Mixed Signal Design Flow
A mixed signal PLL case study

by

Ramin Shariat-Yazdi

A thesis

Presented to the University of Waterloo

in fulfillment of the

thesis requirement for the degree of

Master of Applied Science

in

Electrical & Computer Engineering

Waterloo, Ontario, Canada, 2001

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Mixed-signal designs are becoming more and more complex every day. In order to adapt to the new market requirements, a formal process for design and verification of mixed signal systems i.e. top-down design and bottom-up verification methodology is required. This methodology has already been established for digital design. The goal of this research is to propose a new design methodology for mixed signal systems.

In the first two chapters of this thesis, the need for a mixed signal design flow based on top-down design methodology will be discussed. The proposed design flow is based on behavioral modeling of the mixed signal system using one of the mixed signal behavioral modeling languages. These models can be used for design and verification through different steps of the design from system level modeling to final physical design. The other advantage of the proposed flow is analog and digital co-design.

In the remaining chapters of this thesis, the proposed design flow was verified by designing an 800 MHz mixed signal PLL. The PLL uses a charge pump phase frequency detector, a single capacitor loop filter, and a feed forward error correction architecture using an active damping control circuit instead of passive resistor in loop filter. The design was done in 0.18-μ m CMOS process technology.

Acknowledgment

I would like to take this opportunity to thank my supervisor, Professor Jim Barby, for his help and support throughout the course of this project. I appreciate the opportunity he has given me to work on an interesting project with which I have developed experience in many areas.

I would also like to thank Professor Ajoy Opal and Professor Mohamed Elmasry for reviewing this thesis and Mr. Phil Regier for his support and help while I was working on this project.

Dedication

To Mom and Dad for everything.

To my lovely wife Maryam, for being a real friend , support and help.

To my brother Keyvan, for being a real brother and friend.

Table of Contents

List of Figures

Chapter 1
Introduction

1.1 Importance of design methodology in market share

With the internet and wireless technology as the latest market drivers, the pace of the electronic market place continues to quicken. New products and new product categories are being created faster than ever before. In order to keep up with the rapid pace of the market, designers must get their products to market more quickly than ever. Those that are successful at bringing significant new capabilities to the market first are usually rewarded with higher profit margins and greater market share. To understand this, consider three scenarios for developing a product with Figure 1.1 showing the expected revenue for each scenario [1]. For the first, consider employing an efficient product development process and being first to market. For the second, consider using the same number of developers with an inefficient development process, which causes the product to be late to market. This results in a much lower return because the product enters a market where a competitor has already established leadership position and because there are fewer available customers left. Finally, consider using an inefficient development process but increasing the number of developers in order to reach the market first. If this were possible, the development costs are higher, but the total return is almost the same as in the first case. This is because the returns are expected to be much greater than the initial development costs.

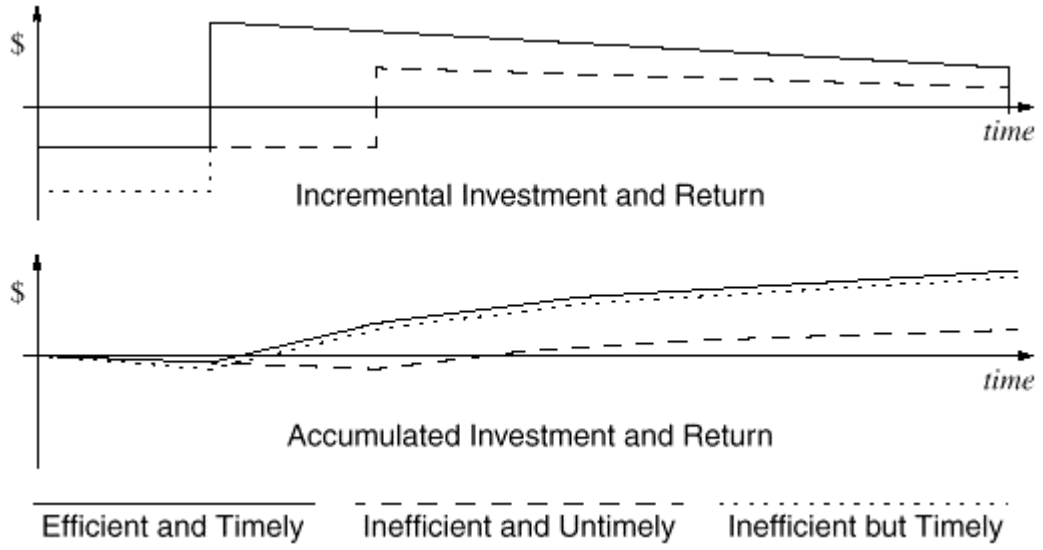

Figure 1.1 Different approaches to design

This example illustrates why it is more important to get a product to the market first than it is to control development costs. Of course this assumes that the product is the right product in that it satisfies the customer's needs, and that it has some new and valuable capability. With follow on products, the situation is somewhat different. Here; the market leadership position is largely determined and the need to develop the product in a timely manner is balanced by the need to control development costs.

Moore's observation that the number of transistors available on an integrated circuit doubles every 18 to 24 months continues to hold. Competitive pressures compel designers to use these transistors to provide additional functionality and to increase the integration level and thereby decreasing the size, weight, power and cost of the product. As a result, designers are confronted with larger and more complex designs. The increasing size and complexity of these designs combines with the shrinking time available to develop and get them to market; making the job of the circuit designer today much more difficult than in the past.

Circuits are getting more complex in two different ways at the same time. First, circuits are becoming larger. Consider wireless products; 40 years ago a typical receiver contained between 5 and 10 transistors whereas it is common for a modern cell phone to contain 10M transistors. Second, the operation of the circuits is becoming more complex. 30 years ago integrated circuits generally consisted of simple functional blocks such as op-amps and gates. Verification typically required simulating the block for two or three cycles. Today, mixed-signal chips implement complex algorithms that require designers to examine their operation over thousands of cycles. Examples include PLLs (Phase Locked Loop), sigma-delta converters and CDMA (Code Division Multiple Access) transceivers.

The CAD (Computer Aided Design) tools and computers employed by designers continually improve, which serves to increase the productivity of designers. However, the rate of productivity increase is not sufficient to allow the designers to keep up with the increasing complexity of designs and decreasing time-to-market requirements. The growing difference between the improvement in productivity needed to satisfy the demands of the market and the productivity available simply by using the latest CAD tools and computers is referred to as the Design Productivity Gap, and is shown in Figure 1.2 . To close this gap, one must change the way design is done. A design style that reduces the number of serial steps, increases the likelihood of first time working silicon, and increases the number of designers that can work together effectively is needed. If a design group fails to move to such a design style, it will become increasingly ineffective.

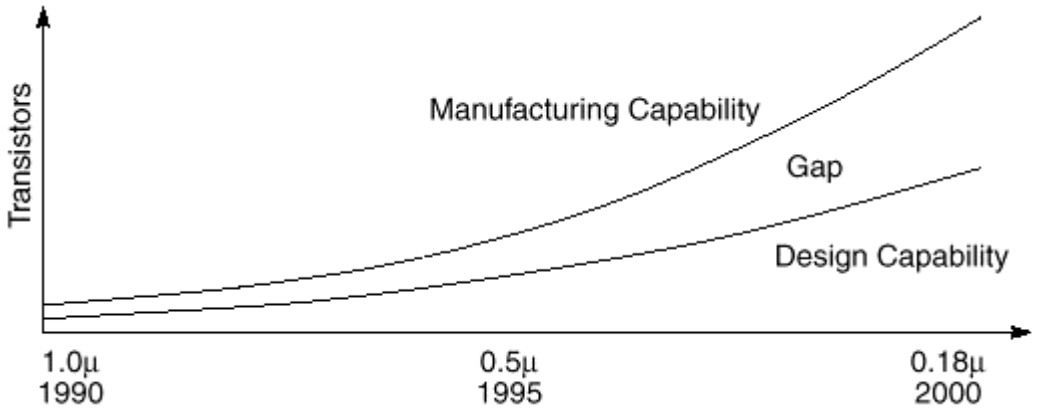

Figure 1.2 IC process technology is improving faster than IC design technology

This involves more than simply a cursory design of the circuit block diagram before designing and then a design of the blocks. Rather, it requires developing and following a specified verification plan and an incremental and methodical approach for transforming the design from an abstract block diagram to a detailed transistor-level implementation. This approach has already been implemented by digital designers and has been used successfully.

1.2 Bottom-up design

The traditional approach to design is referred to as bottom-up design. In this approach, the design process starts with the design of the individual blocks, which are then combined to form the system. The design of the blocks starts with a set of specifications and ends with a transistor level implementation. At this point, each block is verified as a stand-alone unit against specifications and not in the context of the overall system. Once verified individually, the blocks are then combined and verified together. At this point the entire system is represented at the transistor level.

While bottom-up design continues to be effective for small designs, large designs expose several important problems with this approach [2]:

- Once the blocks are combined, simulation takes a long time so verification becomes difficult and perhaps impossible. The amount of verification must be reduced to meet time-to-market goals.
- For complex designs, the greatest impact on the performance, cost and functionality is typically found at the architectural level. With a bottom-up design style, little if any architectural exploration is performed so these types of improvements are often missed.
- Any errors or problems found when assembling the system are expensive to fix because they involve redesign of the transistor-level blocks.
- Communication between designers is critical, yet an informal and error-prone approach to communication is usually employed.
- Several important and expensive steps in the bottom-up design process must be performed serially, which stretches the time required to complete the design. Examples include system level verification and test development [3,4].

1.3 Top-down design

To address the above issues of bottom-up design, many design teams are looking at the top-down design methodology. In a basic top-down approach, the architecture of the chip is defined as a block diagram and simulated and optimized using either an MS-HDL (Mixed-Signal Hardware Descriptive Language) simulator or a system simulator.

From the high-level simulation, requirements for the individual circuit blocks are derived. Circuits are then designed individually to meet these specifications. Finally, the entire chip is laid out and verified against the original requirements

A well-designed top-down design process methodically proceeds from architecture to transistor-level design. Each level is fully designed before proceeding to the next and

each is fully leveraged in design of the next. It acts to partition the design into smaller, well-defined blocks and so allows more designers to work together productively. This tends to reduce the total time required to complete the design. A top-down process also formalizes and improves communication among designers. The formal nature of the communication also allows designers to be at different sites and still be effective.

Following a top-down methodology also reduces the impact of changes that come late in the design cycle. If the circuit has to be partially redesigned, the infrastructure put in place as part of the methodology allows the change to be made quickly. The models can be updated and the impact on the rest of system can be quickly evaluated. The simulation plan and the infrastructure for mixed-level simulations are available and can be quickly applied to verify changes.

1.4 The role of chip architect

The chip architect is the leader of the top-down design process. He or she is expected to develop the simulation and modeling plans and to coordinate with the other designers to make sure that the plans are followed. The primary responsibility of the chip architect is to see that the system operates as expected when finally implemented. This must be a designer who has experience in the type of system being designed so that he or she can anticipate and plan for problems that are likely to occur. Preferably, the person's experience covers aspects of both system and block design.

The chip architect owns the top-level schematic for the design. This schematic must be captured before any block design begins, even though, it is likely to change before the design is complete. The top-level schematic specifies the partitioning of the design into blocks and the interface for each block, so each block should be "pin-accurate", which means, in the top-level schematic each block and each pin on each block is represented and the type of each pin is carefully defined and documented. For example, an enable line on a block may be denoted “3V CMOS active high”. In this way, the top-level schematic provides clarity of intention to the design team [4].

Once the top-level schematic is captured, the top-level models are written and the system is completely verified according to the simulation plan. The top-level schematic and models are then distributed to everyone on the design team. As the design progresses, the chip architect coordinates any changes to the block. As the block designers work, they provide transistor-level schematics (pre- and post-layout) which are verified with mixed-level simulation.

During the design phase, the chip architect works with the test engineers to develop the test plan and test programs. The availability of a working model of the system early in the design process allows test engineers to begin the development and testing of test programs early.

1.5 Simulation and Modeling Plans

An important focus in a good top-down design methodology is the development of a comprehensive simulation plan, which in turn leads to a modeling plan. This is done by the chip architect with input from the whole design team. The process begins by identifying particular areas of concern in the design. Plans are then developed for how each area of concern will be verified. The plans specify how the tests are performed and which blocks are at the transistor level during the test. For example, if an area of concern is the loading of one block on another, the plan might specify that one test should include both blocks represented at the transistor level. For the blocks for which models are used, the effects required to be included in the model are identified for each test. This is the beginning of the modeling plan; typically, many different models will be created for each block.

It is important to avoid writing models that are more complicated than necessary. Start with simple models and model additional effects only as needed. Also, the emphasis when writing models should be to model the behavior of the block (behavioral modeling) rather than modeling the structure. A simple equation that relates the signals on the

terminals is preferred to a more complicated model that tries to mimic the internal working of the block.

It is also unnecessary to model the behavior of a circuit block outside its normal operating range. Instead, you can add code in a model that looks for inappropriate situations and reports them. Consider a block that supports only a limited range of input biases, it is not necessary to model the behavior of the block when the input is outside the desired range if in a properly designed circuit it will never operate in that mode. It is sufficient to simply generate a warning that an undesirable situation has occurred. Following these general rules will result in faster simulations and less time spent writing models.

A formal planning process generally results in more efficient and more comprehensive verification, meaning that more flaws are caught early and there are fewer design iterations. The simulation and test plans are applied initially to the high-level description of the system, where they can be quickly debugged. Once available, they can be applied during the mixed-level simulations of the blocks, reducing the chance that errors will be found late in the design cycle.

System-level design is generally performed by system engineers. Their goal is to find an algorithm and architecture that implements the required functionality while providing adequate performance at minimum cost. They typically use system-level simulators, such as Simulink, that allows them to explore various algorithms and evaluate trade-offs early in the design process. These tools are preferred because they represent the design as a block diagram, run quickly and have large libraries of predefined blocks for common application areas.

This phase of the design provides a greater understanding of the system early in the process. It also allows a rapid optimization of the algorithm and moves trade-offs to the front of the design process where changes are inexpensive and easy to make. Unworkable approaches are discarded early. Simulation is also moved further up in the design

process, where it is much faster and can also be used to help partition the system into blocks and budget their performance requirements.

1.6 Mixed-signal hardware description languages

Both Verilog-AMS and VHDL-AMS have been defined and simulators that support these languages are emerging. These languages are expected to have a big impact on the design of mixed-signal systems because they provide a single language and a single simulator that are shared between analog and digital designers. It will be much easier to provide a single design flow that naturally supports analog, digital and mixed-signal blocks, making it simpler for these designers to work together. It also becomes substantially more straightforward to write behavioral models for mixed-signal blocks. Finally, the AMS (Analog Mixed Signal) languages bring strong event-driven capabilities to analog simulation, allowing analog event-driven models to be written that perform with the speed and capacity inherited from the digital engines**.**

It is important to recognize that the AMS languages are primarily used for verification. Unlike the digital languages, the AMS languages will not be used for synthesis in the foreseeable future because the only synthesis that is available for analog circuits is very narrowly focused [5,6,7].

Verilog-A is an analog hardware description language patterned after Verilog-HDL. Verilog-AMS combines Verilog-HDL and Verilog-A into a MS-HDL that is a super-set of both seed languages [5]. Verilog-HDL provides event-driven modeling constructs, and Verilog-A provides continuous-time modeling constructs. By combining Verilog-HDL and Verilog-A it becomes possible to easily write efficient mixed-signal behavioral models. A unique feature of Verilog-AMS is that it provides automatic interface element insertion so that analog and digital models can be directly interconnected even if their terminal/port types do not match. It also provides support for real-valued event-driven nets and back annotating interconnect parasitics.

VHDL-AMS is a superset of VHDL 1076-1993, retaining all the language principles of VHDL 1076, e.g. modularity and strong typing, while adding new powerful language elements and mechanisms to describe analog and mixed signal systems.

VHDL-AMS adds continuous time modeling constructs to the VHDL event-driven modeling language. Like Verilog-AMS, mixed-signal behavioral models can be directly written in VHDL-AMS. Unlike with Verilog, there is no analog-only sub-set. VHDL-AMS inherits support for configurations and abstract data types from VHDL, which are very useful for top-down design [6,7].

1.7 Thesis Organization

Chapter 2 gives an overview of the existing analog and digital design flows and discusses the need for a mixed signal design flow and finally a mixed signal flow is introduced in this chapter.

In chapter3, the architecture of charge pump phase locked loops and the basic mathematical modeling will be introduced.

In chapter 4, the PLL's components are modeled using VHDL-AMS and simulation results are discussed.

Chapter 5 covers the circuit design of the PLL components.

Chapter 6 is conclusion and some ideas for future work.

Chapter 2

Mixed Signal Design Flow

2.1. Introduction

During the last decade the computer and telecommunication industry has experienced a huge evolution. Previously communication networks were only available to restricted groups of users; recently it has become affordable for consumer markets (e.g., cellular phones, broad band cable applications, PCs, etc.). This on going evolution has led and is leading to the introduction of new standards and protocols to improve the channel capacity, robustness and reliability. In addition to economic decisions, a number of technical barriers had to be crossed to enable these achievements. Former electronic products were assembled from discrete components and expensive technologies. Thanks to the huge investments and persisting research, cheaper technologies have become feasible, and new circuit topologies have emerged that overcome earlier existing problems with traditional topologies. The use of CMOS technology has recently been proven feasible for fully integrated transceivers. To address this market expansion, a telecommunication company will have to increase its production volumes, reduce the overall costs, and diversify its products. What is even more important is the short time-to-market of its product to consolidate a large market share.

This has some implications on the design methodology. In a research environment, expert designers develop much of the analog front-end. This is a very knowledge intensive part of the design. However, when the product is transferred from the research to the development department, most of the technological problems have already been tackled. A broad range of new products now has to be developed that fit to the different existing standards. Time-to-market is very critical. At this stage, reusability is a key issue to shorten the design cycle. Reusability is a strict meaning implies a copy-and-paste action of an existing component into the new design. Here, however, reusability in a broader

sense is intended. Reuse starts at a much earlier stage. Starting from existing design frame works and sets of necessary equations, the decision trajectory of previous designs is followed as much as possible. This allows tuning of certain design parameters, studying quickly the influence on the system performance, and tailoring the design to different applications.

By following a systematic design methodology consistently, the reuse of a large part of the design can be drastically accelerated. Typically a top-down design and bottom-up verification method is used [8]. Therefore, different abstraction levels are identified. At each level, the behavior of the overall circuit is evaluated using models of the composing blocks. Each model calculates its output variables given the input variables. The format of these variables depends on the abstraction level. In an analog design environment, high-level variables might be a complete signal spectrum, or harmonic distortion components, whereas low-level variables are voltages, currents, widths, and lengths of transistors. Top-down implementation then implies that lower level model parameters are assigned a value such that the higher-level model meets its specifications at a minimum cost (i.e. minimum power consumption or chip area). After having given all variables a value and having designed and laid out every block, a verification of each implementation is performed to take lower level second-order effects into account in the performance at the higher level.

This design methodology is supported by a number of behavioral models at different levels.

2.2. Defining different level of abstraction

A top-down bottom-up verification methodology starts from the definition of different abstraction levels. In the digital design methodology, these levels are well defined as seen in Figure 2.1. At the behavioral level, signal transfers are usually described under the form of an algorithm. Signals are represented using reals and integers. The next level is the register-transfer level. The execution time frame of the algorithm is partitioned into

clock cycles. Moreover, all data have a binary representation. At the functional level, all operations are scheduled on a number of functional blocks (multipliers, accumulators, registers, etc.). Control signals are clearly determined. At the gate level, these functional blocks are decomposed into elementary logic gates. Finally the logic gates are replaced by their transistor equivalents at the transistor level.

In the analog domain, these different levels of abstraction are less distinguished as seen in Figure 2.1. At the functional level, the basic signal flow is described in terms of mathematical functions. No conservation laws on the interconnecting nodes have to be satisfied. One level lower, at the behavioral level, these mathematical functions are replaced by a number of high-level blocks e.g., linear transfer functions, op-amps, A/D converters, etc. The conservation laws are now enforced. Still one level lower, at the macro level, the circuit consists of elementary components, such as resistors, capacitors, controlled sources. By adding more detail to these models, second-order effects (slew rate, finite gain, etc.) can be taken into account as well. Finally at the circuit level, the circuit is decomposed into its elementary components and all the design parameters can be assigned values [8].

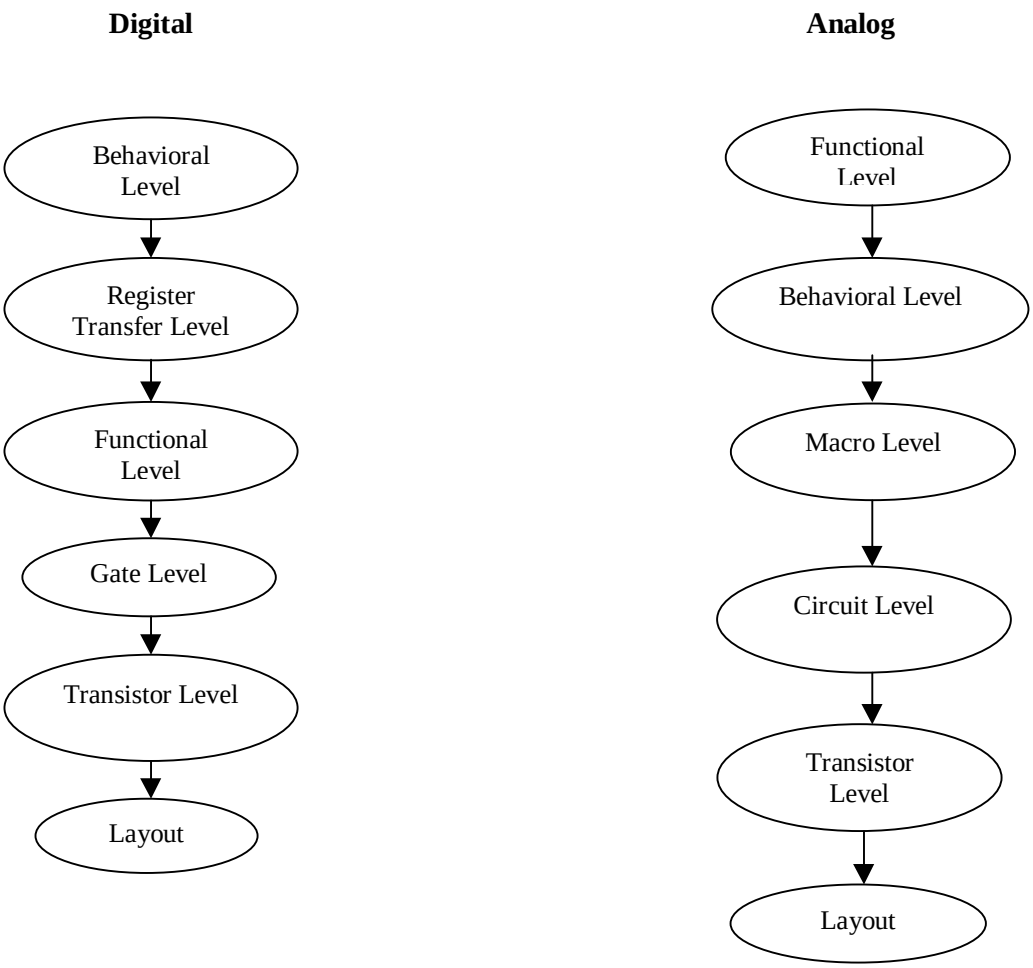

Figure 2.1 Different abstraction level

2.3. Modeling Requirements

Setting up libraries with models for the different building blocks is a necessary effort. Fortunately, the availability of an appropriate model library has a number of advantages to justify all this work. The introduction of hierarchy and abstraction in the digital domain allows the designer to handle larger degrees of complexity. The complexity of the analog part also tends to increase employing different types of models at different levels can then elevate the analog design abstraction.

Once analog building-block models are developed, they can be instantiated to perform a system design. When implementing a lower level of the design, the higher-level description is used as a specification and reference.

When developing models, some requirements have to be fulfilled. The functionality should be modeled in a generic, parameterized way. The model can then cover a wide range of design spaces of underlying lower level implementations.

When one wants to evaluate a certain specification, all circuit aspects that influence this specification should be included in the model. In general, a tradeoff can be made between the accuracy of the model and the necessary evaluation time. In the first stages of a top-down implementation, less accurate models can be allowed to get a rough estimate of the design space.

Finally, all these models should be implemented in a standardized manner. This will ease the exchange and reuse of models. In digital designs, the use of VHDL or Verilog is widespread, both for simulation and synthesis. In the analog domain, however, the VHDL-AMS and Verilog-AMS standards are about to be finalized and commercial simulators will soon be released.

2.4. Existing Analog and Digital Design Flow

2.4.1 Digital Design Flow

There are various versions of a digital design flow in industry. Each company or research lab, bases theirs on available resources, tools and type of design. As such, they come up with different versions of a design flow. Digital design flows are well established in the industry and the reason for this is the many CAD tools in the market that can help designers to complete the design (http://www.cmc.ca/).

In every digital design, the following shows major steps in a digital design :

- High-level System Design
- Architectural Exploration
- RTL Simulation
- Design For Test
- Timing-Driven Logic Synthesis with Scan Insertion
- Gate-Level Simulation
- Floorplanning and Timing-Driven Placement
- Extraction and Delay Calculation
- Pre-Clock Tree Synthesis Timing Check
- Clock Tree Generation
- Pre-Route Golden Verilog Netlist Verification
- Routing
- Post-Layout Static Timing Analysis
- Physical Verification (DRC & LVS)
- Manufacturability
- Tape-Out

2.4.2 Analog Design Flow

As already mentioned in the previous section, design flows are dependent on the design environment's CAD tools. CMC's analog design flow was selected for the analog part of the proposed mixed signal design flow (http://www.cmc.ca/). The steps will be further described in the next section. The primary objective of the analog design flow is to produce working parts and the secondary objective is to make the parts re-usable.

The following shows the major steps in a typical analog design flow:

System-Level Design

- *Set Design Goals and Priorities*
- ***Preview Spec Gate***
- *Functional Capture &Architectural Exploration*

- *Packaging Selection*
- *Verification: Simulation and Test Plan*
- *Partitioning & Block Behavioral Modeling*
- *Top-Level & Block Specification Documentation*
- ***First-Pass Gate***

Block-Level Design

- *Topology Selection & Block Schematic Capture*
- *Test Plan Update*
- *Pre-layout Simulation with Estimated Parasitics*
- *Optmization*
- *Layout*
- *DRC/LVS*
- *Post-layout Simulation*
- ***Block-Level Gate & Risk Assessment***
- *Block Specification Update*
- ***Pre-top Level Gate***
- *Top-level Layout Design*
- ***Post-layout Gate***
- *Complete Documentation & Test Plan*
- *Create ROL (read only library) & Archive*
- *Tape Out*

2.5. Mixed Signal Design Flow

There are two different approaches for designing mixed signal systems. In the first approach, the mixed signal system is decomposed into pure digital and pure analog sub-systems and each of these sub-systems is designed individually using the analog and digital design methodologies. Because of the interaction between the analog and digital sub-systems the designer should find an equivalent model of the analog circuit at the connection port of analog and digital systems. Finding the circuit equivalent of the

analog sub-system could be very difficult and in most cases, it is not accuracy enough to model the actual behavior of the analog circuit. On the other hand, generating digital signals in an analog simulation environment is difficult and in most of the cases the designer assumes simplified equivalent control signals. In this approach, the simulation of the overall system is only possible at the final stage of design and after completing the layout. Any changes at this stage are difficult and time consuming. Before the post-layout simulations, designers can only simulate the effect of digital and analog interactions at the system level.

By comparing different abstraction levels in the digital and analog design flows, we can see some similarities between the two flows. The second approach for designing the mixed signal system is based on analog-digital co-design. In this approach, after the system level design, the behavioral / RTL model of the overall system will be developed and verified. VHDL-AMS can be used to have a mixed signal model of the chip. The digital part would be described using a RTL synthesizable subset of the language, while the analog part would be partitioned into functional blocks at the functional or behavioral level, e.g. Filters, VCOs, opamps, etc. The whole model can be simulated using test benches written in VHDL-AMS. The next step is the block design which has different steps for digital and analog blocks. The digital part of the chip can be synthesized using a logic synthesizer to produce a gate level netlist. Analog blocks are individually designed at the transistor level. After completing each block, it is possible to test the block in the interaction with other blocks using the test benches developed earlier.

Standard cells place and route tools can produce the layout of the digital part from the gate-level netlist. The layout of the analog block is usually created manually or through dedicated module generators. From the layout the parasitic elements are extracted. Those elements related to the digital part are used to compute delays that are stored in SDF (Standard Delay Format) files. The final simulation can be done using the extracted layout view of the overall chip. Figure 2.2 shows this approach.

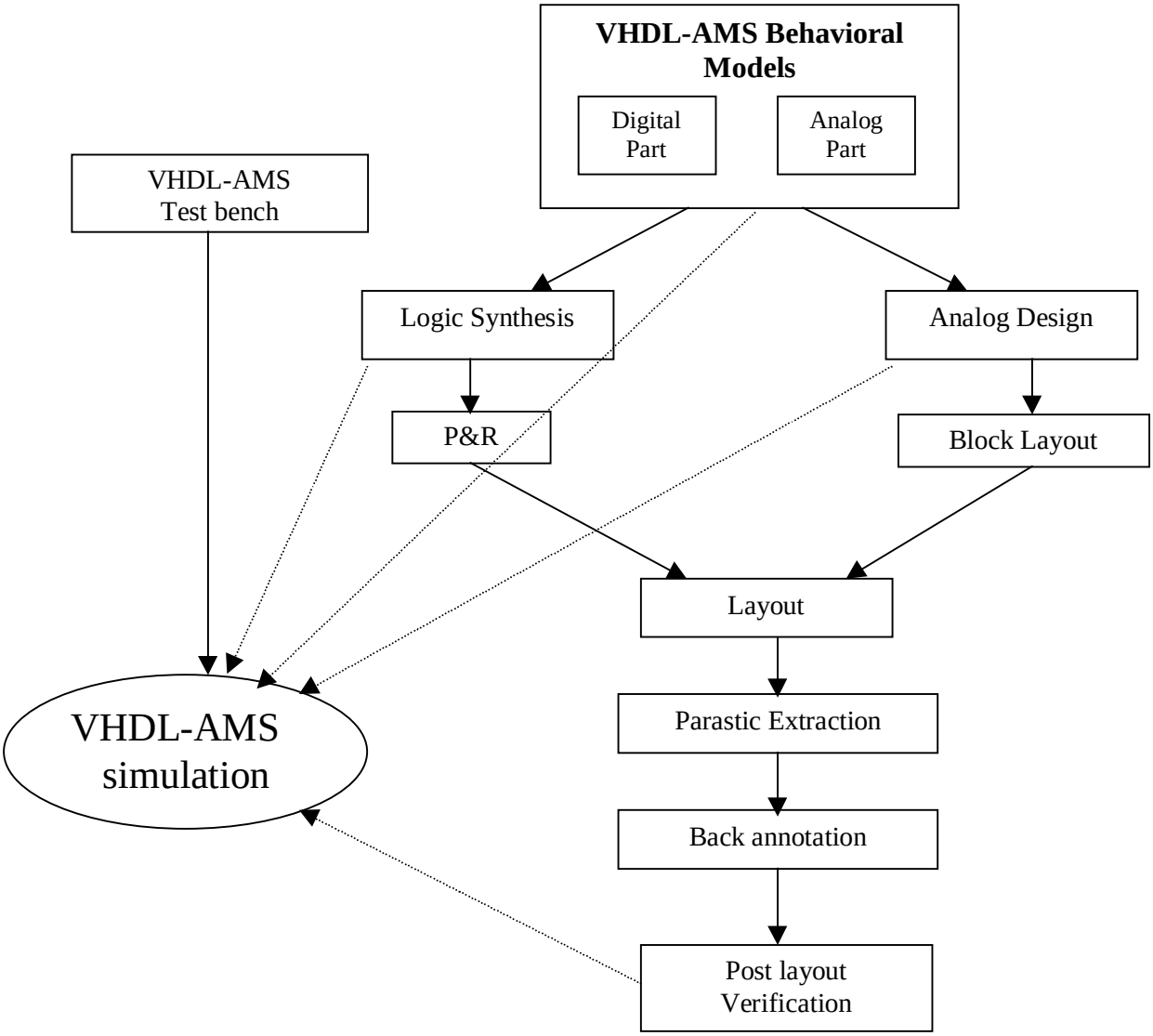

Figure 2.2 Analog / Digital co-design

2.5.1. Proposed mixed signal design flow

The proposed mixed signal design flow is based on analog-digital co-design methodology and has two distinguished levels of abstractions.

- System Level Design
- Block Level Design

System level Design

Figure 2.3 shows this part of design flow. The following major steps shall be followed at this level:

Set Design Goals

A design process starts with a clear statement of the problem, a search of existing state of the art solutions, clear objectives of the current design, identification of a possible solution for achieving the objectives and selection of implementation for the solution. A marketing or preview spec can be generated at this point on paper for peer review. The specs should include descriptions of functions, estimated performance metrices (speed, power, noise, etc.) and projected operating constraints (bias, thermal, I/O impedance, proximity, etc.).

Preview Spec Gate

With goals and priorities set and resources planned, a peer review should be done to ensure that the overall scope of the design project is acceptable and the right decisions have been made before proceeding further.

System Level modeling

The preview system specs are captured with Matlab/Simulink or VHDL-AMS for a more precise definition and verification of the specs and to allow the exploration of appropriate architectures by using available modules from existing libraries and/or mathematical representations created by the designer.

At this stage the use of re-usable (IP) blocks should be considered as their availability can have a strong impact on the selection of architecture and development time.

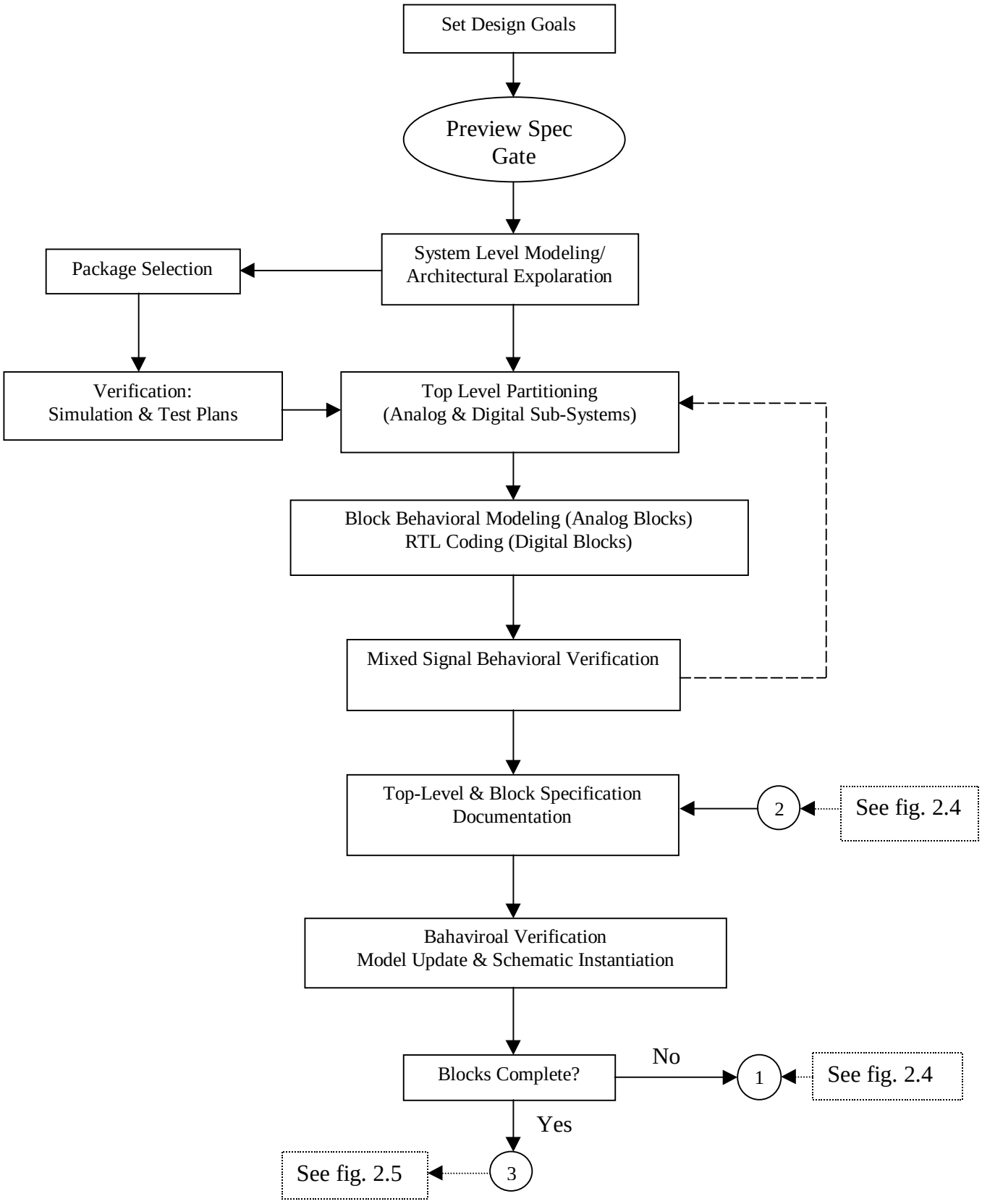

Figure 2.3 System Level

Packaging Selection

The designer should select the packaging solution early because the design or the final product will eventually need to interact with the outside world through its packaging and the packaging chosen may significantly affect the design's behavior. Considering packaging effects in the early design stage is crucial. The decision on the packaging can have a strong impact on design. It can constrain partitioning, improve the accuracy of behavioral modeling if a packaging model is available and enhance the validity of simulation/test plans.

Verification: Simulation and Test Plan

In parallel to the architecture exploration, the verification strategy including simulation plan and test plan should be considered. The issues to be considered include how top level netlist and each block or sub-block will be modeled and simulated, what types of simulations (transient, ac, noise, frequency domain) of simulations at each level will be used, how the stimuli will be created, how the interference between the blocks will be modeled, what tests to be performed, what test equipment to be used, how to bias the chip, what supply decoupling is satisfactory, what DFT techniques should be used to facilitate testing and diagnosis. Some of verification plan details can also be derived after design partitioning and behavioral modeling. It is an iterative process between verification planning and partitioning and behavioral modeling.

Top Level Partitioning

The architecture is partitioned into digital and analog blocks and each of the digital and analog blocks are further partitioned into basic sub blocks. The architecture must be partitioned in a way that maintains as much hierarchy as possible, makes use of common implementable functional blocks, minimize critical connections between blocks and must be consistent with the chosen packaging technology in terms of electrical, mechanical and thermal characteristics.

Block Behavioral and RTL coding

Behavioral modeling can be done for both analog and digital blocks using VHDL-AMS .

The overall behavioral model of the system can be simulated and verified. There could be different levels of abstraction for each behavioral model starting with simple models and then designing more complex models. Digital blocks can be modeled both at the behavioral and RTL levels. Power domain and clock strategy must be considered to obtain the optimal power distribution and consumption, to enhance routability, reduce interference among blocks, facilitate clock signal generation and distribution.

Mixed Signal Behavioral Verification

Top level behavioral simulations must be performed to achieve satisfactory functional and performance results against the preview specs before proceeding further down the flow. Otherwise the designer needs to go back to structural mapping, partitioning and behavioral modeling until the targets are met.

Top-Level and Block Specification Documentation

When the results are satisfactory, the designer needs to document the functionality, performance, interface conditions, physical size and power consumption for the top level design as well as each autonomous and re-usable block. The documentation is crucial in tracking the design optimization process, helpful in guiding design convergence and essential for passing the gating process. It is also required for revision tracking.

Block Level Design

Block level design for digital and analog blocks is different. Design of digital blocks is based on digital synthesis and standard cell libraries. It is a process that can be done using available synthesis tools. Designing analog blocks is done by the designer and his knowledge of analog circuit design. Figure 2.4 shows this part of the design flow. The steps for each of these design processes are explained briefly.

Analog Block Design

Topology Selection and Block Schematic Capture

A schematic corresponding to the block behavioral description will be created (in

Cadence Composer) and it must be properly linked to the behavioral model for later instantiation. Each schematic must have the proper pins so a symbol can be created from the schematic and be used to construct a test bench for circuit simulations.

In terms of selecting a circuit topology one needs to decide on which active and passive devices to use and to take in account several factors including gain, frequency range, power handling capability, availability of models, etc.

Test Plan Update

This is a good time to revise the test plan since more design details are available at this point and to take into account all the necessities to test the final circuit.

Pre-layout Simulation with Estimated Parasitics

When constructing the block schematic, the designer needs to consider ways of incorporating parasitics into the simulations. Therefore, a rough layout for extracting or estimating critical parasitics prior to schematic simulation is desirable since it would help identifying problems early and facilitate design convergence. The appropriate circuit topology can be selected with confidence and necessary circuit adjustment can be made before spending a significant amount of effort on detailed layout. A rough layout also helps estimate the block size that can be important in the overall design.

Hspice or Spectre can be used to perform several types of simulations including: DC, transient, AC (noise), and nonlinear frequency domain analysis. A DC analysis is used to establish proper biasing.

Optimization

At this point, performance and yield optimization is performed, if necessary, on the analog design of each block. Performance optimization requires fine tuning of circuit components and biasing or may be the addition of circuit components. The circuit is resimulated and adjustments are made to meet the predetermined performance criteria.

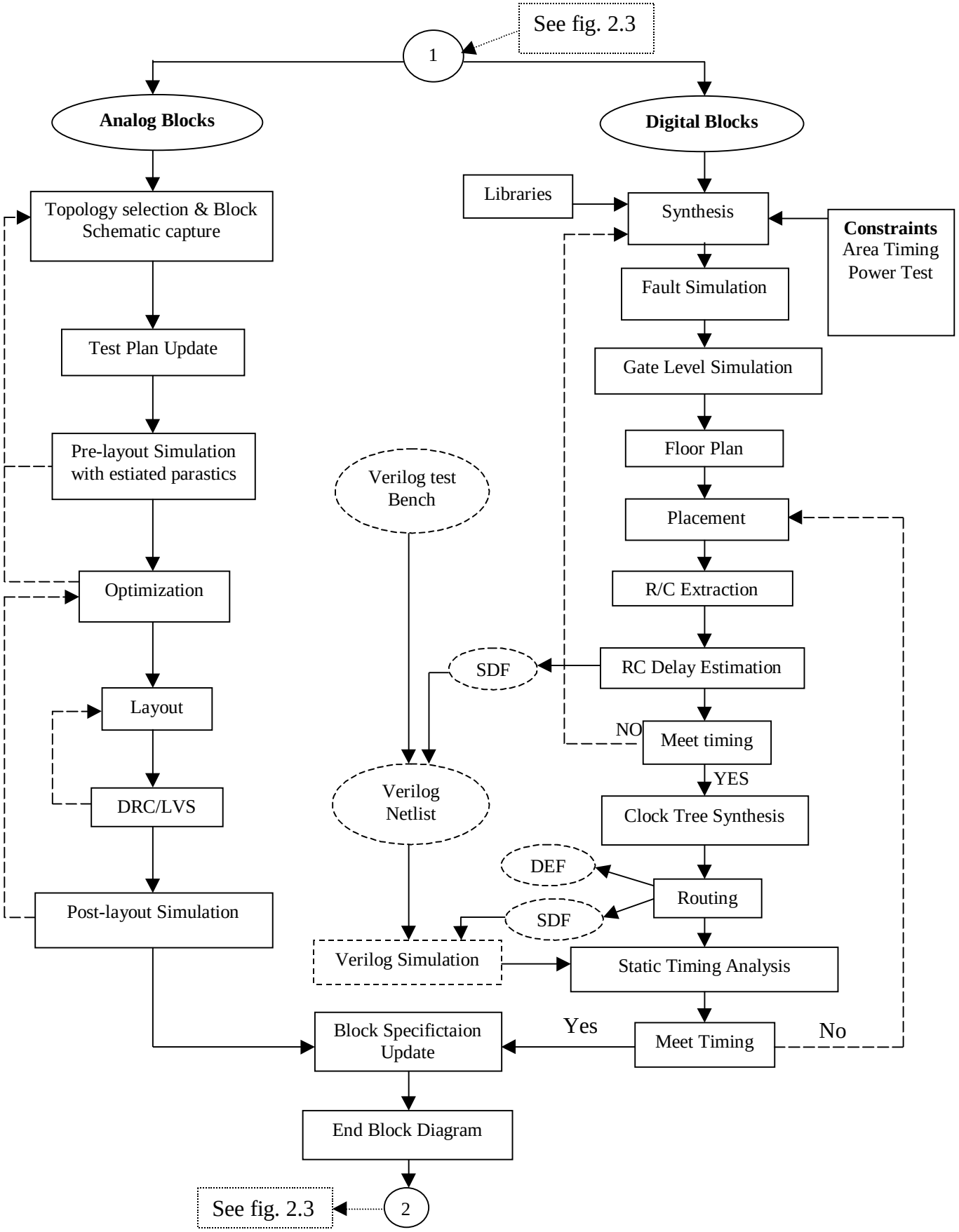

Figure 2.4 Block Design

Performance optimization tends to maximize the performance margin without considering manufacturing yield. Even though a nominal design can meet specifications, a significant number of chips may fail when component values are allowed to vary within their tolerance. A process called Yield Optimization is used to overcome the weakness of performance optimization which takes the tolerance of devices into consideration.

Layout

After the schematic has been optimized to meet specifications, Cadence Virtuoso-XL can be used to perform schematic driven placement for the cells or devices and routing the layout.

Block layouts are likely to be used as macros for place and route at a higher level. Therefore, some consistencies regarding what type of metal layers to use and where power/ground pins are located is desirable. Hierarchial layout style is recommended for handling the complexity of a large block design.

DRC/LVS

Frequent design rule checks (DRC) on layout, particularly with the ones constructed manually, picks up errors early and makes debugging simpler. When running DRC be aware of the various switches that control the turning on or off of particular rule sets in case designers want to deal with selected DRC problems one at a time. Eventually, the design must be clean of violations against all rules. For deep sub-micron technologies, space fill rules and antenna rules, are as critical as other rules and must be observed.

After layout and DRC, Layout Versus Schematic (LVS) verification is performed to ensure that the netlist created by the schematic and that of the extracted layout match. If they do not, the errors should be corrected.

Post-Layout Simulation

An extraction of the layout followed by post layout simulation ensures that most of the parasitics are as expected and any unaccounted for parasitics have not significantly affected the design's performance. The post layout simulation results may indicate that some adjustments or optimization is required, perhaps even going back to block schematic capture.

Digital Block Design

The digital design blocks are mostly the same design flow as in the CMC digital design flow for synthesis.

Logic Synthesis

This step includes the creation of timing budget for digital blocks; scan insertion, technology dependent mapping and optimization. If the design's HDL code is not synthesizable, the RTL code should be modified. The designer should define the design environment, constraints, design rules, technology libraries and compile strategy. In this thesis, the Design Compiler was used to synthesize HDL description into technology specific gate level implementation. After synthesizing the design into a gate level netlist, timing analysis was done. This process is interactive and might require modifying the original HDL code or the synthesis constraints.

A scan chain can be inserted. This process will replace all the flip-flops with their scannable equivalent. The multiplexed flip-flop scan style is the most commonly used DFT technique.

Gate Level Simulation

The gate level simulation enables the designer to check the functionality of the structural netlist against the RTL simulation. The testbench used previously for the RTL simulation is used here. Using VHDL-AMS, the designer can verify the functionality of the synthesized block in interaction with analog blocks.

Floor planning

This steps involves the creation of rows around the perimeter of the design area for placing the I/O pad cells, core area with spacing the I/O pads, rows or columns or both in the core area. The designer may also create a power grid prior to placement. This step may also include placement of cell groups or macro blocks to optimize the connectivity between groups and blocks. The automatic placement tests potential placements for the design and tries to optimize the placement for overlap removal, routing congestion balancing, power balancing, wire length and timing assurance.

Extraction and Delay Calculation

This step is needed to extract parasitic capacitance and resistance from the layout to calculate and apply delays in static timing analysis and/or full timing simulation using Verilog-XL. The parasitic information is extracted from the layout, and interconnects delays included in the SPF (Standard Parasitic Format) file.

Pre-Clock Tree Synthesis Timing Check

The static timing analysis should be performed using projected parasitics to verify that all timing goals/constraints set after synthesis are still met.

Clock Tree Generation

The designer has to build a clock tree when a large number of cells are clocked by a single driver cell. In this case we are trying to control the signal skew at the clocked cell's input. It is assumed that the physical library includes timing data in a Timing Library Format (TLF). All modifications to the netlist are saved in a DEF (Design Exchange Format) file for back annotation to the original netlist.

Routing

This step includes global and final routing. Global routing usually consists of a coarse regular wiring layout based on obstructions resulting from special wiring, clock wiring and placement. Analyzing the routing congestion map before attempting the final routing

is recommended. Final routing creates the detailed regular wiring layout. Post layout timing analysis may be done after routing. It can be done by back annotation of SDF file.

Post Layout Static Timing Analysis

Using SDF, CAP and RES files with accurate timing information post layout simulation and timing verification can be performed by back annotating the SDF file.

DRC & LVS verification

It is very important to run DRC and LVS on the layout to be sure that the connectivity, the geometry and the spacing are correct and the layout matches the schematic. This step includes a flat extraction of the layout.

Block Specification Update

After each block is done, it is possible that an update on the block specs is required and therefore the respective block documentation will have to be modified.

To verify the updated behavioral model and physical layout of each block, the designer needs to perform two simulations from the top level, one with and one without circuit instantiation of the target block. Other blocks should remain at the behavioral or RTL level for these simulations. The same test bench created at the partitioning and behavioral modeling stage should be used for this regression simulation.

Top Level Layout Design

At this stage of the flow, the layout of all analog and digital blocks are ready and we have to integrate them. Cadence Virtuso-XL can be used to perform schematic driven placement for the blocks at the top level based on the top level schematic created earlier at the partitioning stage. DRC and LVS are performed to ensure correctness of the layout. Figure 2.5 shows this part of the design flow. Post layout extraction and simulation are done to verify the top level parasitic modeling. Any errors revealed by DRC/LVS or any undesirable parasitics revealed by post layout simulation need to be corrected by going back to top level layout or block layout.

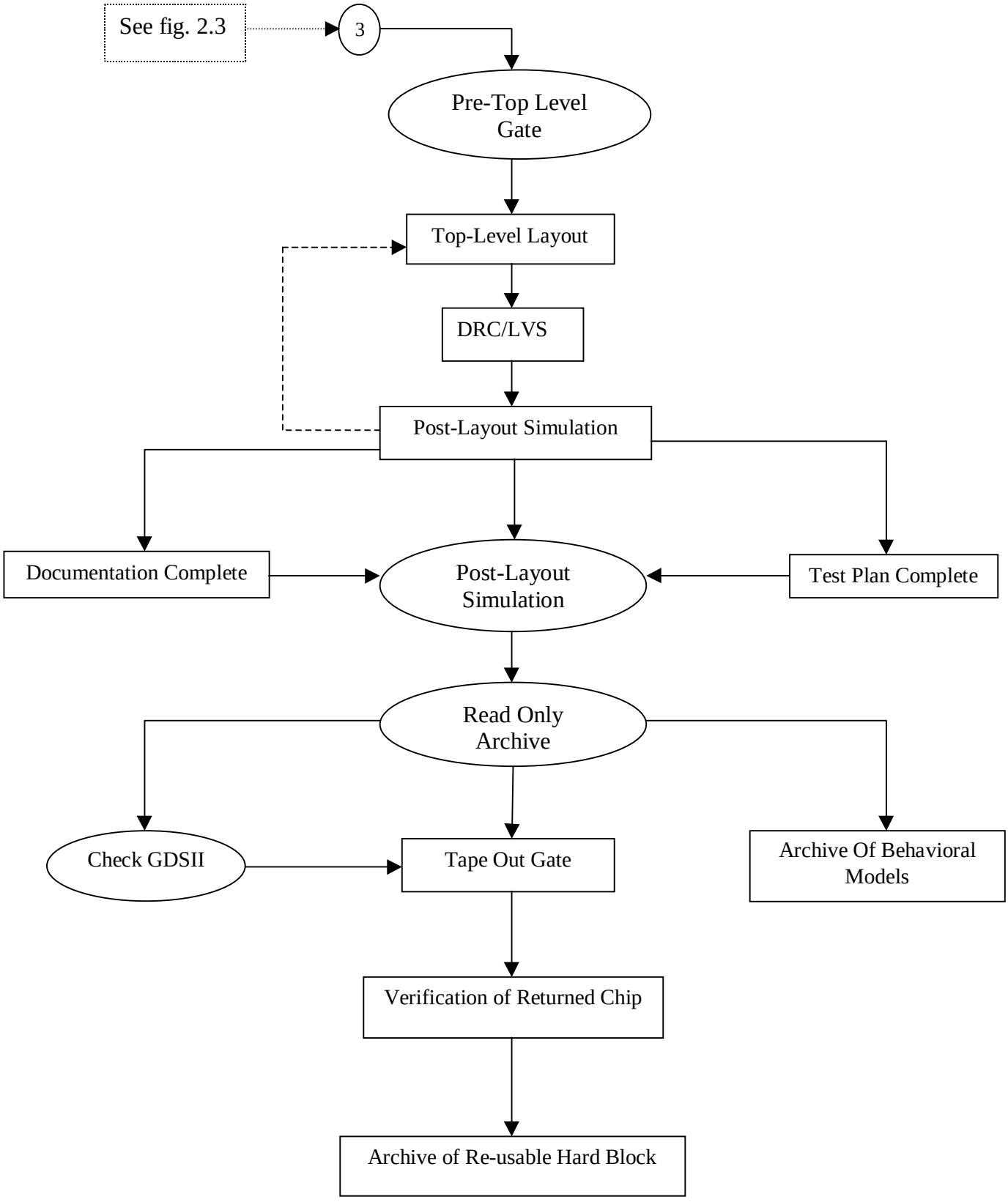

Figure 2.5 Chip Integration

Post Layout Gate

When the top level design passes post layout simulation, a post layout gating should be done. Gating is a peer review of the process to ensure that a post layout simulation has been performed properly and to check on manufacturing issues such as power/thermal considerations, metal migration issues, power IO to signal IO ratio, ground bounce problem, proper design ID.

Complete Documentation & Test Plan

In parallel, top level design documentation needs to be completed or updated after successful post layout simulation and so does the test plan. At this point, the exact test setup or procedures down to what pin is connected to what instrumentation through what fixturing can be described. All that will converge back to post layout gating. Complete design documentation and test plan are essential components of passing the post layout gate.

Read Only Archive

After passing the post layout gate, a read only library (ROL) should be created to archive the design. Preferably the design data is archived in a standard data formats such as GDSII/DEF/LEF. The design must be frozen at this point so the right version of the design can be used for debugging later on. Archiving designs using consistent format, style, directory structures makes re-use easier. For re-use purposes, it is even more important to archive the technology independent behavioral models than the physical data files. The behavioral models must be properly documented and stored. Before sending out the GDSII file for fabrication, it is desirable to read the file back into Cadence to perform an LVS against the original layout to ensure there are no translation problems occured.

Tape Out

The GDSII file can then be sent out for fabrication.

Chapter 3
Charge Pump PLL

3.1. Introduction

Phase locked loops (PLLs) are used in many applications, such as frequency synthesizers, clock recovery circuits, receiver demodulators and modulators. In particular, as a result of the boom in wireless communications, the design of the frequency synthesizers has recently received a lot of attention in both industry and universities. The IC market is pushing toward a higher level of integration and lower power consumption. The inherent lower power cost and higher density of CMOS technology make it attractive for mixed signal devices.

PLLs are for external phase synchronization and clock multiplication, and are widely used for high speed microprocessor applications. A PLL allows the clock signal to be generated on-chip. This important feature of the clock generator avoids problems relating to signal integrity. Another important feature is that the PLL aligns the core clock with the reference clock frequency by including the clock tree buffers in the feedback loop of the PLL. This accurate alignment allows synchronous fast and efficient data transfer between the core and the external world.

However, power-supply noise generated by large switching digital circuits like microprocessors perturbs the analog circuits used in the PLL. The output clock period may change with the power-supply noise and with other sources of noise (for example thermal noise in MOS devices). It is common to refer to this change as jitter, which is a

variation of the clock period from one cycle to another cycle compared to the average clock period. The clock jitter directly affects the maximum running frequency of the processor because it reduces the usable cycle time. When the clock period is small, the digital circuits in the critical path may not have enough usable time to process the data in one period, resulting in the failure of the processor (critical path failure).

The supply noise generated by the switching activity of the processor appears to induce an important part of the jitter and phase misalignment. This supply noise problem is even more significant when the microprocessor shares the same substrate as the analog circuits and when the power consumption of the microprocessor is large.

The power supply noise perturbs the analog circuit in many ways. The most common is the ripple induced on the supply voltage, which perturbs analog circuits having a high sensitivity to power supply. Another source of perturbation is the current flowing in the substrate that creates voltage gradients in the substrate, which modulates the threshold voltage of MOS devices. As the maximum frequency of the core clock is of primary concern in this design, the performance of the clock generator in terms of jitter and phase alignment are crucial [9].

3.2. Charge pump PLL Architecture

A PLL circuit is a negative feedback control system as shown in Figure 3.1. As can be seen, a charge pump phase locked loop consists of a phase detector, a charge pump, a loop filter, a voltage-controlled oscillator and a divider. The loop filter can be either passive or active. The negative feedback adjusts the phase of the of the divided VCO output to match the input phase and forces the frequency of the divider output to be equal to the input reference frequency.

The phase detector compares the phase of the reference input to the phase of the feedback signal, i.e, the divided output of the VCO. Triggered by the rising or falling edge of the inputs, the phase detector creates a pulse in the two outputs, UP and DOWN,

respectively. The difference between the duration of the two pulses is proportional to the phase difference.

Phase detector output signals, UP and DOWN switches the current flow into or out of the loop filter. Thus the charge pump with the loop filter performs a conversion from the digital outputs of the phase detector to an analog voltage to control the VCO frequency.

The phase detector output controls the amount of charge pumped into or out of the loop filter depending on the relative phase of the inputs. The loop filter converts the charge into a control voltage that alters the VCO frequency. For example, if the input reference has a higher frequency, the UP current is turned on to increase the control voltage, which in turn speeds up the VCO until the VCO frequency becomes N times the input reference

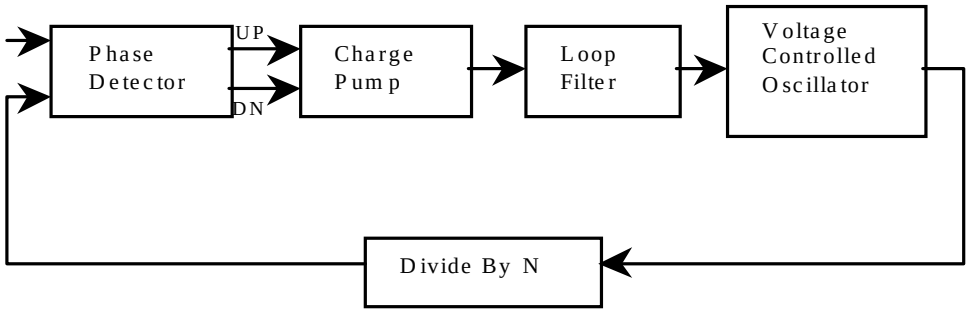

Figure 3.1 Block diagram of a charge pump phase locked loop

frequency. Thus when the loop locks, there is no net charge going into or out of the loop filter in each comparison cycle. A dc control voltage is maintained to run the VCO at the lock frequency [10].

3.3. PLL Modeling

Modern high speed microprocessors employ deep-submicron CMOS devices to achieve gigahertz operating frequencies. As the gate level is shrunk to achieve faster operation,

the power supply voltage is reduced to avoid break down and reliability failures.

Analog circuit design becomes more difficult for 0.18 μm CMOS process that requires a 1.9 V power supply voltage. Making matters worse, the allowable clock jitter decreases at higher clock frequencies for a given clock skew tolerance; e.g., a jitter less than 4% of the clock cycle time is typically required to avoid functional failures in a microprocessor. Moreover, the high degree of integration leads to the generation of substantial digital switching noise that is coupled through the power-supply network and the substrate into noise-sensitive analog circuits [11].

A charge-pump phase-locked loop (PLL) often employs a series RC loop filter where R is added to form a left-half-plane (LHP) zero that stabilizes the loop. Figure 3.3 shows a second-order RC loop filter connected to the charge pump circuit. This approach is limited by process, voltage, and temperature variations of the resistance. Process variations alone are typically 30% for an ion-implemented resistor in a digital CMOS process. Since the damping factor is proportional to R [11,13], loop stability changes dramatically with process, voltage and temperature variations.

The proposed design, as described in the next chapter, is a resistorless architecture. To stabilize the loop, feed forward current injection was implemented using an auxiliary charge pump.

There are two region of operation in a PLL. In the unlocked condition when no input is present, the VCO runs at the free frequency, ω_0 , which corresponds to the VCO frequency with zero applied control voltage. Once an input is applied, the loop operates in a non-linear fashion to acquire frequency locking by varying the VCO frequency. When the loop reaches the locked condition, the loop can be modeled as a linear system with constant gain assigned to each building blocks.

K_{PD} is the phase detector gain in amperes per radian. Z_f (s) is the loop filter transfer function. K_0 is the VCO gain in radians/second per volt. N is the divider ratio. Modeling

the loop as a feedback system as in Figure 3.2, one can derive the loop transfer function. Breaking the loop at the VCO output, the open loop gain would be

$$A(s) = \frac{K_{PD}.Z_f(s).K_0}{s} \qquad (3.1)$$

The pole at the origin comes from the integration of frequency in the VCO to obtain the output phase. The order of a PLL is defined by the number of poles in loop transfer function. Thus, if the passive filter is a second-order filter, the loop associated with it is a third-order PLL. Using feedback analysis, since the feedback factor, f, is 1/N, the loop transfer function becomes

$$H(s) = \frac{\Theta_0(s)}{\Theta_i(s)} = \frac{N.G(s)}{1+G(s)} \qquad (3.2)$$

where G(s) is the loop gain.

$$G(s) = A(s).f = \frac{K_{PD}.Z_f(s).K_0}{N.s} \qquad (3.3)$$

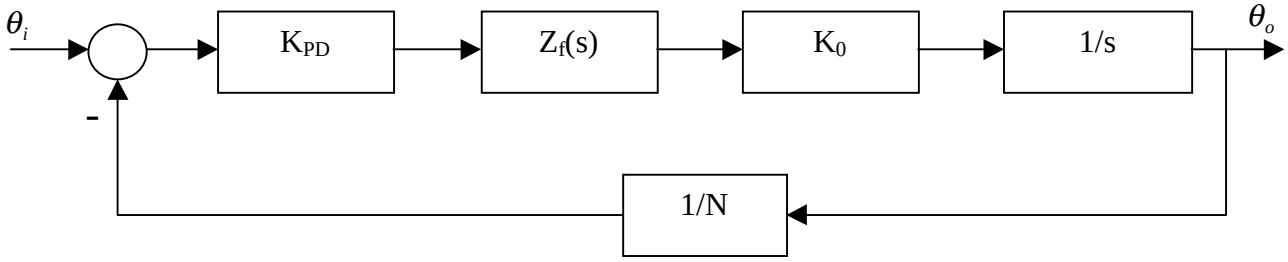

Figure 3.2 PLL Loop

Using feedback analysis, the relationship between the PLL loop bandwidth and the design parameters can be shown. From equation 3.2, it can be observed that the PLL loop bandwidth is the unity gain bandwidth of G(s), the loop gain. $Z_f(s)$ determines the poles and zeros of G(s) as they are related by equation 3.3. Figure 3.3 shows the charge pump circuit with a second-order loop filter. Assuming C_1 is much larger than C_2 in Figure 3.3, the location of poles and zeros are as follows.

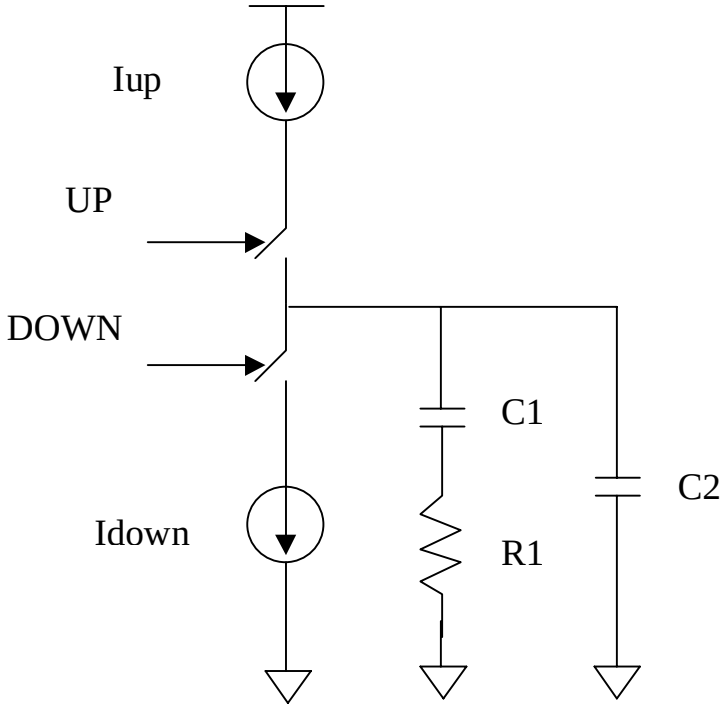

Figure 3.3 Charge pump with second-order loop filter

$$Z_f(s) = \frac{K_1.(1+\frac{s}{z_1})}{s.(1+\frac{s}{p_2})} \quad (3.4)$$

$$z_1 = \frac{1}{R_1.C_1}$$

$$p_2 = \frac{1}{R_1.C_2}$$

Usually, C_2 is small compared to C_1, so that the low-frequency response of the loop filter is essentially the same as a second-order loop without C_2. G(s) contains the poles and zeros of $Z_f(s)$ and an additional pole close to dc from the VCO.

As equation (3.2), shows, the PLL loop bandwidth, ω_{PLL} is where the magnitude of loop gain, G(s), reaches 0 dB. Assuming that C_1 is much larger that C_2, $Z_f(s)$ at the frequencies around the loop bandwidth is approximately equal to R_1. Substituting R_1 into $Z_f(s)$ and $j\omega_{PLL}$ into s in equation (3.3), the PLL loop bandwidth is

$$\omega_{PLL} = \frac{K_{PD}.R.K_0}{N} \quad (3.5)$$

Phase angle, $\Phi(\omega)$ and phase margin, $\Phi_m(\omega)$, depend on the location of the zeros and pole.

$$\Phi(\omega) = \tan^{-1}\left(\frac{\omega_{PLL}}{\omega_{z_1}}\right) - \tan^{-1}\left(\frac{\omega_{PLL}}{\omega_{p_2}}\right) - 180° \quad (3.6)$$

$$\Phi_m(\omega) = \tan^{-1}\left(\frac{\omega_{PLL}}{\omega_{z_1}}\right) - \tan^{-1}\left(\frac{\omega_{PLL}}{\omega_{p_2}}\right) \quad (3.7)$$

Taking the derivative of the phase angle with respect to frequency and setting the derivative to zero determines the location of the maximum in the phase response. In other words,

$$\frac{\partial\Phi(\omega)}{\partial\omega} = 0 \quad (3.8)$$

The result can easily be shown that the location of the maxima is the geometric mean of the pole and zero.

$$\omega_m = \sqrt{\omega_{z_1}.\omega_{p_2}} \quad (3.9)$$

To maximize the phase margin for stability, it is desirable that the maxima of the phase response occurs at the PLL loop bandwidth. Then the phase margin becomes

$$\Phi_m = \tan^{-1}\sqrt{\frac{\omega_{p_2}}{\omega_{z_1}}} - \tan^{-1}\sqrt{\frac{\omega_{z_1}}{\omega_{p_2}}} \quad (3.10)$$

Phase margin increases as the zero and pole are placed further apart [12].

3.4 Resistorless Charge pump PLL

Charge pump PLLs usually have a series RC loop filter where R is added to form a left hand plane zero that stabilizes the loop. This approach has limitation because of process, voltage and temperature variations that cause a change of loop stability. The approach in the proposed design is to stabilize the loop using feed forward current from an auxiliary charge pump circuit. A block diagram of this PLL is shown in Figure 3.4. It uses a three-state phase/frequency detector (PFD), and its loop filter capacitor, Cp, is referenced to the separate analog supply. A PLL that uses a single capacitor loop filter without a resistor is marginally stable [13].

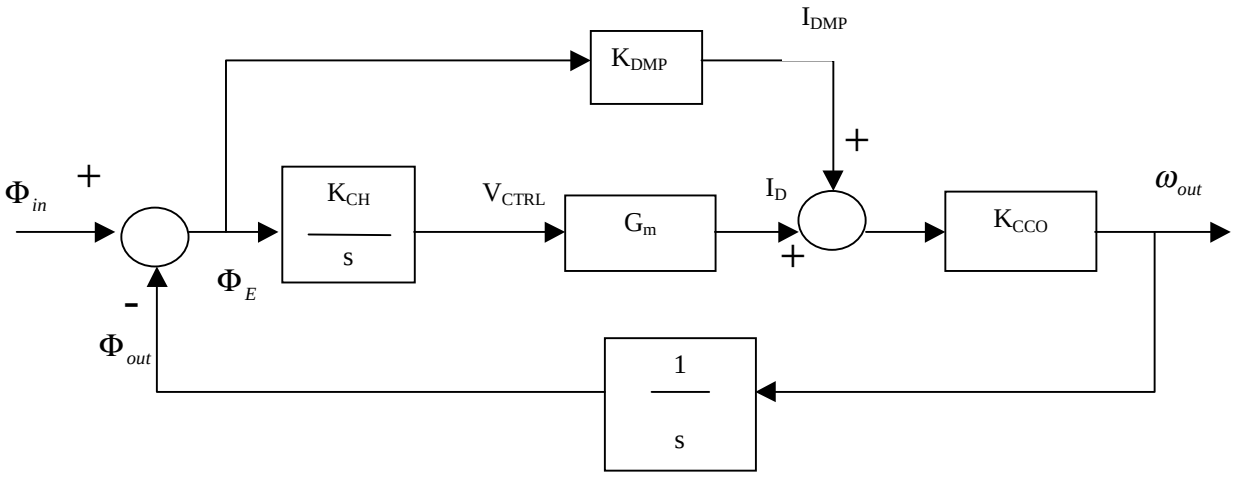

Figure 3.4 Continuous time PLL model

The open loop transfer function of this system is:

$$A(s) = \left(\frac{K_{CH} \cdot G_m}{s} + K_{DMP} \right) \cdot \frac{K_{CCO}}{s} = \frac{K_{CH} \cdot G_m \cdot K_{CCO}}{s} \cdot \frac{1 + \frac{s}{\omega_z}}{s} \quad (3.11)$$

$$\omega_z = \frac{K_{CH} \cdot G_m}{K_{DMP}} \quad (3.12)$$

From the above equations, note that the PLL with feed forward error correction is equivalent to a PLL with a loop filter that has a zero at ω_z and a pole at the origin. K_{DMP} is an important parameter that can be varied independently of the natural frequency and open loop gain. The natural frequency and damping factor of the closed loop system are:

$$\omega_n = \sqrt{K_{CH} \cdot G_m \cdot K_{CCO}} \quad (3.13)$$

$$\xi = \frac{K_{DMP}}{2} \cdot \sqrt{\frac{K_{CCO}}{K_{CH} \cdot G_m}} \quad (3.14)$$

As K_{DMP} increases, the zero frequency decreases and the damping factor increases. If the unity gain frequency is high, the phase margin of the loop increases. The stabilizing loop zero is created by a feed-forward path from the PFD output to the voltage-to-current (V-I) converter input of the voltage controlled oscillator (VCO).

3.5. PLL Performance Measures

There are several figures of merit which determine the PLL's performance. The three most important figures of merits which were considered are: lock time, jitter and power consumption.

Lock time is the time it takes for the PLL to re-enter the locked condition when switching from one frequency to another. One of the most successful ways to reduce power in digital signal processors is to turn off the PLL when it is not in use. Since the time it takes for the PLL to turn on again could be as much as a few milliseconds, it is important for a PLL to have a small lock time.

Another figure of merit that is important in PLL design is jitter. Jitter refers to the random variation in the generated clock period due to various noise sources in the PLL.

The three fundamental sources of noise at each block are:

- Thermal noise
- Coupling of noise from the power supply
- Coupling noise from the substrate

3.6 Design specification

Input:

Amplitude of input signal (V_{ref}) = 1.8 V (± 10 %)

Impedance of signal source (R_s) = 50 Ω

Input frequency (ω_{ref}) = 150 ~ 200 MHz

Power supply : 1.8 V (± 10 %)

Minimum PLL supply voltage (800 MHz): 1.4 v

Operating frequency range (V_{DD} = 1.8 V):

CCO 200 MHz ~ 1 GHz

PLL output 200 MHz ~ 800 MHz

Peak-to-Peak period Jitter (600 MHz): ≈ 60 ps

Loop bandwidth: ~ 2.5 MHz

Power dissipation: ~ 25 Mw

Locking time: < 100 nsec

Thermal environment: 10 °C ~ 80 °C

Process technology:	0.18 μ m CMOS
Die size:	<2.5 mm x 1.5 mm

3.7 Package Selection

Since there are high frequency nodes connected to the outside world, small packaging would decrease the effect of parasitics. Package models are available for estimation of parasitics. Because of low power consumption of the chip, no heat sinking is required. Cavity size should be greater than 2.5mm x 1.5 mm to accommodate the design. As a result, the CFP (ceramic flat pack) was selected. The pin count depends on the number of I/Os, biasing, testing points and power supplies. It is estimated that about 25 pins are required. Thus, either the 24-pin CFP or the 44-pin CFP could be selected.

Chapter 4
Behavioral Modeling

4.1. Phase Detector

Phase Detector, as its name implies, is capable of detecting a phase difference between two inputs. The output is proportional to the phase difference. The simplest implementation of a phase detector is an XOR gate. A phase frequency detector, on the other hand, is capable of detecting both phase and frequency differences.

A type 4, phase frequency detector has been used for this PLL. In this architecture there are a total of four possible states.

If the frequency of the REF signal is lower than the frequency of the FBK signal, then the DOWN signal goes high. This is an entirely digital block and can therefore be modeled behaviorally by identifying the events that result in a change of the state of the phase detector. The following events were defined to model the phase detector.

Event1: (Rising edge of REF) AND (NOT RESET) sets A after delay.
Event2: (Rising edge of FBK) AND (NOT RESET) sets B after delay.
Event3: (A) AND (B) sets RESET after delay.
Event4: (RESET) resets (A) and (B) after delay.
Event5: (A) sets (NOT UP) after delay.
Event6: (B) sets (DOWN) after delay.

One important issue concerning phase frequency detectors is the phase resolution that they are able to detect. It is important to model the correct rise time, fall time and delay associated with the transition of every state. Mismatch in delay in the generation of the

UP and DOWN signals results in unequal times in pump-up and pump-down cycles and hence causes jitter in the feedback clock during phase lock. Delay causes jitter in the output and locking time of the PLL will be increased. Figure 4.1 shows the simulation results of the behavioral model of the phase detector. All the simulations in this chapter were done using the Advance MS (ADMS) simulator.

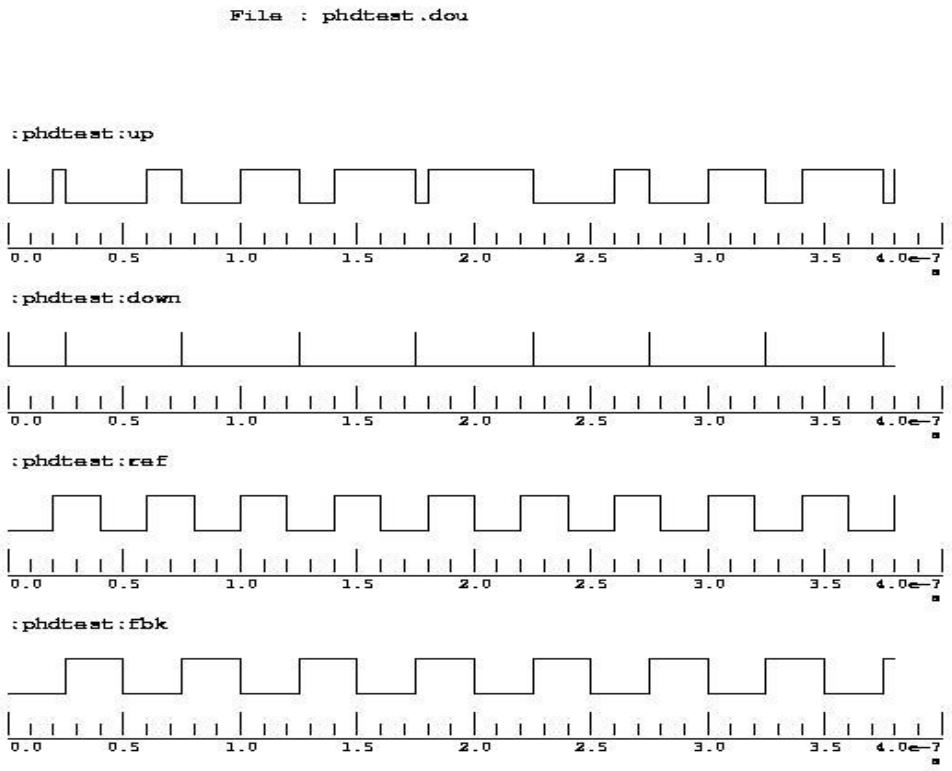

Figure 4.1 Phase detector simulation

4.2. Charge Pump and low pass filter

The charge pump, pumps current into the low pass filter. This action is controlled by the UP and DOWN signals from phase detector. During phase lock, ideally one would want equal pump-up and pump-down cycles. The difference between the pump-up and pump-down currents contributes to jitter in the feedback clock.

The behavioral model of the charge pump is a mixed signal model. The inputs are digital signals coming from PFD (phase frequency detector) block while current sources and switches are modeled as analog blocks. Figure 4.2 shows the simplified diagram of the charge pump.

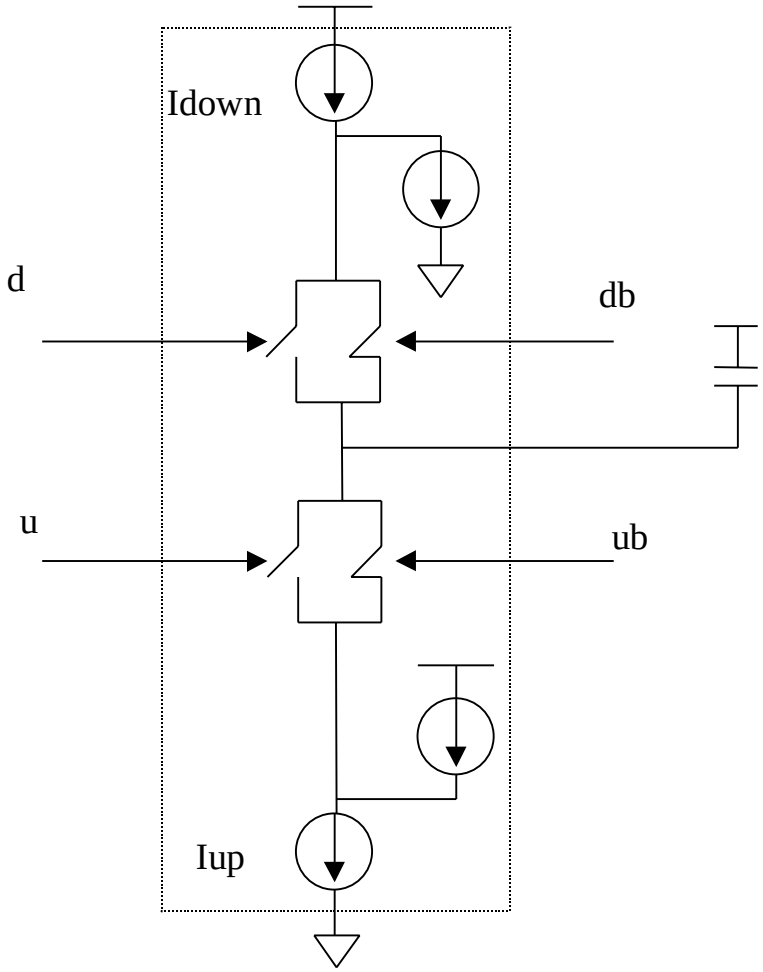

Figure 4.2 Charge Pump

A single capacitor and the active damping factor control block represent the low pass filter in this design. The damping factor controller is modeled as part of the voltage to current (V-I) converter block. The mixed mode simulation environment of VHDL-AMS supports the simultaneous simulation of behavioral blocks and analog circuit equations.

In order to be able to model the effect of switching delay in the operation of the charge pump, turn-on and turn-off delays were considered in the switch model as well as turn-on and turn-off resistance of the switch.

4.3 Voltage Controlled Oscillator

The VCO comprises of four different sub-blocks:

- V-I converter
- Damping factor controller
- Current controlled ring oscillator (CCO)
- Voltage level shifter

4.3.1 V-I converter

The V-I converter was modeled as a non-ideal voltage controlled current source. In the actual circuit, a transistor based current source would perform as the V-I converter. The effect of the following non-idealities were considered :

- Threshold voltage of MOS transistors (V_{th})
- Output conductance of the current source (G)

By changing the gain of the current source (G_m) and the above non-linearity factors, one can model the behavior of the V-I converter with good precision.

4.3.2 Damping Factor Controller

The DFC (damping factor controller) block consists of two parallel voltage controlled

current sources and two sets of complementary switches. In terms of behavioral modeling, we can say that it is a mixed signal block, which has both digital inputs and analog terminals. Figure 4.3 shows the block diagram of this block. $V_{control}$ is the same input to the VI converter block. The digital input signals are UP and DOWN and their complements are coming from the PFD block. They divert and add current to the input of the CCO and act as a damping factor controller in the system.

When all four input signals u, ub, d, and db are static (no pulse generated), the left branch current flows to ground through the drain current source while the right branch current adds to the main CCO driving current. When pulses are applied to the differential pairs due to a phase error at the phase detector inputs, both branch currents flow either to ground or to the CCO, depending on the polarity of the phase error. The amount of current that is subtracted from or added to the CCO driving current is proportional to the magnitude of the phase error. The loop stability increases by adding the damping factor controller block to the loop [13].

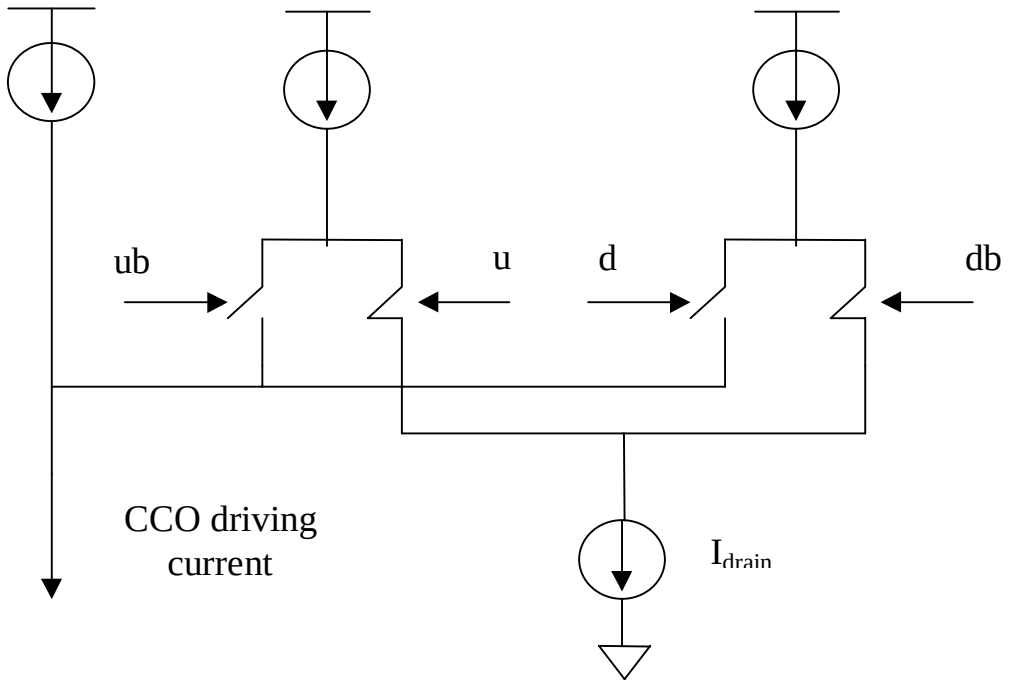

Figure 4.3 Damping factor control block diagram

4.3.3 Current controlled oscillator

The current controlled oscillator (CCO) used in this design is a balanced five stage, single-ended ring oscillator shown in Figure 4.4. In order to model this CCO one can either use the structural model shown in Figure 4.5 or develop a behavioral model based on the electrical model of the CCO.

The problem with structural model approach is simulation time and convergence of the oscillator response. On the other hand, this modeling technique does not have enough accuracy and we can not model input resistance and threshold voltage effect of the oscillator.

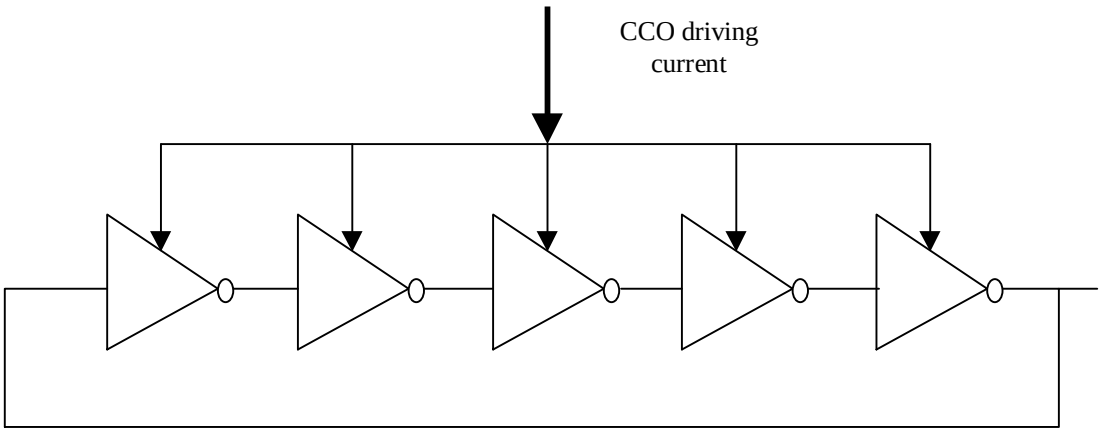

Figure 4.4 CCO structural model

The second method of modeling is based on behavioral modeling, in this method the *current-frequency* characteristic of the oscillator is modeled using a linear or non-linear equation. The input impedance of the oscillator is modeled using a simple resistor. In order to increase the accuracy of the modeling, a second order equation for *I-F*

characteristic was used. The coefficients of this equation can be parameters of the model.

Figure 4.5.a, 4.5.b show the equivalent circuit and block diagram of the current controlled oscillator and modeling parameters [14,16].

Figure 4.6 shows the frequency-current characteristic of the CCO. This curve was plotted using the data obtained from CCO circuit simulation. The CCO gain in this figure can be used for a behavioral model of the CCO.

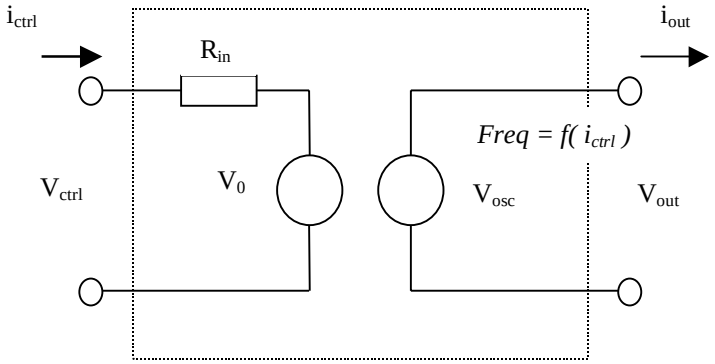

Figure 4.5.a Equivalent circuit diagram

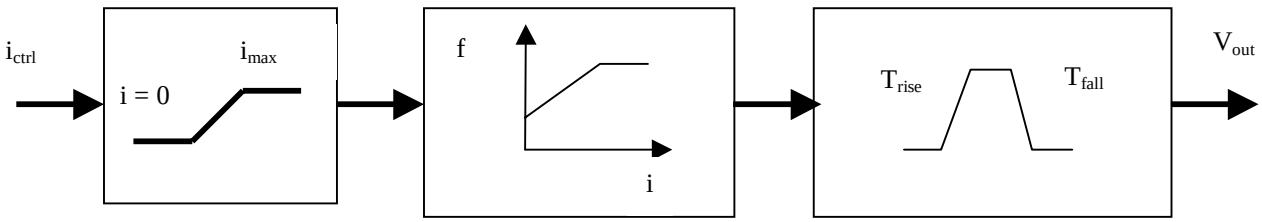

Figure 4.5.b Current controlled oscillator block diagram

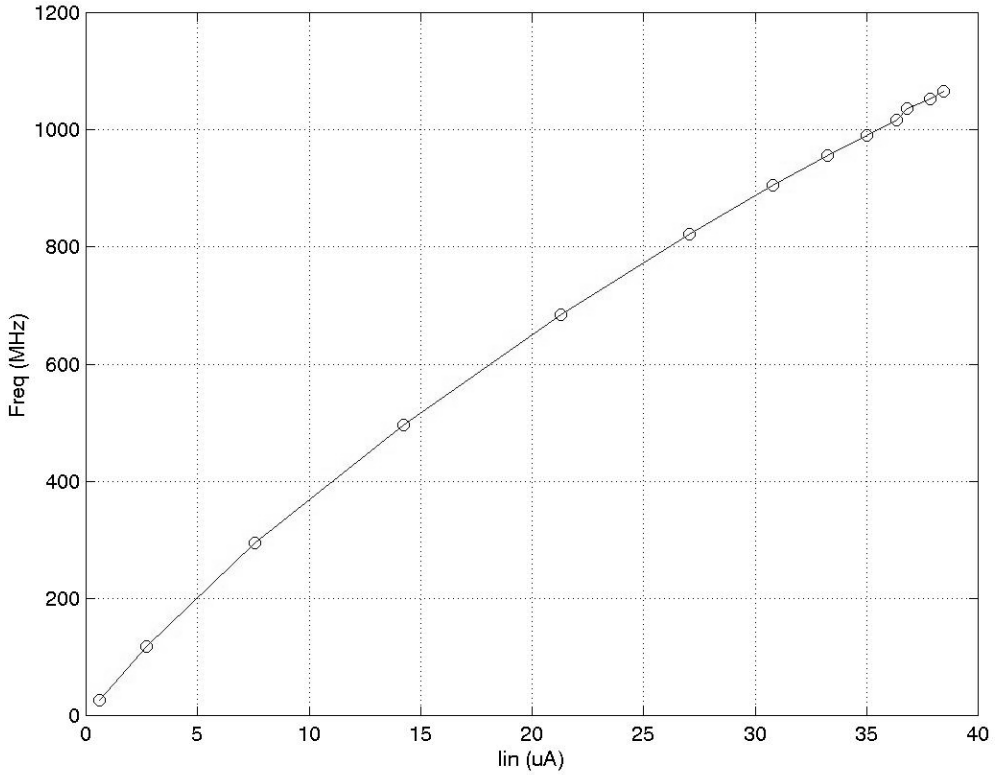

Figure 4.6 CCO *F-I* curve

4.3.4 Voltage Level Shifter

The level shifter (LS) block changes the level of the oscillator as well as buffering the output of the oscillator before feeding it into frequency divider.

The frequency divider is usually implemented using flip-flops which need fairly sharp clock edges for triggering. The output of CCO usually does not have such a characteristic. The other functionality of LS block is to provide sharp edges for proper functioning of the divider.

The frequency divider is a digital block while the output of the oscillator is an analog

signal, the analog to digital transformation is performed in LS block. Figure 4.7 shows the input and output wave forms of the level shifter.

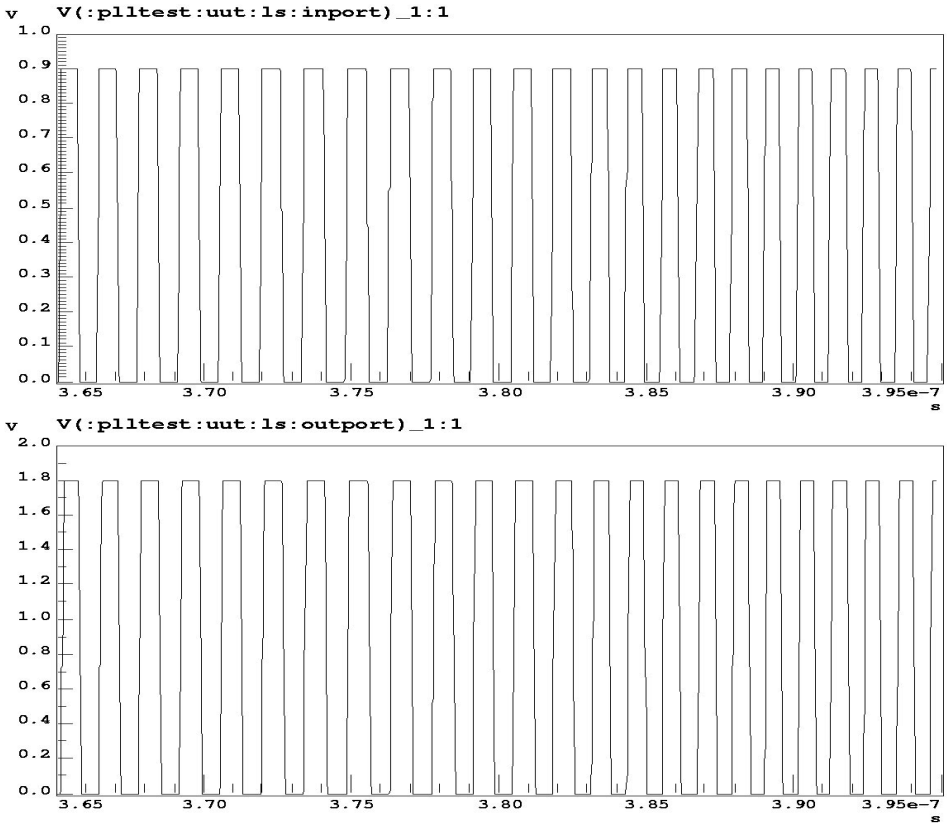

Figure 4.7 level shifter simulation

.

4.4 Simulation results

Figures 4.8 and 4.9 show the simulation wave forms of the input (ref), output of divider (fbk), outputs of PFD (up, down) and also the control voltage (vcontrol) of PLL in two different conditions. In Figure 4.8, the condition after locking is shown, while in Figure 4.9 the PLL is still in transient time before locking.

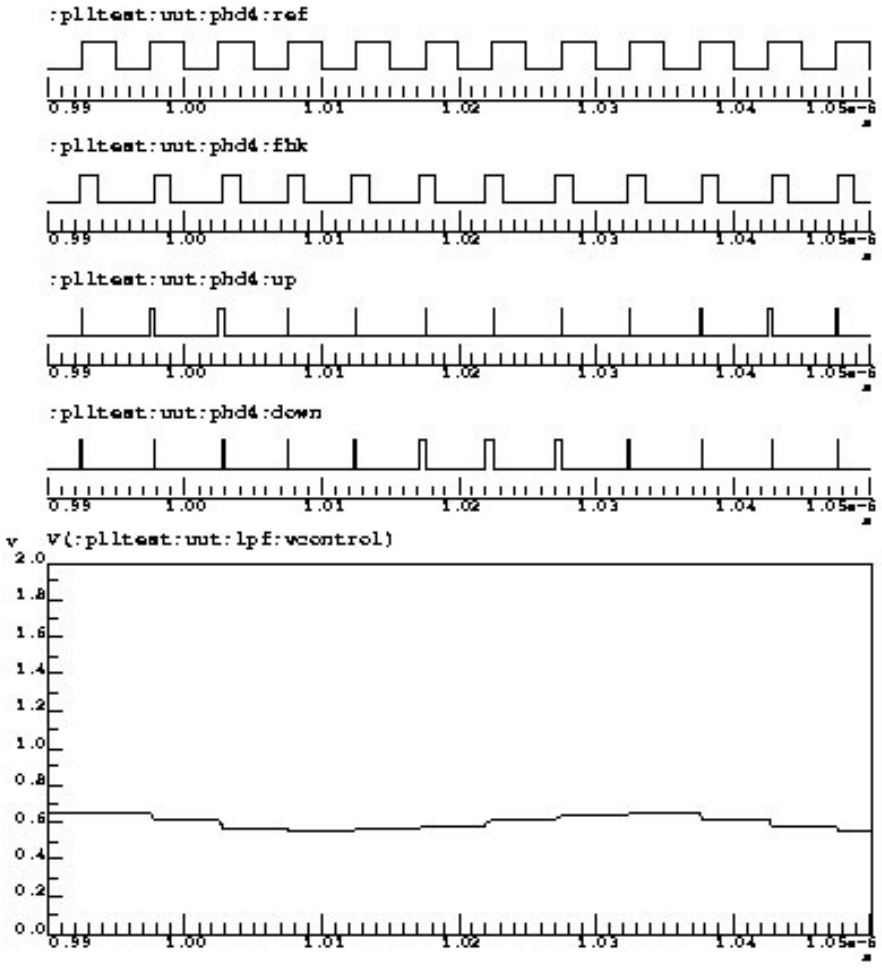

Figure 4.8 PLL simulation steady: after locking

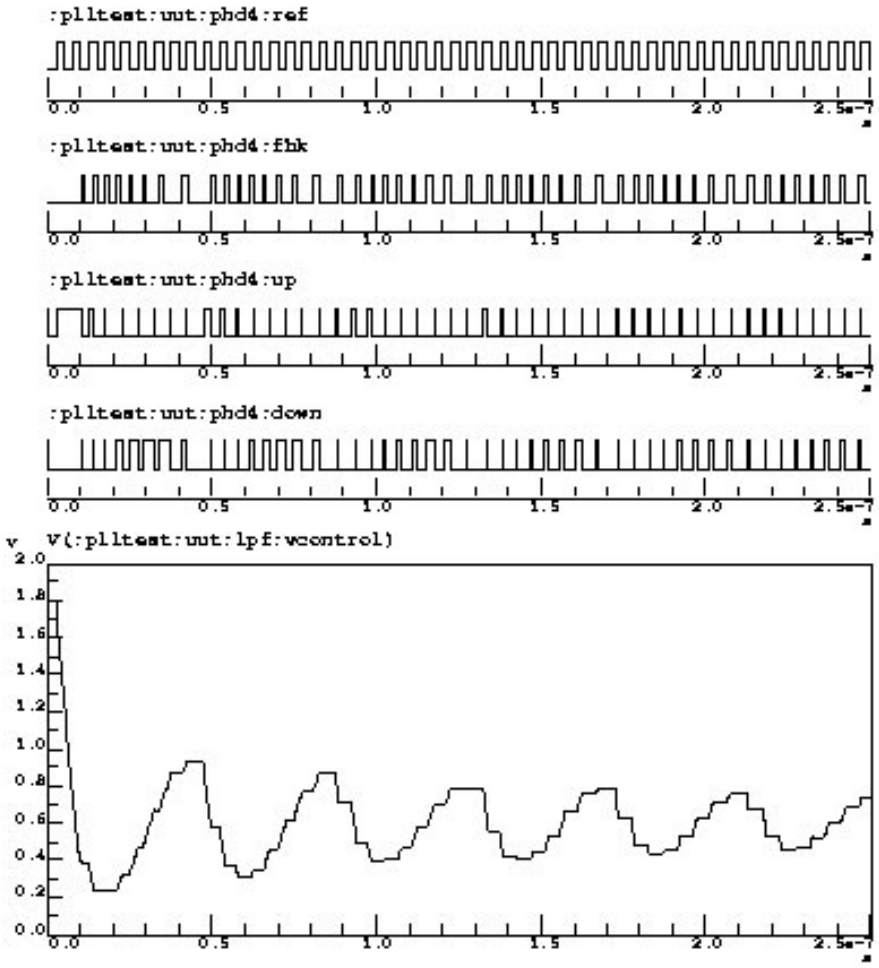

Figure 4.9 PLL simulation : Before locking

Chapter 5
Block Level Design

5.1 Block Schematic Capture

At this stage of the design flow, schematics have been created corresponding to the behavioral models developed in the previous chapter. For analog blocks, the circuit design starts with exploring possible circuit configurations by looking at the required specifications of the circuit and proceeding based on designer's experience and knowledge in analog design.

Digital synthesis maps digital behavior onto digital gates that are arranged in a rather constrained topology. The simple nature of gates combined with the constrained topology makes synthesis feasible. With analog circuitry, the fundamental building blocks are much more complex and varied and the topology is completely unconstrained. These two factors make analog synthesis a fundamentally much more difficult problem than digital synthesis. Analog synthesis so far has resisted all attempts at automation except in limited cases, such as analog filters. Work continues, but having universal analog synthesis is still in the future.

Without analog synthesis, analog design is done the old fashioned way, with designers manually converting specifications to circuits. While this allows for more creativity, it also results in more errors and requires a process of trial and error with running simulations, which is time consuming.

To overcome this problem, mixed-level simulation is employed in a top-down design

methodology for analog and mixed-signal circuits (this represents a significant but essential departure from the digital design methodology). Mixed-level simulation is required to establish that the blocks will function as designed in the overall system. To verify a block with mixed-level simulation, the model of the block in the top-level schematic is replaced with the transistor level schematic of the block before running the simulation. The system described at a high level, acts as a test-bench for the block, which is described at the transistor level. Thus, the block is verified in the context of the system, and it is easy to see the effect of imperfections in the block on the performance of the system. Mixed-level simulation requires that both the system and the block designers use the same simulator and that it be well suited for both system- and transistor-level simulation.

Mixed-level simulation allows a natural sharing of information between the system and block designers. When the system-level model is passed to the block designer, the behavioral model of a block becomes an executable specification and the description of the system becomes an executable test bench for the block. When the transistor level design of the block is complete, it is easily included in the system-level simulation by the chip architect.

Mixed-level simulation is the only feasible approach currently available for verifying large complex mixed-signal systems. Some propose to use either timing simulators (sometimes referred to as fast or reduced accuracy circuit simulators) or circuit simulators running on parallel processors. However, both approaches defer system-level verification until the whole system is available at transistor level, and neither provides the performance nor the generality required to verify most mixed-signal systems.

Once a block is implemented, the designer could update the models that represent it to more closely mimic its actual behavior. This improves the effectiveness of mixed-level and system-level simulation and is referred to as bottom-up verification. To reduce the chance of errors, it is best done during the mixed-level simulation procedure.

In this way, the verification of a block by mixed-level simulation becomes a three-step process. First the proposed block functionality is verified by including an idealized model of the block in system-level simulations following a detailed behavioral model. Then, the functionality of the block as implemented is verified by replacing the idealized model with the netlist of the block. This also allows the effect of the block's imperfections on the system performance to be observed. Finally, the netlist of the block is replaced by an extracted model. Figure 5.1 shows these steps. By comparing the results achieved from simulations that involved the netlist and extracted models, the functionality and accuracy of the extracted model can be verified. From then on, mixed-level simulations of other blocks are made more representative by using the extracted model of the block just verified rather than the idealized model.

In a top-down design process, SPICE-level simulation is used in order to get its benefits without incurring its costs. All blocks are simulated at the transistor level in the context of the system (mixed-level simulation) in order to verify their functionality and interface. Areas of special concern, such as critical paths, are identified up front and simulated at the transistor level. The performance of the circuit is verified by simulating just the signal path or key pieces of it at the transistor level. Finally, if start-up behavior is a concern, it is also simulated at the transistor level. The idea is not to eliminate SPICE simulation, but to reduce the time spent in SPICE simulation while increasing the effectiveness of simulation in general by careful planning.

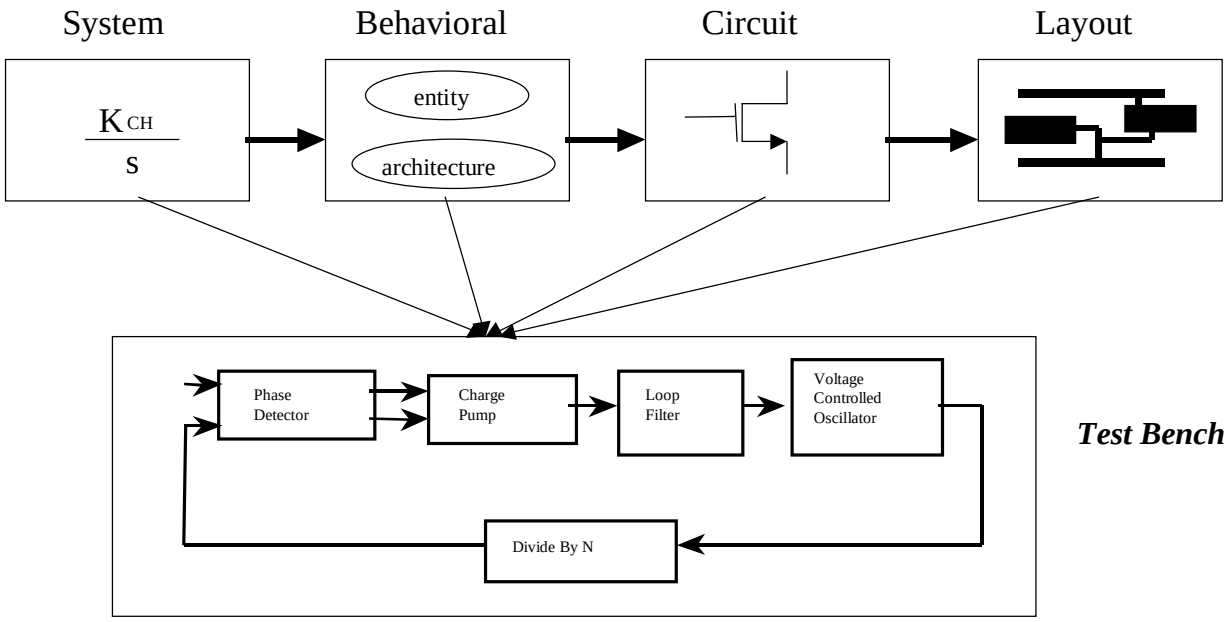

Figure 5.1 Mixed signal test bench

In the following sections, the circuit level design of PLL block will be described. For each block, first a simple test bench is setup in the Cadence environment to be able to design the basic functionality of the block and later, the design will be verified in the overall system test bench. This will be done for both transistor level and extracted layout view.

5.2 Phase Frequency Detector

The PLL uses the three-state PFD shown in Figure 5.2.a. The PFD is designed to generate symmetrical charge-up (u) and charge-down (d) pulses. The potential dead zone is eliminated by the propagation delay of the four-input NOR gate, which produces a minimum pulse width at the PFD output even when the phase error is zero [15].

The NOR gates were designed so as to give 3.2% phase error while maintaining low power design conditions. Figures 5.2.b and 5.2.c show the NOR gates designed for PFD.

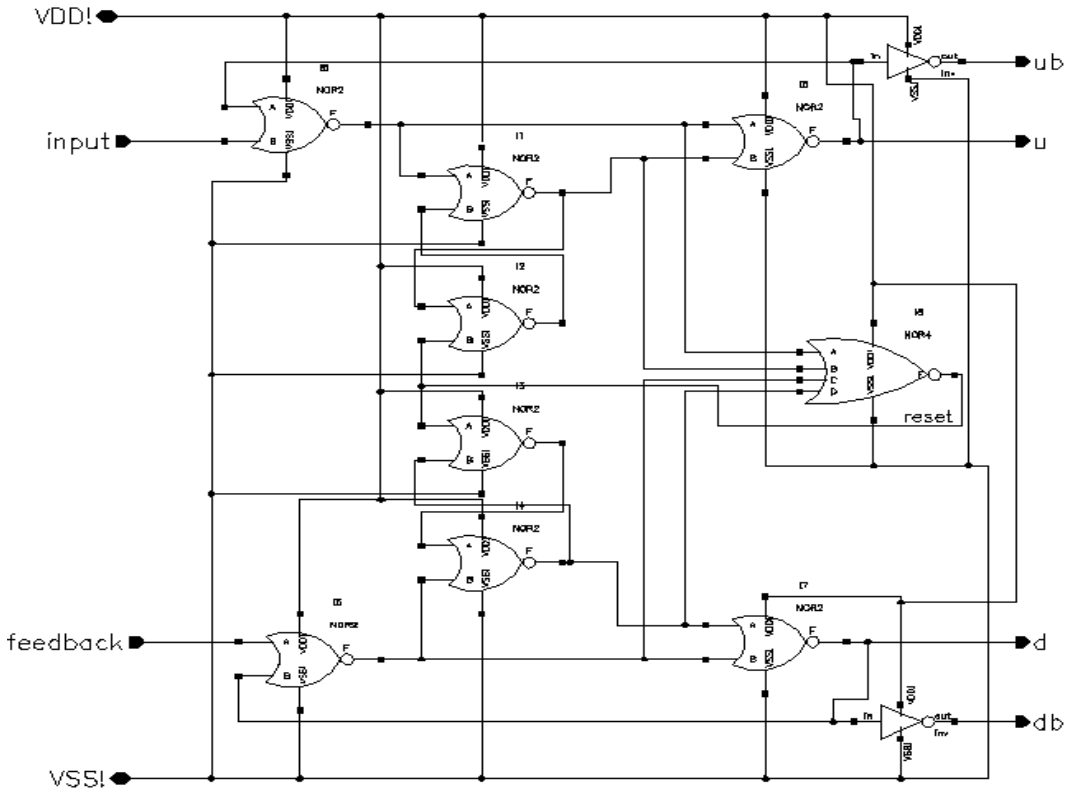

Figure 5.2.a Schematic of phase detector

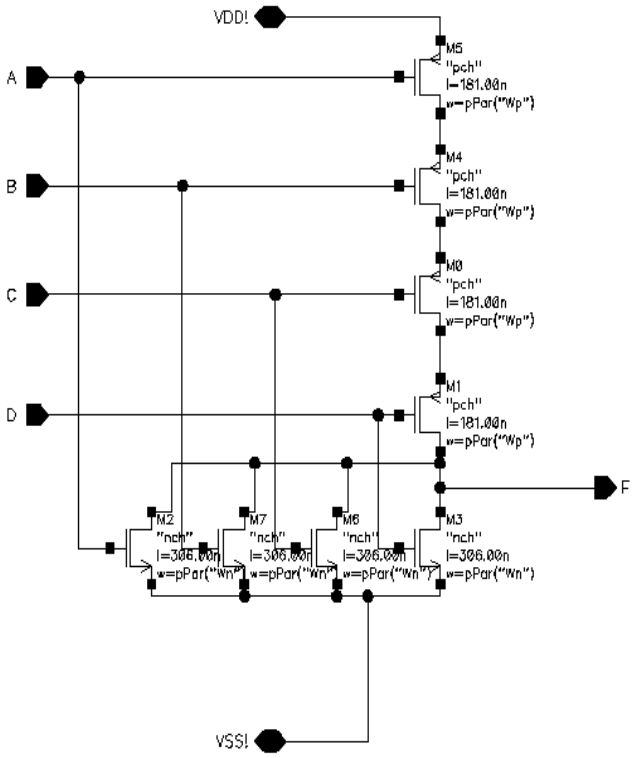

Figure 5.2.b 4 input NOR gate

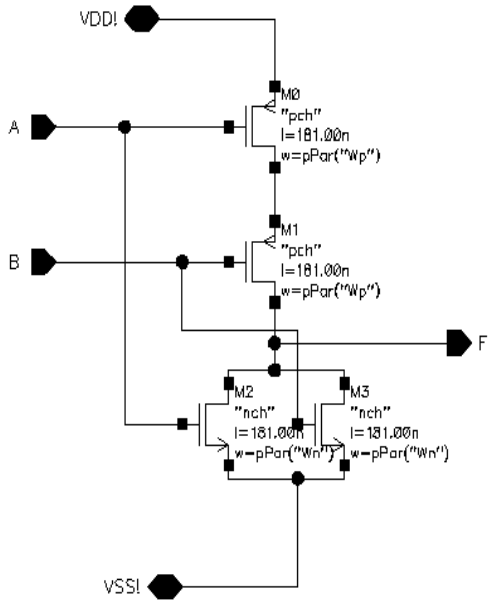

Figure 5.2.c 2 input NOR gate

Figure 5.3 shows the layout of phase detector cell. It is necessary to simulate the layout after extraction to compare the result with schematic design and design specification for the block. Figure 5.4 shows the simulation result for the transistor and layout view.

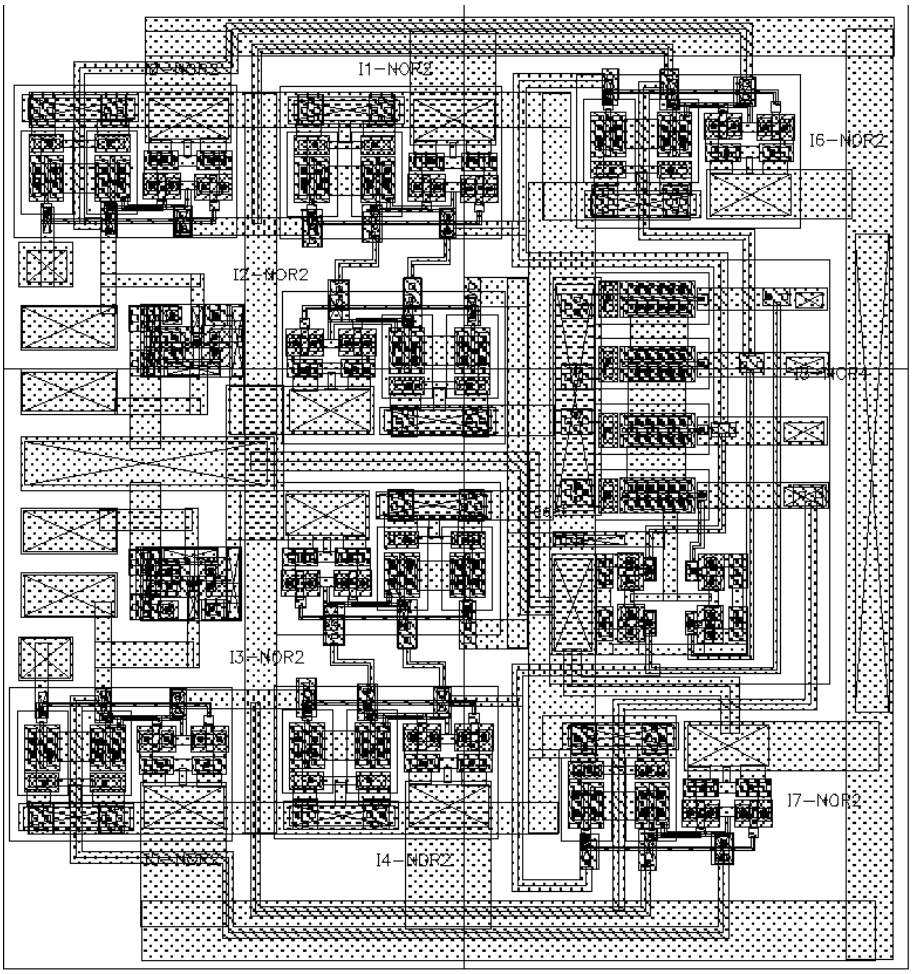

Figure 5.3 PFD layout

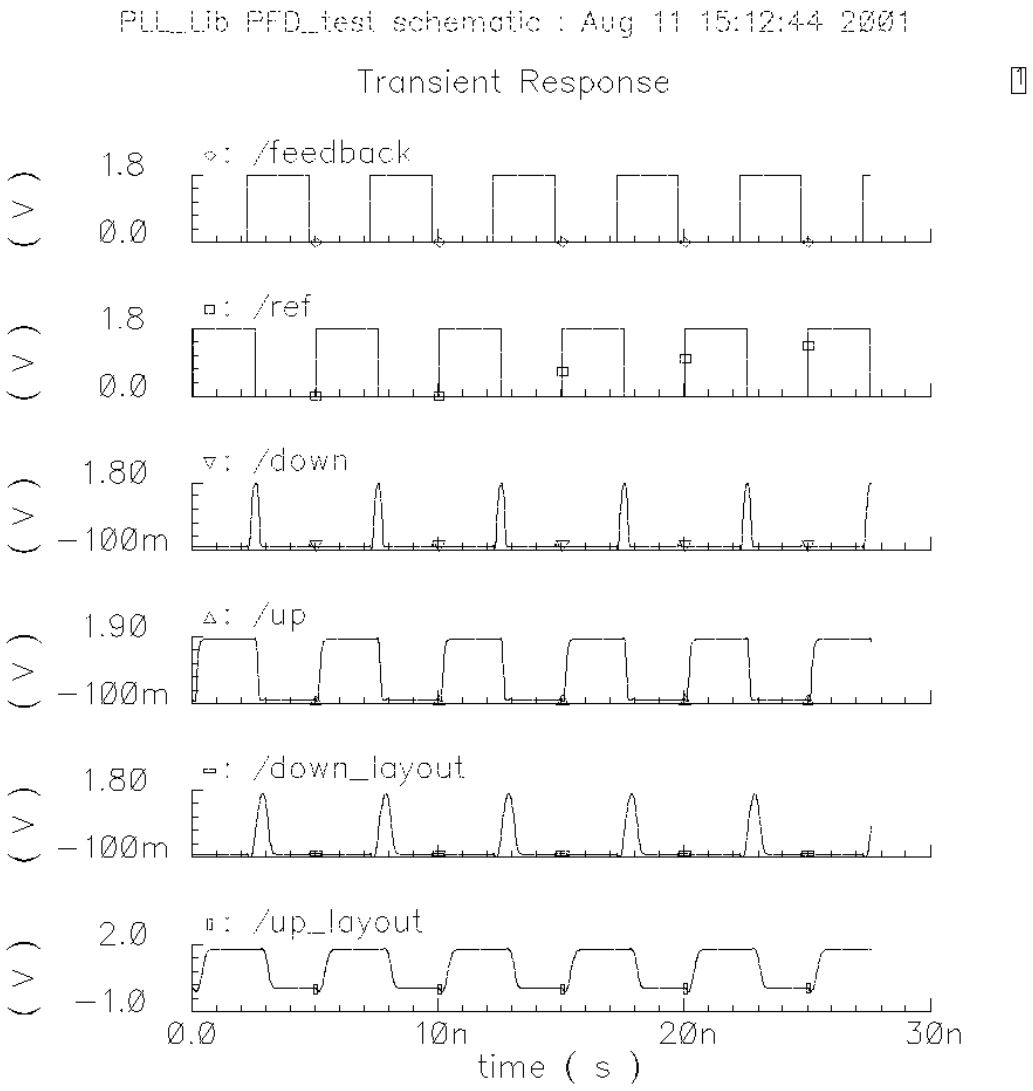

Figure 5.4 phase detector simulation

5.3 Charge Pump

The charge pump comprises complementary current sources, switches, and source followers. Figure 5.5 shows the circuit diagram of the charge pump circuit. The source followers are driven by the *Vcontrol* output; their threshold voltages determine the voltages across the complementary switches when they are OFF. This technique reduces the current error between charge up and down due to the mismatch of charge sharing

effects when the switches are turned ON [16]. Stated another way, the current mismatch is ideally independent of *Vcontrol*. Figure 5.6 shows the layout view of the charge pump cell. Figure 5.7 shows the simulation result. As it can be seen, the effect of mismatch is reduced in the design.

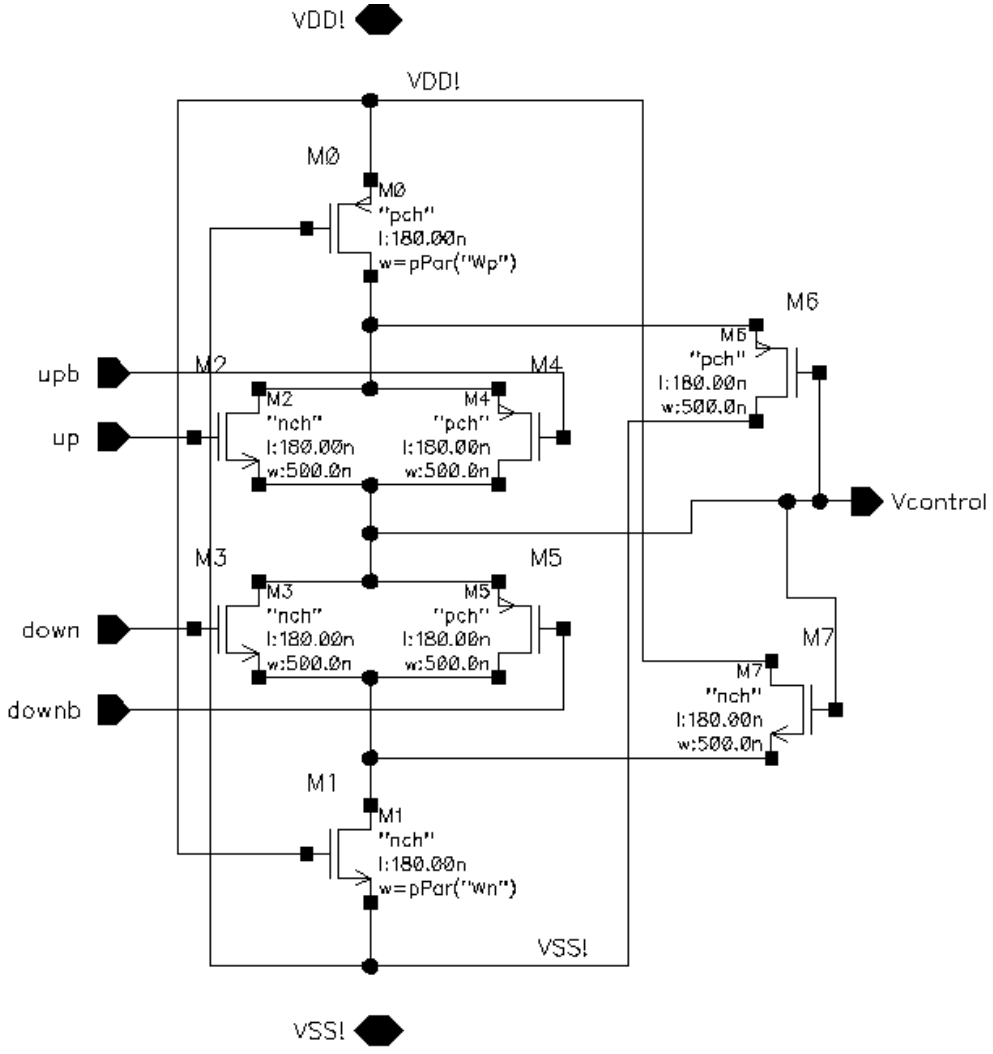

Figure 5.5 Charge pump circuit

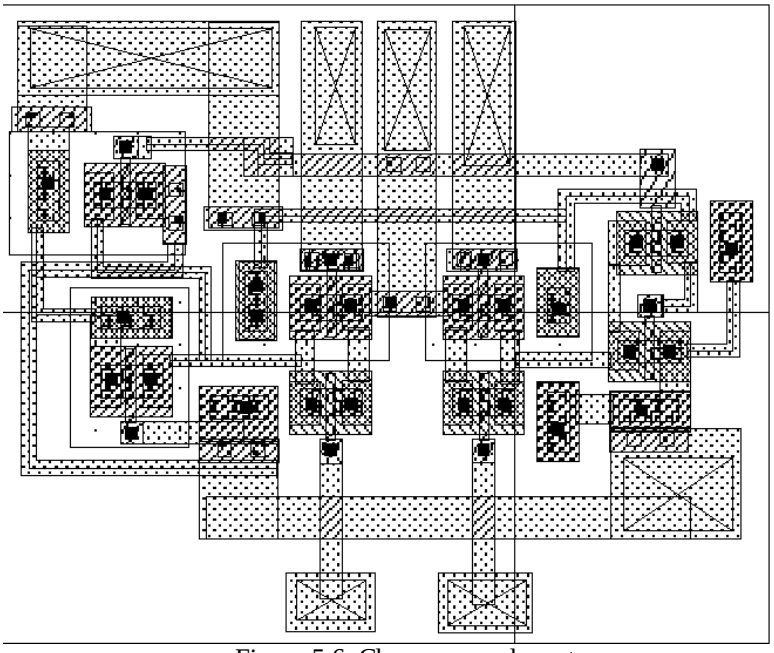

Figure 5.6 Charge pump layout

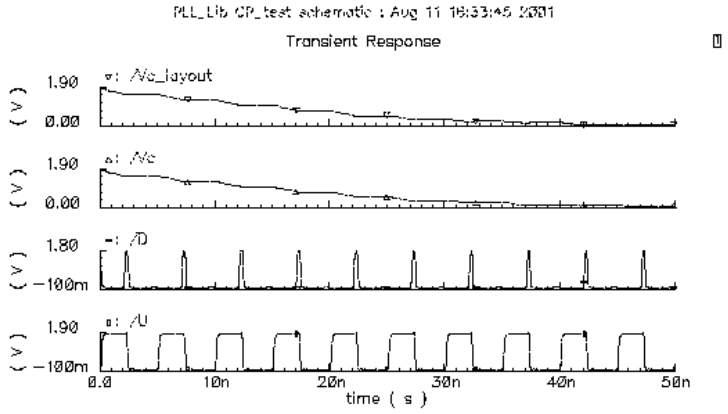

Figure 5.7 Charge pump simulation

5.4 Voltage Controlled Oscillator

The VCO comprises four sub-circuits: a V-I converter, current-controlled ring oscillator (CCO), voltage-level shifter, and damping factor control circuit.

While it is desirable to implement the PLL in differential form so as to suppress the effect of common mode noise, low supply voltages (<3) limit the headroom, making it difficult to utilize differential control for CMOS oscillators.

Thus, the PLL circuit is single-ended, but it employs current mode control signals to lower the sensitivity to supply and substrate noise.

5.4.1 V-I converter

Transistors M1, M4 and M5 (in Figure 5.8a), form a PMOS regulated cascode V-I converter that sources drive current to the CCO; compensation capacitor C1 stabilizes the regulated loop and suppresses the injection of high frequency supply noise into the CCO. Figure 5.8 shows the schematic and layout view of the V-I converter circuit. Owing to the regulated cascode, the small signal output resistance of the CCO driving current source is very high. Hence, the driving current is nearly independent of the supply voltage for a given input voltage *Vcontrol*, and excellent power supply noise rejection characteristics are achieved. Figure 5.9 shows the I_{out}-$V_{control}$ characteristics for V-I converter [13].

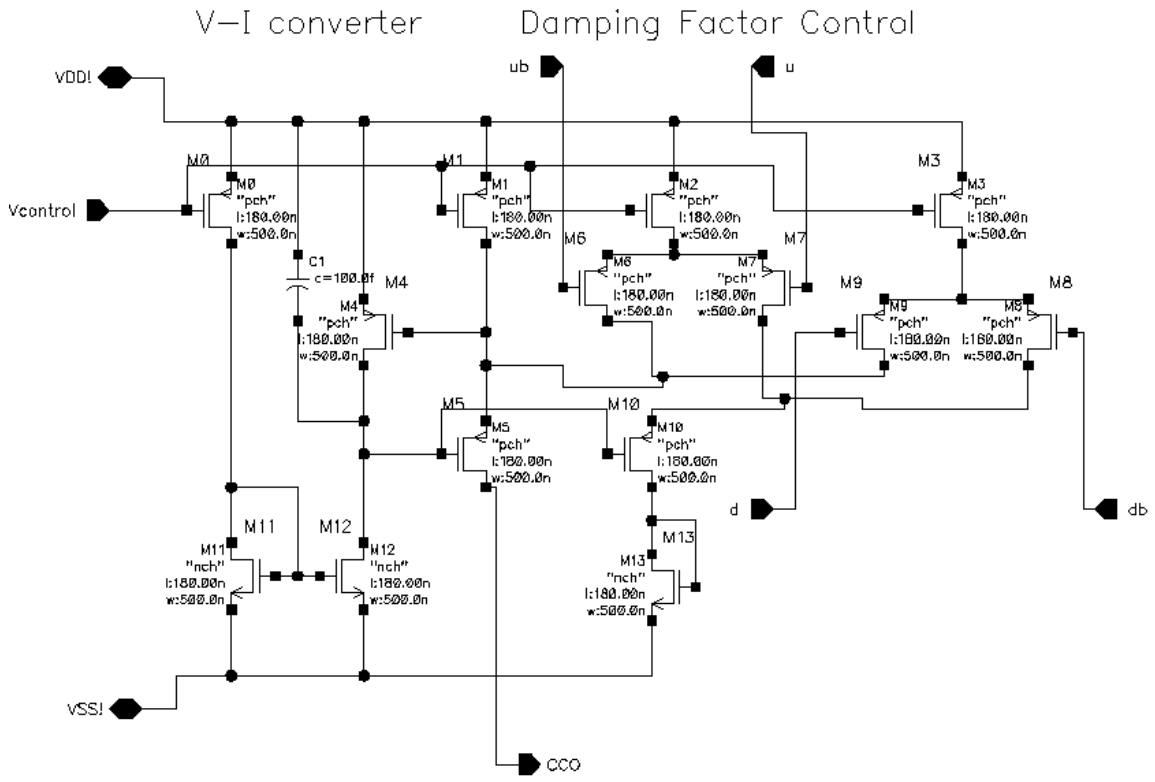

Figure 5.8.a V-I converter and Damping Factor Control

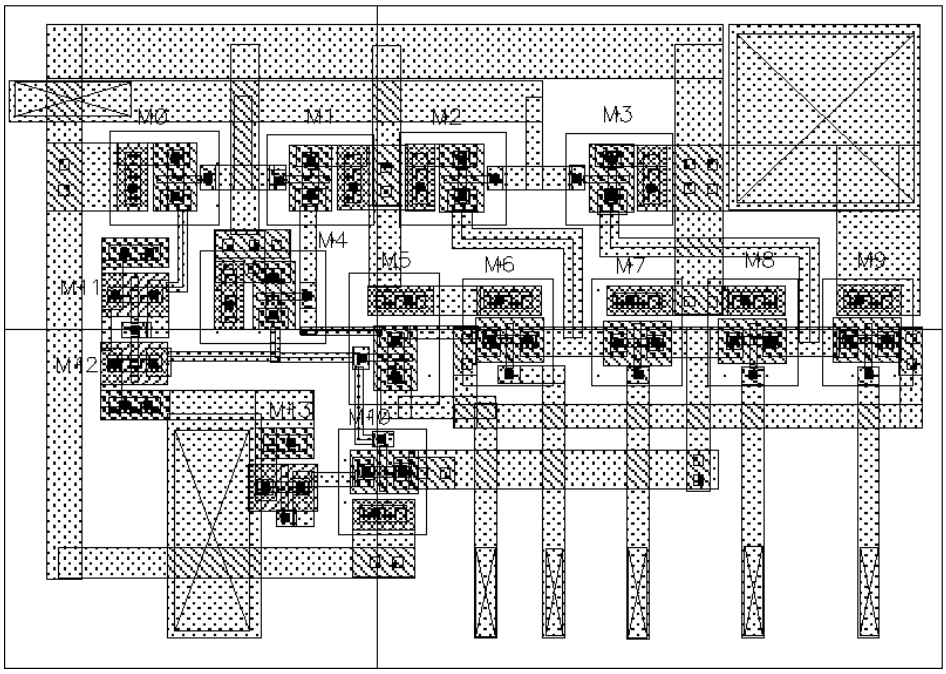

Figure 5.8.b Layout view

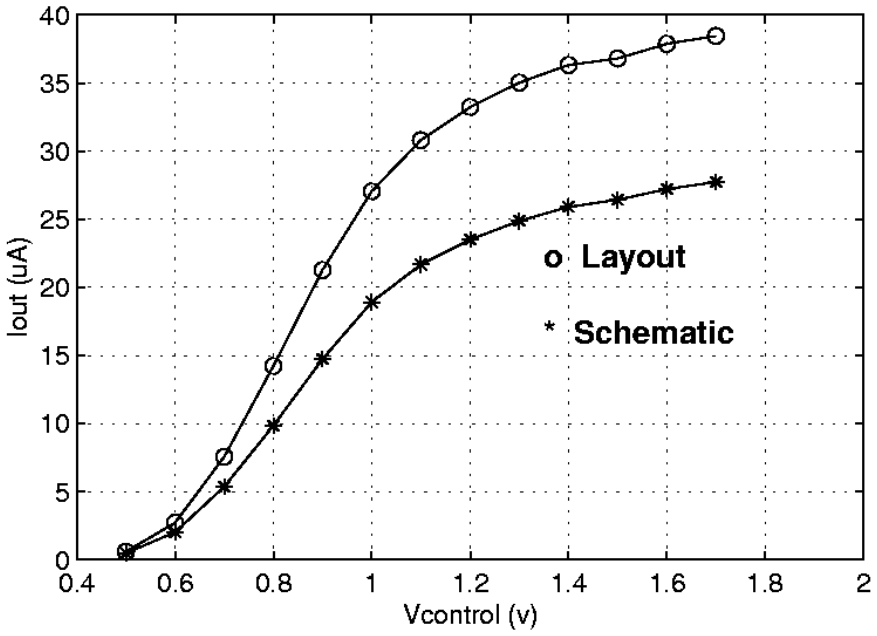

Figure 5.9 V-I converter input-output characteristics

5.4.2 Damping factor control

The damping factor control circuit is biased using PMOS branches driven by Vcontrol (Figure 5.8.a). When all four input signals **u**, **ub**, **d** and **db** are static (no pulse generated), the left branch current flows to ground through the diode-connected NMOS transistor while the right branch current adds to the main CCO driving current. When pulses are applied to the differential pairs due to a phase error at the PFD inputs, both branch currents flow either to ground or to the CCO, depending on the polarity of the phase error. The amount of the current that is subtracted from or added to the CCO driving current is proportional to the magnitude of the phase error and to the static current level, which depends on the operating frequency. The loop stability and equivalent damping factor increases with the magnitude of the dc bias currents applied to the damping factor control circuit.

5.4.3 Voltage controlled oscillator (VCO)

The gain of the VCO and the PD/charge-pump circuits are important factors in

determining the loop bandwidth (BW). Since the loop filter output voltage is limited in a low-voltage design, high VCO gain is required to achieve a wide operating frequency range [14]. The VCO gain is chosen based on a trade off between operating frequency range and loop bandwidth. The CCO used in the VCO is a balanced five stage, single-ended ring oscillator. Figure 5.10 shows CCO Lin-Freq characteristic diagram. The measured VCO gain is 1.75 GHz/V (Figure 5.11) with good linearity over a wide range of operating frequencies from 100 to 900 MHz. Figure 5.12 shows the the VCO Layout.

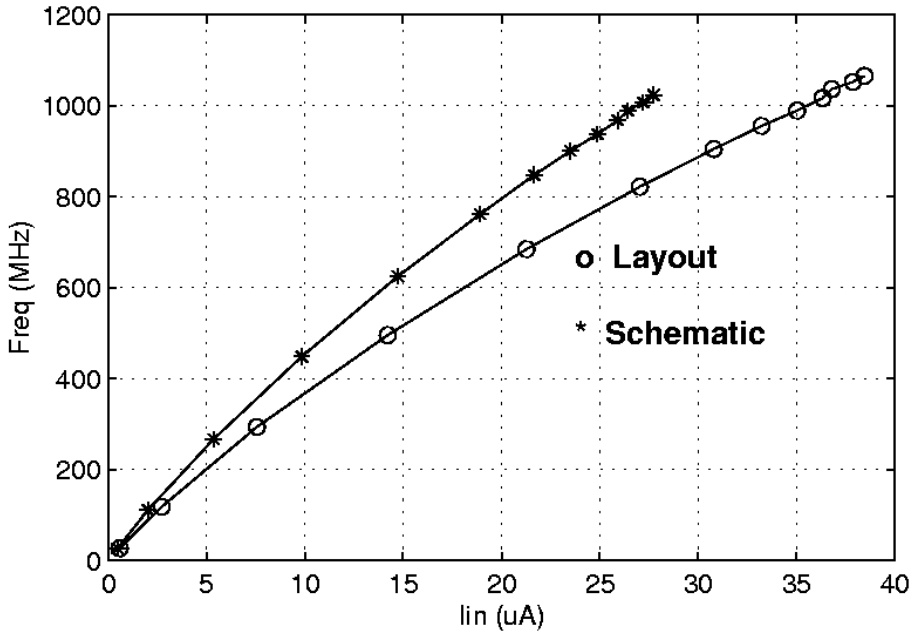

Figure 5.10 CCO input-output characteristics

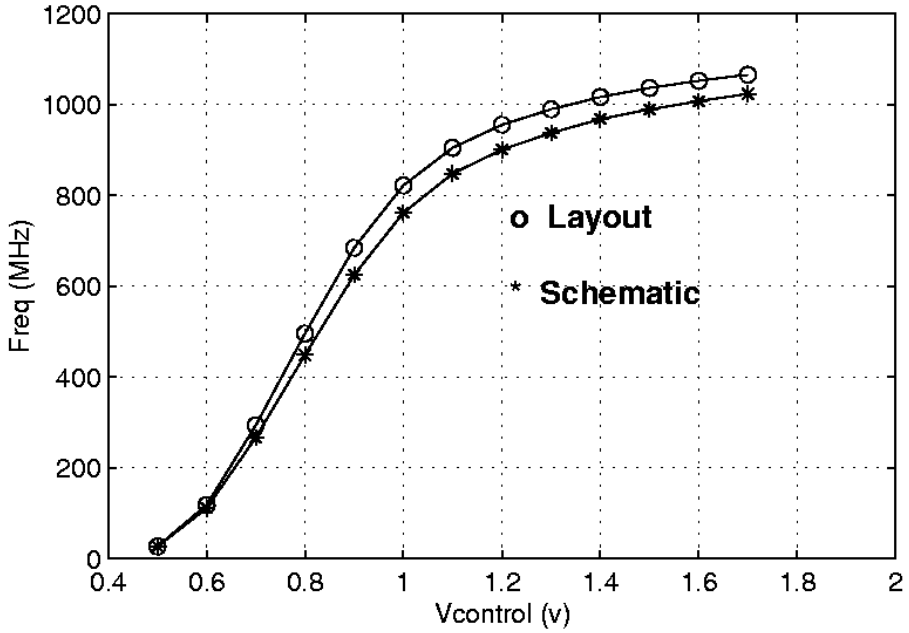

Figure 5.11 VCO characteristic

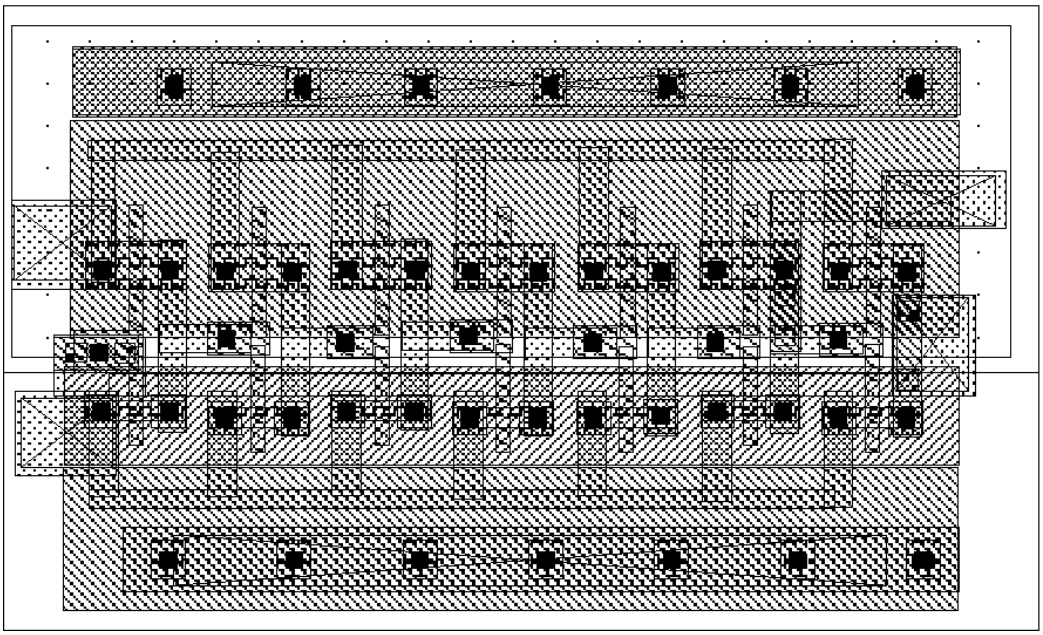

Figure 5.12 VCO layout

Figure 5.13 shows the transient response of the VCO (VCO-Output) as well as inputs of this block.

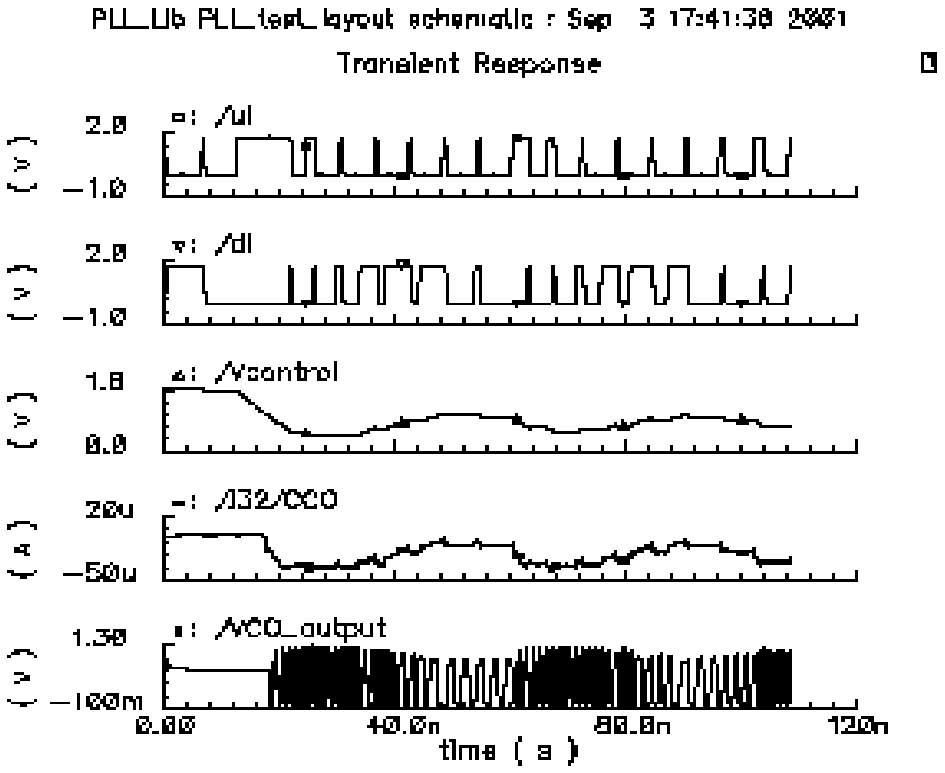

Figure 5.13 VCO Transient response

5.4.4 Voltage level shifter

Due to the voltage drops across M1 and M5 in Figure 5.8.a, the CCO operates from about VDD/2 to zero. Hence, a level-shifting buffer circuit follows the CCO to provide a rail-to-rail output signal. Figure 5.14 shows the schematic and layout view of the voltage level shifter.

Since the voltage level shifter is a buffer too, it should be able to provide enough current

for charging the output load while maintaining sharp edges required for triggering the counter.

Figure 5.15 shows the simulation results of the level shifter block.

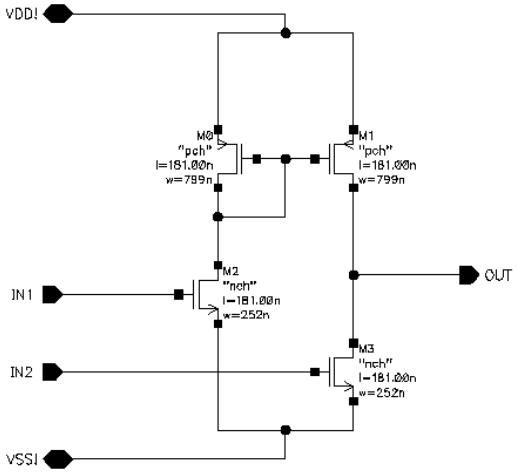

Figure 5.14.a schematic

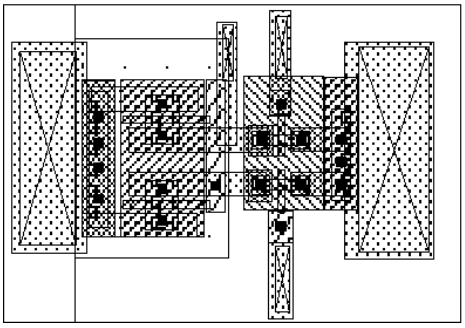

Figure 5.14.b layout

Figure 5.14 Level Shifter

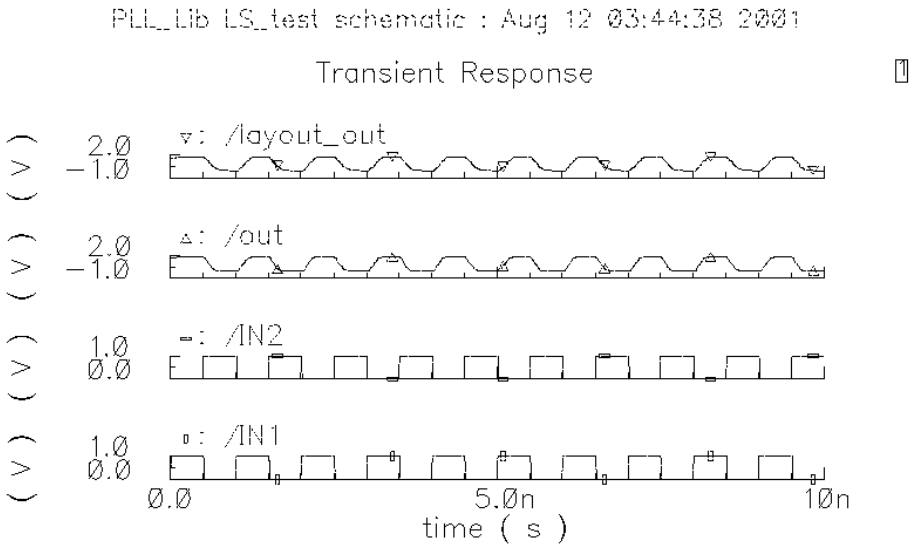

Figure 5.15 Level Shifter simulation

5.5 Frequency Divider

The frequency divider is the only pure digital block in the system. It was simply implemented as a divide by four counter. Figure 5.16 shows the steps required for designing digital blocks as part of a mixed signal circuit. It starts from writing the RTL code of the block in VHDL or Verilog and then synthesizing the RTL and then following the steps shown in Figure 5.16. Figure 5.17 shows the schematic diagram of the counter after synthesis and the layout; all the cells in this schematic are standard library cells. Figure 5.18 shows the simulation results for the frequency divider block.

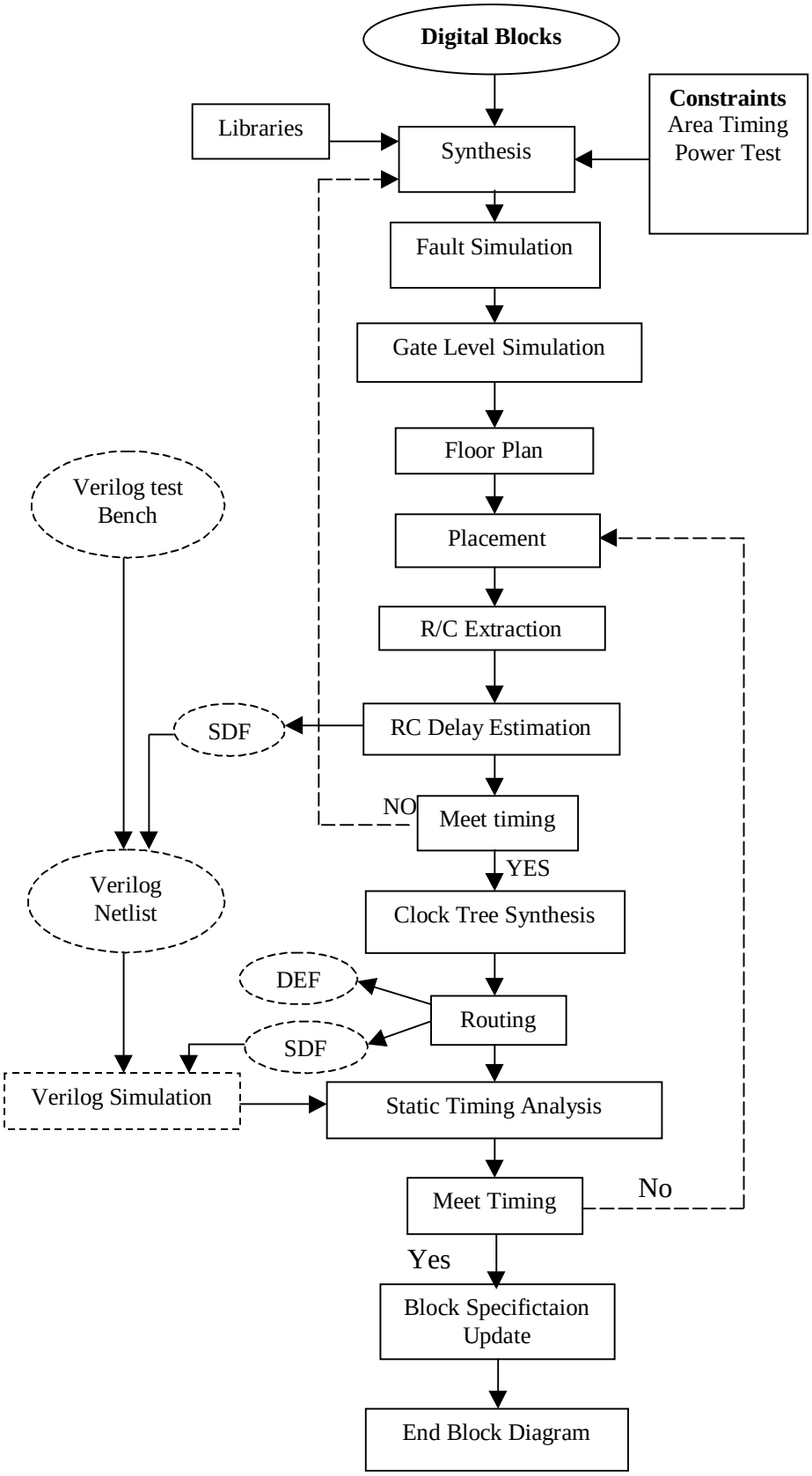

Figure 5.16 Digital design flow

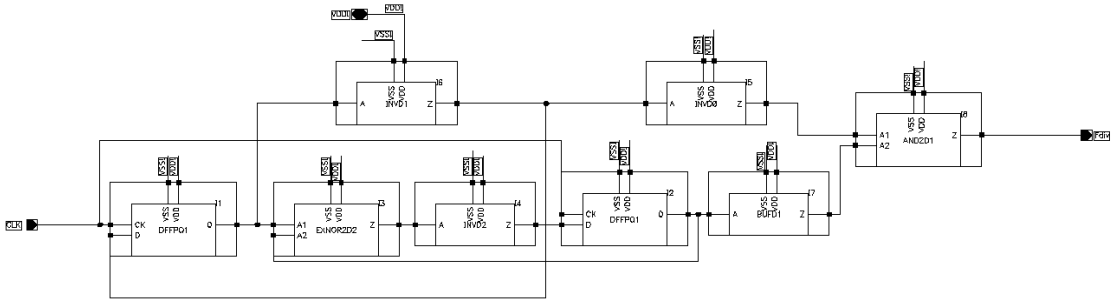

Figure 5.17.a Counter schematic after synthesis

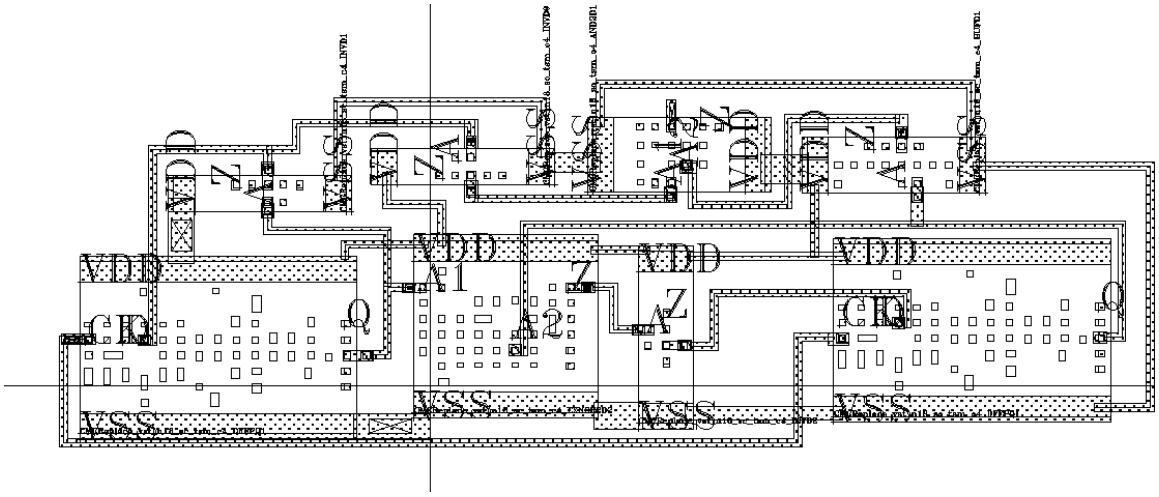

Figure 5.17.b Layout of counter

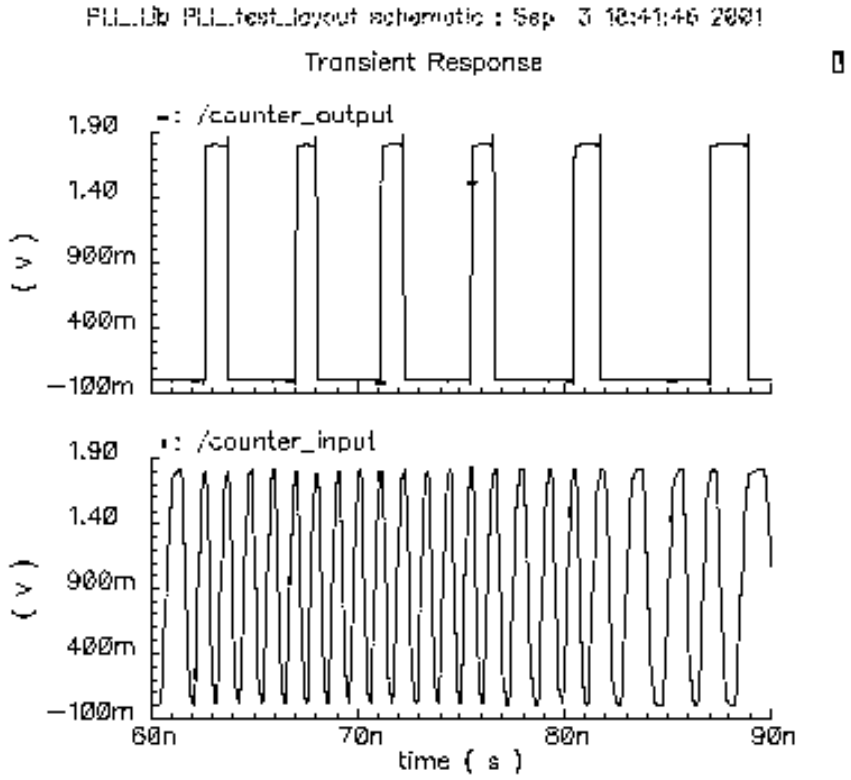

Figure 5.18 Frequency divider simulation result

5.6 Top-level layout design and simulation

After the design is verified block by block, top level placement and routing is done in Cadence/ Virtuso. DRC and LVS are performed to ensure correctness of the layout. The layout view of the entire chip is shown in Figure 5.19. Post-layout simulation results are shown in Figure 5.20.

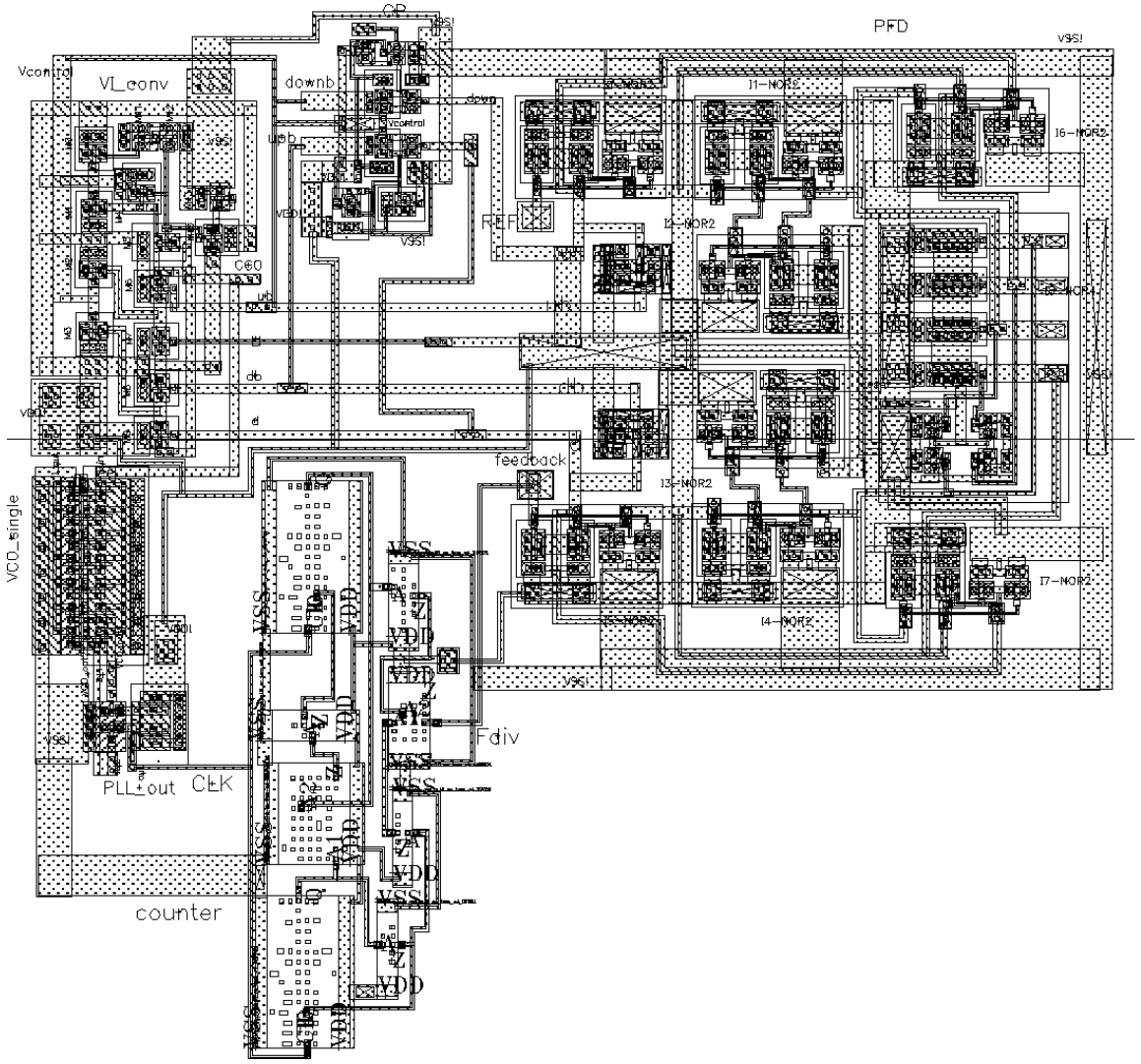

Figure 5.19 Chip layout

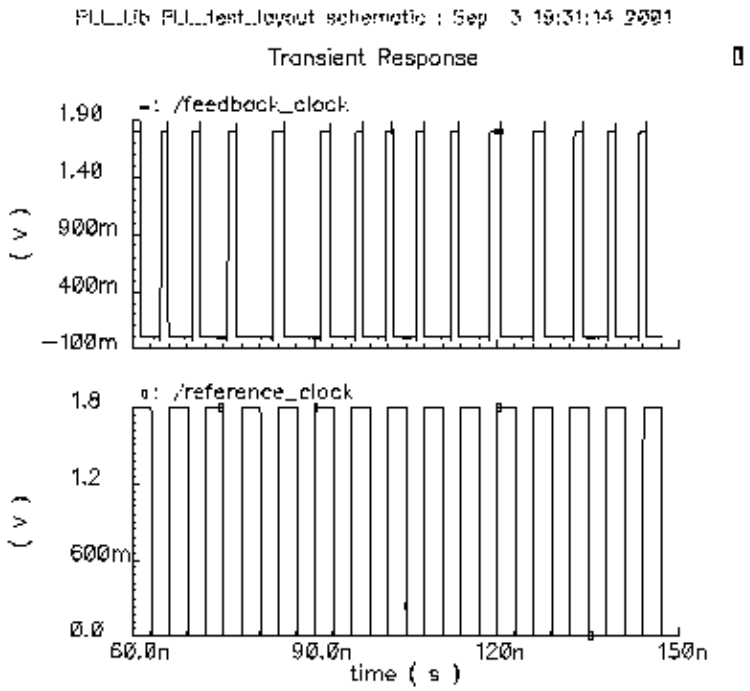

Figure 5.20 Post layout simulation

Chapter 6
Conclusion

6.1 Contributions

This research had two major phases, the first phase was a study of the existing digital and analog design flows and the problems associated with mixed signal system design. The major goal was defining features of a mixed signal methodology to address those problems. The second phase of the research was designing a mixed signal PLL based on the proposed design flow. The following are contributions achieved in this research:

1. A mixed signal design flow was introduced based on existing analog and digital design methodologies. The major features of the proposed methodology are:

- It is a top-down design and bottom-up verification methodology.
- It is based on analog and digital co-design.
- The same verification environment (test bench) can be used through different steps in the design. This is major feature for SOC (system on chip) design methodologies.
- It is based on the VHDL-AMS modeling language.

2. In order to examine the design flow, a charge pump PLL was designed based on the proposed methodology.

3. A detailed generic behavioral model of the PLL sub-blocks was developed. These models can be used as generic models for designing charge pump PLLs. This is a major step in mixed signal design automation.

Listed below are the conclusion and some of the facts regarding the use of the proposed design flow in a real design problem.

- Mixed signal design flow is useful in shortening design cycle.
- Mixed signal design flow can be useful in discovering integration problems at earlier stages of design.
- VHD-AMS can be used to develop a mixed signal behavioral model and verification environment.
- The VHDL-AMS simulations at different steps in the design flow would require standard or user defined packages.
- The detail steps of each design flow depend on the availability of tools and can be different in different design environments.

6.2 Future work

The digital design flow has already been established for many years and has been extensively used in industry while analog and mixed signal flow still following their first steps toward a fully automated and robust flow. In the digital flow, after RTL coding, the designer can rely on available tools for synthesis and layout. In the analog design, a gap still exists for moving from system to behavioral and from behavioral to circuit level design. Research is going on for developing an analog synthesis but still there is a long way toward a comprehensive analog synthesis solution. The other existing gap is between system level modeling tools such as Matlab/Simulink and behavioral modeling languages such as VHDL-AMS and Verilog-AMS. Even though these languages have the capability of system level modeling, most of system engineers prefer graphical environment of Simulink and available toolboxes of Matlab environment. Developing a tool that can fill this gap can be useful in shortening the design cycle period and also faster debugging.

Glossary

AMS	Analog-Mixed Signal
CAD	Computer Aided Design
CCO	Current Controlled Oscillator
CDMA	Code Division Multiple Access
CFP	Ceramic Flat Package
CMC	Canadian Microelectronics Corporation
CMOS	Complementary Metal Oxide Semiconductor
DEF	Design Exchange Format
DFT	Design For Test
DRC	Design Rule Check
IO	Input-Output
LEF	Library Exchange Format
LVS	Layout Versus Schematic
MOS	Complementary Metal Oxide Semiconductor
MS-HDL	Mixed Signal Hardware Descriptive Language
PFD	Phase Frequency Detector
PLL	Phase Locked Loop
ROL	Read Only Library
RTL	Register Transfer Level
SDF	Standard Delay Format
SOC	System On Chip
SPF	Standard Parasitic Format
TLF	Timing Library Format
VCO	Voltage Controlled Oscillator
VHDL	VHSIC Hardware Descriptive Language
VHSIC	Very High Speed Integrated Circuit

References

[1] J. Oudinot, C. Vaganay, M. Robbe, and P. Radja, "Mixed-Signal ASIC Top-Down and Bottom-Up Design Methodologies using VHDL-AMS," http://www.mentor.com/dsm/tpapers/ms_asic.html, (current 12 Nov. 2001).

[2] G. Gielen, "Top-Down Design of Mixed Mode Systems: Challenges and Solutions," J. Huijsing, R. van de Plassche, and W. Sansen, Ed. Norwell, MA: Kluwer, 1998, ch. II.6.

[3] J. Holmes, F. James, and I. Getreu, "Mixed-Signal Modeling for ICs," Integrated System Design Magazine, June 1997, http://www.isdmag.com/editorial/1997/coverstory9706.html, (current 12 Nov. 2001).

[4] K. Kundert, "A Formal Top-Down Design Process for Mixed-Signal Circuits," http://www.planetanalog.com/story/OEG20001004S0004, (current 12 Nov. 2001)

[5] OVI Working Group, OVI Standard VERILOG AMS: Language Reference Manual- Analog and Mixed-Signal Extensions, 1998.

[6] IEEE Std 1076.1- 1999 IEEE Standard VHDL Analog and Mixed-Signal Extensions, 18 March 1999.

[7] IEEE 1076.1 Working Group, IEEE Standard VHDL 1076.1 Language Reference Manual-Analog and Mixed-Signal Extensions to VHDL 1076, July 1997.

[8] B. Smedt and G. Gielen, "Models for Systematic Design and verification of Frequency Synthesizers," IEEE Transaction on Circuits and Systems-II, Vol. 46, No.10, October 1999, pp. 1301-1308.

[9] B. Razavi, "Design of Monolithic Phase-Locked Loops and Clock Recovery Circuits- A Tutorial," in Monolithic Phase Locked Loops and Clock Recovery Circuits: Theory and Design. New York: IEEE Press, 1996.

[10] Floyd M. Gardner, "Charge Pump Phase-Lock Loop," IEEE Transaction on Communications, vol. com-28, no. 11, pp. 1849-1858, Nov. 1980.

[11] P. R. Gray and R. G. Meyer, "Design and Analysis of Analog Integrated Circuits," third edition, John Wiley & Sons, 1993.

[12] B. Razavi, K. F. Lee and R. H. Yan, "Design of High-Speed, Low-Power

Frequency Dividers and Phase-Locked Loops in Deep Submicron CMOS," IEEE Journal of Solid State Circuits, vol. 30, no. 2, Feb. 1995, pp.101-109.

[13] H. Ahn, D. Allstot, "A Low-Jitter 1.9-V CMOS PLL for Ultra SPARC Microprocessor Applications," IEEE Journal of Solid State Circuits, vol. 35, no.3, March 2000, pp. 450-454.

[14] V. von Kaenel, " A High Speed, Low Power Clock Generator for a Microprocessor Application," IEEE Journal of Solid State Circuits, vol. 33, no.11, Nov. 1998, pp. 1634-1639.

[15] H. Johansson, "A Simple Precharged CMOS Phase Frequency Detector," IEEE Journal of Solid State Circuits, vol. 33, no.2, Feb. 1998, pp. 295-299.

[16] C. Park and B. Kim, " A Low_Noise, 900 MHz VCO in 0.6-μm CMOS," IEEE Journal of Solid State Circuits, vol. 34, no.5, May 1999, pp. 586-591.

Appendix

Behavioral Models

```
library DISCIPLINES, IEEE;
use DISCIPLINES.ELECTROMAGNETIC_SYSTEM.all;
use IEEE.MATH_REAL.all;

entity cco is

      generic ( Rin : REAL := 11.1e3;        -- CCO input resistanec
                V0  : REAL := 0.84  ;        -- input OC voltage
                Vp  : REAL := 0.9   );       -- output peak to peak voltage

       port (terminal in_cco  : ELECTRICAL;      -- input port
             terminal out_cco : ELECTRICAL;      -- output port
             terminal REF : ELECTRICAL           -- ref port

              );

end entity cco;

architecture bhv of cco is

      constant pi: REAL  := math_pi;
      constant ph1: REAL := 0.0;
      quantity v_in  across i_ctrl through in_cco to REF;
      quantity v_out across i_out through out_cco to REF;
      quantity phase : REAL;
      quantity freq  : REAL;
      quantity v_sin : REAL;
      signal   sout  : REAL  := 0.0 ;

      constant A     : REAL := -0.3485;       -- poly factor
      constant B     : REAL :=  45.5585;      -- poly factor
      constant C     : REAL :=  44.5526;      -- poly factor

      constant ctrl_limit_low  : REAL := 0.0;
      constant ctrl_limit_high : REAL := 35.0e-6;

      constant Trise : REAL := 100.0e-12;     -- output pulse rise time
      constant Tfall : REAL := 100.0e-12;     -- output pulse fall time

begin

       v_in == Rin* i_ctrl + V0;
```

```
        if domain = quiescent_domain use
              phase == 0.0;     -- initial condition for phi
        else
              if i_ctrl >= ctrl_limit_high use

                freq ==((A* ctrl_limit_high * ctrl_limit_high*
                        1.0e12) + (B* ctrl_limit_high * 1.0e6) + C )* 1.0e6;

              elsif i_ctrl <= -ctrl_limit_low use

                      freq == 0.0;
               else
                     freq ==(( A * i_ctrl* i_ctrl * 1.0e12 ) +
                      ( B * i_ctrl* 1.0e6 ) +C )* 1.0e6;
              end use;
        end use;

        break on

        i_ctrl'ABOVE(ctrl_limit_high),i_ctrl'ABOVE(ctrl_limit_low);

        phase'DOT == 2.0* pi* freq;

        v_sin == sin( phase);

       p: process ( v_sin'ABOVE(0.0))

        begin

              if v_sin > 0.0 then
                    sout <= Vp;

              else
                    sout <= 0.0;

              end if;

        end process p;

        v_out == sout'ramp(Trise,Tfall);

end architecture bhv;


library DISCIPLINES;
use DISCIPLINES.ELECTROMAGNETIC_SYSTEM.all;


architecture bhv of cplpf is

        constant Kn     : REAL := 303.0e-6   ;
        constant Wn     : REAL  := 220.0e-6   ;
        constant Ln     : REAL  := 180.0e-6   ;
        constant vthn   : REAL  := 0.47  ;
        constant  Kp    : REAL  := 78.0e-6   ;
        constant  Wp    : REAL  := 220.0e-6   ;
```

```
        constant  Lp    : REAL  := 180.0e-6   ;
        constant  vthp    : REAL  := -0.44  ;

        terminal N1, N3, N4 : ELECTRICAL;           -- internal nodes
        quantity vR across iR through N4   to  Vcontrol;
        quantity vC across iC through VDD  to  N4;
        quantity Vout across  Vcontrol to REF;

begin

SWNU:  entity work.SWN
        generic map ( Ron => Ron, Roff => Roff)
        port map ( p => Vcontrol, m => N3, CTRL => UP);

SWPU:  entity work.SWP
        generic map ( Ron => Ron, Roff => Roff)
        port map ( p => Vcontrol, m => N3, CTRL => UPB);

SWND:  entity work.SWN
        generic map ( Ron => Ron, Roff => Roff)
        port map ( p => N1, m => Vcontrol, CTRL => DOWN);

SWPD:  entity work.SWP
        generic map ( Ron => Ron, Roff => Roff)
        port map ( p => N1, m => Vcontrol, CTRL => DOWNB);

---- Iu current source

NFET2: entity work.NFET
        generic map ( Kn => Kn, Wn => Wn, Ln => Ln, vth => vthn)
        port map ( G => VDD, D => N3, S => REF);


NFET1: entity work.NFET
        generic map ( Kn => Kn, Wn => Wn, Ln => Ln, vth => vthn)
        port map ( G => Vcontrol, D => VDD, S => N3);

---- Id current source

PFET2: entity work.PFET
        generic map ( Kp => Kp, Wp => Wp, Lp => Lp, vth => vthp)
        port map ( G => REF, S => VDD, D => N1);


PFET1: entity work.PFET
        generic map ( Kp => Kp, Wp => Wp, Lp => Lp, vth => vthp)
        port map ( G => Vcontrol, S => N1, D => REF);

        vR == R* iR;

        if domain = quiescent_domain use
                vC == 0.0;
        else
                iC == c* vC'DOT;
        end use;
end architecture bhv;
```

```
library DISCIPLINES, IEEE;
use     DISCIPLINES.ELECTROMAGNETIC_SYSTEM.all;
use     IEEE.MATH_REAL.all;

entity ls is

        generic ( vth : REAL := 0.5 );          -- threshold voltage
        port ( terminal inport, outport, REF : ELECTRICAL;
             signal Sout: out bit );            -- bit output

end entity ls;

architecture bhv of ls is

        constant Trise : REAL := 100.0e-12;      -- output pulse rise
time
        constant Tfall : REAL := 100.0e-12;      -- output pulse fall
time
        constant VDD : REAL := 1.8;
        quantity vin across inport to REF;
        quantity v_out across i_out through outport to REF;
        signal  Sout_real :  REAL := 0.0;

begin

        P: process
        begin
             if vin'above(vth) then
                  Sout <= '1';
                  Sout_real <= VDD;

             else
                  Sout <= '0';
                  Sout_real <= 0.0;

             end if ;

             wait on vin'above(vth);

        end process P;

        v_out == Sout_real'ramp(Trise,Tfall);

end architecture bhv;



library DISCIPLINES;
use DISCIPLINES.ELECTROMAGNETIC_SYSTEM.all;


entity NFET is
         generic ( Kn    : REAL  := 303.0e-6   ;
                   Wn    : REAL  := 220.0e-9   ;
                   Ln    : REAL  := 180.0e-9   ;
```

```
            vth   : REAL  := 0.47  );

      port ( terminal G, D, S    : electrical);
end entity NFET;

architecture simple of NFET is

      quantity Vgs across  Ig through G to S;
      quantity Vds across Ids through D to S;

begin

if Vgs <= Vth use
      Ids == 0.0;

elsif ( Vgs > Vth and Vds < Vgs - Vth) use

      Ids == (Kn/2.0)*(Wn/Ln)*(2.0* (Vgs-Vth)*Vds - Vds*Vds);

else
      Ids == (Kn/2.0)*(Wn/Ln)* ((Vgs-Vth)*(Vgs-Vth));
end use;

Ig == 0.0;

end architecture simple;


library DISCIPLINES;
use DISCIPLINES.ELECTROMAGNETIC_SYSTEM.all;

entity PFET is

      generic ( Kp    : REAL  := 78.0e-6   ;
            Wp    : REAL  := 220.0e-9   ;
            Lp    : REAL  := 180.0e-9   ;
            vth   : REAL  := -0.44  );
        port ( terminal G, D, S    : electrical);

end entity PFET;

architecture simple of PFET is

      quantity Vgs across  Ig through G to S;
      quantity Vds across Ids through D to S;

begin

if Vgs >= Vth use

      Ids == 0.0;

elsif ( Vgs < Vth and Vds > Vgs - Vth) use

      Ids == -(Kp/2.0)*(Wp/Lp)*(2.0* (Vgs-Vth)*Vds - Vds*Vds);
```

```
else
      Ids == -(Kp/2.0)*(Wp/Lp)* ((Vgs-Vth)*(Vgs-Vth));

end use;

Ig == 0.0;


end architecture simple;



library PLL;
use PLL.all;
use work.all;

architecture str of phd4 is

      constant td2 : time := delay_NOR2 ;    -- delay of nor2 gates
      constant td4 : time := delay_NOR4 ;    -- delay of nor4 gates
      constant tdinv : time := delay_INV;  -- delay of inverter

      signal RESET, x1, x2, x3, x4, x5, x6, x7, x8, xu, xd : bit;
      signal en : bit := '0';


begin

      G1:   entity work.nor2
            generic map (tdelay => td2)
            port map ( IN1 => xu, IN2 => REF, nor_OUT => x1);

      G2:   entity work.nor2
            generic map (tdelay => td2)
            port map ( IN1 => x1, IN2 => x3, nor_OUT => x2);

      G3:   entity work.nor2
            generic map (tdelay => td2)
            port map ( IN1 => x2, IN2 => RESET, nor_OUT => x3);

      G4:   entity work.nor2
            generic map (tdelay => td2)
            port map ( IN1 => RESET, IN2 => x5, nor_OUT => x4);

      G5:   entity work.nor2
            generic map (tdelay => td2)
            port map ( IN1 => x4, IN2 => x6, nor_OUT => x5);

      G6:   entity work.nor2
            generic map (tdelay => td2)
            port map ( IN1 => FBK, IN2 => xd, nor_OUT => x6);

      G7:   entity work.nor2
            generic map (tdelay => td2)
            port map ( IN1 => x1, IN2 => x2, nor_OUT => x7);
```

```
    G8:   entity work.nor4
          generic map (tdelay => td4)
          port map ( IN1 => x1, IN2 => x2, IN3 => x5, IN4 => x6,
                     nor_OUT => RESET);

    G9:   entity work.nor2
          generic map (tdelay => td2)
          port map ( IN1 => x5, IN2 => x6, nor_OUT => x8);

    INV1: entity work.inv
          generic map ( tdelay => tdinv)
          port map ( D_IN => x7, D_OUT => UPB);


    INV2: entity work.inv
          generic map ( tdelay => tdinv)
          port map ( D_IN => x8, D_OUT => DOWNB);

    en   <= '1' after 1 ns;
    xu   <= en and x7;
    xd   <= en and x8;
    UP   <= x7;
    DOWN <= x8;

end architecture str;



library DISCIPLINES;
use DISCIPLINES.ELECTROMAGNETIC_SYSTEM.all;


entity SWN is

    generic ( Ron   : REAL  := 1.0e-3   ;    -- switch on
resistance
              Roff  : REAL  := 1.0e12   ;    -- switch off resistance
              Tdon  : time  := 15.0 ps  ;   -- switch turn on time
              Tdoff : time  := 20.0 ps  );   -- switch turn off time

    port ( signal  CTRL    : in  bit ;       -- control signal
           terminal p,m    : electrical);


end entity SWN;

architecture stepped of SWN is

    quantity Vsw across Isw through p to m;
    signal mode  : bit :='0';      -- '0' off, '1' on

begin

switching: process( CTRL )

begin
```

```
        if CTRL = '1' then
                mode <= '1' after Tdon;

        else
                mode <= '0' after Tdoff;

        end if;

end process switching;

if mode = '1' use

        Isw == Vsw/Ron;
else
        Isw == Vsw/Roff;
end use;

end architecture stepped;


library DISCIPLINES;
use DISCIPLINES.ELECTROMAGNETIC_SYSTEM.all;


entity SWP is

        generic ( Ron   : REAL  := 1.0e-3   ;    -- switch on
resistance
                  Roff  : REAL  := 1.0e12   ;    -- switch off resistance
                  Tdon  : time  := 15.0 ps   ;   -- switch turn on time
                  Tdoff : time  := 20.0 ps  );   -- switch turn off time

        port ( signal  CTRL    : in  bit ;        -- control signal
               terminal p,m    : electrical);

end entity SWP;

architecture stepped of SWP is

        quantity Vsw across Isw through p to m;
        signal mode  : bit :='0';      -- '0' off, '1' on

begin

switching: process( CTRL )

begin
        if CTRL = '0' then
                mode <= '1' after Tdon;

        else
                mode <= '0' after Tdoff;

        end if;

end process switching;
```

```
if mode = '1' use

      Isw == Vsw/Ron;
else
      Isw == Vsw/Roff;
end use;

end architecture stepped;


library DISCIPLINES, IEEE;
use DISCIPLINES.ELECTROMAGNETIC_SYSTEM.all;
use IEEE.MATH_REAL.all;

entity viconv is

      generic ( G    : REAL := 0.0;        -- viconv output conductance
                vth  : REAL := 0.6;        -- threshold voltage
                Gm   : REAL := 32.4e-6   ); -- transconductance value

     port (terminal in_vi  : ELECTRICAL;      -- input port
           terminal out_vi : ELECTRICAL;      -- output port
           terminal VDD    : ELECTRICAL          -- ref port

            );

end entity viconv;

architecture bhv of viconv is

       quantity vc   across i_in through VDD to in_vi;   -- to REF;
      quantity v_out across Ic through VDD to out_vi;   -- to REF;

begin

      i_in == 0.0;

      if vc < vth use

            Ic == 0.0;
      else

            Ic == Gm*(vc-vth) + G*v_out;

      end use;

      break on vc'ABOVE (vth);

end architecture bhv;


library DISCIPLINES, IEEE;
use DISCIPLINES.ELECTROMAGNETIC_SYSTEM.all;
use IEEE.MATH_REAL.all;
```

```
entity dampctrl is

      generic ( G    : REAL := 0.0;        -- viconv output
conductance
                vth  : REAL := 0.6;        -- threshold voltage
                Gm   : REAL := 32.4e-6   ); -- transconductance value

      port (terminal VDD  : ELECTRICAL;        -- VDD port
            terminal CTRL : ELECTRICAL;        -- control port
            terminal OUT1 : ELECTRICAL;        -- output port
            terminal OUT2 : ELECTRICAL;        -- output port
            signal   IN1  : in  bit ;
            signal   IN2  : in  bit );

end entity dampctrl;

architecture bhv of dampctrl is


      terminal N : ELECTRICAL;
      quantity Vd across Id through VDD to N ;
      quantity vc across i_in through VDD to CTRL;

begin

SWP1:   entity work.SWP
              port map ( p => N, m => OUT1, CTRL => IN1);

SWP2:   entity work.SWP
              port map ( p => N, m => OUT2, CTRL => IN2);

      i_in == 0.0;

      if vc < vth use

            Id == 0.0;
      else

            Id == Gm*(vc-vth) + G*vd;

      end use;

      break on vc'ABOVE (vth);

end architecture bhv;


architecture bhv of fdiv is

signal tmp : bit := '0' ;

begin

p: process (pulse_in)
```

```
variable counter : integer := 0;
variable N1      : integer := N ;
variable N2      : integer := N + 1;

begin

        if pulse_in = '1' then

            counter := counter +1;

            if tmp = '1' then
                tmp <= '0';
            end if;

            if counter = N1 and sc = '0' then

                tmp <= '1';
                counter := 0;

            elsif counter = N2 and sc = '1' then
                tmp <= '1';
                counter := 0;

            end if;
      end if;

end process p;

pulse_out <= tmp;

end architecture bhv;


library DISCIPLINES, IEEE, PLL;
use     DISCIPLINES.ELECTROMAGNETIC_SYSTEM.all;
use     IEEE.MATH_REAL.all;
use     PLL.all;
use     work.all;

entity PLL is

port (  signal   PLL_in  : in  bit;       -- PLL input
      signal PLL_out : out bit;          -- PLL output
      terminal PLL_out_analog, REF : electrical);

end entity PLL;

architecture bhv of PLL is

      constant RD : REAL := 1.0e3;      -- ground resistor

      signal UP, UPB, DOWN, DOWNB : bit;
      terminal VDDA, Vcontrol : electrical;
      terminal CCO_in, D, CCO_out : electrical;
      signal   pulse_div :bit ;
      signal   dig_out :bit;
```

```
        signal   sc : bit := '0';
        quantity VD across ID through D;
        quantity Vsupply across Isupply through VDDA ;

begin

        PHD4: entity work.PHD4
              generic map ( delay_INV => 1.0 ps, delay_NOR2 => 5.0 ps,
                        delay_NOR4 => 5.0 ps)
              port map ( REF => PLL_in, FBK => pulse_div, UP => UP,
                       UPB => UPB, DOWN => DOWN, DOWNB => DOWNB);

        LPF: entity work.CPLPF

             generic map ( Iup => 28.0e-6, Idown => 25.0e-6, R => 0.0,
                         C => 1.5e-12, Ron => 1.0e1, Roff => 1.0e12)
              port map ( UP => UP, DOWN => DOWN, UPB => UPB, DOWNB =>
             DOWNB, VDD => VDDA, REF => electrical_ground,
                       Vcontrol => Vcontrol);

        VIC: entity work.VICONV
              generic map ( G => 0.0, vth => 0.6, Gm => 32.4e-6)
              port map ( in_vi => Vcontrol, VDD => VDDA, out_vi => CCO_in);

       CCO: entity work.CCO
              generic map ( Rin => 11.1e3, V0 => 0.84, Vp => 0.9)
              port map ( in_cco => CCO_in, out_cco => CCO_OUT,
                       REF => electrical_ground);

        LS:  entity work.LS
              generic map ( vth => 0.5)
              port map ( inport => CCO_OUT, REF => electrical_ground,
                       Sout => dig_out, outport => PLL_out_analog);


        DIV: entity work.fdiv
              generic map ( N => 4)
              port map ( pulse_in => dig_out, pulse_out => pulse_div,
                        SC => sc );

        VD == RD * ID;
        Vsupply == 1.8;
        PLL_out <= dig_out;

end architecture bhv;
```