Behavioral Simulation of a Basic OFDM Transceiver using the CppSim Program
Part I: Discrete-Time

Farinaz Edalat
http://www-mtl.mit.edu/~perrott

May 13, 2004
Revised: July 2008

Table of Contents

Setup 2
Introduction 5
A. Discrete-Time Transmitter Architecture 6
B. Channel Model 8
C. Discrete-Time Receiver Architecture 8
Performing Basic Operations within Sue2 and CppSimView 9
A. Opening Sue2 Schematics 9
B. Running CppSim Simulation 11
C. OFDM System Example 13
Plotting Frequency-Domain Results 14
A. Channel Frequency Response 14
B. Scatterplots (Constellations) of Transmitted and Received Symbols 15
Plotting Time-Domain Results 16
A. Transmitted and Received Bits 16
B. Error Bits and Total Bits 17
Examining Effect of AWGN 19
Channel Models 21
A. COST259-Model A 21
B. COST259-Models B and C 24
C. Channel Model for a Wideband OFDM System from COST259-Model A 24
D. Naftali Channel Model 26
Simulation Design Issues: 27
A. Clocking 27
B. BER and SNR Measurements 30
Conclusion 31
Acknowledgement 31
References 32

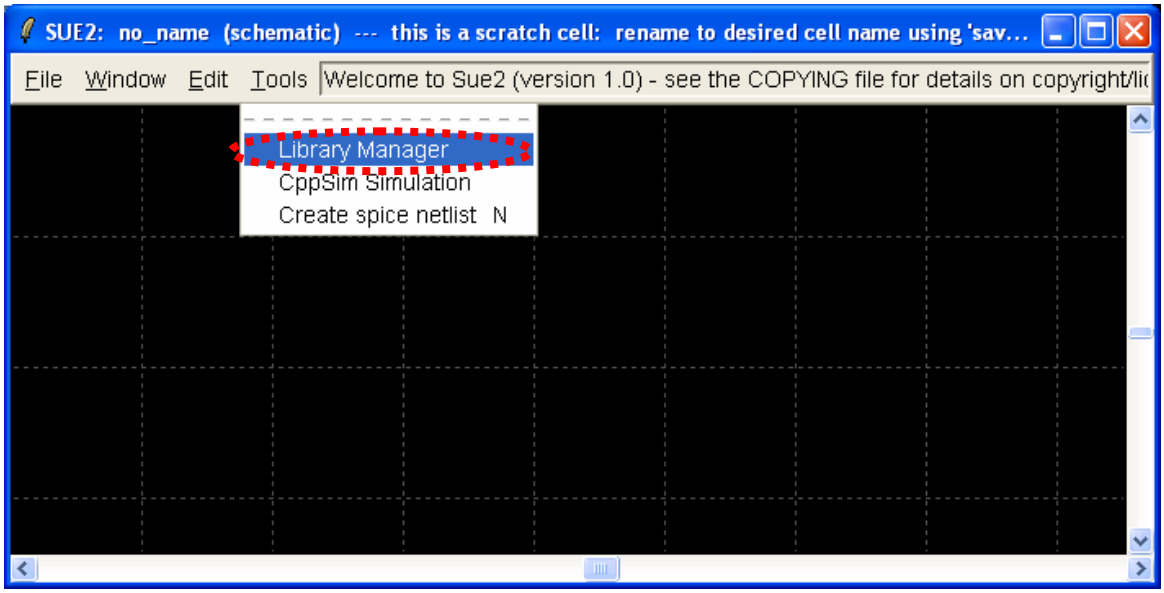
Setup

Download and install the CppSim Version 3 package (i.e., download and run the self-extracting file named **setup_cppsim3.exe**) located at:

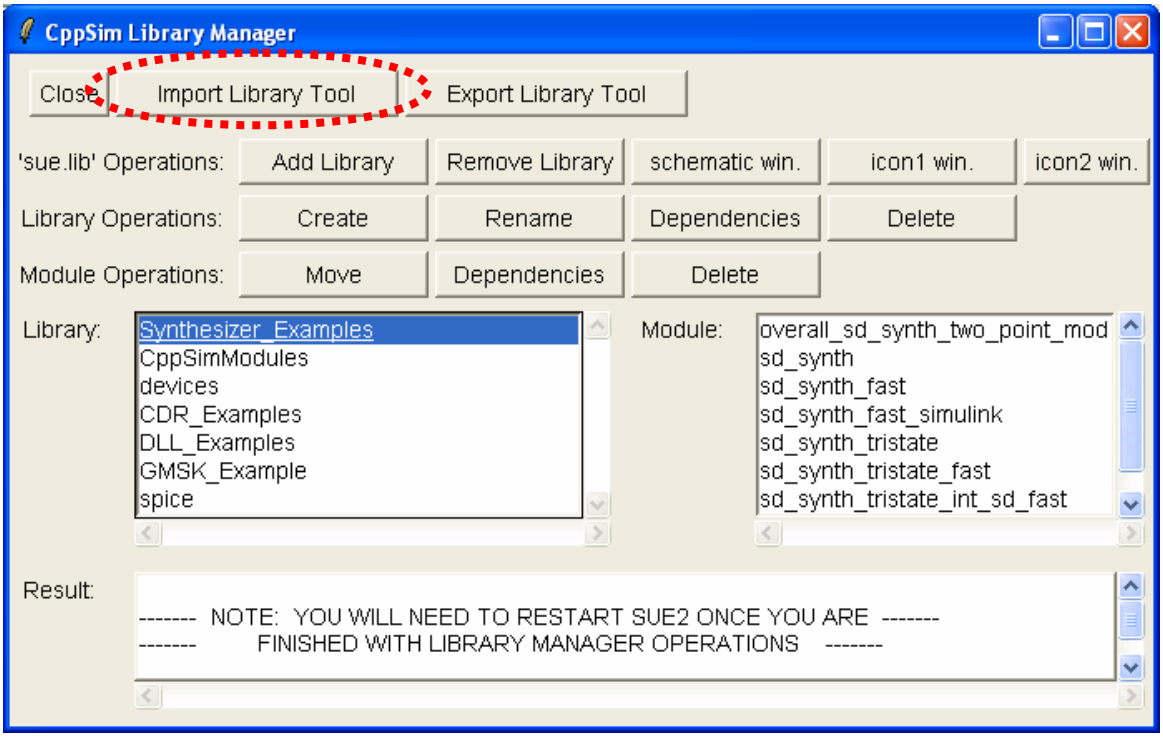
http://www.cppsim.com

Upon completion of the installation, you will see icons on the Windows desktop corresponding to the PLL Design Assistant, CppSimView, and Sue2. Please read the "**CppSim (Version 3) Primer**" document, which is also at the same web address, to become acquainted with CppSim and its various components. You should also read the manual "**PLL Design Using the PLL Design Assistant Program**", which is located at http://www.cppsim.com, to obtain more information about the PLL Design Assistant as it is briefly used in this document.

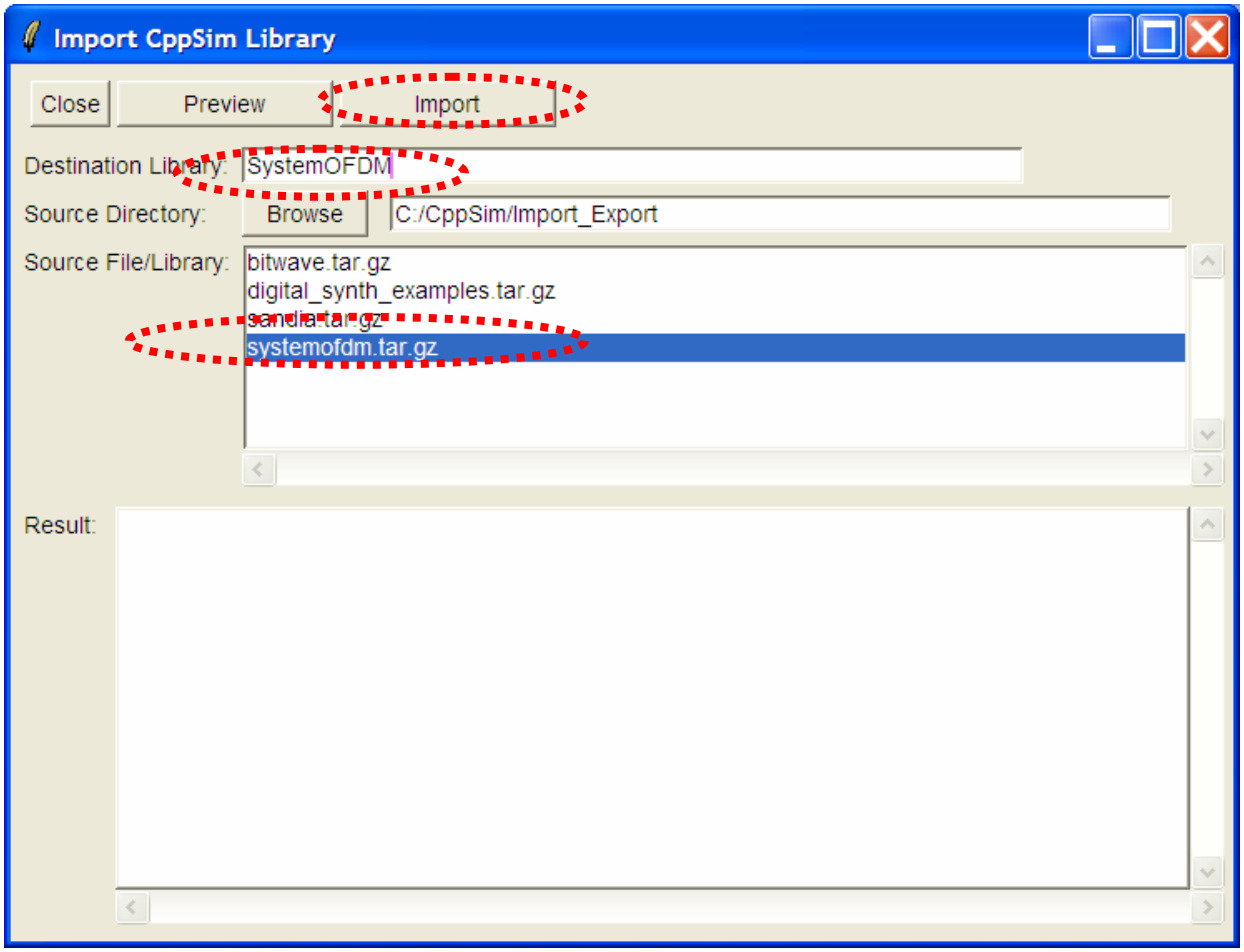
To run this tutorial, you will also need to download the files **sytemofdm.tar.gz** and **ofdm_example.tar.gz** available at http://www.cppsim.com, and place them in the **Import_Export** directory of CppSim (assumed to be **c:/CppSim/Import_Export**). Once you do so, start up **Sue2** by clicking on its icon, and then click on **Tools->Library Manage**r as shown in the figure below.



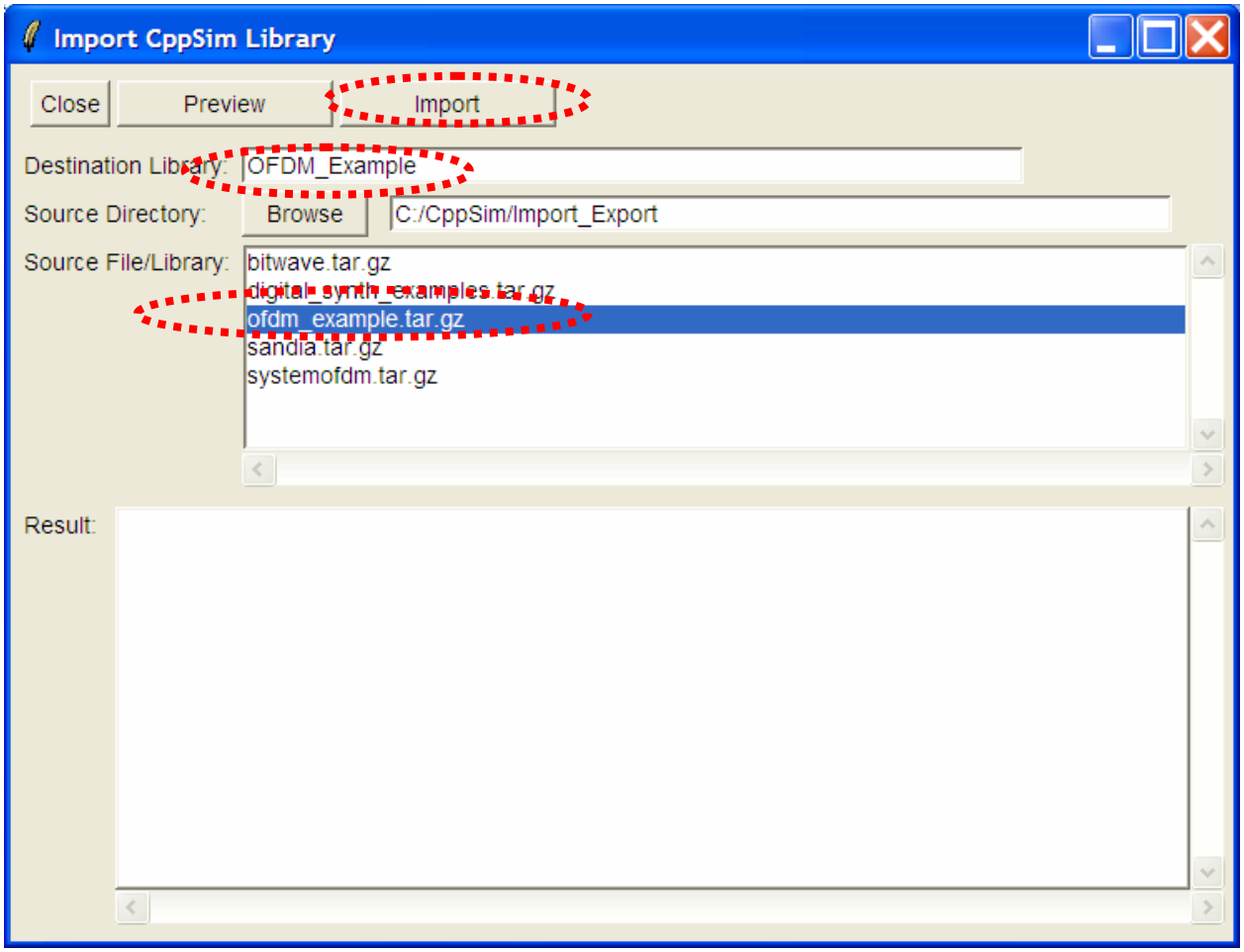
In the **CppSim Library Manage**r window that appears, click on the **Import Library Tool** button as shown in the figure below.



In the **Import CppSim Library** window that appears, change the **Destination Library** to **SystemOFDM**, click on the **Source File/Library** labeled as **systemofdm.tar.gz**, and then press the **Import** button as shown in the figure below. Note that if **systemofdm.tar.gz** does not appear as an option in the **Source File/Library** selection listbox, then you need to place this file (downloaded from http://www.cppsim.com) in the **c:/CppSim/Import_Export** directory.



Once the import operation completes, change the **Destination Library** to **OFDM_Example**, click on the **Source File/Library** labeled as **ofdm_example.tar.gz**, and then press the **Import** button as shown in the figure below. Note that if **ofdm_example.tar.gz** does not appear as an option in the **Source File/Library** selection listbox, then you need to place this file (downloaded from http://www.cppsim.com) in the **c:/CppSim/Import_Export** directory.



Once you have completed the above steps, restart **Sue2** as directed in the above figure. **NOTE:** the order in which you import these two libraries matters since the import tool pays attention to what has already been installed when it brings in a new package. Consequently, the **systemofdm.tar.gz** library must be imported before the **ofdm_example.tar.gz** library.

Introduction

This document will explore the implementation of a system level representation of an OFDM (Orthogonal Frequency Division Multiplexing) transceiver using the CppSim behavioral simulator. Indeed, OFDM is the principle technology that has made possible the communication at high data rates in the multipath frequency-selective fading of the wireless channel. OFDM is used in high-speed wireless applications such as the Wireless Local Area Networks (WLAN) by the IEEE 802.11a WLAN standard (in the U.S.) and HIPERLAN/2 (in Europe), Digital Audio/Video Broadcasting (DAB/DVB), and Wireless Local Loop (WLL). However, the peculiar characteristics of the OFDM signal cause several challenges in system and circuit designs to meet the performance requirement. For instance, an OFDM system has a higher Peak-to-Average power Ratio (PAR) and is more sensitive to the frequency synchronization.

Hence, there are many system and circuit design challenges that need to be resolved. This simulator provides a starting point for such investigation. The simulation is done at baseband to save unnecessary computational time and complexity. One can use the simulation to examine and verify

performance of system algorithms or to determine the hardware specifications of many of the RF components as well as the data converters by adding models of these components into the simulation.

The tutorial on OFDM system is divided into two parts. In this part, Part I, the simulation in the discrete time domain is introduced. This is useful for an analysis that requires ideal performance from other blocks such as the data-converters; hence, the data-converters' low-pass filters at the transmitter and receiver and the sampler can be eliminated. Part II extends this simulation to include the data converters. However, most of the blocks used in part II will be provided in this part for the interested user. In addition, MATLAB 6.5 (Release R13) is used to do some post-processing on the simulated signals.

In this section, the discrete-time OFDM transmitter and receiver architecture are explained first. Next, we will go through a step-by-step simulation of an OFDM system in CppSim and show how CppSim can be utilized in design of an OFDM system.

A. Discrete-Time Transmitter Architecture

Figure 1 indicates the transmitter path of an OFDM signal with N subcarriers and symbol rate per subcarrier of $1/T_{ofdm}$. First, the bits are generated by a **bit source generator**, such as a Bernoulli random generator in this simulation. Next, the **subcarrier modulation selector** chooses a modulation constellation for each subcarrier – to map block of bits into symbols - by specifying the number of bits per symbol for each (i^{th}) subcarrier, b_i, for $i = 0,..., N-1$. For instance, in this simulation b_i can take values of {0, 1, 2, 4, 6, 8} which translate to {no-data, BPSK, 4-, 16-, 64-, and 256-QAM} modulations at the subcarrier modulator. The serial stream of bits is parallelized into N blocks of b_i bits per subcarrier by the Serial-to-Parallel converter (**S/P**), converted into symbols from the corresponding constellations by the **subcarrier modulator**, and fed into the N-point **IFFT**.

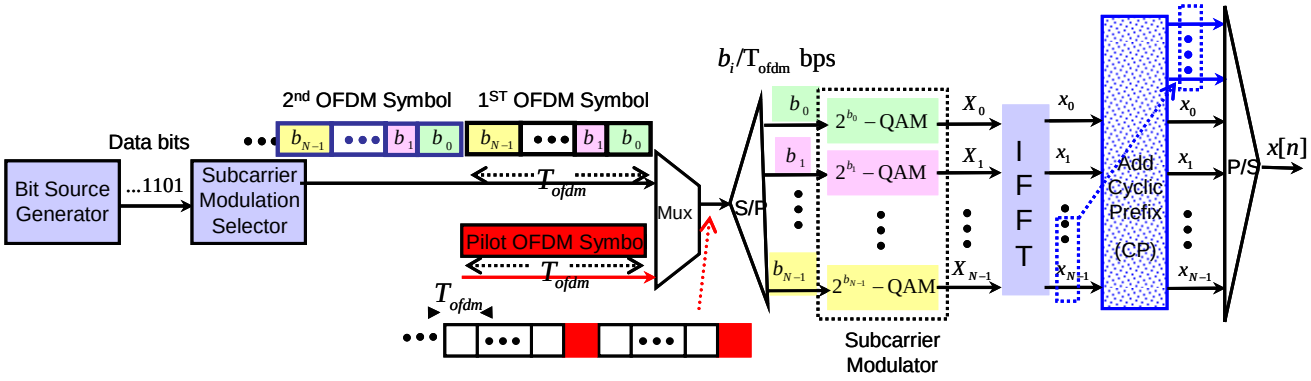


Figure 1: OFDM Transmitter in discrete-time domain

The complex symbols, X[k], at the input of IFFT are the values of an OFDM symbol at equally-spaced discrete frequencies, $f_k = \frac{2\pi k}{N}$, for $k = 0,..., N-1$. In fact, as illustrated in Figure 2, due to the periodicity of the discrete frequency spectrum with 2π, at the baseband, X_0 corresponds to the frequency-domain value at the DC (f = 0), and $X_{N/2-1}$, $X_{N/2}$, and X_{N-1} correspond to the spectrum values at the frequencies $\frac{(N-1)\pi}{N}$, $-\pi$, and $\frac{-2\pi}{N}$ respectively.

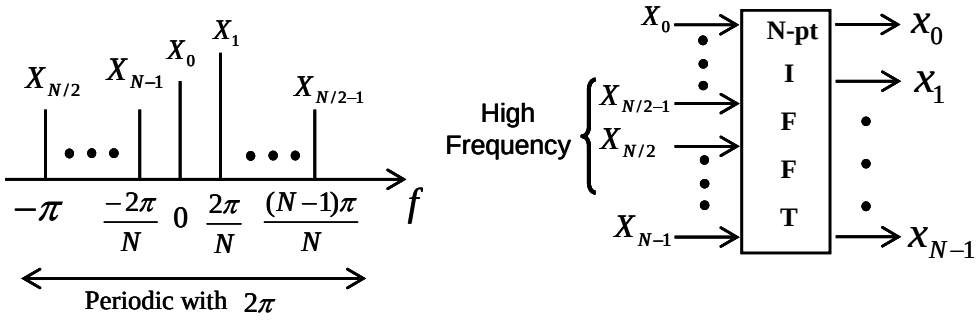


Figure 2: Frequency-domain representation of the OFDM symbol.

The OFDM time-domain samples at the output of IFFT, $\{x_1, x_2, \ldots, x_{N-1}\}$, are subsequently guard-padded by enough samples to eliminate the interference between adjacent OFDM symbols caused by the delay-spread of the multipath channel. Hence, for each OFDM symbol, some of the time-domain samples of the OFDM symbol are cyclically added from its end to the beginning. Figure 3 indicates how the channel's delay spread causes Inter-Symbol Interference (ISI) and how the cyclic prefix (CP) eliminates this ISI. Finally, the parallel time-domain samples are converted to the serial time-domain OFDM samples though a Parallel-to-Serial converter (**P/S**). The resulting discrete-time signal x[n] is transmitted through the channel.

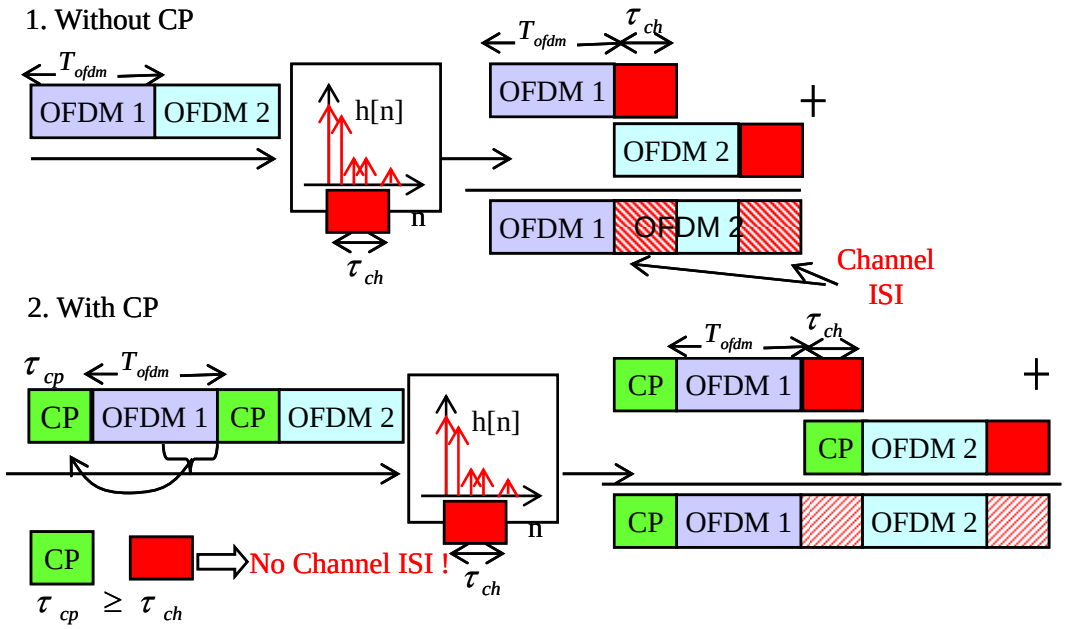


Figure 3: Cyclic prefix (CP) eliminates the channel Inter-Symbol Interference (ISI).

In addition, to do channel estimation at the receiver, known training symbols (called pilot symbols) are sent before the data on all subcarriers to sense the channel response. The pilot OFDM symbols can be inserted between data transmissions at a rate less than the rate of the time-variation of the channel (for instance on the order of a second for an indoor WLAN channel). At the receiver, the channel estimator compares the received pilot symbols at the FFT output with the known transmitted pilot symbols to extract the channel frequency response.

B. Channel Model

The signal x[n] is subsequently transmitted through the wireless channel. For simulation at baseband, an appropriate complex baseband-equivalent model of the channel, h[n], is needed. Next, the thermal noise from the resistor of the receiver antenna, which is modeled as an Additive White Gaussian Noise (AWGN) n[n], is added to produce y[n], the input signal to the receiver.

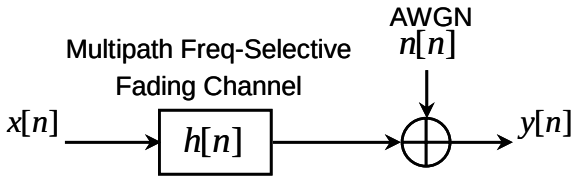


Figure 4: Channel model

C. Discrete-Time Receiver Architecture

At the receiver, the corrupted time-domain OFDM samples y[n] are parallelized, the cyclic-prefix samples are removed, and the OFDM time-domain samples are converted to the corresponding frequency-domain constellation symbols by an N-point **FFT**. The channel estimations for each subcarrier are used by the **one-tap equalizer** to remove the attenuation and phase rotation induced by the channel. By choosing OFDM subcarrier bandwidths less than the coherence bandwidth of the wireless channel, the channel response will be approximately flat over each subcarrier (or bin). In this case, the flat-fading channel frequency response over each bin can be modeled by a complex constant, $a_i e^{j\theta_i}$, which is estimated by the **channel estimator** for each subcarrier i. Finally, the **demodulator** uses the minimum-distance rule to detect the transmitted symbols and the corresponding bits.

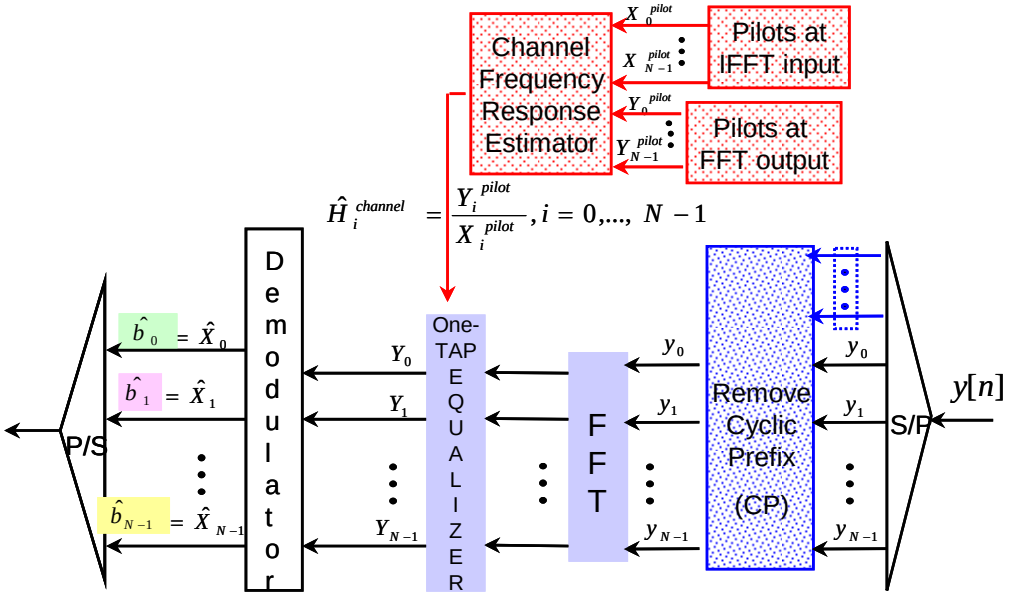


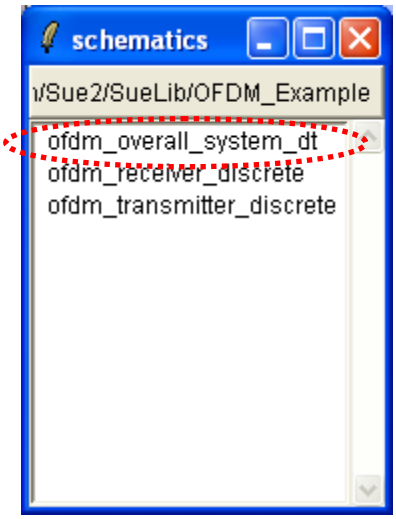
Figure 5: OFDM Receiver in discrete-time domain.

Performing Basic Operations within Sue2 and CppSimView

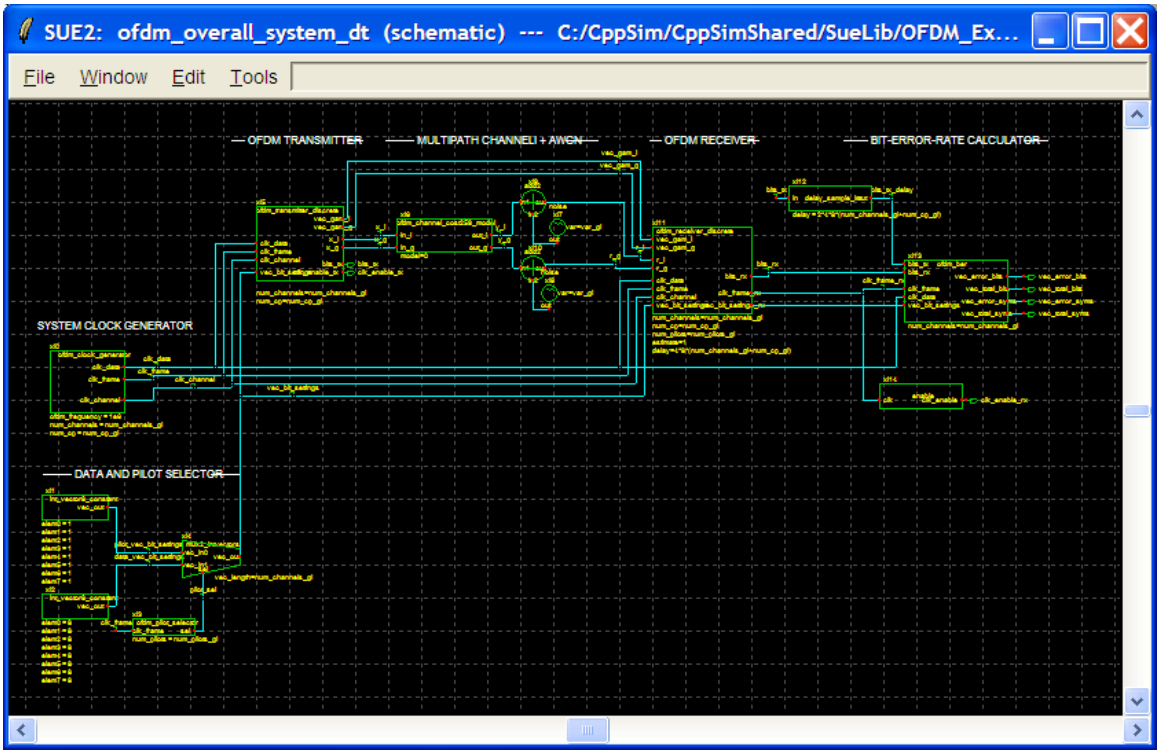
In this section, the user will be guided through basic tasks such as opening the OFDM example system within the Sue2 schematic editor and running CppSim simulations.

A. Opening Sue2 Schematics

- Click on the Sue2 icon to start Sue2, and then select the **OFDM Example** library from the **schematic listbox**. The **schematic listbox** should now look as follows:



Select the **ofdm_overall_system_dt** cell from the above **schematic listbox**. The Sue2 schematic window should now appear as shown below. This cell consists of the system's main blocks: the system clock generator, data and pilot (modulation-type) selector, transmitter, channel, receiver, and Bit-Error-Rate (BER) calculator. Almost all the OFDM modules are located in **SystemOFDM** library in **SueLib**. To insert a module in the current cell, **SystemOFDM** must be open in **icon1**.



- The key system parameters that must be defined by the user are
  - **ofdm_frequency**: this is the spacing between OFDM subcarriers – or bin size.
  - **num_channels**: this is the number of subcarriers of an OFDM symbol.
  - **num_cp**: this is the number of subcarriers used in creating the cyclic prefix.

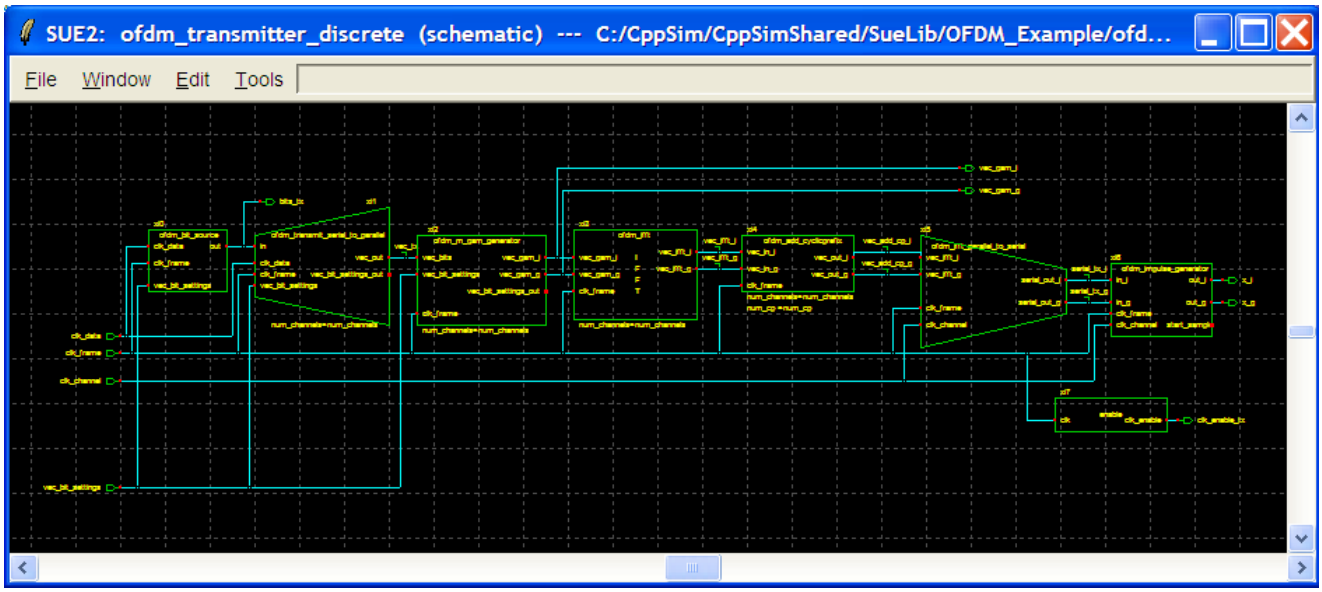
  The above 3 parameters are used to set the clock frequencies in the simulation. Clocking issues are explained in detail in the section on Simulation Design Issues.

  - **num_pilots**: this is the number of pilot OFDM symbols that are sent before the data to estimate the channel.
  - **estimate**: this is a flag to turn the channel estimator on (estimate=1) or off (estimate=0).
  - **pilot_vec_bit_settings** and **data_vec_bit_settings:** these are used to specify the modulation type – through specifying the number of bits/symbol – of each pilot and data subcarrier. The prefix "vec" is used to denote Vector signals. Thus, these are vectors of length **num_channels**.

- In addition, key signals for this schematic include
  - **x_i/x_q** and **y_i/y_q**: correspond to the channel's input and output I/Q signals to generate the Power Spectral Density (PSD) of the channel.
  - **bits_tx** and **bits_rx**: correspond to the transmitted bits and received bits that are used to calculate the probability of bit error or Bit-Error-Rate (BER), which is used as a figure-of-merit to measure the system performance.

- Select the **ofdm_transmitter_discrete** icon within the above schematic, and then press **e** to descend down into its associated schematic. You should now see the schematic shown below. Some key signals in the transmitter schematic include

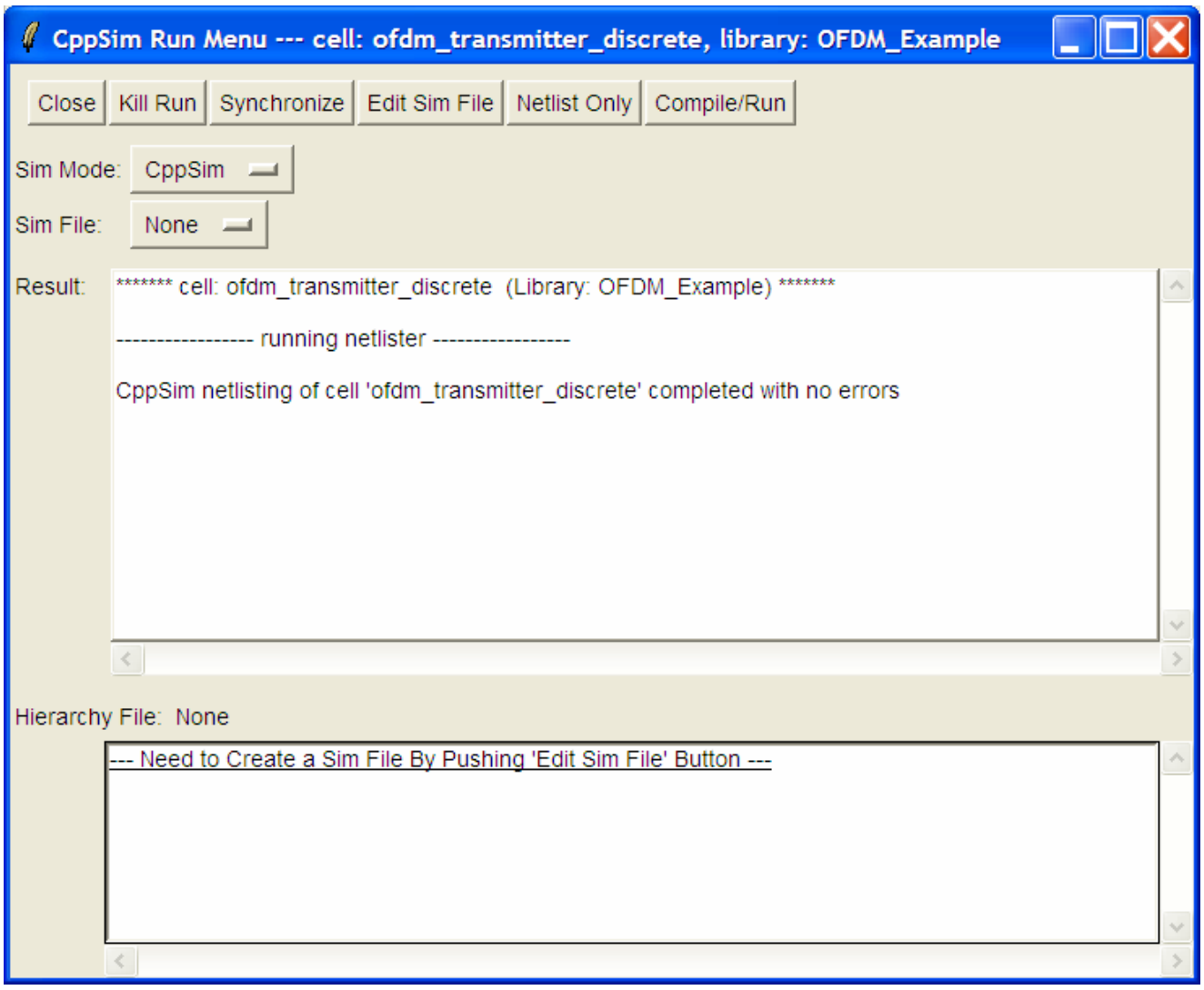
- **vec_qam_i** and **vec_qam_q**: correspond to the N (=num_channels) subcarriers' transmitted symbols chosen from their corresponding constellations. For pilot symbols, these are used by the channel estimator to estimate the channel response per bin. As it will be shown later, these signals are used along with the FFT outputs, **vec_dqam_i** and **vec_dqam_q,** in the receiver to compare their scatterplots (or constellations) to see the amount of channel attenuation and rotation on the transmitted symbols.
- **x_i** and **x_q**: correspond to the transmitted OFDM time-domain samples that can be used to measure the peak-to-average power ratio of the OFDM signal.



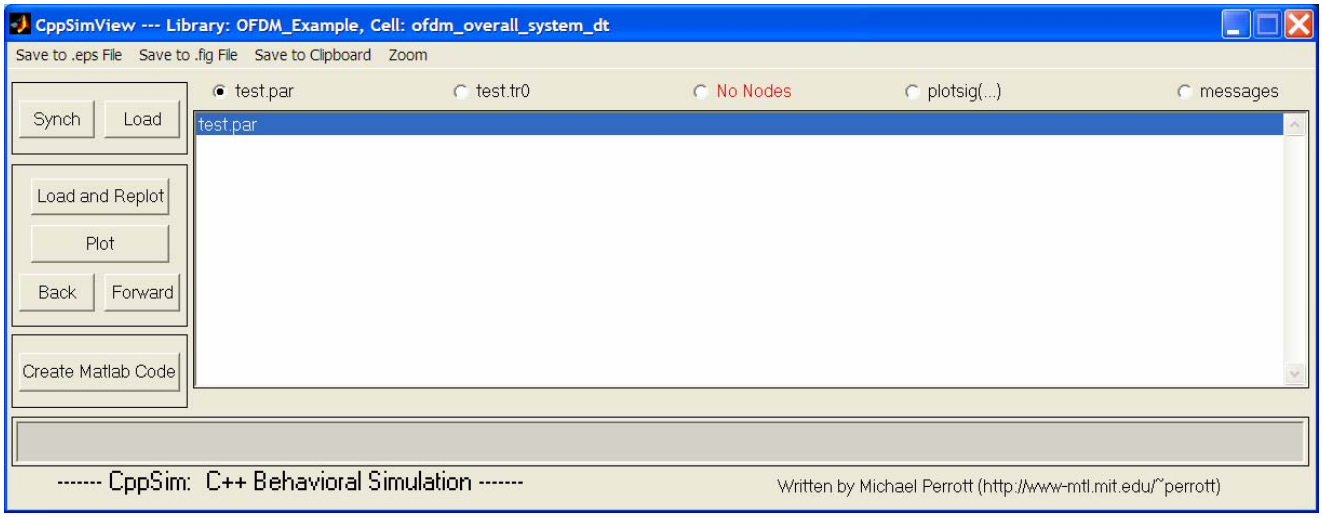
- Press **Ctrl-e** to return to the **ofdm_overall_system_dt** cellview.

B. Running CppSim Simulation

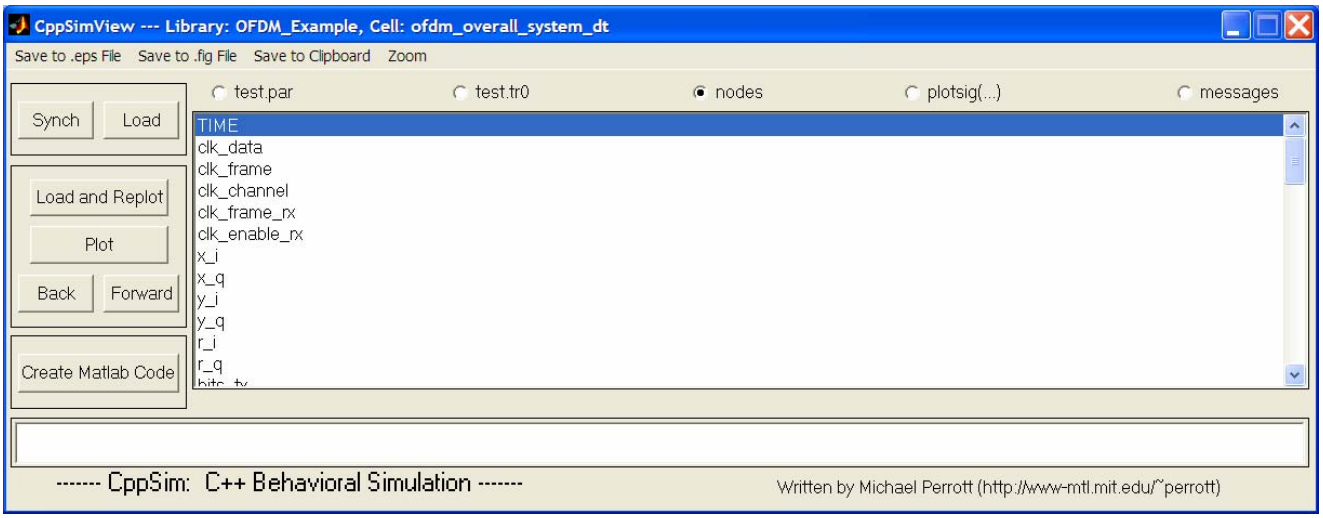
- Within Sue2, click on the **Tools** menubar button, and select **CppSim Simulation**. The CppSim Run Menu window should appear as shown below:



- Click on the **Edit Sim File** button in the CppSim Run Menu – the simulation parameter file (**test.par**) should open in an Emacs window and indicate that **num_sim_steps** is set to 1e5 and **Ts** is set to 1/256e6. You may then close the Emacs window if you like (Press Ctrl-x Ctrl-c).
- Click on the **Compile/Run** button to run the simulation.
- Click on the CppSimView icon to start the CppSim viewer. The viewer should appear as shown below – notice that the banner indicates that it is currently synchronized to the **ofdm_overall_system_dt** cellview.



- If a **No Output File** radio button appears adjacent to the **test.par** radio button, click on it and then select **test.tr0** as the output file. Otherwise, if a **test.tr0** radio button appears adjacent to the **test.par** radio button, click on it.
- Click on the **No Nodes** radio button to load in the simulated signals. CppSimView should now appear as shown below



C. OFDM System Example

In the next few sections, we will show some of the features of CppSim in an OFDM system simulation. The OFDM system has 8 subcarriers, each 1-MHz wide and 256-QAM modulated, and 1 OFDM BPSK-modulated pilot symbol. Finally, using COST259-model A for the multipath channel over the 8-MHz signal bandwidth, we would need 4 subcarriers for the cyclic-prefix to append at the beginning of the OFDM symbol since this is the length of the channel impulse response. For more information regarding COST259 channel models – adopted by BRAN#8 for HIPERLAN/2 simulations – a separate section is designated to Channel Models.

In examining this OFDM system example, we start with no AWGN noise present to illustrate the effects of the multipath channel's fading on the OFDM system. Next, we will add AWGN and examine the combined effects on the system performance.

To start simulation of the OFDM example system, click on **Edit Sim File** and confirm the parameters are set to the values shown below.

- **num_sim_steps**: 1e5
- **Ts** = 1/256e6
- global_param: **var_gl**=0, **num_channels_gl**=8, **num_cp_gl**=4, **num_pilots_gl**=1, **t_ofdm_gl**=1e-6, **ts_gl**=Ts

Note that the AWGN noise variance, **var_gl**, is first set to 0. In addition, in **ofdm_overall_system_dt** cell, under Data and Pilot Selector, confirm that the 8 pilot subcarriers are modulated by 1 bit/symbol and the 8 data subcarriers are modulated by 8 bits/symbol.

Next, click on **Compile/Run** in the CppSim Run Menu.

Plotting Frequency-Domain Results

Given the completion of the above simulation, in this section, we will plot the frequency power spectrum of the channel and the estimated channel responses for each bin. Next, the scatterplot for each bin compares the transmitted symbols with the received symbols before and after the one-tap equalizer.

A. Channel Frequency Response

Now perform the following operations in MATLAB in the following directory ~/CppSim/SimRuns/ofdm_Example/ofdm_overall_system_dt/,

- At the MATLAB prompt, run **channel_impulse_freq**. This file loads in the simulated channel's complex impulse response (the value at each tap), and the average relative power of each tap, and the estimations of the channel response's magnitude and phase of all subcarriers from the channel estimator. Next, it plots the simulation channel's impulse response, the frequency power spectrum and the power-delay-profile (average relative power versus delay spread) (Figure 6). In the channel power spectrum, the stars (*) indicate the perfect channel estimations for the 8 bins by the channel estimator. In addition, this plot shows that the subcarriers at 1 MHz and 2 MHz are attenuated the most by the channel and the subcarrier at -2 MHz is attenuated the least.

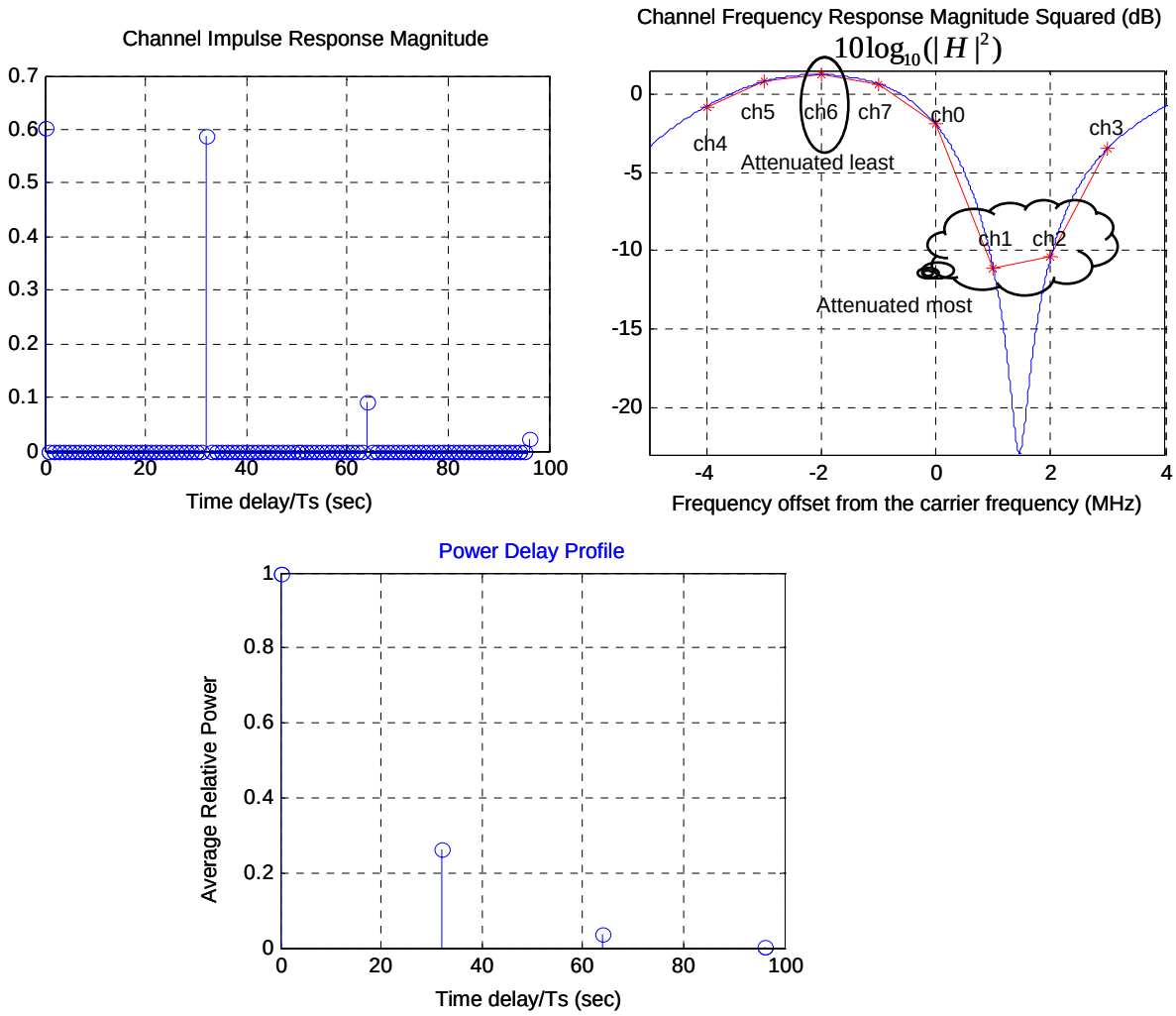


Figure 6: Simulated channel impulse response magnitude (top left), channel frequency response magnitude squared (top right), and power delay profile (bottom) of the 8-MHz wide OFDM system.

B. Scatterplots (Constellations) of Transmitted and Received Symbols

Now perform the following operations in MATLAB:

- At the MATLAB prompt, run **scatter_plots**. Within this file, you can choose to plot constellations per bin of transmitted symbols and received symbols – before and after the one-tap equalizer. Figure 7 shows the scatterplots of subcarrier 1 (at 1 MHz) and subcarrier 6 (at -2 MHz) before and after the equalizer, which experience the most and least amounts of channel fading respectively. Transmitted symbols are indicated by circles (o) and received symbols by stars (*). The more channel fading on subcarrier at 1-MHz is quite apparent in scatterplots before equalization. In addition, the perfect alignment of transmitted and received symbols after equalization shows that channel estimations are perfect. The reason that some of symbols in the 256-QAM constellations below are not transmitted is because the simulation **num_sim_steps** (1e5) in **test.par** was not sufficient. Increasing **num_sim_steps** to 1e6 would result in transmission of all symbols in the 256-QAM constellations.

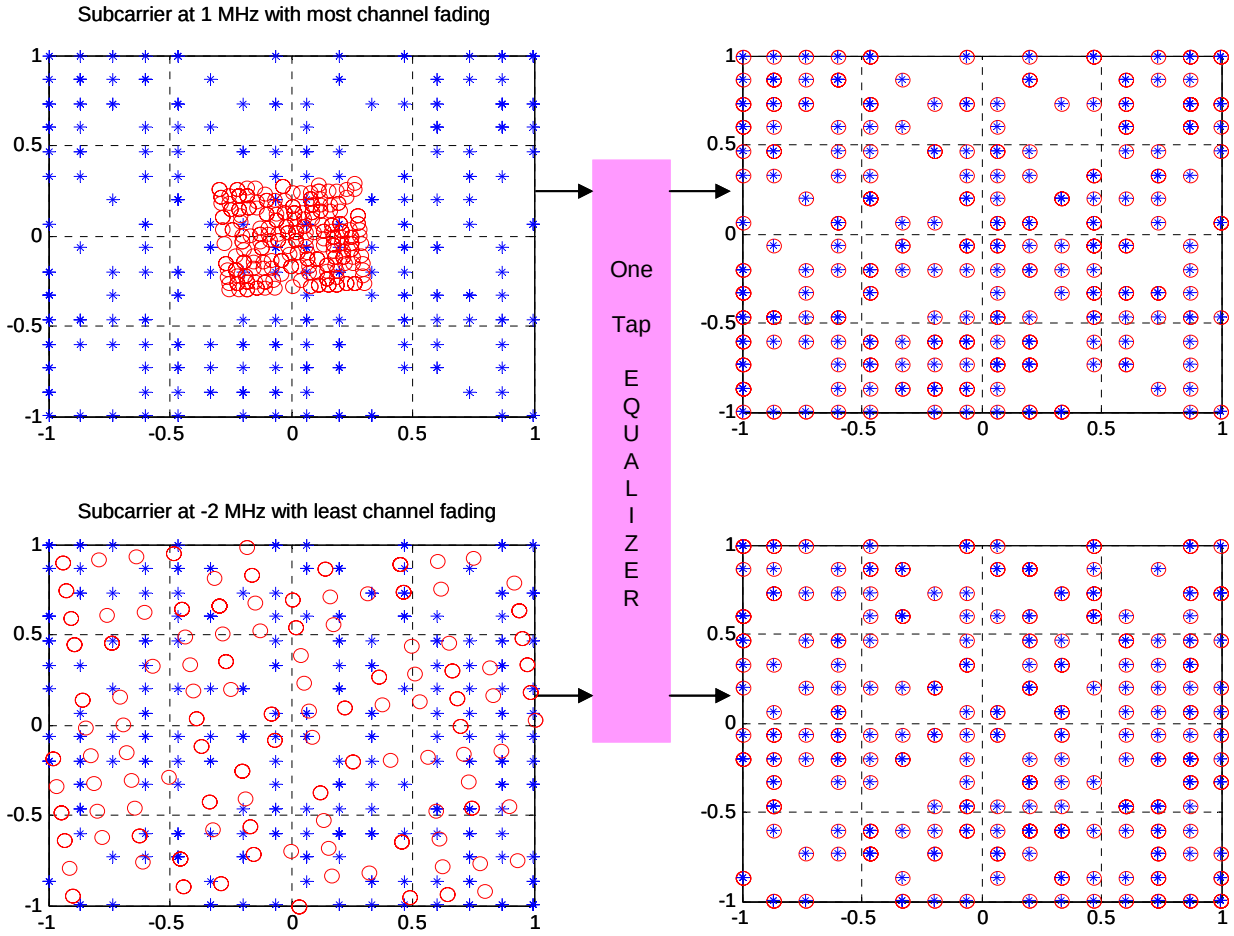


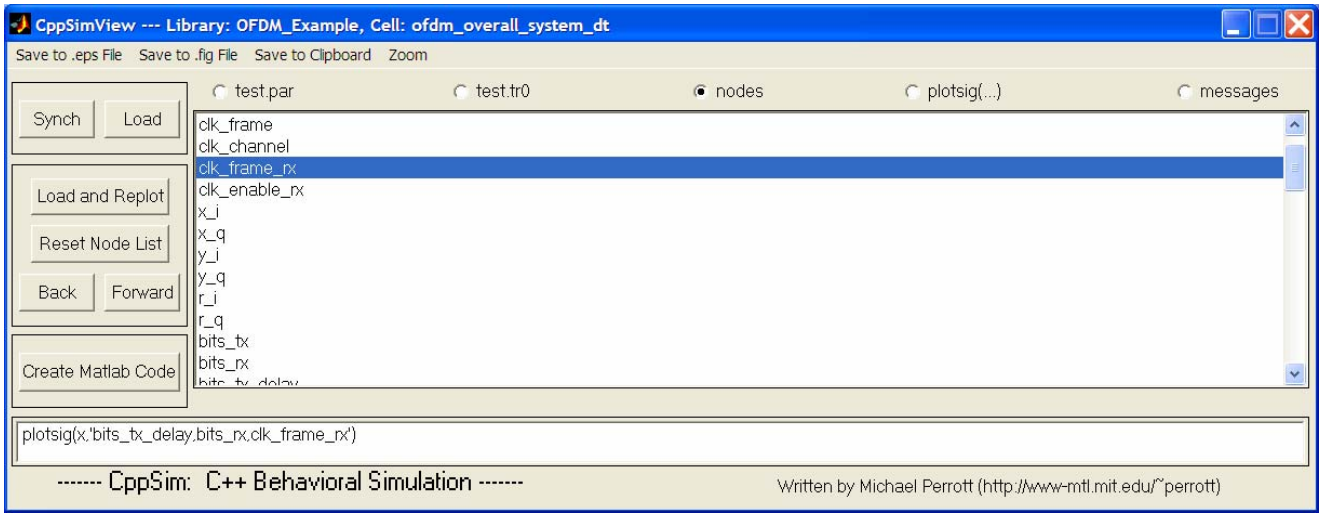
Figure 7: Scatterplots of the worst and best subchannels - at 1 MHz and -2 MHz respectively - before and after equalization in the absence of AWGN, showing effect of multipath fading channel and the perfect equalization.

The scatterplot is the most immediate way to visualize the effects of the multipath channel on attenuating symbol amplitudes and rotating symbol phases. In addition, it shows the performance of channel estimator and equalizer in removing the channel distortions.

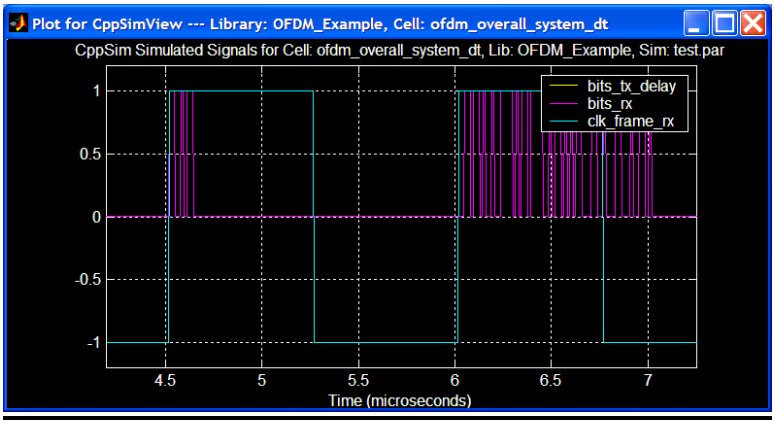
Plotting Time-Domain Results

A. Transmitted and Received Bits

- In the CppSimView window, double-click on signals **bits_tx_delay**, then **bits_rx** and finally **clk_frame_rx**.

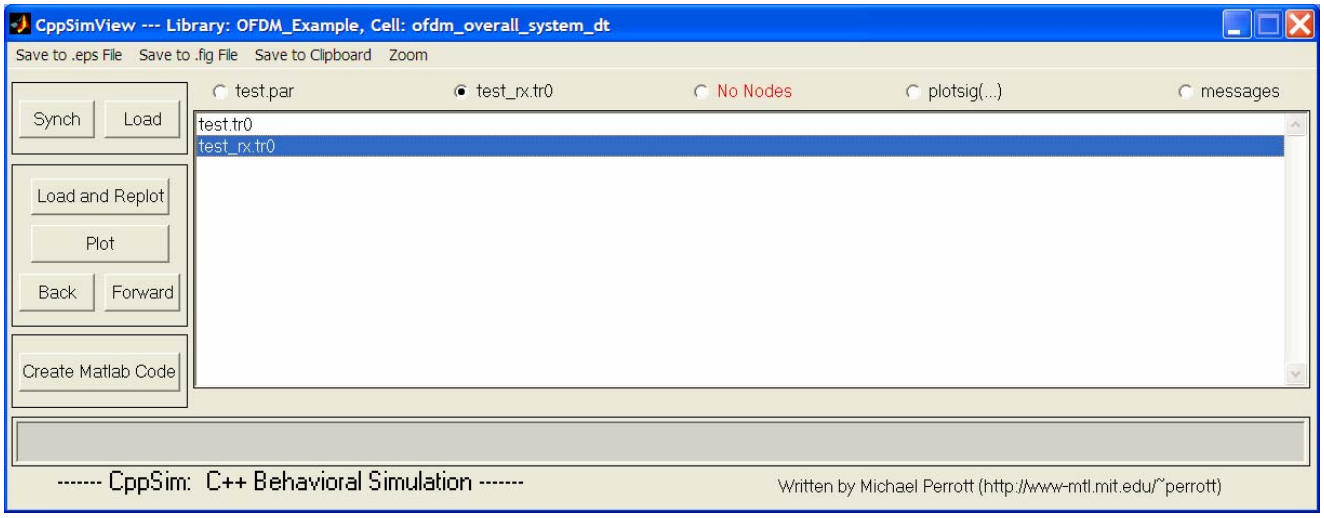


You should see plots of transmitted and received bits coincided since the channel estimations were perfect and the scatterplots of transmitted symbols and received symbols at the input of the detector indicated no transmission error. Each cycle of **clk_frame_rx** indicates the beginning and ending of an OFDM symbol and hence the bits sent in each OFDM symbol. Note that the first symbol has only 8 bits since it is the pilot BPSK-modulated OFDM symbol, while the subsequent OFDM symbols are each 64 bits wide (8 bits in each of the 8 subcarriers). The transmitted bits (**bits_tx**) are delayed by one **clk_frame** cycle – which gives **bits_tx_delay** – to synchronize with the received bits (**bits_rx**).

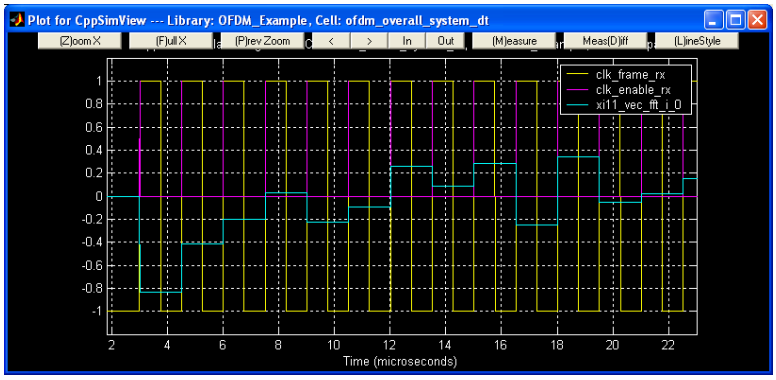


B. Error Bits and Total Bits

Now, in CppSimView window, click on **test.tr0** radio bottom and select **test_rx.tr0** which is used for more memory-efficient saving of signals to save only the signal points of interest, such as the values of signals in each bin in this case.



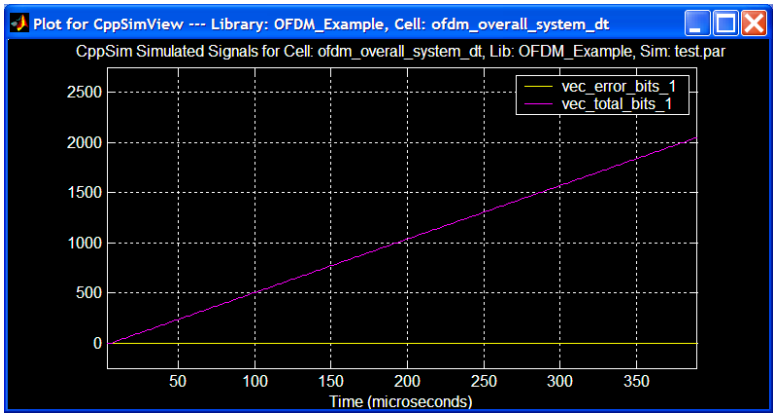
The signals in **test_rx.tr0** are created by using an enable clock, **clk_enable_rx**, that as shown below is 1 at rising edges of **clk_frame_rx** (which is delayed version of **clk_frame**) and 0 otherwise. This eliminates repetition in saving signal values since for each subcarrier the OFDM signal value changes at each rising of **clk_frame**.



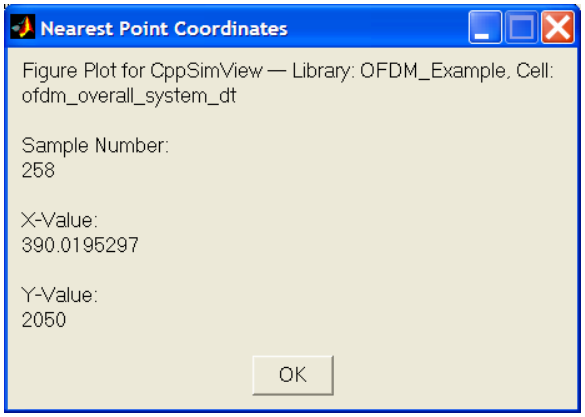
The **clk_enable_rx** is used in **test.par** as below:

```
output: test_rx enable=clk_enable_rx
probe:      vec_error_bits, vec_total_bits
            vec_error_syms, vec_total_syms
      xi11.vec_fft_i, xi11.vec_fft_q
      xi11.vec_dqam_i, xi11.vec_dqam_q
      xi11.vec_pilot_i, xi11.vec_pilot_q
      xi11.vec_z_i, xi11.vec_z_q
```

Next, click on **nodes** bottom and double-click on **vec_error_bits_1** and **vec_total_bits_1** to get the following plots of the number of error bits and total bits in the worst subchannel 1 as simulation time passes. This confirms one more time that there is no bit error in this worst subchannel (at 1-MHz).



To measure the total bits, press letter **l** to see the simulation points, next (Z)oomX on the last simulation points, click on (M)easure bottom, point the mouse on the last simulation point on **vec_total_bits_1** and left click on it, and finally right click to see the value. You should see the following plot.



This indicates that 2050 total bits (including the pilot bit) or 2049 total data bits were sent over this subchannel. Running **find_ber** at MATLAB prompt uses **vec_error_bits** and **vec_total_bits** to find the BER for all subchannels.

Examining Effect of AWGN

Let us now examine how the AWGN combined with the above multipath channel affects the performance BER. We do this in two steps:

- Step 1: Run the simulation without AWGN to estimate the channel response by having the following parameter values (and all other parameters set as before).
  - **test.par**: **num_sim_steps** = 1e5, **var_gl** = 0 (AWGN noise variance)
  - **ofdm_overall_system_dt** cellview, **ofdm_receiver_discrete** cell: **estimate** = 1
- Step 2: After completion of above simulation, run the simulation including the AWGN. In this run, more simulation points are needed for a correct comparison of the simulated BER to the target BER of 10^{-3}. In addition, the channel estimator should be turned off now to use the estimated channel values obtained in Step 1. Hence, we modify the following parameters:
  - **test.par**: **num_sim_steps** = 1e6, **var_gl** = 1e-5*Ts
  - **ofdm_overall_system_dt** cellview, **ofdm_receiver_discrete** cell: **estimate** = 0

Next, at MATLAB prompt, run **find_ber** and **find_snr** to get the BERs and SNRs in Table 1. The SNR per subchannel *i* is the normalized SNR by signal power, defined as $SNR_i = \left|H_i^2\right| / E[N_i^2]$, where $\left|H_i^2\right|$ and $E\left[N_i^2\right]$ are the average energies due to channel and noise in the ith subchannel. For detailed information on how SNR is estimated, reader is referred to the BER and SNR Measurements section in Simulation Design Issues.

Subchannel #	0	1	2	3	4	5	6	7
error bits/total bits	0/20801	156/20801	121/20801	0/20801	0/20801	0/20801	0/20801	0/20801
BER	0	0.0075	0.0058	0	0	0	0	0
SNR (dB)	36.0190	26.9921	27.5686	34.6151	36.9843	38.5737	39.0621	38.7633

Table 1: Simulated BER and SNR of each subchannel

Comparing this result to the channel frequency response examined earlier, this confirms our speculation of subchannels 1 and 2 (at 1- and 2-MHz) having the most error and lowest SNR. Let us look at the scatterplots of subchannel 1 (with high BER) and subchannel 6 (with low BER) at the input of demodulator after one-tap equalizer to see the effect of adding an AWGN (Figure 8).

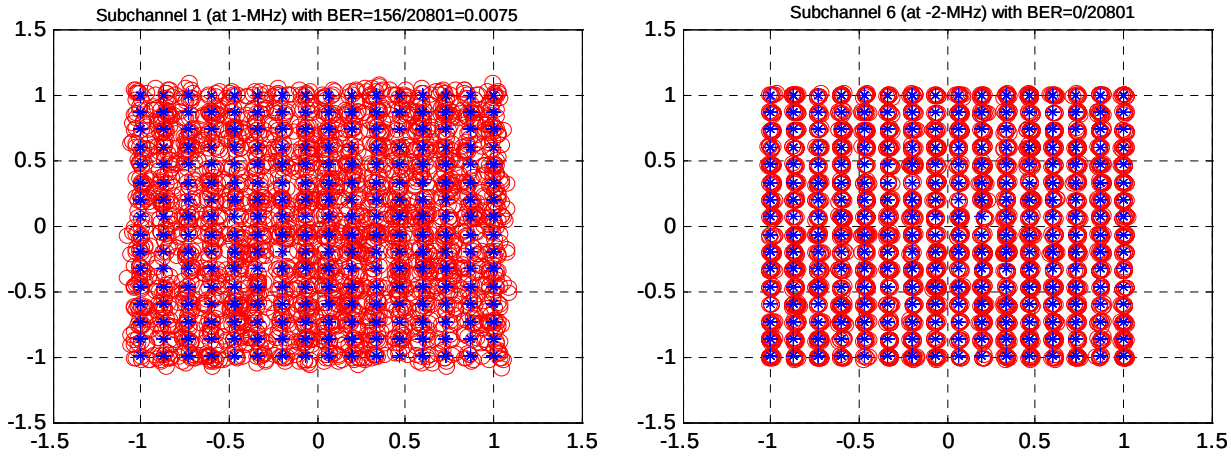


Figure 8: Scatterplots of worst and best subchannels (left and right respectively) after equalizer showing effect of the AWGN.

The scatterplots indicate that the high channel attenuation in subchannels 1 and 2 for the given amount AWGN limits the allowable modulation type for these subchannels below 256-QAM level to meet the target BER of 10^{-3}. Hence, we reduce the modulation level on subcarriers 1 and 2 to 64-QAM and examine the performance.

To do this, Step 2 is repeated with the following bit-assignment of the **data_vec_bit_settings** = {8, 6, 6, 8, 8, 8, 8, 8}. Table 2 shows that the BER requirement will be met for all subchannels.

Subchannel #	0	1	2	3	4	5	6	7
error bits/total bits	0/20801	0/15601	0/15601	0/20801	0/20801	0/20801	0/20801	0/20801
BER	0	0	0	0	0	0	0	0
SNR (dB)	36.0225	26.9784	27.4171	34.4262	37.1746	38.8237	39.2799	38.7331

Table 2: Simulated BER and SNR per subchannel after reducing modulation level of the worst subchannels to 64-QAM.

In this case, subchannel 1 will have the following clean scatterplot (64-QAM) at the input of demodulator (Figure 9).

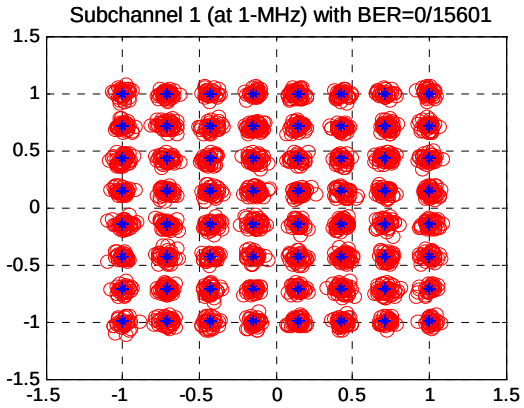


Figure 9: Scatterplot of the worst subchannel (at 1 MHz) after equalizer with a lower modulation level.

Channel Models

In this section, we will introduce the channel model used in this simulation and specifically go through the process of choosing appropriate channel parameters from the given model based for a wideband 128-MHz OFDM system.

A. COST259-Model A

We have used the channel model of a typical office environment for Non Line-of-Sight (NLOS) conditions with root-mean-square (rms) delay spread of $\tau_{rms} = 50ns$. This model (called model A) is proposed by the European Cooperation in the field of Scientific and Technical research (Euro-COST 259), and is adapted by BRAN#8 for HIPERLAN/2 simulations at the 5 GHz band [1].

Modeling ([1])
The model is a tapped-delay-line type model that is described by a few measurable parameters and maintains a sufficient level of complexity to provide a realistic modeling of the relevant channel characteristics. The channel impulse response, $h(\tau,t)$, is modeled as,

$$h(\tau,t) = \sum_{k=1}^{N} a_k(t)\delta[\tau-(k-1)\Delta\tau]$$

where *t* is time, τ is delay, a_k are complex amplitudes, and $\Delta\tau$ is the tap spacing with respect to time. By Nyquist's sampling theorem, for a given bandwidth, *W*, the channel is unambiguously determined if $\Delta\tau < 1/W$. Hence, for instance for W=128 MHz, $\Delta\tau$ should be less than 7.8ns. The total number of taps, *N*, is determined from the maximum excess delay and $\Delta\tau$. The fading of each tap is assumed to follow a Rayleigh probability distribution and the different taps are uncorrelated. In addition, the average power per tap $\overline{a}_k$ is assumed to decline exponentially with time according to

$$\bar{a}_k = A\exp\left[-\frac{(k-1)\Delta\tau}{2\Gamma}\right]$$

where Γ is the expected rms delay spread and A is a normalization constant. The time-variation of impulse response, due to $a_k(t)$, is characterized by the corresponding Doppler spectrum. Assuming that angles of arrivals of received signals have a uniform distribution, the Doppler spectrum for a moving receiver with an omni-directional antenna is given by

$$D(f) = \frac{1/(2\pi f_{\max})}{\sqrt{1-(\frac{f}{f_{\max}})^2}}$$

where *f* is the Doppler frequency, and $f_{\max} = \nu/\lambda$ where ν is the receiver velocity and λ is the carrier wavelength ($\lambda = \frac{3\times10^8 m/s}{f_c}$). To generate the channel impulse-response taps, complex Gaussian random process generators are used – which give Rayleigh probability distribution of tap amplitudes – and the time-variation is given by Doppler filtering. This means that a sequence of impulse responses, equidistant in time, is generated. The distribution is transformed to the Doppler frequency domain, by a DFT of the time sequence of each tap, and then filtered by the corresponding Doppler spectrum. Finally, the distribution is transformed back to time domain via an IDFT.

Figure 10 indicates the power delay profile (PDP) of model A (the average relative power versus time delay). The tap spacing is non-uniform to reduce the number of taps needed. From the PDP, $P_h(\tau)$, the rms delay spread, τ_{rms}, and the mean excess delay, τ_{mean} are obtained via,

$$\tau_{rms} = \sqrt{\frac{\int_0^{\tau_{\max}} (\tau-\tau_{mean})^2 P_h(\tau)d\tau}{\int_0^{\tau_{\max}} P_h(\tau)d\tau}}, \qquad \tau_{mean} = \sqrt{\frac{\int_0^{\tau_{\max}} \tau P_h(\tau)d\tau}{\int_0^{\tau_{nax}} P_h(\tau)d\tau}}$$

Thus, the model characterizes a channel of coherence bandwidth $B_c \approx 1/(5\tau_{rms})$ and $B_c \approx 1/50\tau_{rms}$ for frequency correlation of 0.5 and 0.9 respectively [2].

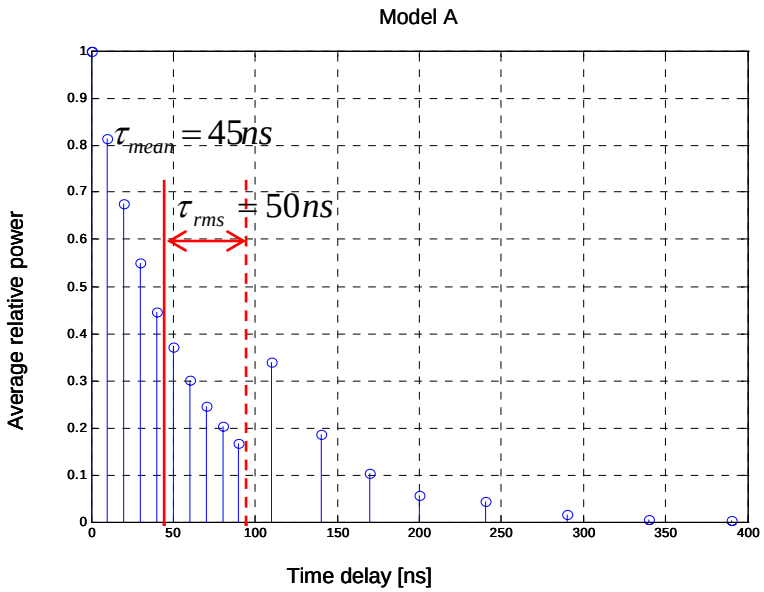


Figure 10: Power Delay Profile of COST259-Model A

Table 3 shows the parameters of Model A that are used to produce the PDP in Figure 10.

Tap Number	Delay (ns)	Average Relative Power (dB)	Ricean K	Doppler Spectrum
1	0	0.0	0	Classical
2	10	-0.9	0	Classical
3	20	-1.7	0	Classical
4	30	-2.6	0	Classical
5	40	-3.5	0	Classical
6	50	-4.3	0	Classical
7	60	-5.2	0	Classical
8	70	-6.1	0	Classical
9	80	-6.9	0	Classical
10	90	-7.8	0	Classical
11	110	-4.7	0	Classical
12	140	-7.3	0	Classical
13	170	-9.9	0	Classical
14	200	-12.5	0	Classical
15	240	-13.7	0	Classical
16	290	-18.0	0	Classical
17	340	-22.4	0	Classical
18	390	-26.7	0	Classical

Table 3: COST259-Model A corresponding to typical office environment for NLOS conditions and 50 ns average rms delay spread.

B. COST259-Models B and C

In addition to Model A, COST259 provides other models B and C that correspond to typical open space and office environments for NLOS conditions with $\tau_{rms} = 100ns$ and typical NLOS large open space with $\tau_{rms} = 150ns$ respectively. These models are based on the tapped-delay-line model described in the previous section. Figure 11 provides power delay profiles of these models, the values of which can be found in MATLAB file **ch_models_cost259.m**. In addition, the taps in all these models have Rayleigh fading statistics (Ricean Factor K=0) and have a classical (Jake's) Doppler Spectrum corresponding to a terminal speed of 3m/s.

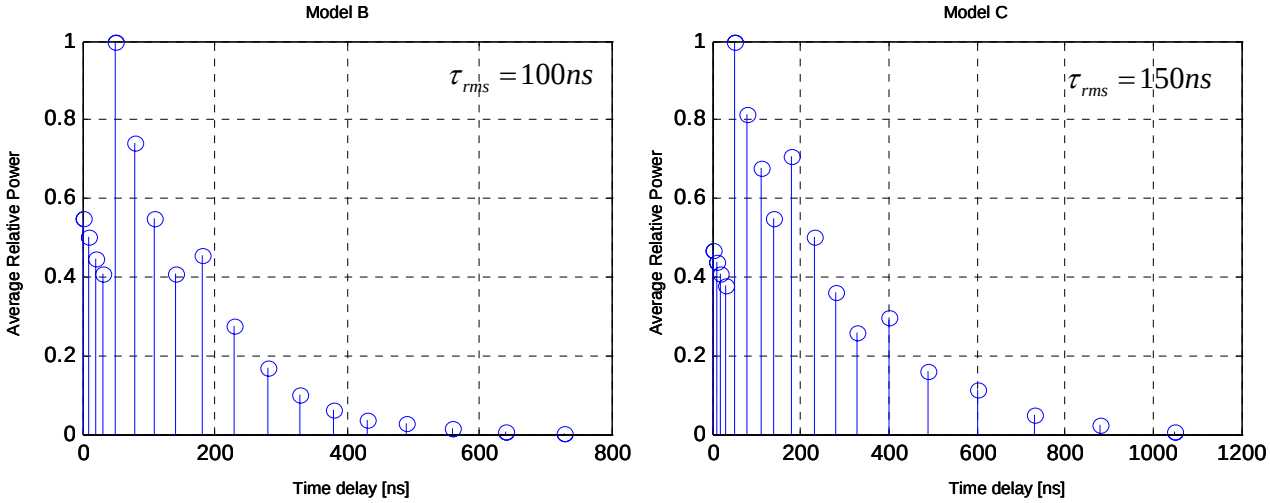


Figure 11: Power Delay Profile of COST259-Model B and C.

The **ofdm_channel_cost259_model** module in SystemOFDM library can choose among the three models A, B or C of COST259 by setting the parameter **model** = 0, 1, or 2 respectively. Note that at this point this module assumes a static channel for the duration of simulation, though time-variation of the channel can be incorporated. In addition, the MATLAB function **delay_spread** is provided which calculates the rms delay-spread and mean excess delay given the variances and delays of each tap in the power delay profile of a give channel impulse response.

C. Channel Model for a Wideband OFDM System from COST259-Model A

Using COST259-Model A with a minimum tap spacing of $\Delta\tau = 10ns$, the channel frequency response is unambiguously determined up to 100 MHz around 5 GHz. On the other hand, for an OFDM system with bandwidth of greater than 100 MHz, a smaller tap spacing would be needed. To show how to derive the channel model for a wideband OFDM system, we examine an OFDM system with 128-MHz bandwidth. To obtain channel response over 128 MHz bandwidth, a PDP with $\Delta\tau < 7.8ns$ is needed. However, sampling the model's PDP at a smaller tap spacing, $\Delta\tau'$, than the model's $\Delta\tau = 10ns$ is legitimate only if the underlying channel measurement contains frequency response measurement over the bandwidth of $W < \frac{1}{\Delta\tau'}$.

Fortunately, this is possible as one of the underlying channel measurements of this model measures the amplitude and phase of the channel frequency response over a 160 MHz band around the 5.06-GHz at 201 equidistant frequencies using a network analyzer. The power delay profile is subsequently obtained by applying a Hanning window and an IFFT to the measured frequency domain data.

Hence, to model the channel frequency response accurately over the 128-MHz band to use in the simulation, the model's PDP is sampled at every 7.8ns. However, using all the uniformly-spaced taps results in $\tau_{rms} = 70ns$ and $\tau_{mean} = 72ns$. Hence, a non-uniform PDP close to that of the model is chosen, as shown in Figure 12 to obtain $\tau_{rms} = 49ns$ and $\tau_{mean} = 46ns$ which are close to those of the model.

Figure 12: PDP (indicated by stars) of the channel to be used in the 128-MHz wide OFDM system simulation obtained from the PDP of COST259-Model A.

Using this power delay profile, a realization of the channel's complex impulse response is obtained by using 19 uncorrelated complex Gaussian random generators (one for each tap), each with 0-mean and variance equal to the corresponding average relative power in the PDP. The result of simulating this channel is shown in Figure 13, which illustrates the channel frequency response over the 128-MHz bandwidth and the perfect channel estimations over each 1-MHz bin performed by the channel estimator (as indicated by starts).

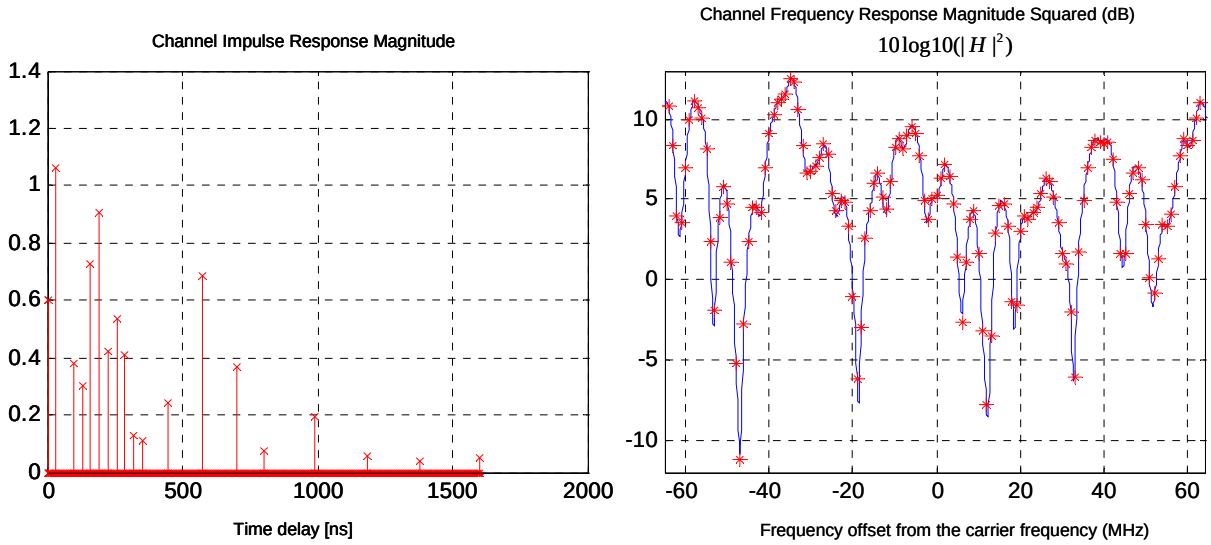


Figure 13: Simulated channel impulse response magnitude (left) and channel frequency response magnitude squared (right) with the 128 channel response estimations (indicated by stars).

These plots are obtained by simulating **ofdm_overall_system_dt** with the following parameters:

- In **test.par**:
  - **Ts** = 1/4096e6
  - global_param:
    - **var_gl**=1e-5*Ts
    - **num_channels_gl** = 128, **num_cp_gl** = 51, **num_pilots_gl** = 1
      t_ofdm_gl = 1e-6, **ts_gl** = Ts

- In **ofdm_overall_system_dt** cellview:
  - In Data and Pilot Selection: use the *int_vector_pilotconstant* and *int_vector_constant* blocks from the SystemOFDM library, and name them xi1 and xi2, respectively (to preserve simulation order). Also set their parameter **length = num_channels_gl**. These blocks set the values of **pilot_vec_bit_settings** and **data_vec_bit_settings**. Use **bits_load.m** script in MATLAB to load desired number of bits/subcarrier to be used by these blocks by typing **bits_load(128)** at the MATLAB prompt.
  - In **mux2_intvectors** module, set the parameter **length = num_channels_gl**.
  - In **ofdm_receiver_discrete** module, replace the *delay_vector8* and *delay_intvector8* blocks with *delay_vector128* and *delay_intvector128*. Make sure that the names of the new modules match those that are being replaced to preserve the simulation order.

D. Naftali Channel Model

In addition to COST259 channel models, another popular channel model is one that was originally proposed by Naftali Chayat in IEEE P802.11-97/96. This model is implemented by the module **ofdm_channel_naftali_filter**.

This section introduces the assumptions and parameters of this model as it is described in [3]. In this model, the channel average power (or power delay profile) is described by an exponentially decaying Rayleigh fading channel model. In addition, the channel is assumed static throughout the packet and generated independently for each packet.

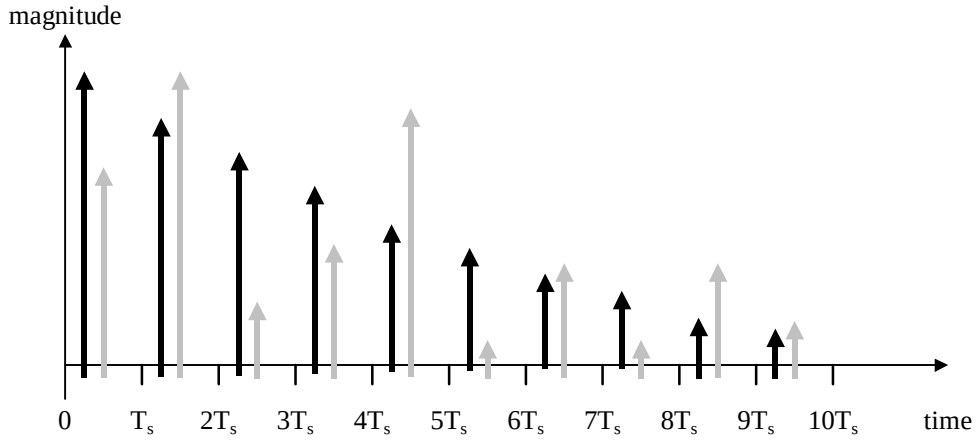


Figure 14: Channel impulse response magnitude; black illustrates average magnitudes, gray illustrates magnitudes of a specific random realization of the channel; the time positions of black and gray samples are staggered for clarity only (adopted from [3]).

The impulse response of the channel, h_i, is composed of complex samples with random uniformly distributed phase and Rayleigh distributed magnitude with average power decaying exponentially as shown in Figure 14.

$$h_i = N(0, \tfrac{1}{2}\sigma_k^2) + jN(0, \tfrac{1}{2}\sigma_k^2)$$
$$\sigma_k^2 = \sigma_0^2 e^{-kT_s/\tau_{RMS}}$$
$$\sigma_0^2 = 1 - e^{-T_s/\tau_{RMS}}$$

where $N(0, \tfrac{1}{2}\sigma_k^2)$ is a zero mean Gaussian random variable with variance $\tfrac{1}{2}\sigma_k^2$, and $\sigma_0^2 = 1 - e^{-kT_s/\tau_{RMS}}$ is chosen so that the condition $\sum \sigma_k^2 = 1$ is satisfied to ensure same average received power. It is assumed that the sampling time T_s in the simulation is shorter than a symbol time (or chip time) by at least a factor of four (typically in simulations it is a sub-multiple of the symbol duration). The number of samples to be taken in the impulse response should ensure sufficient decay of the impulse response tail, e.g. $k_{\max} = 10\tau_{rms}/T_s$.

Simulation Design Issues:

A. Clocking

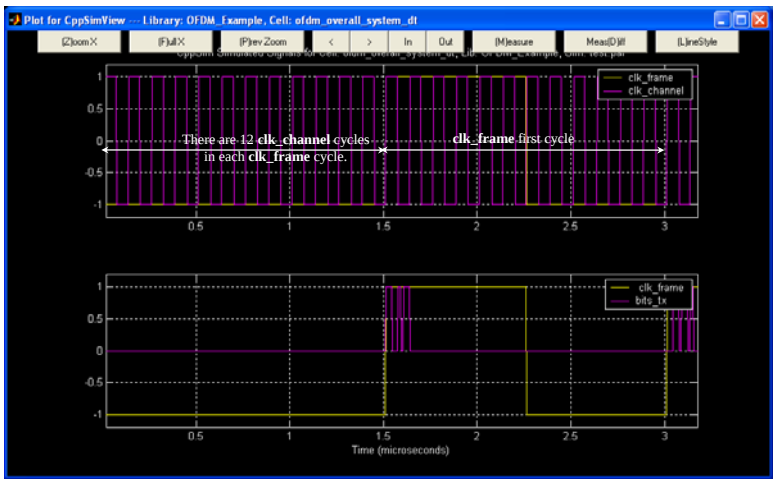
Appropriate clocking in OFDM system simulation is quite important for different blocks to be able to synchronize to OFDM symbols as well as to the subchannels within each OFDM symbol and the bits per symbol. A look at **ofdm_transmitter_discrete** and **ofdm_receiver_discrete** modules indicates

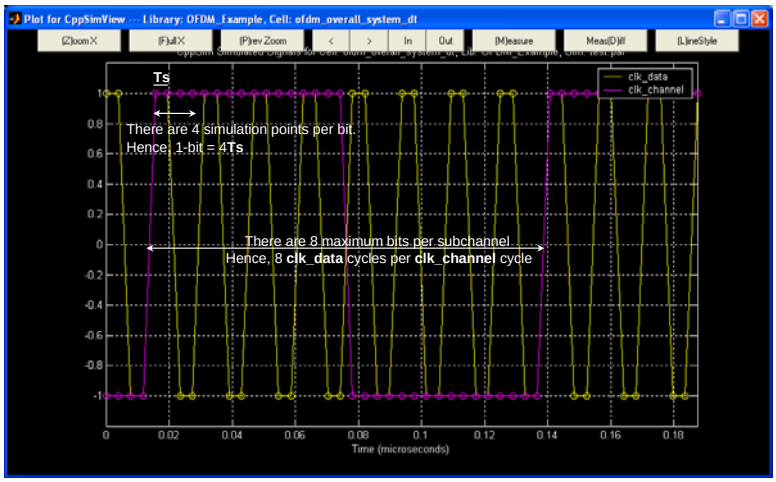
that almost all the blocks in the transmitter and receiver are clocked. There are 3 clocks that are needed, as follows:

- **clk_data**: this clock is used for data bits, where one bit is generated per cycle. It is used by *ofdm_bit_source* and *ofdm_transmit_serial_to_parallel* (in transmitter), *ofdm_receiver_parallel_to_serial* (in receiver), and *ofdm_ber*.
- **clk_channel**: this clock is used for subchannels in each OFDM symbol. It is used by *ofdm_ifft_parallel_to_serial* and *ofdm_impulse_generator* (in transmitter), and *ofdm_fft_serial_to_parallel_cp* (in receiver).
- **clk_frame**: this clock is used for OFDM symbols. It is used by almost all blocks. Its first rising edge cycle signals the beginning of data transmission. Each cycle is for duration of one OFDM symbol data + cyclic prefix, where each OFDM symbol data is 1/**ofdm_frequency** long (**ofdm_frequency** = bin size).

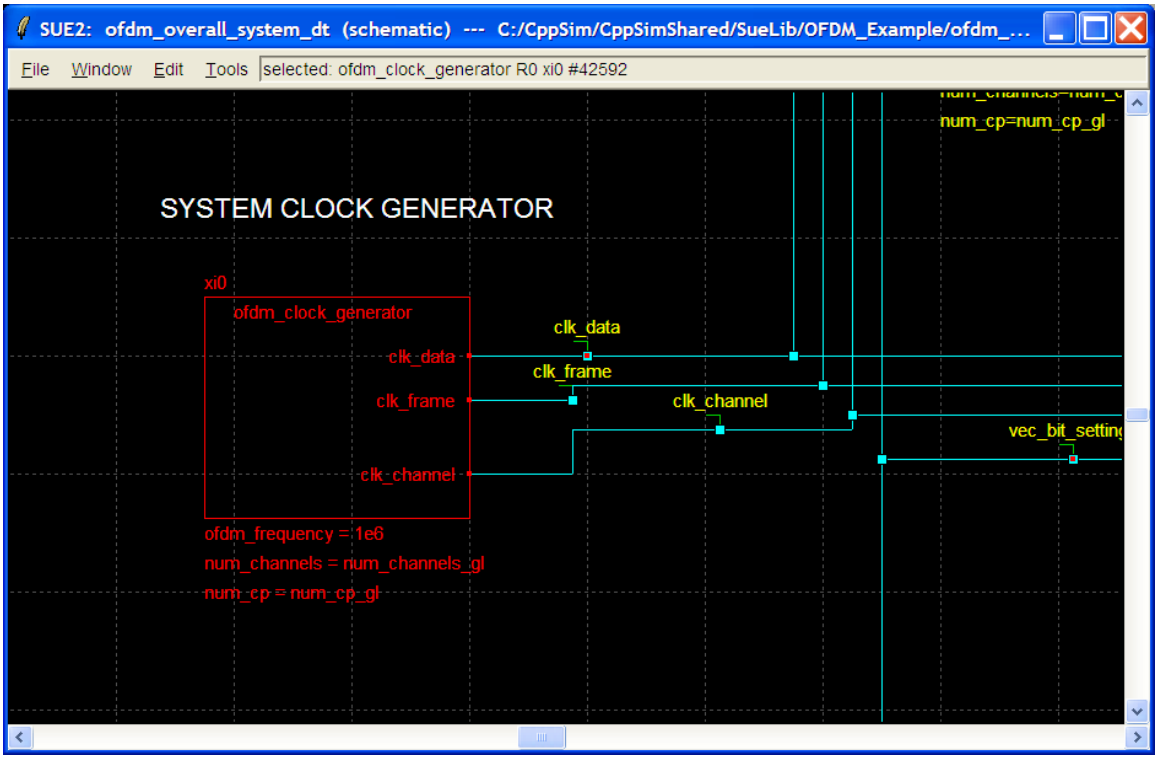
Finally, the simulation clock with period **Ts** (time steps between simulation points) must be related to these clocks properly. To detect the clock edges, the fastest of which is **clk_data**, one needs at least 4 simulation samples in each **clk_data** cycle. Hence, 1/**Ts** must be at least 4 times **clk_data** rate.

In figures below, plots of these clocks indicate the relationship between them, when **num_channels**=8 and **num_cp**=4.

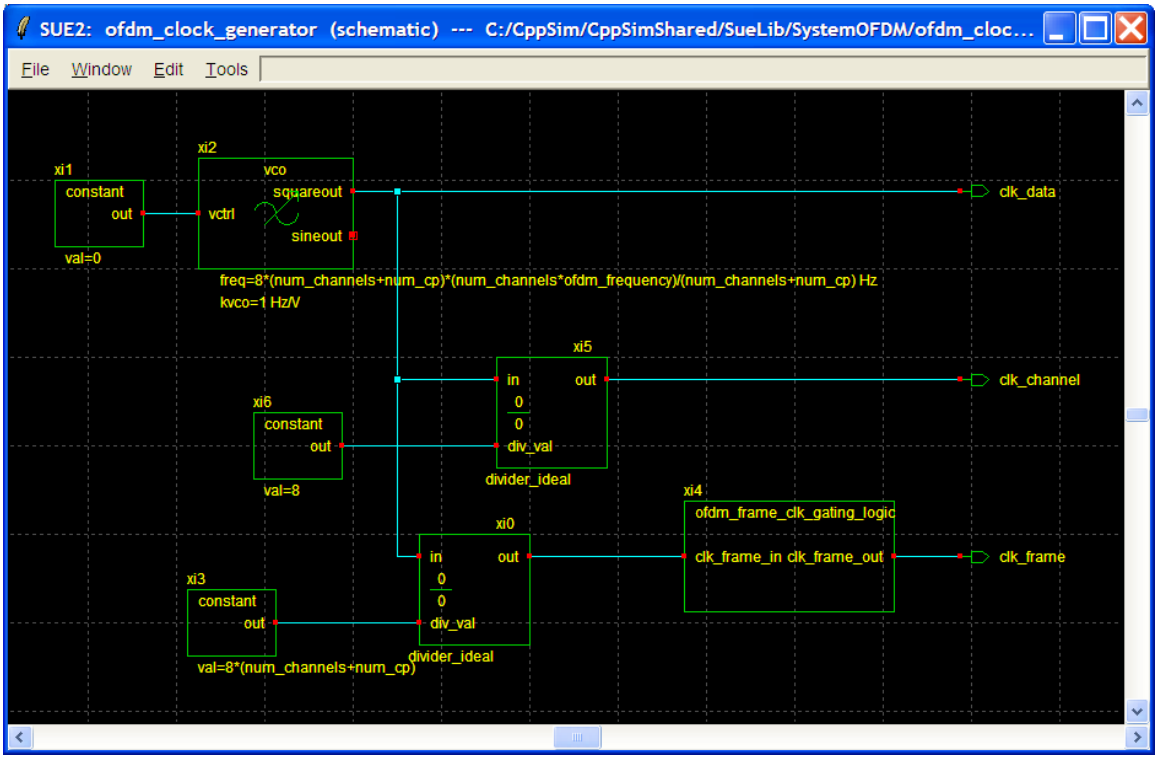




These three clocks are generated at the beginning of the simulation by **ofdm_clk_generator** in **ofdm_overall_system_dt** as below by specifying **ofdm_frequency**, **num_channels** and **num_cp**.



Click on **ofdm_clk_generator** and press **e** to see how the clocks are generated. You will see:



This indicates that the simulation clocks must be related as below. Ts is set in **test.par**.

Ts = 1 / (4 * 8 * num_channels * ofdm_frequency)
clk_data = 8 * num_channels * ofdm_frequency
clk_channel = num_channels * ofdm_frequency
clk_frame = (num_channels * ofdm_frequency) / (num_channels + num_cp)

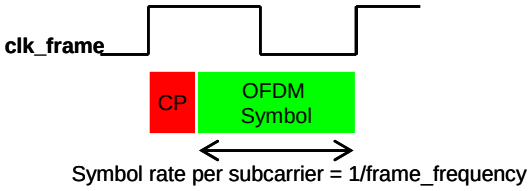


Figure 15: A clk_frame cycle includes a CP + OFDM symbol duration.

B. BER and SNR Measurements

In performance evaluation of communication systems, the relevant Signal-to-Noise Ratio (SNR) is the SNR at the input of the demodulator (or detector). Under an AWGN performance, this SNR is related the BER in a close from expression [4]. Here we derive the BER-SNR expression for rectangular M-QAM symbols such as {4-, 16-, 64-, 256-QAM} – as used in this simulation. Since a rectangular M-QAM transmission (where M = 2^{2k}) is equivalent to two independent $\sqrt{M}-PAM$ transmissions, the symbol error probability of an M-QAM, P_{error}^{M-QAM}, is related to the symbol error probability of an $\sqrt{M}-PAM$, $P_{error}^{\sqrt{M}-PAM}$, by

$$P_{error}^{M-QAM} = 1-(1-P_{error}^{\sqrt{M}\_PAM})^2, \qquad P_{error}^{\sqrt{M}-PAM} = 2(1-\frac{1}{\sqrt{M}})Q\left(\sqrt{\frac{3}{M-1}SNR_{symbol}}\right)$$

In addition, the symbol error probability for BPSK modulations is $P_{error}^{BPSK} = Q\left(\sqrt{2SNR}\right) = BER^{BPSK}$.

By using gray-encoding and decoding - where each symbol in the constellation differs in exactly one bit from each of its nearest neighbor symbols - BER can be approximated from the symbol error probability, P_{error}, as below, by assuming that a symbol error results in a bit error. Thus,

$$BER^{M-QAM} = P_{error}^{M-QAM} / (\text{Number of bits per symbol})$$

In this simulation, once the SNRs are estimated, one can use the theoretical BERs to compare with the simulated BERs to determine whether the performance follows that of an AWGN channel. In the OFDM system, the overall BER is determined by the BER of the subchannel with the lowest SNR. Hence, we are interested in BER per subchannel to be less than the target BER. Therefore, we examine BER-SNR relationships on a subchannel basis – i.e. measure BER and SNR for all bins $i = 0,...,N-1$. In addition, to see the effect of channel fading on SNR more directly, we look at the SNR normalized by the transmitted signal power, i.e.

$$SNR_i = \frac{E[X_i^{\,2}]|H_i|^2}{E[N_i^2]} \;\rightarrow\; \text{Normalized-}\, SNR_i = \frac{|H_i|^2}{E[N_i^2]}$$

The BER versus normalized SNR for each subchannel *i* is shown in Figure 16 for the modulation types used in this simulation. This plot can be obtained by running **ber_snr_curves** in MATLAB.

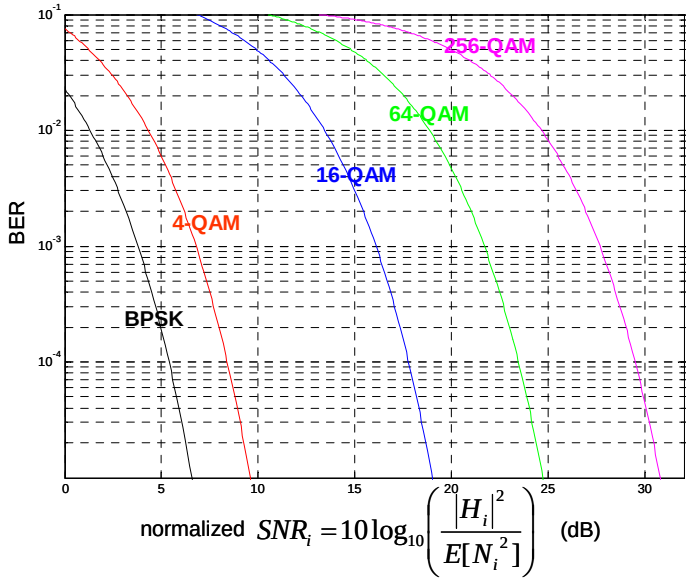


Figure 16: Plot of BER versus normalized SNR for ith subchannel for {BPSK, 4-, 16-, 64-, 256-QAM}.

Conclusion

This tutorial provides a discrete-time simulation platform of narrowband or wideband OFDM systems. In addition, the package includes MATLAB scripts for post-processing of simulated signals for further analysis. It introduces the main OFDM signal characteristics through some simulation results. This concludes the part I of the tutorial that is in the discrete-domain. It is particularly useful for analysis of system algorithms that require perfect performance of the data-converters and RF front-ends.

A natural extension of this OFDM transceiver model in CppSim would be to include the data-converters and some RF front-end components. The codes for some of these modules have already been provided in Part I. So, the interested user can add models of RF blocks to this simulation.

Acknowledgement

I would like to thank my supervisor, Prof. Charles G. Sodini, for his invaluable guidance and encouragement throughout this work. In addition, I would like to thank Prof. Michael Perrott for providing the codes for some of the main OFDM modules and technical support of CppSim simulation tool. I am very grateful to Prof. Davide Dardari for the valuable discussion and providing the parameters of COST259 and Naftali Channel Models. This work is funded by the National Science Foundation (NSF).

References

[1] "European co-operation in the field of scientific and technical research," Cost 259 meeting report TD(98)070.

[2] Rappaport, T. S. (1999). Wireless Communications Principles and Practice. Upper Saddle River, NJ: Prentice Hall.

[3] IEEE 802.11-97/96, Naftali Chayat, September 1997.

[4] Proakis, J. G., & Salehi, M. (2002). Communication Systems Engineering (4th ed.). NY: Prentice Hall.