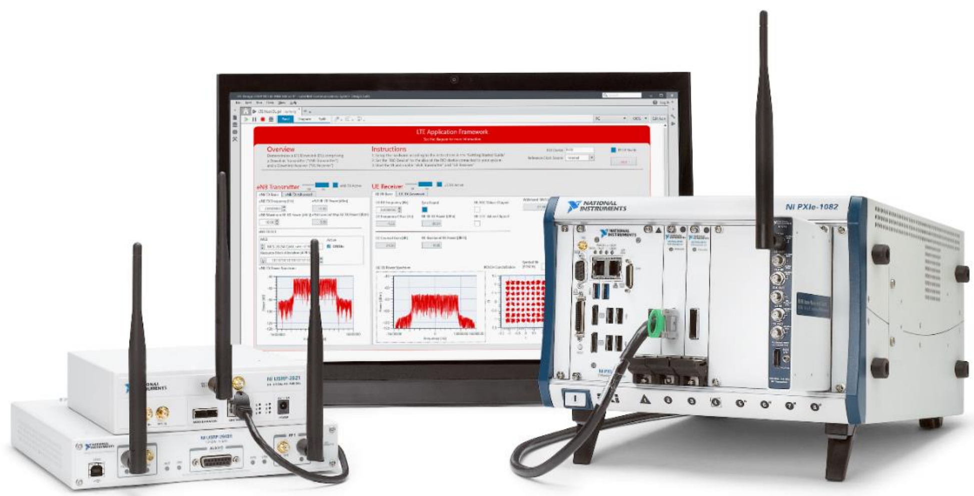
SOFTWARE DEFINED RADIO HANDS ON: FPGA PROTOTYPING WITH OVER-THE-AIR SIGNALS

Version 4.0



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

Q1 2016 Edition
Part Number 351260C-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 Communications, 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 six 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. You will also understand how to use this workflow to modify open, modular PHY IP for LTE and 802.11

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
- Want to start with standards-compliant PHY references designs for LTE and/or 802.11

Time Required to Complete the Course

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

Required Background

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. If you are new to LabVIEW, the Introduction to the LabVIEW Editor lesson offers a general overview of the LabVIEW Communications environment.

To find it, open LabVIEW Communications System Design Suite, select **File » Learn » Getting Started**.

Required Equipment

Hardware

- USRP RIO transceiver (such as 2940R)
- USRP power adapter
- USRP RIO connectivity kit
- SMA loopback cables with 30 dB attenuator
- RH901S or other whip antenna

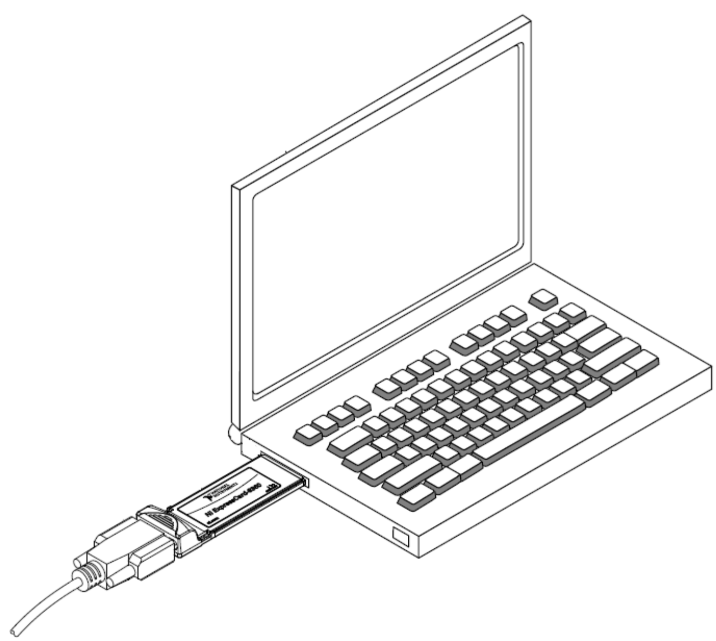
Software

- LabVIEW Communications System Design Suite 2.0 or later

Configuring Hardware

Follow these steps on your computer:

1. Install LabVIEW Communications. This will install the necessary drivers.
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

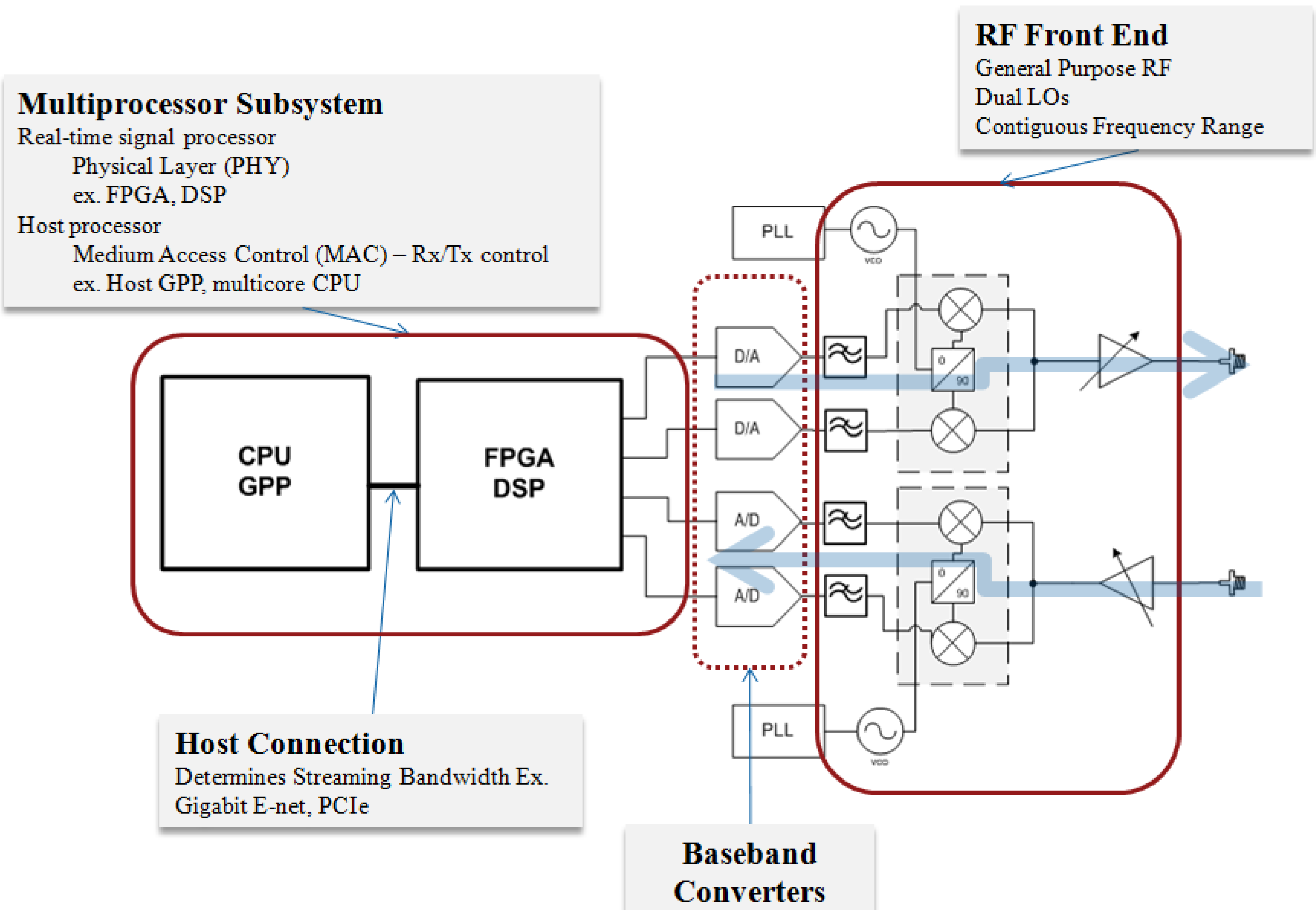


What Is a Software Defined Radio?

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 Communications 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.



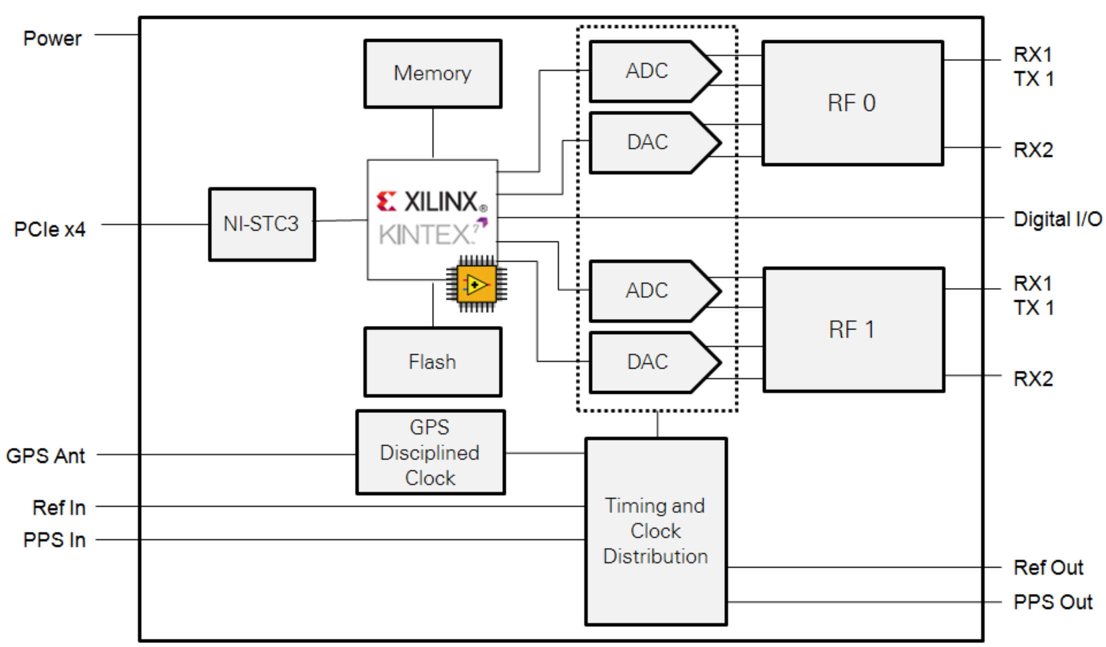
[1] http://www.sdrforum.org/pages/documentLibrary/documents/SDRF-06-R-0011-V1_0_0.pdf

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 Communications 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.

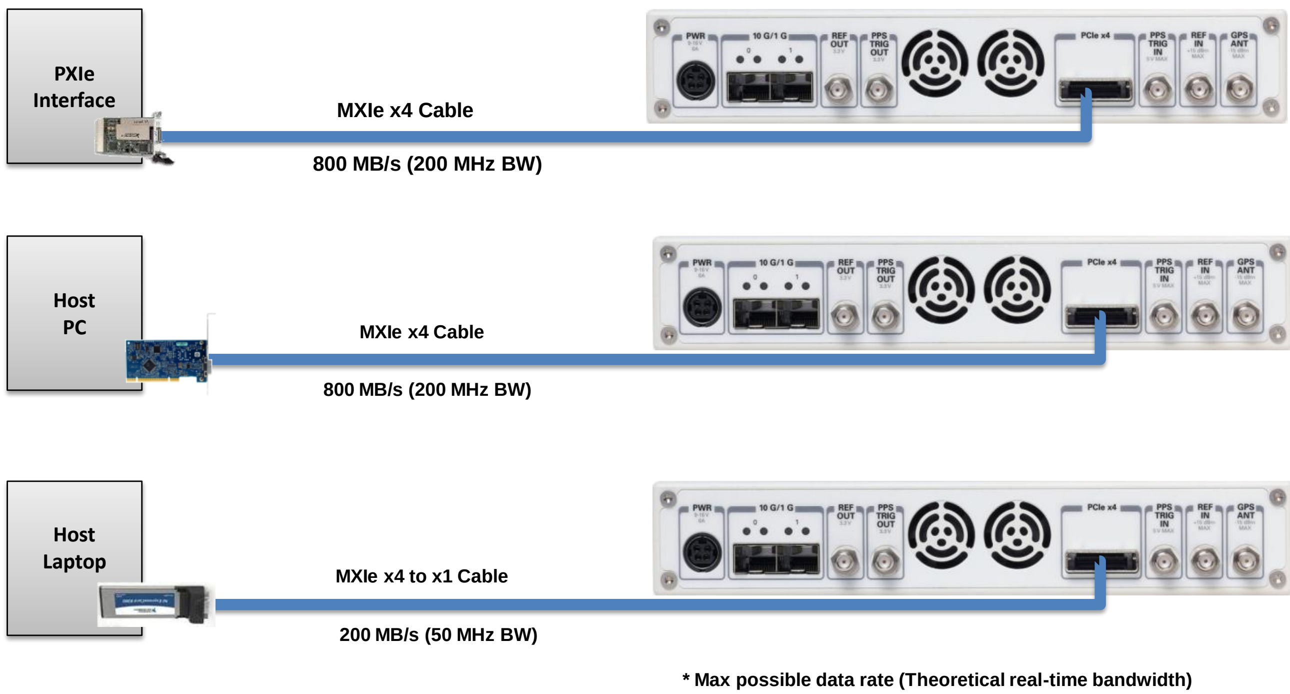
The USRP RIO combines two full-duplex transmit-and-receive channels with 40 MHz per channel of real-time bandwidth and a large DSP-oriented Kintex-7 FPGA in a half-1U rack-mountable form factor. The analog RF front end interfaces with the large Kintex-7 410T FPGA through dual analog-to-digital converters (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 eight different USRP RIO devices with frequency options that span from 10 MHz to 6 GHz and user-programmable digital I/O lines for controlling external devices. The Kintex -7 FPGA is a reconfigurable LabVIEW target that incorporates DSP48 co-processing 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

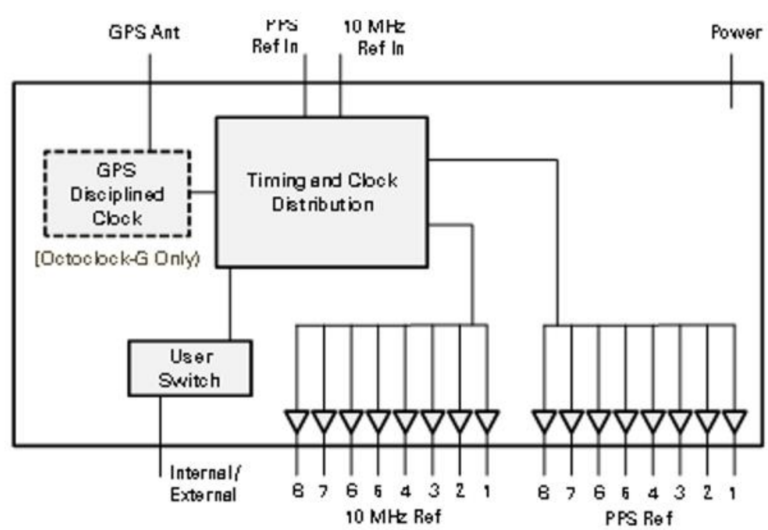
The primary interface bus for the USRP RIO is PCI Express x4, which provides an effective connection for high-bandwidth and lower latency applications such as PHY/MAC applications. With the PXI Express x4 bus, you can stream data at up to 800 MB/s and customize the FPGA in LabVIEW Communications. The 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.

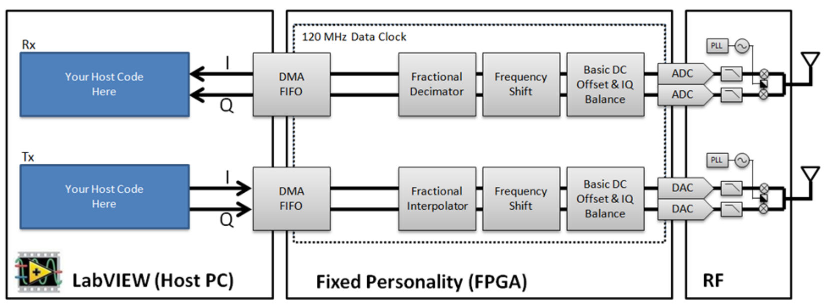


Application Programming Interface (API)

The USRP RIO takes advantage of two complementary LabVIEW Communications-based software driver experiences: a host-based driver (NI-USRP) and a fully open and customizable 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.

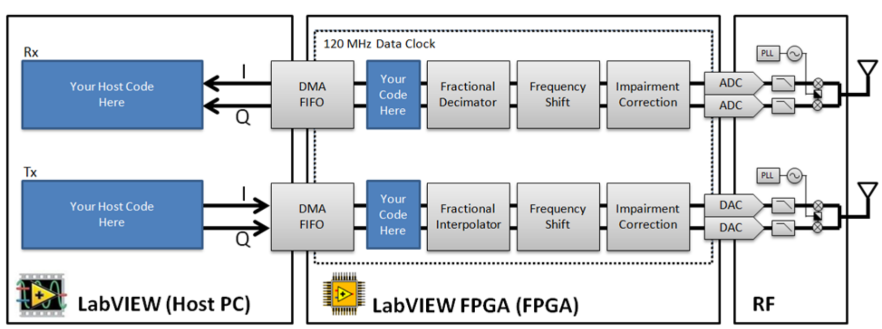
NI-USRP Driver

NI is releasing USRP RIO hardware with NI-USRP 15.5 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 Communications 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 co-processing 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 using technologies in LabVIEW Communications. 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.

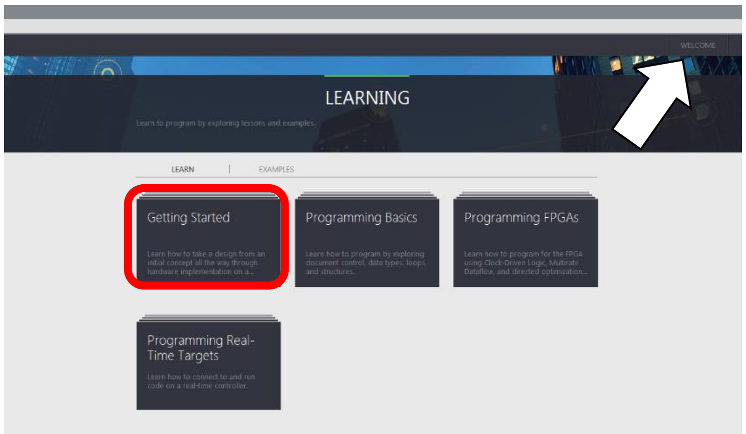


EXERCISE 1 – Demodulating an FM Signal with MathScript

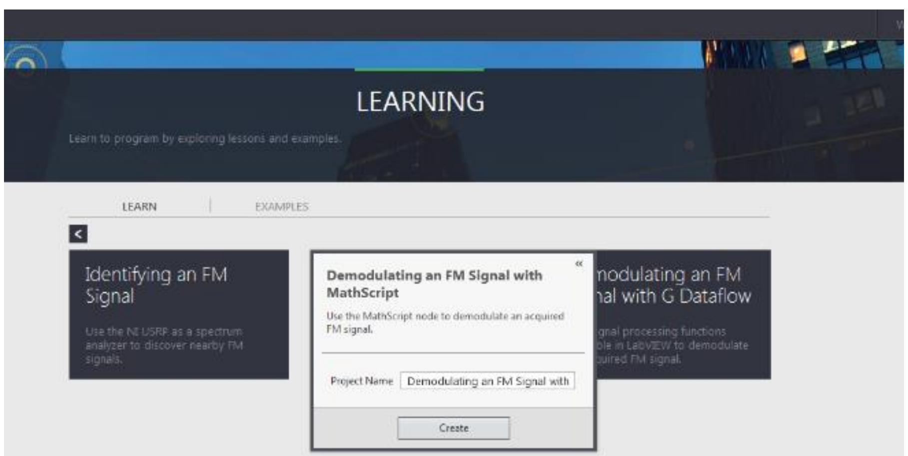
Part A— CODE IMPLEMENTATION

1. Open the Demodulating an FM Signal with MathScript example.

A



B



DETAILED INSTRUCTIONS

In this exercise, use the ***Demodulating an FM Signal with MathScript*** example as the starting point of the application.

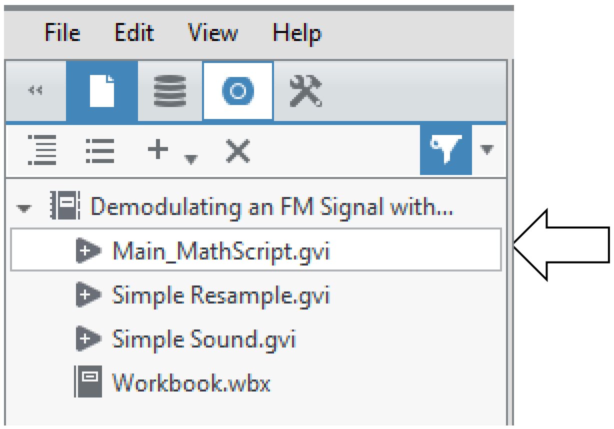
- ***Launch NI LabVIEW Communications 2.0***
  Launch LabVIEW Communications 2.0 by navigating to ***Start » All Programs » National Instruments » LabVIEW Communications 2.0» LabVIEW Communications 2.0.***

- ***Open the Example Project***
  Under the ***Learning*** tab, find the project by navigating to ***Getting Started» Demodulating FM Signals with the NI… » Demodulating an FM Signal with MathScript*** as shown in ***Figure A.***

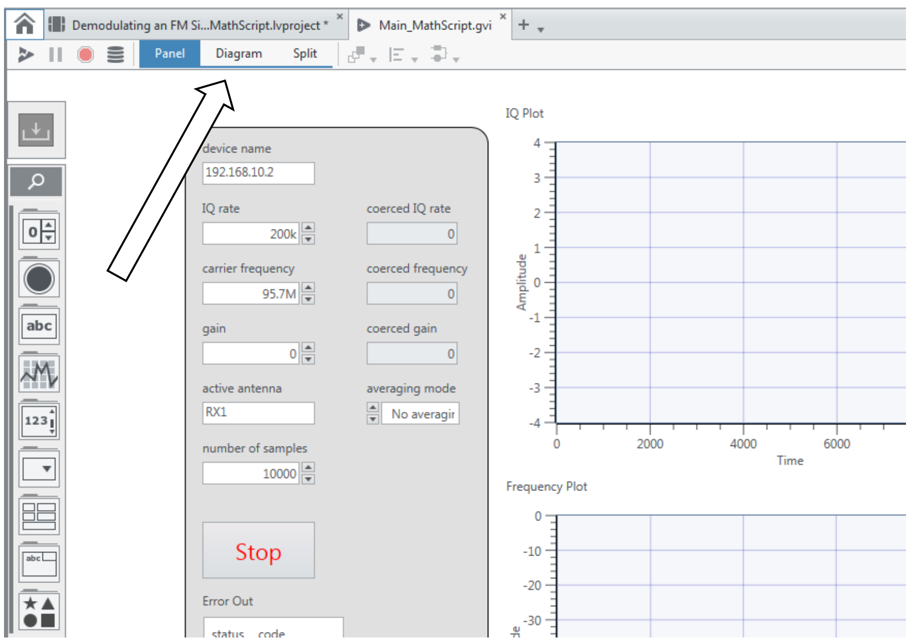
- ***Save the Example Project***
  Provide a name in **Project Name** and click ***Create*** as shown in ***Figure B***.

2. Explore the Main VI.

A



B



DETAILED INSTRUCTIONS

- ***Close the Workbook tab and follow this manual***
  Click the X on the Workbook tab. On your own time, you can follow these tutorials for guided learning, but today we will use this manual.

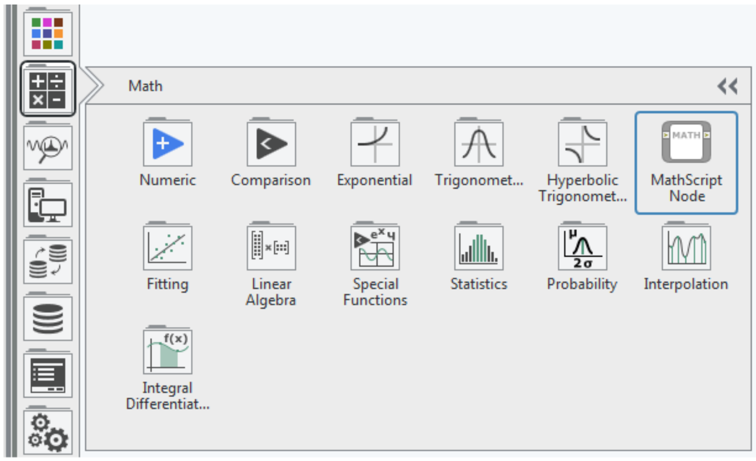
- ***Open the Main MathScript VI***
  Double click ***Main_MathScript.gvi*** as shown in ***Figure A.***

- ***Explore the Panel and the Diagram***
  When you open up Main_MathScript.gvi, it displays the Panel, or user interface. Switch to the Diagram to show the source code by pressing <Ctrl-E> on your keyboard or clicking ***Diagram*** as shown in ***Figure B***.

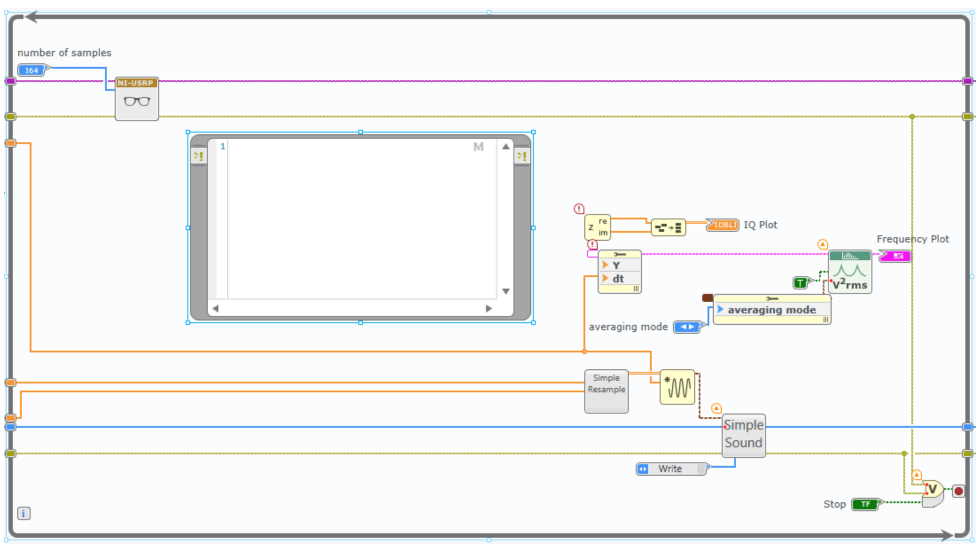
  Note the 6 function blocks titled **NI-USRP**. These **open**, **configure**, and **start** your receive session, **read data** off the buffer, then **stop** the session. The **read** function is in a While Loop, and will continuously pull bits from the hardware. This is where our FM demodulator will go.

3. Add a MathScript Node.

A



B



DETAILED INSTRUCTIONS

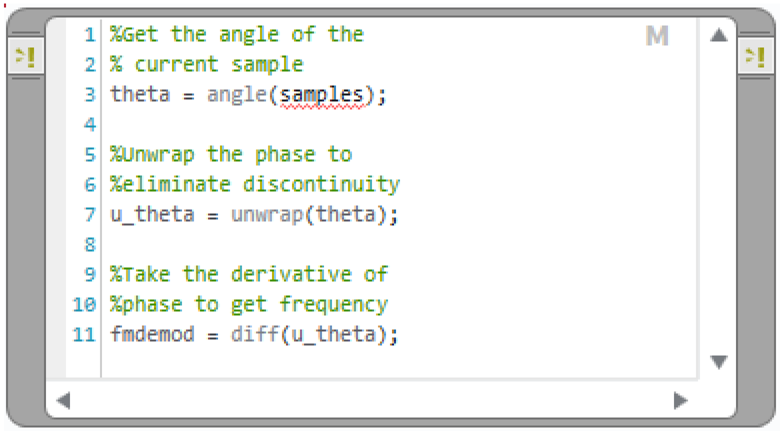
You can use the MathScript Node to implement any of your custom .m files into the LabVIEW Communications environment. The diagram for the Main MathScript VI already includes the nodes and subVIs necessary to resample, graph, and listen to an FM radio signal after the MathScript Node acquires the signal.

- ***Select the MathScript Node***
  Navigate to the Math palette as shown in ***Figure A***.

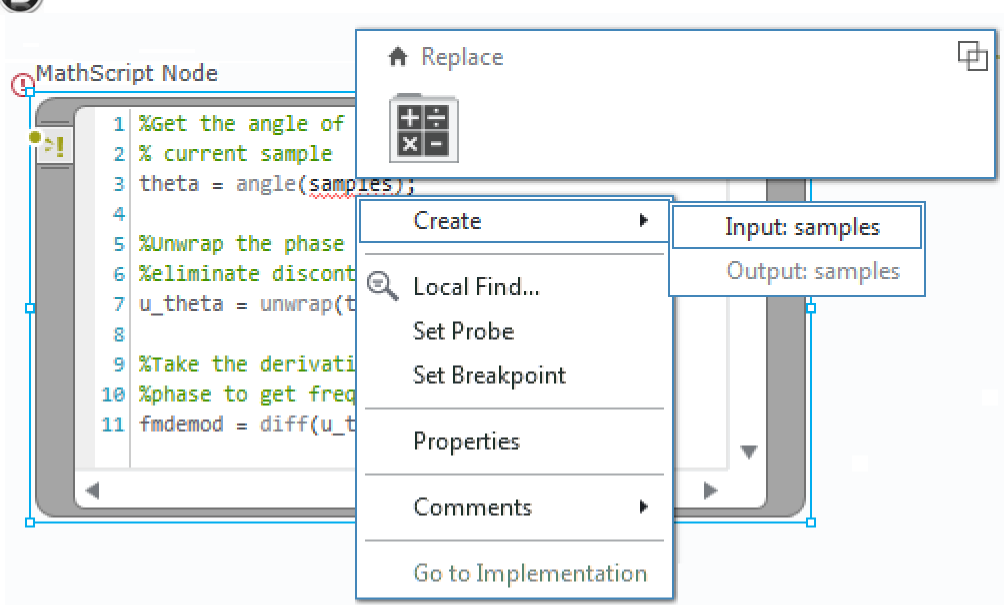
- ***Place the MathScript Node inside the While Loop***
  Click the diagram inside the While Loop, and drag the cursor to create a MathScript Node as shown in ***Figure B***.

4. Code the MathScript Node.

A



B



DETAILED INSTRUCTIONS

- ***Add .m script into the MathScript Node***
  Type the following .m script into the MathScript Node:

```
%Get the angle of the
%current sample
theta = angle(samples);

%Unwrap the phase to
%eliminate discontinuity
u_theta = unwrap(theta);

%Take the derivative of
%phase to get frequency
fmdemod = diff(u_theta);
```

  Your MathScript Node should now resemble the image in ***Figure A.***

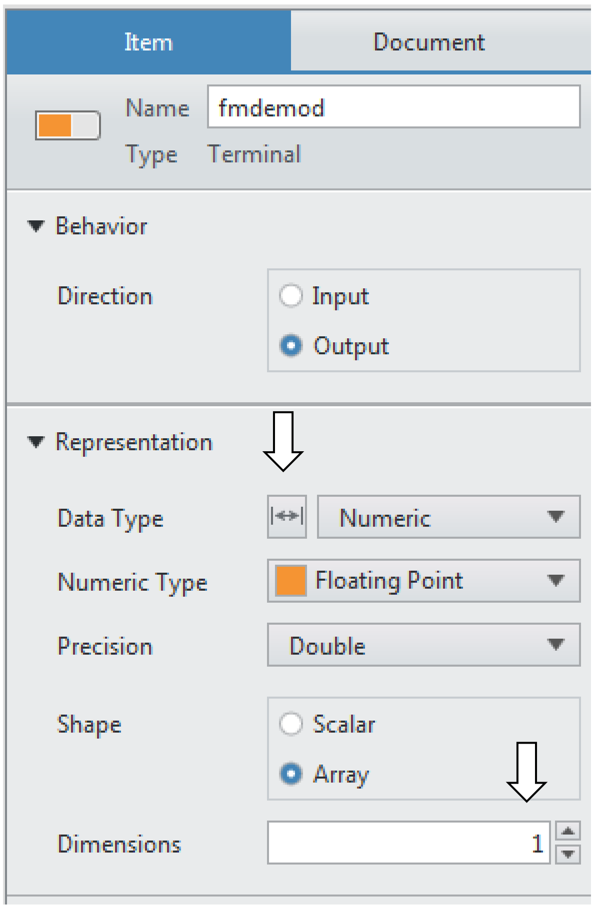
- ***Create an input to the MathScript Node***
  Right click the variable *samples* and select ***Create » Input: samples*** as shown in ***Figure B***.

- ***Create an output to the MathScript Node***
  Right click the variable *fmdemod* and choose ***Create » Output: fmdemod***.

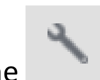
5. Configure variables inside the MathScript Node.

A

B

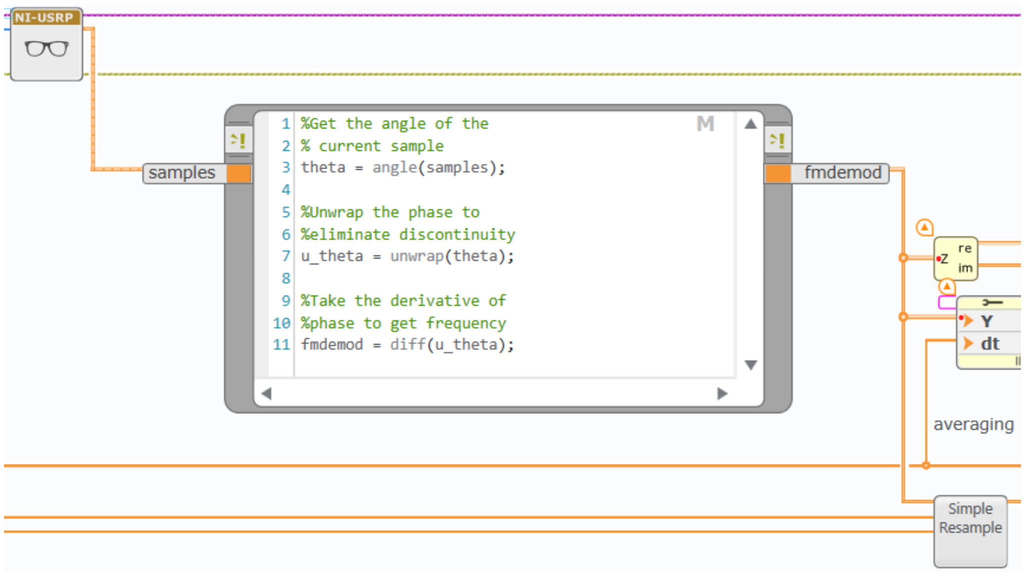


DETAILED INSTRUCTIONS

- ***Select the fmdemod output***
  Click the orange box next to ***fmdemod*** as shown in ***Figure A.*** If you collapsed the Item pane, click on the  in the upper-right corner.

- ***Configure the fmdemod representation***
  On the ***Item*** tab. Under **Representation,** next to **Data Type** click the double-arrow icon next to the data type selector to prevent the data type from adapting automatically. Change ***Dimensions*** to 1. Your configuration should match ***Figure B***.

6. Integrate the MathScript Node.



DETAILED INSTRUCTIONS

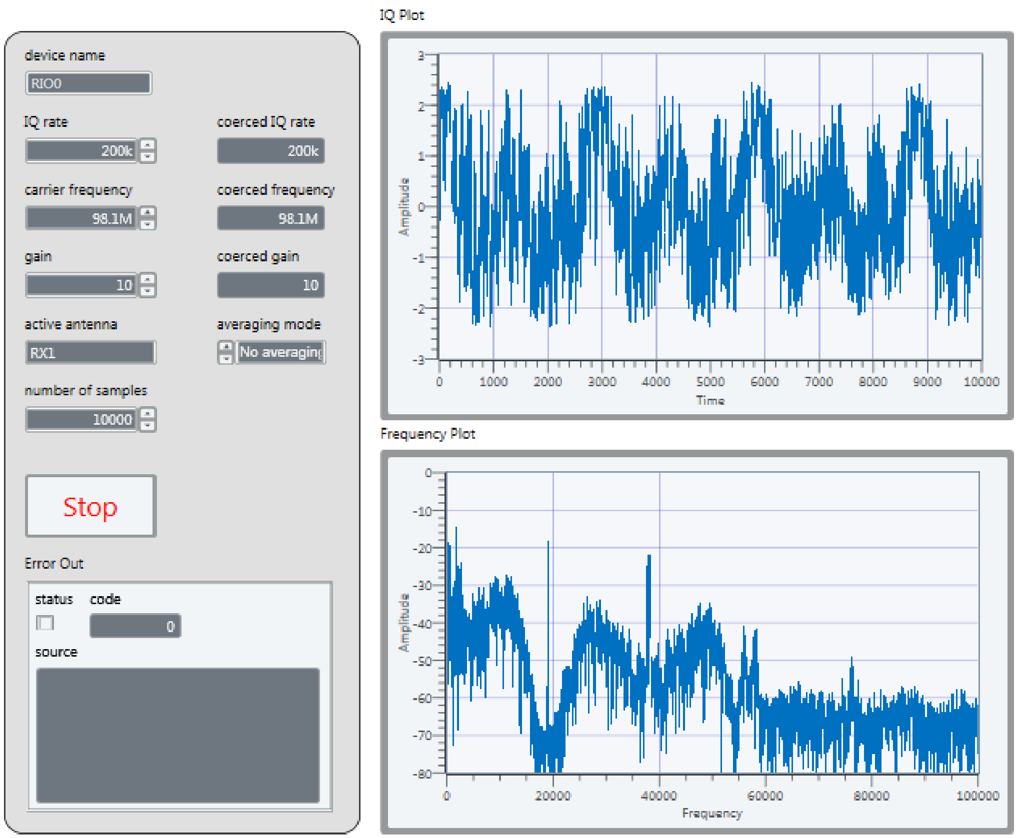
- ***Wire the MathScript Node to match the image***
  Wire the output terminal of USRP Fetch Rx to ***samples.*** Wire ***fmdemod*** to the Simple Resample node, Y, and z.

  Press and hold <Ctrl> while clicking a wire to create a branch off of that wire. By doing this, you wire the raw digitized data the USRP acquired from the air to your FM demodulator, and then to your downstream processing, sound playing, and plotting.

Part B – RUN THE APPLICATION

Listen to the Radio Station.

After you program the VI to demodulate an FM frequency using MathScript, you can listen to a radio station by configuring the settings on the panel.



DETAILED INSTRUCTIONS

- ***Switch to the Panel of the Main MathScript VI***

- ***Configure the controls on the Panel***
  Change the ***Device Name*** to ***RIO0.*** Configure the ***carrier frequency*** for your favorite radio station.

- ***Run the VI***
  LabVIEW generates the audio signal of the FM radio station you specify. Your panel should resemble the panel to the left.

  Note that you can visualize the various spectral components of the radio station as seen in the theoretical spectrum below. If you toggle "averaging mode," you can get a cleaner

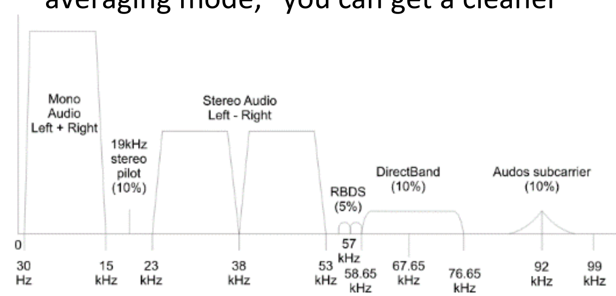


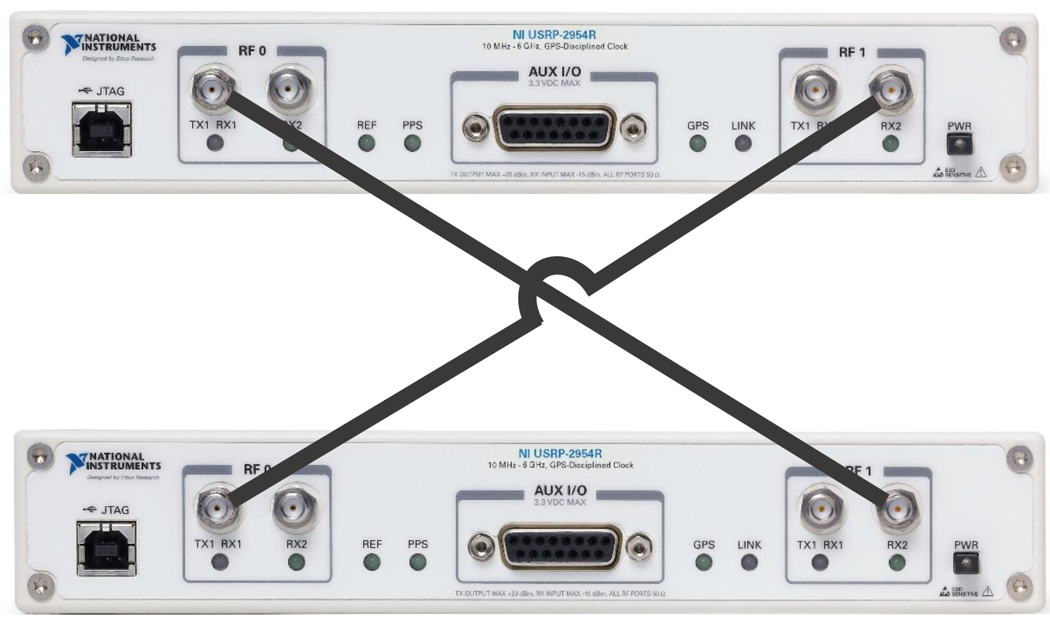
Image from https://en.wikipedia.org/wiki/FM_broadcasting

GROUP EXERCISE 2 – Video Streaming with the LTE Application Framework

Part A— Project Overview

Connect eNB USRP-RIO to UE USRP-RIO

1. Create a bi-directional link between two USRP-RIOs with the station next to you. Ask instructor for help.



DETAILED INSTRUCTIONS

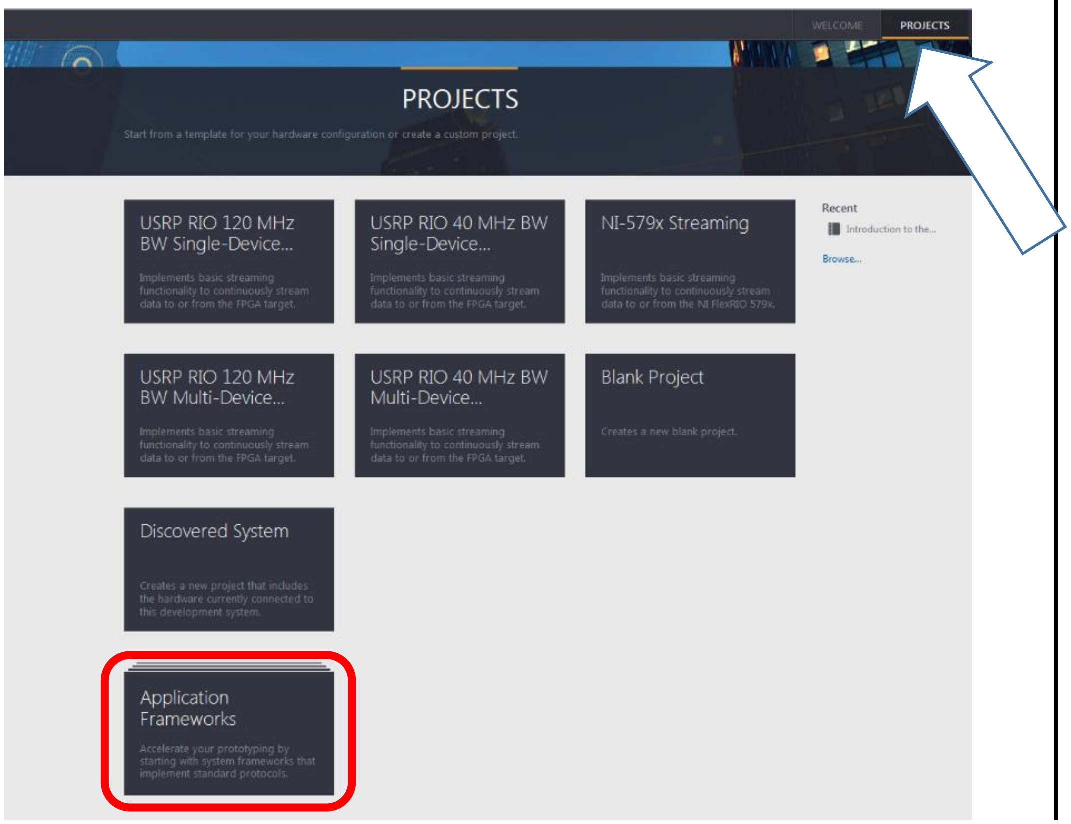
<u>Partner with your neighboring station for this exercise.</u>

- ***Remove the antennas from your USRP-RIOs***
  As antennas may be designed for different frequencies, and to avoid interference, we will use SMA cables for this exercise.
- ***Create a cabled bi-directional link between two USRP-RIOs***
  Connect **RF0/TX1** of one USRP-RIO to **RF1/RX2** of the second USRP-RIO. Connect **RF0/TX1** of the second USRP-RIO to **RF1/RX2** of the first USRP-RIO.

Your setup should resemble the schematic on the left. You now have the hardware set-up for bi-directional communication.

Create a LTE Application Framework Project

1. Open the LTE Design USRP RIO Example.



DETAILED INSTRUCTIONS

In this exercise, use the LTE Application Framework to transmit and receive a video stream between two USRP RIOs.

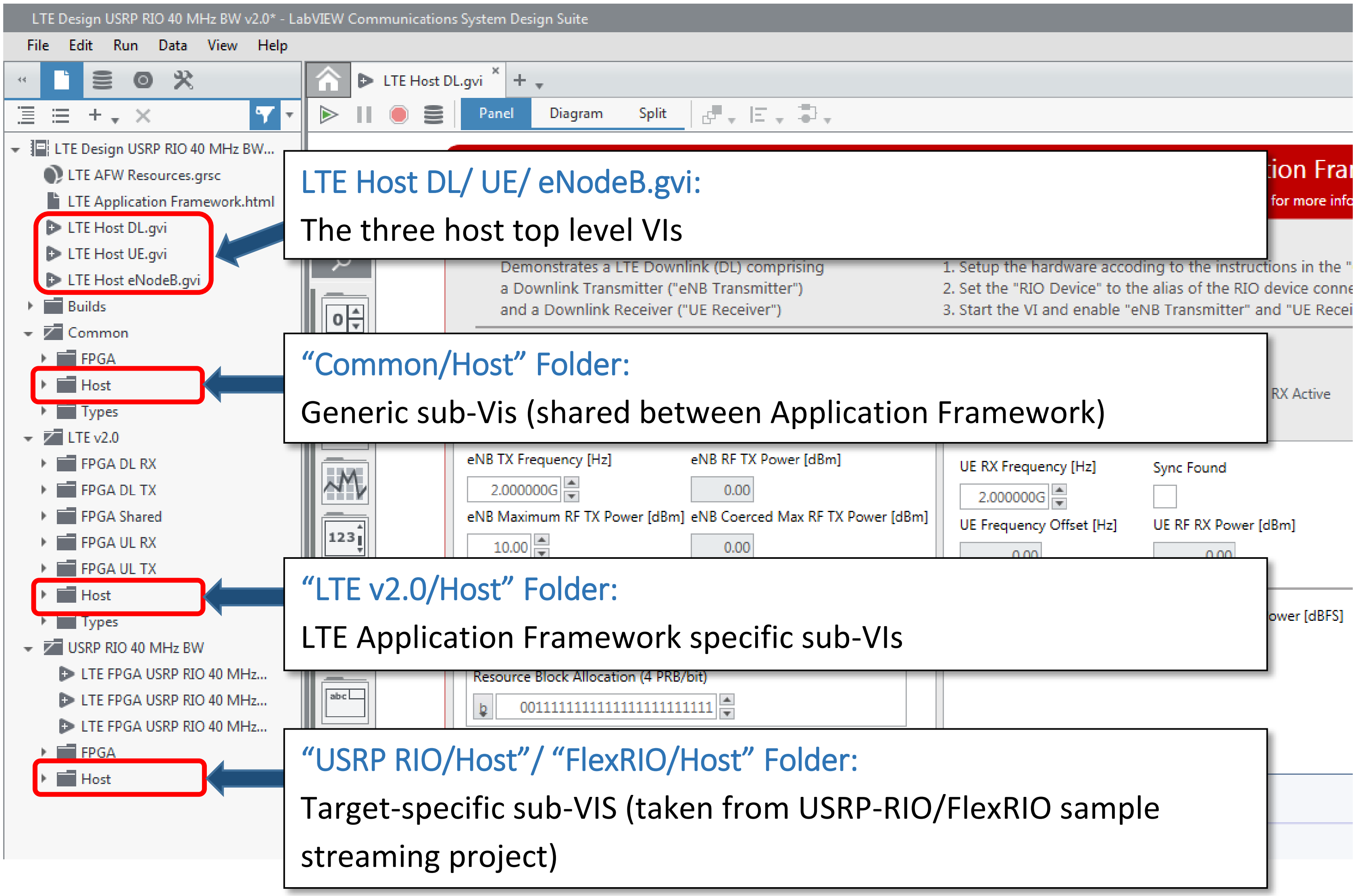
- ***Close Exercise 1***
  Go to **File>>Close>>Close Project**
- ***Open the Example Project***
  Under the ***Projects*** tab, find the project by navigating to ***Application Framework» LTE Design USRP RIO 40 MHz BW v2.0***

- ***Save the Example Project***

After Part A, one person will follow the instructions for Part B (transmitter) while the second person will skip to and follow the instructions for Part C (receiver).

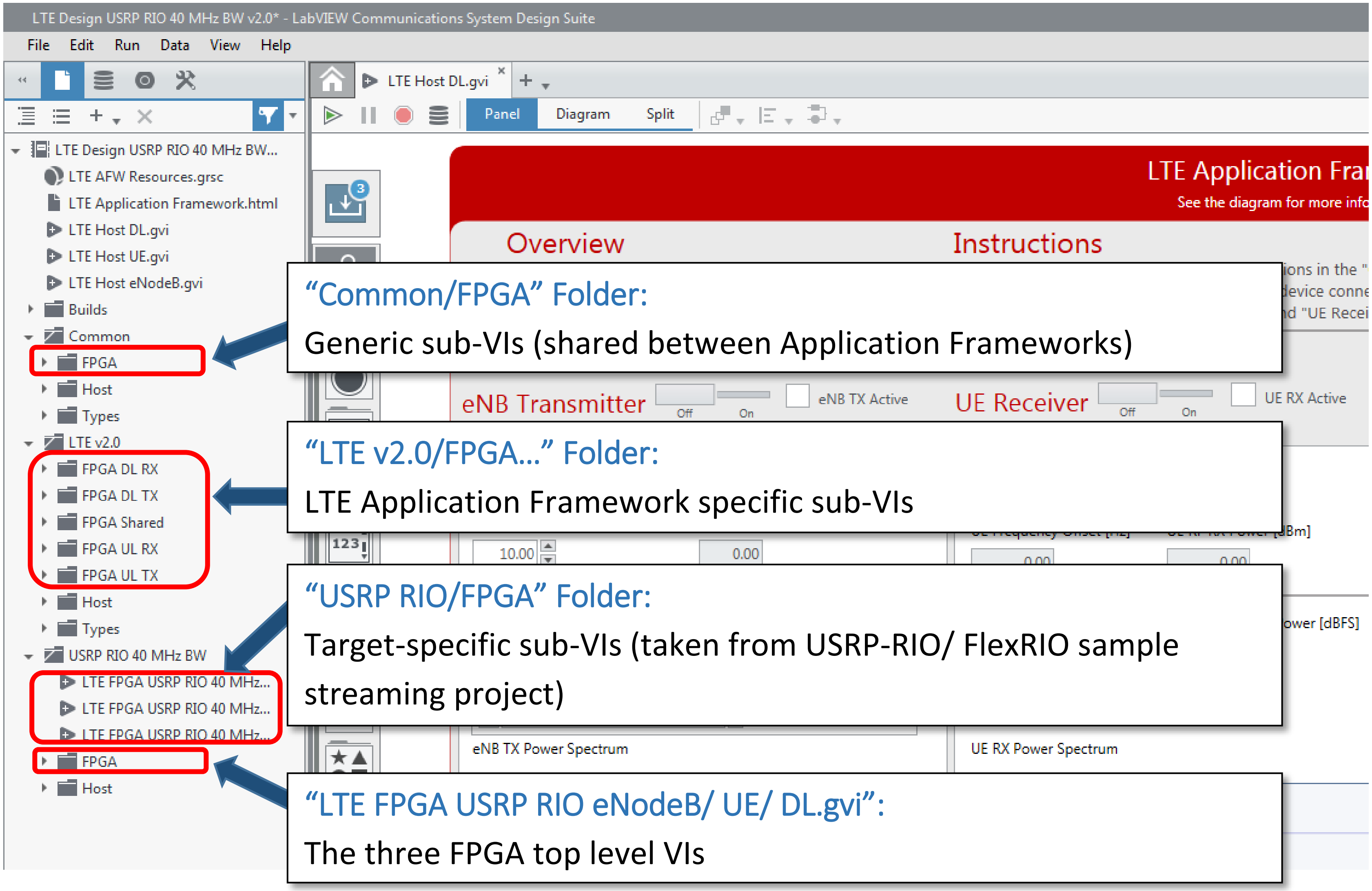
2. Project Overview: Host Code

Explore the Host code in the Project Files tab.



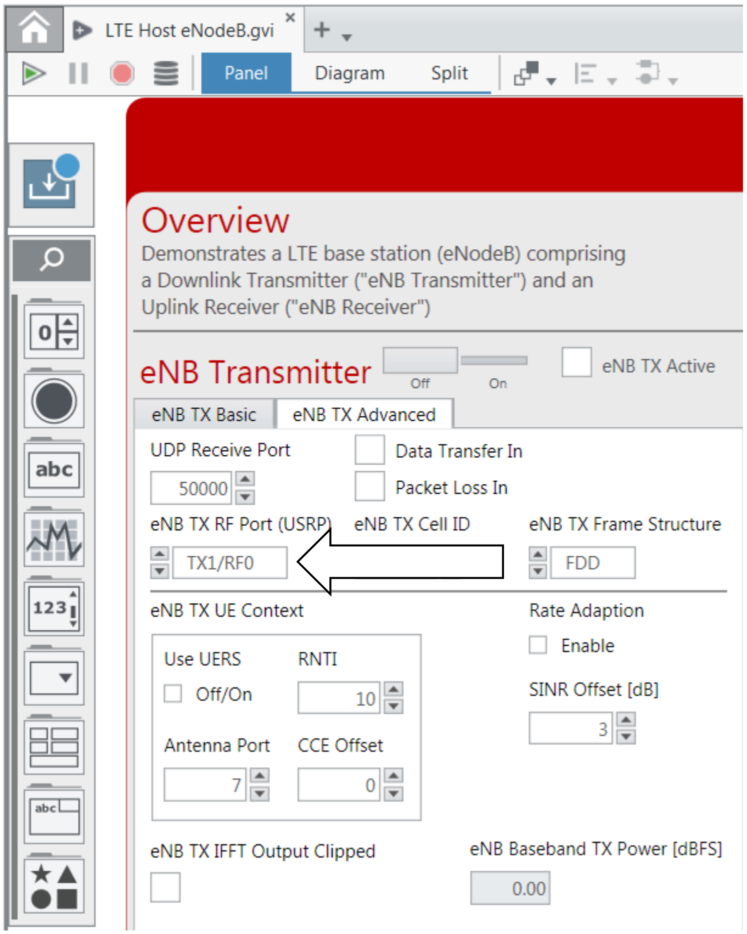
3. Project Overview: FPGA Code

Explore the FPGA code in the Project Files tab.



(Partner 1) Part B— Code Implementation for the Transmitter (Partner 2 jump to Part C)

1. Configure the Transmitter



DETAILED INSTRUCTIONS

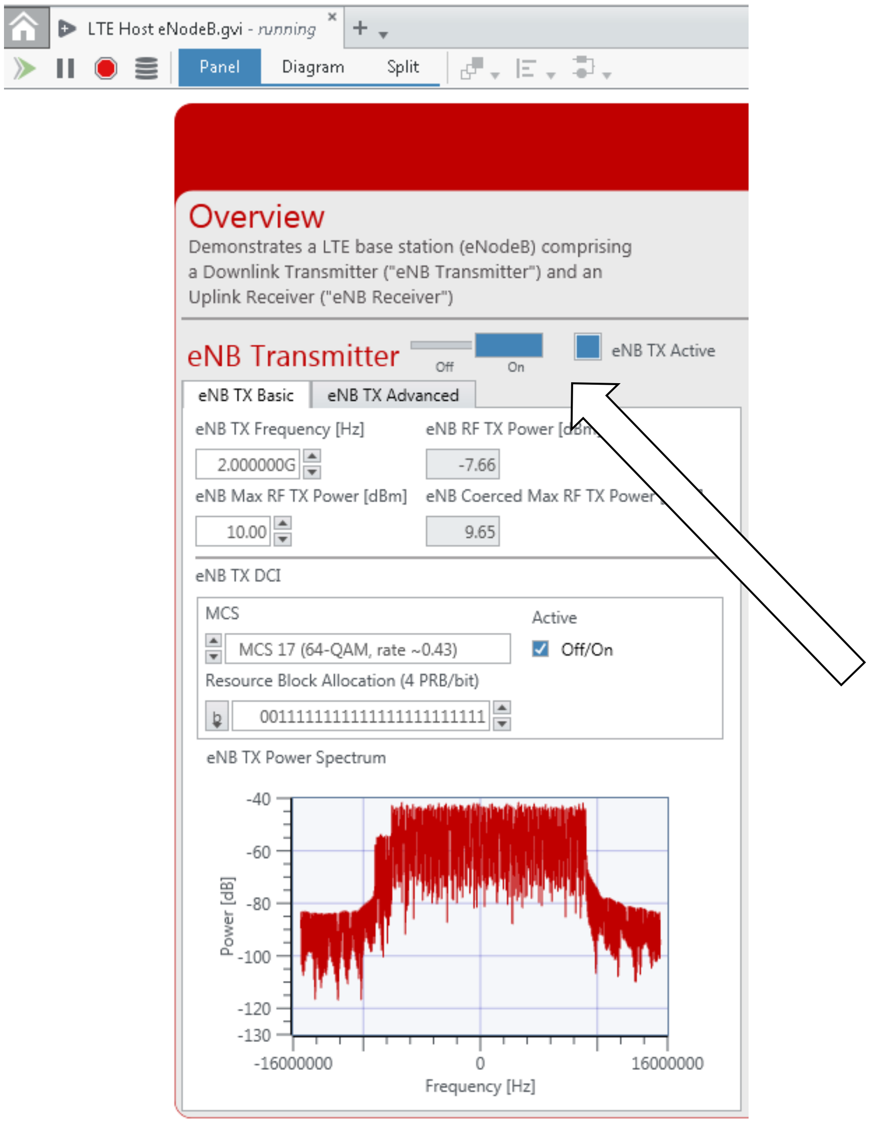
- ***Open the LTE Host eNodeB.gvi***
  The downlink (DL) operation mode can be usedeither a single-device setup or in a double-device setup. The eNodeB/UE operation modes require a double-device setup. Because of this, we are using the base station (eNodeB) for our transmitter.

- ***Select RIO0 for the device name on the top right corner***

- ***Configure the correct TX RF Port***
  Select the ***eNB TX Advanced*** tab and notice the options to configure UDP Receive Port and eNB TX RF Port for the USRP.

  Confirm the correct *eNB TX RF Port* based on the USRP setup (TX1/RF0).

2. Turn on the eNB Transmitter and observe the results



DETAILED INSTRUCTIONS

- ***Configure the Transmitter Signal***
  Select the ***eNB TX Basic*** tab and notice the options to configure eNB TX Frequency, eNB Max RF TX Power, MSC, and Resource Block Allocation. The value you will set in ***enB TX Frequency*** on the front panel will be the same as the value your partner sets in ***UE RX Frequency*** on the receiver side.

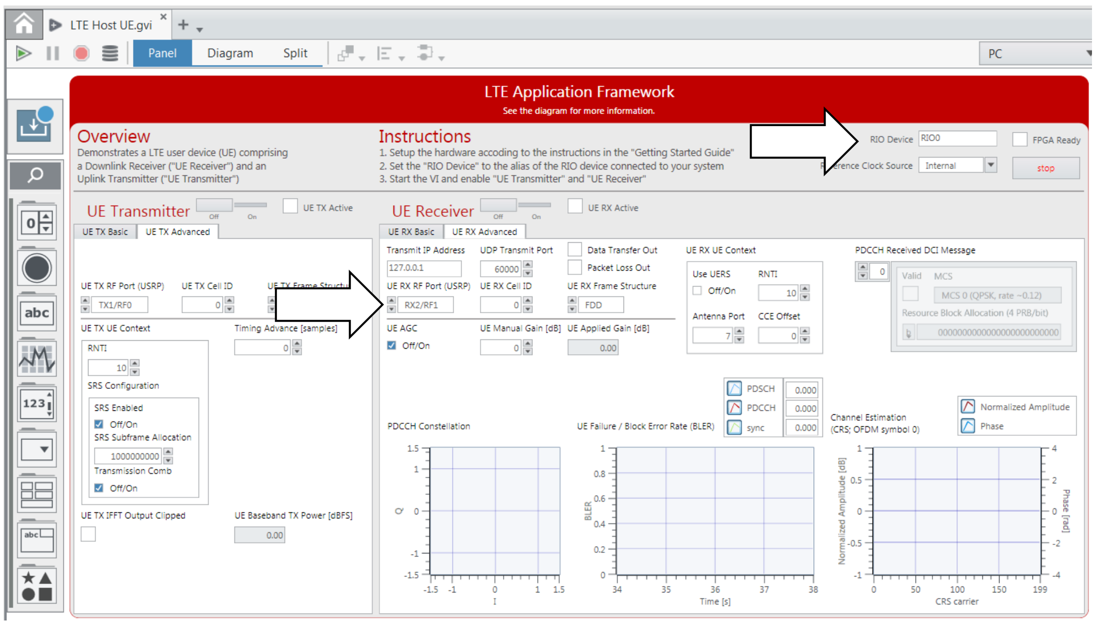
- ***Run the VI***
  You might receive a compile message the first time you run the VI. If so, just wait for the host code to compile. When the ***FPGA Ready*** indicator at the top right corner of the panel turns blue, you are ready to transmit.

- ***Turn on the eNB Transmitter***
  Slide the switch above the TX tabs to ***On*** and notice how the ***eNB TX Active*** turns blue.

- ***Observe the eNB TX Power Spectrum***
  Change your signal configurations like MSC or Resource Block Allocation and observe how it changes the power spectrum.

(Partner 2) Part C— Code Implementation for the Receiver (Partner 1 does Part B, above)

1. Configure the Receiver



DETAILED INSTRUCTIONS

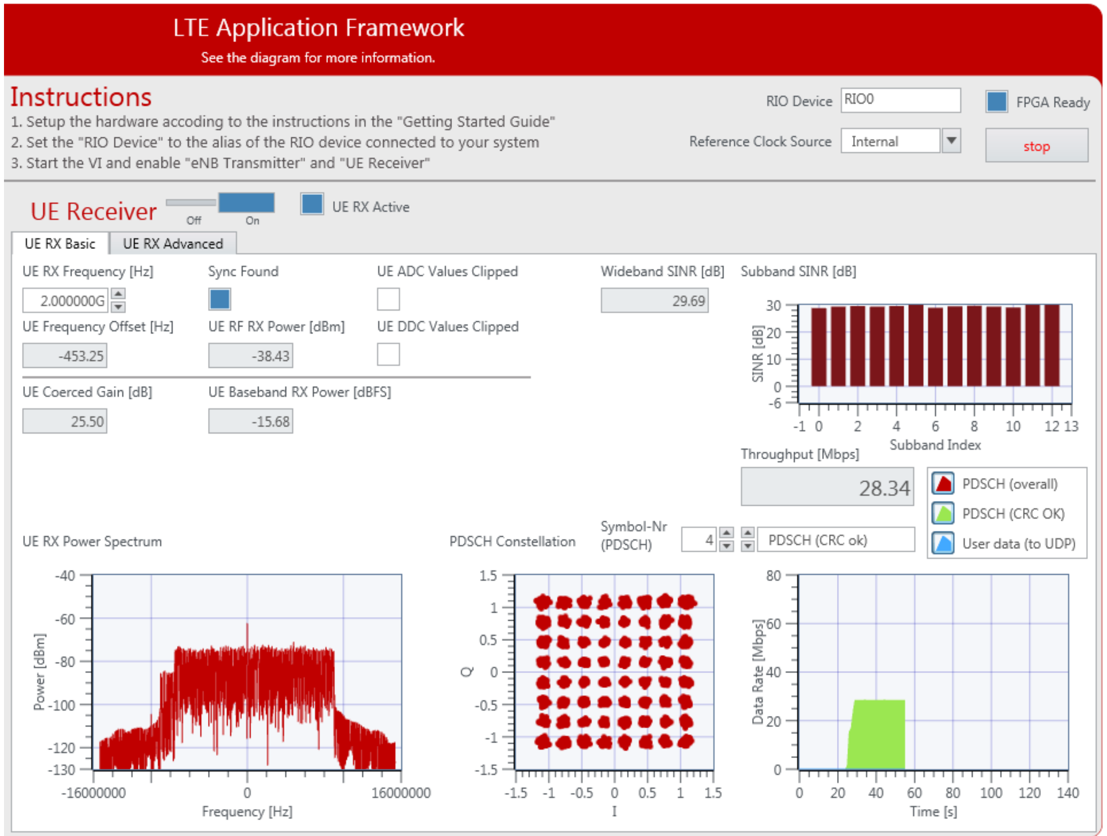
- ***Open the LTE Host UE.gvi***
  The downlink (DL) operation mode can be used either a single-device setup or in a double-device setup. The eNodeB/UE operation modes require a double-device setup, so we are using the user equipment (UE) for our receiver.

- ***Select the RIO0 for the device name on the top right corner***

- ***Configure the correct TX RF Port***
  Select the ***UE RX Advanced*** tab and notice the options to configure UDP Transmit Port and UE RX RF Port for the USRP.

  Confirm the correct *eNB TX RF Port* based on the USRP setup (RX2/RF1).

2. Turn on the Receiver and observe results



DETAILED INSTRUCTIONS

- ***Configure the Transmitter Signal***
  Select the ***UE RX Basic*** tab and configure the UE RX Frequency to match your partner's TX Frequency.

- ***Run the VI***
  You might receive a compile message the first time you run the VI.

  When the ***FPGA Ready*** indicator at the top right corner of the panel turns blue, you are ready to transmit.

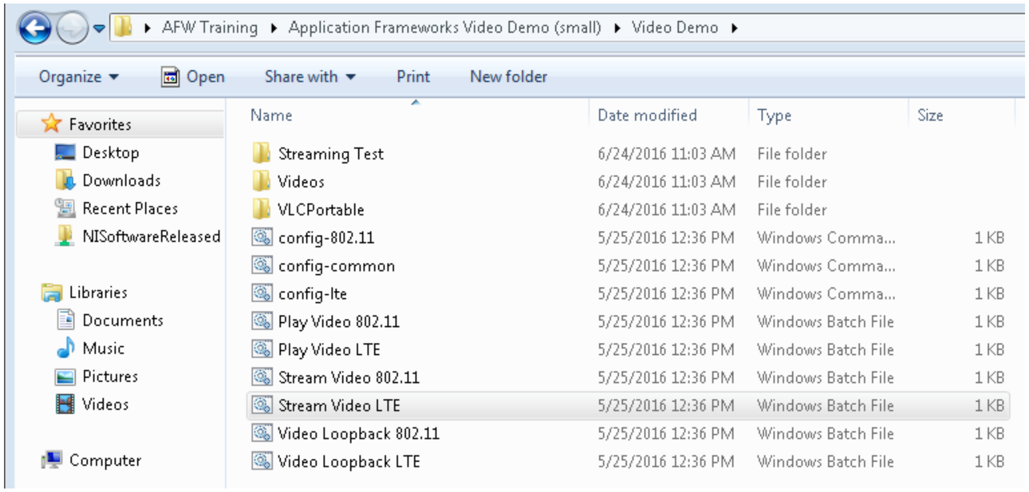
- ***Turn on the UE Receiver***
  Slide the switch above the RX tabs to On and notice how the ***UE RX Active*** turns blue.

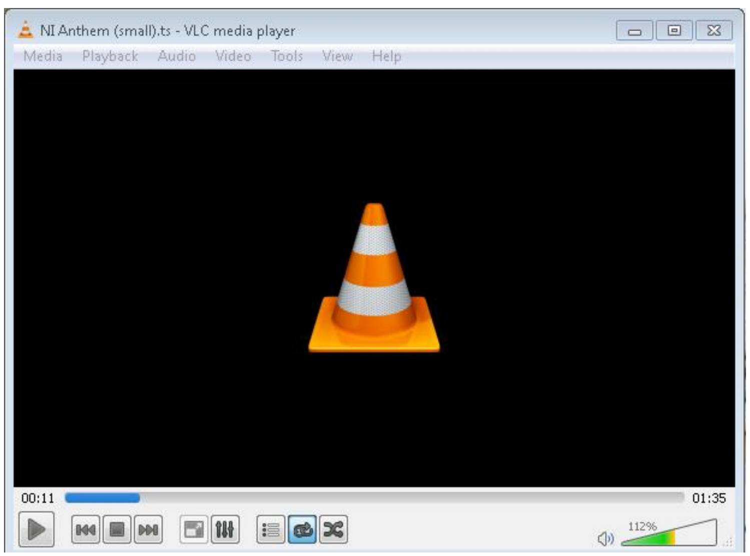
- ***Observe the Results***
  The UE RX Power Signal should match the transmitted signal.

  The PDSCH Constellation should match the QAM from the transmitter.

(Partner 1) Part D— Video Streaming (with User Data) (Partner 2 jump to Step 2)

1. Instructions for the transmitter (Receiver skip to step 2)





DETAILED INSTRUCTIONS

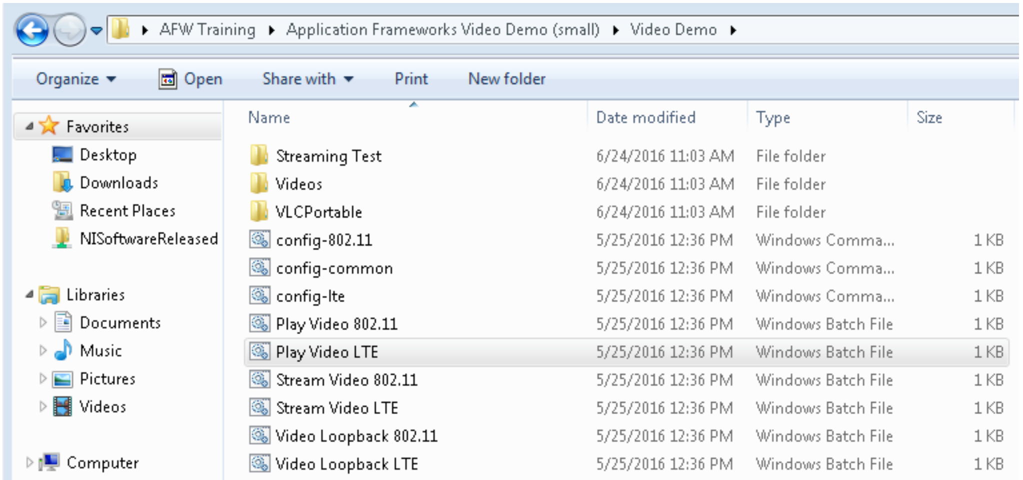
- ***Run the Stream Video LTE batch file***
  Open the Video Demo folder on your desktop and double-click on the ***Stream Video LTE*** batch file.

- ***A VLC screen should pop up***
  The NI Anthem (small).ts will start playing with no video.

  Observe the video streaming on the receiver.

  Notice how the video will stop playing on the receiver if you slide either the eNB transmitter or the UE receiver to the off position.

2. <u>(Partner 2)</u> Instructions for the Receiver <u>(Partner 1 complete Step 1, above)</u>



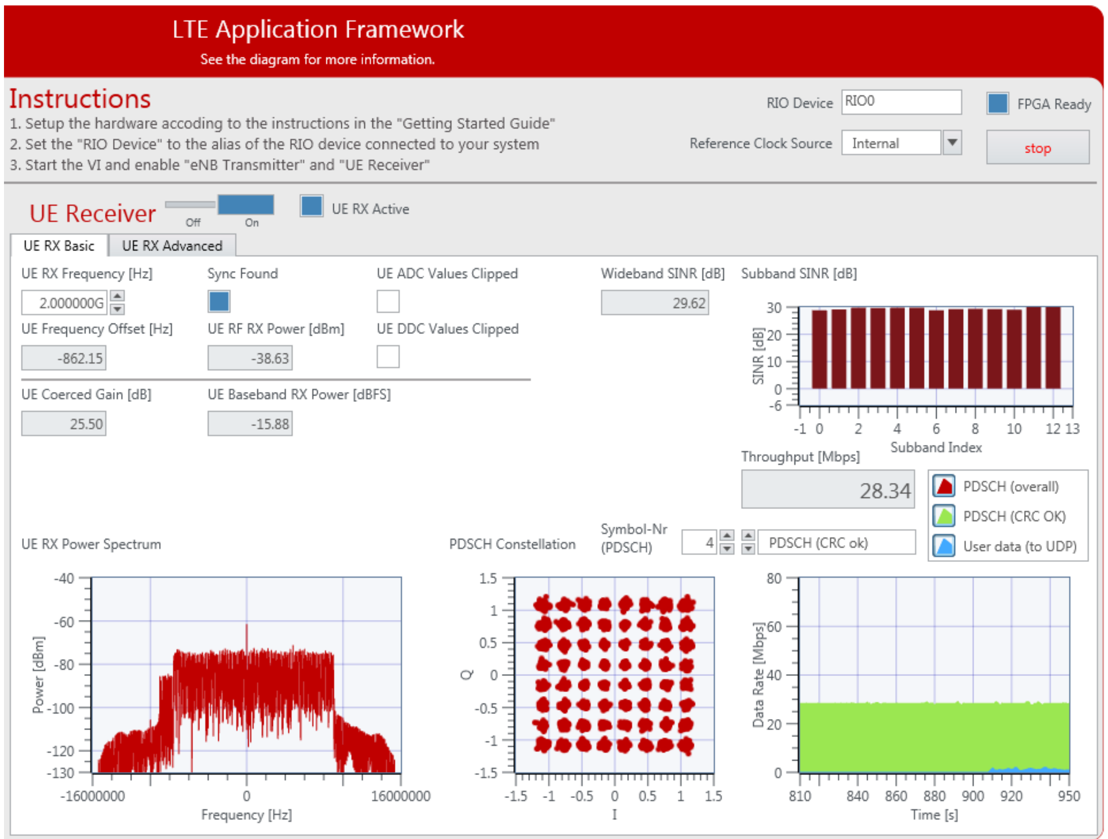
DETAILED INSTRUCTIONS

- ***Run the Play Video LTE batch file***
  Open the Video Demo folder on your desktop and double-click on the ***Play Video LTE*** batch file.

- ***A VLC screen should pop up***
  The NI Anthem video will start playing if both the eNB transmitter and UE receiver are turned on and the transmitter is streaming the video.

  Notice how the video will stop playing if you slide either the eNB transmitter or the UE receiver to the off position.

3. Observe the receiver data



DETAILED INSTRUCTIONS

- ***Notice the User data in the bottom right graph***
  When the video is streaming, notice the blue user data that shows up on the graph.

  Note that as you change the MCS and Resource blocks on the transmitter, the throughput changes.

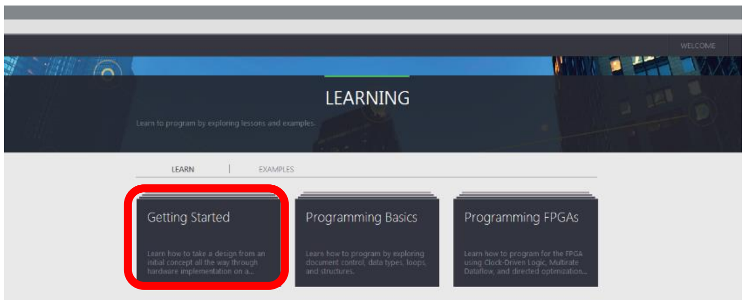
EXERCISE 3 – Algorithm Design and Testing

Part A— Build an OFDM Modulator with a Multirate Diagram

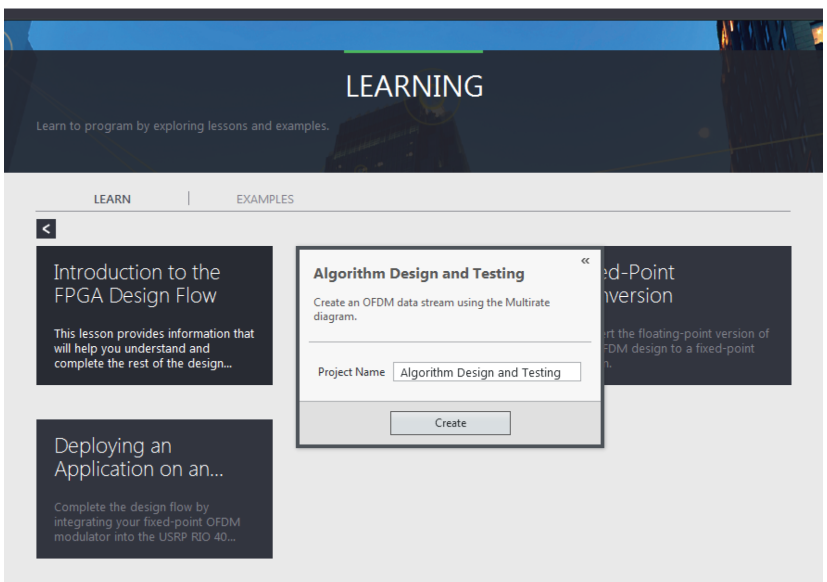
Explore the LabVIEW Communications Project and VI

1. Open example Algorithm Design and Testing.

A



B



DETAILED INSTRUCTIONS

In this exercise, use ***Algorithm Design and Testing*** from the ***Getting Started*** lessons to create an algorithm that implements orthogonal frequency-division multiplexing (OFDM).

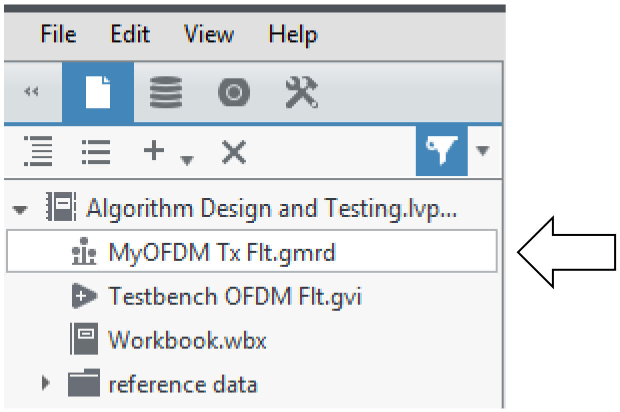
- ***Close Exercise 2***
  Go to **File>>Close>>Close Project**

- ***Open the Lesson Project***
  Under the ***Learn*** tab, find the project by navigating to ***Getting Started» Overview of the FPGA Design Flow» Algorithm Design and Testing,*** as shown in ***Figure A.***

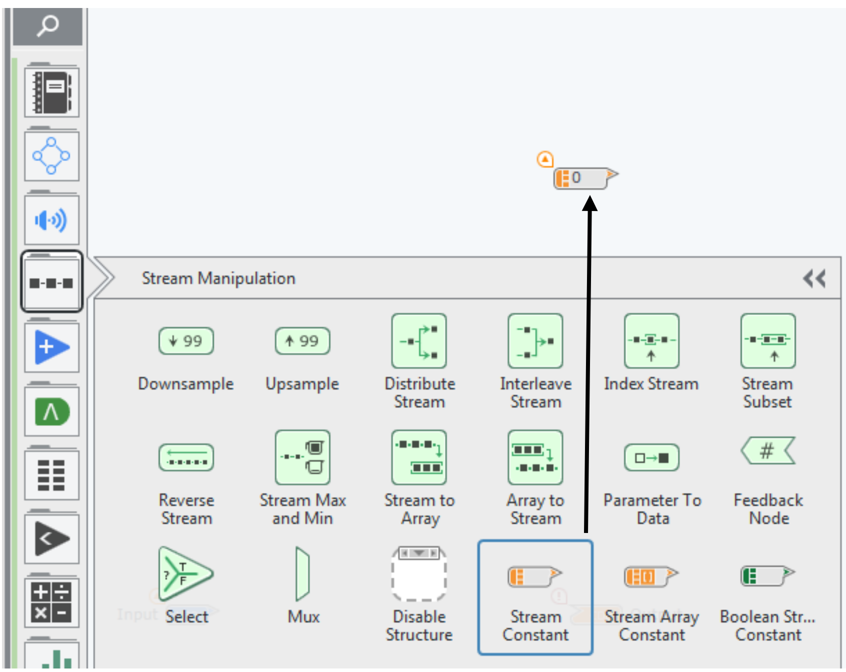
- ***Save the Example Project***
  Select ***Algorithm Design and Testing***, enter a project name and press the ***Create*** button, as shown in ***Figure B***.

2. Explore the Multirate Diagram

A



B



DETAILED INSTRUCTIONS

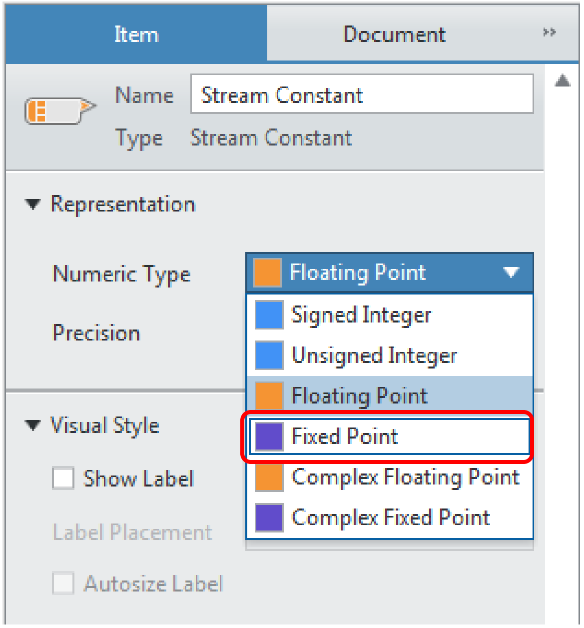
- ***Close the Workbook tab and follow this manual***
  Click the X on the Workbook tab. On your own time, you can follow these tutorials for guided learning, but today we will use this manual.

- ***Open the Multirate Diagram***
  Double-click ***MyOFDM Tx Flt.gmrd,*** as shown in ***Figure A.***

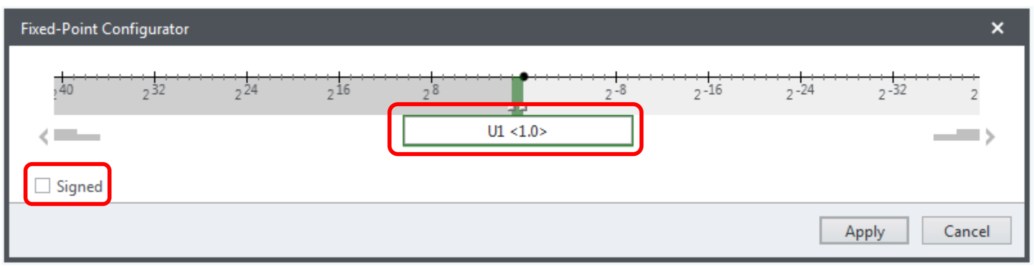
- ***Place a Stream Constant on the Diagram***
  Open the Stream Manipulation palette and place a ***Stream Constant*** on the diagram as shown in ***Figure B***.

3. Configure the Stream Constant

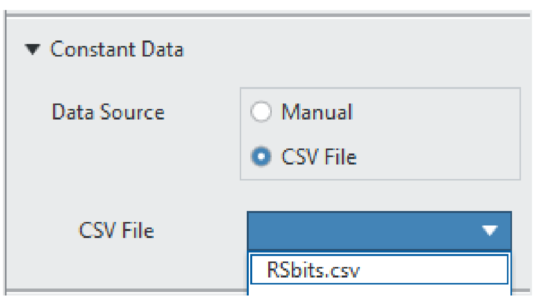
A



B



C



DETAILED INSTRUCTIONS

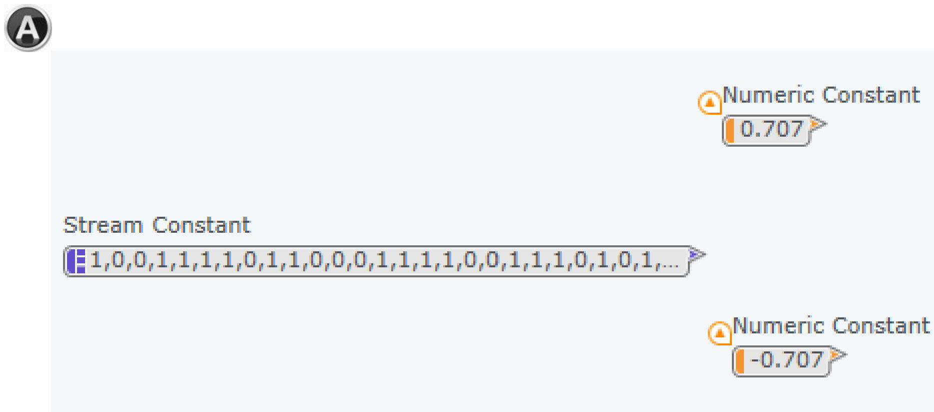
- ***Configure the Stream Constant representation***
  Either right-click on the Stream Constant and select ***Properties*** or select the Stream Constant and go to the ***Item*** tab to the right. Press the drop-down arrow next the ***Numeric Type*** and select ***Fixed Point,*** as shown in ***Figure A***.

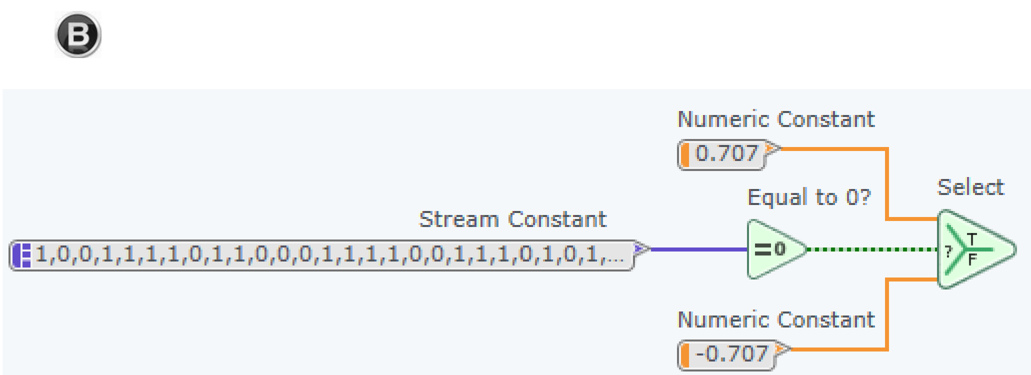
- ***Configure the Fixed-Point Stream Constant***
  Uncheck the ***Signed*** box on the bottom left of the ***Fixed-Point Configurator dialog box***.

  Change the precision by clicking in the green box and typing ***1.0***. Press ***Apply*** to save the changes, shown in ***Figure B***.

- ***Configure the Constant Data***
  On the ***Item*** tab to the right, select ***CSV File*** for the ***Data Source*** and then select ***RSbits.csv*** from the ***CSV File*** drop-down, as shown in ***Figure C***.

4. Map a binary stream of interleaved IQ Data into 4-QAM





DETAILED INSTRUCTIONS

- ***Place a Numeric Constant on the diagram***
  Change the value of the ***Numeric Constant*** to ***0.707***.

- ***Create a copy of the Numeric Constant***
  Hold <Ctrl> and click and drag the ***Numeric Constant***. Change the value to ***-.0.707,*** as shown in ***Figure A***.

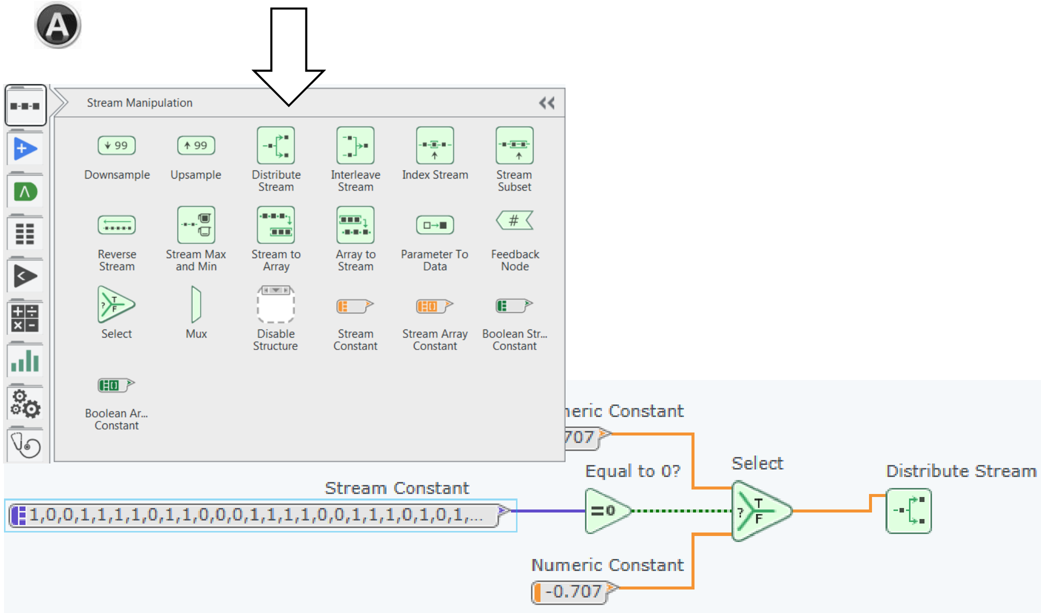
- ***Place an Equal to 0 node and Select node onto the block diagram***
  Place and wire an ***Equal to 0*** function to the right of the Stream Constant.

  Place a ***Select*** function next to the ***Equal to 0*** function.

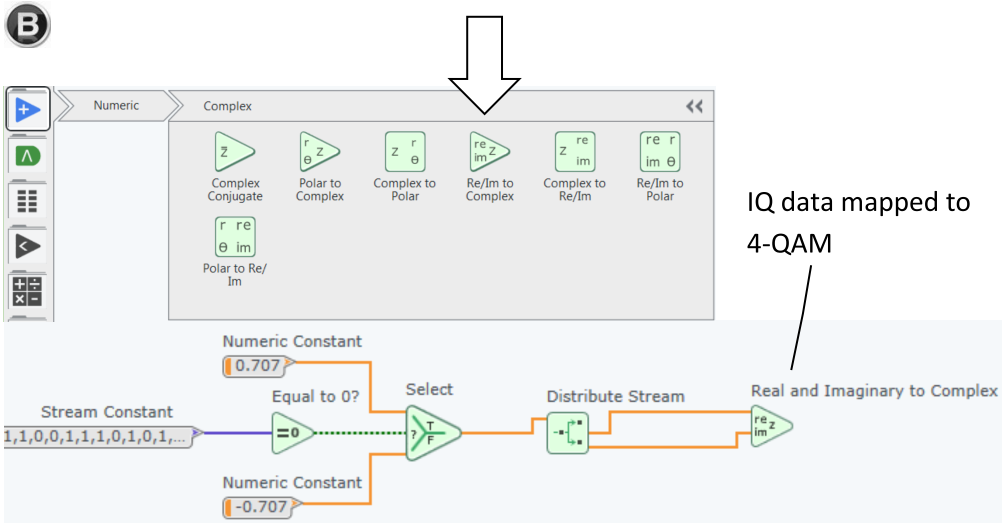
  Wire the output of the ***Equal to 0*** to the **x** input of the ***Select*** function.

  Wire the ***0.707*** Numeric Constant to the **True** input of the ***Select*** function. Wire the ***-0.707*** Numeric Constant to the **False** input of the ***Select*** function to match ***Figure B***.

A



B



DETAILED INSTRUCTIONS

- ***Place a Distribute Stream onto the diagram***
  Place a ***Distribute Stream*** function from the ***Stream Manipulation*** palette on the diagram and wire it to the ***Select*** function, as shown in ***Figure A***.

  Distribute Stream waits for an I and a Q, then splits into two different streams of data.

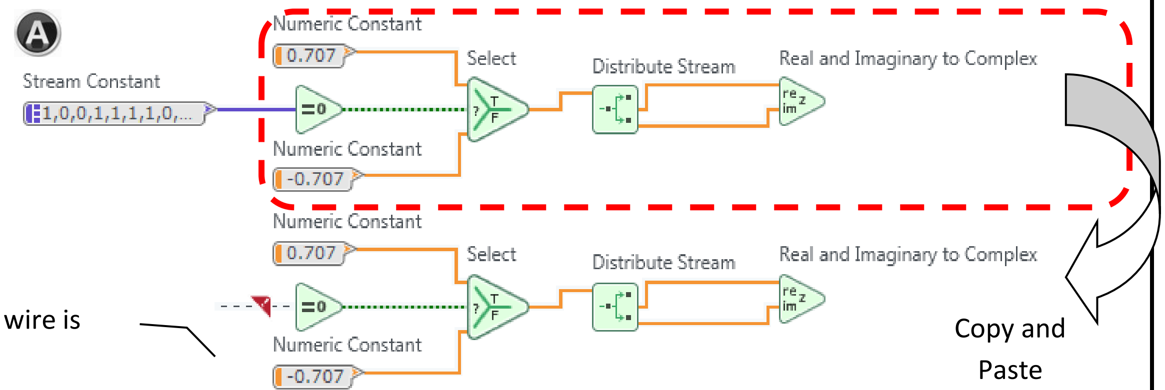
- ***Place a Real and Imaginary to Complex function on the diagram***
  Add a ***Real and Imaginary to Complex*** function from the ***Numeric>>Complex*** palette next to the ***Distribute Stream*** function.

  Wire the top output from ***Distribute Stream*** to the top input of ***Real and Imaginary to Complex***.

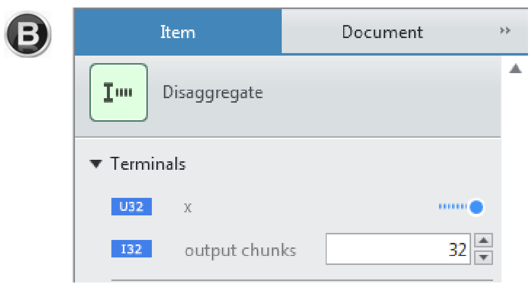
  Wire the bottom output from ***Distribute Stream*** to the bottom input of ***Real and Imaginary to Complex,*** as shown in ***Figure B***.

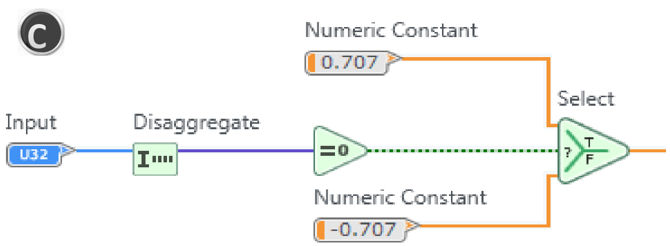
✓ ***Real and Imaginary to Complex*** rebuilds I and Q into a complex number mapped to 4 QAM.

Copy and paste the your code to create the following diagram:



A broken wire is
normal





DETAILED INSTRUCTIONS

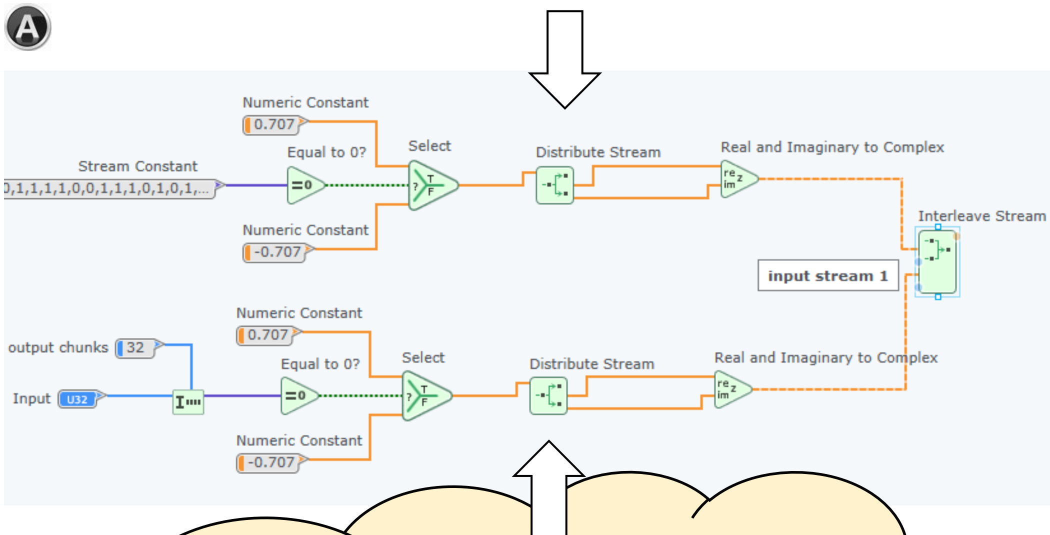
- ***Replicate your signal processing chain***
  Copy and paste the code in the red box to create the following diagram. You can select all the code and either hit <Ctrl-C> or hold <Ctrl> and drag the selected code. Your diagram should match ***Figure A.***

- **Break IQ data in the form of a 32-bit integer into individual bits**
  Next to the broken wire, place a Disaggregate function.

  While the Disaggregate function is selected, change ***output chunks*** to 32 in the ***Item*** pane, as shown in ***Figure B***

  Move and wire the existing Input U32 terminal to the ***Disaggregate*** function.

  Wire the ***Disaggregate*** function to the broken wire, as shown in ***Figure C***. A 32-bit input value will now be broken into a stream of individual bits representing I, Q, I, Q, and so on.



Did you notice the rate conversion?

Note that **Distribute Stream** waits for two tokens before proceeding. The Multirate diagram handles this rate change for you.

DETAILED INSTRUCTIONS

- ***Place an Interleave Stream function on the diagram***

  Place an ***Interleave Stream*** function in between the two ***Real and Imaginary to Complex*** functions.

  Wire the top output of ***Real and Imaginary to Complex*** to **input stream 0** of ***Interleave Stream***.

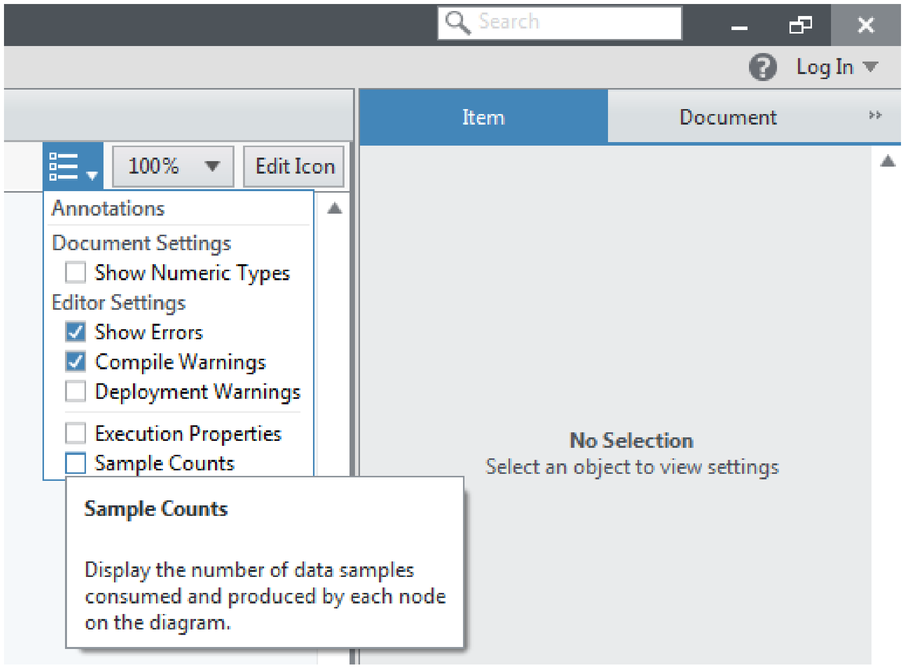
  Wire the bottom output of ***Real and Imaginary and Complex*** to **input stream 1** of ***Interleave Stream*** to match ***Figure A***.

  The top input produces the pilot or reference symbol, represented as a complex floating-point number, and the bottom input produces the data symbols as complex floating-point numbers.

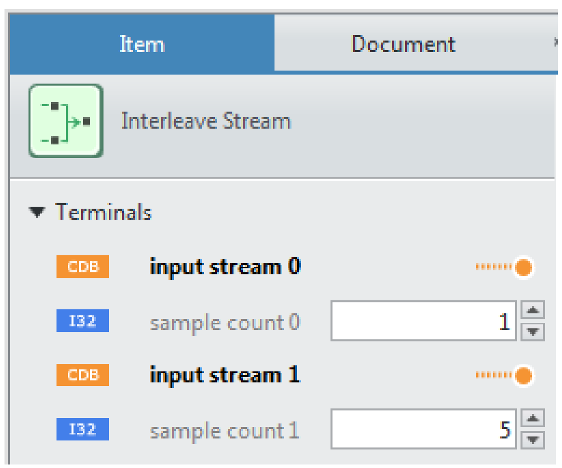
  How might you add to this algorithm? How might you complete the OFDM transmitter?

5. Display and Configure Sample Counts

A



B



DETAILED INSTRUCTIONS

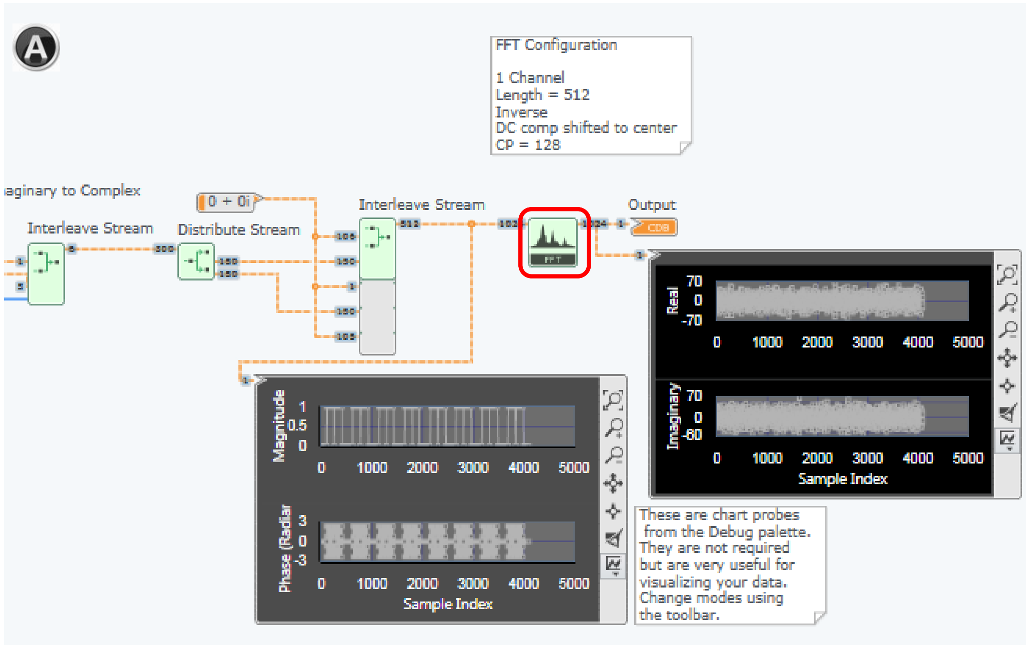
To help you visualize Multirate Dataflow, LabVIEW Communications provides the ability to display the sample counts of each node terminal on a Multirate diagram. The sample count for each input terminal represents the number of data samples that input requires before the node can execute. The sample count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

- ***Display Sample Counts***
  Click the ***Annotations*** drop-down on the toolbar and select ***Sample Counts***, as shown in ***Figure A***.

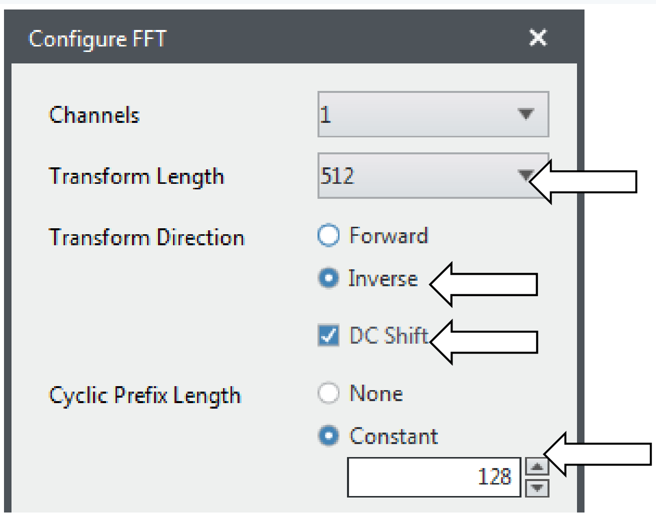
- ***Configure sample counts for Interleave Stream***
  Click the ***Interleave Stream*** function. On the ***Item*** tab to the right, change ***sample count 1*** to ***5,*** as shown in ***Figure B***.

We do this because in the OFDM specifications, 1 pilot symbol should be sent for every 5 data symbols.

Part B – Open the Solution file



B



DETAILED INSTRUCTIONS

For time considerations, we will skip ahead to the solution OFDM transmitter implemented in the Multirate diagram and adjust a configuration.

- ***Open the solution file***
  From the Files pane, open the solution folder and select ***OFDM Tx Flt solution.gmrd***.

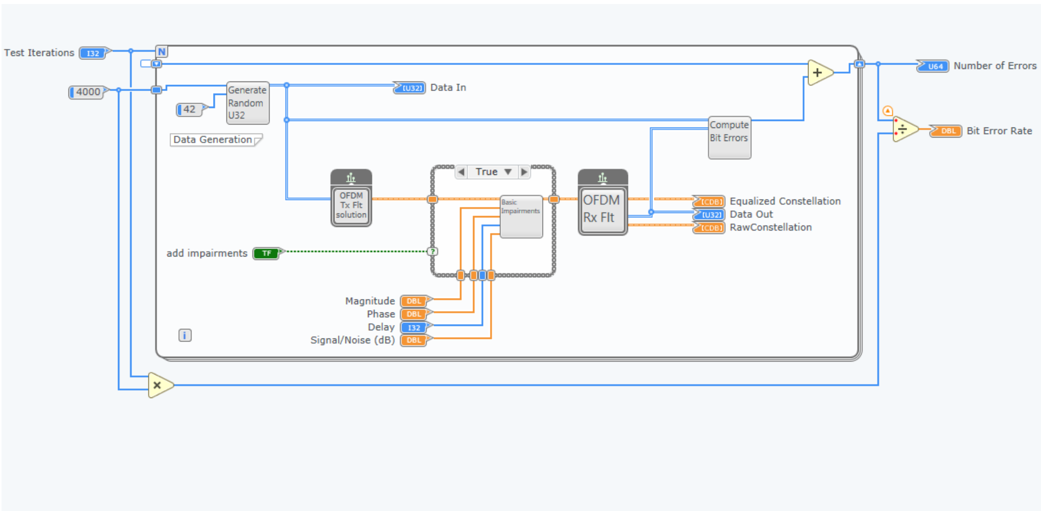
  Figure A shows the diagram of the completed Multirate diagram

- ***Configure FFT***
  Select the FFT function, go to the Item tab, and choose ***Configure***. Adjust the configuration to match the values shown in ***Figure B***. The FFT should have a ***Transform Length*** of ***512***, be ***Inverse***, select ***DC Shift***, and include a 128-element Cyclic Prefix Length. **Please look at Figure B closely!**

- ***Run the solution to verify simulated design***
  Notice that prior to the IFFT we can visualize 8 OFDM symbols (which corresponds to the amount of data we fed the Multirate diagram).

Part C – Test the OFDM Algorithm

Now that we have verified our OFDM implementation produces data as expected, we want to run it through a testbench that exercises the transmit and receive algorithm.



DETAILED INSTRUCTIONS

- ***Open the Testbench OFDM Flt.gvi and view the diagram***
  The testbench modulates and then demodulates a stream of random bits using your OFDM algorithm.

- ***Run the testbench with the default solution file***
  This VI uses the solution file, OFDM Tx Flt solution.gmrd.

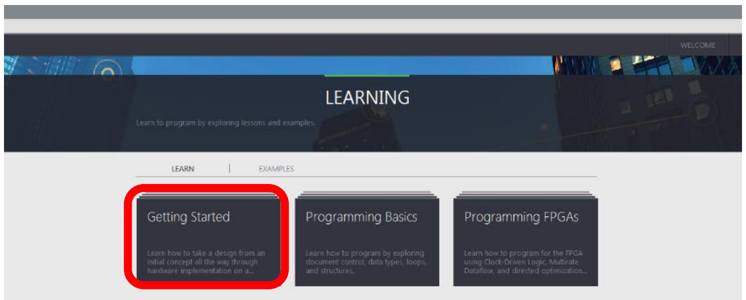
- ***Move to the panel and run the testbench again***
  The test should report zero errors with the Add Impairments checkbox unchecked.

EXERCISE 4 – Fixed Point Conversion

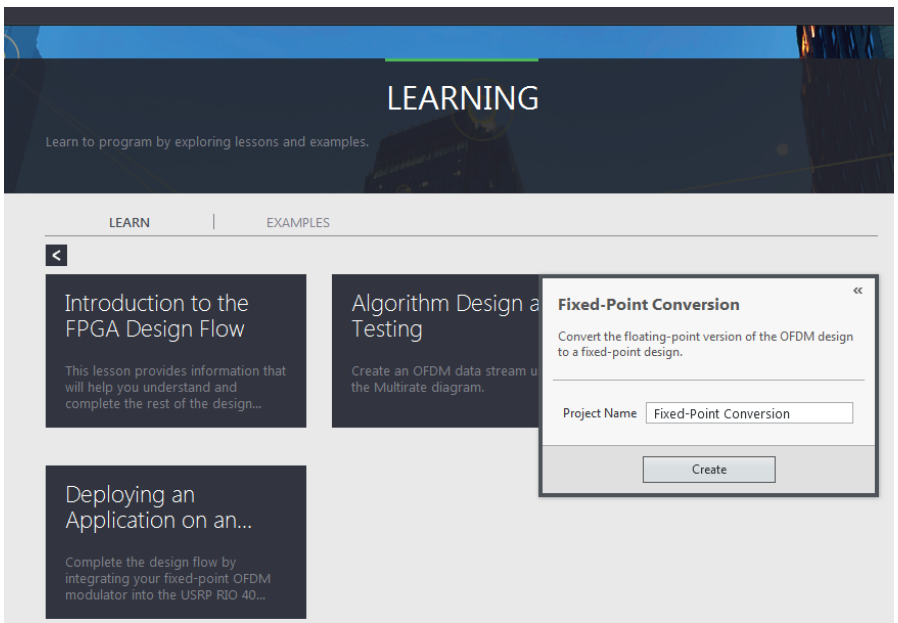
Part A— Convert a floating-point algorithm to a fixed-point design

1. Open example Algorithm Design and Testing.

A



B



DETAILED INSTRUCTIONS

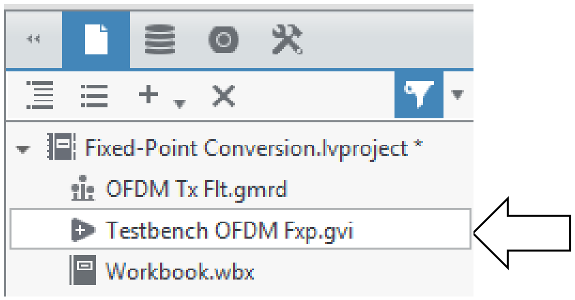
In this exercise, use the ***Fixed-Point Conversion*** from the ***Getting Started*** examples to convert a floating-point algorithm to a fixed-point design for implementation on an FPGA. This conversion is done because deploying a floating-point algorithm to an FPGA can be costly in terms of FPGA resources and power consumption.

- ***Close Exercise 3***
  Go to **File>>Close>>Close Project**
- ***Open the Example Project***
  Under the ***Learn*** tab, find the project by navigating to ***Getting Started» Overview of the FPGA Design Flow» Fixed-Point Conversion*** as shown in ***Figure A.***

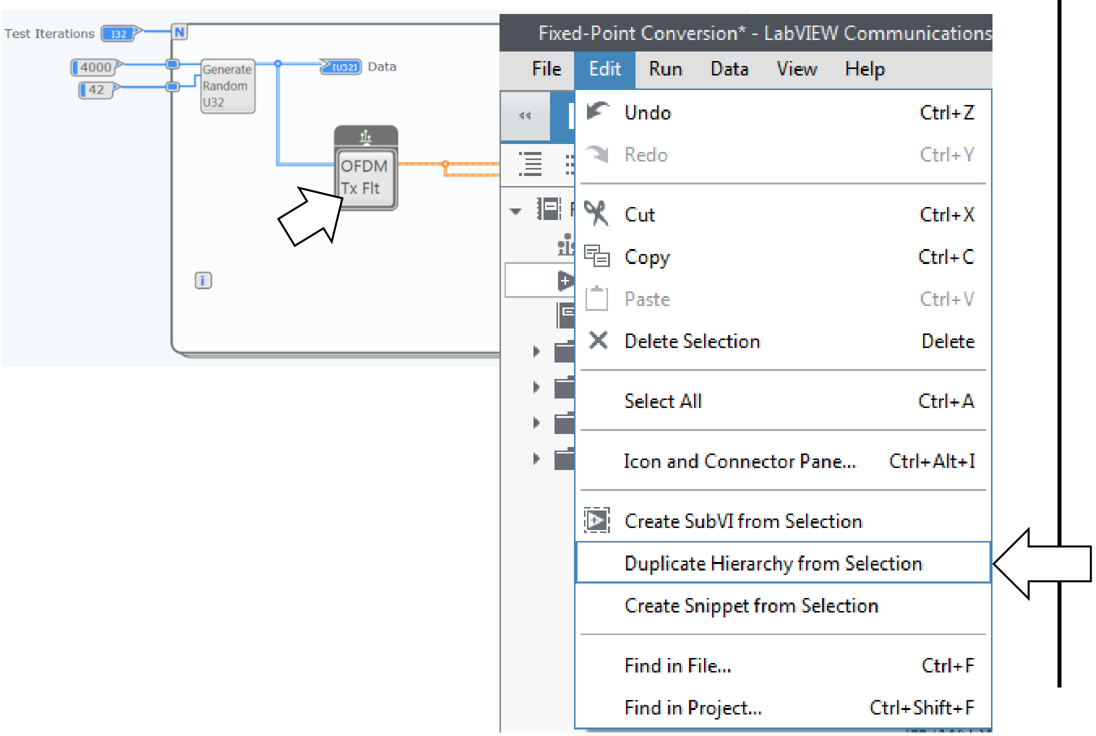
- ***Save the Example Project***
  Select ***Fixed-Point Conversion***, enter a Project Name and press the ***Create*** button as shown in ***Figure B***.

2. Duplicate the Hierarchy

A



B



DETAILED INSTRUCTIONS

Before you begin converting data types, you need to duplicate the parts of the code that you plan to convert. The duplicated code is used to compare the output of the converted fixed-point design to the output of the original floating-point design.

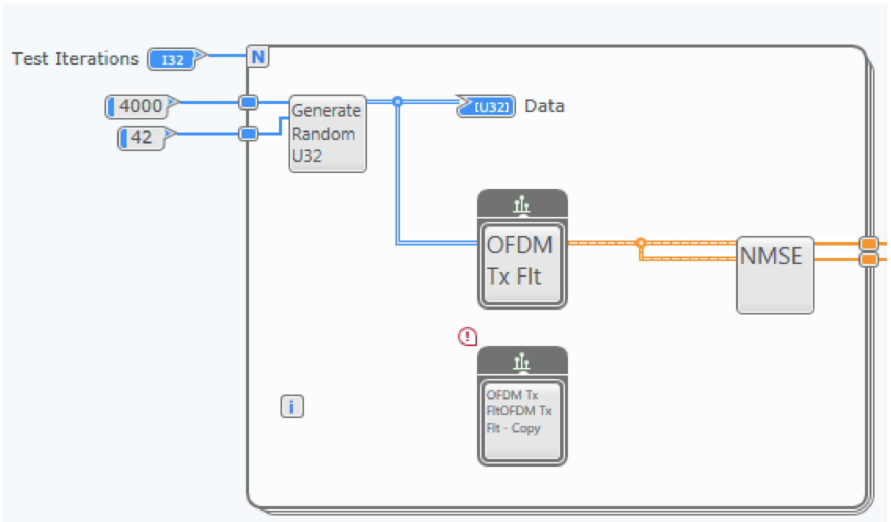
- ***Close the Workbook tab and follow this manual***
  Click the X on the Workbook tab. On your own time, you can follow these tutorials for guided learning, but today we will use this manual.

- ***Open the Testbench OFDM Fxp.gvi***
  Double Click on ***Testbench OFDM Fxp.gvi*** as shown in ***Figure A*** and switch to the diagram.

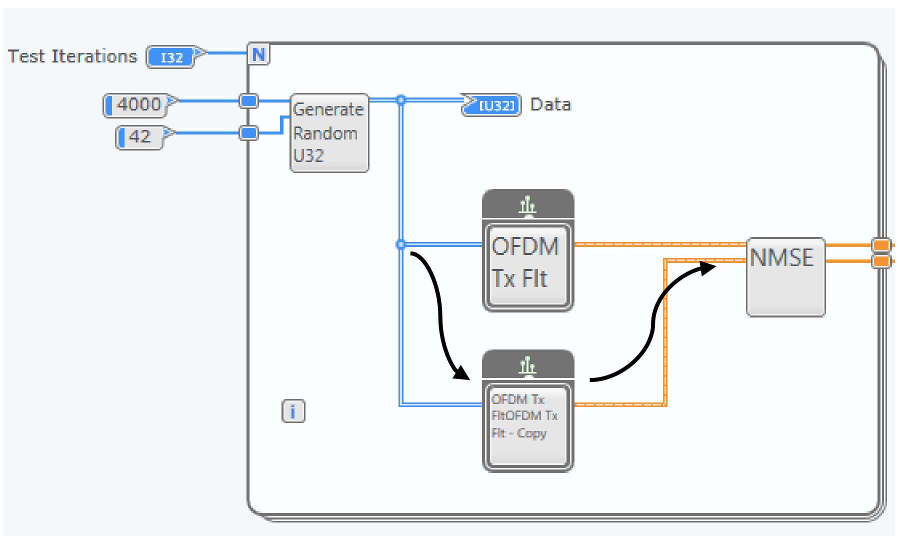
- ***Duplicate Hierarchy for OFDM Tx Flt Multirate diagram***
  Click the OFDM Tx Flt Multirate diagram and ***Edit» Duplicate Hierarchy from Selected*** as shown in ***Figure B***. Name it "**FXP copy**".

- ***Duplicate Hierarchy Dialog Box***
  Click ***Duplicate and Place Multirate Diagram on Cursor.***

A



B



DETAILED INSTRUCTIONS

- ***Place the duplicate Multirate diagram on the diagram***
  Place your cursor below the OFDM Tx Flt Multirate diagram and create the duplicate as shown in ***Figure A***.

- ***Wire the Generate Random U32 node to the OFDM Tx Flt Copy Multirate diagram***
  Hold <Ctrl> while clicking on a wire to create a branch off of that wire.

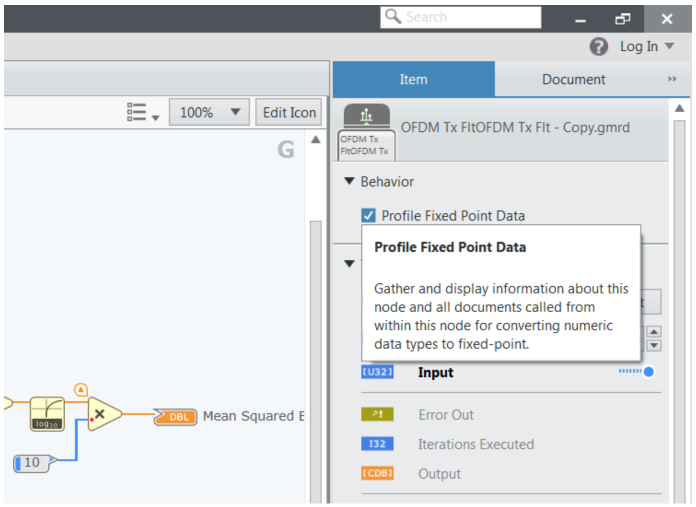
- ***Wire the output of OFDM Tx Flt Copy to the bottom input of NMSE***
  The code should match ***Figure B***.

- ***Run the testbench***
  Press <Ctrl-E> to switch to the panel.

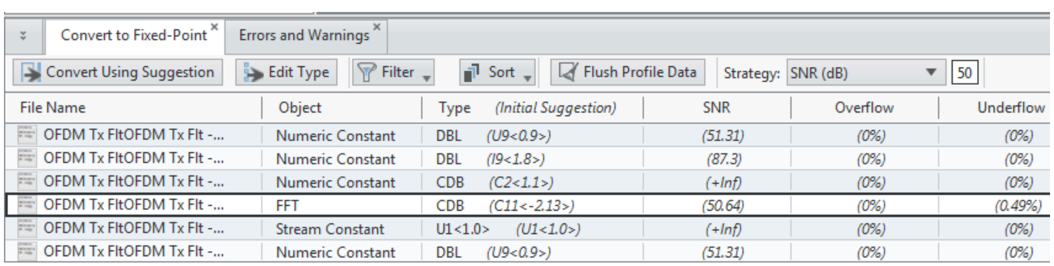
  Run the testbench and observe the mean squared error (MSE). Since we created an exact copy of the OFDM Tx Flt Multirate diagram, there is no difference between the two and thus the error in dB is "-Infinity".

3. Profile the Duplicated Hierarchy

A



B



File Name	Object	Type (Initial Suggestion)	SNR	Overflow	Underflow
OFDM Tx FltOFDM Tx Flt -...	Numeric Constant	DBL (U9<0.9>)	(51.31)	(0%)	(0%)
OFDM Tx FltOFDM Tx Flt -...	Numeric Constant	DBL (I9<1.8>)	(87.3)	(0%)	(0%)
OFDM Tx FltOFDM Tx Flt -...	Numeric Constant	CDB (C2<1.1>)	(+Inf)	(0%)	(0%)
OFDM Tx FltOFDM Tx Flt -...	FFT	CDB (C11<-2.13>)	(50.64)	(0%)	(0.49%)
OFDM Tx FltOFDM Tx Flt -...	Stream Constant	U1<1.0> (U1<1.0>)	(+Inf)	(0%)	(0%)
OFDM Tx FltOFDM Tx Flt -...	Numeric Constant	DBL (U9<0.9>)	(51.31)	(0%)	(0%)

DETAILED INSTRUCTIONS

- ***Save the project***
  Click ***File» Save All.***

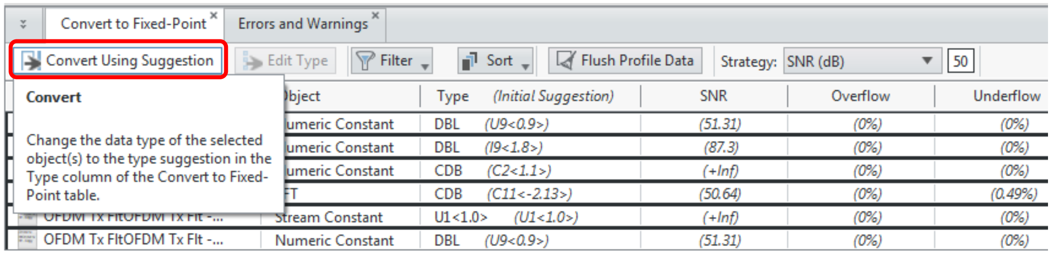
- ***Profile the Fixed-Point Data***
  Switch to the diagram and select the OFDM Tx Flt-Copy Multirate diagram.

  On the ***Item*** tab, select ***Profile Fixed Point Data*** as shown in ***Figure A***.

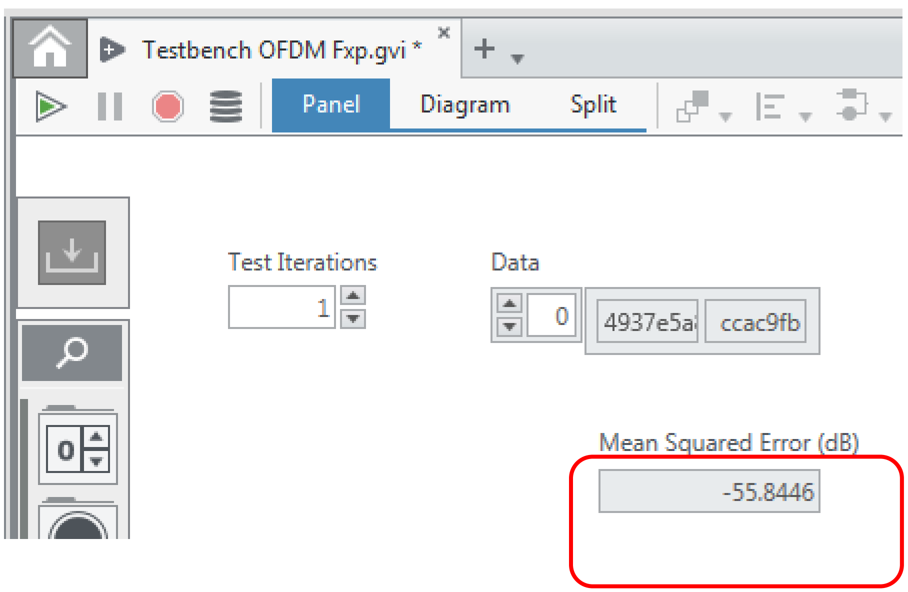
- ***Run the program***
  The fixed-point conversion tool analyzes the diagram and suggests values that produce an output that meets the standard you set in Strategy. LabVIEW Communications populates the table with suggested types. LabVIEW Communications predicts the signal-to-noise ratio (SNR) on the FFT to be 50.64 dB as shown in ***Figure B***.

4. Convert Data Types to Fixed-Point

A



B



DETAILED INSTRUCTIONS

- ***Select all rows in the table***
  Click once inside the Convert to Fixed-Point table to select one of the rows.

  Press <Ctrl-A> to select all the rows in the table.

- ***Click Convert Using Suggestion***
  as shown in ***Figure A***.

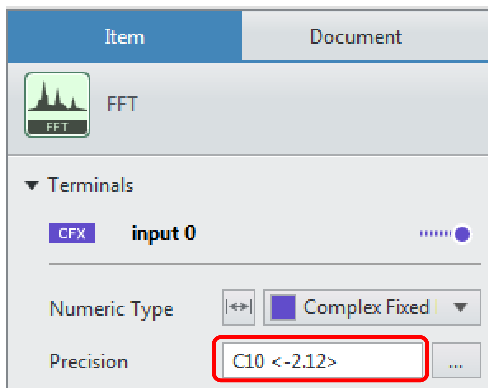
  LabVIEW Communications converts the data types to the suggested fixed-point data types. LabVIEW Communications automatically inserts a conversion node after each constant to preserve the original data type of constants.

- ***Run the program***
  Press <Ctrl-E> to switch to the panel, then run the program to see the results as shown in ***Figure B***.

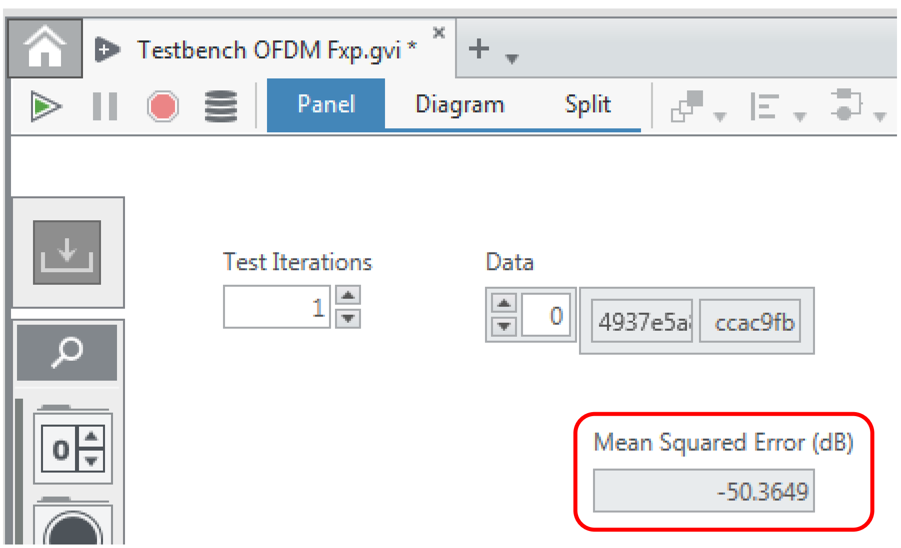
  Compare the mean squared error (MSE) that the testbench calculated with the SNR in the table. The two values are almost identical.

5. Fine-Tune the Fixed-Point Design

A



B



DETAILED INSTRUCTIONS

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

- ***Open the OFDM Tx Flt Copy Multirate diagram***
  Switch to the diagram and double-click to open the OFDM Tx Flt Copy Multirate diagram.

- ***Change the output data type for the FFT node***
  Select the FFT node. In the ***Item*** tab, change the ***Precision*** from -2.13 to ***-2.12*** as shown in ***Figure A***.

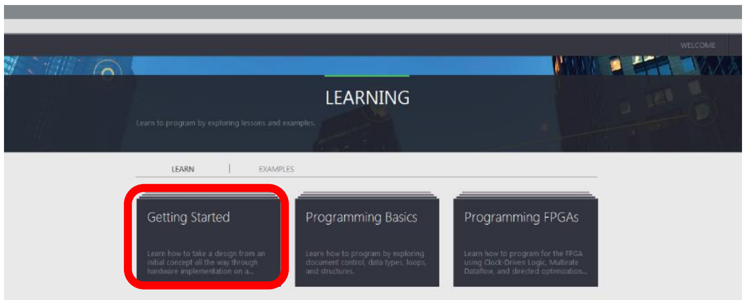
- ***Run the program***
  Press <Ctrl-S> to save the program.

  Switch to the Testbench OFDM Fxp panel and run the program. Notice that the MSE increased from -55 dB to -50 dB as shown in ***Figure B***. In most cases, you can continue making adjustments and running your testbench until you are satisfied with the results.

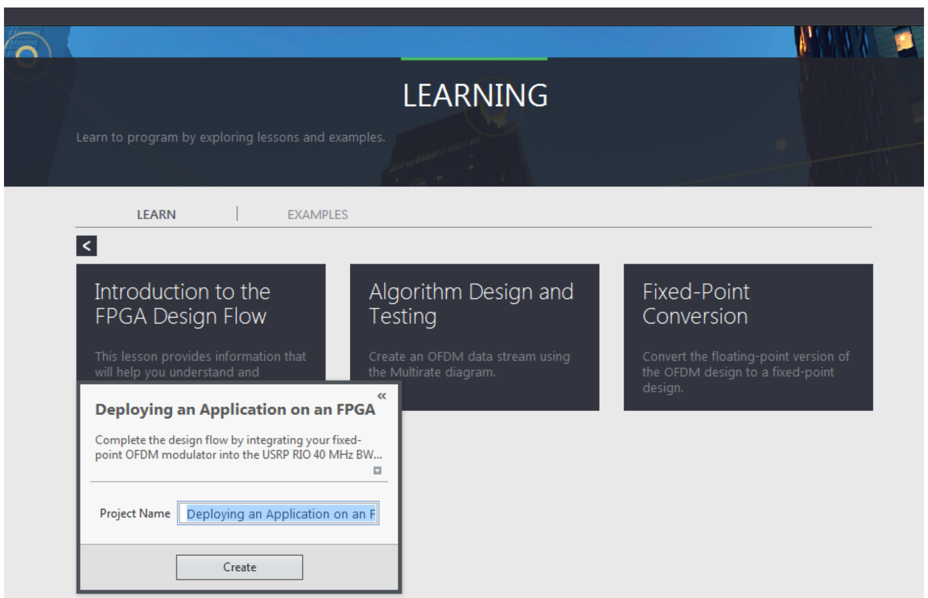
EXERCISE 5 – Deploying an Application to an FPGA

1. Open the Deploying an Application on an FPGA example.

A



B



DETAILED INSTRUCTIONS

In this exercise, 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, compile the build specification, and deploy the application on the FPGA target.

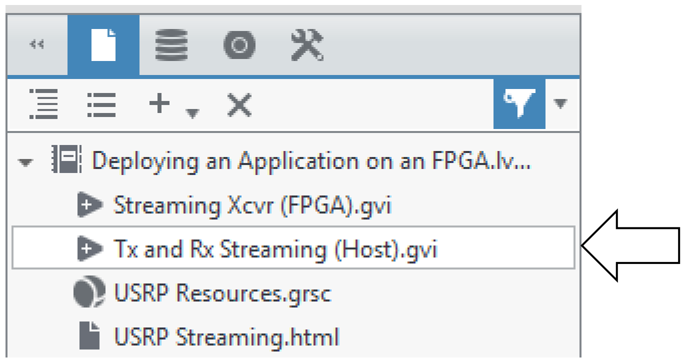
- ***Close Exercise 4***
  Click **File>>Close>>Close Project**

- ***Open the example project***
  On the ***Learn*** tab, find the project by navigating to ***Getting Started» Overview of the FPGA Design Flow» Deploying an Application on an FPGA*** as shown in ***Figure A.***

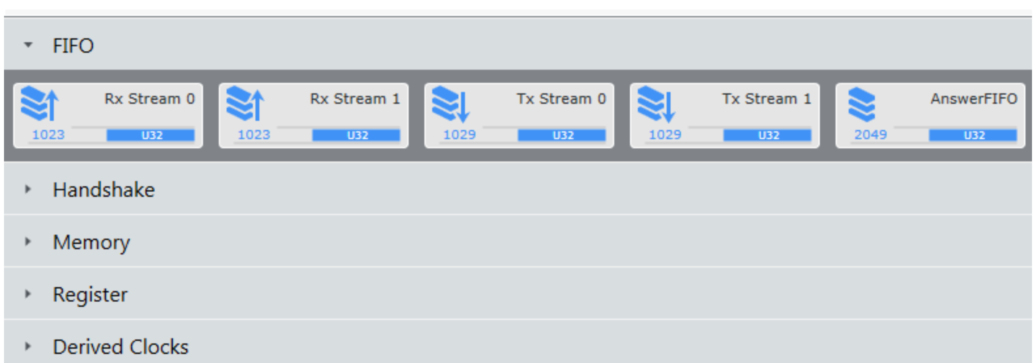
- ***Save the example project***
  Select ***Deploying an Application on an FPGA***, enter a Project Name, and click ***Create*** as shown in ***Figure B***.

2. Explore the Host VI and the FIFOs

A



B



DETAILED INSTRUCTIONS

This example transmits a simple tone using the enabled transceivers. Later, you will modify this VI to send a random bit stream instead of the tone single.

- ***Close the Workbook tab and follow this manual***
  Click the X on the Workbook tab. On your own time, you can follow these workbooks for guided learning, but today we will use this manual.

- ***Open the Tx and Rx Streaming Host VI***
  Double-click on ***Tx and Rx Streaming (Host).gvi*** as shown in ***Figure A.***

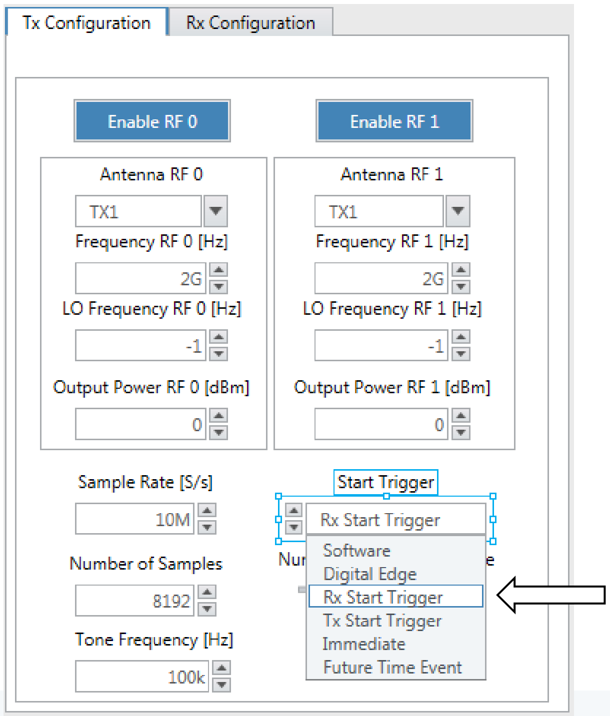
- ***Observe the FIFOs in this project***
  Double-click on ***USRP Resources.grsc*** to view the FIFOs in this project.

  Tx Stream 0 and ***Tx Stream 1*** are Host-to-Target FIFOs that you can use to stream Tx data for RF transceivers RF 0 and RF 1, respectively.

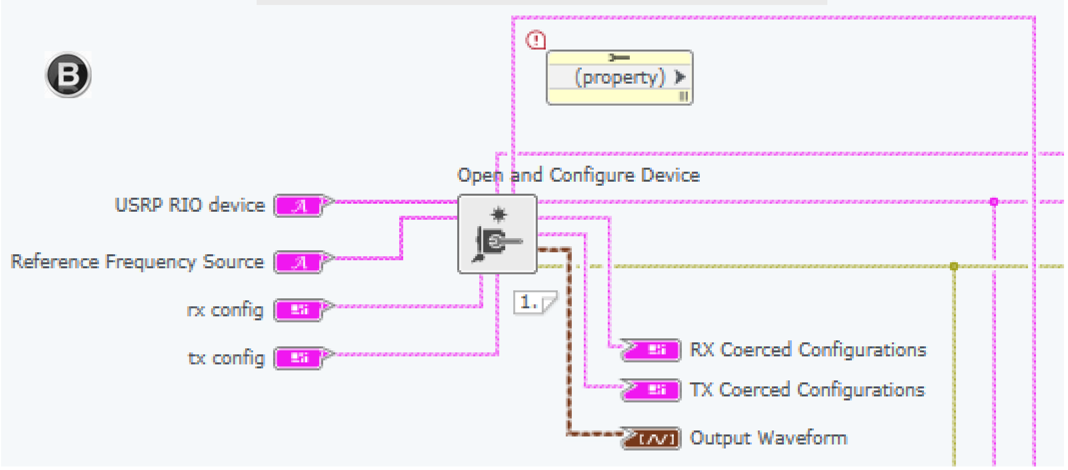
  Rx Stream 0 and ***Rx Stream 1*** are Target-to-Host FIFOs that you can use to stream Rx data from RF transceivers RF 0 and RF 1, respectively.

3. Modify the Tx and Rx Streaming (Host) VI

A



B



DETAILED INSTRUCTIONS

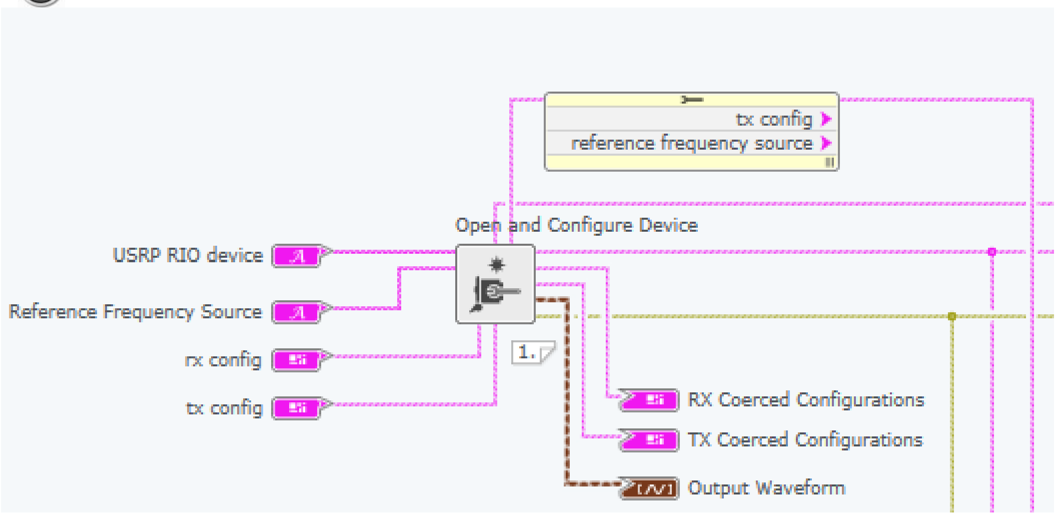
The Tx and Rx Streaming Host VI uses the Initiate and Generate subVI to generate and transmit random samples of a tone to the Streaming Xcvr FPGA VI using the Tx Stream 0 FIFO. You will modify the Tx and Rx Streaming Host VI to generate and transmit random numbers to the Streaming Xcvr FPGA VI.

- ***Adjust the Tx Configuration***
  On the panel of the ***Tx and Rx Streaming Host). VI,*** click the ***Tx Configuration*** tab and update the ***Start Trigger*** control to ***Rx Start Trigger*** as shown in ***Figure A***.

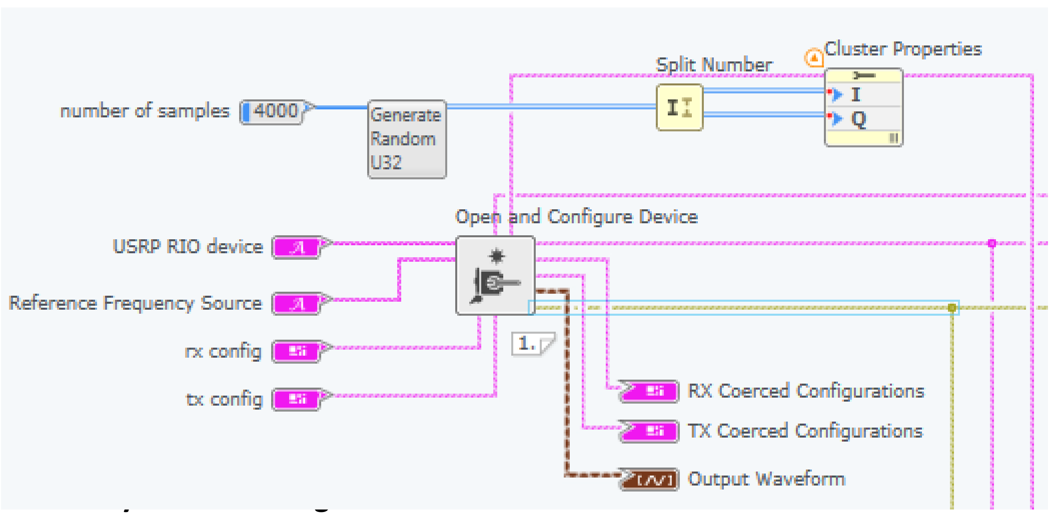
- ***Add a Cluster Properties node to the diagram***
  Switch to the diagram and find the Open and Configure Device subVI on the left side of the diagram.

  Place a Cluster Properties node in the space above Open and Configure Device as shown in ***Figure B***.

A



B



DETAILED INSTRUCTIONS

- ***Expand the Cluster Properties node as shown in Figure A***

- ***Delete and add wires to the Cluster Properties node as shown in Figure A.***

- ***Configure the Cluster Properties node***
  Using the drop-down menu for each property, change ***tx config*** to ***I*** and ***reference frequency*** source to ***Q***.

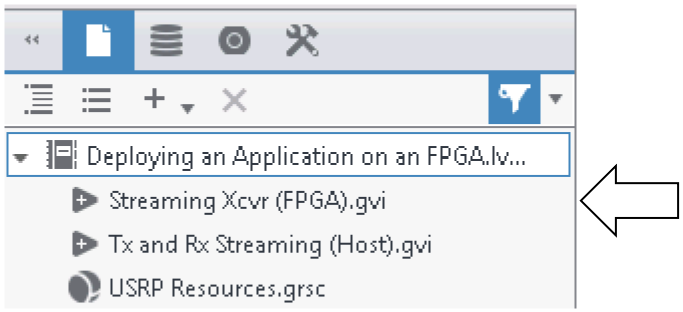
  Right-click the Cluster Properties node and select ***Set All to Write***.

- ***Add Split Number to the diagram***

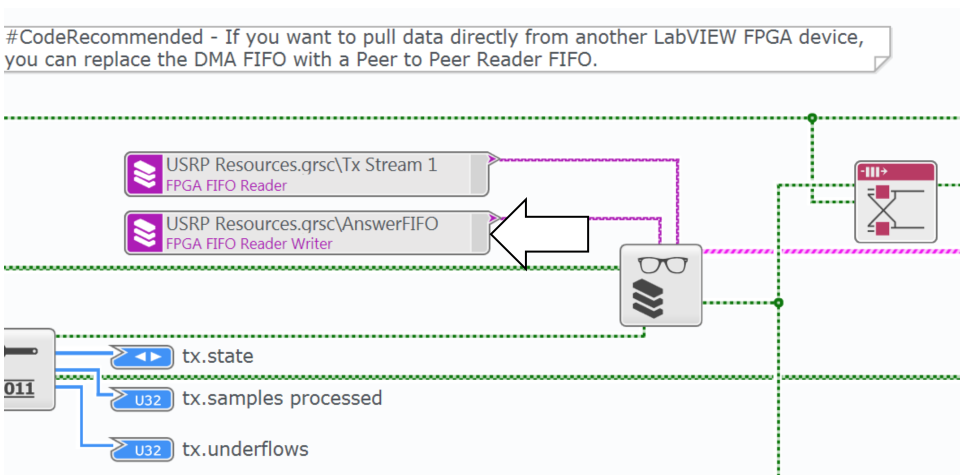
- ***Add Generate Random U32.gvi to the diagram***
  Expand the Deployment Lesson folder on the ***Project Files*** tab.
  - Drag ***Generate Random U32.gvi*** onto the diagram.
  - Right-click the ***number of samples*** input of Generate Random U32 and create a constant. Update that value to ***4000***.

- ***Wire the diagram as shown in Figure B***

A



B



DETAILED INSTRUCTIONS

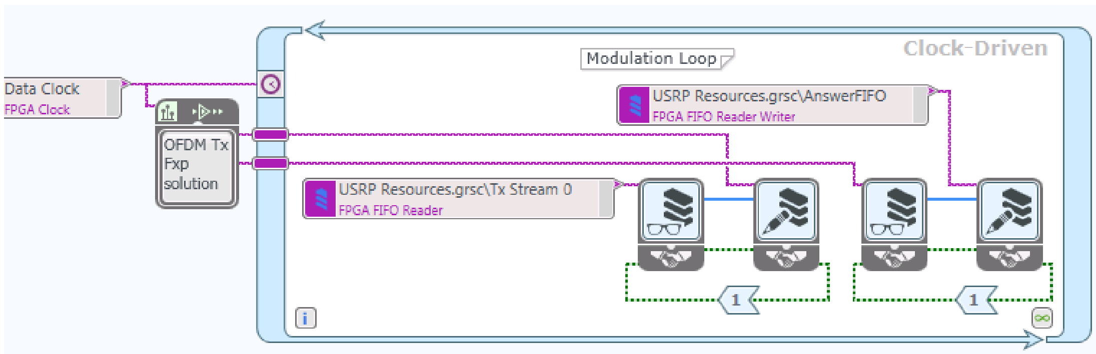
You will now insert your OFDM transmitter onto your FPGA design so that it takes samples from the host FIFO and then passes them to the rest of the transmit chain.

- ***Open the Streaming Xcvr FPGA VI***
  Double-click on ***Streaming Xcvr (FPGA).gvi*** as shown in ***Figure A.***

- ***Locate the Tx Stream 0 FIFO terminal in the large Clock-Driven Loop.***
  The Read FIFO (Multi-Channel) node below this terminal, seen in ***Figure B***, reads from this FIFO and passes data on for upconversion and transmission.

  Hover over the right side of the **Tx Stream 0** terminal until you see a drop down arrow. Click the arrow and select **USRP Resources.grsc\AnswerFIFO**. AnswerFIFO is a local FIFO that allows for the transfer of streams of data between Clock-Driven Loops.

A



DETAILED INSTRUCTIONS

- ***Find the Clock-Driven Loop labeled Modulation Loop at the bottom of the diagram***
  Complete the diagram to match ***Figure A*** and save the VI.

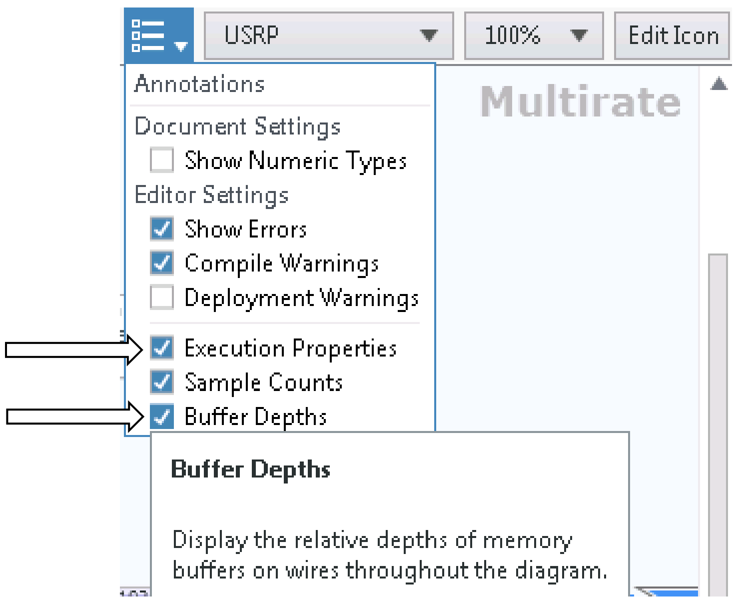
  Notice that when you place ***OFDM Tx Fxp solution.gmrd*** on the diagram from the ***Deployment Lesson*** folder on the ***Project Files*** tab, LabVIEW places it within a container that provides a clock input and FIFO references for all input and output ports. Nodes without the loop use these FIFO references to pass data to and from the algorithm.

  In this Clock-Driven Loop, the first Read FIFO reads data from Tx Stream 0. Read FIFO then passes the data to a Write FIFO that writes the data to the input FIFO of the OFDM modulator. The second Read FIFO reads data from the modulator and passes it to Write FIFO, which writes to AnswerFIFO. AnswerFIFO also sends data to the original Read FIFO (Multi-Channel) node in the main loop. By completing these steps, you have added a modulation stage to the RF0 Transceiver.

4. Perform Design Space Exploration

A

B



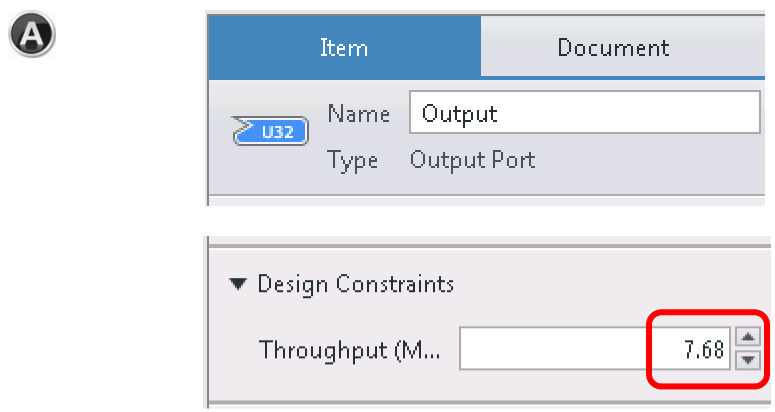
DETAILED INSTRUCTIONS

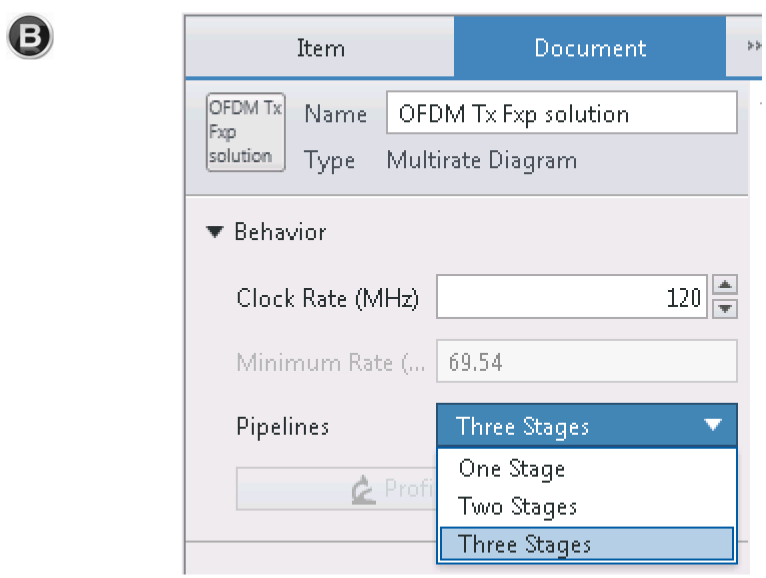
The next stage of the design flow is design space exploration. This is the process of analyzing the performance of your VI or Multirate diagram and exploring design alternatives prior to implementation. For Multirate diagram, LabVIEW computes the appropriate implementation scheme to meet your throughput and clock requirements.

- ***Open the OFDM Tx Fxp solution Multirate diagram by double-clicking it on the diagram*** Ensure that is targeted to the USRP as shown in ***Figure A.***

  On the ***Annotations*** drop-down menu shown in ***Figure B***, enable ***Execution Properties*** and ***Buffer Depths***.

  The properties shown for each node are Executions per Diagram Iteration, Execution Time (cycles), and Initiation Interval (cycles). The properties for each input and output port are Executions per Diagram Iteration, Design Throughput (MS/s), and Calculated Throughput (MS/s).





DETAILED INSTRUCTIONS

- ***Configure the Output port.***
  Select the ***Output*** port. On the ***Item*** tab, in the ***Design Constraints*** section, set ***Throughput (MS/s)*** to ***7.68*** as shown in ***Figure A***. This is one of the standard sampling rates for LTE.

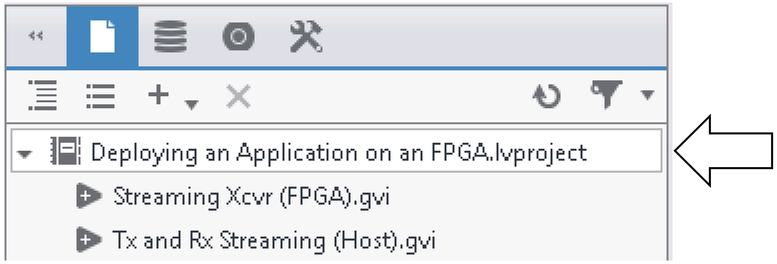
  On the ***Document*** tab, verify that the ***Clock Rate*** equals 120 MHz. When you set the ***Clock Rate***, LabVIEW automatically configures the hardware implementations of the different nodes on your diagram and configures the buffers between them for proper execution.

  On the ***Document*** tab, set ***Pipelines*** to ***Three Stages*** as shown in ***Figure B***. Notice that the sizes of several buffers change on the diagram.

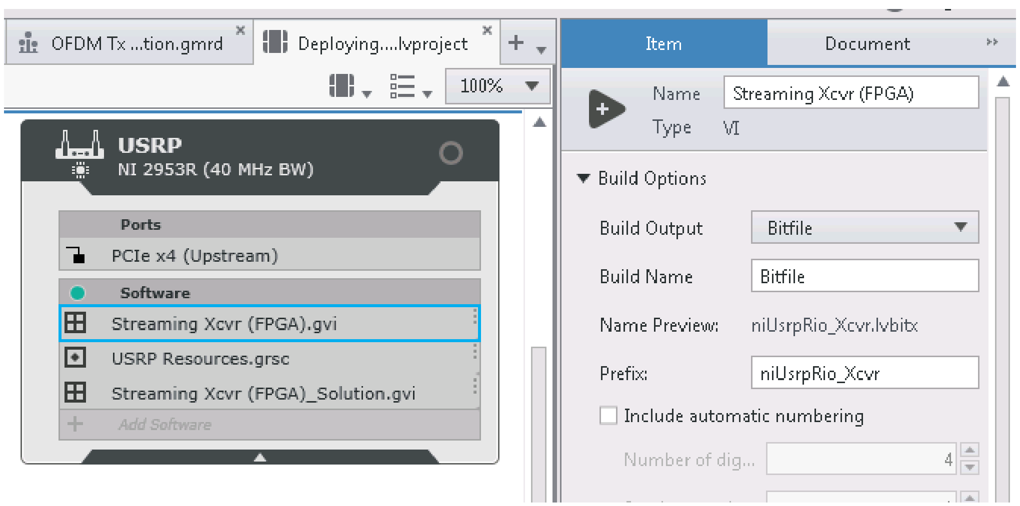
- ***Save the Multirate Diagram***

4. Compile the USRP RIO FPGA VI

A



B



DETAILED INSTRUCTIONS

The last step before running your system is compiling your modified Streaming Xcvr FPGA VI. To compile, your code needs to have a build specification.

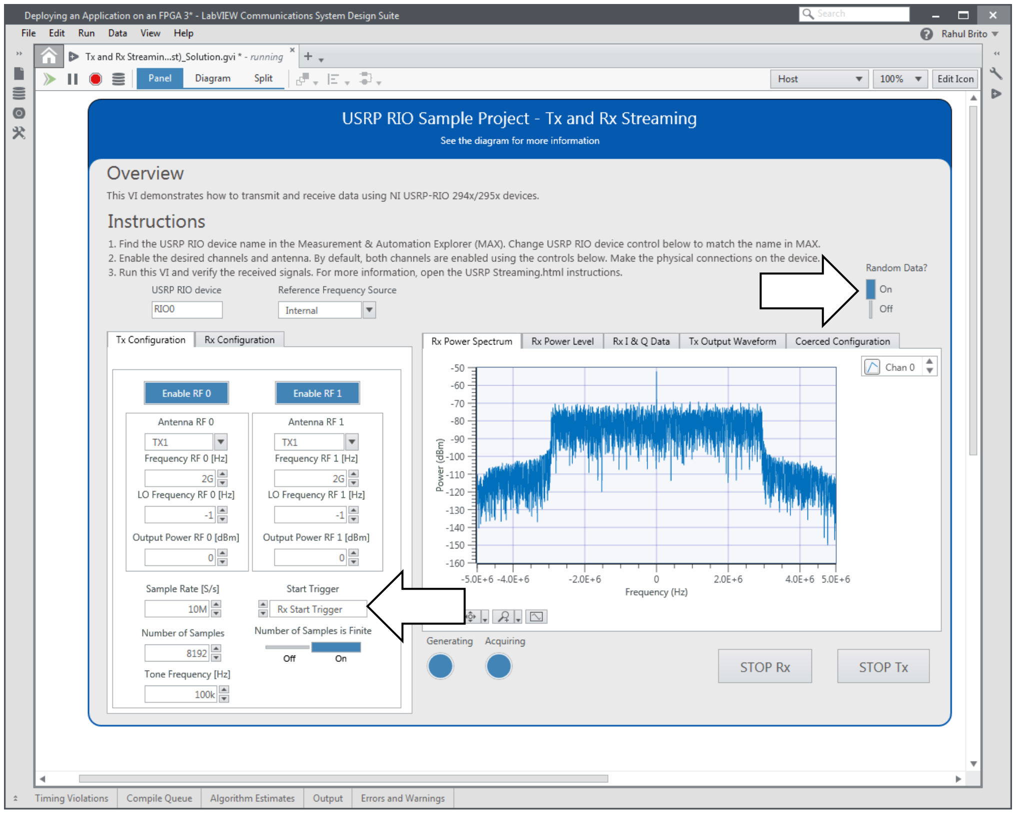
- ***Open SystemDesigner***
  Double-click on ***Deploying an Application on an FPGA.lvproject*** in the ***Project Files*** tab to open SystemDesigner as shown in ***Figure A.***

- ***Select the Streaming Xcvr FPGA VI***
  In the Software category on the USRP RIO target, select the Streaming Xcvr FPGA VI.

  On the ***Item*** tab, verify that ***Build Output*** is set to ***Bitfile***.

4. Run the VI to transmit and receive an OFDM symbol via loopback

A



DETAILED INSTRUCTIONS

Normally, you would compile the FPGA bitfile by pressing Build in the Item tab, but that process can be lengthy. For the seminar, open the solution, which includes a precompiled bitfile.

- ***Open the Tx and Rx Streaming (Host) Solution***
  Navigate to ***Deployment Lesson » Solution Files » Tx and Rx Streaming (Host)_Solution.gvi.***

- ***Configure the Data Source***
  On the Panel, turn the ***Random Data*** control **On**, as shown in ***Figure A.***

- ***Configure when the transmitter triggers***
  In the ***Tx Configuration*** tab, change ***Start Trigger*** to **Rx Start Trigger**, as shown in ***Figure A.***

- ***Run the VI***
  Note that you receive an OFDM symbol, and can verify that the OFDM transmitter we designed has a bandwidth of 5MHz.

  You can stop the VI, change parameters, and re-run to see their effects. All these parameters are exposed by the designer, and you can choose to expose or hide them in your prototype based on your needs.