# PLLSim – An Ultra Fast Bang-Bang Phase Locked Loop Simulation Tool

Michael Chan
School of ITEE
University of Queensland
Brisbane, 4067
e-mail : mchan@itee.uq.edu.au

Adam Postula
School of ITEE
University of Queensland
Brisbane, 4067

Yong Ding
Nanosilicon Pty Ltd
Brisbane, 4113

**Abstract - This paper presents a simulation tool targeted specifically at bang-bang type phase locked loop systems. The aim of this simulator is to quickly and accurately predict important PLL transient characteristics such as capture range, locking time, and jitter. We present a behavioral model for bang-bang type PLLs, and show how the application of this model in a simulator can speed up simulation time by four to five orders of magnitude. With this performance, Monte-Carlo simulation techniques become not only feasible, but convenient. The simulator also models the major non-idealities typical of phase locked loop systems. The accuracy of the simulator is confirmed via detailed analysis and comparison with Matlab Simulink based models.**

## I. Introduction

Bang-bang phase detector based phase locked loops (PLL's) are becoming more and more important in today's multi-gigabit communications systems. As well as having a simpler structure than their linear counterparts, bang-bang phase detectors can run at the highest speed an IC fabrication process can make a working flip flop. Furthermore, since data is typically sampled as a part of their operation, they exhibit no systematic phase error [1-3, 5].

Ref. [4] presents an analytic expression to predict a conservative capture range of a bang-bang PLL, but no such equations exist yet to predict locking time, and PLL jitter and tracking performance in lock. Consequently the design process for bang-bang phase locked systems is very reliant on simulation tools. At the system level, time step simulators such as Matlab Simulink are typically employed. Time step simulation techniques are very inefficient though. A PLL's bandwidth is several orders of magnitude less than its reference signal, and a clock divider in the PLL feedback path means the frequency of the recovered clock can be higher still. With each cycle requiring hundreds of time slices to reproduce with minimal quantization error, a straight time domain simulation can require tens to hundreds of millions of iterations translating into excessively long simulation times.

Various techniques to speed up PLL simulation time have been reported. Ref. [6] describes a uniform time step simulator targeting PLL's with fractional-N frequency dividers. Ref. [6] achieves simulation speedup by employing discreet time behavioral models of the PLL components. Ref. [7] also utilizes behavioral models for fast simulation. Specifically, non-linear phase domain macro models are derived for various PLL components, and accuracy on a par with SPICE simulations is reported. Benefits of behavioral level simulation of phase locked systems are also discussed in [8].

This paper presents a simulation tool targeted specifically at bang-bang phase detector based phase locked loops. A model describing the behavior of bang-bang PLLs is presented, and a simulation tool based on this model is written. The simulation tool also models the various non-idealities typical of phase locked systems. By specifically targeting bang-bang type phase locked loops, astounding simulation performance is realized. Benchmarks against simple PLL models implemented in Matlab Simulink yield simulation speedup on the order of five orders of magnitude. Careful analysis of the results produced by PLLSim and Matlab Simulink confirms the validity of the model used.

## II. Bang-bang PLL Model

Fig. 1 shows a 2nd order charge pump based bang-bang phase locked loop. $\omega_{ref}$ is the frequency of the reference signal, and $\omega_{fb}$ is the frequency calculated as the loop filter capacitor voltage $V_{cap}$ multiplied by the VCO gain $k_{VCO}$ (both in radians per second). The actual frequency of the recovered clock signal varies around $\omega_{fb}$, depending on the sign of the current flowing through the loop filter resistance R. $\varphi_{ref}$ and $\varphi_{fb}$ refer to the phase of the reference signal, and recovered clock signal respectively. The state of this system can be characterized by a frequency error $\omega_{err} = \omega_{fb} - \omega_{ref}$, and a phase error $\varphi_{err} = \varphi_{fb} - \varphi_{ref}$. The bang-bang phase detector measures the sign of the phase error on every tick of the recovered clock.

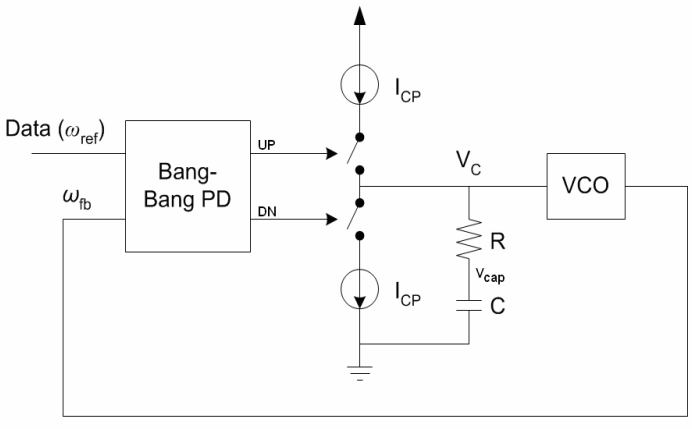


Fig. 1 – Bang-bang PLL Block Diagram

1-4244-0630-7/07/$20.00 ©2007 IEEE.

For a negative phase error, the UP signal is asserted (an UP pulse), otherwise the DN (down) signal is asserted (a DN pulse).

The state of the PLL changes every time step $T_X$ according to:

$$\omega_{err}(t+T_X(t)) = \omega_{err}(t) + \zeta\Delta\omega(t) \quad (2.1)$$

$$\phi_{err}(t+T_X(t)) = \phi_{err}(t) + \zeta\Delta\phi(t) + \omega_{err}(t)T_X(t) \quad (2.2)$$

Where $T_X(t)$ is the period of the recovered clock during the last UP or DN pulse, $\zeta$ equals 1 for an UP pulse and -1 for a DN pulse, and $\Delta\varphi(t)$ and $\Delta\omega(t)$ are the PLL phase step and frequency step respectively. Fig. 2 shows the ripple on the VCO control voltage Vc in response to a series of UP and DN pulses. From Fig. 2, the PLL phase step and frequency step can be calculated as:

$$\Delta\omega(t) = \Delta\omega` \frac{T_X(t)}{T_{ref}} \quad (2.3)$$

$$\Delta\phi(t) = (\Delta\phi` - \frac{1}{2}T_{ref}\Delta\omega`)\frac{T_X(t)}{T_{ref}} + \frac{1}{2}T_X(t)\Delta\omega(t) \quad (2.4)$$

Where

$$\Delta\omega` = \frac{k_{VCO}I_{cp}T_{ref}}{C} \quad (2.5)$$

$$\Delta\phi` = K_{VCO}I_{cp}RT_{ref} + \frac{1}{2}T_{ref}\Delta\omega` \quad (2.6)$$

We define the constants $\Delta\varphi`$ and $\Delta\omega`$, since they are useful for summarizing the operation of a bang-bang PLL. Furthermore, $\Delta\varphi`$ and $\Delta\omega`$ are approximately equal to $\Delta\varphi(t)$ and $\Delta\omega(t)$. During an UP pulse, the average frequency of the recovered clock is given by:

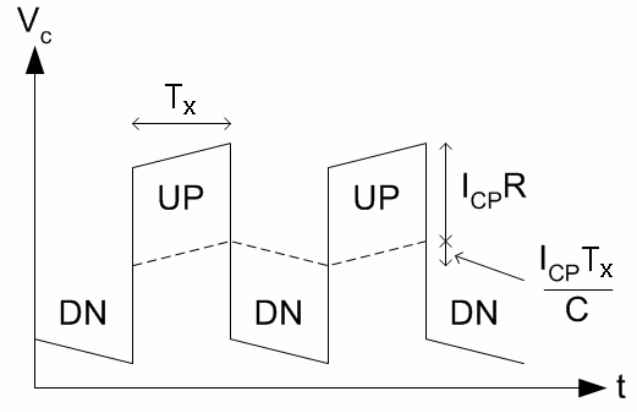


Fig. 2 – PLL Control Voltage $V_c$

$$\omega_X(t) = \omega_{fb}(t) + k_{VCO}I_{cp}R + \frac{1}{2}\Delta\omega(t) \quad (2.7)$$

Since $\Delta\omega(t)$ is approximately equal to $\Delta\omega`$ regardless of time, and $\Delta\omega`$ is negligible when compared to $\omega_{fb}(t)$, a small error in $\Delta\omega(t)$ is tolerable and it is valid to set $\Delta\omega(t)$ to the constant $\Delta\omega`$ as defined in equation 2.5. Thus the period of the recovered clock during an UP pulse can be given by:

$$T_{X\_up}(t) = \frac{2\pi}{\omega_{fb}(t) + k_{VCO}I_{cp}R + \frac{1}{2}\Delta\omega`} \quad (2.8)$$

Or equivalently:

$$T_{X\_up}(t) = \frac{2\pi}{\omega_{fb}(t) + \frac{\Delta\phi`}{T_{ref}}} \quad (2.9)$$

Similarly, the period of the recovered clock during a DN pulse is given by:

$$T_{X\_dn}(t) = \frac{2\pi}{\omega_{fb}(t) - \frac{\Delta\phi`}{T_{ref}}} \quad (2.10)$$

This section has presented equations that model the behavior of a bang-bang PLL. A simplified version of this model is successfully used in [4] to derive an expression for PLL capture range.

## III. Modeling Non-Idealities

The non-ideal behavior of the components comprising a PLL system can cause system behavior to deviate significantly from the ideal. This is particularly true of bang-bang PLL systems, due to their chaotic nature during acquisition. PLLSim models the effects of some of the major non-idealities in bang-bang phase locked loops: non-linear VCO response, non-zero phase detector deadzone, and non-zero phase detector latency.

Any arbitrary VCO response can be entered into PLLSim. The VCO response is used calculate VCO gain given PLL frequency error. This gain is used for the calculation of $\Delta\varphi`$ and $\Delta\omega`$ in equations 2.5 and 2.6. Phase detector deadzone is simply modeled by suppressing UP and DN pulses if the system phase error is less than the specified. Modeling phase detector latency is more involved.

Phase detector latency describes the time between when a bang-bang phase detector measures the sign of a system's phase error, and the time it asserts either an UP or DN pulse. This latency is measured in fractions of a cycle period of the recovered clock, and is dependant on the architecture of the phase detector. Fig. 3 and Fig. 4 show circuit and timing diagrams for an Alexander phase detector [5] and a three times over-sampling bang-bang phase detector.

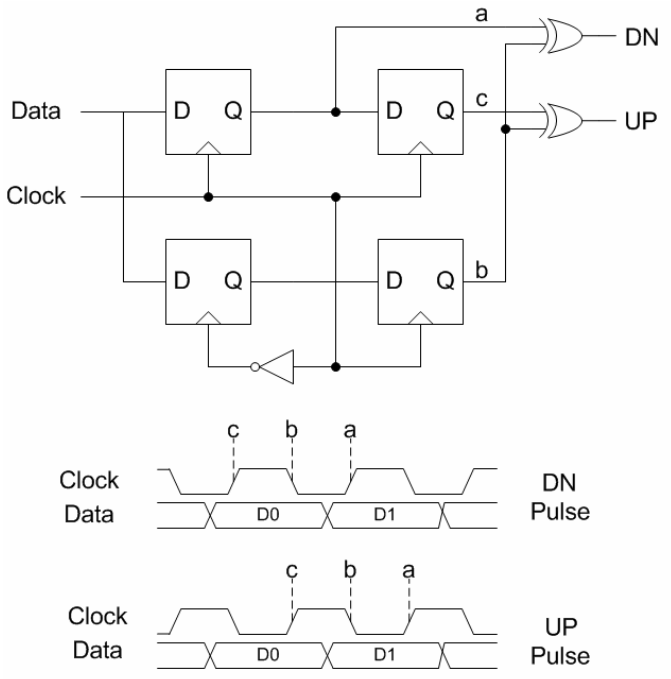


Fig. 3 – Alexander Phase Detector and Timing

The Alexander phase detector samples the data three times using both edges of the clock. If the data is determined to have transitioned between clock edges *a* and *b*, a DN pulse is generated, otherwise an UP pulse is generated. This forces the falling edge of the clock signal to line up to the instant data transitions. The sign of the phase error measured is valid at the instant edge *b* occurs, but the corresponding control pulse is generated at edge *a*. Thus an Alexander phase detector should be modeled with half a clock cycle of latency. The three times over-sampling phase detector in Fig. 4 synchronizes data to the rising edge of clock120, but generates control pulses with the rising edge of clock240. This phase detector would be modeled with a 120 degree, or one third of a clock cycle latency.

```
while time < end_time {
    calcGainVCO()
    calcParams()
    If (lastPhaseErr(γ) > deadzone) &&
        (rand() < transition_density) {
        if (Φerr < 0) { // UP pulse
            TX = 2 * π / (ωref - ωerr + ΔΦ` / Tref)
            Φerr = Φerr + calcPhaseStep() + ωerr * TX
            ωerr = ωerr + calcFreqStep()
        } else { // DN pulse
            TX = 2 * π / (ωref - ωerr - ΔΦ` / Tref)
            Φerr = Φerr + phaseStep() + ωerr * TX
            ωerr = ωerr + freqStep()
        }
    } else {
        TX = 2 * π / (ωref)
        Φerr = Φerr + ωerr * TX
    }
    store_data()
    time = time + TX
}
```

Fig. 5 – PLLSim Operation

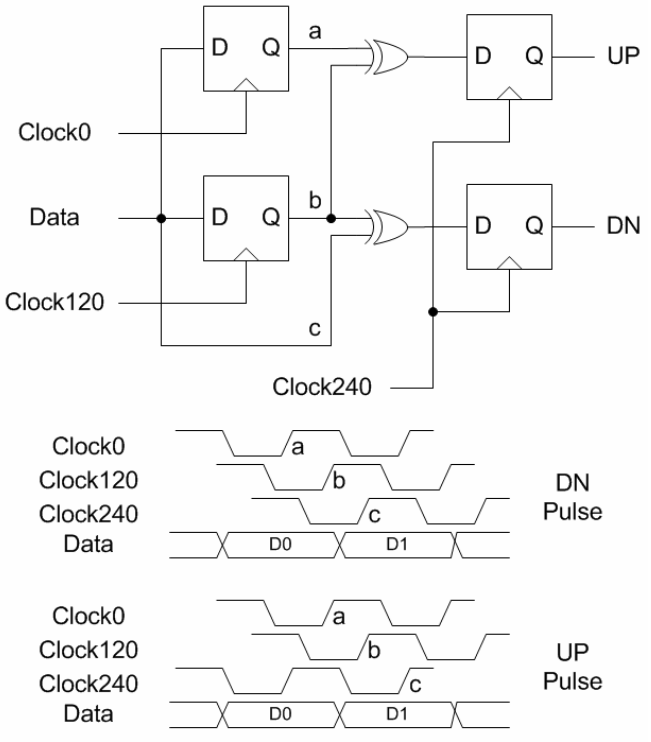


Fig. 4 – Three Times Over-Sampling Phase Detector and Timing

Phase detector latency is an important parameter to model correctly, since it has a direct impact on the magnitude of PLL jitter due to the bang-bang control – the larger the latency, the larger jitter is in lock. Non-zero phase detector latency also results in smaller PLL capture range, and increased locking time.

To model this latency, PLLSim calculates the system phase error γ cycles before the current clock tick (where γ is the phase detector latency as a fraction of a clock cycle), and uses this value to determine whether to generate an UP or DN pulse. This phase error is approximated by linearly interpolating between the phase error at the current clock tick, and the previous clock tick.

## IV. PLLSim Algorithm

Fig. 5 outlines the algorithm PLLSim uses to simulate bang-bang PLL behavior. Each iteration of the algorithm corresponds to a clock cycle of the recovered clock signal. The function *calcGainVCO()* is responsible for calculating the gain of the VCO given its frequency. *calcParams()* uses the result of *calcGain()* to determine values for system parameters $\Delta\varphi`$ and $\Delta\omega`$. The function *lastPhaseErr()* calculates the phase error given a phase detector latency of γ cycles, and *rand()* returns a floating point number from zero to one. Finally, *calcPhaseStep()* and *calcFreqStep()* return values for $\Delta\varphi(t)$ and $\Delta\omega(t)$ according to equations 2.3 and 2.4.

## V. Software Interface

PLLSim was written in Visual Basic dot Net. Particular care was taken to create a user friendly and efficient interface. Fig. 6 shows the main window of PLLSim, and the plot setup

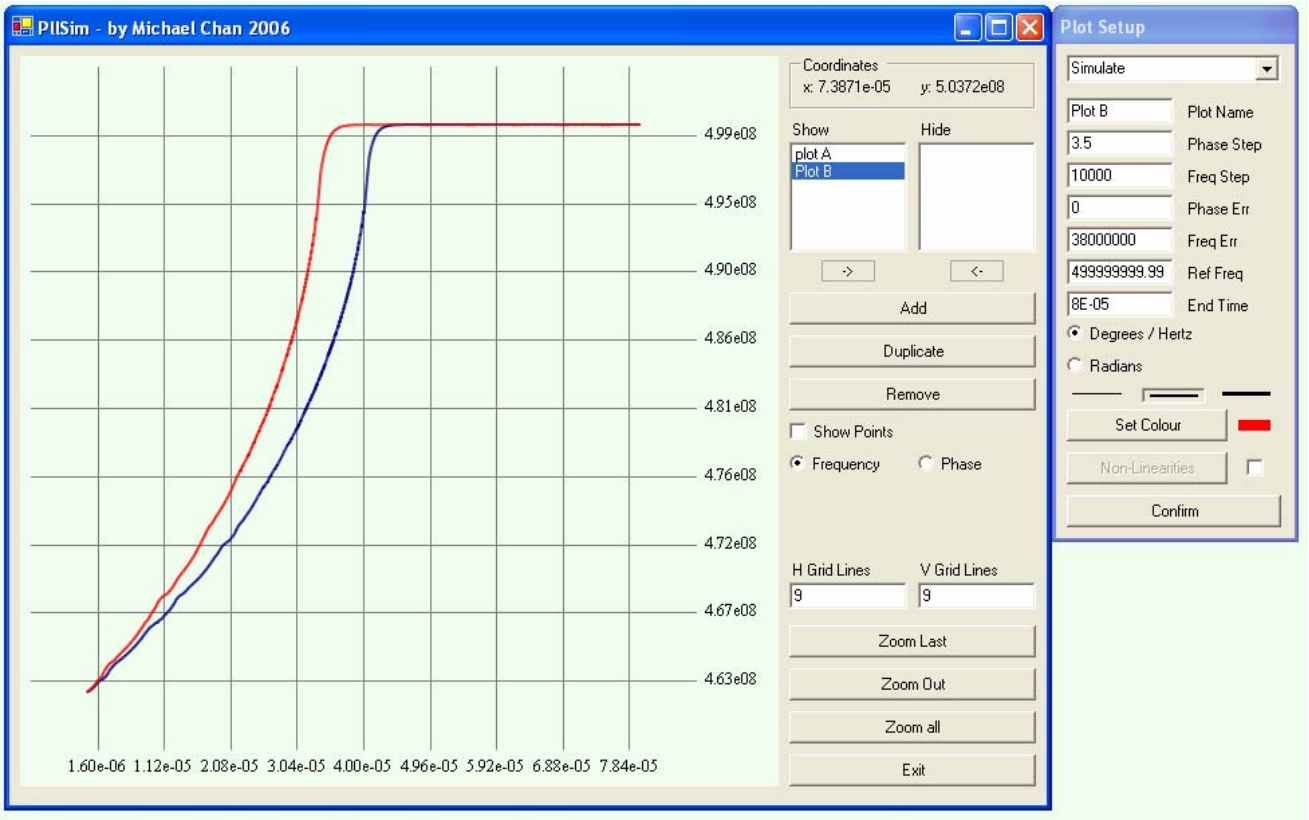


Fig. 6 – PLLSim Interface

window along side it.

The *add button* is used to create a new plot. The new plot appears in the *Hide list*, and is highlighted. Using the *<- button*, the plot can be moved to the *Show list*. Plots in the *Show list* are drawn on the main form. The *-> button* can similarly be used to move plots from the *Show list* to the *Hide list*. The *Duplicate button* is a convenience that allows one to create a new plot with the same plot configuration as the selected plot. This is useful for creating a plot with a similar configuration to an already defined plot.

Only one plot in the *Show list* and *Hide list* can be selected at a time. The selected plot is configured via the plot setup window. *Phase Step* and *Frequency step* correspond to the PLL parameters $\Delta\varphi$` and $\Delta\omega$`, *Phase Err* and *Freq Err* specify the initial state of the PLL, *Ref Freq* corresponds to the frequency of the reference signal ($\omega_{ref}$), and *End Time* specifies how long to simulate the PLL for. The appearance of the plot can be configured by setting the line thickness, and the color. The plot color selected is indicated via a colored square next to the Set Color button. The *confirm button* saves the plot configuration, and also runs the simulation. Due to the speed of the algorithms used, the simulation data is generated almost instantaneously. Finally the list box in the plot setup window specifies the plot type to be *Simulate*. The plot type can also be set to *File*. If *File* is selected, raw plot data points can be loaded from file, instead of being generated via simulation. This is convenient for benchmarking the results of PLLSim against other simulation tools, or just for taking advantage of the graphing capabilities of PLLSim. The *Degrees / Hertz* and *Radians* radio buttons determine the units used to interpret the plot setup data.

Fig. 7 shows the form used to describe the non-ideal effects that PLLSim models. This window is accessible via the *Non-Linearities button*.

The first *VCO Response Listbox* allows one to describe any arbitrary VCO response. (X, Y) points can be added to or removed from the list box; the X value refers to the VCO frequency normalized to the reference frequency, and the Y value is the amount the VCO gain should be scaled by. The phase deadzone can be modeled by entering a value in degrees into the *deadzone textbox*. If the phase error is less than the deadzone, the PLL phase detector will not generate an UP or DN pulse. Pseudo-random data sequences can be described by entering a value in the *Transition Probability textbox*. The transition probability of the reference data signal depends on how the signal is encoded. For instance, a truly random data signal would be modeled with a transition density of 0.5. Finally, phase detector latency is specified in this form.

Checking the *Show Points checkbox* causes plots to be drawn with the raw data points highlighted as shown in Fig. 8. Fig. 8 shows a plot of phase in degrees vs. time. The radio buttons *Frequency* and *Phase* determine whether phase or

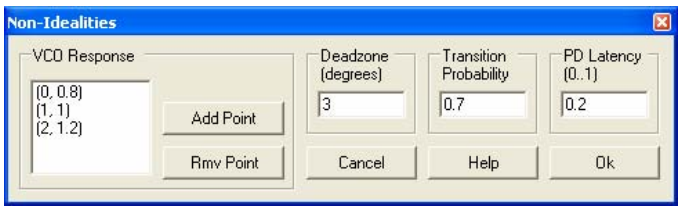


Fig. 7 – Configuring Non-Idealities

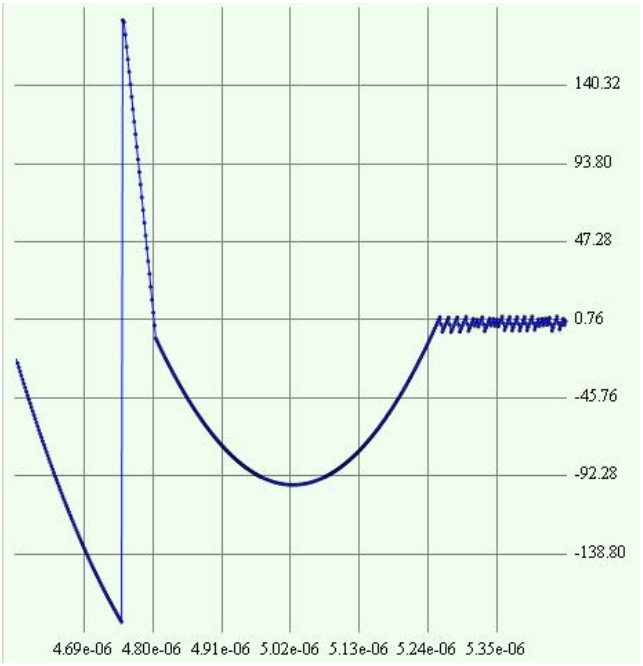


Fig. 8 – Phase Plot with Raw Data Points Shown

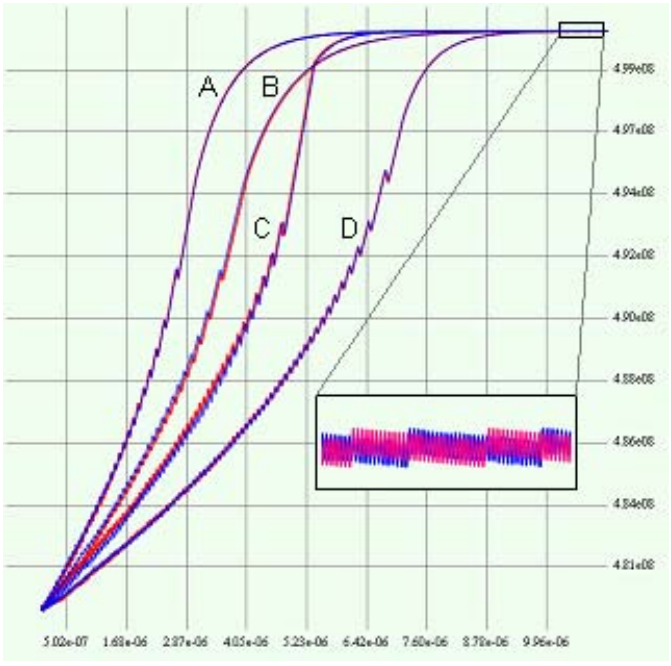


Fig. 9 – Transient Simulation Results

frequency is graphed.

PLLSim also provides comprehensive zooming functionality, which facilitates analysis of simulation results. Zooming in is achieved by using the mouse to draw a rectangle in the draw area. The draw area will display the contents of the rectangle. The *Zoom Out* button increases the plot area viewed by 10%. PLLSim also keeps a history of all views. The *Zoom Last* button reverts the zoom to the previous view. Finally, the *Zoom All* button adjusts the zoom such that all plots in the *Show List* will be completely displayed.

## VI. Analysis of Results

A bang-bang PLL model was created in Matlab Simulink, in order to validate the results produced by PLLSim. The Simulink model incorporated an Alexander phase detector, and modeled the charge pump and loop filter using a linear transfer function block. An ideal VCO was used in the model, and the transition density of the reference data signal was set to one. Simulations were run in both Matlab Simulink, and PLLSim using the configurations listed in table 1. The results of these eight simulations are shown in Fig. 9. As can be seen, the results generated by Matlab Simulink and PLLSim closely overlap each other in all four cases tested.

Analyzing the waveforms in lock shows that PLLSim produces the exact same jitter characteristics as Matlab Simulink; this also confirms that phase detector latency is correctly modeled. When far from lock, close analysis reveals small variations between the results PLLSim generates and the results obtained from Simulink. It was determined that for certain phase and frequency errors, an Alexander type bang-bang phase detector may generate no control pulse, or both an UP and DN pulse (which is equivalent to no pulse). Specifically, if the recovered clock is running slower than the reference signal, and the system's phase error is close to 180 degrees, then all edges *a*, *b* and *c* (see Fig. 3) will sample the same data resulting in no control pulse. On the other hand, if the recovered clock is faster than the reference signal, then edge *b* may sample a different value to edges *a* and *c*, resulting in an UP and DN pulse at the same time. This phenomenon is correctly modeled in Simulink, but not in PLLSim. As can be seen from Fig. 9, the impact of this difference is minor.

Simulation times were compared for PLLSim and Matlab Simulink. The Simulink simulation was configured with a fixed time step of 10 ns. Table 2 contains the simulation times measured for the four cases in Table 1. The simulations were performed on a Pentium 4 2.8 GHz machine running Windows XP Service Pack 2, with 1 Gb of Ram,

## VII. Future Work

There exists a lot of scope to improve PLLSim to make it a more complete and useful simulation tool. In its current state, it is an extremely effective tool for predicting PLL transient behavior, and analyzing jitter due to the bang-bang phase

Table 1 – Simulation Configurations

| Plot | $\Delta\Phi$` | $\Delta\omega$` | $\omega_{ref}$ | $\omega_{err}(0)$ | $\Phi_{err}(0)$ |
|---|---|---|---|---|---|
| A | 5° | 20kHz | 500MHz | 20MHz | -90° |
| B | 5° | 15KHz | 500MHz | 20MHz | -90° |
| C | 3° | 20KHz | 500MHz | 20MHz | -90° |
| D | 3° | 15KHz | 500MHz | 20MHz | -90° |

Table 2 – Performance Comparison

| Simulation Configuration | Simulation Time | PLLSim Time | Simulink Time | Speedup |
|---|---|---|---|---|
| A | 20 us | 5.9 ms | 389 s | 6.95e04 |
| B | 30 us | 8.8 ms | 730 s | 8.30e04 |
| C | 20 us | 5.9 ms | 403 s | 6.83e04 |
| D | 30 us | 8.7 ms | 697 s | 8.01e04 |

control of the system.

We current plan to extend PLLSim's modeling capabilities to allow more flexibility in specifying the input signal. Specifically, rather than a data stream with fixed frequency, we plan to model frequency modulated data. This will be particularly useful for studying the tracking performance of different loop parameters. We also plan to include options to model noise in the reference signal, and noise sources in the PLL's individual components; the PLL VCO in particular is a significant contributor to overall phase noise. Finally, the speed advantage of PLLSim makes fast Monte-Carlo simulation possible. This is of particular benefit in understanding PLL behavior when far from lock, since PLL behavior in this region can appear chaotic. Future versions of PLLSim will include options for configuring Monte-Carlo simulations.

Current and future versions of PLLSim can be downloaded from [9].

## VIII. Conclusions

This paper has presented a new PLL simulator especially targeted at bang-bang type phase locked loops. This simulator is based on a system level model of PLL operation, and consequently can run orders of magnitude faster than a time step simulator such as Matlab Simulink. In the cases analyzed, PLLSim was able to generate results in just a few milliseconds. Matlab Simulink required several minutes to produce the same results. Careful comparison with Matlab Simulink has demonstrated that PLLSim produces correct and accurate results. The difference between the two sets of results was negligible, and attributed to a quirk in the PLL's phase detector that PLLSim did not model. The extremely fast simulation time of PLLSim, coupled with the user friendly controls also make it an attractive tool for studying and gaining insight into bang-bang PLL operation.

PLLSim was extended to model various non-idealities common to PLL systems. Future work aims to increase the modeling capabilities of PLLSim to make it a more complete and useful simulation tool.

## References

[1] M. Ramezani, C. Andre, and T. Salama, "A 10Gb/s CDR with a half-rate bang-bang phase detector," Circuits and Systems, 2003. ISCAS '03. Proceedings of the 2003 International Symposium on, 2003.

[2] H. Nosaka, K. Ishii, T. Enoki, and T. Shibata, "A 10-Gb/s data-pattern independent clock and data recovery circuit with a two-mode phase comparator," Solid-State Circuits, IEEE Journal of, vol. 38, pp. 192-197, 2003.

[3] J. Lee, K. S. Kundert, and B. Razavi, "Analysis and modeling of bang-bang clock and data recovery circuits," Solid-State Circuits, IEEE Journal of, vol. 39, pp. 1571-1580, 2004.

[4] M. Chan, A. Postula, Y. Ding, and L. Jozwiak, "A bang-bang PLL employing dynamic gain control for low jitter and fast lock times," Mixed Design of Integrated Circuits and Systems (MIXDES'2005), Proceedings of the 12th International Conference on, pp. 23-27, 2005

[5] B. Razavi, "Challenges in the design high-speed clock and data recovery circuits," Communications Magazine, IEEE, vol. 40, issue. 8, pp. 94-101, 2002

[6] M. H. Perrott, "Behavioral Simulation of Fractional-N Frequency Synthesizers and Other PLL / DLL Circuits," Design Automation Conference, 2002. Proceedings. 39th, pp. 498-503, 2002

[7] X. Lai, Y. Wan, J. Roychowdhury, "Fast PLL Simulation Using Nonlinear VCO Macromodels for Accurate Prediction of Jitter and Cycle-Slipping due to Loop Non-Idealities and Supply Noise," Design Automation Conference, 2005. Proceedings of the ASP-DAC 2005, pp. 459-464 Vol. 1, 2005

[8] A. Demir, E. Liu, A. L. Sangiovanni-Vincentelli, I. Vassiliou, "Behavioral Simulation Techniques for Phase / Delay-locked systems," Custom Integrated Circuits Conference, 1994., Proceedings of the IEEE 1994

[9] M. Chan, "PLLSim - A Bang-Bang Phase Locked Loop Simulation Tool," (2006). [Online]. Available: http://www.itee.uq.edu.au/~mchan