Graphical System Design for Heterogeneous Platforms

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™
Moore's Law for Instrumentation

History of SW Continuity & Expanding Value

1980
1990
2000
2010

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

LabVIEW
LabVIEW Real-time
LabVIEW FPGA & RIO
LabWindows™/CVI

Instrument Control Interfaces
The NI Approach – Integrated Platforms

- High-Speed Digitizers
- High-Resolution Digitizers and DMMs
- Multifunction Data Acquisition
- Digital I/O
- Instrument Control
- Dynamic Signal Acquisition
- Counter/Timers
- Machine Vision
- Motion Control
- Distributed I/O and Embedded Control

High-Speed Digitizers

- Keypad
- LCD
- Sound
- Battery
- RF Signal
- Temperature Monitoring
- Process Control
- Motor and Valve Control
- Waste Monitoring

- Acoustics
- Body & Chassis
- Safety
- Engine
- Emissions
- Electronics
- Tire & Brake
- Durability
- Audio
The National Instruments Vision Evolved…

Graphical System Design

Virtual Instrumentation
- Instrumentation
- RF
- Digital
- Distributed

Industrial Embedded
- Industrial Control (PAC)
- Machine Control
- Electronic Devices
- Code Generation

Real-time Measurements
Embedded Monitoring
Hardware-in-the-loop

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.”

ni.com
Graphical System Design
Empowering Users Through Software

LEGO®
MINDSTORMS® NXT
“the smartest, coolest toy of the year”

LabVIEW™
Graphical System Design Platform

CERN Large Hadron Collider
“the most powerful instrument on earth”

ni.com
LabVIEW Targets

• Scalable from distributed network to sensors
High Speed & High Precision Control with LabVIEW Real-Time & FPGA

Scanning Probe Microscope with PLL

Ultrastable Atomic Force Microscope

Nanoimprint Lithography (Tsao)

Precision Servo-Hydraulic Control
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...

System Flexibility and Price

Number of Systems Deployed

PXI RIO
PCI RIO
Compact RIO

Windows
Real-time Processor
FPGA

I/O
I/O
I/O
Custom I/O

Compact RIO Integrated
Single-Board RIO
Zynq targets

LabVIEW

ni.com
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

- High-speed data transfer
- Tight timing & synchronization of I/O
Future uP and FPGA in one Chip

XILINX® Zynq Extended Processing Platform
HPC meets tough Real-Time Challenges

Solving the most sophisticated control applications

Visualization (User Interface)

ni.com
The Y-Chart System Design Methodology

Application Logic → Analysis & Mapping → Performance Evaluation → Platform Architecture


High-Level Design Models

Dataflow

C Code

Textual Math

1. A = [1 3; 4 2];
2. B = [6 7; 2 3];
3. C = A*B;
4. eigC = eig(C);
5. D = k*A

Simulation

Statechart

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

Block Diagram Window

Boolean Control

Graph Indicator

Input Terminals

Output Terminal

ni.com
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

Solution Wizard

Printed Circuit Board Inspection

Problem Description
Printed Circuit Board Inspection
Determine whether PCB components are present and in the correct location using pattern matching. Match areas of inspection to a template of valid components. Use shift-invariant matching to inspect an oriented
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.

Click the Add Item button to add timing engines and functions.
Click the Add Item button to add timing engines and functions.

Timing Engines tree
Displays the timing engines and functions that you add in a tree. The timing engines contain the functions you add.

PWM (Connector0/DIO1)
- Resource: Connector0/DIO1
- Polarity: Active High

NATIONAL INSTRUMENTS™

ni.com
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

- Nanoseconds: Backplane timing, IO synchronized with a global clock
- Microseconds: Software programmed FPGAs, Timed loops, Software constructs: FIFOs | Queues
- Milliseconds: Software structured dataflow

Time Scale vs. Flexibility
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

ni.com
The Challenge Going Forward

Application Trends
• 1000’s of parallel tasks
• Large node/channel counts
• High performance requirements
• E.g. streaming DSP applications

Platform Trends
• 100’s of processing elements
• Heterogeneous processors and memories
• Distributed I/O
• E.g. FPGA targets

How to map the tasks and data in a concurrent application to the processing and memory resources on a platform?
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
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 tradeoff without diving into implementation details
  – Tune performance with high-level constraints
  – Access the details if needed
MoCs for Streaming Applications

Expressive vs. Analyzable

Kahn Process Networks
Integer Dataflow
Boolean Dataflow
SHIM
Heterochronous Dataflow
Static Dataflow
Parameterized Dataflow
Cyclo-static Dataflow
Homogeneous Dataflow

Deterministic?
No
Yes

Synchronous?
No
Yes

Deadlock and boundedness decidable?
No
Yes

Static scheduling?
No
Yes

Key trade-off: Analyzability vs. Expressibility

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

**Abstraction**

- **Hard IPs**
  - PowerPC, DSP, ...
- **RTL/Pin Level IPs**
  - All HDL IPs, ...
  - *IP-XACT (IEEE 1685)*
  - AMBA, AXI
- **Transaction Level IPs**
  - SystemC/TLM
- **Domain Specific Abstract IPs**
  - NI actors, Xilinx actors

**Providers**

- **Xilinx**
- **Altera**
- **Almost everyone**
- **Mostly verification IP providers**
- **System solution providers**
  - NI, Xilinx

ni.com

**Domain Specific Abstract IPs**

- NI DSP Designer

---

**NI Logo**
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

- Protocol details
- Cycle accurate behavior
- Optimized Code Gen

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

Data Accessor

Din(4X1)

Read

Data

sum

Dout

ni.com
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 Input Count = 3
Input access Pattern = \([1,0,1,0,1,0]\)

\(3 \to \text{in} \quad \text{out} \quad 2\)

token Output Count = 2
Output access Pattern = \([0,1,0,0,0,1]\)

- Execution Time = 8
- Initiation Interval = 6
- Is ET Exact = True

\(\text{ET}=8\)

\(\text{II}=6\)

- IP=\([1,0,1,0,1,0]\)
- OP=\([0,1,0,0,0,1]\)
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