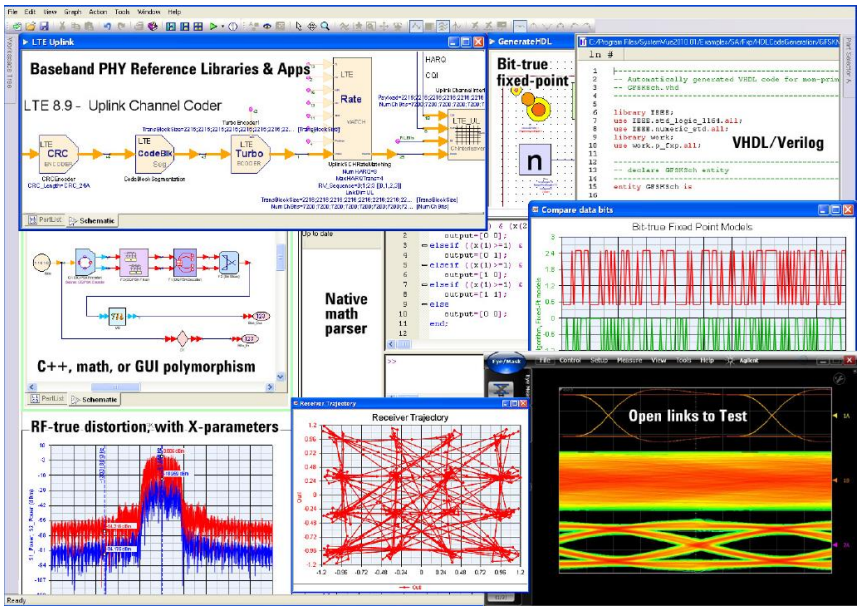
Analysis and Design-RF and Digital Systems Using PathWave System Design (SystemVue)

Training Deliverable



KEYSIGHT TECHNOLOGIES

Contents

Acronyms and Abbreviations

3G	Third-Generation
3GPP	Third-Generation Partnership Project
ACPR	Adjacent Channel Power Ratio
ADS	Advanced Design System
AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
CMOS	Complementary Metal Oxide Semiconductor
CNR	Carrier-to-Noise Ratio
CPU	Central Processing Unit
CW	Continuous Wave
DC	Direct Current
DR	Dynamic Range
DSP	Digital Signal Processing
E-UTRA	Evolved UMTS Terrestrial Radio Access
EVM	Error Vector Magnitude
FDD	Frequency Division Duplex
FSPL	Free-Space Path Loss
G	Gain
HB	Harmonic Balance
HDL	Hardware Description Language
IF	Intermediate Frequency
IMD	Intermodulation Distortion
IP	Intellectual Property
LO	Local Oscillator
LOS	Line-of-Sight
LTE	Long Term Evolution
MDS	Minimum Discernible Signal
MIMO	Multiple Input Multiple Output
NF	Noise Figure
OFDM	Orthogonal Frequency Division Multiplexing
OFDMA	Orthogonal Frequency Division Multiple Access
P1dB	1-dB Compression Point
PLL	Phase-Locked Loop
PRB	Physical Resource Blocks
QPSK	Quadrature Phase-Shift Keying
RF	Radio-Frequency
SAE	System Architecture Evolution
SC-FDMA	Single Carrier Frequency Division Multiple Access
SDF	Synchronous Data Flow
SINAD	Signal-to-Noise and Distortion Ratio
SISO	Simple Input Simple Output
SNR	Signal-to-Noise Ratio (also S/N)
SP	Scattering Parameters
SPARCA	Spectral Propagation and Root Cause Analysis
TA	Transient Analysis
TDD	Time Division Duplex

THD	Total Harmonic Distortion
TOI	Third-Order Intercept Point (also IP3)
UMTS	Universal Mobile Telecommunication System
VCO	Voltage-Controlled Oscillator
VSWR	Voltage Standing Wave Ratio
XP	X-Parameters*

General Introduction

PathWave System Design (SystemVue)

PathWave System Design (SystemVue) is a focused EDA environment for electronic system-level (ESL) design that allows system architects and algorithm developers to innovate the physical layer (PHY) of next-generation wireless and aerospace/defense communications systems. It provides unique value to RF, DSP and FPGA/ASIC implementers who rely on both RF and digital signal processing to deliver the full value of their hardware platforms.

PathWave System Design (SystemVue) replaces general-purpose analog, digital and math environments by offering a dedicated platform for ESL design and signal processing realization. PathWave System Design (SystemVue) "speaks RF," cuts PHY development and verification time in half, and connects to your mainstream EDA flow.

For more information about PathWave System Design (SystemVue), please visit: www.keysight.com/find/eesof-systemvue

Scope

This primer is intended to highlight the system-level simulation techniques and paradigms included in the PathWave System Design (SystemVue) software package.

- Chapters 1-3 focus on how to use this software to architect RF systems, as well as how to model RF blocks.
- Chapters 4-8 delve into the basics of digital communication and provide an introduction into the theory and structure of a digital radio system.

Objectives

1. Learn how Keysight simulators work for RF system-level modeling and begin to document this operation with a specific focus on how these simulators can be used for architecting RF systems.
2. Learn about and document Keysight's collection of RF models, capturing their ranges of operation, support for linear and nonlinear simulation modes, support for noise, etc.
3. Propose a recommended methodology to design RF systems using PathWave System Design (SystemVue).
4. Develop some applications of RF modeling for emerging wireless and aerospace/defense.
5. Learn the basics of digital communications and the theory and structure of a typical digital radio system.

6. Construct the basic components of a digital radio system using PathWave System Design (SystemVue) and investigate typical design considerations.
7. Simulate and analyze results from PathWave System Design (SystemVue) simulations of your digital communication components.

Chapter 1: PathWave System Design (SystemVue) Integrated Simulators

1.1 Introduction

Keysight integrated design environments like PathWave Advanced Design System (ADS) and PathWave RF Synthesis (Genesys) mostly implement the traditional linear and nonlinear simulation techniques for RF design. These techniques do not cover all the behavioral modeling aspects of digital and RF systems. To complete this set of design tools, PathWave System Design (SystemVue) was created. PathWave System Design (SystemVue) is leading-edge system-level design and simulation software developed to support high-level architecting of RF and baseband systems. It includes various simulation technologies covering the baseband and RF fields and operating in both frequency and time domains. PathWave System Design (SystemVue) includes the Data Flow Simulator and Spectrasys simulation cores. It also includes a unique tool for IF frequency planning called the WhatIF frequency planner (also provided in PathWave Advanced Design System (ADS) and PathWave RF Synthesis (Genesys)).

1.2 Traditional Simulation Techniques

PathWave Advanced Design System (ADS) allows for evaluation of an RF schematic using linear and nonlinear simulation techniques. These techniques include, but are not limited to:

- Scattering Parameters (SP) calculation: this kind of simulation evaluates the linear frequency response of an RF device. It is based on the computation of the network scattering and noise parameters.
- Harmonic Balance (HB): a frequency-domain simulation method used to calculate the steady-state response of a nonlinear RF device. It computes, for example, metrics such as total harmonic distortion (THD) and third-order intercept (TOI) points.
- Transient Analysis (TA): this is a time-domain simulation technique used to evaluate the RF device response over time.
- DC Circuit Analysis: this allows the computation of the DC properties of an RF circuit.

Most of these techniques are slow and do not allow the behavioral description of RF systems. For this reason, PathWave System Design (SystemVue) incorporates simulation techniques that are capable of fully describing the behavior of an RF system/device, as well as accurately evaluating its time- and frequency-domain parameters.

1.3 Behavioral Modeling

Keysight introduced a method of architecting and simulating RF systems that is based mainly on behavioral modeling. Keysight products such as PathWave System Design (SystemVue) support this method through the Data Flow Simulator and Spectrasys simulation engines. Behavioral models support different types of spectrum. They are also flexible enough to support future spectrum. Furthermore, in Spectrasys, each port is by default an input and output pin at the same time with regard to the type of spectrum used.

Behavior models support linear and nonlinear devices, as well as frequency- and time-domain analyses. A behavioral model recognizes various types of devices and system models:

- Schematics: a device can be modeled using a circuit schematic that captures its behavior. The schematic blocks may be predefined device models. Dedicated parts such as *RF_LINK* and *CIRCUIT_LINK* can be used to encapsulate low-level circuits and allow their use in high-level behavioral models (e.g., with Data Flow Simulator).
- Equations (e.g., Mathlang/MATLAB): many properties of the device behavior can be modeled using mathematical equations. However, the entire device behavior may not be fully described using equations.
- VB Scripts: Visual Basic scripting language can be used to configure behavioral models. Many of the device properties can be modeled using scripting.
- File-Based Models: using a data file like Touchstone containing an S-parameters file (with possible noise parameters) or X-parameters, the device behavior can also be defined.

1.3.1 Data Flow Simulator

A data flow simulation is used to analyze a communication system at the algorithmic level using time-domain analysis for baseband and RF signals. An RF analysis in Data Flow consists of the time-domain analysis of the modulation information centered at the RF carrier frequency. The Data Flow analysis of RF systems can be carried out using either RF Data Flow models or co-simulation with an RF architecture simulator using the PathWave System Design (SystemVue) RF Design Kit. The Data Flow modeling paradigm is also designed for the computation of baseband system responses. It enables the design and analysis of digital signal processing (DSP) and fixed-point systems, as well as the generation of hardware description language (HDL) code.

The Data Flow simulation controller interacts with Spectrasys (the RF simulator incorporated in PathWave System Design (SystemVue)) using the *RF_Link* part. Using the Data Flow Simulator for mixed baseband and RF systems with dynamic behavior, the designer can figure out the best design topology for the intended circuit, possibly using integrated intellectual property (IP) from previous designs, thereby reducing time-to-market.

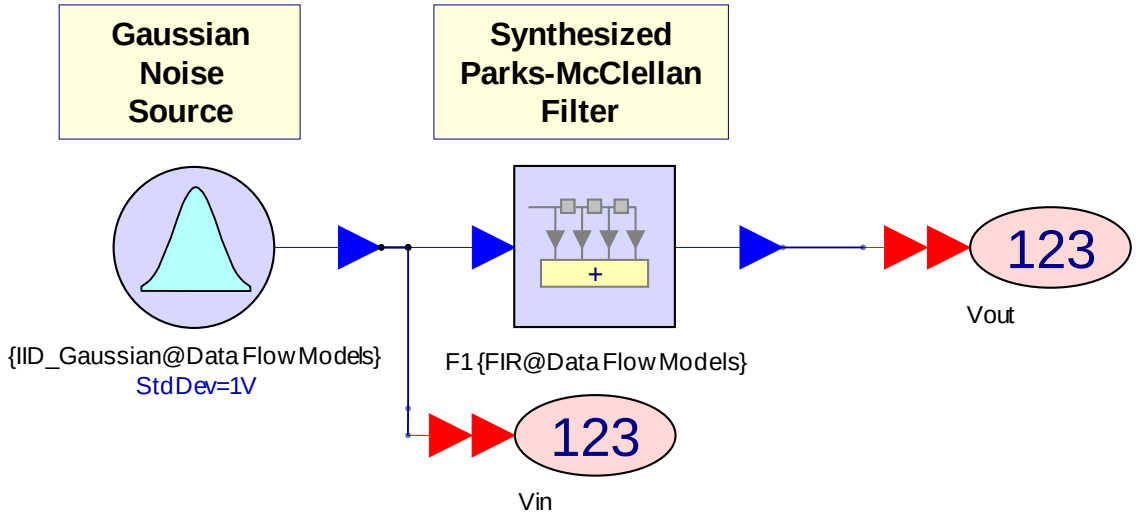
In the Data Flow modeling paradigm, blocks and parts representing RF devices are connected into a Data Flow graph. The graph operates based on a data-driven execution. A given block can execute only when

it has sufficient data in its input edges. It carries out its computations and delivers the resulting data in its output edges. Figure 1.1.a shows an example of a Data Flow simulation where some digital baseband parts are connected together to design a bandpass FIR filter. The input and filtered signals are shown in both time (Figure 1.1.b) and frequency (Figure 1.1.c) domains.

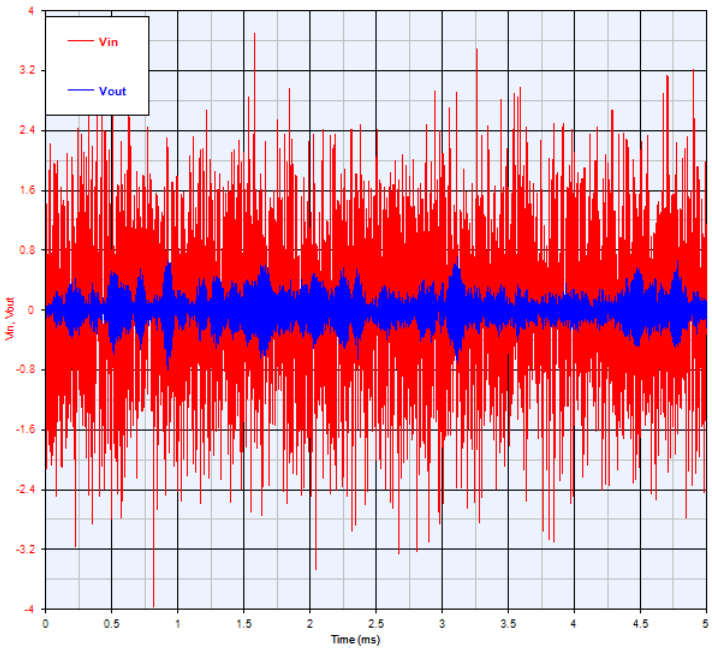
Most of PathWave System Design (SystemVue)'s parts and blocks work according to the Synchronous Data Flow (SDF) technique[1] where the number of samples and the source production and target consumption rates are preconfigured and known before simulation. However, some parts in PathWave System Design (SystemVue) can be used to model the dynamic behavior of communication systems. Among these parts, we cite the special blocks: *DynamicPack_M* and *DynamicUnpack_M*.[2]

The Data Flow simulation technology has various other features that are implemented in PathWave System Design (SystemVue). Among these features, the most outstanding are:

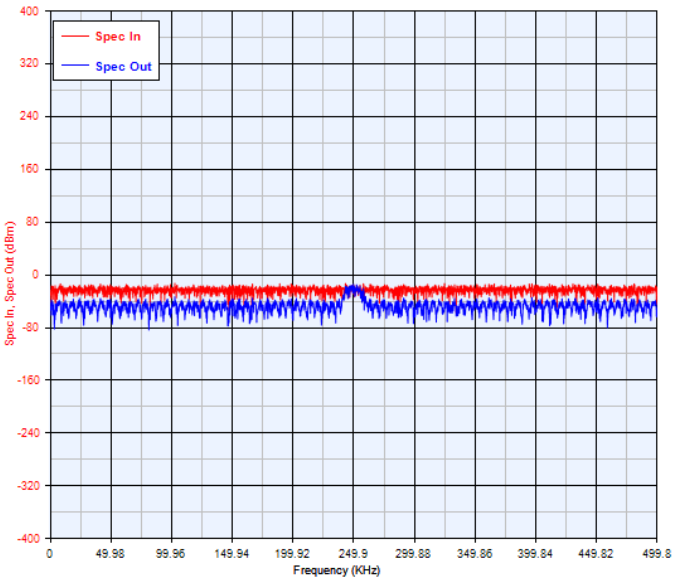
- Usage of multiple data type. This feature allows for the mixed use of various types of data that are scalar, fixed-point or matrix, for example.
- Fixed-point simulation. PathWave System Design (SystemVue) provides in its HDL an extensive set of fixed-point parts and models that can be used to build, simulate and analyze fixed-point systems. These fixed-point systems can then be directly converted into HDL and C++.
- Distributed simulation. PathWave System Design (SystemVue) provides an option to run Keysight's PathWave RFIC Design (GoldenGate) simulations remotely. This option is useful if you want to run the simulation on a machine with more memory or a faster CPU.
- Co-simulation with other languages and simulation packages. PathWave System Design (SystemVue) allows the designer to co-simulate HDL code using ModelSim and .m files using MATLAB.



(a)



(b)



(c)

Figure 1.1. An example of digital baseband parts used in Data Flow simulation: (a) Typical baseband filtering schematic (b) Input and Filtered Gaussian Noise Waveforms (c) Input and Filtered Gaussian Noise Spectra.

1.3.2 Spectrasys

Spectrasys is a Keysight technology that implements the Spectral Propagation and Root Cause Analysis (SPARCA) simulation technique to support measurement-based and behavioral modeling. Spectrasys grants PathWave System Design (SystemVue) a unique ability to perform high-level modeling and

architecting of RF devices and systems with accuracy and ease-of-use.

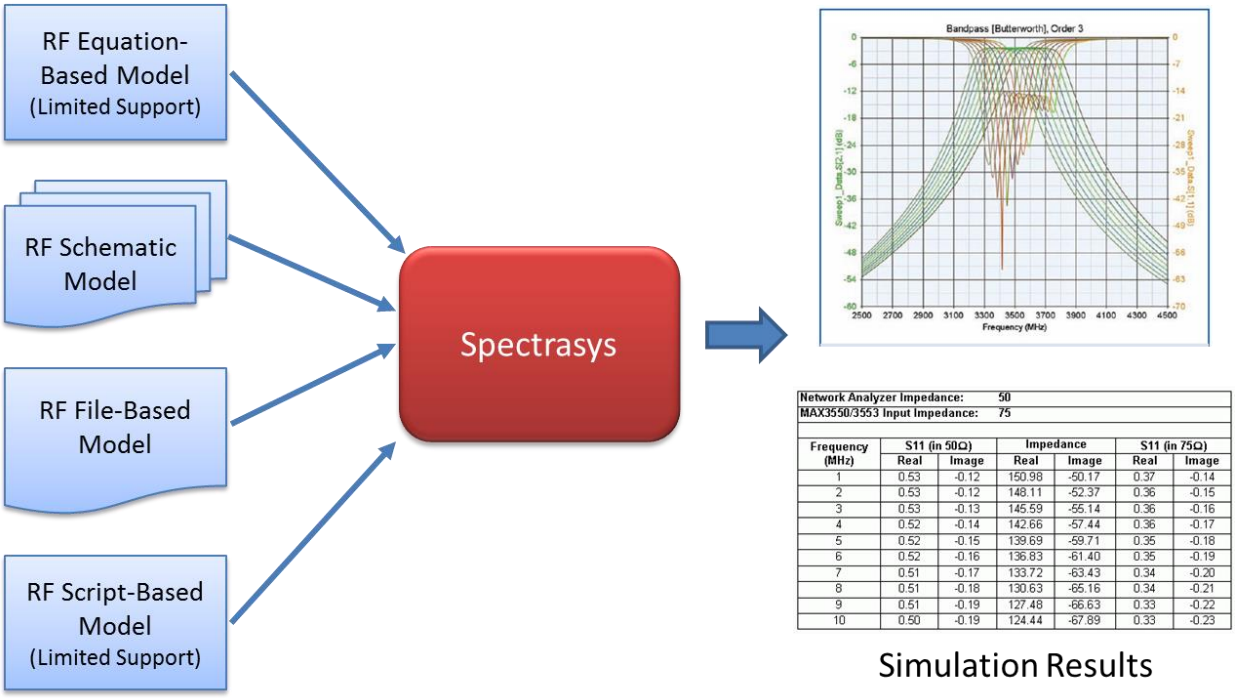
SPARCA supports the following types of spectrums:

- Signal: includes source signals and carriers.
- Intermodulation and Harmonics: resulting from the nonlinearity of certain RF devices such as mixers and amplifiers.
- Broadband Noise*:* resulting from the thermal noise in the RF circuits.
- Phase Noise: covers the issue of phase noise through an RF system.

SPARCA considers the fact that each spectrum propagates both forward and backward to every node of the schematic. Then, all noise and linear and nonlinear parameters are computed accordingly. Besides the fact that SPARCA simulation is very fast compared to traditional nonlinear simulation techniques, which require mostly convergence criteria and huge algebraic computations, this novel simulation technique has several advantages. A few of the benefits for using SPARCA instead of traditional simulation techniques are:

- Bandwidths are calculated for all spectrums
- Broadband noise can be easily estimated
- Behavioral phase noise effect analyses
- Multiple path analysis
- Signal leakage path analysis
- Flexibility for future spectrum support and enhancements

Figure 1.2 presents the behavioral models that can be handled by Spectrasys. The simulation engine accepts RF architectures with different RF parts and blocks (Figure 1.3) and delivers the subsequent response.



Frequency (MHz)	S11 (in 50Ω)		Impedance		S11 (in 75Ω)	
	Real	Image	Real	Image	Real	Image
1	0.53	-0.12	150.98	-50.17	0.37	-0.14
2	0.53	-0.12	148.11	-52.37	0.36	-0.15
3	0.53	-0.13	145.59	-55.14	0.36	-0.16
4	0.52	-0.14	142.66	-57.44	0.36	-0.17
5	0.52	-0.15	139.69	-59.71	0.35	-0.18
6	0.52	-0.16	136.83	-61.40	0.35	-0.19
7	0.51	-0.17	133.72	-63.43	0.34	-0.20
8	0.51	-0.18	130.63	-65.16	0.34	-0.21
9	0.51	-0.19	127.48	-66.63	0.33	-0.22
10	0.50	-0.19	124.44	-67.89	0.33	-0.23

Figure 1.2. Various RF models can be handled by Spectrasys.

SPARCA was introduced through Spectrasys to complement existing simulation techniques. It allows more accuracy, speed and productivity at the system level and facilitates the study of the specifications and trade-offs that relate to a given RF architecture.

Spectrasys has a myriad of outstanding features to help designers build, simulate and analyze communication systems and RF architectures. Among these features, are:

- **Sources**: Spectrasys supports various types of complex spectral-domain RF sources and spectrum plots such as continuous-wave (CW), wideband, noise and complex spectral data created by simulation or measurement.
- **Channels**: Spectrasys makes measurements using mainly frequency channels that consist of a center frequency and a bandwidth.
- **Paths**: As stated before, Spectrasys considers that each part pin is both an input and output port and computes the frequency response of incoming and ongoing waves. Additionally, Spectrasys allows the definition of paths along the arbitrary architectures. This enables the examination of the frequency response of each path and helps designers better investigate the system.
- **Level Diagrams***:* these diagrams give the user a quick visual indication of the performance of the entire cascade. It displays measurements of cascaded stages along a user-defined path.
- **Spectral Origin Identification**: since each spectrum is tracked individually, the user can find the origin and path of each spectrum of interest.

- **Broadband Noise**: Spectrasys computes through the SPARCA simulation technique, the computation of the broadband noise over the different stages in a given path and keeps track of the noise level in each device part.
- **Intermodulation and Harmonics**: Spectrasys computes intermods, harmonics and intercept points for each spectrum. It allows advanced analysis of the simulation results by displaying various parameters of nonlinear behavioral models (such as reverse isolation, intercept points, and harmonics, etc.).
- **Coherency**: Since all signals in Spectrasys are treated on an individual basis, coherency for each of the spectrums is created during the simulation. Coherent signals will add in both voltage and phase, whereas non-coherent signals will add in power.
- **Behavioral Phase Noise**: The SPARCA engine supports behavioral phase noise that can be specified for certain source and oscillator models. Phase noise is an independent type of spectrum and as such measurements can operate on this spectrum in the presence of other spectrums of different types. This independence allows phase noise to be modified through mixers, multipliers and dividers without affecting the parent spectrum.
- **Sweep and Circuit Co-Simulation**: Spectrasys allows sweeping operations and co-simulation of RF architectures using other simulators (linear simulators, Harbec for harmonic simulation, etc.).
- **Optimization**: Optimization is a technique used to solve practical design problems such as standard values and component tolerances that may be purely theoretical. In particular, it is used when the number of design variables is high and the tuning becomes less effective due to the huge multidimensional variable space.
- **Monte-Carlo and Yield Analysis**: Monte Carlo and Yield Analysis are probabilistic techniques used to simulate component variations caused by the production process. These methods can be used to determine which components need to be low tolerance (usually more expensive components) or to help create designs that are able to accommodate parameter variation.

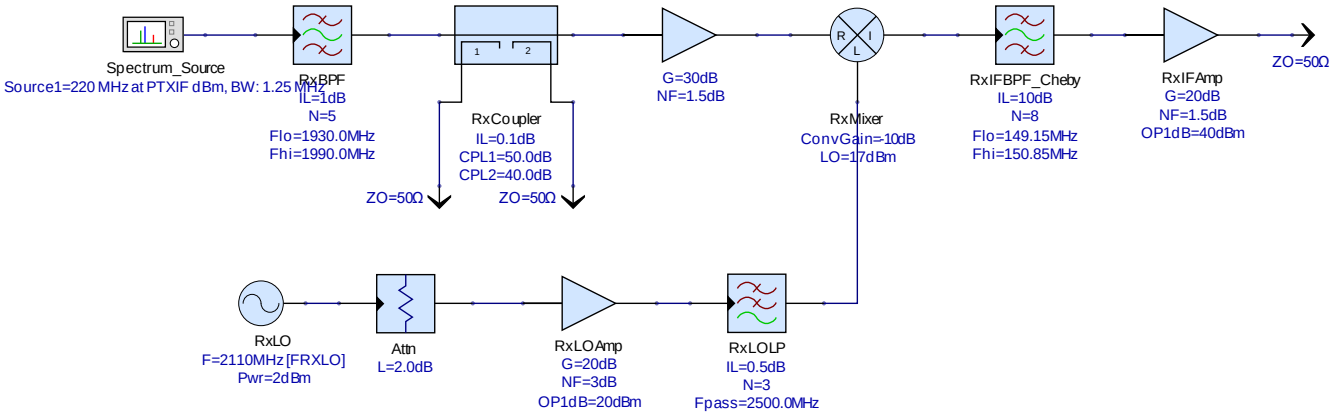


Figure 1.3. An example of RF receiver chain built in PathWave System Design (SystemVue).

1.3.3 WhatIF Frequency Planner

The WhatIF Frequency Planner is a unique synthesis technique to determine the desired and undesired RF, LO and IF signals. This technique replaces home-made custom tools, as well as the awkward spreadsheets methodology since the latter consists only of analytical equations programmed for each path-of-interest. It is highly limited in results because it is a scalar and unilateral method and assumes unfiltered and flat frequency response. It does not account for various effects such as VSWR. For these reasons, the WhatIF Frequency Planner was conceived to automate the search of IF frequencies that are spur-free.

The WhatIF Frequency Planner computes all harmonic combinations of the RF and LO signals and shows their spurious performance, as well as the spurious free regions of the spectrum, on the same graph (Figure 1.4). The displayed information includes bandwidths and spurious responses amplitude. The WhatIF Frequency Planner simulation technique is fast, very useful and handy for system designers, enabling them to investigate the trade-offs that relate to the intended system architecture. It plays a substantial role in helping designers select a suitable frequency plan for their system.

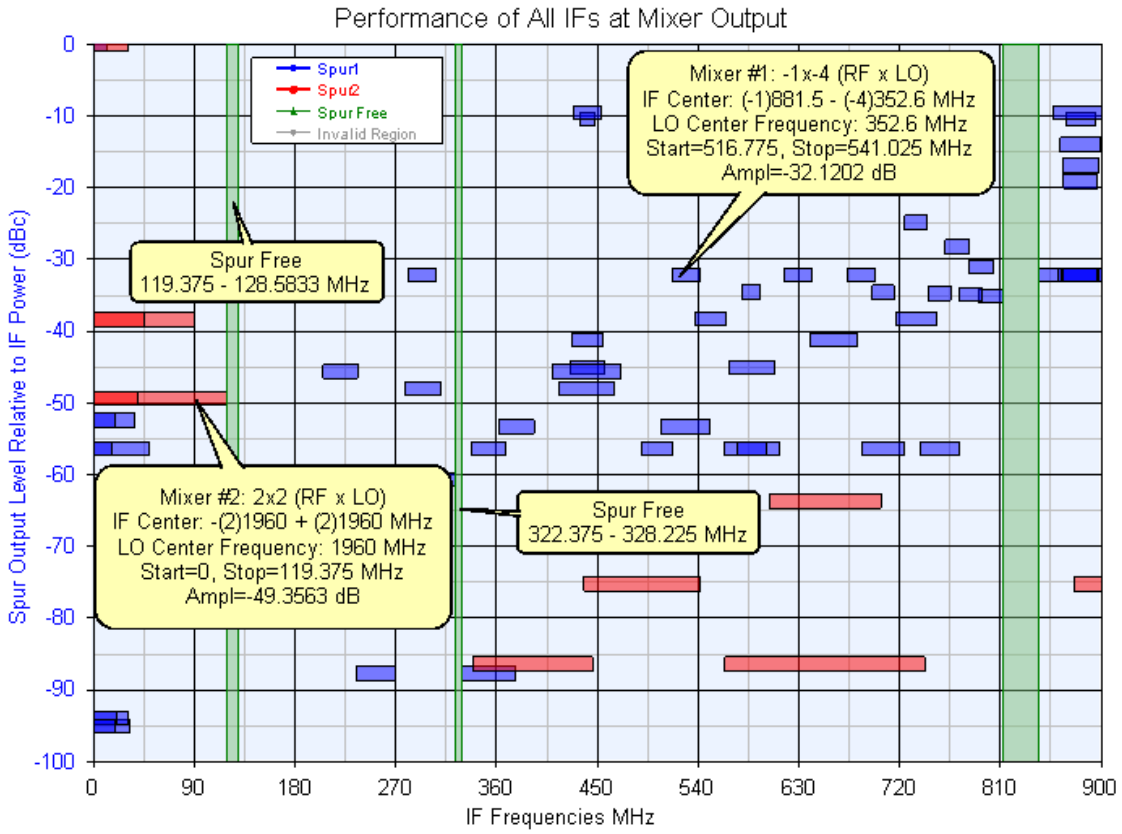


Figure 1.4. The results of the WhatIF Frequency Planner Simulation for a typical RF mixer.

1.3.4 Parameter Sweep

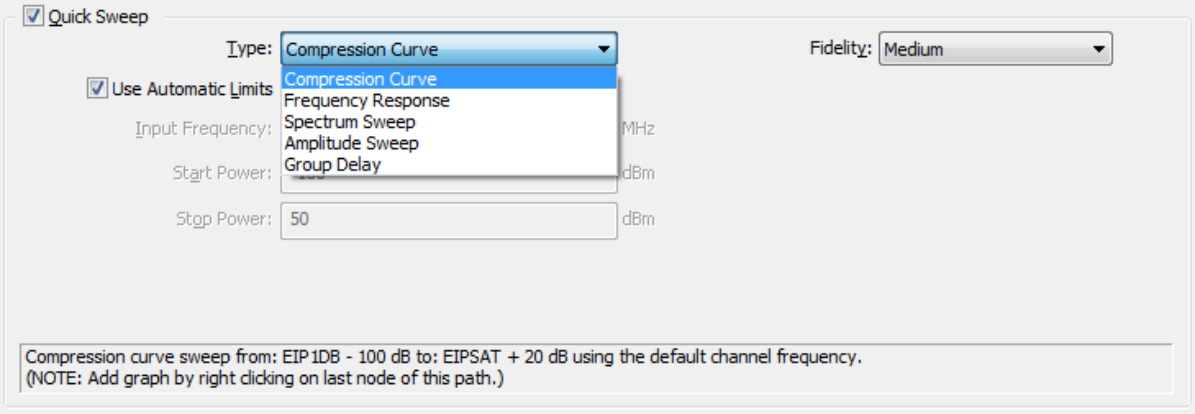
A parameter sweep is used to automatically create analysis results that are a function of a given parameter tuned in a given range. This range is often specified by the user.

There are two ways to make a parameter sweep using PathWave System Design (SystemVue):

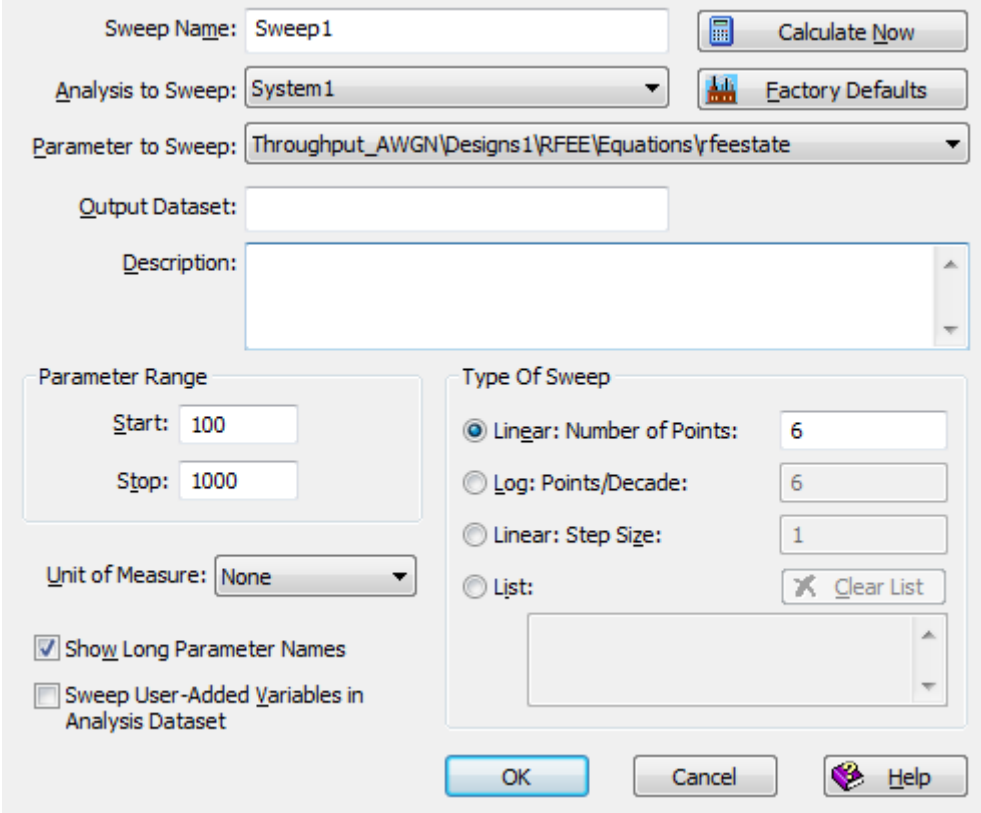
a- **Quick Sweep***:* This option checks the path properties and allows the sweeping of several traditional parameters such as frequency, amplitude, spectrum, and group delay (Figure 1.5.a).

b- **Parameter Sweep***:* This sweep technique allows the variation of any parameter specified by the user and made tunable. It then computes the analysis results for all user-specified points (Figure 1.5.b).

Parameter sweeping allows the analysis automation of the circuit at various points and the evaluation of its response at each one of them. This helps the designer estimate the circuit performance and modify it accordingly to fit initial specifications.



(a)

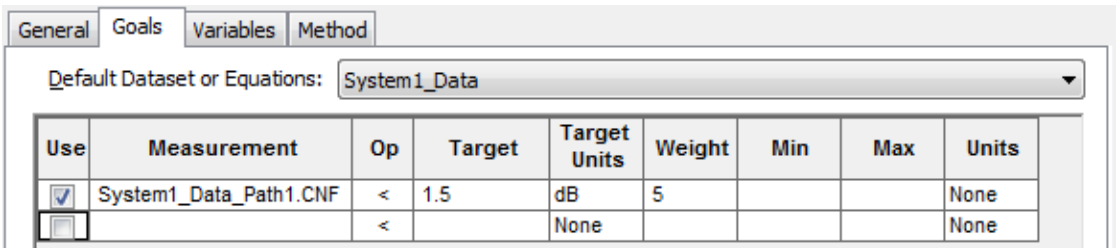


(b)

Figure 1.5. Parameter Sweep: (a) Quick sweep (b) Parameter sweep properties.

1.3.5 Behavioral Optimization

PathWave System Design (SystemVue) allows for the automatic variation of circuit variables. It includes an optimization tool that can be used to conduct an automated search of the optimal values in a wide variable space. Using this tool, the user can specify the target performance (using optimization goals) and the part variables to tune (Figures 1.6.a and 1.6.b). During the optimization process, the values of the specified variables are varied according to a given algorithm and an error function is evaluated in real time to check if the required goals are met. The circuit response and schematic are then automatically updated (Figure 1.6.c). When the optimization process comes to its end, the old performance and the new one are both displayed on a chart to let the designer decide about the new circuit configuration (Figure 1.6.d).

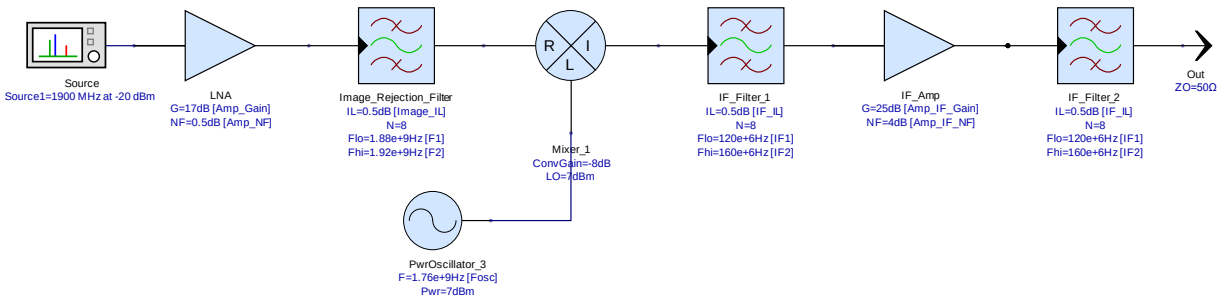


(a)

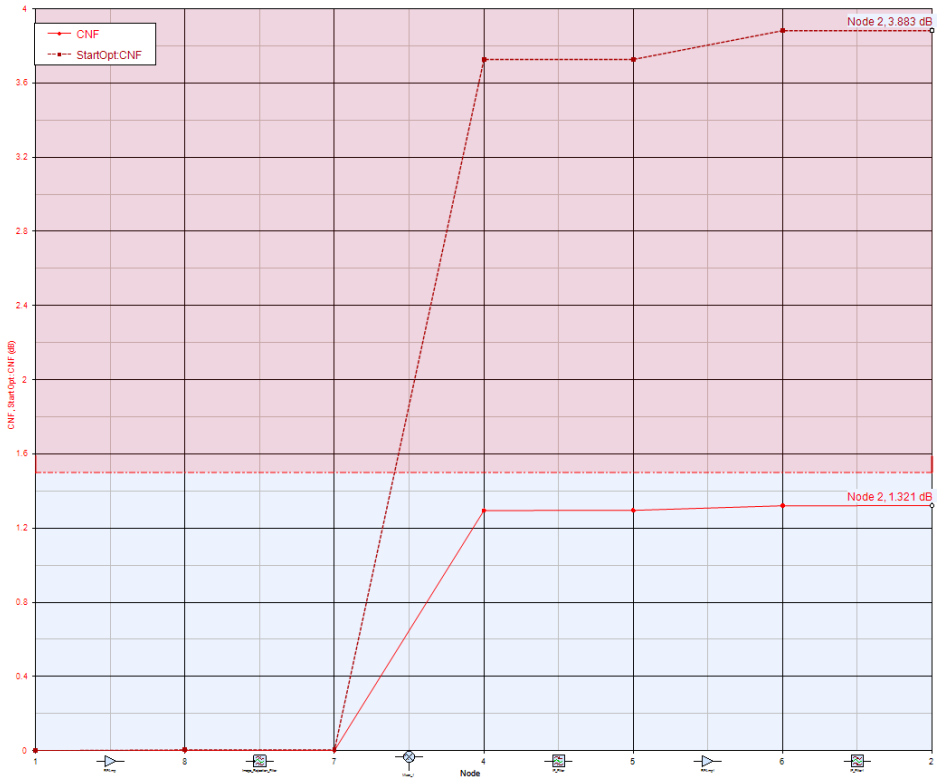
General | Goals | Variables | Method

Use	Variable	Min	Max	Units
☑	Equation1.Amp_Gain			None
☑	Equation1.Amp_NF			None
☑	Equation1.Image_IL			None
☑	Equation1.IF_IL			None
☑	Equation1.Amp_IF_Gain			None
☑	Equation1.Amp_IF_NF			None

(b)



(c)



(d)

Figure 1.6. Optimization properties: (a) Example of optimization goals (e.g., Cascaded Noise Figure) (b) Example of optimization variables (c) Example of component values being tuned in real-time during the optimization (d) The results of Cascaded Noise Figure optimization.

1.4 Summary

PathWave System Design (SystemVue) is Keysight's cutting-edge software for high-level design and simulation of RF architectures in communication systems. PathWave System Design (SystemVue) allows users to build baseband and RF architectures from a systems perspective. It includes two simulation engines that allow the simulation of arbitrary architectures. The Data Flow Simulation engine fully supports digital systems. It enables the evaluation of RF architectures through RF Design parts. This simulation technique can be used jointly with Spectrasys, the RF simulation core of PathWave System Design (SystemVue). Intermodulation products, noise, harmonics, and intercept points can be easily depicted using Spectrasys. The optimization and tuning tools included in PathWave System Design (SystemVue) help designers figure out the best trade-offs. PathWave System Design (SystemVue) also includes the WhatIF Frequency Planner tool that allows the user to easily evaluate the spurious performance of a given system. Spurious-free bandwidths and contaminated frequencies are shown, as well as the amplitude of the different signals. Furthermore, the software includes tools that can automate parameter sweeping and optimization of circuit variables.

Chapter 2: RF System Design Basics

2.1 Introduction

In the previous chapter, we presented some features and capabilities of PathWave System Design (SystemVue)'s integrated simulators. In this chapter, we will provide some necessary RF definitions and basic system design concepts that will be needed in the following chapter to illustrate PathWave System Design (SystemVue)'s design and simulation capabilities.

2.2 Transceiver Design

A transceiver is a radio communication system that consists of a transmitter and receiver. A transmitter is a radio system that converts baseband data into an RF signal and then transmits it through a communication channel, commonly using an antenna (Figure 2.1). An RF transmitter performs three basic functions: modulation, frequency translation and power amplification. Three main performance metrics are required in a typical transmitter: accuracy, spectral emission and output power level.

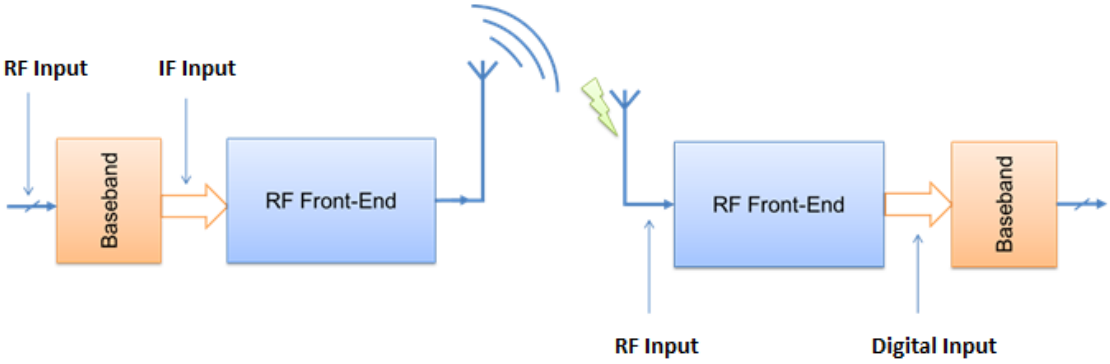


Figure 2.1. An example of a transceiver.

Various architectures are used to design an RF transmitter depending on the cost, form factor and power required for each application. The number of communication standards that have to be supported by the transmitter impacts the architecture to be selected. Additionally, most transmitting architectures are similar to those used in the receiver design such as heterodyne and homodyne architectures.

Transmitter architectures can be classified into two main categories:

- **Mixed-based architectures**: among these architectures, we count heterodyne (i.e. dual-conversion) and homodyne (i.e. direct-conversion) transmitters. These architectures can operate with constant and non-constant envelope modulation schemes and are convenient for multi-standard operation.

- **Phase-Locked Loop (PLL)-based architectures**: these architectures present a more compact form factor due to the removal of discrete components. However, they are less-suited for multi-standard operation because they support only constant-envelope modulation schemes.

A PLL-based architecture is generally built around a voltage-controlled oscillator (VCO). Figure 2.2 shows a typical PLL-based transmitter architecture. In this transmitter, the PLL is designed to maintain a constant center frequency to accurately control the transmitted signal. Refer to references 3, 4 and 5 for more detailed examples of transmitter architectures.

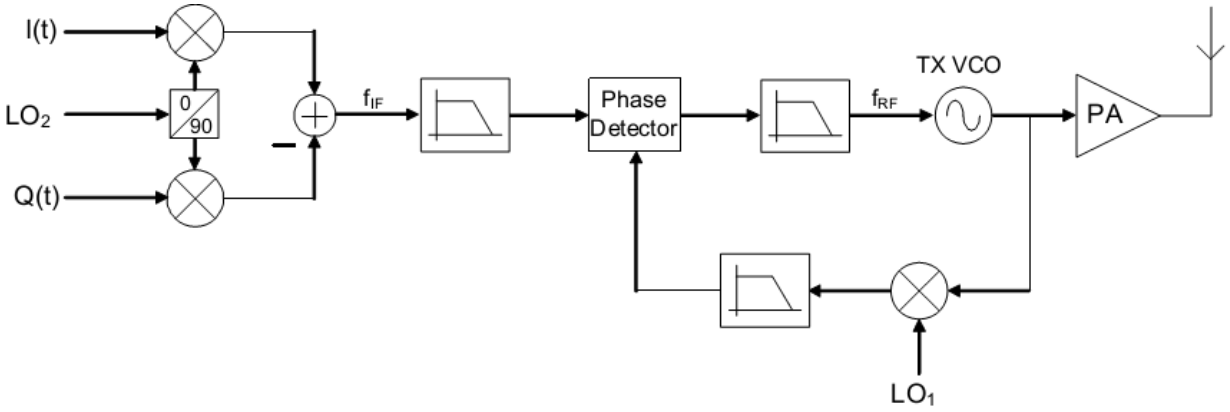


Figure 2.2. Offset PLL architecture.[3]

2.3 Receiver Architectures

A receiver is a radio communication system intended to receive a range of frequencies and process them to recover the transmitted data. It is mainly composed of an antenna system, an RF front-end and a baseband stage where basic signal processing operations are carried out. Figure 2.3 shows the general structure of an RF receiver.

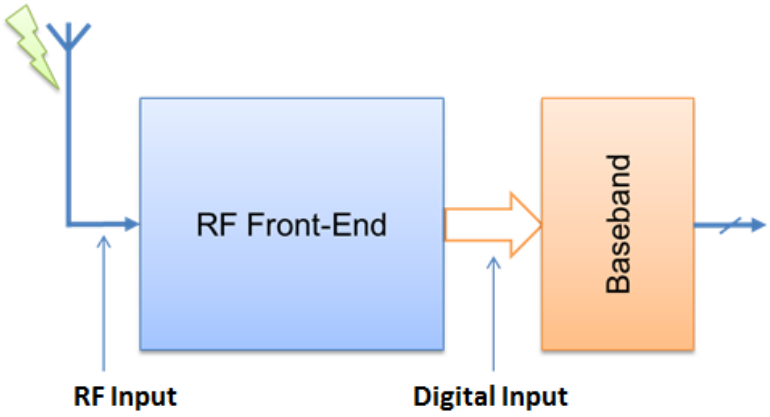


Figure 2.3. Typical receiver architecture.

Historically, RF designers have experienced a myriad of receiver architectures. The most common architectures used to build receivers are briefly presented here; however, for more extensive details on this topic, refer to reference 6.

2.3.1 Super-Heterodyne Receivers

A super-heterodyne receiver (also called two-stage conversion) is composed of two banks of mixers that downconvert signals from RF to IF in the first stage, and from IF to baseband in the second stage (Figure 2.4). Multiple filters with different technologies and specifications are required either to select the RF channel or for image rejection.

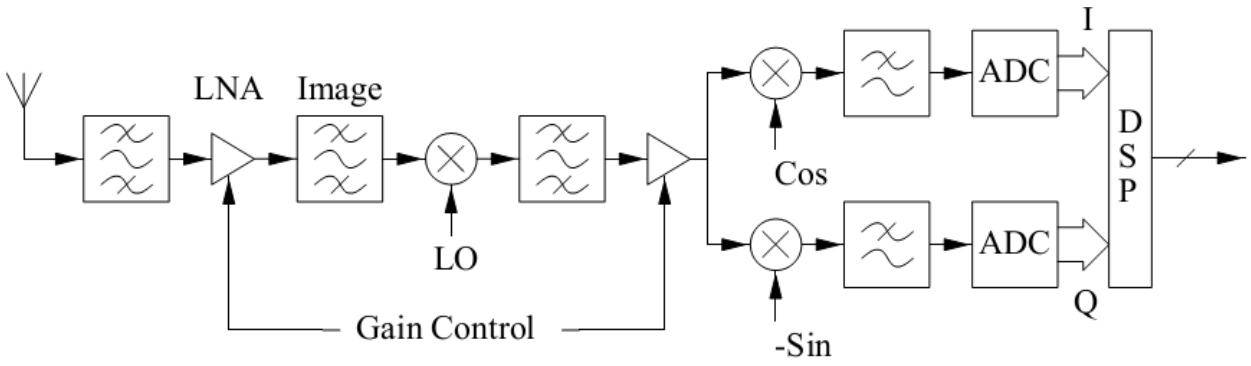


Figure 2.4. A typical super-heterodyne receiver.

2.3.2 Low-IF Receivers

To accommodate different modulation schemes, modern designers use a low-IF architecture in developing receivers (Figure 2.5). This architecture downconverts RF signals to a very low IF frequency. Using a high-resolution analog-to-digital converter (ADC), the low-IF signal can be directly converted into baseband where basic signal processing operations are carried out.

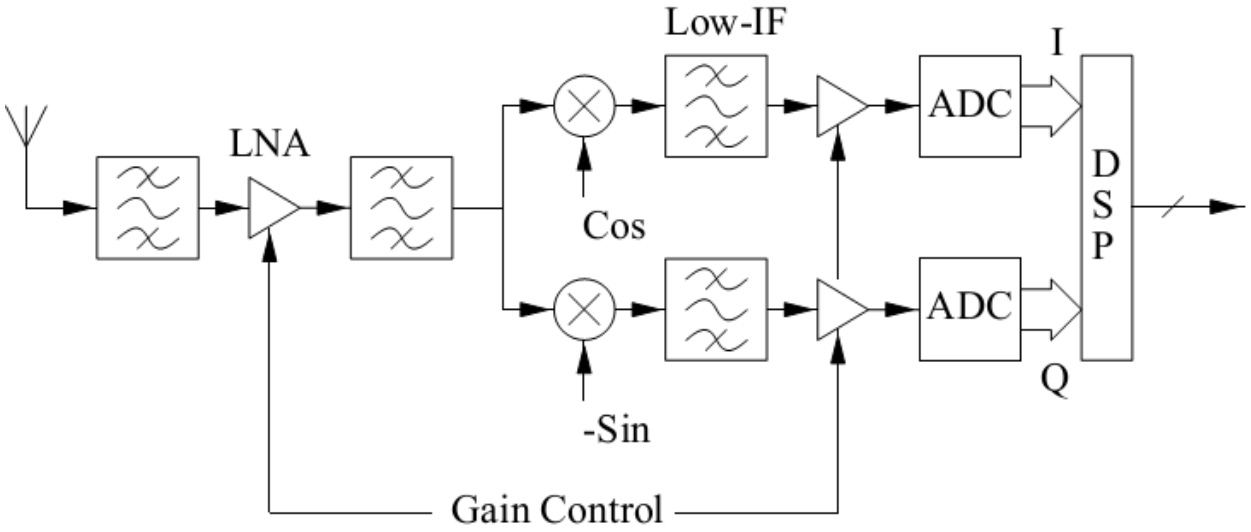


Figure 2.5. Typical low-IF receiver.

2.3.3 Direct-Conversion Receivers

A direct-conversion receiver (also called zero-IF) is a homodyne receiver that translates the desired RF signal directly to baseband where the initial information has been modulated (Figure 2.6). When translating directly from RF to baseband, a DC component is obtained at the output of the mixer. The DC component (also called DC offset) should be removed to avoid desensitization of the baseband modulator.

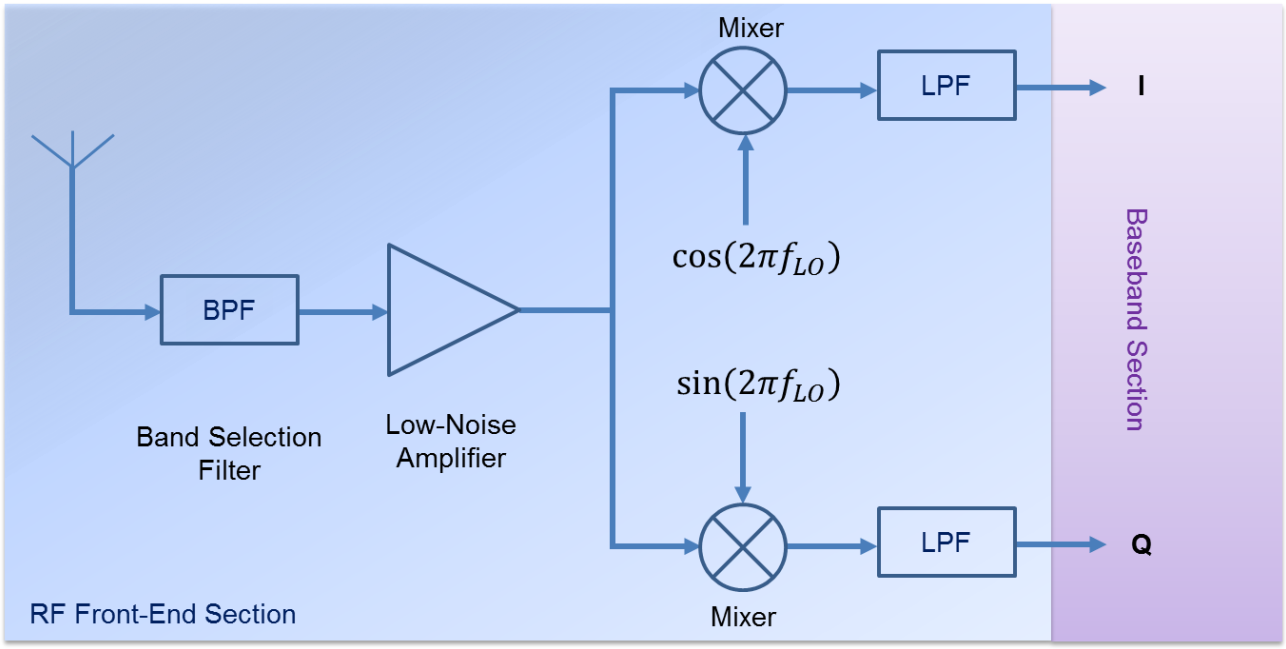


Figure 2.6. The architecture of a typical homodyne direct-conversion receiver.[7]

An extensive presentation of direct-conversion receivers is presented in reference 8.

2.3.4 Sub-Sampling Receivers

A sub-sampling receiver consists of a homodyne architecture that performs an IF downconversion followed by a sub-sampling operation that facilitates the use of high-speed digital signal processing (DSP). The architecture uses two digital mixers that prevent additional noise and distortion. The downconversion does not cause DC offset and $1/f$ noise. The digitization of the carrier frequency allows easy integration in CMOS processes. Figure 2.7 shows a general scheme of a sub-sampling receiver architecture.

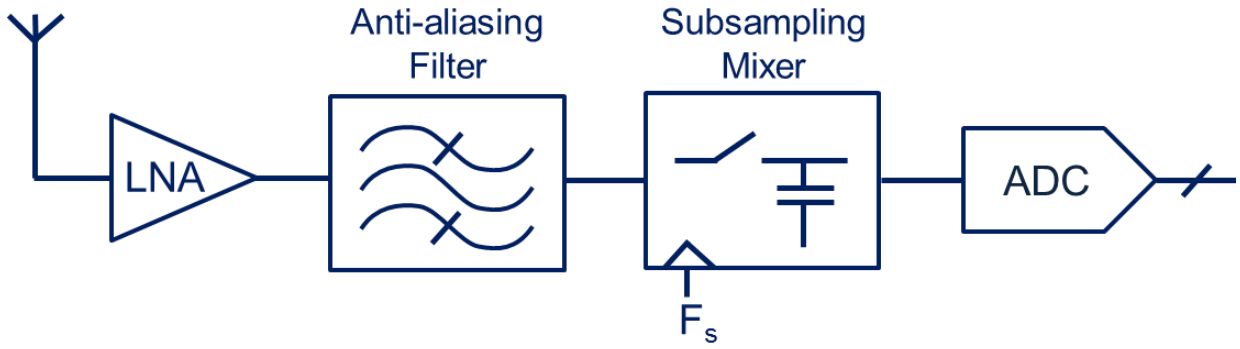


Figure 2.7. Typical sub-sampling receiver architecture: F_S is the sampling frequency and equals at least twice the carrier frequency.[9]

2.3.5 Receiver Architecture Benchmark

Table 2.1 depicted from reference 10 presents a benchmark of the most common receiver architectures.

Parameter	Heterodyne	Direct Conversion	Sub-Sampler	Low-IF
Selectivity	High	High	High	High
Analog requirements	High	Moderate	Moderate	Low
Flexibility	Low	Low	High	High
CMOS compatibility	Low	Moderate	Moderate	High
Noise	Low	Moderate	High	Low
Dynamic range	High	High	High	High

Table 2.1. Benchmark of the common receiver architectures.

2.4 System-Level Considerations

When designing a radio communication system, the designer should consider various system-level considerations.

2.4.1 Figures-of-Merit

In communication systems design, figures-of-merit are mathematical criteria used to evaluate the performance of the system being built or some of its building blocks. Among these figure-of-merit, are:

- **Noise Figure (NF):** describes the amount of noise handled by the communication system (see Keysight's application note about Noise Figure Measurement[11]).
- **Signal-to-Noise Ratio (SNR)**: consists of the ratio of the useful signal to the noise floor being created in the communication system.

- **Adjacent Channel Power Ratio (ACPR)**: introduces the ratio between the total adjacent channel power (intermodulation signal) to the main channel's power (useful signal).
- **Gain**: introduces the ratio between the output and input power or amplitude in a given device.
- **Third-Order Intercept Points**: is a figure-of-merit for nonlinear devices such as amplifiers and mixers, as well as communication systems.
- **1-dB Compression Point**: characterizes the point where the gain has been reduced by 1 dB.

Some devices, such as amplifiers, have their own figures-of-merit (e.g., efficiency, output dynamic range and stability.) In this next section, we present some system-level figures-of-merit and explain how they can be cascaded along with a communication chain.

2.4.2 Range Equations

A communication link includes the entire communication path from the information source to all the encoding and modulations steps, through the transmitter and the channel, and up to and including the receiver.[12] To determine the link budget required for safely communicating with a system and without degrading the link performance, many parameters have to be considered. Among these parameters, the range equation relates power received to the distance between the transmitter and receiver.

The development of the fundamental power relationship between the receiver and the transmitter usually begins with the assumption of an omnidirectional RF source, transmitting uniformly over 4π steradians. The power density at distance d is given by $p(d)$ and the power extracted with the receiving antenna is given by P_r, where A_{er} is the absorption cross section (effective area) of the receiving antenna.[12] Figure 2.8 shows an illustration of the range equation concept.

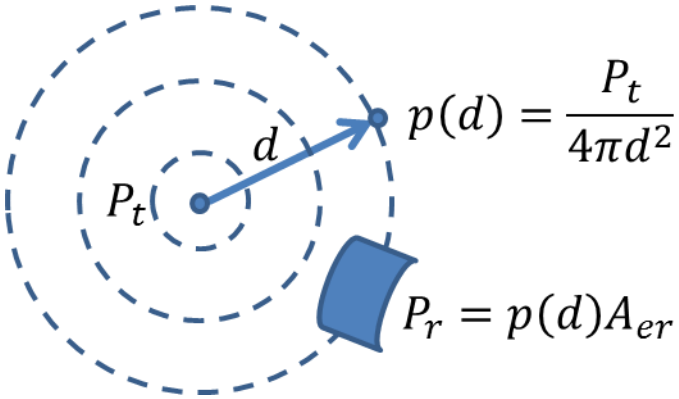


Figure 2.8. Range equation expresses received power in terms of distance.[10]

2.4.3 Link Budget

A radio communication system is usually composed of a transmitter, a receiver and a transmission channel (e.g., free space for wireless communication). When there is a direct path between the transmitter and the receiver, this path is designated as a line-of-sight (LOS) link (Figure 2.9).

The wireless link budget analysis looks for answers to the questions: How far can the communication

link go? What will be the throughput? In fact, several factors can impact the performance of the radio system. These factors include output power, antenna gains, receiver sensitivity, and environmental conditions (e.g., snow, rain and terrain.), just to name a few. The link budget consists of the calculation of three main components (Equation E.2.1):

$$\text{Received Power (dBm)} = \text{Transmitted Power (dBm)} + \text{Gains (dB)} - \text{Losses (dB)} \qquad \text{E.2.1}$$

Gains consist mainly of the antenna and amplifier gains, while losses consist of free-space path loss, antenna misalignment loss, environment-related losses (e.g., rain and snow) and gaseous absorption (e.g., clouds and fog).

In LOS links, free-space path loss (FSPL) is the major factor attenuating the emitted signal. It is calculated using equation E.2.2:

$$\text{FSPL(dB)} = 10 \cdot \log\left[\frac{4\pi df}{c}\right]^2 \qquad \text{E.2.2}$$

where d is the distance between the transmitter and the receiver (in meters), f is the transmitted signal frequency (in Hz) and c is the speed of light in a vacuum ($\approx 3 \cdot 10^8$ m/s).

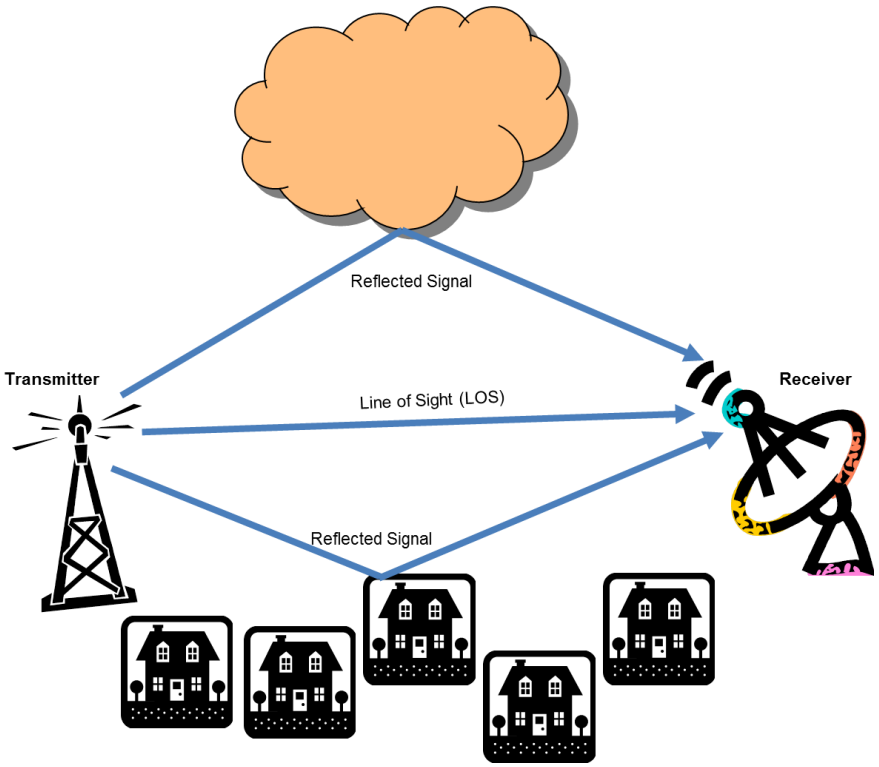


Figure 2.9. Typical communication link.

A LOS link is often subject to multipath and fading phenomena. Multipath occurs when a signal travels along different paths and causes interference at the receiver. In some cases, the travelling signals reach the receiver completely out of phase and cancel each other. Fading is a random deviation of the attenuation affecting the transmitted signal. It may be the result of multipath propagation. Some fading models, such as Nakagami and Rayleigh, were developed to estimate the fading margin required to reduce the impact of multipath. The signal attenuation can be greater than 30 dB due to multipath induced fading. Thus, a significant fading margin has to be considered in order to overcome the resulting loss (Equation E.2.3).

$$\text{Link Margin} = \text{Received Power} - \text{Receive Sensitivity} \qquad \text{E.2.3}$$

The throughput that can be achieved on a given transmission channel is generally related to the bit-error ratio (BER) that can be achieved at the receiver. BER is closely tied to the signal-to-noise ratio (SNR) determined by the modulation technique (Equation E.2.4).

$$\text{SNR} = \text{Received Power} - \text{Channel Noise} \qquad \text{E.2.4}$$

For in-depth insight, references 13 and 14 present some comprehensive examples regarding how to calculate a typical link budget considering the previous considerations.

2.4.4 Sensitivity and Selectivity

Sensitivity and selectivity are among the key specifications of any radio receiver because they define the receiver's ability to pick up the required level of the desired signals.

2.4.4.1 Sensitivity:

Common receiver sensitivity specification methods assume that the limiting factor of the sensitivity is the noise levels present inside and outside the receiver and not the available level of amplification that could be applied to the signal. For this reason, many figures-of-merits have been defined to specify the receiver's sensitivity. Among these parameters, we cite the following:

- SNR
- SNR and Distortion Ratio (SINAD)
- Noise Factor
- Noise Figure
- Carrier-to-Noise Ratio (CNR)
- Minimum Discernible Signal (MDS)
- BER

Generally, sensitivity is defined as the signal level required for a particular quality of received information. Practically, it consists of the power level needed to achieve a required SNR.

2.4.4.2 Selectivity:

Selectivity is an important property required in all radio receivers to ensure that only desired signals are

received. Depending on the receiver's architecture, selectivity covers the selection of a single band or many reception channels. The selected bandwidth can also be constant or variable, and narrowband or broadband. Selectivity is mainly achieved using RF and IF filters.

The transmission types the receiver will be used for determines the required bandwidth, the subsequent selectivity and the type of filters to be used. For instance, selectivity specifies how much out-of-band signals will be attenuated but does nothing to eventual in-band interference.

2.4.5 Image Rejection

In a heterodyne receiver architecture, the RF signal is translated to a lower non-zero IF one where signal processing is performed. As shown in Figure 2.10, there is a problem called image inherent that affects this type of architecture. It occurs when the useful signal is mixed with another signal located at the opposite side from the LO frequency and at an equal distance from it. The image frequency may cause serious distortion of the desired signal, particularly if its power is greater than the desired one.

To resolve the image signal problem, an image-reject filter can be added in the reception chain, before the mixing stage, to attenuate the image signal. The use of specific mixer structures such as Hartley and Weaver can also resolve the image frequency problem.

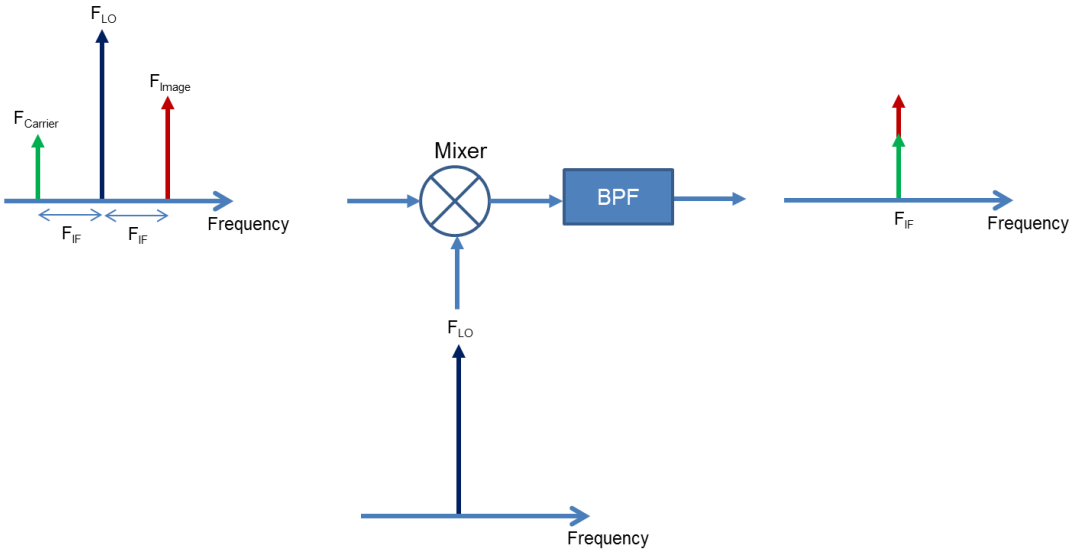


Figure 2.10. Operation of the mixer in a typical heterodyne receiver.

2.4.6 Phase Noise

Phase noise describes the random frequency fluctuations of the local oscillator. When these fluctuations are injected in the mixer, noise sidebands are created on both sides of the carrier frequency (Figure 2.7a). Noise sidebands are generally specified in terms of dBc/Hz at a given frequency from the carrier frequency (Figure 2.7b). Phase noise is important to control because of the spurs it may generate in the passband. It is also a major concern in Orthogonal Frequency Division Multiplexing (OFDM) systems where it can destroy the signal orthogonality. Refer to reference 15 for in-depth insight into phase noise.

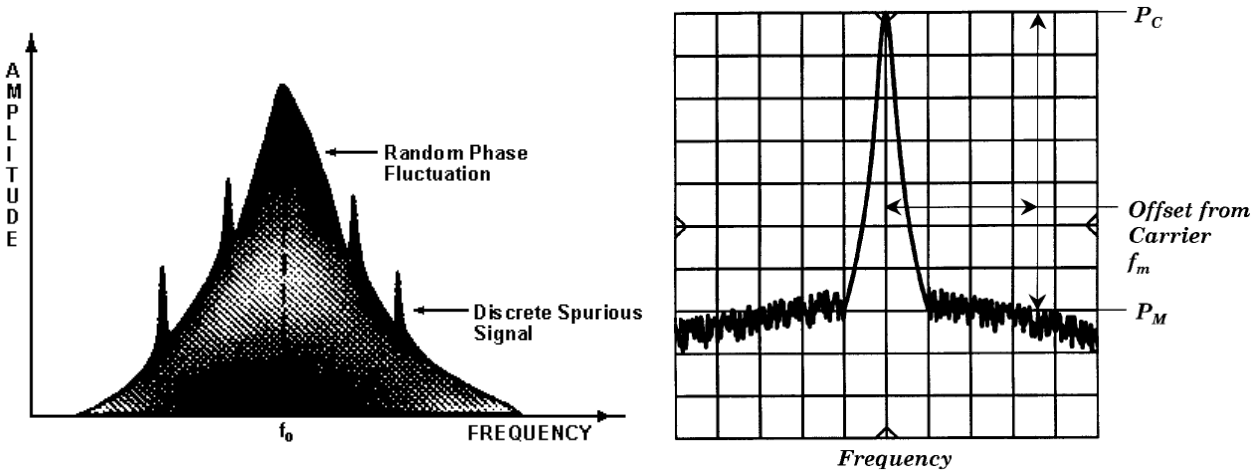


Figure 2.11. Phase noise: (a) Spectrum analyzer display of phase noise (b) Double-sideband phase noise plot around the carrier frequency.

2.4.7 DC Offset

The DC offset effect manifests itself mainly in direct-conversion architectures where an undesired strong signal may be downconverted along with the desired signal directly to baseband. This is called "self-mixing." In this case, the strong level of the undesired downconverted signal causes severe interference with the desired signal.

Three main self-mixing mechanisms can drive the appearance of the DC offset effect[7]:

1. **LO leakage.** The poor isolation between the LO and the LNA, particularly on silicon-based substrates, can lead to the leakage of the LO signal through the RF port and its self-mixing with the main LO signal. The situation becomes even worse if the leaking LO signal is amplified in the LNA before reaching the mixer RF input.
2. **LO re-radiation**. In some cases, the LO signal can be reradiated by the antenna and reflect off obstructions. The reradiated signal is selected among the wanted signals and mixes itself with them. In this case, the reradiated signal will not only bother the local user but also other users capturing the same channels.
3. **Strong in-band interferer**. A strong nearby interferer can also generate significant DC-offset that can alter the received information.

The DC-offset can also result from second-order in-band nonlinearities generated by the receiver RF front-end.

2.4.8 Nonlinear Behavior of RF Systems

A nonlinear RF system generates an output signal that is not a linear combination of input signals (Figure 2.12). Let's assume that: $V_{in} = A\cos(\omega_1 t) + A\cos(\omega_2 t)$

Figure 2.12. Nonlinear system schema.

The resulting output signal is given by Equation E.2.5:

$$V_{out} = aA(\cos(\omega_1 t) + \cos(\omega_2 t)) - bA^2(\cos(\omega_1 t) + \cos(\omega_2 t))^2 - bA^3(\cos(\omega_1 t) + \cos(\omega_2 t))^3 + \cdots \quad \text{E.2.5}$$

2.4.9 Intermodulation Distortion

Intermodulation distortion (IMD) is an effect present in nonlinear communication systems. IMD occurs when two or more signals occupy the same transmission channels. At the output of the mixer, a nonlinear response manifests itself as bi-products of the fundamental frequency called intermodulation distortion. The undesired signals show up in the receive channels as blockers.

Figure 2.13 shows the first-, second- and third-order intermodulation products for two-tone carriers. The frequencies of the two-tone intermodulation products can be computed with Equation E.2.6:

$$mf_2 \pm nf_1 \text{ where } m, n = 0, 1, 2, 3, \ldots \quad \text{E.2.6}$$

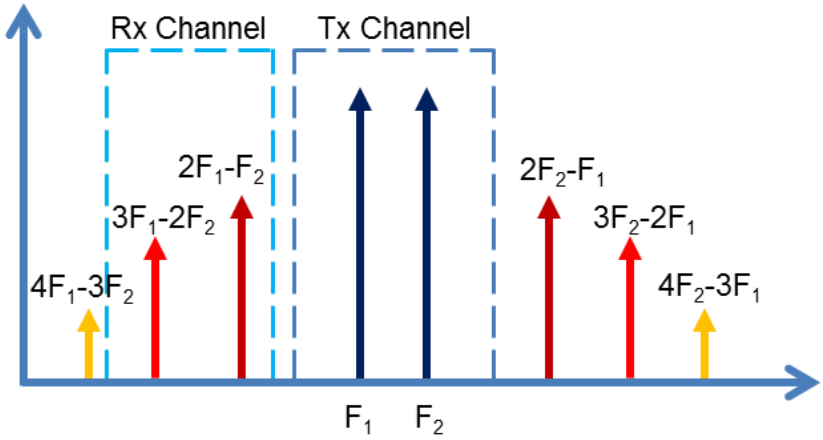


Figure 2.13. Intermodulation products.

2.4.10 Noise Figure

Noise figure is a measure of the degradation of the SNR caused by each device in the transmission and/or reception chain. It can be computed directly from the noise factor F using Equation E.2.7.

The noise factor is the ratio of the input SNR to the output SNR. The noise ratio is the difference between input and output SNR as given in Equation E.2.8.

$$F = \frac{SNR_{in}}{SNR_{out}} \quad \text{E.2.7}$$

$$NF = 10\log(F) = SNR_{input,dB} - SNR_{output,dB} \quad \text{E.2.8}$$

Each RF device is characterized by its noise factor since its electronic structure adds noise power to the useful signal, and it may have its own gain (Figure 2.14). When various components are cascaded, the resulting noise factor is calculated using Friis formula illustrated in Equation E.2.9. See reference 11 for more information.

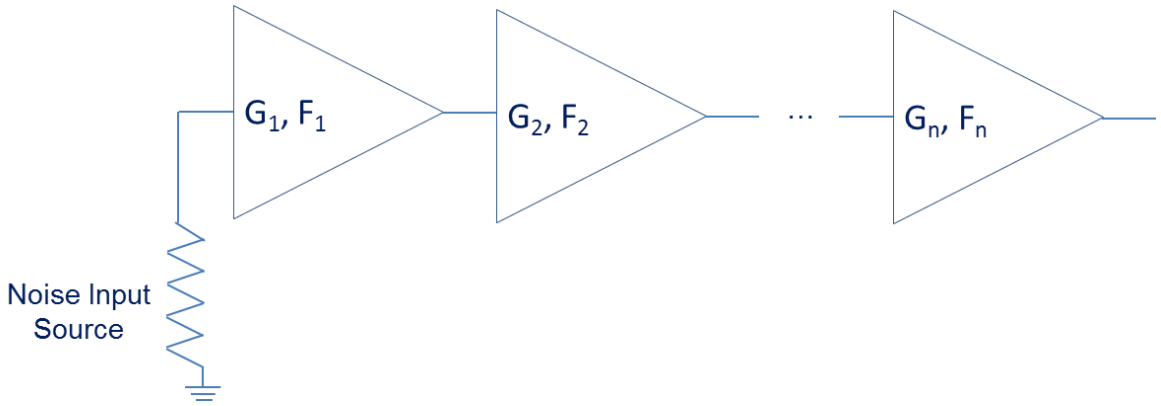


Figure 2.14. Cascading noise figure in an RF chain.

$$F = F_1 + \frac{F_2 - 1}{G_1} + \frac{F_3 - 1}{G_1 G_2} + \frac{F_4 - 1}{G_1 G_2 G_3} + \cdots + \frac{F_n - 1}{G_1 G_2 G_3 \ldots G_{n-1}} \quad \text{E.2.9}$$

2.4.11 In-Band and Out-of-Band Interferers and Blockers

Every transmitter causes some interference in each receiver. This interference has to be controlled in order to prevent the desensitization of receivers. Two main categories of interferers, as cited in reference 16, are:

- **In-Band Interferers**: are caused by the overlap in frequency between the transmitter and the receiver bandwidths. Both devices can operate properly, but only if they are geographically separated so that the interfering transmitter signals are attenuated enough and do not overcome the desired signals.
- **Out-of-Band Interferers**: this type of interference does not exhibit any overlap between the transmitter and receiver bandwidths. Interference is caused by: (1) in-band emissions

received from out-of-band due to the receiver filtering and attenuation imperfections, and (2) out-of-band emissions received in-band due to transmitter impairments and nonlinearities.

2.4.12 Adjacent Channels

Adjacent channels are the immediate frequency ranges located at the upper and lower bounds of the desired channel (Figure 2.15). Spectral regrowth, as well as harmonic and intermodulation distortions, can impact these channels and bother the transmissions of other users. As a result, particular attention must be given to these side effects.

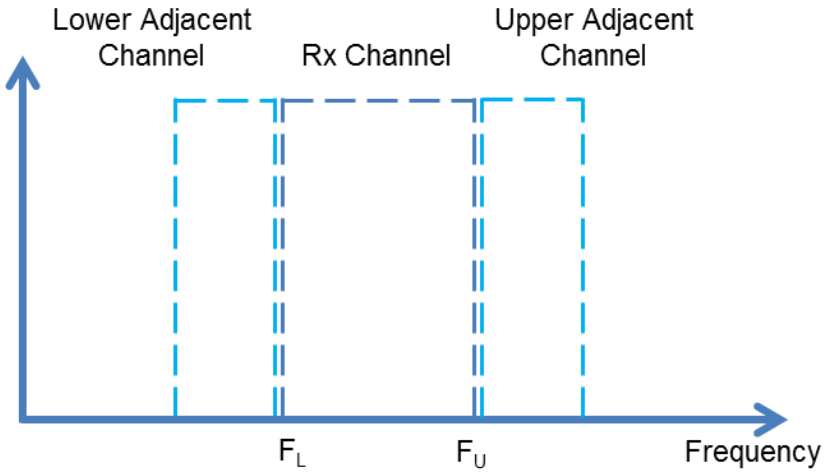
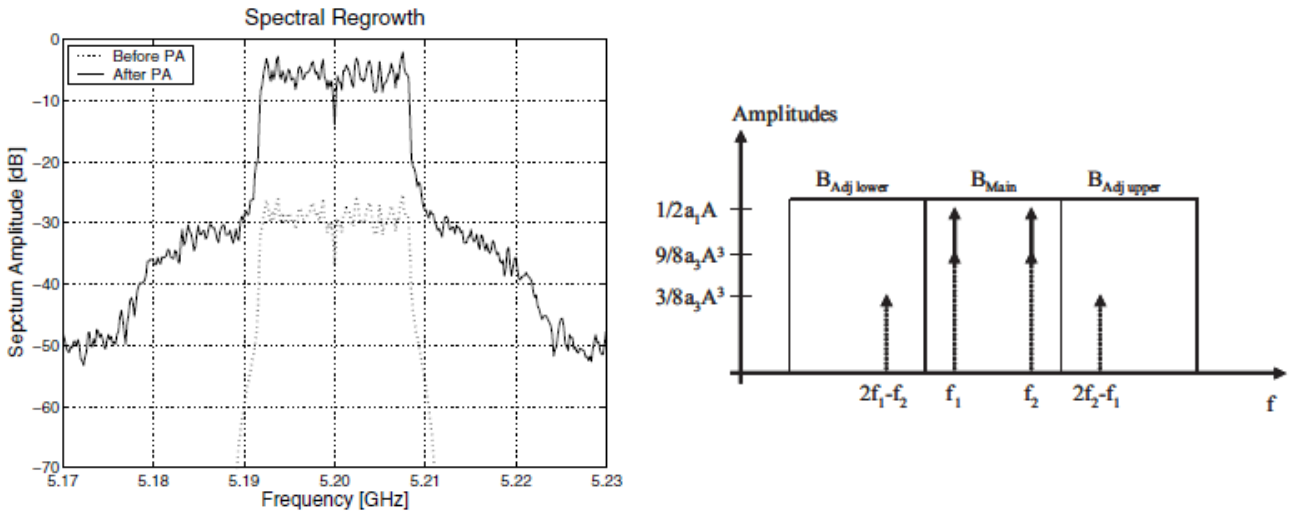


Figure 2.15. Upper and Lower Immediate adjacent Channels.

Adjacent Channel Power Ratio (ACPR) is a measurement of the amount of interference, or power, in the adjacent frequency channel. ACPR is usually defined as the ratio of the average power in the adjacent frequency channel (or offset) to the average power in the transmitted frequency channel. The cascaded ACPR can be calculated according to the following formula[17]:

$$ACPR_{output} = 10\log\left[\left(\sum_{k=1}^{n}\left[10^{\frac{ACPR_k}{10}}\right]^{\frac{1}{2}}\right)^2 + 10^{\frac{ACPR_{input}}{10}}\right] \qquad \text{E.2.10}$$

Figure 2.16 (a) and (b) illustrate the concept of ACPR.



(a) (b)

Figure 2.16. (a) Spectral Regrowth due to power amplification of an OFDM signal (b) Illustration of in-band and out-band intermodulation: ACPR is the difference between the main in-band tone power and adjacent channel intermods.[17]

2.4.13 Gain Compression

Gain compression is a phenomenon that exhibits itself in nonlinear RF systems. In a linear system, any change of input power causes a linear change in output power. However, in nonlinear systems, the output power changes linearly until saturation happens. At this point, the output power is no longer a linear function of input power.

2.4.13.1 Gain:

The gain is the ratio of output to input power or amplitude. It is commonly designated by G and measured in decibels.

$$G_{dB} = 10\log\left(\frac{P_{out}}{P_{in}}\right) \qquad \text{E.2.11}$$

The cascaded gain is a linear sum of the gain obtained at each stage:

$$G_{tot,dB} = \sum_{i=1}^{n} G_{i,dB} \qquad \text{E.2.12}$$

2.4.13.2 Third-Order Intercept Point:

Commonly designated by $IP3$ or TOI, the third-order intercept point is a characterization of nonlinear devices and systems. It relates nonlinear products caused by third-order nonlinear terms to the linearly amplified signal. The second-order intercept point ($IP2$) is similar to $IP3$, but considers the second-order terms. The cascaded $IP3$ can be calculated using equation E.2.13 picked from reference 17:

$$\frac{1}{IP3_{tot}} = \frac{1}{IP3_n} + \frac{1}{G_n IP3_{n-1}} + \frac{1}{G_{n-1}IP3_{n-2}} + \cdots + \frac{1}{G_n} + \cdots + G_2 IP3_1 \qquad \text{E.2.13}$$

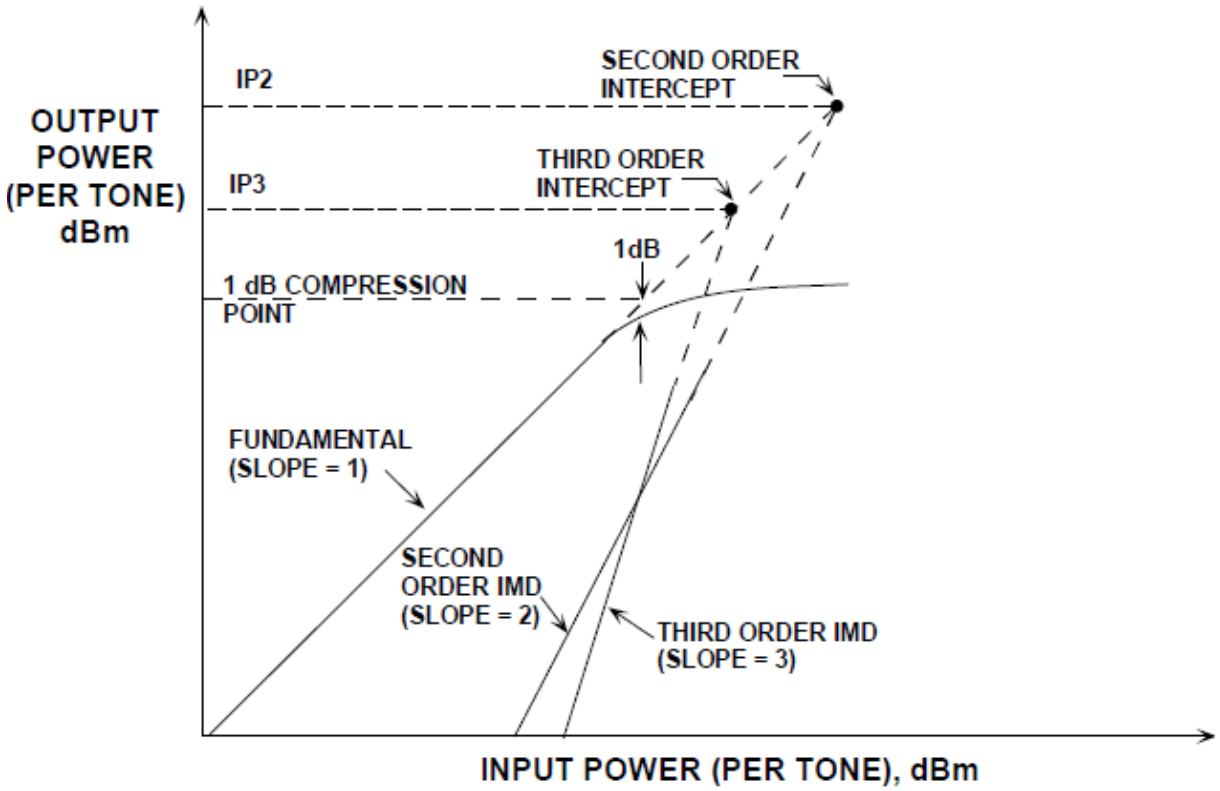


Figure 2.17. Third-Order Intercept Point.

2.4.13.3 1-dB Compression Point

The 1-dB compression point commonly denoted as P_{1dB} characterizes, in a nonlinear system, the input power at which the small signal gain departs from a linear curve and compresses by 1 dB. Figure 2.17 illustrates how to figure out the 1-dB compression point for a nonlinear system. The P_{1dB} can be calculated recursively for a cascade of devices as shown in equation E.2.14.

$$P_{1dB,output} = \frac{1}{10\log\left[\frac{1}{G_n P_{1dB,n-1}} + \frac{1}{P_{1dB,n}}\right]} \qquad \text{E.2.14}$$

2.4.14 Dynamic Range

Dynamic range commonly designated DR, is the useful range of the signal that the receiver can recognize and process properly. Generally, it is defined as the difference between the 1-dB compression point and the noise floor, all determined in a given stage (E.2.15).

$$DR = P1dB - MDS \qquad \text{E.2.15}$$

2.5 Transceiver Design Trade-Offs

2.5.1 Signal-to-Noise Ratio

As given in Equation E.2.16 and illustrated in Figure 2.18, Signal-to-Noise Ratio (commonly noted as SNR or S/N) measures the level of the useful signal to the noise floor level. It represents the ratio between the integrated signal power and the integrated noise power across the bandwidth of interest.

$$SNR_{dB} = 10\log\left[\frac{P_{signal}}{P_{noise}}\right] = P_{signal,dB} - P_{noise,dB} \qquad \text{E.2.16}$$

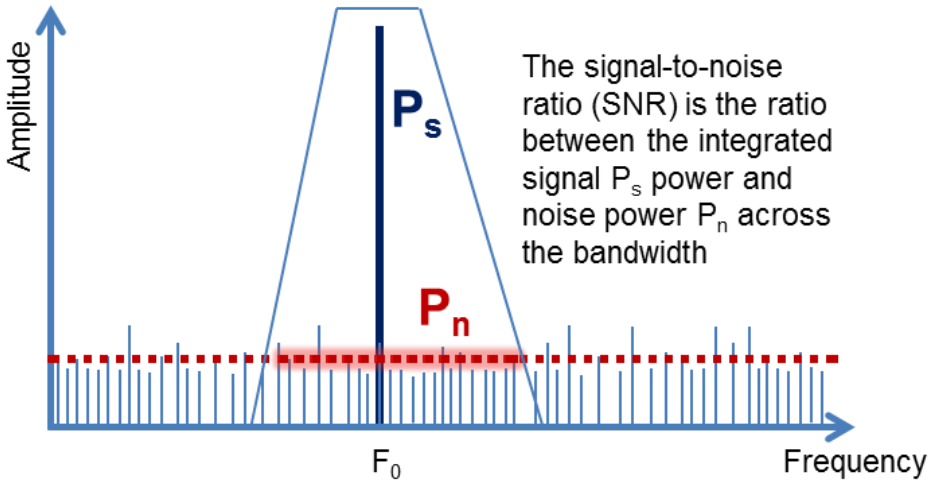


Figure 2.18. Signal-to-Noise Ratio.

2.5.2 Frequency Planning

In radio communications, the transmitter uses a channel called uplink to transmit signals and the receiver uses another channel called downlink (or forward link). The uplink and downlink bandwidths are equal (Figure 2.19). To ensure communication, a couple of uplink and downlink channels must be used. For every communication standard, uplink and downlink bands, the channel spacing, and the band separation are standardized. For instance, cellular communication lies on the 824 – 849 MHz and 869 – 894 MHz bands for uplink and downlink, respectively, with a band separation of 20 MHz and a channel spacing of 30 KHz for CDMA.[17]

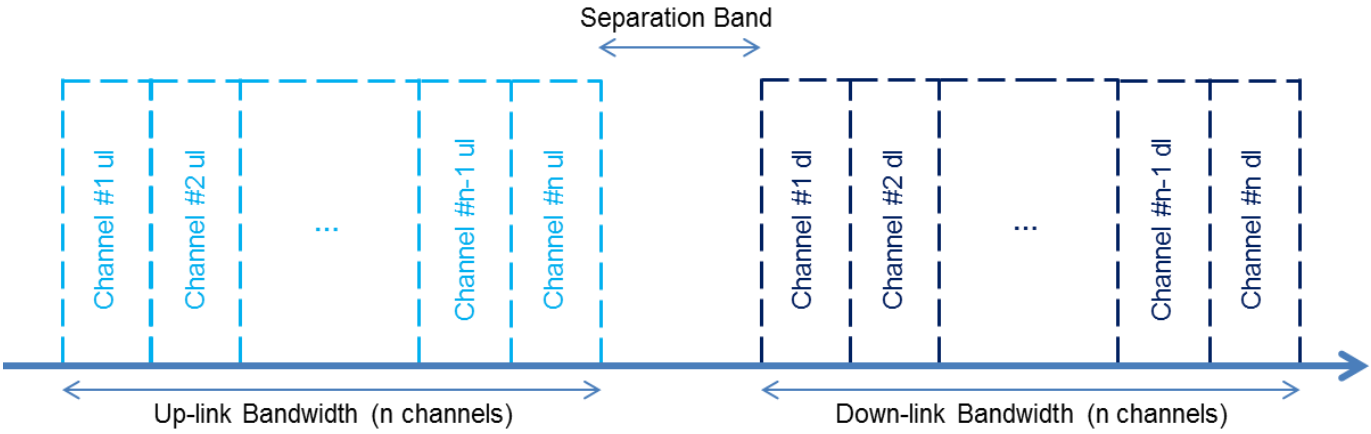


Figure 2.19. Uplinks and downlinks with channelization and band separation.

In RF design, frequency allocation has a major influence on the frequency planning, which consists of selecting the intermediate frequencies that could minimize the spurious response in super-heterodyne transceivers. This problem is more tedious if the transceiver is full-duplex where the transmitter and receiver signals can generate harmonics and reversely contaminate each RF block.

To minimize the impairments, the selection of an intermediate frequency (IF) should follow two basis rules[17]:

- **Rule 1**: If the receiver and transmitter share the IF local oscillator, for better performance IF_{RX} and IF_{TX} should be selected as given in Equation E2.17:

$$\begin{cases} IF_{RX} = IF_{RX} + \Delta F_{RX-TX} \\ \text{where } F_{LO} > F_{RX} > F_{TX} \\ \text{or } F_{TX} > F_{RX} > F_{RX} \end{cases} \quad \text{E2.17}$$

- **Rule 2**: To prevent potential receiver in-band jamming, IF_{RX} should respect Equation E2.18:

$$\begin{cases} IF_{RX} > B_{TX} + B_S + B_{RX} \\ \quad or\ IF_{RX} < B_S \end{cases} \quad \text{E2.18}$$

To validate a frequency plan, one should carry out the spurious analysis to measure the harmonics lying in the bands of interest. This calculation is straightforward for fundamental signals:

$$f_S = m \cdot f_A \pm n \cdot f_B \quad \text{E2.19}$$

where A and B are the fundamental signals.

2.6 Basic RF Transceiver Building Blocks

As presented above, all transmitter and receiver front-ends are composed of various RF building blocks that carry out basic analog functions like frequency translation, frequency selection and power amplification. In this section, we present the most important RF building blocks. For further information, refer to references 19 and 20.

2.6.1 Antenna

An antenna provides an interface between the communication channel (e.g., free space) and the receiver front-end. A transmitter requires a matching circuit at the output of the power amplifier to allow the radiation of maximum power into the communication channel. A receiver requires a bandwidth selection filter and a LNA to carefully select the right channel and get the signal at a detectable level. The most important antenna performance properties are: gain, directivity, radiation loss, resistive loss, bandwidth, and radiation efficiency. For more information about antennas, refer to references 21, 22 and 23.

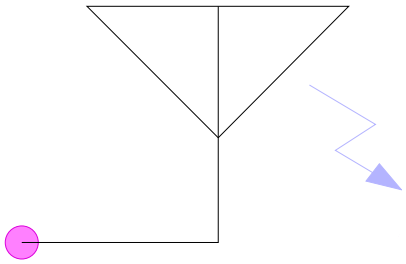


Figure 2.20. Antenna symbol.

2.6.2 Amplifier (Power/Low-Noise)

An amplifier is a device that elevates the power level of a given signal. In a transmitter, it is called power amplifier because it is required to strengthen the signal at the maximum allowed linear output. However, in a receiver, it is required to bring the input signal to an acceptable level for processing with minimal noise to avoid degrading the quality of modulated data (EVM). Amplifiers are nonlinear devices. Thus, a strong focus on their nonlinear behavior is required. Among their common performance parameters, are: gain, noise figure, gain compression, and intermodulation and harmonic distortion. Further analysis of RF amplifiers is available in references 24 and 25.

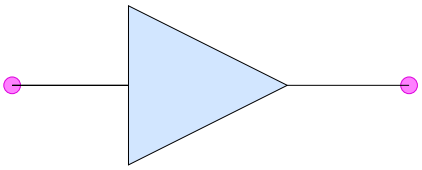


Figure 2.21. Amplifier symbol.

2.6.3 Filter

RF filters are devices that carry out the frequency selection by only passing the desired range of frequencies. The unwanted signals are strongly attenuated. An ideal filter passes the desired signals with no losses and perfectly extinguishes undesired ones. But most commercial filters introduce small losses in the passband (i.e., the range of desired frequencies) and apply enough attenuation at the stopband (i.e., the range of undesired frequencies). The most outstanding performance parameters of a filter are the passband and stopband attenuation, as well as the bandwidth and group delay. Depending on the application, various types of filters are used. Low pass and high pass filters pass a range of frequencies below or above a given cutoff frequency. A bandpass filter passes only a range of frequencies between

two given frequencies. A bandstop filter exhibits the reverse operation of a bandpass one.

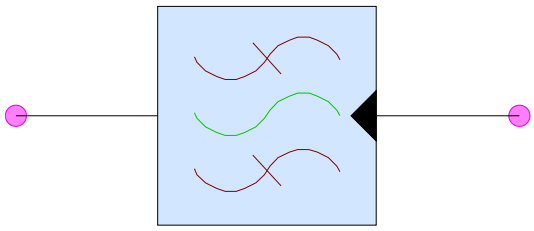


Figure 2.22. Filter symbol.

Filters are a topic widely treated in RF literature. Refer to references 26 and 27 for more information.

2.6.4 Duplexer

A duplexer is a filter bank required only in FDMA systems with a single antenna configuration where both the receiver and the transmitter operate simultaneously. To avoid receiver desensitization and leakage from receiver to the antenna output, a duplexer is required to perfectly isolate the reception stage from the transmission one with a minimum of loss in the wanted bandwidth. The insertion loss is an important parameter in a duplexer. Refer to reference 28 for more information.

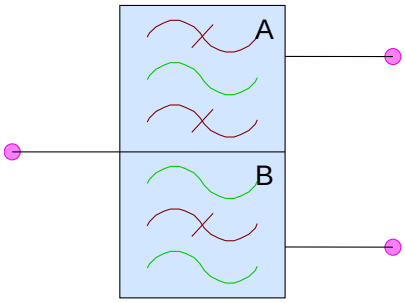


Figure 2.23. Duplexer symbol.

2.6.5 Mixer

A mixer is a nonlinear device that allows the translation of frequencies. It is used to bring either the modulated signal to the RF carrier or translate the RF signal to IF. When mixing two signals, a mixer generates both higher and lower translations (i.e., the sum and difference of input signals). A low pass or a high pass filter is required to select the desired frequency product. Generally, a LO is required as input to allow the signal down- or up-conversion.

Various technologies are used to build mixers: A mixer can be based on diodes or transistors. In both cases, the device is nonlinear. Its nonlinear behavior has to be monitored (e.g., its 1dB-compression point and intermodulation distortion, etc.). Refer to references 20 and 29 for more information about mixers.

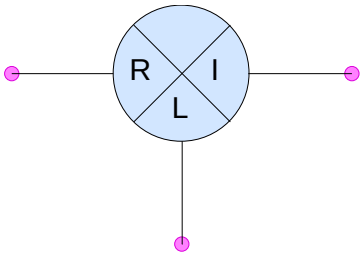


Figure 2.24. Mixer symbol.

2.6.6 Local Oscillator

A local oscillator is a device that generates a reference signal to be used as one of the mixer inputs and serves for up- and down-conversion. The LO output is a large signal that drives the mixer and causes the generation of both fundamental frequency and various harmonics.

An LO is generally characterized by various parameters such as phase noise, tuning range and frequency stability. Refer to references 30 and 31 for more information.

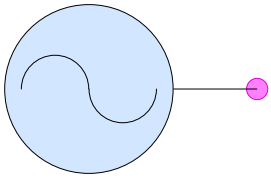


Figure 2.25. Oscillator symbol.

2.6.7 Detector

A detector is a device that recovers the signal shape. Depending on the modulation type, it can be used to recover the signal phase or the amplitude. Refer to references 32 and 33 for more information.

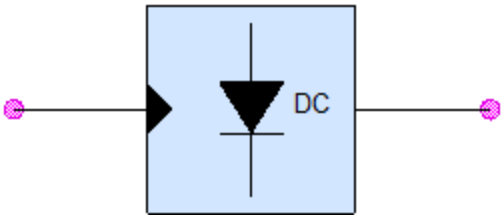


Figure 2.26. Detector symbol.

2.6.8 Analog-to-Digital Conversion

An analog-to-digital converter (ADC) is a master component in modern wireless radios. It transforms an analog signal into digital words that can be used in baseband signal processing. A typical ADC is composed mainly of four building blocks:

- **Prefilter**: an anti-aliasing filter that is used to prevent the aliasing of high-frequency signals.

- **A sample-and-hold circuit**: which maintains a constant input analog signal to the ADC during the time the signal is converted to an equivalent output digital code. This period is called conversion time.
- **Quantizer**: which segments the reference into subranges. Generally, there are 2^N subranges where N is the number of bits of the output code. The quantization consists of finding the subrange that corresponds to the analog input sample.
- **Encoder**: depending on the quantization result, the digital preprocessor encodes the corresponding digital bits.

For in-depth information about ADCs, refer to references 34 and 35.

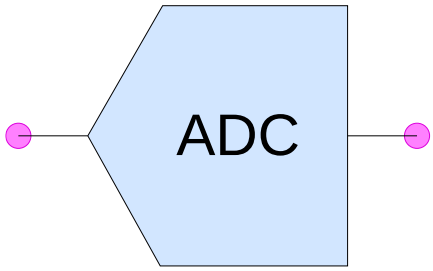


Figure 2.27. Analog-to-Digital Converter symbol

2.7 Summary

In this chapter, we summarized some basics of RF design and in particular, detailed the system-level considerations with which RF designers should be familiar. This chapter was not intended to give an in-depth course in RF design; however, the reader can refer to the literature references given in this chapter to learn more about the notions and concepts presented. Finally, Keysight's application note 1312[36] presents a practical view of most of the concepts presented in this chapter.

Chapter 3: Designing RF Systems Using PathWave System Design (SystemVue)

Case Study: Long-Term Evolution (LTE) Front-End Design

3.1 Introduction

In Chapter 1 we highlighted some features and capabilities of PathWave System Design (SystemVue). In Chapter 2 we presented an overview of RF basics. In this chapter, we'll first highlight the common RF approaches, as well as the system-level design paradigm, and then present a detailed case study using the capabilities of PathWave System Design (SystemVue) to design and analyze a tri-band LTE scanning receiver.

3.2 Highlights about Design Methodologies and Approaches

A wireless system is generally composed of two main blocks: a baseband stage that is in charge of carrying out the basic signal processing operations and an RF front-end that provides the air-interface facilitating communication with other systems using radio waves (e.g., other wireless radios and base stations). The design process of a typical wireless system includes the analysis, implementation, integration, and verification of both baseband and RF subsystems (Figure 3.1).

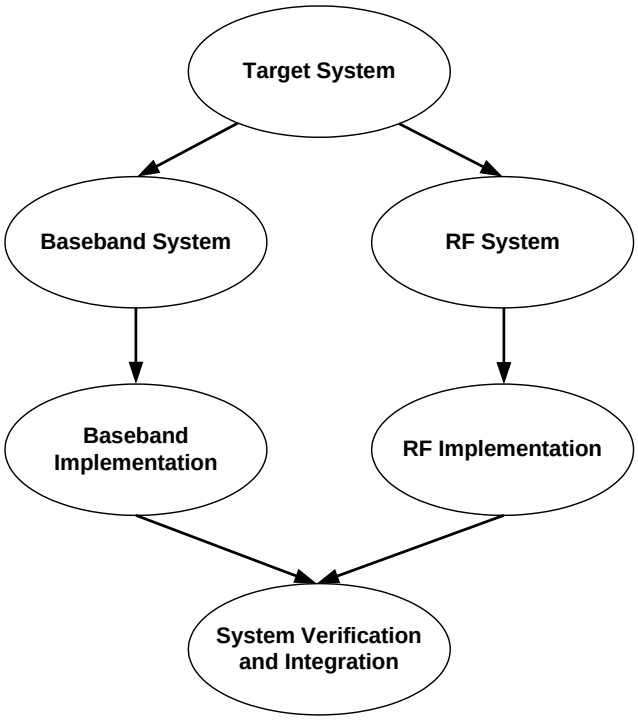


Figure 3.1. The design process of a typical communication system is generally subdivided in two subsystem design processes: baseband and RF

The primary focus of this primer will be on the RF design process.

3.2.1 Common Design Approaches

Over the years, RF systems have been designed in many ways. Every methodology impacts the productivity, design reliability and verification process. The three most common approaches used when performing RF systems design are:

1. **Bottom-Up Design**: With this traditional design approach, the process starts by designing individual blocks. These blocks are then integrated into more complicated subsystems that will be part of the final system. This design approach may be very effective for small designs but can cause severe problems for large designs. At this level, the verification process becomes more difficult and fixing bugs may be very expensive.[37]. The higher-level simulations may be time-consuming. For example, simulating a single Power Amplifier (PA) may take seconds, but performing a BER analysis on the whole system might take an hour.
2. **Top-Down Design**: With this approach, the system is defined by a system-level block diagram. It is optimized and the requirements of its underneath blocks are derived. The individual blocks are then implemented and tested to be later integrated in the whole system. RF designers mostly use this approach when they begin the system design by exploring the gain, noise figure and distortion budget. Valuable trade-offs can take place from the beginning to meet the specifications and help the design be more effective. However, this approach may experience a discontinuity in the design flow resulting from the disparities between the system architecture and its implementation. This discontinuity slows down the design process.[37]
3. **Performance-/Constraint-Driven Design**: This approach consists of the alternation of a top-down flow used in design and a bottom-up flow used in verification. At each level, the system is subdivided into subsystems to be implemented. The assembled blocks are then verified against the specifications. This approach helps RF designers catch errors at the early stages of the design flow.[38]

This reminder of the common approaches used in RF design is important to help designers adjust their design practices according to their project requirements in terms of productivity, efficiency, and communication between the RF architect and the individual component designers.

3.2.2 Overview of System-Level Modeling

An RF system is commonly composed of subsystems or modules linked to each other to implement the system functionality. These modules are composed of low-level circuits and devices such as transistors, transmission lines and discrete components. Three abstraction levels are then generally defined for the description of an RF system (Figure 3.2)[39]:

- **System Level**: At this level, the RF system is considered a “black box” composed of a bunch of building blocks to ensure a given functionality (e.g., a low-IF receiver). The system is described by generic parameters such as cascaded noise figure and cascaded gain.

- **Module/Component Level**: A component (also called module) is a part of the system. It has a given functionality that helps build the system functionality. For example, a filter is a component intended to select a given range of frequencies. A receiver contains various filters in order to select the desirable band. The filter insertion loss is used to calculate the cascaded noise figure of the whole receiver as the noise contribution of that filter impacts the noise level in the receiver.
- **Circuit/Device Level**: At the device level, the physical aspects of a component are considered. A component is then considered an assembly of circuits and devices that interact between them to fulfill the component's functionality. For example, a typical amplifier is composed of transistors, transmission lines and discrete components (e.g., capacitors and resistors). Each device has its set of parameters such as operating current and voltage, and resistance. These parameters, when combined with other devices' parameters, and depending on the way those devices are mounted, define how they impact the component's parameters.

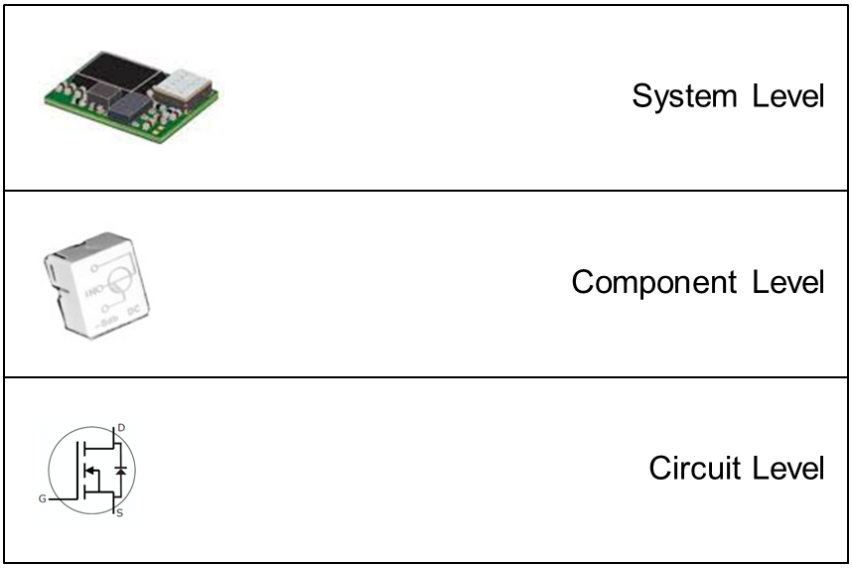


Figure 3.2. Common abstraction levels in RF systems.

At the system-level, components are typically represented as behavioral blocks, described in terms of parameters such as gain, noise figure and insertion loss. This model captures the system specifications and allows for an effective exploration of the design space by masking unnecessary parameters generally related to the physical implementation of the system.[40] This approach has two major advantages:

- It is suitable for a top-down design approach where the designer starts by selecting an "apparently" appropriate architecture for the intended system.
- It enables analysis of the selected architecture and decides if it fits the requirements. If not, either an optimization process may be performed, or a different architecture may be proposed.

PathWave System Design (SystemVue) is a simulation software that allows system-level modeling and analysis of both RF and baseband systems. To help designers architecting RF systems, PathWave System

Design (SystemVue) includes a set of models and tools that allow the system-level analysis, evaluation and optimization of RF architectures. The software provides powerful capabilities that help the designer evaluate the linear and nonlinear responses of the RF system. For instance, it allows the broadband noise analysis, intermodulation computations and IF selection to prevent in-band and out-of-band impairments. For more information, refer to Chapter 1, as well as the software's comprehensive help menu and tutorials.

3.3 Case Study: LTE Tri-Band Receiver Design

In the first sections of this chapter, we presented an overview of RF design approaches and system-level modeling. In this section, we present a case study that shows how an LTE receiver can be designed using PathWave System Design (SystemVue). The main purpose is to illustrate how system-level modeling can be useful for RF design and how PathWave System Design (SystemVue) helps the designer analyze and optimize real RF architectures.

3.4 LTE Overview

The third-generation (3G) Long-Term Evolution (LTE), established by the third-generation partnership project (3GPP), was proposed to provide cellular services to meet the increasing demand in data transmission speeds and lower latency over cellular communication networks. To achieve its high transmission rates and reliable communication, LTE features various technologies:

- **OFDM (Orthogonal Frequency Division Multiplex*):*** OFDM technology enables high data bandwidths and provides a high degree of resilience to reflections and interference. Two types of access methods are used: OFDMA for the downlink and SC-FDMA (Single Carrier Frequency Division Multiple Access) for the uplink.
- **MIMO (Multiple Input Multiple Output)**: MIMO was incorporated to deal with issues such as multipath and fading interference, as well as to improve throughput. Multiple antennas are used to differentiate the different signals coming from different paths.
- **SAE (System Architecture Evolution)**: LTE's high data rate and low latency requirements were achieved by moving several functions out of the network and to the periphery to improve the performance.

LTE also uses the traditional TDD (Time Division Duplex) and FDD (Frequency Division Duplex) access schemes.

The LTE air-interface specifications were documented by 3GPP in reference 41 for user equipment (UE) and base stations.[42] For more information about LTE, refer to references 43, 44, 45, and 46.

3.4.1 Receiver Specifications and Design

A low-cost scanning receiver for LTE TDD system applications was recently proposed in reference 47. The receiver designers used Keysight PathWave Advanced Design System (ADS) to analyze and simulate their architecture. The receiver supports three LTE bands (34, 39 and 40 in the E-UTRA operating

bands[41]) and uses two switches to select the desired band. However, the proposed receiver was only designed to operate from 2320 to 2370 MHz (in the third band) and does not cover the frequency range of the 40th band (i.e., from 2300 to 2400 MHz). This seems to be a trade-off guided by the selection of a 140-MHz standard intermediate frequency. In the following, we will consider the entire frequency range of the 40th band and use PathWave System Design (SystemVue) capabilities to illustrate the different system-level analyses.

3.4.1.1 Receiver Specifications

Table 3.1 summarizes the tri-band LTE receiver specifications.

Operating Bands	Band 1: 1880 – 1920 MHz (E-UTRA band #39)
	Band 2: 2010 – 2025 MHz (E-UTRA band #34)
	Band 3: 2300 – 2400 MHz (E-UTRA band #40)
Modulation	QPSK
Coding Rate	1/3
Channel Bandwidth	10 MHz
Antenna Configuration	Single Input Single Output (SISO)
Channel Type	Additive White Gaussian Noise (AWGN)
Input SNR	10 dB
Relative Throughput	≥ 95%
Number of Physical Resource Blocks (PRBs)	50
Duplexing Scheme	TDD

Table 3.1. A summary of the LTE receiver specifications.

3.4.1.2 RF Architecture Selection

We start designing the LTE receiver by selecting the RF architecture that will be used in its implementation. In this case study, we first consider a traditional super-heterodyne architecture with two intermediate frequencies. The first IF (namely IF1) will be implemented by a typical mixer in the receiver RF front-end, while the second one (namely IF2) will be included in the demodulator (Figure 3.3).

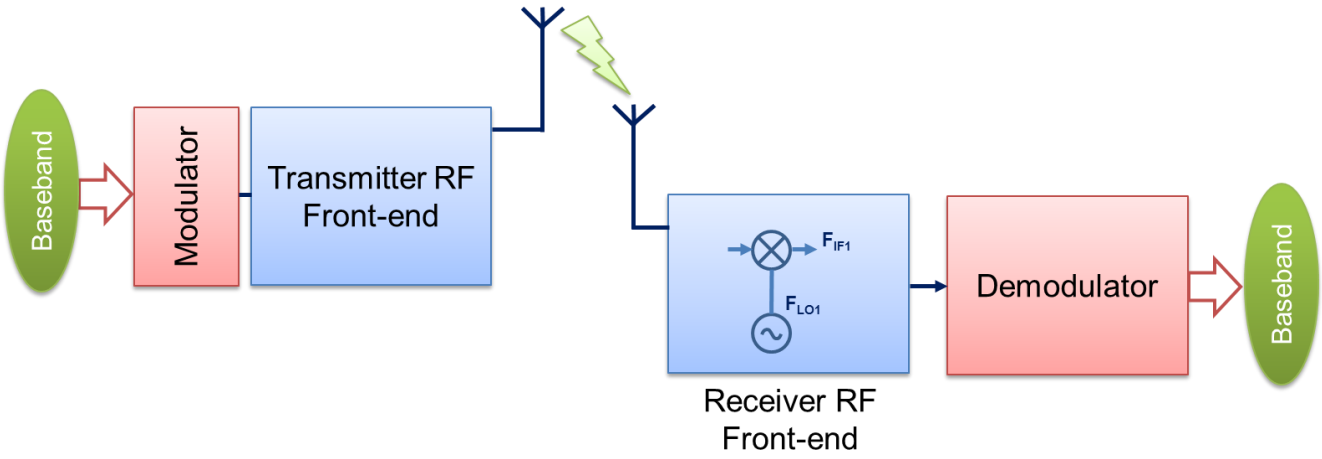
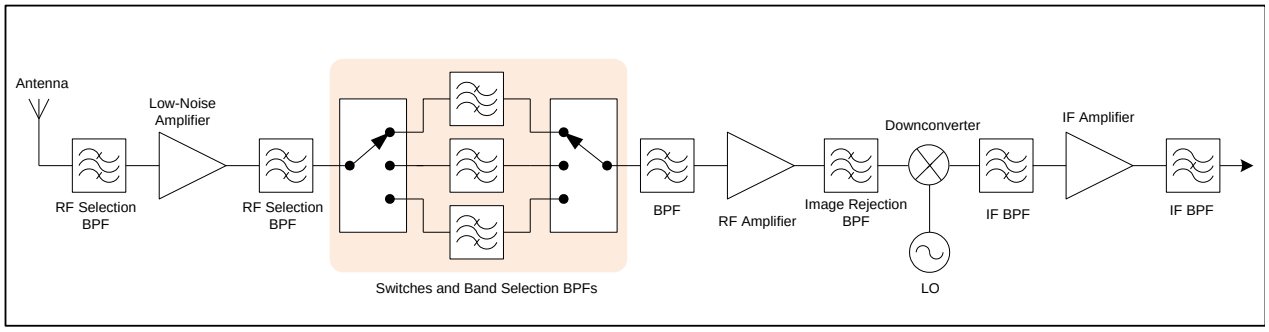
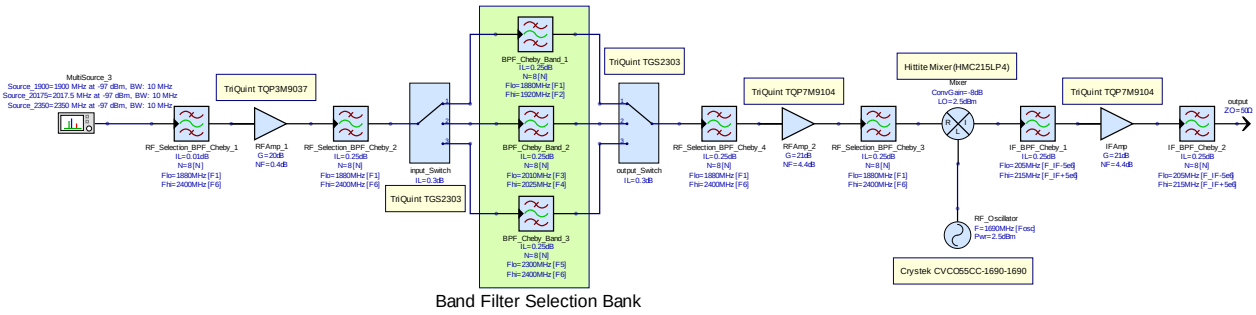


Figure 3.3. A block diagram of the transceiver in which the receiver part is represented by the RF front-end and the demodulator.

Figure 3.4.a shows the block diagram of the proposed architecture for the LTE receiver. Figure 3.4.b shows the same architecture implemented using PathWave System Design (SystemVue) parts.



(a)



(b)

Figure 3.4. Proposed super-heterodyne architecture for the LTE receiver: (a) block diagram (b) PathWave System Design (SystemVue) model.

The RF front end is composed of a selection filter to attenuate frequencies out of the desired bands. A low-noise amplifier is used to amplify the input weak signal. Two switches and a bank of filters allow the

selection of each band. The mixer downconverts the RF signal to IF1. Then, the demodulator brings IF1 to IF2 to demodulate the LTE signal.

3.4.1.3 Frequency Planning

The aim of frequency planning is the selection of an intermediate frequency that eliminates, or at least minimizes, in-band and out-of-band spurs. Traditionally, designers use spreadsheets to figure out spur-free IFs. PathWave System Design (SystemVue) includes a unique tool called the WhatIF Frequency Planner, which helps in the elaboration of a chart of all IFs at the output of the mixer.

In our case study, the receiver supports three different bands (Table 3.2). The intermediate frequency selection should fit with the different bands simultaneously.

	Lower Frequency F_L (MHz)	Upper Frequency F_H (MHz)	Central Frequency F_0 (MHz)	Bandwidth BW (MHz)
Band 1	1880	1920	1900	40
Band 2	2010	2025	2017.5	15
Band 3	2300	2400	2350	100

Table 3.2. Bands supported by the LTE receiver.

To illustrate how the WhatIF Frequency Planner works, we looked for the spur-free regions where an IF frequency can be selected for the first band only. As shown in Figure 3.5, there are several spur-free regions (green-colored columns) in which we can select an IF frequency. The lowest region lies between 75 and 347 MHz.

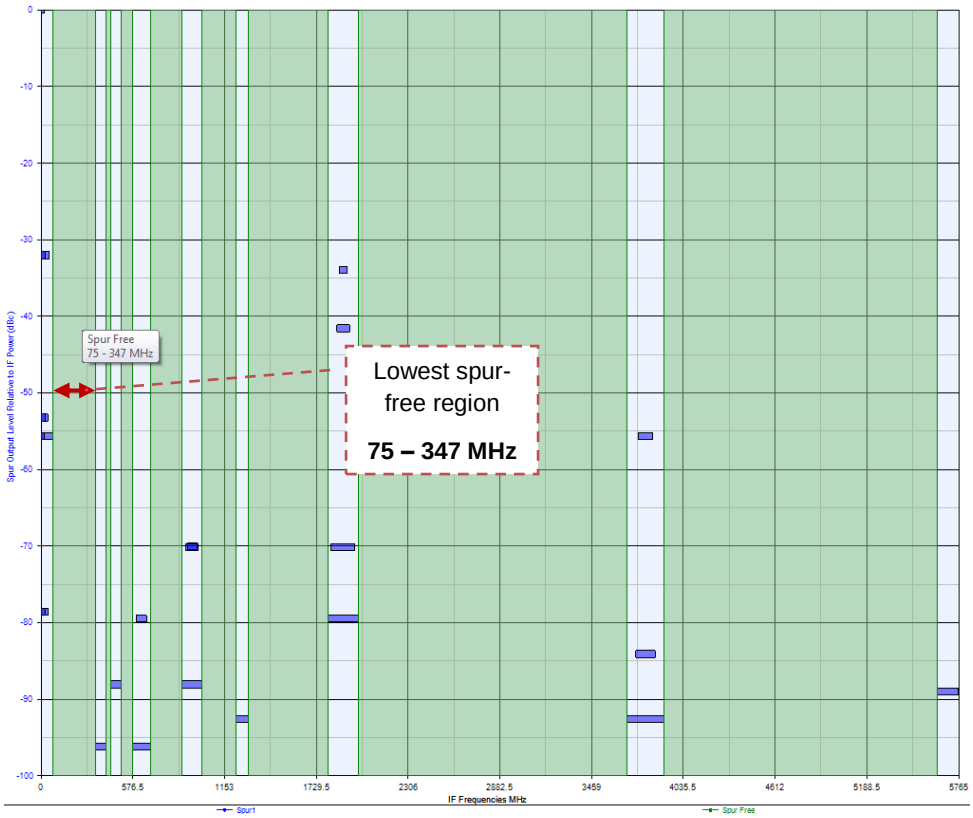
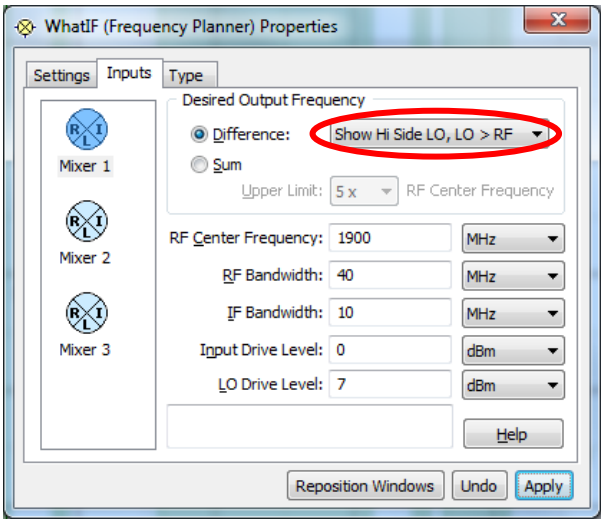
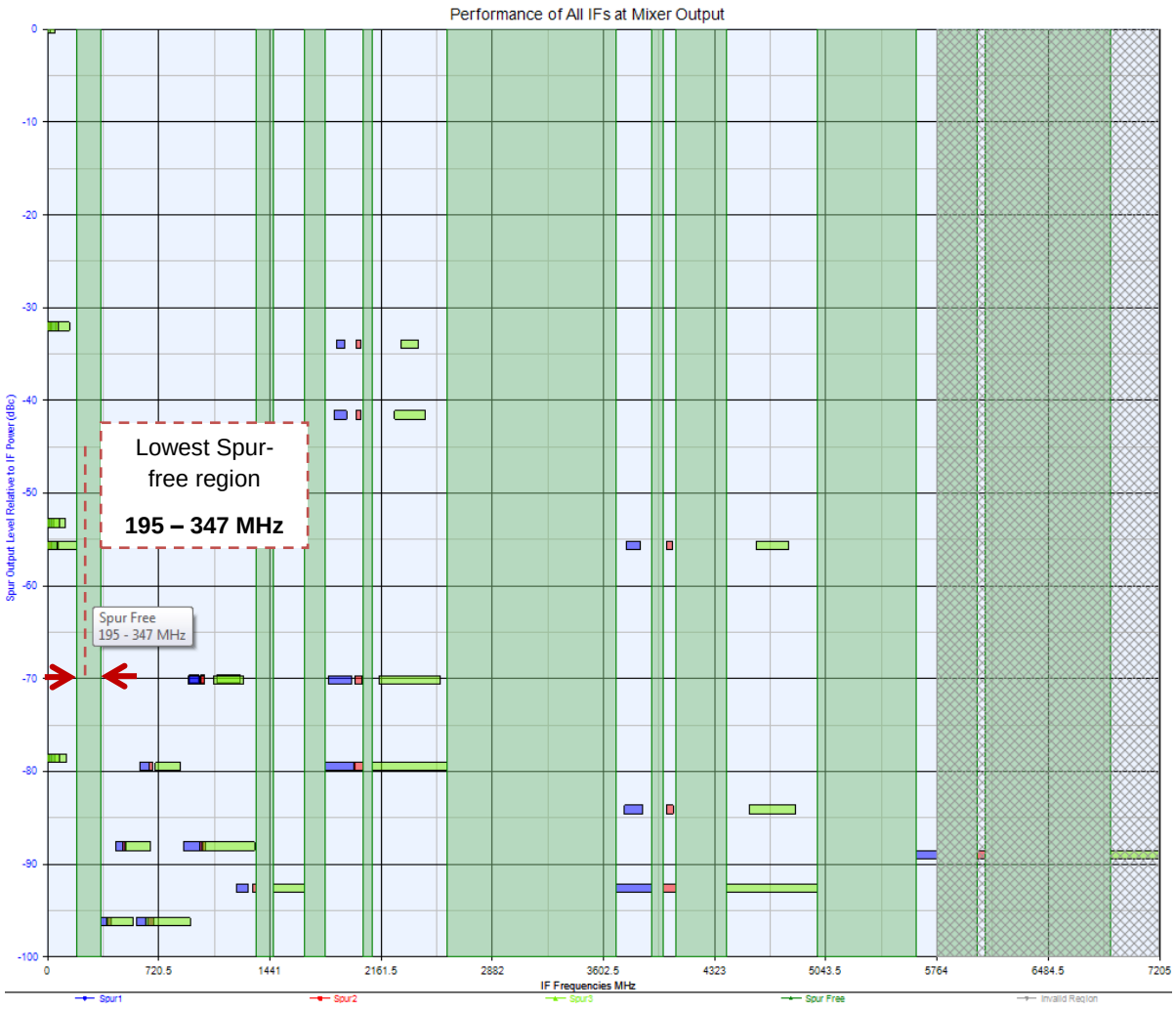


Figure 3.5. The mixer's output for band 1.



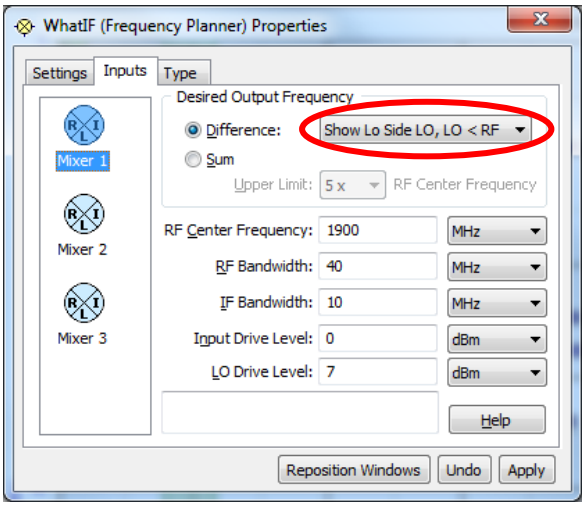
(a)



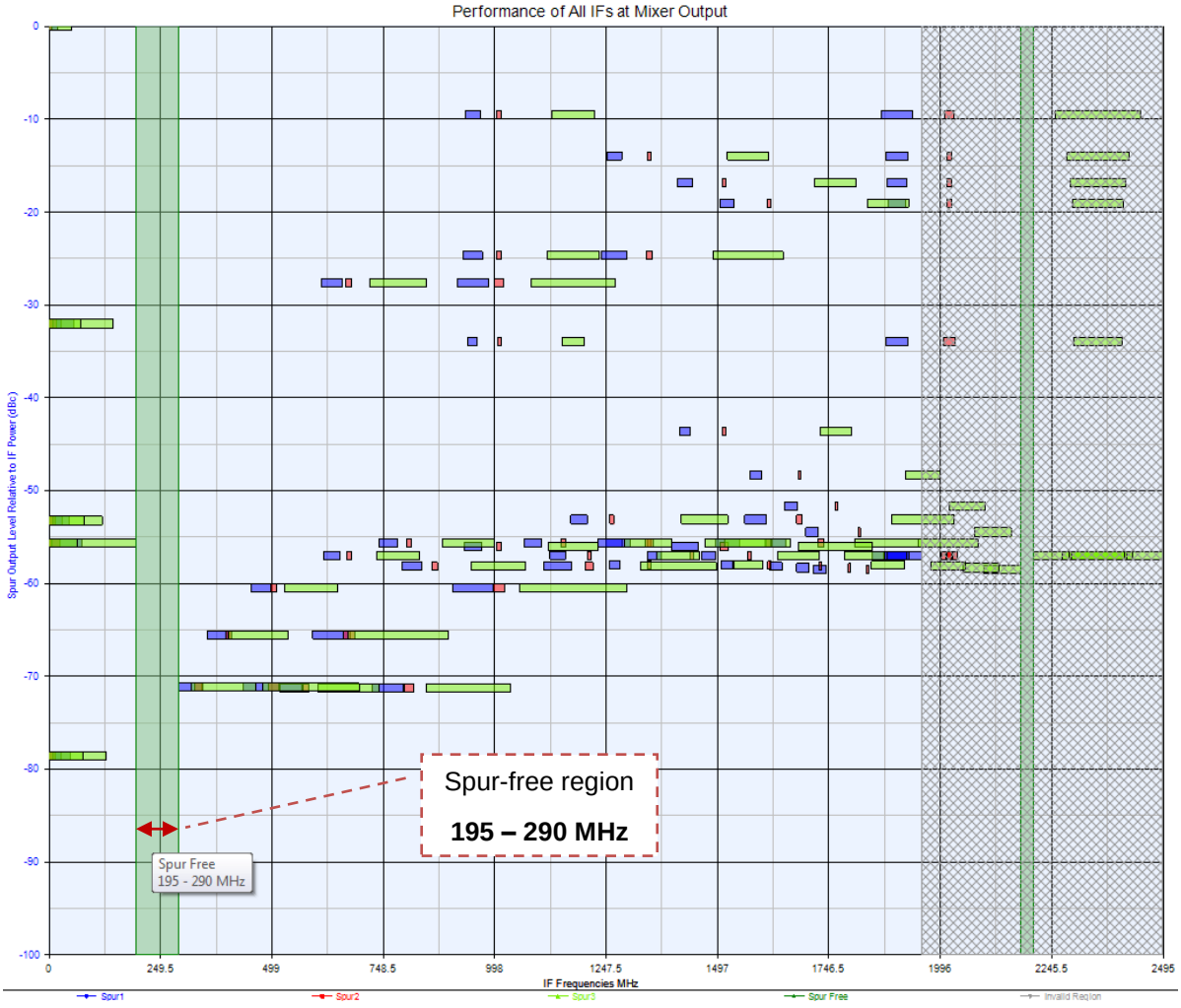
(b)

Figure 3.6. All IFs for band 1, 2 and 3 (high-side mixer): (a) WhatIF Frequency Planner Tool (b) Resulting IFs chart.

However, we need to select an IF that suits all three bands. Using the WhatIF Frequency Planner, we can look for IFs for up to five different bands.



(a)



(b)

Figure 3.7. All IFs for band 1, 2 and 3 (Low-side mixer): (a) WhatIF Frequency Planner Tool (b) Resulting IFs chart.

There are two possible configurations: High-side mixers where LO frequency is less than RF frequency and low-side mixers where the RF frequency is greater than the LO frequency. The lowest spur-free region in the resulting chart lays between 195 and 347 MHz. Figure 3.6.b shows the IF output for the three bands, considering the high-side mixer configuration, while Figure 3.7.b shows the IF output for the low-side mixer configuration. The lowest spur-free region in the resulting chart lies between 195 and 290 MHz.

The high-side configuration presents more spur-free regions where an IF can be selected than the low-side configuration. The former is used to avoid possible RF leakages but is less practical and more costly than the latter. For these reasons, we chose the low-side configuration and select an IF frequency equal to 210 MHz.

3.4.1.4 Architecture Optimization

In this case study, we chose to investigate how a superheterodyne architecture can be used in the implementation of a multiband LTE receiver. We started by analyzing an "ideal" configuration where all components are virtually considered ideal. Filter attenuation is high, and their frequency response approximates an ideal shape. Switches introduce little insertion loss. Nonlinear components such as amplifiers, mixers and oscillators have a wide linear operation margin. Figure 3.8.a shows the block diagram of the "ideal" configuration.

The second step was to replace "ideal" components with off-the-shelf devices used in the real world. We used amplifiers and switches from TriQuint. The mixer was chosen from Hittite and the oscillator from Crystek. Due to the particularities of the LTE bands, and the IF frequency used in this design ($F_0 = 210$ MHz and $BW_{IF} = 10$ MHz), we developed custom filters using PathWave RF Synthesis (Genesys). We then used the Circuit_Link part to include the custom filters in the PathWave System Design (SystemVue) schematic. Figure 3.9 shows how a filter synthesized by PathWave RF Synthesis (Genesys) can be imported in PathWave System Design (SystemVue) using the Circuit_Link part. Figure 3.8.b shows the realistic design of the LTE receiver.

Now let's consider the system-level analyses of both "ideal" and realistic designs, looking specifically at the following parameters:

- Cascaded Noise Figure (CNF)
- Cascaded Gain (CGAIN)
- Carrier to Noise Ratio (CNR)
- Channel Noise Power (CNP)
- Channel Power (CP)
- Stage Dynamic Range (SDR)

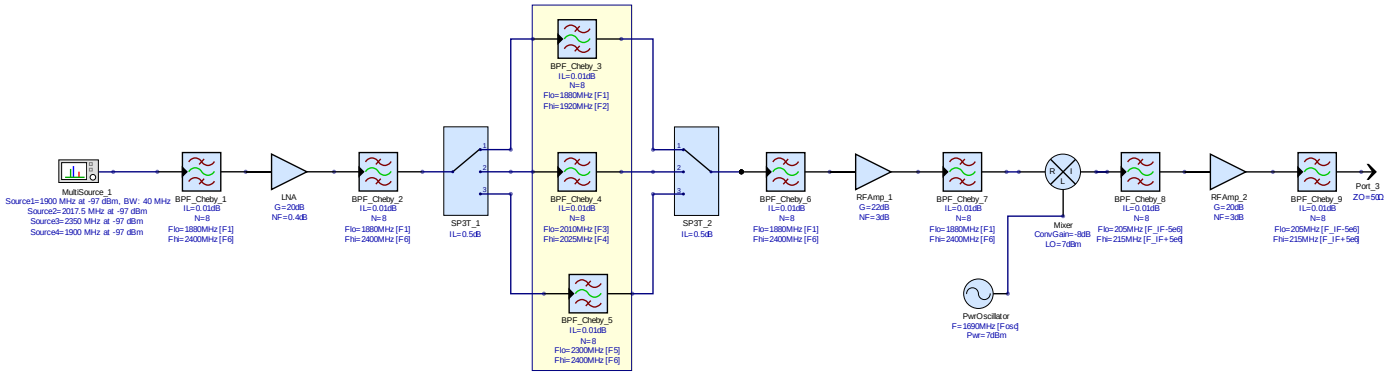
For each band and for each design, we used PathWave System Design (SystemVue) to simulate the parameters listed above. Table 3.3 summarizes the simulation results.

	CGAIN (dB)	**CNF (dB)**	**CNR (dB)**	**SDR (dB)**	**CP (dBm)**	**CNP (dBm)**

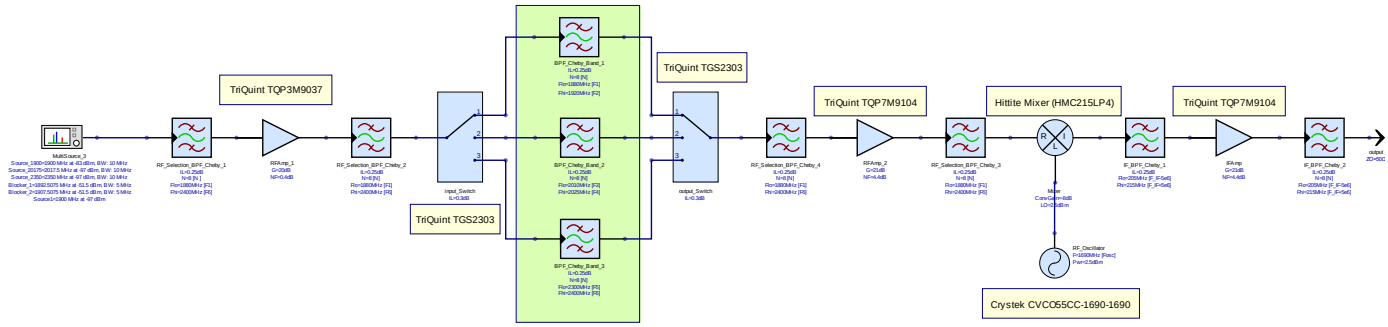
Ideal Configuration	**Band 1**	51.184	1.057	15.918	137.347	-45.702	-61.734
	Band 2	50.606	1.104	15.871	137.7	-46.279	-62.265
	Band 3	51.341	0.903	16.072	139.162	-45.549	-61.731
Realistic Configuration	**Band 1**	48.53	1.606	15.22	139.615	-48.338	-63.69
	Band 2	44.28	3.121	13.705	142.492	-52.535	-66.425
	Band 3	49.547	1.236	15.589	140.706	-47.33	-63.042

Table 3.3. Simulation results of some system-level parameters for both architectures.

Figure 3.10 through Figure 3.13 show the evolution of each parameter throughout the different stages of both designs. Figure 3.14 shows also the compression curves that help predict the linear and nonlinear operation regions of each design.



(a)



(b)

Figure 3.8. The LTE transceiver schematic: (a) ideal configuration (b) realistic configuration.

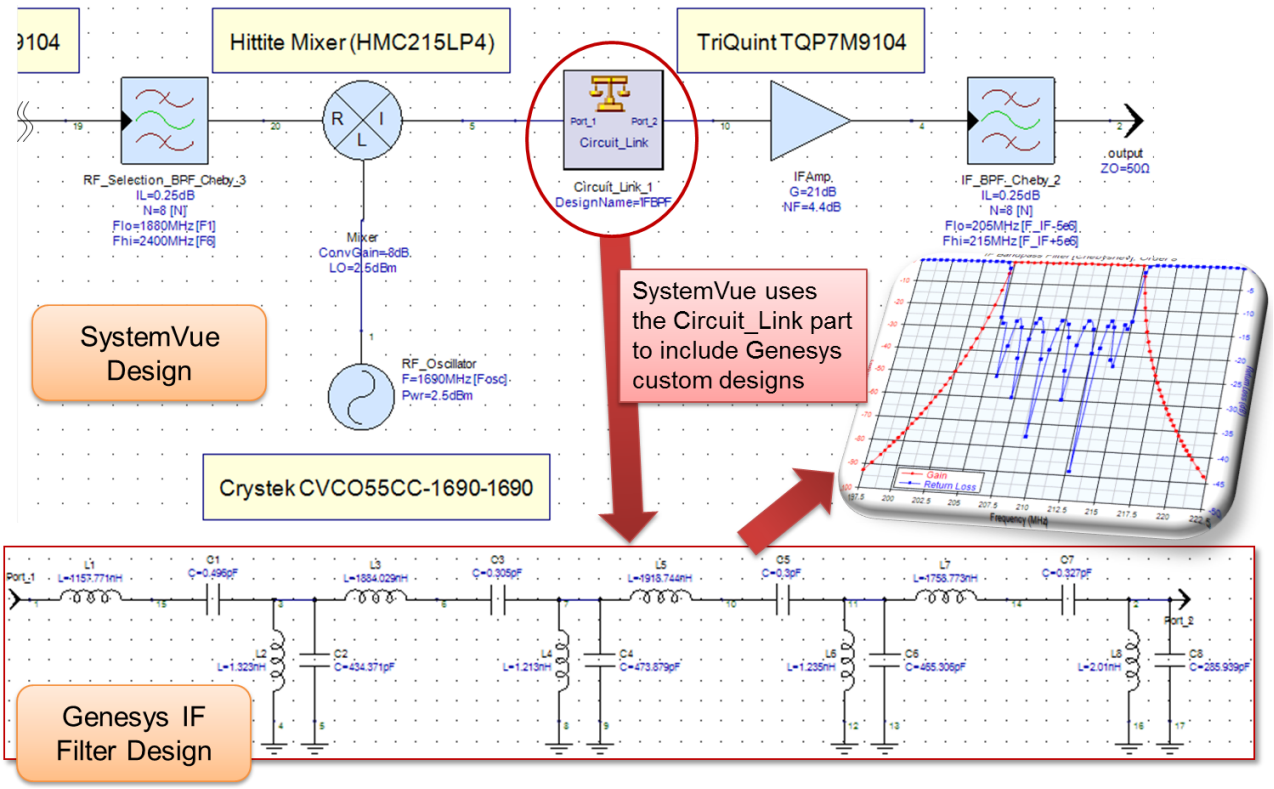


Figure 3.9. PathWave System Design (SystemVue) uses the Circuit_Link part to include custom designs (e.g., from RF Synthesis (Genesys)).

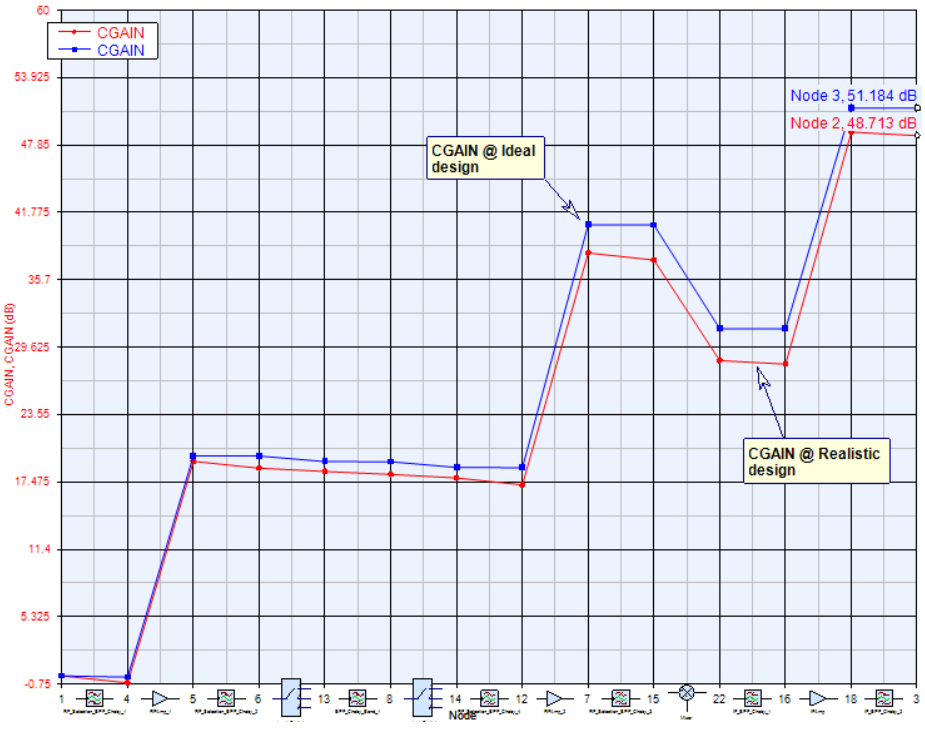


Figure 3.10. Cascaded Gain.

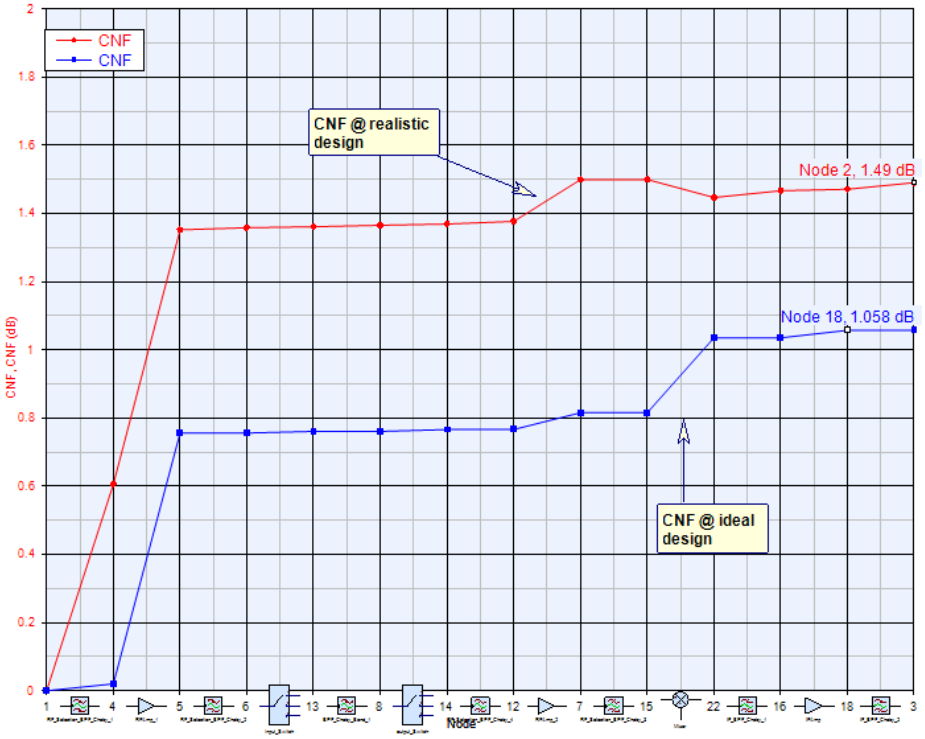


Figure 3.11. Cascaded Noise Figure (CNF) for band 1.

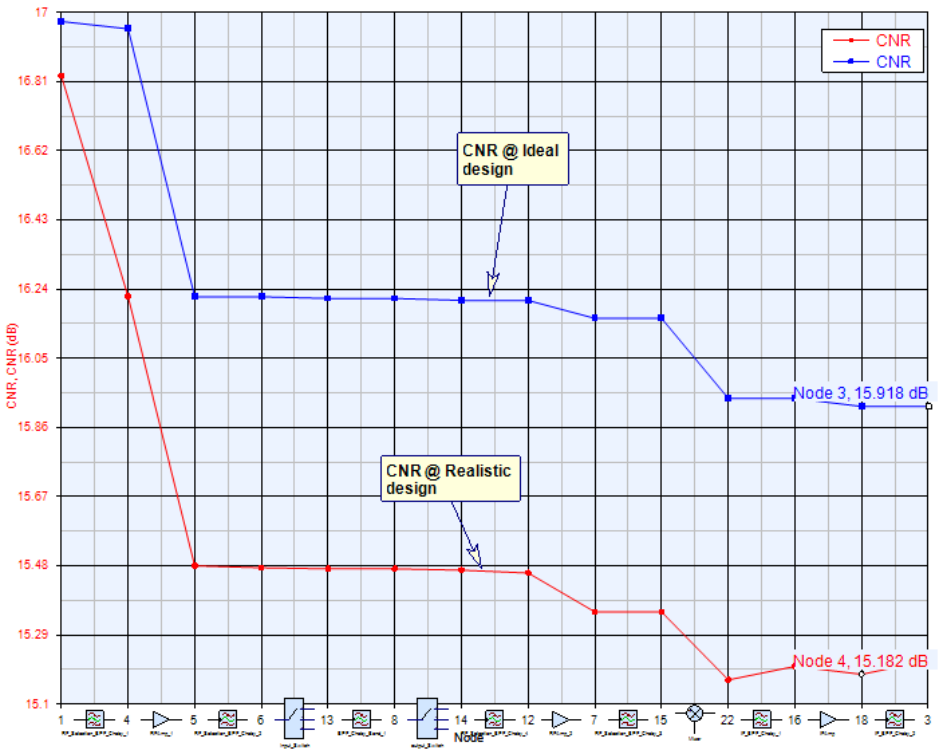


Figure 3.12. Carrier-to-Noise Ratio for band 1.

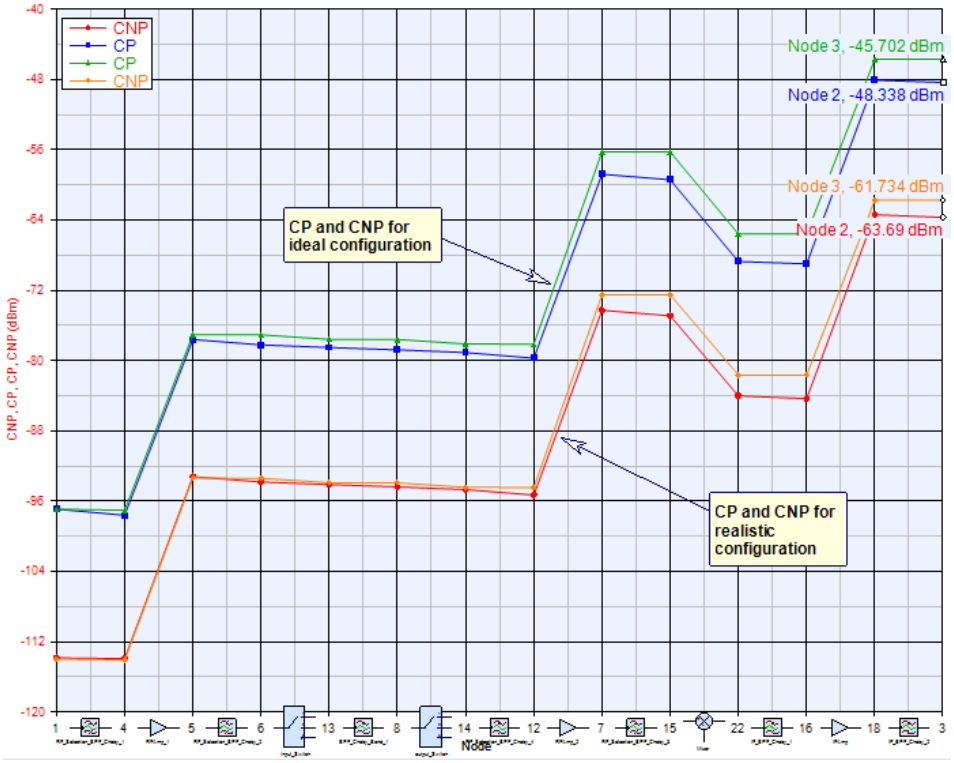


Figure 3.13. Channel Power and Channel Noise Power for ideal and realistic configuration.

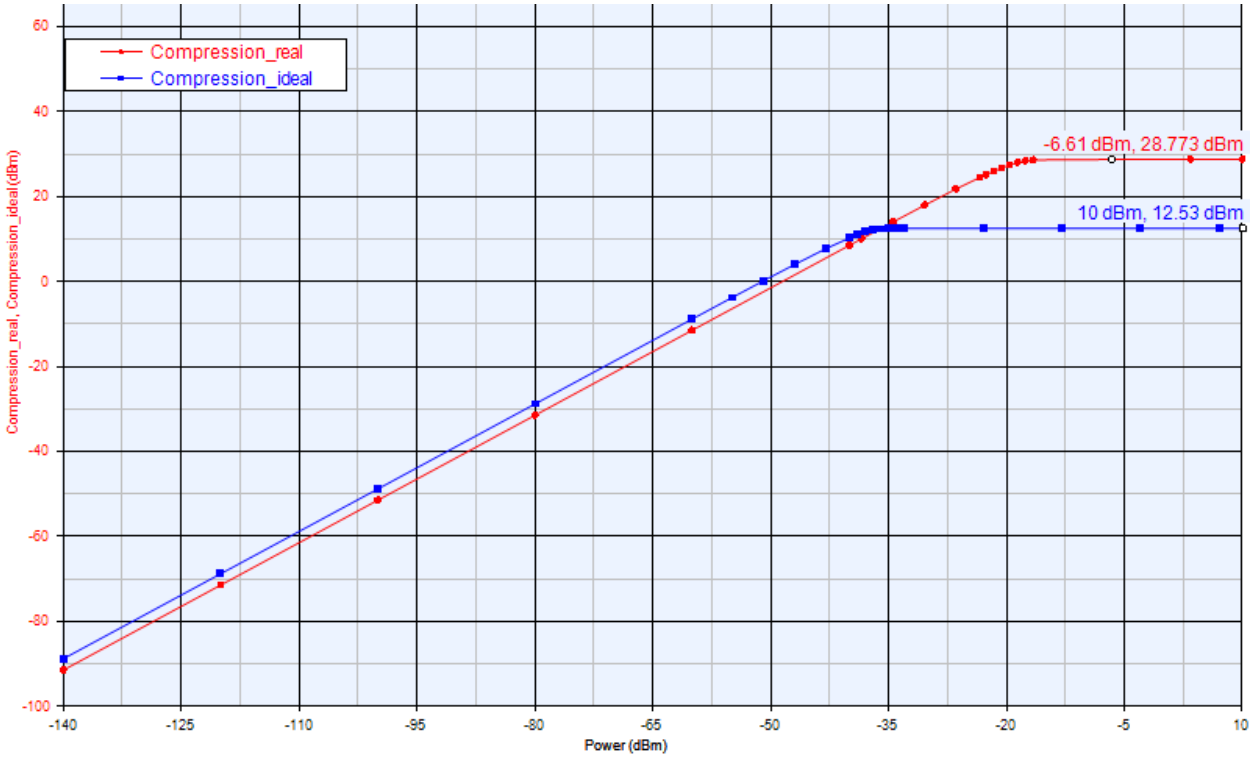


Figure 3.14. Compression curve for band 1: ideal configuration (red), real configuration (blue).

3.4.1.5 Performance Evaluation

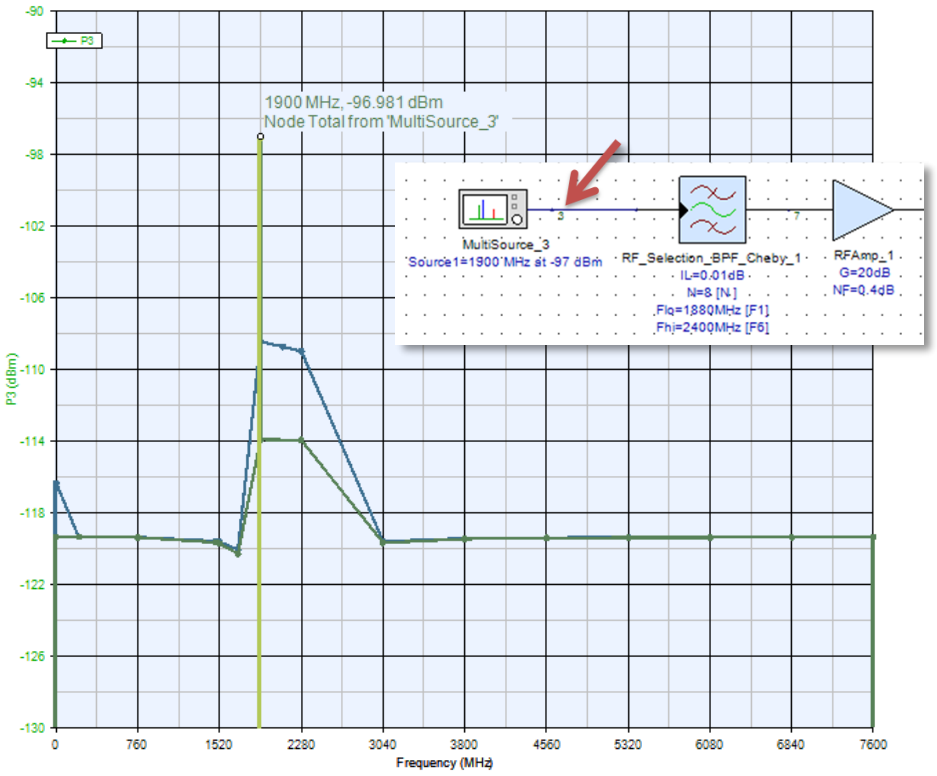
Section 7 of the LTE user equipment specifications[41] defines the requirements that an LTE receiver should meet in terms of sensitivity, adjacent channel selectivity (ACS) and blocking characteristics (narrowband, in-band and out-of-band), as well as the spurious response and intermodulation. In the following sections, we will illustrate how to use PathWave System Design (SystemVue) to test and validate the receiver architecture regarding the LTE standard.

Table 3.4 presents the reference sensitivity specifications, assuming the modulation scheme is QPSK and the channel bandwidth is 10 MHz. It also presents the maximum input power specified for a relative throughput greater than 95%.

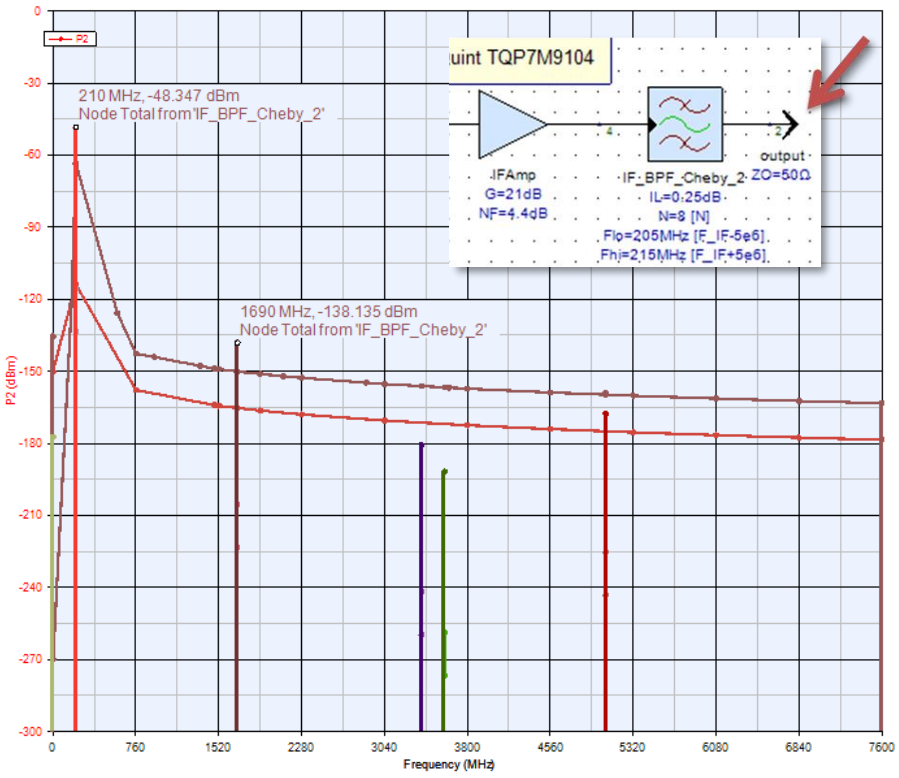
E-UTRA Band	Channel Bandwidth (MHz)	Reference Sensitivity for QPSK (dBm)	Maximum Input Power (dBm)
34	10	-97	-25
39	10	-97	-25
40	10	-97	-25

Table 3.4. Reference Sensitivity and Input Power specifications for TDD bands 34, 39 and 40.

Figure 3.15.a shows an input tone at 1900 MHz and has a total power of -97 dBm. At the input, a tone of -48 dBm is located at 210 MHz and the LO tone is attenuated up to -138 dBm.



(a)

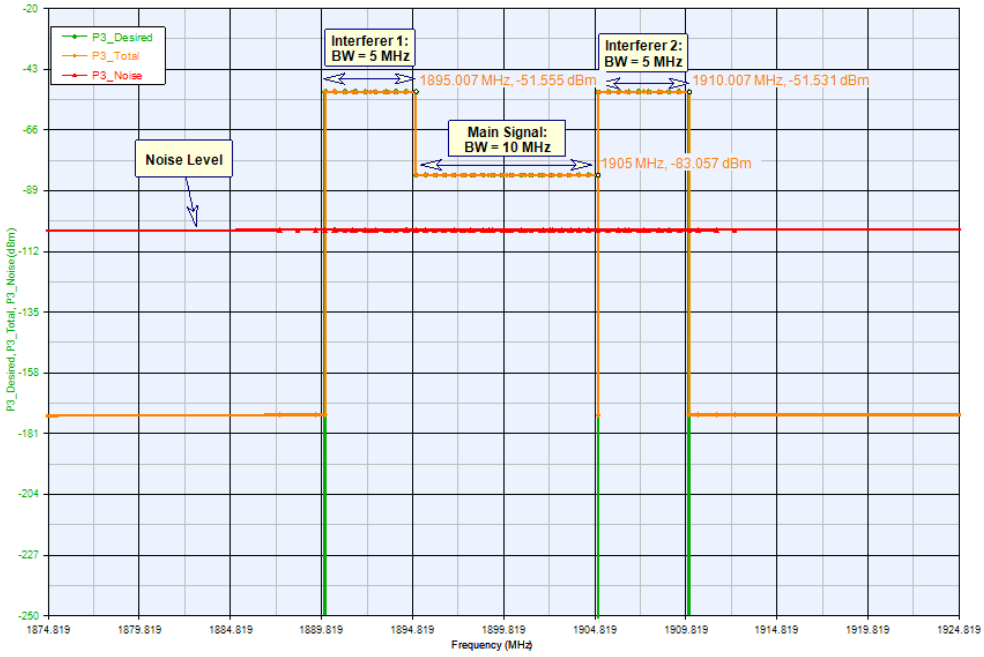


(b)

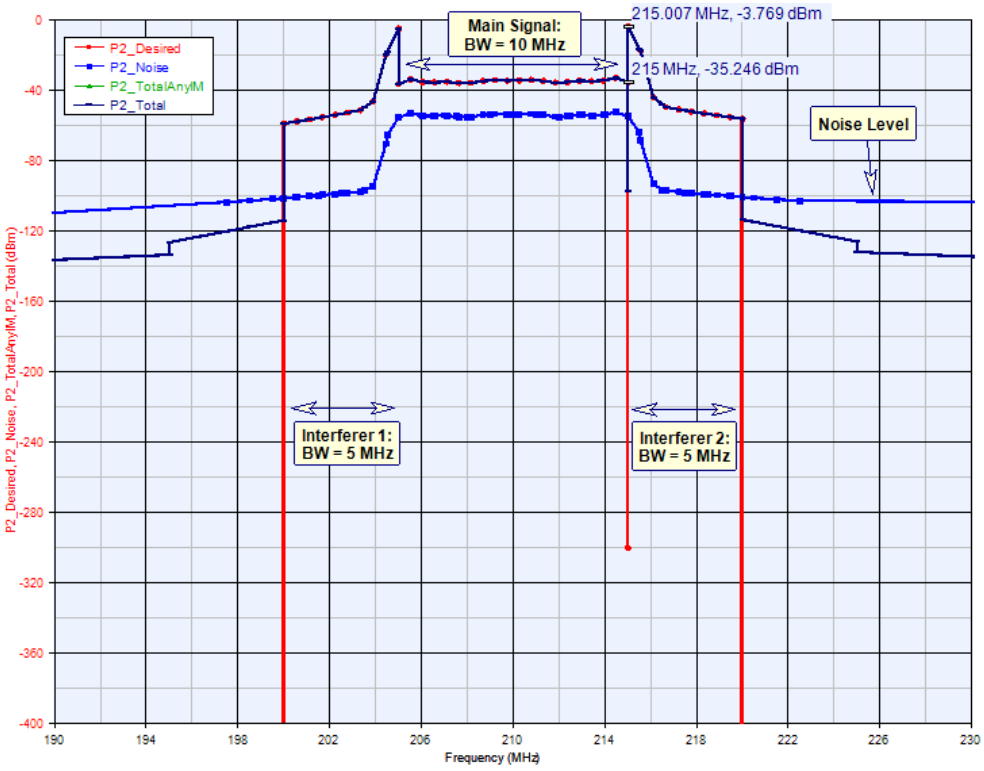
Figure 3.15. Input and Output Spectrum for a 1900-MHz tone: (a) Input Signal (b) Output Signals including the LO tone at 1690 MHz.

3.4.1.6 Adjacent Channel Selectivity

The ACS, also called Adjacent Channel Rejection (ACR), measures the ability of a receiver to select an LTE signal at its assigned channel frequency in the presence of an adjacent channel signal at a given frequency offset.



(a)



(b)

Figure 3.16. Adjacent Channel Selectivity: (a) Input blockers and main signal (2) Output spectrum (ACS ≈ 30 dB).

Reference 41 states that ACS should be equal to or greater than 33 dB in all circumstances, assuming a

maximum allowed input power of -25 dBm and a relative throughput equal or greater to 95%.

			Channel Bandwidth	
Receiver Parameter	**Units**	**…**	**10 MHz**	**…**
Input Signal Power	dBm		Reference Sensitivity + 14 dB (i.e. = -97 + 14 = -83)	
Interferer Power	dBm		Reference Sensitivity + 45.5 dB (i.e. = -97 + 45.5 = -51.5)	
Interferer Bandwidth	MHz		5	
Interferer Frequency Offset	MHz		±(7.5 + 0.0075)	

Table 3.5. Summary of ACS test parameters (based on Table 7.5.1-2 of reference 39).

Based on the test parameters presented in Table 3.5, we used the Spectrasys simulator to evaluate the ACS of the realistic design. Figure 3.16.a shows the input signals we used to stimulate the RF design (two 5-MHz wide interferers at a power level of approximately -51.5 dBm), as well as the desired signal (10-MHz wide channel at a power level of approximately -83 dBm). The output signals are shown in Figure 3.16.b. Both the interferers and the desired signal were amplified and downconverted by our architecture. Interferers lay at the low and high sides of the desired band from approximately 200.0075 MHz to 205.0075 MHz and from 215.0075 MHz to 220.0075 MHz, respectively. Both interferers' power level is about -3.7 dBm. The desired signal power level is about -35.25 dBm. The ACS is around 31.45 dB, which does not meet the standard requirements (i.e., $ACS \leq 33$ dB). For this reason, we added two quite sharp baseband filters to gain an additional 3 to 5 dB and enhance the system's overall selectivity (Figure 3.18 below).

3.4.1.7 Blocking Characteristics

The blocking characteristic measures the receiver's ability to select the desired signal at its assigned channel frequency in the presence of unwanted interferers other than those caused by spurious response and adjacent channels. There are several categories of blocking types: in-band, narrow-band and out of band.

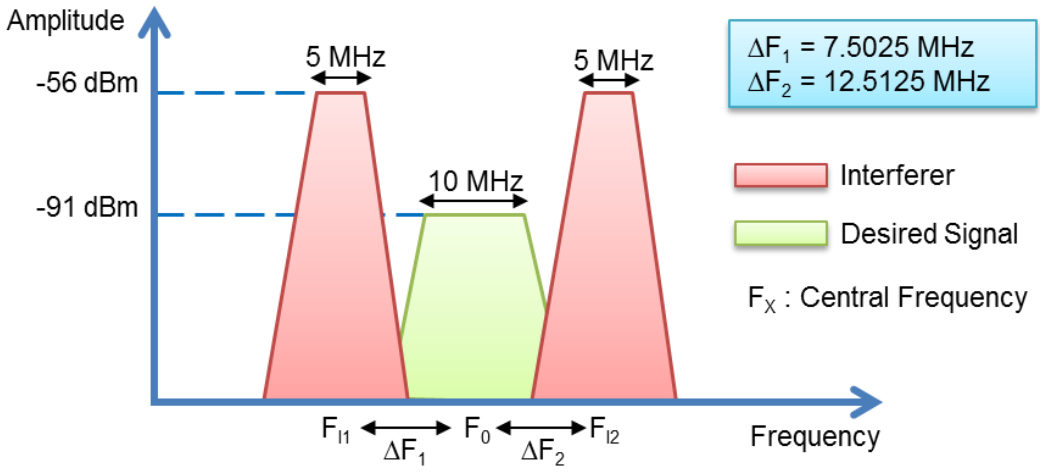


Figure 3.17. Out-of-band blocking specifications.

In this section, we focus on how to use PathWave System Design (SystemVue) to measure the impact of in-band blockers on our LTE receiver. Figure 3.17 illustrates an in-band blocking test detailed in Tables 7.6.1.1-1 and 7.6.1.1-2 of the UE Specifications.[41]

To evaluate the response of the LTE receiver regarding the introduced in-band blockers, the relative throughput (also called throughput fraction) is measured. The relative throughput must be greater than or equal to 0.95 (95%) to state that the receiver passed the in-band blocking test.

A full LTE data flow testbench is used to evaluate the receiver's performance. Figure 3.18 shows the data flow parts using the LTE library to modulate, transmit, receive and demodulate an LTE TDD signal. It is worthwhile to mention that the LTE data flow testbench uses mainly the Data Flow simulator, which interacts automatically with the Spectrasys simulator to evaluate the RF design.

To evaluate the receiver's performance, we need to consider the following two main scenarios:

- Analysis of in-band blockers impact
- Analysis of the standard thermal noise impact

We proceeded gradually in this study by activating (or/and deactivating) blocking and noise density parts at each iteration. Table 3.6 summarizes the results of our simulations. We used the linear channel estimator in the LTE Downlink Receiver part. According to rows 1 through 4, the blockers have no concrete impact on the receiver's performance in the absence of noise. Furthermore, when the noise density part is activated, the throughput fraction is less than 95%. This impact is confirmed by rows 9 and 10 where the RF architecture is not considered, and the noise impact is evaluated regarding a reference LTE setup.

No.	Noise Density?	Blocker 1?	Blocker 2?	Throughput (bps)	Fraction (%)
1	No	No	No	1603200	100

2	No	No	Yes	1603200	100
3	No	Yes	No	1603200	100
4	No	Yes	Yes	1603200	100
5	Yes	No	No	1406328	87.72
6	Yes	No	Yes	1480224	92.329
7	Yes	Yes	No	1418736	88.494
8	Yes	Yes	Yes	1437072	89.638
9	No	-	-	1603200	100
10	Yes	-	-	1423128	88.768

Table 3.6. Simulation results in the case of a linear channel estimator.

To enhance the performance of the system, we replaced the linear channel estimator of the LTE downlink receiver part by the Minimum Mean Square Error (MMSE) channel estimator, which is known to be more resistive to noise. The simulation results are summarized in Table 3.7. As expected, the performance of the receiver is better. The throughput fraction is 100% despite the presence of thermal noise and blockers.

Note that we did not examine all blocking tests.

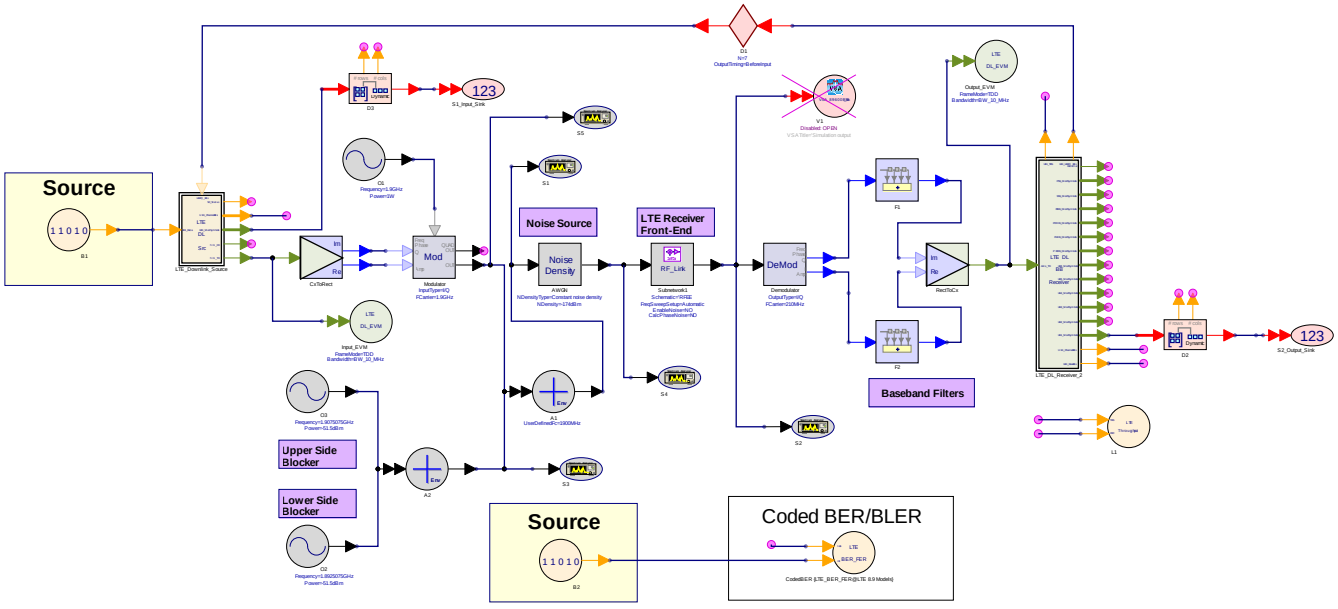
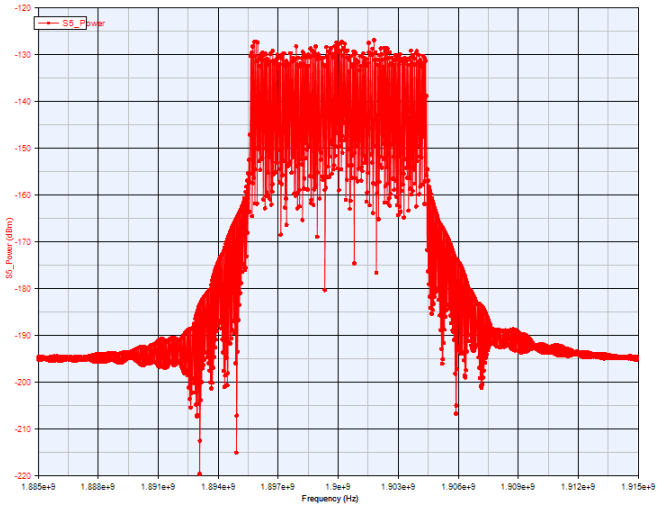


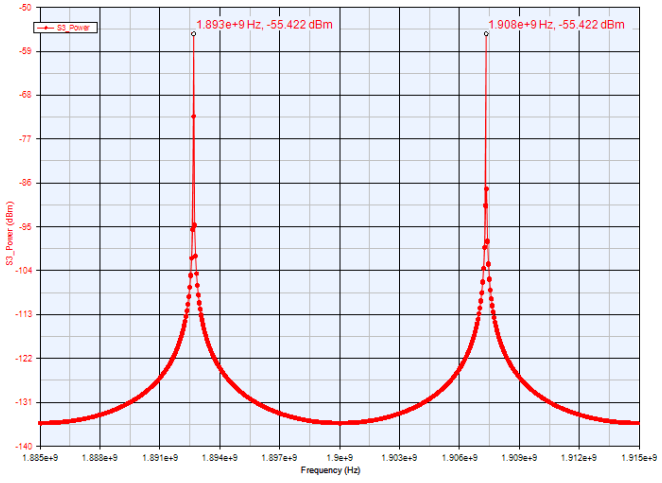
Figure 3.18. LTE Data Flow setup to test the LTE receiver front-end.

No.	Noise Density?	Blocker 1?	Blocker 2?	Throughput (bps)	Fraction (%)
1	No	No	No	1603200	100
2	No	No	Yes	1603200	100
3	No	Yes	No	1603200	100
4	No	Yes	Yes	1603200	100
5	Yes	No	No	1603200	100
6	Yes	No	Yes	1603200	100
7	Yes	Yes	No	1603200	100
8	Yes	Yes	Yes	1603200	100
9	No	-	-	1603200	100
10	Yes	-	-	1603200	100

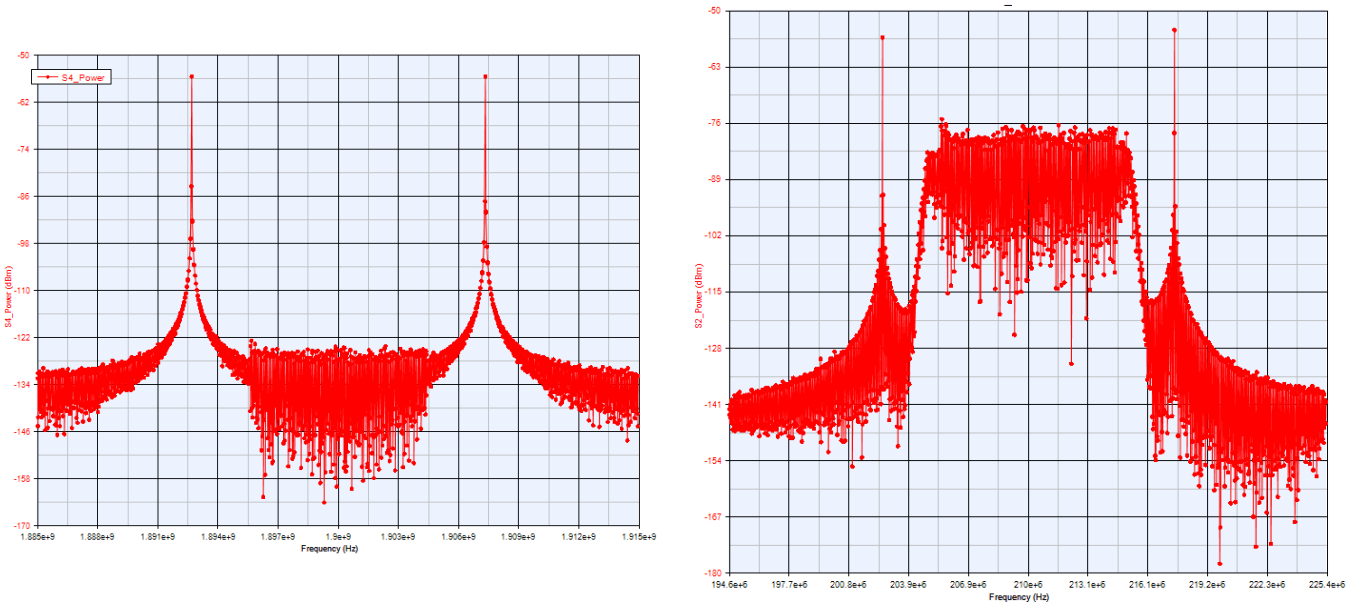
Table 3.7. Simulation results in the case of the MMSE channel estimator.



(a)



(b)



(c) (d)

Figure 3.19. Band 1 spectrum: (a) Transmitter signal (b) Blockers (c) Noisy signal (d) Downconverted signal.

Figure 3.19 shows the LTE signal of band 1. The original signal is shown in Figure 3.19.a. Figure 3.19.b shows the blockers. Figure 3.19.c shows the received signal with noise and blockers. And, Figure 3.19.d shows the downconverted signal measured at the output of the RF front-end.

3.5 Summary

In this chapter, we introduced the concepts of a design methodology and system-level modeling. We selected a case study from the real world to illustrate how some aspects of the RF architecture can be designed and evaluated at the system-level using the various features and capabilities of PathWave System Design (SystemVue). There are many other RF aspects and design issues (e.g., the effects of phase noise and nonlinearities) that could also have been investigated. These applications and others are available in a range of PathWave System Design (SystemVue) videos, tutorials and examples.

For additional information about Keysight PathWave System Design (SystemVue) software, please visit

- http://www.keysight.com/find/eesof-systemvue
- http://www.keysight.com/find/eesof-systemvue-videos

Chapter 4: Introduction to Digital Communication

4.1 Introduction

This chapter will present the basics of digital communication and introduce the theory and structure of a digital radio system. Most computing devices today require some sort of wireless functionality, and with the rise of portable electronics this trend will only increase. With this increase comes a need for better, efficient, and more reliable signals, which in turn has cleared a path for the use and design of different wireless data transmission methods. One of the most common ways to implement data transfers over long distances is by way of digital communication.

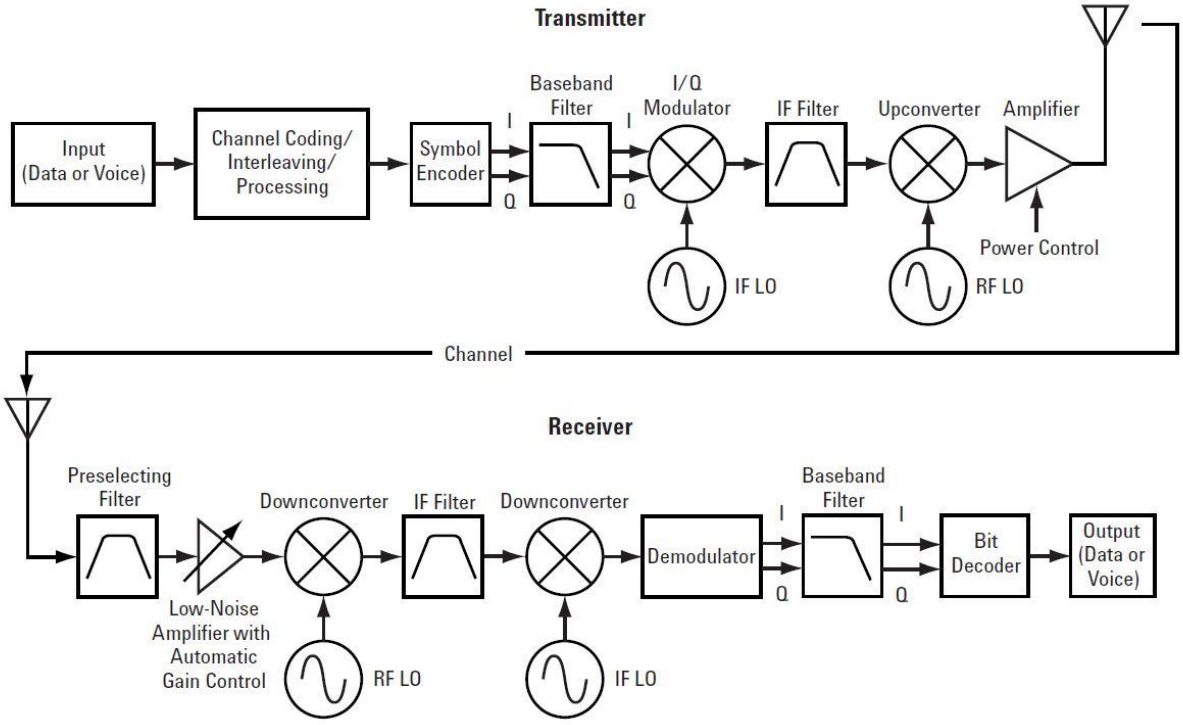


Figure 4.1. Graphical diagram of a basic digital communication system.

Figure 4.1 shows a rudimentary block diagram of a digital radio communication system, which is usually made up of a transmitter, channel, and receiver. The digital radio signal experiences many transformations in its migration from a baseband signal at the transmitter to its replication at the receiver. The block diagram reveals the transformation process the signal undergoes from origination to reception. The system-level diagram in Figure 4.1 displays the symmetry of the digital radio. To a certain degree, the receiver can be considered a reverse implementation of the transmitter. Consequently, the measurement challenges are similar for both parts of the digital radio system. However, unique problems exist at various locations in the system since modern day receivers and transmitters face

different design problems. For example, the receiver has correction and synchronization schemes to help clear and decipher noise contaminated and time-delayed signals.[48]

4.2 Transmitter

The digital radio transmitter accepts in a baseband waveform and translates that signal into a waveform that it can effectively transmit through the channel. When data first enters the transmitter, it is coded to be more easily transmitted, which helps with error correction and is more secure. Before the transformation from baseband to a Radio Frequency (RF) channel, the waveform is digitized to leverage the advantages of digital modulation. Coding is applied to the signal to more efficiently use the available bandwidth and to minimize the effects of noise and interference that will be introduced by the channel. The coded signal is filtered, modulated, and changed back to an analog waveform that is converted to the desired frequency of transmission. Finally, the RF signal is filtered and amplified before it is transmitted from the antenna.

4.3 Receiver

The digital radio receiver can be implemented several ways, but certain components exist in all receivers. The receiver must extract the RF signal in the presence of potential interference. Consequently, a preselecting filter is the first component of the receiver, and it attenuates out-of-band signals received by the antenna. A Low-Noise Amplifier (LNA) boosts the desired signal level while minimally adding to the noise of the radio signal. A mixer downconverts the RF signal to a lower Intermediate Frequency (IF) by mixing the RF signal with a Local Oscillator (LO) signal. The IF filter attenuates unwanted frequency components generated by the mixer and signals from adjacent frequency channels. After the IF filter, the variations in receiver design manifest themselves.

4.4 Channel

In between the transmitter and receiver is the channel, which is a kind of medium that can range from copper wire cables to open air. Because our primer is mostly designed around the digital domain, we really can't go in-depth into the channel, as it deals with the RF domain. However, the channel still dictates the parameters on which the baseband system is built. For example, through a copper wire signal integrity is much stronger then the integrity of a signal sent through the viscous air, so data can be sent in faster, more error prone modulation schemes. In the air transferred signals suffer from fading and echoing and must be modulated at a slower but safer modulation scheme. The channel is an important part of dictating the system requirements. It ensures communications have good signal integrity, and a low BER (bit error rate).

4.5 Other Considerations

Besides modulation schemes, there are various other design choices to consider. As said earlier, the type

of encoding used on the input data can help counteract data loss, create a secure channel, or even interleave multiple data streams upon one transmission. Not only that but the frame structure, or way the data itself is structured, can decrease the BER by including frame synchronization data. Filters can be changed and modified to decrease noise, occupy fewer frequencies, and/or are more desirable long range. Other filters cost less and are energy efficient despite only being reliable through a short range. So, depending on the specifications, there are a number of transmitter-receiver setups.

Chapter 5: Transmitter Design

5.1 Transmitter Basics

The transmitter can be seen as one half of a whole in digital communications, and is responsible for coding, converting and sending a signal along the channel. It can be set up in a number of different ways; however, for the purpose of this document, this chapter will just briefly cover the setup of the transmitter. It will also illustrate how to construct the transmitter in PathWave System Design (SystemVue), while providing basic theory at the end of each section.

A general example of a transmitter layout can be seen in Figure 5.1. The first step is to convert a continuous analog signal to a discrete digital bit stream. This is called digitization.

The next step is to add voice coding for data compression. Then some channel coding is added. Channel coding encodes the data in such a way as to minimize the effects of noise and interference in the communications channel. Channel coding adds extra bits to the input data stream and removes redundant ones. Those extra bits are used for error correction or sometimes to send training sequences for identification or equalization. This can make synchronization (or finding the symbol clock) easier for the receiver.

The symbol clock represents the frequency and exact timing of the transmission of the individual symbols. At the symbol clock transitions, the transmitted carrier is at the correct I/Q (or magnitude/phase) value to represent a specific symbol (a specific point in the constellation). Then the values (I/Q or magnitude/phase) of the transmitted carrier are changed to represent another symbol. The interval between these two times is the symbol clock period. The reciprocal of this is the symbol clock frequency. The symbol clock phase is correct when the symbol clock is aligned with the optimum instant(s) to detect the symbols.

The next step in the transmitter is filtering. Filtering is essential for good bandwidth efficiency. Without filtering, signals would have very fast transitions between states and, therefore, very wide frequency spectra. A single filter is shown for simplicity in Figure 5.1, but in reality, there are two filters; one each for the I and Q channels. This creates a compact and spectrally efficient signal that can be placed on a carrier.

The output from the channel coder is then fed into the modulator. Since there are independent I and Q components in the radio, half of the information can be sent on I and the other half on Q. This is one reason digital radios work well with this type of digital signal. The I and Q components are separate.

The rest of the transmitter looks similar to a typical RF or microwave transmitter. The signal is converted up to a higher intermediate frequency (IF), and then further upconverted to a higher radio frequency (RF). Any undesirable signals produced by the upconversion are then filtered out.[49] The transmitter system is graphically described in Figure 5.1.

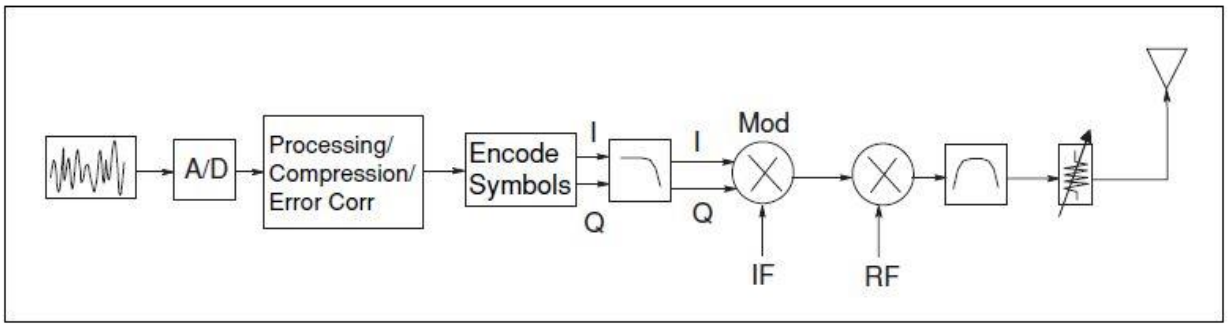


Figure 5.1. A graphical model of a basic transmitter.

5.1.1 Reference: LinearModulation example

The goal of this chapter will be to create your own transmitter in PathWave System Design (SystemVue). The example we'll be using is based off the LinearModulation example. This is a basic example that will cover most of the general transmitter design.

An easy way to reach this example, and all examples in PathWave System Design (SystemVue), is to use the Open Example option under the Help tab, or use "Ctrl E." To navigate to the LinearModulation example:

1. Help -> Open Example/Ctrl E (shown below in Figure 5.2).

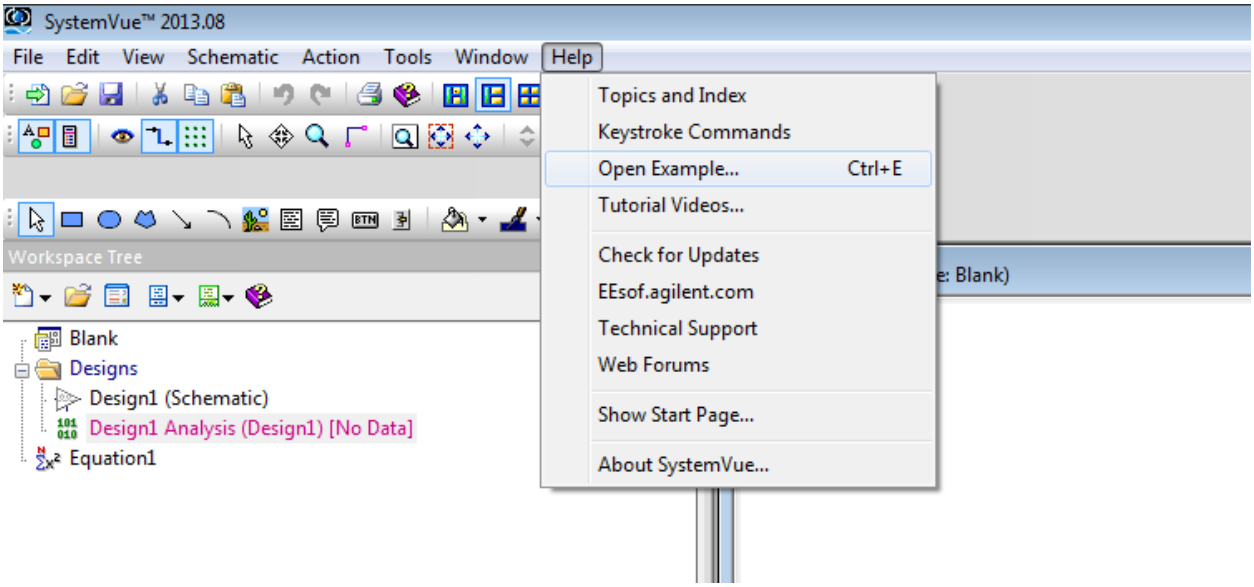


Figure 5.2. "Open Example" location.

2. From here go the Comms folder.
3. Open the DigitalModulation folder. The example should be located here.

This should lead you to a workspace example with the schematic layout shown in Figure 5.3, which will form the basis of our transmitter design.

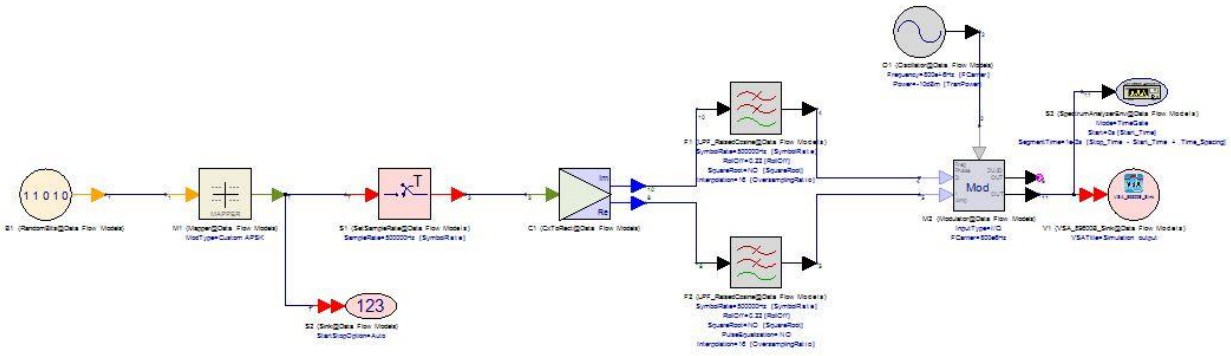


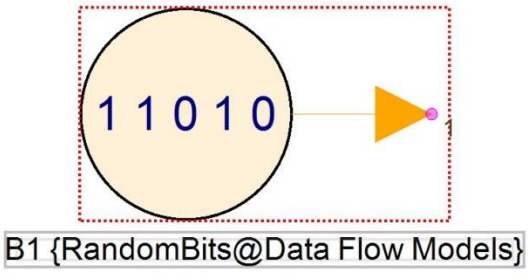
Figure 5.3. Schematic from the LinearModulation example.

It's important to note that our final schematic will be akin to that pictured in Figure 5.3, although it won't be an exact replica.

5.2 Input Data

To start, let's

1. Create a new workspace in PathWave System Design (SystemVue) and save it as "Cook_Transmit."
2. Name the Schematic "Transmitter," the Data Flow Analysis "Transmitter_DF," the Folder "Source," and the Equation "TransVari."
3. Go to Data Flow Analysis -> set the Design to transmitter, the Dataset to "Transmitter_Data."
4. Search for "Bits" in the parts selector, or press "B" and then click to place the block (Figure 5.4).
5. Double click on Bits and enable "Advanced Parameters." Change the "SampleRateOption" to "0:Untimed."



Name	Value	Units
ProbOfZero	0.5	( )
BitRate	Sample_Rate	Hz
ShowAdvancedParams	1:YES	( )
SampleRateOption	0:UnTimed	( )
SampleRate	Sample_Rate	Hz
InitialDelay	0	( )
BurstMode	0:OFF	( )

(a) **(b)**

Figure 5.4. The Bits model (a). The part of the Bits properties window (b).

This will represent our data that we wish to send from our transmitter in our digital communications example.

Framing: Unlike what we have in the above example where random bits are sent out with no meaning, in a real transmitter, you usually want to have coherent and real information in a compartmentalized way. This usually involves a combination of changing the structure of the data you wish to send out and adding extra bits to it. When a process like this is planned out, it is called a frame structure. The frame structure acts as a protocol that informs the receiver on how to accept and decipher the incoming transmissions by adding on useful synchronization information and loss detection. A basic frame structure is shown in Figure 5.5

Figure 5.5. A diagram of basic frame structure.

Here the Idle Interval and the Guard represents a string of null bits that show whether the packet is starting or ending (although depending on the transmitter, either the Idle or the Guard can be sent first). The Preamble is a sizable amount of sorting, channel estimation, equalization, and miscellaneous data that's used to help the receiver sort out the packet. The Payload is the data itself.

To implement this in PathWave System Design (SystemVue) usually you would have to go through either the equations editor or place a "MathLang" block and code it yourself. Instead, we've provided an all in one transmitter block in the DigMod Library called "DigMod_SourceL_RF." This block is specially designed to act as an ideal source. In it you can set up different parameters like Preamble length, Payload length and Preamble Modulation (Mod) type.

5.3 Encoding

The method we're using to channel code is called convolutional code (Figure 5.6). To implement this:

1. Search for and place the "ConvolutionalCoder" block
2. Attach the Bits output to ConvolutionalCoder input.

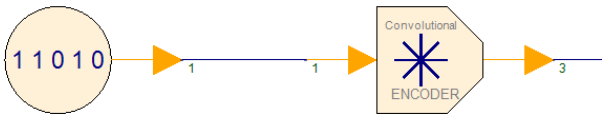


Figure 5.6. Bits block and convolutional encoder block in series.

Channel Coding and Convolution Coding: When transmitting over most mediums, noise may interfere with the information in the signal. Channel coding is a general process added to help compress the signal and structure the outbound data in such a way that it becomes error resistant. Coding can be

done in many different ways, although this example uses convolutional coding.

Convolution coding is an error-correcting code. It is first structured by polynomial equations. Usually the polynomials equate to registers based on their power, and each equation is usually set up to be an individual output. An example of it is shown in Figure 5.7. Here, you can see the coder in action.

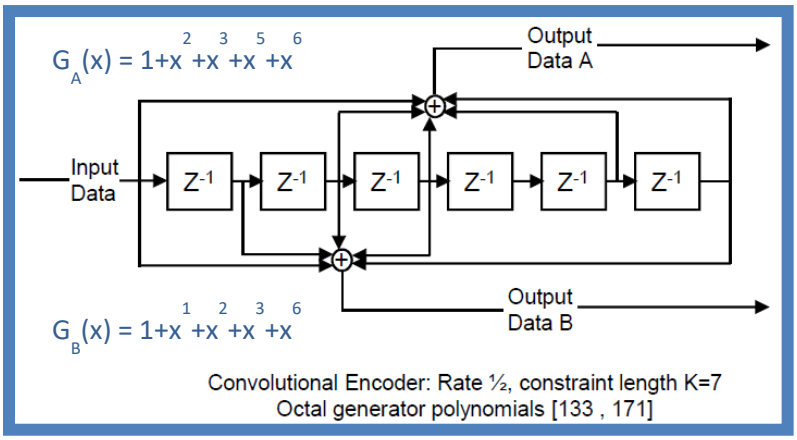


Figure 5.7. Convolution coding diagram with equations representing the different outputs.

The convolutional coder works by entering the data into a series of registers that hold the data stored. Each clock second, the data is moved from register to register, which in turn, feeds into a series of adders. Each adder in this case represents a different equation and is set up by the equation parameters. If you look at the powers of the polynomials, they correspond to the output of the register shown in the diagram. So, for example, if you look at equation G_A there is a polynomial with the 5^{th} power, which is represented by the adder coming from Output Data A. This isn't shown in Output Data B, because there's no 5^{th} power polynomial in G_B. This changes bits streams regardless of length, effectively doubling each bit that comes in. You can modify the input/output ratio and the polynomial powers by editing the block. More advanced convolution topics and coding will need to be done in Math Lang, or by adding adders and registers.

5.4 Mapping

Next, we want to place our encoded data into symbol form.

1. Find and place the "Mapper: Complex Signal Mapper" on the diagram.
2. Connect the Mapper to the Encoder.
3. Edit the Mapper block and change its Mod type to 16 QAM.

By the end of these steps, your block diagram should look a bit like that shown in Figure 5.8.

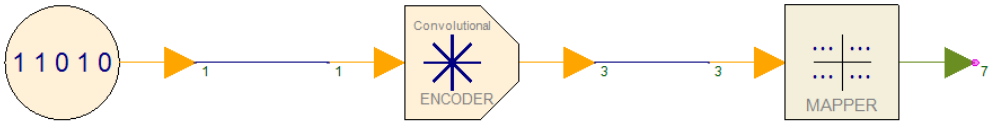


Figure 5.8. A Bits, Convolutional Encoder, and Mapper block all in series.

Mapping: Mapping is the act of bunching groups of bits together to form symbols. The amounts of bits used to form groups are usually arbitrary but have an effect on the transfer rate of data and the integrity of the signal. Usually the more bits grouped together, the more information can be transferred, and when this happens usually there is less room for error. These symbols can usually be represented by graphing them on what's called a constellation map. This map is integral for IQ Modulation.

The Mapping block in PathWave System Design (SystemVue) can be modified to support different modifications from QAM's to your own custom APSK.

5.5 Pulse Shape Filtering

To set up the channel filtering for the simulation,

1. Place a "SetSampleRate" block and connect it to the mapper.
2. Place a "CxToRect" block and connect it to the SetSampleRate (Figure 5.9).

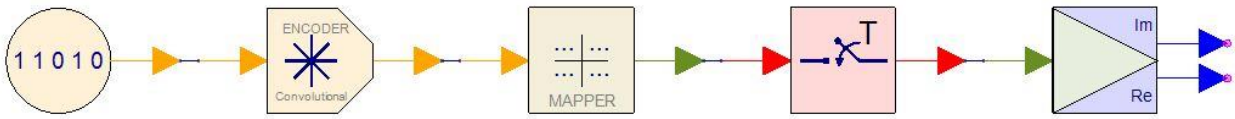


Figure 5.9. Your Transmitter schematic diagram so far.

In the SetSampleRate block properties, we could normally just set the SampleRate value to a constant, but we will have to use the symbol rate at other times within this simulation.

Let's create a symbol rate variable in the Equation window.

3. Double-click TransVari in the Workspace tree and in the text editor, set the variable "SymbolRate" to 500e3 (Figure 5.10).

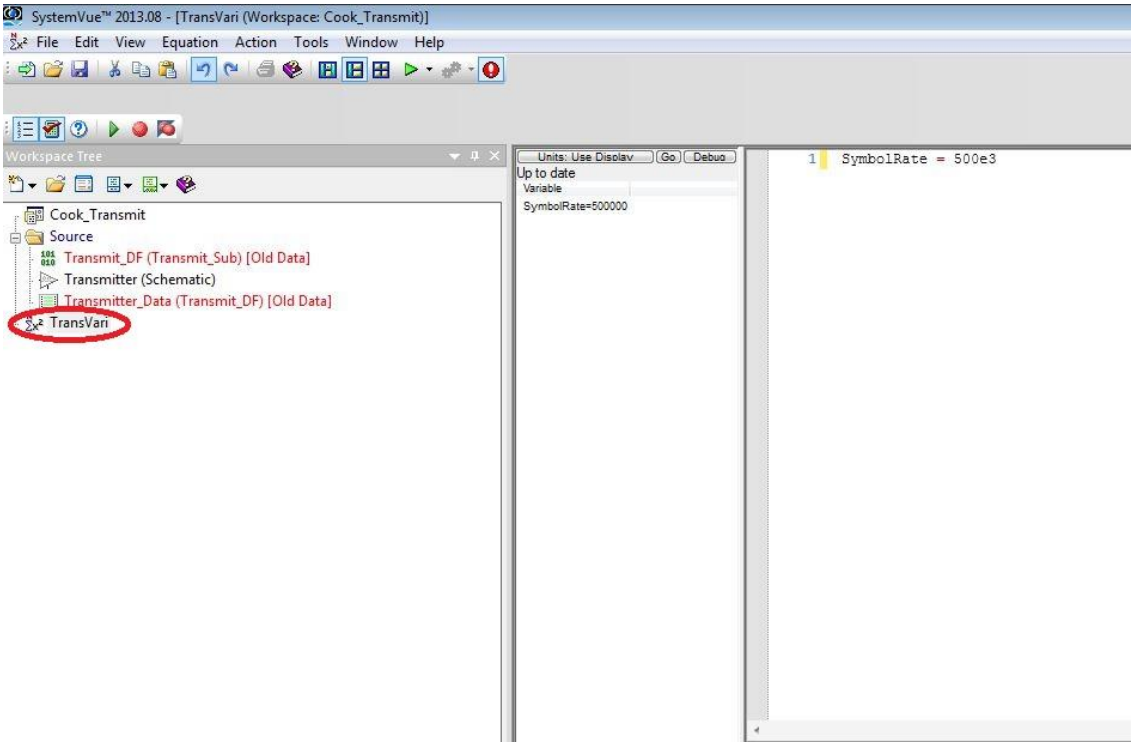


Figure 5.10. Snapshot of both the equation page and declaration of the SymbolRate variable.

4. Go back to the schematic and edit the SetSampleRate block's SampleRate value to "SymbolRate."

   Now it's time to implement the pulse shaping filter for the transmitter. To start:

5. Find and place a "FIR" filter.
6. Edit the FIR filter by clicking the "Filter Designer" button at the bottom of the FIR filter's Properties page.
7. Under Specifications, change the Filter Response drop down tab to Low pass from Custom. The menu should change to show a list of filter designs. Under the FIR section, select Raised Cosine and then click OK to close the window.
8. Double-click the FIR filter again. The resulting window should look relatively the same. Change the Model (under the "Description," above "Model Help") to "LPF_RaisedCosine@Data Flow Models." Alternatively, you could just type "LPF_RaisedCosine@Data Flow Models" into the Model prompt area; however, these steps illustrate the different options you have when designing a filter block.)

   Here we see many parameters. For most of them, we will want variable names.

   Go into the Equation editor as shown in Figure 5.11 and start setting the following variables to the following values.

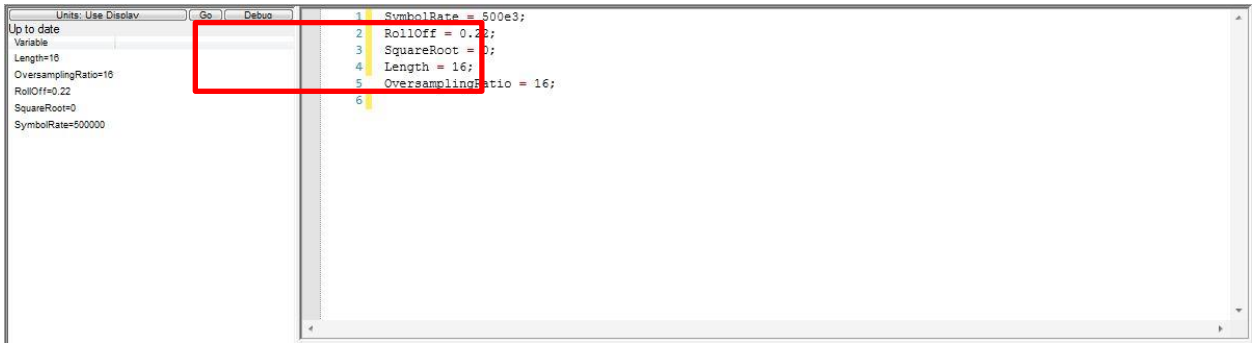


Figure 5.11. The red box contains the added variables in the TransVari Equation page.

After setting these variables, it's time to put them into the raised cosine filters we plan on using. Double click the LPF and:

9. Change SymbolRate to "SymbolRate," RollOff to "RollOff," and SquareRoot to "SquareRoot." Change LengthOption to "2:Number of Symbols," Length to "Length," and Interpolation to "OverSamplingRatio."

   The changes are shown in Figure 5.12.

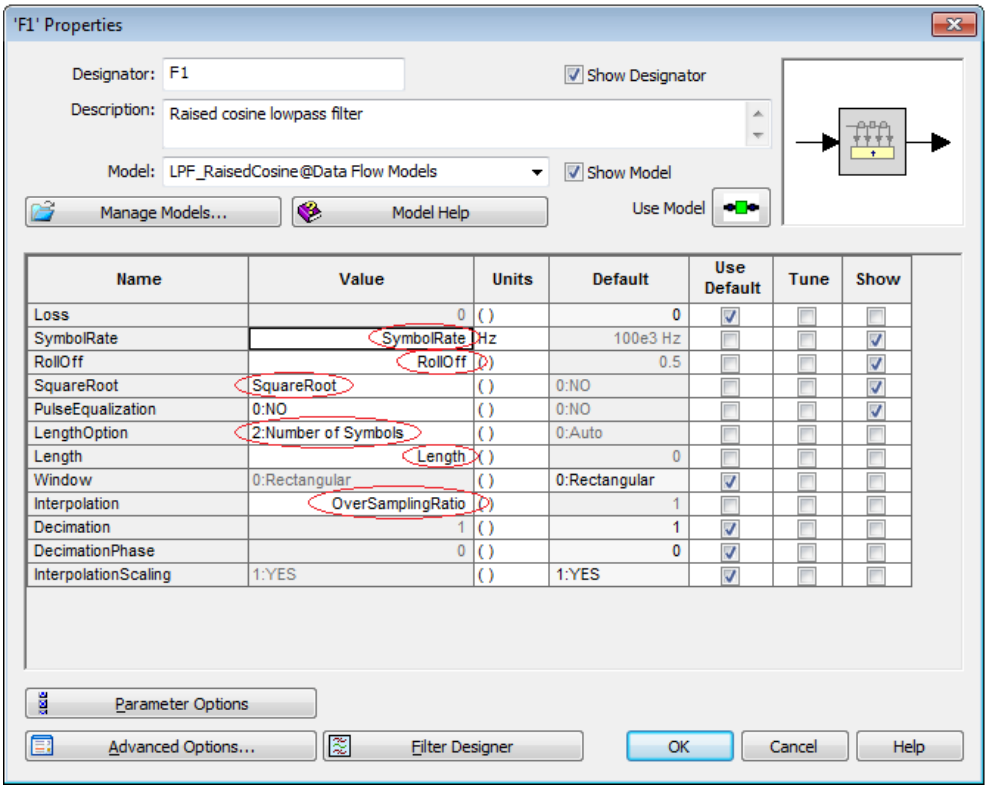


Figure 5.12. LPF properties window with the changes required circled.

10. Click “OK” and make a copy of the LPF filter (right click copy/Ctrl c). Paste a duplicate into your schematic. Next, attach each filter individually to the output lines of Im and Re on the CxToRect block.

After, your schematic should resemble that shown in Figure 5.13.

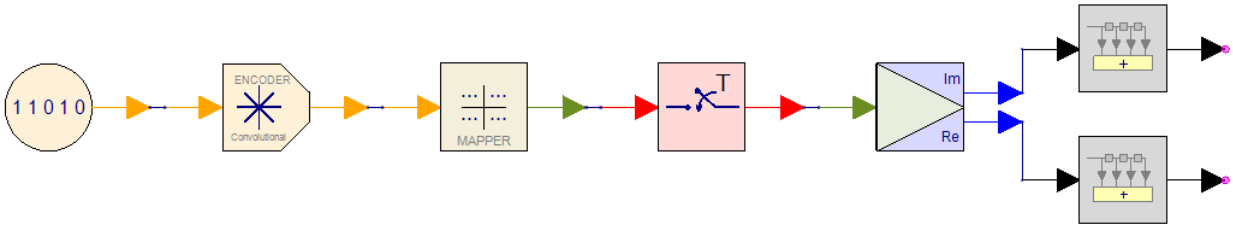
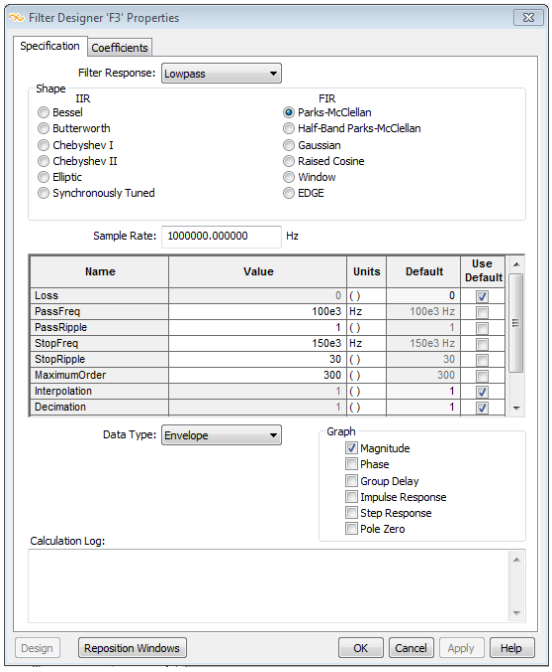
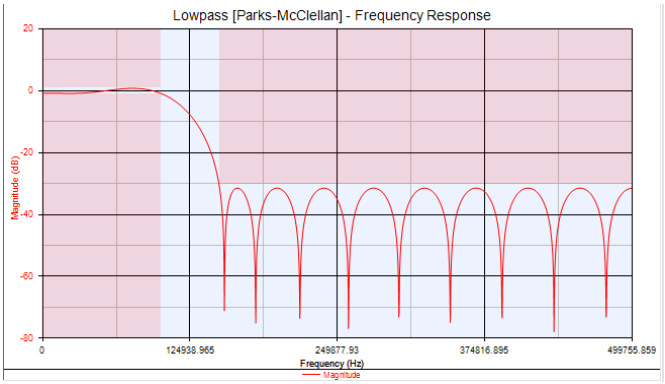


Figure 5.13. Partially completed transmitter including filtering.

Designing Filters: Filter design in PathWave System Design (SystemVue) provides the user a great amount of control. PathWave System Design (SystemVue) comes with a versatile Filter Designer option that is usually accessed by modifying the “Filter” block. It’s usually accessed in the filter blocks settings. There is a filter designer button near the bottom of the window that will bring the design window up. The window is shown in Figure 5.14.



(a)



(b)

Figure 5.14. Filter Designer properties window (a) and Filter Designer specifications graph (b).

To the left of the main window, we see a whole host of different filter responses, shapes and parameters. Filter responses low pass, high pass, bandstop, and bandpass are all available, as well as an option to create a custom response. Under that is the shape function, allowing you to choose the shapes by design method. Below the sample rate you can modify different parameters like symbol rate, roll off, and upsampling (interpolation). Afterwards you can see the frequency response of your newly designed filter by hitting design (Figure 5.14 on the right).

5.6 I/Q Modulation

Now to set up the IQ modulation.

1. Find the “Modulator” block and place it in the schematic.
2. Connect the ends of the filters to the two inputs of the modulator. When connecting the modulator, make sure the “Im line” connects to the Q (Freq Phase) input, and the “Re line” connects to the I (Amp) input.

   Notice that on the top of the Modulator block there is another input. This input is used for an optional carrier signal which will be used in the simulation. As an additional note, our Modulator block has a default carrier signal option that will be used if another signal is not fed into the input. You can change the frequency by modifying FCarrier, and/or modifying the initial phase.

3. Search Oscillator in the parts selector -> place it onto the schematic, connecting it to the (top) frequency carrier input in the Modulator block.

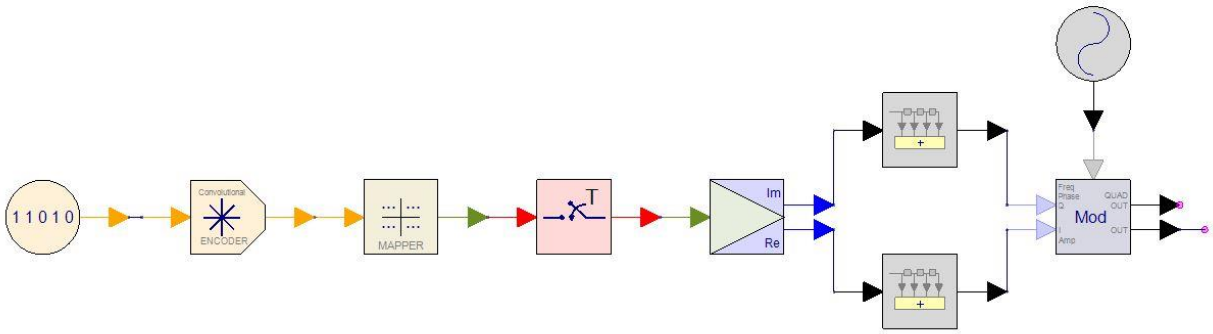


Figure 5.15. A diagram of the Cook_Transmit schematic with an added oscillator and modulation block.

Your example should look like Figure 5.15.

4. Open the TransVari Equations window and input the variables FCarrier = 800e6, TranPower = -10, and SamplingRate = Symbol Rate*OverSamplingRatio, as shown in Figure 5.16.

```
 8   FCarrier = 800e6;
 9   TranPower = -10;
10   SamplingRate = SymbolRate*OverSamplingRatio;
```

Figure 5.16. Added variable initialization lines to the TransVari Equation page.

With these variables defined, we can now edit the Oscillator.

5. Change the Frequency to FCarrier, Power to Tran Power in dBm, Advanced Parameters to "1:YES," SampleRateOption to "1:Timed form SampleRate," and SampleRate to "SamplingRate," as shown in Figure 5.17.

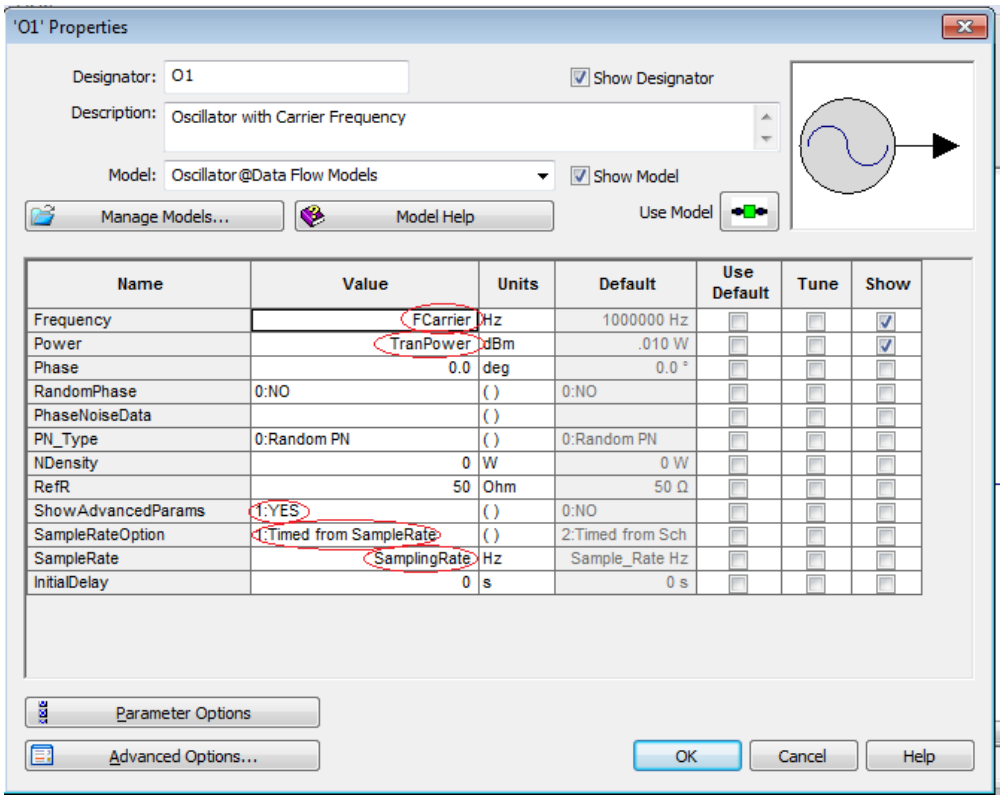


Figure 5.17. Oscillator properties window with changes circled.

IQ Modulation: A process that uses a modulating waveform to modify a carrier wave form. The modulating waveform is represented by the data we want to send. In this case, we have lumped bits together in a specific order dictated by a pre-set mapping scheme. It's arranged that way so that the points on the constellation map allow us to conveniently create the modulating waveform. In this case, I and Q can represent different axes of a coordinate system, and the graphed points are turned into waves. This information is then used to make two different waveforms of two orthogonal signals, which are combined to form a signal where the information can be easily extracted.

Modulating the type of modulation in PathWave System Design (SystemVue) is relatively straightforward, just modify the block itself. Choices range from I/Q, Amplitude/Phase, and Amplitude Frequency.

5.7 Analyzing the Results

Last, we want to see and test the output of our transmitter. To do this, it is preferable to use the 89600 VSA. The VSA is a multi-measurement signal analyzer that is incredibly useful for debugging and testing our current transmitter.

1. Attach a regular sink and a "VSA_89600_Sink" to Mod "Out" and then run a simulation.

   Wait a short period of time to allow the VSA and PathWave System Design (SystemVue) to

interact. When the VSA comes up:

2. Go into “MesaSetup” which will lead into “Measurement Type Vector” and then click “Digital Demod.”

   You should now see a whole bunch of messed symbols on an ill-fitting constellation map and a spectrum analyzer.

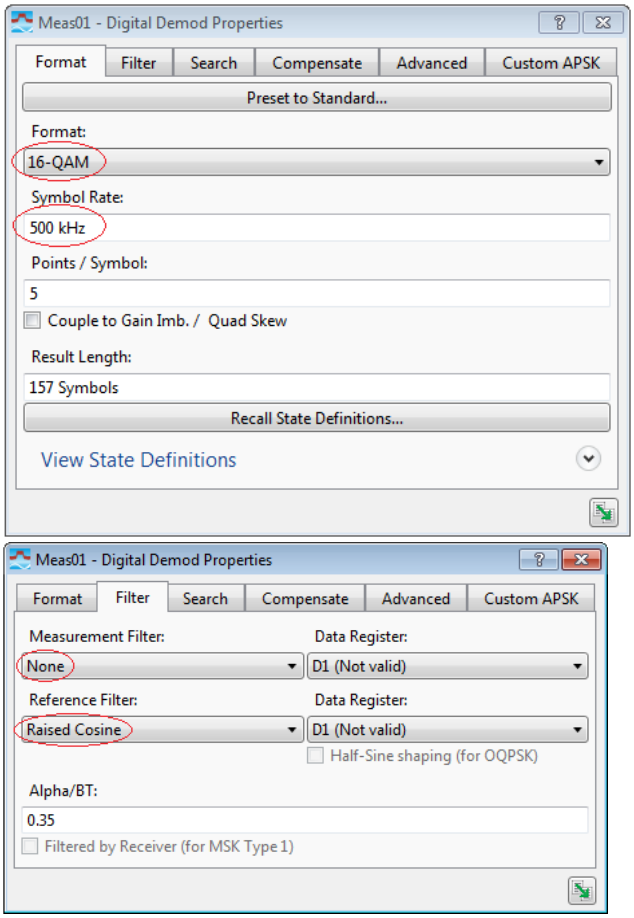


Figure 5.18. VSA software’s Digital Demod Properties window, with the circled areas indicating parameters that need to be modified.

3. Afterwards go to “Digital Demod Properties,” under Mesa Setup. Change the Format to 16-QAM and the Symbol Rate to 500 kHz. The changes are shown in Figure 5.18.
4. Then, click the “Filter” tab and select Measurement Filter “None.” Ensure the Reference Filter is on “Raised Cosine” and then stop the simulation and run it again.

If your signal looks like Figures 5.19-5.21, you probably have it right.

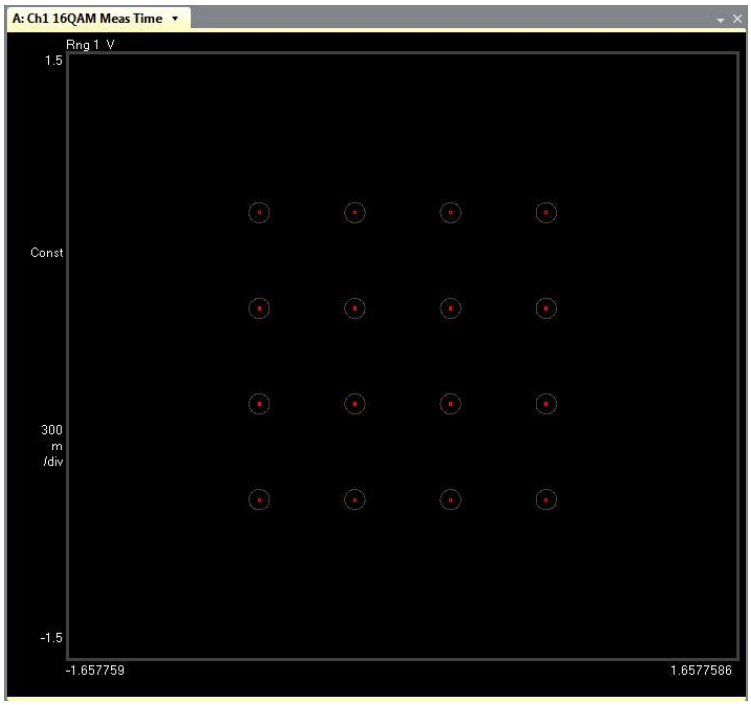


Figure 5.19. A VSA 89600 constellation map of the 16 QAM transmitter without the yellow tradjectory symbol information.

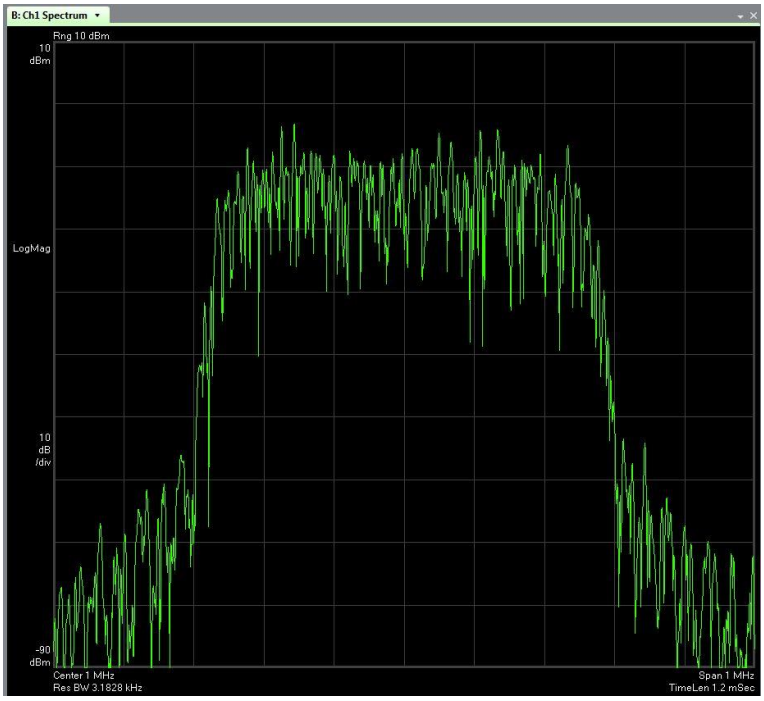


Figure 5.20. A VSA 89600 frequency spectrum of the 16 QAM signal.

Since we have theoritically created a perfect signal output without noise, check if your error is less than about 500 m%rms on average. You can find the EVM (error vector magnitude) In Figure 5.21.

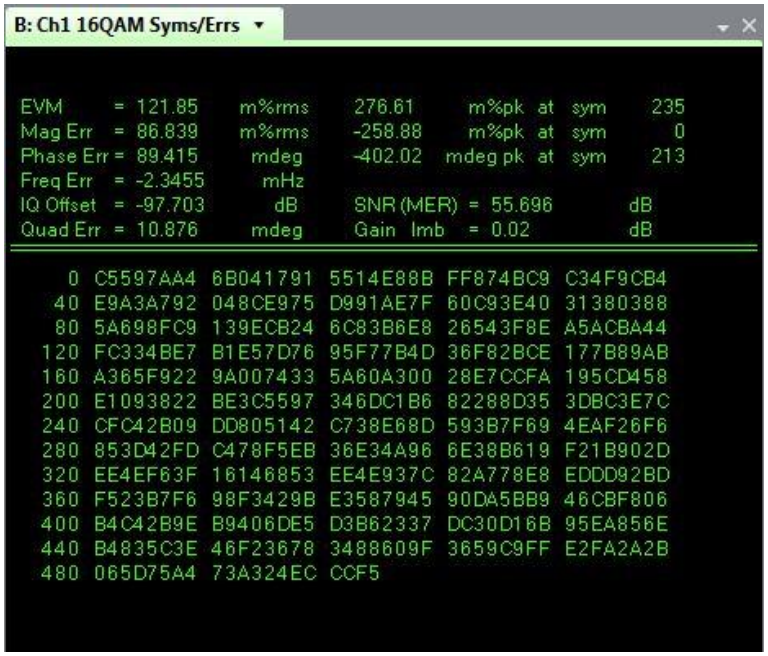


Figure 5.21. A VSA 89600 system error summary window.

When you are finished, your final model should look like that shown in Figure 5.22.

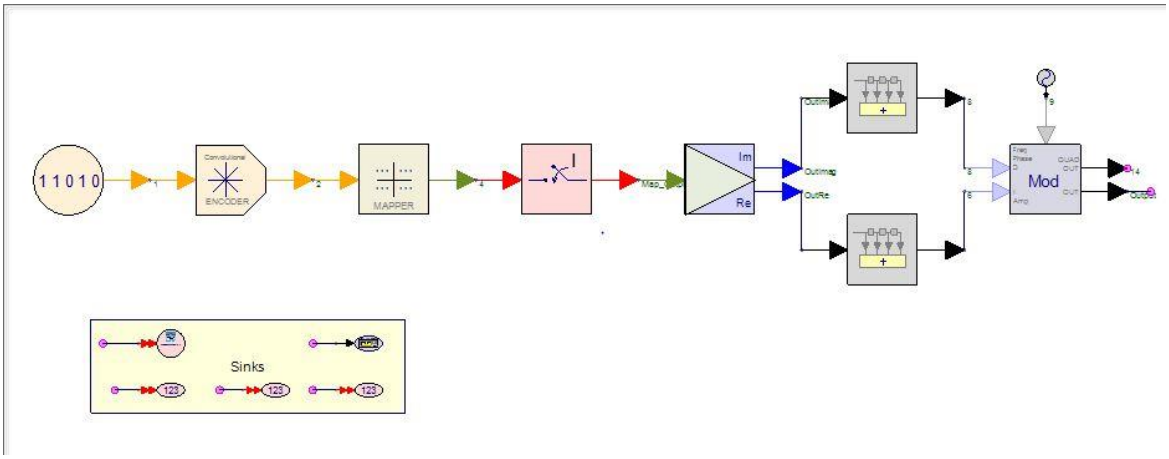


Figure 5.22. Simplified final schematic result.

Chapter 6: Receiver Overview

6.1 Receiver Basics

The receiver can be seen as the other half of a whole in digital communications. It takes an incoming signal and decodes the wave. The incoming (RF) signal is first down-converted to an Intermediate Frequency (IF) and demodulated. The ability to demodulate the signal is hampered by factors including atmospheric noise, competing signals and multipath or fading.

Generally, receiver signal processing involves the following stages:

1. Carrier frequency recovery (carrier lock).
2. Symbol clock recovery (symbol lock).
3. Signal decomposition to I and Q components.
4. Determining I and Q values for each symbol ("slicing").
5. Decoding and de-interleaving.
6. Expansion to original bit stream.
7. Digital-to-analog conversion, if required.

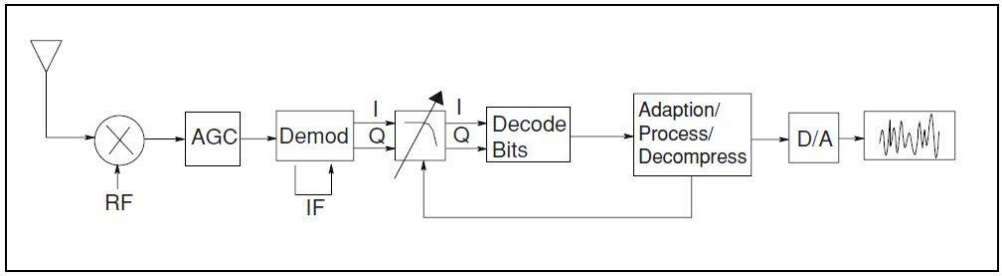


Figure 6.1. A diagram of the receiver.

These steps can be seen in Figure 6.1 above. In more and more systems, the signal starts out digital and stays digital. It is never analog in the sense of a continuous analog signal like audio. The main difference between the transmitter and receiver is the issue of carrier and clock (or symbol) recovery.

Both the symbol-clock frequency and phase (or timing) must be correct in the receiver to demodulate the bits successfully and recover the transmitted information. A symbol clock could be at the right frequency but at the wrong phase. If the symbol clock was aligned with the transitions between symbols rather than the symbols themselves, demodulation would be unsuccessful.

Symbol clocks are usually fixed in frequency and this frequency is accurately known by both the transmitter and receiver. The difficulty is aligning both in phase and timing. There are a variety of techniques and most systems employ two or more. If the signal amplitude varies during modulation, a receiver can measure the variations. The transmitter can send a specific synchronization signal or a predetermined bit sequence such as 10101010101010 to "train" the receiver's clock. In systems with a

pulsed carrier, the symbol clock can be aligned with the power turn-on of the carrier.

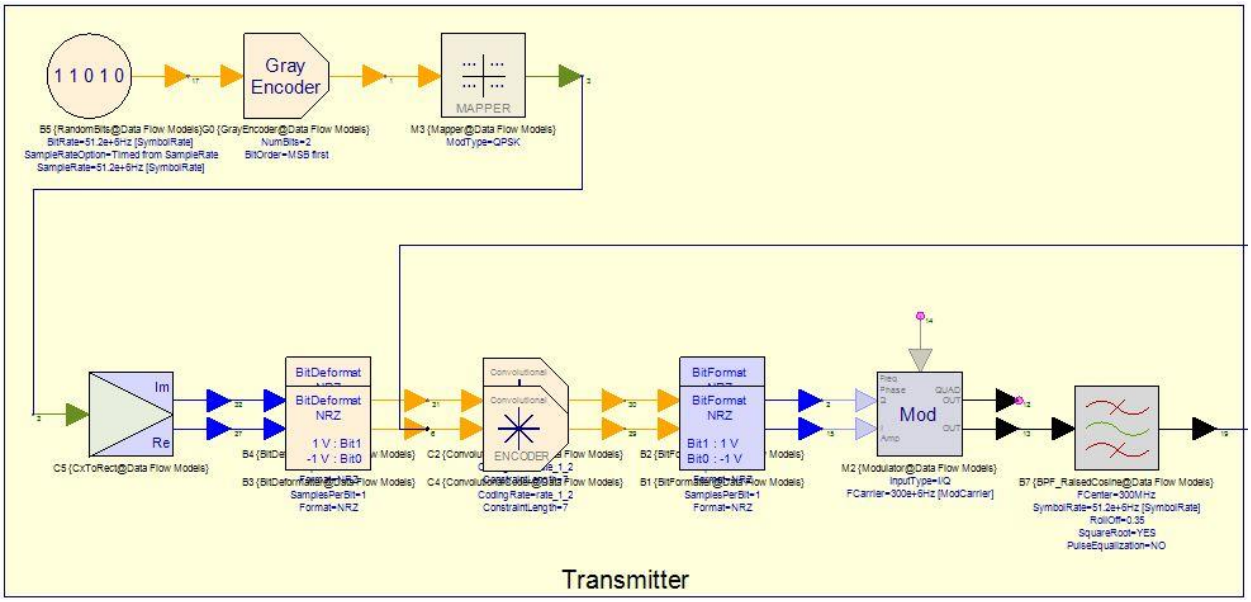
In the transmitter, it is known where the RF carrier and digital data clock are because they are being generated inside the transmitter itself. The receiver lacks this luxury. It can approximate where the carrier is, but the receiver itself has no phase or timing symbol clock information. A difficult task in receiver design is to create carrier and symbol-clock recovery algorithms.

It might be natural to assume from the Figure 6.1 that creating a receiver would be as easy as creating a sort of inverse transmitter. Both the transmitter and receiver tend to mirror each other in corresponding places regarding their inverse functions. In reality, the receiver side deals with more complex issues that keep the two parts different, this makes creating the opposite receiver very similar to a transmitter. A PathWave System Design (SystemVue) example, taken from the example QPSK_BER_Coded_Viterbi, is shown in Figure 6.2.

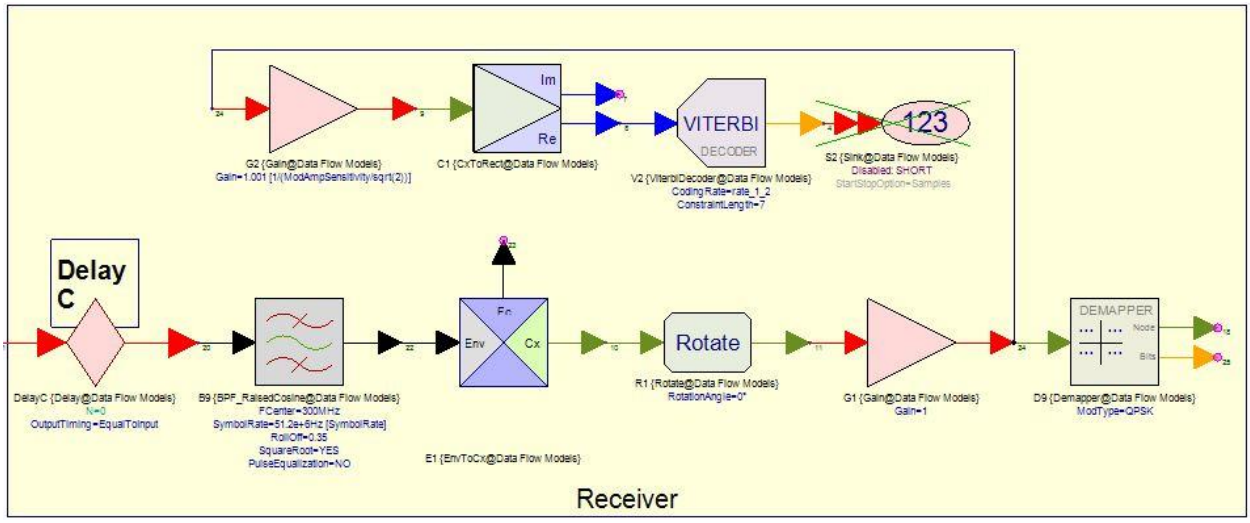
Reference: QPSK_BER_CodedViterbi

Can be found by going to:

1. Open Example Page.
2. Open the Comms Folder.
3. Open the BER Folder. The example is located here.



(a)



(b)

Figure 6.2. From schematic QPSK_Design. A shot of the transmitter (a) and a shot of the receiver (b).

As you can see, while there are differences when putting together both a transmitter and a receiver from basic parts, there are enough parallels involved that make creating one after the other redundant. Because of this, this chapter will be different from the previous one in that it won't take a step by step process to create a receiver. Instead, it will analyze a fully functional receiver block. This will help users familiarize themselves with PathWave System Design (SystemVue)'s more advanced functions and, at the same time, showcase a hassle-free alternative to creating a digital communication system from scratch. However, if you desire more creative freedom over your designed receiver, then Figure 6.2 can be used as a reference.

Reference: DigMod_Tx_Rx_AWGN

To find the reference example, go to the Open Example page:

1. Open the Baseband Verification folder.
2. Open the DigMod folder.
3. Open the Equalizer folder and it should be located here.

Note: The name DigMod_Tx_Rx_AWGN refers to a digital modulation example made primarily of a transmitter and receiver with additive white Gaussian noise (known as AWGN). AWGN represents a simplified set of conditions that affect the channel during transmission. A linear addition of white noise with a constant spectral density is introduced into the channel with a Gaussian distribution of amplitude.

This workspace will be the basis for our examples, illustrating the receiver and how it is modeled in PathWave System Design (SystemVue). First, in Figure 6.3 we have a communication system modeled in PathWave System Design (SystemVue).

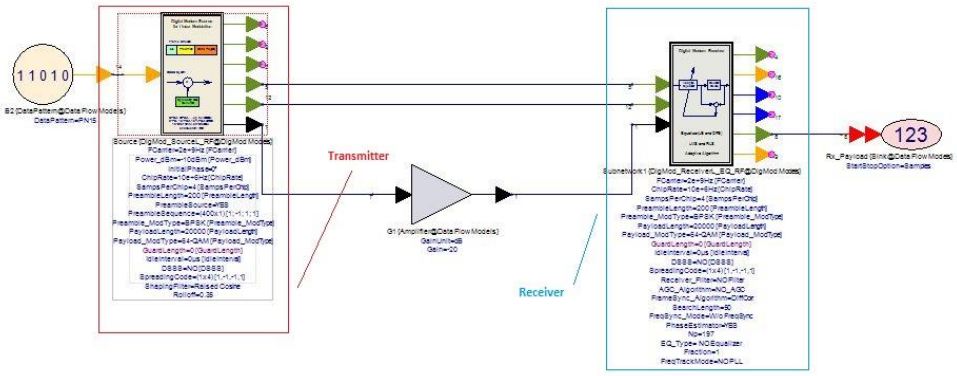


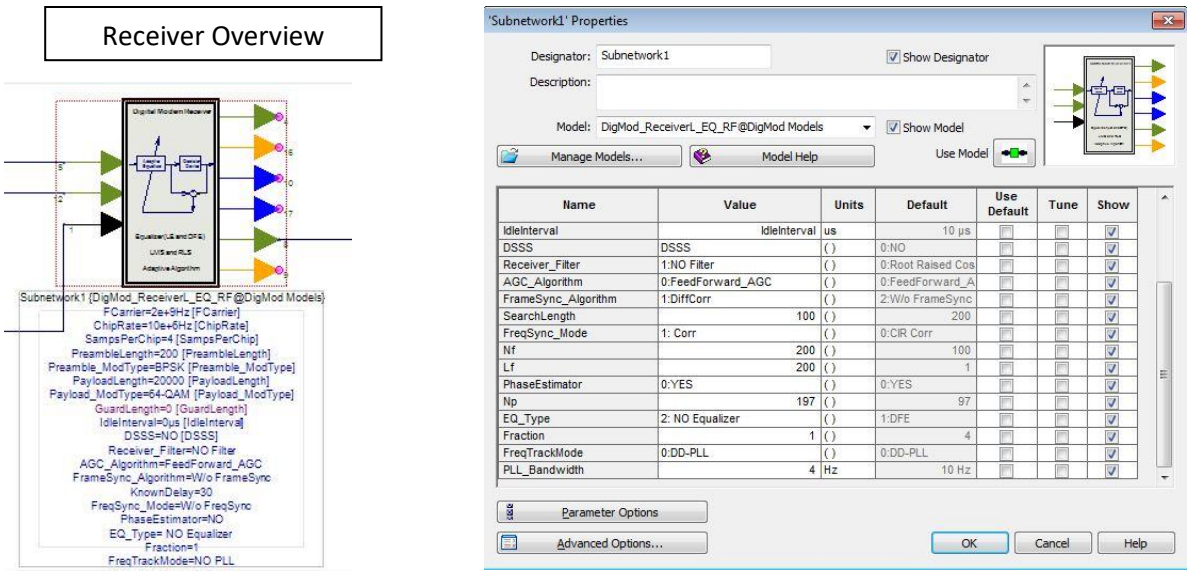
Figure 6.3. This is the schematic called DigMod_TxRx_wo_FrameSync, located in the DigMod_Tx_Rx_AWGN workspace example.

The schematic shown in Figure 6.3 is from the folder "1.1 WoFrameSyncEffect," and starts with an input of random bits and an output to a sink, with two giant blocks in-between. The block on the left is a full transmitter, and the block on the right is a full receiver. Each of these blocks are RF sources and receivers with linear modulation. Set with a number of parameters, the blocks are meant to be fully customizable. By themselves they serve as a sample model to mimic a digital communication system, which will be helpful to analyzing receiver parts. Later on, in the example DigMod_Tx_Rx_AWGN, there will be slightly different variations of this schematic, but functionally they'll all do the same thing.

The giant transmitter and receiver blocks used to create this schematic can be found under the DigMod Parts Library with the names DigMod_SourceL_EQ_RF and DigMod_ReceiverL_EQ_RF, respectively.

6.2 Receiver Subnetwork Overview

The focus of this chapter will revolve around the DigMod_ReceiverL_EQ_RF model. This block is a comprehensive receiver simulation block equipped with flexible parameters. The purpose of this section is to understand the receiver part and to effectively measure and test a communication system in PathWave System Design (SystemVue). The approach to the chapter will be different from the previous one. It will assume the user is competent in creating their models through the block and equation system and instead focuses on some of the advanced blocks PathWave System Design (SystemVue) has provided. It will also address and solve some of the problems one may face when they attempt to create a digital communication system of their own.



(a) **(b)**

Figure 6.4. The Receiver block diagram (a). The Receiver block Properties (b).

In Figure 6.4 (b) we can see DigMod_ReceiverL_EQ_RF's modifiable elements such as the modulation types (both payload and preamble), frame synchronization algorithms, phase estimators, and frequency synchronization modes. This block can also output various data streams like payload in bits, payload demodulated, frequency offset, and the synchronization index. We can go even further and examine how this process works. To do that, simply right click the block and go:

1. Open
2. Model/Subcircuit

From here you will see a subcircuit of the receiver (Figure 6.5). Notice how it matches up to some of the key ideas and characteristics of an ideal subcircuit.

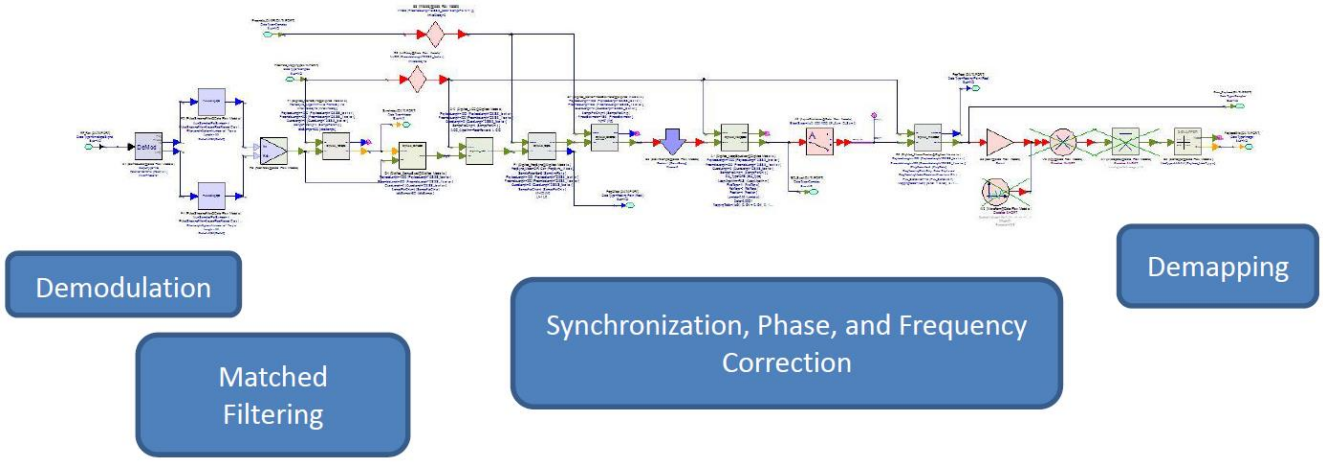


Figure 6.5. Shown here is the receiver subcircuit.

Note: If you wish to change anything in this example or anything inside the high-level source and receiver blocks themselves, do not mess with the core example version and save. Instead, create an empty schematic or subcircuit, copy the contents of the workspace/schematic/files over, and then use the new created copy as your test bed or new project.

6.3 Demodulator

Reference: Folder 1.1, WoFrameSyncEffect

Demodulation reverses the modulation process by extracting the embedded information from the carrier signal. There are a couple of ways to demodulate an incoming signal, but the most common way is to multiply the input signal by orthogonal wave functions.

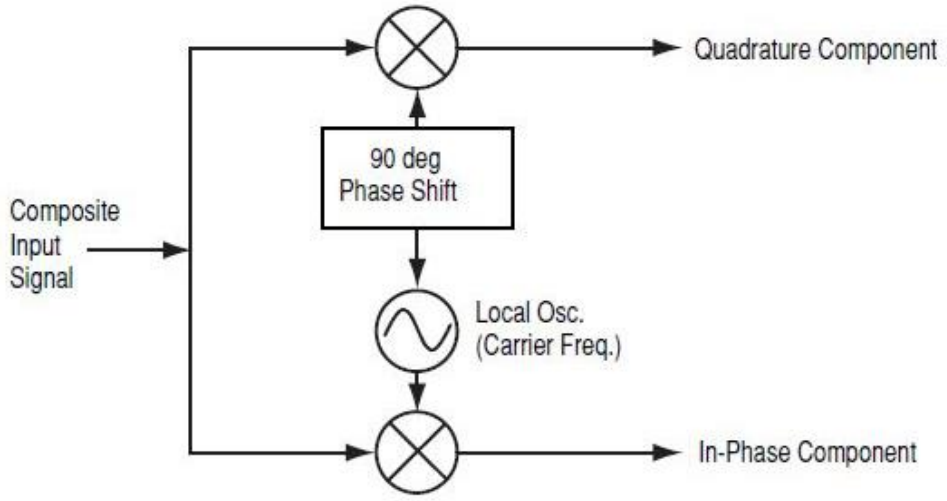


Figure 6.6. A graphical representation of a carrier signal about to be demodulated.

Quadrature amplitude demodulation is performed with two orthogonal oscillators as shown in Figure 6.6. When multiplying the carrier signal with the orthogonal oscillating signals, each orthogonal signal only generates half of the necessary information to extract the modulating signal. With this information from a single waveform we can extract both I and Q values, allowing us to put together the symbols and then subsequently the data itself. That means when demodulating, if the oscillator does not match phase with the received signal, the output has been affected by a variable gain. This will be explained further in section 6.6, “Phase and Timing Error Correction.”

To change the modulation and demodulation processes in this system, edit the blocks and change both the preamble and payload to have individual modulation and demodulation schemes (Figure 6.7).

PreambleLength	PreambleLength
PreambleSource	1:YES
PreambleSequence	Preamble(:)
Preamble_ModType	Preamble_ModType
PayloadLength	PayloadLength
Payload_ModType	Payload_ModType

(a)

PreambleLength	PreambleLength
Preamble_ModType	Preamble_ModType
PayloadLength	PayloadLength
Payload_ModType	Payload_ModType

(b)

Figure 6.7. Source block location for preamble settings (a). Receiver block location for preamble settings (b).

When demodulating the signal, the most optimal conditions usually have the carrier frequency match the oscillating frequency of the orthogonal signals, so when editing both blocks, make sure they match. The two demodulated I/Q parts then arrive at the filtering process.

6.4 Matched Filtering

When filtering for the transmitter, the objective is to limit the signal sent to a narrow frequency band that will not interfere with other sent signals and random noise by the outside world. However, even when sent along a narrow frequency, when the signal reaches the receiver it has usually collects white noise. A filter on the receiver side is usually there to filter the noise out we have added on through the channel. While helpful, it's not entirely necessary and should be added on in a case by case basis. This receiver's filter is usually a based on the source's filter and designed to be matched, thereby eliminating errors in data and excess noise.

In this simulation, we can see that the transmitter uses a raised cosine filter to pulse shape the output signal and that the receiver is not using a filter at all. This is because the cosine filter is a bandpass filter. It will only read in the right values at a certain frequency to be modulated. Because of this, a matching filter is not needed for the receiver side. However, if we want, we can change this. Simply refer to Figure 6.8.

DSSS	0:NO
Receiver_Filter	0:Root Raised Cosine
Rolloff	0.35

Figure 6.8. Settings area to change the filter.

6.5 Synchronization

When a signal is sent wirelessly across a distance, it's inevitable that a certain amount of delay will be introduced. This delay can be difficult to deal with because it changes both symbol timing and carrier phase. To fix this, we can implement a method to minimize the clock offset. However, when measuring the delay caused by distance/mediums, we have to measure the symbol timing and carrier phase separately. This is due to the different magnitudes of frequency they both operate at. First, let's look at symbol timing errors.

Our example will use the basic transmitter and receiver system set up in the introduction. It'll first be set

up without any frame synchronization (Figure 6.9).

AGC_Algorithm	0:FeedForward_AGC
FrameSync_Algorithm	2:W/o FrameSync
KnownDelay	80

Figure 6.9. Parameters that turn on frame synchronization shown here. Currently set to off.

Let's see the effects of the symbol timing errors in action. To accurately view this, first we must have some way of examining the incoming signal. To do this, we'll use a constellation diagram, which in this case, is set at 64 QAM. A graph of the constellation without frame synchronization is shown in Figure 6.10.

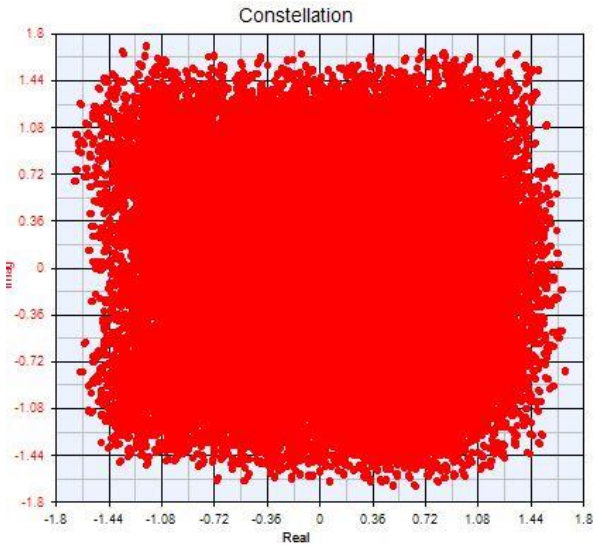


Figure 6.10. Symbol map without frame synchronization.

What you see in Figure 6.10 is a hodgepodge of dots. What it means in this case, is that the frame structure is being mistimed. Ordinarily if this signal was being read in correctly, we would see clumps of dots by the constellation points representing points of data; however, that requires the receiver to read the symbols at consistent times which sync up to the symbol frequency rate. Without the sampling rate mimicking the frame structure frequency at the same time, we sample points all along the carrier wave giving us values on and in-between the constellation points.

Note: In this case, this smearing of points means that the symbol rate is being mistimed, but this shouldn't be assumed when you encounter a constellation diagram with this messy data characteristic in the future. An issue like this can be attributed to one of (or a combination of) three things. The issue can either be caused by a signal rate mistiming, a difference in the receiver filter, or/and an unlocked carrier signal.

To fix this, we can simply turn on a frame synchronization algorithm. To do this in the PathWave System Design (SystemVue) block, just edit and change FrameSync_Algorithm to "1:DiffCorr" as shown in Figure 6.11.

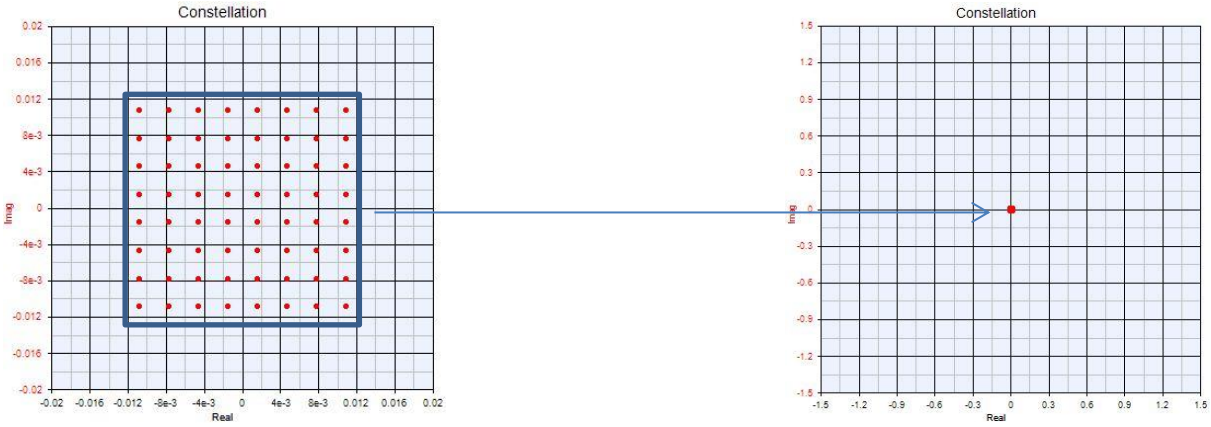
AGC_Algorithm	0:FeedForward_AGC
FrameSync_Algorithm	1:DiffCorr
SearchLength	100

Figure 6.11. Shows the change in the frame synchronization algorithm settings to differential correlation.

Note: Another way to modify blocks is to double click the blue parameter text below the name and description of the block diagram.

Reference: Folder 1.2, WoAGC

When we turn on differential correlation, and the timing fixes itself, we can see a drastic improvement. Figure 6.12 (a) shows that now the data is being read in correctly, matching the constellation structure of 64 QAM. However, if scaled out, Figure 6.12 (b) shows that the points themselves are not spaced correctly in magnitude.



(a) (b)

Figure 6.12. This Figure showcases a difference in scaling on the WoAGC simulation. The first graph shows the constellation when the simulation is run (a). The second is a comparison graph of the constellation with the proper scale (b).

This is because when reading in the baseband and RF signal filters from both the receiver and transmitter a gain occurs that cuts the power of the signal. Gain is also lost when traveling through any medium, be it wire or air.

To fix this, we must turn on or change the AGC (Automatic Gain Control) Algorithm which will restore the power values to their original state and allow them to occupy the correct regions on the constellation.

The change step can be seen in Figure 6.13.

Receiver_Filter	1:NO Filter
AGC_Algorithm	0:FeedForward_AGC
FrameSync_Algorithm	1:DiffCorr

Figure 6.13. Example showing when to turn on the AGC algorithm.

The resulting effect is shown in Figure 6.14, which showcases a boosted and normalized constellation map in terms of magnitude.

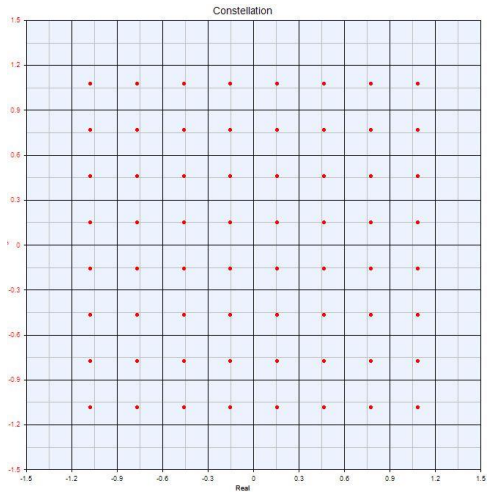


Figure 6.14. Constellation graph with AGC on.

6.6. Phase and Timing Error Correction

Reference: Folder 2.1, CarrierPhaseEffect

A popular correction technique is called PLL (Phase-Locked Loop). PLLs use a Loop filter that feeds back into a voltage-controlled oscillator. It works based on the principle:

$$\int_{-\infty}^{\infty} \sin[\theta(t)] \cos[\theta(t)]\, dt = 0 \qquad \text{(E.6.1)}$$

With Equation 6.1, if the phase error is any amount that's not orthogonal to the original signal, then we can estimate what the phase error is.

Sending a signal from the transmitter to the receiver adds a phase shift in the signal, which applies to the constellation diagram.

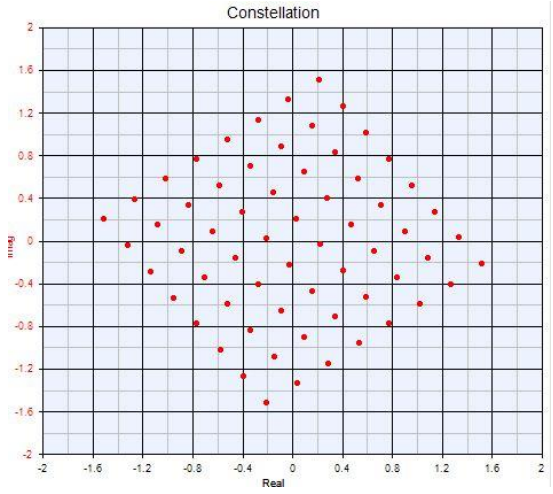


Figure 6.15. A 64 QAM signal rotated 135 degrees.

We can see this problem illustrated clearly in Figure 6.15. To fix this, we simply enable the Phase Estimator to Yes (Figure 6.16).

FreqSync_Mode	3:W/o FreqSync
PhaseEstimator	0:YES
Np	197

Figure 6.16. Shows where to change the phase estimator.

Reference: Folder 2.2, CarrierPhaseEstimator

Then, we receive the fixed constellation shown in Figure 6.17 with the corrected phase shifts.

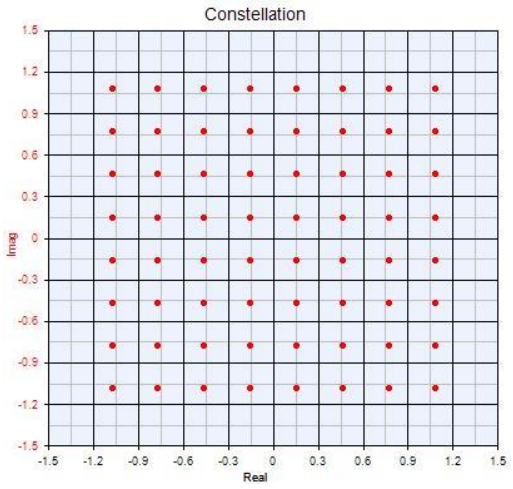


Figure 6.17. The corrected constellation previously phase shifted.

6.6 Frequency Error Correction

Reference: Folder 3.1, CarrierFrequencyOffsetEffect

When signals are sampled at a rate that doesn't match up to the carrier frequency, we call this frequency offset. The magnitude of inaccuracy and whether the sampling rate is over or under the carrier frequency usually determines the speed of the travelling points, and the direction in which they move. An example of frequency offset is shown in Figure 6.18, with the carrier frequency offset by 50 Hz.

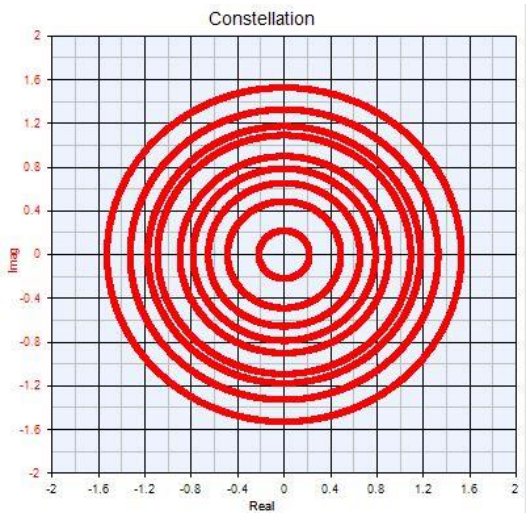


Figure 6.18. The effects of frequency offset shown in a constellation map.

To correct for this frequency shift, we can enable a carrier frequency synchronization algorithm (Figure 6.19). Setting the FreqSync_Mode to Corr (Correlation) corrects the signal and produces a diagram resembling a constellation but will result in a small frequency shift due to residual error. Another option is to change FreqSync_Mode to CIR_Corr (Carrier Impulse Response Correlation), which will clean up the error and produce a better graph.

Reference: Folder 3.2, CarrierFrequencySync

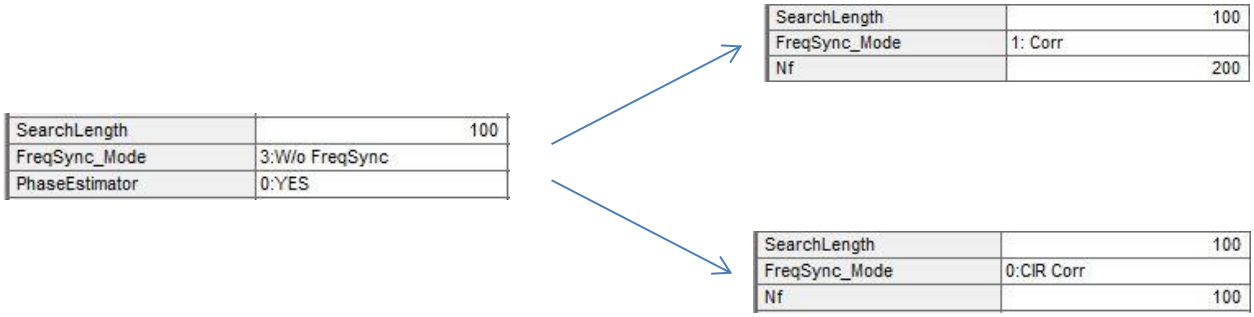
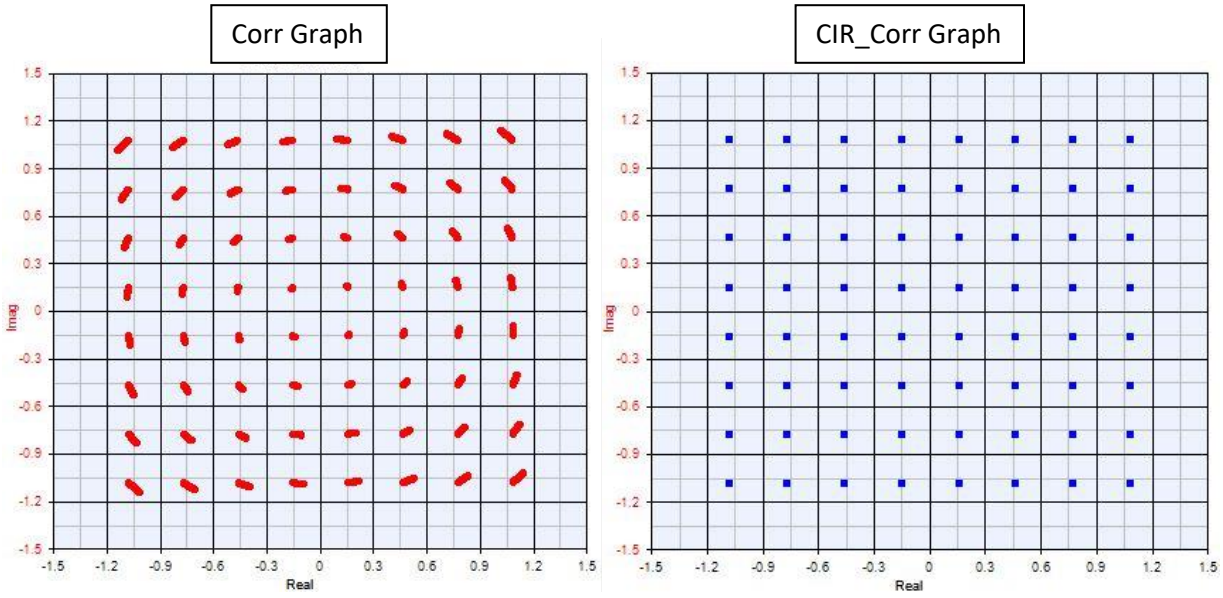


Figure 6.19. Different selectable options for the Frequency Synchronization Mode

Which leads to:



(a) (b)

Figure 6.20. The different resulting graphs when trying to fix the frequency offset. Correlation's graph is shown under (a). CIR Correlation's graph is shown under (b).

When looking at Figure 6.20, notice how the Correlation (a) offset compares to the Carrier Impulse Response (b). The Corr graph shows the constellation gradually shifting clockwise until it arrives to where the CIR_Corr is located; this is because the Corr method requires time for the feedback system to correct the initial frequency measurements as shown in Figure 6.21.

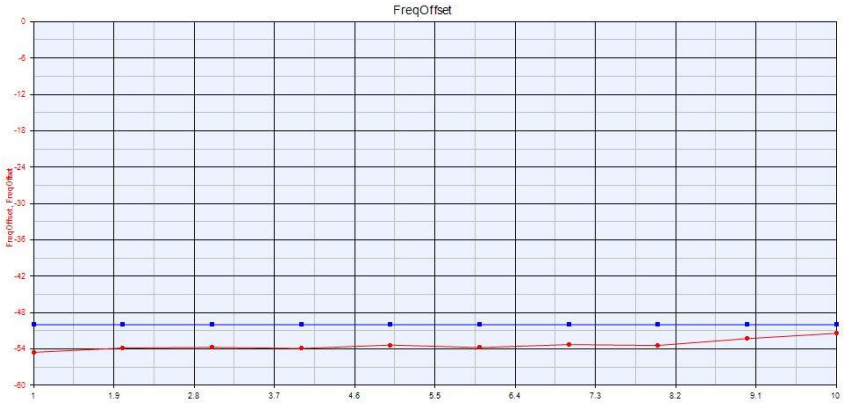


Figure 6.21. Offsets made by the receiver to counter the transmitted frequency offset.

Reference: Folder 3.3, CarrierFrequencyTracking_PLL

Next, we'll be showcasing what happens when the frequency offset is small. Instead we can just turn on the PLL offset, (which also uses PLLs) to calculate frequency offset (Figure 6.22).

Fraction	1
FreqTrackMode	0:DD-PLL
PLL_Bandwidth	60

Figure 6.22. Shows where to set the starting frequency and the PLL_Bandwidth option.

The PLL offset will take a variable amount of time to sync to the right frequency depending on how far off the signal offset is from the demodulation wave. This method will also remove the carrier offset, as shown in Figure 6.23.

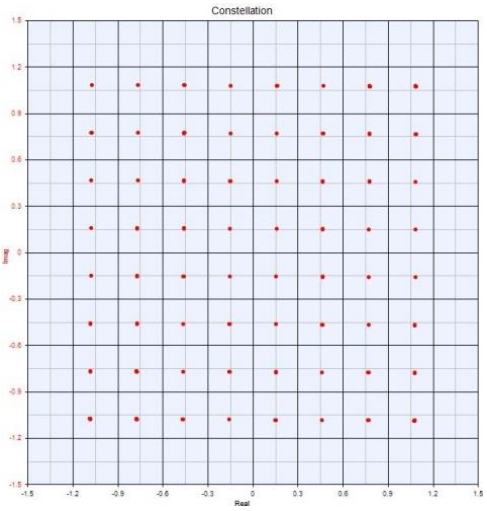


Figure 6.23. The effects of the PLL on a signal with frequency offset.

The error correction behavior is shown in Figure 6.24.

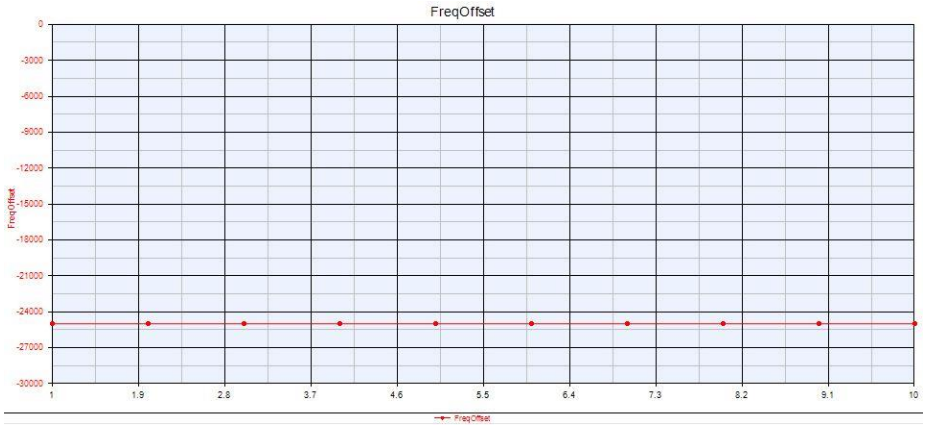
Figure 6.24. A frequency tracking graph of the PLL adjusting to the offset over time.

Figure 6.24 shows the phase being constantly adjusted by a feedback system over a number of samples until it finds the correct frequency.

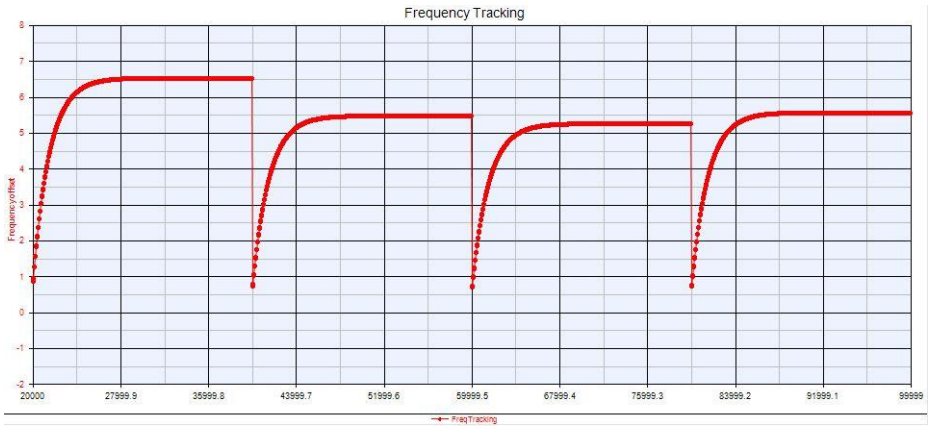
Reference: Folder 3.4, CarrierFreqSync_and_Tracking

Now what happens when the offset is large? The PLL offset works well when you know the frequency, but without it you tend to have small traces of frequency offset. This stems from the fact that the algorithm cannot approach the offset value quickly enough. The same property holds true for correlation. To account for this, you can use a combination of PLL Offset and CIR_Corr/Corr to fix the

error. For example, a large frequency offset of 24990.9 would usually set one or the other back individually, but when combined, we can see that PLL offset and correlation helps the receiver to approach the offset value. The correlation frequency synchronization method acts as coarse detection, setting the frequency offset near its actual value. The PLL offset, which uses frequency tracking, is more of a fine synchronization method that brings and matches the frequency offset when it is close or near to the offset value. See Figure 6.25.



(a)



(b)

Figure 6.25. This graph shows the frequency offset value taken by the Correlation method (a). This graph shows the frequency tracking of the PLL in tandem, making smaller corrections (b).

This result of combining carrier frequency synchronization with frequency tracking is shown in Figure 6.26.

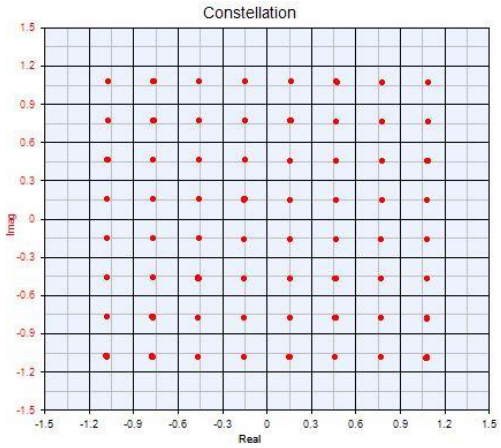


Figure 6.26. Resulting constellation from the PLL and correlation corrections in tandem.

6.7 Channel Estimation

Channel estimation is based on gathering knowledge on the errors through the medium in which the signal travels. These effects include scattering, fading and power decay with distance. To use channel estimation, you need to know the channel characteristics themselves. It starts with initial channel estimation and the tracking of channel changes.

The different channel estimation techniques are blind, semi-blind and training based.

6.8 Demapping

The rest of the receiver is similar in design to the transmitter. For the most part, mapping is taken care of with demodulating when setting the modulation types. There are, however, different ways to demap a signal. A basic way is called slicing. It's where you set a threshold for a level or certain level of voltages and assign bits or symbols to each of those thresholds. Then, when read in, you get the boundaries for the binary data of the sent signal.

6.9 Code Correction

Similar to the convolution coder posed in the transmitter chapter, the R-S (Reed Solomon) coding also uses a series of registers. It works by acting similarly with the convolution encoder, but instead of reading in from one polynomial it reads in from two. The R-S error correction codes are non-binary, cyclic error-correcting codes. They are a systematic method for creating codes capable of detecting multiple random symbol errors.

In addition, these blocks can change the frame structure when editing the Payload_Length and Preamble_Length (Figure 6.27).

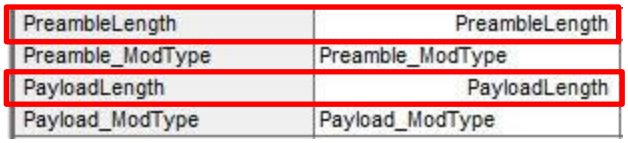


Figure 6.27. Location to change the payload length and preamble length.

Chapter 7: System Level Modeling

7.1 BER

Now that we've established how to make the digital communication models, it's time to start testing and examining how accurately PathWave System Design (SystemVue) can simulate and test the model created. We can start with BER. BER stands for Bit Error Rate and is a unit-less percentage that tells us how error prone our communication system is.

Reference: DigMod_Tx_Rx_AWGN, Folder 4. DigMod_TxRx_NoISI_BER

To calculate the BER in PathWave System Design (SystemVue), we usually have one of two methods. One way to check BER is to use the equation's functionality. To test a theoretical model on error based on frequency strength (in dBs) we need to set up a few equations. The first general BER equation, Equation 7.1, is specifically for QPSK modulation with AWGN.

$$BER = \frac{1}{2} erfc(\sqrt{E_b/N_0}) \qquad \text{E.7.1}$$

However, for most BER cases, Equation 2 is only a basic component of an overall BER calculation. Therefore, when someone uses different communication schemes, we must change or add to the BER calculation equation. With PathWave System Design (SystemVue), we don't have to worry about that, instead we can calculate the BER through the schematic and use blocks to create BER graphs. These data sets are created through the BER_FER block (Bit and Frame Error Rate) shown in Figure 7.1.

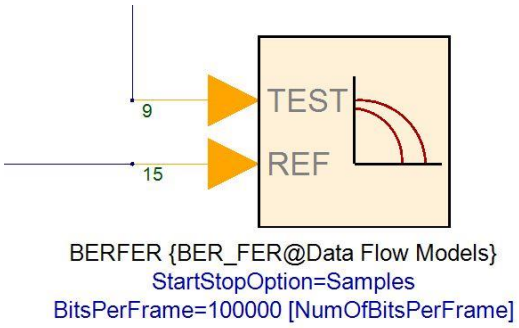


Figure 7.1. The BER block: line 9 is the receiver output and line 15 is the input into the transmitter.

The following example from the "DigMod_TxRx" takes the BER and compares it to the theoretical limits of the modulation type. It sets up a set of equations in an equations page filed under the workspace Theoretical_QAM_BER (Figure 7.2).

```
EbNodB_1024=-6:2:30
EbNo=10.^(EbNodB_1024/10);
%1024-QAM BER
k=10;
M=2^k;
x=sqrt(3*k*EbNo/(M-1));
QAM_1024_Pb=(4/k)*(1-1/sqrt(M))*(1/2)*erfc(x/sqrt(2));

EbNodB=-6:2:24
EbNo=10.^(EbNodB/10);

%256-QAM BER
k=8;
M=2^k;
x=sqrt(3*k*EbNo/(M-1));
QAM_256_Pb=(4/k)*(1-1/sqrt(M))*(1/2)*erfc(x/sqrt(2));


%64-QAM BER
k=6;
M=2^k;
x=sqrt(3*k*EbNo/(M-1));
QAM_64_Pb=(4/k)*(1-1/sqrt(M))*(1/2)*erfc(x/sqrt(2));

%16-QAM BER
k=4;
M=2^k;
x=sqrt(3*k*EbNo/(M-1));
QAM_16_Pb=(4/k)*(1-1/sqrt(M))*(1/2)*erfc(x/sqrt(2));

%QPSK (4-QAM) BER
k=2;
M=2^k;
x=sqrt(3*k*EbNo/(M-1));
QAM_4_Pb=(4/k)*(1-1/sqrt(M))*(1/2)*erfc(x/sqrt(2));

```

Figure 7.2. The Theoretical_QAM_BER equation page showing the calculations for theoretical BER.

From these variables you can plot both BER versus EbN0 on the same graph and compare how close the simulated QAM communication design is to be an optimized system. In the example itself, this graph can be seen as the QAM_BER_vs_Ebn0.

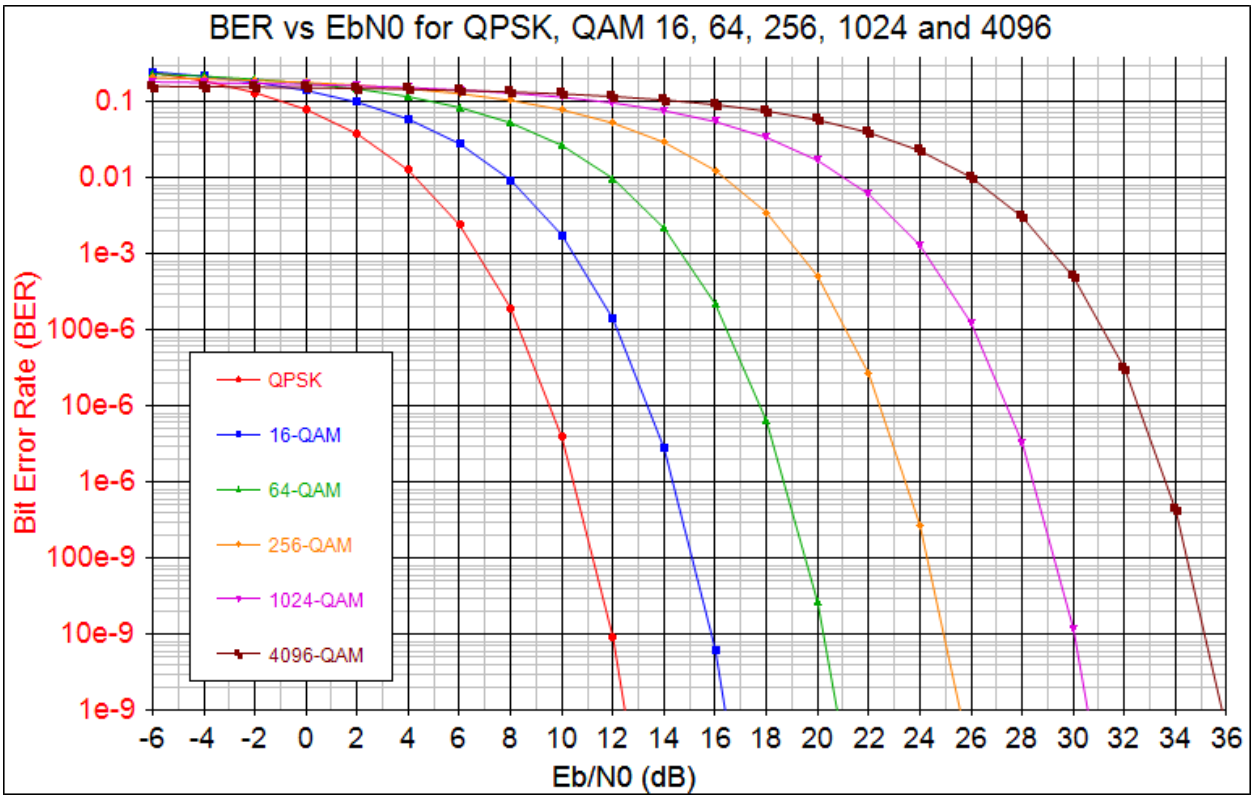


Figure 7.3. The theoretical and measured BER rates graphed together.

Figure 7.3 shows the BERs matched up against each other and that the theoretical limits and simulated schematics results are extremely close. This shows that the receiver block provided by PathWave System Design (SystemVue) produces results close to an optimized receiver and, therefore, can be used as a reference. With a receiver already so close to theoretical limits we've shown that in most cases there's no need to use equations to measure the theoretical BER rates, and that our reference receiver will mimic its results.

Note: If you wish to implement BER analysis in your PathWave System Design (SystemVue) example, here are the steps:

1. Before you can plot the data, you need to make sure you have a variable that can be analyzed. This is called the tuning variable, and it allows you to use something called a sweep evaluation to measure the BER at different gains. Go to your schematic page and create an equations page. In this page, set the following code:

   EbN0 = ?0; %Tuning variable
   Power_dBm = x; %Transmit power in dBm
   BitsPerSymbol = x;
   SamplesPerChip = x; %Or samples per symbol
   Chiprate = x; %Symbol rate
   NDensity = Power_dBm - 10*log10(ChipRate) - 10*log10(SampsPerChip) -

10*log10(BitsPerSymbol) - EbN0;

The x's represent places where you can input values based on the system you're trying to create. EbN0 is denoted as the tuning variable with the question mark shown right in front of the value zero.

2. Now we can create a Sweep evaluation. This is done by going into the workspace tree and New Item -> Evaluations -> Add Sweep.
3. Open the Sweep page. Here we have a couple of areas to modify. First, set Analysis to Sweep as your analysis you wish to run -> go to Parameter to Sweep and set it as the tunable variable -> change the start and stop range to whatever values you wish to measure the BER in between -> choose sweep (another way of saying step size/incrementing). In this example, we'll go over Linear: Step Size -> then hit "Calculate Now."
4. Going into the data set created when running the Sweep. Right-click BERFER_BER -> Add to Graph -> New Graph Series Wizard -> Y vs. X Graph -> "Parameters_EbN0_Swp_BERFER_BER_Index" set to X and BERFER_BER set to Y.

7.2 Cross-Domain Simulation

Note: To use cross-domain analysis you need the W1719 RF Design Kit. The example in this section uses both the licenses for the W1719 RF Design Kit and the W1910 3GPP LTE Baseband Verification Library.

Creating any part of a digital communications system requires both RF and baseband expertise. However, these two fields are different from one another and are traditionally know as separate regions in the EE workspace. RF deals with a world set in the frequency domain, a world filled with analog filters and amplifiers. Digital design is oriented in the time domain, where sampling rates and modulation are key.

It is often time consuming and costly to manage RF and baseband at the same time. When prototyping a receiver, engineers will often have to build their product from the ground up. The process usually takes months, with a guarantee that design specifications will change on the fly. These changes cause engineers to sink time and money into creating new iterations of their original design. And, because communication systems are both RF and digital design intensive, one small change in development from either division can cause a mess in the other. In other suites (or even PathWave System Design (SystemVue) itself), you could use Mathlang/MATLAB equations to model device behavior, but often these equations are limited and cannot precisely describe how an RF part would act.

This is where PathWave System Design (SystemVue) comes in. The software works well as a top-level design suite, with the capability to bridge the RF domain and the digital domain together in a process called cross-domain simulation. While PathWave System Design (SystemVue) originates from the baseband process, it can model RF behavior and create an overarching simulation that can imitate both sides of the analog and digital domains.

The capability to test results for both the digital and analog realms allows RF engineers to start creating

communication prototypes without having to worry about building circuits from the ground up. With a suite like this, you can easily cut down on development cost and time. For example, if your digital specifications change, you can quickly change the RF specifications without having to redesign your board from the ground up. This also means you have precise simulations that can be shown to customers, suppliers and competitors alike. An example of a cross-domain simulation will be shown in the following example.

Reference: 3GPP_LTE_DL_SISO_BER_RF_Link

To reach this example:

1. Head to Open Example.
2. Open the RF Architecture Design folder.

Now open "LTE_FDD_DL_AWGN_BER." It should look similar to Figure 7.4.

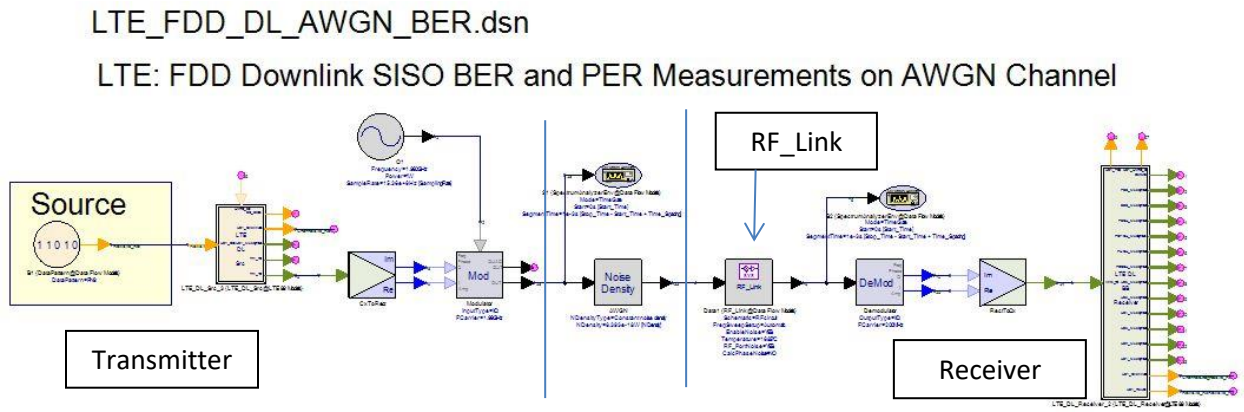


Figure 7.4. The schematic diagram LTE_FDD_DL_AWGN_BER.

Figure 7.4 is similar to the previously explained transmitters and receivers. The LTE blocks (blocks with over two outputs) can be seen as mapping/demapping and channel coding/decoding blocks, respectively. Everything else should look relatively standard. We have denoted the transmitter and receiver side by two lines going down the diagram. In between is the Noise Density block that simulates the medium through which the signal passes. The block of focus is called RF_Link, or the pink block at the beginning of the receiver.

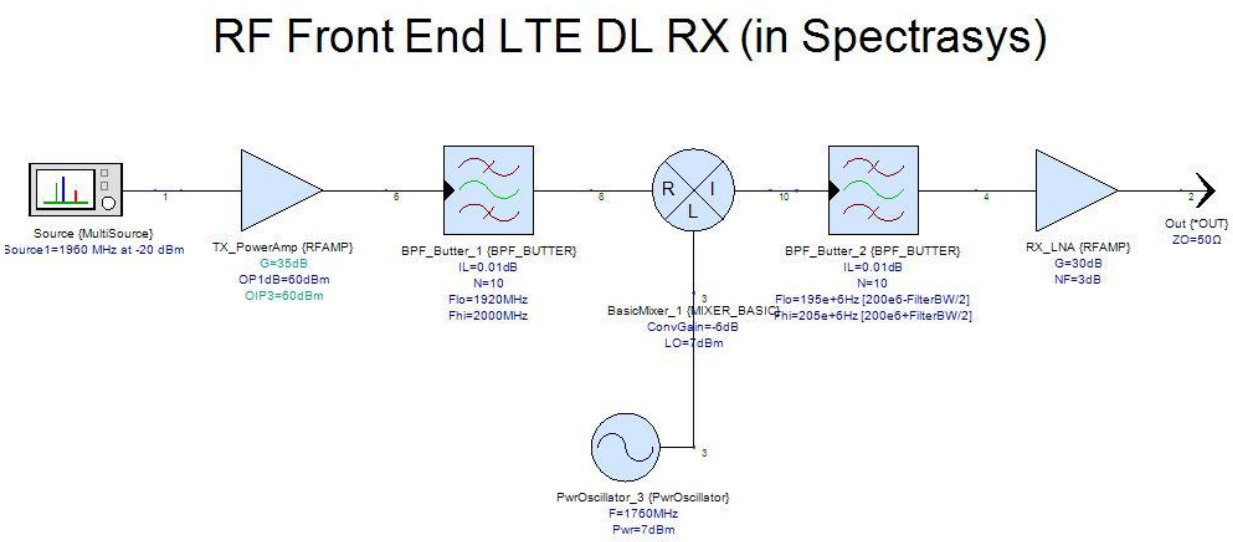


Figure 7.5. An example of Spectrasys design. This particular schematic referenced by the RF_Link block in the 3GPP_LTE_DL_SISO_BER_RF_Link example.

First, we start by creating a Spectrasys design (Figure 7.5). A Spectrasys design is a schematic filled with RF design characteristics. The RF_Link part then refers to the schematic like a sub-circuit. An example of this is the schematic named RFcircuit which can be found in the "RFBlocks" folder. Once the schematic is complete, the RF_Link block can be set to refer back to the Spectrasys design in the Properties area. Depending on your PathWave System Design (SystemVue) version you may need to specify which source is the input part and what port is the output part. This sets up the RF System analysis.

With this, you can set up the RF System analysis which implements a behavioral description of an RF system. Here you can refer the system analysis back to the Spectrasys design, enabling you to create a data set for the RF system in the frequency domain.

Finally, the RF_Link block located in the main high-level schematic can be analyzed in a normal Data Flow Simulator. This allows you to simulate low-level circuits in high-level behavior models. With this ability you can analyze your design topology and integrate whatever designs you want, all without having to build an RF circuit from scratch.

7.3 HDL Generation

Note: You will need the W1717 Hardware Design Kit in order to enable FPGA hardware design.

As an encompassing software suite, we would be remiss if we didn't include some form of Hardware Description Language (HDL) generation for FPGAs. With PathWave System Design (SystemVue) comes the W1717 Hardware Design Kit that allows you to create HDL code to simulate prototype circuits. But before we can create the HDL, we'll need to reconstruct the top-level system design models into a more DSP friendly format. In order to understand why, it's important to examine the differences between fixed- and floating-point numbers.

These two numeric formats differ in how bits store the values of numbers. Typically, a fixed form is the traditional bit form. It represents an exact number. Floating point format is set up differently. Instead, in the total bit word, a large part of the bits is used to set integer values or a base number, while the other bits are an exponent. This expands the original number range and allows for more precision when expressing any number.

While the floating-point format is much more robust, it has limitations that prevent it from being the standard on DSP processors or FPGAs. Its ability to encompass a great range of numbers increases the complexity of designing circuits for each device. Increased number size means an increase in memory usage, power consumption and size. Fixed point hardware is much less complicated then floating point hardware and draws less power. Since the circuits become more complicated and draw more power, the cost of manufacturing floating point hardware goes up. All these factors make fixed point hardware much more appealing to mobile devices that need to be small, light and relatively cheap.

So why did we even decide to use floating point tools to simulate the telecommunications model in first place? Fixed point verification is notoriously difficult. Fixed point designing is much more suited towards the lower level building. The result of this means that most telecommunications hardware is designed at the top level using floating point form but created in the lower level from fixed point form. And, because both are at their core different number schemes, they use different algorithms; therefore, generating HDL code for the fixed-point scheme requires a specific HDL program.

Reference: QPSK_transceiver_with_symbol_timing_recovery

To reach the example:

1. Go to Open Example.
2. Open the Hardware Design folder.
3. Open the CodeGeneration folder.
4. Open the QPSK_transceiver_with_symbol_timing_recovery folder.

In PathWave System Design (SystemVue), HDL simply involves using the hardware design library. This library contains copies of many of the algorithm blocks in fixed point form. They're denoted by the color pink and the "Fxp" in their name. Simply create an additional schematic or sub-network model and start placing the Fxp blocks down to start designing. PathWave System Design (SystemVue) greatly simplifies the HDL generation process and quickens coding by converting many high-level functions and circuits into optimized and efficient HDL code. This, in turn, cuts down on development time through the graphical interface.

An example of a fixed-point model is shown in Figure 7.6.

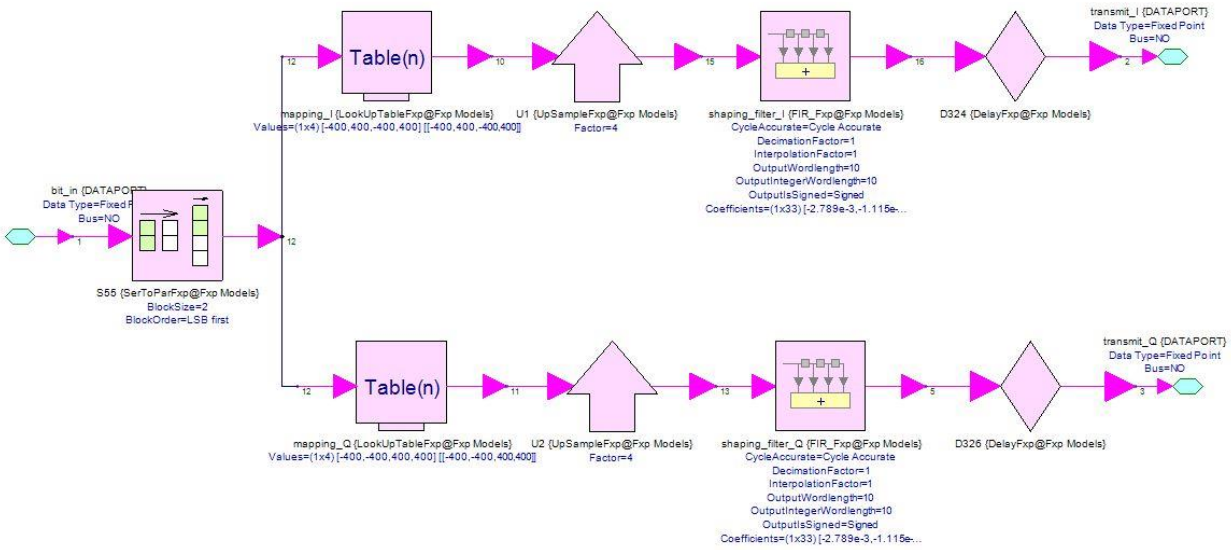


Figure 7.6. The qpsk_mod subcircuit under the Tx folder from the QPSK_transceiver_with_symbol_timing_recovery example.

Notice how in Figure 7.6 the different blocks have algorithm design equivalents to make the transition from designing in the algorithm simulation HDL implementation smooth.

To generate HDL code:

1. Go to the workspace tree and go to Code Generators (located under the new document creation).
2. Add HDL Code Generator. You will be brought to a window like that in Figure 7.7.

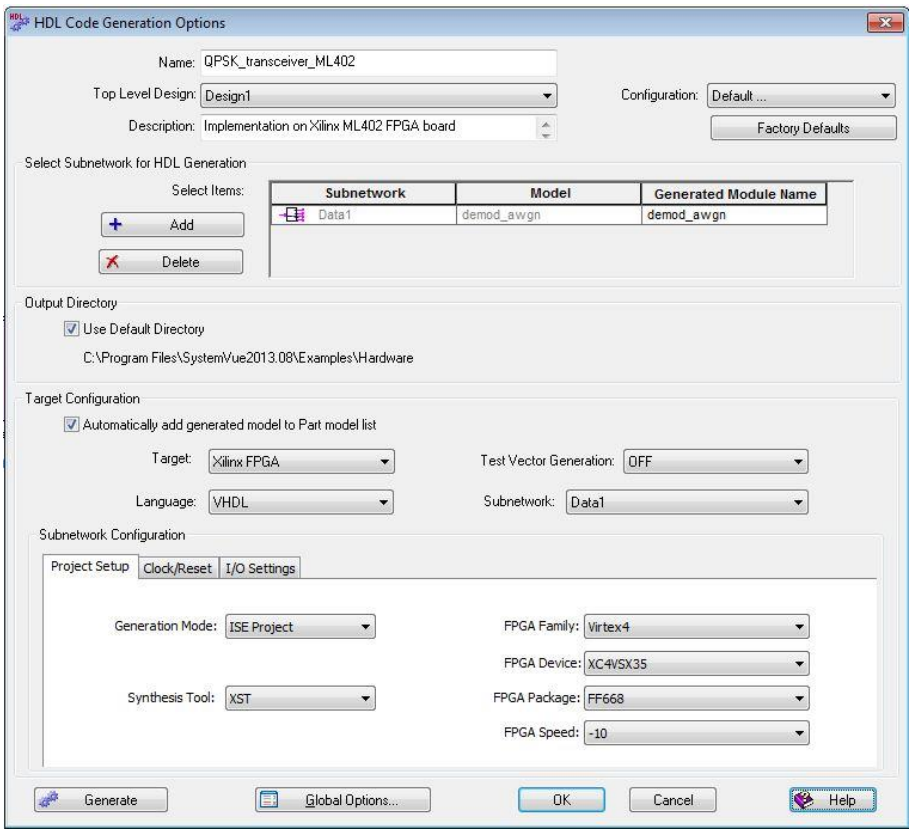


Figure 7.7. HDL Code Generation window.

Figure 7.7 shows how you can turn sub-circuits and top-level designs into HDL. The code can be generated in either VHDL or Verilog for any number of FPGA's. PathWave System Design (SystemVue) also provides a path to configure the clock and reset for your HDL design and set up an ISE project or generate a bit file directly.

Chapter 8: OFDM

8.1 Introduction

Now that we've covered the basics of digital communication it's time to look into the different variations that communication come in. It seems the same design challenges are always present when creating a digital communication system. One of these challenges is the drive to create faster networks and connections. This used to be accomplished by increasing the symbol rate, but that method is limited by an increase in BER. When sending information over a channel, signal distortion is usually on the forefront of a designer's mind. In turn, the different methods of digital communication are built considering effects like multi-path distortion and RF interference. In addition to the demand for signal reliability, there's also a push to create faster networks and connections. A digital communication scheme that tries to solve all these issues is called OFDM.

Orthogonal Frequency Division Multiplexing (or OFDM) works by transmitting data on multiple adjacent sub-band frequencies, something referred to as a multi-carrier modulation method. A single stream of data is broken down into multiple parallel signals as shown in Figure 8.1. The multiple signals are then set so they are orthogonal, or that the frequency responses have minimal overlap at their peak points, and therefore, don't affect each other. Each different signal has a different modulation scheme based on how much Signal-to-Noise Ratio (or SNR) there is on that particular frequency. If the SNR is low, then that frequency isn't used. If the SNR is high, a complex modulation scheme is used. And, if the SNR is medium, there will be a simple modulation scheme. With this, OFDM has great reliance to channel effects without the traditional equalization filters.

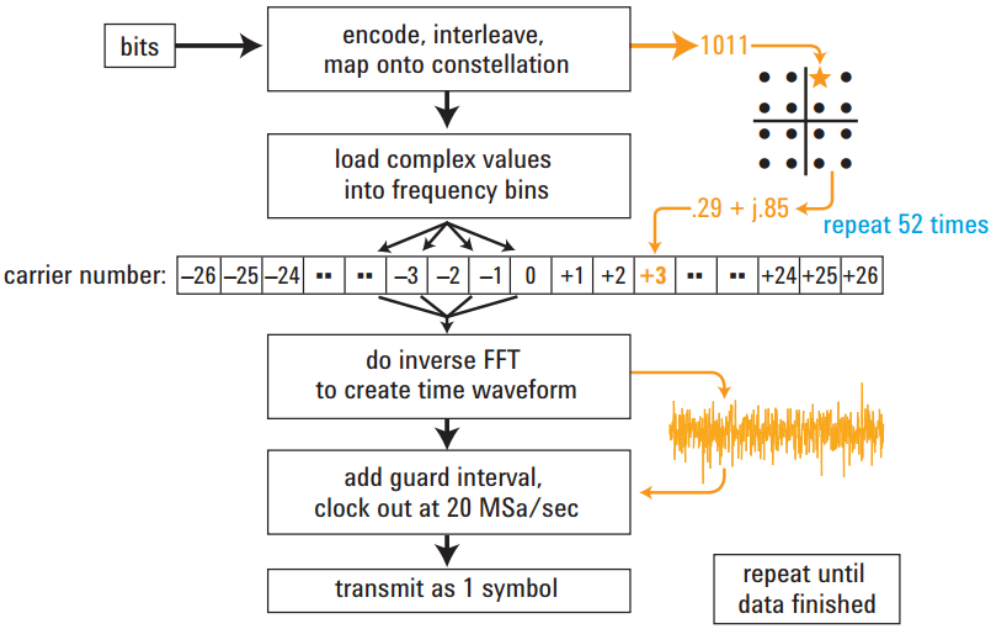


Figure 8.1. Example of an OFDM transmitting process.

OFDM presents a plethora of benefits. Its configuration makes it spectrally efficient, allowing the packing of signals and information into a tight frequency spectrum. Another benefit is that with the multiple signals it can transmit data many times the rate of a normal single input/output digital communication scheme. It's also structured so that any multipath and narrowband interference is mitigated, as fading on the corresponding frequencies causes those paths to not be used.

Because OFDM is good at weathering harsh channel conditions, it is used in a wide range of communication protocols. It's used in WiMAX™ and 4G LTE. It is now widely adopted in Ethernet and fiber optic transmission. In optical transmission systems, OFDM has the capability to overcome a variety of limitations such as modal dispersion and self-phase modulation.[50] Satellite and radios also use this as a form of communication with uses in avionics, GPS and military applications.

That doesn't mean OFDM is flawless. Because the system outputs multiple signals on different frequencies, it suffers from a high-Power Average Ratio (PAR). OFDM also has trouble when the system itself is incorrectly synchronized, which causes the carriers to lose orthogonality. Additionally, modulation in subcarriers must be reduced to keep a low noise floor in and out of the band.

Reference: WNW_OFDM_TxRx

To reach WNW_OFDM_TxRx:

1. Go to Open Example.
2. Open the Comms Folder.
3. Open the OFDM folder, it should be located here.

First let's look at the WNW_OFDM_Tx schematic in Figure 8.2.

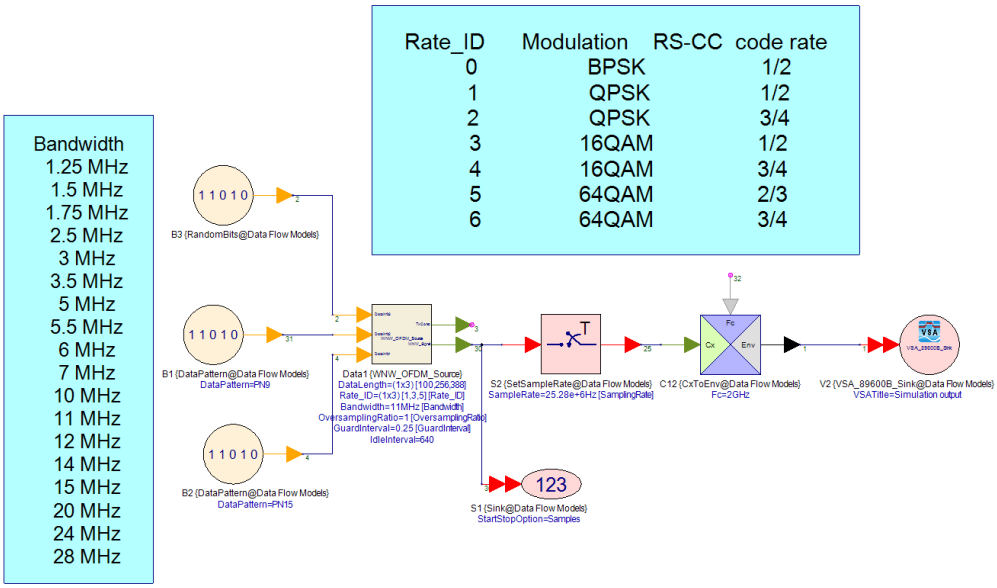


Figure 8.2. The WNW_OFDM_Tx schematic.

This schematic illustrates a top-level view of the OFDM transmitter, with blue boxes showing what

options are available for the Bandwidth and Rate_ID variables. These can be changed in the Parameters tab. In Figure 8.3, bit blocks representing three separate data inputs (e.g., preamble, idle, payload, and training) feed into the OFDM transmitter block. After the OFDM transmitter output is sampled, it heads to a CxToEnv, a modulator block that converts complex signals to I/Q values. These values are then read into a VSA block to be analyzed.

As a top-level view, this schematic can't do justice to the sub-circuit that shows the underpinnings of OFDM. Consequently, this chapter will also refer to the WNW_OFDM_Source model.

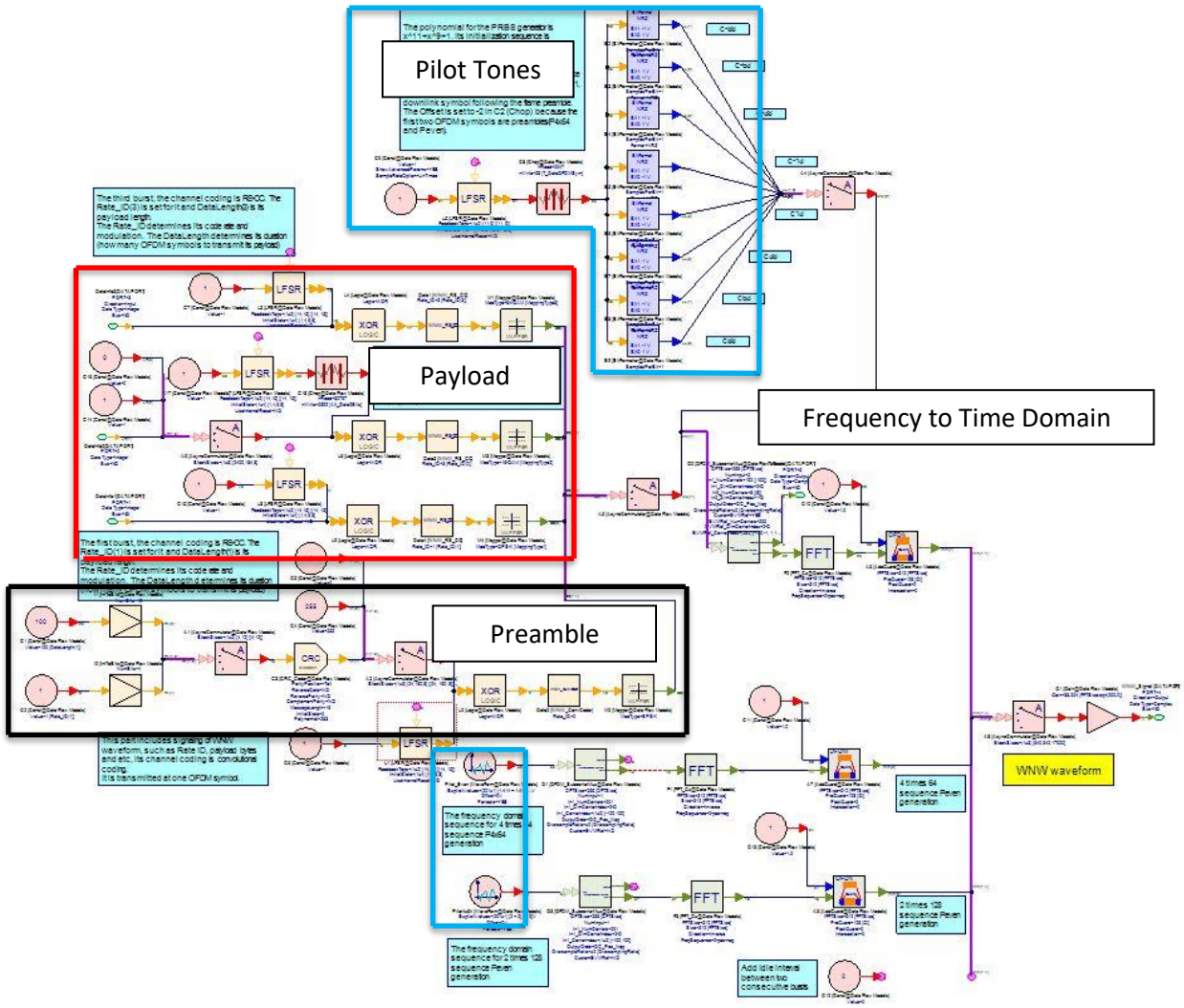


Figure 8.3. A zoomed out overview of the WNW_OFDM_Scource subcircuit with illustrative boxes.

While Figure 8.3 may look intimidating, it has been broken down into general sections allowing us to more easily understand this sub-circuit. The section containing the payload data is circled in red and takes three inputs from the bit sources in Figure 8.2. The pilot is circled in blue on the top of the schematic. The lower bottom portion encircled in black represents the preamble, while the blue area at the bottom represents a set of preamble/pilot hybrids. Almost everything else not circled has to do with frequency-to-time domain conversion.

8.2 OFDM Frame Structure

The basic frame structure for most OFDM examples looks like Figure 8.4.

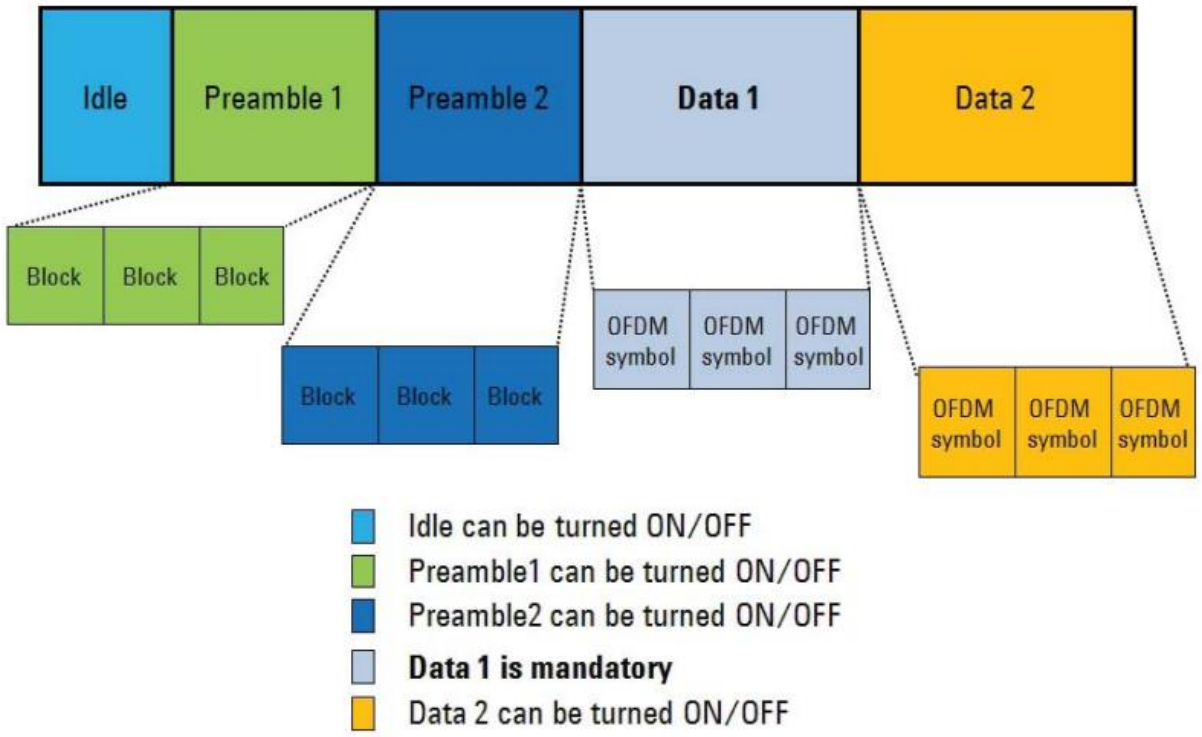


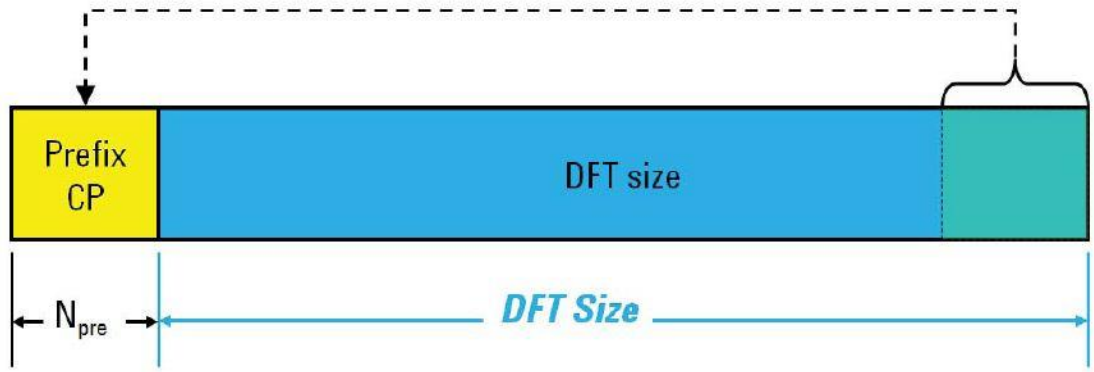
Figure 8.4. OFDM data structure.

Most frame structures start with an idle, preamble, and payload. OFDM, more often than not, starts with multiple preambles and payloads as illustrated in Figure 8.4. The payload and the preamble can each have multiple sections. In our WNW_OFDM_TxRx example, we have a single payload made of three parts and a single preamble. Preambles themselves are used in data transmission schemes that have burst (non-continuous) modes. Both the Preamble 1 and 2 sequences can be defined in the frequency domain or time domain according to the system specification. In addition to keeping the timing and frequency synchronization information, preambles are used for things like automatic gain control (AGC) adaptation, channel estimation and initial phase reference estimation. Preambles often come with cyclic prefixes and postfixes just in case of problems (Figure 8.5). This helps with intersymbol interference by acting as a guard interval, and allowing for simple frequency-domain processing, such as channel estimation and equalization.

Figure 8.5. Example preamble structure of an OFDM system.

8.3 OFDM Payload

The OFDM payload signal is generated when preforming an Inverse Fast Fourier Transform (IFFT) on the complex-valued signal. An OFDM payload can come in various modulation formats that are different from subcarrier to subcarrier.



(a)

(b)

Figure 8.6. Example of OFDM payload with cyclic prefix (a). Example of OFDM payload symbol with zeros prefix (b).

An OFDM symbol is built by appending a cyclic prefix to the beginning of each block generated. Figure

8.6 gives an example of two prefixes. Figure 8.6 (a) shows a cyclic prefix, which is used so that a channel group delay does not cause successive OFDM symbols or interference in adjacent subcarriers. Figure 8.6 (b) shows a payload with a guard interval full of zeros.

8.4 Pilot Signals

A pilot signal is a usually a single frequency transmitted for synchronization, control continuity, supervisory, or reference purposes.[11] In a continuous OFDM system, the pilots are used for timing and frequency synchronization and channel estimation.

There are four major pilot patterns in current OFDM standards (Figure 8.7). They include:

In Figure 8.7 (a), the OFDM systems do not include a pilot. In some cable OFDM systems (e.g., the ERDF G3-PLC system), there should not be any pilot because the pre-equalizer is adopted.

Figure 8.7 (b) represents OFDM systems that have continuous pilots. WLAN series standards (802.11a/g/n) only have continuous pilots (pilot subcarrier indexes are fixed in all OFDM symbols). These continuous pilots are used for phase tracking in Figure 8.4 (b) because these OFDM systems have a preamble sequence for channel estimation.

Figure 8.7 (c) shows OFDM systems with scattered pilots. Mobile WiMAX™ and 3GPP LTE cell communication standards are examples of systems with scattered pilots (that is, the pilot subcarrier indexes are alterable in each OFDM symbol). In this case, the scattered pilots are used for channel estimation because the OFDM system has preambles or synchronization channels for timing and frequency synchronization.

Figure 8.7 (d) represents OFDM systems that have both continuous and scattered pilots. Video standards (e.g., ISDB-T and DVB-T/H/T2/C2) have scattered pilots and continuous pilots. The scattered pilots are used for channel estimation, while the continuous pilots are used for timing and frequency synchronization.

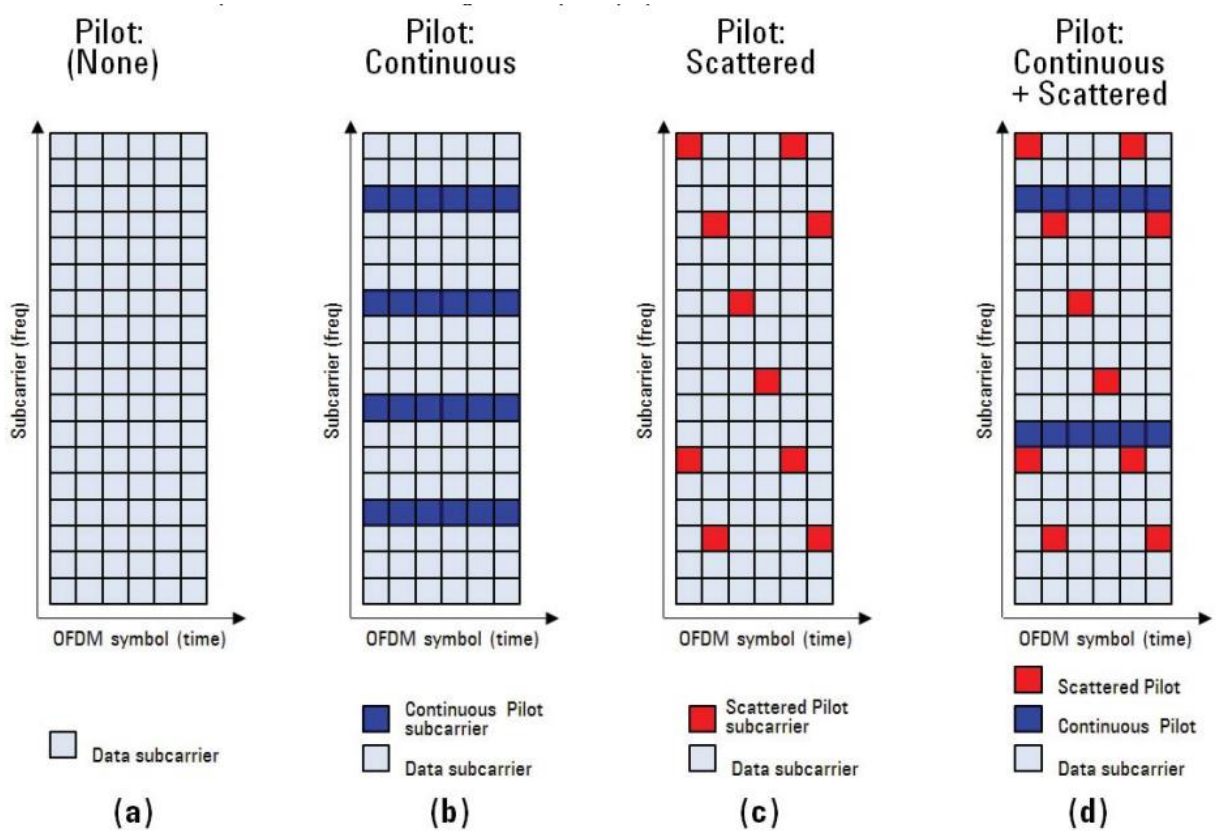
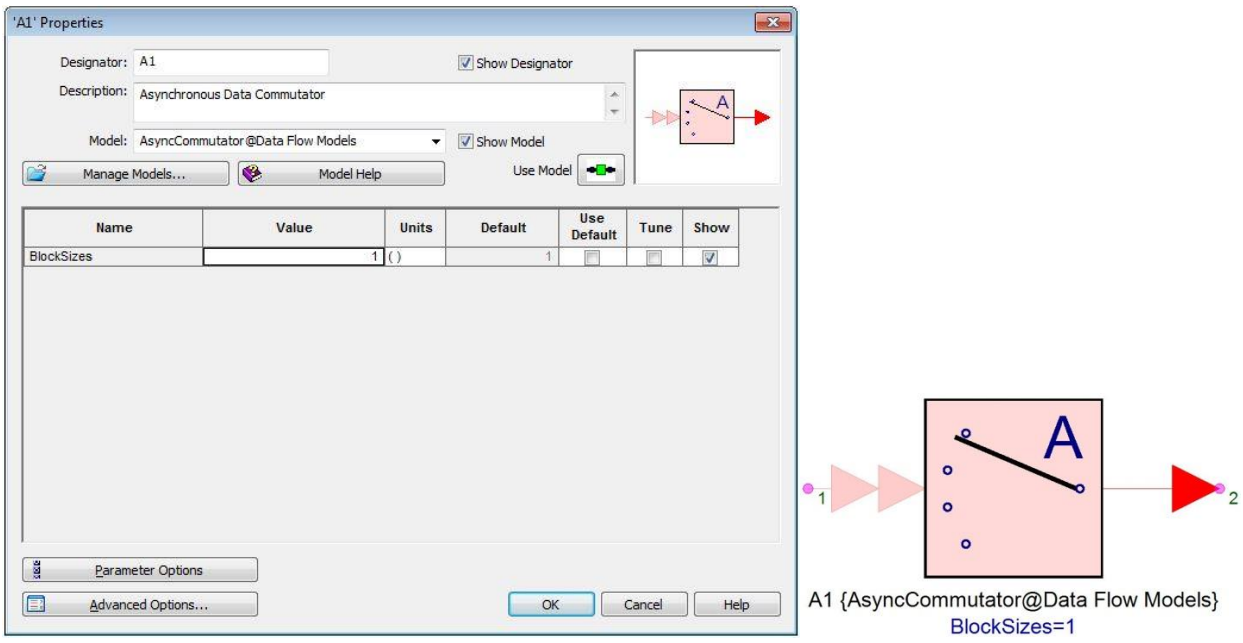


Figure 8.7. Various pilot structures.

8.5 Frequency-to-Time Domain

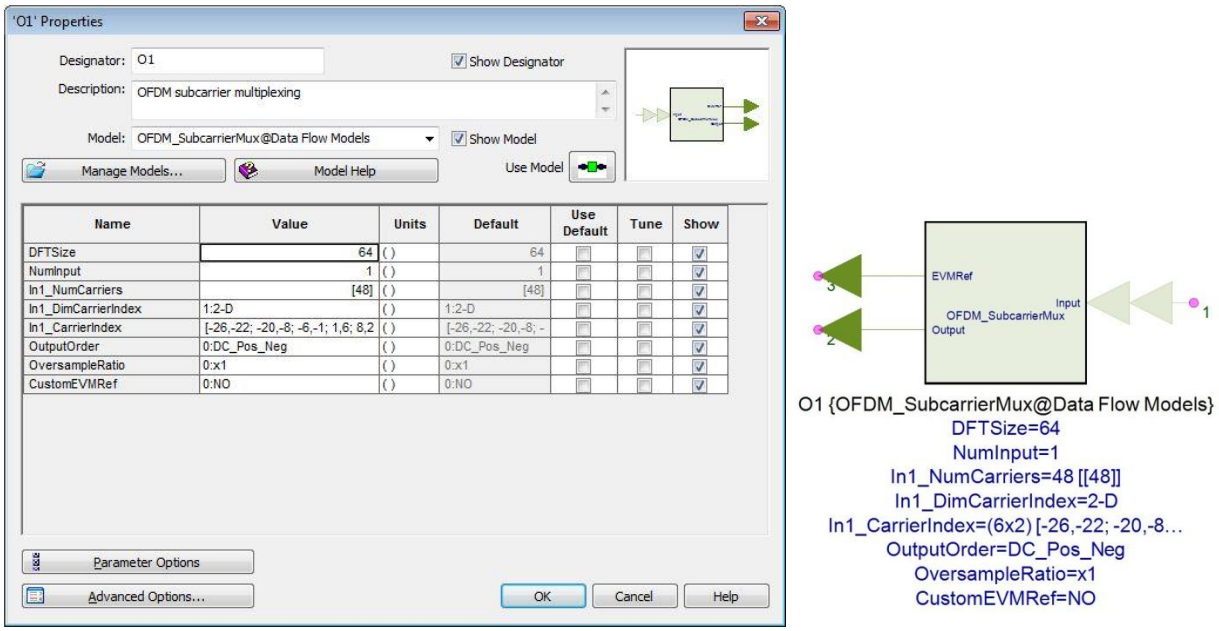
OFDM is different from multiple other communication formats in that it spreads the data sent in the frequency domain as sub-carriers, and then sends it to be Inverse Fast Fourier Transformed into the time domain for transmission as a modulating signal. Implementing this in PathWave System Design (SystemVue) is easy.



(a) **(b)**

Figure 8.8. AsyncCommutator Properties window (a). AsyncCommutator block (b).

First you start with the mapper outputs, which represent the signals making their way to be spread along the frequency spectrum. In this case, they feed into an AsyncCommuntator (Asynchronous Commutator) block. This block takes parallel muxed signals and puts them sequentially into one output (Figure 8.8).



(a) **(b)**

Figure 8.9. OFDM_SubcarrierMux Properties Window (a). OFDM_SubcarrierMux block (b).

The sequentially placed data then heads towards PathWave System Design (SystemVue)'s special OFDM_SubcarrierMux block, which automatically translates the symbol information into an OFDM frequency domain format (Figure 8.9). In this manner, the information is spread out between orthogonally spaced frequencies in the frequency domain. Before running, make sure that the NumInput variable matches the amount of inputs coming in.

Figure 8.10 shows how the sub-frequencies are spaced. The primary frequencies are the more oval, tall circles, while the ridges below represent the rest of the sinc frequency spectrum. This spectrum has zero crossings at $\pm\frac{1}{T}, \pm\frac{2}{T}, \pm\frac{3}{T}$, etc.

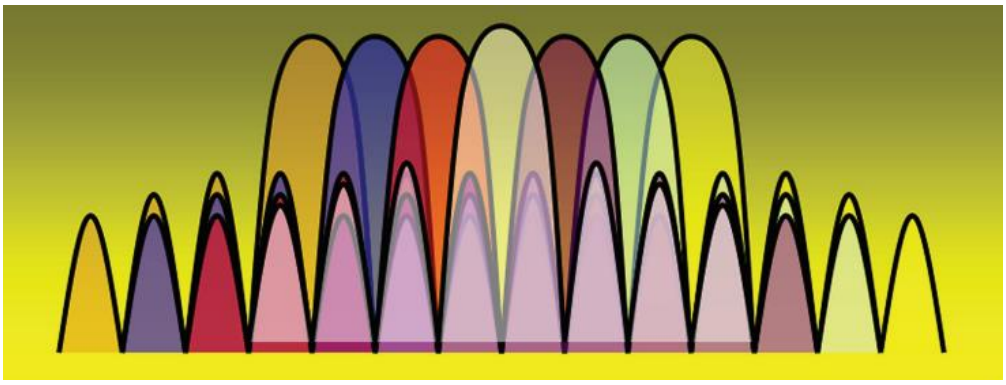
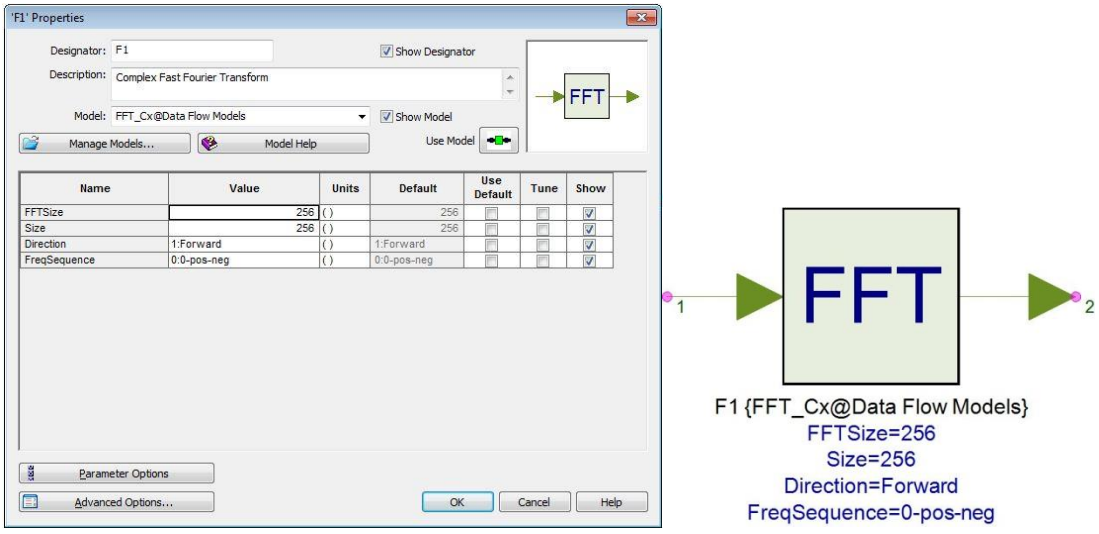


Figure 5.10. Orthogonal waveforms.

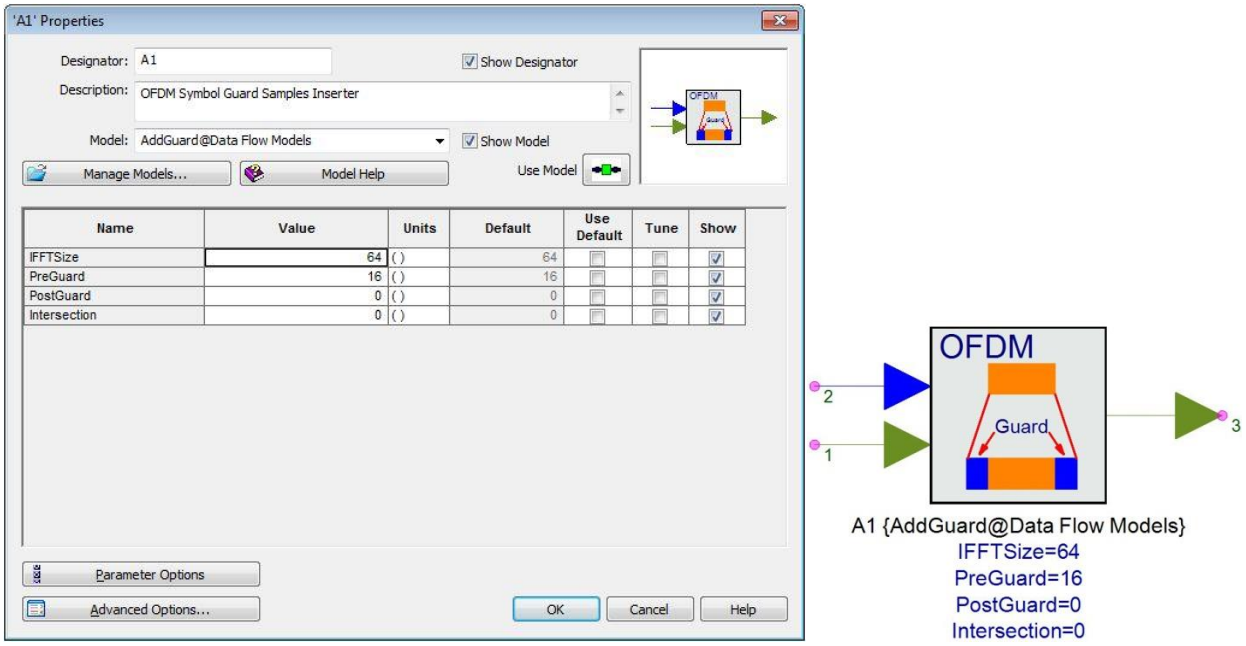
Therefore, if these waves are spaced by 1/T Hz at the desired frequencies they do not interfere with one another and are orthogonal.



(a) (b)

Figure 8.11. FFT_Cx Properties window (a). FFT_Cx block (b).

With the information encoded in this form it'll be prepped to apply the IFFT. This changes the frequency domain data into the time domain, allowing us to output a modulating waveform. To do so in PathWave System Design (SystemVue), take the FFT_CX block and modify it "Inverse" instead of "Forward," (Figure 8.11).



(a) **(b)**

Figure 8.12. AddGuard Properties window (a). AddGuard block (b).

After the waveform is created, add the AddGuard block so that guards are added to either side (Figure 8.12). The guard value will be determined by the top input. You can see a sample final result in Figure 8.13.

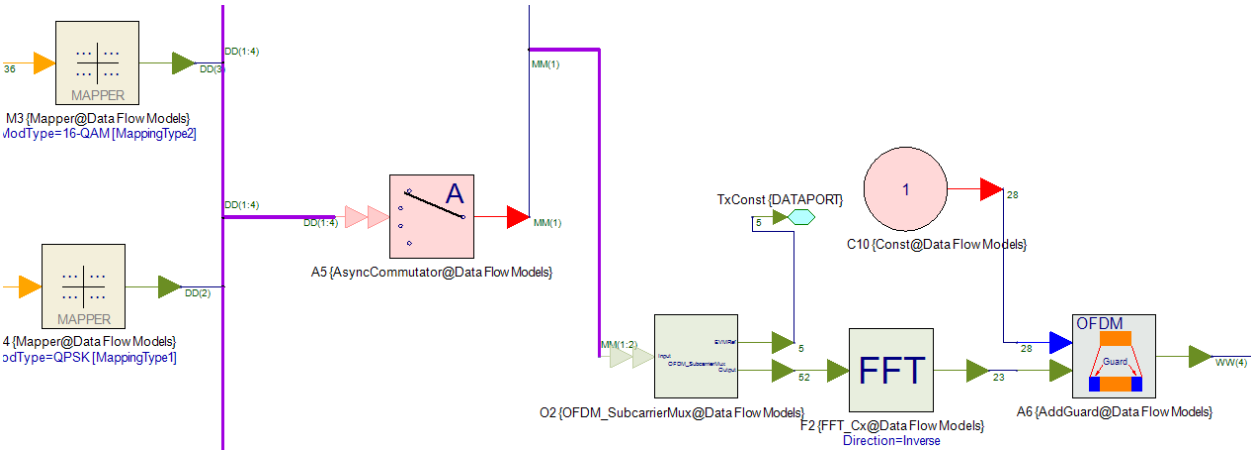


Figure 8.13. A small part of model WNW_OFDM_Source, which shows the subcarrier multiplexing, IFFT and added guards.

Chapter 9: Spread Spectrum CDMA Tutorial

9.1 Introduction

Code Division Multiple Access (CDMA) systems allow users to simultaneously transmit data over the same frequency band. In this transmission technique, the frequency spectrum of a data signal is spread using a special coding scheme (where each user is assigned a code). This tutorial builds and examines the design process of a simple spread spectrum system (CDMA) and exposes all necessary tools to simulate, display and analyze critical aspects of the system using PathWave System Design (SystemVue).

CDMA is a form of a direct sequence spread spectrum communication where the transmitted data is coded at a very high frequency. In spread spectrum, to send data, the signal bandwidth is much larger than necessary to support multi-user access. In addition, the large bandwidth ensures interference of other users does not occur. Multi-user access is achieved using a code that is pseudo-random. A pseudo-random code is generated using a special coding generator to separate different users. The code appears random but is in fact known, allowing the receiver to reconstruct the code. In this tutorial the pseudo-random code is generated using a Walsh code generator from an orthogonal set of codes.

During signal transmission, the data signal is modulated with the pseudo code and the resultant signal modulates a carrier. The carrier is then amplified and broadcasted.

At the receiving end, the carrier is received and amplified. The signal information from the coded signal is then recovered by correlating the received code with a generated code (the same Walsh code at the transmission) at the receiver. Thus, the receiver can reconstruct the code and extract the transmitted data.

This tutorial is comprised of 8 step-by-step lab sessions that lead to a final completed digital modulation system. Using BER to assess the performance of our CDMA spread spectrum system, we can observe the bit stream over the designed communication channel.

9.2 Lab 1: Bit Generation Pattern

In this lab we will create a very simple bit generation pattern and analyze its output in the form of an eye diagram.

1. Start PathWave System Design (SystemVue) (This tutorial uses version 2011.10) by double clicking on the desktop icon.
2. After loading, the “welcome” page featuring intuitive tutorial videos will appear. For now, skip this. Click close.
3. The next window allows you to open existing workspaces or templates. For this session, click “cancel” to create a blank workspace.

This will bring up the schematic window where your design work will take place (Figure 9.1). The top of

the screen presents the toolbar; on the left-hand side you will see a directory structure that is your "workspace tree." The workspace tree allows you to organize designs, displays, equations, etc… by employment of a hierarchy folder structure. Below the workspace tree, is the "tune window," which is one of the most powerful features of PathWave System Design (SystemVue) that allows you to use tuned variables (real-time tuning of values in variables) anywhere in your design. The right-hand side displays a library of parts (called Part Selector). At the bottom of the screen, information of errors is displayed. Also, notice the tabs placed below the error screen showing other log information.

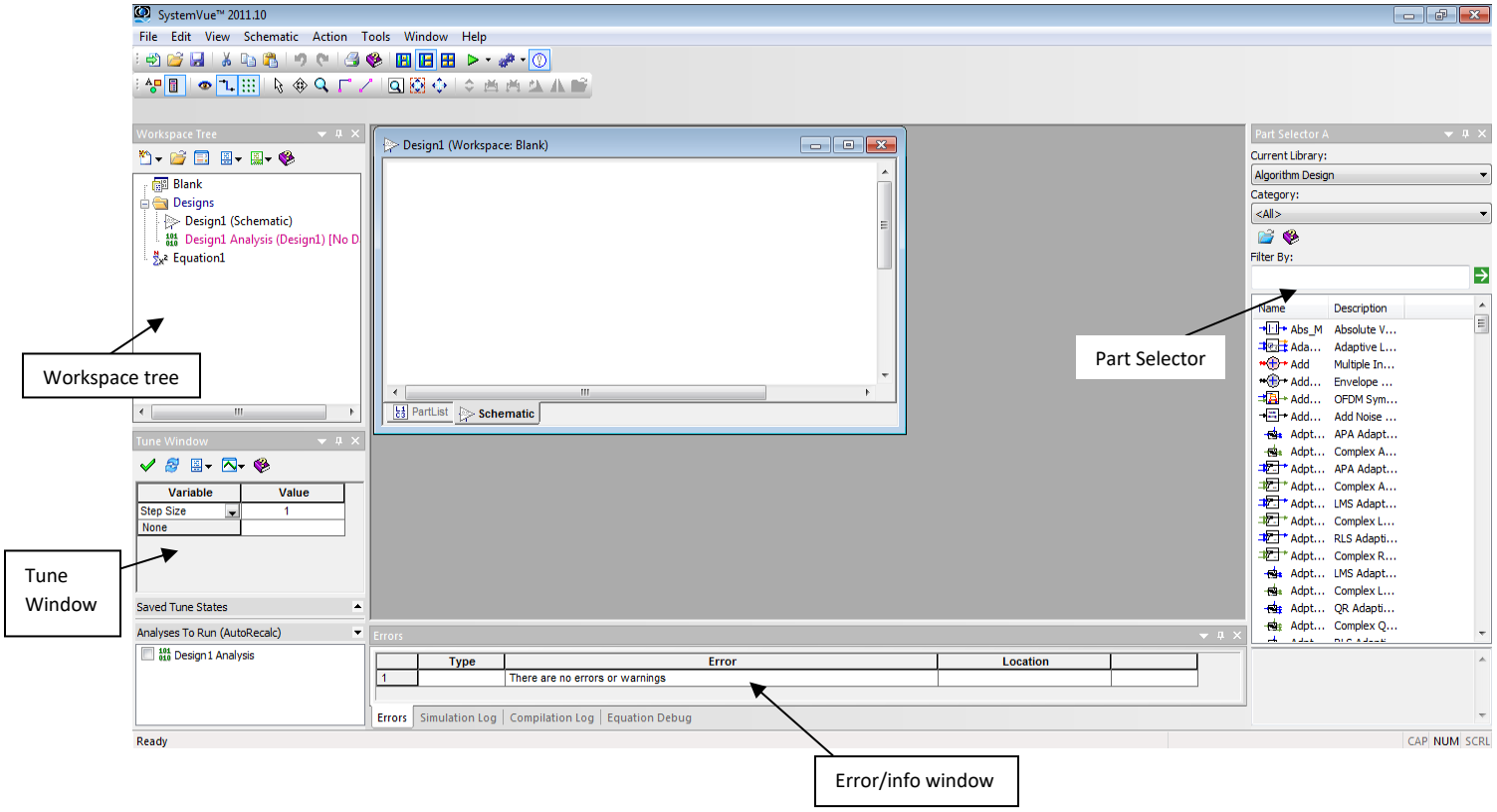


Figure 9.1. PathWave System Design (SystemVue) schematic window.

Let us begin.

1. Enter **File**>**Save As**> **My_CDMA_Sys**. This will save the entire work session by name "My_CDMA_Sys." It is a good idea to save the workspace after each lab by performing File>Save.

By default, PathWave System Design (SystemVue) would have created a folder and populated it with a design and an analysis. Rename the folder "Designs" by placing the cursor over the "Designs" folder. Right Click> Rename. Give it the name "Step1_Bits." Rename the schematic Design1 as "Bits" and Design Analysis as "DF_Bits." This is the design flow analysis that gathers information about the set of values set and runs a design program-similar to a compiler.

2. Double-click **DF_Bits** and enter the values shown in Figure 9.2.

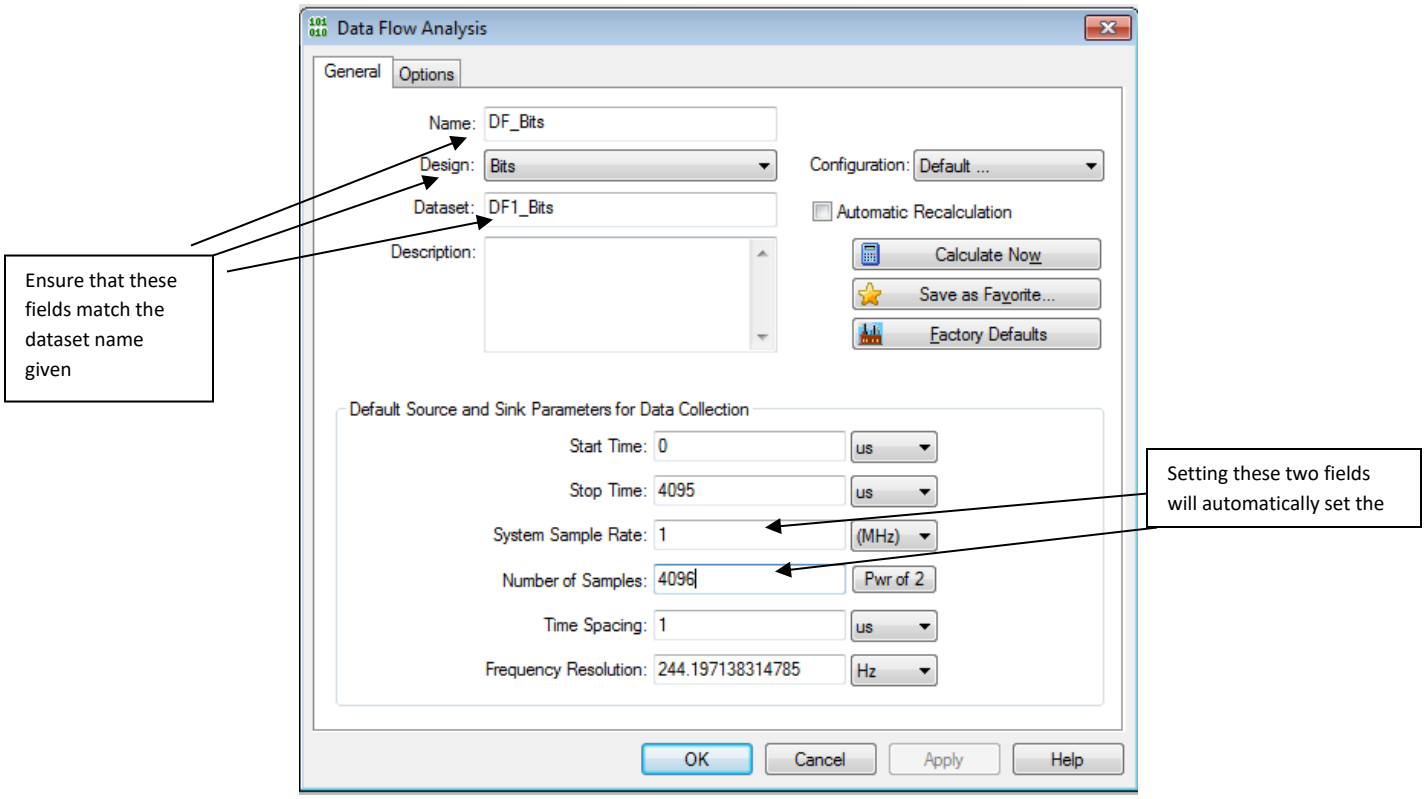


Figure 9.2. Entering values in the design flow analysis.

The key parameter is System Sample Rate. The other essential parameter is the Number Samples, which determines the length of simulation. Start and Stop times are automatically set. The frequency resolution is calculated from Stop Time.

1. Press “Ok” to dismiss the window.
2. Maximize the design workspace to full screen.
3. From “Part Selector A,” type “Bits” into the “filter by” field and press “enter.” You will be shown all the parts related to the word “Bits.” Single Click on “Bits” and move your cursor across to the blank schematic. Click again to release the part.
4. Double Click on the “Bits” Part. One of the nice and perceptive features of PathWave System Design (SystemVue) is that most parts are polymorphic models.

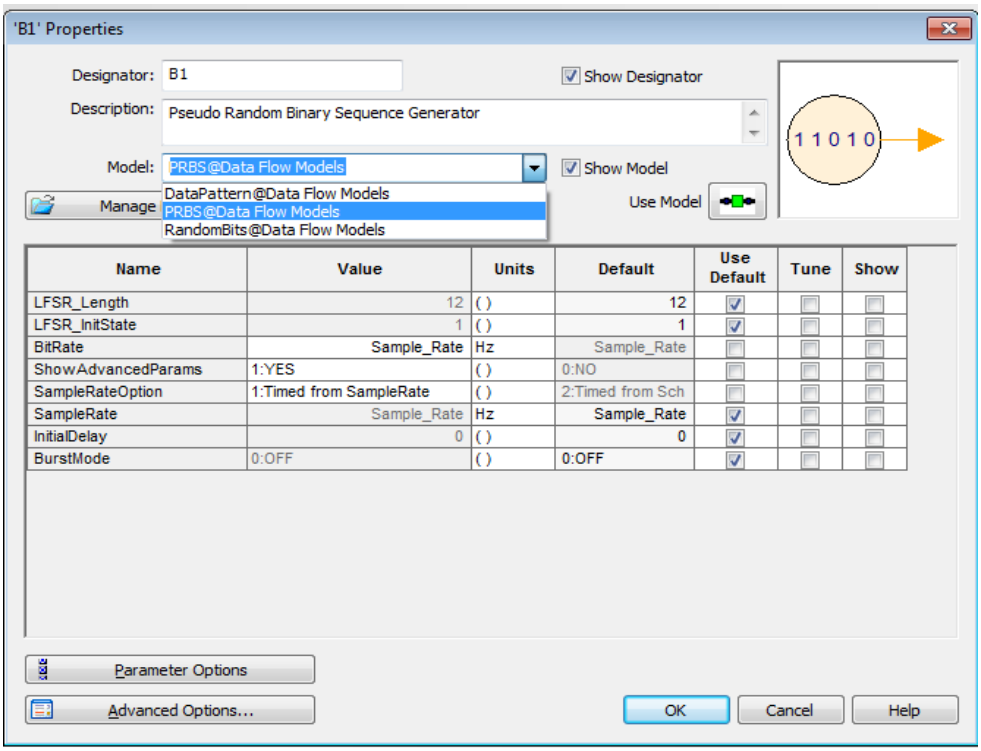


Figure 9.3. Drop down B1 properties menu.

5. Click on the drop down menu next to "model" field and select "PRBS@ Data Flow Models." This will generate a pseudo-random bit sequence.
6. Fill in the fields as shown in Figure 9.3. Note that you can add your own models to this list, making PathWave System Design (SystemVue) highly customizable.

   Notice that the "BitRate" is set to the value of the "Sample Rate." With these settings we have one sample per bit.

   Now, let's connect a sink to the bit source and display our bit pattern. Type "Sink" into the Part Selector. Click on the "Data Sink" and place it next to the bits source. A connection will be formed and shown in Figure 9.4.

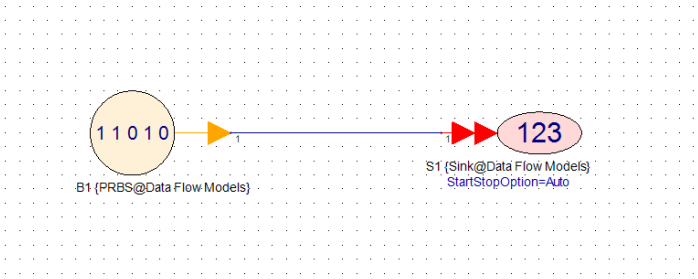


Figure 9.4. Connecting a sink to the bit source.

Shortcut: On the Schematic, you can type "s" on your keyboard and then click on the

schematic to place a sink. Type "w" to place a wire (drag a wire from one component to another), "f" for filter, etc…

Right-click on the "DF_Bits (Bits)" and select "Run (calculate now)." You will observe that a dataset has been added after simulation. From this dataset, we can see the output.

To stay organized, right click on the dataset> Rename. Call it "DF1_Bits."

Since the dataset name has now been changed, we must ensure that the data analysis simulates the correct dataset.

7. Double click on the analysis "DF_Bits"> next to field Dataset>Type "DF1_Bits">Click Ok.

   Notice that Data Flow and Dataset are now coloured in red. This means that the simulation is not up to date as we have made a change to our design.

8. Right click on "DF_Bits">Run(Calculate Now).

   Upon errorless simulation, let's graph the bits:

9. Double click on the dataset "DF1_Bits" to open it and you will see variables corresponding to the sink.
10. Right mouse click on S1>Add to Graph>New Graph Series Wizard (Figure 9.5).

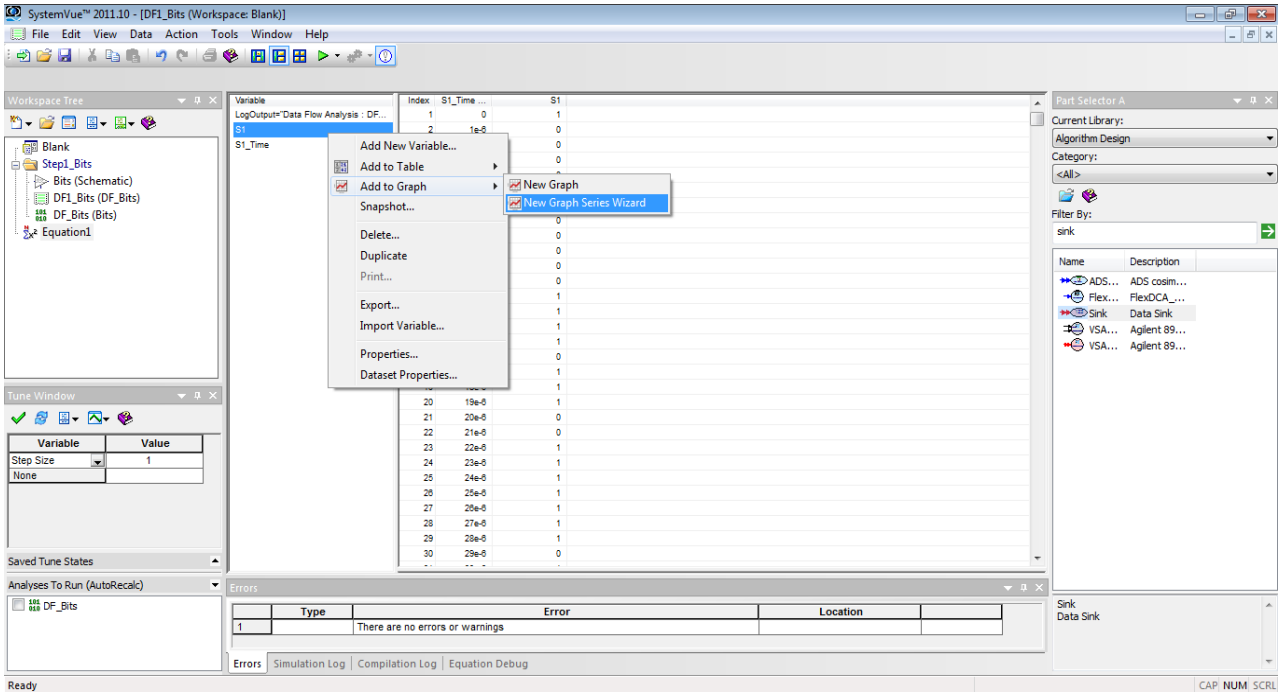


Figure 9.5. Selecting the New Graph Series Wizard.

11. Under "select type of series," choose "Time." Ensure that in the field box "Data Selected," the "S1" box is checked as this is the output we wish to graph (Figure 9.6).

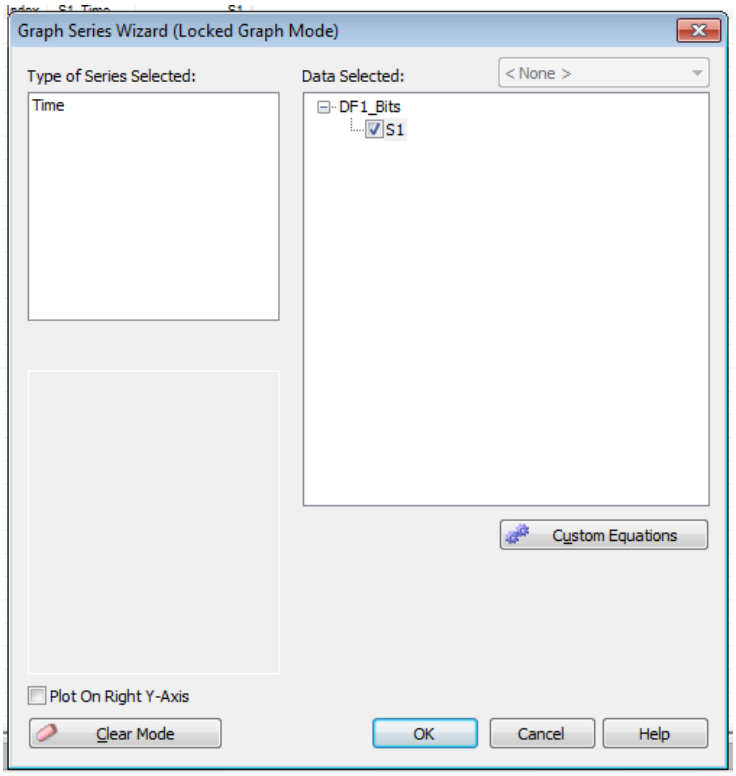


Figure 9.6. Select "S1" so that its output will be graphed.

12. Press Ok. Doing so will bring up a graph properties dialog with "S1" as the only variable to be plotted (Called MyVar).
13. Press Ok. This will lead you to plot our design "Bits" in time domain (Figure 9.7).

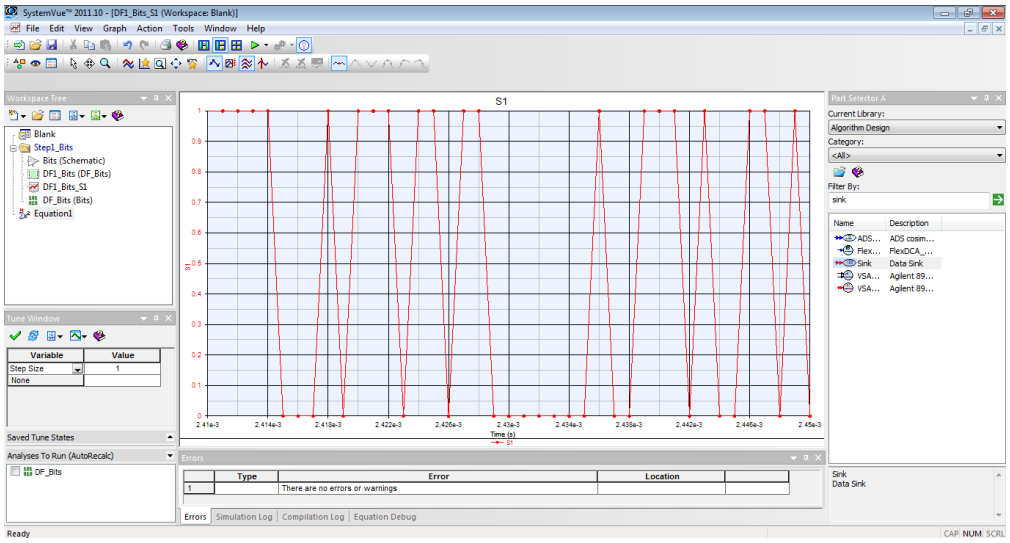


Figure 9.7. Graph of the design "Bits."

14. Use your mouse wheel to zoom in on the x-axis of the graph in Figure 9.7. As you scroll forward, you should gradually see the random 1's and 0's more clearly.

Eye diagrams are often used in communication to provide designers insight into signal characteristics.

1. Modify the schematic to match that shown in Figure 9.8 by copying the “Bits” schematic created previously (check that the same parts used have the same variables as before).

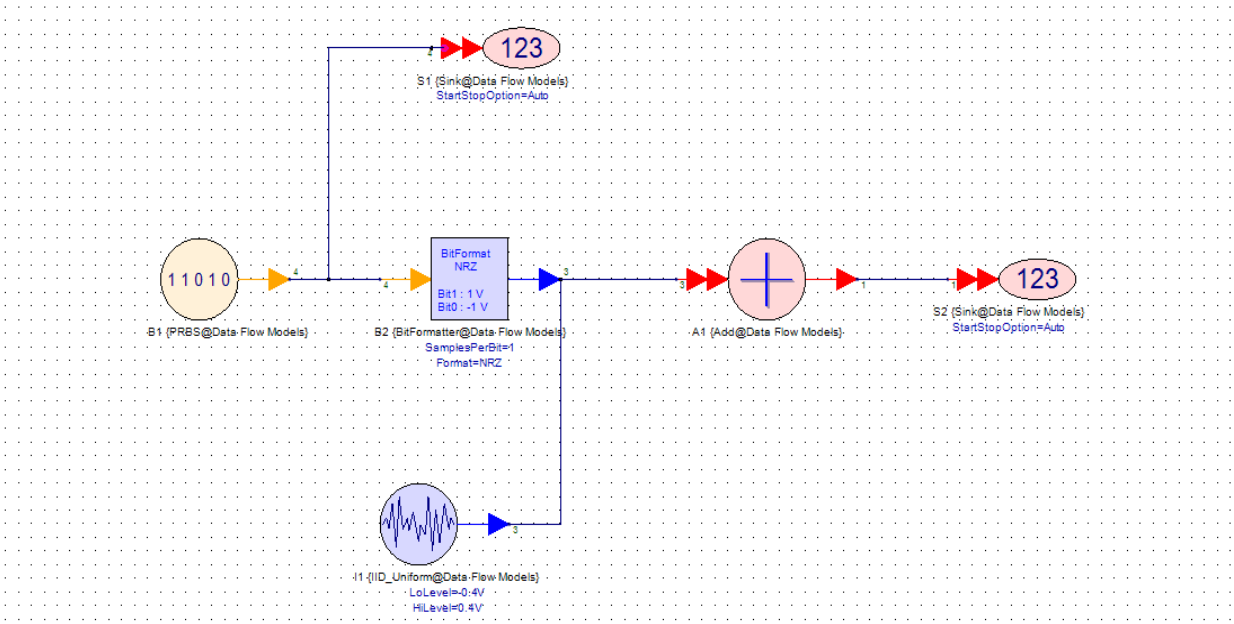


Figure 9.8. An eye diagram can be created for this schematic to better understand its signal characteristics.

2. The BitFormat can be found in part selector by typing “NRZ” into the search field. Similarly, for the IID Uniform Noise Source, type “Noise” and the Adder can be withdrawn from the parts library by searching for “Add.” The NRZ part will transform the digital bits to a voltage of our choice; we have added some noise to make the eye diagram more interesting and realistic.
3. Double click on the noise source and set the LoLevel voltage to -0.4V and the HiLevel voltage to 0.4V. This can be achieved by clicking on the fields and entering the values.
4. Simulate the design by right clicking on “DF_Bits”>Run.
5. Double click on the dataset and right mouse click on S2>Add to Graph>New Graph Series Wizard>Eye>Custom Equations. Modify the equations to:

```
SymbolRate = 1e6;        % in Hz
NumCycles = 3; % Number of cycles to plot before wrapping
StartUpDelay = 2;        % Number of start up samples that will be removed
```

6. Click Ok three times to dismiss the windows. You should now see the eye diagram in Figure 9.9.

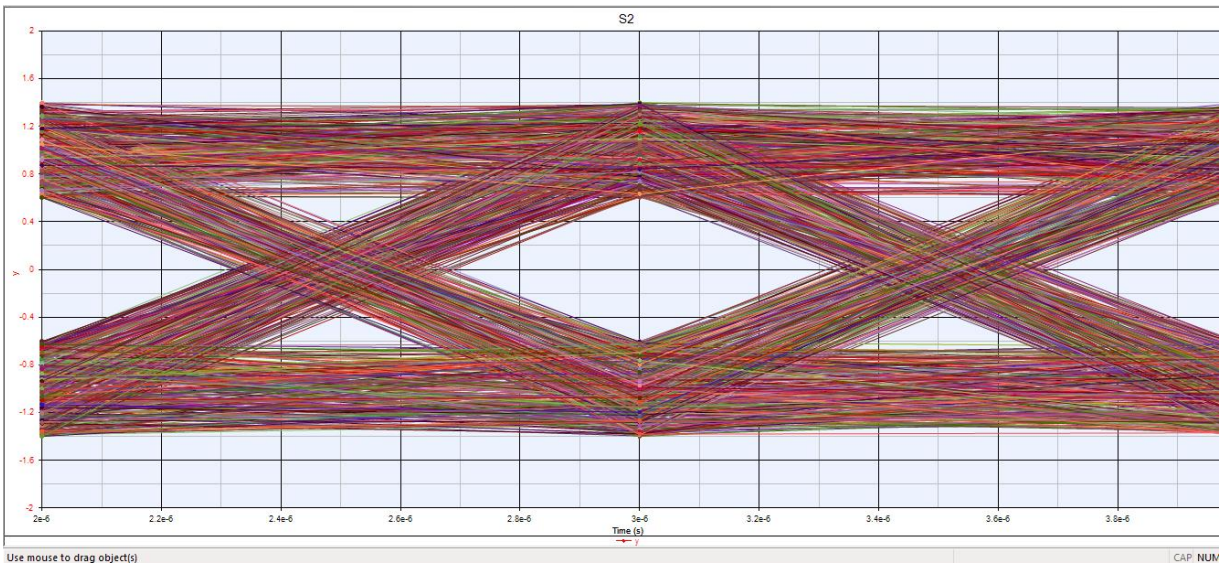


Figure 9.9. Eye diagram of the schematic in Figure 9.8.

You can modify the graph by double clicking on it. This will bring up the graph properties dialog. Notice that variable "y" has been named with context "<post proceed>." This means that you can post proceed your time domain data to generate the eye diagram. Pressing "edit" within the dialog window will take you back to the graph wizard where you can modify the equations to alter your graph.

Lab 1 is now complete. Don't forget to save your workspace.

9.3 Lab 2: Mapping

In this lab, we map the bit data onto a popular digital modulation format. We will view the plot on a constellation diagram.

Create a new folder called "Step2_Mapping," by following these steps:

1. Right click on path "My_CDMA_Sys," which is placed at the top of the workspace tree, and >Add>Add Folder>. Give it the name "Step2_Mapping."
2. Right click on the folder "Step2_Mapping">Add>Designs>Add Schematic>. Name the design "Mapping."
3. Add a dataset by right clicking on the folder "Step2_Mapping">Add>Add Data>. Name it "DF1_Mapping."
4. Right click on the folder "Step2_Mapping">Add>Analysis>Add DataFlow Analysis>. Call it "DF_Mapping." Within the DataFlow Analysis pop window, ensure that the design and dataset fields correspond to this simulation as shown in Figure 9.10.

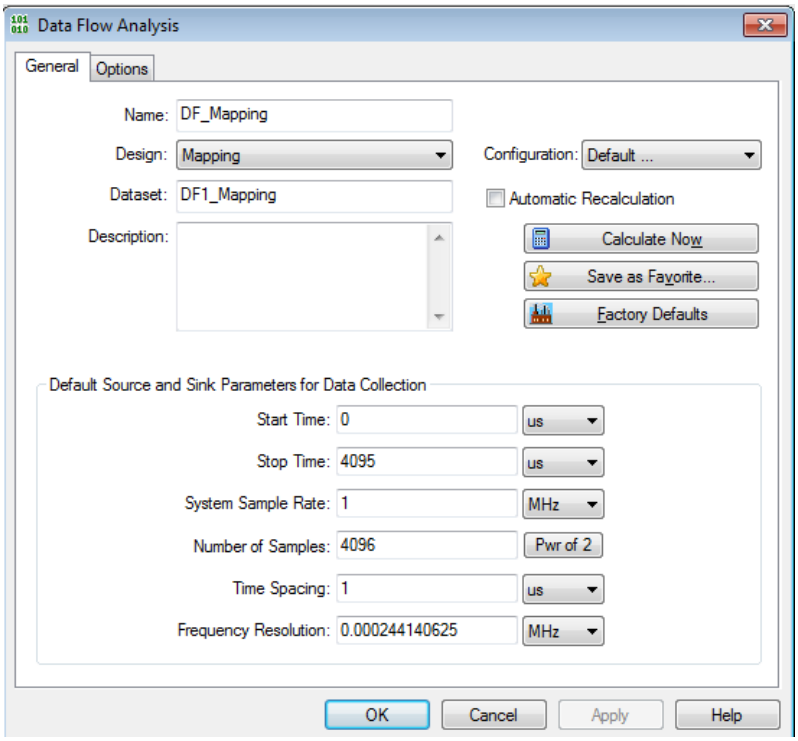


Figure 9.10. The design and data fields must correspond to the simulation.

In schematic view, let's map the bits using QPSK modulation as shown in Figure 9.11. For the Bits part, be sure to set the parameters as you did in Lab 1, #4.

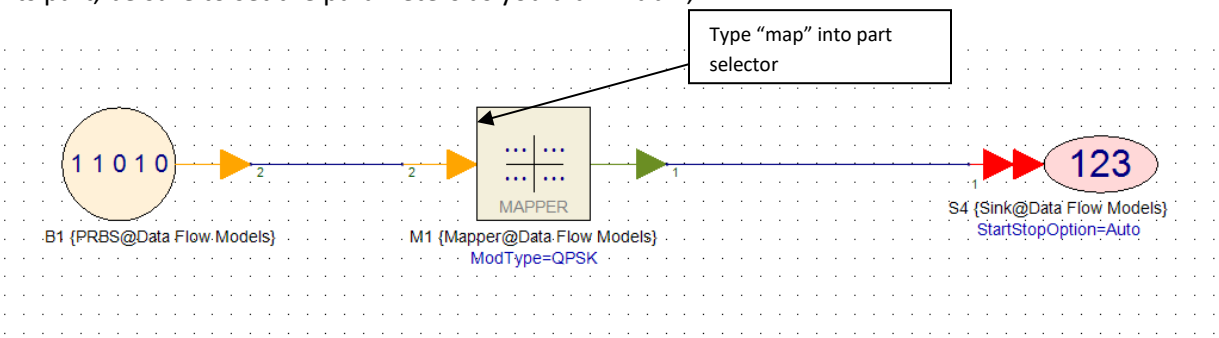


Figure 9.11. Mapping the Bits.

Run the simulation:

5. Right click on "DF_Mapping">Run(Calculate Now).
6. Double click on the dataset and right mouse click on the sink (S1)>Add to Graph>New Graph Series Wizard>Constellation>Ok>Ok again.

You should see a perfect constellation diagram (Figure 9.12).

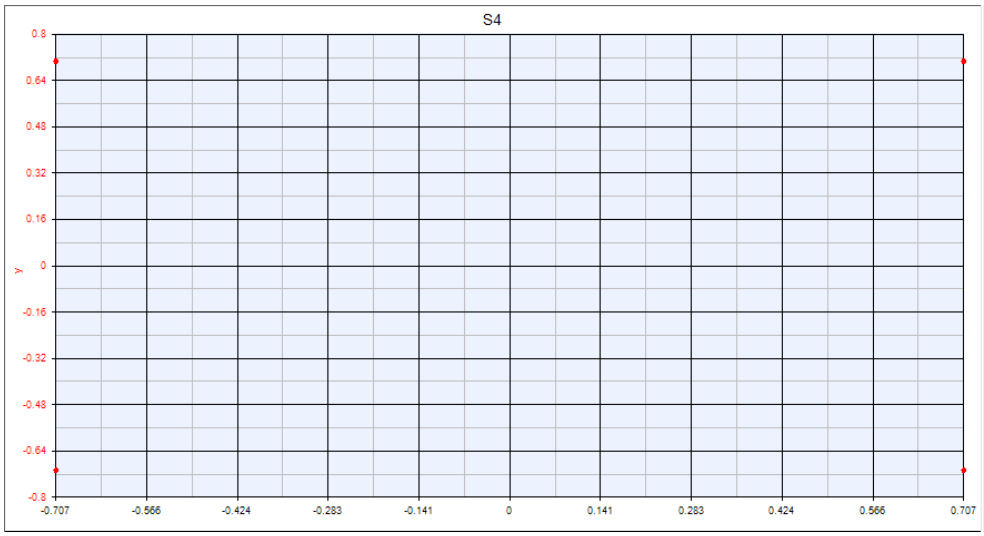


Figure 9.12. Perfect constellation diagram.

We can add some noise to the system and see what happens to the constellation diagram.

7. Modify the design as shown in Figure 9.13.

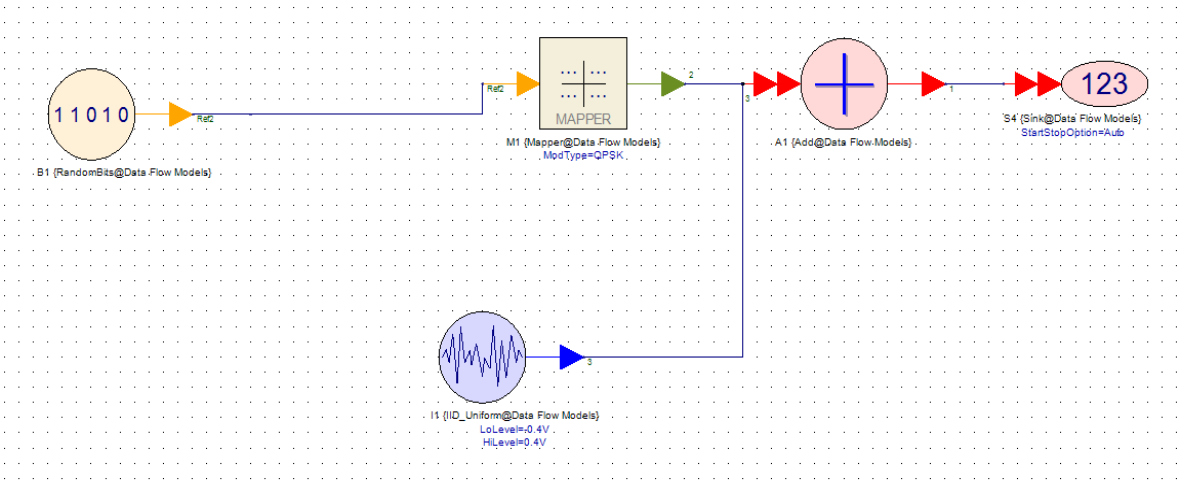


Figure 9.13. Modifying the constellation diagram.

It can be good practice to name wires for neatness. For this, right click on the wire>Net> Edit Net name. In this case, we have chosen the name Ref.

8. Now run the simulation by right clicking “DF_Mapping”>Run.
9. Navigate back to the previous graph made and you will see the noise added to the constellation (Figure 9.14).

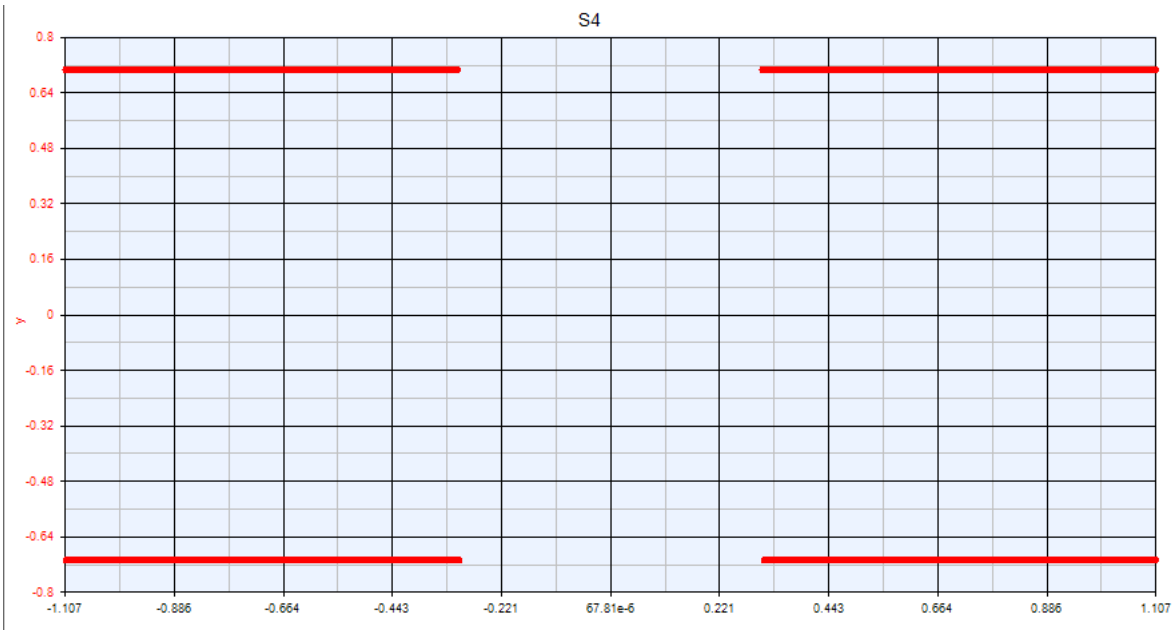


Figure 9.14. Constellation diagram with noise added.

End of Lab 2.

9.4 Lab 3: Walsh Code

In this lab we will generate a Walsh code for our data signal

1. Right click “My_CDMA_Sys”>Add>Add folder. Name it “Step3_WalshCode.”
2. Right click on the folder “Step3_WalshCode”>Add>Designs>Add Schematic. Name the schematic “WalshCode.”
3. Right click on the folder “Step3_WalshCode”>Add>Analysis>Data Flow Analysis. Call it “DF_WalshCode.”
4. Right click on the folder “Step3_WalshCode”>Add>Add Data. Title it “DF1_WalshCode.”
5. Double click on the Data Flow Analysis and alter the parameters so that the ‘design’ and ‘dataset’ are coherent, that is, the design field reads “WalshCode” and the dataset field reads “DF1_WalshCode.” All other parameters should match the analysis previously set.

   Copy the schematic from “Step2_Mapping” and modify the design as shown in Figure 9.15.

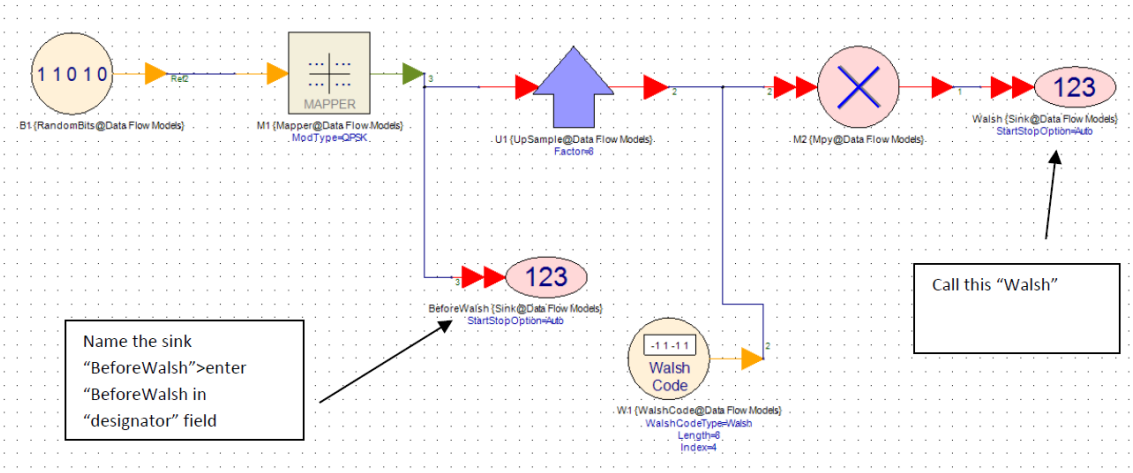


Figure 9.15. Modified schematic.

After the data is plotted for mapping, the symbol rate is reduced by a factor of 4, due to dual polarization and QPSK. An up sampler increases the sample rate back a certain factor. In this case, we up sample by a factor of 8 so the symbol rate is 4 MHz giving us enough bandwidth to transmit on for the carrier and over the communication channel.

6. Double click on the Up Sample part>Set the factor of the value to 8. Set the mode to be "HoldSample." The input sample is repeated by a factor of 8 at the output.
7. Double click on WalshCode Generator part. Set the following values: as shown in Figure 9.16.

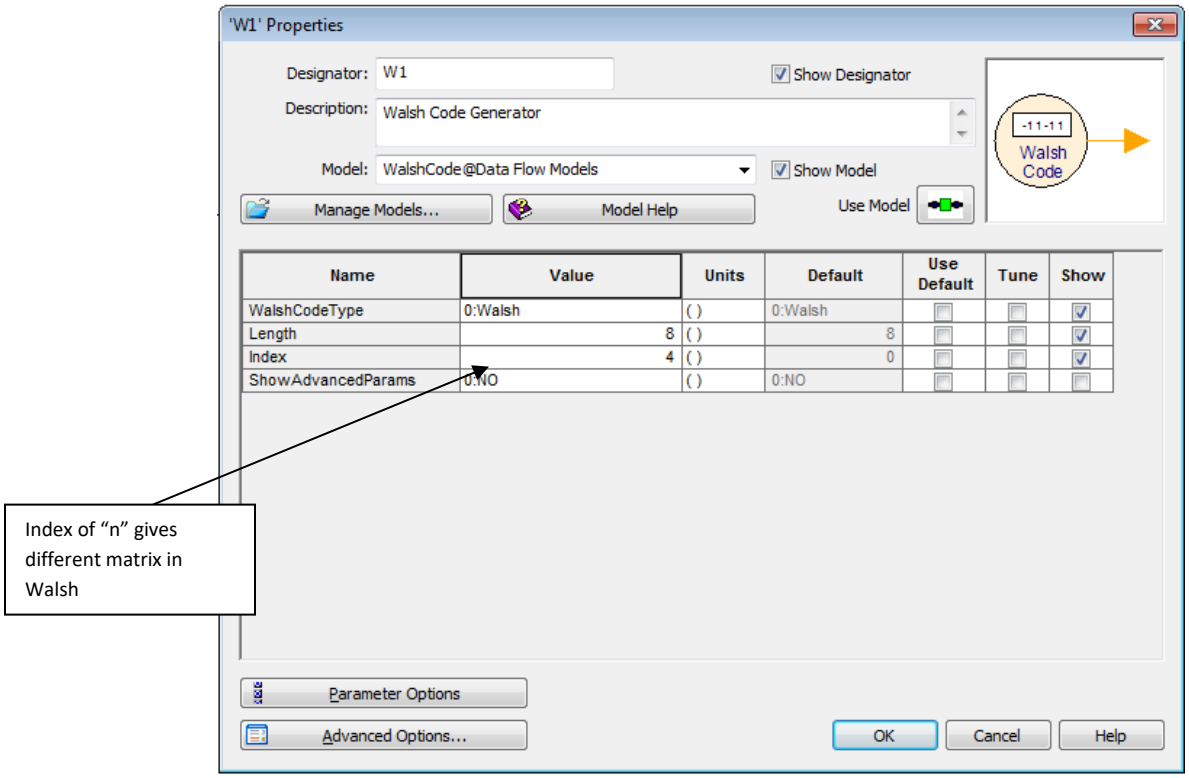
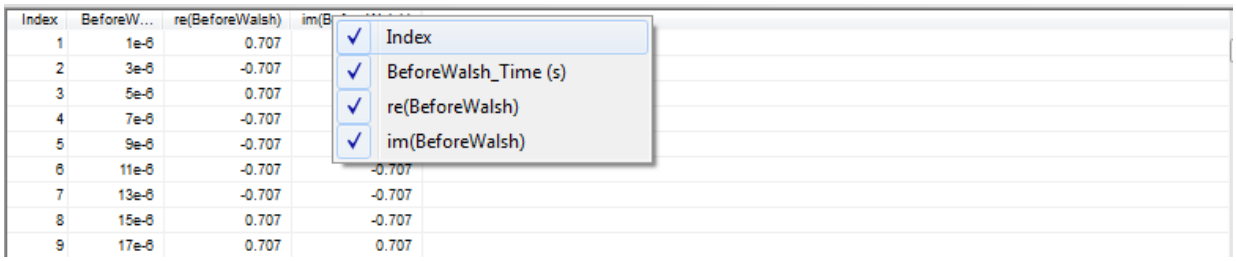


Figure 9.16. Setting the values for Walsh code generation.

Let's view the values of the code before and after Walsh Code generation.

8. Run the simulation.
9. Double click on the dataset>Right Click on "BeforeWalsh" >Add to Table>New Table>Click Close.
10. Right click on the top of the table, next to the value names (Figure 9.17).



Index	BeforeW...	re(BeforeWalsh)	im(B...
1	1e-6	0.707	
2	3e-6	-0.707	
3	5e-6	0.707	
4	7e-6	-0.707	
5	9e-6	-0.707	
6	11e-6	-0.707	-0.707
7	13e-6	-0.707	-0.707
8	15e-6	0.707	-0.707
9	17e-6	0.707	0.707

Figure 9.17. In the table, click on "Index."

11. Click on "Index" and "BeforeWalsh_Time(s)." You should see the respective columns disappear in the table.
12. Double click on the Walsh Code dataset>Right Click on Walsh>Add to Table>Add to "DF1_WalshCode_BeforeWalsh."
13. Maximize the window to support full screen. You can reject column "Walsh_Time(s)" as it's not needed.

Observe and compare the values before and after the Walsh Code is generated. Notice that the values differ. This highlights that our code is securely protected with our Walsh code. We now must ensure that the same Walsh code is applied at the receiving end to decode our data and retrieve the original code of information. During the final lab, we will use some examples and look at what happens to individual bits as they are correlated with Walsh coding.

Additional Exercise:

Change the index of the Walsh Code Generator via properties. Create the table of values again or before WalshCode generation and after. Notice the difference in values.

Lab 3 is finished.

9.5 Lab 4: Filter Design

In this lab we will create a folder that will conserve the modulated bandwidth.

1. Create a folder and call it "Step5_Filter."
2. Add a schematic design called "Filter."
3. Add a dataset and name it "DF1_Filter."
4. Add a data flow analysis and call it "DF_Filter." Set Number of Samples to 4096.

Copy the schematic created in Step3 and paste it into the new blank schematic created. Modify the design as shown in Figure 9.18

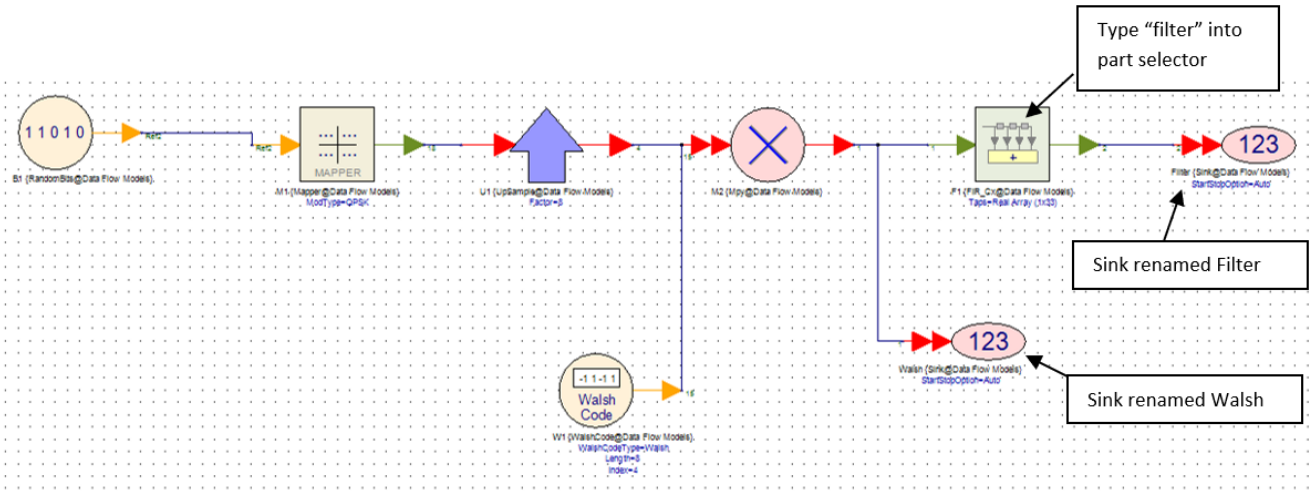


Figure 9.18. Modify the design as shown here.

5. Double click on the filter part. This should bring up the design properties. Select the Filter Designer button. Change the properties to match those in Figure 9.19. Close the property page once you have completed the changes.

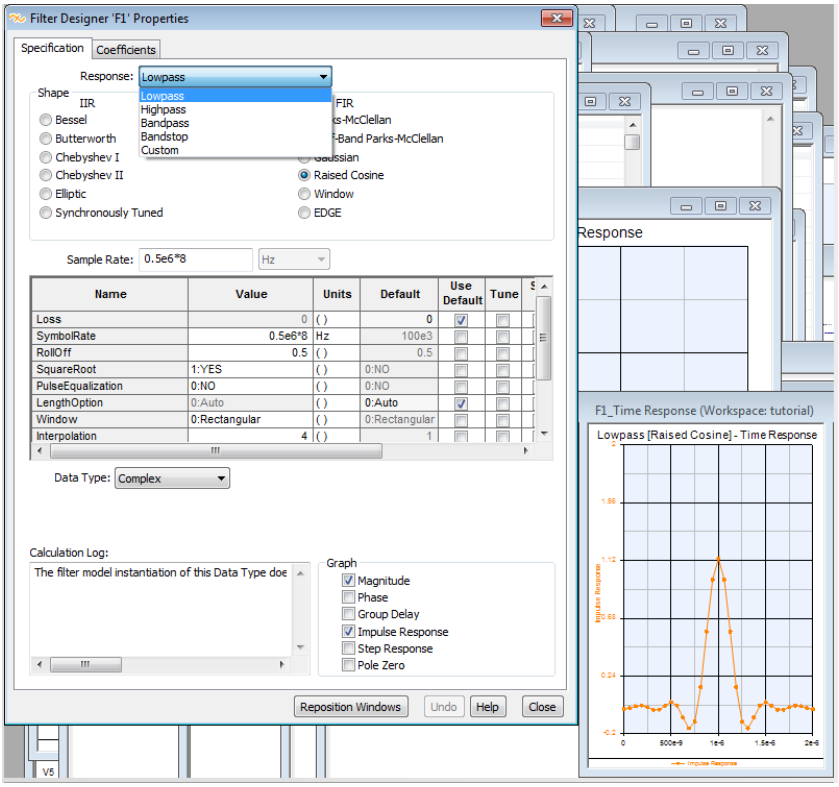


Figure 9.19. Design Properties menu.

6. Run the analysis
7. Double click on the filter dataset (DF1_Filter)>Right Click on Walsh>Add to Graph>New Graph Wizard>Spectrum>Press Ok two times to get the screen shown in Figure 9.20.

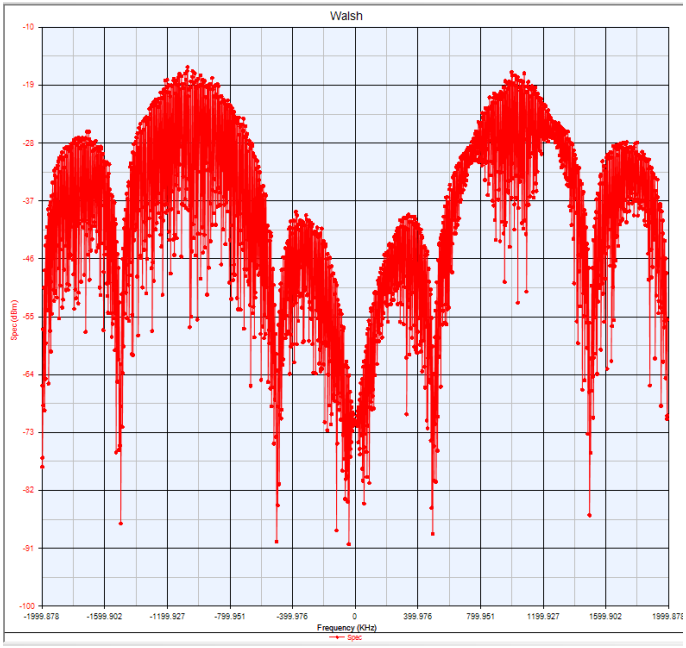


Figure 9.20. Resulting graph.

8. Now plot a spectra graph for “Filter.” Double click on the filter dataset (DF1_Filter)>Right Click on Filter>Add to Graph>New Graph Series Wizard>Spectrum>Press Ok two times. You can now see the effects of the filter (Figure 9.21).

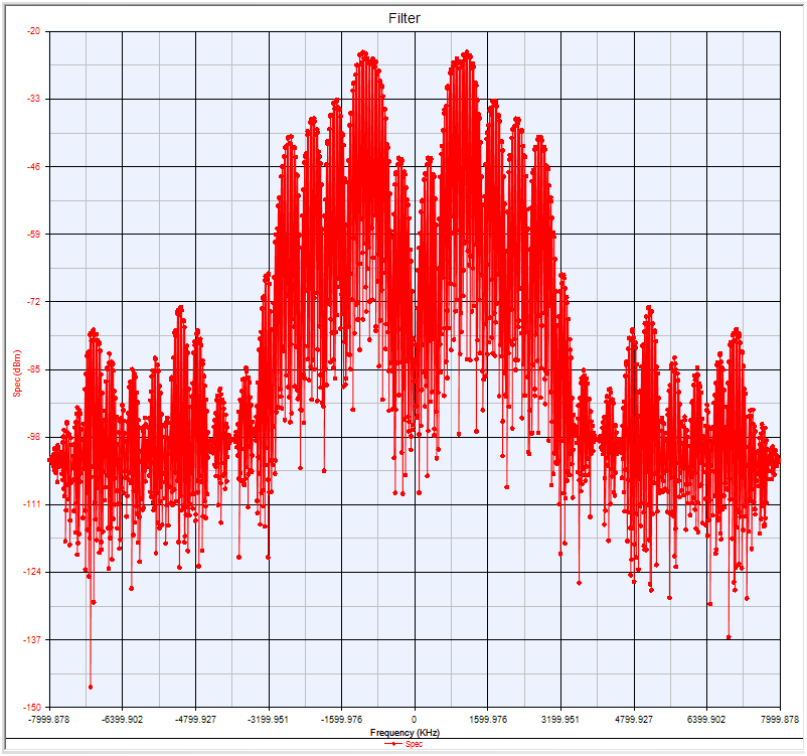


Figure 9.21. Spectra graph.

End of Lab 4.

9.6 Lab 5: Modulation

Create a new folder called “Step5_Modulation.” Add a new schematic called “Modulation.” With it, add a data flow analysis (“DF_Modulation”) and a dataset (“DF1_Modulation”). Set Number of Samples to 4096 of data flow analysis.

1. Copy the schematic created in Step 4 and paste it in the new blank schematic.
2. Delete the “output” sink.
3. In part selector, search for component “cxt” and select the part “Cxt To Rect.” This will separate the real and imaginary parts of the signal.
4. Place it adjacent to the output of the filter.
5. Add a modulator (type “modulator” in the part selector library).
6. Place it at the output of the complex to real and imaginary convertor.
7. Place a noise density part by typing “noise density” into the part selector. Add a spectrum analyzer at the output.
8. Modify the component parameters as shown on Figure 9.22.

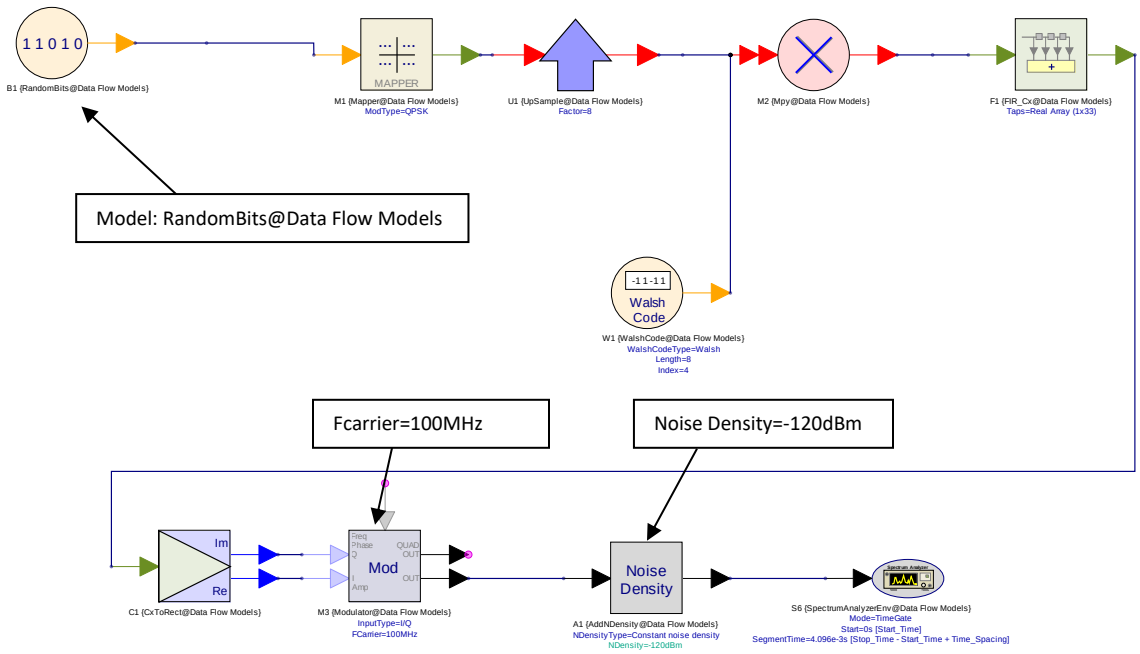


Figure 9.22. Place a noise density part in the schematic.

9. Let's see the power spectrum using a spectrum analyzer in PathWave System Design (SystemVue). Double click on the dataset "DF1_Modulation" and right click on 'S6_Power' >Add to the Graph>New Graph Series Wizard>General>Click Ok twice (Figure 9.23).

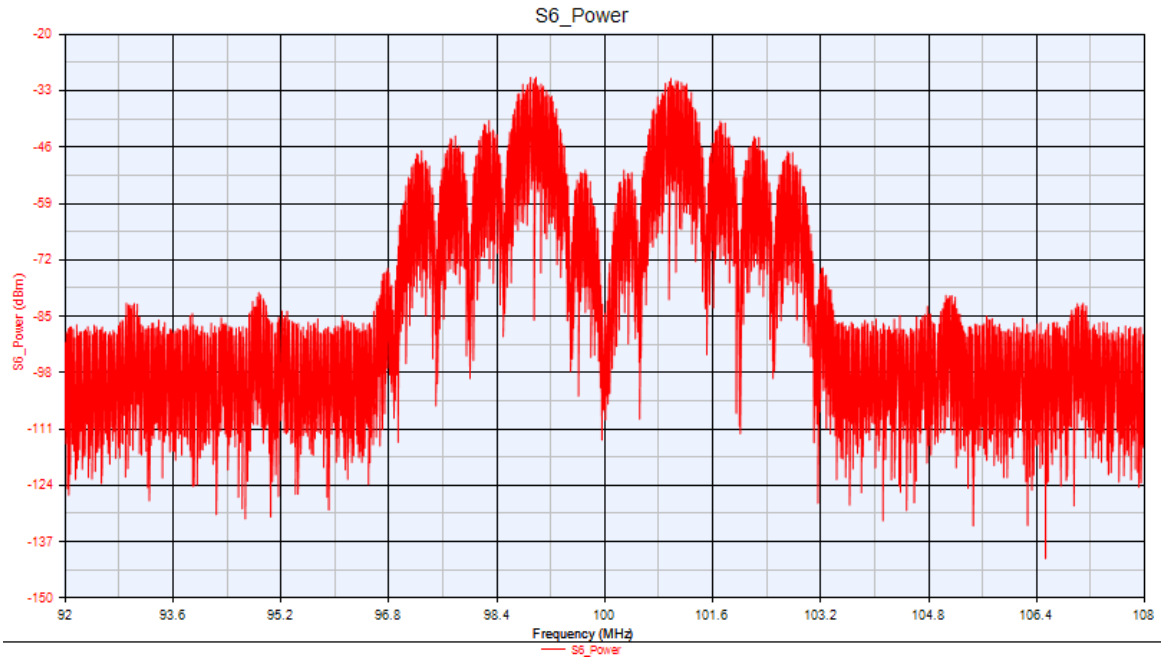


Figure 9.23. Power spectrum.

Lab 5 is now complete.

9.7 Lab 6: RF Link

In this lab we will design an RF Link for the modulated signal to transmit to the receiver.

1. Add a new folder for step 6> Call it "Step6_RFLink."
2. Create a folder called "Step6_RFLink" and insert Schematic (RFLink).
3. In blank schematic view, navigate to the part selector and scroll down the drop-down menu of "Current Library" and select "RF Design."
4. Leave the "category" displaying "<All>" to give us all parts.
5. Create the RF link to match the design below. You can type the following abbreviations into part selector to grab the part and form the design (Figure 9.24).

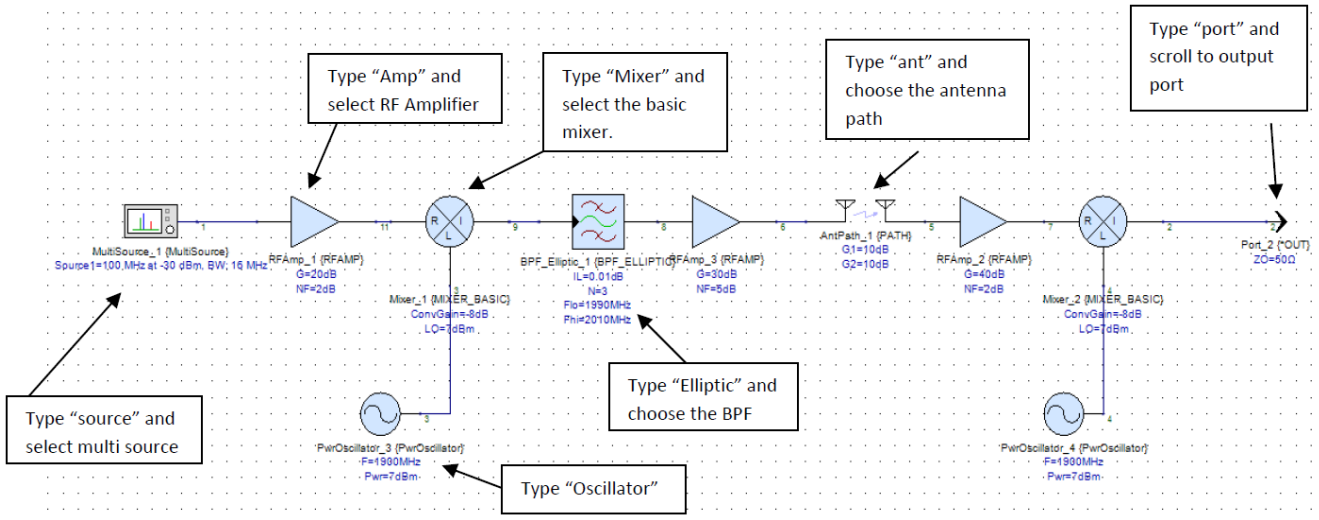


Figure 9.24. Design for the RF link.

Please use your mouse wheel to zoom into the page to set parameters. Below is each setting in detail.

For parameters:

- **MultiSource**: Double click on part and in the Source Summary box click "edit." Ensure the radio button is set to "wideband" and a bandwidth of 16 MHz is given. For parameters, set the centre frequency to 100 MHz and the power to -30 dBm. Keep the phase at 0 deg.
- **RF Amplifier**: Double click on part. Default values are set. Change the noise figure to 2 dB. Double check that the gain is at 20 dB.
- **Mixer**: Default values should be set to -8 dB for the gain and 7 dBm for the LO. Set the SUM parameter to 1:Sum.
- **Oscillator**: Set the frequency to 1900 MHz. The power should be default at 7 dBm.
- **BPF**: Set the lower pass band edge frequency (Flo) to 1990 MHz. Set FHi to 2010 MHz. The insertion loss should be already set to 0.01 and the filter order at 3.

- **RF Amp at output of BPF**: Set the gain to 30 dB and the noise factor to 5 dB.
- **AntPath**: Set G1=10dB and G2=10dB.
- **RF Amp at output of antenna path**: Set the gain to 40 dB and the noise factor to 2 dB.
- **Mixer after RF Amp**: Set ConvGain to -8 dB and LO is 7 dBm. Also, set the SUM parameter to 1:Sum.
- **Power Oscillator**: Set the frequency to 1900 MHz and the power to 7 dBm.

Add a port at the final output and set the impedance to 50 Ohms.

6. Right click on Step6_RFLink folder> Add> Add data (for dataset) and call it "System1_Data."
7. Right click again on Step6_RFLink > Add > Analysis > Add RF System Analysis.
8. Double click on the analysis (Figure 9.25).

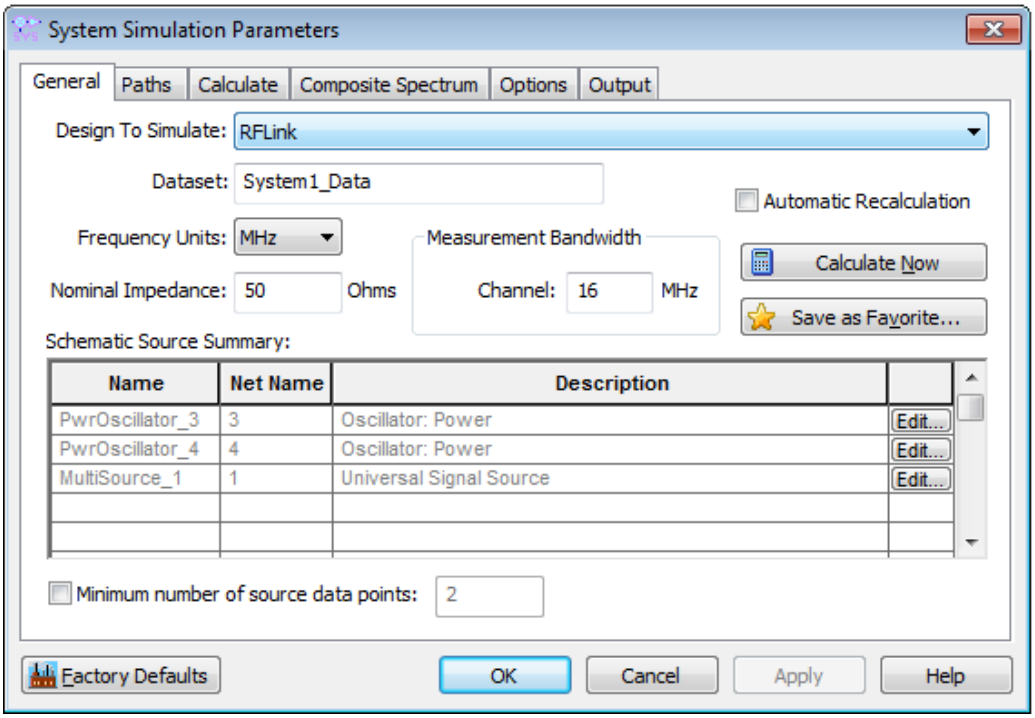


Figure 9.25 System Simulation Parameters window.

In the general tab, ensure that RFLink is present in the design to simulate field and that the dataset chosen is "System1_Data."

9. Move across to "Paths" tab. Select "Add all paths from all sources."
10. Press Ok.
11. Run the simulation by right clicking on System1 (RFLink)> Run. You may encounter an additional folder "System1_Data_Folder" that is created.

In schematic view, hover your cursor over the net values. Let's take the first net value placed at the output of the multisource. If you place your cursor over the net value, you should see a right angled cursor adjacent to your arrow.

12. Right click (Figure 9.26).

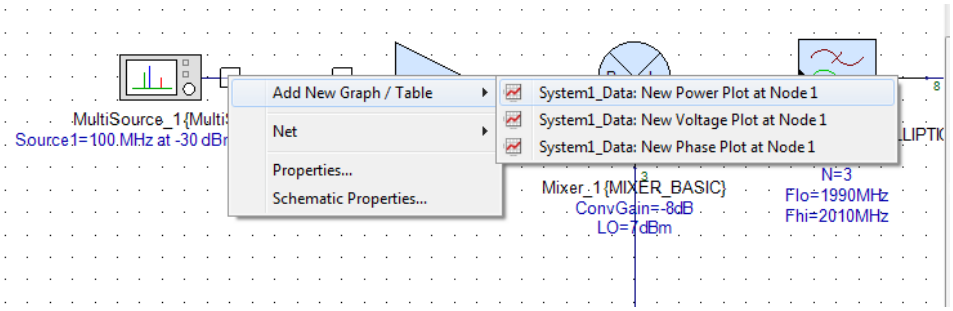


Figure 9.26. In schematic view, select the first net value placed at the output of the multisource.

13. Add New Graph/Table>Select to graph a power plot (Figure 9.27).

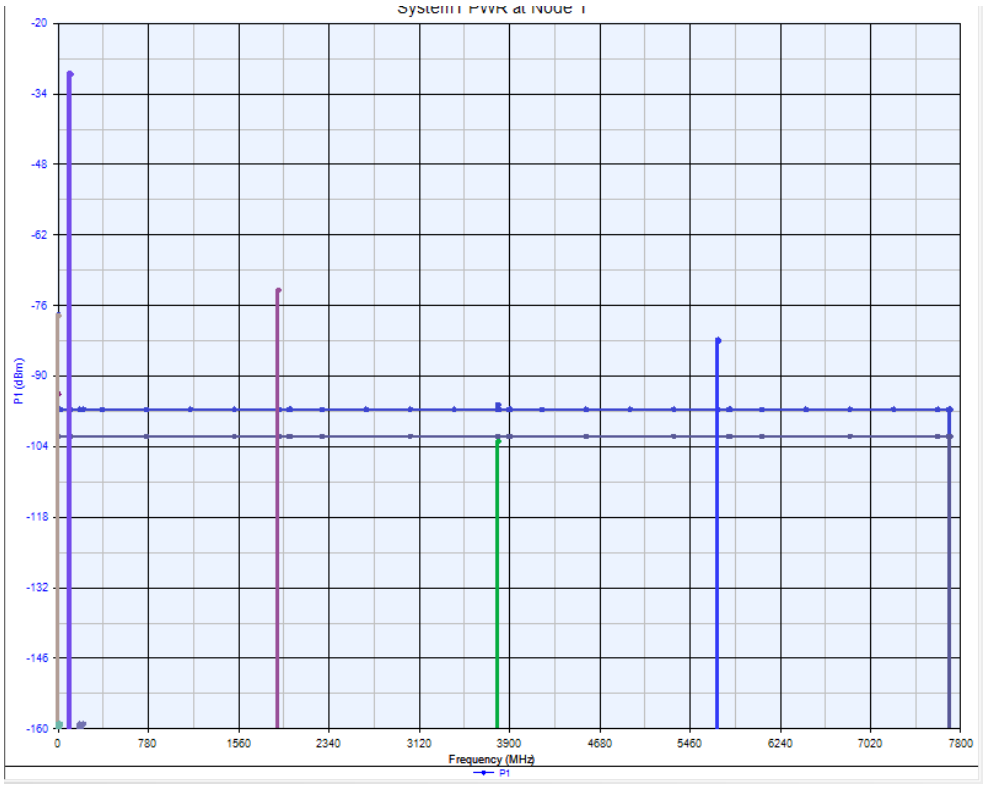


Figure 9.27. Power plot.

Here we can observe how clear our system is and if any interference is present. We can see our data signal at around 100 MHz (set by our source, this is the center frequency and is set quite high during RF communication channel). We can also observe from this power plot other frequency components and we can confirm that there are no signals interfering with our data at this point.

14. Capture the output port power (Figure 9.28).

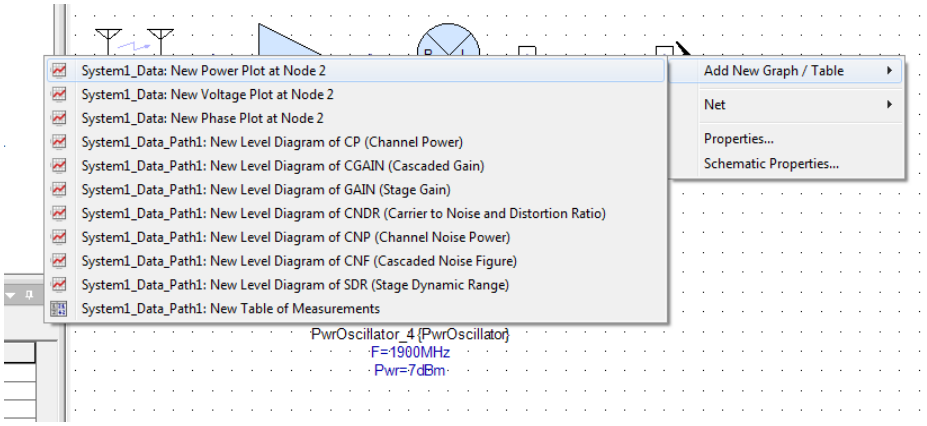


Figure 9.28. Capture the output port.

Plot the new power plot (Figure 9.29).

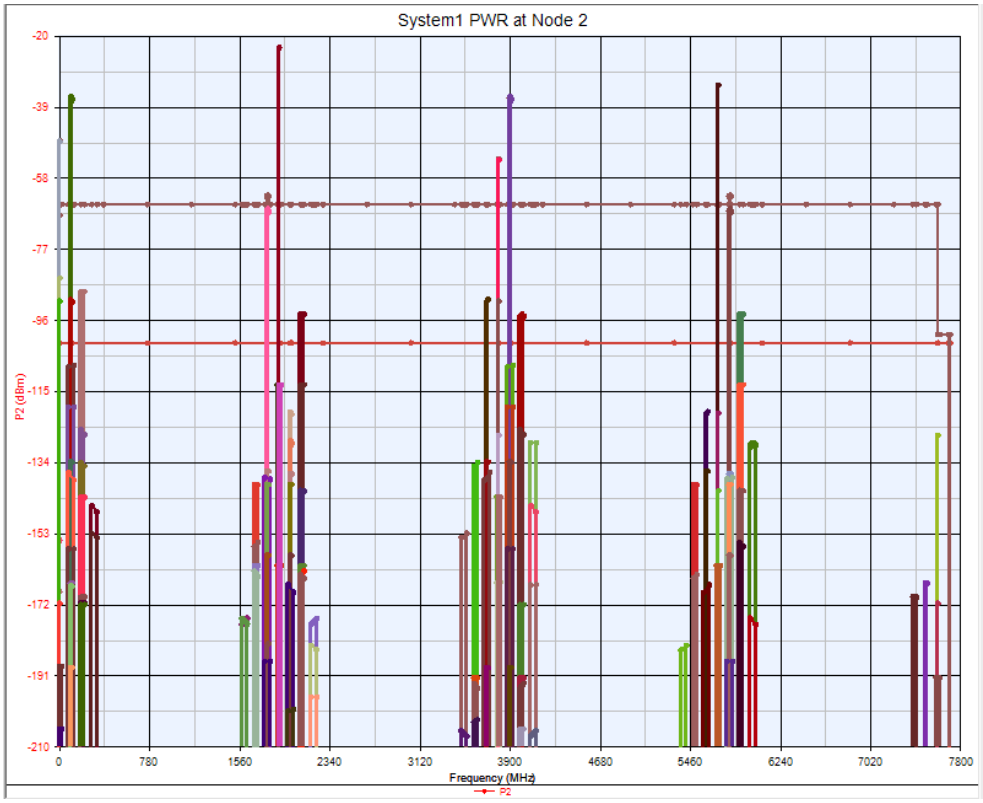


Figure 9.29. Power plot.

We can see the strength of our signal here at around 1900 MHz, which is the carrier frequency. We observe the harmonics around our carrier wave. In addition, we can see other frequency components at higher and lower frequencies, showing the reality of transmission system. We can also conclude that our signal is still strong after being transmitted over the communication channel, and therefore, we can proceed to demodulating the signal and retrieving our original transmitted data; hopefully with no errors!

Lab 6 is now finished.

9.8 Lab 7: Step 7 Demodulator

In this lab we will demodulate the signal and design the receiving side of the system.

1. Create a new folder called “Step7_Demodulation”>. Add a new blank schematic and call it “Demodulation”>. Add a data flow analysis and name it “DF_Demodulation.” Finally create a dataset called “DF1_Demodulation.”
2. Copy the design created in Step5_Modulation and paste the parts into the new blank schematic “Demodulation.”
3. In the workspace tree, expand the folder “Step6_RFLink.” Single click on the RFLink schematic and drag and drop (anywhere in the design) into the current “demodulation” schematic.

   A neat part of PathWave System Design (SystemVue) is that new names can be edited to a name of your choice and can be joined without a physical wire, keeping your designs tidy and organized.

4. Click on the wire between “Noise density” and “Spectrum Analyzer,” (Figure 9.30).

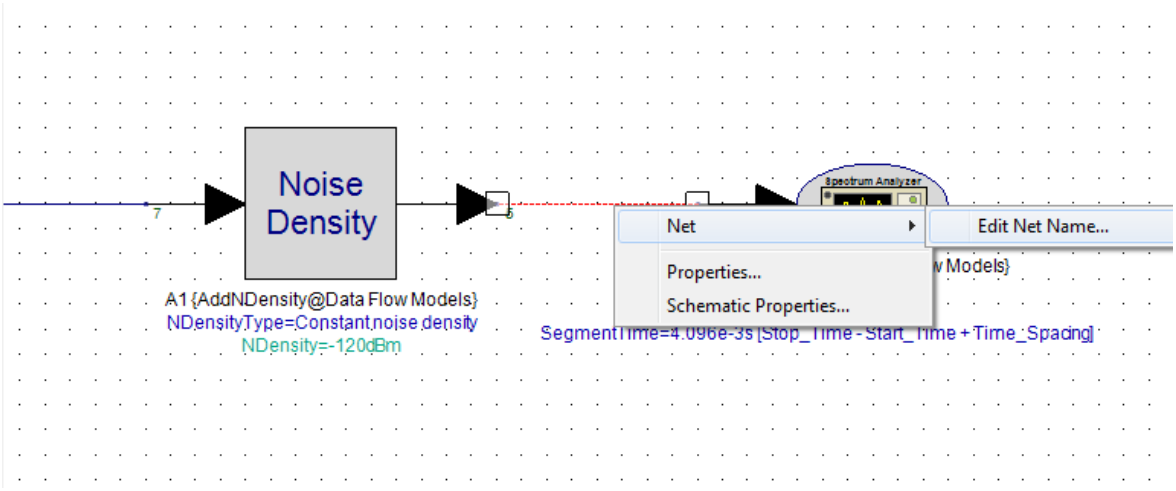


Figure 9.30. Edit net name.

5. Right click on the wire> Net> Edit Net Name. Give the new name: “IF_in.”

   On the left hand side of the RF_Link, create a wire to allow us to pick the net and give it a name (Figure 9.31).

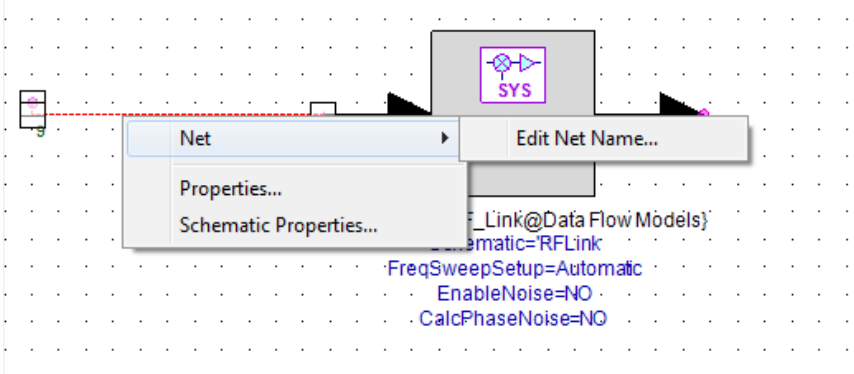


Figure 9.31. Create a wire, pick the net and name it.

Give the wire the same name as the wire output from the noise density ("IF_in"). This will join the two parts together. Noise Density and the RF_Link are now connected.

6. Create a demodulation system. Fundamentally this will be a mirror of the modulation system created.
7. Add a spectrum analyzer after the RF_Link. You may need to change the "Current Library" to Algorithm Design in the part selector. We will observe the power decrease following transmission. Copy the respective parts from Step5_Modulation as the parameters will have been saved too (Figure 9.32).

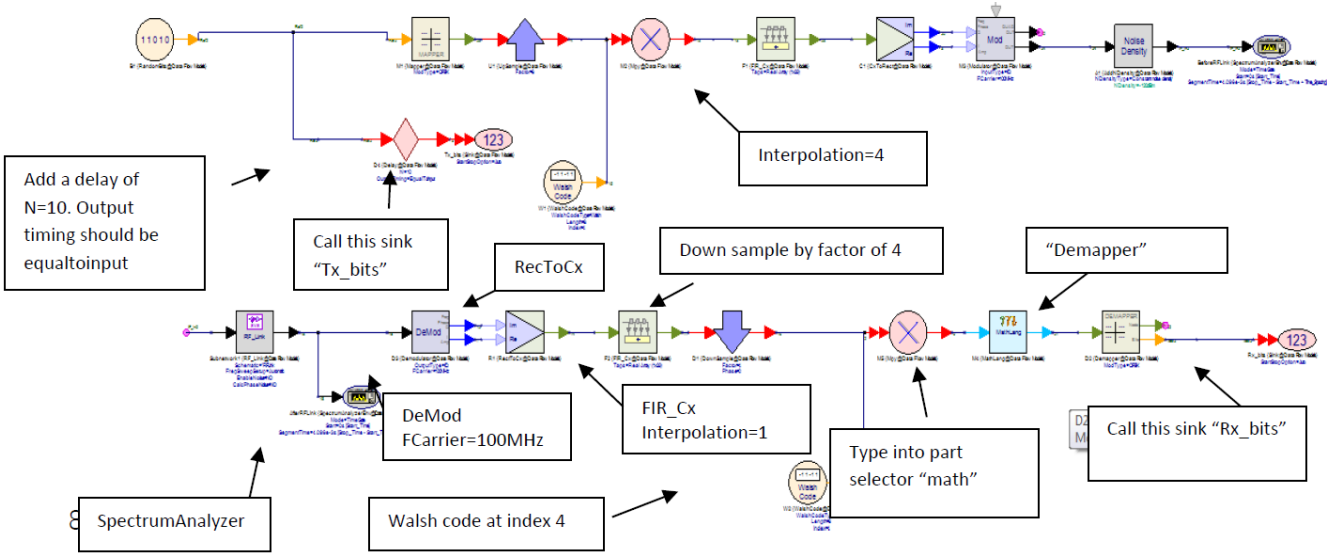


Figure 9.32. Adding a spectrum anayzer.

The Math Lang model uses math language equations to process input data and produce output data. We can use this to ensure that our sample rate is correct for demapping.

8. Double click on the math lang part and in the Equations tab copy that shown in Figure 9.33. (output=sum(input)/8)

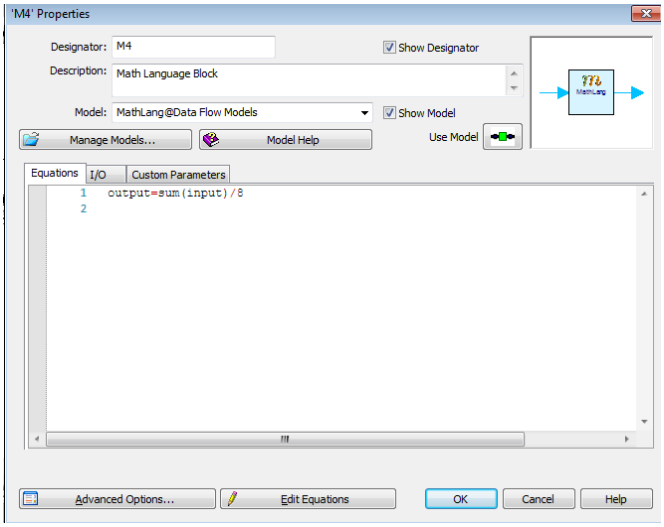


Figure 9.33. Copy the Equations tab information.

In the "I/O" tab, set the number of input port rates to 8 (the output is left as default at 1).

9. Click Ok.

10. Run the simulation, ensuring that the data flow analysis has the correct values (Figure 9.34).

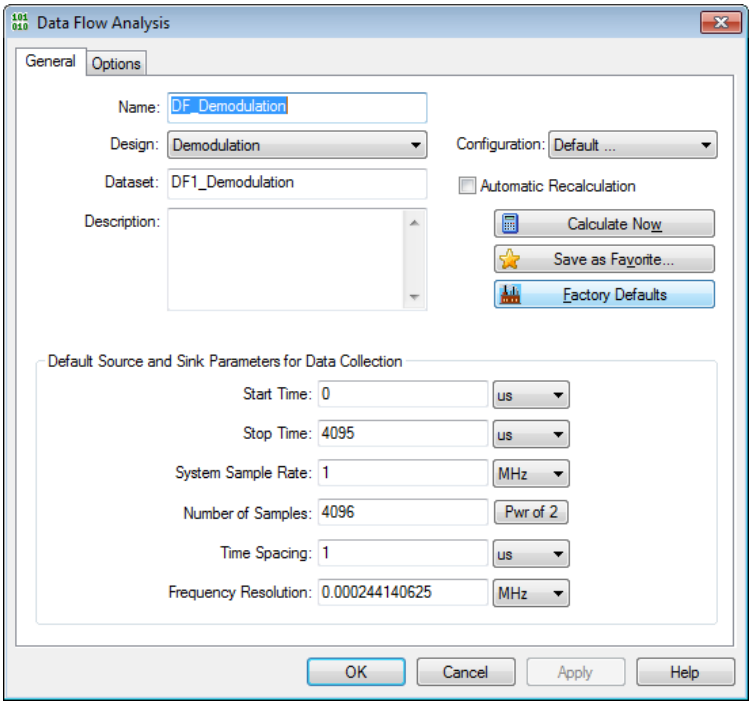


Figure 9.34. Use the values shown for simulation.

Let's view the power on the spectrum analyzer before the RFLink and after it in which the data stream travels across the RF channel.

11. Click on dataset and create a graph of the power spectrum of the signal before RFLink Double click on the dataset “DF1_Demodulation” and right click on ‘S6_Power’ >Add to the Graph>New Graph Series Wizard>General>Click Ok> On the Properties window select the Y-Axis tab, change units to W> Click Ok (Figure 9.35).

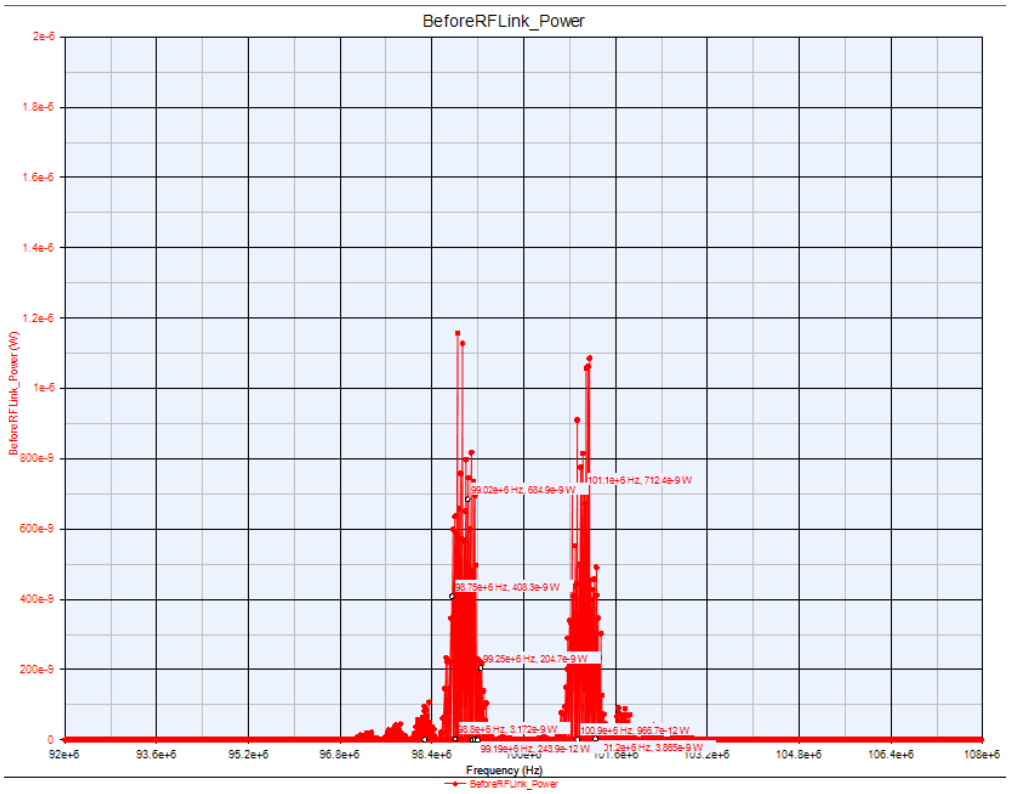


Figure 9.35. Power spectrum graph before the RFLink.

12. Now let’s see the power spectrum after our data is streamed across the RF channel (Figure 9.36).

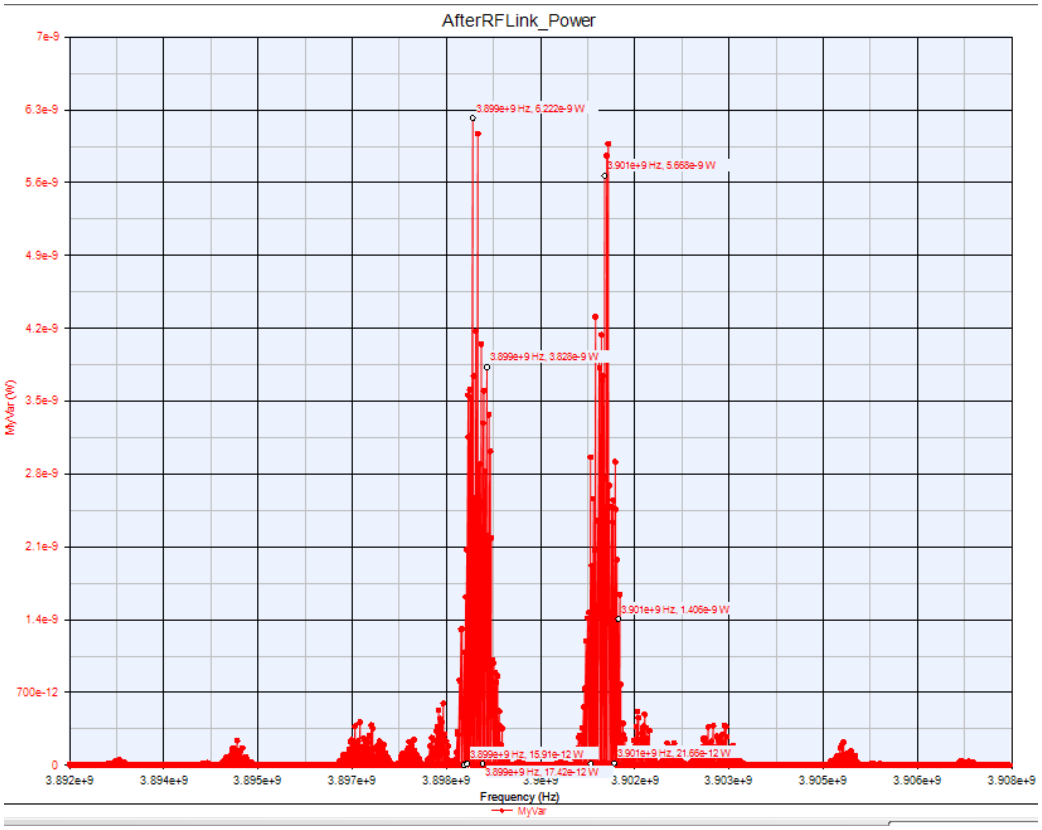


Figure 9.36. Power spectrum graph after the RFLink.

We can see that the power at 10 MHz, before our data is modulated to a carrier, is around 1.18e-6W. After the signal is passed through the RFLink, the power is dropped to around 6.3e-9W. This power loss simulates real-life conditions as the signal travels in air between the transmitting and receiving antenna, and signal power maybe lost to trees, buildings and other general surroundings. We can also see the sidebands; it is safe to say that this will not affect our data at the receiving end as the Walsh code will reconstruct the information above the noise floor.

Now let's see the individual bits and observe how good the transmission is.

13. In dataset, instead of producing a graph, conduct a table view of the transmitted bits and compare that with the received bits (Rx_bits and Tx_bits), as shown in Figure 9.37.

    Need step by step instructions? Refer back to Lab 3 where we produced a table to view our Walsh codes.

Index	Rx_bits	Tx_bits
1	0	0
2	1	0
3	1	0
4	0	0
5	1	0
6	0	0
7	1	0
8	0	0
9	0	0
10	1	0
11	0	0
12	1	1
13	0	0
14	0	0
15	0	0
16	0	0
17	0	0
18	0	0
19	0	0
20	0	0
21	0	0
22	0	0
23	0	0
24	0	0
25	0	0
26	0	0
27	1	1
28	0	0
29	0	0

Figure 9.37. Comparing transmitted bits to received bits.

We can conclude from the bits received that our system performs well.

9.9 Lab 8: Completed System

In this lab we will calculate the Bit-Error-Rate (BER) to test its performance.

1. Add a new folder “Step8_CompletedSystem.”
2. In the folder create a schematic “CompletedSystem”, create a dataset “DF1_CompletedSystem”, and a data flow analysis “DF_CompletedSystem” (set Number of Samples to 4096)
3. Copy the components in the schematic created in Step 7 and paste it in the new blank schematic.
   Let’s cross correlate our two signals. This will allow us to compare the two signals and see the degree of similarity between them. In our CDMA system, it will compare the sent code and the saved code at the receiver produced by our Walsh code generator. If the codes are similar then the receiver will decode it and produce the original transmitted signal.

4. In blank space within your schematic, add a cross correlator estimator Part Selector> type "cross" and select the cross correlator part.
5. Add two sinks, one at the delay (call it "Delay_Estimate") and one called "Cross_Estimate."
6. Add a wire at each of the inputs and edit the names. Call the y input "Test" and the x input "Ref." This can be achieved by double clicking on the wire and editing the default string (Figure 9.38).
7. Correspondingly name the output wire of the Random Bit Generator "Ref" in your design and the output wire of your system "Test." (Figure 9.38)

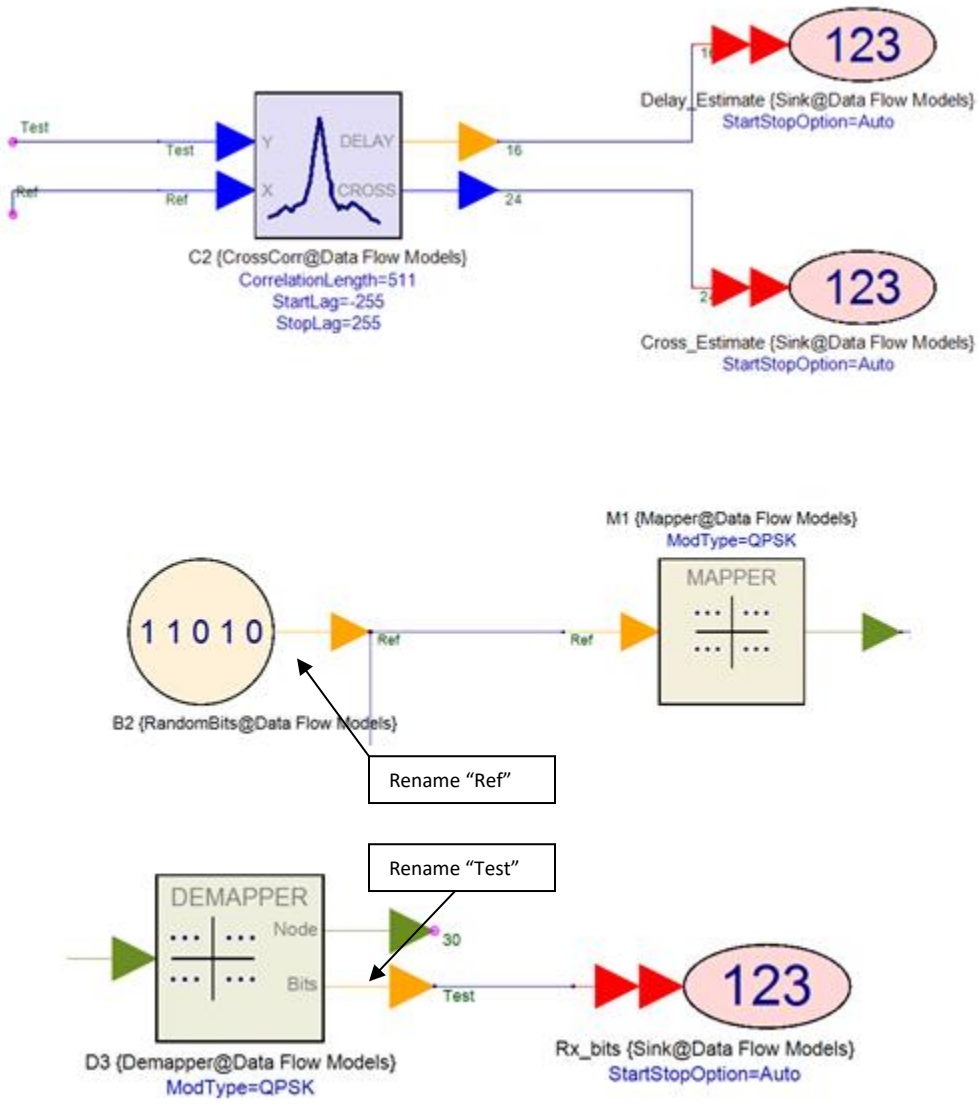


Figure 9.38. Adding a wire of each of the inputs and editing the names.

8. Run the simulation.
9. Plot the delay estimation graph as a cross correlation graph (Dataset>Delay_Estimate>New Series Graph Wizard>Cross Correlation).
10. Now plot the cross estimation graph (Figure 9.39).

Figure 9.39. Cross estimation graph.

We can imagine the transmitted signal and the signal saved at the receiver (by our Walsh code generator) slide over one another and as they do, the codes become increasingly matched and the cross-correlation peaks form a triangle-like graph. The peak represents the offset, which is the point at which the codes are nearly perfectly matched.

Let's now test our performance by employing the BER method. BER is used to measure the performance of a communication system as it is the rate at which errors occur in a transmission system. The BER is defined in Equation 9.1 as:

$$BER = \frac{No.of\ bit\ errors}{Total\ no.of\ bits\ transmitted} \qquad \text{E.9.1}$$

If the medium between the transmitted and receiving sides of a system is good then the signal-to-noise ratio will be high and the BER will be low. If a system has lots of noise, and coding techniques don't match—in our case, the Walsh coding scheme—then the bits received differ greatly from the original transmitted bits and results in a high BER.

1. Choose a blank space within your schematic.
2. Type "BER" into part selector and inject the relevant part into your design. The parameter SampleStart on the BER_FER model should be set to 10, BitsPerFrame equal to 1000, and StartStopOption = Samples.
3. Add a delay of N=10 to the REF input terminal. This delay is added to ensure that the bit streams are synchronized, otherwise the BER will be wrong.
4. Add wires at both inputs so that we can name the wires. Connect the TEST input to our TEST signal. This can be achieved by naming the wire "TEST." Give the other wire at the REF input the

name “REF.” This will allow the BER measurer to calculate the bit-error-rate of the transmitting signal and the end receiving signal, after the Walsh code has decoded the signal, (Figure 9.40).

5. Run the simulation

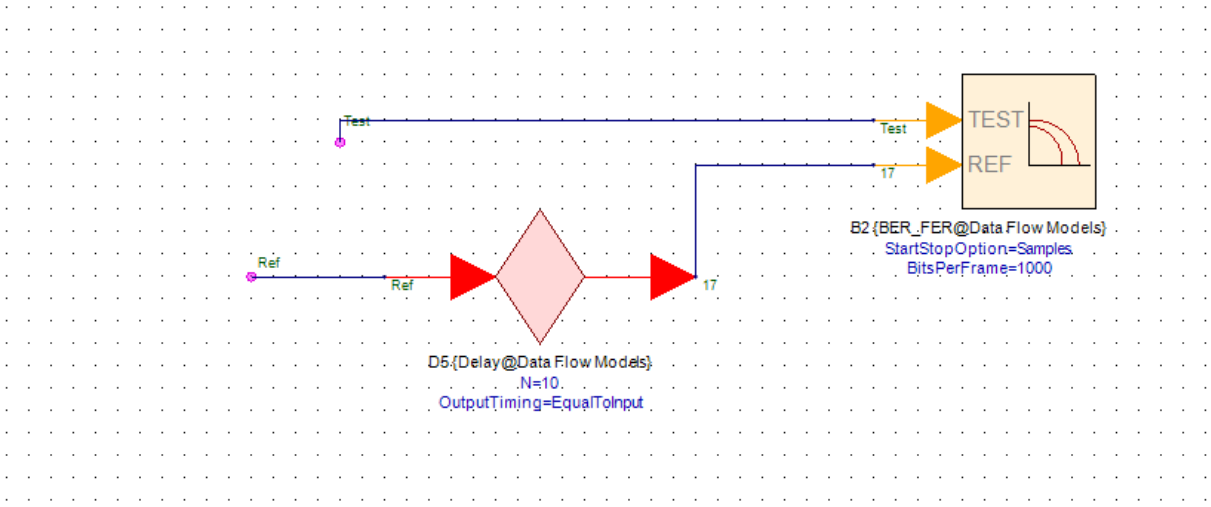


Figure 9.40. Enabling the BER measurer to calculate the BER of the transmitting signal and the end receiving

Now let’s set up BER equations and display our result in a text box.

6. Click on Equation1 in the Workspace Tree (Figure 9.41). In the coding box type in the following (B3_BER represents the BER variable in the dataset):
    1. using('DF1_CompletedSystem');

2. BER = B3_BER;

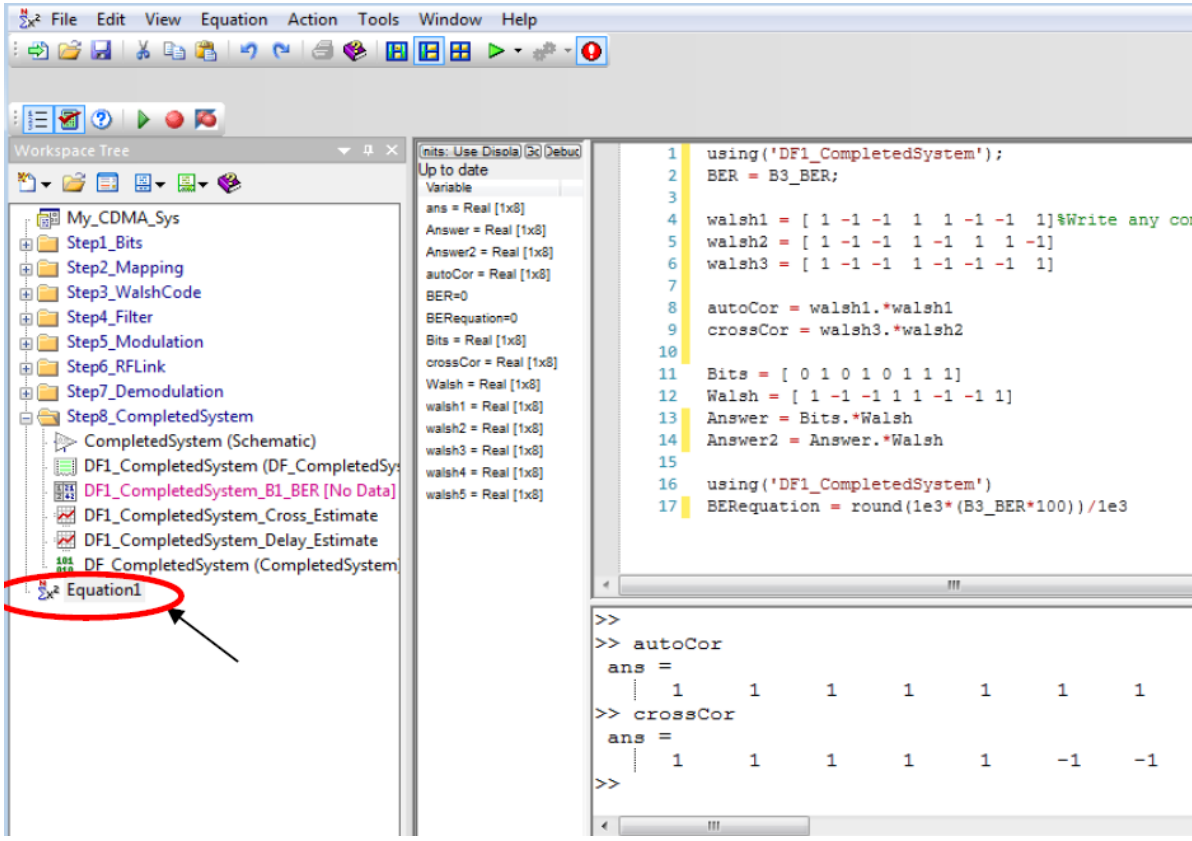


Figure 9.41. Inserting BER Equation.

7. Click on "Show/Hide Annotation Toolbar" situated in the top corner of the PathWave System Design (SystemVue) application (Figure 9.42).

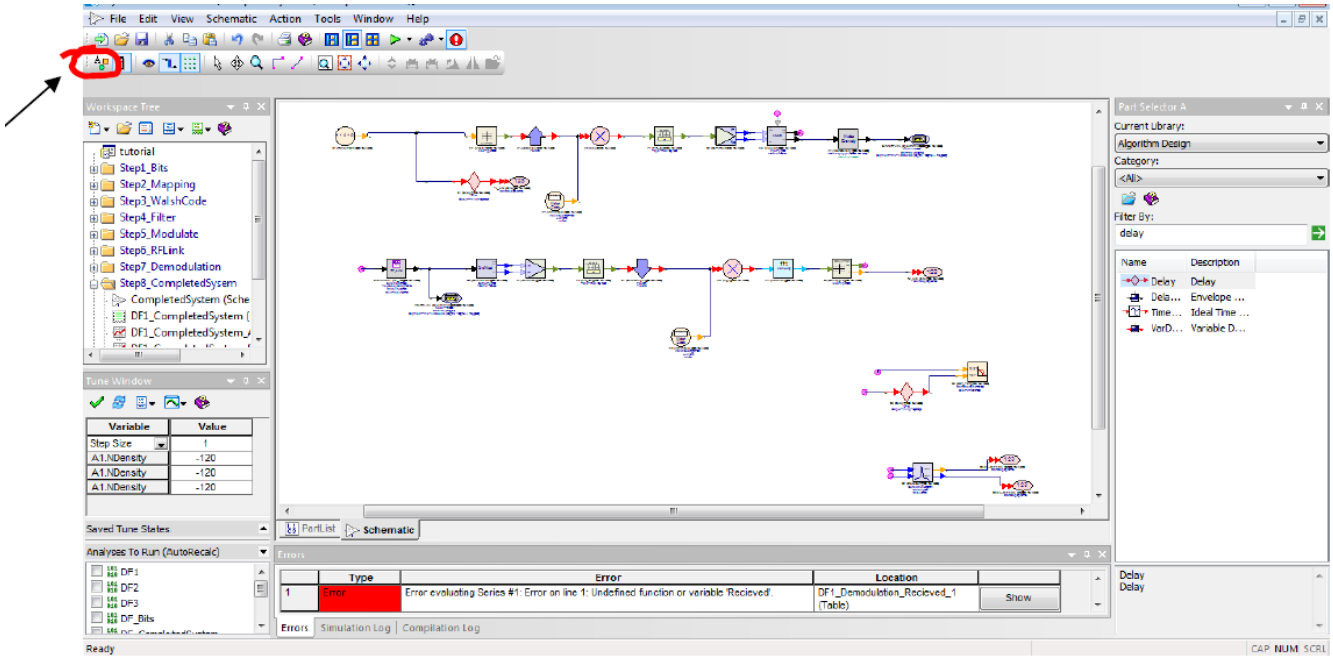


Figure 9.42. PathWave System Design (SystemVue)'s Show/Hide Annotation toolbar.

A tool bar should appear, click on the square text box, as shown in Figure 9.43.

Figure 9.43. Square text box.

Create two text boxes side by side; one to display the value of our BER and the other to Name the varialble (Figure 9.44).

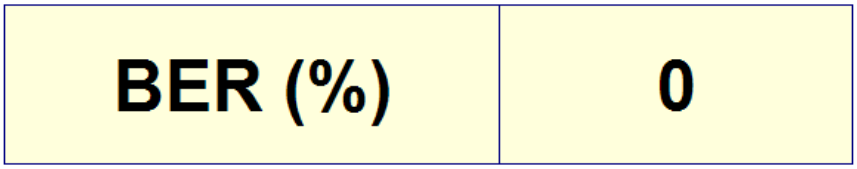


Figure 9.44. Two side-by-side text boxes.

8. Double click on the text box on the right and in the text space key in "=BER," as shown in Figure 9.45.

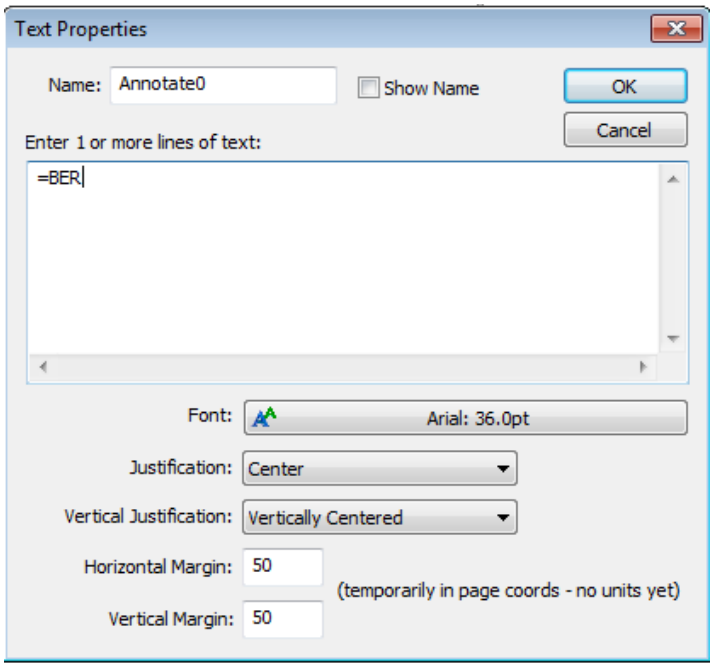


Figure 9.45. Text properties page.

Here you can change the position of the text, size, font, and color to suit your requirements.

9. Now, double click on the text box on the left and in the text space type "BER (%)."
   After clicking ok, if your transmitting Walsh code generator index matches the Walsh code generator at the receiving end, then you should have a 0% BER. This shows that our performance is perfect with all data transmitted having been successfully decoded by the Walsh code and received.

Let's change the index of one of our Walsh codes and see what happens.

10. Click on one of the Walsh code generators and change the index to any number up to 8 (a different number to what is already set).
11. Run the simulation.

Notice that your BER changes to above 0%. Try changing the Walsh index once again and continue to run and observe the changes in the BER.

We have completed our CDMA Spread spectrum and tested its performance using the BER in PathWave System Design (SystemVue) software.

As a final exercise, let's use the equations feature of PathWave System Design (SystemVue) to observe the actual bits, Walsh codes and an alternative way to display BER. This exercise will use math language equations to process output data.

1. In the workspace tree > Right click on "tutorial" > Add > Add Equation.

Let's make some example Walsh codes.

2. In the coding box, starting on line1, copy the following (Figure 9.46).

```
1  walsh1 = [ 1 -1 -1 1 1 -1 -1 1] %Write any comments here
2  walsh2 = [ 1 -1 -1 1 -1 1 1 -1]
3  walsh3 = [ 1 -1 -1 1 1 -1 -1 1]
```

Figure 9.46. The "%" enables you to write comments.

Notice that each Walsh code has equal numbers of 1s and -1s.

Notice the variables will be set on the left-hand side tree.

3. Press Go to compile the code. This needs to be pressed after a new variable is created.
4. In the command window (bottom of screen), type any one of the Walsh code variables such as Walsh1. Please note that commands are case sensitive.

```
>> walsh1
      ans =
           1  -1      -1       1       1       -1       -1       1
>>
```

Figure 9.47.Type a Walsh code variable, such as Walsh1.

Let's now employ correlation techniques.

5. Auto correlate walsh1 by typing in the equation below into the coding box
   - autoCor = walsh1.*walsh1
6. Cross correlate walsh3 with walsh2 by typing in the equation below into the coding box
   - crossCor = walsh3.*walsh2

Observe the corresponding values via command window (Figure 9.41). Notice that autocorrelation produces a single entity of 1s. Here we can prove that autocorrelation, which is the same as cross correlation with the same code, will mean that the data can be decoded and information can be retrieved (since there is a match). Cross correlating two different Walsh codes produces a mismatch as a

series of 1s and -1s are created. Here there is a mismatch and the data cannot be decoded. Consequently, we have seen and proven that in order to decode and obtain the original data, the Walsh code generator at the transmitting end must produce the same Walsh code at the receiving end.

If we create a bit pattern, we can see what happens to our data as it is encrypted using Walsh code (Figure 9.48).

```
Bits = [ 0 1 0 1 0 1 1 1]
Walsh = [ 1 -1 -1 1 1 -1 -1 1]
Answer = Bits.*Walsh
```

Figure 9.48. Data encrypted using a Walsh code.

Check the resultant array using the command box (Figure 9.49).

```
>> Answer
      ans =
          0     -1     -0      1      0     -1     -1      1
>>
```

Figure 9.49. The resultant array.

If we now take this coding and correlate it with our Walsh code ("Walsh"), we can retrieve our original bit pattern (Figure 9.50)

```
Bits = [ 0 1 0 1 0 1 1 1]
Walsh = [ 1 -1 -1 1 1 -1 -1 1]
Answer = Bits.*Walsh
Answer2 = Answer.*Walsh
```

Figure 9.50. Original bit pattern.

Check the resultant array using the command box (Figure 9.51)

```
>> Answer
      ans =
          0     1     0     1     0     1     1     1
>>
```

Figure 9.51. The resultant array.

Using the equations feature, we can view our BER result straight from the command window. Using this code:

```
using('DF1_CompletedSystem')
BER = round(1e3*(B2_BER*100))/1e3
```

This will give us the BER as a percentage.

9.10 Conclusion

Hopefully, this Spread Spectrum CDMA system building exercise was instructive and that you acquired all the necessary tools to design, simulate, display, and analyze results in the various forms explored; time plots, eye diagrams, spectra plots, power diagrams, frequency domain graphs, and BER. This information should equip you to build more complex systems.

PathWave System Design (SystemVue) has an extensive library of examples, tutorial videos and web forums, all of which can be accessed via its help menu.

Other tutorial and workshop material is available to help you expose the benefit of using PathWave System Design (SystemVue) for your design application. Please refer to our EEsof homepage or contact us for more information on trials and licenses at http://www.keysight.com/find/eesof

Appendix

Some useful information and links to key areas discussed in this tutorial are as follows:

- Keysight application note 1298.This application note provides an introduction to digital modulation concepts, discusses different digital modulation types and gives you an understanding of how digital transmitters and receivers work. Access it at: http://literature.cdn.keysight.com/litweb/pdf/5965-7160E.pdf

- More information about PathWave System Design (SystemVue) is available at:
  - Product information http://www.keysight.com/find/eesof-systemvue
  - Product Configurations http://www.keysight.com/find/eesof-systemvue-configs
  - Request a 30-day Evaluation http://www.keysight.com/find/eesof-systemvue-evaluation
  - Downloads http://www.keysight.com/find/eesof-systemvue-latest-downloads
  - Helpful Videos http://www.keysight.com/find/eesof-systemvue-videos
  - Technical Support Forum http://www.keysight.com/find/eesof-systemvue-forum

References

`1. C. Hsu, S. Ramasubbu, M. Ko, J. L. Pino, and S. S. Bhattacharyya, “Efficient Simulation of Critical Synchronous Dataflow Graphs,” Design Automation Conference Proceedings, San Francisco, California, July 2006.

2. C. Hsu, J. L. Pino, F. Hu, “A Mixed-Mode Vector-Based Dataflow Approach for Modeling and Simulating LTE Physical Layer,” Design Automation Conference Proceedings, Anaheim, CA, June 2010.

3. Jeffery A. Weldon, “High Performance CMOS Transmitters for Wireless Communications,” PhD Dissertation, University of California at Berkeley, 2005.

4. V. W. Leung, L. E. Larson, and P. Gudem, “An Improved Digital-IF Transmitter Architecture for Highly-Integrated W-CDMA Mobile Terminals,” pp. 1335-1339, 2003.

5. Behzad Razavi, “RF Transmitter Architectures and Circuits,” IEEE CICC, pp. 197-204, 1999.

6. Moumita Mukherjee, “Advanced Microwave and Millimeter-Wave Technologies Semiconductor Devices, Circuits and Systems,” pp. 495-520, 2010.

7. R. Svitek, S. Raman, “DC Offsets in Direct-Conversion Receivers: Characterization and Implications,” IEEE Microwave Magazine, pp. 76-86, September 2005.

8. Asad A. Abidi, “Direct-Conversion Radio Transceivers for Digital Communications,” IEEE Journal of Solid-Sate Circuits, volume 30, no. 12, pp. 1399-1410 1995.

9. Christopher A. DeVries, Ralph D. Mason, “Subsampling Architecture for Low Power Receivers,” IEEE Transactions on Circuits and Systems, volume 55, pp. 304-308, 2008.

10 J. H. Mikkelsen, “Front-End Architectures for CMOS Radio Receivers,” Division of Telecommunications, Aalborg University, pp. 1–18, 1996.

11. *Keysight Fundamentals of RF and Microwave Noise Figure Measurements*, Application Note 57-1, Keysight Technologies, 2010.

12. Bernard Sklar, “Digital Communications: Fundamentals and Applications,” Second Edition, Prentice Hall PTR, 2001.

13. TRANZEO Wireless Technologies, “Wireless Link Budget Analysis: How to calculate link budget for your wireless network,” Whitepaper, 2010.

14. Intersil, “Tutorial on Basic Link Budget Analysis,” Application Note, AN9804.1, June 1998.

15. A. Hajimiri, T. H. Lee, “A General Theory of Phase Noise in Electrical Oscillators,” IEEE Journal of Solid-State Circuits, Vol. 33, No. 2, February 1998, pp. 178-194.

16. John Pahl, “Spectrum Liberalisation and Interference Management,” International Telecommunications Union Website, 2006.

17. Qizheng Gu, “RF System Design of Transceivers for Wireless Communications,” Springer, 2005.

18. Salvatore, Ragusa, et al., “Adjacent Channel Power Ratio Analysis for an OFDM Signal,” IEEE International Symposium on Signal Processing and Information Technology, 2005.

19. Matt Loy, “Understanding and Enhancing Sensitivity in Receivers for Wireless Applications,” Wireless Communication Business Unit, Texas Instruments, 1999.

20. S. Maas, “Nonlinear Microwave and RF Circuits,” Second Edition, Artech House, 2003.

21. G. Breed, ”Basic Principles of Electrically Small Antennas,” High Frequency Electronics Magazine, pp. 50-53, 2007.

22. A. Hoorfar, V. Jamnejad,”Electromagnetic Modeling and Analysis of Wireless Communication Antennas,” IEEE Microwave Magazine, pp. 51-67, 2003.

23. D. M. Pozar, ”Microstrip Antennas,” Proceedings of the IEEE, vol. 80, no. 01, pp. 79-91, 1992.

24. F. H. Raab et al., “RF and Microwave Power amplifier and Transmitter Technologies – Part 1-4,” High Frequency Electronics, 2003.

25. G. Gonzalez, “Microwave Transistor Amplifiers: Analysis and Design,” Prentice Hall, Second Edition, 1996.

26. S. Winder, “Analog and Digital Filter Design,” Second Edition, Newnes, 2002.

27. R. W. Rhea, “HF Filter Design And Computer Simulation,” Noble Publishing, 1994.

28. D. M. Pozar, “Microwave Engineering,” Third Edition, John Wiley & Sons, 2005.

29. S. Maas, “Microwave Mixers,” Second Edition, Artech House, 1993.

30. A. Grebennikov, “RF and Microwave Transistor Oscillator Design,” John Wiley & Sons, 2007.

31. M. Odyniec, “RF and Microwave Oscillator Design,” Artech House, 2002.

32. I. J. Bahl, “Broadband Power Detectors,” IEEE Microwave Magazine, pp. 82-86, 2007.

33. *Diode Detector Simulation using Keysight EDA ADS Software*, Application Note 1156, Keysight Technologies, 2007.

34. P. E. Allen, D. R. Holberg, “CMOS Analog Circuit Design,” Second Edition, Oxford University Press, 2002.

35. D. Kester, “The Data Conversion Handbook,” Newnes, 2005.

36. *Understanding GSM/EDGE Transmitter and Receiver Measurements for Base Transceiver Stations and their Components*, Application Note 1312, Keysight Technologies.

37. K. Kundert, "Principles of Top-Down Mixed-Signal Design," Designer's Guide Consulting Inc., 2006, pp. 1–31.

38. G. Gielen, R. Rutenbar, "Computer-Aided Design of Analog and Mixed-Signal Integrated Circuits," IEEE Proceedings, vol. 88, 2000, pp. 1825–1852.

39. Frevert, et al., "Modeling and Simulation for RF System Design," Springer, 2005.

40. J. E. Chen, "Modeling RF Systems," Designer's Guide Consulting Inc., 2011, pp. 1–41.

41. Third-Generation Partnership Project – Technical Specification Group Radio Access Network, "Evolved UMTS Terrestrial Radio Access: User Equipment (UE) Radio Transmission and Reception," release 11, March 2012.

42. Third-Generation Partnership Project – Technical Specification Group Radio Access Network, "Evolved UMTS Terrestrial Radio Access: Base Station (BS) Radio Transmission and Reception," release 11, March 2012.

43. *LTE and the Evolution to 4G wireless – Design and Measurement Challenges*, Keysight Technologies Publication, 2009.

44. H. Holma, A. Toskala, "LTE for UMTS – OFDMA and SC-FDMA Based Radio Access," John Wiley & Sons, 2009.

45. S. Sesia, I. Toufik, M. Baker, "LTE – The UMTS Long Term Evolution: From Theory to Practice," John Wiley & Sons, 2009.

46. Motorola Inc., "Long Term Evolution (LTE): Overview of LTE Air-Interface," White Paper, 2007.

47. L. Tian, W. Xu, "LTE Front-End Scans Three Bands," http://mwrf.com/Article/ArticleID/23956/23956.html, Microwaves & RF.

48. *Testing and Troubleshooting Digital FR Communications Receiver Designs*, Application Note 1314, Keysight Technologies.

49. *Digital Modulation in Communication Systems - An Introduction*, Chapter 7, Application Note 1298, Keysight Technologies.

50. *Testing Next-Generation Optical Systems with Simulated OFDM Signals*, Application Note 5990-9972EN, Keysight Technologies.

Published in USA, January 31, 2020
5992-0197EN
www.keysight.com