EE/TE 4385

Lab 7: PAM Receiver

Lab Report Due: Wendesday, 11/1/2006, 2PM

<u>PAM Receiver with Carrier Tracking: C6713 DSK Implementation Using CC</u>

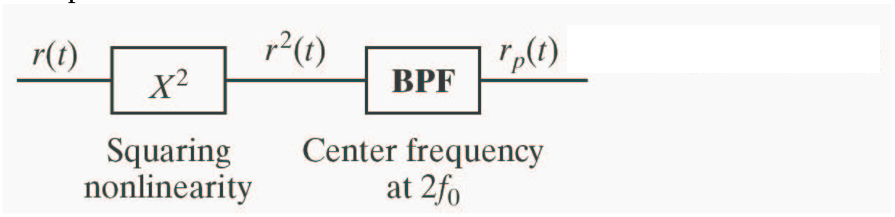
This is a follow up lab to ideal PAM receiver. This lab builds a carrier tracking function using the Phase Lock Loop (PLL). PLL scheme uses a squaring nonlinearity followed by a narrowband bandpass filter at 4KHz (since the carrier frequency is 2KHz).

- Use your project files from PAM receiver and transmitter labs. We will add additional functions to build the PLL to the PAM receiver program.
- Again, you need the help of your neighboring group to generate the PAM signal for you.
- Create a lab7 project directory under myprojects subdirectory in c:\ti folder.
- Copy your PAM receiver files into this directory. Make sure that your program still compiles without any problem. Keep the sampling rate 16 KHz.
- Ask you neighboring group to generate the PAM signal.
- Make sure your PAM receiver works as before.
- Your carrier tracking function needs two new FIR filters: BPF and LPF.
- <u>Implement all your filters using circular buffers</u>
- You can use fdatool in MATLAB to generate the coefficients of these filters.
    - BPF: bandpass, equiripple, specify order:48, fs=16KHz
        - fstop1:3.5KHz,fpass1:3.8KHz,fpass2:4.2KHz,fstop2:4.5KHz
    - LPF:lowpass,equiripple,specify order:48,fs=16KHz
        - Fpass:100Hz,Fstop:500Hz
- Export the filter coefficients into your receiver code. Define them as arrays with length of 49. You can use `bpf` and `lpf` as the array names.
- <u>Download and install The C67x FastRTS library, which is a collection of ptimized floating-point math functions for C programmers of the C67x DSP generation, from http://focus.ti.com/docs/toolsw/folders/print/sprc060.html</u>
    - <u>To use the FastRTS Library in Code Composer Studio:</u>
    - <u>First you have to specify to the linker to link the FastRTS Library into your application. You do this by adding the FastRTS Library, fastrts67x.lib file to your project then specify</u>
    - <u>For each of your C files that you plan to use an DSP Library kernel in, you must include the appropriate header file for each kernel. For instance, to use the kernel "sinsp()", you must #include <fastrts67x.h> in that file.</u>
- You still need to demodulate the received signal with both sin and cos waves. However, this time you need to use built-in sin and cosine functions to generate sine and cosine samples to include the phase and carrier offset estimates. Be careful with incrementing the carrier phase since it will increase without bound unless it is checked against 2π.

```
#include <fastrts67x.h>
#define fo 2
```

```
#define fs 16
#define deltaf 2*PI*fo/fs
float theta1=0;
.. .
theta1=theta1+deltaf; //PLL in the next section
                      // updates this part
If (theta1>=2*PI)
theta1=theta1-2*PI;
.. .
Sinsample=sinsp(theta1);
Cosinesample=cossp(theta1);
```

- Note that theta1 and theta2 will be generated by the carrier tracking loops to be explained next. Initially, they should be set to zero. Please compile your program at this point to verify that your PAM receiver still works as before without any problem. Set the time knob of oscilloscope at least to 1 sec to see that the amplitude of the signal changes at the rate of the carrier offset.
- The below is the block diagram of the first step of the carrier tracking loop. This step takes square of the input samples and applies bandpass filter to the squared input samples.

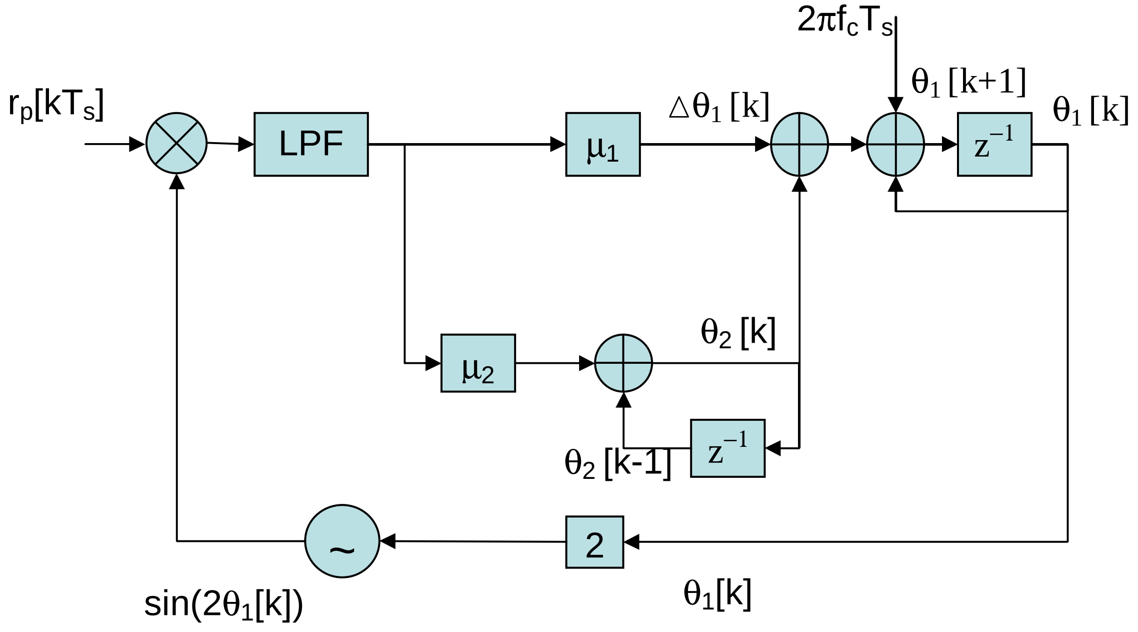


- This is alternative implementation for the PLL loop with single sin() function. The phase and carrier tracking loops are done using the single LPF output. Select $\mu_1 = 0.02$ and $\mu_2 = 0.0001$. θ_1 which is obtained from these loops at each sampling instant should be used in the demodulation block of PAM receiver. Note that both θ_1 should also be checked against 2π to keep them increasing without bound. The same applies to the original carrier phases. You can use Phi generated above as the main phase increment due to the carrier frequency. The output of the BPF above is used below. After mixing with sin wave, the mixed signals are processed by the LPF filter separately. Update equations are given below:

$$\Delta\theta[k] = LPF(r_p * \sin(2\theta_1[k])$$

$$\theta_2[k] = \theta_2[k-1] + \mu_2 \Delta\theta_1[k]$$

$$\theta_1[k+1] = \theta_1[k] + \theta_2[k] + \mu_1 \Delta\theta_1[k] + 2\pi f_c T_s$$



- Inspect your demodulated baseband PAM signal and compare it to the ideal baseband PAM signal. This time, the signals on both channels should not oscillate if your PLL works. Get a snapshot of the eye diagram on the screen. Set the time knob of oscilloscope to 1 sec to see that the amplitude of the signal remains the same after PLL locking.