

LabVIEW™ FPGA Module Release and Upgrade Notes

Version 2009

These release notes contain instructions for installing the LabVIEW FPGA Module, introduce new features, and provide upgrade information. Refer to the resources listed at the end of this document for information about developing applications with LabVIEW and the FPGA Module.

Contents

System Requirements.....	2
Installing the LabVIEW FPGA Module	3
Activating the FPGA Module	4
LabVIEW 2009 FPGA Module Features.....	4
Timing Violation Analysis Window	4
Compilation Process Improvements	5
Math and Analysis VIs and Functions Improvements.....	5
FIFO and Memory Improvements	6
Support for the Enhanced Feedback Node	7
Support for User-Defined I/O Variables.....	7
CLIP Improvements.....	7
Upgrade and Compatibility Issues	8
Upgrading from the FPGA Module 8.6.x to 2009	8
Upgrading from the FPGA Module 8.5.x to 8.6.x	11
Upgrading from the FPGA Module 8.2.x to 8.5.x	13
Upgrading from the FPGA Module 8.0.x to 8.2.x	14
Where to Go from Here	15
Related Documentation and Examples	16
NI Web Site	16
Known Issues	16

System Requirements

The development computer is a PC or PXI system on which you install the LabVIEW development system and the LabVIEW FPGA Module.

Table 1 describes the requirements the development computer must meet to run the FPGA Module. The FPGA Module system requirements are in addition to the LabVIEW system requirements listed in the *LabVIEW Release Notes*.

Table 1. System Requirements for the FPGA Module

Supported Platforms	Media and System Requirements	Important Notes
<ul style="list-style-type: none"> • Windows XP Pro (Service Pack 1, 2 or 3) • Windows Vista (32-bit) • Windows Vista (64-bit) running LabVIEW 32-bit only* • Windows 2000 Pro (Service Pack 2, 3, or 4) † 	<ul style="list-style-type: none"> • 1.2 GHz Pentium processor or a compatible processor of equal or higher speed • 2 GB memory‡ • At least 4 GB additional available disk space • LabVIEW 2009 Full or Professional Development System • FPGA target hardware and driver software, such as an NI Reconfigurable I/O device and the NI-RIO 3.2 software version. 	<p>* Windows Vista 64-bit—The LabVIEW FPGA Module uses the 32-bit version of the Xilinx tools, even on the Vista 64-bit OS.</p> <p>† Windows 2000—The FPGA Module uses Xilinx tools that do not officially support Windows 2000. National Instruments obtained permission from Xilinx to allow FPGA Module customers to use the tools on this platform, with the disclaimer that Xilinx will not be able to fix any bugs found that are specific to this platform. National Instruments tested the Xilinx tools that the FPGA Module uses, and did not find any issues. If you encounter problems with the Xilinx tools and Windows 2000, you might be required to compile using Windows XP or Vista 32-bit. In such cases, you might want to install the LabVIEW FPGA Compile Server on a remote computer. Refer to Table 2 for requirements to install the Compile Server on a remote computer. National Instruments will not be liable for any problems or issues related to the use of Xilinx tools with Windows 2000.</p> <p>‡ Memory Requirements—Memory requirements vary by FPGA target. Also, although these memory requirements apply to typical FPGA VIs, the unique characteristics of each FPGA design can affect whether LabVIEW can compile the design using the available memory. You can monitor the memory usage of the <code>xst.exe</code> process in the Windows Task Manager to determine if you need to install additional memory.</p>

If the FPGA design or target requires more than 2 GB of memory, National Instruments recommends that you install the LabVIEW FPGA Compile Server on a remote computer. Refer to the *Compiling an FPGA VI Remotely* topic in the *LabVIEW Help*, available by selecting **Help»Search the LabVIEW Help**, for more information about running the Compile Server on a remote computer. Table 2 describes the recommended operating systems for installing the LabVIEW FPGA Compile Server on a remote computer.

Table 2. Recommended Systems for Installing the Compile Server on a Remote Computer

Supported Platform	Important Notes
<ul style="list-style-type: none">Windows XP Pro (Service Pack 1, 2, or 3)Windows Vista (32-bit)Windows Vista (64-bit)	Memory Requirements —On a remote computer with Windows Vista 64-bit, the Compile Server can take advantage of up to 4 GB of memory.

Installing the LabVIEW FPGA Module

This section includes information about installing the FPGA Module on a development computer.



Note Refer to the *LabVIEW Release Notes* for information about installing the LabVIEW development system.

Complete the following steps to install LabVIEW and the FPGA Module.

1. Log in to the development computer as an administrator or as a user with administrative privileges.
2. Insert the LabVIEW Platform DVD and select to install LabVIEW.



Tip You also can double-click `setup.exe` from the media to launch the installer.

3. Follow the instructions on the screen for installing software.
 - a. Select the following software from the product list tree:

- **LabVIEW**
- **FPGA Module** in the **Embedded (Real-Time, FPGA, and Microprocessor)** folder
- (Optional) **Real-Time Module** in the **Embedded (Real-Time, FPGA, and Microprocessor)** folder
- **Device Drivers**

NI-RIO is device driver software. NI-RIO includes driver software for R Series, FlexRIO, and CompactRIO devices. You might need to install different or additional driver software for the FPGA target you use.



Note You must install the NI-RIO 3.2 drivers to use NI-RIO with the FPGA Module 2009. Earlier versions of the NI-RIO drivers do not support the FPGA Module 2009.

- b. Continue to follow the instructions on the screen to complete installation of the software. The installer might prompt you to insert another DVD to continue installation.

The LabVIEW Platform DVD installs program files and documentation and copies files from Xilinx ISE to the `x:\NI\FPGA2009` directory, where `x` is the drive on which you installed LabVIEW 2009. Xilinx ISE is third-party software that the FPGA Module uses to compile FPGA VIs into code that runs on the FPGA target

4. (Optional) If the FPGA target does not use NI-RIO drivers, install the appropriate drivers and FPGA Module support files for the FPGA target you will use. Refer to the specific hardware documentation for information about the appropriate drivers and for information about installing and configuring the FPGA target.

Activating the FPGA Module

The FPGA Module relies on licensing activation. You have a temporary license for a 30-day evaluation period that includes both the FPGA Module and the Xilinx tools that the FPGA Module uses. When the evaluation period expires, you must activate a valid FPGA Module license to continue using the FPGA Module. Activating the FPGA Module license also activates the license for the Xilinx tools.

You can use the NI License Manager, available by selecting **Start»All Programs»National Instruments»NI License Manager**, to activate National Instruments products. Refer to the *National Instruments License Manager Help*, available by selecting **Help»Contents** in the NI License Manager, for information about activating NI products.

LabVIEW 2009 FPGA Module Features

The FPGA Module 2009 includes the following new features to help you better manage and implement the components of an FPGA application. Refer to the *LabVIEW Help*, available by selecting **Help»Search the LabVIEW Help** in LabVIEW, for more information about these features.

Timing Violation Analysis Window

You can use the new **Timing Violation Analysis** window to identify components in the FPGA VI that cannot execute within a specified clock rate.

If the **Compilation Status** window reports timing violation errors, you can click the **Timing Violation Analysis** button to display the

Timing Violation Analysis window. The window lists the propagation delay of the FPGA VI components that cause the timing violation. You can double-click an item in the list to locate the node on the block diagram.

Compilation Process Improvements

Configuring Xilinx Compilation Options

If the target supports the new **Xilinx Options** page in the **FPGA Target Properties** dialog box, you can adjust options Xilinx uses to compile the FPGA VI. In general, you do not need to adjust the options on this page unless the FPGA VI fails to compile. Use the information from the **Compilation Status** window to determine which options on this page might help the FPGA VI compile successfully.

Improved Feedback when Compiling FPGA VIs

With the new **Compilation Status** window, LabVIEW now provides information throughout the compilation process that can help you monitor the compilation status and more quickly determine whether the FPGA VI will meet the resource and timing constraints of the FPGA.

The progress bar in the **Compilation Status** window indicates the current step in the Xilinx compilation process. You can watch the progress bar as you continue working in LabVIEW. In addition, you can disconnect from the compilation server through the **Compilation Status** window and queue up multiple VIs to compile.

Math and Analysis VIs and Functions Improvements

High Throughput Math Functions

Use the High Throughput Math functions to perform high-throughput math and analysis with fixed-point numbers on FPGA targets. These functions are similar to the LabVIEW Numeric functions but support higher throughput rates, handshaking terminals inside a single-cycle Timed Loop, internal input/output registers, and automatic pipelining. You also can use these functions outside a single-cycle Timed Loop.

The following is a list of new functions:

- High Throughput Add
- High Throughput Divide
- High Throughput Exponential
- High Throughput Hyperbolic Sine & Cosine
- High Throughput Inverse Tangent
- High Throughput Multiply

- High Throughput Natural Logarithm
- High Throughput Polar to Rectangular
- High Throughput Reciprocal
- High Throughput Rectangular to Polar
- High Throughput Sine & Cosine
- High Throughput Square Root
- High Throughput Subtract
- High Throughput To Fixed-Point

Scaling VIs

Use the Scaling VIs in host VIs to convert the clock and sample rate for the Loop Timer Express VI and to reconfigure input settings and post-process data from the FPGA Math & Analysis VIs. The Scaling VIs are located on the **FPGA Interface** palette for host VIs under an RT or My Computer target. The following is a list of new VIs:

- Butterworth Coefficients
- FFT to Spectrum
- Normalize Signal Generation Parameters
- Notch Coefficients
- Sample Rate to Loop Time
- Scale Period

FIFO and Memory Improvements

FIFO and Memory Name Controls

You can use the FIFO and Memory name controls to pass FIFO and memory references to and from VIs, just like you use the FPGA I/O name control and the FPGA Clock name control. You also can use FIFO and Memory constants to reference FIFO and memory items.

New FIFO Methods

The FPGA FIFOs support the following FIFO methods. Refer to the *LabVIEW Help* for more information about these methods.

- Number of Elements to Read
- Number of Elements to Write

Implement Memory Using LUTs

You can use look-up tables (LUTs) as well as embedded block memory to implement memory items in an FPGA VI. Using LUTs for memory items, especially memory items with a small number of elements, reserves embedded block memory for other functions.

Support for the Enhanced Feedback Node

FPGA VIs support the multi-cycle delays, enable signals, and initialization options of the enhanced Feedback Node. You also can give the Xilinx compiler the option to implement this node by using shift register lookup tables (SRLs) instead of flip-flops. This change improves resource usage on FPGA targets.

Support for User-Defined I/O Variables

If you execute an FPGA VI on an FPGA target that supports the NI Scan Engine, you can create user-defined I/O variables to send I/O data between that the FPGA VI and an RT VI running on a single controller.

Refer to the **Fundamentals»Accessing Scanned I/O Data»Concepts** book on the **Contents** tab in the *LabVIEW Help* for general information about the NI Scan Engine and I/O variables. Refer to the *Transferring Data Between the FPGA and the Host VI* topic on the **Contents** tab in the *LabVIEW Help* for information about using I/O variables in FPGA VIs.

CLIP Improvements

Specifying Clock Domains for CLIP I/O

Use the following XML tags in a CLIP declaration file to avoid metastable data in CLIP I/O.

- `RequiredClockDomain`—If you require a clock domain, any I/O Node on the block diagram that uses this CLIP I/O must be within the clock domain you specify.
- `UseInLabVIEWSingleCycleTimedLoop`—If you require that the IP in the CLIP get or receive values on every clock edge, setting this tag makes LabVIEW enforce this requirement.

Clocks Defined in CLIP

You can instantiate clock circuitry in a CLIP and use CLIP clocks in the same way you use other clocks that the target provides. You also can route external clocks supplied by socketed CLIPs to LabVIEW. Clocks in the CLIP declaration file appear automatically under the CLIP item in the **Project Explorer** window.

LabVIEW treats CLIP clocks like external clocks, except that you cannot use the Start Enabling FPGA Clock and Start Disabling FPGA Clock VIs with CLIP clocks to protect circuitry when the clock might not be available.

Additional Data Types for CLIP

CLIP supports the I64 and U64 data types.

Upgrade and Compatibility Issues

The following sections provide information about upgrade and compatibility issues specific to different versions of the FPGA Module.

Upgrading from the FPGA Module 8.6.x to 2009

You might encounter the following compatibility issues when you upgrade from the FPGA Module 8.6.x to the FPGA Module 2009.

Dialog Boxes Replaced by the Compilation Status Window

The new **Compilation Status** window replaces the **Compiling VI for FPGA**, **Compilation Failure**, and **Successful Compile Report** dialog boxes.

Changes to VI-Scoped FIFOs and Memory

Because of the new Memory control and FIFO control, you can pass memory and FIFO references to and from subVIs, just like you use the FPGA I/O control and FPGA Clock control. You also can use the new Memory constant and FIFO constant in the same way you use the FPGA I/O constant and FPGA Clock constant.

To accommodate this new functionality, the VI-Scoped FIFO Configuration node and the VI-Scoped Memory Configuration node are now the VI-Defined FIFO Configuration node and VI-Defined Memory Configuration node, respectively. The difference between the current and previous version of these nodes is the addition of an output terminal that you can use to do the following tasks.

- Wire to subVIs
- Create indicators that pass information to the connector pane of the VI
- Wire FIFO and Memory Method nodes
- Create FIFO and Memory control constants that correspond to that item

The VI-Scoped FIFO Configuration nodes and VI-Scoped Memory Configuration nodes you created in previous LabVIEW versions migrate automatically when you open them in LabVIEW and the FPGA Module 2009.

Fixed-Point Math Library Changed to High Throughput Math Functions

If you created a VI that uses the High Throughput Math functions (previously known as the Fixed Point Math Library from ni.com/labs) in LabVIEW 8.6.x and earlier and open that VI in LabVIEW 2009, be aware of the following changes in behavior:

- In previous versions of LabVIEW, some functions contain the **Default type** option. In LabVIEW 2009, the name of this option is **Adapt to source**. LabVIEW maintains the setting from previous versions.
- In LabVIEW 8.6.x and earlier, you can place the Add, Subtract, and To Fixed-Point functions either inside or outside a single-cycle Timed Loop without specifying the location. In LabVIEW 2009, you must specify whether you will place all High Throughput Math functions inside or outside a single-cycle Timed Loop.

When you load these functions in LabVIEW 2009, LabVIEW selects the **Inside single-cycle Timed Loop** option. If a function is outside a single-cycle Timed Loop and you try to compile the FPGA VI that contains these functions, LabVIEW returns errors. You must reconfigure the function by double-clicking the function and selecting the appropriate option inside the **Execution Mode** section of the configuration dialog box.

- In LabVIEW 8.6.x and earlier, the functions have three handshaking terminals: **input valid**, **output valid**, and **ready for input**. In LabVIEW 2009, the functions also have a fourth handshaking terminal: **ready for output**. The default value of this terminal is TRUE, which means the behavior of the function does not change between LabVIEW versions even if you do not wire a value to this terminal.
- In LabVIEW 2009, if you place a function inside a single-cycle Timed Loop, you can add one or more sets of internal registers to the function. Adding internal registers reduces the length of the combinatorial path, which can help prevent timing violations when you compile the FPGA VI. However, adding registers also increases the latency of the function.
- The High Throughput Math functions are not supported in previous versions of LabVIEW.

Changes to the Butterworth Filter and Notch Filter Express VIs

In the FPGA Module 2009, the Butterworth Filter and Notch Filter Express VIs always display the **reset** input terminal (previously the **initialize** terminal) and the **scaled coefficients** or **configuration** input terminal. The configuration dialog boxes for these Express VIs no longer have the **Show configuration terminal** and **Show reset terminal** checkboxes. You can leave the **reset**, **scaled coefficients**, and **configuration** input terminals unwired.

LabVIEW changed the implementation of the Butterworth Filter when you configure the filter such that the **Expected sample rate to Cutoff frequency** ratio is greater than 100. To get the same implementation that LabVIEW used in LabVIEW 8.6.x, leave the **scaled coefficients** input terminal unwired. If you wire the **scaled coefficients** input, the filter will produce the same results as the filter in LabVIEW 8.6.x if you placed a checkmark in the **Show configuration terminal** checkbox.

Changed Units of Measurement for Input Terminals of Signal Generation Express VIs

In LabVIEW 8.6.x and earlier, the Sine Wave Generator and Square Wave Generator Express VIs have input terminals with no physical meaning. In LabVIEW 2009, these terminals are in units of periods and ticks of the FPGA clock. Additionally, these terminals use the fixed-point data type.

If you have controls or constants wired to these terminals and load the FPGA VI in LabVIEW 2009, LabVIEW coerces the data types to a fixed-point representation of $\langle +, 32, 0 \rangle$.

Changes to the Bitfile

The extension of the bitfile that LabVIEW 2009 creates is `.lvbitx`. You can still use bitfiles with the `.lvbit` extension in LabVIEW 2009. However, the bitfiles with the `.lvbitx` extension are not compatible with earlier versions of LabVIEW. If you try to open a `.lvbitx` file in an older version of the LabVIEW, you receive an error.

Changes to Output Data Types from Case Structure Tunnels

In LabVIEW 2009, LabVIEW determines the data type from the Case structure output tunnel by using a data type that can handle all cases in the structure, including cases that never execute.

For example, consider a Case structure with two cases, TRUE and FALSE. In the TRUE case, a U8 data type is wired to an output tunnel. In the FALSE case, a U32 data type is wired to the output tunnel. In version 8.6.x, if you wire a constant to select the TRUE case, the data type from the output tunnel is U8 because the constant prevents the FALSE case from ever implementing. In LabVIEW 2009, if you wire a constant to select the TRUE case, the data type from the output tunnel is U32.

Wiring constants to Case structures is an uncommon practice. However, this change in output terminal data type behavior might cause VIs created in LabVIEW 8.6.x to break in LabVIEW 2009 if the output data type is a fixed-point number or fixed-sized array.

Memory Extension Utility Memory Items Do Not Support Execution on Development Computer

If the FPGA VI uses memory items created with the FPGA Module Memory Extension Utility from ni.com, you cannot execute the FPGA VI on execution a development computer. You can replace the extension utility memory items with native FPGA Module memory items.

Upgrading from the FPGA Module 8.5.x to 8.6.x

You might encounter the following compatibility issues when you upgrade from the FPGA Module 8.5.x to the FPGA Module 8.6.x.

Changes to Host VIs

For LabVIEW projects saved in the FPGA Module 8.5.x or earlier, if you selected the **On–Use Random Data for Inputs** or **On–Use Target Hardware for I/O** option from the **Emulator** shortcut menu or **FPGA Target Properties** dialog box, the option had no effect on the FPGA Interface functions. In the FPGA Module 8.6.x, the FPGA Interface functions control the FPGA VI running on the development computer. Use the text underneath the icon for the Open FPGA VI Reference function to determine where the FPGA VI executes.

Configuring Open FPGA VI Reference Functions

The Open FPGA VI Reference function no longer includes the **Select VI**, **Select Bitfile**, **Open and Run**, **Open**, **Bind to Typedef**, and **Select Address** shortcut menu options. Instead, you can configure the Open FPGA VI Reference function using the **Configure Open FPGA VI Reference** dialog box. Right-click the Open FPGA VI Reference function and select **Configure Open FPGA VI Reference** from the shortcut menu

to display the dialog box. To specify the FPGA target on which you want to run the FPGA VI, use the **Resource** field on the **FPGA Target Properties** page or the **resource name** input on the Open FPGA VI Reference function.

Passing an FPGA VI Reference between FPGA Interface Functions

The FPGA Interface functions no longer use a cluster to pass information between functions. Instead, the FPGA Interface functions use an FPGA VI reference. Right-click an FPGA VI reference control, constant, or indicator and select **Configure FPGA VI Reference** from the shortcut menu to configure the reference.

Call VI Function No Longer Supported

The FPGA Module 8.5.x removed the Call VI function from the **FPGA Interface** palette. The FPGA Module 8.6.x replaces the Call VI function in existing applications with a series of functions that maintains the functionality of the Call VI function.

Saturation Arithmetic VIs No Longer Supported

The Saturation Arithmetic VIs are no longer available on the **Saturation Arithmetic** palette. However, existing applications that use the Saturation Arithmetic VIs continue to work.

Change to the Analog Period Measurement VI

If you wired a signed integer to the **hysteresis** input on the Analog Period Measurement VI in an existing FPGA VI and open that VI in LabVIEW and the FPGA Module 8.6.x, LabVIEW changes the input to an unsigned fixed-point terminal and adds a To Fixed-Point function to convert to the unsigned type with saturation. This change provides improved edit-time error checking to prevent inadvertently passing a negative number to the **hysteresis** input. However, this change requires additional resources on the FPGA, and the To Fixed-Point function takes one extra clock cycle to execute. You can restore the original performance by wiring an appropriate unsigned type to the **hysteresis** input and removing the To Fixed-Point function.

Additional FPGA Resources Required

The FPGA Module 8.6.x provides improved reliability of the FPGA during a system restart. As a result, if you recompile an existing FPGA VI using the current FPGA Module, the FPGA VI might require additional resources and might not fit on the FPGA.

Path to Compile Tools

You no longer can use the **Configure Server** dialog box to specify the path to the directory containing the Xilinx compile tools. Instead, the FPGA Module installer defines the path.

Increased Minimum Depth for FIFOs

For DMA FIFOs, the default depth is now 15 elements. For target-scoped FIFOs, the default is now 20 elements.

Upgrading from the FPGA Module 8.2.x to 8.5.x

You might encounter the following compatibility issues when you upgrade from the FPGA Module 8.2.x to the FPGA Module 8.5.x.

Changes to DMA FIFOs

In the FPGA Module 8.2.x and earlier, if you use the Invoke Method function with the *FIFO»Configure* method, the default of the **Depth** parameter is twice the number of elements in the DMA FIFO item in the project. In the FPGA Module 8.5.x, the default of the **Depth** parameter is 10,000 elements.

In the FPGA Module 8.2.x and earlier, if you use the Invoke Method function with the *FIFO»Stop* method, you cannot read data from the host part of the FIFO, but you can read data from the FPGA part of the FIFO. In the FPGA Module 8.5.x, if you use the *FIFO»Stop* method, you cannot read data from the host or FPGA parts of the FIFO. As a result, FPGA VIs that use DMA FIFOs now require additional FPGA resources. So, some existing FPGA VIs might not fit on the FPGA.

Now if an error occurs while the *FIFO»Stop* method executes, it executes normally and sets its own error status in **error out**.

Discrete PID VI

The FPGA Module no longer supports the Discrete PID VI. Use the PID (FPGA) Express VI, available with the LabVIEW PID Control Toolkit, to implement single-channel and multi-channel PID controllers on FPGA targets. The PID Control Toolkit is available with the LabVIEW Real-Time Module. You also can purchase the PID Control Toolkit separately.

Changes to the Scale by Power of 2 Function

The Scale by Power of 2 function now consumes less space on the FPGA. However, the maximum clock rate for the function is now slightly slower for non-constant signed shifts. Therefore, existing applications that include the function might not meet existing timing specifications.

Using the Invoke Method to Force a Download

The Download method on the Invoke Method function no longer includes the **Force Download** parameter. When you call the Download method, the method now always forces an FPGA VI to download to the FPGA target. However, existing applications that use the **Force Download** parameter continue to work, because the FPGA Module replaces the previous version of the Download method with a Case structure. The selector of the Case structure is the value wired to the **Force Download** parameter in the previous version of the Download method. The Case structure executes the new version of the Download method only when the value of the selector is True.

FIFO Read and FIFO Write Parameter Renamed

The **Full** output on the FIFO Write function and the **Empty** output on the FIFO Read function were both renamed to **Timed Out?**.

Upgrading from the FPGA Module 8.0.x to 8.2.x

You might encounter the following compatibility issues when you upgrade from the FPGA Module 8.0.x to the FPGA Module 8.2.x.

Memory Read and Memory Write Functions

The Memory Read or Memory Write functions from LabVIEW 8.0.x or earlier do not automatically update to the current functionality of the Memory Method Node. However, all Memory Read and Memory Write functions from 8.0.x or earlier continue to work in the FPGA Module 8.2.x.

If you created memory blocks using the FPGA Module Memory Extension Utility on ni.com, the FPGA Module 8.2.x does not convert the memory blocks to use a memory item. The memory blocks continue to work. However, the memory blocks might not be supported in a future release of the FPGA Module. To replace a memory block, create a new memory item with the same configuration as the memory block and configure the Read and Write memory methods to access the new memory item.

Wait on Occurrence Function

In the FPGA Module 8.0.x and earlier, when you use the Wait on Occurrence function in an FPGA VI, the function uses ticks as the unit for the **ms timeout** parameter. When you use the Wait on Occurrence function in a host VI, the unit is milliseconds. In the FPGA Module 8.2.x, the Wait on Occurrence function uses milliseconds in both FPGA VIs and VIs running under the **My Computer** target. It also includes a Wait on Occurrence with Timeout in Ticks function.

When you open an FPGA VI from LabVIEW 8.0.x or earlier in the FPGA Module 8.2.x, LabVIEW replaces the Wait on Occurrence function with the new Wait on Occurrence with Timeout in Ticks function.

However, the Wait on Occurrence function does not change in the following cases:

- The VI was last saved on a non-FPGA target in LabVIEW 8.0.x and you open and save (or mass compile) the VI in a non-FPGA target in LabVIEW 8.2.x.
- The VI was last saved in LabVIEW 7.x and you open and save (or mass compile) the VI in a non-FPGA target in LabVIEW 8.2.x.
- The FPGA VI was saved or mass compiled in LabVIEW 8.2.x before installing the latest FPGA Module.

In the cases listed above, you must manually replace the Wait on Occurrence function with the Wait on Occurrence with Timeout in Ticks function to continue using the ticks unit.

Tunnels and Shift Registers on For Loops

In previous versions of the FPGA Module, if the value wired to the count (N) terminal on a For Loop is 0, the outputs from the tunnels and shift registers are undefined. In the FPGA Module 8.2.x, the output tunnels and shift register terminal on the right side of the For Loop contain an extra MUX to handle a 0 value wired to the count terminal in a manner consistent with LabVIEW for Windows. As a result, FPGA VIs that use the output tunnels or right shift register terminal in a For Loop might use more FPGA resources and/or compile at a slightly lower clock rate.

Where to Go from Here

National Instruments provides many resources to help you succeed with your NI products. Use the following resources as you start exploring LabVIEW and the FPGA Module.

Related Documentation and Examples

Use the following resources to learn more about using LabVIEW and the FPGA Module.

- **LabVIEW Help**—Available by selecting **Help»Search the LabVIEW Help** in LabVIEW. Browse the **FPGA Module** book in the **Contents** tab for an overview of the FPGA Module and hardware-specific information. Browse the **FPGA Interface** book in the **Contents** tab for an overview of the FPGA Interface information.
- **Context Help Window**—Available by selecting **Help»Show Context Help**. Context help provides brief descriptions of VIs, functions, and dialog boxes. Context help for most VIs and functions include a link to the complete reference for a VI or function.
- **Hardware-Specific Documentation**—Some FPGA targets provide printed documentation as well as content in the *LabVIEW Help*. Use hardware-specific documentation for information about using the FPGA target with LabVIEW and for information about hardware specifications.
- **Examples**—The driver software for many FPGA targets includes corresponding examples. Refer to the specific hardware documentation for information about whether the FPGA target you use comes with corresponding examples.

You can start with an existing example and use it as a starting point for developing FPGA VIs and host VIs. From LabVIEW, launch the NI Example Finder by selecting **Help»Find Examples**. Browse the examples by directory or by task.

NI Web Site

Visit ni.com/fpga for the latest NI Developer Zone articles, examples, and support information for the FPGA Module.

Refer to ni.com/info and enter the info code `fpgatrn` to access online training for the FPGA Module.

Known Issues

Refer to the National Instruments Web site at ni.com/info and enter the info code `fpgaki` to access the known issues for the FPGA Module.

National Instruments, NI, ni.com, and LabVIEW are trademarks of National Instruments Corporation. Refer to the *Terms of Use* section on ni.com/legal for more information about National Instruments trademarks. Xilinx is the registered trademark of Xilinx, Inc. Other product and company names mentioned herein are trademarks or trade names of their respective companies. For patents covering National Instruments products/technology, refer to the appropriate location: **Help»Patents** in your software, the `patents.txt` file on your media, or the *National Instruments Patent Notice* at ni.com/patents.