Costas loop ASIC problem V2.0

EE287 S'22 semester project

This semester, we are working for a company that is implementing a BPSK data link as part of a larger ASIC. We have the assignment of constructing a Costas loop BPSK decoder. This decoder is a digital PLL with quadrature mixing. It is very sensitive to phase shifts, and is used to decode Binary Phase Shift Keying signals. You will create the Costas loop, search for a sync signal, and then provide a set of output bits 10/8 re-coded bits. Other blocks outside this design will use the bits. A typical Costas loop is illustrated below:

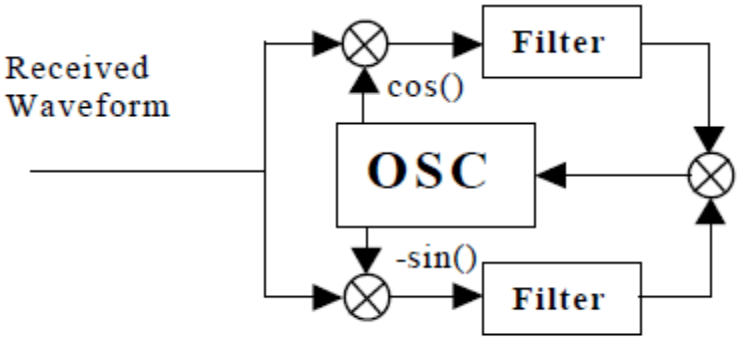

Fig 2 -- A basic Costas loop uses two filters, and matching them is far easier in software

Illustration from The Costas Loop – An Introduction by Eric Hagemann

The oscillator is a NCO, and implemented as an adder with feedback to form an accumulator and a sin look up table. The sin table has 12 bits in, and 16 bits out. This table will be provided, or you can type one in by hand. The filters are simple FIR filters, and contain 43 taps. The coefficients are in the costas0.cpp code. The circles with an X indicate multiplication. The received waveform is 10 bit ADC values sampled at 50 MHz. The carrier signal is 5 Mhz, and the data is encoded as 2.5 cycles per bit. The data is obtained on the output of the cos filter. A positive value indicates a 0, and a negative value indicates a 1. The loop can lock to the data inverted. The sync circuit will determine which way the loop locked, and invert the bits as needed.

The loop will lock to phase or frequency of the incoming data. The frequency lock range is about 3% of the carrier. The loop takes about 2000 samples to lock. (200 cycles of input). The test bench will send a carrier of around 2000 clocks before sending the sync signal.

The data is coded using 8/10 coding as specified in http://en.wikipedia.org/wiki/8b/10b_encoding

The sync character is 28.7 sent once. Note the cpp code does not contain the correct sync or end of packet values. The end of packet signal is 28.1. The 8/10 coding scheme ensures that neither of these codes will appear in the data stream. You must convert the bit stream from the costas loop to 8 bit data. You must send the sync signal on the first byte of data, and the lastByte signal on the last byte of data. This will require keeping things around for a byte until you determine if this is the last data.

The design has a stopin signal, and an eight entry output FIFO will be required.

Module Interface description

The module has a very simple interface. It consists of the following signals

Name	Bits	Dir	Comments
Clk	1	In	System clock. 200 Mhz frequency
Reset	1	In	Reset the module
ADC	10	In	ADC values into the block
PushADC	1	In	An ADC value is present. Arrive about every 9-11 clocks (10 on average)
Byte	8	Out	The data byte
pushByte	1	Out	Pushes output data
Sync	1	Out	Comes on the first byte of data after a sync
lastByte	1	Out	Comes on the last byte of data in the block
stopIn	1	In	Stop from the test bench. Test bench did not take the data

The ADC values are 10 bit signed numbers. They represent numbers between +1 and -1;

You are provided a set of sin and cos values for use with a NCO. The NCO is an accumulator that adds a phase value at every sample point, The upper 10 bits are used as input to the lookup function. The lower bits hold sub phase values. This makes a very accurate NCO. The project assumes the NCO is 32 bits. This allows sub Hertz resolution at the 5 MHz carrier frequency.

A table is available of 16 bit -sin and cos values assuming 0 to 2*PI is scaled to 0-1024.