# NI LabVIEW 的前世今生

— 你所不知道的 LabVIEW

## 20 多年的持续创新

20 多年以后，当 LabVIEW 成为了工程师和科学家们的标准图形化设计平台，为其工程创新不断提供源动力之时，LabVIEW 的最初创始人们一定会回想起 1.0 版本诞生时他们对这一革命性的图形化编程环境所抱有的期待和愿景……

1983 年，NI 的工程师们受到了电子制表软件为金融领域带来巨大便利的启发，也决定着手发明一种同样高效的工具，帮助工程师和科学家们简化测试测量自动化项目的开发过程。

与此同时，苹果公司推出的 Macintosh 计算机的一系列图形化特性也为他们提供了崭新的思路。他们发现，相对于输入一串串的命令行进行操作，人们使用鼠标和图形化界面时所发挥的创造力和高效率是前所未有的，因此“图形化”编程理念成为了 LabVIEW 最根本的核心。

LabVIEW 从最初就被设计为一种强大的高层架构型编程语言，自 1986 年 1.0 版诞生以来，纵观其 20 多年的发展（图 1），可以发现，每次 LabVIEW 的主要升级版本的发布都包含了很多全新的特性。

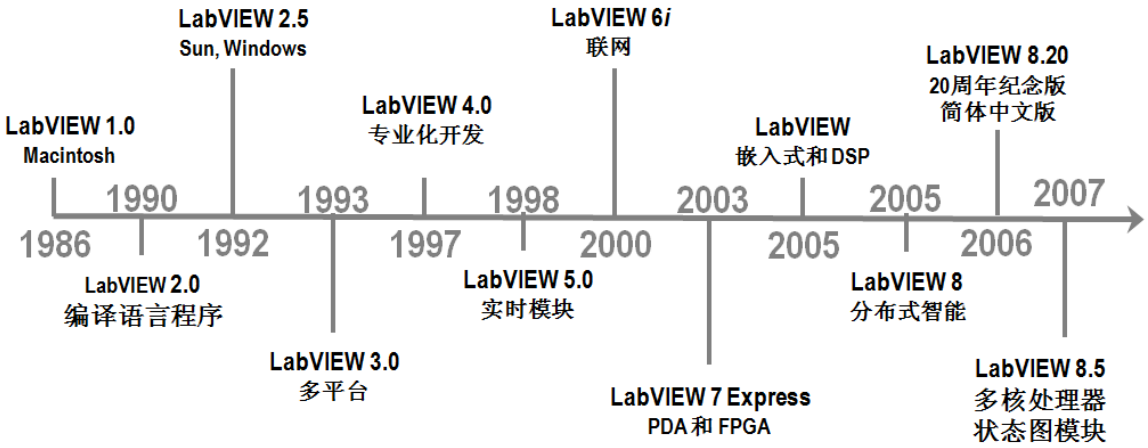


图 1 LabVIEW 20 多年的持续创新

LabVIEW 3.0 首次实现了多平台兼容的特性，保证相同的代码可以运行在多个操作系统中；LabVIEW 5 则推出了实时（Real Time）模块，允许工程师们将在主机上开发的 LabVIEW 代码进行自动编译，使其运行在实时硬件对象中。通过降低在实时系统中部署代码的复杂度，这个创新的理念帮助工程师以一种更方便的方式进行控制应用的开发；而 LabVIEW 7 与 FPGA 技术的结合则又是该理念的进一步升华，从而让不具备 VHDL 编程经验的工程师们也同样可以进行硬件设计，并且 LabVIEW 本质上的数据流并行性非常符合 FPGA 并行电路特性，在此基础上可以达到很好的空间利用和定时性能；LabVIEW 8.2 作为 20 周年的纪念版，首次推出了中文版，使中国工程师们也能用自己的母语编程，最大程度地提升开发效率；最新的 LabVIEW 8.5 则更为多核处理器技术提供了强有力的支持，同时也推出了基于 UML 语言规范的状态图设计模块。综合而言，LabVIEW 通过不断地融入最新商业可用技术（图 2），让使用者无需花费过多的精力去学习每个技术的细节就可以直接使用，提升了系统的性能，保证了工程师们长期的投资。

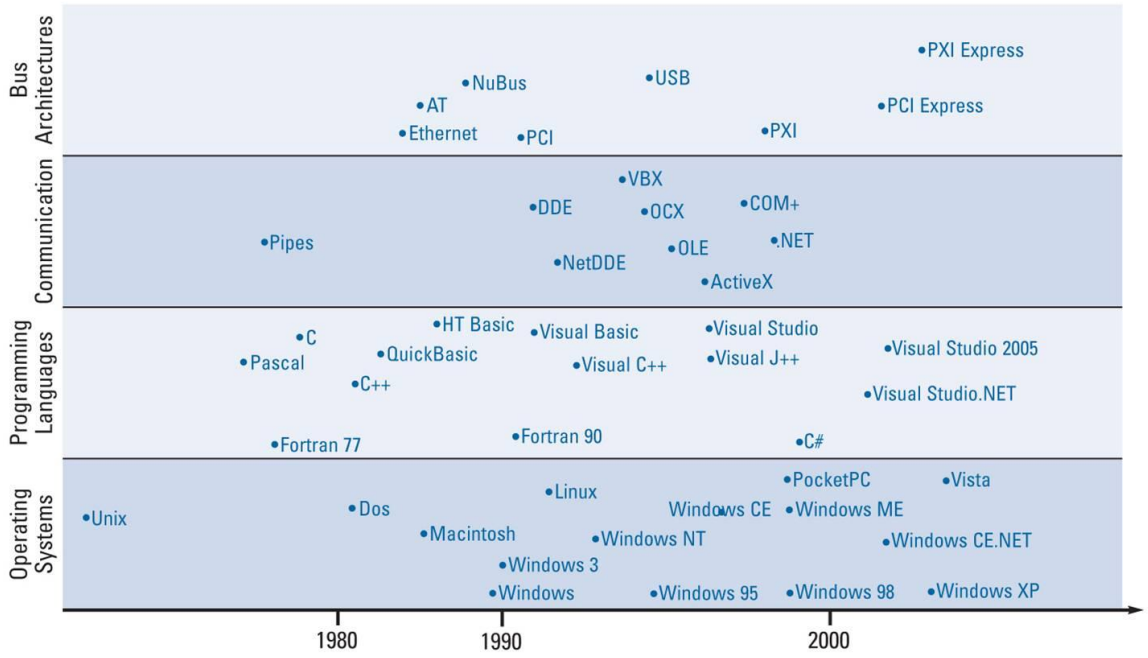


图 2 LabVIEW 不断融入最新商业可用技术

经过这一路 20 多年的持续创新，LabVIEW 凭其核心的图形化编程理念，突破了原先数据采集与仪器控制的应用领域，蜕变为设计、控制与测试的图形化系统设计标准平台，其强大的特性（详全的专业附加工具包、灵活多样的计算模型、从 PC、FPGA 到芯片级的运行平台等）进一步帮助工程师在同一个 LabVIEW 平台上集成从设计、原型到发布的全过程，全面提高整个工程流程的效率。

## 强大的图形化设计平台

### 详全的专业附加工具包

我们知道，作为功能强大的图形化系统设计平台，LabVIEW 所涉及的应用领域变得越来越广泛，因此为了让不同应用领域的工程师们都能以一种更灵活的方式来使用 LabVIEW 进行系统开发，安装附加工具包成为了一个很好的解决方法。

无论是信号处理、自动化测试、工业控制还是嵌入式设计等等，LabVIEW 都提供了专业的附加工具包，从而方便工程师们通过灵活的组合实现高效的开发。

例如，仅仅在信号处理方面，LabVIEW 就提供了声音与振动分析套件（倍频程及阶次分析等）、调制工具包（AM、FM、ASK、QAM 等调制算法）、频谱测量工具包（星座图、I-Q 数字解调等）、数字滤波器设计工具包以及高级信号处理工具包（时频联合分析、小波分析等等）。与其它编程语言有所不同，这些 LabVIEW 工具包将各自专业领域的算法和程序进行了优化的封装，让工程师通过直接调用其中的子 VI（甚至仅仅通过简单的配置）得到相应的分析结果，大大减少了开发的时间和精力。

值得一提的是，除了 NI 官方提供的附加工具包以外，LabVIEW 爱好者们也会共享一些自己编写的小型工具包，帮助 LabVIEW 应用在更多更广的领域。这种类似于 Wiki 百科的大规模协作的发展模式使 LabVIEW 能够以一种更积极、开放的方式不断发展创新。

### 灵活多样的计算模型

计算模型，简而言之，是一种用于描述软件模块功能的表达方式，在学术界这个术语一直被用来抽象定义计算机系统。由于不同的计算模型在不同领域和场合的应用上往往都存在相对的优势和劣势，为了实现图形化系统设计的远景目标，LabVIEW 必须具备使用不同计算模型进行编程的能力，而让我们欣喜的是，这个想法已经逐渐成为现实。

如今的 LabVIEW 已经不再仅限于数据流编程这一种方式，它还包括了可以通过 DLL 将 C 或 Java 等文本语言直接调用，使用 Mathscript 节点实现文本数学编程，在 LabVIEW 下进行仿真建模以及使用基于 UML 规范的状态图实现高抽象层的系统架构等等。这些灵活多样的计算模型允许工程师们根据不同的应用领域选择最为合适的一种进行开发，一方面能够让工程师们使用他们熟悉的计算模型进行开发，另一方面，又可以充分利用其它计算模型的优势和特性，实现系统级开发的效率最优。

例如，LabVIEW 用户在设计一个激光控制系统时，可以使用状态图来定义状态，使用数据流方式在 FPGA 芯片中实现控制逻辑，并使用仿真模型来对激光进行动态仿真。可

见，这种“采各家之长”的理念，让 LabVIEW 甚至超越了编程语言的范畴，成为了更高层的系统级的设计平台。

### 从 PC、FPGA 到芯片级的运行平台

自 LabVIEW 诞生以来，NI 的工程师们就有一个梦想，希望能够将 LabVIEW 的代码“编译直接下载到目标硬件”中，但在当时他们并不清楚如何将其变为现实。

2005 年推出的 LabVIEW 8 中为分布在不同计算目标上的各种应用程序的开发与发布提供了有力的支持。这种“分布式智能”的架构使相同的 LabVIEW 代码可以下载到不同的硬件平台中运行，而这正是实现上述梦想的一个基础条件。

如今的 LabVIEW 已经可以通过不同的模块将代码下载到从 PC、FPGA 到芯片级的硬件平台中，这个特性使工程师们在产品设计、原型到发布三个过程中都能利用相同的代码，减少了代码移植所带来的风险和问题。

其中，LabVIEW 与 FPGA 的完美结合是最为靓丽的一抹。FPGA 作为一种主流的技术，它通常需要使用 VHDL 这样的硬件语言来开发，这种语言需要很长的学习时间，以及深厚的硬件技术背景，“进入门槛”相对较高。而 LabVIEW 本身并行化的编程方式允许工程师们能以直观的方式来实现 FPGA 的逻辑功能（图 3），因此无需 VHDL 就可以让更多的工程师都能得益于 FPGA 技术。

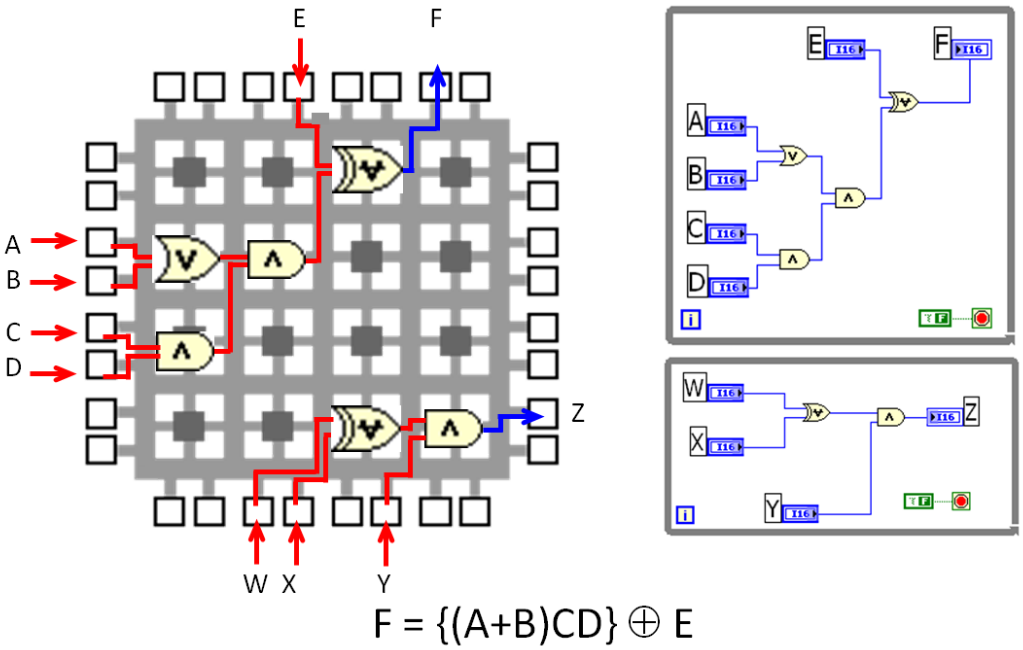


图 3 LabVIEW 简化 FPGA 的开发过程

除 FPGA 之外，最新版的 LabVIEW 还能够将代码运行于各种其它嵌入式平台中，包括工业触摸屏、ADI Blackfin 微处理器、DSP 芯片以及基于 ARM 的微控制器等等。因此，可以预见的是，将 LabVIEW 代码“下载到任何目标硬件”的时代已不再遥远。


图 4 强大的 LabVIEW 图形化设计平台

最后，当我们从开发平台的角度（图 4）再来审视一下 LabVIEW 上述这三个重要特性时，我们会发现，详全的附加工具包让工程师们能够在具备特定应用领域的专业知识和计算功能的同时，也能享受到图形化编程所带来的各种便利；而灵活多样的计算模型则使 LabVIEW 超越其它编程语言、成为系统级设计平台的一个必要条件；最后，通过 LabVIEW 各种模块来实现不同硬件的代码部署，工程师们可以在产品设计、原型到生产这些不同的阶段都能使用统一的开发平台，保证了代码的长期投资。

## LabVIEW 助力工程创新

如今的 LabVIEW 在测试、测量与自动化领域已经处于“无处不在”的领先地位，帮助工程师和科学家们高效地完成各自的应用，实现工程创新：

美国伊利诺伊大学（University of Illinois）的工程系学生们使用 LabVIEW 设计了世界上第一台可以用人的思维去控制的轮椅（图 5）。借助 LabVIEW 强大的信号合成、频谱分析以及数字信号处理功能，他们高效地开发了复杂的算法将神经信号翻译成控制命令，创造性地为残疾人提供了福音。


图 5 使用 LabVIEW 设计世界第一台用思维控制的轮椅

世界上最大的粒子物理实验室 CERN，是一个致力于研究物质的基本构成及物质间的相互作用的研究组织。选用 LabVIEW 以及 FPGA 模块，CERN 成功开发了基于 FPGA 的运行控制系统，实现了在世界上最强大的粒子加速度器–大型强子对撞机（LHC）上进行实时测量与控制大量组件位置，从粒子束核心中吸收粒子能量，同时也确保了可靠性和精确性。利用 LabVIEW FPGA，使得他们可以快速地整合所需的功能，省却了不必要的成本和学习周期，降低了系统对人力资源的需求。

与此同时，在消费电子（iphone 的产线测试）、汽车电子（BMW 氢能 7 系硬件在环测试）、航空航天（波音飞机噪声定位测试）、石油化工（Nexans 公司石油勘察系统）、绿色工程（加州大学对哥斯达黎加进行雨林环境监测）等等领域，LabVIEW 也都扮演着举足轻重的角色，甚至于 2008 年北京奥运会的“鸟巢”与“水立方”体育场健康监测系统都是用 LabVIEW 来开发完成的。

我们欣喜地看到，LabVIEW 正与工程师们一起，合力创造一个更好的工程世界！

## 展望

那么，未来的 LabVIEW 将如何发展?

对于这个问题，让我们先从多核处理器技术谈起。众所周知，由于芯片能耗与热效应的限制，芯片制造商已经开始转向全新的芯片结构，那就是多核技术。以往，当电脑升级到一个更快速的 CPU 后，也就意味着每一条独立指令的运行速度都会加快。而如今使用多核之后，如果要想继续提高性能，开发者就需要开发一个并行程序来取代顺序程序。然而这对于许多习惯于开发单线程应用的开发者来说是一个极大的挑战，他们需要专门的语义创建和管理线程，并且在线程安全方式下进行数据的传送。

相比之下，LabVIEW 由于其本身就是一种并行的编程结构，因此非常适合于创建并行的多线程应用；而在 LabVIEW 5 时就开始支持多线程，编译器可以自动识别线程并创建线程到不同的任务和循环上，再由操作系统分配到不同的核上运行（图 6）。

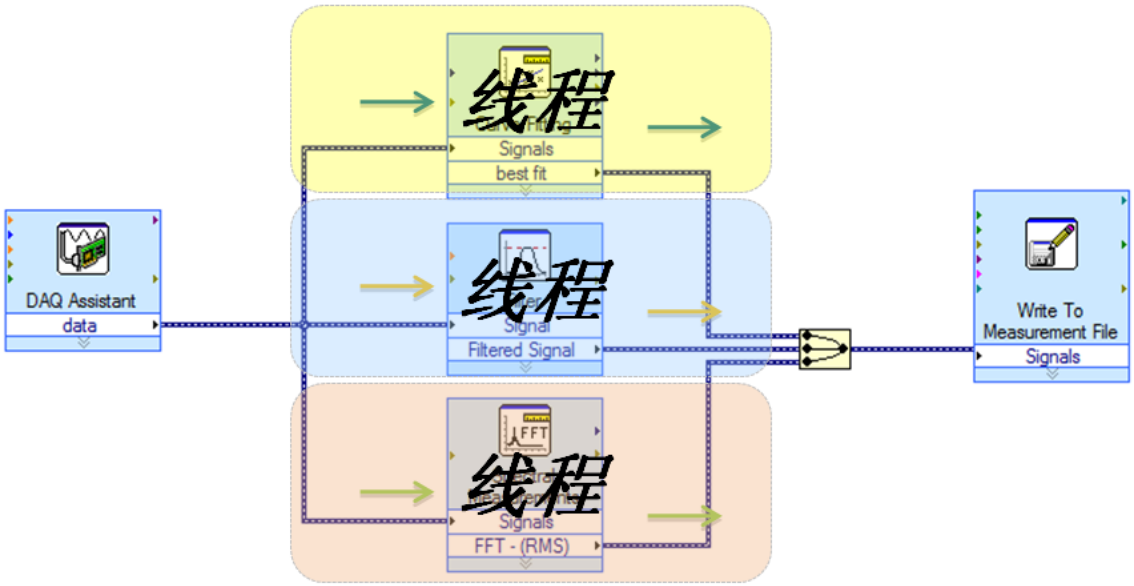


图 6 LabVIEW 本身就是自动多线程的编程语言

为了更好地与多核技术完美结合，LabVIEW 8.5 还针对性地提供了更多的特性：例如工程师可以根据自己需求手动设置线程运行在特定的核上，将时间确定性要求苛刻的采集与控制任务放在单独的核上运行，而将对确定性要求不高的界面响应、数据录入等任务放在另外一个核上运行（图 7）。

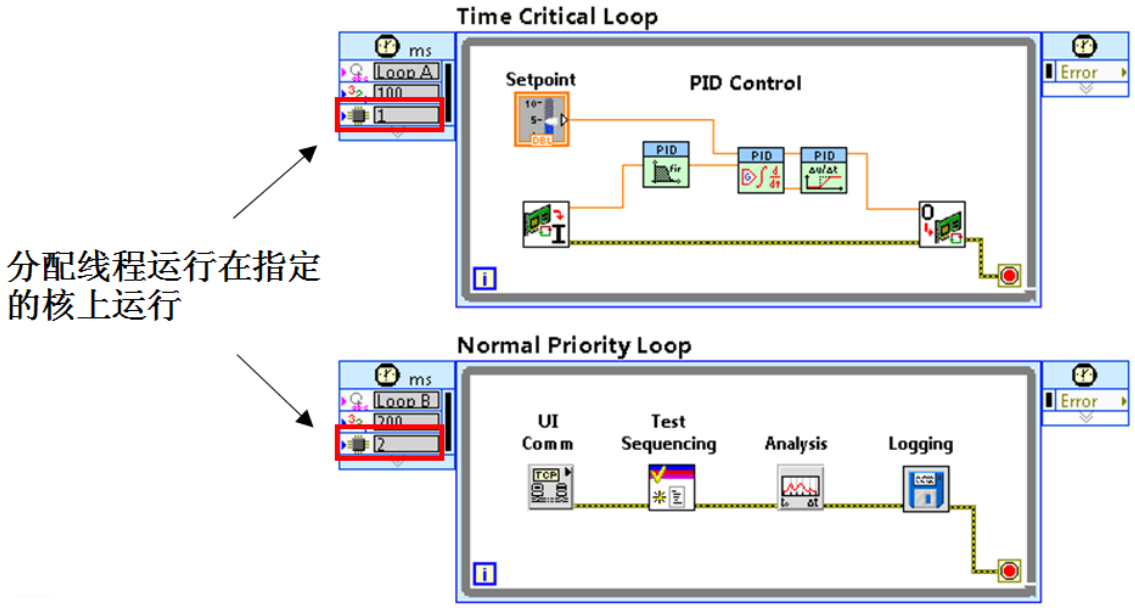


图 7 LabVIEW 允许用户手动分配线程在指定的核上运行

纵观 LabVIEW 的前世今生，我们可以很明显地感受到，新的技术的不断吸收和融合，是推动 LabVIEW 飞跃发展的源动力，多核处理器技术就是其中最为典型的一例。而这些主流的商业可用技术将向更多新的应用领域敞开大门，从而开启一个又一个新的纪元。

那么，当我们在真切地感受到了如今 LabVIEW 强大的功能和广泛的应用领域的同时，对于它的未来，您是否也和我一样，留有着更多的兴奋与期待呢？！