MCC150-Implementation of Digital Signal Processing
Lab 4: Carrier Phase Recovery and Demodulation

Sundar Murugan Ramaswamy, Chalmers University of Technology.

I. Introduction

The RF transceiver which was built in lab 3 is expanded in order to add support for Carrier Phase Recovery and Demodulation. The DSP architecture is built using Simulink and hardware is generated in VHDL. The model is built and then examined and synthesized using Quartus prime.

II. PRE-LAB TASKS

1. The output data is found to match with the input because when we compare *tx.data* and *rx.data*, the bitstream is found to be the same with zero errors. The *xor* operator performs the *or* operation where if any difference in the corresponding bitstream in the transmitted and received data, the *error count* becomes 1. The *sum* operation adds the number of differences in the bitstream of *tx.data* and *rx.data* after performing the *xor* operation. The code is explained alongside with the comments.

```
tx.data = randsrc(1000, 1, [0 1]); %random
bitstream
tx.symbols = exp(1i*pi.*tx.data);
%bitstream to complex
chan.phase = pi/2*rand() - pi/4 %phase
rx.symbols =tx.symbols.*exp(1i*chan.phase);
%R(t)=[I(t)+Q(t)]*e^-j(phase)

demodulated_bitstream =
(rx.symbols)*(cos(chan.phase)-
(1i*sin(chan.phase))); %R(t)(cos(phase)-
isin(phase))
for i=1:length(demodulated_bitstream)
    if real((demodulated_bitstream(i))>0)
%conversion of complex into real values and
storing in the array
        rx.data(i)=0;
    else
        rx.data(i)=1;
    end
end
rx.data=rx.data';
error_count= sum(xor(tx.data, rx.data));
%logical operator to find if there is any
change in bitstream values
```

2. The implementation still works if the phase offset is emulated as a random walk. This can be confirmed by comparing the corresponding bit values of *tx.data* and *rx.data*.

```
tx.data = randsrc(1000, 1, [0 1]);%random
bitstream
tx.symbols = exp(1i*pi.*tx.data);%bitstream to
complex
chan.phase = cumsum(randn(size(tx.data))*1e-
2);  %phase (random walk)
rx.symbols = tx.symbols.*exp(1i*chan.phase);
%R(t)=[I(t)+Q(t)]*e^-j(phase)
phase = ((1/2) *angle(rx.symbols.^2));

demodulated_bitstream =
(rx.symbols).*(cos(phase)-
(1i*sin(phase)));%R(t)(cos(phase)-isin(phase))
for i=1:length(demodulated_bitstream)
    if real((demodulated_bitstream(i))>0)
%conversion of complex into real and storing
in the array
        rx.data(i)=0;
    else
        rx.data(i)=1;
    end
end

rx.data=rx.data';
error_count= sum(xor(tx.data, rx.data));
%logical operator to find if there is any
change in bitstream values.
```

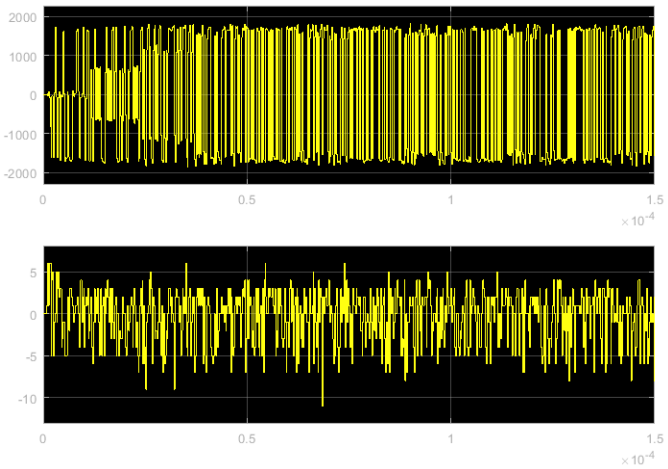
3. The MATLAB CPR implementation can be transferred to a DSP builder design by making use of the CORDIC block. It is based on rotating the phase of a complex number. This block also minimizes the resource utilization by reducing the number of gates required, eliminates the need of a hardware multiplier and also calculates the trigonometric equations thereby eliminating the *trig* block.

4. Assuming a random walk with phase offset values greater than $\pi/2$ and lesser than $-\pi/2$, the transmitted input and the demodulated received output bitstream are found to differ (which was verified by comparing the bit values in MATLAB). The *error count* was also found to increase.

III. CARRIER PHASE RECOVERY

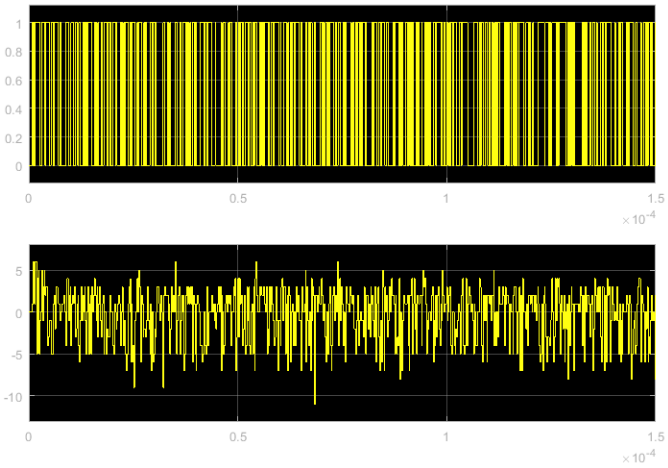
There is always a residual phase offset that is needed to be eliminated before the signal is being demodulated. Phase offset appears constant previously in our Simulink model. The phase offset will be fluctuating due to phase noise which severely impacts the quality of the received signals. A CPR unit is added to the DSP builder. Once the phase offset is known, it can be eliminated by rotating the input symbol by the phase value in the opposite direction($-\Theta$). This can be performed by using the CORDIC block. Two *cordic* blocks are made use of. The first *cordic* block is inputted with squaring and adding the real and imaginary parts of the signal (*subtract* block is used as $i^2=-1$).

This first *cordic* block is set to perform in mode 2 where it finds the angle required to make the rotation. Then it is passed through a *logical right shift* block to divide it by 2 eliminating the use of a *mult* block thus reducing utilization of resources. This output is then passed into the second *cordic* block which works on mode 1 where the input vector is rotated by the specified angle. The phase is recovered which can be inferred as the Q signal magnitude is almost zero. The output of the CPR is shown below.



IV. DEMODULATION

A demodulation block is created to the output of the receiver which outputs the demodulated bitstream. The I signal output of the CPR block is passed into the *cmpLT*(compare less than) block which produces a Boolean output for the specified condition. In this case, it outputs true (1) if the input is less than 0 and 0 if the condition is not satisfied (as given as default value in the *select* block if the condition dissatisfies). The demodulated output is shown below.



V. SYNTHESIS AND VERIFICATION IN QUARTUS PRIME

The HDL code of the transmitter which is generated using the DSP builder is loaded into Quartus prime for hardware synthesis. Symbol files are created for the transmitter and then the updated transceiver block is connected to the AD9361 controller. Then the design is compiled, and the compilation report is studied for design utilization, timing results and other design statistics.

VI. POST LAB HOME ASSIGNMENT

1. The functionality of *CPR* and *demodulator* blocks and their subsystems are explained in sections 3 and 4 respectively. The CPR design can be verified by checking the magnitude of the Q signal which must be near zero. The demodulator block design can be verified by comparing the transmitted and the demodulated waveforms by viewing their corresponding scopes.

2. The captured signals are shown in the image below.

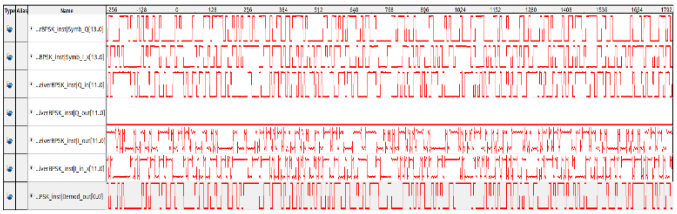
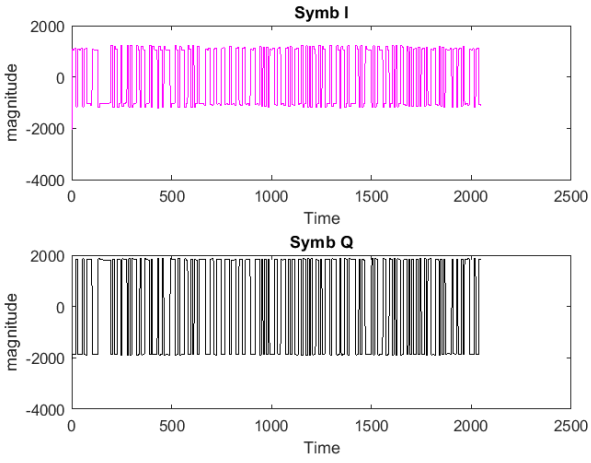
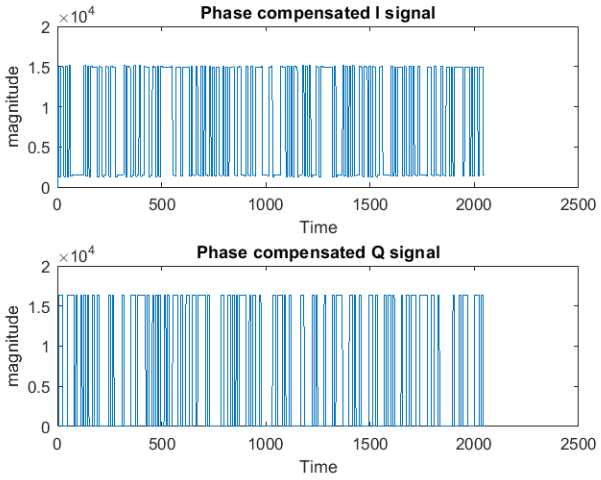


Figure 1 Signals captured in Quartus prime

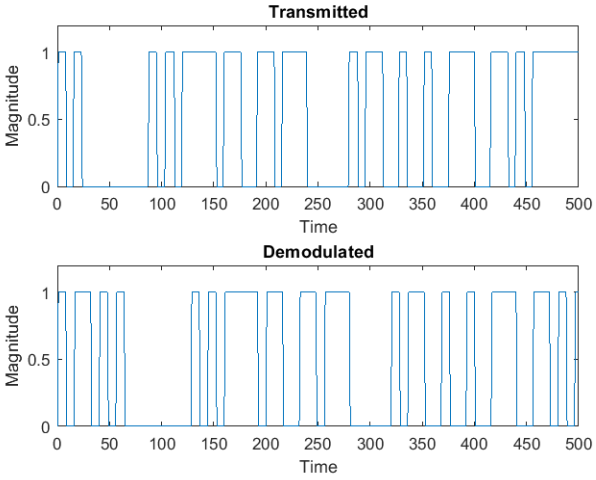
The signals captured are I_in, I_out, Q_in, Q_out, Demodulated output, Symb_I and Symb_Q. I_in, I_out, Q_in, Q_out represents the In-phase and quadrature signals of the data (real and complex parts) respectively. The Symb_I and Symb_Q are the outputs from the decimation block. The Demod_out represents the demodulated output from the receiver.

3. The Symb_I and Symb_Q signals captured in lab 3 are compared with phase compensated signals in lab 4. The respective signals are attached below. It can be found that there is a rapid increase in the magnitude value of the phase compensated signals which is due to the gain added when the signals pass through the *cordic* block. Also we can see that the phase compensated signals are square enough compared to the lab 3 outputs.





4. The transmitted and received data stream as a function of time is added below. The data is been recovered correctly but with a phase offset which can be confirmed by viewing both the transmitted and demodulated waves. The latency is found to be 41 unit time.



5. In a real communication system, the limitation when the phase values crosses the boundaries (which causes the capture to be incorrect) can be overcome by using multiple antennas.

6. The *cordic* block was made use of to reduce the utilization of resources which was explained in section 3. The input *p* in the *cordic* block represents the phase angle. The accuracy will be a tradeoff for the size by specifying less number of bits which helps to reduce the number of stages inside the *cordic* block. Thus the efficiency reduces with the number of bits linearly. And as the value of SNR decreases, the signal quality reduces which makes it difficult to capture at the receiver end and demodulate despite the increased phase noise.