LabVIEW User Group Meeting

Long Island Chapter
2 September 2010
Agenda

• Introductions
• Upcoming Events
• What’s New in LV 2010
Robert Berger

- BS Electrical Engineering from Texas A&M
- Joined NI’s Engineering Leadership Program in 2001
- Supported and trained customers for ~4 years
- Migrated to Long Island in April 2007
- Covers Long Island and NYC
- Available for demos, onsite seminars, technical consultation, specification assistance, loaner equipment…
Upcoming Events

• **Local seminars**
  - On-sites at any time
  - LV for Embedded Applications Hands-On September 8
  - User Group, 1\textsuperscript{st} Thursday of Dec.

• **Training**
  - LabVIEW Core 3, Sept 13, Melville, NY
  - LabVIEW Software Engineering, Sept 16, Melville, NY
  - On-line Weekly

• **NI-Week**
  - Conference, August 2011

• **NI Certification**
  - Available at local testing sites
What’s New in LabVIEW 2010

Robert Berger
Sr. District Sales Manager, National Instruments
What’s New in LabVIEW 2010?

• Improved Performance
• LabVIEW Environment Enhancements
• Large Application Development
• Improved Data Transfer and Distributed HW Configuration
• What’s New in the LabVIEW Modules
• Building LabVIEW Add-ons
New Features in LabVIEW 2010

- **Compiler Optimizations**
  - Run your code up to 20 percent faster with new back-end compiler technologies and custom code optimizations.

- **Feedback-Driven Improvements**
  - Facilitate development with 14 new features based on user feedback.

- **Stream Data over the Network**
  - Stream data continuously between LabVIEW applications using the new Network Streams API.

- **Save VIs without Compiled Code**
  - Simplify source code management by separating the compiled objects from the actual LabVIEW source code.

- **SubVI Inlining**
  - Improve your application’s performance by removing the overhead associated with subVI calls using this new execution option.

- **Export Graph Data to Excel**
  - Easily export your data to Microsoft Excel with a simple right-click option.

- **Web-Based Hardware Configuration**
  - Set up and maintain your remote hardware with a router-like configuration experience.

- **Import External FPGA IP**
  - Ease FPGA development by importing Xilinx Core Generator DSP IP into LabVIEW.

- **Instrument Drive**
  - Shorten the time to first measurement by easily finding and installing more than 10,000 certified instrument drivers.

- **Packed Project Libraries**
  - Simplify code deployment and distribution by packaging your source code into a single file with packed project libraries.

ni.com
IMPROVED PERFORMANCE
LabVIEW Compiler

Abstracts the complexities of programming

- Memory management
- Thread allocation
- Language syntax
Optimizing the LabVIEW Compiler

DataFlow Intermediate Representation (DFIR)
- High-level representation
- Preserves dataflow, parallelism, and execution semantics

Low-Level Virtual Machine (LLVM)
- Low-level representation
- Sequential
- Knowledge of target machine characteristics, instruction sets, alignment, etc.
LabVIEW 2010 Performance Metrics

Run-Time Performance Improvement in LabVIEW 2010

% Improvement LabVIEW 2010 vs. LabVIEW 2009

0%  50%  100%  150%  200%

Complex Math - Black-Scholes PDE solver
DAQ - Digital and Analog Waveform Reads
Real-time Math (PXI-8196) - MathScript Heat Equation
Bit Manipulation - UnpackBits
Real-time Control (cRIO 9022) - Advanced PID
Real-time Control (cRIO 9022) - Single Channel PID
Parallel For Loop - Mandelbrot
Large Array Math - Linear Scale (Multiply and Add)

480%
LabVIEW Compiler Decompositions

Common Subexpression Elimination

Unreachable Code Elimination
LabVIEW Compiler Optimizes Your Code

ni.com
LabVIEW Compiler Optimizes Your Code

Only this portion of the code will execute

Because the input is constant, the compiler can determine which code will execute, and remove the unnecessary code
LabVIEW Compiler Optimizes Your Code

Sequence Structure is now unnecessary

The Match Pattern primitive will not change from iteration to iteration
LabVIEW Compiler Optimizes Your Code

Decompositions Used

- SubVI Inlining
- Unreachable Code Elimination
- Dead Code Elimination
- Loop Invariant Code Motion
SubVI Inlining
Maintain Code Modularity With Minimum Overhead

- Removes all subVI call overhead
- Automatically updates callers when callee’s code changes
LabVIEW 2010 Performance Metrics

What Won’t Get Faster?

- Hardware Calls
- DLL Calls / Optimized C Code
- User Interface Interaction

Increased Compile Time

- Compile Time: 5x
- Mass Compile Time: 35%
- Application Build Time: 35%

ni.com
Smarter Installer
Select Your Software Based on Serial Number(s)

Enter your Serial Number(s), and the installer will determine what to install.
The current boolean diagram constant is potentially confusing and too elaborate.

**Confusing**, because it almost looks like a toggle switch, so the new user might click on the right half, expecting an unconditional FALSE. However, there are no active areas, and an inversion of the current value occurs no matter where we click.

**Too elaborate.** All we need to see is the current value! Why do we need to see the "other" value greyed out?? We can guess that by simple elimination. 😞 There is too much redundant information, wasting twice as much diagram space than actually needed to display relevant information. The current design also makes e.g. 2D boolean diagram constant very confusing. Have a look at the image. Can you immediately tell that the 2D array on the left is only true on the diagonal? (I did not think so!). Now look at the suggestion on the right. Ahh... much better! 😊

**Suggestion:**
The boolean diagram constant should be **smaller, simpler, and cleaner**.
The image shows the current design on the left and the suggested design on the right.

What a difference in clarity and economy!!
<table>
<thead>
<tr>
<th>Feature Name</th>
<th>LabVIEW 2009</th>
<th>LabVIEW 2010</th>
<th>User</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Number of Undo Steps</td>
<td>Maximum undo steps per VI 8</td>
<td>Maximum undo steps per VI 99</td>
<td>PJM_LabVIEW</td>
</tr>
<tr>
<td>Local Variable Redesign</td>
<td><img src="image1.png" alt="Diagram" /></td>
<td><img src="image2.png" alt="Diagram" /></td>
<td>Altenbach</td>
</tr>
<tr>
<td>String Radix</td>
<td><img src="image3.png" alt="Diagram" /></td>
<td><img src="image4.png" alt="Diagram" /></td>
<td>Altenbach</td>
</tr>
<tr>
<td>Wire Labels</td>
<td><img src="image5.png" alt="Diagram" /></td>
<td><img src="image6.png" alt="Diagram" /></td>
<td>Falkpl</td>
</tr>
<tr>
<td>Growable Merge Error Node</td>
<td><img src="image7.png" alt="Diagram" /></td>
<td><img src="image8.png" alt="Diagram" /></td>
<td>Dany.</td>
</tr>
<tr>
<td>Move Switch Items in the connector pane</td>
<td><img src="image9.png" alt="Diagram" /> 8 Mouse Clicks</td>
<td><img src="image10.png" alt="Diagram" /> 2 Mouse Clicks</td>
<td>tst</td>
</tr>
</tbody>
</table>
DEMO

USER REQUESTED FEATURES
HTTP(S) Nodes

- Use the new **HTTP Client VIs** to build a LabVIEW web client
- Interact with servers, Web pages, and Web services
- Works with LabVIEW or 3rd-party Web services
LARGE APPLICATION DEVELOPMENT
Separate Compiled Code From Source File

Improved Source Code Control

Eliminate the need to re-save and re-submit files to source code control unless the graphical source code has been changed by the developer.
Packed Project Libraries
Distribute and Reuse LabVIEW Code Easily

- Deploy the VI hierarchy with a single file
- Shorten build times for calling VIs
- Simplify code deployment
- .lvlibp file type

<table>
<thead>
<tr>
<th>Example</th>
<th># Source VIs</th>
<th>EXE Build Time</th>
<th># VIs Built Into PPL</th>
<th>EXE Build Time</th>
<th>Build Time Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agilent 34401 Acquire and Graph - SW Triggered.vi</td>
<td>53</td>
<td>6.3 s</td>
<td>22</td>
<td>5.15 s</td>
<td>18.2%</td>
</tr>
<tr>
<td>E-Mail Notification.vi</td>
<td>102</td>
<td>8.66 s</td>
<td>68</td>
<td>5.82 s</td>
<td>32.8%</td>
</tr>
<tr>
<td>Update Weather Data.vi</td>
<td>71</td>
<td>12.97 s</td>
<td>46</td>
<td>5.48 s</td>
<td>57.8%</td>
</tr>
<tr>
<td>Custom Example</td>
<td>1000</td>
<td>53.93 s</td>
<td>999</td>
<td>15.94 s</td>
<td>70.4%</td>
</tr>
</tbody>
</table>
LabVIEW Add-Ons for Software Validation

Unit Test Framework Toolkit
- 30% faster test execution
- Custom definition of test vector ranges

VI Analyzer Toolkit
- Create your own tests in VI Analyzer using LabVIEW Scripting
TARGET-TO-HOST DATA TRANSFER
Network Connectivity Options in LabVIEW

TCP/IP and UDP
Define low-level communication protocols

Remote Front Panels
Quickly embed a front panel in a browser

Shared Variables
Quickly develop distributed systems through drag-and-drop configuration

ni.com
DEMO

TARGET-TO-HOST DATA TRANSFER
LabVIEW 2010 Real-Time Products

**LabVIEW Real-Time Module**
- Web-based configuration and monitoring of networked targets
- Simpler host-to-target transfer of data using Network Streams
- Publish variables via Web Services
- Software IEEE 1588 as timing source for Timed Loop

**NI-Real-Time Hypervisor 2.0**
- Shared memory for higher data transfer rates between OSs
- Higher customization for CPU partitioning
- Added Linux support
LabVIEW 2010 FPGA Module

IP Integration Node - Directly import Xilinx .xco files or your own VHDL easily

New Compilation Flow - Earlier Compilation Estimates and Build Specifications

Cycle-Accurate Simulation - Use ModelSim for Cycle-Accurate Simulation

More IP Blocks - New IP for Statistics, Complex Multiplication, and More

Host Improvements – New Dynamic reference for Host VI reuse
LabVIEW 2010 FPGA Compilation

LabVIEW FPGA Compile Farm Toolkit

Compilation "Smart" Server

LabVIEW FPGA Development Machines

Compilation Workers
LabVIEW 2010 FPGA Compilation

LabVIEW FPGA Compile Cloud Service (Beta)

rick.kuhlman@ni.com

High-RAM Dedicated Workhorse Computers in the Cloud

LabVIEW FPGA Development Machines
LabVIEW 2010 MathScript RT Module

MathScript Node
• Validate your custom .m files for deterministic behavior
• Automatically create output variables

MathScript Window
• Performance improvements
• Enhanced text-editor
  ▪ Syntax highlighting
  ▪ Line numbers
  ▪ Find/replace text dialog box
  ▪ Bookmarks

```plaintext
1 % Program P2_1
2 % Simulation of an M-point Moving Average Filter
3 % Generate the input signal
4 n = 0:100;
5 s1 = cos(2*pi*0.05*n); % A low-frequency sinusoid
6 s2 = cos(2*pi*0.47*n); % A high frequency sinusoid
7 x = s1+s2;
8 % Implementation of the moving average filter
9 M = input('Desired length of the filter = ');
10 num = ones(1,M);
11 y = filter(num,1,x)/M;
```
LabVIEW 2010 Robotics Module

- **IP** for sensing, navigation, motion control 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
BUILDING LABVIEW ADD-ONS
LabVIEW 2010
Extending the Platform

Licensing and Activation for 3\textsuperscript{rd} Party Add-ons

- Commercial Grade Activation solution from Concept Software
- Implement 30-day software trials for LabVIEW Add-ons you create
- Fully integrated in LabVIEW 2010

LabVIEW Scripting

- Intended for power users to enhance the capabilities of LabVIEW during editing
- Used to inspect, modify, or generate LabVIEW code automatically
LabVIEW 2010 Resources

- LabVIEW 2010 New Features
- LabVIEW 2010 Performance Update
- LabVIEW Compiler: Under the Hood
- Timing and Synchronization in LabVIEW