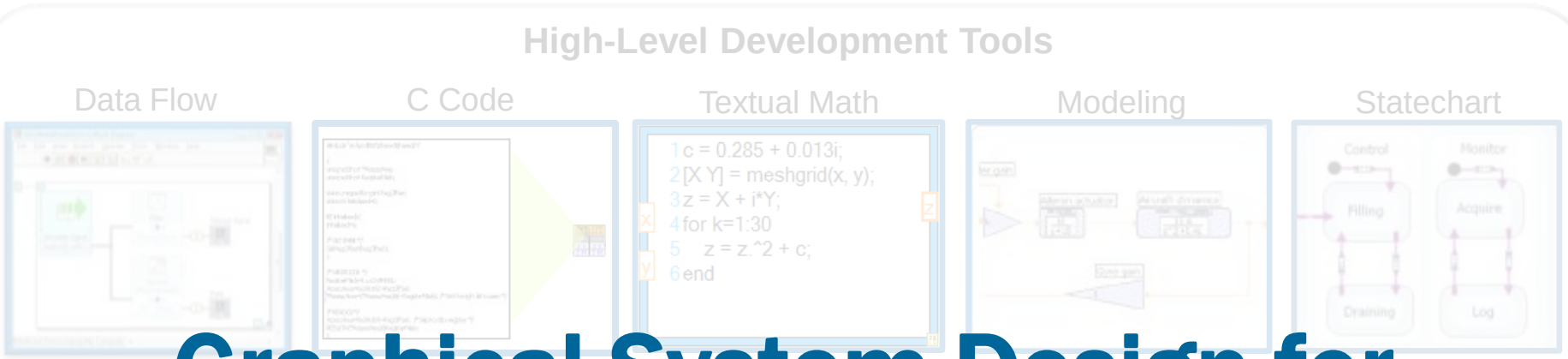


Graphical System Design for Heterogeneous Platforms

Graphical System Design Platform

May 1, 2011

*Kaushik Ravindran, **Hugo A. Andrade**, Guang Yang*

FCCM Pre-Conference Workshop:
High-Level Synthesis and Parallel Computation Models

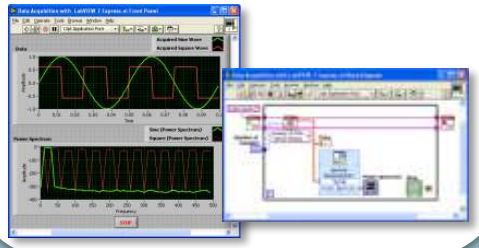
Agenda

- NI Vision Overview
- LabVIEW Context
- RIO Architecture/Platform Described
- Explicit Language Notation
- Implicit Language Notation
- Research Challenges

What We Do

Modular Measurement and Control Hardware

Productive Software Development Tools

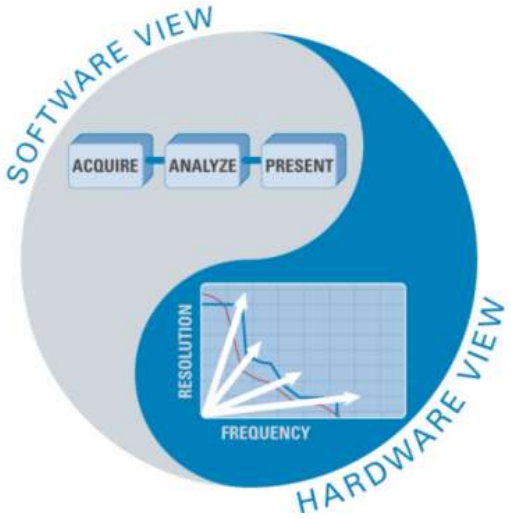


Highly Integrated Systems Platforms

Used By Engineers and Scientists for Test, Design and Control

The National Instruments Vision

"To do for test and measurement
what the spreadsheet did for financial analysis."



Virtual Instrumentation

with **NI LabVIEW™**

History of SW Control Layers & Expanding Value

1980 | 1990 | 2000 | 2010

LabVIEW FPGA & RIO

LabVIEW Real-time

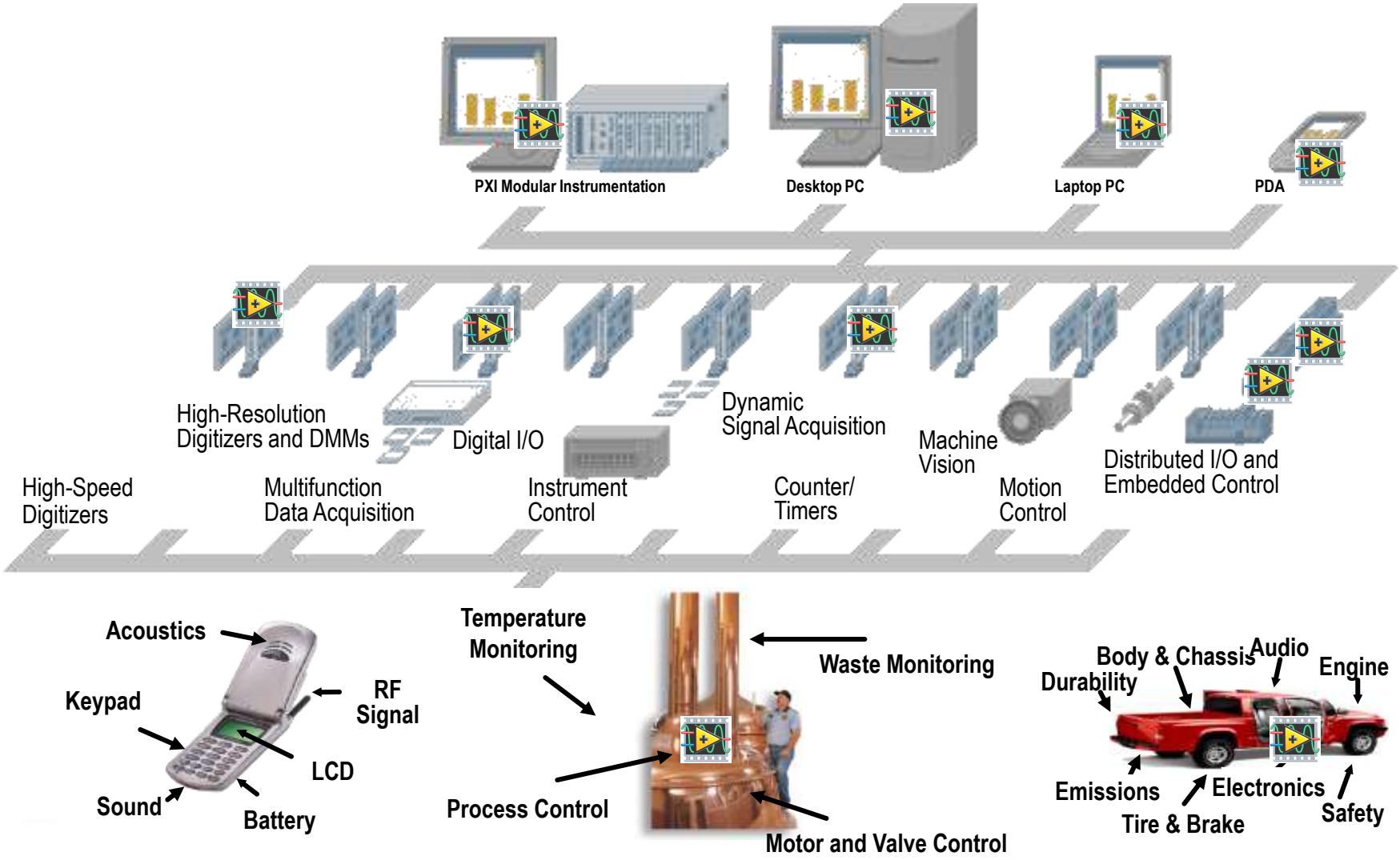
LabWindows™/CVI

LabVIEW

Instrument Control Interfaces

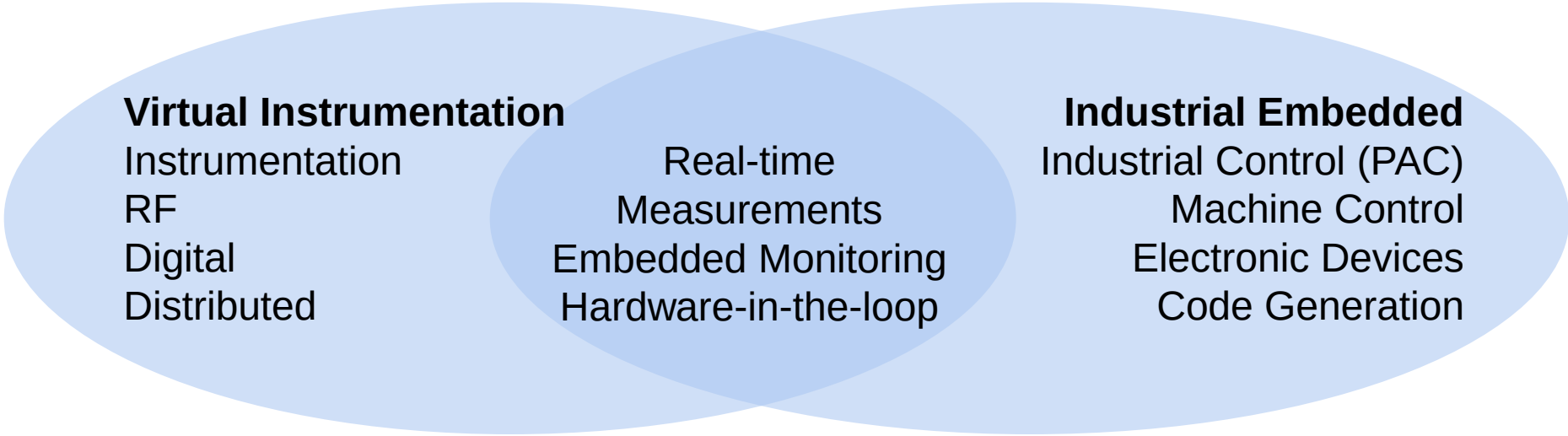
Transistor | Decreased by a factor of 2,000 in size

The NI Approach – Integrated Platforms



The National Instruments Vision Evolved…

Graphical System Design



Hardware and Software Integration differentiate our solution

"To do for test and measurement what the spreadsheet did for financial analysis."

"To do for embedded what the PC did for the desktop."

Graphical System Design

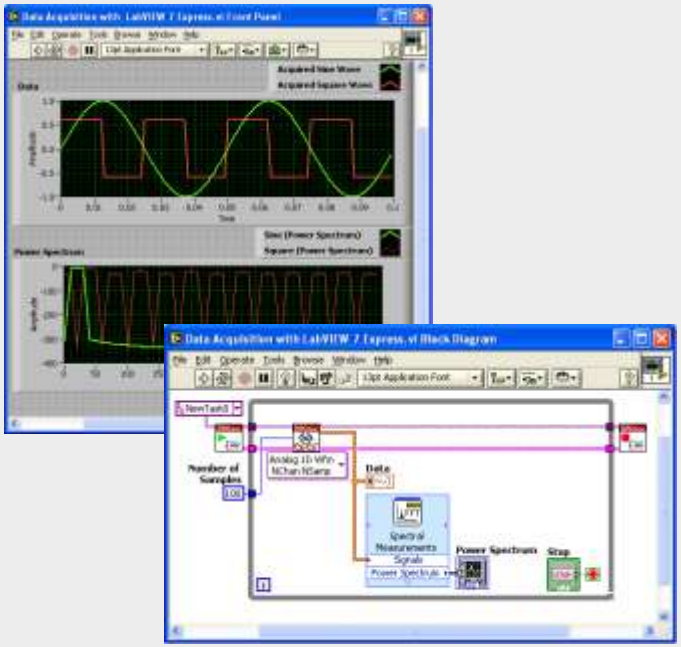
Empowering Users Through Software

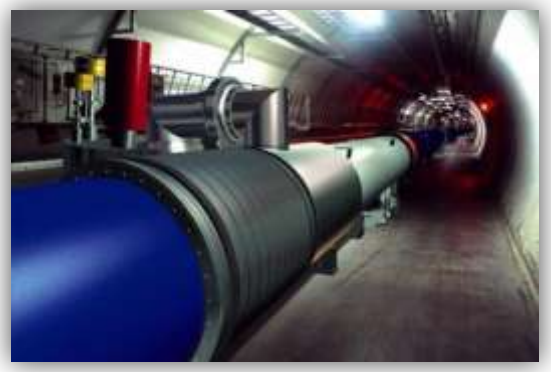
LEGO® MINDSTORMS® NXT

"the smartest, coolest toy of the year"

NATIONAL INSTRUMENTS LabVIEW™

Graphical System Design Platform



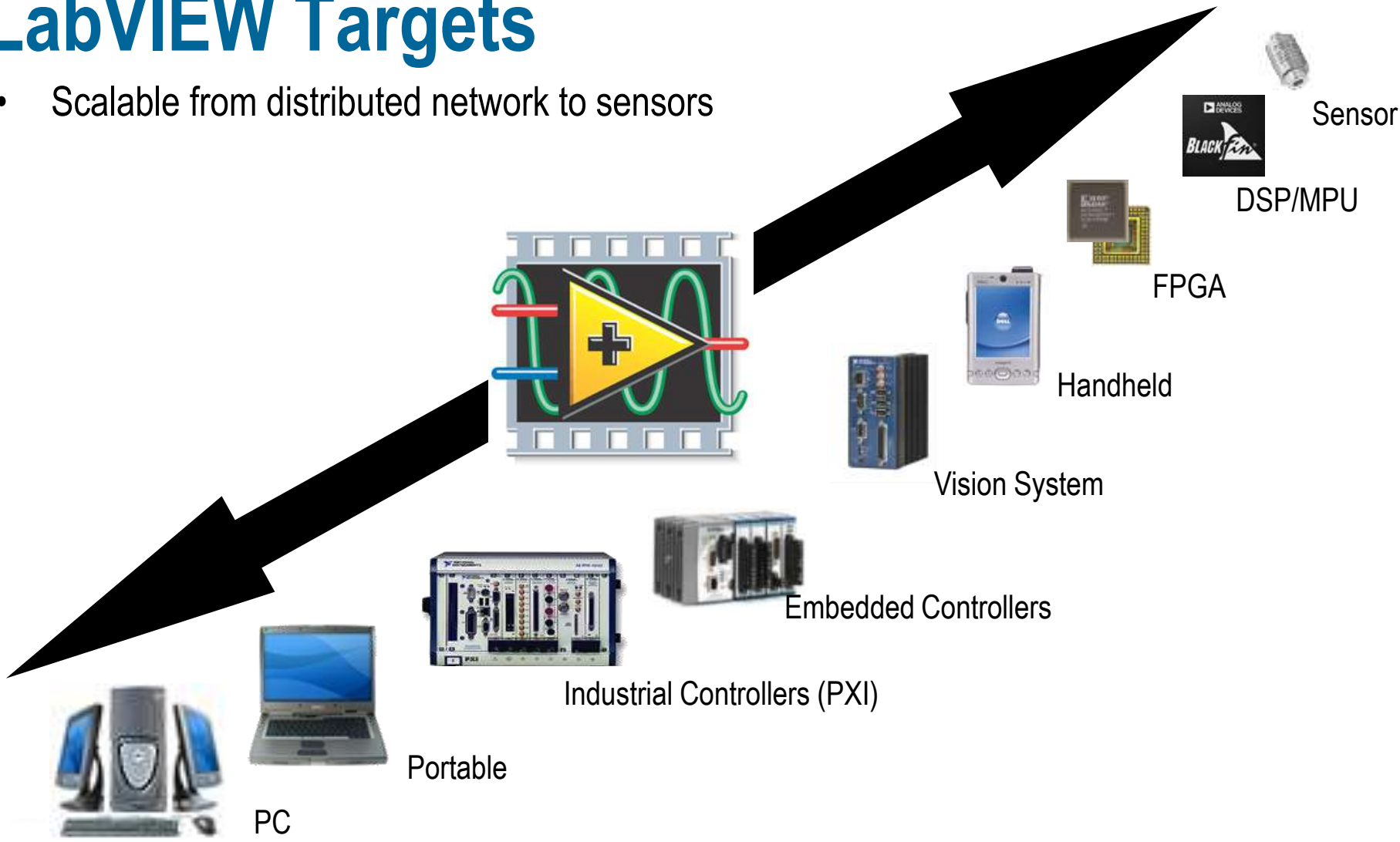


CERN Large Hadron Collider

"the most powerful instrument on earth"

LabVIEW Targets

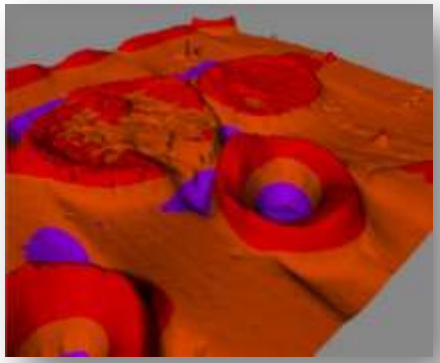
- Scalable from distributed network to sensors



High Speed & High Precision Control with LabVIEW Real-Time & FPGA

Scanning Probe Microscope with PLL

NANONIS
The SPM Application Experts

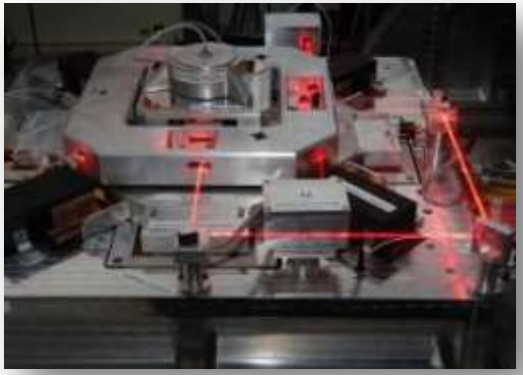


Ultrastable Atomic Force Microscope

Colorado
University of Colorado at Boulder

Nanoimprint Lithography (Tsao)

Ucla



Precision Servo-Hydraulic Control

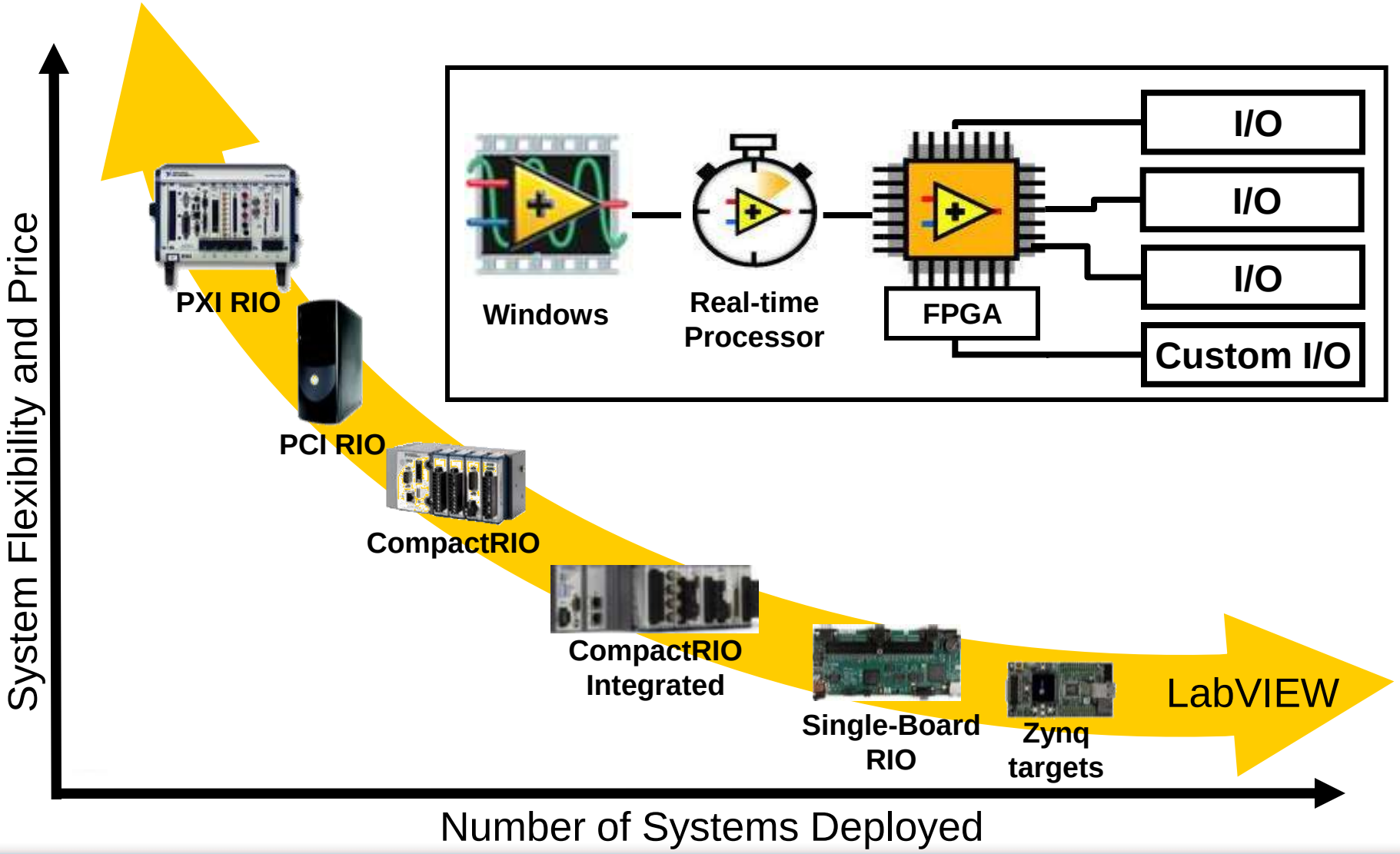
Product Technology Partners

Controlling the World's Largest Fuel-Cell Hybrid Locomotive with LabVIEW and CompactRIO

- Control and monitor the safety and operation of a 250 kW fuel-cell locomotive
- CompactRIO, LabVIEW FPGA Module, Real-Time Module
- Complex control algorithms at very fast loop rates

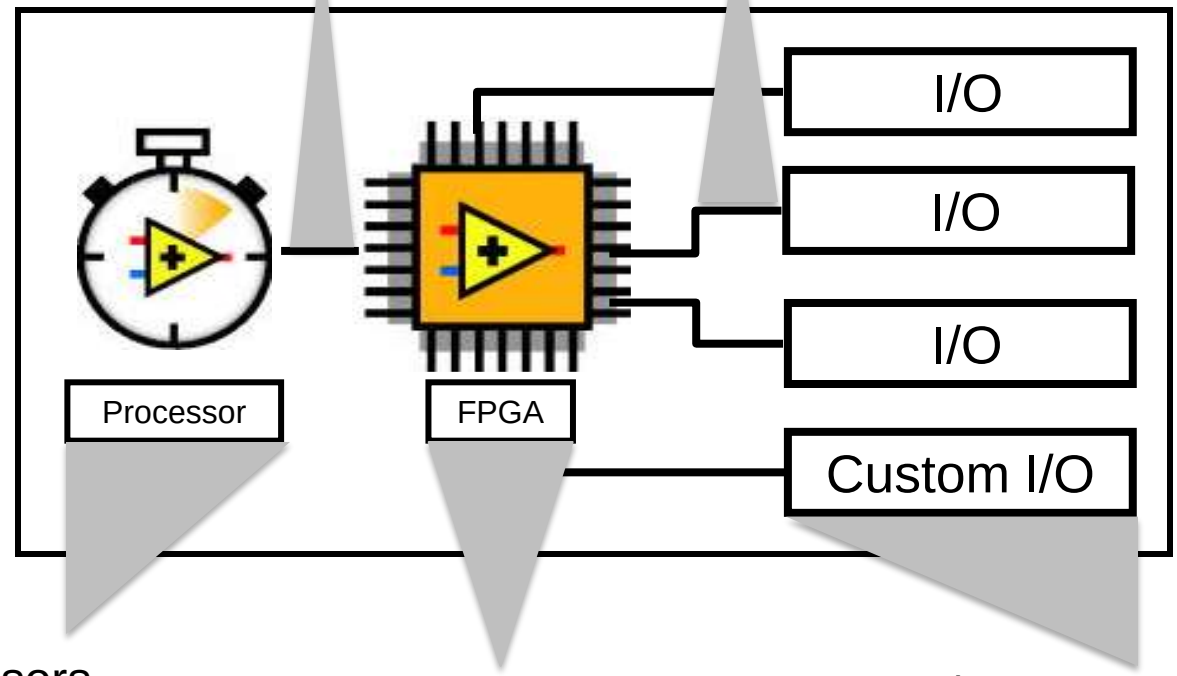
"We chose LabVIEW and CompactRIO because the NI C Series modules with integrated signal conditioning helped us implement fast monitoring of the various I/O points while connecting to a wide range of specialty sensors such as flowmeters and pressure sensors." Tim Erickson – Vehicle Projects LLC

Scalable Platform…



High-Speed Data Streaming
- Synchronize memory access
- Fast data links for maximum performance

A/D Technology
- Multirate sampling
- Individual channel triggering



Microprocessors
- Floating-point processing
- Communications
- Multicore technology
- Reprogrammable

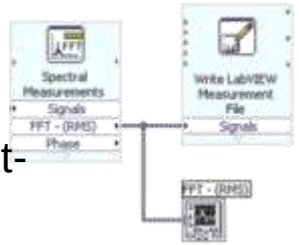
FPGAs
- High-speed control
- High-speed processing
- Reconfigurable
- True Parallelism
- High Reliability

I/O
- Custom timing & triggering
- Modular I/O
- Calibration
- Custom modules

LabVIEW Real-Time

- Multicore programming

- Analysis, control and communication functions

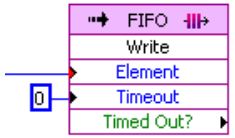


- Integrate C code and text-based math

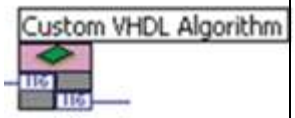
LabVIEW FPGA

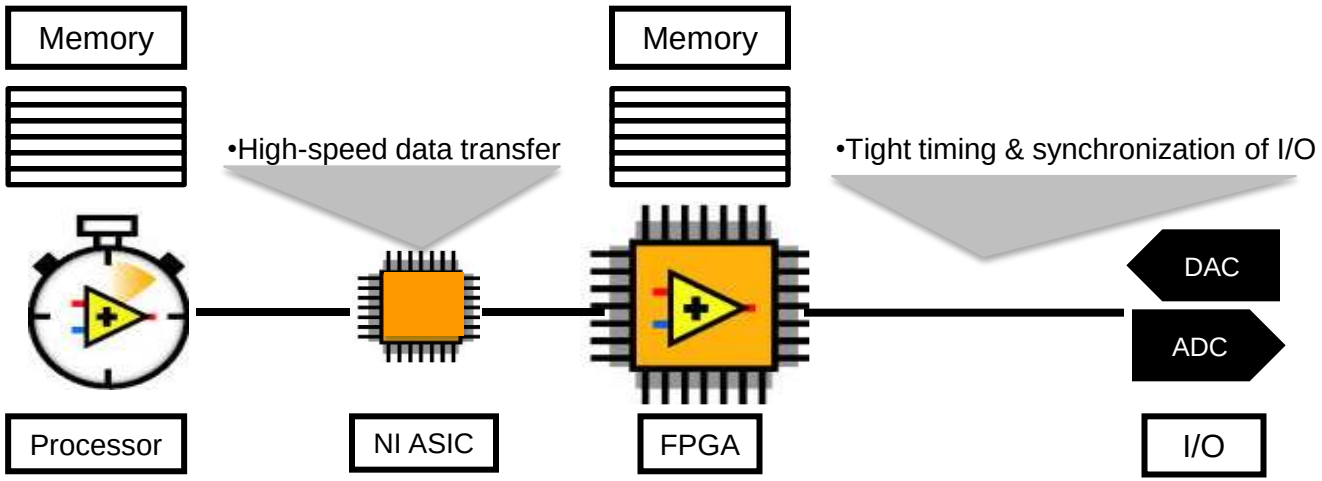
- Graphical FPGA Design

- Fixed-point processing



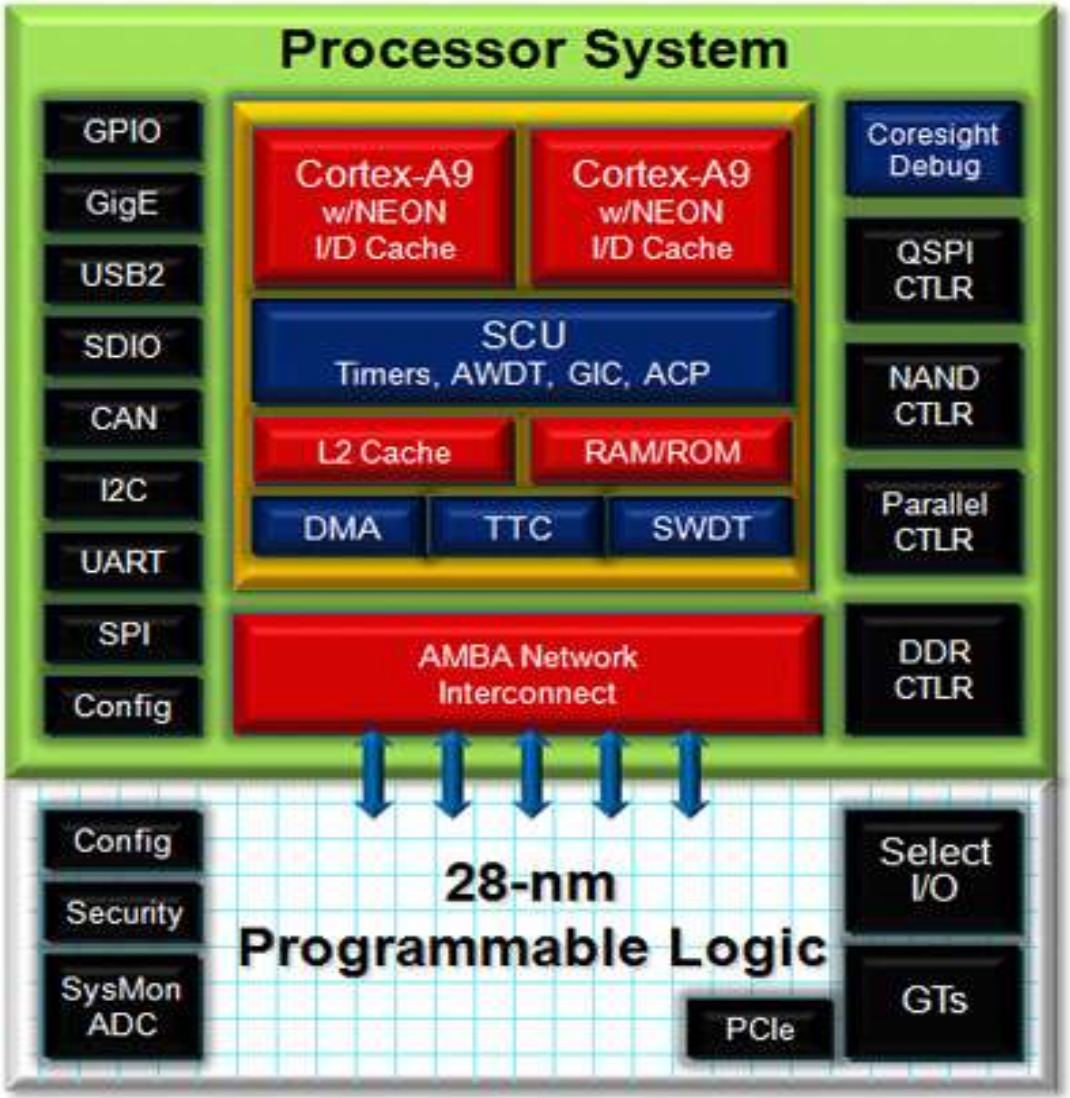
- Analysis, control and communication functions
- Integrate VHDL IP





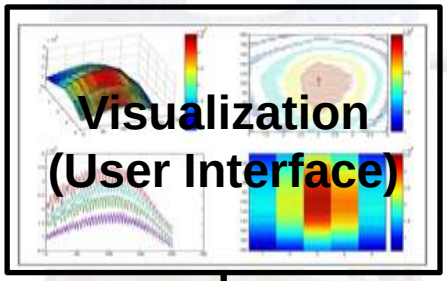
Future uP and FPGA in one Chip

XILINX® Zynq Extended Processing Platform



HPC meets tough Real-Time Challenges

Solving the most sophisticated control applications



FPGA

Sensors

Actuators

Multicore CPU

Sensors

Actuators

FPGA

Sensors

Actuators

Multicore CPU

Sensors

Actuators

GPGPU

Sensors

Actuators

GPGPU

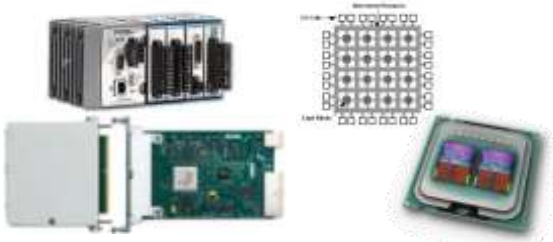
Sensors

Actuators

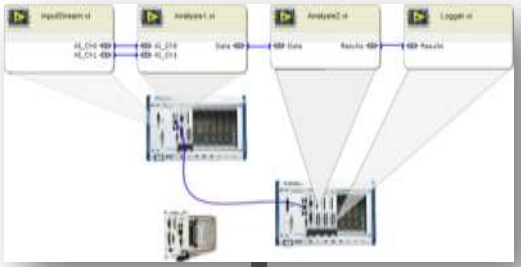
The Y-Chart System Design Methodology

Application Logic

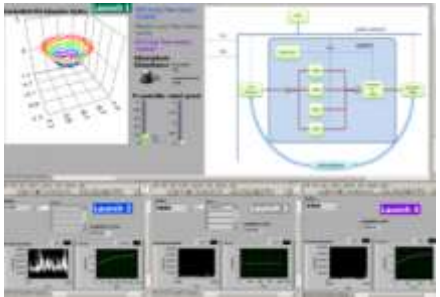
Platform Architecture



Analysis & Mapping



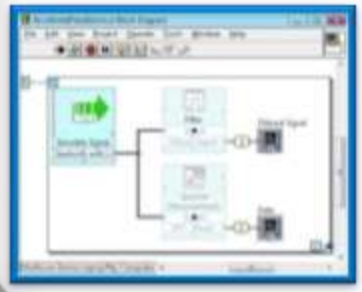
Performance Evaluation



1. Kienhuis, Deprettere, van der Wolf, and Vissers., "A Methodology to Design Programmable Embedded Systems - The Y-Chart Approach. Embedded Processor Design Challenges: Systems, Architectures, Modeling, and Simulation" - SAMOS, p.18-37, Jan. 2002.
2. Keutzer, Newton, Rabaey, Sangiovanni-Vincentelli, "System-level Design: Orthogonalization of Concerns and Platform-based Design," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 19(12): p. 1523-1543, Dec. 2000.

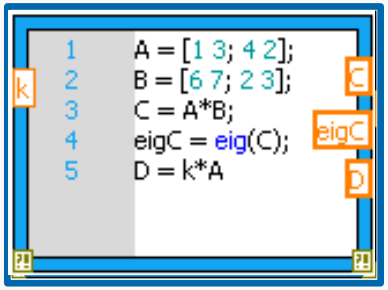
High-Level Design Models

Dataflow

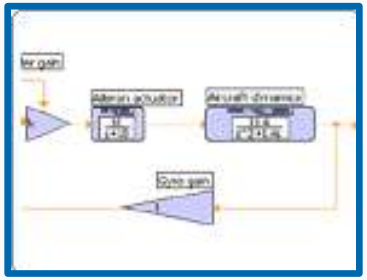


C Code

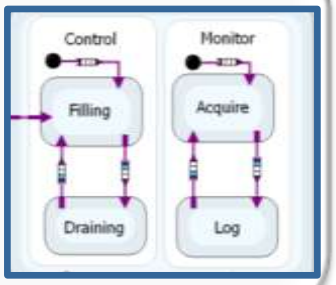
Textual Math



Simulation



Statechart



NATIONAL INSTRUMENTS LabVIEW™

Graphical System Design Platform

PC/Mac/Linux

PXI

CompactRIO

FlexRIO

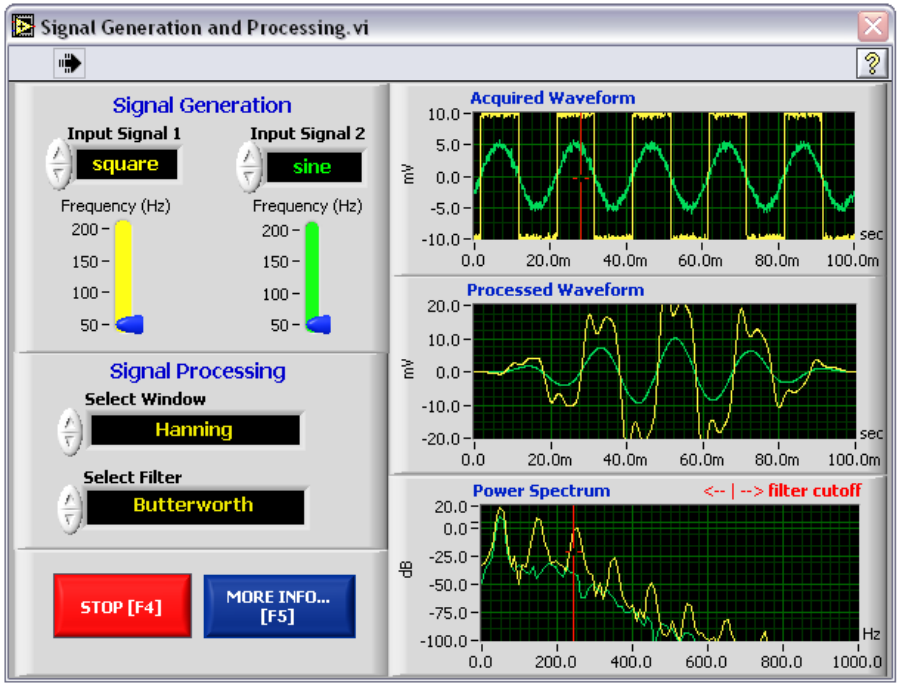
Custom

LabVIEW Today – LabVIEW 2010

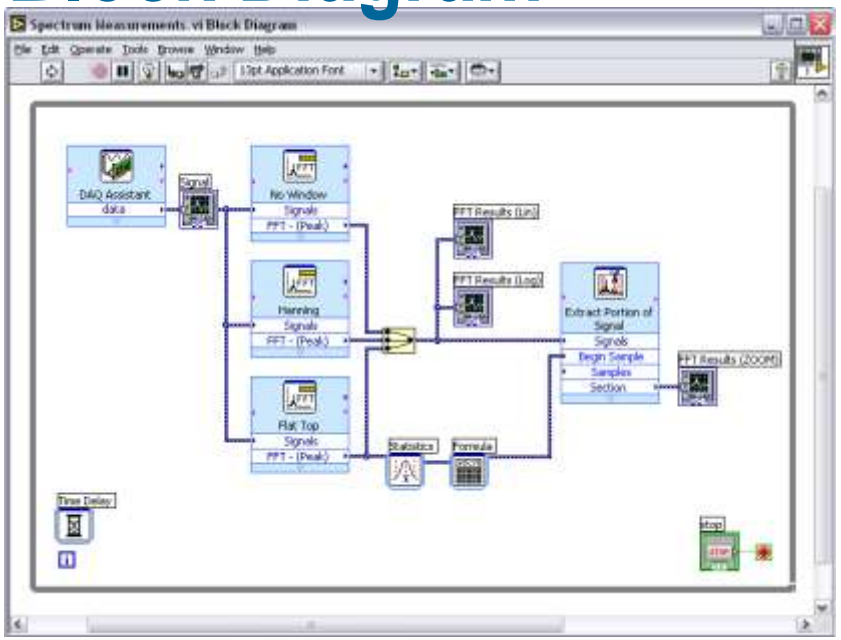
- LabVIEW
  - What is LabVIEW
  - Product Family
- Embedded System Design
- Embedded Design Platforms Brochure
- Downloadable Slides from Embedded Design Session

LabVIEW Virtual Instrument

Front Panel

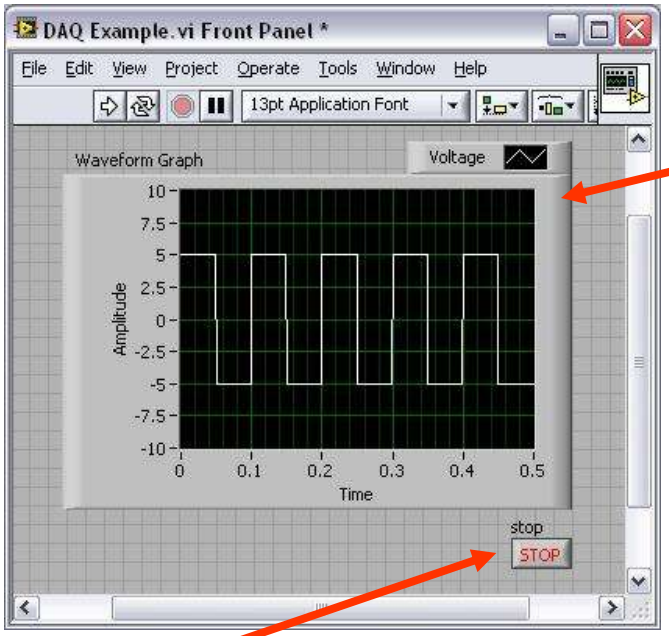


Block Diagram



Creating a VI

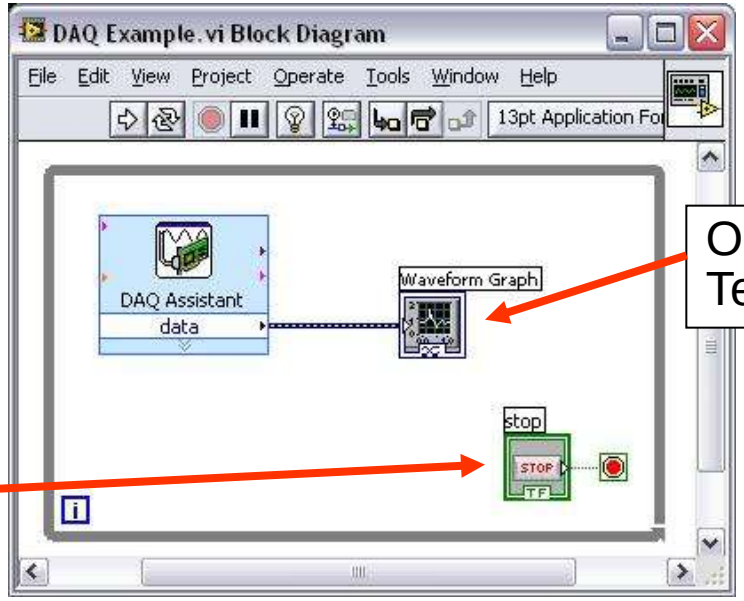
Front Panel Window



Graph Indicator

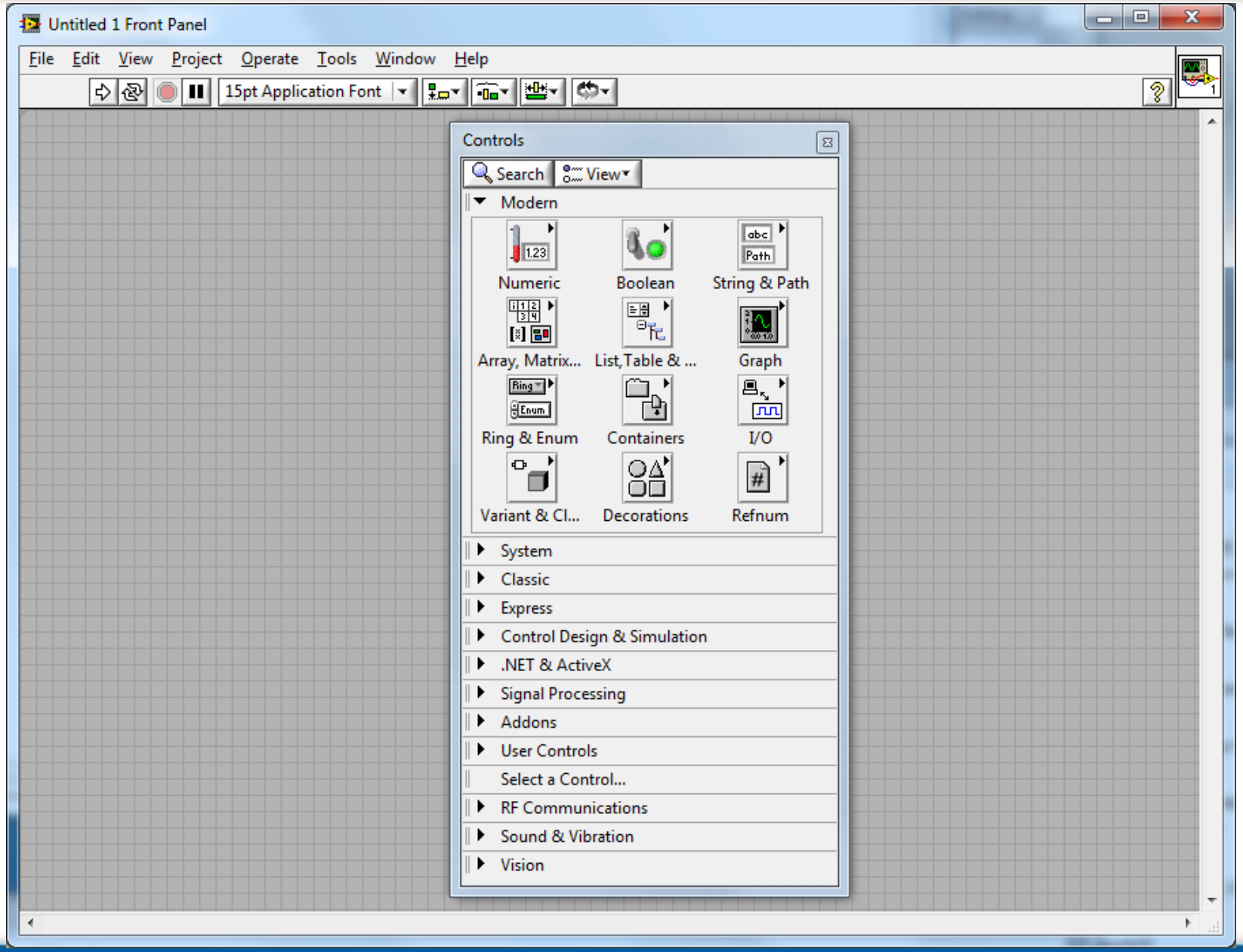
Boolean Control

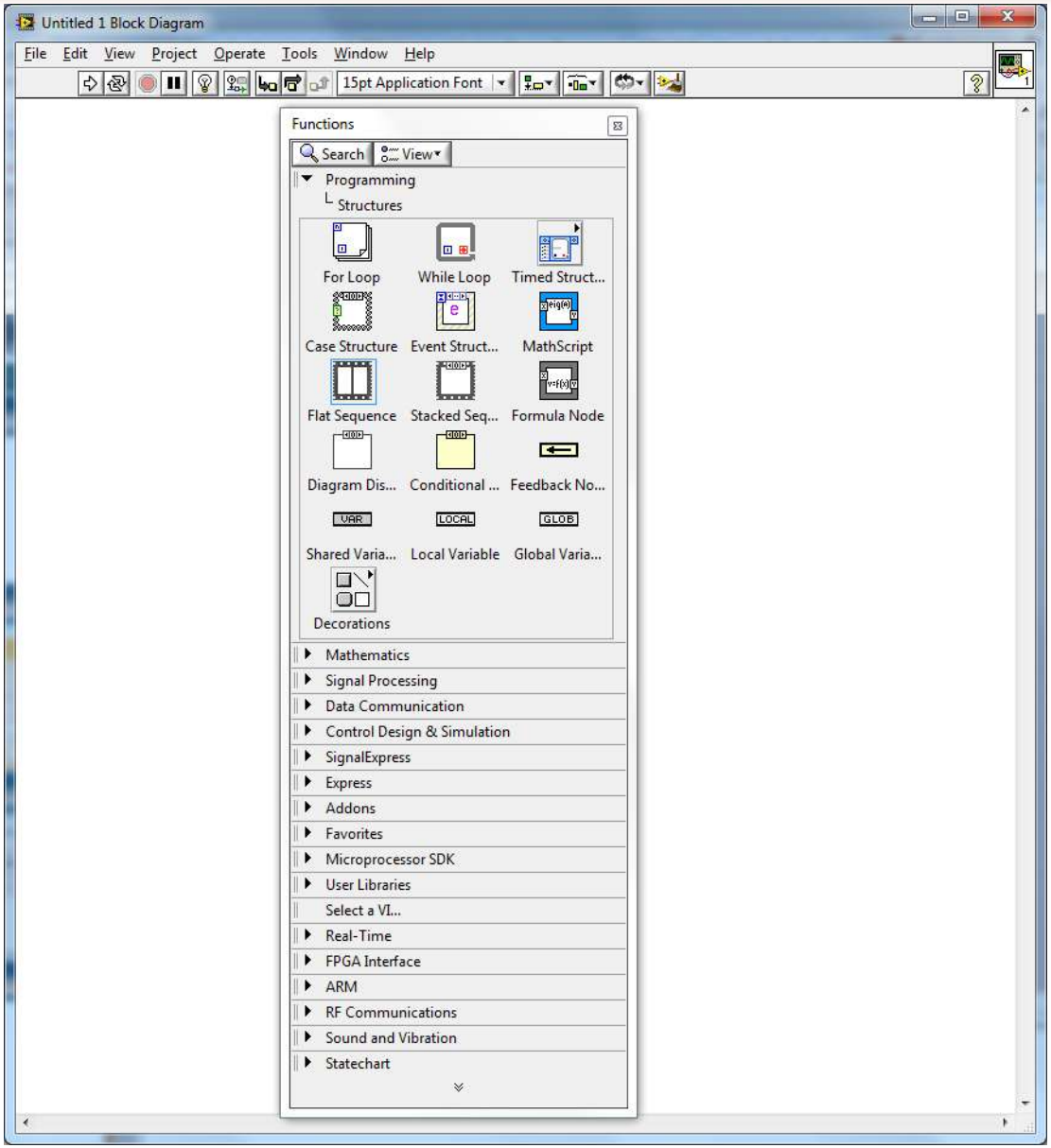
Block Diagram Window



Output Terminal

Input Terminals



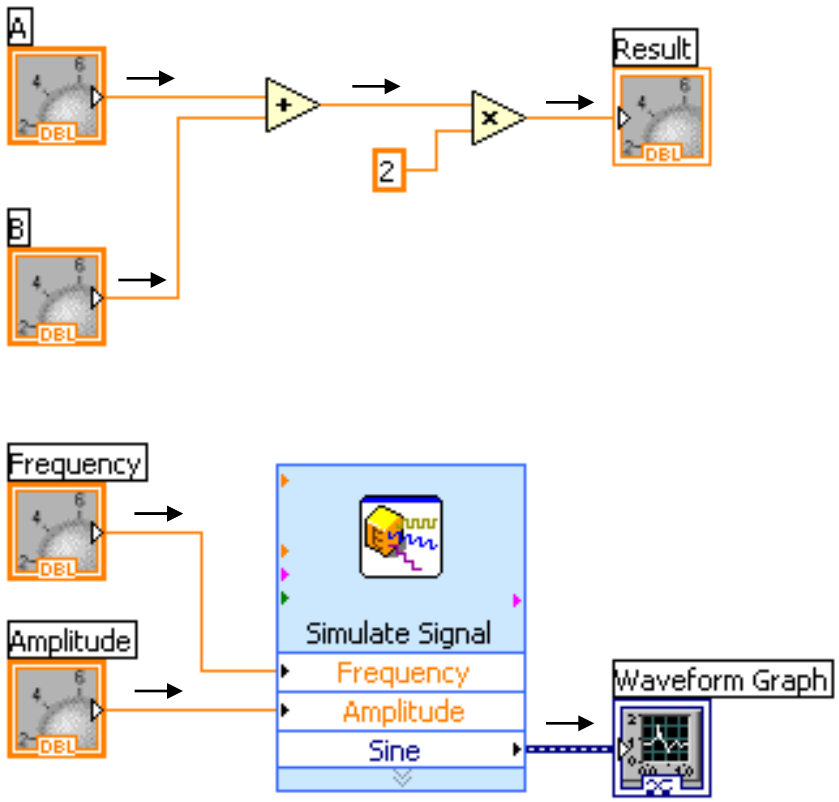


The G (LabVIEW) Language Model

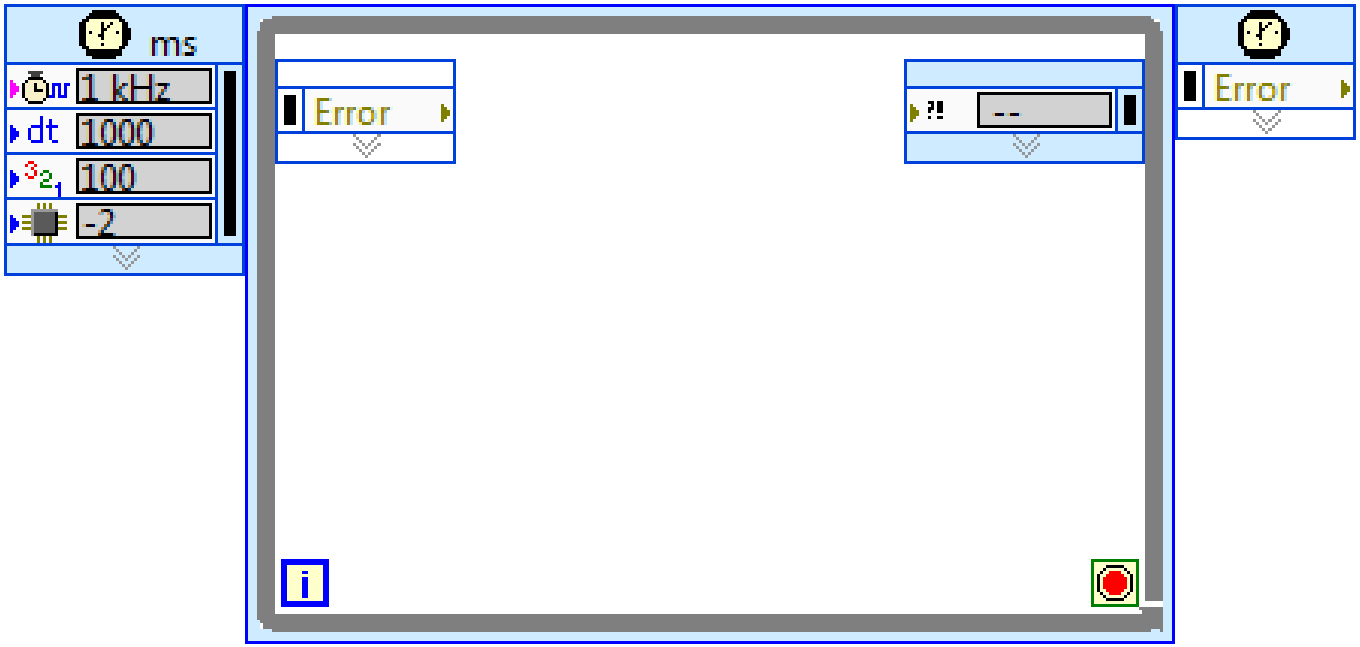
- Homogenous dataflow language
  - Structured case (switch, select) and loops
    - “Structured dataflow”
- Run-time scheduling
  - Explicit task level parallelism
  - Implicit parallelism heuristically identified
- Synthesizable language
  - To machine code on x86 and PPC processors
  - To VHDL for FPGAs
  - To C for embedded processors
- Turing complete

Dataflow Programming

- Block diagram execution
  - Dependent on the flow of data
  - Block diagram does NOT execute left to right
- Node executes when data is available to ALL input terminals
- Nodes supply data to all output terminals when done



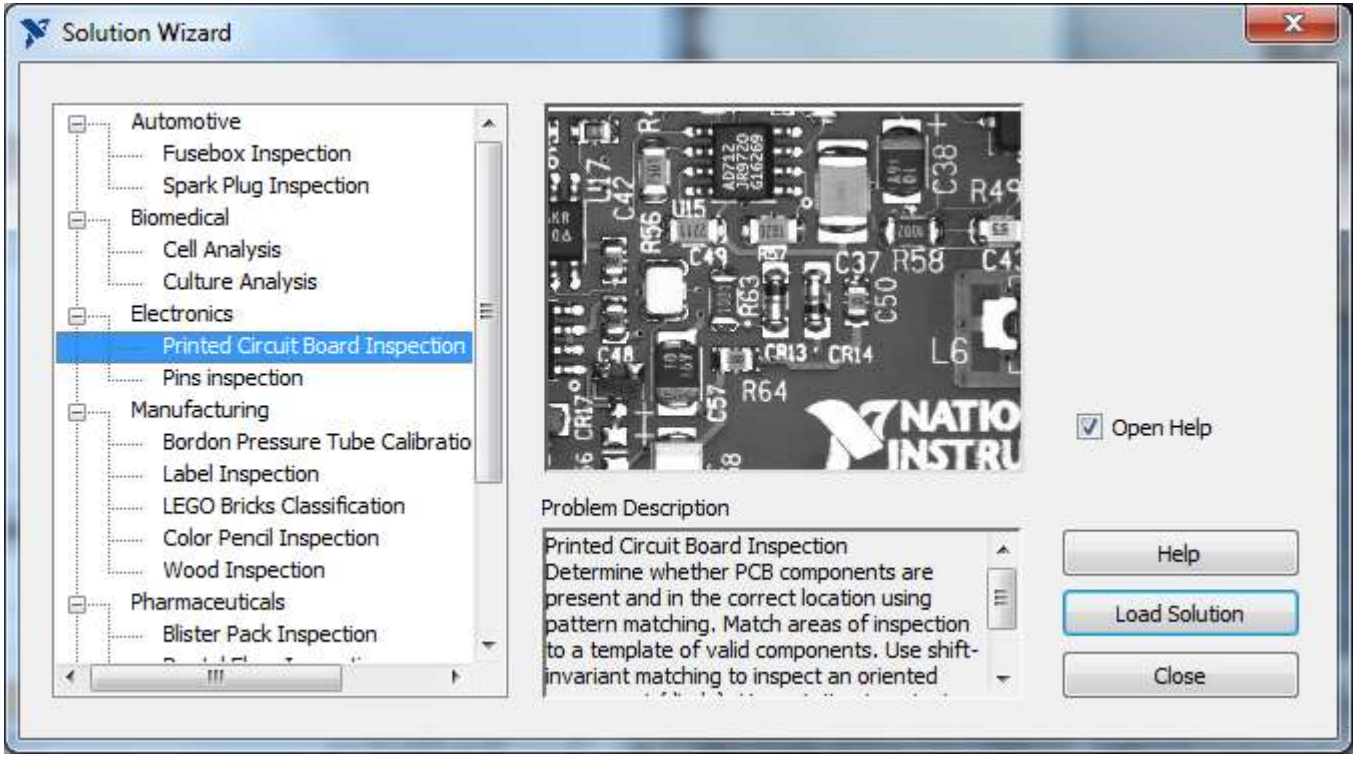
Structured Dataflow



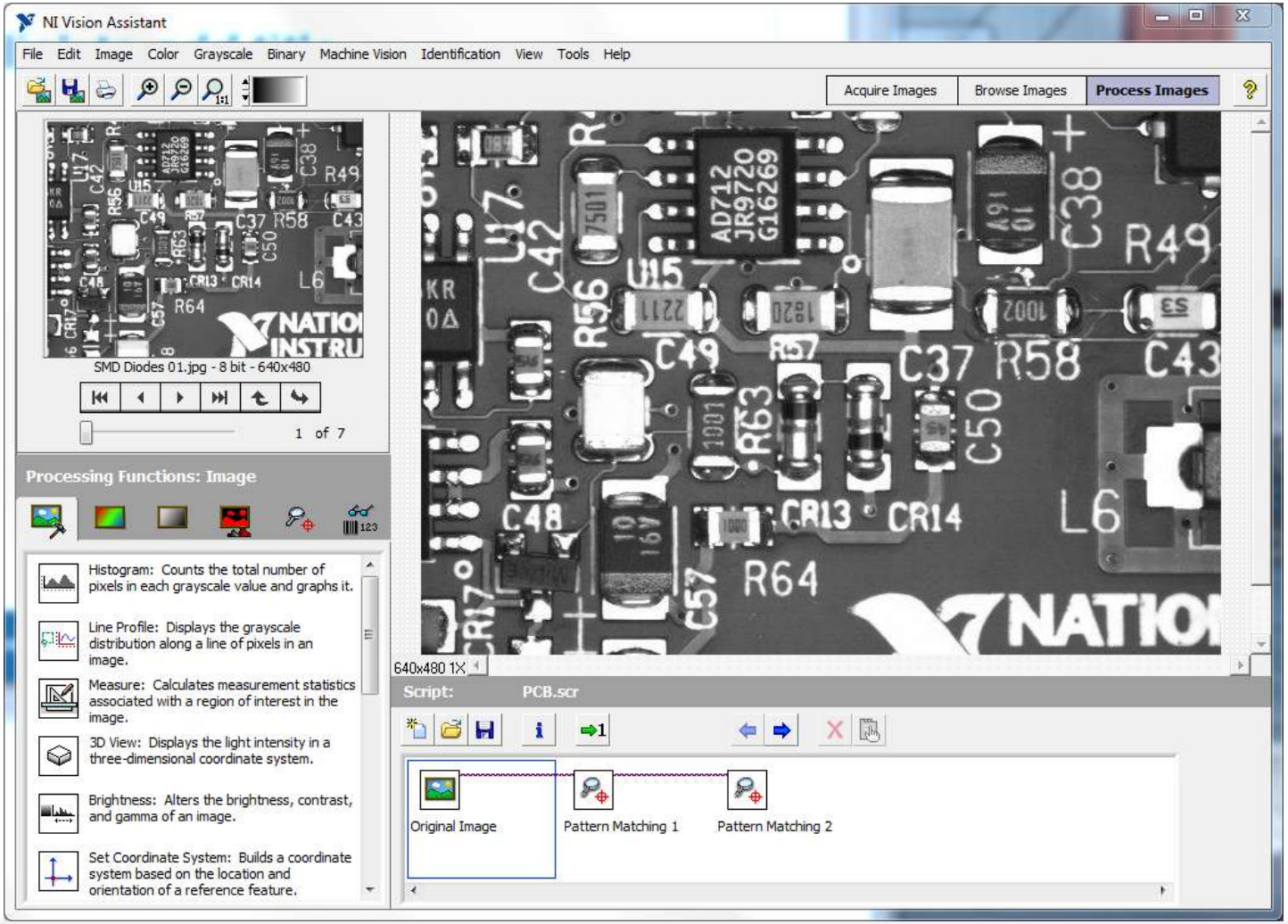
LabVIEW as a Target Language

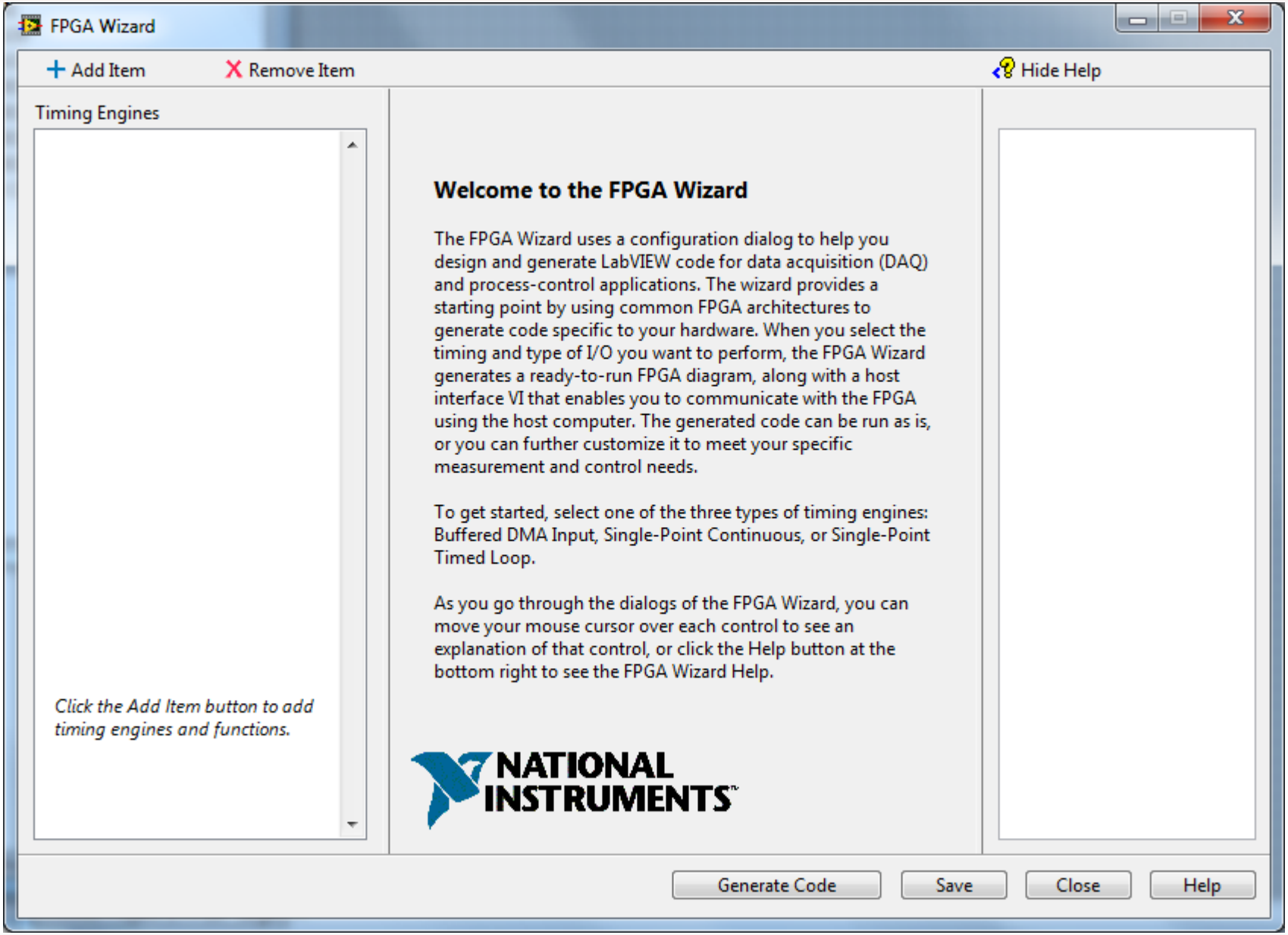
- Application Wizards – Patterns
- StateCharts
- MathScript
- Control and Simulation Diagram
- Express Nodes and X-nodes
- I/O Nodes

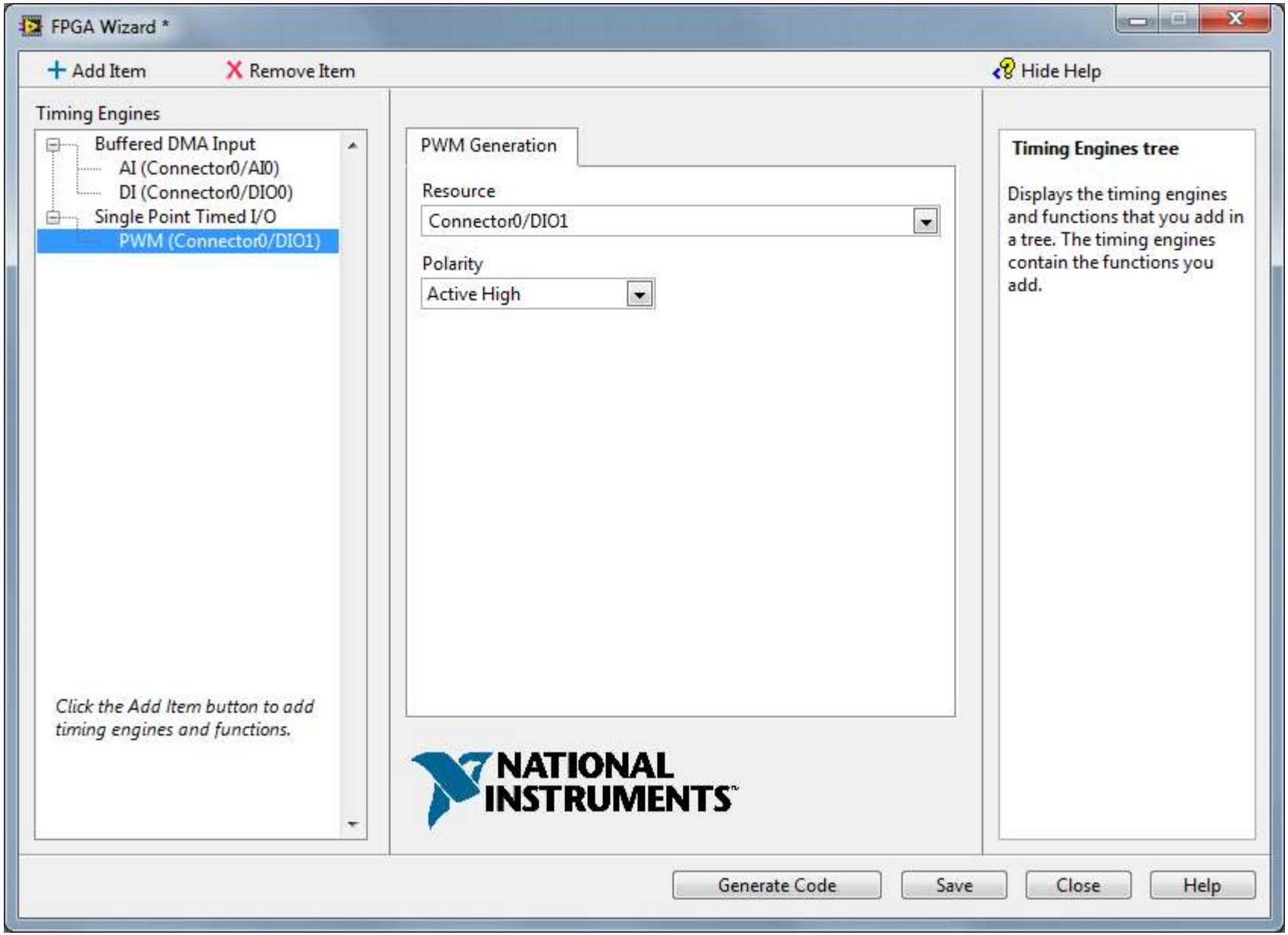
Application Wizards - Patterns

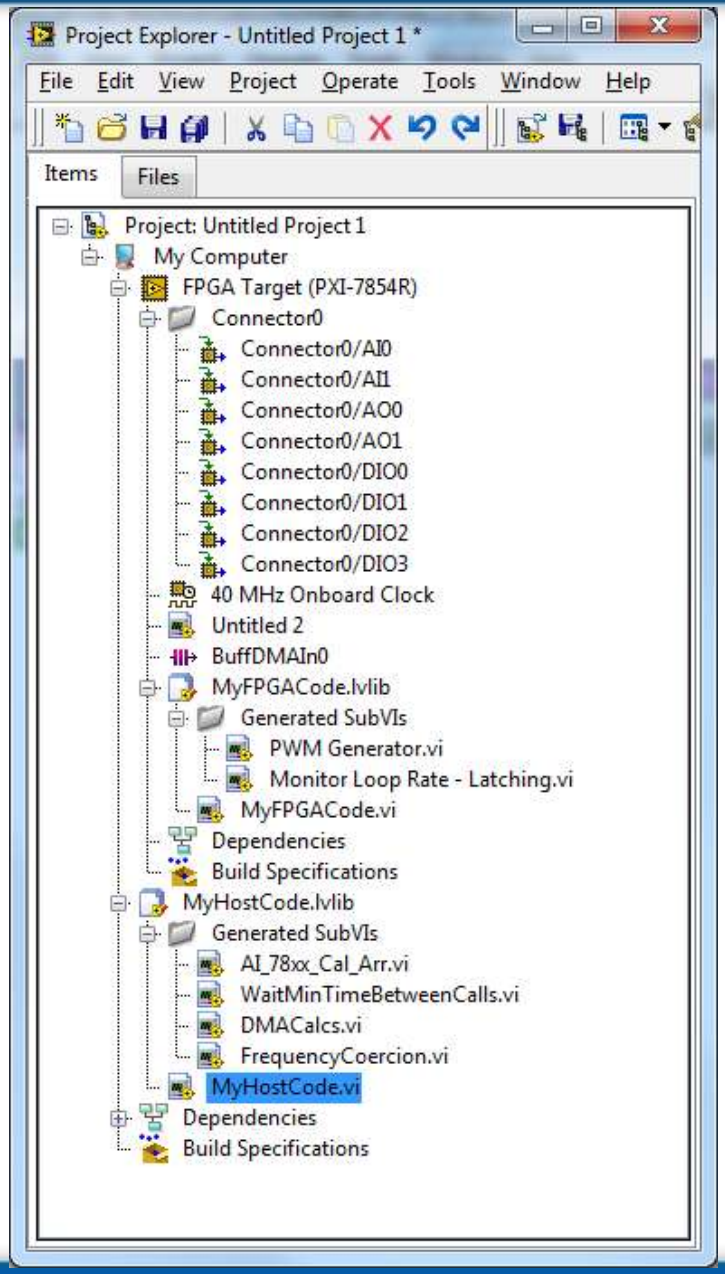


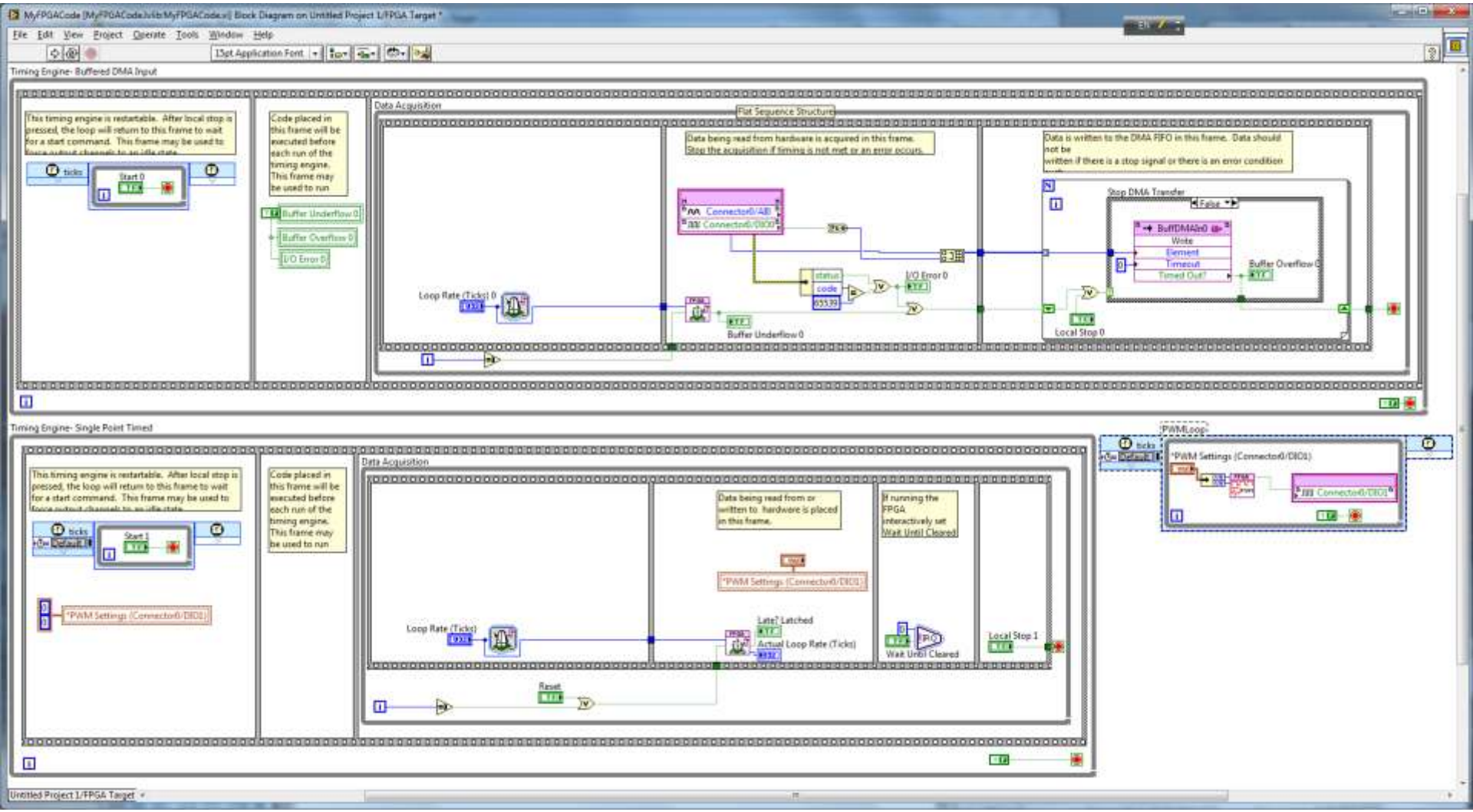
Application Wizards - Patterns




FPGA Wizard
Add Item
Remove Item
Hide Help
Timing Engines
Click the Add Item button to add timing engines and functions.
Welcome to the FPGA Wizard
The FPGA Wizard uses a configuration dialog to help you design and generate LabVIEW code for data acquisition (DAQ) and process-control applications. The wizard provides a starting point by using common FPGA architectures to generate code specific to your hardware. When you select the timing and type of I/O you want to perform, the FPGA Wizard generates a ready-to-run FPGA diagram, along with a host interface VI that enables you to communicate with the FPGA using the host computer. The generated code can be run as is, or you can further customize it to meet your specific measurement and control needs.
To get started, select one of the three types of timing engines: Buffered DMA Input, Single-Point Continuous, or Single-Point Timed Loop.
As you go through the dialogs of the FPGA Wizard, you can move your mouse cursor over each control to see an explanation of that control, or click the Help button at the bottom right to see the FPGA Wizard Help.
NATIONAL INSTRUMENTS
Generate Code
Save
Close
Help


FPGA Wizard *
Add Item
Remove Item
Hide Help
Timing Engines
Buffered DMA Input
AI (Connector0/AI0)
DI (Connector0/DIO0)
Single Point Timed I/O
PWM (Connector0/DIO1)
Click the Add Item button to add timing engines and functions.
PWM Generation
Resource
Connector0/DIO1
Polarity
Active High
NATIONAL INSTRUMENTS
Timing Engines tree
Displays the timing engines and functions that you add in a tree. The timing engines contain the functions you add.
Generate Code
Save
Close
Help





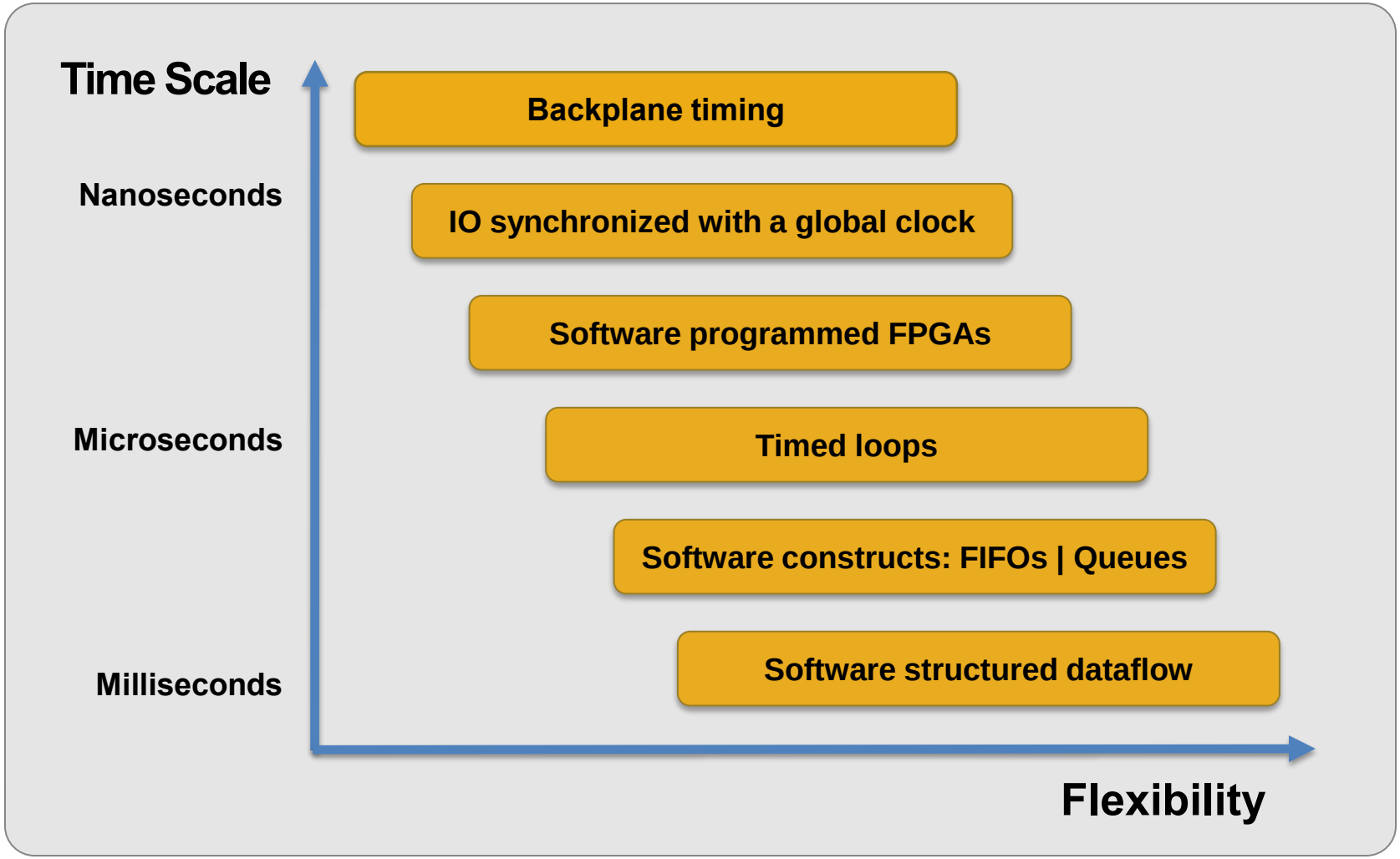
System Deployment

- Target aware synthesis
- I/O Port Abstraction
  - I/O Classes
  - Protocol generation
- Channel Abstraction
  - FIFO
  - Loop-to-loop
  - Peer-to-peer
  - Board-to-host (DMA)

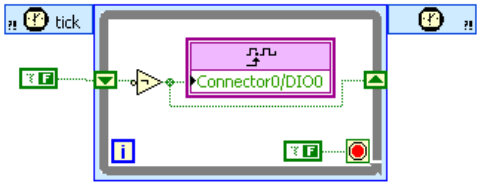
System Deployment

- Timing
  - Expressing an order
    - Language constructs
    - Operating Environments
  - Reality of Platform timing
    - Static analysis

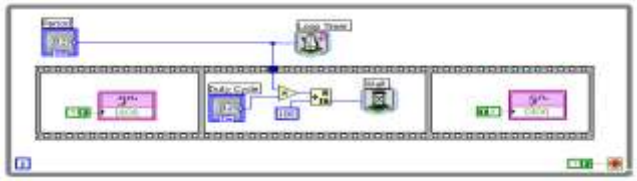
System Level Integration of Time



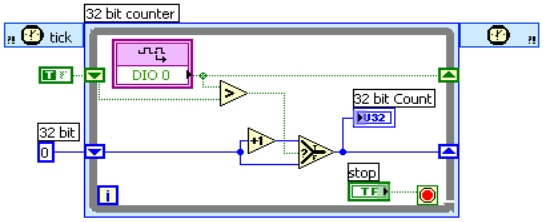
FPGA-based I/O Applications



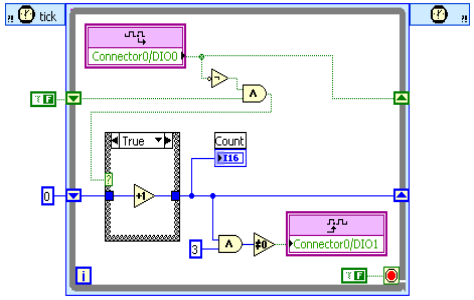
Clocks



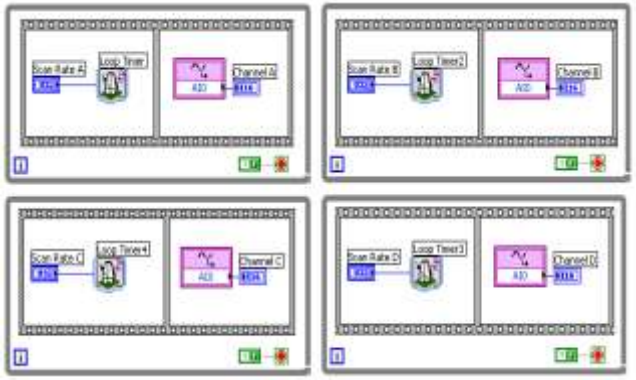
PWM



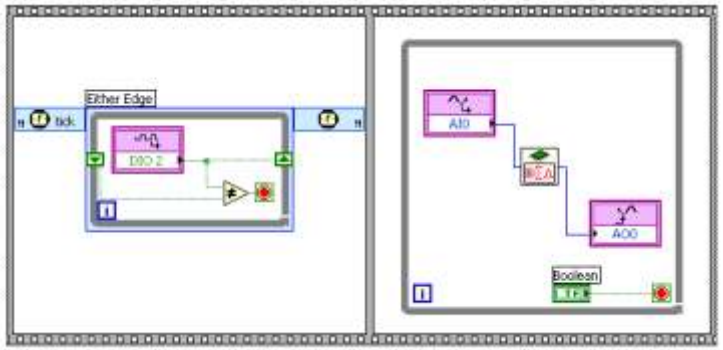
Counters



Custom Counters

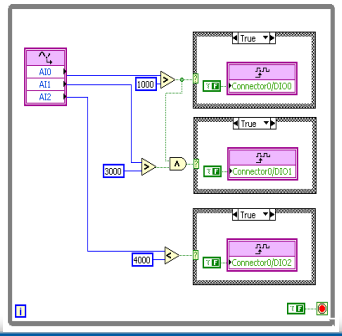


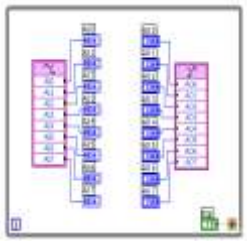
Multiple Scan Rates



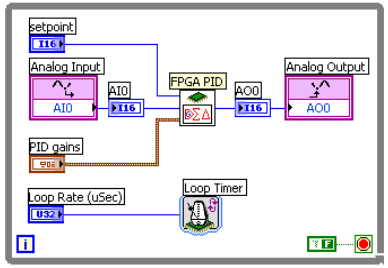
Custom Timing and Synchronization

Custom Analog Triggering





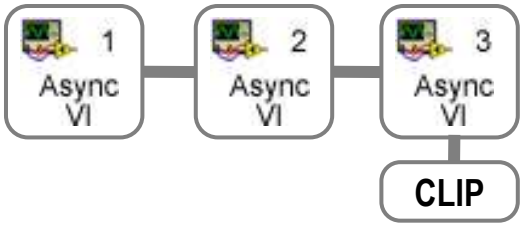
Custom Analog I/O



Built-in IP Processing Blocks

The Challenge Going Forward

Concurrent Application



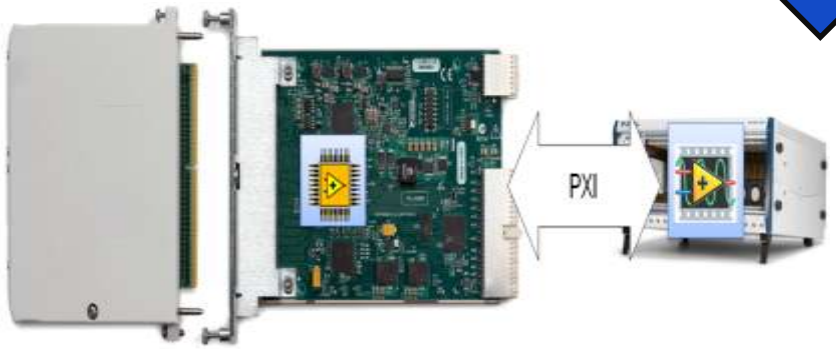
Application Trends

- **1000's of parallel tasks**
- Large node/channel counts
- High performance requirements
- E.g. streaming DSP applications

Implementation Gap

How to map the tasks and data in a concurrent application to the processing and memory resources on a platform?

Parallel Platform



Platform Trends

- **100's of processing elements**
- Heterogeneous processors and memories
- Distributed I/O
- E.g. FPGA targets

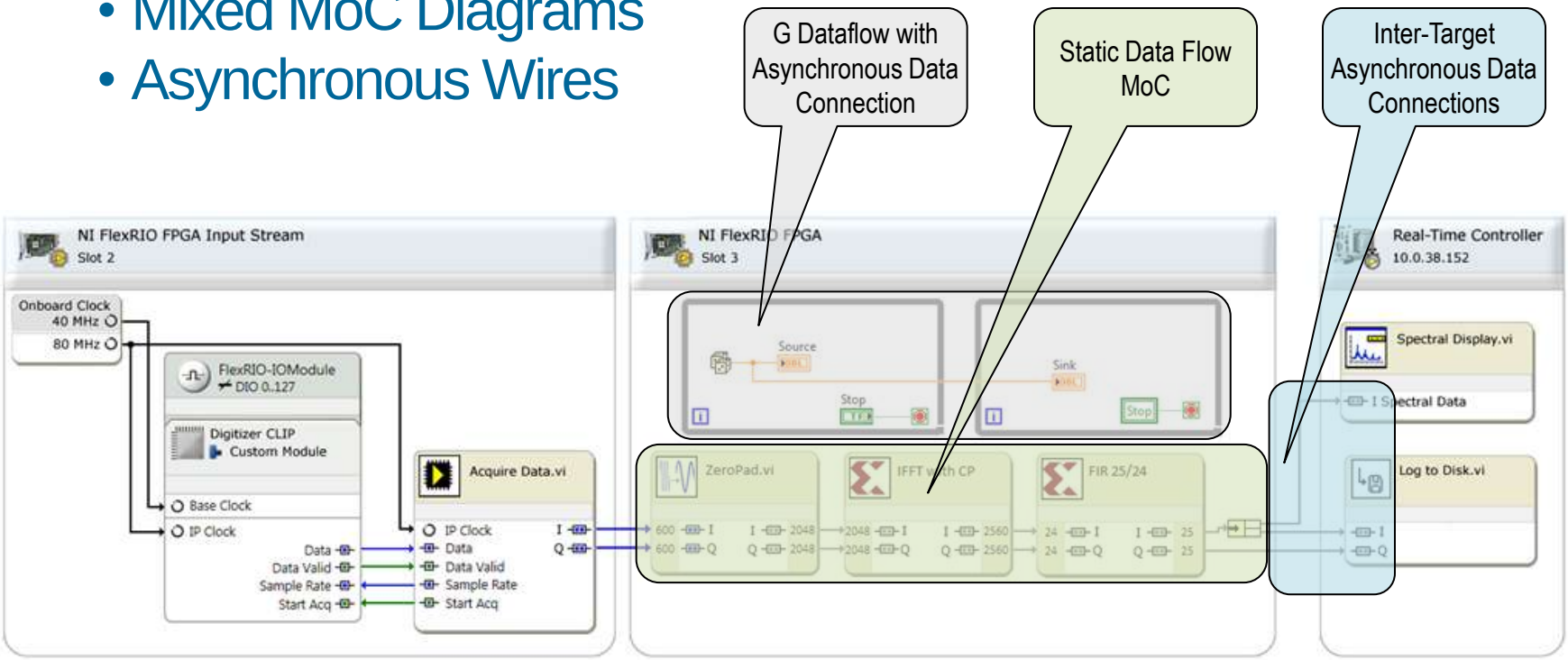
Key Challenges

- Model of computation
- Analysis and optimization back end
- Performance models and timing library
- Actor definition
- IP modeling and integration
- Simulation and verification
- Code generation and implementation

Modeling System-Level Designs

System-level designs introduce new modeling constructs:

- Systems
- Targets
- Mixed MoC Diagrams
- Asynchronous Wires



LabVIEW System Designer

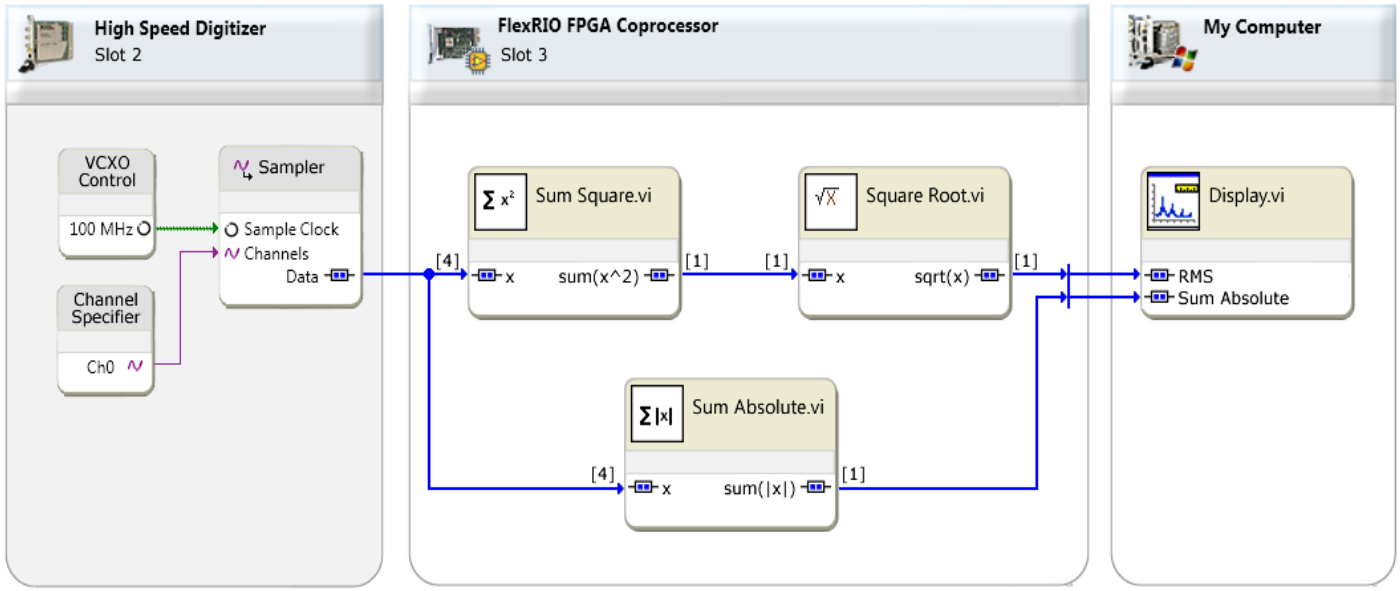
Home View Search

Compile Deploy Run Abort | Cut Copy Paste Undo Redo Delete | Snap To Grid Snap To Edges Show Gridlines Show Snaplines | Create Schedule View Schedule Iterations to Schedule: 2 Show Firing Ratios Show Latency | Recommend Buffer Sizes Accept All

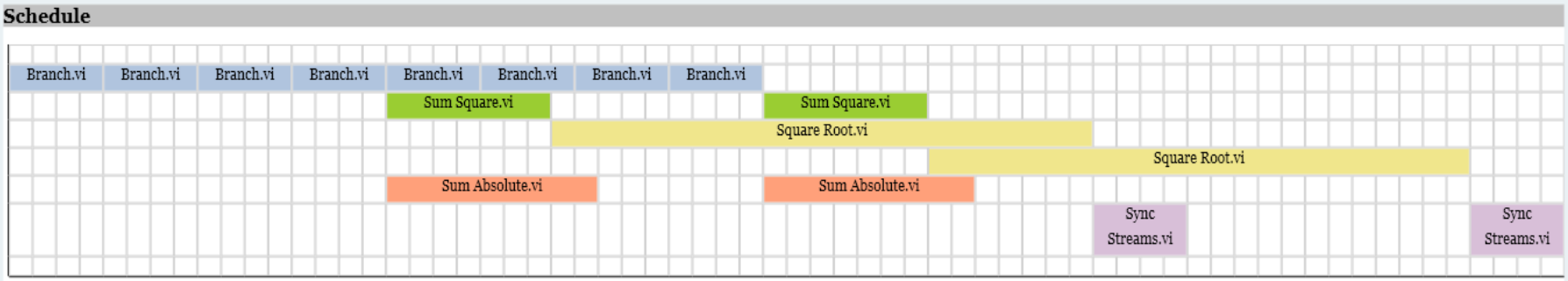
Execute Editing Alignment Schedule Buffers

Streaming Computation.lvsd Welcome

Palette Properties



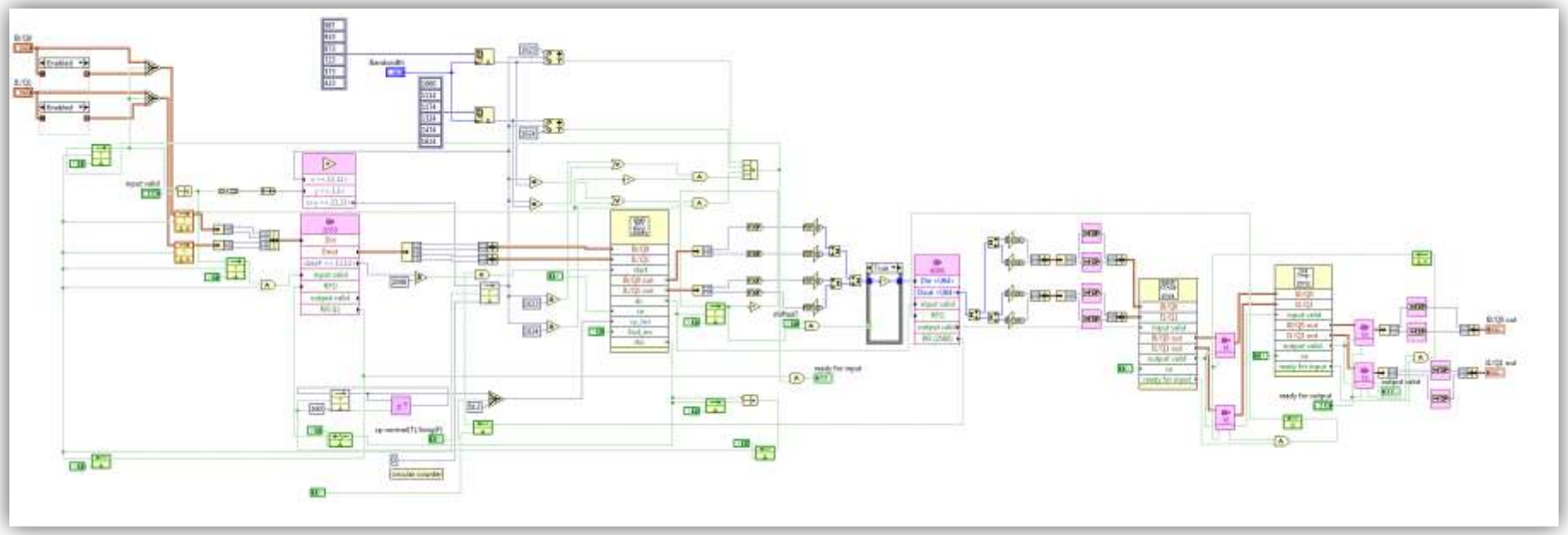
Schedule



AVI (Task) Estimates | Terminal Size Estimates | Schedule

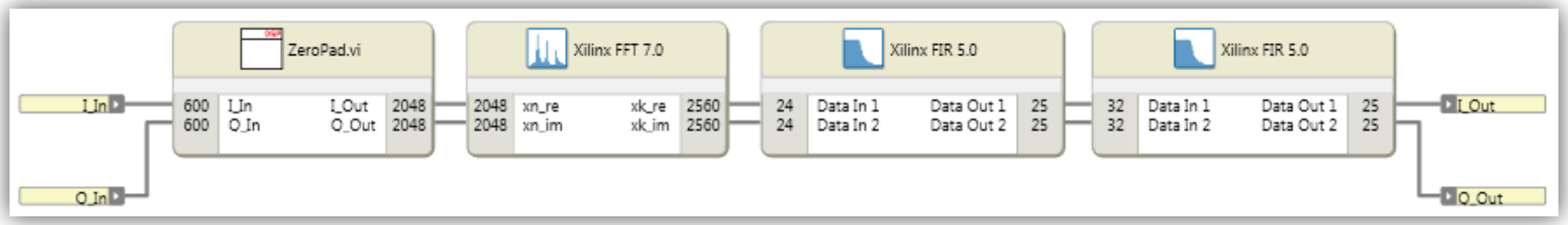
Change In Progress 1046.00, 424.00 100%

High-Speed Streaming is Complex Today



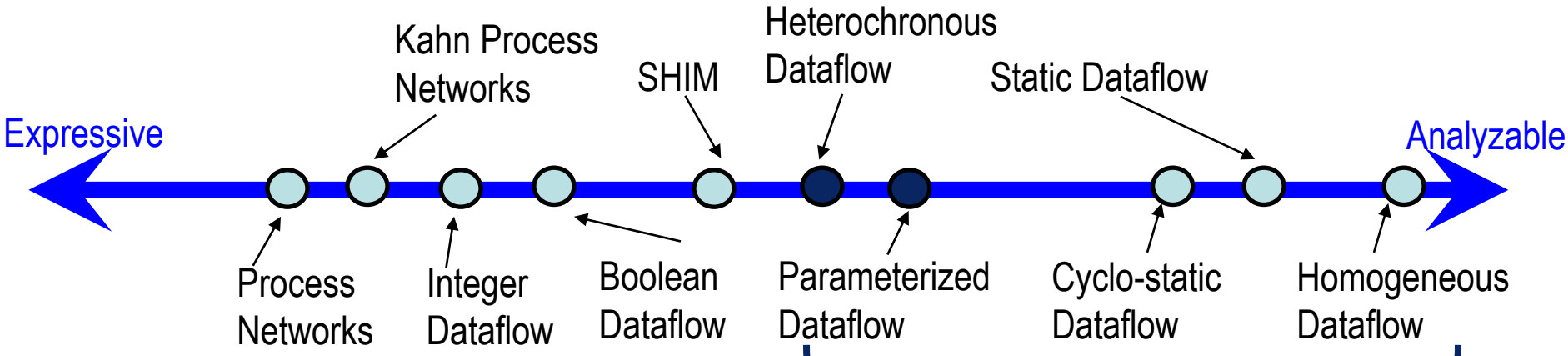
- Challenges
  - LabVIEW G model
    - Original specification from algorithm designer
    - Not feasible for highly efficient implementation on FPGA targets
  - Implementation challenges
    - Floating to fixed point conversion
    - Array data to point-by-point data conversion
    - Explicit concurrency representation
    - FPGA target constraints
    - Integration with internal and third-party IP

Domain Expert Expectations for High-Speed Streaming



- High-level DSP representation that matches algorithm theory
  - Algorithms written independently of hardware target
  - Deal in domain terms of token rate, throughput, and latency
- Explore high-level design tradeoffs without diving into implementation details
  - Tune performance with high-level constraints
  - Access the details if needed

MoCs for Streaming Applications



Deter-ministic?	No	Yes
Synchronous?	No	Yes
Deadlock and boundedness decidable?	No	Yes
Static scheduling?	No	Yes

Analyzable Models

Key trade-off: Analyzability vs. Expressibility

[1] Edward A. Lee, "Concurrent Models of Computation for Heterogeneous Software", EECS 290, 2004.
[2] Stephen Edwards, "SHIM: A Deterministic Model for Heterogeneous Embedded Systems", UCB EECS Seminar, 2006.
[3] Thanks: Abhijit Davare, UCB.

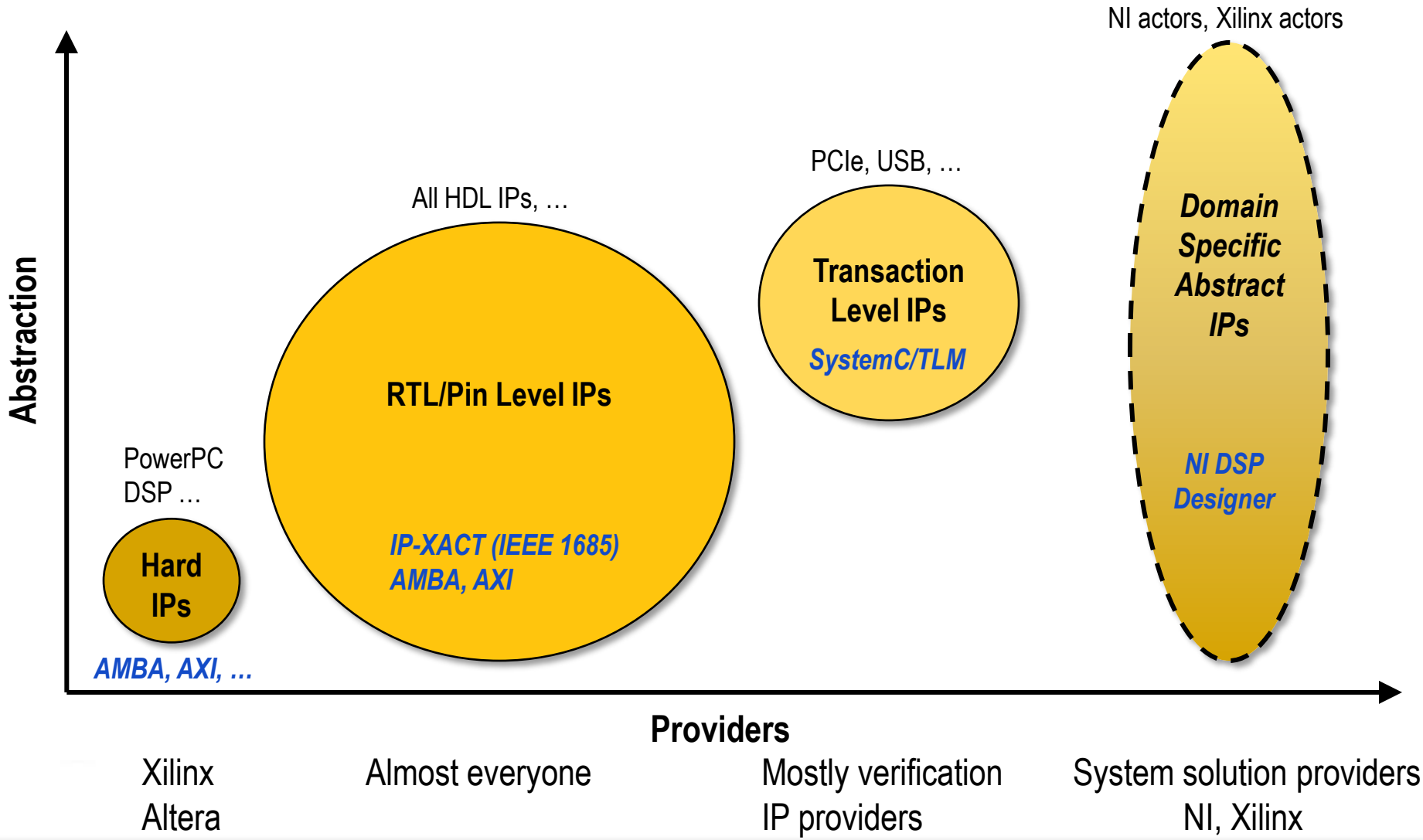
Analysis and Optimization Features

- Core dataflow optimizations
  - Model validation (deadlock and unboundedness detection)
  - Throughput and latency computation
  - Buffer size optimization (under throughput constraints)
  - Schedule computation
- Hardware specific optimizations
  - Resource constrained schedule computation
  - Actor fusion
  - Joint optimization of latency, throughput, and buffer size
  - Rate matching
  - IP configuration selection
  - Implementation strategy selection

Directions Ahead

- Graphical syntax and analysis extensions
  - Parameterized cyclo-static dataflow (PCSDF) model
  - Access patterns for hardware implementations
- Specification for control and timing with dataflow
  - Scenario aware dataflow
  - Heterochronous dataflow
- Other hardware specific problems
  - Self timed Vs scheduled implementation strategy selection
  - IP interface standardization

Re-use Drives IP Abstraction Levels

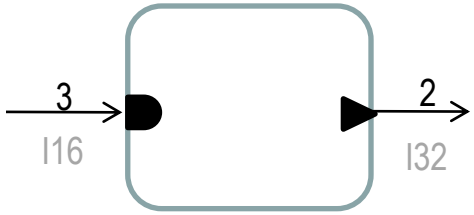


Current Challenges of IP Integration

- Fragmented IP that lacks standards
  - Some standards on meta-data and structural interfaces (IP-XACT), and protocols (AXI)
- But vendors not adopting standards to:
  - Describe IP Interface
  - Capability
  - Behavior
  - Provide coherent simulation models
  - Pragmatically provide an integration experience for configuring the IP
  - Interface to high-level description languages

Describe Just Enough IP for the Domain Expert

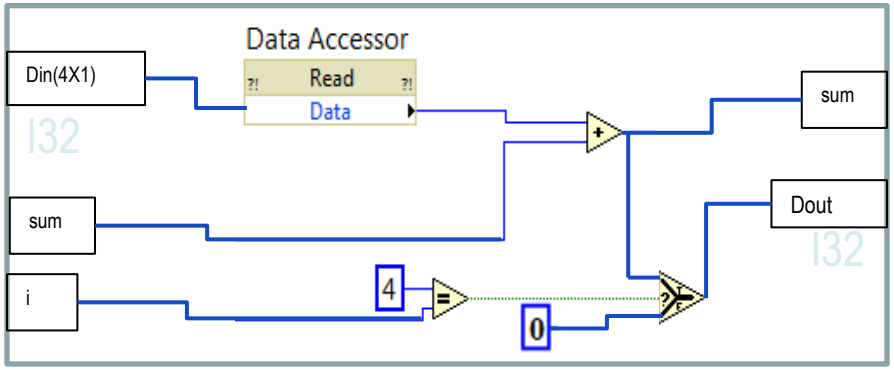
DSP Designer User



Modeling Concerns:
- MoC Behavior
- Simulation
- Exploration
- Analysis

Describe IP Protocol Details for the Tools

Actor Designer

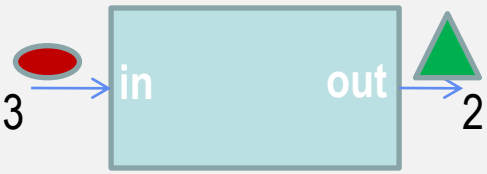


Implementation Concerns:
- Protocol details
- Cycle accurate behavior
- Optimized Code Gen

Basic Description of IP <IC, OC, II, ET, IE, IP, OP>

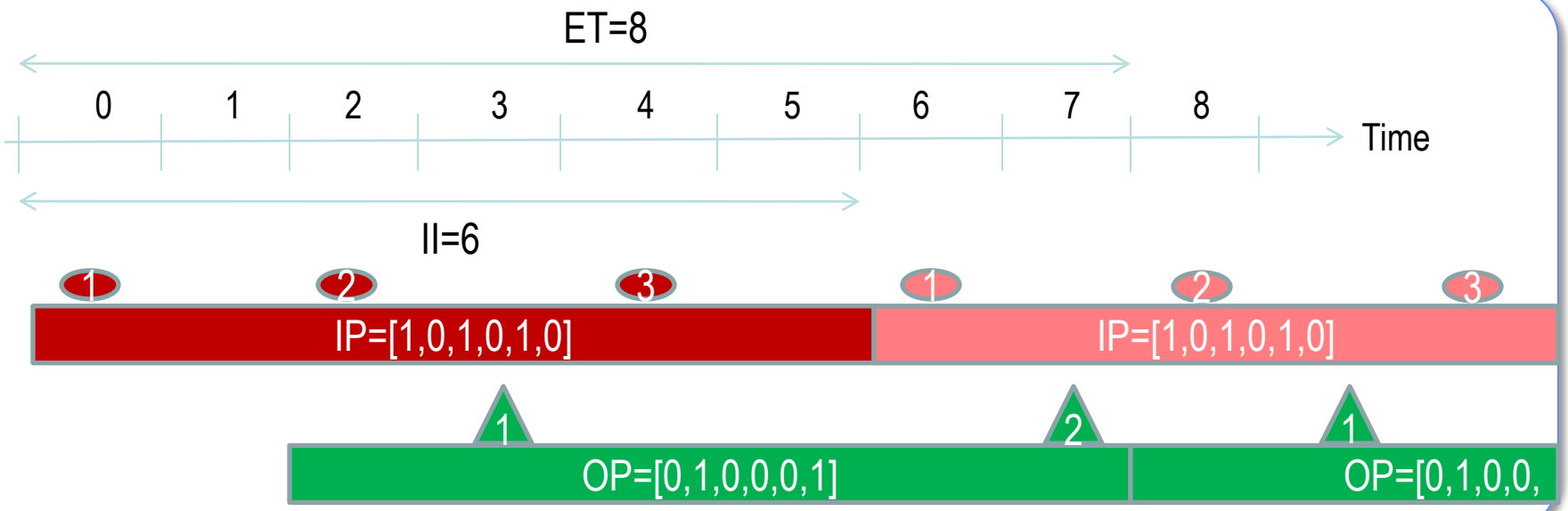
<3,2,6,8,T,[1,0,1,0,1,0],[0,1,0,0,0,1]>

token *I*nput *C*ount = 3
*I*nput access *P*attern = [1,0,1,0,1,0]



token *O*utput *C*ount = 2
*O*utput access *P*attern = [0,1,0,0,0,1]

*E*xecution *T*ime = 8
*I*nitiation *I*nterval = 6
*I*s ET *E*xact = True



Future Research Challenges

- IP exchange mechanisms that include model and protocol descriptions – standardization needed
- High-level Models of Computations to efficient implementations
- Compilation time
- Fast early estimation (timing, performance, area, power, etc.) from high level models
- Multi-level soft-cores and virtual fabrics
- Dynamic partial reconfiguration
- HW/SW operating systems
- Standard floating/fixed point representation and automatic conversion

Thank You