基于 FPGA 原型设计能为您做些什么?

最大的优点可能是当首个 SoC 芯片进入实验室时就可立即运行嵌入式软件。

作者: Doug Amos
业务开发经理
解决方案营销部
Synopsys 公司
damos@synopsys.com

Rene Richter
公司应用工程经理
硬件平台部
Synopsys 公司
Rene.Richter@synopsys.com

Austin Lesea
首席工程师
赛灵思实验室
Austin@xilinx.com

Xcell 期刊非常荣幸地在第二章全文介绍一本令人振奋的新书《基于 FPGA 的原型方法手册（FPMM）》。这本书由 Synopsys 公司的 Doug Amos 和 Rene Richter 以及赛灵思内部员工 Austin Lesea 共同编著。《基于 FPGA 的原型方法手册》是一部“为原型设计”的综合性实用指南。如欲详细了解《基于 FPGA 的原型方法手册》一书并下载它的电子书，敬请访问：www.synopsys.com/fpmm。纸质书籍可从Amazon.com网站、Synopsys 出版社和其他书店购买。虽然这本书讲的是使用赛灵思和 Synopsys 产品的实例，但其方法和经验教训适用于任何原型项目，不管使用的是什么工具或者什么 FPGA。

作为基于 FPGA 原型方法的拥护者，有人可能会认为我们只片面地看到了这种方法的优点，而对其缺陷视而不见。但那绝非我们的本意。我们这本《基于 FPGA 的原型方法手册》旨在全面揭示基于 FPGA 的原型的利弊，因为最终我们并不想看到有人本来可以使用其他方法更好地达到目的（比如说用基于System C的虚拟原型），却行进在这种方法的漫漫征途上。

让我们来更深入地研究一下基于FPGA原型方法的目的和局限性，以及其对于系统级验证和其他目的的适用性。把重点始终放在原型项目的目的上，让我们在平台、IP 使用、设计导出、调试及其他设计方面更容易地做出决策。这样，我们就能够通过分析世界上其他原型设计团队的案例，从他们的项目中汲取经验。

基于 FPGA 的原型可满足不同目的需求

原型设计不是一个按几个按钮就能完成的过程，在它不同的阶段需要仔细的关注和思考。除说明这个过程需要完成的工作和涉及到的专业知识外，我们还应解释在 SoC 项目中该进行（或者不该进行）原型设计的原因。

在与原型设计人员多年交谈中，我们最常问到的一个问题是“为什么您这么做？”。答案有多种多样，我们把它们总结成了表 1 中几条常见的理由。举例来说，“真实环境中的数据效应”，这可能指的是某个团队的工作是通过原型设计得到某个系统全速运行时的模型，并将其与其他系统或外设相连，目的可能是为了测试对某个新接口标准的合规情况。他们进行原型设计的大致理由是“与真实环境接口”，而且原型设计也确实在真正的芯片器件面世之前，提供了实现这个目的的最快、最准确的途径。

系统了解这些项目的目的和我们进行原型设计的原因，将有助于我们判断基于 FPGA 的原型设计是否能为我们的下一个项目提供帮助。

因此，让我们探究一下表 1 所述的目的以及基于 FPGA 的原型方法如何能帮助实现这些目的。在许多情况下，我们还会给出真实环境中的一些实例，笔者藉此提前感谢那些奉献自己经验、指导他人走向成功的人士。

高性能与准确度

只有基于 FPGA 的原型才能提供正确测试设计各个方面所需的速度和准确度。我们把这个理由放在首位的原因是，虽然项目有许多需要实现的给定目的，但对需要进行原型设计的团队来说，这可能是所有理由中最根本的原因。举例来说，这个团队的目的可能是验证某些 SoC 的嵌入式软件，观察其在真实硬件上全速运行的情况，但使用原型的根本原因是为了确保高性能与准确度。我们在虚拟系统中可以在更高的性能水平下验证该软件，但我们无法达到使用真实的 RTL 所能实现的准确度。

实时数据流

难以验证 SoC 的原因之一是因为其状态取决于许多变量，包括其之前的状态、输入的次序以及更广泛的 SoC 输出系统效应（以及可能的反馈）。将 SoC 设计与系统的其他部分相连并以实时速度运行，可以让我们立即观察到实时条件、输入和系统反馈的变化带来的效应。

葡萄牙波尔图市 Synopsys 公司 IP 团队开发的 HDMI 原型中的实时数据流就是一个很好的例子。在本实例中，高清(HD)媒体数据流经处理内核的原型输出到高清显示器上，如图 1 的方框图所示。注意方框图底部显示的是实时音频和高清视频数据流，从接收器（从外部源）接收，流经原型，输出到与外部监控器相连的实时 HDMI PHY 的整个流程。

通过使用投片前的原型，我们可以立即看到和听到不同的高清数据在我们的设计上的效果，反之亦然。只有采用基于 FPGA 的原型方法才支持这种实时数据流，不仅给此类多媒体应用带来极大好处，也能给许多其他要求对输入数据流做出实时响应的应用带来诸多裨益。

项目目的	进行原型设计的原因
测试实时数据流 早期软硬件集成 早期软件验证	高性能和高精度
测试真实环境中的数据效应 测试实时人机界面 调试稀有数据依存关系	与真实环境接口
可行性（概念验证） 测试算法	实验室内使用
公开展览 鼓励投资者	实验室外演示
扩展 RTL 测试与调试	其他目的

表 1　采用基于 FPGA 原型的常见目的与原因

软硬件集成

在上述实例中，读者可能已经注意

到原型使用了一块小型 MicroBlaze™ CPU，并备有外设和存储器，从而体现了一个 SoC 的所有常见模块。在这个设计中，运行在 CPU 上的软件主要用于加载和控制 A/V 处理。然而，在许多 SoC 设计中，软件最耗精力。

鉴于软件已成为 SoC 开发工作的主体部分，软件工作在项目日程中占据关键位置越来越常见。当 SoC 能够有效达到量产标准的时候，决定项目实际完成日期的是软件开发和验证工作。在这种情况下，系统开发团队如何才能提升软件开发和验证工作的效率呢？要回答这个问题，我们需要查看软件开发团队把时间都花在什么地方。

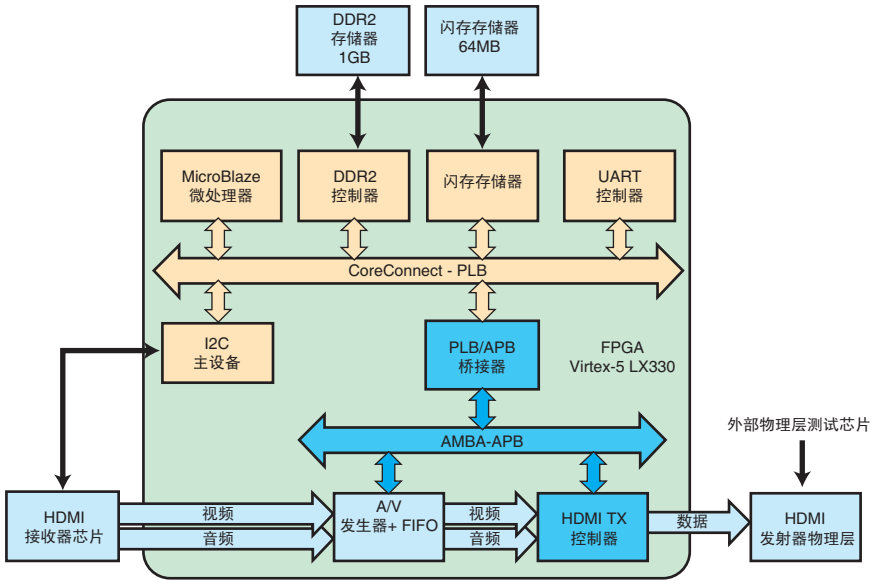

图 1　HDMI 原型方框图

为软件开发建立 SoC 的模型

软件由于自身的复杂性，很难做到完美。对我们在日常使用计算机的过程中遇到的软件升级、服务包和漏洞修补的情况，我们都已经司空见惯。但是，具体到嵌入 SoC 中的软件，这种无休止的软件改进方法就遇到了障碍。另一方面，相比于通用的计算机软件而言，与嵌入式软件互动的系统，其设定的使用模式和环境条件都更容易确定。而且，为较简单的系统开发的嵌入式软件可以比较简单，也就更易于全面验证。举例来说，控制车辆子系统或电子玩具的 SoC 比在实时操作系统 (RTOS) 上运行许多应用和流程的智能手机更容易进行全面测试。

如果我们更加仔细地观察运行在这类智能手机上的软件，例如图 2 所示的 Android 软件，我们可以看到一种多层布置，这称为软件协议栈。（该图基于软件设计师 Frank Abelson 在其所著的《Unlocking Android》一书中的原始图。）

在观察软件协议栈时，我们会发现，协议栈的最底层——也就是那些最接近硬件的部分，主要是为了满足将软件映射到 SoC 硬件上的需求。这就需要对硬件有绝对的了解，甚至包括地址和时钟周期等。软件协议栈最底层的设计人员往往称自己为平台工程师，他们的工作就是准确描述硬件，以便协议栈的更高层次能够识别和重复使用。这种描述被某些 RTOS 厂商称为板支持包 (BSP)，与我们日常使用的 PC 的基本输入/输出系统 (BIOS) 类似。

协议栈从下往上第二层包含 RTOS 的内核以及将较高层次的软件与被描述的硬件相连的必要驱动程序。在这些协议栈的最底层中，平台工程师和驱动程序开发人员需要在真实的 SoC 或完全准确的SoC 模型上验证他们的代码。这个层面的软件开发人员需要全面了解各时钟周期软件的行为。

作为软件开发人员的另一极，在协议栈的顶层，我们可以看到用户空间，在这里可以同时运行多个应用，比如像智能电话中的联系人管理器、视频显示器、互联网浏览器和实际呼叫的电话子系统。这些应用中的每一个都不能直接访问 SoC 硬件，而且实际上在某种程度上违背了所有硬件考虑事项。这些应用依赖运行在协议栈较低层的软件代表自己与 SoC 硬件及系统其他部分通信。

我们可以归纳为：在协议栈的每一层，软件开发人员只需要一个足够准确的模型来让自己的代码认为自己运行在目标 SoC 上即可。超过必要的准确度只会让模型在模拟器上的运行速度下降。实际上，任何层面的 SoC 建模，都要求我们把硬件和协议栈描述为比当前层面更低的一层，以便进行验证。而且在理想的情况下，我们应该只要求够用的准确度，以实现最高性能。

举例来说，协议栈顶层的应用开发人员可以在真实的SoC或 SoC 模型上测试代码。在这种情况下，模型的准确度只要能够让应用认为自己运行在真正的 SoC 上就足够，它不需要精确到时钟周期，也不需要了解硬件的细致结构。但这里速度非常重要，因为在许多情况下有多个应用会同时运行，并与真实环境中的数据接口。

这种只为软件层提供“够用的准确度”的建模方法为不同的软件开发人员提供了多种不同的建模环境，供他们在

SoC 项目的不同阶段使用。可以采用 SystemC 这样的语言进行事务处理层面的建模，创建出一个准确度低但速度足够快的仿真器模型，用来同时运行许多应用。如果实时的真实数据的处理不是很重要，那么考虑采用虚拟原型方法比较好。

不过，必须完整运行整个软件协议栈或必须处理真实环境中的数据时，最适合采用基于 FPGA 的原型方法。

使用原型验证软件的实例

只有采用基于 FPGA 的原型方法才能够打破建模方法中准确度与性能之间内在的相互牵制关系。采用 FPGA，我们既能实现实时的速度，又能以完全的 RTL 周期精度建模。这样，单个原型不仅能供低层软件验证要求的准确模型使用，又能供高层应用开发人员需要的高速模型使用。实际上，整个 SoC 软件协议栈都可以在单个基于 FPGA 的原型上建模。德克萨斯州奥斯汀市 Freescale Semiconductor公司移动产品部的 Scott Constable 及其团队开展的项目就是采用 FPGA 验证软件的一个很好的例子。

Freescale非常想加快 SoC 开发进程，因为手机市场上产品生命周期短，需要产品尽快打入市场。这不仅是为了赢得竞争，也是为了避免迅速过时。通过分析流程中耗时最多的环节，Freescale发现通过加快手机3G协议测试工作可以带来最明显的效果。如果测试工作能够在流片前完成，Freescale 就可以将项目时间缩短数月。与通常只有一到两年的产品生命周期而言，这非常重要。

协议测试是一个复杂的过程，就算以较高的实时速度进行，也需要一天才能完成。使用 RTL 级仿真需要花上数年，而在较快的仿真器上运行也要数周时间，这都不切合实际。采用 FPGA 是因为这是实现必要的时钟速度，及时完成测试的唯一途径。

协议测试需要开发产品的各种软件特性，包括硬件驱动程序、操作系统和协议栈代码。虽然如前所述主要的目的是协议测试，通过使用 FPGA，所有这些软件开发工作都能够在流片前完成，从而大大加快各种最终产品的开发进度。

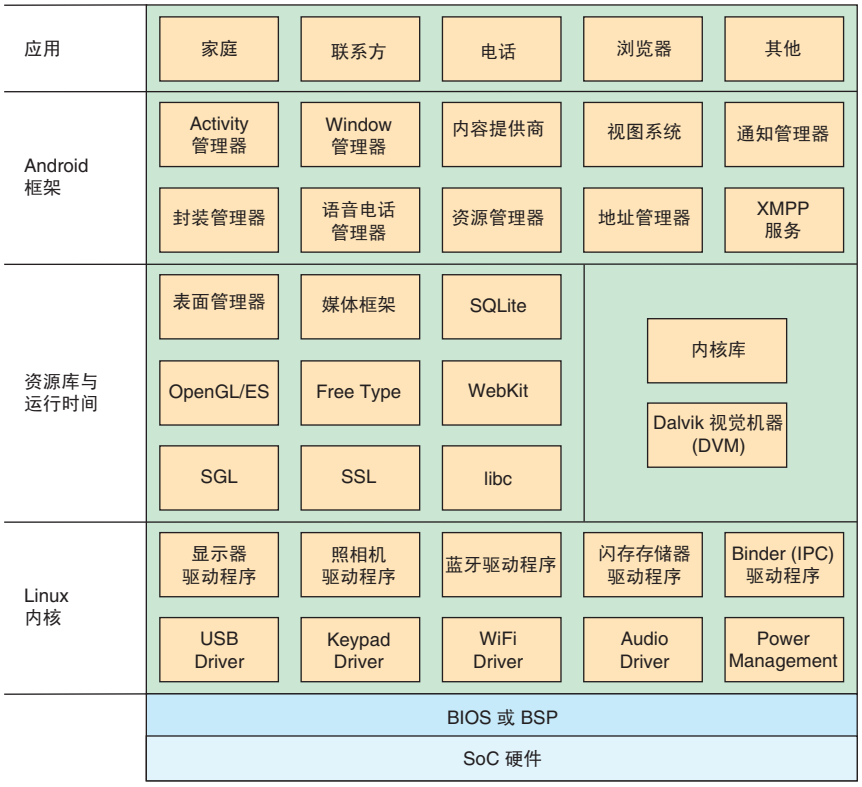

图 2　Android 软件协议栈

Freescale 构建了一个多芯片系统的原型，其中包括一个双核 MXC2 基带处理器和一个 RF收发器芯片的数字部分。基带处理器内置一个用于调制解调器处理的 Freescale StarCore DSP 内核、一个用于用户应用处理的ARM® 926 内核，以及 60 多个外设。

Synopsys HAPS-54 原型板用来实现原型（如图 3 所示）。该基带处理器有 500 多万个 ASIC门，Scott 的团队使用 Synopsys Certify 工具将其在原型板上划分给三个赛灵思 Virtex-5 FPGA，同时把数字 RF 设计布置在第四个 FPGA 中。Freescale 决定不构建模拟部分的原型，而是直接从 Antritsu 协议测试盒中以数字形式提供移动网络数据。

较早的内核使用的某些设计技术对 ASIC 来说非常有效果，但对 FPGA 来说却不太好用。另外，RTL 的一部分是从系统级设计代码中自动生成的，由于其过于复杂的时钟网络，对 FPGA 来说也是相当不利。因此，必须对 RTL 进行一些调整，使其更加兼容 FPGA，这样做的成效非常显著。

除了能够加快协议测试，到 Freescale 的工程师完成首个芯片时，

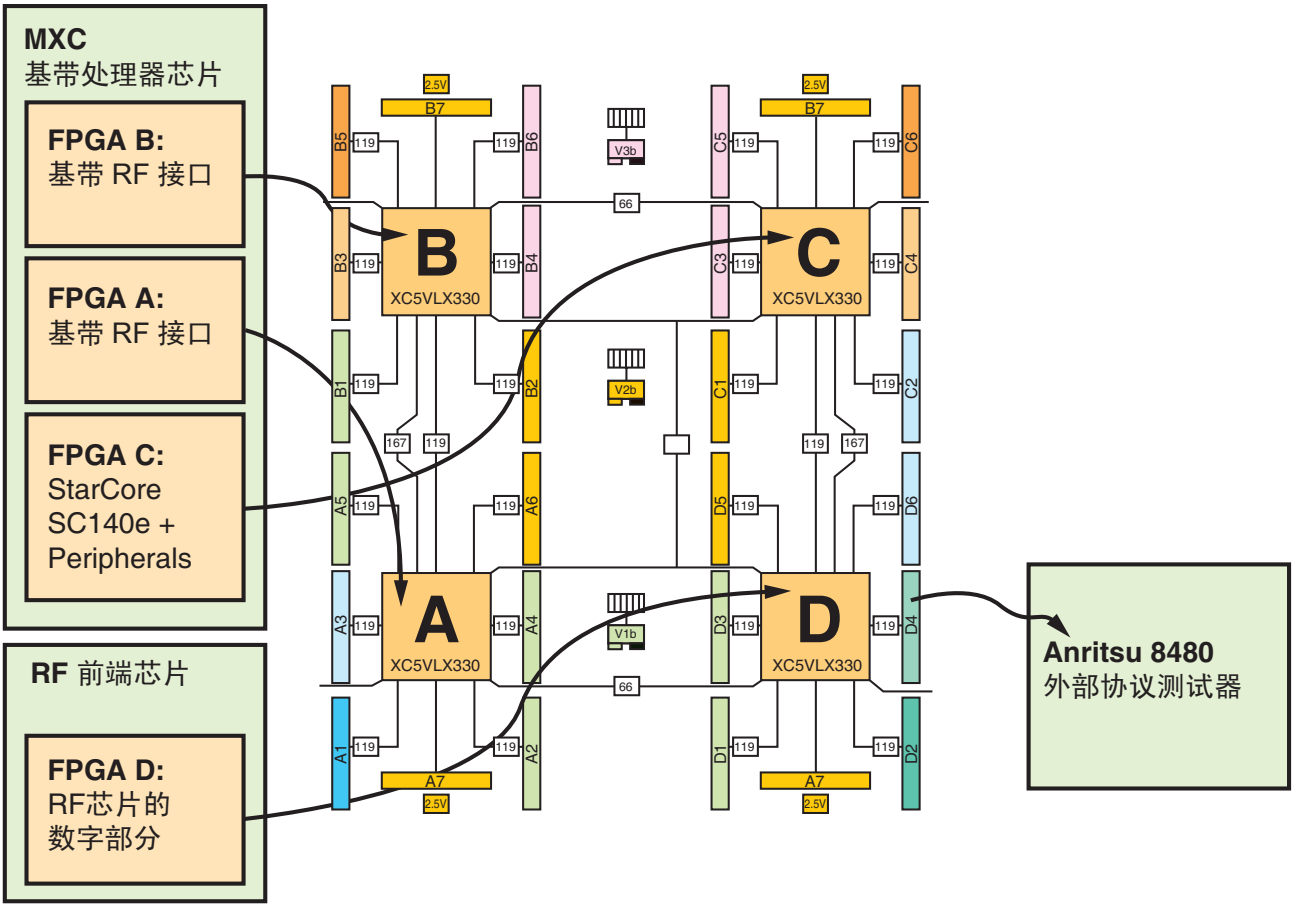

图3　Freescale 的 SoC 设计在 HAPS-54 原型板上的分区

他们还能够：

- 发布调试器软件，且在芯片实现后无需进行大的修改；
- 完成驱动程序软件；
- 在操作系统提示符下引导SoC；
- 实现调制解调器阵营(modem camp)和注册。

仅在完成首个芯片后一个月，Freescale 团队就成功地从这个系统中拨出了第一个移动电话呼叫，把产品开发进度缩短了6个多月，这非常具有里程碑意义。

如 Scott Constable 所说，“除了加速我们所述的协议测试目标外，我们的 FPGA 系统原型还加快了众多其他领域的项目进度，再次证明了它的价值。也许最重要是给开发人员带来的不可估量的好处：它能够让工程人员尽早参与项目，让从设计到软件到验证再到应用的所有开发团队在芯片完成前半年时间内就对产品了如指掌。这种对产品专业知识认知的进程的加快是难以用甘特图(Gantt chart) 来衡量的，却可能是最有益处的。”

“鉴于如此多的优势，用基于 FPGA 的原型解决方案来加快 ASIC 的开发进程是一件非常自然的事情。我们随后将这种方法介绍给了 Freescale 网络和微控制器部，还将其用于最新 IP 验证、驱动程序开发、调试器开发和客户演示。”

这个例子说明基于 FPGA的原型方法能够给软件开发团队提供什么样的增值工具，能够在产品质量和项目进程方面带来怎样显著的回报。

接口优势：测试真实条件下的数据效应

很难想象有这样一种 SoC 设计可以不遵守输入数据、处理数据、生成输出数据的基本结构。实际上，如果我们深入 SoC 设计，就会发现无数的子模块遵循着同样的结构，直到单个门级。

要在这些层级中的每一个层级验证正确的处理，要求我们提供完整的输入数据集，并观察处理结果的输出数据是否正确。对单个门来说，这个工作很简

单，对小型 RTL 模块来说，也是可能的。但随着系统日趋复杂，从统计上来说基本没有可能确保输入数据和初始条件的完整性，尤其是在有软件运行在一个以上的处理器的时候。

为了提升传统验证方法的效率和覆盖面，以及应对这种复杂性所带来的挑战，我们已经投入了大量的研究工作和资金。在完整的 SoC 层面，我们需要使用多种不同的验证方法来覆盖所有输入组合并和剔除极端情况下的组合。

最后一点非常重要，因为不可预测的输入数据能扰乱所有的 SoC 系统，即便是精心设计的关键SoC设计也难以幸免。与新输入的数据或者输入数据不寻常的组合或序列相结合的，是非常多的 SoC 可能的前置状态，可能会使 SoC 处于某种无法验证的状态。当然，这种情况不一定是什么问题，SoC 可以在无需系统的其他部分干预的情况下恢复，或者用户根本就没有察觉。

但是，不能验证的状态必须在最终芯片中避免，因此我们需要尽可能全面地测试设计的方法。在设计的功能仿真过程中，验证工程师会采用有力的方法，比如受约束随机激励和高级测试工具来完成多种测试，旨在达到可接受的测试覆盖面。但是，完整性仍受验证工程师选择的方向和给定的约束条件的限制，并受限于可用于运行仿真的时间。结果虽然受约束随机验证永远不可能穷尽，但能够大大增强我们已经测试了所有输入的组合（包括可能的输入和极端情况输入）的信心。

为了屏蔽极端情况下的组合，我们可以通过观察真实条件下运行在基于 FPGA 原型上的设计来完善我们的验证工作。通过将 SoC设计植入原型，我们能够以与最终芯片媲美的速度和准确度来运行设计，从而在最终的环境数据中进行“浸入式”测试，就如在最终芯片上进行的一样。

西班牙瓦伦西亚的 DS2 采用基于 FPGA 的原型将 SoC 设计浸入到真实环境中就是一个很好的例子。

实例：浸入到真实环境数据中

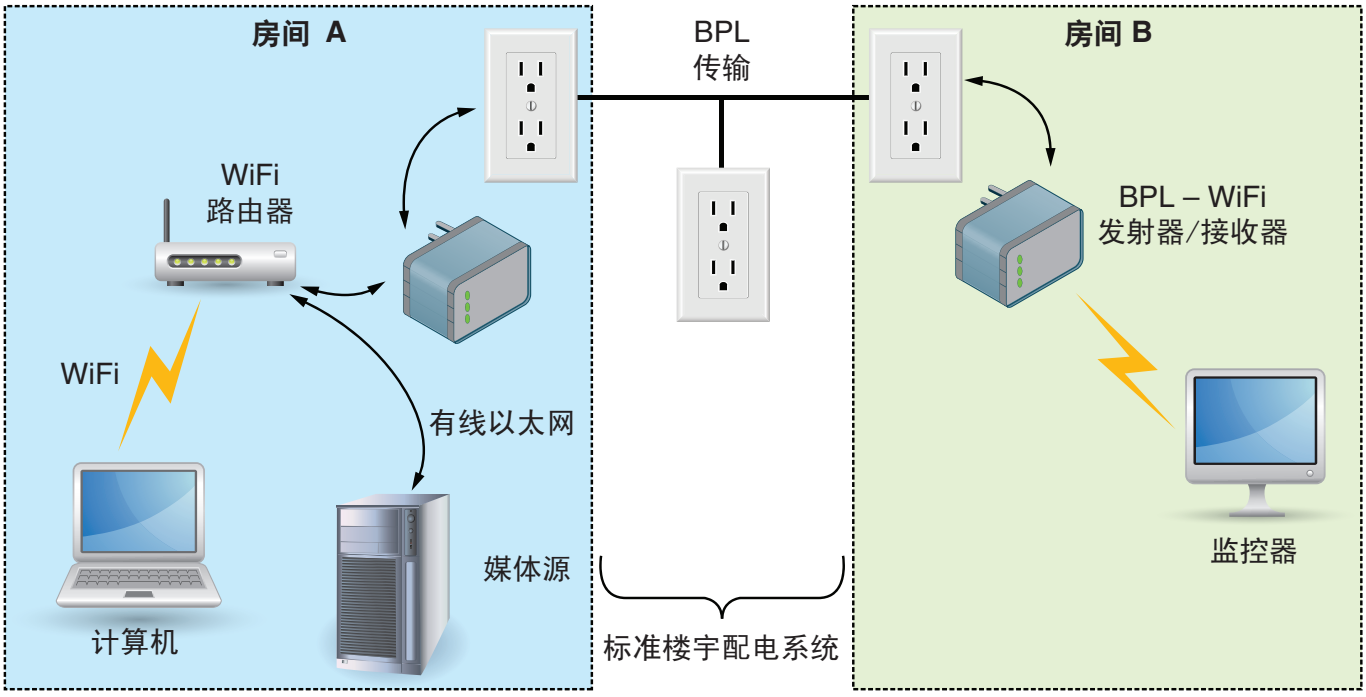

图4　WiFi 范围扩展器采用的电力线宽带 (BPL) 技术

电力线宽带 (BPL) 技术一般采用无法检测的信号来通过输电线发送和接收信息。BPL 的一种典型应用是将高清视频从接收器通过输电线传输到室内的任意一台显示器，如图 4 所示。

DS2 的 BPL 设计核心是复杂的硬件和嵌入式软件算法，其可对电力线输入输出的高速传输信号进行编码和检

索。这些电力线可能工作在噪声极高的电气环境中，因此开发工作的关键部分是在各种真实情况下验证这些算法。

DS2 的 ASIC 设计经理 Javier Jimenez 说明了基于 FPGA 的原型设计对他们的用处："采用稳健的验证技术开发可靠的高速通信非常重要。它需要采用不同的通道和噪声模型进行大量的尝试，只有基于 FPGA 的原型可帮助我们全面测试算法，在原型上运行设计的嵌入式软件。另外，我们还可将原型拿出实验室，进行广泛的现场测试。我们可将多个原型放在真正的家居和办公环境中，其中部分电气环境非常恶劣。我们不能考虑将仿真器系统用于该目的，因为其价格非常昂贵，也不便携带。"

这种在实验室外使用基于 FPGA 的原型设计非常有指导意义，因为我们明白打造可靠、便携的平台对取得成功非常重要。

对实验室可行性实验的优势

在项目的初始阶段，需要对芯片拓扑、性能、功耗以及片上通信结构做出基本决策。部分决策采用算法或系统级建模工具便可良好执行，但也可以采用 FPGA 进行某些额外的实验。这是否是真正基于 FPGA 的原型设计呢？我们正使用 FPGA 进行某个概念的原型设计，但这与使用算法或数学工具不同，因为我们需要某些可能是由这些高级工具生成的 RTL。一旦进入 FPGA，就可采集早期信息帮助推进算法和最终 SoC 架构的优化。基于 FPGA 的原型为项目该阶段带来的优势是，可使用更准确的模型，而且这些模型的运行速度非常快，能够与实时输入互动。

这种类型的实验性原型值得一提，因为它们是在全面的 SoC 项目中使用基于 FPGA 的原型设计硬件和工具的又一途径，可为我们的投资带来更高的回报。

在实验室外使用原型

基于 FPGA 的原型设计可用于验证 SoC 设计的一个真正独到之处，是其独立工作的能力。这是因为 FPGA 可通过闪存 EEPROM 卡或其它独立介质进行配置，无需主机 PC 管理。因此该原型不但可独立运行，而且还可用于各种环境下的 SoC 设计测试，这与其它建模技术（如需要依赖主机干预的仿真）提供的环境俨然不同。

在极端情况下，原型可以完全从实验室中取出，用于现场真实环境中。比如将原型安装在开动的车辆上，研究设计对外部噪声、移动、天线场强等条件变化的依赖性。比如，本文作者就曾将移动电话的基带原型安装在车辆上，通过公共 GSM 网络在移动中拨打电话。

芯片架构师与其他产品专家需要与早期客户互动，展示其算法的重要特性。基于 FPGA 的原型设计在项目极早期的这个阶段可能是非常关键的优势，但这种方法与主流 SoC 原型设计略有不同。

将基于 FPGA的原型用于实验室外的另一种极为常见的用途是在商业展示会上进行新产品功能的预制造展示。下面让我们来研究一下英国 BBC 下属的研发部门将基于 FPGA 的原型用于实验室外和商业展示会的案例。

实例：现实世界中的原型

FPGA 独立运行的强大功能在英国用来推广 DVB-T2 的 BBC 研发项目中得到了充分展现。DVB-T2 是业界领先的最新开放式标准，其可实现通过地面发射器传播高清电视。

使用基于 FPGA 的原型的原因和大多数国际标准一样，DVB-T2 技术规范的完善花费了数年时间，来自世界各地的研究人员和技术专家用了 3 万个工程小时。只有 FPGA 才能高度灵活地满足开发过程中不断变更的需求。该规范于 2008 年 3 月终稿，并于三个月后在 6 月 26 日以《DVB 蓝皮书》的方式发行。

由于 BBC 在规范制定的同时就已经在使用基于 FPGA 的原型，由 BBC 研发部 Justin Mitchell 带领的 BBC 实施团队就能够为 DVB-T2 开发一种基于硬件的调制器和解调器。

该调制器建立在具有赛灵思 Virtex-5 FPGA 的 Synopsys HAPS-51 卡基础之上。该卡连接至 BBC 研发部设计的子卡上。该子卡提供一个 ASI 接口，可接收输入的传送流。输入的传送流随后传输给 FPGA，按 DVB-T2 标准编码，然后传回子卡，直接上变频为 UHF。

该调制器用于业界首次从现场 TV 发射器传输 DVB-T2 标准信号，时间是该规范发行的同一天。

解调器也采用 HAPS 作为另一种 FPGA 原型的基础，完善了端对端的工作链，并在 2008 年阿姆斯特丹的 IBC 展会上进行了演示，时间是规范最终确定后三个月。这是一项不平常的成就，帮助建立了该系统于 2009 年投入使用的信心。

此外，BBC 研发部还参与了 DVB-T2 项目的其它重要部分，包括 2009 年 3 月在 Turin 举办的非常成功的插拔测试大会。在这次大会上，共有五种不同的调制器和六种不同的解调器亮相，在各种模式下协同工作。BBC 原型稳健的便携式构造使其成为本次插拔测试大会上的亮点。

Justin Mitchell 对基于 FPGA 的原型评论道："FPGA 最大优势之一是能够在从预备阶段到宣布传输的这段时间里跟踪规范的最新变化。根据规范的变动迅速对调制器做出调整的能力非常重

要。很难想出有另一种可如此快速开发调制器与解调器并支持便携性的技术，其可使调制器与解调器独立应用于现场发射器和公共展会。”

基于 FPGA 原型有何不足?

我们撰写本文的目的是公正地看待基于 FPGA 原型的优势与局限性，因此在前面谈及各种优势之后，我们将在下面介绍部分局限性。

首先最重要的是，FPGA 原型不是 RTL 模拟器。如果我们的目的是编写一些 RTL，然后尽快在 FPGA 中实施，以查看它是否能工作，那么我们应该重新思考所忽略的东西。模拟器有两个基本组件，可以把它们考虑成引擎和仪表盘。引擎的作用是激励模型，记录结果，而仪表盘的作用则是帮助我们检验这些结果。我们可以以小幅增量运行模拟器，然后通过我们的仪表盘进行调整；我们可能采用某些非常复杂的激励方式，这基本上都是仿真器的工作。基于 FPGA 的原型能够完成相同的工作吗？当然不能。

FPGA 对运行 RTL“模型”来说确实是一种速度更快的引擎，但当我们开始设置该模型的时候，速度优势就会大打折扣。此外，模拟器的仪表盘部分能够完整地控制激励和掌握结果。我们应该思考仪表化 FPGA 的方法，深入了解设计的功能性，但即便是在这方面最完善的设计，也只能提供一部分真正能用于 RTL 模拟器仪表盘的信息。因此，该模拟器是用于重复编写和评估 RTL 代码更加理想的环境，因此我们应该等到模拟基本完成后，RTL 相当成熟后才能将其交付给 FPGA 原型设计团队。

基于 FPGA 原型不是 ESL

Synopsys 的 Innovator 或 Synphony 等电子系统级 (ESL) 工具或算法工具可在 SystemC 中完成设计，或通过预定义模型库进行构建。然后，我们不但可在相同的工具中模拟这些设计，而且还可深入了解其系统级性能，包括运行软件，在项目初期阶段进行软硬件权衡。

使用基于 FPGA 原型方法，我们需要 RTL，因此它不太适合研究算法或架构，因为这两者通常不采用 RTL 方式表达。对软件来说，FPGA 原型设计的优势是在当 RTL 成熟得可以构建硬件平台的时候，软件可在更加准确以及更加真实的环境中运行。对那些具有天马行空想法的人来说，可以编写少量 RTL 在 FPGA 上运行，进行可行性研究。这是一种极少而又非常重要的 FPGA 原型设计的使用方法，但别把它和整个 SoC 的系统级或算法研究混淆在一起。

持续性是关键

优秀的工程师往往会为其工作选择适当的工具，但应该随时有一种方法可以将半成品交给他人继续完成。我们应该能够在尽量不增加工作量的情况下，将来自 ESL 模拟的设计移交给基于 FPGA 的原型。此外，部分 ESL 工具还可通过高层次综合实现设计，生成 RTL 供 SoC 项目整体使用。基于 FPGA 的原型能够接收该 RTL，并以高周期精度在电路板上运行。但我们需要再次等到 RTL 相对稳定下来，这需要等到项目软硬件分区和架构研究阶段完成后。

采用 FPGA 进行原型设计的原因是什么呢?

当前 SoC 是从算法研究人员到硬件设计人员，乃至软件工程师和芯片布局团队等众多专家的工作结晶，在项目不断发展的同时，各类专家也都有自己的需求。SoC 项目的成功很大程度上取决于上述各类专家所使用的硬件验证、软硬件联合验证以及软件验证的方法，基于 FPGA 原型设计可为每一类专家带来各种不同的优势。

对于硬件团队而言，验证工具的速度可对验证吞吐量产生巨大的影响。在大多数 SoC 开发中，有必要随项目的成熟运行多次仿真，重复回归测试。仿真器和模拟器是用于这类 RTL 验证的最常用平台。然而即便使用基于 TLM 的模拟与建模，由于长时间的运行，RTL 内部或 RTL 与外部激励物之间的部分互动仍然不能在仿真或模拟中重新创建。因此一些团队采用基于 FPGA 原型为这种硬件测试提供具有更高性能的平台。例如，我们可以在近乎实时的条件下运行整个操作系统的引导程序，节省需要花上数天才能达到相同目的的模拟时间。

对于软件开发团队而言，基于 FPGA原型可为目标芯片提供独特的流片前模型，能够在开发接近尾声时高速、高度准确地进行软件调试。

对于整个团队而言，SoC 项目的关键阶段是在软硬件初次结合的时候。硬件将由最终软件执行，而执行方式可能是单纯硬件验证方案难以预见或预测的，从而最终将出现新的硬件问题。这在多核系统中或者在那些运行同步实时应用的系统中特别普遍。如果这种软硬件的采用要等到第一个器件制造完毕后，那么毫不夸张地说，到那时再发现新的缺陷就不太好了。

基于 FPGA原型有助于尽早地将软件引入具有高周期精度的高速硬件模型。SoC 团队经常告诉我们，FPGA 原型设计的最大优势就是在第一个器件上市时，系统和软件都已准备就绪，当天便可运行。