INTRODUCTION

This document describes the design of a QPSK modem, using model based design techniques.

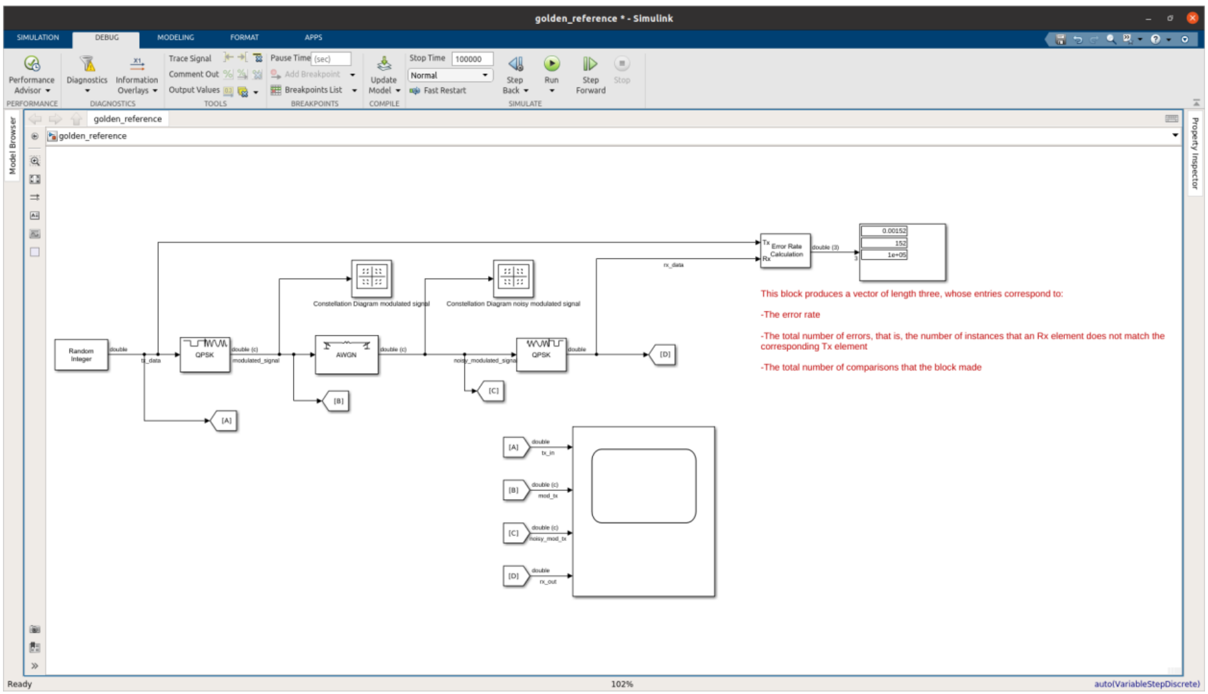
https://www.faststreamtech.com/products/qam-modulator-and-demodulator/

https://scialert.net/fulltext/?doi=jas.2013.385.392

https://www.tutorialspoint.com/principles_of_communication/principles_of_communication_digital_modulation_techniques.htm

THE GOLDEN REFERENCE MODEL

The golden reference model, used as a starting point, is shown below:



A random data source is driving a baseband QPSK modulator. Its output goes through an AWGN channel, and the noisy data is received by a baseband QPSK demodulator.

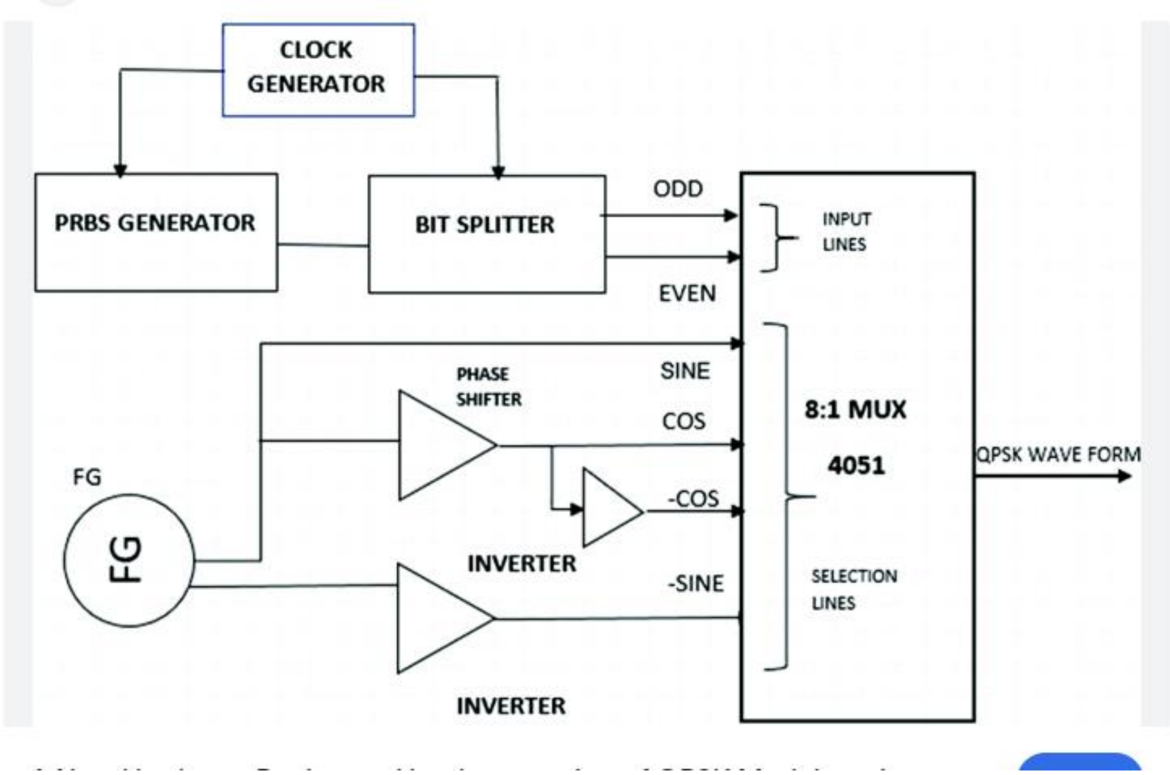
The error rate, key waveforms, and constellation diagrams are shown, during the model simulation.

HIGH LEVEL TRANSMITTER

We can now to implement a QPSK transmitter, such the one described in:

https://scialert.net/fulltext/?doi=jas.2013.385.392

The block diagram is:

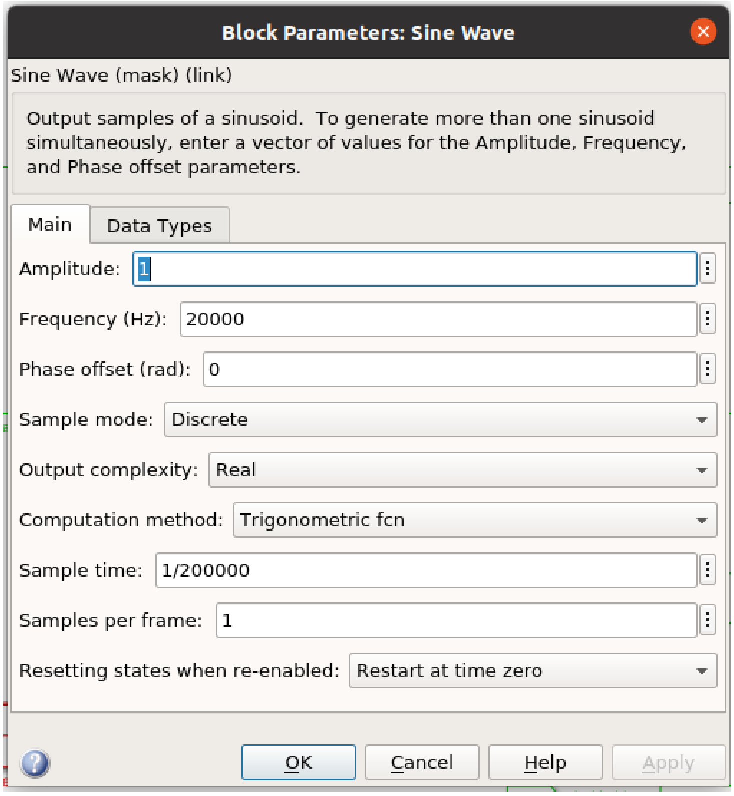


In a QPSK (Quadrature Phase Shift Keying) modulator, the random data typically consists of bits that are grouped into pairs. Each pair of bits corresponds to a specific phase shift in the modulated signal: sin, -sin, cos, -cos.

Since QPSK uses two bits per symbol, the possible values for the random data are all combinations of two bits: 00, 01, 10, and 11.

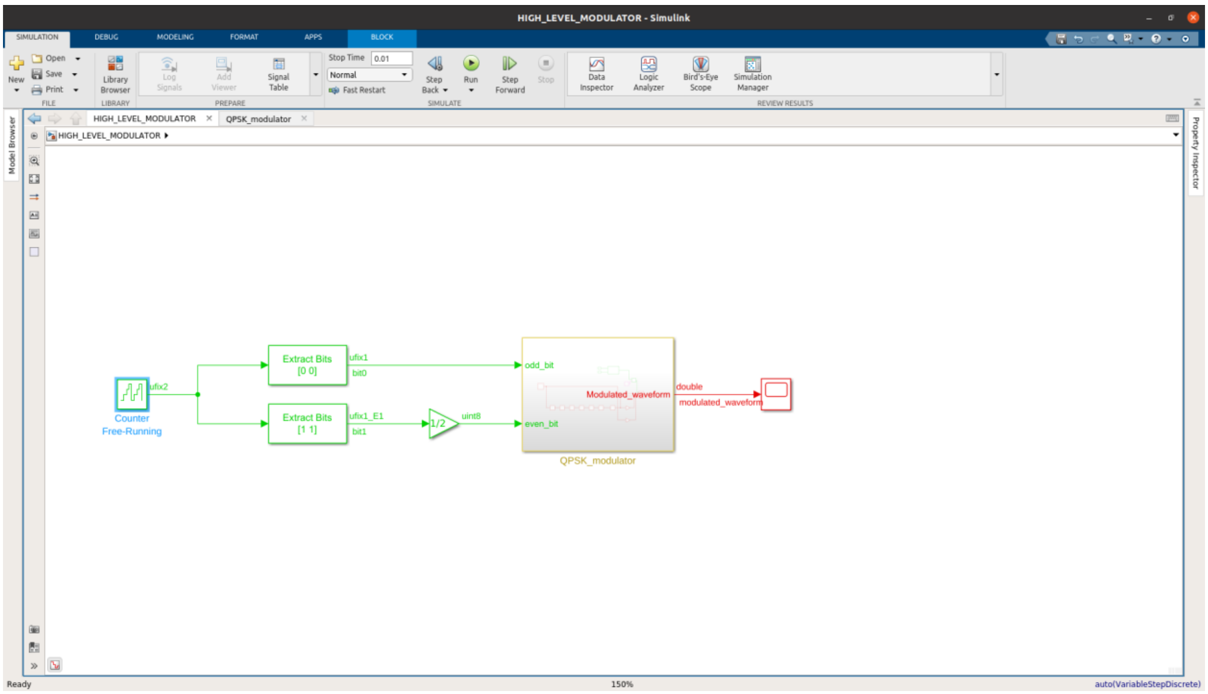
These pairs determine the phase of the transmitted signal, usually mapped to four distinct points on the constellation diagram. So, the random data can assume any of these four two-bit combinations.

That said, we are going to use a carrier with:

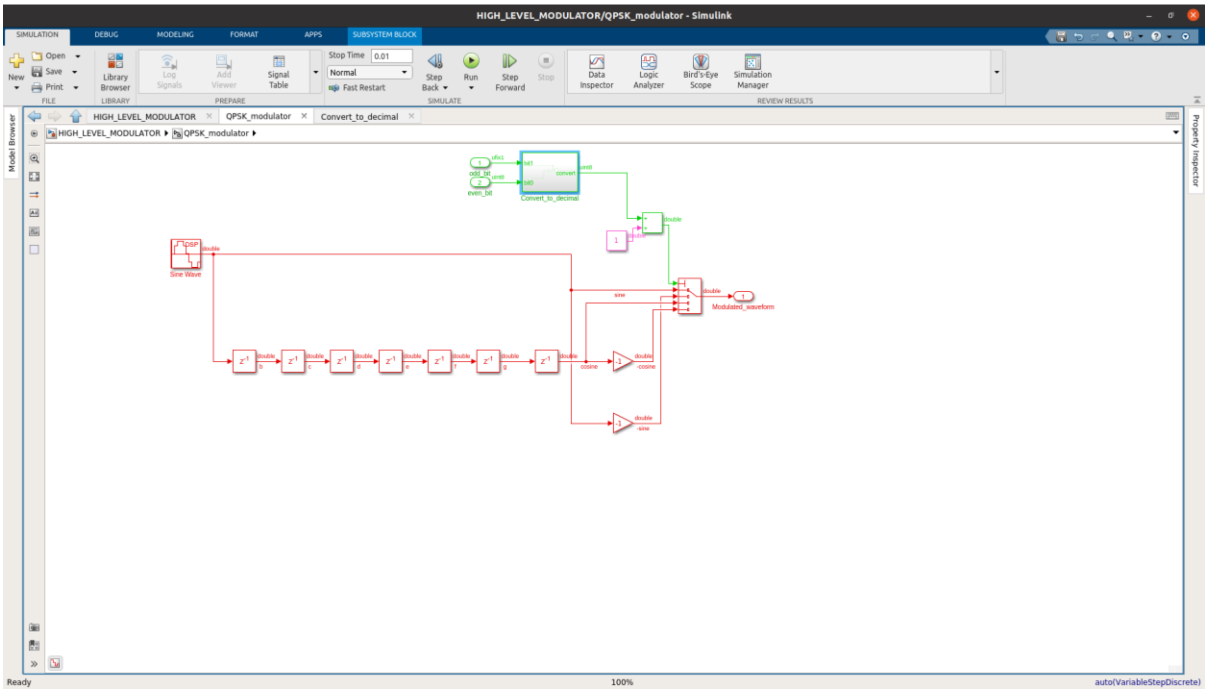


A frequency of 20 KHz, and a sample rate of 200 KHz.

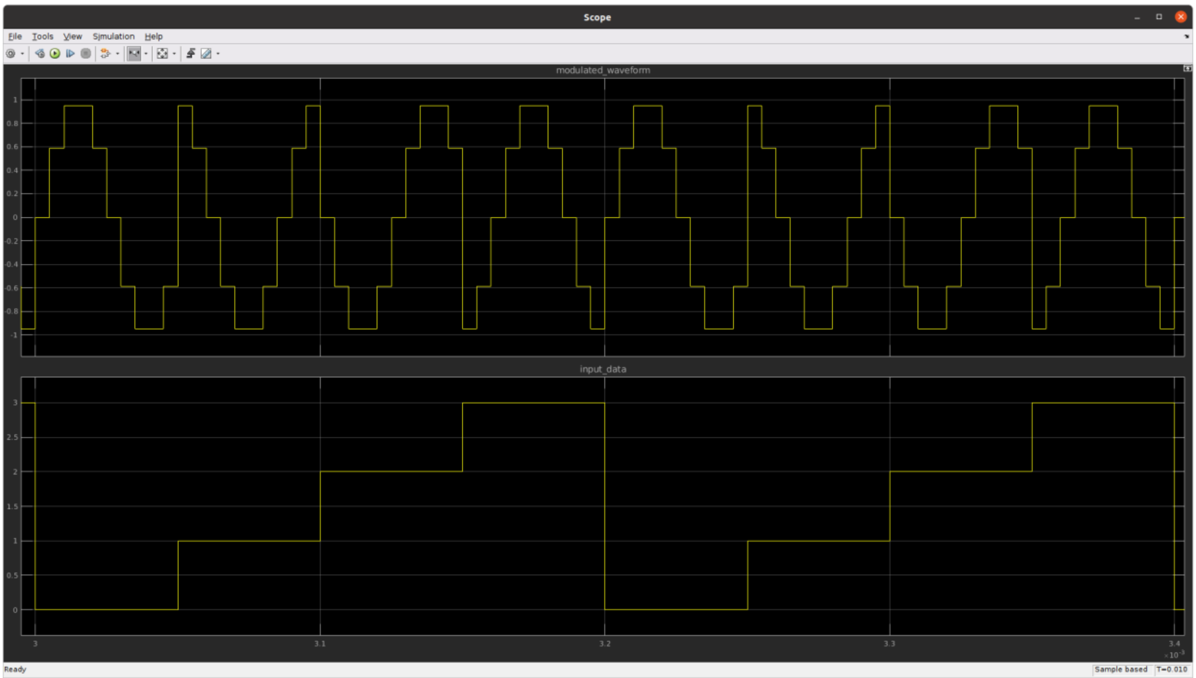
The transmitter has been modelled as follows:



And fed as follows, by the QPSK baseband modulated signal:



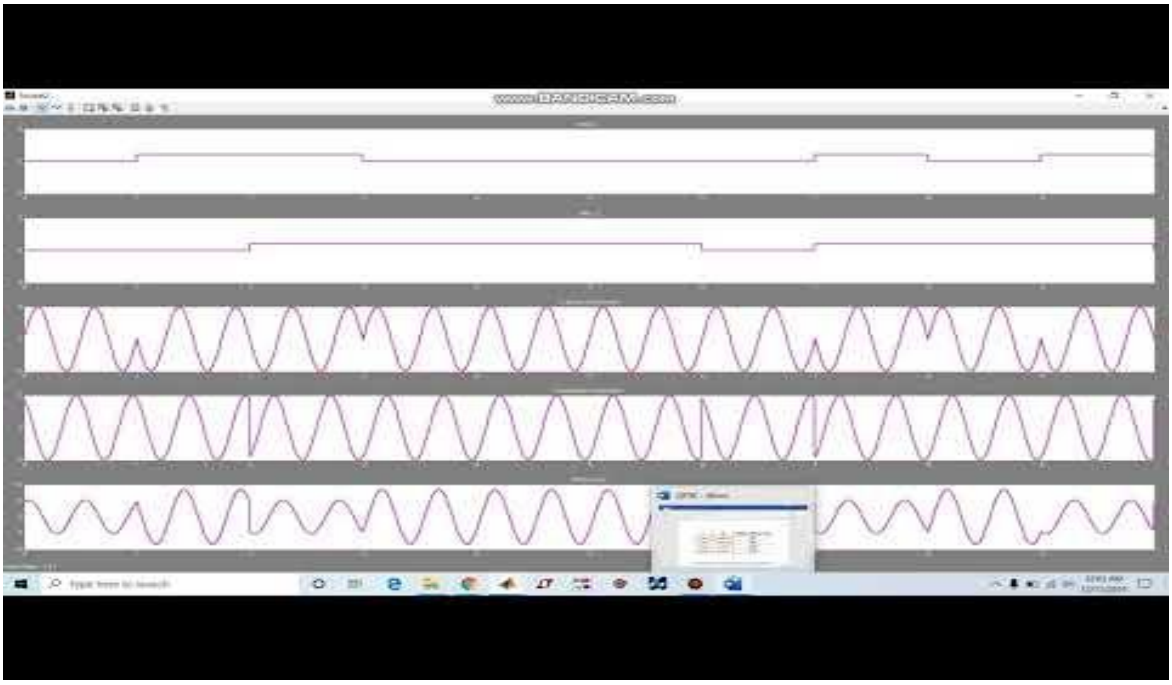
By running the simulation, we can have a look at the output waveforms:



COHERENT RECEIVER

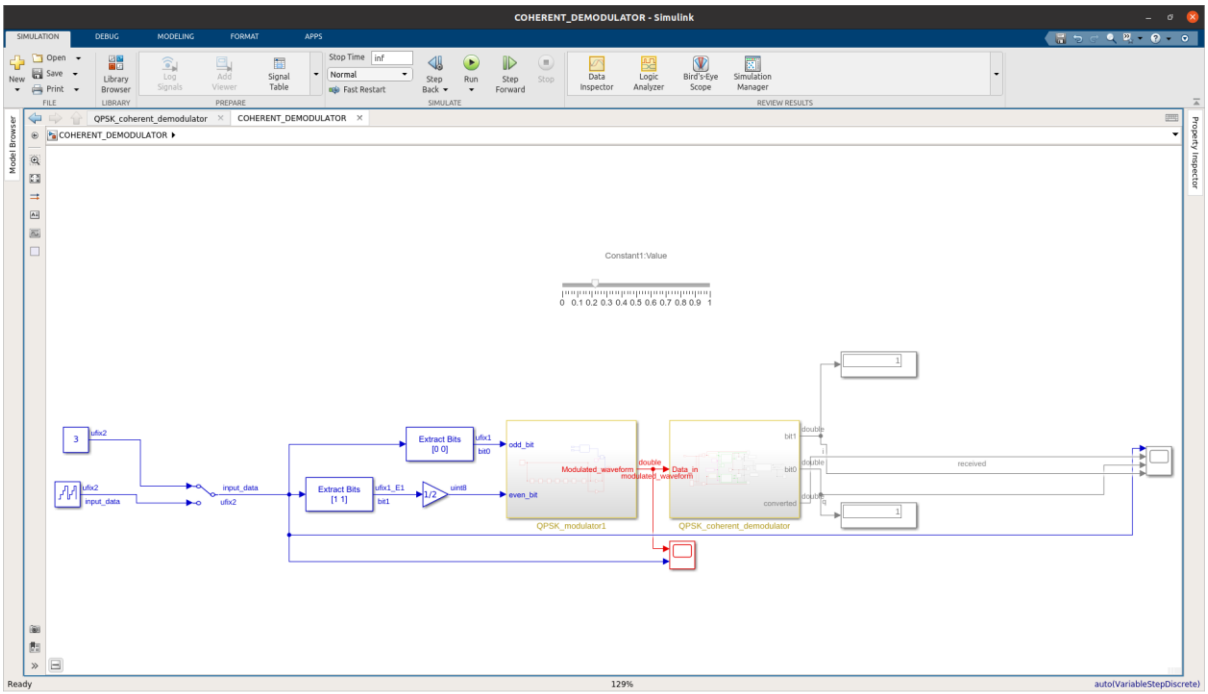
The following sources, have been used as references:

https://www.youtube.com/watch?v=w5chDiZdx0Y



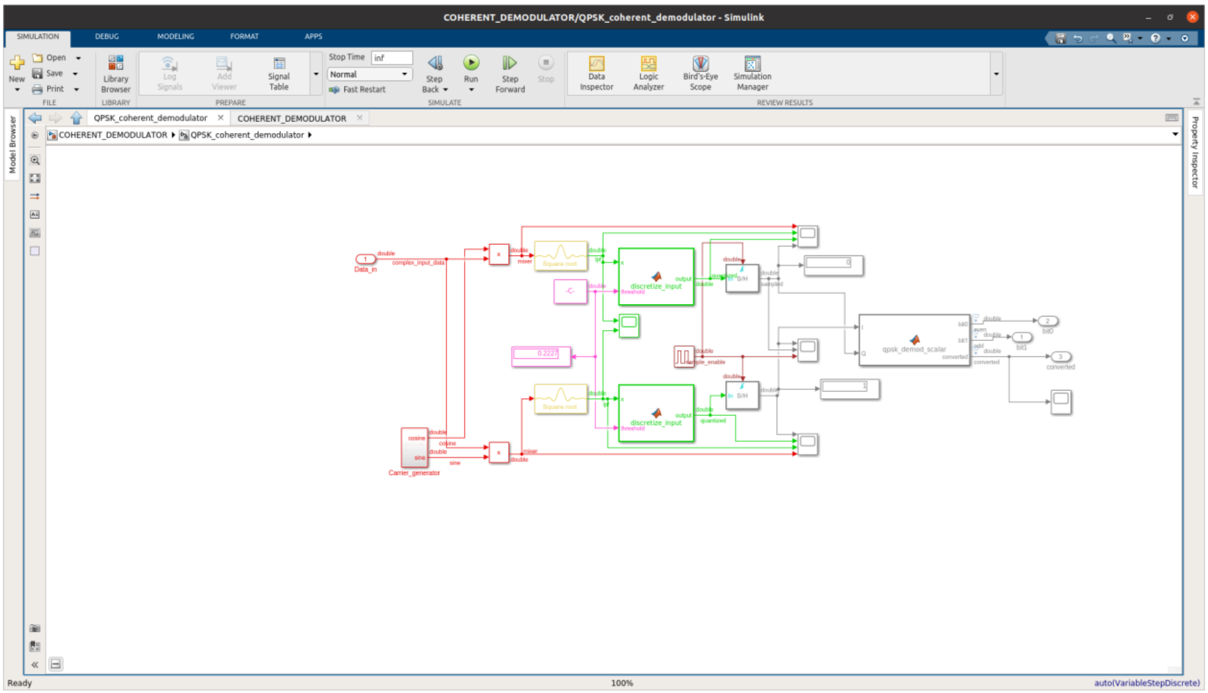
https://www.tutorialspoint.com/digital_communication/digital_communication_quadrature_phase_shift_keying.htm

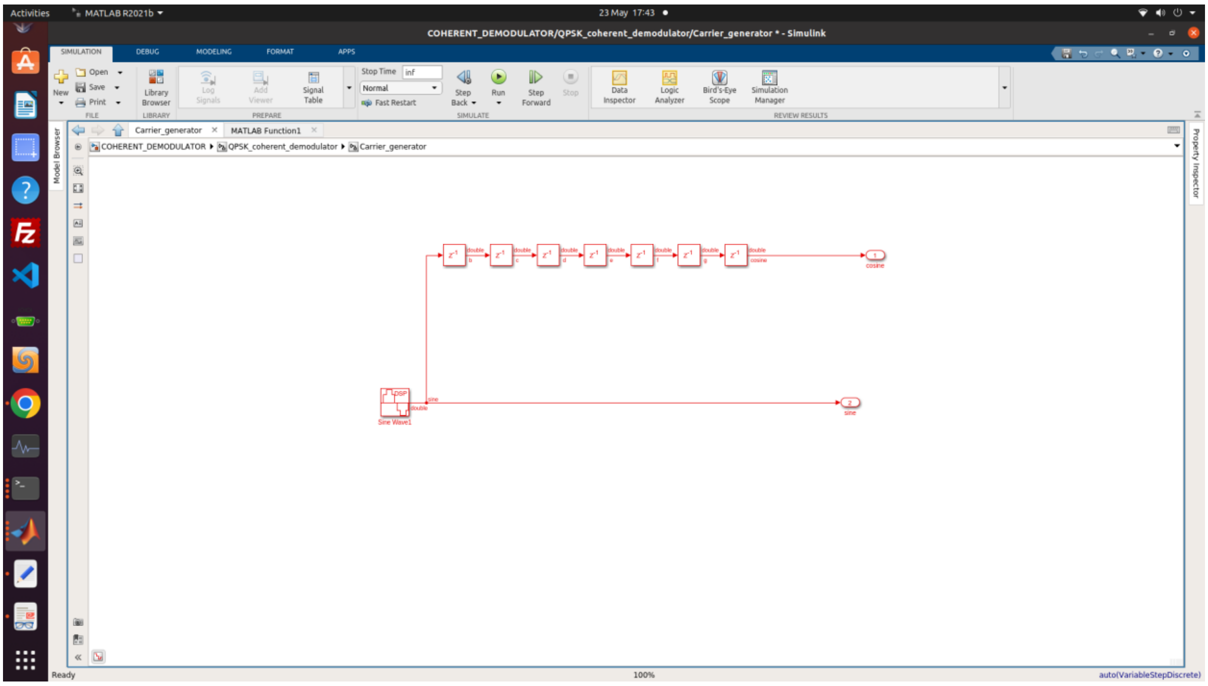
A simulink model consisting of a free-running counter and a constant used as sources, plus a modulator, a demodulator and a scope, has been prepared:

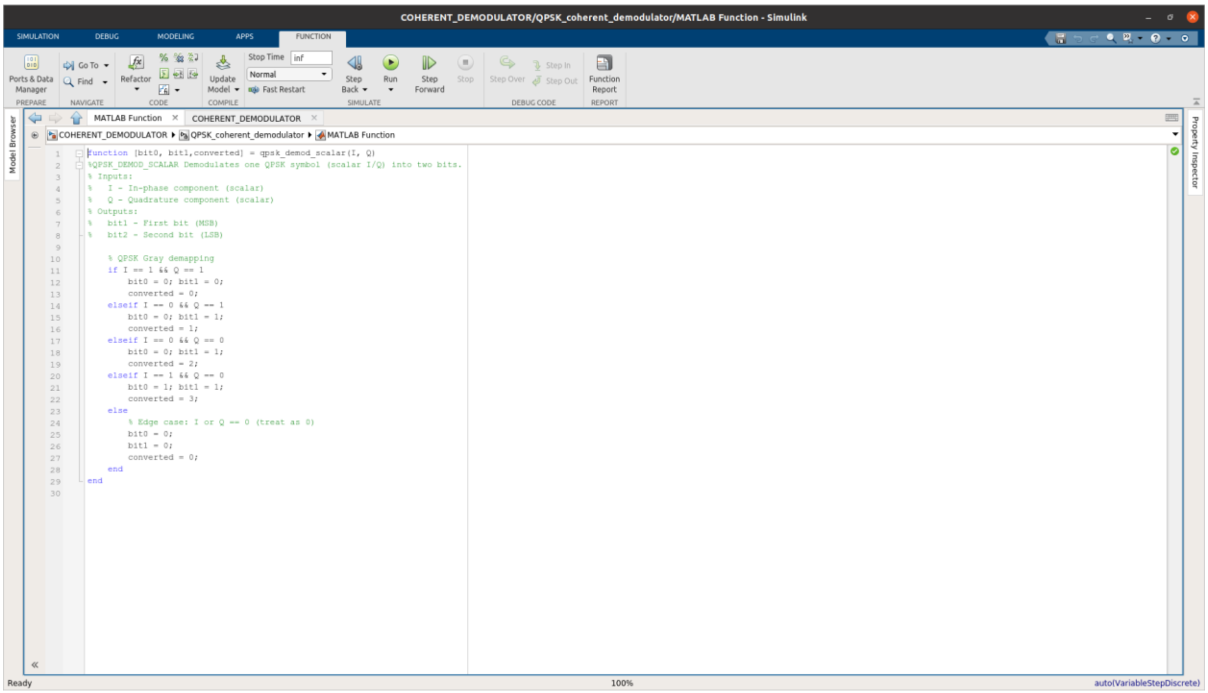


A slider gain has been used to adjust the threshold in the receiver.

Now let's have a look at the internal details of the receiver:

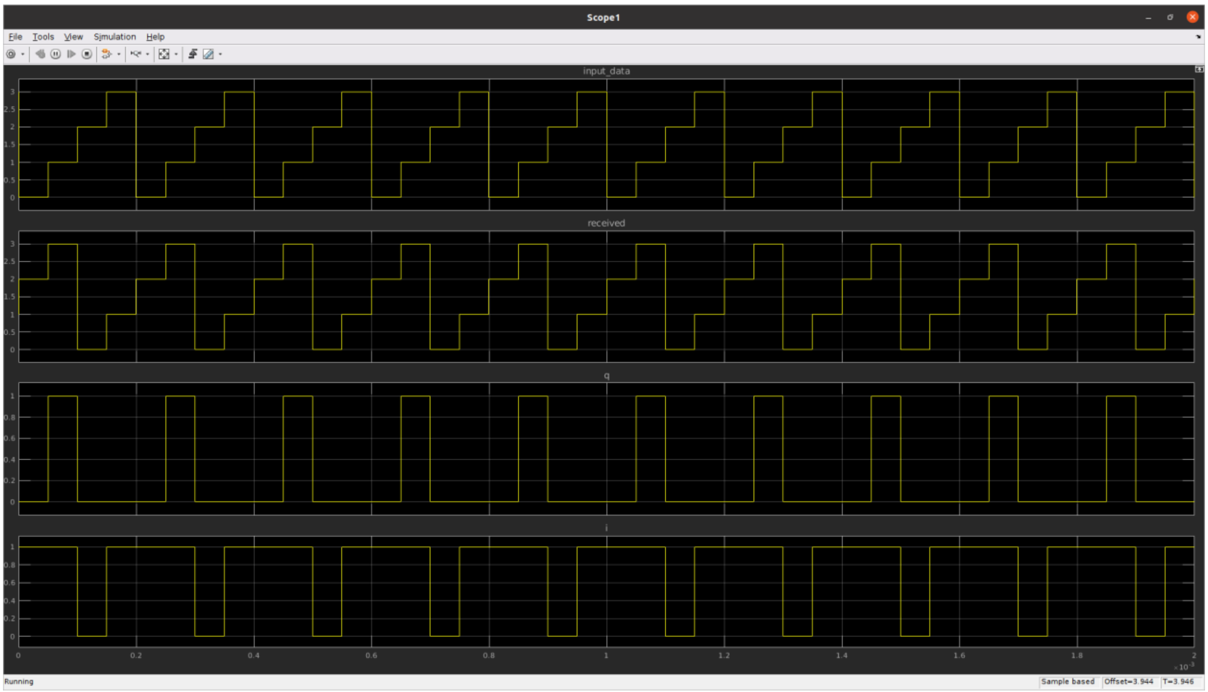






It closely resembles the classic coherent QPSK receiver topology.

By feeding the transmitter with a counter, the slider has been used to tune the receiver until the same incrementing counting sequence has been received.



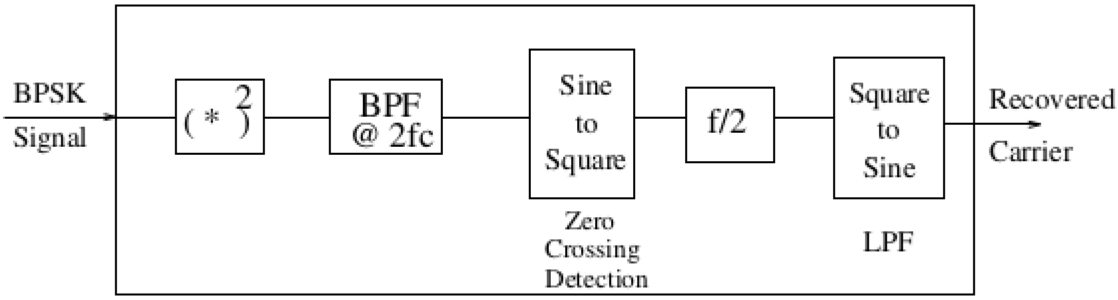
CARRIER RECOVERY

Coherent detection requires the receiver to have a local oscillator that is phase-locked to the carrier signal of the transmitter.

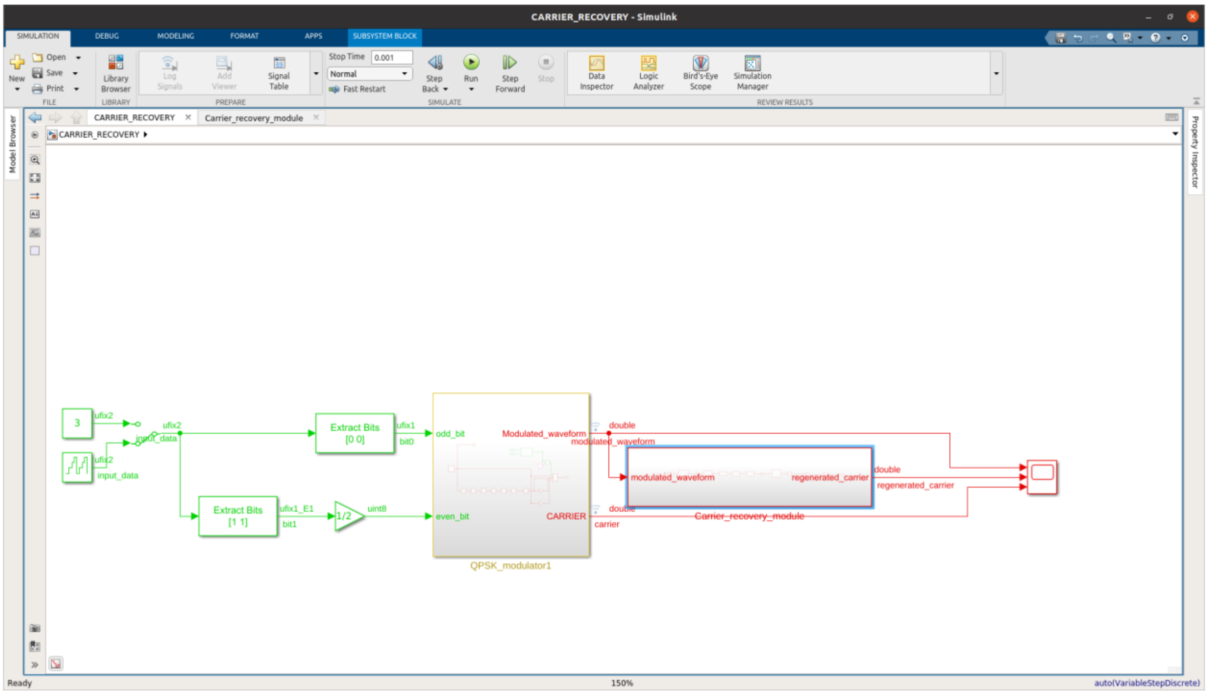
Any phase error between the received signal and the local oscillator degrades performance (increases bit error rate).

Carrier recovery techniques estimate and track the phase (and often frequency) of the received carrier signal to allow correct demodulation. Once recovered, this carrier is used to mix down the signal and extract the I/Q components for symbol decoding.

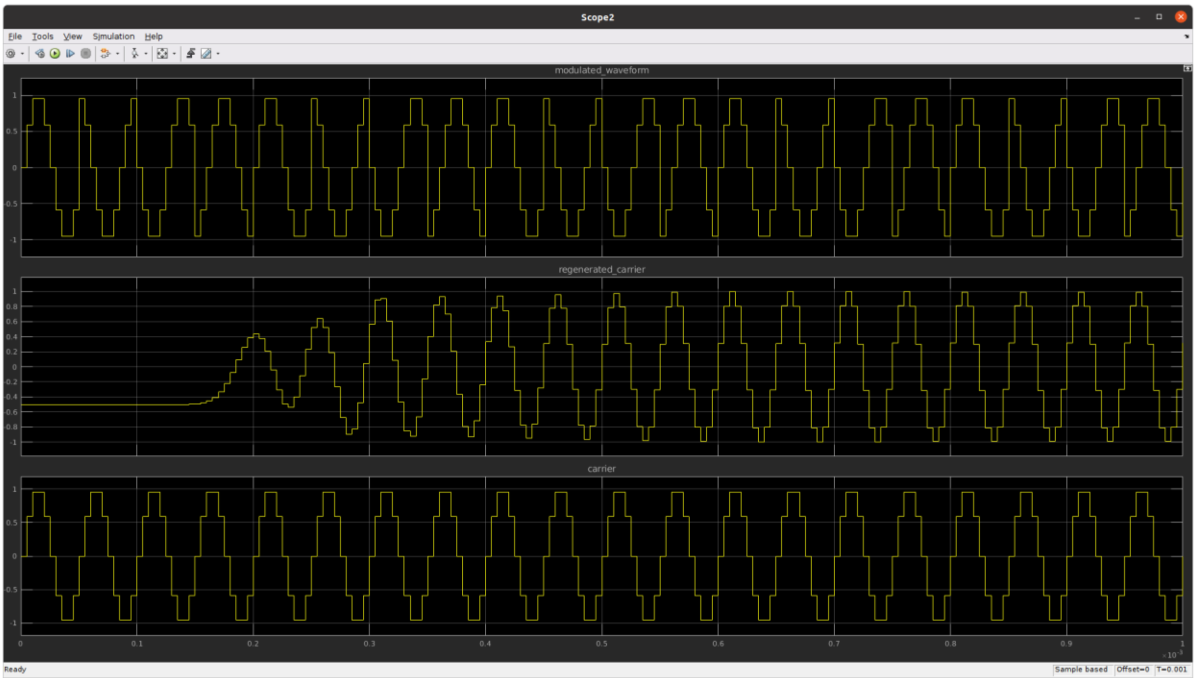
The block diagram of the carrier recovery mechanism to be used follows:



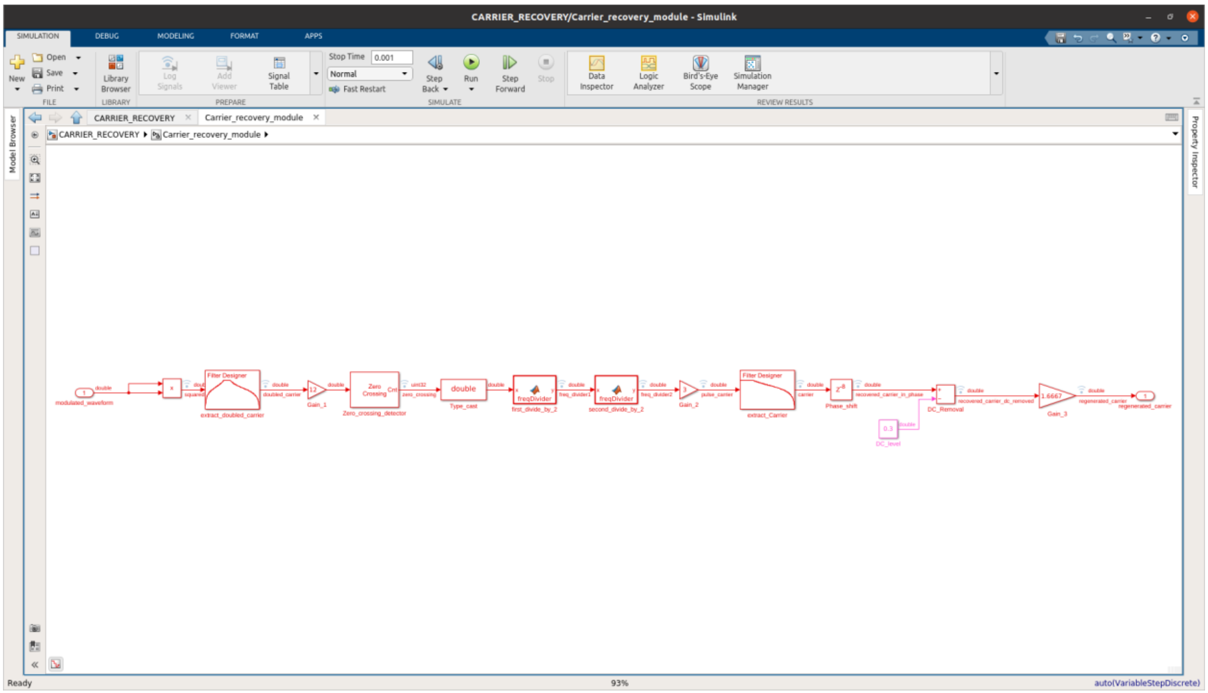
A model has been built to implement it:



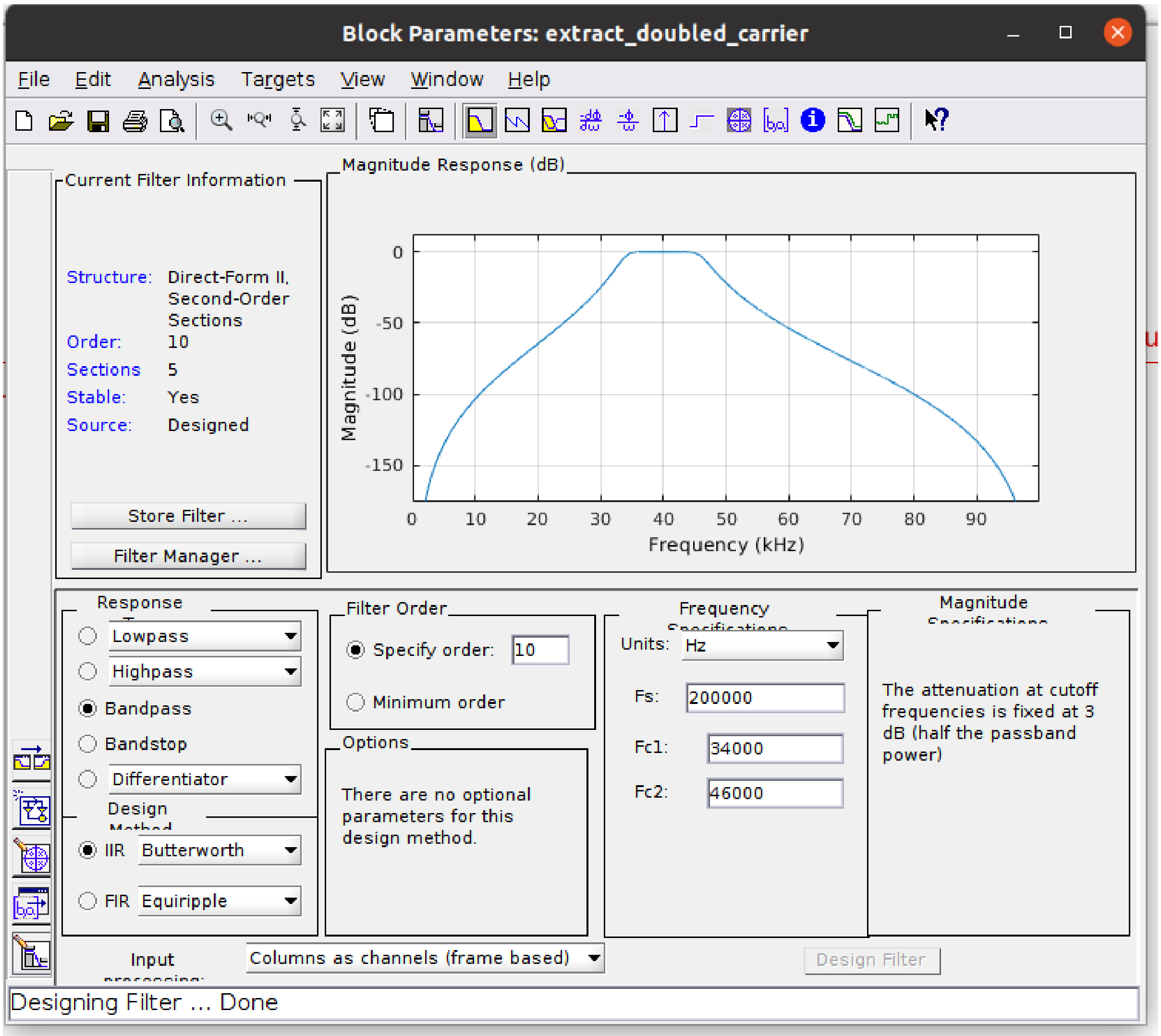
A QPSK modulator is feeding the carrier recovery, on an oscilloscope the carrier, the recovered carrier and the modulated QPSK stream are visualized for a comparison:



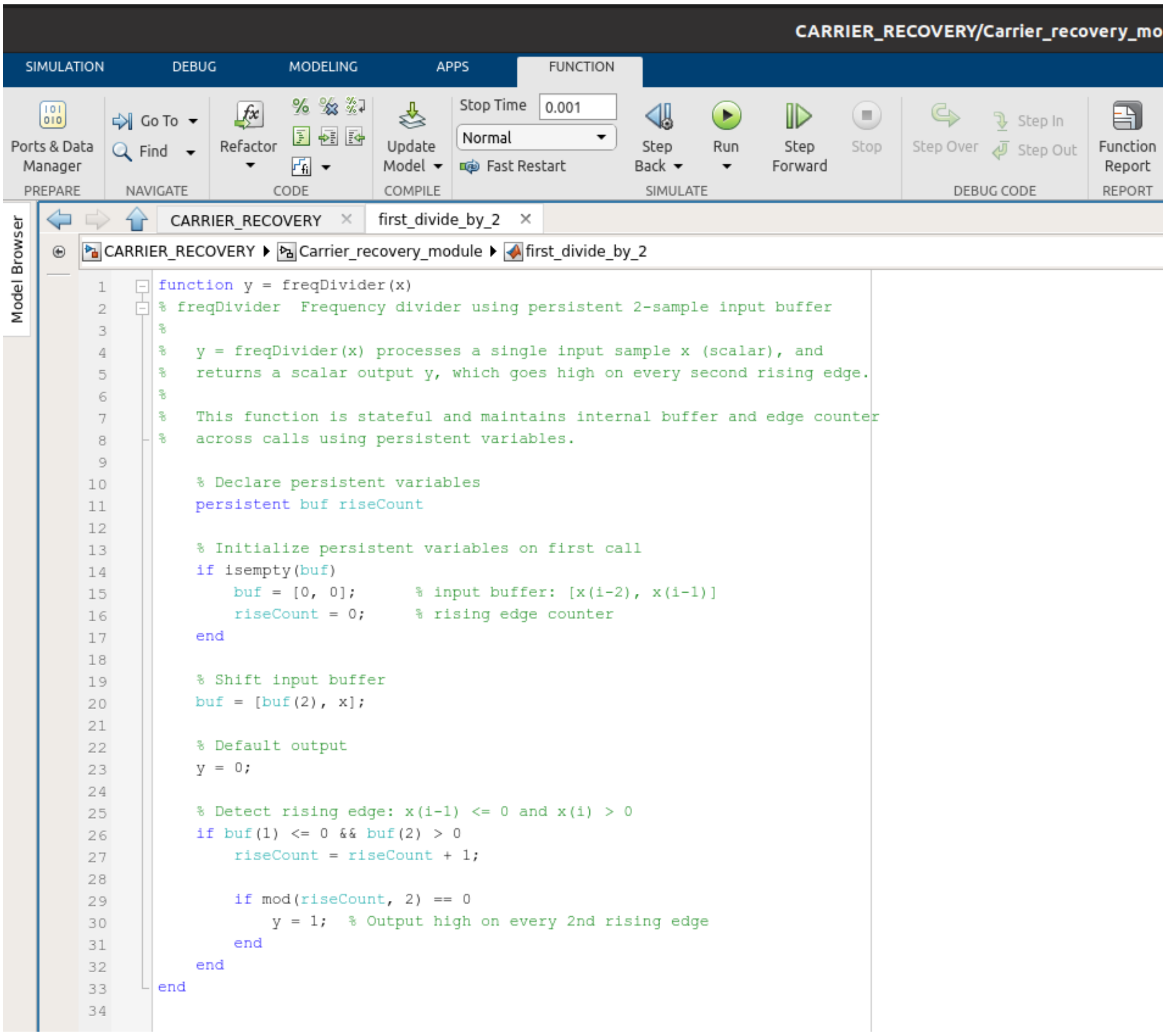
Details of the carrier recovery module follow:



The band pass filter to extract the squared carrier has been configured as follows:

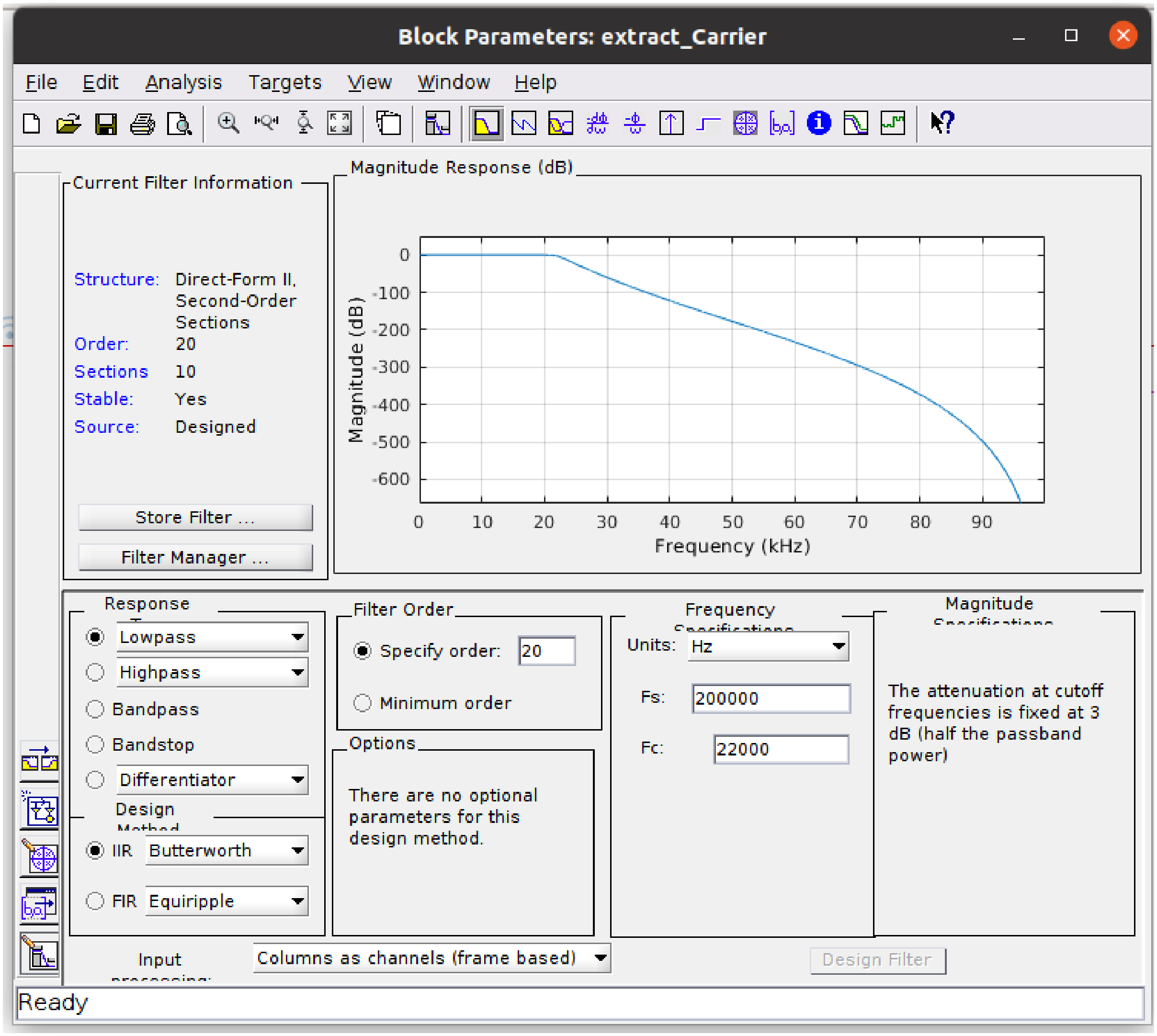


A couple of Matlab blocks in cascade, have been used to divide by 4 the zero crossing pulses:




```
function y = freqDivider(x)
% freqDivider  Frequency divider using persistent 2-sample input buffer
%
%   y = freqDivider(x) processes a single input sample x (scalar), and
%   returns a scalar output y, which goes high on every second rising edge.
%
%   This function is stateful and maintains internal buffer and edge counter
%   across calls using persistent variables.

    % Declare persistent variables
    persistent buf riseCount

    % Initialize persistent variables on first call
    if isempty(buf)
        buf = [0, 0];       % input buffer: [x(i-2), x(i-1)]
        riseCount = 0;      % rising edge counter
    end

    % Shift input buffer
    buf = [buf(2), x];

    % Default output
    y = 0;

    % Detect rising edge: x(i-1) <= 0 and x(i) > 0
    if buf(1) <= 0 && buf(2) > 0
        riseCount = riseCount + 1;

        if mod(riseCount, 2) == 0
            y = 1;  % Output high on every 2nd rising edge
        end
    end
end
```

The low pass filter used to extract the carrier has the following configuration parameters:



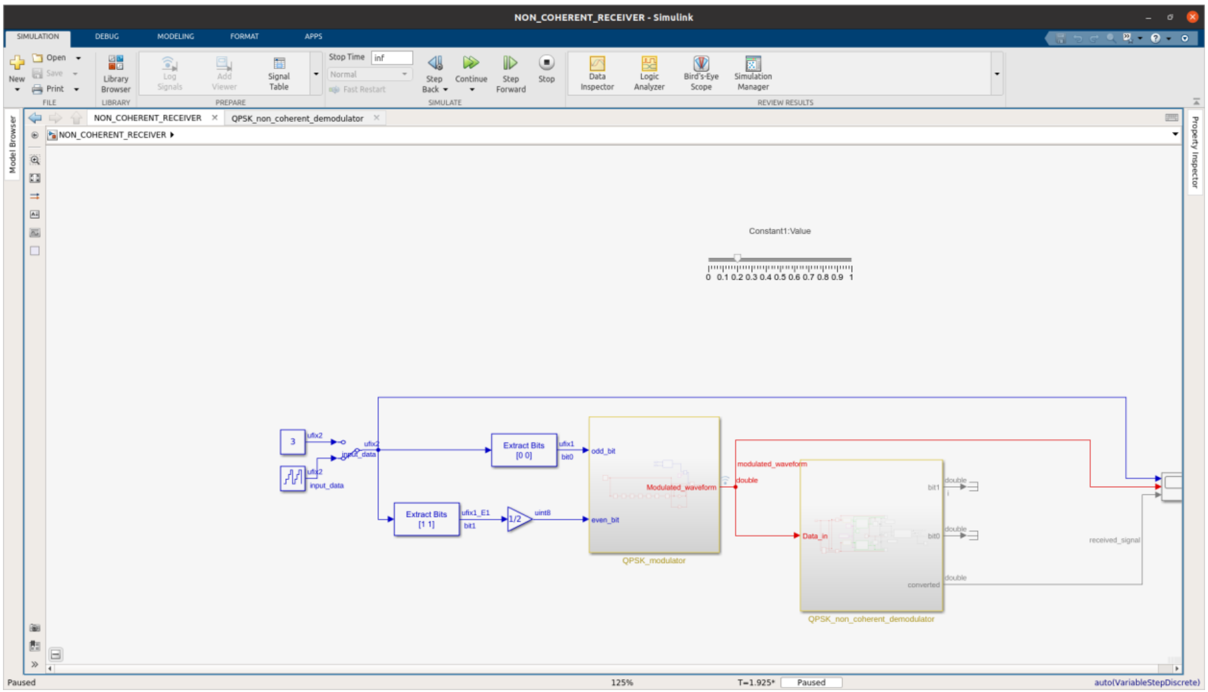
The next step now, is to build a non-coherent QPSK demodulator.

NON-COHERENT RECEIVER

A non-coherent QPSK (Quadrature Phase-Shift Keying) demodulator doesn't require knowledge of the carrier phase to extract the data. Instead, it relies on observing the relative phase differences between consecutive symbols to determine the transmitted data. This simplifies the receiver design by eliminating the need for complex carrier recovery circuits. However, non-coherent QPSK generally offers a lower bit error rate (BER) performance compared to coherent QPSK.

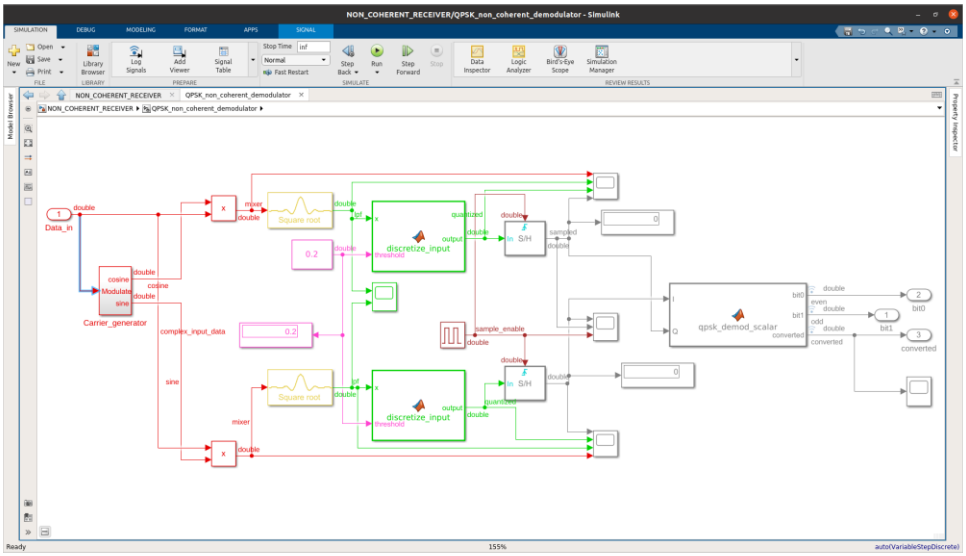
The non coherent receiver is like the coherent one, but with the carrier recovery module in place of the oscillator, to create the sine and cosine waveforms.

The following block diagram has been prepared:

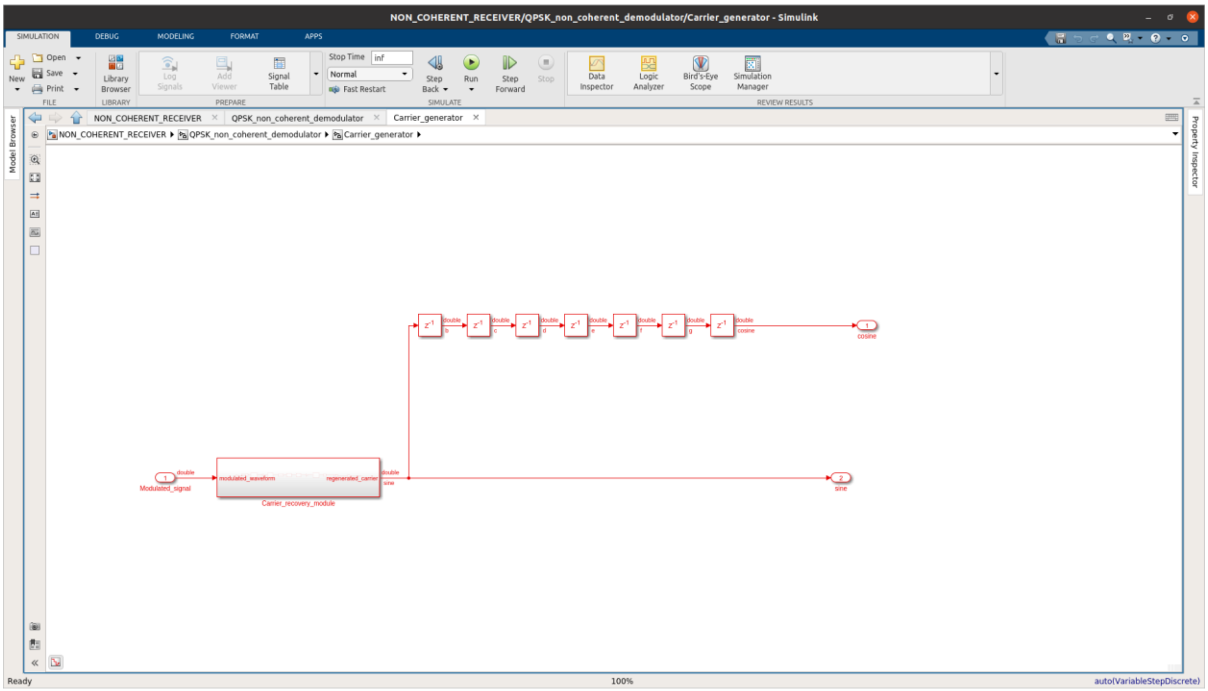


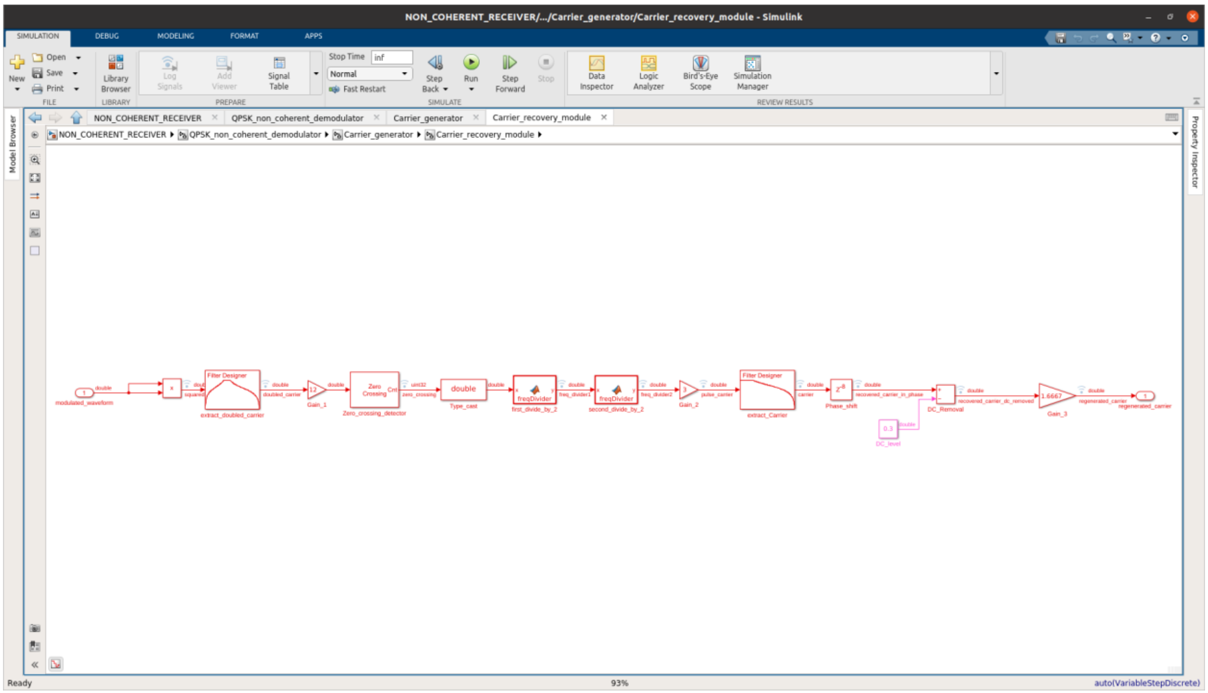
A counter is fed to a modulator, that in turn feeds the non-coherent demodulator.

Details about the non-coherent demodulator follow:



It is mostly identical to the coherent one, but the carrier recovery block has been used to generate the sine and cosine waveforms:


NON_COHERENT_RECEIVER/QPSK_non_coherent_demodulator/Carrier_generator - Simulink
NON_COHERENT_RECEIVER
QPSK_non_coherent_demodulator
Carrier_generator
Modulated_signal
modulated_waveform
regenerated_carrier
Carrier_recovery_module
cosine
sine
Ready
100%
auto(VariableStepDiscrete)


NON_COHERENT_RECEIVER/.../Carrier_generator/Carrier_recovery_module - Simulink
NON_COHERENT_RECEIVER
QPSK_non_coherent_demodulator
Carrier_generator
Carrier_recovery_module
modulated_waveform
Filter Designer
extract_doubled_carrier
Gain_1
Zero Crossing
Zero_crossing_detector
Type_cast
freqDivider
first_divide_by_2
second_divide_by_2
Gain_2
extract_Carrier
Phase_shift
DC_Removal
DC_level
Gain_3
1.6667
regenerated_carrier
Ready
93%
auto(VariableStepDiscrete)