Virtual Instrumentation

What is Virtual Instrumentation?

Virtual instrumentation is the use of customizable software and modular measurement hardware to create user-defined measurement systems.

Why is Virtual Instrumentation necessary?

Virtual instrumentation is necessary because it delivers instrumentation with the rapid adaptability required for today's concept, product, and process design, development, and delivery.

Only with virtual instrumentation can engineers and scientists create the user-defined instruments required to keep up with the world's demands. The only way to meet these demands is to use test and control architectures that are also software centric.

Because virtual instrumentation uses highly productive software, modular I/O, and commercial platforms, it is uniquely positioned to keep pace with the required new idea and product development rate.

INTRODUCTION

The term "scientific computing," or "computational science" refers to the use of computers in solving scientific problems.

Scientific computing applications usually follow a three-step process:

1. data acquisition
2. data analysis
3. data visualization/presentation.

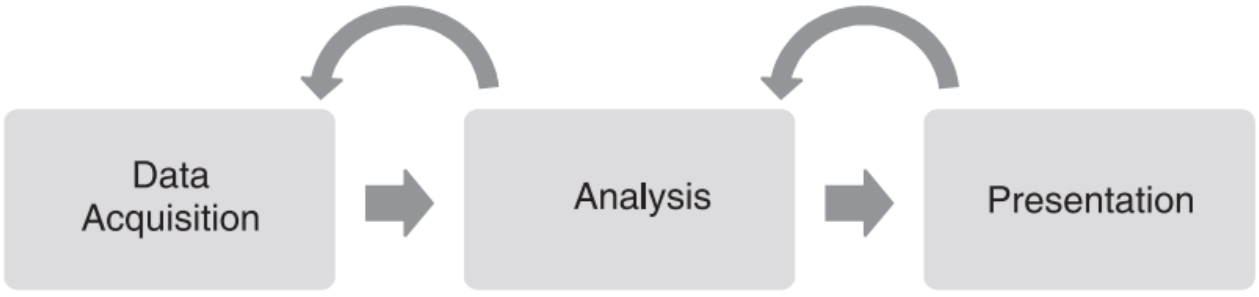

Figure 1.1 Virtual instrumentation model.

The original conceptualization in the early 1980s and it has been expanded into a more comprehensive model known as graphical system design shown in Figure 1.2

Figure 1.2 Graphical system design model.

This model focuses is to accelerate the research and development cycle, delivering mathematical models to embedded real-time computers faster and easier.

This design flow acceleration is achieved by using NI LabVIEW software and its G programming language as a common system-level design tool for all the different phases in the design-to-deployment flow.

GRAPHICAL SYSTEM DESIGN (GSD) MODEL

Figure 1.2 Graphical system design model.

In reality, the virtual instrumentation model is applied in each of the three phases of the graphical system design model as shown in Figure 1.3.

In the graphical system design model, data acquisition, analysis and presentation functions are used in the design.

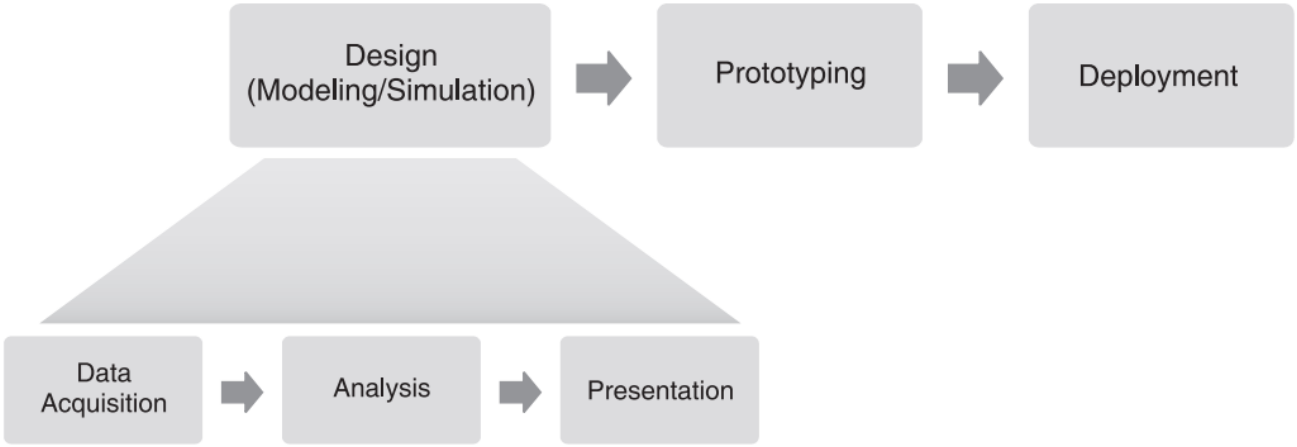

Figure 1.3 Graphical system design and virtual instrumentation.

DESIGN MODEL in GRAPHICAL SYSTEM DESIGN (GSD) MODEL

The design phase as shown in Figure 1.4

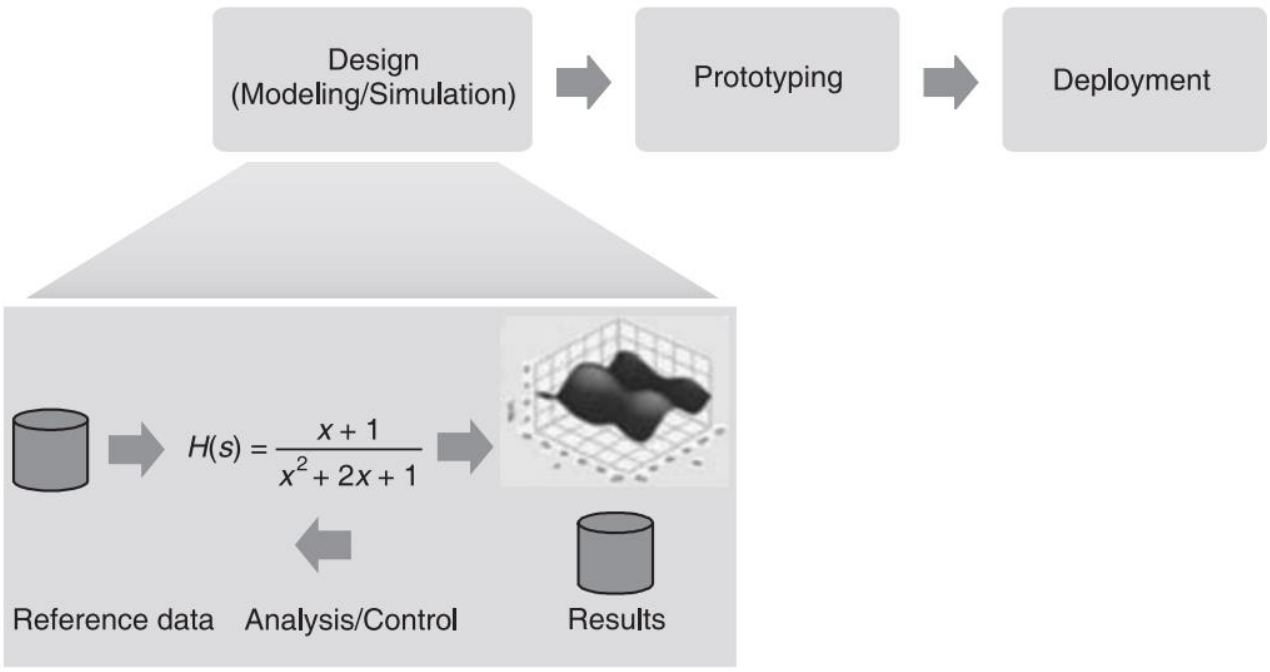

Figure 1.4 The design phase of the graphical system design model.

In the design phase, the researcher develops a mathematical model of the system, including sensors, actuators, plants and controllers, and simulates them under a variety of initial conditions and constraints.

In the design phase,The researcher uses different numerical methods with the objective of validating the performance of the model and optimizing it.

In this phase, researchers can acquire reference data from files or databases and incorporate it into the model.

In this phase, A “virtual plant/process” is created, which can be used later for hardware-in-the loop (HIL) tests. Results from the simulation process are saved for post-analysis and visualization and can be used to introduce changes into the model. This is usually a software-centric process with a strong focus on numerical methods/analysis and mathematics.

for complex or computationally intensive models, high-performance computing (HPC) using grid computers, standard computers with graphical processing units (GPUs), and multicore based computers is a key factor.

In those cases, the hardware has an important impact on the performance of the model solution and simulation. Multicore-ready software tools **LabVIEW** provides the user with a powerful yet easy-to-use programming language that can take advantage of multicore processors and parallel programming.

PROTOTYPE (Lab) in GRAPHICAL SYSTEM DESIGN (GSD) MODEL

The prototype phase is shown in Figure 1.5

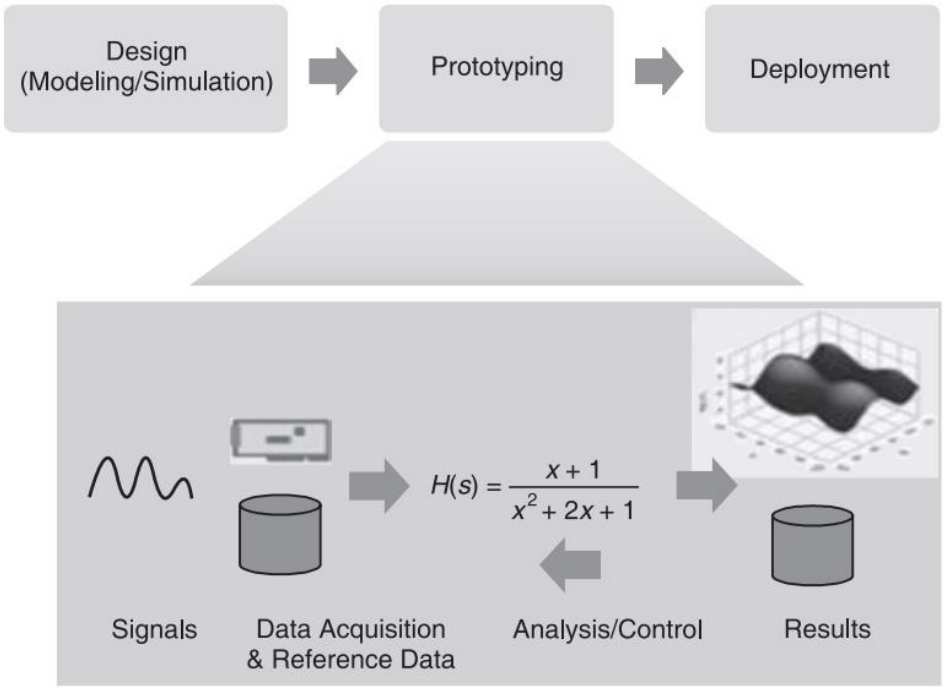

Figure 1.5 The prototyping phase of the graphical system design model.

If experimental validation of the model is required, researchers develop and test a prototype in the laboratory.

In this phase, Signal processing and analysis as well as visualization can be implemented online while data is being measured and acquired, or while the process is being controlled.

The "virtual plant/process" defined in the design phase can be used for HIL tests. The experimental results obtained in this phase can be used to modify and optimize the original model, which in turn may require additional experiments.

.

The prototyping phase is executed on standard PCs or PXI computers, using PCI/PXI data acquisition devices or external measuring devices connected to a PC via USB, Ethernet, GPIB, or serial ports.

The prototyping phase is usually more software/hardware-centric because sensors, actuators, data acquisition devices, controllers, and the controlled/analyzed plant itself are all key elements in the experimental setup.

The 2D or 3D differential equations can be solved in real time, or large sets of matrices can be processed at high speed and in real time using parallel programming in multicore computers while sharing part of the signal processing load with an FPGA or a GPU.

Data is shared between the multicore processors and an FPGA via DMA first-in-first-out memory buffers (FIFOs).

Finally, technical data mining algorithms can be implemented by combining LabVIEW with NI DIAdem.

NI DIAdem is an excellent tool for managing, analyzing, and reporting technical data collected during data acquisition and/or generated during the modelling, simulation or analysis phases.

DEPLOYMENT (Field) in GRAPHICAL SYSTEM DESIGN (GSD) MODEL

The Deployment phase is shown in Figure 1.5

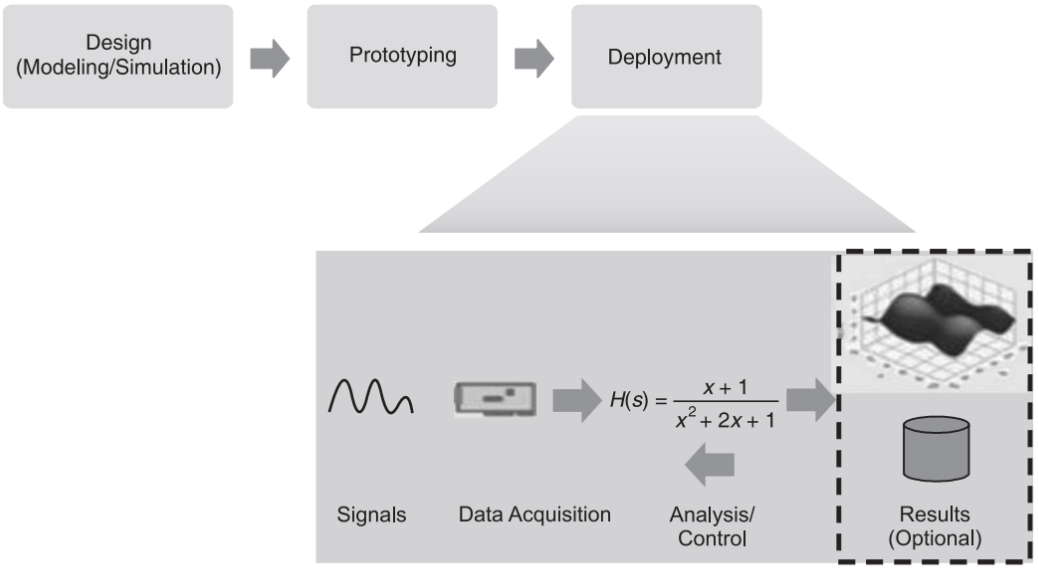

Figure 1.6 The deployment phase of the graphical system design model.

The model is deployed in the field or lab using either a PC or PXI(PCI eXtensions for Instrumentation), or it can be downloaded to a dedicated embedded controller.

Using the LabVIEW Real-Time Module, symmetric, multiprocessing (SMP) techniques can be easily applied.

The transition from the prototyping phase to the deployment phase can be very fast and efficient because the same set of tools used for prototyping can, in most cases, be applied to the final deployment of the system in the field.

In many cases, the system is deployed as a headless system, executing the analysis/control algorithms in real time as a dedicated device.

If on-the-field graphical user interfaces (GUIs) or operator interfaces (OIs) are needed, the LabVIEW TouchPanel Module and industrial-grade monitors with touch-sensitive screens can be used locally and remotely, data can be shared and visualized via Ethernet with applications developed using LabVIEW.

DESIGN FLOW WITH GSD

The typical embedded system software and hardware design flows is shown in Figure 1.7.

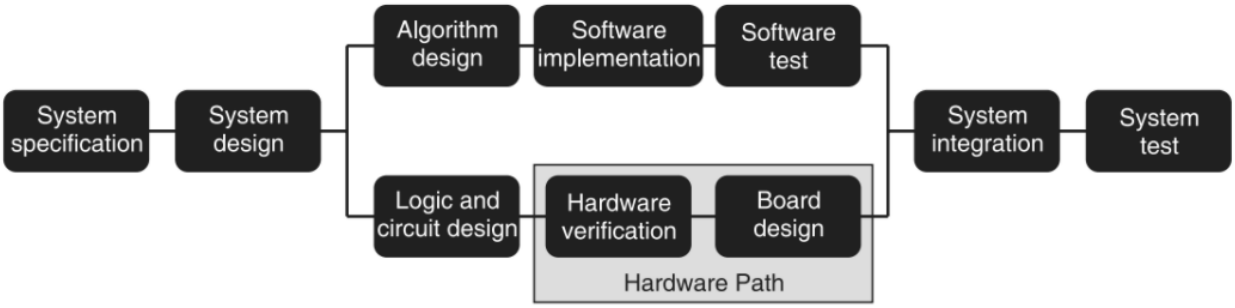

Figure 1.7 Typical embedded system software and hardware design flow.

If you are creating custom hardware for final deployment, it is difficult to have the software and hardware developed in parallel as the software is never tested on representative hardware until the process reaches the system integration step.

We do not want the software development to be purely theoretical, because waiting until the system integration test to include I/O and test the design with real signals may mean that you discover problems too late to meet design deadlines.

Designers currently use a solution like an evaluation board to prototype their systems. However, these boards often only include a few analog and digital I/O channels and rarely include vision, motion, or ability to synchronize I/O.

Additionally, designers often have to waste time developing custom boards for sensors or specialized I/O, just to complete a proof of concept.

Using flexible Commercial-off-the-shelf (COTS) prototyping platforms instead can truly streamline this process, as shown in Figure 1.8.

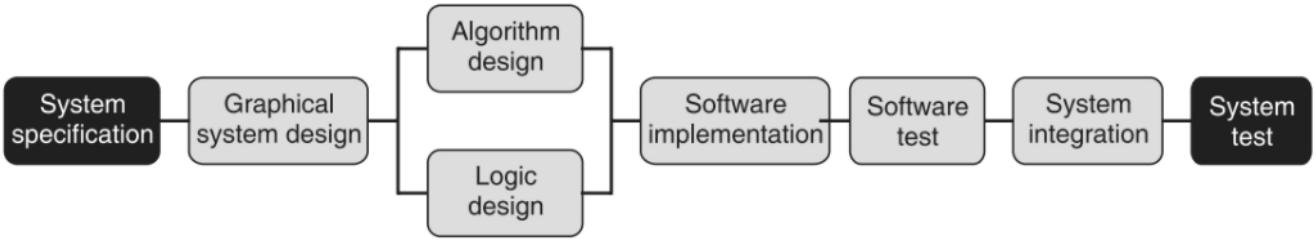

Figure 1.8 Stream-lined development flow with graphical system design.

Stream-lined development flow with graphical system design eliminates much of the work required for hardware verification and board design. Much like PCs today, in which case anyone can go to an electronics store and plug components such as memory, motherboards and peripherals together to create a PC, graphical system design strives to achieve the same standardization for prototyping platforms.

For most systems, a prototyping platform must incorporate the same components of the final deployed system. These components are often a real-time processor for executing algorithms deterministically, programmable digital logic for high-speed processing or interfacing the real time processor to other components, and varied types of I/O and peripherals as shown in Figure 1.9.

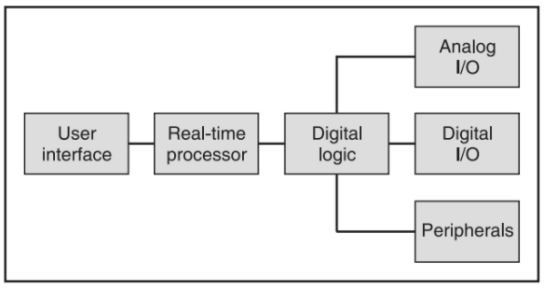

Figure 1.9 Typical components of an embedded system.

Finally, as with any system, if the off-the-shelf I/O doesn't serve all of your needs, the platform should also be extensible and customizable when needed.

VIRTUAL INSTRUMENTATION

Virtual instrumentation as shown in Figure 1.10

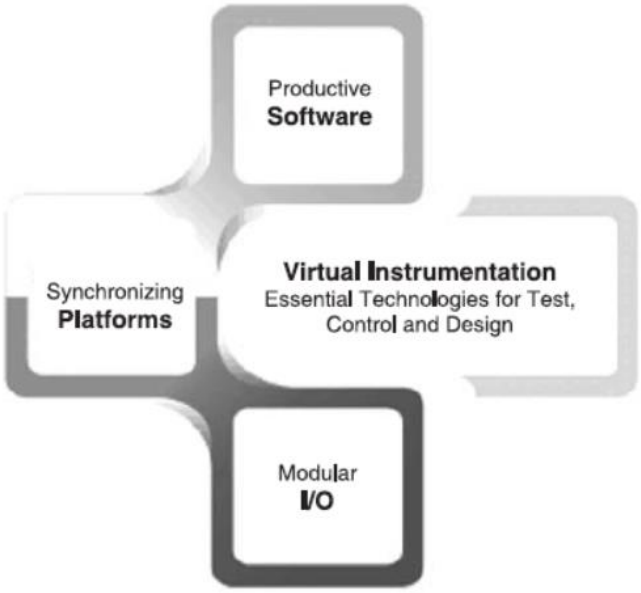

Figure 1.10 Virtual instrumentation combines productive software, modular I/O and scalable platforms.

Virtual instrumentation combines mainstream commercial technologies, such as the PC, with flexible software and a wide variety of measurement and control hardware.

Engineers use virtual instrumentation to bring the power of flexible software and PC technology to test, control and design applications making accurate analog and digital measurements.

Engineers and scientists can create user-defined systems that meet their exact application needs.

Industries with automated processes, such as chemical or manufacturing plants use virtual instrumentation with the goal of improving system productivity, reliability, safety, optimization and stability.

Virtual instrumentation is computer software that a user would employ to develop a computerized test

and measurement system for controlling from a computer desktop, an external measurement hardware device, and for displaying, test or measurement data collected by the external device on instrument-like panels on a computer screen.

Virtual instrumentation as shown in Figure 1.10 uses highly productive software, modular I/O and commercial platforms.

National Instruments LabVIEW, a premier virtual instrumentation graphical development environment, uses symbolic or graphical representations to speed up development.

The software symbolically represents functions. Consolidating functions within rapidly deployed graphical blocks further speeds up development.

Another virtual instrumentation component is modular I/O, designed to be rapidly combined in any order or quantity to ensure that virtual instrumentation can both monitor and control any development aspect.

The third virtual instrumentation element using commercial platforms, often enhanced with accurate synchronization, ensures that virtual instrumentation takes advantage of the very latest computer capabilities and data transfer technologies. This element delivers virtual instrumentation on a long-term technology base that scales with the high investments made in processors, buses and more

innovation mandates use of software to accelerate a new concept and product development, it also requires instrumentation to rapidly adapt to new functionality. Because virtual instrumentation applies software, modular I/O and commercial platforms, it delivers instrumentation capabilities uniquely qualified to keep pace with today's concept and product development

VIRTUAL INSTRUMENT AND TRADITIONAL INSTRUMENT

A traditional instrument is designed to collect data from an environment, or from a unit under test, and to display information to a user based on the collected data.

Such an instrument may employ a transducer to sense changes in a physical parameter such as temperature or pressure, and to convert the sensed information into electrical signals such as voltage or frequency variations.

The term "**instrument**" may also cover a physical or software device that performs an analysis on data acquired from another instrument and then outputs the processed data to display or recording means. This category of instruments includes oscilloscopes, spectrum analyzers and digital millimeters.

The types of source data collected and analyzed by instruments may thus vary widely,including both

- **Physical parameters** such as temperature, pressure, distance, light and sound frequencies and amplitudes,

and also

- **Electrical parameter**s including voltage, current and frequency.

A virtual instrument (VI) is defined as an industry-standard computer equipped with user-friendly

application software, cost-effective hardware and driver software that together perform the functions of traditional instruments.

Virtual instrumentation software based on user requirements defines general-purpose measurement and control hardware functionality.

With virtual instrumentation, engineers and scientists reduce development time, design higher quality products, and lower their design costs.

Virtual instrumentation is necessary because it is flexible. It delivers instrumentation with the rapid adaptability required for today's concept, product and process design, development and delivery.

Virtual instrumentation is necessary because it is flexible. It delivers instrumentation with the rapid adaptability required for today's concept, product and process design, development and delivery.

To meet the ever-increasing demand to innovate and deliver ideas and products faster, scientists and engineers are turning to advanced electronics, processors and software. Consider modern cell phones. Most of them contain the latest features of the last generation, including audio, a phone book and text messaging capabilities. New versions include a camera, MP3 player, and Bluetooth networking and Internet browsing.

Virtual instruments are defined by the user while traditional instruments have fixed vendor-defined functionality.

In a conventional instrument, the set of components that comprise the instrument is fixed and permanently associated with each other. Nevertheless, there is some software that understands these associations. Thus, the primary difference between a virtual instrument and a conventional instrument is merely that the associations within a virtual instrument are not fixed but rather managed by software.

Every virtual instrument consists of two parts—software and hardware.

A virtual instrument typically has a sticker price comparable to and many times less than a similar traditional instrument for the current measurement task. However, the savings compound over time, because virtual instruments are much more flexible when changing measurement tasks. By not using vendor-defined, prepackaged software and hardware, engineers and scientists get maximum user-defined flexibility.

A traditional instrument provides them with all software and measurement circuitry packaged into a product with a finite list of fixed-functionality using the instrument front panel.

A virtual instrument provides all the software and hardware needed to accomplish the measurement or control task. In addition, with a virtual instrument, engineers and scientists can customize the acquisition, analysis, storage, sharing and presentation functionality using productive, powerful software.

Without the displays, knobs and switches of conventional, external box-based instrumentation products, a virtual instrument uses a personal computer for all user interaction and control. In many common measurement applications, a data acquisition board or card, with a personal computer and software, can be used to create an instrument. In fact, a multiple-purpose virtual instrument can be made by using a single data acquisition board or card. The primary benefits of applying data acquisition technology to configure virtual instrumentation include costs, size, and flexibility and ease of programming. The cost to configure a virtual instrumentation-based system using a data acquisition board or cards can be as little as 25% of the cost of a conventional instrument.

Traditional instruments and software-based virtual instruments largely share the same architectural

components, but radically different philosophies as shown in Figure 1.11.

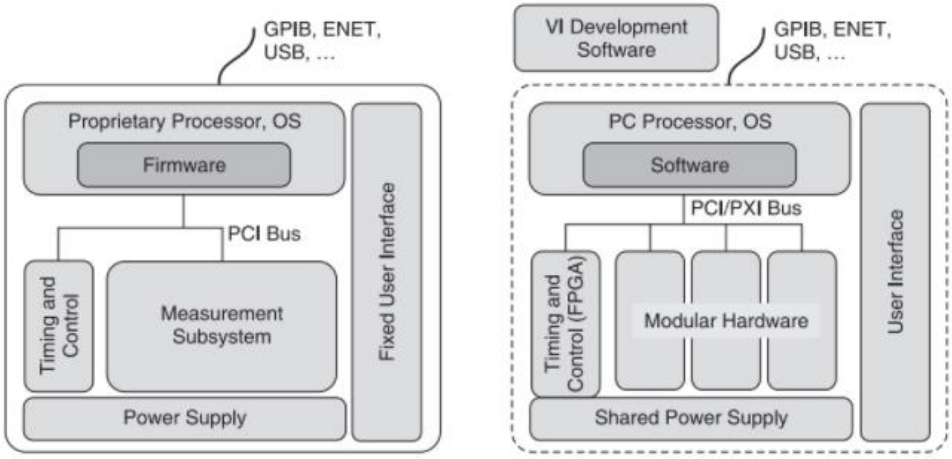

Figure 1.11 Traditional instruments (left) and software based virtual instruments (right).

Conventional instruments as compared to a virtual instrumentation can be very large and cumbersome. They also require a lot of power, and often have excessive amounts of features that are rarely, if ever used.

Most conventional instruments do not have any computational power as compared to a virtual instrument. Since the virtual instrument is part of a person computer configuration, the personal computer's computational as well as controlling capability can be applied into a test configuration.

Virtual instruments are compatible with traditional instruments almost without exception. Virtual instrumentation software typically provides libraries for interfacing with common ordinary instrument buses such as GPIB, serial or Ethernet.

Except for the specialized components and circuitry found in traditional instruments, the general architecture of stand-alone instruments is very similar to that of a PC-based virtual instrument. Both require one or more microprocessors, communication ports (for example, serial and GPIB), and display capabilities, as well as data acquisition modules. What makes one different from the other is their flexibility and the fact that we can modify and adapt the instrument to our particular needs.

A traditional instrument might contain an integrated circuit to perform a particular set of data processing functions; in a virtual instrument, these functions would be performed by software running on the PC processor. We can extend the set of functions easily, limited only by the power of the software used.

By employing virtual instrumentation solutions, we can lower capital costs, system development costs, and system maintenance costs, while improving time to market and the quality of our own products.

Virtual instrumentation has achieved mainstream adoption by providing a new model for building measurement and automation systems. Keys to its success include rapid PC advancement; explosive low-cost, high-performance data converter (semiconductor) development; and system design software emergence. These factors make virtual instrumentation systems accessible to a very broad base of users.

Virtual instruments take advantage of PC performance increase by analyzing measurements and solving new application challenges with each new-generation PC processor, hard drive, display and I/O bus.

These rapid advancements combined with the general trend that technical and computer literacy starts early in school, contribute to successful computer-based virtual instrumentation adoption.

The virtual instrumentation driver is the proliferation of high-performance, low-cost analog-to-digital (ADC) and digital-to-analog (DAC) converters. Applications such as wireless communication and high-definition video impact these technologies relentlessly. Virtual instrumentation hardware uses widely available semiconductors to deliver high-performance measurement front ends.

Finally, system design software that provides an intuitive interface for designing custom instrumentation systems furthers virtual instrumentation. Various interface standards are used to connect external devices to the computer. The PC is the dominant computer system in the world today.

HARDWARE AND SOFTWARE IN VIRTUAL INSTRUMENTATION

Role of Hardware in Virtual Instrumentation

Input/Output plays a critical role in virtual instrumentation. To accelerate test, control and design, I/O hardware must be rapidly adaptable to new concepts and products.

Virtual instrumentation delivers this capability in the form of modularity within scalable hardware platforms.

Virtual instrumentation is software-based; if we can digitize it, we can measure it. Standard hardware platforms that house the I/O are important to I/O modularity.

Laptops and desktop computers provide an excellent platform where virtual instrumentation can make the most of existing standards such as the USB, PCI, Ethernet, and PCMCIA buses.

Role of Software in Virtual Instrumentation

Software is the most important component of a virtual instrument. With the right software tool, engineers and scientists can efficiently create their own applications by designing and integrating the routines that a particular process requires.

You can also create an appropriate user interface that best suits the purpose of the application and those who will interact with it. You can define how and when the application acquires data from the device, how it processes, manipulates and stores the data, and how the results are presented to the user.

With powerful software, we can build intelligence and decision-making capabilities into the instrument so that it adapts when measured signals change inadvertently or when more or less processing power is required.

An important advantage that software provides is modularity.

When dealing with a large project, engineers and scientists generally approach the task by breaking it down into functional solvable units. These subtasks are more manageable and easier to test, given the reduced dependencies that might cause unexpected behaviour.

A virtual instrument is not limited or confined to a stand-alone PC. In fact, with recent developments in networking technologies and the Internet, it is more common for instruments to use the power of connectivity for the purpose of task sharing. Typical examples include supercomputers, distributed monitoring and control devices, as well as data or result visualization from multiple locations.

Every virtual instrument is built upon flexible, powerful software by an innovative engineer or scientist applying domain expertise to customize the measurement and control application. The result is a user-defined instrument specific to the application needs.

Virtual instrumentation software can be divided into several different layers like the application software, test and data management software, measurement and control services software as shown in Figure 1.12.

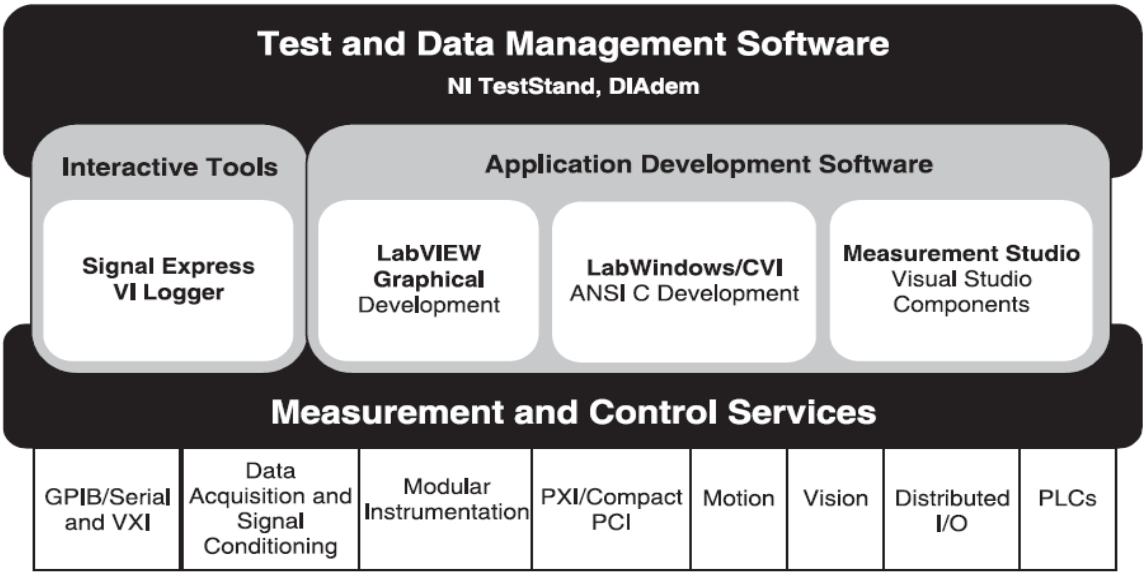

Figure 1.12 Layers of virtual instrumentation software.

The primary development environment for building an application. It includes software such as LabVIEW, LabWindows/CVI (ANSI C), Measurement Studio (Visual Studio programming languages), Signal Express and VI Logger.

Measurement and control services software is equivalent to the I/O driver software layer. It is one of the most crucial elements of rapid application development. This software connects the virtual instrumentation software and the hardware for measurement and control. It includes intuitive application programming interfaces, instrument drivers, configuration tools, I/O assistants and other software included with the purchase of hardware.

VIRTUAL INSTRUMENTATION FOR TEST, CONTROL AND DESIGN

Virtual instrumentation has been widely adopted in test and measurement areas. It has gradually increased addressable applications through continuous innovation and hundreds of measurement hardware devices. The benefits that have accelerated test development are beginning to accelerate control and design.

Virtual Instrumentation for Test

Virtual instrumentation delivers as shown in Figure 1.13 user-defined instruments and customizable hardware for test systems:

- Intuitive software tools for rapid test development
- Fast, precise modular I/O based on innovative commercial technologies
- A PC-based platform with integrated synchronization for high accuracy and throughput

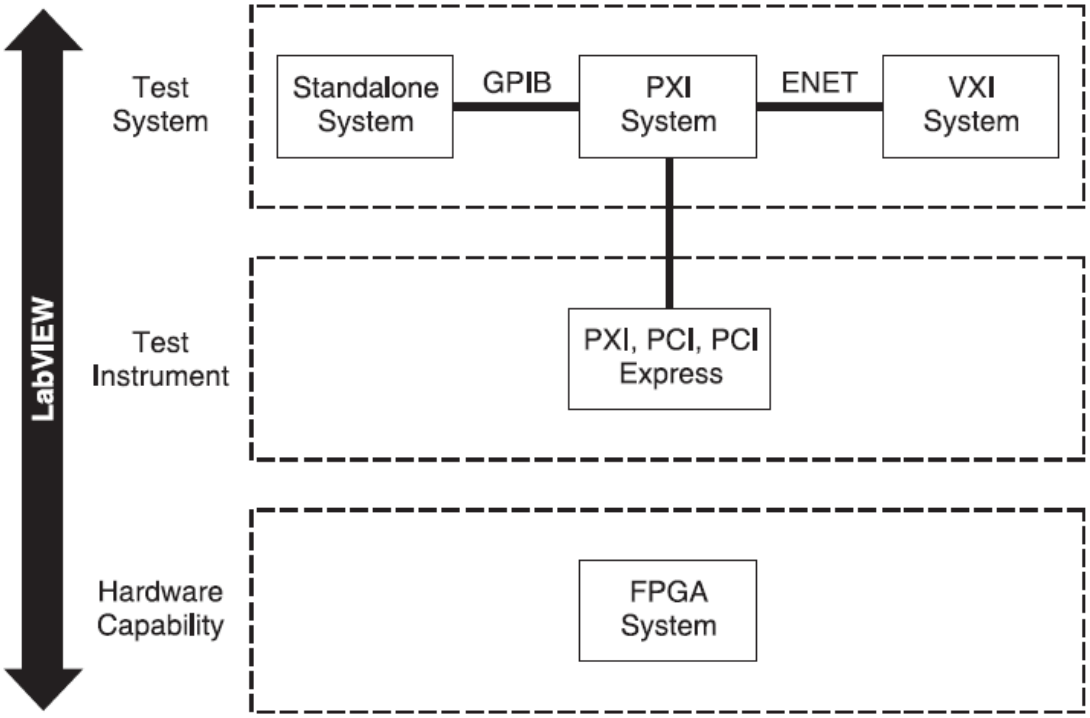

Figure 1.13 User-defined instruments and customizable hardware for test systems.

Test has been a long-proven field for virtual instrumentation. As the pace of innovation has increased,so too has the pressure to get new, differentiated products to market quickly.

Consumer expectations continue to increase; in electronics markets, for example, disparate function integration is required in a small space and at a low cost. All of these conditions drive new validation, verification and manufacturing test needs.

A test platform that can keep pace with this innovation is not optional; it is essential.

The platform must include rapid test development tools adaptable enough to be used throughout the product development flow.

To test the complex multifunctional products that consumers demand requires precise, synchronized measurement capabilities. And as companies incorporate innovations to differentiate their products, test systems must quickly adapt to test the new features.

Virtual instrumentation is an innovative solution to these challenges. It combines rapid development software and modular, flexible hardware to create user-defined test systems.

Engineers and scientists have always been able to use virtual instrumentation software to create

highly integrated user-defined systems using modular I/O, but they can now extend custom configurability to the hardware itself. This degree of user-configurability and transparency will change the way engineers build test systems

Virtual Instrumentation for Industrial I/O and Control

PCs and PLCs both play an important role in control and industrial applications. PCs bring greater software flexibility and capability, while PLCs deliver outstanding ruggedness and reliability. But as control needs become more complex, there is a recognized need to accelerate the capabilities while retaining the ruggedness and reliability.

Independent industry experts have recognized the need for tools that can meet the increasing need for more complex, dynamic, adaptive and algorithm-based control.

The Programmable Automation Controllers (PACs) provide multi domain functionality (logic, motion, drives and process) and the concept of PAC supports multiple I/O types.

Logic, motion and other function integrations are a requirement for increasingly complex control approaches. PACs deliver PC software flexibility with PLC ruggedness and reliability.

LabVIEW software and rugged, real-time, control hardware platforms are ideal for creating a PAC.

Virtual Instrumentation for Design

The same design engineers that use a wide variety of software design tools must use hardware to test prototypes. Commonly, there is no good interface between the design phase and testing/ validation phase, which means that the design usually must go through a completion phase and enter a testing/validation phase. Issues discovered in the testing phase require a design-phase reiteration as shown in Figure 1.14.

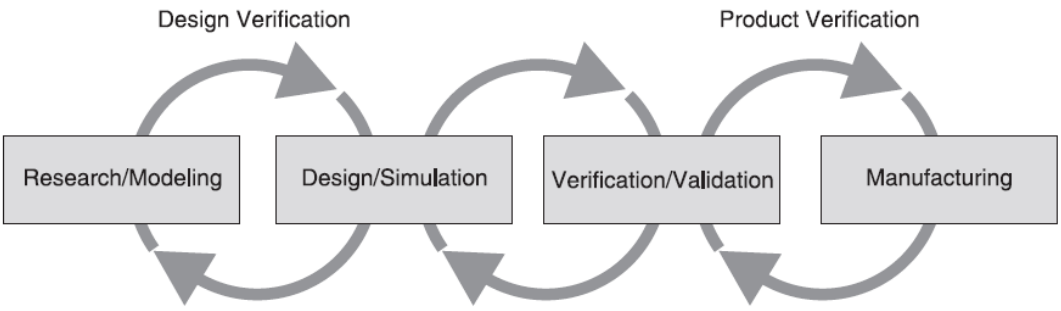

Figure 1.14 Simulation test plays a critical role in the design and manufacture.

In reality, the development process has two very distinct and separate stages—design and test—which are two individual entities. On the design side, Electronic Design Automation (EDA) tool vendors undergo tremendous pressure to interoperate from the increasing semiconductor design and manufacturing group complexity requirements.

Engineers and scientists are demanding the capability to reuse designs from one tool in other tools as products go from schematic design to simulation to physical layout.

Similarly, test system development is evolving towards a modular approach. The gap between these

two worlds has traditionally been neglected, first noticeable in the new product prototype stage. Traditionally, this is the stage where the product designer uses benchtop instruments to sanity-check the physical prototypes against their design for correctness.

The designer makes these measurements manually, probing circuits and looking at the signals on instruments for problems or performance limitations. As designs iterate through this build-measure tweak-rebuild process, the designer needs the same measurements again. In addition, these measurements can be complex—requiring frequency, amplitude and temperature sweeps with data collected and analyzed throughout. Because these engineers focus on design tools, they are reluctant to invest in learning to automate their testing.

Systems with intrinsic-integration properties are easily extensible and adapt to increasing product functionality. When new tests are required, engineers simply add new modules to the platform to make the measurements.

Virtual instrumentation software flexibility and virtual instrumentation hardware modularity make virtual instruments a necessity to accelerate the development cycle.

GRAPHICAL SYSTEM DESIGN USING LabVIEW

The development of VI is linked to the evolution of small inexpensive personal computers and progress in computer hardware and software.

Graphical User Interfaces (GUIs) too played a vital role in the development of VI. While the original application of replacing the controls with a computer link used a command line interface, the use of sophisticated GUI became an essential component in the later versions of VI software.

The major developments that helped VI become successful were development of low cost computer systems of adequate computing power, evolution of good GUI, development of standards for buses, and increasingly stable networking platforms

The computing power of the processors improved by leaps and bounds permitting the development of all these resource intensive techniques. Standardization and progress in computing hardware increased. Developments of various interfacing standards lead to a massive reduction in development effort. All these developments have occurred in parallel and have taken us to the situation where VI is the dominant tool for the development and implementation of instrumentation applications and systems.

The term “VI” owes a lot to the development of the Laboratory Virtual Instrument Engineering Workbench (LabVIEW) by National Instruments, Austin, Texas, USA.

In 1983, National Instruments began to search for a way to minimize the time needed to program instrumentation systems.

LabVIEW made its appearance as an interpreted package on the Apple Macintosh in 1986.

LabVIEW2 was released in 1990, and was a compiled package as opposed to an interpreted package. The first reasonably stable graphical environment (Windows 3.0) made its appearance only in 1990.

In 1992, they introduced LabVIEW for Windows and LabVIEW for Sun based on the new portable architecture.

LabVIEW 3 arrived in 1993 for Macintosh, Windows and Sun operating Graphical System Design 17 systems. LabVIEW 3 programs written on one platform could run on another.

In 1999, LabVIEW became available for the Linux platform as well.

LabVIEW 4, released in 1996, featured a more customizable development environment so that users could create their own workspace to match their industry, experience level and development habits. In addition, LabVIEW 4 added high powered editing and debugging tools for advanced instrumentation systems, as well as OLE-based connectivity and distributed execution tools.

Networking support on smaller systems was first introduced in Version 5. LabVIEW 5 and 5.1 (in 1999) continued to improve on the development tool by introducing a built-in Web server, a dynamic programming and control framework (VI server), integration with ActiveX, and easy sharing of data over the Internet with a protocol called DataSocket.

In 2000, LabVIEW 6 (sometimes called 6i) provided both an easy and intuitive programming interface.

In 2001, LabVIEW 6.1 introduced event-oriented programming, remote Web control of LabVIEW.

Version 7 has expanded the horizon to make it simpler for the inexperienced user by providing various Express utilities and Assistants. LabVIEW has two closely related products—BridgeVIEW and LabVIEW RT (for realtime applications). BridgeVIEW can also be called LabVIEW industrial. The RT module of LabVIEW was originally designed to support distributed computing and real-time applications.

National Instruments released LabVIEW 8.5, the latest version of the graphical system design platform for test, control and embedded system development. LabVIEW 8.5 simplifies multicore as well as FPGA-based application development with its intuitive parallel dataflow language.

NI LabVIEW 8.6 includes the platform installation DVDs, Block Diagram Cleanup, Quick Drop and Web services creation.

With the enhanced releases of LabVIEW graphical programming software, system-level engineers as well as domain experts with little to no embedded expertise can accurately work with systems of increased complexity and scale, thereby drastically reducing the time from concept to prototype.

GRAPHICAL PROGRAMMING AND TEXTUAL (TRADITIONAL)PROGRAMMING

Text-based programming	Graphical programming
• *Syntax* must be known to do programming.	• *Syntax* is knowledge but is not required for programming.
• The execution of the program is from *top to bottom.*	• The execution of program is from *left to right.*
• To check for the *error* the program has to be compiled or executed.	• *Errors* are indicated as we wire the blocks.
• *Front panel* design needs extra coding or needs extra work.	• *Front panel* design is a part of programming.
• Text-based programming is *non interactive.*	• Graphical programming is highly *interactive.*
• This is text-based programming where the programming is a *conventional method.*	• The programming is *Data Flow Programming.*
• *Logical Error* finding is easy in large programs.	• *Logical Error* finding in large programs is quiet complicated.
• Program flow is *not visible.*	• Data flow is *visible.*
• It is *test-based* programming.	• It is *icon-based* programming and wiring.
• Passing parameters to *sub routine* is difficult.	• Passing parameters to *sub VI* is easy.