A Software Phase-Locked Loop from Theory to Practice: TMS320C6000 DSP Based Implementation and Analysis

Sithamparanathan Kandeepan
Wireless Signal Processing Group
National ICT Australia, Canberra
RSISE, Australian National University
Kandeepan.Sithamparanathan@nicta.com.au

Abstract

The study of phase locked loops (PLL) has been heavily treated in literature and most of the theoretical and the analytical results of such are verified using simulations. Here we provide a real-time implementation of a PLL on a digital signal processor (DSP) and analyse and verify the theoretical results associated with it on the implemented system. Such work takes us one step above from the traditional simulation and analysis of PLL to real-time implementation and analysis. The steady state and the acquisition of the PLL are analysed. Issues such as quantization errors are also discussed.

1. Introduction

The phase locked loop [1-4] is a useful control systems tool used heavily in communications engineering, radar, sonar, control engineering and many other applications. In communications PLLs are used for carrier tracking, frequency synchronization, phase synchronization and symbol timing synchronization. Here we design a PLL in software on DSP to track carrier signals and also to track the fundamental frequency component of periodical signals. The DSP used is the Texas Instrument's C6713 based floating point processors [11-16]. The implemented software based PLL is used to analyse the performance and compare it with the theoretical analysis. The advantage of such software implemented PLL and analysis is that the loop could be easily modified to incorporate signal processing components to improve the performances of the PLL. Generally, with the trade-off between the acquisition and the tracking performance of PLL, such signal processing elements may be used to improve either the acquisition or the tracking performance depending on the application. In many cases researchers use simulations to analyse the performances of PLLs, but we go further a step up and use real-time implementation of PLL to analyse the performance. Here, apart from the theoretical background, implementation and experimental results, we also provide detailed explanation on the hardware platform and design structures of the PLL on the DSP. Section-2 introduces to the theoretical background of the PLL and its corresponding loop designs. In section-3 we provide the hardware details and the interrupt based DSP implementation with details on the Analog-to-Digital (ADC) and Digital-to-Analog (DAC) converters. Sections 5 and 6 provide the experimental results and comparisons with the theoretical results such as the steady state phase error, phase jitter and the steady state instantaneous frequency jitter. We also look at the phase plane portrait of the loop. Finally we provide some conclusion in Section-7.

2. Phase Locked Loop

A typical PLL implemented in hardware consists of an error detector, loop filter and a voltage controlled oscillator (VCO). For software implemented PLL, the VCO is augmented by a numerically controlled oscillator (NCO). The NCO is a software implemented sinusoidal waveform generator that changes its frequency depending on the numerical input. The block diagram of software implemented PLL is given in Fig-1. The error detector is a multiplier based sinusoidal [1-4] phase detector, which also has a tendency of producing 2nd harmonic frequency components at the output.

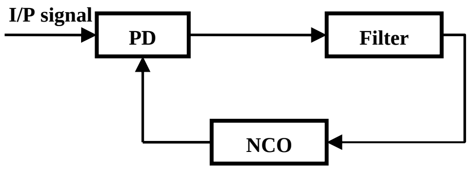


Figure – 1, Digital phase locked loop model

An alternative approach to design the phase detector is to use the four quadrant arctan function. The four quadrant arctan based phase detector, which requires the inphase and the quadrature inputs of the signal, is easily implemented in software [8,9] and has got several advantages over the sinusoidal phase detector. In this paper however we consider the traditional sinusoidal based error detector for our implementation and analysis. The loop filer plays a major role on the performance of the loop; a single pole loop filter gives a second order loop, which is capable of tracking frequency offsets present in the received signal. The loop filter we use here is an imperfect type of integrator with a single pole. The mathematical model of the NCO is given by,

$$V(z) = \frac{k}{z-1} \tag{1}$$

where, k is the constant that controls the NCO. The output of equation (1) produces the phase required to generate the sinusoidal signal, which is the local oscillator signal. The locally generated signal is given by,

$$r[n] = \cos(\theta[n]) \tag{2}$$

where, θ[n] is the phase produced by equation (1). The linearised loop for the block diagram in Fig-1 is given by the closed loop transfer function,

$$H(z) = \frac{D(z)V(z)}{1 + D(z)V(z)} \tag{3}$$

where, D(z) is the transfer function of the loop filter. By ignoring the higher order harmonic components, the error signal produced at the output of the phase detector is given by,

$$\varepsilon = \sin(\varphi_e) \tag{4}$$

where, φ_e is the phase error between the received signal and the locally generated signal. Another loop parameter that defines the loop performance is the closed loop bandwidth. The closed loop bandwidth for a discrete loop is given by [2],

$$2B_L = \frac{B_i}{2\pi j} \oint_c H(Z)H(1/Z)Z^{-1}dZ \tag{5}$$

where, B_i is the noise equivalent input bandwidth to the loop. Having defined the loop model, in the following sections we see how we design and implement the loop on the DSP.

3. Hardware Test-bed

The hardware test-bed used here includes two Texas Instrument's C6713 based floating point DSPs [11-16] attached to host PCs. The processor runs at a speed of 225Mhz with two arithmetic and logical units and 8-functional units each, allowing 1800 million instructions per second (MIPS). The Development board that contains the processor, manufactured by 'Spectrum Digital' also includes onboard memory, an audio codec chip, digital interfaces and several others which we present in this section. The development board is also supported by a software development tool called the Code Composer Studio (CCS) [21,22] developed by Texas Instruments. The CCS allows to target the DSP development board from the host PC through a USB based Joint Target Action Group (JTAG) interface, which gives access to the onboard peripherals and interfaces. The CCS also contains necessary board support (BSL) and chip support (CSL) [20,25] libraries to initialize and setup the hardware. Fig-2 shows some of the functional block diagram of the hardware peripherals on the development board. The AIC23 [23] is the analog interface to the DSP and the external world, which we discuss in more detail in the subsequent section. The audio codec is connected to the DSP through two multi channel buffered serial ports (McBSP), one of which is used for controlling the audio codec (McBSP0), and the other is used for data transfer (McBSP1). The memory onboard the system is interfaced by the Enhanced Memory Interface or the EMIF with 256KB of flash memory and 16MB of onboard memory. Some of the other units on the development board are, the memory expansion unit, external JTAG interface, four dip-switches and four light emitting diodes which are controllable using the CCS from the host PC. The development board is also capable of running on stand-alone mode without the host PC. The flash memory is used for such purposes and can be programmed to perform particular tasks during boot-up mode.

3.1. AIC23 Audio codec

The AIC23 audio codec chip manufactured by Analog Instruments is the interface to the analog world. Alternatively the daughter card expansion slot may also be used to have daughter cards which would by pass the onboard audio codec (AIC23) and function as the analog interface. The AIC23 has four 3.5mm audio ports for input and output purposes. The four ports are, a stereo microphone input, a stereo line-in, a stereo headphone output and a stereo line-out. The interfaces can handle a peak to peak voltage of 6v on the analog interface side of the pins. The codec is clocked by a 12MHz crystal clock allowing the sampling frequency to change between 8kHz to 96kHz. The sigma-delta technique is used on the codec to improve the signal to noise ratio of the received signal. The received samples are passed through an interpolation filter, modulator and a decimator before encoding it using the 2's complement. Hardware interrupts are performed by the DSP or the EMIF to transfer data from and to the audio codec through the McBSP.

4. Software Implementation

The implementation of the loop makes use of the multiple functional units and the dual arithmetic and logical units in the DSP. The locally generated signal is produced by a lookup table method to produce the sine values. The received samples are stored in the memory location r and multiplied by the locally generated signal x to produce the error signal e. (MPY denotes the multiplication operation and, ADD denotes addition in assembly language)

$$x, r \quad \text{MPY} \quad e$$

The error signal is then passed through the IIR single pole loop filter. The filtering operations are done in parallel to make use of the available resources on the processor. The time domain difference equation of the IIR filter is given by,

$$w[n+1] = ae[n] + (1-a)w[n]$$

where, a is the filter coefficient. The corresponding assembly codes with parallel instructions are given by,

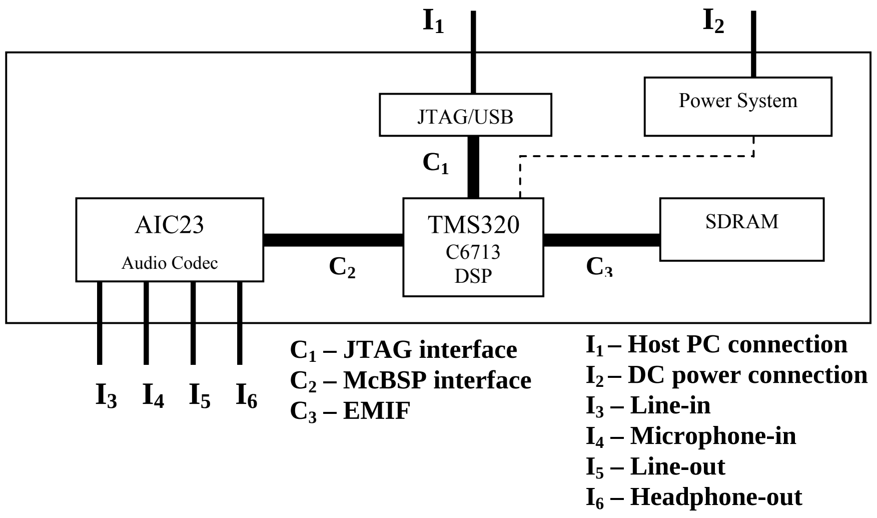


Fig-2, Test-bed hardware, DSP development board and its main functional subsystems

```
||    a , e        MPY   b1
||    (1-a), w     MPY   b2
      b1, b2       ADD   w
```

Then the output of the filter is input to the NCO, which is essentially a phase accumulator, to produce the local signal.

```
      k , w     MPY   u
      u, teta   ADD   teta
```

A single pass in a loop takes up to seven cycles together with data movement to memory segments to produce the output. This gives us more operational cycles allowing us to use additional complex signal processing operations in real-time within the loop before the next sample is taken in.

5. Linear Analysis

In this section we present some standard linear analysis conducted on PLL, which are used in the following sections to verify the real-time software implemented PLL. We look into the steady state performance of the linear loop in (3).

5.1.Time Series Analysis

The phase error process in the loop shows a transient and a steady state response as any typical feedback loop. The time series model of the phase error process for the linear loop is of great interest to us.

$$\theta[n] = \Omega_O n - \frac{\Omega_O}{k} + \frac{\Omega_O r^n}{ak\sin(\delta)}\begin{bmatrix} r^2 \sin((n-1)\delta) \\ -2r\sin(n\delta) + \sin((n+1)\delta) \end{bmatrix} \quad (6)$$

For a frequency error of Δf in the received signal the time series expression for the linear loop in (3) is found by solving the difference equation for the error signal, which is given by (6) [8,9], where, $r = |z|$, $\delta = \angle z$, $\Omega_o = 2\pi\Delta f T_s$ and z, z* are the poles of the under damped imperfect 2nd order loop. The sampling duration of T_s of the audio codec AIC23 is set to 1/(96 kHz), where the sampling frequency is given by f_s = 96kHz. In (6) we assume that the carrier frequency is set to zero and the loop tracks the frequency error directly. This is usually used in base-band processors to offset for frequency errors in the received signal. Equation (6) clearly shows the dying out transient component together with the steady state component.

5.2. Steady state phase error

The steady state phase error of the loop is the constant phase error value after transient. For an imperfect loop the steady state phase error is useful in determining the lock-in range of the loop. The steady state phase error may be found by setting n→∞, in (6), minus the first term. However, in many situations a direct closed form solution to the time series model is unobtainable; in such cases we use the final value theorem in the discrete domain to find the steady state phase error. For the loop in (3) the steady phase error is given by,

$$\varphi_{e-ss} = \frac{2\pi\Delta f T_s}{k} \quad (7)$$

From (7) we can see that the loop is dominantly controlled by the NCO parameter k, which also known as the loop gain.

5.3. Acquisition Time

The acquisition time is the time for the loop to acquire and lock on to the fundamental frequency of the received signal. For the noiseless case we define the acquisition time where the transient of the error dies-out with +/-5 percent of the final steady state value. For the loop defined in (3) the acquisition time is given by,

$$T_{acq} = T_S\left[\frac{\ln[0.05ak\sin(\delta)]}{\ln[r]}\right] \quad (8)$$

5.4. Steady state phase noise

Due to the presence of additive noise in the received signal, the loop experiences some jitter in the phase error process, this is known as phase noise or loop noise. The phase noise depends on the input signal to noise ratio (SNR) and the closed loop bandwidth B_L. For a linear loop with additive white Gaussian noise at the input signal, Gardner [1] for a continuous loop and Lindsey [2] for a discrete loop, have shown that the phase noise is given by,

$$\sigma_\varphi^2 = \frac{N_0 B_L}{P} \tag{9}$$

where, P is the input signal power and N_0 is the single sided power spectral density of the noise process.

5.5. Steady state instantaneous frequency jitter

The input noise process causes a jitter in the frequency that the loop locks onto. This is known as the instantaneous frequency jitter of the loop. The instantaneous frequency jitter for the discrete loop is given by [7,9],

$$\sigma_f = \frac{\sigma_\varphi}{T_s\sqrt{2\pi}} \tag{10}$$

The jitter associated with the instantaneous frequency of the NCO is relatively high with respect to the lock-in frequency.

6. Experimental Results

Two C6713 based hardware development boards were used to conduct the experiments. The PLL was implemented on one of the boards and the other was used as a signal source. Two communication channels were used, one a direct wiring of the two development boards through a 3.5mm pin based audio cable, and the other a wireless channel in the FM-band (88.7MHz). Initially a black box based testing was conducted to study the additive noise process. The analog components in the audio codec hardware produce some amount of thermal noise which we analyse and present here. The noise samples taken from the analog input were analysed on its statistical properties. The noise process follows a Gaussian distribution as shown in Fig-3. The noise is zero mean with a variance of σ^2 = 2.1868 in units sample-amplitude. Although the codec contains nonlinear devices we treat the entire hardware module as a black box and assume the noise process is flat within the frequency range of the discrete system. The single sided power spectral density of the noise process shown in Fig-3 is computed to be as N_0 = -43.4143 dB/Hz over a bandwidth of 48kHz. The PLL was tested with the two DSPs attached with the 3.5mm cable and the loop behaviour was recorded for post-analysis. Fig-4 shows the recorded phase error process of the loop with time. The figure also shows the expected theoretical response for the phase error process and its corresponding steady state value. The steady state value of the phase error process reaches 6-deg as seen in the figure, which can also be calculated from the expression for the steady state phaser error given in equation (7).

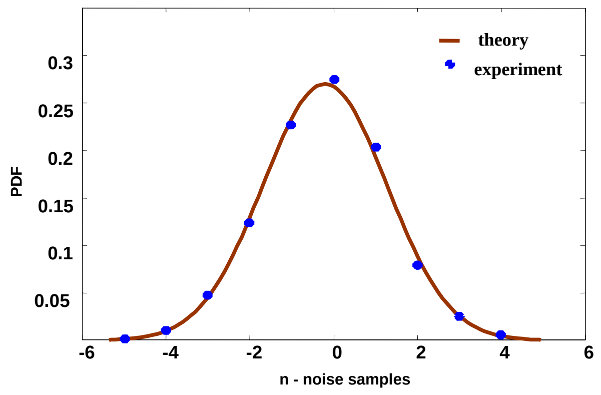


Fig-3, Thermal noise statistics, from DSP-1 to DSP-2

The phase error at steady state experiences some jitter due to noise. We see in later section that the jitter is not only caused by the additive Gaussian noise at the receiver but also by the finite precision representation of the signal samples, or the quantisation noise in other terms.

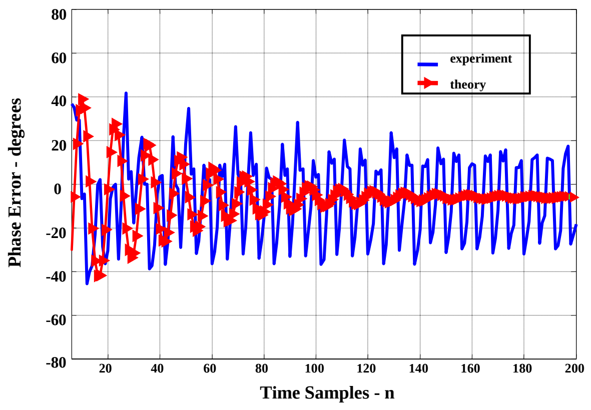


Fig-4, Phase error process of the loop, experimental and theoretical

Next, we examine the instantaneous frequency error of the loop. The instantaneous frequency error of the loop is shown in Fig-5, both the experimental and theoretical results are shown. The instantaneous frequency error also experiences jitter at steady state due to the additive thermal noise and the quantisation noise. The two figures showing the phase error process and the frequency error process, matching with the theoretical and the experimental results, validate our design and analysis strongly. The noise performance of the loop is also of great interest to us. We analyse and compare the phase jitter and the instantaneous frequency jitter of the loop during steady state operation. Experiments were conducted to measure the noise performance of the loop. We should note here that our main aim in the design and implementation process of the PLL was to bring the error performance of the loop to the theoretical limit as much as possible, and to make the loop linear. To measure the jitter performance of the loop during steady state, we varied the transmit power at the transmitter hence varying the received signal to noise ratio, and measured the standard deviation of the phase noise and the instantaneous frequency error respectively.

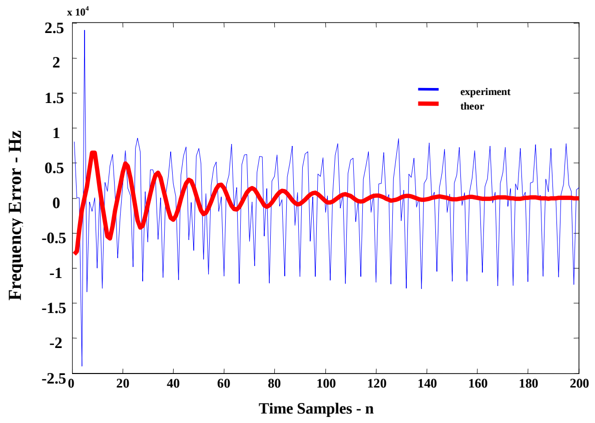


Fig-5, Instantaneous frequency error process of the loop

The experimental results were then analysed with the theoretical results given in (9) and (10) respectively. Figures 6 and 7 depict the phase and instantaneous frequency jitter performances of the loop for various loop signal to noise ratio levels. From the figures we see that the jitter performance of the loop is very close to the theoretical performance.

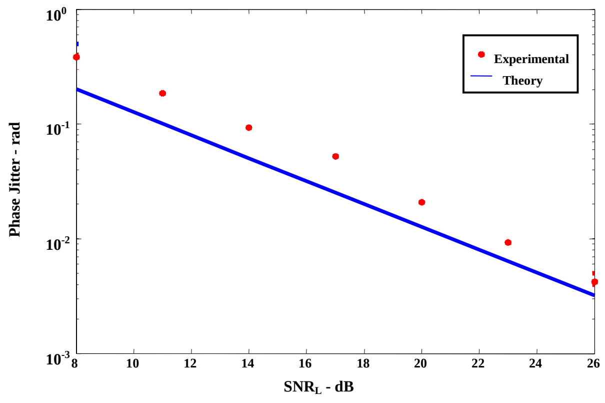


Fig-6, Steady state phase jitter performance of the loop

The slight difference between the obtained performance and the theoretical performance may be verified using the finite precision representation of the signal samples in the DSP. As stated before, the audio codec is a 16-bit codec which produces significant quantisation noise. The quantisation noise is not considered in the linear theoretical analysis presented in the previous sections here. Characterisation of quantisation noise is application specific and depends on the signal model that is used. The treatment of quantisation noise in the linear loop analysis is beyond the scope of this paper. Next we look at the loop dynamics by analysing the phase plane portrait of the feedback system. The phase-plane portrait [4] for the PLL is the plot of the trajectories with phase error on the x-axis and the frequency error on the y-axis for various initial conditions. This shows the pull-in process of the loop with the frequency error and the phase error reaching steady states in a single graph. Fig-8 depicts the phase-plane portrait of the imperfect 2nd order loop for a single initial value. From the figure we see how the loop pulls-in the signal by acquiring the phase and the frequency of the incoming signal. The phase error and the frequency error reach the steady state values and wander around it due to noise.

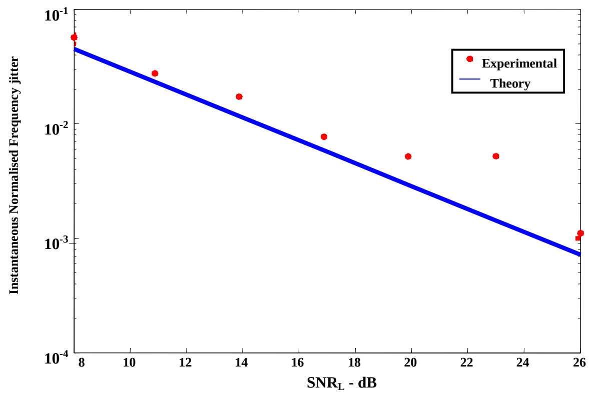


Fig-7, Instantaneous frequency jitter of the loop

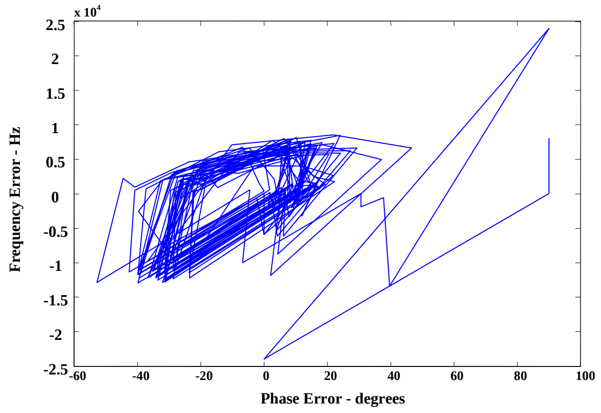
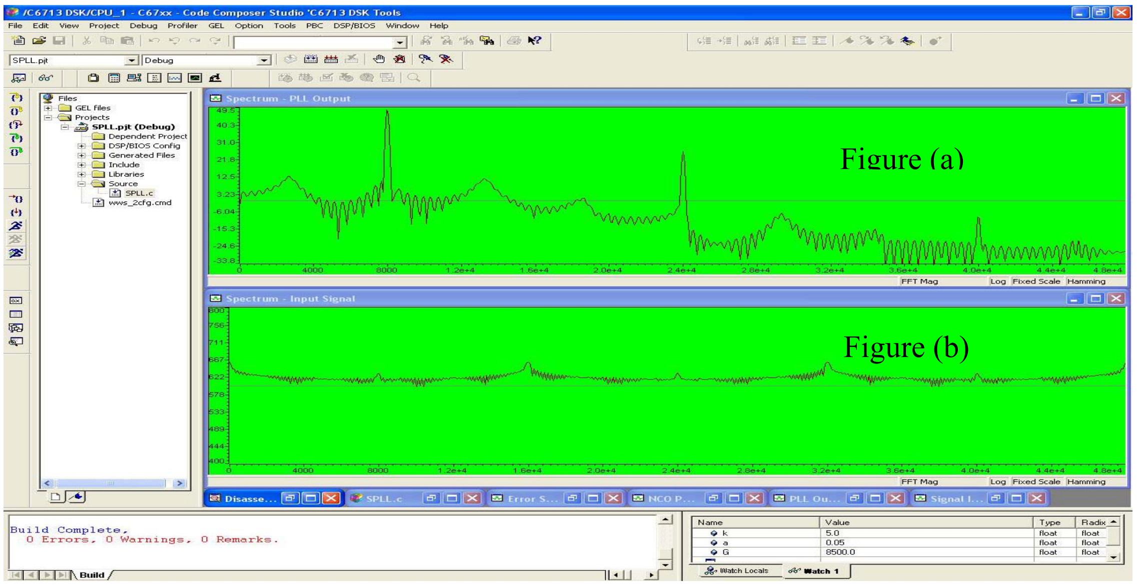


Fig-8, Phase plane portrait of the 2nd order difference system

The PLL was then used to track carrier signals for an indoor wireless channel operating in the FM-band at 88.7MHz. The purpose of this experiment was to see how well the PLL performs under non-ideal situations such as operating in an indoor wireless channel. The transmit DSP was attached to an FM transmitter and the receive DSP running the PLL was attached to an FM receiver. The spectral domain performance of the synchronisation system for the FM test-bed is shown in Fig-9. The figure shows the received in-band signal spectrum transmitted through the indoor wireless channel with received signal embedded in it. As we see from the figure, the received signal is hardly distinguishable within the channel, and the PLL tuned to the carrier signal pulls-in the carrier signal with high gain as shown in the same figure. The figure is taken out form the software tool CCS running on the receiver host computer targeting the real-time DSP.

7. Conclusion

A software-implemented PLL was presented in this paper. The PLL was implemented on TI's C6713 based DSP and was analysed on its performance in real time with theoretical analysis. The goal of the design, implementation and analysis was to bring the PLL performance to the achievable theoretical limits.



```
Build Complete,
  0 Errors, 0 Warnings, 0 Remarks.
```

Name	Value	Type	Radix
k	5.0	float	float
a	0.05	float	float
G	8500.0	float	float

Fig-9, Spectral properties of the FM-band test-bed with software PLL (a) PLL output spectrum (b) FM-band received signal spectrum

It has been seen from the experimental results that this was achieved with limitations due to finite precision representation of sampled signal. The key motivation of the work was that traditionally PLL are designed and analysed using simulations, but here we go a step forward and analyse the loop in real-time implemented on DSP.

8. Acknowledgement

The author would like to thank the Wireless Signal Processing program at the National ICT Australia. National ICT Australia is funded through the Australian Government's *Backing Australia's Ability initiative*, in part through the Australian Research Council

9. References

[1] F.M.Gardner,*"Phase Lock Techniques"*,NY,Willey, 1979.
[2] W.C.Lindsey, and C.M.Chie, *"A Survey of DPLL"*, Proceedings of the IEEE, pp296-317 April 1981.
[3] H. Meyer, G Ascheid, *"Synchronisation in Digital Communications"*,vol-1, John Willey & Sons, 1990
[4] A.J.Viterbi,*"Principles of Coherent Comms"* McGraw-Hill, 1966
[5] M. P. Fitz and R. J.-M. Cramer, "A Performance Analysis of a Digital PLL Based MPSK Demodulator," *IEEE Trans on Comms*, vol. 43 No.2/3/4, pp. 1192-1201, Feb/Mar/Apr 1995.
[6] R.C.Tausworthe,"A Second/Third-Order Hybrid Phase Locked receiver for Tracking Doppler Rates",*JPL Tech Rep* 32-1526,vol.1,pp 42-45
[7] S. Kandeepan, S. Reisenfeld, "Frequency Jitter of a Digital Phase-Locked Loop and Comparison with a Modified CRB",*Comm Systems, 2002. ICCS 2002. The 8th International Conference on*, pp.96-100, Vol.1, 25-28 Nov 2002, Singapore
[8] S.Kandeepan, S.Reisenfeld, "Frequency Tracking and Acquisition with a Four-Quadrant arctan-Based Digital Phase-Locked Loop", *ICICS-PCM 2003 proceedings of*,Vol.1,pp 401-405,15-18 Dec 2003, Singapore.
[9] S.Kandeepan, *"Synchronisation Techniques for Digital Receivers"*, PhD Thesis, University of Technology Sydney, 2003
[10] M.C.Jeruchim, P. Balaban, and K. S. Shanmugan, *Simulation of Communication Systems, Modelling, Methodology and Techniques*, 2nd ed: Kluwer Academic/Plenum Publishers, 2000.
[11] R. Chassing, Digital Signal Processing and Applications with the C6713 and C6416 DSK, John Wiley & Sons, 2005.
[12] R.Chassing, *DSP Applications Using C and TMS320C6x DSK*: John Wiley & Sons, 2002.
[13] N. Kehtarnavaz and B. Simsek, *C6X-Based Digital Signal Processing*: Prentice Hall, 2000.
[14] N.Dahnoun, *DSP implementation using the TMS320C6000 DSP platform/Naim Dahnoun*, 1st ed. Harlow, England, New York, Prentice Hall, 2000
[15] *How to Begin Development Today with the TMS320C6713 Floating Point DSP*, SPRA809, Texas Instruments, Texas, 2003
[16] TMS320C6713 *Floating point Digital Signal Processor*, SPRS186, Texas Instruments, Dallas Texas
[17] TMS320C6000 *Programmers Guide*, SPRU198G, Texas Instruments, Dallas, Texas, 2002
[18] TMS320C6000 *CPU and Instruction Set Reference Guide*, SPRU189F, Texas Instruments, Dallas, Texas, 2000
[19] TMS320C6000 *Peripherals Reference Guide*, SPRU190D, Texas Instruments, Dallas, Texas, 2001
[20] TMS320C6x *Peripheral Support Library Programmers Reference*, SPRU273B, Texas Instruments, Dallas, Texas, 1998
[21] *Code Composer Studio Users Guide*, SPRU328B, Texas Instruments, Dallas, Texas, 2000
[22] TMS320C6000 *Code Composer Studio Tutorial*, SPRU301C, Texas Instruments, Dallas, Texas, 2000
[23] TLV320AIC23 *Stereo Audio Codec, 8- to 96-kHz, with Integrated Headphone Amplifier Data Manual*, SLWS106G, Texas Instruments, Dallas, Texas, 2003
[24] TMS320C6000 *DSP/BIOS User Guide*, SPRU423, Texas Instruments, Dallas, Texas, 2002
[25] TMS320C6000 *Chip Support Library API User's Guide*, SPRU401F, Texas Instruments, Dallas, Texas, 2003