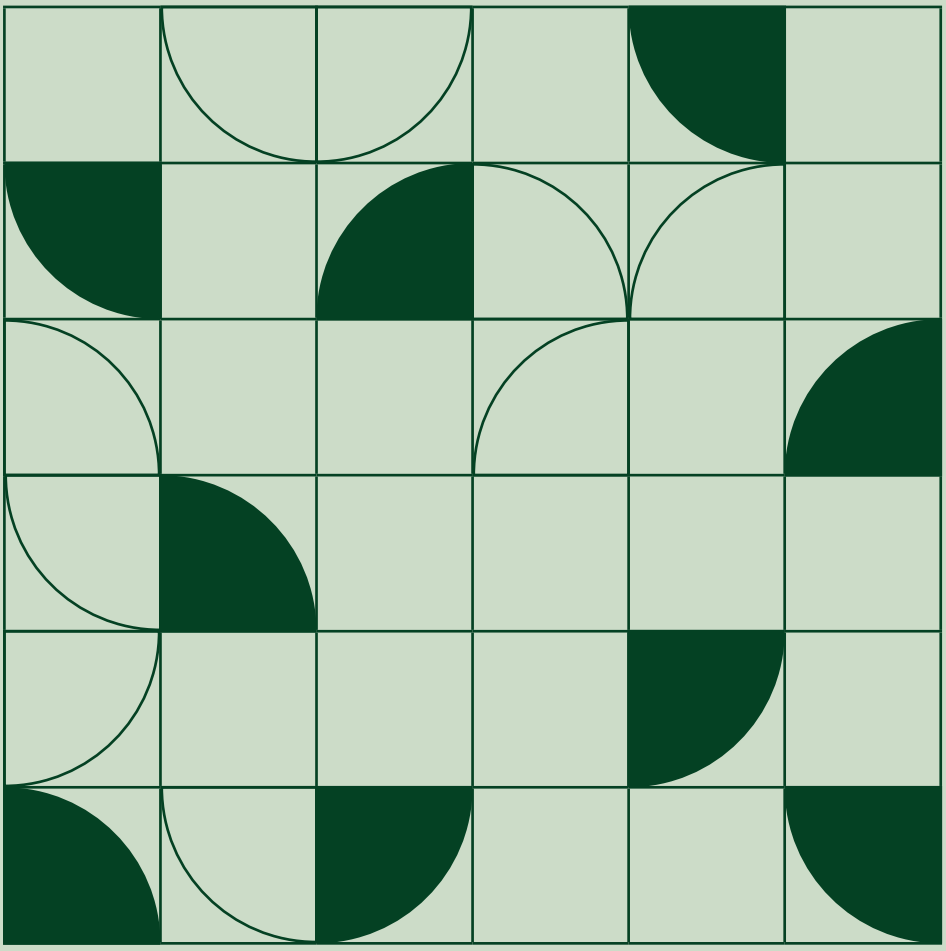
自动化测试系统总体拥有成本的建模



简介

大多数企业并未将生产测试视为重中之重，但有必要进行生产测试以避免代表公司品牌形象的产品在客户购买后出现重大质量问题。但是，生产测试成本可能十分高昂，有时会难以衡量，尤其是在没有简单的方法可以量化高质量产品或缩短产品上市时间对业务的积极影响时。但生产测试是“无可避免的麻烦事”这种观点并未束缚行业领先的企业，因为他们会力图了解开发、部署和维护测试系统的总成本，从而摆脱这种观点。实际的自动化测试成本远不只是测试系统的资本成本和操作员的人工成本。

通过本指南，您可以了解有助于评估测试机构测试能力的工具和信息、提出可显著节约成本的方案以及帮助公司做出更明智的投资决策来逐年提高盈利能力。

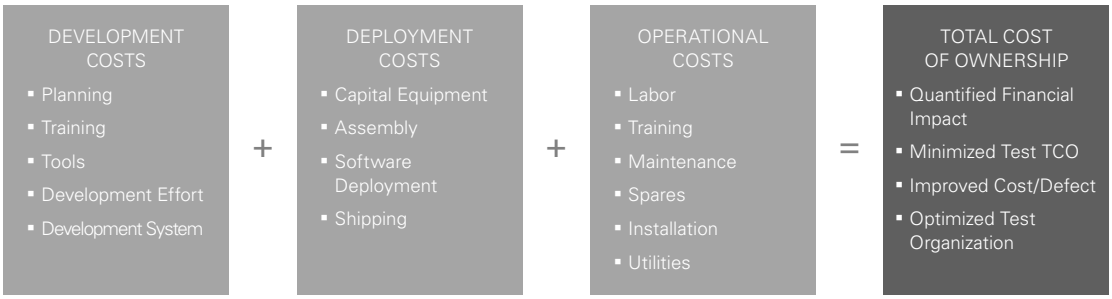


图 1 正确进行总体拥有成本建模可反映出特定测试资产的整个使用周期成本，并为评估未来战略投资提供财务框架。

开发成本

对于大多数应用，相较于部署以及操作和维护方面的成本，有关构建定制自动化测试系统的开发成本是最低的。这通常因为在开发过程中所构建的系统仅是用作性能基准测试和测试范围评估的概念性验证。但是，开发测试系统的总成本可能会因最终目标不同而有很大差异。公司在创建新产品时，通常会使用不同的架构和仪器来开发和比较多个测试系统，以确定最佳方法。

研发(工程)团队负责产品设计和构建大部分开发系统，因此成本将计入该团队的预算或成本中心。相对更成熟的测试部门会与其研发团队共同参与产品设计，此过程通常称为测试设计或DFT，他们同时也致力于开发测试系统。这是一种最佳做法，但并非适用于每个测试机构。

对于为测试单个设备或组件的功能而构建的测试系统，有关需求收集、仪器选择、连接件和软件开发的工作量相对有限。但若测试部门正在设计一个多功能的标准化测试系统，用于验证多个设备或组件的功能，开发成本可能有所增加。在这种情况下，必须花费更多时间来确定系统必须具备的全部功能排列、待测设备(DUT)连接件必须足够灵活、软件必须可以进行扩展，以便在新设备添加至产品组合时能轻松进行更改。

其他如编写硬件、测量抽象层或大规模互连系统均需要投入更多的前期开发成本，但是在应对快速的技术变化、面对长使用周期系统的仪器报废(EOL)问题时，测试机构会获得相应的投资回报。

开发自动化测试系统的相关主要成本：

- **规划工作**—正确识别测试系统的所有可行方案所需的时间和成本。包括浏览供应商网站、参与产品演示会、评估、贸易展会和论坛所花费的时间。
- **开发人员培训**—包括学习新的软件开发工具(集成开发环境[IDE]或测试执行程序)和硬件平台(例如，带有SCSI或PXI的机架和堆栈)所需的时间和培训课程费用。
- **开发工具**—购买测试软件(IDE或测试执行程序)开发许可证的相关成本。
- **开发工作**—开发概念性验证测试系统软硬件所需的时间。
- **开发系统**—购买初始概念性验证或演示测试系统的相关资本成本，用于对当前或其他新系统进行基准测试。

部署成本

当产品投入生产时，必须扩大概念性验证或演示测试系统以满足产品的批量生产需求。测试系统的吞吐量(单位时间内测试的单元数)直接影响所需的系统数量，而产品管理和销售渠道决定预测数量。除了测试功能的覆盖范围，在开发阶段中，应考虑的首选因素是所需的测试系统数量，这直接影响总部署成本。

而增加测试系统部署成本的另一因素是运输。规模较小的公司认为这一因素并不具有挑战性，因为其制造测试部门和研发部门可在同一地点办公，或两者的办公地点至少在地理位置上非常接近。然而，如果规模较小的公司缺乏制造和测试其设备或组件的能力或专业知识，也会选择外包产品制造和测试工作。但大公司的制造测试部门和研发部门可能设立在同一国家的不同地区，甚至设立在完全不同的国家/地区。这将显著增加部署成本，特别是在制造测试系统尺寸较大和/或较为沉重的情况下。较慢的货运方式的确可降低部署成本，但仅限于不考虑时间因素的情况下。最佳做法是在开发阶段就将测试系统的物理尺寸和重量加以考虑，尤其是在两种方案之间进行比较时，这有助于显著减少下游成本。

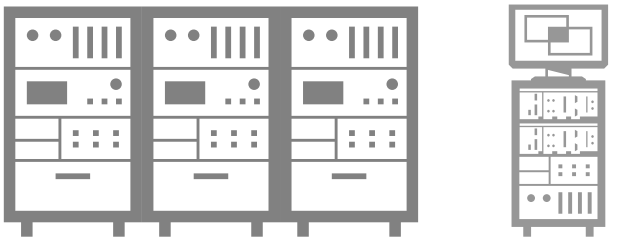


图2 在性能相似的两个测试系统之间选择时，请选择尺寸更小、更轻便的测试系统以降低部署成本。

部署自动化测试系统的相关主要成本：

- **资本设备**—所需测试系统的数量(取决于产品需求和测试系统吞吐量)，直接影响设备成本。
- **系统组装**—将单个组件组装到测试系统中所需的时间，组装过程包括构建19英寸或21英寸仪器机架或其他机械外壳、安装所有测试仪器、连接线缆和接线，安装开关和大规模互连以及连接件。
- **软件部署**—编译或构建软件组件集合，并将这些组件从开发计算机导入到目标机器上执行的过程中所涉及的成本。
- **运输和物流**—测试系统的尺寸和重量以及生产或制造设施所需的测试系统数量直接影响该成本。运输距离和接收货物所需的时间窗口也会影响此成本。系统的耐用程度决定是否需要使用特殊包装。

操作和维护成本

最后与测试系统相关的成本是操作和维护成本，这一类型的成本经常为人们所忽视或低估。这些成本通常并不属于负责产品或设备初始设计的研发团队，而几乎总是归结至制造或生产团队；这种成本中心的分离使得跨部门协作成为常见的难点。在公司选择外包其产品制造和测试工作的情况下，合同制造商承担这些单项成本并与公司协商这些服务的固定费率或小时费率。

操作和维护自动化测试系统的相关成本：

- **每小时操作**—测试系统操作员和支持技术人员的人工成本，用于确保系统在制造过程中正常运行。测试系统的数量和操作系统所需的技能水平直接影响该成本。
- **操作员培训**—每名操作员学习测试系统使用方法所需的时间。无论培训采用何种形式(操作手册、在线或面授课程)，该成本通常仅限于每名操作员必须参加培训的时间。对于拥有多种测试系统的公司，必须在两种模型(每名操作员都可以操作每个测试系统和每个操作员专门操作单个测试系统)之间决定人员配置策略。
- **维护**—保持测试系统和仪器正常工作的相关成本。该成本通常包括每年校准设备的成本，以及故障时更换仪器所需的预测成本。系统维护的难易程度也会影响该成本。
- **备用库存**—在意外停机(如仪器故障)或计划停机(如校准)时运行备用仪器所需的成本。对于拥有多个独特测试系统的公司，需要为每个测试系统准备备用仪器；由于产品结构多样，公司还需要为其测试系统提供更多的备用仪器和零件，以确保较长的正常运行时间。
- **安装**—消耗大量电力或产生大量热量的测试系统需要安装特殊的大功率电气设备或冷却塔以确保性能正常。
- **公共服务**—为测试系统供电、冷却和提供场地(占地面积)的相关成本。制造车间每平方英尺的价格和电价可能因地理位置的不同而有很大差异。

财务分析方法

由于开发和部署成本按多年摊销，而操作和维护成本属于还未发生的费用，因此必须使用财务模型来确定测试系统的总体拥有成本。对于传统投资情况，项目将产生收入和利润。然而这种情况下，不会产生收入或利润，而是要比较一个测试系统相对另一个所节省的成本。例如投资高效照明或房屋保温项目，投资人需要在前期投入资金，但从长远来看，项目将通过减少基础设施费用来而产生收益。

- **投资回收期(PP)**—这是实现项目资金回笼所需的时间。这种计算方法包括两部分。首先，必须明确开发和部署新测试系统与部署更多旧系统之间的差额以确定前期成本。因为旧系统已开发完成，所以不会产生相应成本。其次，将这个差额除以每年根据新系统效率(吞吐量)所节省的操作成本。

$$\text{投资回收期(PP)[年]} = \frac{\text{前期成本[\$]}}{\text{每年节省[\$/年]}}$$

- **投资回报率(ROI)**—代表项目使用周期内赚取的资金与投资资金的比率，以百分比表示。由于需要分别计算出新旧系统的预计总体拥有成本，然后明确两者差额，因此计算较为复杂。随后将此差额除以更具成本效益选项的总成本，并从所得商中减去1 (100%)以得到百分比形式的结果。

$$\text{投资回报率(ROI)[\%]} = \frac{\text{净节省总额[\$]}}{\text{总成本[\$]}} - 1$$

- **其他模型**—要确定项目或金融投资的可行性，可使用许多其他财务模型，如内涵报酬率(IRR)、净现值(NPV)和修正后内涵报酬率(MIRR)等。但在比较两种方案时，使用这些模型的高级建模往往会失效，因此您可以将分析仅简化为PP和ROI。

实际示例

以下实际示例可演示如何使用总体拥有成本的财务分析，来做出有关购置新测试系统架构的明智决策，而非遵循旧方法。

概述

市值2亿美元的B公司是一家基于IP的卫星通信系统制造商。该公司当前的生产测试系统采用传统的机架堆叠式台式仪器。B公司开发这些测试系统并将其部署给合同制造商，后者以每小时30美元的固定费率向他们收取产品测试费用。

当前测试系统的最显著特征如下:

- 功能齐全且测试范围全面
- 资本成本适中
- 测试机构接受过操作方法的全面培训
- 吞吐量非最佳

由于B公司最近投资了更大的销售渠道，其雷达产品由此进入新市场，因此生产能力必须从每年10000台增加至25000台。

他们的工程团队与NI合作，指定了一个基于PXI的新测试系统，该系统可将每个待测设备(DUT)的测试速度提升3倍。但新的解决方案需要在前期投入开发和部署成本，因此在做出决策之前，相对于基于先前的架构购买其他测试系统，必须对迁移产生的业务影响进行建模。

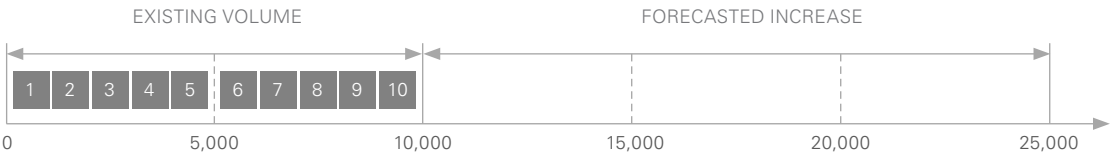


图3 每年产品需求增长15000。

开发和部署成本

评估过程中最常见的假设是，在测试机构已经过充分的培训并不会产生开发成本的前提下，基于现有架构购买其他测试系统更经济合算。因为系统已构建完毕，只需复制即可。然而，新系统在开发期间需要考虑规划、设计、培训和其他非重复性工程(NRE)成本。

然而，新系统的吞吐量优势不容忽视；因为吞吐量直接决定了为达到预测的数量增长，而必须购买其他或新测试系统的数量。在这种情况下，如增加现有测试系统的数量需要15个其他系统，而仅需购买基于PXI的新系统即可满足吞吐量需求。

现有的机架堆叠式系统	
NRE资金投入：	N/A
NRE开发时间：	N/A
资本支出：	每个系统100000美元
#现有的测试系统：	10
测试时间：	每台设备40分钟
总量/吞吐量：	每年1000台设备

基于PXI的新系统	
NRE资金投入：	90000美元
NRE开发时间：	150000美元
资本支出：	每个系统120000美元
#现有的测试系统：	N/A
测试时间：	每台设备13分钟
总量/吞吐量：	每年3000台设备

其他财务变量	
摊销时间表：	5年
替换现有系统：	不，继续运行
每小时操作成本：	30美元(合同制造商)
所需吞吐量：	每年25000台

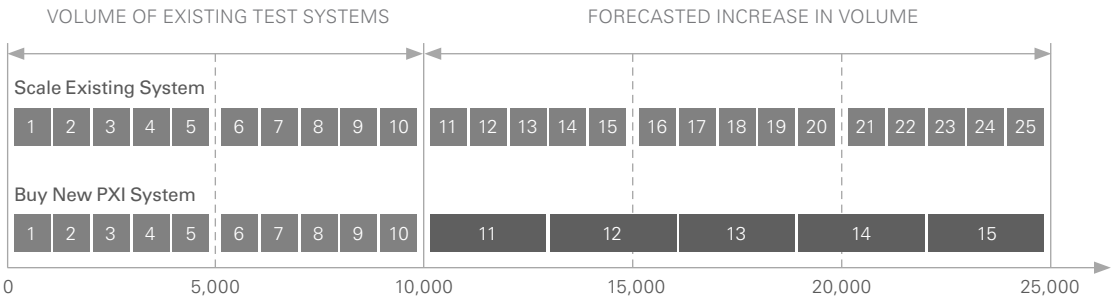


图 4 新的PXI测试系统将吞吐量提高了3倍，大大减少了满足额外产品需求所需的系统数量。

确定每种方法所需的测试系统数量后，可以比较开发和部署的相关总成本，并直接了解吞吐量、资本支出和NRE的影响。

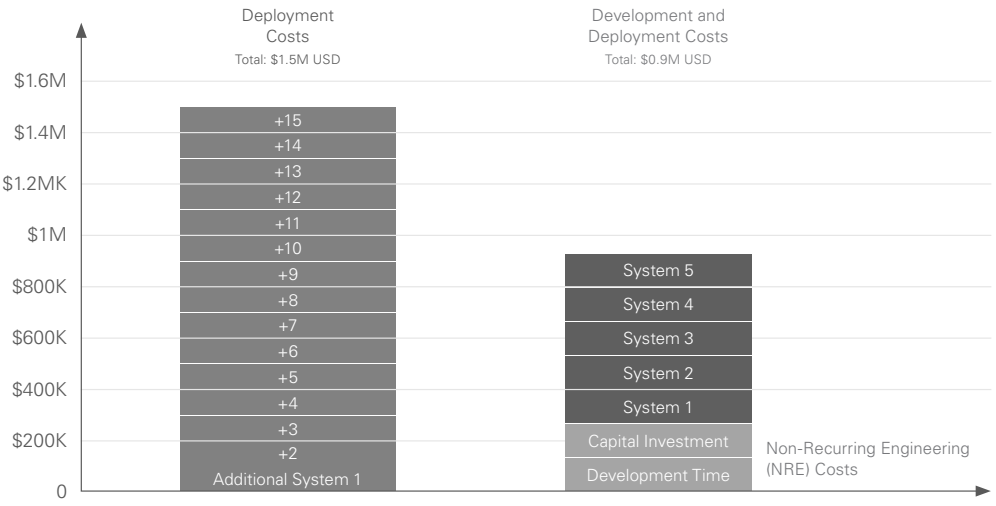


图5 尽管基于PXI的新测试系统会产生NRE开发成本，但新系统的开发和部署总成本会减少600000美元。

对于此给定的示例，在比较开发和部署成本时，购买新解决方案比扩展现有的测试系统更具成本效益。扩展现有系统而产生过高成本的最主要原因是系统的吞吐量降低。为实现所需吞吐量，测试系统的数量需要增加3倍，那么吞吐量本身就增加了部署成本。

但如果变量变化，会发生什么呢？对不同的假设分析场景建模，以确保即使在最不利的情况下也能实现盈利。

待建模的假设场景：

- 如果新系统的开发时间是原来的两倍，因此成本也会是原来的两倍吗？
- 如果资本支出因货币通胀而增加10%，会怎么样？
- 如果吞吐量仅提高了1.5倍而不是3倍，会怎么样？
- 如果销售量从25000台改为仅20000台，会怎么样？
- 如果可增加的占地面积有限，会怎么样？
- 如果测试设施必须安装额外的电源或冷却装置，会怎么样？
- 如果以前的仪器现在停产了，会怎么样？

操作和维护成本

在开发和部署了所需数量的测试系统后，您还必须在整个项目或产品使用周期内对其进行操作和维护。操作和维护测试系统的相关成本通常归属于企业的制造部门，而测试系统的开发和部署成本则由研发(工程)部门承担。如果缺少领导层指导，工程团队很可能默认仅在开发和部署方面进行成本优化，而不考虑操作和维护成本的影响。

在以上仅考虑开发和部署成本的示例中，相对于额外购买基于先前架构的测试系统，购置新测试系统更经济实惠。现在对两种方案在项目前5年间的操作和维护成本进行分析，以了解其对测试总成本的影响。

在这种情况下，B公司已外包其产品制造和测试工作。合同制造商向B公司收取每小时30美元的测试系统操作费用。

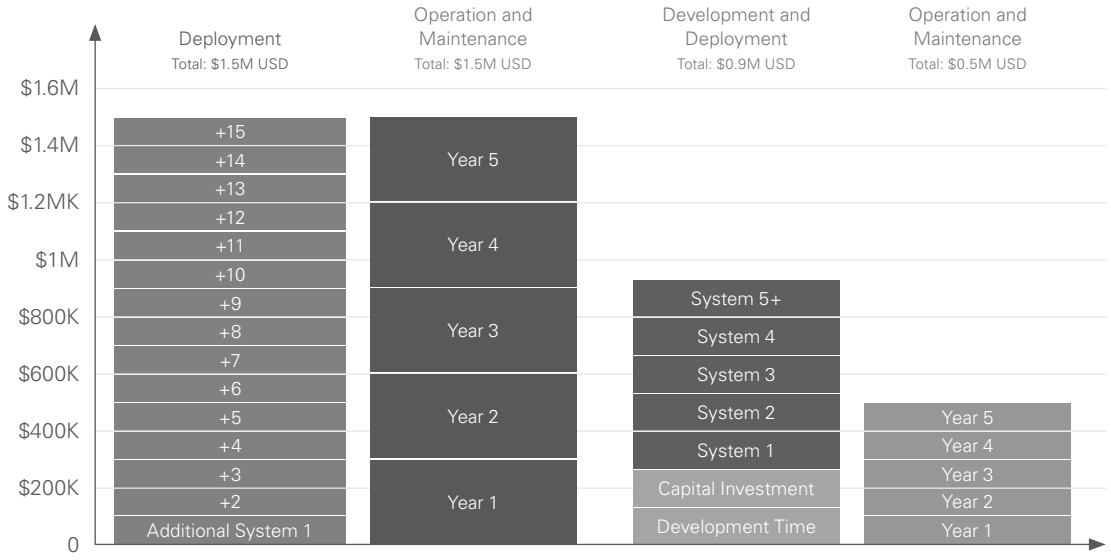


图 6 除了开发和部署成本明显减少外，基于PXI测试系统的操作和维护成本相较以前系统的相应成本也大大降低。

总体拥有成本

尽管在这种情况下，PXI测试系统是最佳选择，但确定总体拥有成本以有效地对新系统的财务效益建模仍然非常重要。此项为期5年的分析提供了对PP、ROI、总节省额以及每个部件测试成本降低幅度等变量的有用信息。此分析中的开发和部署成本在5年内平均(固定)摊销。

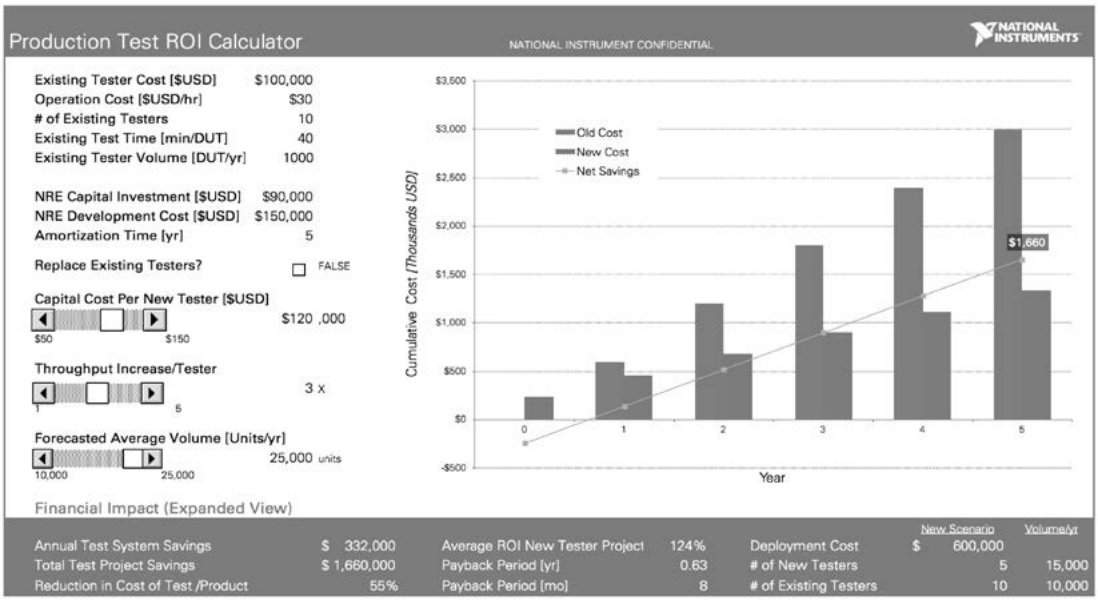


图 7 与扩展现有解决方案相比，新测试系统在5年内共节省了166万美元，投资回收期为11个月。

示例总结

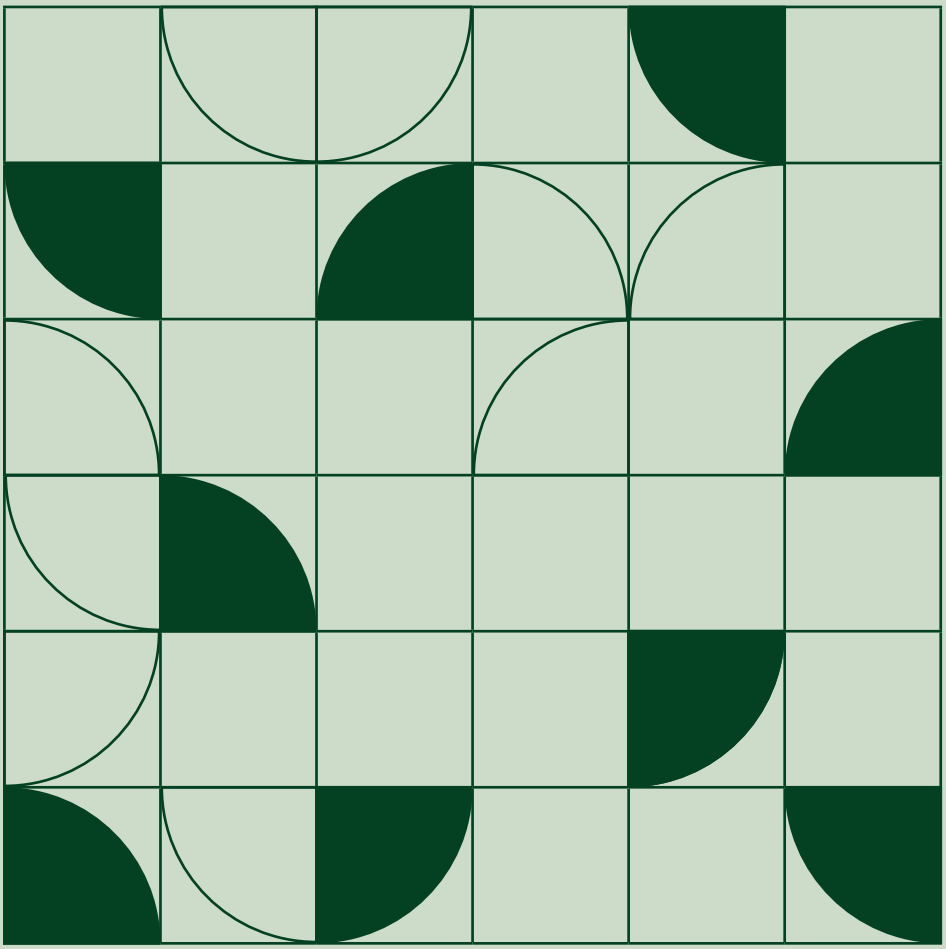
在两种解决方案选项之间抉择时，需要考虑许多因素。常见的假设认为扩展旧解决方案更容易、成本更低，但进一步的分析表明，投资于更新、更高性能的系统是更高明的财务决策。PXI系统的财务优势在于吞吐量提高了3倍，这使B公司仅购买1/3的测试系统就能完成相同的任务，从而节省了资本投资。在5年间，这也大大降低了B公司向合同制造商支付的操作和维护成本，从而实现了项目11个月的投资回收期和124%的投资回报率。

结语

随着设备复杂性和上市时间压力的持续飙升，自动化测试系统的总体拥有成本将继续在公司的盈利能力中占据重要地位。为实现盈利，您在考虑制定采购决策时不应局限于测试系统的初始资本成本，而是确保所有相关成本因素都已纳入考量。本指南侧重于自动化生产测试，但您可以此类推，将相同的概念应用于产品从初始概念到最终用户的其他阶段(包括研发、特性分析、查证和验证)。

作为PXI平台、LabVIEW图形系统设计软件和TestStand测试管理软件的开发者，以及PXI系统联盟的创始成员，NI拥有40余年的丰富经验，能够帮助半导体生产、航空航天和国防等行业中的企业开发自动化测试系统。我们在全球50多个国家/地区的现场工程师团队致力于为任何规模的公司提供帮助，在确保最高产品质量的同时降低测试成本。有关后续流程，请联系您当地的NI销售代表。

仪器选择



引言

工程师们普遍认同：工欲善其事，必先利其器。如果使用的工具不当，可能会浪费时间并影响质量，而运用适当的工具则可在短时间内获得预期的成效。

在构建自动化测试系统时，使用的主要工具是测量仪器。这些仪器包括数字万用表(DMM)、示波器和波形发生器等大家耳熟能详的仪器，以及各种新的和不断变化的产品类别，如矢量信号收发仪和多功能一体式示波器。

为了选择合适的仪器，技能熟练的测试工程师必须对以下方面有着深入的了解：

- 待测设备(DUT)的技术测量要求
- 会影响应用的关键仪器规格
- 市面上的各种仪器类型，以及功能、尺寸、价格等方面的权衡
- 特定仪器类别中产品型号之间的细微差异

为工作选择合适的工具知易行难，尤其是需要了解和权衡许多因素时更是如此。本指南介绍了市面上仪器的主要类别，以及常见的选择标准，可帮助您缩小选择范围，从而选出最适合应用的仪器。

模拟和射频仪器

模拟和射频测试仪器的范围非常广泛，分为数百个产品类别，共计数千种产品型号。而且，这些仪器也受相关物理定律的制约，即体现在放大器技术和制造仪器所需的模数转换器(ADC)上的噪声与带宽的基本原理。由于存在这些基本的物理限制，工程师经常需要在测量精确度和数据采集速度之间进行取舍。下图显示的是随着传统和模块化仪器的技术进步，速度与分辨率之间的关系如何随着时间的推移而变化。

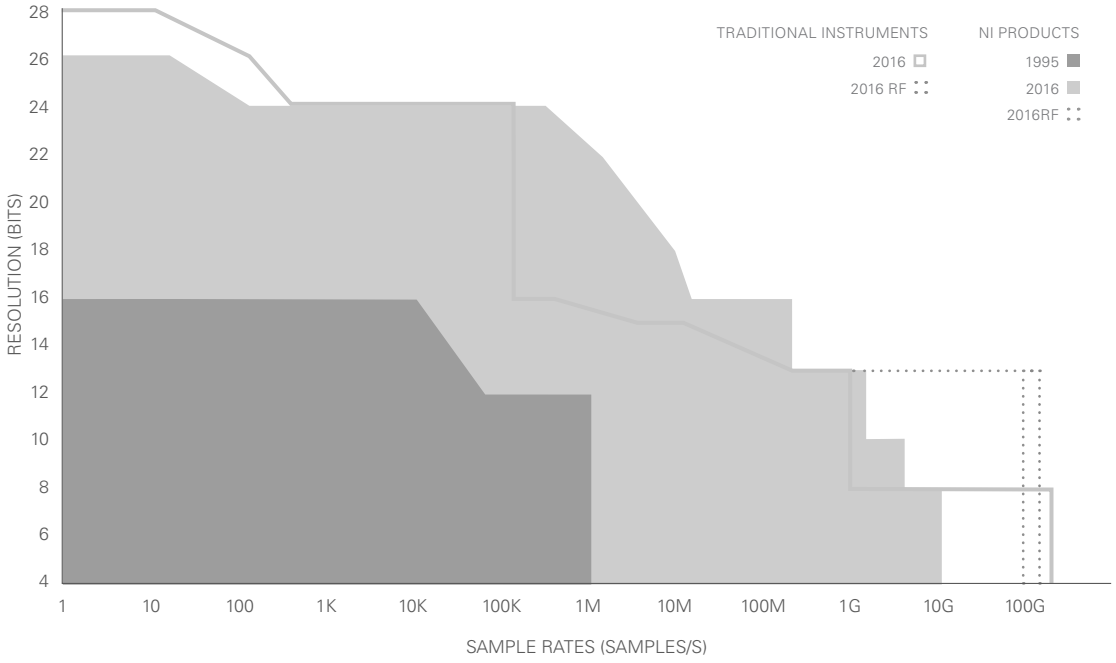


图1 仪器的分辨率与采样率

模拟和射频仪器类别

图1中的曲线代表了不同仪器类别的示例。DMM位于图的左上角，能以低速提供高精度，示波器位于图的右下角，能以较低的分辨率进行高频采集，而数据采集(DAQ)产品位于图的左下角，具有更高的通道密度和更低的成本。

如果要确定从哪个类别的仪器开始研究，首先要考虑与测量任务相关的几个关键问题：

- 信号的方向是什么？(输入、输出或两者皆有)
- 信号的频率是多少？(DC、千赫、兆赫或千兆赫)

在确定关于方向和速度的这两个关键问题的答案后，便可参考表1来确定从哪种类型的仪器开始着手。

模拟仪器类别

	DC和电源	低速模拟	高速模拟	RF和无线
输入，测量	数字万用表	模拟输入，数据采集（DAQ）	示波器、频率计数器	RF分析仪 功率计 （频谱分析仪、矢量信号分析仪）
输出，生成	可编程电源	模拟输出	函数/任意波形发生器 （FGEN、AWG）	RF信号发生器 （矢量信号发生器、连续波源）
同一设备上的输入和输出	DC电源分析仪	多功能数据采集设备 （多功能DAQ）	多功能一体式示波器	矢量信号收发仪（VST）
同一引脚上的输入和输出	源测量单元（SMU）	LCR测试仪	阻抗分析仪	矢量网络分析仪（VNA）

这个图表虽然有用，但包含的仪器类型有限，特别是缺少垂直规格或特定用途的仪器。

该表并未提及但值得注意的一些领域包括：

- 专用直流仪表，如静电计、微欧姆计、纳伏表等
- 音频频带分析和生成(也称为动态信号分析仪)
- 专业模拟产品，包括脉冲发生器、脉冲源/接收器等

需考虑的关键规格

当为测量任务选定特定仪器类别之后，下一步是对该类别的产品进行比较和权衡，需考虑的要求包括：

- **信号范围、隔离和阻抗**—首先，确保仪器的输入信号范围足够大，能够捕获感兴趣的信号。此外，需考虑仪器的输入阻抗以及仪器与地面的隔离，前者会影响测量装置的负载和频率性能，后者会影响抗噪声性和安全性。
- **模拟带宽和采样率**—接下来，确保仪器的模拟带宽(以千赫、兆赫或千兆赫为单位)能够传递感兴趣的信号，并且ADC的采样率足够快，能够捕获感兴趣的信号(以每秒样本数表示，例如每秒千个样本、每秒百万个样本或每秒十亿个样本)。
- **测量分辨率和精度**：最后，评估仪器垂直规格中影响测量质量的多个参数，如ADC分辨率(模拟信号的数字量化，通常在8位至24位之间)、测量精度(最大测量误差随时间和温度的变化，一般以百万分之一或百万分率表示)和测量灵敏度(可检测的最小变化，通常以绝对单位表示，例如微伏)

对于在量程、精度和速度等功能维度表现出色的仪器，可能需要在价格、尺寸、功耗和通道密度方面进行取舍，而所有这些因素都会影响仪器的实用性。

图2显示了通用测量仪器的模拟输入路径简图，包含四级主要输入、每一级影响的仪器规格，以及每一级影响的典型DMM和典型示波器的仪器规格示例。

下方的简图概括了筛选仪器规格时的一些思路，这些规格通常使用各种仪器类别和仪器供应商的各种不同命名法来表示。这些级在影响关键规格时通常是相互依存的。例如，输入放大器还可以影响仪器的输入带宽和有效分辨率。类似地，仪器的输入阻抗可能对带宽具有显著影响。

模拟仪器的四级输入

	输入隔离和端接	输入耦合和滤波	输入放大器	模数转换器(ADC)
规定的产品规格	**隔离输入阻抗**	**AC/DC耦合 模拟带宽**	**最大电压范围 最低电压灵敏度**	**采样率 分辨率**
DMM示例:	隔离电压高达330 V Cat II，10 MΩ(可选择)	DC耦合 200 kHz带宽	高达300 V输入 低至10 nV灵敏度	10k Hz读取速率 6½位数(24位)分辨率
示例: 示波器:	接地参考50 Ω或1 MΩ(可选择)	DC或AC耦合(可选择) 350 MHz带宽	高达40 Vpp输入 低至1 mV灵敏度	高达5 GS/s采样率 8位分辨率

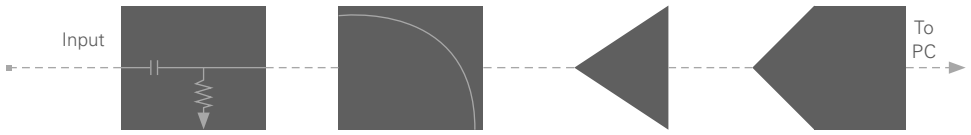


图2 模拟仪器的四级输入

模拟和射频仪器类别

在比较DUT的测量要求以及仪器测试DUT的能力时，请记住以下关键比率。

测试准确度比= 4:1

当测试组件(例如测量组件的参考电压)时，请确保测量设备的准确度远远高于待测组件的准确度。如果不满足该标准，则测量误差可能是由DUT和测试设备两方面引起的，因此便无法找到确切的误差源。出于这一原因，需要采用测试准确度比(TAR)这一概念来描述测量设备和待测组件的相对准确度。

可接受的TAR值为4及以上，具体取决于所执行的测试和所需的测试确定度。

$$TAR = \frac{\text{待测组件所需的准确度}}{\text{测量设备的准确度}}$$

带宽比= 5:1

上升时间和带宽直接相关，其中一个值已知的情况下，就可以计算出另一个值。上升时间定义了信号从满量程值的10%上升到90%所需的时间。

使用以下公式，可根据上升时间计算出信号的带宽：

$$\text{带宽} = \frac{0.35}{\text{上升时间}}$$

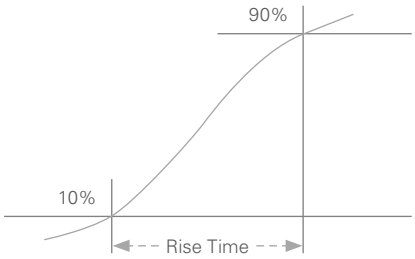


图3 模拟信号上升时间

理想情况下，所用数字化仪的带宽应该是信号带宽的三到五倍，如上式所计算。换句话说，数字化仪的上升时间应该是信号上升时间的1/5到1/3，才能以最小的误差采集信号。

您可以随时使用以下公式可进行逆推，确定信号的实际带宽：

$$T_m = \sqrt{T_5^2 + T_d^2}$$

T_m =测得的上升时间，

T_5 =实际信号的上升时间，

T_d =数字化仪的上升时间

时域采样率= 10:1

带宽描述了可以以最小衰减进行数字化的最高频率正弦波，而采样率仅仅表示数字化仪或示波器中的ADC提供时钟以对输入信号进行数字化的速率。采样率和带宽不直接相关；然而，可遵循以下通用原则，在这两个重要参数之间建立所需的关系：

数字化仪的实时采样率=

输入信号带宽的10倍

奈奎斯特定理指出，为了避免混叠现象，数字化仪的采样率需要至少是被测信号最高频率分量的两倍。然而，采样率只为最高频率分量的两倍并不足以精确地再现时域信号。为了准确地对输入信号进行数字化处理，数字化仪的实时采样率至少应为数字化仪带宽的3到4倍。具体原因请看下图，并思考您希望在示波器上显示哪种数字化信号。

虽然在两种情况下通过前端模拟电路的实际信号都是相同的，但是左图属于欠采样，会使数字化信号失真。相反，右图具有足够的采样点，可以精确地重建信号，从而实现更精确的测量。清晰地表示信号对于上升时间、过冲或其它脉冲测量的时域应用非常重要，因此采样率更高的数字化仪更适用于这些应用。

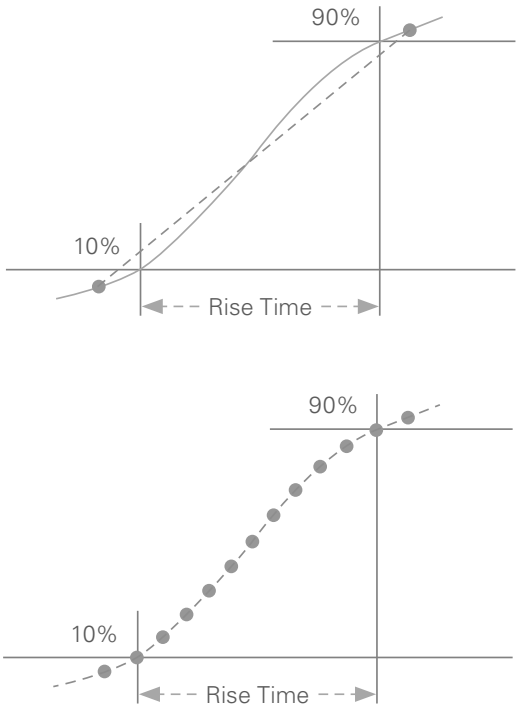


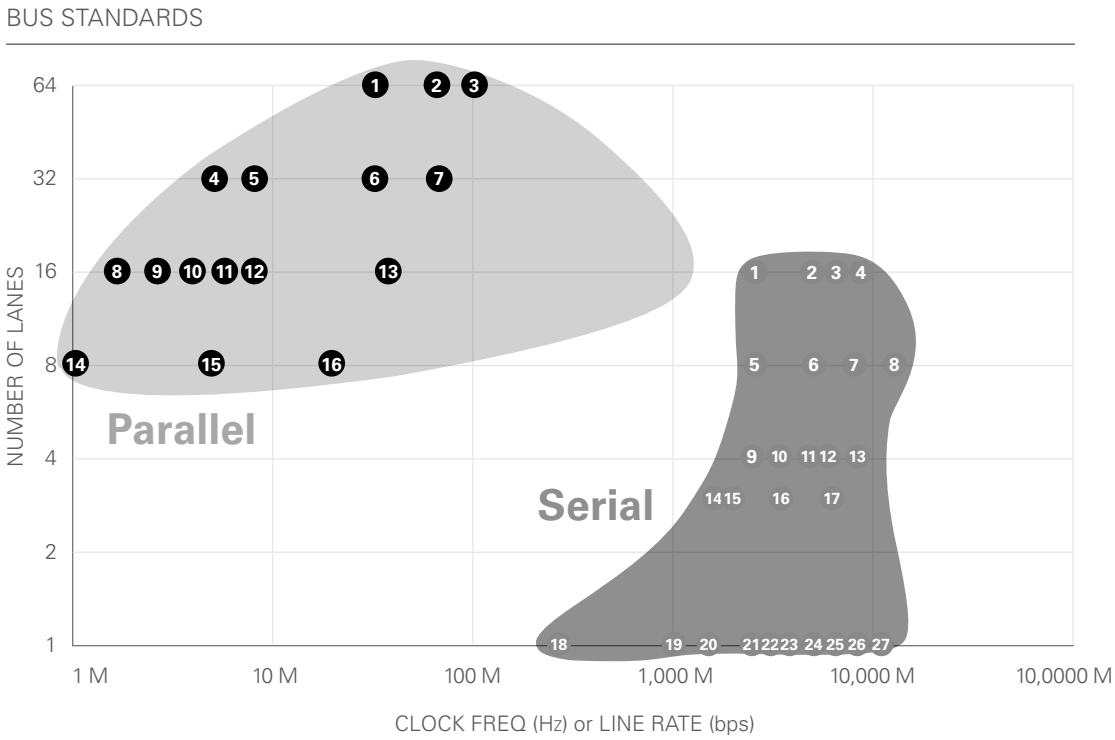
图4 右图显示的是采样率足够高的数字化仪，可精确地重建信号，从而实现更精确的测量。

数字仪器

在电子功能测试环境中，数字仪器用于与数字协议连接并测试协议的电气特性和通信链路特性。影响仪器是否适用于特定任务的一个最关键的因素是，仪器采用并行还是串行数字通信。

并行与串行标准比较

由于1 GHz至2 GHz附近的并行总线时钟频率存在物理限制，串行标准越来越受欢迎。这是因为由各个时钟和数据线引入的偏移会以更快的速率引起误码。高速串行总线会以单个差分信号形式发送同时包含数据和时钟信息的编码数据，从而避免了并行总线的速度限制。它会将数据进行串行化处理并以更快的速率发送，有助于减少集成电路(IC)的引脚数，从而减小尺寸。此外，串行通道可以以快得多的时钟速度运行，它们还可以实现比并行总线更高的数据吞吐量。



Parallel Bus

1. PCI 64-bit/33 MHz
2. PCI 64-bit/66 MHz
3. PCI 64-bit/100 MHz
4. Front Panel Data Port
5. EISA
6. PCI 32-bit/33 MHz
7. PCI 32-bit/66 MHz
8. IDE (ATA PIO 0)
9. ATA PIO 1
10. ATA PIO 2
11. ATA PIO 3
12. ATA PIO 3
    ISA 16-bit/8.33 MHz
13. Ultra-2 wide SCSI
14. RapidIO Gen1.1
15. GPIB
16. SCSI
    ISA 8-bit/4.77 MHz

Serial Bus

1. PCIe Gen1x16
2. PCIe Gen2x16
3. Serial RapidIO Gen2
4. PCIe Gen3x16
5. PCIe Gen1x8
6. PCIe Gen2x8
7. PCIe Gen3x8
8. JESD204B
9. PCIe Gen1x4
10. Serial RapidIO Gen1.3
11. PCIe Gen2x4
12. DisplayPort
13. PCIe Gen3x4
14. HDMI 1.0
    DVI
15. HDMI 1.3
16. HDMI 2.0
17. SD-SDI
18. Gigabit Ethernet
19. SATA 1.0
20. Serial FPDP
    PCIe Gen1x1
21. SATA 2.0
    3G-SDI
    JESD204A
    10 Gigabit Ethernet
22. PCIe Gen2x1
    USB 3.0
23. SATA 3.0
24. PCIe Gen3x1
25. USB 3.1

图5 该图显示了一些常见的总线标准及其通道数量与线路速率的关系。串行标准的线路速率远远高于并行标准，从而可实现更高吞吐量。

数字仪器类别

与模拟仪器一样，您可以通过回答几个关键问题来缩小数字仪器的选择范围：

- **需要完成什么任务？**(数字连接、定制数字连接或电气和定时测试)
- **链路的速度有多快？**(静态和kbps、Mbps或Gbps)

	静态，低速	同步与高速并行(100 MBPS范围)	高速串行(10 GBPS范围)
接口(标准)	低速标准接口卡(I2C，C) 同步协议接口 (ARINC 429、CAN、GPIB、I2C、SPI)		接口卡(10千兆比特以太网、光纤通道、PCI Express等)
接口(自定义)	数字I/O (GPIO)	数字波形发生器/分析仪，信号发生器	基于FPGA的高速串行接口 Aurora、Serial Rapid I/O、JESD204b
电气测试和定时测试(基本接口)	数字引脚电子器件， 单引脚参数测量单元(PPMU)		BERT，示波器

硬件定时与软件定时

数字通信方案主要通过两种方法实现：软件定时和硬件定时。软件定时应用的输入输出不需要使用任何类型的时钟。软件负责控制I/O，而编程语言则通过软件来控制定时。这种编程语言通常在操作系统上运行，可能需要数毫秒来执行软件调用。对于软件定时，可以使用操作系统定时器来确定定时操作的速率。通常，报警、电机和报警器的监测和控制等低速应用使用软件定时。

有两种类型的软件定时通信可供选择：确定性控制和非确定性控制。使用实时操作系统，可以实现高达1 µs的精确度；但是，实时操作系统不会提高通信速率，仅仅是增加确定性。非实时系统，如Microsoft Windows，是非确定性的。在这些系统中，软件命令在硬件中执行所花费的时间是不一致的，并且可能要花费数毫秒。计算机内存、处理器速度和在操作系统上运行的其他应用程序等因素都可能会影响执行所需的时间。

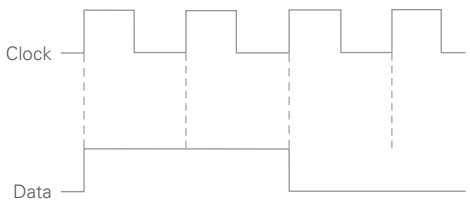


图6 通过硬件定时操作，您可以利用实时、确定的数字信号输出。

相比之下，硬件定时设备使用时钟的上升沿或下降沿确定性地生成或采集数据。您可以使用此类定时设备，以非常高的确定性在千兆位每秒的速率下采集或生成数字数据，并且可以在预确定的位置可靠地输出数据。

使用硬件定时的应用包括：

- 芯片测试
- 协议仿真和测试
- 数字视频和音频测试
- 数字电子测试

时钟速率

时钟速率是硬件定时数字应用的一个重要考量因素。如果设备所能达到的最大时钟速率不够块，将很难弥补。您可以使用NI高速数字I/O设备实现高达200 MHz的单端信号采样速率和高达200 MHz的差分信号，从而实现协议测试、数字音频和视频测试以及数字电子测试等。当设备可能无法满足串行数据流所需的时钟速率要求时，可以使用串行/解串器(SERDES)来采集更高频率的数字信号。但是，根据使用的SERDES类型，采用SERDES可能会减少可用线路的数量。

规格参数

除了认识从物理角度进行正确测量所需的模拟前端外，您还需要采用稳定、可重复、快速且能连接PC的仪器(这是工作的一部分)。

这有助于您根据场景/环境做出决定：

- **试验台和实验室**—准确性、可重复性、底层控制、易于安装且能够自动执行重复测试
- **制造车间**—速度、吞吐量、准确度、通过编程接口进行优化，以及调试

显然，就如何选择实验室与制造车间所需的仪器而言，有着相似之处和不同之处。人们通常会根据终端部署的一系列主要成功标准对仪器的规格参数进行评估。以下是可能适用于制造环境的一组典型的评估标准。

硬件部署检查清单

功能需求	测试工程记录
仪器，需要I/O?	
处理，需要计算?	
数据吞吐量，存储空间是多少?	
是否同步?	
未来有哪些要求?	
在未来几年需要部署的系统数量有多少?	
计划支持的年限有多长?	
复制的全球站点数量是多少?	
部署场景的环境稳定性如何?	
初始设置、配置和维修如何管理?	
是否支持机架式安装?	
尺寸、重量和功耗是多少?	
采用何种连接件和连接类型?	

选择总线类型

今天，USB、PCI Express和以太网/LAN作为仪器控制的有效通信选项而备受关注。有些测试和测量供应商和行业权威人士称，这些总线中的某一种，其本身就代表了所有仪器需求的解决方案。实际上，未来的测试和测量系统很可能仍然会同时采用多种总线技术，因为每种总线都有独特的优势。

带宽

在考虑其他总线的技术优点时，带宽和延迟是需要考虑的两个最重要的总线特性。带宽衡量的是通过总线发送数据的速率，通常以兆字节每秒为单位。高带宽总线在给定周期内传输的数据比低带宽总线更多。大多数用户都认识到了带宽的重要性，因为带宽会影响能否以采集和生成速率在总线与共享主机处理器之间传输数据，同时还决定着仪器所需的板载内存。带宽对于复杂波形生成和采集等应用以及RF和通信应用十分重要。高速数据传输对于虚拟和合成仪器架构尤其重要。虚拟或合成仪器的功能和特性由软件定义；在大多数情况下，这意味着数据必须移至主机PC进行处理和分析。图7显示了本指南中介绍的所有仪器总线的带宽(和延迟)。

延迟

延迟衡量的是通过总线传输数据产生的延迟。打个比方，如果将仪器总线比作高速公路，带宽则对应于车道数量和行驶速度，而延迟则对应于入口匝道和驶出匝道引入的延迟。低(越低越好)延迟总线在一端发送数据与另一端处理数据之间产生的延迟较短。延迟虽然比带宽更难以观测，但是会直接影响需要通过总线快速连续地发送间断性短命令的应用，例如DMM和开关之间的握手以及仪器配置等。

GPIB

IEEE 488总线(通常称为GPIB)是专为仪器控制应用而设计且经过验证的总线。作为一种强大可靠的通信总线，GPIB已经应用了30年，由于具有低延迟和可接受的带宽，至今仍然是仪器控制中最常应用的通信总线。GPIB目前在行业中的应用最为广泛，市场上有10,000多种仪器型号采用GPIB连接。

GPIB的最大带宽约为1.8 MB/s，非常适合用于独立仪器的通信和控制。最新的高速版本HS488将带宽提高至8 MB/s。数据传输基于消息实现，这些消息通常采用ASCII字符的形式。多台GPIB仪器可以通过电缆连接，总距离可长达20 m，并且带宽由总线上的所有仪器共享。虽然带宽相对较低，但GPIB延迟显著低于(优于)USB，尤其低于以太网。GPIB仪器在连接到系统后不会自动检测或自动配置；但是GPIB软件仍是最佳选择之一，而且坚固耐用的电缆和连接器也适用于恶劣的物理环境。无论是实现现有设备自动化的应用，还是需要高度专业化仪器的系统，GPIB都是它们的理想选择。

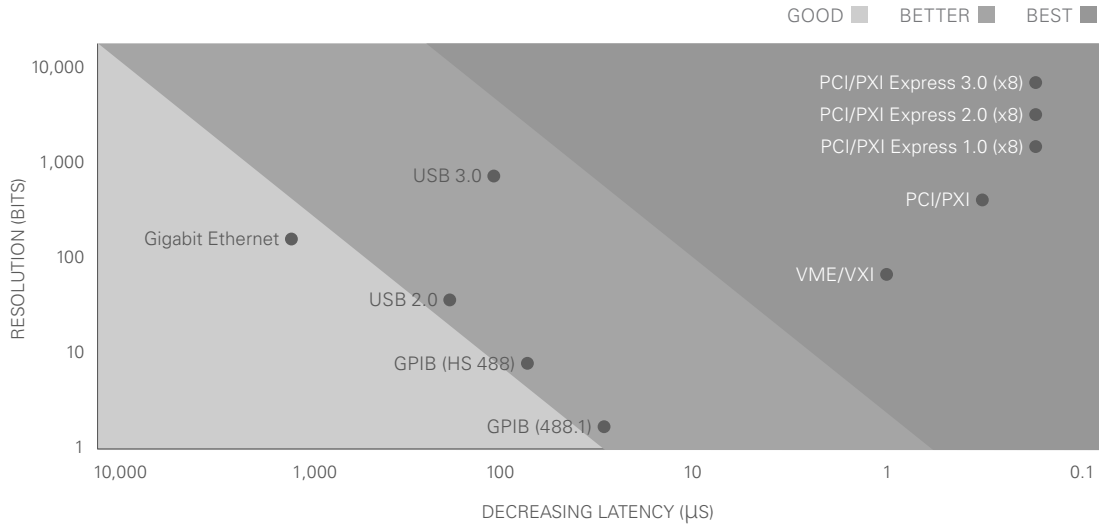


图 7 不同仪器总线的带宽和延迟关系图

USB

USB近年来已广泛用于连接计算机外设。这种趋势已经扩展到测试和测量领域，越来越多的仪器供应商在仪器中添加了USB设备控制器功能。虽然大多数笔记本电脑、台式机和服务器可能配有多个USB端口，但这些端口通常都连接到同一个主机控制器，因此USB带宽要由所有端口共享。

USB的延迟相对较短(介于延迟最长的以太网和延迟最短的PCI和PCI Express之间)，并且电缆长度最长为5 m。USB设备可自动检测，这意味着与LAN或GPIB等其他技术不同，当用户将USB设备连接至PC时会立即被PC识别和配置。在本文所介绍的所有总线中，USB连接器的可靠性和安全性最低。可能需要使用外部电缆扎带将其固定到位。

USB设备非常适合于需要采用便携式测量、笔记本电脑或台式数据记录以及车载数据采集的应用。USB普遍应用于PC并且具有即插即用的易用性，因此已经成为独立仪器普遍采用的通信总线。USB测试和测量类(USBTMC)产品规范可满足各种测试和测量设备的通信要求。

PCI

在本文介绍的所有仪器总线中，PCI和PCI Express的带宽和延迟性能最为出色。PCI带宽为132 MB/s，由总线上的所有设备共享。PCI延迟性能非常出色，以700 ns为基准，而以太网延迟为1 ms。PCI总线基于寄存器进行通信。与本文提到的其他总线不同，PCI不能连接外部仪器。它是一种用于PC插入卡和模块化仪器系统(如PXI)的内部PC总线，因此无法直接测量通信距离。尽管如此，当连接到PXI系统时，使用NI光纤MXI接口，PCI总线可以延长至200 m。因为PCI连接位于计算机内部，我们可以认为，连接器的可靠性受其所在PC的稳定性和耐用性限制。

PXI模块化仪器系统基于PCI信号构建，通过高性能背板连接器和多个螺栓端子增强了连接性，保持连接到位。在插入PCI或PXI模块后启动，Windows就会自动检测并安装模块的驱动程序。通常，PCI仪器的成本较低，因为它们依赖于托管它们的PC的电源、处理器、显示器和存储器运行，而不是将以上硬件并入仪器本身中。

PCI Express

PCI Express和PCI类似，是PCI标准的最新版本。因此，前文对PCI的大部分评价也适用于PCI Express。

PCI和PCI Express性能的主要区别在于PCI Express具有更高的带宽，并为每个设备提供专用带宽。在本指南涵盖的所有总线中，只有PCI Express为总线上的每个外设提供专用带宽。而对于GPIB、USB和LAN，带宽均由所连接的外设共享。数据通过点对点连接传输，这些连接称为通道，第1代链路每个方向的传输速率为250 MB/s。每个PCI Express链路可以由多个通道组成，因此PCI Express总线的带宽取决于其在插槽和设备中的实现方式。x1链路(单个通道)、x4链路和x16链路分别提供250 MB/s、1 GB/s和4 GB/s的专用带宽。PCI Express实现了软件向后兼容性，意味着迁移到PCI Express标准的用户可以保留其采用PCI时进行的软件投资。PCI Express也可通过外部电缆进行扩展。

高速的内部PC总线专为快速通信而设计。因此，PCI Express是需要高带宽的高性能数据密集型系统的不二之选，也是集成和同步多种类型仪器的理想选择。

以太网/LAN/LXI

以太网长期以来一直用于仪器控制。它是一种成熟的总线技术，已广泛用于测试和测量以外的其他许多应用领域。100BASE-T以太网理论上的最大带宽为12.5 MB/s。千兆以太网或1000BASE-T将最大带宽提高到125 MB/s。在所有情况下，以太网带宽都在网络上共享。125 MB/s千兆以太网的速度在理论上要比高速USB快，但是当多台仪器和其他设备共享网络带宽时，该性能会迅速下降。以太网总线基于消息进行通信，其中通信数据包显著增加了数据传输的开销。因此，以太网的延迟性能在本指南所介绍的总线技术中最差。

尽管如此，以太网仍然是创建分布式系统网络时的上佳之选。在不使用中继器的情况下，以太网的工作距离长达85 m到100 m，在使用中继器时不存在距离限制。它与控制PC或平台的分离距离是其他总线无法比拟的。与GPIB一样，以太网/LAN无法进行自动配置。您必须以手动方式为仪器分配IP地址和进行子网配置。与USB和PCI一样，以太网/LAN连接也普遍应用于现代PC中。这使得以太网非常适用于分布式系统和远程监测应用。它通常与其他总线和平台技术结合使用，用以连接测量系统节点。这些本地节点本身可以由基于GPIB、USB和PCI的测量系统组成。物理以太网连接的可靠性高于USB连接，但不如GPIB或PXI。

LXI (LAN在仪器领域的扩展)是一种基于LAN的新兴标准。LXI标准定义了采用以太网连接的独立仪器的规范，增加了触发和同步特性。

尽管从概念上说，将某个总线或通信标准指定为最终或理想技术较为方便，但历史经验表明，多种替代标准可能会持续共存，因为每种总线技术都有其独特的优点和缺点。表4汇总了前文所述各种总线的性能标准。我们应该清楚，没有一种总线的所有性能指标都优于其他总线。

您可以创建混合测试和测量系统，将来自模块化仪器平台(如PXI和独立仪器)的组件相结合，并通过GPIB、USB和以太网/LAN进行连接，从而充分发挥多种总线和平台的优势。创建和维护混合系统的一个关键是，实现一个能够透明地识别多种总线技术并利用开放的多供应商计算平台(例如PXI)的系统架构，以实现I/O连接。

成功开发混合系统的另一个关键是，确保在驱动程序、应用程序和测试系统管理级别选择的软件具有模块化设计。虽然有些供应商可能为特定仪器提供垂直软件解决方案，但最有用的系统架构是将软件功能分解为可互换的模块化层，从而使系统既不依赖特定硬件，也不依赖特定供应商。这种分层方法有助于最大限度地实现代码复用和模块化，并能延长使用寿命。例如，虚拟仪器软件架构(VISA)是一种不依赖于供应商的软件标准，用于配置和编程基于GPIB、串行(RS232/485)、以太网、USB和/或IEEE 1394接口的仪器系统以及对其进行故障排除。VISA是一款非常有用的工具，因为对于各种通信接口而言，用于编程VISA函数的API都是类似的。

借助混合系统，您可以结合许多类型仪器的优势，包括传统设备和专用设备。虽然找到通用的仪器解决方案非常有吸引力，但在现实中，您需要采用合适的仪器和相关总线技术来满足特定的应用需求。

总线性能对比

	带宽(MB/S)	延迟(µs)	距离(M)(不带延长器)	设置和安装	连接器坚固性
GPIB	1.8 (488.1) 8 (HS488)	30	20	好	最好
USB	60 (USB 2.0)	模拟输出	5	最好	好
PCI (PXI)	132	0.7	内部PC总线	较好	较好 最好(PXI)
PCI EXPRESS (PXI EXPRESS)	250 (x1) 4,000 (x16)	0.7 (x1) 0.7 (x4)	内部PC总线	较好	较好 最好(PXI)
以太网/LAN/LXI	12.5(快速) 125(千兆比特)	1,000(快速) 1,000(千兆比特)	100 m	好	好

定时和同步

PXI平台是一种模块化的标准测试和测量平台，为集成定时和在仪器之间同步提供了一个很好的例子。PXI Express保留了原始PXI规范提供的10 MHz背板时钟，以及单端PXI触发总线和长度匹配的PXI星形触发信号。PXI Express还在背板上增加了100 MHz差分时钟和差分星形触发器，以提供更高的抗噪能力和行业领先的同步精度(分别为250 ps和500 ps的模块间延迟差)。NI定时和同步模块旨在充分利用PXI和PXI Express机箱中的高级定时和触发技术。

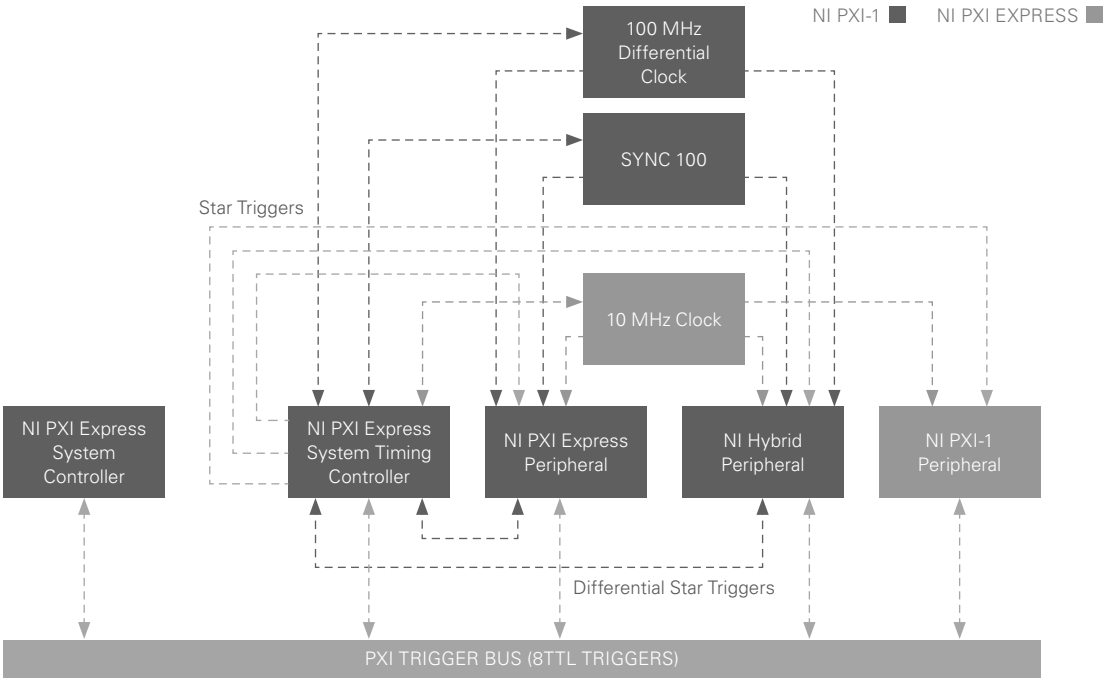
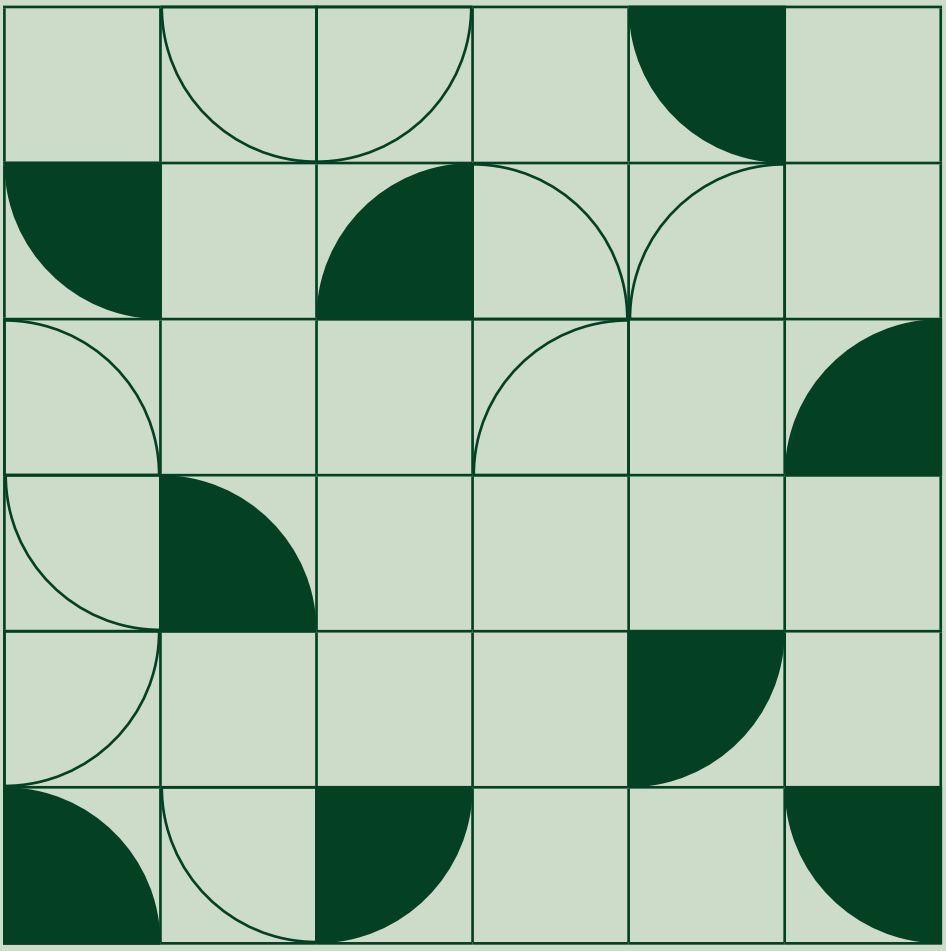


图 8 PXI机箱定时和同步特性示例

附加信息

阅读“仪器基础知识”系列白皮书，了解更多关于使用测试和测量仪器的基础知识。该系列白皮书涵盖从模拟采样理论到接地考虑因素等主题，旨在提高测量质量。

自动化测试系统的电源规划



引言

为自动化测试系统或自动化测试设备(ATE)供电不同于为个人计算机(PC)和台灯供电。测试系统由许多异构的内部组件构成，其中一些组件需要大电流和大功率，并且这些系统通常需要部署到全球各地具有不同电源规格且质量不一的设施中。此外，测试系统的许多组件来自多个供应商，并且必须由测试工程师集成，因此供电问题势必更加复杂。不过，如果遵循电源布局和设备选型方面的最佳实践，挑选适合的组件并作出适当的设计决策则要简单得多。

组件的功率需求有时可能会超出配电系统提供的功率，而精心设计的电源布局可以有效避免这一瓶颈问题，从而确保所有组件正常运行。特别是在组件可能因为功率不足而危及整个系统的运行时，更应当注重电源布局的设计。本指南列出了创建电源布局的步骤和注意事项，以此介绍测试系统的电源规划。

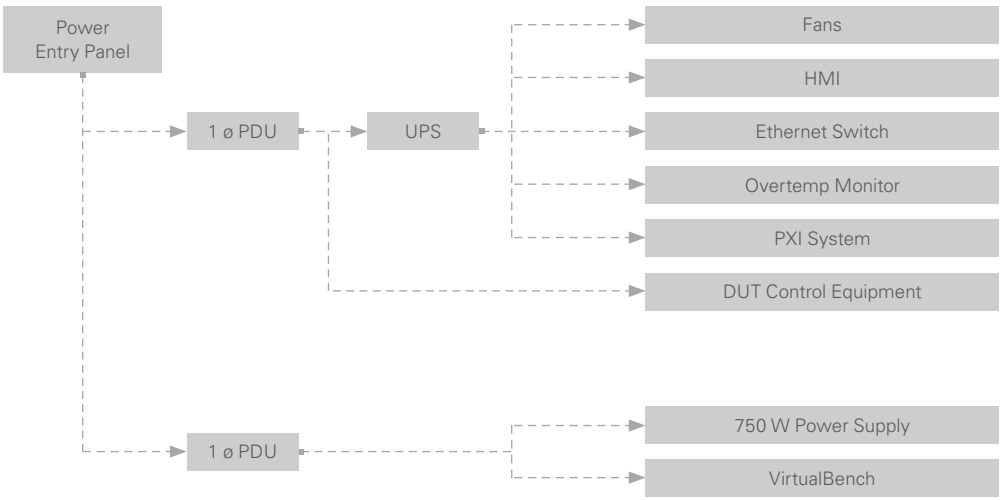


图1 电源布局包括测试系统中的所有设备，并呈现从电源到测试系统再到最终用电设备的功率流。

为系统接入电源

为ATE系统接入电源的最佳做法是使用电源输入面板或电源进线面板。这样可以将内部电源接线与主电压的施加点隔离开来。使用电源输入面板，可以为测试系统配备合适的电源连接器，即额定电压和电流能够满足系统供电需求的电源连接器。NI电源输入面板支持多种连接器类型和额定功率，可满足各种电源要求和地理位置要求。图1给出了电源面板连接器的示例。优质的电源面板还应内置电路保护(包括断路器和保险丝)，以保护系统免受电源浪涌或选用电源不当所带来的损害。更高级的电源面板还会内置电磁干扰(EMI)滤波器、浪涌抑制器以及用于将信号传递到系统的其他连接功能。

Low-Power Configuration
- IEC 60320 (C20)
- 1ø (16 A) 100-240 V

Medium-Power Configuration
- IEC 60309
- 1ø (32 A) 100-240 V

High-Power Configuration
- IEC 60309
- 3ø (16 A) 380-480 V (Red:3P+N+E)

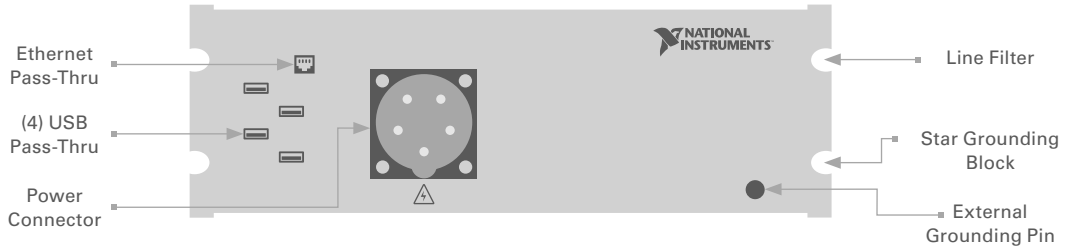


图2 电源输入面板提供将电源接入系统所需的连接功能。电源输入面板可以采用多种标准电源连接器类型中的一种，并且优质的电源面板还具备滤波或急停开关继电器等附加功能。

地理位置考虑因素

为测试系统选择电源面板时，测试仪或测试设备的地理位置是需要重点关注的一个细节。此外，在规划新的测试系统时，还应考虑电源标准和电网基础设施、安全要求以及部署难易程度，这些因素都会受到地理位置的影响。

电网标准

不同国家或地区的公共电网所提供的线路功率有所不同。世界各个国家或地区均针对其电网中的RMS电压、交流电频率、连接器和电流范围制定了相关标准。

公共电网具有以下几种电源配置：

- 单相电源由一根用于传导交流电的火线和一根零线组成。这种线路的电压一般从100 V到240 V不等。例如，日本的线路电压为100 V，而输送电压在220 V和240 V之间。美国和加拿大公共电网的输送电压为110 V到120 V。
- 两相电源也称为分相电源，由两根火线和一根零线组成，两根火线以给定正负偏移电压供电。在美国，两相电源通常为120 V，两根火线之间的相位差为180度。由于两根火线分别传输120 V和-120 V的电压，可以分别将两根火线与零线搭配使用，形成两个120 V的单相电源，也可以使用两根火线，形成一个240 V的单相电源。
- 三相电源由三根火线和一根零线组成，三根火线相互之间的相位差为120度。美国大多数建筑使用208 Y/120 V电源，通过三根火线传导120V电压，电源电路输出恒定，为208 V。许多工业建筑使用480 Y/277 V，可提供大型机械所需的480 V电压。

全球部署

测试系统的设计和部署地点往往不同，甚至涉及多个地理位置。将单个系统部署在多个地理位置会为系统引入一系列新的要求。将系统部署到马来西亚与将系统部署到研发地的工厂甚至是研发机构所在大楼是截然不同的。例如，在底特律的一家研发机构开发一款汽车引擎控制单元测试系统，但却要将其部署在墨西哥的工厂。在设计该系统时应考虑墨西哥的电网标准和质量，并在系统发运之前确认系统满足在墨西哥部署所需的所有安全和监管认证。设计在全球部署的测试系统时，需要考虑以下事项：

- 电网电压标准和配置
- 电网质量和可靠性
- 材料合规性，如是否符合RoHS标准
- 能源合规性，如是否CE、PSE或KC认证标准
- 贸易合规性和进出口法规

如果计划将测试系统部署到测试系统研发地以外的国家或地区，则需要事先了解测试系统部署地的电网电压，以及是否需要对该电压进行转换才能确保测试系统中的设备正常运行。在前文的例子中，测试系统要部署到马来西亚和墨西哥。幸运的是，美国和墨西哥的电网均提供110 V–120 V和60 Hz的电源。如果测试系统是在德国设计但要部署到墨西哥，则情况会略微复杂，因为这两个国家的市电电压并不相同。

电源转换器和不间断电源(UPS)可以帮助调节标准电源，从而满足系统需求。例如，如果测试系统中包含仅能接受120 V电压的设备，可能就需要配备电源转换器，将230 V单相电源转换为该设备所需的单相120 V电源。当然，最好是直接评估和选择支持全球输入电压的设备，以免麻烦。

认证

许多国家都有特定的电气安全标准，例如欧洲CE、日本PSE或韩国KC。电气测试设备的合规性测试通常涵盖辐射水平和频率、触摸安全以及浪涌保护。之所以必须获得这些认证，是因为只有获得认证才能将系统部署到相应国家或地区，或者证明设备能够在工厂中运行。在任何国家或地区运行测试系统时，都需要调查该国家或地区所需的认证。如果未获得认证，将来使用测试系统可能会面临诸多问题。各组件也只有获得这些认证后才能进口，因此如果未获得相应认证，部件将很难更换或维修。

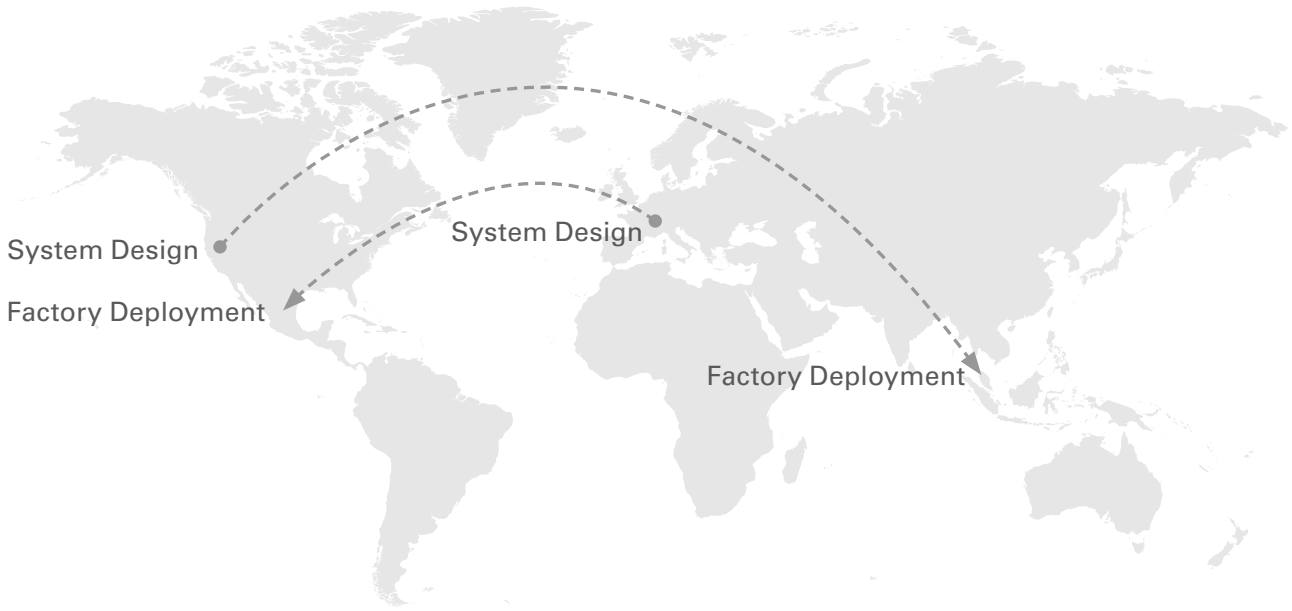


图3 如要将测试系统部署在多个国家或地区，就需要灵活地进行系统设计。在开发这类测试系统时，务必要事先考虑相关国家或地区的电源标准和认证。

电磁干扰或线路滤波器

电网承载的高能信号通常会发射电磁噪声。电源线产生的大多数噪声相对一致，可以提前进行相关规划。但是，世界上没有完美的电网，电源信号中很可能会存在一些非标准噪声。非标准噪声可能会影响系统中的仪器所进行的测量，甚至导致系统违反认证要求。为了保护测试系统免受来自于输电线路的意外噪声源的影响，最常用的方法是使用EMI和线路滤波器。线路滤波器必须在一定电压和电流下工作，并且滤波的信号也有一定的频率范围。例如，线路滤波器的最大电压和电流可以为250 V、10 A，工作频率范围为150 kHz至1 MHz。确保根据测试系统的功率选择适合的线路滤波器来滤除不需要的噪声频率。NI电源输入面板配有EMI/线路滤波器，可为敏感的测量设备提供保护。

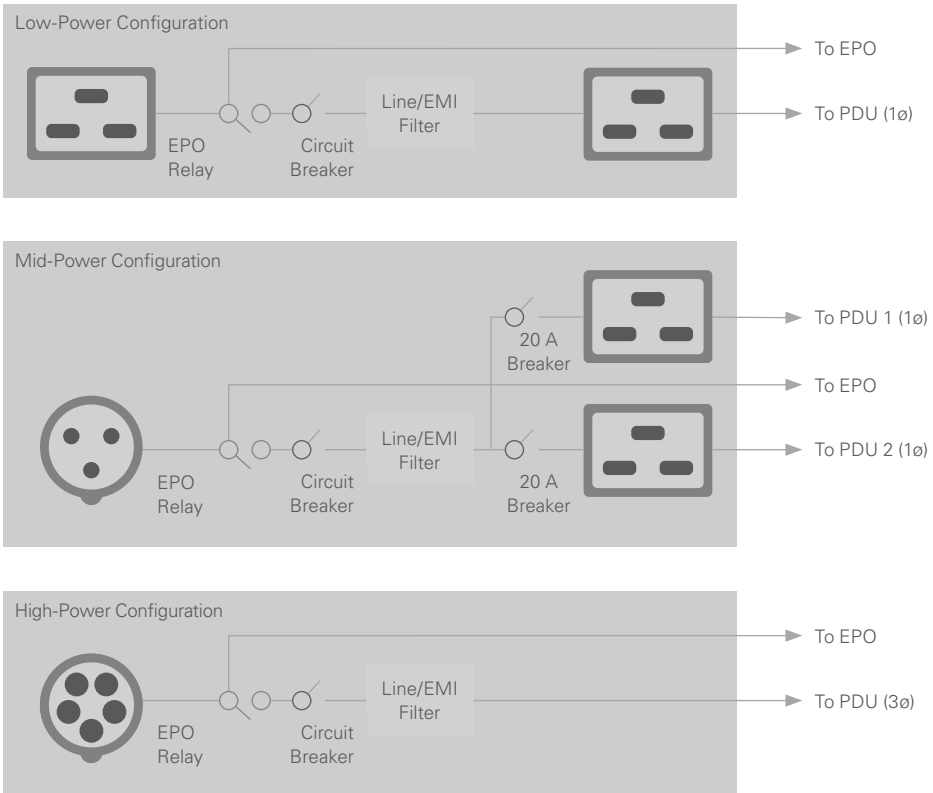


图4 断路器和线路/EMI滤波器对于保护测试系统中的设备以及确保仪器正常运行和精准测量至关重要。图中给出了电源输入面板的低功率、中功率和高功率配置示例。

功率预算

功率预算是测试系统资源和组件规划过程中的关键环节。任一给定设备必须能够在正确的电压下获得适量的电流。功率预算必须针对整个系统以及系统内的每个配电点执行。在通过计算确定所需的功率量后，可以对这些计算出的值应用一些标准规则，从而优化测试系统中的功率分配。

系统功率预算

在确定系统功率预算时，首先需要确定测试系统中所有设备的最大功率需求。需求总和中应包含测试系统中所有组件的预期指标，包括电压、电流和功率瓦数。在许多情况下，功率预算中最重要的指标是电流。由于系统中的给定传输线路只能流过一定量的电流，因此通常必须使用配电单元(PDU)仔细分配在整个系统中的电流。

给定设备的功耗通常在用户手册中写明，有时还会涉及不同条件下的多种功率需求。在某些情况下，设备标注了典型功耗和最大功耗(即最坏情况下的功耗)规范。最好使用最大功率需求作为相对保守的安全值，然后再减去给定的百分比(通常为30%到40%)，得到较为符合实际情况的功率值。图5显示了独立式仪器集成到测试系统中时的最大功率需求。

VirtualBench多功能一体式仪器指定了用电量较大时所需的最大功率，而不是典型功率或平均功率

⚠ 注意:如果未按《NI VB-8034 安全、环境、法规信息》文档描述的方法使用，VIRTUALBENCH 硬件提供的保护功能可能会下降。

电压输入范围	100 VAC～240 VAC，50/60 Hz
功耗	最高150 W
电源输入连接器	IEC C13电源连接器
断电	可通过交流电源线断开主设备供电。请勿将设备放置于不易断开电源线的位置。按前面板电源按钮不能关闭内部电源。

下面以表1中的测试系统为例，简单快速地加以说明。首先了解测试系统中每台设备的最大功耗。确保考虑到子系统和瓶颈问题。PDU具有最大电流限制(本例中为16A)，因此需要做出相应规划。下一步是基于最大功耗值计算得出所需的典型功耗值，也就是取最大功耗的60%到70%。在本例中，保险起见我们使用70%，则该测试系统的功耗约为1,920 W。另外，最好在该值的基础之上再增加约20%，以便将来可以为系统扩展或添加新功能，而不必再添加电源。

设备	最大功耗	所用平均功耗	110 V时的电流
PDU 1			
风扇	50 W	35 W	0.03 A
HMI	100 W	70 W	0.06 A
以太网交换机	25 W	17.5 W	0.02 A
过热监视器	10 W	7 W	0.01 A
PXI系统	526.9 W	369 W	3.4 A
DUT控制泵	1,000 W	700 W	6.4 A
PDU 1总计		**1,198.5 W**	**11.0 A**
PDU 2			
VirtualBench	150 W	105 W	1.0 A
750 W电源	1,100 W	770 W	7.0 A
PDU 2总计		**875 W**	**8.0 A**
系统总计		**2,073.5 W**	**19.0 A**

表1 开始计算功率预算时，首先要了解系统所有组件的最大功耗，乘以平均功耗利用率，然后将它们相加在一起。记住要考虑瓶颈问题和子系统

以下三个简单的最佳实践可以显著简化功率预算：

01 基于每个组件所需最大功率的约60%到70%计算系统功率需求。

02 在规则1最终得出的功率计算值基础之上再增加约20%作为安全缓冲量，用于应对用电量大的时段以及测试系统未来的任何必要的扩展。

03 请记住，有些组件通过PDU和UPS连接，因此大型系统需要使用电源子系统。

子系统功率预算

上述功率预算的计算过程中省略了一个步骤，即如何将大型测试机架中的子系统考虑在内。子系统可以是大型测试系统中设备的任何设备子集，全部共用一个公共电源，比如使用单组PDU的多个仪器或PXI等模块化仪器系统。

模块化仪器的优势在于可以简化电源管理。如果PXI机箱中包含的所有仪器在测试系统中都是相互独立的，则必须单独将各个仪器考虑在内。PXI机箱为其包含的所有仪器提供高质量的安全电源，并提供多种电源和仪器插槽选项。

在将PXI系统纳入功率预算时，可选择以下两种方案：

- 使用PXI机箱规定的整个PXI系统的最大功耗。例如，PXIe-1085 PXI机箱的最大功耗为791 W，在乘以平均利用率70%后，可得出功耗预算为554 W。
- 将PXI系统中所有模块的最大功耗相加，得出非常准确的功率预算值。有关详细计算PXI系统功率预算的示例，请参见图5。

另外，模块化仪器系统明显地比传统的仪器组合更高效，因为它避免了测试系统内需要安装冗余的监视器和冷却系统等由测试系统内部供电的共用组件。

在以下准确计算PXI系统功率预算的示例中，使用的是系统吞吐量为24 GB/s的PXIe-1085 PXI机箱，其中包括一个PXIe-8880 PXI控制器、六个PXIe-4139精密系统源测量单元(SMU)、两个PXIe-5162 PXI示波器，一个PXIe-6570数字模式仪器、两个PXIe-4081 7 ½位数字万用表(DMM)和四个PXIe-2527多路复用开关模块。有关如何计算PXI系统功率预算的说明，请参见图6。

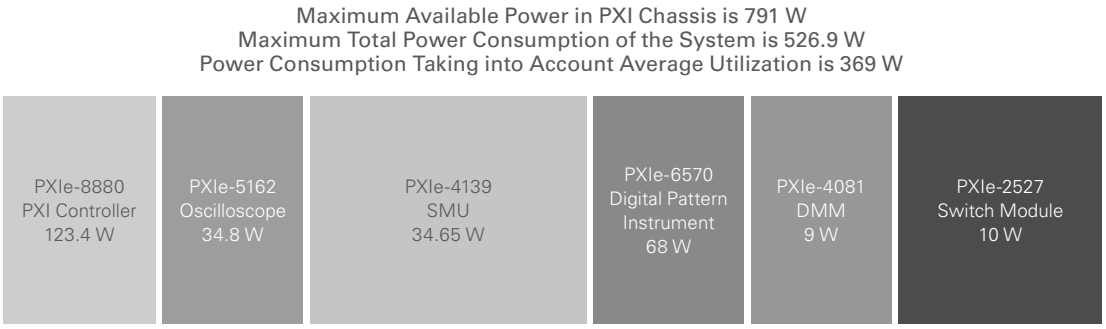


图5 PXI机箱的总功耗是机箱中所有模块的功耗之和。如上图所示，一个完整的仪器机箱在最坏情况下的功耗为526.9 W。

配电单元

PDU主要用于获取输入电源信号并将其分配到多个电源输出口，从而为系统的各个组件供电。PDU的这些内部电源插座具有额定电压和电流，通常既可用于交流电也可用于直流电。

最佳PDU选项具有诸多特性：

- 远程开/关使操作员能够通过电源机构和EPO更改电源状态。通过这种方式，操作员可以从一个方便的位置完全控制系统状态。操作员还可以通过本地和全球EPO机构禁用系统中的电源。
- 内置保险丝等电路保护装置，可以保护贵重但脆弱的设备免受意外电力事件的影响，从而节省数万甚至数十万美元。
- 插座组排序可以确保特定设备在其他组上电之前先上电。例如，连接到外部控制器的PXI机箱或从另一个主PXI机箱扩展的PXI机箱需要在主机控制器之前启动。在这种情况下，PDU应在启动包含主PXI机箱的插座组之前启用包含从PXI机箱的插座组。
- 通过多个组来处理一定量的功率有助于平衡PDU上的功率负载，防止出现可能损坏测试系统中设备的过流情况。例如，有一个PDU包含三组电源插座，每组可提供16 A电流，这样就可以防止连接到PDU的任何一台设备的电流超过16 A。这也意味着必须注意跨多个组分配设备所需的电流。
- 直流电源可以为状态LED或冷却系统等组件供电，这些组件同样需要远程开/关和组排序。其中一些组件甚至在“电源启动”系统状态下非常有用，从而需要远程供电功能。

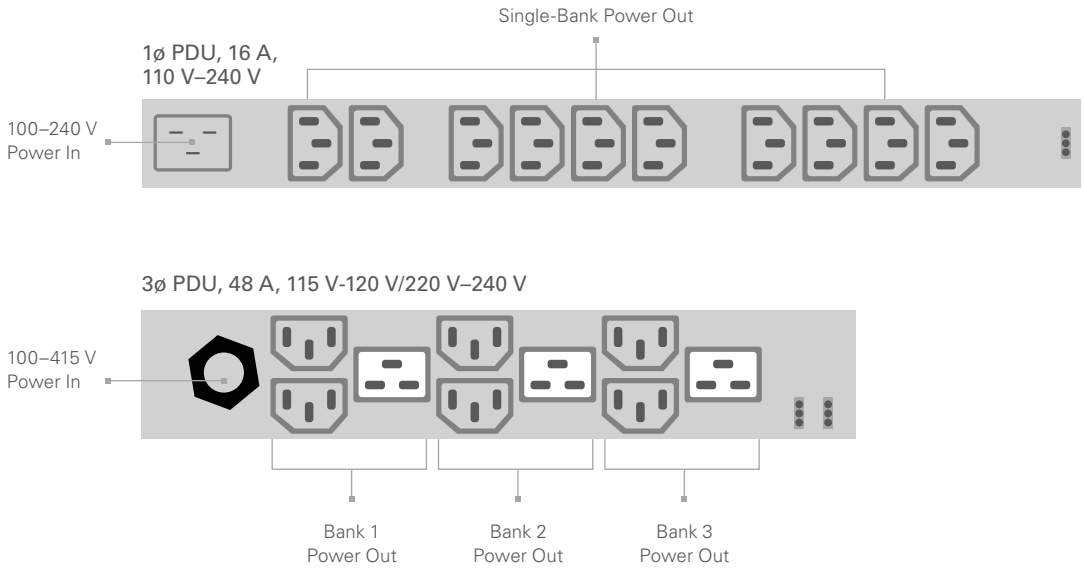


图6 PDU可采用不同的接线方式和架构。上面的PDU只有一插座组，可为设备提供最高16 A的电流，而下面的PDU具有三个插座组，每个组可提供16 A的电流，总共可提供最高48 A的电流。

为测试系统中的关键组件供电

确保测试系统中的关键组件(如主机控制器和敏感仪表等)能够通过UPS供电。测试系统中的有些组件乍看起来不起眼，但其实很重要。例如，如果冷却系统在电源事件后未恢复运行，则主机控制器可能会过热。如果测试系统中的触摸面板显示器断电，技术人员将无法排除故障或记录电源事件数据。仔细思考会发现有些组件需要时刻保持运行状态，甚至是在停电后或紧急情况下也不例外。

系统供电开销和支持

在分配电源时，记得考虑测试系统的开销和基础设施，如温度控制、网络连接和用户界面等。过热或网络连接断开可能会导致生产中断，这与测试仪器发生故障一样具有危害性。

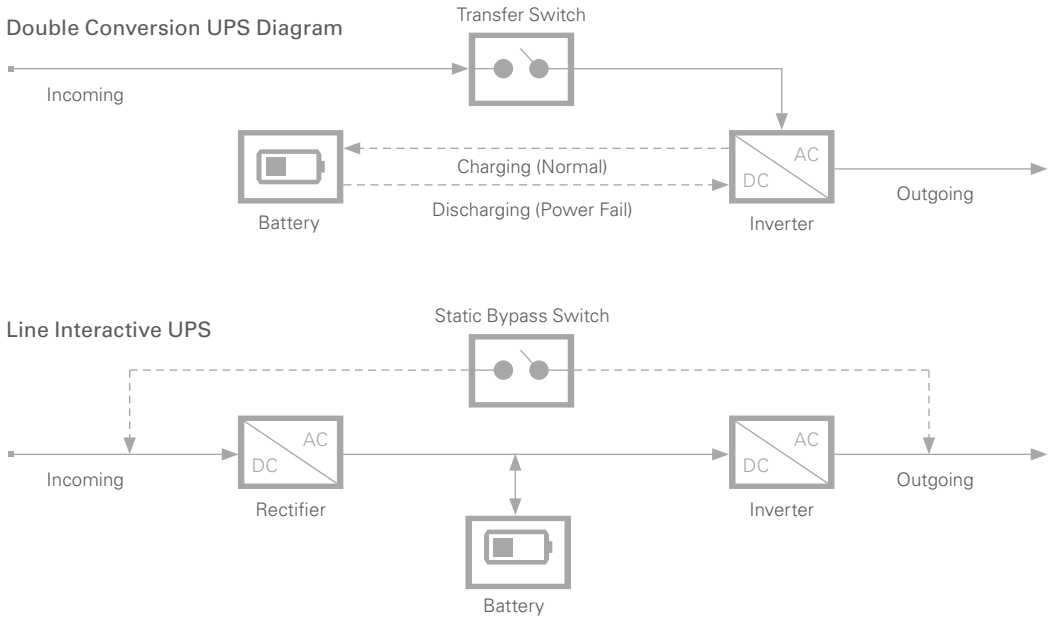


图 7 UPS用于提供清洁、可靠的电源，也允许在停电或掉电的情况下正常关机。双变换UPS始终为电池充电，从而为系统提供稳定的电源。

不间断电源

优秀的测试系统设计工程师会考虑电网的质量，在设计系统时会设法避免在停电或掉电的情况下出现未定义的行为。无论是在上述情况下，还是在正常运行期间，都可以使用UPS为测试系统中的关键组件供电。

UPS可以提供可靠的电压和电流，并且还可以在发生停电或严重掉电后充当电池电源。UPS是确保测试系统坚固耐用的关键组件，在电网不可靠的区域更是如此。

UPS主要有以下两种类型：

- **在线互动式UPS**—在在线交互式UPS中，有源线路输入直接连接到电源输出。UPS随后会监视输入功率，以确保其不会低于给定阈值。如果输入功率下降幅度过大，则会切换到电池，电池随后通过反向运行UPS来为输出信号供电。在这种情况下，测试系统在运行期间无需任何调节即可接收线路功率，并且UPS会在电源出现故障时为其供电。
- **双变换UPS**—双变换UPS将输入电源线连接到电池，电池会持续充电，然后为UPS的输出线路供电。双变换UPS的电源非常稳定，因为它是通过板载电池供电。双变换UPS还有一项额外优势，电池总是充满电，可以随时用作备用电源。如发生停电或严重掉电，无需切换电源即可使关键系统正常关闭。尽管双变换UPS的效率略低，但能够始终在测试系统内提供稳定、准确的电源，因此成为ATE应用的理想之选。

电源质量和可靠性

世界上没有完美的电网，但大多数电气设备的设计都是以在理想的电源条件下运行为前提。当电网提供的功率与设计的系统功率不同时，系统就不能按预期运行，仪器可能会出现测量效果不佳或输出信号不正确的情况。设备和系统可能会意外开启和关闭，导致重要设置丢失或默认设置不正确。这种意外行为可能导致测试结果不理想、待测设备(DUT)受损，甚至更严重的后果。双变换UPS还有一项额外优势，即持续进行滤波，使用输入电源为内部电池充电，然后提供高度可靠、清洁的电源。

停电和掉电

当电网供电完全关闭时，就会发生停电。在电网发达的情况下，很少会出现停电，当停电时，通过两种方式管理系统的行为：(1) 由电池为系统中的部分或所有组件供电，使其运行一小段时间，以便正常关闭；(2) 因停电而直接关闭。

掉电和电涌在电网中更为常见，尤其是在工厂等耗电量大的设施中，并且更难以处理，因为它们会导致系统出现不确定的行为。掉电可能是电网中的电压或电流骤降或出现毛刺，导致输入到测试系统的功率降低。浪涌是指电网的电压或电流瞬间高于正常水平的情况。

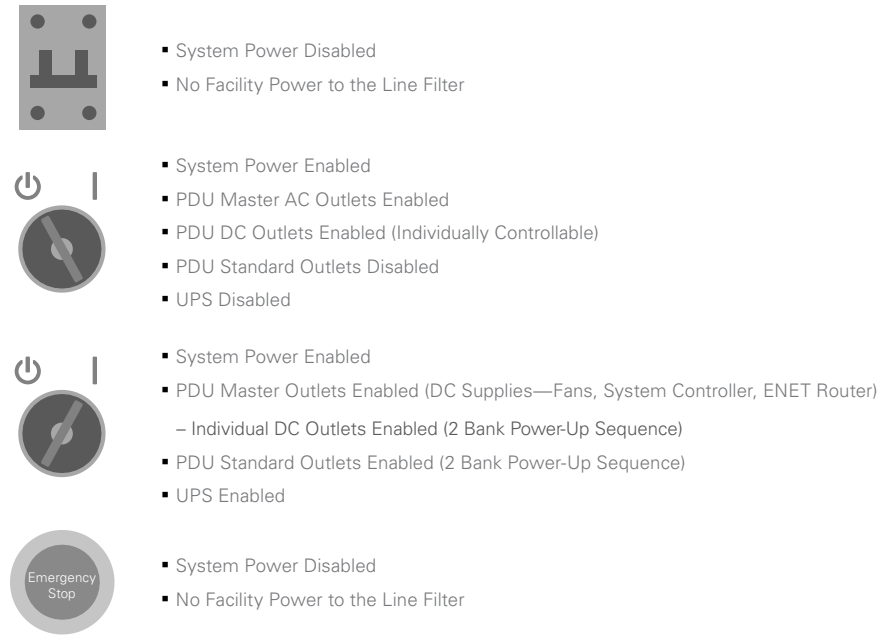
UPS内部配有电池，可以在发生停电或严重掉电后为测试系统中的必要设备提供充足的电源，为新电源(如发电机)上线供电留出时间。必要设备包括主机控制器和用户界面以及任何其他关键设备。在由电池供电的这段时间内，系统可以保存必要的数据，避免损坏或出现不安全的软件状态。

电源状态

测试系统通常需要具有多种运行状态，以便实现调试和维护、省电和安全等目的。

优秀的测试系统设计方案应能实现四种运行状态：

- **关闭**—系统完全禁用，没有电力通过线路滤波器或测试系统的任何内部组件。
- **启用**—电力通过线路滤波器进入任何直接供电的设备。通常，所有设备都通过PDU供电。在启用状态下，只有PDU的主插座可能被激活。在某些情况下，PDU上的直流电源也会激活，用于为系统支持组件及其他组件供电。例如，在启用状态下，以太网路由器和实时系统控制器可以会通电，以便技术人员可以监测测试系统的运行状况。
- **开启**—切换到该状态后会进入测试系统的主上电序列。所有PDU都会接收电力并输出其他系统设备。在许多情况下，当某些系统组件必须在其他组件启动后才能开始运行时，分阶段启动电源序列是有帮助甚至是必要的。有关PDU的更多信息，请参见“电源布局”部分。
- **紧急断电(EPO)**—当用户或系统监视器识别到不可接受的工作条件时，EPO会立即切断测试系统的电源。



- System Power Disabled
- No Facility Power to the Line Filter

- System Power Enabled
- PDU Master AC Outlets Enabled
- PDU DC Outlets Enabled (Individually Controllable)
- PDU Standard Outlets Disabled
- UPS Disabled

- System Power Enabled
- PDU Master Outlets Enabled (DC Supplies—Fans, System Controller, ENET Router)
  - Individual DC Outlets Enabled (2 Bank Power-Up Sequence)
- PDU Standard Outlets Enabled (2 Bank Power-Up Sequence)
- UPS Enabled

- System Power Disabled
- No Facility Power to the Line Filter

图 8 测试系统需要具有多种电源状态，包括关闭、启用、开启和EPO，以确保高效运行。

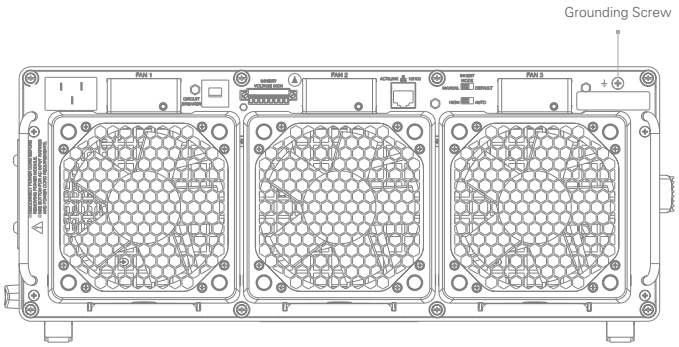


图9 PXIe-1085 PXI机箱有一个接地螺钉，用于将机箱和所有仪器直接连接到外部接地模块。为保障安全，最好将机架中的所有设备都接地。

尽管上述接地路径通常即可满足需求，但如果将测试系统中的每个设备都单独接地，则可以保障安全。NI的电源输入面板有一个星形接地块，如图2所示，它连接到整个机架的所有其他接地模块。然后，电源输入面板外部的接地螺柱可以连接到机箱外部真正的接地线。此外，每台设备通常都有一个可以直接接地的接地螺柱。NI PXI机箱的接地螺钉如图9所示。将每个设备都连接到分布在整个机箱中的接地模块，可确保每个设备安全接地，并且所有接地引线都非常短，这有助于降低电磁噪声。

确保尽量缩短与接地平面的电气连接。接地回路较长会产生驻波，导致系统内出现射频辐射。如果需要使用较长的传输线连接到接地平面，应采用双绞线配置将信号与接地信号耦合，以此降低电磁噪声。如果是浮地或不以大地为参考，则应纳入信号的正负基准地。

紧急断电

当测试系统遇到严重问题或设施内发生紧急情况时，操作员需要能够干净利落地关闭测试系统的电源。测试系统内置紧急断电(EPO)机构，简化了连接并避免了直接开关电源。操作员可以使用EPO重置处于错误状态的系统，防止DUT受损甚至可以防止对自己造成伤害。EPO功能也是IEC和UL等安全标准机构的要求。

EPO通常是一种方便使用的物理机构，例如按钮或开关，操作员只需按下它即可切断测试系统中所有设备的电源。理想情况下，EPO面板会与测试系统中的所有设备连接，以确保所有设备都能快速关闭。大多数EPO都会将系统置于关闭状态，需要将系统重置为启用状态后，才能重新激活系统并且为所有设备供电。这样可以防止系统在断电后在存在非安全条件的情况下意外重启。

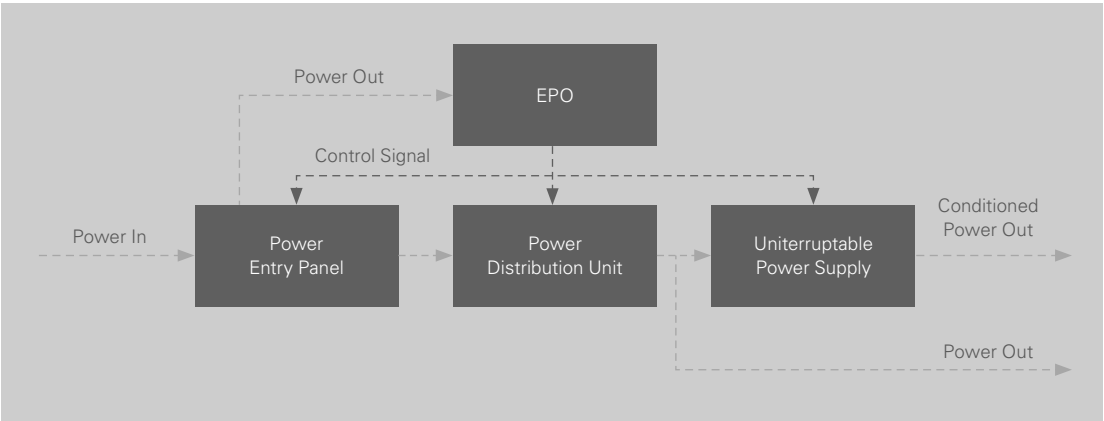


图10 EPO连接到测试系统中的所有设备，必要时可禁用所有连接的设备以保持安全。

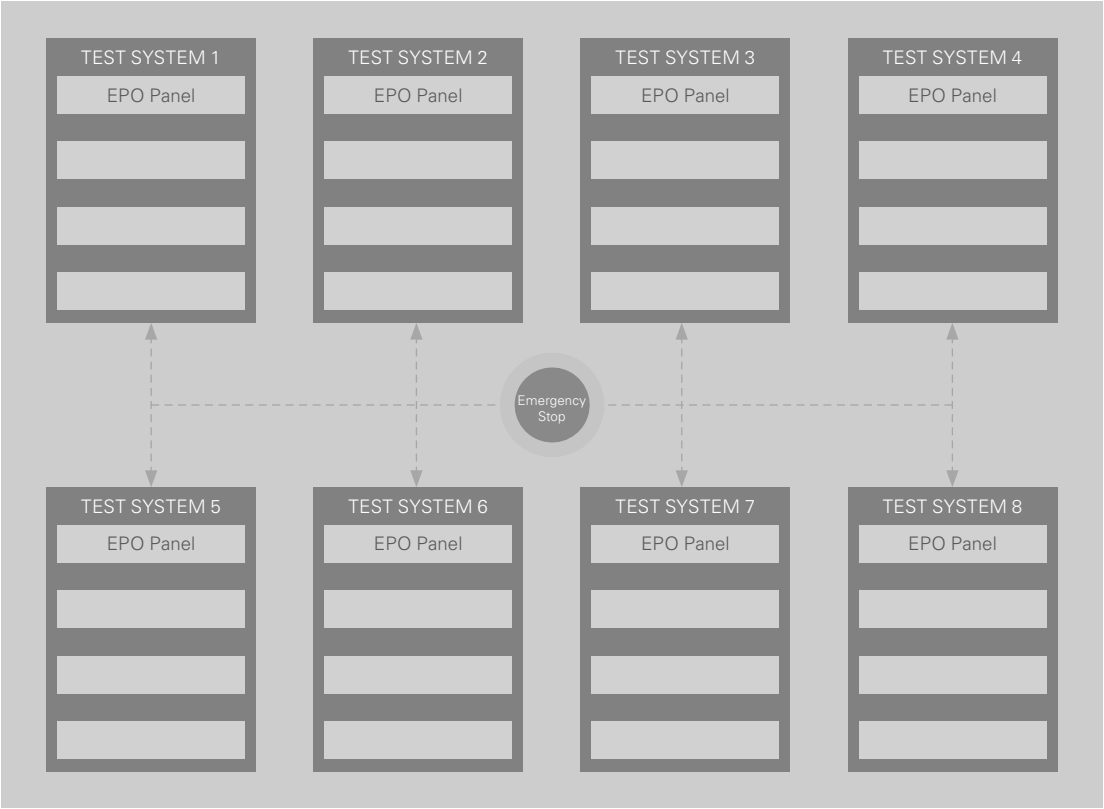


图11 在某些情况下，需要使用全局EPO来禁用设施中的所有测试系统及设备。全局EPO是一个断电机制，可启用各个系统本地的EPO。

接地

接地是测试系统设计的关键部分，原因主要有两点：安全性和测量质量。

确保测试系统妥善接地可保障安全，这意味着测试系统中所有设备的电流都可以通过适当的路径流向真正的地。电源输入面板必须连接到妥善接地的电源。之后，应能够选择测试系统中的任一台最终耗电设备，并按照其接地路径返回电源输入面板。以太网开关的接地电流路径应遵循图1所示的测试系统电源布局。以太网交换机地连接到UPS地，UPS地应连接到PDU地，PDU地应连接到电源输入面板的地。接地回路会形成一条流向地的电流路径，防止在系统中积聚危险电荷，避免产生的电弧造成DUT受损或操作员触电。

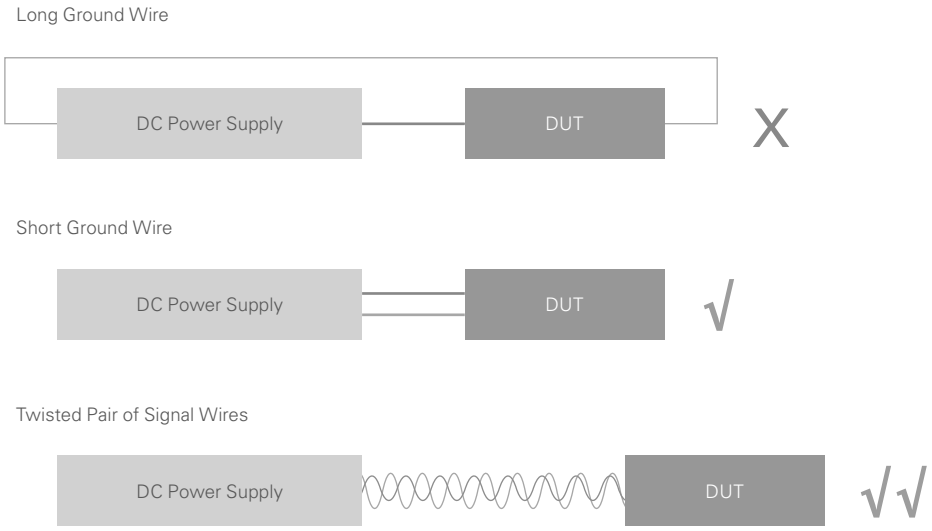


图 12 如果系统中的接地线较长、不匹配，可能会形成较长的接地回路并充当噪声信号的天线。最好使用短一些的接地线，但仍有可能无法避免噪声。为实现最佳性能，应在系统中使用双绞线作为信号线和接地线。

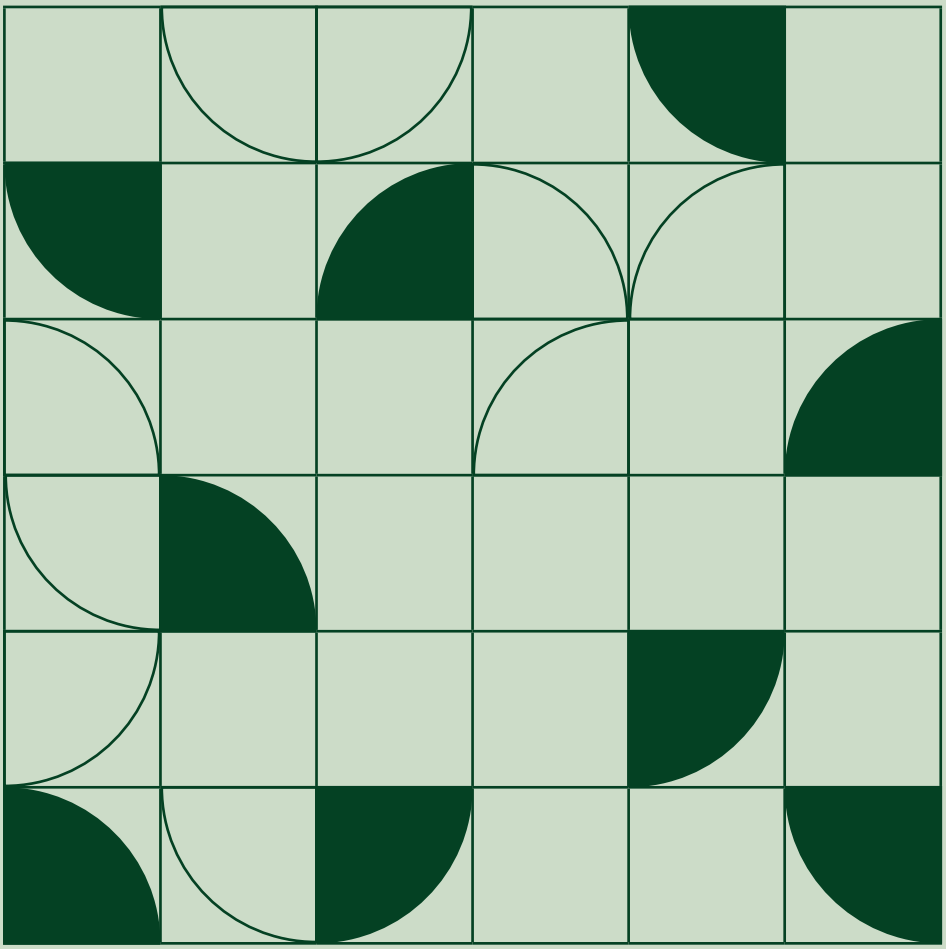
有关针对具体测量建立正确连接所需的全部信息，请参见白皮书《现场接线和噪声注意事项综合指南》。

组件采购最佳实践

材料的采购渠道与测试系统的构建方法可谓五花八门。如果构建的系统生命周期较长，则应考虑采购渠道能否长期提供支持以及系统是否具备应对未来新增需求的可扩展性。要实现这两点，最好从商业供应商处采购系统组件，因为这类供应商可长期为产品和消费者提供支持。人们通常会从供应商处采购PDU、UPS、系统控制器和仪器等组件，但从供应商处采购互连装置和线缆等小型组件可以获得长期回报。值得信赖的连接器供应商和测试仪器供应商会全力为您服务，让您的系统能够运行长达十年之久。

在极少数情况下，测试系统可能会因为一些特殊要求或合乎情理的原因而无法使用市面上的产品，这时需要选择定制设备和解决方案，专攻此类解决方案的公司有许多。但请记住，这类解决方案通常是为个别消费者量身定制的，因此随着时间的推移，很有可能性能会发生变化或变得过时。

开关和多路复用



简介

许多自动化测试应用需要将信号路由至各种仪器和待测设备(DUT)。克服这些应用挑战的理想方法通常是部署一个开关网络来实现仪器和DUT之间的信号路由。开关不仅可用于路由信号，也是增加昂贵仪器仪表的通道数以及提高测量灵活性和可重复性的一种低成本方法。

将开关添加至自动化测试系统时，通常有三个选择：自行设计和搭建开关网络，使用由GPIB或以太网控制的独立式(“台式”)开关盒，或者使用包含数字万用表(DMM)等一种或多种仪器的模块化平台。开关几乎总是与其他仪器搭配使用，因而必须与这些仪器紧密集成。现成的模块化方法可解决大多数常见测试系统固有的集成难题。本指南概述了将开关和多路复用集成到测试系统的最佳做法。

开关架构

开关是扩展仪器通道数的一种经济实惠的选择，但并不一定是最好的选择。

开关架构主要有以下四种类型：

- 无开关
- 仅测试机架采用开关
- 仅测试连接件采用开关
- 测试机架和测试连接件均采用开关

下表概述了所有四种开关架构的优缺点。

低于平均值 ○　平均值 ◒　高于平均值 ●

	灵活性	吞吐量	成本	低电平测量误差(MV、μA、MΩ)
无开关	○	●	○	●
仅测试机架采用开关	●	◒	●	○
仅测试连接件采用开关	○	◒	◒	◒
测试机架和测试连接件均采用开关	●	◒	◒	◒

表 1　各种开关架构的优缺点

无开关

在第一种架构中，待测设备(DUT)与测试系统中的仪器之间未使用开关进行信号路由。此类系统通常为每个测试点都提供一个专用的仪器通道。

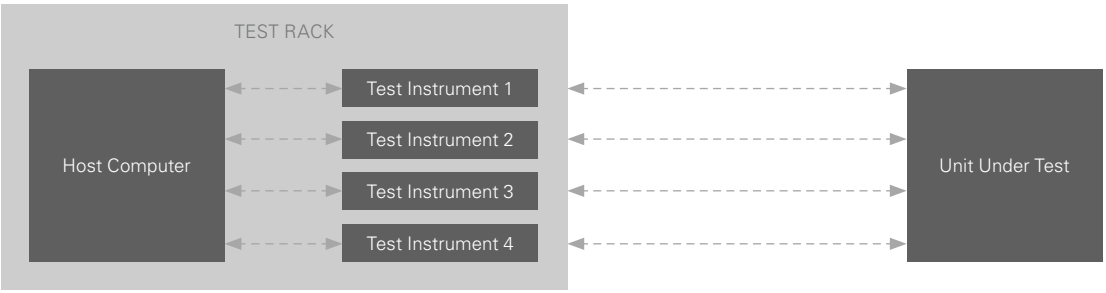


图 1　无开关的测试系统直接将每个仪器连接到待测设备

仅测试机架采用开关

第二种开关架构仅使用商用现成(COTS)开关在测量仪器与DUT之间路由信号。在测试机架中采用开关不仅可以利用现成开关产品，还提供了最简单的扩展途径。挑选COTS开关平台时，务必确保平台能够提供丰富的功能和易于扩展的选项。否则测试系统在使用期间可能需要重新设计，并因此产生大量支出。

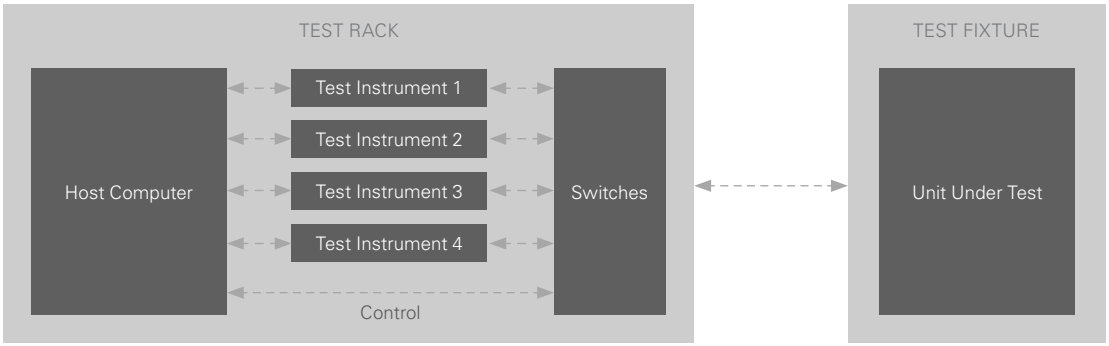


图2 部分测试系统在测试机架内集成了开关以便扩展

例如，PXI平台提供600多种不同类型的开关模块，可以路由高达600 V、40 A和40 GHz的信号。单单NI就提供了100种不同的PXI开关模块，可以配置为200多种不同的开关拓扑。

在测试机架中采用开关的优点

使用COTS开关解决方案可以节省大量开发时间，包括印刷电路板(PCB)的设计和驱动程序的开发。COTS开关还提高了测试系统的可扩展性，因为用户只需从开关供应商处购买额外的模块即可添加更多开关，而不需要重新设计整个测试连接件。

此外，每家开关供应商提供的解决方案都有自身独特的优势。例如，NI开关配有板载EEPROM，可跟踪模块上每个继电器被激活的实例数量，同时还提供其他功能来监视继电器的运行状况，例如功能性和阻性继电器测试。借助这些功能，用户可以预测特定继电器机械寿命终结的时间点，以便进行预测性维护。例如，在维护高通道数开关系统时可能很难进行手动调试，或者在制造生产车间发生意外停机时可能会导致严重且代价高昂的延迟，上述功能在这些情况下会十分实用。

此外，还可以使用NI开关模块来提高测试应用的吞吐量，具体方法是将开关连接列表下载到开关模块的存储器中，然后在开关与任何仪器之间使用双向触发循环该列表，整个过程不会被主机处理器中断。

在测试机架中采用开关的缺点

使用开关通常会减慢测试过程，因为需要按顺序对任何给定DUT上的各个测试点进行测量，而不是像前述的无开关架构一样并行测量。将所有开关置于测试机架内还会增加布线总量。除了开关与测量仪器之间需要布线之外，DUT与开关之间也需要布线。这可能会导致敏感型测量出现误差，例如泄漏电流或低电阻测量。

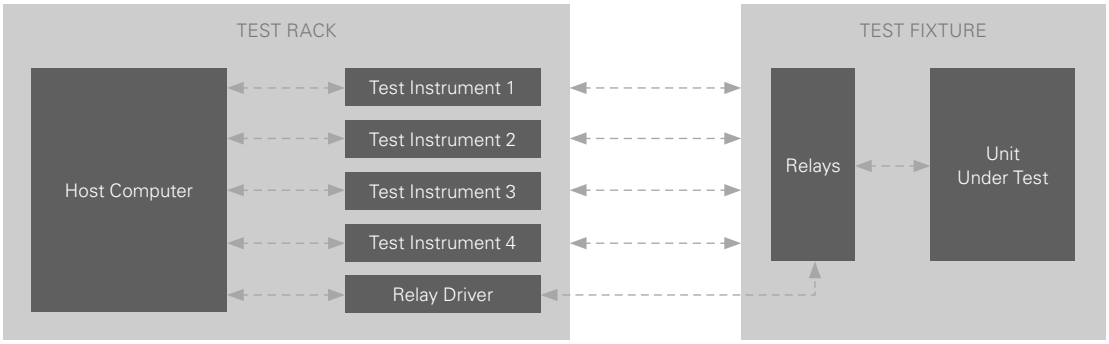


图3 在测试连接件中采用开关的测试系统需配备继电器驱动器

仅测试连接件采用开关

第三种开关架构仅在测试连接件中使用开关。在这种情况下，系统使用位于连接件附近的PCB上或连接件之中的各个继电器将来自测量仪器的信号路由到DUT上的各个测试点。如果使用这种架构，则需要在测试机架中安装一个继电器驱动器，以控制测试程序中的各个继电器。COTS继电器驱动器的典型示例为PXI-2567，它是一个64通道的继电器驱动器模块，允许使用NI-SWITCH驱动器通过标准API来控制外部继电器，而无需自定义编程。此外，也可以设计一个外部电路来驱动继电器，但需要开展额外的设计工作。

在测试连接件中采用开关的优点

如前文所述，在任何位置采用开关都有助于降低测试成本。此外，在测试连接件中采用开关可免除DUT与开关之间的布线。减少布线也有助于降低测量误差。

在测试连接件中采用开关的缺点

如前文所述，使用开关通常会减慢测试过程。此外，在测试连接件中接入自定义开关需要具备PCB设计经验，因此可能并非适用于所有人。在测试连接件中采用开关还会导致测试系统难以扩展，无法满足更多测试点的需求。

无开关架构的优点

线缆和开关通常会降低信号的完整性。如果不使用开关，则可以在信号与测量仪器之间建立更直接的路径，从而提高测量精度。此外，无开关架构还可以提高测试速度。在无开关架构中，每个测试点都有专用的仪器，因此可以并行测量而不必按顺序测量，从而提高测试吞吐量。

无开关架构的缺点

事实证明，为每个测试点配备专用仪器的成本非常高。另一个缺点是可扩展性差。如果构建的测试系统不采用开关，则测试机架中很容易出现空间不足的情况。在这种情况下，测试系统可能需要完全重新设计，随之而来的硬件更改、软件更新和重新验证也都会产生额外的成本。例如，假设有一个测试系统使用20个PXIe-4081数字万用表(DMM)并行测试20个电阻式温度传感器(RTD)。现在，需要扩展该系统以测试40个RTD传感器。为此，需要再添加20个DMM，同时需要再添加20个PXI插槽。但是，如果采用开关，则使用一个PXIe-4081 7½位数字DMM万用表和两个PXI插槽即可按顺序测试所有40个RTD传感器。

什么情况下适合构建无开关的测试系统

如果需要进行极其敏感的测量(添加线缆和开关会导致失真)或者需要尽可能缩短测试时间，通常建议构建无开关的测试系统。例如，一些半导体测试应用会为芯片上的每个引脚专门配备一个单独的参数测量单元或源测量单元，而由于半导体属于大产量行业，累积起来的测试成本会占芯片总制造成本的很大一部分。使用专用仪器通道进行并行测量，可以最大限度地缩短测试时间，从而显著降低测试成本。此外，在半导体行业中，测试仪通常是针对特定的芯片组或芯片组系列量身定制的，因此在其生命周期内通常不会进行扩展。

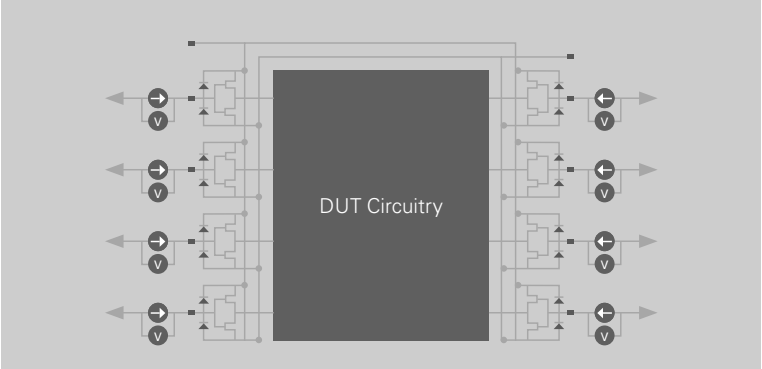


图 4 部分半导体应用使用专用仪器并行测试给定芯片上的所有引脚

这种架构的另一个缺点是需要针对特定的安全与合规性标准设计定制电路板，因此会产生额外的成本。如果测试高压设备，可能需要打造一个符合UL、CE和VDE等各种法规的开关连接件。所用的PCB上的众多继电器必须符合上述各种标准的爬电距离和电气间隙要求，这种设计极具挑战性。在这种情况下，使用COTS开关有助于降低成本。许多COTS供应商均已根据各种安全标准对其模块进行认证。例如，所有额定电压大于60 VDC/30 VAC和42.2 Vpk的NI开关模块都被视为高压设备，因此是根据以下安全标准构建的。

测试机架和测试连接件均采用开关

最后一种开关架构在测试机架和测试连接件中均采用开关。这种架构兼具两种COTS开关解决方案的优点，同时还将开关放置在靠近DUT和测试连接件的位置，最大限度地降低了特定敏感型测量中的误差。利用PXI-2567继电器驱动器和其他基于PXI的开关，用户可以通过完全受支持的标准驱动程序API对整个开关系统(包括测试机架中的COTS开关以及测试连接件中的定制继电器)进行编程。

在测试机架和测试连接件中均采用开关的优点

通过在测试机架中采用COTS开关并在测试连接件中采用继电器，可以构建一个易于扩展的开关系统，同时最大限度地降低关键测量或低电平测量中的误差。使用这种架构，可以将用于路由敏感信号的开关置于测试连接件中，并将所有其余开关置于测试机架中。除了可扩展性之外，使用COTS开关还有助于充分利用不同供应商的特有功能，例如NI PXI开关模块中的继电器计数跟踪和硬件触发功能。

在测试机架和测试连接件中均采用开关的缺点

使用开关通常会拖慢测试进程，因为用户需要按顺序对各个测试点进行测量，而非开展并行测量。此外，在测试连接件中接入定制开关还可能非常耗时，并且需要掌握大量的PCB设计专业知识，尤其是在涉及高压或高频信号时。

图5 NI开关模块符合多种安全与合规性标准

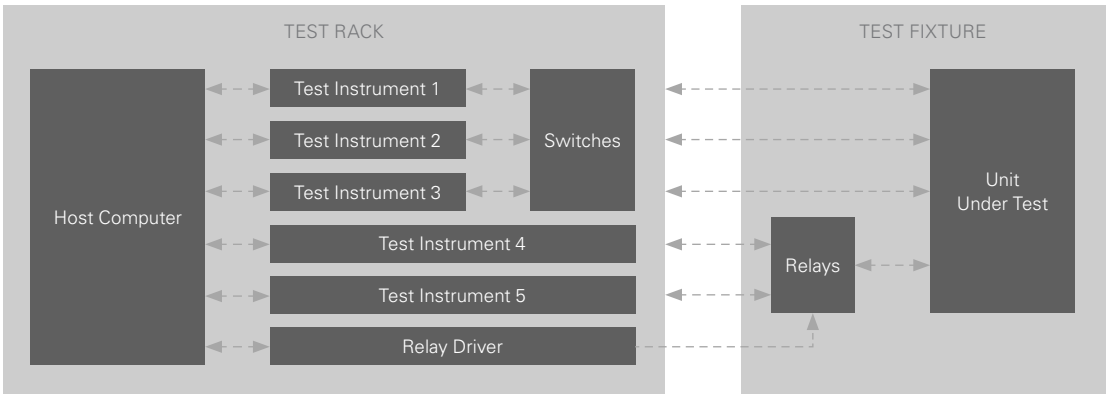


图6 在测试机架和测试连接件中采用开关的测试系统极具灵活性，但需要额外的设计工作

如何针对具体应用挑选最合适的开关

除了开关位置之外，还需要比较各种开关拓扑和继电器类型，以确保开关子系统符合信号要求和测试目标。对于自动化测试应用，“开关”通常指使用继电器在多个DUT与仪器之间切换信号的COTS设备。开关可以采用多种方式排列继电器，形成不同的开关拓扑，例如通用继电器、多路复用器和矩阵。不同的继电器类型在尺寸、信号额定值和预期寿命等方面各有取舍。本部分将介绍常见的开关拓扑、常用继电器类型、主要开关规格，以及在自动化测试系统中使用开关的常用技巧和窍门。

常见开关拓扑

一旦确定您的应用更适合采用有开关的架构，下一步便是选择最合适的开关拓扑，即选择如何排列继电器来构建大型开关网络。大多数开关供应商的开关产品分为三大类：通用继电器、多路复用器和矩阵。某些开关(例如PXIe-2524)具有多种拓扑结构，具体可以在软件中进行配置。PXIe-2524有五种不同的拓扑结构可供选择，能够轻松应对不断变化的需求。选择拓扑结构时，务必要考虑所需的连接总数、最大同时连接数以及将来是否需要针对测试系统的变化进行扩展。

通用继电器

通用开关由多个独立使用的继电器组成。如果只是想闭合/断开电路内的连接或两个可能的输入或输出之间的开关，通用继电器是一个不错的选择。继电器通常按刀掷的数量进行分类。继电器的刀是所有路径的公共端，刀可以连接的位置称为掷。

单刀单掷(SPST)继电器类似于具有闭合和断开状态的标准电灯开关。SPST继电器有两种形式：A型和B型。A型SPST继电器处于常开状态，直到继电器被激活才会使继电器触点接通，形成完整的电路。B型SPST则处于常闭状态，直到继电器被激活才会使继电器触点断开连接，从而断开电路。

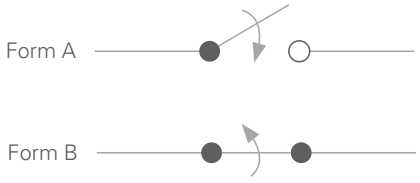


图7 SPST继电器的两种形式：常开(A型)和常闭(B型)

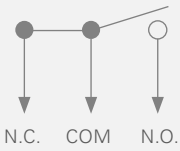
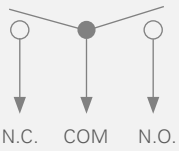
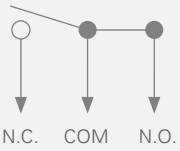
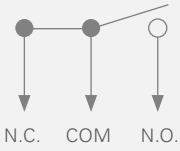
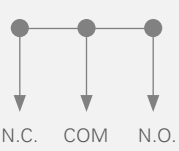
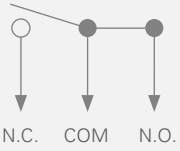
TASK	OPEN	DURING OPERATION	OPERATION COMPLETE
Form C			
Form D			

图 8 SPDT继电器也有两种形式：C型和D型

单刀双掷(SPDT)继电器有一个刀(或公共连接端)，可以在常开触点与常闭触点之间切换。SPDT继电器分为C型和D型两种。C型SPDT在激活时会先断开常闭信号路径，然后将继电器连接到常开触点。这种SPDT继电器操作称为“先开后合”或BBM。D型继电器在激活时会先闭合常开信号路径，然后断开常闭信号路径。这种SPDT继电器操作称为“先合后开”或MBB。

双刀单掷(DPST)继电器会同时驱动两个A型SPST继电器，这两个继电器通常使用相同线圈并封装在一起。当需要同时断开或闭合两条信号路径时，DPST是理想之选。理论上也可以使用两个独立控制的A型SPST继电器构建DPST，但驱动这两个继电器时可能存在一定的时间差。

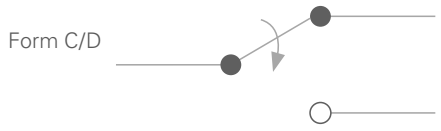


图 9 SPDT继电器在两个掷(连接)之间共享一个公共端(刀)

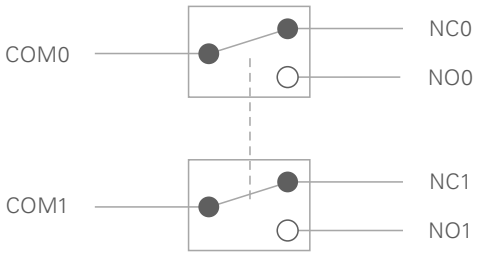


图 10 DPST继电器可同步控制两个A型SPST继电器

多路复用器

多路复用器是多个继电器的一种组合方式，允许将一个输入连接到多个输出，或将一个输出连接到多个输入。多路复用器可有效地将多个DUT连接到单个仪器。但是，使用这种开关架构时必须事先了解哪些DUT连接需要访问各个仪器。

多路复用器有时会使用多个末端相连的A型SPST继电器。这种构建方法简单有效，但缺点是未使用的信号路径会导致交流信号反射，进而降低开关的额定带宽。

多路复用器有时也会通过级联C型SPDT继电器的方式构建，以此确保交流信号的完整性。这种类型的多路复用器通常需要更多的PCB空间，但同时也减少了可能会降低开关带宽的任何桩线或额外的非端接信号路径。

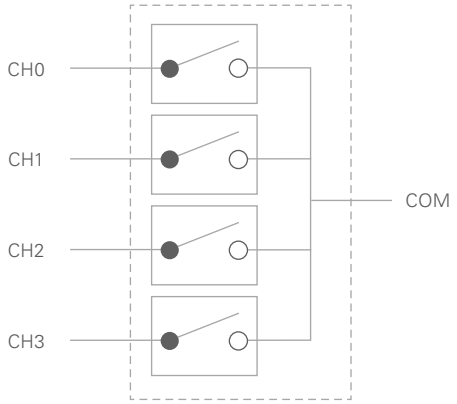


图 11 由多个A型SPST继电器并联而成的4 x 1多路复用器

矩阵

矩阵是最灵活的开关配置。与多路复用器不同，矩阵可以同时连接多条信号路径。矩阵分为行和列，每个行列交叉点上都配有一个继电器，因此可以连接列-列、列-行和行-行的信号路径。借助矩阵的灵活性，用户无需预先定义即可通过各种信号路径将所有开关通道彼此连接。通常建议在硬件规划阶段规划开关路由，但如果使用矩阵，则可以根据测试要求的变化灵活更改开关路由。

矩阵大小通常用M行 x N列(M x N)配置来描述。常见的配置类型包括4 x 64、8 x 32和16 x 16。不过，在大多数情况下，行与列并无明显区别。如果您更习惯从行而不是列的角度进行分析，则可以转置开关矩阵，例如使用64 x 4矩阵代替4 x 64矩阵。

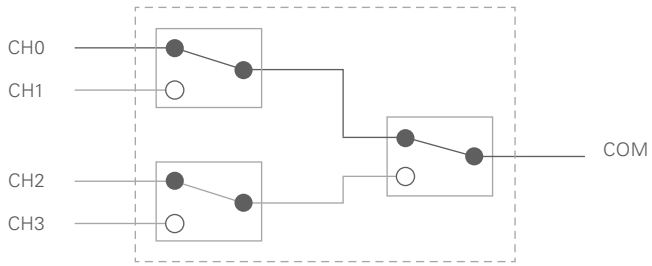


图 12 由多个C型SPDT继电器级联而成的4 x 1多路复用器

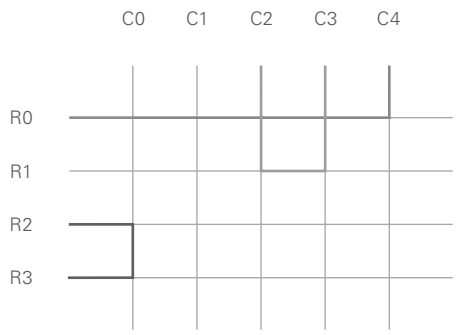


图 13 矩阵可最大程度确保信号路由的灵活性

其他拓扑

绝大多数开关都属于通用继电器、矩阵和多路复用器，不过也有其他专用的开关拓扑，例如稀疏矩阵或故障插入单元(FIU)。

稀疏矩阵是一种混合方案，介于矩阵和多路复用器之间，通常用于射频(RF)应用。通过将两个多路复用器的公共端连接在一起，可以构建包含多个行和列的伪矩阵，但在任何给定时间点只能连接一条信号路径。多路复用器的通道密度通常比矩阵高，因为矩阵的每个行列交叉点都至少需要一个继电器。因此，虽然稀疏矩阵在给定空间内的通道密度相对较高，但每次仍只能在行列之间连接一条信号路径。在某些交流应用中，信号带宽可能会受到传统矩阵的非端接行和列所形成的桩线影响，这种情况下也非常适合使用稀疏矩阵。

另一种专用开关架构是故障插入单元(FIU)，通常用于硬件在环(HIL)测试系统。对于负责测试嵌入式控制单元可靠性的测试系统而言，硬件故障插入(也称为故障注入)是一项关键的考虑因素，嵌入式控制单元必须能够对故障条件做出已知且可接受的响应。为此，应在测试系统的I/O接口与ECU之间插入FIU，以便测试系统可以在正常运行状态与故障状态(例如对电源短路、对地短路、引脚对引脚短路或开路)之间切换。有关FIU的更多信息，请阅读《使用故障插入单元(FIU)进行电子测试》白皮书。

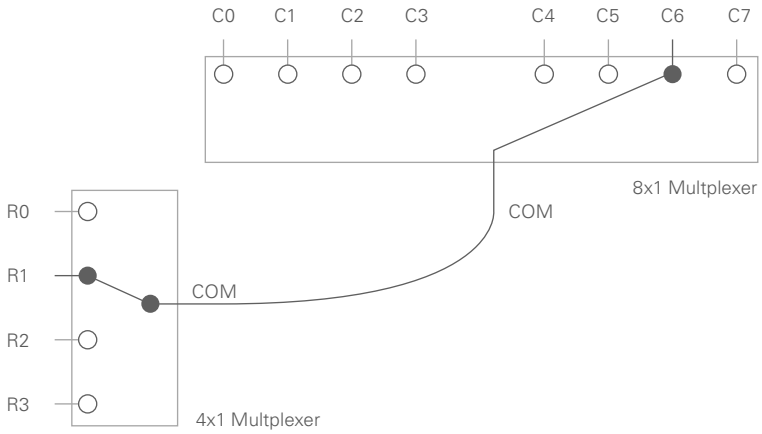


图14 稀疏矩阵是通过将两个或多个多路复用器的公共端连接在一起构建而成，通常用于切换RF信号

继电器类型

继电器是用于闭合或断开电路连接的远程控制设备。市面上有很多类型的继电器，但最常见的主要有四种：机电继电器、簧片继电器、固态继电器和场效应晶体管(FET)继电器。每种继电器类型都会在开关系统的性能、成本、预期寿命和密度之间进行不同程度的权衡取舍，因此务必要根据应用需求来挑选最合适的继电器类型。

请注意，单个继电器和成品开关产品的规格在大多数情况下是不同的。继电器规格(例如带宽、额定功率和接触电阻)仅适用于单个继电器，不包括将继电器连接到开关拓扑的PCB布线或为开关拓扑提供用户接口的连接器。例如，单个继电器可能具有0.05 Ω的接触电阻和300 V的额定电压，但成品开关产品可能具有更大的路径电阻(例如1 Ω)，其中包括多个继电器和复杂的PCB走线，并且单个继电器可能不需要满足在300 V电压下安全操作开关产品的PCB爬电距离和电气间隙要求。

机电继电器

机电继电器(EMR)或电枢继电器使用流过电感线圈的电流来感应磁场，以此将电枢移至断开或闭合位置，从而使两个触点接触形成电路。EMR有多种类型，例如锁存型和非锁存型，但其在操作上的差异很小。非锁存EMR使用单个线圈，在电流停止流动后会返回到其默认位置。而锁存EMR在电流停止流动后仍会保持在当前的位置。有些锁存EMR使用单个线圈，通过反转电流来反转磁场方向，从而将电枢推拉到所需的位置。其他锁存EMR使用电枢任一侧的线圈来推动电枢断开或闭合。

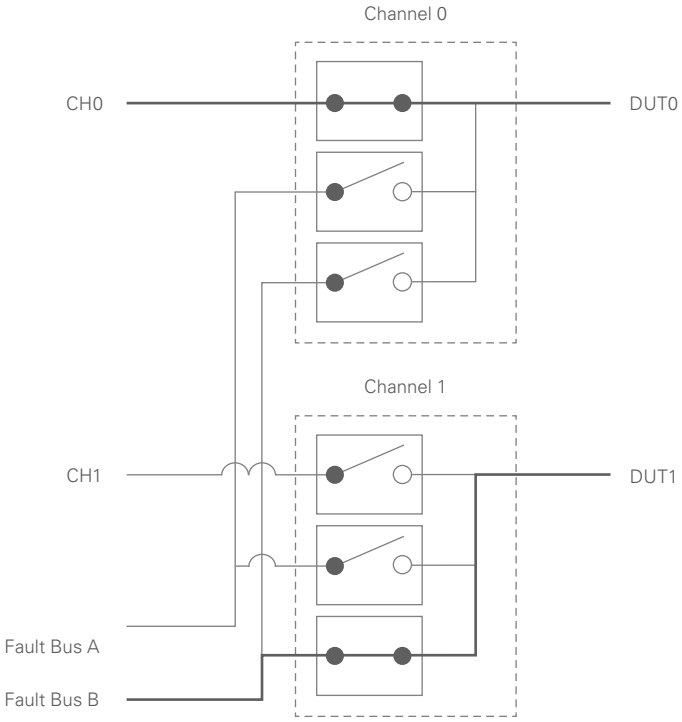


图15 FIU支持自动化故障条件测试，通常用于测试嵌入式系统(例如汽车ECU)的可靠性

EMR支持低电压/电流、高电压/电流以及直流到GHz频率等各种信号特性。此外，EMR具有较低的接触电阻(通常远小于1 Ω)，并且可以应对意外的浪涌电流和最高300 W的功率。因此，几乎所有用户都可以找到符合测试系统所需信号特性的EMR。但是，EMR会占用大量的PCB空间，与其他继电器类型相比速度较慢(150次关断/秒)，并且部件的可移动性会导致生命周期较短(最多10^6次关断)。

鉴于这些优缺点，如果需要大功率、高电流或高带宽的耐用型继电器，同时又不介意速度较慢和需要定期更换(因为EMR的性能会随时间的推移而降低)等问题，EMR会是一个不错的选择。

簧片继电器

簧片继电器同样使用流经电感器的电流来产生用于连接物理触点的磁场。但是，簧片继电器的触点远比EMR更小、更轻。簧片继电器的线圈缠绕在两个重叠的铁磁性叶片(称为簧片)上，簧片密封在充满惰性气体的玻璃或陶瓷胶囊中。当线圈通电时，两个簧片被拉到一起，触点互相接触，在继电器内形成信号路径。当电流不再流过线圈时，簧片的弹簧力会将触点分离。

由于簧片继电器的体积更小，因而可以在更小的空间内安装更多继电器，并且其开关速度高于EMR，最高可达2000次关断/秒。此外，继电器中有限的移动机械部件和隔离的环境可实现更长的机械寿命，最多支持10^9次关断。

但是，由于触点尺寸较小，簧片继电器无法处理太大的功率，并且更容易因自热或电弧而受损，导致簧片出现小面积烧坏。如果两个相互连接的簧片在熔化后又发生固化，则触点可能会焊接在一起。在这种情况下，继电器将始终保持闭合，但如果弹簧力足以将两个簧片拉开，则会折断其中一个簧片。为了防止损坏，需要监测可能由于热切换容性负载而引起的大浪涌电流信号，并使用内联保护电阻来降低电流尖峰的电平和持续时间。有关保护簧片继电器的更多信息，请阅读《簧片继电器保护》白皮书。簧片继电器兼具小巧尺寸与高速特性，因此成为许多应用的理想之选。簧片继电器更常见于矩阵和多路复用器模块，而不是通用开关模块。其中一款典型的产品为PXI-2530B，它是一种COTS开关，可以通过切换各种前置接线端子配置为13种独特的矩阵或多路复用器拓扑。

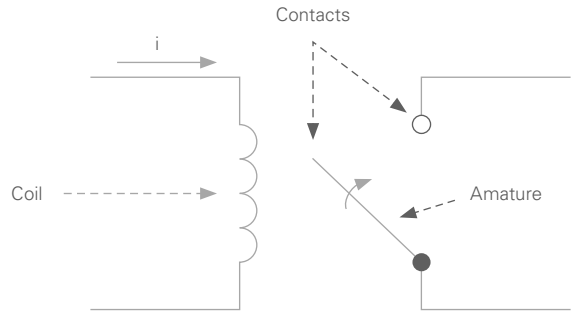


图16 单线圈机电继电器使用磁场来断开和闭合机械开关

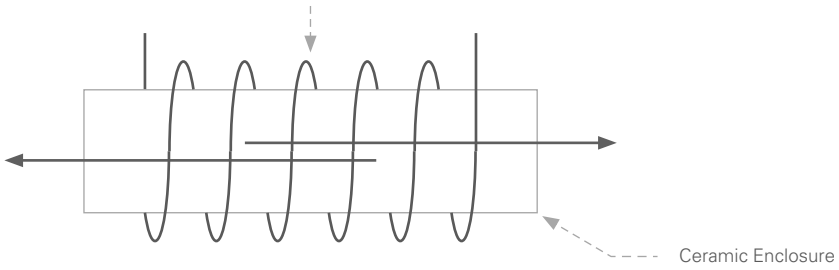


图17 当电流停止流过线圈时，簧片的弹簧力将触点分开

固态继电器

固态继电器(SSR)属于电子继电器，其内部组件包括用于响应输入的传感器、将电源切换到负载电路的固态电子开关装置以及无需机械部件即可激活控制信号的耦合机构。这种继电器通常使用具有LED的光敏金属氧化物半导体场效应晶体管(MOSFET)器件来驱动设备。

SSR的速度略快于EMR，高达300次关断/秒，因为SSR的开关时间取决于点亮和熄灭LED所需的时间。由于没有机械部件，SSR不太容易因为受到物理振动而损坏，因而具有无限长的机械寿命。

但是，SSR也有缺点。首先，SSR不如EMR稳定，如果信号电平超出其额定值，则容易损坏。其次，SSR较为昂贵，并且产生的热量比其他类型的继电器高。最后，由于SSR通过晶体管进行连接而不是采用物理金属连接，因而路径电阻较大，最小也要接近1Ω，最大则可能超过100Ω。大多数新式SSR都针对路径电阻进行了优化，以减小其影响。

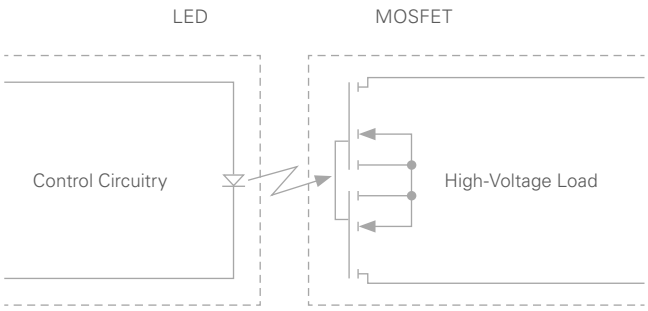


图18 SSR使用具有LED的感光MOSFET来驱动设备

当使用中小信号电平，且需要一个可以持续多个周期的继电器时，具有无限长机械寿命的SSR将是绝佳之选。COTS SSR开关的一个示例为PXI-2533，它是一个4 x 64矩阵，额定开关功率为55 W，并且具有无限长的机械寿命。

FET继电器

与SSR类似，FET继电器不是机械装置，而是使用晶体管来路由信号。与SSR不同的是，控制电路直接驱动晶体管的栅极而不是驱动LED。

直接驱动晶体管栅极可实现远超上述其他类型继电器的开关速度，最高可达60,000次关断/秒。此外，由于不含机械部件，FET继电器的体积比机电或簧片继电器小得多，并且不易受到冲击和振动问题的影响，具有无限长的操作寿命。然而，FET继电器的路径电阻比任何其他继电器类型都高得多，通常在8Ω至15Ω范围内，并且缺少物理隔离，因此仅适用于低电平信号。

低于平均值 ○ 平均值 ◒ 高于平均值 ●

功能	电枢继电器	簧片继电器	FET继电器	SSR继电器
大功率	●	◒	○	◒
高速	○	◒	●	◒
小尺寸封装	◒	●	●	●
低路径电阻	●	◒	○	◒
低电压偏移	◒	○	◒	●
更长的使用寿命	○	◒	●	●

表2 不同继电器类型的对比

对于低电平信号以及需要快速操作继电器或无限长机械寿命的应用而言，FET继电器是绝佳之选。COTS FET开关的一个示例是PXI-2535，它是一个4 x 136矩阵，能够在不到16 µs的时间内执行继电器操作。

开关扩展

如果自行创建开关拓扑，则可以创建一个精确满足应用尺寸需求的矩阵或多路复用器。但是，许多客户选择使用具有固定尺寸的COTS开关来减少开发工作。因此，了解如何组合多个矩阵或多路复用器以创建更大的矩阵或多路复用器非常重要。

另一种方法是增加一个多路复用器来连接多个多路复用器的公共端，多路复用器的内部结构仅允许有一条通往公共端的通道路径，因而需要更多的多路复用器。但是，这种方法仍然会形成PCB桩线，导致带宽性能下降。

多路复用器扩展

扩展多路复用器通道数的最简单方法是直接将多个多路复用器的公共端连接在一起。这种方法存在多个输入通道同时短路的风险，可能会造成硬件损坏。因此，需要确保在任何给定时间点只有一个通道连接到公共端。可以使用某些开关软件(如Switch Executive)定义软件排除条件，防止有多个输入路径同时连接到公共端。这种方法的另一个缺点是未使用和未端接的路由会形成桩线，从而增加电容并降低高频性能。

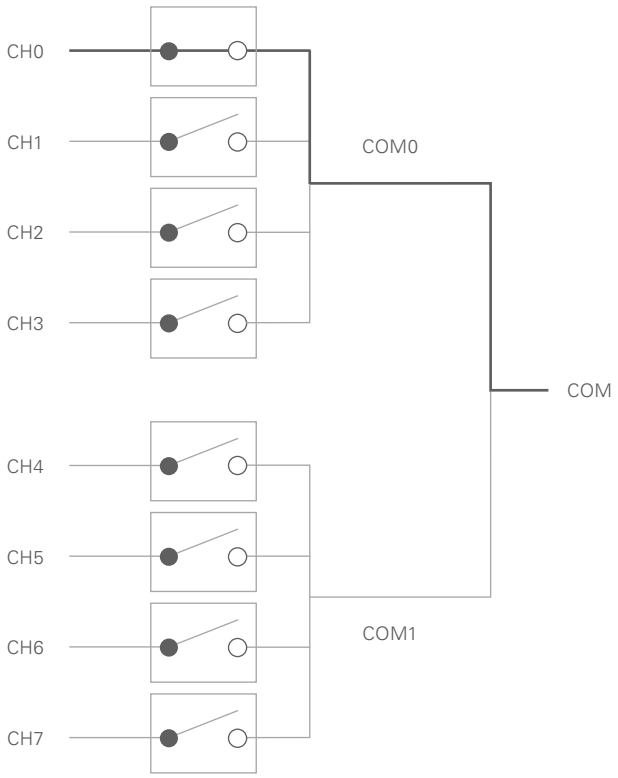


图19 通过将两个4 x 1多路复用器的公共端连接在一起，形成一个8 x 1多路复用器

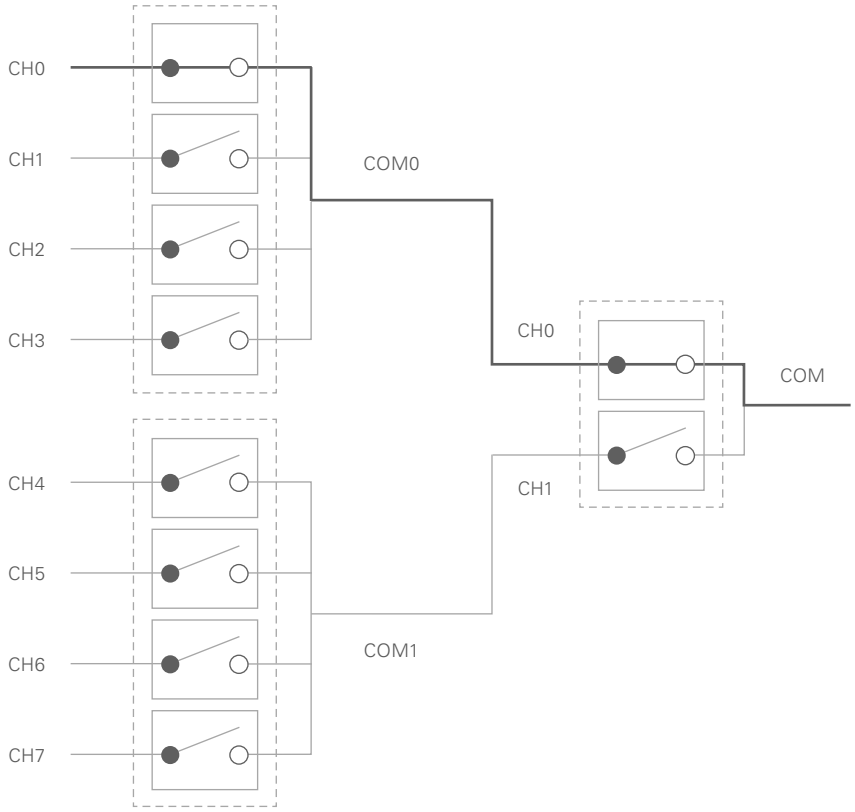


图 20 通过增加一个多路复用器来切换两个4 x 1多路复用器的公共端，形成一个8 x 1多路复用器

对于高频应用，应使用C型SPDT继电器来创建大型多路复用器。此选项可确保有效信号路径上不存在任何桩线，有助于提高开关的带宽。

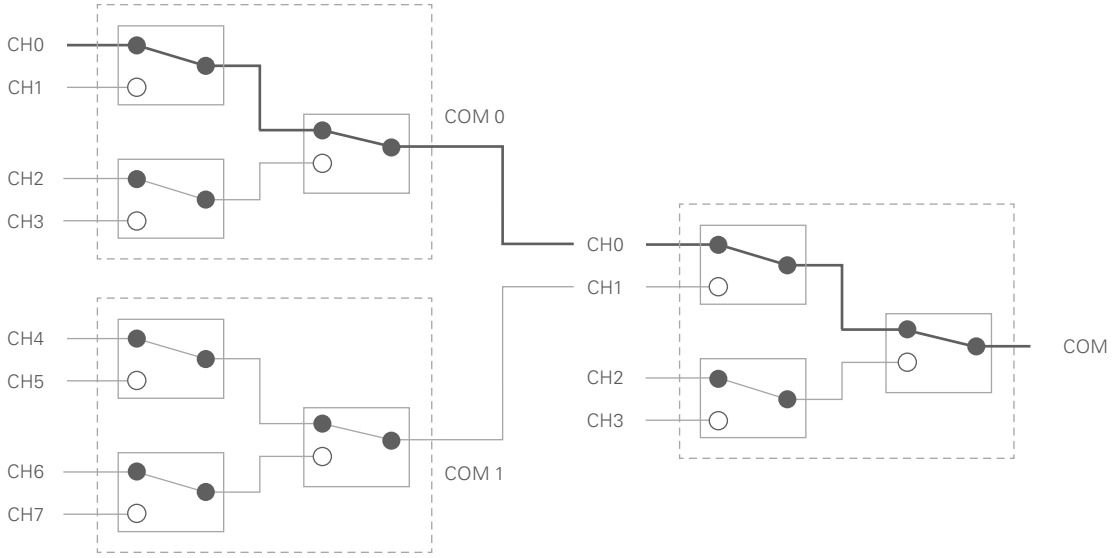


图 21 通过级联三个采用C型SPDT继电器的4 x 1多路复用器，形成一个8 x 1多路复用器

矩阵扩展

开关矩阵还可以作为构件，用于创建尺寸远大于单个COTS矩阵开关的大型矩阵。扩展矩阵的方法有两种。列扩展是指连接两个或多个矩阵模块的每一行，使扩展矩阵的列数加倍。行扩展是指连接两个或多个矩阵模块的每一列，使扩展矩阵的行数加倍。对于低电平信号以及需要快速操作继电器或无限长机械寿命的应用而言，FET继电器是绝佳之选。COTS FET开关的一个示例是PXI-2535，它是一个4 x 136矩阵，能够在不到16 µs的时间内执行继电器操作。

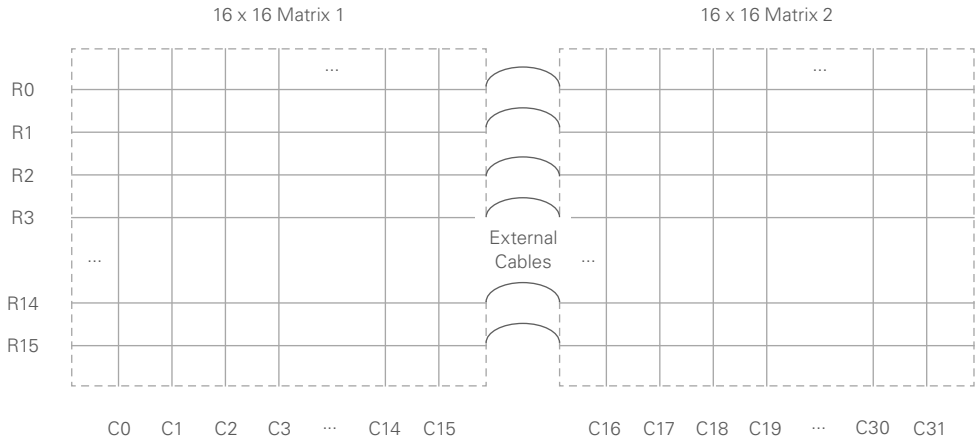


图22 通过列扩展创建的16 x 32矩阵

为了简化矩阵扩展，一些COTS矩阵开关(如PXIe-2532B)提供专用电缆，可轻松连接多个开关模块的行来组合矩阵。不过，即使没有预先提供附件，所有矩阵也都支持扩展。如果要手动扩展矩阵，可使用外部导线连接各个矩阵的行或列。有关矩阵扩展的更多信息，包括示例和常见问题，请阅读《PXI开关模块的矩阵扩展指南》。

主要开关规格

除了继电器类型和开关拓扑之外，还必须确保开关子系统所连接的信号的完整性。大多数开关按照信号类型分为两类：低频/直流和RF开关。

低频/直流开关规格

开关通常会标出额定电压和电流，但还应注意最大开关功率，也就是触点可以开关的功率上限。例如，150 V 2 A开关的功率上限为60 W，所以不得同时使用150 V电压和2 A电流(300 W)。因此，除了最大电压和电流之外，务必还要考虑信号的最大功率。

在处理开关时，信号频率也是一个棘手的问题。简单的正弦波经常使用基频来描述信号。但对于方波或具有尖锐边缘的信号，则需要注意，方波具有远高于基频的谐波频率，可能会形成尖锐边缘。如果打算切换方波，请选择一个额定切换频率为信号基波频率7到10倍的开关。例如，如果使用额定频率为10 MHz的开关路由10 MHz方波，输出信号的波形将更接近正弦波而不是方波。

有关开关带宽的更多信息，请阅读《开关带宽选择》白皮书。

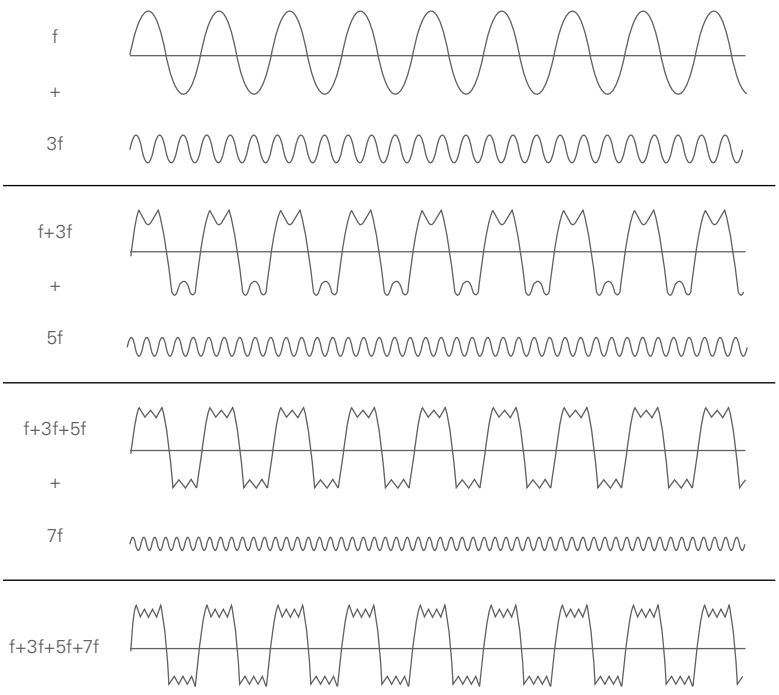


图 23 方波信号具有远高于基频的谐波频率

开关路径电阻、热EMF和失调电压会影响低电平信号测量，例如DMM电阻测量。因此，应该选择对测量影响最小的开关，并采用适当的测量技术来抵消这些误差。

有关在切换低电平信号时如何减少误差的更多信息，请参见以下白皮书之一：

第一部分：切换低电压信号时如何减少误差

第二部分：切换低电流信号时如何减少误差

第三部分：切换低电阻信号时如何减少误差

RF开关规格

额定频率大于10 MHz或20 MHz的开关一般称为RF开关。RF开关通常具有较低的信道密度，可有效保持信号完整性，因此适用于需要较高带宽的信号路径。但是，在挑选RF开关时，仅知道拓扑和带宽信息还不够。

所有RF开关都具有额定特性阻抗，这一传输线参数能够反映出传播信号在信号路径中的传输或反射情况。组件制造商在设计设备时会专门将特性阻抗设计为50 Ω或75Ω，因为RF系统中所有组件的阻抗必须互相匹配才能最大程度地减少信号损耗和反射。市场上的RF系统多使用50Ω的特性阻抗，如大多数通信系统。75Ω RF系统较为少见，主要用在视频RF系统中。请务必确保电缆和连接器等组件以及可能安装在测试系统中的其他仪器实现阻抗匹配。

除了带宽和特性阻抗之外，其他RF开关规格也可能直接影响信号完整性，如插入损耗、电压驻波比(VSWR)、隔离度、串扰和RF功率。插入损耗衡量的是信号通过开关后的功率损耗和信号衰减。VSWR是反射波与透射波之比，具体来说是“驻波”模式中的电压峰值(当反射波同相时)与电压谷值(当反射波异相时)之比。隔离度是在开路电路上耦合的信号幅度，串扰是在电路(比如独立的多路复用器组)之间耦合的信号幅度。

RF开关一个有趣的现象是所有这些规格会随信号频率而变化。因此，在挑选RF继电器或开关时，应根据信号的特定频率比较这些规格。否则，RF开关很可能无法发挥应有的性能。

有关RF开关选型的更多信息，请阅读《了解主要RF开关规格》白皮书。

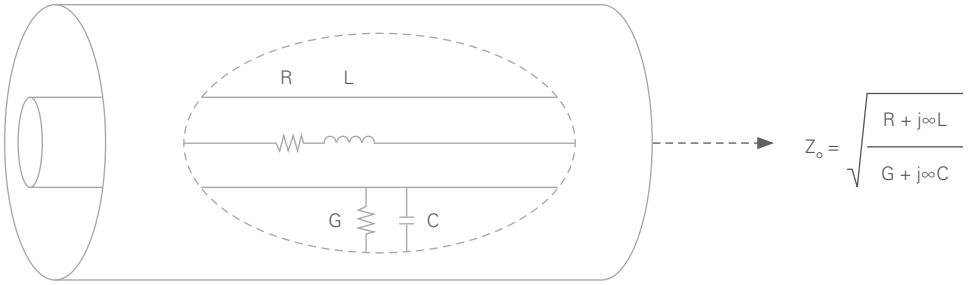


图 24 传输线的特性阻抗

开关使用技巧和窍门

在规划自动化测试系统的开关部分时，可以借助一些常规技巧来构建一个可保持信号完整性的高效开关系统。

总测试点数与同时连接数

使用矩阵时，不仅要考虑可能的最大连接数量，还要考虑同时连接的最大数量。如果只关注可能连接的总数，则通常会导致每一行专用于每个仪器的每个I/O引脚。此时可能需要构建不必要的大型矩阵。例如，如果有22个仪器引脚和106个DUT测试点，则需要创建一个22 x 106矩阵(2,332个继电器)，其中22个I/O引脚连接到行，106个DUT测试点连接到列。

但是，如果在任一给定时间最多只需要连接四个仪器引脚，则22 x 106矩阵就显得过大，会造成不必要的浪费。可以改为将仪器放置在22个额外的列上，并使用行在列之间进行路由，此方法可将矩阵尺寸减小至4 x 128(512个继电器)，大约为原尺寸的20%。这不但可以节省空间和成本，而且不会影响测试时间或质量。

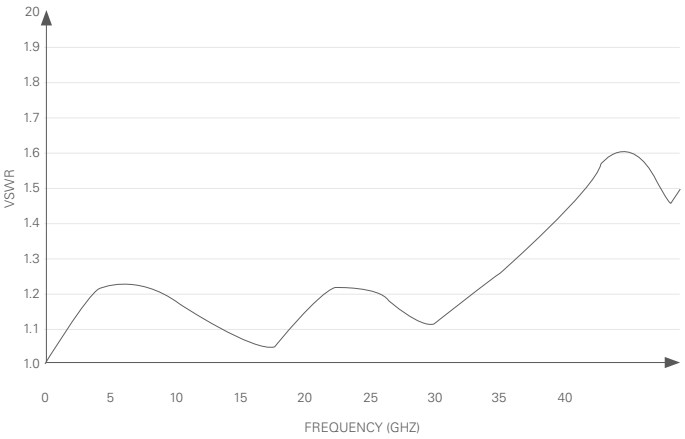
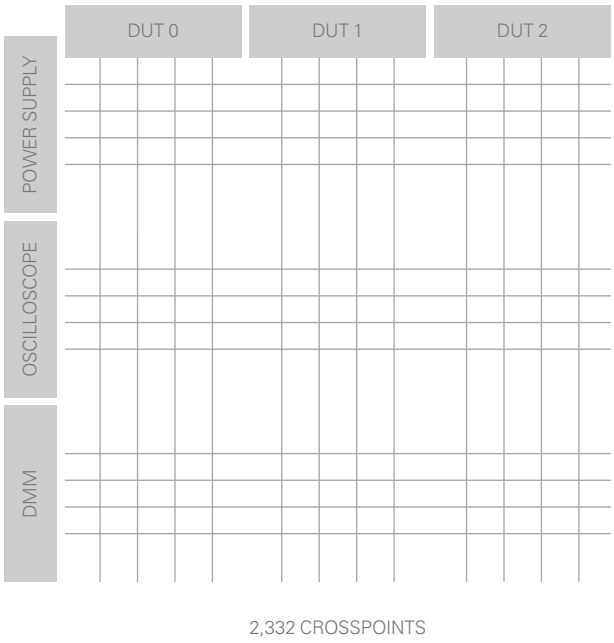


图 25 方波信号具有远高于基频的谐波频率



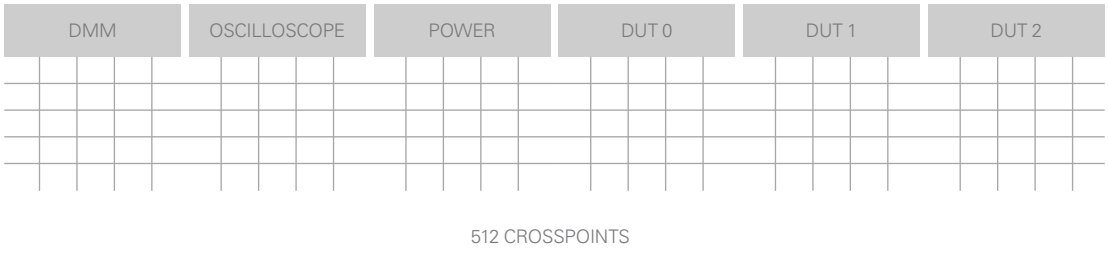


图 26 如果将仪器放置在列上并使用行在列之间进行路由，则可以节省矩阵空间，但需要按顺序执行测试；如果依旧将仪器放置在行上，则可以满足更快速的并行测试需求

N线开关

许多矩阵或多路复用器开关模块可以在给定拓扑内切换两个或四个信号路径，这一点与标准的1线开关模式有所不同。而在执行测量时，可以使用1线开关将各种信号路由到可能以单个信号或地面为参考的仪器。

用户有时需要同时切换多个信号。2线或差分开关可使用一个命令控制两条信号路径。这种方法可以方便地切换差分信号，同时具有出色的共模噪声抑制性能。4线开关通常用于4线电阻测量，其中两条引线用于激励，另外两条引线用于测量DUT两端的压降。

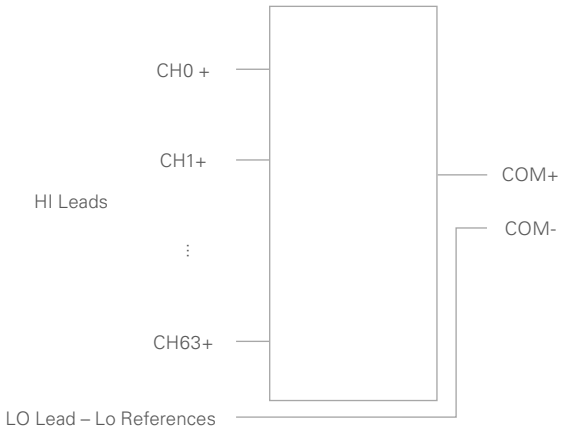


图27 单端多路复用器非常适合以共享信号或地面为参考的测量

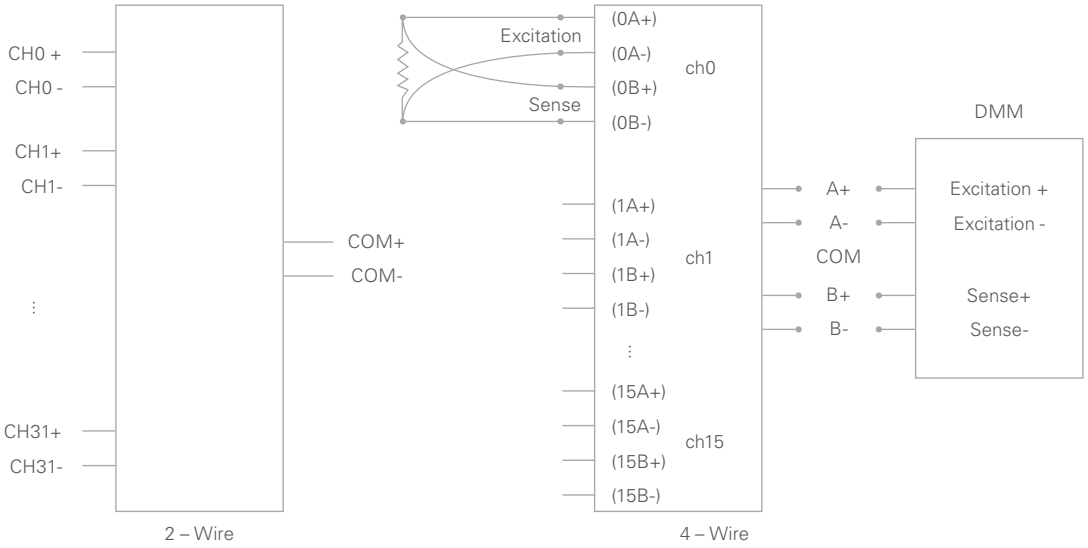


图28 使用2线或4线开关同时切换多条信号路径

开关功率

很多时候，测试需求规划中会涵盖最大电压和电流值，但却忽略了瞬时功率。假设某个开关或继电器的额定电压和额定电流分别为100 V和2 A，这并不一定意味着它可以处理200 W的功率。许多开关的最大额定功率与额定电压和额定电流并无明确关联。例如，常见簧片继电器的额定电压和额定电流可能分别为100 V和500 mA，但其最大额定功率可能只有10 W。因此，挑选开关时还应考虑最大瞬时功率。

将高电平信号与通用信号或低电平信号隔离

用于大功率或高频信号的开关在通道密度方面通常低于用于通用信号的开关。因此，应将大功率或高频信号与主开关系统隔离，以维持主开关系统的通道密度。如果为了处理高电平信号而尝试构建一个适用于所有信号的开关，则可能需要很大的开关尺寸和高昂的成本。

基于信号频率比较RF规格

比较RF开关时，应根据信号频率评估规格。许多RF规格，例如隔离度、VSWR、插入损耗和RF载波功率都会随信号频率而变化。为了进行准确的比较，需要查看详细的开关规格，找到所需频率对应的规格。此外，一些开关供应商会针对每种类别的开关分别发布保证的规格和典型的规格，而其他开关供应商仅会发布典型的规格，其参数明显优于保证的规格。

考虑使用硬件触发开关实现开关速度最大化

在许多自动化测试场景中，时间就是金钱。许多开关使用软件命令进行单独控制，此时总线延迟会降低每次开关操作的速度，并且会产生额外的软件开销。某些开关提供硬件定时和触发，这样可以将开关连接列表加载到开关内存上，并使用硬件触发器依次触发列表中的连接。每次开关操作完成后，开关会向仪器发送触发命令，启动下一次测量。

这种操作称为开关握手，可以避免与传统软件触发开关相关的软件开销和总线延迟。开关握手对于高速型继电器(例如FET或SSR)尤其重要，因为软件开销和总线延迟会大幅降低每次开关操作的速度。使用开关握手时，簧片继电器的总开关时间可缩短10倍，而FET开关的总开关时间可缩短100倍甚至更多。继电器速度越快，开关握手可以提高的吞吐量就越多。

附加信息

NI开关产品

无论是在十几个测试点上执行高精度低速测量，还是对集成电路进行高通道高频特性分析，NI基于PXI的灵活模块化开关解决方案都能满足您的需求，帮助您最大限度地重复利用设备、提高测试吞吐量和系统可扩展性。

进一步了解NI PXI开关产品

Switch Executive

Switch Executive是一款智能开关管理与路由应用程序，能够加速开发过程、简化对复杂开关系统的维护。该应用程序提供只需轻点鼠标的图形化配置、自动路由功能以及直观的通道别名，可帮助用户轻松设计开关系统并进行文档记录。

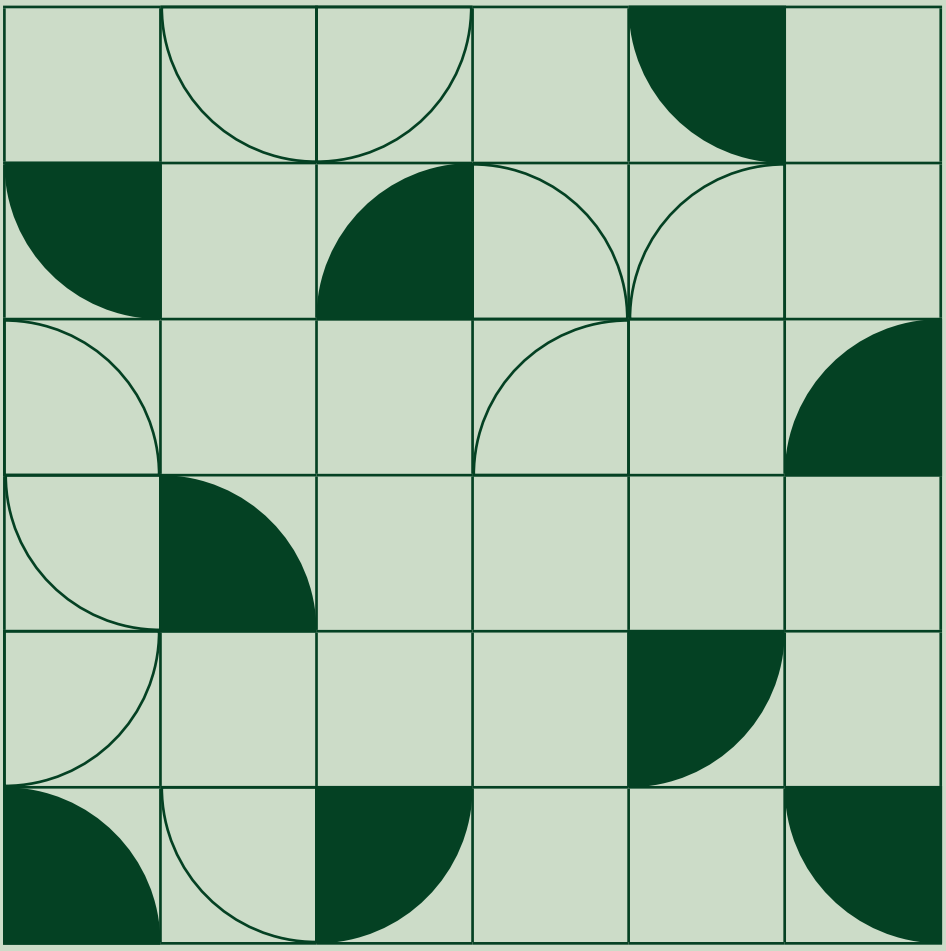
进一步了解Switch Executive

NI Switch Health Center

为了简化高通道数系统的继电器维护并提高系统可靠性，NI Switch Health Center会将信号沿开关的每个路由发送出去，以验证每个继电器的状态。它会提醒用户继电器是否发生故障或卡在断开/闭合状态，并报告电阻的变化，以便用户确定继电器是否即将“寿终正寝”。

进一步了解NI Switch Health Center

测试执行软件



简介

大多数测试系统都是围绕着“效率”和“成本”这两个概念进行设计。无论是在消费电子产业还是半导体生产领域，测试工程师都非常关心测试系统的独立测试时间和总吞吐量，以及这些参数如何影响资源。当应用程序不断扩展到包含多种测试、各种仪器和多个待测设备(UUT)时，便不可避免地需要监督测试执行软件，以解决成本和效率问题。

测试执行软件通常由客户在内部自行开发，或者作为商用现成(COTS)产品购买。在典型的构建还是购买的争论中，测试架构师必须确定编写自定义测试执行程序更合适，还是购买现成解决方案加以集成更具成本效益。在决定自主开发还是购买测试执行软件之前，测试架构师必须了解此类软件的目的和核心功能。本指南总结了测试执行软件的主要功能，并介绍了这一技术的实际应用场景。

背景

测试执行软件可实现大型测试系统的自动化和精简化。该解决方案位于软件堆栈的最上层，整合了各测试级别的通用功能，如测试执行、结果收集和报告生成。测试执行软件的功能特性并非专属于特定UUT，因此各种应用均可将其作为框架使用。这意味着开发人员在使用LabVIEW图形化语言、C、.NET或其他语言编写测试代码的时候可以专注于测试具体的设备，而所有UUT的常用功能都由最上层测试执行软件维护。总而言之，测试执行软件从开发、成本和维护的角度，有效地定义了这些通用功能。

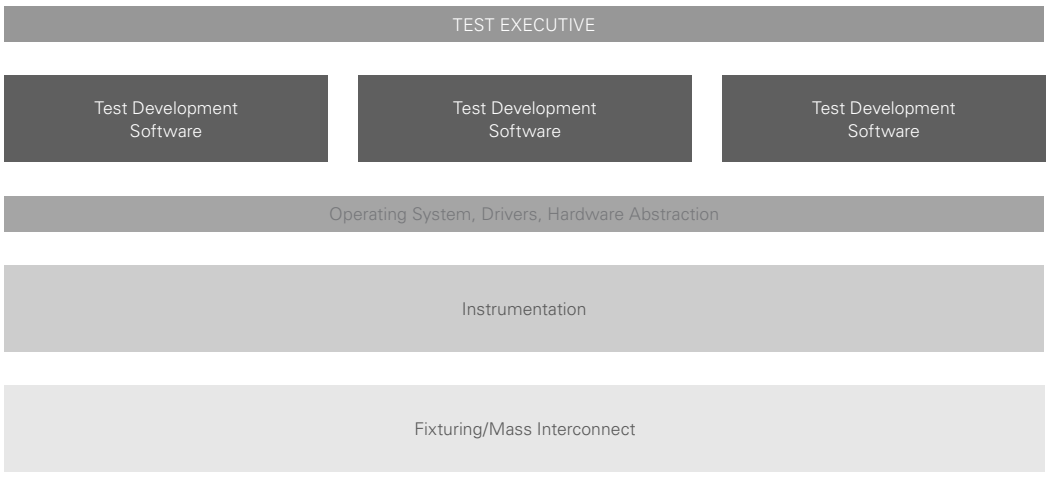


图1 通过在更高抽象级别上完成所有的通用测试任务，测试执行软件将个体测试开发与整体测试系统架构需求分离

测试执行软件的特性

根据公司大小、测试仪具体规模以及被测设备的种类，执行测试软件分为简单版到高级版等不同复杂度。本指南概述了该软件可能包含的常见特性。某些特性对于测试执行软件的所有实现都至关重要，另一些则并非绝对必要。每种特性都预估了完成开发的用时。这些预估数据基于数百个自动化测试客户的经验，具体可见《测试执行软件-构建还是购买？基于NI TestStand的财务分析比较》。

测试序列开发环境

测试执行软件提供用于构建测试序列的开发环境。该特性可为整个执行过程提供开发接口，既是不可或缺的基础，同时也十分复杂。序列架构包括运用分支或循环逻辑的能力、导入测试限制的方法以及独立测试代码的规范和整理。与测试代码的交互需要具有使用各种编译格式的灵活性，例如DLL、VI和脚本，以及与不同开发环境集成的能力。测试执行软件也可以使用源代码控制程序提供的测试代码。

使用自定义测试执行软件构建测试序列开发环境需要约100个人日，而购买商用解决方案可立即获得现成的环境。自行开发解决方案将耗费大量开发时间，因为开发环境涉及的功能范围十分广泛。然而，序列开发环境是序列架构体验的基础，不可忽视。

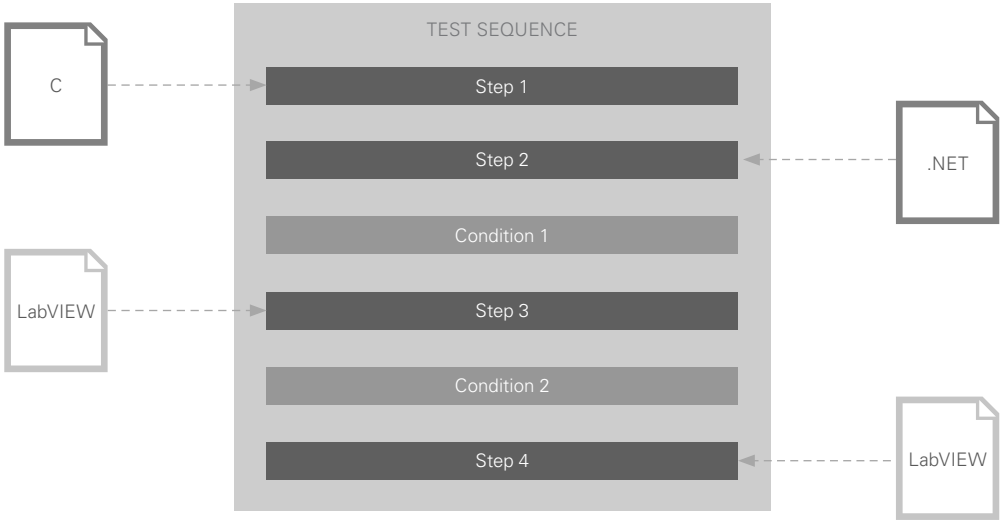


图2 高效的序列开发环境可帮助测试工程师开发和调试复杂的序列，并调用到现有测试代码中

自定义操作界面

操作界面是操作员用来与测试系统进行交互的显示器。操作界面通常允许操作员选择关键输入参数，例如UUT标识符、待执行的测试序列或报告路径。其中还包含一个“运行”或“开始”按钮，用以控制执行过程。如今，许多大型测试系统都需要专业GUI，该GUI因应用或公司而异，由开发人员所选的编程语言编写而成。除了自定义功能，高度功能性的界面还具有加载、显示和运行测试序列的能力，同时配有交互式用户提示、执行进度指示器、测试数据可视化和本地化功能。

开发自定义操作界面可能需要8到32个人日。COTS解决方案提供现成的库和UI控件，可以缩短该时间。无论是自行构建还是购买测试执行解决方案，自定义操作界面的开发都需要投入相当一部分时间。认为操作界面对其系统无足轻重的测试工程师可能会指示操作员直接在开发环境中工作。

序列执行引擎

测试执行软件的核心配置是序列执行引擎。序列执行引擎负责UUT评估所需的所有操作，包括调用独立测试代码、创建测试间的执行流程以及管理测试间的数据。无论是在开发环境中、通过自定义操作界面，还是在已部署的测试仪上，序列执行引擎均可执行给定的测试序列。

自行开发一款序列执行引擎需要至少15个人日的工作量。但它对于所有测试执行软件来说不可或缺。

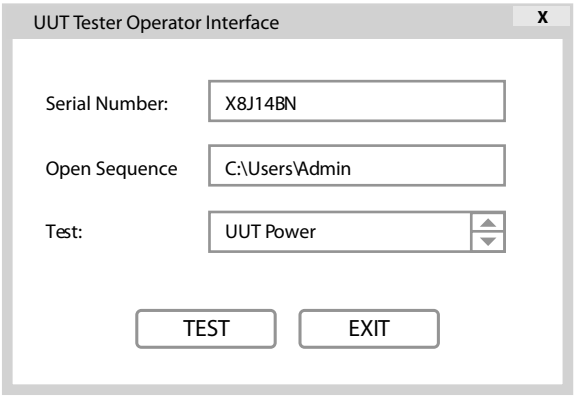


图3 客户操作界面能够识别给定测试序列的唯一UUT、公司、应用程序、测试和操作员角色

结果报告

测试执行软件扮演着抽象的角色，负责整合个体测试数据、将数据临时储存至内存，以及发布综合测试结果。报告采取多种格式，包括XML、text、HTML和ATML格式。数据也可以在执行后存储到数据库。通过可扩展的报告选项，测试执行软件提供多样化的报告格式。结果报告对于许多测试系统来说必不可少。

从零开发结果收集功能和报告生成器大约需要15个人日，具体取决于所需的报告。COTS解决方案内置了报告生成器，可以在1个人日或更短时间内通过自定义结果报告满足应用需求。

用户管理

将不同角色在测试执行软件级别的责任区分开来很有必要。用户管理工具有效划分了总测试架构师、编写和调试测试代码的个体测试开发人员以及运行测试的操作员或生产经理之间的责任。某些面向特定用户的功能甚至可能受到密码保护，以防测试序列出现误用。

使用自定义测试执行软件开发用户管理系统大约需要五个人日。尽管并非测试执行软件的必需工具，但用户管理工具无需开发人员投入过多精力即可实现，而且可以简化测试执行软件职责的执行。

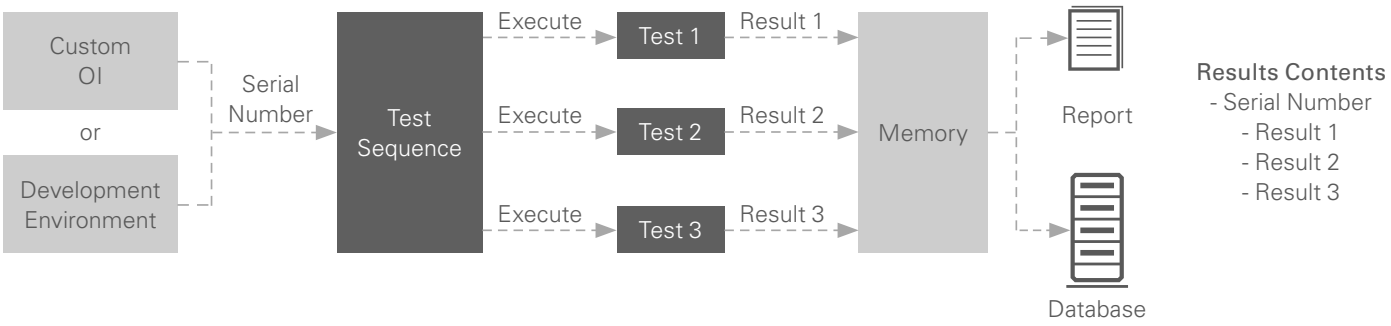


图 4 测试执行软件在测试系统中的部分职责是整合执行结果并将其发布到报告或数据库中

权限	架构师	开发人员	操作员
编辑	√	-	-
保存	√	-	-
部署	√	√	-
循环	√	√	-
运行	√	√	√
退出	√	√	√

用户	级别
Mark	操作员
Larry	操作员
Julie	开发人员
Scott	开发人员
Lauren	架构师

表 1 与Windows文件权限类似，用户管理器可将不同角色在测试执行软件级别的责任区分开来

并行测试功能

并行测试涉及同时测试多个设备，与此同时仍保持代码模块性能、结果收集和UUT跟踪功能正常运行。并行测试方法的范围涵盖流水线执行（测试顺序保持不变，但测试执行软件可以同时跨多个测试座测试）、动态优化、批处理以及其他复杂的执行方式。

对于测试执行软件开发人员来说，开发并行测试功能通常极为耗时，从头开始可能需要100个人日。尽管并行测试需要较长的开发时间，但在大型测试系统中，通过扩展执行以减轻吞吐量需求的能力至关重要。许多公司在首次部署测试执行软件时并未考虑并行测试功能，后来才意识到此功能不可或缺，但为时已晚。

单元/设备跟踪和序列号扫描

同时测试多个UUT设备时，有必要对每个被测设备进行唯一的标识和跟踪。这些信息可以与测试结果一起存储，以进行特定单元或批次分析，或者用于在出现问题时查明错误来源。设备跟踪的方式包括由操作员在键盘上手动输入，以及使用全自动扫描仪读取条形码以加载UUT信息等。

完全自主开发此功能大约需要5个人日，使用COTS解决方案进行定制开发则需要大约1个人日。并非所有测试系统都需要UUT跟踪。但此功能对于半导体或消费电子等需要大容量、高吞吐量测试的行业来说非常有用。

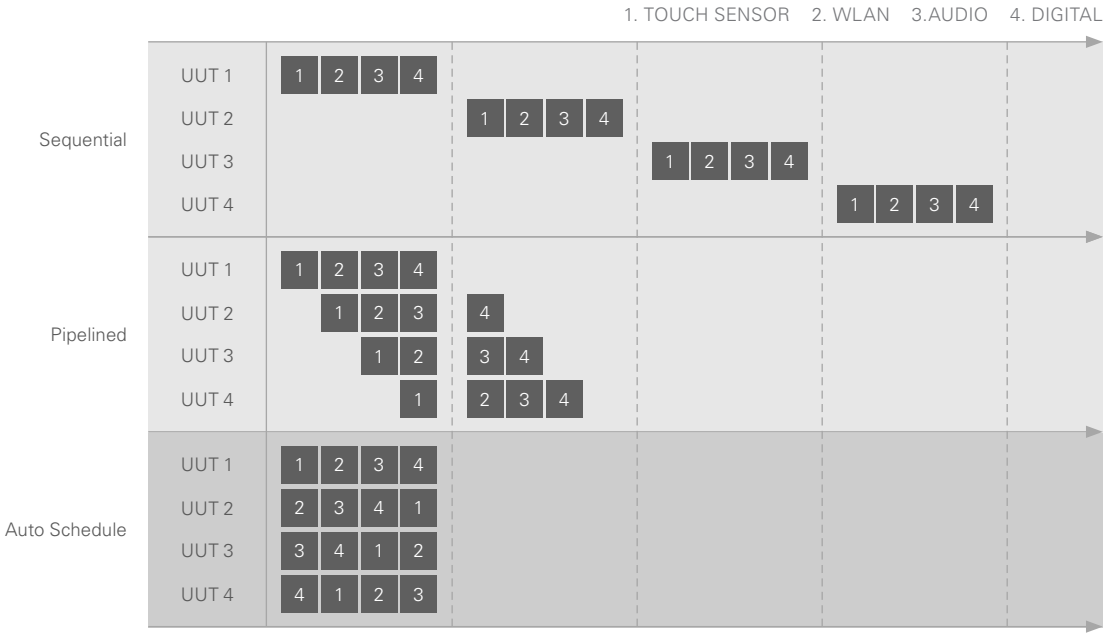


图 5 并行测试功能无需重建测试执行软件架构即可提升系统吞吐量

测试部署工具

大多数大型测试系统并不会采用孤立架构，而是多个测试站点或整个生产车间的解决方案。测试执行软件在系统部署中发挥着关键作用，它提供一种机制或实用程序，可将整个软件堆栈打包到内置的可分发单元中。测试系统可通过多种方式分发，架构师可能更希望部署测试系统的镜像或者部署包含所有必要依赖项和运行时间的全功能安装程序。有关此主题的更多信息，请参阅《测试系统构建基础知识》系列技术白皮书的“软件部署”部分。

部署是一项艰巨的任务，如果从头做起，开发团队可能需要多达20人日的工作量。即便使用商用测试执行软件提供的现成部署实用程序，可能仍然需要三个人日才能成功部署。考虑到此工具适用于多个测试站点，通常有必要在测试执行解决方案中配备此工具。

维护

就像大型测试系统中的任何其他组件一样，测试执行软件必须得到适当的维护，以确保经久耐用。维护内容包括进行扩展以涵盖新测试、维持软件或操作系统升级的兼容性，以及修复检测到的所有错误。测试执行解决方案的维护甚至延伸到文档领域，文档是操作员、开发人员和架构师在与测试执行人员合作时所赖的重要资源。

尽管无法得知具体测试仪的需求，但在每年开发自定义测试执行软件的初期，有15%的时间需要用于维护。总开发时间包括生成充足文档所需的时间（预计为20天）。测试执行软件支持的时间间隔因具体情况而异，有可能大幅增加上述开发的成本估算值。尽管如此，任何测试执行解决方案都不应轻视维护工作。

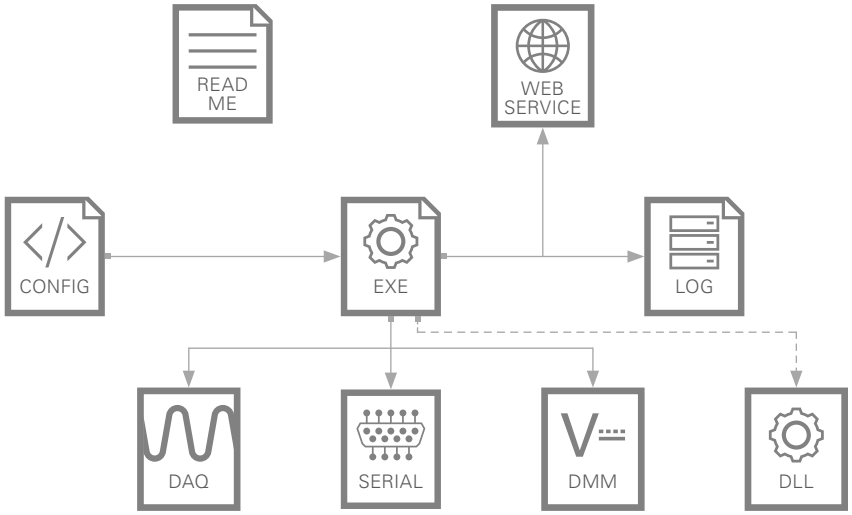


图 6 部署涉及使用部署工具或编译服务器打包测试系统的所有必要组件，然后将打包后的内容分发到目标测试站

实际场景1

Jonathan在一家小型公司的设计实验室担任测试工程师，该公司主要生产低成本消费电子产品。每个设备由一个远程控制器控制，Jonathan负责编写测试代码，在设备投入生产之前检验远程控制器和设备之间的发射器-接收器通信情况。随着公司近来规模不断扩大，Jonathan对于每台设备的测试时间越来越有限。他希望让系统自动执行现有验证代码，以便留出更多时间为新设备编写代码。因此，他决定使用测试执行软件来执行测试代码序列。

Jonathan对测试执行软件的需求如下表所示。

功能特点	实现方式
测试序列开发环境	Jonathan需要一个开发环境来构建序列。他一直以来使用的测试代码已经相当模块化，所以他只需要在这个环境中调用和循环测试代码。
自定义操作界面	Jonathan希望能够以最少的交互执行一组测试代码。他希望只需要通过指定几个相关参数就能够识别设备、所需测试和报告路径。但是，鉴于他是最终操作员，是否拥有一个与开发环境分离的接口并不重要。
序列执行引擎	这是Jonathan对测试执行解决方案的核心需求。每个测试由几个必须按序执行的独立LabVIEW VI组成。
结果报告	现有测试代码会提示用户为给定原型生成新的技术数据管理流（TDMS）文件。随后的每个VI都将最少的测试结果存入该文件。测试执行软件只需自动创建该TDMS文件，然后执行其余的测试代码，即可按约定生成结果。
用户管理	用户管理并非优先事项，因为该测试执行软件由Jonathan本人设计、开发和操作。
并行测试功能	Jonathan每次只测试一个设备，因此UUT的数量不是问题。
单元/设备跟踪和序列号扫描	测试是在设计原型上完成的，因此Jonathan无需跟踪分配的序列号，而是通过操作员在运行时输入的唯一名称跟踪每个UUT。
测试部署工具	Jonathan不打算将此代码部署到其他测试仪。他的测试台为设计实验室所独有，与制造设施互相独立。
维护	这个项目完全属于Jonathan。他将使用和维护所有选中的测试执行软件。他不打算通过文档记录自己的工作，因为他将是唯一一名操作和使用此测试执行软件的用户。

表2 Jonathan对测试执行软件的需求主要集中于现有验证代码的简单自动化

Jonathan决定自主开发一款测试执行软件。他没有复杂的序列执行或报告需求，也不打算将此系统部署到其他用户或测试站。如果购买了商用测试执行软件，他可能之后无法取得投资回报，因为其中的大部分功能都派不上用场。相比之下，他只需依靠自己的软件知识和先前所购买的应用软件，即可在短短10个人日内开发出满足需求的序列执行器。

Jonathan在LabVIEW软件中开发了他的测试执行软件。他的设计拥有简单界面，让操作员能够调用一组预定的测试步骤并选择TDMS日志路径。在为新的原型引入额外测试时，Jonathan偶尔会对序列执行器做一些小改动。总的来说，得益于这款序列执行器，设计实验室的生产力得到了显著提升。

在这种情况下，自行开发测试执行软件是满足Jonathan标准的最佳选择。通常来讲，自主开发测试应用软件是需要序列生成或完全自动化时所采取的第一步，相比于生产环境的需求，此方法可能更适合测试台应用。

功能特点	实现方式
测试序列开发环境	高效的开发环境必不可少，这个环境必须支持测试执行软件的主要功能特性，且必须支持LabVIEW、.NET和Python代码的序列执行。
自定义操作界面	Dave的最终需求是一个为公司量身定制的操作界面。他还希望删除“运行”按钮之外的多数功能。
序列执行引擎	显而易见，此功能是为了满足该系统的吞吐量需求。
结果报告	目前，每次测试都会将数据单独记录到SQL数据库中。因此，测试执行软件需要整合收集的结果、将结果汇总传送到数据库并使用序列号进行标识。
用户管理	大多数与测试仪的交互由生产级别的操作员进行。Dave更希望用户管理工具或可自定义界面能够从操作员的界面中删除开发权限。
并行测试功能	只要测试仪吞吐量与生产吞吐量相匹配，Dave就不需要一次性测试多个UUT。
单元/设备跟踪和序列号扫描	序列号用于标识每个组件和组装的UUT。条形码扫描仪用于跟踪此类信息。测试执行软件必须能够在其执行的不同测试之间传输此类信息。
测试部署工具	Dave需要将最终产品部署到另外10个测试仪中。
维护	测试工程部门将维护测试执行软件，可能是在内部解决方案维护全部功能，也可能在COTS软件中按需维护。

表3 | Dave对测试执行软件的评估主要基于功能测试仪的潜在吞吐量需求

如果...

- 几个月后，设计实验室聘请了新的测试工程师来协助测试。该工程师该如何学习序列执行工具的操作方法，或有效解决任何出现的错误或漏洞？
- Jonathan调到另一个部门，或者离开公司。更新和维护序列执行工具所需的知识库应当如何维护？
- 序列执行工具有必要对整个原型进行功能评估。如何将不同工程师用其他语言编写的附加测试代码、不同的编程范式和报告技术整合到一起？
- 测试执行软件被移植到生产环境中，以确保测试的一致性。这些软件能否因地制宜，满足此类需求？

实际场景2

Dave的公司正在设计一款新的功能测试仪，将在生产线末端使用。目前，运行UUT测试的方式是手动执行一系列已有的零散代码片段。这一过程极大限制了生产线的吞吐量，Dave希望使用测试执行软件将其自动化。该公司没有针对测试执行软件的统一标准，每个小组通常会从少部分商用解决方案和众多自定义解决方案中选择所需软件。

Dave对测试仪的基本要求如表3所示。

为了做出决定，Dave还从财务角度对测试仪进行了权衡。他估计新的测试仪将是一个大型的高性能PXI机箱和嵌入式控制器的组合。考虑到评估UUT所需测试的性质，机箱将包含从数据采集卡和模块化仪器（如数字化仪和任意波形发生器）到射频测试设备的多个模块。无论采用何种测试执行软件，每台测试仪的成本都将在10万美元左右。

在评估软件堆栈时，Dave注意到购买COTS解决方案会增加项目成本。购买测试执行软件的开发许可证需要几千美元，并且每增加一台测试仪，都要另外花费500美元获得测试仪的部署许可证。

Dave相信他可以通过使用Python构建自定义解决方案来节省测试执行软件的成本。该语言是开源的，并且开发环境可免费使用，他认为这两项优势足以抵消在内部开发测试执行软件所需的额外开发时间。

测试工程团队精通Python，最终在规定的时间范围内提供了顺序序列执行引擎、数据库连接和现有测试代码复用等核心功能。测试执行软件现已成功部署到生产线。测试工程师偶尔会被要求修复一台或多台测试仪的错误。

如果...

- 生产线的生产需求增加，以至于现有的测试执行软件无法满足吞吐量需求，此时需要添加并行测试功能。
  - 尝试实现此功能需要多少额外的开发时间？这对自定义测试执行软件和COTS解决方案的成本对比有何影响？
  - 假设由于Python语言在多处理方面的局限性，测试仪吞吐量无法满足需求。Dave的团队需要购买额外的硬件，以便再次利用目前已有的测试执行软件，或是选择另一种全新的测试执行软件。这对自定义测试执行软件和COTS解决方案的成本对比有何进一步影响？
- 因为还有其他优先任务，测试工程团队无法随时为测试执行软件提供服务或升级。
  - 当出现此类需求而团队无法及时提供帮助时，生产会受到怎样的影响？此类停机对系统维护成本有何影响？
  - 测试工程团队花费在维护测试仪上的时间是如何量化的？此因素对系统维护成本有何影响？

实际场景3

Karen在一家设计和生产小型医疗设备的公司工作。该公司为每种产品设立了单独的全自动生产线。虽然每个小组都采用测试执行软件进行最上层的系统管理，但该公司尚未实现测试执行软件的标准化。最近，该公司的新任测试经理上任，有意对测试执行软件进行标准化。Karen的任务是选择商用解决方案、现有内部产品或进行新的开发来实现测试执行软件的标准化。

Karen为负责每种产品的各个小组编写了以下要求列表。

功能特点	实现方式
测试序列开发环境	测试开发人员需要一个灵活的开发环境，特别是可以与他们的LabVIEW和VB.NET代码交互。Tortoise SVN用于源代码控制，开发环境需要与此工具集成。
自定义操作界面	测试经理希望根据正在开发或测试的产品自定义操作界面。操作员表示他们希望在监控测试仪时能有一个进度指示器实时更新测试状态。
序列执行引擎	这是所有测试仪必需的功能。
结果报告	所有生产系统都必须符合公司的HTML报告标准。
用户管理	测试工程团队由少数系统架构师和大量测试开发人员组成。测试经理想要将这两个角色的职责区分开来。
并行测试功能	在对组装好的单元进行功能测试时，生产线一次评估一个UUT。然而，板卡级测试应进行优化，以提升执行速度。为了满足所有测试仪的需求，需要添加并行测试功能。
单元/设备跟踪和序列号扫描	公司每个产品和板卡的UUT信息通过操作员输入进行跟踪。
测试部署工具	该公司拥有专门编写测试代码的测试工程师团队。该团队必须能够将所在实验室的开发环境部署到所需的生产环境。目前，部署过程需手动完成。
维护	作为标准化工作的一部分，测试经理需要一个正式的维护计划。该计划需要考虑到公司在今年晚些时候当前系统生命周期结束时进行的操作系统迁移。

表4 | Karen对测试执行解决方案的兴趣源于对不同测试仪进行标准化的需求

根据此标准，Karen排除了所有现有的内部解决方案。这些解决方案大多只能用于单个测试仪，其架构缺少一致性，无法扩展到其他生产线，特别是在序列执行需求、操作界面定制和有效部署实践方面。此外，过往事实证明，当软件出现问题或对设备进行修改时，很难追踪到负责该测试执行软件的测试工程师。

因此，Karen向她的经理提议购买商用解决方案。该测试执行软件由知名供应商制造，同一供应商生产的其他硬件和软件工具已广泛应用于测试仪中。该测试管理软件支持“即开即用”，可以满足测试仪的序列执行需求，并采用所要求的特定报告格式。该测试执行软件还包括一组可满足其他测试仪需求的工具，包括用户管理工具和部署实用程序。由于商业供应商提供维护服务，Karen的经理不必担心日后操作系统迁移时会产生不兼容的问题。

Karen的公司最终做出了成功的决策。总体而言，测试执行软件提供了一个适用于不同生产线的灵活框架。购买测试执行软件的统一标准为公司带来了额外的好处。供应商提供培训服务，可帮助测试工程师熟悉新软件。测试执行软件的购买合同还包括维护服务，由供应商同意提供常规补丁和升级。公司还可以获得技术支持资源，有助于解决测试序列的问题。

这款商用解决方案至今仍然是Karen公司的标准。当测试工程师因升职、退休或自然减员等原因而需要换人时，测试经理可以聘请具有测试执行软件操作经验的人员。该公司成功地从一个过时的操作系统迁移到两个完整版本，同时仍保留所选的测试执行软件。随着新产品的开发，可扩展架构可以持续满足生产需求。

结论

无论公司规模、行业或个人测试标准如何，都有必要部署测试执行软件来实现最上层的系统管理。这意味着将引入一定程度的抽象，将系统的常用功能与测试代码的特定功能分离。在构建最终解决方案前，必须对测试执行软件的需求进行完整的评估。许多测试工程师都在自主开发和购买测试执行软件犹豫不定，需要从成本、功能和维护的角度仔细考量每个解决方案的优势，进行取舍。

附加信息

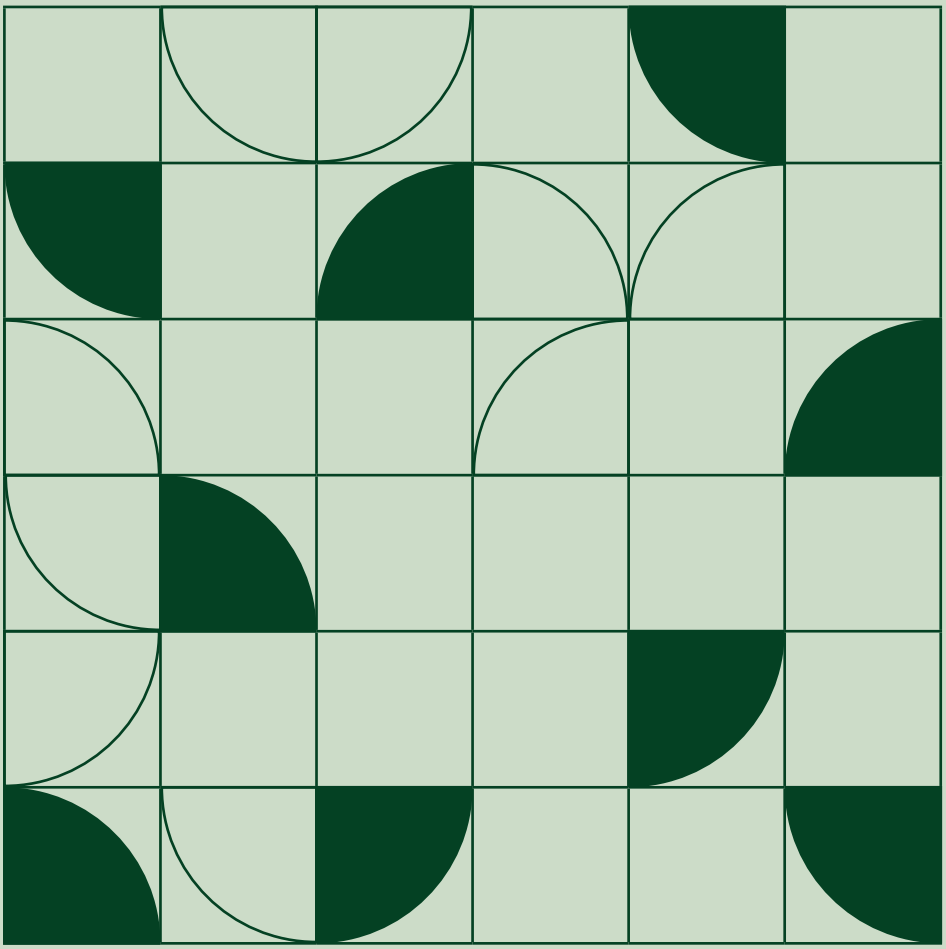
TestStand是一款行业标准的测试管理软件，可帮助测试和验证工程师更快地构建和部署自动化测试系统。TestStand包含可立即运行的测试序列引擎，支持多种测试代码语言、灵活的结果报告和并行/多线程测试。

TestStand不仅包含了许多开箱即用的功能，而且也具有高度可扩展性。因此，全球数以万计的用户选择TestStand来构建和部署自定义自动化测试系统。NI提供培训和认证计划，每年培训和认证超过1000名的TestStand用户。

了解更多关于TestStand的信息

硬件和测量抽象层

Grant Gothing，Bloomy Controls公司ATE研发经理



引言

从初始规划到软硬件开发再到最终集成，自动化测试设备(ATE)的设计和开发过程面临诸多挑战。在该过程的每个阶段，更改都变得日益困难，且实现成本日益增加。此外，由于在开发周期中软件开发通常在硬件开发之后进行，因此许多开放式项目都留给软件工程师去解决。良好的规划可以大大降低常见的风险，但无法预防所有问题，尤其是测试开发周期非常短的情况，最终集成阶段可能会出现许多问题。人们常常认为软件比硬件更具灵活性，因此往往会形成“用软件解决就好了”这一误解。但是，硬件和软件是密不可分的，大部分问题通常需要对两者同时进行更新。这不会随着初始部署而结束，而是会持续出现在系统的整个生命周期中。

随着产品变得越来越复杂，测试它们所需的系统也越来越复杂。由于ATE仪器成本越来越受到重视，将仪器复用于多个产品的能力就显得非常必要。而且，由于开发时间不断缩短，工程师需要并行开发软件和硬件，但开发需求通常并不明确。另外，测试系统部署之后，由于产品生命周期较长，意味着仪器故障或淘汰以及产品和测试需求的变更可能会给测试设备带来更多挑战。因此，模块化、灵活性和可扩展性对于成功开发自动化功能测试系统至关重要。

从硬件的角度来看，这通常通过模块化仪器和由可互换测试连接件进行的互连来实现。但是如何使测试软件像硬件一样具有适应性呢？完成这个任务的设计方法有多种，硬件抽象层(HAL)和测量抽象层(MAL)是其中最有效的两种。与在测试序列中采用设备特定的代码模块不同，抽象层可将测量类型和仪器特定的驱动程序从测试序列中分离出来。由于测试程序通常根据所使用的仪器类型(例如电源、数字万用表[DMM]、模拟输出和继电器)来定义，而不是根据特定仪器进行定义，因此采用抽象层会使得测试序列更快开发，更易于维护，更适应新的仪器和要求。使用硬件抽象来将软件和硬件分开，硬件和软件工程师便能够并行工作，从而缩短开发时间。开发用于序列和底层代码实现的通用API，可帮助系统架构师维护通用函数库，进而实现标准化和可复用性。这使得测试开发人员能够专注于单个待测单元(UUT)序列的开发，减少编写底层代码所需的时间。

ATE软件挑战

开发	维护
开发周期紧迫 需求不明确 测试程序不断更新 硬件设计完成之前就要开始软件开发 软件和硬件工程师分离	产品生命周期长 ■ 仪器故障或过时 ■ 仪器变更 产品更新 ■ 测试程序变更 ■ 需要采用新硬件 制造工程师通常不是最初的测试开发人员

软件抽象的优势

开发	维护
软硬件分离 序列开发与代码(驱动程序)开发分离 提供用于仪器的通用API 优化代码复用 缩短开发时间 将架构师与测试开发人员的角色分离	降低硬件淘汰或硬件变更的风险 ■ 降低对特定仪器的依赖性 ■ 无需修改测试序列即可变更硬件 降低代码复杂性，方便未来对测试进行支持/变更提高代码在不同平台上的兼容性

了解HAL和MAL之间的区别非常重要。HAL属于代码接口，支持应用软件在通用层而非特定设备层与仪器进行交互。通常，HAL定义了仪器类别或这些仪器须符合的类型、标准参数和功能。换句话说，从仪器的角度来看，HAL提供了与仪器通信的通用接口。MAL属于软件接口，提供了可以在一组抽象硬件上执行的高级操作。从UUT的角度来看，这些操作是使用多种仪器执行同一项任务的一种方式。这些共同组成了一个硬件抽象框架。

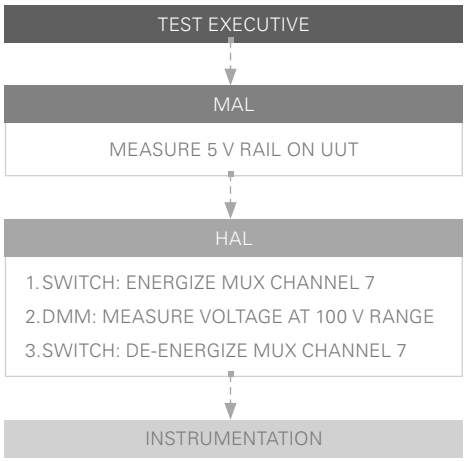


图1 抽象框架概览

打印机对话框就是一个典型的HAL/MAL日常用例。使用计算机打印时，不必打开终端，将原始串行、USB或TCP命令发送到打印机进行初始化和配置，然后再发送要打印的数据。硬件驱动程序实现用于执行配置和打印的方法。为了提升打印机的易用性，所有打印机制造商均遵循特定标准，将这些方法部署到驱动程序中。在同一硬件上执行多个任务时，所使用的通用接口就是HAL。那么，是否需要编写代码调用HAL的抽象方法来配置和打印文档呢？不需要，选择打印时，会显示打印对话框。此对话框提供了一个通用接口，用于调整配置参数，并将可打印数据发送到设备。这就是MAL，它让您能够直观地使用所有打印机，而无需了解打印机设备的底层功能。与打印文档一样，ATE HAL定义了所有仪器类型都必须遵循的通用底层任务集，而MAL则提供了执行高层操作从而使机器运行的一种通用方法。

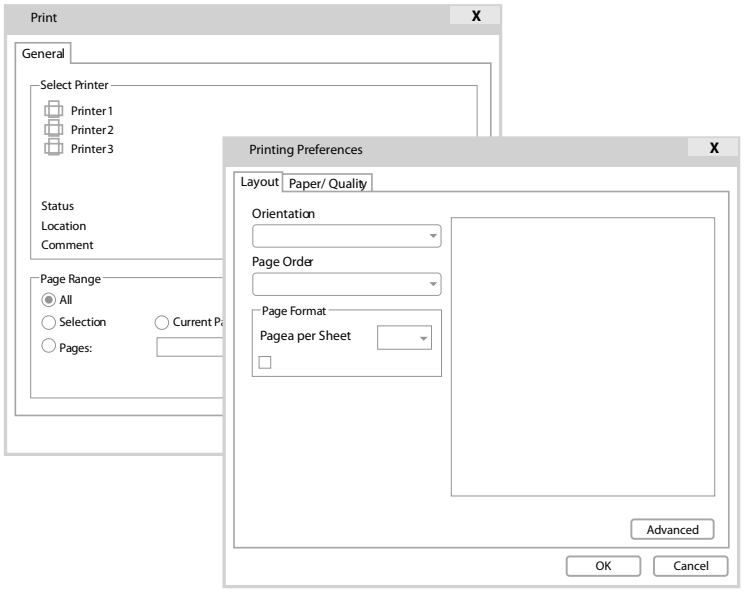


图2 打印机对话框就是一个典型的HAL/MAL日常用例

现有的HAL/MAL

测试和测量工程师采用了许多方法来解决HAL和MAL的问题。其中大部分方法可以直接使用，或者集成到更大型的自定义HAL/MAL方法中，从而以最小的投入实现功能扩展。以下是几个最常见的例子。

现成的软件抽象层

抽象	描述	类型	优点	缺点
供应商特定的驱动程序产品系列 驱动程序（NI-DAQmx、模块化仪器、Pickering PILPXI）	HAL	供应商特定的驱动程序产品系列为供应商的某些常见仪器组提供通用接口。这些驱动程序集可以连接每个特定系列的数十到数百种仪器。示例包括NI驱动程序（例如NI-DAQmx、NI-DCPower、NI-DMM、NI-Scope、NI-SWITCH和NI-FGEN）和Pickering PILPXI。	■ 为所支持的仪器提供通用的直观接口 ■ 有详细文档记录并经过测试 ■ 提供所有可用的功能 ■ 培训周期短-同一个驱动程序可控制该系列的所有仪器	■ 仅对每个供应商的特定驱动程序有效 ■ 并非所有仪器都支持所有功能
行业标准接口	HAL	IVI是仪器驱动程序软件的标准，可以增强仪器的互换性以及与符合IVI的仪器连接时的灵活性。该标准定义了13种仪器的规范，目前许多制造商都遵循这些规范，因此单个驱动程序可控制多种类型的仪器。仪器类型包括DMM、示波器、任意波形/函数发生器、DC电源、开关、功率计、频谱分析仪、RF信号发生器、计数器、数字化仪、下变频器、上变频器和交流电源。	■ 适用于从USB到PXI等各种仪器 ■ 兼容现有的许多GPIB，串行和LXI仪器 ■ 即插即用所有驱动程序都采用标准编程模式 ■ 上层仪器API支持仿真设备	■ 只指定API，而不实现-对于相同的测量，两个“可互换”的实现可能会返回不同的结果 ■ 无法用于不符合标准的仪器 ■ 可能无法实现所需的所有功能 ■ 可能会出现仪器无法支持的功能
Switch Executive	MAL	Switch Executive是一款开关管理与路径规划应用程序，用于将符合标准的开关矩阵和多路复用器仪器组合成一个虚拟开关设备。此虚拟开关可以使用命名的信号通道和路径直观地进行配置和驱动。	■ 直观的开关路径设置和操作 ■ 基于UUT或以测试为中心的名称定义通道和路径 ■ 定义无连接路径以提高安全性	■ 要求开关符合NI或IVI标准 ■ 不支持通过NI-DAQmx控制的继电器

现成的抽象能够帮助我们通过少量的自定义获得诸多功能。但是，这些抽象无法统一。IVI驱动程序和NI产品驱动程序对于符合标准的仪器而言是有效的HAL，但是从以仪器为中心的角度，它们仍然需要开发测试序列。从以测试为中心的角度来看，Switch Executive非常适合抽象开关路径，但它只能用于符合NI或IVI标准的开关连接(无模拟或数字I/O、DMM、示波器、电源等)。使用统一的HAL/MAL，可以更有效地开发以UUT为中心的序列，从而连接多种仪器，更好地处理仪器通道和连接的变更。

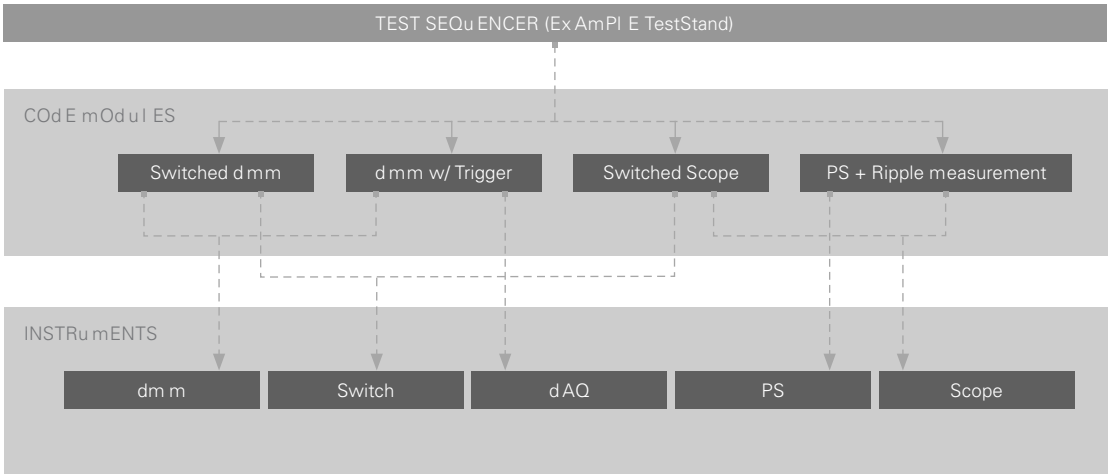


图3 不使用抽象的自动化测试软件的结构图

虽然HAL和MAL会带来诸多好处，但它们通常需要工程师基于过去的经验进行大量预测。他们需要考虑多个不同层次的抽象。有些抽象是软件和时间密集型，有些则是现成即用。一般来说，对特定仪器和测量进行抽象的程度越高，需要的框架规划和软件开发越高级。构建大型抽象框架非常耗时，并且如果没有适当规划，可能会有风险。如初始假设或实现不恰当，可能会持续产生积极和消极后果。务必要找到满足特定需求的硬件替代产品。如果不确定如何进行，请从简单处着手，保持可扩展性，并尽可能使用内置抽象。

背景

为了更好地理解HAL/MAL的实现方式，必须了解自动化测试软件的结构。从顶层看，自动化测试软件采用测试执行程序(或序列生成器)，例如TestStand。执行程序会调用一系列测试步骤，这些步骤通常属于代码模块或函数，使用LabVIEW软件中的图形化语言、C语言、.NET或ActiveX等并结合仪器特定的自定义方案进行开发；这些代码模块具有特定用途，例如使用DMM和开关的开关式DMM，或同时使用电源和示波器用于纹波测量的电源等。这样做会带来一定的好处，使每位开发人员都能够编写所需的特定功能，但同时需要大量的交叉功能，并且可能难以开发、部署和管理。而且，它要求每位测试开发人员都精通底层软件(如LabVIEW)。

不使用抽象

如果不使用硬件或测量抽象，就必须使用代码模块，直接引用驱动程序来与仪器连接。这导致测试序列与特定仪器和特定驱动程序代码密不可分。

不采用HAL/MAL框架时，通常会发生四个不可避免的问题：

- 由于仪器过时或需求变化，需要更换仪器-如果不采用抽象，则每次调用仪器时都需要更改驱动程序，在典型的测试序列中这可能涉及几十个步骤。每次更换仪器都会引起一些列软件更改。
- 由于出现新的需求，导致驱动程序的功能发生变化-如果驱动程序需要更新，则可能需要更新该驱动程序的每个例程以匹配新代码，尤其是输入或输出发生变更时。而且，直接调用驱动程序代码模块要求每位测试开发人员都要了解他们所使用的每个驱动程序的内部工作原理，特别是在使用多功能操作引擎的情况下。如要使用所有这些功能，测试工程师还必须精通软件。
- 测试序列是从仪器的角度进行开发的-通过使用仪器特定的驱动程序，所有测试序列都使用以仪器为中心的通道名称开发(例如使用以仪器为中心的名称开发测试序列)，而不是以UUT或测试为中心的名称(例如5v_Rail、LED_Control、VDD)进行开发。由于从UUT的角度开发测试程序，开发和调试变得非常困难。而且，对测试进行任何变更都需要对仪器、连线和互连系统有着非常深入的了解。
- 测试序列的开发通常与硬件开发同步进行-为了满足紧迫的时间期限，软件和硬件开发通常同步进行。因此，在开发测试序列时，并不总是能够掌握仪器和通道的详细信息。如果不采用抽象，就需要为驱动程序、通道编号和连接预留占位符。任何硬件信号发生变更，都需要更新测试序列。

例如，使用自定义方案，多路复用DMM测量代码模块可能类似于下图，即常见的开关式DMM LabVIEW VI。代码模块包含对特定仪器类型的特定调用集合。在本例中，这些集合是NI Mux和NI DMM。该代码模块基于输入通道和开关拓扑结构连接开关，基于一些输入参数使用DMM进行测量，然后断开开关。在测试执行程序中，必须了解要填写哪些字段，以及从仪器的角度来看需要哪些通道、拓扑和配置。此外，还必须确保将开关和DMM测量数据正确地传递到代码模块。

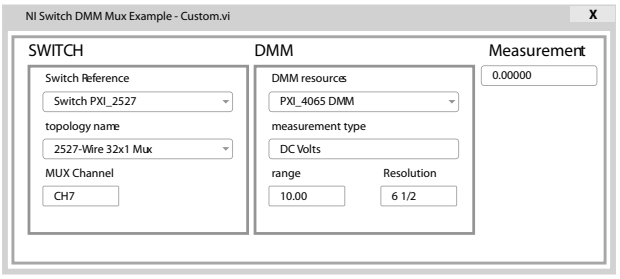


图 4 LabVIEW中典型多路复用DMM测量应用程序的前面板

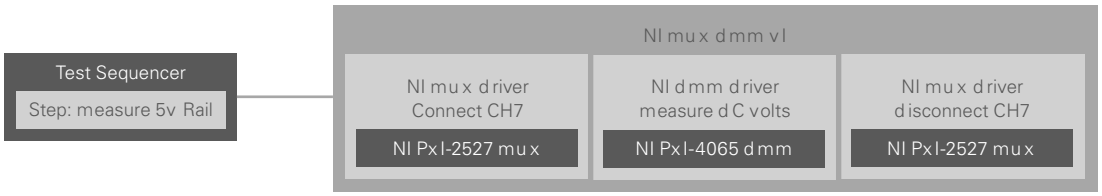


图 5 执行多路复用DMM测量的嵌套式命令调用

从测试执行程序的角度来看，可以调用代码模块来执行特定的函数(多路复用DMM)。该函数可对其适用的仪器进行特定调用。以下程序框图显示的是命令调用的嵌套。在图中，测试执行程序包含调用代码模块的步骤。代码模块使用驱动程序与特定仪器进行通信。每个外部项都依赖于其内部调用。

如果有仪器必须更换，依赖关系中的每个函数都必须更改。例如，如果初始多路复用器的通道不足，并且需要变换为通道数更高的矩阵，则由于存在依赖关系链，必须进行一系列更改：

- 仪器—PXI-2527多路复用器更改为PXI-2532B矩阵
- 驱动程序—NI Mux驱动程序更改为NI 矩阵(通道更改为行/列)
- 代码模块—NI Mux DMM VI必须更改为NI Matrix DMM VI
- 函数调用—必须更新测试执行程序对对代码模块的调用
- 序列—每次调用该代码模块时，都必须更新测试序列

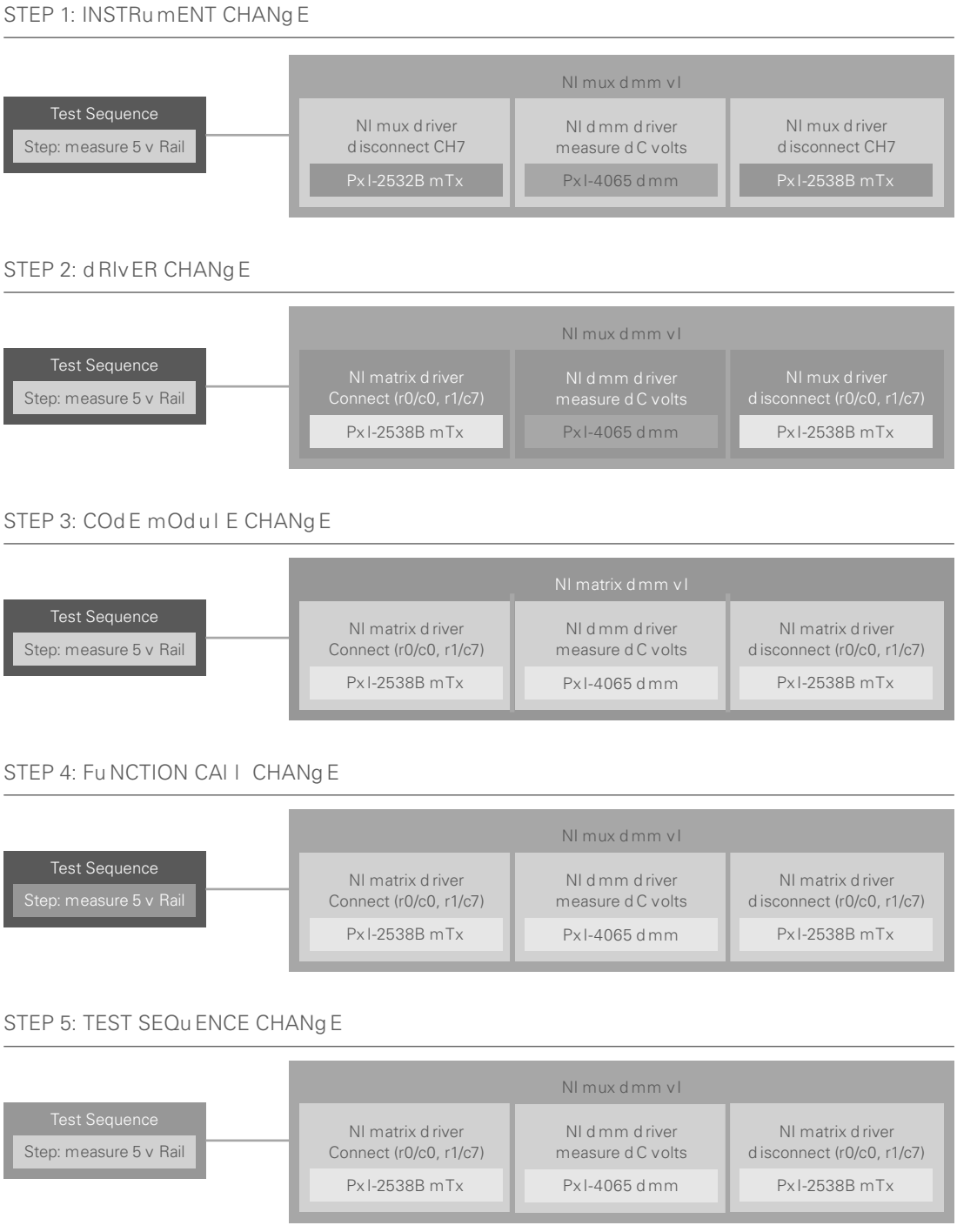


图 6 依赖关系链所需的非抽象更改

使用抽象

硬件和测量抽象打破了测试执行程序和与仪器交互的代码模块之间的耦合。测试执行程序不调用直接与特定仪器交互的代码模块，而是与MAL进行交互。这定义了基于通用仪器类型执行常见任务的操作或步骤类型。这些操作是仪器通用的，通常具有高级别名称，例如“信号输入”、“信号输出”、“连接”、“电源”和“负载”等。它们还采用测试特定的参数(而不是仪器特定的参数)，例如信号名称、连接名称、电源别名、电压/电流和负载方法(CV、CC、CP)等。映射框架使用配置文件将通用操作的测试特定参数转换为仪器特的定参数，如仪器引用、通道编号、矩阵行和列、GPIB地址和仪器配置约束等。此框架与HAL连接，以与配置文件定义的特定仪器进行通信。它基于MAL操作类型调用每个特定仪器对应的方法，并采用从配置文件中提取的仪器特定的参数。

如果将每个步骤视为一份烹饪食谱(煎饼)，则配置文件中的信息将是配料(鸡蛋、牛奶、黄油、面粉)，操作就是烹饪功能(混合、搅拌、拍打)，驱动程序就是厨房工具(碗、搅拌机、煎锅)，框架就是将所有这些整合在一起的指令。

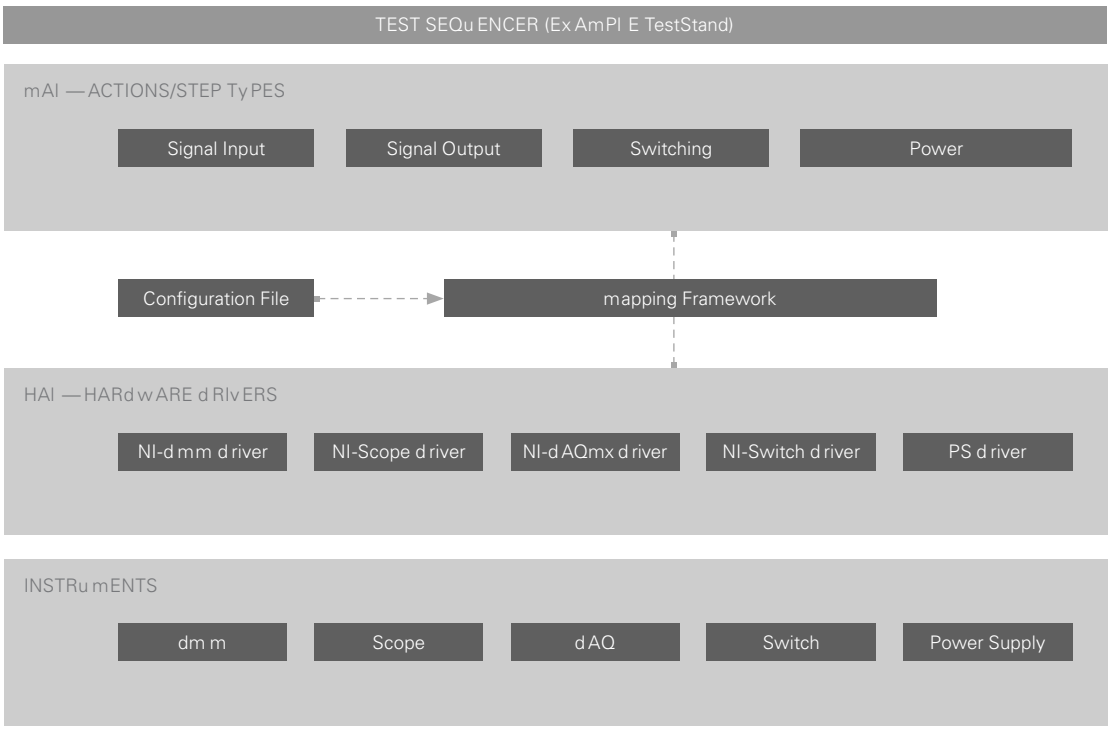


图 7 使用抽象的自动化测试软件的结构图

本节将继续介绍使用抽象的多路复用DMM示例。在本示例中，测试执行程序使用步骤特定的输入参数5v Rail来调用通用步骤类型“信号输入”。在这个特定的框架中，“信号输入”定义为三个设备操作：连接信号路径、读取测量设备、断开信号路径。这些操作使用5v Rail参数传递至映射框架。映射框架读取配置文件，查找5v Rail的仪器和通道详细信息。这些信息对应于PXI-2527多路复用器通道7的连接，以及PXI-4065 DMM在DC电压模式下执行的测量。然后，此框架调用适当的抽象驱动程序NI-Switch和NI-DMM，与配置文件定义的特定仪器进行通信。

执行不使用抽象的示例中的更改，用PXI-2532B矩阵替代PXI-2527复用器，事实证明，使用HAL/MAL框架执行更改更容易。由于所有仪器特定的详细信息都存储在配置文件中，并且HAL提供了与类似仪器进行交互的通用接口，因此只需更改配置文件即可。将PXI-2527复用器：通道7替换为PXI-2532B矩阵：r0/c0，r1，c7后，映射框架将自动提取更新后的详细信息并使用新参数调用新矩阵，无需更改测试序列或代码模块。

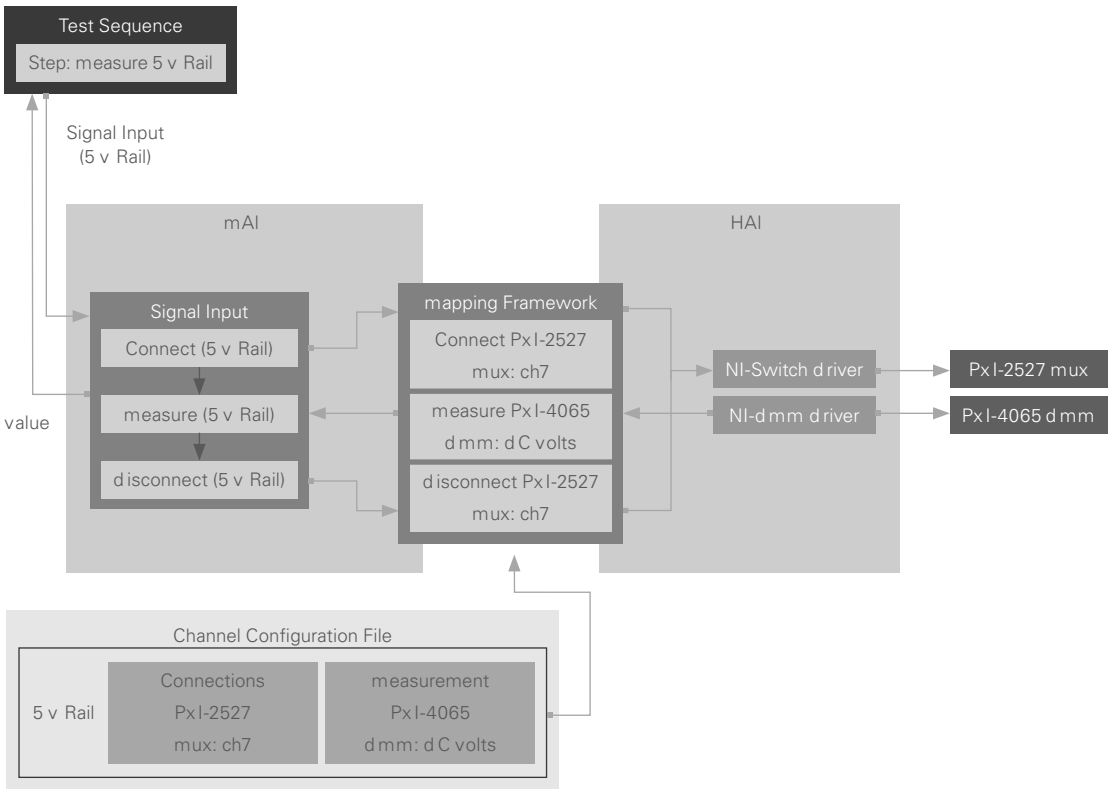


图8 在DMM测量时使用抽象框架调用函数

方法

在确定抽象框架时要考虑的首要问题是所有其他决策所基于的抽象范围。一种极端情况是不使用抽象，每个硬件接口都是对仪器驱动程序的直接调用。另一种极端情况是完全使用抽象，组件、通信协议、测量和配置格式之间每个可能的接口都定义一个抽象。本节将探讨涵盖各种可能性的一些方法。

方法1：仪器专用驱动程序

仪器专用驱动程序方法是自动化测试中最常用的方法，主要是因为该方法只需很少的编码、预测和规划工作。采用这种方法，可开发出连接特定仪器的底层代码模块。这些代码模块通常被称为底层驱动程序或仪器驱动程序，然后由更高层的代码模块或直接由测试执行程序调用。下面的框图显示了专为特定仪器开发的每个仪器驱动程序。在这种情况下，如果更换仪器，必须同时更改驱动程序和更高级别的调用。

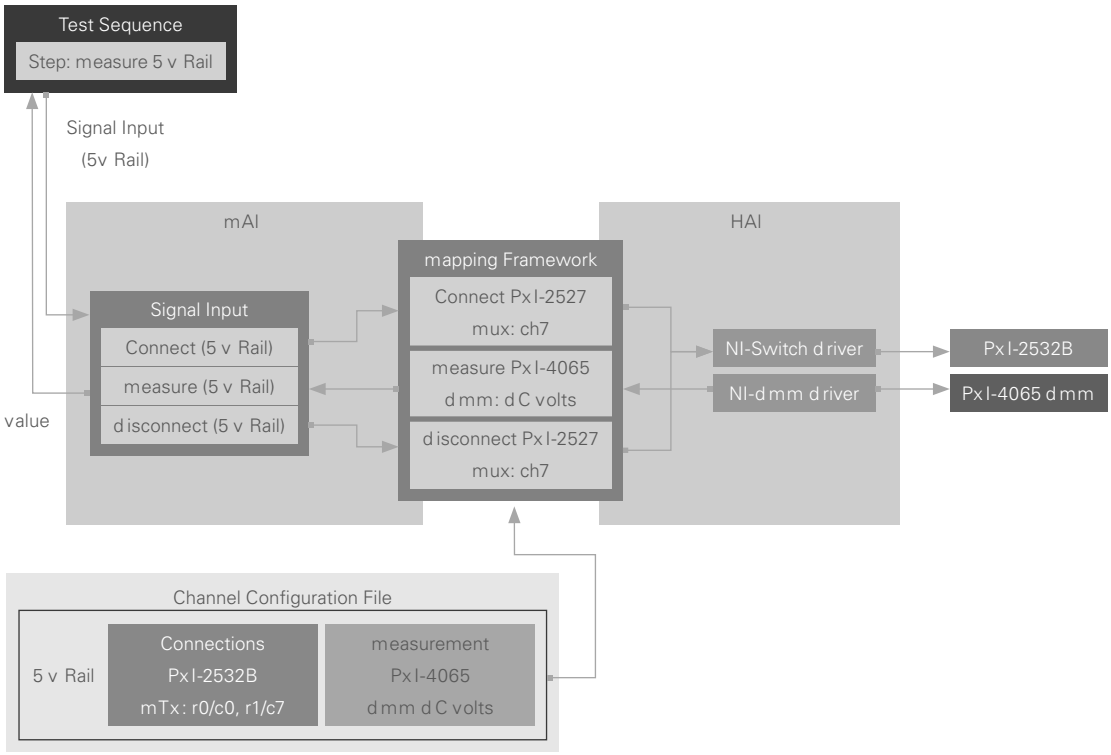


图 9 使用抽象时，能够轻松更新硬件且只需很少的软件更新操作—只需更新配置文件即可

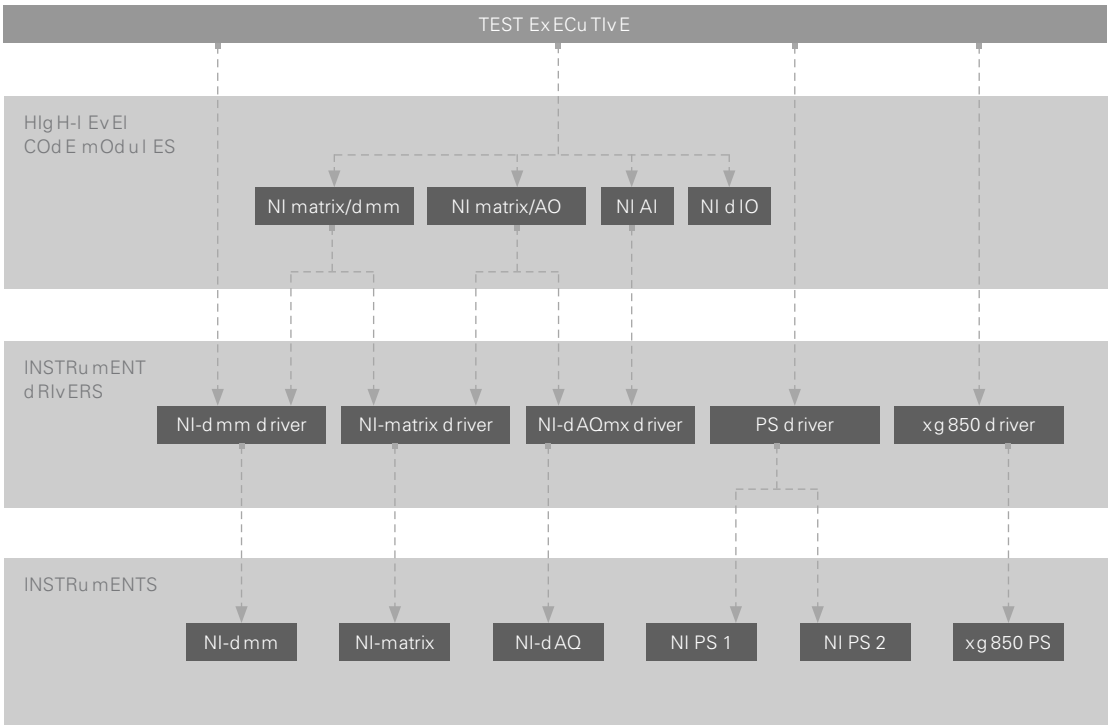


图 10 不使用抽象时自动化测试软件的仪器专用驱动程序方法概览

虽然此方法不包括任何抽象，但在提高驱动程序开发和交互的可靠性方面，仍然值得借鉴：

- 开发或使用仪器驱动程序包来连接每个仪器。
  - 底层驱动程序包实现了初始化、交互以及关闭与仪器的连接所需的所有功能。
  - 功能应简单、单一。
  - 驱动程序应该能够处理同一仪器类型的多个实例(例如同一系统中的两个相同电源)。
- 开发包装器仪器驱动程序，简化仪器接口。
  - 已有的驱动程序可能会包含数十个难以理解的函数，可以将已有的全功能仪器驱动程序封装到更简单的包装器仪器驱动程序中，提高易用性。
- 确保所有仪器连接都经过仪器驱动程序。
  - 这为所有仪器通信提供了单一进入点，从而简化了调试，减少了争用条件，并支持通过一个统一的位置管理仪器状态。
  - 如果已开发包装器仪器驱动程序，它应该成为单一进入点。
  - 驱动程序可以由测试执行程序直接调用，或者由更高级别的代码模块调用。
- 不要在驱动程序层实现测试特定的函数。
  - 测试特定的算法应该由更高级别的代码模块实现或在测试执行程序中实现。
- 确保仪器驱动程序之间不会交互。
  - 应由调用各个仪器驱动程序的更高层级代码模块或测试执行程序执行仪器之间的交互。

方法2：现成的HAL/MAL

若要将抽象整合到仪器驱动程序架构内，最快的方法是使用已有的HAL和MAL。虽然目前市面上提供的完全集成的HAL/MAL抽象框架有限，但许多硬件供应商已经在其仪器中实现了一定级别的硬件抽象；Switch Executive就是专门针对开关连接和路径规划而开发的MAL。基于这些已有的抽象来构建代码模块，就能够以最小的开发工作量提高ATE软件的适应性和抽象性。

现成的硬件抽象

已有的硬件抽象会采用通用的底层接口连接各种仪器。这能够减少所需仪器专用驱动程序的数量，同时降低系统中仪器更换所造成的影响。测试执行程序和更高级别的代码模块可以引用常规驱动程序，从而减少开发工作量并降低仪器更换造成的影响。当实现下面定义的其中一种抽象类型时，特定接口的I/O是固定的。因此，仪器更换通常不会导致代码模块更改。

可以通过两种方式利用已有的硬件抽象：仪器系列驱动程序和通信标准。仪器系列驱动程序往往是供应商特定的驱动程序，可以控制该供应商目录中特定仪器类型的许多仪器。通信标准提供了一种行业公认的方法，用于连接多个不同供应商提供的某些类型的仪器。可以使用这些标准来开发仪器驱动程序，从而控制各种类似的仪器。

通过仪器系列驱动程序实现硬件抽象

仪器系列驱动程序是供应商特定的驱动程序，可与常见的仪器产品系列进行通信。与IVI驱动程序类似，仪器系列驱动程序使用通用驱动程序实现与多个不同仪器的通信。常见的示例包括NI模块化仪器(NI-DMM、NI-Switch、NI-d CPower和NI-Scope)和Pickering PII PXI。仪器系列驱动程序可增强其所适用的产品系列之间的互换性。虽然它们不支持跨供应商或跨产品系列复用，但这些驱动程序通常直观、易于实现，并且包含每个仪器的大部分(就算不是全部)功能。

基于通信标准的硬件抽象

许多仪器制造商都遵循设备通信的行业标准。如遵循行业标准，制造商的仪器就可以与其他类似仪器进行互操作。两个最常见的标准包括可编程仪器的标准命令(SCPI，通常发音为“skippy”)和可互换虚拟仪器(IVI)。

SCPI

SCPI定义了测试和测量行业中用于控制可编程仪器的语法和命令标准。通过这些命令，用户可以设置和查询仪器的通用参数。SCPI命令可以通过多种通信协议实现，包括GPIB、LAN和串行协议等。开发符合SCPI标准的单一驱动程序后，就可以与同一类型的多台仪器(DC电源、电子负载等)进行通信，而无需开发仪器特定的驱动程序。在开发SCPI驱动程序时，请注意，虽然SCPI定义了一个通用的命令和语法标准，但不同的供应商实现此标准时有时会有微小的差异，因此开发100%标准的驱动程序有些困难。在选择符合SCPI的仪器和开发驱动程序时，务必要密切关注每个仪器的命令详细信息。

IVI

IVI是仪器驱动程序软件的标准，可增强仪器的互换性以及与符合IVI的仪器连接时的灵活性。该标准使用VISA定义了I/O抽象层。由于SCPI已经整合到IVI中，许多符合SCPI的仪器在定义上也符合IVI标准。IVI标准定义了13个仪器类型的规范，许多制造商都在遵循这一标准，因此每种类型的单一驱动程序就能够控制不同供应商的多种特定仪器。仪器类型包括DMM、示波器、任意波形/函数发生器、DC电源、开关、功率计、频谱分析仪、RF信号发生器、计数器、数字化仪、下变频器、上变频器和交流电源。许多PXI和现有仪器都遵循IVI标准，并且已有驱动程序支持多种编程语言，并可用于测试执行程序。

通过使用IVI驱动程序为符合IVI标准的仪器开发测试序列和代码模块，不同供应商的仪器看起来就会相同。您可以针对每种类型使用同一个驱动程序集，以连接许多可互换的仪器。相比使用仪器特定的驱动程序，使用具有类似功能的仪器替换符合IVI标准的仪器时，代码和序列更新工作将会大大减少。然而，虽然IVI驱动程序可以实现符合标准的仪器的大多数功能，但是有些仪器可能仍需要特定代码来执行自定义功能，而有些仪器则可能无法处理所有符合IVI标准的功能。最后，虽然两台仪器可以执行相同的IVI功能，但获得的结果可能不一定相同。因此，只要进行更换，就需要验证和测试仪器的功能。

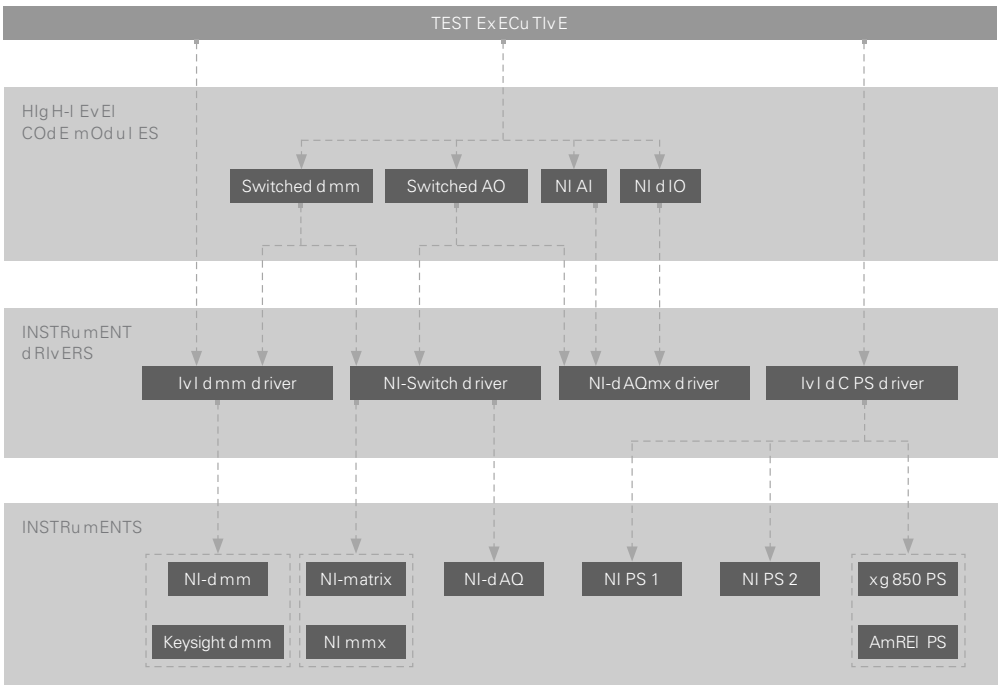


图 11 采用现成抽象的自动化测试软件概览

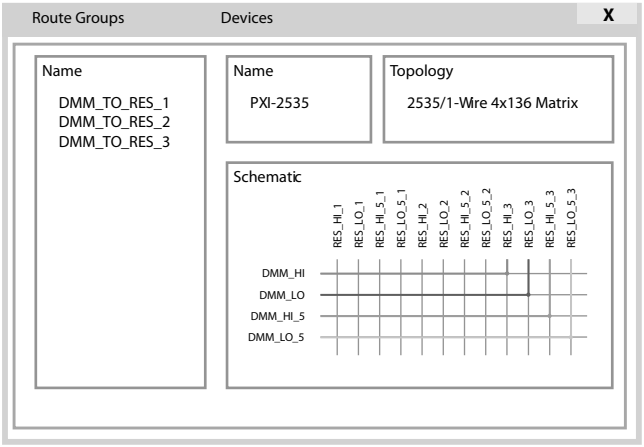


图 12 Switch Executive MAL配置界面

现成的测量抽象

虽然仪器普遍已经具有硬件抽象，但它仅从仪器的角度来实现抽象，而测量抽象是非常有限的。由于各测试系统的定制化程度都非常高，很难为测量操作定义一个标准。Switch Executive是最常用的现成测量抽象层，这是一款开关管理和路径规划应用程序，用于将符合标准的开关矩阵和多路复用器仪器组合成一个虚拟开关设备。此虚拟开关可以使用用户命名的通道和路径直观地进行配置和驱动。虽然Switch Executive仅适用于符合NI-Switch和IVI标准的设备，但却为从UUT或测试角度定义开关路径提供了一种极佳的方法。

首先，Switch Executive是一种图形化配置实用程序，用于在单台仪器内和跨多台仪器设置开关通道名称和路径。行、列、通道和路径组都可以进行配置和命名，以便直观地设置开关机制。其次，Switch Executive可集成到LabVIEW和TestStand中，为按名称设置和查询预配置的路径提供了强大的接口。当与TestStand测试执行软件一起使用时，Switch Executive可以逐步执行，以便在执行步骤的代码模块之前为开关仪器提供用户命名的接口。

Switch Executive是一个有用的MAL，可将开关连接抽象为测试特定的名称而不是仪器特定的名称。当与IVI开关硬件抽象结合使用时，会形成一个非常出色的集成HAL/MAL框架。但是，当使用非IVI开关或外部数字输出控制的继电器时，Switch Executive不适用。此外，Switch Executive仅适用于开关路径规划，无法扩展到其他测量类型。如果需要开关之外的其他集成HAL/MAL框架，就需要开发自定义代码。

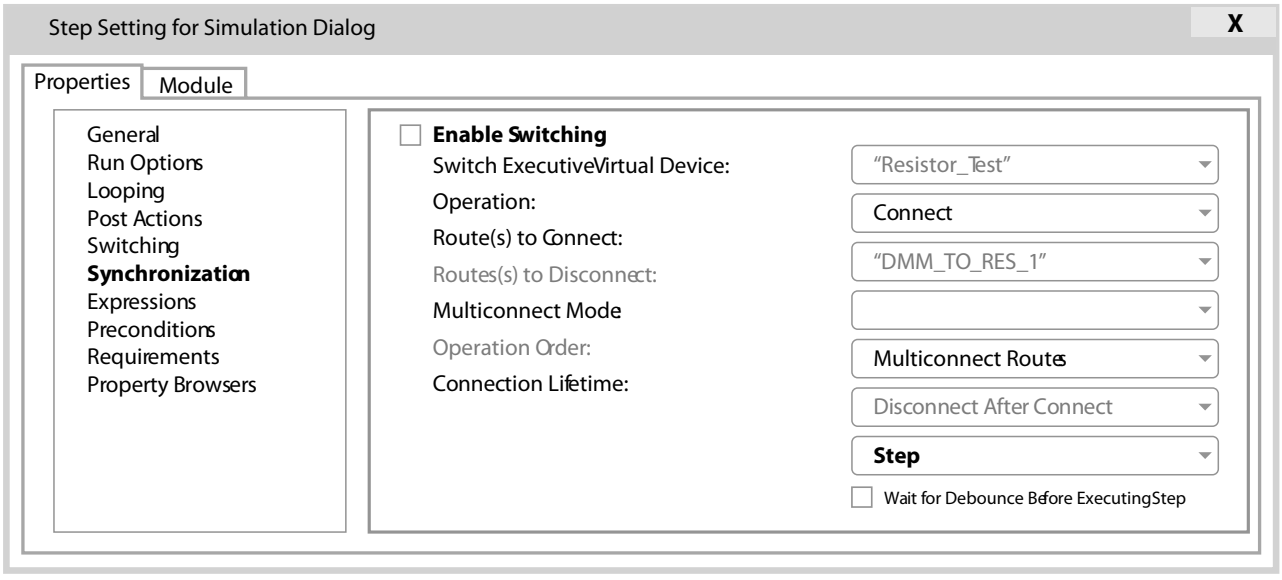


图 13 Switch Executive MAL测试设置

方法3：集成式HAL/MAL框架

集成式HAL/MAL框架提供了一种架构，用于实现由测试执行程序(MAL)调用的高级别动作，连接低层级驱动程序以与仪器(HAL)进行通信，以及在两者之间映射详细信息。此框架由三种主要类型的代码模块实现：动作、映射框架和硬件驱动程序。每种代码模块都由一组API定义。API是一组用于软件应用程序的工具(函数、协议、参数、语法)，定义了代码模块的运行方式及其与周围软件的交互方式。在基本的HAL/MAL框架中，有四种常见的API：测量API、配置API、硬件驱动API和仪器API。代码模块、API及其交互如下图所示。

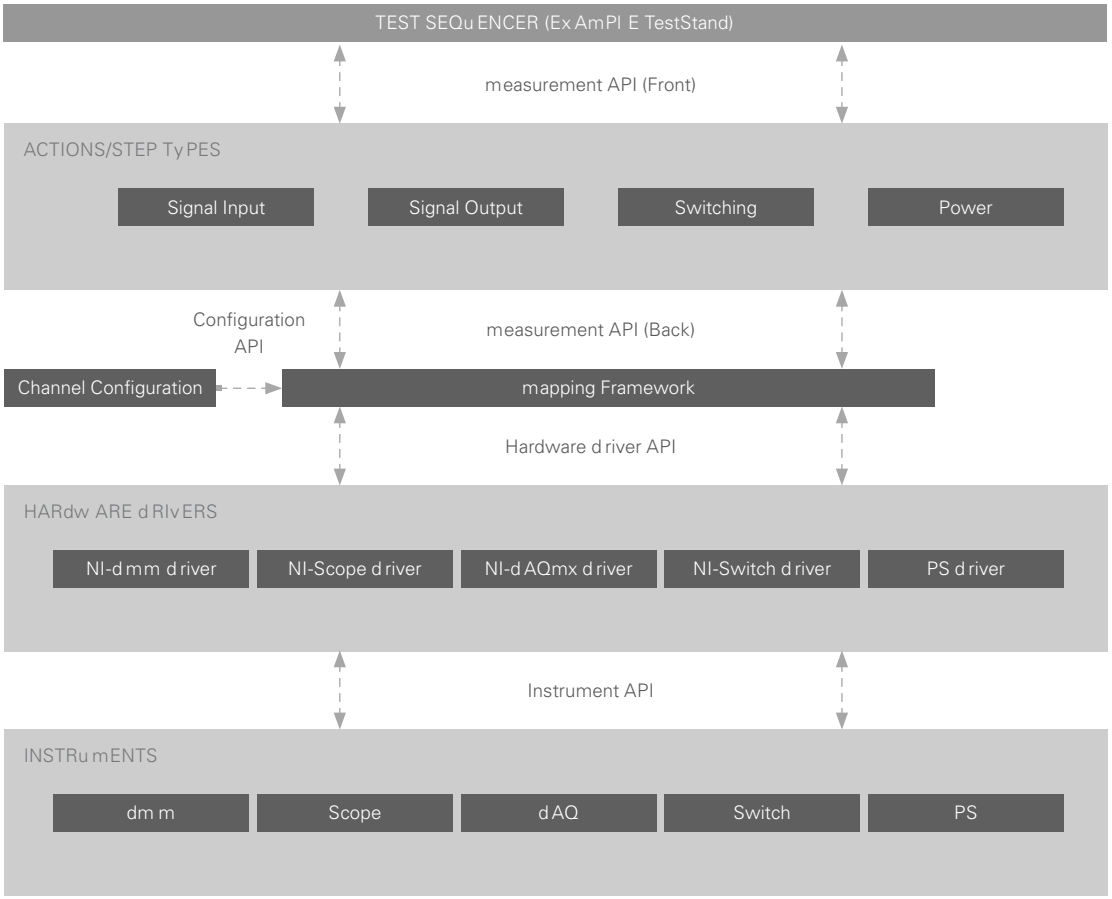


图 14 采用集成式MAL和HAL的自动化测试软件概览

三种类型的代码模块包括：

- **动作/步骤类型**—动作定义了MAL的功能。特定动作定义了每个测量类型(输入或输出)。动作可以像调用一种仪器类型的一个函数那样简单，例如进行开关连接；也可以像调用多个仪器的多个函数那样复杂，例如将开关连接与设置电源电压、电流和启用状态相结合。这些代码模块实现了用于定义其方法和参数的测量API。
- **映射框架**—映射框架是指使用配置文件中的默认值将高层级动作链接到底层仪器设备的内部代码。映射框架代码模块通过硬件驱动程序API与硬件驱动程序交互，并通过测量API与动作交互。
- **硬件驱动程序**—硬件驱动程序代码模块将通用设备类型函数调用(DMM、电源、开关等)转换为仪器特定通信(SCPI、IVI、NI-d CPower和专有通信)。因此，硬件驱动程序在一端实现硬件驱动程序API，在另一端实现仪器特定的API。

HAL/MAL抽象框架至少包含以下四个API中的一个：

- **测量API**—测量API定义了高层级动作及其特定参数。这是MAL的定义。测量API定义了一个所有动作都必须遵循的通用框架，然后支持每个动作定义执行其特定函数所需的API(参数和方法)。每个动作都必须至少实现后端测量API，映射框架使用此API将人类可读别名链接到特定的开关和测量仪器以及相应的通道。或者，可以开发API的前端，为每个动作提供更直观的接口。这个前端通常是一个配置对话框/向导。信号输入的测量API示例可定义信号输入别名和返回值输出。API还将为别名定义连接、测量，然后断开连接。
- **配置API**—映射框架使用配置API来详细说明如何将测量API转换为硬件API。配置API用于定义配置文件或数据库的参数、语法和内容。只有映射框架使用此API。例如，配置API可以规定配置文件是Microsoft Excel文件，并且每个信号别名应具有以下属性：名称、类型、连接详细信息、仪器、仪器配置和扩展。
- **硬件API**—硬件API是一种抽象的API，用于定义特定仪器类型必须实现的通用参数和方法。此API定义了HAL。例如，DMM Hardware API可能规定所有DMM必须能够初始化、配置(电压、电流、电阻、范围、分辨率)、测量(返回值)和关闭。
- **仪器API**—仪器API由每个单独的仪器定义，因此不是抽象层。每个仪器特定的硬件驱动程序都用于实现控制其特定仪器所需的函数和命令。这与在仪器特定的代码接口中使用的API相同，并且可实现用于该特定仪器的特定通信协议和命令。

为了更好地理解代码模块和API之间的交互，请查看多路复用DMM示例，其中详细说明了每个代码模块的输入和输出。

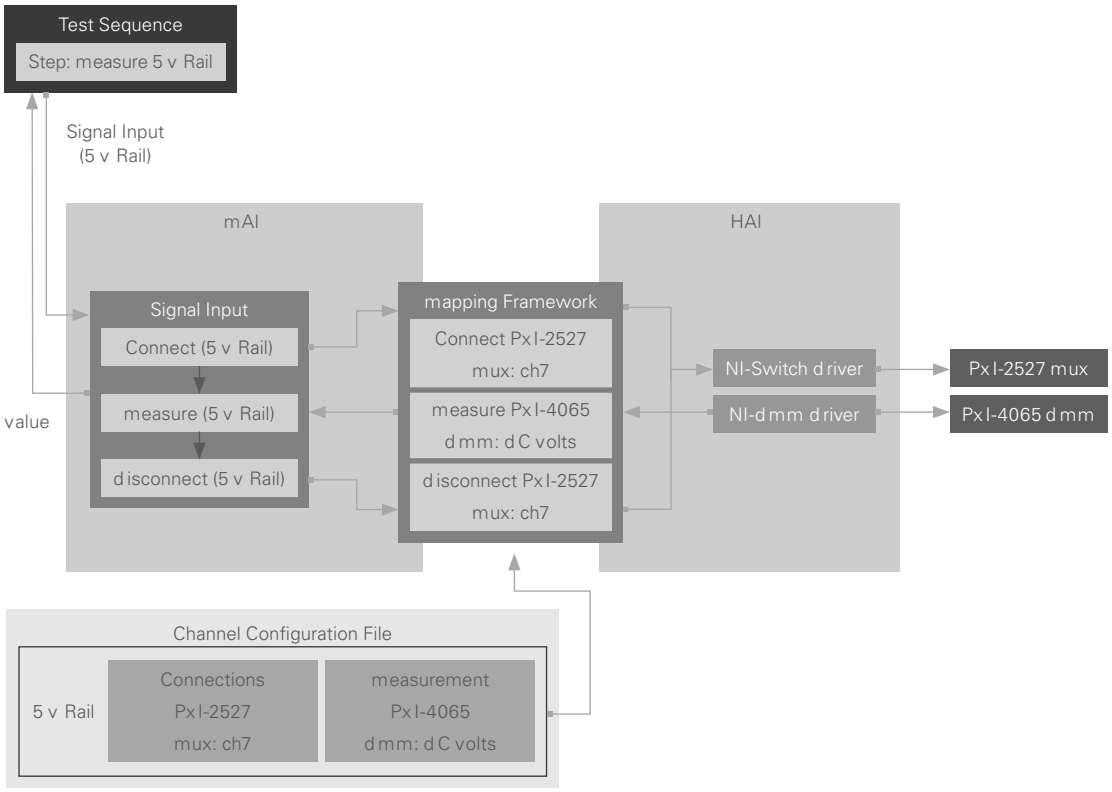


图 15 使用抽象框架的多路复用DMM测量

在本例中，信号输入块是动作代码模块，定义了信号输入应执行Switch Device Connect函数、Measurement Device Measure函数，然后执行Switch Device Disconnect函数。此函数的测量API明确了代码模块需要一个别名，此别名从测试执行程序接收，并传递到映射框架，然后从映射框架获取返回值传递回测试执行程序。

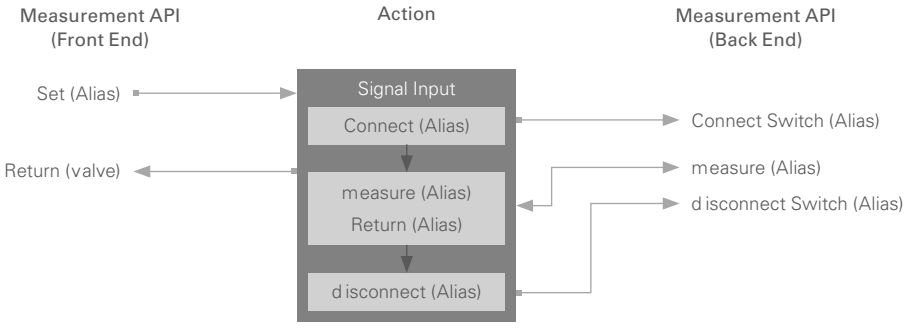


图 16 信号输入的MAL动作API示例

映射框架通过测量API接收来自动作的命令。然后，通过配置API解析配置文件中的别名数据，以获得正确的仪器ID和参数。配置API定义了系统配置的文件格式、语法和字段。然后，映射框架通过硬件驱动程序API将仪器特定信息传递到适当的驱动程序。

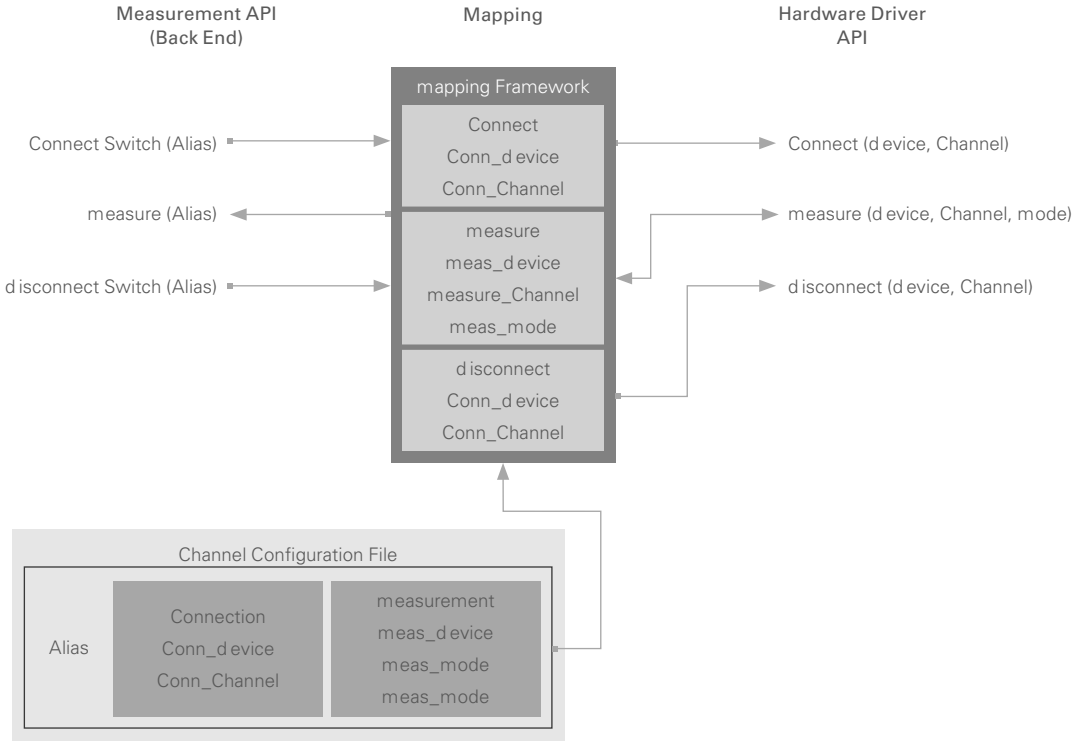


图 17 信号输入的映射框架API示例

映射框架使用通用硬件驱动程序API调用各个硬件驱动程序。然后，每个驱动程序都会说明通用设置的详细信息，并使用现成的方法和参数与特定仪器进行通信。

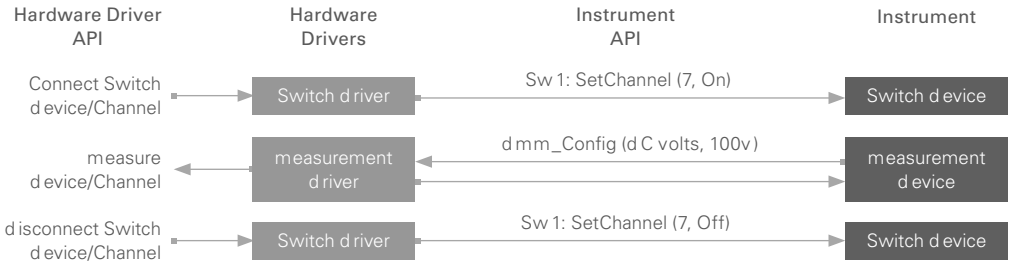


图 18 信号输入的硬件驱动程序API示例

方法4：HAL/MAL插件架构

插件是对集成框架的补充，可能非常有价值。真正的插件只是一个软件组件，可以在部署后进行修改，而无需重新部署整个应用程序。插件与主应用程序和/或框架分开存储在磁盘上，并在运行时动态加载。

虽然开发插件架构会带来一些挑战，但也能明确限制添加或修改功能的范围和风险，从而简化软件回归测试。开发时不使用插件的框架，在每次需要新的测量类型、仪器驱动程序或配置格式时，都必须重新构建。在不使用插件的情况下，整个源代码都内置到单个EXE中，因此即使对仪器库进行微小变更，也不能保证其不会无意中影响应用程序的其他特性。由于难以完全掌握源代码修改可能产生的所有影响，必须进行彻底测试。

插件架构便于开发人员添加或修复插件代码，而无需修改或重新部署底层框架，因而可实现最高的软件模块化水平。这通过编写一个仅依赖于抽象类或模块的框架来实现，该框架可动态加载所需的具体插件，并且通常仅在需要时才加载。插件架构成功与否取决于接口设计是否完善。换句话说，为了在测试框架中使用插件，框架必须知晓如何调用任何可能的插件组件。如果所有插件都采用一致的软件接口，那么在运行时加载这些插件只需要框架或测试应用程序知晓在哪里找到这些插件即可。

虽然这些是抽象框架的一些更常见的过程、API和代码模块，但它们肯定不是唯一的。每个框架都是独一无二的，并且有自己的要求、过程和实现方法。对于某些团队来说，这种抽象层级可能超出了需求。然而，在其他情况下，系统架构师可能需要注入额外的抽象层。根据框架架构师和用户的需求和能力，也可以采用不同的方法来实现这些API。有些工程师使用简单的动作引擎实现所有抽象，有些使用更高级的面向对象的编程，有些使用插件，还有一些更倾向于使用单个代码库。关键在于找到适合特定需求和能力的抽象和实现范围。同时还要明白，并非所有问题都可以通过抽象来解决，有时仍然需要采用仪器特定的代码。因此，在开发抽象层时，务必允许开发自定义代码来实现高级功能。这可通过允许更高级别的代码模块或测试执行程序获得仪器引用来实现。高级开发人员永远不应被框架所束缚。

抽象层方法的特性比较

无 ○ 部分 ◒ 全部- ●

抽象方法	方法1 不使用抽象	方法2 现成抽象	方法3 基本自定义	方法4 采用插件
支持使用以下组件替换各个仪器：				
采用相同通信协议的仪器	●	●	●	●
符合IVI标准或属于同一产品系列的仪器	○	●	●	●
采用不同通信协议的仪器	○	◒	●	●
无需修改测试序列即可更改仪器通道/接线(修改配置文件)	○	◒	●	●
从测试/UUT的角度执行测量/任务	○	◒	●	●
无需修改框架即可添加新仪器或新的测量	○	○	○	●

实际场景1

您是一家商业产品公司的测试工程师，负责开发新产品电子组件的功能测试。您需要测试三个PCBA和一个最终成品板。现在已有一个通用ATE仪器平台，但它已经过时，并且先前的测试项目因设备故障和仪器过时进展并不顺利。幸运的是，在项目进行过程中，已经开发了一个新的ATE平台，该平台支持可互换的测试头根据不同的成品板对仪器进行调整。

您的任务是开发测试序列和代码模块软件，以便与硬件工程师开发的仪器和连接件进行交互。您具备使用测试执行程序(与之前的平台使用的测试执行程序相同)的相关经验，并且有多年的软件应用程序开发经验。为了解决问题，已经讨论过使用抽象来帮助缓解之前系统中存在的过时问题。您必须确定这种方法是否可行以及如何实现。

是否使用抽象……

首先，您必须决定是否开发抽象框架，而不考虑抽象级别如何。考虑到已有IVI等现成抽象，答案几乎总是肯定的。只有在唯一一种情况下不需要开发抽象，即项目生命周期100%已知，并且永远不需要进行变更，但这几乎是不可能的。

您是否需要HAL?

接下来，您必须决定使用什么级别的硬件抽象。此时决策变得更为复杂，因为这涉及诸多因素。硬件抽象通常更容易理解，因此实现成本低于MAL。如果可以合理地使用预抽象驱动程序(如IVI和产品系列驱动程序)，则成本将更低。但是，一旦必须使用不属于某类驱动程序的仪器，则可能需要为每种仪器类型开发一个通用接口。例如，如果系统中有一些符合IVI标准的电源以及一个不符合该标准的电源，则可能需要开发适用于其中一种类型的抽象电源定义。确定抽象硬件定义通常需要了解该特定类型的大多数仪器的工作原理，然后可以使用这些信息来定义系统内每种仪器类型的通用方法和参数。

开发的目标是覆盖每台设备在合理预期下所使用的80%的功能。与团队讨论，确定每个仪器类型必须通过每个抽象仪器驱动程序实现的核心功能和参数。例如，团队可能会确定所有电源的核心功能应该是初始化、设置电压/电流/启用状态、回读电压/电流/启用状态以及关闭。虽然某些其他功能可能会在以后被电源使用，但这些功能并不一定必须作为系统标准架构的一部分。如果您对某种特定仪器类型不够了解，或者不确定需要哪些功能，请先构建小型架构。未来，您可以随时将新功能添加到标准架构中，但是在该框架被多个驱动程序使用后，很难更改该功能的参数或详细信息。

以下流程图可以帮助您确定适合选用哪种级别的硬件抽象。如果您不确定答案，可以假设一个更抽象的解决方案，或者假设一个不太抽象的解决方案。更抽象的解决方案需要进行更多的前期设计，但从长远来看可能会节省时间，而不太抽象的解决方案可以更快地启动和运行，但未来可能会出现问题。需要注意的一点是，要先问一下自己：我是否需要MAL。这是因为如果没有精心设计的HAL，就不能有效地实现MAL。

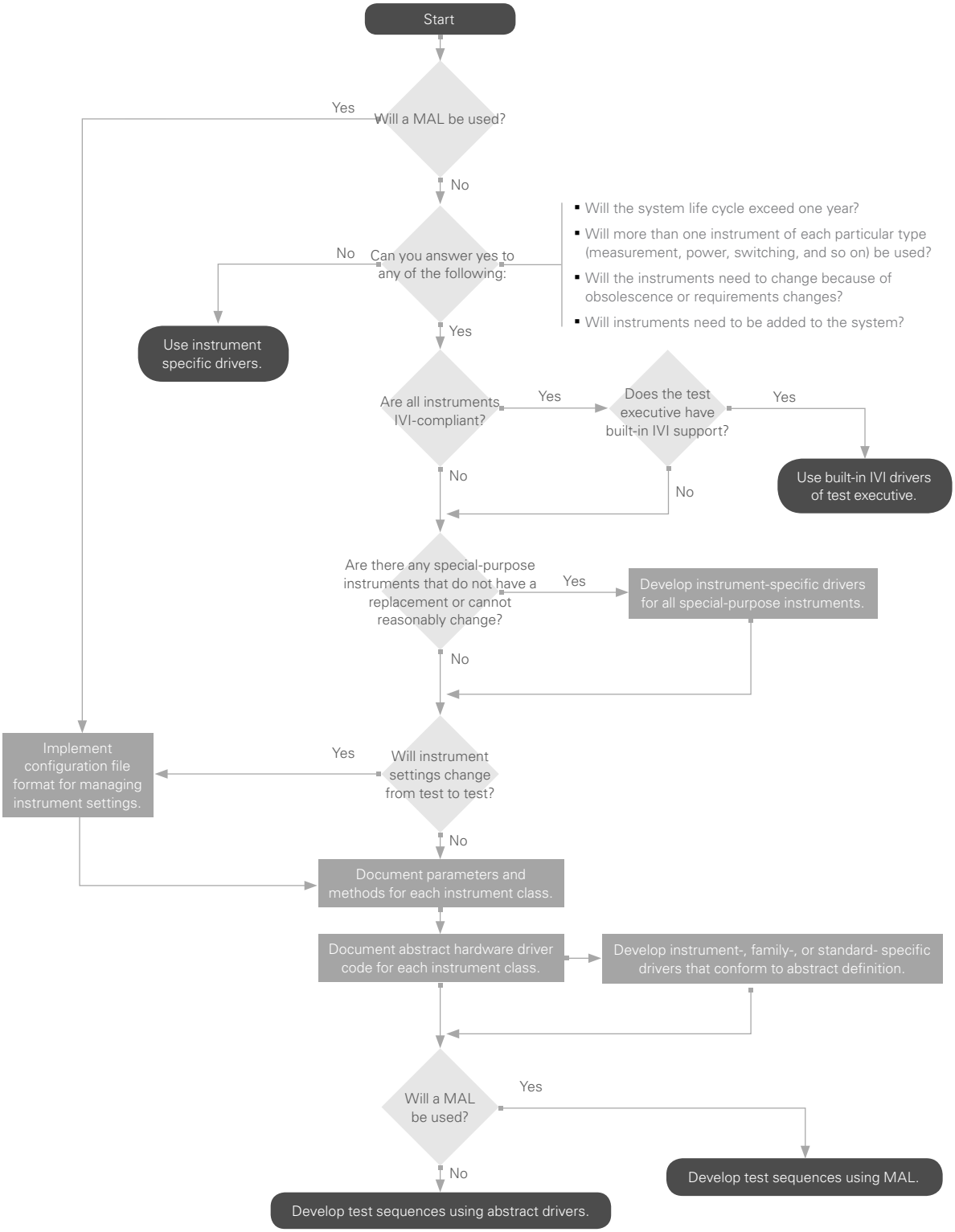


图 19 确定所要实现的抽象级别的决策流程图

您是否需要使用MAL?

要确定是否使用HAL，首先要确定是否需要使用MAL。这是因为如果不依赖硬件抽象，MAL几乎无法实现。因此，这个问题实质上是在问您是否需要采用集成式抽象框架。如果团队中的多名测试开发人员都缺少底层软件开发经验，则适合采用HAL/MAL。

以下几个主要问题可以帮助您决定是否开发MAL：

- 是否会有软件架构师可以规划和支持该框架？如果没有架构师/负责人，HAL/MAL难以获得有效的支持。
- 是否有多名测试开发人员缺乏软件开发经验？抽象框架的一个突出的优势在于，它能够缩短测试开发的培训周期。
- 系统生命周期是否足够长，可以支持多种产品？这可能需要一大笔前期投资，但使用频率越高，回报就越大。
- 您是否愿意开发和支持MAL？不使用抽象的效果要优于使用定义不明确和未有效实现的抽象。从长远来看，简单易用的HAL/MAL可以节省大量时间；但是，如果过于复杂或设计不当，可能会很麻烦，反而会增加开发和调试时间。

如果您对这些问题中的大多数回答都是肯定的，那么开发一个集成式抽象框架从长远来看很可能会获得回报。

实际场景2

即使抽象的所有好处已知，仍然存在如何衡量成本与回报的关键问题(单位通常是时间)。虽然抽象决策的第一部分通常从技术角度考虑，但是成本/效益决策必须在更高的业务层面做出。

成本是多少?

这个问题很难回答，因为它在很大程度上取决于开发人员的经验、编码能力和所需的抽象级别。但是，您可以估计各种组件的大致数量级，如下表所示。

抽象框架任务和成本

类别	任务	描述	估计小时数(低)	估计小时数(高)
规划	架构定义	记录动作类型、设备类型以及动作和设备之间的通用接口类型	24	48
	每种设备类型的HAL定义	记录每种设备的输入和输出以及方法	8 (每台设备)	16 (每台设备)
	每个动作的MAL定义	记录每种测量/动作的输入和输出以及方法	8 (每个动作)	16 (每个动作)
	配置定义	定义配置文件或数据库的格式、语法和内容	24	48
实现	映射框架开发	实现所有的软件，将配置文件映射到动作和抽象驱动程序-大部分底层框架都在此时开发	60	120
	抽象设备驱动程序开发	按设备类型进行抽象设备接口代码的软件开发-实质上是在构建仪器	4 (每台设备)	24 (每台设备)
	仪器驱动程序开发	为使用HAL的每个仪器特定驱动程序进行软件开发-填写到每个特定驱动程序的模板中	4 (每台仪器)	24 (每台仪器)
	动作开发	为MAL定义的每个动作进行软件开发-实现用于连接测试执行程序和映射框架的前端和后端API	4 (每个动作)	24 (每个动作)
总计		开发框架所需的总时间(不包括单个仪器驱动程序)-假设有五种设备类型和五个动作，并且每种设备有一个仪器特定的驱动程序	248	776

这表明完全集成式HAL/MAL抽象层所需的开发时间可以低至250小时，也可能超过750小时。根据抽象的级别，甚至可能超过1000小时。

如何降低成本?

涉及软件开发时，成本与复杂性密切相关。复杂性包括有利的复杂性和不利的复杂性，具体取决于其性质。目标是增加有利的复杂性，同时避免不利的复杂性。可以增加功能的复杂性是有利的。每个特性通常都会增加功能。采用灵活、可扩展的模块化代码来实现这些目标往往会更加复杂。但是当以简洁的方式实现时，这种复杂性会带来益处。由于规划不当、功能冗余和繁琐的意大利面条式代码而产生的复杂性是不利的，因为它增加了开发成本，但却没有增加功能。

您可以通过以下四种方式降低ATE抽象框架的复杂性：

- 预先规划架构。与大多数开发过程一样，前期规划和编写文档可以在开发过程中节省大量的时间和避免诸多麻烦。在前期进行规划并为API和代码模块编写文档，可以减少交叉功能和不必要的相互依赖，从而增强代码的可靠性，并减少不必要的复杂性。您不必针对每个API和代码模块的每个细微差别进行规划，但需要定义软件的主要交互、参数和基本功能。
- 规划不要太过超前。在开发大型架构时，工程师往往会过度设计，并尝试规划所有可能的场景。虽然前瞻性思维方式可能会带来益处，但最好是针对已知的情况进行设计。工程师在设计系统时经常会考虑最坏但通常不会发生的情况。这些工作只占总工作量的20%，但需要80%的时间去完成。结果将花费更多的时间来尝试处理假定的边缘情况，而不是专注于软件中大多数时间都会使用的功能。
- 认清一个事实，抽象并不能解决一切问题。抽象非常有用，但试图抽象出每一个可能的接口并不值得。相反，应在框架中包含自定义硬件交互，以解决通用接口无法实现的情况。为系统制定切实可行的规则，以减少抽象层。例如，将配置文件限制为单一格式(.INI、.XLS、database)，以降低映射框架的复杂性，或将动作限制为三个独立的硬件调用，以防止需要实现递归硬件驱动程序API调用的情况。
- 保持灵活、可扩展和模块化。虽然灵活性、可扩展性和模块化确实会增加复杂性，但在开发大型架构时大有裨益。此时非常适合采用插件架构，因为插件架构可定义低层级框架，而细节通过唯一的代码库实现。这意味着新功能可以在旧功能的基础上进行扩展，而不会对原有功能造成破坏。精心规划的插件架构不仅可满足已知情况的需求，还可在未来根据新的挑战进行扩展。

是否值得?

虽然抽象框架的开发可能非常耗时，即使顺利实现也是如此，但由于其回报往往大于付出而得到广泛应用。有多个关键因素可以提高回报率，让框架更为有效。其中许多回报可以通过节省的时间或精力来量化。下表列出了与任务相关的一些典型成本，并比较了非抽象系统与使用HAL/MAL抽象框架的系统之间的差异。

非抽象和抽象系统中与任务相关的成本

任务	估计用时(标准)	估计用时(抽象)	如何产生回报?
为新测试工程师提供测试软件平台培训	60小时 (每名工程师)	40小时 (每名工程师)	掌握如何使用抽象框架通常需要了解测试执行程序以及框架。在任何一种情况下,开发人员都必须了解如何与测试系统硬件进行交互。当使用仪器特定的驱动程序时,工程师必须了解每个驱动程序的详细信息和使用方法。然而,在学习抽象框架时,工程师只需要理解要执行的高层级动作,仪器详细信息都留在框架中。通常,这些高层级动作比各种仪器特定的驱动程序更直观、更容易实现。
基本功能测试序列的开发和调试(由经验丰富的工程师完成)	80小时 (每个序列)	40小时 (每个序列)	会加快测试序列的开发速度,因为硬件的详细信息都存储在一个固定位置,而不是存储在序列中的每个驱动程序调用中。从UUT的角度来看,测试序列会与硬件进行交互,从而使测试序列更直观、更好地匹配测试步骤。一般来说,如果框架直观易懂,可以将开发和调试时间缩短一半。
由新工程师执行的测试序列开发和调试	120小时 (每个序列)	60小时 (每个序列)	如果由新工程师或经验不足的工程师开发测试序列,会进一步缩短开发时间。由于框架规定了一组规则和功能,与使用仪器特定的驱动程序和代码相比,经验不足的工程师可以更好地使用已有步骤来开发序列。此外,如果框架直观易用,有产品意识的测试工程师无需掌握底层软件语言即可开发序列。
因仪器故障/过时或出现新的仪器需求而更新测试序列	驱动程序开发需要8小时,再加上4到20小时(每个序列)	驱动程序开发需要8小时,再加上<1小时 (每个序列)	当系统中的仪器需要更换时,测试序列也必须随之更改。对于非抽象平台,这意味着必须针对新仪器更新驱动程序调用的每个实例。仪器被引用的次数越多,更新所需的时间就越长。而采用抽象框架时,工程师虽然需要开发一个新的仪器驱动程序,但在开发完成之后,只需修改配置文件/数据库即可。
将测试序列转移到新的ATE硬件平台	40至80小时 (每个序列)	<8小时 (每个序列)	有时,整个系统都会升级,所有测试序列都必须迁移到新系统。通常,这些新系统采用的仪器与旧系统完全不同。在这种情况下,无论是否使用抽象框架,都必须开发新的驱动程序,但这些驱动程序开发出来之后,必须更新测试序列才能使用。如果使用非抽象序列,这一过程非常麻烦,有时还不如从头编写序列来得简单。然而,抽象序列通常可以在一天之内通过配置文件进行更新,而不必触及测试序列软件。

您可以对前面假设的商业产品公司的场景进行扩展,并基于这些数字看看开发集成抽象框架是否有意义,或者何时开发有意义。

首先,假设您自己开发全部四个测试序列。您必须从开发框架开始。在标准的非抽象场景中,必须开发仪器特定的驱动程序。在第二个场景中,主要使用现成的抽象开发驱动程序。在第三个场景中,您需要开发集成式HAL/MAL。

任务	开发时间（标准）	开发时间（现成的抽象）	开发时间（集成式HAL/MAL）
框架/驱动程序开发	80小时	100小时	500小时
测试开发(4个测试)	80×4=320小时	80×4=320小时	40×4=160小时
总计	400小时	420小时	660小时

表5 集成式HAL/MAL所需的前期开发工作量最大

当完成初始开发时，集成式HAL/MAL方法比标准方法多花费约260小时，但使用现成抽象的方法仅多花费约20小时。然而，初始开发结束并不意味着测试项目结束。

六个月后，研发部门发现需要执行更多测量，因此系统中的32通道多路复用器已无法满足需求，需要使用4×128矩阵代替。现在您必须开发一个新的驱动程序，并更新每个测试序列，以使用矩阵代替多路复用器。然而，如果您使用现成的抽象驱动程序，则不需要开发任何驱动程序即可处理新矩阵，并且序列中的函数调用也不需要更改—只需更改详细信息即可。如果使用集成式HAL/MAL，序列更新只需要在通道配置文件中完成。

任务	开发时间（标准）	开发时间（现成的抽象）	开发时间（集成式HAL/MAL）
新驱动程序开发	4小时	0小时	0小时
由于采用新矩阵，需要更新2个简单的测试序列	2×4=8小时	2×2=4小时	2×1=1小时
由于采用新矩阵，需要更新2个复杂的测试序列	2×16=32小时	2×12=24小时	2×2=4小时
额外时间	44小时	28小时	7小时
总计	444小时	448小时	667小时

表6 集成式HAL/MAL方法更容易更新，但仍需要执行更多的开发工作

即使到现在，集成式抽象层还未开始获得回报，而现成的硬件抽象几乎已经达到收支平衡。现在假设又有一个新的测试项目，需要对额外四个成品板进行测试。遗憾的是，您的工作太忙，无法自己开发这些序列，因此招聘了两名新的测试工程师。您必须对他们进行系统培训，然后让他们开发序列。

任务	开发时间（标准）	开发时间（现成的抽象）	开发时间（集成式HAL/MAL）
培训/学习时间	60×2=120小时	50×2=100小时	40×2=80小时
测试开发(4个测试)	120×4=480小时	100×4=400小时	60×4=160小时
额外时间	600小时	500小时	240小时
总计	1044小时	948小时	907小时

表7 当开发更多测试或需要进行重大变更时，集成式HAL/MAL方法从长远来看能够获得回报

此时，最初开发框架的500小时投入已经开始获得回报，比标准开发方法缩短了100小时。随着新测试的开发、变更以及产品生命周期的持续，初始投资将持续获得回报。

在主观层面，使用抽象还会获得更多回报，但这些回报难以用数字来呈现。开发测试所需的时间也会大大缩短，因为采用HAL/MAL后，在硬件尚未完全定义之前开发软件变得更容易。通过维护标准框架，您可以确保在一个单独的存储库中添加新的驱动程序和测量，可以管理错误，并且可以减少工程师之间的代码差异。标准化有助于让每个人(测试工程师、制造工程师和技术人员)保持一致，从而更好地为系统提供支持。尽管如本文档所述，抽象还有诸多其他优势，仍需根据个人能力与ROI计算结果确定哪种级别的抽象符合您的特定需求。

附加信息

TestStand

TestStand是一款行业标准的测试管理软件，可帮助测试和验证工程师更快速地构建和部署自动化测试系统。TestStand包含可立即运行的测试序列引擎，支持多种测试代码语言、灵活的结果报告功能和并行/多线程测试。TestStand不仅包含了许多现成的功能，而且也具有高度的可扩展性。因此，全球数以万计的用户均选择TestStand来构建和部署自定义的自动化测试系统。NI提供培训和认证业务，每年培训和认证的TestStand用户超过1000名。

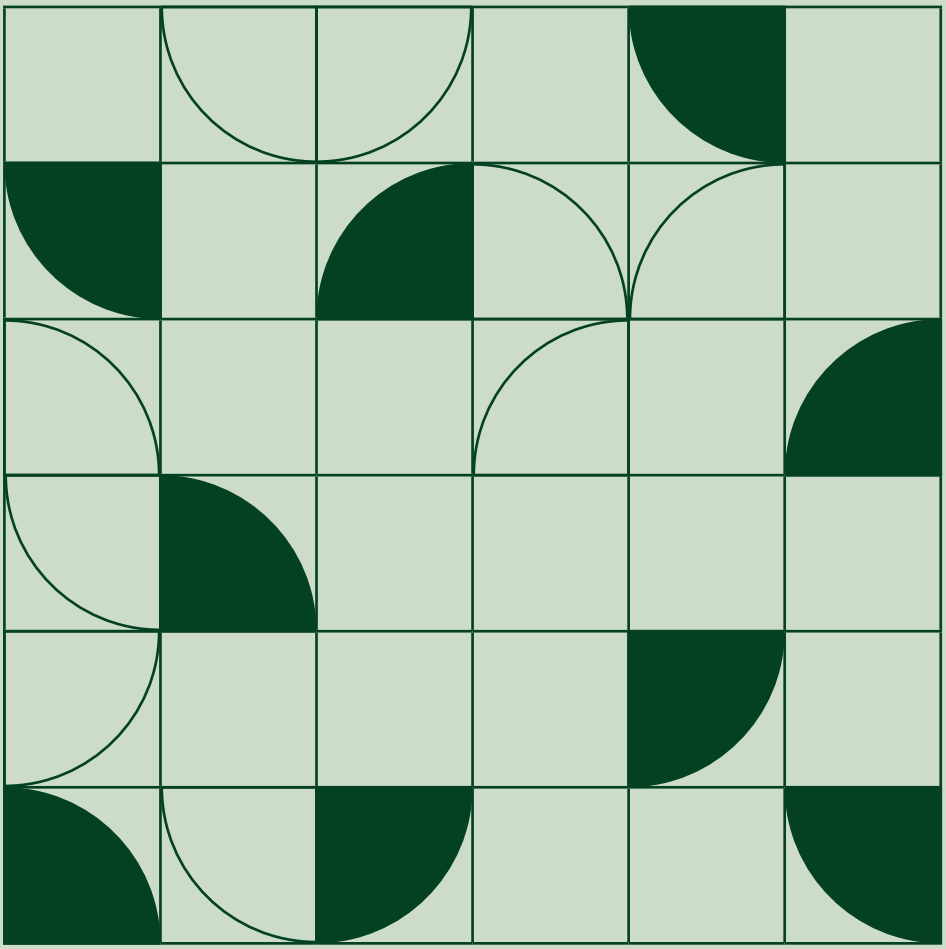
了解有关TestStand的更多信息。

关于Bloomy

Bloomy致力于提供电子功能测试，航空电子设备、电池和BmS硬件在环(HIL)测试以及航空航天系统集成实验室(SIL)数据系统相关的产品和服务；同时提供一流的LabVIEW、TestStand和VeriStand应用程序开发服务。Bloomy是NI联盟合作伙伴，自24年前联盟合作伙伴成立之初便已加入，一直是NI最重要的白金级和精选级合作伙伴。

了解有关Bloomy UTS软件套件(包括集成式HAL/MAL)的更多信息。

机架布局和热分布



简介

每时每刻都会有新项目摆在测试工程师的桌前，期望他们开发的测量系统不仅能满足规格要求和发布期限，并且质量优秀，性能可靠。在理想的情况下，工程师会有充足的时间和资源来进行深入研究、建模和仿真，可以打造出完美的系统。不幸的是，实际的项目安排通常不允许花费大量时间和资源来开发完美的系统。Control Engineering公司在2014年8月进行的系统集成调研表明，只有67%的系统项目能在预算内按时完成。面对着紧迫的发布进度和苛刻的项目时间要求，必须全面考虑测量系统中所有可能影响测量质量的因素，否则可能会导致进度推迟、成本增加甚至性能降低。这些因素包括仪器的选择、连接和线缆的质量以及测量方法的实现。然而，热量对测量质量和测量系统可靠性的影响却时常遭到忽略。

本文将介绍通过改进设计来规避风险的相关知识，帮助您了解热量对测量质量的影响、学习基本设计方法并探索用于设计机架式测量系统的热建模工具。

机架式设计中热量的重要性

在通用测量系统中，热量的来源有多种可能；然而，在机架式测量系统中，机架内产生的热量以及与机架周围环境的热交换是影响测量的热变化的主要来源。

出于以下几个原因，您需要关注热量问题：

- **良好的设计实践**—了解热量的影响并在设计系统时考虑这些因素是很好的设计习惯。在了解到热量如何影响系统后，就可以避免让热量成为影响测量的主要变数之一。请记住，在规定的温度范围之外操作仪器，可能会影响仪器的质量和预期寿命，正因如此，设备需要具备良好的热设计。
- **系统不确定性**—热量会始终存在且难以完全消除。因此，通过更深入地理解热量是什么，可以更好地评估热量对系统不确定性的影响，并在测量误差和测量结果中更准确地解释这些影响。
- **系统稳定性**—稳定性对于良好的测量系统不可或缺。如果观察到易变性，通常很难确定根本原因和/或解决方法。而系统中的热变化可能增大易变性，进而导致错误的测量结果。控制系统中的热量有助于最大程度降低这种风险。
- **产品质量**—产品需要特定的热环境以确保最佳性能，特别是在调整过程中。尽量减少系统热量对产品性能的影响有助于提高整体产品质量。

热量和仪器

对于仪器而言，热量是必须着重考虑的因素。仪器只有遵循特定温度要求才能满足规范。大多数仪器都会出现温度漂移，如果温度不稳定或超出规定要求，则测量结果会有所不同。要正确理解和信任测试解决方案的测量结果，就必须明确这些影响。

例如，电信和IT等相关行业都已按照建议或允许的温度范围提供了最佳实践。大多数设备制造商都会遵循这些最佳实践。但一些设备有其自己的规范，此时，设计需要同时满足这些特定的规范以及行业最佳实践。这些行业的主要关注点包括长期可靠性、系统正常运行时间和较低的总体拥有成本(TCO)，这些因素与自动化测试密切相关。同样，自动化测试工程师也应考虑与机架系统和热量有关的潜在影响。

在这些行业中，如果热量管理不善，可能会带来更高的运营成本。例如，如果冷却系统出现故障，温度的升高会增大系统其他部分的负荷，从而导致设备寿命缩短。如果温度过高，IT系统可能会在CPU级别出现计算错误，引发应用程序故障。我们可以部署备用的冷却系统，但这种做法会增加TCO。最大的问题是，由于自动关机导致的停机会导致服务中断，任何停机都等同于资金损失。

国家标准和国际标准

在国家标准方面，网络设备构建系统(NEBS)和美国采暖、制冷与空调工程师协会(ASHRAE)分别为电信和IT设备制定了指导原则和最佳实践。

ASHRAE是一个致力于为多种领域提供最佳实践的组织，而NEBS则更加专注于电信设备。ASHRAE可能会参考NEBS中关于外壳和机架式设备的一些指导，但ASHRAE的最佳实践看上去能够更全面地涵盖外壳设计的方方面面。

虽然这些国家标准无法直接适用于测试和测量认证，但它们在机架设计和性能方面与自动化测试遵循相同的原则和准则。

国际电工委员会(IEC)制定的国际标准涉及外壳的热性能。外壳制造商主要参考这些标准来设计或测试外壳，或为客户提供使用指导。

- IEC 61587-1规定了在室内条件下空外壳(即机柜、机架、机框和机箱)的环境、测试和安全要求。
- IEC 62194-1提供了在室内和室外条件下评估空外壳热性能的方法。

电信和IT行业的设计目标与测试和测量行业类似，但主要关注的领域和面临的挑战略有不同。测试和测量行业的设计目标更侧重于满足单独设备的规范，因为尽管该行业内各公司都有自己的最佳实践，但该没有针对测试和测量设备机架的通用标准。用户的主要关注点是确保每台仪器以及被测设备(DUT)符合规范。这一点甚至比长期可靠性或正常运行时间更为重要，因为长期可靠性或正常运行时间在很高的温度下才会受到影响，而测量精度在相对较低的温度下也有可能下降。

就挑战而言，自动化测试系统存在着更多的限制。其中之一是测试机架通常用于人进人出的环境或无管控的生产设施中。与只有静止物体的无人服务器机房不同，这增加了房间/工厂的整体热分布的随机性。

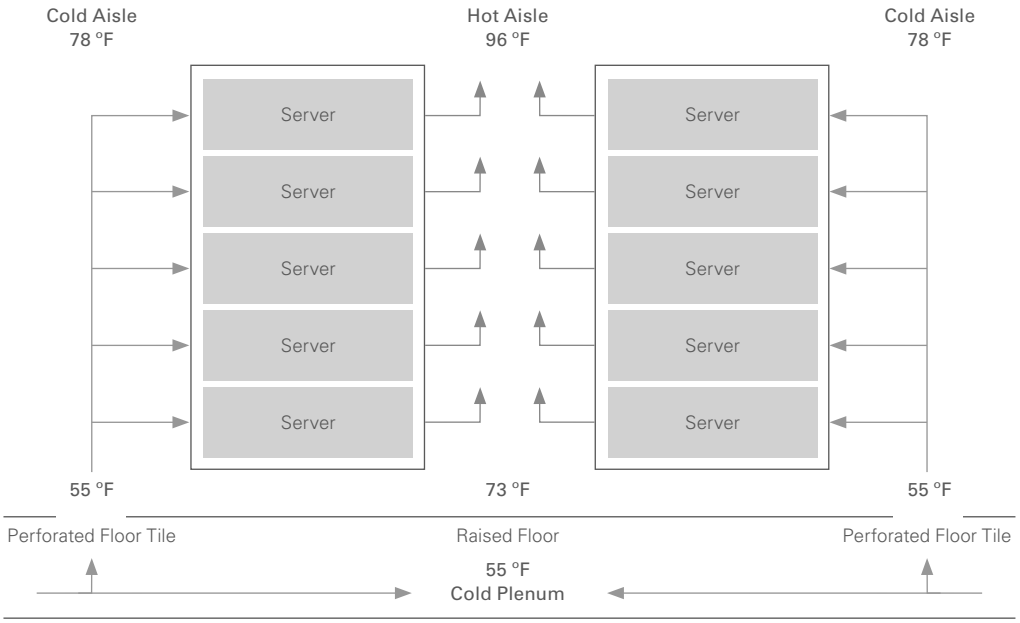


图1 服务器机房/设施处于受控环境中，与放置在无管控的混乱环境中的测试系统不同

热量对DUT、测试系统或测试结果的影响

首先，最根本的影响在于仪器和DUT的精度。如果无法确保仪器始终处于正确的环境温度下，仪器精度将会降低。

例如，如果环境温度在调整和验证阶段的变化足以改变仪器或DUT的精度，则校准可能无效。这也可能导致用户在制造过程中无法正确判断产品是否合格。管理此风险的最佳方法是在计算测量不确定度时考虑热偏移。

即使温度处于规定的操作范围内，来自不同测试站的数据也可能存在一些差异。在开发环境而非生产环境中收集的数据也是如此，测试站周围环境温度的变化是造成此种差异的主要原因。

尽管存在偏移，但大多数仪器仍会在环境温度下工作。这意味着，环境温度即使发生微小的变化，也可能导致仪器温度发生变化，从而增加各个测试站数据互相存在差异的可能性。

温度的变化不仅会导致开发和生产数据存在偏差，也可能在验证和确认以及测试开发阶段之间产生偏差。通常，验证和确认是在办公室环境中的台式设备上执行的，而测试开发则是在受控环境中使用机架式测试站完成的。即使仪器和后续测试系统是一样的，这也会导致仪器所处的环境完全不同。某些仪器甚至在某些温度范围内具有不同的测量规范，因此在设计和测量过程中使用适当的计算规范非常重要。

此外，前文探讨的环境都无法完美地模拟生产制造环境，因此如果环境中存在未知因素，建议在设计中采取保守的做法，谨慎应对这些环境可能出现的问题。例如，下图对办公区和受控环境中测试台前端几个小时内的室内环境温度数据做了比较。

受控环境是一个配备专用空调的小房间，可以看到随着温控器开启和关闭，温度会发生明显的变化；该房间温度保持在23°C至25°C之间。办公区温度更稳定，但更热一些。

工作日刚开始时(图表最右侧)，两个区域的温度均小幅上升；当员工到达时，室内温度会由于体热和开门而有所升高。请注意，办公区域的温度会随着季节、位置、楼层和其他因素而发生变化。相比之下，受控环境由于配有专用空调，全年温度相对恒定。鉴于上述事实，在进行验证和确认或开发测试和收集数据时，必须始终跟踪环境温度，以便在验证和确认数据与测试数据存在差异时进行数据分析。

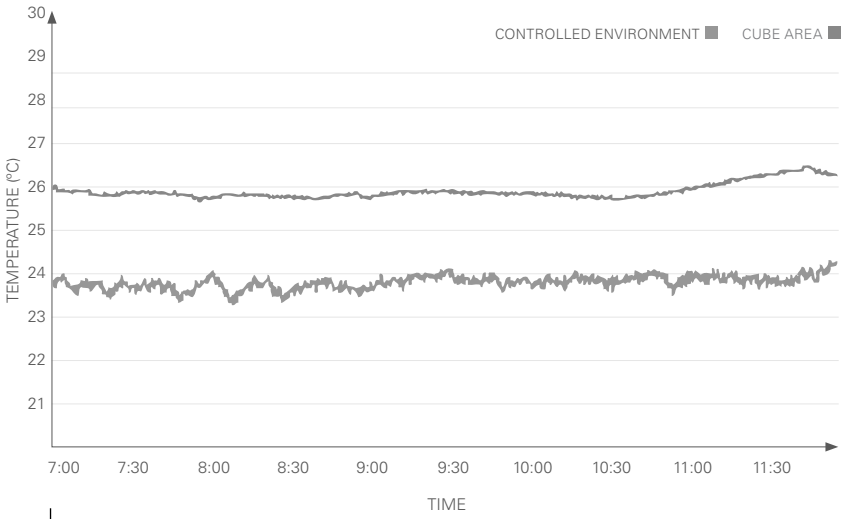


图2 环境温度

机架式系统的热分布

在考虑系统中的热分布时，人们往往想得过于简单。最典型的莫过于“下冷上热”的观点。还有人认为温度梯度在机架上是均匀分布的，但在大多数情况下，这些简化模型并不完全正确。

在实际系统中，热分布受到多个变量的影响，因此，热分布也会不断变化。如果用红外测温枪或热电偶适当地测量系统，可以发现机架式系统存在局部热区和水平/垂直轴温度梯度不均匀等特性。这是因为环境中并非只有冷空气和热空气；热分布取决于机架布局、单个设备的风扇速度、进气口和出气口的位置、系统中每个设备的功耗以及系统中所有风扇综合形成的气流。

这一点很重要，因为这意味着机架顶部不一定是最需要关注的位置，不能根据常见的简化观点一概而论。需要进行全面评估以了解每个系统特有的热分布情况并解决相关问题。

以下示例有助于解释热能在典型机架式测试系统中的分布。

图 3 通常的假设是上热下冷，热量均匀分布，这一假设会导致结果不理想

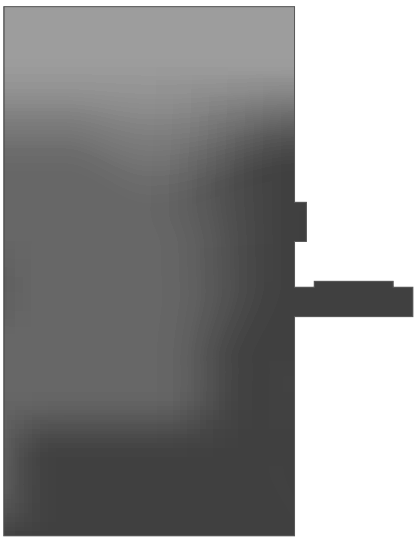


图 4 局部热区导致整个测试机架的不同位置具有不同的温度

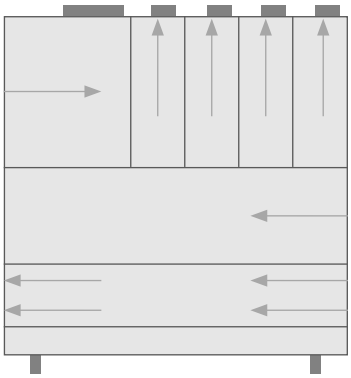


图 5 模块化电源的气流流向俯视图

首先要注意的是局部热区，这取决于系统的机架布局、设备规格及使用情况。在本测试站示例中，电源周围的空气温度最高，其次是PXI机箱。测试站其他部分的温度似乎比这些局部热区要低，根据系统的不同，处理这些局部热区的方式也可能有所不同。另一点要注意的是，底部热区在x轴上的分布并不均匀。

热不均匀性是如何产生的?

通常,热不均匀性是由设备的气流模式而引起的。例如,电源由多个单独的电源单元和电源主机箱组成。从图中可以看出,主机箱内的气流方向为从左到右。这意味着大部分暖空气处于仪器的右侧。

后端排出的气流来自不同的模块;因为并非所有模块都同时运行,所以后端的温度通常不会像右侧一样高。此外,机架系统的热分布还会随使用情况而变化,因此需要进行特性分析,而不是简单地观察给定情景下的温度。

系统的热分布应该是怎样的?

需要了解机架式系统的多个方面,才能确定热分布的情况。

首先要了解各系统的独特需求:

- 系统级的规范要求是什么?
- 系统会在什么环境中运行?
- 需要使用哪些仪器,这些仪器有什么温度要求?
- 应用程序只需将温度维持在一定范围内,还是需要温度保持稳定?

例如,如果您的DUT是PXI模块,并且在切换DUT时需要启动和关闭DUT PXI机箱,则机架的热分布会反复变化。要了解具体的变化情况,需要先了解机架热分布中的所有不稳定因素。

最后还要注意,并非机架内的所有点都必须保持温度不变。部分区域的温度比其他区域更高属于常见现象,只要保证所有仪器进气口都在规定的温度范围内即可。

设计方法

下一节将重点介绍机架式系统从设计到部署的最佳实践。

在开始机架设计之前,首先应该了解所使用仪器的几个关键元素,这些元素会影响整体设计:

- **评估设备的进气口和出气口**
  首先应仔细研究仪器,了解模块上进气口和出气口的位置。满足设备温度要求的能力在很大程度上取决于仪器进气口的温度以及散热点的位置。了解这些信息将有助您顺利地绘制出机架的温度分布图。
- **了解规格和温度要求**
  通常,设备会有特定的存放、运行和校准温度要求。您关心的是哪些部分?您对于这些要求是如何理解的?请了解每种仪器的温度规定,以及温度对仪器性能或与担保性能相关的具体规格的影响。例如,3458A规定环境温度必须保持在23°C±5°C间才能保证设备符合规格。此外,如果温度相比上次自动校准后有±1°C的相对变化,便需要再次执行自动校准。第一项规格涉及环境的绝对温度,第二项则是相对于上次校准的温度。需要了解这些差异及其对解决方案的潜在影响。

- **了解设备对环境温度的定义**
  大多数传统的台式仪器将环境温度定义为仪器周围环境中的空气温度。通常，对于PXI产品和机箱，环境温度定义为紧邻机箱风扇进风口处的空气温度。因为像PXI-1045这样的机箱需要大约1.75英寸的间隙来实现空气的适当流动，可以放心地假设，在进气风扇周围的上述间隙内测量温度，即可得到所需的环境温度。一种常见的错误是认为仪器的环境温度就是仪器所在房间的温度。

  一般来说，如果在桌面上使用仪器，并且附近没有干扰性的热源，这个观点可能是正确的；但是，在机架式设计中，必须使用机架内部特定区域内的空气温度作为仪器的环境温度。机架式设计中的仪器更易受到热问题的影响。

  务必根据具体应用和仪器的使用情况准确了解每个设备的环境温度。

在选择机架或开始具体的机架设计之前，请了解机架可能遇到的预期热负载。可通过对系统要使用的所有电子元件进行功率预算来轻松获得相关信息。了解功耗有助于深入了解热负载。

对所有电子元件进行功率预算

应考虑设计中包含的所有仪器和外围设备，包括测量设备、PC、显示器、电池备份以及机架内任何可能的发热源。对于这些设备，请参考产品规格以确定每个设备的功耗。

一般来说，产品规格列出了最差情况下的功耗(在完全使用或满负载的情况下)，该数据通常不代表设备的一般或平均性能。在设计中通常以额定最大功耗的60%为基准。尽管如此，机架在将来可能会用于其他用途，导致热负载增加，因此在计算功率预算时应留有余量。

理想情况下，如果可以提前测量设备的实际功耗，则可以最有效地预测整体功耗；但这在系统规划阶段基本无法实现。目前最合理的做法是先完成系统设计，再回头进行测量和文档记录。

温度要求和气流分布

根据仪器温度要求和气流分布，绘制出仪器所需的大致位置。热量由下往上流通，因此通常情况下，机架底部较冷，顶部较热。设计时应将最敏感的仪器放在机架的底部。可以使用特定技术为其他机架位置处的仪器创建可接受的热环境，但通常会产生一定的成本。

某些仪器的放置可能受到可用性的限制，需要评估和了解仪器位置的影响。也许需要额外考虑如何处理气流或如何为非常规的设备进气口提供较冷的空气。在设计时请始终记住这些考量因素。此外，还要考虑仪器规定的任何间隙限制。通常，仪器必须与设备或设备的进气口/出气口保持一定距离。请确保这些规范得到满足。

基于布局确定机架尺寸

可通过仪器布局确定所需的机架尺寸。在选择机架时应考虑外部限制因素，例如占地面积和房间高度。

所有仪器的废气应通过一条通畅的通道排出机架。在此示例布局中，可以看到有一个仪器阻碍了其下方仪器的排气。另一个错误的做法是让冷热空气相遇。您可以通过更巧妙地布置机架来解决这些问题，但首先需要充分了解仪器的进气口/出气口以及预期的气流。

重新放置仪器，为所有仪器的废气提供通畅的排气路径。此外，通过机械分离以确保所有仪器尽可能从外部空气中实现进气。

对于仪器进气口位于机架内部的情况，由于仪器的安装方式，某个仪器的废气可能会再循环其他仪器的进气口。可能会有多个仪器的废气进入彼此的进气口，导致机架内的环境温度显著升高，并且越靠上的位置温度越高。

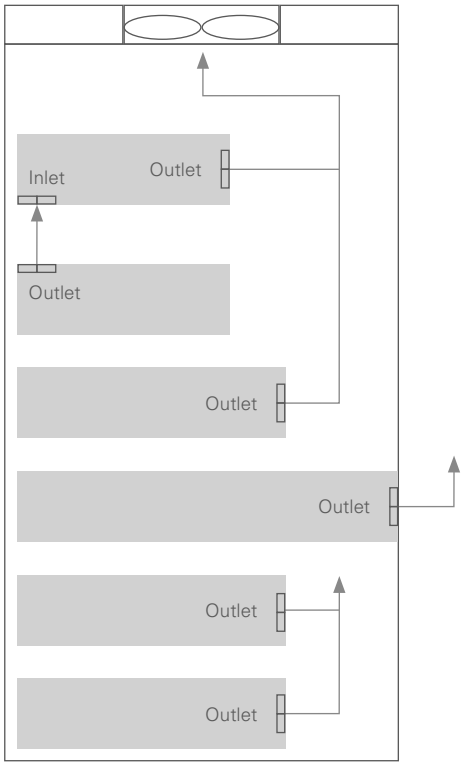


图 6 机架系统中仪器阻塞排气示例

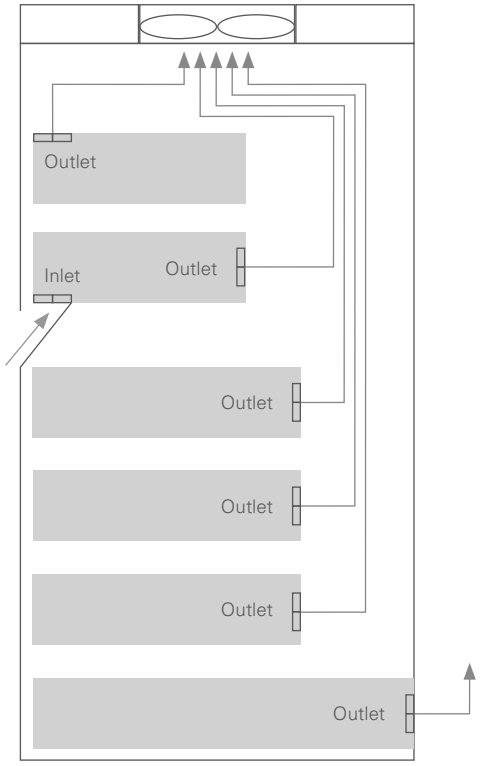


图 7 机架系统中正确排气布局示例

在这些情况下，理想的方法是提供一条从机架外部到仪器进气口的隔离路径。此方法可确保正确了解和控制仪器的环境温度。

请记住，仅仅考虑进气口温度是不够的。请确保根据仪器的规格为每台仪器提供足够的间隙，在仪器周围实现适当的绝缘和空气流通。为了节省空间，很多设计人员会忽略这些限制。然而，要想获得仪器规定的性能，必须遵循这些间隙规范。

从图中可以看出，仪器之间正在形成局部热气流。请记住，使用红色热量箭头只是为了表示该热量将被阻止进入进气口。请为设备进气口设计隔离路径，使空气可以在机架周围和上方流动。大多数机架组件会为所有仪器的侧面提供足够的间距，以确保可以形成适当的“烟囱效应”。仪器释放的热量应被允许在进气口隔离屏障周围流动。有很多方法可以确保在不影响仪器的前提下从机架中正确排出热量。以上所述只是其中几个例子。

热传递和气流

如前所述，大多数仪器的出气口温度与环境温度接近。

二者的差异源于自热和空气加热：

- **自热**—由于来自其他组件的热气以及自热效应，电子设备上的任何组件都会升温至环境温度以上。我们无法控制自热。
- **空气加热**—巧妙布局的机架系统可以最大限度地减少空气加热，而合适的机架冷却系统或室内冷却系统可以解决环境加热的问题。因此，在系统设计阶段可以控制此偏差。

在本例中，由于两个机箱在不同位置安装了不同的PXI卡，其自热和空气加热情况略有不同。

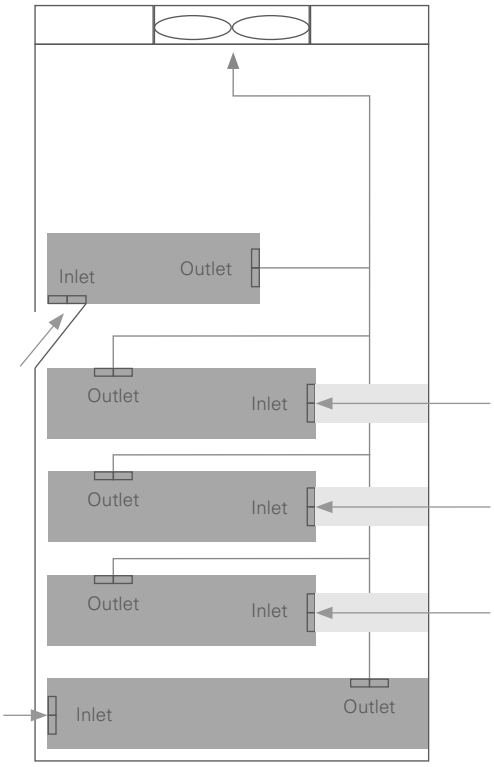


图 8 机架中的自定义进气口示例

被动冷却

被动冷却柜的目的是最大限度地发挥内部安装设备通过风扇进行自我冷却的能力。在此方法中，设备产生气流，机架的表面和通风口进行热量交换。

主动冷却

被动冷却仅依赖于设备风扇和热传递，而主动冷却柜需要使用额外的风扇和/或鼓风机进行战略性部署，以补充空气，增加散热。

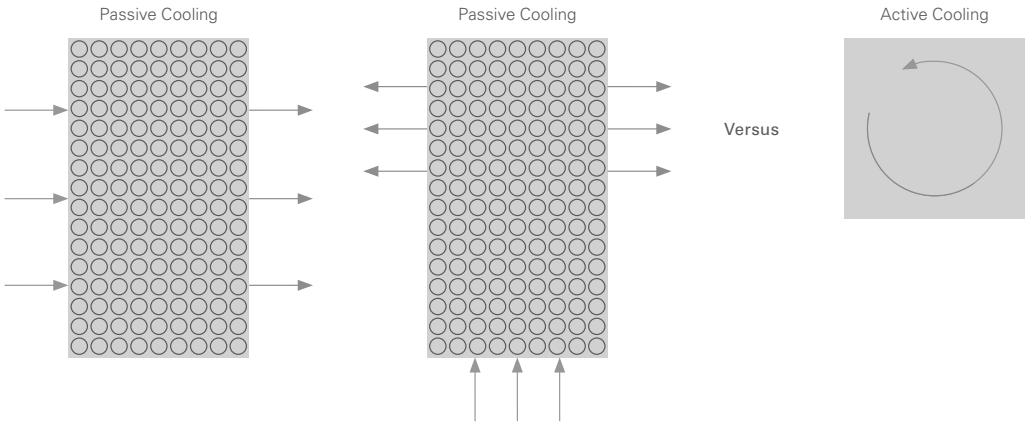


图 9 被动冷却依赖于内部仪器的风扇，而主动冷却则使用安装在机架中的辅助风扇和鼓风机

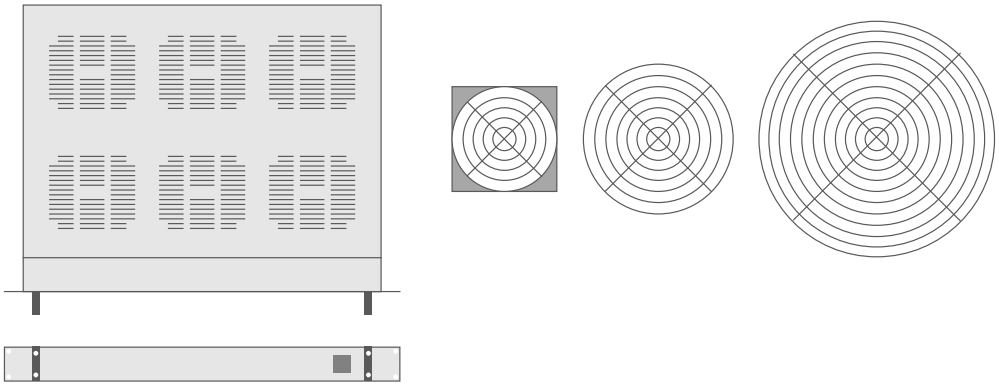


图 10 主动冷却选项包括内部托盘以及侧面或顶部安装式风扇等

通常情况下，机架式测试系统需要强制风冷。常见做法是机架顶部安装一个尺寸合适的排气扇；但是，通风口和气流需要根据排气扇的位置进行规划。如果空气路径发生弯曲、遇到障碍物、或者在某个区域集中了多个大功率仪器，则建议在机架中间放置风扇盘以协助通风。如果机架有需要冷却的热点，也可以考虑局部散热。为此，可以使用单独的风扇或风扇盘。

风扇容量

为了更好地了解合适的风扇大小，需要计算风扇每分钟能够移动的空气量，以立方英尺/分钟(CFM)为单位，同时还要考虑机架中所有设备的总功率瓦数和机架内外空气间的温差。要获得较高的CFM，可能需要在成本、振动和声学噪声方面做出一些妥协。此外，无论风扇的CFM有多高，机架都无法冷却到低于房间环境的温度。此方程的目的是在所有参数之间实现良好的平衡。空气阻力也会影响风扇的冷却能力。空气阻力会随着气流路径上物体横截面积的增大而增加，包括进气口的面积和气流路径上设备的面积。因此，风扇的CFM额定值应留有余量，以确保其能够克服空气阻力并仍提供必要的空气流动。

在评估总瓦数时，请避免使用设备的额定功率；额定功率通常是设备可能消耗的最大功率，但设备很少会达到这一功率。建议使用50%到60%的额定功率。更合理的方法是通过机架系统的PDU来计算实际功率并使用该值。

Delta T(ΔTc)等于需要从机架带走的热量。此数值取决于设备要求或仿真结果，可能根据机架内的位置不同而有所变化，通常在仪器出气口或机架顶部的数值较高，因此请务必了解适合您系统的Delta T，以及如何确认是否实现了该Delta T。

建模和验证

如果系统成本高昂，如组件交付周期长、系统是关键或战略应用程序的一部分，或者存在许多未知因素，则设计步骤中应包括建模。

机架设计建模

除了理论计算外，机架系统的建模和仿真还可以显著加快设计的优化，并能够提供有效的反馈来帮助您了解需要改进的领域。

请在软件中输入尽可能多的设计规范以获得最准确的建模。如果无法找到必要的规范，请评估类似的组件/仪器，并询问团队里的资深成员是否有使用这些设备的经验，以便获得估算值。未知事项越少，建模就越准确。

在对设计进行了详细的研究后，无论是通过评估仪器的温度要求、进行计算来优化气流和温度，还是对设计进行仿真，都相当于执行了真正的合格性测试。随后即可根据设计进行制造并验证其性能。

可以使用温度传感器或热成像仪来表征性能。应关注机架内的关键区域，例如机架的进气口和仪器的进气口。在收集整个机架的温度数据时，请为机架供电并以常规的方式运行仪器，因为待测产品通常是在此种条件下工作。这样可以最真实地了解温度情况。

有时可能还需要测试某些最坏情况下的负载条件(例如当热负载处于最高或最低值时)，以确保设计仍然可以适应这些条件。需考虑测试时间点、测试持续时间和测试条件，例如有多少操作人员，可能与测试站有哪些正常交互等，这些都是可能影响结果的因素。仔细分析结果，找出任何以前未识别的异常情况或可能需要解决的问题。

验证方法

首先，在设计初期使用标准图表、方程和仿真工具，以熟悉系统。随后，使用温度传感器或热成像仪执行系统特性分析，以验证设计并对其进行迭代，直到满足要求。最后，执行Gage R&R研究以验证稳定性和性能，确保站到站的性能与生产测试及验证和确认阶段的性能一致。

基于系统监测的可维护性

考虑实施支持实时评估的健康和监测系统，以确保系统仍能满足预期的性能。对系统性能的反馈可确保您至少在测试期间做出有据可依的决策，同时帮助您更好地了解测量数据并更好地预测系统的维护需求。

重点领域包括：

- 用于监测系统性能的独立系统
- 用于报告维护问题的系统看门狗
- 用于验证测试条件的测试反馈
- 用于评估和趋势分析的历史记录

规格验证

在推导规格时需了解假设条件。如果实际采集数据时的环境条件与推导假设条件不同，请务必考虑这一偏差。确保在所提供的温度条件下获得预期的性能。

限值计算

与规格验证类似，推导限值时应考虑温度差异和温度变化。如果测试环境的温度范围与产品规格中规定的温度范围不同，则应适当使用设备的温度系数来补偿差异。例如，NI开关模块的工作温度通常规定为0至55°C，但用于测试这些模块的环境一般是标准制造测试车间，温度通常维持在24±4°C。在确定测试限值时，应从规格和测量不确定度中减去最坏情况下的等效温度系数。

将设计标准应用于产品

在制定规格/限制时考虑热量因素

使用机架设计采集数据期间，在对制造测试限值进行规格验证时，请勿忽视热量的影响。如果预期结果和实际结果之间存在差异，原因可能是热量因素。

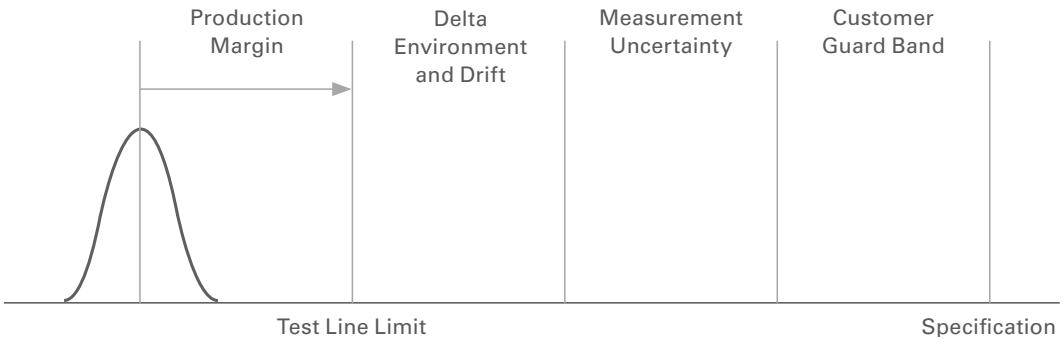


图 11 通用规格模型示例

独立系统监测

实时监测温度

实时温度监测可助您在测试期间做出灵活的决策。您可以在确定违反了某个操作要求后立即停止测试；或者出于稳定性考虑，在继续测试前先执行自校准或施加延迟。此外，当未来出现性能或测量结果相关问题时，历史数据可以提供有用的信息。

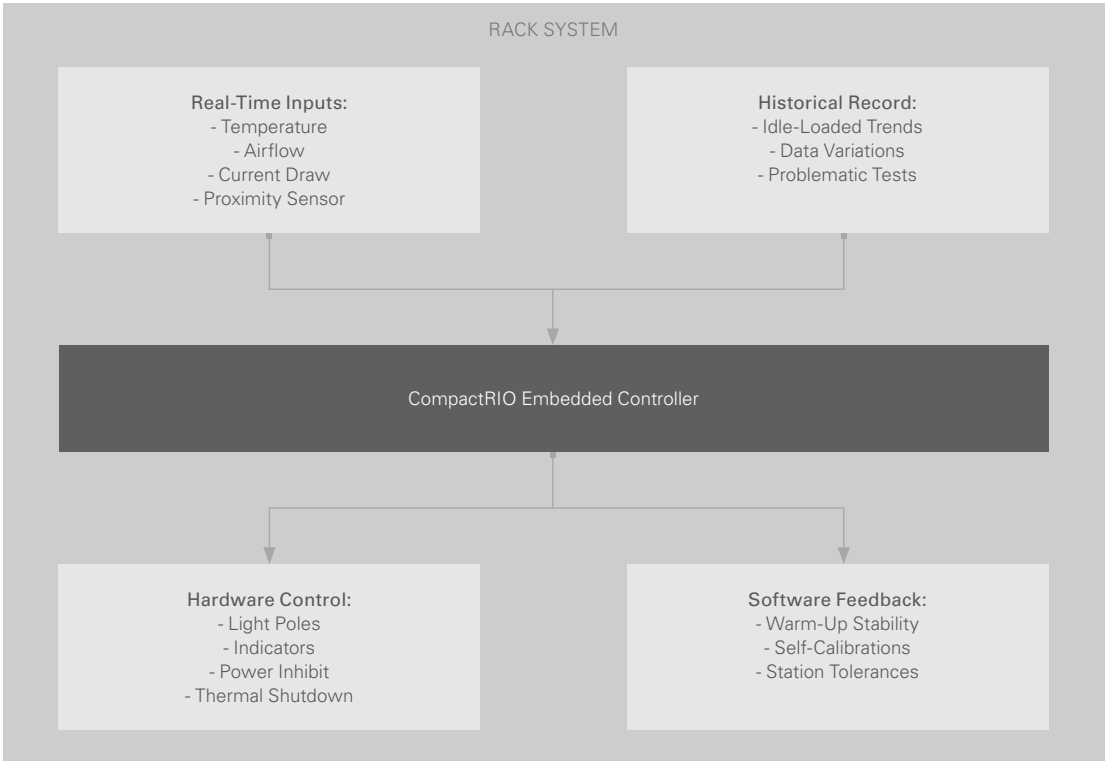


图 12 基于CompactRIO的独立监测系统示例

例如，可以使用嵌入式独立系统来监测和控制测试站设计的某些方面。您可以监测温度并在测试执行期间根据相关数据做出决策，也可监测和管理测试站资源以支持并行测试。一般来说，独立监测系统可以用于多个任务，这不仅有助于机架系统的设计和验证，而且也有利于部署和长期使用。

此方法应用十分广泛，主要包括：

- 监测
  - 整个机架的环境温度
  - 仪器的气流、电流消耗和内部温度
  - 维护所需的仪器健康数据
  - 机架门，使用接近传感器检测系统是否开启
  - 温度状态，以便实行热停机机制来保护系统
- 获取数据以在测试应用中做出实时决策
- 记录历史数据以供将来分析
- 通过灯杆、指示灯或显示器向测试站用户提供测试站的状态反馈，并报告任何超出容差的情况。

健康和监测系统可以帮助您实时评估系统，在测试期间做出有据可依的决策，更准确地了解测量数据和预测系统的维护活动。

附加信息

NI联盟伙伴网络

NI联盟伙伴网络项目囊括了全球950多家独立的第三方公司，旨在为工程师提供基于图形化系统设计的完整解决方案和高品质产品。从产品和系统到集成、咨询和培训服务，NI联盟伙伴均可通过其独有的产品和技术来帮助用户应对当前一些最为严峻的工程挑战。

查找联盟伙伴

NI PXI机箱冷却

NI机箱经专门设计和验证，能够满足甚至超越大多数高功率PXI模块的冷却要求。NI设计的机箱远远高于PXI和PXI Express要求，可分别为PXI和PXI Express机箱的每个外设插槽提供30 W和38.25 W的供电和冷却功率。这种高标准的供电和冷却有助于高性能模块(如数字化仪、高速数字I/O和RF模块等)在需要连续采集数据或高速测试的应用中发挥高级功能。

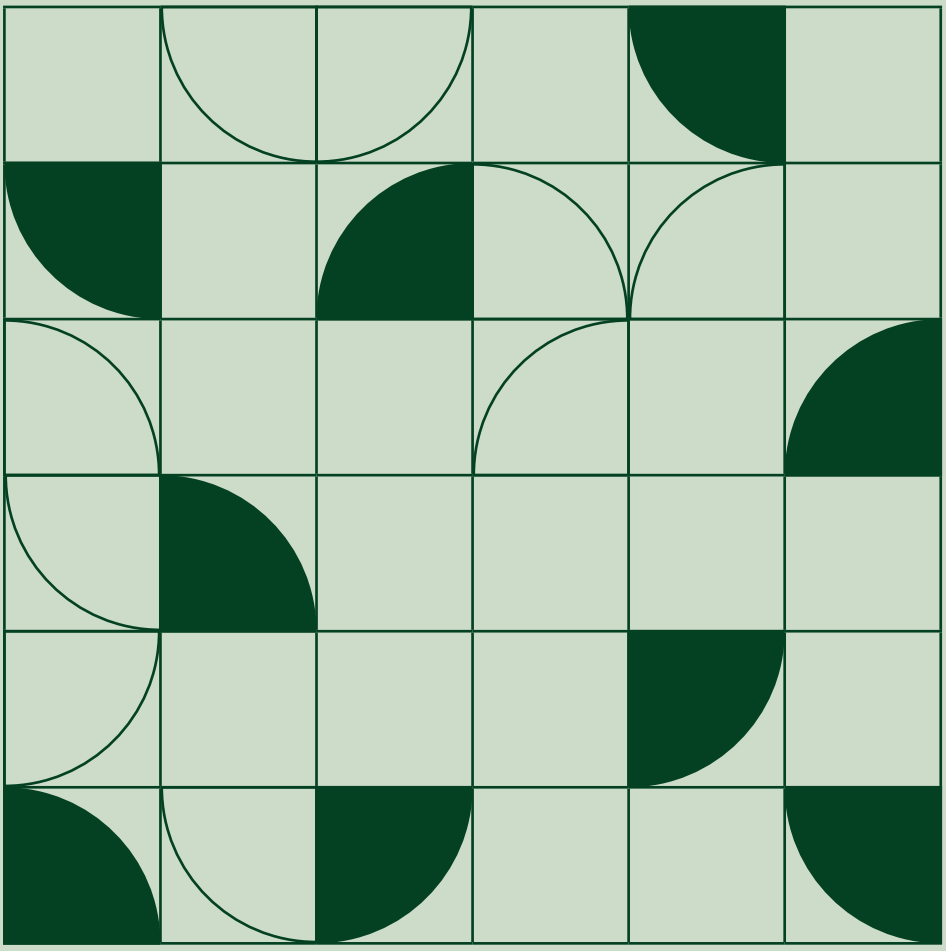
进一步了解NI PXI机箱的设计优势

立即构建您的专属PXI测试系统

NI是PXI的缔造者和领先供应商，这一模块化仪器标准已经拥有70多家供应商，提供超过1,500种产品。您可以为您的应用选择合适的机箱、控制器和模块，并根据在线配置指南的建议选择系统所需的组件和附件。

配置您的专属PXI系统

大规模互连系统和连接件



引言

在搭建测试系统时，如果未考虑如何将仪器与待测设备(DUT)相连，就像是驾驶没有车轮的汽车。这样的汽车或许配备了马力强劲的发动机和意大利真皮座椅，但没有车轮，就无法到达目的地。在自动化测试系统中，大规模互连系统和测试连接件就相当于汽车的车轮。在确定所需的仪器、开关数量以及开关在测试系统中的安装位置后，下一步就是要选择合适的大规模互连系统，并设计适当的连接件，将DUT无缝连接到系统的其余部分。

大规模互连系统概述

大规模互连系统是一种机械互连系统，专为轻松连接接收自或发送至DUT或DUT连接件的大量信号而设计。大规模互连系统不是逐一连接每个信号，而是可以同时连接和断开所有信号。对于自动化测试系统，大规模互连系统通常需要配备一些可互换的机械外壳，所有信号均通过这些外壳从仪器(通常位于测试机架中)连接到DUT，从而方便用户快速更换DUT，或保护仪器前端的电缆连接，避免重复连接和电路断路。

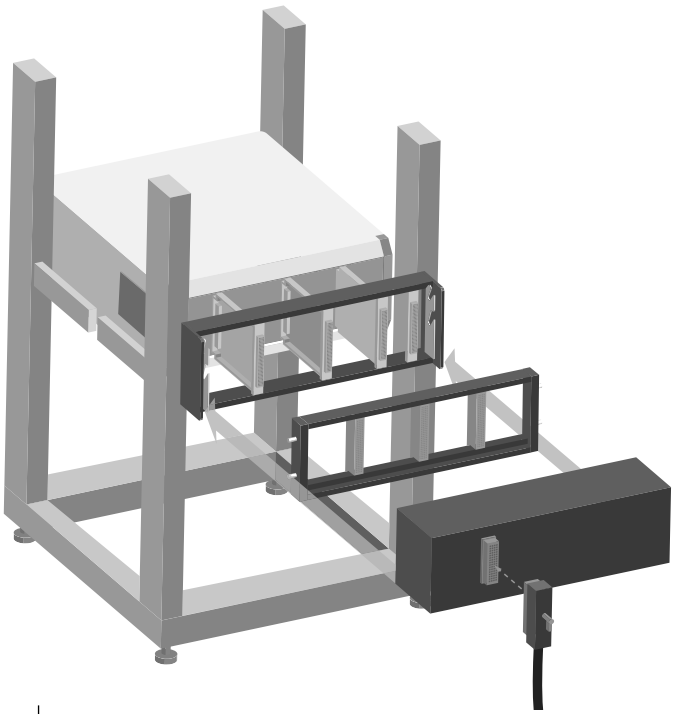


图1 使用CompactRIO的独立式监控系统示例

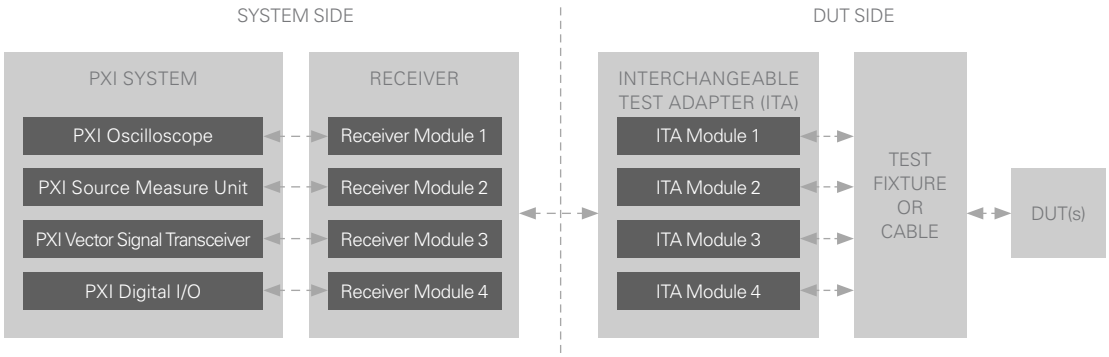


图2 大规模互连系统可以简化为两部分。其中一部分是系统端组件(通常称为接收器)，用于将仪器连接到大规模互连系统，充当可互换测试适配器(ITA)的“插座”。另一部分是DUT端组件(通常称为ITA)，用于将DUT连接到大规模互连系统，充当接收器的“插头”。接收器和ITA在单个机械操作中相互配合，为各种DUT重复使用一组通用测试硬件提供了一种简单的方法

系统端组件

系统端组件包括仪器和大规模互连系统之间的所有组件。系统端组件是测试系统通用组件的一部分，即使更换ITA和连接各种DUT的连接件，系统端组件仍保留在测试系统中。下面简要说明各个系统端组件。

接收器

接收器是大规模互连系统中测试系统端的核心组件。它提供的机制能够将多个仪器同时连接到DUT。接收器系统包括框架、安装硬件、接收器模块以及将接收器模块与仪器相连的连接系统。

安装硬件

安装硬件用于将接收器固定在机架或PXI机箱的正面。这些硬件通常安装在19英寸机架的正面，便于测试人员进行操作。有些安装硬件带有铰链，或安装在滑动杆上，便于测试人员接触接收器后面的仪器或电缆。

接收器模块

接收器模块安装在接收器中，便于在一次机械操作中与所有其他接收器模块共同完成从仪器到接收器主连接器(一类标准连接器，可连接到大规模互连系统的DUT端)的所有适当连接。这些连接是从仪器和其他辅助设备到接收器模块内的相应触点的连接，通常根据所通过的信号的密度、带宽、电流或其它特殊要求来确定。

如需将仪器和其他辅助设备连接到接收器模块，可以单独使用或组合使用以下两种方法：

- **电缆组件**—使用电缆组件，可以将标准或定制电缆从仪器直接连接到接收器中的触点。电缆组件会使安装和接收器模块的放置更为灵活，但接收器模块和仪器之间的信号路径通常更长(24英寸或更长)，如果管理不当，可能会影响性能。
- **接口适配器**—接口适配器通常用于将仪器连接器(例如DIN、D-SUB、SCSI等)的所有I/O连接到接收器模块。适配器始终与仪器直接相连，并使用印刷电路板(PCB)、柔性电路或电缆为特定仪器提供最有效的连接方法。接口适配器的优势在于，可最大程度地缩短接收器模块和仪器之间的信号路径(通常为6英寸)，且信号性能可变，但接口适配器灵活性较差，因此需要进行更周密的前期规划，安装位置必须精确，不能灵活更改。

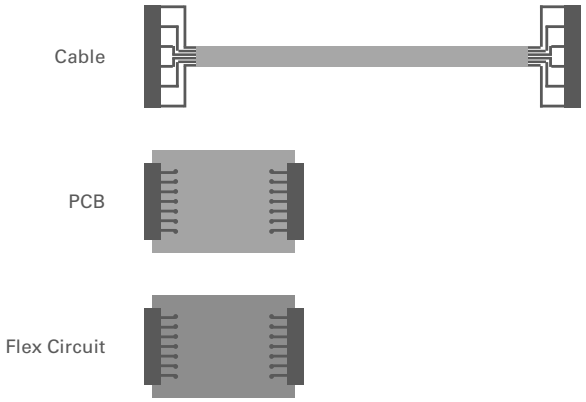


图 3 电缆组件(上图)会使安装和接收器模块的放置更为灵活，但通常对应的信号路径更长。PCB(中图)和柔性电路(下图)可最大限度地缩短信号路径，保持信号质量，但灵活性低于电缆组件。

低 ○ 平均 ◒ 高 ●

部分特性	电缆	采用线缆的大规模互连	采用PCB或柔性电路的大规模互连
DUT之间频繁切换	○	●	●
针对设计和特性分析进行优化	●	○	○
针对验证和确认(V&V)进行优化	◒	◒	◒
针对试生产进行优化	○	●	●
信号质量	◒	◒	●
性能连续性(系统到系统)	◒	◒	●
系统易于维护与升级	○	●	●
系统重配置(即可扩展性)	○	●	●
易于复制(例如，全球部署)	○	◒	●
仪器到模块的引脚效率	○	●	◒
现场可维修性	●	●	○
仪器卡版本控制容差	●	●	○

表 1 电缆适合设备的设计和特性分析，但大规模互连非常适合生产测试环境

DUT端组件

DUT端组件包括大规模互连系统与连接件或DUT之间的所有组件。DUT端组件组装在同一个装置中(通常称为ITA)，并且可以轻松互换，便于使用一组通用仪器对不同的DUT进行测试。下面简要说明各个DUT端组件。

ITA

ITA是大规模互连系统中DUT端的核心组件，包含外壳或机械框架以及与接收器配对的ITA模块和触点，用于将系统输入和输出传输到DUT。如果接收器是插座，那么ITA就是插头。许多测试系统的设计都支持在不更换测试系统和接收器的情况下，通过更换不同的ITA来测试各种DUT。

ITA模块

ITA模块安装在ITA内部，安装方式与接收器模块在接收器内部的安装方式大致相同。它们为主互连系统提供了经由接收器传输的各种信号，并通过电缆、PCB或ITA外壳内的其他连接来实现这些连接。所选的ITA模块和触点要与之前指定的接收器模块和触点相匹配。然后，才能将适当的信号从外壳内部传输到连接件或DUT连接器。

外壳

外壳是容纳ITA和相应ITA电缆/模块的机械外壳。通常会将ITA外壳和测试连接件集成到单个框架或物理平台上，并在其中放置DUT(例如在消费电子或半导体测试中)；或者用一条电缆将ITA和DUT相连。虽然可以选择标准外壳，但实际上几乎每个外壳都进行了一定程度的定制，以满足DUT的要求。

测试连接件

每个DUT都不相同，因而需要采用独特的连接方法才能实现最高效的测试。例如，有些DUT适合用一根电缆将ITA和DUT相连，而其他DUT则更适合采用集成式测试连接件(例如针床式测试连接件)，从而实现无需电缆的直接连接方法。

如何选择适合的大规模互连系统

对大规模互连方案进行前期规划和设计，可以确保测试系统能够发挥其全部潜力，从而充分利用所选仪器和仿真的所有功能，满足数据采集要求。然而，在决策时，性能只是需要考虑的一个方面，还应对比不同方案的成本。在计算相关的总体成本时，还应考虑设计验证程序、定制组件的生命周期管理和文档记录相关成本，以及在自动化测试设备和连接件预期寿命中的总体维护成本。在前期可以与大规模互连解决方案供应商合作，节省大量时间和成本。

您可以借助以下常规步骤确定哪些大规模互连组件适用于您的系统，但是您还应咨询大规模互连专家和NI联盟合作伙伴(例如MAC Panel或Virginia Panel Corporation (VPC))，获得决策相关指导。除提供建议外，有些大规模互连供应商还提供针对特定PXI资产和通用仪器预先设计且经过配置的解决方案，减少配置和记录接口所需的时间和精力。您只需提供系统所需的仪器和其他资产的列表，他们就可以提供相应的已配置部件编号列表，其中包含各种大规模互连(接口)和配套(连接件)组件。

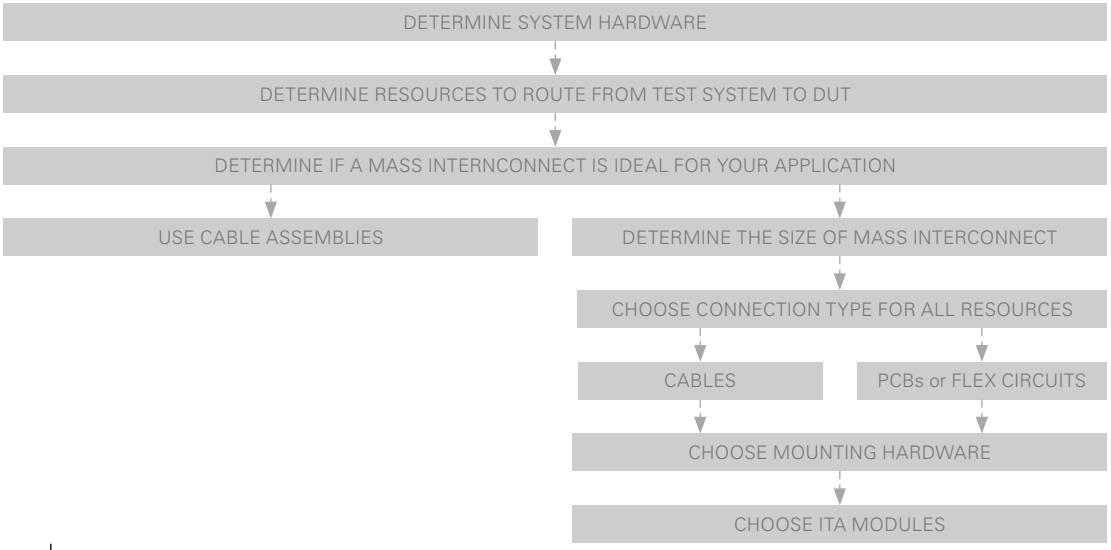


图 4 大规模互连系统的考量因素

01

确定系统硬件。

首先，确定测试DUT所需的系统硬件，其中包括仪器、开关、需要连接到DUT的任何辅助硬件(例如电源)以及测试机架的尺寸和样式。

02

确定测试系统中的哪些资源需要连接到DUT。

根据测试系统和被测设备的复杂程度，您很可能会需要通过大规模互连系统连接所有必需的仪器和辅助组件。有些测试组件不需要通过大规模互连系统进行连接，例如PXI嵌入式控制器或RAID存储系统。

在确定哪些资源需要通过大规模互连系统进行连接时，必须考虑测试需求在未来可能发生的变化，以便构建一个灵活的测试系统来满足未来的需求。例如，如果您拥有额外的仪器或电源通道等额外的资源，但当前DUT测试可能并不需要，而未来几代的DUT测试却有可能需要，那么可以根据未来预计会发生的变化规划好这些资源的连接，从而节省时间和成本。

03

确定最适合应用的连接类型—电缆或大规模互连系统。

选择连接类型时应综合考虑多种因素，包括系统的复杂性、技术性能要求、灵活性和总体拥有成本。决策时请使用表1作为指导，同时也需要咨询大规模互连专家，以确保为特定测试系统选择适合的连接类型。假设您决定使用大规模互连系统，可参考以下步骤确定接收器和ITA的尺寸，以及选择适合的接收器模块、ITA模块和安装硬件。

04

确定接收器和ITA的尺寸。

接收器有许多不同的尺寸和样式可供选择，具体取决于前面几个问题的答案。最好保留接收器中的备用插槽，以备未来扩展时使用。类似于PXI系统仪器的规划，应遵循一般的指导原则，即初期设计时至少保留20%的插槽不被占用。请注意，ITA的尺寸应始终与接收器的尺寸相匹配。

05

选择接收器模块和连接方法(电缆或接口适配器)，以满足所有必要资源的连接需求。

选择接收器模块和连接方法时，应综合考虑测试目标、所选仪器和任何其他辅助要求。同样，您可以使用表1作为指导，但请务必咨询大规模互连专家，以确保为特定的测试系统选择适合的连接器模块和连接方法。

您可以从多种触点样式和预先设计的接口适配器、跳线或电缆中进行选择，以满足所有信号类型的要求，其中包括：

- 低频AC信号
- 电源信号
- RF信号
- 微波信号
- 热电偶信号
- 光纤信号
- 气动信号
- 高速信号和数据传输

06

选择安装硬件。

在选择接收器模块和连接方法后，接下来要选择安装硬件，这需要综合考虑多种因素。大多数使用大规模互连的系统都会安装到19英寸的机架组件上。如果采用电缆系统，建议将接收器安装在铰接框架或滑动杆上，以便于接触仪器和机箱。如果使用接口适配器，则建议使用标准安装法兰将接收器安装到PXI机箱上，然后将接收器和机箱安装到机架内的滑架上。

07

选择匹配的ITA模块和触点。

配置ITA模块和触点，使其与在步骤五中所做的选择相匹配。ITA模块中的触点可能不需要完全填充也可完全匹配接收器模块中的触点。在大多数情况下，来自特定仪器的所有资源都会传递到接收器模块。然而，在测试特定DUT时，ITA端并不需要使用所有这些资源；因此，并非所有ITA模块都需要完全填充触点。

测试连接件概述

测试连接件是一种在测试系统和DUT之间提供可重复连接的装置，通常采用定制设计来满足特定DUT的需要。在连接件背面使用大规模互连系统时，可以在使用通用仪器机架的情况下互换各种测试连接件；还可以通过轻松互换不同DUT的测试连接件来重复利用测试设备，这是通用测试仪或高混合测试仪的理想选择。

在设计测试连接件时，请提前了解您要执行的测试类型，因为设备设计、DUT特性分析、验证和确认(V&V)以及生产测试的需求是完全不同的，因此需要的测试连接件功能也各不相同。

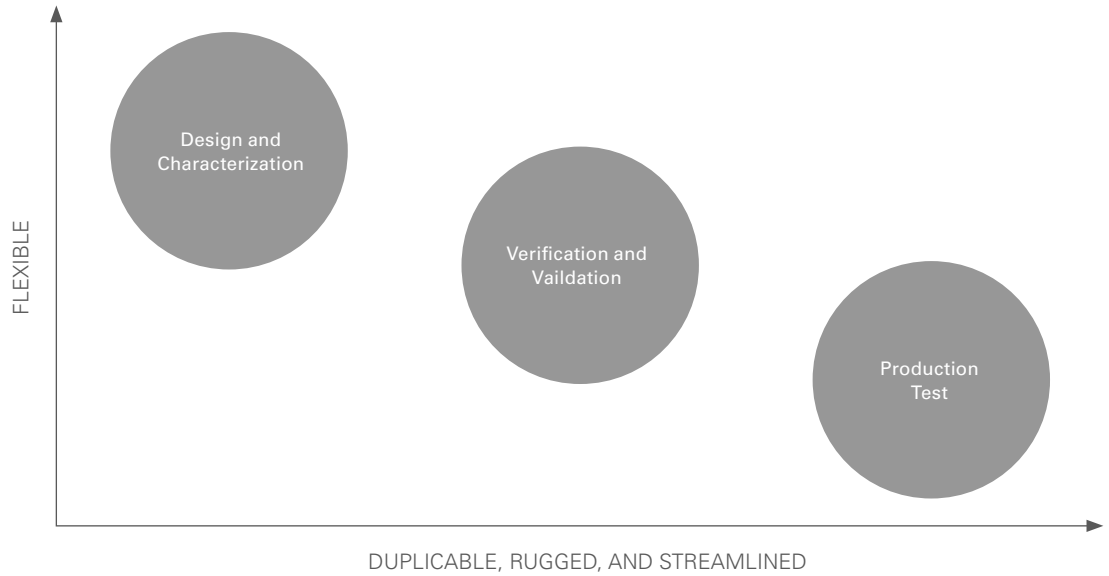


图5 生产测试系统需要采用坚固耐用、易用且易于复制的测试连接件，而仪器系统设计和特性分析需要采用电缆、探头和开尔文夹具来保障灵活性。

设计和特性分析

在产品设计阶段，我们必须能够灵活地连接或断开DUT到测量仪器之间的电缆或针脚，以便对设计中的所有元件进行测试或排除故障。因此，需要采用一种高度灵活的方式连接仪器，而不采用包含大规模互连系统和连接件的简单或坚固耐用的测试系统。许多设计工程师采用电缆、探针和开尔文夹具连接仪器，这样可轻松地更改连接。除了故障排除外，设计工程师还经常需要对设备的实际行为进行特性分析或记述，以便确定设备的规格参数，为测试工程师开发V&V测试台或生产测试系统提供指导。

验证和确认

V&V是通过对具有统计代表性的一组产品进行测试来检查特定产品是否满足设计规格的过程。

验证

验证是一个客观过程，用于检查设备是否满足相应法规和规范。验证通常在设计和开发阶段以及开发阶段完成后执行。在开发阶段，验证测试有时需要模拟系统其余部分的行为或对其进行建模，以预测或预先了解所开发设备的行为。在开发完成后，验证测试可以采用回归测试形式执行，从而重复执行所设计的多项测试，确保随着时间的推移，设备能够持续满足设计要求。

确认

确认是一个比较主观的过程，通过回顾形成新产品需求的问题描述，主观评估设备是否满足最终用户的操作需求。确认测试的要求来自于用户需求、规格和/或行业法规。在确认规范时，我们的目标是确保规范能够涵盖用户需求，而不是确保给定设备满足其规范要求。

在V&V过程中收集的测试数据也可用于确定生产测试系统的测试限制。虽然只有一小部分给定产品需要进行V&V，但几乎所有最终产品都需要通过生产测试。因此，专为V&V过程设计的连接件通常有助于减少DUT和仪器之间的连接，使得这些连接件不如生产测试所用的连接件坚固。

生产测试

生产或功能测试通常在制造过程结束时进行，目的是测试产品是否满足已发布的规格和质量标准。很多时候，为优化吞吐量并减少人机交互导致的错误，功能测试可采用自动化方式执行。有时，生产测试包括对产品的实际工作环境进行仿真或模拟。最重要的是，功能测试使用的是客户最终使用的连接器，而不是PCB上的各种测试点。

由于制造和发运的每台设备都必须经过生产测试，连接件的设计必须坚固耐用，以最大程度地延长正常运行时间，并且必须易于使用和符合人体工程学。同时，还应该尽量减少测试操作员所需的交互。将DUT对准连接件或使用电缆将仪器连接到连接件会花费一些额外时间，因此会对吞吐量产生负面影响并增加测试成本，同时还会增大由于人机交互而导致错误的风险。最后，生产测试系统(包括测试连接件)应易于复制以便用于其他场合。

连接件考量因素

在设计测试连接件时，应确保连接件使用适合的接线类型和技术，尽可能使用PCB而不是电缆，并尽可能多地实现自动化连接。此外，还需要为测试连接件制定预防性维护计划，以确保长期成功应用。

选用适当的线缆

电线通常会产生噪声和误差，因此应该为测试连接件选择最佳类型的电线。为了确保信号完整性，有些电线具有绝缘、屏蔽、防护或双绞线等特性。有些仪器手册建议使用专用电缆，但这通常取决于所执行的测量类型。例如，双绞线非常适合在执行差分测量时抑制噪声。使用屏蔽电缆也能抑制噪声，但务必要根据信号源和输入配置的接地情况采用适合的接地方案。最后，防护特性通常用于消除数字万用表(DMM)或源测量单元(SMU)的HI和LO端子之间漏电流和寄生电容的影响。

最大程度减少操作员交互

大规模互连系统的目标是提高测试设备的复用性，并减少可能导致错误、降低吞吐量和增加测试总成本的操作员交互。除了使用大规模互连系统来减少用户交互之外，良好的测试连接件应能最大程度地增加测试操作者通过单次交互在测试连接件和DUT之间实现的连接数量。例如，有些测试连接件会利用由单个操作手柄驱动的连接方法或者利用电动或气动马达自动地完成多个连接。执行生产测试时，通常使用客户最终用于连接设备的连接器来进行连接。

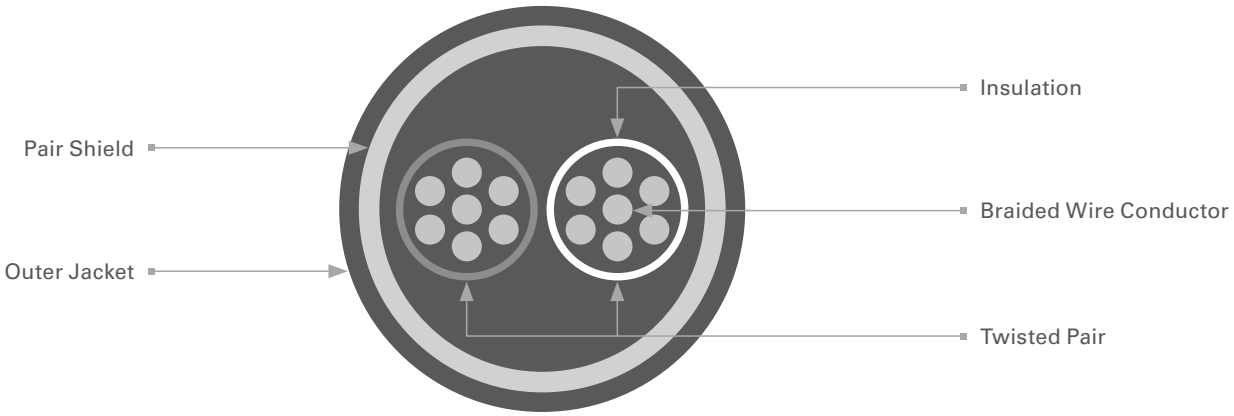


图6 为了保持信号完整性，不同的电缆具有绝缘、屏蔽、防护或双绞线等特性。

构建可复制且易于扩展的连接件

某些接线技术可以保持信号完整性，但建议在测试连接件中使用PCB来提高信号完整性，同时减少测试操作员或技术人员在安装阶段的接线工作。使用可一次性完成多个连接的测试连接件有助于提高吞吐量，实现测试可重复性和满足用户人体工程学要求，但同时必须减少测试连接件内部的接线量并且在可能的情况下用PCB予以替换，从而进一步提高信号完整性，并缩短相同测试系统的安装和接线所需的时间。这要付出一定的代价，即会增加定制PCB设计所需的前期成本和工作量，但这些付出将在第一个系统以及随后复制的测试系统部署后获得回报。

为测试连接件制定预防性维护计划

为了确保测试连接件对测试系统的长期支持，在整个测试系统维护计划中应包括测试连接件维护计划。这应包括在测试系统的整个生命周期内定期检查和/或更换连接器、电缆、弹簧式探针、继电器和其他组件。在确定检查频率时，需要考虑给定组件的故障率。判断故障率时，请参考供应商提供的平均故障间隔时间(MTBF)或公司特定的在役故障率。虽然这两个指标在比较或分析仪器时都很有用，但在役故障率可能更有用，因为它表明了仪器在特定应用中的实际使用情况。如果无在役故障率数据，可通过MTBF数据导出理论故障率，从而规划与成本和风险承受能力相符的检查间隔。

附加信息

NI PXI配置指南

PXI配置指南可帮助您比较仪器选项并配置基于PXI的测试系统，包括PXI机箱、控制器、模块、软件、服务和辅助项目。

使用PXI配置指南配置您的测试系统

MAC Panel

MAC Panel的解决方案可在测试或测量环境中实现可靠、经济高效的电气连接，从而为具有此类需求的公司提供支持。除了全系列大规模互连产品(包括专为PXI设计的MAC Panel SCOUT大规模互连系统)，MAC Panel还提供定制布线服务、钣金加工和定制设计支持服务，为支持全球范围内的自动化测试设备提供了一系列选择。

了解MAC Panel的更多详情

Virginia Panel Corporation

Virginia Panel Corporation (VPC)是一家通过ISO认证的大规模互连解决方案制造商。VPC拥有150多名员工，能够为测试和测量行业设计和制造各种大型或小型I/O连接器和接口。除了提供面向PXI平台的大规模互连解决方案外，VPC还提供多种增值服务，如高速PCB设计、预配置测试解决方案、定制设计支持服务、产品支持文件的网络访问和在线配置工具。

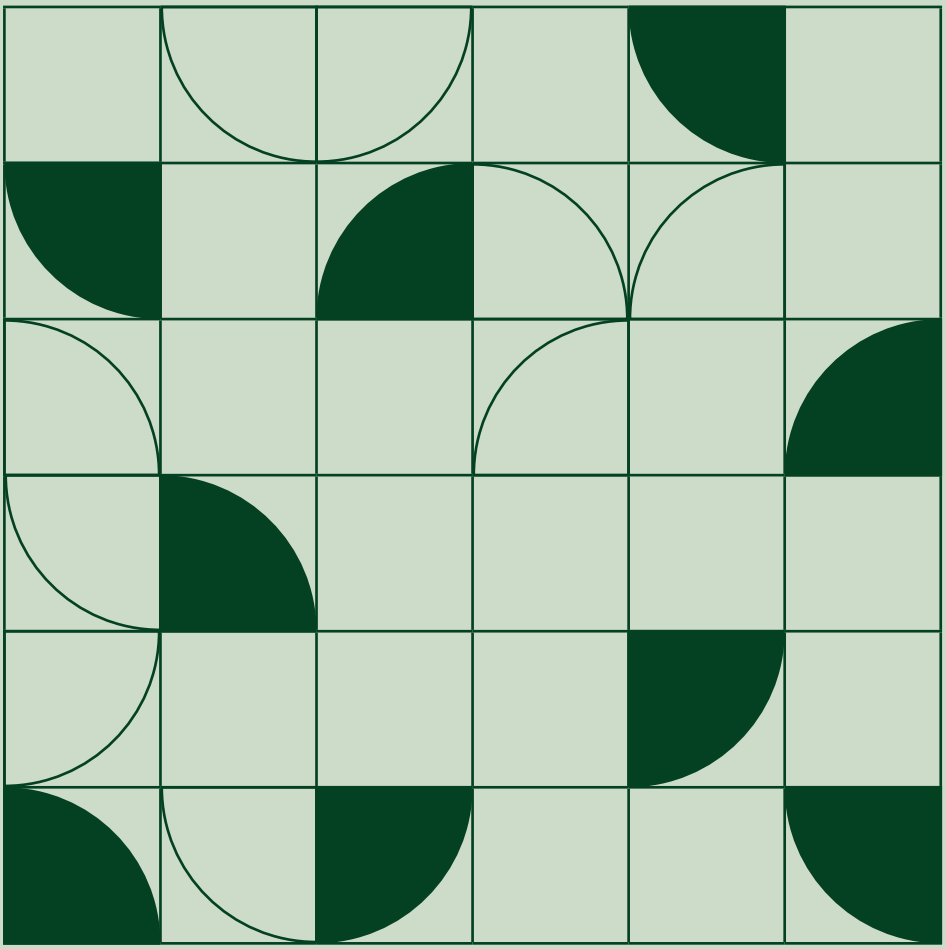
了解VPC的更多详情

NI联盟合作伙伴目录

NI联盟伙伴网络是一个由全球1000多家独立的第三方公司参与的项目，他们可提供完整的产品以及集成、咨询和培训服务。其中一些公司，如MAC Panel和VPC，提供定制布线解决方案、大规模互连系统和完整的连接件解决方案。

浏览NI联盟合作伙伴目录

软件部署



简介

由于设备日益复杂化，测试工程师需要创建更复杂、更高级的混合测试系统，而且通常面临着时间期限紧迫和预算缩减等挑战。构建测试系统的一个重要步骤是将测试系统软件部署到目标机器上。这也是最繁琐、最令人沮丧的一个步骤。对于那些只是单纯寻求最便宜和最快速解决方案的工程师，市场上琳琅满目的部署方法往往会增加他们的选择难度。此外，测试系统开发人员也面临着许多其系统特有的考量因素和敏感问题。

在本指南中，部署定义为编译一系列软件组件并将这些组件从开发计算机导入到目标机器上执行的过程。测试工程师之所以采用部署方法，而不是直接在开发环境中运行测试系统软件，主要是出于成本、性能、可移植性和保护方面的考虑。

以下是测试工程师需要从开发环境执行转换到定制的二进制部署的一些常见拐点：

- 每个测试系统的应用软件开发许可证成本开始超过预算限制。为每个系统使用部署许可证是更具吸引力、更高效的解决方案。
- 由于内存限制或依赖关系等原因，测试系统的源代码变得难以移植。
- 测试系统开发人员并不希望最终用户能够编辑或者查看系统的源代码。
- 测试系统在开发环境中运行时，会遇到执行速度缓慢或内存管理问题。编译执行代码可提高性能并减少内存消耗。

本指南提供并比较了不同的考量因素和工具，以解决测试系统部署过程中的疑难和困惑。尽管本指南可以介绍有关测试系统部署的许多不同主题，比如源代码控制最佳实践或安装程序的创建，但所选主题应涵盖大多数通用部署问题。

每部分的末尾均提供了基本用例和高级用例的最佳实践建议：

- 基本用例是由一个可执行文件构成的简单测试系统，会按顺序运行测试步骤并调用少量硬件驱动程序。此类系统包含的测试功能通常少于 200 种。
- 在每个基本用例最佳实践的末尾，都会说明出现何种迹象或指示时应考虑高级用例。

高级用例表示一个大规模生产测试系统，该系统将可执行文件、模块、驱动程序、Web服务或第三方应用相结合，用于执行混有大量不同测试序列的测试。此类系统通常包含数百甚至数千种测试功能。

管理和确定系统组件

定义组件

在软件开发中，组件是系统中使用的任何信息实体，比如二进制可执行文件、数据库表、文档、库或驱动程序。要成功完成部署，必须先确定与测试系统相关的组件并确保每个组件都采用适当的部署方法。此步骤在不同场景下的复杂性可能有很大差异。例如，简单测试系统的组件可能是单个可执行文件和必要的硬件驱动程序。

复杂系统组件

但在复杂的测试系统中，这些组件通常是XML配置文件、数据库表、自述文本文件或Web服务。随着系统复杂性的提高，需要引入更高级的部署选项。例如，可能需要频繁更新配置文件，以便根据天气的季节性变化校准获取的数据，而主要的可执行文件几乎无需更新。没有必要在每次出现更新需求时都将可执行文件与配置文件一起重新部署，因此，配置文件可单独采用与可执行文件不同的部署方法。

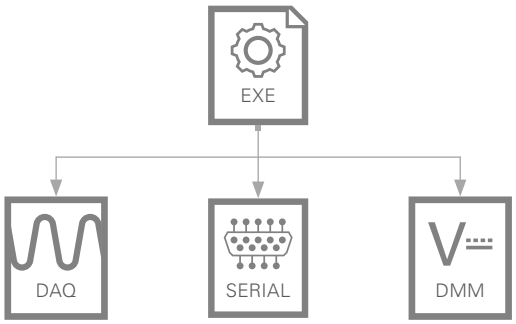


图1 基于DAQ、串口和DMM驱动程序的简单测试系统可执行文件示例

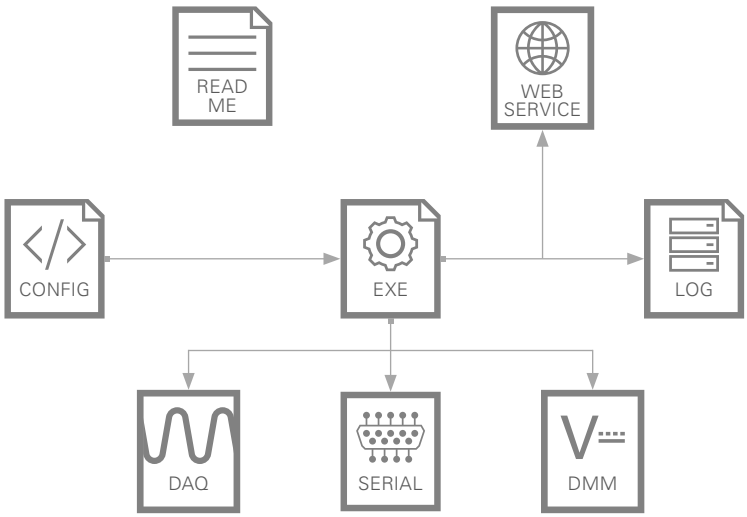


图2 具有复杂依赖关系的测试系统示例

除了确定每个系统组件并规划其部署方法外，确定系统组件之间的关系并确保部署方法不会中断这些关系也十分重要。在需要频繁更新配置文件的示例中，工程师可能需要将配置文件安装到每个部署系统上的相同位置，以便可执行文件可以在运行时找到配置文件。

依赖关系跟踪

维护各个依赖项之间的关系涉及建立依赖关系跟踪方案，此举可确保每个组件的依赖组件均得以部署。尽管在手动确定每个系统组件后依赖关系看似明显，但依赖关系通常会深度嵌套，并需要随着系统扩展而自动识别。例如，系统B中的可执行文件可能依赖于.dll才能正确执行，但创建部署计划的工程师却忘记将.dll文件标识为必要组件或没有意识到该依赖关系。在这些情况下，可使用编译工具自动识别已生成应用的大部分或全部依赖关系。

下面举例介绍了一些编译软件应用：

- **LabVIEW Application Builder**—识别特定上层VI集的依赖项(子VI)，并将这些子VI包含在编译的应用程序中
- **TestStand Deployment Utility (TSDU)**—以TestStand工作区文件或路径作为输入，确定系统的依赖项代码模块；自动编译这些模块并将其包含在编译的安装程序中
- **ClickOnce**—Microsoft的一项技术，可帮助开发人员为其.NET应用轻松创建安装程序、应用程序甚至是Web服务；可配置为在安装程序中包含依赖项，或提示用户在部署后安装依赖项
- **JarAnalyzer**—用于管理Java应用程序间依赖关系的实用程序，可遍历整个目录、解析目录中的每个jar文件并确定各文件之间的依赖关系

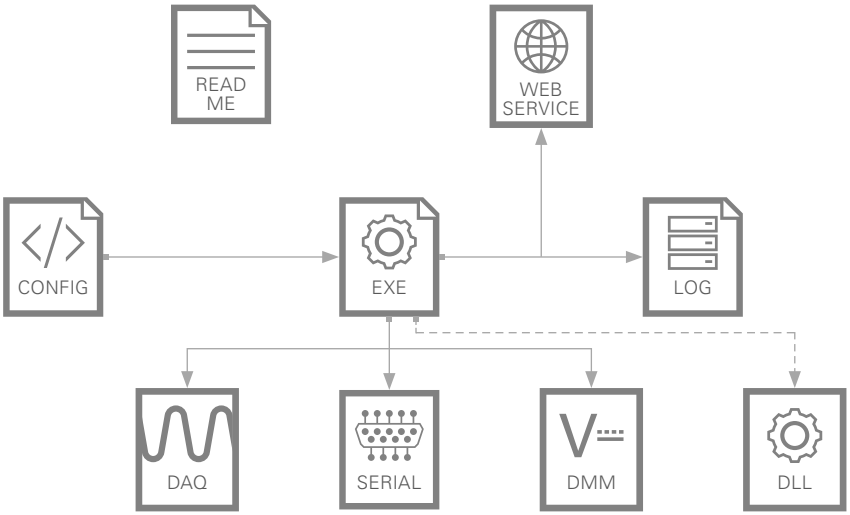


图 3 复杂测试系统中出现意外的依赖关系

关系管理

通常来说，不仅主测试程序可执行文件与其相关组件之间存在关系，各个单独的组件之间也存在关系。因此，需要考虑不同组件或软件模块之间关系的本质。随着系统不断扩展，解析不同库、驱动程序或文件之间的依赖关系可能变得非常复杂。例如，一个测试系统可能使用三个不同的代码库，三者之间的关系如下图所示。

对于这些复杂的系统，通常需要使用依赖关系解析器来识别依赖项冲突并管理无法解决的问题。尽管可以在内部编写依赖关系解析器，但工程师经常使用程序包管理器来管理依赖关系。例如，NuGet就是专为.NET框架程序包设计的免费开源程序包管理器。另一个例子是用于LabVIEW软件的VI Package Manager，通过该管理器，用户能够分发代码库并通过API提供自定义代码库管理工具。

最佳实践

基本用例：对于基本或简单的系统，通常可以手动跟踪所有必要的组件。使用软件应用程序或程序包管理器来管理依赖关系可能并无必要，且这种做法的前期安装成本过高。然而，如果出现警告提示，比如不断遇到依赖项缺失问题或依赖项不断增多，通常表示需要采用更高级的依赖关系管理方案。

高级用例：采用可扩展的依赖关系管理系统时，复杂的系统更容易维护和升级。这可能意味着需要使用程序包管理器来诊断程序包之间的关系，或使用软件应用程序来了解和识别各组件的依赖关系，无论是何种情况，维护此类系统对于系统的长期正常运行都至关重要。

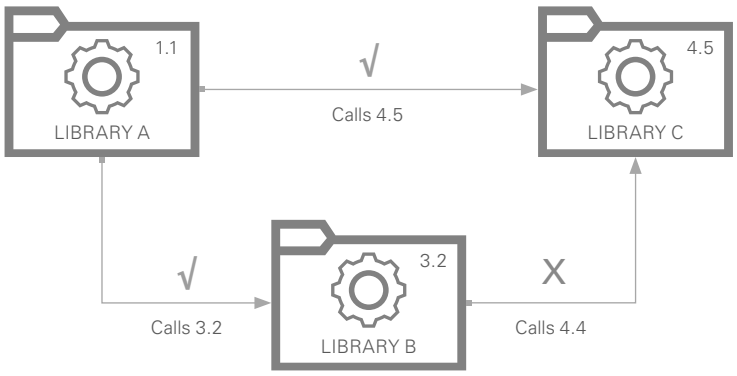


图4 由于库A依赖于库C版本4.5，因此库B对库C版本4.4的依赖会导致无法解决的依赖关系问题

硬件检测

硬件断言

有特定硬件设置需求的测试系统需要确定该硬件是否存在于系统中，并在该硬件不存在或不兼容部署计划的情况下执行应急计划。尽管开发人员经常通过检查测试机外观并将硬件组件与原始开发系统匹配来完成硬件断言，但更好的做法是假设测试系统由第三方所创建。测试系统的客户如何确定是否有不兼容的硬件？对于将正确模块插入不正确的插槽或端口的情况，系统能否做出应对？系统能否解决或调整硬件缺失的问题？尽早解决这些问题有助于简化测试系统的扩展和分发。

硬件标准化

硬件断言的最终目标是确定预期系统与实际物理硬件系统之间并无差异。

为此，最高效的做法通常是先在每个测试系统上对将要使用的硬件组件集合进行标准化：

- **文档记录**—标准硬件集合中的组件列表应该可供所有新系统访问。文档中必须包含供应商、产品编号、订单号、组件数量、可更换组件、保修、支持政策、产品生命周期等信息。
- **可维护**—硬件标准化最困难的问题之一是确保每个测试系统中使用的硬件组件将来仍然可用。通常，较旧的硬件会被制造商标记为生命周期结束(EOL)，需要刷新测试系统的标准硬件组件。从硬件升级和测试系统停机的角度来看，这种刷新会产生高昂的成本。与硬件制造商共同探讨硬件组件的生命周期策略有助于缓解日后的挑战。大多数硬件制造商(如NI)都提供生命周期咨询服务，并支持各硬件组件在生命周期内的逐步下线。
- **可复制**—应考虑是否需要全球性甚至区域性地部署硬件。必须确保有成熟的硬件部署方案，以便在偏远位置快速搭建新系统。对于很多系统，保证备用硬件组件的供货渠道以便进行维护或紧急更换也至关重要。

开机自检(POST)

即使测试系统选用合适的硬件并已正确连接，也必须对硬件进行简单测试，以确保系统运行后硬件能够按预期工作。幸运的是，大多数硬件组件都包含制造商预置的自检功能，通过该功能，可对设备的通道、端口和内部电路板进行简单检查。在为每个测试系统供电后，应为所有连接的设备执行自检程序，作为对硬件故障的早期检查。例如，每个NI设备都具有自检功能，可通过设备的驱动程序API以编程方式调用。因此，测试系统供电后的第一步可能是在每台设备上调用自检功能，并警告操作员任何硬件故障。

别名配置

遗憾的是，对硬件集合进行标准化并不能完全确保配置完全相同。通常，需要使用Measurement & Automation Explorer (MAX)等硬件配置软件将硬件设备重新映射到别名。例如，在安装所有硬件组件并为系统供电后，工程师可以使用MAX检测系统中存在的NI硬件，并使用Windows设备管理器查找非NI硬件。随后即可编辑.ini配置文件，将硬件设备正确映射到别名。该过程可能的输出如下图所示。

别名	设备名称
PXI NI-4139	PXI 1插槽1
PXI NI-3245	PXI 1插槽2
PXI NI-2239	PXI 2插槽1

表1 使用hw_config.ini文件将物理硬件映射到测试系统别名

编程配置

LabVIEW软件中用于NI硬件的System Configuration API等库可通过编程方式生成所有可用在线硬件的列表并配置别名映射。例如，测试系统可执行文件可以调用System Configuration API的Find Hardware函数来生成可用NI硬件列表。在该列表中，每个设备的别名属性均可通过硬件节点设置为预定义的名称。此方法可能导致系统出现问题，如导致硬件设备映射到不当的别名。因此，工程师应将此方法与其他保护措施结合使用，比如手动确认映射列表或标准化硬件集合。

最佳实践

基本用例：对于基本或简单的系统，务必要确保系统中存在预期的硬件。硬件标准化是适用于所有系统的最佳实践，随着硬件系统数量不断增加，这种做法尤为重要。应记录正确执行测试系统所需的机箱、模块和外设，并定期对文档进行修订更新。但通常只需要借助MAX等工具进行手动检查即可验证系统上运行的设备是否正确，无需采用编程或重新配置的方法。而随着硬件系统中的模块和设备数量不断增加，可能就需要换用更高级的解决方案，以防出现硬件缺失的问题。

高级用例：在复杂系统中，应综合使用不同的解决方案来跟踪系统中必需的硬件或系统中存在的硬件。与基本用例的做法一样，应跨系统进行硬件标准化和文档记录。如需检测故障硬件，应执行开机自检(POST)以确保连接的硬件按预期运行。此外，当硬件标准化失败时，应使用编程方式或手动操作别名映射系统，将预期设备自动重新映射到系统的别名。

依赖关系解析

依赖关系断言

一种切实可行的做法是制定计划来处理已部署系统上现有的依赖项和缺失的依赖项。通常，所部署的测试机器上已经安装了测试系统镜像的部分依赖项。对于规模较小的系统，只需重新安装所有依赖项便可确保万无一失。但是，对于规模较大的系统，重新安装所有依赖项十分耗时，建议先检查系统上是否存在相应依赖项。这种做法被称为依赖关系断言，有助于缩短部署时间，但需要针对依赖项差异进行规划。“组件化”部分将进一步讨论如何通过组件化加快部署。

例如，测试系统可能与14.0和15.0版本的NI-DAQmx驱动程序均兼容。测试系统可能会要求安装NI-DAQmx 15.0，但允许14.0版本充当依赖项。然而，用14.0版本代替15.0版本后，虽然不会影响兼容性，却可能会改变测试系统的行为。系统可能会跳过某些测试步骤或调用不同的函数。需要记录所有这些变化并进行测试。

依赖关系断言的第二个要素是确定如何处理缺失的依赖项。如前文所述，较好的做法是假定将测试系统部署到客户的机器上。是否应通知负责部署的工程师缺少依赖项？缺少的依赖项应该在后台静默安装，还是需要用户手动查找并安装？尽早解决这些问题有助于加快部署速度并正确处理缺失的依赖项。

最佳实践：

基本用例：对于基本的测试系统，通常没有必要进行依赖关系解析和断言。安装所有测试系统的依赖项，而不考虑其是否存在于系统中，这种做法通常比尝试确定缺失的依赖项并仅安装缺失的元素更加简单。但随着测试系统的扩展，系统总体安装时间可能会增加，达到一定规模时，开发依赖关系断言和解析工具将成为更有效的解决方案。

高级用例：即使网络连接稳定或镜像已进行压缩处理，部署时间也可能快速增加到不合理的程度。对于大部分高级测试系统，为了避免重新安装所有组件，有必要进行一定程度的依赖关系断言。在部署过程中，可以加入用于查找系统中安装的NI软件的System Configuration API或用于生成Windows机器上所有程序列表的WMIC命令集等工具。这样，安装程序便可跳过特定组件或允许存在版本差异。

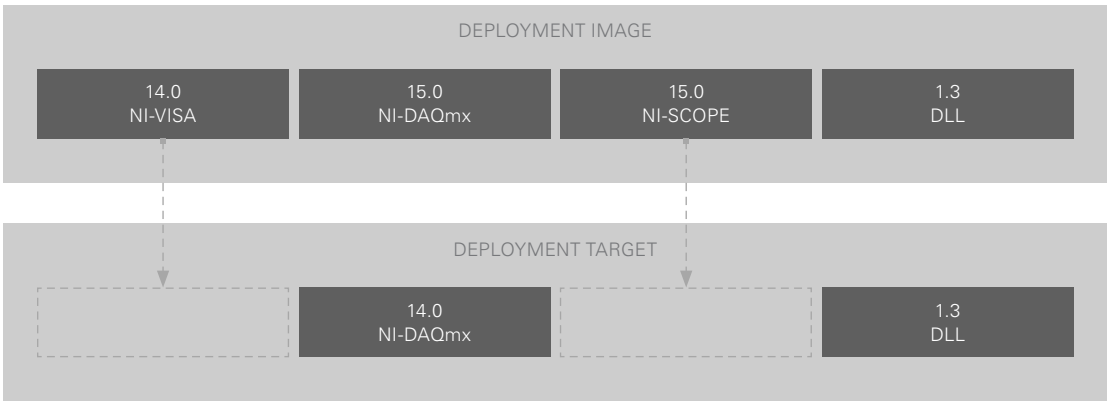


图 5 依赖关系断言

版本管理

通常，工程师需要了解当前部署在测试系统中的软件镜像版本，或者能够提供版本部署历史记录。

如有此类必要需求，则应配备版本管理系统来解决以下各项问题：

- 当前部署到系统A的是哪一版本？
- 系统B的最新部署状态是什么？
- 版本1、2和3部署到哪一系统？
- 系统A的版本历史是什么？

在大多数测试环境中，工程师需要使用铅笔和剪贴板系统来手动记录这些信息，但是，也有一些工具可以自动记录版本信息并提供特定系统的版本历史记录。这些版本管理工具可以集成到集成开发环境(IDE)中，也可以作为独立的版本管理工具存在。

示例工具：

- **Visual Studio Release Management**—Visual Studio IDE附带了这款支持自动化部署、版本历史跟踪和版本安全性管理的工具。
- **Jenkins Release Plugin**—利用这款用于Jenkins持续集成(CI)服务的插件，开发人员可以指定编译前和编译后的操作，以管理集成了Jenkins服务的开发版本。
- **XL Deploy**—此应用版本自动化(ARA)软件可以扩展到企业级别，并提供可视化状态仪表板、安全功能和分析功能来管理版本。

尽管上述示例工具可作为IDE和独立部署解决方案，但更常见的做法是将版本管理工具与CI服务器和端到端部署过程相结合。这种方法非常直观，因为对于部署过程来说，更常见的问题是某个机器上存在哪些特定代码，而不是使用哪个版本。对于已编译的系统镜像，这一点可能很难通过手动观察来确定。要实现版本管理最佳实践，有必要跟踪从开发到部署所用的代码。

端到端系统自动化

要为测试系统部署开发更为复杂的端到端流程，高效的版本管理必不可少。从开发到部署，这一过程中的每一步都与先前的步骤紧密相关；如果源代码管理良好，测试和编译也可以得到有效管理。有了良好的测试和编译流程，版本管理就只是原系统的简单扩展。下图所示为典型的端到端系统。

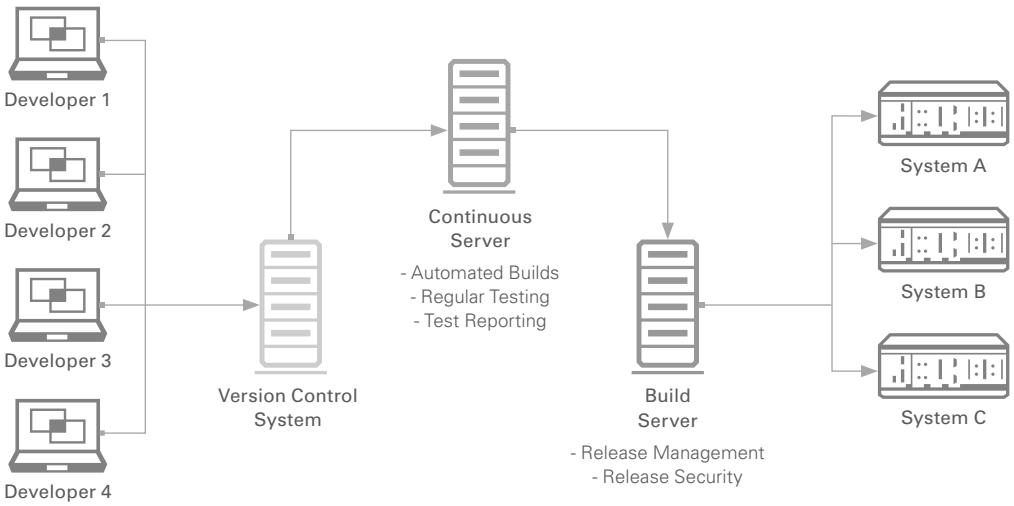


图 6 开发人员将代码提交到版本控制存储库，然后可以在CI服务器中进行编译和测试。在版本控制存储库中，生成版本可存储在编译服务器中并进行版本管理

在此架构中，测试系统开发人员会定期开发源代码并将其提交到版本控制存储库。随后，CI服务可将源代码从版本控制存储库提取到自己的存储库，并适当地编译和测试代码。此时，开发人员可通过自动或手动方式将通过CI测试的代码移动和存储到编译服务器或存储库。编译服务器通过报告和跟踪功能将每个软件版本链接到特定的测试机器，实现版本管理。通常，测试机器通过请求安装测试系统的特定版本来启动部署过程；但是，开发人员也可以配置编译服务器以将镜像推送到选定的机器上。

如果基本系统也需要采用一定级别的版本管理，则应根据版本需求的内在复杂性选择最实用的解决方案。如果需求是跟踪部署到系统的版本，则通过配置文件或作为编译可执行文件的组件进行手动版本管理即可。如果需求范围扩大、测试系统数量增加或应用程序版本数量增加，则有必要使用定义的版本管理系统。

最佳实践：

高级用例：对于需要进行版本管理的复杂测试系统，采用特定形式的端到端自动化方案通常效果最佳。最简单的方式是选择Jenkins或Bamboo等CI服务，这些服务可将版本管理与版本测试和源代码控制相结合。

版本测试

回归测试

在软件工程中，回归测试是指在先前开发的系统发生更改后所进行的测试。回归测试的目的是保持每个版本的完整性，并跟踪系统在特定更新或补丁处的错误。对于组件化系统，回归测试对于确定模块A的升级是否会导致模块B中出现意外行为尤为重要。例如，升级系统中的NI-DAQmx硬件驱动程序可能会导致硬件抽象库出现问题，该库调用的较旧NI-DAQmx版本中的函数现已在较新版本中弃用。回归测试有两种类型：功能测试和单元测试。

功能测试

在测试系统中，关于软件更新的最重要的问题是，这些更改是否会破坏系统的功能？系统是否仍然按照预期的方式运行？功能测试会验证系统在采用一组已知输入时是否会生成预期输出，可以帮助用户解答有关整套系统的这些问题。这种类型的测试通常采用“黑盒”方法；不分析系统的内部机制，只分析系统的输出是否符合预期。对于测试系统，此测试可验证更新硬件配置、更改驱动程序或添加测试步骤是否不会改变原有的测试功能。工程师可在使用可模拟待测设备(DUT)的测试系统上执行功能测试，验证这些系统经校准后是否通过特定测试。例如，用于测试对象是否为圆形的系统由四个组件组成：相机控制器、周长传感器、直径传感器和体积传感器。如果系统从版本1.0更新为版本1.1并对直径传感器进行了更改，则下图中的第二个待测圆形在更新前可通过圆形测试仪的测试，更新后则无法通过测试。

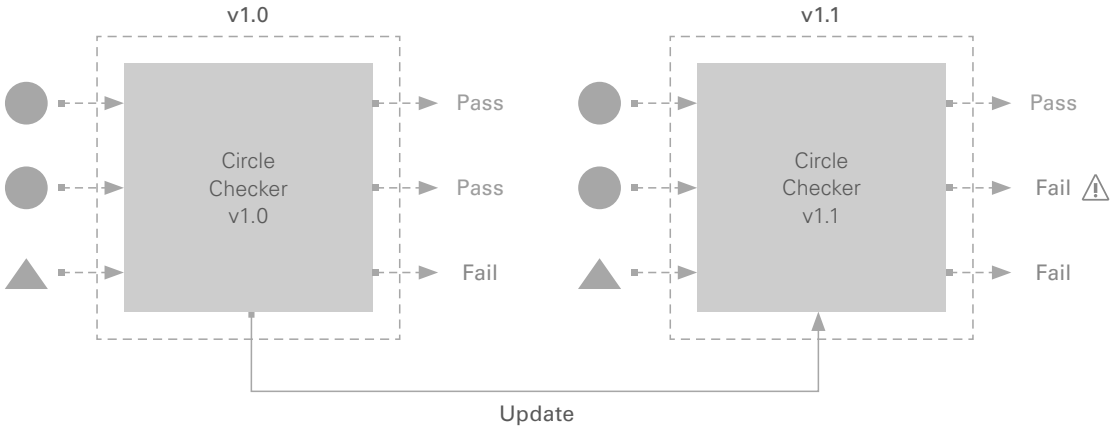


图 7 对测试系统中的模块进行少量更新可能会导致功能测试出现故障，这是一种伪故障

单元测试

功能测试针对的是整个系统，而单元测试则是针对特定模块、组件或功能。此类测试旨在跟踪测试系统特定部分的质量，而不仅仅是正确性。例如，如果测试结果记录到数据库中，则可以在数据库控制器上进行单元测试以测量数据吞吐量。通过这种方式，不仅可以分析对数据库控制器进行的更改以确定日志记录功能是否可正常运行，还可以了解软件更改是加快还是减慢了系统的日志记录能力。除了有助于查找错误之外，单元测试还可以将发现的性能升降情况与特定更改相关联。之前提到的圆形测试仪示例可以解释单元测试和功能测试之间的区别。假设圆形测试仪的直径传感器软件组件像之前一样升级，则可以只对直径传感器进行单元测试，而不是对整个系统进行功能测试。对于单元测试，可以为特定组件提供表示具有特定直径的圆形的二进制图像数据，并测试输出是否与该已知直径匹配。这种方法可验证模块的正确性并进行定量测量，例如测量模块的执行时间。

在这种特定情况下，升级明显降低了模块的速度。还可以推断，由于升级后系统未通过功能测试但通过单元测试，相机控制器与直径传感器之间的通信很可能存在软件错误。这种验证系统正确性和个体模块功能的能力可确保仅将高质量的版本部署到测试机器上。

测试过程

在大多数测试系统中，为节省开发时间，回归测试与源代码控制、编译或版本管理同时进行。这样便可重复使用更新相对不频繁的测试代码。但是，针对测试代码的编译开发时间制定计划和预算也很重要。通常，回归测试是CI服务或IDE的组成部分，其中源代码控制、编译和测试都会按顺序进行。

最佳实践：

基本用例：对于所有测试系统，在将系统部署到新机器之前，都应该进行一定程度的功能测试。这种功能测试既包括使用模拟硬件在开发环境中手动运行应用程序的简单情况，也涉及基于配置文件运行一系列功能测试等稍显复杂的场景。对于比较简单、相对单一的应用，单元测试可能并无必要，但随着测试系统复杂性的提高，就需要考虑进行单元测试。随着添加的模块越来越多，有必要通过特定的自定义测试来跟踪漏洞或确保系统满足某些规范。

高级用例：对于复杂的测试系统，不应仅对测试系统每个新版本的各种输入进行功能测试，还应针对系统的每个单独模块开发单元测试。两种回归测试方法都应该在部署过程中最有效的时间点进行。例如，可以在编译每个版本后执行功能测试，在每个源代码控制提交点执行单元测试，这样便很好地混用了这两种回归测试。通常，这些测试对于系统来说是强制或默认执行的，尤其是对于航空航天和国防工业领域的系统。

图 8 对v1.0和v1.1执行单元测试后，处理时间被识别为更新后的一项问题，导致过程功能测试出现伪故障

组件化

由于部署时间是大型测试系统的常见考量因素，所以最佳做法是只更新测试系统中需要更改的单个组件，而不是重新编译整个系统。本指南的“依赖关系解析”部分介绍了该主题的部分内容，但仍需要单独探讨如何开发模块化架构或基于插件的架构以提高部署效率。无论工程师选择何种架构，最佳做法都是定期更新外设模块，而更核心的模块在开发时保持相对恒定，无需重新编译。这种做法自然会导致有关更新频率的问题，本节稍后将对此进行探讨。

部署插件架构

软件部署中使用的插件是一个代码模块，在安装上独立于主应用程序，在功能上独立于其他插件，遵守全局插件接口规定，用于编译的应用程序中时可避免名称冲突。主应用程序随后应能够动态加载每个插件，通过标准接口调用每个插件，并将每个插件用作扩展，而不需要重新编译。成功开发后，插件框架可对部署进行组件化处理，即仅更新或安装特定或缺失的插件，而无需重新编译主应用程序或任何未受影响的插件。

例如，为简单应用程序开发的插件框架可能包含一个主要的可执行文件，该可执行文件会在加载时搜索插件目录或在运行时定期搜索插件目录，并通过标准接口执行该插件。这样，无需编辑主应用程序即可将插件持续部署到系统的插件目录中。

硬盘驱动器复制

通常，代码库、硬件驱动程序或特定文件是测试系统核心的组成部分，不需要像其他模块化外设组件一样频繁更新。在这类情况下，可通过硬盘驱动器复制对环境进行标准化处理，以作为进一步开发的基准。工程师可以将开发机器或归零测试机器的硬件驱动器复制并克隆到其他测试机器上。驱动器复制完成后，测试机器将拥有共同的起点，通常包括主测试应用程序或程序、必要的硬件驱动程序、系统驱动程序集和关键的外设应用程序，例如用于硬件配置的MAX。但必须意识到，硬盘驱动器复制有其自身的限制，比如需要在各测试机器之间使用相同的计算机硬件或分发占用大量内存的镜像，因此硬盘复制不适用于软件频繁更新的情况。

使用硬盘驱动器复制为进一步的测试开发奠定基础的一个示例是将常用的硬盘驱动器复制工具Symantec Ghost与TestStand Deployment Utility (TSDU)结合使用。在下图(A)中的第一个框中，开发机器将其核心软件堆栈(红色)复制到目标机器上。该核心软件堆栈是Windows操作系统、硬件设备驱动程序、运行引擎和MAX的组合。目标机器生成镜像后，便可在开发机器上进行开发(B)，使用TestStand和LabVIEW(绿色)创建测试序列。开发人员随后可以使用TSDU将测试序列移动到目标机器。如果需要频繁对测试序列进行更新，开发人员可以继续使用TSDU以节省开发时间，因为核心软件堆栈不需要更改。有时，开发可能是在未部署到目标机器(C)的开发机器上进行。这种系统不匹配可能会导致依赖项缺失问题。在这种情况下，开发人员可以不使用TSDU来更新目标机器，而是选择重新为开发机器生成镜像并将其复制到目标机器上，以同步这两台机器(D)。之后，开发人员可以继续使用TSDU进行频繁更新，如果将来出现系统不匹配问题，可以使用Ghost重新为目标机器的硬盘驱动器创建镜像。

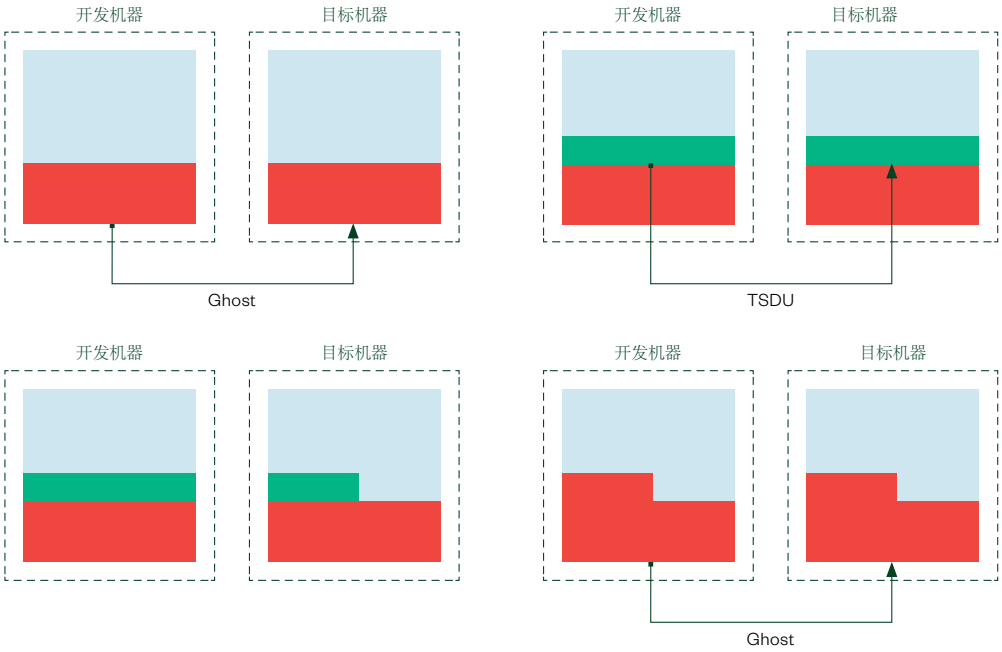


图 9 TSDU和硬盘驱动器复制示例

持续集成和持续部署

持续集成(CI)是指通常在单独的CI服务器上持续提交、编译和测试代码的做法。大多数测试系统会使用CI服务提供编译、测试和部署系统软件所必需的框架。这些服务在CI服务器上定期自动运行，具有多种配置选项，可用于创建编译计划、自动化测试规则和版本部署等。使用CI服务器最明显的优势之一是能够跟踪和管理不同的编译和部署。

编译

版本	状态	上次编译时间
1.2	未通过	2016年5月24日
1.1	通过	2016年4月2日
1.0	通过	2015年12月7日

部署

版本	状态	上次编译时间	机器
1.0	通过	2015年6月7日	A
1.1	未通过	2015年6月9日	B
1.1	通过	2016年3月16日	C

表 2 CI服务提供仪表板来跟踪应用程序的编译和部署

CI工具涵盖多种功能，开源开发人员和软件公司都积极开发CI工具。后者还具有为系统设置提供支持的额外优势。

- **Jenkins**—Jenkins被誉为“领先的开源自动化服务器”，因其安装和配置的便捷性而成为目前最受欢迎的CI服务之一。Jenkins几乎可与所有编程语言一起使用，因为它可以通过命令行接口或丰富的Jenkins插件与程序连接。
- **Bamboo**—Bamboo是软件公司Atlassian开发的一项领先的专有CI服务。除了Bamboo的测试、编译和集成功能之外，Atlassian还可提供优于Jenkins的“一流的部署支持”。
- **Travis CI和Circle CI**—这两项开源CI服务可提供强大的扩展能力，但只与位于GitHub存储库中的项目集成。

总体而言，CI的目标是提供自动化且可配置的工具，使开发人员能够在编译和测试软件的同时继续编码。

最佳实践：

基本用例：简单的系统通常无需过度关注组件化。尽管系统使用很少的代码模块或不采用插件架构，但每个测试系统通常可以部署为独立的应用。但是，如果安装时间变得非常长，并开始导致部署速度变慢，则可能需要转为使用更加组件化的方法，避免重新安装所有组件。

高级用例：当测试系统变得庞大、复杂或使用插件架构时，就有必要从单一部署镜像转为可单独更新每个组件的模块化部署。利用插件架构可快速实现这种模块化设置，但配置CI服务也是一种可行方法。

实际案例

高级部署框架的一个案例是一家音频设备生产公司使用TestStand和LabVIEW对其产品进行功能性电气测试。这家音频设备制造商的测试部门在全球部署了超过50个测试系统。每个系统都使用混有多种模块的PXI机箱，包括数据采集模块、数字I/O模块、数字信号采集模块、数字万用表模块和频率计数器卡模块。

负责部署的测试工程师为每个即将上线的新测试系统执行下述程序。

01

创建基本系统镜像

每个新测试系统都有一个确保系统安全所必需的软件列表，其中包括公司开发的软件和第三方软件。公司的IT部门需要使用这些杀毒软件、VPN安全应用程序和Windows组策略配置规范。其次，每个系统都需要一套基础软件来执行其必要的测试序列。该软件的主要组件是一组对照已发布NI系统驱动程序集进行交叉检查的驱动程序。也就是说，一个版本的测试系统可能包含NI-DMM 14.0、NI-Switch 15.1、NI-FGEN 14.0.1和NI-DAQmx 14.5驱动程序。此外，还需要LabVIEW 2014和TestStand 2014的运行引擎来运行主要的测试系统可执行文件。下表列出了所有必要软件的概况。

要部署到新测试系统，需要先在开发机器上创建此镜像，并使用硬盘驱动器镜像软件复制此镜像。该软件镜像可能是以前创建的，因此可以在多台机器上重复使用。这有助于降低部署成本，因为每批相同的测试机器只需安装一次。

通过安装所有必要的软件生成基本系统镜像后，使用Symantec Ghost复制硬盘驱动器并将新镜像上传到编译服务器。编译服务器位于总部，该服务器只需要维护一个大容量存储器，以便将多个系统镜像保存在服务器上。

02

部署基本镜像

将基本镜像上传到编译服务器后，测试工程师会将新的测试系统连接到公司网络，然后使用Web界面连接镜像服务器并浏览各种可供安装的基本系统镜像。选择适当的版本后，Symantec Ghost会使用复制镜像对新系统硬盘驱动器进行镜像。至此，测试系统便已具备执行测试序列必备的基本软件。

03

验证硬件

在将必要的硬件模块物理安装到PXI机箱并启动系统后，测试工程师需要在软件中将系统别名映射到实际设备。尽管已提供模块列表及相关插槽编号，但测试工程师必须使用配置系统设置来映射别名，以便模块位置可在系统之间切换。对于该公司，每个测试系统都使用工程师编辑的.ini文件以提供实际系统硬件到测试系统别名的映射。此过程通过在MAX中识别设备并手动编辑.ini文件以创建适当的映射来实现。

软件	版本
NI-DAQmx驱动程序	14.5.0
NI-DMM驱动程序	14.0.0
NI-Switch驱动程序	15.1
NI-FGEN驱动程序	14.0.0
LabVIEW运行引擎	2014
TestStand运行引擎	2014
内部杀毒软件	3.2

表3 创建基本系统镜像时，必须明确列出将包含的驱动程序和运行引擎版本

04

安装应用程序和组件

至此，测试系统已安装基本系统镜像并验证了实际硬件。现在，工程师的任务是安装最新版本的测试应用程序。本例中，应用程序是由TSDU生成的TestStand安装程序，其中包含所有必要的代码模块、序列文件和支持文件。为明确该安装程序的生成方式，请务必查看生产公司采用的开发系统。每个开发人员会在LabVIEW中创建特定的测试步骤，或在TestStand中创建测试序列，并将测试步骤或测试序列提交到Apache Subversion源代码控制存储库。该存储库位于运行CI服务(Jenkins)的服务器上。Jenkins服务会对提交的代码模块运行测试，使用TestStand序列分析器通过命令行验证序列，然后使用TSDU命令行接口将必要的测试序列编译到安装程序中。每个安装程序在编译完成后，都会连同其必要的支持文件一起自动通过Jenkins Deploy Plugin部署到编译服务器。

05

执行

将TestStand安装程序部署到编译服务器后，测试工程师即可将安装程序下载到新的测试系统。然后，工程师随后运行该安装程序，找到主要的测试可执行文件，并开始运行基本测试系统。

利用此部署系统，测试工程师可以快速轻松地对每个测试系统进行更改。现有的硬盘镜像系统可用于进行大规模代码修订或驱动程序集升级，而较为轻量级的编译服务器可用于部署对主测试应用程序或单个组件和插件的细微更改。

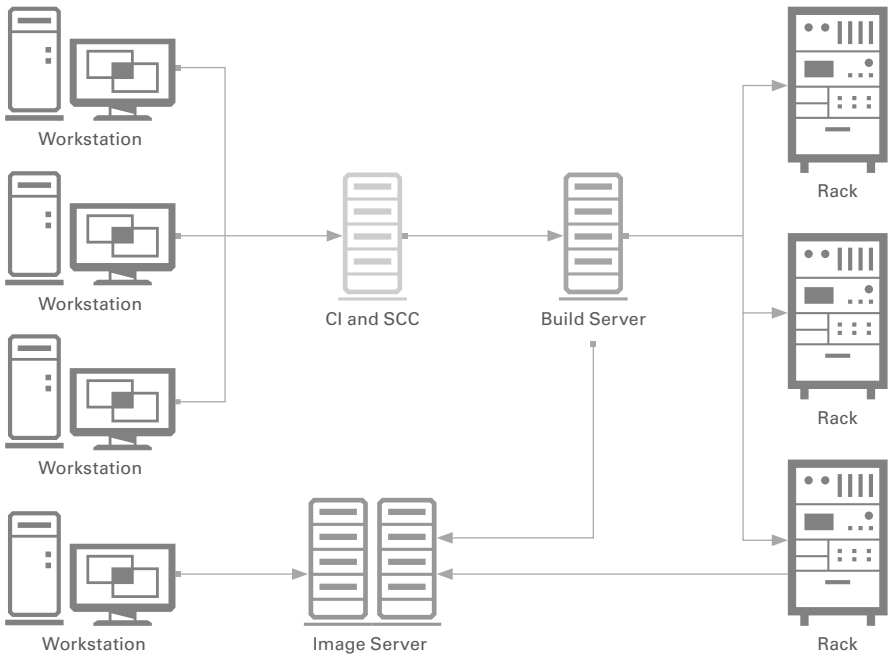


图10 此测试部署系统使用镜像服务器来存储和部署基本系统镜像，从而使各个测试站彼此保持同步。随后，开发人员会定期将源代码上传到持续集成和源代码控制服务器，以定期编译和测试提交的代码。提交的代码通过了所有必要的测试之后，系统便会将编译的镜像添加到编译服务器，由该服务器负责处理测试软件系统镜像的大规模分配。

总结

测试系统部署通常是一个复杂的过程，随着测试系统的升级和数量的扩展，复杂性还将持续提升。在测试系统开发的早期敲定部署过程是实现可扩展的成功部署的关键。要打造成功的部署过程，首先应确保已确认并定义所有必要的测试系统工件，并且采用适当的部署方法。动态硬件配置选项也是很多部署系统的重要考量因素。对于规模更大、更高级的系统，动态解析部署镜像与目标机器之间的依赖关系有助于降低部署过程的复杂性，并可缩短升级或重新创建系统镜像所需的时间。管理和测试部署镜像的各个版本是测试系统开发人员的另一个重要考量因素。无论是采用持续集成服务还是配置文件，都有必要维护一个可扩展的版本管理系统以实现分布式部署。测试系统的部署方法必须根据测试系统的功能和性质进行高度自定义。本指南中提供的构建可扩展解决方案的建议为通用建议，与使用的工具或系统功能无关。

TestStand Deployment Utility

TestStand Deployment Utility会自动执行部署中涉及的众多步骤，包括收集测试系统的序列文件、代码模块和支持文件，以及为这些文件创建安装程序，能够降低TestStand系统部署的复杂性。

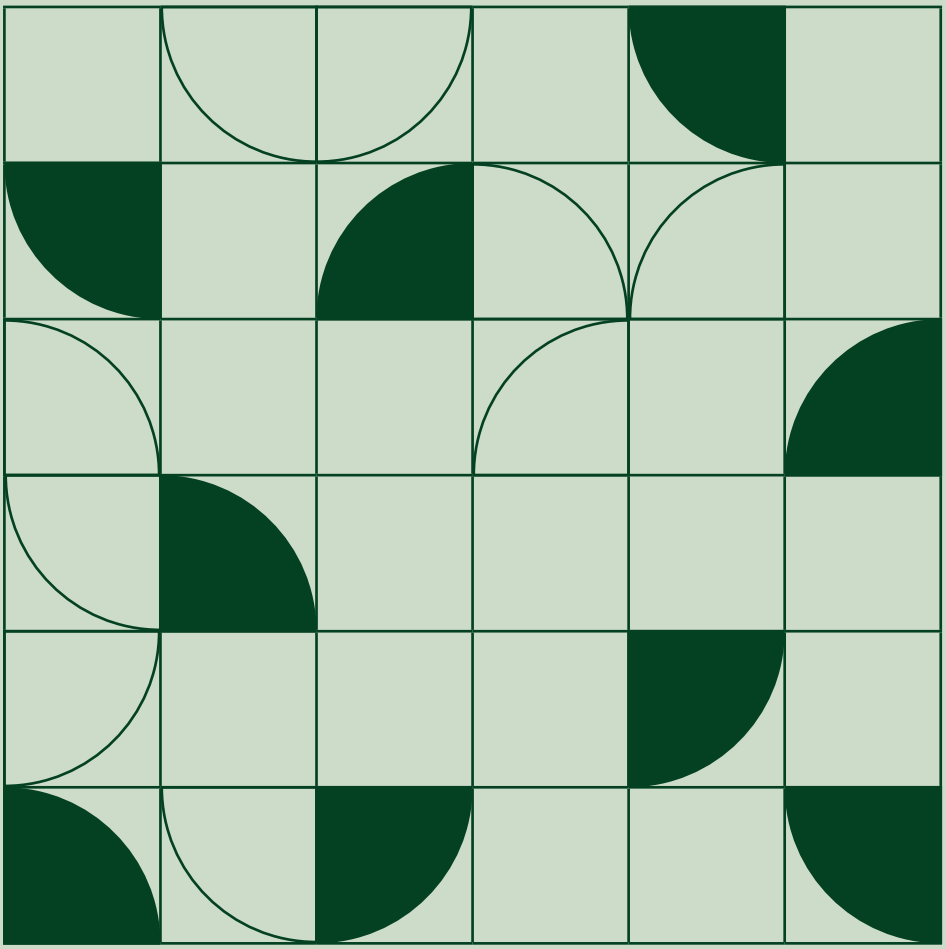
了解TestStand Deployment Utility的更多信息

LabVIEW Application Builder最佳实践

LabVIEW Application Builder最佳实践可简化LabVIEW应用程序的管理和组织。这些建议可帮助工程师在开发之前确定指导原则和操作步骤，以确保其应用程序适用于大量VI和多个开发人员，从而节省开发时间和资源。

开始使用Application Builder最佳实践开展LabVIEW项目

系统维护



简介

在理想情况下，系统永远不会出现故障。我们只需要安装系统、打开系统、然后运行系统，之后就完全放任不管，直到10年、15年、20年甚至更长时间后将系统淘汰。不幸的是，现实并非如此，或者说至少还没有实现。系统故障和突发的意外故障可能会带来高额的损失。即使有最周全的计划，我们也不能完全消除发生故障的风险，但我们可以降低这些风险。维护策略可以帮助您管理此成本并降低故障风险。

维护计划对于确保自动测试设备(ATE)系统在整个生命周期内具有最低总拥有成本至关重要。在设计中融入可维护性的系统与完善的维护计划相结合，有助于：

- 保持系统的功能性和延长使用寿命，最大限度地提高资本投资收益
- 管理物流、调度和备件库存，最大限度地减少停机成本

任何维护计划的目标都是保持系统尽可能长时间地正常工作，而且在系统停机时尽快使其恢复正常工作。另外，还要以尽可能低的成本实现这一目标。

概念和定义

维护是指执行相关服务来保持系统正常运行并在系统出现故障时对系统进行修复的一种活动。维护分为三个方面：预测性维护预防性维护、和纠正性维护。

可维护性是指执行维护的难易程度。有些行业将其称为可服务性。可维护性越高，控制维护成本就越容易。

预测性维护通过状态监测提前检测出系统故障，在工业中称为基于状态的维护。当预测到潜在故障时，就会安排维护活动来对系统进行维护。这些活动可以延长系统使用寿命，并避免计划外停机。预测性维护活动通常在检测到维护需求时才会开展，并会导致计划内停机，而计划内停机的成本通常远远低于计划外停机的成本。计划内停机成本可以分摊到接受维护的许多其他系统。预测性维护的目标是在发生故障前尽可能长时间地使用系统/组件，从而最大限度地提高资本投资收益，并尽可能减少计划外停机成本。随着物联网以惊人的速度向前发展，智能机器的概念已经深入人心，智能机器可以自我监控，并在需要维护时与其他机器组成的网络进行通信。传感器、嵌入式控制器、FPGA、网络和大模拟数据(Big Analog Data™)分析等领域的技术进步使得预测性维护比以往更容易，也更具成本效益。预测性维护的衡量指标是引发的停机时间；这个时间称为平均预测性维护时间(MPdMT)。

预测性维护活动包括：

- **状态监测**—这可确保系统正常运行、检测故障的发生，并识别可能导致系统故障的组件隐患或性能退化。基于经济高效的嵌入式微处理器和FPGA技术，内置自检和状态监测技术已经得到广泛应用。这有时被称为预测和健康管理(PHM)或系统健康监测。该概念是指检测系统中的性能变化和隐藏故障，防止出现更严重的系统故障。

  如今，大多数汽车都配有发动机自动健康监测系统，用于检测问题并使发动机检查灯闪烁，提醒维护人员在发动机永久损坏之前及时进行维修。测试系统可以监测温度、风扇转速、内存使用、测试时间、测量精度、计数继电器操作等。

- **维修系统组件**—这有助于减缓损坏速度并延长系统的使用寿命。

  一些汽车轮胎会配有检查气压的传感器。如果气压不合适，会缩短轮胎的使用寿命并影响油耗性能。如果将测试系统用于灰尘较多的环境，则可能需要清除空气过滤器和外壳内部的灰尘，以免系统过热，导致电子器件的使用寿命缩短。监测系统的内部温度或气流可以帮助您了解何时可能需要清理灰尘过滤器。

- **更换系统组件**—在组件发生故障之前更换组件，以免出现计划外停机。

  测试系统可以使用继电器来开关信号，以测试待测设备。根据所开关的电气负载，继电器仅可持续预计的操作次数。因此，与等到发生故障并出现计划外停机后再采取措施相比，监测操作次数并在继电器模块发生故障之前予以更换，通常更具成本效益。

- **通过校准补偿漂移**—测量系统的目的是提供可信的测量数据。如果测量数据不可信，则系统运行不正常。

  大多数测试系统的电子器件都需要以一定时间间隔进行校准。但是，如果使用的是尖端技术，可能我们还不能很好地把握校准间隔。因此，建议通过监测测量漂移来了解何时适合安排校准维护。

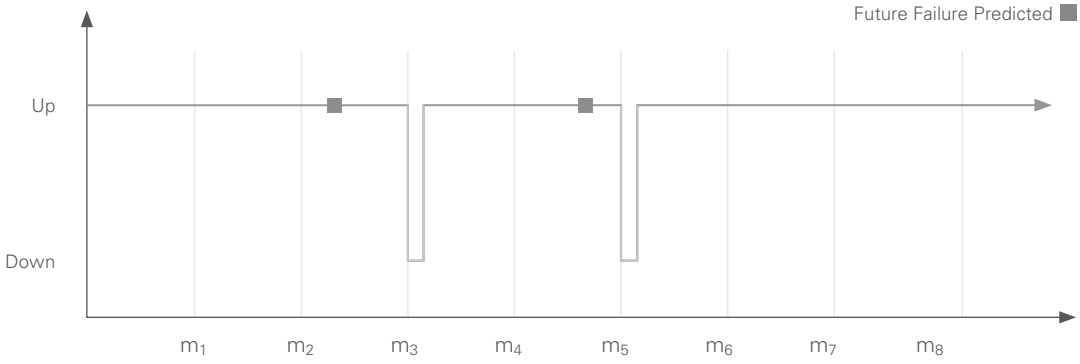


图1 查看预测性维护的正常运行时间和停机时间随时间的变化。预测性维护可降低停机频率并将昂贵的计划外停机转换为成本较低的计划内停机，从而最大限度地利用您的资本投资并减少停机成本，但需要借助故障监测设备和预测软件

- **验证**—这可确保系统在重新联机之前正常工作。如果在故障情况下联机，只会增加停机时间。
- **系统重新联机**—必须始终考虑这一点，因为对于某些应用，这并不是一项简单的任务。

  例如，如果测试是制造过程的一部分，则让系统重新联机可能需要停止生产线并且使测试装置与生产流程重新同步。

预防性维护活动包括：

- **维修系统组件**—这有助于减缓损坏速度并延长系统的使用寿命。

  这就是为什么汽车机油需要定期更换的原因。测试系统通常运行复杂的软件程序，这些程序可能存在隐藏的资源泄漏和/或故障，最终导致系统故障。简单的系统重启可以将软件刷新到新状态。如果将测试系统用于灰尘较多的环境，则可能需要清除空气过滤器和外壳内部的灰尘，以免系统过热，导致电子器件的使用寿命缩短。如果不能监测温度和/或气流，则可能需要安排定期维护。

预防性维护是指对系统进行维护以防系统发生故障并能延长系统使用寿命的活动。预防性维护活动通常需要按计划进行，并会导致计划内停机。计划内停机成本可以由接受预防性维护的许多其他系统分摊。预防性维护的目标是最大限度地减少计划外停机成本。预防性维护的衡量指标是引发的停机时间；这一时间称为平均预防性维护时间(MPMT)。

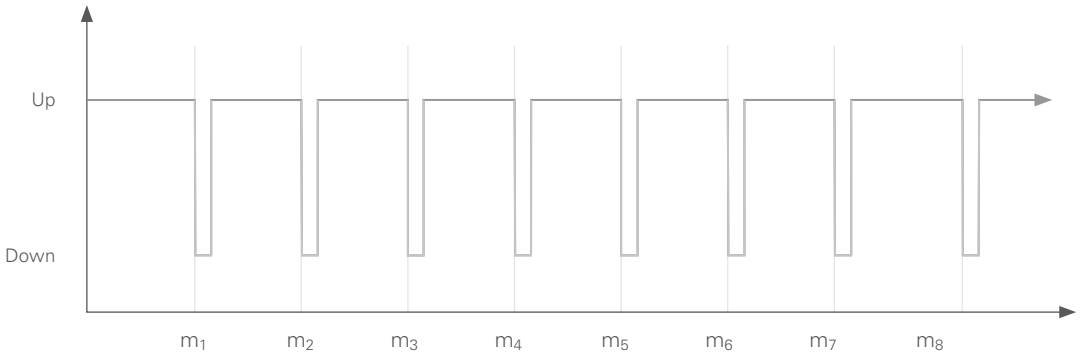


图 2 预防性维护并不总能最大限度地利用您的资本投资，但它有助于避免昂贵的计划外停机，从而最大限度地减少停机成本

- **更换系统组件**—在组件发生故障之前更换组件，以免出现计划外停机。

  汽车上的轮胎或刹车片在汽车行驶一定里程后需要更换，以防出现故障，从而避免发生事故或汽车半路抛锚。测试系统可能配有测试该设备的连接器引脚，这些引脚往往会在100,000次连接之后就会出现损坏。按每小时测试50台设备计算，连接器应在持续使用约2,000小时或83天后才会出现损坏并发生故障。预防性维护应每隔约80天安排一次，以更换连接器。在发生故障之前更换通常比等到故障发生并出现计划外停机后再采取措施更具成本效益。

- **通过校准补偿漂移**—测量系统的目的是提供可信的测量数据。如果测量数据不可信，则系统运行不正常。

  大多数测试系统的电子器件都需要以一定时间间隔进行校准。

- **验证**—这可确保系统在重新联机之前正常工作。如果在故障情况下联机，只会增加停机时间。
- **系统重新联机**—必须始终考虑这一点，因为对于某些应用，这并不是一项简单的任务。

  例如，如果测试是制造过程的一部分，则让系统重新联机可能需要停止生产线并且使测试装置与生产流程重新同步。

纠正性维护是指修复故障系统以让其恢复到正常工作状态的活动。纠正性维护活动通常不会事先安排，因而会导致计划外停机。纠正性维护的目标是在故障发生前尽可能长时间地使用系统/组件，从而最大限度地提高资本投资收益，并在故障发生之后尽可能减少计划外停机成本。纠正性维护的衡量指标是故障引发的停机时间；这个时间称为平均修复时间(MTTR)。

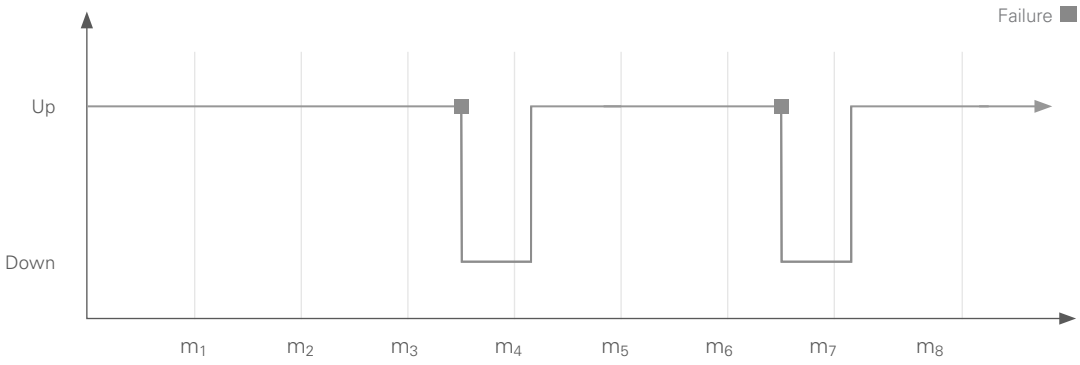


图3 查看纠正性维护的正常运行时间和停机时间随时间的变化。纠正性维护可最大限度地利用您的资本投资，但由于会引发计划外停机，不会将停机成本最小化。您可以采取措施尽量缩短计划外停机时间或MTTR

纠正性维护活动包括：

- **检测**—尽快检测系统故障，可最大限度地减少成本高昂的计划外停机，并可能防止对系统中的其他组件和/或同一流程中使用的其他系统造成损坏。

  汽车中的压力传感器可以尽快检测到油压下降，从而提醒驾驶员，防止对发动机造成永久性损坏。油压下降可能是油泵出现故障或漏油导致油位降低，而修理油泵或将泄漏处密封然后加油要比购买新的发动机便宜得多。对于ATE系统，电子器件可能会发生故障，从而影响关键测量并导致测试结果不正确。如果故障需要一段时间才能检测出来，则公司可能在不知情的情况下将不合格产品提供给客户，或者冷却风扇可能出现故障，导致机箱温度可能上升到会损坏部分电子器件的水平。

- **诊断和隔离**—在检测到故障后进行正确诊断和故障隔离可以帮助操作人员和维护人员快速找到并维修故障组件，从而最大限度地减少计划外停机并节省成本。

  汽车机械师可借助复杂的诊断设备来有效且高效地诊断问题。这可降低修错或换错部件的风险，从而节省时间和成本。对于复杂的ATE系统也是如此，如果没有适当的诊断工具，对问题进行诊断就可能花费数小时甚至数天时间。

- **维修**—通过维修或更换故障组件来修复系统。计划外停机时间在很大程度上受到有无备件的影响。根据应用、环境和人员技能水平的不同，就近提供备用系统或备用部件可能不一定具有成本效益，也不一定切实可行。

  大多数人不会在车辆没有备胎的情况下开车穿越全国，但如果仅开几个街区，他们很可能会这么做。

  备件战略对于控制成本至关重要。此时需要考虑以下问题：备件是否应存放在现场或附近的服务中心？如果要求供应商提前从工厂发出替换件，是否需要支付费用，或者只能等到设备故障再返给供应商维修？答案取决于计划外停机的成本。组件的数量、组件的平均故障间隔时间(MTBF)和补充备件所需的时间决定了所需备件的数量。一些公司提供不同级别的备件服务来帮助评估所需备件的数量、物流和备件管理成本。

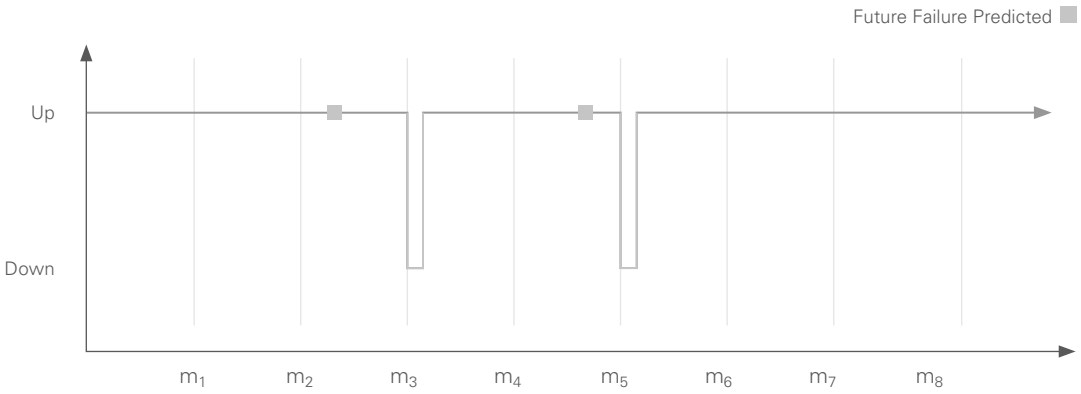
- **验证**—这可确保系统在重新联机之前正常工作。如果没有此步骤，系统可能仍然不能正常运行，只会导致更多的计划外停机。

  这就好比汽车的刹车修好了之后，直接在高速公路上高速驾驶汽车，而没有首先测试和验证刹车是否能够正常工作。

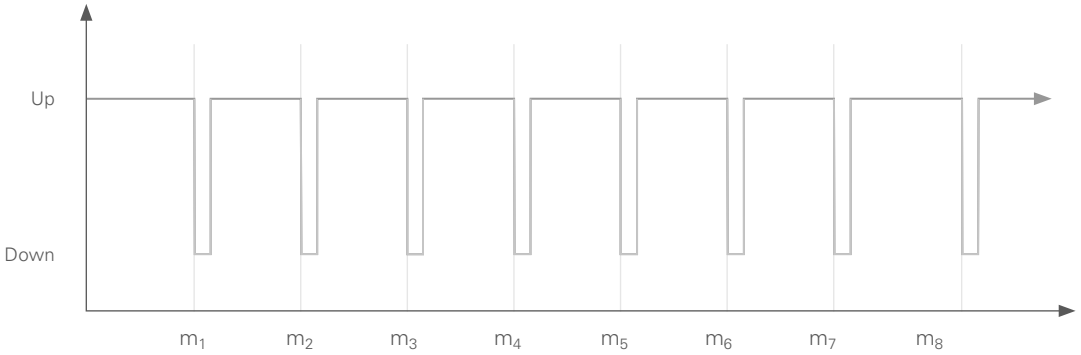
- **系统重新联机**—必须始终考虑这一点，因为对于某些应用，这并不是一项简单的任务。

  例如，如果测试是制造过程的一部分，则让系统重新联机可能需要停止生产线并且使测试装置与生产流程重新同步。

PREDICTIVE



PREVENTIVE



CORRECTIVE

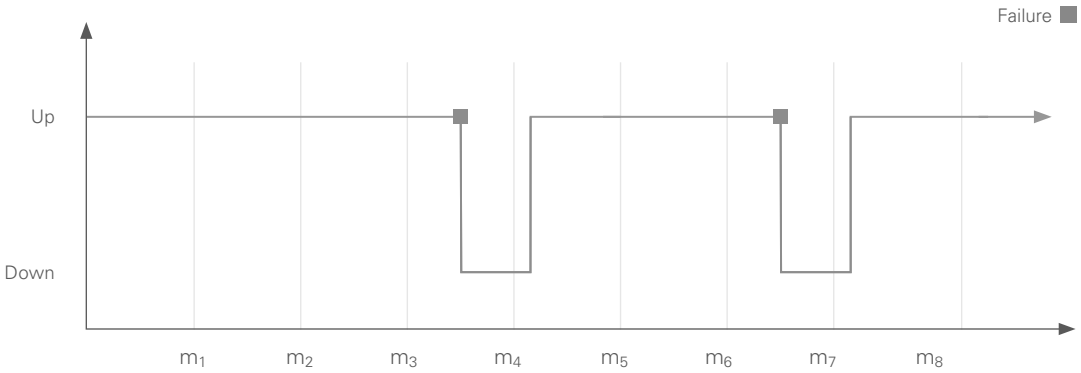


图 4 对比一下正常运行时间和停机时间，可以看出计划外的成本通常远远高于计划内停机的成本

在设计中融入可维护性

系统的设计极大地影响了能否部署有效、高质量和可控的维护计划。若要在设计自动化测试系统时就考虑提高系统可维护性，建议考虑以下最佳实践和指南。

自检和监测

自检和监测对于减少计划内和计划外停机非常重要。从开始设计时就将自检和监测功能纳入系统中对于实现高效、有效的健康监测、故障检测、故障诊断和隔离以及系统验证至关重要。

模块化设计

模块化设计可简化维护、更换、维修和校准系统组件相关的活动，减少相关的时间，同时还能优化系统诊断和故障隔离，并在计划外停机期间节省宝贵的时间。此外，模块化设计还可降低与备件相关的成本。您无需在库存中备用几台完整的系统，只需备用一些组件、子系统或模块。组件通常具有不同的故障率，故障率较低的组件需要较少的备件，而故障率较高的组件需要较多的备件。

标准化

标准化可以大大降低成本，因为它简化了物流，减少了备件数量、所需维护工具和设备数量以及培训成本。

例如，有些航空公司使用10种甚至更多型号的飞机。然而，西南航空公司只使用一种型号的飞机—波音737。这有助于成本节省。机械师只需针对一种型号的飞机进行培训，而且库存也只需准备一种飞机的备件。他们可以在最后一刻换出飞机进行维修。机队中的飞机完全可以互换。所有机组人员和地勤人员都非常熟悉这种飞机。而且，飞机的存放方式和存放位置也不会有任何挑战，因为所有飞机都具有相同的形状和尺寸。

标准化对于控制维护过程具有很大帮助。控制良好的过程可以重复且可预测，因此我们设计的系统必须采用统一的方式来完成各种维护任务。如果西南航空公司的维修人员使用不同的工具，执行不同的维修任务，则每个维修人员所交付的成果质量各不相同，完成任务所花费的时间也各不相同，这使得维护成本难以控制和管理。

简易性

操作和维护应尽可能简单。换句话说，确保能够轻松完成所需的工作。这可减少所需的文档记述和培训量，提高工作的一致性，并缩短维护所需的时间。

环境和人为因素

始终考虑环境和人为因素。例如，如果系统在多尘环境中使用，则可能需要在通风口上安装滤尘器。维修滤尘器是否容易？系统是否需要脚轮，以便可以移动，方便维护？如果是这样，请确保它们具有合适的重量且适合所处的地形。操作人员和维护人员的技能水平如何，需要多少培训？是否能以用户友好的方式设计硬件和软件接口？

设计原则	预测性维护	预防性维护	纠正性维护
自检和监测	■ 状态监测 ■ 功能验证	■ 功能验证	■ 故障检测 ■ 故障诊断和定位 ■ 功能验证
模块化设计	■ 状态监测 ■ 维护 ■ 更换 ■ 校准 ■ 功能验证	■ 维护 ■ 更换 ■ 校准 ■ 功能验证	■ 故障检测 ■ 故障诊断和定位 ■ 维修 ■ 功能验证
标准化	■ 状态监测 ■ 维护 ■ 更换 ■ 校准 ■ 功能验证 优化任务一致性	■ 维护 ■ 更换 ■ 校准 ■ 功能验证 ■ 优化任务一致性	■ 故障检测 ■ 故障诊断和定位 ■ 维修 ■ 功能验证 ■ 优化任务一致性
简易性	■ 降低文档记述和培训成本 ■ 优化任务一致性	■ 降低文档记述和培训成本 ■ 优化任务一致性	■ 降低文档记述和培训成本 ■ 优化任务一致性
环境和人为因素	■ 降低预测性维护事件的频率和/或缩短MPdMT ■ 减少人为失误 提高安全性	■ 降低预防性维护事件的频率和/或缩短MPMT ■ 减少人为失误 ■ 提高安全性	■ 降低故障率和/或缩短MTTR ■ 减少人为失误 ■ 提高安全性

表 1 该表格简要概括了每个设计原则在每种维护方法上的具体体现

维护策略

应该使用哪种方法？预测性维护策略会在检测到潜在的未来故障后，在方便的时间安排维护或更换系统组件。预防性维护策略会按一定的时间间隔定期对系统组件进行主动维护、更换和/或校准，以最大限度地降低故障风险和计划外停机的成本。纠正性维护策略会等到某个组件发生故障以最大限度地利用资本投资，然后尽快进行维修以尽量减少意外停机的成本，或尽量缩短MTTR。对于每种策略，您可以自己执行或与供应商签订服务协议，也可以不采取任何措施，在发生故障时持乐观态度，当然我们并不建议这样做。

下文介绍了不同的技巧组合，并解释了哪种维护策略最适合用于不同的子系统或组件。此处讨论的方法包括状态监测可行性、基于可靠性的维护(RCM)和故障成本分析。RCM方法需要了解运行时间对系统组件故障率和组件故障成本的影响。以下三幅图显示了三种策略对应的故障率与运行时间的函数关系。每幅图描绘了不同类型组件的特性。实际应用当然不止这三种场景，但这三种最为常见，可帮助您理解RCM的工作原理。

图5显示了故障率随时间推移而升高的情况。在这种情况下，组件的故障率可能一开始表现为恒定，但在系统达到预期使用寿命之前组件就开始出现损坏。换句话说，组件的使用寿命明显短于系统运行时长。这可能是最直观的场景，因为风扇、连接器、机电继电器、固态硬盘驱动器、电池、电子器件校准系统等机械组件都会呈现这种趋势。在每次预防性维护事件发生之后，故障率都会重新降至出厂水平，从而恢复系统的可靠性。

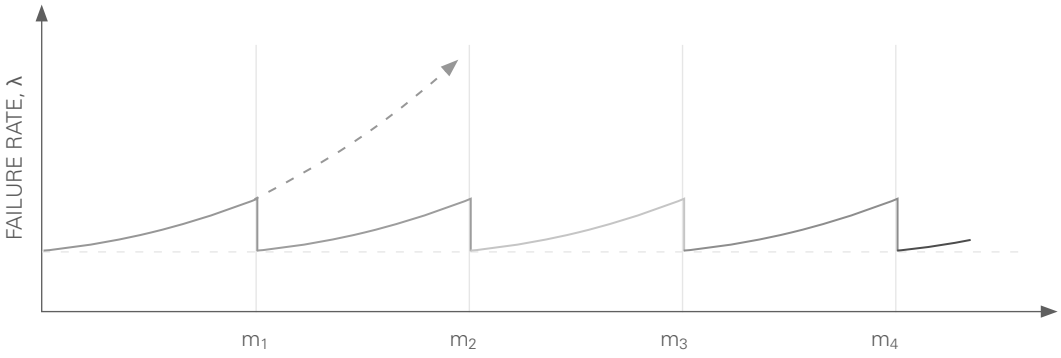


图5 预防性维护会在故障率升高时启动维护事件，使故障率重新降至出厂水平

图6显示了故障率随时间保持恒定的情况，这种故障率有时称为稳态故障率。在这种情况下，组件会在远远超出系统预期使用寿命(不包括校准)后才开始出现损坏。换句话说，组件的使用寿命远远超过系统运行时长。这一场景常见于电子器件，比如IC、电阻器、陶瓷电容器、二极管、电感器等。现代电子器件的使用寿命通常远远超过10至15年。实际上，在测试系统淘汰之前，它们不会出现损坏。

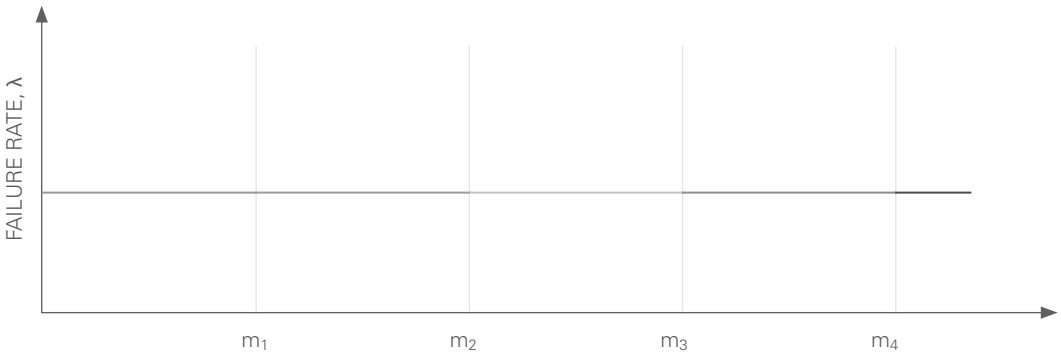


图 6 对于预防性维护，由于故障率随时间保持恒定，每个维护事件对故障率几乎没有影响或完全没有影响

在每次预测性维护事件发生之后，故障率不会发生改变，因此在组件发生故障之前更换组件并不会带来任何益处。从数学上来讲，这种故障率被视为“随机机会”。因此，使用新组件替换正在运行的旧组件并不能提高系统可靠性。

图7显示了故障率随时间推移而降低的情况。这可能是最不直观的场景，常见于软件和复杂的计算机系统。对软件和固件进行重大升级或添加新功能、新技术等可能会引入缺陷(错误)，从而增大系统发生故障的可能性。在每次预防性维护事件之后，故障率都会升高到更高的水平，降低系统的可靠性。但是，有些情况下我们必须升级软件，以应对操作系统更新或硬件过时等情况。

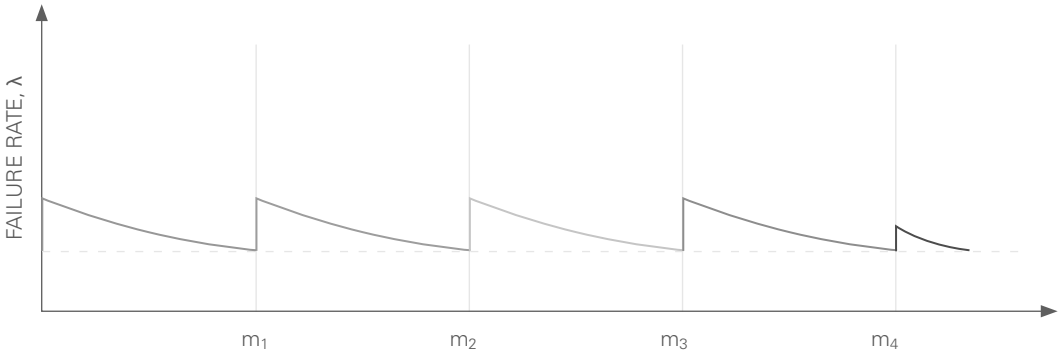


图 7 故障率随时间下降，预防性维护实际上在每次维护事件之后都提高了故障率

此外，在某些情况下，并无足够的数据来了解故障率是随时间升高、保持恒定还是降低。这常见于新产品、新技术或新设计。使用预测性维护策略来监测故障随时间的变化有助于了解组件的情况，前提是监测的成本相比故障的成本更具效益。即使趋势尚未确定，预测性维护策略通常也会最大限度地提高资本投资收益，同时尽量降低停机成本。

使用此方法为整个系统制定维护策略时，可以将系统分解为多个子系统和/或组件，然后对每个组件进行评估，确定最佳的维护策略。

以下提供了一些有用的指导原则：

- 能否在组件故障导致系统故障之前就检测出故障的发生？
  - 考虑到纠正性维护事件的故障成本和预测性维护事件的额外计划停机时间，对该组件故障进行状态监测是否具有成本效益？
- 该组件的故障率是随时间升高、保持恒定还是降低，或者您对此是否了解？
  - 故障是否严重，故障成本是否很高？

下图显示了一个决策流程图，可帮助您为系统的每个组件和故障模式选择最佳策略。当然，这个流程图应根据实际需要进行调整。

ATE系统示例

基于PXI Express的ATE系统由基本组件或子系统组成，每个组件或子系统都可以分解为更小的组件，以实现自己的维护策略。

机箱

机箱背板可能难以通过监测来发现潜在故障。它的故障率恒定，使用寿命为10至15年甚至更长。电子器件基本上都是数字的，不需要校准，最适合采用纠正性维护策略或“运行至故障”方法。

此示例中的机箱电源不提供监测功能。电源通常使用较大的液体电容器，有些还配有冷却风扇。根据负载和环境条件，这些组件的典型使用寿命约为7至10年。采用预测性或预防性维护策略较为合适。

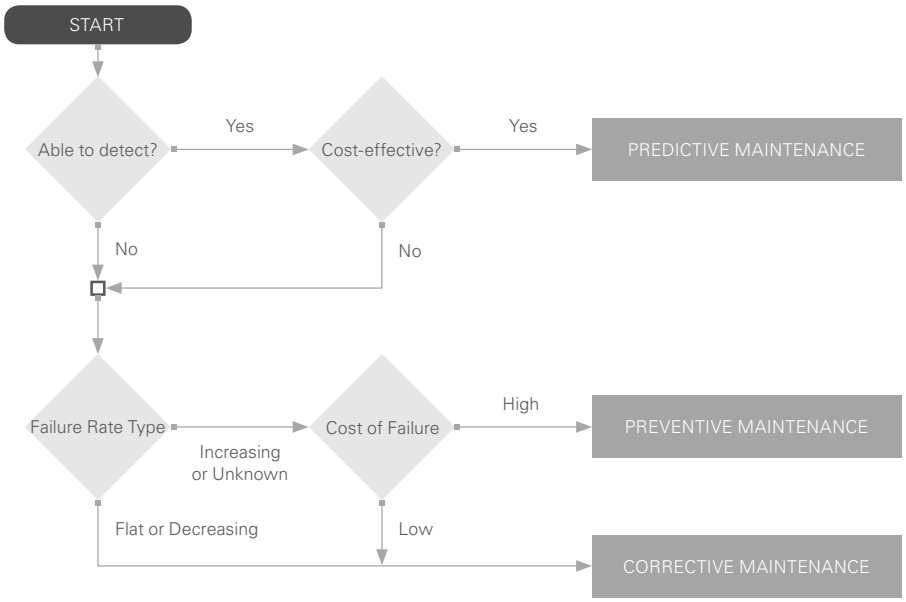


图 8 维护策略决策流程图

机箱风扇速度和机箱温度可以监测。如果速度开始减慢或者机箱温度开始升高，则会发出警告，可以在近期方便的时间安排维护。这一场景适合采用预测性维护策略。

控制器

控制器的集成电路和电子部件(不包括硬盘和RAM)可以提供监测工具来识别潜在故障。本例中，实现这些功能需要耗费大量的开发时间，并且不具有成本效益。此外，控制器的故障率恒定，典型的使用寿命为10至15年甚至更长。电子器件基本上都是数字的，不需要校准，通常最适合采用纠正性维护策略或“运行至故障”方法。

控制器的RAM具有可自动运行的错误校正码(ECC)，可以通过监测找到并纠正错误。如果这些错误的频率持续增加，则可能需要安排时间对RAM进行更换。RAM不需要校准，最适合采用预测性维护策略。

在此示例中，控制器的硬盘驱动器是一个固态硬盘驱动器(SSD)，可监测读写次数。SSD经过一定次数的读写后寿命就会出现损坏。因此，当读写次数接近使用寿命时，应安排更换SSD。SSD不需要校准，最适合采用预测性维护策略。

软件具有一些独特的特性：它不会出现损坏，并且不受环境的影响，只会因为设计缺陷或错误而出现故障。我们可以监测资源泄漏，如内存使用和碎片化；然而，许多故障在崩溃之前是无法发现的。软件的一个优点是不会出现损坏，只需重启系统就会正常工作。这样就解决了所有问题，直至错误再次导致软件崩溃。因此，采用预防性维护策略，每周或每月重启一次软件就可以解决许多问题。影响软件可靠性的一个更具挑战性的因素是软件升级。软件偶尔需要升级，因为需要安装新功能，需要与其他软件包兼容，或者可能需要补丁来修复错误。问题是，每次引入新软件就会改变整个生态系统，从而可能会引入更多错误，但只有升级之后才会知道结果如何。这种动态变化使得软件升级后软件故障风险立即上升，然后在运行一段时间后稳定下来。软件最常用的升级维护方法是延迟升级，需要时才升级。

仪器模块

仪器模块上的集成电路可能难以通过监测来发现潜在故障。它们的故障率恒定，使用寿命为20年或更长。模拟电子器件可能随时间发生漂移，因此需要校准。为解决漂移问题，需要采取预防性维护策略来进行校准。许多校准实验室可以在校准后对模块进行最终验证测试，以证明一切正常。该测试可以有效地找出已发生故障或濒临故障的其他电子元件。但是没有测试是完美的，对于电子器件的部分其他故障模式，可能适合采用纠正性维护策略或“运行至故障”方法。此时，最适合采用组合策略。

开关模块

开关模块的基板主要由集成电路组成，这些集成电路通常不具备用于监测电子器件健康状况的工具。开关模块的故障率恒定，典型使用寿命为10至15年甚至更长。电子器件基本上都是数字的，不需要校准，最适合采用纠正性维护策略或“运行至故障”方法。

开关的机电继电器具有监测操作次数的工具。继电器经过一定次数的操作后就会出现损坏，具体取决于所开关的电气负载。您可以使用制造商提供的数据和公式来估算开关的次数。因此，当操作次数接近使用寿命时，应安排更换开关模块。开关模块不需要校准，最适合采用预测性维护策略。

电缆

固定线缆基本上保持连接状态，从不断开，或者在极少情况下需要重新连接，不会产生任何影响。固定电缆几乎不会出现故障，除非受振动干扰或人为滥用。其故障率恒定并且非常低。因为，最适合采用纠正性维护策略。

动态电缆经常连接和断开，在经过一定次数的重新连接后就会出现损坏。其故障率随时间推移而升高，并且检测出潜在故障可能并不容易，但故障率可以估计。所允许的重新连接次数可以询问制造商。如果重新连接的平均次数已知，并且了解每小时、每天、每个单元需要多少次重新连接，那么就可以安排预防性维护。在这种情况下，最适合采用预防性维护策略。

子组件	预测性维护	预防性维护	纠正性维护
机箱背板	-	-	√
机箱电源	-	√	-
机箱风扇	√	-	-
控制器主板	-	-	√
控制器RAM	√	-	-
控制器固态硬盘驱动器	√	-	-
控制器软件	-	√	-
仪器模块	-	校准	√
开关模块基板	-	-	√
开关模块继电器	√	-	-
固定线缆	-	-	√
动态线缆	-	√	-

表2 此维护策略适用于基于PXI Express的ATE的每个主要组件。请注意，每个组件的最佳策略因应用的具体情况而异

结论

预测性维护、预防性维护和纠正性维护各有其优势、挑战和适用情况。在大多数情况下，与维护相关的最大一项支出是计划外停机成本(故障成本)。通过状态监测和预测将计划外停机转换为计划内停机通常是有好处的。

每年，状态监测设备、网络、服务器和大模拟数据(Big Analog Data™)分析都会不断降低成本并提高性能，因此行业趋向于采用更智能的设备和预测性能更高的维护策略。对于计划外停机无法避免的情况，有效的备件和维修策略是管理并最大限度地降低维护成本的关键。

在设计中融入可维护性的系统与完善的维护策略相结合，将帮助您管理故障成本和降低故障风险，避免昂贵的计划外停机，从而降低维护成本和总拥有成本。若要在设计阶段就考虑尽量提高系统可维护性，自检功能、模块化设计、标准化、简单性和环境/人为因素是基本的考量因素。

附录：维护成本

许多公司在采购测试设备时主要根据价格来作出决策，而不考虑部署、操作和维护设备的成本。他们甚至很少考虑设备停机的成本。在测试系统的整个生命周期内，停机(或故障)和维护成本可能远高于购买价格，通常会高出两到三倍。最大的罪魁祸首是停机或故障成本。这就是为什么需要制定维护计划以及系统的可维护性变得越来越重要的原因。

本附录提供了一个简单的总维护成本(TCM)模型，可用于估算系统在其使用寿命内的潜在停机和维护成本。计算测试系统的TCM可能会非常繁琐和复杂。该模型基于一定的复杂程度和详细程度进行了充分的估算，对于大多数应用来说都足以满足要求且易于管理。

总维护成本(TCM)

TCM = CD + M

CD = 计划外停机和计划内停机成本

M = 维护成本

将投资的维护费用(M)与在系统生命周期内减少的停机成本(CD)或者TCM成本的总减少量进行比较，就可以衡量维护计划的投资回报率(ROI)。有些公司将计划内停机的成本与维护成本相结合，并将其与计划外停机时间的成本进行比较，因为他们的主要重点是避免计划外停机和故障。每个公司可能有自己的方法来估算TCM和维护的ROI，具体取决于公司想要跟踪的指标。

停机成本(CD)

停机成本有时看起来像是“运气”钱，因为有些公司发现这种成本很难估算。但停机成本是真实存在的。停机有两种类型：计划内(安排的)和计划外(未安排的)停机。维护计划的目标是尽可能减少所有停机，并在经济上可行的条件下将尽可能多的计划外停机转换为计划内停机。

计划外停机的成本始终是最高的，因为这种停机发生在您需要使用设备时。计划外停机永远不会在适当的时机发生，并且可能由于生产中断、产品损失、对其他设备的附带损坏、劳动力损失(劳动力可能不得不“坐等”系统修复)，以及其他依情况而定的物流成本而造成巨额的利润损失。一些制造公司估计其计划外停机的成本约为每小时8,000美元。石化、电力和运输公司估计的每小时停机成本就更高了。计划外停机成本因产品、情况、公司和行业而异。时间就是金钱；所以我们需要采用包含适当备件策略的纠正性维护计划，最大程度地缩短故障系统的平均修复时间(MTTR)。

计划内停机的成本同样很高，但低于计划外停机成本，因为计划内停机安排在对生产影响最低的时段，可最大程度地减小产量损失及对其他设备造成的附带损坏风险，不会造成劳动力损失，并最大限度地降低物流成本(因为经过培训的人员、工具和部件都在现场，随时可进行维护)。计划内停机的持续时间相比计划外停机可能更短，并且其成本可以分摊到需要维护的许多其他系统。由于计划外停机的成本通常高于计划内停机，许多公司已经实施了预测性和预防性维护计划。

CD = UD + PD

UD = 计划外停机成本

PD = 计划内停机成本

$UD = \lambda \times MTTR \times T_U \times$ 每小时成本

λ = 稳态故障率(每小时故障次数)

系统的稳态故障率是指在系统的生命周期或使用寿命内预期的故障率。稳态故障率阶段在系统生命周期中介于使用初期(系统老化)和寿命耗尽阶段之间，在系统寿命耗尽时，系统故障率预计将显著升高，系统应退役。以下数学关系适用于系统生命周期中故障率达到稳定状态时的阶段。

$$\Lambda = \frac{1}{MTBF_{系统}}$$

$MTBF_{系统}$ = 系统故障平均间隔时间(小时)

T_U = 系统在整个生命周期的总运行时间(小时)

电子器件的运行时间通常包含系统在执行工作和处于空闲状态时的上电时间。

MTTR = 平均修复时间(小时)

MTTR不仅仅是维修或更换故障组件的时间。它还包括：

- 检测故障所需的时间
- 对系统进行诊断并确定哪些系统组件出现故障所需的时间
- 检查、维修或更换故障组件所需的时间(在很大程度上受到有无备件和/或多余组件的影响)
- 验证系统恢复正常工作所需的时间
- 系统重新联机所需的时间

显然，MTTR在很大程度上取决于备件可用性、系统位置、设计和通常发生的故障类型。

$$MTTR = \frac{\Sigma(\Lambda_i \ T_i)}{\Sigma \ \Lambda_i}$$

λ_i = 第i种故障模式的故障率

t_i = 发生第i种故障模式后的系统维修时间

故障模式定义为发生的故障类型或故障的根本原因。

$PD = (\lambda \times MPdMT + f_{PM} \times MPMT) \times T_U \times$ 每小时的计划内停机时间

预测性维护的频率应与系统的故障率相关。预测性维护不是在发生故障之后执行，而是在检测到潜在故障条件之后，在故障发生之前的某个时间按计划执行。

MPdMT = 平均预测性维护时间(小时)

MPdMT包括：

- 检修所需时间
- 维修和/或更换组件所需的时间(在很大程度上受到有无备件和/或多余组件的影响)
- 验证系统是否正常运行所需的时间
- 系统重新联机所需的时间

$$MPdMT = \frac{\Sigma(\Lambda_i \ T_i)}{\Sigma \ \Lambda_i}$$

λ_i = 第i种预测性维护活动的频率

t_i = 对系统执行第i种预测性维护活动所需的时间

f_{PM} = 预防性维护的频率(每小时)

MPMT = 平均预防性维护时间(小时)

MPMT包括：

- 检修所需时间
- 维修、更换和/或校准组件所需的时间(在很大程度上受到有无备件和/或多余组件的影响)
- 验证系统是否正常运行所需的时间
- 系统重新联机所需的时间

$$MPMT = \frac{\Sigma(f_i \ T_i)}{\Sigma \ f_i}$$

f_i = 第i种预防性维护活动的频率

t_i = 对系统执行第i种预防性维护活动所需的时间

维护成本(M)

M = PdM + PM + CM

PdM = 预测性维护成本

PM = 预防性维护成本

CM = 纠正性维护成本

预测性维护成本(PdM)

$PdM = \Lambda \times T_U \times$ PdM 事件 + 工具成本

PdM 事件= PdM事件的平均成本

PdM事件= (MPdMT ×每小时计划内停机的劳动力成本) + 维护或更换成本 + 备件成本 + 物流成本

计划内停机的劳动力成本包括对系统执行预测性或预防性维护所需的劳动力成本以及按小时估算的劳动力培训成本。

工具成本= PdM所需的软件和硬件工具成本

工具成本通常为一次性支出，包括：

- 状态监测软件和硬件成本
- 拆卸和更换组件所需的工具成本
- 测试设备和软件验证成本(可能与纠正性维护一样)
- 设备和软件维护成本

注意：这些工具通常可用于预测性、预防性和纠正性维护。如果可以使用工具，则工具成本只需计入一次，而不是三种类型中每种维护都要计入一次。

如前文所述，MPdMT主要受到是否有合适的设备/工具以及操作人员的技能水平的影响。

预防性维护成本(PM)

$$PM = f_{PM} \times T_U \times PM\ 事件 + 工具成本$$

f_{PM} = 预防性维护的频率(每小时)

PM事件 = PM事件的平均成本

PM事件 = (MPMT × 每小时计划内停机的劳动力成本) + 校准、维护或更换成本 + 备件成本 + 物流成本

系统的MPMT越小，预测性维护的成本就越低。如前文所述，MPMT主要受到是否有合适的设备/工具、操作人员技能水平以及校准策略的影响。许多系统供应商都提供各种校准选项。根据具体情况，现场校准服务或向系统供应商购买校准服务协议可能更具成本效益。一个标准的供应商校准项目一般就足以满足要求。

纠正性维护成本(CM)

$$CM = \Lambda \times T_U \times CM\ 事件 + 工具成本$$

CM事件 = CM事件的平均成本

CM事件 = (MTTR × 每小时计划外停机的劳动力成本) + 维修或更换成本 + 备件成本 + 物流成本

计划外停机的劳动力成本包括系统维修所需的劳动力成本和按小时估算的劳动力培训成本。

系统的MTTR越小，系统可用性就越高，计划外停机的成本也就越低。如前文所述，MTTR主要受位置、系统设计、是否有合适的设备/工具、人员技能水平和备件策略的影响。许多系统供应商都提供各种备件选项。根据具体情况，配备备件或向系统供应商购买服务协议以提供备件，或购买两者兼具的一些混合协议可能更具成本效益。如果计划外停机的成本足够低，则可能就不需要配备备件，只需依赖供应商的标准维修服务可能就足以满足需求。