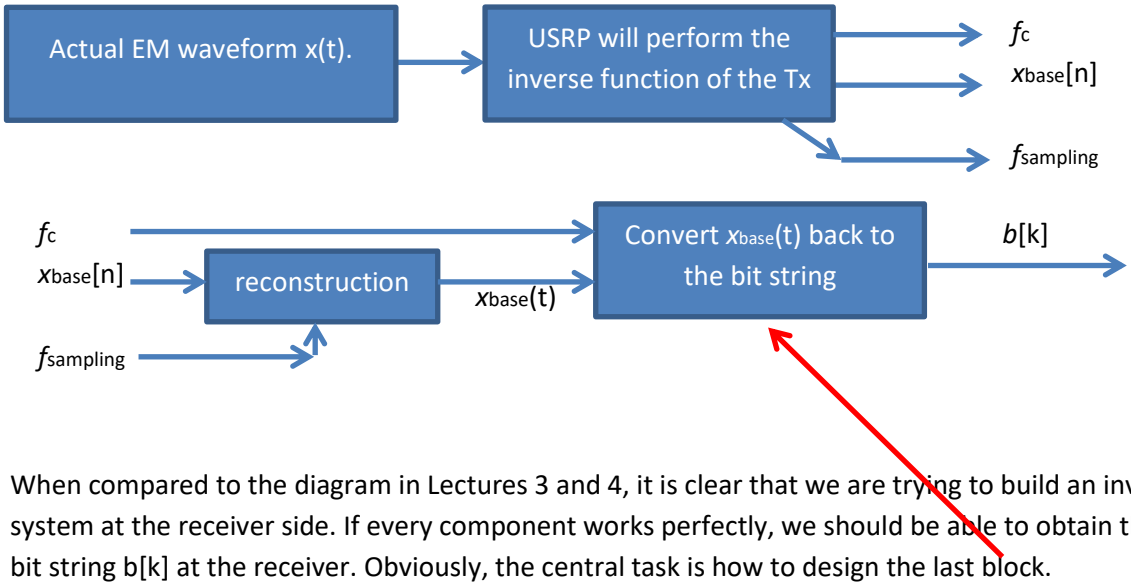
Lecture 5 / Homework 5 for The Software Defined Radio (SDR) of Purdue VIP

Prepared by Prof. Chih-Chun Wang

In this lecture, we will talk about the receiver side of a digital communication system. Basically, we are focusing on the following inverse system



When compared to the diagram in Lectures 3 and 4, it is clear that we are trying to build an inverse system at the receiver side. If every component works perfectly, we should be able to obtain the original bit string b[k] at the receiver. Obviously, the central task is how to design the last block.

Before, we consider how to design the last block, there is one more challenge we need to address. Recall from Tasks 1 and 2 in Lecture 2, even with the same signal x(t), there may be multiple ways of deriving f_c and $x_{base}(t)$. As a result, we would like to make sure that the f_c used at the receiver is the same as the f_c used at the transmitter so that the $x_{base}(t)$ sent by the transmitter is the same as the $x_{base}(t)$ heard by the receiver. This turns out to be a non-trivial task since during propagation, the frequency of the analog waveform will change due to the so-called Dopper effect. Therefore, even when we design f_c=900MHz at the transmitter, the received signal may have fc=901MHz or fc=899MHz. The problem is that we do not know what the actual fc will be as it changes from time to time. Therefore, we need to "learn" what is the right fc value to use at the receiver.

The way we learn the right fc value is by the so-called Costas loop. The following is a diagram of the Costas loop, which is copied from the following website http://michaelgellis.tripod.com/mixerscom.html by Michael Ellis.

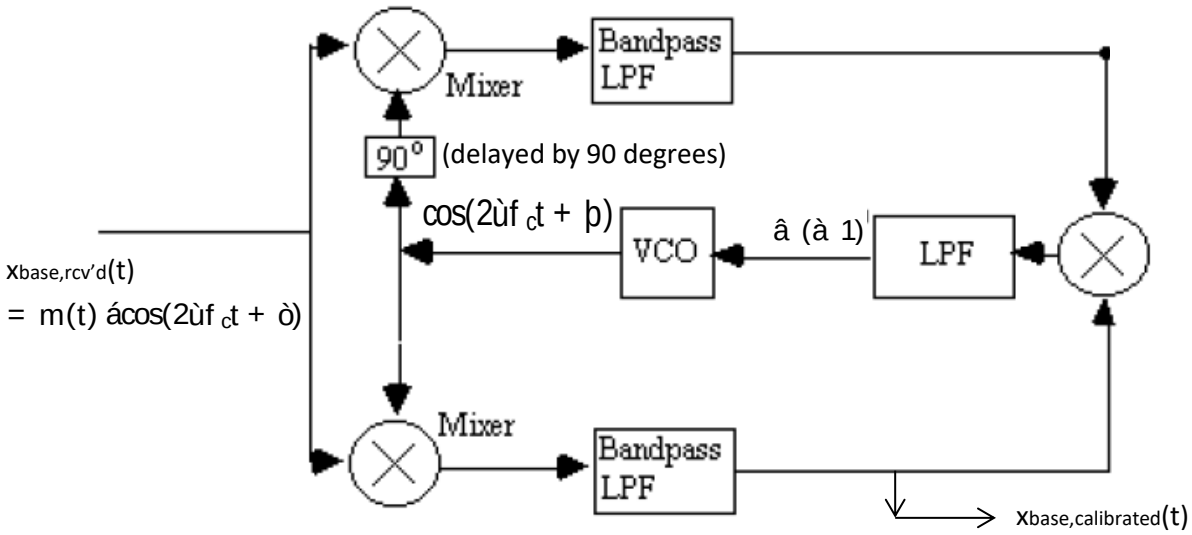


Figure 4. Costas loop for BPSK demodulation

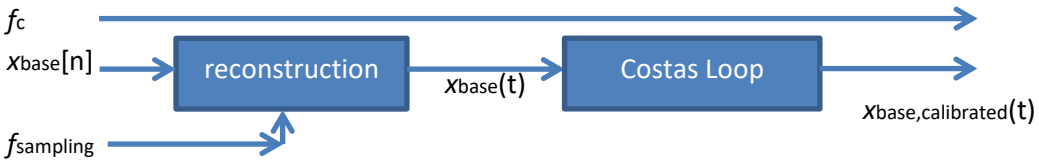
That is, when we receive $x_{base,rcv'd}(t)$, which is computed based on assuming a wrong f_c value, the above Costas loop can calibrate the signal and generate new $x_{base,calibrated}(t)$ corresponding to the correct f_c value. The VCO (Voltage Controlled Oscillator) used in the Costas loop does the following. When the input of the VCO is a positive value (voltage), then the oscillation of the VCO will speed up. When the input of the VCO is a negative value, then the oscillation of the VCO will speed up. When the input of the VCO is zero, then the oscillation of the VCO will be of a constant frequency. That is, the output will be a sinusoidal signal.

Disclaimer: the above Costas loop works only when there is no imaginary part in the $x_{base}(t)$ sent by the transmitter.

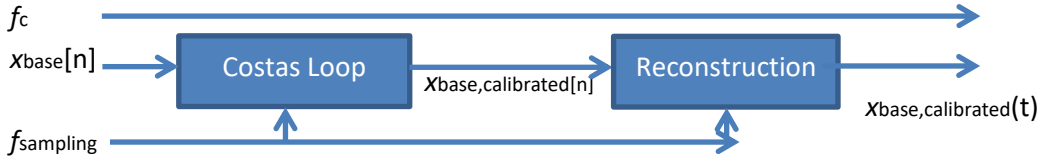
Task 1:

Discuss qualitatively, why the above Costas loop can speed-up and slow-down the VCO until the output of the VCO generates a sinusoidal signal that matches input $x_{base,rcv'd}(t)$. For discussion purpose, you can assume that $x_{base,rcv'd}(t) = x_{base,calibrated}(t)(\cos(2\pi f_c t + \pi/3))$ and assume that the VCO in the figure in the top of this page will increase ϕ if the input is positive; decrease ϕ is the input is negative; and keep the same ϕ if the input is 0. You should discuss why in the end the ϕ value will converge to $\pi/3$.

The above Costas Loop is very effective. However, it is described as a continuous-time system. That is, the Costas loop is inserted as part of the receiver design (taking $x_{base}(t)$ as input and outputting $x_{base,calibrated}(t)$):



Our next goal is to reproduce the Costas Loop as a discrete-time system.



A simple MATLAB code for performing the Costas Loop in the discrete-time domain is sent to you separately. Perform the following task.

Task 2:

Run the Costas Loop. You will see that the output signal is only ½ of the original signal m(t). Why do we have such a mismatch?

Task 3:

If you look at the program carefully, the MATLAB code is actually different from the original Figure in the last page. Draw the diagram corresponding to the MATLAB code. What is the difference between the MATLAB code and the diagram in the last page?

Task 4:

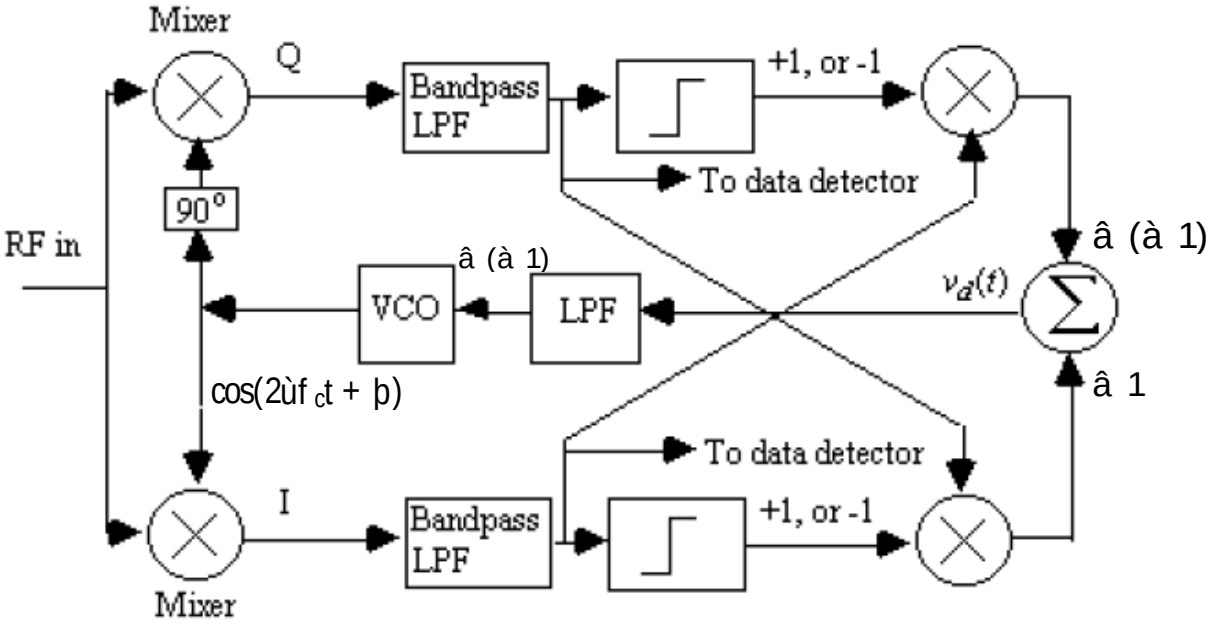
Change the parameter "`to_be_tracked_phi`" to π/8. Rerun the program. Notice that the program is still able to output the right $x_{base,calibrated}[n]$.

Task 5:

Change the parameter "`actual_fc`" to 4005. Rerun the program. Notice that the program is still able to output the right $x_{base,calibrated}[n]$. Explain intuitively why the Costas Loop can still "lock" and output the correct $x_{base,calibrated}[n]$. Hint: Look at the second output figure of your MATLAB code.

As mentioned earlier, the above program can only handle the case when there is no imaginary part in the $x_{base}(t)$ sent by the transmitter, or equivalently when there are no "sin" component in $x_{base}(t)$, see Equation 1 of Lecture 2.

The following diagram provides a Costas loop that is capable of tracing both the cosine and the sine component of the input.



Task 6:

Please modify the MATLAB program to implement the above Costas loop. Your input signal is included in the MATLAB program. (First search for the keyword "Task 6" in the .m file. You can then simply uncomment that part of the MATLAB code.) Is your new code able to successfully trace the I and Q components? *Note that for this program your input signals have to be rectangular waves of the same amplitude (but can be of arbitrary, different duty cycles).*