PLL Transistor Level Implementation

A

Project Report

Submitted in Partial Fulfillment of the Bangalore University

for the Degree

of

Bachelor of Technology

in

Electronics and
Communication Engineering

by

Abstract

PLL TRANSISTOR LEVEL IMPLEMENTATION

Group Members: Mohammed Hamza, Nanda kishore, def,
xyz

Guide: Dr BPH, Dr HMV

Keywords:

In this project we have endevoured to Design transistor level circuit to design a pll at technology of 180nm

A PLL is closed loop frequency system that locks phase of an output signal to an input reference signal ie. the ouput signal is an function of the input signal and the term lock hear means. PLL is used in many applications such as clock generation, clock recovery, frequency synthesis, demodulation and so on. The PLL is a feedback system that compares the phase of the output signal with the phase of the input reference signal and adjusts the output signal to minimize the phase difference. The PLL consists of three main components: a phase detector, a loop filter, and a voltage-controlled oscillator (VCO). The phase detector compares the phase of the input reference signal with the phase of the output signal and generates an error signal that represents the phase difference. The loop filter processes the error signal to remove high-frequency noise and generates a control voltage that is used to adjust the frequency of the VCO. The VCO generates an output signal whose frequency is proportional to the control voltage.

The design is carried out using LTspice with focus to demonstrate proof of concept

Dedication

To our families, friends, Guide, and all others that supported us in this work.

Acknowledgements

We sincerely express our gratitude to our guide, **Dr BPH**, for his invaluable guidance, support, and encouragement throughout this project. We would also extend our gratitude towards Our Department's Esteemed HoD **Dr. Kiran K**. We are thankful to the Department of Electronics and Communication Engineering, UVCE, for providing us with the resources and opportunity to work on this project. Special thanks to our peers and family for their continuous motivation and assistance. We would also like to acknowledge the support of the administrative staff, whose help ensured smooth progress of our work. Additionally, we are grateful to the authors and researchers whose work has significantly contributed to our understanding and implementation. Finally, we recognize the inspiration and insights gained from discussions and feedback provided by our friends and colleagues.

Contents

List of Figures

List of Tables

Acronyms and Abbreviations

PLL	Phase Locked Loop
VCO	Voltage Controlled Oscillator
PID	Proportional-Integral Derivative
PD	Proportional Derivative
PFD	Phase Frequency Detector
TSMC	Taiwan Semiconductor Manufacturing Company
OTA	Operational Transconductance Amplifier
FD	Frequency Divider
LF	Loop Filter
CP	Charge Pump

Symbols

C	Capacitance
R	Resistance
W	Width
L	Channel length
L	Inductor
t	Time
μ	Electron mobility
$\mathbf{C_{ox}}$	Oxide Capacitance
$\mathbf{V_{ov}}$	Voltage Overdrive
ϕ	Phase
ζ	Damping factor
ω	Frequency in radians
τ	Time constant

Chapter 1

Introduction

A PLL is a feedback system that includes a VCO, phase detector, and low pass filter within its loop.Its purpose is to force the VCO to replicate and track the frequency and phase at the input when in lock. The PLL is a control system allowing one oscillator to track with another.

$$\phi_{\text{out}}(t) = \phi_{\text{in}}(t) + \text{const.} \tag{1.1}$$

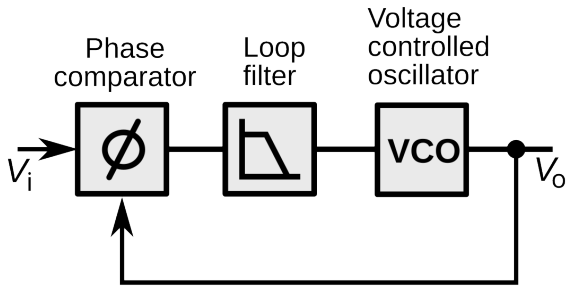


Figure 1.1: Simple analog phase locked loop

1.1 Motivation

Our team chose to work on the PLL project to gain practical knowledge about the fundamentals of phase-locked loops and their applications in modern electronics. We were particularly interested in the design and implementation of a PLL circuit at the transistor level to understand the underlying principles of PLL operation and the challenges involved in designing such circuits. We were motivated by the opportunity to work with simulation tools and techniques, such as LTspice, to analyze and optimize.

1.2 Brief History and Applications of PLL

- The PLL was invented in 1932 by Harold Stephen Black, an engineer at Bell Labs.
- The original PLL was used in telephone systems to eliminate noise and improve the quality of voice signals.
- Since then, the PLL has been widely used in various applications, including:
  - Clock generation: Ensuring that all components operate synchronously.
  - Frequency synthesis: Generating a range of frequencies from a single reference frequency.
  - Demodulation: Used in communication systems like FM radio and digital communication systems.
  - Data recovery: Recovering clock signals from data streams for proper synchronization.
  - Jitter reduction: Improving the performance of high-speed digital circuits.
  - Phase alignment: Aligning the phase of multiple signals for proper timing and synchronization.
  - Frequency modulation: Modulating signals in communication systems like FSK and PSK.
  - Signal conditioning: Filtering and conditioning signals to improve quality and reduce noise.
  - Clock recovery: Ensuring proper synchronization between transmitting and receiving devices.

1.3 Objective of the Project

- To design a PLL circuit at the transistor level using LTspice, with specific characteristics and specifications.
- Design targets:
  - Reference Frequency = 20 MHz
  - Output Frequency = 2.4 GHz
  - Power Consumption = 2 mW
  - Deterministic Jitter = 10 ps, pp
  - Random Jitter = 2 ps, rms

  - Supply Voltage = 1 V

- Carry out simulations to verify the performance of the designed PLL circuit.
- Analyze the results and compare them with the design targets.
- Identify the charachteristics of the PLL circuit for example:
  - Phase noise
  - Jitter
  - Lock time
  - Frequency stability
  - Power consumption
  - Output waveform
  - Phase margin
  - Loop bandwidth
  - frequecy capture range
- Document the design process, challenges faced, and lessons learned during the project.

Chapter 2

Literature Review

Phase-Locked Loops (PLLs) are critical components in modern electronic systems, widely used for frequency synthesis, clock generation, and signal synchronization. This section reviews the foundational concepts, advancements, and applications of PLLs as presented in the literature.

2.1 Fundamentals of PLLs

2.2 Types of PLLs

Several types of PLLs have been developed to cater to different applications:

- **Analog PLLs (APLLs):** These are the traditional PLLs that use analog components such as voltage-controlled oscillators (VCOs) and phase detectors.
- **Digital PLLs (DPLLs):** With advancements in digital technology, DPLLs have gained popularity due to their robustness and programmability.
- **All-Digital PLLs (ADPLLs):** These PLLs eliminate analog components entirely, offering better integration in digital systems.

2.3 Applications of PLLs

PLLs are employed in a wide range of applications:

- **Communication Systems:** PLLs are used for carrier recovery, clock recovery, and frequency synthesis in wireless and wired communication systems.
- **Microprocessors:** Modern processors use PLLs for clock generation and synchronization to achieve high-speed operation.
- **Power Electronics:** PLLs are utilized in grid synchronization for renewable energy systems and motor control applications.

2.3.1 Recent Advancements

2.3.2 Challenges and Future Directions

Despite significant progress, challenges remain in designing PLLs for ultra-low power applications, high-frequency operation, and integration in advanced semiconductor technologies. Future research is expected to address these challenges by leveraging emerging technologies such as quantum computing and advanced materials.

Chapter 3

Methodology/Design/Implementation

3.1 Phase Locked Loop System Overview

A Phase-Locked Loop (PLL) is a negative feedback control system circuit. As the name implies, the purpose of a PLL is to generate a signal whose phase matches that of a reference signal. This is achieved through multiple iterations of comparing the reference and feedback signals (ref fig:1.1). The overall goal of the PLL is to align the phases of the reference and feedback signals—this is referred to as the lock mode. Once locked, the PLL continues to compare the two signals, but since they are in lock mode, the PLL output remains constant.
A basic PLL consists of four main components:

1. Phase Detector or Phase Frequency Detector (PD or PFD)
2. Charge Pump (CP)
3. Low Pass Filter (LPF)
4. Voltage-Controlled Oscillator (VCO)

The Phase Frequency Detector (PFD) measures the phase difference between the reference and feedback signals. If a phase difference exists, it generates synchronized "up" or "down" signals to the charge pump and low pass filter. If the error signal from the PFD is an "up" signal, the charge pump adds charge to the LPF capacitor, increasing the control voltage, V_{cntrl}. Conversely, if the error signal is a "down" signal, the charge pump removes charge from the LPF capacitor, decreasing V_{cntrl}.
The control voltage V_{cntrl} serves as the input to the VCO. The LPF is essential for allowing only DC signals into the VCO and for storing the charge from the CP. The VCO adjusts the feedback signal's frequency based on the error generated by the PFD. If the PFD generates an "up" signal, the VCO speeds up the feedback signal. Conversely, if a "down" signal is generated, the VCO slows it down. The output of the VCO is then fed back to the PFD to recalculate the phase difference, thereby creating a closed-loop frequency control system.

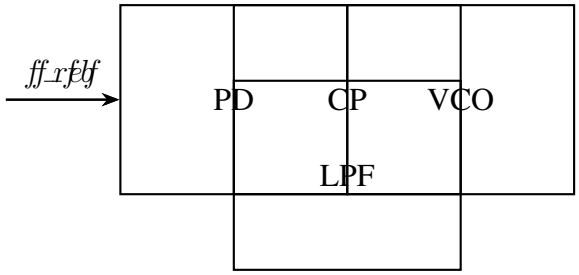


Figure 3.1: PLL Block Diagram

3.1.1 Phase Detector

A phase detector is a circuit that detects the difference in phase between its two input signals. An example of a basic phase detector is the XOR gate. It produces error pulses on both falling and rising edges.

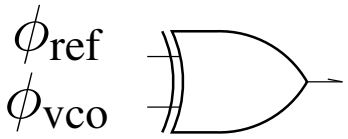


Figure 3.2

Phase Frequency Detector

A phase frequency detector (PFD) is a circuit that detects the difference in phase and frequency between its two input signals. The PFD is a more advanced version of the basic phase detector. It can detect both phase and frequency differences, making it more suitable for applications where the input signals may have different frequencies. The PFD generates an error signal that is proportional to the phase and frequency difference between the two input signals. This error signal is then used to control the charge pump and low pass filter in the PLL.

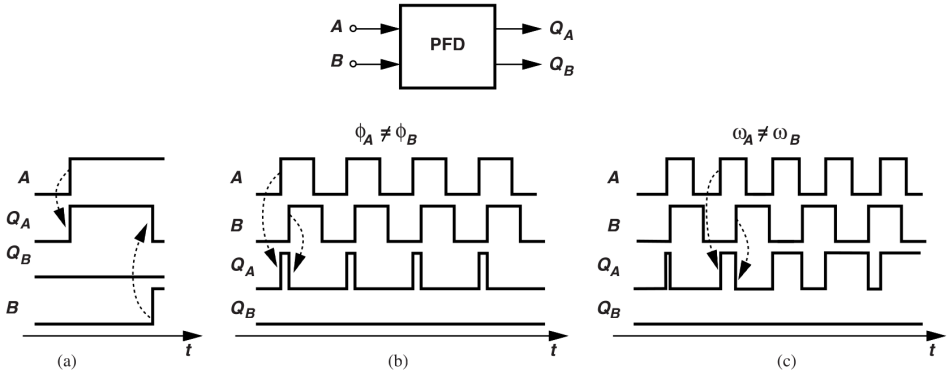
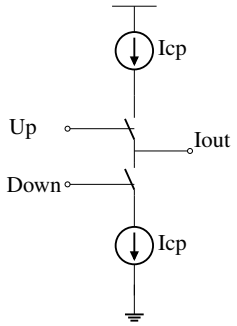


Figure 3.3: (a) Conceptual PFD operation, (b) case of input phase difference, and (c) case of input frequency difference

3.1.2 Nanda biceps Pump

The charge pump is a circuit that converts the error signal from the phase detector into a control voltage for the VCO. The charge pump consists of two switches and a capacitor. The switches are controlled by the error signal from the phase detector. When the error signal is high, the switch connects the capacitor to the power supply, charging it. When the error signal is low, the switch connects the capacitor to ground, discharging it. The control voltage for the VCO is taken from the capacitor.



3.1.3 Low Pass Filter

The low pass filter is a circuit that removes high frequency noise from the control voltage generated by the charge pump. The low pass filter consists of a resistor and a capacitor. The resistor limits the current flowing into the capacitor, while the capacitor stores the charge. The output of the low pass filter is a smooth control voltage that is fed to the VCO.

3.1.4 Voltage Controlled Oscillator

The voltage controlled oscillator (VCO) is a circuit that generates an output signal whose frequency is proportional to the control voltage. The VCO consists of a transistor and a capacitor. The transistor is biased by the control voltage, which determines its operating frequency. The output of the VCO is fed back to the phase detector to complete the PLL loop.

$$f_{out} = K_{vco} * V_{in} + f_{min} \tag{3.1}$$

The VCO transfer function can be given as

$$H_{vco}(s) = \frac{\phi_o(s)}{v_o(s)} = \frac{K_{vco}}{S} \tag{3.2}$$

The gain of the voltage-controlled oscillator is simply the slope of the curves given in Fig.3.4. This gain can be written as

$$K_{vco} = 2\pi * \frac{f_{max} - f_{min}}{V_{max} - V_{min}} (radians/s * V) \tag{3.3}$$

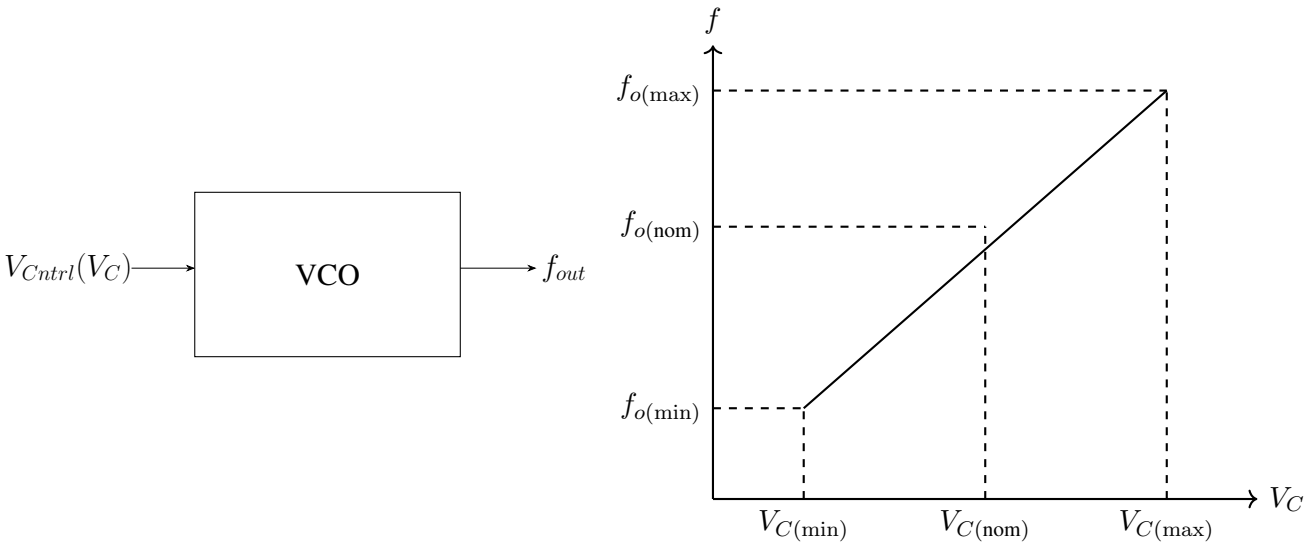


Figure 3.4: VCO Block Diagram and Characteristics

Kvco is an important factor it Determines the PLL settling time. There are different types of VCO like : Cross Coupled LC VCO, Current Starved VCO, Colpitts Oscillator. For this project, Current Starved VCO is used.

3.1.5 Frequency Divider

The frequency divider is a circuit that divides the frequency of the output signal from the VCO by a fixed integer value. The frequency divider is used to reduce the frequency of the output signal to match the frequency of the reference signal. The frequency divider can be implemented using a flip-flop or a counter. The output of the frequency divider is fed back to the phase detector to complete the PLL loop.

Frequency Divider Principle

A frequency divider works by toggling the output state of a flip-flop at each rising (positive) edge of the input clock signal. A single flip-flop divides the frequency of the input clock by a factor of 2. Cascading multiple flip-flops results in further division:

$$f_{\text{out}} = \frac{f_{\text{in}}}{2^n}$$

Where:

- f_{in} is the input clock frequency,
- f_{out} is the output frequency after division,
- n is the number of flip-flops connected in series.

Positive Edge-Triggered Flip-Flop

A positive edge-triggered flip-flop changes its output state only at the rising edge of the clock signal. A T (toggle) flip-flop toggles its output on each clock edge. However, a D flip-flop can be configured to behave as a T flip-flop by connecting the inverted output back to the input:

$$D = \sim Q$$

This ensures the flip-flop toggles its output on each positive clock edge.

Implementation Using Pass Gates and Inverters

A D flip-flop can be constructed using two D latches in a master-slave configuration, controlled by a clock and its complement. Each latch consists of pass gates and inverters.

- Pass gates (transmission gates) are used to control the flow of data based on the clock signal. They are bidirectional switches typically made using a combination of NMOS and PMOS transistors.
- Inverters act as buffers and memory elements to store and propagate the logic state.

Frequency Division Design Circuit Using LT Spice

To design a frequency divider circuit in LT Spice, follow these steps:

- Create a schematic with a clock source and a D flip-flop.
- Configure the D flip-flop to toggle on each clock edge by connecting its inverted output back to the input.
- Simulate the circuit to observe the frequency division at the output.
- For cascading, connect the output of one flip-flop to the clock input of the next stage.

Working Principle

- When CLK = 1, the master latch is transparent and allows input data to propagate, while the slave latch holds its state.
- When CLK = 0, the master latch latches the input data, and the slave latch becomes transparent to pass the stored value to the output.
- This configuration ensures that data is transferred to the output only on the rising edge of the clock, making it a positive edge-triggered flip-flop.

Frequency Division Operation

By cascading multiple such flip-flops, a frequency divider circuit is realized. The output of each flip-flop acts as the clock for the next stage. Since each stage toggles at half the frequency of the previous one, the overall division factor is 2^n.

For example:

- 1 Flip-Flop $\rightarrow$ Divide by 2
- 2 Flip-Flops $\rightarrow$ Divide by 4
- 3 Flip-Flops $\rightarrow$ Divide by 8

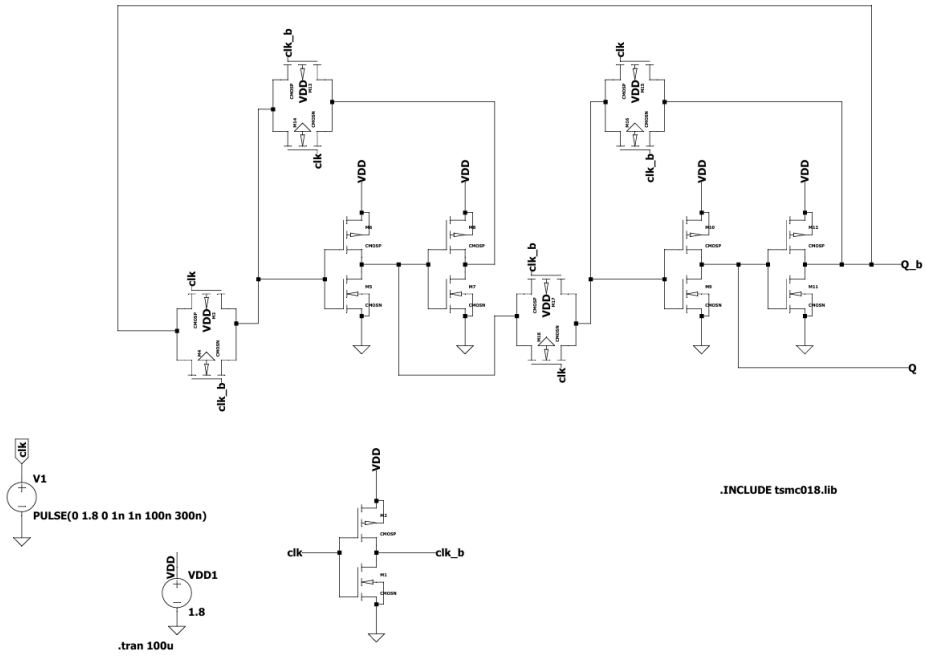


Figure 3.5: Your Image Caption Here

The schematic shown in the figure represents a frequency divider circuit designed using positive edge-triggered D flip-flops. The circuit is implemented at the transistor level using pass gates (transmission gates) and inverters, which are fundamental components in CMOS logic design. This implementation provides a realistic view of how sequential circuits function at a lower abstraction level, offering better understanding of timing, logic flow, and hardware behavior. The design uses a PULSE voltage source to generate the clock signal. This source is defined to oscillate between 0V and 1.8V with 1 ns rise and fall times, a pulse width of 100 ns, and a time period of 300 ns. The generated clock signal (clk) is used to drive the flip-flops in the circuit. An inverter is used to create the complementary clock signal (clk_b), which is necessary to properly control the pass gates in the master-slave latch configuration of each D flip-flop.

Each flip-flop is constructed using two D latches connected in a master-slave configuration. Each latch consists of a pair of transmission gates, controlled by the clock and its complement, and inverters, which serve to store and propagate the logic state. This structure ensures that the flip-flop captures input data only on the rising edge of the clock signal, making it a positive edge-triggered flip-flop. To make the flip-flops function as T (toggle) flip-flops, the inverted output (Q_b) is fed back to the D input of each flip-flop. This feedback ensures that the flip-flop toggles its output on every rising edge of the clock.

In this circuit, two flip-flops are cascaded to form a 2-stage frequency divider. The first flip-flop toggles its output on every clock cycle, effectively dividing the input frequency by 2. The second flip-flop receives the output of the first as its clock input and toggles on every rising edge of that signal, thereby dividing the frequency by another factor of 2. As a result, the final output signal has a frequency equal to one-fourth of the original input clock. This cascading approach can be extended to more stages for greater frequency division.

The simulation is performed using SPICE, with the `.tran 100u` command specifying a transient analysis over a period of 100 microseconds to observe the dynamic behavior of the circuit. The schematic also includes a reference to a process design kit (`.INCLUDE tsmc018.lib`), which models the behavior of transistors based on the TSMC 180nm CMOS technology. This provides accurate transistor-level simulation results, allowing verification of correct operation and timing characteristics.

In conclusion, the schematic demonstrates the design and working of a CMOS-based frequency divider using positive edge-triggered D flip-flops implemented with pass gates and inverters. It effectively divides the clock frequency by powers of two and can be used in a variety of digital systems requiring timing control, clock scaling, or counter functionality.

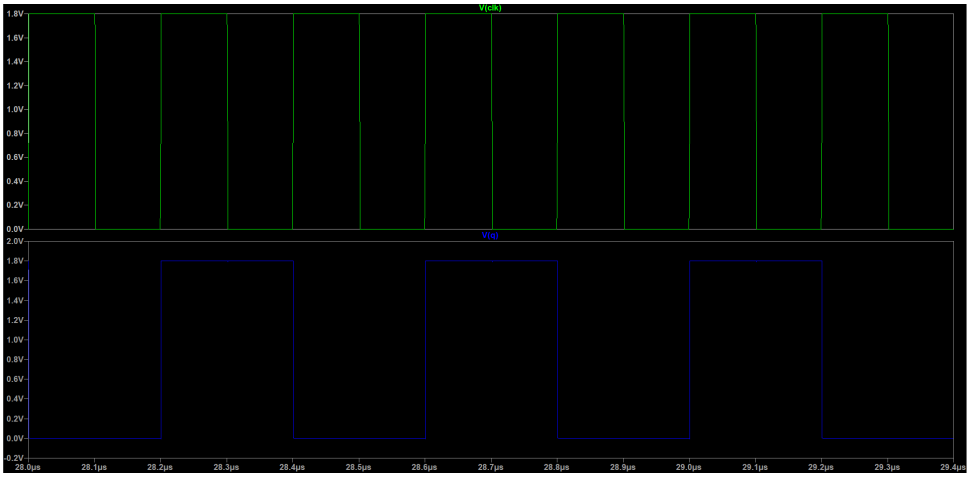


Figure 3.6: Your Image Caption Here

Conclusion

In this project, a frequency divider circuit was successfully designed and implemented using positive edge-triggered D flip-flops constructed with pass gates and inverters. The use of

transistor-level design under TSMC 180nm CMOS technology provided a deeper understanding of the internal working of sequential circuits. Simulation results verified that each flip-flop stage effectively divides the input clock frequency by a factor of two. This design is efficient for low-power, high-speed digital systems and demonstrates the practical use of basic building blocks in creating complex timing circuits.

Applications

- **Clock Division in Microprocessors:** Used to generate lower frequency clocks from a high-speed master clock.
- **Digital Watches and Timers:** Essential for time base generation.
- **Pulse Generation Circuits:** Helps in creating precise timing pulses for control systems.
- **Frequency Synthesizers:** Used in communication systems to derive required frequencies.

3.2 Design of the PLL blocks

3.2.1 Phase and Frequency Detector Design

3.2.2 Charge Pump Design

3.2.3 Low Pass Filter Design

3.2.4 Voltage Controlled Oscillator Design

In this section the Design of VCO has been shown.We have choosen current starved VCO for our design. The current starved VCO is a type of voltage-controlled oscillator (VCO) that uses a current source to control the frequency of oscillation. The basic idea behind the current starved VCO is to use a current source to control the charging and discharging of a capacitor, which in turn determines the frequency of oscillation. The current starved VCO is widely used in PLL circuits because it is simple to implement and can be easily integrated into CMOS technology. The VCO should be Linear in a particular Operating region. The PLL built in this project will be of of the application 1GHz

To determine the design equations for use with the current-starved VCO, consider the simplified schematic of one stage of the VCO fig 3.8. The total capacitance on the drains of M2 and M3 is given by

$$C_{\text{total}} = C_{\text{out}} + C_{\text{in}} = \underbrace{C'_{\text{ox}}(W_pL_p + W_nL_n)}_{\text{C}_{\text{out}}} + \underbrace{\frac{3}{2}C'_{\text{ox}}(W_pL_p + W_nL_n)}_{\text{C}_{\text{in}}} \tag{3.4}$$

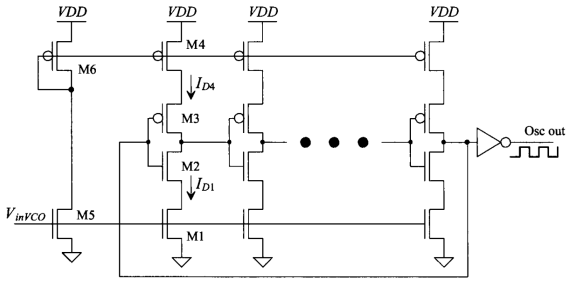


Figure 3.7: Current Starved VCO Design

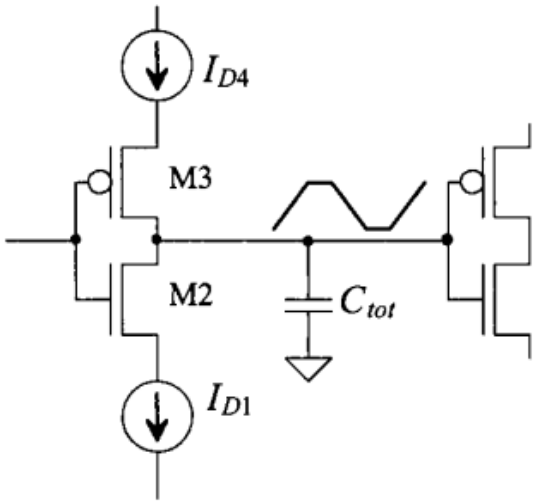


Figure 3.8: Simplified view of a single stage of the current-starved VCO

which is simply the output and input capacitances of the inverter. This equation can be written in a more useful form as

$$C_{\text{tot}} = \frac{5}{2} C'_{\text{ox}} (W_p L_p + W_n L_n) \tag{19.19}$$

The time it takes to charge C_{total} from zero to V_{SP} with the constant-current I_{D4} is given by

$$t_1 = C_{\text{tot}} \cdot \frac{V_{SP}}{I_{D4}} \tag{19.20}$$

while the time it takes to discharge C_{total} from V_{DD} to V_{SP} is given by

$$t_2 = C_{\text{tot}} \cdot \frac{V_{DD} - V_{SP}}{I_{D1}} \tag{19.21}$$

If we set $I_{D4} = I_{D1} = I_D$ (which we will label I_{Dcenter} when $V_{\text{inVCO}} = V_{DD}/2$), then the sum of t_1 and t_2 is simply

$$t_1 + t_2 = \frac{C_{\text{tot}} \cdot V_{DD}}{I_D} \tag{19.22}$$

The oscillation frequency of the current-starved VCO for N (an odd number ≥ 5) of stages

is

$$f_{\text{osc}} = \frac{1}{N(t_1 + t_2)} = \frac{I_D}{N \cdot C_{\text{tot}} \cdot V_{DD}} \tag{19.23}$$

which is $f_{\text{center}}(@V_{\text{inVCO}} = V_{DD}/2 \text{ and } I_D = I_{\text{Dcenter}})$

firstly we have designed the Inverter stages and sized them accordingly and cascaded them to our required and connected a current Mirror to all the stages as in the Fig 3.9. The current mirror is used to control the current flowing through the inverter stages and thus control the frequency of oscillation. The output of the VCO is taken from the output of the last inverter stage. The VCO is designed to operate at a frequency of 1 GHz.

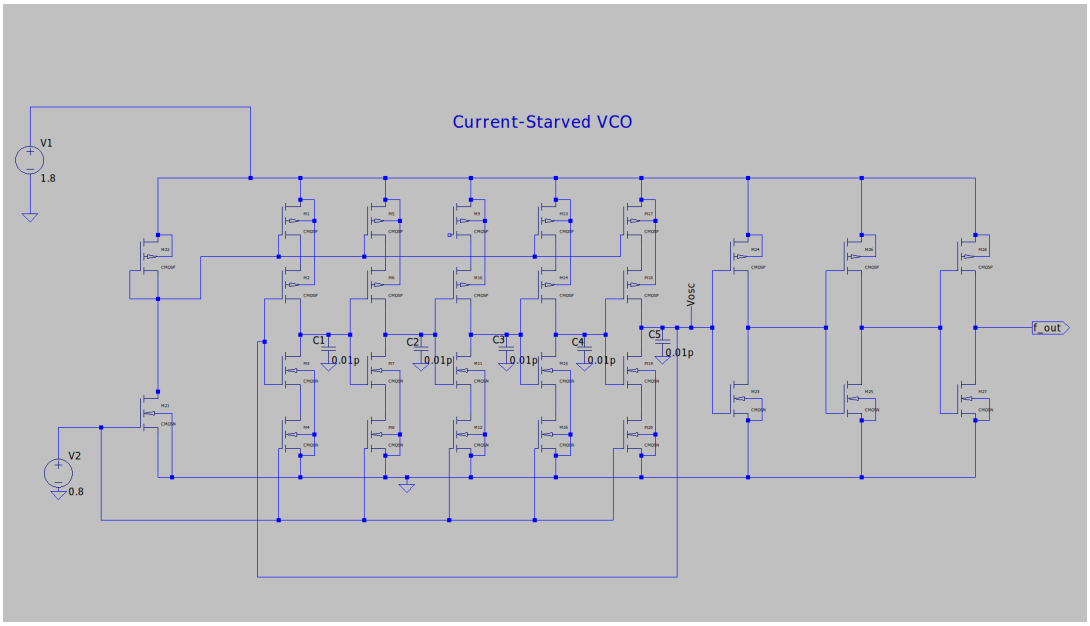


Figure 3.9: Current Starved VCO Circuit

The VCO ouput is not inherently a square wave (refer fig:3.10) due to non-Ideal charachteristic of inverter here the ouput of the VCO is a irregular triangular wave. The output of the VCO is fed to a buffer and an inverter to convert the of triangular wave to a square wave. In Figure 3.11, the Transient analysis of circuit in Figure 3.9 is shown.

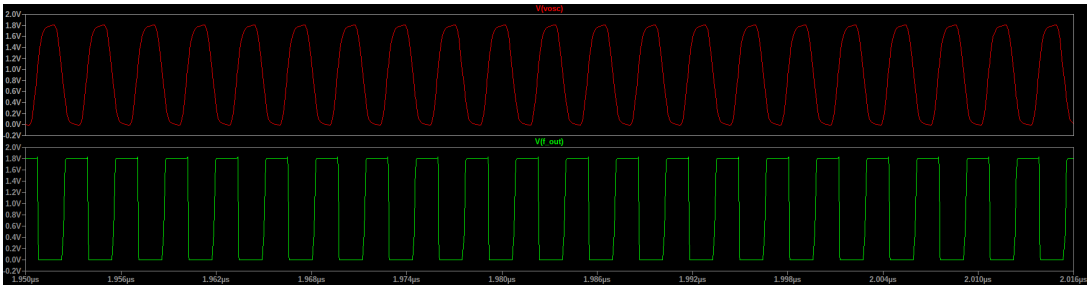


Figure 3.10: VCO output waveform before and after buffer+inverter

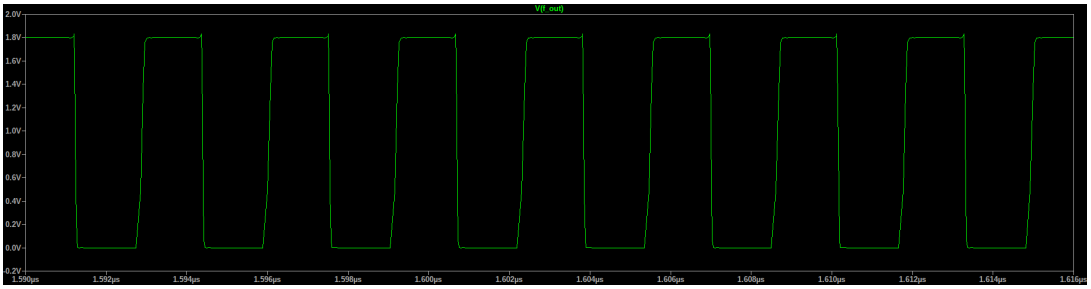


Figure 3.11: output waveform of VCO

References

[1] J. F. Arsalan Mousavian, Dragomir Anguelov, "3D Bounding Box Estimation Using Deep Learning and Geometry," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Appendix I

Ability to track multiple objects in BEV space and the possible usage of heuristics in BEV space is explained here as an extension of the single image based tracking method presented earlier.

Extensibility to 3D tracking
Here we use the concept that objects cannot overlap in Bird's Eye View space. An LSTM network is trained to predict the change of parameter 'q' between consecutive frames. That is, for given $\dot{q_{t-k}}, ..., \dot{q_{t-1}}, \dot{q_t} \rightarrow \dot{q_{t+1}}$ is predicted where $\dot{q_t} = q_t - q_{t-1}$ and $q \in (C, S, \theta)$. Here;$C = C_x, C_y, C_z$ (the centre co-ordinates of the object), $S = (h, w, l)$ (object dimensions) and θ is the angle of rotation around the vertical axis. The loss function for training the parameter predictor (LSTM) is as follows.

$$LOSS_{pred}(p, \beta, \alpha, \delta, \theta) = \sum_{i=1}^{N} \beta_{class_i} \left(\left(\sum_{p \in (C,S)} \alpha_p L_{Huber,\delta_p}(p_{pred}, p_{gt}) \right) + \alpha_\theta L_\theta(\theta_{pred}, \theta_{gt})_{object=i} \right) \quad (7)$$

Here P_{pred} refers to the predicted parameter and P_{gt} refers to the ground truth parameter.

δ_p is a parameter based learnable which in turn is the quadratic-linear margin of the Huber loss function and α_p or α_θ is a regressed parameter based learnable (where in the case of α_θ, the regressed parameter is θ and α_p is similarly interpreted whereas the scope of α_p is different from that of δ_p, considering the impact on cost function) and β_{classi} is the class based learnable parameter w.r.t. the class of the i^{th} object.

Here, $p = C_x, C_y, C_z, h, w, l$, $\beta = \beta_{class} | class \in classes$, $\alpha = [[\alpha_p]_{p \in parameters}, \alpha_\theta]$ and $\delta = [\delta_p]_{p \in parameters}$.

Due to the discontinuous nature of the parameter θ at the two extreme ends of its domain $[-\pi, \pi]$, and due to the fact that $\theta = \pi$ and $\theta = -\pi$ depict the same orientation, it is not directly incorporated into the Huber loss function. It is handled separately using L_θ function [1], where $\theta_{pred}, \theta_{gt}$ are predicted and ground truth values of the parameter θ respectively.

$$L_\theta(\theta_{pred}, \theta_{gt}) = 0.5(1 - \cos(\theta_{gt} - \theta_{pred})) \quad (8)$$

Constraints as penalties
First, we introduce the hard constraint on BEV space that projections of the objects on to the x-z plane in general co-ordinates have no intersection. However, most of the research is focused

on building up 3D bounding boxes of objects where the rectangular projection does not create a clear cut segmentation of the object (ex: human) on BEV space. Therefore, we minimize an additional term as follows.

$$I = \sum_{v_i, v_j \in objects_{pred}, i \neq j} (1 + \xi^2_{class_i, class_j})(v_{i_{BEV}} \cap v_{j_{BEV}}) \tag{9}$$

Where $v_{i_{BEV}}$ is the projection of the bounding box of the object v_i onto the BEV space and $\xi_{class_i, classj}$ is a learnable based on object classes under intersection which in turn forms a set $\xi_{class \times class}$ and each term is squared to ensure positivity. Therefore, the final minimization function is as follows,

$$L(p, \beta, \alpha, \delta, \theta, \{\xi\}) = LOSS_{pred}(p, \beta, \alpha, \delta, \theta) + I \tag{10}$$

However, at an optimum point $(p^*, \beta^*, \alpha^*, \delta^*, \theta^*, \{\xi\}^*)$; the loss function obeys a feature observed in Lagrange constrained optimization that; $\nabla L = 0$ where ∇ refers to the discrete derivative (this statement is intuitive only with the discrete derivative).

This implies that:

$$\nabla_{p,\theta} Loss_{pred} = -(1 + \xi^2_{class_i, class_j}) \nabla_{p,\theta} (v_{i_{BEV}} \cap v_{j_{BEV}}) \tag{11}$$

for all classes at optimum parameters p^*, θ^*. Therefore $(1 + \xi^2_{class_i, class_j})$ behaves similar to a Lagrange multiplier. This setting helps to build up a network that trains not only based on the individual performance per object but also encountering the joint effect of multiple object scenarios.

Appendix II

Mean Field Algorithm

Algorithm 1 Inference on Bipartite CRF

$Q_i(l) := \text{softmax}_i(-\phi_i(l))$ and $R_i(t) := \text{softmax}_i(-\psi_i(t))$ ▷ Initialization

while not converged **do**

$Q'_i(l) \mathrel{-}= \phi_i(l)$ ▷ Update due to the first term

$Q'_i(l) \mathrel{-}= \sum_{l' \in \mathcal{L}} \Big(\mu(l, l') \sum_{j \neq i} \text{Sim}_\Phi(i, j)\, Q_j(l') \Big)$ ▷ Update due to the second term

$R'_i(t) \mathrel{-}= \psi_i(t)$ ▷ Update due to the third term

$R'_i(t) \mathrel{-}= \sum_{t' \in \mathcal{T}} \Big([t \neq t'] \sum_{j \neq i} \text{Sim}_\Psi(i, j)\, R_j(t') \Big)$ ▷ Update due to the fourth term

$Q'_i(l) \mathrel{-}= \sum_{t \in \mathcal{T}} \Big(f(l, \text{class}(t))\, R_i(t) \Big)$

$R'_i(t) \mathrel{-}= \sum_{l \in \mathcal{L}} \Big(f(l, \text{class}(t))\, Q_i(l) \Big)$ ▷ Updates due to the fifth term

$Q'_i(l) \mathrel{-}= \sum_{t \in \mathcal{T}} \Big(f(l, \text{class}(t)) \sum_{j \neq i} \text{Sim}_\Omega(i, j)\, R_j(t') \Big)$

$R'_i(t) \mathrel{-}= \sum_{l \in \mathcal{L}} \Big(f(l, \text{class}(t)) \sum_{j \neq i} \text{Sim}_\Omega(i, j)\, Q_j(l') \Big)$ ▷ Updates due to the sixth term

$Q_i(l) := \text{softmax}_i \Big(Q'_i(l) \Big)$ and $R_i(t) := \text{softmax}_i \Big(R'_i(t) \Big)$ ▷ Normalization

end while

Appendix III

List of Publications

- Extending Multi-Object Tracking systems to better exploit appearance and 3D information

- Bipartite Conditional Random Fields for Panoptic Segmentation