Top-Down Design of a Phase Locked Loop For a Video Driver System

by Iasson F. Vassiliou

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee:

Professor Alberto Sangiovanni-Vincentelli

Research Advisor

Date

* * * * * *

Professor Paul Gray

Second Reader

Date

Abstract

To accelerate the design of analog circuits, a top-down, constraint-driven design methodology has been proposed [1]. In this work, the methodology is applied in the design of a programmable, phase-locked loop based frequency synthesizer for a video driver system. The complete design flow is presented, from system level specifications to layout. At the higher hevels of the hierarchy, the system is described using behavioral models. Behavioral simulation and optimization techniques are used to map higher level performance specifications to constraints for the lower levels of the design hierarchy. A modified version of the supporting hyperplane optimization algorithm is implemented to cope with the limited accuracy of the available behavioral and circuit simulation tools. The final layout is extracted and the system performance is verified using hierarchical simulation.

Contents

List of Figures

List of Tables

Acknowledgements

There are many people I would like to thank for their support in my research. First I would like to thank my principal research advisor, Professor Alberto Sangiovanni-Vincentelli, for his support and guidance in my research. He gave me valuable insight and motivation for this work. I would also like to thank Professor Paul Gray, my research co-advisor, who gave me precious guidance in circuit and system design issues.

There are many collegues that I would also like to thank for their valuable assistance: Henry Chang for the project idea and his cooperation; Ed Liu for the project idea and useful research discussions; Alper Demir for the behavioral simulation and modelling and the useful research discussions; Edoardo Charbon for patiently spending so much time helping with the layout tools, for useful research discussions and also for the VCO layout generator; Robert Neff for his help with the optimization algorithms and circuit design advice; Paolo Miliozzi for helping in the constraint-driven VCO layout; Sherman Yee, also for his help with the optimization algorithms; Eric Felt for his help with OPTZ; Todd Weingandt for interesting discussions on PLLs; Keith Onodera for useful circuit design advice. I would also like to thank all the staff for their assistance: Flora Oviedo, Kia Cooper, Tim Castro and Brad Krebs.

This research was supported by SRC (DC-324-010). This support is greatly acknowledged.

Chapter 1

Introduction

1.1 Goals

This paper presents the application of a top-down, constraint-driven design methodology [1] in the design of a Phase Locked Loop (PLL) for a video driver system. The video system was chosen as an example to illustrate the value of the top-down methodology for large analog systems. In this work, the design methodology is used for the design a PLL, while the necessary optimization tools and behavioral models are developed. The final result is the layout of the overall display driver system, which is extracted and its performance is verified. We chose to implement a PLL since it is a challenging analog design problem with many applications including frequency synthesis, data acquisition, carrier recovery in RF receivers and many more. The PLL was embedded in a larger system to show the usefulness of the methodology for the design of large systems.

The importance of the proposed methodology in the design of analog circuits has been discussed extensively before. The goal of the methodology is to ease and accelerate the design of analog circuits, which is often the most time consuming part in the design of a mixed signal system. Following the methodology, the design in done in a hierarchical way, starting from the behavioral description of the higher level building blocks. This allows early detection of design faults and exploration of the design space, since fast behavioral simulators can be used.

Each building block is characterized by a set of parameters. In the top-down design methodology, the system level performance specifications are mapped onto constraints for the behavioral parameters of the high level blocks. The mapping is being done using constrained optimization, where the objective is to maximize a measure of the easiness of the design (flexibility) [1]. This process is repeated until the layout level is reached or a module meeting these constraints is found in the library.

The goal of the methodology is not the complete automation the design of analog circuits. Such a task is believed to be rather unrealistic. The goal is to help the analog designer to accelerate the design cycle of complex systems, exploiting hierarchy and early verification together with providing him with the necessary tools.

Following this methodology analog circuits such as Digital-to-Analog Converters [2] and Sigma-Delta Analog-to-Digital Converters [3] have already been designed and manufactured. With this system, the methodology is applied to yet larger systems.

In Chapter 2, an overview of the most important PLL design concepts and building blocks is given. Chapter 3 describes the tools that were used or developed to support the methodology for this system. In Chapter 4, the design synthesis flow for the PLL is presented in detail. Conclusions and future work are discussed in Chapter 5.

Chapter 2

Analog PLL Design Background

2.1 PLL Design Concepts

Phase Locked Loops (PLLs), are important analog building blocks for a lot of applications. They can be found in receivers for carrier recovery and frequency demodulation, clock recovery in digital systems, frequency synthesizers and various other systems. A PLL basically consists of a Phase Detector (PD), a Loop Filter and a Voltage Controlled Oscillator (VCO). Its function is to adjust the phase of the VCO produced signal to the phase of the incoming signal. A simplified PLL block diagram is given in figure 2.1.

The Phase Detector compares the phase of the incoming signal F_{in} with the VCO

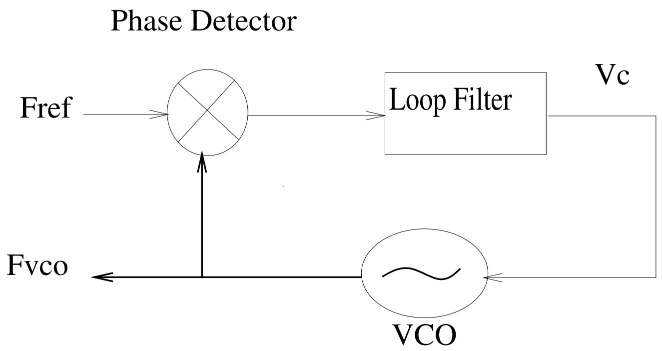

Figure 2.1: PLL simplified diagram

signal F_{vco}. The output of the PD depends on the phase difference of the two signals. The difference voltage is passed through a filter and then applied to the VCO control voltage. The feedback loop is trying to adjust the VCO voltage so that the two compared signals have the same phase. A more quantitative analysis follows.

2.1.1 PLL Linearized Analysis

The PLL is a feedback system that exhibits a strong non-linear behavior, especially when the phase difference of the two compared signals is large and the system is trying to null it. Analyses of the non-linear problem have been done [4], but even a linearized solution can give a lot of information. A classical linearized analysis can be found also in [4]. Assuming that:

- the PD output for small phase differences is given by

$$v_d = K_d \cdot (\theta_i - \theta_o) \tag{2.1}$$

  where θ_i is the input phase, θ_o is the VCO phase and K_d is a constant

- the transfer function of the loop filter is $F(s)$
- The output frequency of the VCO is given by:

$$\omega_{vco}(s) = \omega_o + K_o \cdot V_c(s) \tag{2.2}$$

it can easily be proved that the PLL transfer function is given by:

$$\frac{\theta_o}{\theta_i} = H(s) = \frac{K_o \cdot K_d \cdot F(s)}{s + K_o \cdot K_d \cdot F(s)} \tag{2.3}$$

The quantity

$$BW = K_o \cdot K_d \tag{2.4}$$

is also referred to as *"PLL bandwidth"*. The precise form of the transfer function depends on the type of the loop filter used. In this analysis it was also assumed that the type of phase detector used was an ideal multiplier.

The linearized PLL analysis can give some insight about the operation of the system but the behavior of the loop is quite non-linear when the system is trying to lock

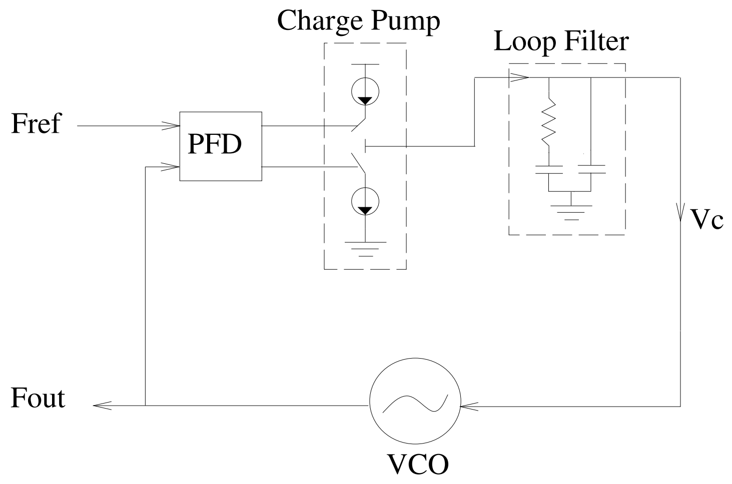

Figure 2.2: Charge Pump PLL

to the applied input. Acquisition and tracking of the applied input are non-linear problems where approximate analysis can be applied [4]. In the transient behavior, the components' non-linearities can be quite significant to the performance of the system. For the evaluation of the effects of all the important non-idealities the best method is to resort to simulation.

2.1.2 Charge Pump PLLs

2.1.2.1 Linearized Analysis

A special type of Phase-Locked loops are Charge-Pump PLLs, which have as a main advantage extended frequency range of operation and low cost. Charge-Pump PLLs use a sequential digital phase/frequency detector (PFD) that converts differences in the phase of the pulses compared into digital waveforms. A charge pump is used to convert the digital waveform to a control voltage for the VCO. A typical implementation of a charge pump PLL is shown in figure 2.2.

The phase/frequency detector is a sequential circuit that generates a digital pulse of duration equal to the phase difference between the two compared pulses. If the upper pulse comes earlier an "up" pulse is generated; if the lower pulse comes earlier a "down" pulse is generated. When the edges of the two pulses are perfectly synchronized no pulse is generated. The function of the PFD can be shown in figure 2.3.

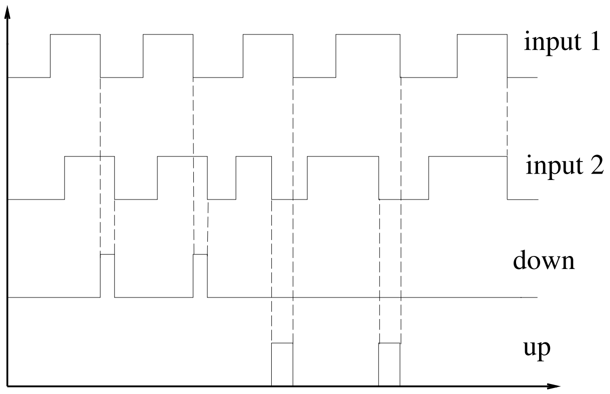

Figure 2.3: PFD I/O relationship

The pulses generated by the PFD turn on the upper or lower current source of the charge pump, which provides the loop filter with charge and changes the VCO control voltage. When the "up" pulse is on, a current I_p is delivered to the loop filter, while when the "down" pulse is on, the current delivered is $-I_p$. In figure 2.2 a second order RC filter is used.

Based on the assumption that the phase differences between the pulses are small and the bandwidth of the PLL is much smaller than the input frequency, a continuous time approximation can be done to quantify the behavior of a charge pump PLL [5]. Assuming that:

- the charge pump current is $\pm I_p$
- the filter impedance is $Z_F(s)$
- the VCO gain is K_o
- θ_i ,θ_o are the input and output phases respectively
- R, C_1, C_2 are the loop filter components

the loop transfer function can be expressed as:

$$\frac{\theta_o}{\theta_i} = H(s) = \frac{K_o \cdot I_p \cdot Z_F(s)}{2 \cdot \pi \cdot s + K_o \cdot I_p \cdot Z_F(s)} \tag{2.5}$$

Assuming a second order loop filter as in figure 2.2 the transfer function becomes:

$$\frac{\theta_o}{\theta_i} = H(s) = \frac{G(s)}{1 + G(s)} \tag{2.6}$$

$$G(s) = \frac{K_o \cdot I_p}{2 \cdot \pi \cdot (C_1 + C_2) \cdot s^2} \cdot \frac{1 + s \cdot R \cdot C_1}{1 + s \cdot R \cdot \frac{C_1 \cdot C_2}{C_1 + C_2}} \tag{2.7}$$

where G(s) is the open loop transfer function. From equation 2.7 the *phase margin* of the PLL can be calculated, which can give a measure of the stability of the system. A large phase margin is desirable to ensure stability under variations of the parameters of the building blocks.

2.1.2.2 Stability considerations

In the previous section a time-averaged analysis was presented. However, the true nature of the system is sampled, so a more detailed analysis is needed in order to reveal potential instability problems due to the sampling operation. A z-domain analysis was performed in [5]. In the case of a second order loop filter as in figure 2.2, the criterion for stability is:

$$K \cdot \tau_2 < \frac{4 \cdot (1 + a)}{\frac{2 \cdot \pi \cdot (b-1)}{b \cdot \omega_i \cdot \tau_2} \cdot [\frac{2 \cdot \pi \cdot (1+a)}{\omega_i \cdot \tau_2} + \frac{2 \cdot (1-a) \cdot (b-1)}{b}]} \tag{2.8}$$

where:

- $K = \frac{K_o \cdot I_p \cdot R}{2 \cdot \pi}$
- $\tau_2 = R \cdot C_1$
- $b = 1 + C_1 / C_2$
- $a = e^{[-\frac{2 \cdot \pi \cdot b}{\omega_i \cdot \tau_2}]}$

2.1.2.3 Non-linear effects

Since PLL operation is non-linear, especially in non-lock, non-linearities should be taken into account in the design of the system.

First, the VCO should be able to accommodate all the frequencies of operation of the PLL plus the frequency excursions due to "peaking" during transient. The peaking during the transient is calculated in [5]:

$$|\Delta\omega_o| = 2 \cdot \pi \cdot K \cdot \left(\frac{b-1}{b}\right) \cdot \left[\frac{b-1}{b}\left(1 - e^{-\frac{b\|\theta_e\|}{\omega_i \cdot \tau_2}}\right) + \frac{|\theta_e|}{\omega_i \cdot \tau_2}\right] \tag{2.9}$$

For the derivation of the equations in section 2.1.2.1 a linear model for the VCO has been assumed. However, the VCO might be linear only for a small range and then overload so this effect might have to be taken into account. Another source of error can be the *dead zone* of the PFD, which is the minimum phase difference to which the circuit can respond. Other non-linearities such as resistor and capacitor non-linearities, charge injection from the switches of the charge pump, digital circuit delays have been found to have small effect to the performance of the PLL due to the feedback nature of the system.

2.1.3 Timing Jitter

Very critical to the performance of a PLL used as a clock generator is the *timing jitter* . In a display driver system this can cause patterns on the screen. This is defined as the standard deviation of the period of the output of the PLL:

$$\overline{\Delta\tau^2} = E\left[(T - E[T])^2\right] \tag{2.10}$$

where T is the output period. The *phase jitter* can be accordingly defined as:

$$E\left[\Theta_{TOT}{}^2\right] = \left(\frac{2\pi}{T}\right)^2 \cdot \overline{\Delta\tau^2} \tag{2.11}$$

In the case where the reference clock is a crystal oscillator, which is typical in clock synthesizer PLLs, the incoming clock is assumed to be “clean” so, the main source of timing jitter is the variation $\overline{\Delta\tau_{VCO}{}^2}$ due to the VCO. The fundamental sources of VCO jitter can be:

- thermal noise
- coupling of digital noise from the power supply
- coupling of digital noise from the substrate

The PLL feedback loop acts in order to correct the variations of the VCO output period that could accumulate to infinity. Depending on how fast the feedback loop acts, the accumulation factor for the jitter varies. Thus, a wide bandwidth PLL would have a smaller accumulation factor than a narrow bandwidth PLL. It is very common that ring oscillators are used for VCOs. Under the assumption that $T \ll RC_1$ and $C_1 \gg C_2$ an analysis of a charge-pump PLL with a second order passive RC loop filter is done in [6]. Given that each of the N stages of the VCO contributes $\overline{\Delta\tau_N{}^2}$, an expression is calculated for the PLL phase noise:

$$\sqrt{E\left[\Theta_{TOT}{}^2\right]} = \alpha \frac{2\pi\sqrt{\overline{\Delta\tau_N{}^2}}}{T} \tag{2.12}$$

where α is the PLL accumulation factor:

$$\alpha = \sqrt{\frac{1}{2I_p K_o RT}} \tag{2.13}$$

The jitter accumulation factor is proportional to the PLL bandwidth $\omega_L \simeq K_o I_p R$, for $T \ll RC_1$ and $C_1 \gg C_2$.

2.2 PLL Building Blocks

2.2.1 Voltage Controlled Oscillator

Voltage Controlled Oscillators are critical components of PLLs. There are various types of VCOs that can be used in PLLs:

- Voltage Controlled Crystal Oscillators (VCXO): Their main advantage is the phase accuracy but they cannot be integrated and the cost is high.
- LC-tuned oscillators: The main disadvantage is the absence of inductors in most IC technologies so integration is impossible which increases the cost. Recently integrated inductors have been manufactured so integrated LC oscillators could be realized.
- Ring oscillator VCOs: They are the most suitable for integration in current digital IC technologies, they have a wide control range and they generate different phases of the output waveform which is useful in various applications. The main disadvantage is their poor jitter performance.

There are various types of ring oscillator VCOs according to the type of delay cell that is being used. The simplest type uses a single-ended inverter cell with some means of controlling the delay of the cell. One way of doing this is to use current-starved inverters as in figure 2.4 where the charging/discharging current is controlled as a means to control the delay of the cell. Such single ended inverter based VCOs suffer from poor power supply

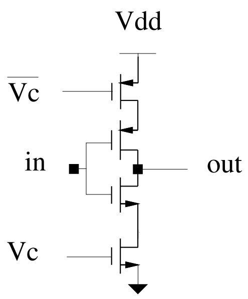

Figure 2.4: Current-starved ring oscillator VCO cell

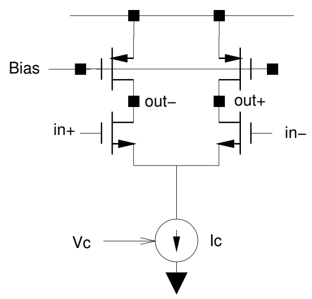

Figure 2.5: Differential delay cell

noise rejection and can be unsuitable in a mixed signal environment where digital switching causes high supply noise.

In order to improve the power supply rejection, differential delay cells are often used, as in figure 2.5.

Typically the loads used in the delay cells are MOS transistors in the triode region for faster switching. If N stages are used, then the oscillation frequency is $F_{out} = \frac{1}{2 \cdot N \cdot T_d}$, where T_d is the delay of one stage. Furthermore, $2 \cdot N$ phases of the output are available.

Differential delay cells do not completely eliminate the problem of power supply rejection, since the transistors of the differential pair are in different operating regions during the transition, but they greatly improve the noise performance. The fundamental limit in the noise performance of the differential cells is set by the device generated noise. For a VCO used in a PLL we are mainly interested in the thermal noise limit, since the 1/f noise is mainly low frequency and is rejected by the PLL loop acting as a high pass filter for the VCO generated jitter [6].

In [7] the average jitter in a ring oscillator VCO using differential stages is calculated to be:

$$\frac{\Delta\tau_{rms}}{t_d} = \sqrt{\frac{kT}{C_L}} \cdot \frac{1}{V_{GS} - V_T} \cdot \xi \tag{2.14}$$

where C_L is the capacitive load of each stage of the ring oscillator, t_d is the delay of one stage and ξ is a design parameter of the cell given by:

$$\xi = \sqrt{1 + \frac{2}{3}\alpha_v(1 - e^{-t/\tau}) + \frac{2\sqrt{2}}{3}\alpha_v e^{-t/\tau}} \tag{2.15}$$

where α_v is the small signal gain of the cell, $\tau \simeq t_d$. From the above equations it can be inferred that there is a trade-off between the power consumption of the VCO and the timing jitter.

2.2.2 Phase Detectors

The function of a Phase-Frequency Detector is described in section 2.1.2.1. A typical implementation of a PFD is shown in figure 2.6. A PFD is typically used in clock generator PLLs since it can provide a wide frequency range of operation.

Multipliers can also be used as phase detectors. They are typically used in receivers and clock recovery circuits in order to retrieve the synchronizing clock from the received data.

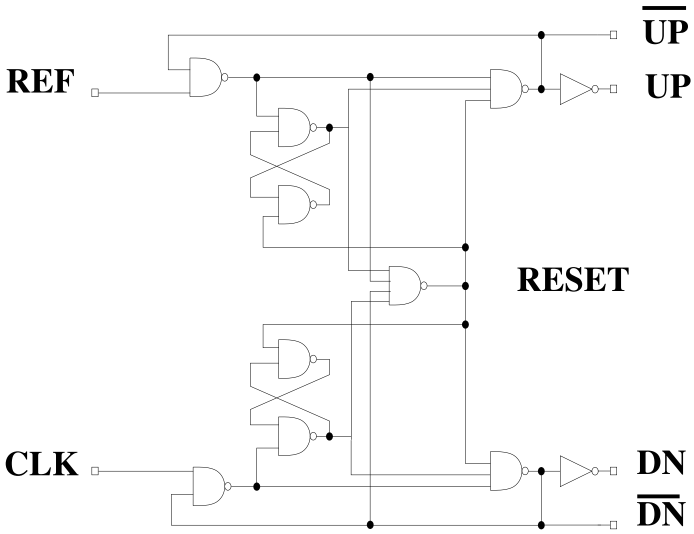

Figure 2.6: Phase-Frequency Detector

Chapter 3

Simulation and Optimization Techniques

3.1 The Use of Simulation and Optimization in the Design Methodology

In the top-down, constraint-driven design methodology proposed in [1], behavioral simulation and optimization play a very important role. Key idea to the methodology is the hierarchical propagation of the top-level specifications to the lower levels of the hierarchy. For example, in figure 3.1 the top level DNL, INL specifications of a 10 bit current source interpolative DAC [2] are being mapped onto constraints for the behavioral parameters of the building blocks (standard deviation of mismatches and mirror output resistances). Following the methodology, since there is an large number of parameter combinations that can meet the specifications, the choice is being done by optimizing for the *flexibility* [1] of the design, which is a measure of the easiness to achieve the design goals. The flexibility functions are typically defined heuristically by the designer. At the highest level of hierarchy, the details of the implementation are not determined yet. Thus, a behavioral simulator is needed to evaluate the system performance from the behavioral parameters of the building blocks. Behavioral simulation allows a fast and efficient exploration of the design space. The optimization algorithm evaluates the performance of each parameter combination by employing the behavioral simulation tool. At the lowest level of hierarchy a circuit simu-

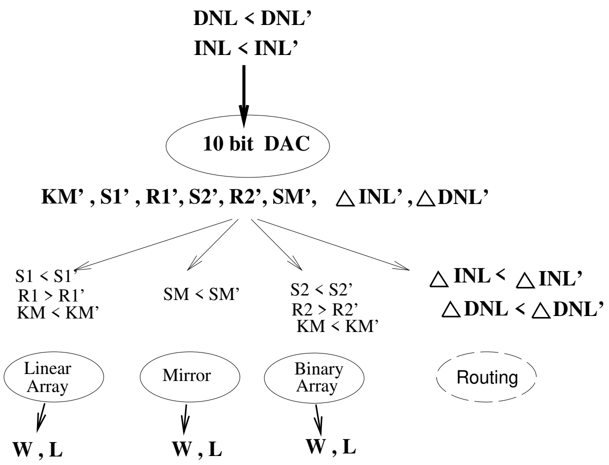

Figure 3.1: 10 bit DAC decomposition

lator can be used in order to map performance constrains to transistor sizes. In figure 3.1 the intermediate level parameters are being mapped onto transistor sizes using a circuit simulator as the performance evaluation tool.

Behavioral simulation is also necessary for fast and efficient hierarchical, bottom-up verification. In a large mixed signal system with thousands of transistors, a flat circuit simulation can be extremely time consuming, if not impossible. With the hierarchical bottom-up extraction and verification, the parameters of the lowest level blocks of the hierarchy are being extracted using a circuit simulator. Then the system is simulated using the extracted high level parameters with a behavioral simulator.

For both hierarchical mapping and bottom-up verification, behavioral modeling and simulation need to be general enough so that they are independent of the low level implementation but at the same time have to capture all the important second order effects that are likely to degrade the performance of a mixed analog-digital system. In the case of the DAC design of figure 3.1, such effects are the transistor mismatches and output resistances that can affect the INL and DNL performance.

In this chapter we describe the optimization and behavioral simulation tools and models that were used or developed for the PLL design.

3.2 Behavioral Simulation Tools

Phase Locked Loops are systems that exhibit very "stiff" behavior. Intermediate nodes may have very fast time constants but the PLL bandwidth may be only a few kilohertz, which results in a very hard simulation problem with slow convergence and extremely high simulation time. In addition to that, timing jitter, which is the most important second order phenomenon affecting the performance of a PLL, cannot be precisely predicted using standard circuit simulation techniques. The most important reason for this is that the numerical noise created by most simulation algorithms can be comparable to the timing jitter. These problems raise the need for an efficient behavioral simulation tool for mixed signal systems.

Such a simulator has been developed in [8]. The simulator uses behavioral representations described in [9]. Those representations comprise a set of non-linear differential equations solved by an iterative algorithm using a variable step, implicit backward differentiation method. Key advantage of this algorithm is that it can have a user defined accuracy, thus eliminating the problem of numerical noise.

In a square wave VCO used in a PLL system, only the exact transition times of the digital waveforms are important to the system performance. An accurate calculation of those waveforms would be a waste of computational power that would slow down the simulation. In the PLL behavioral simulator, the waveform at the output of the VCO is represented by its zero crossings which can be calculated from the derivative of the instantaneous VCO phase given by:

$$\frac{\partial \phi}{\partial t} = 2\pi f(v(t)) \tag{3.1}$$

where $v(t)$ is the VCO control voltage and $f(v)$ is the VCO frequency/voltage characteristic. A phase transition is detected when $\phi(t) = n\pi$. The exact point of the transition is determined by a bisection search. In the following paragraphs the behavioral models for each PLL building block will be described. Finally, the differential equations describing the PLL will be given.

state	*a crossing*	*b crossing*	*out*
0	(1 , t_d)	(2 , t_d)	0
1	(1 , t_d)	(0 , t_d)	UP
2	(0 , t_d)	(2 , t_d)	DN

Table 3.1: PFD State Transition Table

3.2.1 PFD Model

The PFD used was the one whose operation is described in figure 2.3. As indicated in [9], such a PFD can be described by a state transition table as in 3.1. In table 3.1, *a crossings, b crossings* are the changes in the positive edge of inputs a and b and t_d is the delay of the response of the circuit.

A significant effect that can deteriorate the performance of a PLL is the PFD *dead zone*, which is the minimum phase difference between the two pulses compared that can produce a PFD pulse. This is due to the digital delays of the logic gates of the PFD and can be eliminated by turning on both the *up* and *down* pulses for a fixed duration even when the pulses are aligned, and changing the duration of one of the pulses according to the relative delay between the edges of the pulses. This effect is modeled in the behavioral simulator as the minimum time difference between the pulses that would generate an UP or DOWN pulse, otherwise no pulse is generated. When the value of t_{dead_zone} was high, there was a significant effect in PLL behavioral simulations. A typical PFD implementation is the one in figure 2.6.

3.2.2 Charge-Pump, Loop Filter Model

Generally, a charge pump can be implemented by two current sources connected to the same node by two switches as in figure 2.2. The most important non-idealities affecting the charge-pump performance can be:

- finite output resistance of the current sources, characterized by R_{out} in the behavioral model
- mismatch between the upper and lower current sources, $\frac{\Delta I_p}{I_p}$

parameter	*description*
I_p	nominal charge-pump current
$\frac{\Delta I_p}{I_p}$	up and down current mismatch
R_{lin}	linear region resistance
R_{sat}	saturation region resistance
V_{satup}	upper source saturation voltage
V_{satlo}	lower source saturation voltage

Table 3.2: Charge-Pump behavioral parameters

- charge injection from the switches to the output node when the switch turns off, Q_{inj}
- finite output range due to the saturation of the transistors of the current sources, modeled by the output resistance at the linear region, R_{lin}

At any time, the charge-pump current is given by:

$$I_{eff} = ST \cdot I_p(1 + \frac{\Delta I_p}{I_p} \cdot ST) - ST\frac{\Delta V}{R_{out}} \tag{3.2}$$

where ST has values $+1$ for UP, -1 for $DOWN$ and 0 when no pulse is generated and I_p is the nominal value of the charge pump current. The value of the output resistance is determined by a piecewise linear model that has two values, R_{lin} for the linear region and R_{sat} for the saturation region. $\Delta V = V_{out} - V_{sat}$, where V_{sat} is the saturation voltage and has different values for the lower and upper current sources. All those parameters can be extracted from a circuit with a DC simulation. Table 3.2 summarizes all the parameters describing the charge-pump.

In the case of a passive, lead-lag RC filter which will be used in this design example, the behavioral model is just the values of its R,C components.

3.2.3 Divider Model

Digital dividers for the purposes of behavioral simulation, can be modeled by their divide ratio n_d and digital delay t_d. Digital delay does not really affect the simulation and is only included for completeness of the model. Timing jitter couild also be included in the bevavioral model but is negligible compared to the jitter caused by the VCO.

parameter	*description*
F_0	VCO central frequency
K_0	frequency-to-voltage gain
V_{max}	maximum applied contr. volt.
V_{min}	minimum applied contr. volt.
$\Delta\tau_{VCO}$	VCO timing jitter

Table 3.3: VCO behavioral parameters

3.2.4 VCO model

In general, a ring oscillator VCO can be approximated by a first order linear model:

$$F_{out} = F_0 + K_0 \Delta V_C \tag{3.3}$$

where F_0 is the VCO center frequency, K_0 is the frequency-to-voltage gain and ΔV_C the applied control voltage. Even though higher order models can be used to match the exact characteristic of an extracted actual VCO circuit, simulations have shown that such an accuracy is not necessary for a PLL behavioral simulation. This was expected due to the feedback nature of the system.

However, the finite output range of the VCO needs to be modeled. For control voltages outside this range, we either have decaying oscillation, or no oscillation at all. This is modeled by returning a practically zero output frequency outside this range. However, the continuity of the function around the extreme points must be ensured for the stability of the numerical algorithms used, so an exponential decay model is assumed outside the region of validity of the linear model.

Timing jitter as defined in equation 2.10, is modeled by the value of its standard deviation and can be extracted from a circuit using noise simulation techniques described in [10] or by using equation 2.14 for differential cell, triode load ring oscillator VCOs.

In table 3.3 all the parameters describing the VCO are summarized.

3.2.5 PLL Behavioral Models

The behavioral models used for the PLL components are based on the ones described in [9] . For a more realistic representation of second order effects more details were added, such as the charge pump output resistance and the saturation of the VCO, as mentioned in previous paragraphs. The behavioral representation of the PLL is a set of first order differential state equations given by the user and the jacobian of the state equations with respect to the state variables. The behavioral simulator solves this set of differential equations as described in section 3.2. In the case of the PLL of figure 2.2, the equations based on the models described in the previous paragraphs are the following:

$$\frac{\partial \theta_i}{\partial t} = 2\pi f_i(t) \tag{3.4}$$

$$\frac{\partial V_c}{\partial t} = \frac{I_{p_{eff}}}{C_2} - \frac{1}{RC_2}V_c + \frac{1}{RC_2}V_x \tag{3.5}$$

$$\frac{\partial V_x}{\partial t} = \frac{1}{RC}V_c - \frac{1}{RC}V_x \tag{3.6}$$

$$\frac{\partial \theta_j}{\partial t} = 2\pi F(V_c(t))n_d \qquad \forall j, j = 1 \dots n_d \tag{3.7}$$

where:

$$I_{p_{eff}} = ST \cdot I_p(1 + \frac{\Delta I_p}{I_p} \cdot ST) - ST\frac{\Delta V}{R_{out}} \tag{3.8}$$

$$F(V_c(t)) = F_0 + K_0 V_c \tag{3.9}$$

The state variables θ_i, V_c, V_x, θ_j represent the phase of the input clock, the VCO control voltage, the voltage on capacitor C and the phases of the n_d stages of the VCO delay stages respectively. The jacobian of the above equations with respect to the state variables is also given analytically to the simulator.

3.2.5.1 PLL timing jitter

Critical for the performance of a PLL in many applications is its timing jitter, as defined in equation 2.10. The user defined accuracy of the simulator allows an arbitrarily accurate jitter simulation.

As discussed in 2.1.3, the most fundamental timing jitter noise source is the thermal noise of the electronic circuits, causing uncertainty in the digital waveform transitions. Assuming that the other timing jitter sources have been dealt with, timing jitter of circuit components can be attributed exclusively to thermal noise and be modeled as a white gaussian random process. In addition to that, we can assume that when the system is in lock, noise is small enough so that it does not drive the system out of lock. The overall PLL jitter is then predicted by adding random noise at the time of each VCO transition and subsequent statistical analysis of the resulting waveform [8].

3.3 Optimization Method

In section 3.1 the use of optimization in the top-down design methodology was explained. Since in most circuit design problems we are dealing with complex nonlinear optimizations, nonlinear methods have to be used. There are several methods in the literature dealing with nonlinear optimization, typically gradient methods that use second derivative information of the constraint and objective functions. Such methods are used in nonlinear optimization packets like MINOS [11]. The basic idea is to find a local minimum by finding the point where the second derivatives of the functions are zero.

Such an approach presents some major disadvantages:

- the values of the first and second order derivatives must be very accurate for the method to converge. When simulators are used to calculate the constraint functions, finite accuracy can cause large errors in the first and second order derivatives.
- they can find only local extremes of the objective function
- often the gradients of the constraint function are not defined outside the feasible region, as in the case of the PLL where we cannot define timing jitter when the system is unstable.

One solution can be using simulated annealing as in [12]. The drawback is that such a solution can be computationally too expensive and does not use the inherent gradient behavior that circuit performance parameters have, thus discarding useful information about the circuit contained in the gradients.

A quite efficient method for such problems has been proved to be the supporting hyperplane method, used in [13], [2]. A modified version of this method was employed and is described in the following section.

3.3.1 The Supporting Hyperplane Method

The Supporting Hyperplane Method can solve any optimization problem of the form:

$$\begin{aligned} &minimize\ \mathbf{c}^T\mathbf{x} \\ &subject\ to\ \mathbf{g}(\mathbf{x}) \leq \mathbf{0} \end{aligned} \tag{3.10}$$

where:

- $\mathbf{x} \in E^n$,
- $\mathbf{g}(\mathbf{x}) \in E^p$,
- $g_i(\mathbf{x})$ are continiously differentiable functions
- the space defined by the constraints $\mathbf{g}(\mathbf{x})$ is convex.

The algorithm works as follows: After an initial feasible point is given, un unconstrained optimization is performed. Then, the nonlinear constraints are checked and if the solution point $\mathbf{P_0}$ is feasible then the algorithm stops and the solution is a global minimum. If a constraint g_i is violated, then the point $\mathbf{u_{k+1}}$ is found on the line joining the initial feasible point $\mathbf{P_0}$ and the last solution $\mathbf{P_{k+1}}$, that lies on the boundary of the feasible region S. Then a linear constraint is added such as:

$$\nabla g_j(\mathbf{u_{k+1}})(\mathbf{x} - \mathbf{u_{k+1}}) \leq 0 \tag{3.11}$$

Following that, the linearized constrained optimization problem is solved again. This process is repeated until a global minimum is found that meets the nonlinear constraints. The algorithm is depicted graphically in figure 3.2. The key idea is to approximate the feasible

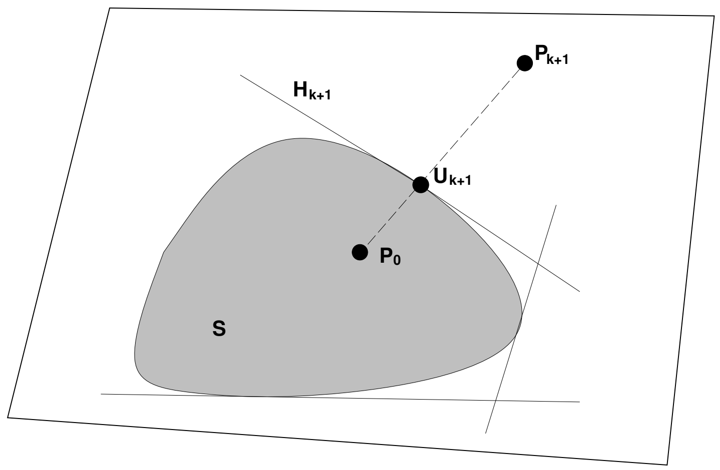

Figure 3.2: Supporting Hyperplane Method

space with a polytope, using the tangent hyperplanes at the border points of the feasible space. Then, the linear problem is solved using linear optimization.

Principal advantages of this method are:

- only first derivative information is needed to compute the equations for the tangent plane. Thus the problem created by simulator inaccuracies is smaller than methods using second derivative information.
- derivatives are only needed in the feasible space, where the constraint functions are well defined. In the case of the PLL a great problem is eliminated, since the jitter constraint is not defined when the system is unstable.
- the linear subproblem can be solved easily and efficiently by employing some linear optimization tool, in this case MINOS [11].
- provided that the feasible space is convex, a global optimum is computed instead of some local optimum that other nonlinear optimization methods compute.

Those advantages are mitigated by the assumption about the convexity of the problem. In most circuit design problems, convexity cannot be guaranted. Sometimes this can cause the algorithm to fail, but in most cases, even in non-convex problems, a local minimum can be found.

```
set_initial_feasible_point;
repeat
    do_linear_optimization;
    foreach non_linear_constraint
        check_constraint;
    if not_feasible
        find_border_point_binary_search;
        compute_gradients;
        add_new_linear_constraint;
until optimal_is_feasible
```

Figure 3.3: Optimization algorithm

Another source of failure of the algorithm can be the inexact computation of the gradients due to simulators' finite accuracy. This can cause the new added constraint not to exclude the last nonfeasible optimal, putting the algorithm in an infinite loop. It can also cause the complete elimination of the feasible region as in the case of non-convex problems.

Great care has been taken to avoid as much as possible the above problems. Since the derivatives are calculated using finite differences, the step for the finite difference is large enough to eliminate the effect of simulator inaccuracy. However, the steps must be taken always towards the feasible direction otherwise the algorithm may wander in the infeasible region where the constraint functions are undefined. The step of the algorithm starts with a high default value and is reduced if the problem is infeasible. If the step is reduced beyond the limit where the difference is higher than the simulator inaccuracy, a different point is used for drawing the tangent plane which is more "inside" the feasible region. This process may cause loss of optimality but makes the algorithm more robust.

The pseudocode for the optimization algorithm is depicted in figure 3.3. This algorithm was used in both optimizations done for the PLL design, the high level optimization and the circuit optimization.

Chapter 4

PLL Synthesis Flow

4.1 Video Driver System Description and Specifications

Video driver systems such as RAMDAC [14] or similar [15, 16, 17, 18] have become indispensable modules in the PC and workstations graphics. The function of a RAMDAC is to convert a set of colors 2^n into another set of colors 2^m using a RAM look-up table and convert the output word into analog signal for the display. Typically $n = 8$, $m = 24$ so 256 out of 16 million possible colors can be viewed at the same time. Timing of the RAMDAC is typically provided by an external oscillator. Integration of a programmable frequency synthesizer on the same die with the RAMDAC [19] further reduces the cost and gives the flexibility of using the same chip in different systems.

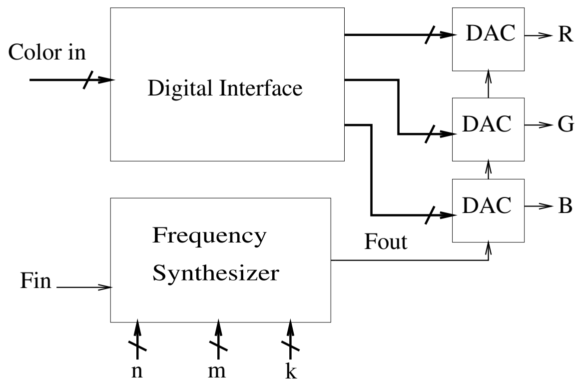

Figure 4.1: Display driver system diagram

Type	*Specification*	*Value*
Performance	Output Frequencies (MHz)	25.275, 30.24, 31.5, 34.634 36, 40, 44.9, 50, 57.284 64, 65, 75, 80, 86, 106.925, 127, 135
	Timing jitter	$\leq 1\%$
	Video signal INL	$\leq$ 1LSB
	Video signal DNL	$\leq$ 0.5LSB
	DAC resolution	8 bits
Operation	Supply voltage	5 V
Technology	Spice models	HP CMOS34
	Design rules	scmos

Table 4.1: Video driver system specifications

The video driver system we chose to implement includes two basic subsystems: first is the frequency synthesizer PLL, and second three D/A converters. The design of the PLL is described in detail in this work. This system is similar to a RAMDAC except that the SRAM lookup table is not implemented. A general block diagram of the system is shown in figure 4.1. The user can select the frequency of operation that corresponds to a specific display type and load digital data for each of the R, G, B outputs. The output of the system are the RGB currents and the synchronizing clocks for the display.

Crucial for the performance of such a system are the speed and resolution of the D/A converters, the frequency range and the timing jitter of the frequency synthesizer. The specifications for the system are given in table 4.1. A number of discrete frequencies need to be generated that correspond to different display standards. An $1u$ HP process and a 5 volt supply will be used.

4.2 Video Driver High Level Synthesis Overview

The principal blocks of the display driver system are three Digital-to-Analog converters, a frequency synthesizer PLL and digital interface for the loading of digital words in the DACs and the programming of the PLL. Figure 4.2 presents the hierarchical decomposition of the design from the highest level blocks to the lowest level components.

The specifications of table 4.1 can be immediately decomposed into DAC and

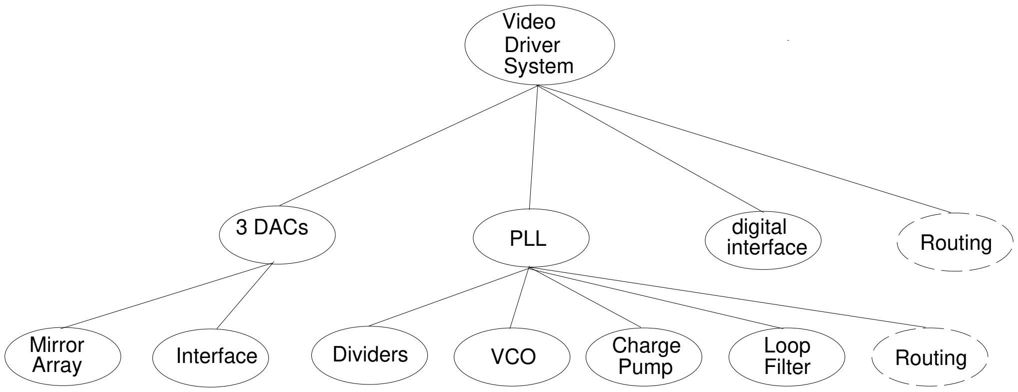

Figure 4.2: Video driver system hierarchy

frequency synthesizer specifications. The D/A synthesis hierarchy stops after the D/A specifications are given, since a module generator [13] is used for their design and layout. The simple digital interface with the rest of the system is customly designed.

This work focuses primarily on the design of the frequency synthesizer which will be described in the subsequent paragraphs.

4.3 PLL Synthesis Overview

Typically, various architectures can be explored using behavioral simulation in order to select the most suitable for the specific application. For our desing though, we selected ad-hoc one of the most popular architectures for CMOS frequency synthesizers, which is a charge-pump PLL using a ring oscillator VCO [18, 20, 19, 21]. This is due to the low cost of this architecture that does not require any external components and can be easily integrated. This comes at the expense of higher timing jitter due to the ring oscillator, which has worse jitter performance than LC tuned oscillators. In this architecture, a lead-lag RC filter is the most common in order to have a wide range of operating frequencies. High tolerances of integrated resistors and capacitors in CMOS processes could become a problem if the design parameters of the PLL are not selected carefully.

The third order charge-pump PLL of 2.2 can be used as a programmable frequency synthesizer when dividers are used inside and outside the loop as in figure 4.3. Various

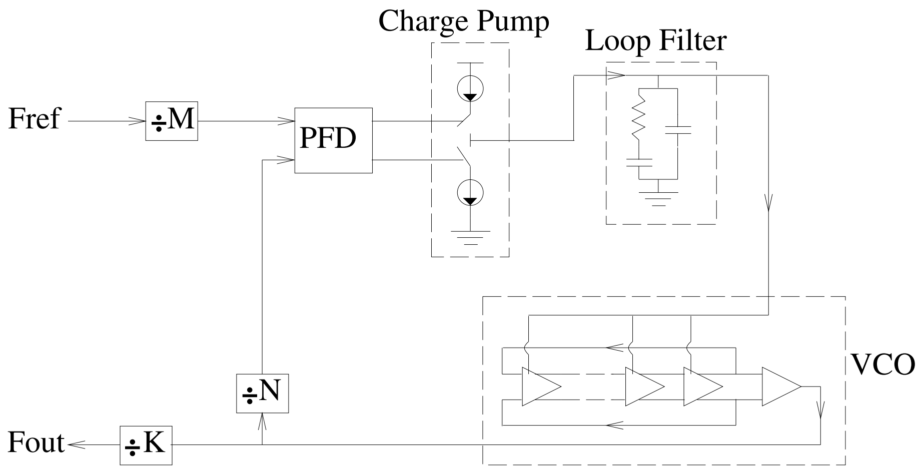

Figure 4.3: PLL programmable frequency synthesizer

integer fractions of the input clock can be realized such as:

$$F_{out} = \frac{N}{M \cdot K} \cdot F_{in} \tag{4.1}$$

The last divider by K is used so that the range of operating frequencies of the PLL is reduced and the design is made easier. For input frequency an external crystal oscillator is used and the divider values needed for the various output frequencies are taken from [19].

Table 4.2 summarizes the divider values that correspond to the desired output frequencies, together with the PLL loop frequencies.

The synthesis proceeds as in figure 4.4. First, a high level mapping is done, using optimization, from the PLL specifications to the high level behavioral parameters of the PLL. Then, the individual cells are designed and once again optimization is used to map their specifications to transistor sizes. This is done for the VCO, which is the most critical component for the PLL performance, while the other cells are customly designed. Finally, the VCO layout is synthesized using constrained optimization. The other cells are once again customly designed except for the PFD which is synthesized using OCTTOOLS.

After the design is completed, hierarchical, bottom-up verification is performed. Performance parameters for the lowest level blocks are extracted using circuit simulation. Then behavioral simulation is used to verify the performance of the whole system.

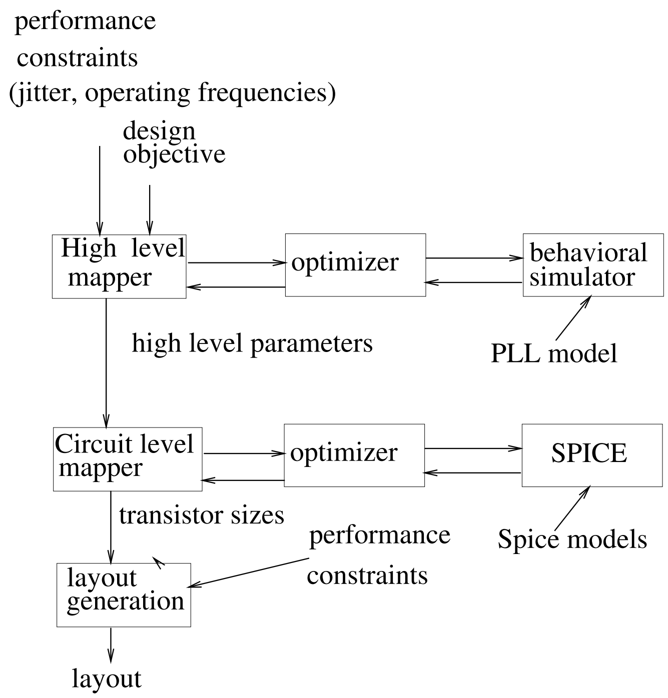

Figure 4.4: PLL synthesis path

$F_{in}(MHz)$	F_{out}	N	M	K	F_{VCO} (xF_{in})	F_{PLL} (MHz)
14.318	25.175	176	25	4	x 7.04	0.5727
	30.24	169	20	4	x 8.45	0.7159
	31.5	88	10	4	x 8.8	1.431818
	36	101	20	2	x 5.05	0.7159
	40	95	17	2	x 5.588	0.8422
	44.9	69	11	2	x 6.2727	1.302
	50	70	10	2	x 7	1.431818
	64	152	17	2	x 8.94	0.8422
	65	118	13	2	x 9.08	1.1013
	75	89	17	1	x 5.23	0.8422
	80	95	17	1	x 5.588	0.8422
	86	60	10	1	x 6	1.431818
	127	133	15	1	x 8.86	0.9545
	135	132	14	1	x 9.43	1.0227

Table 4.2: Frequency synthesizer divider values

Type	*PLL input Frequency*	*N-divider*	*VCO frequency*	*Value*
Stability	0.56 MHz	100 250	56 MHz 140 MHz	
$\frac{\Delta T}{T}$	0.56 MHz	250	140 MHz	≤ 0.007
Phase Margin				$\geq 45^o$

Table 4.3: PLL specifications

4.4 PLL specifications

The specifications for the PLL performance can be derived directly from table 4.1 combined with table 4.2. However, if the performance was to be checked for each one of the operating frequencies the optimization problem would be unnecessarily complex. From the PLL linear analysis of chapter 2, we can guarantee stability at all operating intermediate operating frequencies if we guarantee stability at the lowest frequency. Also at high frequencies, VCO overload can affect the stability of the system. Consequently, after verifying those assumptions using behavioral simulation, only two extreme frequencies for the VCO were used in the optimization that were determined by the divider values. A frequency slightly lower than the lowest PLL operating frequency was used, which is $0.572MHz$ from table 4.2. For timing jitter, the worst case accumulation occurs at the highest value of the division ratio, since the VCO jitter events accumulate by this factor, thus only this case was considered in the optimization. An additional phase margin performance constraint was added to verify PLL stability under random high level parameter deviations which can be due to lower level component process and temperature variations. For example, typical R, C variations due to IC process can be as high as 30 % of the nominal value. A phase margin of 45^o was found to be adequate to verify stability under such conditions.

Using those observations, all specifications can be mapped to the ones of table 4.3. Specifications have been tightened with respect to the ones of table 4.1 in order to provide for an extra safety margin.

Module	*Parameter name*	*Description*
VCO	K_o	Voltage to frequency gain
	F_o	Central frequency
	$\frac{\Delta T}{T}$	Timing jitter
	V_{min}	Lower saturation Voltage
	V_{max}	Upper saturation Voltage
Charge-pump	I_p	nominal charge-pump current
	$\frac{\Delta I_p}{I_p}$	up and down current mismatch
	R_{lin}	linear region resistance
	R_{sat}	saturation region resistance
	V_{satup}	upper source saturation voltage
	V_{satlo}	lower source saturation voltage
Loop Filter	R	Resistor value
	C_1, C_2	Capacitor values
PFD		State transition table
	t_{dead_zone}	Dead zone
	t_d	Delay
Divider	t_d	Delay

Table 4.4: PLL High level parameters

4.5 High Level Synthesis

According to the behavioral models for the charge-pump PLL presented in chapter 3, the intermediate level parameters for the high level synthesis are shown in table 4.4.

Each one of the performance constraints can be interpreted as a function of the above variables. However, an optimization problem including all the above parameters would be unnecessarily complicated since many parameters have little effect on the performance constraints and given a technology, values close to optimal could be chosen manually simply using designer's experience.

In the case of the VCO, given the topology described in section 4.6.1.1, typical values can be easily chosen for V_{min}, V_{max}. The VCO center frequency F_o, can be set in the middle of the desired frequency range. Manual selection can also be used for the parameters $\frac{\Delta I_p}{I_p}$, R_{lin}, R_{sat}, V_{satup}, V_{satlo} of the charge pump, since small variations of them hardly affect the PLL performance as shown by behavioral simulation. Digital delays do not affect the system performance and the PFD dead zone problem can be eliminated with techniques described in section 3.2.1.

Module	*Parameter name*	*Value*
VCO	F_o	100MHz
	V_{min}	1.8V
	V_{max}	5V
Charge-pump	$\frac{\Delta I_p}{I_p}$	5%
	R_{sat}	$2M\Omega$
	V_{satup}	4.8V
	V_{satlo}	0.2V
PFD	t_{dead_zone}	0
	t_d	2nsec
Divider	t_d	2nsec

Table 4.5: Constant high level PLL parameters

Selecting manually the parameters which do not affect optimality does not invalidate the top-down approach. Those values will be constraints for the next levels of hierarchy. Table 4.5 summarizes the parameters that were manually chosen.

In order to determine the remaining parameters , crucial to the PLL performance, optimization needs to be used. As mentioned before, analytical equations for the PLL fail to capture the transient PLL behavior so behavioral simulation is used to detect stability and compute the timing jitter. For the phase margin constraint calculation, equation 2.7 is used.

4.5.1 Flexibility Functions

The objective of the optimization is to maximize the *flexibility* of the design. Flexibility is a heuristic measure of the easiness to meet the design parameters. The objective function for the high level optimization is the sum of the flexibility functions for each of the remaining design parameters. Two types of functions were used, one hyperbolic and one parabolic:

$$flex_1(x) = -\alpha x^2 + c \tag{4.2}$$

$$flex_2(x) = -\frac{\beta}{x} + c \tag{4.3}$$

The hyperbolic type is used when a higher value of the parameter is harder to meet while the parabolic is used when the opposite is true. In the case of the parabolic function, the intuitive criterion is that a zero parameter value is almost impossible to achieve, while for

flexibility type	*Parameter name*	*"Easy"*	*"Hard" 0*
parabolic	$\sqrt{\Delta\tau^2_{VCO}}$ @140MHz	7psec	1.4psec
hyperbolic	K_0	50MHz/V	80MHz/V
	I_p	$20\mu A$	$200\mu A$
	R, C_1, C_2	$0.05mm^2$	$0.1mm^2$

Table 4.6: Flexibility coefficients

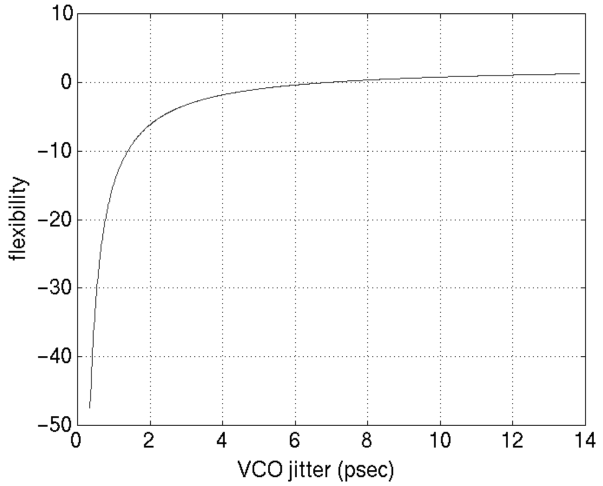

Figure 4.5: $\Delta\tau_{VCO_{rms}}$ flexibility function

higher parameters it becomes easier and for high enough values the flexibility "increase" is insignificant.

The criterion used to build the flexibility function was attributing a value of $flex(x) = -10$ for a parameter value that is "hard" to obtain and $flex(x) = 0$ for a parameter value that is "easy" to obtain. The coefficients are subsequently calculated by solving equations 4.2, 4.3. Table 4.6 summarizes the values used for the coefficient selection for the high level parameters of the PLL. The flexibilities for the R, C elements of the loop filter are expressed in terms of the area they occupy in the layout.

The flexibility functions for parameters $\Delta\tau_{VCO_{rms}}$ and K_0 are shown in figures 4.5, 4.6 respectively.

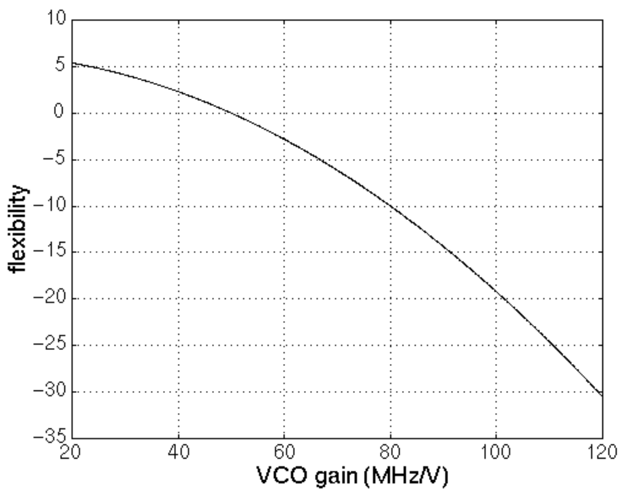

Figure 4.6: K_o flexibility function

4.5.2 Optimization Problem

The high level optimization problem mapping of the high level constraints to high level building block constraints can be expressed as:

$$maximize\ \ flex_{PLL}(\Delta\tau_{VCO}, K_0, I_p, R, C_1, C_2) \tag{4.4}$$

$$subject\ to\ \ PM_{PLL}(\Delta\tau_{VCO}, K_0, I_p, R, C_1, C_2) \geq 45^o \tag{4.5}$$

$$\Delta\tau_{PLL140MHz} \leq 50psec \tag{4.6}$$

where

$$\begin{aligned} & flex_{PLL}(\Delta\tau_{VCO}, K_0, I_p, R, C_1, C_2) = \\ & \quad flex_1(\Delta\tau_{VCO}) + flex_2(K_0) + flex_2(I_p) + \\ & \quad flex_2(R) + flex_2(C_1) + flex_2(C_2) \end{aligned} \tag{4.7}$$

Equation 4.5 expresses the phase margin constraint, necessary to guarantee a result tolerant to parameter variations. Phase margin is computed using equation 2.7. Timing jitter constraint is calculated using the behavioral simulation tool described in section 3.2. The stability constraints of table 4.3 are not explicitly set as constraints in the optimization problem described in equations 4.4- 4.6, since stability and jitter are detected in only one simulation for a given set of parameters. Stability detected by the behavioral simulator is not a continuous constraint so it would be hard to compute a measure for it. If instability is detected during the timing jitter simulation, then the simulated point is in the infeasible region so no gradients need to be calculated according to the algorithm described in 3.3.1.

Higher PLL bandwidth results in better jitter performance according to equations 2.12, 2.13. This would result in the optimization algorithm pushing the solution close to the instability region, as it can be inferred from equation 2.8. This is partially avoided again by adding a tight phase margin constraint.

In order to speed up the optimization, the phase margin constraint is always computed first and if it is satisfied the timing jitter constraint is calculated using behavioral simulation. To find the border point in the optimization algorithm of 3.3.1, phase margin constraint has priority over the jitter constraint, since it is computationally much less expensive.

From equations 3.4- 3.7 it can be shown that PLL parameters are not independent. For this reason C_2 was held constant at the optimization at 5pF. The value for C_2 was selected to be the minimum possible that could guarantee tolerance to layout parasitics.

The parameter values for the initial feasible point needed by the supporting hyperplane algorithm were selected with the help of behavioral simulation. Since the optimization algorithm will try to optimize the design any feasible point is good for the algorithm, provided that it is not on the border.

The supporting hyperplane algorithm needs a convex feasible space in order to guarantee convergence. In the case of the PLL, using the parameters of table 4.6 and the constraints of equation 4.6 results in a non-convex feasible space causing the algorithm to fail. To override this problem, the value of parameter K_0 was held constant and optimizations were done with different values of K_0. Since the VCO parameters suffer from large temperature and process variations and a heuristic measure of optimality is used, we can claim that the optimality of the solution is not affected.

4.5.3 High Level Mapping Results

Table 4.7 presents the optimization results for different values of K_0. The optimizations were done in a DEC Alpha-Server 2100 5/250 with 256Mb of memory and 4

		Optimization results			
Parameters	K_0 *(MHz/V)*	35	40	50	60
	$\Delta\tau_{VCOrms}$ *(psec)*	3.1	3.33	3.46	3.33
	I_p *(μA)*	18.7	15.8	14.58	11.44
	R *($K\Omega$)*	258.2	200.5	198.7	179.6
	C_1 *(pF)*	40	57.8	51.95	66.9
	C_2 *(pF)* 5	5	5	5	5
Constraints	$\Delta\tau_{PLLrms} \leq 50psec$ *(psec)*	48.08	50.42	51.05	50.32
	Phase margin $\leq 45^o (^o)$	43.5	43.6	43.17	44.55
Objective	*Flexibility*	4.34	2.79	1.14	-3.11
CPU Time (sec)			7606.1	9198	11529
Iterations		11	8	8	9

Table 4.7: Optimization results

		Initial solutions			
Parameters	K_0 *(MHz/V)*	35	40	50	60
	$\Delta\tau_{VCOrms}$ *(psec)*	1.03	1.03	1.03	1.03
	I_p *(μA)*	5	5	5	5
	R *($K\Omega$)*	220	220	220	220
	C_1 *(pF)*	200	200	220	200
	C_2 *(pF)* 5	5	5	5	5
Objective	*Flexibility*	-38.6	-39.5	-48.9	-44.67

Table 4.8: Initial solutions for optimization algorithm

CPU's. In every case the optimization converges in not more than 11 iterations and within reasonable computation time. Table 4.8 shows the initial points used in each optimization. In every case, the results of table 4.7 show a great improvement over the initial solution.

The optimal results appear to violate slightly the constraints sometimes. This is due to the numerical calculation of the border point used in the supporting hyperplane algorithm: small violations are allowed in order to detect if a point belongs to the "border" region. The violations are minimal and have insignificant effect in the performance.

From the results of table 4.7 the one that corresponds to $K_0 = 35$ (MHz/V) yields the highest flexibility. However the result for $K_0 = 40$ (MHz/V) was selected since behavioral simulation showed that is less affected by parameters variations. This by no means reduces the value of the methodology since it also involves designer's judgment in the constraint propagation process. Optimization is used as a tool to propose various solutions while the designer makes the final decision.

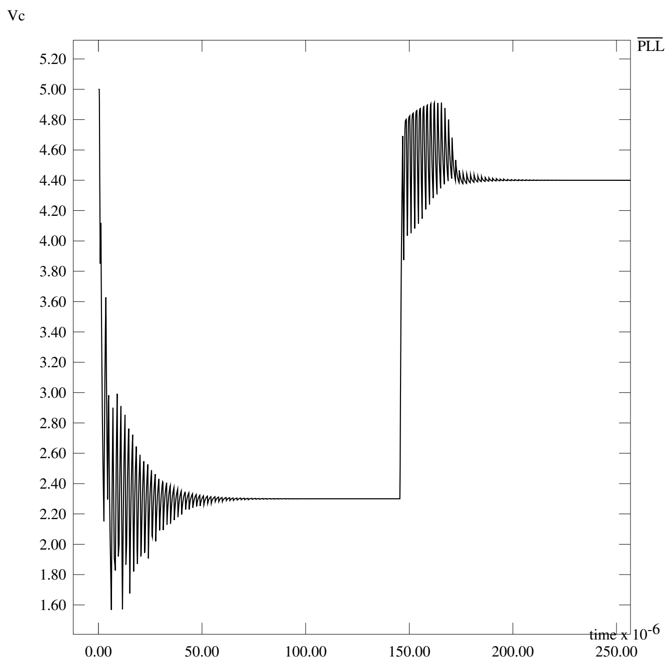

Figure 4.7: PLL behavioral simulation

Figure 4.7 shows the output of the behavioral simulator for the selected parameters. The graph presents the transient simulation of the VCO control voltage. The PLL is simulated for two values of the loop divider.

In figure 4.8 a graph of the input and output timing jitter is shown from the Monte Carlo simulation for the jitter computation. The graph presents the input and output jitter events in every transition. The open loop characteristic for gain and phase is depicted in figure 4.9 as calculated from equation 2.7.

Since parameters variations were only taken into account indirectly , through the phase margin constraint, it was necessary to verify the performance of the PLL at the worst case variations of the top level parameters. For the R, C parameters those worst case variations were given by the process and for I_p ,K_0 typical parameters were selected. The PLL was found to meet the constraints for the variations of table 4.9.

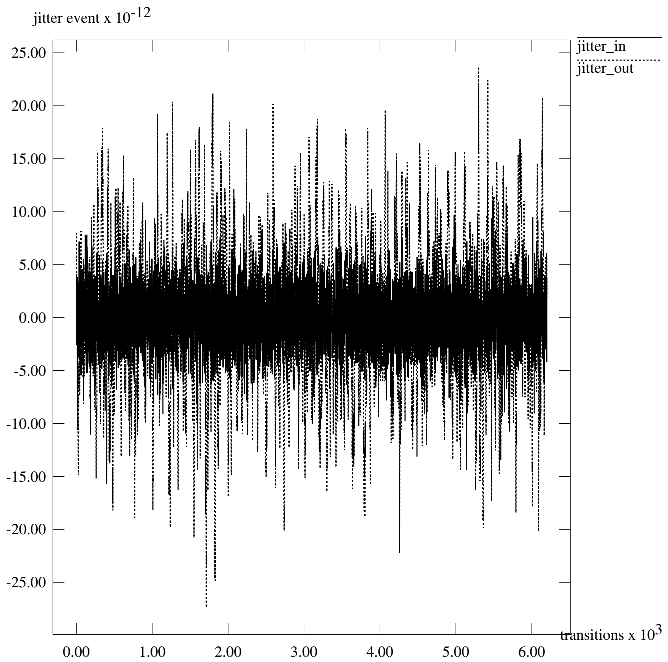

Figure 4.8: PLL jitter simulation

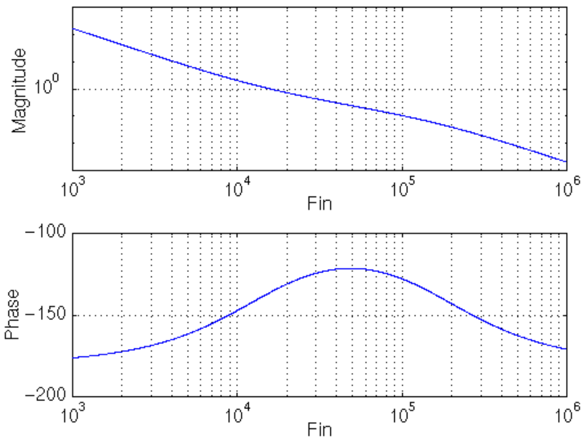

Figure 4.9: PLL open loop characteristic

Parameter	**Value**	**Tolerance**
K_0	$40MHz/V$	$\pm10MHz/V$
F_0	$100MHz$	$\pm30MHz$
I_p	$16\mu A$	$\pm1\mu A$
R	$200K\Omega$	$\pm60K\Omega$
C_1	$57.8pF$	$\pm6pF$
C_2	$5pF$	$\pm1pF$

Table 4.9: PLL maximum parameter tolerances

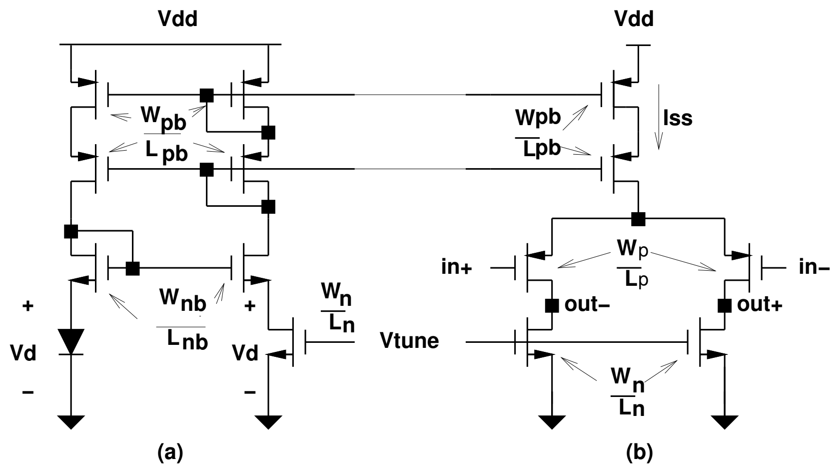

Figure 4.10: VCO delay cell and bias circuit

4.6 Circuit Level Synthesis

After the selection of the high level parameters, the next step of the methodology is building the low level blocks. The most important block that affects the PLL performance is the VCO and will be discussed extensively.

4.6.1 Voltage Controlled Oscillator

4.6.1.1 Circuit description

As mentioned before, a ring oscillator VCO topology was selected. The best topology to decrease the effect of power supply and substrate coupling is the differential cell with CMOS loads in triode region, as used in [18, 20, 22].

A modified version of the cell topology described in [18] was used, since it was the simplest way our requirements could be met. The topology of the cell with the bias circuit is shown in figure 4.10. In the bias circuit, the voltage across the NMOS transistor in triode is forced by the feedback loop to be equal to the voltage across the diode. Assuming a simple triode model and ignoring short channel effects, the bias current can be given by:

$$I_{ss} = \mu_n C_{ox} \frac{W_n}{L_n}(V_{tune} - V_{THN} - \frac{V_d}{2})V_d \tag{4.8}$$

I_{ss} is mirrored by the top mirror to the delay cell current. The NMOS loads are made

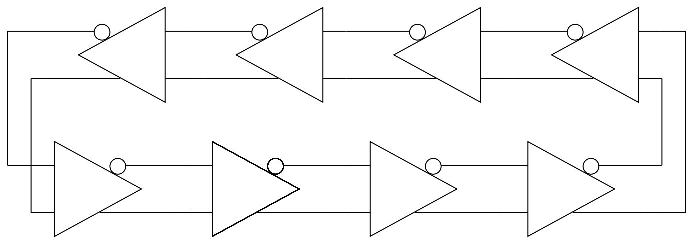

Figure 4.11: Ring oscillator VCO

equal to the NMOS bias transistor, so when the current fully switches to one branch, the maximum swing across the output of the delay cell is limited to the voltage across the diode V_d, which is approximately a constant.

Using a first order approximation, the delay of one cell can be given by:

$$t_d = \frac{V_d C_L}{I_{ss}} \tag{4.9}$$

where C_L is the capacitive load of the output node. Thus the frequency of oscillation for a N-stage ring oscillator would be:

$$F_o = \frac{I_{ss}}{2NV_d C_L} \tag{4.10}$$

which shows at first order a linear dependency with V_{tune} as desired. For the given HP 1μ technology, it was found by simulation that 8 cells were needed to give the desired VCO gain. The VCO frequency-to-voltage characteristic is linear at first order, provided that the NMOS bias transistor is in the triode region of operation. For low values of V_{tune} the transistor is in saturation and equation 4.8 is not valid.

For the ring oscillator to oscillate, it is necessary that the gain a_v of each cell is high enough. Typically $a_v \simeq 1.5$ is enough to guarantee oscillation.

4.6.1.2 VCO optimization problem

Following the proposed methodology, the performance constraints for the VCO must be mapped onto component values and layout parasitics. This is depicted in figure 4.12.

The bias circuit parameters $W_{nb}, L_{nb}, W_{pb}, L_{pb}$ do not affect the VCO performance as long as the current mirrors work properly. The parameters that really affect the VCO

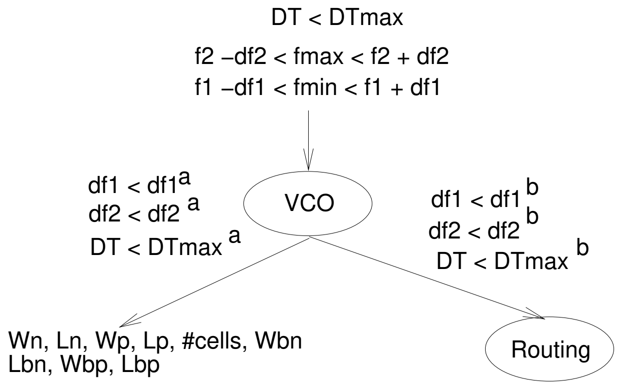

Figure 4.12: VCO constraint propagation

W_{nb}	L_{nb}	W_{pb}	L_{pb}	L_p
$10\mu m$	$1\mu m$	$20\mu m$	$1\mu m$	$1\mu m$

Table 4.10: VCO bias circuit and cell pre-selected parameters

performance are W_n, L_n, W_p, L_p. Since the objective of the optimization is to minimize power dissipation, parameter L_p can also be set to the minimum possible value so that the load capacitance is minimized and the I_{ss} needed to achieve a certain delay is smaller according to equation 4.10. The transistor sizes selected for the bias circuit and L_p are given in table 4.10.

The remaining parameters were obtained by optimization. The objective was to minimize power. The VCO constraints of tables 4.9, 4.7 can be converted to constraints for a minimum and a maximum VCO frequency. The corresponding minimum and maximum applied voltages are almost constants for the specific topology, given our technology. In figure 4.12 it is shown how the constraints are being partitioned between the circuit design and the layout. We partition each constraint P_i in ${P_i}^{\alpha}$, ${P_i}^{\beta}$, such as $P_i = {P_i}^{\alpha} + {P_i}^{\beta}$, thus making sure that the performance constraints will be met after the final layout. The timing jitter constraint is being partitioned into a part depending on thermal noise, which is being taken care of in the circuit optimization and a part depending on substrate and power supply coupled noise, which depends on the layout. $\Delta {f_i}^{\alpha}$ are used for the circuit level optimization so that the method converges and $\Delta {f_i}^{\beta}$ are budgeted for the layout phase.

The sensitivity of the VCO performance to layout parasitics can be greatly reduced if those sensitivities are used in the optimization process. In our specific optimization problem, the optimization algorithm will try to adjust I/C_{out} in order to achieve the desirable frequency performance, while minimizing power dissipation. This will result in the use of minimum size transistors, so the final circuit will be extremely sensitive to layout parasitic capacitances, which was observed in preliminary optimizations.

Let $\mathbf{P}$ a performance vector, $\mathbf{C}$ the parasitics vector and $\mathbf{\Delta P_{max}}$ the corresponding maximum allowed performance degradation due to those parasitics. Assuming a linear model around the nominal performance and small parasitics, the performance degradation ΔP_i can be given by:

$$\Delta P_i = \left[\mathbf{S}_{\mathbf{C}}^{P_i}\right]^T \cdot \mathbf{\Delta C} \tag{4.11}$$

where

$$\left[\mathbf{S}_{\mathbf{C}}^{P_i}\right]^T = \left[\left.\frac{\partial P_i}{\partial C_1}\right|_{C_1 = C_{1\,NOM}}, \left.\frac{\partial P_i}{\partial C_2}\right|_{C_2 = C_{2\,NOM}}, \ldots, \left.\frac{\partial P_i}{\partial C_{N_C}}\right|_{C_{N_C} = C_{N_C\,NOM}} \right] \tag{4.12}$$

and $\mathbf{\Delta C}$ is the deviation from the nominal value of the parasitics. Given an estimate $\mathbf{\Delta C_{max}}$ of the maximum deviations from the nominal parasitics, we can include the following constraint to our optimization problem:

$$\left[\mathbf{S}_{\mathbf{C}}^{\mathbf{P}}\right]^T \cdot \mathbf{\Delta C_{max}} \leq \mathbf{\Delta P_{max}} \tag{4.13}$$

This will force the optimization result to have a reduced sensitivity to parasitics.

For the case of the VCO optimization, the parasitic capacitances at the outputs of the differential gates were used, since they have a significant effect to the output frequency. The layout effect was based on an a-priori estimation of the maximum allowed deviation from the nominal value of layout parasitics, which is subsequently used as a constraint for the VCO layout generator. The constraints were evaluated at a maximum deviation of 50% from a nominal estimate of $15pf$ for the parasitics, which were found to be reasonable values for the layout synthesis phase.

Another two circuit specific constraints were added regarding the minimum and maximum output voltages. Those constraints are necessary for the level restoring circuit to work. Table 4.11 summarizes the constraints for the VCO optimization problem.

The overall optimization problem for the VCO can be expressed as:

Parameter name	*Parameter description*	*Constraint Value*
F_{max}	output at $V_{in} = 4.8V$	$\leq 173MHz$ $\geq 163MHz$
F_{min}	output at $V_{in} = 1.8V$	$\leq 58MHz$ $\geq 38MHz$
$\Delta\tau_{VCOrms}$	rms VCO jitter at 135 MHz	$\leq 3psec$
V_{max}	maximum output voltage	$\geq 0.5V$
V_{min}	minimum output voltage	$\leq 0.1V$
$\frac{\Delta F_{max}}{F_{max}}$	maximum variation of output due to layout parasitics	$\leq 10\%$

Table 4.11: VCO optimization constraints

$$\begin{aligned}
minimize\ \ Power_{VCO}(W_n, L_n, W_p) & & & \quad (4.14)\\
subject\ to\ \ F_{maxVCO}(W_n, L_n, W_p) &\leq F_{maxmax} & & \quad (4.15)\\
F_{maxVCO}(W_n, L_n, W_p) &\geq F_{maxmin} & & \quad (4.16)\\
F_{minVCO}(W_n, L_n, W_p) &\leq F_{minmax} & & \quad (4.17)\\
F_{minVCO}(W_n, L_n, W_p) &\geq F_{minmin} & & \quad (4.18)\\
\Delta\tau_{VCOrms}(W_n, L_n, W_p) &\leq \Delta\tau_{max} & & \quad (4.19)\\
V_{max}(W_n, L_n, W_p) &\geq V_{max} & & \quad (4.20)\\
V_{min}(W_n, L_n, W_p) &\leq V_{min} & & \quad (4.21)\\
\frac{\Delta F_{max}}{F_{max}} &\leq P_{tol} & & \quad (4.22)
\end{aligned}$$

The optimization problem was solved using the supporting hyperplane algorithm described in section 3.3.1. All constraints were evaluated using SPICE simulations except for the timing jitter constraint that was evaluated using equation 2.14. Since transistor sizes can only take discrete values, the results were truncated to the nearest feasible value, taking into account the technology specific parameters ΔL, ΔW, $\lambda = 0.6\mu m$. Table 4.12 summarizes the optimization results. The initial point satisfying the constraints is provided for the supporting hyperplane algorithm to work.

		Initial	Final
Parameter name	W_n	$3.8\mu m$	$2.6\mu m$
	L_n	$4\mu m$	$4\mu m$
	W_p	$55\mu m$	$36\mu m$
Performance variable	F_{max}	165MHz	163MHz
	F_{min}	50MHz	48MHz
	$\Delta\tau_{VCO rms}$	1.25psec	1.5psec
	V_{max}	0.67	0.66 V
	V_{min}	0 V	0 V
	$\frac{\Delta F_{max}}{F_{max}}$	6.8%	10%
Objective	Power	8.6 mW	6.2mW
Iterations		4	
CPU time		2069 CPU sec	

Table 4.12: VCO optimization results

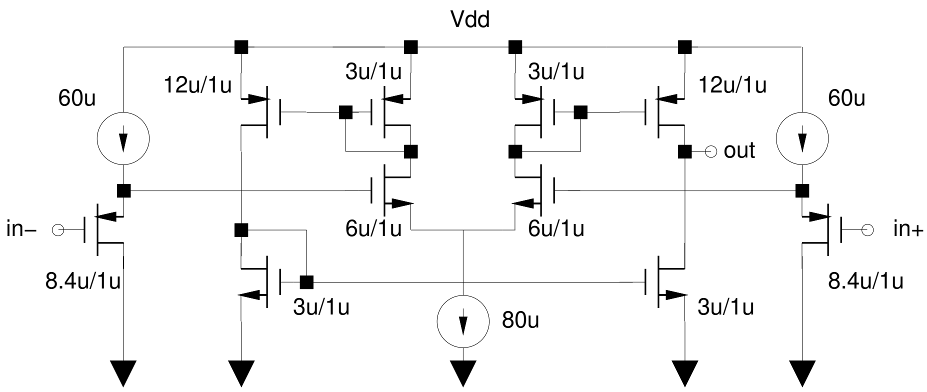

Figure 4.13: Level restoring circuit

4.6.2 Level restoring circuit

The specific VCO topology used produces a signal of V_d maximum amplitude differential signal. This signal has to be converted to full CMOS logic levels for the needs of the digital parts of the chip. This task is performed by the level restoring circuit of figure 4.13.

The differential signal is level shifted by two level shifters and then applied to a unity gain differential pair. The output of the differential pair is amplified and restored to singled ended CMOS levels. The signal is buffered by a buffer chain not shown in figure 4.13. The total static power dissipation of the level restorer is 1mW.

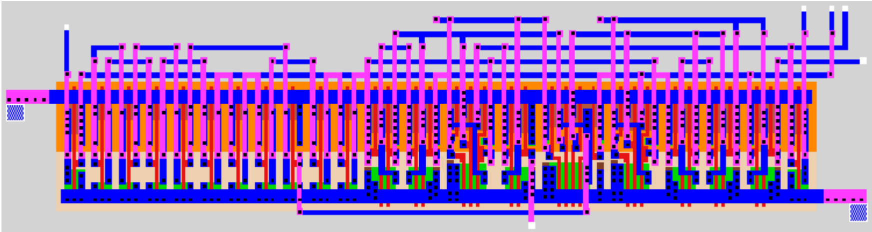

Figure 4.14: PFD layout

4.6.3 Phase-Frequency Detector

The Phase-Frequency Detector used is the one of figure 2.6. The dead zone problem is avoided by adding delays in the loop using buffers. The PFD generates two minimum length UP and DOWN pulses even when the compared waveforms are perfectly synchronized. Those pulses are identical in length, so the up and down currents cancel each other assuming good matching. When the pulses are not perfectly synchronized, the difference between the pulses equals the phase difference between the compared clocks. The first order function of the PFD is identical to the one described in figure 2.3.

The PFD layout was automatically synthesized from the input schematic using OCTTOOLS [23] and is shown in figure 4.14.

4.6.4 Charge Pump

The charge-pump used is the one of figure 4.15 and is based on the one used in [20]. It consists of two cascode current sources with equal currents. The sources were cascoded in order to make the two currents match as much as possible. Since the PFD eliminates the dead zone by turning both sources on at the same time and for the same period, a mismatch could inject extra noise to the output node. They are biased from a high swing cascode bias circuit shown in figure 4.16 and is based on the one in [24].

In order to avoid the problem of turning on and off the sources every time they switch, which could cause charge injection and slow response of the charge pump, when any of the UP or DOWN pulses is low, the corresponding current source is redirected to the output of a unity gain buffer kept at the same voltage as the output of the loop filter. This

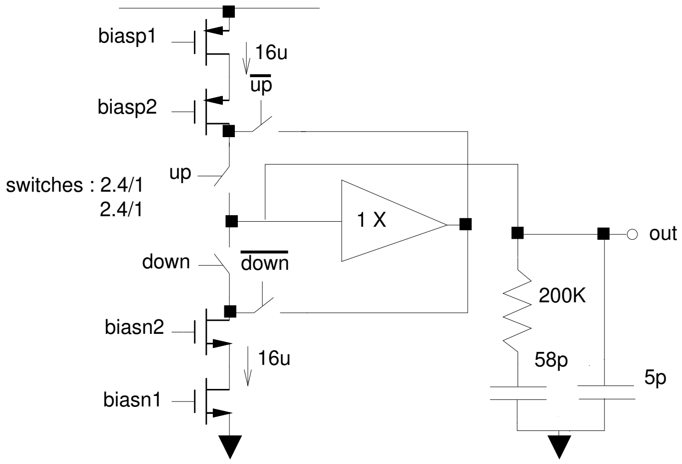

Figure 4.15: Charge pump circuit

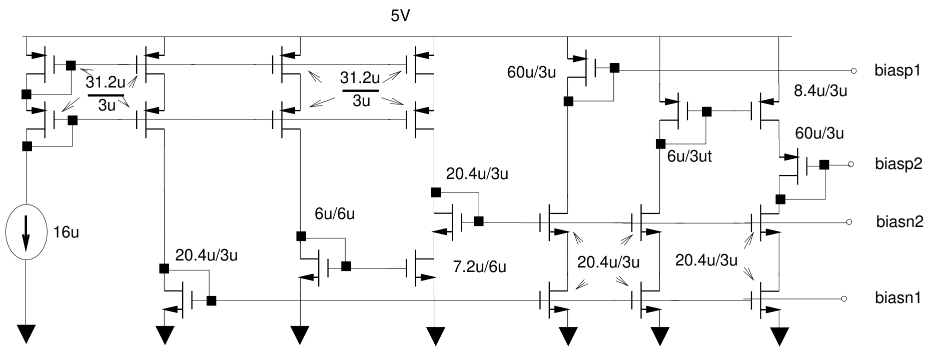

Figure 4.16: Charge pump bias circuit

prevents the current fluctuations which are due to finite output resistance of the current sources.

4.6.5 Dividers

The divider used was the synchronous programmable divider of figure 4.17. It is basically a down counter which after reaching zero it loads the programming word. The logic family used was true single phase clock family (TSPC), since fast dividers operating at around 150MHz were needed. The basic D flip-flop element used is shown in figure 4.18.

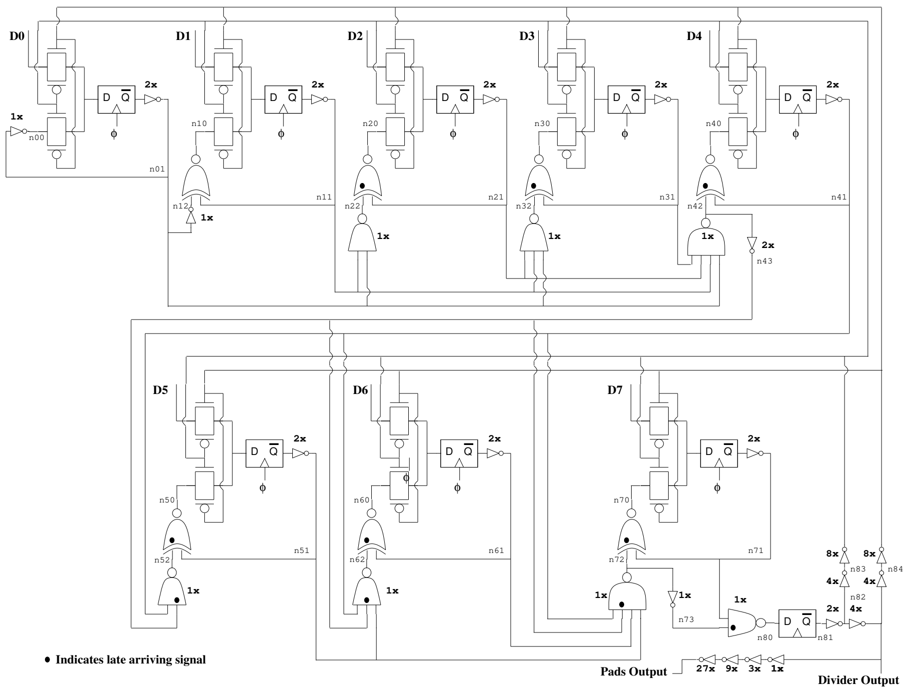

Figure 4.17: Programmable divider

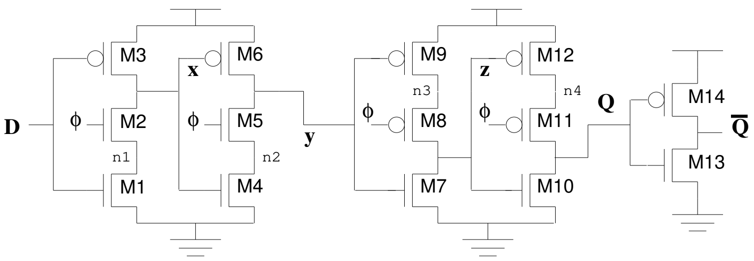

Figure 4.18: TSPC flip-flop

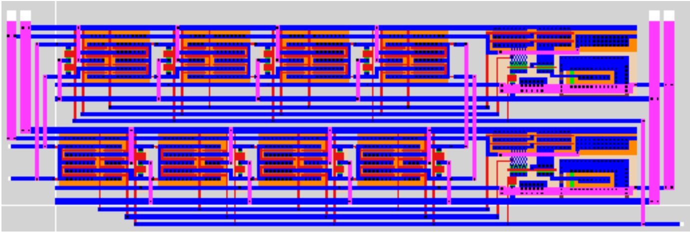

Figure 4.19: Ring oscillator VCO layout

4.7 Layout

4.7.1 Voltage Controlled Oscillator

The hierarchical, constraint-driven approach was also used in the layout phase for the VCO. The constraints set in the optimization problem of equations 4.14 - 4.22 were used in the layout generation. In addition to that, constraints for all parasitics were generated using the constraint generation techniques described in [25].

A parametric layout generator was written for the specific VCO topology. It takes as parameters the number of delay cells, the device sizes and the parasitic constraints. The final result is shown in figure 4.19.

4.7.2 PLL

Since parasitic constraints were not as critical for the other PLL sub-blocks, the charge-pump, dividers, bias and level restoring circuits described in the previous sections were done in a custom mode. On-chip decoupling capacitors were used to minimize the effect of power supply coupling to the overall jitter performance. The final PLL system was routed using Mosaico [26]. The result is shown in figure 4.20. The PLL area is approximately $0.45mm^2$.

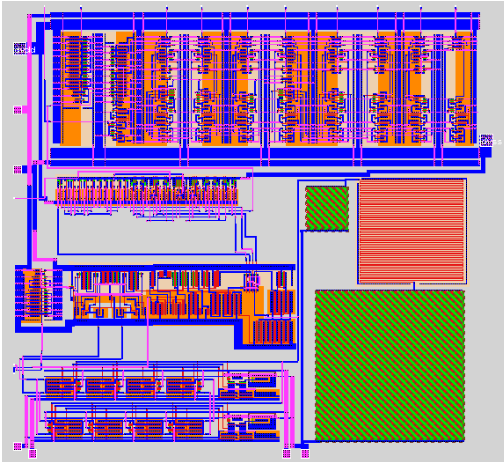

Figure 4.20: PLL layout

4.7.3 Video Driver System

The final layout for the Video Driver System was also put together using Mosaico. It is shown in figure 4.21. The die is approximately $3.4mmx3.9mm = 13.26mm^2$. Different analog and digital supplies were used and special supplies were provided for the VCO in order to avoid as much as possible supply coupled noise which can contribute to timing jitter.

4.8 Bottom-up verification

The overall design was verified using hierarchical simulation. First the functionality of the lowest level sub-blocks, such as VCO, dividers, charge-pump, was verified using SPICE and the parameters needed for the higher level blocks were extracted. The timing jitter performance of the VCO was extracted using the non-Monte Carlo, non-linear noise simulator described in [10]. Then behavioral simulation was used to verify the performance of the whole system.

The extracted parameters of the blocks used in the PLL behavioral simulation are given in table 4.13. To illustrate the value of bottom-up verification, the result of a flat circuit simulation is compared to the result of the behavioral simulation in figure 4.22. The waveform simulated is the control voltage of the VCO when the PLL is in acquisition

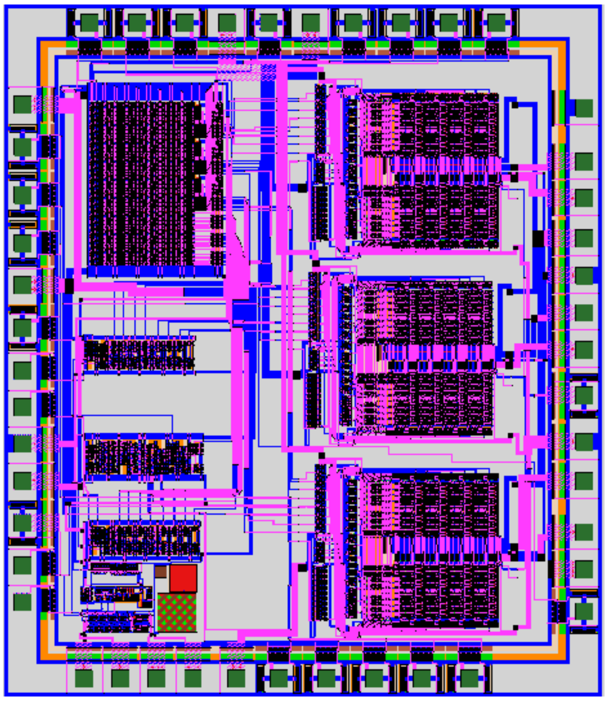

Figure 4.21: Video Driver System layout

Module	*Parameter name*	*Value*
VCO	K_o	39 MHz/V
	F_o	110 MHz/V
	$\Delta\tau_{VCO\,rms}$	4psec
	V_{min}	1.4V
Charge-pump	I_p	16u
	$\frac{\Delta I_p}{I_p}$	10%(est)
	R_{linup}	$20K\Omega$
	R_{satup}	$500M\Omega$
	V_{satup}	$4.7V$
	R_{linlo}	$13.3K\Omega$
	R_{satlo}	$2000M\Omega$
	V_{satlo}	$0.33V$
Loop Filter	R	$200K\Omega$
	C_1	58 pF
	C_2	5 pF
PFD	t_{dead_zone}	0
	t_d	2nsec
Divider	t_d	2nsec

Table 4.13: PLL extracted parameters

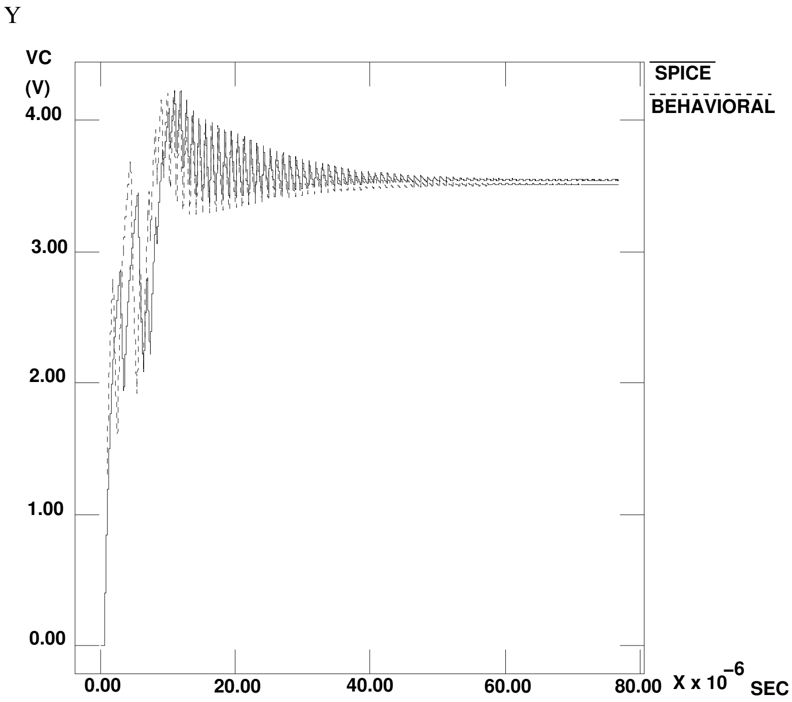

Figure 4.22: PLL Verification

Type	*PLL input Frequency*	*N-divider*	*VCO frequency*	*Constraint*	*Result*
Stability	0.56 MHz	100 250	56 MHz 140 MHz	ok ok	ok ok
$\frac{\Delta T}{T}$	0.56 MHz	250	140 MHz	≤ 0.007	0.01
Phase Margin				$\geq 45^o$	45^o

Table 4.14: PLL verification results

mode and is done to detect stability in the worst case divide ratio. Waveforms from both simulations are almost identical.

For the behavioral simulation, 560 CPU seconds were needed, while the full circuit simulation took 20CPU hours (using macromodels for the dividers). Both simulations were done in a DEC Alpha-Server 2100 5/250 with 256Mb of memory and 4 CPU's.

The results of the bottom-up verification are summarized in table 4.14. Noise performance when the VCO noise is calculated using the noise simulation tool is different, which is expected since equation 2.14 which is used to calculate the VCO jitter during optimization is only an approximation. The noise simulator could not be used during the optimization since it is very expensive computationally. Still, using the extracted jitter performance from the noise simulator the original constraints are met because the VCO optimization problem was overconstrained in order to compensate for the approximations done in the jitter calculations. The jitter constraint is not a hard functional constraint. The projected performance is based only in the calculation of the thermal jitter of the VCO, which sets the fundamental performance bound. Still, the performance of the actual system is expected to be close to the one predicted since care has been taken to eliminate all other jitter sources.

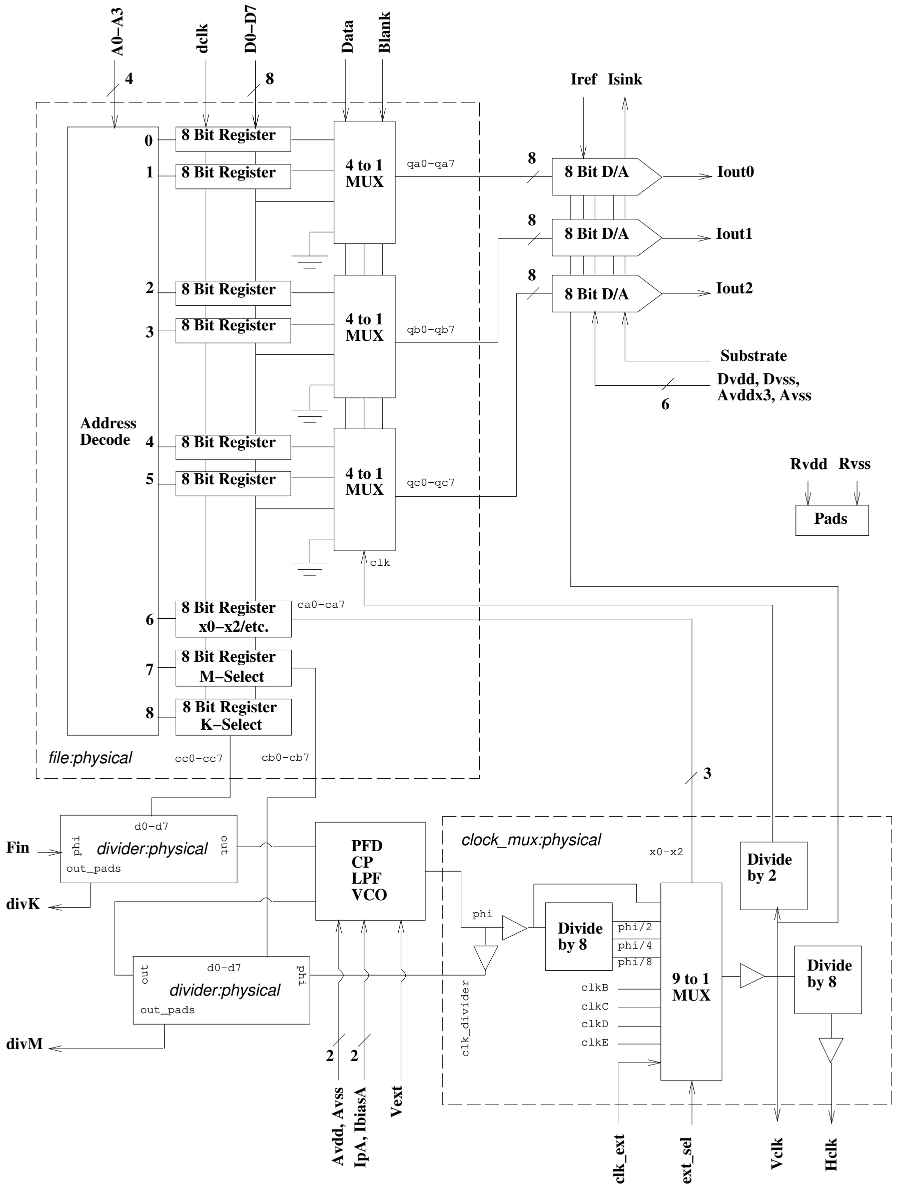

Figure 4.23: Video Driver System schematic diagram

Chapter 5

Conclusions

We have presented the full design flow of a PLL based frequency synthesizer, following the top-down, constraint-driven paradigm. The frequency synthesizer is part of a video driver system that was also designed in a top-down mode.

Fundamental to our approach is the use of behavioral simulation and optimization for constraint propagation. For the needs of our design example, existing behavioral simulation tools had to be enriched with accurate models capturing the most important second order effects which affected the system performance. An optimization algorithm tolerant to simulator inaccuracies was also implemented. The design was carried out until the layout of the overall system was generated. Bottom-up verification was used to guarantee the system level performance to be within specifications.

This work has shown the applicability of the proposed methodology [1] in yet more complex systems. Our methodology can help reduce the analog circuit design cycle by making the analog design process more systematic. We plan to fabricate and test the system we have designed to further validate our approach. In the future, we plan to expand the domain of applicability of the proposed methodology in yet more complex and high performance systems, such as RF receivers.

Bibliography

[1] H. Chang, A. Sangiovanni-Vincentelli, F. Balarin, E. Charbon, U. Choudhury, G. Jusuf, E. Liu, E. Malavasi, R. Neff and P. Gray, "A Top-down, Constraint-Driven Design Methodology for Analog Integrated Circuits", in *Proc. IEEE CICC*, pp. 841–846, May 1992.

[2] H. Chang, E. Liu, R. Neff, E. Felt, E. Malavasi, E. Charbon, A. Sangiovanni-Vincentelli and P. R. Gray, "Top-Down, Constraint-Driven Methodology Based Generation of n-bit Interpolative Current Source D/A Converters", in *Proc. IEEE CICC*, pp. 369–372, May 1994.

[3] H. Chang, E. Felt and A. Sangiovanni-Vincentelli, "Top-Down, Constraint-Driven Design Methodology Based Generation of a Second Order Σ-Δ A/D Converter", in *Proc. IEEE CICC*, pp. 533–536, May 1995.

[4] F. M. Gardner, *Phaselock Techniques, 2nd ed.*, J. Wiley & Sons, New York, 1979.

[5] F. M. Gardner, "Charge-Pump Phase-Lock Loops", *IEEE Trans. on Communications*, vol. COM-28, n. 11, pp. 1849–1858, November 1980.

[6] B. Kim, T. C. Weigandt and P. R. Gray, "PLL/DLL System Noise Analysis for Low Jitter Clock Synthesizer Design", in *Proc. IEEE Int. Symposium on Circuits and Systems*, May 1994.

[7] T. C. Weigandt, B. Kim and P. R. Gray, "Analysis of Timing Jitter in CMOS Ring Oscillators", in *Proc. IEEE Int. Symposium on Circuits and Systems*, May 1994.

[8] A. Demir, E. Liu, A. Sangiovanni-Vincentelli and I. Vassiliou, "Behavioral Simulation Techniques for Phase/Delay-Locked Systems", in *Proc. IEEE CICC*, pp. 453–456, May 1994.

[9] E. Liu and A. Sangiovanni-Vincentelli, "Behavioral Representation for VCO and Detectors in Phase-Lock Systems", in *Proc. IEEE CICC*, pp. 1231–1234, May 1992.

[10] A. Demir, E. Liu and A. Sangiovanni-Vincentelli, "Time-Domain non-Monte Carlo Noise Simulation for Nonlinear Dynamic Circuits with Arbitrary Excitations", in *Proc. IEEE ICCAD*, pp. 598–603, November 1994.

[11] B. A. Murtagh and M. A. Saunders, "MINOS 5.1 User's Guide", Technical Report Rep. SOL 83-20R, Dept. of Operations Research, Stanford University, Stanford, CA, January 1987.

[12] R. Harjani, R. A. Rutenbar and L. R. Carley, "OASYS: A Framework for Analog Circuit Synthesis", *IEEE Trans. on CAD*, vol. 8, n. 12, pp. 1247–1266, December 1989.

[13] R. Neff, P. Gray and A. Sangiovanni-Vincentelli, "A Module Generator for High Speed CMOS Current Output Digital/Analog Converters", in *Proc. IEEE CICC*, pp. 481–484, May 1995.

[14] Brooktree Corporation, *Brooktree Graphics and Imaging Product Databook*, 1993.

[15] G. Work, G. Talbot, A. Ferris, N. Henderson and P. McGuiness, "A 20 nS color lookup table for faster scan displays", in *Proc. IEEE International Solid-State Circuits Conference*, pp. 310–311, February 1985.

[16] K. Chi et al., "A CMOS triple 100-Mbit/s video DA converter with shift register and color map", *IEEE Journal of Solid State Circuits*, vol. SC-21, n. 6, pp. 989–996, December 1986.

[17] J. Leonard, N. Weste, L. Bodony, S. Harston and R. Meaney, "A 66-MHz DSP Augmented RAMDAC for Smooth-Shaded Graphic Applications", *IEEE Journal of Solid State Circuits*, vol. SC-26, n. 3, pp. 989–996, March 1991.

[18] D. Reynolds, "A 320MHz CMOS Triple 8b DAC with On-Chip PLL and Hardware Cursor", in *Proc. IEEE International Solid-State Circuits Conference*, pp. 50–51, February 1994.

[19] Chrontel, Inc., San Jose, CA, *CH8398 16-bit Interface ChronDAC*^{TM} *True-Color RAMDAC + Dual Clocks*, 1994.

[20] I. A. Young, J. K. Greason and K. L. Wong, "A PLL Clock Generator with 5 to 110 MHz of Lock Range for Microprocessors", *JSSC*, vol. 27, n. 11, pp. 1599–1607, November 1992.

[21] D. Jeong, G. Borrielo, D. Hodges and R. Katz, "Design of PLL-Based Clock Generation Circuits", *IEEE Journal of Solid State Circuits*, vol. SC-22, n. 2, pp. 255–261, February 1987.

[22] B. Kim, D. Helman and P. R. Gray, "A 30-MHz Hybrid Analog/Digital Clock Recovery Circuit in 2-μm CMOS", *IEEE Journal of Solid State Circuits*, vol. 25, n. 3, pp. 1385–1394, December 1993.

[23] D. Harrison, P. Moore, R. L. Spickelmier and A. R. Newton, "Data Management and Graphics Editing in the Berkeley Design Environment", in *Proc. IEEE ICCAD*, pp. 24–27, November 1986.

[24] C. A. Laber, C. F. Rahim, S. F. Dreyer, G. T. Uehara, P. T. Kwok and P. R. Gray, "A Family of Differential NMOS Analog Circuits for a PCM Codec Filter Chip", *IEEE Journal of Solid State Circuits*, vol. SC-22, N.2, pp. 181–189, April 1987.

[25] U. Choudhury and A. Sangiovanni-Vincentelli, "Constraint Generation for Routing Analog Circuits", in *Proc. IEEE/ACM DAC*, pp. 561–566, June 1990.

[26] J. Burns, A. Casotto, M. Igusa, F. Marron, F. Romeo, A. Sangiovanni-Vincentelli, C. Sechen, H. Shin, G. Srinath and H. Yaghutiel, "MOSAICO: An integrated Macro-cell Layout System", in *VLSI '87, Vancouver, Canada*, August 1987.