A Framework for Designing Reusable Analog Circuits

Dean Liu, Stefanos Sidiropoulos†, and Mark Horowitz

Computer Systems Laboratory, Stanford University
Stanford, CA 94305
† also with Aeluros Inc., Mt. View, CA

ABSTRACT

Recent analog design tools have started to allow designers to archive not only the sized schematics but also some of the objectives that the circuit is trying to achieve. This paper first describes STAR (Schematic Tool for Analog Reuse), a system that captures designer's knowledge as part of the archival circuit representation, and then describes how this system can be used to create portable design modules. Creating portable analog modules require more than just the optimization criteria for the cell. It must also include the constraints on the cell's environment (for proper operation), and how these constraints should scale with technology. Furthermore, the system must help the designer in the current task of creating the design, since it is rare that a designer thinks about creating IP for someone else. We demonstrate the capability and utility of this system by examining the reuse of a phase-locked loop.

1. INTRODUCTION

Analog circuit design is usually a repetitive process iterating between circuit synthesis and design verification. During circuit synthesis, the designer chooses a circuit topology, formulates analytic equations, and then adjusts transistor sizes as the result of local optimizations. Tightly coupled to the synthesis process is design verification which ensures that under all conditions the circuit meets the performance envelop set by the target application. Typically, the design verification is accomplished through using a circuit simulator like SPICE [1], or one of its derivatives. Running a circuit simulation requires three kinds of input - the circuit to be simulated, which is normally a sized circuit schematic, the simulation control file, which is the description of what inputs should be applied to the circuit, and the measurement file, which is the description of what data should be collected and how it should be analyzed. The algorithms implemented by these pre- and post-processing runs contain the implicit optimization goals and heuristics that capture the designer's thought process and are a critical part of the design record. Often these additional files are lost, or poorly documented, so when another designer attempts to reuse this design, she/he needs to reconstruct what the critical design issues are and then recreate the control and measurement files. The net result is that designs get more brittle with time, as information about the original issues is lost.

Recently there have been a number of tools that attempt to help automate this analog design process, allowing the designer to state some of the optimization objectives at a higher level and automatically generate some of the simulation control and measurement files that are needed. In fact some of the recent tools use sophisticated optimization engines to automate the design/validate loop. While these tools help with the documentation problem, by allowing the user to specify some of the design objectives at a higher level, they address only some of the issues with design reuse.

This paper describes an annotated schematic capture system that focuses on the design reuse problem. First, like some of the newer analog design tools, it integrates into the schematic capture system a method of specifying the desired circuit's optimization objectives. In addition, this tool allows the user to specify constraints on the circuit as well. These constraints are checked every time the circuit is used, and any violations are reported to the user. The support for optimization and constraints in this tool is extensible - the user can easily add new types of constraints of optimization goals. Using this system we explore how to code circuits to make them portable, and how to encourage designers to create circuits in this form.

We use a phase-locked-loop (PLL) design as a concrete example of analog/mixed signal design, so the next section briefly reviews PLL design, and some of the important issues that need to be addressed. Section 3 describes some of the current systems for allowing a designer to enter higher-level constraints on a schematic. We leverage these ideas to create a more flexible system, STAR, which is described in Section 4. This system introduces the notion of "active comments", which are schematic annotations which control simulation files. The active comments are broken into two main forms - measurement comments, which are similar to other integrated systems, and constraint comments, which are like assertions in normal programs. Unfortunately, building a circuit with active comments does not make it portable. Section 5 addresses these issues, and gives some guidelines for creating robust portable modules. These issues are explored in a PLL design, and portable and non-portable active comments are shown. Section 6 describes the results of using STAR to help port and re-optimize this PLL design. While not perfect, our overall approach seems promising, Section 7 summaries the advantages of our approach and issues that still need to be resolved in future systems.

2. PLL DESIGN

Phase-locked loops (PLL) are often found on contemporary digital and mixed-signal VLSI systems. For example, PLLs are used as clock generators in all the microprocessors designed today and as clock recovery circuits in many I/O subsystems. The main goal of a PLL is to create an internal clock with precise timing relations relative to an external reference. This is usually accomplished by using a feedback system to compensate for process, voltage and temperature (PVT) variations. While there are many design objectives for a PLL, the primary ones usually focus on the quality of the generated clock, specifically, phase offset and jitter. Phase offset is the "DC" error in the timing of the output clock and jitter is the "AC" phase noise. Many circuit parameters affect phase offset and jitter, and we design the circuits to minimize their effects.

Fig. 1 shows the block diagram of the example PLL which was

ICCAD'03, November 11-13, 2003, San Jose, California, USA.

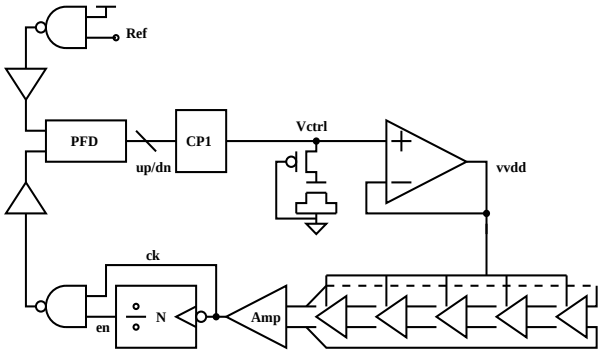

Figure 1: PLL Block Diagram

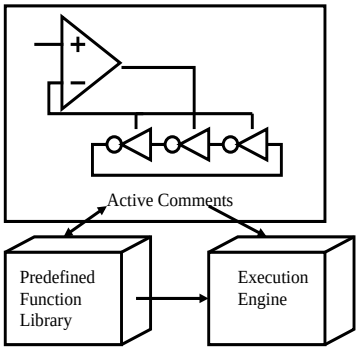

Figure 2: Operation of Active Comments

implemented in a 0.35μm technology and follows the architecture proposed in [2]. Since jitter is often dominated by power supply coupling into the voltage controlled oscillator (VCO), this design uses a regulator to both control the oscillation frequency and to isolate power supply noise. The VCO is composed of two coupled five-stage inverter rings to generate differential multi-phase clocks. Following the VCO is a two-stage amplifier that converts the low-swing VCO output to full CMOS levels. The programmable divider outputs a pulse that fully overlaps the high phase of the VCO clock to qualify the clock when the number of rising clock edges reaches the divide value. The clock buffers after the qualifying NAND gate complete the clock feedback path. The phase-frequency detector (PFD) compares the feedback clock and the reference clock and controls the charge-pump to dump either positive or negative charge onto the capacitor. The matching of delay between the clock buffers and buffers in the feedback path is critical for low phase offset, as is the design of the PFD.

Finally, since a PLL is a feedback system, we need to ensure that the loop is stable. This loop has two poles at the origin: one is the integrator at the output of the charge pump, and the other arises in the VCO (the loop measures the output phase of the VCO, but controls its frequency). To make the loop stable, we add a zero to the loop filter by adding a series resistor. To achieve a wide operating range, this stabilizing zero must not only compensate for the PVT variations but also track the operating frequency. The PMOS transistor in series with the loop capacitor implements the proportional control (stabilizing zero). The PMOS resistor is biased with Vctrl to allow the loop dynamics to track across a wide operating frequency [3].

3. RELATED WORK

Many CAD tools have been developed that address a part of the analog design problem. Recent design capture tools take a more integrated approach to the design process. The Synopsys CosmosSE[4] merges the schematic capture tool, the circuit simulator, and the results analyzer into one environment to allow the designers to specify the stimuli and the analysis routines directly in the schematics. The Cadence Analog Design Environment[5] extends this approach by providing an interactive circuit simulation environment and a scripting language to make it easier to specify the pre- and post-processing routines. The designers can use prebuilt measurements to construct their own. The Agilent ADS[6] extends this integrated approach even further by allowing the user to cascade the measurements from different schematics. These design capture systems make it easier to specify the pre- and post-processing routines. However, when these circuits are instantiated in a higher level of the design hierarchy, only the circuit topology and sizes are propagated to the top level, but not the checks. Consequently, some of the knowledge that we are trying to capture is lost.

We want to design analog circuits and encapsulate the design information as part of the circuit representation such that the circuit becomes a portable module. We built a design capture system to make it easier for a designer to encode constraints for a new circuit, one which the tool and tool's designer had not seen before. From the capture tools, we leverage the notion of an integrated design document, but we loosen the requirement of having an integrated tool set. We extended the scripting approach to provide a more generalize way to specify the pre- and post-processing routines. The user specifies these routines directly on the schematic as comments. We found these annotations generally fall under two classes. One is the traditional optimization/measurement comments, and the other represent constraints on operating conditions. We describe our system in the next section.

4. STAR

A widely used approach for annotating an analog design is to add comments to the schematics to clarify the designer's intention. One problem with writing these comments is that once they are written they are not always updated to reflect the latest implementation. Our approach to address this problem is to make the comments part of the implementation. In the context of analog design process, we would like to turn these comments in the schematics into executables to automate the generation of simulation and data collection scripts. This way, in addition to annotating the design, these comments also aid the designers in doing their tasks, thereby offering incentives to the designers to use these comments. We call these comments *Active Comments*.

The basic operation of the Active Comments is illustrated in Fig. 2. The designer adds the comments into the circuit schematic. These comments are composed of predefined functions from a library, and an execution engine processes these comments to generate the necessary routines. Our goal is to create a flexible library which can be extended by user specified functions. These user supplied extensions can then be archived along with the annotated schematic, enabling design reuse.

The Active Comments in our developed framework can be broadly categorized into two types: Measurements and Assertions. A Measurement Comment sets up the simulation runs and measures the circuit parameters. It also allows the user to formulate analytical expression and evaluate the expression based on the simulation results. An Assertion Comment monitors the simulation outputs to prevent the circuits from operating outside the constraints. The following sections will describe these two types of comments in more detail.

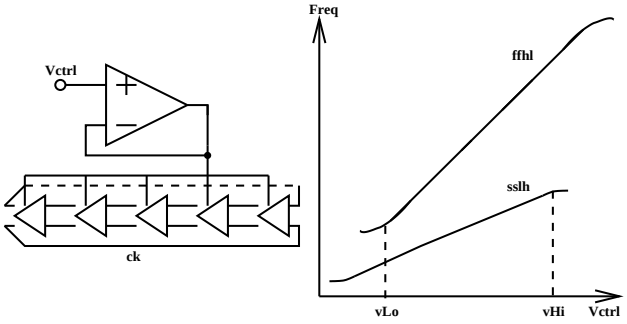

Figure 3: VCO Circuit and Transfer Curve

4.1 Measurements

Measurement Comments basically consist of three parts. The first part specifies what simulation to run and what operating conditions to use, the second part describes how to analyze the simulation results, and the third part allows one to name some of the results so they can be used for other routines.

We use Measurement Comments to find critical circuit parameters. An example of such a parameter is the operating range of a voltage-controlled oscillator (VCO). Fig. 3 shows a representative circuit of a CMOS inverter-based VCO along with two typical oscillation frequency vs. control voltage transfer curve at the extreme corners. To keep the PLL's behavior predictable by a linear control model, the operating range is defined to be the linear part of the curve. We would like to find the upper and lower bounds of the control voltage by sweeping the control voltage across the entire supply range, plotting the transfer curve and finding the inflection points:

```
# DEFINE vLo=LoRange(vctrl,Freq(ck)) w/ \
   SweepV(vctrl,gnd,vdd) @ ffhl

# DEFINE vHi_c=HiRange(vctrl,Freq(ck)) w/ \
   SweepV(vctrl,gnd,vdd) @ sslh
```

We use keywords, $\#\ DEFINE, w/$, and @, to separate the different fields in the comments, and the fields are processed in reversed order, starting from the end. The comment specifies the process corner, $ffhl$ ($sslh$) to find the lower (upper) inflection point of the curve. The pre-processor, $SweepV()$, is a function that sweeps the control voltage, $vctrl$, from gnd to vdd, while the post-processing function, $LoRange()$ ($HiRange()$) finds the lower (upper) inflection point of the VCO transfer curve. The voltage levels at the inflection points are assigned to the two variables, vLo and vHi. These variables are used in other Active Comments to control other PLL simulations.

Using Measurement Comments allows a designer to specify simulation runs and analysis routines in the circuit's schematic representation. Furthermore, it encourages the simulation scripts to be up-to-date with the circuit implementation since the designer is actively using these scripts. To provide further incentives, regression tests could be run on all circuits as part of the design to find stale files. Finally, circuit schematics convey relevant design information which other design tools might be able to leverage.

In fact the new analog synthesis tools would provide an addition capability for the Measurement Comments, since they automate the sizing of transistors in a given analog circuit by performing complicated multi-variable optimizations. The downside to these powerful tools is that the interface is more complex, making it harder for designer/users to add new optimization scripts. For example in convex optimization [7][8], the analog circuits objectives and constraints must be in a specific form for the optimization to work. This task is currently done by experts within the vendor company. While simulation based optimization, [9], [10], [11] and [12], can handle more general constraints, care is still required in the problem setup. Similar to STAR, these synthesis tools also capture the designer's knowledge in some way in order to optimize the circuit. However, the objective of the synthesis tools is to generate optimally sized circuits, while we focus on finding ways to make it easier for designers to create the simulation and analysis procedures to simplify the measurement of circuit parameters. As a result, our system is less powerful but can be more easily used to capture different classes of circuits. Thus our plan is to integrate these synthesis tools into our system to create more powerful library operations that the designer can use, in additional to the simpler operations we have already implemented. For established blocks, there might be one optimize/measure comment that essentially generates the entire circuit.

4.2 Assertions

In addition to optimizing the circuit parameters, a designer must also check that circuit components always satisfy their operating constraints. For example, transistors acting as current sources need to be in the saturation region, noise on the VCO's bias voltage should be minimized, and delay matching circuits must maintain certain timing margin.

While designing each component, the circuit designer must guarantee that the component functions correctly under the worst possible simulation corner. However, as each component is integrated into different parts of the system, some of the assumptions about the operating conditions might be violated. As a result, all the checks must propagate from the component level to each instance of this component in the design. The purpose of embedding the assertion comments in the design is to tie the operating constraints to the design itself. So, the designer only needs to embed the checks on the properties of interest, and these assertions will monitor all the properties in all the simulations, across the system hierarchy.

An example of an assertion is the saturation margin check. Monitoring the saturation margin of a current source can warn the designer when the transistor is about to leave the saturation region. The assertion to monitor the saturation margin of a transistor can be coded as follows:

```
# ASSERT SatMargin(Mtail) >= 0.05 * Vdd
```

where $\#\ ASSERT$ denotes an assertion, and $SatMargin()$ is a predefined function. The $SatMargin()$ function first finds the transistor named $Mtail$ in the schematic, then, in a pre-processing step, generates the appropriate spice probe statements, and finally produces some post simulation directives to analyze the simulation waveforms. The circuit designer can then include the generated file when running the simulation and use the post-processing directives to analyze the results. The right hand side of the inequality sets the margin to be 5% of the supply voltage. When a transistor's $V_{ds} - V_{dsat}$ is less than $0.05 * Vdd$, the assertion prints out a failure message.

Embedding assertions in every simulation can warn the user when margins are violated and prevent the circuit from operating outside the designed limits. This check can spot errors that are missed when only functionality is tested. Often when the constraints are violated, the circuit will still function but with reduced robustness. These checks guarantee that all circuits are operating within the conditions for which they are valid. An added benefit of coupling the

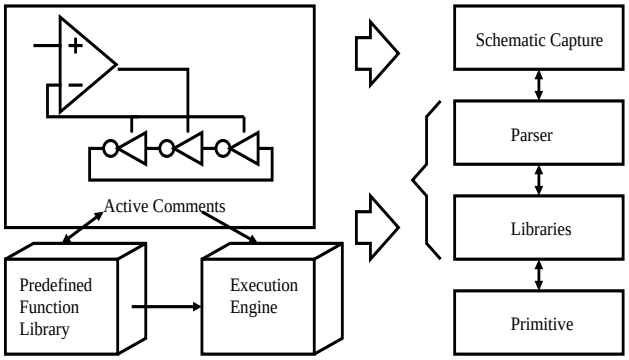

Figure 4: Layering of the Prototype Tool

assertion with the design is that it enables seamless design reuse: when the block is integrated as part of a system by another designer, the same set of checks will be performed automatically.

4.3 Prototype Implementation

The prototype tool is composed of four layers, as shown in Fig. 4. The top-most layer, the schematic layer, implements the schematic user interface written in Tcl/Tk while the middle two layers form the library and the execution engine written in Perl. The primitive layer contains C code optimized for the calculation intensive data processing routines.

Typically, the circuit designers interface only with the schematic and the library layers. For our prototype, we modified a public domain schematic tool, SUE[13], which is written in Tcl/Tk. We found that the while these Active Comments may be easily parsed by a computer program, they require some annotation to clarify their intentions to the other circuit designers. Adding detailed description to each Active Comment in the schematic is prohibitive as it may fill the schematic with text. To address this, we created a new view, comments view, in addition to the typical schematic and iconic views. This view can be edited using a Comments Editor and with it, the user can not only code the Active Comments for the underlying schematic but also enter detailed description about these comments. This way, the comments view becomes a live document for the schematic as the circuit evolves.

In addition to the schematic layer, the designer also interfaces with the library layer. This layer is designed to be flexible and extensible because when the designer creates a new circuit, ones which the tool and tool's designer had not seen before, she/he may need to use functions that are not available in the default library. Consequently, the layer is designed to enable the users to add their own functions to private libraries in the local work area to extend or override the default functions provided by the tool distribution. In a large design group there may be a number of designers each with her/his own function library. In order to avoid conflicting function names, we need to manage the name space of the user provided functions similar to how the schematic names are managed (e.g., prepending unique project and module names to the function name). With the user-provided functions, the design knowledge is split between the schematic and the local function library. Therefore, the designers must check-in both the circuit schematics and the supporting library as part of the archival process.

The functions stored in the library may change depending on the class of circuits being designed. For the type of circuits used to build a phase-locked loop, we were able to construct all the functions needed for our Active Comments based on the set of functions listed in Table 1.

Table 1: Summary of Functions

Name	Description
SetV, SweepV, RampV	Input stimuli to set, sweep, or ramp (for pseudo DC simulations) a voltage
MeasRch, MeasCap	Use simulator to measure transistor's channel resistance, and gate capacitance
V, I	Access raw simulation output
FindDelay, FindSlope, RunningAvg	Process output waveforms to find the delay, slope, or running average

5. PORTABLE COMMENTS

While Active Comments can be used to capture the design knowledge, simply embedding them into analog circuits does not automatically turn these circuits into reusable components. How well a circuit can be reused depends on how portable its design representation is. As a first pass at constraining the design, one may specify the constraints in a way that closely resembles the specification which are typically in absolute values such as seconds, volts, and amps. Unfortunately, as the target process technology changes, some of these constraints may no longer hold true, thus rendering these constraints non-portable. One way to improve the portability of the constraints is to transform these absolute values into relative values by using process independent metrics such as gate delays, supply voltages, and reference currents. Using process independent metrics reduces the dependencies on the technology, but does not convey much information about the underlying circuit. The best way to constrain the circuit is to use some inherent properties of that circuit (i.e., we should use what the circuit is designed to do to constrain the circuit). This constraint then describe the circuit at a higher level and can offer better insight into what the circuit is suppose to do. Ideally, this is the design information we want to archive.

We applied Active Comments to constrain the interface between the PFD and charge-pump of the PLL circuit to show how the comments evolve and become more robust and scalable. Together, the PFD-charge-pump pair (Fig. 5) converts the phase information into voltage. The PFD compares the rising edges of the reference and feedback clock to output a pair of pulses, up and down, such that the difference in the up and down pulse widths is proportional to the phase error. The charge-pump dumps charge onto or takes charge away from the loop filter capacitor during the up and down signal pulses.

To reduce static phase offset, we want to overlap the up and down

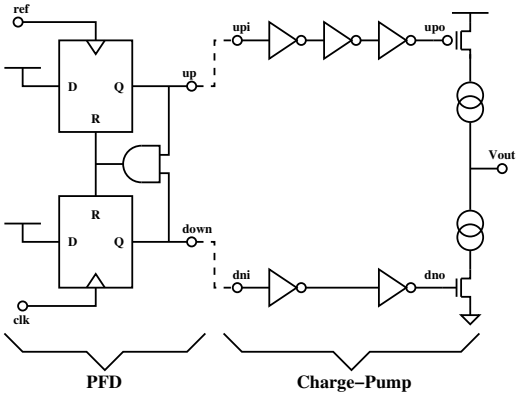

Figure 5: PFD-Charge-pump Diagram

currents while minimizing the current pulse widths to reduce the effect of current mismatches. To achieve this, a charge-pump often has tuned delay-matching inverter chains to shape the up and down control signals. However, if the up and down pulses from the PFD are too narrow, they will disappear in the inverter chains because of the limited bandwidth of the inverters. From simulation, we found that if the pulse width is less than 500ps, then the pulses will disappear in the inverter chain under some process corners. To ensure the pulses are wide enough, we constrain the pulse width to be at least 500ps as follows:

ASSERT PulseWidth(up) <= 500ps

While the 500ps constraint correctly bounds the pulse width for the up and down control signals for the 0.35μm technology, that value is too wide for technologies with finer geometries. To improve the scalability of this assertion, we normalize the pulse width to the delay of a fanout-of-four (FO4) inverter, and transform the constraint to be 3*FO4:

ASSERT PulseWidth(up) <= 3*FO4

Recoding this assertion using gate delays reduces the its dependency on the process technology and thus improves the scalability of the comment. We can update the value of these metrics as part of the technology calibration step.

By examining what the comment is trying to constrain, we may be able to further refine it. It would be best if we can constrain the circuit using its inherent properties. In this example, we are interested in making sure that if there is a pulse at the input of the inverter chain, a pulse appears at the output of the chain. Taking advantage of the fact that the PLL is periodic, we can turn the constraint into checking for a pulse at the output when there is a pulse at the input:

ASSERT PulsePropagation(upi,upo)

By looking for the existence of pulses, the assertion now checks exactly that the circuit is designed to do – propagate pulses through the inverter chain. Coding the comment this way not only offers more insight to the circuit's intended function, but also make the constraint completely independent of the process technology.

Transforming the constraint from absolute value to relative value usually can be done by normalizing the absolute values to a process independent metric. However, recasting the constraint from a relative value to using circuit properties requires a good understanding of the underlying circuit. It may be unreasonable to expect the designer who designed the original circuit to write all the Active Comments in the scalable form. However with the capability of our tool, we expect other designers to continue to refine the comments as they reuse the designs. Over time, with better understandings of the circuit, the designers can rewrite the comments to be more intrinsic to the underlying circuit, and therefore making the design representation more robust and scalable. To encourage the designers to move along this path, we can enlist the help of other experts to reformulate the constraints as part of the design review process.

6. RESULTS

We wanted to reuse our example PLL as as a building block to synchronize the data transmission and to recover the clock and data in an optical transceiver testchip, fabricated in a 0.25μm process. In addition to porting our PLL from 0.35μm to 0.25μm, we also needed to increase the maximum operating frequency while retaining a wide operating range. All the analog circuits need to be re-optimized. In the original design, not all of the Active Comments where written in the scalable form. In order to use the existing Active Comments to guide the porting process, all the absolute constraints were recoded to be more scalable. We were able to constrain many of the comments using the circuit's properties as a result of our improved understanding of the design.

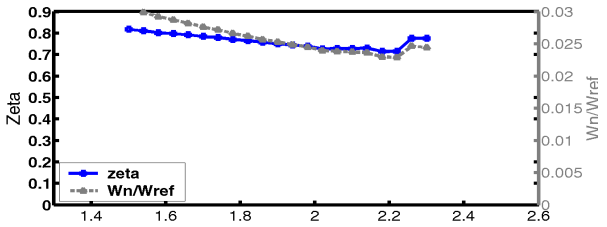

Figure 6: Tracking of PLL Loop Dynamics

Our primary objective in designing the PLL is to reduce jitter and static phase offset. To have small jitter, the PLL needs to have an over-damped response to minimize noise amplification, and a high loop bandwidth to minimize noise accumulation. In terms of the standard loop parameters, this means that the damping factor ζ should be greater than 0.7 and the loop bandwidth ω_n should be close to its theoretical maximum of 10% of the reference frequency. For our example PLL, ζ and ω_n can be expressed as[14]:

$$\zeta = 0.5 * R * \sqrt{I_{CP} * K_{VCO} * C_{CP}/N} \quad (1)$$

$$\omega_n = 2 * \zeta/(R * C_{CP}) \quad (2)$$

Since each term in the loop dynamics equations is implemented by a different sub-block in the PLL, we used the Measurement Comments embedded in each block to find the circuit parameters in the sub-circuit, then evaluated the damping factor and loop bandwidth equations at the system level to help guide our optimization decisions. Fig. 6 shows the resulting loop dynamics tracking across the operating range with ζ always being above 0.7 and ω_n being close to its theoretical maximum.

In addition to automatically measuring and evaluating the critical circuit parameters, STAR is also used to monitor the circuit performance in the background. The tool uncovered a hidden error in the lower hierarchy that causes a static phase error between the internal clock and the external reference. Fig. 7a shows the feedback clock aligned with the reference clock at the PFD input. From these waveforms, one would expect that the PLL is working properly, and all the sub-circuits are performing within the specification. However, the tool found a few assertion failures in the divider. These failures were caused by the logic in the divider not meeting setup time. As a result, the en signal (shown in Fig. 1) is pushed out and arrives after the rising edge of the clock, ck, that it is suppose to qualify, as shown in Fig. 7b. Consequently, there is a static phase error between the internal clock and the reference clock. While the PLL is still functional despite this error, the design margin is nevertheless reduced. Uncovering this error made the ported design more robust.

Using STAR, we are able to successfully port and reuse this PLL design. Simulation results indicate that the operating range of the final PLL is 125-1000MHz. At at 800MHz VCO frequency and N=4, the final design achieves a period jitter of 20ps and peak to peak jitter of $<$ 80ps with 10% step on the power supply.

7. CONCLUSION

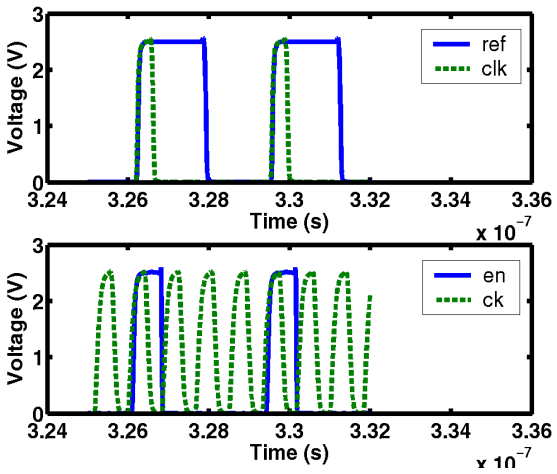

Figure 7: Hidden Errors

Reuse of analog circuits requires the archival representation to be more than just the circuit topology and sizes. The representation must include some design knowledge about how the circuit was designed and what constraints were assumed when the device was simulated. We need to check these constraints not only when designing the circuit in isolation, but also for each instance of the circuit where it is used.

We created a system that addresses many of these issues using Active Comments. This system is flexible enough to allow the designers to easily extend it to capture the constraints of new designs. We were able to capture our PLL using a combination of ten base functions.

The key additional requirement for making reusable blocks is that the metrics in Active Comments should be coded in a process independent way, preferably by using the inherent properties of the underlying circuit. While currently this approach requires some additional thought, we hope if systems like this become more common, designers will become more comfortable with it, and start coding this way naturally.

8. ACKNOWLEDGMENTS

The authors acknowledge the support from the C2S2 Marco Center for supporting this work.

9. REFERENCES

[1] Nagel, L. W., *SPICE2: A Computer Program to Simulate Semiconductor Circuits*, PhD thesis, University of California at Berkeley, May 1975.

[2] Sidiropoulos, S., Liu, D., Kim, J., Wei, G.-Y., and Horowitz, M., "Adaptive Bandwidth DLLs and PLLs using Regulated Supply CMOS Buffers", in *Proc. Symposium on VLSI Circuits*, pp. 124–127, June 2000.

[3] Mansuri, M., and Yang, C.-K. K., "A Low-Power Low-Jitter Adaptive-Bandwidth PLL and Clock Buffer", in *ISSCC Digest of Technical Papers*, pp. 430–431, Febuary 2003.

[4] *CosmosSE User's Guide*, Synopsys, Inc., Mountain View, CA, 2002.

[5] *Analog Design Environment*, Cadence, 2002.

[6] *Advance Design System User's Guide*, Agilent Technology, 2002.

[7] del Mar Hershenson, M., Boyd, S. P., and Lee, T. H., "Optimal Design of a CMOS Op-Amp via Geometric Programming", *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, n. 1, pp. 1–21, January 2001.

[8] Mandal, P., and Visvanathan, V., "CMOS Op-Amp Sizing Using a Geometric Programming Formulation", *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, n. 1, pp. 22–38, January 2001.

[9] Kransnicki, M., Phelps, R., Rutenbar, R. A., and Carley, L. R., "MAELSTROM: Efficient Simulation-Based Synthesis for Custom Analog Cells", in *Proc. Design Automation Conference*, pp. 945–950. IEEE/ACM, June 1999.

[10] Phelps, R., Krasnicki, M., Rutenbar, R. A., Carley, L. R., and Hellums, J. R., "Anaconda: Simulation-Based Synthesis of Analog Circuits via Stocastic Pattern Search", *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, n. 6, pp. 703–717, June 2000.

[11] Kransnicki, M., Phelps, R., Hellums, J. R., McClung, M., Rutenbar, R. A., and Carley, L. R., "ASF: A Practical Simulation-Based Methodology for the Synthesis of Custom Analog Circuits", in *Proc. International Conference on Computer Aided Design*, pp. 350–357. IEEE/ACM, November 2001.

[12] "NeoCircuit", URL http://www.neolinear.com/sections/products_solutions/neocircuit_brochure.pdf.

[13] Tavrow, L., *Schematic User Environment*, Micro-Magic Inc., Sunnyvale, CA, 1994.

[14] Gardner, F., "Charge-Pump Phase-Locked Loops", *IEEE Trans. Come.*, vol. 28, n. 11, November 1980.