

# **CDMA Baseband Processing on a TMS320C54x DSP**

Claude Winborn

EE 6390 Introduction to Wireless Communications Systems

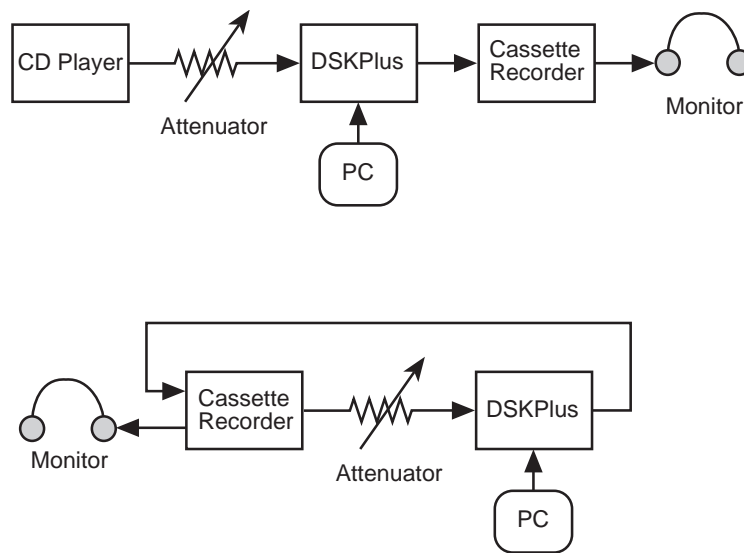
University of Texas at Dallas

## **Abstract**

The objective of this project is to program a TI TMS320C54x digital signal processor to perform rudimentary baseband processing needed in a wireless DS-CDMA mobile phone. A baseband CDMA transmitter will generate signals to be recorded on audio tape for later playback to the CDMA baseband receiver. The research required for this project therefore centered around practical CDMA transmitter and receiver design.

## I. Introduction

The objective of this project is to program a TI TMS320C54x digital signal processor to perform rudimentary baseband processing needed in a wireless CDMA mobile phone. A baseband CDMA transmitter will generate signals to be recorded on audio tape for later playback to the CDMA baseband receiver. The research required for this project therefore centered around practical CDMA transmitter and receiver design. Since single user DS-CDMA is bandwidth inefficient [1], the bandwidth limitation of the audiotape will ultimately limit the bandwidth of the transmitted speech. The purpose of this project, however, is to make a working transmitter and receiver.



*Fig1 Block Diagram of Transmitter and Receiver*

## II. Research Objective

Research was concentrated on the receiver module since design of the transmitter module is relatively straightforward. Because the receiver module must receive the signal from the audio tapes asynchronously, a local version of the chip oscillator and the symbol clock that is accurate in phase must be recovered from the received signal. With these two signals, the coherent detection of

the chip pulses and the proper time to sample the output can be made by any number of detection schemes. Selecting an appropriate detection scheme for this system was part of the research criteria.

### III. Phase Locked Loop

The design criteria for chip oscillator recovery can be found in Proakis[2], in the implementation of a phase locked loop (PLL). A block diagram of a PLL can be found in Figure 2.

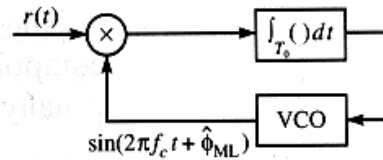


Fig. 2 Block diagram of a phase Locked loop

The received input signal is multiplied by the output of a voltage-controlled oscillator (VCO).

The product of the two signals is

$$e(t) = c \cos(2\pi f t + \phi) \sin(2\pi f t + \phi)$$

$$e(t) = \frac{1}{2} \sin(\phi - \phi) + \frac{1}{2} \sin(4\pi f t + \phi + \phi)$$

This signal is filtered to remove the latter, high frequency term and the resultant voltage is used to control the frequency of the VCO. In the case of this implementation the oscillator will be a numerically controlled oscillator (NCO) since the PLL will be implemented in DSP code. The filter is usually selected to have the following simple transfer function.

$$G(s) = \frac{1 + \tau_2 s}{1 + \tau_1 s}$$

Assuming the transfer function of the VCO is  $K/s$ , an integrator, where  $K$  is a gain factor, the transfer function for the PLL feedback loop can be written as

$$H(s) = \frac{(2\zeta\omega_n - \omega_n^2 / K) s + \omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$$

where  $\omega_n = \sqrt{K / \tau_1}$  is the resonant frequency of the feedback loop and  $\zeta = (\tau_2 + 1/K) / 2\omega_n$  is damping factor for the feedback loop. A PPL designed to track the chip oscillator will be first modeled in Matlab and appropriate values for  $K$ ,  $\tau_1$  and  $\tau_2$  will be determined.

#### IV. Symbol Synchronization

A design for symbol synchronization was also selected from Proakis[3]. The early-late gate synchronizer seems well suited to CDMA detection since the code correlator can be implemented as just another part of the synchronizer. Figure 3 is the block diagram for the synchronizer. The scheme used in this synchronizer is based on the fact that the code correlator output will ramp up to a maximum value as the phase advances through the optimum decision point, and then will ramp down as the phase passes through. Two code correlators are used. One samples in advance of the other by some time  $\delta$ . The loop will drive the phase of the voltage-controlled clock (VCC) until the magnitude of the two samples are equal, as in Figure 4.

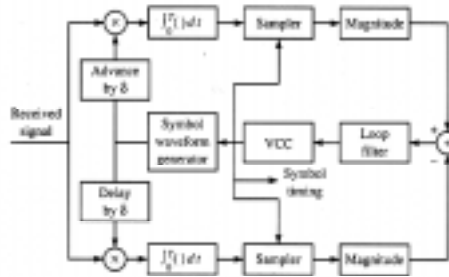


Fig 3 Block diagram of an early-late gate

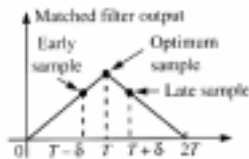


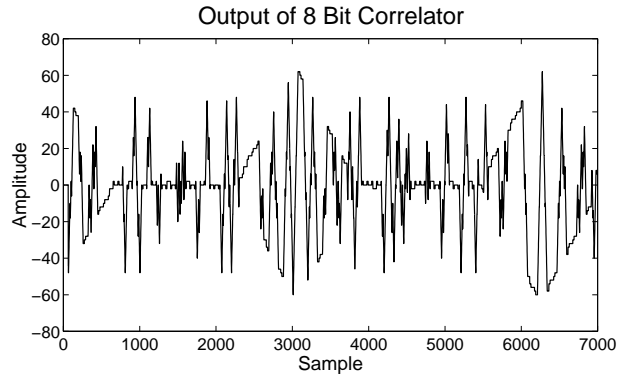
Fig 3. Correlator output

If the absolute value of the late sample is greater than the absolute value of the early sample, a condition where the local clock is running behind the transmitter, the positive voltage that emerges from the loop filter will speed up the VCO, bringing the oscillators back into synchronization. Since the early-late gate synchronizer is driven by the output of correlators, the bandwidth of the closed loop control system is relatively narrow compared to other schemes. This provides more averaging of additive noise, to the detriment of the ability of the synchronizer to track a rapidly changing symbol period. This symbol synchronizer also has a control loop and filter similar to the phase locked loop and the same equations apply. Values for  $\kappa$ ,  $\tau_1$  and  $\tau_2$  will be determined and the synchronizer will be modeled in Matlab.

## **V. Matlab Simulation One**

Matlab was used to select an 8 Bit spreading code that had good autocorrelation properties. A transmitter simulator was designed that would create an 8000 sample signal vector. The modulating signal chosen was a 60 Hz. tone quantized into 8 bit samples. The bits of the signal samples were selected in sequence, starting with the MSB, to be used to determine the polarity of the next spreading code sequence.

The first receiver simulator implemented in Matlab simply correlated the incoming signal with the spreading code. Whereas the transmitted signal was modulated at 500 BPS, the receiver was set to receive a signal modulated at 505 BPS. This would allow the incoming signal to pass through the region several times where a code match with the correlator would be expected. The output of the receiver correlator yielded a signal that was unusable for any kind of synchronization. This was thought to be caused by the pseudo random changes in polarity of the match code due to the modulation of the signal. It was concluded that some kind of decision directed approach to synchronization would be needed.



*Fig 4 Output from First Matlab Simulation*

## **VI. Matlab Simulation Two**

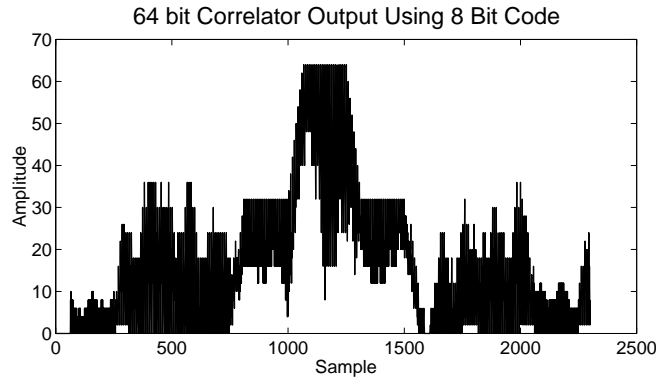
The 8 bit spreading code was again used, but this time the correlation was lengthened to 64 bits, consisting of eight separate 8-bit correlations. The results of these correlations were either added or subtracted from the total result depending on whether they were positive or negative, respectively.

This algorithm, it was hoped, would mitigate the effect of the modulation that was preventing a good synchronization signal from being formed. The waveform that resulted was indeed better.

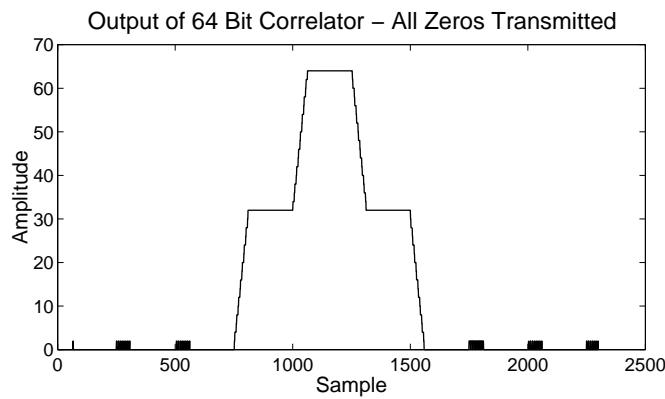
There was a definite correlation peak, but the signal was too noisy to use. Heavy filtering of samples for input to an early-late gate synchronizer would only slow convergence.

The improvement made to the waveform by removing the modulation of the spreading code, that is, forcing the transmitted bits to one, suggested a different approach to the problem. If an initial synchronization period were introduced with a consistent waveform, synchronization would be easy using an early-late gate phase locked loop. Decision directed synchronization would then take over at the end of the synchronization period.

It was also thought that increasing the length of the spreading code from 8 to 16 chips would produce a larger peak when a code match occurs relative to where no match occurs.



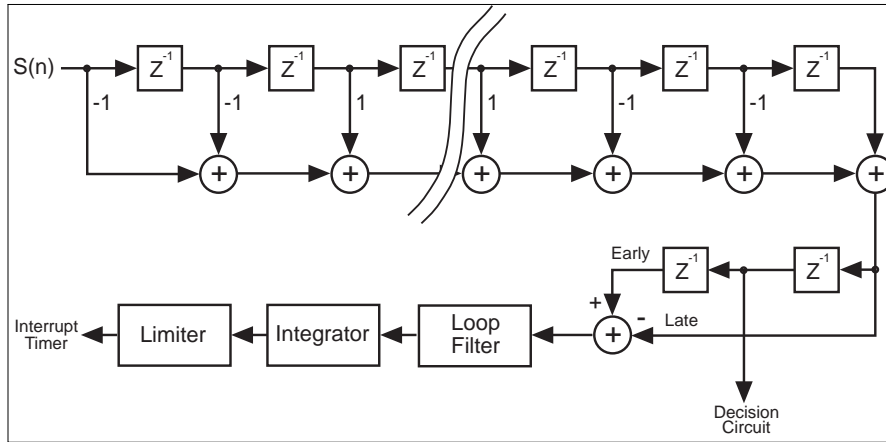
*Fig 5 Output of 64 Bit Correlator*



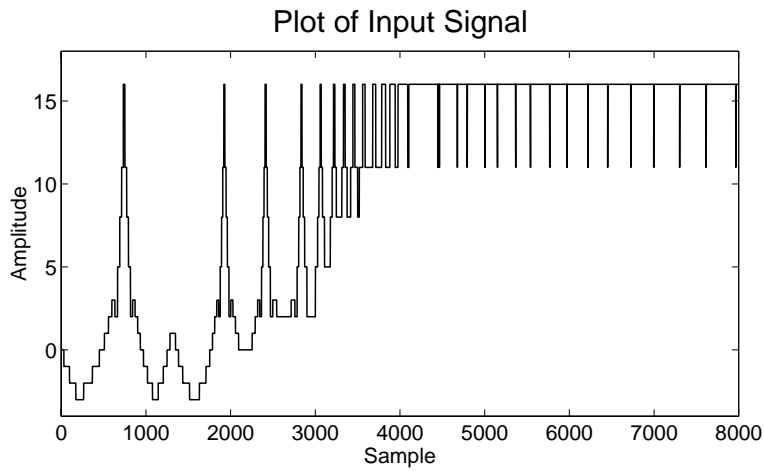
*Fig 6 Output Without Modulation*

## VII. Third Matlab simulation

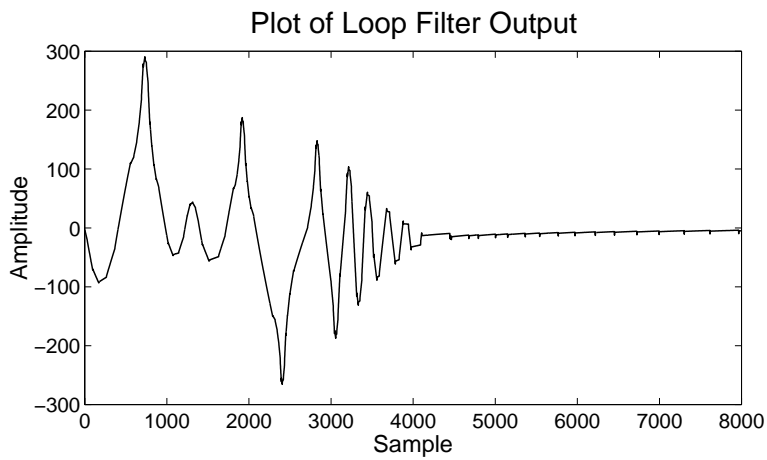
Matlab was again used to select a 16 Bit spreading code with good autocorrelation properties. A simulator was coded to test the early-late gate phase locked loop. The symbol rate was dropped to 125 symbols per second to accommodate the larger spreading code. The initial NCO frequency was set to 126 Hz. Values for the loop filter,  $a$ ,  $b$ , and  $k$ , were picked for a good response. A block diagram of the early-late gate implementation can be seen in Figure 7. The Matlab simulator for this configuration was run with different values for the PLL parameters. Figures 8, 9 and 10 show the loop signals for the final values chosen.



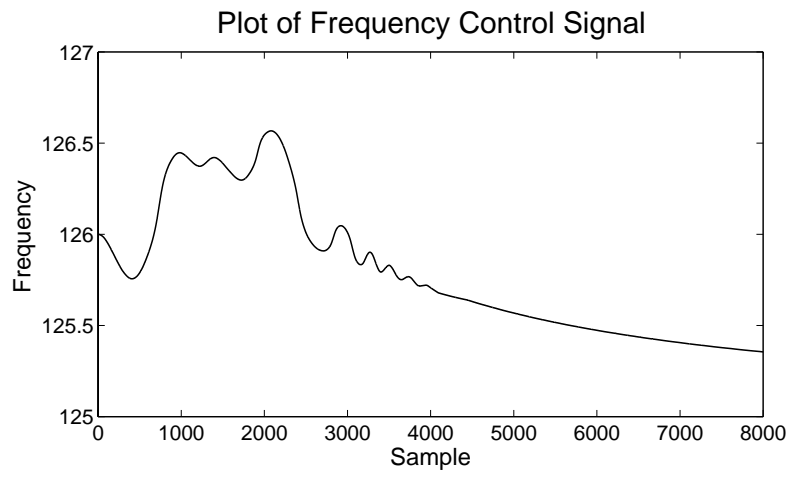
*Fig 7 Early-Late Gate Block Diagram*



*Fig 8 Input Signal Showing Synchronization*



*Fig 9 Filter Output During Synchronization*



*Fig 10 Frequency Output During Synchronization*

## References

- [1] W. C. Y. Lee, "Overview of Cellular CDMA", *IEEE Transactions on Vehicular Technology*, May, 1991, Vol. 40, No. 2
- [2] J.G Proakis. *Digital Communication*, McGraw-Hill, Third edition., ISBN 0-07051726-6, 1995, pp 341-343
- [3] J.G Proakis. *Digital Communication*, McGraw-Hill, Third edition., ISBN 0-07051726-6, 1995, pp 362-365
- [4] J.G Proakis. *Digital Communication*, McGraw-Hill, Third edition., ISBN 0-07051726-6, 1995, pp 849-859