为不同种类的流量进行 DRAM 接口最优化。根据流量类，您可以为每条流量路径调整 NoC QoS 值，以满足应用需求。例如，对于需要特定带宽的视频应用，可将其配置为常时等量流量，对于其他 NoC 流量主单元，则可配置为尽力实现最高流量。欲知详情，请访问此链接以参阅《Versal Adaptive SoC Programmable Network on Chip and Integrated Memory Controller LogiCORE IP 产品指南》(PG313) 中的相应内容。另请参阅 GitHub 仓库中提供的 Vivado 设计教程：Versal 片上网络/DDR 存储器控制器性能调优。

功耗要求

根据前述章节中的信息来制定有关应用分区的正确决策，这样才能有效使用 Versal 自适应 SoC 发挥最佳性能。最佳性能意味着所有引擎都能得到有效利用并侧重 AI 引擎，运行的所有引擎都能保持较高的使用率并以低功耗提供高性能（例如，为高度计算型应用提供更高的单位功耗性能）。

如果 AI 引擎核使用率较低，则应考虑在相同 AI 引擎 tile 上运行更多低使用率的内核。这些内核无法同时运行，但如果应用允许，那么此方法可以提升总体 AI 引擎阵列效率。

在 AI 引擎阵列内还可对未使用的 tile 进行时钟门控，此功能默认开启。要将 tile 视作为未使用的 tile，不得启用其中任何组件，即不使用存储体、互连（包括布线穿越）或 AI 引擎核。如有必要，可以使用边界框构造函数来指引内核布局，以确保可实现最大的时钟门控。如需了解更多信息，请参阅《AI 引擎工具和流程用户指南》(UG1076)。

相关信息

功耗和散热规划

安全保障要求

在设计周期早期正确制定有关应用安全保障要求的决策至关重要。Versal 架构提供了诸多安全保障功能特性，包括外设和存储器访问保护、篡改监控与响应以及安全启动流程。

要了解有关安全与隔离功能特性的更多信息，请参阅下列文档：

- 《Versal 自适应 SoC 技术参考手册》(AM011)
- 《Control, Interface and Processing System LogiCORE IP 产品指南》(PG352)
- 《Versal 自适应 SoC 安全手册》(UG1508)

  注释：此文档需要从设计安全性专区下载有效的 NDA。

性能建模和仿真

性能建模和仿真是在 AMD Vivado™ IP integrator 中使用抽象仿真模型来完成的。

概述

性能建模是使用流量生成器、检查器和性能监控器块来模拟应用流量，以分析系统级别性能和时延的一种方法。

在 Versal 自适应 SoC 中，高速 NoC 可提供到嵌入式硬核块的固定连接，也可提供到 PL 块、DSP、MRMAC（通过 PL 布线）和 AI 引擎（通过 PL 串流）的可编程连接。每个 NoC 接口均可与基于 AXI4 的主器件或从器件对接，此类器件支持存储器映射传输事务或串流事务。

对于系统级别性能分析，重要的是对从动态随机存取存储器 (DRAM) 流经 NoC 直到 Versal 自适应 SoC 处理硬件块的流量进行建模。

Versal 自适应 SoC 的以下所有资源块均可使用 NoC 访问来自 DRAM 的数据：

- 处理器系统 (PS)
- 加速器硬件块（如 DSP 和 AI 引擎）
- 基于 PCIe 接口的端点直接存储器访问 (DMA)
- 高速 MRMAC 块

流量场景示例

以下是基于 PCIe 接口的系统的流量分析示例，它通过 PCIe 链路捕获从外部主机到端点器件的数据流阶段：

1. 基于 PCIe 接口的 DMA 通过 NoC 将数据写入 DRAM。

2. 加速器块会通过 NoC 从 DRAM 提取数据，并将数据存储在片上存储器中。
3. DRAM 会存储已处理的数据。
4. 基于 PCIe 接口的 DMA 会通过 PCIe 接口将数据发送回主机 DDR 存储器。
5. 下游 PCIe 接口中断会将有关数据传输完成的信息告知主机处理器。

以下是对应嵌入式系统设计的类似示例，其中主数据可源自以太网接口或任何辅助存储器件：

1. 嵌入式 DMA 器件通过 NoC 将数据写入 DRAM。
2. 加速器硬件通过 NoC 提取数据，并（可选）将数据存储在片上存储器中。
3. 数据移动器块会通过 NoC 将来自加速器块的最终输出数据写入 DRAM。
4. 数据移动器块会将有关数据传输完成的信息告知嵌入式处理器。

Vivado IP integrator 中的性能建模

Vivado IP integrator 中的性能建模可分解为如下若干个阶段：

- NoC/DDR 存储器接口和数据流建模
- 加速器块数据流建模

在第一阶段中，使用 AMD Traffic Generator IP 对 NoC 数据流进行建模，此 IP 可配置为生成类似加速器数据流的传输事务。例如，如果加速器需处理三维数据立方体，并且数据在 DDR 存储器内以线性格式排列，那么来自 NoC 主单元的提取地址并非线性地址。您可为所需寻址模式（例如，三维）配置流量生成器，并在 NoC-PL 接口中监控性能。

如果吞吐量低于预期（通常是由 DRAM 效率所致），那么您可调整 DRAM 地址映射，提升 DRAM 接口效率。此外，NoC 可提供服务质量 (QoS) 选项。根据流量类，您可以为每个 NoC 主单元和从单元调整 QoS 值，以满足应用需求。例如，对于需最低时延的视频应用，您可配置低时延流量，对于其他 NoC 主单元，则可配置为尽力实现最高流量。

在下一阶段，通过为加速器生成流量以模拟来自 NoC/DDR 存储器的真实数据流，对加速器块进行建模。如果加速器接口支持 AXI4 串流协议（例如，AI 引擎块），那么您可使用 AMD 流量生成器或仿真 PLIO 对流量建模并对性能进行调优。您可基于监控器块报告的性能对 NoC 和加速器配置进行调整。

建议：单独为 NoC/DDR 存储器和加速器硬件执行流量建模有助于对每个块进行并行评估。这样您即可先对配置和参数进行微调，然后再将该解决方案部署到更大的系统中，在此类系统中，一般瓶颈主要在于仿真时间。

欲知详情，请访问此链接以参阅《Versal Adaptive SoC Programmable Network on Chip and Integrated Memory Controller LogiCORE IP 产品指南》(PG313) 中的相应内容，另请参阅 NoC DDR 存储器控制器 Versal 器件架构教程。

DDR 存储器使用情况

以 Versal 自适应 SoC 为目标时，请关注器件中集成 DDR 存储器控制器的目标使用率、带宽和性能。例如，图像处理使用大量数据（如图像数据、权重数据、层级数据等），这些数据在不同时间点所需带宽都不尽相同。这些类型的应用会使用大量 DDR 存储器，从而影响 DDR 存储器带宽和时延。根据应用类型及其存储器带宽和性能要求，您需要精确掌握应用生成的流量上的 QoS 要求。

有关 DDR 存储器使用情况的另一个需要考量的重要因素是存储器分区的影响。根据您的应用带宽需求以及对于 DDR 存储器的需求，可能最好跨多个 DDR 对存储器进行交织处理，或者将多个 DDR 存储器控制器各自作为单独的存储器来处理。如需了解 DDR 存储器控制器到 NoC 配置准则，请访问此链接以参阅《Versal Adaptive SoC Programmable Network on Chip and Integrated Memory Controller LogiCORE IP 产品指南》(PG313) 中的相应信息。

HBM 存储器使用情况

AMD Versal™ HBM 系列包含最多 2 个 HBM2e 存储器栈（最高各 16 GB）和集成 AXI HBM 控制器。就像 DDR 存储器一样，您应关注 HBM 控制器的目标使用率、带宽和性能。HBM 控制器可搭配 NoC 一起使用，因此您需要了解 NoC 架构，以便充分利用器件上的 HBM 存储器。HBM 适合各种应用，例如，机器学习、数据库加速、下一代防火墙、航空航天与先进的网络测试器。Versal HBM 系列通过集成的 HBM 提供了强大的并行处理功能和海量存储器带宽。

建议使用适合应用所需数据模式的 AXI Traffic Generator IP 对 NoC 数据流进行建模，以便尽可能发挥其处理能力和性能。您须了解 NoC 架构及相关因素，方能尽可能发掘 HBM 的吞吐量以供应用使用。关于性能，需要考量的因素包括 HBM 工作频率、传输事务规模、AXI 端口频率以及 HBM 的地址映射。如需了解有关 HBM 控制器及其使用准则的更多信息，请参阅《Versal Adaptive SoC Programmable Network on Chip and Integrated Memory Controller LogiCORE IP 产品指南》(PG313)。

性能建模示例与教程

如需了解有关性能建模的更多信息，请参阅 GitHub 仓库中提供的下列教程：

- Versal 片上网络性能 AXI Traffic Generator 教程
- Versal 片上网络/DDR 存储器控制器性能调优教程

系统设计注意事项

仅限硬件的系统设计注意事项

在仅限硬件的系统设计中，关键注意事项之一是设计中的数据流。通常这些设计都具有下列组件：

- 多个高速 I/O 接口
- 内部数据缓冲和存储，具有由片上 RAM 和外部 DDR 存储器组成的存储器层级
- 内部数据处理逻辑

对于能够处理外部和内部流量带宽和时延要求的设计，为其创建 DDR 存储器控制器到 NoC 配置至关重要。AMD 建议先执行流量分析以评估并最终明确流量，然后再继续执行设计的整体集成和实现阶段。

仅限硬件的系统与嵌入式系统之间的主要差异在于，仅限硬件的系统不依赖于嵌入式处理器或外部处理器。仅限硬件的系统使用 PMC 作为对已实现的硬件设计进行编程和控制的主要途径。在大部分情况下，Versal 自适应 SoC 仅限硬件的系统设计遵循 AMD UltraScale+™ 器件设计方法论，但唯一例外是 Versal 器件上实现的硬核 IP，具体来说就是 DDR 存储器控制器、NoC 和硬化的控制器（例如，PCIe® 接口和 MRMAC IP）。

相关信息

性能建模和仿真

嵌入式系统设计注意事项

鉴于软件栈与硬件开发流程交互紧密，本章节涵盖了嵌入式系统设计值得考量的特殊注意事项。以下是此开发流程中的关键步骤。根据随附的是嵌入式系统还是服务器系统，每个步骤所面临的难题都不尽相同。

1. 硬件与软件协同开发
    a. 架构
    b. 数据路径和传输层
    c. 控制层
    d. 存储器层级
2. 软件开发，目标是有效利用硬件加速
    a. 启动和操作系统注意事项
    b. 软件应用开发
    c. 软件调试

软硬件计算加速开发

加速器软硬件协同设计

以下加速器设计步骤适用于嵌入式系统。

架构

系统设计架构过程中需要解决的主要难题之一是功耗和性能最优化。您所选的加速硬件（无论是 PL 还是 AI 引擎）取决于算法的类型以及数据入口和出口路径。对于往来传感器（例如，LiDAR、RADAR、双镜头视觉系统）的串流数据入口和出口，数据可通过高速收发器以供互连结构使用。此数据是从 AXI4-Stream 总线上的外部协议接口汇总所得，可分发至 PL 或 AI 引擎。

标量引擎（处理器子系统）、自适应引擎（可编程逻辑）和智能引擎（AI 引擎）共同构成紧密集成的异构计算平台。标量引擎可提供复杂的软件支持。自适应引擎可提供灵活的定制计算和数据移动功能。鉴于其高计算密度，AI 引擎非常适合用于基于矢量的算法。

在此步骤中，将开发从核应用以及每个算法到 Versal 自适应 SoC 中最适合的架构区域（例如，AI 引擎、PS、PL、NoC 和 DDR 存储器控制器）的映射。其中包括映射应用中的所有主要块，并考量这些主要块在带宽和可用性方面的要求。此应用映射和设计分区步骤采用手动操作。

对于 PL 和 AI 引擎中无需并发使用的各区域，您可为其实现其他功能，例如，时钟门控。对于传统多时钟域互连结构设计和数据路径时钟域交汇，可采用与 FPGA 架构相同的方法来进行处理。

相关信息

应用映射和设计分区

数据路径和传输层

在嵌入式设计中，数据路径的主要目的是采集系统应用中感兴趣的数据流。对于视觉处理系统，数据流可能包含来自图像传感器（如，摄像头、LiDAR 等）的传入数据，这些数据存储在片上和片外系统存储器（例如，BRAM、UltraRAM、DRAM）以供内联处理。对于网络系统，数据流可能是 2 种标准（如以太网和 Interlaken）之间的协议桥接器。

PL 加速器数据路径（入口和出口）映射到 AXI4-Stream 接口和控制接口，后两种接口则映射到 AXI 存储器映射互连。控制接口支持通过软件栈来控制这些加速器。

同样，AI 引擎加速器数据路径映射到 AXI4-Stream 接口。AI 引擎内核具有运行时参数 (RTP) 接口，支持运行时更新。您可通过 Xilinx Runtime (XRT) API 访问 RTP 功能特性。AI 引擎还支持全局存储器访问 DDR 存储器以查找特定于应用的缓冲器。

控制层

控制路径用于统筹系统控制功能，例如，加速器初始化、数据传输初始化和加速执行等。AMD 建议使用 XRT API 来控制嵌入式系统的路径设计。如需了解更多信息，请参阅《XRT 版本说明》(UG1451)。

存储器层级

对于需要高带宽存储器的数据密集型应用，最佳方法是构造 1 个应用专用的存储器层级，例如，从 DDR 存储器到 PL RAM (UltraRAM/BRAM) 的存储器高速缓存。Versal 器件可通过 NoC 接入 DDR 存储器。DDR 存储器连接至器件互连结构，Arm Cortex-A72 与 AI 引擎则通过 NoC 相连。在 PL 中使用 DMA 数据移动器即可协调数据在硬核 IP 与 DDR 存储器之间的往来迁移，并利用中间 PL RAM 阶段进行高速缓存或数据缓冲。

注释：与先前架构一样，Versal 器件同样支持软核存储器控制器。

Versal 器件可使用 DMA 接入 DDR 存储器。DDR 存储器通过硬核存储器控制器和 NoC 连接到器件互连结构和其他硬核 IP。您可在 PL 中使用 DMA 数据移动器来协调往来 DDR 存储器的数据迁移。您还可使用 Versal 器件上提供的 DDR 存储器控制器来配置 NoC 以实现最大带宽。

提示：通过使用 NoC 编译器即可找到最优解决方案，以获取所需聚合带宽。

软件开发

启动和操作系统

Versal 器件具有集中平台管理控制器 (PMC) 用于在上电复位之后启动器件。Versal 器件支持不同启动模式，包括主启动模式（JTAG、SD、eMMC、OSPI、QSPI、SelectMAP）和辅助启动模式 (PCIe)。根据启动时间要求，您必须选择相应的启动器件。

- 基于 PCIe 的应用要求在系统上电后的 100 ms 内检测到端点，对于此类应用，请使用更快的启动器件，如 OSPI。可通过采用镜像分区，使最小启动镜像能够在 100 ms 内从 OSPI 启动模式完成启动，而较大的启动分区则可通过 PCIe 以串联 PCIe 启动模式来进行传输。
- 请将 QSPI、SD 和 eMMC 启动器件用于一般嵌入式应用。请将 eMMC 启动模式用于需要更高密度的嵌入式应用。
- 请将 JTAG 启动模式用于系统初始化和调试。要启用系统调试，请确保启动模式管脚设置为 JTAG 或者通过从 TAP 链配置启动模式寄存器来切换至 JTAG 启动模式。
- 如果多个 Versal 器件需要单一外部启动控制解决方案，那么对于此类应用可使用 SelectMAP。

对于需要器件级安全性的应用，需实现 Versal 自适应 SoC 硬件所支持的启动镜像加密和身份验证。启用加密和身份验证后，系统启动时间会相应增加。如需了解更多信息，请参阅 AMD 网站上的设计安全性专区（需注册）中提供的《Versal 自适应 SoC 安全手册》(UG1508)、《非对称硬件信任根安全启动》(XAPP1357) 和《对称硬件信任根安全启动》(XAPP1358)。如需了解有关启动时间估算器的更多信息，请参阅技术支持页面。

Versal 自适应 SoC 闪存启动器件支持分区回退以避免出现灾难性的启动故障。在现场升级期间，如果升级后的镜像包含错误，那么 Platform Loader and Manager (PLM) 可回退至黄金镜像，以恢复启动模式。黄金镜像能够与其他升级镜像驻留在相同的启动器件内。

请根据您的具体应用用例来选择操作系统。如果应用需要缓冲器管理、锁定访问和中断处理且有多个进程并行运行，请使用 Linux 操作系统。这些应用可以利用 Linux 开源框架所提供的更高的抽象层。典型应用包括视频、OpenCL™、OpenCV 和可利用 Linux 框架的网络栈。AMD 提供了 Linux 运行时支持，它使用 Xilinx Runtime (XRT) 栈来处理中断管理、内核启动与停止、缓冲器分配以及共享。XRT 能够与更高级别的软件栈（例如，OpenCL、OpenCV、FFmpeg 和基于 Python 的框架）进行交互。使用基于 Linux 的栈的缺点之一是栈开销不适合实时操作。并且，Linux 操作系统要求全功耗域 (FPD) 通电。需要省电的应用可使用能在低功耗域 (LPD) 下运行的 Arm Cortex-R5F 处理器。

需要实时处理的应用则可使用 Versal 自适应 SoC 中的 Arm Cortex-R5F 处理器，ArmCortex-R5F 处理器符合 ASIL-C 安全规范。典型应用映射包括系统监控、硬件监控、直接硬件控制（使用轻量级栈）等。如需使用 Arm Cortex-R5F 处理器以保证应用的功能安全性，AMD 建议将应用代码置于紧密耦合存储器 (TCM) 内，而不是从 DDR 存储器访问此代码。Arm Cortex-R5F 处理器还可作为 Linux 操作系统的协同处理器来工作，用于监控特定硬件功能并向 Linux 应用提供硬件状态。Linux 操作系统与 Arm Cortex-R5F 上的 RTOS/裸机操作系统之间能够通过处理器间中断来进行通信。

Versal 器件支持虚拟化。您可针对使用虚拟机管理器的多个访客操作系统使用相同硬件。虚拟化硬件的中断处理开销耗时较长。虚拟化可能导致时延。如果您的应用对时延敏感，请勿使用虚拟化。

软件应用开发

Versal 自适应 SoC 包含多种设计流程，其中包括 2 个面向包含软件应用的工程的流程。适用的建议则因所用设计流程而异。如需了解更多信息，请参阅《Versal 自适应 SoC 系统软件开发者指南》(UG1304)。

固定硬件平台建议

为使用固定硬件平台设计流程创建的设计开发软件应用时，AMD 建议使用定制 Linux 驱动程序来管理应用与 PL 资源之间的交互。

可扩展硬件平台建议

开发软件应用以供通过 Vitis 平台设计流程和 `v++` 连接器构建的硬件使用时，可以使用标准 AMD 嵌入式开发流程，但为了提升 Linux 应用开发的易用性，AMD 建议使用 Xilinx Runtime (XRT) API 来管理 PL 内核与 AI 引擎 graph。对于用户管理的内核，您可以采用 XRT C++ API 来读写内核的 AXI4-Lite 控制接口。XRT API 已经过最优化，以便与这些接口进行交互，并提供抽象方法，用于与加速器进行交互。此外，使用 XRT API 也支持访问内建的剖析和调试功能。

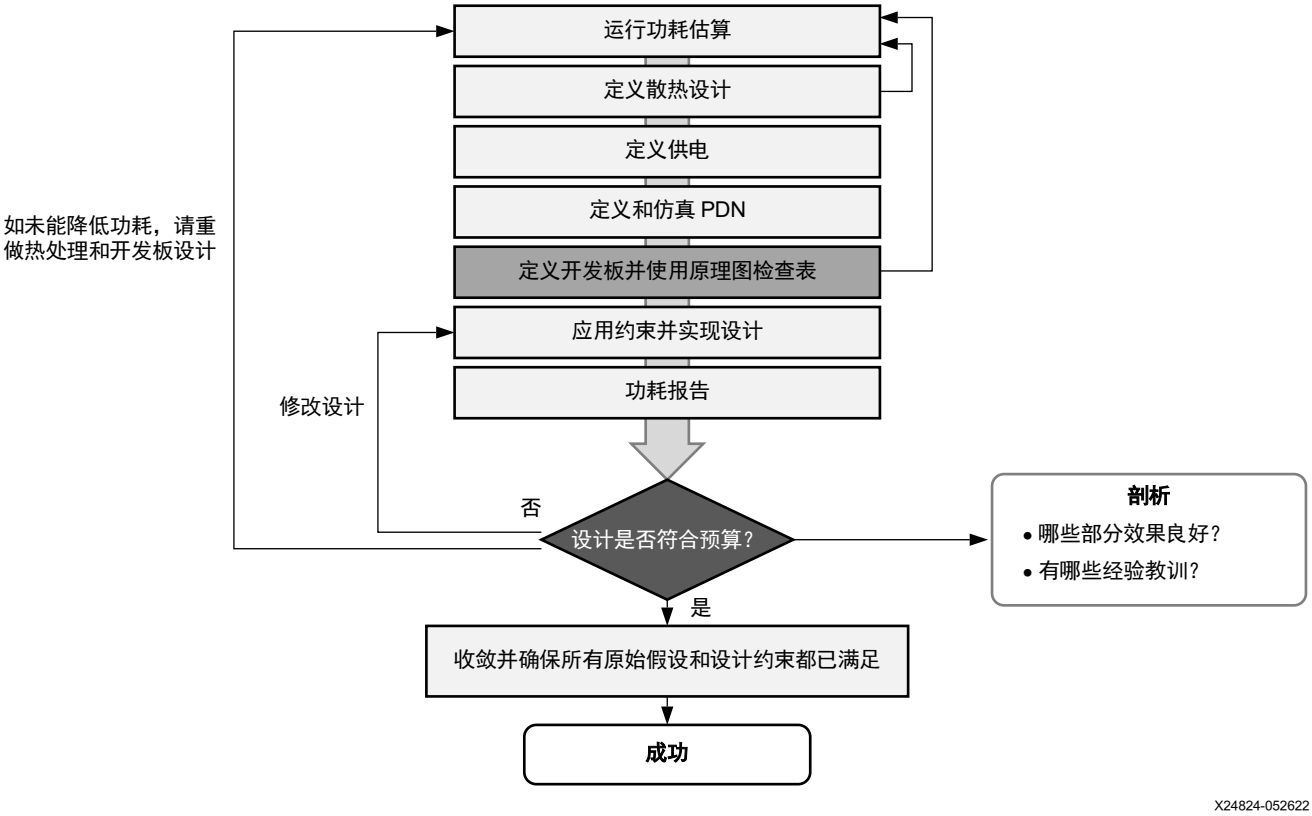
由于可使用 XRT API 进行交互的对象仅限于使用 Vitis 环境链接到平台的 PL 和 AI 引擎加速器，因此直接包含在平台内的 PL 资源就必须由开发者使用定制驱动程序来进行显式管理。AMD 还建议对设计架构进行微调，以允许应用将用户定义的 PL IP 复位至已知良好运行的状态，以便解决各种错误，并且这样也可以在 AI 引擎完成异步软核复位和重新加载之后按需进行 PL 状态复位。如需了解有关 Vitis 环境和 XRT 的更多信息，请参阅《Vitis 统一软件平台文档》(UG1416)。

第 4 章

功耗和散热规划

为确保开发板设计成功，您必须考量与功耗估算、散热设计、供电和去耦相关的所有关键组件，并针对每个组件使用正确的起点。开发板变更或重新设计从上市时间 (TTM) 和一次性工程 (NRE) 成本两方面来看，代价都是非常昂贵的。AMD 建议采用下图所示以及本章中所述的方法论来确保开发板设计使用正确的输入从而产生最佳的成果。

图 11：**功耗和散热方法论**



运行功耗估算

开发板设计的基础是功耗估算，功耗估算用于确定散热、供电和去耦网络要求。理想情况下，功耗估算通过约束 AMD Vivado™ 工具或 AMD Vitis™ 环境中的设计来确保开发板设计正确无误。使用 AMD Versal™ 自适应 SoC 电源设计管理器 (PDM) 工具来估算功耗要求。如需了解更多信息，请参阅以下资料：

- 《电源设计管理器用户指南》(UG1556)
- 《使用 Xilinx Power Estimator 实现准确的最差情况功耗分析七步法》(XAPP1348)

- AMD 网站上的电源效率页面
- AMD 网站上的电源设计管理器 (PDM) 工具（从 china.xilinx.com/power 下载）页面

注释：您可将现有 AMD Zynq™ UltraScale+™ MPSoC 设计移植到 Versal 自适应 SoC PDM 工具。但请确保您的设计可充分利用 Versal 自适应 SoC 架构块来最大限度提升设计中的效率。

定义散热设计

散热设计是所有开发板设计过程中至关重要的一环，并且必须对于应用的机械设计同样有效。散热设计必须确保应用保持处于其热限值范围内。AMD 为所有器件都提供了散热模型，可与 Siemens Flotherm 和 Ansys IcePack 搭配使用。

您可通过热仿真来获取系统的 Theta Ja，热仿真可确保功耗估算和供电解决方案更精确。Versal 器件支持无盖封装，这样可提供最优化的热处理解决方案，但要求现有有盖设计将目标定为无盖封装。Versal 器件支持温度漂移，上限为 110°C 且不超过器件寿命的 3%，从而提供额外的散热裕度并简化散热设计。

您可利用热仿真的结果来优化初始功耗估算，以获取更准确的结温估算结果。输入应用的最大环境温度（以 Theta Ja 为单位），以便提供最差情况下的静态功耗并确保正确设计供电解决方案。

如需了解更多信息，请参阅 AMD 网站上的热设计页面以及《利用温度漂移扩展热处理解决方案》(WP517)。

供电定义

借助与各家领先的供电供应商的协作，AMD 提供了优化的解决方案，以帮助您找到适合自己的 AMD 应用的供电解决方案。这些经硬件验证的参考设计可确保以最优方式满足所有 AMD 功耗规范，并遵循 AMD 支持的上电/下电排序。请参阅 AMD 网站上的供电解决方案页面以获取有关这些解决方案的详细信息。

定义和仿真 PDN

Versal 自适应 SoC PDM 工具可动态生成设计所需的去耦电容。通过使用 PDM 工具中的“Power Design”（电源设计），即可查看精确的去耦要求（基于其估算）。请按 PDM 工具或《Versal 自适应 SoC PCB 设计用户指南》(UG863)中所述方式对这些去耦电容进行布局。

AMD 提供了 S 参数模型来确认您的印刷电路板 (PCB) 设计，并调整去耦要求，同时确保配电网络 (PDN) 适合您的设计。请访问 AMD 网站上的此页面申请获取这些模型。

开发板的定义和板级原理图检查表的使用

定义开发板和板级原理图布局时，请考量先前所有步骤的结果，包括功耗估算、供电以及散热设计和去耦要求等。AMD 还提供了板级原理图检查表，用于确保开发板设计的所有关键阶段中的问题都能得到解决。如需了解更多信息，请参阅以下资料：

- 《Versal 自适应 SoC PCB 设计用户指南》(UG863)

- 《Versal 自适应 SoC 板级原理图审查检查表》(XTP546)

约束应用、设计实现与功耗报告

欲知详情，请访问此链接以参阅《Versal 自适应 SoC 设计指南》(UG1273) 中的相应内容。

验证设计约束

在这最后一步中，您必须确保估算、仿真、开发板布局和设计全部结合在一起仍能有效运作，并确保设计能满足以下要求：

- 在供电限制范围内，并在 AC 纹波和 DC 容限前提下，能有效运作
- 在最大电流要求限制范围内，能有效运作
- 使最大结温保持在规格要求范围内并与热仿真相互关联

最后，分析设计进程并建立起各项结果之间的关联，将所学到的经验教训应用于将来设计中。以下是需要解答的问题：

- 初始估算的准确性如何？

  如果初始估算准确，则可将相同方法应用于具有类似应用的设计，并且能够沿用使能和翻转率。
- 热仿真与硬件仿真的接近程度如何？所需裕度是多少？能否改进容限？是否施加了足够的压力？

  AMD 建议施加每平方英尺 (PSI) 20 到 50 磅力。
- 总体上开发板设计是否成功？如果成功，那么有哪些方面表现良好，又有哪些方面值得改进？如果不成功，有哪些方面需要改变以确保开发板能够在最初取得成功？

第 5 章

系统调试规划

AMD Versal™ 自适应 SoC 包含调试架构，支持系统调试方法论增强功能。此调试架构设计为可在任何环境下工作，包括实验室、数据中心和边缘计算环境。如需了解有关调试的更多信息，请访问此链接以参阅《Versal 自适应 SoC 系统集成和确认方法指南》(UG1388) 中的相应内容。

Versal 自适应 SoC 调试架构包含集中调试包控制器 (DPC)，它是用于调试架构的包处理引擎。由 DPC 所处理的包被称为调试追踪包 (DTP)。这些包经 DPC 解码以判定命令、目标以及任何潜在更高层次的流量控制和管理任务。DPC 会对主机发送的 DTP 进行处理、执行包中内嵌的任意命令并生成响应以发送回主机。如需了解有关 DPC 的更多信息，请访问此链接以参阅《Versal 自适应 SoC 技术参考手册》(AM011) 中的相应内容。

注释：安全启动时，默认禁用 JTAG。如需了解更多详情，请参阅《Versal 自适应 SoC 安全手册》(UG1508)。本手册需要从“设计安全性专区”下载有效的 NDA。

选择调试接口

Versal 架构可提供多个物理接口，用于将调试器连接到 DPC。下表显示了针对不同用例建议使用的调试接口。

注释：PS 的各 APU 和 RPU 均可通过集成到 PS 中的 Arm® CoreSight™ 基础架构来进行调试。CoreSight 架构可通过 JTAG-DAP、HSDP、PL 和 PCIe® 接口来访问。欲知详情，请访问此链接以参阅《Versal 自适应 SoC 技术参考手册》(AM011) 中的相应内容。

表 4：**基于调试目标建议使用的调试接口**

建议的调试接口	调试目标	注释
JTAG	· 基本硬件调试，使用 AXIS-ILA、AXIS-VIO 和其他硬件调试核 · 小型 PDI 下载	使用 JTAG 允许低速连接至设计中的所有调试核，无需进行额外设计修改
JTAG + HSDP（含 SmartLynq+ 的 Aurora）	· 高级硬件调试，使用许多 AXIS-ILA、AXIS-VIO 和其他硬件调试核 · 不使用 SD 卡的情况下启动大型 Linux 镜像，使用 XSDB 初始化存储器空间 · 大型 PDI 下载	将 HSDP 与 SmartLynq+ 模块搭配使用即可建立速度比 JTAG 快得多的调试连接，且只需对设计进行些许修改即可

通过 JTAG 进行调试

PMC 包含 JTAG 接口，此接口可用于对 Versal 自适应 SoC 上运行的设计进行编程和调试。此 JTAG 接口包含 2 个级联块：调试访问端口 (DAP) 和测试访问端口 (TAP)。DAP 主要用于访问 PS 的各项调试功能，也可用于对 PS 地址范围内的任何可访问的寄存器和存储器位置进行低带宽的读写访问。TAP 接口则主要用于访问器件配置和边界扫描基础架构，也包含用于配置和访问 DPC 功能的指令。

通过 Aurora 进行调试 (HSDP)

PS 包含集成 Aurora 64B/66B 块，专用于通过基于高速 GT 的接口来接入 DPC。此协议可访问称为高速调试端口 (High-Speed Debug Port, HSDP) 的 DPC。HSDP 可提供从外部主机调试/追踪模块到器件的双向访问，并支持高速调试和追踪操作。您可将 SmartLynq+ 模块连接到 Aurora 接口以便访问 Versal 自适应 SoC 中的 HSDP。欲知详情，请访问此链接以参阅《SmartLynq+ 模块用户指南》(UG1514) 中的相应内容。如需了解有关 HSDP 四通道可用性的信息，请访问此链接以参阅《Versal 自适应 SoC 技术参考手册》(AM011) 中的相应内容。

注释：集成 HSDP Aurora 接口并非在所有 Versal 自适应 SoC 中都可用。此接口需在 Control, Interfaces, and Processing (CIPS) IP 内进行额外配置方可使用，并且它使用额外的千兆位收发器。

通过 PL 互连结构进行调试

DPC 也可通过 PL 互连结构来进行访问。这样可支持您采用除专用 CPM/PCIe 途径、集成 Aurora 和 JTAG 以外的其他方法来将调试集成到自己的系统中。

通过 CPM PCIe 接口进行调试

在数据中心应用和其他 PCIe 接口托管的系统中，同样可通过 CPM 内用于主机到自适应 SoC 通信的 PCIe 接口来访问 DPC。您可以选择将调试函数映射到 PCIe 物理功能和 BAR 空间的方法。调试函数包含专用 PCIe HSDP DMA 引擎，用于在 DPC 与主机存储器之间移动调试数据。

注释：CPM 可用性与器件相关。如需了解更多信息，请参阅《Versal 架构和产品数据手册概述》(DS950)。

AI 引擎调试规划

您可使用多种方法来调试 AI 引擎计算图与内核。通常在计算图与内核调试早期阶段，需确保计算图与内核功能准确。AMD 可提供各种仿真流程以帮助确保功能准确性，并且强烈建议在设计流程中尽早运行这些仿真。

事件 (Events) 是 AI 引擎调试和性能分析的一项重要功能。事件类似于触发器。如果周期内与事件关联的条件为 true，则此周期内的事件信号为高电平。事件示例包括：Conflict DM bank 0、Lock 11 Released、Floating point Overflow 和 PC event 0。每个事件都有唯一的 7 位编号，每个 AI 引擎内最多包含 128 起事件。事件追踪功能允许您捕获与硬件中的 AI 引擎、存储器模块和接口模块相关联的事件。

提示：aiecompiler 包含为 AI 引擎计算图启用事件追踪的选项。这些选项用于设置事件追踪路径，以便在硬件上运行设计时捕获事件追踪数据。

事件追踪流程由下列步骤组成：

1. 运行事件追踪构建流程。
2. 在硬件中运行设计并在运行时捕获追踪数据。
3. 查看和分析追踪数据。

建议：在流程中尽早启用事件追踪选项。如果在流程后期才启用事件追踪，那么您必须重新编译设计，并重新运行 v++ 链接步骤和 v++ 封装步骤。

如需了解有关硬件上的 AI 引擎计算图应用的性能分析的更多信息，请访问此链接以参阅《AI 引擎工具和流程用户指南》(UG1076) 中的相应内容。

AI 引擎阵列中的每个 AI 引擎都具有 1 个调试接口，此接口可用于对每个 AI 引擎寄存器执行读写。读取和写入 AI 引擎寄存器的请求通过 AXI4 存储器映射 (AXI4-MM) 接口发送，随后被转发至 AI 引擎调试接口。AI 引擎中的所有寄存器都在 AXI4-MM 上映射。AXI4-MM 接口具有 1 个 32 位读/写总线。您可指定任意 AXI4-MM 映射地址以通过 AXI4-MM 接口来执行读取。任何外部 AXI4-MM 主接口（例如，PS）均可通过写入控制/状态寄存器来向特定 AI 引擎发出停滞信号。系统控制（例如，常规编程流程）和调试器各自分别使用独立的寄存器。AMD Vitis™ 系统调试器 (System Debugger) 可提供全功能的源代码调试器以帮助调试 AI 引擎计算图与内核。如需了解有关 AI 引擎应用调试的更多信息，请访问此链接以参阅《AI 引擎工具和流程用户指南》(UG1076) 中的相应内容。

如需了解有关 AI 引擎硬件剖析和调试方法论的更多信息，请访问此链接以参阅《AI 引擎工具和流程用户指南》(UG1076) 中的相应内容。

PS 调试规划

DAP 支持采用 Arm CoreSight 对 PS 进行调试和追踪。其中包括安全调试支持和非安全调试支持。根据 Arm 调试接口第 5 版 (ADIv5)，DAP 还可通过 JTAG 连接至外部 Arm 调试工具。TAP 可通过 JTAG 边界扫描 (Boundary Scan) 接入包含各种错误和警报状态位的 ERROR_STATUS 寄存器。Vitis System Debugger 可提供集成设计环境 (IDE) 用于对裸机应用和基于 Linux 的应用进行 PS 调试。

PL 和硬核块调试规划

如果遇到在 PL 逻辑仿真中难以复现的情况，您可能需要调试可编程逻辑 (PL) 和硬核块。PL 和硬核块支持使用以下 ChipScope™ 调试 IP 核和硬核块进行逻辑调试：

- AXI Streaming Integrated Logic Analyzer (AXIS-ILA)：AXIS-ILA 核支持您通过触发硬件上事件并以设计速度捕获数据来对实现后设计执行系统内调试。
- AXI Streaming Virtual Input/Output (AXIS-VIO)：AXIS-VIO 核支持您实时监控和驱动设计信号，以取代物理输入或输出元素，如开关或指示灯 (LED)。
- Integrated Bit Error Ratio Tester (IBERT) GTY/GTYP：Versal 自适应 SoC GTY/GTYP 包含内置的 IBERT Serial Analyzer 功能，此功能支持系统内串行 I/O 确认和调试。此解决方案无需额外的 PL IP。
- NoC DDR 存储器控制器校准调试：集成到 Versal 自适应 SoC NoC 内的 DDR 存储器控制器支持校准调试接口，此接口可通过 AMD Vivado™ 硬件管理器来访问。
- PCI Express 链路调试：Versal 自适应 SoC PCI Express® 集成块支持链路调试接口。如果启用此集成块，那么您可在 Vivado 硬件管理器中查看链路训练状态机 (LTSSM) 状态转换。

软件调试规划

根据设计中使用的处理器和软件栈的类型，软件调试可能需要在设计中启用特定硬件接口。在启动整个软件栈之前，您必须确保处理器、硬件外设和互连都正常运行。您可以使用 JTAG 接口来运行基本启动测试，以确保硬件块正常运行。下一步是启动操作系统 (OS) 框架。特定调试可能需要查看启动 log 日志，检查操作系统加载是否成功。加载操作系统后，您可以使用 AMD 提供的调试工具或开源调试工具来运行特定应用和调试应用，在应用处理单元 (APU) 中添加内置定时器来剖析特定函数调用等。

下面是有关特定软件调试方法的详细信息：

- 基于 XSCT 的调试：软件命令行工具 (XSCT) 提供了一组用于启用硬件调试的命令。您还可使用赛灵思系统调试器 (XSDB) 来运行基于 Tcl 的命令和过程，以检查特定硬件的状态。在启动整个软件栈之前，通过运行 XSDB 命令，检查硬件是否正常运行，如《Vitis 嵌入式软件开发流程文档》(UG1400) 中的软件命令行工具 中所述。如果设计包含 DDR 存储器，您可以运行读取/写入测试以验证 DDR 存储器是否正常运行。要调试裸机应用，请使用 XSCT 调试器。您可以使用 XSCT 下载应用，并使用 Vitis 调试器启用单步断点插入来调试特定函数。Vitis 调试器支持您查看存储器和处理器寄存器内容以帮助分析故障代码。如需了解更多信息，请参阅 vitis -debug 命令行。

- 使用软件工具进行调试：您可以使用 GNU 调试器 (GDB) 来调试基于 Linux 的应用，包括插入断点和恢复程序。您还可以使用基于 Linux 的 Valgrind 工具来帮助分析程序中的存储器泄漏问题。构建 Linux 镜像时，启用 GDB 和 Valgrind 工具以支持特定于应用的调试。

- 特殊处理器调试：要调试在特殊处理器（如 AI 引擎或 MicroBlaze™ 处理器）上运行的代码，必须在设计中启用特定硬件接口。AI 引擎处理器阵列具有事件/追踪功能，可帮助查看 PC 事件或执行追踪。您必须在 AI 引擎编译阶段启用此硬件接口。MicroBlaze 处理器具有 MicroBlaze Debug Module (MDM) 接口，此接口可连接到系统 JTAG 链。如需了解更多信息，请参阅《MicroBlaze 处理器参考指南》(UG984) 和《AI 引擎工具和流程用户指南》(UG1076)。

欲知详情，请访问此链接以参阅《Versal 自适应 SoC 系统集成和确认方法指南》(UG1388) 中的相应内容。

第 6 章

系统验证规划

AMD Versal™ 自适应 SoC 及其不同计算域所附带的复杂性给传统 FPGA 仿真方法带来了诸多挑战。对于传统 FPGA 仿真，大部分设计都可使用逻辑仿真来验证。对于 Versal 自适应 SoC，可编程逻辑只是其中的计算域之一，仿真方法必须考量软件域和所用的 AI 引擎域。

Versal 自适应 SoC 的系统仿真方法论以分层方法为基础。此方法论确认需要对每个计算域独立进行仿真，同时还要能够在适当时候对整个系统进行仿真。

系统仿真方法论是围绕以下关键概念构建的：

- 仿真范围：仿真可包含整个系统或者也可仅包含部分系统。AMD 建议先单独对各块和功能进行测试，然后再将其集成到整个系统中并进行仿真。您可使用不同仿真流程来测试不同的计算域，包括 PS、PL 和 AI 引擎。
- 仿真的抽象层：在某些情况下，您可在不同的抽象层级对特定功能进行仿真。例如，您可将 AI 引擎代码作为未定时模型或周期近似模型来进行仿真。您可将 HLS 代码作为未定时模型或周期精确的 RTL 模型来进行仿真。您可将特定 Versal 自适应 SoC 基础架构块（例如，NoC 或 DDR 存储器控制器）作为 SystemC 传输事务级模型 (TLM) 或者作为 RTL 模型来进行仿真。抽象层支持您在仿真速度与仿真精度之间取舍平衡。
- 仿真的目的：每个仿真的目的各异。例如，以功能确认为重还是以性能测量为重？目的是测试单一功能还是测试多个功能之间的交互？不同仿真目的依赖于不同仿真设置与配置。而仿真目的则与范围和抽象层紧密关联。

注释：如需了解更多信息，请参阅《Versal 自适应 SoC 设计指南》(UG1273) 和《Versal 自适应 SoC 系统集成和确认方法指南》(UG1388)。

仿真建议

以下是 Versal 自适应 SoC 仿真建议：

- 根据范围和目标用途选择适当的仿真流程和抽象层。
- 先对每个组件单独进行仿真并验证，然后再将其加以整合并运行硬件仿真（包括，AI 引擎 graph、HLS 内核、RTL 块、硬件平台和 PS 代码）。
- 测试不同块和功能时，尽可能复用测试激励文件和测试矢量。例如，如果某个块的输出为另一个块的输入，那么复用测试矢量来对这两个块进行仿真即可简化集成流程。
- 执行递进式系统集成。您无需从头开始在整个系统上运行仿真。对一小部分 PS、PL 和 AI 引擎组件运行仿真即可奠定已知的基础，以便您逐步添加功能。
- 对每一次设计更改进行仿真和验证。问题发现得越早，越易于解决。

第 7 章

系统确认规划

AMD Versal™ 自适应 SoC 的系统确认规划要求您把关键基础架构构建到系统中，如前述章节中所述。这样即可支持对整体设计特性、性能、功耗等进行系统化确认，这是将系统定性为适合量产所必需的。本章主要围绕基于系统设计类型制定系统确认规划的关键领域展开。如需了解有关确认的信息，请访问此链接以参阅《Versal 自适应 SoC 系统集成和确认方法指南》(UG1388) 中的相应内容。

以下是制定系统级确认规划时的典型步骤：

- 上电并执行电源检查
- 基本启动和器件配置
- 初始化器件的功能子系统
- 初始化软件栈，并运行所有运行时驱动程序或应用编程接口 (API)

  注释：对于部分系统，此步骤为可选。
- 性能确认
- 功耗确认

在上电和电源确认阶段，系统设计师必须参阅《Versal 架构和产品数据手册概述》(DS950) 以确保所有电源都达到期望电平且各功耗域内的耗电量都符合预期。系统设计师必须制定计划，根据系统中完成的电源合并来对开发板上的电压域开展测试。设计师可使用 PDM 工具或 `report_power` 基于设计估算结果来获取不同功耗域中的电流量。

判断电源和耗电量正常后，下一个初始化阶段通常是器件配置。器件初始化的典型方法是使用 Versal 器件上的 JTAG 端口。器件可从标准 AMD Vivado™ 硬件管理器进行检测和寻址，如《Vivado Design Suite 用户指南：编程和调试》(UG908) 中所述。此外，Versal 自适应 SoC 具有复杂的系统监控器 (System Monitor)，允许监控片上供电和温度。硬件系统设计师可以遵循《Versal 自适应 SoC 系统监控器架构手册》(AM006) 中的要求启用系统监控器。

初始化后，仅限硬件的系统的焦点在于初始化内部硬化的 IP 子系统，包括任何复杂的 GT 和 I/O 以及设计中的 DDR 存储器控制器子系统。欲知详情，请参阅以下资源中的硬件确认部分：

- 对于在硅片上进行 NoC/DDR 存储器初始化，请参阅 Vivado 设计教程：Versal 片上网络/多 DDR 存储器控制器，另请访问此链接以参阅《Versal Adaptive SoC Programmable Network on Chip and Integrated Memory Controller LogiCORE IP 产品指南》(PG313) 中的相应内容。
- 对于 AI 引擎子系统的初始化，请参阅 Vitis 教程：AI 引擎波束成形。
- 对于 PS 子系统，请参阅 Versal 自适应 SoC 嵌入式设计教程。

提示：其中大部分硬化的 IP 子系统均可独立于最终系统功能设计来单独进行初始化，以便您检查整体系统集成的基本功能。

通过基本测试确认整体 Versal 自适应 SoC 子系统正常后，系统设计师即可开始对系统的真实功能模型进行初始化。这通常包含在系统级别开展部分测试开发，此类测试开发由整体系统设计驱动。根据系统类型，测试开发可能涉及软件栈。

确定系统设计的基本功能后，系统设计师即可运行性能级别测试案例来对关键性能要求进行校准和确认。示例如下：

- 64 天线系统的信号处理吞吐量，如 Vitis 教程：AI 引擎波束成形中所示
- 经修改的国家标准和技术协会 (MNIST) 数据库中使用 LeNet 卷积神经网络 (CNN) 进行图像识别的每秒可识别图像数，如 Vitis 教程：AI 引擎 LeNet 中所示
- 对于仅限硬件的系统，NoC/DDR 性能，如 Vivado 设计教程：Versal 片上网络/DDR 存储器控制器性能调优中所示

设计运行性能达标后，系统设计师即可为 Versal 器件上的静态功耗和动态功耗执行功耗测量。要制定功耗测量规划，系统设计师必须确保开发板设计支持接入功耗管理集成电路 (PMIC) 接口，从而支持对运行性能达标的系统执行实时电流测量。

下表基于设计类型汇总了系统确认信息。

表 5：**系统确认汇总**

设计类型	启动与器件配置	子系统功能确认	系统功能确认	系统性能确认示例	系统功耗确认
仅限硬件的系统	• JTAG • QSPI/OSPI • SD/eMMC	使用如下工具： • IBERT • XSDB/XSCT • AXI-ILA/AXI-VIO • 系统监控器	仅限硬件	• MAC 吞吐量 • 存储器吞吐量	• PDM 工具，用于估算 • 合并功耗域并在开发板上执行评估
嵌入式系统	• JTAG • QSPI/OSPI • SD/eMMC	使用如下工具： • IBERT • AMD Vitis™ 嵌入式软件工具 • 系统监控器 • XSCT • Linux 目标通信框架 (TCF)、GDB 和 Valgrind	需使用 XRT 或 Linux 进行交互	• 软件性能（如 PS 基准测试） • 硬件性能	
嵌入式 AI 引擎系统	• JTAG • QSPI/OSPI • SD/eMMC	使用如下工具： • IBERT • AMD Vitis™ 嵌入式软件工具 • Vitis AI 引擎调试器 • Vitis 分析器 • 系统监控器	交互需要 aiecompiler、XRT、Linux 和 AI 引擎驱动程序	• 含 AI 核的软件性能（每秒图像数） • 用于无线设计的每秒兆次采样率 (MSPS) • AXI Performance Monitor (APM) 核	

注释：如需了解有关子系统功能确认的更多信息，请访问此链接以参阅《Vivado Design Suite 用户指南：编程和调试》(UG908) 中的相应内容。

仅限硬件的系统确认规划

仅限硬件的系统设计通常包含处理器核、硬化的 IP 和可编程逻辑 (PL)。您可使用以下系统级别调试工具来对系统中存在的系统级别问题进行诊断：

- 赛灵思系统调试器 (XSDB)

- AXI Streaming Integrated Logic Analyzer (AXIS-ILA)
- AXI Streaming Virtual Input/Output (AXIS-VIO)
- Integrated Bit Error Ratio Tester (IBERT)

系统级别设计调试的第一步是使用 XSDB 工具读取硬件状态，然后使用下表中所列的工具查找问题根源。

注释：如需了解有关 XSDB 命令的更多信息，请参阅《Vitis 嵌入式软件开发流程文档》(UG1400) 中的软件命令行工具。

表 6：**仅限硬件的系统调试工具**

硬件	调试工具
基于处理器的系统级别调试	· XSDB
PL 逻辑系统调试	· AXIS-ILA · AXIS-VIO
PL 逻辑调试	· IBERT · AXIS-ILA · AXIS-VIO
硬化的 IP	· IBERT · AXIS-ILA · AXIS-VIO
GT I/O 硬核块调试	· IBERT

嵌入式系统确认规划

嵌入式系统设计需要软硬件协同设计，并以嵌入式处理器核、专用硬件引擎和可编程逻辑为目标。嵌入式系统确认需要使用以下 AMD 工具和第三方工具执行系统级别调试：

- Vitis 嵌入式软件工具
- Vitis AI 引擎调试器
- GNU 调试器 (GDB)
- Arm® Development Studio (DS-5) 调试器

下表显示了上述每项工具的用途。

表 7：**嵌入式系统确认规划调试工具**

调试工具	用途
Vitis 嵌入式软件工具	· 管理源代码 · 针对特定平台编译源代码 · 将源代码下载到硬件平台中 · 启用调试

表 7：**嵌入式系统确认规划调试工具** (续)

调试工具	用途
Vitis AI 引擎调试器	· 在代码中插入用户指定的断点 · 启用单步步进 · 生成代码的反汇编视图 · 基于所使用的处理器架构和操作系统提供存储器转储
GNU 调试器	· 调试分段故障 · 调试代码中的内存泄漏
Arm DS-5 调试器	· 调试 Arm Cortex®-A72 和 Cortex-R5F 处理器上运行的代码 · 如果由于代码库过大导致难以使用单步步进或者难以设置断点，则回溯源代码

调试加速器时，请注意：

- 如果嵌入式系统设计包含基于 PL 或基于 AI 引擎的硬件加速器块，则请使用 AXI4 存储器映射接口或 AXI4 串流接口来设计加速器。大部分 Versal 自适应 SoC IP 都兼容这些协议。此外，相比于添加具有原生非 AXI 接口的 ILA，更简单的做法是使用 Vitis 工具流程添加额外的 Integrated Logic Analyzer (ILA) IP 用于调试。
- 要调试 PL 加速器块，必须在设计内包含 ILA 核。
- 如果加速器设计流程使用 Vitis 编译器将加速器块链接到 Vitis 平台，则必须在 `v++ -l` 阶段输入额外的命令行实参，以便将 ILA 核添加到内核接口中。
- 要测量和调试加速器性能，可向平台添加 AXI Performance Monitor (APM) 核。APM 核会为 AXI4 串流接口和存储器映射接口生成接口追踪，以显示内核活动时间、空闲时间和停滞时间。根据导致停滞时间的块，可聚焦此块执行额外的调试。

嵌入式 AI 引擎系统确认规划

包含 AI 引擎作为加速器的嵌入式设计均包含可执行特定算法的数据流 graph 供 AI 引擎使用。此数据流 graph 是使用 aiecompiler 编译的，输出二进制文件则在硬件上运行。此数据流 graph 可使用以下任一方法进行验证：

- aiesimulator：使用激励生成器和检查器。
- 硬件仿真流程：包含其他系统级别 IP，例如，CIPS、NoC、DDR 存储器和 RTL IP。

验证后，内核应可在硬件上正常工作，而无需对硬件进行进一步调试。但硬件与仿真环境之间可能存在功能和性能差异，原因主要是硬件中存在时序差异或者仿真模型与硬件行为之间存在差异，例如，AI 引擎仿真器与 AI 引擎硬件之间并非周期精确关系。

您可使用自适应数据流 (ADF) graph 生成 PC 事件追踪或执行追踪，以启用事件追踪剖析。您可在 Vitis 分析器中查看事件追踪数据，检查内核运行时是否存在特定的停滞。您也可以使用事件追踪检查是否因传入或传出串流发生停滞而导致出现特定的串流切换网络性能降级。

附录 A

附加资源与法律声明

查找其他文档

文档门户

AMD 自适应计算文档门户是旨在使用您的网页浏览器提供健全的文档搜索和导航的在线工具。要访问文档门户，请转至 https://docs.xilinx.com。

注释：单击链接将打开英语版本，但您可从下拉列表中选择简体中文版本（如可用）。请注意，简体中文版本可能比英语版本旧。

Documentation Navigator

Documentation Navigator (DocNav) 是预安装的工具，支持访问 AMD 自适应计算文档、视频和支持资源，您可在其中通过筛选和搜索来查找信息。要打开 DocNav，请执行以下操作：

- 在 AMD Vivado™ IDE 中，单击“Help” → “Documentation and Tutorials”。
- 在 Windows 上，单击“Start”（开始）按钮并选中“Xilinx Design Tools” → “DocNav”。
- 在 Linux 命令提示中输入 `docnav`。

注释：如需了解有关 DocNav 的更多信息，请参阅《Documentation Navigator 用户指南》(UG968)。

注释：您无法从 DocNav 访问简体中文版本。请使用设计中心网页。

设计中心 (Design Hub)

AMD 设计中心提供了根据设计任务和其他主题整理的文档链接，可供您用于了解关键概念以及常见问题解答。要访问设计中心，请执行以下操作：

- 在 DocNav 中，单击“Design Hubs View”选项卡。
- 转至设计中心网页。

支持资源

如需获取答复记录、技术文档、下载以及论坛等支持资源，请访问技术支持。

参考资料

以下技术文档是非常实用的补充资料，可配合本指南一起使用：

1. 《Versal 自适应 SoC 系统监控器架构手册》(AM006)
2. 《Versal 自适应 SoC AI 引擎架构手册》(AM009)
3. 《Versal 自适应 SoC 技术参考手册》(AM011)
4. 《Versal 自适应 SoC AIE-ML 架构手册》(AM020)
5. 《Versal 架构和产品数据手册概述》(DS950)
6. 《Versal Adaptive SoC Programmable Network on Chip and Integrated Memory Controller LogiCORE IP 产品指南》(PG313)
7. 《Control, Interface and Processing System LogiCORE IP 产品指南》(PG352)
8. 《Versal 自适应 SoC PCB 设计用户指南》(UG863)
9. 《Vivado Design Suite 用户指南：编程和调试》(UG908)
10. 《MicroBlaze 处理器参考指南》(UG984)
11. 《AI 引擎工具和流程用户指南》(UG1076)
12. 《AI 引擎内核与计算图编程指南》(UG1079)
13. 《Vitis 嵌入式软件开发流程文档》(UG1400) 中的软件命令行工具
14. 《Versal 自适应 SoC 设计指南》(UG1273)
15. 《Versal 自适应 SoC 硬件、IP 和平台开发方法指南》(UG1387)
16. 《Versal 自适应 SoC 系统集成和确认方法指南》(UG1388)
17. 《Vitis 统一软件平台文档》(UG1416)
18. 《XRT 版本说明》(UG1451)
19. 《Versal 自适应 SoC 开发板系统设计方法指南》(UG1506)
20. 《Versal 自适应 SoC 安全手册》(UG1508)
21. 《SmartLynq+ 模块用户指南》(UG1514)
22. 《电源设计管理器用户指南》(UG1556)
23. 《AI 引擎机器学习内核与计算图编程指南》(UG1603)
24. 《利用温度漂移扩展热处理解决方案》(WP517)
25. 《使用 Xilinx Power Estimator 实现准确的最差情况功耗分析七步法》(XAPP1348)
26. 《非对称硬件信任根安全启动》(XAPP1357)
27. 《对称硬件信任根安全启动》(XAPP1358)
28. 《Versal 自适应 SoC 使用 PUF 实现外部安全存储》(XAPP1359)
29. 《启动启用 eFUSE 的故障缓解功能》(XAPP1367)
30. 《Versal 自适应 SoC 经身份验证的 JTAG》(XAPP1369)
31. 《Versal 自适应 SoC NoC 中的隔离方法》(XAPP1371)

32. 《Versal AI Core 系列产品选型指南》(XMP452)
33. 《Versal Prime 系列产品选型指南》(XMP453)
34. 《Versal Premium 系列产品选型指南》(XMP463)
35. 《Versal AI Edge 系列产品选型指南》(XMP464)
36. 《Versal HBM 系列产品选型指南》(XMP465)
37. 《Versal 自适应 SoC 板级原理图审查检查表》(XTP546)

修订历史

下表列出了本文档的修订历史。

章节	修订综述
2023 年 11 月 15 日 2023.2 版	
第 2 章：系统设计类型	在表格中添加 AIE-ML 链接。
AI 引擎吞吐量和时延	添加 UG1079 和 UG1603。
2023 年 5 月 24 日 2023.1 版	
文档标题	标题更改为《Versal 自适应 SoC 系统和解决方案规划方法指南》(UG1504)。
第 2 章：系统设计类型	添加 HBM 和 AI Edge。更新基于平台的设计流程。
不含 AI 引擎的 Versal 器件的系统设计类型	添加 HBM 系列产品选型指南。
仅限硬件的系统	添加 NoC 描述。
嵌入式系统	添加 XRT 注释描述。
含 AI 引擎的 Versal 器件的系统设计类型	添加 AI Edge 系列产品选型指南。
嵌入式 AI 引擎系统	更新嵌入式描述。
应用映射和设计分区	添加 HBM 和 AI Edge。
系统计算	添加 AIE-ML 描述。
存储器和数据移动	添加 AIE-ML 描述。
AI 引擎吞吐量和时延	添加 AM020。
HBM 存储器使用情况	添加 HBM。
第 5 章：系统调试规划	添加启动注释。

请阅读：重要法律声明

本文档所示信息仅做参考，其中可能包含不准确的技术信息、疏漏和印刷错误。受诸多原因影响，此处所含信息可能发生更改，也可能无法准确呈现，这些原因包括但不限于产品和路线图变更、组件和主板版本更改、新增模型和/或产品发布、不同制造商之间存在的产品差异、软件更改、BIOS 刷新、固件升级等。任何计算机系统均存在安全性漏洞风险，无法彻底阻止也无法缓解这类风险。AMD 没有任何义务来更新或者以任何其他方式纠正或修改这些信息。但 AMD 保留随时修改这些信息和更改文档内容的权利，AMD 没有任何义务将此类修改或更改通知任何人。此处信息“按原样”提供。AMD 对于本文档内容不作任何陈述或保证，并且对于这些信息中可能出现的任何不准确、错误或疏漏问题不承担任何责任。对于有关任何暗含的非侵权、适销性及适合特定用途的保证，AMD 特此声明不承担任何责任。无论在任何情况下，对于任何人因使用此处包含的任何信息而形成的依赖或者引发的任何直接、间接、特殊或其他后果性损害，AMD 概不负责，即使 AMD 已明确获悉存在发生此类损害的可能性也是如此。

关于与汽车相关用途的免责声明

如将汽车产品（部件编号中含“XA”字样）用于部署安全气囊或用于影响车辆控制的应用（“安全应用”），除非有符合 ISO 26262 汽车安全标准的安全概念或冗余特性（“安全设计”），否则不在质保范围内。客户应在使用或分销任何包含产品的系统之前为了安全的目的全面地测试此类系统。在未采用安全设计的条件下将产品用于安全应用的所有风险，由客户自行承担，并且仅在适用的法律法规对产品责任另有规定的情况下，适用该等法律法规的规定。

版权声明