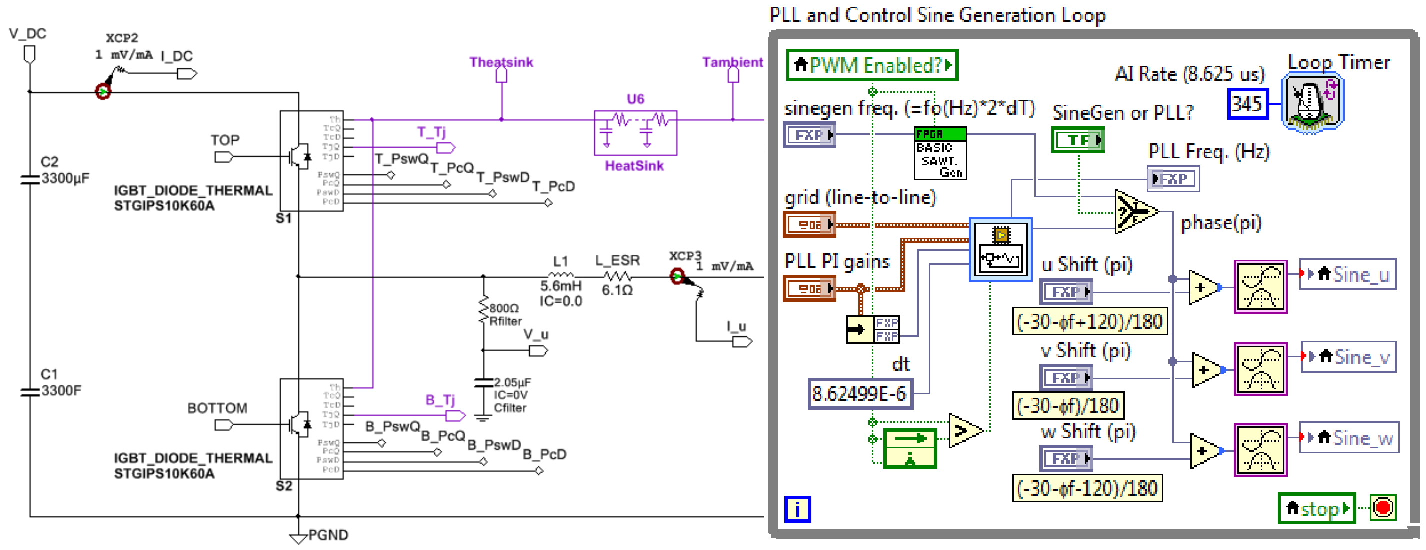
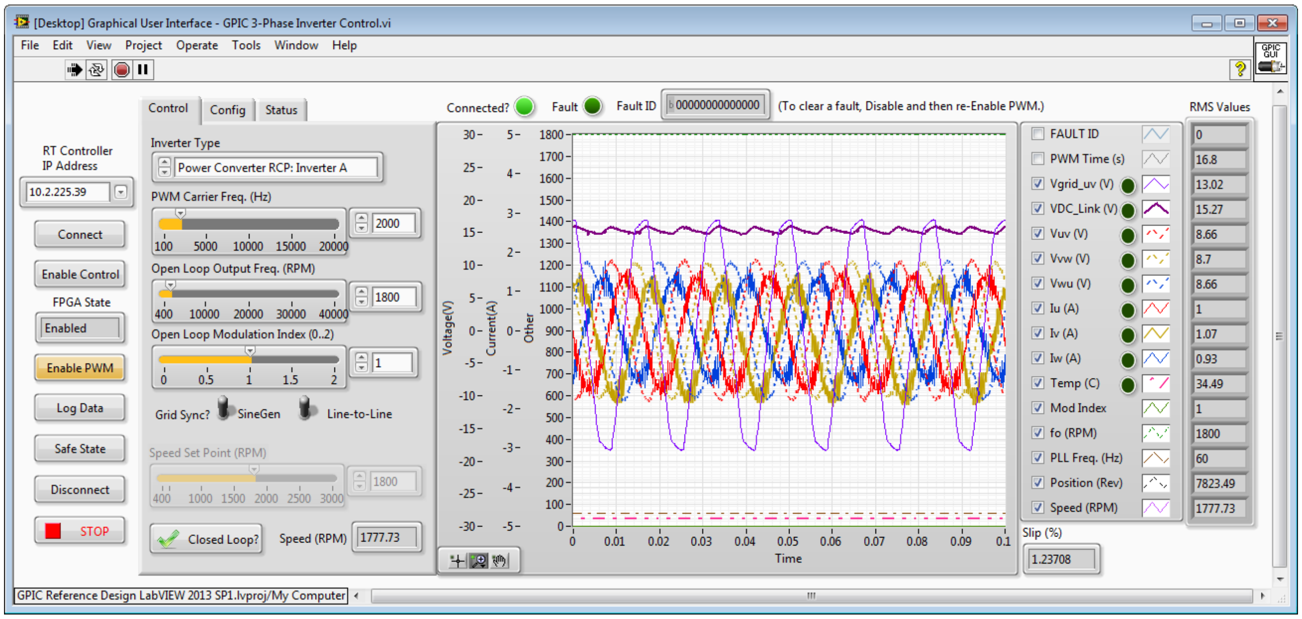




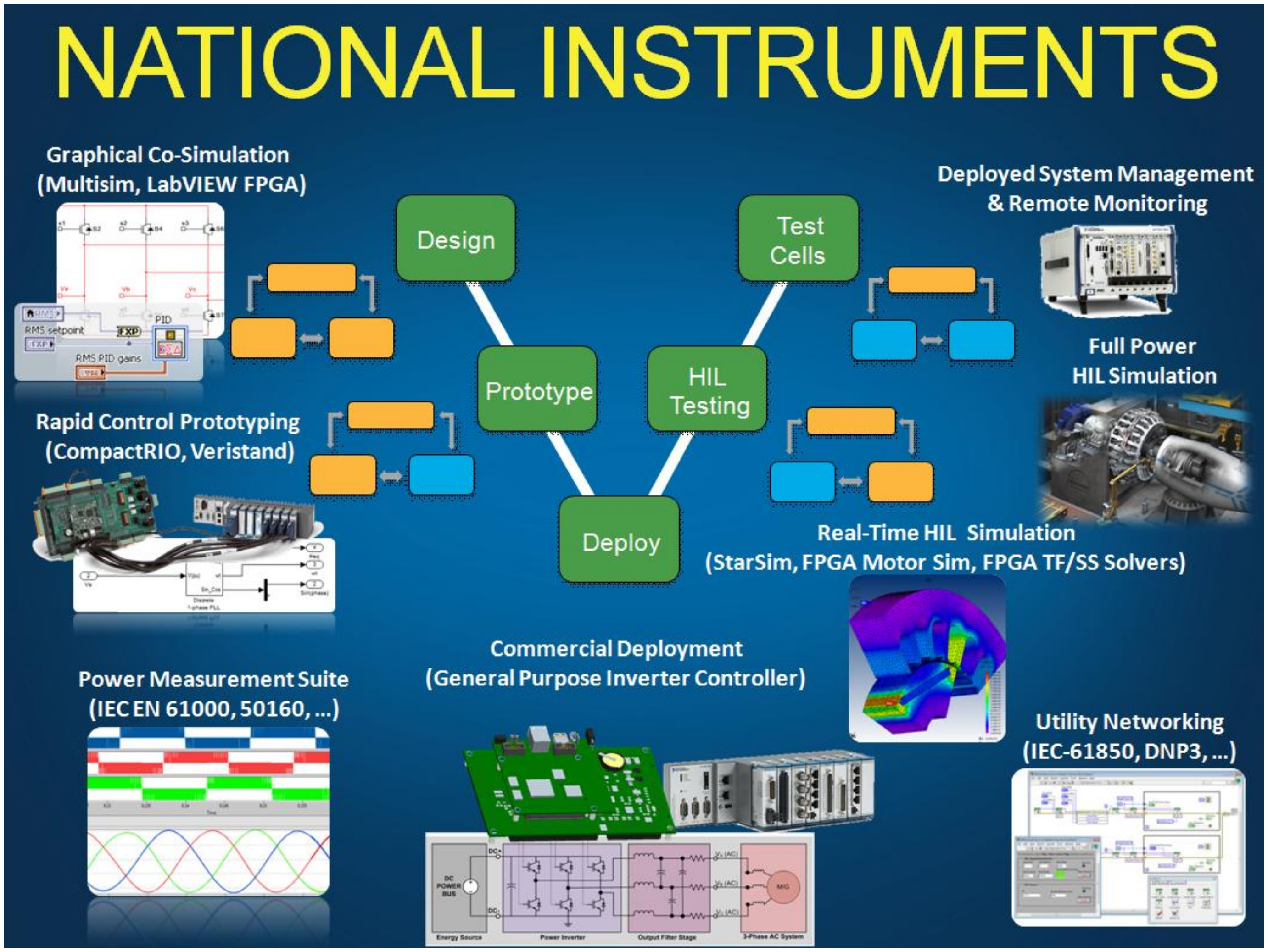
NI Power Electronics Control Design V Hands-On Workshop

Table of Contents

Introduction

Purpose

The purpose of this workshop is to introduce fundamentals of power electronics control design and verification, introduce the NI design V Toolchain using LabVIEW FPGA and Multisim co-simulation tools and the NI General Purpose Inverter Controller (GPIC) reference design code. Your questions, comments and feedback are invited. Please join the developer community at ni.com/powerdev.


NATIONAL INSTRUMENTS
Graphical Co-Simulation (Multisim, LabVIEW FPGA)
Design
Test Cells
Deployed System Management & Remote Monitoring
Full Power HIL Simulation
Prototype
HIL Testing
Rapid Control Prototyping (CompactRIO, Veristand)
Deploy
Real-Time HIL Simulation (StarSim, FPGA Motor Sim, FPGA TF/SS Solvers)
Commercial Deployment (General Purpose Inverter Controller)
Power Measurement Suite (IEC EN 61000, 50160, ...)
Utility Networking (IEC-61850, DNP3, ...)

Agenda

NI Power Electronics Control Design V Training Workshop

Workshop Agenda

1. Power Electronics
2. NI Design V Toolchain
3. Training Kits & Reference Design
4. The NI LabVIEW RIO Approach
5. Hands On Modules 1-3
6. Case Studies & Latest Releases

Hands-On Exercises

Module 1: Single Phase Power Conversion

1. Configuring your NI RIO Embedded System
2. Understand the GPIC Reference Design App.
3. Half-Bridge DC-to-AC Inverter Control
4. Pulse Width Modulation Logic
5. Sine-Triangle Pulse Width Modulation
6. Comparing Simulated vs. Experimental Results

Module 2: Three Phase Grid Synchronized Inverter Control

1. Understanding 3-Phase Power and the PLL
2. 3-Phase DC-to-AC Inverter Control
3. Comparing Simulated vs. Experimental Results
4. Grid Synchronized Inverter Control

Module 3: AC Induction Motor Control

1. Understanding the 3-Phase AC Induction Motor
2. Sine-Triangle AC Induction Motor Control
3. Comparing Simulated vs. Experimental Results
4. Voltage-over-Frequency Control
5. PID Control Tutorial

Module 4: Power Electronics Real-Time HIL Simulation

1. LabVIEW FPGA Floating Point Math Palette
2. Transfer Function and State-Space Modeling
3. Single-Phase Inverter HIL (Transfer Function)
4. Three-Phase Inverter HIL (State-Space)
5. AC Induction Motor HIL (Custom)
6. Automatic Multisim-to-FPGA Conversion HIL

Pre-Requisites

Please complete the pre-requisites below before beginning this hands-on workshop. If you hit a snag or have questions, make a note of it and move on to the next exercise. Email your notes to the instructor when finished.

1. Install the development tools on your computer. Follow the instructions in the PDF document below. Install in evaluation mode.

Download and install the NI development tools and NI CompactRIO Driver Software

2. Review the introduction to LabVIEW tutorials:

Introduction to NI LabVIEW

3. Complete the introduction to Multisim exercise:

Introduction to Multisim: Learn to Capture and Simulate in Less Than 30 Minutes

4. Complete the introduction to FPGA co-simulation exercise, beginning with part 3.

Introduction to Digital and Analog Co-simulation Between NI LabVIEW and NI Multisim

5. Download and unzip the power electronics hands on training software and instruction manual.

Download and unzip the training software (NI GPIC Reference Design Project)

(You must unzip to a short directory path such as C:\PowerDev.)

Download the instruction manual

[Module 1, Exercise 2]

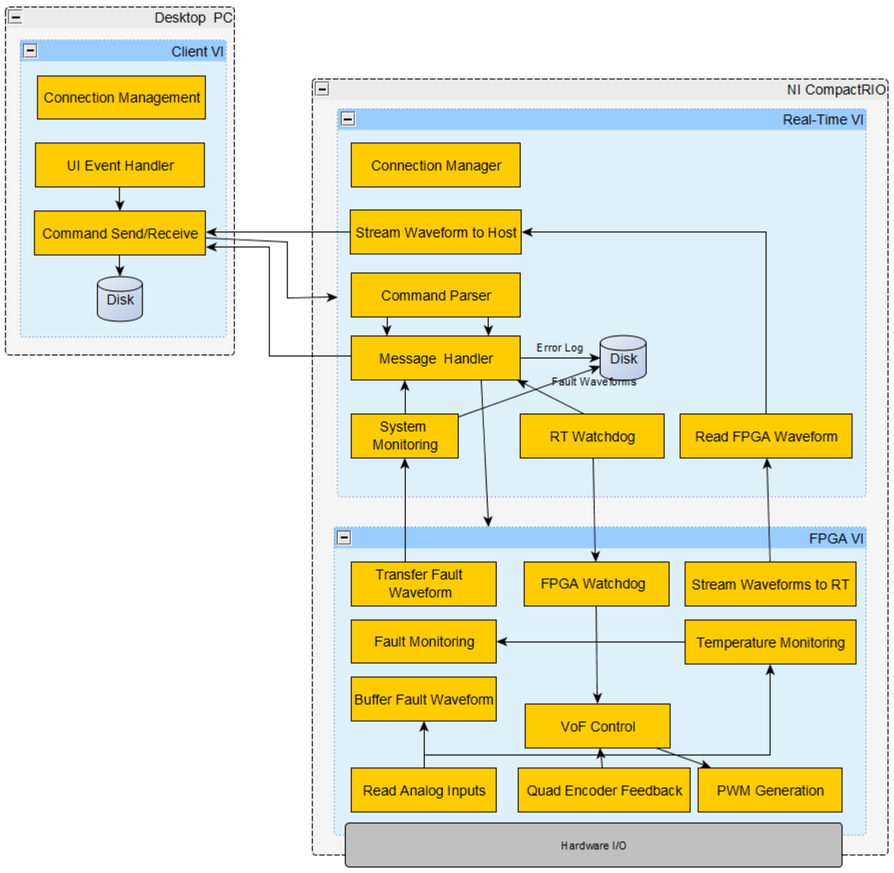
Understanding the GPIC Reference Design Application

Induction Motor Drive (Inverter A) and Grid Synchronized Inverter (Inverter B)

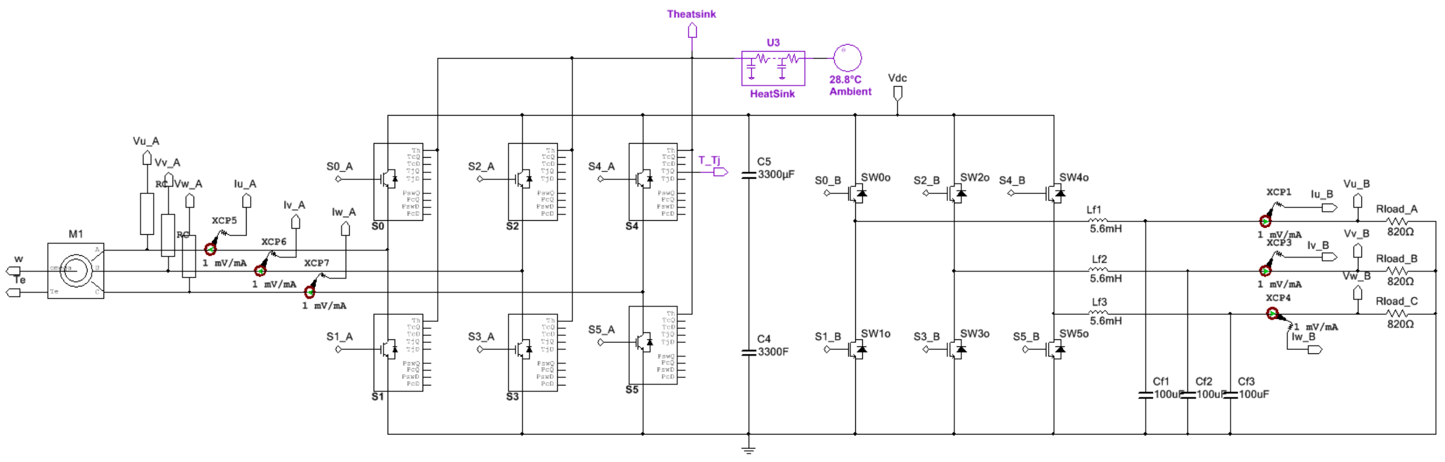
NI GPIC Reference Design Software

- Fully instrumented turnkey control demonstration software with network user interface, live scope display, and configurable control settings
- 10 kHz waveform graphing and data logging with live status display and automatic fault capture (up to 116 kS/s)
- Support for external isolated phase voltage and grid voltage sensors with configurable sensor gain, offsets and filtering
- Datalog and fault log viewing and analysis with Microsoft Excel, The MathWorks, Inc. MATLAB®, NI DIAdem, and NI LabVIEW (including 3-phase power quality analysis)
- Open source reference design code
- Download: ftp://ftp.ni.com/evaluation/powerdev/training/GPICReferenceDesign.zip

Software Block Diagram



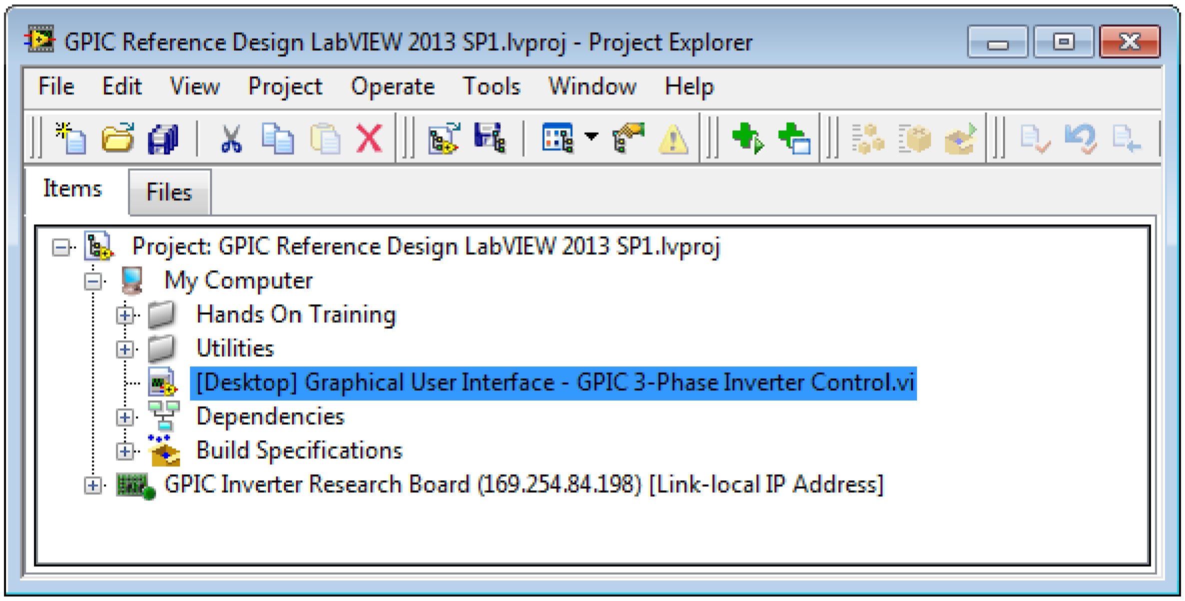
Circuit Schematic



Exercise time: 15-30 minutes

1. Confirm that the LED on your sbRIO GPIC control system is blinking on and off, indicating that the RT and FPGA applications are running and waiting for a network connection. (If not, deploy the startup executable.)

2. Under My Computer, open **[Desktop] Graphical User Interface - GPIC 3-Phase Inverter Control.vi**.

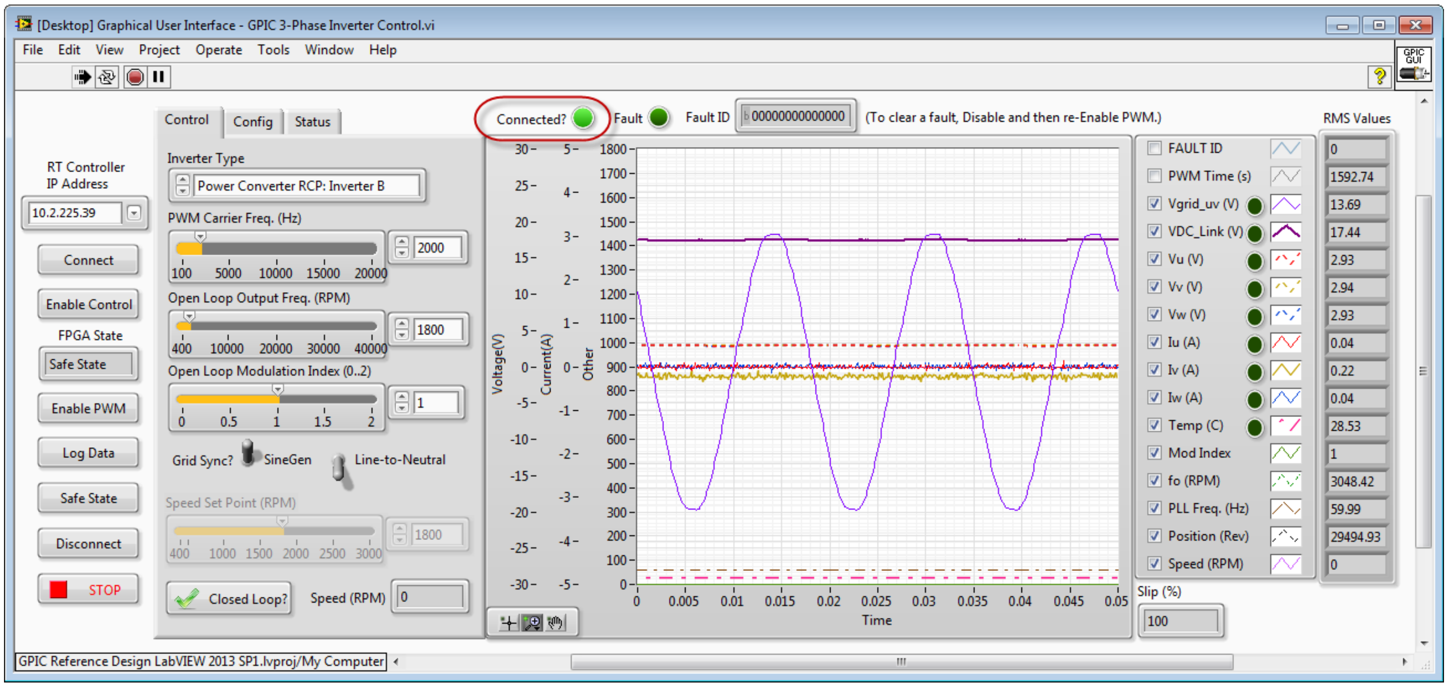

GPIC Reference Design LabVIEW 2013 SP1.lvproj - Project Explorer

File Edit View Project Operate Tools Window Help

Items Files

```
Project: GPIC Reference Design LabVIEW 2013 SP1.lvproj
    My Computer
        Hands On Training
        Utilities
        [Desktop] Graphical User Interface - GPIC 3-Phase Inverter Control.vi
        Dependencies
        Build Specifications
    GPIC Inverter Research Board (169.254.84.198) [Link-local IP Address]
```

3. Run the application. It will scan the Ethernet network and populate the **RT Controller IP Address** ring control with NI LabVIEW RIO targets. The **Connected?** indicator should illuminate. If you are not automatically connected, select the appropriate **RT Controller IP Address** and click **Connect**.



[Desktop] Graphical User Interface - GPIC 3-Phase Inverter Control.vi

RT Controller IP Address: 10.2.225.39

FPGA State: Safe State

Inverter Type: Power Converter RCP: Inverter B

PWM Carrier Freq. (Hz): 2000

Open Loop Output Freq. (RPM): 1800

Open Loop Modulation Index (0..2): 1

Grid Sync? SineGen — Line-to-Neutral

Speed Set Point (RPM): 1800

Speed (RPM): 0

Connected? — Fault — Fault ID: 00000000000000 (To clear a fault, Disable and then re-Enable PWM.)

	RMS Values
☐ FAULT ID	0
☐ PWM Time (s)	1592.74
☑ Vgrid_uv (V)	13.69
☑ VDC_Link (V)	17.44
☑ Vu (V)	2.93
☑ Vv (V)	2.94
☑ Vw (V)	2.93
☑ Iu (A)	0.04
☑ Iv (A)	0.22
☑ Iw (A)	0.04
☑ Temp (C)	28.53
☑ Mod Index	1
☑ fo (RPM)	3048.42
☑ PLL Freq. (Hz)	59.99
☑ Position (Rev)	29494.93
☑ Speed (RPM)	0

Slip (%): 100

GPIC Reference Design LabVIEW 2013 SP1.lvproj/My Computer

4. Click **Enable Control**, then **Enable PWM**. You should hear the pre-charge contactor close, and waveforms from the RLC line reactor filter (**Power Converter RCP: Inverter B**) should appear.

5. Try changing the **PWM Carrier Freq. (Hz).** How does a higher carrier frequency impact the power quality (harmonic distortion) of the voltage and current waveforms?

6. Try changing the **Open Loop Output Freq. (RPM)**. Note that an open loop output frequency of 1800 is equivalent to 60 Hz (divide by 30 to convert from RPM to Hz).

7. Try changing the **Open Loop Modulation Index (0..2)**. Are the voltage and current RMS values linearly proportional for the modulation index values ranging from 0 to 1? What about modulation index values greater than 1? (The inverter amplitude response is non-linear when overmodulation is used.) What happens to the voltage and current waveforms when the modulation index is set to 2? (The voltage waveforms approach a square wave. The current waveforms are highly non-sinusoidal.)

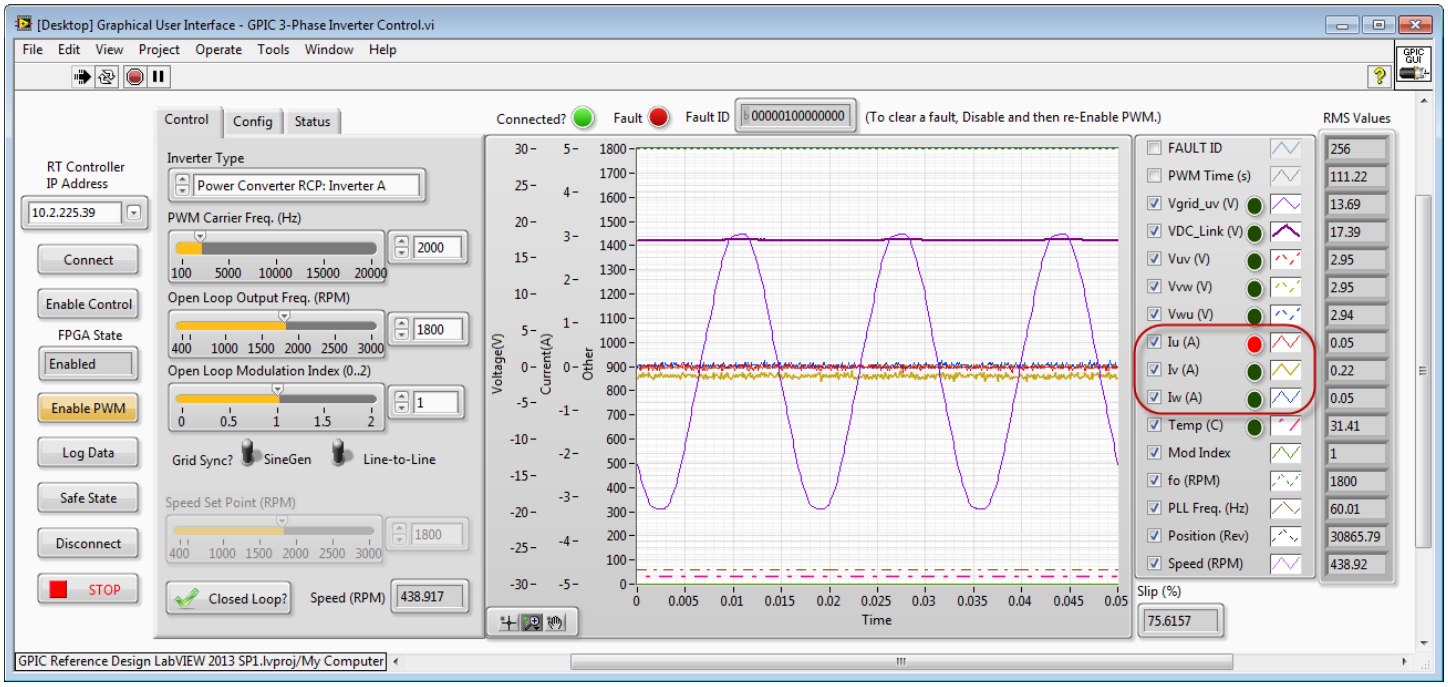
8. Try clicking the **Line-to-Neutral** button to go to **Line-to-Line** mode. Note that the labels for the phase voltages change, and the voltage waveform is now bipolar rather than unipolar.

9. Try clicking the **Grid Sync?** button to phase lock the 3-phase inverter with the grid input signal. Which of the three line-to-line inverter output voltages is phased locked with the single phase grid input signal?

10. Change the **Grid Sync?** mode back to **SineGen**. Now change the **Inverter Type** to **Power Converter RCP: Inverter A**. The 3-phase induction motor begins spinning. In some cases, a fault may occur due to overcurrent since you just “hard started” the induction motor. If that occurs,

set the **Open Loop Modulation Index (0..2)** to 0, disable and re-enable PWM to clear the fault, and then bring the modulation up slowly from 0 to 1.

11. What is the **Slip (%)** and motor **Speed (RPM)**? This is the difference between the frequency of the excitation voltage from the 3-phase inverter and the motor rotation frequency.
12. Try increasing the **Open Loop Output Freq. (RPM)** in 100 RPM increments. (You can use the up/down arrows on the control.) What is the slip when the excitation frequency is 3000 RPM?
13. To fault the inverter intentionally, try changing the **Open Loop Output Freq. (RPM)** to 1800 RPM in a single step. (Type in the value 1800 and hit enter.) Looking at the LEDs next to the channel labels on the graph, which phase current caused the overvoltage fault? The phase that causes the fault depends on the rotation angle of the motor when you changed the excitation frequency suddenly. A sudden rush of current occurs because the excitation voltage does not match the rotational angle (and rotor flux) of the motor.



[Desktop] Graphical User Interface - GPIC 3-Phase Inverter Control.vi

RT Controller IP Address: 10.2.225.39

FPGA State: Enabled

Inverter Type: Power Converter RCP: Inverter A

PWM Carrier Freq. (Hz): 2000

Open Loop Output Freq. (RPM): 1800

Open Loop Modulation Index (0..2): 1

Grid Sync? SineGen / Line-to-Line

Speed Set Point (RPM): 1800

Closed Loop? Speed (RPM): 438.917

Fault ID: b00000100000000 (To clear a fault, Disable and then re-Enable PWM.)

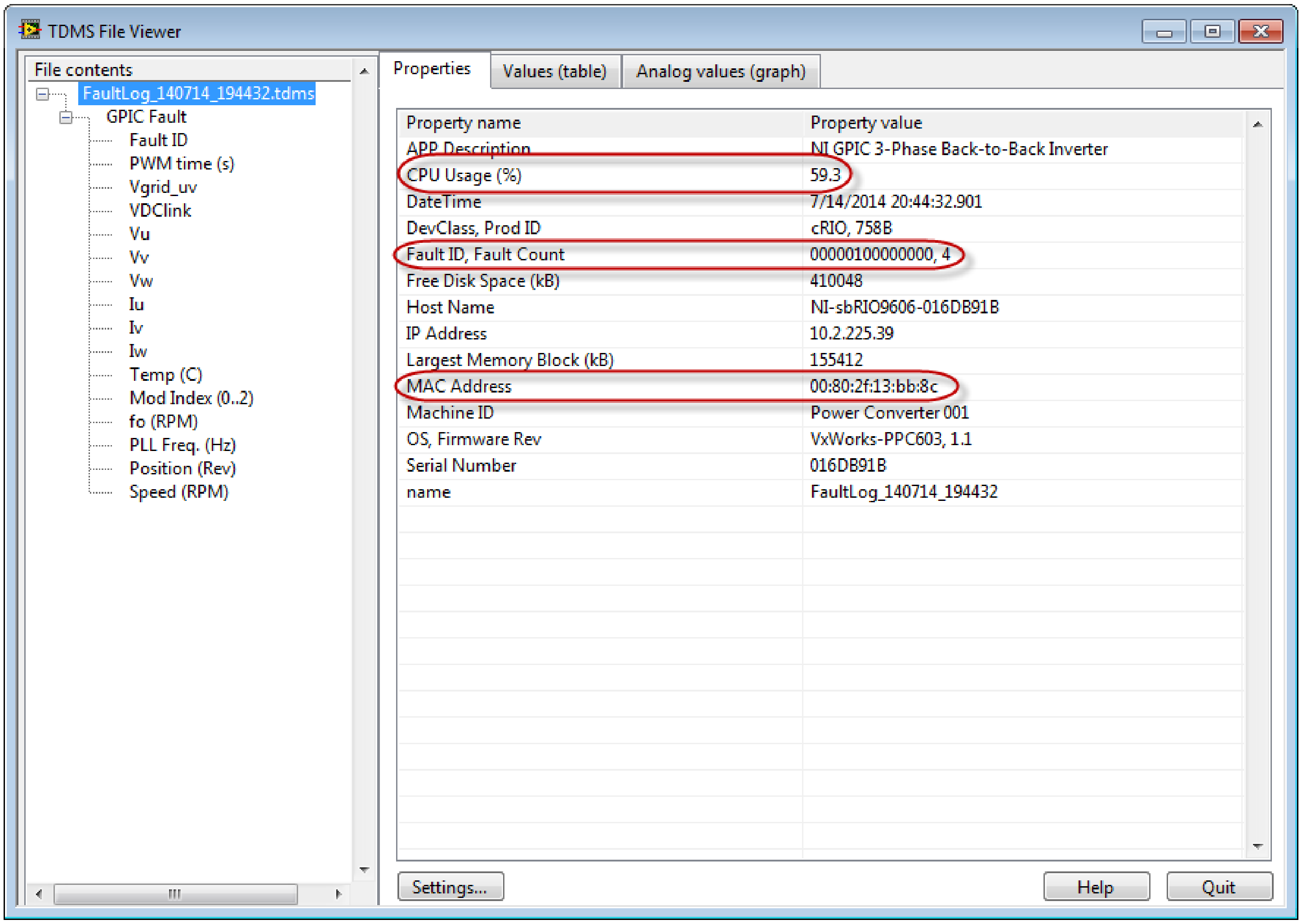
	RMS Values
FAULT ID	256
PWM Time (s)	111.22
Vgrid_uv (V)	13.69
VDC_Link (V)	17.39
Vuv (V)	2.95
Vvw (V)	2.95
Vwu (V)	2.94
Iu (A)	0.05
Iv (A)	0.22
Iw (A)	0.05
Temp (C)	31.41
Mod Index	1
fo (RPM)	1800
PLL Freq. (Hz)	60.01
Position (Rev)	30865.79
Speed (RPM)	438.92

Slip (%): 75.6157

GPIC Reference Design LabVIEW 2013 SP1.lvproj/My Computer

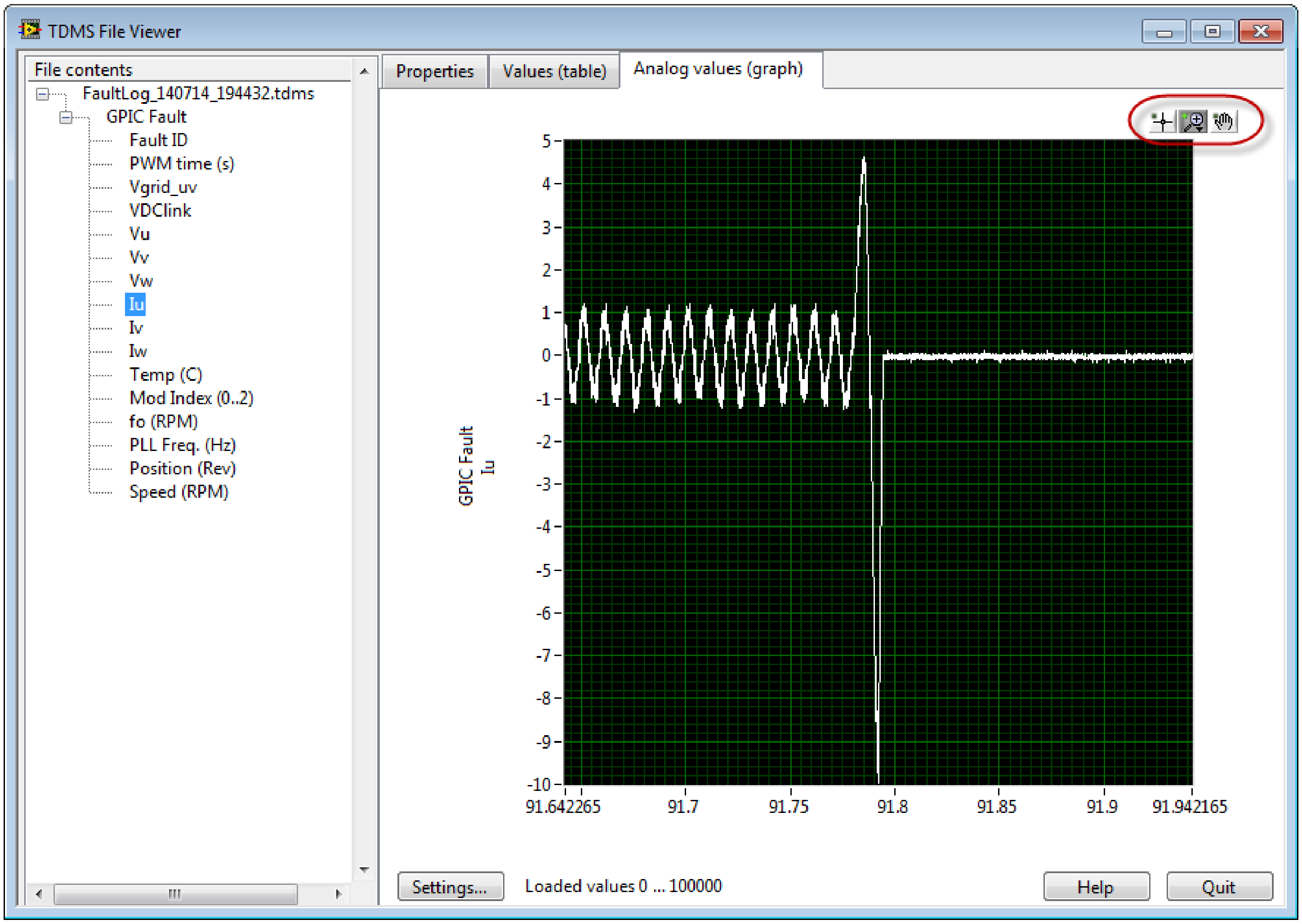
14. A TDMS log file containing the inverter waveform data before and after the fault automatically appears. Note that the Status tab on the GUI application displays the **Fault Data File Path**. This data is saved in FPGA RAM and then transferred to the GPIC RT controller if a fault occurs, and is logged to the non-volatile storage on the sbRIO GPIC system. The desktop GUI application is notified that a fault log file is available and it is retrieved. Then the GPIC RT controller deletes its local copy.
15. Click on the top item in the **File contents** tree to observe the meta data saved with the fault log. What was the GPIC RT **CPU Usage (%)** when the fault occurred. What is the **Fault ID**, and how many faults have previously occurred (**Fault Count**)? Note that the **MAC Address** is listed on a

label located on the top of your sbRIO GPIC control board and provides an easy way to identify it.



Property name	Property value
APP Description	NI GPIC 3-Phase Back-to-Back Inverter
CPU Usage (%)	59.3
DateTime	7/14/2014 20:44:32.901
DevClass, Prod ID	cRIO, 758B
Fault ID, Fault Count	00000100000000, 4
Free Disk Space (kB)	410048
Host Name	NI-sbRIO9606-016DB91B
IP Address	10.2.225.39
Largest Memory Block (kB)	155412
MAC Address	00:80:2f:13:bb:8c
Machine ID	Power Converter 001
OS, Firmware Rev	VxWorks-PPC603, 1.1
Serial Number	016DB91B
name	FaultLog_140714_194432

16. In the **File contents** tree, click on the current signal corresponding to the phase current that caused the fault. Then select **Analog values (graph)**. Do you observe the current measurement going over the limit? 91% of the time you will not see the actual sample when the fault occurred displayed on the graph. That's because the fault buffer data is being acquired at 10 kHz (100 usec) intervals, whereas the FPGA simultaneous analog input and fault detection logic is execution at 115,942 Hz (8.625 microseconds). The fault buffer contains 3,000 data points, so at 10 kHz this provides 0.3 seconds of data. (Optionally, fault buffer recording can be executed at up to 6 MHz rates, or at slower rates to provide a longer duration of data.)



17. Try zooming in and out on the current waveforms. Also click to view the other signals that were captured. What happened to the DC link voltage (**VDClink**) when the inverter tripped off? Note that **Fault ID** always goes true in the middle of the fault capture waveform, which contains the inverter data before and after the fault.

18. Click the Quit button to close the **TDMS File Viewer**.

19. Back on the GUI application, click the **Log Data** button to continuously log data on your local computer. Then click Enable PWM to disable, and then click again to re-enable. Note that the FPGA application clears the fault on the rising edge of **Enable PWM**.

20. Now click the **Closed Loop?** button to change the FPGA control mode to Voltage-over-Frequency (V/f) control. Does the motor **Speed (RPM)** match your setpoint? Try changing the **Speed Set Point (RPM)** in 100 RPM increments.

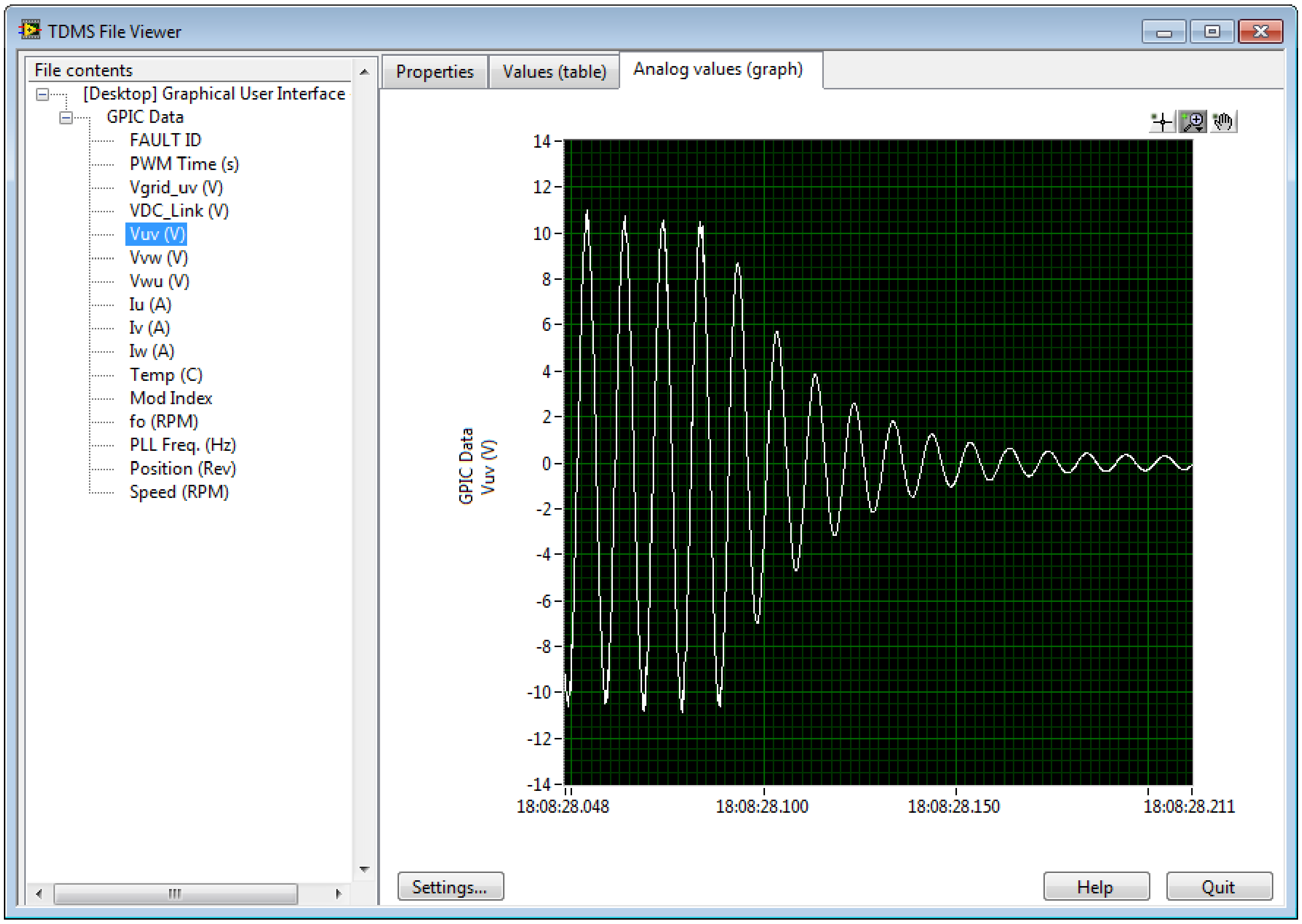
21. What is the inverter excitation frequency, fo (Hz) when the **Speed Set Point (RPM)** is **3000**?

22. Click **Enable PWM** to disable the inverter. Then click the **STOP** button. You should hear the pre-charge contactor open.

23. The **TDMS File Viewer** automatically appears with your data file you recorded.

24. Click on the top item in the **File contents** tree to observe the meta data saved in the TDMS log file. Now expand the tree and click on the **Vuv (V)** waveform data. Since this is a long recording, click on **Settings…** and then select **Load all values** under **Number of values (graph)**.

25. Zoom in to the time period at the end of the waveform when you disabled the inverter. Note that even though you removed the excitation voltages from the induction motor instantaneously, the phase voltages did not decay immediately. The reason is that there are still currents circulating in the rotor of the induction motor, and these induce a decaying back EMF voltage that is observed at the inverter terminals. The time from peak to 63.3% decay is the electrical time constant (tau) of the induction motor. In this case, the time constant is about 45 milliseconds, or 22.2 Hz. From that, we can infer the minimum (10/tau) and recommended (100/tau) control loop rate in Hz, as a rule of thumb, for controlling the rotor flux. (In this case, the recommended control loop rate would be 2222 Hz.)



26. Using the zoom capability of the file viewer, explore the other waveforms. Then click **Quit** to close the **TDMS File Viewer**.

27. This completes the exercise. Let the instructor know you are finished, and ask them if you have any unanswered questions about the GPIC Reference Design GUI Application.

Exercise 3: Half-Bridge Inverter Control

Exercise 3 | Half-Bridge Inverter Control

Summary

In this exercise, you will open and run a preconfigured file that simulates an application running on the FPGA using the Desktop Execution Node. The application switches two IGBTs in order to produce an analog signal. To open the application and run the simulation you will complete the following tasks:

- Open the application through the project window
- Run the windows application
- Vary the controls for the IGBT switches

Background

The field-programmable gate array (FPGA) allows a user to perform a variety of tasks, including signal processing, at higher speeds than most processors. The FPGA Desktop Execution Node allows a user to run an FPGA application on the development computer with simulated I/O. By specifying the desired FPGA clock speed, as well as the number of ticks before inputs are read and outputs are written, a user can determine the functionality of their FPGA application without needing to compile and deploy the application to the FPGA hardware.

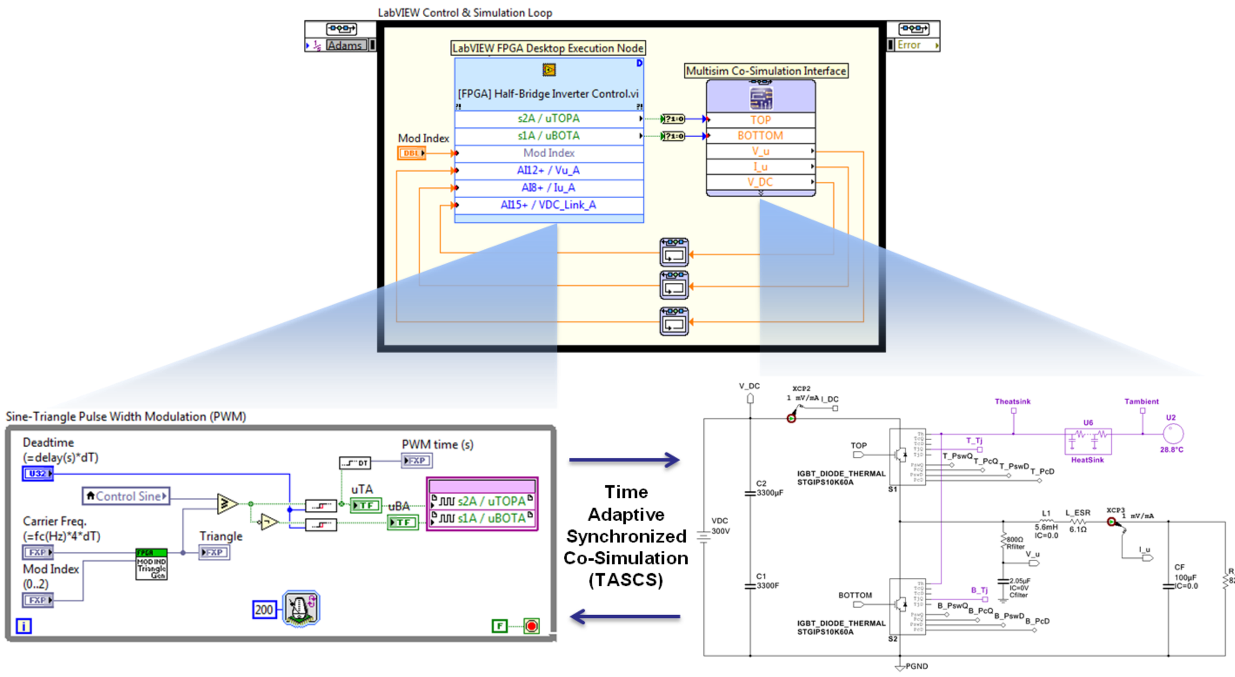
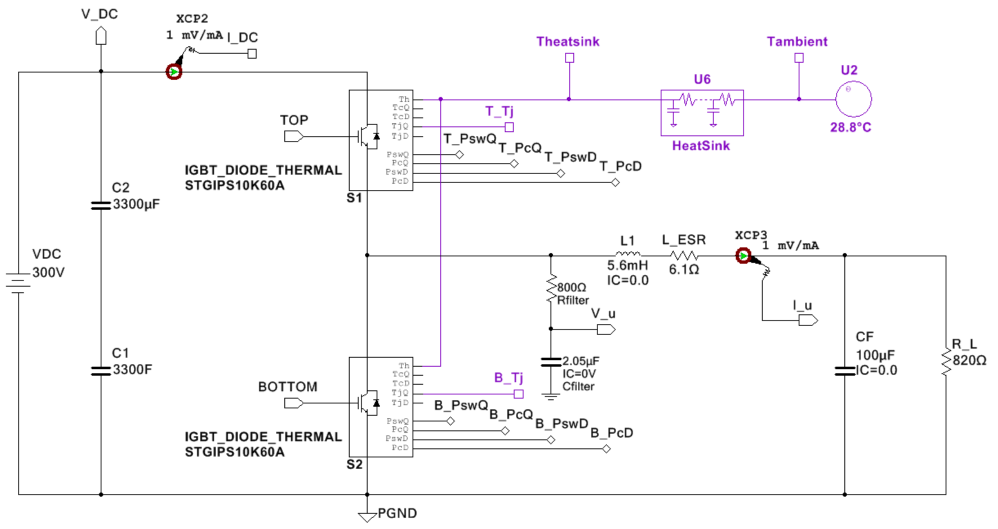


Figure 1. To enable high fidelity co-simulation, the LabVIEW FPGA Desktop Execution Node interfaces to the LabVIEW FPGA Application while the Multisim Co-Simulation Interface connects to the power electronics circuit model. A desktop testbench application is shown above.

For these exercises, we will be using LabVIEW FPGA in order to control a power inverter. A power inverter consists of a voltage source, two insulated-gate bipolar transistors (IGBTS), an inductor, a capacitor, and a resistor. The IGBTs switch in order to change the current flowing through the resistive load while the inductor and capacitor filter and smooth the output signal. The power inverter performs DC-to-AC to conversion.



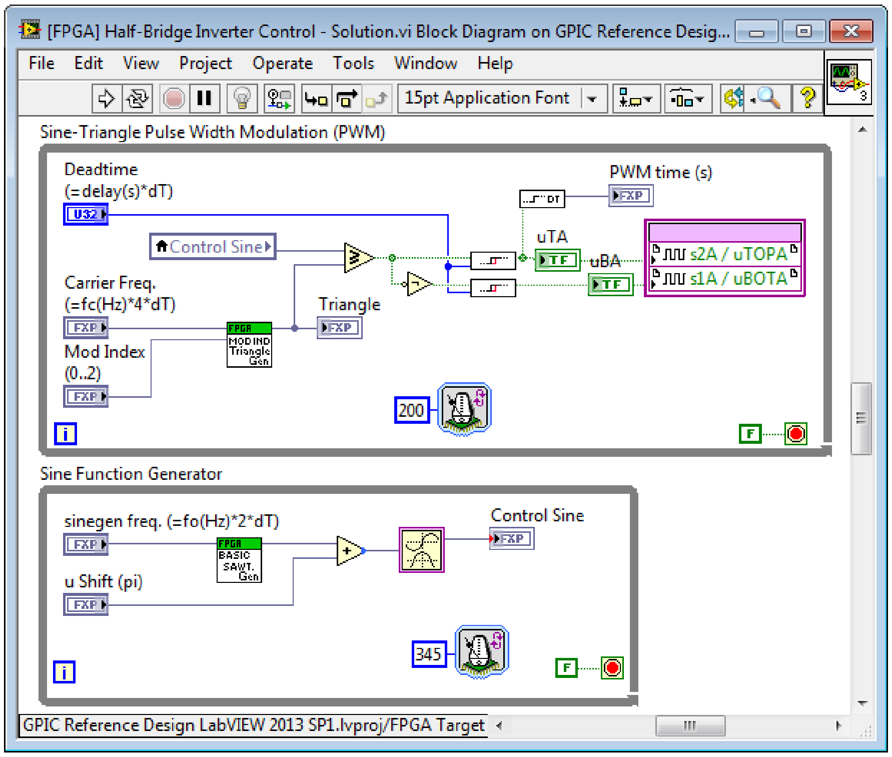
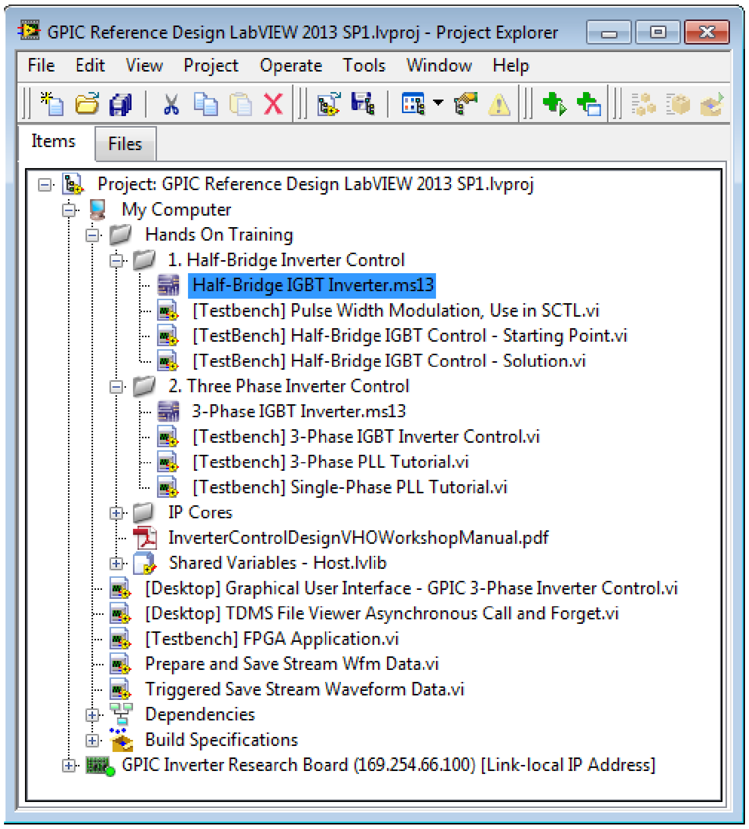


Figure 2. The Multisim power electronics circuit (top) and LabVIEW FPGA control application (bottom) are co-simulated with a common sense of simulated time

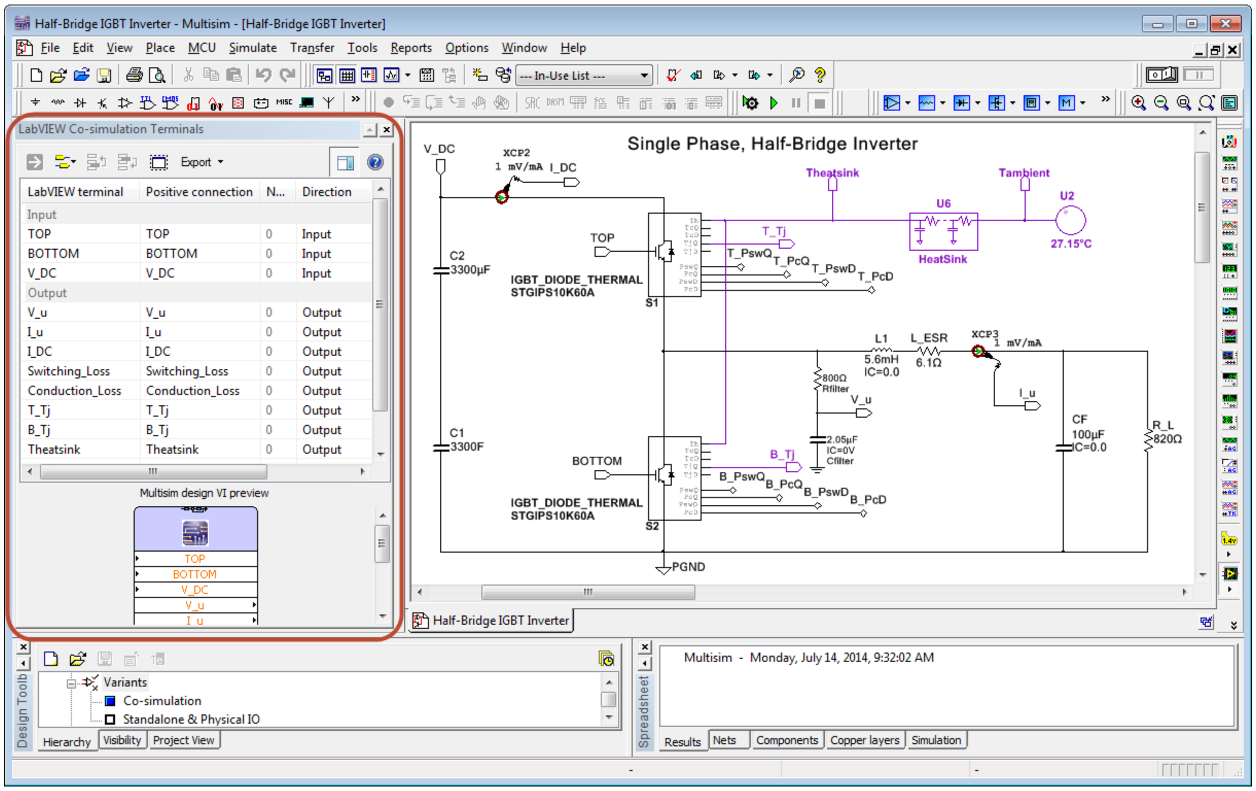
Explore the Co-Simulation Application

1. Open the LabVIEW project for this hands on workshop by navigating to **C:\ Powerdev \GPIC\GPIC Reference Design\Hands On Exercises\ GPIC Reference Design LabVIEW 2013 SP1.lvproj**.

2. In the project window, expand out **My Computer**, **Hands on Training**, and then **1. Half-Bridge Inverter Control**. There will be a Multisim file and two TestBench VIs. **Double-click** the Multisim schematic file for the half-bridge circuit, **Half-Bridge IGBT Inverter.ms13**.

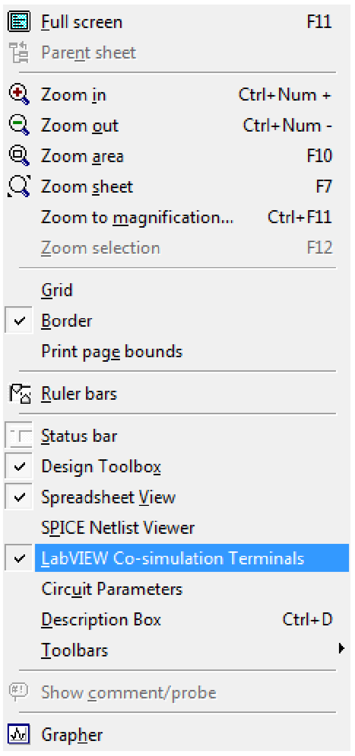

GPIC Reference Design LabVIEW 2013 SP1.lvproj - Project Explorer
File Edit View Project Operate Tools Window Help
Items
Files

```
Project: GPIC Reference Design LabVIEW 2013 SP1.lvproj
  My Computer
    Hands On Training
      1. Half-Bridge Inverter Control
        Half-Bridge IGBT Inverter.ms13
        [Testbench] Pulse Width Modulation, Use in SCTL.vi
        [TestBench] Half-Bridge IGBT Control - Starting Point.vi
        [TestBench] Half-Bridge IGBT Control - Solution.vi
      2. Three Phase Inverter Control
        3-Phase IGBT Inverter.ms13
        [Testbench] 3-Phase IGBT Inverter Control.vi
        [Testbench] 3-Phase PLL Tutorial.vi
        [Testbench] Single-Phase PLL Tutorial.vi
      IP Cores
      InverterControlDesignVHOWorkshopManual.pdf
      Shared Variables - Host.lvlib
    [Desktop] Graphical User Interface - GPIC 3-Phase Inverter Control.vi
    [Desktop] TDMS File Viewer Asynchronous Call and Forget.vi
    [Testbench] FPGA Application.vi
    Prepare and Save Stream Wfm Data.vi
    Triggered Save Stream Waveform Data.vi
    Dependencies
    Build Specifications
  GPIC Inverter Research Board (169.254.66.100) [Link-local IP Address]
```

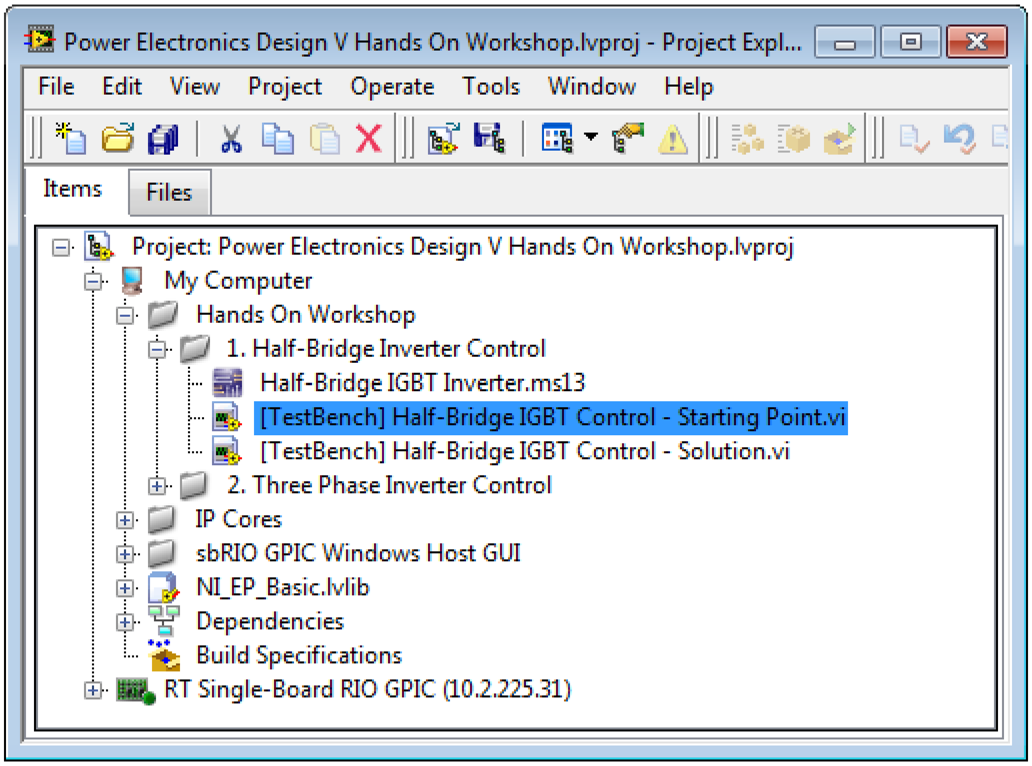

Half-Bridge IGBT Inverter - Multisim - [Half-Bridge IGBT Inverter]
File Edit View Place MCU Simulate Transfer Tools Reports Options Window Help
LabVIEW Co-simulation Terminals
Export
Multisim design VI preview
Single Phase, Half-Bridge Inverter
Half-Bridge IGBT Inverter
Variants
Co-simulation
Standalone & Physical IO
Hierarchy Visibility Project View
Results Nets Components Copper layers Simulation
Multisim - Monday, July 14, 2014, 9:32:02 AM

LabVIEW terminal	Positive connection	N...	Direction
Input			
TOP	TOP	0	Input
BOTTOM	BOTTOM	0	Input
V_DC	V_DC	0	Input
Output			
V_u	V_u	0	Output
I_u	I_u	0	Output
I_DC	I_DC	0	Output
Switching_Loss	Switching_Loss	0	Output
Conduction_Loss	Conduction_Loss	0	Output
T_Tj	T_Tj	0	Output
B_Tj	B_Tj	0	Output
Theatsink	Theatsink	0	Output

3. In the Multisim editor environment, if you do not see the **LabVIEW Co-Simulation Terminals** pane (red box in the screenshot above), go to the **View** menu, select it and then position it in the desired location.



4. In the LabVIEW Project, double-click on **[TestBench] Half-Bridge IGBT Control - Starting Point.vi**.

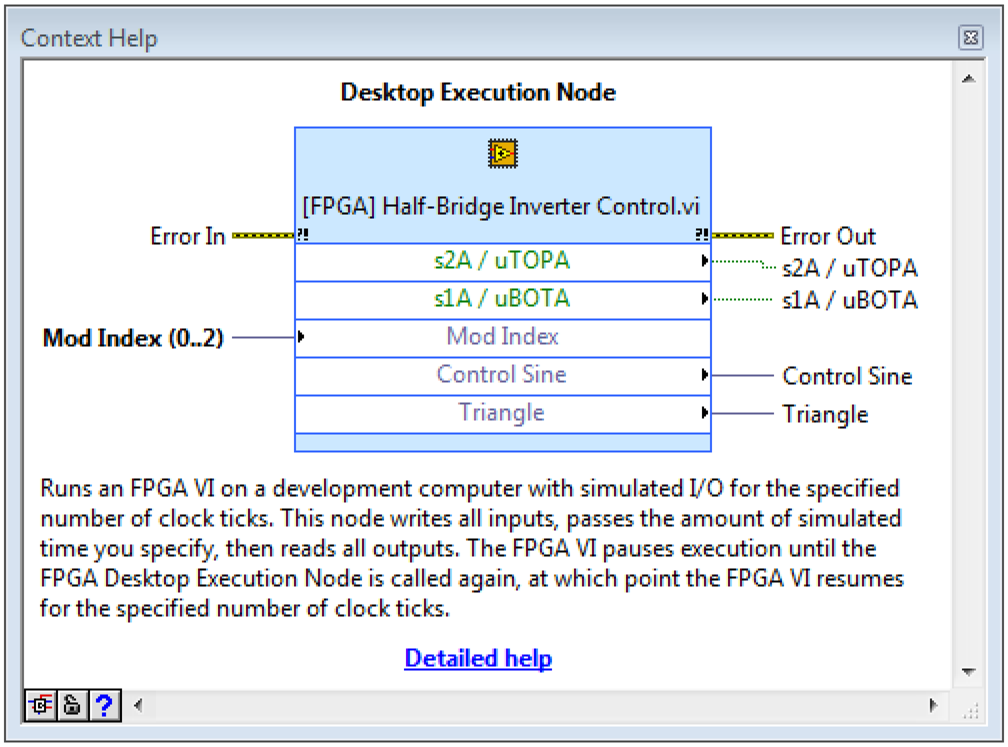

Power Electronics Design V Hands On Workshop.lvproj - Project Expl...
File Edit View Project Operate Tools Window Help
Items
Files

Project: Power Electronics Design V Hands On Workshop.lvproj
My Computer
Hands On Workshop
1. Half-Bridge Inverter Control
Half-Bridge IGBT Inverter.ms13
[TestBench] Half-Bridge IGBT Control - Starting Point.vi
[TestBench] Half-Bridge IGBT Control - Solution.vi
2. Three Phase Inverter Control
IP Cores
sbRIO GPIC Windows Host GUI
NI_EP_Basic.lvlib
Dependencies
Build Specifications
RT Single-Board RIO GPIC (10.2.225.31)

5. The VI may take a few moments to open. This is the front panel of the Half-Bridge IGBT Controller. You will notice a circuit diagram complete with two IGBT switches. The goal of this exercise will be to control these switches in such a way as to avoid damaging the hardware.
6. In addition to the switches, other controls include a slider for the modulation index and a button for stopping the program. Important indicators include the IGBT temperatures, a dial for the output voltage, both instant and RMS, as well as a chart which plots various waveforms from the circuit.

Note: During this exercise, the program will always be run from this VI.

7. Open the block diagram by pressing **CTRL+E** or by navigating to **Window»Show Block Diagram** and observe the layout. Press **CTRL+H** to open the **Context Help** window and place the mouse over the blue box which reads **Desktop Execution Node**. This will cause the **Context Help** window to display information about the **Desktop Execution Node.**



Desktop Execution Node

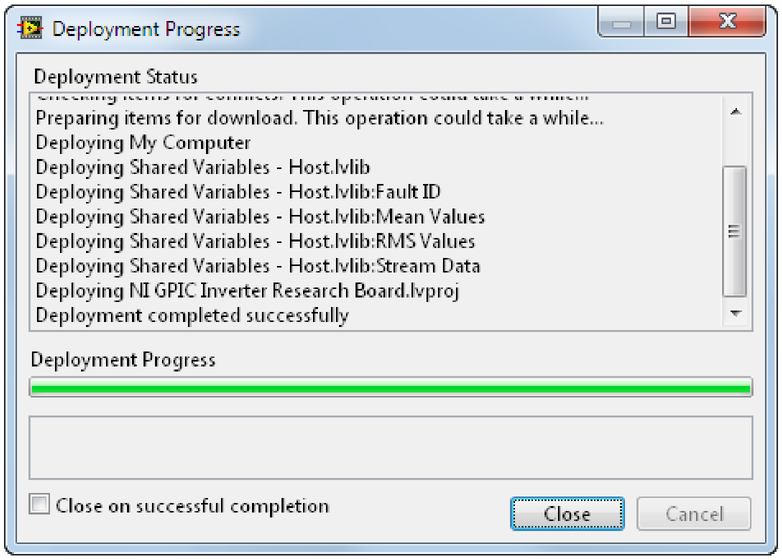
Runs an FPGA VI on a development computer with simulated I/O for the specified number of clock ticks. This node writes all inputs, passes the amount of simulated time you specify, then reads all outputs. The FPGA VI pauses execution until the FPGA Desktop Execution Node is called again, at which point the FPGA VI resumes for the specified number of clock ticks.

Detailed help

8. The Desktop Execution Node allows users to run FPGA VIs on the development computer with simulation I/O. Double-click the node to open up [FPGA] Half-Bridge Inverter Control.vi, but leave the font panel of [TestBench] Half-Bridge IGBT Control - Starting Point.vi open. It will be helpful to see both front panels while running the program.

Running the LabVIEW FPGA Control System

1. It is time to run the program and experiment with the IGBT switches. Navigate to the front panel of [TestBench] Half-Bridge IGBT Control - Starting Point.vi. Run the application by pressing the Run button on the toolbar. Check the box Close on successful completion. Select Close once the deployment is complete.


Deployment Progress

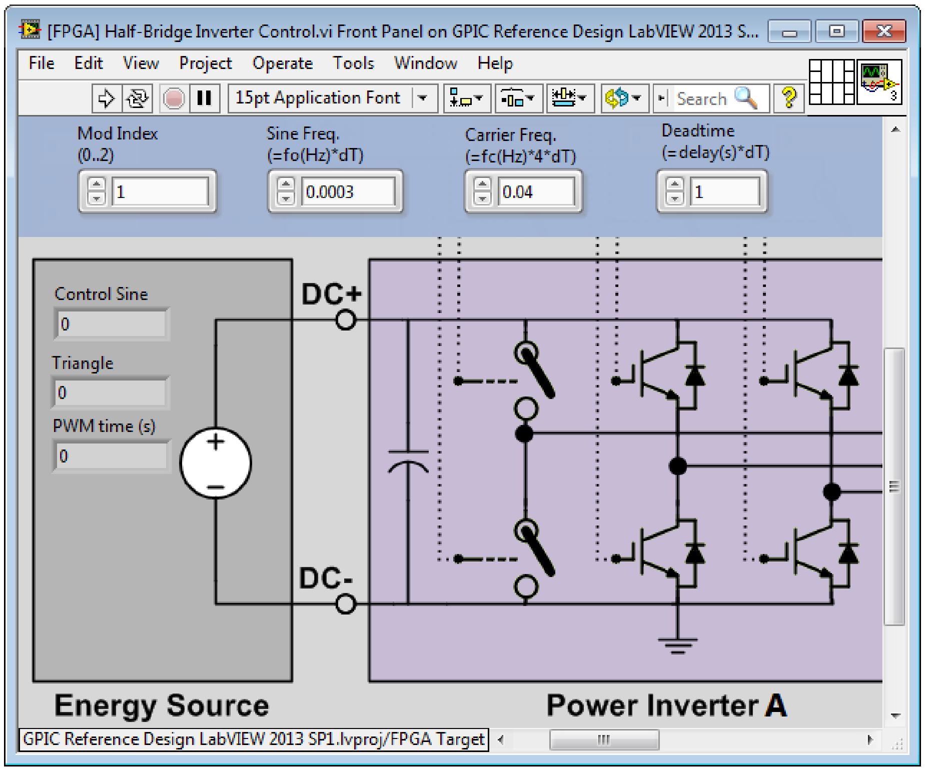
Deployment Status

Checking items for conflicts. This operation could take a while...
Preparing items for download. This operation could take a while...
Deploying My Computer
Deploying Shared Variables - Host.lvlib
Deploying Shared Variables - Host.lvlib:Fault ID
Deploying Shared Variables - Host.lvlib:Mean Values
Deploying Shared Variables - Host.lvlib:RMS Values
Deploying Shared Variables - Host.lvlib:Stream Data
Deploying NI GPIC Inverter Research Board.lvproj
Deployment completed successfully

Deployment Progress

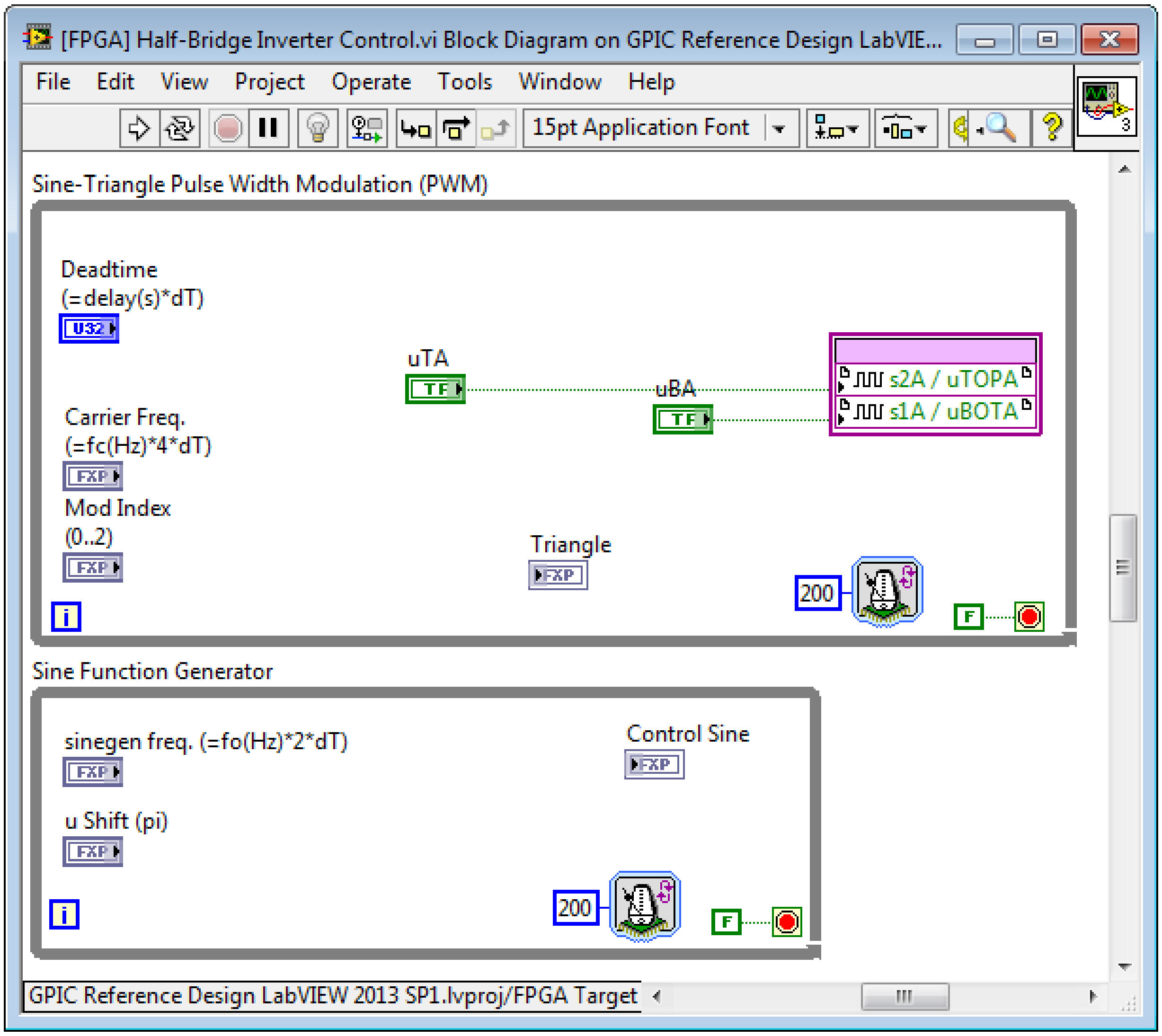
2. The application will take a moment to begin running. Once running, the waveforms will begin to be plotted on the graph.

3. While the program is running, open the front panel to **[FPGA] Half-Bridge Inverter Control.vi**. Left-click the **top switch** to close it. Observe the graph on the front panel of the **[TestBench] Half-Bridge IGBT Control - Starting Point.vi**. What happens?


[FPGA] Half-Bridge Inverter Control.vi Front Panel on GPIC Reference Design LabVIEW 2013 S...

Mod Index (0..2): 1
Sine Freq. (=fo(Hz)*dT): 0.0003
Carrier Freq. (=fc(Hz)*4*dT): 0.04
Deadtime (=delay(s)*dT): 1
Control Sine: 0
Triangle: 0
PWM time (s): 0

GPIC Reference Design LabVIEW 2013 SP1.lvproj/FPGA Target


[FPGA] Half-Bridge Inverter Control.vi Block Diagram on GPIC Reference Design LabVIE...
File Edit View Project Operate Tools Window Help
15pt Application Font
Sine-Triangle Pulse Width Modulation (PWM)
Deadtime
(=delay(s)*dT)
U32
Carrier Freq.
(=fc(Hz)*4*dT)
FXP
Mod Index
(0..2)
FXP
uTA
TF
uBA
TF
s2A / uTOPA
s1A / uBOTA
Triangle
FXP
200
F
i
Sine Function Generator
sinegen freq. (=fo(Hz)*2*dT)
FXP
u Shift (pi)
FXP
Control Sine
FXP
200
F
i
GPIC Reference Design LabVIEW 2013 SP1.lvproj/FPGA Target

4. Try closing only the bottom switch and then closing both switches. **What happens?** Congratulations! You are now a power electronics engineer. Observe the IGBT temperatures and input power when both switches are closed at the same time.

5. Press the **Stop** button with a red square on the front panel of **[TestBench] Half-Bridge IGBT Control - Starting Point.vi** in order to stop application execution.

Exercise 4 | Pulse Width Modulation Logic

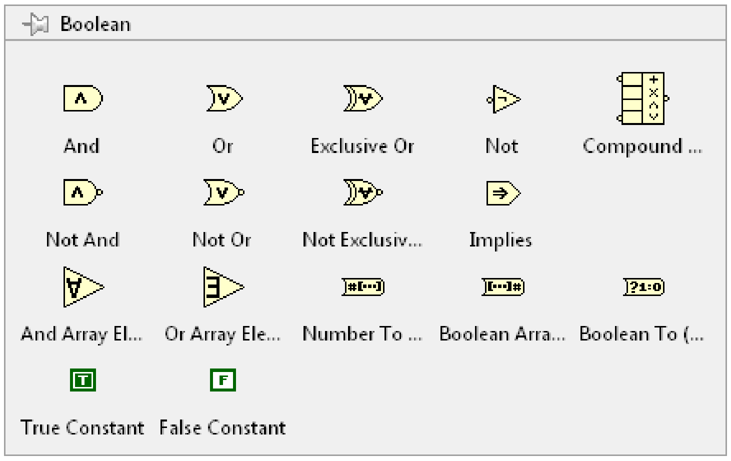
Summary

In order to implement your FPGA control application, it needs to be altered so as to not damage the IGBT power transistors. This exercise will work through the steps needed to change the operation of the FPGA switching logic using LabVIEW graphical programming. In this exercise you will implement these functionalities:

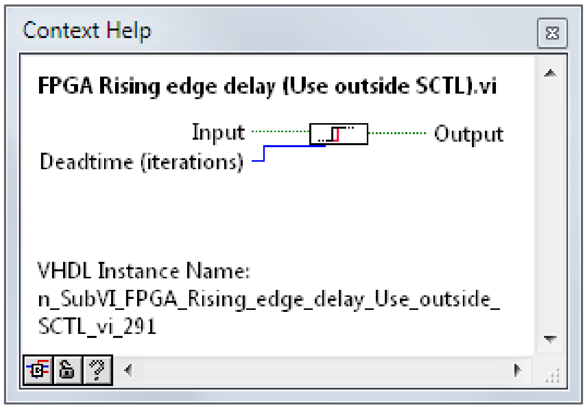
- Automatic switch inversion logic to produce complementary signals for the top and bottom switches
- Dead-time insertion to delay switch closings and prevent shoot-through

Background

In the previous exercise the temperature on the IGBTs should have spiked when both switches were closed. This is because closing both switches effectively creates a short circuit. In order to avoid this condition, known as shoot through, it is necessary to implement some control logic that ensures that the top and bottom switches are not closed at the same time. The LabVIEW FPGA programming palette provides a wide variety of logical operators. For this exercise we will be investigating the use of the NOT function. The bottom switch shall be configured to take a value that is the logical opposite, or complement, of the top switch. By inserting a NOT function, the logical opposite value of the top switch can be obtained.



Another important concept to understand is the physical limitation of the IGBT. Although it may appear that the bottom switch opens and closes in perfect unison with the top switch, there is actually a small delay in practice. If the delay is not accounted for, the shoot through condition may occur momentarily when changing states. This is where the use of a deadtime function is helpful. You will use an IP core that delays the closing of a switch (transition from False to True) for a specified number of rising edges of the FPGA clock, while not delaying a command to open a switch (transition from True to False). This will ensure that at no point are both switches closed.


Context Help

FPGA Rising edge delay (Use outside SCTL).vi

VHDL Instance Name:
n_SubVI_FPGA_Rising_edge_delay_Use_outside_SCTL_vi_291

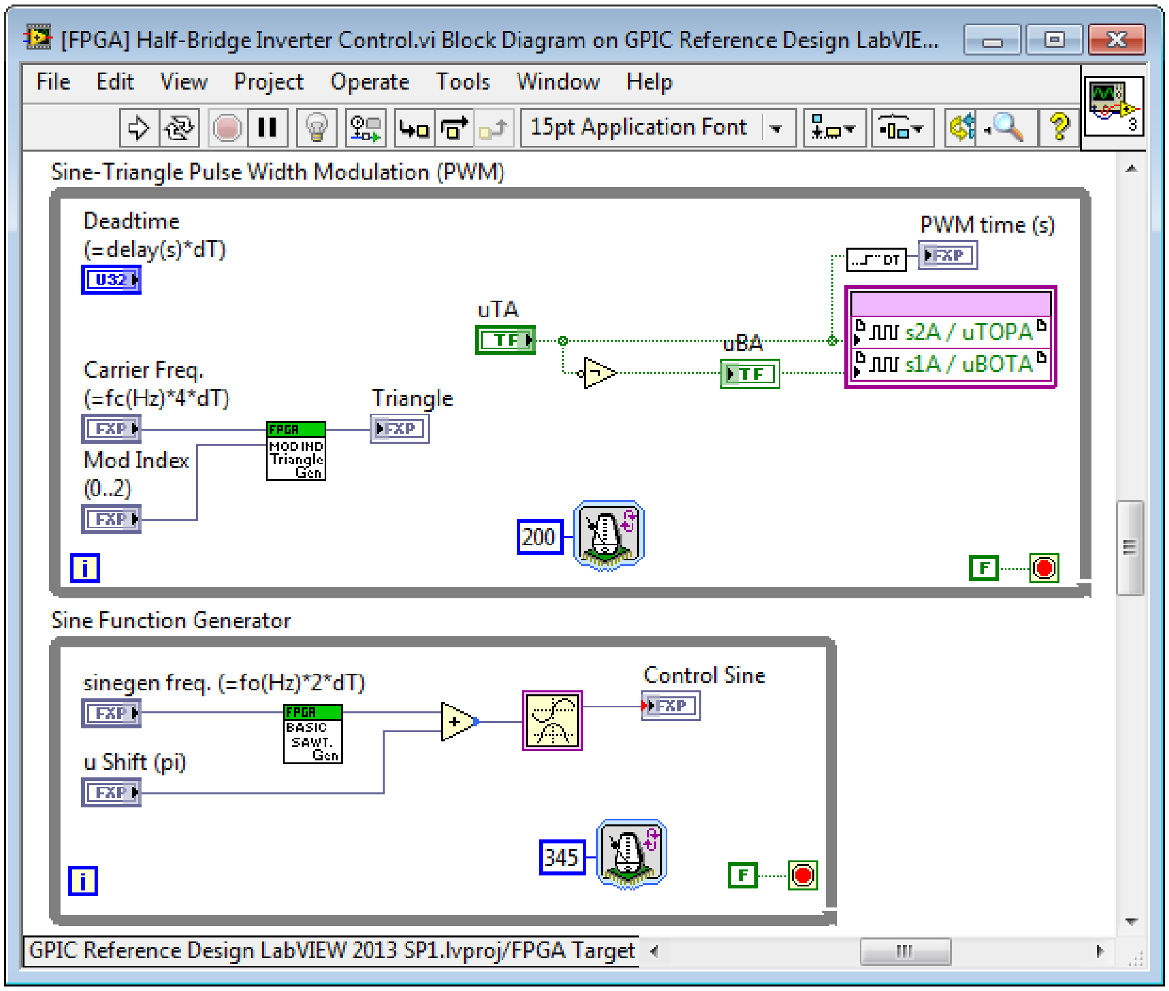
Adding Complementary Switching Logic

1. If they are not already, open the TestBench and FPGA VIs that were used in exercise 1.

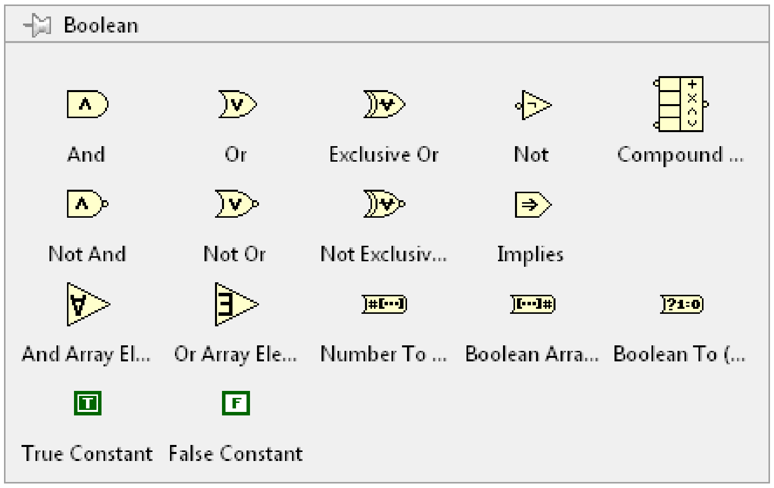
2. Navigate to the block diagram of the [FPGA] Half-Bridge Inverter Control.vi by pressing CTRL+E.

3. Note that the top switch is labeled as *uTA* and the bottom switch is labeled as *uBA*. Both switches are wired into the purple FPGA I/O node. This node is used to read and write signals to the FPGA.

4. Place the block diagram into edit mode by pressing CTRL+M. In this section, you will add a Not function and wire the block diagram as shown below.


[FPGA] Half-Bridge Inverter Control.vi Block Diagram on GPIC Reference Design LabVIE...
File Edit View Project Operate Tools Window Help
15pt Application Font
Sine-Triangle Pulse Width Modulation (PWM)
Deadtime
(=delay(s)*dT)
U32
PWM time (s)
FXP
uTA
TF
uBA
TF
s2A / uTOPA
s1A / uBOTA
Carrier Freq.
(=fc(Hz)*4*dT)
FXP
Triangle
FXP
Mod Index
(0..2)
FXP
200
F
Sine Function Generator
sinegen freq. (=fo(Hz)*2*dT)
FXP
Control Sine
FXP
u Shift (pi)
FXP
345
F
GPIC Reference Design LabVIEW 2013 SP1.lvproj/FPGA Target

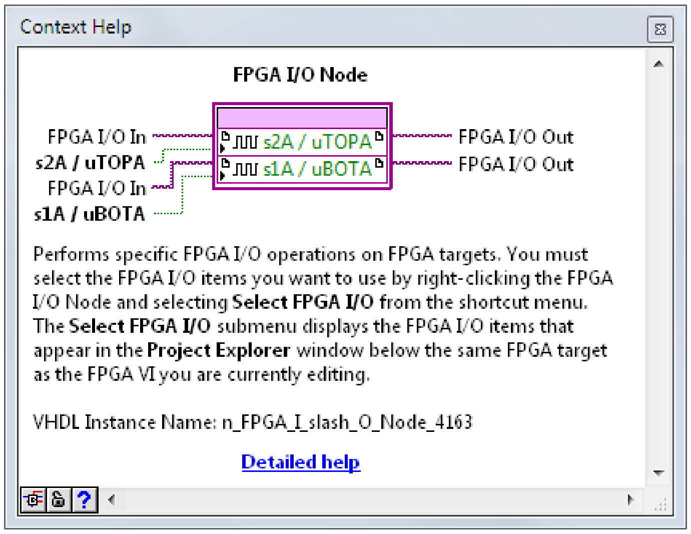
5. To do so, open the functions palette by **right-clicking** the block diagram. Navigate to the programming tab, select Boolean, and then left-click the **Not** function. Place the Not function onto the block diagram.



6. Delete the wire running from the bottom switch, *uBA*, to the FPGA I/O node.

7. **Right-click** the bottom switch, *uBA*, and select **Change to Indicator.** Wire the indicator to the output of the Not function. Your block diagram should appear similar to the one below.

Note: There are two inputs to *uBOTA* on the pink FPGA node. Wiring the **Not** function to the top input (**FPGA I/O In**) will not work. This input is used to select the FPGA, not the input for the switch. Be sure to wire the **Not** function to the bottom most input which is color coded green and labeled *s1A / uBOTA*.



Performs specific FPGA I/O operations on FPGA targets. You must select the FPGA I/O items you want to use by right-clicking the FPGA I/O Node and selecting **Select FPGA I/O** from the shortcut menu. The **Select FPGA I/O** submenu displays the FPGA I/O items that appear in the **Project Explorer** window below the same FPGA target as the FPGA VI you are currently editing.

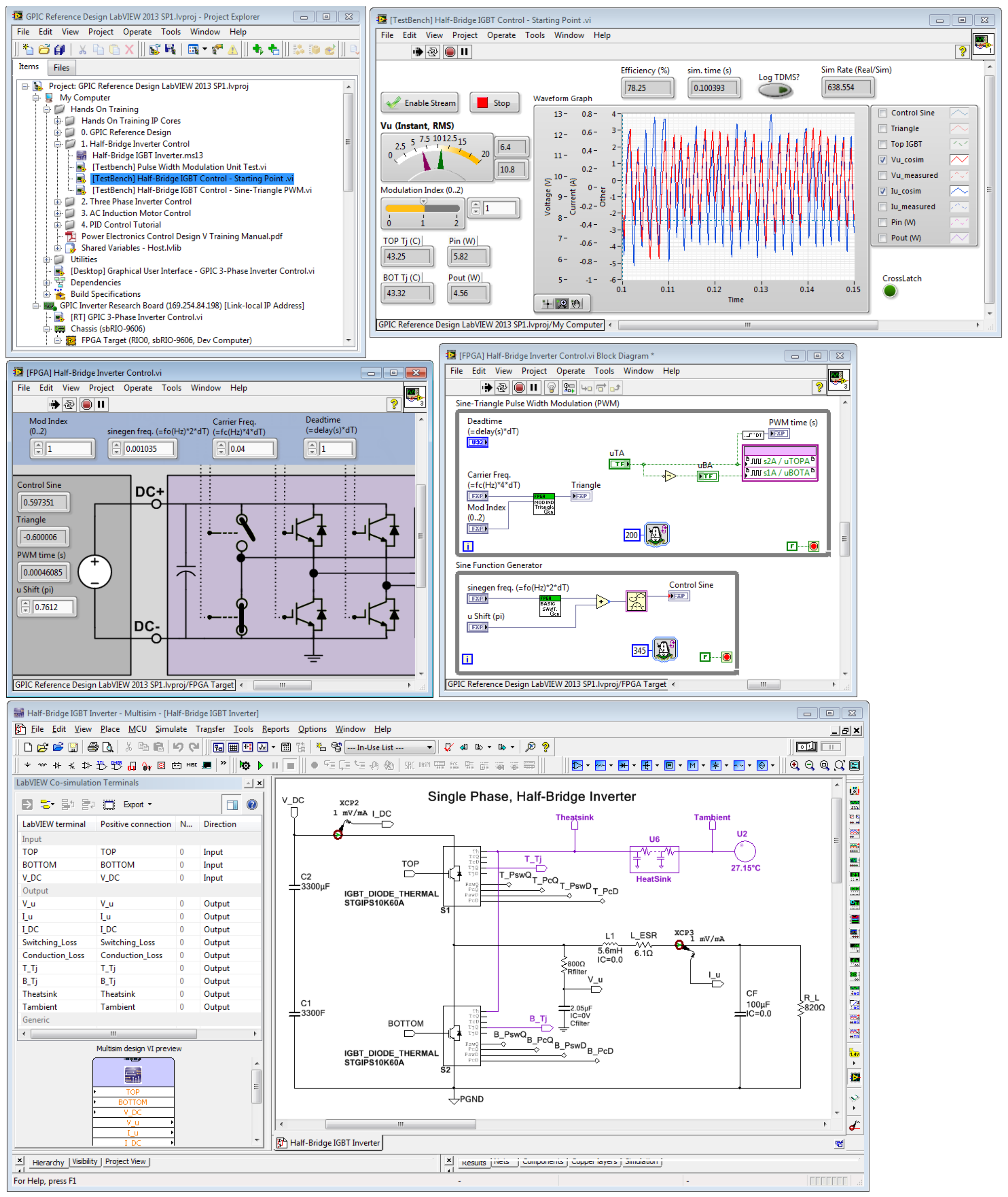
VHDL Instance Name: n_FPGA_I_slash_O_Node_4163

Detailed help

8. **Left-click** the node on the left side of the **Not** function and then **left-click** the wire running from the top switch, *uTA*, to the FPGA I/O node. This will place down a wire between these two points. **Left-click** the node on the right side of the **Not** function and then **left-click** the bottom input to the FPGA I/O node.

9. Try running the application again. This time, only press on the **top-switch**. What happens?

10. Try to press the **top-switch** on and off to produce pulses on the output. This is known as pulse width modulation (PWM). Try to modulate the pulse width to obtain an Vu RMS value of 10 V_{RMS}. How close can you get?



GPIC Reference Design LabVIEW 2013 SP1.lvproj - Project Explorer
Project: GPIC Reference Design LabVIEW 2013 SP1.lvproj
My Computer
Hands On Training
Hands On Training IP Cores
0. GPIC Reference Design
1. Half-Bridge Inverter Control
Half-Bridge IGBT Inverter.ms13
[Testbench] Pulse Width Modulation Unit Test.vi
[TestBench] Half-Bridge IGBT Control - Starting Point .vi
[TestBench] Half-Bridge IGBT Control - Sine-Triangle PWM.vi
2. Three Phase Inverter Control
3. AC Induction Motor Control
4. PID Control Tutorial
Power Electronics Control Design V Training Manual.pdf
Shared Variables - Host.lvlib
Utilities
[Desktop] Graphical User Interface - GPIC 3-Phase Inverter Control.vi
Dependencies
Build Specifications
GPIC Inverter Research Board (169.254.84.198) [Link-local IP Address]
[RT] GPIC 3-Phase Inverter Control.vi
Chassis (sbRIO-9606)
FPGA Target (RIO0, sbRIO-9606, Dev Computer)
[TestBench] Half-Bridge IGBT Control - Starting Point .vi
Enable Stream
Stop
Vu (Instant, RMS)
Modulation Index (0..2)
Waveform Graph
Voltage (V)
Current (A)
Other
Time
Control Sine
Triangle
Top IGBT
Vu_cosim
Vu_measured
Iu_cosim
Iu_measured
Pin (W)
Pout (W)
CrossLatch
Log TDMS?
GPIC Reference Design LabVIEW 2013 SP1.lvproj/My Computer
[FPGA] Half-Bridge Inverter Control.vi
DC+
DC-
GPIC Reference Design LabVIEW 2013 SP1.lvproj/FPGA Target
[FPGA] Half-Bridge Inverter Control.vi Block Diagram *
Sine-Triangle Pulse Width Modulation (PWM)
Deadtime (=delay(s)*dT)
Carrier Freq. (=fc(Hz)*4*dT)
Mod Index (0..2)
Triangle
uTA
uBA
PWM time (s)
s2A / uTOPA
s1A / uBOTA
Sine Function Generator
sinegen freq. (=fo(Hz)*2*dT)
u Shift (pi)
Control Sine
Half-Bridge IGBT Inverter - Multisim - [Half-Bridge IGBT Inverter]
LabVIEW Co-simulation Terminals
Export
Multisim design VI preview
Single Phase, Half-Bridge Inverter
V_DC
XCP2
1 mV/mA
I_DC
Theatsink
Tambient
U6
U2
27.15°C
HeatSink
TOP
T_Tj
T_PswQ
T_PcQ
T_PswD
T_PcD
C2 3300µF
IGBT_DIODE_THERMAL STGIPS10K60A
S1
L1 5.6mH IC=0.0
L_ESR 6.1Ω
XCP3 1 mV/mA
800Ω Rfilter
V_u
I_u
2.05µF IC=0V Cfilter
CF 100µF IC=0.0
R_L 820Ω
C1 3300F
BOTTOM
B_Tj
B_PswQ
B_PcQ
B_PswD
B_PcD
S2
PGND
Hierarchy
Visibility
Project View
Results
Nets
Components
Copper layers
Simulation
For Help, press F1

Efficiency (%)	sim. time (s)	Sim Rate (Real/Sim)
78.25	0.100393	638.554

TOP Tj (C)	Pin (W)	BOT Tj (C)	Pout (W)
43.25	5.82	43.32	4.56

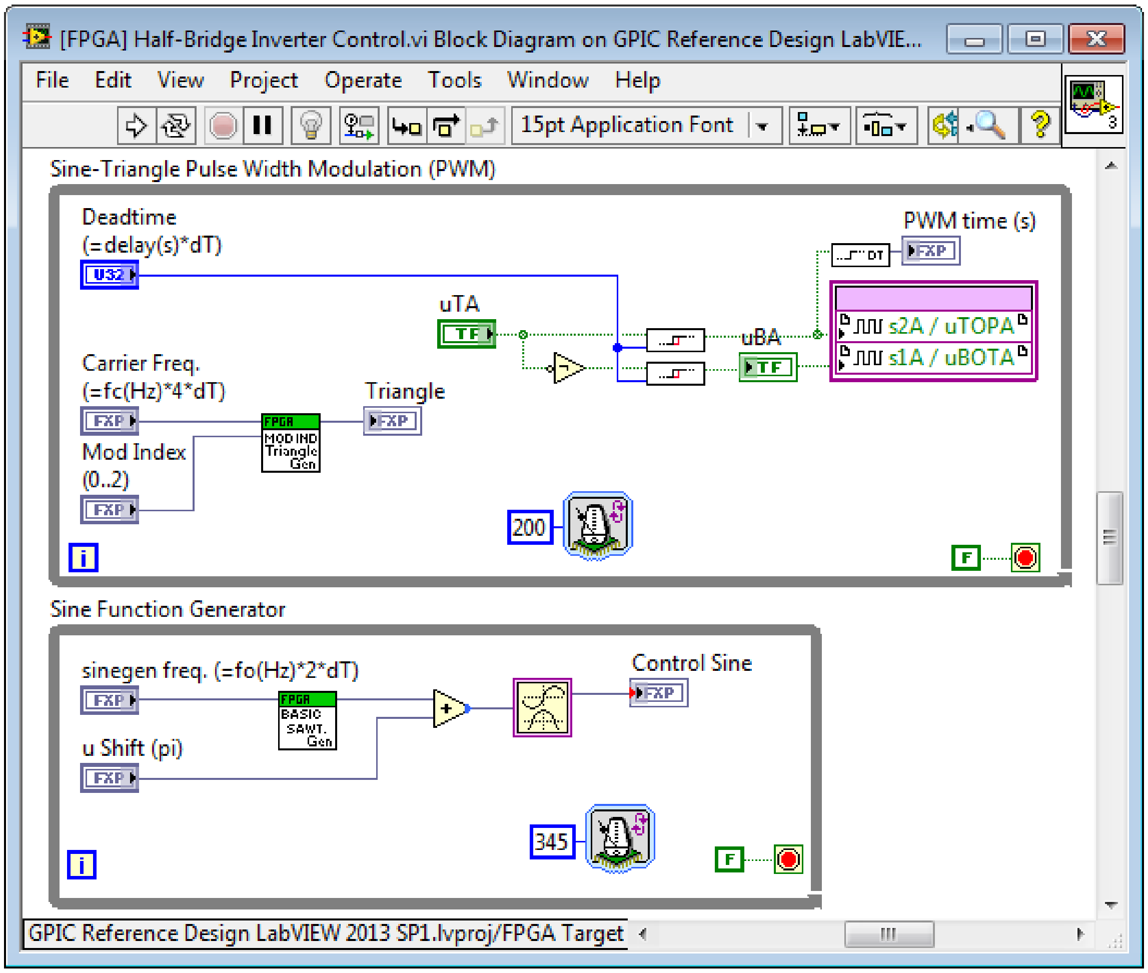
Mod Index (0..2)	sinegen freq. (=fo(Hz)*2*dT)	Carrier Freq. (=fc(Hz)*4*dT)	Deadtime (=delay(s)*dT)
1	0.001035	0.04	1

Control Sine	Triangle	PWM time (s)	u Shift (pi)
0.597351	-0.600006	0.00046085	0.7612

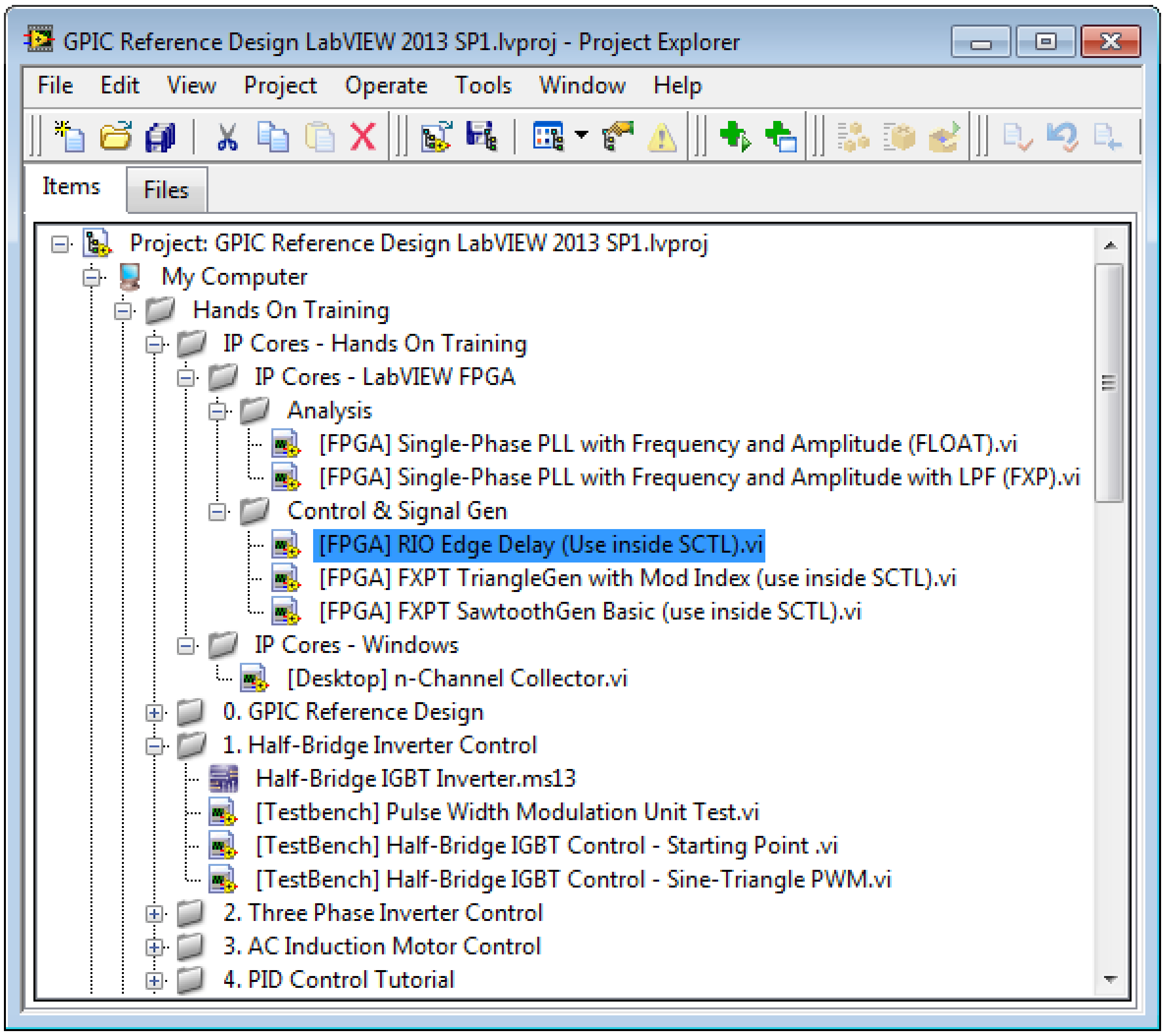
LabVIEW terminal	Positive connection	N...	Direction
Input			
TOP	TOP	0	Input
BOTTOM	BOTTOM	0	Input
V_DC	V_DC	0	Input
Output			
V_u	V_u	0	Output
I_u	I_u	0	Output
I_DC	I_DC	0	Output
Switching_Loss	Switching_Loss	0	Output
Conduction_Loss	Conduction_Loss	0	Output
T_Tj	T_Tj	0	Output
B_Tj	B_Tj	0	Output
Theatsink	Theatsink	0	Output
Tambient	Tambient	0	Output
Generic			

Adding Dead-Time Logic

1. While the circuit appears to be operating correctly, it actually takes some time for the IGBT to open and close. Without dead-time logic, there may be a brief moment in which both the top and bottom switches are closed simultaneously, producing a short-circuit. To avoid this, a LabVIEW FPGA IP core that delays rising edges will be implemented into the FPGA code. In this section, you will add a rising edge delay IP core to produce the FPGA control application shown below.


[FPGA] Half-Bridge Inverter Control.vi Block Diagram on GPIC Reference Design LabVIE...
File Edit View Project Operate Tools Window Help
15pt Application Font
Sine-Triangle Pulse Width Modulation (PWM)
Deadtime
(=delay(s)*dT)
U32
PWM time (s)
FXP
uTA
uBA
s2A / uTOPA
s1A / uBOTA
Carrier Freq.
(=fc(Hz)*4*dT)
Triangle
Mod Index
(0..2)
MOD IND Triangle Gen
200
Sine Function Generator
sinegen freq. (=fo(Hz)*2*dT)
BASIC SAWT. Gen
u Shift (pi)
Control Sine
345
GPIC Reference Design LabVIEW 2013 SP1.lvproj/FPGA Target

2. To do so, navigate in the LabVIEW Project tree to IP Cores>IP Cores – LabVIEW FPGA>Control & Signal Gen. Then click and drag [FPGA] RIO Edge Delay (Use inside SCTL).vi onto your block diagram.


GPIC Reference Design LabVIEW 2013 SP1.lvproj - Project Explorer

File Edit View Project Operate Tools Window Help

Items | Files

```
Project: GPIC Reference Design LabVIEW 2013 SP1.lvproj
  My Computer
    Hands On Training
      IP Cores - Hands On Training
        IP Cores - LabVIEW FPGA
          Analysis
            [FPGA] Single-Phase PLL with Frequency and Amplitude (FLOAT).vi
            [FPGA] Single-Phase PLL with Frequency and Amplitude with LPF (FXP).vi
          Control & Signal Gen
            [FPGA] RIO Edge Delay (Use inside SCTL).vi
            [FPGA] FXPT TriangleGen with Mod Index (use inside SCTL).vi
            [FPGA] FXPT SawtoothGen Basic (use inside SCTL).vi
        IP Cores - Windows
          [Desktop] n-Channel Collector.vi
      0. GPIC Reference Design
      1. Half-Bridge Inverter Control
        Half-Bridge IGBT Inverter.ms13
        [Testbench] Pulse Width Modulation Unit Test.vi
        [TestBench] Half-Bridge IGBT Control - Starting Point .vi
        [TestBench] Half-Bridge IGBT Control - Sine-Triangle PWM.vi
      2. Three Phase Inverter Control
      3. AC Induction Motor Control
      4. PID Control Tutorial
```

3. Hold down **CTRL**, click on the rising edge delay VI, and drag in order to create a duplicate copy of the VI.

4. Delete the wire connecting the top switch, *uTA*, to the FPGA I/O node and the wire connecting the **Not** function to the FPGA I/O Node.

5. Insert the first delay VI between the top switch and the FPGA I/O node by wiring the output of the top-switch to the input of the VI and the output of the VI to the input of the FPGA I/O node labeled *s2A / uTOPA*.

Note: There are two inputs to the rising edge delay VI. Be sure to wire the Boolean switch to the top input labeled *Input* and not to the bottom input labeled *Deadtime (iterations)*.

6. Insert the second delay VI between the output of the Not function and the input of the FPGA I/O node labeled *s1A / uBOTA*. Be sure to re-wire the bottom-switch indicator if it has been disconnected.

7. Wire the **Deadtime (=delay(s)*dT)** control to the bottom input of both delay VIs.

8. The VI will ensure that for the number of cycles indicated by the deadtime control, the closing of the switch is delayed. On the other hand, the command to open a switch will not be delayed.

9. To test whether your application is working properly, navigate back to the **[TestBench] Half-Bridge IGBT Control - Starting Point.vi**. Run the VI and observe the output on the graph as well as the output voltage dial.

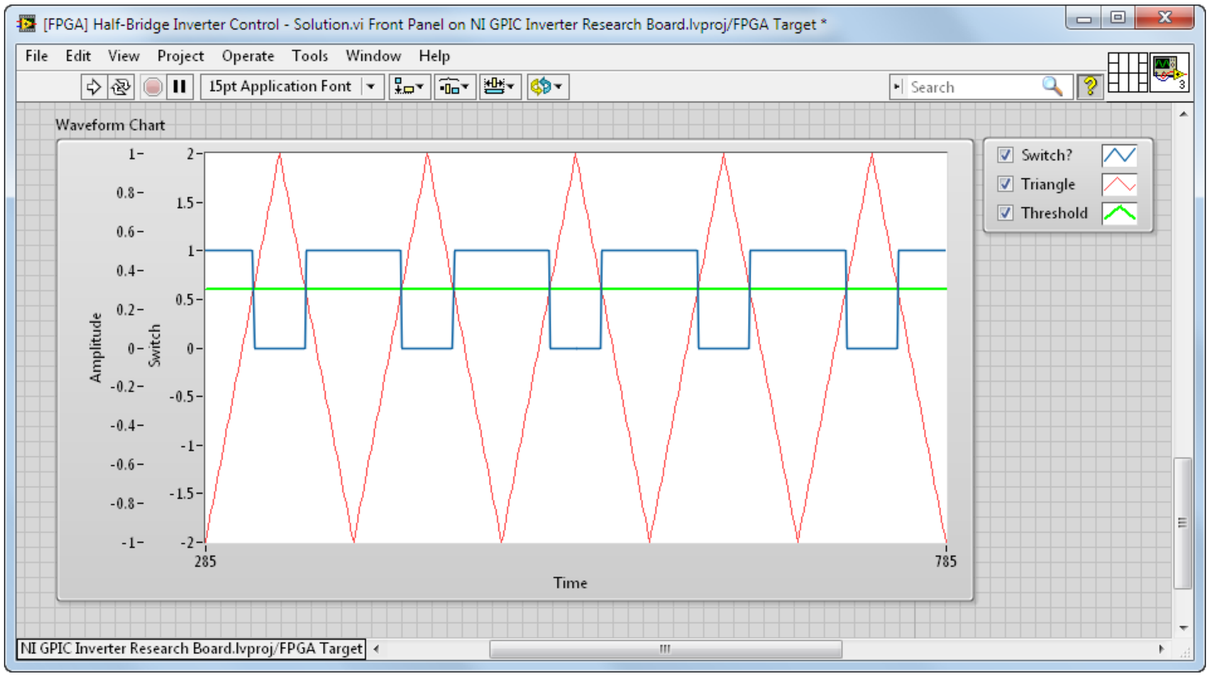
Exercise 5 | Sine-Triangle PWM

Summary

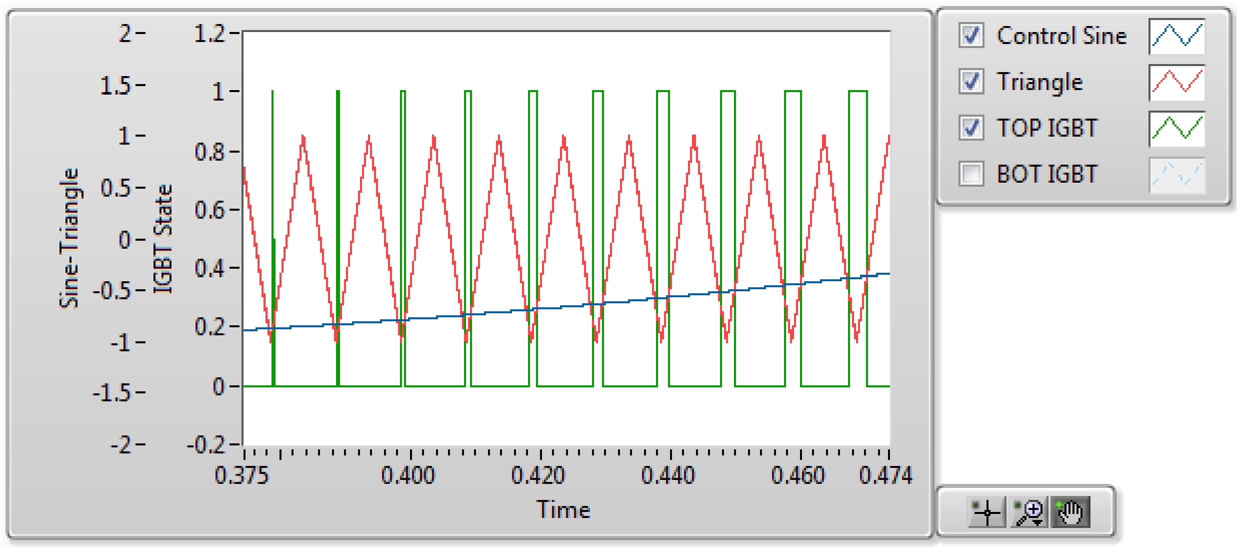
Now that the application can switch the IGBTs without creating a short circuit, it is time to automate the pulse width modulation using a popular technique called Sine-Triangle PWM.

Background

The next step towards developing your inverter will be to remove the need for manual operation. This will be accomplished through a technique called pulse width modulation. PWM uses pulses to turn the switches on and off. The time duration between pulses allows for the control of the output signal. This exercise will involve the use of two types of PWM. The first and more basic type of PWM is triangle PWM. Whenever the Triangle surpasses a threshold, the switch is thrown. The following image shows an example where the threshold has been set to .3. Notice that the *Switch?* signal goes high for the duration that the triangle signal is above the threshold.

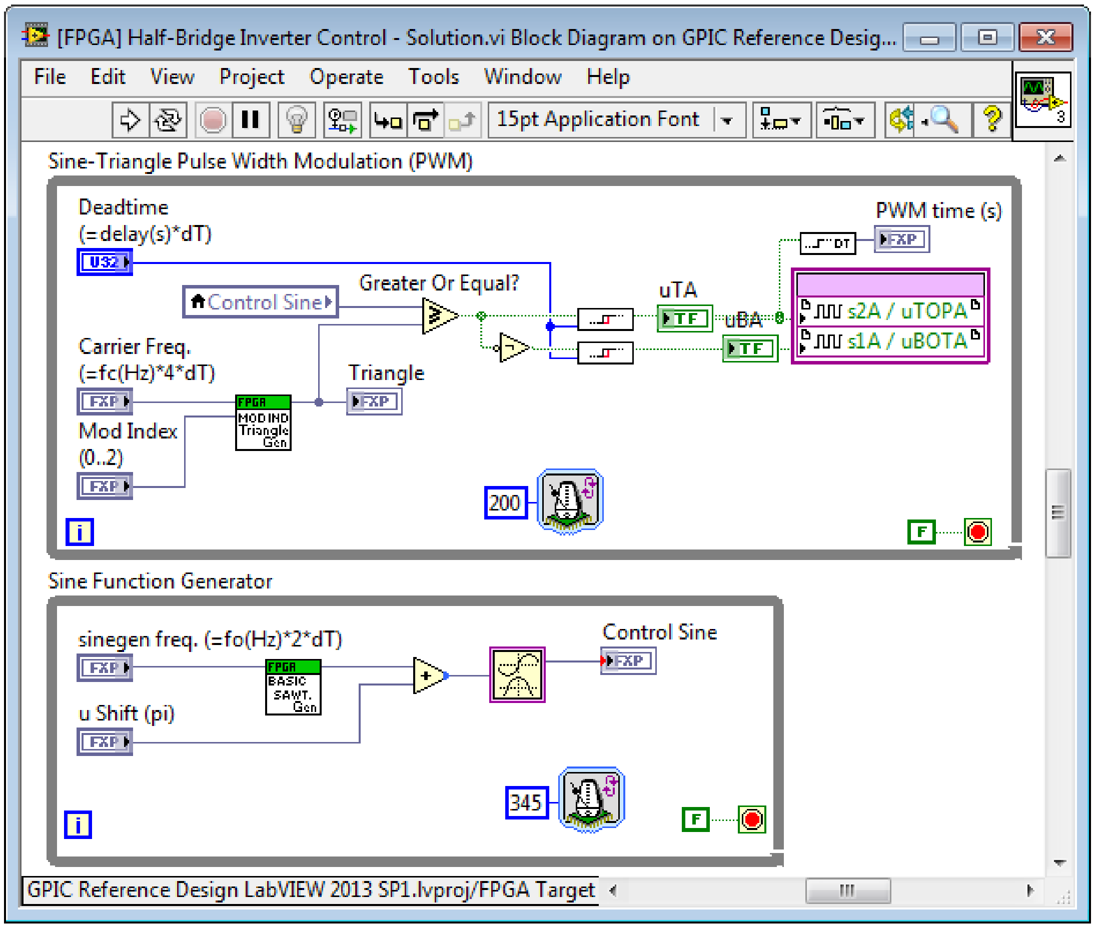

[FPGA] Half-Bridge Inverter Control - Solution.vi Front Panel on NI GPIC Inverter Research Board.lvproj/FPGA Target *
File Edit View Project Operate Tools Window Help
15pt Application Font
Search
Waveform Chart
Amplitude
Switch
Time
285
785
Switch?
Triangle
Threshold
NI GPIC Inverter Research Board.lvproj/FPGA Target

The second type of PWM that you will implement is sine-triangle. Sine-triangle PWM is exactly like triangle PWM except now the threshold has been replaced with a sinusoidal signal. Whenever the triangle signal surpasses the value of the sinusoidal signal, the switch is thrown. Notice that this leads to a variable pulse duration.



Sine-Triangle PWM

1. Open the TestBench and FPGA VIs that were saved in exercise 2. In this section, you will add a triangle waveform generator IP core and comparison function to produce the FPGA control application shown below.


[FPGA] Half-Bridge Inverter Control - Solution.vi Block Diagram on GPIC Reference Desig...

2. To do so, the switches must now be configured to modulate their switching based on the signal produced by the triangle and control sine generators. The goal is to produce a pulse width modulated (PWM) version of the control sine wave with a PWM switching frequency equal to the carrier frequency of the triangle waveform.

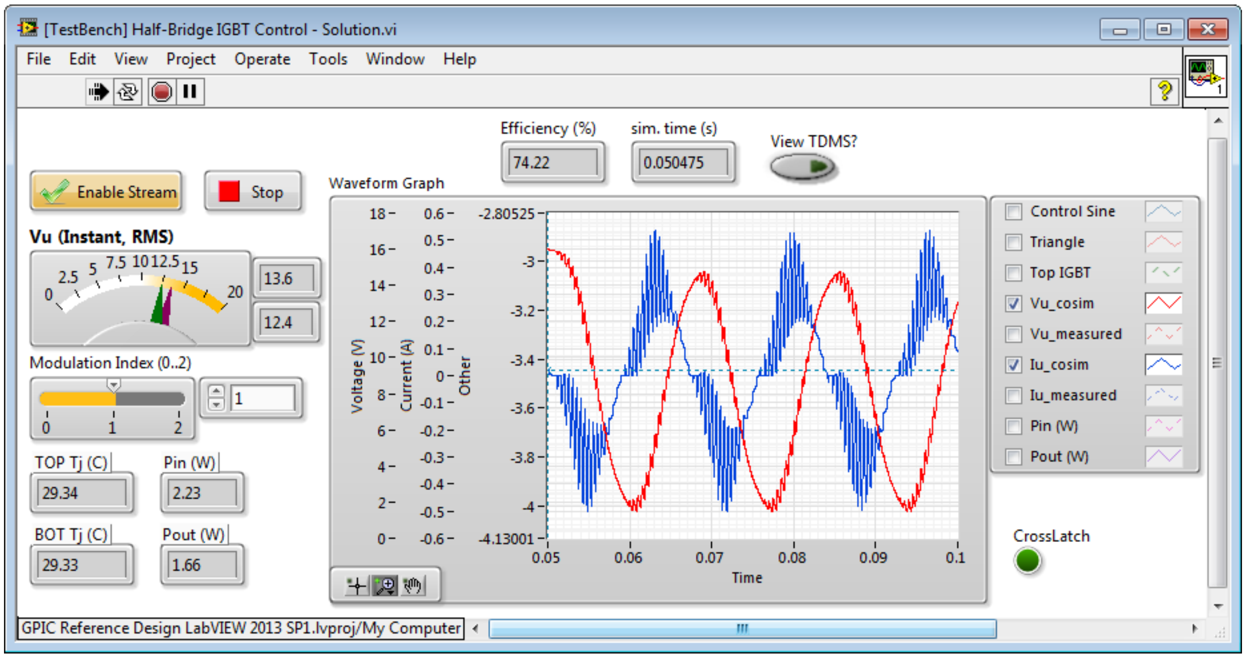
3. Right-click on the block diagram to open the **functions palette.** Navigate to the programming tab, select Comparison, and then left-click the **Greater or Equal?** function. Place this function onto the block diagram inside the **Sine-Triangle Pulse Width Modulation (PWM)** while loop.

4. Wire the **Triangle** output to the bottom input of the **Greater or Equal?** function, labeled *y*.

5. **Right-click** on **Control Sine** indicator and then select **Create** followed by **Local Variable**. Place the local variable in the Sine-Triangle while loop. Then right-click on the local variable and select **Change to Read**.

6. By following these steps, you have successfully created a custom FPGA-based sine-triangle based PWM algorithm. Navigate back to the **[TestBench] Half-Bridge IGBT Control - Starting Point.vi**. Run this VI and observe the output on the graph as well as the output voltage dial.

7. When you vary the modulation index, how does the behavior change? What is the value of the phase current, **Iu_cosim**, when the modulation index is 0? What happens to the phase voltage and current when the modulation index is greater than 1?


[TestBench] Half-Bridge IGBT Control - Solution.vi
File Edit View Project Operate Tools Window Help
Enable Stream
Stop
Vu (Instant, RMS)
Modulation Index (0..2)
Waveform Graph
View TDMS?
Voltage (V)
Current (A)
Other
Time
CrossLatch
GPIC Reference Design LabVIEW 2013 SP1.lvproj/My Computer

Efficiency (%): 74.22
sim. time (s): 0.050475
Vu (Instant, RMS): 13.6, 12.4
Modulation Index (0..2): 1
TOP Tj (C): 29.34
Pin (W): 2.23
BOT Tj (C): 29.33
Pout (W): 1.66

☐	Control Sine
☐	Triangle
☐	Top IGBT
☑	Vu_cosim
☐	Vu_measured
☑	Iu_cosim
☐	Iu_measured
☐	Pin (W)
☐	Pout (W)

Exercise 6 | Comparing Simulated vs. Experimental Results

Summary

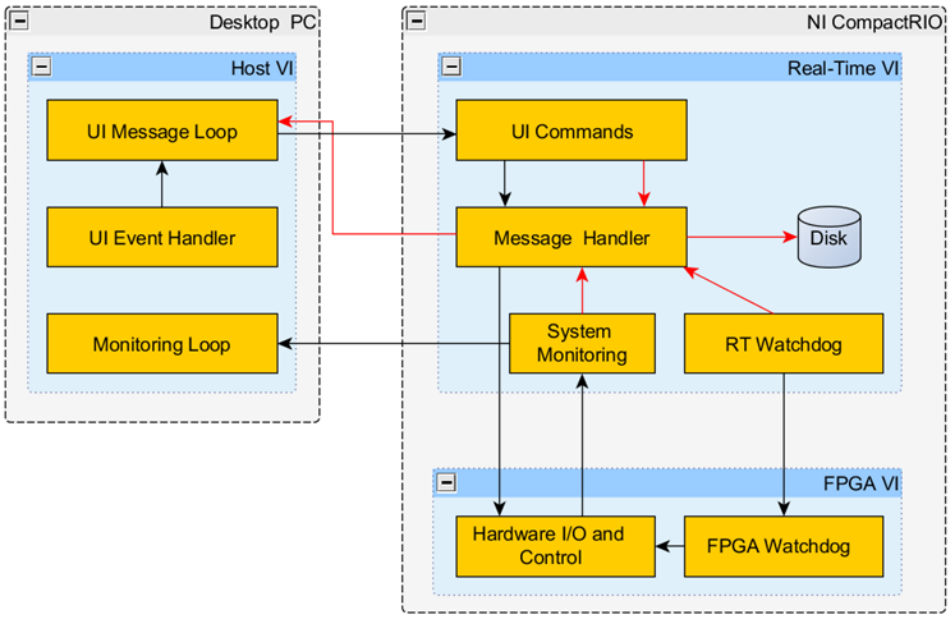
The final step in our simulation and testing process will be to compare the simulated results to the results as measured on the NI General Purpose Inverter Controller (GPIC) board. In order to compare the two it will be necessary to connect to the embedded real-time processor and FPGA and connect to the network stream. In this exercise you will:

- Configure a connection between the host computer and the GPIC board
- Enable the network stream to obtain data from the GPIC board
- Plot the waveforms for simulated and measured results and compare

Background

Although this exercise does not go into the specifics of how data is transferred between the FPGA board, the Real-Time processor, and the host computer, it is worth mentioning the basic communication architecture.

Hardware I/O and control is performed on the FPGA board. Information is then exchanged directly with the Real-Time application running on the RIO. The Real-Time application is then able to perform several time-sensitive functions before exchanging information with the host application running on a computer.

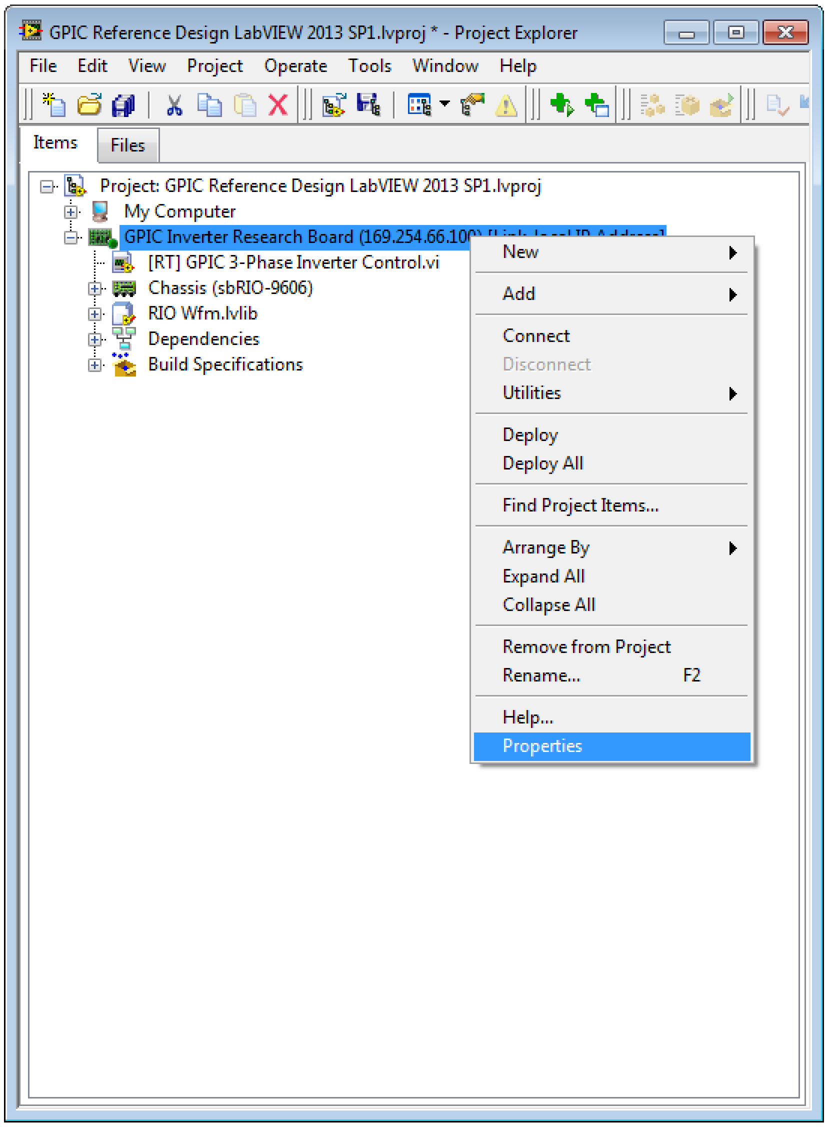


In this exercise, each target will be performing specific tasks. The FPGA will be controlling the inverter circuit and passing on sensor data. The Real-Time application will be prepare the input gathered from the FPGA board to be streamed to the host computer. The host application will perform the inverter board simulation while gathering and displaying the data obtained from the Real-Time application. As part of this exercise, you will deploy and configure the Real-Time application as well as the host application in order to obtain measurement data.

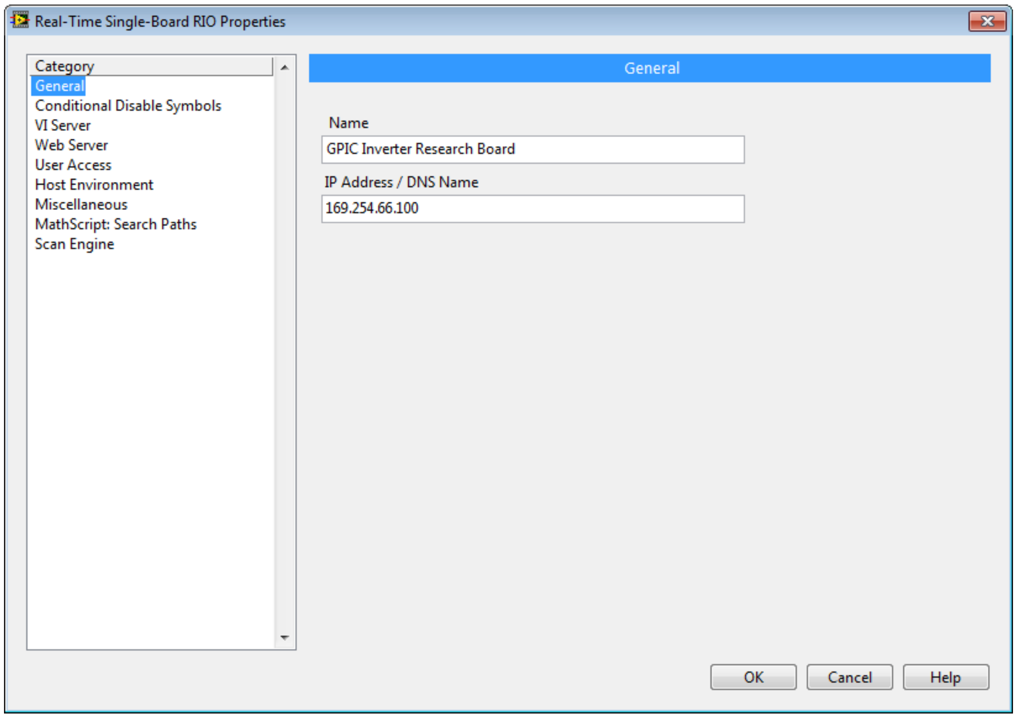
Configuring Network Stream

1. In order to obtain measurements from the hardware, it will be necessary to establish a connection with the Single-Board RIO GPIC. Look at your sbRIO-9606 control board and see if the LED is blinking on and off once per second. If so, this means the LabVIEW Real-Time (and FPGA) applications are running and waiting for a network connection—skip to step 8 below.

2. If your control board LED is not blinking on and off once per second, navigate back to the NI GPIC Inverter Research Board project window. **Right-click** the **GPIC Inverter Research Board** target and select **Properties**.



3. This will cause the properties window to pop-up. Select the **General** category and enter in the IP address that has been assigned to your single-board RIO control system under the **IP Address / DNS Name** label. Select **OK** when finished.



Real-Time Single-Board RIO Properties

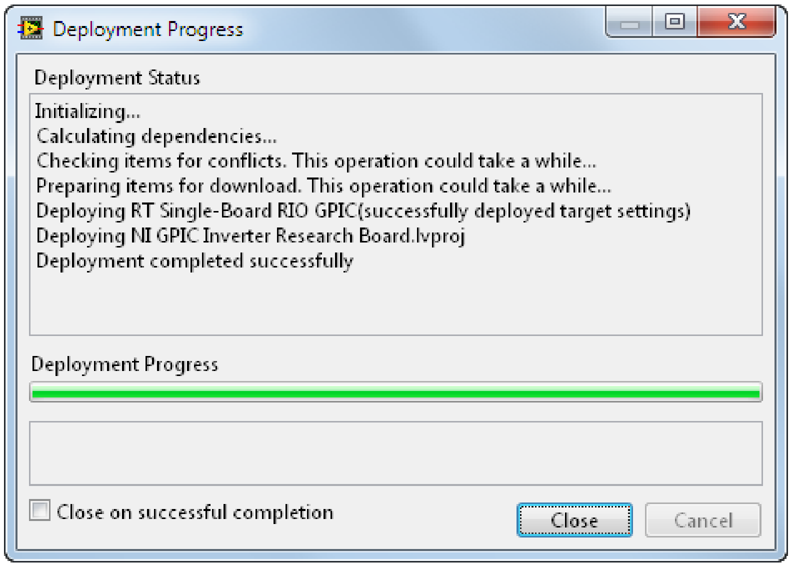
Category: General, Conditional Disable Symbols, VI Server, Web Server, User Access, Host Environment, Miscellaneous, MathScript: Search Paths, Scan Engine

General

Name: GPIC Inverter Research Board

IP Address / DNS Name: 169.254.66.100

4. It is time to connect to the GPIC control system. Once again, right click on the **RT Single-Board RIO GPIC** target. This time, select **Connect**. A dialog window will appear and display information about the deployment. Once deployment process has finished, press **Close**.



Deployment Progress

Deployment Status

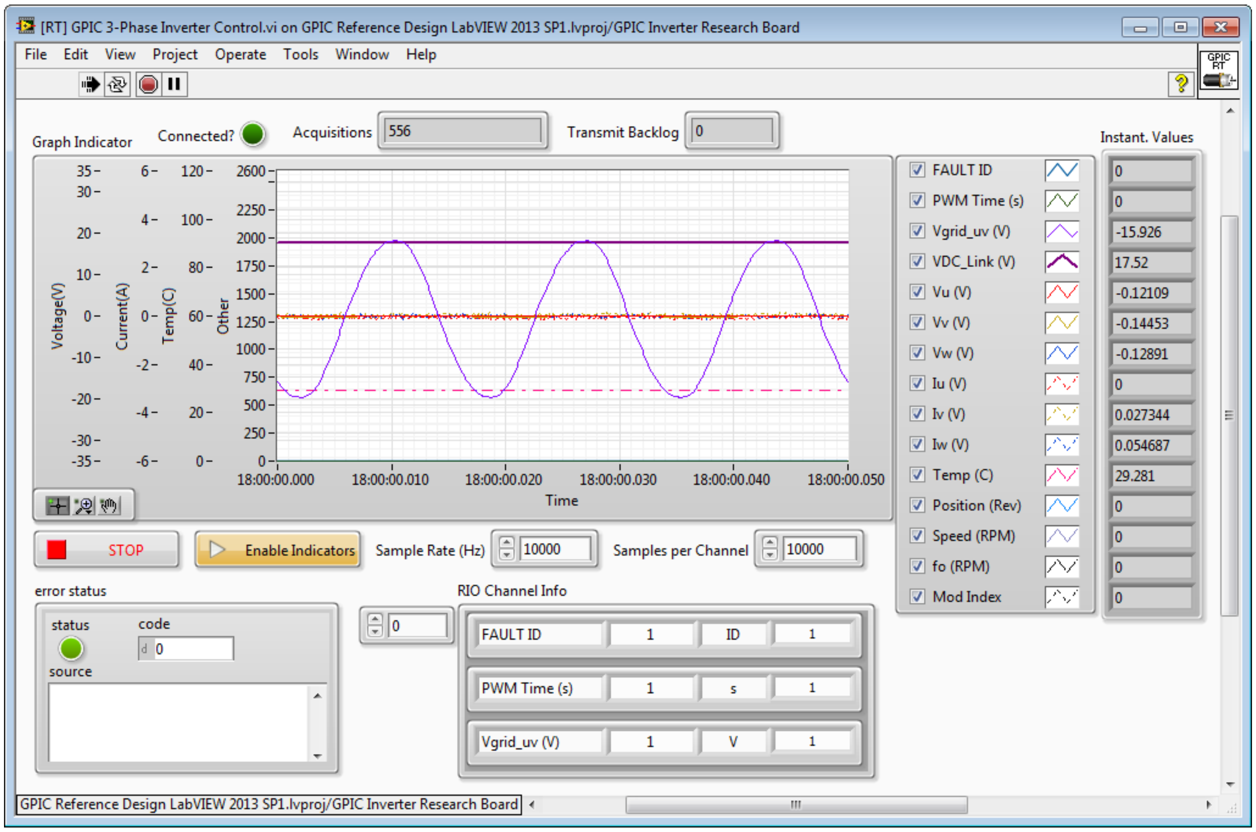
```
Initializing...
Calculating dependencies...
Checking items for conflicts. This operation could take a while...
Preparing items for download. This operation could take a while...
Deploying RT Single-Board RIO GPIC(successfully deployed target settings)
Deploying NI GPIC Inverter Research Board.lvproj
Deployment completed successfully
```

Deployment Progress

☐ Close on successful completion

5. The next step is to run the Real-time host VI in order to download and run the FPGA control system and obtain a measurement data stream from the GPIC system. Expand the **RT Single-Board RIO GPIC** target and open **[RT] GPIC 3-Phase Inverter Control.vi**.

6. Run the VI and click **Enable Indicators** to view the waveforms and instantaneous values.


[RT] GPIC 3-Phase Inverter Control.vi on GPIC Reference Design LabVIEW 2013 SP1.lvproj/GPIC Inverter Research Board
File Edit View Project Operate Tools Window Help
Graph Indicator
Voltage(V)
Current(A)
Temp(C)
Other
Time
STOP
Enable Indicators
error status
RIO Channel Info
GPIC Reference Design LabVIEW 2013 SP1.lvproj/GPIC Inverter Research Board

Connected? ●
Acquisitions 556
Transmit Backlog 0

	Instant. Values
☑ FAULT ID	0
☑ PWM Time (s)	0
☑ Vgrid_uv (V)	-15.926
☑ VDC_Link (V)	17.52
☑ Vu (V)	-0.12109
☑ Vv (V)	-0.14453
☑ Vw (V)	-0.12891
☑ Iu (V)	0
☑ Iv (V)	0.027344
☑ Iw (V)	0.054687
☑ Temp (C)	29.281
☑ Position (Rev)	0
☑ Speed (RPM)	0
☑ fo (RPM)	0
☑ Mod Index	0

Sample Rate (Hz) 10000
Samples per Channel 10000

status ● code 0
source

RIO Channel Info: 0

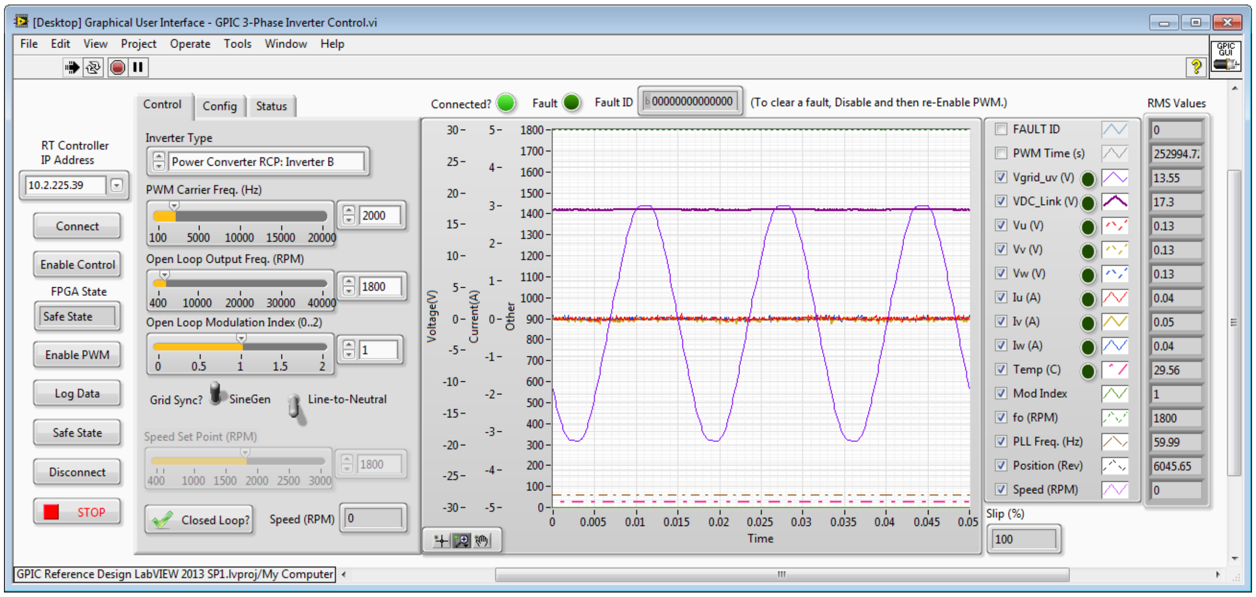
FAULT ID	1	ID	1
PWM Time (s)	1	s	1
Vgrid_uv (V)	1	V	1

7. Click **Enable Indicators** again to turn off the local display, since it consumes microprocessor resources. Leave the VI running in the background.

Desktop Graphical User Interface

8. Navigate back to the project window. Under **My Computer**, open **[Desktop] Graphical User Interface - GPIC 3-Phase Inverter Control.vi**. Run the VI. This application searches the network for NI LabVIEW RIO targets and provides a network user interface with live scope display, configurable control settings and data logging capabilities.

9. Using the control labeled **RT Controller IP Address,** select the IP address of your GPIC system. If you are not already connected, hit the Connect button. The **Connected?** Indicator should light up.



[Desktop] Graphical User Interface - GPIC 3-Phase Inverter Control.vi

RT Controller IP Address: 10.2.225.39

FPGA State: Safe State

Inverter Type: Power Converter RCP: Inverter B

PWM Carrier Freq. (Hz): 2000

Open Loop Output Freq. (RPM): 1800

Open Loop Modulation Index (0..2): 1

Grid Sync? SineGen / Line-to-Neutral

Speed Set Point (RPM): 1800

Closed Loop? Speed (RPM): 0

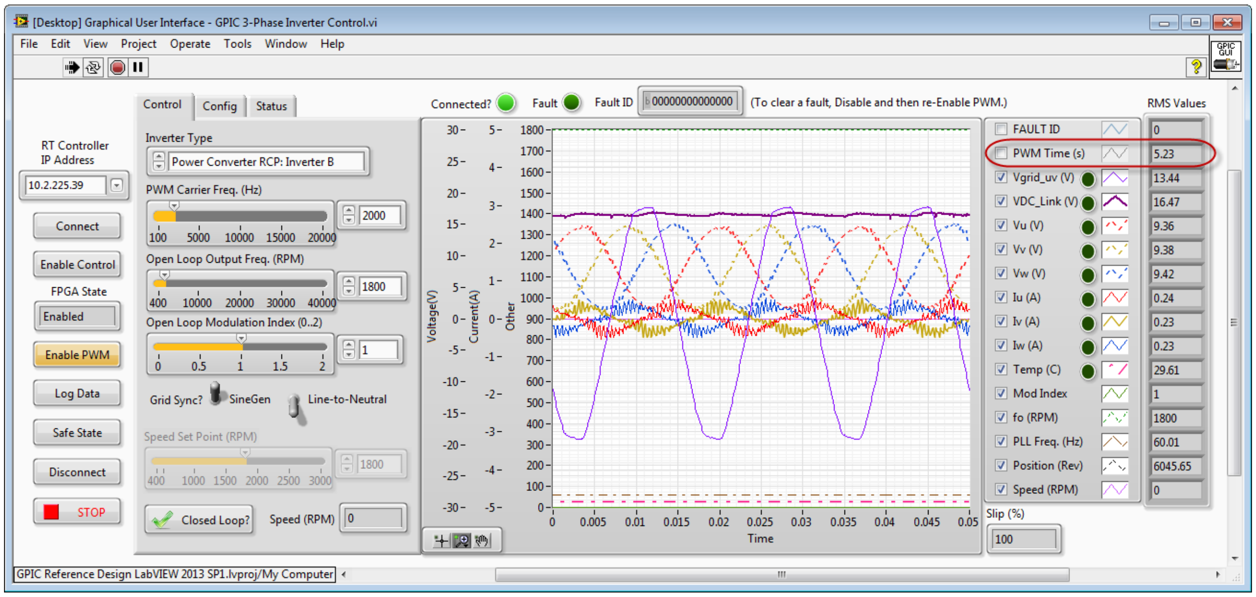
Connected? Fault Fault ID: 0000000000000 (To clear a fault, Disable and then re-Enable PWM.)

	RMS Values
FAULT ID	0
PWM Time (s)	252994.7
Vgrid_uv (V)	13.55
VDC_Link (V)	17.3
Vu (V)	0.13
Vv (V)	0.13
Vw (V)	0.13
Iu (A)	0.04
Iv (A)	0.05
Iw (A)	0.04
Temp (C)	29.56
Mod Index	1
fo (RPM)	1800
PLL Freq. (Hz)	59.99
Position (Rev)	6045.65
Speed (RPM)	0

Slip (%): 100

GPIC Reference Design LabVIEW 2013 SP1.lvproj/My Computer

10. Next, press **Enable Control.** The **FPGA State** indicator should indicate **Enabled**. Now click **Enable PWM**. The 3-phase inverter should begin operating, with the waveforms displayed on the graphical user interface.



[Desktop] Graphical User Interface - GPIC 3-Phase Inverter Control.vi

RT Controller IP Address: 10.2.225.39

FPGA State: Enabled

Inverter Type: Power Converter RCP: Inverter B

PWM Carrier Freq. (Hz): 2000

Open Loop Output Freq. (RPM): 1800

Open Loop Modulation Index (0..2): 1

Grid Sync? SineGen / Line-to-Neutral

Speed Set Point (RPM): 1800

Closed Loop? Speed (RPM): 0

Connected? Fault Fault ID: 0000000000000 (To clear a fault, Disable and then re-Enable PWM.)

	RMS Values
FAULT ID	0
PWM Time (s)	5.23
Vgrid_uv (V)	13.44
VDC_Link (V)	16.47
Vu (V)	9.36
Vv (V)	9.38
Vw (V)	9.42
Iu (A)	0.24
Iv (A)	0.23
Iw (A)	0.23
Temp (C)	29.61
Mod Index	1
fo (RPM)	1800
PLL Freq. (Hz)	60.01
Position (Rev)	6045.65
Speed (RPM)	0

Slip (%): 100

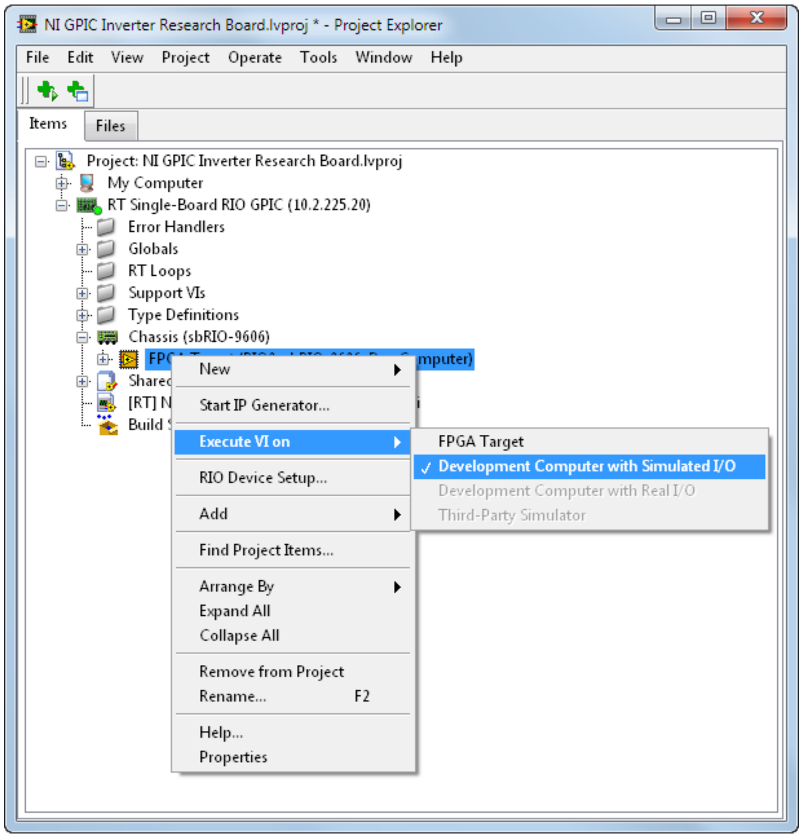
GPIC Reference Design LabVIEW 2013 SP1.lvproj/My Computer

11. Note that the **PWM Time (s)** counter is reset and begins counting up when the PWM is enabled. When PWM Time (s) is reset, the user interface application automatically captures a 1-2 second waveform that can be used for comparing simulated versus measured results. Click the **Enable PWM** button again to disable pulse width

modulation. Note the clicking sound caused by the opening of the pre-charge contactor. The inverter output stops but the network stream data is still available.

Comparing Simulation vs. Measurement Waveforms

1. Now that the network stream data has been established, it is time to compare the simulation results to the measurement results obtained from the GPIC power converter. Navigate back to the project window. Expand the GPIC Inverter Research Board target, then Chassis. Right-Click the FPGA Target and select Execute VI on. Verify that Development Computer with Simulated I/O is checked. If it is not, select it now.

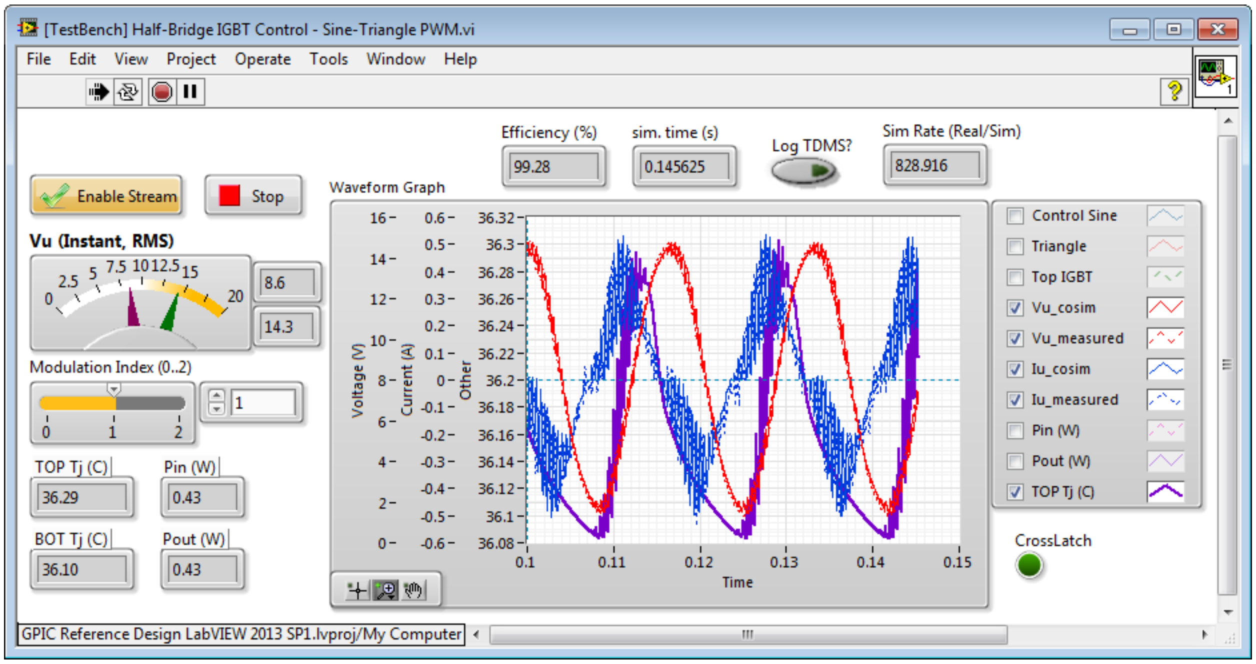


2. Underneath My Computer, expand Hands on Workshop, then 1. Half-Bridge Inverter Inverter Control. Open the testbench application you built in the previous exercise, or select and open [TestBench] Half-Bridge IGBT Control - Sine-Triangle PWM.vi.

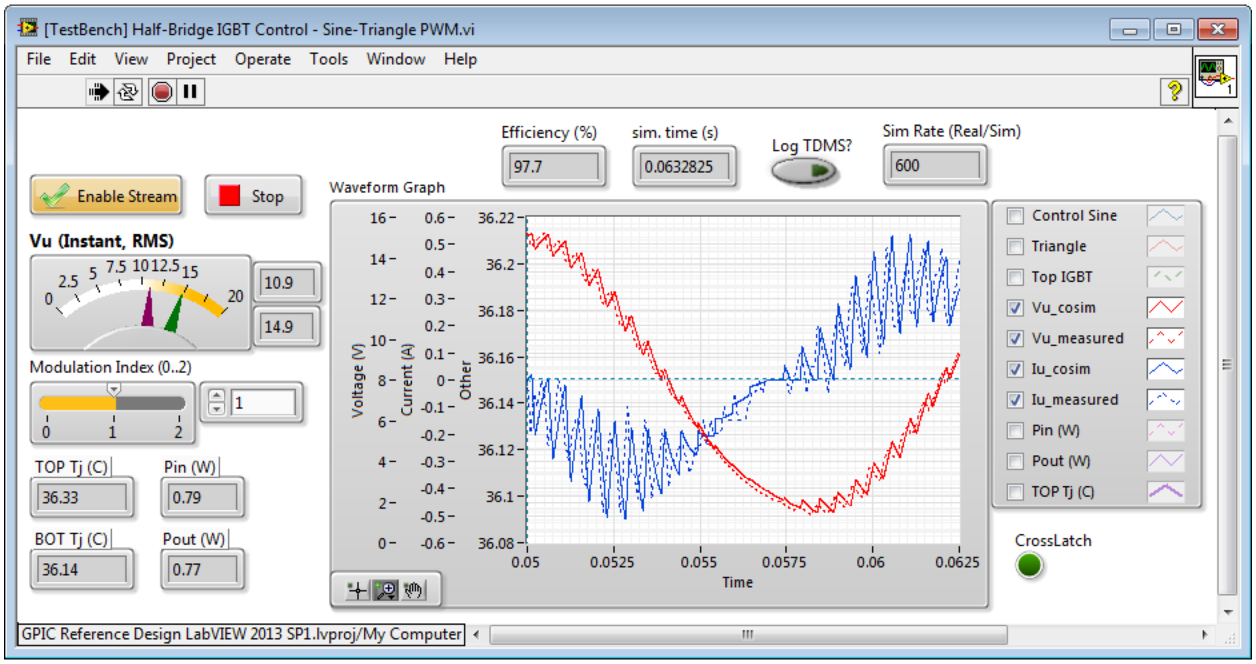
3. To the right of the graph, there is a list of available waveforms to plot. Make sure there is a mark to the left of the label Vu_measured, Iu_measured, Vu_cosim and Iu_cosim. This will cause the measured output voltage and current and the simulated output voltage and current to be shown on the graph.

4. Click the Enable Stream button. (If Enable Stream is false, previously saved experimental waveform data will play back during the simulation.)

5. Run the VI. After a few moments, the VI will begin to function and the graph will update. After the simulated system reaches steady state, how well do the simulated waveforms match up with the measured waveforms?

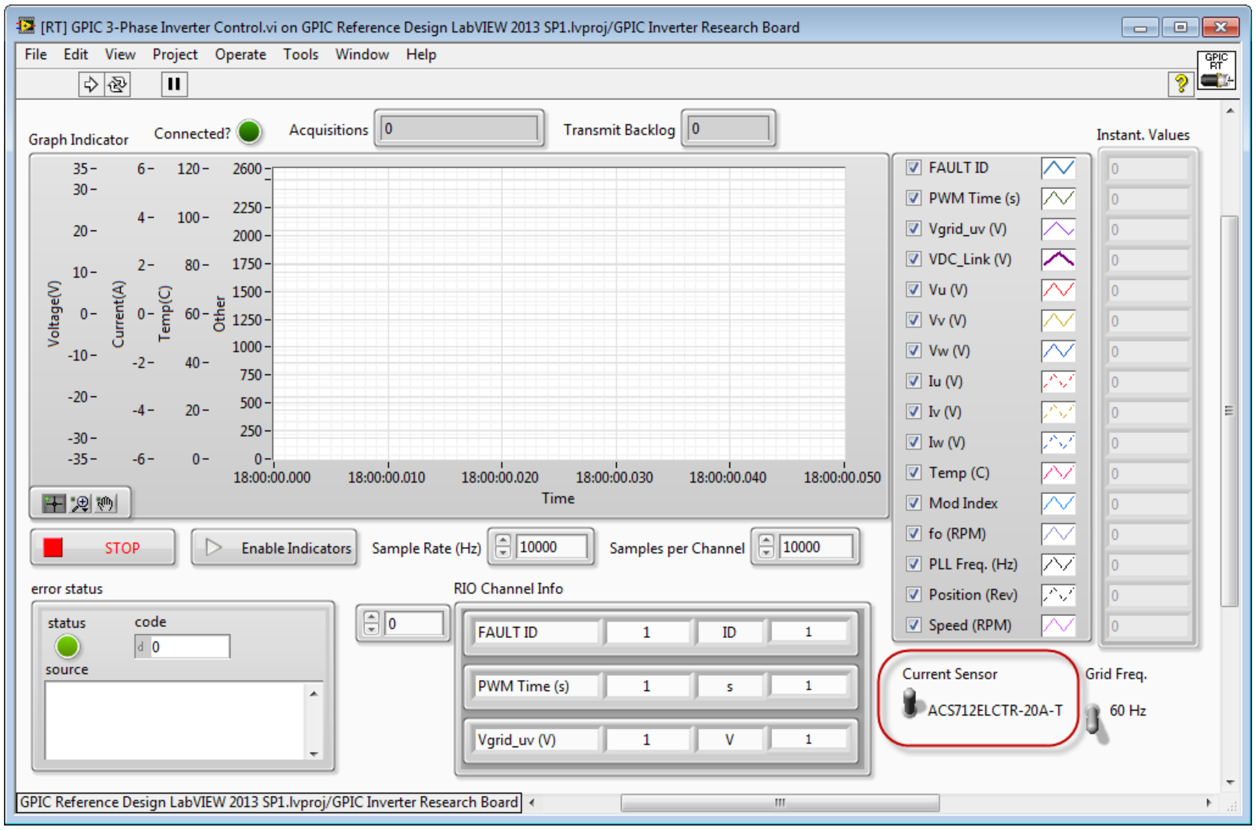


Efficiency (%): 99.28
sim. time (s): 0.145625
Sim Rate (Real/Sim): 828.916
Vu (Instant, RMS): 8.6, 14.3
Modulation Index (0..2): 1
TOP Tj (C): 36.29
Pin (W): 0.43
BOT Tj (C): 36.10
Pout (W): 0.43



Efficiency (%): 97.7
sim. time (s): 0.0632825
Sim Rate (Real/Sim): 600
Vu (Instant, RMS): 10.9, 14.9
Modulation Index (0..2): 1
TOP Tj (C): 36.33
Pin (W): 0.79
BOT Tj (C): 36.14
Pout (W): 0.77

Note: If the amplitude of your current waveforms do not match, the current sensors on your GPIC power electronics research board may be different. If so, navigate to the LabVIEW Real-Time application, click the STOP button to stop it. Then select the other current sensor setting and restart the application.



[RT] GPIC 3-Phase Inverter Control.vi on GPIC Reference Design LabVIEW 2013 SP1.lvproj/GPIC Inverter Research Board
File Edit View Project Operate Tools Window Help
Graph Indicator
Voltage(V)
Current(A)
Temp(C)
Other
Time
Instant. Values
STOP
Enable Indicators
error status
RIO Channel Info
GPIC Reference Design LabVIEW 2013 SP1.lvproj/GPIC Inverter Research Board

Connected? ●
Acquisitions 0
Transmit Backlog 0

Sample Rate (Hz) 10000
Samples per Channel 10000

status ● code 0
source

FAULT ID	1	ID	1
PWM Time (s)	1	s	1
Vgrid_uv (V)	1	V	1

✓ FAULT ID	0
✓ PWM Time (s)	0
✓ Vgrid_uv (V)	0
✓ VDC_Link (V)	0
✓ Vu (V)	0
✓ Vv (V)	0
✓ Vw (V)	0
✓ Iu (V)	0
✓ Iv (V)	0
✓ Iw (V)	0
✓ Temp (C)	0
✓ Mod Index	0
✓ fo (RPM)	0
✓ PLL Freq. (Hz)	0
✓ Position (Rev)	0
✓ Speed (RPM)	0

Current Sensor ACS712ELCTR-20A-T
Grid Freq. 60 Hz

6. When you wish to stop the program, first press the **Stop** button on the **[TestBench] Half-Bridge IGBT Control - Solution.vi** application.

Module 2:

3-Phase Inverter Control

Exercise 1 | Understanding the Phase-Lock Loop (PLL)

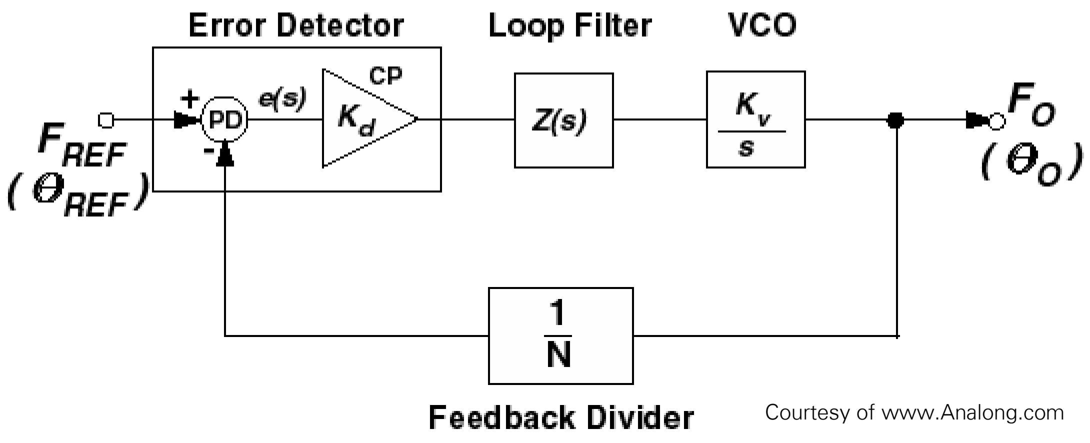
Summary

In this exercise, you will be creating your own 3-phase phase-locked loop (PLL) in order to understand how one might be implemented in a 3-phase inverter. Phase locked loops are important to grid tied power because in order to supply power to a grid, the inverter output must be phase locked to the grid frequency. Tasks you will complete this exercise include:

- Implementing a 3-phase PLL
- Simulating a PLL application using the desktop execution node
- Experimenting by varying the input and outputs to your PLL application

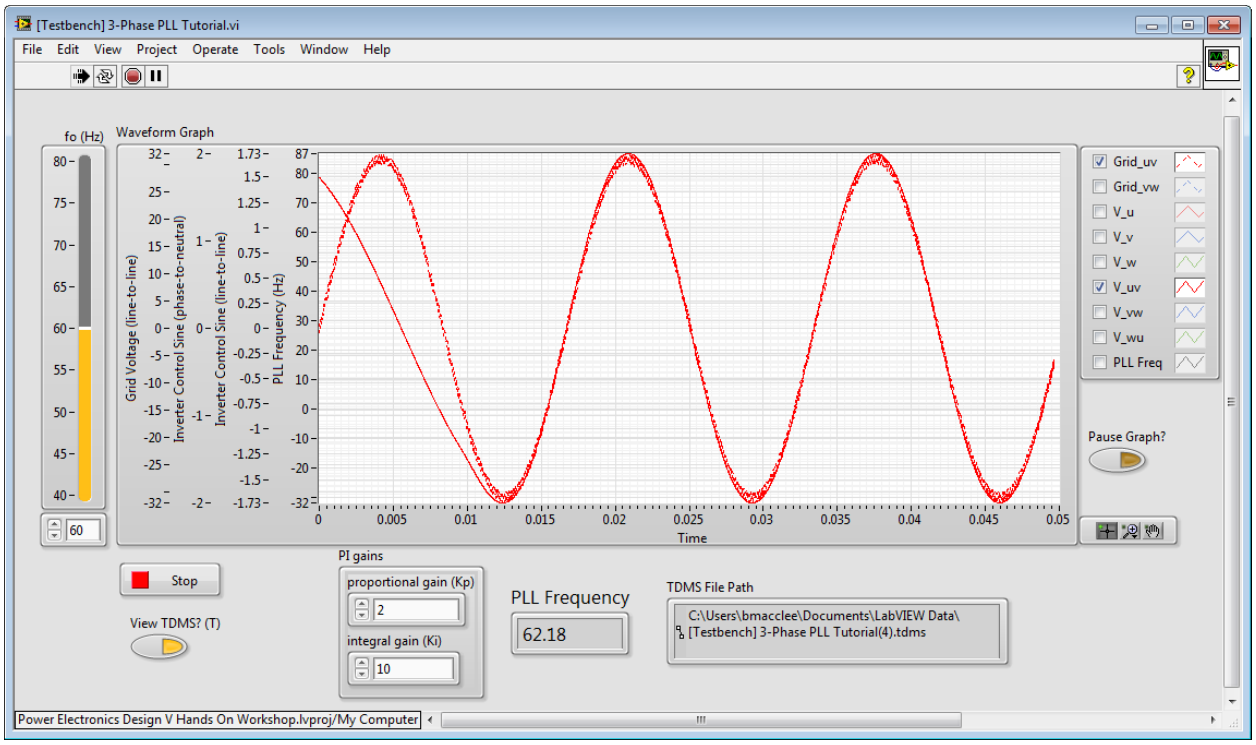
Background

Phase-locked loops are a form of feedback control. A PLL produces an oscillating output, which begins operating with a set frequency. An input is then passed to the PLL to serve as a reference point. The PLL adjusts the output to then closely match the input phase. The longer the PLL runs, the more closely the PLL output is able to track the input, given that the input is stable.



Courtesy of www.Analong.com

PLLs are useful in a variety of synchronous applications. Such applications include clock multipliers, demodulation, and power grid phase locking. Since this workshop centers on inverters, we will be discussing the PLL in the context of grid-tied inverters. In order to connect an inverter to the grid, the inverter output's phase must be within a few degrees of the grid signal phase. This is where the PLL will be applied. By using the grid signal as a reference, the PLL is capable of generating a waveform which will be phase-locked to the grid signal. This output can then be used to drive PWM in order to generate the inverter output.


[Testbench] 3-Phase PLL Tutorial.vi
File Edit View Project Operate Tools Window Help
Waveform Graph
Grid Voltage (line-to-line)
Inverter Control Sine (phase-to-neutral)
Inverter Control Sine (line-to-line)
PLL Frequency (Hz)
Time
Pause Graph?
Stop
View TDMS? (T)

fo (Hz): 60

☑ Grid_uv
☐ Grid_vw
☐ V_u
☐ V_v
☐ V_w
☑ V_uv
☐ V_vw
☐ V_wu
☐ PLL Freq

PI gains
proportional gain (Kp): 2
integral gain (Ki): 10

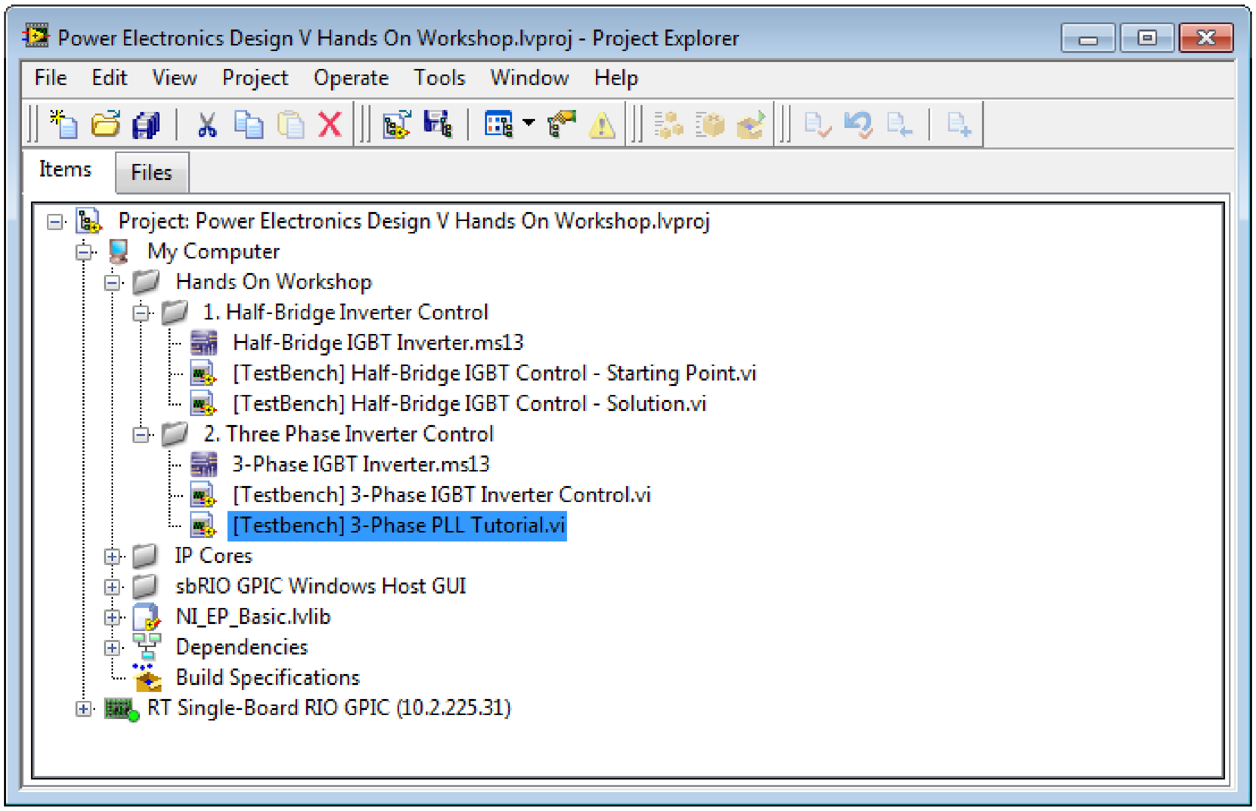
PLL Frequency: 62.18

TDMS File Path: C:\Users\bmacclee\Documents\LabVIEW Data\[Testbench] 3-Phase PLL Tutorial(4).tdms

Power Electronics Design V Hands On Workshop.lvproj/My Computer

Running the 3-Phase PLL Tutorial

9. In the project window, expand out **My Computer**, **Hands on Workshop**, and then **2. Three Phase Inverter Control**. There will be a Multisim file and three VIs. Double-click on **[Testbench] 3-Phase PLL Tutorial.vi**.

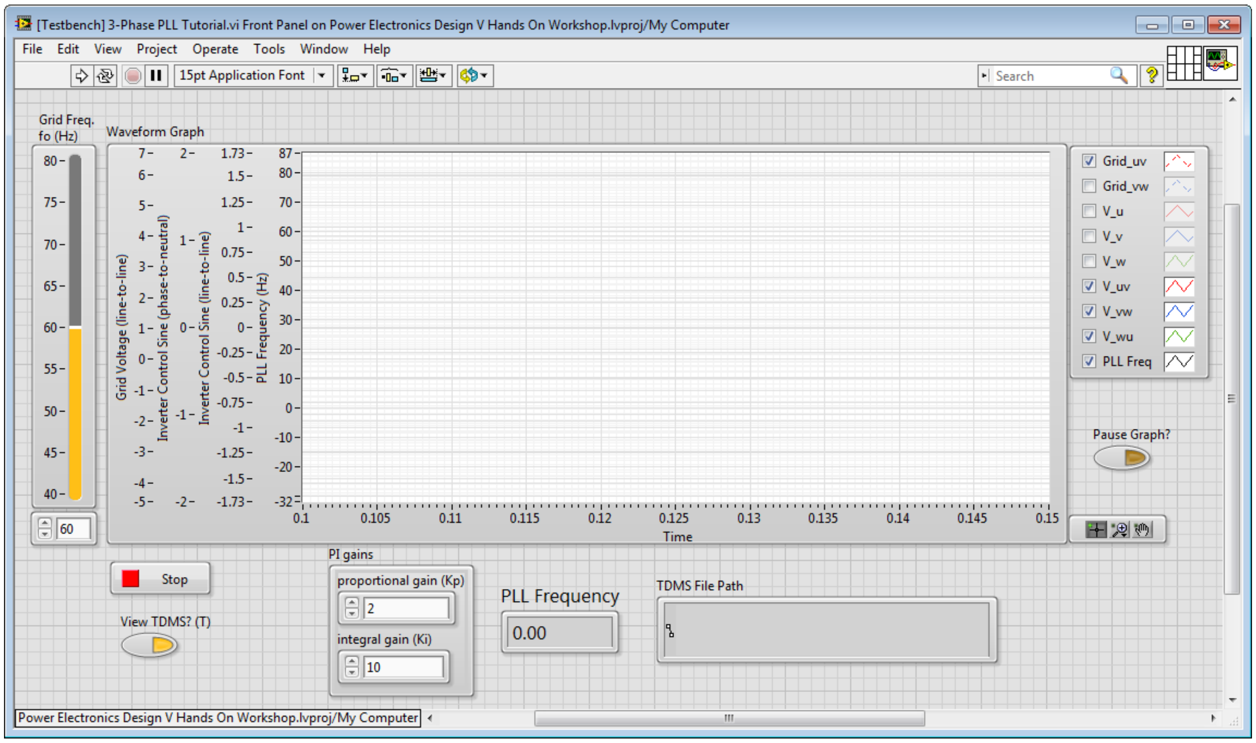
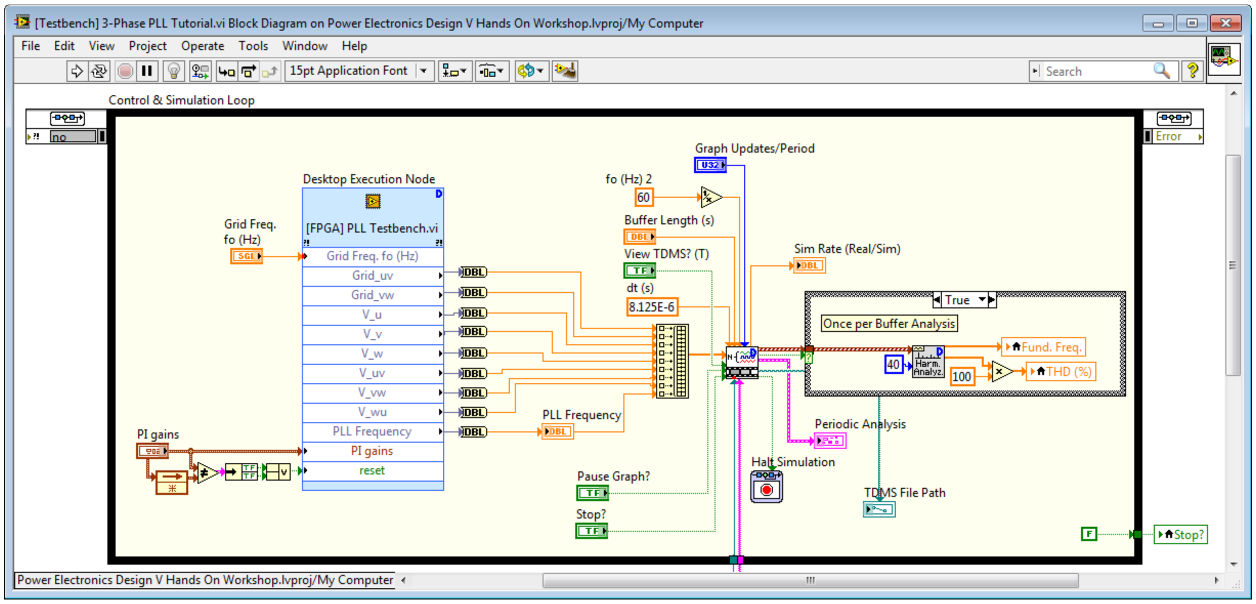

Power Electronics Design V Hands On Workshop.lvproj - Project Explorer

File Edit View Project Operate Tools Window Help

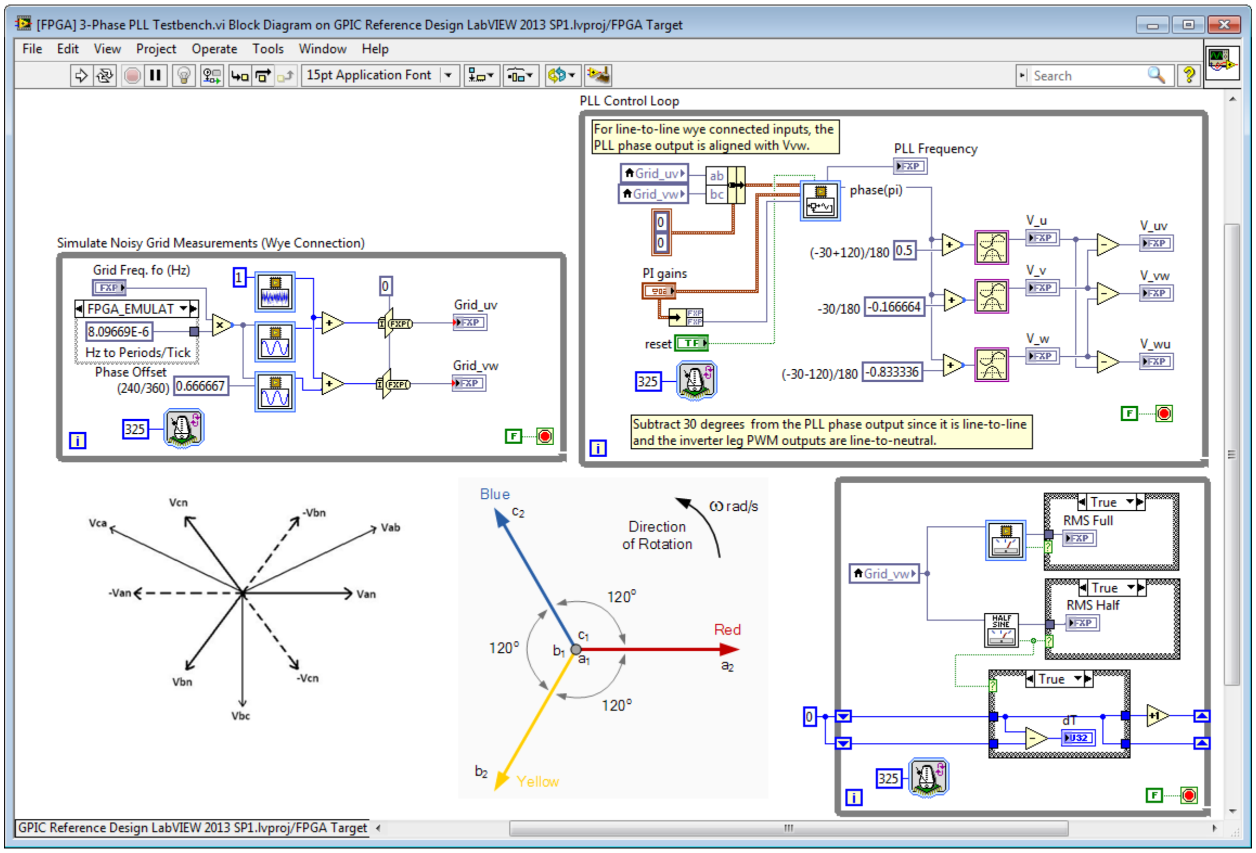
Items Files

```
Project: Power Electronics Design V Hands On Workshop.lvproj
  My Computer
    Hands On Workshop
      1. Half-Bridge Inverter Control
        Half-Bridge IGBT Inverter.ms13
        [TestBench] Half-Bridge IGBT Control - Starting Point.vi
        [TestBench] Half-Bridge IGBT Control - Solution.vi
      2. Three Phase Inverter Control
        3-Phase IGBT Inverter.ms13
        [Testbench] 3-Phase IGBT Inverter Control.vi
        [Testbench] 3-Phase PLL Tutorial.vi
    IP Cores
    sbRIO GPIC Windows Host GUI
    NI_EP_Basic.lvlib
    Dependencies
    Build Specifications
  RT Single-Board RIO GPIC (10.2.225.31)
```

10. The VI may take a few moments to open. This is the front panel of the PLL application you will be running. You will notice a vertical slider on the left to control frequency of the simulated power grid waveforms.

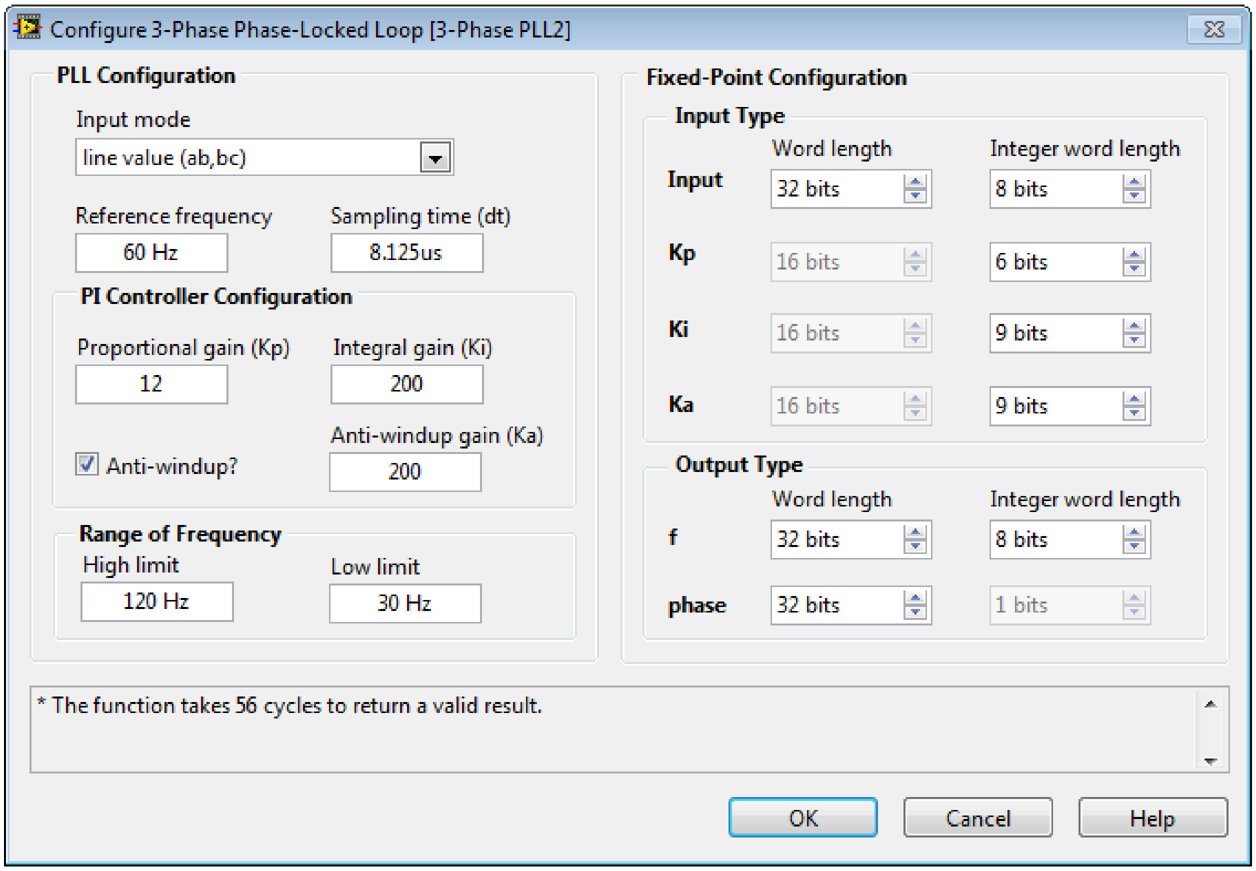

[Testbench] 3-Phase PLL Tutorial.vi Front Panel on Power Electronics Design V Hands On Workshop.lvproj/My Computer
File Edit View Project Operate Tools Window Help
15pt Application Font
Search
Grid Freq. fo (Hz)
Waveform Graph
Grid Voltage (line-to-line)
Inverter Control Sine (phase-to-neutral)
Inverter Control Sine (line-to-line)
PLL Frequency (Hz)
Time
Grid_uv
Grid_vw
V_u
V_v
V_w
V_uv
V_vw
V_wu
PLL Freq
Pause Graph?
60
Stop
View TDMS? (T)
PI gains
proportional gain (Kp)
2
integral gain (Ki)
10
PLL Frequency
0.00
TDMS File Path
Power Electronics Design V Hands On Workshop.lvproj/My Computer


11. Navigate to the block diagram and double click the **Desktop Execution Node** to open up the FPGA application named **[FPGA] PLL Testbench.vi**.


[FPGA] 3-Phase PLL Testbench.vi Block Diagram on GPIC Reference Design LabVIEW 2013 SP1.lvproj/FPGA Target
File Edit View Project Operate Tools Window Help
15pt Application Font
Search
PLL Control Loop
For line-to-line wye connected inputs, the PLL phase output is aligned with Vvw.
PLL Frequency
phase(pi)
Grid_uv
Grid_vw
PI gains
reset
(-30+120)/180 0.5
-30/180 -0.166664
(-30-120)/180 -0.833336
V_u
V_v
V_w
V_uv
V_vw
V_wu
Subtract 30 degrees from the PLL phase output since it is line-to-line and the inverter leg PWM outputs are line-to-neutral.
Simulate Noisy Grid Measurements (Wye Connection)
Grid Freq. fo (Hz)
FPGA_EMULAT
8.09669E-6
Hz to Periods/Tick
Phase Offset (240/360) 0.666667
Grid_uv
Grid_vw
325
Vcn
-Vbn
Vca
Vab
-Van
Van
Vbn
-Vcn
Vbc
Blue
Direction of Rotation
ω rad/s
120°
Red
Yellow
True
RMS Full
RMS Half
dT
GPIC Reference Design LabVIEW 2013 SP1.lvproj/FPGA Target

12. Open up the block diagram and take note of the items that have already been placed. The left loop, labeled **Simulate Noisy Grid Measurements (Wye Connection)**, contains a sine generator, similar to the one used in the half-bridge FPGA controller. The right loop, labeled **PLL Control Loop**, contains the LabVIEW FPGA 3-Phase PLL IP core.

13. Place the block diagram into edit mode by pressing **CTRL+M**. **Double-click** on the **3-Phase PLL** function to view its configuration. Click **OK** when you are finished.


Configure 3-Phase Phase-Locked Loop [3-Phase PLL2]

PLL Configuration

Input mode: line value (ab,bc)

Reference frequency: 60 Hz

Sampling time (dt): 8.125us

PI Controller Configuration

Proportional gain (Kp): 12

Integral gain (Ki): 200

☑ Anti-windup?

Anti-windup gain (Ka): 200

Range of Frequency

High limit: 120 Hz

Low limit: 30 Hz

Fixed-Point Configuration

Input Type

	Word length	Integer word length
Input	32 bits	8 bits
Kp	16 bits	6 bits
Ki	16 bits	9 bits
Ka	16 bits	9 bits

Output Type

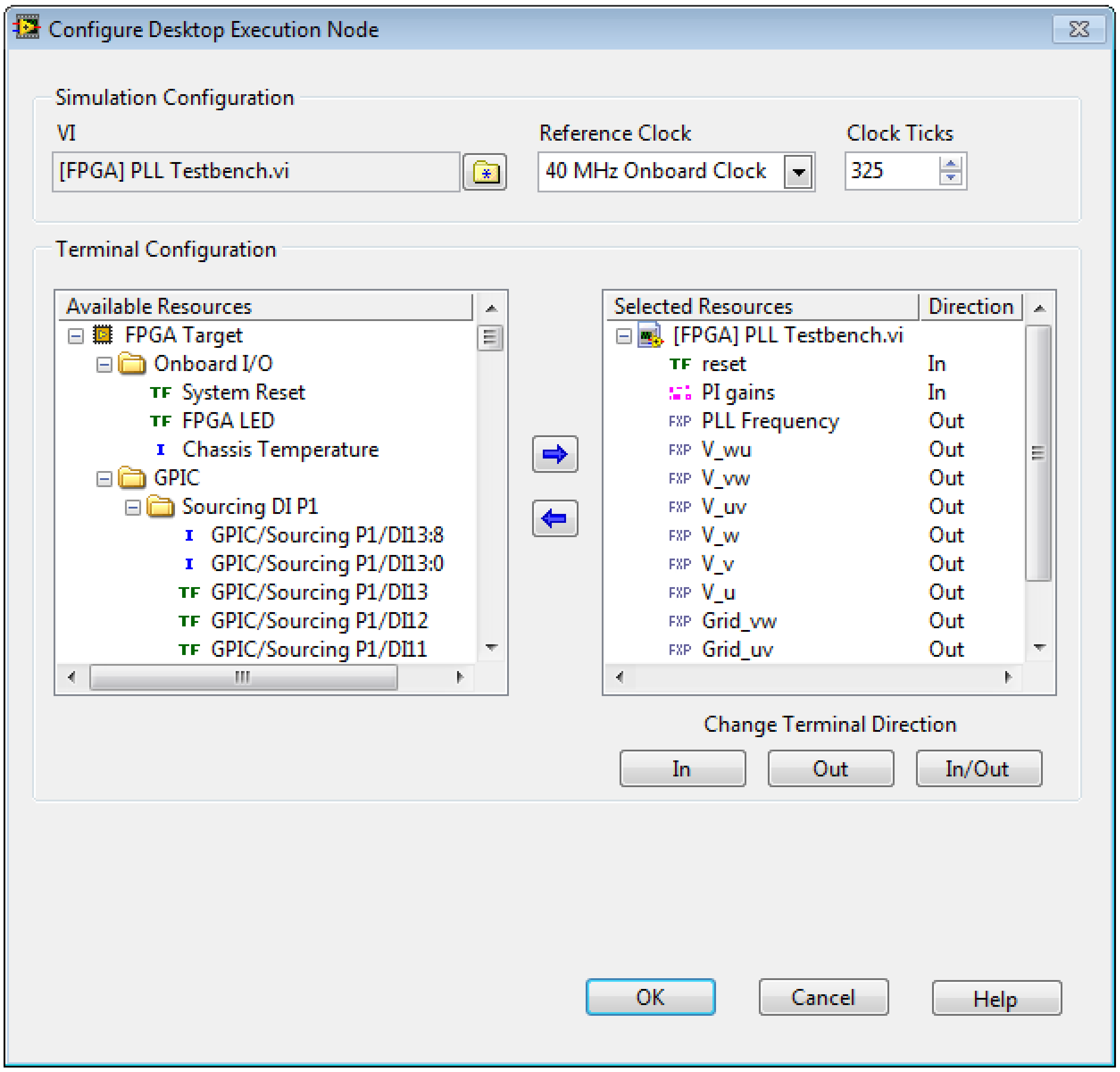
	Word length	Integer word length
f	32 bits	8 bits
phase	32 bits	1 bits

* The function takes 56 cycles to return a valid result.

14. In a similar manner, double click the IP cores in the grid simulation loop to the left to view their configuration. Click OK when you are finished.

Testing the FPGA PLL Application

1. Now that you understand the FPGA based grid simulation and 3-phase PLL application, it is time to test it. Navigate back to, or open, [Testbench] 3-Phase PLL Tutorial.vi Once open, navigate to the block diagram. This VI has been preconfigured for you to test your application. Right-click on the Desktop Execution Node and select Configure Desktop Execution Node.
2. Note that the Clock Ticks setting is set to 325 ticks. Thus, every 325 ticks of the 40 MHz FPGA clock, every 8.125 microseconds of simulated time, the Desktop Execution Node will return the register values for the items chosen in the Selected Resources dialogue. Note that the FPGA resources you can choose from include both front panel controls and indicators as well as I/O resources. In this case, no changes to the configuration are necessary, so click Cancel to exit


Configure Desktop Execution Node

Simulation Configuration

VI: [FPGA] PLL Testbench.vi

Reference Clock: 40 MHz Onboard Clock

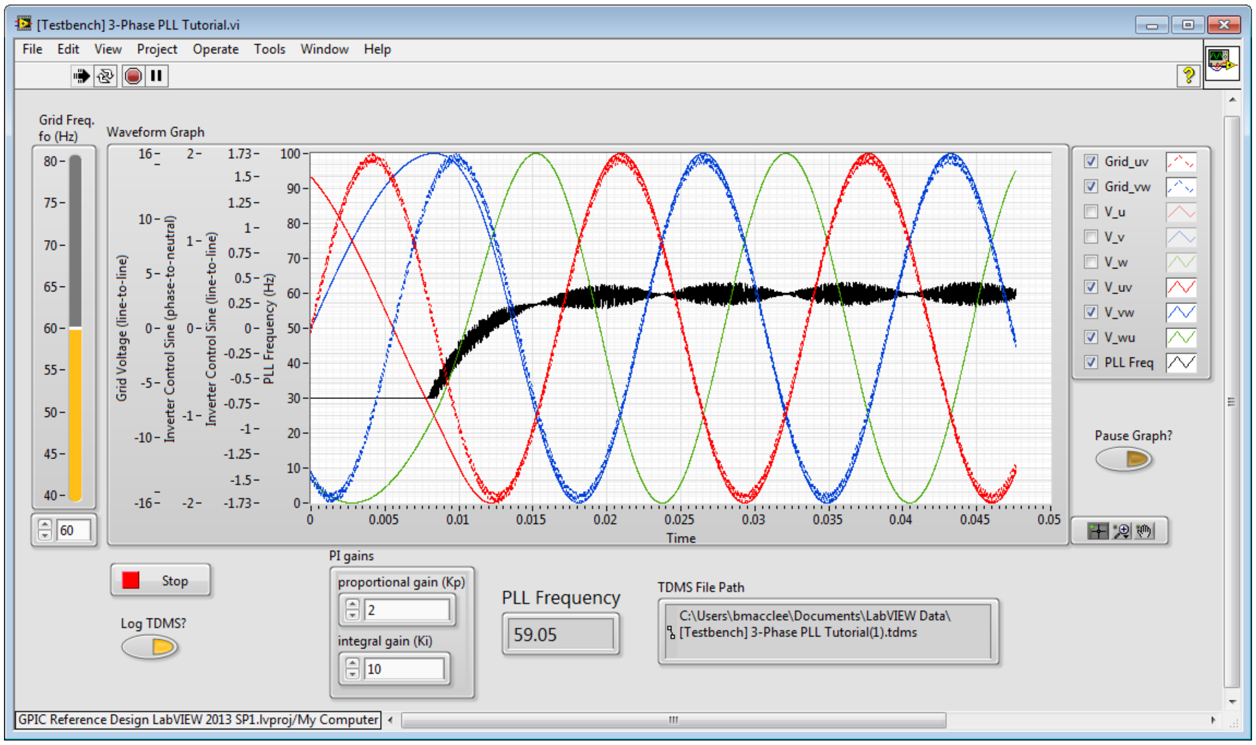
Clock Ticks: 325

Terminal Configuration

Available Resources
- FPGA Target
  - Onboard I/O
    - TF System Reset
    - TF FPGA LED
    - I Chassis Temperature
  - GPIC
    - Sourcing DI P1
      - I GPIC/Sourcing P1/DI13:8
      - I GPIC/Sourcing P1/DI13:0
      - TF GPIC/Sourcing P1/DI13
      - TF GPIC/Sourcing P1/DI12
      - TF GPIC/Sourcing P1/DI11

Selected Resources	Direction
[FPGA] PLL Testbench.vi	
TF reset	In
PI gains	In
FXP PLL Frequency	Out
FXP V_wu	Out
FXP V_vw	Out
FXP V_uv	Out
FXP V_w	Out
FXP V_v	Out
FXP V_u	Out
FXP Grid_vw	Out
FXP Grid_uv	Out

3. Run the TestBench application. Notice how PLL Frequency is initially 30 Hz and takes a half cycle to begin tracking the noisy simulated grid waveforms. You can see the frequency of PLL in the indicator labeled **PLL Frequency**. Due to the noisy nature of the 3-phase grid waveforms and the aggressive tuning of the PLL proportional-integral (PI) control system, the PLL Frequency oscillates around the correct grid frequency.



[Testbench] 3-Phase PLL Tutorial.vi

Grid Freq. fo (Hz): 60

proportional gain (Kp): 2

integral gain (Ki): 10

PLL Frequency: 59.05

TDMS File Path: C:\Users\bmacclee\Documents\LabVIEW Data\[Testbench] 3-Phase PLL Tutorial(1).tdms

GPIC Reference Design LabVIEW 2013 SP1.lvproj/My Computer

4. While the application is running, note that the red **V_uv** waveform is phase aligned with the red **Grid_uv** waveform. Meanwhile, the blue **V_vw** and green **V_wu** waveforms are shifted by +240/-120 degrees and +120 degrees respectively.

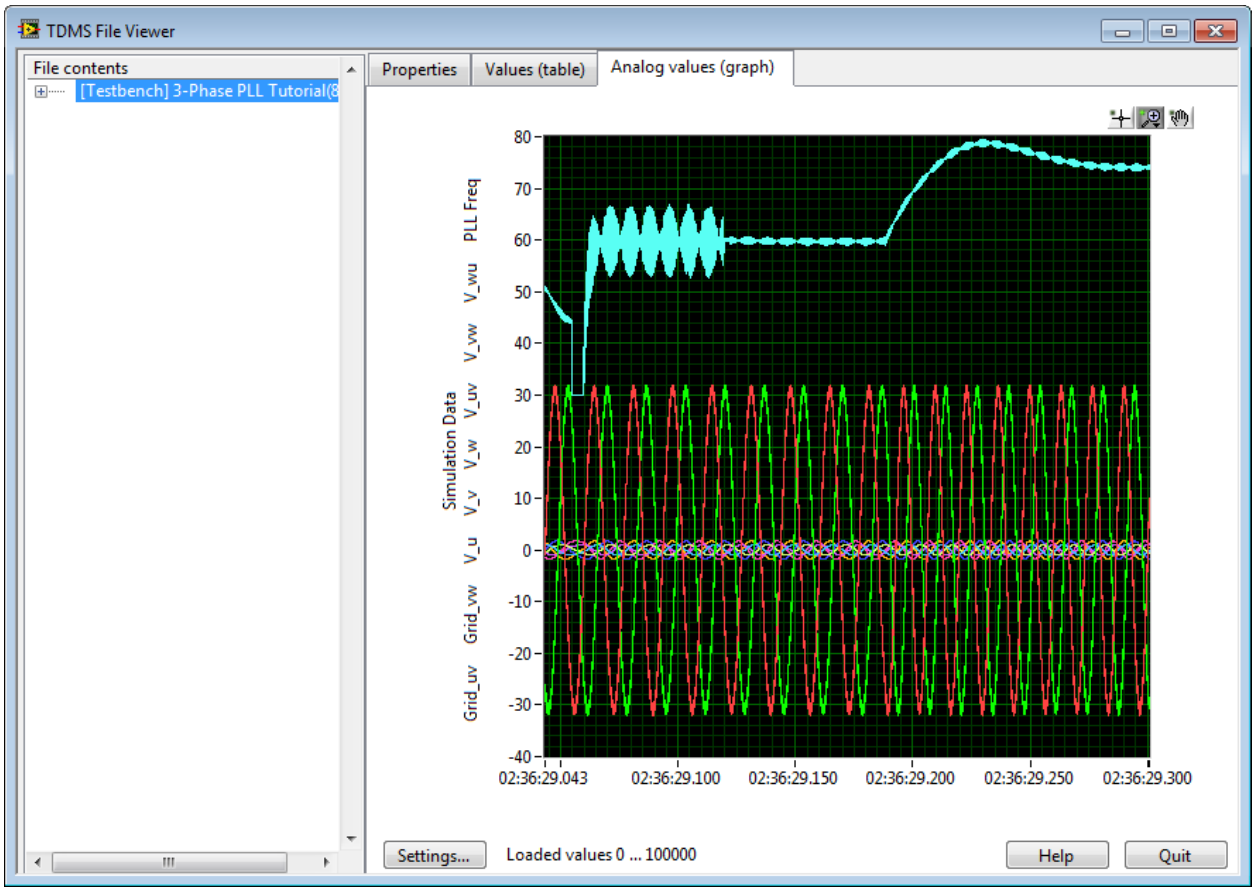
5. Vary the frequency of the simulated grid by changing the slider value labeled **Grid Freq.fo (Hz)** . Observe the fast tracking response of the PLL and generated three phase signals.

6. Changing the PI gains **proportional gain (Kp)** to a value of **0.2**. How does reducing the proportional gain impact the magnitude of the PLL frequency oscillation? Now, vary the frequency of the simulated grid again and observe the tracking response. Why does reducing the PI controller gain result in less aggressive tracking?

7. Click the **Stop** button on the front panel of the TestBench application. (Do not use the abort button.) Since the **Log TDMS?** button was asserted, all of the simulation data displayed on the graph indicator was automatically saved to a TDMS data file located in your Windows user folder **Documents\LabVIEW Data**. This facilitates automated testing and analysis of the simulation results using a variety of tools, including NI DIAdem and NI Teststand.

8. When the simulation is stopped, the **TDMS File Viewer** window appears. Click on the **Analog values (graph)** tab and then click on the File contents of the data log file. Explore

the TDMS File Viewer. For more information, click the **Help** button. Click the **Quit** button when you are finished.



9. Close out the application without saving.

Exercise 2 | 3-Phase Inverter Control and Experimental Comparison

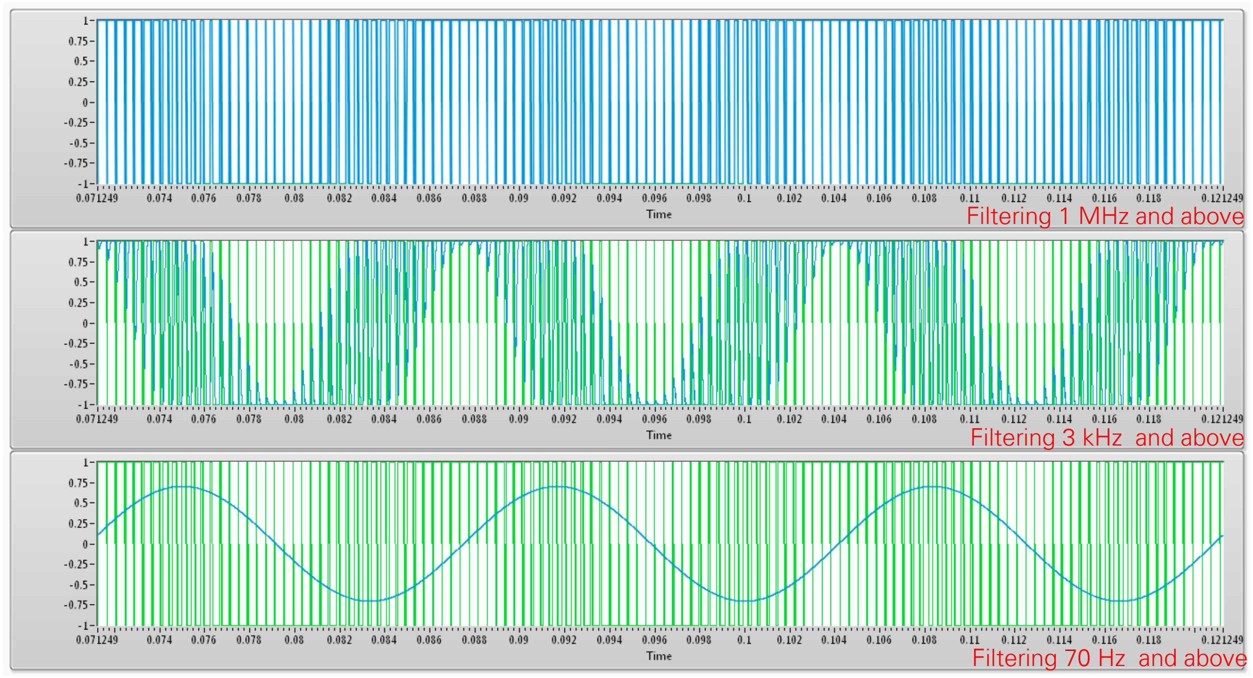
Summary

Now that you understand the basics of the three phase PLL, it is time to run our simulation and compare the results with the measurements from a grid synchronized 3-phase inverter control system. For this exercise, you will:

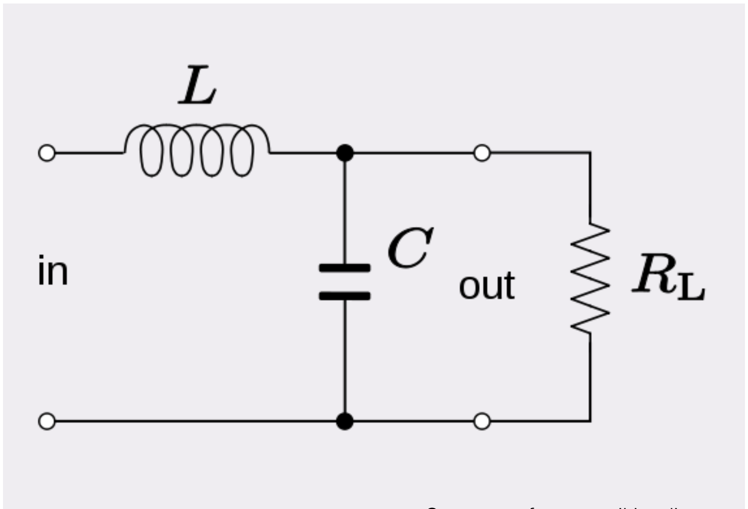
- Construct an RLC line-reactor filter circuit and attach it to the inverter
- Run the co-simulation application for the 3-phase inverter
- Compare the simulated and measured results

Background

The output from the IGBTs of the inverter will not be a smooth sine wave. When performing PWM, the signal generated on the output of the IGBTs will be a modulated square wave. In order to produce a sine wave, we must first build a 3-phase line reactor RLC low-pass filter. Filtering out the higher frequencies will leave us with the ~60 Hz outputs that we desire and reduce the total harmonic distortion of the inverter to an acceptable level. In the image below, you can see the raw PWM output of an inverter in green and the output after the line reactor filter in blue. The line reactor filter filters out the high frequencies to produce a nice sinusoid, but also adds phase shift to the inverter output.



A low-pass RLC filter can be constructed by using an inductor in series with a parallel combination of a smoothing capacitor and your resistive load. In one implementation, each of the 3-phases might have a line reactor filter like the one shown below.

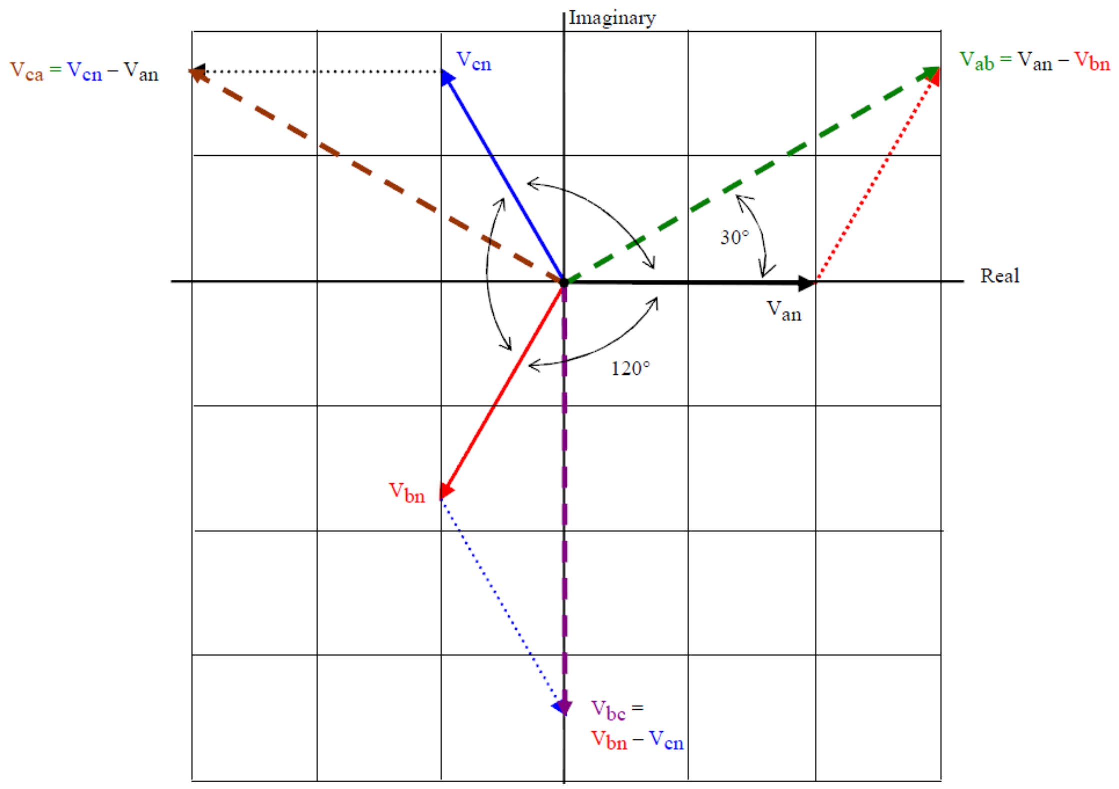


Courtesy of www.wikipedia.org

Understanding Line-to-Neutral versus Line-to-Line Voltages

In the image below, you will see phasors for both line-to-neutral measurements and line-to-line measurements. Items labeled V_{xn} where x is either a, b, or c, represent the line-to-neutral phasors. Notice that these three phasors are 120° offset from one another. The remaining phasors represent the line-to-line measurements, which are obtained by taking the difference between the line-to-neutral phasors. You will notice that these phasors are also 120° offset in relation to one another, but they are shifted by 30° from the original set of phasors.

Courtesy of Prof. Mack Grady
Understanding Power System Harmonics

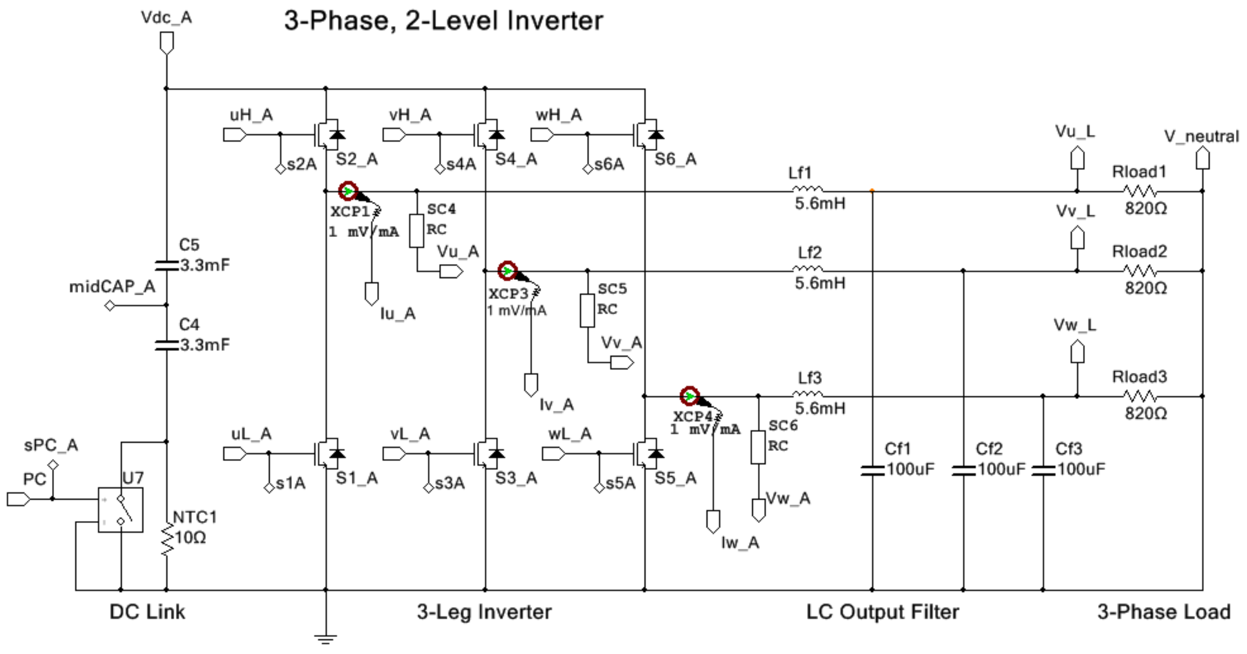


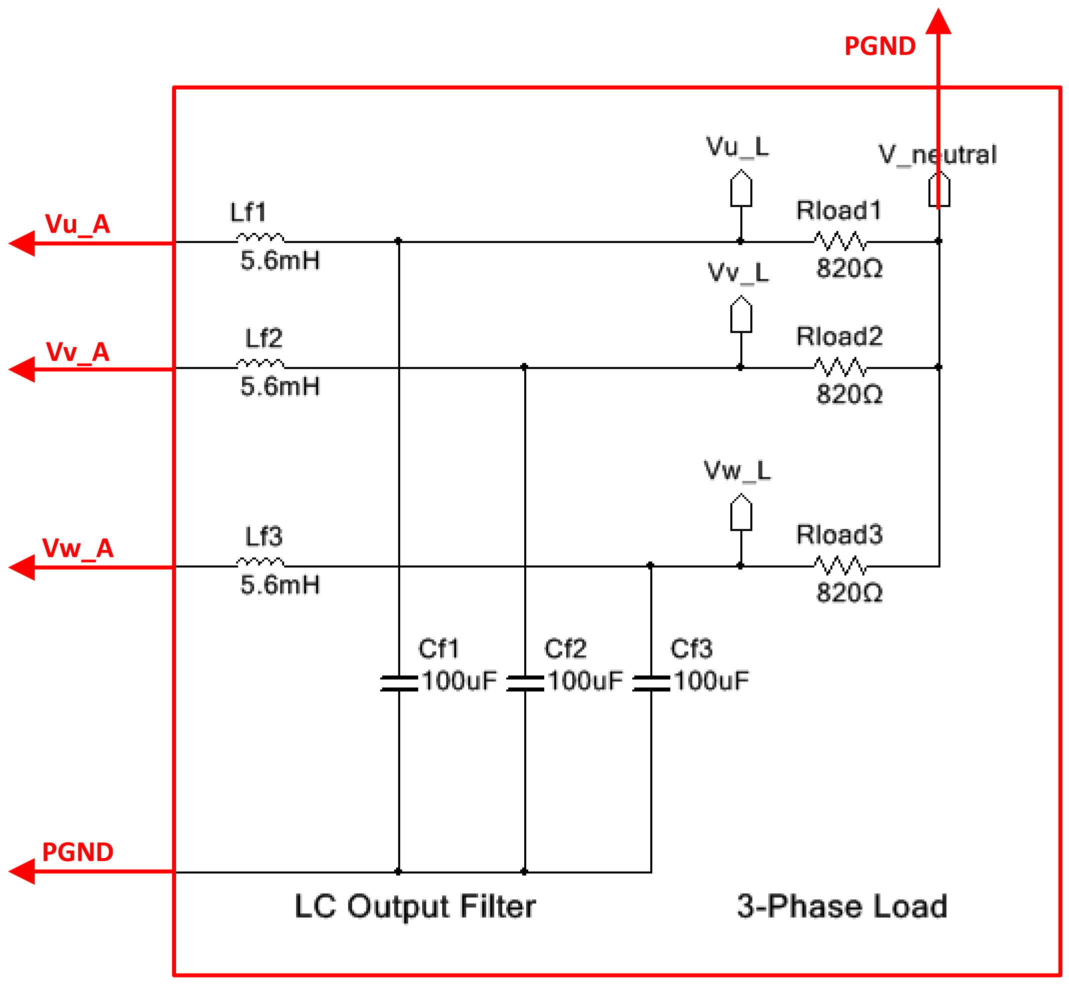
We will see that the PLL we used in the last exercise is being used to drive the IGBTs for this 3-phase inverter. However, inverter half-bridge outputs produce line-to-neutral voltages in proportion to the PWM duty cycle, and span from PGND to VDC. Line-to-line measurements are taken by measuring the voltage from one output phase to another. Measuring line-to-line, our 3-phase inverter produces bi-polar that span from –VDC to +VDC.

The issue with measuring line-to-line is that there is an inherent phase shift when measuring from line-to-line when compared to line-to-neutral. All thee output measurements will remain 120° phase shifted from one another, but they will be 30° out of phase with our line-to-neutral measurements. We can account for this simply by phase shifting the PLL reference signal by 30° after the PLL. This will ensure that the output of the PLL, which will be used to drive sine-triangle PWM will be locked in the correct phase relationship. We might also wish to add a positive phase shift to counteract the negative phase shift caused by our line-reactor filter, proportional to the phase shift of the filter at the nominal 50 or 60 Hz output frequency of the inverter.

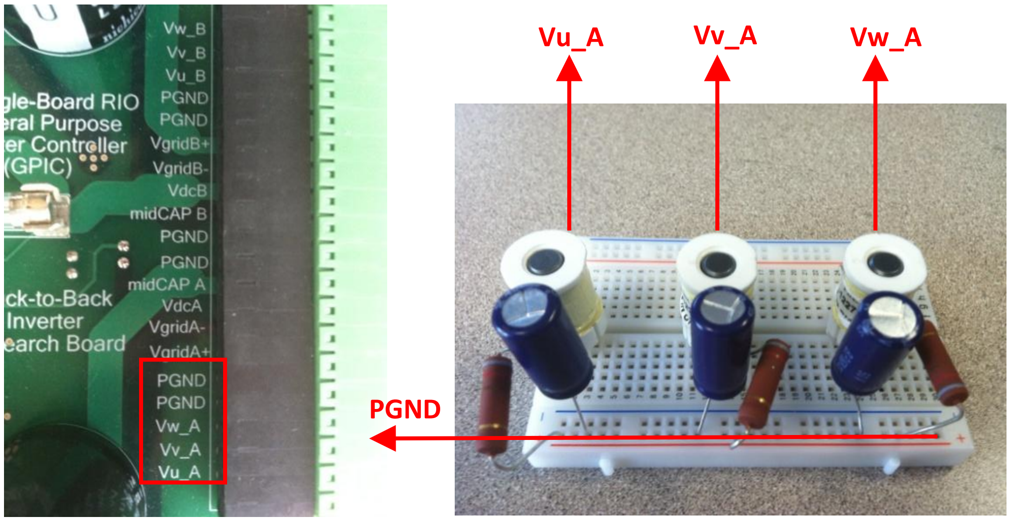
Examining the 3-Phase Line Reactor Filter Circuit

1. In the project window, expand out **My Computer**, **Hands on Workshop**, and then **2. Three Phase Inverter Control**. Open the Multisim schematic, **Half-Bridge IGBT Inverter.ms13**. The LC output filter and resistive load are connected to three half-bridge circuits to make a three-phase DC to AC inverter.





2. Using the image above, verify that the points on your filter are correctly wired into the GPIC Inverter Research Board. The labels in red, as seen in the image above, will match the labels on the pins for the Inverter board.



3. With the RLC filter configured and connected, it is time to enable the Inverter board. If you RT host application is not already running, open [RT] NI GPIC Inverter Research Board v08.vi, under the RT target, run, and deploy the application. If prompted to save, press Save All.
4. Open the Windows host application, Desktop] Graphical User Interface - GPIC 3-Phase Inverter Control.vi, under My Computer. Enter the IP address of your sbRIO GPIC Real-Time Controller. Then run the application.
5. Make sure Connected? is indicating true. Next, press Enable Control and then Enable PWM. You should now see the inverter voltage and current waveforms on the chart. When the control system reaches steady-state, click the Enable PWM button again to disable pulse width modulation. The LabVIEW Stream Data now contains the first 1-2 seconds of waveform data after the PWM Time (s) counter begins counting.

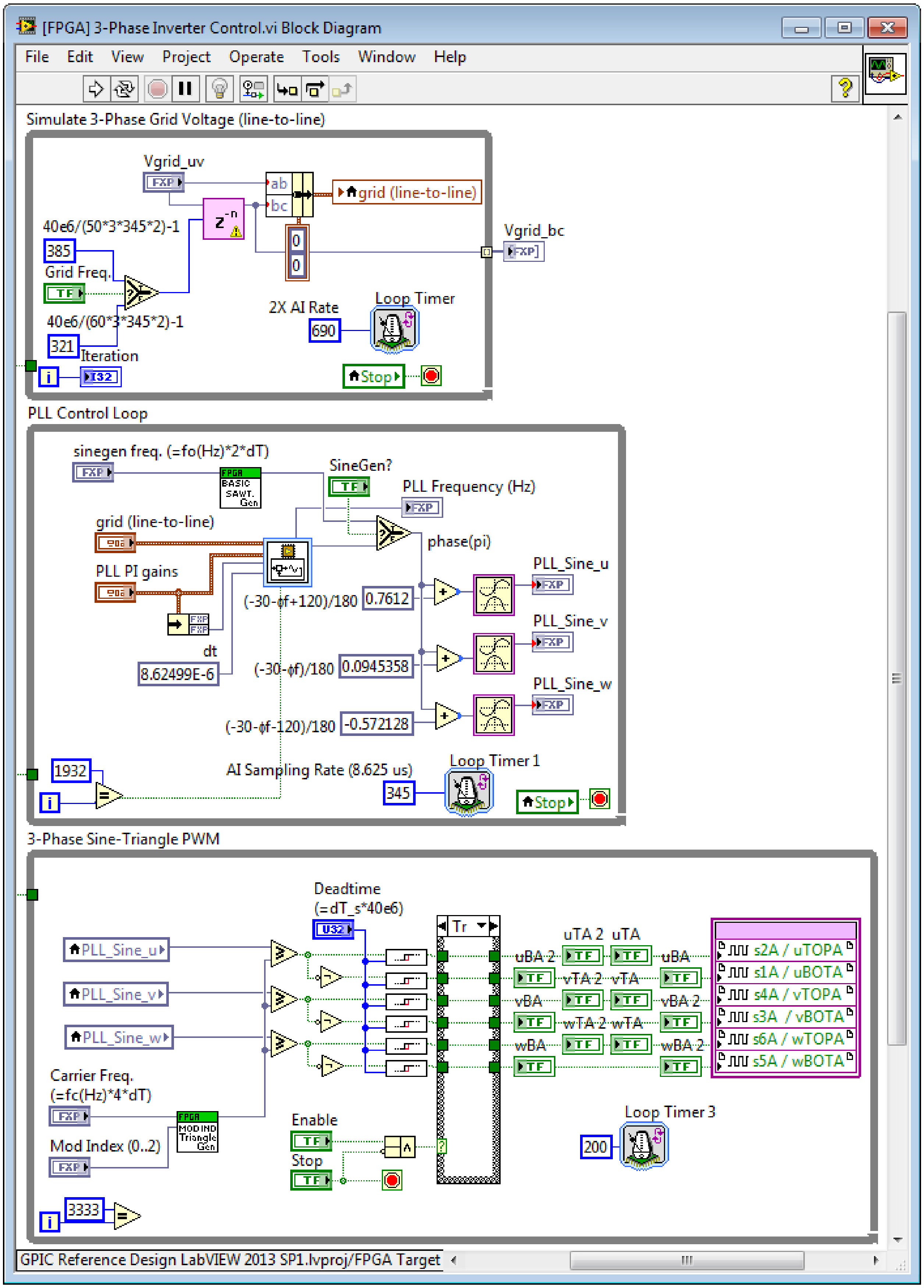
Running the 3-Phase IGBT Inverter Control Application

1. Now that the Windows host has received measurement data from the Inverter board, we can run our 3-phase inverter simulation and compare results. Navigate back to the project window. Open [Testbench] 3-Phase IGBT Inverter Control.vi.

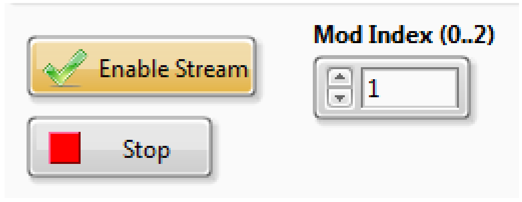
2. This testbench application will perform co-simulation of the FPGA application, Inverter control, and RLC filter, in order to produce an output. The output from the simulation can then be plotted alongside the results obtained from the board. First, open up the block diagram and double-click on the Desktop Execution Node in order to open up the FPGA application.

3. Type Control-E to open up the block diagram of the FPGA control application. You should recognize most of the elements here. The Three-Phase Lock Loop (PLL) while loop contains a PLL that generates three phase differing sinusoids, much like the previous examples. This type of PWM is identical to the sine-triangle PWM that was investigated during the first seminar section. However, instead of one inverter half-bridge, we are now driving three inverter phases.

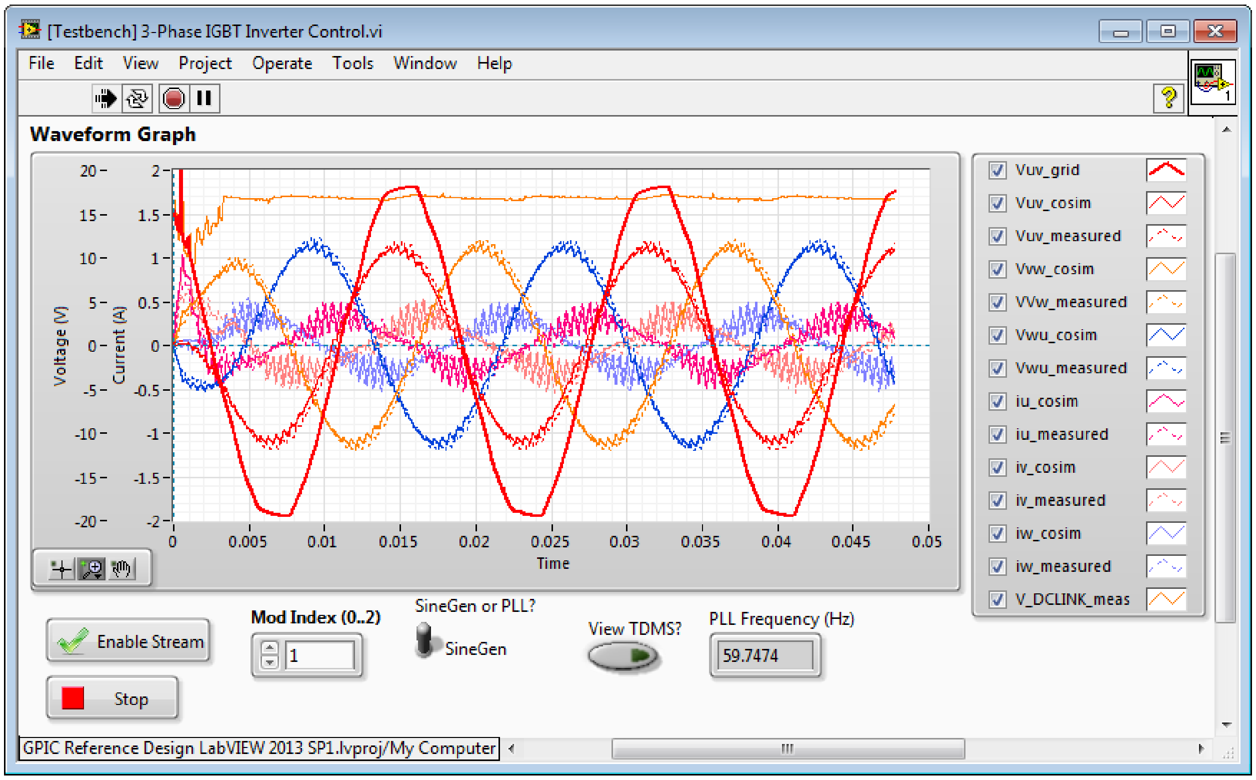
4. There is also a loop that takes the single phase voltage from the grid step down voltage transformer and produces a 120 degree phase shifted version to simulate three-phase voltage for the 3-phase PLL function. This code is for demonstration purposes only.


[FPGA] 3-Phase Inverter Control.vi Block Diagram
File Edit View Project Operate Tools Window Help
Simulate 3-Phase Grid Voltage (line-to-line)
Vgrid_uv
grid (line-to-line)
40e6/(50*3*345*2)-1
385
Grid Freq.
40e6/(60*3*345*2)-1
321
Iteration
Vgrid_bc
2X AI Rate
690
Loop Timer
Stop
PLL Control Loop
sinegen freq. (=fo(Hz)*2*dT)
SineGen?
PLL Frequency (Hz)
grid (line-to-line)
phase(pi)
PLL PI gains
PLL_Sine_u
(-30-φf+120)/180
0.7612
dt
8.62499E-6
(-30-φf)/180
0.0945358
PLL_Sine_v
(-30-φf-120)/180
-0.572128
PLL_Sine_w
1932
AI Sampling Rate (8.625 us)
345
Loop Timer 1
Stop
3-Phase Sine-Triangle PWM
Deadtime (=dT_s*40e6)
PLL_Sine_u
PLL_Sine_v
PLL_Sine_w
Carrier Freq. (=fc(Hz)*4*dT)
Mod Index (0..2)
Enable
Stop
3333
uBA 2
uTA 2
uTA
uBA
vTA 2
vTA
vBA
vBA 2
wTA 2
wTA
wBA
wBA 2
s2A / uTOPA
s1A / uBOTA
s4A / vTOPA
s3A / vBOTA
s6A / wTOPA
s5A / wBOTA
Loop Timer 3
200
GPIC Reference Design LabVIEW 2013 SP1.lvproj/FPGA Target

5. Navigate back to the TestBench application. Select the **Enable Stream** button. This will ensure that the measurement data we plot is coming from the inverter board.



6. Run the application. After a few moments, both the simulated and measured waveforms will begin to be plotted.



7. Using the panel to the right of the chart, display other waveforms to compare the simulation results to the measured results.

8. When you are done viewing the waveforms, stop the Inverter Controller application by pressing the **Stop** button.