Xcell journal

软件刊

为全可编程世界量身打造的
软件解决方案

03 期
2016 年，夏季刊

在 Zynq MPSoC 上运行 Doom 游戏

利用安富利 PicoZED SDR 的自动工作流程加速 SDR 开发

Linux/RTOS AMP 让嵌入式两全其美

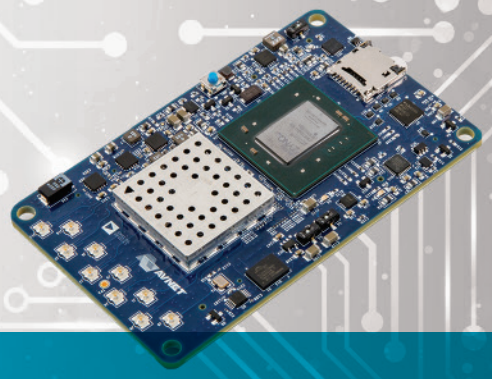
充分利用嵌入式社区积累的丰富知识构建更出色的系统

基于 FPGA 的基础架构将神经可塑性运用到云计算

XILINX
ALL PROGRAMMABLE™

china.xilinx.com/xcell

PicoZed™ SDR Z7035/AD9361



电子元件　　科技

picoZed™ SDR software defined radio

DESIGNED BY AVNET

安富利的 PicoZed™ 软件无线电 (SDR) 套件 – PicoZed™ SDR Z7035/AD9361 – 将 Analog Devices AD9361 integrated RF Agile Transceiver™ 器件与赛灵思 Z7035 Zynq®-7000 All Programmable SoC 完美地结合在一起。小型手持式 PicoZed SDR 可提供 70 MHz – 6.0 GHz 范围内的频率捷变、宽带 2x2 接收和发送，是固定和移动 SDR 应用的理想选择。PicoZed SDR 是全面验证了的系统级模块 (SOM)，无缝集成了关键 RF 信号链和高速可编程逻辑器件，构建了无线通信系统的 RF – 基带信号处理平台，从而让您能够集中精力实现产品差异化。 PicoZed SDR 不仅提供用于快速原型设计的载卡，而且还得到与赛灵思 Vivado® 设计套件无缝集成的稳健可靠的仿真和代码生成工具的强大支持，因此能显著缩短软件无线电产品的设计周期。

特性：

- 可用于最终产品开发的全面验证的低功耗耐用型系统级模块 (SOM)
- 可获得 MATLAB® 和 Simulink® 的支持，可实现数据流传输，可用于 Zynq 设计
- **数字处理部分**
  - 赛灵思 Zynq XC7Z035-2L FBG676I AP SoC
  - 1GB DDR3L SDRAM
  - 256Mb QSPI 闪存
  - microSD 接口
  - 10/100/1000 以太网 PHY
  - USB 2.0 OTG ULPI PHY
  - 超过 205 个 用户 I/O + 4 个 GTX 通道
- **无线电收发器部分**
  - Analog Devices AD9361-BBCZ Integrated RF Agile Transceiver ™
  - 带有内置 12 位 DAC 与 ADC 的 RF 2 × 2 收发器
  - 频带：70 MHz-6.0 GHz
  - TDD 和 FDD
  - 可调通道带宽：<200 kHz - 56 MHz
  - 支持 MIMO 无线电：ADC 和 DAC 同步差异小于 1 个样点
- 小型 RF 连接器 – 4 个发送器、4 个接收器、2 个发送监控器
- 利用良好电源失效保护电路实现电压调节和排序
- 通过 4 个微型插头连接器连接电源和信号
- 时钟和频率综合电路
- 包括原理图、材料清单、HDL、Linux 驱动程序和应用软件

套件包括：

- PicoZed SDR Z7035/AD9361 SOM（针对量产）

目标应用：

- 便携式捷变无线通信
- P25 公共安全无线电
- 点对点通信
- 家庭基站和微微基站
- 便携式仪器

器件：

器件型号	描述	零售价
AES-Z7PZ-SDR2-G	PicoZed SDR Z7035/AD9361 SOM	$1095.00 USD

相关器件：

器件型号	描述	零售价
AES-Z7PZ-SDR2-DEV-G	PicoZed SDR AD9361 Development Kit	$1799.00 USD

联系我们

北京：010 8414 8118　西安：029 8835 6518　南京：025 8483 8138　青岛：532 8097 0718　福州：591 8771 0381
重庆：023 6879 7512　武汉：027 8732 2806　广州：020 2808 7351　杭州：571 8580 0912　香港：852 2212 7848
上海：021 3367 8387　沈阳：189 0400 8421　成都：028 8652 8262　深圳：755 2658 4925　苏州：512 6956 7753

Xcell Journal
软 件 刊

发行人 Mike Santarini
mike.santarini@xilinx.com
1-408-626-5981

编辑 Diana Scheben

艺术总监 Scott Blair

设计/制作 Teie, Gelwicks & Assoc.
1-408-842-2627

广告销售 Judy Gelwicks
1-408-842-2627
xcelladsales@aol.com

国际 Melissa Zhang,
Asia Pacific
melissa.zhang@xilinx.com

Christelle Moraga,
Europe/Middle East/Africa
christelle.moraga@xilinx.com

Tomoko Suto,
Japan
tomoko@xilinx.com

过往期刊订购 1-408-842-2627

编辑顾问 Tomas Evensen
Lawrence Getman
Mark Jensen

Xilinx, Inc.
2100 Logic Drive
San Jose, CA 95124-3400
电话：408-559-7778
传真：408-879-4780
www.xilinx.com/xcell/

发行人致信

Xilinx 开发环境生态系统为您提供更好的选择

赛灵思过去 30 多年来一直为硬件设计人员提供基于 FPGA 的开发工具。近年来，公司打造了 SDx™ 系列工具，包括 SDSoC™、SDAccel™ 和 SDNet™ 开发环境。这些工具主要面向不熟悉 Verilog 和 VHDL 等硬件描述语言的开发人员，帮助他们用 C/C++ 和 OpenCL™ 等软件语言对赛灵思器件进行编程。赛灵思不是此类工具的唯一供应商； 提供能够支持赛灵思器件的流行设计环境的知名第三方供应商包括 MathWorks®（MATLAB™ 和 Simulink™）、国家仪器（LabVIEW 和 LabVIEW FPGA）以及 Topic Embedded Products（Dyplo）。

我冬天在德国纽伦堡的嵌入式世界大会上观看了 Topic Embedded Systems 公司的 Dyplo 与赛灵思 Zynq®-7000 SoC 配合的视频，这段视频给我留下深刻印象。Dyplo 代表动态流程加载程序 (Dynamic Process Loader)，能让程序用 SoC 的部分重配置功能在 Zynq-7000 SoC 的可编程逻辑中切换使用不同的定制编译硬件加速器。我在“Xilinx 午后加油站”博客中就指出：“这真是一种很棒的技术。”（见：“Topic Embedded 的 Dyplo 框架将 Zynq-7000 SoC 变为多任务软硬件执行引擎”）

如您认为自己早就知道赛灵思器件怎么开发代码了，我建议您真的应该尝试摆脱固定思维的束缚，欢迎了解 SDSoC 以及其它赛灵思 SDx 开发环境的发展，尤其要了解 MathWorks、国家仪器和 Topic Embedded Products 提供的产品。

致敬和道别 Mike Santarini

《Xcell 软件刊》和赛灵思对曾任本杂志发行人八年的 Mike Santarini 深怀感激之情。Mike 已经改任其他工作，但他的贡献具有深远的意义。我们赛灵思的全体人员对您，Mike 衷心道一声“谢谢您！”祝您在未来的前程中一路顺风！

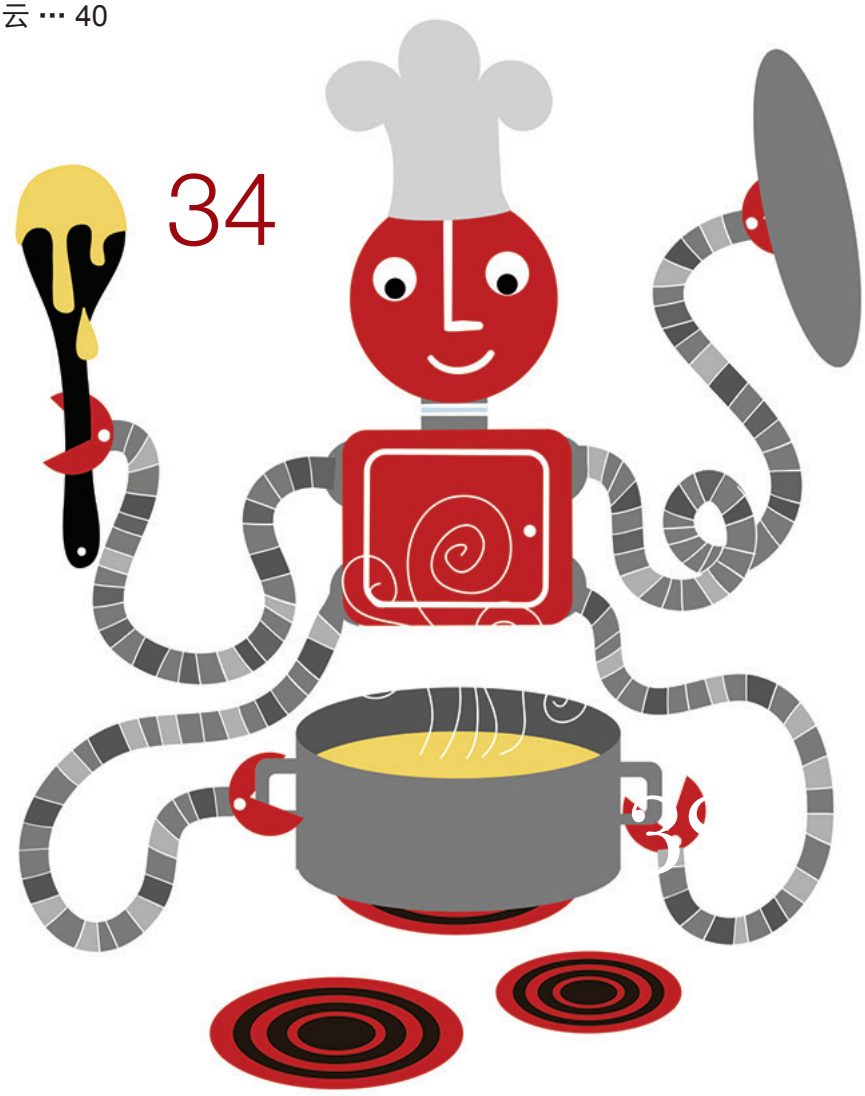
目录

2016 年
第二季度
第三期

视点

封面专题

6



在 Zynq MPSoC 上运行 Doom 游戏

通过这篇有趣的教程，熟悉运行在赛灵思 Zynq UltraScale+ MPSoC 上的 Xen 管理程序。

作者：Zach Pfeffer
赛灵思公司嵌入式软件开发总监
zachp@xilinx.com

Edgar Iglesias
赛灵思公司高级主任软件工程师
edgari@xilinx.com

Alistair Francis
赛灵思公司软件工程师
alistai@xilinx.com

Nathalie Chan King Choy
赛灵思公司主任软件工程师
nathalie@xilinx.com

Rob Armstrong Jr.
赛灵思公司嵌入式专家现场应用工程师
ra@xilinx.com

W

赛灵思和 DornerWorks 的系统软件团队在赛灵思的 Zynq® Ultrascale+™ MPSoC 上启动 Xen Project 管理程序时，我们发现可通过运行 1993 年的流行电子游戏 Doom 来演示和测试系统。这款射击游戏使团队可以探讨 Xen 工程设计话题，以求将知识和经验传授给以后的管理程序使用者。

我们团队使用适用于 QEMU（开源 Quick Emulator）的 Zynq UltraScale+ MPSoC 仿真模型为软件进行 Doom 演示做好准备，当芯片到达时，只需数小时就能启动运行，不再需要等上数天。

如何针对 Zynq UltraScale+ MPSoC 通过 QEMU 在 Xen 上运行 Doom 呢，在详细介绍具体步骤之前，我们先来了解什么是管理程序，以及它们如何与 Zynq UltraScale+ MPSoC 上的处理器协同工作。

管理程序及其工作原理

管理程序是一种可虚拟化处理器的计算机程序。运行在虚拟化处理器上的应用程序和操作系统似乎完全拥有系统，但事实上管理程序负责管理

虚拟处理器对物理机资源（例如存储器和处理内核）的访问。管理程序之所以流行，是因为能实现设计分区以及系统上运行的独立软件元素之间的隔离。

为了支持虚拟化，物理处理器必须提供一个供管理程序运行的特殊“模式”。因此，介绍处理器模式有助于理解管理程序如何完成处理器魔法。

所有处理器都有一些指令，这些指令可操作寄存器中存储的值，并可读写存储器。处理器的模式是指令和寄存器的集合，以及利用指令访问寄存器和存储器时要遵守的规则。为了便于解释，我们以通用处理器为例来介绍，并使用与结构无关的术语。在这个实例中，处理器具有特定的寄存器、指令和模式。寄存器包括 RegisterA、RegisterB、RegisterC、UserProgramCounter、Register-Super 和 SuperProgramCounter。指令包括以下内容。

ADD Register3 Register1 Register2 将 Register1 与 Register2 相加，并把结果存入 Register3，即 Register3 = Register1 + Register2。

MOVTO Register2 Register1 将 Register1 中地址所指向的存储器内容移动到 Register2。

MOVFROM Register2 Register1 将 Register1 的内容移动到 Register2 中地址所指向的存储器。

ENTERSUPER 进入处理器的 SUPER 模式。

EXITSUPER 退出 SUPER 模式并进入 USER 模式。

在 USER 模式下，处理器的指令的功能受到限制。本例中，指令可对除 RegisterSuper 和 SuperProgramCounter 以外的所有寄存器进行读和写操作，处理器可执行除 EXITSUPER 以外的所有指令。

此外，在 USER 模式下，所有指令只能读和写一部分存储器，例如从地址 0x0000_0100 到 0x0FFF_FFFF。在 USER 模式下，如果程序尝试执行不应该执行的指令，或者访问无权访问的寄存器或存储器位置，那么处理器将暂停出错指令 (offending instruction)。

SUPER 模式下，处理器的指令可以读 / 写上述所有寄存器，包括 RegisterSuper 和 SuperProgramCounter。以上所列的所有指令，包括 EXITSUPER，都可以执行，另外，附加的指令 ENTERHYPER 也可执行（后面详细介绍该指令）。此外，在 SUPER 模式下，指令可以访问系统中的全部存储器（从 0x0000_0000 到 0x7FFF_ FFFF）。

采用带模式的处理器，使我们可以利用设计分区来更简单地解决软件工程设计问题。以上实例中，只有一种方法进入 SUPER 模式：执行 ENTERSUPER 指令。同样，只有一种方法退出 SUPER 模式： 执行 EXITSUPER。此外，在 USER 模式下程序只能访问机器的部分存储器。有了这种方案，我们可编写一个程序让处理器同时运行多个 USER 模式程序。这个“操作系统”(OS) 程序运行在 SUPER 模式，并管理在 USER 模式中运行的程序。

当 OS 运行时，会查看需要运行的所有 USER 模式程序，选择一个运行，然后使用 EXITSUPER 这样的指令通知处理器切换到 USER 模式以运行程序。所选的程序会一直运行，直到有事件导致处理器切回 SUPER 模式。这类事件可以是来自 USER 模式程序的 ENTERSUPER 指令，或外部事件，例如定时器，它可以不提醒正在 USER 模式下运行的程序将处理器切换到 SUPER 模式。无论切换如何发生，每当事件发生时，我们都可构建 OS 以根据相应策略相继选择和运行 USER 程序。当切换快速进行时，用户认为 USER 程序同时运行。

HYPER 模式的用处是让很多 SUPER 程序运行。SUPER 模式下的每个程序都可以是 OS；这些 OS 本身会让很多 USER 程序并行运行。

SUPER 处理器模式还能防止 USER 程序干扰运行在 SUPER 模式的程序或其他 USER 模式程序。USER 模式程序的任何错误或违规都可被控制在该程序自身的实例中，不会破坏或干扰为 SUPER 模式操作保留的系统存储器和寄存器。

听起来很好，但能否用另一个模式实现一些功能？

对我们的机器稍加扩展，就可以引入 HYPER 模式。HYPER 模式可以读 / 写所有初始寄存器（RegisterA、RegisterB、RegisterC、UserProgramCounter、RegisterSuper 和 SuperProgramCounter）以及两个附加寄存器：RegisterHyper 和 HyperProgramCounter。

HYPER 模式下的指令包括初始集以及下面的斜体字。

ADD Register3 Register1 Register2 将 Register1 与 Register2 相加并把结果放在 Register3 中，即 Register3 = Register1 + Register2。

MOVTO Register2 Register1 将 Register1 中地址所指向的存储器内容移到 Register2。

MOVFROM Register2 Register1 将 Register1 的内容移到 Register2 中地址所指向的存储器。

MOVTOPHYS Register2 Register1 将 Register1 中物理地址指向的存储器内容移到 Register2。

MOVFROMPHYS Register2 Register1 将 Register1 的内容移到 Register2 中地址指向的物理存储器。

ENTERSUPER 进入处理器的 SUPER 模式。

EXITSUPER 退出 SUPER 模式并进入 USER 模式。

ENTERHYPER 进入处理器的 HYPER 模式。

EXITHYPER 退出处理器的 HYPER 模式。

SWITCHSUPER RegisterHyper 切换到 SUPER 程序，该程序将使用 RegisterHyper 中的值来执行下一个 SUPER 程序。

HYPER 模式中的附加指令和寄存器允许处理器切换哪个程序在 SUPER 模式中运行，就像 SUPER 模式允许处理器切换哪个程序在 USER 模式中运行一样。HYPER 模式的一个特性是能够切换哪个存储器 SUPER 模式能看到；当一个在 HYPER 模式中运行的程序执行 SWITCHSUPER RegisterHyper 时，底层存储器完全断开。这就是说当 HYPER 模式中的程序执行了 EXITHYPER 之后，下个 SUPER 程序运行之时，SUPER 模式看到的实际物理存储器与运行在 SUPER 模式中的另一个程序使用的物理存储器不同。SUPER 模式程序仍使用相同地址访问存储器，但是该地址指向不同的物理位置。图 1 显示了执行 SWITCHSUPER RegisterHyper 前后的处理器存储器视图。

HYPER 模式很有用，是因为它允许很多个 SUPER 程序运行。SUPER 模式中每个程序都可以是 OS； 这些 OS 本身可以让很多 USER 程序并列运行 . 这意味着，我们可以在相同硬件上运行多个 OS，例如 Windows 和 Linux；在一个处理器上运行 20 个 Linux 实例；或者之间的任意组合。由于每个虚拟 OS 实例无法看到另一个 OS 实例，因此如果一个崩溃，不会使另一个实例也崩溃。HYPER 模式的特性还有其他应用： 我们可以在多个 OS 之间对系统资源分区；监测 HYPER 模式下每个 OS 的执行，以在崩溃时重启； 以及在虚拟 OS 运行时密切关注系统状态。

随着处理器从 USER 切换到 SUPER 模式，再从

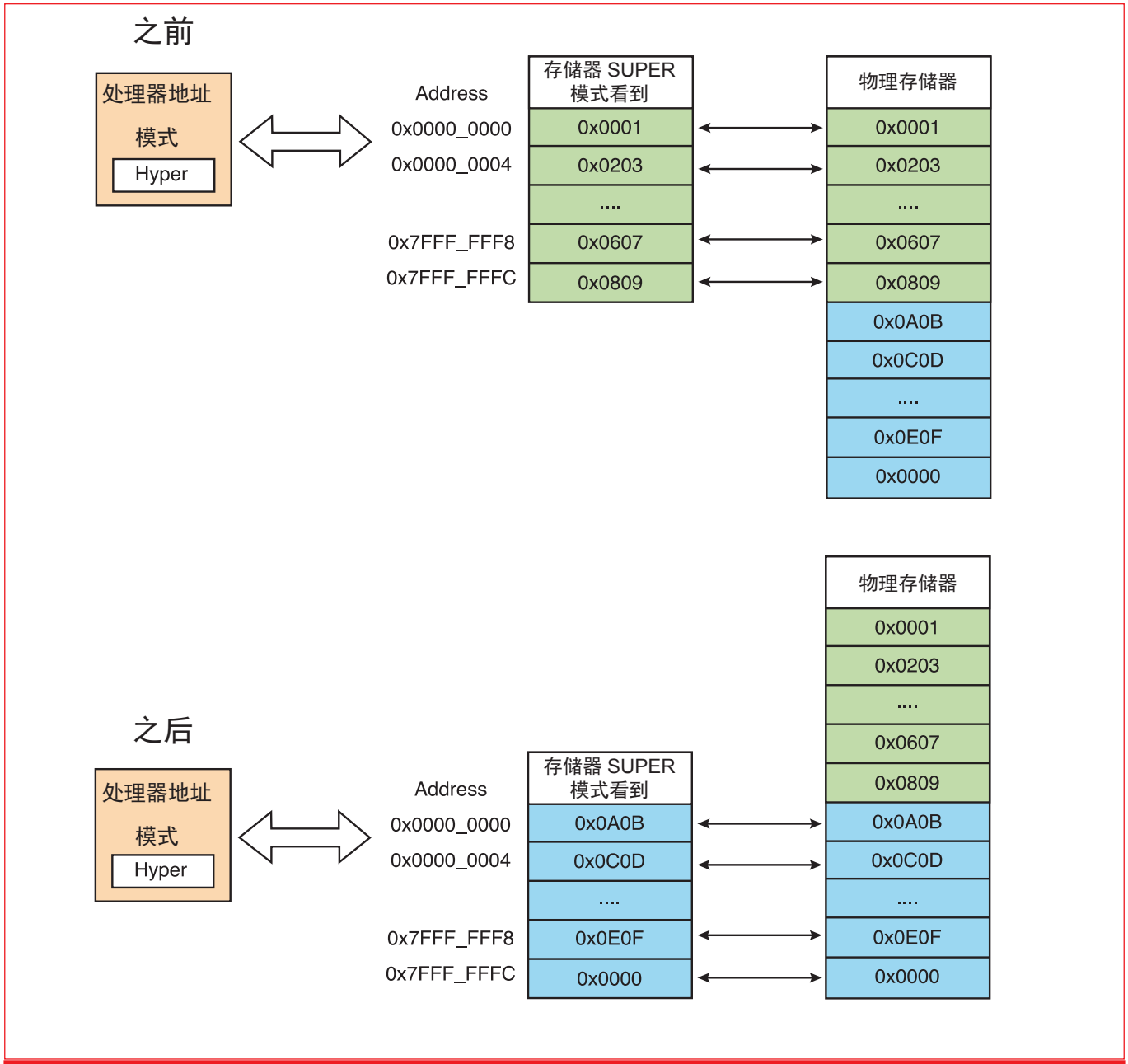


图 1 — HYPER 模式下执行 SWITCHSUPER RegisterHyper 的前后区别

SUPER 切换到 HYPER 模式，机器会赋予执行代码更多特权。本例中，USER 模式程序只有权使用四个寄存器（RegisterA、RegisterB、RegisterC 和 UserProgramCounter）和四个指令：（ADD、MOVTO、MOVFROM 和 ENTER- SUPER）。此外，USER 程序只能读写 0x0000_0100 至 0x0FFF_ FFFF 的存储器。一旦进入 SUPER 模式，处理器允许指令与 RegisterSuper 和 SuperProgramCounter 对话，并允许执行 EXITSUPER 和 ENTERHYPER。此外，SUPER 程序可以访问从 0x0000_0000 至 0x7FFF_FFFF 的存储器。

最后，一旦处理器进入 HYPER 模式，其指令就可以操作 RegisterHyper 和 HyperProgramCounter，而且程序可执行 SWITCH-SUPER 和 EXITHYPER。

HYPER 模式还允许处理器读写所有虚拟存储

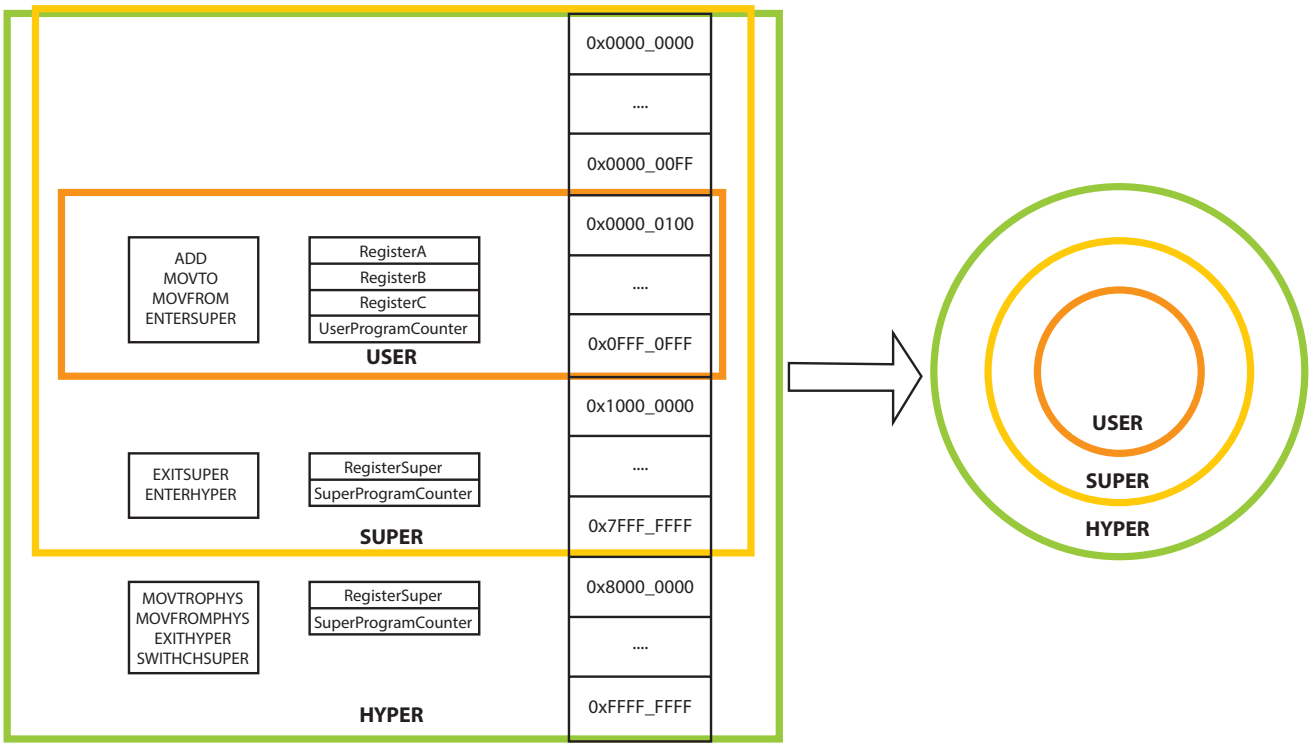


图 2 — 各种模式如环形所示

器，0x0000_0000 至 0xFFFF_FFFF，以及读写实际物理存储器。这些特权等级通常被直观地用环形来描述（图 2）。主环，即 HYPER 环为特权等级较低的环赋予权限，最终可控制整个系统。

理论结合实践

ARM® 创建处理器设计，供 ARM 合作伙伴构建芯片用。ARM 处理器包含一个或多个内核。每个内核实现一个 ARM 架构。例如，Zynq UltraScale+ MPSoC 包含一个 ARM Cortex™-A53 处理器及四个 ARMv8-A 物理内核（图 3）。

当查看 ARM 处理器的文档和代码时，这种区别很重要；为了全面理解具有一个 ARM 内核的“芯片”，可参考有关架构（如 ARMv8-A）和处理器（如 Cortex-A53）的文档。

ARMv8 架构中有四个例外等级（来源：*ARM 架构参考手册*，D1-1404）：

1. 例外等级 0 (EL0)，无需特权即可执行；

2. 例外等级 1 (EL1)，执行 OS 以及任何执行特权指令的内容；

3. 例外等级 2 (EL2)，允许硬件被虚拟化；以及

4. 例外等级 3 (EL3)，允许在安全与非安全处理器状态之间切换。

以下程序通常在这些模式下运行，如 *ARM 架构参考手册* (D1–1404) 中所述：EL0，应用程序；EL1，OS 内核以及通常所描述的相关特权函数；EL2，管理程序；EL3，安全监控器。我们的理论实例直接对应 ARMv8 执行模式 EL0 至 EL2：USER 对应 EL0，SUPER 对应 EL1，HYPER 对应 EL2。ARM 添加第四个特权等级，即 EL3； 利用这个特

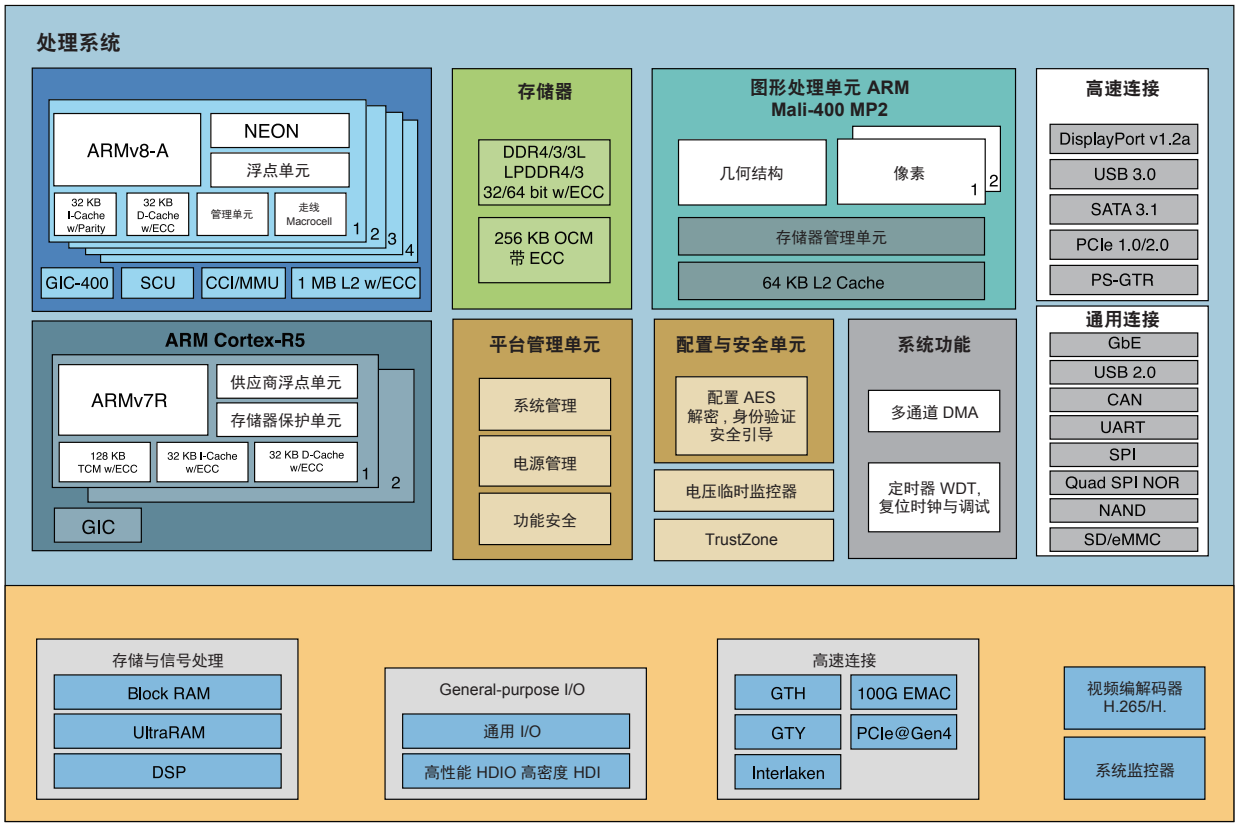


图 3 — Zynq UltraScale+ MPSoC

权等级，我们可在安全与非安全环境之间切换 EL0 和 EL1。尽管 EL3 的使用是一个很重要的论题，能够为架构增加大量的功能，但是在本实例中我们将其忽略，并着重介绍 EL0-EL2（利用管理程序的虚拟化）。如果对计算机如何保护金融交易感兴趣，可以参阅 ARMv8 EL3 文档（免费提供，需注册）。这是非常好的参考文档，从中可以获得极为详细的介绍。

进入和退出例外模式

在真实系统中，模式之间的切换比我们的实例更复杂一些。ARM 总结了 ARMv8-A 架构的行为并在参考手册中给出。手册中介绍，只有在接到例外或从例外返回时，才能改变执行所处的例外等级。在接到例外时，例外等级只能升高或保持不变；在从例外返回时，例外等级只能降低或保持不变。只有三个指令能生成针对下个例外等级的例外：SVC (Supervisor Call)，生成针对 EL1 的例外；HVC (Hypervisor Call)，生成针对 EL2 的例外；SMC (Secure Monitor Call)，生成针对 EL3 的例外。这些指令取值范围为 0-65,555，允许每个例外等级有 2^{16} 个系统调用。这些指令针对下个例外等级，而且是唯一可供运行在较低例外等级的程序从运行在较高例外等级的程序请求某些内容的机制。在我们的理论实例中，SVC 是 SWITCHSUPER，HVC 是 SWITCHHYPER。

在前一个部分，我们介绍了能够让运行在 USER 模式 (EL0) 的程序进入 SUPER 模式 (EL1) 的

PetaLinux 工具包含一组命令，以供用户在赛灵思 FPGA 和 SoC 上轻松创建和扩展 Linux 系统。

事件。大多数运行在 USER 模式的程序生成的事件是请求存储器。当运行在 EL0 中的用户空间程序从运行在 EL1 中的 OS 请求存储器时，这个用户空间程序的 C 代码可能调用函数 malloc()，再由该函数调用 mmap() 或 sbrk()，以从 OS 请求一个指向可用存储器的指针。在 ARMv8-A 架构中的 Linux 上，这个过程在幕后转化为 SVC 系统调用。该系统调用会把处理器转换为 EL1，从而将控制权送回 OS，后者会解读调用内容并提供正确的响应 —— 本例中是指向所请求存储器区域的指针，或者是一个错误，用以指出没有可用存储器。

演示创建和工具

现在我们来介绍我们团队在 Zynq UltraScale+ QEMU Model 上运行 Doom 时所采用的步骤。这些步骤展示了如何获得和构建运行演示所需的每个组件，如何运行以及以什么顺序运行每个组件，以及如何与演示交互。成功完成该演示之后，你会获得一个环境，用来在上面进行实验，以了解 Xen 管理程序在仿真的 Zynq UltraScale+ MPSoC 上的运行情况。还需要将此迁移植 Zynq UltraScale+ MPSoC 芯片，这可作为练习由用户来完成。

为了让过程更简单，赛灵思提供基础的根文件系统，这样用户就无需花时间和精力自己构建。此演示所需的所有下载内容在以下网址中均有提供：*www.wiki.xil- inx.com/Doom+on+Xen+Demo.*

该演示首先通过更新由赛灵思提供的预编译根文件系统 (rootFS)，可包含所需的组件。然后，利用赛灵思的 PetaLinux 工具运行演示。rootFS 包含运行于 Linux 系统上的大部分程序 —— 具体来说就是用来启动系统的一组脚本，以及用来实现系统的应用程序与函数库集。我们用来扩展演示中的基础 rootFS 所使用的两个工具分别是 Buildroot 和 PetaLinux。我们使用 Buildroot 为赛灵思提供的基础 rootFS 构建 Doom 二进制文件，同时使用 PetaLinux 创建 rootFS 的剩余部分并引导演示。

Buildroot

Buildroot 是一个简单的构建系统，用于为 Linux 系统创建 rootFS。它使用 make menuconfig 接口，这是一个用来配置 Linux 内核本身的常用方法。Buildroot 包含对 PrBoom 的默认支持，这对于本演示很有帮助。（PrBoom 是我们所使用的 Doom 游戏的 GNU 通用公共许可证 [GPL] 版本。这里我们会穿插使用 PrBoom 和 Doom 这两个术语。）Buildroot 对 Xen 构建不提供本地支持（尽管它可创建用于构建 Xen 所需的所有库和工具链），因此赛灵思提供 Xen、Xen 工具和为用户预编译的 Xen 库以及其他一些所需的库，以让过程简单直观。

PetaLinux

PetaLinux 工具包含一个命令集，以便让用户在赛灵思 FPGA 和 SoC 上轻松创建和扩展 Linux 系统。该演示使用 petalinux-build 和 petalinux-boot 命令。petalinux-build 命令用于创建全部所需的组件。petalinux-boot 命令（外加几个变量）用于启动在 QEMU 仿真器上运行的所有组件。介绍 PetaLinux 工具中的所有命令超出了本文的范围，但是通过此演示系统应该很容易发掘这两个命令和其他命令的功能。参考 *PetaLinux 工具文档 — 参考指南 UG1144 (v2015.4)* 了解更多信息。

项目先决条件

该项目需要一个运行 Linux 的工作站或虚拟机，具有满足 UG1144 (v2015.4) 中所列的 PetaLinux 工具安装要求的环境，而且环境中需要安装赛灵思 PetaLinux Tools v2015.4 版本。

步骤 1： 构建 ROOTFS

首先，我们需要构建 rootFS。从赛灵思下载 doom_demo.tar.gz，打开下载目录中的一个 terminal； 你可在以下网址中找到全部所需文件：*www.wiki.xilinx.com/Doom+on+Xen+Demo*。我们将该目录称为 <down_dir>。

解压文档。

```
$ cd <down_dir>
$ tar -xzf doom_demo.tar.gz && cd doom_demo
```

我们会看到一个文件夹，我们将把它存到根文件系统（一个用于 Dom0，另一个用于 DomU）。现在，我们需要构建 PrBoom，并复制到 rootFS。

首先，需要下载 Linux 内核，这样我们随后就可以构建 rootFS。我们使用 v4.3 标签。

```
$ git clone -b v4.3 https://github.com/torvalds/linux.git
```

下载 Buildroot 源文件，并更改到 Buildroot 目录。

```
$ git clone https://git.buildroot.net/buildroot
&& cd buildroot
```

现在我们需要配置 Buildroot，以构建可以使用的套件。

```
$ make menuconfig
```

我们选择以下选项：

Target options ---> Target Architecture ---> AArch64 (little endian)

Target packages ---> Games ---> prboom ---> [*]

Target packages ---> Games ---> shareware Doom WAD file ---> [*]

应自动选择全部所需的库。

$ make #（这需要几分钟时间，取决于机器。）

现在，我们将所有 PrBoom 相关文件复制到 targetfs 目录，确保我们在 buildroot 目录下的 ./output/target/ 目录。

```
$ for i in $(find ./-name '*oom*'); do cp ${i}
<down_dir>/doom_demo/targetfs/${i}; done
```

现在，我们完成了 Buildroot 操作。我们移到上一个目录 doom_demo 目录。

```
$ make # Build the host and guest rootFS.（这需要几分钟时间，取决于你的机器。）
```

注意： 可能还存在额外配置选项，这主要取决于使用的内核版本。这些额外配置选项未被我们提供的配置预先选择。使用默认选项即可（需点击回车键）。

步骤 2： 构建基础设置

接下来，我们为平台构建嵌入式系统软件的剩余部分，包括引导装载程序、ARM Trusted Firmware (ATF)、Linux 内核和设备树。赛灵思的 PetaLinux 工具让这个过程简单直观。我们创建一个针对赛灵思 ZCU102 开发板的 PetaLinux 项目。参考 2015.4 UG1144 和 AR#66249 中 QEMU 和 MPSoC PetaLinux 的快速入门材料。访问 *china.xilinx.com*，将 ZCU102 BSP（板支持包）下载到 <petalinux_bsp_dir> 目录下。

```
$ cd <down_dir>
$ petalinux-create --type project -s <petalinux_bsp_dir>/ Xilinx-ZCU102-v2015.4-final.bsp
--name doom_demo_zynqMP
```

这样将在 <down_ dir>/doom_demo_zynqMP 中创建我们的 PetaLinux 项目。

我们转到 PetaLinux 项目，并构建 PetaLinux。

```
$ cd <down_dir>/doom_demo_zynqMP
$ petalinux-build
```

现在，我们需要为本用例手动编辑设备树。

编辑 xen-overlay.dtsi 文件 (subsystems/linux/ configs/device-tree/xen-overlay.dtsi)。

将 dom0 下的

‘reg = <0x0 0x80000 0x3100000>;’

替换为

‘reg = <0x0 0x80000 0x4100000>;’

将 dom0 下的

‘xen,xen-bootargs = “console=dtuart dtuart=serial0 dom0_mem=512M bootscrub=0 maxcpus=1 timer_ slop=0” ;’

替换为

‘xen,xen-bootargs = “console=dtuart dtuart=serial0 dom0_mem=512M bootscrub=0 maxcpus=4 timer_ slop=0” ;’

将 dom0 下的

‘xen,dom0-bootargs = “console=hvc0 earlycon=xen earlyprintk=xen maxcpus=1” ;’

替换为

‘xen,dom0-bootargs = “rdinit=/bin/sh console=hvc0 earlycon=xen earlyprintk=xen maxcpus=4” ;’

编辑 zynqmp.dtsi 文件 (subsystems/linux/configs/device-tree/zynqmp.dtsi)。

将 dom0 下的

‘compatible = “cdns,uart-r1p12” ;’ 替换为

‘compatible = “cdns,uart-r1p8” , “cdns,uart-r1p12” ;’ 现在，手动构建 Xen 设备树。

```
$ dtc -I dts -O dtb -i ./subsystems/linux/configs/device-tree/ -o ./images/linux/xen.dtb ./subsystems/linux/configs/device-tree/xen.dts
```

最后，我们需要将 Peta- Linux 构建的 rootFS 替换为我们此前构建的 rootFS。之所以这样做，是因为 PetaLinux 不包含 PrBoom，因为我们提供自己的 rootFS。我们还需要将 xen.ub 镜像替换为赛灵思预先构建的镜像，因为 Xen 和 Xen 工具版本必须匹配。

```
$ rm <down_dir>/doom_demo_zynqMP/images/linux/Image && rm <down_dir>/doom_demo_zynqMP/images/linux/xen.ub
$ cp <down_dir>/doom_demo/Image <down_dir>/doom_demo_zynqMP/images/linux/Image && cp <down_dir>/doom_demo/xen.ub <down_dir>/doom_demo_zynqMP/images/linux/xen.ub
```

使用 u-boot 引导加载程序引导。

```
$ petalinux-boot --qemu --u-boot --qemuargs="-net nic -net nic -net nic -net nic -net us- er,net=192.168.129.0,dhcpstart=192.168.129.50,host=192.168.129.1,hostfwd=tcp:127.0.0.1:5900-192.168.129.50:5900"
```

> setenv serverip 192.168.129.1

> tftpb 4000000 xen.dtb; tftpb 0x80000 Image; tftpb 6000000 xen.ub; bootm 6000000 - 4000000

/boot.sh

/xen-doom.sh 1

步骤 3： 开始演示

现在，我们可以打开虚拟网络计算 (VNC) 查看器，并在运行 QEMU 的机器上连接 localhost:5900 以观看 Doom 游戏。（注意： 以上命令行只能重定向 5900 端口，因此当开始演示时只能连接到第一个 Doom 实例。如果想连接多个实例，需要为 QEMU 添加更多 hostfwd 变量，并连接到下个可用的端口 [5901 用于下个实例，5902 用于第三个实例，以此类推]，然后将这些实例连接。）

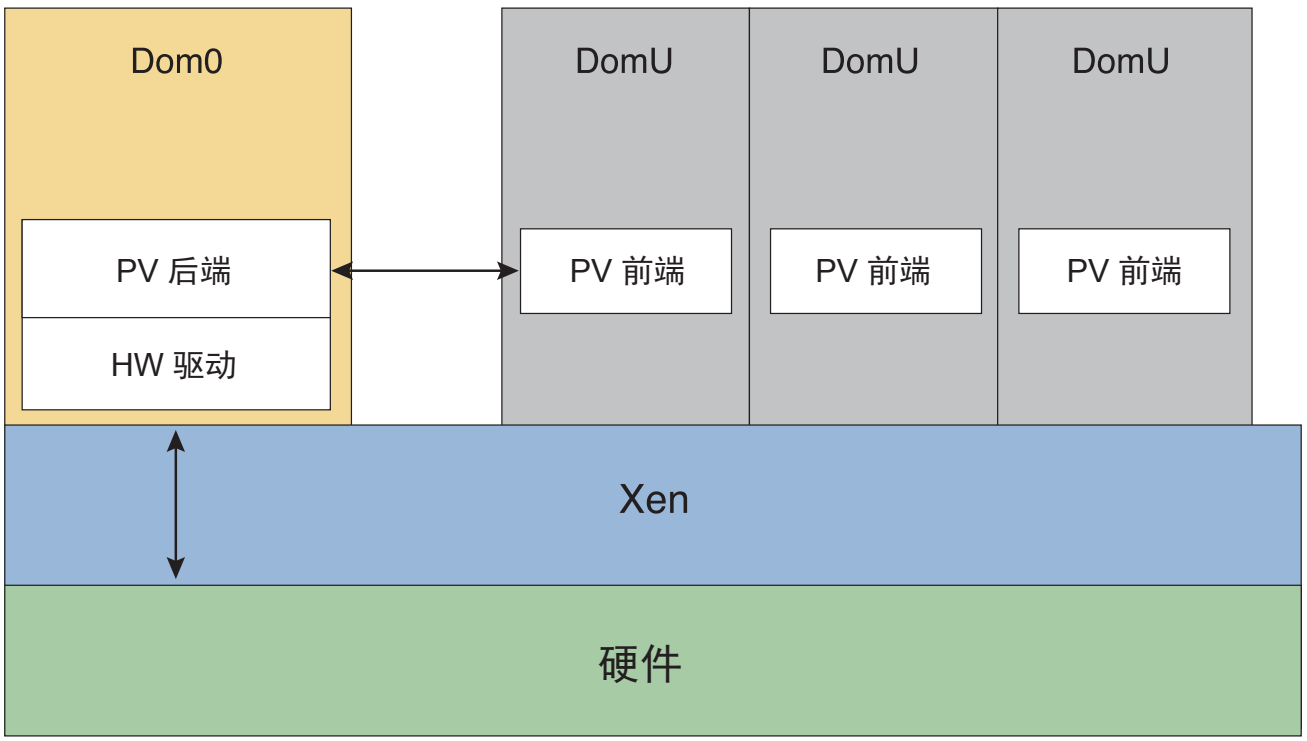


图 4 — 作为 Type 1 管理程序，Xen 在本机硬件上运行，虚拟机在 Xen 之上运行（来源：“带虚拟化扩展的 Xen ARM”白皮书）。

一旦 Doom 启动，你就可以使用键盘和鼠标控制游戏。应记住，可能需要点击 ESC 键来开始游戏。还应记住，你已经很长时间没玩 Doom 游戏了，因此你可能走不了多远。别气馁。使用自己构建的系统绝对“可行”。

XEN 深入探讨

正如“Zynq MPSoC 获得 Xen 管理程序支持”（*赛灵思中国通讯，第 58 期*）中所介绍，Type 1 管理程序在本机硬件上运行，Type 2 管理程序不是软件的最底层，而是托管在 OS 上。Xen 属于 Type 1 管理程序（图 4）。

以前，我们提到了虚拟处理器（也称虚拟机）。在 Xen 中，这些被称为域。特权最高的域被称为 Dom0；无特权的客户域是 DomU 域。

Dom0 是 Xen 管理程序在引导时创建的初始域。它是特权域，并驱动平台上的设备。Xen 将 CPU、存储器、中断和定时器虚拟化，为虚拟机提供一个或多个虚拟 CPU、系统存储器的一部分、一个虚拟中断控制器和一个虚拟定时器。除非配置为其他方式，否则 Dom0 可直接访问所有设备并驱动它们。Dom0 还运行一组名为半虚拟化 (PV) 后端的驱动，为无特权虚拟机提供对磁盘、网络等设备的访问权。Xen 提供用于发现和初始通信设置的所有工具。作为 DomU 的 OS 通过运行相应的 PV 前端驱动程序来获得对一组通用虚拟设备的访问权。根据 DomU 的数量，单个后端可服务多个前端。有一对适用于所有最常见设备类型（磁盘、网络、控制台、帧缓冲器、鼠标、键盘等）的 PV 驱动程序。PV 驱动程序通常位于 OS 内核（即 Linux）中。几个 PV 后端也可以在用户空间中运行，通常在 QEMU 中。前端在存储器的共享页上使用简单的环协议连接后端。从 Dom0 与管理程序交互要求程序使用定义的管理程序调用（类似于系统调用）。Xen 提供一个名为 Xen Tools（也可写成 xen-tools）的、带有库的参考工具箱。xen-tools 包含一个名为 xl 的

利用设备半虚拟化，可在管理程序与客户机之间就如何进行通信达成协议。常见的通信协议为 Xen Bus 和 VirtIO。

程序，该程序可与其他程序一起检查状态和创建客户机。

xl 中的“create”命令要用到描述客户机的配置文件，如果配置文件规定客户机需要一个由 VNC 会话支持的虚拟帧缓冲器 (VFB)，那么 xl 会在 Dom0 用户空间中自动启动虚拟化代码（本演示中，为每个客户机启动一个）。

doom VM 的配置文件如下所示：

客户机名称 name =“guest1”

要引导的内核镜像 kernel =“/boot/Image”

内核命令行选项

extra =“console=hvc0 rdinit=/doom.sh”

最初存储器分配 (MB) memory = 56

VCPUS 数量 vcpus = 1

vfb = [‘type=vnc, vnclisten=0.0.0.0’]

XEN 中的设备

为客户机提供设备有三种常用方法： 仿真、半虚拟化和直通（图 5）。

对于设备仿真，当客户机向仿真设备的存储器写入时，写入操作会触发陷阱。陷阱通常就是页面错误。陷阱使处理器能够切换到管理程序，以仿真设备。仿真是灵活的，但速度慢，因为要处理所有陷阱，而且要有人为所有需要仿真的设备编写模型。而且，很难找到方法来加速仿真，因为几乎没有硬件加速；完全是软件方法。

利用设备半虚拟化，可在管理程序与客户机之间就如何进行通信达成协议。通常有一个共享的存储器区域（以及协议），这看起来像一个设备，而且管理程序在该区域处理请求。例如，为了在 Linux 上支持半虚拟化帧缓冲器，Linux 前端驱动会把从用户空间获得的帧缓冲器写入共享存储器区域；然后使用管理程序调用向管理程序发信号，以通过后端驱动来输出帧。客户机只能通过半虚拟化驱动程序与主机 (Dom0) 和其他客户机 (DomU) 对话。这种方案的优势是： 用户可以在很多客户机之间共享设备； 运行快速； 客户机可以运行大部分都没修改的内核。要求的变动在标准接口下面，因此对于应用程序以及内核其余部分来说，前端驱动程序看起来就像正常的网络接口、磁盘或其他设备。支持客

	速度	设备共享	安全
仿真	慢	是	是
半虚拟化	中	是	是
直通	快	否	是（但必须有 SMMU*）

* 系统存储器管理单元

图 5 — 方案、半虚拟化和直通方案的对比

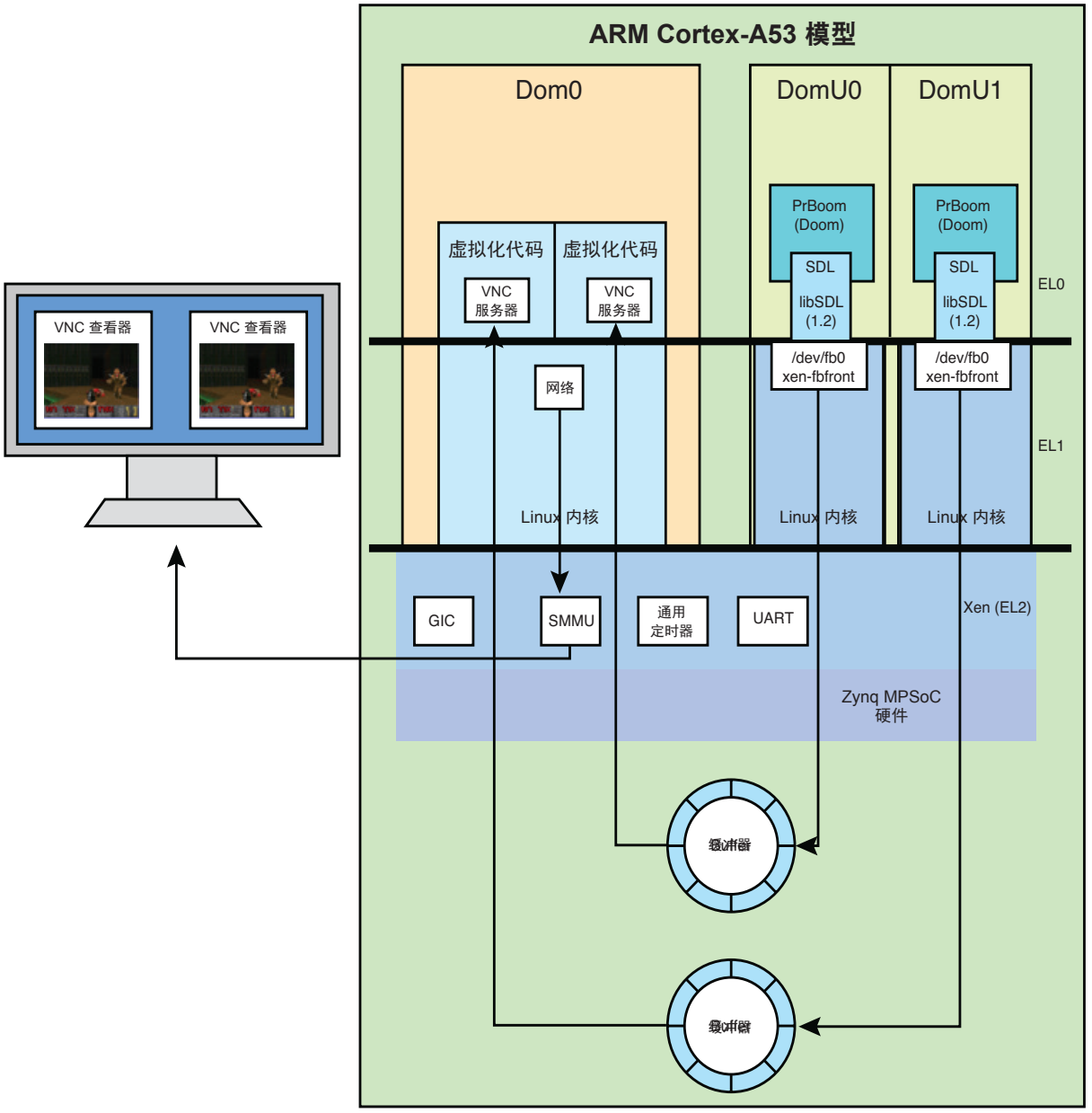


图 5 — 方案、半虚拟化和直通方案的对比

户机通信的两个常用协议是 Xen Bus 和 VirtIO。

在直通模式下，主机将设备“交给”一个客户机。这意味着每次只有一个客户机可以使用该设备。

设备性能与安全

一般来说，与通过直通方式提供的设备相比，仿真的设备性能比较低； 半虚拟化方案则趋向于具备足够性能。半虚拟化方案和仿真方案的优势在于管理程序可以让设备访问多个实体，而不会将这些实体相互暴露。

原理简介

Doom-on-Zynq Ultra- Scale+ MPSoC 的处理上下文环境就像洋葱一样有很多层（图 6）。Cortex-A53

中是四个 ARMv8 内核。在每个内核上，管理程序运行在 EL2 中，客户机（Dom0 或 DomU）运行在 EL0/EL1 中。每个 DomU 客户机都运行 Linux；Doom (PrBoom) 运行在用户空间中。Doom 使用简单直接媒体层 (SDL)，通过 SVC 指令（最终）与帧缓冲器前端驱动对话。帧缓冲器前端将缓冲器写入 Dom0 建立的共享存储器区域。前端驱动通过协议（例如 Xen Bus 或 VirtIO）使用 HVC 指令（最终）与 Dom0 上运行的虚拟化代码通信。在 Dom0 上运行的虚拟化代码提供一个用于显示的后端，然后该后端由虚拟化代码的 VNC 服务器进行编码，并通过网络送到 VNC 客户端。

此信息和演示能够为管理程序的进一步研究和实验提供很好的基础。当你能够在 QEMU 上用仿真来运行演示之后，就可使用 PetaLinux 工具在 Zynq UltraScale+ MPSoC 芯片上运行。

如需更多出色的开发人员资源，敬请访问赛灵思的软件开发人员专区。■

软件无线电从概念到部署

利用安富利 PicoZed SDR 的自动工作流程缩短开发时间并实现差异化设计。

作者：Robin Getz
Analog Devices 公司全球联盟部门工程设计总监
Robin.Getz@analog.com

Luc Langlois
安富利电子产品市场营销部全球解决方案小组总监
Luc.Langlois@avnet.com

W

无线通信将在多种新兴技术中扮演重要角色，例如自动驾驶汽车车队，以及连接数百万工业传感器的异构网络。此类应用环境需要用到即时调整调制方案、频带和系统协议的可重配置软件无线电 (SDR)。通过在全面验证的系统级模块 (SOM) 中紧密集成关键 RF 信号路径和高速可编程逻辑，安富利的 PicoZed SDR 能够在一副扑克牌大小的设备中提供灵活的软件无线电技术，实现 70MHz-6.0GHz 频率范围内的捷变频、宽频带 2x2 接收和发送路径，以满足不同固定和移动 SDR 应用的需求。

PicoZed SDR 将 Analog Devices AD9361 集成式 RF Agile Transceiver™ 与赛灵思 Z-7035 Zynq®-7000 All Programmable SoC 相结合。[1] 这种架构理想适用于复杂应用的软硬件混合实施方案，例如数字接收器，其中的数字前端（物理层）在可编程逻辑中实现，而上面的协议层则运行于双 ARM® Cortex™-A9 处理器的软件中。让我们看一看整个开发过程中 PicoZed SDR 的软件相关特性。

利用 PICOZED SDR 环内无线快速进行概念验证

要想充分发挥 PicoZed SDR 的潜力，需要采用稳健可靠的多域仿真环境来对从 RF 模拟

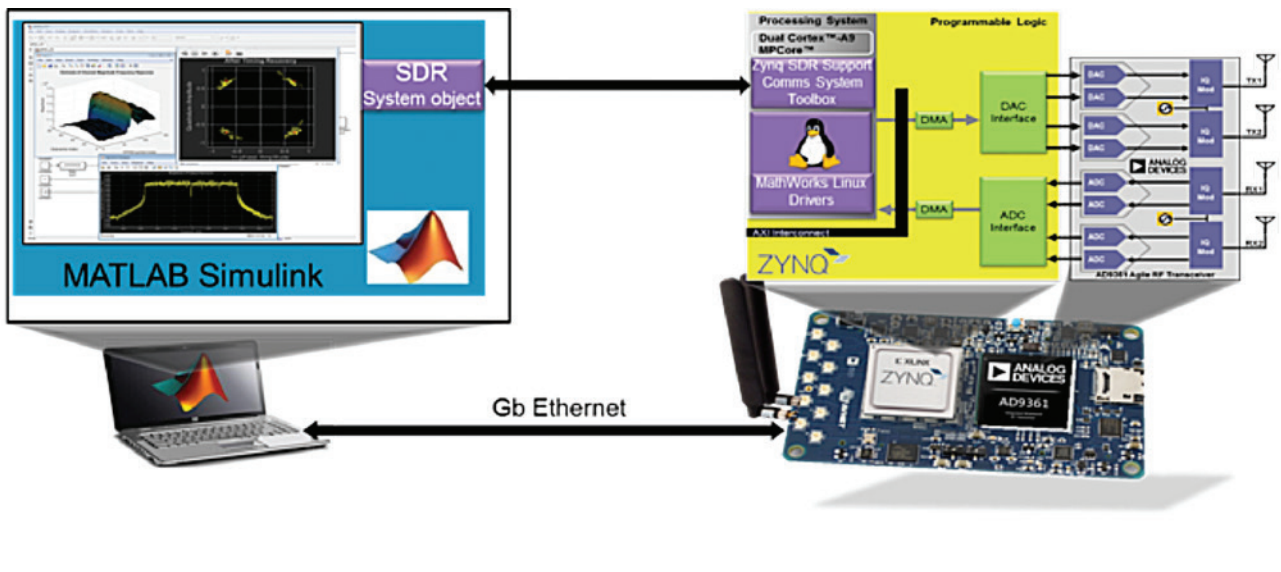


图 1 - 利用 System object 实现的 PicoZed SDR 流数据

电子到基带数字算法的整个信号链进行建模。这是 MathWorks 公司基于模型的设计方法所固有的价值，系统模型居于开发流程的中间环节，整个过程涵盖设计的需求定义、代码生成、实现和测试等。安富利携手 Analog Devices 和 MathWorks 合作，从最初的原型设计阶段开始，在设计过程从各个方面为 PicoZed SDR 开发支持架构。[2]

面对缩短开发周期的持续压力，工程师需要探寻合适的解决方案以便在可靠的硬件上进行快速、精确的概念验证，从而演示产品在“真实世界”条件下的可行性。MathWorks 使用名为 System objects™ 的 MATLAB® 软件架构为基于赛灵思 Zynq 的无线电创建程序支持包，该支持包能够将作为 RF 前端的 PicoZed SDR 用来构建开箱即用的 SDR 设计原型。System objects 软件架构专门针对用于处理大数据流的迭代计算而优化，能够在环内无线配置中自动实现 PicoZed SDR 与 MATLAB 和 Simulink® 环境之间的流数据（图 1）。类似于以对象为中心的编程概念，System objects 也是通过构造函数调用类名称的方式来创建，可采用 MATLAB 代码或者用作 Simulink 模块。System object 实例化后，

可调用各种方法在仿真过程中通过 System object 发送流数据。MathWorks 提供的针对赛灵思 Zynq 无线电的 Communications System Toolbox™ 支持包，含有针对 PicoZed SDR 接收器和发送器预定义的类别，每个类别包含针对 AD9361 的可调节配置属性，例如 RF 中心频率和采样速率。图 2 中的代码实例可用来创建一个 PicoZed SDR 接收器 System object，以接收单通道上的数据，AD9361 本地振荡器频率设定为 2.5GHz，基带采样速率为 100 Msps。使用日志保存获取的数据。

LIBIIO 库

Analog Devices 开发了 Libiio 库 [3, 4]，可简化用于连接 Linux Industrial I/O (IIO) 器件（例如 PicoZed SDR SOM 上的 AD9361）的软件的开发工作。开源 (GNU Lesser General Public License V2.1) 库抽象出硬件的低层细节，并提供一个可用于高级项目的简单但完整的编程接口。

该库由一个高级应用编程接口和一套后端组成，如图 3 所示。

您可以使用 Libiio 在项目原型设计阶段与 PicoZED SDR 进行接口连接，以便向或从 MATLAB、Simulink 或 GNURadio 等工具的模型发送或接收样本流。

- 本地后端与 Linux 内核通过内核的 sysfs 虚拟文件系统进行接口连接。这个后端具有 C、C++ 和 Python 绑定，用以支持运行于 PicoZED SDR 上的远程部署应用程序。
- 网络后端通过网络链路与 IIO Daemon (iiod) 服务器进行接口连接。网络后端支持多种操作系统（Linux、OS X、Windows）以在运行 MATLAB 和 Simulink [5]、GNURadio [6] 或 IIO Oscilloscope [7] 等应用程序的更强大主机平台上进行基于 GUI 的远程调试。

用户需要使用 Libiio 在项目的原型设计阶段与 PicoZED SDR 进行接口连接，以向 / 从 MATLAB、Simulink 或 GNURadio 等工具的模型发送或接收样本流，这些工具可以建模物理层（(QPSK、QAM、OFDM 等）或整个媒体访问控制器 (MAC)。Libiio 支持中等数据速率（约 8Msps）下的流传输（不丢失样本），以及最大数据速率 (61.44 Msps) 下的突发模式（获得样本突发传输流（高达 1M 左右的样本），会在突发流之间丢失数据）。通常，当进行 PHY 开发时应使用速率较低的数据流，然后，利用突发模式在生成 HDL/C 代码之前快速验证设计。

利用针对 PICOZED SDR 的软硬件协同设计进行系统集成

在使用 PicoZed SDR 环内无线完全验证了算法模型之后，接下来是生成 HDL/C 代码并将 IP 核打包，以集成到更大的系统。例如，在 MATLAB 和

```
rx = sdrrx('PicoZed SDR',...
           'IPAddress','192.168.3.2',...
           'CenterFrequency',2.5e9,...
           'BasebandSampleRate', 1e6,...
           'ChannelMapping', 1);

Log = dsp.SignalSink;
for counter = 1:20
    [data, dataLength, lostSample] = step(sdrrx);
    if lostSample ~= 1 % no dropped samples
        if dataLength == sdrrx.SamplesPerFrame % received desired data
            step(Log, data);
        end
    else
        disp('lostSamp=1, data lost');
    end
end
```

图 2 - PicoZed SDR 接收器 MATLAB System object

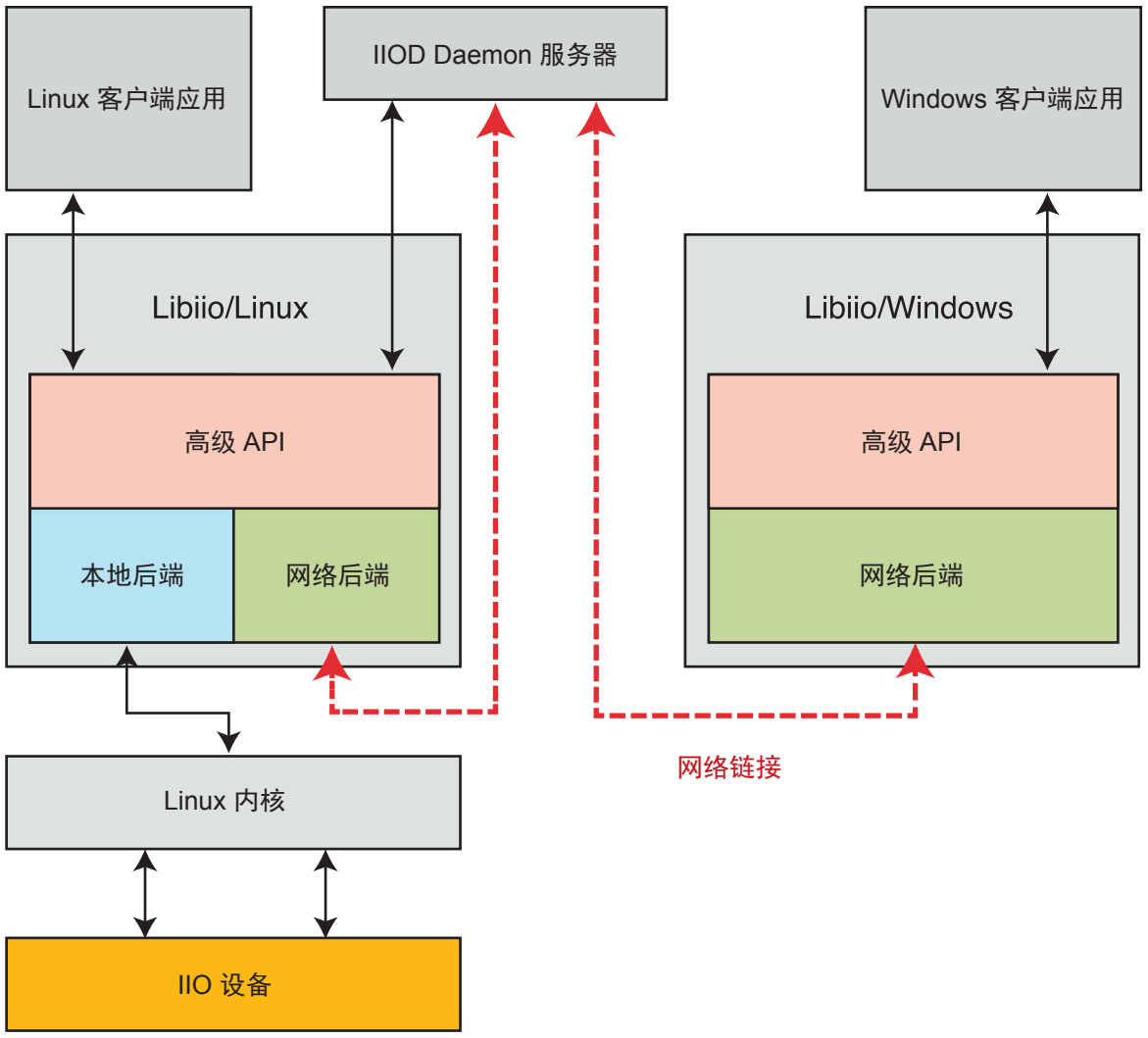


图 3 - Libiio API 和后端

Simulink 中建模的无线接收器子系统可能被指定用于点对点无线电链路，以从安装在 PicoZed SDR 载卡上的安富利摄像机模块传输实时视频。

MathWorks HDL Coder™ 中的软硬件协同设计流程使用户可以探索针对 Zynq SoC 的软件与硬件之间的最佳设计分区（图 4）。针对可编程逻辑的部分可自动封装为 IP 核，包括硬件接口组件，例如 ARM AMBA® AXI4 或 AXI4-Lite 接口访问寄存器，AXI4 或 AXI4-Lite 接口，AXI4-Stream 视频接口，以及外部端口。MathWorks HDL Workflow Advisor IP 核生成流程使用户可以将生成的 IP 核插入赛灵思 Vivado® 集成设计环境中的预定义嵌入式系统项目。[8] HDL Workflow Advisor 包含 Vivado IDE 将用户设计部署到 SoC 平台所需的所有元素，除了用户生成的自定义 IP 核和嵌入式软件。

如果用户有 MathWorks Embedded Coder® 许可证，就可以自动生成软件接口模型，从中生成嵌入式 C/C++ 代码，并在 Zynq SoC 中 ARM 处理器上的 Linux 内核上构建和运行可执行文件。所生成的嵌入式软件包含 AXI 驱动程序代码。该代码由

osc 应用支持 Linux、Windows 和 OSX。由于支持 HDMI 视频显示，因此它可运行于远程连接的主机 PC 或 PicoZed SDR FMC 载卡。

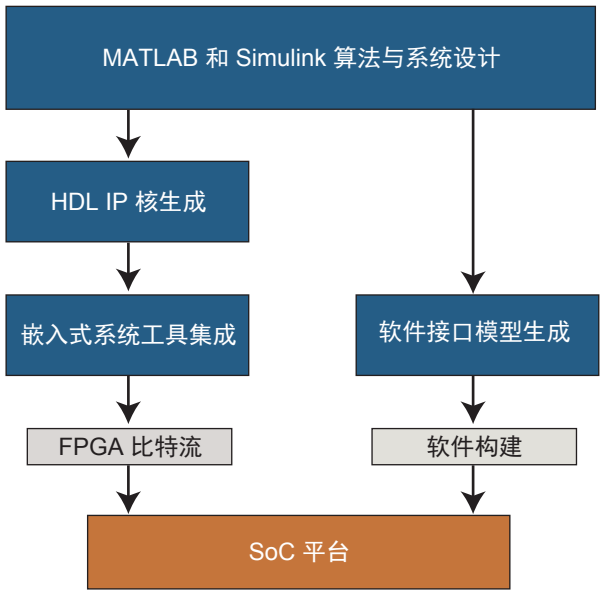


图 4– MathWorks 软硬件协同设计流程

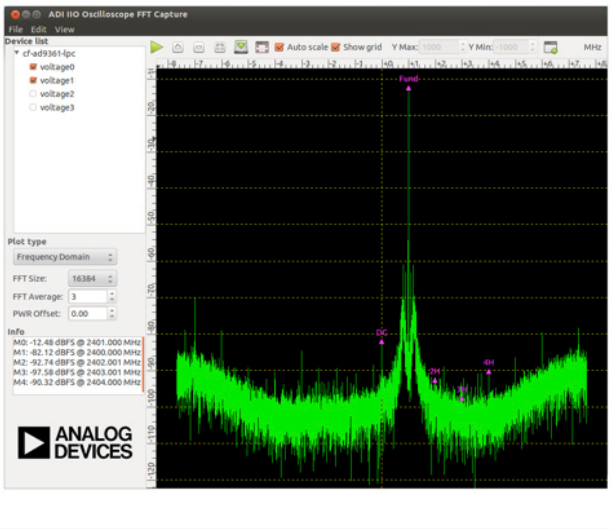


图 5 - ADI IIO 示波器

AXI 驱动程序模块生成的，用以控制 HDL IP 核。或者，用户也可以编写嵌入式软件，手动为 ARM 内核构建。

IIO 示波器

ADI IIO 示波器 (osc) 是一个示例应用，用以演示如何对 Linux 系统中的不同 Linux IIO 设备进行接口连接。该应用允许用户以四种模式（时域、频域、星座图和互关联）绘制捕获的数据，并查看和修改多个 IIO 设备设置。

osc 应用支持 Linux、Windows 和 OS X。由于支持 HDMI 视频显示，以及图形环境，因此它可运行于远程连接的主机 PC 或 PicoZed SDR FMC 载卡上。提供有关如何在 Linux 上构建 osc 应用的使用说明，以及预先创建的 Windows 二进制文件。图 5 给出了 PicoZED SDR 的两个通道 (I/Q) 的快速傅里叶变换 (FFT)，建立了标记以查看单音信号和测量谐波。

直接在 PicoZED SDR 上构建 osc 应用只需要 (1) 下载源代码：

```
> git clone https://github.com/analogdevicesinc/iio-oscilloscope.git
> cd iio-oscilloscope
> git checkout origin/master
```

(2) 构建并安装：

```
rgetz@pinky:~/iio-oscilloscope$ make
rgetz@pinky:~/iio-oscilloscope$ sudo make install
```

凭借强大的处理系统（双 ARM Cortex-A9 处理器，运行频率为 1GHz，外加 1GB DDR3 SDRAM），在 Zynq SDR SOM 上进行本机编译过程

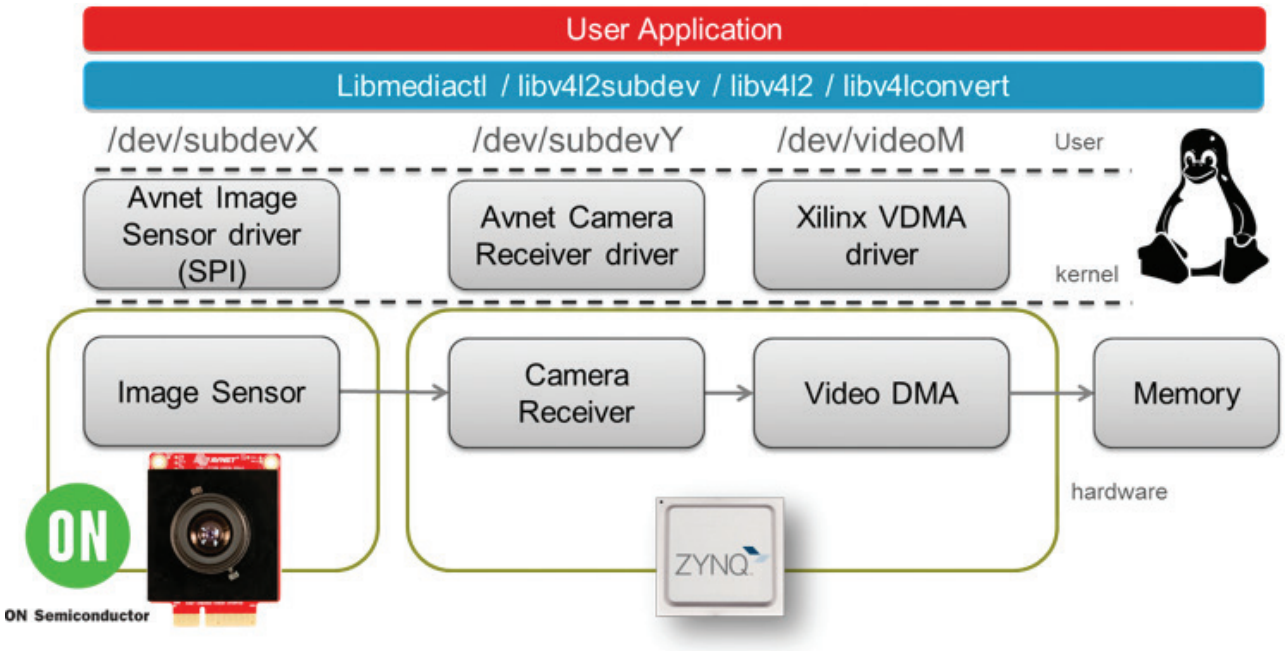


图 6 - 采用安富利 ON PYTHON-1300 摄像机模块的 V4L2 视频捕获流水线

非常快速。

采用 PICOZED SDR 的实时视频捕获系统

高性能视频已成为自动汽车、军用视觉系统、监控系统和无人机等无线应用领域中智能系统的重要组成部分。这些应用产品将高像素率视频捕获与实时分析相结合，可打破纯软件方案的性能限制。凭借 Zynq SoC 和赛灵思的 SDSoC™ 开发环境，嵌入式视觉系统设计人员能兼具两者的优势，利用原有软件图像处理算法的丰富功能以及硬件加速，以高帧率实时处理高清晰视频。安富利 ON PYTHON-1300 摄像机模块采用安森美 (ON Semiconductor) 的 PYTHON-1300 彩色图像传感器，能够达到 210 fps 下的 SXGA 分辨率（1,280 x 1,024 像素）。[9]。安富利的多个基于 Zynq SoC 的开发平台 —— 包括 PicoZed SDR —— 都支持该模块（以实现视频分析的无线传输）。系统设计人员可以使用符合 Video4Linux2 API 规范（V4L2；图 6）的安富利软件驱动程序将摄像机模块集成到完整的 Linux 系统中。V4L2 框架可以实现被称为流水线的完整视频数据路径。典型的视频捕获流水线从摄像机接收器获取视频，选择性地处理视频，然后使用视频 DMA 引擎将内容发送至外部帧缓冲器。安富利提供与 Vivado IP Integrator 兼容的“摄像机接收器”IP 核 HDL 源代码（不收取费用和版权费），以及以 Linux 补丁形式提供的 V4L2 子设备 Linux 驱动程序。

本文中我们已经介绍过，通过安富利的 PicoZed SDR 中提供的自动工作流程，用户可大大缩短从概念带部署的开发时间，同时专注于实现 SDR 产品的差异化特性。■

参考资料

1. PicoZed SDR 开发套件
2. 用 MATLAB 进行无线通信设计
3. 什么是 libiio?
4. analogdevicesinc/libiio GitHub
5. IIO System Object
6. GNU 无线电
7. IIO 示波器
8. ADI 参考设计 HDL 用户指南
9. 安富利 ON PYTHON-1300-C 摄像机模块

COMTECH 科通

高性能软件无线电模块 - HT76 系列

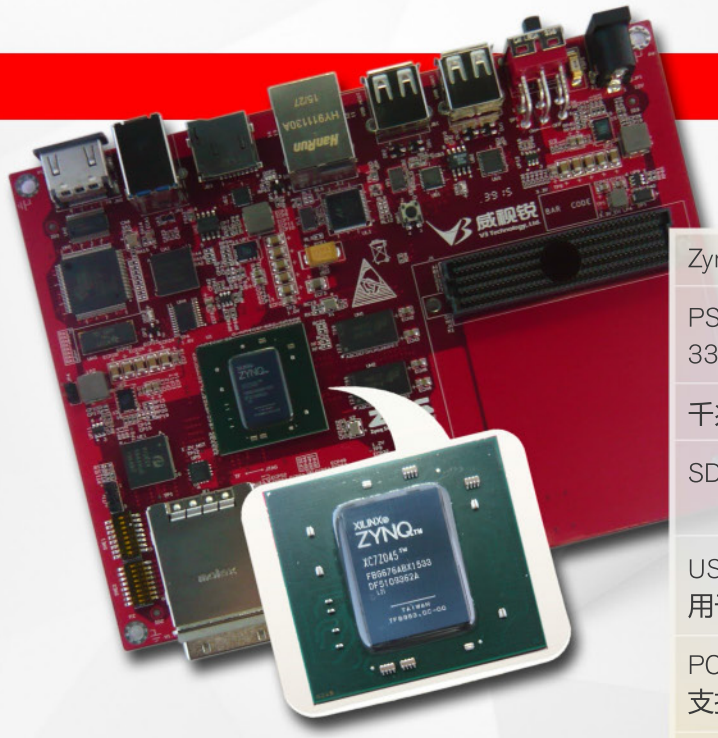
基于 Xilinx ZYNQ SoC XC7Z030/35/45

射频前端 ADI AD9361，70MHz~6GHz，MIMO 2x2

针对 1.44GHz 无人机图传专用
频段优化设计，其他频段可定制

发射功率 1W，视距
覆盖 6~8 公里半径

高性能通用 ZYNQ 开发平台



Zynq-7000 XC7Z030/45	512MB的DDR3内存颗粒;
PS系统时钟源： 33.333MHz LVCMOS晶振;	USB2.0 OTG接口， 支持WiFi扩展和U盘储存;
千兆以太网接口;	4GB Nand FLASH;
SDIO接口扩展TF卡;	PL系统时钟源： 50MHz LVCMOS晶振;
USB_UART接口， 用于系统调试;	USB3.0 设备接口， 支持5Gbps传输速率;
PCI Express2.0接口， 支持20Gbps传输速率;	HDMI输出接口;
用户IO：拨码开关、按键、LED灯;	FMC 扩展接口;
JTAG调试接口;	提供FMC的扩展接口

联系我们

北京：010 - 5380 8788　**南京：**025 - 8689 2261　**武汉：**027 - 8769 0155
成都：028 - 8513 9576　**上海：**021 - 5169 6680　**西安：**029 - 8450 8551
广州：0755 - 2674 3157　**杭州：**0571 - 8102 0941　**深圳：**0755 - 2674 3210

AMP 为您的下个 SoC 项目助力

作者：Scott McNutt
高级软件工程师
DesignLinx Hardware Solutions 公司
smcnutt@designlinxhs.com

充分利用 Linux 的实时性能和丰富特性。

嵌入式系统一般分为两大类：需要硬实时性能的；和不需要硬实时性能的。过去，我们不得不做出艰难抉择，即选择实时操作系统的性能还是我们钟爱的 Linux 系统的丰富特性，然后努力弥补不足之处。

如今，嵌入式开发人员再也不需要在二者之间艰难选择。非对称多处理 (AMP) 兼备二者的优点。几款新型片上系统 (SoC) 产品集成了多个 CPU、多种标准 I/O 外设和可编程逻辑。例如，赛灵思 Zynq-7000® All Programmable SoC 系列包含一个双核 ARM® Cortex™-A9、标准外设（例如千兆位以太网 MAC、USB、DMA、SD/MMC、SPI 和 CAN）以及庞大的可编程逻辑阵列。我们可将这些 SoC 产品作为 Linux/RTOS AMP 系统的基础，助其实现高度的灵活性。

典型的 AMP 配置在很多方面类似于基于 PCI 的系统，即 Linux 域起到主机作用，RTOS 域起到适配器作用，并有一个或多个共享存储器域用来实现两个域之间的通信。不过与 PCI 不同，AMP 配置能更方便、动态地为一个或另一个域分配资源（标准外设和自定义逻辑）。此外，Linux/RTOS AMP 系统能根据运行时间要求 —— 例如各种外部设备的有无 —— 动态地重新配置可编程逻辑。

灵活程度通常会与建立 AMP 系统所涉及的复杂性和难度息息相关。不过请放心，Linux 开发社区已经将很多功能引入到核心，能大大简化 AMP 配置与使用。

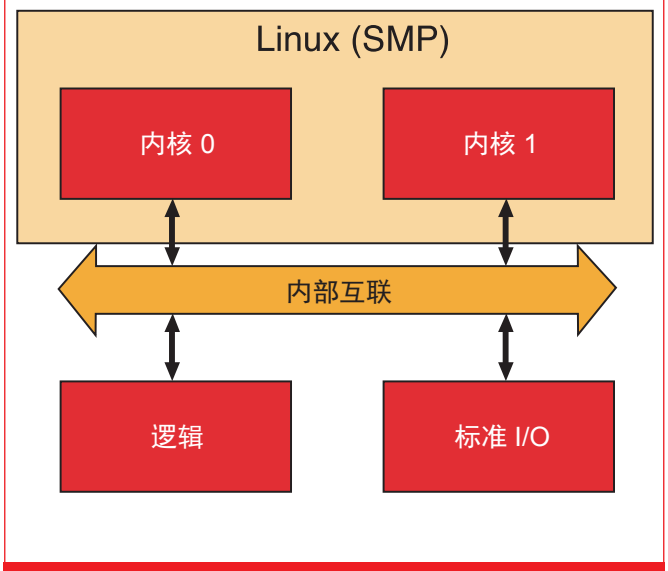


图 1 — 对称多处理。SMP 核心可在多个内核上同时运行。

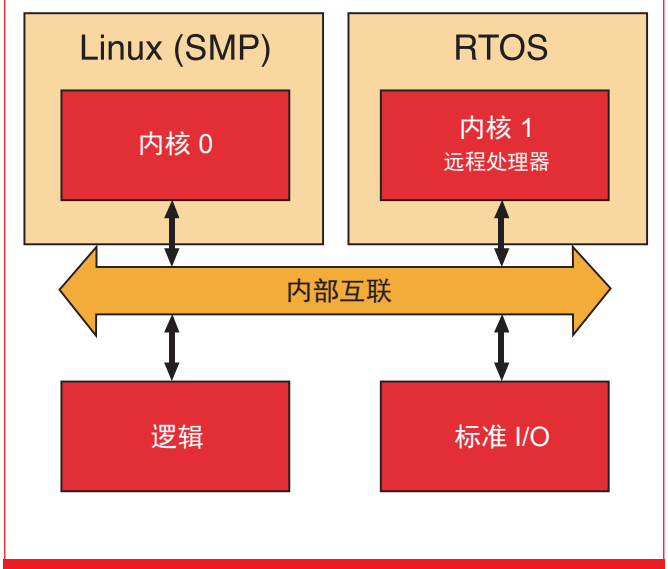


图 2 — 具有 Linux SMP 核心的 AMP

LINUX 多处理简介

就多处理而言，Linux 核心分为两种：单处理器 (UP) 核心和对称多处理器 (SMP) 核心。无论有多少个内核，UP 核心只能在单个内核上运行。AMP 系统可包含两个或更多个单处理器内核的实例。

SMP 核心可在一个内核或同时在多个内核上运行（图 1）。可选的核心命令行参数控制系统初始化之后 SMP 核心所使用的内核数量。核心运行时，各种命令行实用程序会控制分配给核心的内核数量。能够动态地控制内核所使用的内核数量，这是 SMP 核心比 UP 核心更受 AMP 开发人员青睐的主要原因。

远程处理器框架

远程处理器 (remoteproc) 框架是一种 Linux 组件，负责启动和停止各个内核（远程处理器），以及在 AMP 系统中加载内核的软件。例如，我们可以将图 1 所示的 SMP 系统动态地重新配置为图 2 所示的 AMP 系统，然后再使用 remoteproc 的功能配置回 SMP。

我们可以通过用户空间应用程序或系统初始化脚本完全控制重配置。重配置控制功能使用户应用可以根据系统的动态需求停止、重新加载和运行多种 RTOS 应用程序。

内核的软件（本例中是指 RTOS 和用户应用程序）从标准的可执行和可链接格式 (ELF) 文件中加载，该文件包含一个资源表的特殊段。资源表类似于 PCI 配置空间，用来描述 RTOS 需要的资源。

这些资源中包括 RTOS 代码和数据所需的存储器。

追踪缓冲区

追踪缓冲区是自动在 Linux 文件系统中作为文件出现的存储器区域。顾名思义，追踪缓冲区向远程处理器提供基本追踪功能。远程处理器向缓冲区写入追踪、调试和状态消息，以便通过 Linux 命令行或定制应用进行检查。

能够动态地控制核心所使用的内核数量，这是 SMP 核心比 UP 核心更受 AMP 开发人员青睐的主要原因。

在资源表中输入条目，以请求一个或多个追踪缓冲区。尽管一般包含纯文本，但追踪缓冲区也会包含二进制数据，例如应用状态信息或警报指示。

虚拟 I/O 设备

我们还可使用资源表定义虚拟输入 / 输出设备 (VDEV)，这种设备主要是支持 Linux 核心与远程处理器之间消息传送的几对共享存储器队列。VDEV 定义包括用来设定队列大小的字段，以及用来在处理器之间发信号的中断。

Linux 核心可处理虚拟 I/O 队列的初始化。远程处理器上运行的软件只需要在其资源表中包含一个 VDEV 描述，然后在开始执行时使用队列； 剩下的都由核心来处理。

远程处理器消息框架

远程处理器消息 (rpmsg) 框架是基于 Linux 核心的虚拟 I/O 系统的软件消息总线。该消息总线类似于局部区域子网络，单个处理器可在其中通过共享存储器创建可寻址端点和交换信息。

核心的 rpmsg 框架起到开关的作用，根据消息中包含的目的地址将消息传送到相应端点。由于消息报头包含源地址，因此可在不同处理器之间建立专用连接。

命名服务

处理器可通过向 rpmsg 框架的命名服务发送消息，以动态宣布特定服务。命名服务功能本身用途不是很大。不过，rpmsg 框架允许将服务名称关联到设备驱动程序，以支持驱动程序的自动加载和初始化。

例如，如果远程处理器宣布 dlinx-h323-v1.0 服务，那么核心可以搜索、加载和初始化与该名称关联的驱动程序。如果系统中服务被动态安装在远程处理器上，那么这样可大大简化驱动程序管理。

管理中断

中断管理有些棘手，尤其在启动和停止内核时更是如此。最终，系统需要在远程处理器启动时动态地将特定中断重定向至远程处理器域，然后当远程处理器停止时收回中断。此外，系统必须保护中断，防止其被错误配置的驱动程序误分配。简言之，必须在系统层面管理中断。

对于 Linux SMP 核心而言，这是一个常规事件，而且是 SMP 核心在 AMP 配置中更受青睐的另一个原因。远程处理器框架能方便地管理中断，只需来自设备驱动程序的最小支持。

设备驱动程序

设备驱动开发是个始终需要关注的问题，因为所需的技能组合可能无法立刻提供。幸运的是，Linux 核心的 remoteproc 和 rpmsg 框架完成大部分重活； 驱动程序只需要实现几个标准驱动程序例程。功能完整的驱动程序可能只需要几百行代码。核心源代码树包含嵌入式开发人员可根据自身要求进行调整的驱动程序范例。

厂商还提供通用的开源设备驱动程序。DesignLinx Hardware Solutions 提供针对 Linux 和 FreeRTOS 的通用 rpmsg 驱动程序。由于通用驱动程序没有假定所交换消息的格式，因此嵌入式开发人员可将其用于多种 AMP 应用，无需做任何修改。

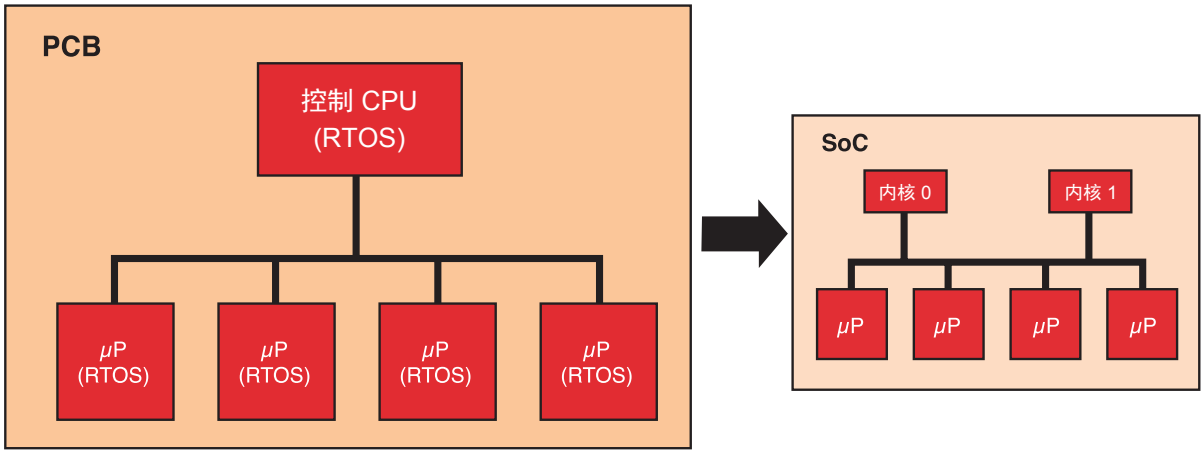


图 3 — SoC 的“引脚内”移动分立 PCB 元件

引脚内移动

核心的多处理支持并不局限于同构多处理系统（使用同一类型处理器的系统）。以上介绍的所有特性也可以用在异构系统中（具有不同类型处理器的系统）。当“在引脚内”移植已有设计时，这些多处理功能尤其有用。

新型 SoC 产品使设计人员能够方便地将各种硬件设计从印刷电路板移植到片上系统（图 3）。过去在 PCB 上作为分立处理器和组件的部分可以完全在 SoC 的引脚内实现。

例如，我们可以使用赛灵思 Zynq-7000 系列 SoC 实现图 3 中的初始 PCB 硬件架构，将其中一个 ARM 处理器作为可编程逻辑中的控制 CPU 和软处理器（例如赛灵思 MicroBlaze™ 处理器），以替代分立处理器。我们可以使用剩余的 ARM 处理器运行 Linux SMP 核心（图 4）。

将 Linux 添加到初始设计中能够为 ARM 内核和软核处理器提供以上描述的所有标准多处理功能（例如启动、停止、重载、追踪缓冲区和远程消息）。而且，还带来丰富的 Linux 功能集，可支持多种网络接口（以太网、Wi-Fi、蓝牙）、网络服务（Web 服务器、FTP、SSH、SNMP）、文件系统

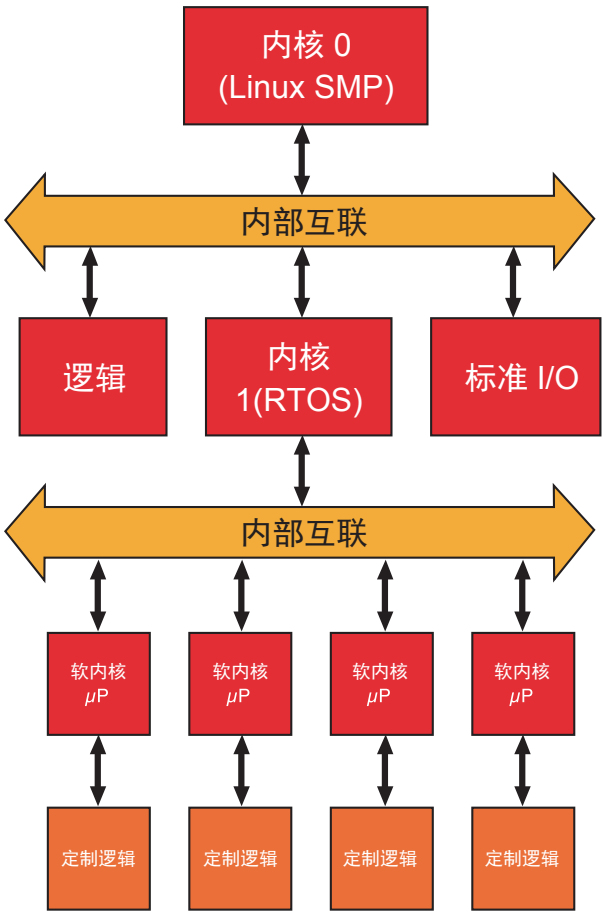


图 4 — 引脚内的多处理

新型 SoC 产品使设计人员能够方便地将各种硬件设计从印刷电路板移植到片上系统。

（DOS、NFS、cramfs、闪存存储器）以及其他接口（PCIe、SPI、USB、MMC、视频）等。这些特性能方便地实现新功能，无需对经过检验的架构做太大改动。

内核不断涌现

过去几年中，针对嵌入式市场的多核 SoC 产品不断增加，而且很适合用于 AMP 配置。

例如，赛灵思 UltraScale+™ MPSoC 架构包含一个 64 位四核 ARM Cortex-A53、一个 32 位双核 ARM Cortex-R5、一个图形处理单元 (GPU) 以及多种其他外设，当然还包括有用的可编程逻辑。这为那些清楚如何驾驭实时操作系统的性能以及 Linux 核心的丰富特性集的设计人员提供了沃土。

如需了解如何设计 Linux/RTOS AMP 系统的更多详情，敬请联系 DesignLinx Hardware Solutions。赛灵思联盟计划的高级成员 DesignLinx 专门从事 FPGA 设计与支持业务，包括系统设计、原理图捕捉和电气封装 / 机械工程设计，以及信号完整性设计。■

安卓开源 5.1 (Lollipop) 操作系统现可支持 Zynq UltraScale+ MPSoC

理想适用于运行高级用户界面的安全及保密产品

全可编程技术和器件的全球领先企业赛灵思公司 (Xilinx, Inc. (NASDAQ:XLNX)) 今天宣布安卓 5.1 (Lollipop) 现已可支持 Zynq® UltraScale+ ™ 多处理器 SoC (MPSoC)。通过其硬件支持计划，Mentor Graphics 凭借其在异构多核平台上丰富的安卓系统经验，成功将安卓开源项目 (AOSP) 代码移植并运行到 Zynq UltraScale+ MPSoC 上，从而打造出了业界首款将安卓平台高级用户界面和基于 64 位 ARMv8 架构的具有更高安全性与保密性和更高处理能力的 Zynq UltraScale+ 器件完美融合的解决方案。

异构 Zynq UltraScale+ MPSoC 将硬件资源隔离开来，使安卓等高级操作系统能够与集成实时处理单元上实时操作系统运行的安全关键功能共存。赛灵思公司软件产品市场营销高级总监 Ramine Roane 指出：“工业自动化、交通运输、医疗保健和无线电市场的设计人员不断寻求各种方法，希望打造出能够在不影响安全性和保密性的情况下囊括语音识别、丰富的用户界面、全套定制化等优异特性在内的产品。Zynq UltraScale+ MPSoC 和 AOSP 的独特组合，为打造此类产品提供了一个具有全面安全性和保密性的平台。”

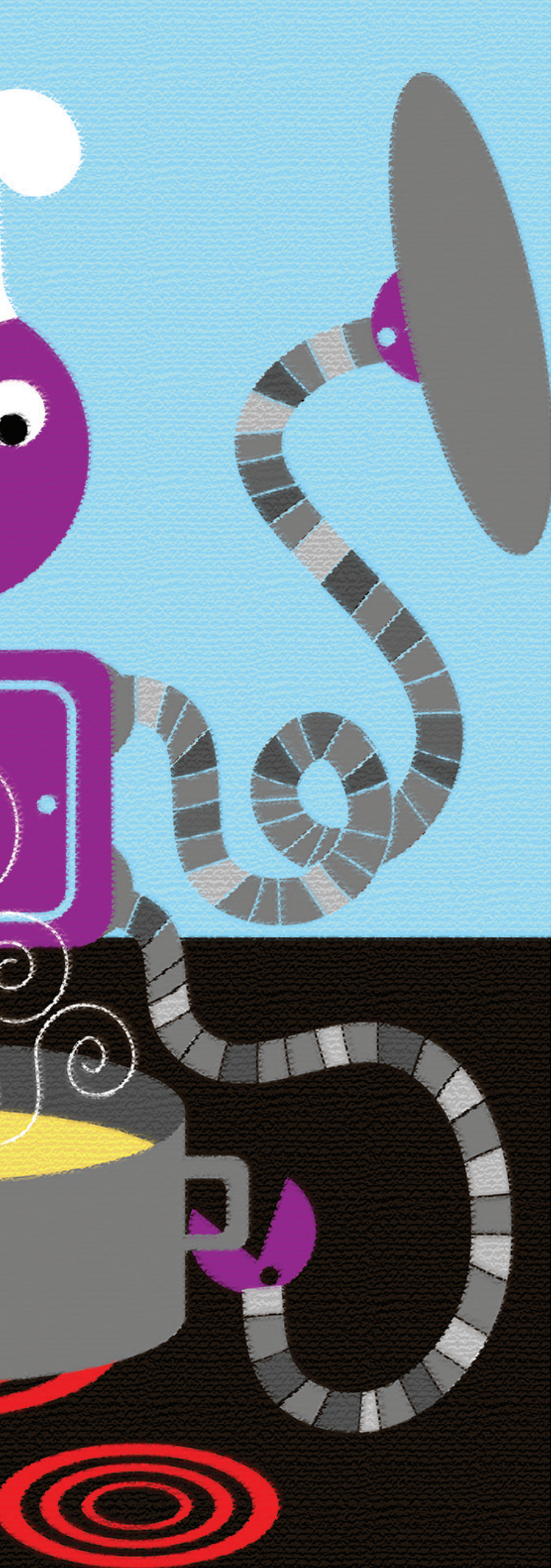
Mentor Graphics 公司嵌入式系统事业部运行时间解决方案产品管理总监 Warren Kurisu 表示：“凭借我们在移植安卓开源项目到 Zynq UltraScale+ MPSoC 上的专业技术，可以为赛灵思客户降低嵌入式开发难度。结合应用程序与实时处理、低功耗、高性能和集成可编程逻辑，以及隔离通用和实时子系统等功能，可为安卓进军非传统市场带来众多机会。我们期待着在安卓上支持 Zynq UltraScale+ MPSoC 客户，同时在 Mentor Embedded 产品线上提供一系列其他软件和工具。”

供货情况

Mentor Embedded 公司将于 2016 年第三季度针对 Zynq UltraScale+ MPSoC 提供安卓 5.1 (Lollipop)。Mentor 将通过商业支持、定制服务、平台移植和特性增强等，直接支持 Zynq UltraScale+ MPSoC 客户。所有上述支持都将成为赛灵思产品纯软件解决方案组合的一部分。更多信息，敬请访问：www.mentor.com/embedded-software/semiconductors/xilinx。

嵌入式系统的秘诀

随着嵌入式系统不断普及，我们可以从积累的开发知识中获得巨大优势，构建更出色的系统。



作者：Adam Taylor
e2v 首席工程师
aptaylor@theiet.org

工程师一刻也没忘记交付达到质量、时间安排和预算目标的项目的需求。您可以借鉴嵌入式系统开发人员社区多年来累计的经验教训，确保您下一个嵌入式系统项目达成这些目标。下面我们来了解一些为嵌入式开发带来了最佳实践的重要经验。

系统地思考

系统工程是一个广泛的专业领域，覆盖从航空母舰及卫星到实现其性能的嵌入式系统的所有开发工作。我们可以运用系统工程方法管理从概念到使用周期结束处置的嵌入式系统工程生命周期。系统工程方案的第一阶段跟常人想象的不一样，，不是确立系统需求，而是制定系统工程管理规划。这一规划不仅将为系统定义工程生命周期以及开发团队将要开展的设计评审，而且还将定义这些评审的预期输入输出。该规划可根据工程事件的次序和每个阶段的先决条件，为项目管理、工程和客户群体做出明确的定义。简而言之，它可展示预期和可交付项。在清楚理解工程生命周期的情况下，系统思考的下一步是确立正在开发嵌入式系统的需求。良好的需求集应覆盖三个方面。功能需求定义嵌入式系统如何开展工作。非功能需求定义法规遵从与可靠性等方面的问题。环境需求定义工作温度和冲击与振动以及电气环境（例如 EMI 和 EMC）等方面的需求。在较大规模的开发工作中，这些需求将从较高层次的规范向下延伸并且可跟踪，比如系统或子系统规范（图 1）。如果没有较高层次的规范，我们

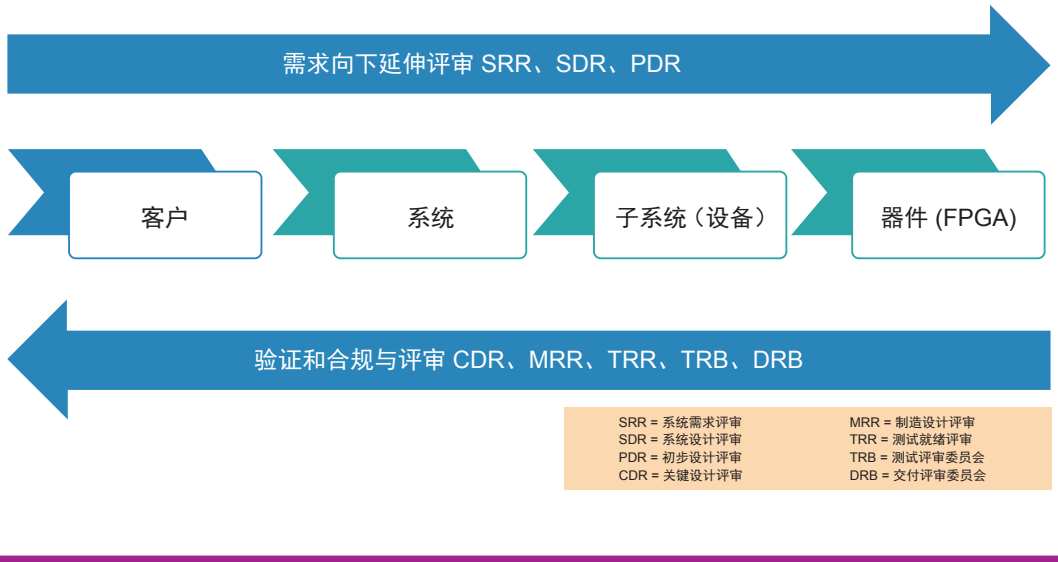


图 1 — 在开发工作中，需求从较高层次的规范向下延伸并且可跟踪。

必须在开发过程中接触利益相关方，确立一套明确的利益相关方需求，然后将其用于确立嵌入式系统需求。

生成一个良好的需求集，需要我们充分思考每一个需求，才能确保其符合这些标准：

1. 它是必要的。没有需求，我们的项目就不会取得成功。
2. 它是可验证的。我们必须确保该需求能通过检验、测试、分析或演示实现。
3. 它是可实现的。在给定的约束条件下，该需求在技术层面上是可以实现的。
4. 它是可追踪的。该需求能够从较低层次的需求进行追踪，而且可追踪较高层次的需求。
5. 它是唯一的。这项标准可防止需求之间的界限不清。
6. 它是简单清晰的。每条需求指定一项功能。

为体现意图，在定义需求时还常常使用特定语言。一般我们对强制性要求使用“必须”，对非强制性要求使用“应该”。非强制性要求可让我们表达必要的系统属性。

在我们确立了我们的需求底线后，最佳实践就是创建一个合规矩阵，说明符合每项需求。我们还可以通过为每项需求分配一种验证方法开始确立我们的验证策略。这些方法一般是测试、分析、检验、演示和交叉读取。根据合规及验证矩阵创建需求能让我们：

- 清晰地了解系统行为。
- 向内部测试团队和外部客户都演示验证方法。这不仅可在开发过程的早期阶段发现任何困难的测试方法，而且还可帮助我们确定所需的资源。
- 确定技术性能指标。这些指标来自合规矩阵，由存在无法合规的风险的各种需求构成。

分配工程预算

每个工程项目都涵盖一定数量的预算，我们应将其分配给在架构中识别的解决方案。预算分配不仅可确保项目实现整体需求，而且还可确保每个模

为在我们架构中使用的每项技术分配一个技术就绪指数，再结合合规矩阵，可帮助我们确定技术风险的所在位置。

块的设计牵头人理解模块的分配，以创建适当的解决方案。我们分配预算的典型领域有功能的总质量、功能的总功耗、用平均故障间隔时间或成功概率定义的可靠性以及设计中信号类型间的正当串扰（一般是一套适用于大量功能的通用规则集）。确立工程预算最重要的方面之一是确保我们有足够的应急分配。但我们必须战胜应急再加应急的想法，因为这会成为影响时间安排和成本的严重技术问题。

管理技术风险

从合规矩阵及工程预算的生成看，我们应该能够识别在技术上有难度的需求。每一个这类有风险的需求都应该有明确的规避计划，其将说明我们将如何实现这一需求。展示这一点的最佳途径之一是使用技术就绪指数 (TRL)。TRL 有 9 级，从所观察到的基本原理 (TRL1) 到完整功能与实地部署 (TRL9) 描述设计成熟度级数。把 TRL 分配给我们架构中使用的每一项技术，再结合合规矩阵，可帮助我们确定技术风险的所在位置。我们随后可启动一个 TRL 开发规划，确保在项目不断推进时，低 TRL 领域会提升到所需的 TRL 水平。该规划涉及的内容可确保我们在项目推进时实现和测试正确的功

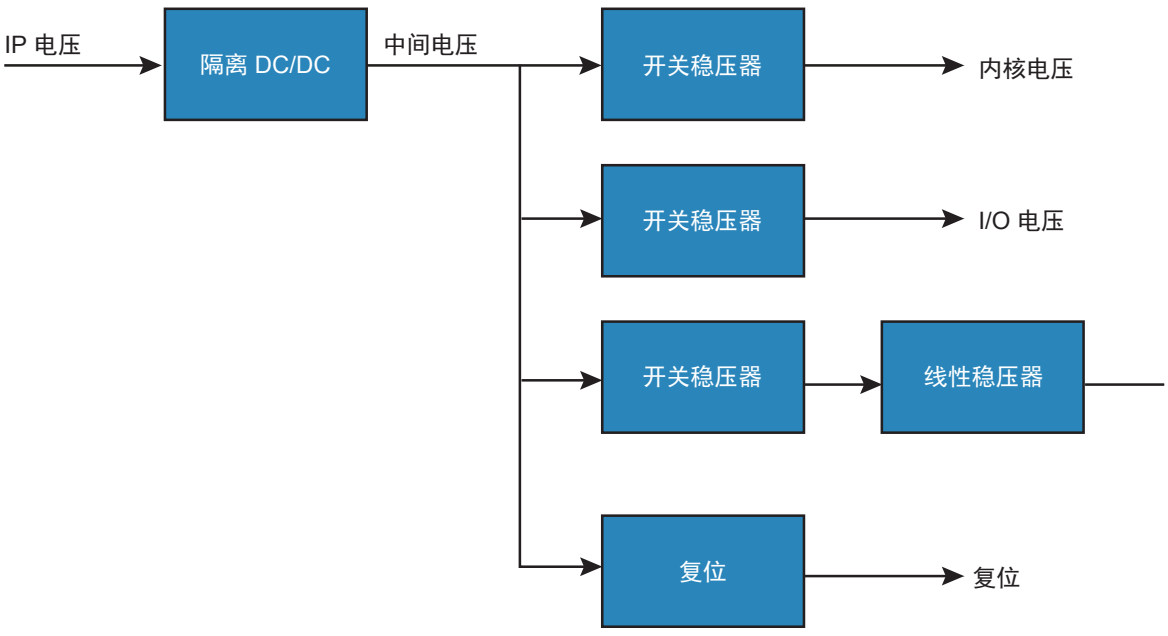


图 2 — 在本电源架构示例中，模块的输出轨需要二级稳压。

该架构不应仅限于硬件（电气）解决方案，还应包含 FPGA/SoC 及相关软件的架构。

能，或是在项目推进的过程中执行功能或环境 / 动态测试。

创建架构

理解嵌入式系统要求的行为后，我们就需要为解决方案创建一个架构。该架构将由分组成功能块的需求构成。例如，如果嵌入式系统必须处理模拟输入或输出，架构就将包含模拟 I/O 模块。其它模块可能会更加明显，比如电源调节、时钟和复位生成。

该架构不应仅限于硬件（电气）解决方案，还应包含 FPGA/SoC 及相关软件的架构。当然，模块化设计的关键是针对模块及功能行为的良好接口文档编制。

该架构的一个关键方面是展现如何在高层次上创建系统，这样工程团队就能轻松理解其实现方式。该步骤也是在系统运行生命周期中为系统提供支持的关键。

在确定我们的架构时，我们需要考虑模块化方法，这样不仅能在当前项目上进行复用，而且还能在未来的项目上进行复用。模块化要求我们从第一天起就考虑可能的复用，并要求我们把每个模块存档为一个独立的单元。就内部 FPGA/SoC 模块而言，像 ARM® AMBA® 高级可扩展接口 (AXI) 这样的通用接口标准有助于实现复用。

模块化设计的一个重大优势就是能够针对某些需求使用商用现成的模块。商用现成（COTS）模块让我们能够以更快的速度开发系统，因为借助 COTS，我们能够把我们的工作重点放在项目从我们的专业能力产生的增值中获益最大的部分上。

系统电源架构是一个需要缜密思考的的设计方面。许多嵌入式系统会要求隔离 AC/DC 或 DC/DC 转换器来确保嵌入式系统的故障不会扩散。图 2 显示的是电源架构的示例。来自该模块的输出轨需要二级调整来为处理内核和转换装置提供电压。我们必须仔细防范这些阶段发生严重的开关损耗和效率下降。因为效率降低意味着系统热耗散增大，如果不正确解决就会影响单元的可靠性。

我们必须仔细了解使用的线性调整器的行为以及在电源线上进行进一步滤波的要求。这一要求的原因是 FPGA 和处理器等器件的开关频率远远高于线性调整器的控制环路所能应对的水平。随着噪声频率提高，线性调整器的噪声抑制能力下降，导致需要采用额外的滤波和去藕技术。如果不了解这一关系，会造成混合信号设备出现问题。

另一个重要的考虑因素是时钟和复位架构，尤其是在有多个需要同步的开发板的情况下。在架构层面我们必须考虑时钟分配网络： 我们是否在跨多个开发板扇出单个振荡器，或是使用多个频率相同的振荡器？为确保时钟分配的稳健可靠性，我们必须考虑：

- **振荡器**启动时间。我们必须确保在整个时间周期内激活复位（如果需要）。
- **振荡器歪斜**。如果我们要在跨多个开发板扇出振荡器，时序是否至关重要？如果是，我们需要考虑线路卡上的歪斜（连接器引起的）和缓冲器自身引起的歪斜。
- **振荡器抖动**。如果我们在开发混合信号设计，我们需要确保使用低抖动时钟源，因为抖动的增大会降低混合信号转换器的信噪比。在我们使用千兆位级串行链路时情况也是一样，因为

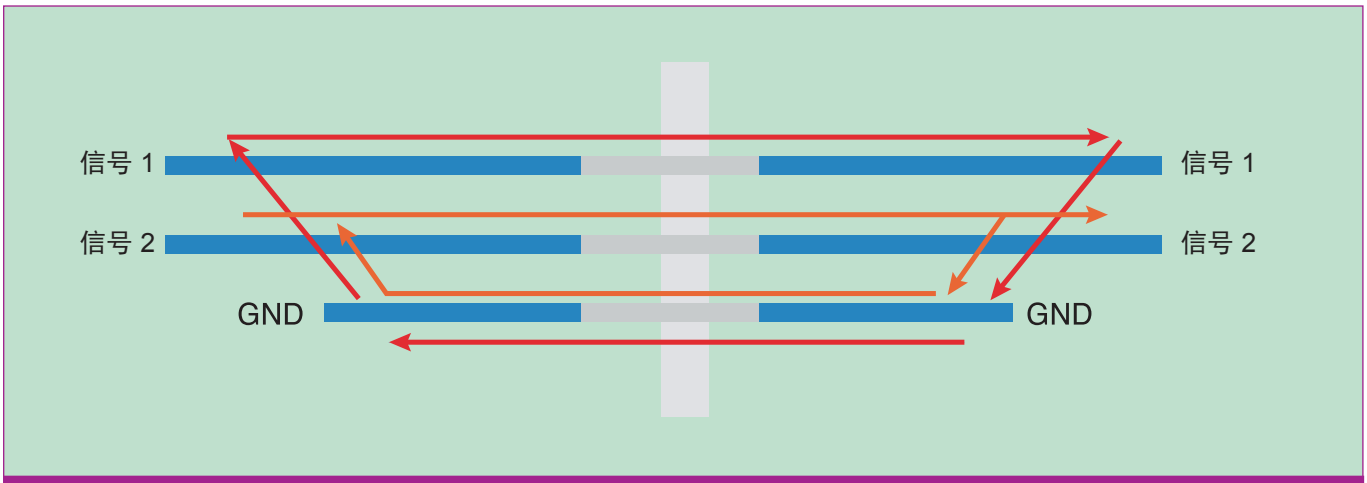


图 3 — 连接化是接口定义最重要的特征之一。

我们需要使用低抖动时钟源在链路上取得良好的误码率。

我们也必须注意复位架构，确保只在需要的地方使用复位。例如基于 SRAM 的 FPGA 一般不需要复位。

如果我们在使用复位的异步激活，我们需要确保移除它不会导致亚稳态问题。

清晰定义接口

内外部接口的正式文档在机械、物理和电气层面为各个接口提供清晰的定义，以及协议和控制流。这些正式文档也往往被称为接口控制文档 (ICD)。当然最好是尽量使用标准通信接口。

接口定义最重要的一个方面是外部接口的“连接化”。这个过程考虑了所需连接器的引脚分配，连接器引脚的额定功率以及所要求的插拔次数，以及任何对屏蔽的要求。

在我们为我们的系统考虑连接器类型的时候，我们应确保不会因为在子系统中使用相同类型连接器而造成不利的交叉连接。通过使用不同类型连接器或采用不同的连接器键位（如果支持），我们就能够避免交叉连接的可能性。

连接化是我们开始使用之前确定的预算要求的首个方面之一。特别是我们可以使用串扰预算来指引我们定义引脚分配。图 3 所示的例子说明了这一流程的重要性。重新安排引脚分配，将接地基准电压 (GND) 引脚布局在信号 1 和信号 2 之间，可以降低互感以及由此引发的串扰。

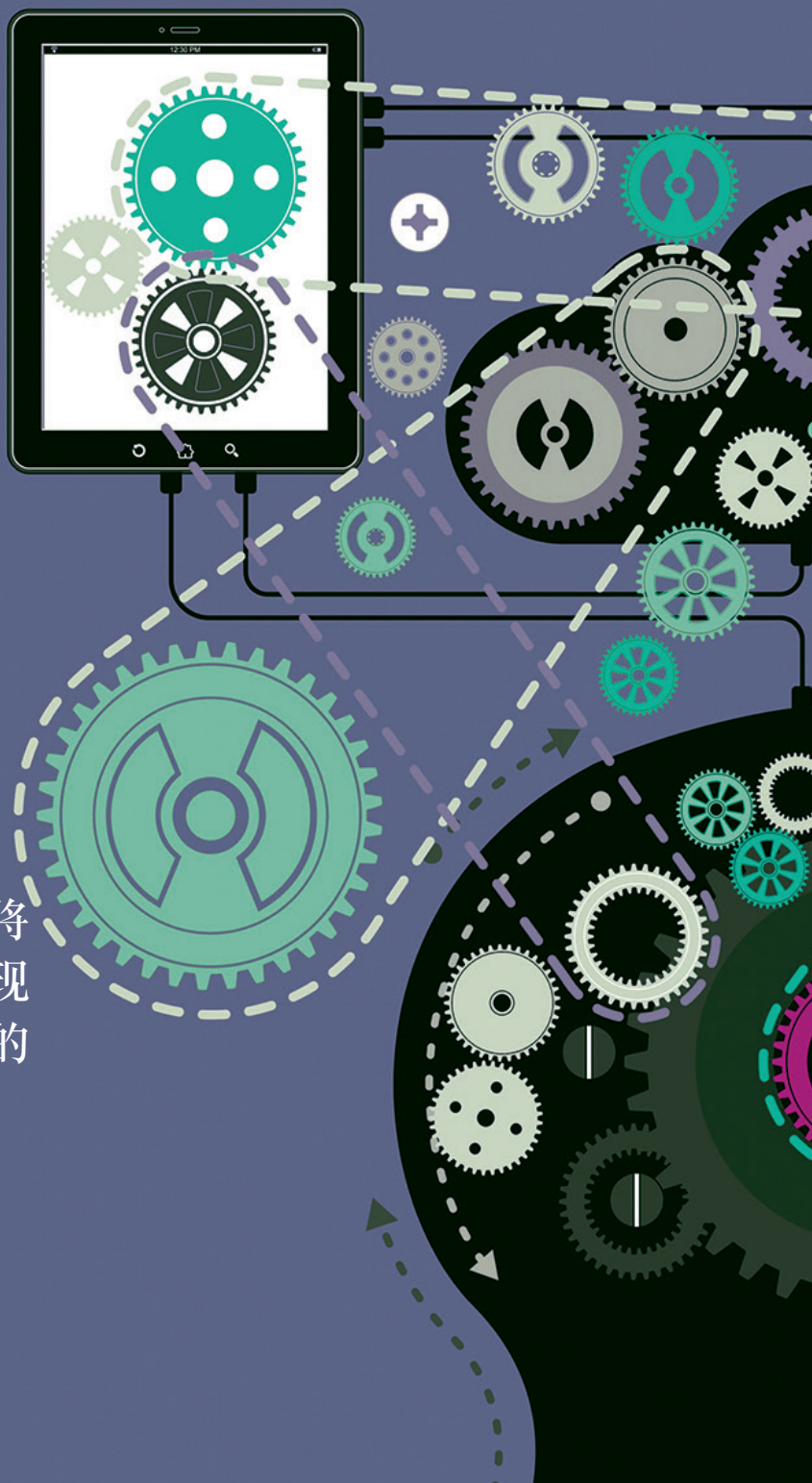
接口控制文档 (ICD) 必须对系统接地进行定义，尤其是在项目要求外部 EMC 的时候。在这种情况下，我们必须小心避免让有噪声的信号地产生辐射。

工程师和项目经理掌握着一系列策略，以确保他们交付的嵌入式系统能够满足质量、成本和调度要求。不过当项目遇到困难时，我们可以确信在项目不发生重大变化的情况下其此前的性能是其未来性能的良好提示。■

参考资料

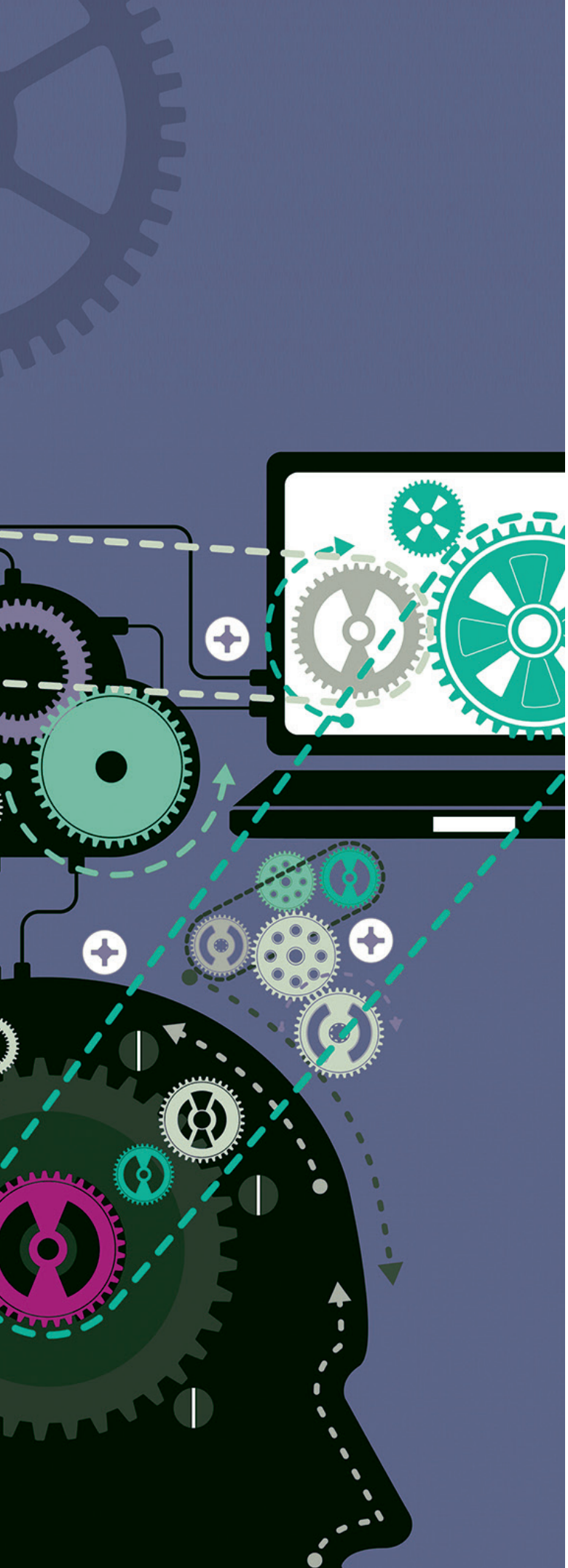
1.《在硬件设计中采用 FPGA 的基本要点》。*赛灵思中国通讯第 47 期*，第 42-49 页。
2.《让硬件设计轻松自如》。*赛灵思中国通讯第 50 期*，第 48-51 页。
3.《设计可靠性：MTBF — 这只是开始》。*赛灵思中国通讯第 53 期*，第 38-43 页。

下一个突破性创新数字业务模式：神经可塑性云



作者：Francesc Fon
MBA 博士
西班牙巴塞罗那伊萨德商学院 (ESADE)
francesc.fons@alumni.esade.edu

基于 FPGA 的基础架构准备将神经可塑性运用到 IaaS，实现高性能、定制化、设计安全的云计算服务。



N

神经科学家曾经认为人脑的结构是不变的，其神经元在幼年即已定型。后来研究证明人脑实际上是可塑的： 为了对环境激励做出响应并借助思维行为本身，人脑在动态地改变自己的结构、功能和神经连接，甚至通过神经再生长出新的神经元。而且这种神经可塑性会一直持续到老。

这种经证明的人脑可塑性正在开启基于灵活、强大、量身定制的云计算基础架构即服务 (IaaS) 的突破性创新数字业务模型。基于可定制云的 IaaS 有望通过综合与请求的服务有关的优化计算来实现创新性产品差异化，交付特定的服务质量 (QoS)，为最终用户带来增值。这种模式要求特别关注高计算强度功能或应用背后的计算机架构性能、定制化和安全性等设计参数。已经出现在可重配置硬件技术的 DNA 中并且现在可部署到 FPGA 驱动的云计算基础架构的这些技术特性，正准备改变数字化业务的游戏规则。

神经可塑性和云计算

近期神经科学的最新进展已经显示我们通过采用新的思维模式和行为方式就可以轻松地重塑

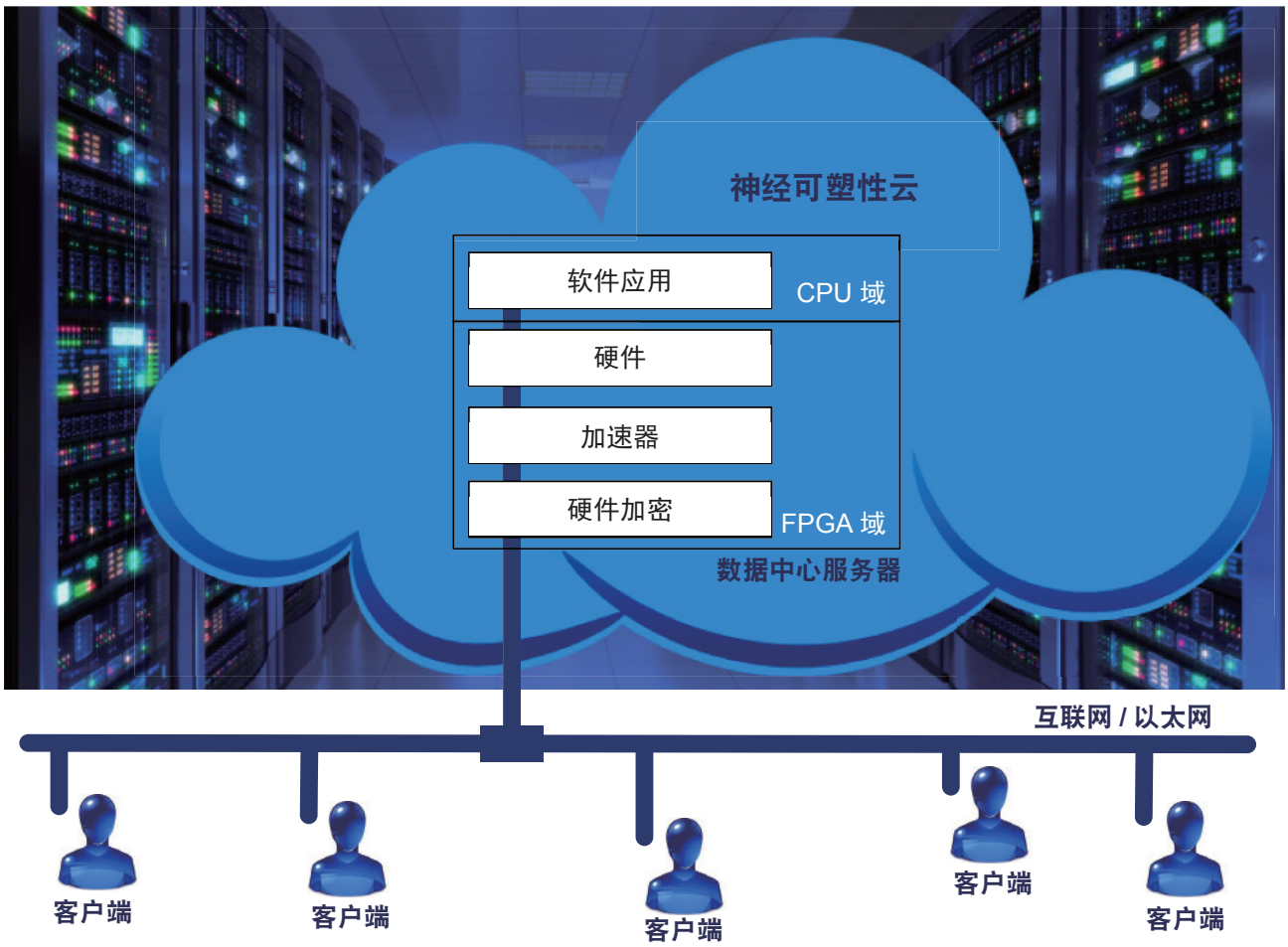


图 1 — 神经可塑性云架构

自己的大脑。大脑和神经系统用于定位行为的神经路径连接着大脑中相对较远的区域，而且每一个路径都与特定的领域或行为息息相关。新的思维和技能一旦出现，就会铺设新的路径。每次我们思考、感受或做什么事情的时候，我们就会通过重复练习来强化相关路径，直至行为成为习惯。

随着神经科学深化我们对大脑的理解，云计算技术在过去十年已取得长足发展，通过互联网为世界各地数十亿用户提供计算、联网和存储服务。个人、企业及其他机构正在尽情享用存储、视频、消息传输、社交网络、线上游戏和 Web 搜索等云服务。云服务不止是一种 IT 现象；它越来越多地成为企业在发展自己的业务的同时借助不断提升的计算能力降低成本、实现创新和业务转型的动力。行业观察家已预计未来数年 IT 基础设施的主要开支将投入在云平台和应用上。

这就是说我们能否在发现神经可塑性和云计算爆炸式增长之间找到某种关系，可能激发新的数字业务模式？

试想对人的神经系统的天生神经可塑性进行抽象，并将这个概念直接移植到云计算基础架构上。通过支持 FPGA 和 SoC 器件中可用的可重配置硬件

众多研发团队深信 FPGA 是云计算服务器中的应用专用加速器的未来实现技术。

技术，突破性 IaaS 模式可根据任何给定时间客户提出的特定需求来适配和优化分布在任何云计算基础架构的数据中心中的服务器计算机架构，从而得到的计算转化为增值。为强化神经科学类比，我们使用“神经可塑性云”一语来描述将软核处理器与可编程逻辑融合在一起 (FPGA)，以提供异构硬件 - 软件处理生态系统的物理云计算架构。通过这种生态系统，数据中心服务器可根据云中使用的特定应用定制和适配自己的计算能力（图 1）。

这一概念激励的是一种云计算模式的转变。在这种转变中，任何给定计算“大脑”的适配能力，一旦该“大脑”在可重配置硬件中综合为移植到云端的定制计算单元，就能提供与在严格和特定的计算要求（往往通过今天的标准云解决方案无法实现）下履行云服务的数字业务相比明显的竞争优势。能从这种方法中获益的应用领域包括金融趋势分析、实时医疗成像处理、生物信息学、计算生物学、基因组测序、能源 / 石油 / 天然气分配实时控制、大数据分析和深度学习。

把可重配置硬件技术引入到云计算领域的理念肇始于学术研究。可重配置计算社区一段时间以来一直在探索将 FPGA 与通用处理器相结合的机会，而且众多研发团队已经确信 FPGA 是未来的云计算服务器中应用专用加速器的实现技术。这一理解尚未转化为广泛可用的商用解决方案； 但微软、IBM、英特尔、高通和百度等云技术主导企业加上 FPGA 厂商正在极力倡导将 FPGA 用于数据中心工作负载的优化。

在这个数字服务经济推动云计算增长的时代，云计算利益相关方逐渐认识到，数据中心采用 FPGA 比采用其它可用替代方案，能提供更高的性能功耗比。这意味着云服务提供商会很快有新的策略，用于交付为他们的客户的特定需求定制的计算基础架构，并通过响应时间、网络安全，当然还包括性能等有意义的 QoS 参数加以量化。任何数字化企业的云计算基础架构都会随着可重配置硬件技术的部署发生明显变化，直至企业停止把基础架构当作商品对待，而是将其当作业务价值链中的关键一环来使用。

三大内在属性支撑着提议的基于可重配置硬件的神经塑性云计算基础架构： 高性能（以更快速度得到结果）、灵活性（经优化的计算，采用完全适配将要运行的特定应用的架构）和安全性（从设计提供的数据隐私、加密和防范网络安全威胁的保护）。综合运用这些特征有望促成数字业务时代的突破性创新。

高性能计算走向云端

高性能计算 (HPC) 基本上是指使用并行处理功能高效、可靠、迅速地运行高级应用程序。过去高性能计算的使用权局限于学术界、工程师和科学家。实际上，高性能计算几乎成为超级计算中心的代名词，用于处理理论物理模型仿真的复杂计算。高性能计算的典型应用领域包括气候建模、碰撞仿真和生物信息学。这些领域在性质上就是高计算强度的。

云计算让高性能计算走入寻常百姓家。随着时间推移，对这一计算能力的需求正在超越传统的超

级计算中心，延伸到公有云、私有云、混合云、社区云乃至个人云中，用户通过便携式甚至是可穿戴嵌入式设备就能够使用。

对许多服务和应用来说，这种对计算能力不断增长的需求意味着严格的响应时间要求，要求提供商使用最先进的技术更新自己的计算平台。在实时图像处理、视频流和大数据分析等这些应用中，降低时延，加快得到结果都是最终用户看重的 QoS 要素。大公司、创业企业和中小型企业 (SME) 逐渐认识到，先进的计算基础架构能带来竞争优势。随着 HPC 进入主流，云计算通过实现对无限计算资源的共享弹性访问，在 HPC 交付方面正起着关键作用，尤其是对已经使用集群和网格计算的工程和科学应用而言。

鉴于云基础架构工作负载对计算能力、灵活性和电源效率提出了更高的要求，基于 FPGA 的首创替代方案能提供超越行业需求的高性能解决方案。借助可重复配置硬件技术，以并行处理支撑的计算平台有望提供快捷平衡的解决方案，尤其是在要求以极快速度处理实时数据的应用中。为了显著降低时延，FPGA 厂商与主要的利益相关方密切协作，通过将综合在 FPGA 逻辑中的加速器连接至现有处理器，正设法实现高性能、更高能效的数据中心。

数据中心中的灵活计算

随着最终用户逐渐习惯和依赖把自己的个人数据托管到云端，预计云计算将成为 IT 交付的默认方式。与之相关的一个趋势是把产品转化为服务，分解成能够在按使用付费业务模式中重新组合以准确满足客户需求的单元。IaaS 就是一种把云计算硬件基础设施（服务器、存储、网络和操作系统）作为互联网上的按需服务交付的途径。许多企业意识到云硬件基础设施对他们的业务不可或缺，但他们把它当作不会给自己带来任何竞争优势的商品对待，因为他们的竞争对手基本上使用相同的技术提供基本上相同的 QoS。

可重配置硬件技术能颠覆这样的模式。该技术带来极具竞争力的性能功耗比提升，降低总体成本，并用作可扩展的可重配置加速平台，能够对任何工作负载进行按需优化，从而实现明显的差异化。配备 FPGA 器件的数据中心服务器能支持可针对特定计算优化的硬件软件计算平台。灵活的硬件是让最终用户用上与云基础架构紧密挂钩的丰富特性的关键，比如设计保障的专用高性能计算。

有几大因素有利于向云端的灵活计算转型。FPGA 的并行处理功能是一个明显优势。此外，通过 FPGA 在云端集成异构硬件资源，为无需依赖持续的 CPU 性能提升就能改善计算效率提供了机遇。

另外，为了增加任何特定计算任务的吞吐量，往往可以在 FPGA 资源中对其实现方案进行流水线操作。借助灵活的硬件对最终用户应用进行精细流水线，可交付最佳性能的定制解决方案。

接着，还可以发挥 FPGA 中硬件资源的部分可重配置性，在运行中把不同的定制协处理器换入和换出特定资源，从而以多路复用的方式动态地按需计算所需应用算法中计算强度最大的部分。这种方法基本不会影响执行时间，同时提供了一种能权衡面积与性能的低成本解决方案。

总之，定制云计算能为一定数量领域的提供商

采用基于云的 IaaS 的可重配置硬件不仅可以扩大云计算的供应，提高定制化水平，还能增强网络安全性。

和从业者带来重大差异化。例如，采用特定计算机架构而非标准架构支持的金融计算应用能加快经纪人对金融趋势的分析，这样他们买入或卖出股票的速度就能比他们的竞争对手更快。如果负责处理所需的实时处理算法的计算机对执行这些算法进行了专门的优化，医疗人员就能在开展远程外科手术的过程中提高成像质量并缩短响应时间。在卷积神经网络上的大数据分析和网络游戏等其他计算领域中，QoS 以时延和开销为衡量指标，高 QoS 能产生显著的竞争优势。

值得信赖的云计算

网络安全和数据保隐私挑战是云计算得到广泛采用的主要障碍。因为云基础架构在不同程度上一直是一种开放和共享资源，它也成了内外部人员恶意攻击的对象。而且今天在共享资源上执行特定计算或存储敏感数据带来的安全影响让云计算难以成为关键应用的选择。在当前的云平台和基础设施中都已经观察到旁路攻击、身份绑架和恶意代码分配。值得信赖的云计算解决方案可以避免这些问题，因此对实现任何人能够随时随地使用的定制化云计算而言十分重要。

采用基于云的 IaaS 的可重配置硬件部件可以扩大云计算的使用和定制化水平，还能增强相对于软件解决方案的安全性。从设计上说，FPGA 器件与云端传统使用的软件解决方案相比能提供明显面积更小、防御更周密的攻击面。基于 FPGA 的云计算基础架构的设计人员能够让安全性满足云系统架构的最高设计标准。下列因素有利于值得信赖的硬件增强型云计算。

- 防范篡改的硬件安全原语和保护 FPGA 提供物理不可克隆功能 (PUF) 等特定安全特性，可作为为每个电路提供唯一标识符的关键存储的替代机制。FPGA 也适用于实现创建加密密钥所需的真随机数生成器 (TRNG)。
- 加密算法的硬件实现高级加密标准 (AES) 和椭圆曲线密码算法 (ECC) 就是在硬件中执行的加密算法的例子。这些算法的加密原语（旋转、与或运算等）与 CPU 上的顺序软件执行相比，更适合于部署在 FPGA 硬件中。例如 AES 就可以分解为一套顺序执行的阶段或步骤，每个阶段内部分解为一个基本运算循环。这些循环通过使用并行执行在硬件上展开，可以以更快的速度运行。此外，步骤可以流水线化来提高性能。这些技术优化了硬件中的加密算法的综合。
- 随时数据保密（移动中数据、使用中数据和闲置中数据）在开发安全解决方案时性能降低是一个关键性的问题。硬件解决方案相对于软件解决方案更具优势，原因是硬件能够在基本没有开销的情况下完成低时延数据加密和解密。通过这种方法，应用通过云管理的一切信息都能够以加密方式发送、接收和存储，例如用加密文本取代明文文本，防止其受到网络攻击。
- 硬件防火墙硬件安全模块能过滤所有通过系统通信总线的数据，加强数据抵御特定类型攻击的耐受力。

要挖掘云的全部潜力，为今天熟悉技术的用户提供服务，需要能够发挥可扩展、定制化计算、联网和存储资源集的作用的新业务模式。

- 数字签名硬件方法还支持用户认证和可验证的认证和证书管理，从而实现可信根(RoT)。

其他优势

行业用例说明了基于硬件的云解决方案如何支持安全、高性能和灵活性等关键属性。这包括 Google Project Vault。该项目把在硬件中综合的加密计算嵌入到微型 SD 设备中。微软的 Azure SmartNIC 基于 FPGA，用于在服务器中从 CPU 卸载软件定义的联网功能。微软的 Catapult Project 则借助 FPGA 技术加快 Bing 搜索引擎的速度。Bitfusion 的 Cloud Adaptor 项目让开发人员得以在云中使用 FPGA。FPGA 技术已经可用于证明数据中心服务器中的神经可塑性云计算概念。赛灵思 Kintex® UltraScale™ FPGA 和赛灵思 Zynq® Ultrascale+™ MPSOC 器件系列都是有效的例子。

其他有利于基于 FPGA 的云计算基础架构的因素有：

- 可扩展性　云计算通过发挥数据中心的规模经济，正在带来明显的成本节约。基于 FPGA 的云计算解决方案的扩展远远比基于 CPU 和 GPU 的解决方案容易。
- 低功耗 在数据中心环境中，与原始性能相比更重要的是性能功耗比。数据中心要求高性能，但这种高性能的功耗特性应不超越数据中心服务器的要求的限值。与市场上的其他替代方案相比

基于 FPGA 的解决方案能提供高得多的性能功耗比。很明显，最大化性能功耗比是提高数据中心可靠性和控制运营成本的必备要求。

- 环境友好性 延伸降低功耗的优势，基于 FPGA 的云计算正在兴起，成为降低计算的碳排放的必行之道。
- 冗余 通过使用 FPGA 技术管理异构硬件资源，开发人员得以综合出满足特定冗余要求的定制解决方案。

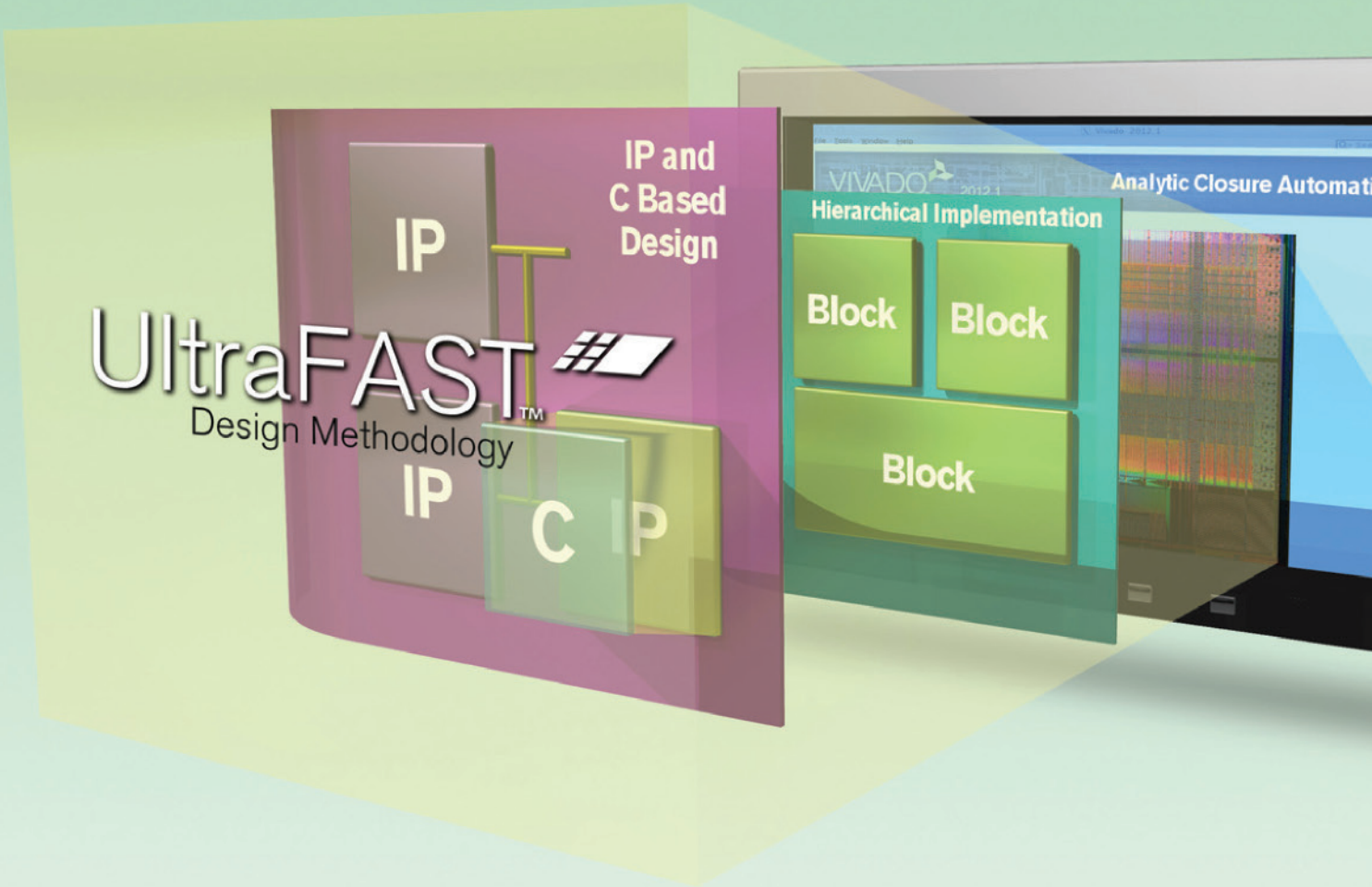
IT 服务的用户对下列要求缺一不可： 移动性、连接性、对信息的即时访问、立即得到计算结果和设计安全。云计算为企业提供了把任务从他们的本地基础架构移到远程经优化的计算集群的手段。要挖掘云的全部潜力，为今天熟悉技术的用户提供服务，需要能够发挥可扩展、定制化计算、联网和存储资源集的作用的新业务模式，从而为所有的客户创造价值。

神经可塑性云计算在配备异构资源的精细粒度 FPGA 器件之上将高性能架构和受信任计算架构融为一体，能改善计算的能力、灵活性和安全性。硬件神经可塑性通过为云端已确立的连接和移动特性添加个性化、个人定制计算功能，掀起人们开展业务方式的革命。

从利益相关方正在积极采取的行为来衡量，可重配置硬件技术迅速融合到云端并不遥远。运行在数据中心服务器 CPU 上的软件代码与直接在硬件中处理的应用关键环节的结合，将实现为最终用户带来显著竞争优势的技术差异化。■

赛灵思推出

Vivado® 设计套件的 UltraFast™ 设计方法



赛灵思的 UltraFast™ 设计方法可加速设计进程并可预测设计周期。

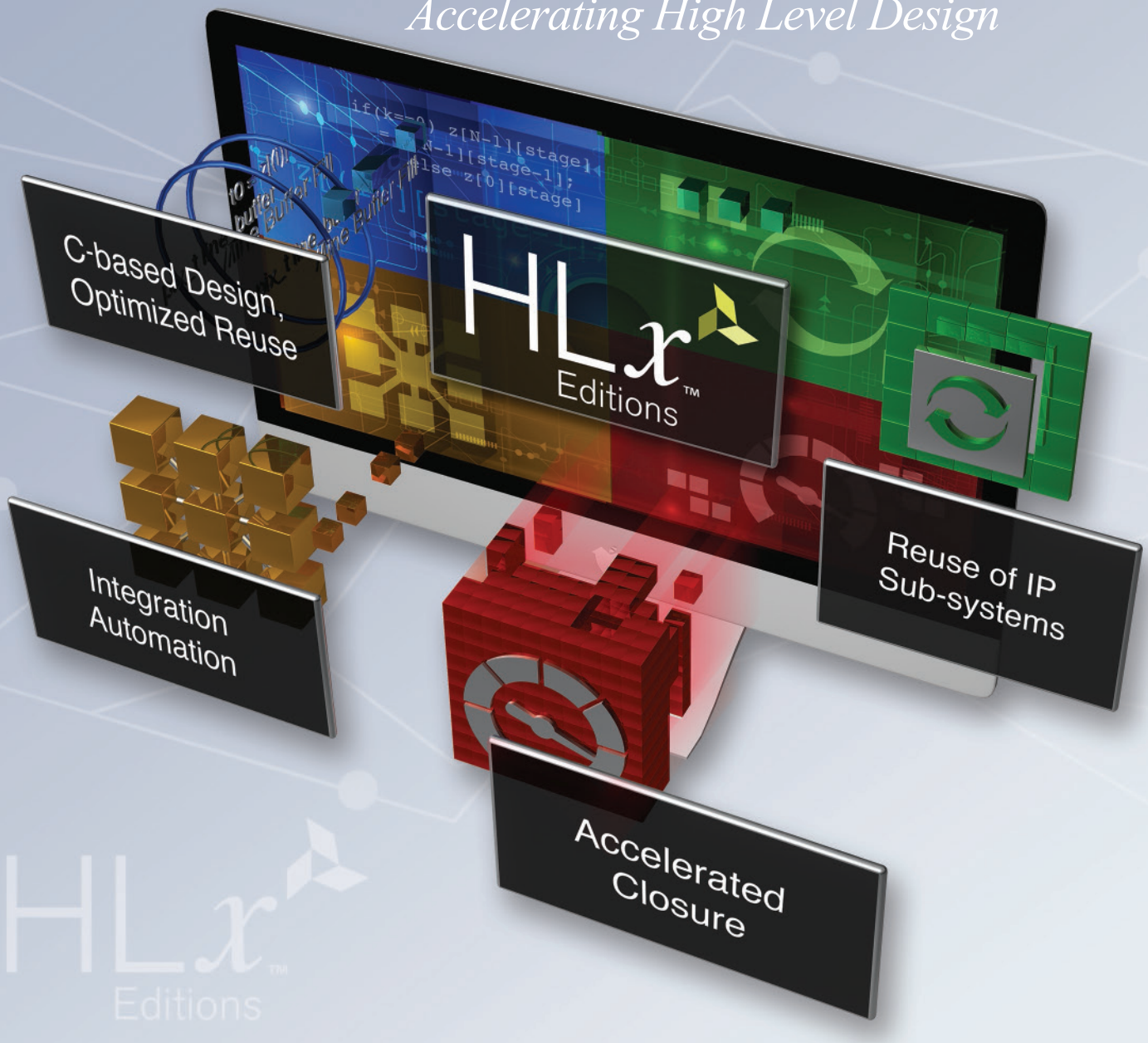
VIVADO

XILINX

A Generation Ahead

The Industry's First
High-Level Design Editions

Accelerating High Level Design



XILINX
ALL PROGRAMMABLE™