MCC150 - Implementation of Digital Signal Processing Systems

Lab 4: Carrier Phase Recovery and Demodulation

Sining An, Erik Börjeson
Per Larsson-Edefors, Zhongxia Simon He

Version 1.1 - March 24, 2021

Contents

1 Introduction

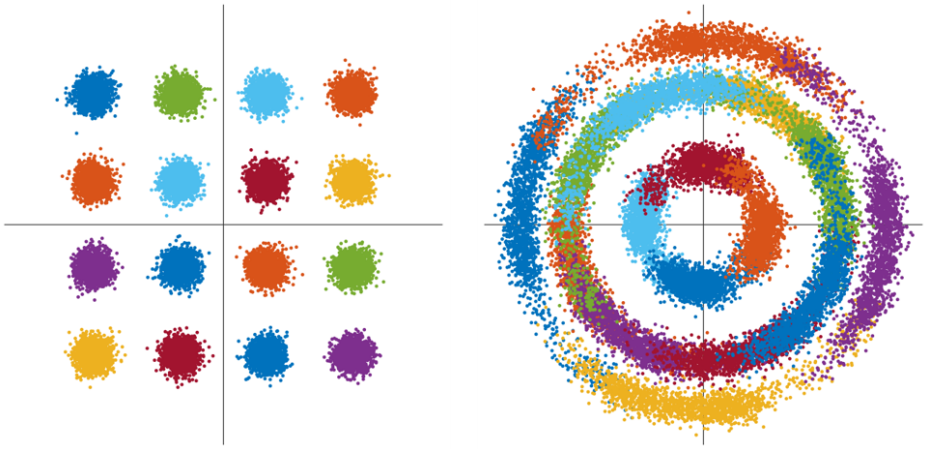
After the first three labs, there are two important receiver parts left to add to your design: a carrier phase recovery (CPR) block, and a demodulator. In this lab session, you will create the implementation in DSP Builder yourself, based on the methods described in the pre-lab reading material. When developing DSP implementations, you will need to take both the correctness of the output and the hardware utilization into account. Some operations are very costly to implement in hardware, and should be avoided, while others can be simplified or approximated. Keep this in mind when creating your design.

2 Pre-Lab Preparation

Read the following section and finish the pre-lab tasks, before starting to work on the lab exercises.

2.1 Carrier Phase Recovery

When you study the received signal after processing in your current receiver design, there is a residual phase offset that needs to be removed before demodulating the signal. You have previously modelled this offset as a constant in your Simulink project, and when transmitting data using the hardware setup the sample window is very short, which makes the phase offset appear constant. In a real system, however, the offset will vary randomly with time due to phase noise. The phase noise is caused by random fluctuations of the phase of the carrier and local oscillators and can severely impact the quality of the received signal, as illustrated by Fig. 1, which shows a 16QAM signal with and without phase noise.



(a) **(b)**

Figure 1: *10,000 received 16QAM symbols, (a) without phase noise , and (b) with phase noise.*

The simplest way to model phase noise is as a random walk

$$\theta_k = \theta_{k-1} + \Delta_k, \tag{1}$$

where θ_k is the phase of the kth sample and Δ_k is a random Gaussian variable with zero mean and a variance describing how fast the phase changes.

In your BPSK receiver, the phase offset, θ can be estimated from the received and decimated symbols. But since the data modulates the phase, as shown in Fig. 2, this modulation must first be removed to circumvent the 180-degree phase ambiguity.

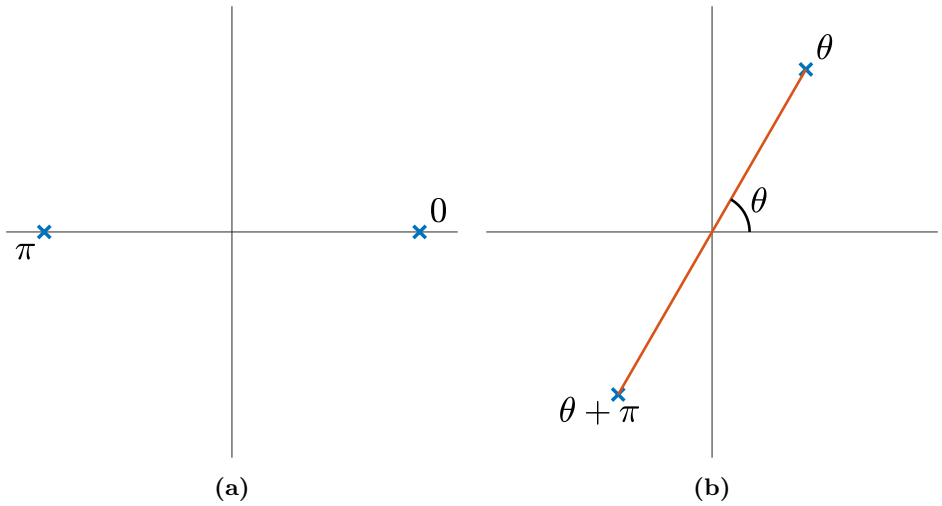


Figure 2: *BPSK constellation points, (a) without phase offset, and (b) with phase offset, θ.*

If R is the received symbol consisting of the received I/Q signals, I_R and Q_R, it can be rewritten as a a function of the transmitted I/Q values, I_T and Q_T, and the phase offset as

$$\begin{aligned} R(t) &= I_R(t) + Q_R(t) \\ &= \big(I_T(t) + Q_T(t)\big)e^{j\theta}, \end{aligned}$$

if all other signal impairments, e.g. additive white Gaussian noise, is ignored. If φ is the modulated phase, i.e. 0 or π, R can be written as

$$\begin{aligned} R(t) &= Ae^{j\varphi(t)}e^{j\theta}, \varphi = 0 \text{ or } \pi \\ &= Ae^{j(\varphi(t)+\theta)}, \end{aligned}$$

where A is the amplitude. The modulation can be removed by squaring the received signal

$$\begin{aligned} R^2(t) &= A^2e^{j(2\varphi(t)+2\theta)}, 2\varphi = 0 \text{ or } 2\pi \\ &= A^2e^{2\theta}. \end{aligned}$$

Since $e^{j0} = e^{j2\pi} = 1$, the result is a signal with a phase twice that of the phase offset, which means that

$$\theta(t) = \frac{1}{2}\angle R^2(t).$$

Once the phase offset is known, it can be removed by rotating the input symbol with $-\theta$. This rotation can be performed in many different ways, e.g. using CORDIC [1, Ch. 2][2, Ch. 6] or complex multiplication as

$$\begin{aligned} \hat{R}(t) &= R(t)e^{-i\theta} \\ &= R(t)\big(\cos(-\theta) + i\sin(-\theta)\big) \\ &= R(t)\big(\cos(\theta) - i\sin(\theta)\big) \\ &= \big(I_R + iQ_R\big)\big(\cos(\theta) - i\sin(\theta)\big), \end{aligned}$$

where $\hat{R}$ is the phase-compensated received symbol. The sine and cosine values can be stored in a look-up table, to reduce hardware utilization. If the phase offset was removed successfully, the Q component of $\hat{R}$ should be close to zero.

2.2 Demodulation

Demodulation of the received BPSK signals can be performed after phase recovery. If $\hat{R}(t) = \hat{I}_R(t) + i\hat{Q}_R(t)$, most of the energy should now be present in the I portion of the signal. It is therefore enough to check if $\hat{I}_R$ is greater than 0, for a bit value of 0. If it is less than 0, the output bit should be a 1.

2.3 Pre-lab tasks

- Create a MATLAB script that performs phase recovery and demodulation of 1000 BPSK symbols, with an added constant phase offset, generated using the MATLAB code below. You can assume that the phase offset is between $-\pi/4$ and $\pi/4$. Check that the output data matches the input. If you store the demodulated bitstream in `rx.data`, you can count the number of errors by doing `sum(xor(tx.data, rx.data))`. Show your MATLAB code in your report and explain how it works.

```
tx.data    = randsrc(1000, 1, [0 1]);
tx.symbols = exp(1i*pi.*tx.data);
chan.phase = pi/2*rand() - pi/4
rx.symbols = tx.symbols.*exp(1i*chan.phase)
```

- Does your implementation still work when the phase offset is emulated as a random walk, shown in the MATLAB code below?

```
tx.data    = randsrc(1000, 1, [0 1]);
tx.symbols = exp(1i*pi.*tx.data);
chan.phase = cumsum(randn(size(tx.data))*1e-2)
rx.symbols = tx.symbols.*exp(1i*chan.phase);
```

- How can you transfer the MATLAB CPR implementation to a DSP Builder design? Draw a block diagram of your suggested solution and explain how it works. How can you minimize the resource utilization?

- Assuming that the random walk starts a $\theta = 0$, what happens if the phase offset increases to $> \pi/2$ or decreases to $< -\pi/2$.

3 Lab

The following sections describe the exercises that are to be performed during this lab session.

3.1 Exercise 1 - Carrier Phase Recovery

- Add a CPR unit to the DSP Builder project from the last lab, based on your results from the the pre-lab assignments. The unit should take the `I_out`, `Q_out`, `qv` and `qc` signals from the *Decimation* subsystem as its input. It should output phase-compensated I and Q signals, and the mandatory `qv` and `qc` signals. You can assume that the phase offset is $-\pi/2 < \theta < \pi/2$. Be aware of the resource utilization when designing your system, and try to keep it as low as possible. Don't use the *Math*, *Sqrt* or *Trig* blocks if you can find more efficient ways of implementing your operations. Verify that your system works by simulation in DSP Builder.

- Connect the `Symb_I` and `Symb_Q` ports to the output of your CPR unit.

3.2 Exercise 2 - Demodulation

- Create a demodulation block in DSP Builder that outputs the demodulated bitstream. The input to the demodulator should be taken from the I/Q outputs of the CPR and its output should be connected to a new data output port. Verify that your system works by simulation in DSP Builder.

3.3 Exercise 3 - Synthesis and Verification

In this exercise you will upload your design to the FPGA and verify that your system works by using the SignalTap Logic Analyzer. For the items marked with a red bullet (•), you will need to have access to a computer connected to the DE10-Standard board. We have provided two servers, with connected hardware, which you can access remotely. See Canvas for instructions on how to book your time slot and how to connect to these machines.

- Import your finished DSP Builder design into the Quartus Prime project provided on Canvas and start a synthesis run.
- Add a SignalTap instance to your project and set it up to capture the data output from the *PRBSGenerator*, the `Symb_I` and `Symb_Q` ports, and the demodulated bitstream. Use the `clk_40M` from the `AD9361` as the SignalTap clock. Compile the design when you have completed the SignalTap settings.
- Upload the design to the FPGA and capture data from a SignalTap run. Take a screenshot of the SignalTap window and save all captured data as a `.csv` file; you will need these files for the home assignment.

4 Post-Lab Home Assignment

A report describing your lab work should be handed in on Canvas before 23:59 on Sunday, May 16. The report should be maximum 4 pages long, and should contain answers to the pre-lab tasks, descriptions of the blocks designed in DSP Builder and their functionality, and solutions to the home-assignment questions below, including the resulting figures. When handing in your report, you will need to submit both a `.pdf` version of your report and a `.zip` archive containing your DSP Builder project.

- Show how you realize the CPR and demodulation in DSP Builder and explain the functionality of all the blocks and subsystems that you have designed. Describe how you verify that your designs are working.
- Show a screenshot of the SignalTap window with captured signals mentioned in Exercise 3. Explain what each signal represents.
- Plot the phase compensated I/Q signals exported from SignalTap as a function of time. Compare them with the `Symb_I` and `Symb_Q` signals captured in the last lab. What is the main difference?
- Plot the transmitted and demodulated data stream, exported from SignalTap, as a function of time and compare them. Did you recover the data correctly? What is the total latency of your system?
- The CPR implementation assumes that the phase offset is between $-\pi/2$ and $\pi/2$. From the pre-lab tasks you know what happens if the phase crosses these boundaries. How can these situations be handled in a real communication system?

- Can you further decrease the resource utilization by reducing the number of bits used in the carrier phase recovery, e.g. for the phase angle? Does this reduction affect the quality of the output in terms of number of erroneously decoded bits? What happens if you decrease the SNR?

References

[1] D. Markovic and R. W. Brodersen, *DSP Architecture Design Essentials*. New York: Springer, 2012.

[2] U. Meyer-Baese, *Digital Signal Processing with Field Programmable Gate Arrays*, 4th ed. Berlin, Heidelberg: Springer, 2014.