INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Návrh řídících systémů v prostředí LabVIEW a jejich HIL testování v prostředí Veristand

Ing. Roman Vala a Ing. Jiří Keprt, Ph.D.

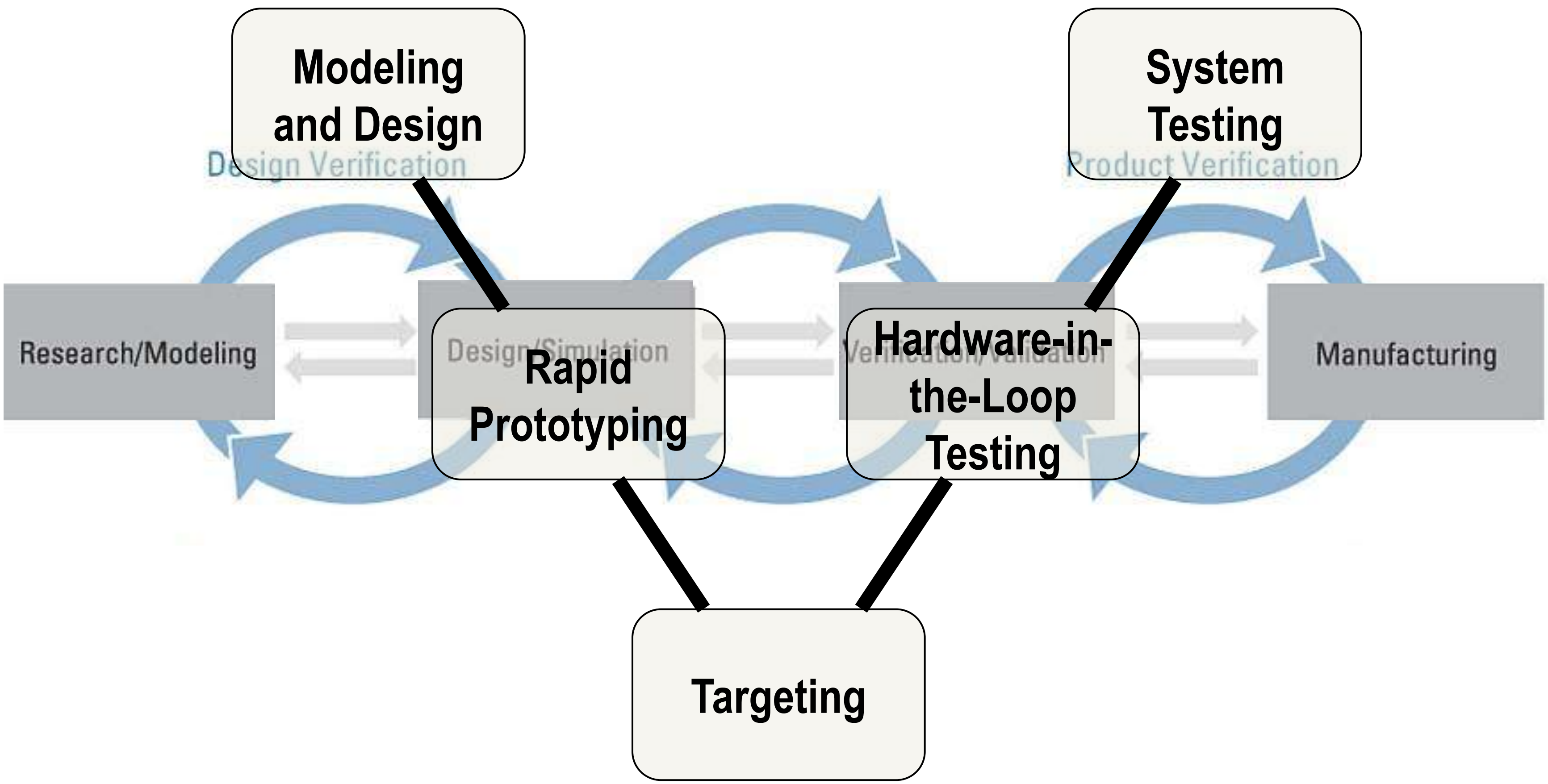
8. dubna 2011

Tato prezentace je spolufinancována Evropským sociálním fondem a státním rozpočtem České republiky.

Agenda

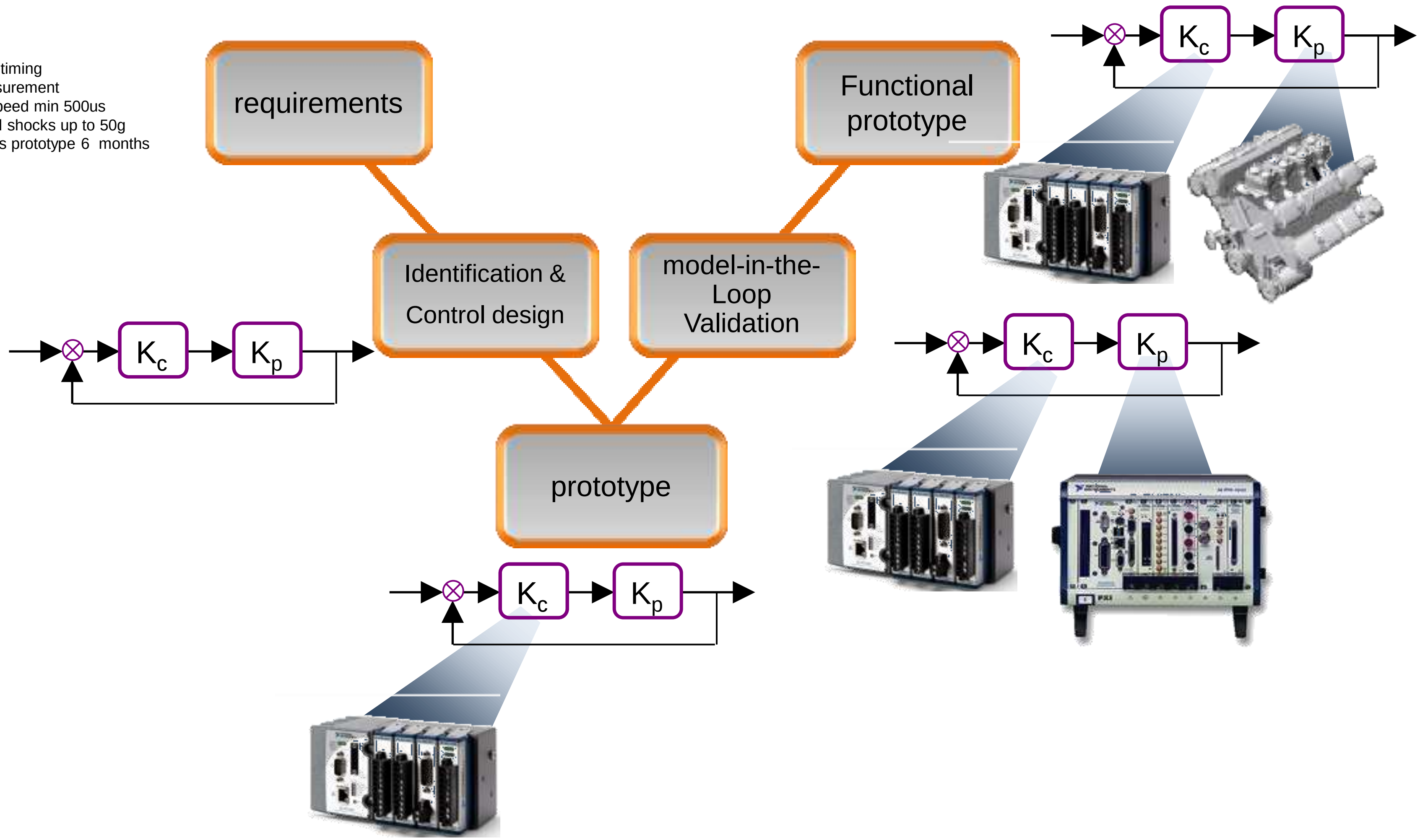
- Úvod
- SW pro smulace a modelování
- Operační systém Real-Time a jeho programování
- Využití obvodů FPGA pro připojení signálů a simulaci snímačů
- HW platformy pro spouštění řídících modelů pod operačním systémem Real-Time
- HIL (Hardware In the Loop) testování

PC-Based Control and Simulation

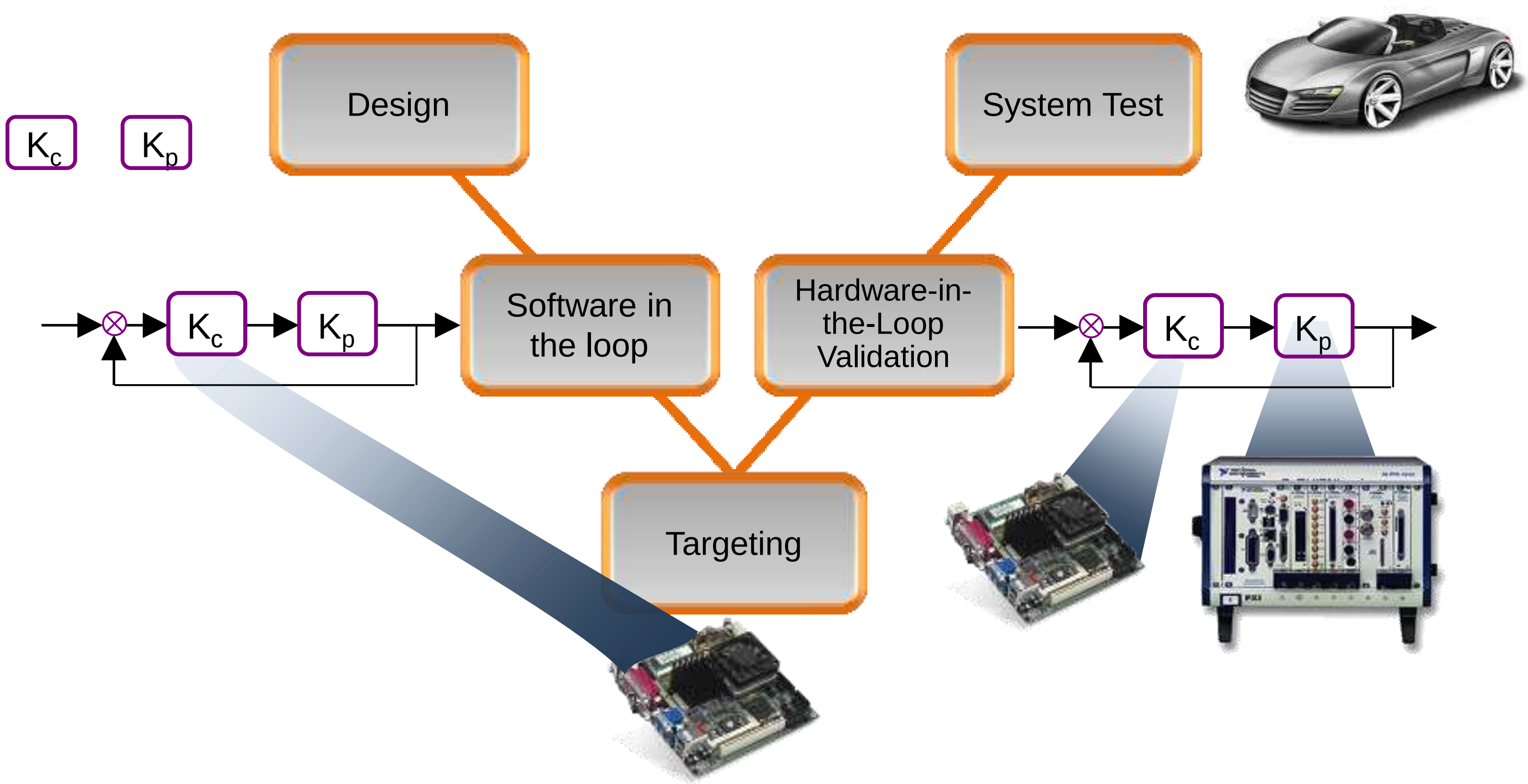


Prototyping V diagram

Gas Engine
Variable valve timing
Vibration measurement
Control loop speed min 500us
Must withstand shocks up to 50g
deadline to finis prototype 6 months

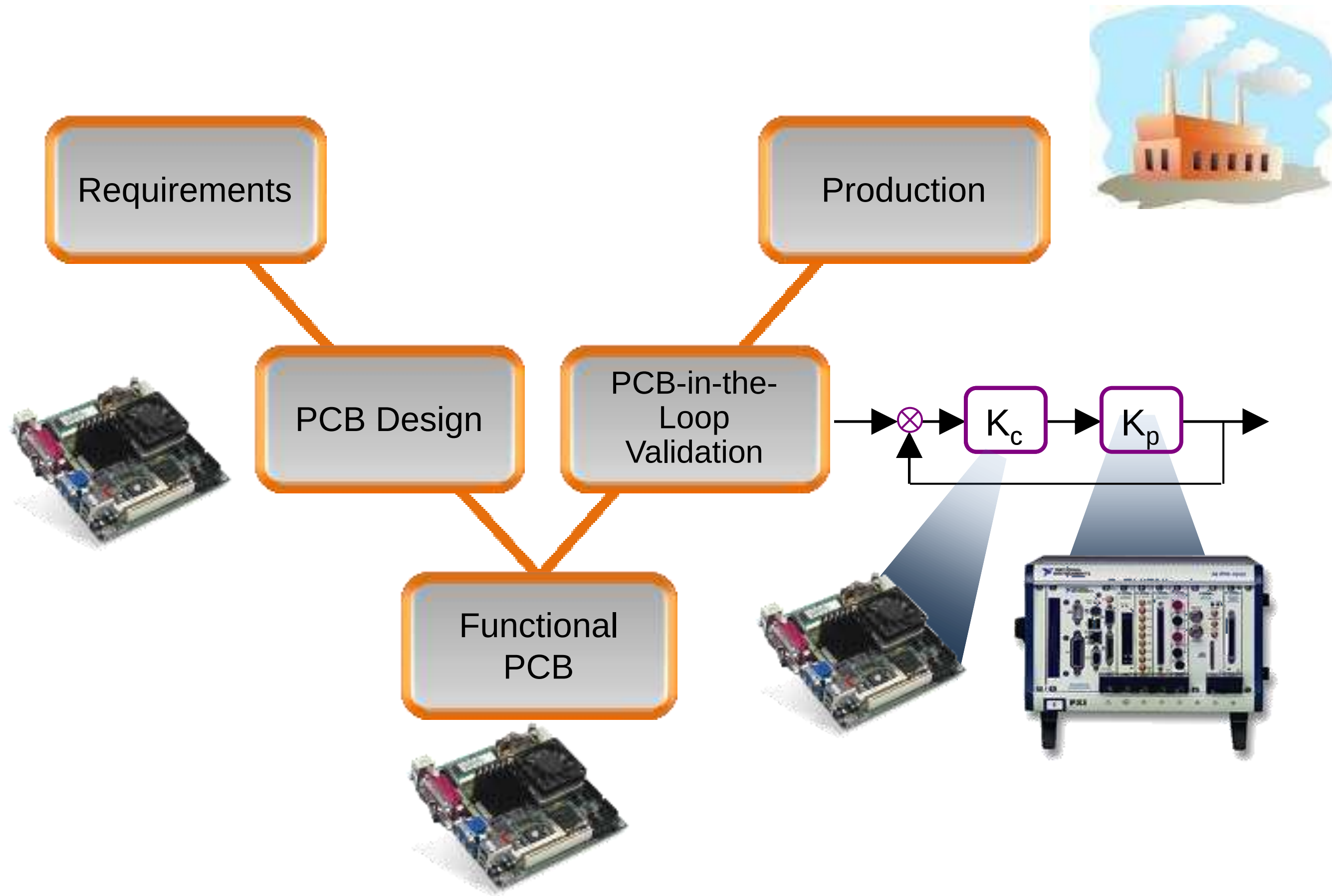


Target deployment V Diagram



Manufacturing V diagram

Based on uP architecture
All tantal capacitor design
Control loop speed min 500us
Must withstand shocks up to 50g
deadline to finish production 6 months



PXI (PCI eXtensions for Instrumentation)

PXI Controller

- Embedded PC or remote PC interface
- Runs all standard software

Chassis

PXI Backplane

- PCI bus
- Synchronization

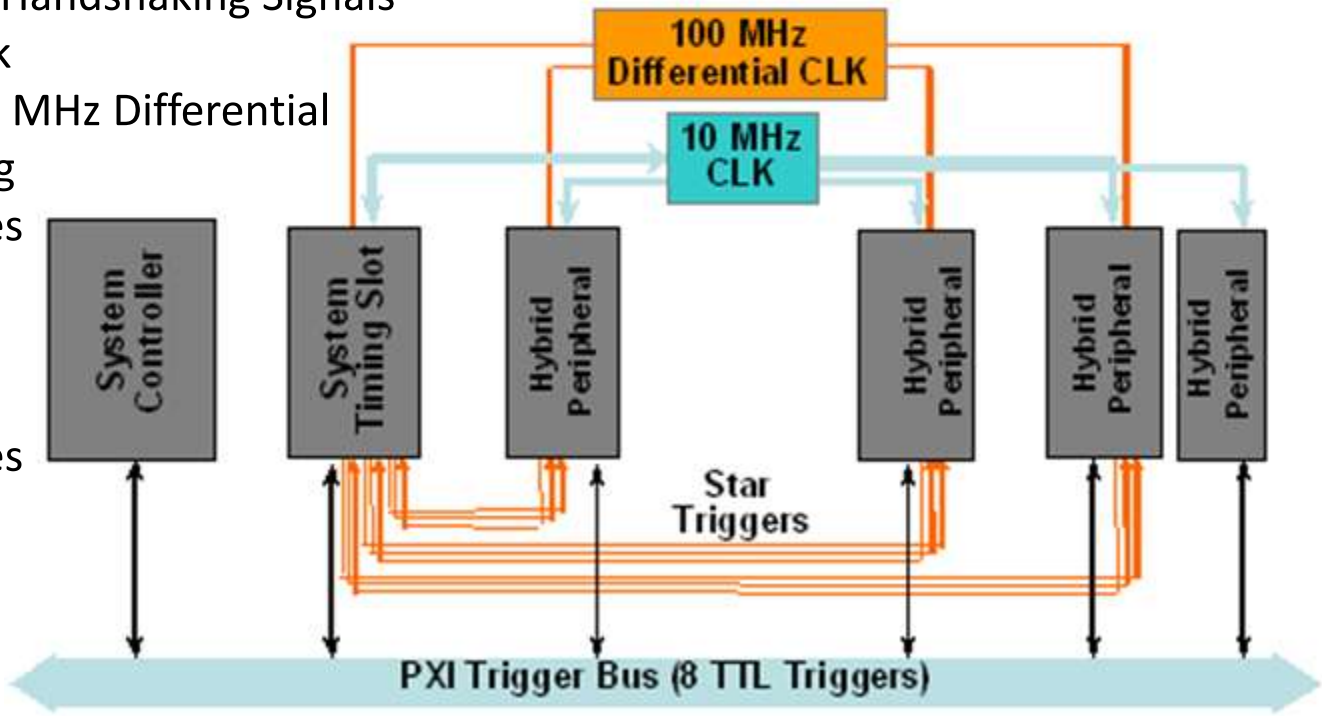
Peripheral Slots

PXI Chassis

- 3U, 6U, and 3U/6U combo
- 4 through 26 slots
- Portable, benchtop, rack mount
- AC and DC power options
- Application specific
  - Ultra rugged, integrated signal conditioning, integrated LCD, etc.

Timing and Synchronization Features of PXI

- PXI Trigger Bus
  - 8 TTL
  - Trigger, Clock, and Handshaking Signals
- System Reference Clock
  - 10 MHz TTL or 100 MHz Differential
  - Phase Lock Looping
  - Equal-Length Traces (< 200 ps skew)
- Star Trigger
  - Differential
  - Equal-Length Traces (< 150 ps Skew)



Embedded PXI System Controllers

General Purpose OSs

- Windows, Linux, etc.
- High performance
- Integrated peripherals
  - Gigabit Ethernet, USB 2.0, ExpressCard, etc.
- Ethernet / LAN control of PXI

Real-Time OSs

- LabVIEW Real-Time, VxWorks, etc.
- Determinism and reliability
- Headless operation

PXI Multiprocessing

Remote PXI System Controllers

PC Control of PXI

- Use latest high-performance PCs
- Build multichassis PXI systems

Laptop Control of PXI

- Control portable applications
- Use with DC-powered chassis for mobile systems

PXI Products. . .

Data Acquisition and Control

Multifunction I/O
Analog Input/Output
Digital I/O
Counter/Timer
FPGA/Reconfigurable I/O
Machine Vision
Motion Control
Signal Conditioning
Temperature
Strain/Pressure/Force/Load
Synchro/Resolver
LVDT/RVDT
Many More. . .

Modular Instrumentation

Digital Waveform Generator
Digital Waveform Analyzer
Digital Multimeter
LCR Meter
Oscilloscope/Digitizer
Source/Signal Generator
Switching
RF Signal Generator
RF Signal Analyzer
RF Power Meter
Frequency Counter
Programmable Power Supply
Many More. . .

Bus Interfaces

Ethernet, USB, FireWire
SATA, ATA/IDE, SCSI
GPIB
CAN, DeviceNet
Serial RS-232, RS-485
VXI/VME
Boundary Scan/JTAG
MIL-STD-1553, ARINC
PCMCIA/CardBus
PMC
Profibus
LIN
Many More. . .

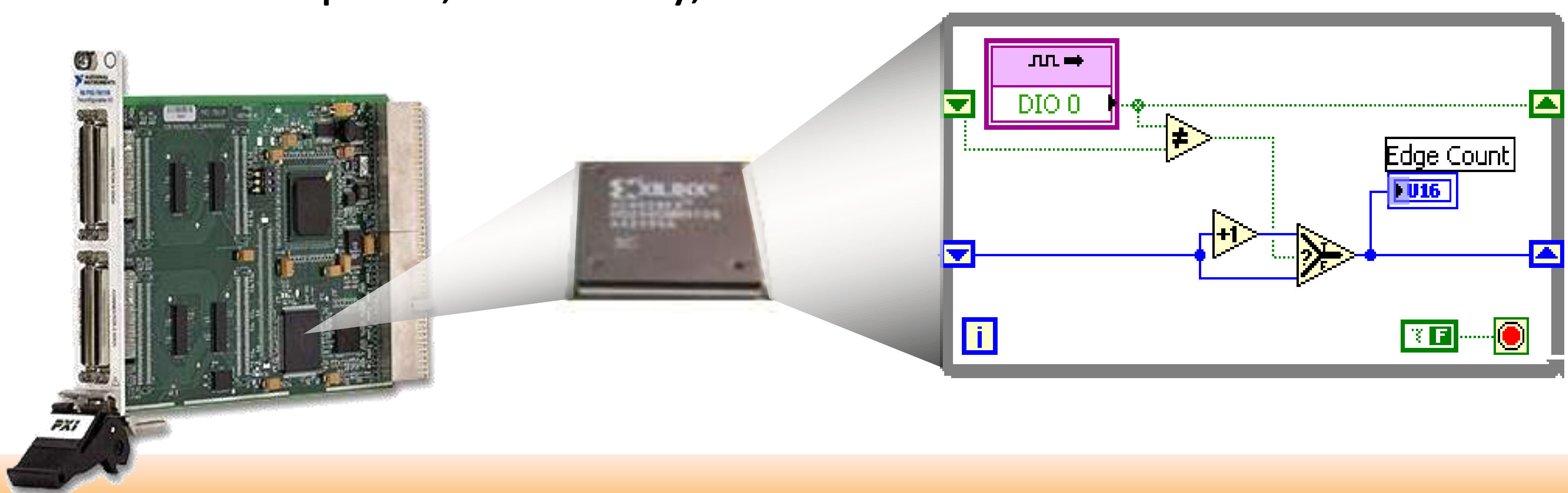
Others

IRIG-B, GPS
Direct-to-Disk
Reflective Memory
DSP
Optical
Resistance Simulator
Fault Insertion
Prototyping/Breadboard
Graphics
Audio
Many More. . .

Over 1500 different modules from more than 70 manufacturers

Reconfigurable IO FPGA

- Intuitive programming for both embedded engineers and domain experts
- High-speed timing and synchronization
- Custom digital protocols **User-defined COTS**
- In-line signal processing
- Hardware speed, reliability, and determinism



Testování pomocí NI Veristand

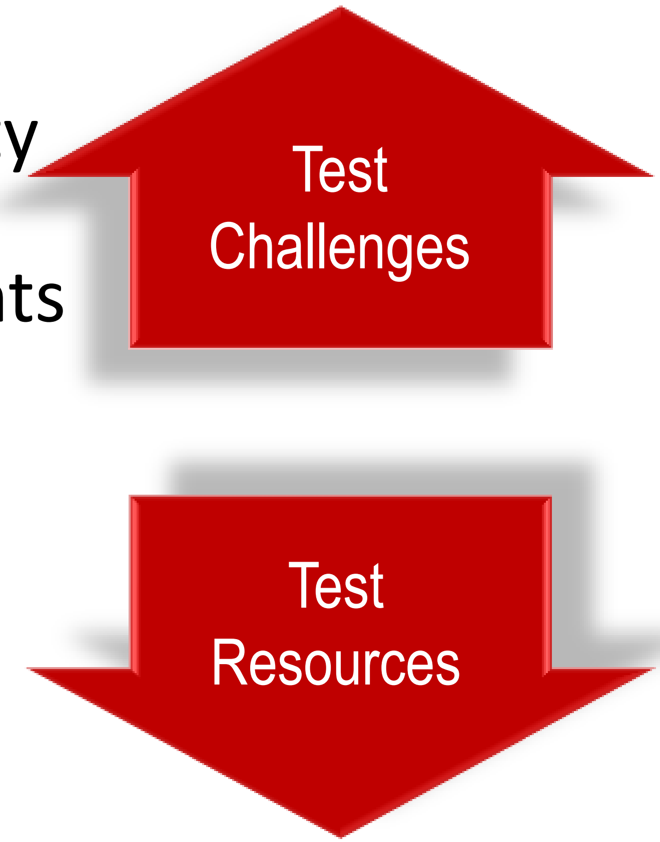
HIL (HARDWARE IN THE LOOP)

Agenda

- HIL introduction
- NI Veristand
- Demo application

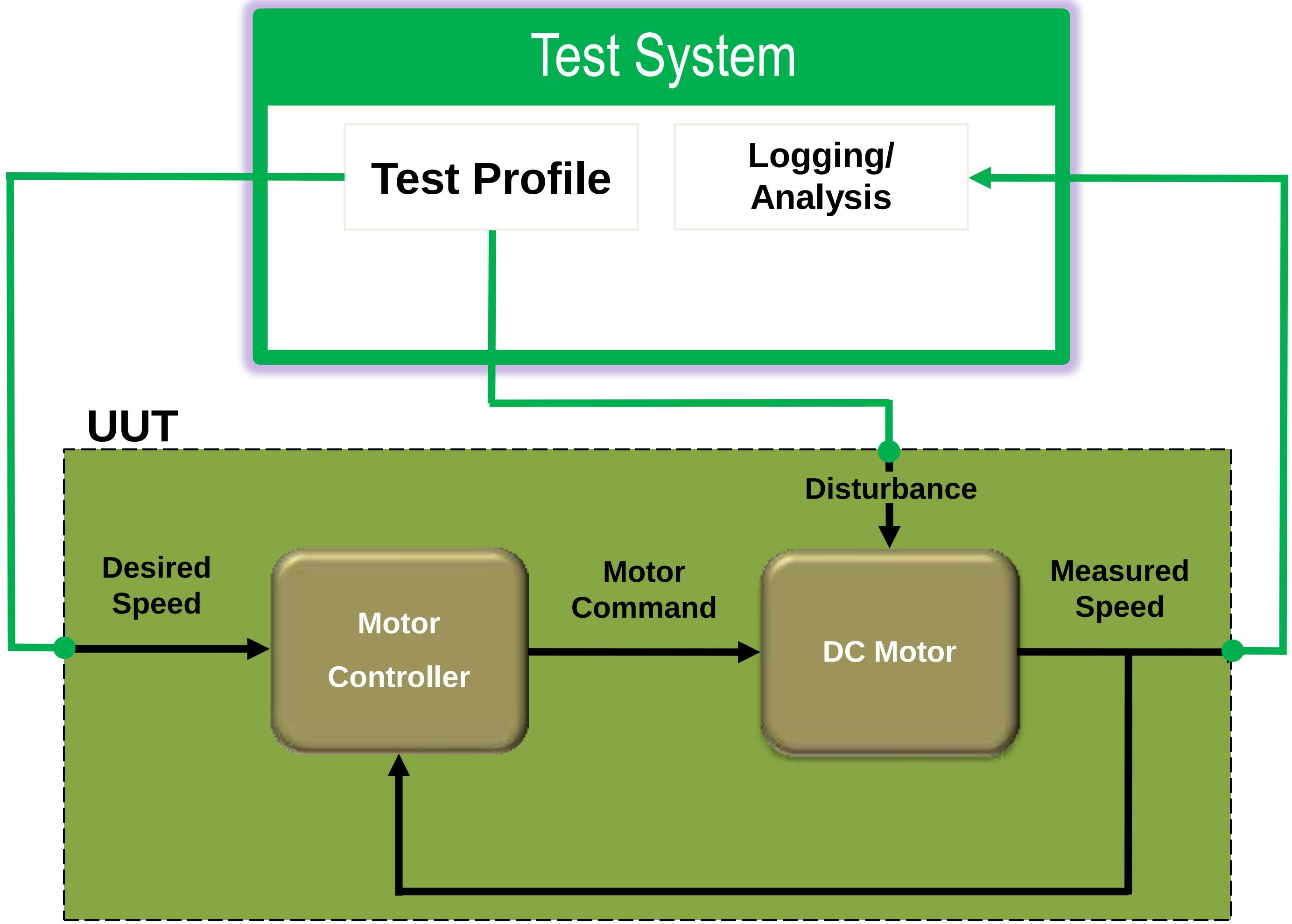
Embedded Control System Challenges

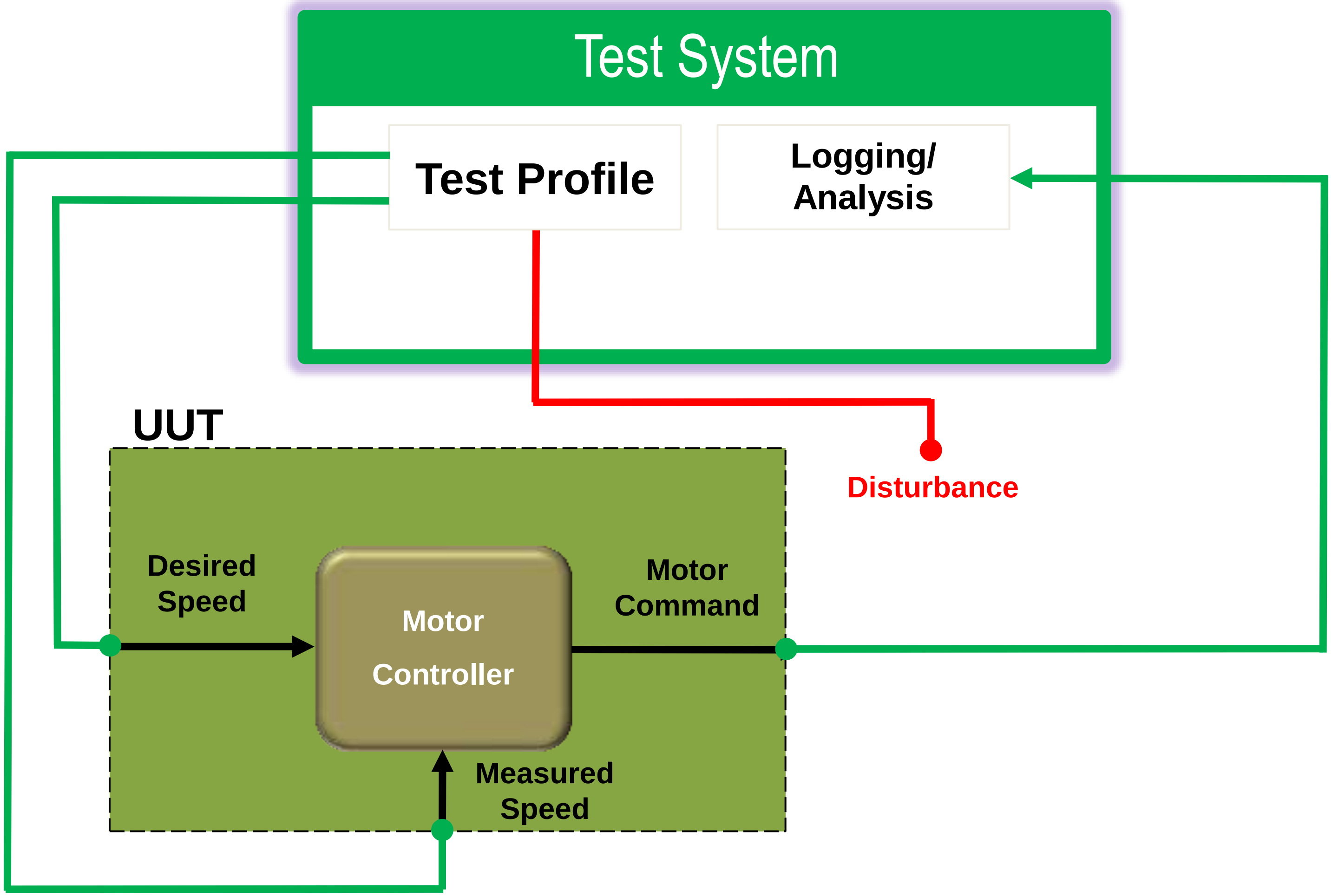
- Increasing application complexity
- Increasing reliability requirements
- Decreasing time-to-market
- Reduce development cost



Are these mutually exclusive?

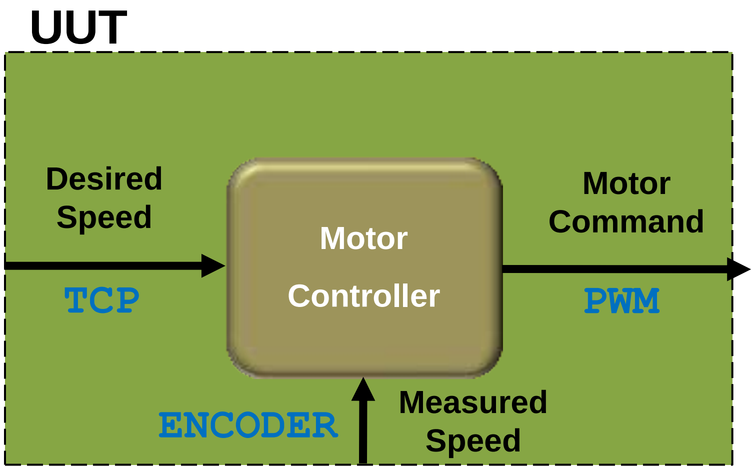
System Level Testing





Test System
Test Profile
Logging/
Analysis
UUT
Disturbance
Desired
Speed
Motor
Controller
Motor
Command
Measured
Speed

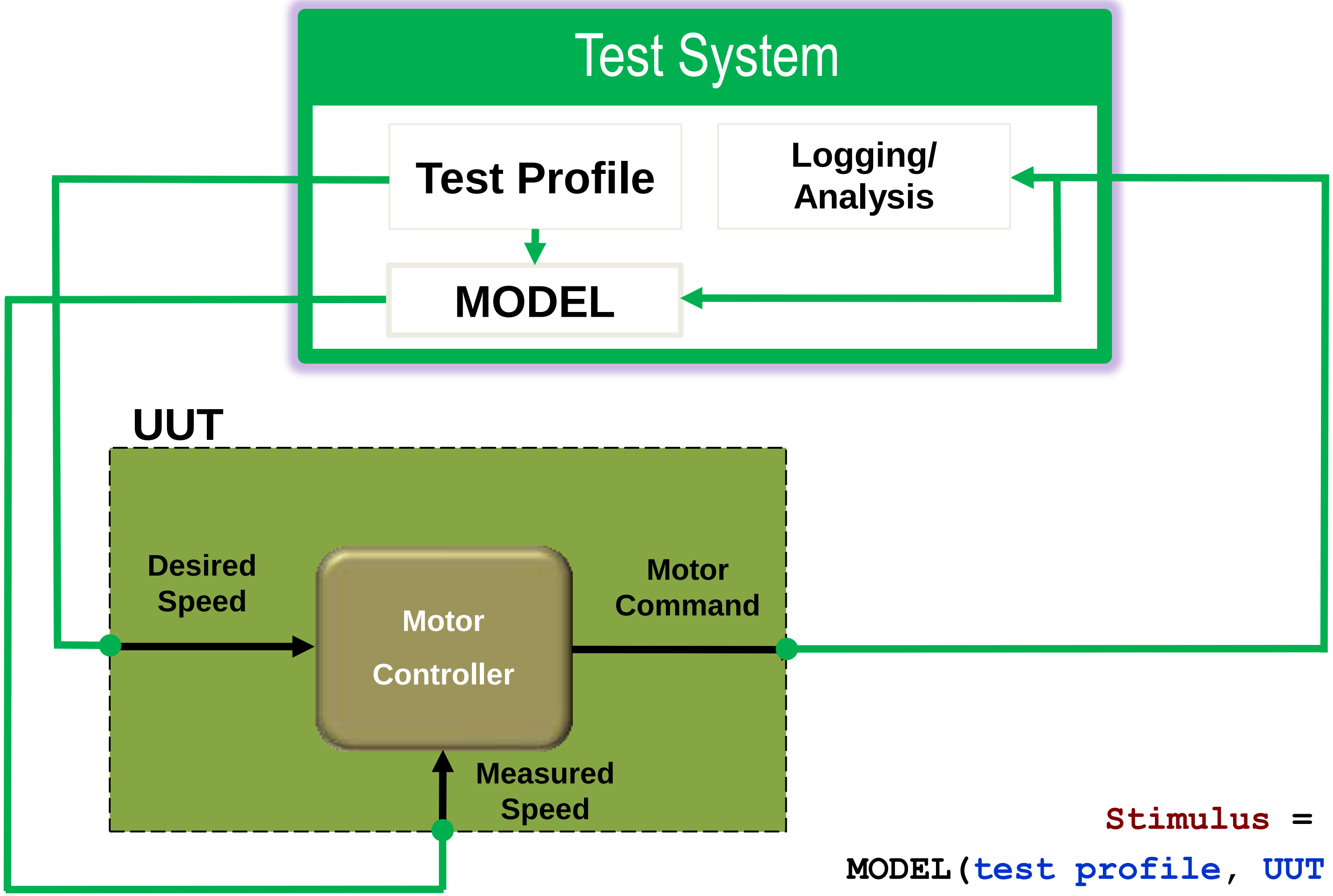
What does your UUT *know* about the world around it?



- Voltage
- Current
- Impedance
- Timing

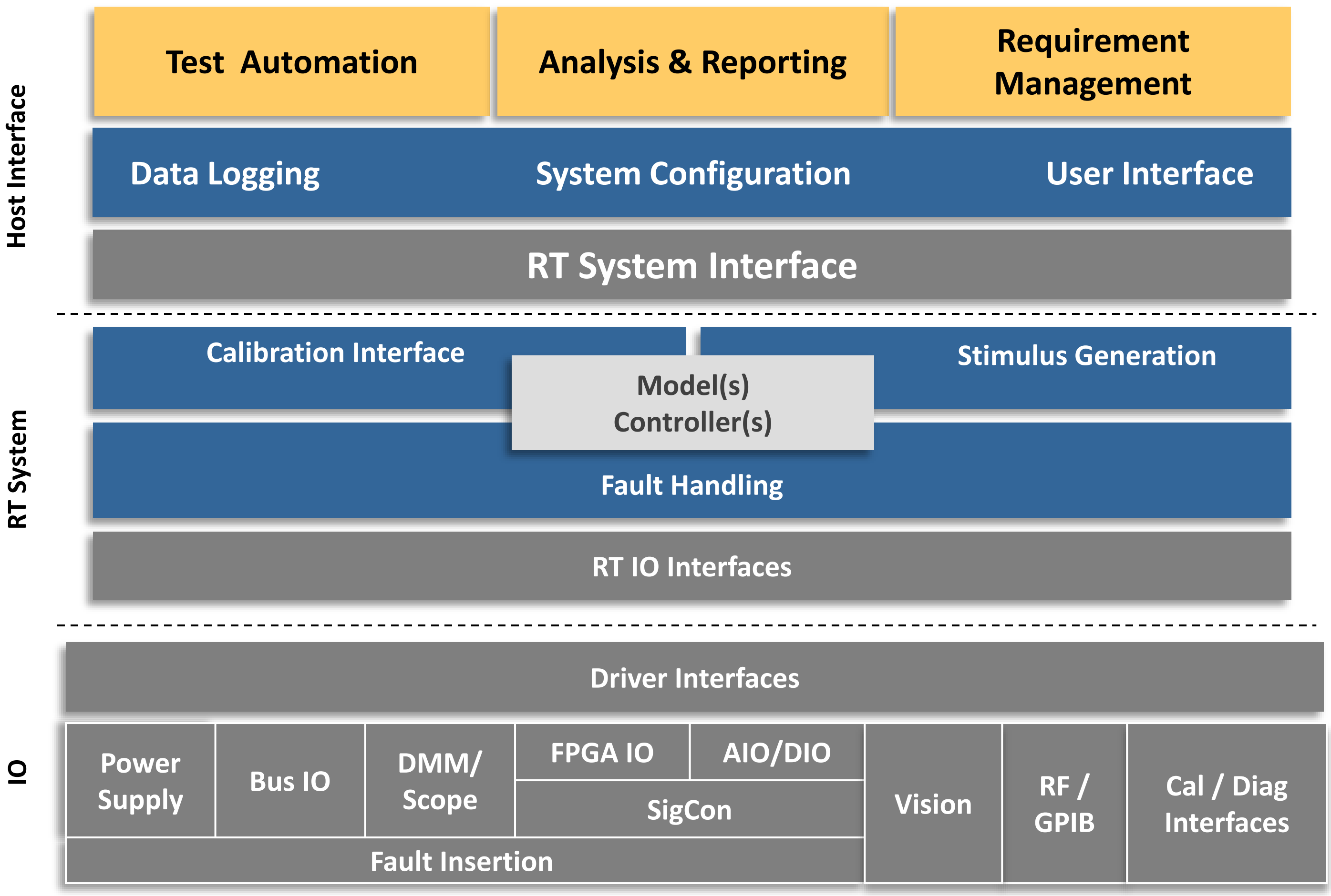
Testing Embedded Control Systems

Virtual System Level Testing

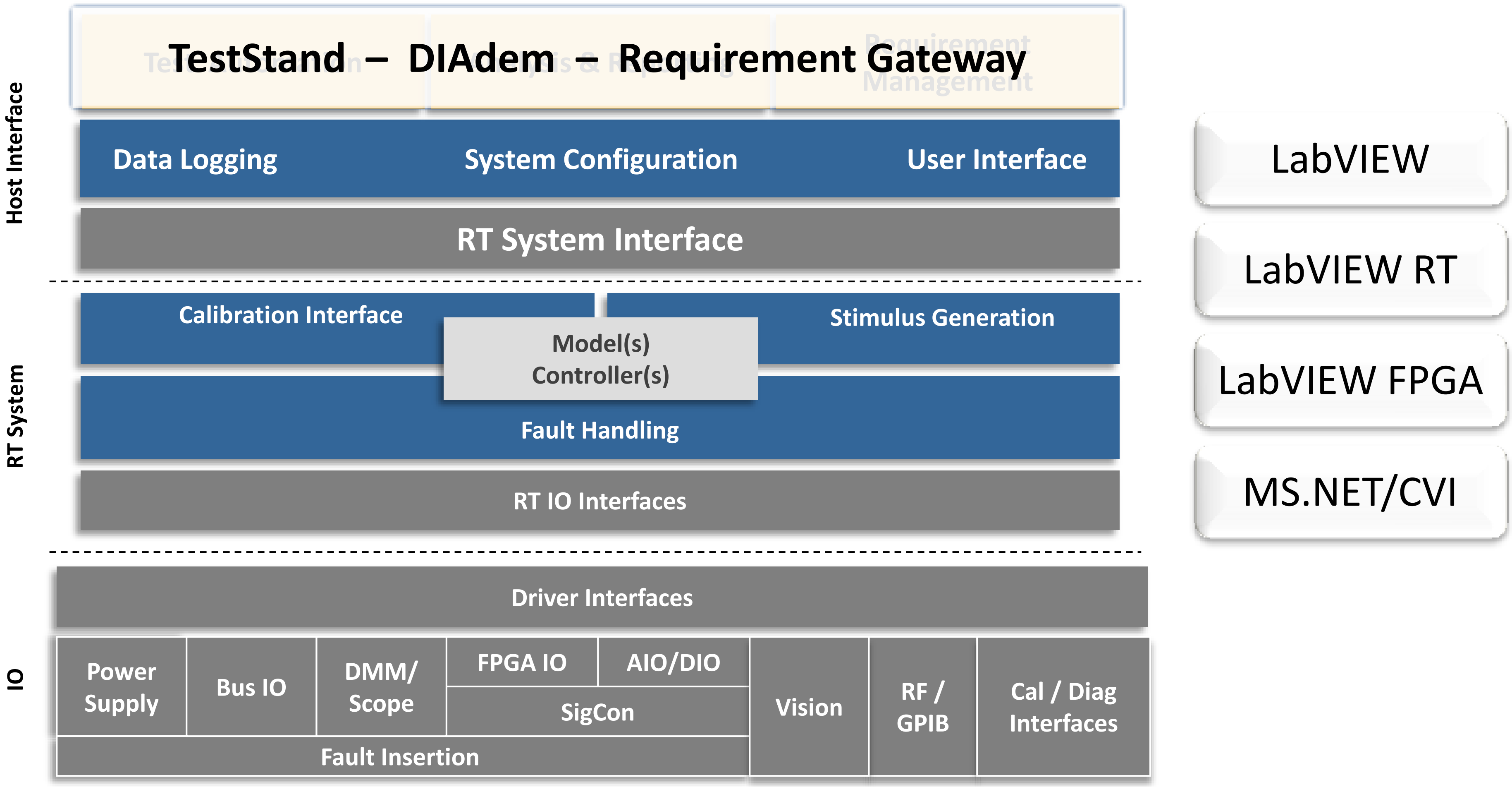


Stimulus = MODEL(test profile, UUT response)

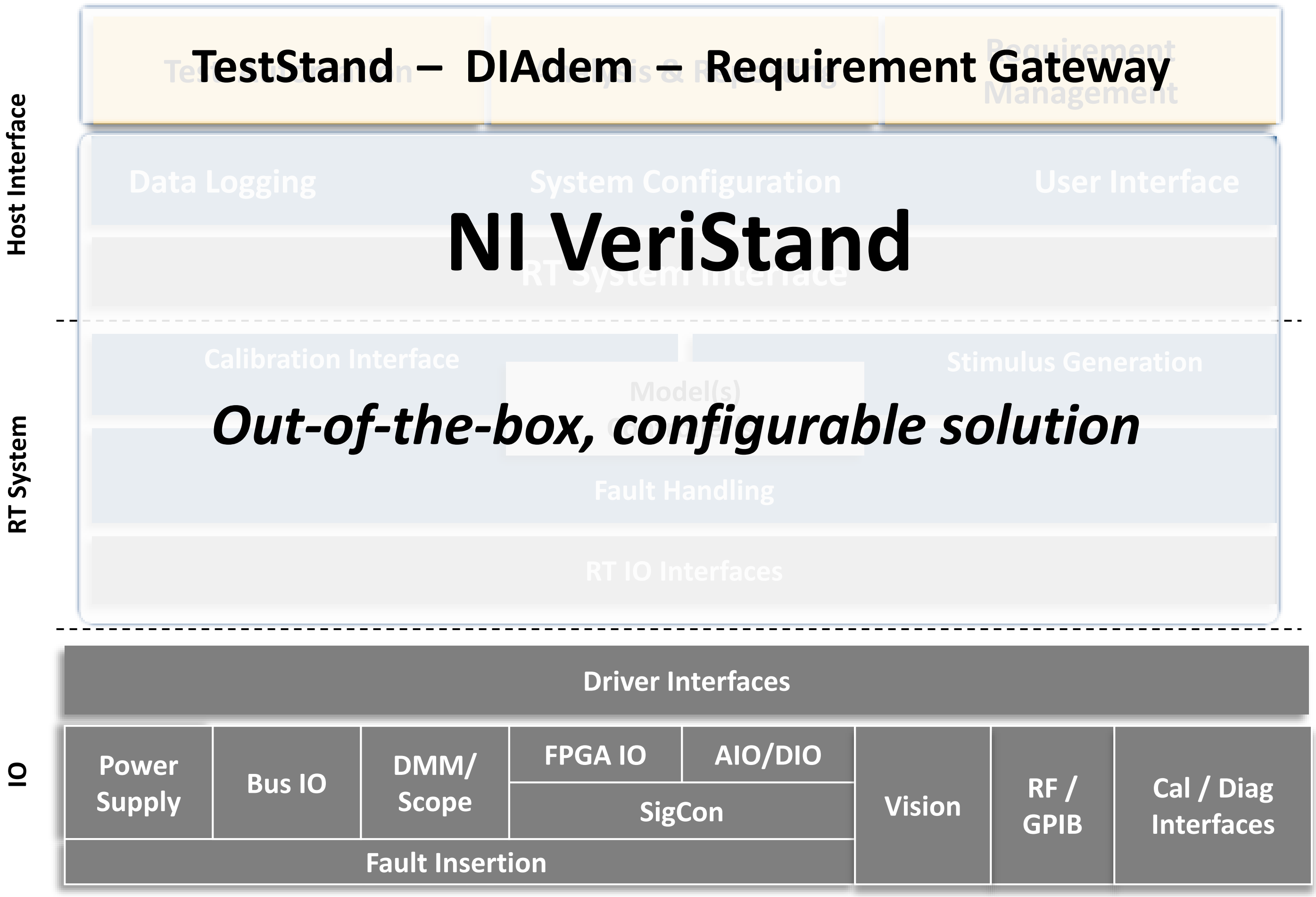
Real-Time Test System Architecture



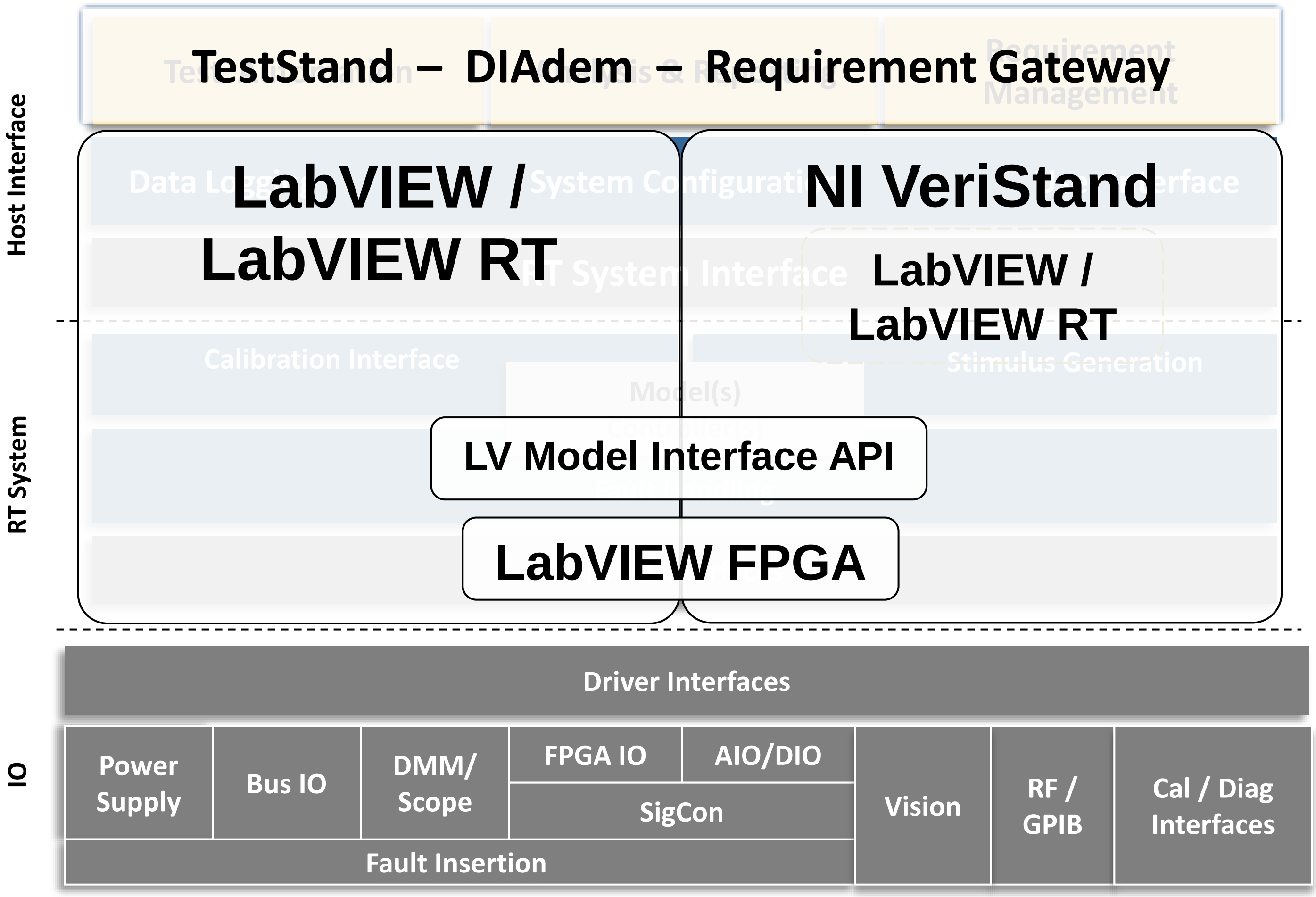
Real-Time Test System Architecture



Real-Time Test System Architecture

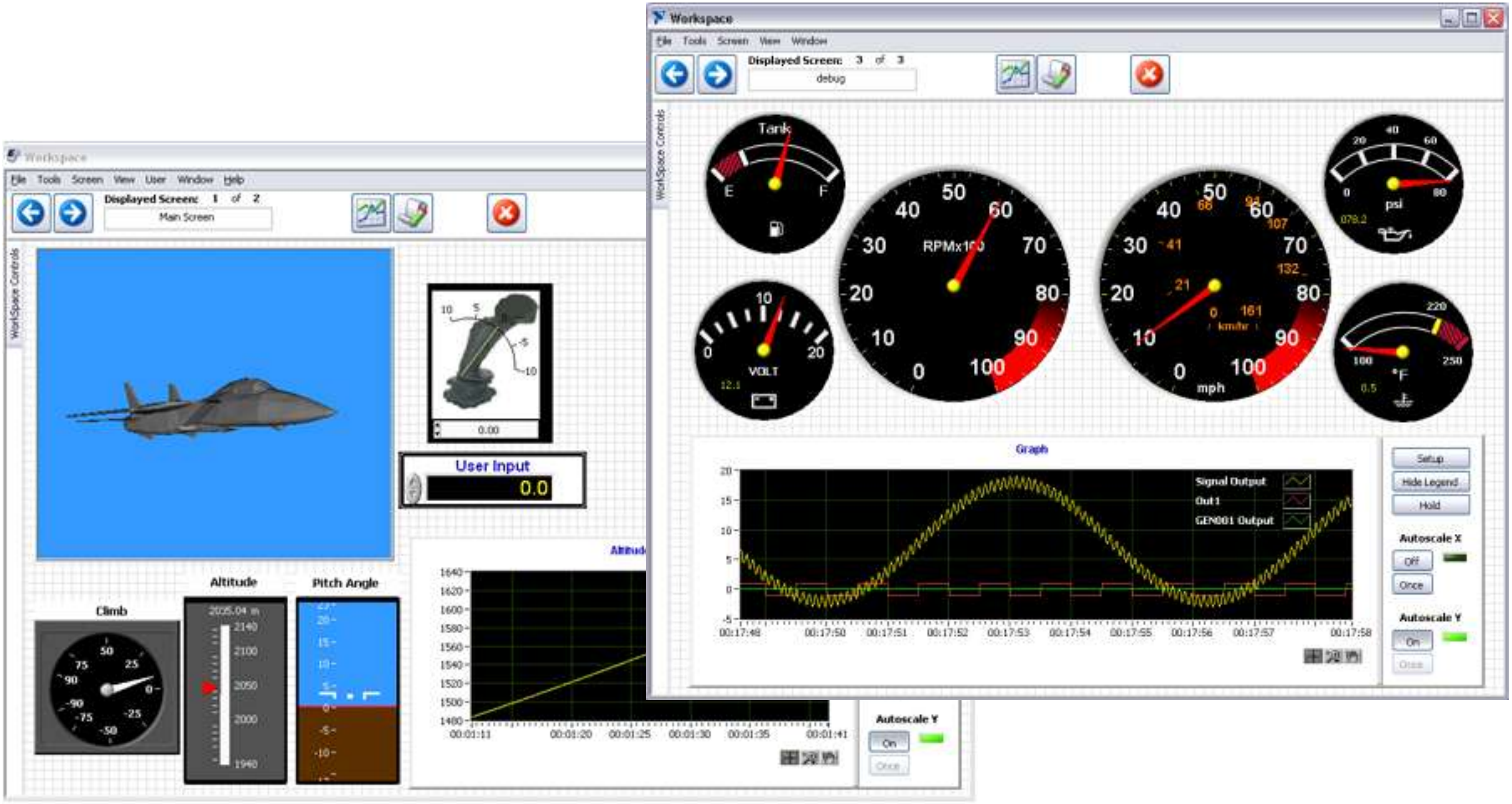


NI Real-Time Testing Platform – *2 Paths*



NI VeriStand

Real-Time Testing and Simulation Software



VeriStand Getting Started Window

NI VeriStand™

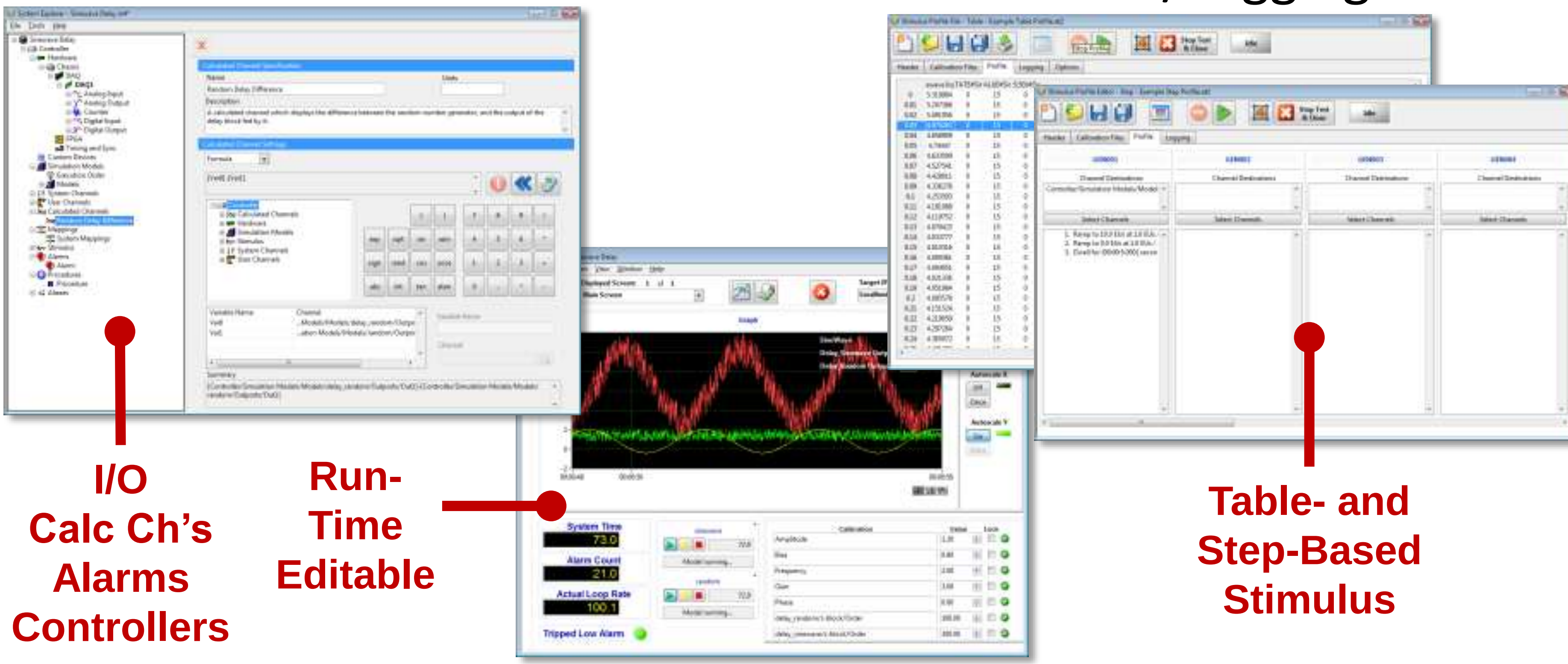
Real-Time Testing and Simulation Software

- Stimulus Generation
- Data Logging
- Configurable I/O
- Alarming
- Calculated Channels
- Run-Time Editable User Interface
- User Management
- Multichassis Synchronization
- Closed-Loop Control
- Deterministic Model Execution

NI VeriStand

Configure Real-Time Application

Deploy Real-Time Stimulus/Logging Profiles



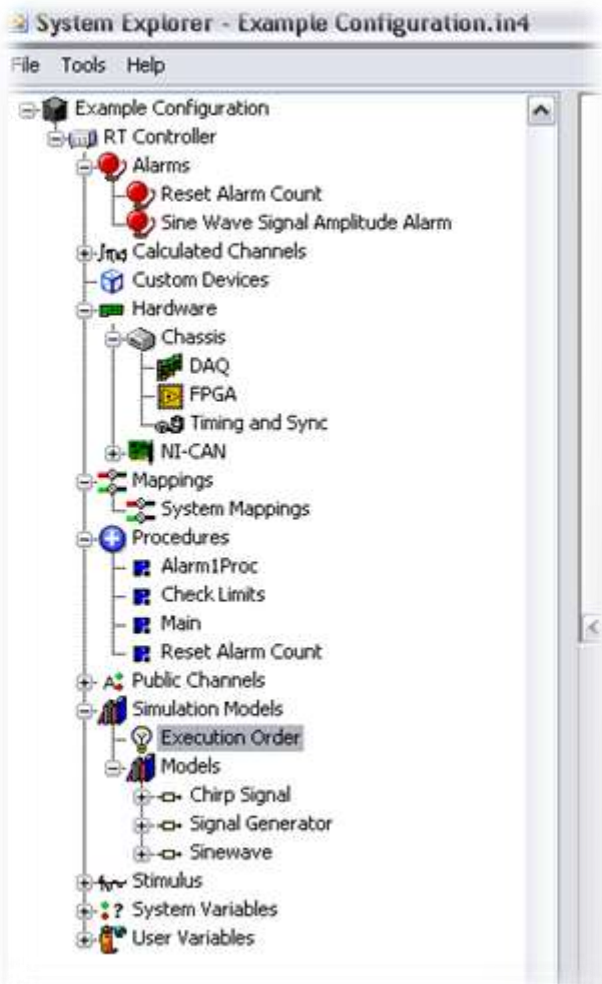
I/O
Calc Ch's
Alarms
Controllers

Run-Time Editable

Table- and Step-Based Stimulus

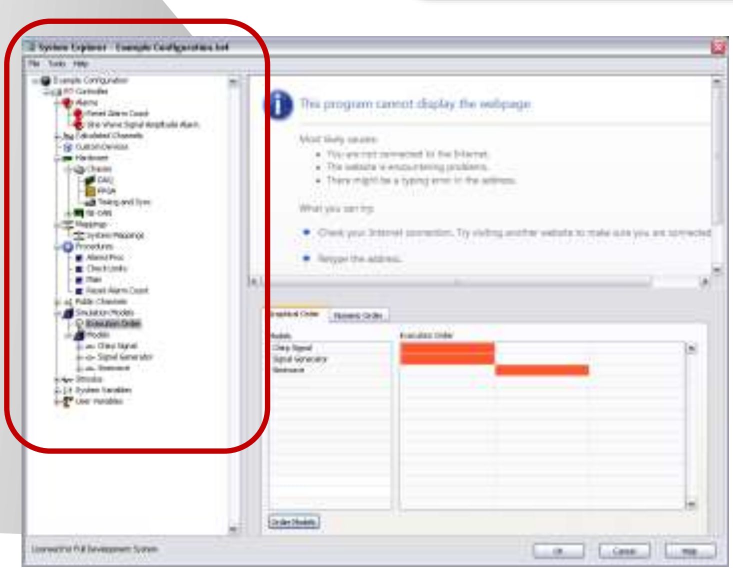
Create UI at Run Time

NI VeriStand Framework

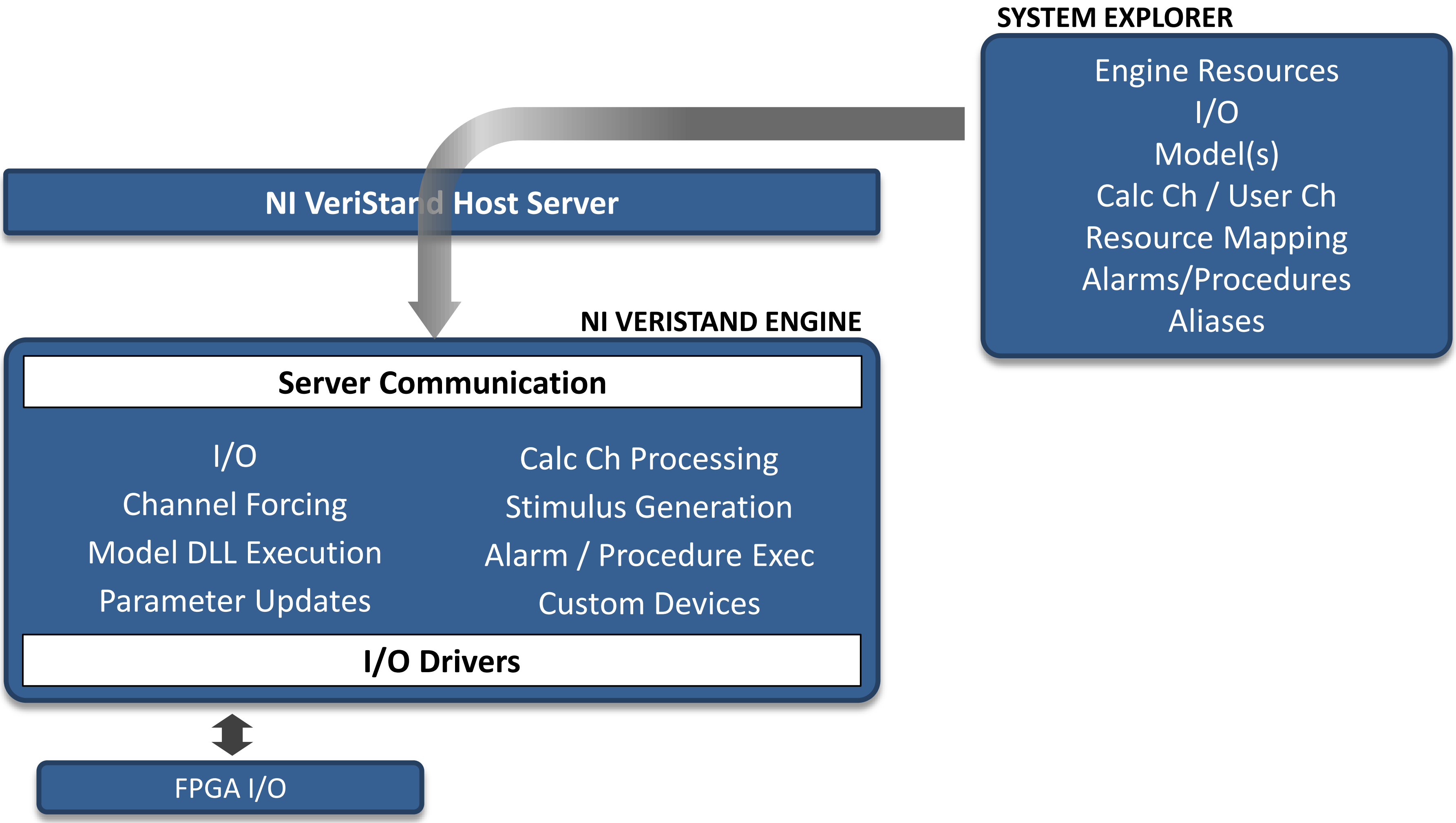


SYSTEM EXPLORER

Engine Resources
I/O
Model(s)
Calc Ch / User Ch
Resource Mapping
Alarms/Procedures
Aliases



NI VeriStand Framework



NI VeriStand Framework

WORKSPACE

User Interface

Utilities

NI VeriStand Host Server

NI VERISTAND ENGINE

Server Communication

I/O	Calc Ch Processing
Channel Forcing	Stimulus Generation
Model DLL Execution	Alarm / Procedure Exec
Parameter Updates	Custom Devices

I/O Drivers

FPGA I/O

OBSERVE:
- Alarm Monitor
- CAN Bus Monitor
- Channel Data Viewer
- TDMS File Viewer
- Real-Time Consol Viewer

ACTION:
- Channel Scaling and Calibration
- Channel Value Forcing
- Stimulus Profile Editor

CONFIGURATION:
- Model Parameter Manager
- Alarm Manager

NI VeriStand Framework

WORKSPACE

User Interface

Utilities

NI VeriStand Host Server

NI VERISTAND ENGINE

Server Communication

I/O	Calc Ch Processing
Channel Forcing	Stimulus Generation
Model DLL Execution	Alarm / Procedure Exec
Parameter Updates	Custom Devices

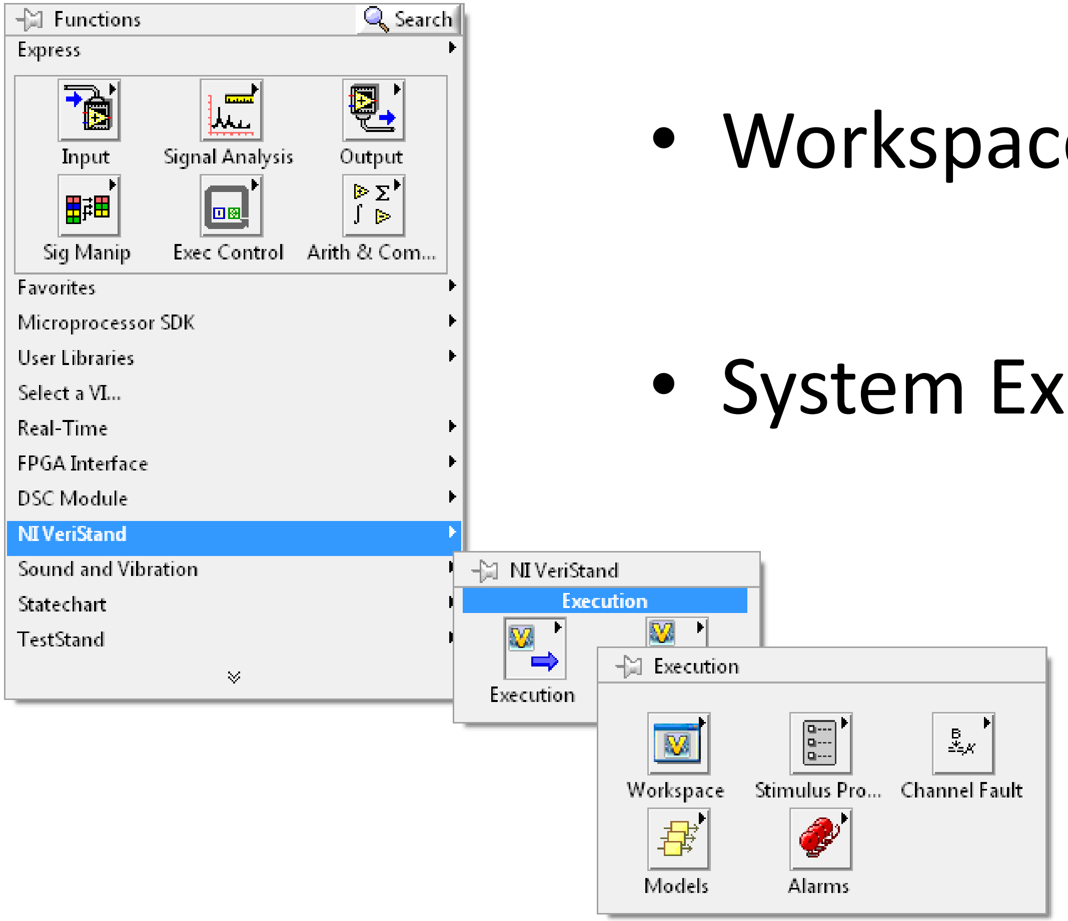
I/O Drivers

FPGA I/O

STIMULUS PROFILE EDITOR:

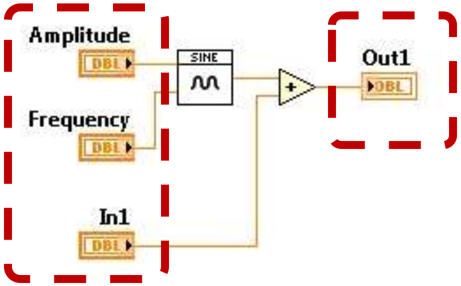
- Generate Test Header
- Load Model Parameters
- Define Stimulus Profiles
- Configure Data Logging

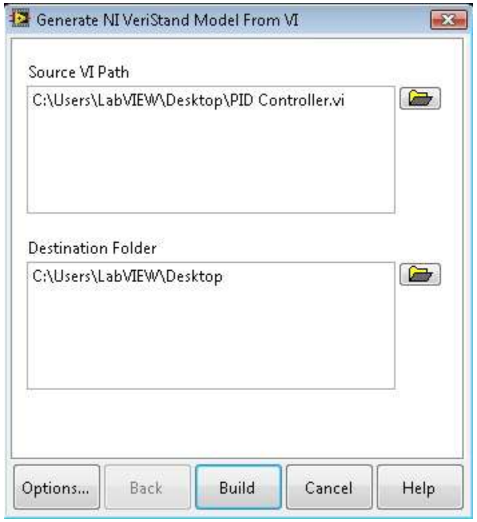
NI VeriStand .NET APIs

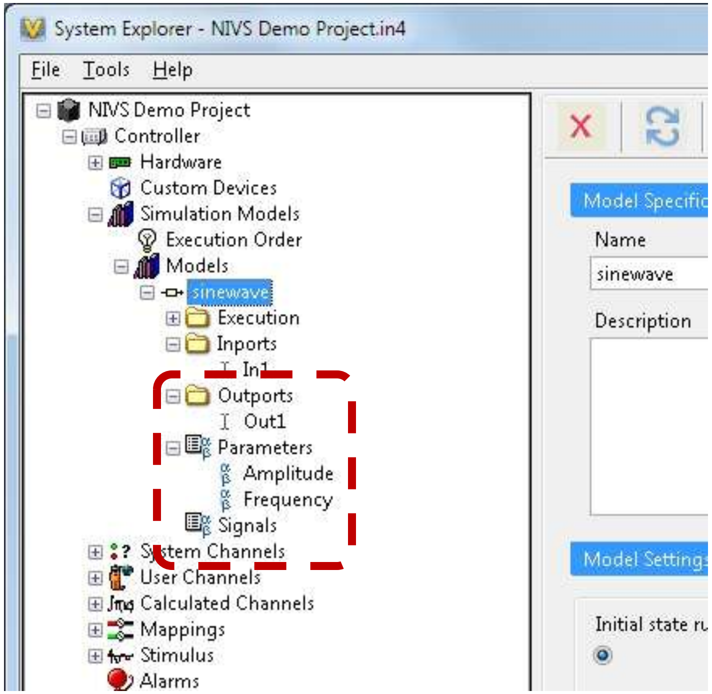


- Workspace Automation API
- System Explorer API

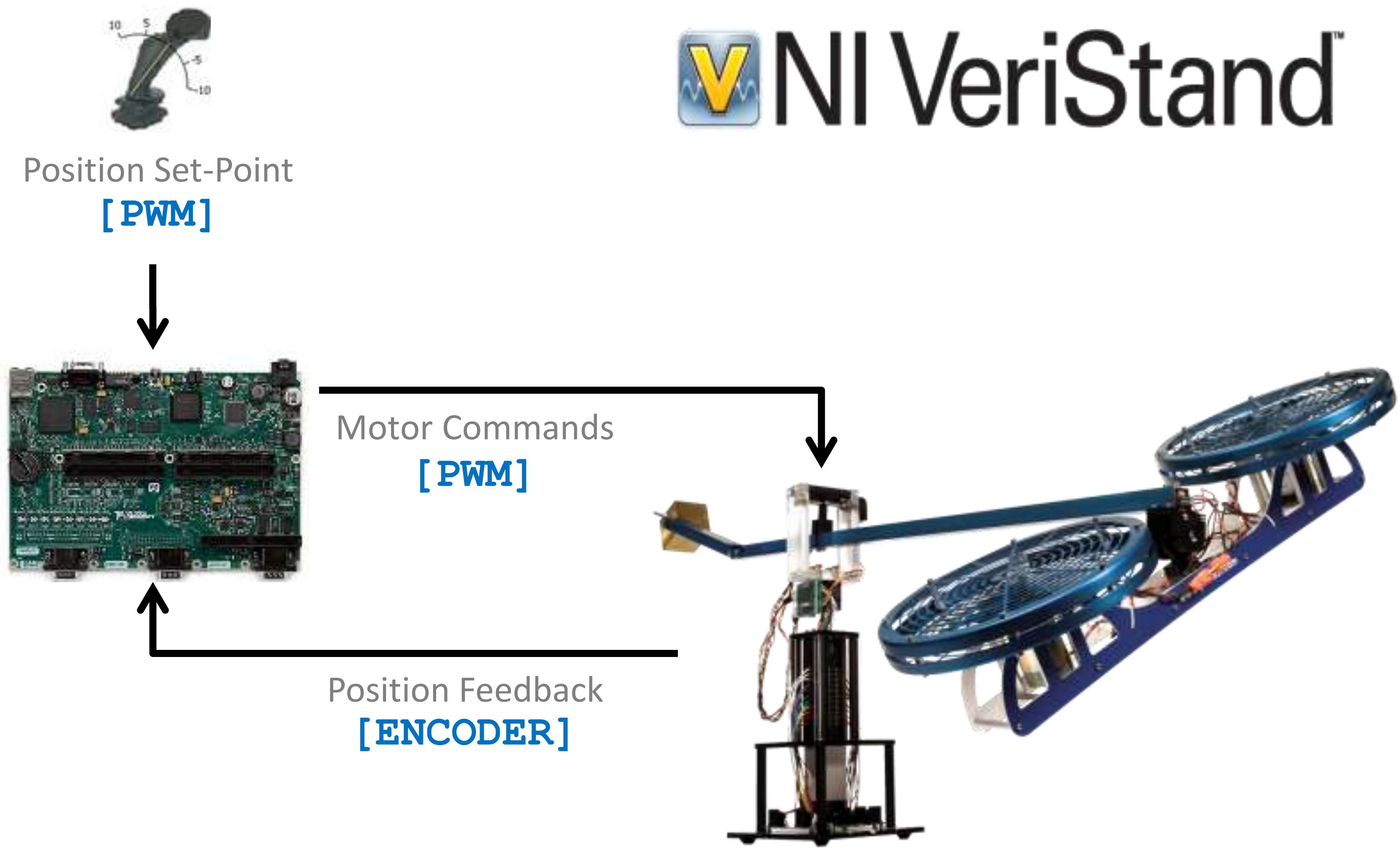
NI VeriStand Model DLLs



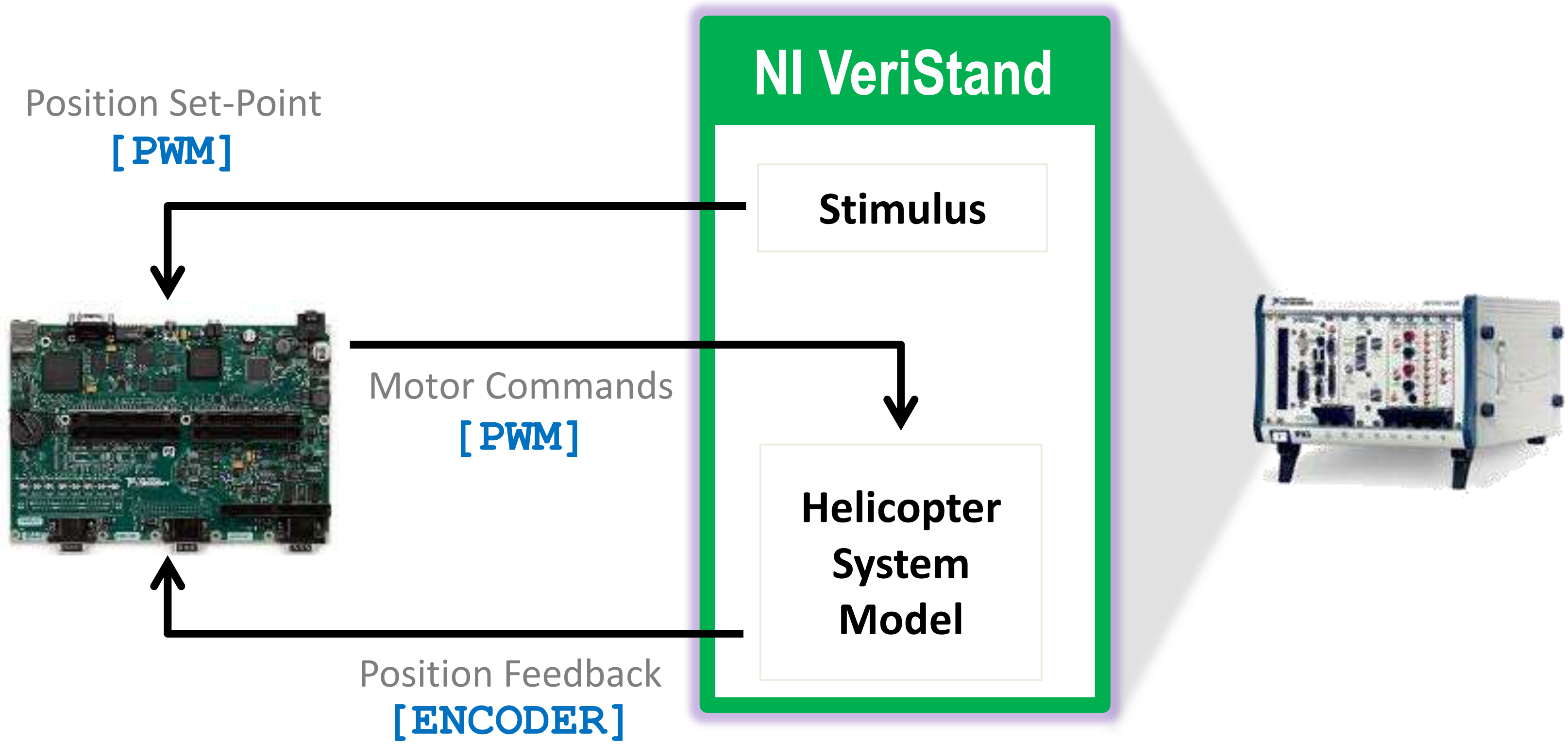




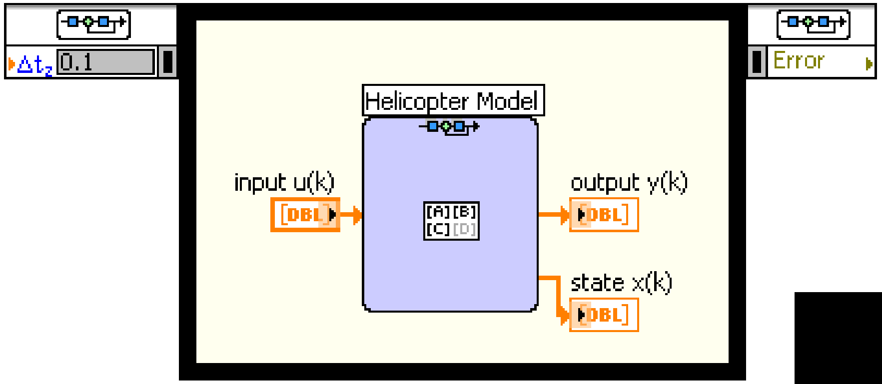
Today's Session– Helicopter Demo



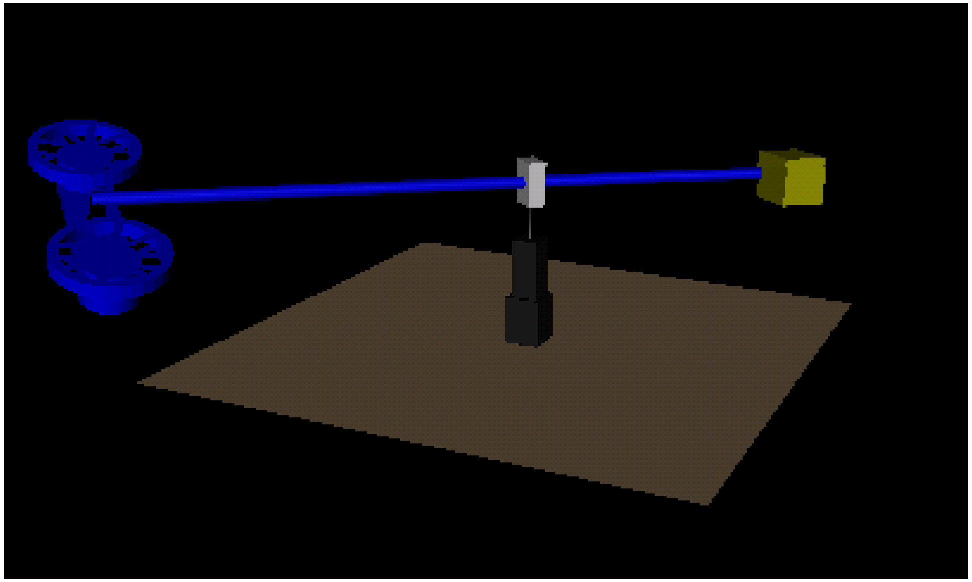
Helicopter Controller HIL Test System



Helicopter model

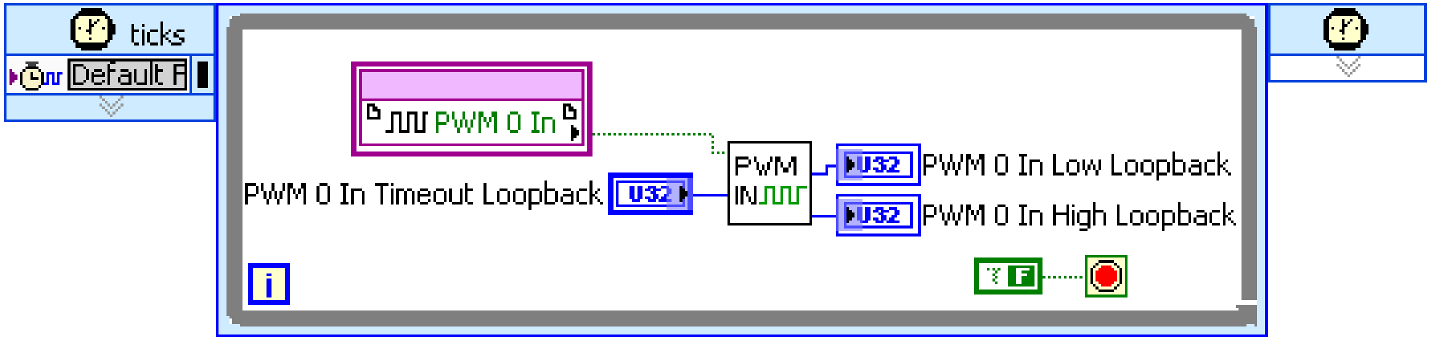


Helicopter model

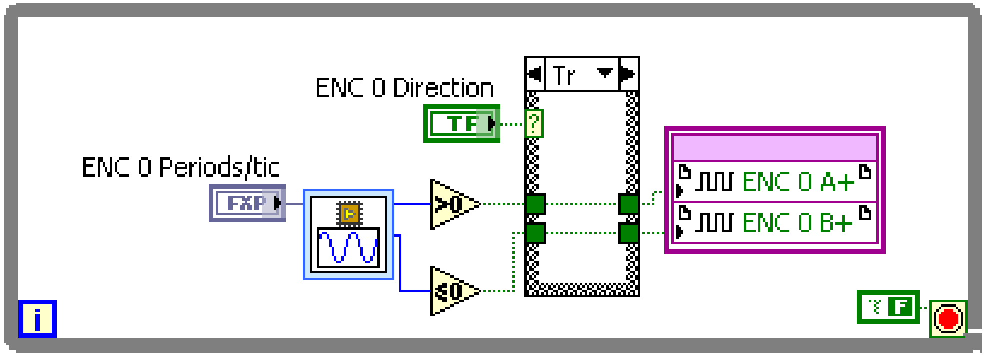


Helicopter 3D visualization

FPGA personality



FPGA PWM Input Code



FPGA Encoder Output Code

Channel Mappings

Source	Destination
[FPGA] Left Fan Power	Motor Voltage 1 [model]
[FPGA] Right Fan Power	Motor Voltage 2 [model]
[model] Elevation speed (deg/sec)	Elevation Velocity [FPGA]
[model] Pitch speed (deg/sec)	Pitch Velocity [FPGA]
[model] Travel speed (deg/sec)	Travel Velocity [FPGA]

Demo Part Overview

Part 1: Configure Input and Output

- Add FPGA-based I/O interface

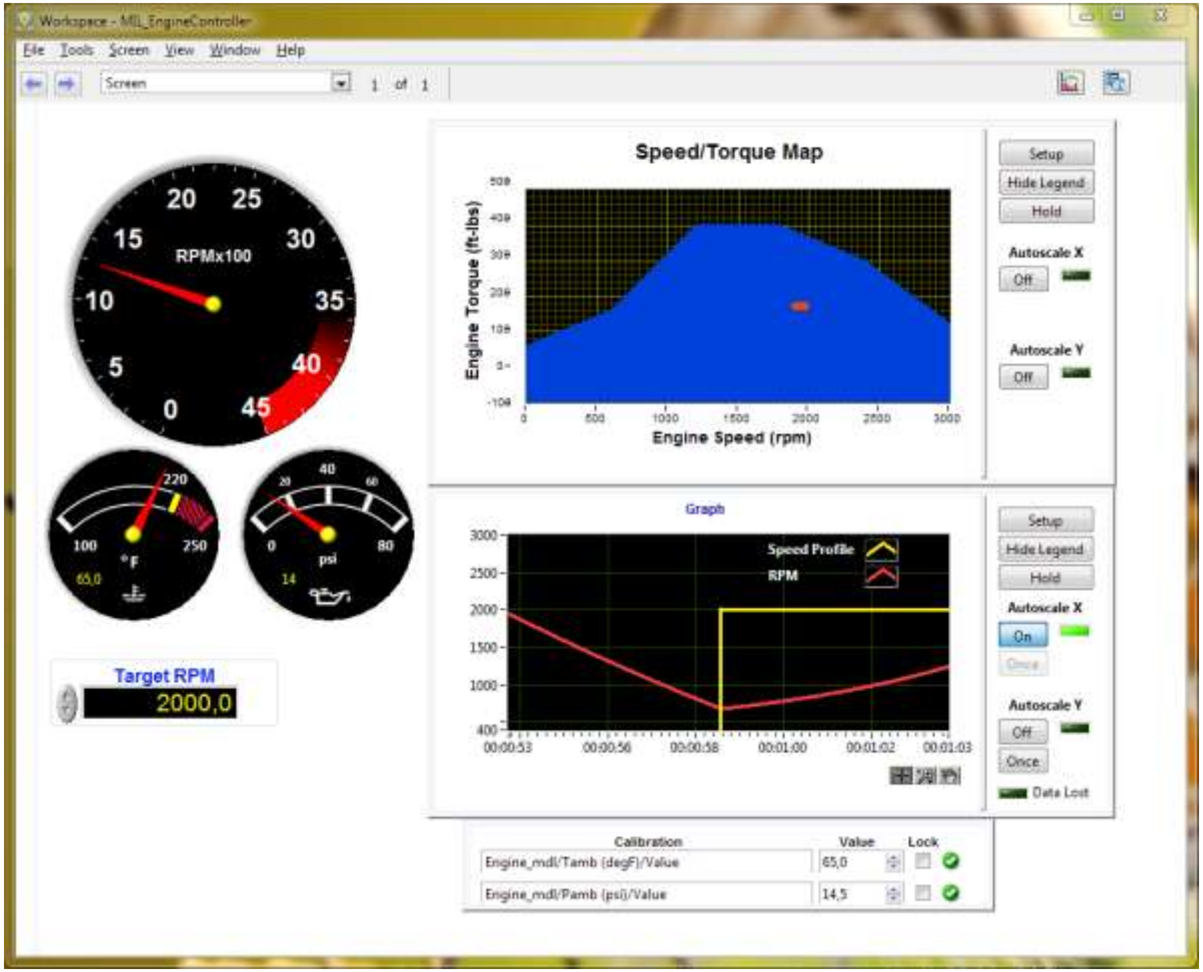
Part 2: Create HIL Test Systems

- Add System Model
- Create Procedure to initialize the unit under test
- Update User Interface

Part 3: Create Real-Time Stimulus Profile

- Create Stimulus Profile
- Configure Logging Task

Engine Controller Demo



Summary

- System-level Real-Time Testing
- Ease of Use
- Quick Prototype Development
- Customization

NI VeriStand

Additional Resources

ni.com/veristand

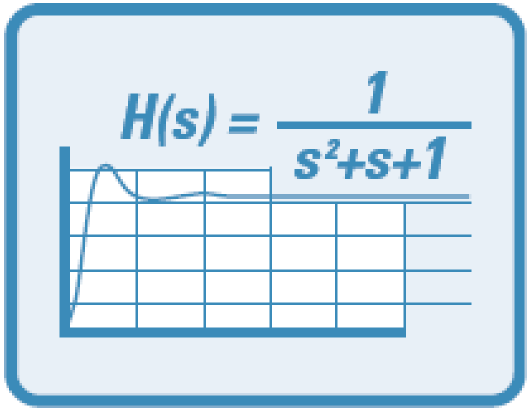
- Demonstration Videos
- Getting Started Resources
- White Papers
- Add-Ons
- Downloadable Evaluation Version

SW PRO SIMULACE A MODELOVÁNÍ

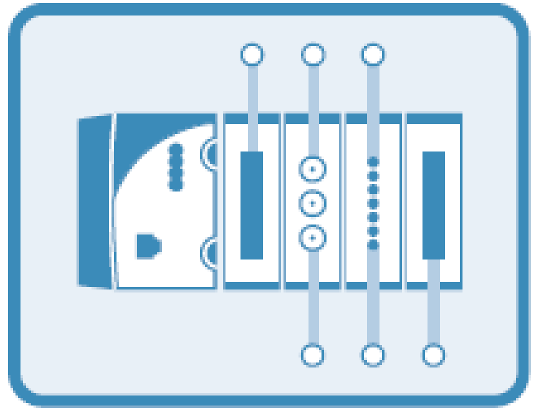
Agenda

- Graphical System Design
- Software Solutions for Creating Effective Control Algorithms
  - System identification
  - Control design
  - Importing models from other development systems
- LabVIEW RT overview
- LabVIEW FPGA overview
- LabVIEW Robotics

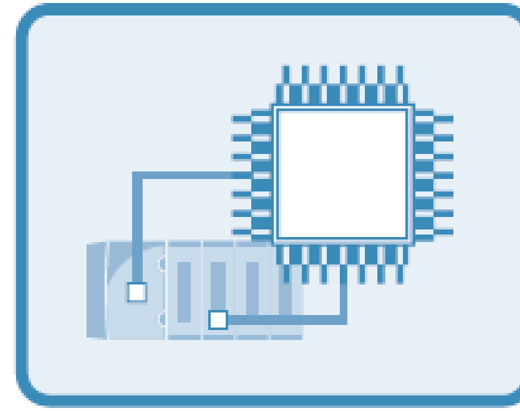
Graphical System Design



Design



Prototype



Deploy

Interactive Algorithm Design

- Control design
- Dynamic system simulation
- Digital filter design
- Advanced mathematics

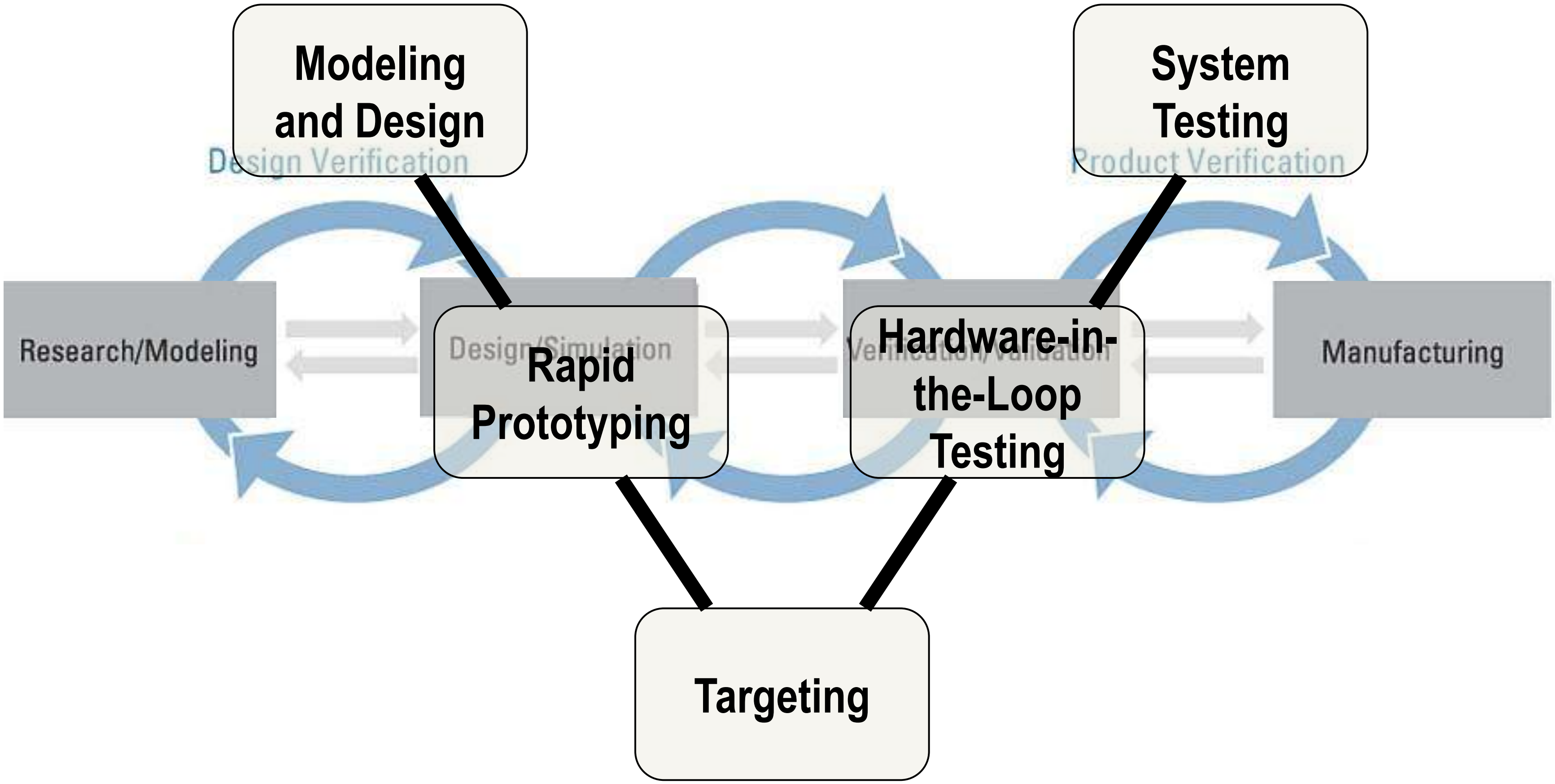
Tight I/O Integration

- I/O modules and drivers
- COTS FPGA hardware
- VHDL and C code integration
- Design validation tools

Deployable Targets

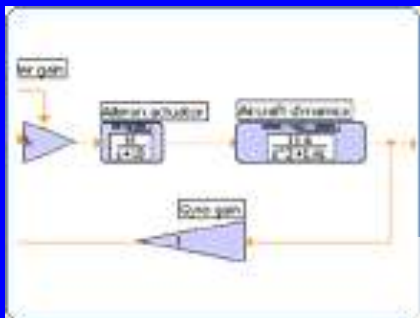
- Rugged deployment platforms
- Distributed networking
- Human-machine interfaces
- Custom designs

PC-Based Control and Simulation

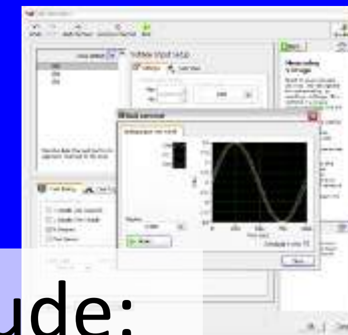


Algorithm Design Software Tools

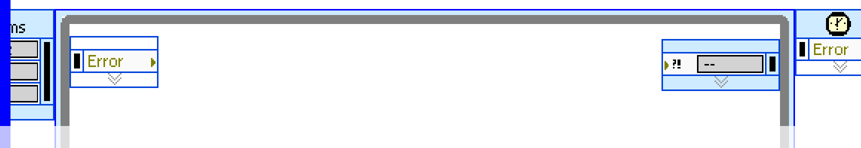
Simulation



Configuration

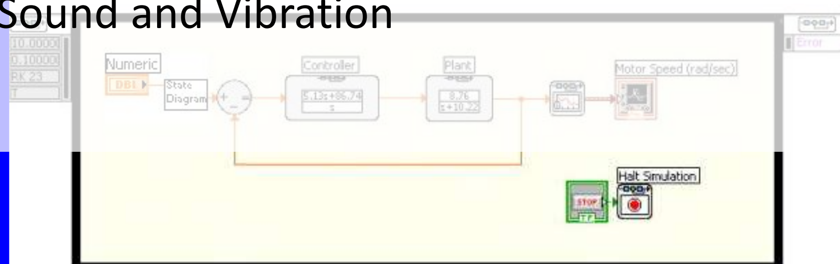


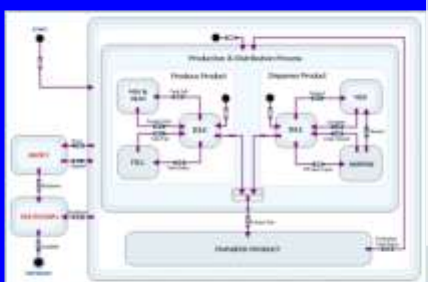
Graphical Dataflow



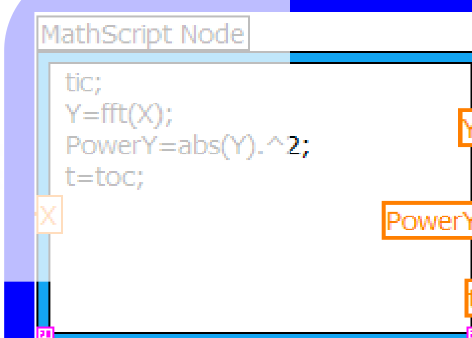
Built-in Design Libraries include:

- Advanced Math, Analysis and Signal Processing
- Control Design
- Digital Filter Design
- Modulation
- Sound and Vibration



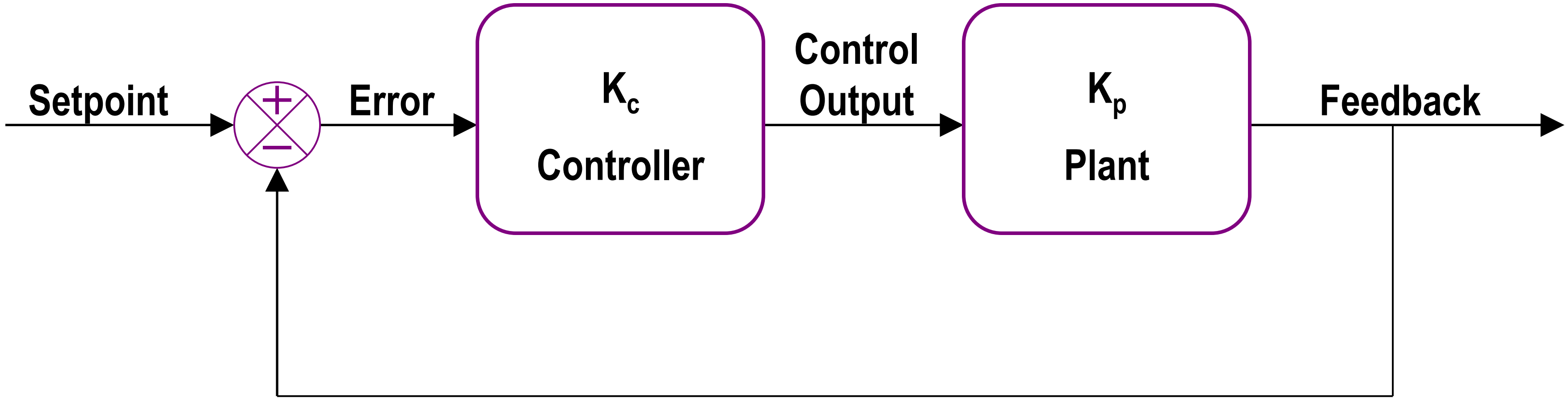


State Diagram

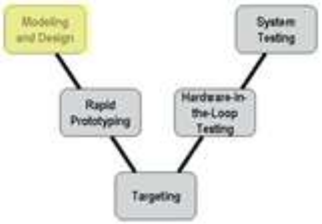


Textual Math

Modeling and Design

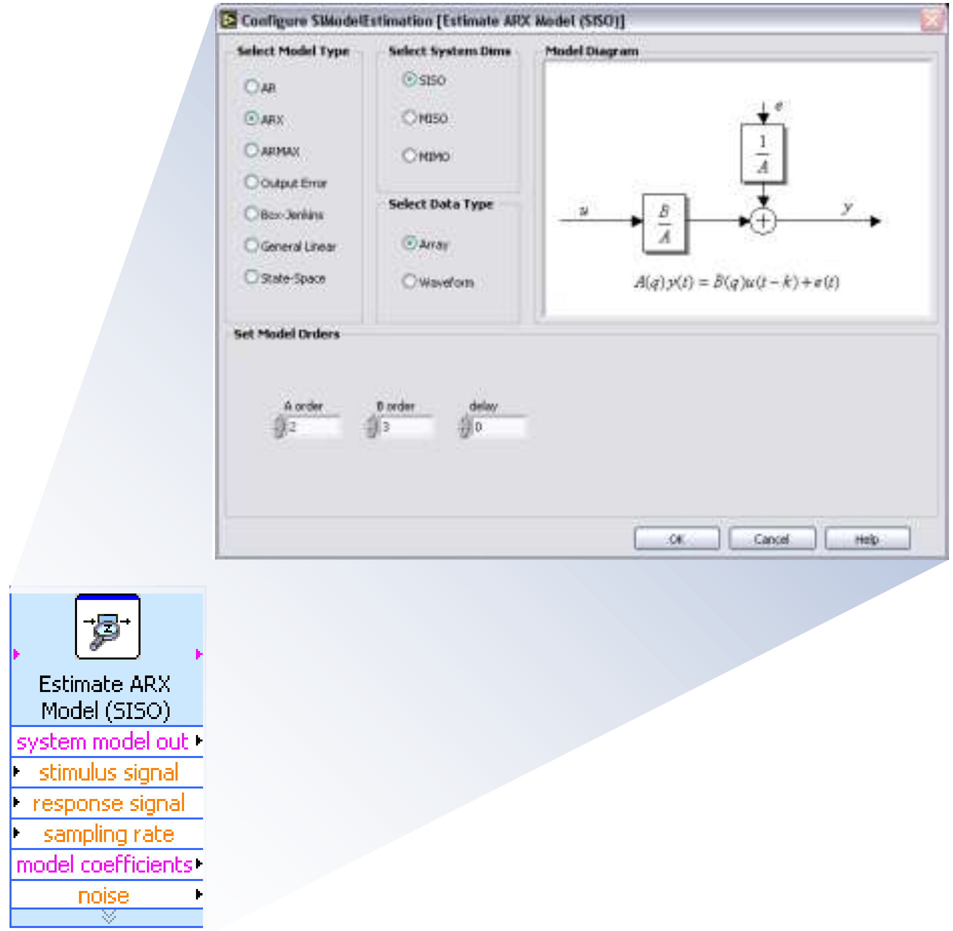


Modeling and design produce controller and plant models



LabVIEW System Identification Toolkit

- **Identify and validate linear models of systems from empirical data**
- **Seamless integration with NI I/O**
- Parametric model estimation (both SISO and MIMO)
- Nonparametric model estimation
- Recursive model estimation
- Data preprocessing
- Model conversion, validation, and presentation
- Closed-loop system identification with feedback detection
- Partially known “grey box” system identification



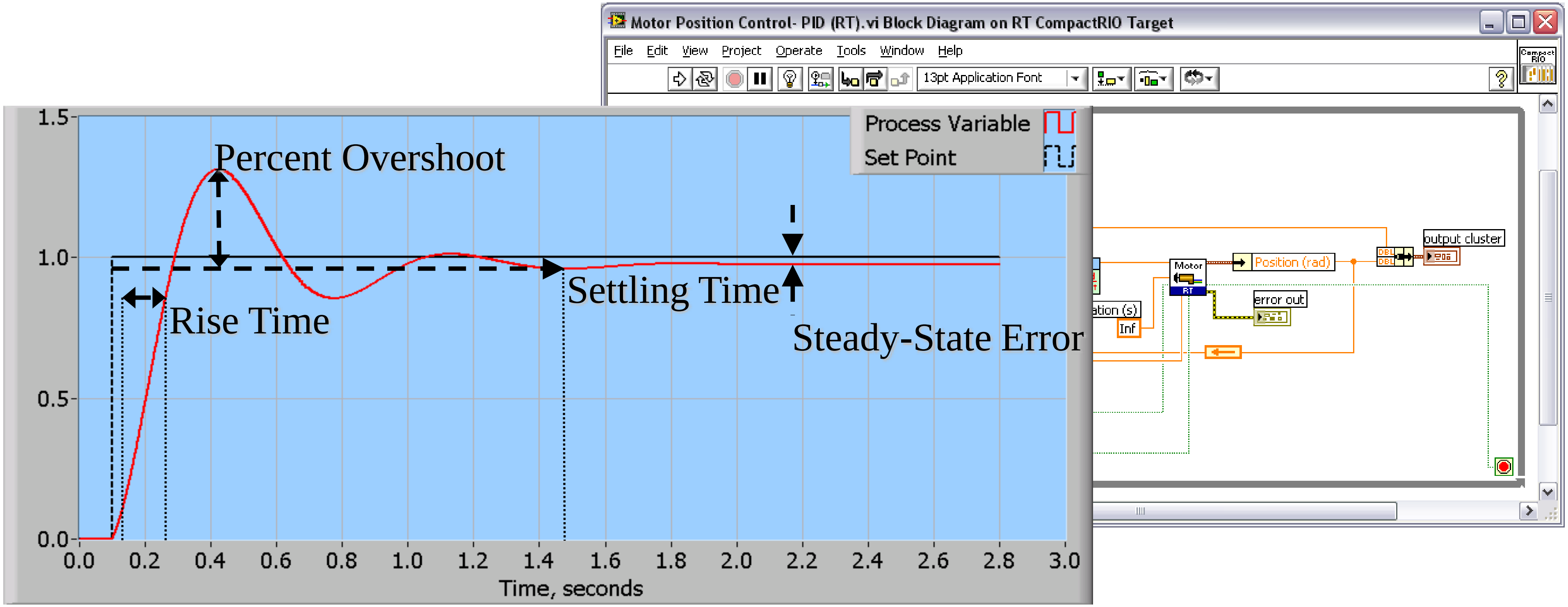
PID and Fuzzy Control Toolkit

PID Control

- Autotuning
- Gain scheduling

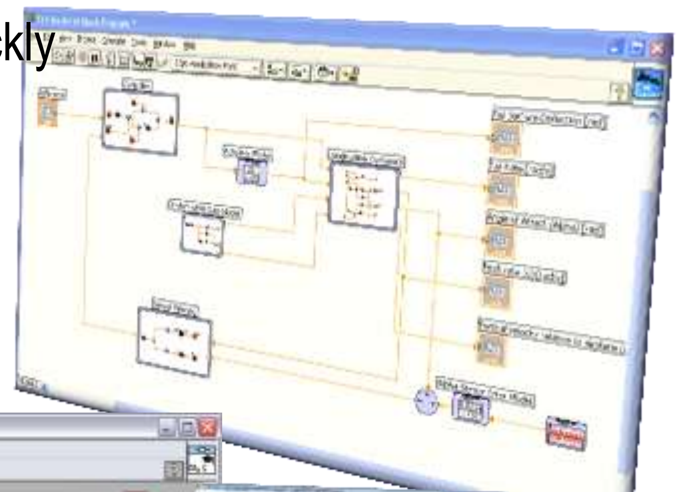
Fuzzy Logic

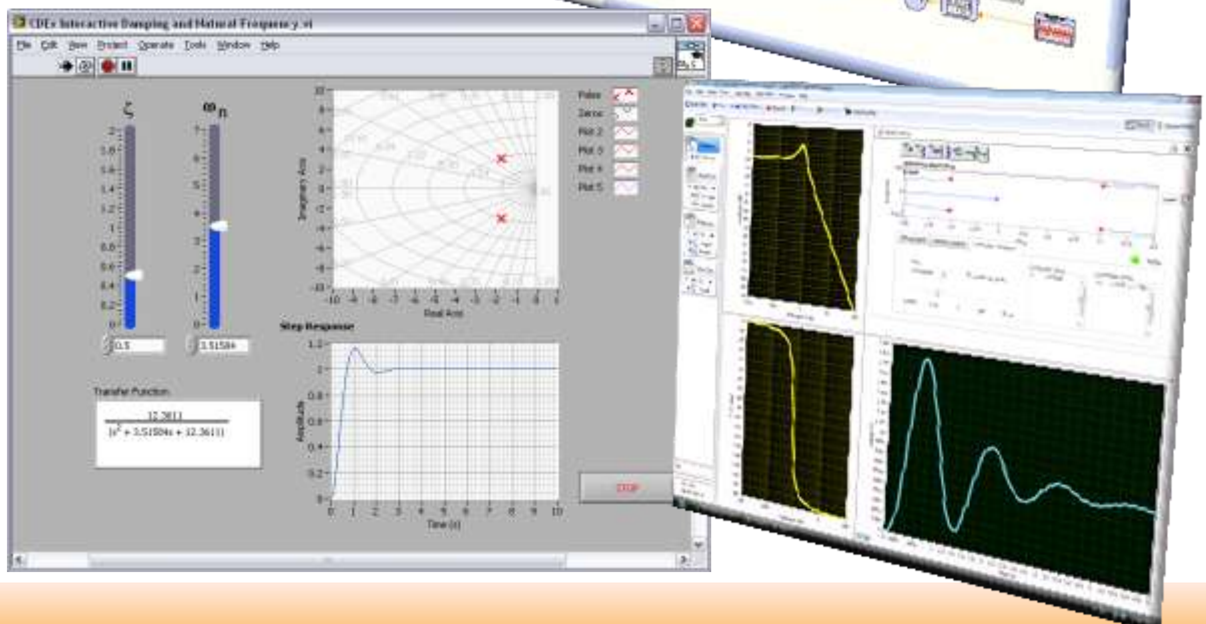
- Control strategies
- Decision making



LabVIEW Control Design and Simulation Module

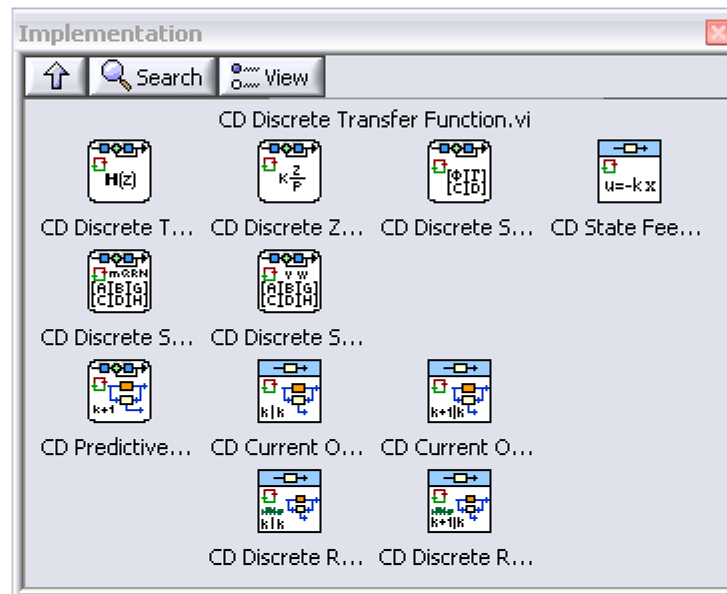
- Develop advanced control and analysis applications quickly
- Use on-board FPGA more efficiently
- Perform PID and non-linear control
- Leverage control capabilities including:
  - Model construction, conversion, interconnection and reduction
  - Time and frequency response
  - State-space model analysis and state feedback design
  - Kahlman filters
  - Simulation for linear and non-linear systems
  - Discrete systems
  - Signal arithmetic





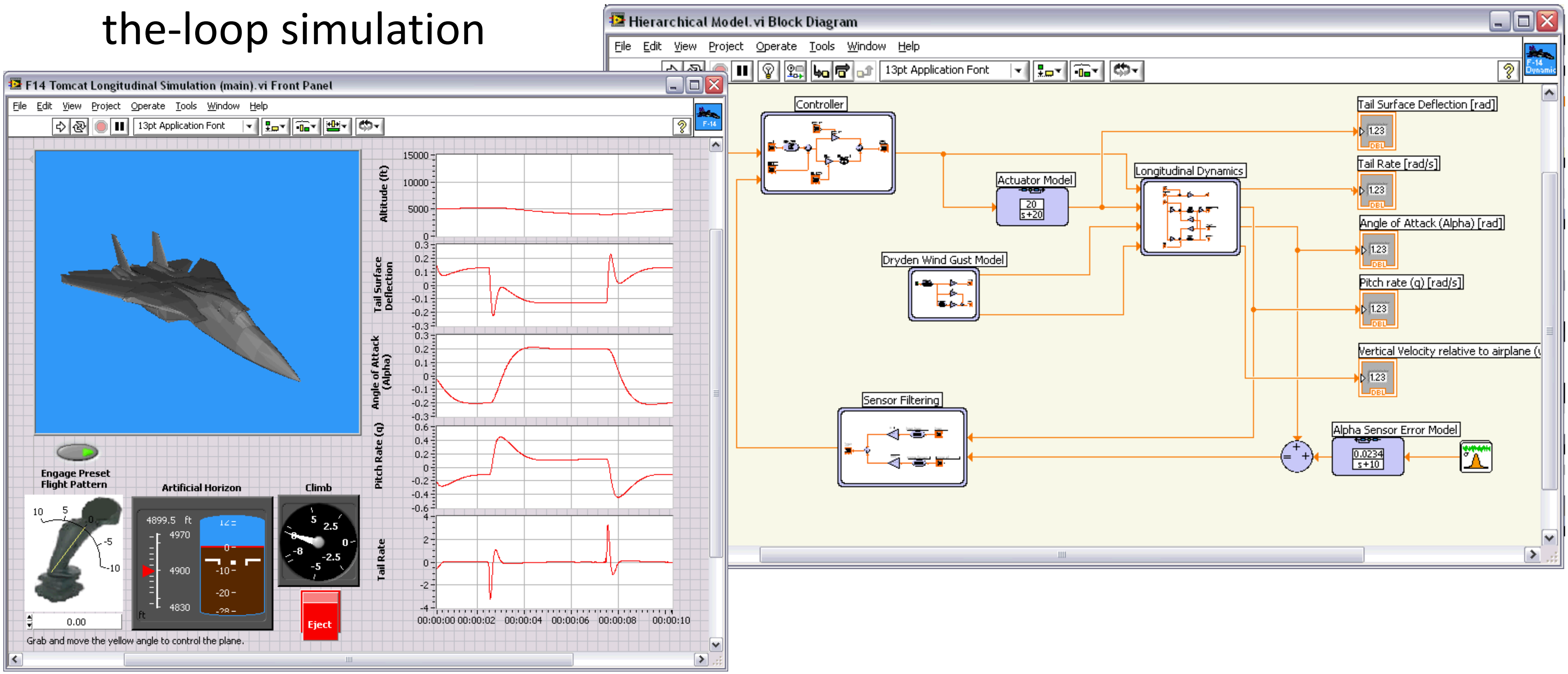
Control Design – Implementation VIs

- **Implementation VIs allow for control prototyping and deployment with LabVIEW Real-Time**
- Control Design Toolkit implementation VIs include:
  - Discrete transfer function, zero-pole-gain, and state-space models
  - State feedback controller
  - Discrete Kalman filter
- Can be used in:
  - LabVIEW Real-Time
  - LabVIEW Embedded projects

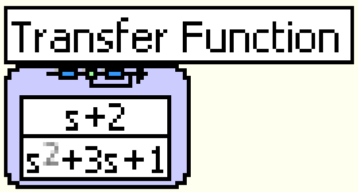


LabVIEW Simulation Loop

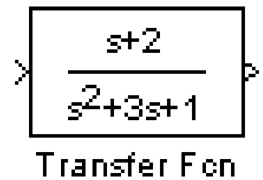
- Simulate dynamic systems including controllers and plants
- Real-time implementation for rapid control prototyping or hardware-in-the-loop simulation



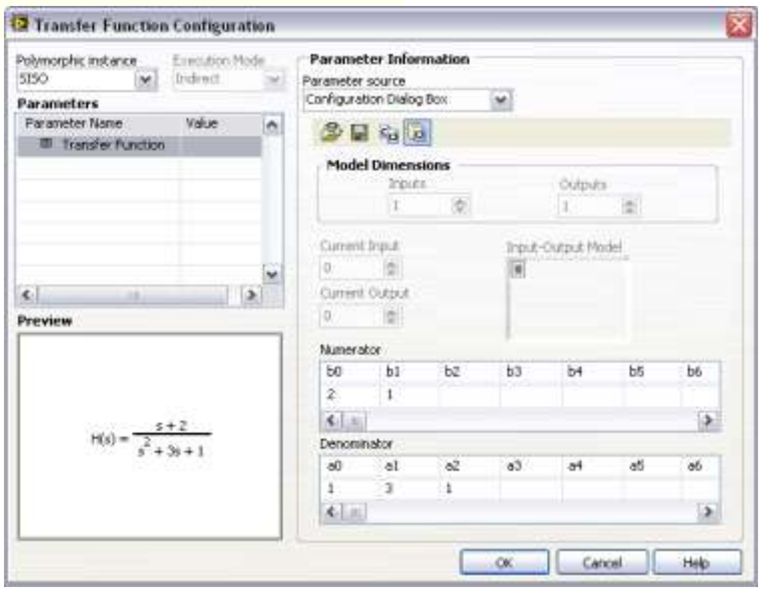
… Little or No Learning Curve for The MathWorks, Inc. Simulink® Software Users

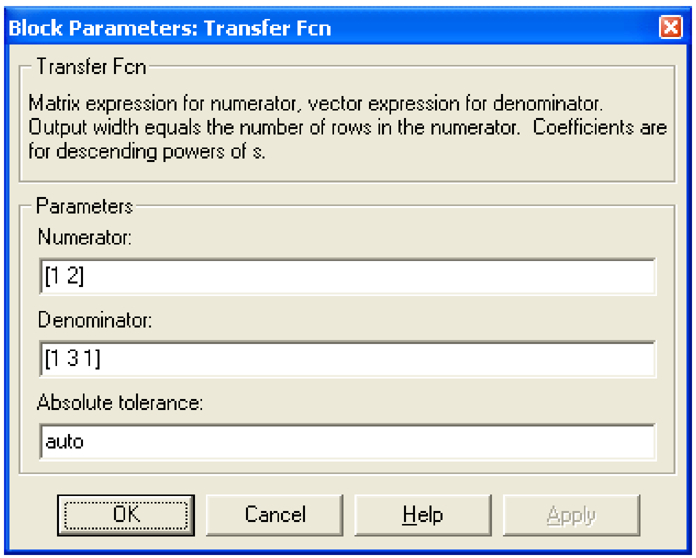


LabVIEW Simulation Module



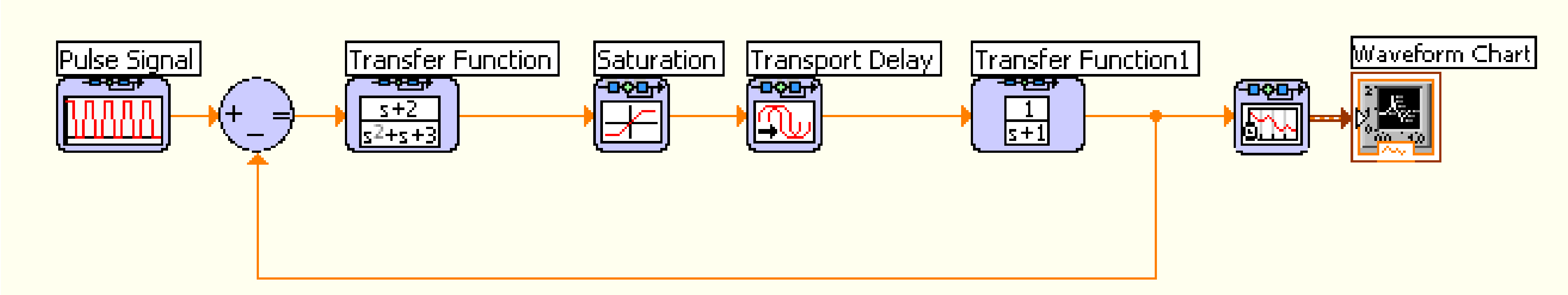
The Simulink® software environment



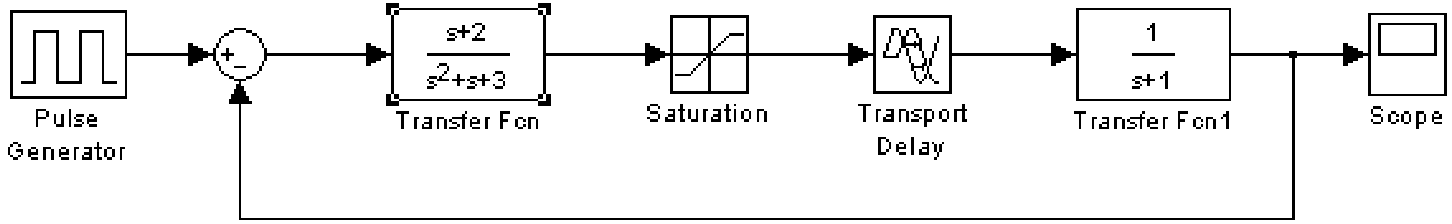


… Little or No Learning Curve for The MathWorks, Inc. Simulink® Software Users

- LabVIEW Simulation Module

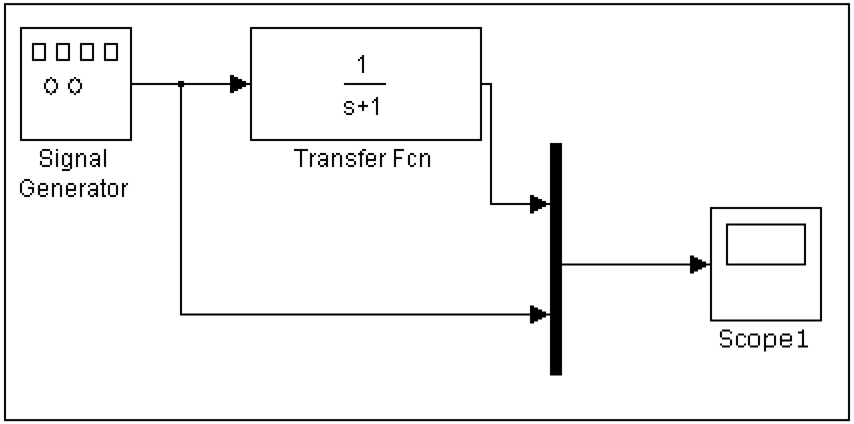


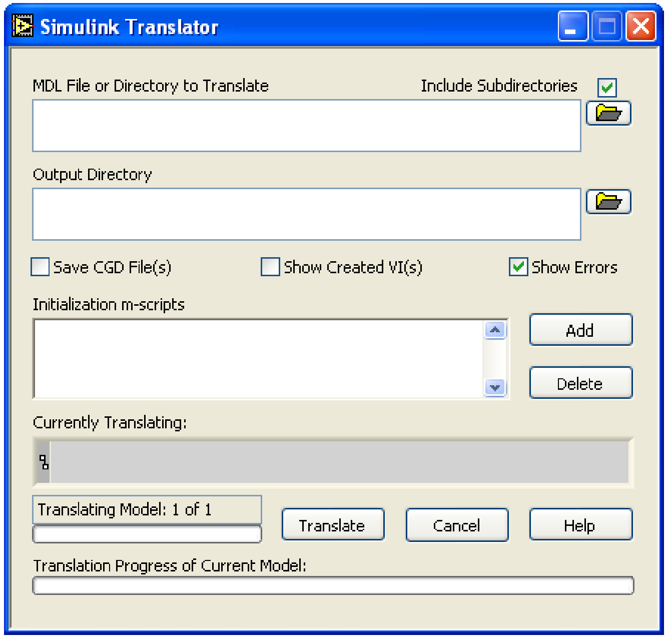
- The Simulink Software Environment

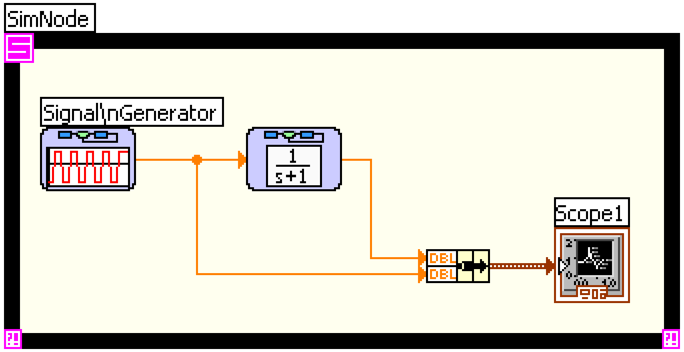


Simulink® Translator

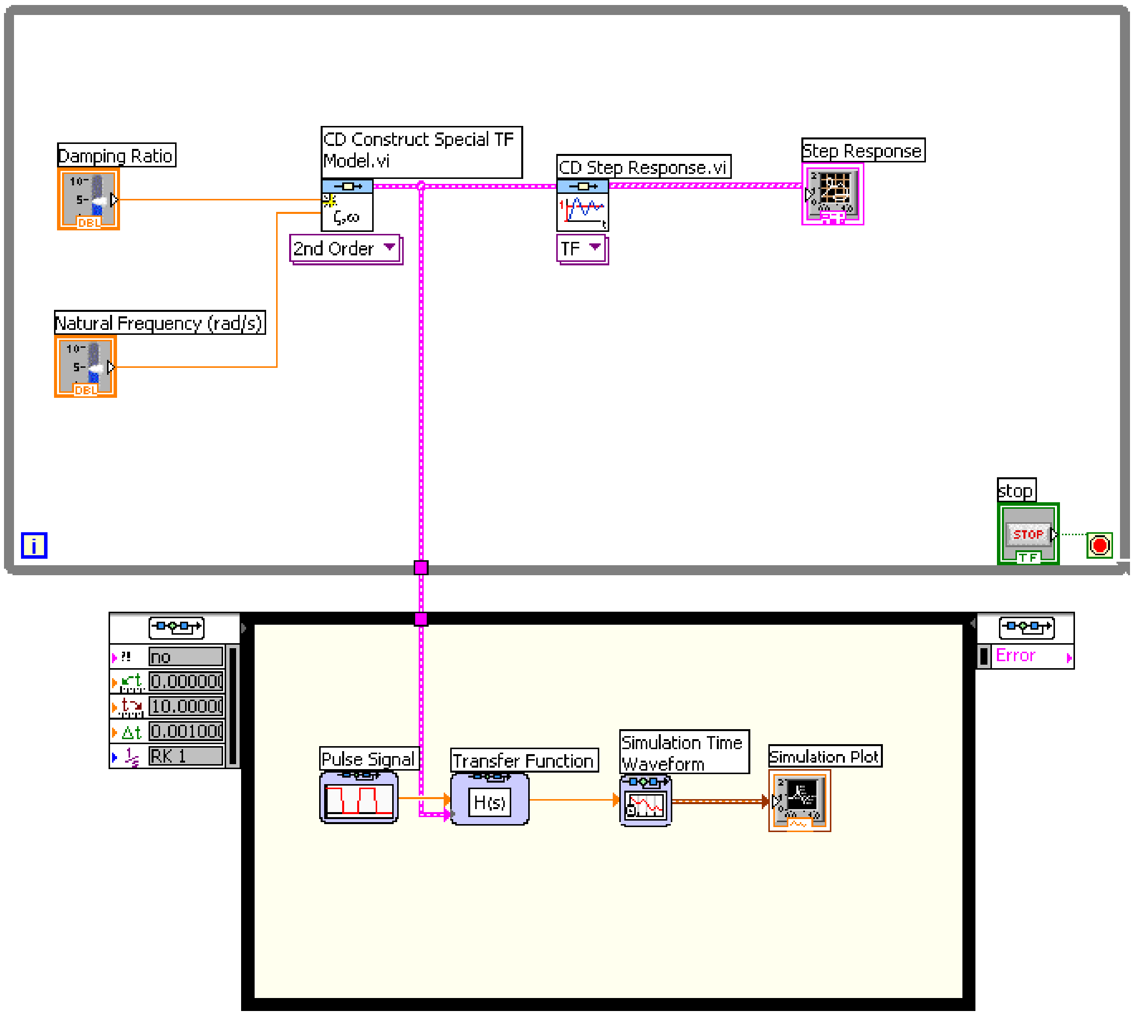
– Translate plant and controller models from Simulink® model file into LV code.







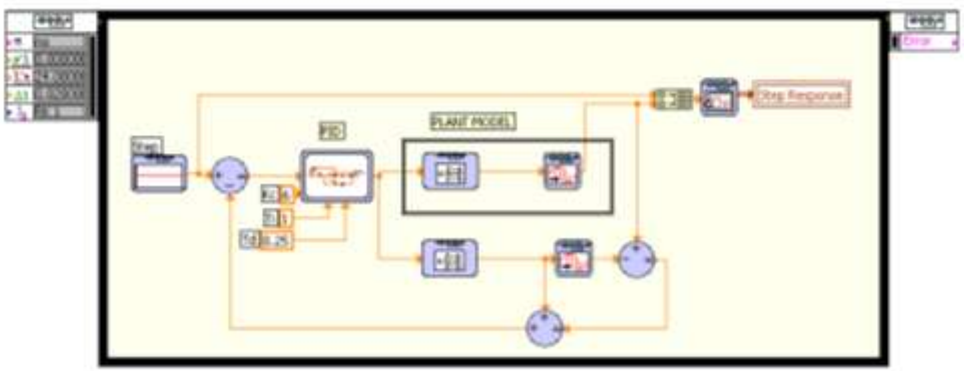
LabVIEW Simulation Looop Demo



DEMO

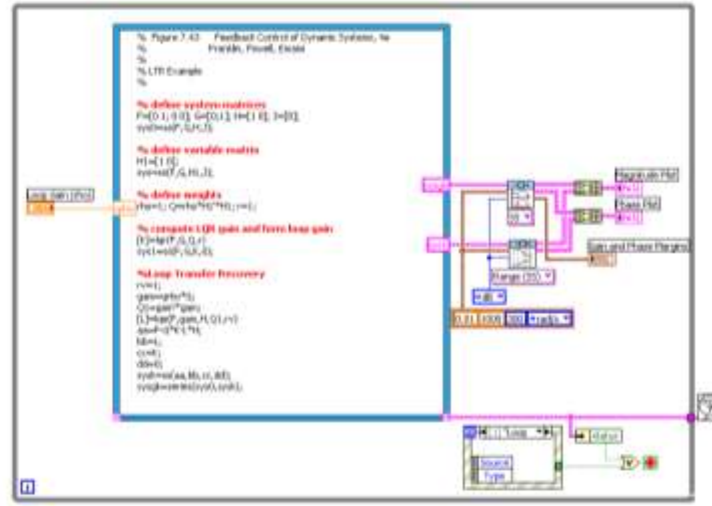
LabVIEW Control Design and Simulation Benefits

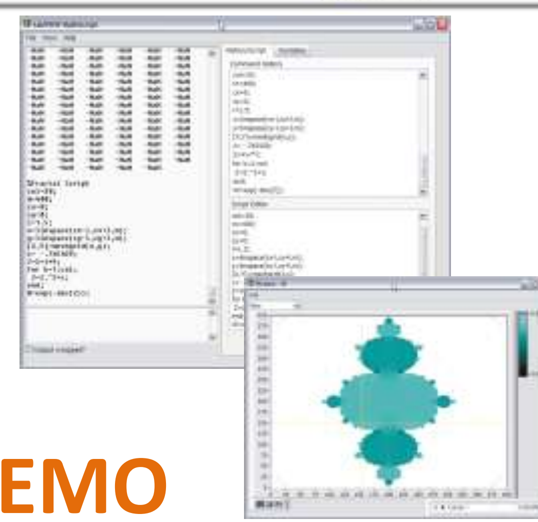
- ***Complete simulation and real-time implementation capability – stay in one environment from design to test to implementation***
- LabVIEW user interface to change and observe parameters as simulation or control system is running
- Use any LabVIEW VI or programming structure inside or outside of simulation loops:
  - Integrated design and simulation, batch simulation
  - DAQ, RIO, vision, or CAN for I/O and feedback
- Easily create parallel and multirate simulation or control loops

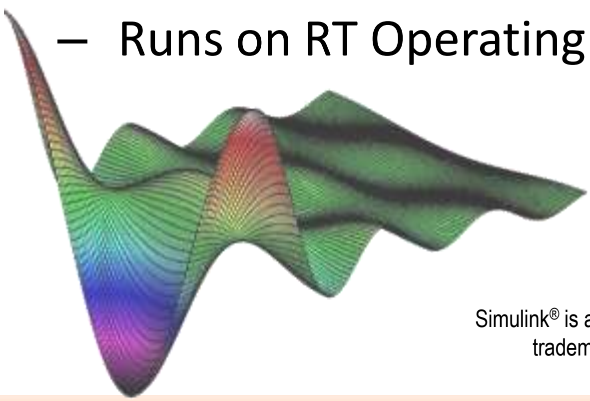


LabVIEW MathScript

- **Powerful textual programming for signal processing, analysis, and math**
  - More than 650 built-in functions
  - Reuse many of your m-file scripts created with The MathWorks, Inc. MATLAB® software and others
- **A native LabVIEW solution**
  - Interactive and programmatic interfaces
  - Does not require third-party software
  - Runs on RT Operating system



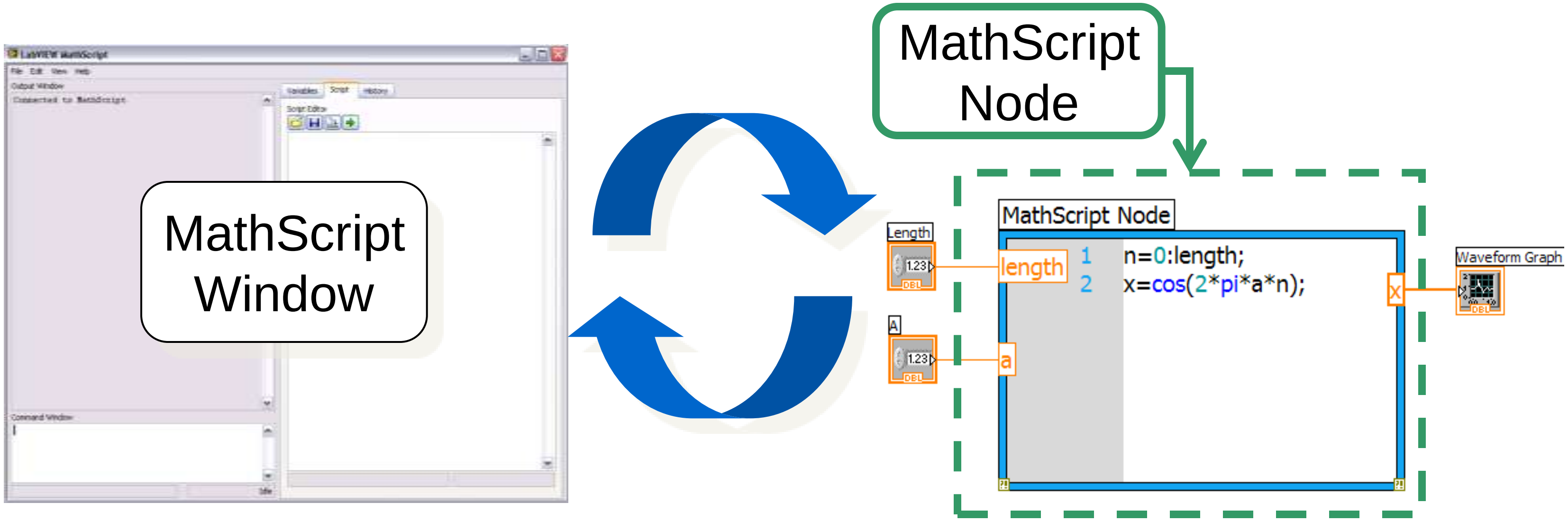




DEMO

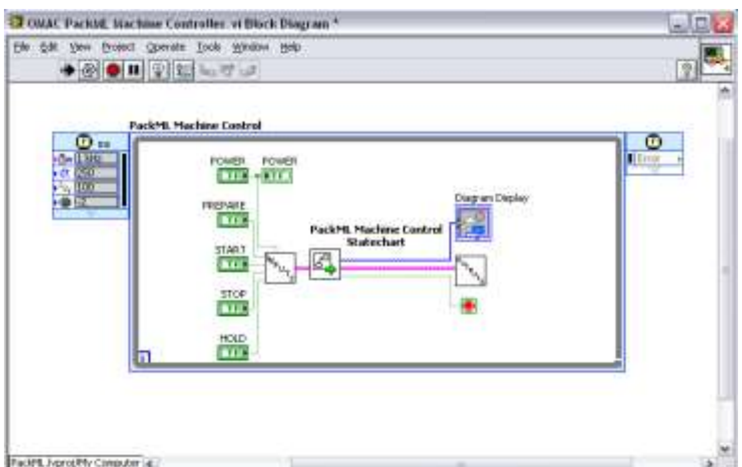
Working with LabVIEW MathScript

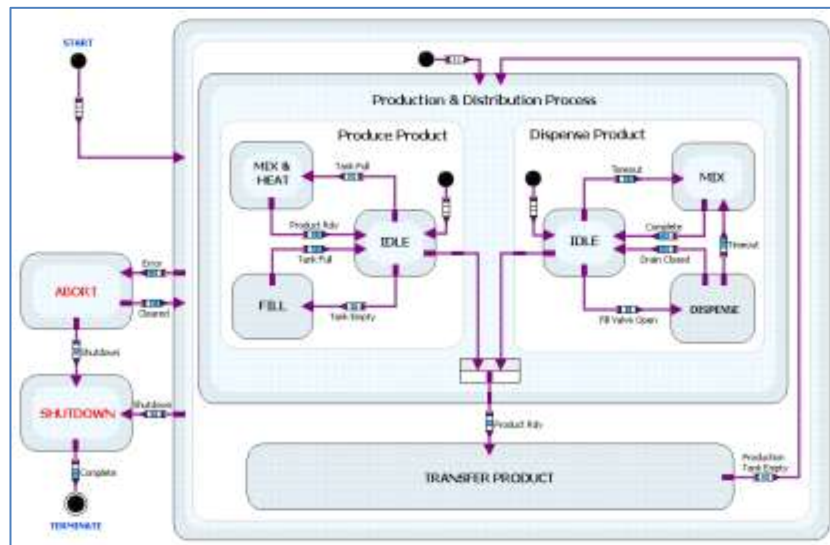
- Develop scripts interactively with the MathScript Window
- *Instrument your algorithms* by deploying with the MathScript Node
- Move back and forth as necessary to complete your work
- MathScript Window and MathScript Node share a global variable space



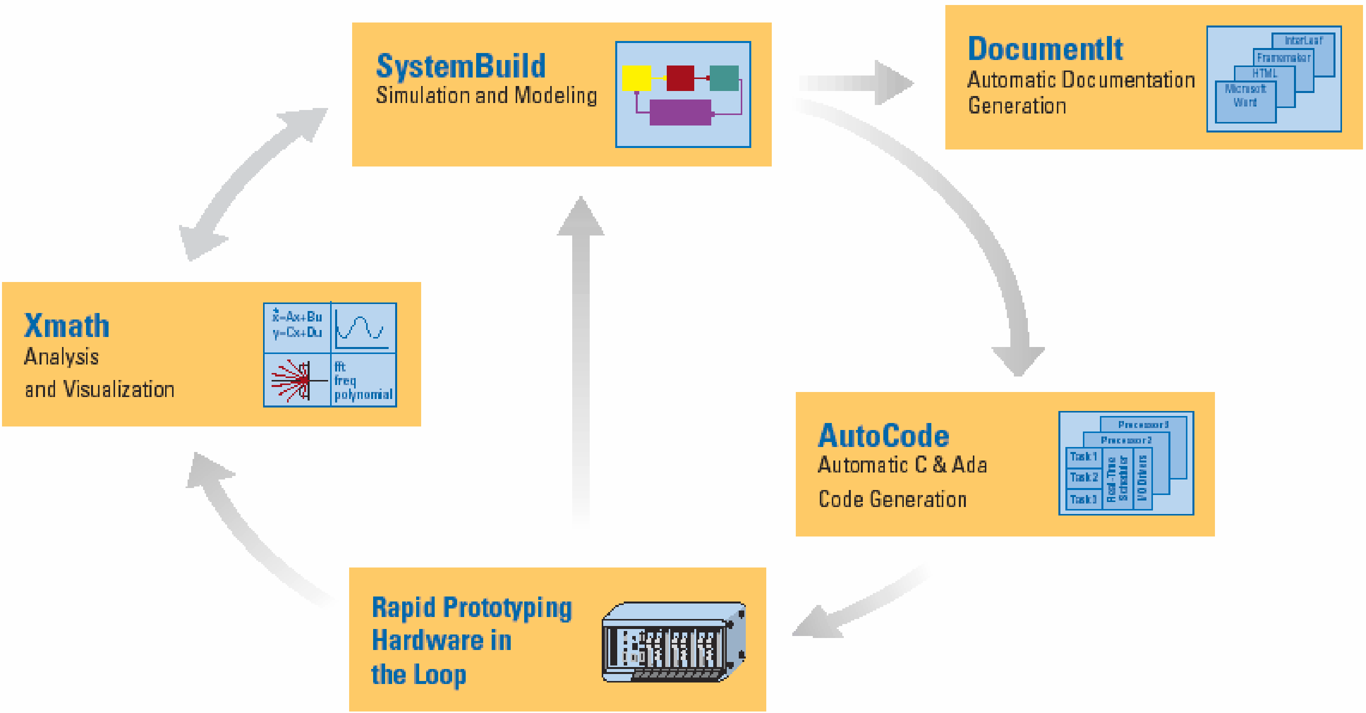
LabVIEW Statechart Module

- Statecharts provide high-level abstraction for state based applications
    - Simple semantics represent complex systems
    - Self-documenting design
- Integrate statecharts into existing LabVIEW applications
- Generate code for desktop, Real-time, FPGA, and embedded targets



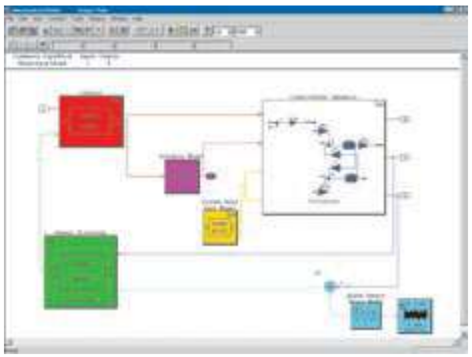


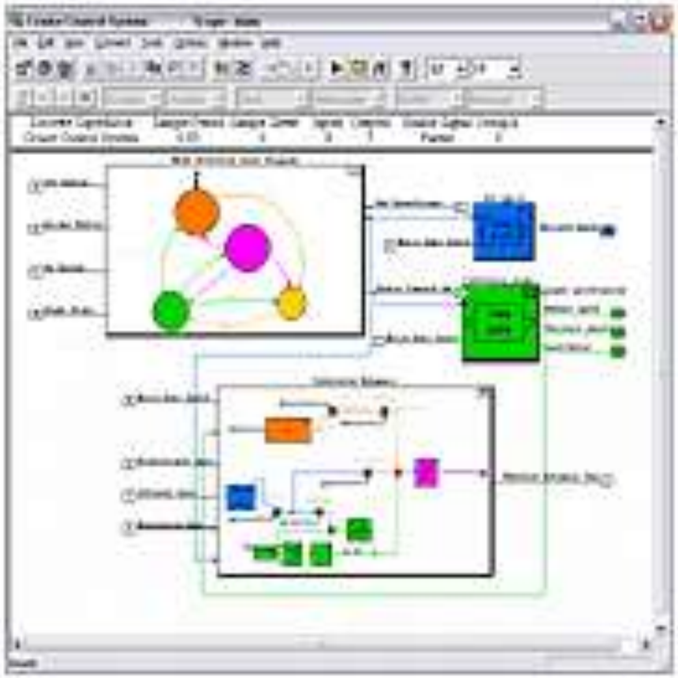
MATRIXx



- integrated suite of software tools for modeling and dynamic simulation, analysis, control design, and automatic code generation

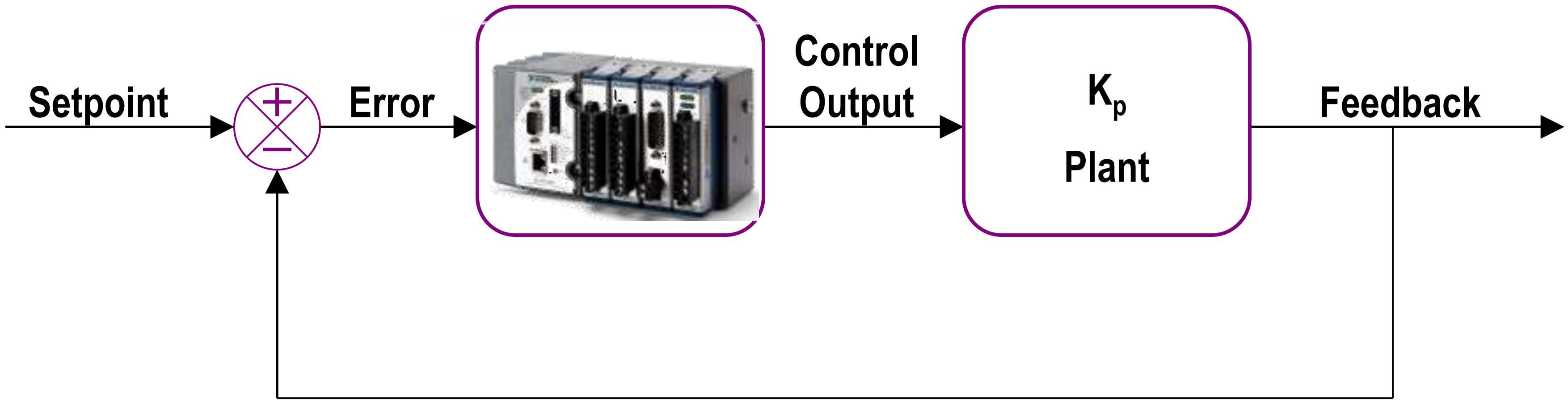
MATRIXx



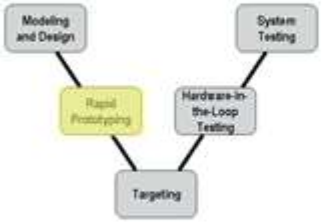


SystemBuild - graphical system modeling and simulation
Xmath - interactive analysis, visualization, and control development
DocumentIt - automatic documentation generation
AutoCode - automatic embedded code generation for C and Ada

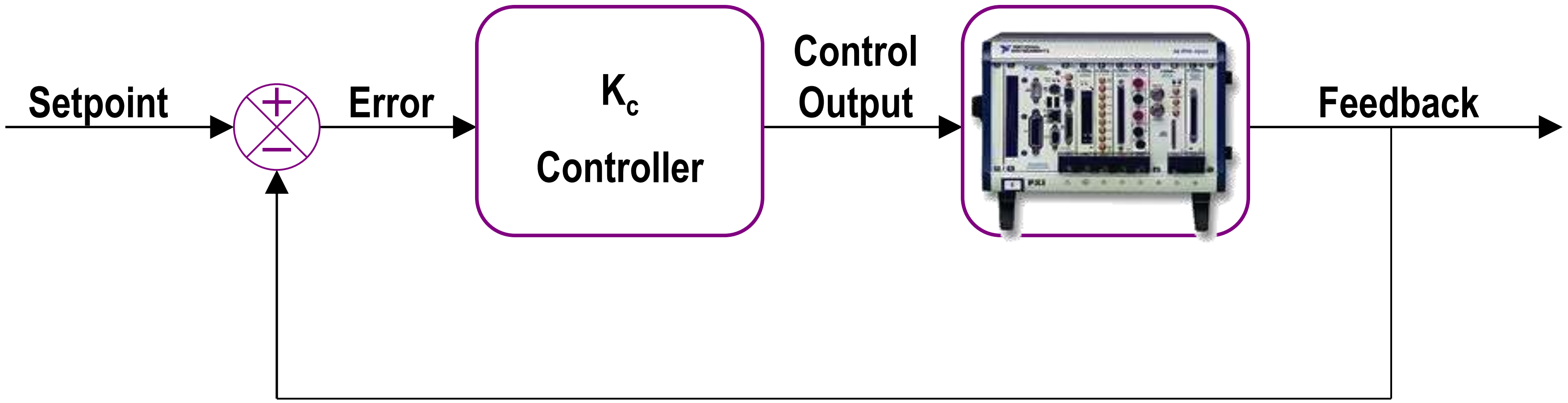
Rapid Control Prototyping (RCP)



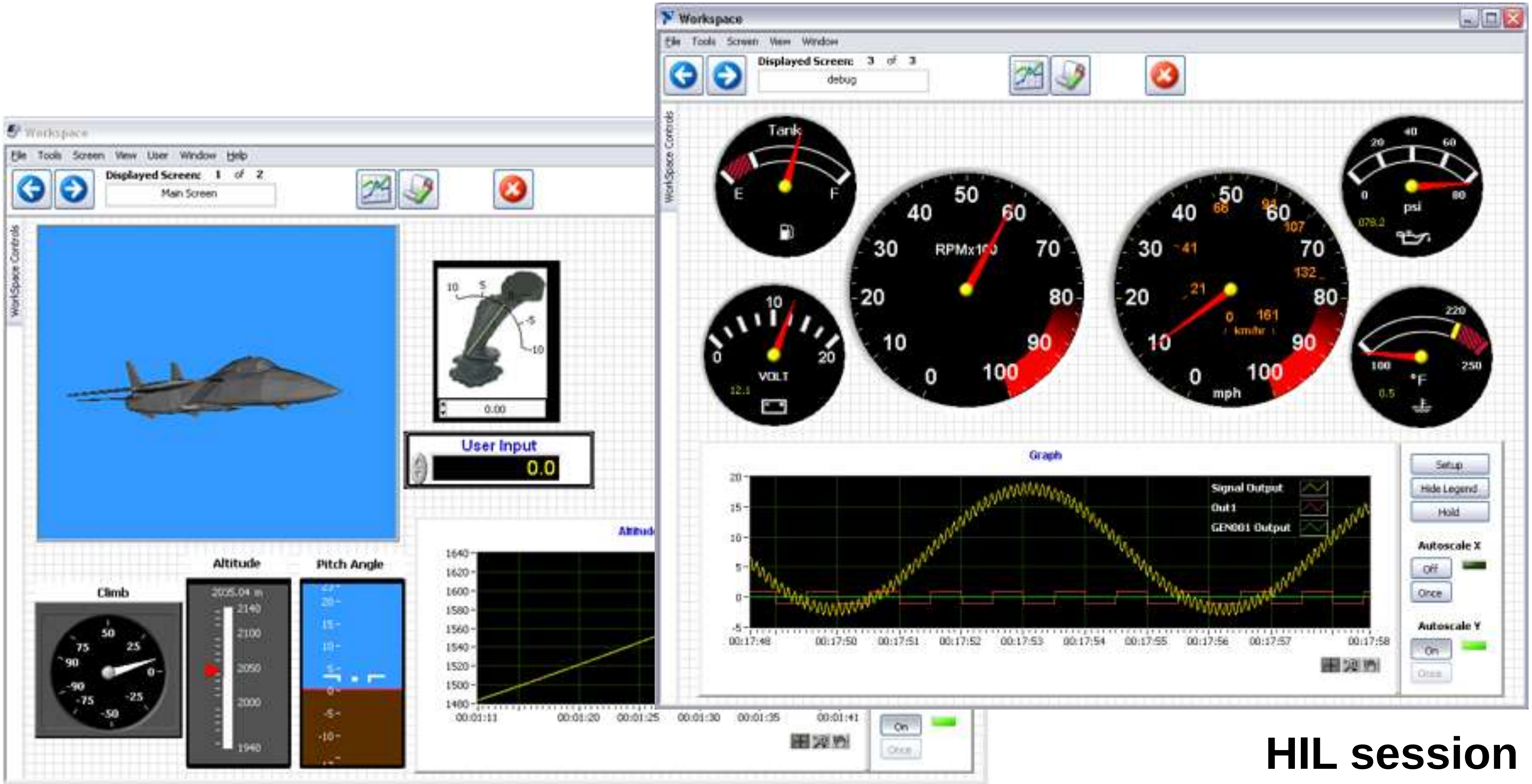
Creating a functional prototype of the controller



Hardware-in-the-Loop (HIL) Simulation



Testing controller with simulated plant

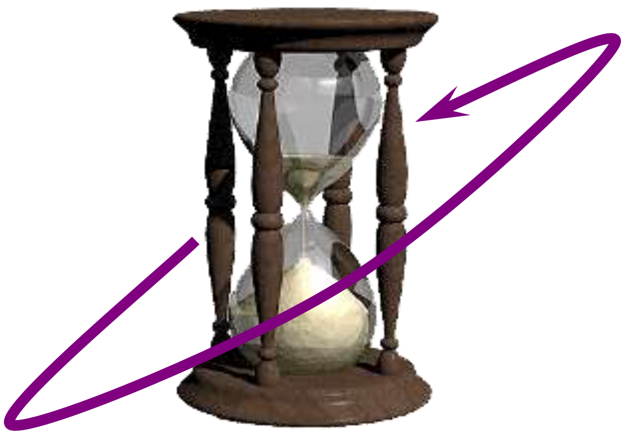


Workspace
File Tools Screen View Window
Displayed Screen: 3 of 3
debug
Tank
RPMx100
VOLT
mph
psi
°F
Graph
Signal Output
Out1
GEN001 Output
Setup
Hide Legend
Hold
Autoscale X
Autoscale Y
Workspace
File Tools Screen View User Window Help
Displayed Screen: 1 of 2
Main Screen
User Input
0.0
Climb
Altitude
Pitch Angle

HIL session

LABVIEW REAL TIME INTRODUCTION

- Real-time **does not** always mean real fast
- Real-time means **absolute reliability**
- Real-time systems have timing constraints that must be met to avoid failure
- Determinism is the ability to complete a task within a fixed amount of time



Operating System Characteristics

General Purpose OS

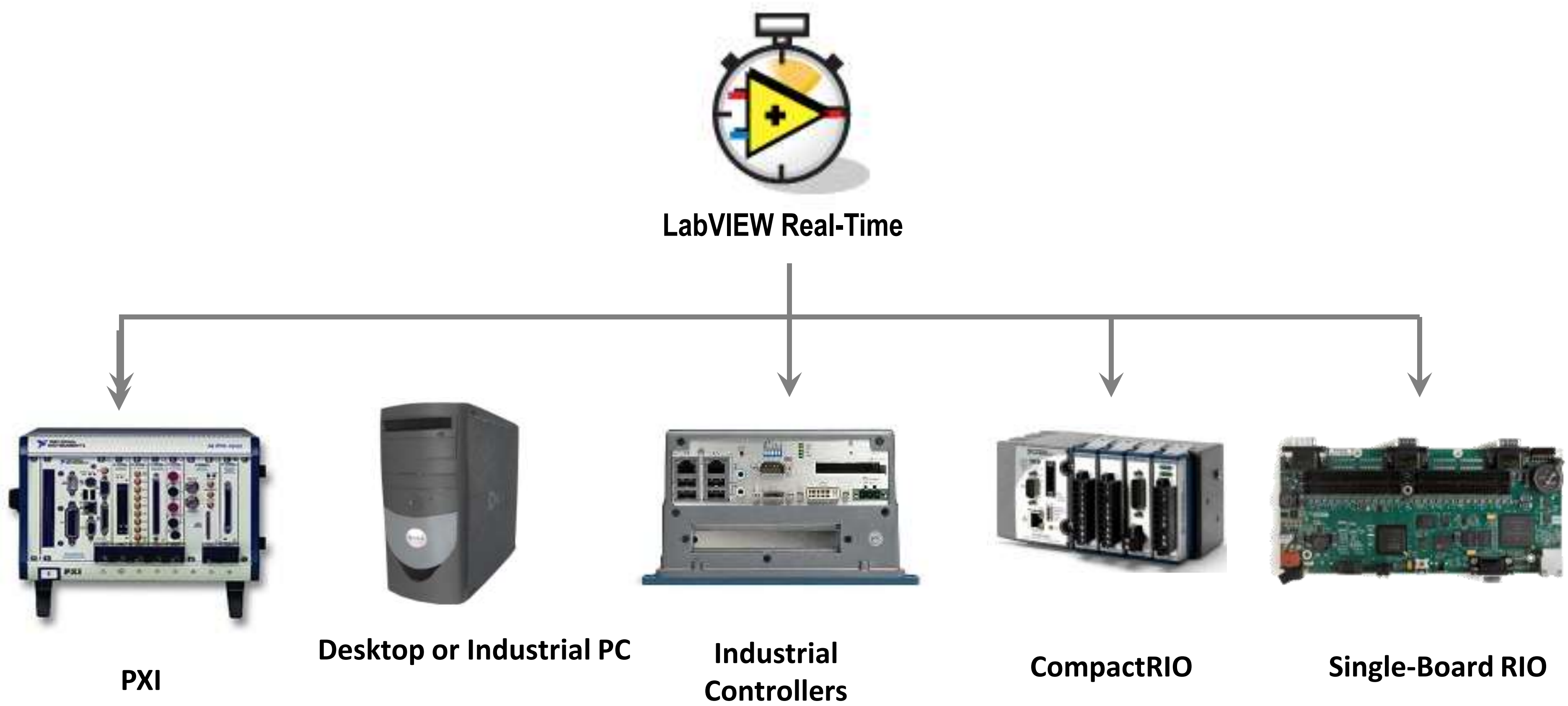
- High-priority tasks can be preempted by lower-priority tasks
- Extraneous background programs
  - Screen savers, disk utilities, virus software, etc.
- Peripheral Interrupts
  - Mouse, keyboard, etc.

Real-Time OS

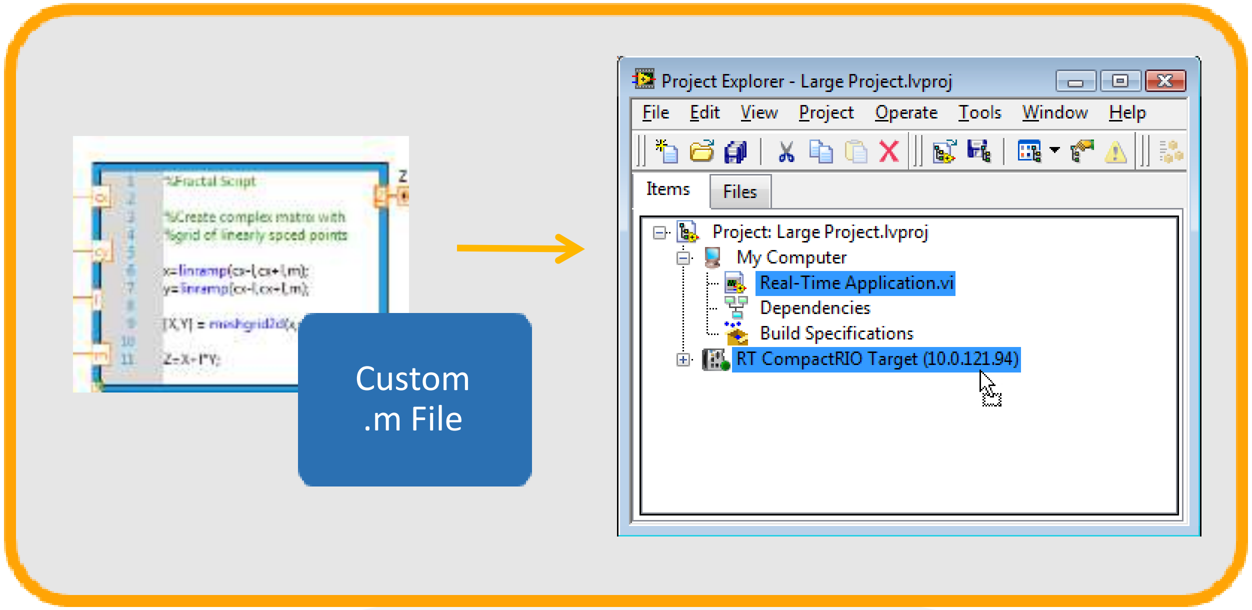
- Scheduler ensures high-priority tasks execute first
- Direct control over all tasks
- Stand-alone (no mouse, keyboard, etc.)

Loop Rate	Software Jitter
10-100 Hz	Unbounded
Up to 100 kHz	Bounded

LabVIEW Real-Time Hardware Targets



LabVIEW 2009 MathScript RT Module



Desktop

NI CompactRIO

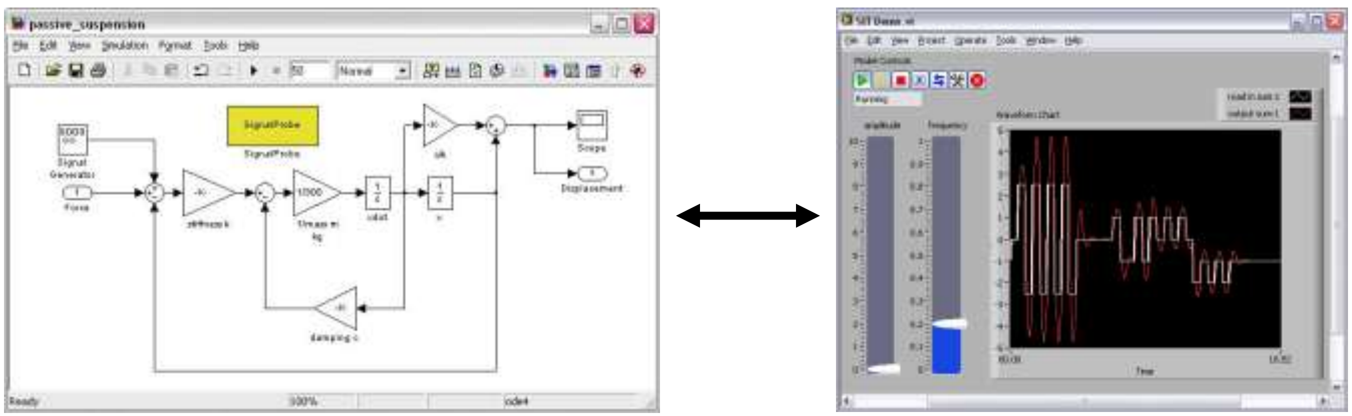
PXI

NI Single-Board RIO

NI Embedded Vision System

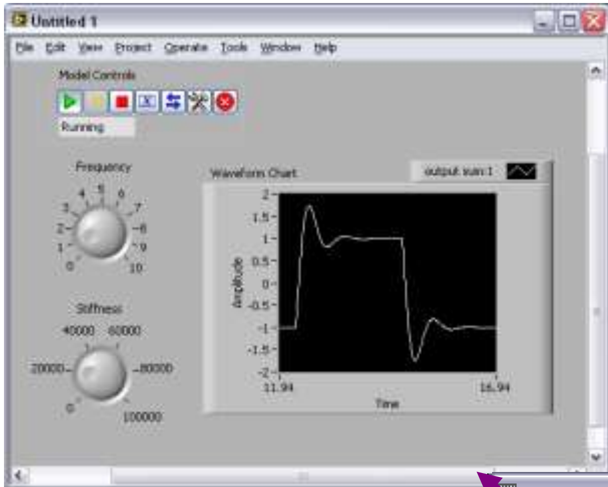
LabVIEW Simulation Interface Toolkit (SIT)

- Connect LabVIEW user interface to The MathWorks, Inc. Simulink® software to enable interaction with your model during simulation
- Connect your model to real-time IO for prototyping, deployment, and HIL simulation

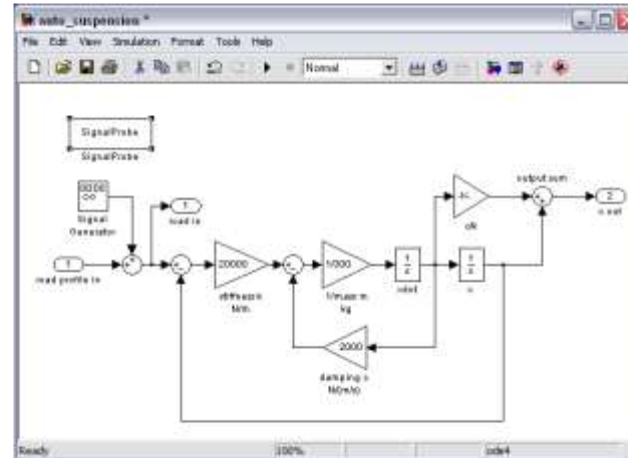


LabVIEW Simulation Interface Toolkit (SIT)

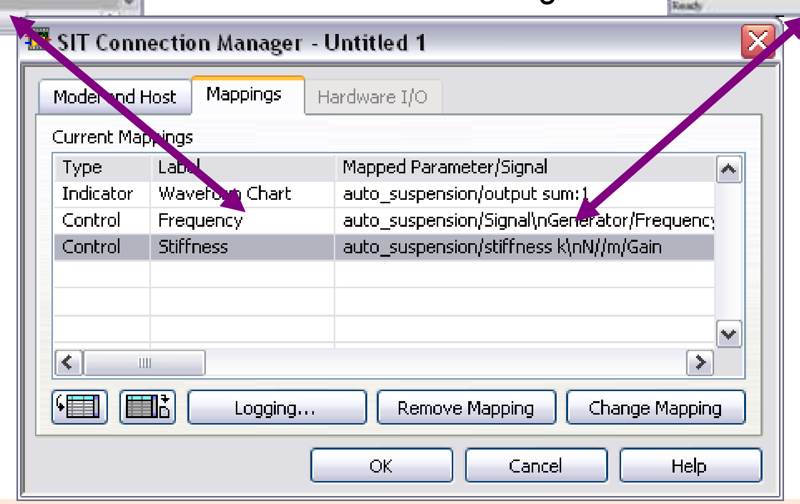
LabVIEW Front Panel



Simulation Model



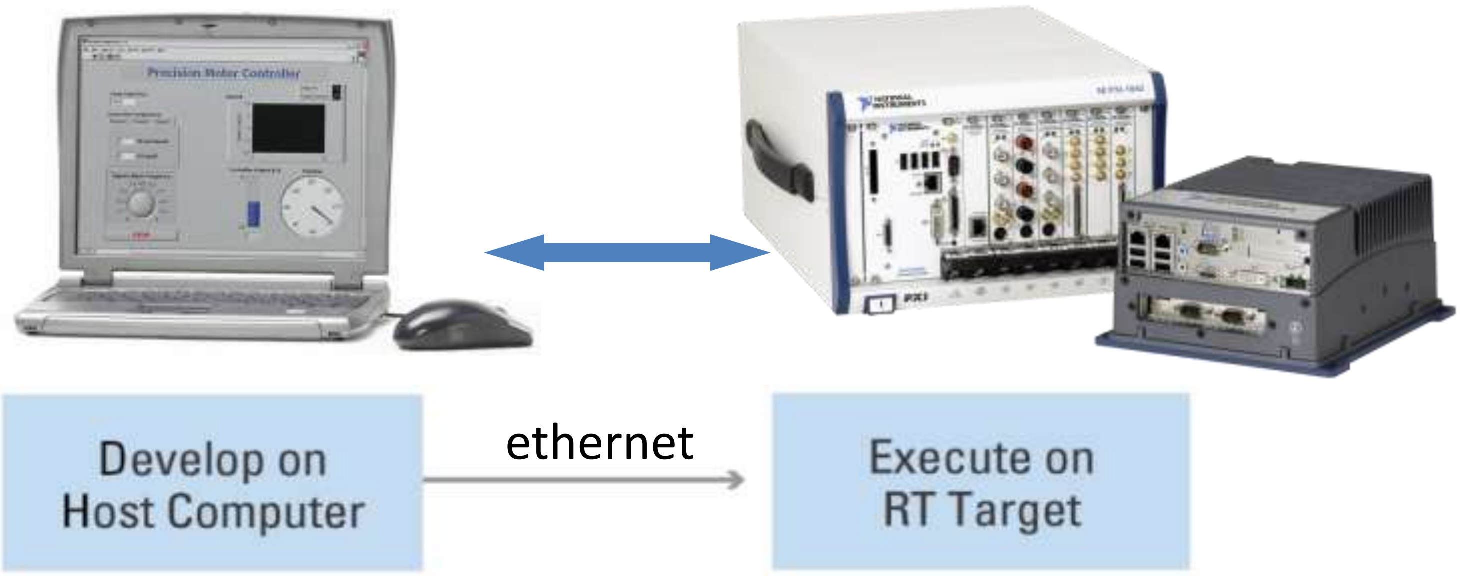
SIT Connection Manager



LabVIEW Controls and Indicators

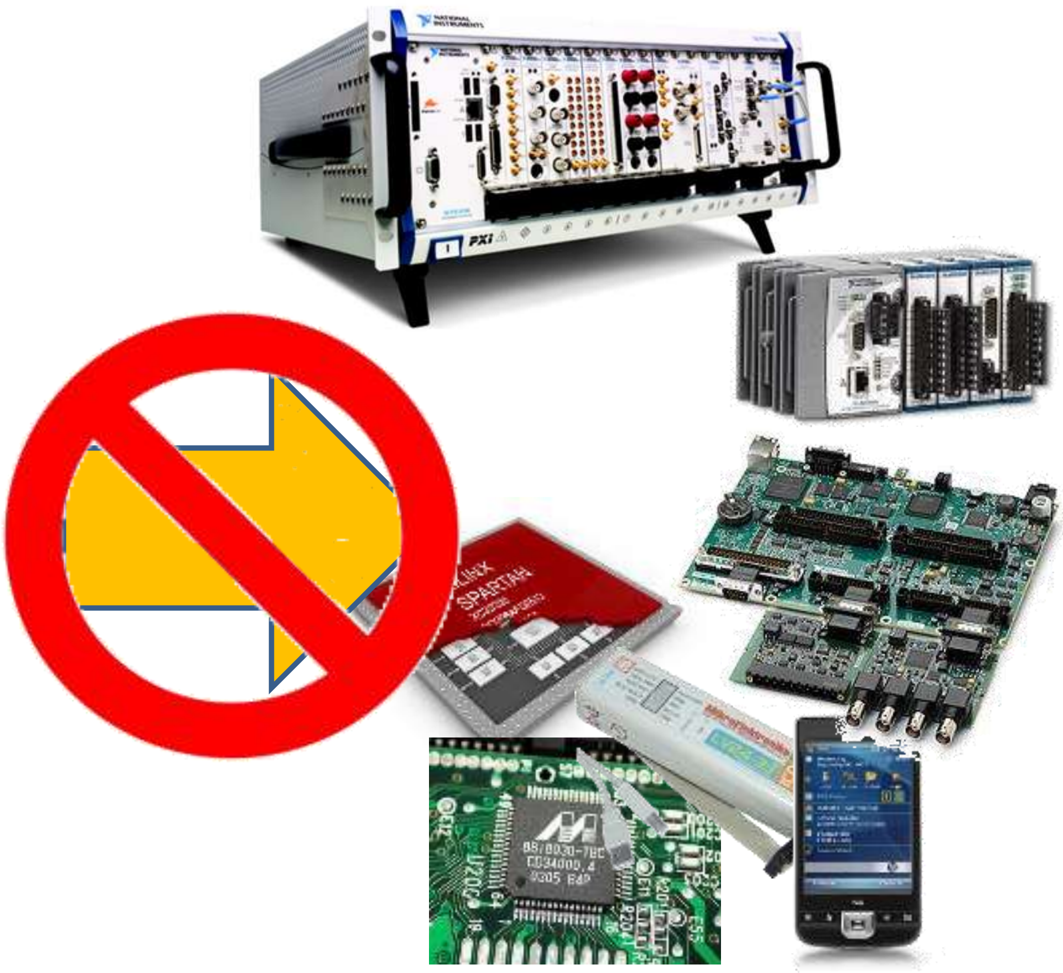
Model Parameters and Signals

LabVIEW Real-Time System Development

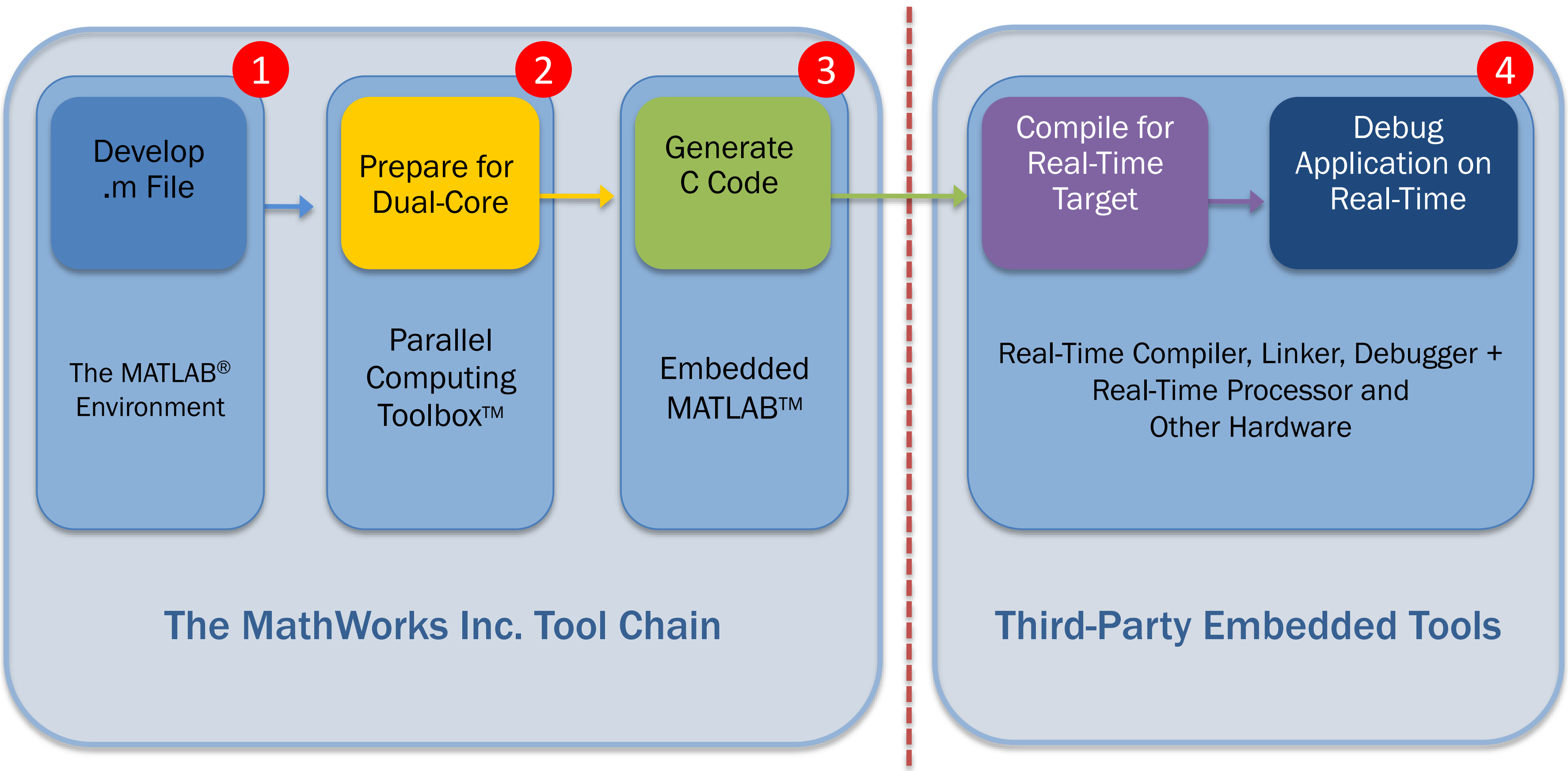


Why Is This Functionality Important?

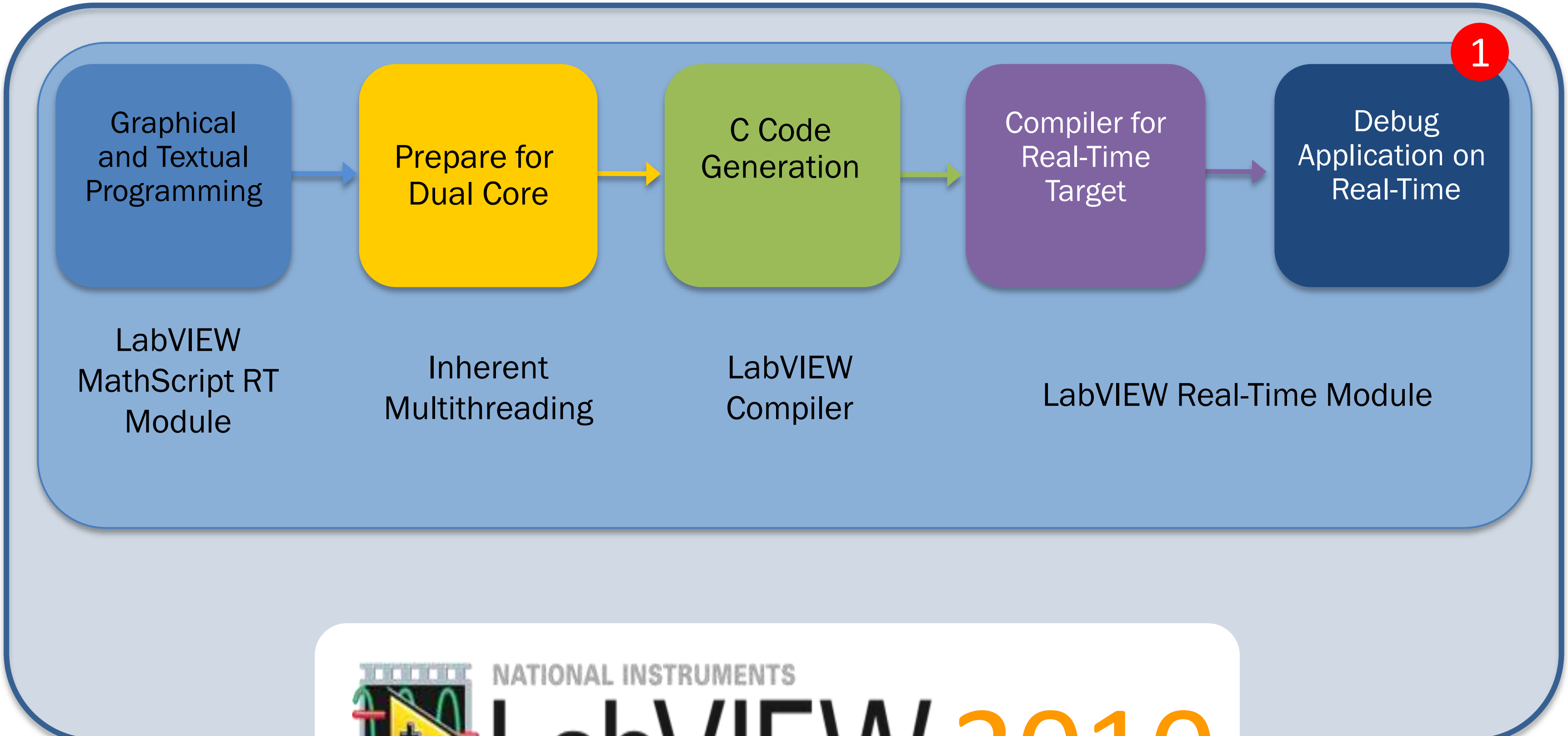
```
% Program P2_1
% Simulation of an M-point Moving Average Filter
% Generate the input signal
n = 0:100;
s1 = cos(2*pi*0.05*n); % A low-frequency sinusoid
s2 = cos(2*pi*0.47*n); % A high frequency sinusoid
x = s1+s2;
% Implementation of the moving average filter
M = input('Desired length of the filter = ');
num = ones(1,M);
y = filter(num,1,x)/M;
% Display the input and output signals
clf;
subplot(2,2,1);
plot(n, s1);
axis([0, 100, -2, 2]);
xlabel('Time index n'); ylabel('Amplitude');
title('Signal #1');
subplot(2,2,2);
plot(n, s2);
axis([0, 100, -2, 2]);
xlabel('Time index n'); ylabel('Amplitude');
title('Signal #2');
subplot(2,2,3);
plot(n, x);
axis([0, 100, -2, 2]);
xlabel('Time index n'); ylabel('Amplitude');
title('Input Signal');
subplot(2,2,4);
plot(n, y);
axis([0, 100, -2, 2]);
xlabel('Time index n'); ylabel('Amplitude');
title('Output Signal');
axis;
```



Deployment with Traditional Tools

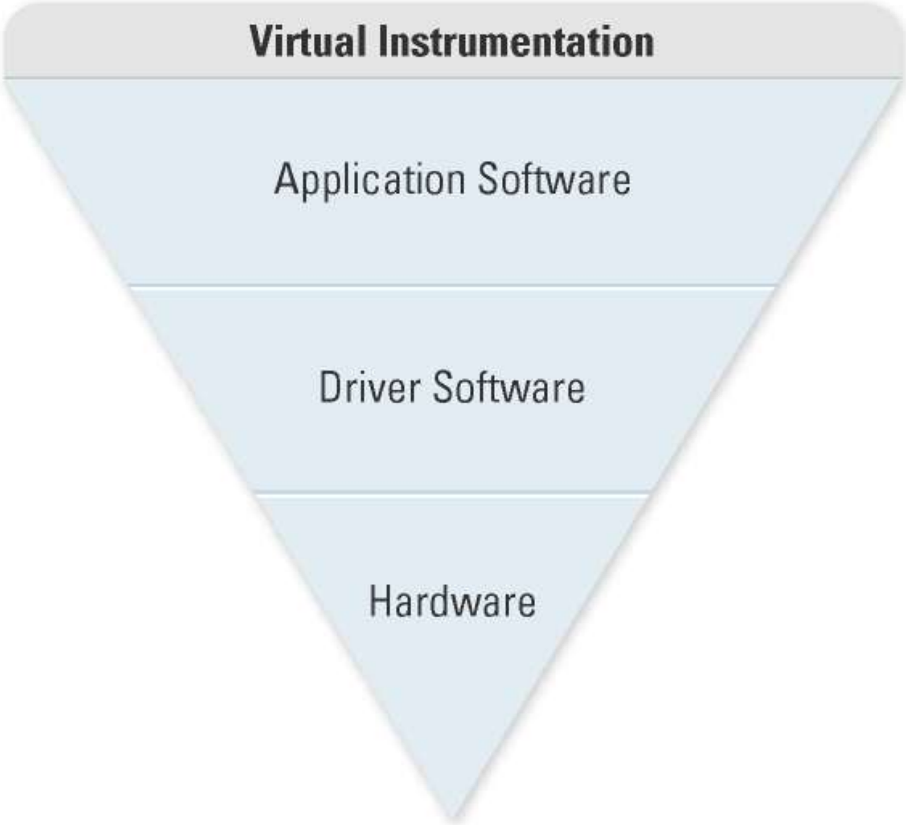


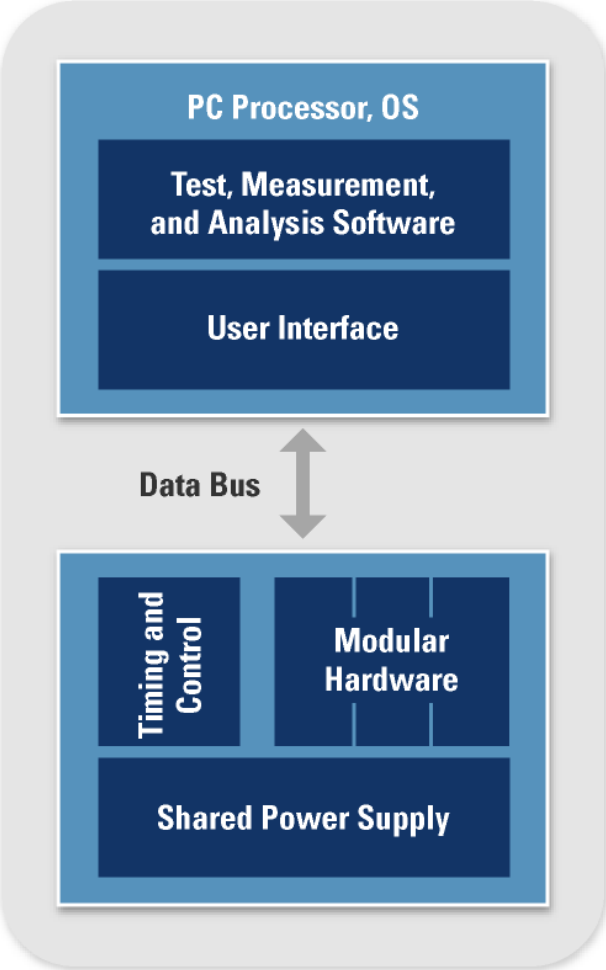
Deployment with LabVIEW



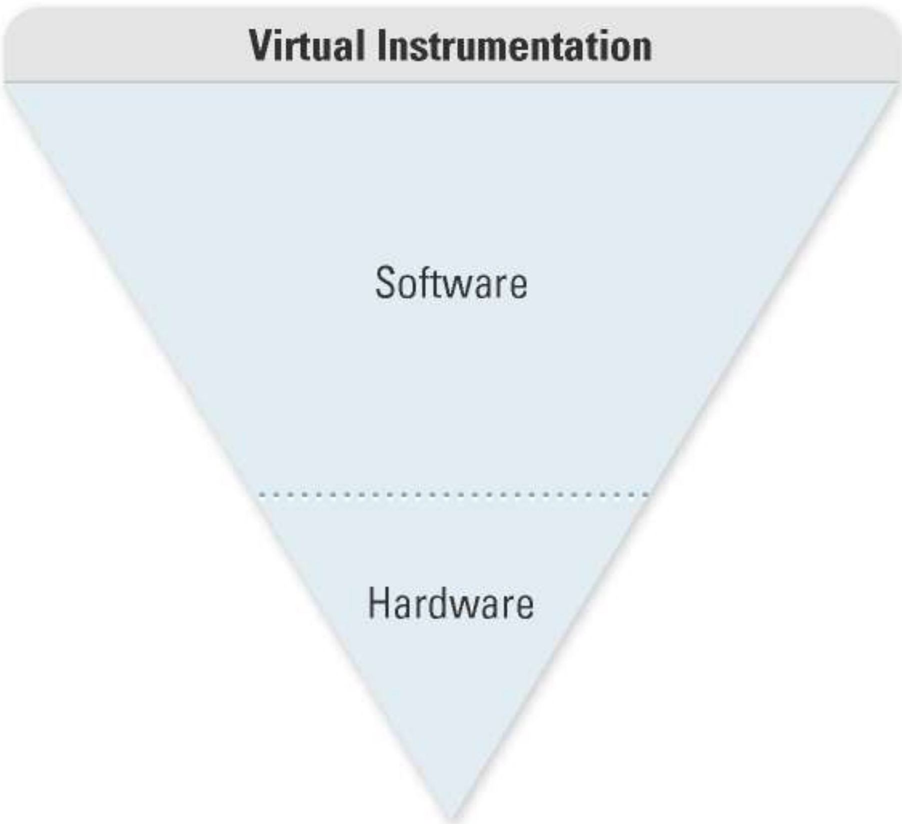
LABVIEW FPGA INTRODUCTION

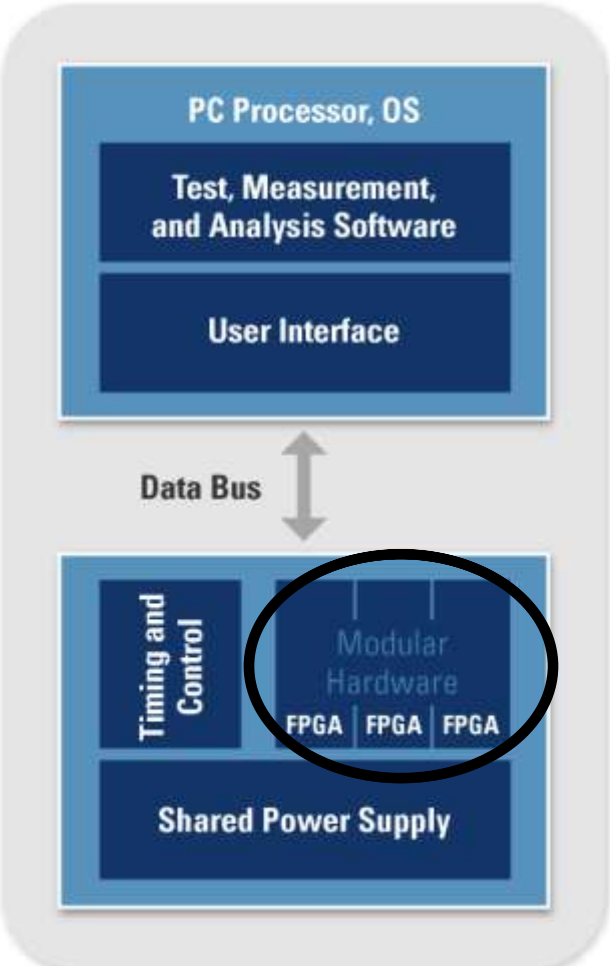
Customizing Instruments with DAQ and MI



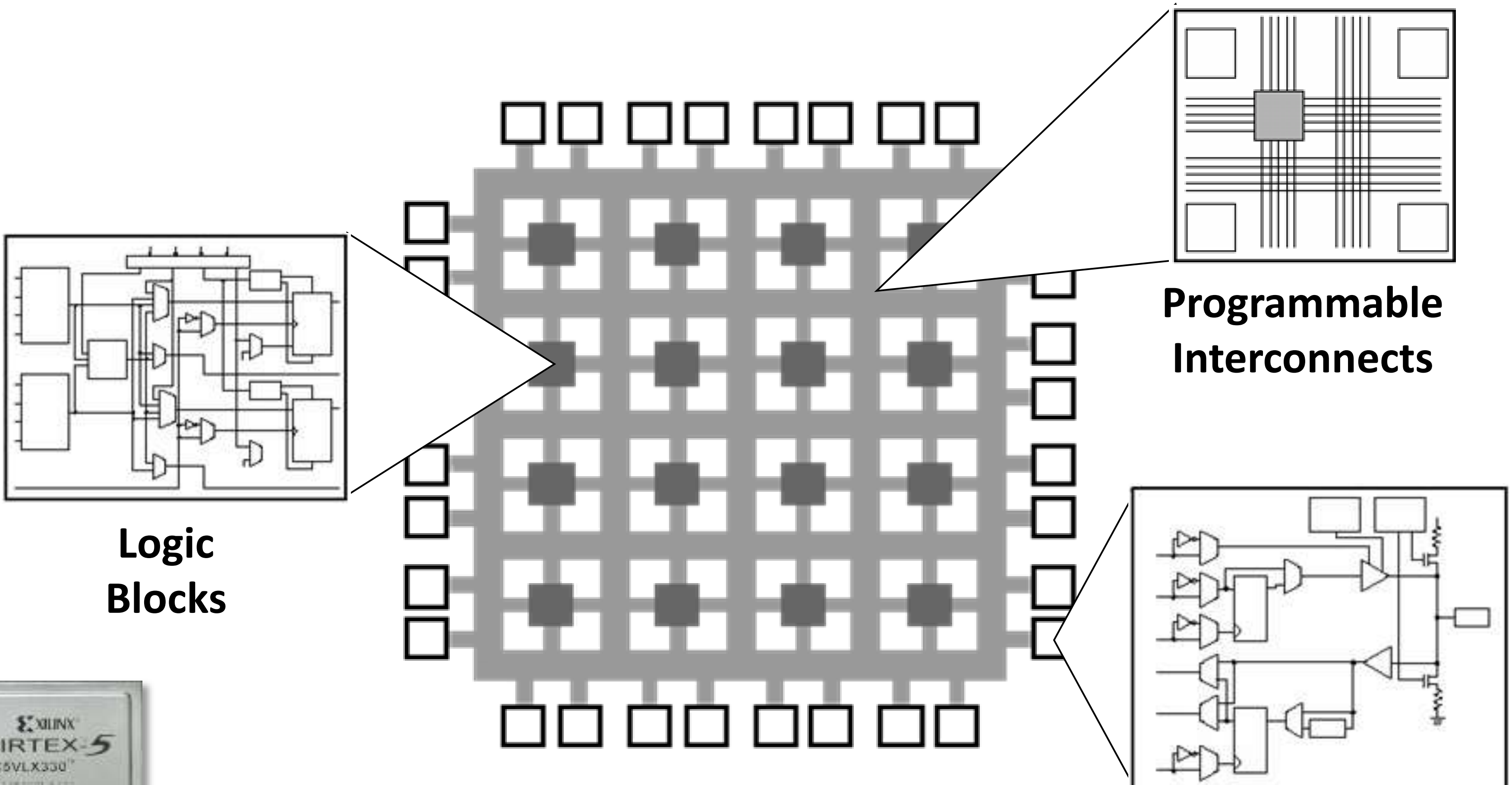


Customizing Instruments with FPGAs





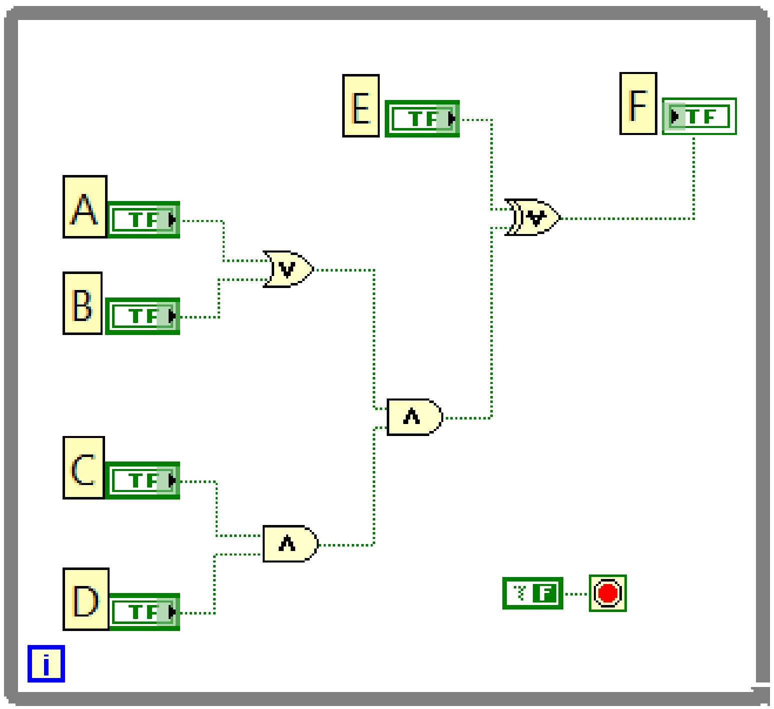
What is an FPGA?

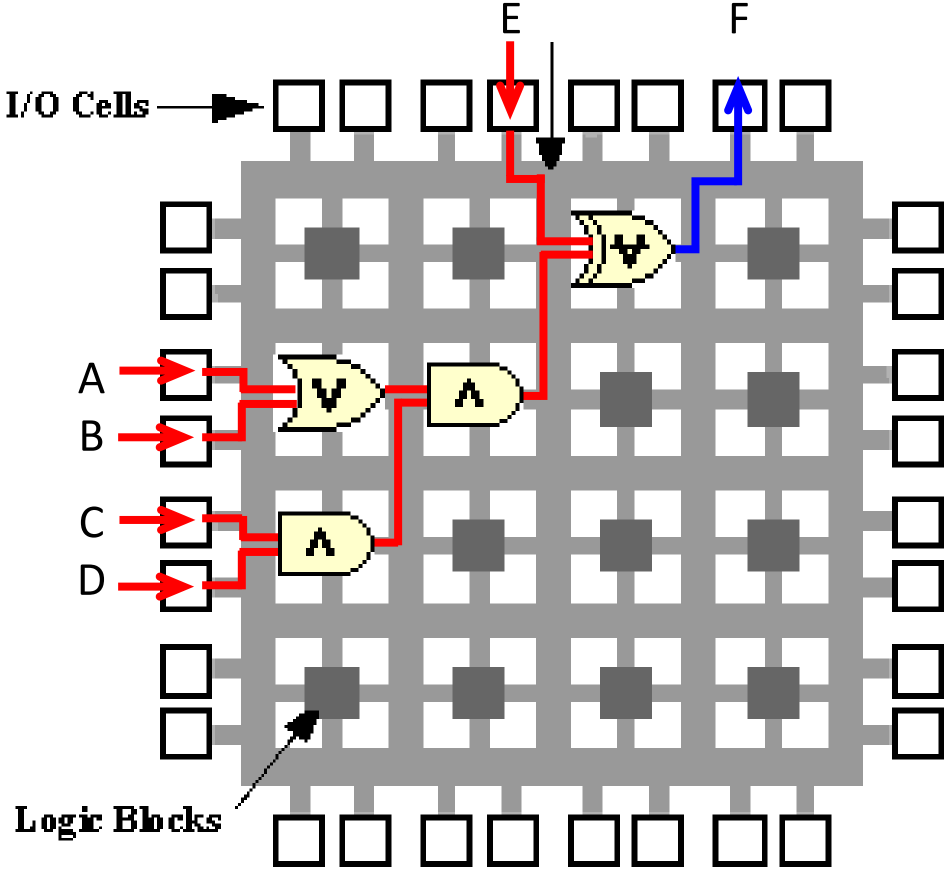


Field-Programmable Gate Array

Implementing Logic on FPGA: F = {(A+B)CD} ⊕ E

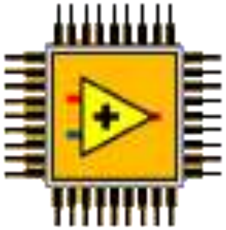
LabVIEW FPGA Code



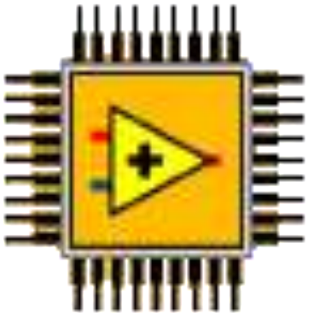


Why FPGAs?

- **High Reliability** – Designs become a custom circuit
- **High Determinism** – Runs algorithms at deterministic rates down to 25 ns (faster in many cases)
- **True Parallelism** – Enables parallel tasks and pipelining
- **Reconfigurable** – Create new and alter existing task-specific personalities

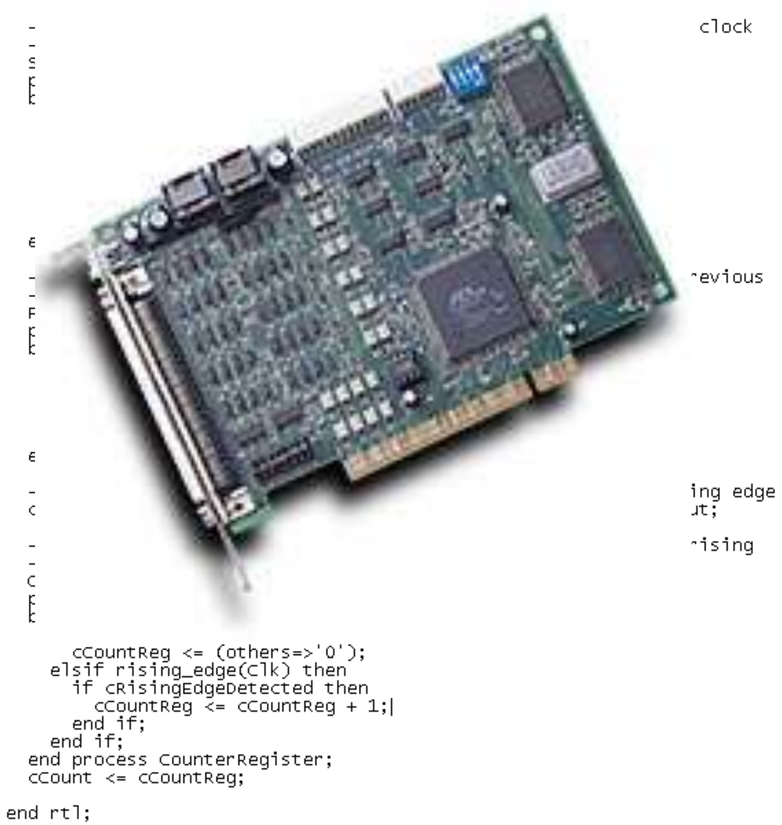


Common FPGA Applications in T & M



- When there is no instrument available
  - Custom digital communication protocols
  - Sensor simulation
  - Co-processing
- When the application requires point-by-point decisions
  - High-speed control
  - On-board processing and data reduction
  - Intelligent DAQ

Traditional Approach to Custom Hardware



Hardware Design: 2 weeks
Prototype A Build/Test: 2 weeks
Prototype B Build/Test: 2 weeks
Software Design/Coding: 2 weeks
Software Testing: 2 weeks
System Testing/Certification: 2 weeks

TOTAL: 12 weeks, €$ €$ €$

Programming FPGA

Counter

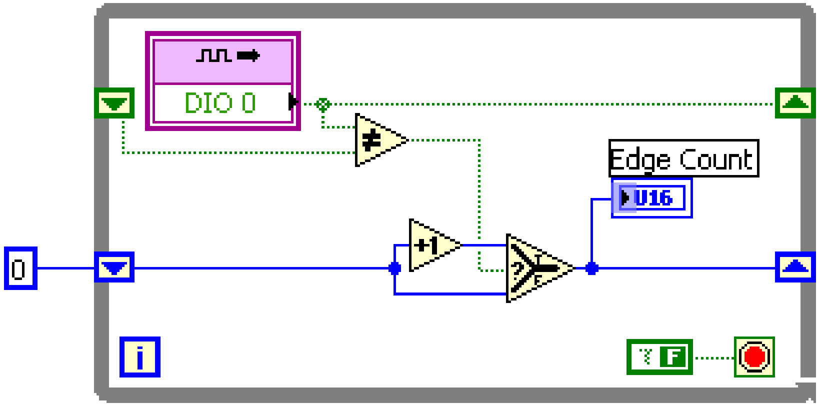
```
-- First we sychronize the asynchronous digital input to our clock
-- by inserting two flip flops.
SynchronizationFFs:
process( aReset, Clk )
begin
  if aReset then
    cDigitalInput_ms <= false;
    cDigitalInput <= false;
  elsif rising_edge(Clk) then
    cDigitalInput_ms <= aDigitalInput;
    cDigitalInput <= cDigitalInput_ms;
  end if;
end process SynchronizationFFs;

-- Then we keep track of what the digital input was on the previous
-- clock cycle by inserting another flip flop
PreviousDigitalInputFF:
process( aReset, Clk )
begin
  if aReset then
    cPrevDigitalInput <= false;
  elsif rising_edge(Clk) then
    cPrevDigitalInput <= cDigitalInput;
  end if;
end process PreviousDigitalInputFF;

-- Then we have a little combinatorial logic to detect a rising edge
cRisingEdgeDetected <= cDigitalInput and not cPrevDigitalInput;

-- And finally we have a register that increments when that rising
-- edge is detected.
CounterRegister:
process( aReset, Clk )
begin
  if aReset then
    cCountReg <= (others=>'0');
  elsif rising_edge(Clk) then
    if cRisingEdgeDetected then
      cCountReg <= cCountReg + 1;
    end if;
  end if;
end process CounterRegister;
cCount <= cCountReg;

end rtl;
```

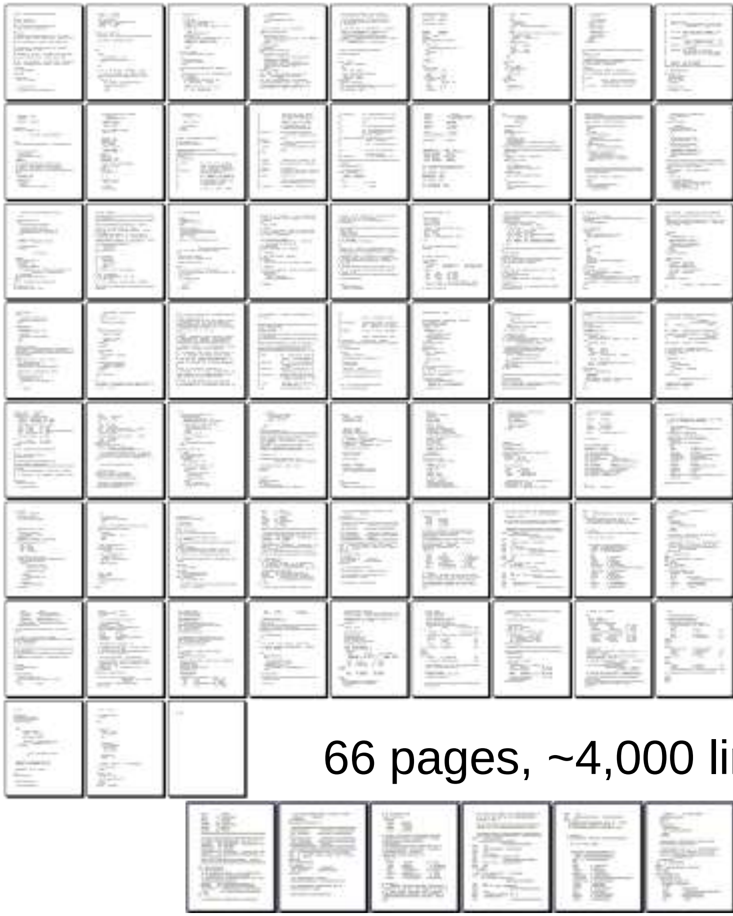


VHDL

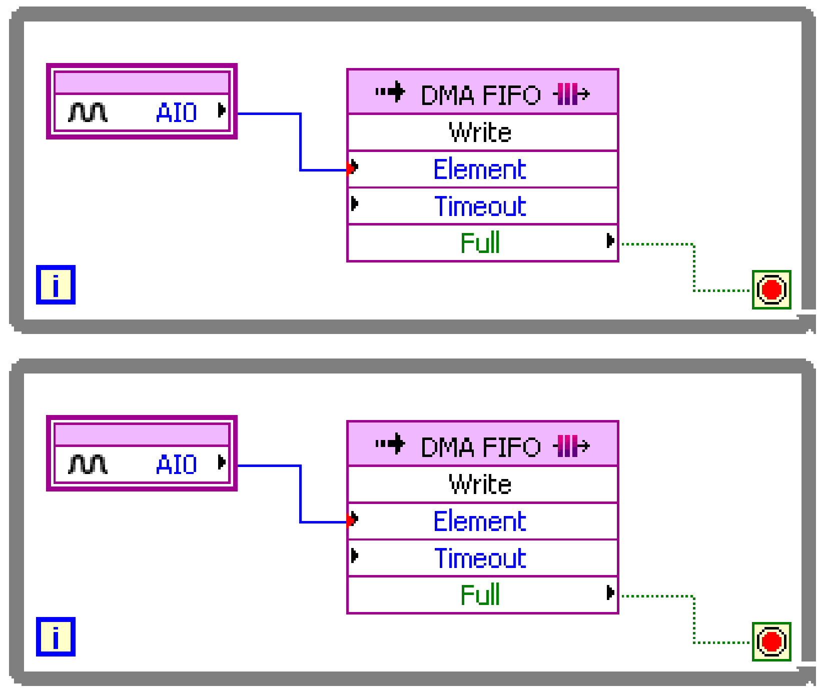
LabVIEW FPGA

Programming FPGA

I/O with DMA

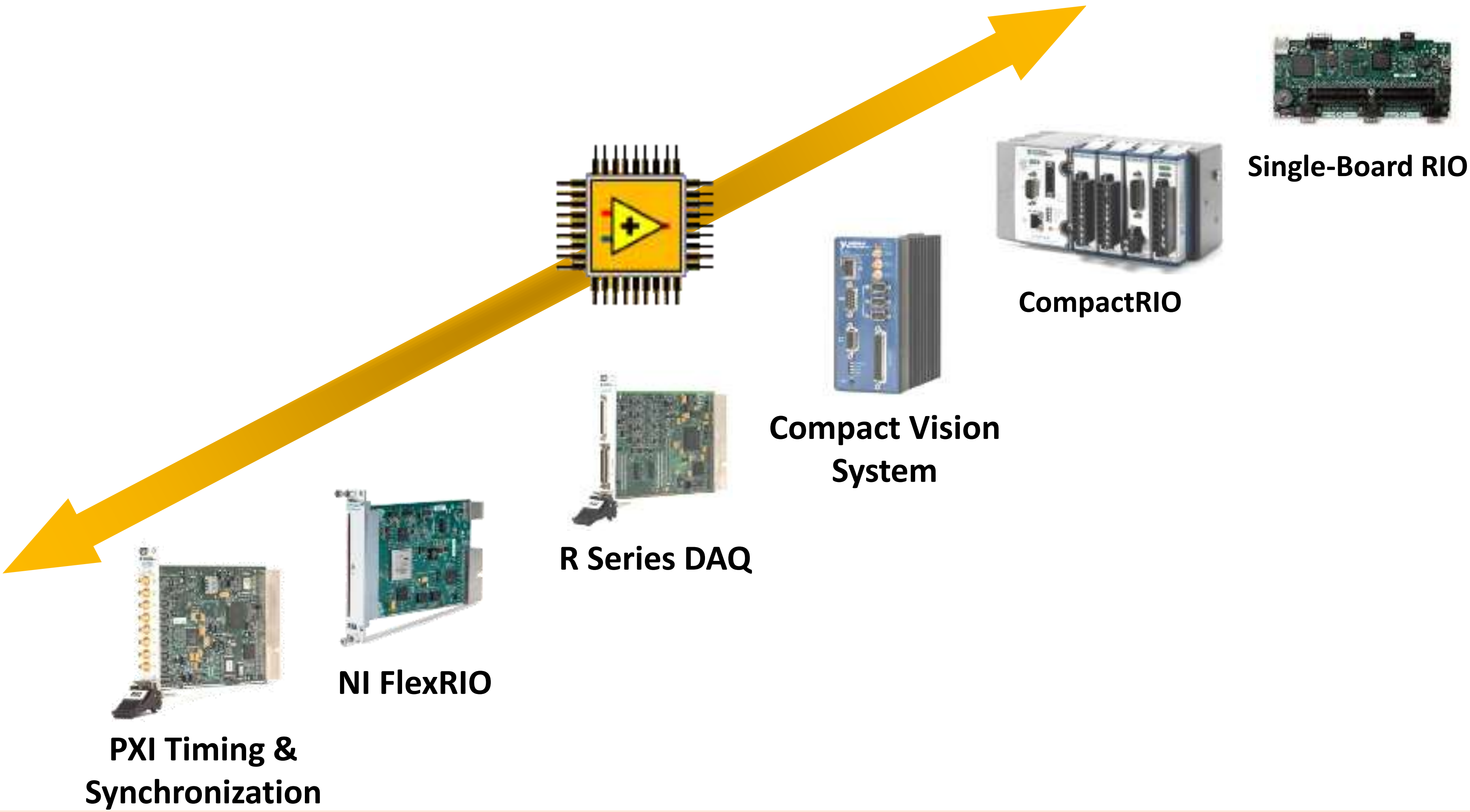


66 pages, ~4,000 lines



LabVIEW FPGA

LabVIEW FPGA HW Targets

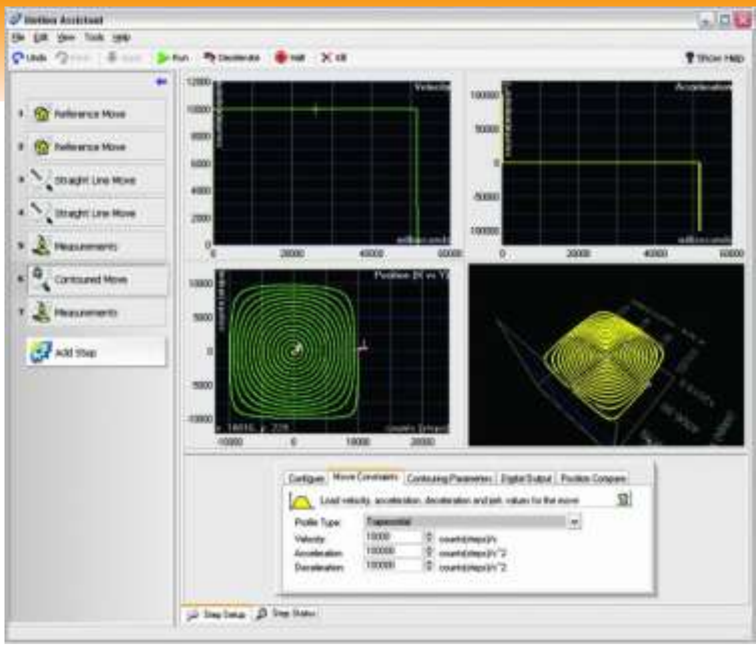


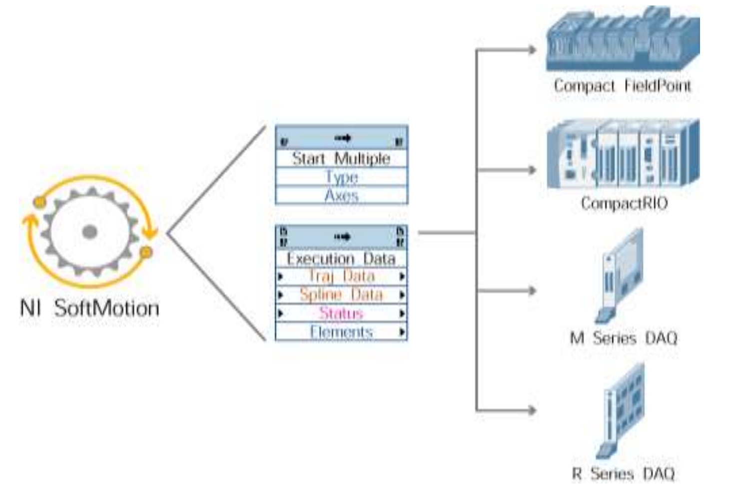
Motion Control with LabVIEW

- **NI Motion Assistant**
  - Interactive environment with 3D visualization
  - Ready-to-run LabVIEW or C code creation
  - Easy trapezoidal or S-curve velocity profile implementation
  - Teach pendant for easy prototyping

- **NI SoftMotion Controller for CANopen and IEEE 1394**
  - Use LabVIEW and NI Motion Assistant to program distributed motion control applications
  - Compatible with intelligent CANopen drives from Copley and IEEE 1394 drives from ORMEC

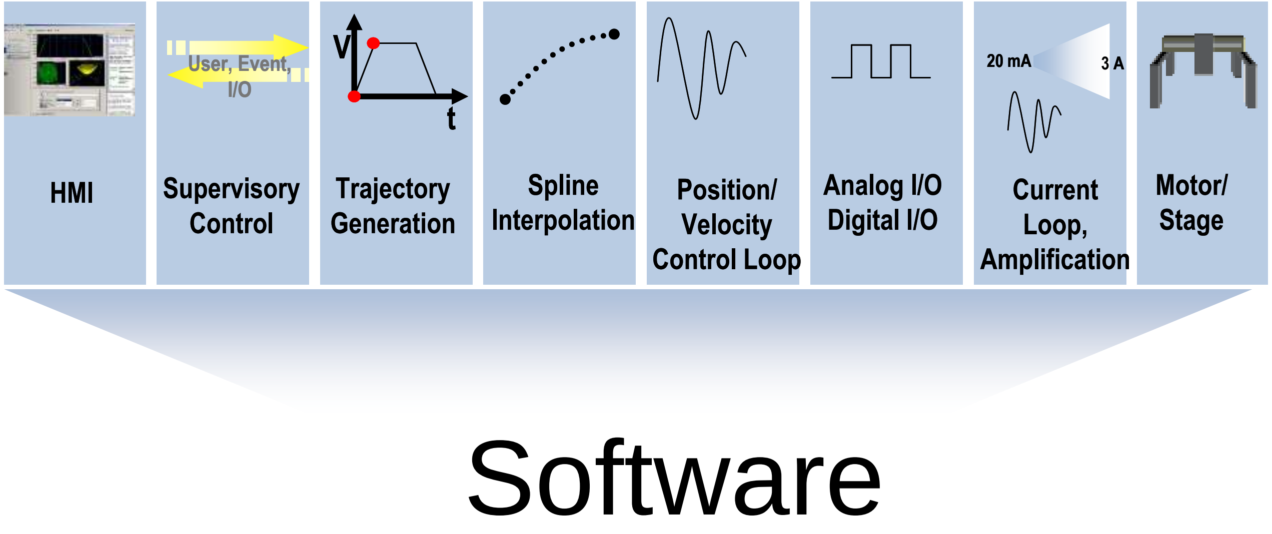
- **NI SoftMotion Development Module**
  - Develop custom motion controllers in LabVIEW Real-Time or LabVIEW FPGA
  - Use trajectory generation, spline interpolation, position, velocity control, and encoder implementation VIs



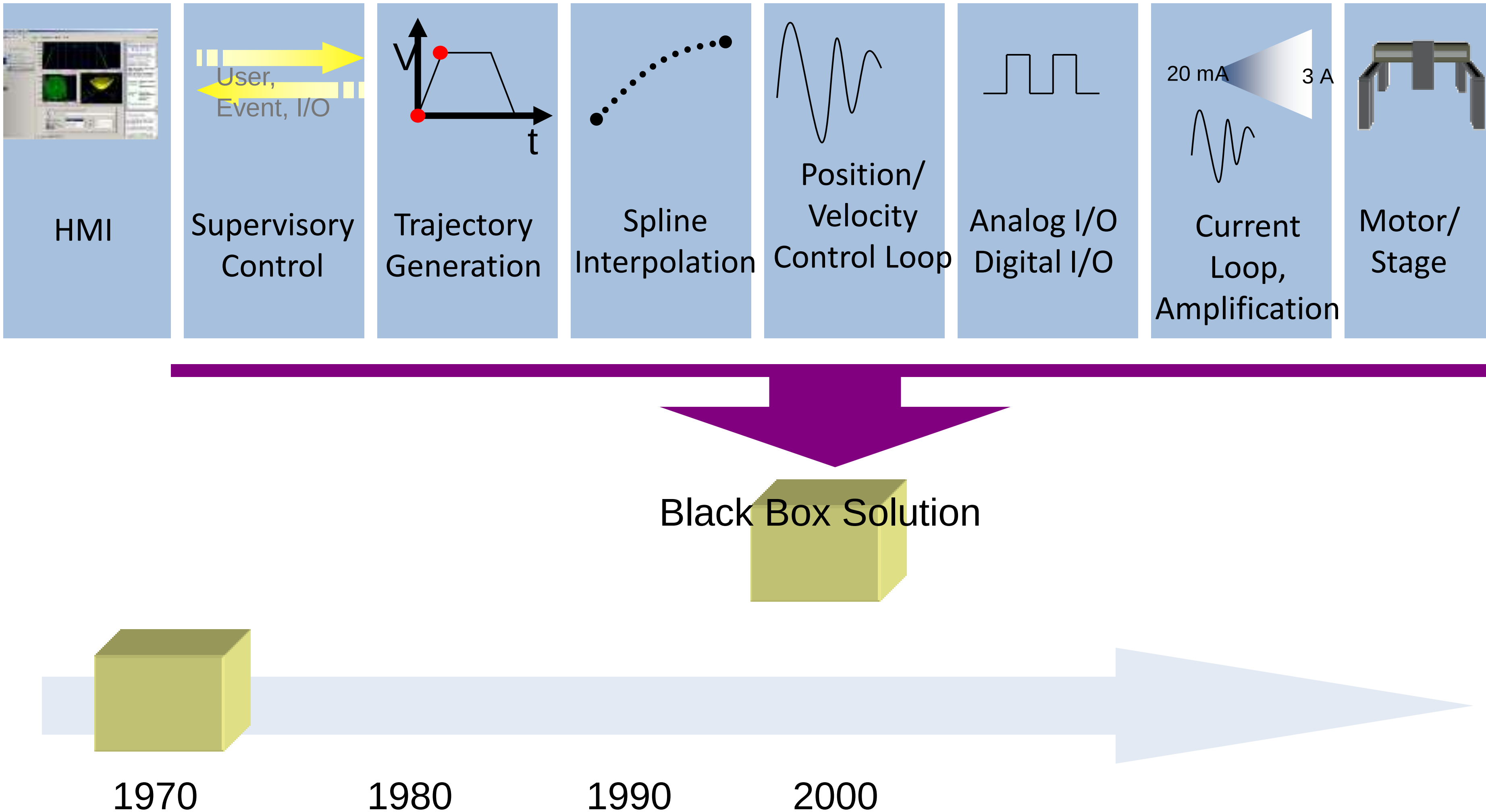


What is Soft Motion?

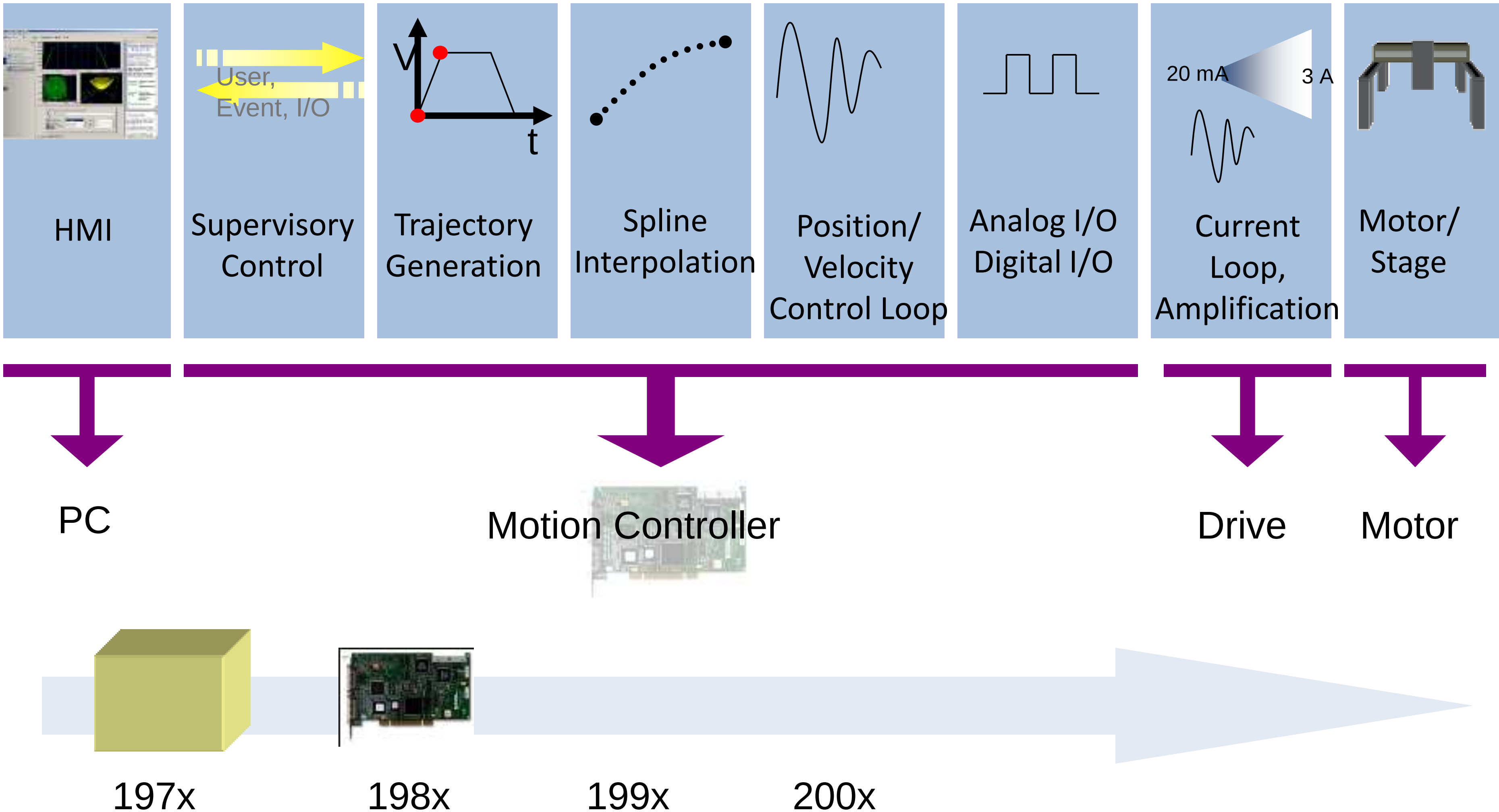
- Implement parts of a motion control system in software
- Disaggregate a motion control system into modular components that are open, flexible, and customizable



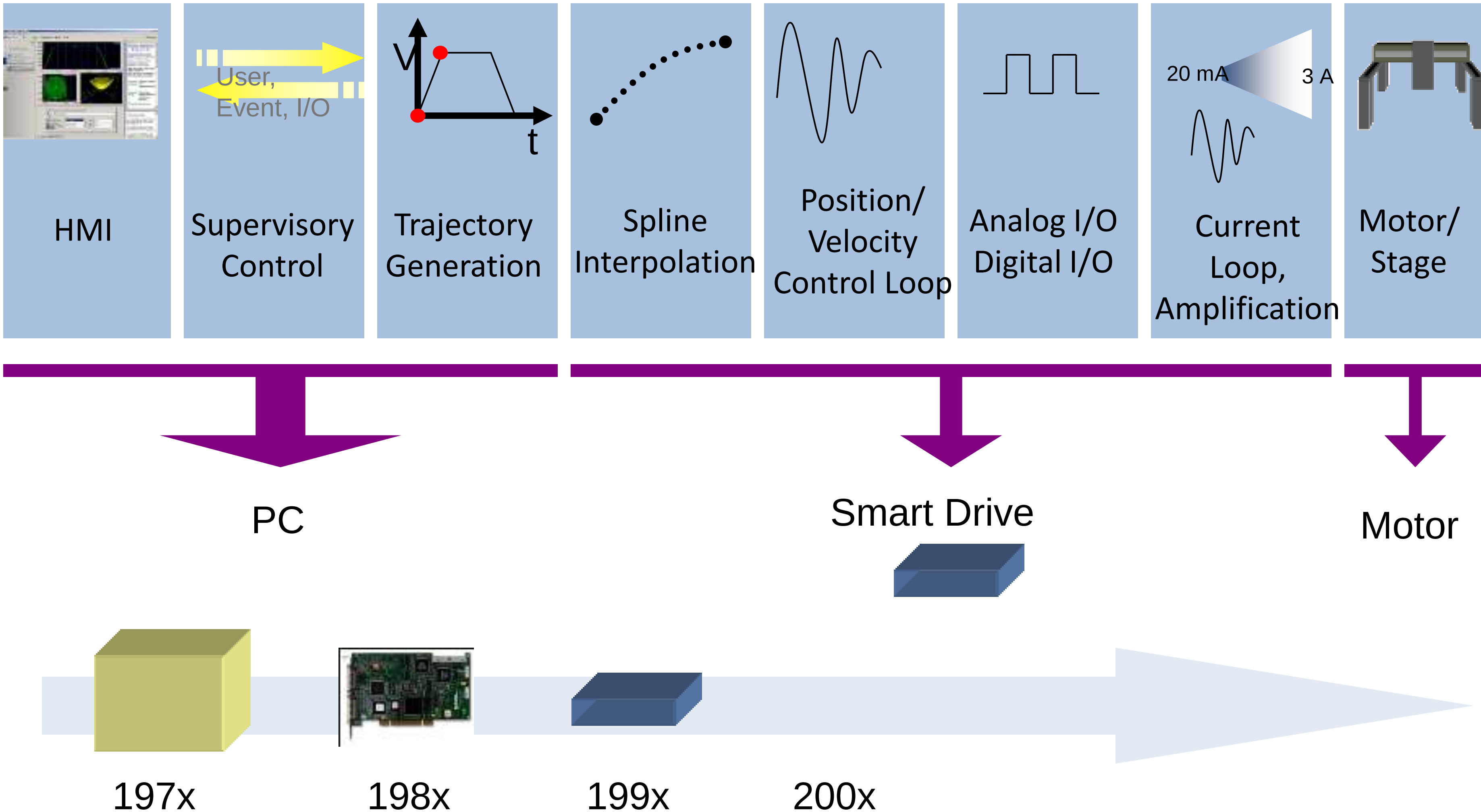
1970s: Black Box Solution



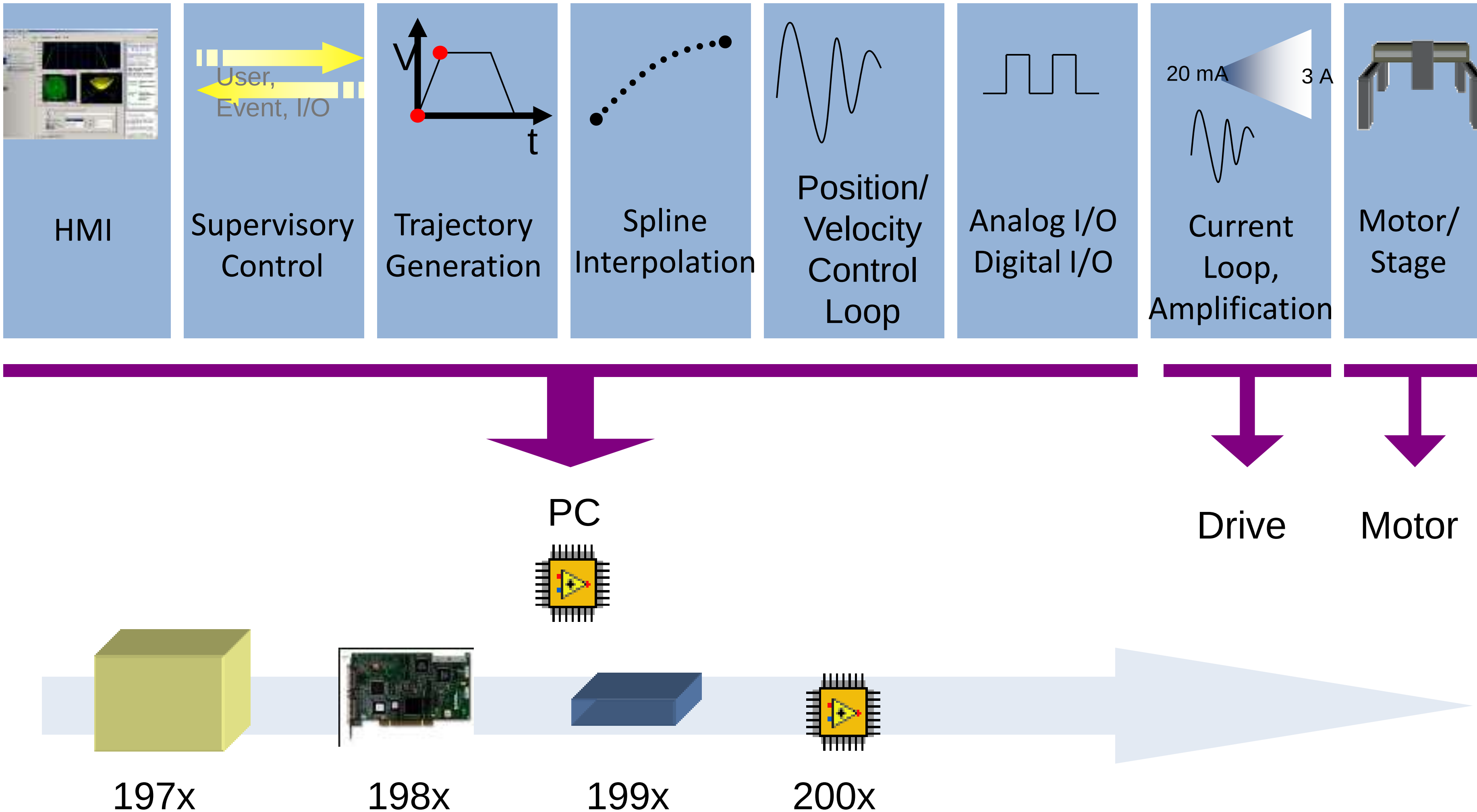
1980s: Emergence of Bus Technology



1990s: Emergence of Smart Drives



2000s: Emergence of RIO Technology



Motion Controller Architecture

Traditional Plug-in Motion Controllers

NI Motion Assistant

NI-Motion for LabVIEW

DSP on Motion Controller

Supervisory Control	Trajectory Generation	Spline Interpolation	Position, Velocity Control Loop

NI 73xx Motion Controllers

NI-SoftMotion Development Module for LabVIEW

Traditional Plug-in Motion Controllers

NI Motion Assistant

NI-Motion for LabVIEW

DSP on Motion Controller

Supervisory Control	Trajectory Generation	Spline Interpolation	Position, Velocity Control Loop

NI 73xx Motion Controllers

NI SoftMotion Technology

NI SoftMotion Development Module for LabVIEW

Supervisory Control	Trajectory Generation	Spline Interpolation	Position, Velocity Control Loop

LabVIEW Real-Time and / or LabVIEW FPGA

Single Board Computers

Compact FieldPoint

M Series DAQ

R Series DAQ

CompactRIO

Choosing the Right Platform

Hardware Platform	Trajectory Generator	Spline Interpolation	Position/Velocity Control Loop	Encoder Implementation	Control Loop Rate	Applications
CompactRIO 	LabVIEW Real-Time	LabVIEW-FPGA			5 us	Ultra high precision machines for nanotech and MEMS applications
PC/PXI with R-Series 	LabVIEW Real-Time	LabVIEW-FPGA			5 us	Ultra high precision machines in nanotech and MEMS applications,
PC/PXI with Plug-in Motion Controller 	DSP on Motion Controller				62.5 us	High precision machines in semiconductor, healthcare applications
PC/PXI with Plug-in Data Acquisition 	LabVIEW Real-Time			-	1 ms	Packaging, material handling applications
Compact FieldPoint 	LabVIEW Real-Time			-	10 ms	Servo hydraulics, conveyors, high inertia systems

LABVIEW ROBOTICS INTRODUCTION

DARPA Urban Challenge

- Autonomously navigate 60 mile urban environment
- Follow all traffic laws
- Safely navigate around obstacles/barriers
- Safely interact with other UGV and manned vehicles at stop signs during passing, parking

Odin – 3rd Place Winner
Virginia Tech
Powered by LabVIEW and CompactRIO

Comparison

LabVIEW / RIO for Robotics

Virginia Tech (LabVIEW + CompactRIO)

Do-it-yourself

MIT (C with a 40 core Linux cluster)

Comparison

- NI dramatically reduces the cost of robotics system design, deployment, and testing
- Key strengths:
  - Rapid development software, Extensive, integrated I/O support, Tightly integrated system design platforms, World class support

LabVIEW Robotics

Connectivity to sensors and actuators from top vendors

Image processing and acquisition libraries

JAUS and Ethernet protocols for communication

Deployment to embedded real-time and FPGA hardware

A* and AD* search algorithms for real-time obstacle avoidance

Steering and kinematics algorithms

"What is" LabVIEW Robotics

New Getting Started Experience

Getting Started Wizard, RIO Hardware Wizard, Template Architectures

Robotics Software

LabVIEW FPGA
LabVIEW Real-Time
LabVIEW Mathscript RT
LabVIEW CD&Sim
NI Vision
NI Soft Motion
LabVIEW Statechart
PID Toolkit
System ID Toolkit

Robotics Drivers

Sensor Drivers
Actuator Drivers
Driver Project Wizard

Robotics IP

Search algorithms
Robotics visualization
Obstacle avoidance
Kinematics
Robotics Examples

LabVIEW Software Bundle

- LabVIEW Real-Time
- NI SoftMotion
- LabVIEW FPGA
- LabVIEW Statechart
- NI Vision
- LabVIEW Mathscript RT
- LabVIEW Control Design and Simulation
- LabVIEW PID Toolkit

3rd Party Connectivity

Development framework

Collaborator	New capabilities
Cogmation	System simulation
Energid	Kinematics
Hokuyo	LIDAR sensors
iRobot	Hardware integration
Microsoft	System simulation
MobileRobots	Hardware integration
MaxonMotors	Smart motor connectivity
Pitsco	Starter kit OEM, Academic
Skilligent	AI vision software
TORC	JAUS Protocol
Velodyne	LIDAR sensors

Open software

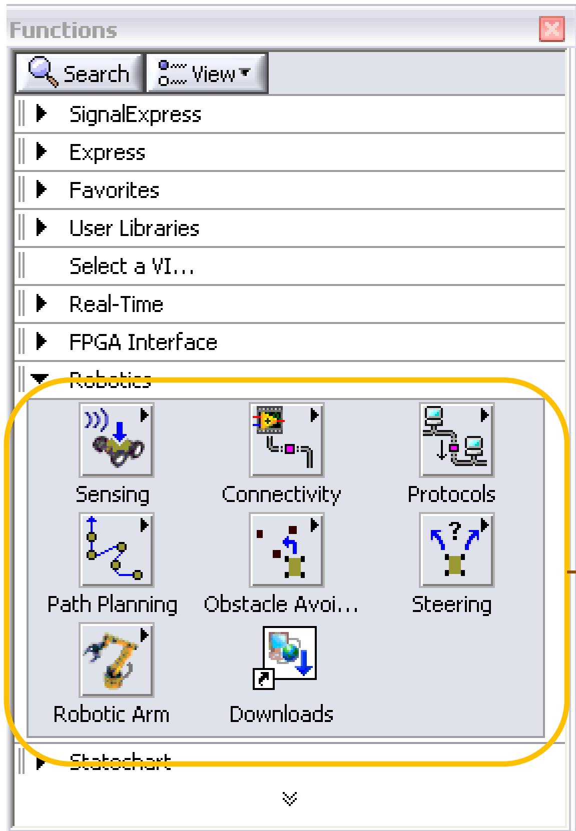
Multiple hardware targets

Code reuse

Advanced Control

Integrate C, HDL, .m files

Robotics IP

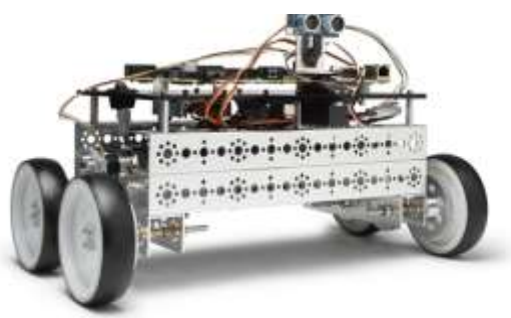


- Sensing
- Connectivity
  MobileRobots, Skilligent, Cogmation
- Protocols
  JAUS, NMEA, FPGA
- Path Planning
- Obstacle Avoidance
- Steering
- Robotic Arm
  Open Source Robotics Toolbox
- Download additional algorithms

Targets for LV Robotics

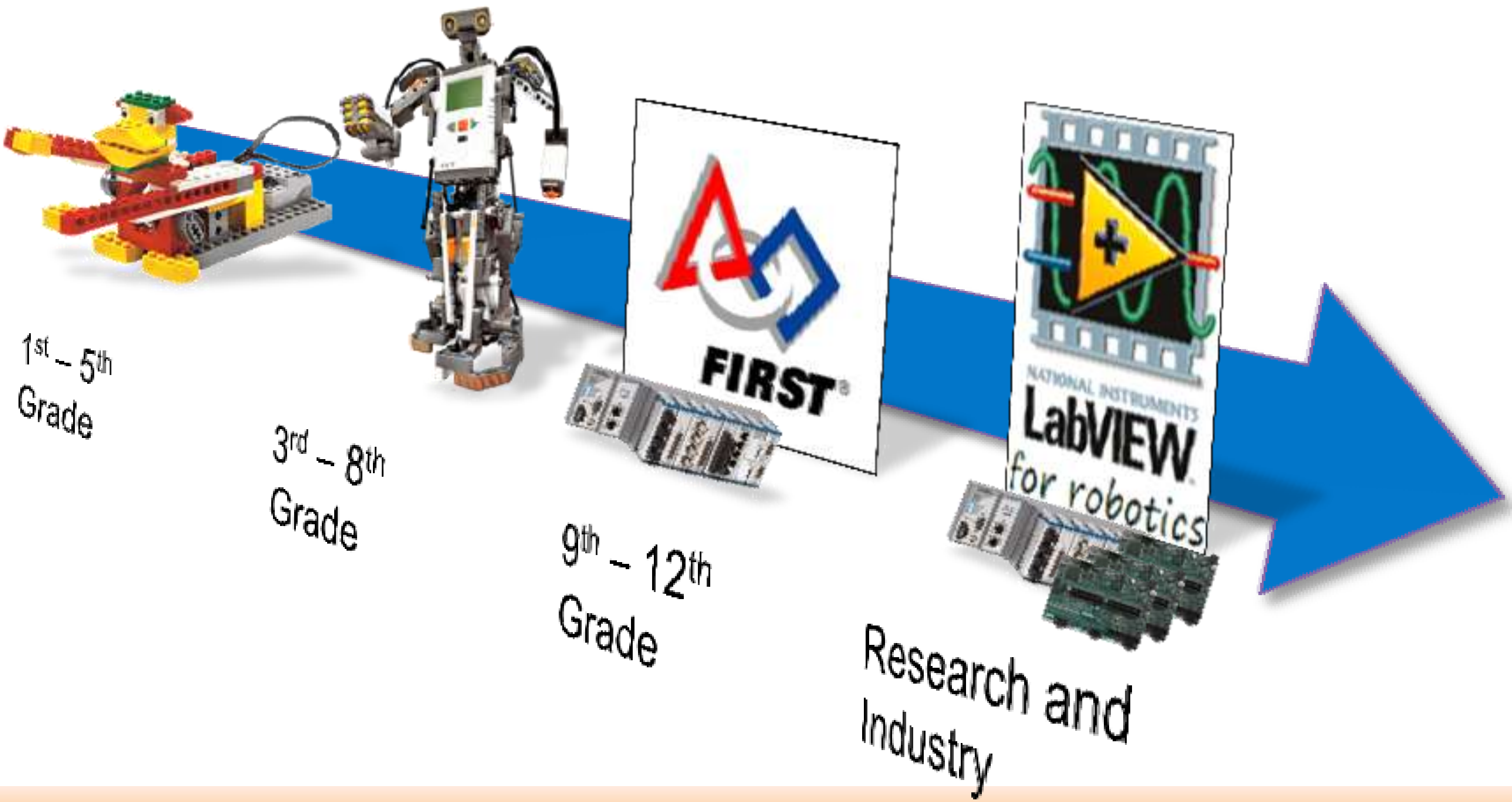
NI Single-Board RIO
- Board-level embedded controller
- Integrated FPGA and real-time processor
- Built-in analog and digital I/O, can be expanded with I/O modules

NI CompactRIO
- Embedded controller with rugged, mechanical enclosure
- Integrated FPGA and real-time processor
- Includes modules for robotics-specific I/O



NI Robot Starter Kit
- Robot kit based on NI Single-Board RIO
- Includes ultrasonic sensor, motors, encoders, battery, and charger
- Performs basic obstacle avoidance out of the box

Extending Graphical System Design for Robotics

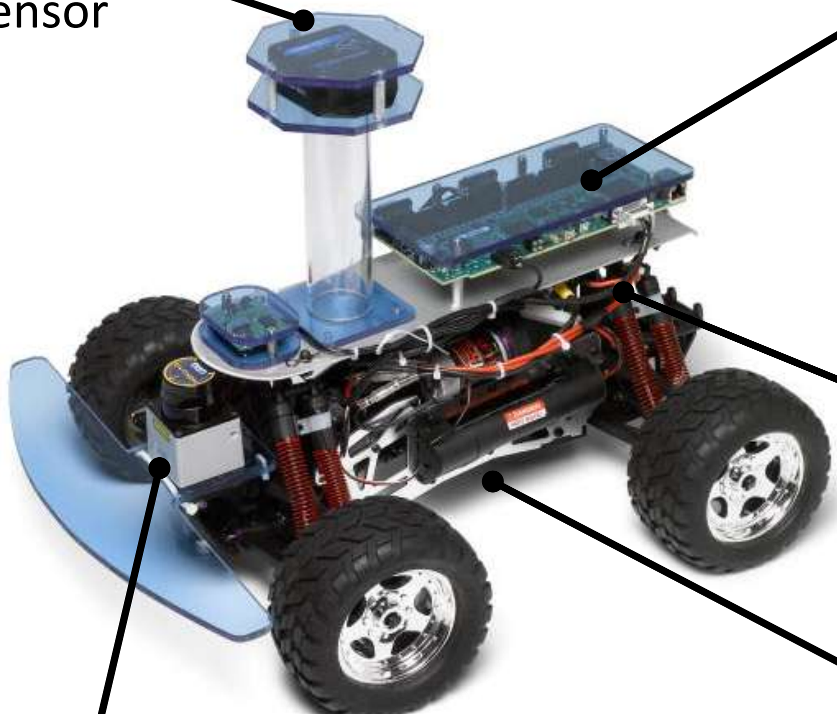


Autonomous Ground Vehicle example

Microstrain®
Inertial Sensor
(inactive)

NI Single-Board RIO

- RT Power PC Processor
- FPGA
- Serial connectivity to sensors
- Ethernet communication to development laptop

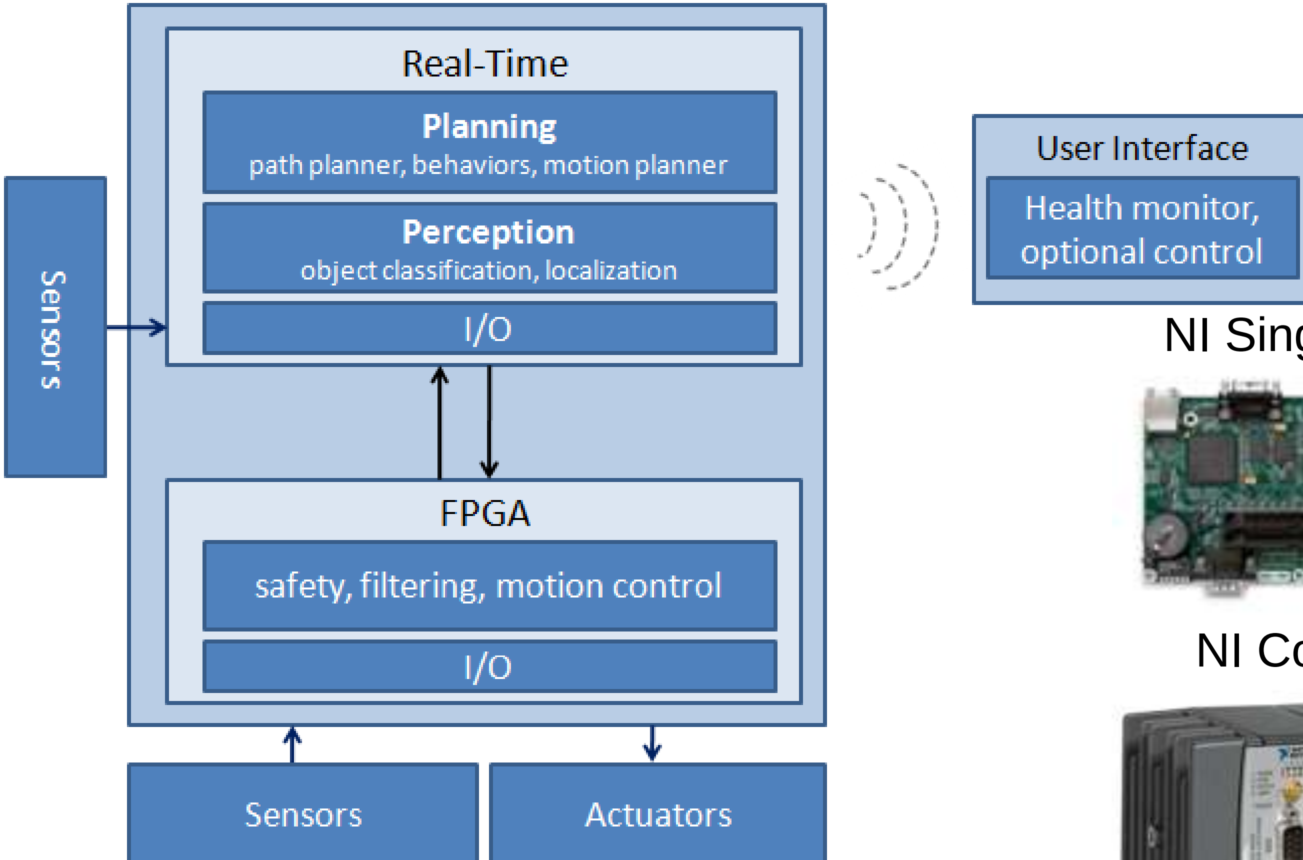


IFI Robotics®Victor 884
Speed Controller

HPI Racing® E-Savage
Sport Hobbyist RC Car

Hokuyo® LIDAR Sensor

LabVIEW Robotics on NI RIO Hardware



NI Single-Board RIO

NI CompactRIO

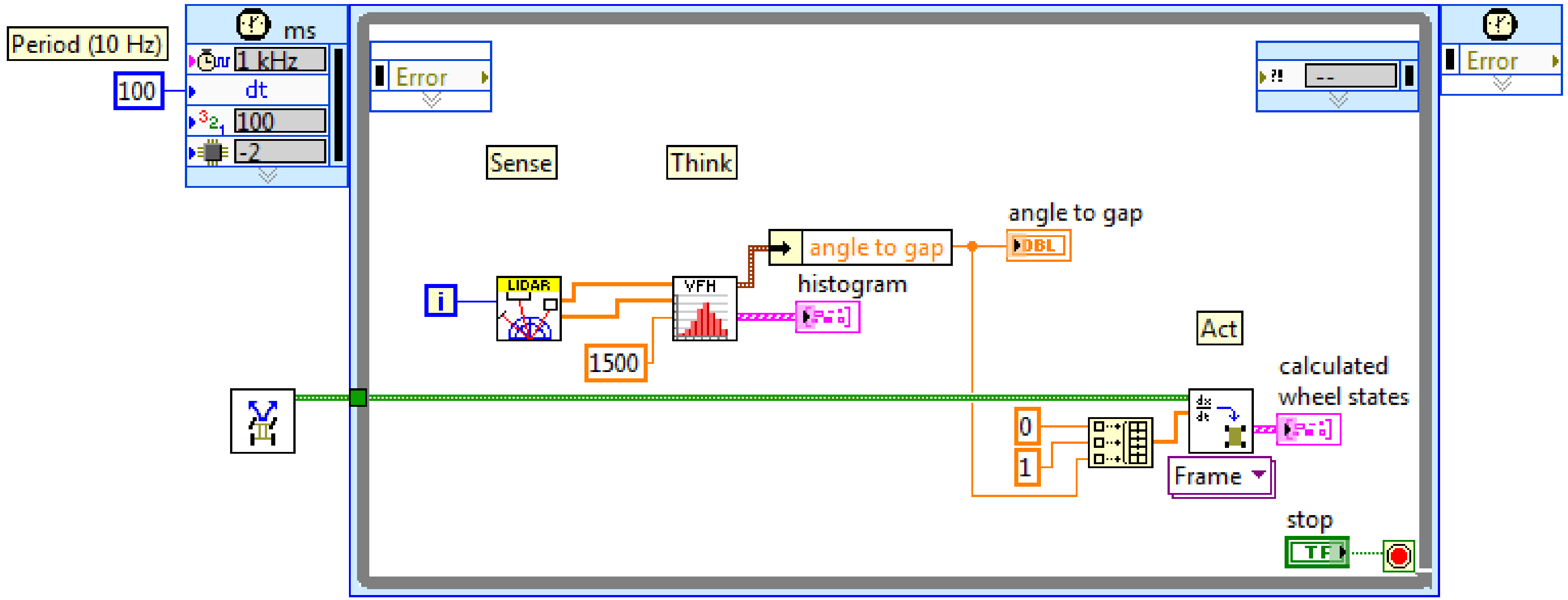
Overview of a Simple Mobile Robot

Sense
- Data acquisition
- Filtering
- Perception

Think
- Navigation
- Localization
- Decision making

Act
- Locomotion
- Kinematics
- Motor control

Code example



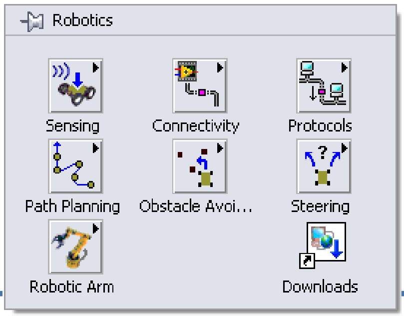
-> Scan area with LIDAR
-> calculate trajectory around objects
-> control motors

NI LabVIEW Robotics Starter Kit for Teaching and Research

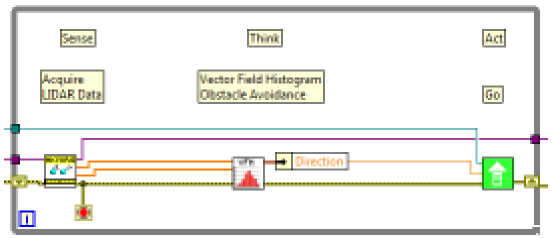
- Complete, out-of-the-box mobile robot platform
- For teaching robotics and prototyping
- Includes NI Single-Board RIO, motors, motor encoders, rotating ultrasonic sensor
- Can be expanded with additional sensors, actuators, and manipulators

LabVIEW Robotics

IP for navigation, steering, kinematics and more



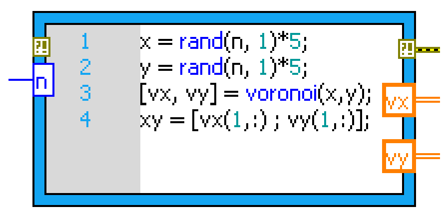
High-level **graphical programming** environment



Deployment to Real-Time and FPGA hardware

Connectivity to sensors and actuators from top vendors

Tools for integrating **text-based algorithms**



Examples of real-world applications

ni.com/robotics/education