Rahmaan Lodhia

GTID: 902502749

ECE 4271

Project #2b

Binary Communication Transceiver Design with LabVIEW

1. Transceiver Algorithm

1.1. Overview

The LabVIEW digital communication simulation has five stages: the transmitter, the addition of white Gaussian noise, the receiver, the BER calculation, and the theoretical BER calculation and comparison. The top level front panel takes the following information from the user: the type of constellation used, the range of SNR in dB to measure BER, the samples per symbol, the total number of symbols, and the parameters for the filter used in the modulation and demodulation. With these values, the program first creates the specified symbol map (**Generate symbol map.vi**) and a stream of bits to simulate a signal (**Random bit generation.vi**). The simulated bits are then sent to the transmitter stage, which will encode the bits into symbols as defined by the specified modulation constellation (**Map symbols.vi**), up-sample the signal (**Upsampling.vi**), and convolve the resulting signal with the filter (**Perform convolution.vi**). Then from the transceiver, the encoded signal will have white Gaussian noise with a standard deviation of one added to it (**Add AWGN.vi**), corrupting the signal, and then the corrupted signal will be scaled according to the specified SNR. The corrupted signal will enter the receiver stage, where it will first be convolved with the matched filter (**Perform convultion.vi**), decimated by the sample up-sample factor (**Decimation.vi**), and then finally truncated to remove the excess transient values from filtering to return the signal to its original size (**Truncation.vi**). Then the resulting signal will be mapped to the estimated symbols in order to eliminate the noise involved and then decoded into bits (**Demap symbols.vi**). The program will take the resulting estimated signal and compare it to the original counting the number of bit errors, and take the total number of errors and divide by the total number of bits to arrive at the bit error rate (**Calculate BER.vi**). Finally, the program will calculate the theoretical BER (**Theoretical BER.vi**), compare it with the measured BER, and output a graph with both values plotted.

1.2. Transmitter Design

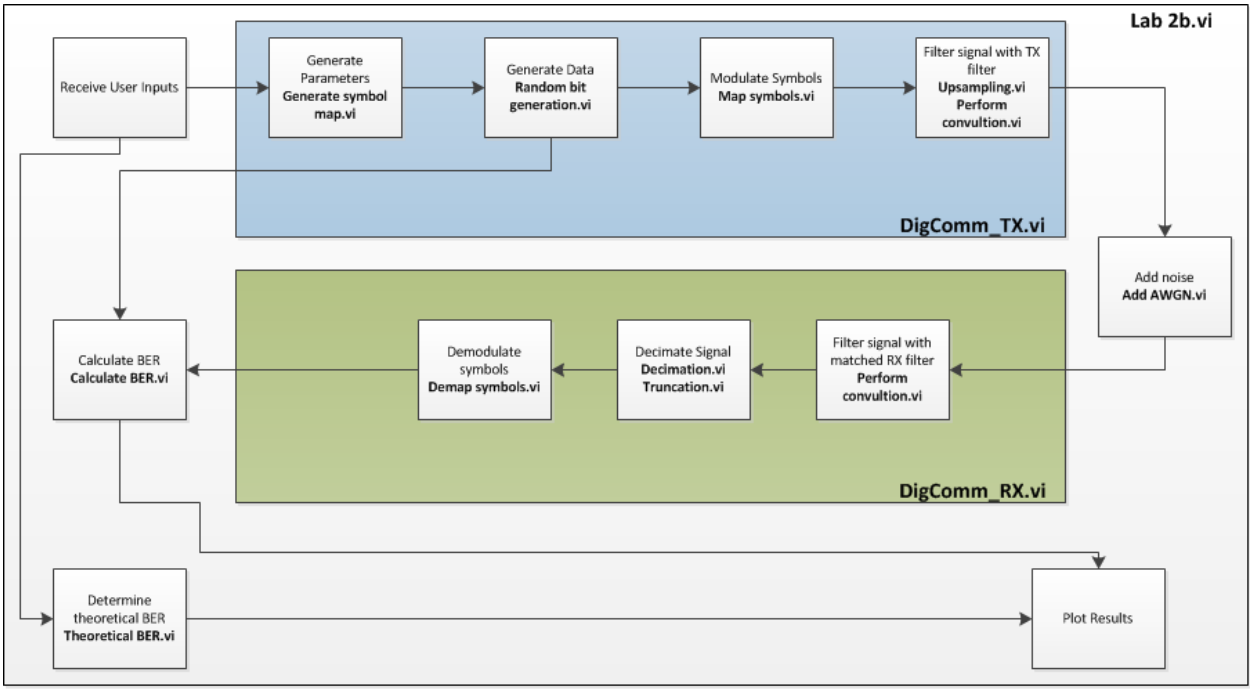
The transmitter stage takes place in the **DigComm_TX.vi** file. The function will first determine from the constellation mapping, the bits per symbol, and generate the random

number of bits. Then, it will iterate through the signal and encode each sample of bits into the proper constellation mapping. When the signal has been encoded, the function will then take the encoded signal and normalize it by dividing the signal by the average signal energy of the constellation type. Taking the normalized signal, the function will then upscale the signal by the number of samples per symbol as indicated by the user. The upscaled signal will then be filtered by the user-defined filter and then sent to be corrupted by white Gaussian noise. Before the signal is corrupted, it is scaled by the symbol power as defined from the SNR, and this output signal will be sent through the channel, where noise is added, into the receiver.

1.3. Receiver Design

After the signal is corrupted by the addition of white noise from the **Add AWGN.vi**, the signal is sent to the receiver. The receiver process is in the **DigComm_RX.vi** file. The program takes in the corrupted signal and it first filters it with the matched filter in correspondence to the filter used by the transmitter. Then, the filtered signal is decimated. Due to the filtering, the decimated signal has extra transient values. The program then truncates the signal by removing extra values from the beginning and the end of the signal depending on the filters used. If a raised root cosine filter is used, the length of the filter is truncated from both the beginning and the end of the signal. If a raised cosine filter is used, half the length of the filter is truncated from the beginning and end of the signal. If a rectangular filter is used, no changes are made to the signal. Next, the program determines which modulation symbol mapping to use to compare the corrupted signal with. The corrupted signal is then scaled down by the average symbol power for the modulation size to a normalized, corrupted signal. When the modulation is selected, the function will then iterate through each corrupted signal and compare it against each symbol of the constellation mapping, and it will determine which of the symbols is closest to the corrupted signal. This signal will be considered the estimated value for the corrupted signal. This process will continue until a vector estimating the correct values for the corrupted signal is created, which will eliminate the noise. Each symbol will then be converted to their corresponding bits as defined by the mapping. This final sequence of estimated bits is sent back to the top-level program where it will calculate the BER.

2. Transceiver LabVIEW Block Diagram



3. Resulting Measurements

3.1. Benchmark and Resulting BER for Constellations

The BER vs. SNR was measured for each of the four constellation types with a measured curve and a theoretical curve. The theoretical BER for the BPSK and the QPSK was calculated with the following function:

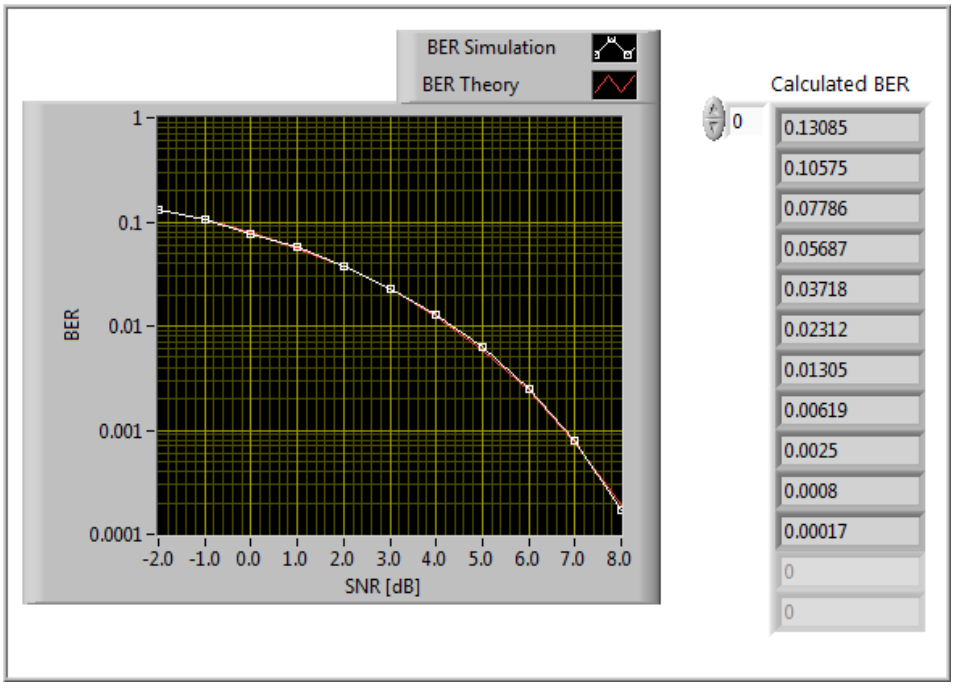
$$BER = \frac{1}{2} erfc\left(\sqrt{\frac{E_b}{N_0}}\right) or\ BER = Q\left(\sqrt{\frac{2E_b}{N_0}}\right)$$

The theoretical BER for the 16QAM and the 64QAM was calculated with this function:

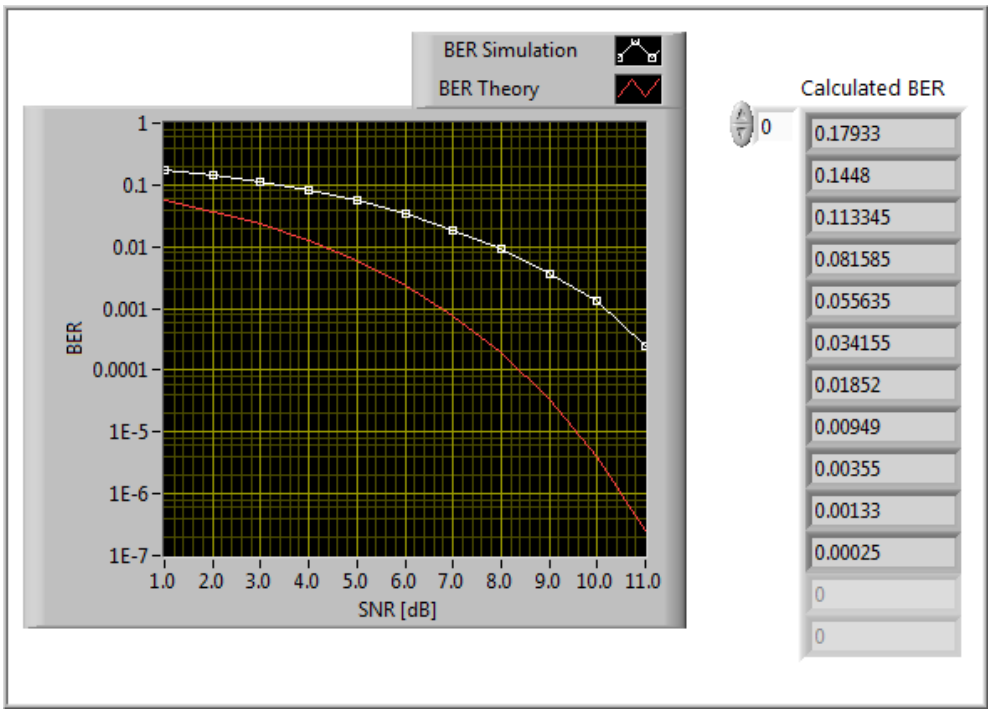
$$BER = \frac{4}{k}\left(1 - \frac{1}{\sqrt{M}}\right) Q\left(\sqrt{\frac{3k}{M-1}\frac{E_b}{N_0}}\right)$$

In the function above, M is the number of symbols in the constellation and k is the number of bits per symbol. The parameters used for the measurements were: 100,000 symbols, 8 samples per symbol, raised root cosine filter, alpha value of 0.5, and filter length of 6. The ranges for the data were (-2:1:8) dB for BPSK, (1:1:11) dB for QPSK, (3:1:13) dB for 16QAM, and (5:1:15) for 64QAM.

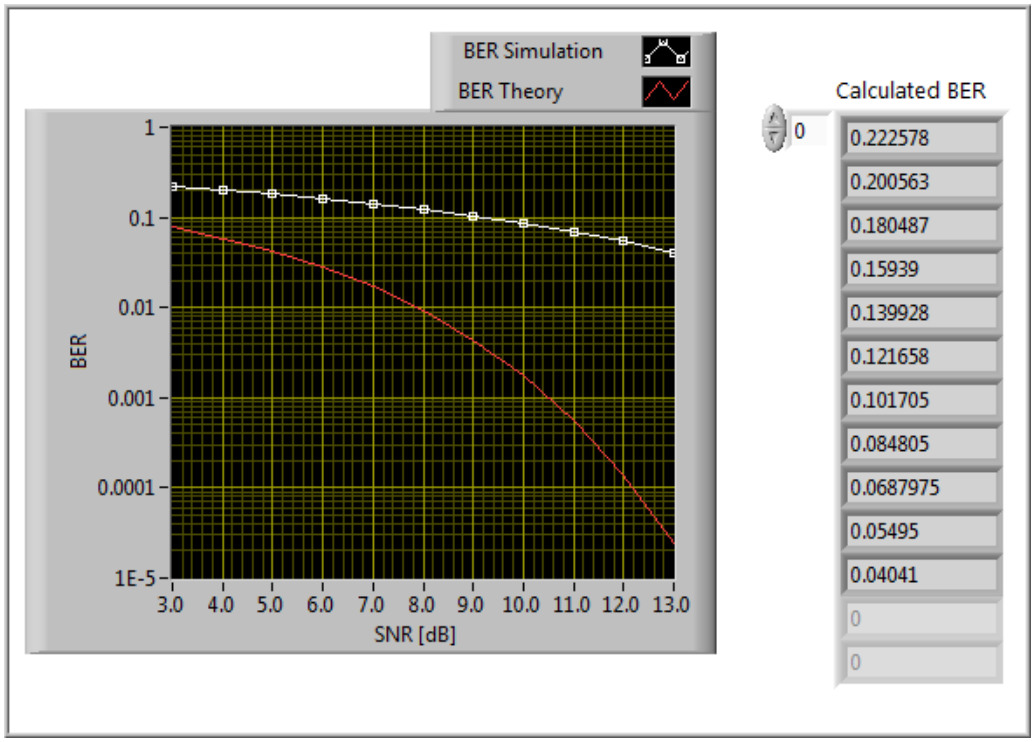
3.1.1. BER vs. SNR for BPSK



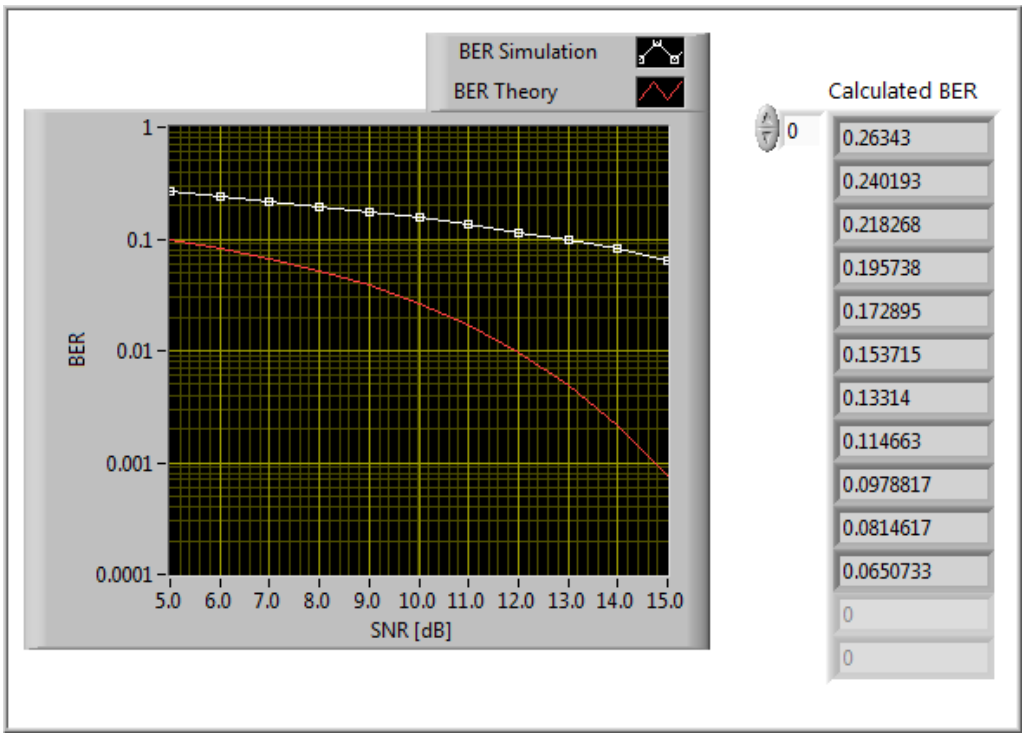
3.1.2. BER vs. SNR for QPSK



3.1.2. BER vs. SNR for 16QAM



3.1.2. BER vs. SNR for 64QAM



3.2. Constellation Effect on BER Performance

From the results, it appears that the most accurate of all the constellations was the BPSK. The BPSK almost had a perfect match with the theoretical BER values. Following the BPSK, the QPSK had the second best performance in terms of BER, although it still varied from the theoretical at higher SNR values. The 64QAM was the least accurate of the four, and it was far from the theoretical values. This error may be due to the spacing between the constellation symbols or the exact mapping itself as the receiver is trying to decode the corrupted signals.

3.3. Filter Effect on Constellation

The filter used in the application did have effects on the constellation graphs and the mapping. The figure below shows a QPSK mapping with the original raised root cosine filter:

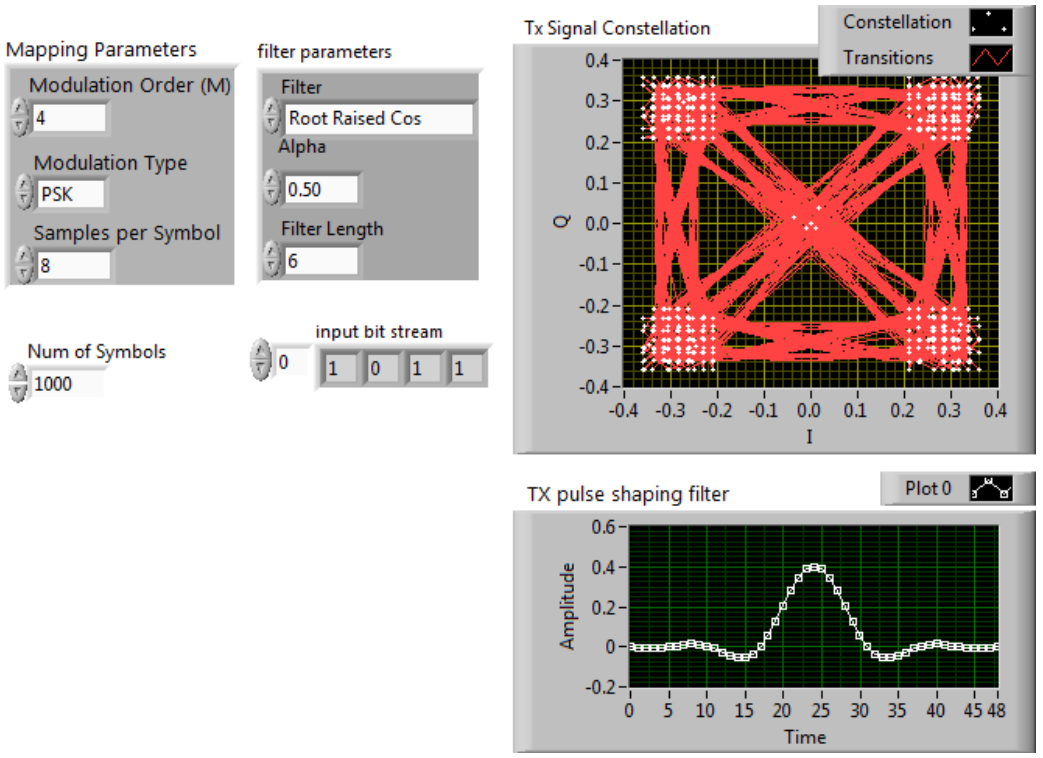


Figure 1. QPSK Mapping with original filter

This mapping was then measured when the filter had an increased alpha. The following figure demonstrates how the constellation changes with an increasing alpha:

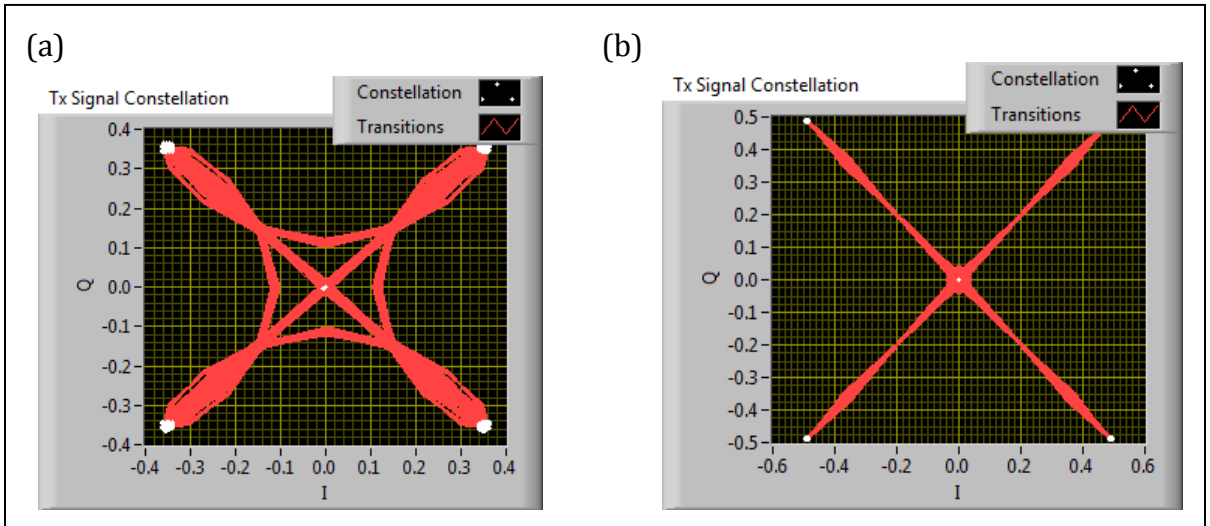


Figure 2. QPSK Mapping with (a) alpha = 1.50 and (b) alpha = 3.50

The figure above shows that increasing the alpha value for the filter limits the transitions that the constellation map contains and moves it toward a rectangular filter effect.

Changing the length of the filter creates a different effect. Figure 3 shows the effect on the mapping with an increasing filter length:

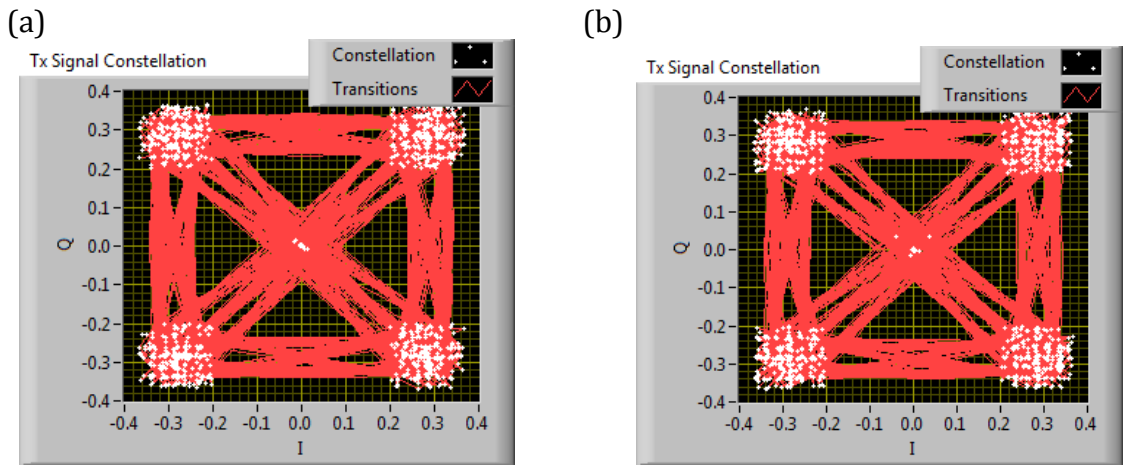


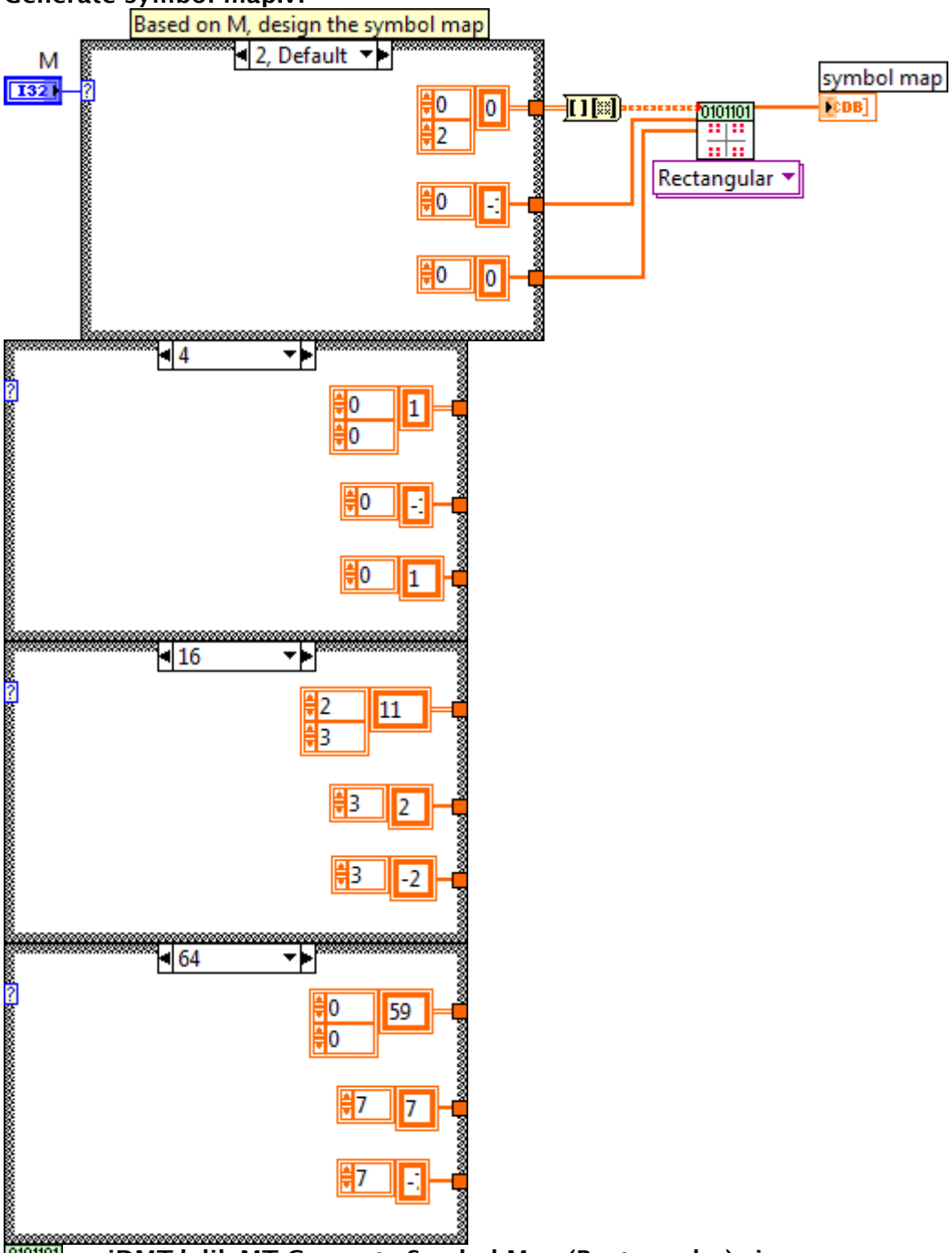
Figure 3. QPSK Mapping with (a) length = 40 and (b) alpha = 100

Increasing the length of the filter seems to slightly increase the transitions between the constellation points. The effect of the length is not as influential as the alpha. However, an odd length may create undesired effects in mapping and cause some points to appear midway between symbols.

These measurements were made using the raised root cosine filter. The raised cosine filter also demonstrated similar results, while the rectangular filter in general produced mapping with very little transitions. It appears that altering the alpha and the length for the cosine filters makes them appear more like the rectangular filter in terms of mapping.

4. LabVIEW Code

Generate symbol map.vi



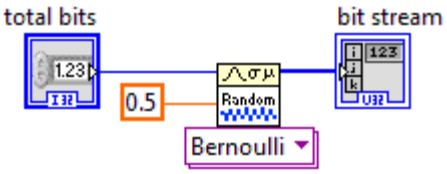
niDMT.lvlib:MT Generate Symbol Map (Rectangular).vi
C:\Program Files\National Instruments\LabVIEW 2011\vi.lib\addons\Modulation\Digital\General\MT Generate Symbol Map (Rectangular).vi

niDMT.lvlib:MT Generate Symbol Map.vi
C:\Program Files\National Instruments\LabVIEW 2011\vi.lib\addons\Modulation\Digital\General\MT Generate Symbol Map.vi

NI_Matrix.lvlib:Array To Matrix - RA2.vi
C:\Program Files\National Instruments\LabVIEW 2011\vi.lib\Analysis\Matrix\Numeric\Conversions\Array To Matrix - RA2.vi

NI_Matrix.lvlib:Array To Matrix.vi
C:\Program Files\National Instruments\LabVIEW 2011\vi.lib\Analysis\Matrix\Numeric\Conversions\Array To Matrix.vi

Random bit generation.vi

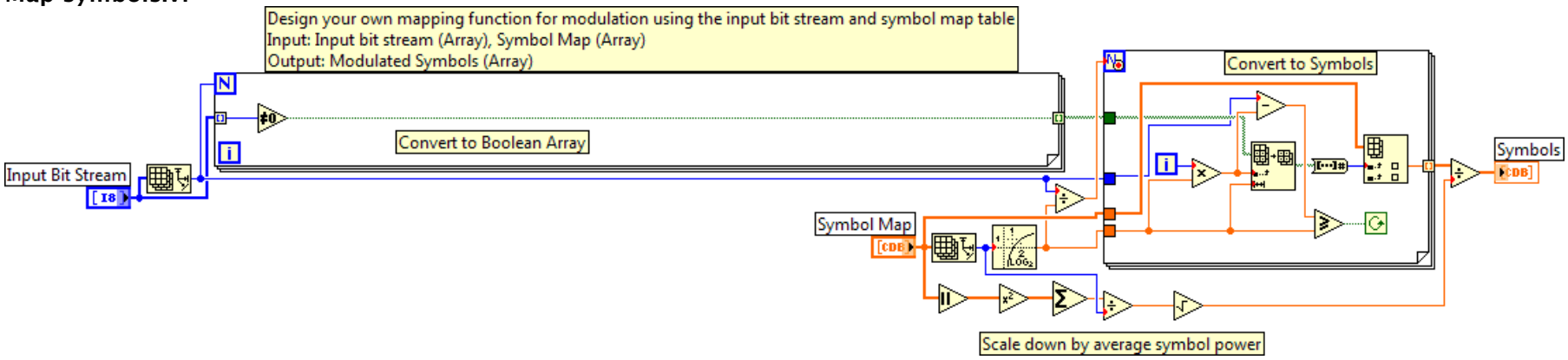


Generate random bit sequences
Input: total bits (scalar)
Output: bit stream (0, 1 array)

NI_Gmath.lvlib:Discrete Random.vi
C:\Program Files\National Instruments\LabVIEW 2011\vi.lib\gmath\statdist.llb\Discrete Random.vi

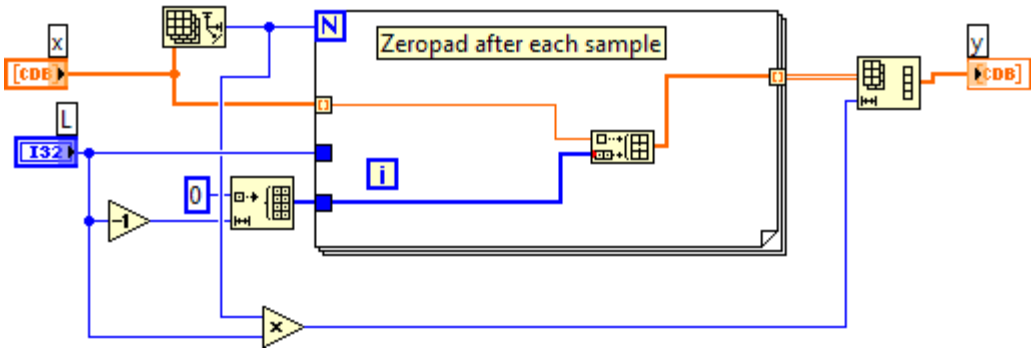
NI_Gmath.lvlib:Bernoulli Random.vi
C:\Program Files\National Instruments\LabVIEW 2011\vi.lib\gmath\statdist.llb\Bernoulli Random.vi

Map symbols.vi

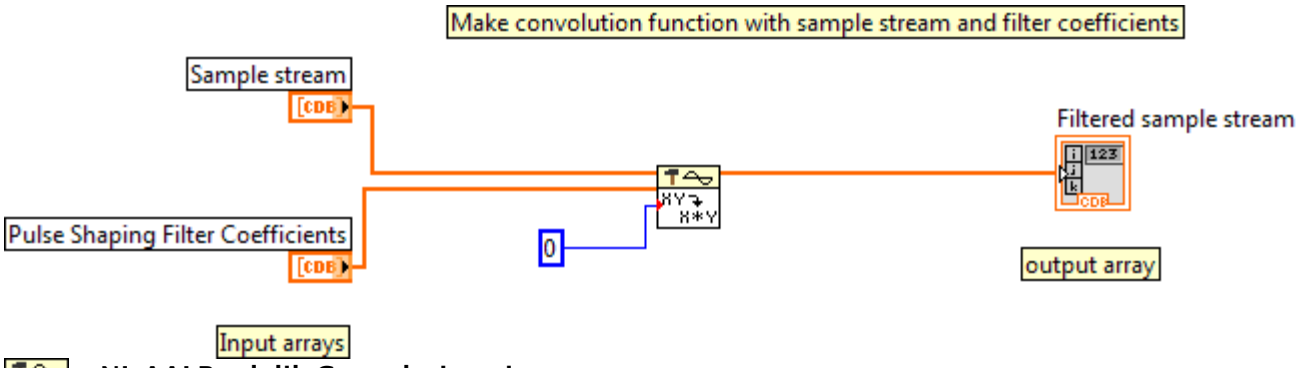


Upsampling.vi

Upsample your signal
Input: x (input signal), L (upsample factor)
Output: y (output signal)



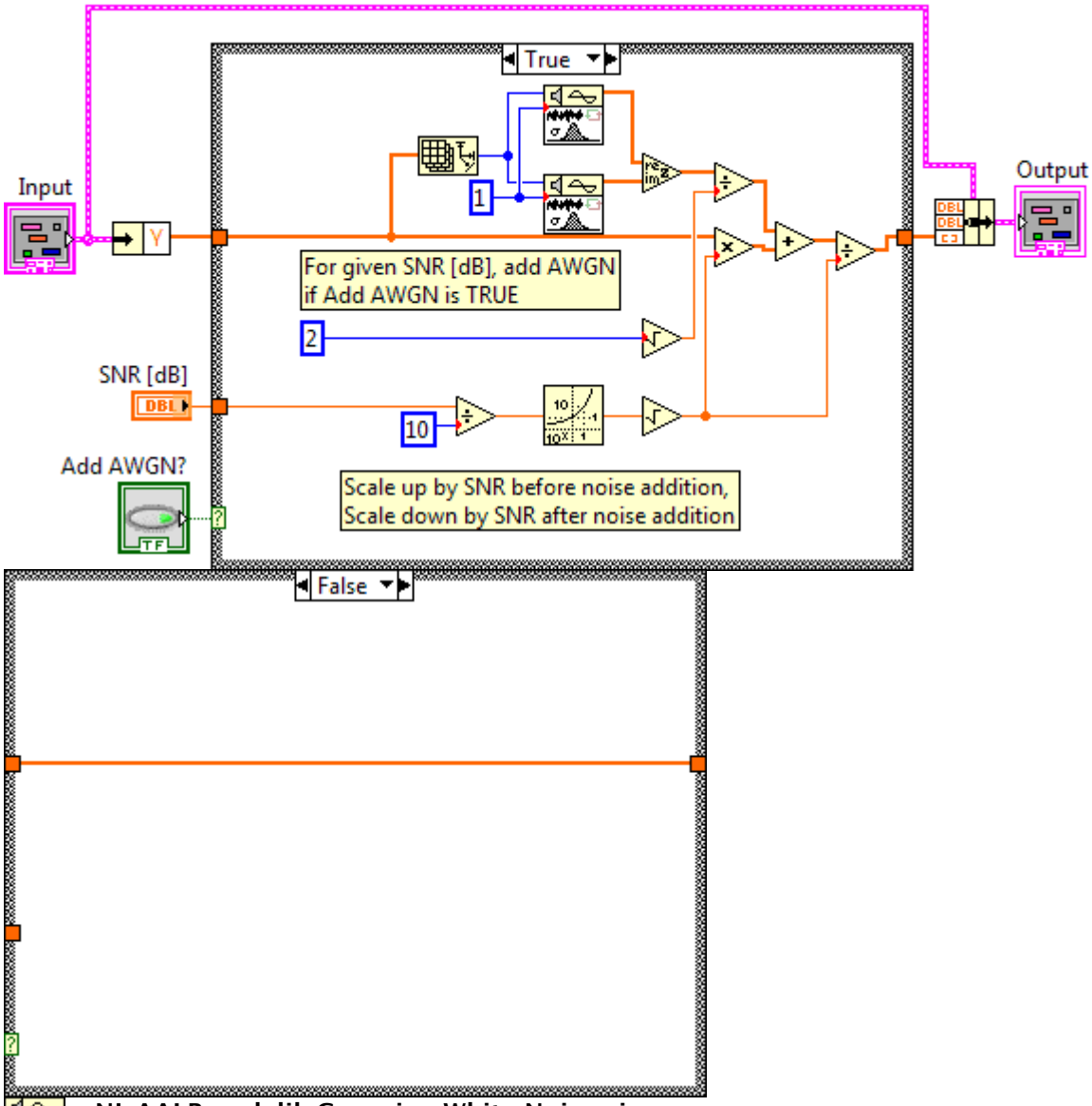
Perform convolution.vi



NI_AALPro.lvlib:Convolution.vi
C:\Program Files\National Instruments\LabVIEW 2011\vi.lib\Analysis\2dsp.llb\Convolution.vi

NI_AALPro.lvlib:1D Convolution (CDB).vi
C:\Program Files\National Instruments\LabVIEW 2011\vi.lib\Analysis\2dsp.llb\1D Convolution (CDB).vi

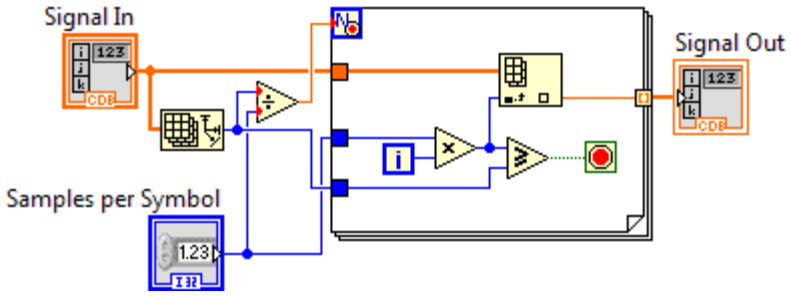
Add AWGN.vi



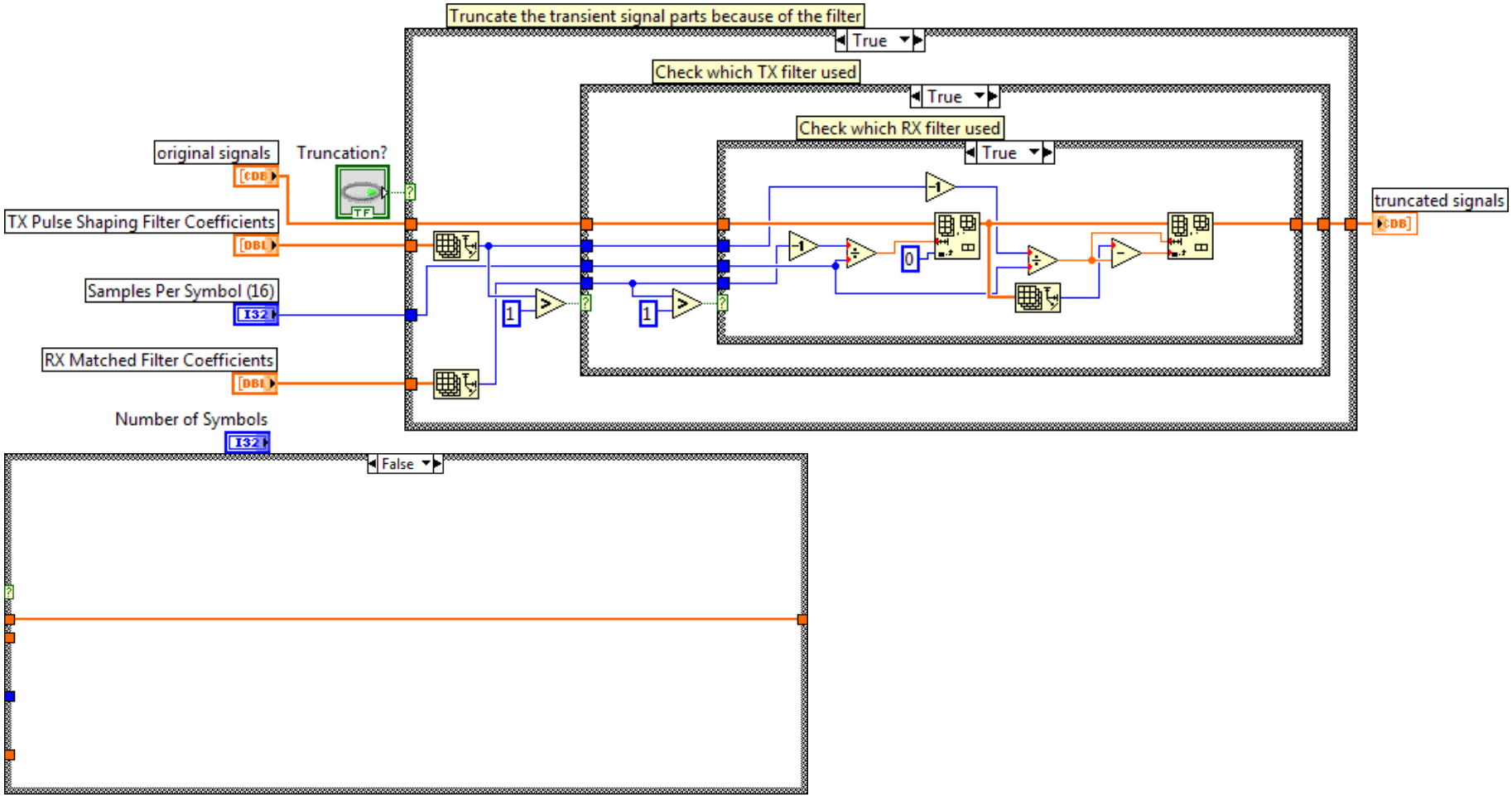
NI_AALBase.lvlib:Gaussian White Noise.vi
C:\Program Files\National Instruments\LabVIEW 2011\vi.lib\Analysis\1siggen.llb\Gaussian White Noise.vi

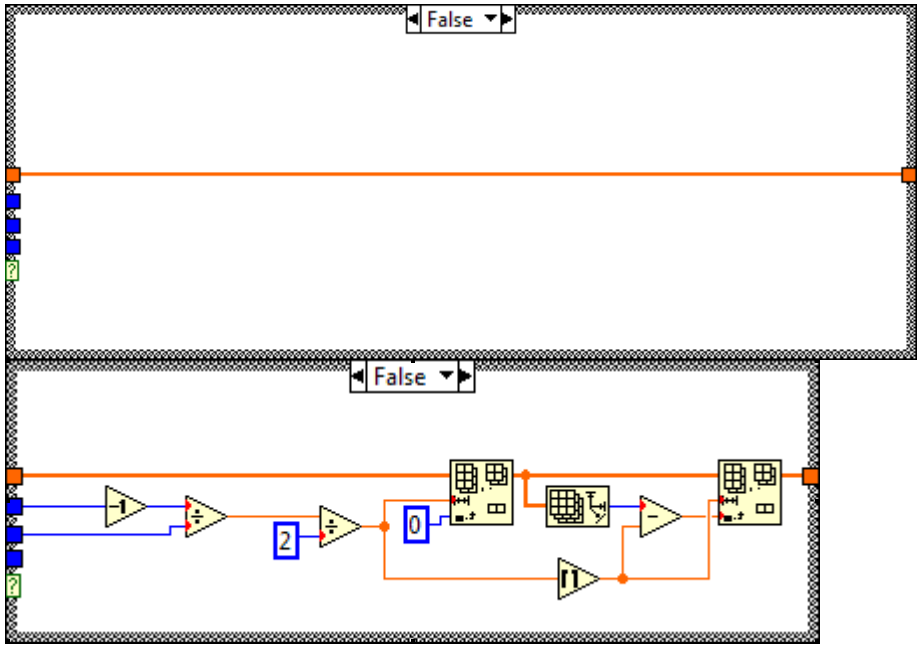
Decimation.vi

Decimate the input signal with the decimation factor (Samples per Symbol)



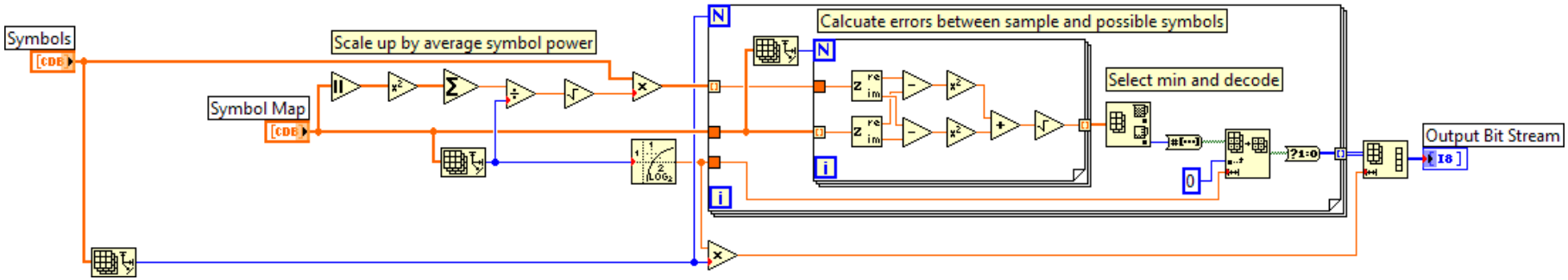
Truncation.vi




False
False
-1
2
0
1

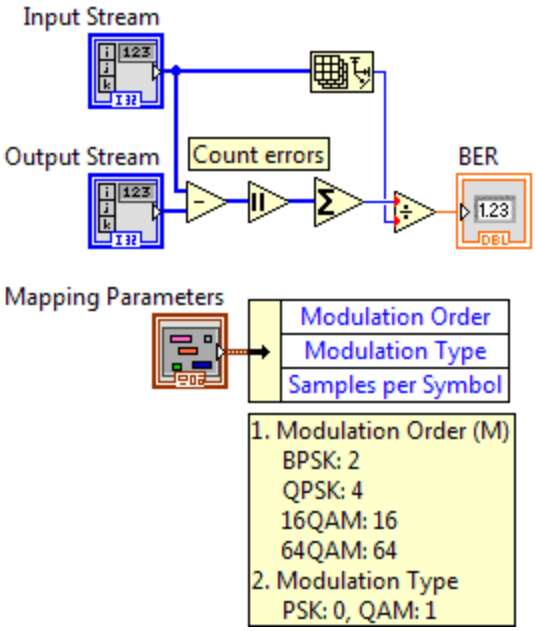
Demap symbols.vi

For given symbols and symbol map, demodulate the symbols into bit stream

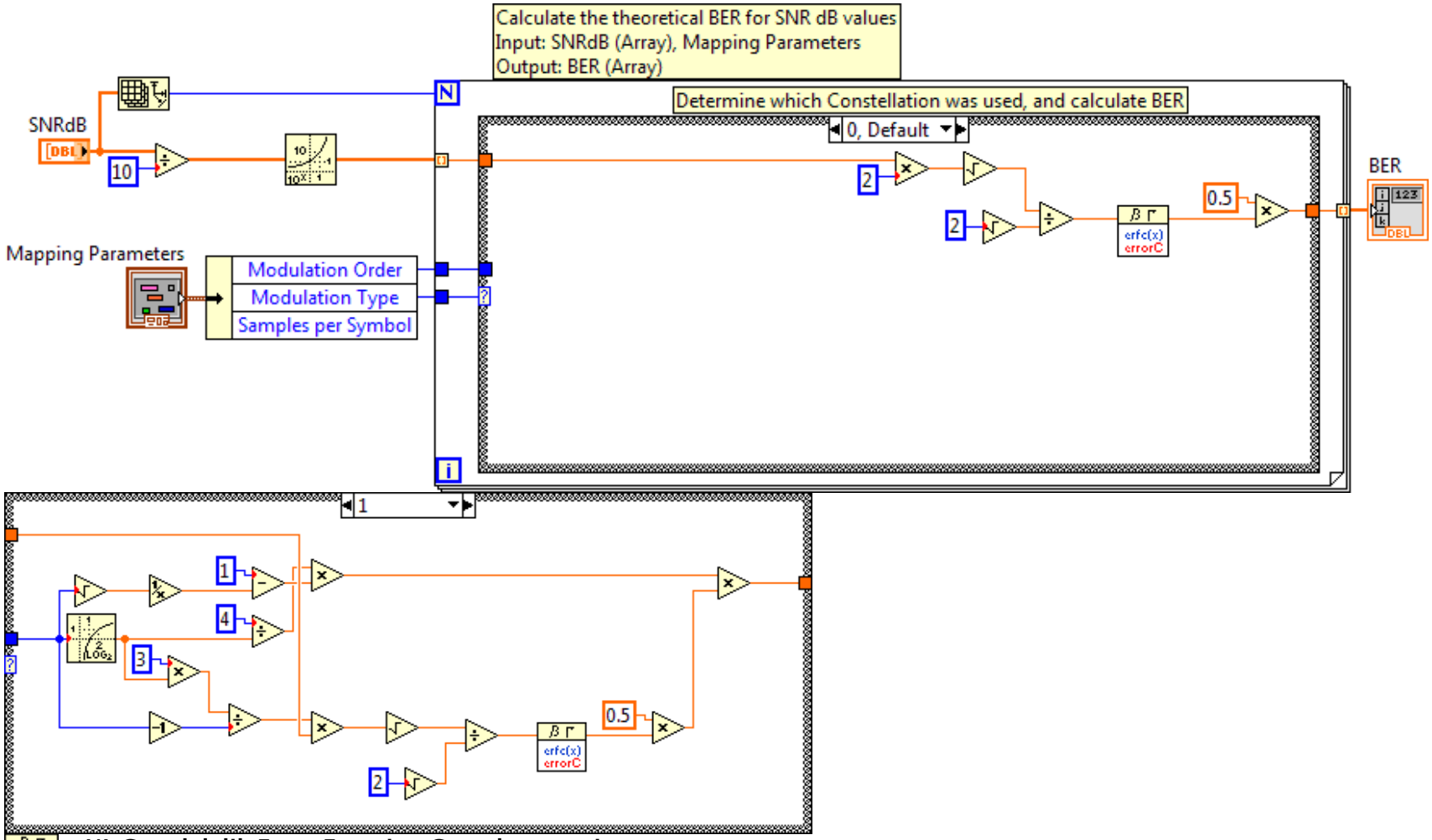


Calculate BER.vi

Build up the block to calculate the BER, BER should be less than 1



Theoretical BER.vi



NI_Gmath.lvlib:Error Function Complement.vi
C:\Program Files\National Instruments\LabVIEW 2011\vi.lib\gmath\SpecialFunctions.llb\Error Function Complement.vi