RAPID PROTOTYPING OF REAL-TIME WIRELESS SYSTEMS

With LabVIEW Communications System Design Suite and NI USRP RIO

Version 2.0

NATIONAL INSTRUMENTS™

Worldwide Technical Support and Product Information
National Instruments Corporate Headquarters
Worldwide Offices

Q2 2015 Edition
Part Number 351260B-01

About National Instruments

Today's engineers and scientists are solving the world's most pressing challenges, such as developing better medical diagnostic and treatment tools, finding renewable energy alternatives, and improving infrastructure stability. National Instruments equips engineers and scientists with tools that accelerate productivity, innovation, and discovery to meet not only grand but also daily engineering challenges in an increasingly complex world. A graphical system design approach leverages productive software and reconfigurable hardware platforms, along with a vast community of IP and applications, to simplify system development and arrive at solutions faster.

Learn more about our vision at http://www.ni.com/company/our-vision/

Simplify

With the NI platform-based approach, you can design systems at the right level—from using small robots to inspire student engagement in engineering to launching the most advanced rockets into space—using a single software framework and reconfigurable hardware. Focus on the solution to your problem by abstracting complexity through software and removing the unnecessary constraints fixed-function tools cause.

Optimize

Invest in tools that allow you to adapt to changing requirements over time while optimizing both performance and cost. By taking advantage of the most advanced components available today, National Instruments delivers solutions that harness rapid technology change to empower you with the most effective tools.

Innovate

Build solutions supported by a thriving community of users, partners, tools, and IP that help ensure your success. National Instruments provides global services and support as part of its commitment to your effort in building and maintaining high-quality measurement and control systems using a graphical system design approach.

Contents

Overview

Purpose of the Hands-On Seminar

With this hands-on seminar, NI seeks to educate industry professionals and researchers on prototyping a wireless communication system by explaining and working through a common design flow. You physically set up and plug in an NI USRP™ (Universal Software Radio Peripheral) reconfigurable I/O (RIO) software defined radio, write programs in LabVIEW, and move the algorithms to a high-throughput FPGA.

The exercises are examples of how to model, simulate, and prototype signal processing algorithms for applications involving wireless signals. Though the exercises are generic, you should be able to apply what you learn to your own applications. For future reference, this manual is available both electronically and in print.

What You Will Do

Through four main exercises you will gain an understanding of FPGA algorithm development, floating point design and testing, fixed-point conversion and finally a deployment of your algorithm to a prototyping device such as the NI USRP RIO.

The presenter starts with a quick overview that defines some common terms and describes how to approach your task with NI hardware and software.

Why You Should Take This Course

Take this course if you

- Are researching wireless communications
- Need to quickly and easily prototype and validate algorithms
- Want to evaluate the usefulness of the software defined radio in your application
- Need exposure to setting up software defined radio

Time Required to Complete the Course

The course should take approximately three hours, but this time can vary depending on your background.

Required Background

NI recommends that you have some exposure to LabVIEW, but it is not required. The instructions for the exercises cover all necessary steps to complete the task. You are expected to learn basic tasks as you progress. The instructions become less detailed and require that you retain some of the knowledge. Visit ni.com/academic/students/learn-labview to learn more. If you are new to LabVIEW, the Introduction to the LabVIEW Editor lesson offers a general overview of the LabVIEW environment.

Select **File » Learn » Getting Started** to find it.

Required Equipment

Hardware

- USRP RIO transceiver (such as 2940R)
- USRP power adapter
- USRP RIO connectivity kit
- SMA loopback cables with 30 dB attenuator

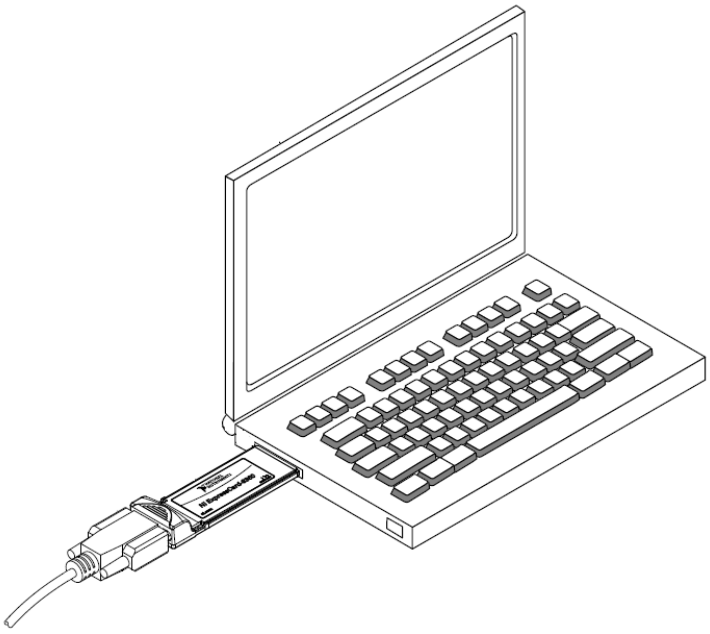
Software

- LabVIEW Communications System Design Suite 1.0 or later

Configuring Hardware

Follow these steps after you install LabVIEW on your computer:

1. Install LabVIEW Communications
2. Power down your computer and NI USRP RIO device
3. Attached your NI USRP RIO device to your computer with the connectivity kit
4. Power on the NI USRP RIO device
5. Power on your computer

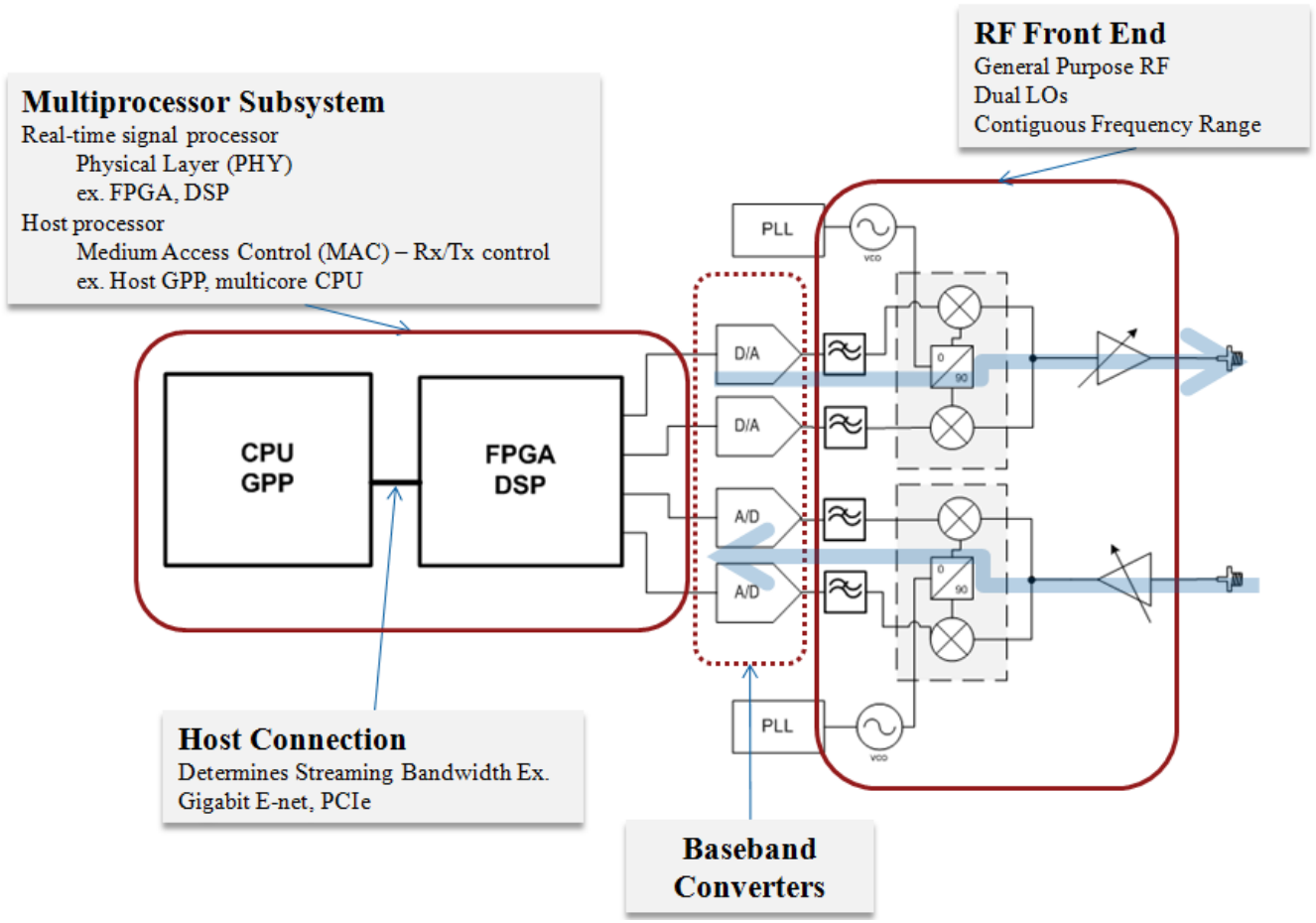


What Is a Software Defined Radio?

The Wireless Innovation Forum defines a software defined radio (SDR) as:

"A radio in which some or all of the physical layer functions are software defined." [1]

SDR refers to the technology wherein software modules running on a generic hardware platform are used to implement radio functions. Combine NI USRP hardware with LabVIEW software for the flexibility and functionality to deliver a platform for rapid prototyping involving physical layer design, wireless signal record and playback, signal intelligence, algorithm validation, and more.



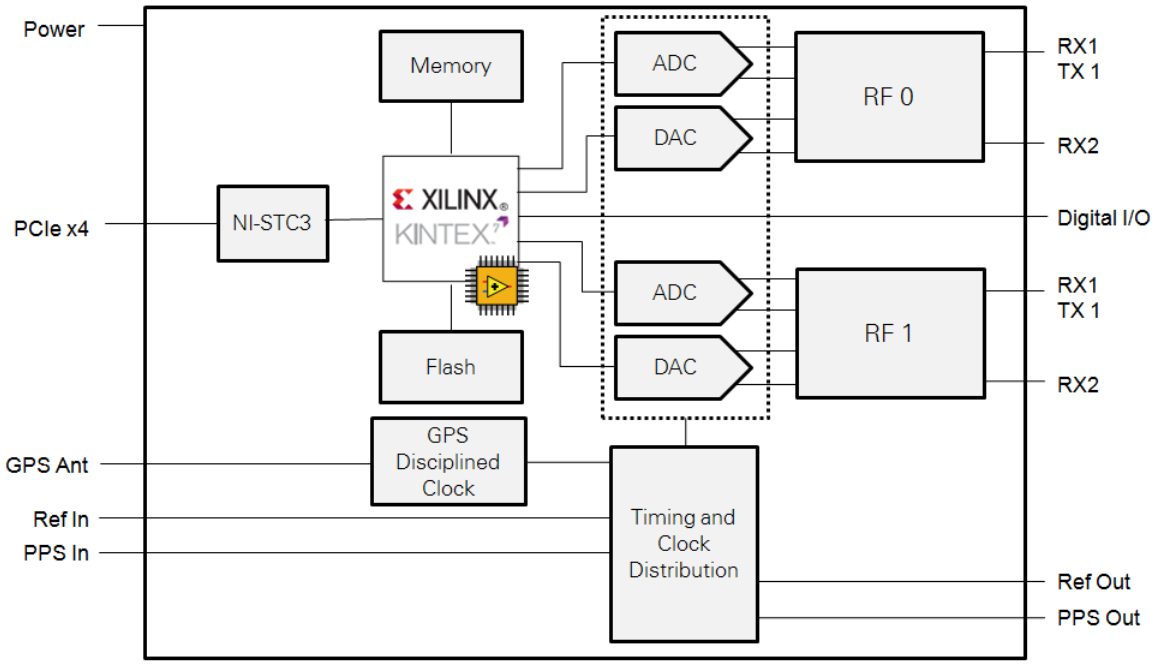
[1] http://www.sdrforum.org/pages/documentLibrary/documents/SDRF-06-R-0011-V1_0_0.pdf

NI USRP RIO Hardware Architecture

The NI USRP RIO offers wireless communications designers an affordable SDR with unprecedented performance for developing next-generation 5G wireless communication systems. The USRP RIO has a state-of-the-art 2x2 multiple input, multiple output (MIMO) RF transceiver with a LabVIEW-programmable DSP-oriented Kintex-7 FPGA. LabVIEW provides a unified design flow that enables wireless communications researchers to prototype faster and significantly shorten time to results. USRP RIO extends the USRP platform with a refined user experience that makes SDR prototyping more accessible by delivering the optimum balance of performance and streamlined software tool flow. It is ideal for a wide range of application areas including 5G wireless communications, massive MIMO, and spectral monitoring.

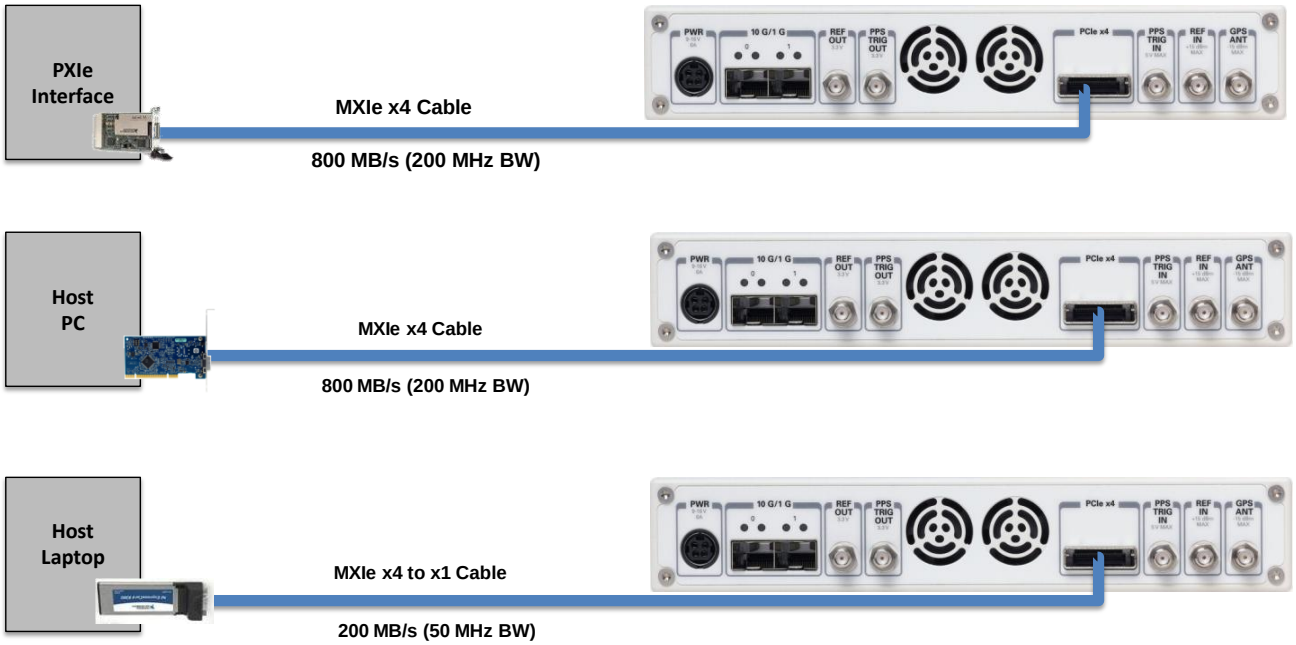
The USRP RIO combines two full-duplex transmit-and-receive channels with 40 MHz per channel of real-time bandwidth and a large DSP-oriented Kintex-7 FPGA in a half-1U rack-mountable form factor. The analog RF front end interfaces with the large Kintex-7 410T FPGA through dual analog-to-digital converters (ADCs) and digital-to-analog converters (DACs) clocked at 120 MS/s. Each RF channel includes a switch that allows for time division duplex (TDD) operation on a single antenna using the TX 1 RX1 port, or frequency division duplex (FDD) operation using two ports, TX1 and RX2.

You can choose from six different USRP RIO devices with frequency options that span from 50 MHz to 6 GHz and user-programmable digital I/O lines for controlling external devices. The Kintex -7 FPGA is a reconfigurable LabVIEW FPGA target that incorporates DSP48 coprocessing for high-rate, low-latency applications. PCI Express x4 connection back to the system controller allows up to 800 MB/s of streaming data transfer back to your desktop or PXI chassis and 200 MB/s to your laptop. With this connection, you can cable up to 17 USRP RIO devices back to a single PXI Express chassis, which you can then daisy chain with other chassis for high-bandwidth, high-channel-count applications.



NI USRP RIO Connectivity Options

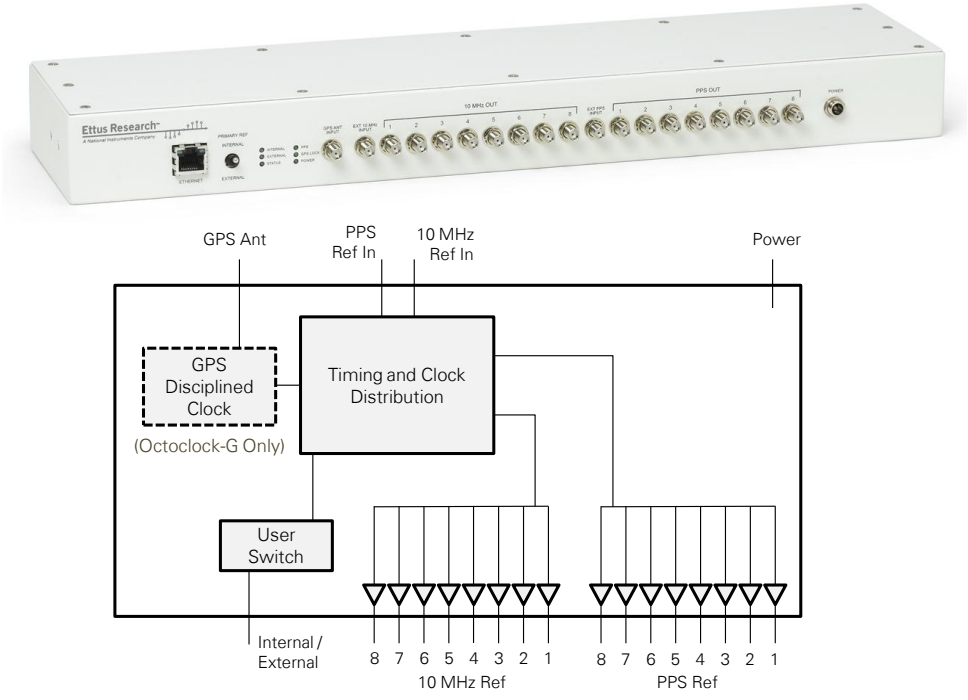
The primary interface bus for the USRP RIO is PCI Express x4, which provides an effective connection for high-bandwidth and lower latency applications such as PHY/MAC applications. With the PXI Express x4 bus, you can stream data at up to 800 MB/s and customize the FPGA in the LabVIEW FPGA Module. The interface is backward-compatible with programs written for the NI USRP-292x and USRP-293x devices. USRP RIO hardware contains several ports for future expansion via software upgrades. These inactive ports include dual SFP+ connections on the rear panel and a USB JTAG debug port on the front panel.



Building High Channel Count Systems

USRP-294xR devices include a temperature compensated crystal oscillator (TCXO) as the base frequency reference, which works well as a general-purpose oscillator. USRP-295xR devices include a precision GPS-disciplined oven-controlled crystal oscillator (OCXO), which offers improved frequency accuracy without using GPS and significantly improved frequency accuracy when disciplined to the GPS satellite network.

All USRP RIO models include options for using an internal or external clock reference with the added ability to export the clock reference and timebase to other devices. The Ref In port accepts a 10 MHz reference from which you can derive the ADC/DAC clocks and local oscillator. You can use PPS In as a standard pulse per second port or as a general-purpose digital trigger input line. With Ref Out and PPS Out, you can export either of those signals to a nearby device for building higher channel count systems. Using amplified clock distribution, featuring the 8-channel OctoClock from Ettus Research, you can build extremely large synchronized systems. Just connect your USRP RIO devices to the Ref In and PPS using several OctoClocks to build systems that exceed 100 synchronized channels.

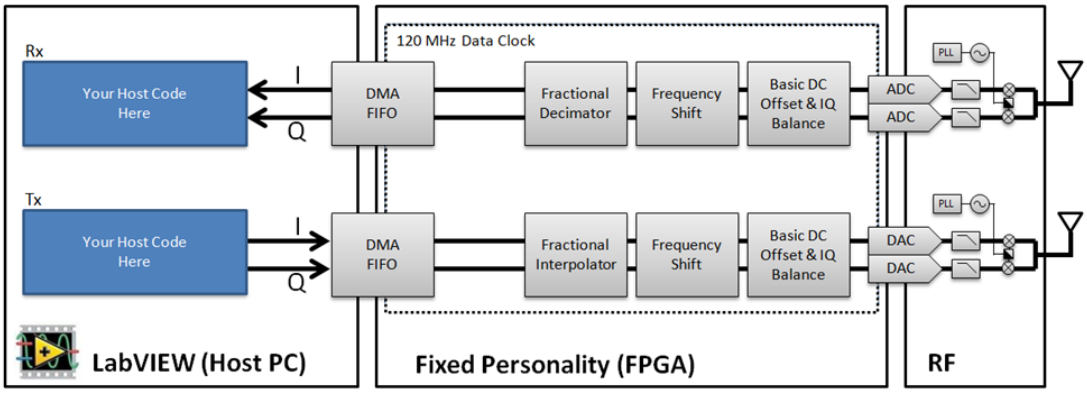


Application Programming Interface (API)

The USRP RIO takes advantage of two complementary LabVIEW-based software driver experiences: a host-based driver (NI-USRP) and a fully open and customizable LabVIEW FPGA (NI-USRP RIO). Both driver interfaces support connectivity over PCI Express and use a similar driver approach so you can efficiently take your design from the host computer to the FPGA.

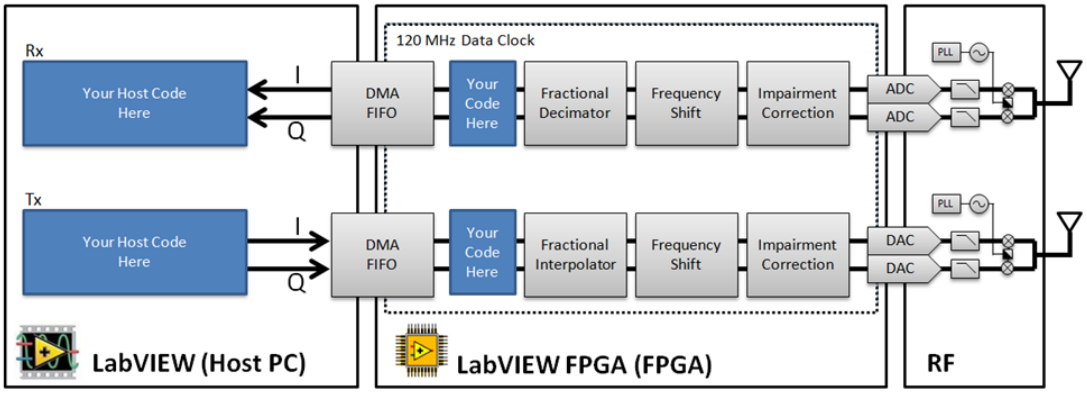
NI-USRP Driver

NI is releasing USRP RIO hardware with NI-USRP 1.3 driver support to provide a seamless host-based interface that is fully backward compatible with USRP-292x and USRP-293x devices. By using a fixed FPGA image configurable from the host API, you can develop your algorithm in LabVIEW and seamlessly move between USRP and NI USRP RIO devices.



NI-USRP RIO Driver

As your applications require increased performance, you can take advantage of the large Kintex-7 FPGA for coprocessing by migrating your design using the NI-USRP RIO driver. This driver provides a streaming sample project that includes an open host processor and FPGA design code written in LabVIEW and LabVIEW FPGA, respectively. You can configure the sample project so that the code runs only on the host and/or modify the FPGA personality to include custom processing. Though the entire FPGA reference design is customizable, you most often need to insert your code in the signal chain near the DMA first-in-first-out (FIFO) memory buffer. The streaming sample project is based on the Instrument Design Library reference design common to NI FlexRIO SDRs and NI vector signal transceiver (VSTs).



Exercises

Exercise 1 – Introduction to LabVIEW Communications

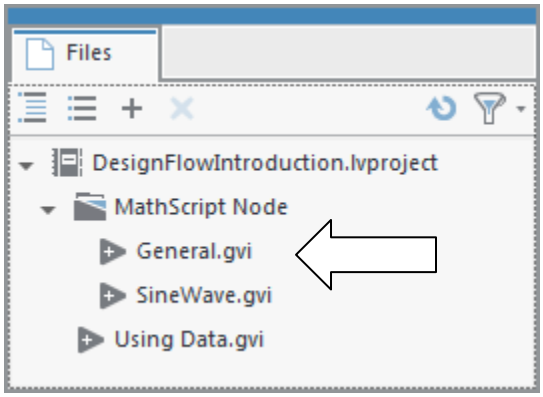
This lesson teaches you how to manage files in the Files pane, use the MathScript node as a starting point for algorithm development, and use captured data between VIs.

1. Open LabVIEW Communication System Design Suite.
2. On the left hand navigation, click **Learn**.
3. Choose **Getting Started»Overview of the FPGA Design Flow»Introduction to the FPGA Design Flow**.
4. Click **Start** or **Start Over**.

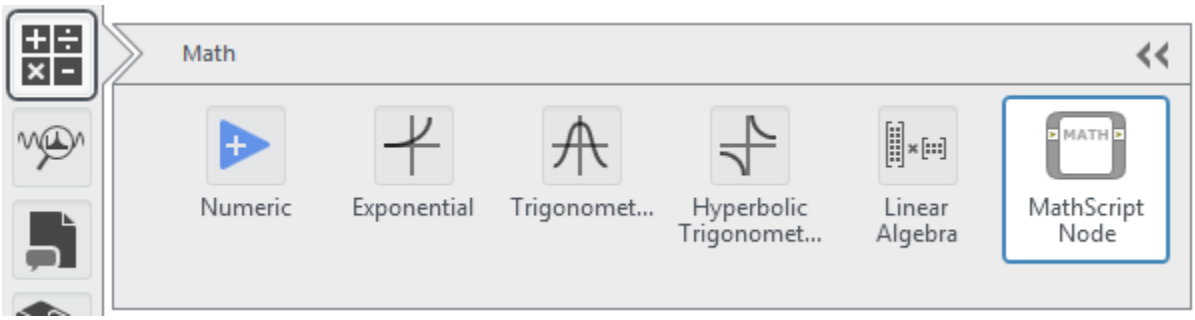
Collapse the Guided Help window using the collapse button.  You will follow this document instead of using Guided Help.

The MathScript Node: A Simple Sine Wave

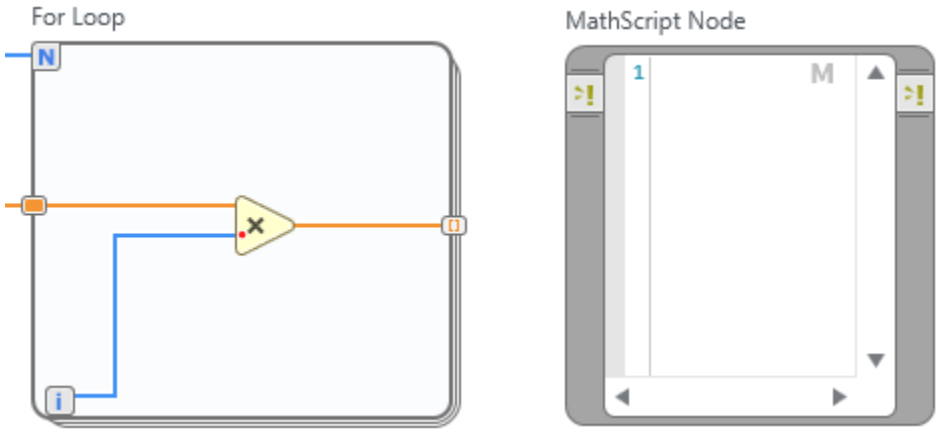
1. From the Files Pane, expand the **MathScript Node** folder and double click to open **SineWave.gvi**.



2. Switch to the diagram of SineWave.gvi to show the source code by pressing <Ctrl-E> on your keyboard.
3. Using the palette, add a MathScript Node from **Math»MathScript Node**.



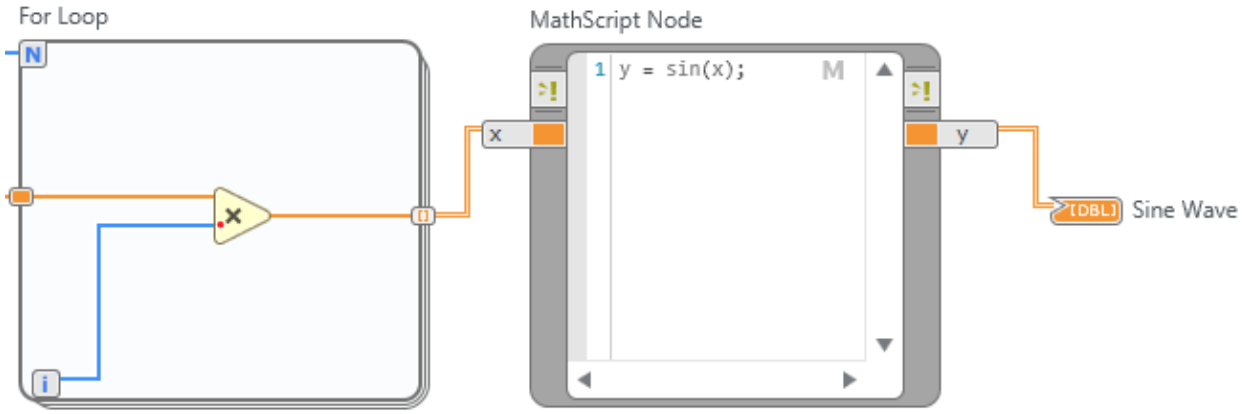
4. "Draw" a MathScript node after the For Loop.



5. Click inside the MathScript node and type:

```
y = sin(x);
```

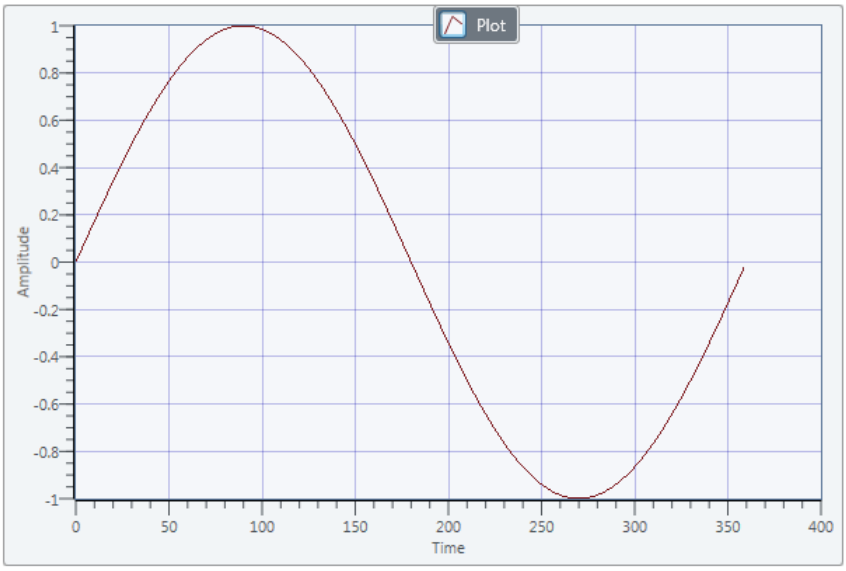
6. Right-click the y variable and choose **Create»Output: y** from the popup menu.
7. Right-click the x variable and choose **Create»Input: x** from the popup menu.
8. Wire the output of the For Loop to the **x** input of the MathScript Node.
9. Wire the **y** output of the MathScript Node to the Sine Wave terminal.



Press <Ctrl-E> on your keyboard to switch to the panel.

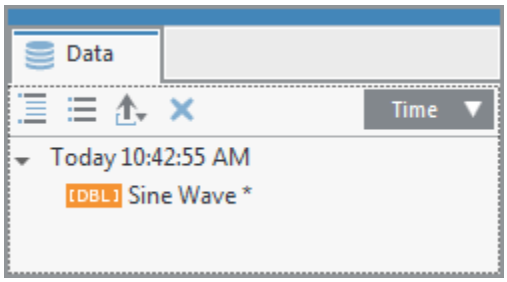
Run the program with the Run button. 

You should have a working program and the Sine Wave graph should display a single cycle of a sine wave.

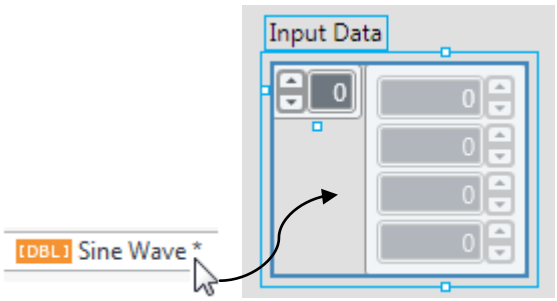


Explore: Using Captured Data

1. Right-click the Sine Wave graph and choose **Capture Data** from the popup menu.



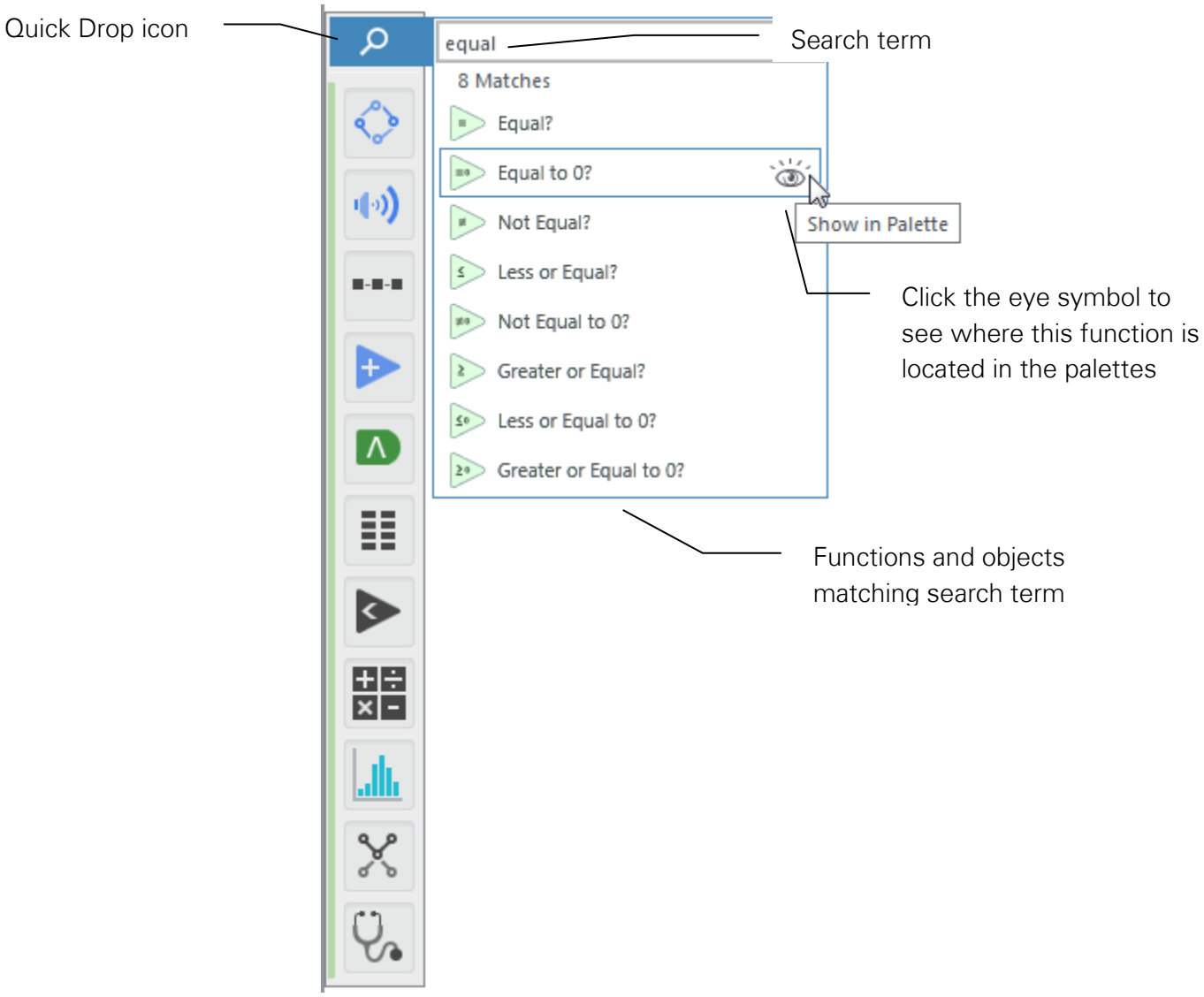
2. Open the **Using Data.gvi** program from the Files pane.
3. Drag the Sine Wave entry from the Data pane onto the Input Data array on the Using Data.gvi program.



4. Run the VI and you should see that the Graph has updated using the captured data you provided.

Please wait for the instructor before continuing

Tip: You can press <Ctrl-Space> to activate quick drop. Quick Drop lets you quickly find and place LabVIEW front panel and block diagram objects without navigating the palettes.



Exercise 2 – Algorithm Design and Testing

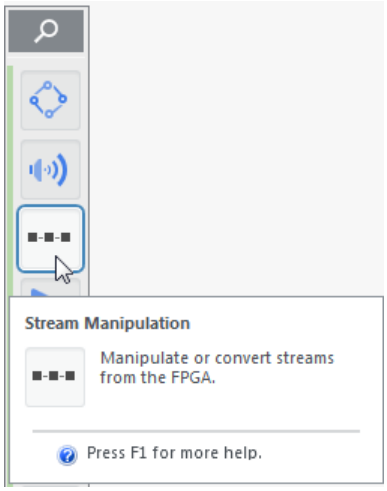
Designing and testing an algorithm are the first steps in the FPGA Design Flow. In this lesson you will create an algorithm to implement orthogonal frequency-division multiplexing (OFDM). You will use a Multirate Diagram, gain familiarity with the OFDM algorithm used in future lessons, and verify your algorithm through testing.

1. Close Exercise 1 from **File » Close All**. You do not need to save any files..
2. Click **Getting Started»Overview of the FPGA»Algorithm Design and Testing**.
3. Click **Start** or **Start Over**.

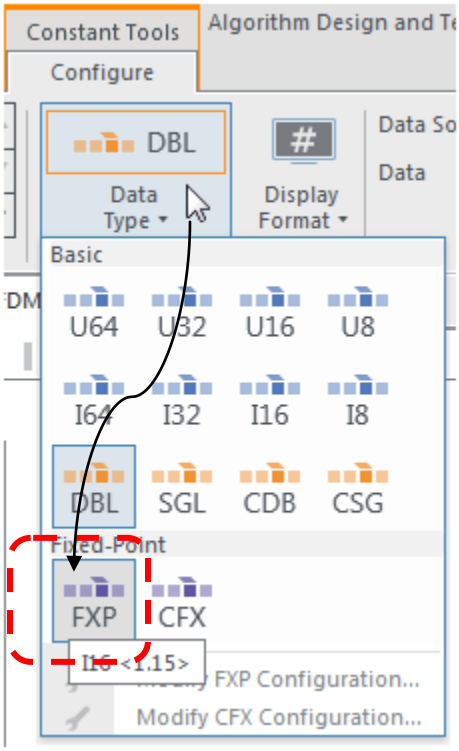
Collapse the Guided Help window using the collapse button.  You will follow this document instead of using Guided Help.

Build an OFDM Modulator with Multirate Diagram

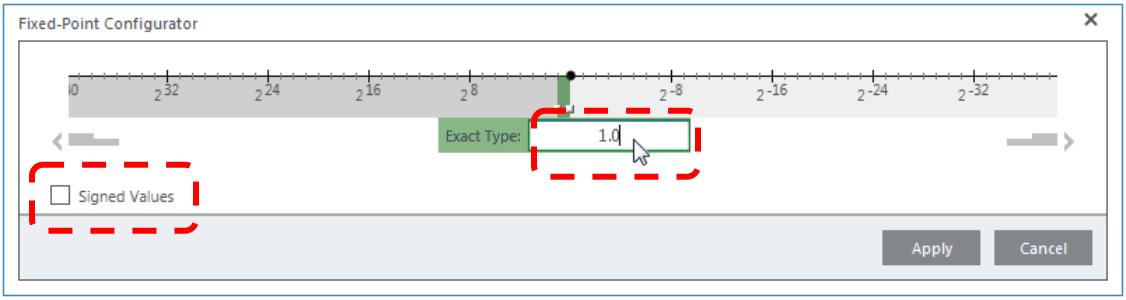
1. Open **MyOFDM Tx Flt.gmrd** from the Files pane.
2. Place a **Stream Constant** from the Stream Manipulation palette.



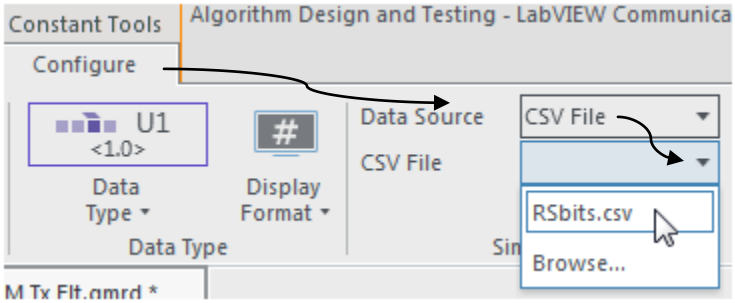
3. From the Configure ribbon at the top, change the Data Type to **FXP**.



4. Configure the stream constant as follows:

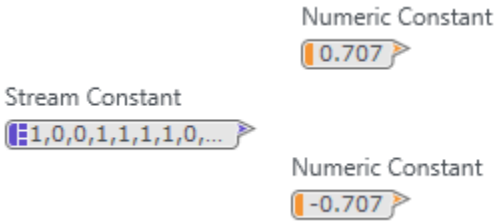


5. Click **Apply**.
6. From the Configure ribbon, change the Data Source to **CSV File**.
7. Change the CSV File drop down to RSbits.csv.

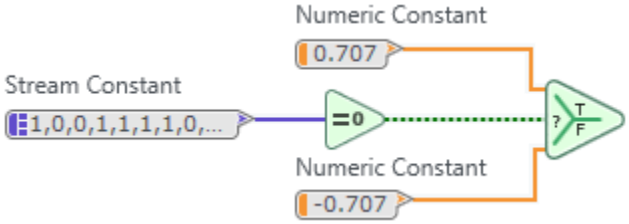


8. Place a **Numeric Constant** on the block diagram, above the stream constant
9. Change the value of the Numeric Constant to **0.707**.

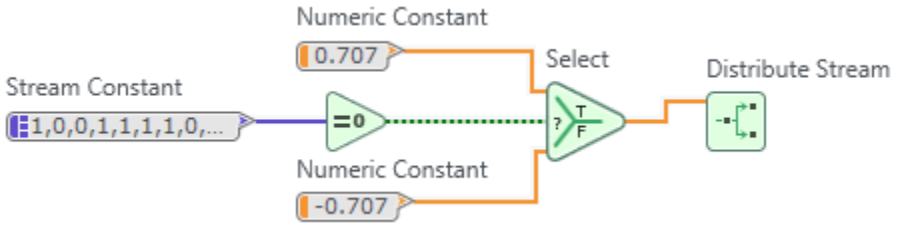
10. Copy and paste the **Numeric Constant** below the Stream Constant and change the value to read **-0.707**.



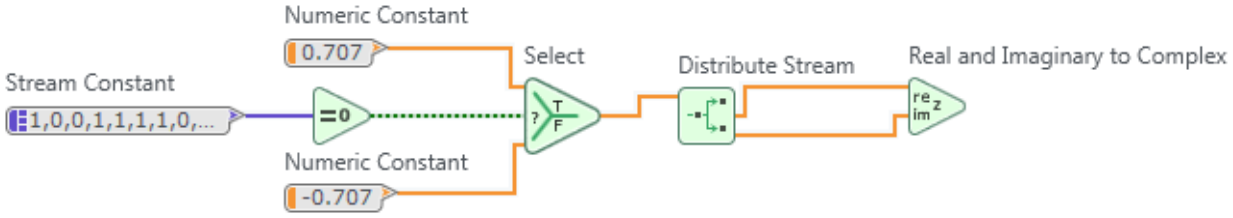
11. Place and wire a **Equal to 0** function to the right of the Stream Constant.
12. Place a **Select** function next to the **Equal to 0** function.
13. Wire the output of the **Equal to 0** to the center input of the **Select** function.
14. Wire the **+ 0.707** Numeric Constant to the top input of the **Select** function.
15. Wire the **- 0.707** Numeric Constant to the bottom input of the **Select** function.



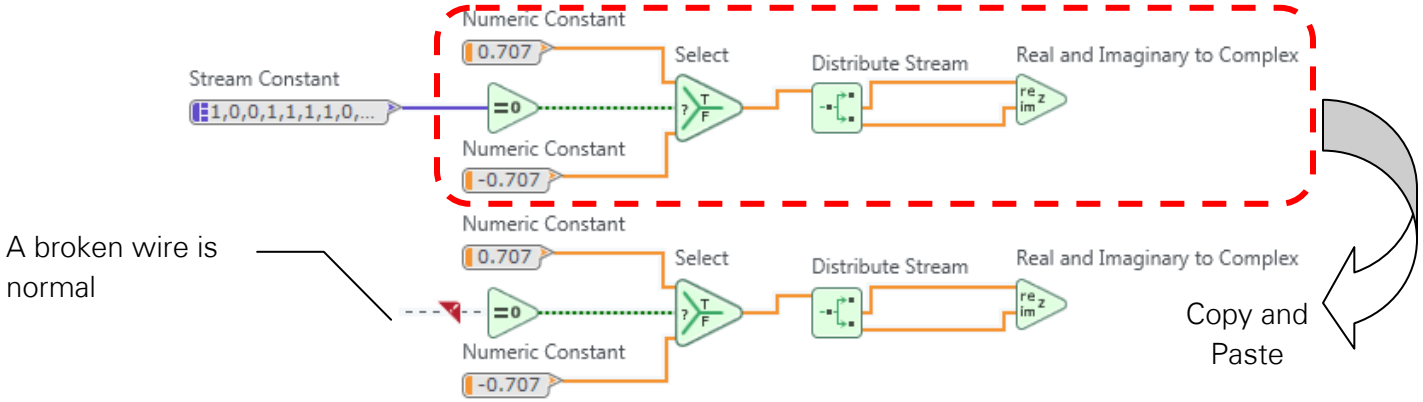
16. Place and wire a **Distribute Stream** function next to the **Select** function.



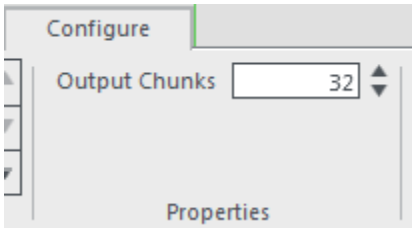
17. Add a **Real and Imaginary to Complex** function next to the **Distribute Stream** function.
18. Wire the top output from **Distribute Stream** to the top input of **Real and Imaginary to Complex**.
19. Wire the bottom output from **Distribute Stream** to the top input of **Real and Imaginary to Complex**.



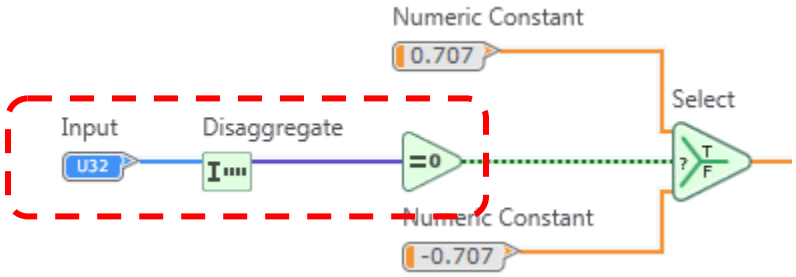
20. Copy and paste the following to create the following diagram:



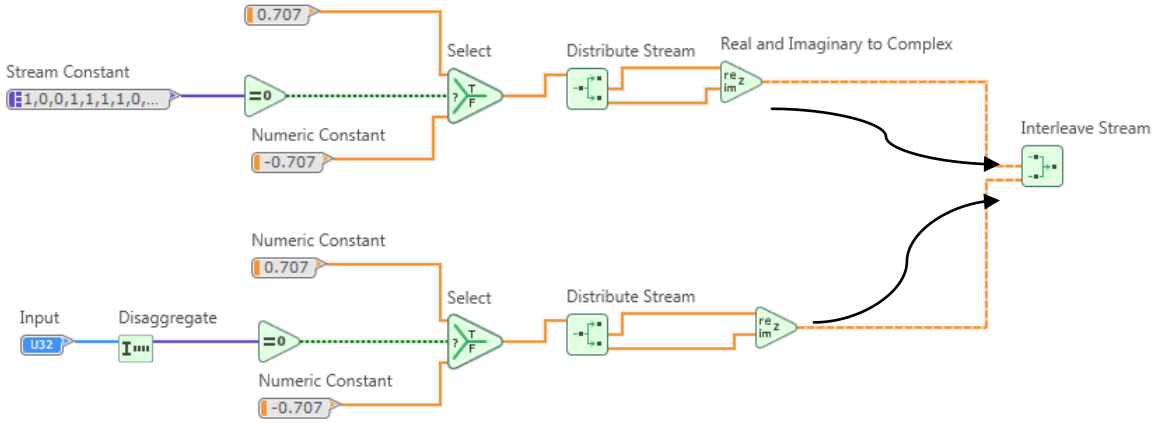
21. Next to the broken wire, place a **Disaggregate** function.
22. While the Disaggregate function is selected, change **Output Chunks** to 32 from the Configure ribbon



23. Move and wire the Input U32 terminal to the **Disaggregate** function.
24. Wire the **Disaggregate** function to the broken wire.

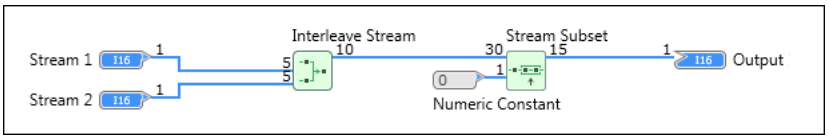


25. Place an **Interleave Stream** function in between the two **Real and Imaginary to Complex** functions.
26. Wire the top output of **Real and Imaginary to Complex** to the top input of **Interleave Stream**.
27. Wire the bottom output of **Real and Imaginary to Complex** to the bottom input of **Interleave Stream**.



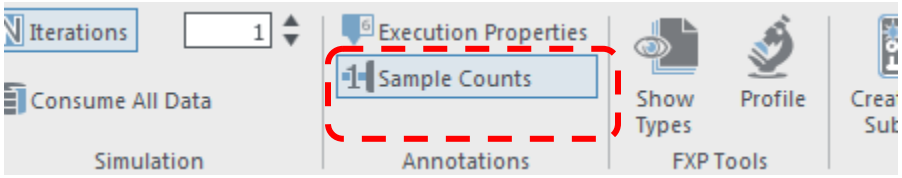
Display and Configure Sample Counts

To help you visualize Multirate Dataflow, LabVIEW provides the ability to display the sample counts of each node terminal on a Multirate Diagram. The sample count for each input terminal represents the number of data samples that input requires before the node can execute. The sample count for each output terminal represents the number of data samples that the node returns. For many nodes, you can configure the number of data samples to input to and output from the node.

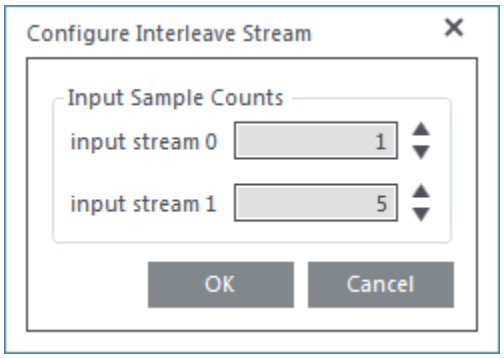


As the example above illustrates, displaying sample counts is useful for visualizing the flow of data streams and making sure each node operates on the correct number of data samples.

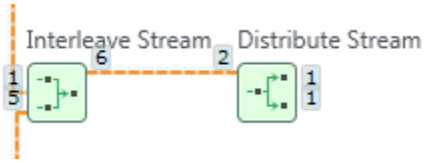
1. From the ribbon, click the **Sample Counts** button.



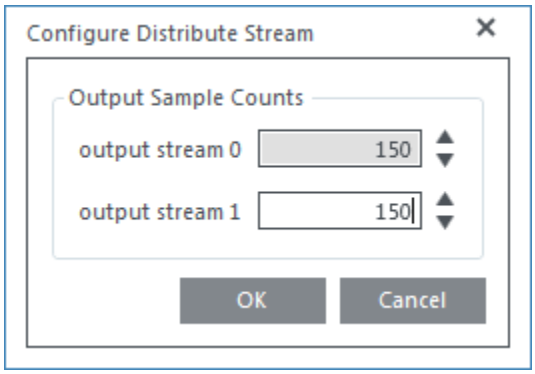
2. Double-click the **Interleave Stream** function.
3. On the dialog box that appears, change to the following settings:



4. Place and wire a **Distribute Stream** function next to the **Interleave Stream** function.

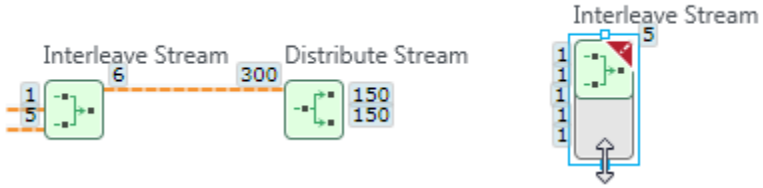


5. Double-click the **Distribute Stream** function and configure as follows:

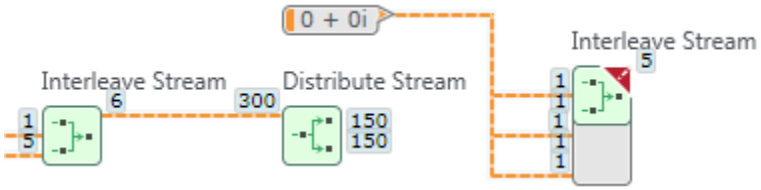


6. Place an **Interleave Stream** function next to the **Distribute Stream** function.

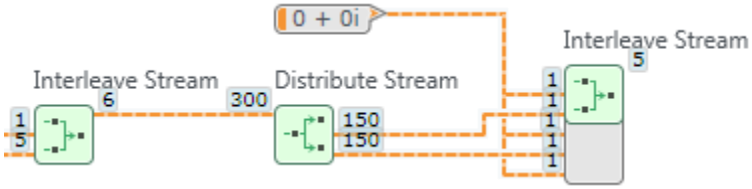
7. Expand the Interleave Stream function so that it accepts a total of 5 inputs.



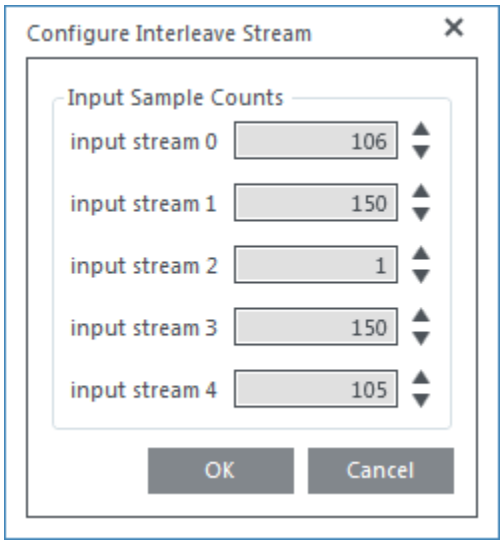
8. Place a **Numeric Constant** above the **Distribute Stream** function.
9. Set the value of the **Numeric Constant** to **0 + 0i**.
10. Wire the **Numeric Constant** to the 1st, 3rd and last inputs of the **Interleave Stream** function.



11. Wire the top output from **Distribute Stream** to the 2nd input of **Interleave Stream**.
12. Wire the bottom output from **Distribute Stream** to the 4th input of **Interleave Stream**.

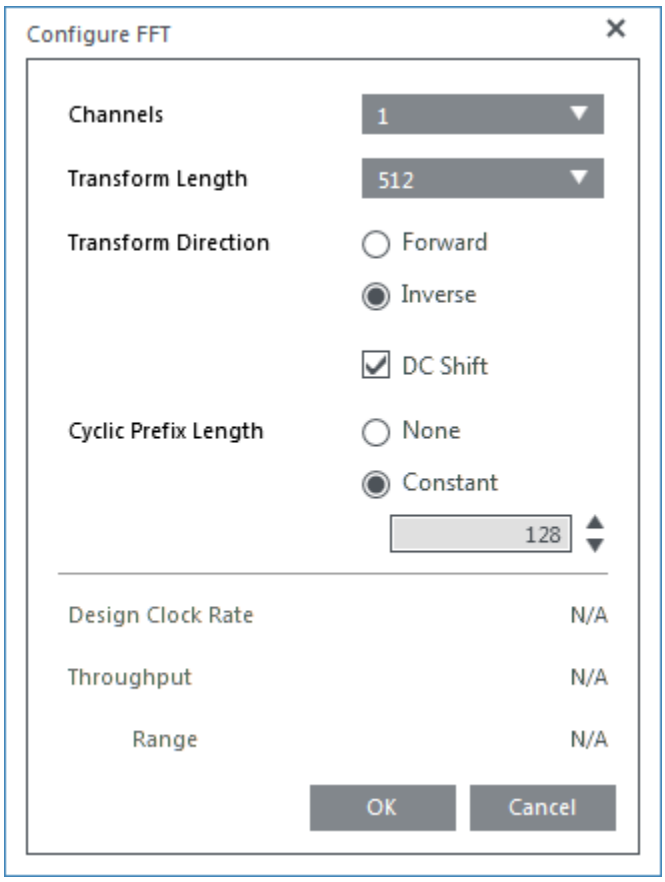


13. Double-click the **Interleave Stream** function and configure as follows:



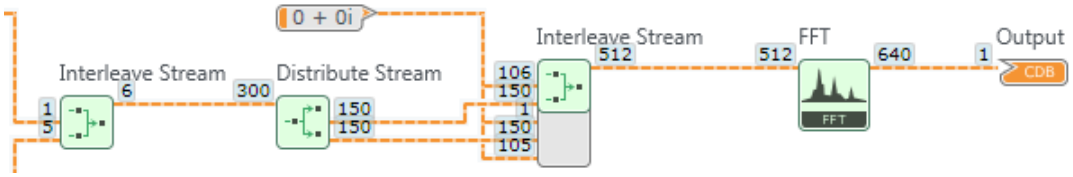
14. Click **OK**.
15. Place and **FFT** function next to the **Interleave Stream** function.

16. Double-click the FFT and configure it as follows:



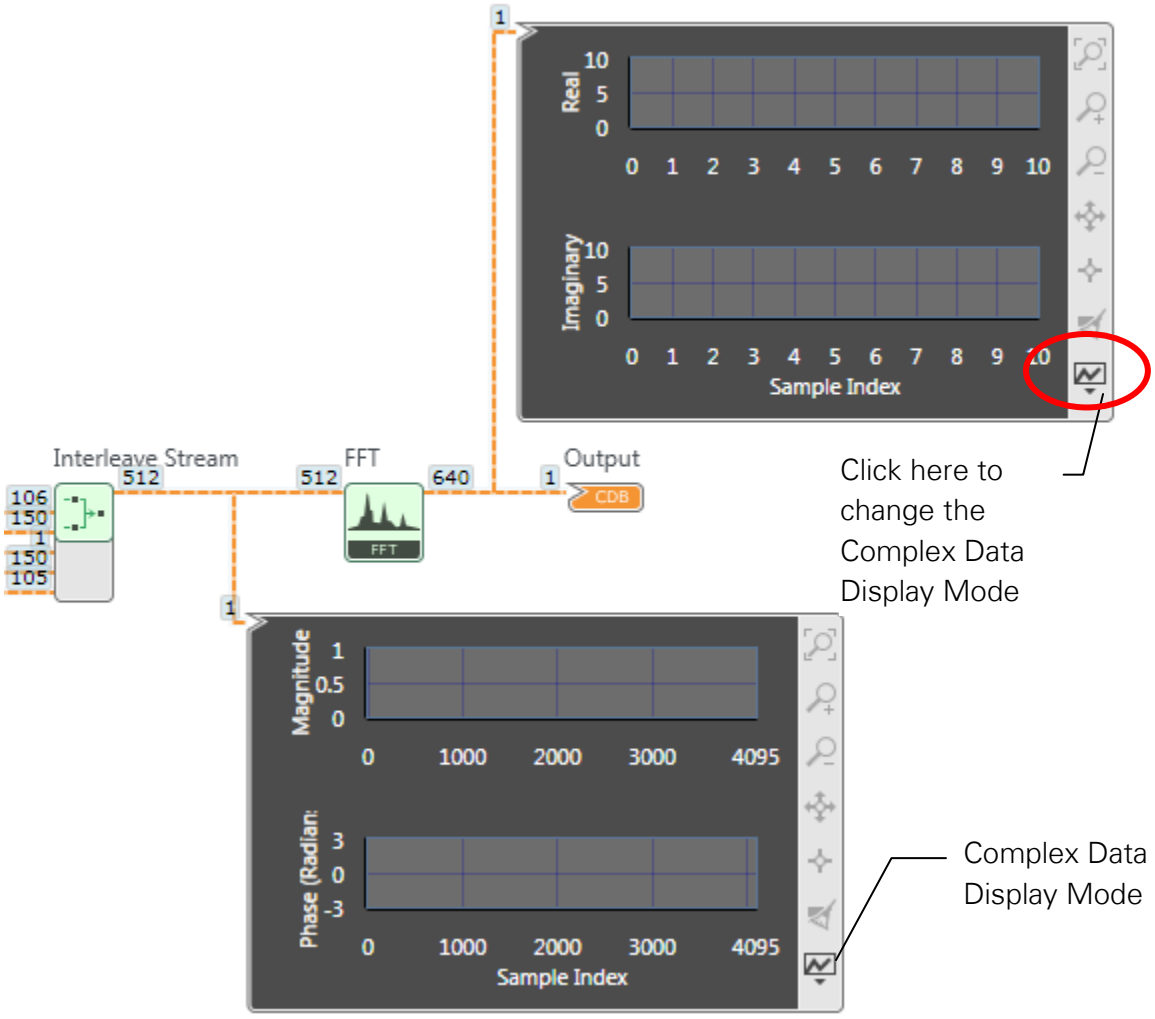
17. Click **OK**.

18. Wire the Interleave Stream function to the FFT then wire the FFT to the Output CDB terminal.



Using Chart Probes

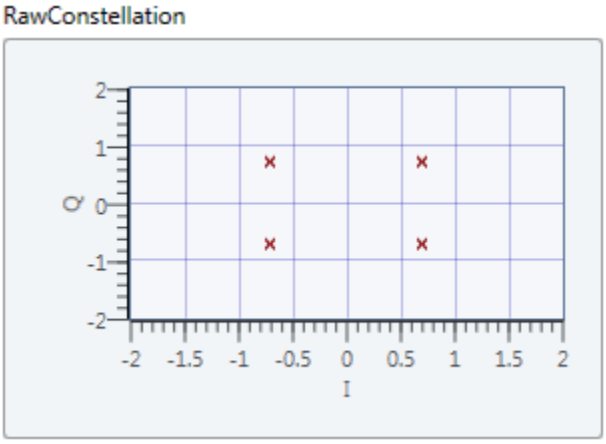
1. Place and wire a **Chart Probe** before the FFT.
2. Change the **Complex Data Display Mode** to **Magnitude/Phase (radians)**.
3. Place and wire a **Chart Probe** after the FFT.
4. Change the **Complex Data Display Mode** to **Real/Imaginary**.



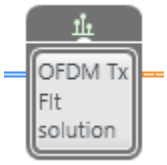
5. Run the MultiRate Diagram and inspect the chart probes.

Test the OFDM Algorithm

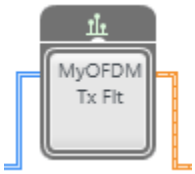
1. Open **Testbench OFDM flt.gvi**.
2. Run the testbench which includes a default solution included on the block diagram.
3. You should see a very nice constellation diagram.



4. Press <Ctrl-E> to switch to the diagram.
5. Find the default solution and delete it.



6. Drag and drop the **MyOFDM Tx Flt.gmrd** multirate diagram to the same location as the solution you just deleted.
7. Wire the input and outputs to the **MyOFDM Tx Flt.gmrd**.



8. Press <Ctrl-E> to switch to the Panel.
9. Run the testbench again and ensure the constellation plots do not change (and that your multirate diagram was created correctly).
10. Experiment with OFDM modulation under more realistic circumstances; check the Add Impairments checkbox. Explore how low you can set the signal to noise ratio before errors appear.

Please wait for the instructor before continuing

Exercise 3 – Fixed Point Conversion

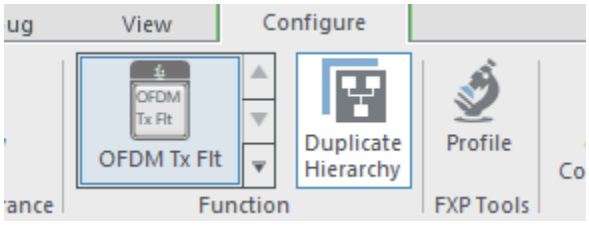
This lesson guides you through converting a floating-point algorithm design to a fixed-point design for implementation on an FPGA. You will use the floating-point OFDM modulator you created in the previous lesson to learn about fixed-point conversion.

1. Close Exercise 2 from **File » Close All**. You do not need to save any files.
2. Click **Getting Started»Overview of the FPGA»Fixed-Point Conversion**.
3. Click **Start** or **Start Over**.

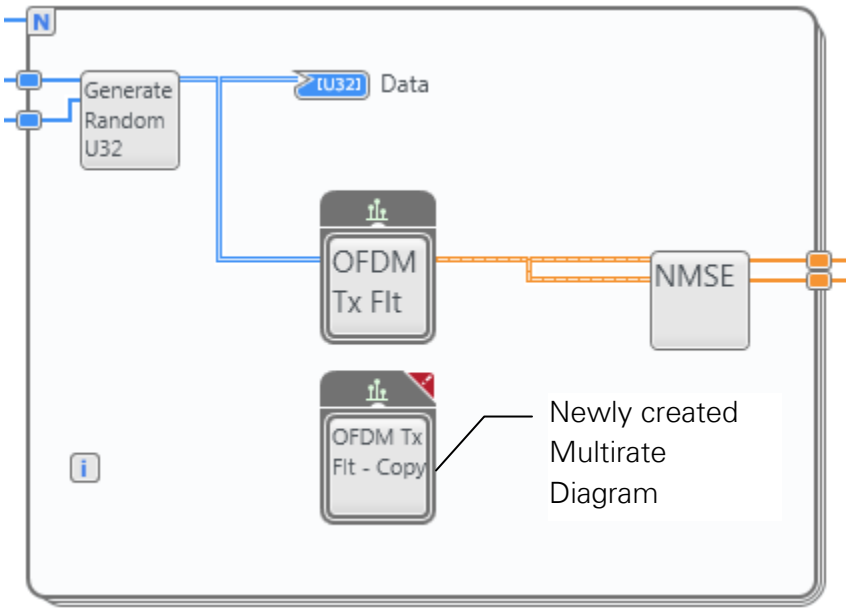
Collapse the Guided Help window using the collapse button.  You will follow this document instead of using Guided Help.

Duplicating a Hierarchy

1. Open **Testbench OFDM Fxp.gvi**.
2. Press <Ctrl-E> to switch to the diagram.
3. Click once on the OFDM Tx Flt multirate diagram.
4. From the **Configure** tab, click the **Duplicate Hierarchy** button.

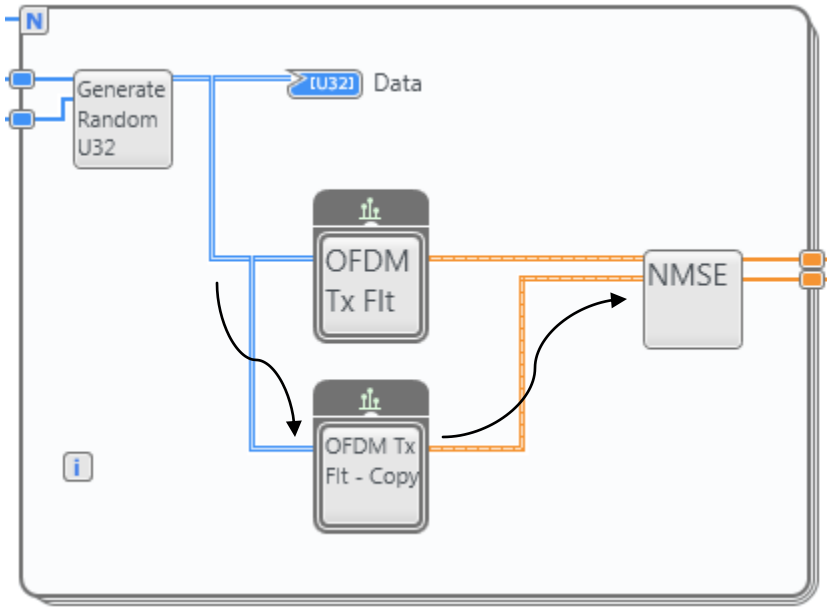


5. On the Duplicate Hierarchy dialog, click the **Duplicate and Place Multirate Diagram on Cursor** button.
6. Place your mouse pointer below the OFDM Tx Flt multirate diagram and create the duplicate here.



7. Wire the **Generate Random U32** function to the **OFDM Tx Flt Copy** multirate diagram.

8. Wire the output of **OFDM Tx Flt Copy** to the bottom input of NMSE.



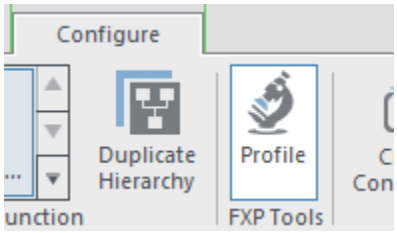
9. Press <Ctrl-E> to switch to the panel.
10. Run the testbench and observe the mean squared error.

Since we created an exact copy of the OFDM Tx Flt multirate diagram, there is no difference between the two and thus the error in dB is Infinity.

Profiling the Duplicated Hierarchy

1. Press <Ctrl-E> to switch to the diagram.
2. Click to select 

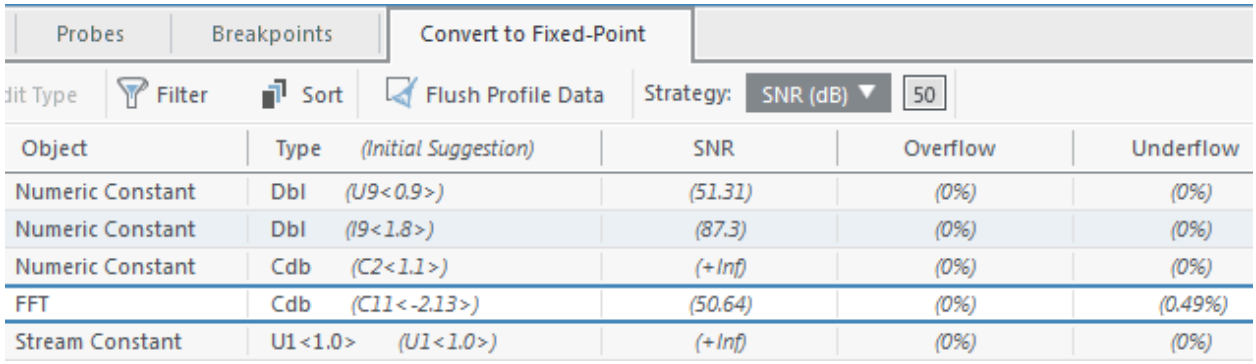
.
3. Click the **Profile** button from the configure ribbon.



4. Run the program.

The fixed-point conversion tool analyzes the diagram and suggests values that produce an output that meets the standard you set in Strategy. LabVIEW populates the table with suggested types.

LabVIEW predicts the signal-to-noise ratio (SNR) on the FFT to be 50.64 dB.



Converting Data Types to Fixed-Point

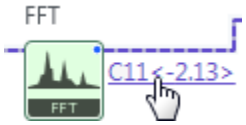
1. Click once inside the Convert to Fixed-Point table to select one of the rows.
2. Press <Ctrl-A> to select all rows in the table.
3. Click the **Convert Using Suggestion** button.
   LabVIEW converts the data types to the suggested fixed-point data types. LabVIEW will automatically insert a conversion node after each constant to preserve the original data type of constants.
4. Press **[Ctrl] + [E]** to switch to the panel, then run the program.

Compare the mean squared error (MSE) that the testbench calculated with the SNR in the table. The two values are almost identical. The fixed-point conversion tool uses the data that the tool stored when you initially profiled this diagram to calculate the SNR whereas the testbench simply runs both versions each time to calculate the MSE.

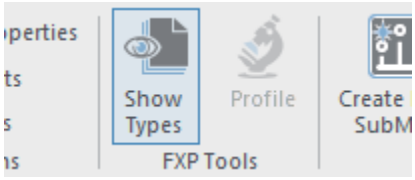
Fine-tuning the Fixed-Point Design

Because the initial suggested fixed-point data types exceeded the target by over 5 dB, you could stop here and continue onto implementing the converted design on an FPGA. However, in many cases, you may need to modify individual data types to meet your SNR target.

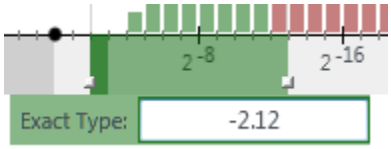
1. Press <Ctrl-E> to switch to the diagram.
2. Double-click to open the **OFDM Tx Flt Copy** multirate diagram.
3. Find the FFT function on the block diagram and click the output data type.



TIP: If you don't see the FFT output data type, click **Show Types** from the Multirate Diagram ribbon



4. Change Exact Type from -2.13 to **-2.12** and click **Apply**.



5. Press <Ctrl-S> to save the program.
6. Switch back to the Testbench OFDM Fxp front panel and run the program.

Notice that the Mean Squared Error increased from -55 dB to -50 dB. This is still acceptable since our target SNR was -50 dB.

In most cases, you would continue making adjustments and running your testbench until you are satisfied with the results.

Please wait for the instructor before continuing

Exercise 4 – Deploying an Application to an FPGA

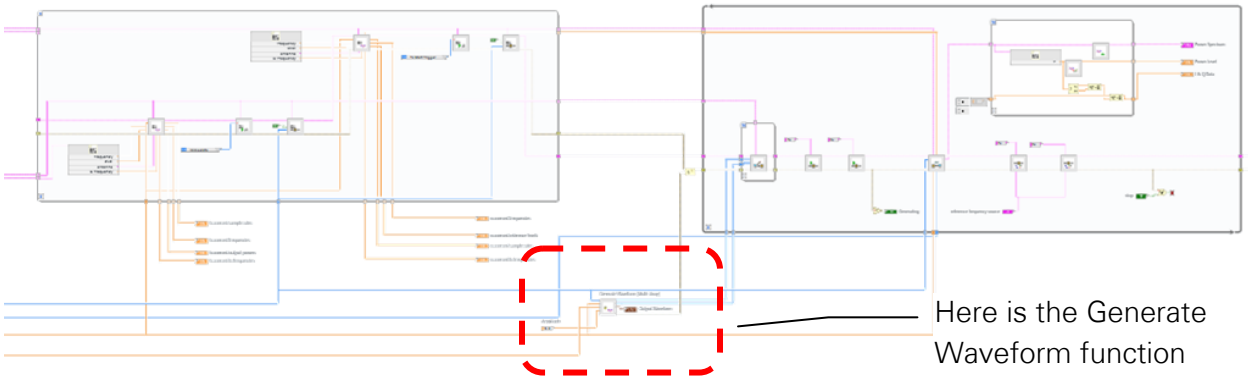
This lesson introduces you to the NI USRP RIO and one of the sample projects included in the LabVIEW Communications System Design Suite. In this lesson, you will integrate the OFDM modulator you created previously into a sample project, target the OFDM modulator to the Xilinx FPGA onboard the NI USRP RIO, complete the design flow by performing design space exploration, and compile the build specification and deploy the application on the FPGA target.

1. Close Exercise 3 from **File » Close All**. You do not need to save any files.
2. Click **Getting Started»Overview of the FPGA»Deploying an Application on an FPGA**.
3. Click **Start** or **Start Over**.

Collapse the Guided Help window using the collapse button.  You will follow this document instead of using Guided Help.

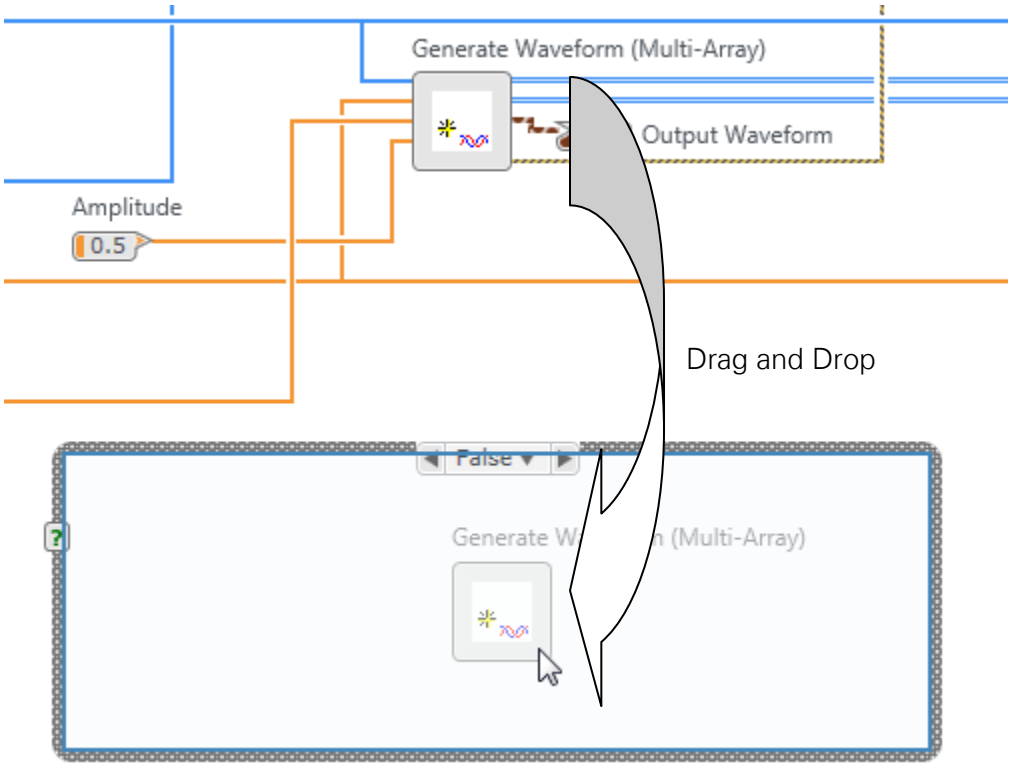
The NI USRP Streaming Sample Project

1. Open **Full Duplex Streaming (Host).gvi** from the Files pane.
2. Press <Ctrl-E> to switch to the diagram.
3. Find the **Generate Waveform (Multi-Array)** function near the center of the diagram.

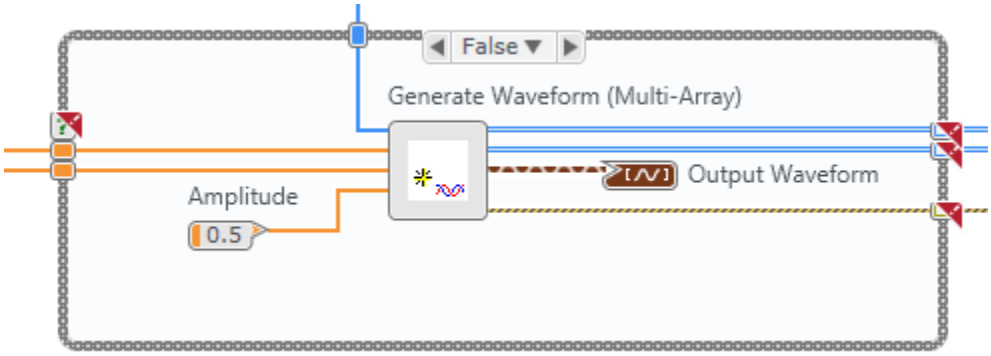


4. Place a **Case Structure** on the diagram below this function.

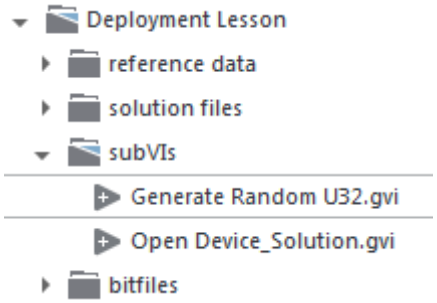
5. Switch to the **False** case and drag the **Generate Waveform (Multi-Array)** function into the Case Structure.



6. Drag **Output Waveform** and **Amplitude** into the Case Structure.

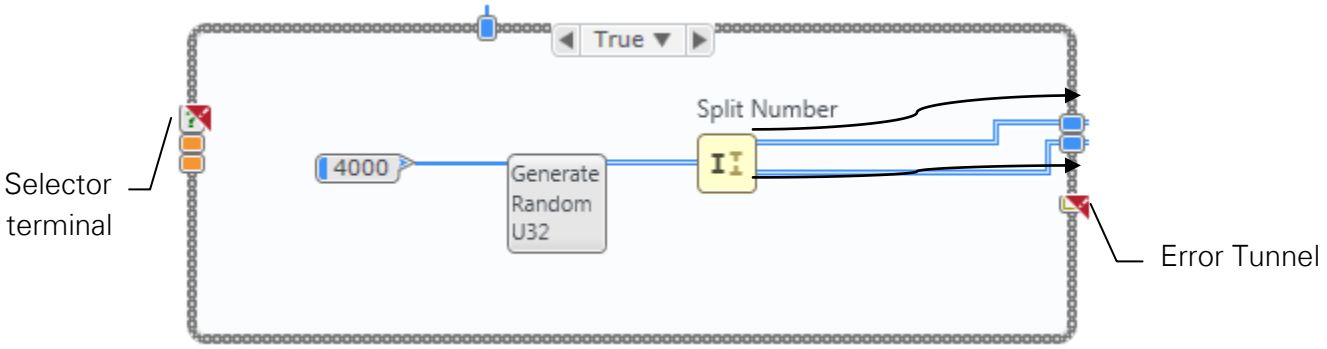


7. Switch to the **True** case.
8. Add the **Generate Random U32** function to the Case Structure from the Files pane.

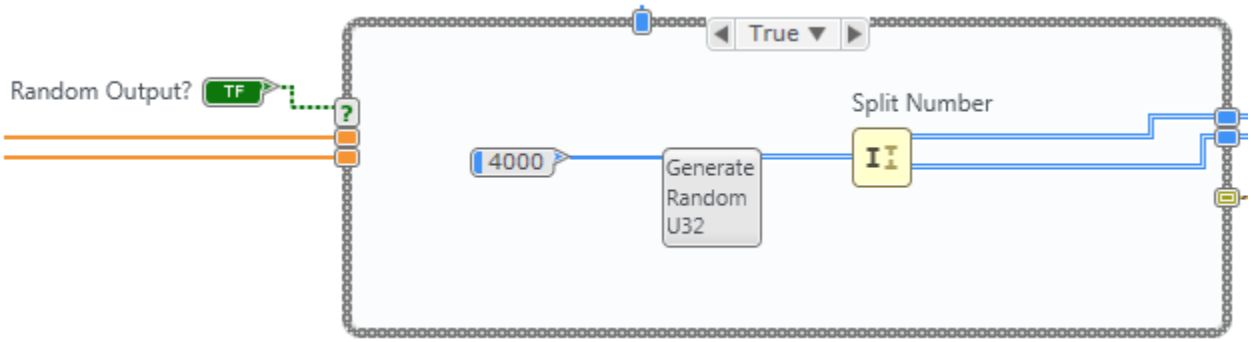


9. Create a **numeric constant** by right-clicking the number of samples input on the **Generate Random U32** function and use a value of **4000**.
10. Add and wire a **Split Number** function after the Generate Random U32 function.

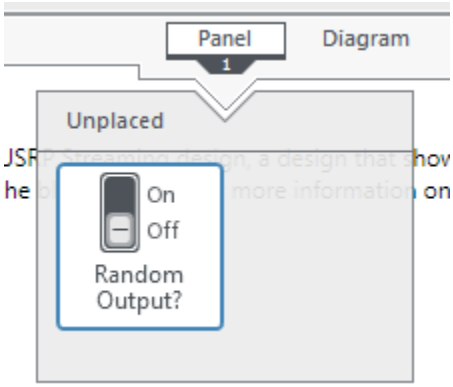
11. Wire the **Split Number** to the output tunnels as follows.



12. Right-click the error tunnel in the Case Structure and select **Default If Unwired**.
13. Right-click the selector terminal and select **Create Control**.
14. Name the control **Random Output?**.

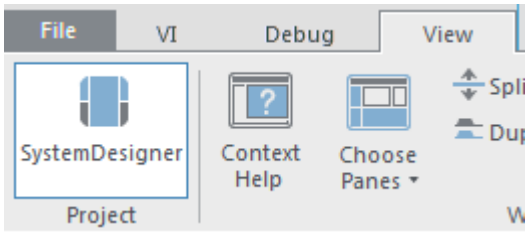


15. Press <Ctrl-E> to switch to the panel.
16. Place the **Random Output?** Control somewhere on your panel.

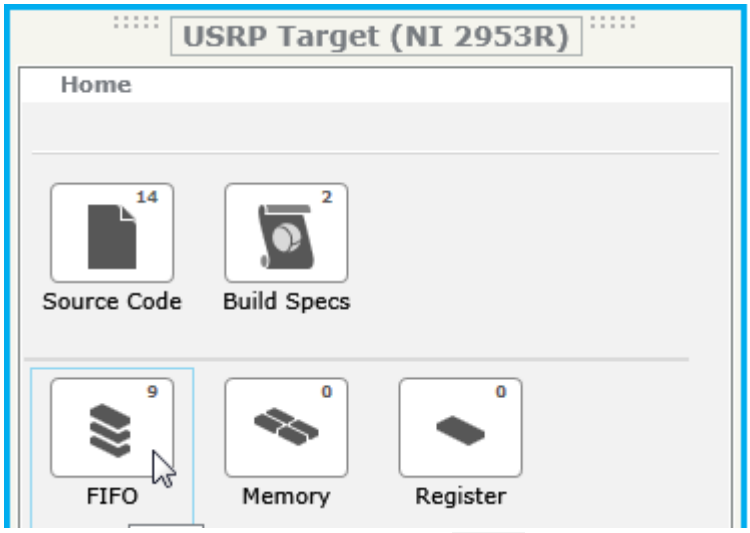


Create a new FIFO using SystemDesigner

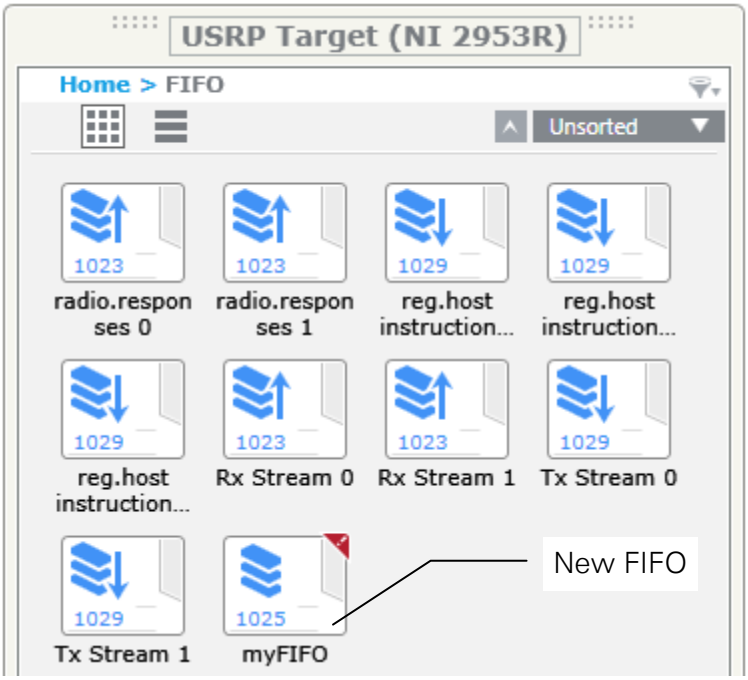
1. From the View ribbon, click on the **SystemDesigner** button.



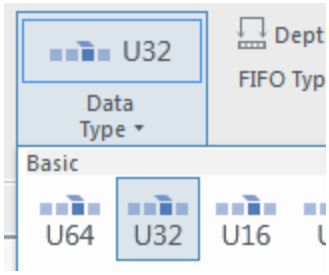
2. In the SystemDesigner, find the USRP Target and navigate to **Home»FIFO**.



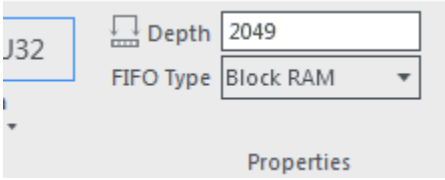
3. Create a new FIFO by clicking the  button.
4. Name the FIFO **myFIFO** and press <Enter>.



5. While myFIFO is selected, use the Configure ribbon to change the Data Type to **U32**.

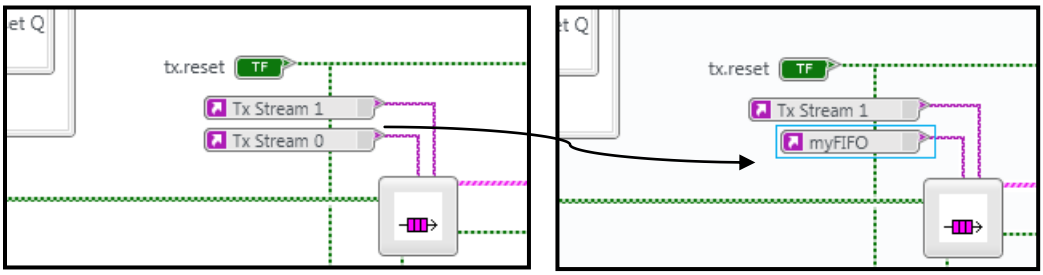


6. Change the **Depth** property to 2049.

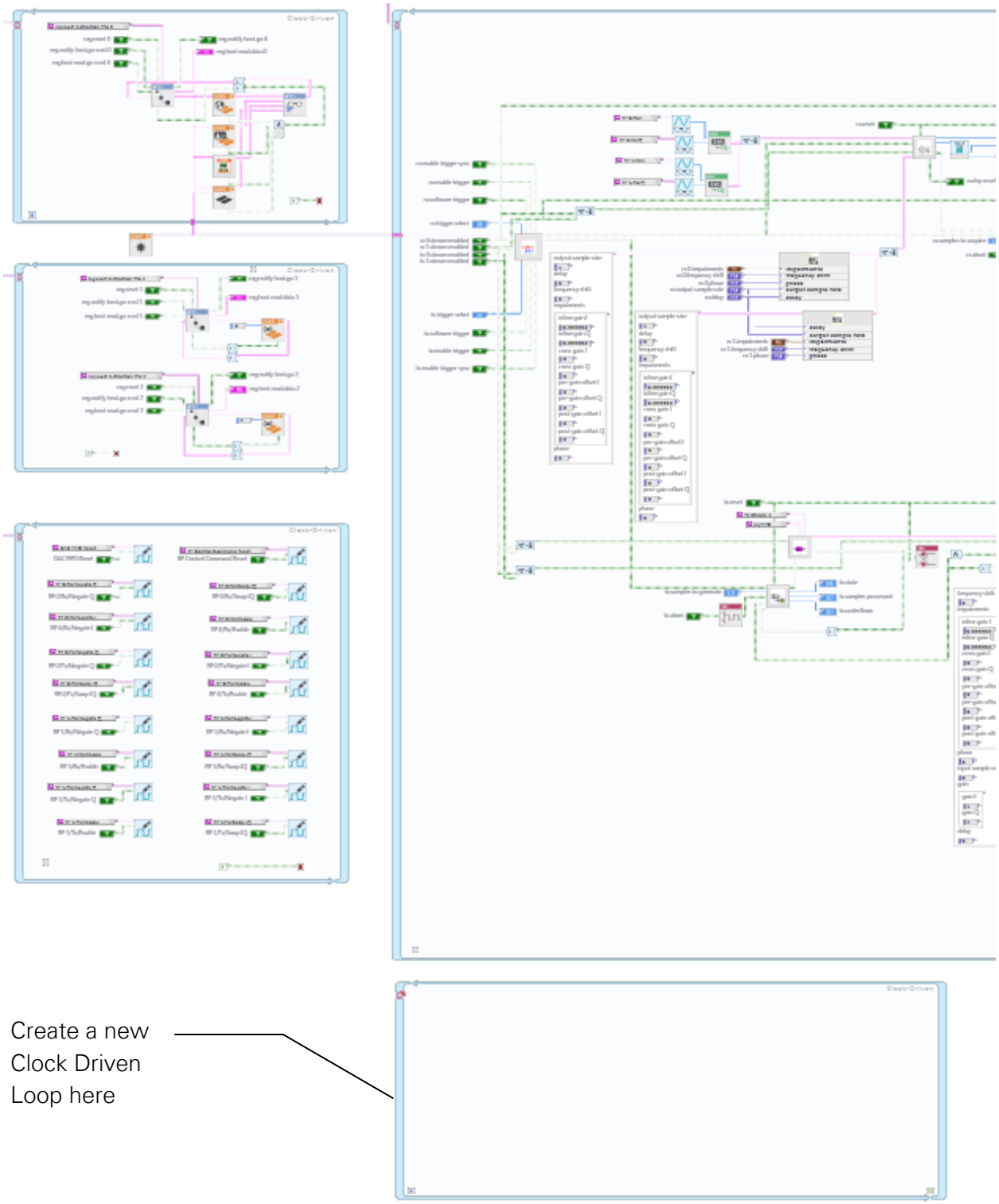


Modifying the Streaming Xcvr FPGA Program

1. Open **Streaming Xcvr FPGA.gvi** from the Files pane.
2. Find the **Tx Stream 0** FIFO constant and change it to **myFIFO** using the drop-down.

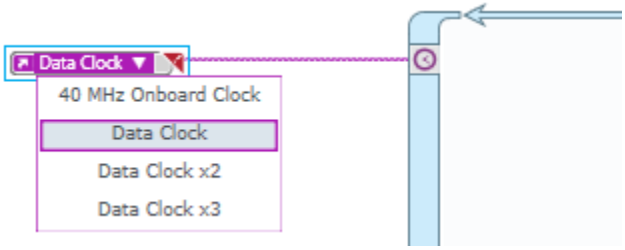


3. Scroll down to the bottom of **Streaming Xcvr (FPGA)** and create your own **Clock-Driven Loop**.

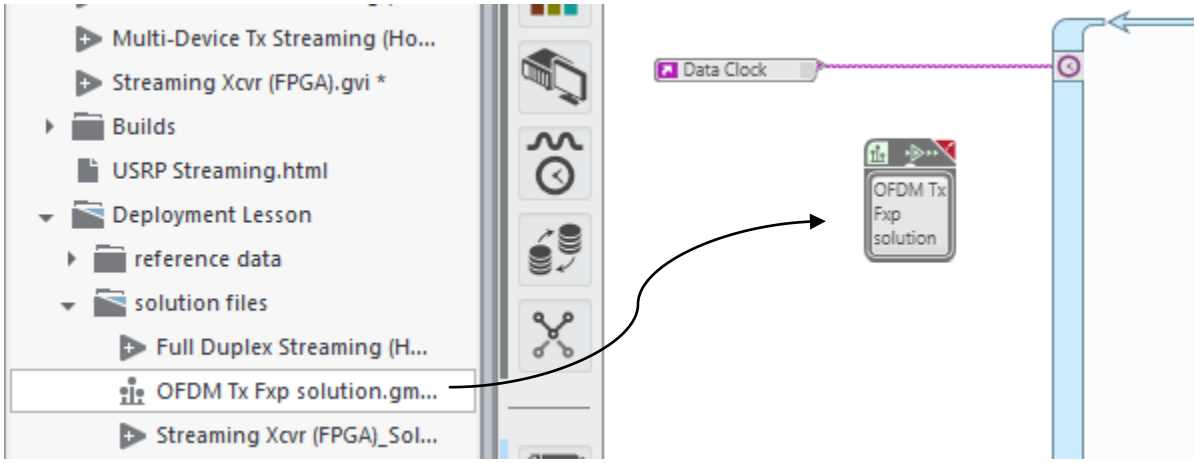


4. On the left side of the new Clock Driven Loop, right-click on the clock icon and **create a constant**.

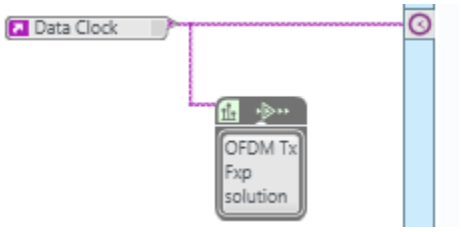
5. Change the constant to **Data Clock** using the drop-down.



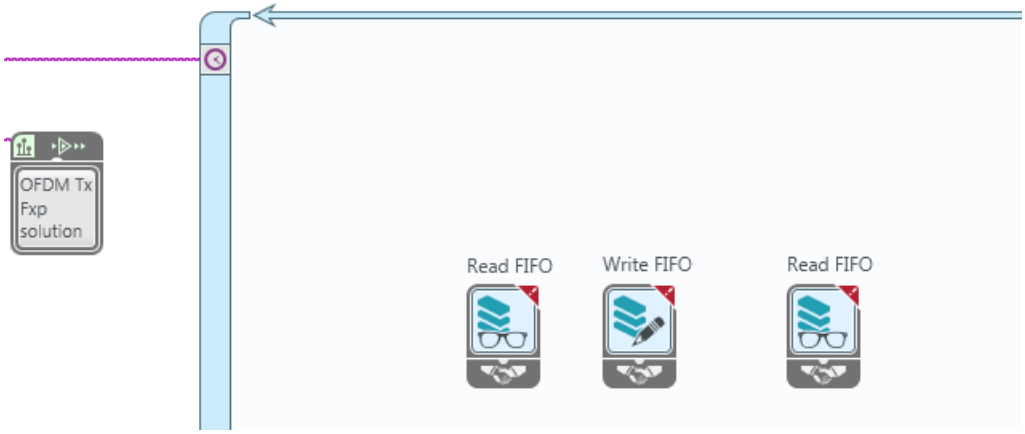
6. Drag and drop **OFDM Tx Fxp solution.gmrd** from the files pane to the **Streaming Xcvr (FPGA)** Diagram.



7. Wire the Data Clock constant to the **Clock** input of the **OFDM Tx Fxp Solution** multirate diagram.

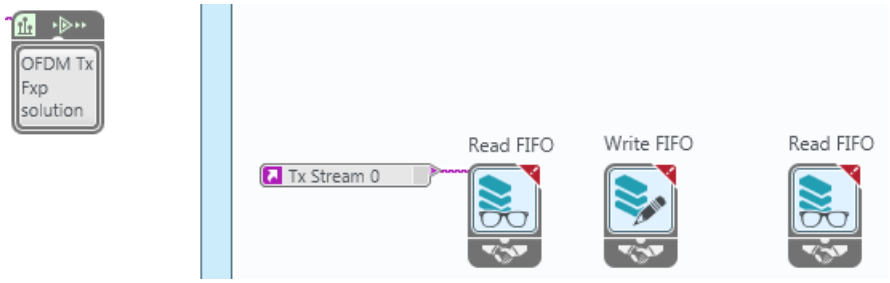


8. Inside the **Clock Driven Loop**, place a **Read FIFO**, **Write FIFO**, and a **Read FIFO** in that order.

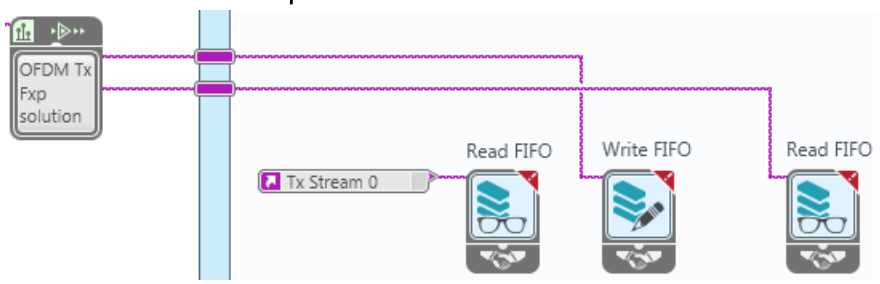


9. Create a **reference in** constant for the left-most Read FIFO.

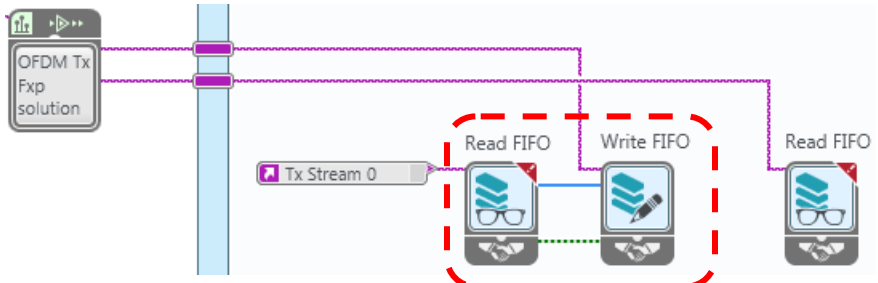
10. Using the drop down, change the constant to **Tx Stream 0**.



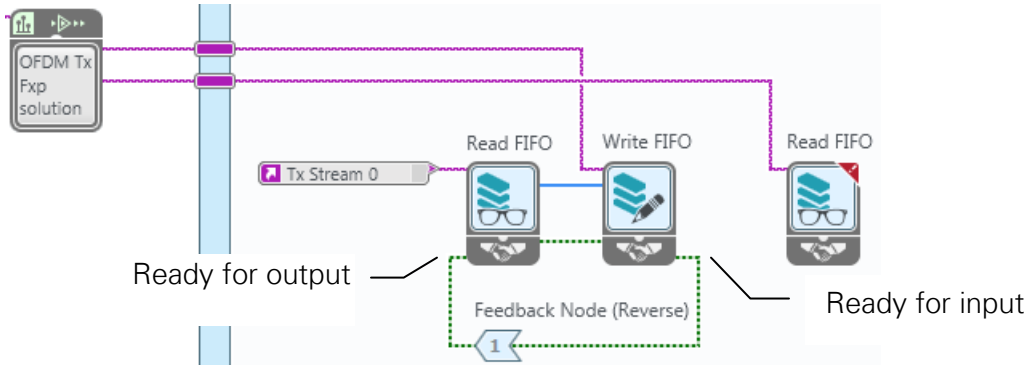
11. Wire the **OFDM Tx Fxp solution** to **Read FIFO** and **Write FIFO** functions.



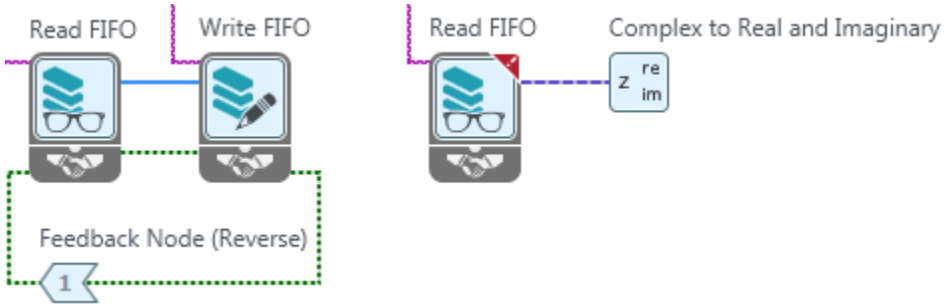
12. Wire the **data** and **output valid** terminals from Read FIFO to the inputs of Write FIFO.



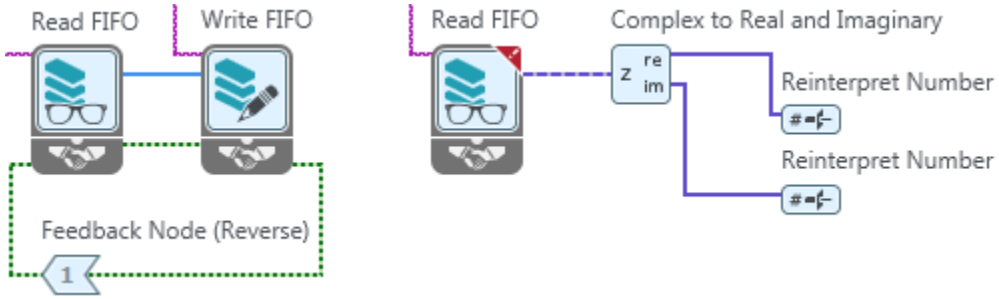
13. Insert a **Feedback Node** below the functions and wire the **ready for input** and **ready for output** terminals.



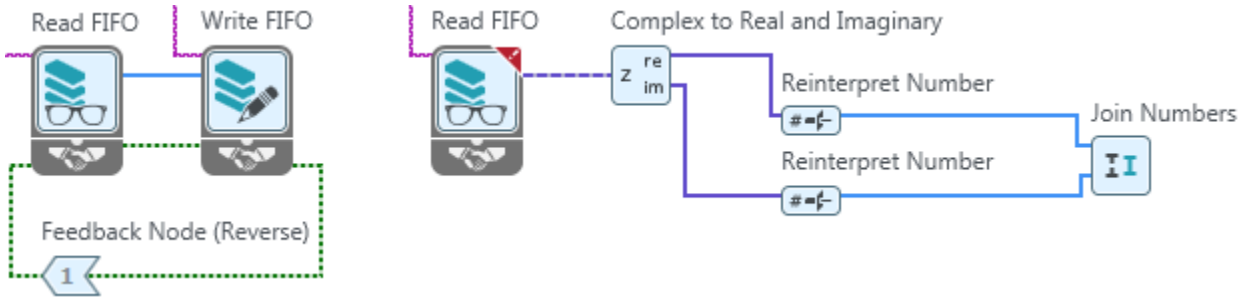
14. Add a **Complex to Real and Imaginary** to the last **Read FIFO** function.



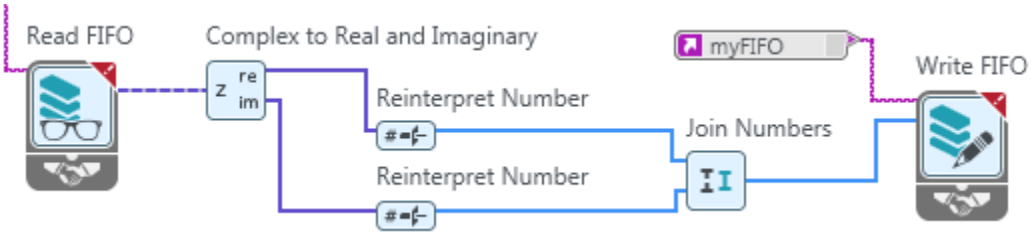
15. Add and wire two **Reinterpret Number** functions for the real and imaginary outputs.



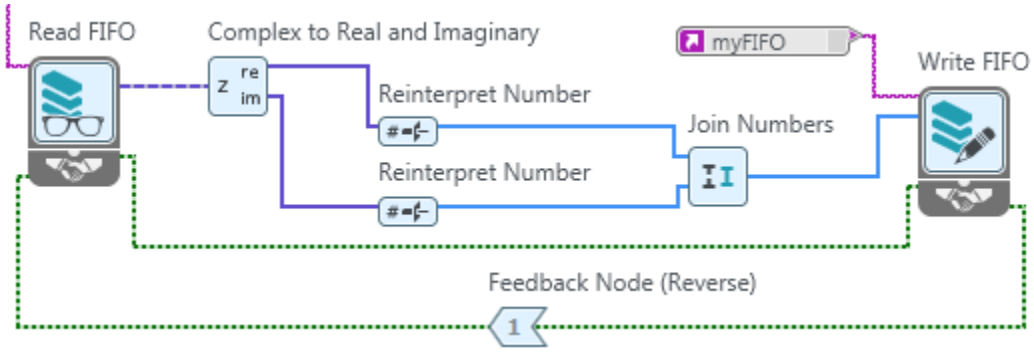
16. Add and wire a **Join Numbers** function to combine the reinterpreted numbers.



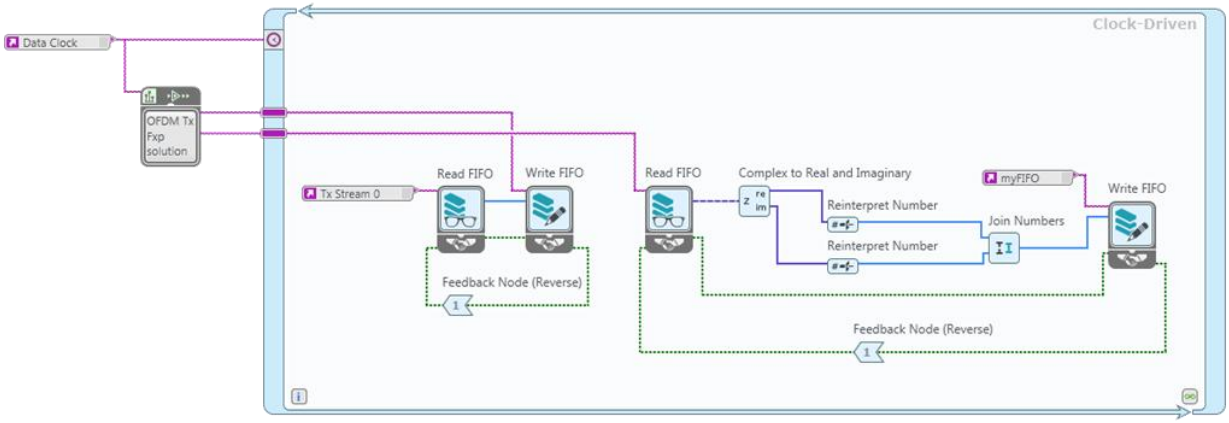
17. Place a **Write FIFO** function after the Join Numbers function
18. Wire the output of **Join Numbers** to the **data input** of the Write FIFO function
19. Create a **FIFO constant** for the Write FIFO function and select myFIFO from the drop-down list.



20. Place and wire **Feedback Node, ready for input, ready for output** and **data valid** terminals.

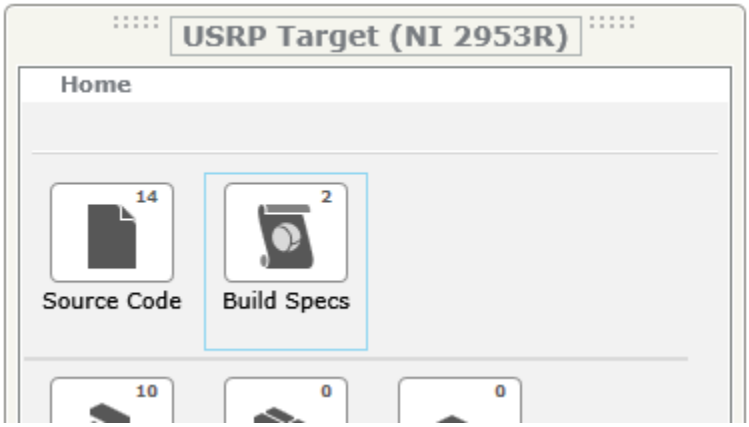


Your finished Clock Driven Loop should look like the following when completed

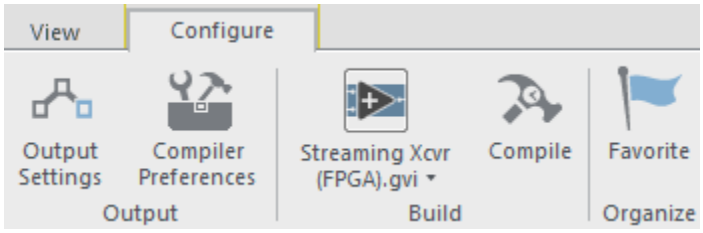


Compile the FPGA Build Specification

1. Return to the SystemDesigner.
2. In the USRP Target, double click to view the **Build Specifications**.



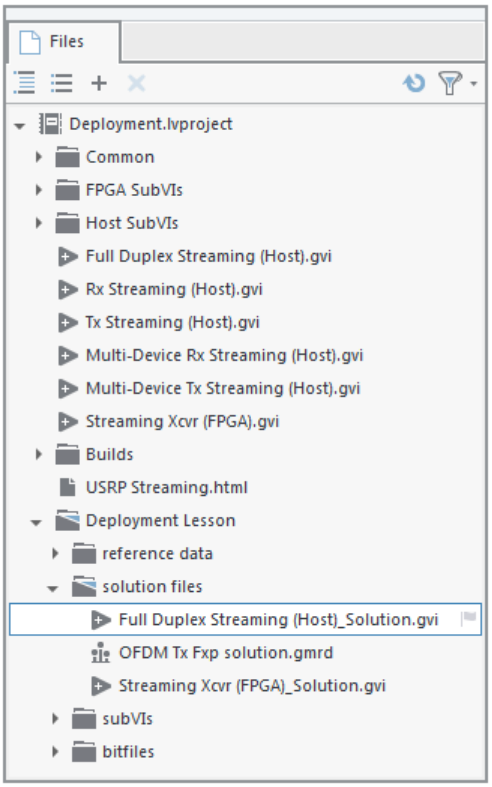
3. Click once on the left-most Build Spec and inspect the Configure ribbon.



Normally you would compile the FPGA bitfile using the build specifications but this process can be lengthy.

For the purposes of this seminar, you will utilize the pre-compiled bit file by opening the solution program which includes a precompiled bitfile.

4. From the files pane, navigate to and open **Deployment Lesson»solution files»Full Duplex Streaming (Host)_Solution.gvi**.



5. Run the program with the Run button  .

Please wait for the instructor before continuing

Solutions to Exercises

Solutions to all exercises can be found in the LabVIEW Communications Guided Help feature.

To access the Guided Help for these exercises:

1. Open LabVIEW Communications.
2. Click Learn»Getting Started»Overview of the FPGA Design Flow.
3. Select an exercise and find the solution folder in the Files pane.