



Layout Design and Simulation of a VCO-Based ADC

BY:

JULIA SZEMIOT

HTAY AUNG WIN

YUE (JERRY) LI

Submitted: March 24, 2017

Advisor:

PROFESSOR STEPHEN BITAR

A Major Qualifying Project Report submitted to the faculty of WORCESTER POLYTECHNIC INSTITUTE in partial fulfillment of the requirements for the degree of Bachelor of Science in Electrical and Computer Engineering.

Abstract

This paper is about simulating and creating layout for voltage-controlled oscillator (VCO) based Analog to Digital Converter (ADC) using 65nm technology. Various configurations of VCOs are simulated as well as different approaches were made to implement the ADC. Challenges encountered throughout the simulation process are documented as well. The final product of the project includes layouts and the schematics of the components which comprise the ADC.

Table of Contents

Abstract.....	2
Acknowledgements.....	8
Introduction.....	9
Literature Review	10
Analog-to-Digital Converters	10
Characterization of ADCs	10
Different Kinds of ADCs	11
Low Speed High Resolution	11
Medium Speed Moderate Resolution	12
High Speed Low Resolution	12
Changing ADC Architectures.....	14
Voltage Controlled Oscillators.....	14
Voltage Controlled Oscillator-based ADC.....	21
Multiplexer (MUX).....	24
Counter Design	25
FPGA (Field- Programmable Gate Array)	25
Methodology	27
Proposed Overall Design.....	27
Linear Shift Register	26
Single-ended Ring Oscillators.....	29
VCO with 1V Supply: 3 Stage	32
VCO with 1V Supply: 5 Stage	36
MUX	39
Analog Transmission Gate Resistance Measurement	40
Buffer	50
Inverter Buffer	50
Tapered Buffer	50

Simulation Results	53
FFT Frequency Analysis With Matlab	59
Block Diagram of Altered Design	62
Lower Frequency Simulations	62
Digital MUX and Counter	67
Layout and Final Design	70
Selection of Ring Oscillators.....	70
Layout of Ring Oscillators.....	70
Layout of MUX and Counter	74
Recommendations for Future Projects	76
References	77
Appendices	78
APPENDIX A: The Young Designer's How-To Guide (WPI Edition).....	78
Chapter 1: Cadence is an Awful Software	79
Chapter 2: Ask for help early, and often	96
Chapter 3: Everything is documented somewhere.....	97
Chapter 4: Analog Layout Fundamentals	98
Chapter 5: Digital Layout and Synopsys.....	123
APPENDIX B: Comprehensive Simulation Results.....	127
MOSFET V-I characteristics	127
VCO with 1.8V Supply: 3 Stage	130
VCO with 1.8V Supply: 5 Stage	134
VCO with 3.3V Supply: 3 Stage	136
VCO with 3.3V Supply: 5 Stage	139
Pseudo-differential VCO	141
Appendix C: Verilog Code.....	144
Appendix D: Performance Analysis on Digital circuitry.....	148
Appendix E: Matlab Code	154

Table of Figures

Figure 1: General ADC Functional Blocks [1].....	10
Figure 2: 3-stage ring oscillator VCO schematic	16
Figure 3: Fully Differential Ring Oscillator Single Stage	17
Figure 4: Fully Differential Ring Oscillator with Four Stages	17
Figure 5: Pseudo-differential 3 Stage VCO	18
Figure 6: Current-starved Pseudo-differential 3 Stage VCO.....	19
Figure 7: VDD-to-frequency curve of a discrete 3-stage ring oscillator VCO using CD4007	19
Figure 8: 3-stage current starved ring oscillator schematic	20
Figure 9: Graphical representation of a VCO-based ADC operation.....	21
Figure 10: Multiphase VCO-based ADC architecture	22
Figure 11: Quantization of phase in VCO-based ADC.....	23
Figure 12: voltage-to-frequency relationship of 3-stage current starved VCO.....	24
Figure 13: 2-1 MUX.....	25
Figure 14: Block diagram of the proposed overall design	27
Figure 15: Ring Oscillator with Indicated PMOS Switch.....	29
Figure 16: Ring Oscillator with Indicated Inverter PMOS.....	30
Figure 17: Ring Oscillator Schematic with Indication of Scaled PMOS.....	31
Figure 18: V-f characteristic of 3 Stage Current Starved VCO	32
Figure 19: V-f characteristic of Current Starved VCO with PMOS switch (small).....	33
Figure 20: V-f characteristic of Current Starved VCO with PMOS switch (medium)	33
Figure 21: V-f characteristic of Current Starved VCO with PMOS switch (large)	34
Figure 22: V-f characteristic of Current Starved VCO with PMOS Switch Sweeping Inverter PMOS Width.....	35
Figure 23: V-f characteristic of Current Starved VCO with PMOS Switch Sweeping PMOS Width.....	35
Figure 24: V-f characteristic of 5 Stage Current Starved VCO	36
Figure 25: V-f characteristic of Current Starved VCO with PMOS switch.....	36
Figure 26: V-f characteristic of Current Starved VCO with PMOS Switch Sweeping Inverter PMOS Width.....	37
Figure 27: V-f characteristic of Current Starved VCO with PMOS Switch Sweeping Current Starving MOSFET Width.....	37
Figure 28: V-f characteristic of Current Starved VCO with PMOS Switch Sweeping PMOS Width.....	38
Figure 29: A transmission gate	39
Figure 30: 4-to-1 Transmission Gate	40
Figure 31: AC response of MUX with PMOS and NMOS width = 400nm	41
Figure 32: AC response of MUX with PMOS and NMOS width = 4 μ m	42
Figure 33: AC Response of MUX with PMOS and NMOS width = 40 μ m	43

Figure 34: AC response of MUX with PMOS and NMOS width = 400 μ m	44
Figure 35: Step Response with PMOS and NMOS width = 400nm	45
Figure 36: Step Response with PMOS and NMOS width = 4 μ m	46
Figure 37: Step Response with PMOS and NMOS width = 40 μ m	47
Figure 38: measuring the delta V and delta I to find the resistance (red = single TG, green = two TG in series, purple = current source)	48
Figure 39: measuring the delta V and delta I to find the resistance of single TG.....	49
Figure 40: Example of an inverter buffer	50
Figure 41: The output of the multiplexer failing to fully oscillate at 1GHz	51
Figure 42: The output of the multiplexer failing to fully oscillate worsening at 10GHz.....	51
Figure 43: 9 stage 3.3V supply connected to a 4 stage tapered buffer.....	52
Figure 44: Waveforms of the 9 stage 3.3V after the VCO and after the tapered buffer	53
Figure 45: Waveforms of the 7 stage 3.3V supply after the VCO and after the tapered buffer.....	54
Figure 46: Waveforms of the 5 stage 3.3V after the VCO and after the tapered buffer	54
Figure 47: 9 stage VCO with 4 stage buffer.....	55
Figure 48: 9 stage VCO with 5 stage buffer.....	55
Figure 49: 7 stage VCO with 4 stage buffer.....	56
Figure 50: 7 stage VCO with 5 stage buffer.....	56
Figure 51: Waveforms of the 11 stage 3.3V after the VCO and after the tapered buffer	57
Figure 52: Waveforms of the 7 stage 1V after the VCO and after the tapered buffer	57
Figure 53: Waveforms of the 9 stage 1V after the VCO and after the tapered buffer	58
Figure 54: Waveforms of the 11 stage 1V after the VCO and after the tapered buffer	58
Figure 55: Waveforms of the 13 stage 1V after the VCO and after the tapered buffer	59
Figure 56: 3.3V, 11stage sampled at 5GHz (without buffer on left, with buffer on right)	59
Figure 57: 1V, 11stage sampled at 5GHz (without buffer on left, with buffer on right)	60
Figure 58: 1V, 11stage sampled at 10 GHz (without buffer on left, with buffer on right)	61
Figure 59: Block diagram of the revised design	62
Figure 60: Determining Optimal Frequency Characteristics (Part I).....	63
Figure 61: Determining Optimal Frequency Characteristics (Part II)	64
Figure 62: Determining Optimal Frequency Characteristics (Part III).....	65
Figure 63: Determining Optimal Frequency Characteristics (Part IV).....	66
Figure 64: Schematic view of synthesized digital MUX.....	68
Figure 65: Schematic view of synthesized ripple counter.....	68
Figure 66: Layout of VCO1.....	71
Figure 67: Layout of VCO2.....	72
Figure 68: Layout of VCO3.....	72
Figure 69: Layout of VCO4.....	73



Figure 70: Layout of MUX and Counter74

Figure 71: No DRC errors75

List of Tables

Table 1: Dimensions of tapered buffer components53

Table 2: Timing measurement (slack)69

Table 3: Selected VCO Characteristics and Validation.....70



Acknowledgements

The Analog IC design major qualifying project team would like to express sincere appreciation to Prof. Bitar for giving this wonderful opportunity to work under him and Prof. McNeill for his guidance and motivation to advance with the project.

The team would also like to thank Sulin Li and Jianping Gong, PhD students at WPI NECAMSID lab, for helping and discussing with us about the project during the progress. We are also very grateful to Ed Burnham for all of his help with permissions and the complex process of Cadence installation.

Introduction

The goal of this project is to design an integrated circuit (IC) for analog to digital conversion using an available process development kit (PDK) in Cadence Virtuoso. We designed a voltage control oscillator using 65nm technology to convert an analog input voltage to a frequency and map it to a digital output.

We started by simulating the V-I characteristics of the MOSFETs and building simple ring oscillators. From there we simulated the current-starved topology, which would limit the amount of current available, and allow the analog input to be independent of the supply voltage. We optimized the performance for full voltage swing, frequency of oscillation able to be processed by the counter, and input voltage range. We used these criteria to layout our VCO designs, confirming they functioned properly by adhering to the Design Rule Check (DRC), Layout-Versus-Schematic (LVS), and parasitic extraction (PEX) and for ease of layout simulation.

These VCOs could be selected using a linear shift register which would activate the MUX, and each VCO would achieve greater resolution, higher voltage range or low power dissipation. The MUX output would be fed to an adequately fast counter. The desired performance would be a 12 bit analog-to-digital converter (ADC) with 10 ENOB (effective number of bits) based on comparison to prior art, and ideally we would have sent out our design for fabrication and finish off-chip calibration using an FPGA with a look-up table and test the ADC with an evaluation board feeding an analog signal directly from a digital-to-analog converter (DAC). Unfortunately, due to time constraints and unforeseen delays, we were unable to achieve this, but we did complete the layout design such that it is ready for fabrication.

Literature Review

Analog-to-Digital Converters

The analog-to-digital converter (ADC) is used to transform the signals commonly found in nature, which are analog, so that they can be digitally processed. Because the input signal is continuous and the output is discrete, the conversion requires sampling of the input and mapping to an approximate value. The ADC generally requires the four functional blocks illustrated in Figure 1.

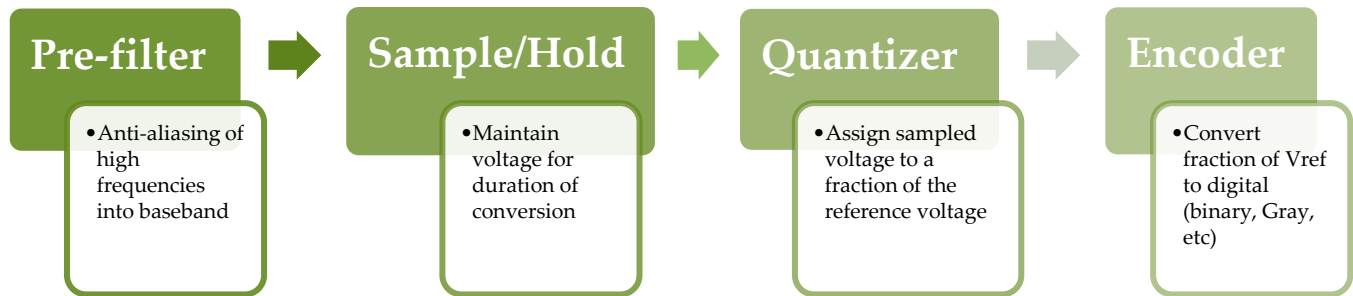


Figure 1: General ADC Functional Blocks [1]

The ADC usually contains a pre-filter, which is responsible for removing distortion and noise from the input signal which may introduce error in identifying the signal frequency. This is also known as anti-aliasing. The input is then fed into the sampling block, where a sample and hold circuit would typically be used to maintain the voltage while the conversion takes place. The output waveform of the sample and hold functional block would be quantized, or assigned a fractional value based on the reference voltage. Afterwards the fraction of the reference voltage would be converted to some kind of digital value, most commonly binary [4].

Characterization of ADCs

The analog-to-digital converter has certain parameters used to distinguish variations in specifications and performance.

There are a few key design specifications. The Full Scale Range (FSR) indicates the entire spectrum of possible inputs that can be detected by a particular ADC, which is typically a voltage [4]. The dynamic range (DR) of the ADC is defined as the FSR divided by the least significant bit (LSB), and must be $6N$ for an N -bit converter [4]. The resolution, in bits, corresponds to the smallest analog change an ADC can perceive. The output sample rate, measured in samples per second (SPS), indicates the speed of the ADC, a very important design parameter in most applications. The power dissipation and operating temperature are also important considerations. Finally, the output data format must correspond to the digital format of the remaining system, and can be binary, Gray, thermometer, or two's complement [1].

• • •

Every ADC structure has distinct performance that is characterized by error, nonlinearity, and noise. The offset error indicates the difference between the offset and the ideal value, which is 0 in unipolar ADCs and -1MSB (most significant bit) in offset bipolar [1]. Gain error is the difference between the actual characteristic and the ideal, which is proportional to input voltage [1]. The integral nonlinearity (INL) is typically given as a percentage of FSR, and indicates the deviation between the ADC characteristic and a straight line. INL can be either determined using the linear approximation at the end point, which is typically more useful, or using the best straight line, which has a smaller linearity error. Differential nonlinearity (DNL) is a measure of how linearly the codes transition, where ideally 1LSB of analog input corresponds to 1LSB of digital output, but phenomena like missing code transitions and irregular step sizes increase nonlinearity[1]. Another important parameter is the signal to noise ratio (SNR)[1,4]. According to Allen, the effective number of bits (ENOB) is defined as

$$ENOB = \frac{SNR_{actual}-1.76}{6.02} \quad (1)$$

where SNR_{actual} is the ADC's actual signal to noise ratio (SNR) [4].

Using the effective number of bits (SNR_{bits}), the sampling frequency (f_{smp}) and power dissipated by the ADC (P_{diss}), the following equation was created by R.H. Walden

$$F = \frac{2^{SNR_{bits}} f_{smp}}{P_{diss}} \quad (2)$$

where F is a figure of merit (FOM) meant to quantitatively evaluate an ADC's effectiveness [10]. The theoretical maximum figure of merit calculated by Walden when applying superconducting conditions is 6.61013 [10].

Different Kinds of ADCs

The different kinds of ADCs are characterized by the tradeoff of speed and resolution. Thus the ADC topology is determined based on the application. The simplest ADC is realized using a comparator, where one comparator corresponds to 1-bit, as it can either have a high value or a low value [1].

Low Speed High Resolution

Integrating (Serial)

The only low speed high resolution topology is the integrating (serial) ADC, with conversion rates below 100Hz and ENOB exceeding 12 bits. This ADC requires a ramp generator, an interval counter, one comparator, an AND gate, and a counter [4].

The integrating architecture can be implemented as a simple single slope. This architecture may be affected by ramp generator error, but it is also unipolar and has a long conversion time up to $2^N T$ where

• • •

N is the number of bits and T is the clock period. This architecture can also be implemented as dual slope, which eliminates the dependence on accuracy and linearity of the slope [4].

Medium Speed Moderate Resolution

Successive Approximation

Inspired by the mathematical method of successive approximation, formerly known as feedback subtraction, this ADC determines the value of each bit by rounding [4]. A successive approximation converter with N bit resolution requires N cycles, or steps, for conversion [4,1]. It contains a comparator, a digital-to-analog converter, and digital control logic containing a successive approximation register (SAR) [1,4].

Pipeline Algorithmic

The pipeline algorithmic converter is capable of N bit resolution when it contains N stages and N comparators, making it as complex as the serial ADC but with shorter conversion time of N clock cycles [4]. This architecture works by taking an input at each stage, doubling it, and adding or subtracting the reference voltage depending on the previous stage. During the following clock cycle, the output of the i th stage is compared to ground before outputting the i th bit such that it converges at the input voltage [4]. Performance is limited by gain and offset errors [4].

Iterative Algorithmic

The iterative algorithmic converter consists of a S-H circuit, an amplifier with gain of 2, a comparator, and a reference subtraction circuit [4]. The iteration process continues once triggered by a switch until the desired number of bits is obtained, such that the MSB is first [4].

High Speed Low Resolution

Flash ADCs

The fastest ADC architecture, called the flash (or parallel) ADC, is comprised of resistors and comparators. It is the fastest because the input signal is applied at every comparator simultaneously [1]. Flash converters contain 2^{N-1} comparators and 2^N resistors for N bit resolution, so the primary concerns are high power dissipation and large chip area [1,4]. The comparator output is affected by noise, crosstalk, BW limitations, and meta-stability [4,1]. Using Gray code rather than binary reduces metastable state errors, as each code varies from the other by one digit rather than complete changes in binary (such as from 011 to 100).

There are some constraints to using this topology. For resolutions of at least 6 bits, the offset voltage must be corrected [4]. The input must be sampled without jitter, so it is useful to incorporate a sample and hold (S-H) circuit at the input and clock the comparators at the same time. Additionally, there is a large input capacitance which limits the input BW, so using S-H circuitry also helps to eliminate the

• • •

BW restriction [4]. In order to ensure identical delay for all the comparators, the supply voltage should be at least twice the reference voltage [4]. A phenomenon known as flashback is triggered by rapid transitions at the comparator's input, which can be reduced by using a pre-amplifier or buffer [4].

IC flash ADCs typically exhibit 8 bit resolution (up to 10 bit), 1GHz maximum sampling rate, and more than 300MHz bandwidth (BW) [1]. Bipolar conversion in 1 clock cycle can also be achieved by using both positive and negative reference voltages [4].

Interpolating ADCs

Interpolating converters share the architecture of flash ADCs for the most part, except they have fewer preamplifiers and dissipate less power [1]. Thus the input capacitance decreases and does not depend as much on input signal level and distortion [4,1]. Biasing all of the comparators to the same threshold voltage and including a S-H circuit at the input improve linearity [4,1].

Performance is limited by two major factors. Interpolating ADCs have problems with sinusoidal inputs due to phase error, which can contribute to nonlinearity [4]. Additionally, the accuracy of the interpolation limits the resolution [4].

Folding ADCs

Folding converters consist of two stages, a coarse quantization stage with 2^{N_1} values and a fine quantization stage where all the 2^{N_1} values are mapped to a single subrange with 2^{N_2} values. This architecture significantly reduces the number of comparators and requires only 1 clock cycle for the conversion [4]. Hence the conversion speed remains the same as the area and power dissipation are reduced.

The folding architecture also has some limitations. It requires a S-H in order for its output BW to exceed the BW of the analog input multiplied by the number of subranges 2^{N_1} . This architecture is discontinuous at input voltages that are at quarter intervals of the reference voltage, but using a bipolar reference voltage eliminates the discontinuity [4]. Additionally, the ADC does not work well below a quarter of the reference voltage, so it is beneficial to incorporate multiple folding structures with different phases [4].

Multiple-Bit Pipeline

The multiple-bit pipeline architecture is beneficial in that it contains multiple bits per stage, but it requires several clock cycles for conversion [4]. It works by subtracting the stage output from the input, generating a residue voltage which is then amplified prior to the next stage. Unfortunately, using residue amplification limits the input BW, but dividing the reference voltage rather than amplifying the signal creates a subrange without hindering the BW [4]. This topology balances the tradeoff of area and speed, and can also easily incorporate digital error correction by adding an extra bit on each stage to verify accuracy [4].

Time-Interleaved

The time-interleaved architecture is comprised of many successive approximation ADCs in parallel allowing for N bit resolution in one clock cycle, which is much faster than individual successive approximation converters [4]. This topology requires matching offsets, gains, and delays for accurate conversion.

Sigma-Delta ($\Sigma\Delta$)

The sigma-delta converter architecture, originally called delta-sigma, contains a delta modulator and a decimation filter, which requires the greatest amount of area and power [1,4]. It benefits from oversampling, which removes the quantization noise from the BW and increases the SNR at low frequencies [1]. The mathematical concept behind the naming convention is taking the integral of a difference [1].

This topology is preferred for high resolution, low frequency applications, and surpasses the performance of all the other architectures for it is inherently monotonic and works well with lower cost CMOS processes, low bandwidth, and low power applications [1,4]. It does not require precise component matching nor much antialiasing at the input, for the small bandwidth functions as an antialiasing filter [4]. In addition, sampling is performed inherently by the same circuits which are responsible for quantization [4].

Changing ADC Architectures

Over time, the semiconductor process technology has been scaled down to nanometers as modern electronic device have kept pushing the limitations for better performance in smaller packages. Smaller technology provides dramatic improvements in size, yield, speed and power consumption. One significant advantage of CMOS nanometer process is better digital power efficiency as lower power supply can be utilized for logic functions. However many existing ADC architecture in the market have not been able to wholly utilize the process scaling due to reduced voltage headroom and reduced analog gain [11]. This drives toward upsurge in popularity of time-based architecture ADCs such as voltage-controlled oscillator-based ADCs which take advantages of higher speed performance and lower power consumption.

Voltage Controlled Oscillators

The voltage controlled oscillator (VCO) is an electronic circuit which produces the output frequency signal directly proportional to input voltage, basically a voltage to frequency converter. The VCO is used in a wide range of circuitry, from low frequency applications such as analog music synthesizers to high frequency applications such as phase-locked loops for radio receivers.

LC VCO

The LC VOs are made of inductors and capacitors, with an amplifier. This architecture is preferred for sinusoidal wave output and the disadvantage being the die area when integrated due to inductors in the circuit [12].

Source coupled VCO

These VCOs are made by using transistors connected back to back. This technique gives high frequency VCOs and is also used for producing rectangular and saw tooth waveforms. Such VCOs can also be designed to consume less power than the current-starved VCOs. The major drawback of this architecture is the need for a capacitor, which is hard to implement in a single-poly pure digital process without using parasitic [12].

Ring Oscillators VCO

The ring oscillator is a member of the class of time delay oscillators, so to understand its operation, first an explanation on gate delay is necessary. In real world, MOSFETs when used as in a circuit such as inverters, for example cannot be turned on instantaneously. The gate capacitance must be charged before current can flow between the source and drain, thus inducing a time delay between each stage. The time delay of each inverter contributes to the delay of signal around the ring of inverters, hence the name ring oscillator. Since the time delay is basically the period of the VCO output waveform, the oscillation frequency is given by the following equation:

$$f_{osc} = \frac{1}{T} = \frac{1}{2 \times t_d \times N} \quad (3)$$

where f_{osc} is the oscillation frequency, N is the number of stages (inverters), t_d = the total propagation time delay [5]. The factor 2 results due to the fact that one complete cycle requires low to high and high to low transitions. $2 \times t_d$ can also be described as $t_r + t_f$ (rise time + fall time). Since the propagation time delay is the time it takes for the current to charge the capacitance of each stage, the equation can be rewritten as

$$f_{osc} = \frac{1}{2 \times N \times V_{ctrl} \times C_{total}} \quad (4)$$

where V_{ctrl} is the control voltage (supply voltage V_{DD} in normal ring oscillator), the total sum of output capacitor of the first stage and input of capacitor of the second stage C_{total} , and the bias current I_d [5].

From above two equations, the oscillation frequency of a CMOS ring oscillator can be tuned by varying the time delay of each stages or the number of stages. Depending on the application requirements, such as low power consumption, speed or less die area, etc., these parameters can be traded off to fit the required specifications. For certain cases, such as VCO-based ADCs, the least number of stages is preferred in most cases to obtain highest VCO output frequency for better ADC performance.

• • •

Single Ended Configuration

The most basic CMOS ring oscillator consists of odd number of CMOS inverters connected in a closed loop with positive feedback. The output of the last stage is fed back into the input of the first stage. The oscillating output waveform is produced since there is no stable operation point exists because of odd number of inversions in ring oscillator. It is required that every stage of ring oscillators should add 180 degree phase shift in order to sustain oscillation.

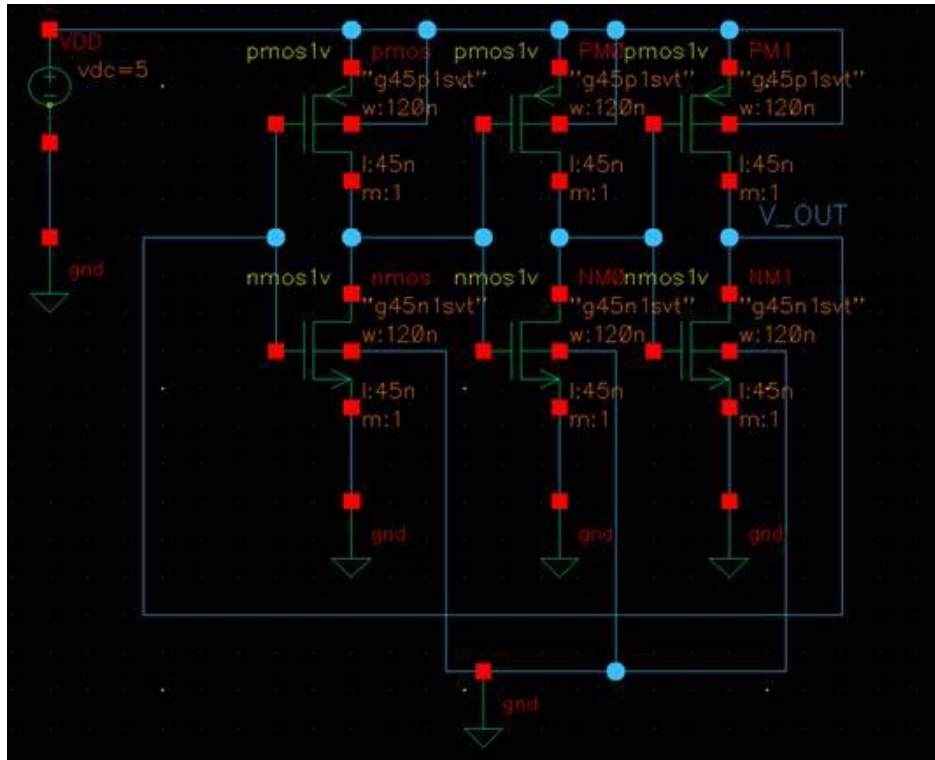


Figure 2: 3-stage ring oscillator VCO schematic

The single-ended configuration is easily affected by variations in supply voltage and requires an odd number of stages [15].

Fully Differential Configuration

The fully differential VCO configuration requires two inputs of anti-phase signals [14]. It employs common mode rejection to reduce jitter and improve signal to noise ratio (SNR) [15, 14]. A single stage of a fully differential ring oscillator is depicted in Figure 3.

VCO-BASED ADC

...

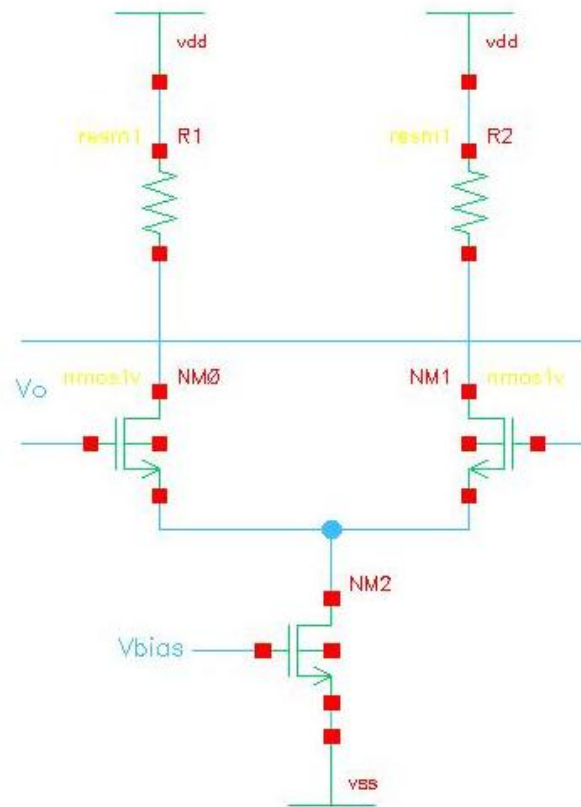


Figure 3: Fully Differential Ring Oscillator Single Stage

This topology makes the ring oscillator immune to changes in the supply voltage, and it also enables the designer to employ an arbitrary number of stages as both phases of the signal are available [15]. An even number of stages can be realized by inverting the stage connections, as illustrated in Figure 4.

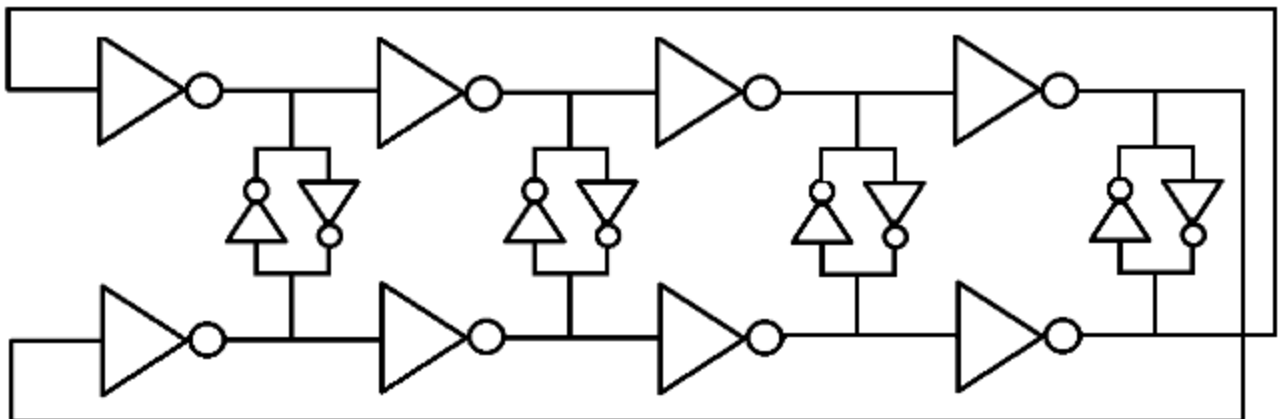


Figure 4: Fully Differential Ring Oscillator with Four Stages

However, using a true differential limits the voltage swing [15].

• • •

Pseudo Differential Configuration

Similarly to the fully differential, the pseudo differential ring oscillator processes the difference between two signals. However, only one of the pseudo differential signals is oscillating, the other is a reference to ground. This is accomplished by using two inverter rings comprised of the same number of stages, and maintained at 180° phase lag through cross-inverter taps [13]. Note the 3 ring configuration in Figure 5.

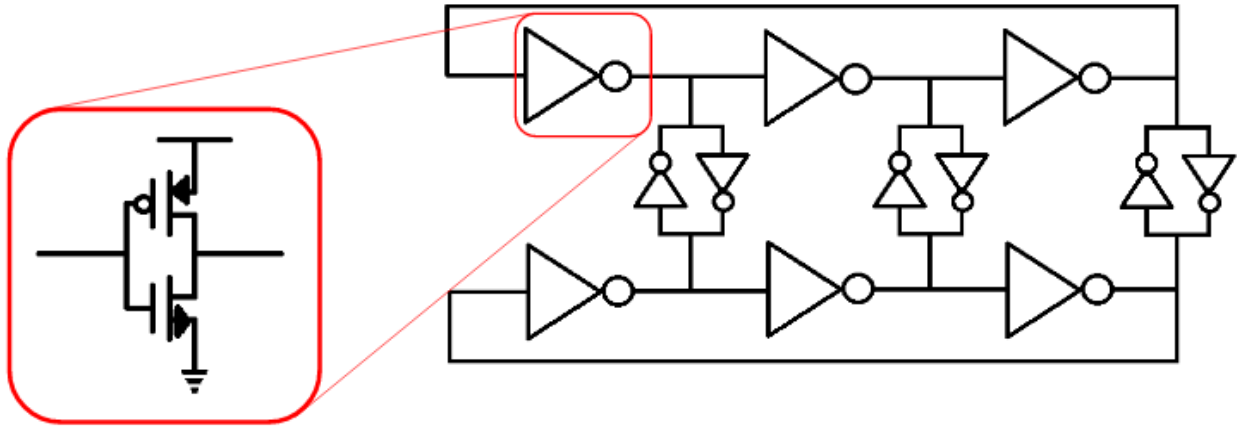


Figure 5: Pseudo-differential 3 Stage VCO

There are a few advantages to using pseudo differential. First of all, it only requires one input unlike the fully differential [14]. The design is therefore simpler and the output DC levels saturate at the supply rails, thereby reducing jitter [13, 15]. In addition, this configuration improves the SNR through something similar to common mode rejection, but not as well as fully differential [14, 15]. There is one caveat: if the signals are perfectly inverted, the even harmonics disappear, but if there are any differences between both signals, they will immediately decrease quality of the signal [13].

Current-starved Ring Oscillators

Other than changing the number of stages, the oscillation frequency can be controlled by modulating the time delay t_d . From equation (2), the time delay can be modified by either changing the current I_D or the capacitance C_{total} . In normal ring oscillators VCO (Figure 2) the amount of current is controlled by the supply voltage (V_{DD}). However, the V_{DD} varies and this produces a large variation in output frequency.

The current-starved variation of the differential configuration is also possible, and depicted in Figure 6. Only the two main inverter chains are current-starved, whereas the cross-coupling are not.

...

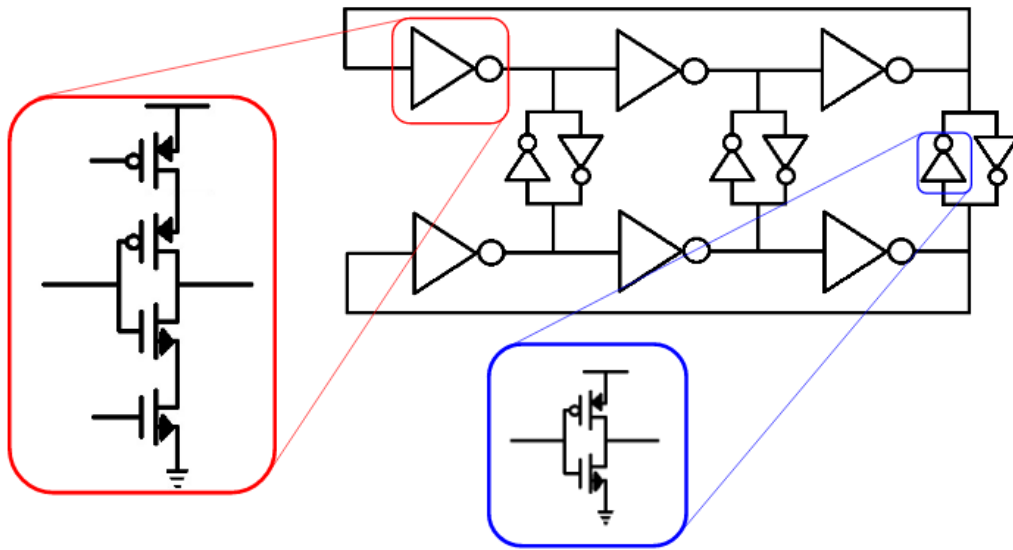


Figure 6: Current-starved Pseudo-differential 3 Stage VCO

Figure 7 below represents the output frequency vs supply voltage plot of a discrete three-stage ring oscillator VCO. Assuming the V_{DD} is 9V with $\pm 10\%$ variation, the difference output frequency will be approximately almost 2 GHz which is a very large variation. The same behavior is observed in smaller CMOS technology as well. Thus the output frequency is not stable when it is dependent on V_{DD} .

Discrete Ring Oscillator V-f Characteristic

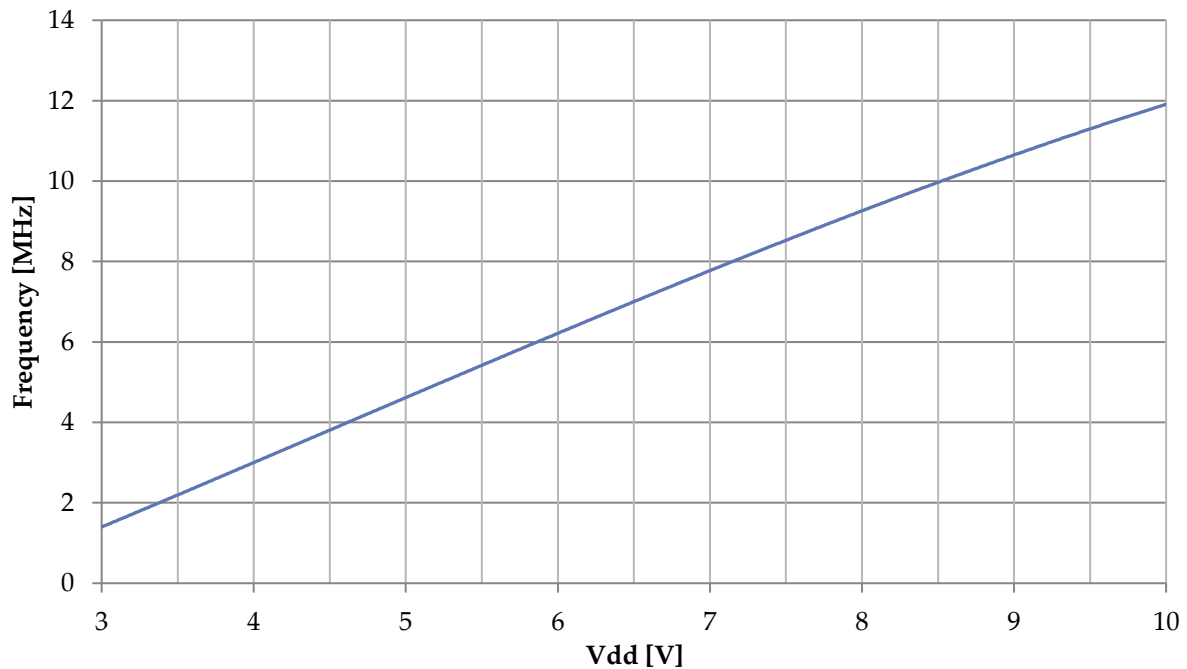


Figure 7: VDD-to-frequency curve of a discrete 3-stage ring oscillator VCO using CD4007

...

This problem can be dissolved by supplying current to each inverter instead of directly using V_{DD} . Such type of circuit is called a current starved ring VCO (Figure 8).

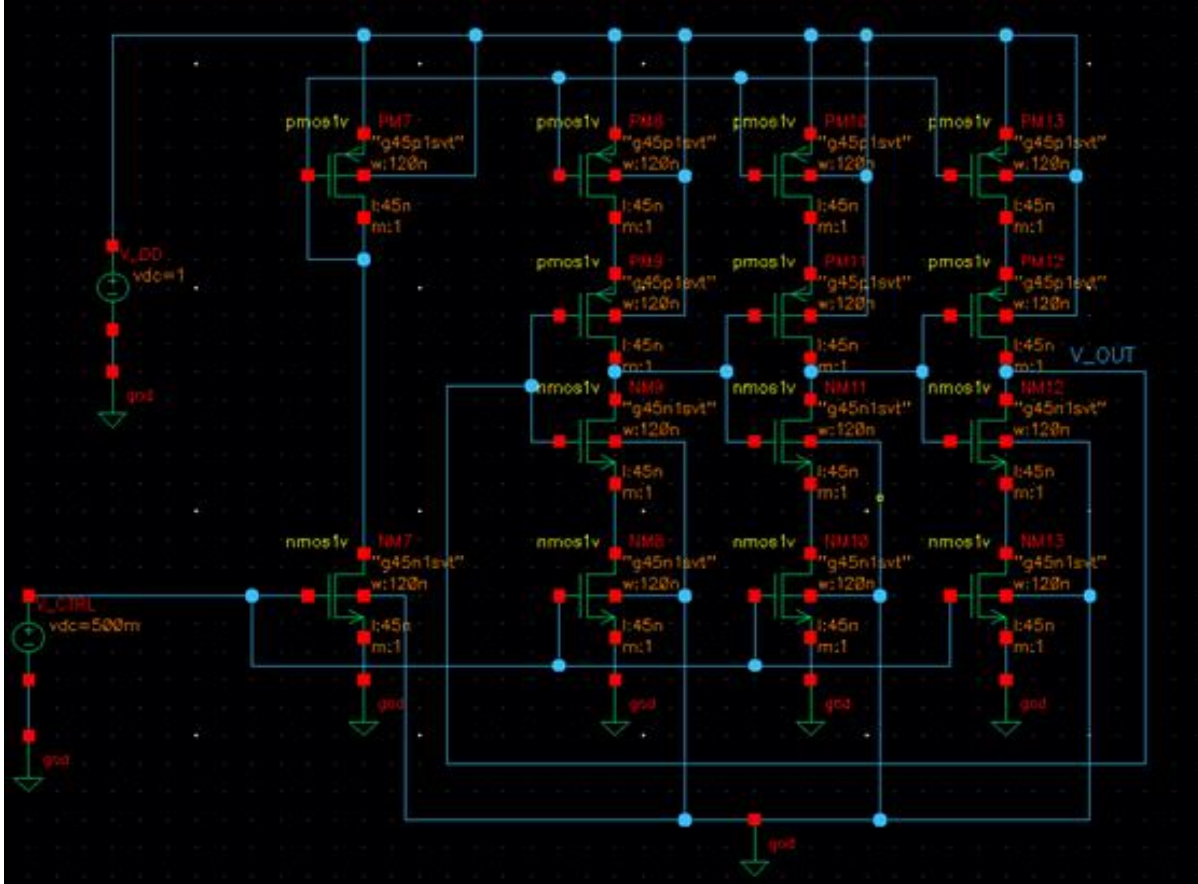


Figure 8: 3-stage current starved ring oscillator schematic

The schematic of a three-stage current-starved VCO is shown in Figure 8. Its operation is similar to the ring oscillator. The middle pair of MOSFETs (M9, M11, M12 NMOS and PMOS) in each stages act as inverters while the outer MOSFETs (M8, M10, M13 pairs) operate as current limiters. The latter limits the current available to the inverters, thus starving the inverters and hence the name current-starved ring oscillator. The control voltage basically modulates the turn on resistances of the pull-down and pull-up limiters through a current mirror. These variable resistances restrict the amount of current available to charge or discharge the load capacitances of each stages.

Another method to control the oscillation frequency is by varying the total capacitance C_{total} . This parameter can be varied by adding one or more voltage dependent capacitors called varactor diodes. A varactor diode is a reverse-bias PN junction which has a capacitance of:

• • •

$$C_{var} = \frac{C_0}{\left(1 - \frac{V}{\Phi_B}\right)^m} \quad (5)$$

where the V is the control voltage, C_0 is zero bias capacitance, Φ_B is the built-in voltage of the junction and m is a constant with value around 0.3-0.5 [11]. Since the adding varactor capacitance increases the total capacitance C_{total} , this action directly affects the output frequency of the VCO itself, resulting in lower frequency range. Additionally, the C_{var} changing non-linearly with the control voltage can become tricky to deal with and might further degrade the performance of VCO.

Voltage Controlled Oscillator-based ADC

A VCO-based ADC is a time-based architecture which generates a frequency signal corresponding to analog input which is then quantized to digital output by other digital circuitry. Figure 9 below is the graphical representation of a simple VCO-based ADC operation [11]. The analog input voltage V_{IN} acts as control voltage to the VCO which generates an oscillating waveform whose frequency is proportional to control voltage. The output of the VCO is then quantized to digital output by the digital counter which counts the number of rising or falling edges (or both) in a given period of time. At the end of every clock cycle, the output of the counter is sampled and then reset to zero. As the analog input varies, the output frequency of VCO changes accordingly thus digital output corresponding to analog input voltage is generated.

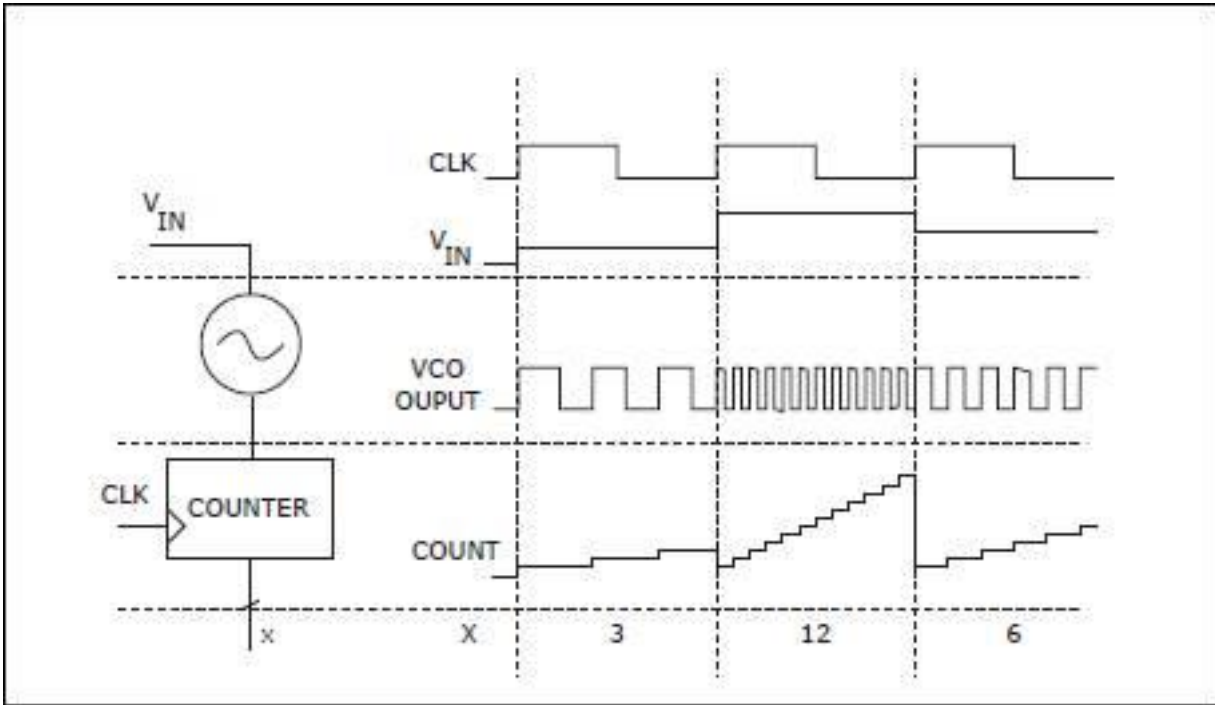


Figure 9: Graphical representation of a VCO-based ADC operation

• • •

An alternate approach of VCO-based ADC design is to have counters connected at each output of the inverters. This topology which uses multiple phase outputs of the VCO together as shown in Figure 10 increases the resolution of the VCO-based ADC [11]. The output frequency of each stage is given by:

$$f_{VCO} = K_{VCO}V_{IN}$$

where K_{VCO} is the slope of voltage-to-frequency characteristics of VCO and V_{IN} is the analog input voltage. The digital output of the ADC is the sum of the number of phase transitions n in a fixed period T_{CONV} of every stages which can be obtained by the following equation:

$$n = 2Nf_{VCO}T_{CONV} = (2NK_{VCO}T_{CONV})V_{IN}$$

where N is the number of stages of the ring VCO. From the above equation, the number of output bits is directly proportional to the analog input voltage, which is the preferred characteristic of an ADC. Moreover, for an N -phase VCO-based ADC, the resolution is $2N$ times higher than that of single-phase inverter. On the other hand, multi-phase inverter demands more chip area, more power consumption and increase in complexity [11].

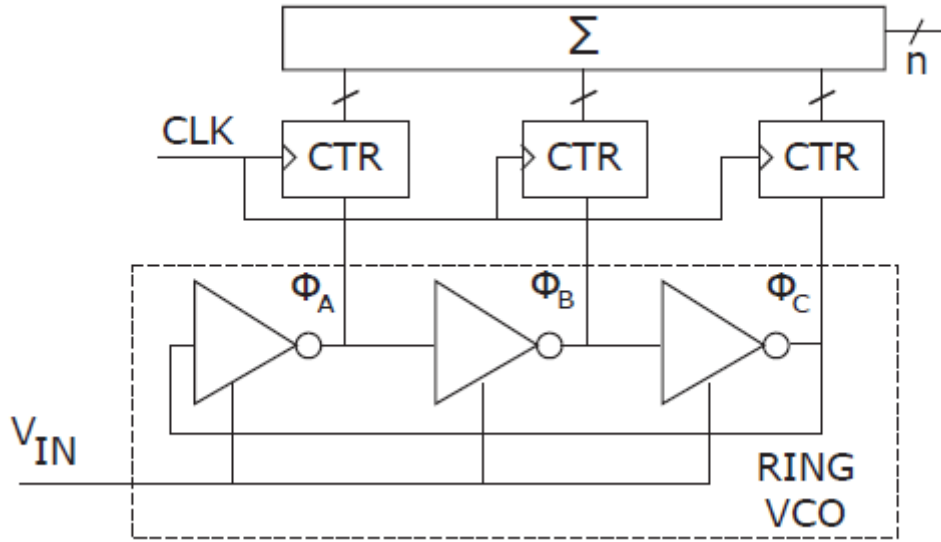


Figure 10: Multiphase VCO-based ADC architecture

Advantages and Disadvantages of a VCO-based ADC

There are certain features of VCO-based ADCs making it more superior to other ADC architectures. One of the most important properties is that the output of the ring VCO is digital in nature, and this makes VCO-based ADCs more suitable in utilizing the high speed and low power consumption

• • •

advantages of nanometer CMOS technology. Additionally, the amplitude of VCO output is not necessarily significant in quantization process [11], eliminating the need for additional analog circuitries such as buffers and amplifiers thus further reducing die size and lower power consumption.

Another advantage is that the VCO-based ADC quantization noise is first-order noise-shaped [11]. The phases are quantized by 2π when counting either the rising edge or falling edge of the clock the clock signal and π when both edges are counted. From Figure 11, we can see that there are some residual phase (quantization error $\phi_q[n-1]$) of the previous sampling period which becomes the initial phase $\phi_i[n]$ of the next period. The output of the N-phase VCO-based ADC is calculated by:

$$y[n] = \frac{N}{2\pi} (\phi_x[n] + \phi_q[n-1] - \phi_q[n]) \quad (6)$$

where $\phi_x[n]$ is the VCO phase change due to analog input [11]. The Z-transform of this equation gives:

$$Y(z) = \frac{N}{2\pi} (\phi_x[z] + \phi_q(z)(z^{-1} - 1)) \quad (7)$$

From the above equation, the quantization phase noise $\phi[n]$ “sees” a high pass filter transfer function $(z^{-1} - 1)$ which is a first-order noise-shaped [11].

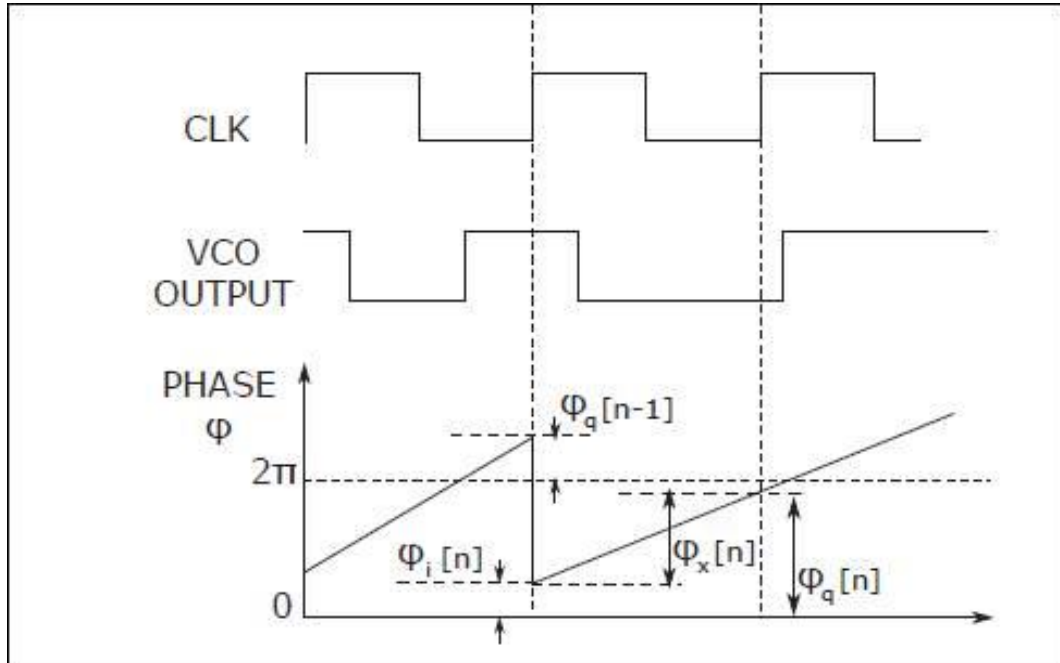


Figure 11: Quantization of phase in VCO-based ADC

With all the favorable advantages of VCO-based ADCs described above, one major drawback of VCO-based ADC is that the voltage-to-frequency curve of the VCO is usually nonlinear which translates directly to non-linearity of the ADC.

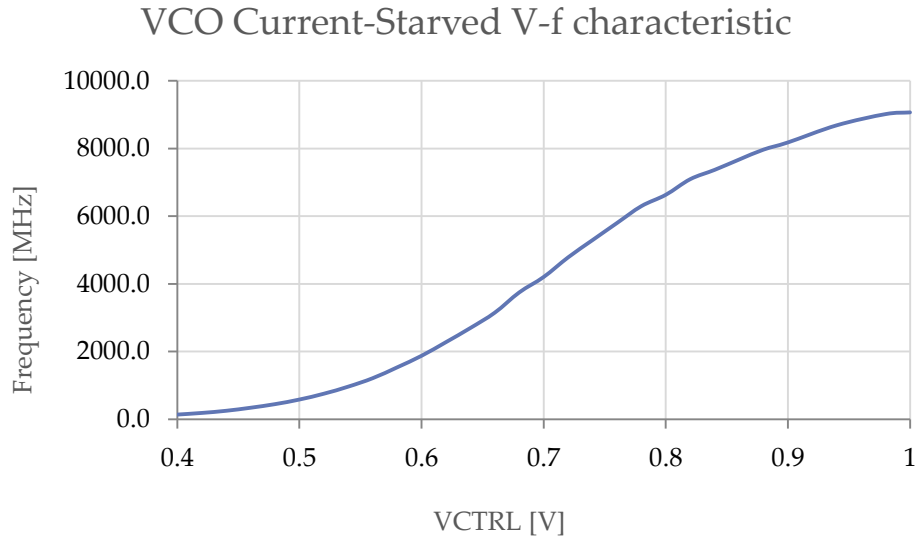


Figure 12: voltage-to-frequency relationship of 3-stage current starved VCO

Figure 12 shows the nonlinear relationship of output frequency of 3-stage current-starved ring VCO to analog input voltage. The slope of this curve is nonlinear with an offset and a varying slope $K_{VCO} = \partial f_{VCO} / \partial V_{IN}$ [11]. Even though the offset can be removed by subtracting the same amount to the digital output code, some sort of calibration is required to correct this non-linear behavior to obtain a SNR better than 40 dB [11] since it has not yet been able to improve the VCO itself.

Why Current-Starved VCO Architecture?

Current-starved VCO is selected as the VCO in VCO-based ADCs which is the goal of this project for several reasons. One of the advantages of current-starved VCO have mentioned above is the stability of the output oscillation frequency. Moreover, this configuration provides a wide output oscillation frequency range over the control voltage. Among different VCO architectures, current-starved ring oscillators provide higher integration ability and more control over the current thus better control over the oscillation frequency. Furthermore, ring oscillator VCO in smaller technology can provide very high output frequency (Figure 8), which is one of the most favorable requirements for VCO-based ADCs.

Multiplexer (MUX)

Multiplexer (MUX) is an electronic component which selects one of several analog or digital input signals and outputs the selected input into a single line. Figure 13 represents a simple 2-to-1 MUX.

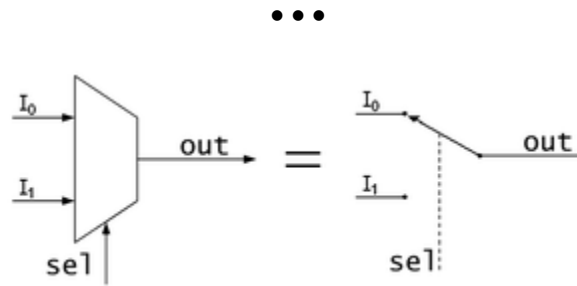


Figure 13: 2-1 MUX

Counter Design

A counter is a circuit that keeps track of how many times an event has occurred relative to a clock signal. In this project, it will be the number of oscillations the VCO output in a fixed period of time. Counters are comprised of a number of flip-flops connected together in some configuration. Counter design generally comes down to 2 types: ripple counters (asynchronous) and synchronous. This project will likely be testing the implementation of both counters.

The ripple counter is a series of flip-flops connect in series with output of one leading directly into the input of the next. The very first flip-flop is connected to an external clock. Hence a “ripple” effect in the bit states will be observed when counting up or down. The propagation delay between flip-flops poses a major problem; if the accumulated delay is long enough, false counts will appear in the output [6].

The synchronous counter has all its flip-flops connected to the same clock signal, thus removing the propagation delay issue by having bits changing states at the same time. AND gates and occasional OR gates are also present in the circuit for controlling the output when counting [7]. As a result of the additional logic gates, synchronous counters have drawbacks in power consumption and size of circuit.

Ripple counters also tend to still be better for high speed applications since only the first flip-flop has to run at the clock signal frequency; each following flip-flop will run at only half the previous frequency. Meanwhile all the flip-flops in a synchronous counter are moving at the same high frequency which means rapid charging and discharging of flip-flop input capacitance. Spikes in supply line and stray capacitive coupling consequently can occur due to usage of a synchronous counter for high speed applications [2].

FPGA (Field- Programmable Gate Array)

The FPGA is an integrated circuit made of a matrix of configurable logic blocks with reprogrammable interconnects unlike a microcontroller which already has preset hardware. The control of a FPGA is in changing the hardware instead of the software running [9]. Interfacing an ADC with a FPGA is actually very common; there exists a variety of different styles. Interfacing can generally be divided into 2 categories, low speed data conversion and high speed data conversion.

• • •

Low speed data conversion usually uses single data rate (SDR) or double data rate (DDR). Single data rate takes 2 clock edges, one for data to transition the transmitter and one to transition the receiver. Double data rate had the data transition completed on every clock edge; the drawback to the faster data conversion is more difficulty with proper sampling timing [8].

High speed data conversion usually uses parallel low voltage differential signaling (LVDS). It uses differential signaling with a P and N wire for each bit instead of a CMOS. The increase in wires from substituting the CMOS decreases power consumption while making routing more difficult. A serializer/deserializer (SERDES) can slow down a high speed converter's output when FPGA logic is much slower than the converter output by being attached in the interface. It slows down converter's output in multiples of $\frac{1}{2}$, $\frac{1}{4}$, or $\frac{1}{8}$ the clock rate [8].

Linear Shift Register

The linear shift register is a shift register capable of storing a parallel load. It is capable of greatly reducing the amount of wires used in the digital part of a circuit since it is made up of very few logic blocks (mostly just D-flip-flops). There are two main types of linear shift registers; they are the shift right only type and the able to shift left and right type [16]. The ability to shift both ways type then further divides into 2 types; the ones with separate control inputs shifting and loading (three inputs) and the ones that do not (two inputs).

Methodology

Proposed Overall Design

This section outlines the procedures taken along the progress of this project. The overview of the system and design approach as well as the brief discussion of individual circuit components are included. Simulations and results are presented afterwards.

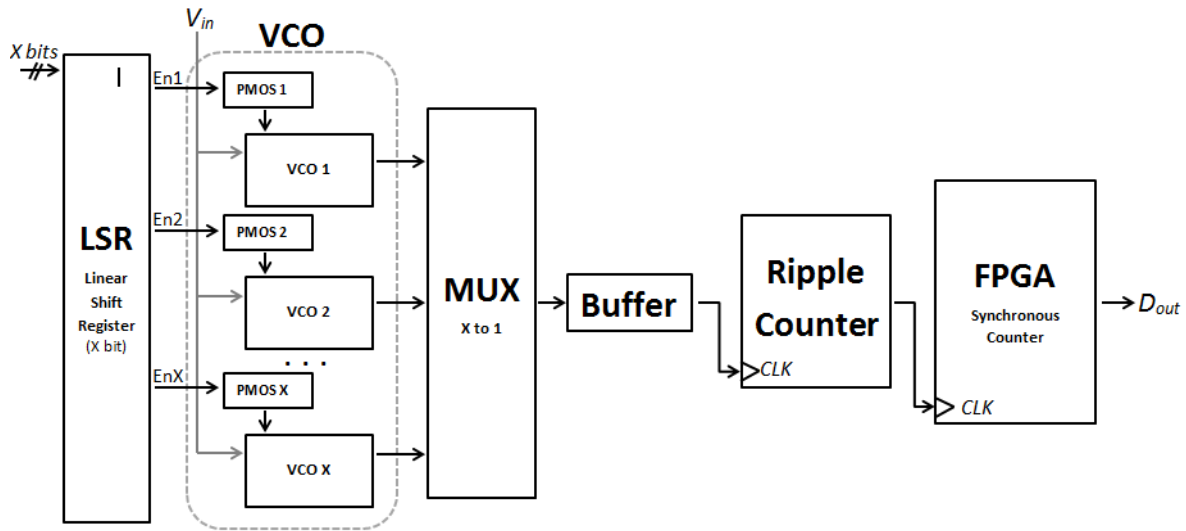


Figure 14: Block diagram of the proposed overall design

Figure 14 depicts the initially proposed design of the ADC. The user will select one of several VCOs by giving inputs to the Linear Shift Register (LSR) which turns on the PMOS switch of the selected VCO. The PMOS switches are connected to the supply rail thus the unselected VCOs will not operate, thus reducing unnecessary power dissipation eradicating the coupling issue between VCOs.

Since the project is to design a research chip, several VCOs of different supply rails and multiple combinations of different number of stages and different widths of individual MOSFETs are included. The number of stages and the width of individual MOSFETs also differ from one VCO to another.

The output of the VCOs are connected to X-to-1 MUX (X being the number of VCOs), which in turn is connected to a buffer. The MUX will select the VCOs via off-chip signals and the buffer is to compensate the degradation in frequency response caused by MUX. Then a ripple counter is used to convert the signal into digital bits which is followed by calibration on FPGA off-chip.

VCO

For this project, we will have multiple current-starved VCOs on the chip, each with different specs such as widths of MOSFETs, number of stages, supply voltages and architectures. By doing this we will explore and compare performances of these VCOs. Additionally, this will provide information for future research on this topic.

PMOS switch

Multiple VCOs on the chip raised a new concern about the ADC performance. Having all of the running at the same time will not only consume power, but also coupling between them will take place which will deteriorate the ADC performance. A solution to this was problem to turn on VCOs one at a time, by using switches to each VCO. The switches were simply big P-type MOSFETs which connects the supply voltage to the ground, thus turning off the VCO when the switch is off. The large width of MOSFETs used as switches is necessary to minimize voltage loss across the switch.

MUX

The use of switches eradicates the power dissipation and coupling problem of having multiple VCOs, however connecting all VCOs to a single counter would give rise to additional problems. Including a counter for each VCO comes at a price of die area, and is probably not the most efficient solution. A better approach would be to use a multiplexer (MUX) to select the VCO and convey the signal to the counter. This approach has its own pros and cons. For pros, it saves significant die area as well as compliments with the uses of switches, as the signal to turn on and off can be used a select input for the MUX. However the MUX can be modeled as a resistor and can form a low-pass filter in conjunction with the total parasitic capacitance of the VCO. This is a critical downside since the VCO output signals have a frequency in GHz domain. The problem was discussed more in the report with detail measurements and different approaches made to solve this problem.

Buffer

Buffers were necessary additions to reduce capacitance that negatively affected the ability of VCOs to oscillate. Stages of them were present before, after and within the MUX to improve the oscillation fidelity going into the counter. Buffers and the simulation results obtained on them are elaborated on in their own section later in the report.

Ripple Counter

Ultimately it is on the counter that converts the actual analog data into digital for processing by outside source. The ripple counter and data collected on it is also explain in more detail in its own section of the report.

Single-ended Ring Oscillators

Rationale

For this MQP project, several measurements were taken, particularly on performance of VCO with different specifications.

Normal VCOs vs CSVCOs

Both standard and current-starved VCO architectures are explored in this MQP. This allows us to observe and understand the functionality of the different architectures and discern their strength and weaknesses.

PMOS switch width

In order to reduce the power consumption, we introduced the PMOS switch. This would enable us to turn the VCOs that are not being used off. Adding the switch affected the output frequency, so we determined the width of the switch that would least affect the output up to an acceptable tolerance, so we would have a reasonable die area. The circuit schematic is presented in Figure 15.

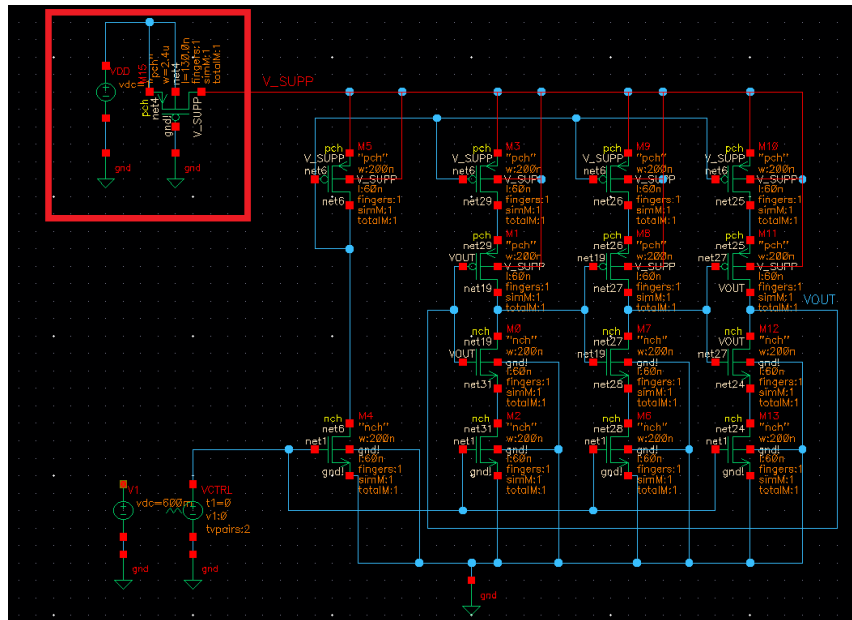


Figure 15: Ring Oscillator with Indicated PMOS Switch

Width increment

For this measurement, the performance of the VCOs are recorded with different width of current-starving and current-sinking MOSFETs. From MOSFET equation, bigger width corresponds to larger current flow thus increasing the frequency of the VCOs. The drawback of bigger width is the die area.

Width of PMOS double the width of NMOS

The widths of the inverter PMOS are scaled to be double the widths of NMOS for this experiment, as indicated in Figure 16. The main purpose is to achieve equal rise and fall time, making the VCO output waveform symmetric which benefits the counter.

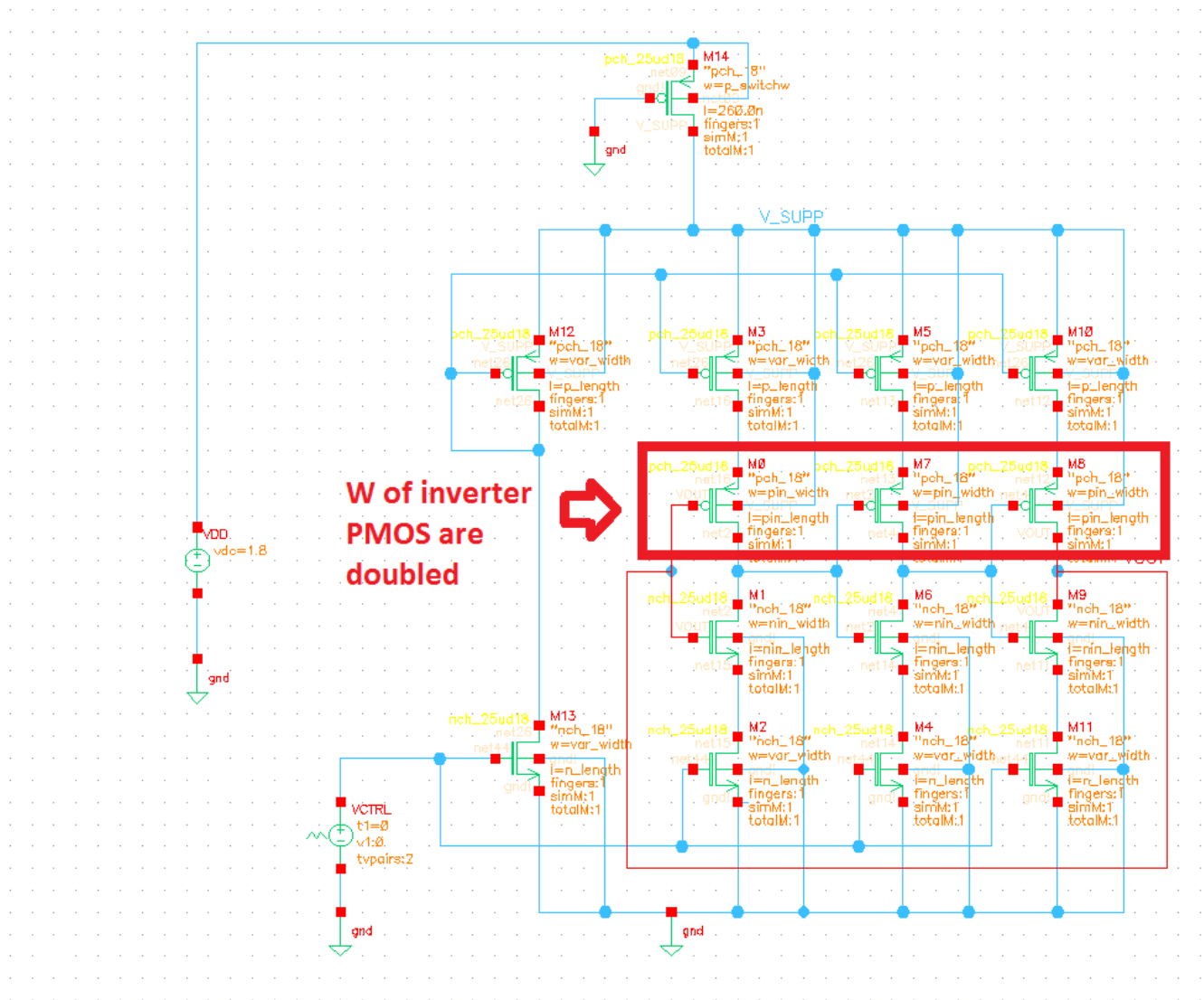
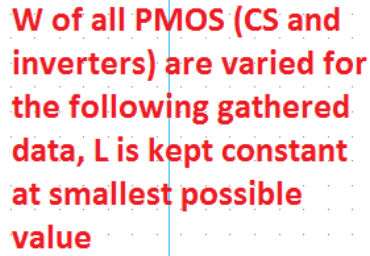


Figure 16: Ring Oscillator with Indicated Inverter PMOS

All PMOS width increment

For this simulation, the widths of all the PMOS in the VCO (except the PMOS switch) are scaled by multiples of two as portrayed in Figure 17. This experiment is to observe the performance of VCOs and ideally to make the output waveform more symmetric. For that reason, the rise and fall times are recorded for comparison.

• • •



The simulation results of the 1V supply 3 stage and 5 stage VCOs are included below. The other two supply voltages, 1.8V and 3.3V, are included in Appendix B. The MOSFET V-I characteristics are also included there.

VCO with 1V Supply: 3 Stage

Current Starved VCO

The three stage current-starved ring oscillator was able to attain nearly 16GHz in simulation, as presented in Figure 18. Note that it only starts oscillating once the control voltage (VCTRL) exceeds 300mV.

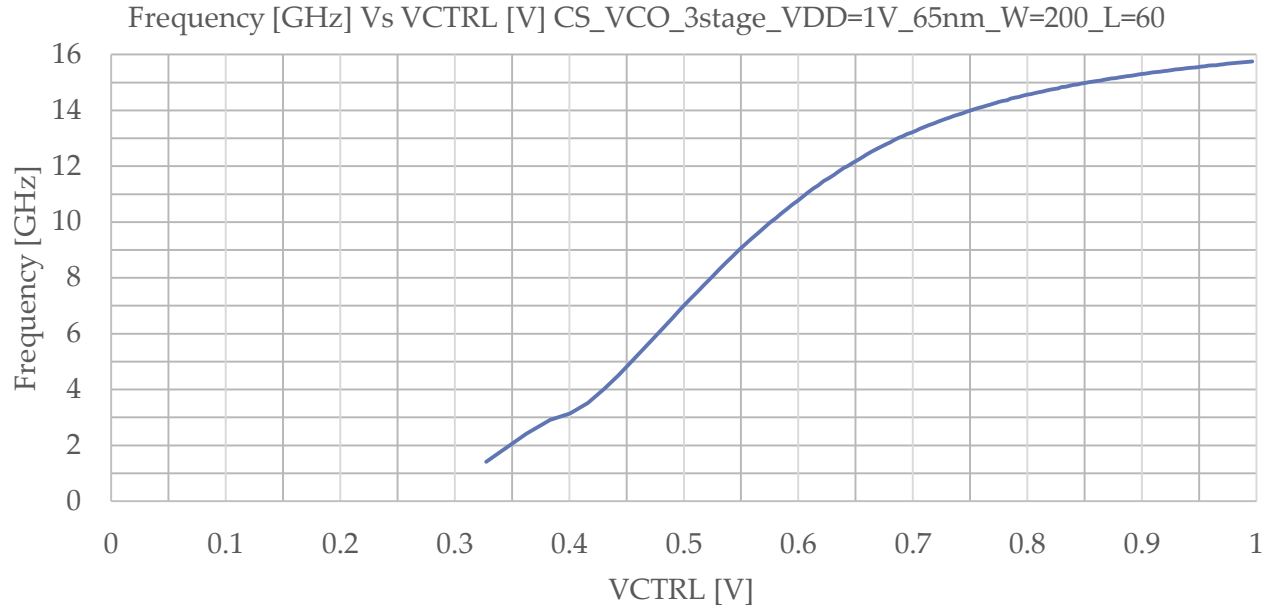


Figure 18: V-f characteristic of 3 Stage Current Starved VCO

Current Starved VCO with PMOS switch

Then we added the PMOS switch and swept the width to find an optimal value approximately equal to the frequency capabilities of the ring oscillator without the switch. In this first case presented in Figure 19, the switch width is varied from 200nm to 3.6 μ m. Note that at first the frequency curve variation is large, but as the PMOS switch width increases, the difference becomes smaller.

VCO-BASED ADC

• • •

V-f characteristic of VCO with addition of PMOS switch
(VCO = 3 stage, VDD = 1V, W = 200, L = 60, CSVCO)

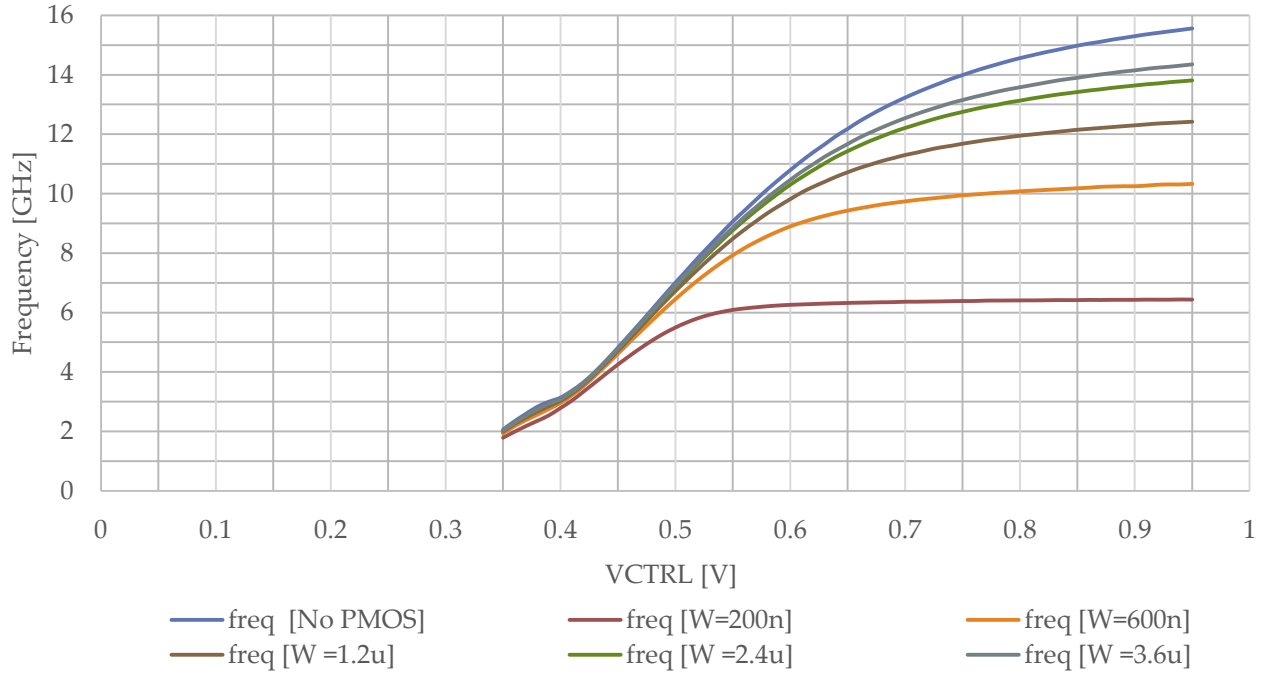


Figure 19: V-f characteristic of Current Starved VCO with PMOS switch (small)

This trend is more obvious in Figure 20, where the PMOS switch width is swept from 2.4 μ m to 9.6 μ m. Note that the variation between no switch and 9.6 μ m only becomes apparent outside the linear region.

V-f characteristic of VCO with addition of PMOS switch
(VCO = 3 stage, VDD = 1V, W = 200, L = 60, CSVCO)

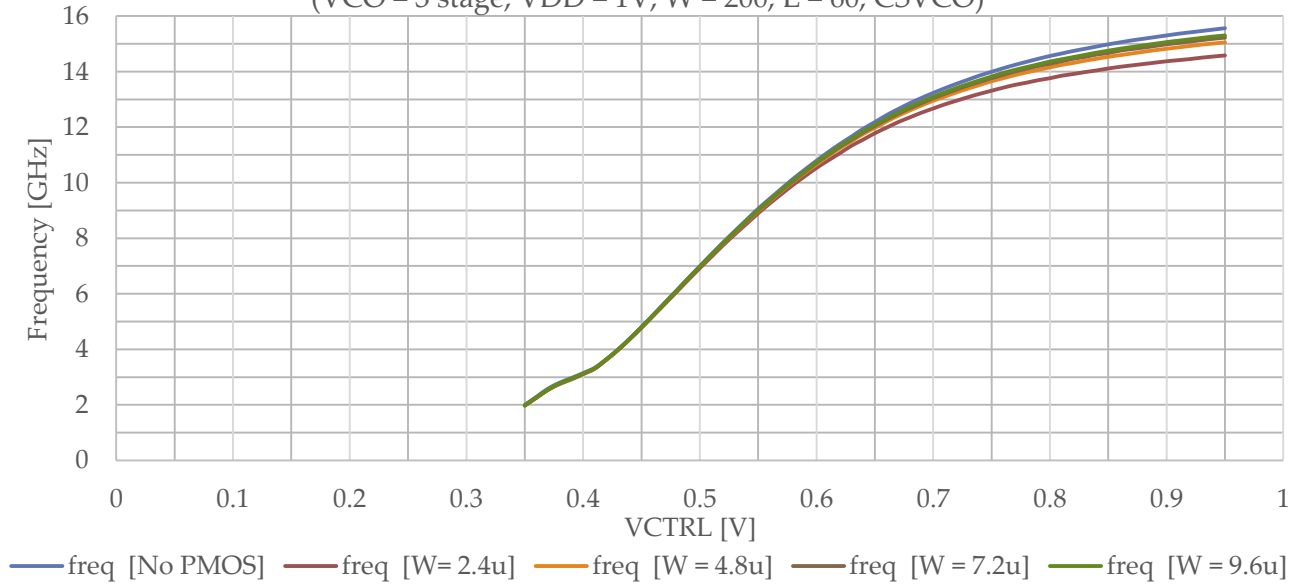


Figure 20: V-f characteristic of Current Starved VCO with PMOS switch (medium)

• • •

The final iteration in Figure 21 from 9.6 μm to 24 μm can only be seen at voltages above 700mV.

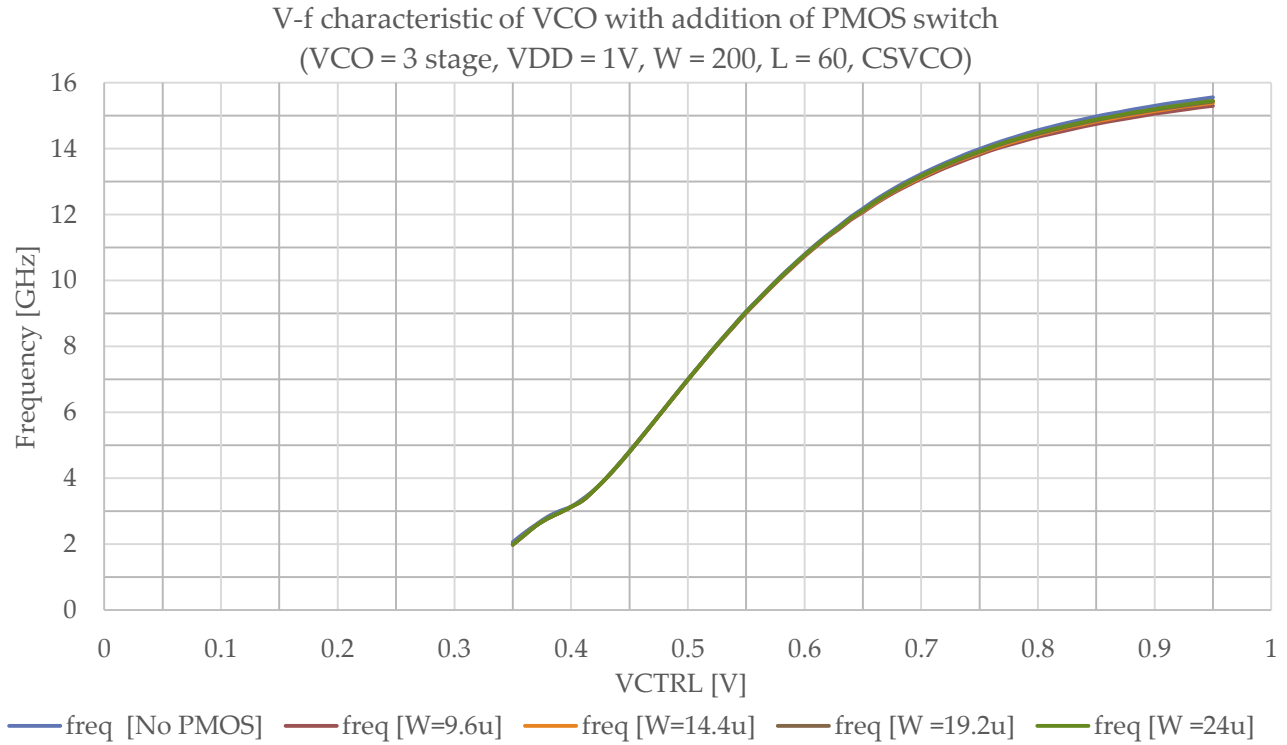


Figure 21: V-f characteristic of Current Starved VCO with PMOS switch (large)

From these simulations, repeated for the other supply voltages and for five stages, we determined that a 9.6 μm switch was acceptable.

Current Starved VCO Incrementing Width of Inverter PMOS

The next step was incrementing the width of the inverter PMOS, but as can be seen in Figure 22, only increasing the inverter PMOS width was not enough as the smaller current-sourcing PMOS were too small to drive them. The result was smaller frequency range for the same control voltage.

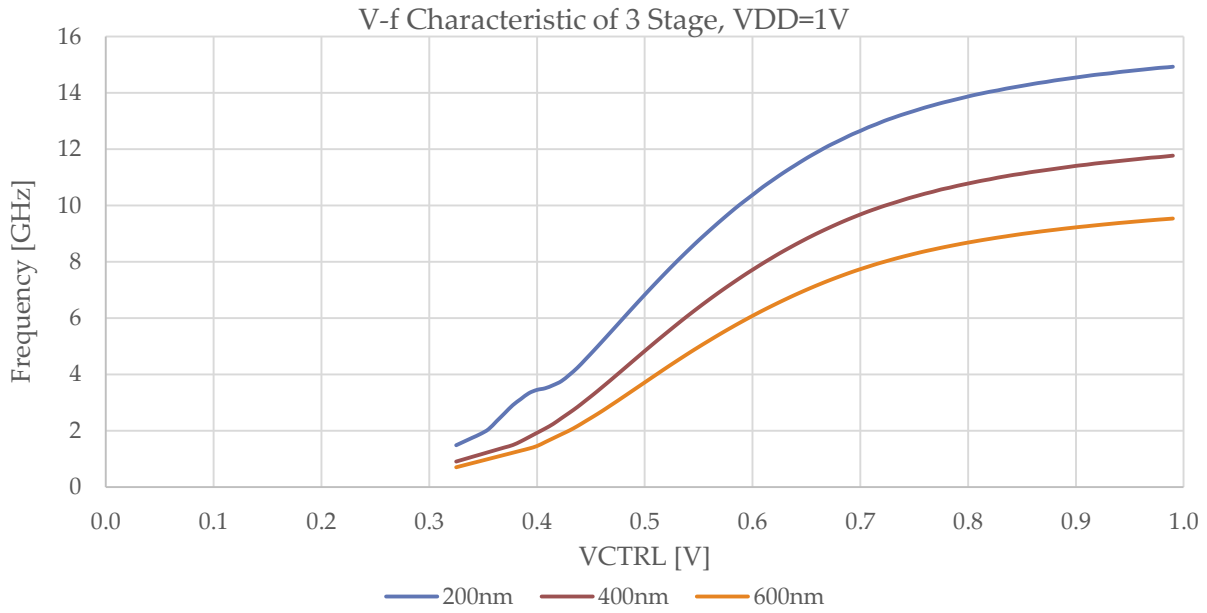


Figure 22: V-f characteristic of Current Starved VCO with PMOS Switch Sweeping Inverter PMOS Width

Current Starved VCO Incrementing Width of all PMOS

From there, we incremented the width of all the PMOS, but the frequency characteristic still suffered, as depicted in Figure 23.

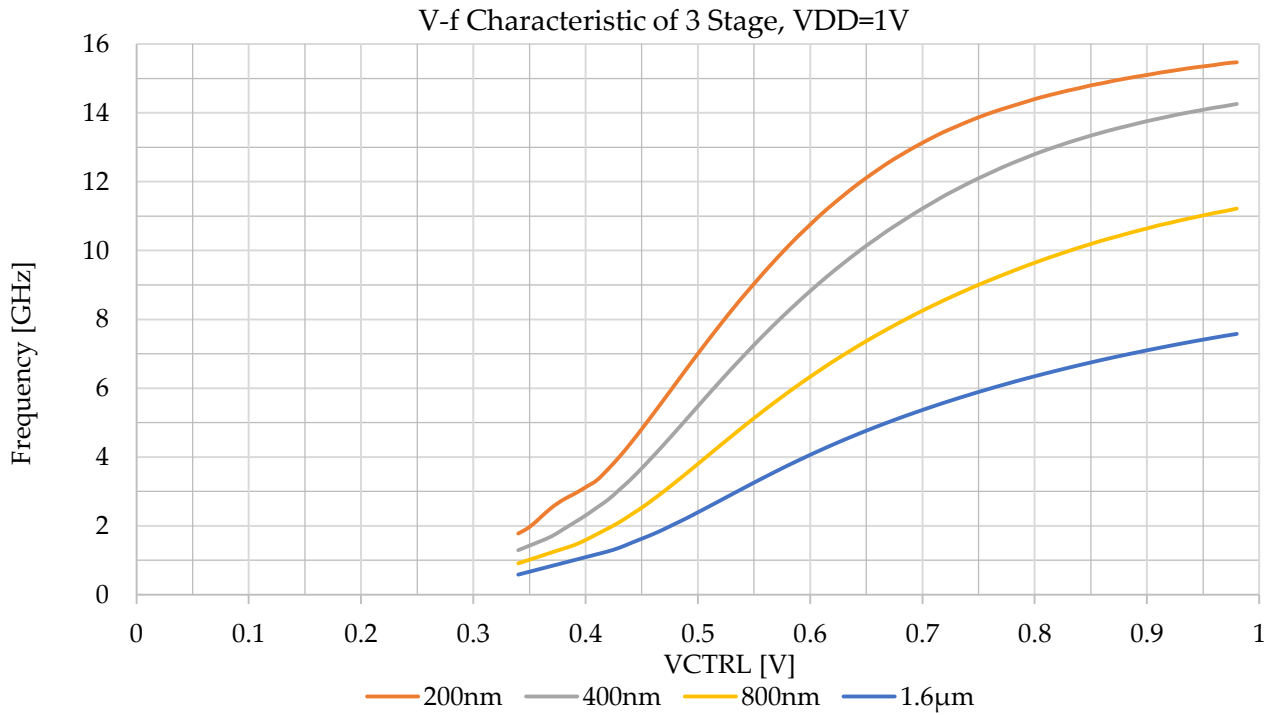


Figure 23: V-f characteristic of Current Starved VCO with PMOS Switch Sweeping PMOS Width

VCO with 1V Supply: 5 Stage

Current Starved VCO

From Figure 24, the five stage current-starved ring oscillator attained approximately 8.5GHz, as compared to the 16GHz for the three stage. Unlike the three stage, it starts oscillating before the control voltage (VCTRL) reaches 300mV.

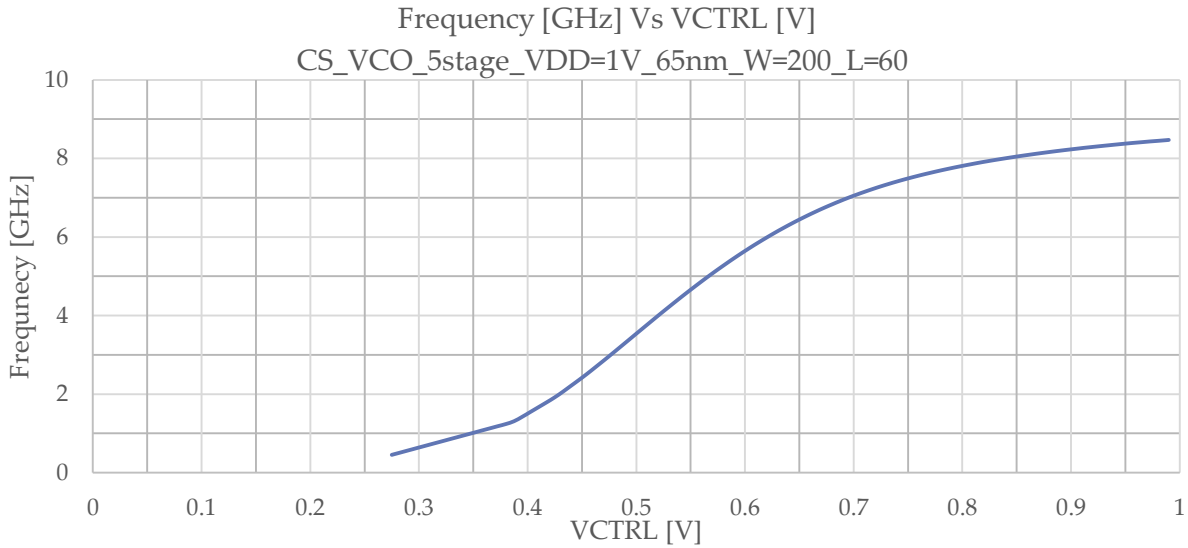


Figure 24: V-f characteristic of 5 Stage Current Starved VCO

Current Starved VCO with PMOS switch

Once we added the PMOS switch as in Figure 25, we saw behavior very similar to the three stage VCO. At about 9.6 μ m width, the PMOS switch hardly made a difference in the linear region.

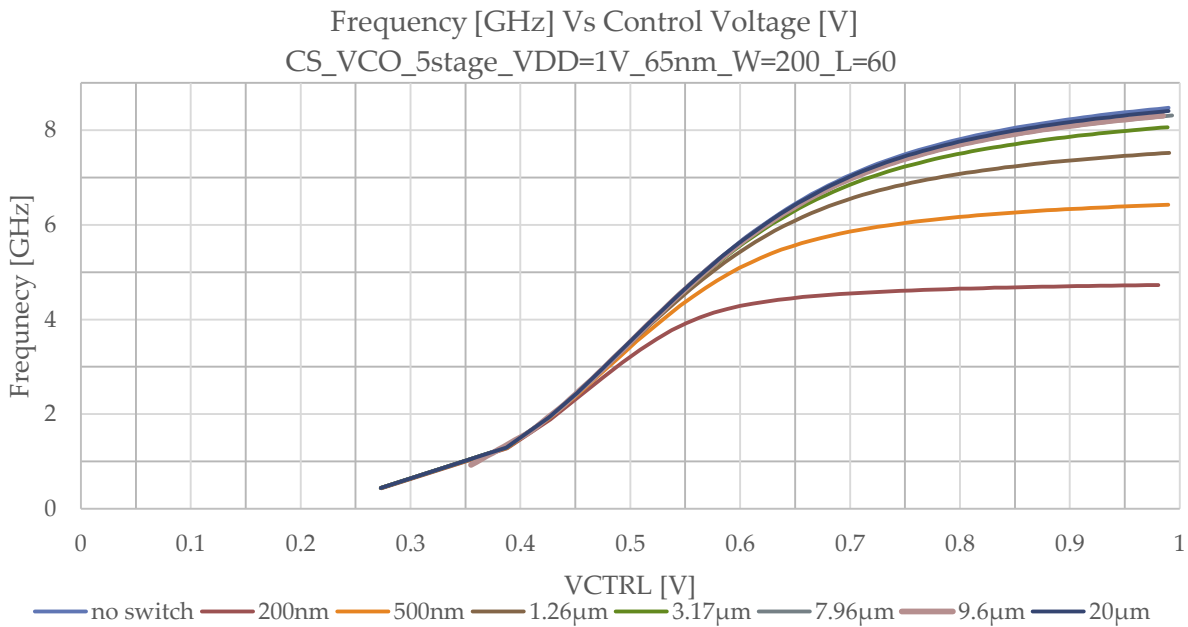


Figure 25: V-f characteristic of Current Starved VCO with PMOS switch

Current Starved VCO Incrementing Width of Inverter PMOS

Incrementing the width of the inverter PMOS yet again cost frequency to voltage resolution and decreased the overall linear region as indicated in Figure 26.

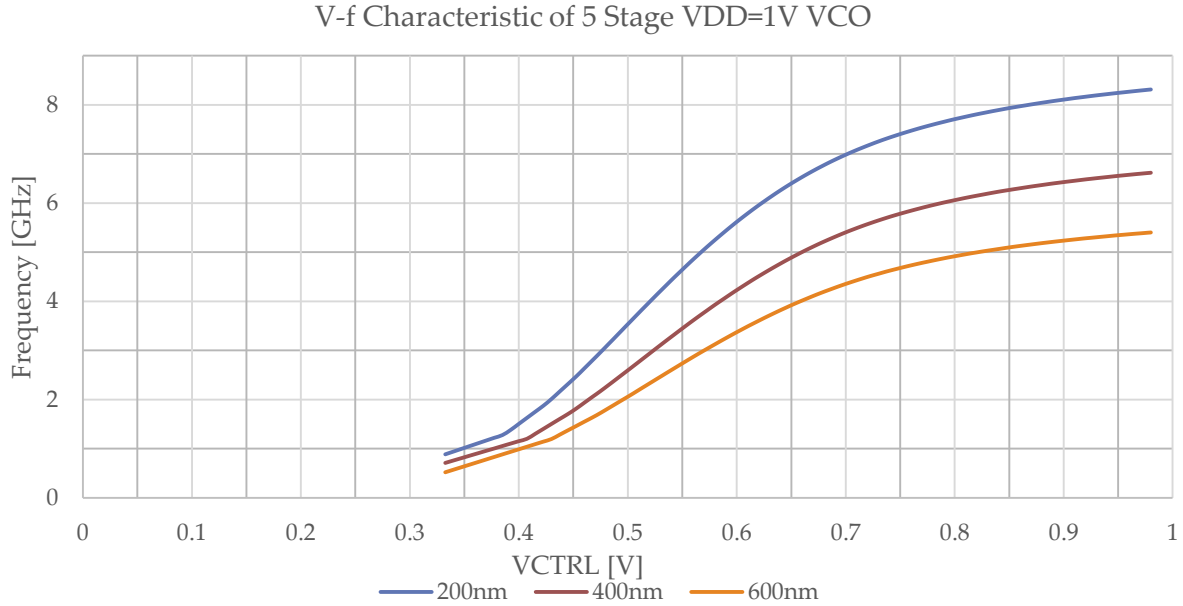


Figure 26: V-f characteristic of Current Starved VCO with PMOS Switch Sweeping Inverter PMOS Width

Current Starved VCO Incrementing Width of Current Sourcing and Starving MOS

Figure 27 examines how scaling the current-starving components impacts the frequency characteristic. Note that too large values, such as $1\mu\text{m}$ and $2\mu\text{m}$ introduce greater nonlinearity, even though they are oscillating for a larger voltage range.

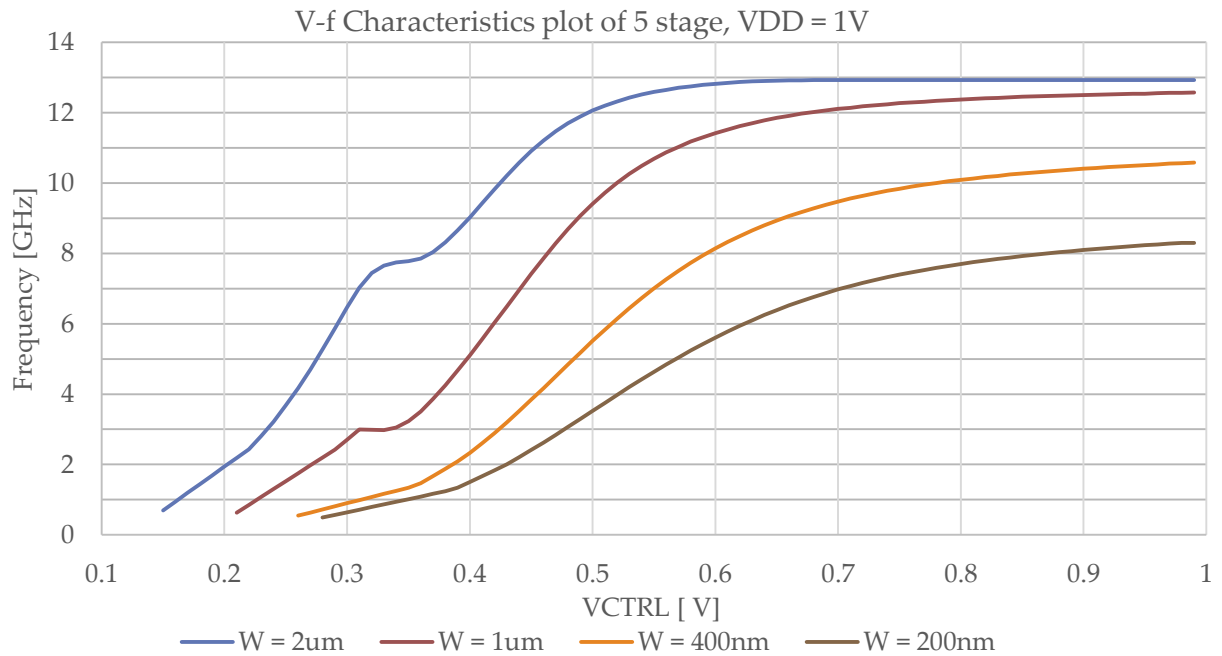


Figure 27: V-f characteristic of Current Starved VCO with PMOS Switch Sweeping Current Starving MOSFET Width

Current Starved VCO Incrementing Width of all PMOS

Scaling the width of all the PMOS decreases the frequency range but increases the relatively linear region.

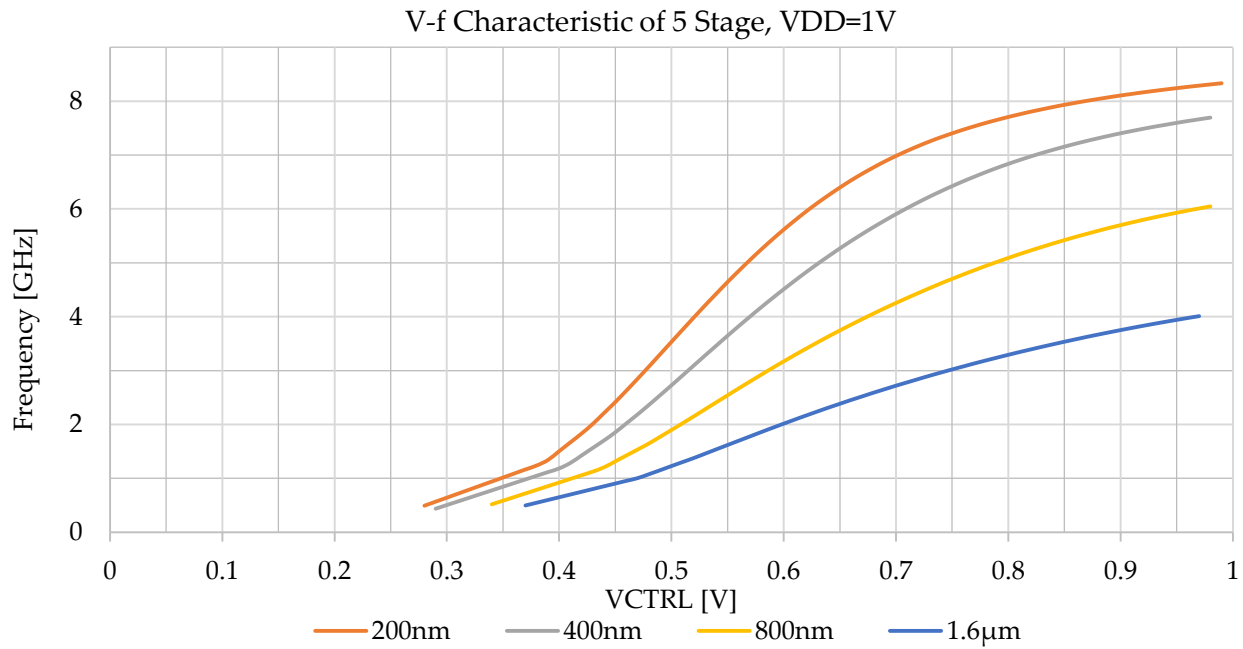


Figure 28: V-f characteristic of Current Starved VCO with PMOS Switch Sweeping PMOS Width

Analog MUX

The Analog MUX made of transmission gates (Figure 29) is designed for the ADC. Firstly, the type and width of MOSFETs to build the transmission gates is determined. Since the VCO outputs range is from 0V to 3.3V, MOSFETs which can support 3.3V are used. Smallest possible width was first used to save die area. However, once we measured the AC response for the 4-to-1 MUX using MOSFETs with baseline dimensions, the resultant bandwidth is in tens of MHz which is relatively too small for GHz outputs of the VCOs.

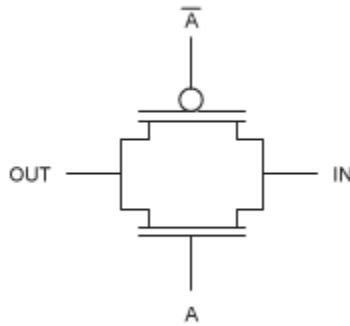
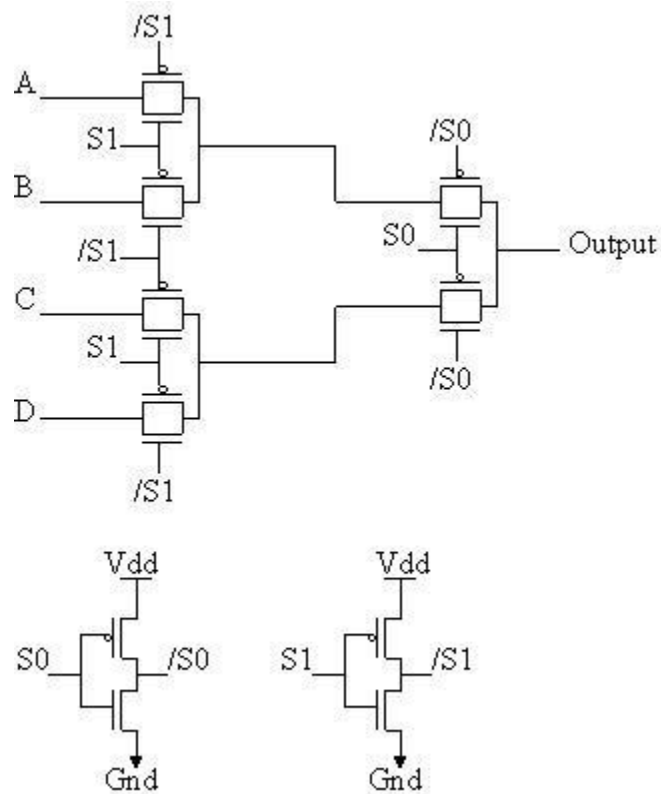


Figure 29: A transmission gate

The expected problem was the resistance of transmission gates, in this case the resistance of two transmission gates in series which are turned on when an input is connected. The 4-to-1 transmission gate is given in Figure 30.

• • •



S0	S1	Output
1	1	A
0	1	C
1	0	B
0	0	D

Figure 30: 4-to-1 Transmission Gate

Analog Transmission Gate Resistance Measurement

In order to measure the resistance of each transmission gate of the 4-to-1 analog MUX, several measurement approaches were done. We started with AC analysis, then step response and finally measuring the slope from V-I characteristics.

AC response

Baseline

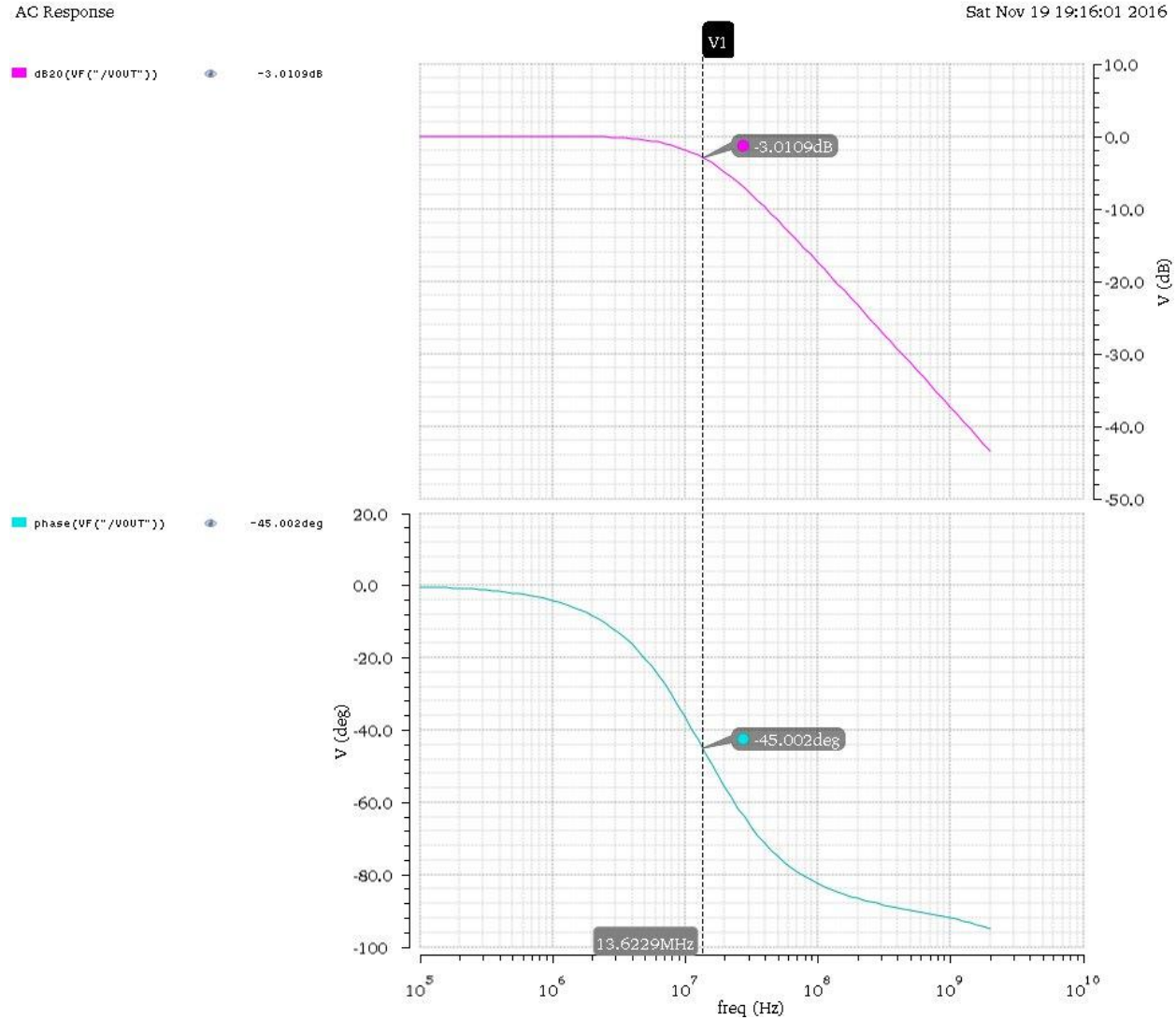


Figure 31: AC response of MUX with PMOS and NMOS width = 400nm

From the AC response plot (Figure 31), the f_{3dB} is 13.6229MHz. By using the following equation:

$$f_{3db} = \frac{1}{2\pi RC} \quad (8)$$

where C is 1pF, the calculated resistance of two transmission gates in a series R is 11.68kΩ. Thus since there are two TGs in series in the signal path, the resistance of each TG is 5.8kΩ. Next the width of PMOS and NMOS are varied to see the change in f_{3dB} as well as the resistance.

VCO-BASED ADC

• • •

PMOS and NMOS width = $4\mu\text{m}$

AC Response

Mon Dec 5 16:18:23 2016

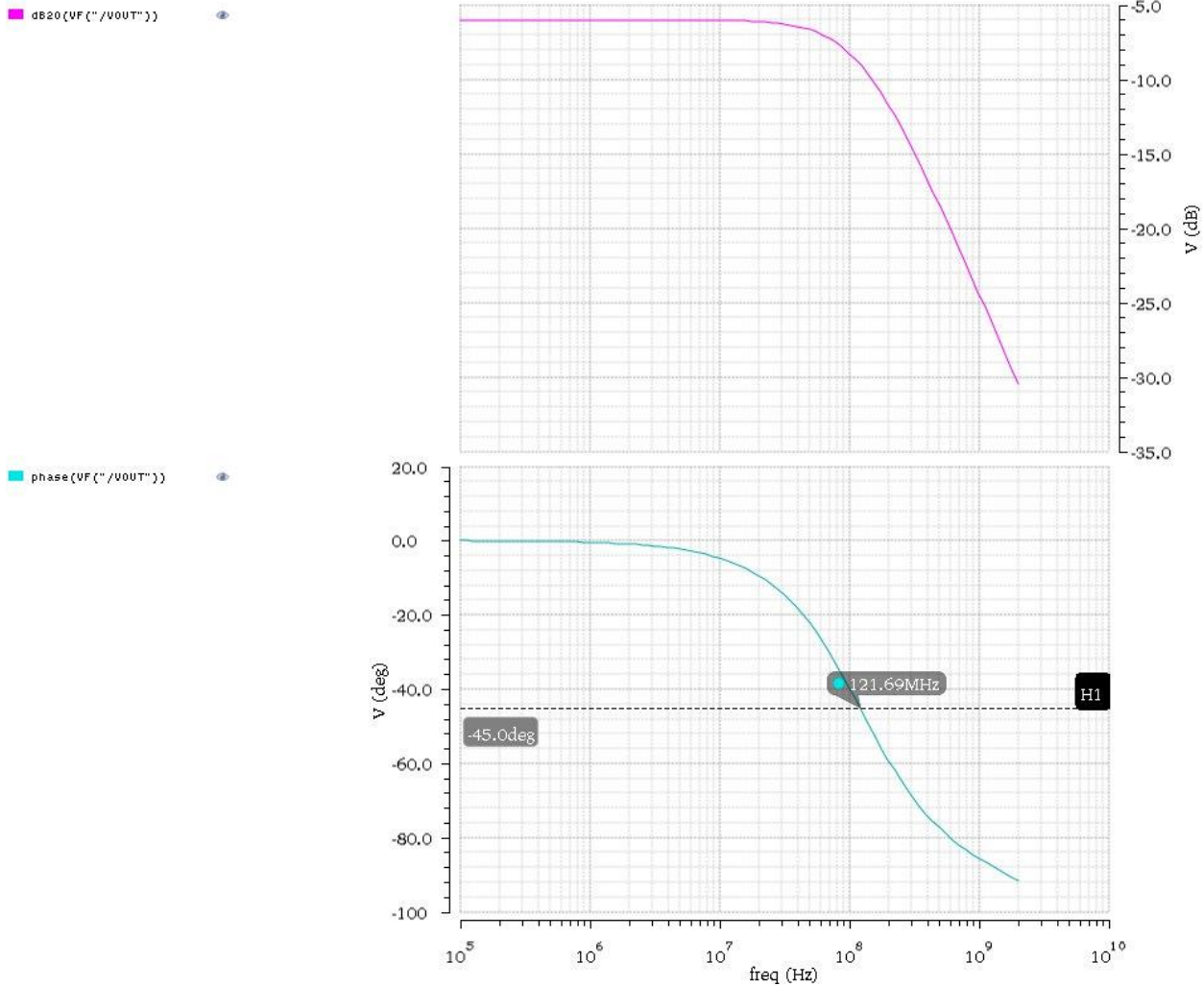


Figure 32: AC response of MUX with PMOS and NMOS width = $4\mu\text{m}$

From Figure 32, the 3db frequency is 121.69MHz, and by using equation (8), the calculated resistance of two transmission gates in a series is $1.307\text{k}\Omega$.

VCO-BASED ADC

• • •

PMOS and NMOS width = 40 μ m

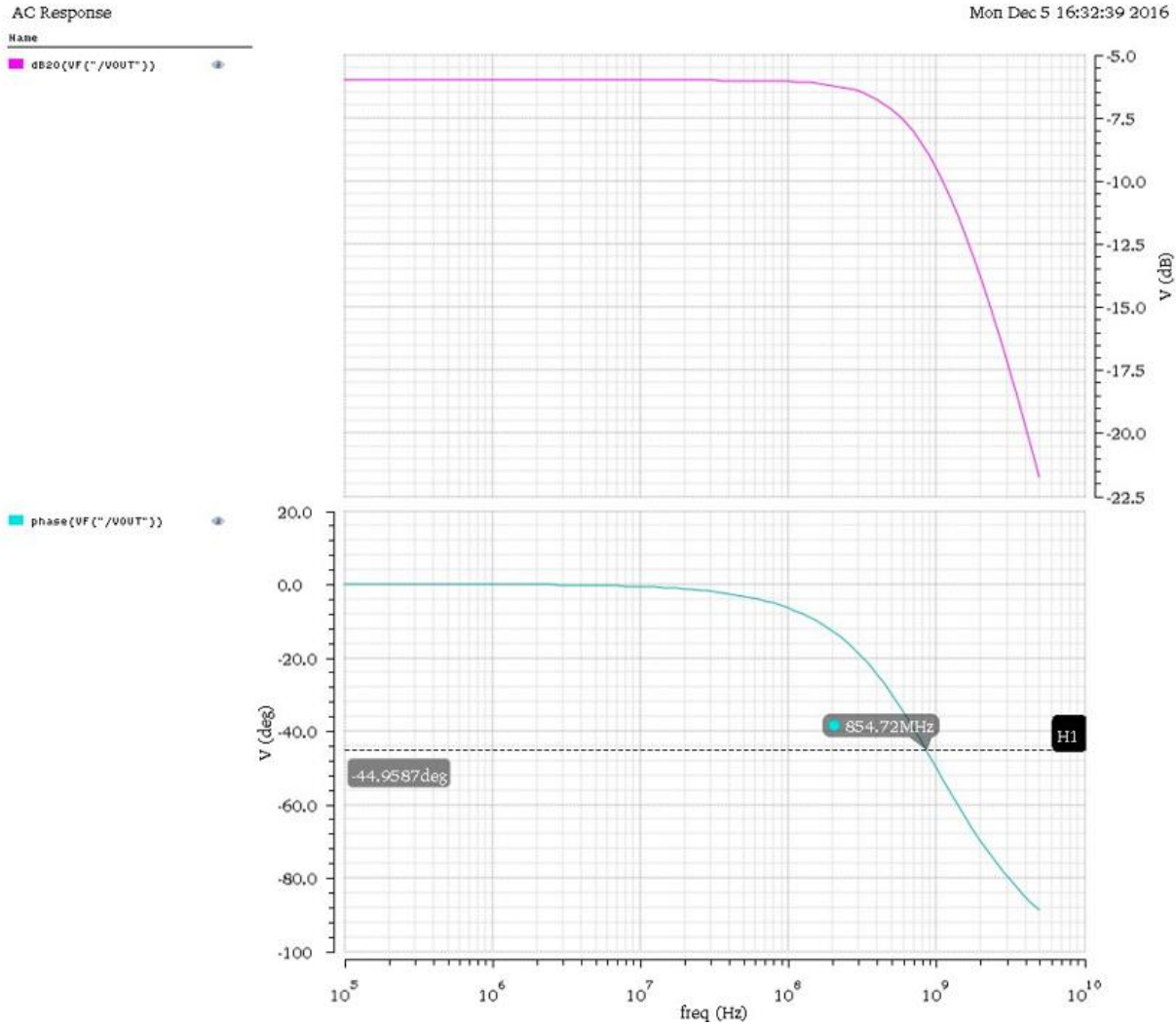


Figure 33: AC Response of MUX with PMOS and NMOS width = 40 μ m

From Figure 33, the 3db frequency is 854.72MHz, and by using equation (8), the resistance of two transmission gates in a series is 186 Ω .

VCO-BASED ADC

• • •

PMOS and NMOS width = 400 μ m

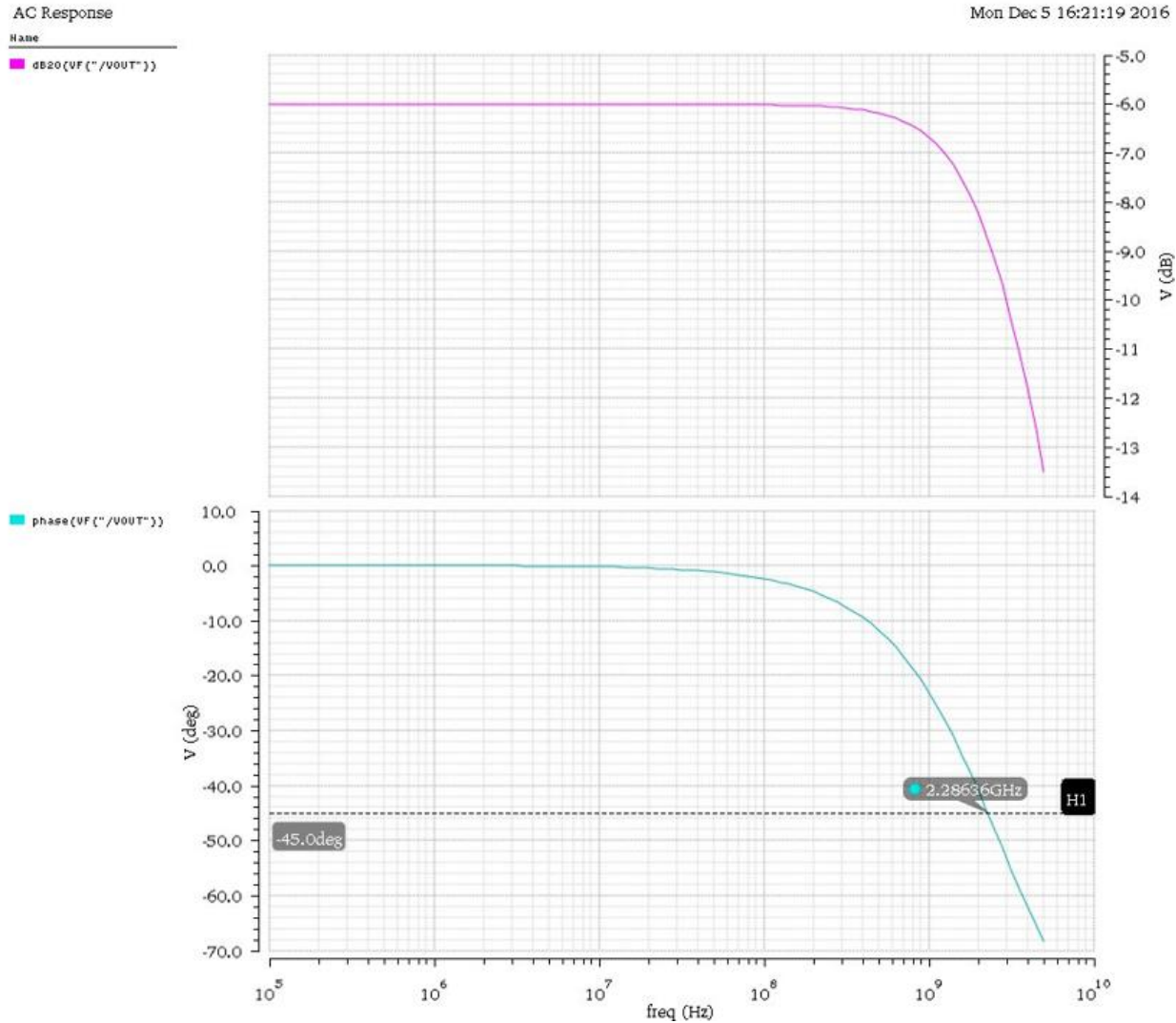


Figure 34: AC response of MUX with PMOS and NMOS width = 400 μ m

From Figure 34, we know that the 3db frequency is 2.29636GHz, and by using equation (8), we receive the total resistance value to be 69.6 Ω .

Step Response

Next we measured the resistance of the MUX by having a square wave at the input and measure the rise time at the output of the MUX. Then using Bandwidth-Risetime relationship

$$BW \times t_r = 0.35 \quad (9)$$

we calculate the 3dB frequency and the resistance.

PMOS and NMOS width = 400nm

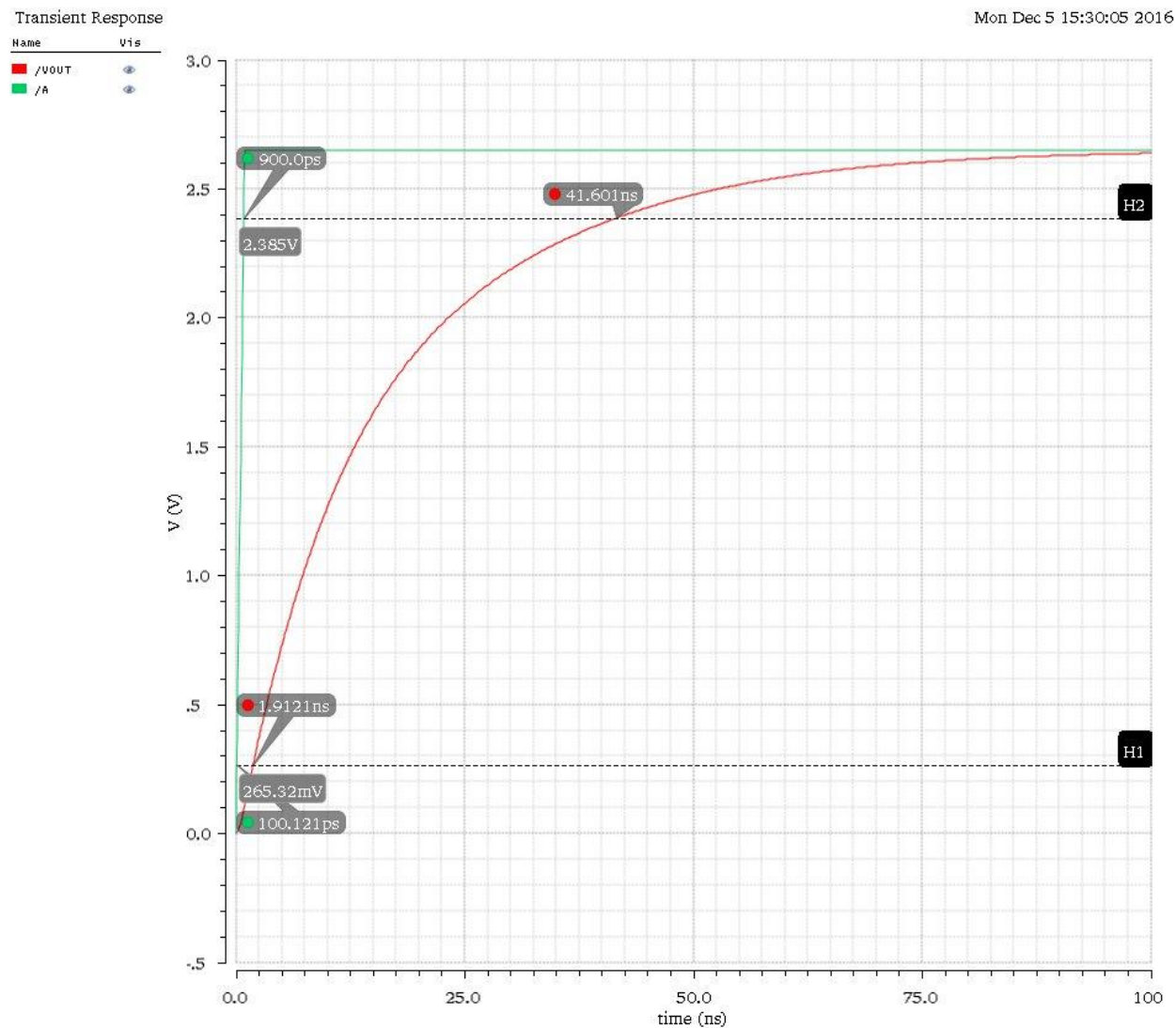


Figure 35: Step Response with PMOS and NMOS width = 400nm

The measured rise time is 39.6879ns. The resistance is 18kΩ.

VCO-BASED ADC

• • •

PMOS and NMOS width = $4\mu\text{m}$

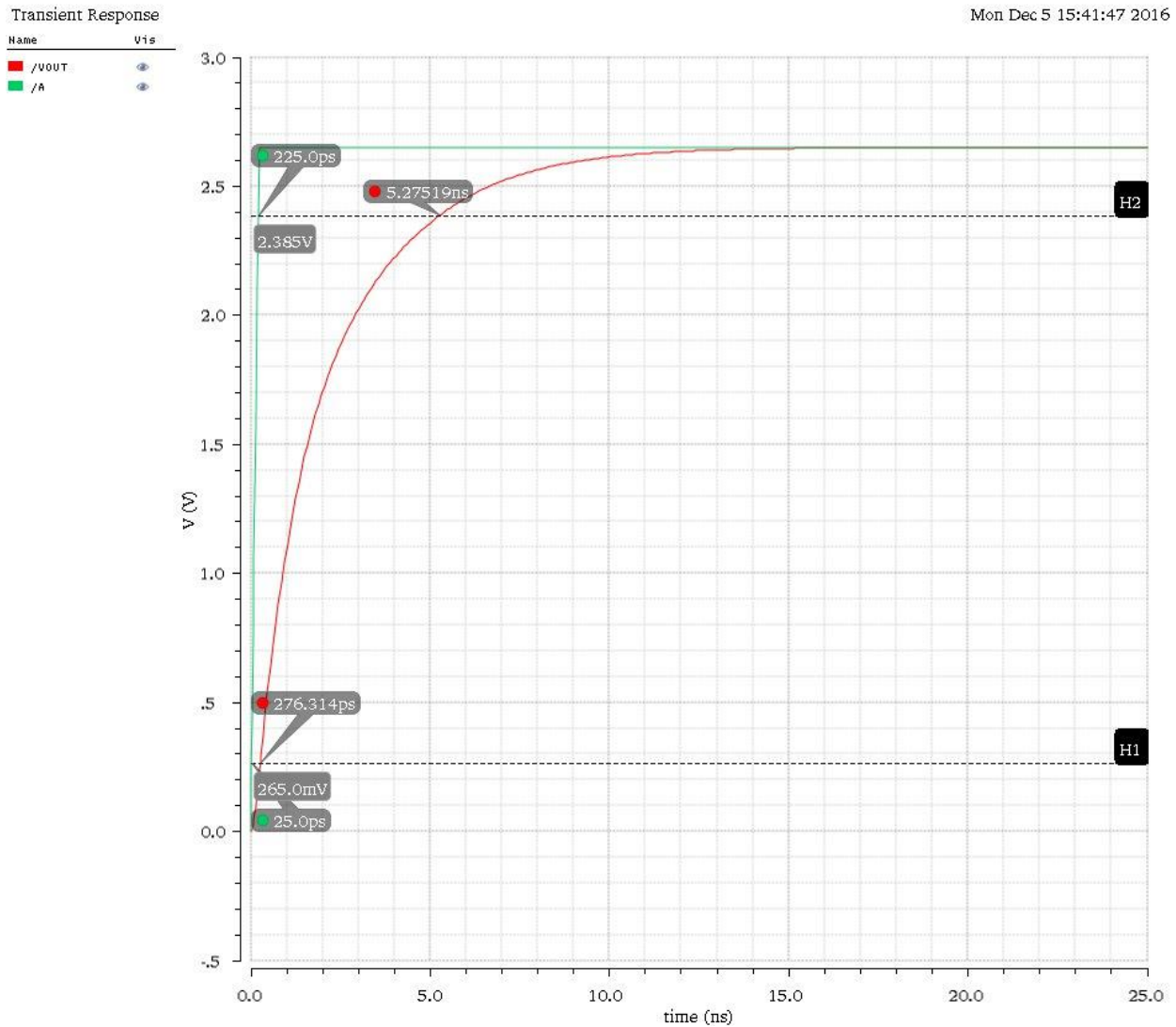


Figure 36: Step Response with PMOS and NMOS width = $4\mu\text{m}$

The measured rise time is 4.99919 ns and the resistance is 2273 Ω .

VCO-BASED ADC

...

PMOS and NMOS width = 40 μ m

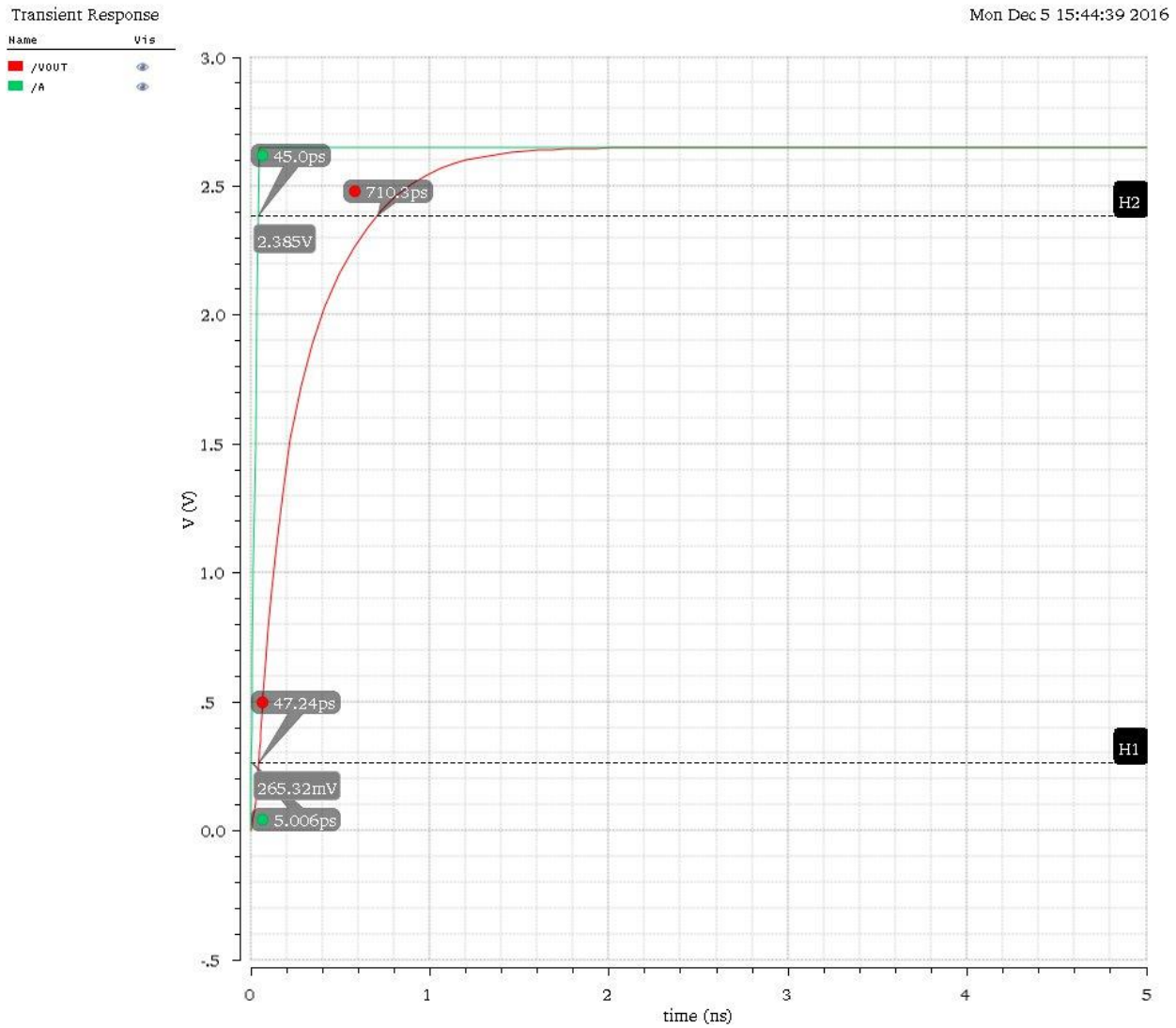


Figure 37: Step Response with PMOS and NMOS width = 40 μ m

The measured rise time is 663ps and the resistance is 301.51 Ω .

• • •

Measuring voltage drop across a single transmission gate

Next we put a current source as shown in Figure 38 and Figure 39 to measure the voltage drop across the transmission gates. Then we receive the value of resistance by dividing the delta in voltage by delta in current.

PMOS and NMOS width = 400nm

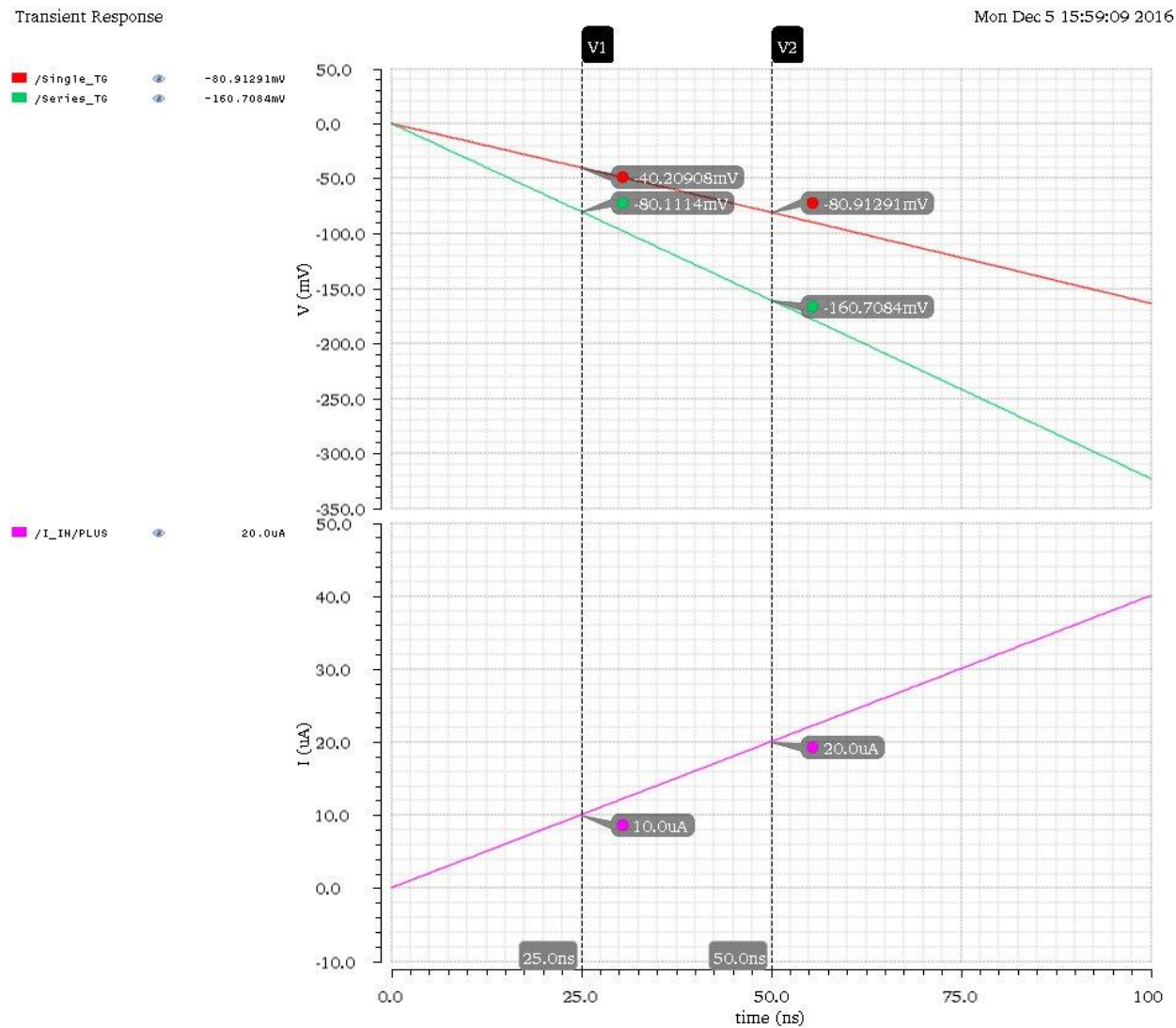


Figure 38: measuring the delta V and delta I to find the resistance (red = single TG, green = two TG in series, purple = current source)

The resistance for single TG is 4kohms and the resistance of series TG is 8kΩ. Thus this value is very close to what we received from step response. However, we measured the current at PMOS and NMOS in TG, we found out that the current at the PMOS source is 0. Thus the PMOS is off and only NMOS is conducting the current. This in turn tells us that the resistance we are measuring seems to be that of

VCO-BASED ADC

• • •

NMOS. So we set a 3.3V source instead of ground at the source of PMOS, basically turning off the NMOS and turning on the PMOS.

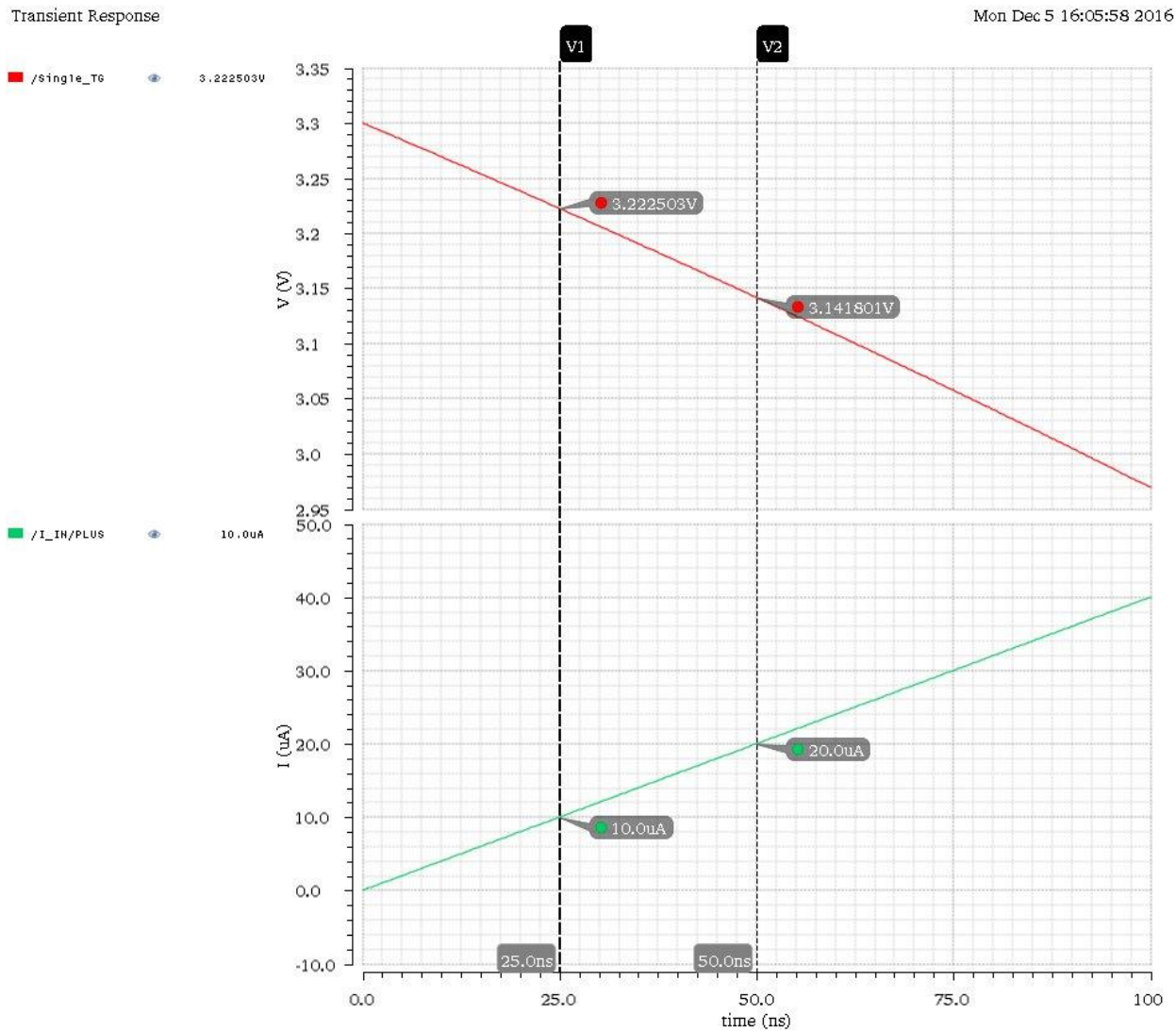


Figure 39: measuring the delta V and delta I to find the resistance of single TG

The resistance that we measured is 8070Ω .

• • •

Conclusion

From the above measurements, the resistance of a single transmission gate is 8-10k Ω . This confirms the original suspicion that a large resistance was behind the bandwidth issues. From all the testing done on the MOSFETs, increasing the width does help reduce the resistance but it is not an ideal solution for the low bandwidth of the MUX. The alternative solution would be using a digital MUX or adding a buffer to the circuit.

Buffer

Inverter Buffer

To mitigate the bandwidth issues coming out of the multiplexer, buffers need to be implemented after each of the transmission gates. The buffers used in the simulations were simple inverters made out of a CMOS; they are similar to the inverters making up the VCO itself except scaled to appropriate dimensions.

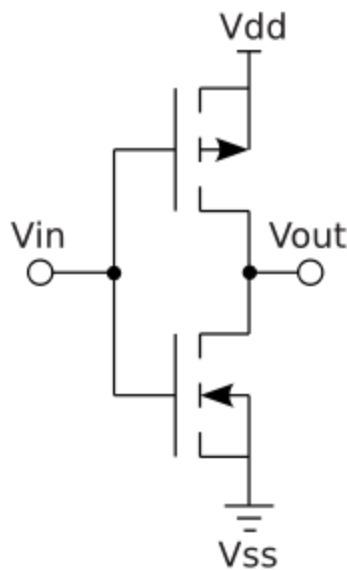


Figure 40: Example of an inverter buffer

Tapered Buffer

A single inverter buffer after each buffer unfortunately does not adequately reduce the total capacitance in the system, as presented in Figure 41 and Figure 42. In order to further reduce the capacitance, progressively larger inverting buffers cascade each other after the multiplexer output. The desired load ultimately is driven using this arrangement since the overall capacitance behavior of the tapered buffer is much less than that of just one inverting buffer.

VCO-BASED ADC

• • •

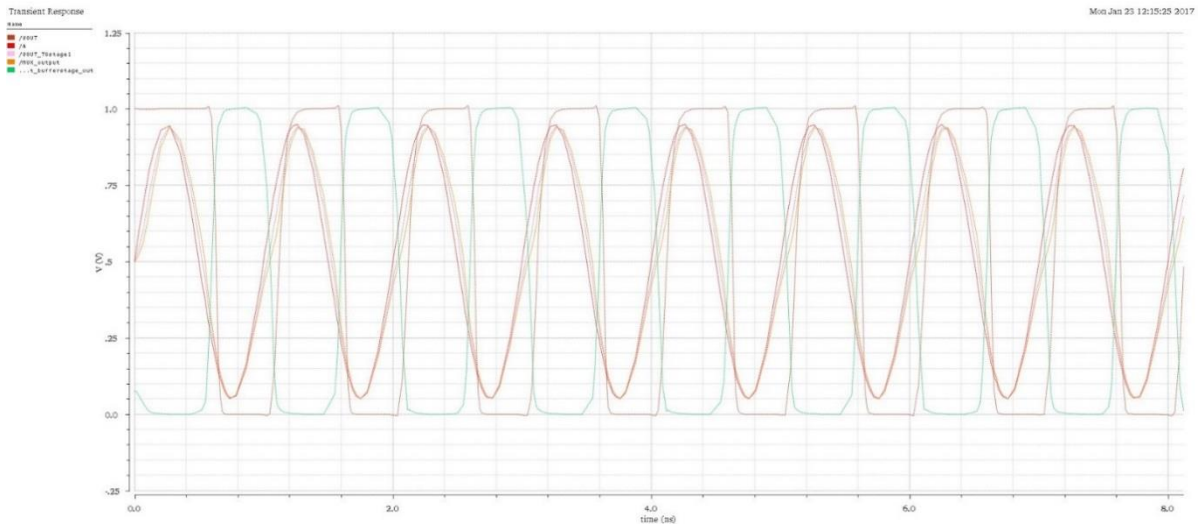


Figure 41: The output of the multiplexer failing to fully oscillate at 1GHz

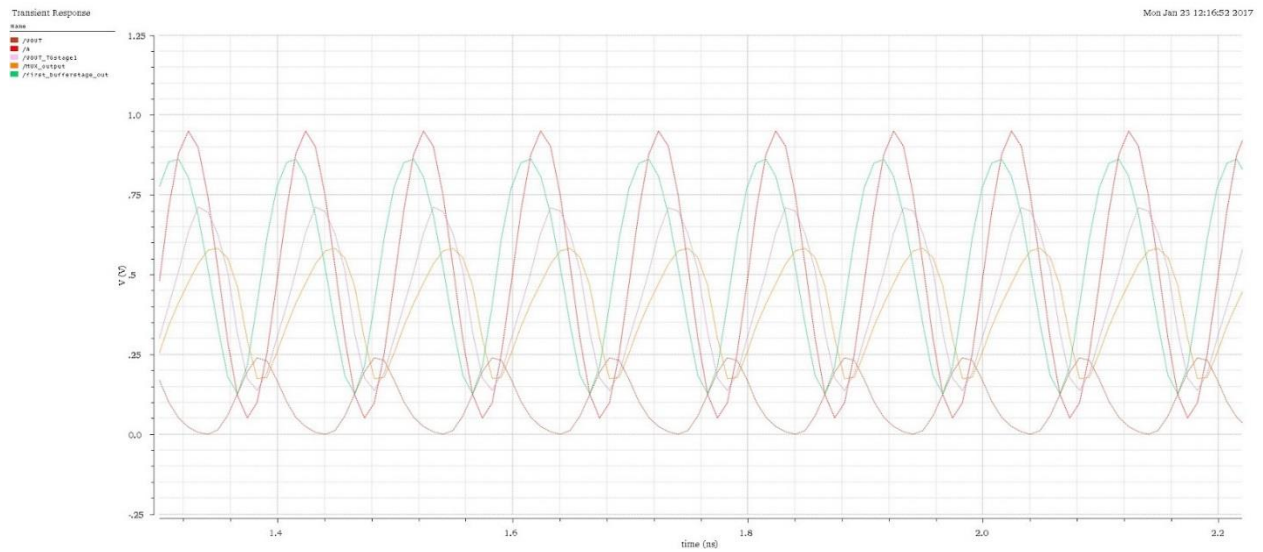


Figure 42: The output of the multiplexer failing to fully oscillate worsening at 10GHz

The tapered buffer schematic is provided in Figure 43.

VCO-BASED ADC

• • •

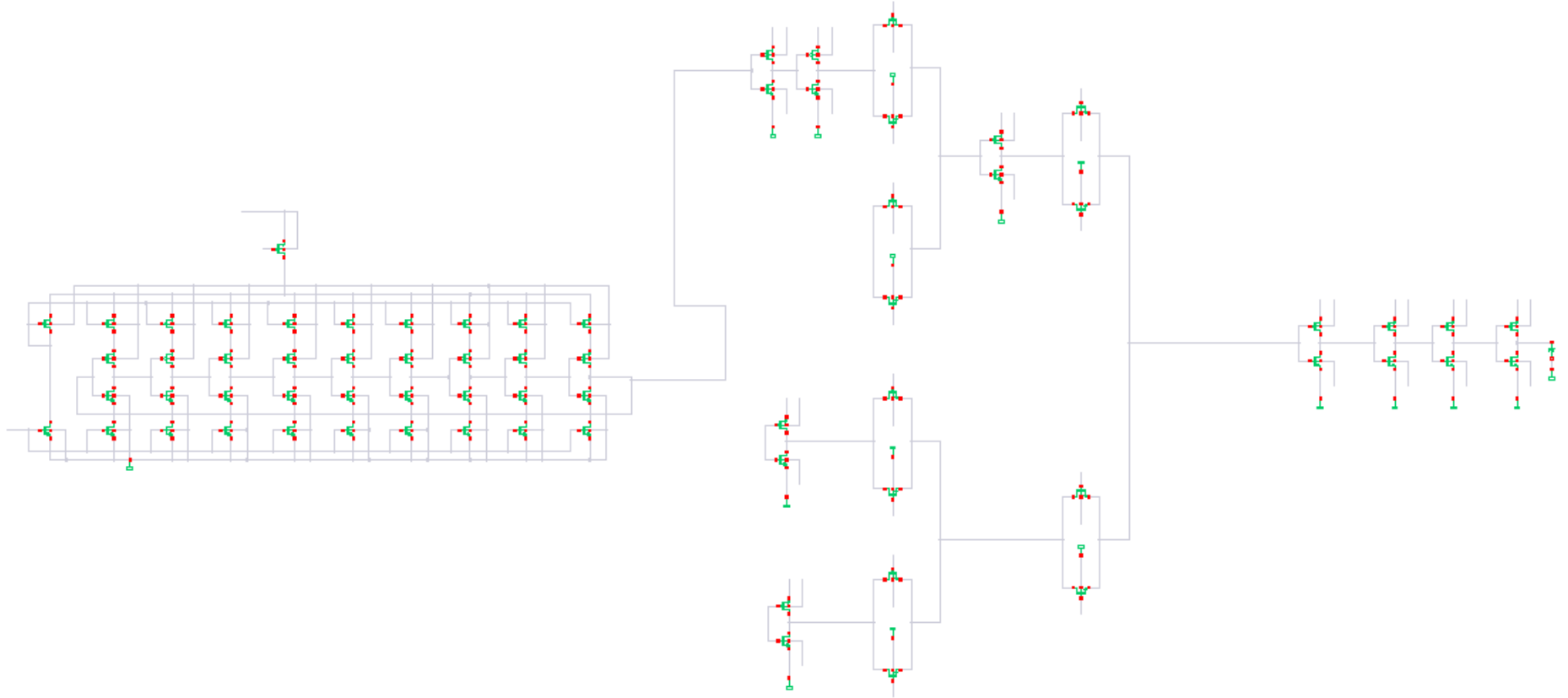


Figure 43: 9 stage 3.3V supply connected to a 4 stage tapered buffer

Simulation Results

Testing the effectiveness of the buffers like with the multiplexer, began with the 3.3V since it was easier to get oscillating with fewer stages in the VCO compared with other voltage levels. The tapered buffer functioned correctly with just 4 stages. Eventually, testing transitioned to the 1V which both had more stages in the VCO and tapered buffer.

For clarity, 4-stage and 5-stage buffers refer to the number of inverting buffer stages after the analog mux; the amount of inverter buffers within and before the analog mux did not contribute to the stage count since they were fixed at 2 and 1 stages respectively. A table has been provide with the widths of the MOSFETs comprising each buffer

Table 1: Dimensions of tapered buffer components

Location of Tapered Buffer components	Width	Length for 3.3V	Length for 1V
First Stage Before Analog MUX	800nm	400nm	60nm
Second Stage Before Analog MUX	2.4 μm	400nm	60nm
Stage within Analog MUX	1.2 μm	400nm	60nm
First stage after Analog MUX	4 μm	400nm	60nm
Second stage after Analog MUX	16 μm	400nm	60nm
Third stage after Analog MUX	64 μm	400nm	60nm
Fourth stage after Analog MUX	128 μm	400nm	60nm
Fifth stage after Analog MUX	256 μm	400nm	60nm

3.3V

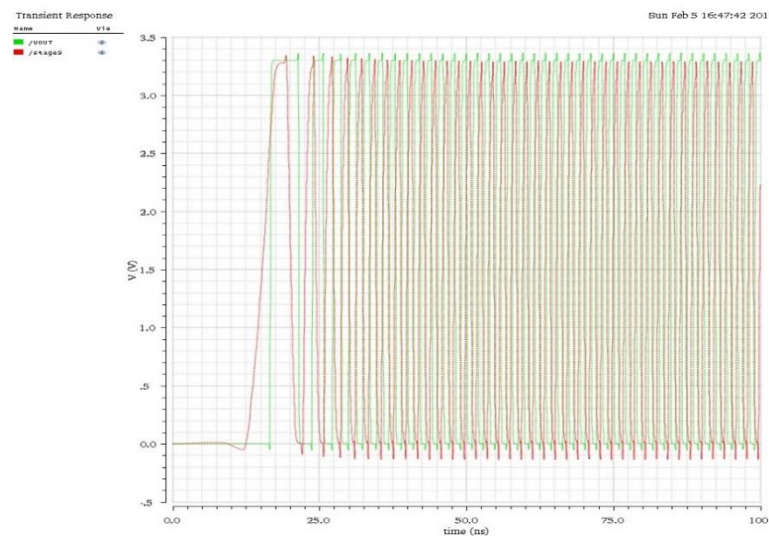


Figure 44: Waveforms of the 9 stage 3.3V after the VCO and after the tapered buffer

VCO-BASED ADC

• • •

The 9 stage VCO was the first to be implemented with a tapered buffer. The largest width of the MOSFETs used in the tapered buffer was $128\mu\text{m}$ in the 4 stage buffer. The 7 stage VCO had its tapered buffer implementation done next. The results were very similar to the 9 stage other than a slightly lower frequency.

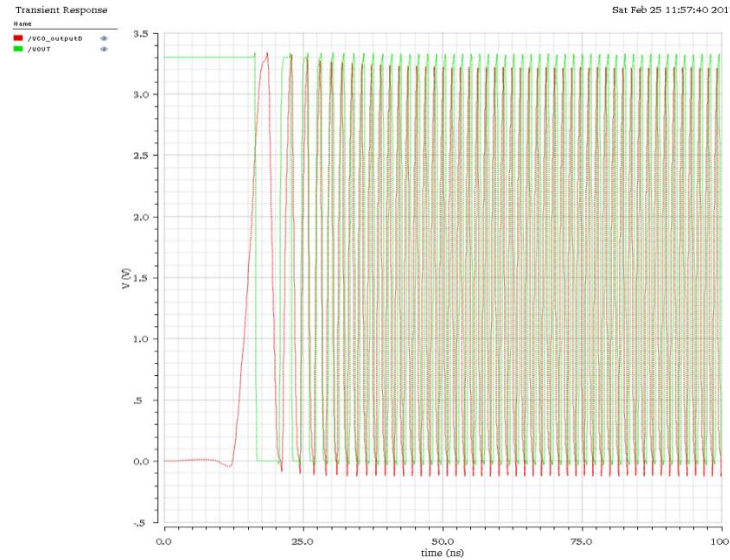


Figure 45: Waveforms of the 7 stage 3.3V supply after the VCO and after the tapered buffer

Unfortunately, the 5 stage VCO did not have the same success since the oscillating waveform at the output of the VCO could not reach the 3.3 V while oscillating.

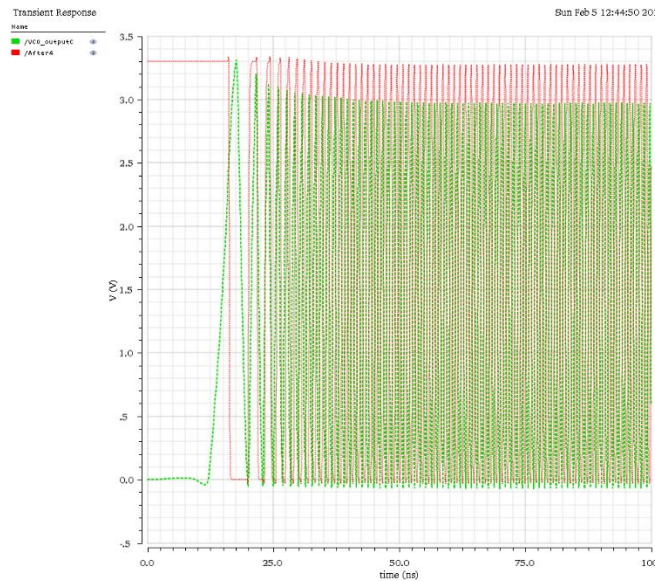


Figure 46: Waveforms of the 5 stage 3.3V after the VCO and after the tapered buffer

VCO-BASED ADC

• • •

While trying to get the 5 stage VCO to oscillate the full 3.3V range output, a 5 stage tapered buffer was experimented with. Both the 7 stage and 9 stage VCOs saw improvement in their range of frequencies as the control voltage was slowly raised over 60ns.

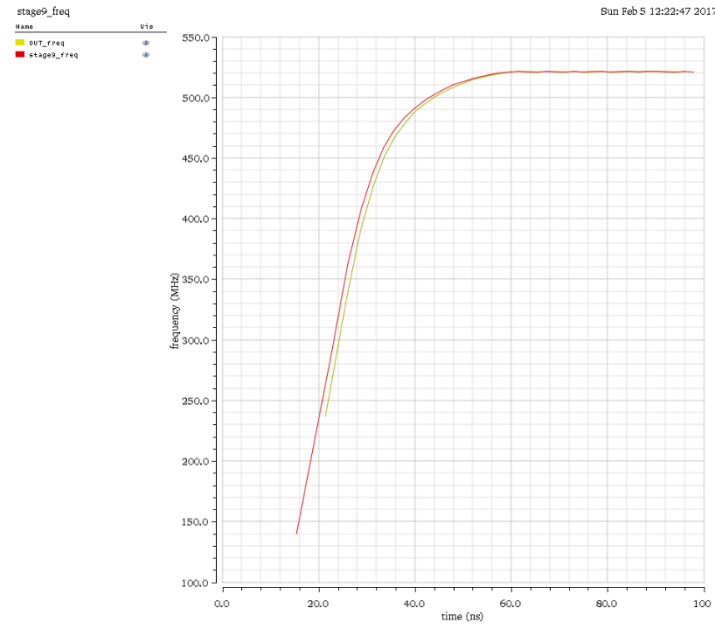


Figure 47: 9 stage VCO with 4 stage buffer

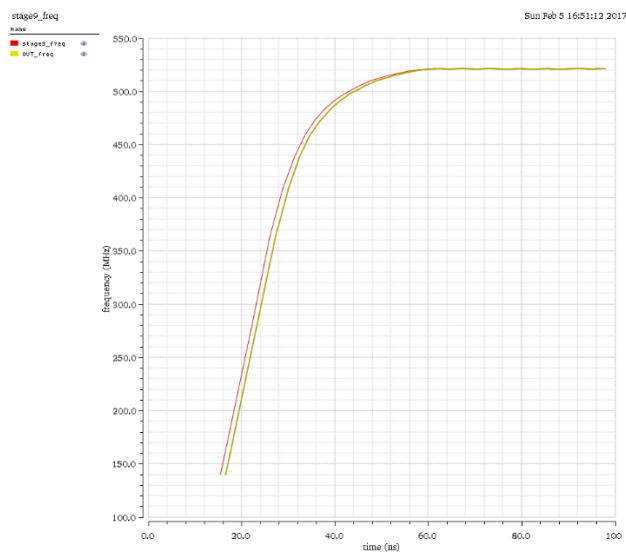


Figure 48: 9 stage VCO with 5 stage buffer

VCO-BASED ADC

• • •

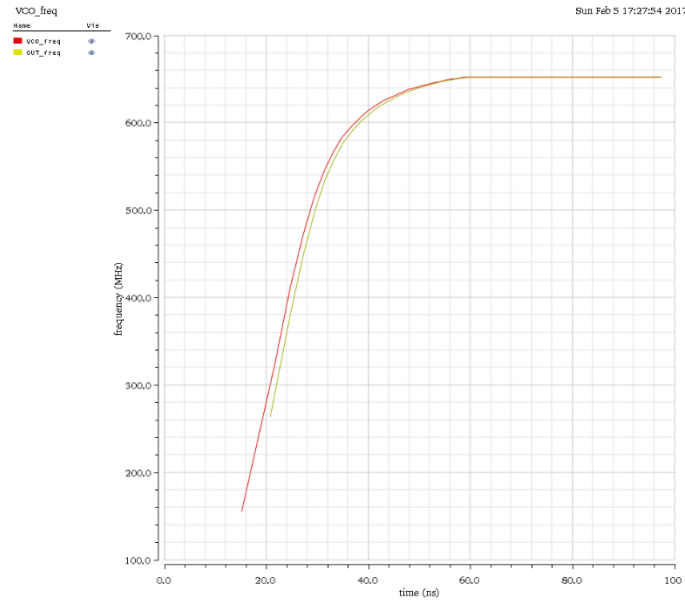


Figure 49: 7 stage VCO with 4 stage buffer

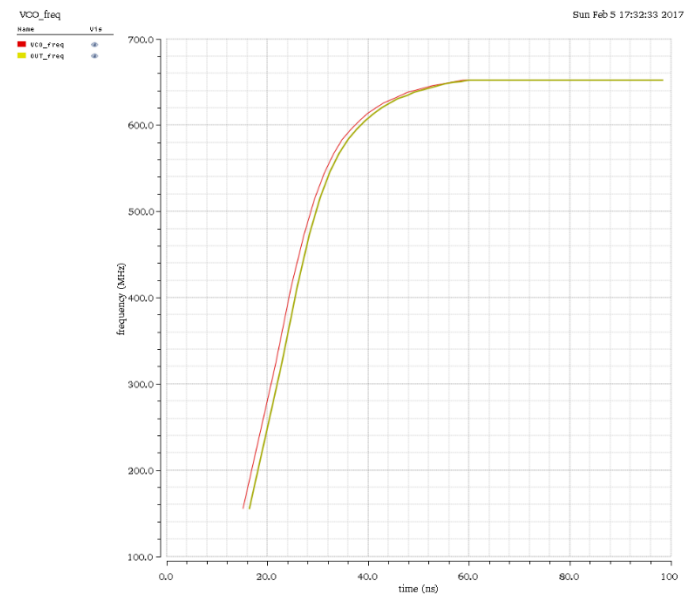


Figure 50: 7 stage VCO with 5 stage buffer

The tapered buffer simulations at 3.3V concluded with examining an 11 stage VCO with both the 4 and 5 stage buffer. Aside from frequency, the wave form behavior matches that of the 7 and 9 stage VCO.

VCO-BASED ADC

• • •

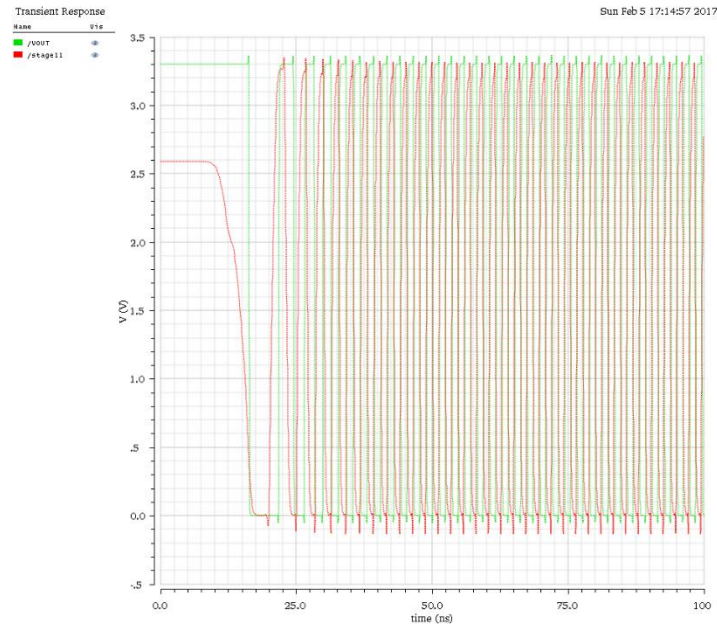


Figure 51: Waveforms of the 11 stage 3.3V after the VCO and after the tapered buffer

1V

The 1 V tests saw more of the same result, albeit at higher frequencies. The 13 stage VCO was also tested on due to the higher frequencies. Only a 5 stage buffer was used in these simulations due to its superior performance in the 3.3V VCOs.

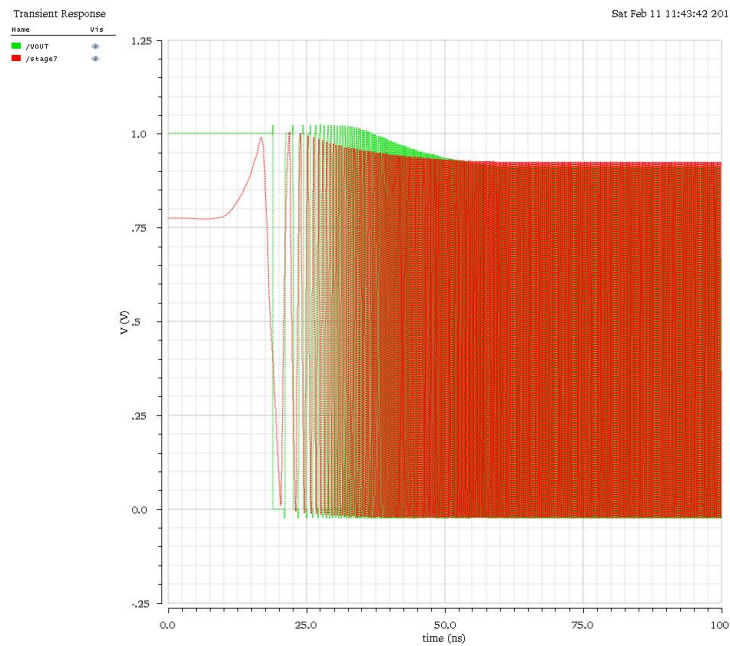


Figure 52: Waveforms of the 7 stage 1V after the VCO and after the tapered buffer

VCO-BASED ADC

• • •

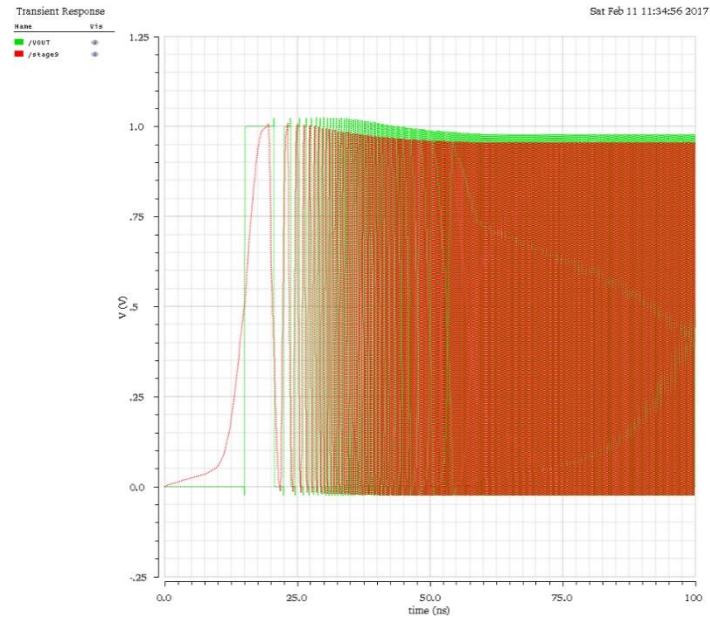


Figure 53: Waveforms of the 9 stage 1V after the VCO and after the tapered buffer

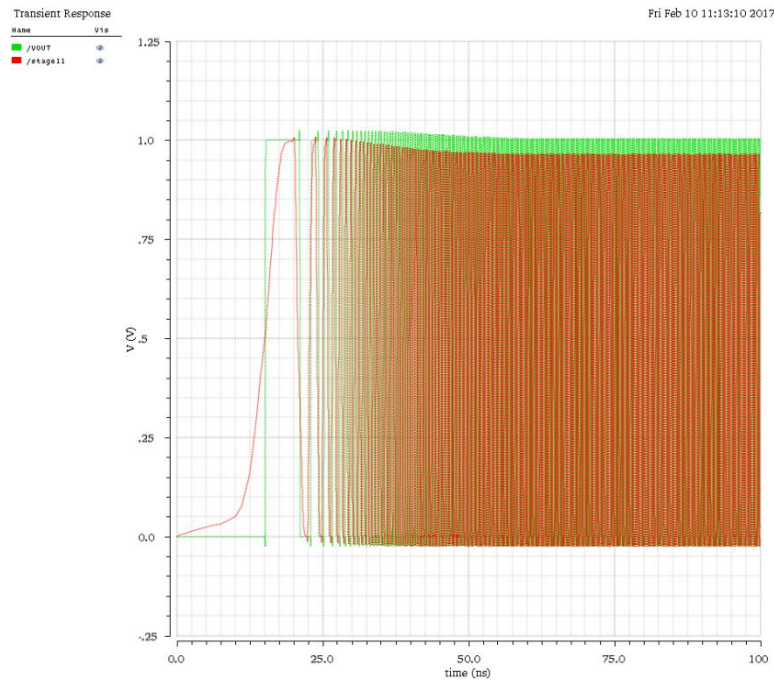


Figure 54: Waveforms of the 11 stage 1V after the VCO and after the tapered buffer

VCO-BASED ADC

• • •

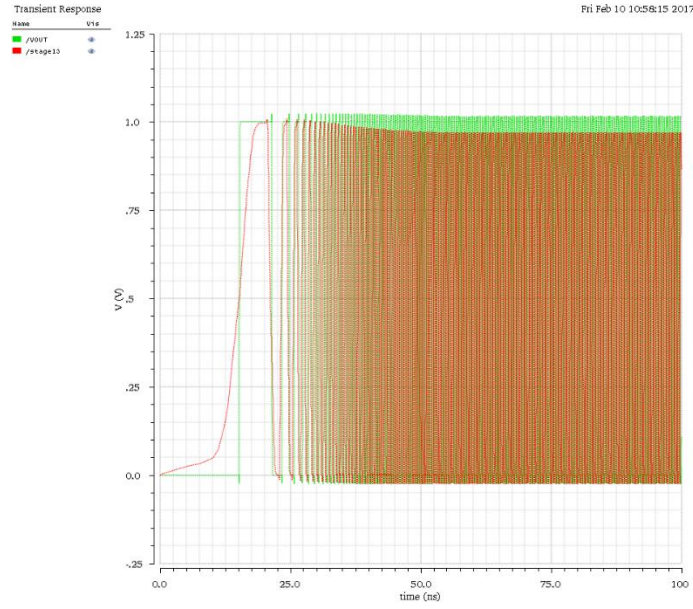


Figure 55: Waveforms of the 13 stage 1V after the VCO and after the tapered buffer

FFT Frequency Analysis with Matlab

The Fast Fourier Transform function in Matlab was used to check if the tapered buffer affected the frequency spectrum of the VCO. The 512 point default settings of the function ($Y = \text{fft}(X)$) was implemented. The actual scripts are available in Appendix E with comments.

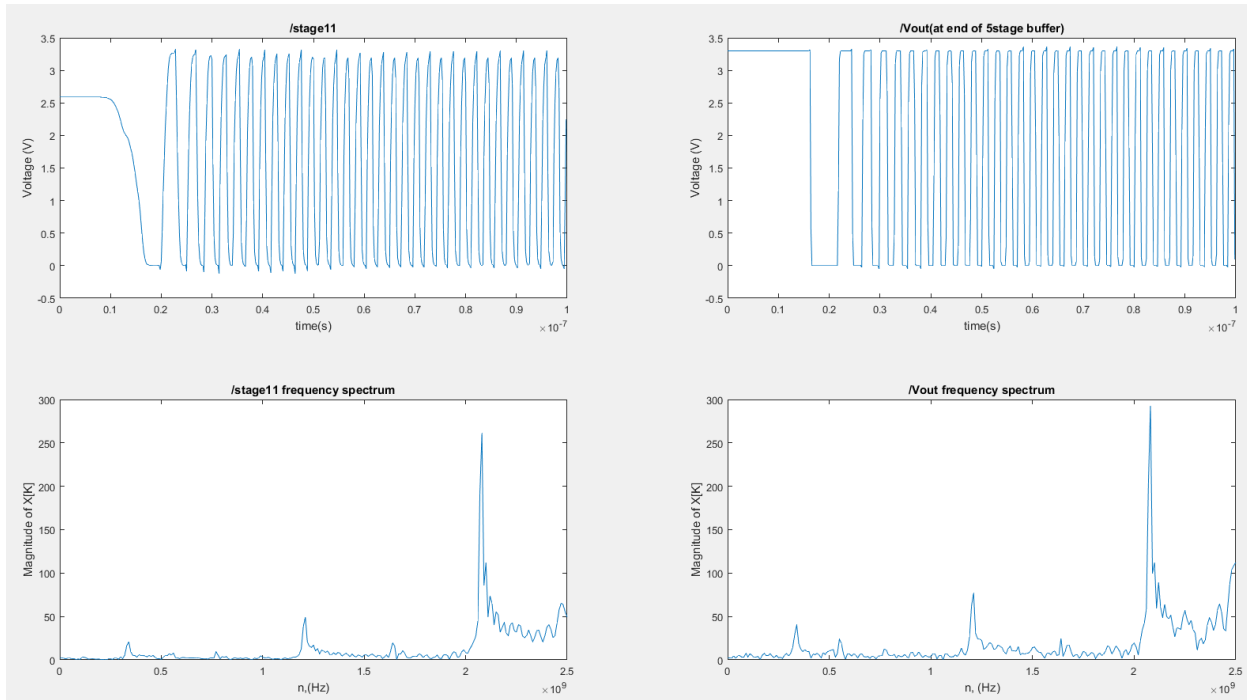


Figure 56: 3.3V, 11stage sampled at 5GHz (without buffer on left, with buffer on right)

• • •

The 3.3V, stage 11 was arbitrarily chosen first. The waveform data collected was sampled at 5GHz (a data point once every 2×10^{-10} seconds) and save in an excel file. A Matlab script saved in the same folder read the excel data. Overall, the frequency spectrum remained nearly enacted after passing through the 5 stage tapered buffer. The dominant frequency appears to be 2.1GHz with decent contributions from frequencies up to 2.5GHz and specifically at 1.25GHz

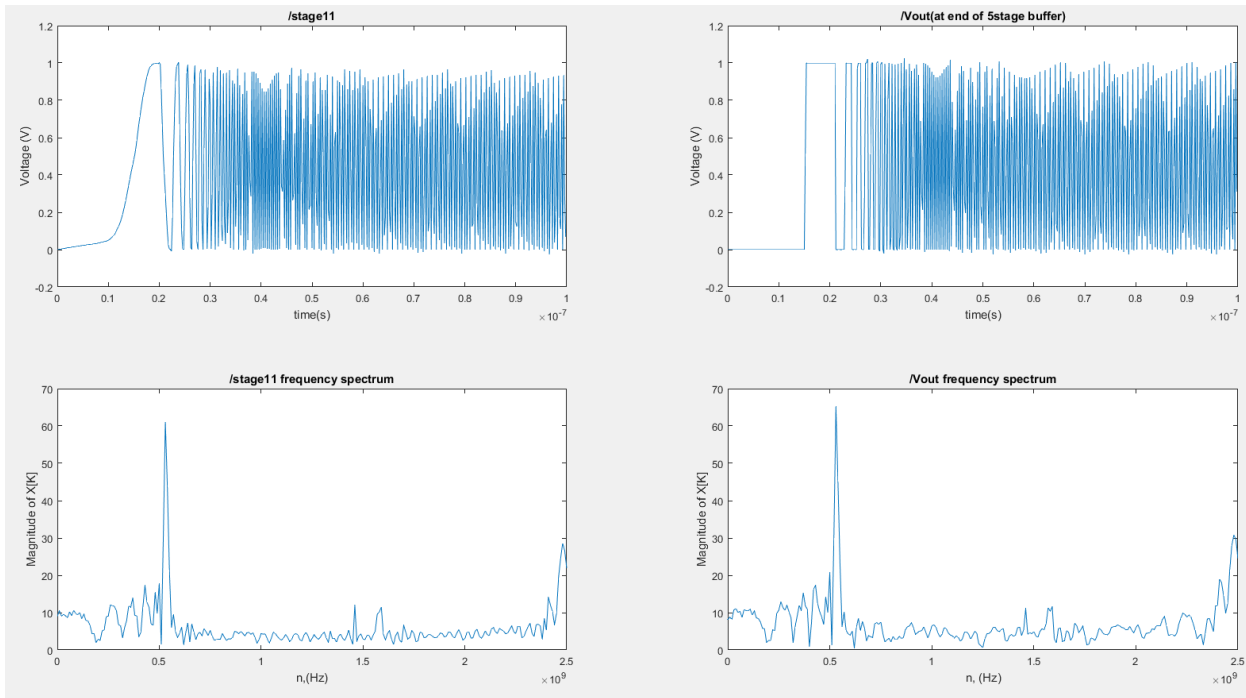


Figure 57: 1V, 11stage sampled at 5GHz (without buffer on left, with buffer on right)

The 1V, stage 11 was arbitrarily chosen next. Once again the waveform data collected was sampled at 5GHz (a data point once every 2×10^{-10} seconds) and save in another excel file. Another Matlab script was saved in the same folder to read its corresponding excel data. Again, the frequency spectrum remained nearly enacted after passing through the 5 stage tapered buffer.

However there is a case of aliasing in the frequency spectrum due to under sampling. The massive spike at around 0.5 GHz makes no sense at the dominant frequency. Since the actual waveform of the 1V VCO is oscillating faster than the 3.3V which has a stated 2.1GHz dominant frequency. The aliasing is somewhat visible by inspection of the reconstructed waveform plots (top 2 plots in Figure 58).

The waveform data collected was re-simulated and sampled at 10GHz (a data point once every 1×10^{-10} seconds). This way the Nyquist rate was increase from 2.5GHz to 5GHz. Another Matlab script was used to apply the Fast Fourier Transform to analyze the frequency spectrum. This time the dominant

• • •

frequency was at 2 GHz with meaning full contributions all the way up to 5 GHz. Again, by inspection of the reconstructed waveform plots, the aliasing problems from the previous frequency spectrum analysis seem to be eliminated by the faster resampling of waveform data.

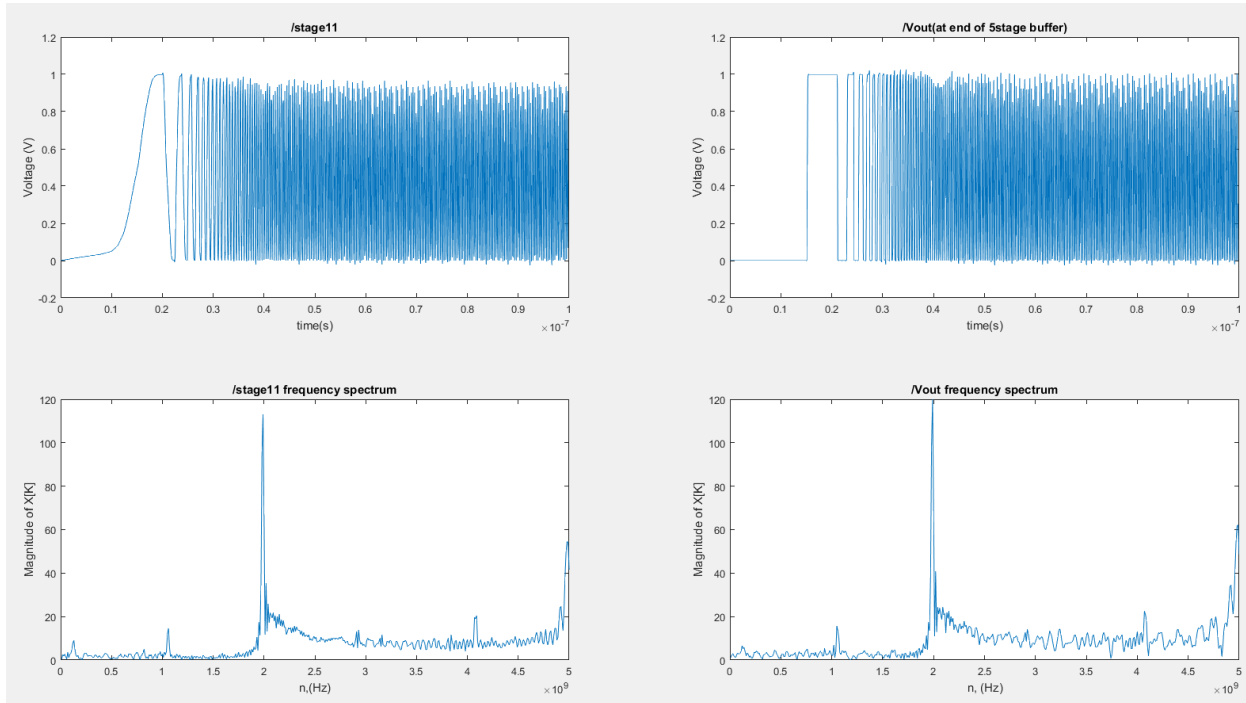


Figure 58: 1V, 11stage sampled at 10 GHz (without buffer on left, with buffer on right)

From these Matlab analysis, it became clear that the 5-stage buffer barely affected the frequency spectrum. This ultimately meant the tapered buffer was doing its job and minimally affecting what the signal the counter processed later. In other words, the capacitance affecting VCO oscillations had been dramatically reduced without introducing any new problems.

Block Diagram of Altered Design

However, our team's encounters with various challenge altered the overall project goal. Due to delays in the availability of the 65nm process technology and digital libraries, problems with ECE servers, vague design goals, and straying from main objectives several times, the design was not ready in time for fabrication. The design was altered to finish in the time allotted. Figure 59 displays the new design, where the VCOs are enabled via simulation only, so a Linear Shift Register is not necessary. We selected four VCOs for the purpose.

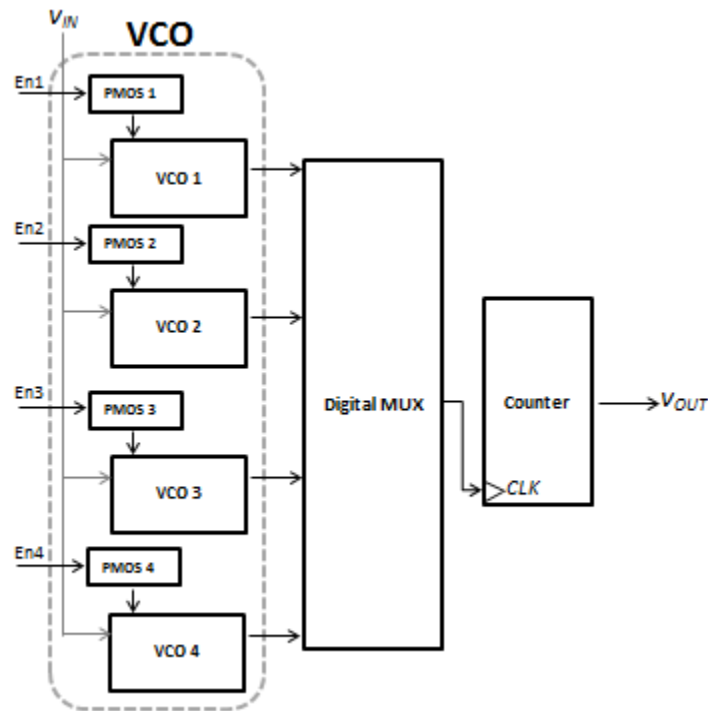


Figure 59: Block diagram of the revised design

Lower Frequency Simulations

Once we revised our project goal and reworked our block diagram, we decided to create between 3 and 4 VCOs that we would design and verify. Based on input from the graduate students regarding counter frequency capabilities given slack from Synopsys simulations, we determined target frequencies within the specifications up to 6GHz. The first we looked at was a seven stage inverter with varied current-starving PMOS and NMOS widths depicted in Figure 60. Here we selected the largest frequency range and defined the corresponding linear region.

VCO-BASED ADC

• • •

V-f characteristics of VCO with different widths of Current-starved MOSFETs
(VCO = 7 stage, VDD = 1 V, CSVCO with PMOS switch)

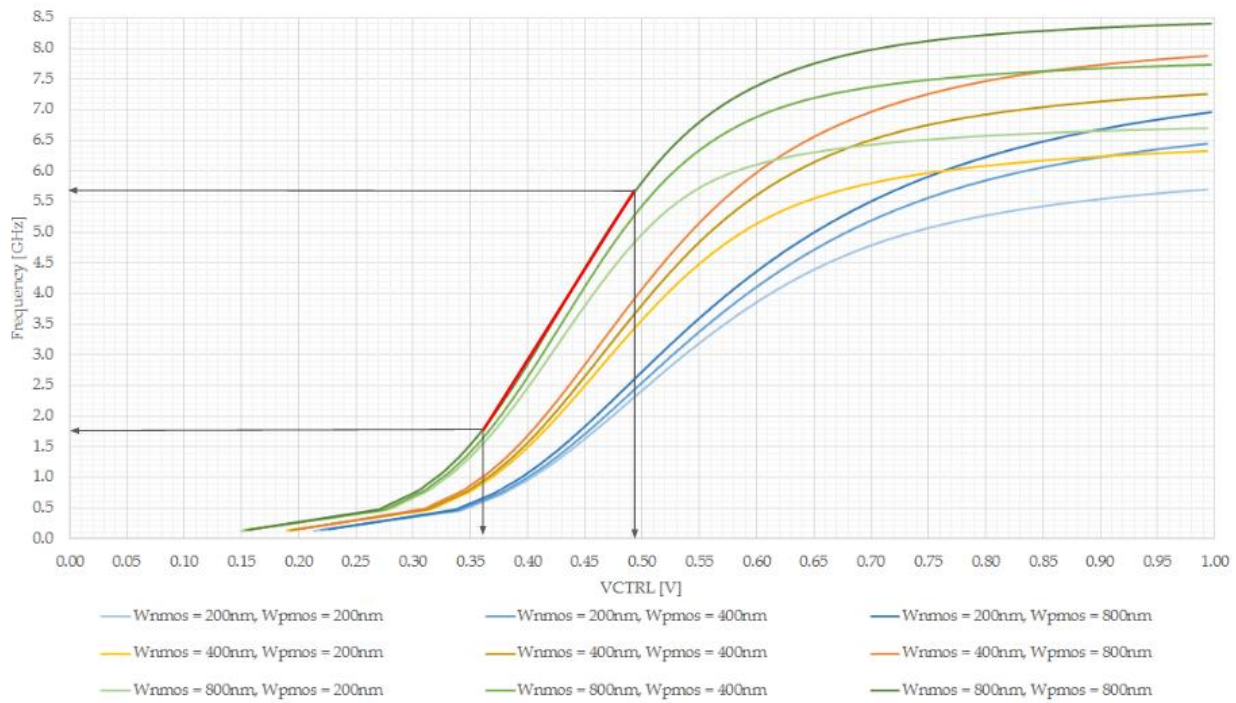


Figure 60: Determining Optimal Frequency Characteristics (Part I)

In the second case, we looked at the same waveform and selected the most linear and highest frequency waveform at a different voltage region as in Figure 61.

VCO-BASED ADC

• • •

V-f characteristics of VCO with different widths of Current-starved MOSFETs
(VCO = 7 stage, VDD = 1 V, CSVCO with PMOS switch)

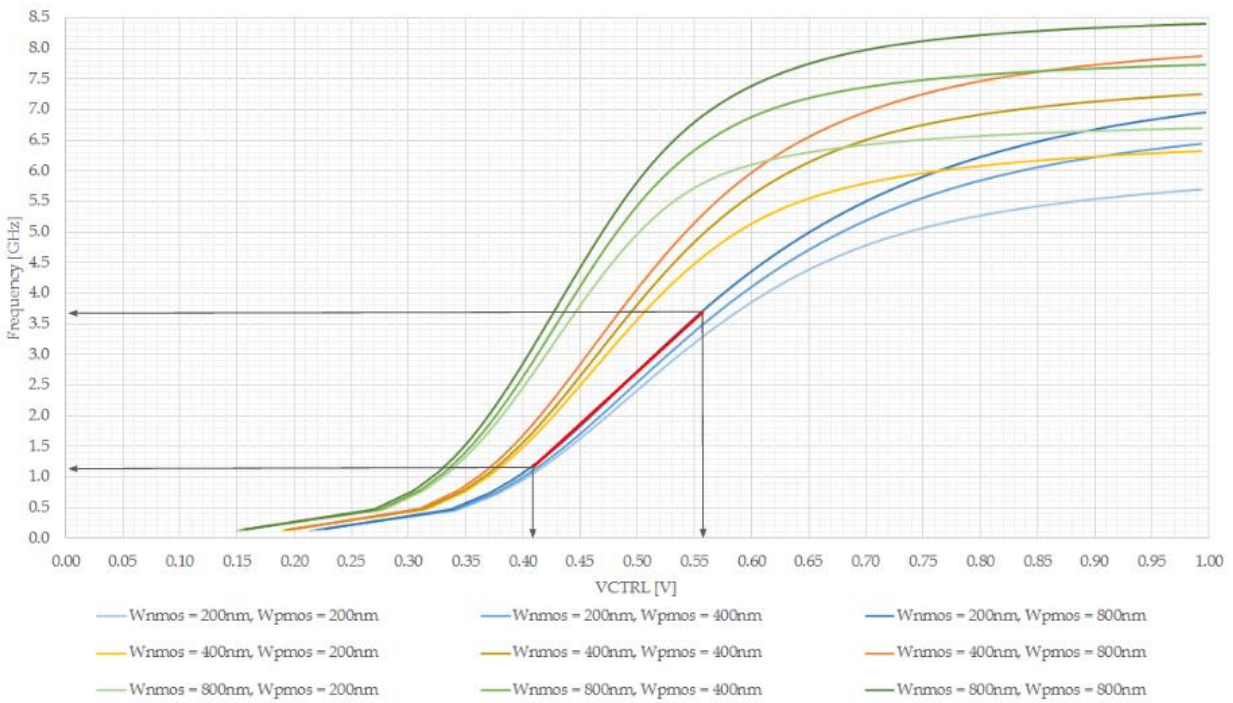


Figure 61: Determining Optimal Frequency Characteristics (Part II)

We repeated the procedure for the 11 stage 1V supply, skipping the 9 stage which could experience aliasing. Once again we looked at the most linear region with highest frequency as in Figure 62. The linear voltage region happened to coincide with our first selection.

VCO-BASED ADC

• • •

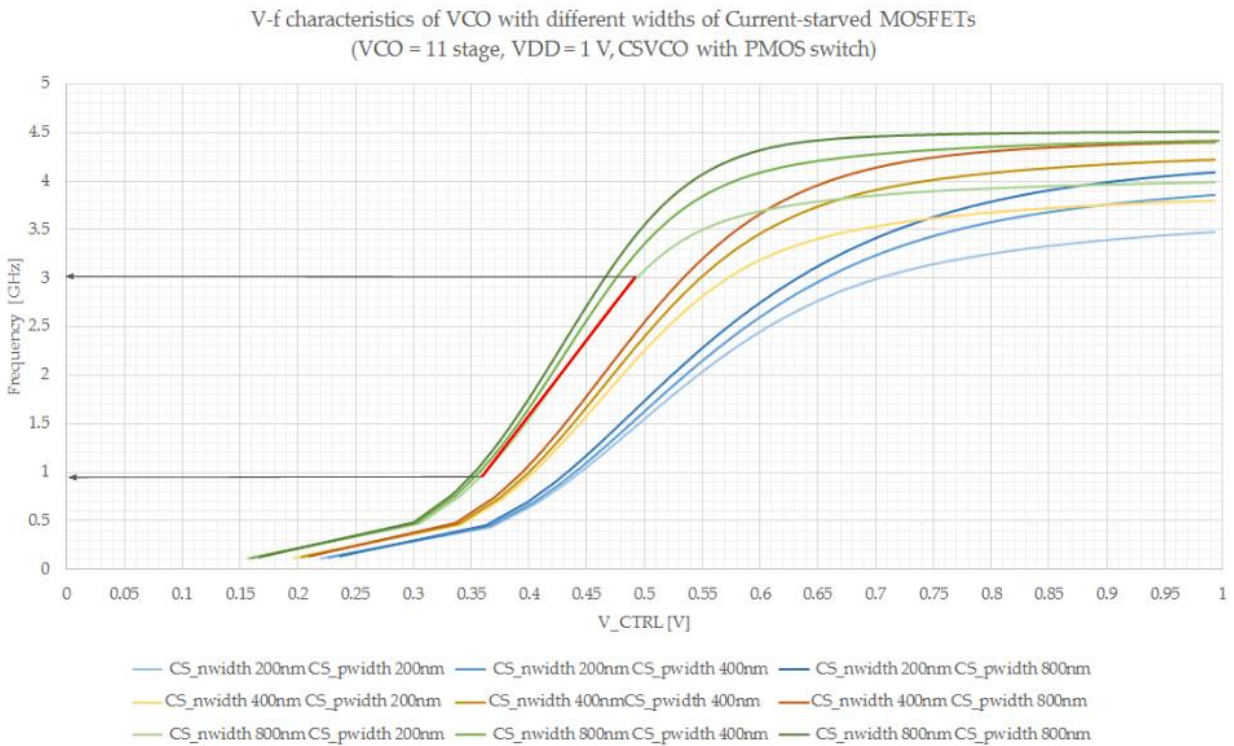


Figure 62: Determining Optimal Frequency Characteristics (Part III)

Finally, we looked at the linear region coinciding with the voltage range from our second selection, as in Figure 63. These lower frequency selections would provide lower power dissipation options at the cost of accuracy.

VCO-BASED ADC

• • •

V-f characteristics of VCO with different widths of Current-starved MOSFETs
(VCO = 11 stage, VDD = 1 V, CSVCO with PMOS switch)

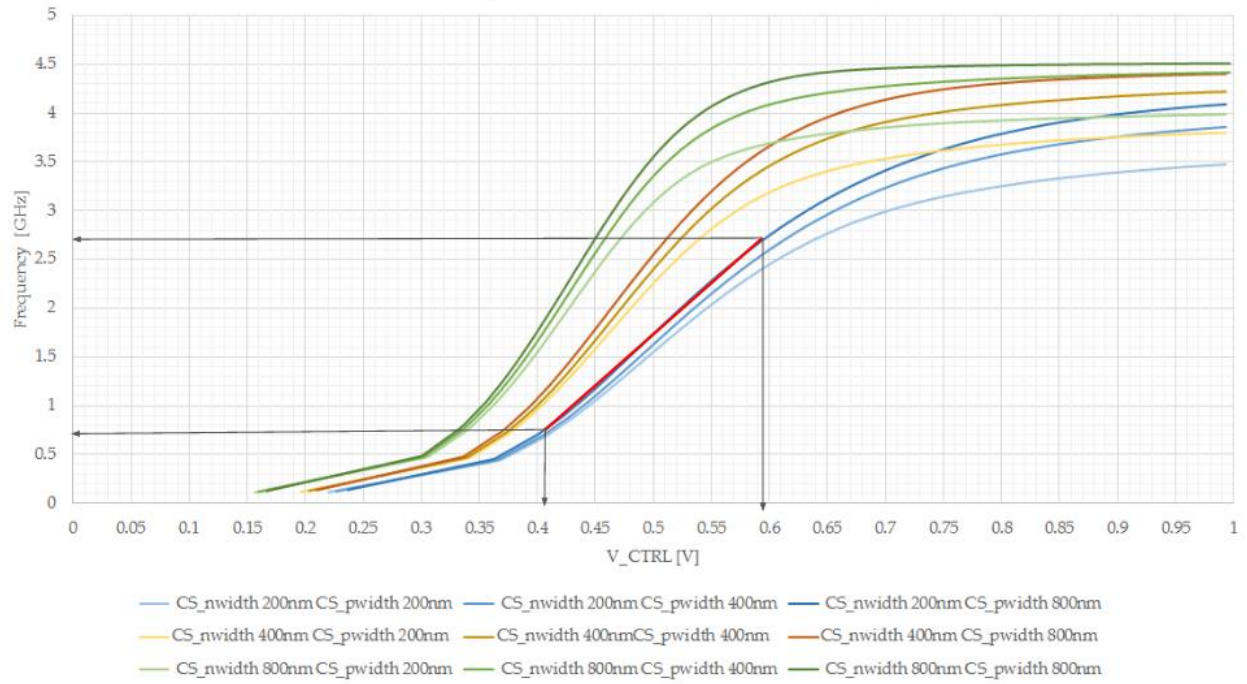


Figure 63: Determining Optimal Frequency Characteristics (Part IV)

Digital MUX and Counter

The analog MUX with tapered buffer works for the current design, however, from the simulation results, it produces the most desirable outcome when using 3.3 V supply MOSFETs. This diverges from initial goal of the project which was suggested by our project co-adviser of having multiple VCOs of different supply voltages connected to a single MUX and counter to be more efficient in die area and power consumption. According to graduate students who are helping with the project, the digital library can synthesize MUX and counter but can only work with 1V supply. After a series of discussion, the team decided to implement the MUX using digital library.

4-to-1 digital MUX and 12-bit ripple counter is created using the digital library of the relative process technology used for this project. By using Synopsys Design Vision and Cadence Encounter software, the digital components and their layout were designed and synthesized. The process of the creating these digital components can be briefly described as following steps (detailed step by step process is included in Appendix A):

- Creating Verilog code which represents desired the digital MUX and digital counter
- Using Synopsys Design Vision software to synthesize MUX and counter which can be translated into analog components of the process technology
- Importing the synthesized design into Cadence Encounter to create layout

Firstly, the digital MUX and digital counter were created using the Verilog hardware description language. The 4-to-1 MUX selects one of the four VCO inputs according to the user defined select which are off chip. The MUX is connected to a 12-bit ripple counter. Since the project aims to create an ADC of 10 ENOB, the output bits has to be more than 10. While including more output bits helps improving the design, trade-offs such as die area, speed, etc are inevitable. Once the Verilog codes are created, they are imported to Synopsys Design Vision software, which creates the digital MUX and counter using the actual components in the process technology digital library. The digital library contains digital contains already made digital components which can be translated to respective analog circuitry of the 65nm process parameter. The figures below show the schematic view of MUX and counter made from components from digital library of the 65nm process technology.

• • •

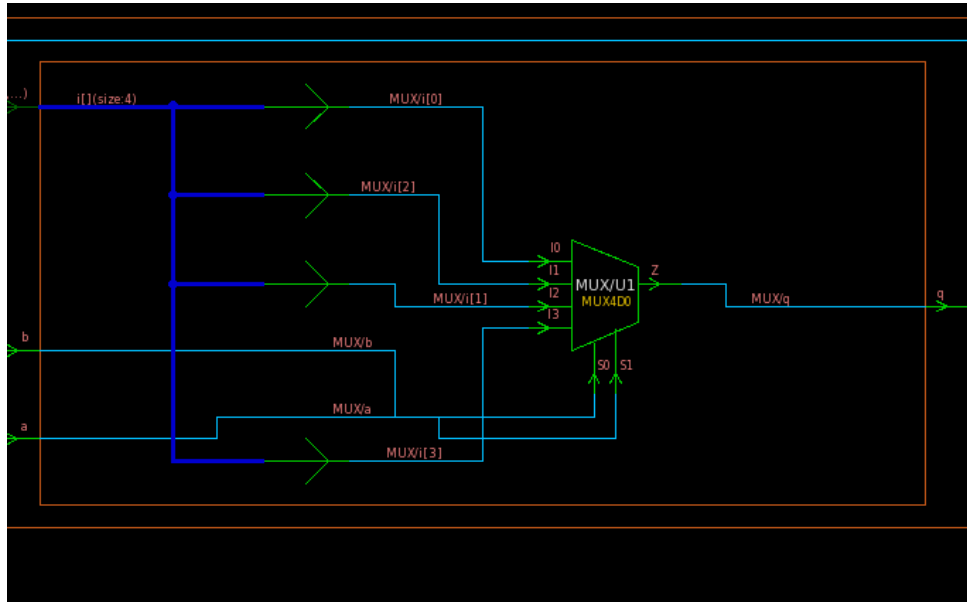


Figure 64: Schematic view of synthesized digital MUX

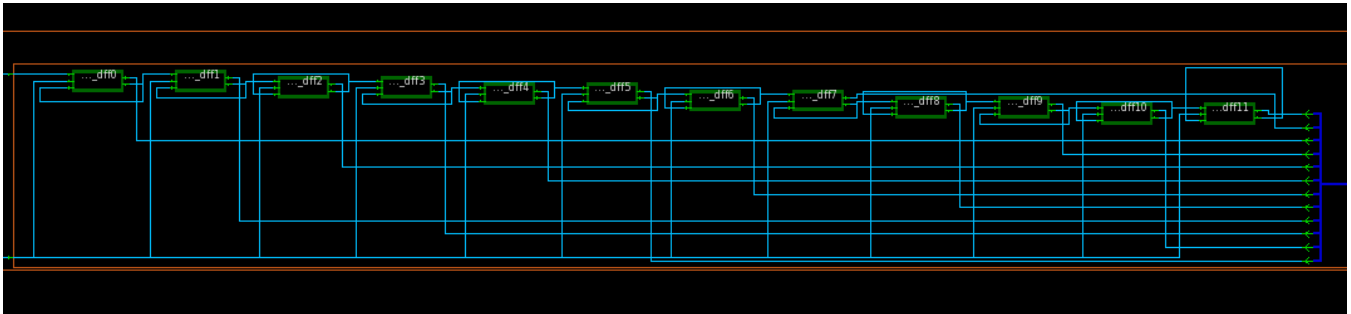


Figure 65: Schematic view of synthesized ripple counter

Next, the performance of this digital circuitry is analyzed, namely, timing (speed), power and area. These analysis can be done by typing in codes in Appendix D into the Synopsys command line.

Table 2: Timing measurement (slack)

input frequency [GHz]	slack
3.33	0.16
4	0.11
4.5	0.08
5	0.06
6	0.03

Timing measurement (slack)

The slack associated with each connection is the difference between the required time and the arrival time. A positive slack s at some node implies that the arrival time at that node may be increased by ns , without affecting the overall delay of the circuit. Conversely, negative slack implies that a path is too slow, and the path must be sped up (or the reference signal delayed) if the whole circuit is to work at the desired speed. [17]

The slack is used to determine the maximum input frequency that the counter can handle. Positive slack is desired and the larger the slack is, the better the performance. Above 6GHz input frequency, the slack becomes negative. Moreover, the simulation is considered as ideal scenario where real-world non-linearity are not included. Thus even though 6 GHz is regarded as upper limit, the frequency of the VCO output which is the input to the MUX should be less than 6 GHz.

The reports of timing, power and area are included in the Appendix D. The slack for different input frequencies are shown in Table 2. These are the same for all MUX inputs.

Layout and Final Design

Selection of Ring Oscillators

Out of the dozens of VCOs simulated, only four made it to the layout step. Pseudo-differential VCOs suffered too much non-linearity in their voltage to frequency relationships to be considered during the selection process, and time constraints did not allow us to investigate them further. The VCOs chosen for layout development exhibited either large input voltage ranges or large frequency ranges with differing power dissipation. The layouts created will ideally be fabricated in the future by another project group, or at least will provide guidance for other projects.

The selected VCOs were all 1V supply current-starved ring oscillators with minimal inverter PMOS and NMOS width (200nm) as well as a 9.6 μ m PMOS switch. They differed in the number of stages and the widths of the current-starving MOSFETs. The four VCOs selected were the following validated in Table 3.

Table 3: Selected VCO Characteristics and Validation

	CS PMOS Width	CS NMOS Width	Stages	Linear Input Range [V]	Frequency Range [GHz]	Validation
VCO1	800nm	800nm	7	0.36-0.49	1.75-5.65	Highest resolution voltage to frequency in voltage range
VCO2	800nm	200nm	7	0.41-0.56	1.10-3.70	Highest resolution voltage to frequency in voltage range
VCO3	200nm	800nm	11	0.36-0.49	0.95-3.00	Lower power dissipation of VCO1 input voltage range
VCO4	800nm	200nm	11	0.41-0.59	0.70-2.70	Lower power dissipation of VCO2 input voltage range

The input range of the chosen VCOs cumulatively covers a wide spectrum of the supply voltage; two of them behave linearly from 0.36V to 0.49V while the other two from 0.41V to 0.56V. The 11 stage VCOs cover similar voltage ranges to the 7 stage, but the range of frequencies is much lower for lower power dissipation. As a result, they suffer from lower accuracy and input range. Ultimately, these parameters were selected to improve the overall Walden figure of merit.

Layout of Ring Oscillators

Completing the layout process is integral to successful fabrication. The layout process includes fully passing Design Rule Check (DRC), Layout-Versus-Schematic (LVS), and Parasitic Extraction (PEX). Each of the four VCOs had five input-output pins, for instance VCO1 had: In1, En1, AGND, AVDD, and VCO1.

• • •

During the course of our project, we were able to layout all four of our VCOs and pass DRC and LVS. Unfortunately, the licensing of Calibre expired and we were delayed by addressing the issue with IT.

Figure 66 shows the layout of VCO1. Note that the PMOS switch is rotated and its gate length is $9.6\mu\text{m}$. Similarly, the inverter MOSFETs are 200nm width whereas the current-starving MOSFETs are 800nm . The connections are various metal drawing layers with minimal spacing for the smallest chip area. Only the bottom row of NMOS have integrated substrates.

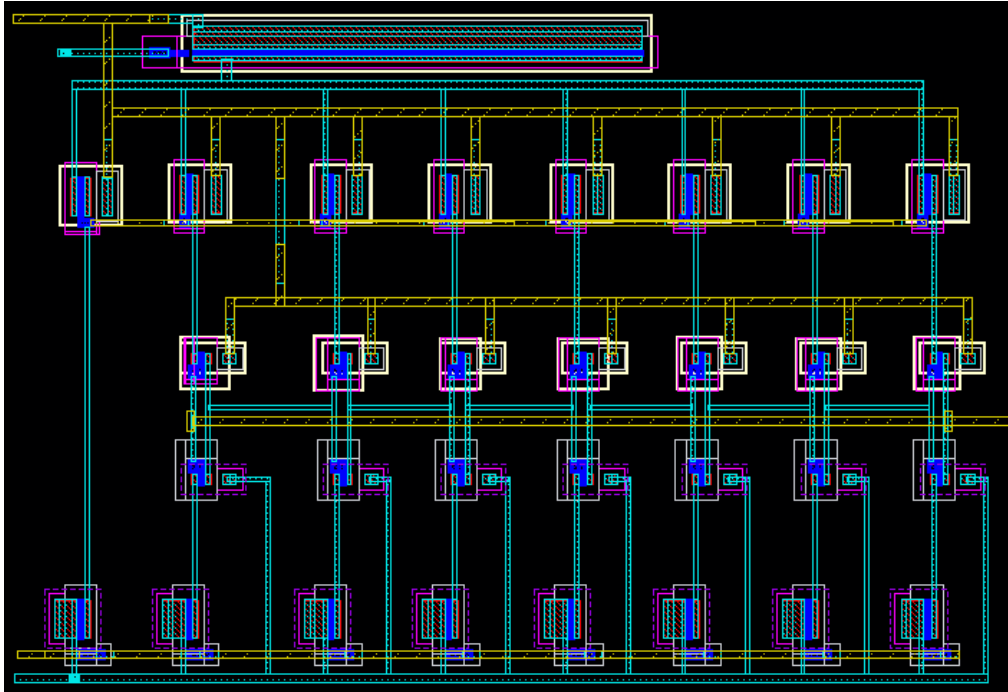


Figure 66: Layout of VCO1

Figure 67 shows the layout of VCO2. Once again the PMOS switch is rotated and its gate length is $9.6\mu\text{m}$. Similarly, the inverter MOSFETs and the current-starving NMOS are 200nm width whereas the current-starving PMOS are 800nm .

• • •

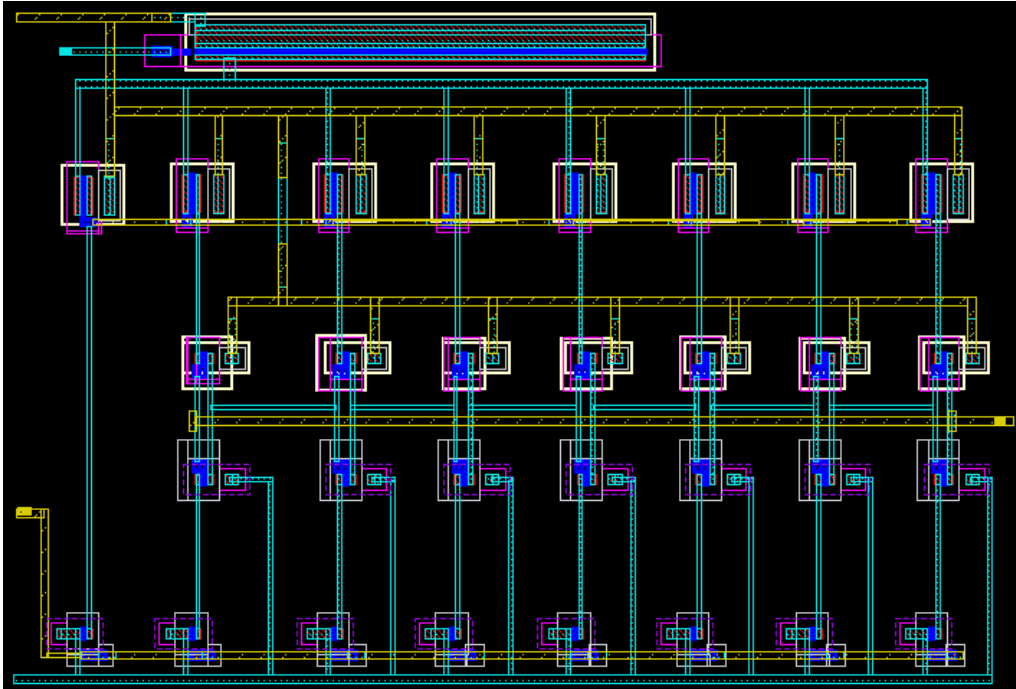


Figure 67: Layout of VCO2

The layout of VCO3 is presented in Figure 68. This rendition has 11 stages. The inverter MOSFETs and the current-starving PMOS are 200nm width whereas the current-starving NMOS are 800nm.

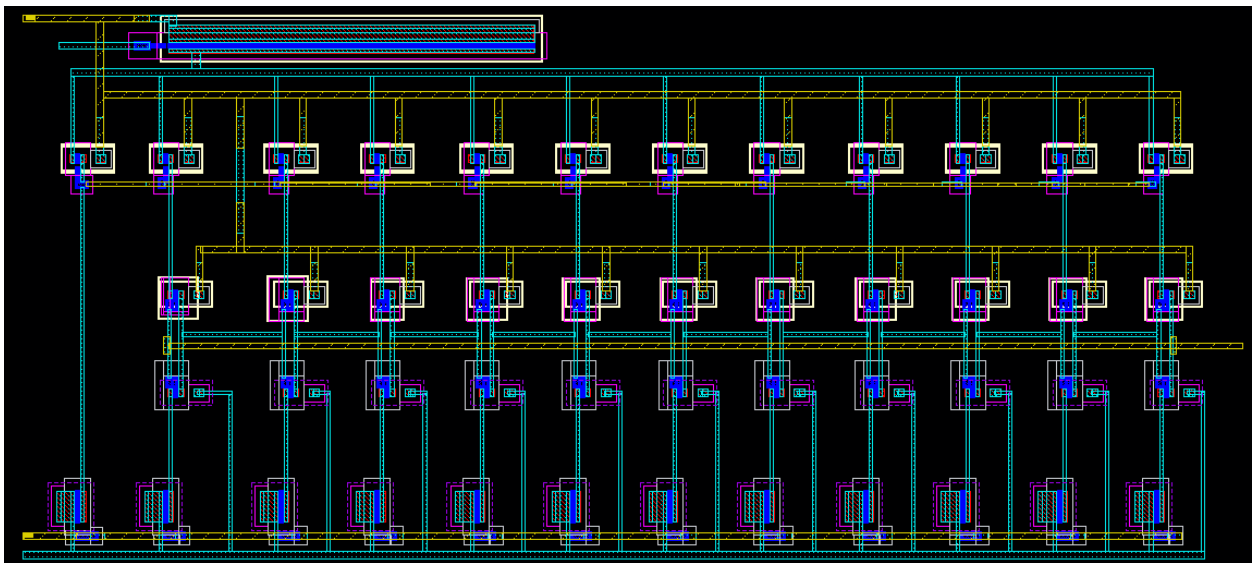


Figure 68: Layout of VCO3

Figure 69 depicts the layout of VCO4, also containing 11 stages. The inverter MOSFETs and the current-starving NMOS are 200nm width whereas the current-starving PMOS are 800nm.

VCO-BASED ADC

• • •

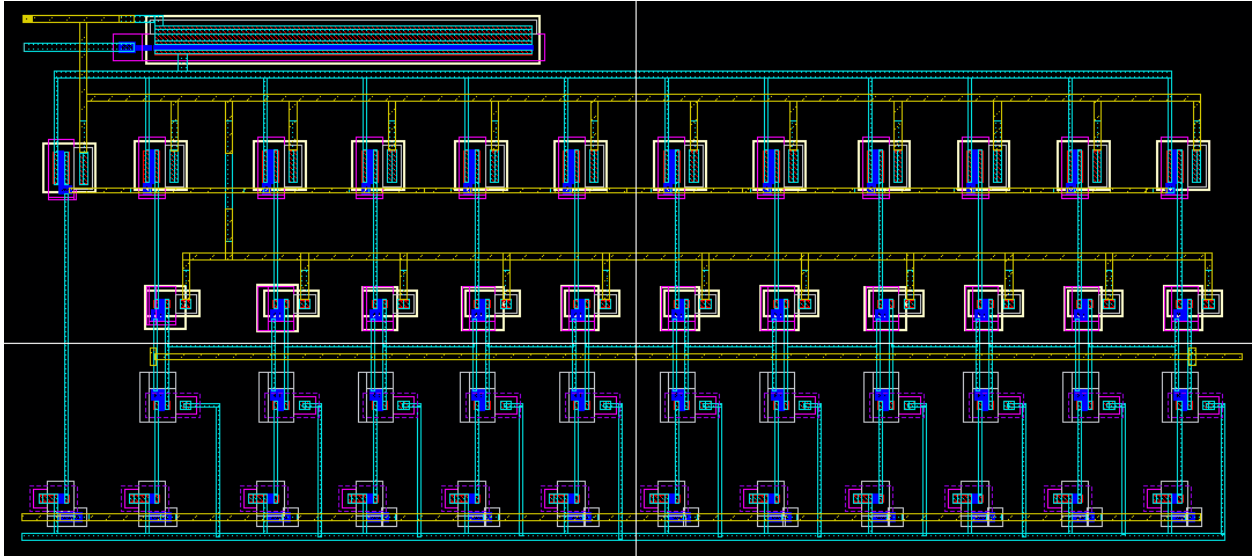


Figure 69: Layout of VCO4

Layout of MUX and Counter

The layout of the MUX and counter are created in Cadence Encounter. The synthesized design created from the Synopsys Design Vision is imported into Encounter. The detailed instructions on creating the layout is included in the Appendix A. The layout is modified until there are no DRC errors. Figure 70 displays the layout of the digital circuitry.

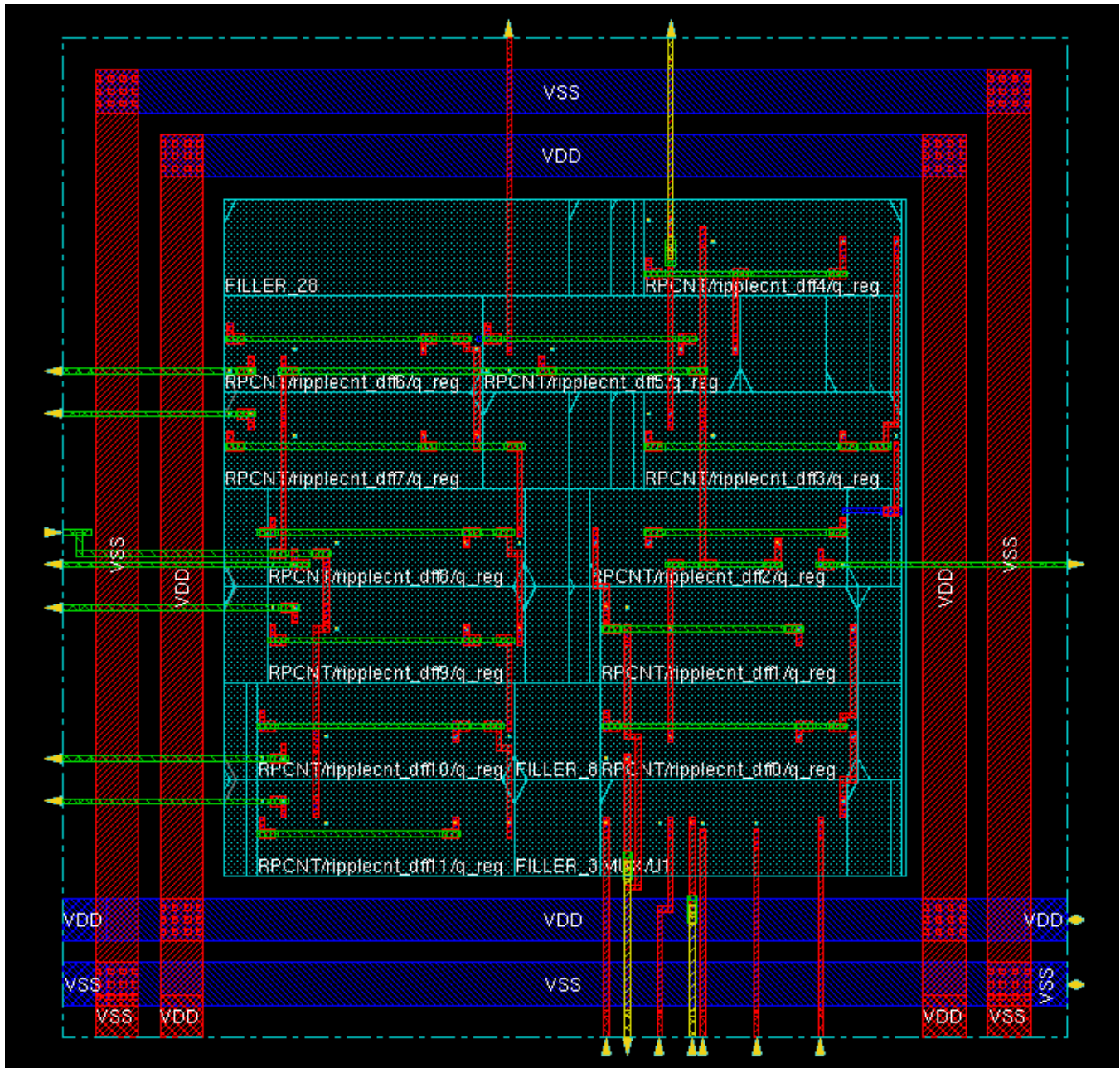


Figure 70: Layout of MUX and Counter

VCO-BASED ADC

• • •

```
encounter 1> *** Starting Verify DRC (MEM: 519.1) ***

VERIFY DRC ..... Starting Verification
VERIFY DRC ..... Initializing
VERIFY DRC ..... Deleting Existing Violations
VERIFY DRC ..... Creating Sub-Areas
VERIFY DRC ..... Using new threading
VERIFY DRC ..... Sub-Area : 1 of 1
VERIFY DRC ..... Sub-Area : 1 complete 0 Viols.

Verification Complete : 0 Viols.

*** End Verify DRC (CPU: 0:00:00.1 ELAPSED TIME: 0.00 MEM: 109.9M) ***
```

Figure 71: No DRC errors

Recommendations for Future Projects

Based on our own experience working with the current-starved VCO topology, we arrived at some crucial recommendations for future projects keen on fabricating VCO-Based ADCs. The data we collected on the VCOs provide better understanding of the trends and patterns associated with this type of ADC topology from a simulation standpoint.

1. Increasing the number of stages decreases frequency range
2. Lower supply voltage increases frequency range
3. Using PMOS switch 48 times larger width than inverter width
4. Increasing PMOS inverter width reduces frequency without improving linearity
5. Increasing the width of all PMOS decreases the frequency and improves linearity
6. Increasing the width of the current-starving MOSFETs simultaneously increases the frequency and introduces nonlinearity
7. Scaling current-starving PMOS and NMOS independently helps optimize frequency characteristic

Regarding our project alone, we would like to see our designs make it to post-layout simulation, co-simulation, and fabrication, and eventually through testing. Providing you with these resources would be immensely difficult, however, as there are non-disclosure agreements and many other pitfalls, but we will try nonetheless.

We also urge that for your own integrated circuit design project, you follow our advice from experience. First of all, start planning early and secure a technology library and a project topic prior to the end of D term before MQP so you can start doing research right away. Additionally, read our *Young Designer's How-To Guide (WPI Edition)* found in Appendix A. It outlines a lot of our struggles, the things to look out for, and provides a tutorial for many things we experienced firsthand throughout our project. Make sure to sit down with your advisor and discuss all of these details extensively (if the primary advisor is going away on sabbatical, do yourself a favor and choose a different project).

References

- [1] Kester, Walter Allan, ed. *Data conversion handbook*. Newnes, 2005.
- [2] "Digital Counters." *Learn about Electronics*. N.p., n.d. Web. 20 Oct. 2016.
- [3] McNeill, John, Rabeeh Majidi, Jianping Gong, and Chengxin Liu. "Lookup-table-based background linearization for VCO-based ADCs." *Proceedings of the IEEE 2014 Custom Integrated Circuits Conference*. IEEE, 2014.
- [4] Allen, Phillip E. Holberg, Douglas R.. *CMOS Analog Circuit Design* (3rd Edition). Oxford University Press, 2012.
- [5] Suman, Shruti, K. G. Sharma, and P. K. Ghosh. "Analysis and Design of Current Starved Ring VCO." *ICEEOT*, 2016.
- [6] "Ripple Counter - Basic Digital Electronics Course." *Electronincs-course.com*. N.p., n.d. Web. 14 Oct. 2016.
- [7] "Synchronous Counters : Sequential Circuits." *All About Circuits*. N.p., n.d. Web. 14 Oct. 2016.
- [8] Interfacing FPGAs to an ADC Converter's Digital Data Output. Tech. Analog Devices, n.d. Web. 14 Oct. 2016.
- [9] What Is an FPGA?" *Embedded Micro Make Technology*. N.p., n.d. Web. 14 Oct. 2016.
- [10] R. H. Walden, "Analog-to -digital converter survey and analysis" , *IEEE Journal on Selected Areas in Communication*, April 1999.
- [11] Pham, Long, and John McNeill. "Improved lookup-table-based algorithm for background linearization of VCO-based ADCs." *Ph. D. Research in Microelectronics and Electronics (PRIME), 2015 11th Conference on*. IEEE, 2015.
- [12] Kulkarni, M., and K. N. Hosur. "Design of a linear and wide range current starved voltage controlled oscillator for PLL." *International Journal on Cybernetics & Informatics* 2.1, 2013.
- [13] Pelgrom, Marcel JM. *Analog-to-digital conversion*. Springer Netherlands, 2010.
- [14] Ohnhäuser, Frank. *Analog-Digital Converters for Industrial Applications Including an Introduction to Digital-Analog Converters*. Springer, 2015.
- [15] McNeill, John A., and David Ricketts. *The Designer's Guide to Jitter in Ring Oscillators*. Springer Science & Business Media, 2009.
- [16] Vahid, Frank. *Digital Design*. Hoboken, NJ: J. Wiley & Sons, 2007. Print.
- [17] Hitchcock, Robert B., Gordon L. Smith, and David D. Cheng. "Timing analysis of computer hardware." *IBM journal of Research and Development* 26.1 (1982): 100-105.

Appendices

APPENDIX A: The Young Designer's How-To Guide (WPI Edition)

An IC application MQP is more than a little overwhelming, so after all of the troubles we faced during our project, we decided to put together a little guide to help direct posterity in the right direction. Just so you know, none of the classes or internships you had so far prepared you for this adventure, so prepare for an exciting project with a lot of disappointments, setbacks, and hard work.

So if you're looking for an easy MQP, choose anything else. If you want an MQP to cumulate your WPI coursework, look elsewhere. If you want the MQP to provide you with a skill set desirable to employers, you probably still want to look elsewhere as three terms are not enough time to impress everyone with your accomplishments. This experience is most useful, based on our experience, if you wish to pursue a PhD in analog layout, perhaps if you desire to continue at WPI, but if you want to attempt to prove us wrong, we hope this guide helps propel you in the right direction.

The Basic Truths:

1. Cadence is an awful software
2. Ask for help early, and often
3. Everything is documented somewhere
4. Fundamentals that no one remembers to tell you: Analog Layout
5. Fundamentals that no one remembers to tell you : Digital Layout and Synopsys
6. Fundamentals that no one remembers to tell you : Cosimulation

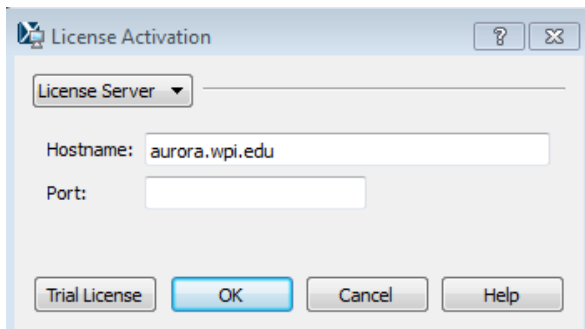
With that in mind, I recommend you read the first three chapters before you begin using Cadence, and definitely Chapters 2 and 3. They include things you really need to know. The other chapters are useful as you work through analog and digital layout.

• • •

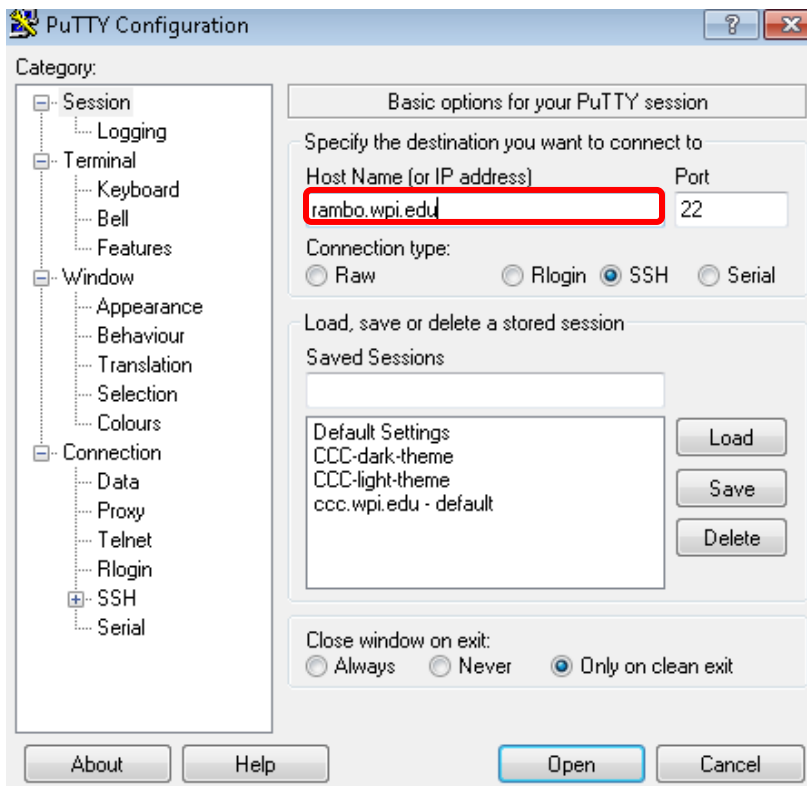
Chapter 1: Cadence is an Awful Software

Most people you ask, and as you delve deeper into Cadence, you will find how many limitations the software has. Unfortunately, it is an industry standard, and in spite of its many problems, it is just about the best software for the job.

At WPI, the best server to use is Rambo, but unfortunately, it is accessed via Linux. Thus, accessing it is hard. To start Rambo, first you need to open X-Win32, any year. If there is a licensing error, change to license server and type in `aurora.wpi.edu`.

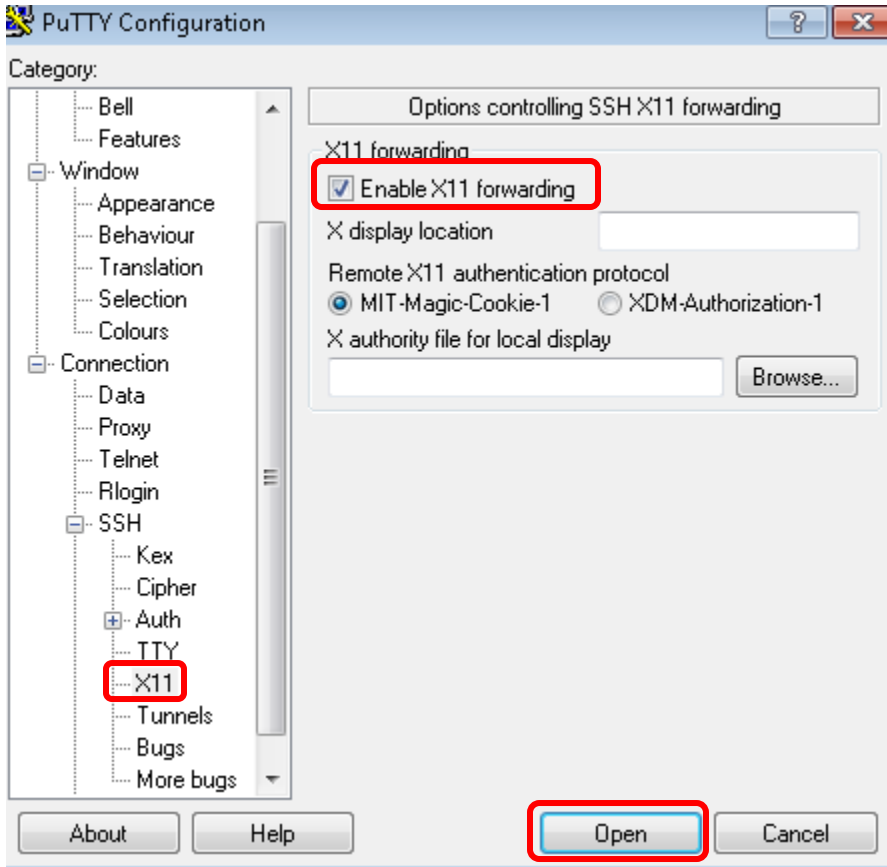


Then open PuTTY, which directs you to PuTTY configuration. For Host Name, type in `rambo.wpi.edu`. The port does not particularly matter.



Then press *Connection* -> *SSH* -> *X11*. Be sure to enable X11 forwarding, as indicated below.

...



After pressing Open, the Rambo.wpi.edu - PuTTY terminal window will open. Log in using your WPI credentials, without @wpi.edu.

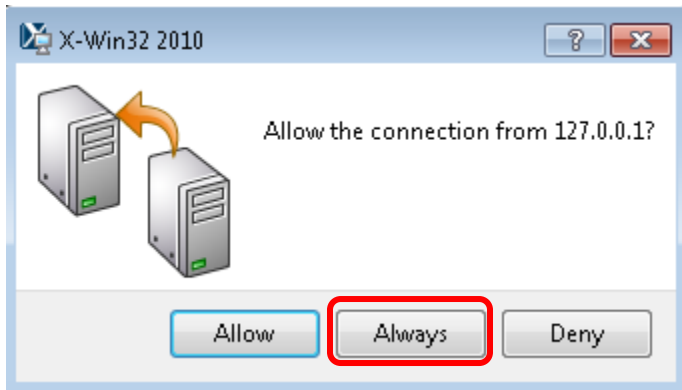
After a welcome note, you will see `-bash-4.1$`, after which you can type commands. Whenever you want to see the details of the directory you are in, type `ls`. The first step is to make a new directory: `mkdir cadence_mqp`. To enter the new directory (here named `cadence_mqp`), `cd cadence_mqp`. Now you are in the new directory where you will work on all things Cadence (for now). To open Cadence Virtuoso, the most essential Cadence software for schematic simulations and layouts, there are a few things you need to do. Type in `module load cadence`, press Enter, then type in `virtuoso &`. **The & (ampersand) is very useful in the terminal, as it lets you type things into the command line while a program is running.**

```
-bash-4.1$ cd cadence_mqp
-bash-4.1$ module load cadence
-bash-4.1$ virtuoso &
[1] 26265
```

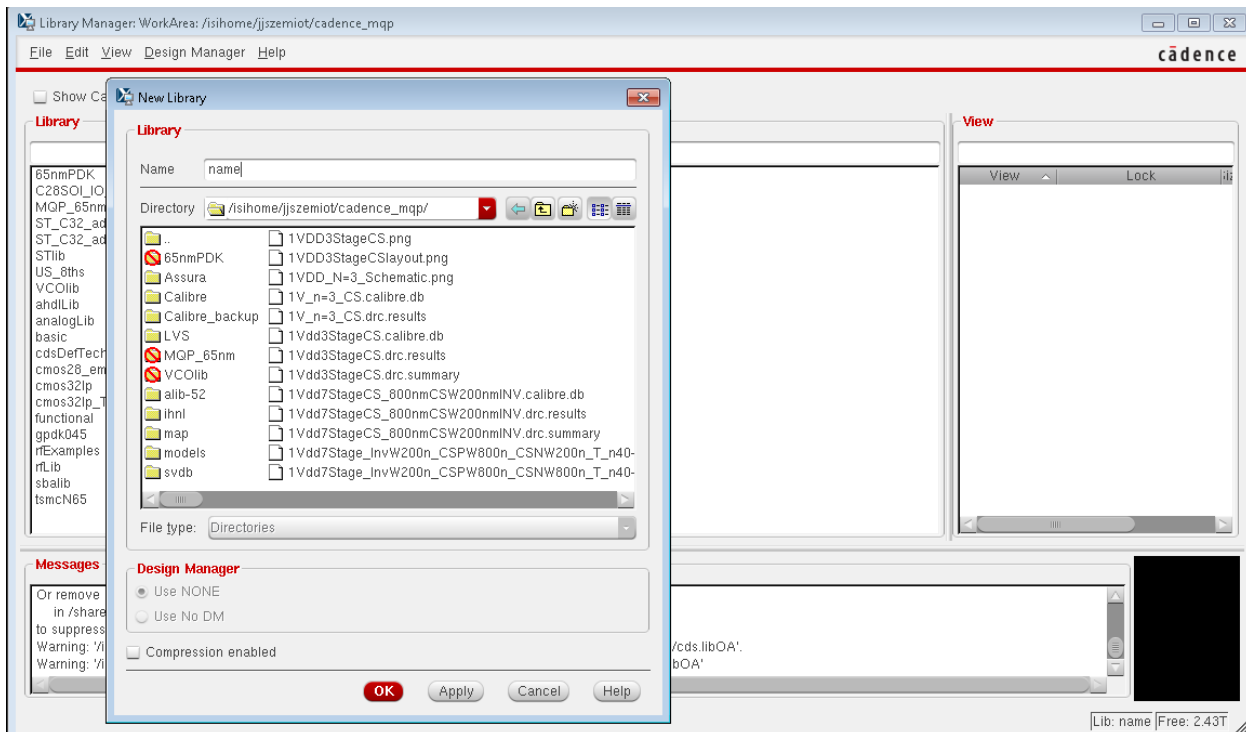
There are other useful terminal commands, such as `find` and `rm`. Use `find` to locate a file based on name, whereas `rm` deletes a file or folder.

• • •

This command opens two windows, Virtuoso ® and What's New Overview. Feel free to close What's New Overview. Closing Virtuoso ® closes Cadence entirely. If you have an option window open, select always.

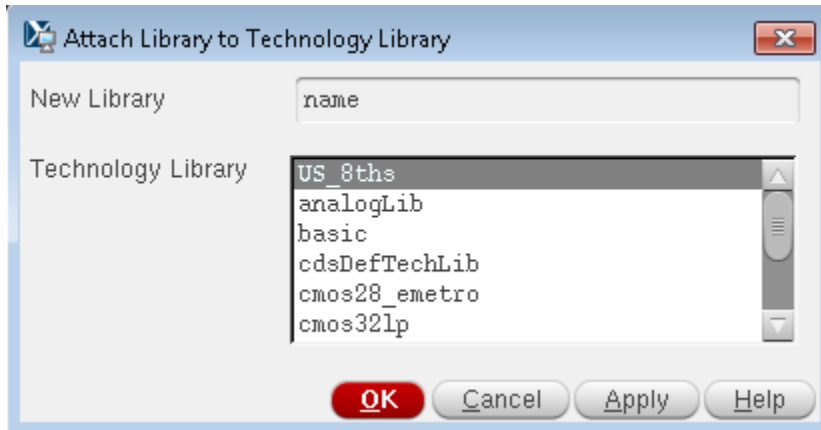


The next step is opening the library manager, accessed through *Tools -> Library Manager*. The generic process development kit (PDK) is in gpdk045, which has components that can be simulated, but cannot be fabricated. While waiting for your technology to become available, experiment with this one. **Before you begin making a circuit schematic, you need to make a new library and attach it to an existing technology.** To do this, press *File -> New Library* from the Library Manager. Give it a name, press OK, then attach to an existing library.

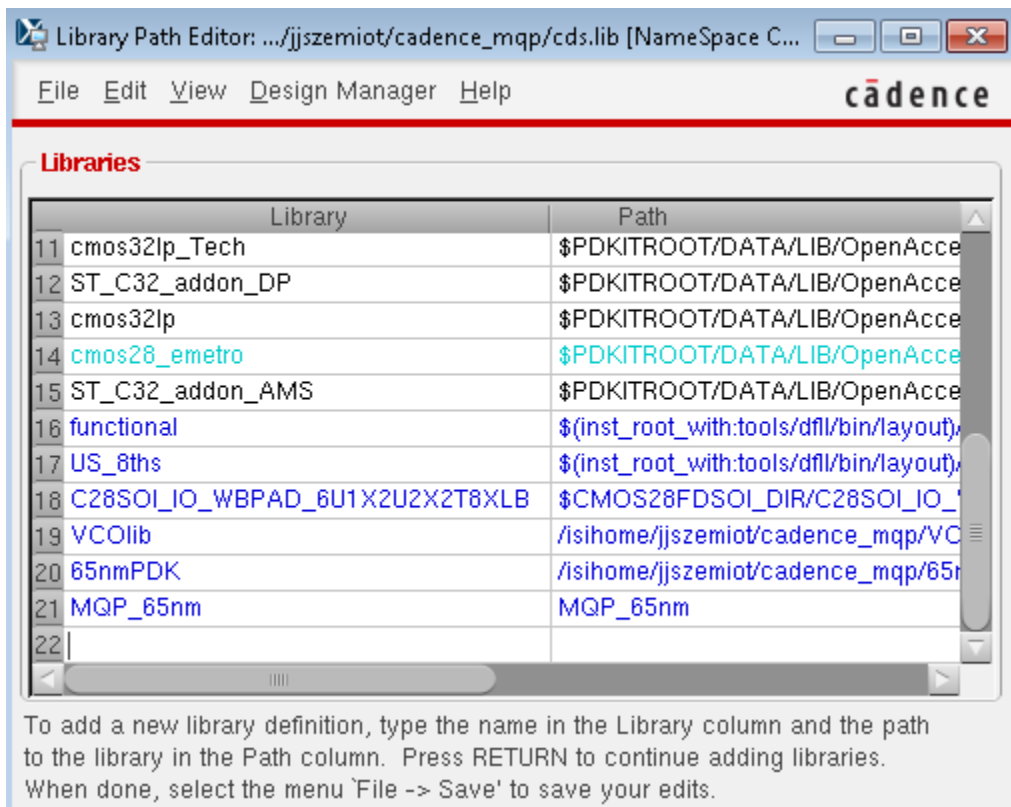


• • •

Find out the name of the existing technology library from your adviser, the PhD students, or whoever else may know. Bottom line, you need to ask. It is vitally important. If it is a new technology, wherein you will need to sign a non-disclosure agreement like we did, you will need to have someone install the PDK onto your user first. Keep in mind, the PDK is comprised of an analog library and a digital library. More on digital later.



When you have a library that does not show up in Library Manager, return to the main Virtuoso window and select *Tools -> Library Path Editor*. Follow the instructions below, namely to provide a library name and the location. Make sure to save it before closing.

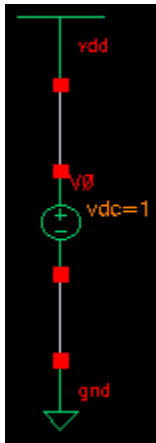


VCO-BASED ADC

• • •

To start creating a schematic, select *File -> New -> Cell View*. Identify the name (not all names are valid, such as those including *(.)* or other characters) and the view (schematic).

All of the power supplies and ground references can be found in *analogLib*. When building a schematic, use *vdd* as the supply rail. In order to define the supply, you must connect it to a DC voltage (*vdc*) with a ground reference (*gnd*).



To speed up the simulation process when sweeping a voltage, use *vpwl* to map an initial voltage and a final voltage to a particular time. This results in a voltage that is linear with time, that can be easily calculated using the slope and the given time. For instance, set the voltage 1 to 0 V at 0 s (*time 1*) and voltage 2 to 1 V at 50 ns (*time 2*) as displayed in the properties of the component.

VCO-BASED ADC

• • •

The screenshot shows the 'Edit Object Properties' dialog box for a VCO-based ADC component. The dialog is organized into several sections:

- Apply To:** Two dropdown menus set to 'only current' and 'instance'.
- Show:** Checkboxes for 'system' (unchecked), 'user' (checked), and 'CDF' (checked).
- Buttons:** 'Browse' and 'Reset Instance Labels Display'.
- Property Table:**

Property	Value	Display
Library Name	analogLib	off
Cell Name	vpwl	off
View Name	symbol	off
Instance Name	V1	off
- User Property Section:**
 - Buttons: 'Add', 'Delete', 'Modify'.
 - Table:

User Property	Master Value	Local Value	Display
Ivsignore	TRUE		off
- CDF Parameter Table:**

CDF Parameter	Value	Display
Frequency name for 1/period		off
Number of pairs of points	2	off
Time 1	0 s	off
Voltage 1	0 V	off
Time 2	50n s	off
Voltage 2	1 V	off
Noise file name		off
Number of noise/freq pairs	0	off
DC voltage		off
AC magnitude		off
- Bottom Buttons:** 'OK', 'Cancel', 'Apply', 'Defaults', 'Previous', 'Next', 'Help'.

To easily access the properties, press q after selecting the component. For more shortcuts, keep reading.



Useful Shortcuts

There are a number of other useful shortcuts in Cadence. In schematic view, these include:

w	creates a wire
i	add instance
r	rotate
o	display options
p	add pin
f	fit screen
l	add wire name
c	point to copy
left click	select
right click	zoom in on area

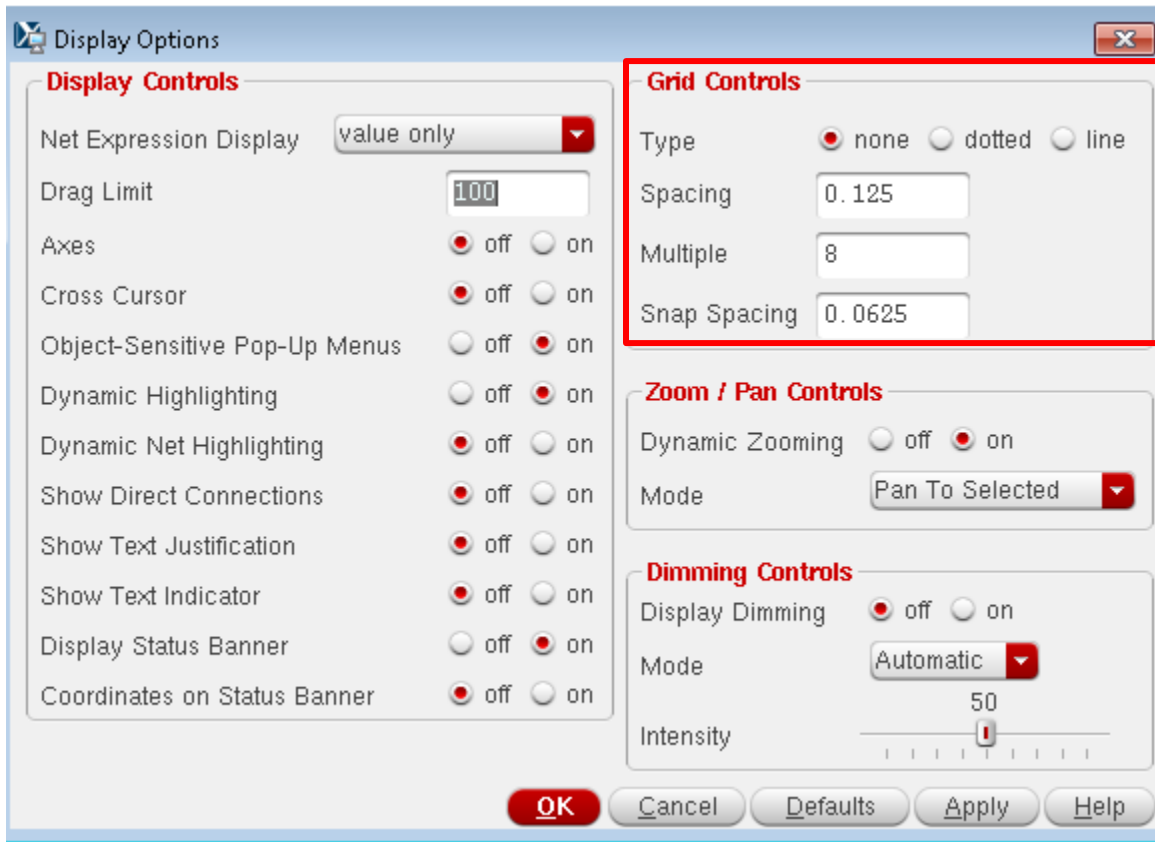
Layout view has quite a few others, including:

q	edit instance properties
e	display options
r	draw rectangle layer
i	create instance
s	stretch layer (do not highlight layer first)
f	fit screen
h / c	copy (do not highlight first unless copying multiple)
k	create ruler
shift+f	reveal hidden
shift+k	remove all rulers

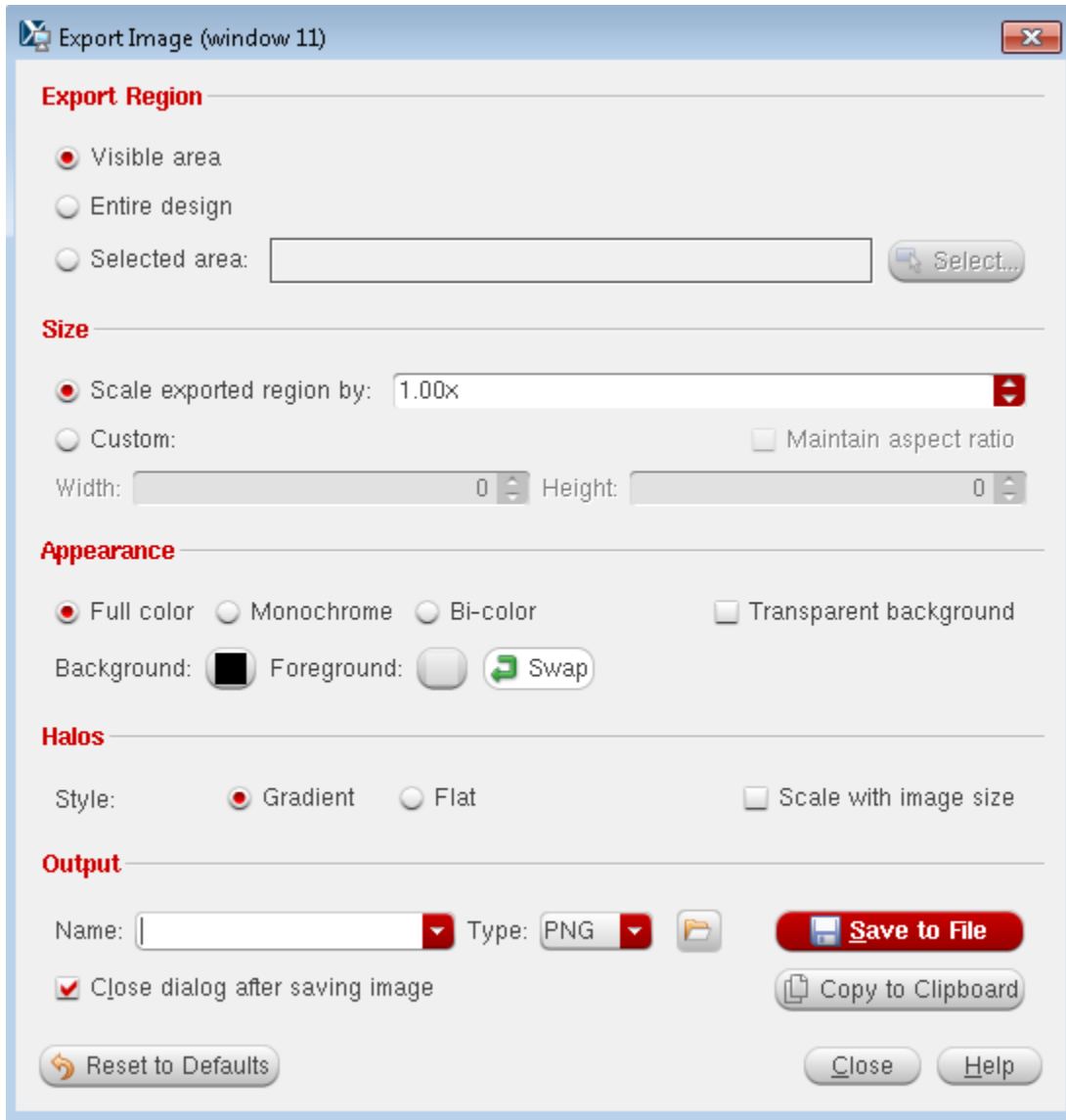
• • •

Saving an Image

In order to save images without the grid, press o, then select under *Grid Controls*, *Type none*.



The grid is gone. Sometimes it is also useful to convert the background from black to white. There is no easy way to achieve this using snipping tool. Instead, press *File -> Export Image*. The following window appears. You can select a portion of the circuit to save or the entirety, scale the size (recommendation is to save at 1x, zoom in on schematic), and change the schematic appearance. Full color maintains the color scheme, but swapping background from black to white sometimes or making it transparent is useful for papers and presentations for better visibility.

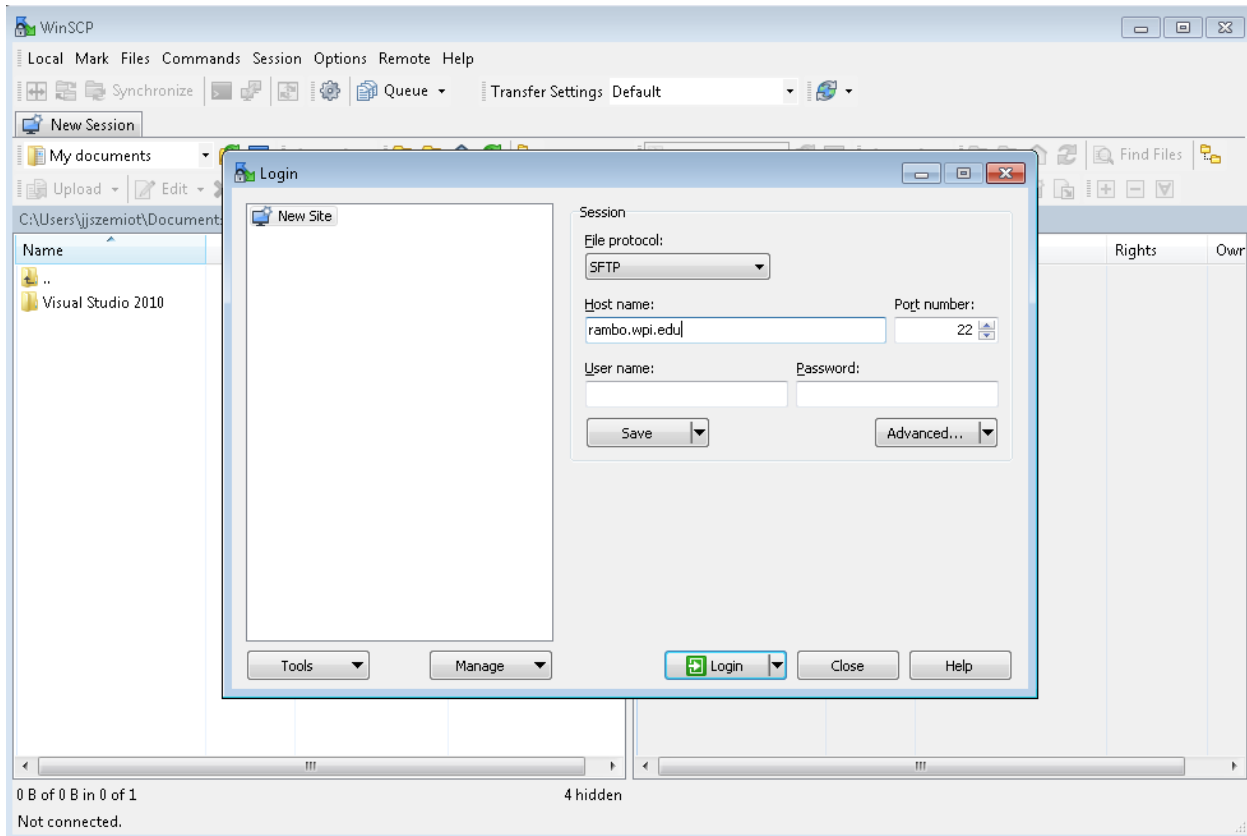


Save the file to your specified location as your preferred type, PNG is the default. Accessing files after saving them is most easily accomplished using WinSCP. More on that in the next section.

Exporting Files From Rambo to Windows

WinSCP is a brilliant solution to moving files between Linux (Rambo) and Windows. There has been a slight change recently, and most of the available computers do not have WinSCP installed anymore, but it is available through remote desktop at windows.wpi.edu. Do not download it online.

Once you open WinSCP, fill in the Host Name with `Rambo.wpi.edu`. Then log on to your WPI user. The file protocol can be either SFTP or SCP.



On the left, you have access to your Windows files. On the right, you have access to your Rambo files. Here you can easily move files from Windows to Linux, or vice versa, and easily see what you are doing. This opens up a world of opportunity.

Closing Cadence

Be sure to close Cadence before closing Rambo, otherwise you will still have it running and it can lock some functionality. In order to close Cadence remotely, log in to Rambo. Type `top` into the command line. You will see a list of open programs in Rambo for all the different users. To filter by user, type `u` then the username you are looking for (probably your own). Note that you can leave it blank to see all users.

VCO-BASED ADC

• • •

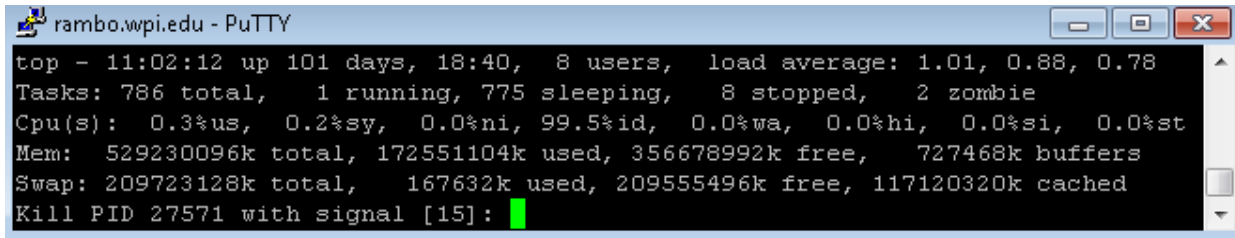
```
rambo.wpi.edu - PuTTY
-bash-4.1$ top
top - 10:56:25 up 101 days, 18:34, 8 users, load average: 0.68, 0.74, 0.74
Tasks: 776 total, 1 running, 765 sleeping, 8 stopped, 2 zombie
Cpu(s): 5.3%us, 2.3%sy, 0.0%ni, 92.4%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 529230096k total, 172419252k used, 356810844k free, 727468k buffers
Swap: 209723128k total, 167632k used, 209555496k free, 117119596k cached
Which user (blank for all):
  PID USER      PR  NI  VIRT  RES  SHR S %CPU  %MEM    TIME+  COMMAND
  1475 rfiida    20   0 10.9g 928m 133m S   5.6   0.2   3582:36 MATLAB
 21694 sli2      20   0 1738m 1.2g  54m S   5.6   0.2  327:19.52 common_shell_ex
 23179 sli2      20   0 1739m 1.2g  54m S   5.6   0.2  582:27.16 common_shell_ex
 27558 jjszemio  20   0 17652 1700  856 R   3.7   0.0   0:00.03 top
 29539 prferrei  20   0 530g 1.2g 180m S   1.9   0.2  680:05.78 MATLAB
 32614 root      20   0 0 0 0 S   1.9   0.0  139:06.73 kondemand/8
    1 root      20   0 21448 1576 1280 S   0.0   0.0   0:11.66 init
    2 root      20   0 0 0 0 S   0.0   0.0   0:00.25 kthreadd
    3 root      RT   0 0 0 0 0 S   0.0   0.0  56:16.28 migration/0
    4 root      20   0 0 0 0 S   0.0   0.0   4:40.44 ksoftirqd/0
    5 root      RT   0 0 0 0 0 S   0.0   0.0   0:00.00 stopper/0
    6 root      RT   0 0 0 0 0 S   0.0   0.0   1:15.34 watchdog/0
    7 root      RT   0 0 0 0 0 S   0.0   0.0  66:16.58 migration/1
    8 root      RT   0 0 0 0 0 S   0.0   0.0   0:00.00 stopper/1
    9 root      20   0 0 0 0 S   0.0   0.0  33:53.89 ksoftirqd/1
   10 root      RT   0 0 0 0 0 S   0.0   0.0   1:37.73 watchdog/1
   11 root      RT   0 0 0 0 0 S   0.0   0.0  36:29.94 migration/2
```

The next step is killing Cadence Virtuoso. Type `k` then the message changes to “PID to kill:”. Note that the PID of Virtuoso in this example is 27571. Type in 27571.

```
rambo.wpi.edu - PuTTY
top - 11:02:12 up 101 days, 18:40, 8 users, load average: 1.01, 0.88, 0.78
Tasks: 786 total, 1 running, 775 sleeping, 8 stopped, 2 zombie
Cpu(s): 0.3%us, 0.2%sy, 0.0%ni, 99.5%id, 0.0%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 529230096k total, 172551104k used, 356678992k free, 727468k buffers
Swap: 209723128k total, 167632k used, 209555496k free, 117120320k cached
PID to kill:
  PID USER      PR  NI  VIRT  RES  SHR S %CPU  %MEM    TIME+  COMMAND
 27571 jjszemio  20   0  863m 235m 134m S   0.3   0.0   0:03.60 virtuoso
 29262 jjszemio  20   0 17668 1840  960 R   0.3   0.0   0:00.07 top
   3292 jjszemio  20   0 52944 4788 3596 S   0.0   0.0   0:00.12 oafsLockD
 21185 jjszemio  20   0 12520  704  604 S   0.0   0.0   7:54.78 clsbd
 27290 jjszemio  20   0 110m 4328 1476 S   0.0   0.0   0:00.12 bash
 27673 jjszemio  20   0  2420  908  712 S   0.0   0.0   0:00.00 cdsMsgServer
 27763 jjszemio  20   0  2248  560  464 S   0.0   0.0   0:00.04 cdsServIpc
 27780 jjszemio  20   0  184m  22m  18m S   0.0   0.0   0:00.17 ipvs
 28553 jjszemio  20   0  2248  532  448 S   0.0   0.0   0:00.04 cdsServIpc
 28570 jjszemio  30  10 28832 3020 1964 S   0.0   0.0   0:00.07 tcsh
 29082 jjszemio  20   0  2248  532  448 S   0.0   0.0   0:00.04 cdsServIpc
 29087 jjszemio  20   0  2248  528  444 S   0.0   0.0   0:00.03 cdsServIpc
 29133 jjszemio  20   0  215m  31m  23m S   0.0   0.0   0:00.37 libManager
 29134 jjszemio  20   0  215m  30m  23m S   0.0   0.0   0:00.35 libSelect
```

• • •

After you define the PID number, you are given the option to define the strength of the signal. The default value is 15. This is usually adequate, but in case it is not strong enough, use a smaller value. As counterintuitive as it may be, less is more powerful.

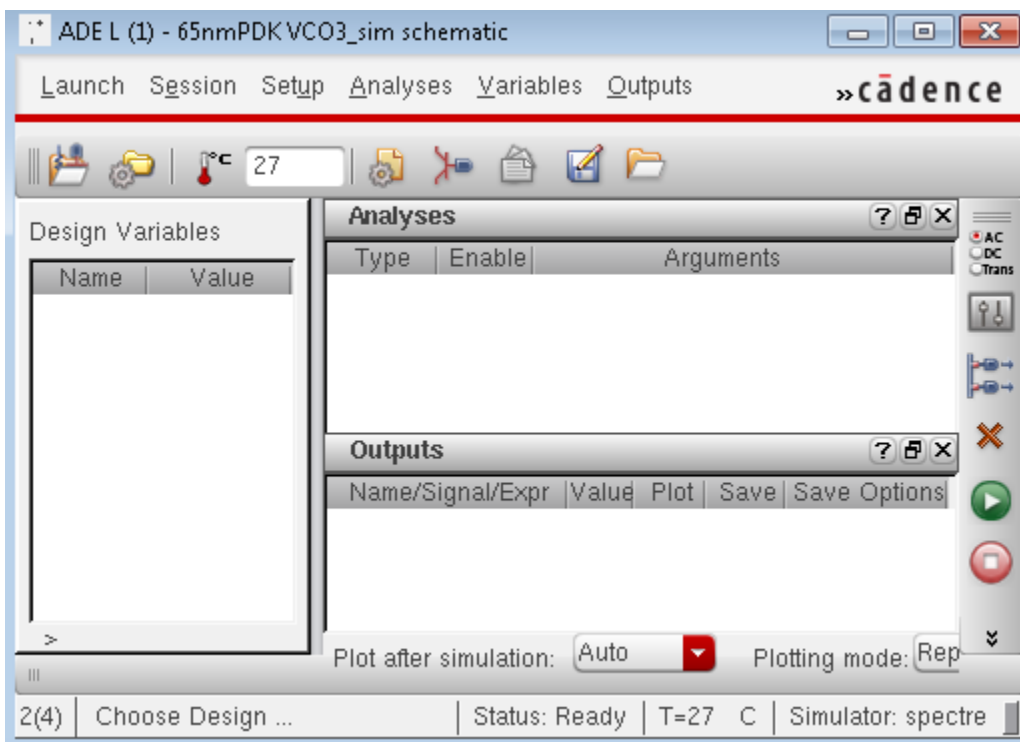


```
rambo.wpi.edu - PuTTY
top - 11:02:12 up 101 days, 18:40,  8 users,  load average: 1.01, 0.88, 0.78
Tasks: 786 total,   1 running, 775 sleeping,   8 stopped,   2 zombie
Cpu(s):  0.3%us,   0.2%sy,   0.0%ni, 99.5%id,   0.0%wa,   0.0%hi,   0.0%si,   0.0%st
Mem:   529230096k total, 172551104k used, 356678992k free,  727468k buffers
Swap: 209723128k total,  167632k used, 209555496k free, 117120320k cached
Kill PID 27571 with signal [15]:
```

Virtuoso will disappear from the list of open programs. Once you are finished, type `q` to quit. Now before you begin feeling all powerful, and that you can remotely close other people's programs to mess with them: no, you don't have the permissions. But if it makes you feel better, try.

Analog Simulations

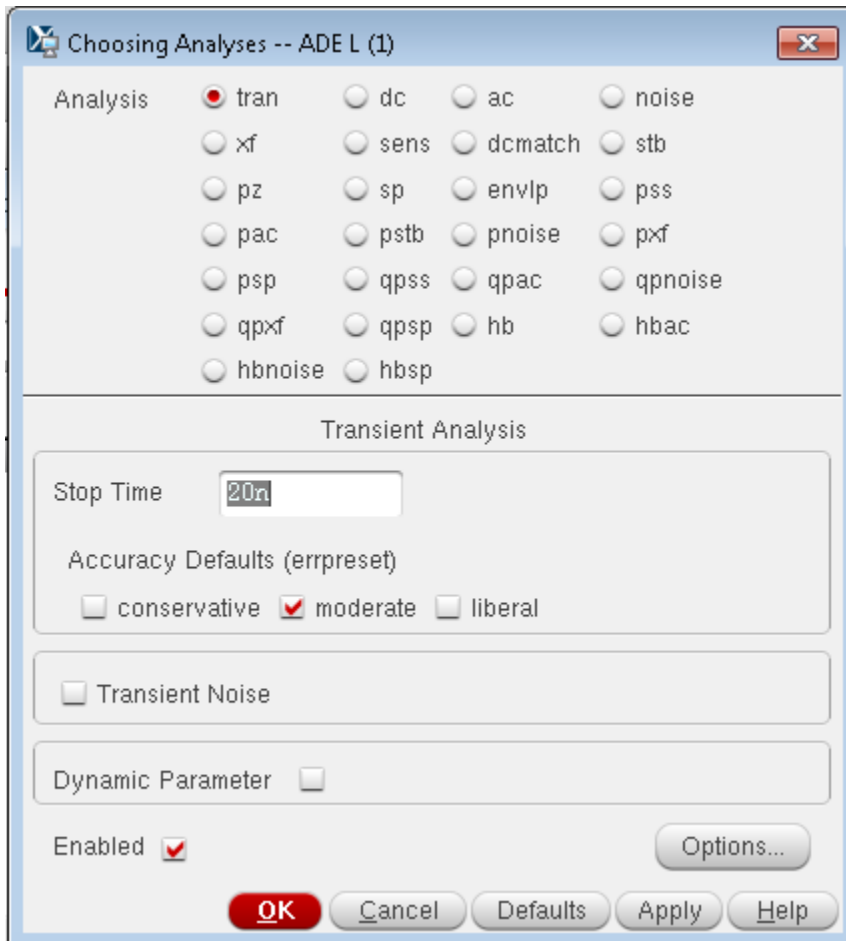
In order to simulate a circuit, you need to create a schematic, then save and check for errors. Once you have no warnings, press *Launch* -> *ADE L* in the top left toolbar of your schematic. ADE stands for Analog Design Environment.



Once you open ADE L, there are a variety of options to choose from. Note that you can change the temperature (Celsius) of the simulation. First, set up the kind of simulation using Analyses. The most

• • •

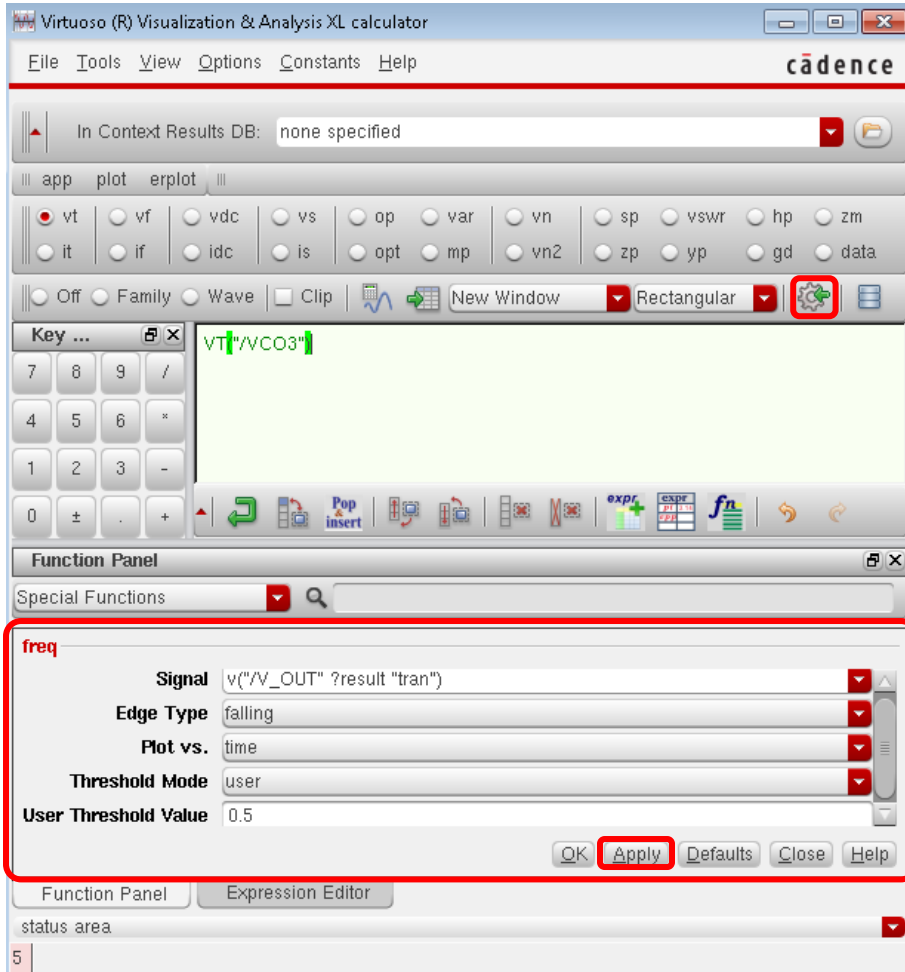
common simulation we were doing was trans(ient), which looks at voltage over time. There are plenty of other options, including AC and DC. Make sure to indicate the stop time, and there are error presets to consider. Setting accuracy defaults to moderate works fine. When defining parameters such as time, do not include units. Only include numerical prefixes such as n (nano), p (pico), u (micro), and so on.



Should you use variables in your schematic, you should select *Variables -> Copy From Cellview*, then define them in your ADE L state. Then define your outputs. It is easiest to select *Outputs -> Setup* then select the desired nodes on the design (or schematic). You can also simulate things, such as frequency, by using the calculator function. To calculate frequency of an oscillating output, *open* the Calculator. Select vt, and identify the location you wish to measure the frequency at. Note that in the example, the selected wire was labeled as VCO3. From the list of functions, select freq.

VCO-BASED ADC

• • •



In the above example, the edge type (rising or falling) does not matter. Frequency is plotted as a function of time given a ramp input using *vpwl*. It is advisable to define the threshold value yourself; in our case, the supply was 1V so the threshold was at the midway point. Generate the expression by pressing *Apply*. To return the calculator expression to the ADE L outputs, press the gear. Close the calculator and return to the outputs. It is convenient to change the name of the frequency expression.

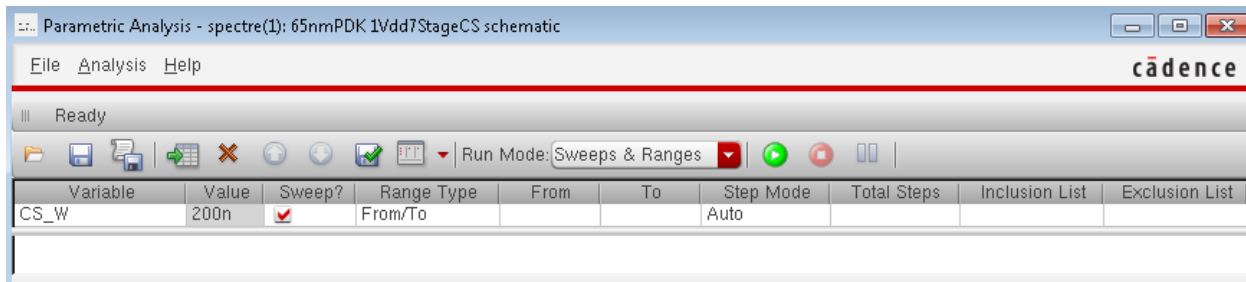
The next step is setting initial conditions to get the circuit oscillating. Select *Simulation -> Convergence Aids -> Node Set...* and identify the same output node as you are taking the frequency from. Set it to 0 voltage. Repeat for the initial condition (*Simulation -> Convergence Aids -> Initial Condition...* and set voltage to 0)

Now press the green button to simulate. The plots will open once the simulation is done if there are no errors. If you want to look at one circuit by sweeping the temperature, or sweeping the width of all of the current starving MOSFETs (you will need to define the width of each as a variable), use parametric analysis. To access it, select *Tools -> Parametric Analysis* from the ADE L window. Begin by defining the variable you wish to sweep, for instance CS_W (width of current-starving MOSFETs), and selecting the

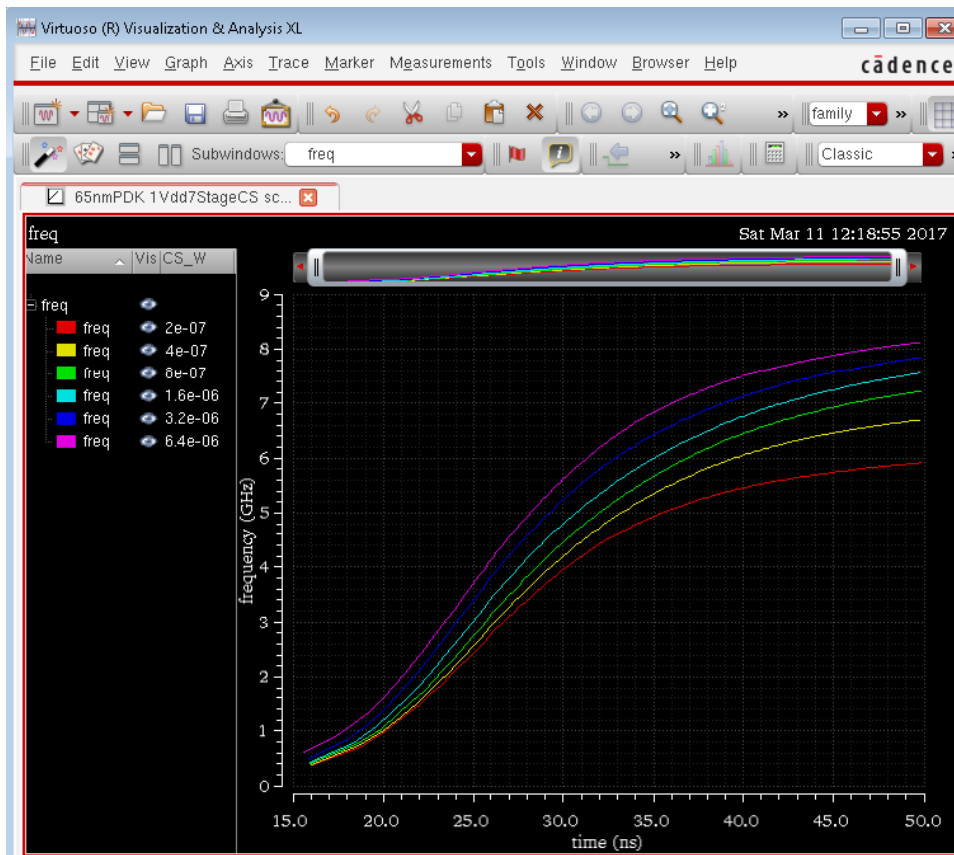
VCO-BASED ADC

• • •

Range Type. When you know where you want to start and approximately how large you want to go, From/To works well. Define your starting point. Note that you cannot size the dimensions of the device to be smaller than minimum, which is typically the default. You have control over the step mode and the total steps. When scaling by a factor, the times step mode is appropriate. Linear steps are also useful for a more linear range. The other options typically give you decimal values, and so we recommend against them.



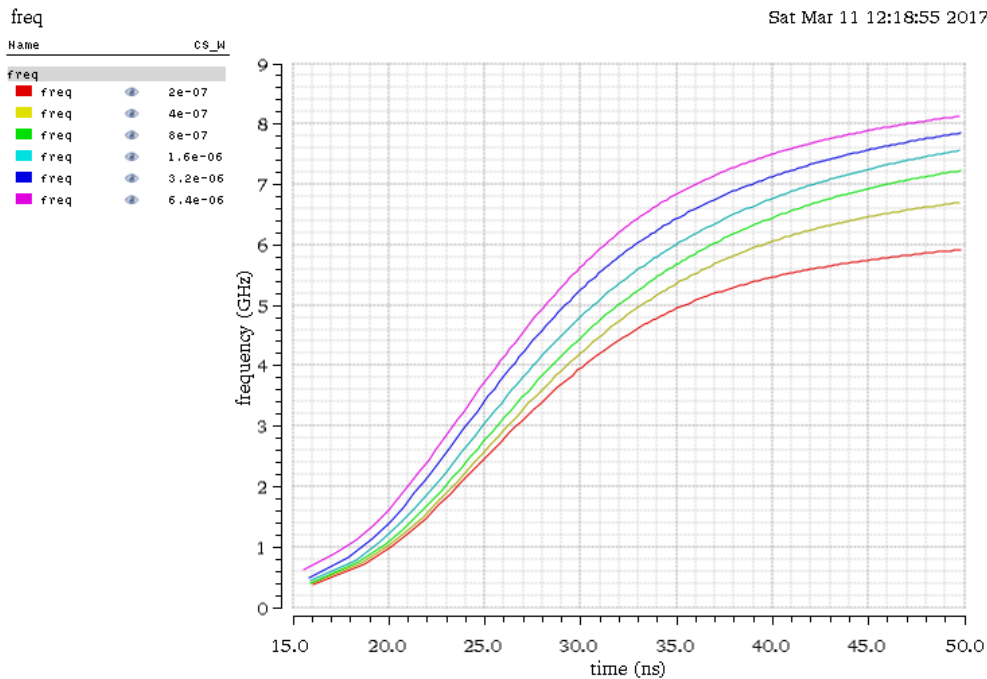
Press the green play button to simulate again. This gives you a family of curves.



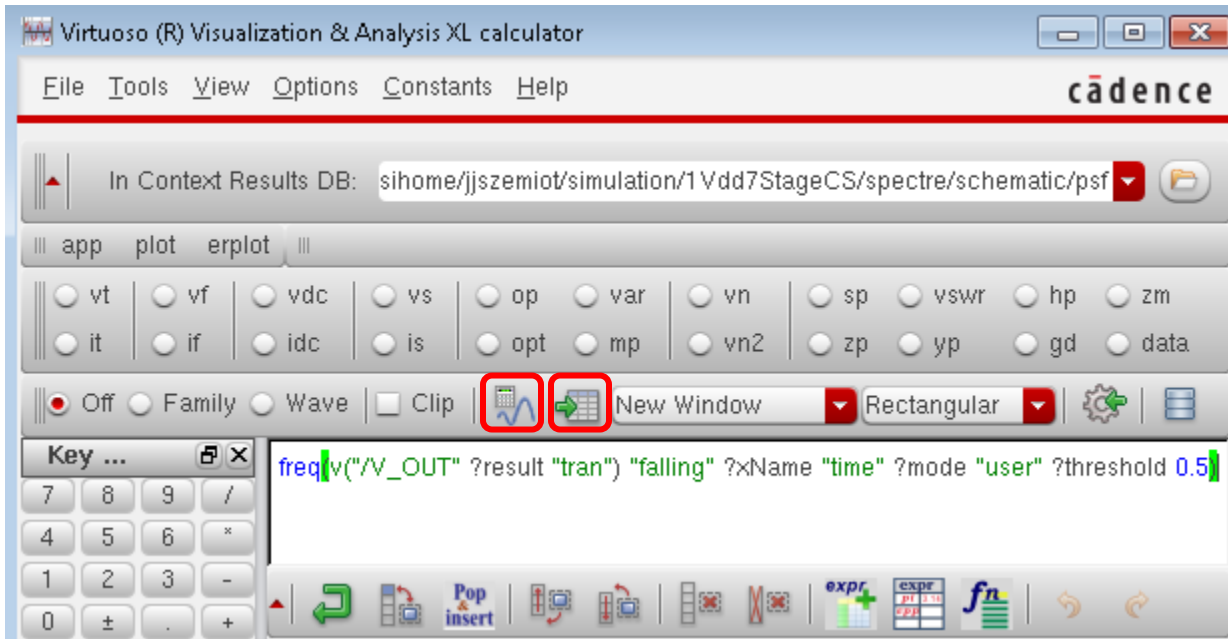
To change the background to white, or any other color really, select File -> Save Image... then give it a name and identify a location. Use WinSCP to open it. The resulting image is shown below.

VCO-BASED ADC

• • •



The other option is to export data instead of saving an image. This is useful for data analysis, and allows you to change the x-axis to voltage instead of time. From the simulation plot press *Tools* -> *Calculator* and evaluate the buffer as a waveform to make sure you are looking at the right function. The icon includes a calculator and an oscillating waveform. When it matches your previous waveforms, press the green arrow over a grid icon. This will give you the values that you can import to Excel.



• • •

Once you have the list of values, save it as CSV. Then, use WinSCP to place it in Windows. Open it with Excel, and all of the values should be labeled and in their own columns. Now you can edit the data for better presentation as you wish.

Once you are satisfied with your simulation, be sure to save the state. You can save state as a directory or as a cellview. Both have their merits and failings. The cellview will show up under your schematic, and when sharing your schematic with a group member, he will have access to it as well. Should you save it as a directory, it is easier to load and doesn't clutter your cellview.

Chapter 2: Ask for help early, and often

This should be how you approach this MQP. No matter what kind of exposure you had to layout or design during an internship, it is nothing like at WPI. Reason being, Cadence is a program that a variety of IC manufacturing companies use, but each PDK is assembled by the manufacturing company, and each location has their own shortcuts to access them, and thus odds are your experience is nothing like at WPI. For this reason, talk to Prof. McNeill and to the graduate students in the NECAMSID lab. They use this version of software and most probably the same PDK on a daily basis, so if anyone knows anything, they would.

Unfortunately, they all have their own things to do, so sometimes they might seem less than willing to help. When this happens, they usually tell you to watch tutorials on youtube or use the Cadence tutorials directly. We did that. Unfortunately Cadence has a lot of different versions, so it's tough to find something applicable that mirrors the kinds of things you are doing.

Don't expect the information to come freely, you have to really dig. We realized as a team that we would receive guidance, advance in that direction, neglect to do something the people we were asking considered obvious, and hit a concrete wall. For example, when we finally had access to the analog library, none of our teammates remembered to attach our library to the new library, so we couldn't do any design rule checks in layout. Additionally, our ADEL simulations presented errors when two kinds of libraries were defined. **When this happens, open ADE L, then Setup -> Model Libraries, and deselect the library that you are not using (ie. gpdk045).**

Another such instance occurred while making layouts of ring oscillators, when I noticed there were no substrates in either the PMOS or NMOS layouts, but they were necessary in the schematic. By the time I reached DRC, I had completely forgotten about this, but I got a lot of errors regarding guard rings even though we were operating at 1 V. Guard rings are only necessary at high voltages, which we didn't have, so I brought the problem to a graduate student. After a longer while, he noticed I was missing substrates, which could be integrated or attached. Note that that the side of attachment matters, and only integrate the substrate (short with drain or source as appropriate) if the drain or source is directly connected to ground or supply, respectively. More layout mishaps will be discussed in later sections.



Chapter 3: Everything is documented somewhere

This applies to just about everything, but it is particularly necessary for layout design. For that reason, this section is dedicated to anecdotes, and for more particulars on layout, look to the next two chapters. The main idea is, if you think you're missing some necessary information about the process, you probably are. It should all be in the appropriate folder containing the PDK, but as you will quickly find, there are layers and layers of directories containing files. Most of them are incomprehensible. The ones you need contain details in particular about design rules, in our case it was titled the Design Manual. Had the grad students not emailed this manual to us, we would have never found it.

Similarly, there is a file regarding the order of verifying layout. In most tutorials we found, the proper order is Design Rule Check (DRC), Parasitic Extraction (PEX), then finally Layout-Versus-Schematic (LVS). In the technology we were using, it was DRC, LVS, and PEX last. Bottom line: ask or look. It is documented somewhere. It's in your best interest to find out where, and soon. For that, it is probably easiest to ask.

The next two chapters most thoroughly lead you through the layout process.

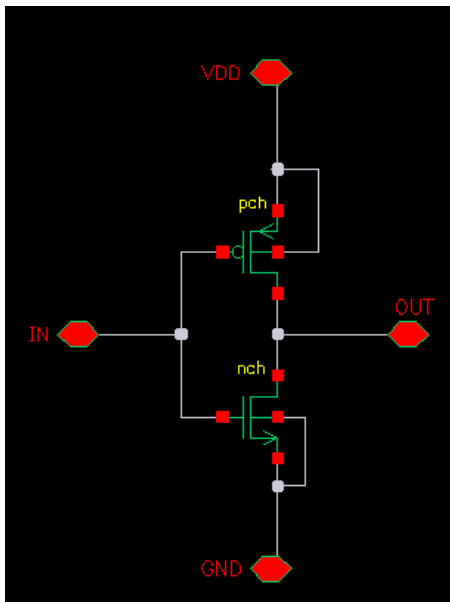
Chapter 4: Analog Layout Fundamentals

Properly doing layout might just be the hardest part. It is very artsy and fun, but as you lose yourself in making connections, do not forget that there are design rules which specify the dimensions and minimum spacings between layers. One of the hardest parts is that there is no real standard to naming layers. That's where the **design manual** comes into play. It is personalized with the technology you are using, so use it.

The other thing is checking your design while you are making it to address errors right away, rather than finishing a design to find hundreds of errors, most of them repeats. One of the most annoying ones is when you placed your components too close to each other. Another really tedious error, hitting me with more than 800 errors, was using a snap spacing that had greater precision than the technology. I had to start all over. So run DRC (Design Rule Check) often.

Setting Up

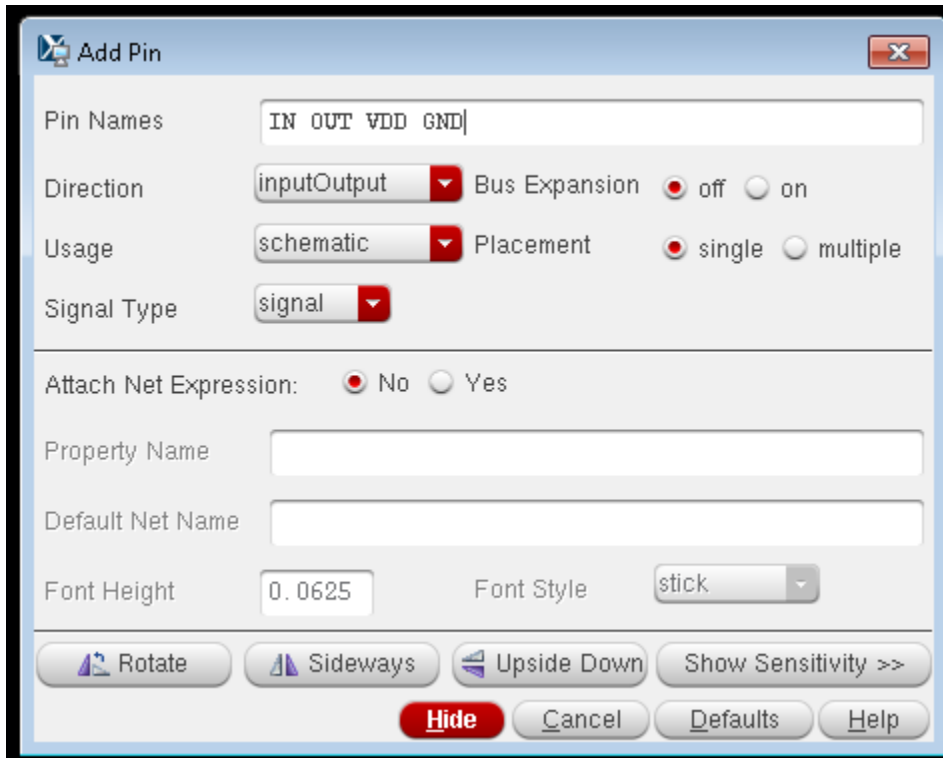
First, you need to have a complete schematic with all of its inputs and outputs marked with pins. Duplicate the schematic you were simulating before, then change the pins. A simple example would be the inverter.



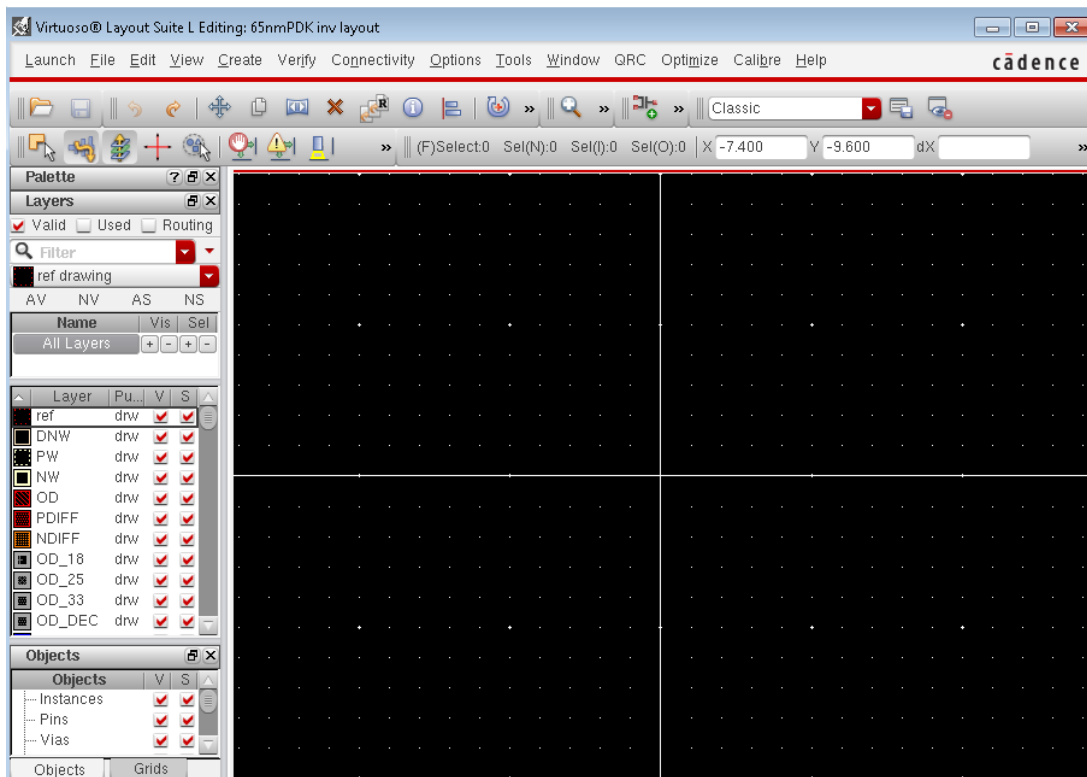
The shortcut for placing pins is `p`. List the pin name(s). You can do all of them at once, but be sure to remember the order you named them in. Otherwise you aren't saving any time. For ease with LVS (Layout-Versus-Schematic) later, mark them all as inputOutput direction.

VCO-BASED ADC

...

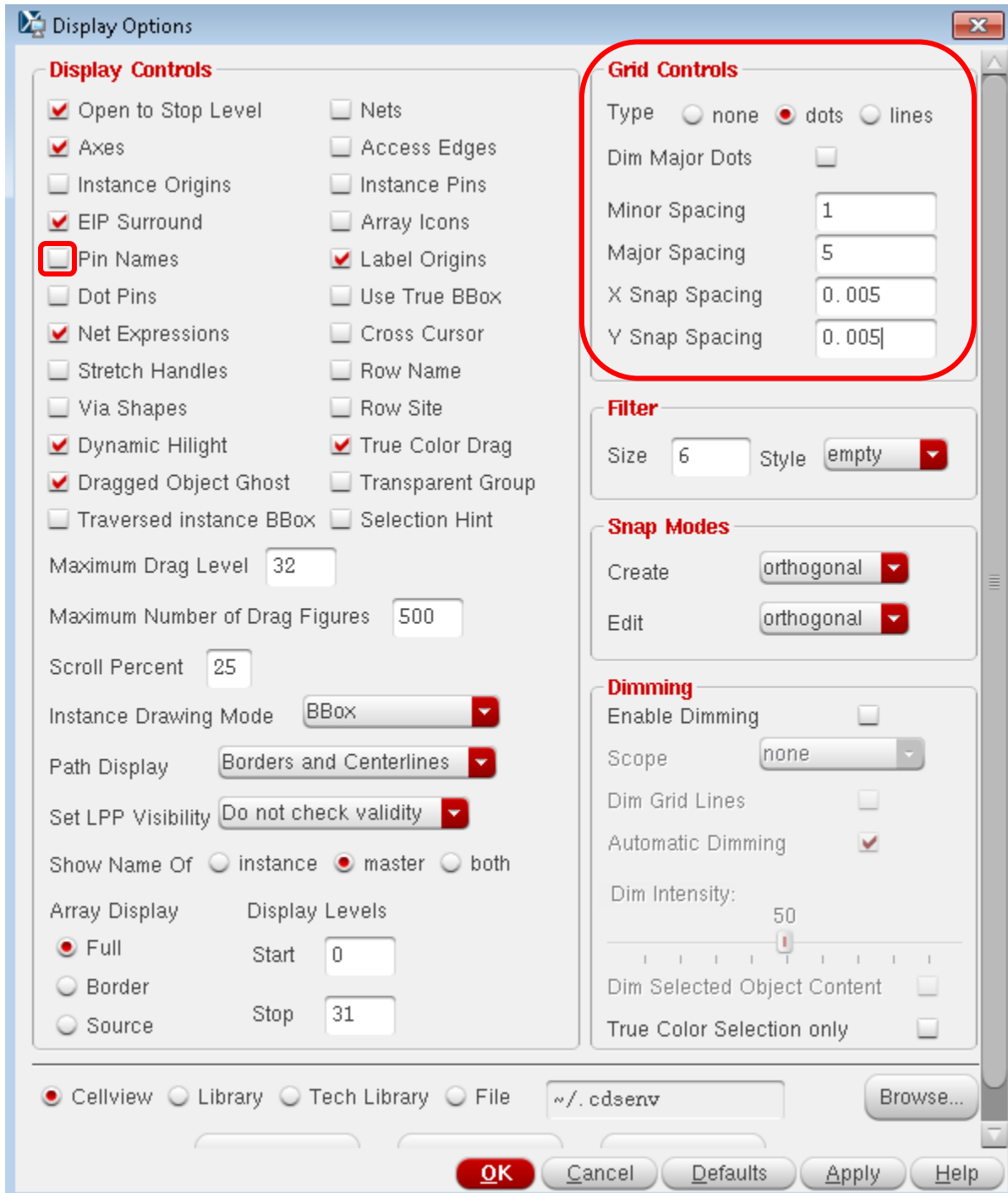


Check and save your schematic, then from the Library Manager select *File* -> *New* -> *Cell View*. Select the view to be layout, and the application to be Layout L. Wait for it to open.



...

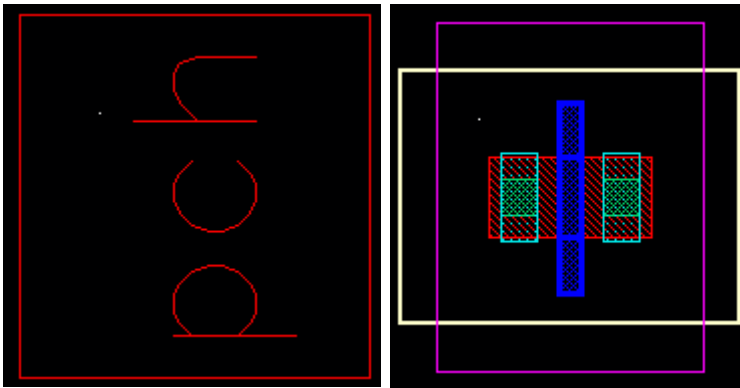
A good place to start would be Display Options, to make sure you don't get nearly a thousand errors. Press **e** to open them. In the technology we were using, the appropriate spacing was 0.005 for X and Y Snap Spacing. This spacing should be small enough so your cursor can rest where all of the layers of a component end, but this may take some trial and error. It might be worthwhile to ask someone who knows. You will need to change these preferences every time you open a layout. Gotta love Cadence.



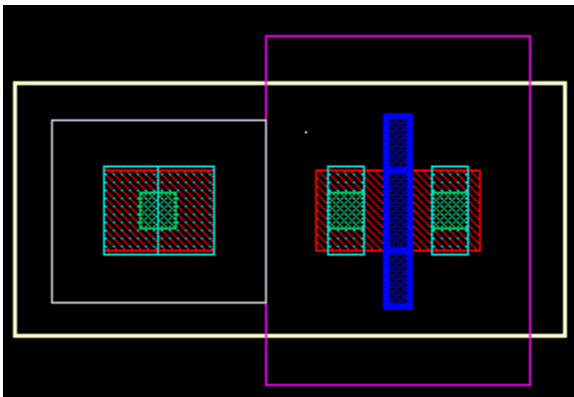
• • •

While you're at it, select pin names to be visible. It will be useful soon.

The next step is placing all of the components, or a portion depending on complexity. The way to do this is to press **i** and select the desired components matching those in the schematic. If you get red outlines and the instance label, press **shift+f** to reveal the hidden layers. You will probably need to do this every time you open the layout.



Some devices are symmetric, but others are not. The text indicates the source on the left, the drain on the right, and the gate in between. Note that the substrate is missing. That doesn't match the schematic, so we need to add it. Select the device and press **q**. In the parameter tab, **make sure to indicate the body** and you are responsible for placing it on the appropriate side. The body can be either attached or detached. For a PMOS, attach the body to the supply, and for an NMOS, attach the body to ground.



There are a few major considerations when making layouts:

1. Do not use polysilicon (poly or PO) unless you need to for passing DRC. Otherwise, when making connections, use metal (M1, M2,...). It has lower resistance and works better for this purpose.
2. Use as few layers as you can. The number of layers impacts manufacturing time heavily, and more layers add much more time.
3. Make your connections as short as possible.
4. Minimize die area, so keep components and wires as close to each other as DRC allows.

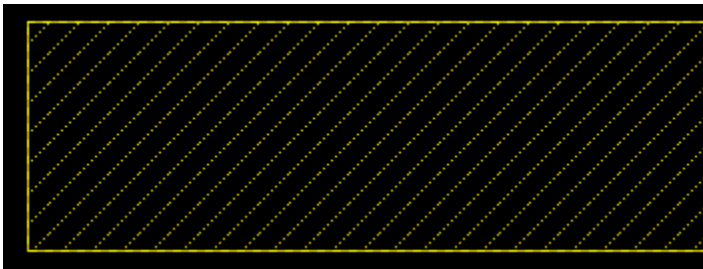
VCO-BASED ADC

• • •

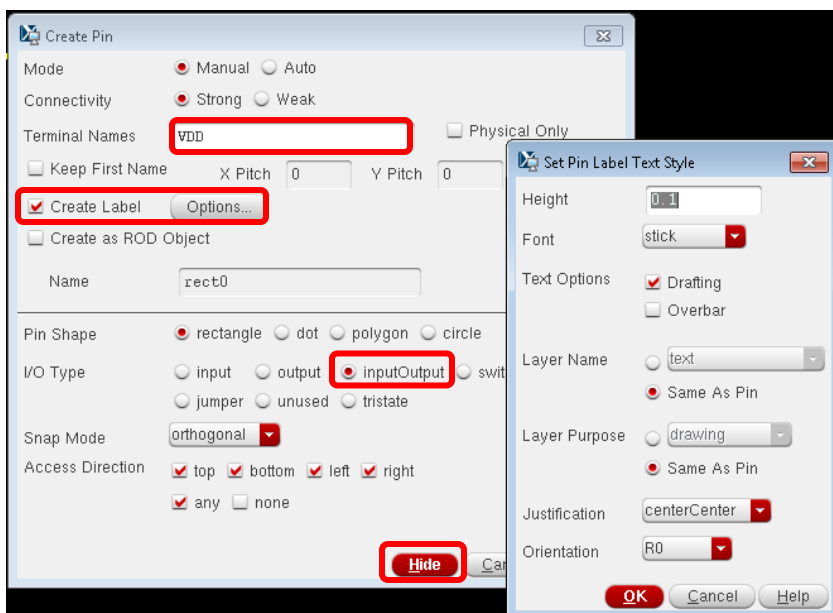
5. Make wires have close to minimum width.
6. Use drw for your standard layers.
7. Polysilicon must be on P+ Source/Drain Ion Implantation (PP) or N+ Source/Drain Ion Implantation (NP). The distance the PP or NP exceeds polysilicon must be at least a specified value.
8. Each layer must be at least a minimum area. If it is not so in a component you are using, you must add more of that layer without changing the dimensions of the device.

The next part involves moving between layers. For this purpose, you need vias or contacts. From polysilicon to metal 1, you need a contact (cont or CO). Between metal 1 and metal 2, you need via 1 (VIA1). From metal 2 to metal 3, via 2 (VIA2). And so on. Note that the dimensions of contacts and vias are set, and they are square. The layers between them must exceed the via or contact by a set amount.

One last step before delving into Calibre, another tool outside of Cadence you need installed to verify your layouts, are those pesky pins of inputs, outputs, and rails. These errors actually come up for LVS, but it is much better to eliminate errors as soon as you can. Here you are looking at the end of the metal 2 layer, which will be connected to VDD. It doesn't matter which metal layer the pins are, but the name must match the schematic.

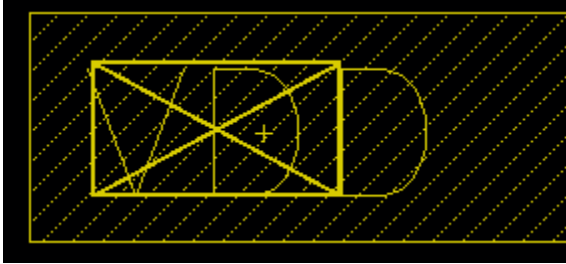


Select *Create -> Pin* and the following menu appears. Fill it out the same way.



• • •

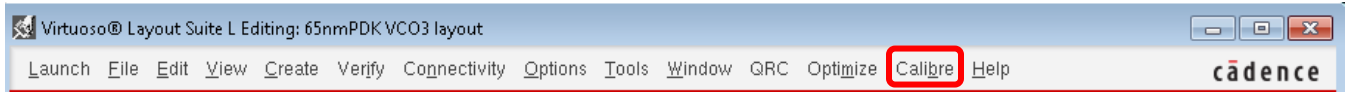
Hide the menu. Notice the different cursor. Select the metal 2 pin layer and draw a smaller rectangle within the metal 2 drawing (drw) layer already present. Place the text label within the metal 2 drw layer as well. Make sure the label layer has purpose pin. This will make the software identify the pin.



Repeat this process for the remaining pins. Save your layout. Now you are ready for the full-fledged verification. Rather, I think those are all of the highlights of the layout adventure prior to Calibre. I'm sure you will stumble upon, or maybe it would be more accurate to say crash into, many more.

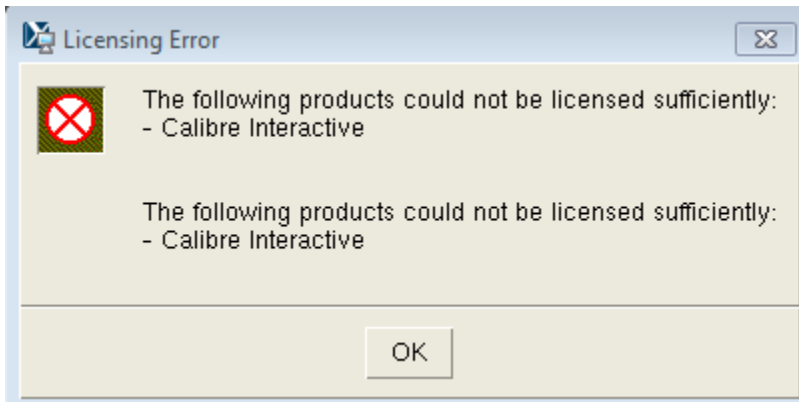
Calibre

Make sure that the *Calibre* toolbar is available. Note that it should appear right next to *Help* when you open a layout. We are using the Calibre GUI RCX flow rather than command line.



As a note, the errors that Calibre spits at you once you run one of the verifications are not always the reason something is not working, just an indication that not all is well. One instance was when I neglected to include the substrate in the MOSFETs. The error I was getting was related to guard rings, which are only needed for high voltages. We were dealing with 1 V supply, just about as low as the technology allowed. Of course it was not the problem. These errors disappeared once I added all of the body terminals. Lots of other errors popped up with that change, of course, but the guard ring error was gone.

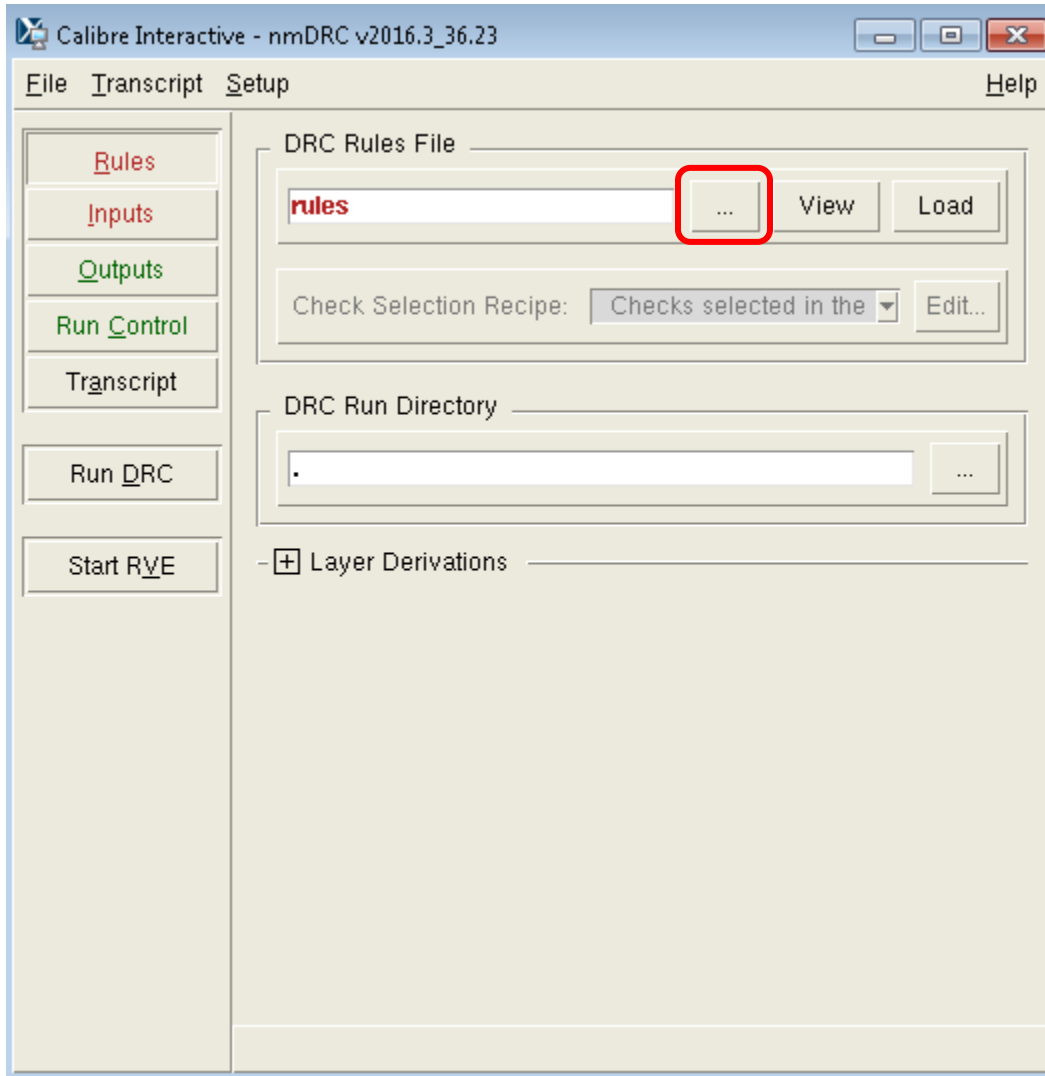
Calibre is licensed through a license server: *cds.ece.wpi.edu*. Should the server be down, expect to see the following message when you try to run any of the verification programs. Our server outage was fixed within a week, but we only found out what happened once it was fixed.



• • •

Design Rule Check

The first step is DRC, making sure the layout passes all of the demands of the design manual. In the technology we were using, it was called nmDRC... and the toolbar looks something like this:

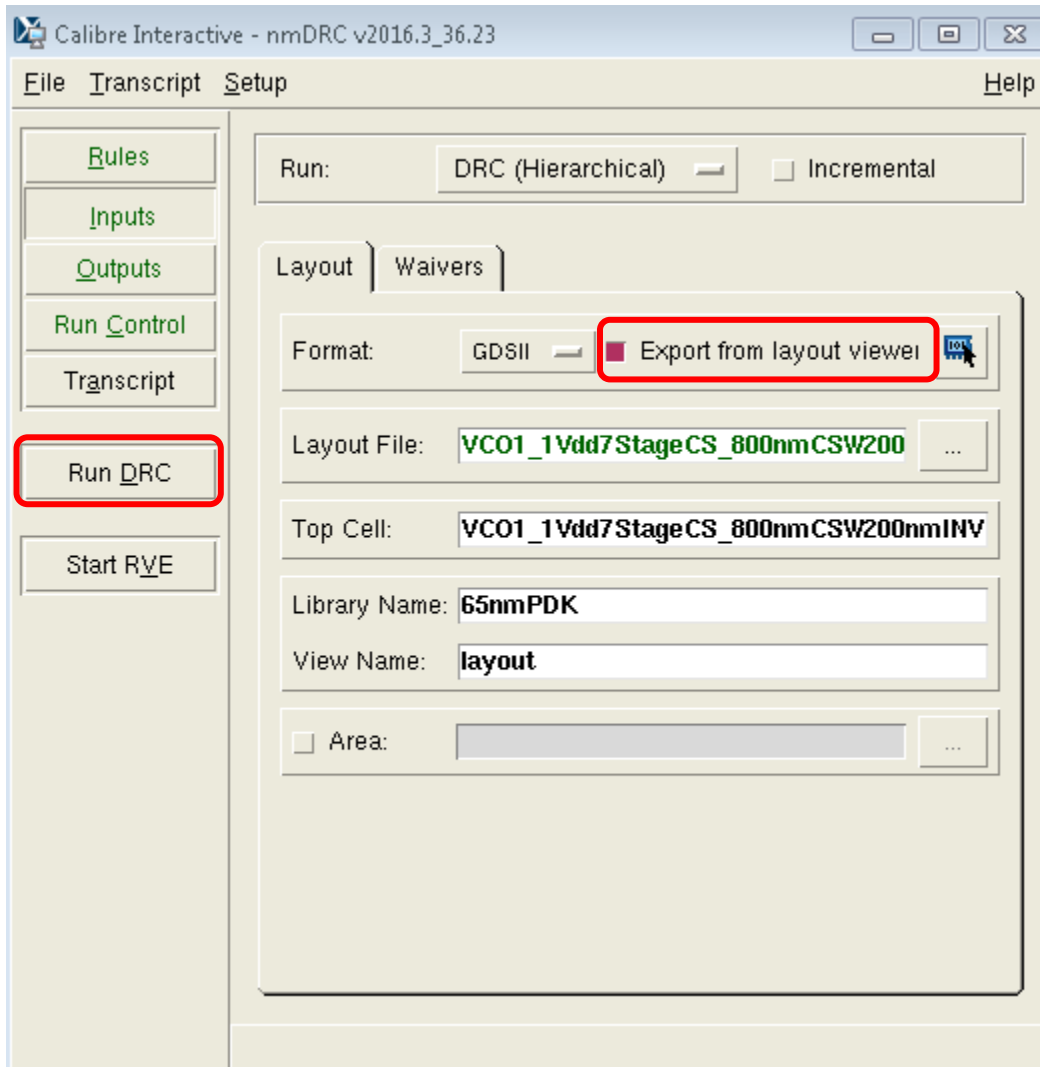


Using the ... button, identify the rules library. Typically it should be found somewhere in the Cadence directory and named something like *calibre.drc*, but it definitely depends on who is installing it and such. In our case, the files were in */share/apps/cadence/TSMC65gp/Calibre* but without permissions, you won't see this location. It is useful to create another directory such as *CalibreRun* for the DRC Run Directory where all of the reports will go.

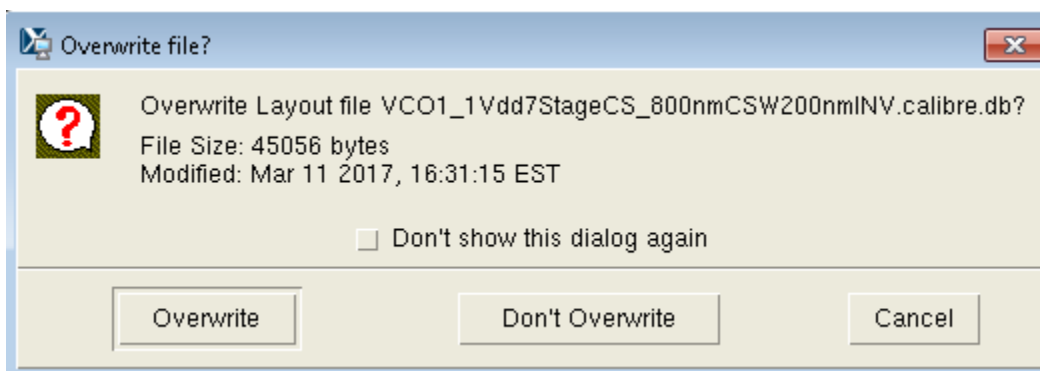
Make sure the inputs are valid, and match the name of the layout and library you are using (mine was 65nmPDK). Be sure to mark the box next to "Export from layout viewer". Then press *Run DRC*.

VCO-BASED ADC

...



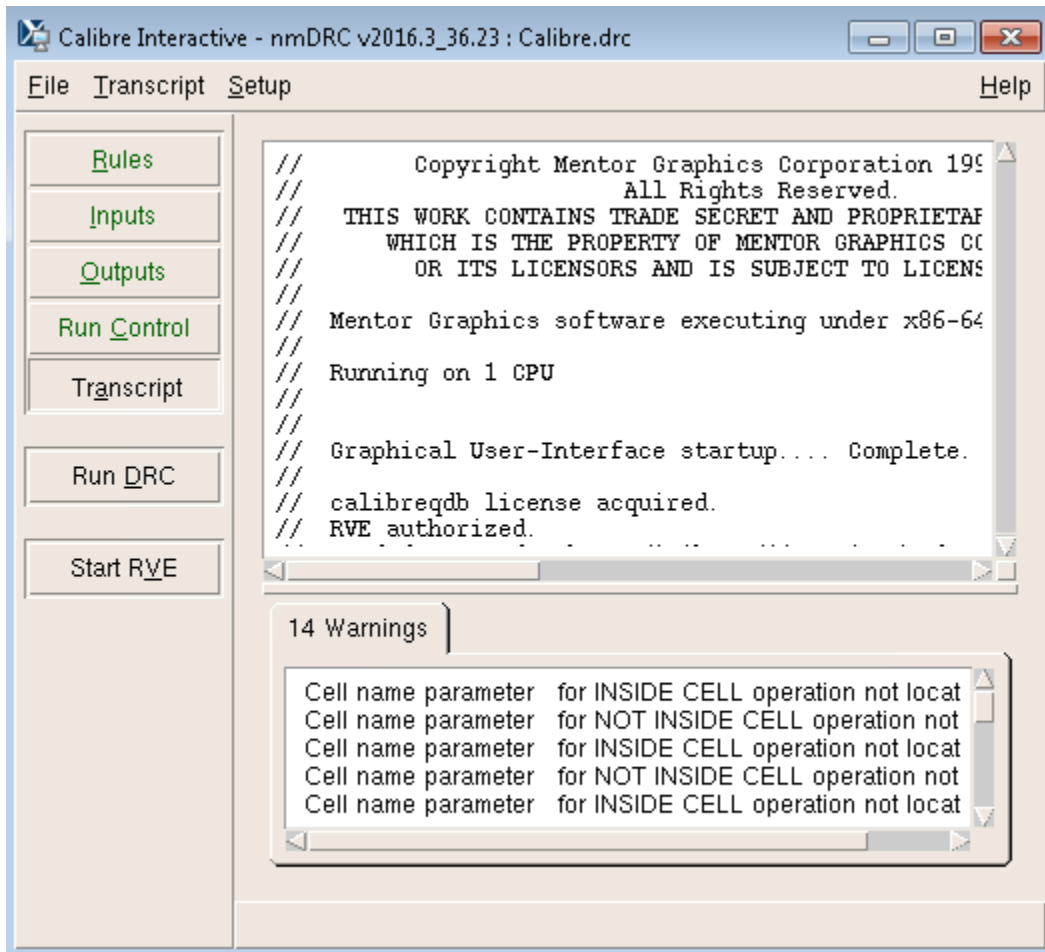
Once you press it, wait. It's pretty slow. If you ran it before, you will get this message. Overwriting is fine.



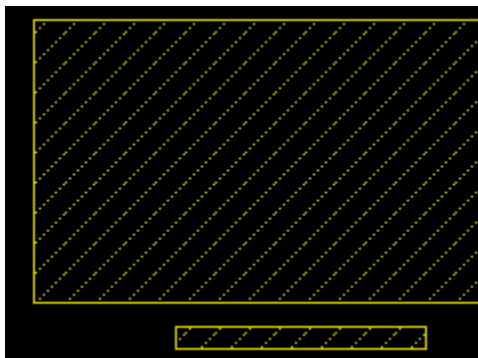
VCO-BASED ADC

• • •

Three windows open: Calibre Interactive, DRC Summary Report, and Calibre - RVE. The warnings in Calibre Interactive can be ignored.

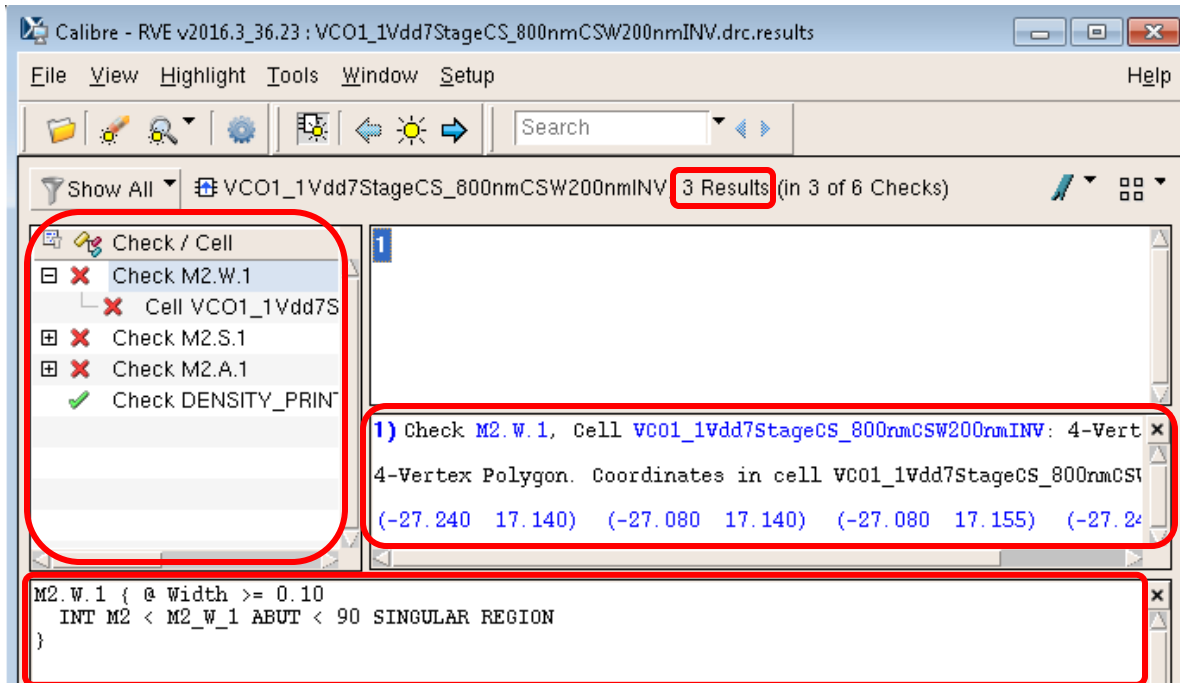


The most important window is Calibre – RVE. **Make sure to address and eliminate all of the errors.** The best way to accomplish this is to fix a few things at a time, then run DRC again to make sure you are fixing the errors. For instance, I added a sliver of M2 next to AVDD, which is both too close and too narrow.

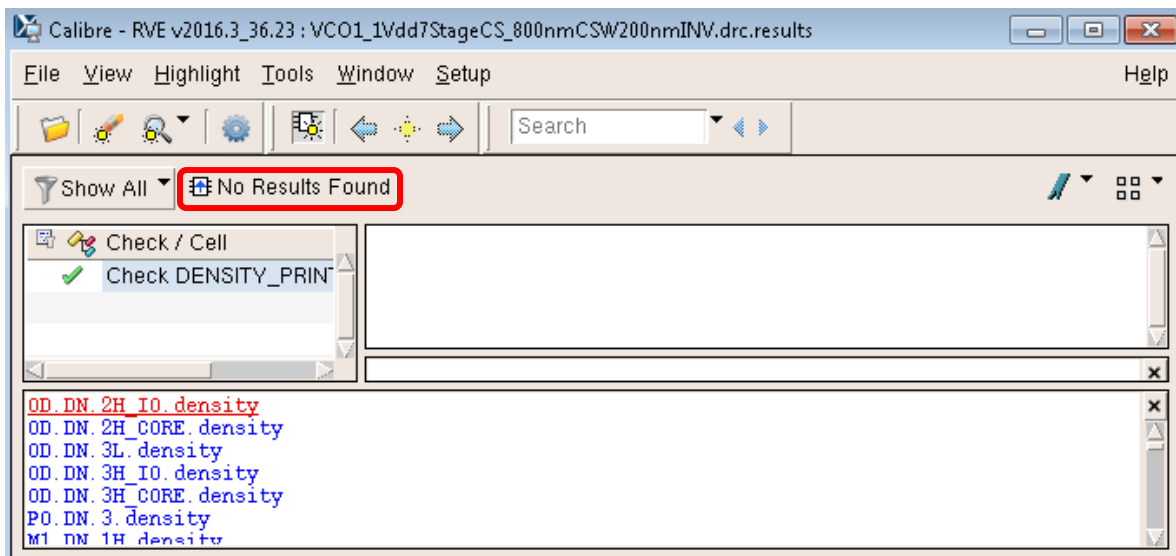


...

Note that this one addition causes three errors. The top left panel indicates the error names as they are referenced in the Design Manual, and that is the resource to use if the error does not have enough description here. The top left panel indicates the number of instances where a particular check failed, and right below that after selecting the instance it provides the coordinates which you can pan or zoom to. I recommend zooming in closely so you can see the details then panning to the point, rather than using the cursor and finding the X-Y coordinates. The bottom panel indicates the error, for instance the layer width having to be at least 0.10 μ m (implied when no unit specified).



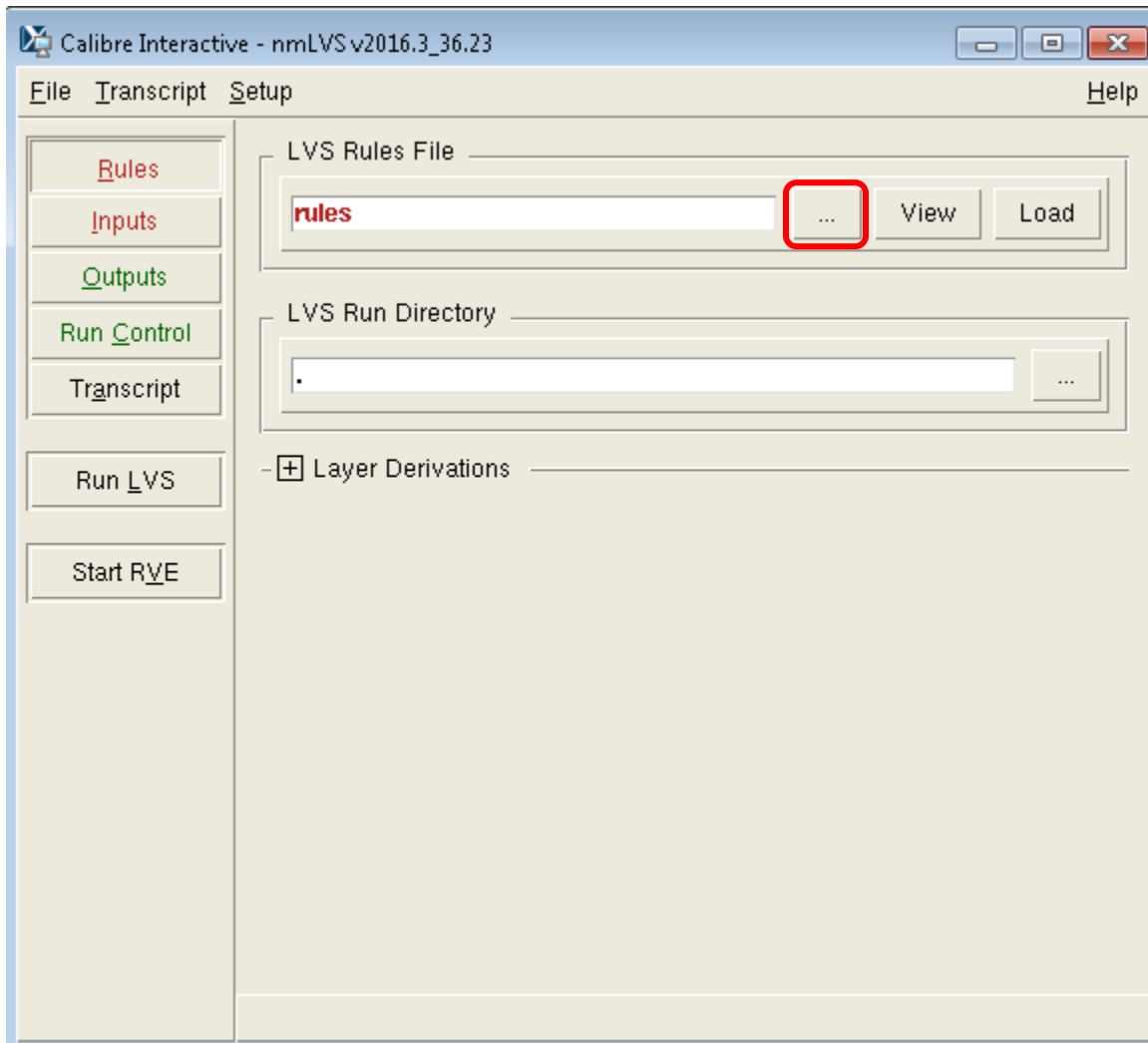
Once all the errors are fixed, the following Calibre-RVE window appears.



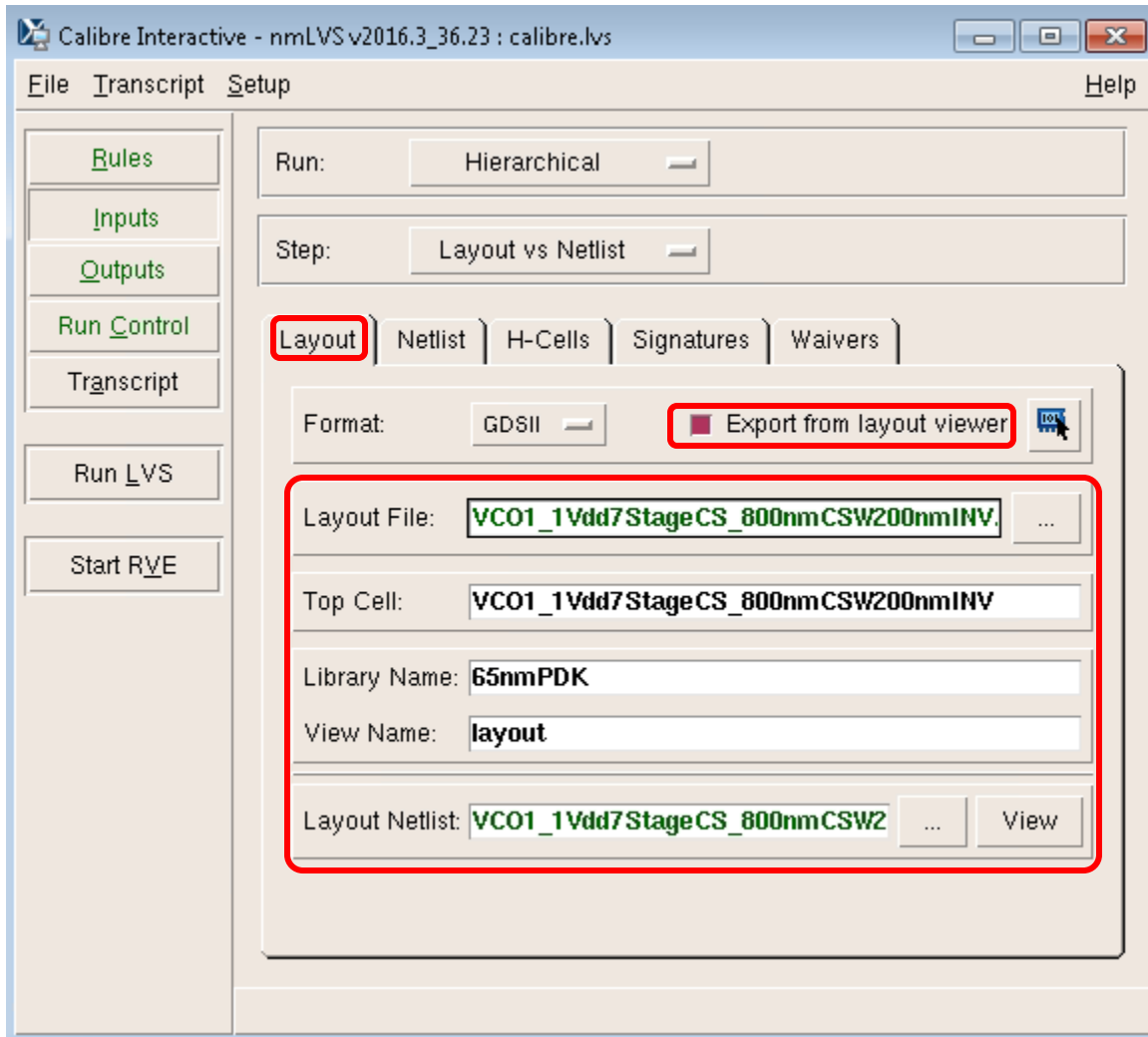
• • •

Layout-Versus-Schematic

Time to move on to Layout-Versus-Schematic (LVS). In the technology we were using, LVS was accessed through *Calibre* -> *Run nmLVS*. The Calibre Interactive – nmLVS window opens. Once again you need to identify the rules file associated with LVS, and specify the inputs. The rules file was in */share/apps/cadence/TSMC65gp/Calibre* once again, this time in the LVS folder. It should be called something similar to *calibre.lvs*.

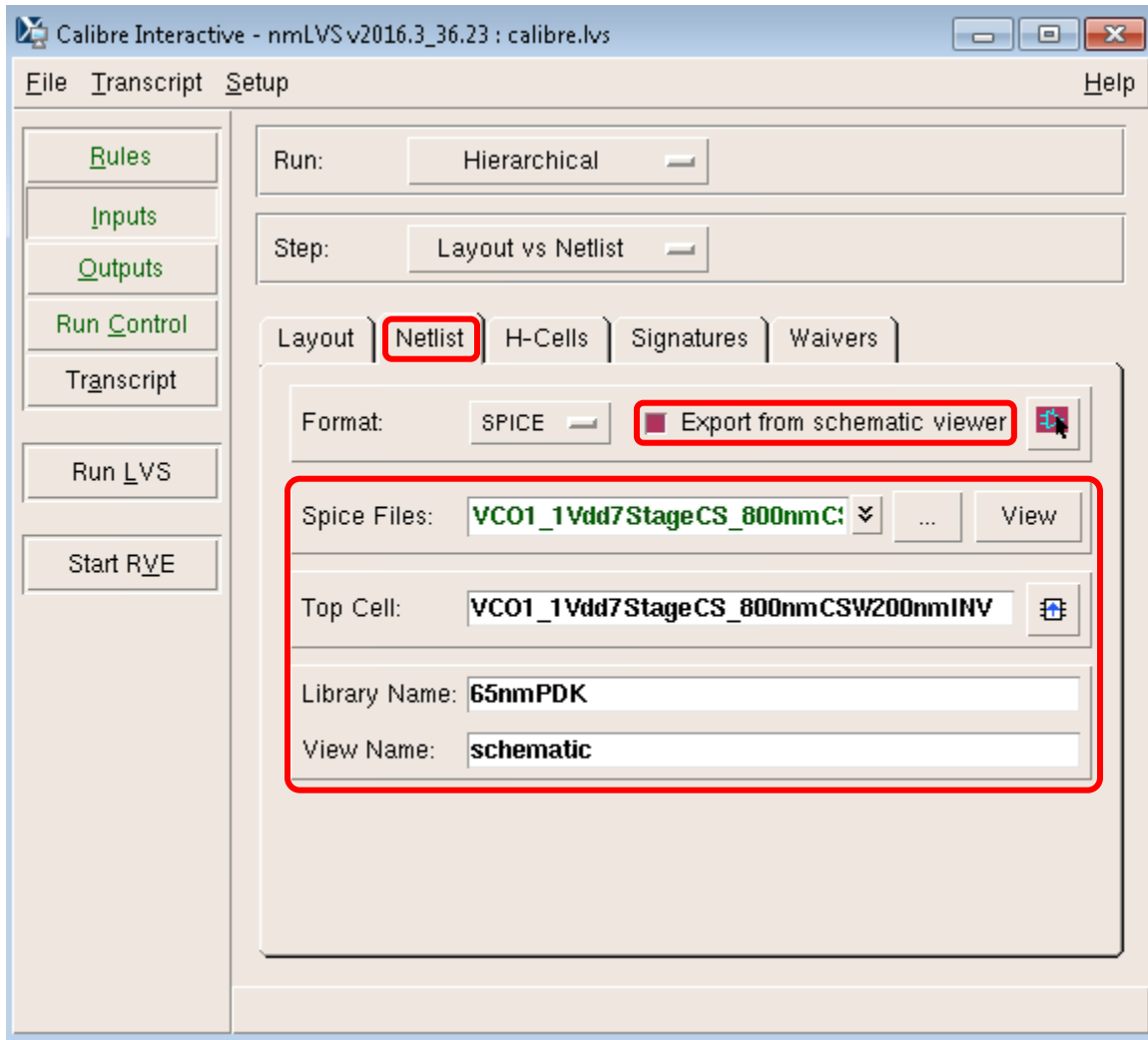


The LVS inputs are a little more demanding. You need to specify both the layout and the schematic for comparison, and make sure they are the appropriate ones. For the Layout tab, the library and view name should be correct by default, but check. Make sure to select “Export from layout viewer” as indicated.



Then select the Netlist tab. Indicate “Export from schematic viewer”. Once again, the library and view name should be correct by default.

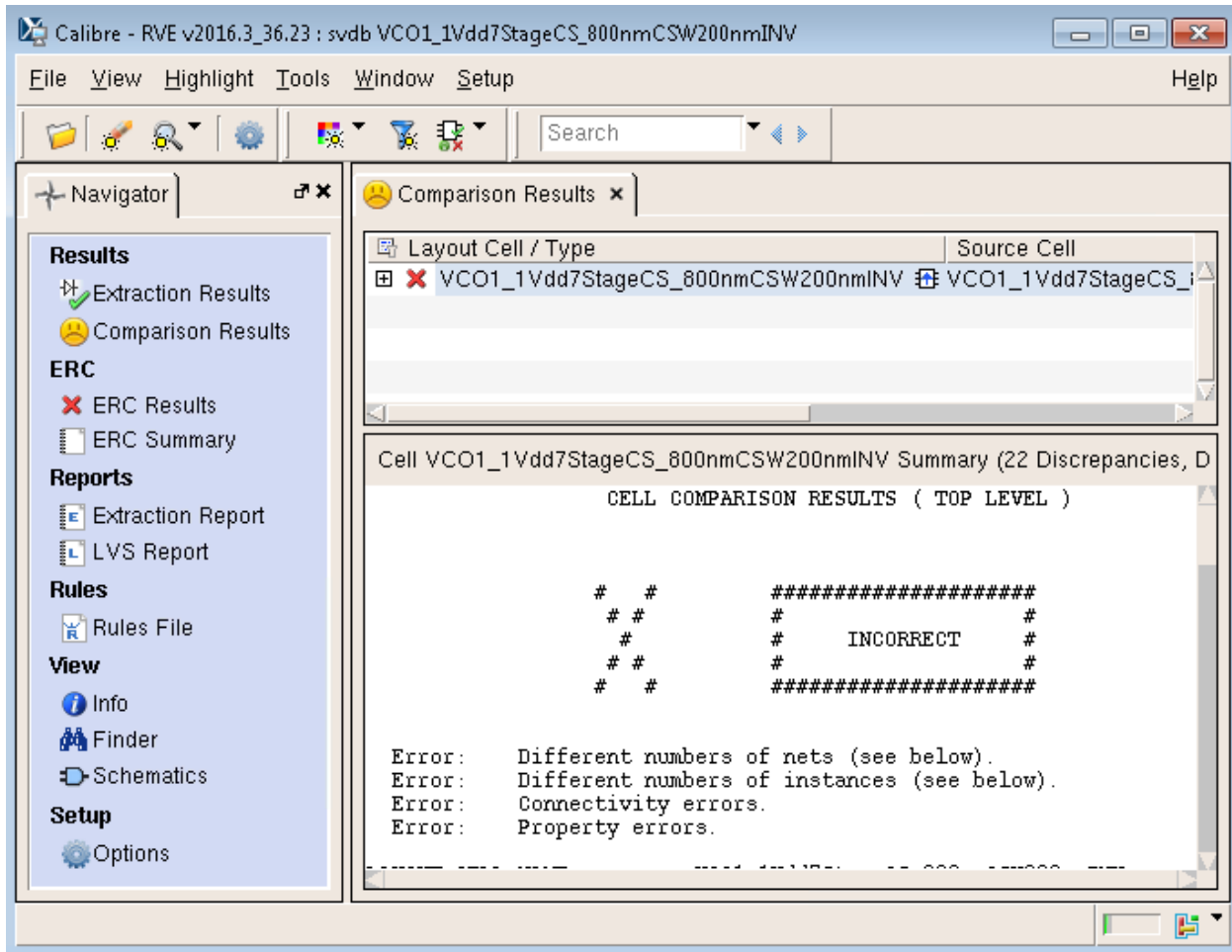
...



The netlist tab contains information about the schematic. It must also specify VCO1. The rest of the defaults are fine, so feel free to press *Run LVS*. Wait while it loads. This is also time consuming. Once it finishes, three windows open: Calibre Interactive – nmLVS, LVS Report File, and Calibre – RVE. Once again, warnings are negligible. Make sure to fix all errors.

If there is an issue, you will get a red unhappy face with descriptions of the error. Shorting all of the inverter PMOS gates with their drains in the layout causes an LVS error, as the schematic no longer matches the layout. The number of nets changes, where the schematic has more nets than the layout.

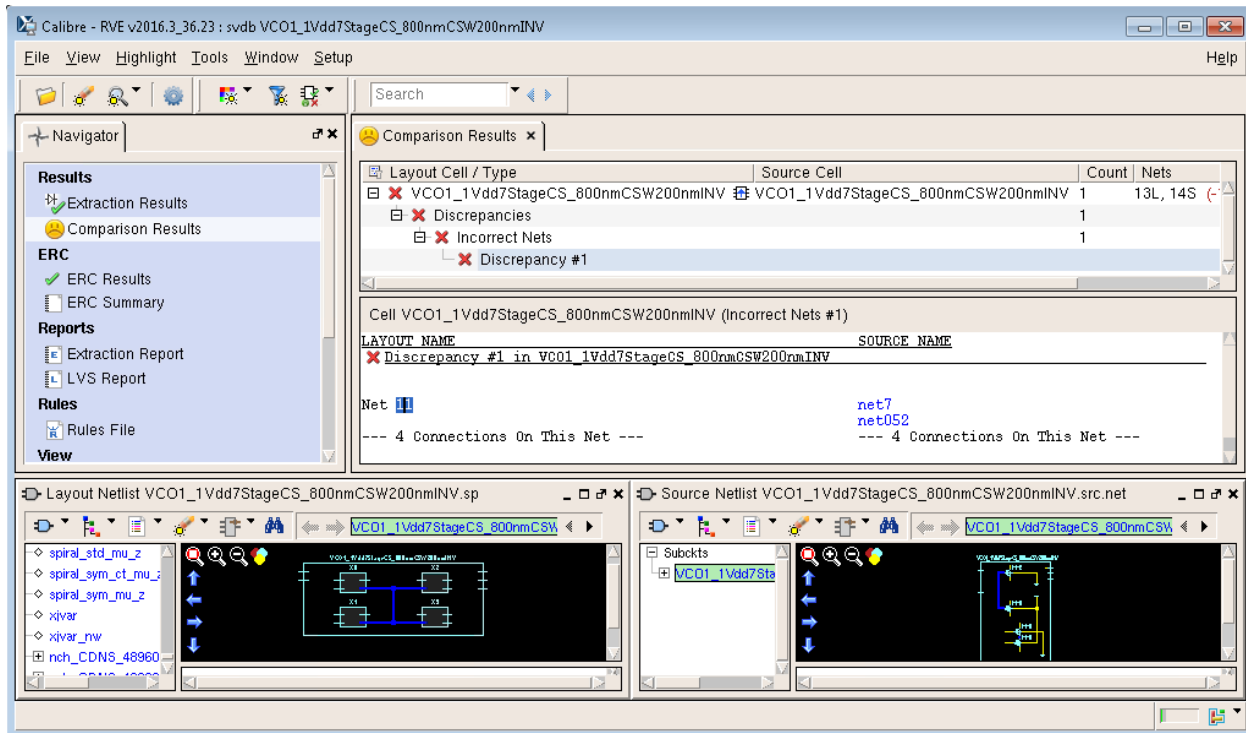
• • •



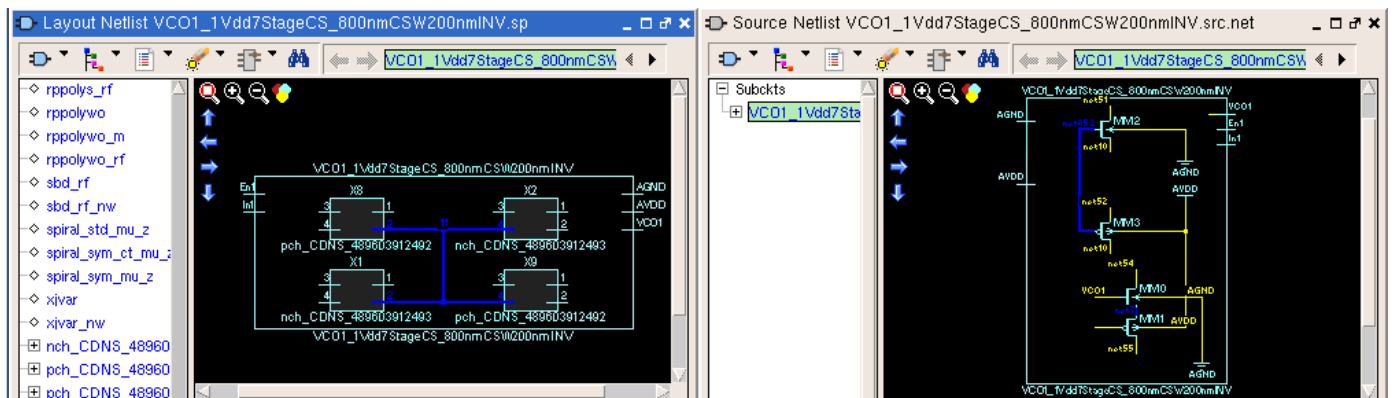
The other issue can be with the schematic. Say for instance the first and second stage of inverters are disconnected. Should this happen, the schematic (referred to as source) will have more nets. For more detailed information, expand the Comparison Results by pressing +. For the first discrepancy, note the different number of nets. Press one of the net values and two diagrams appear, one of layout and the other of the schematic.

VCO-BASED ADC

...



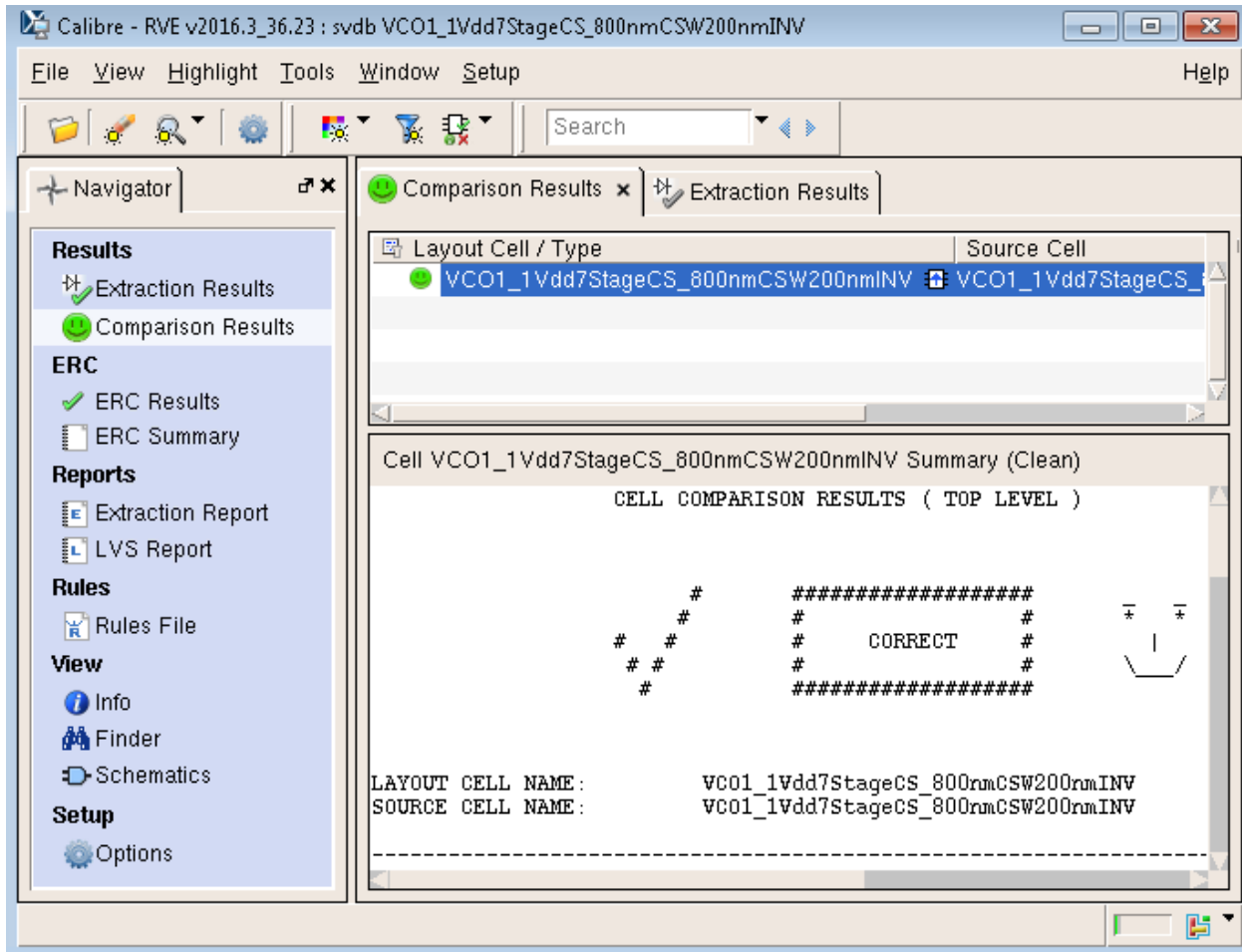
Note the net names. This will help direct you to the exact location of the problem in both the layout and the schematic.



If all is well and the layout and schematic match perfectly, you will get a green smiley face.

VCO-BASED ADC

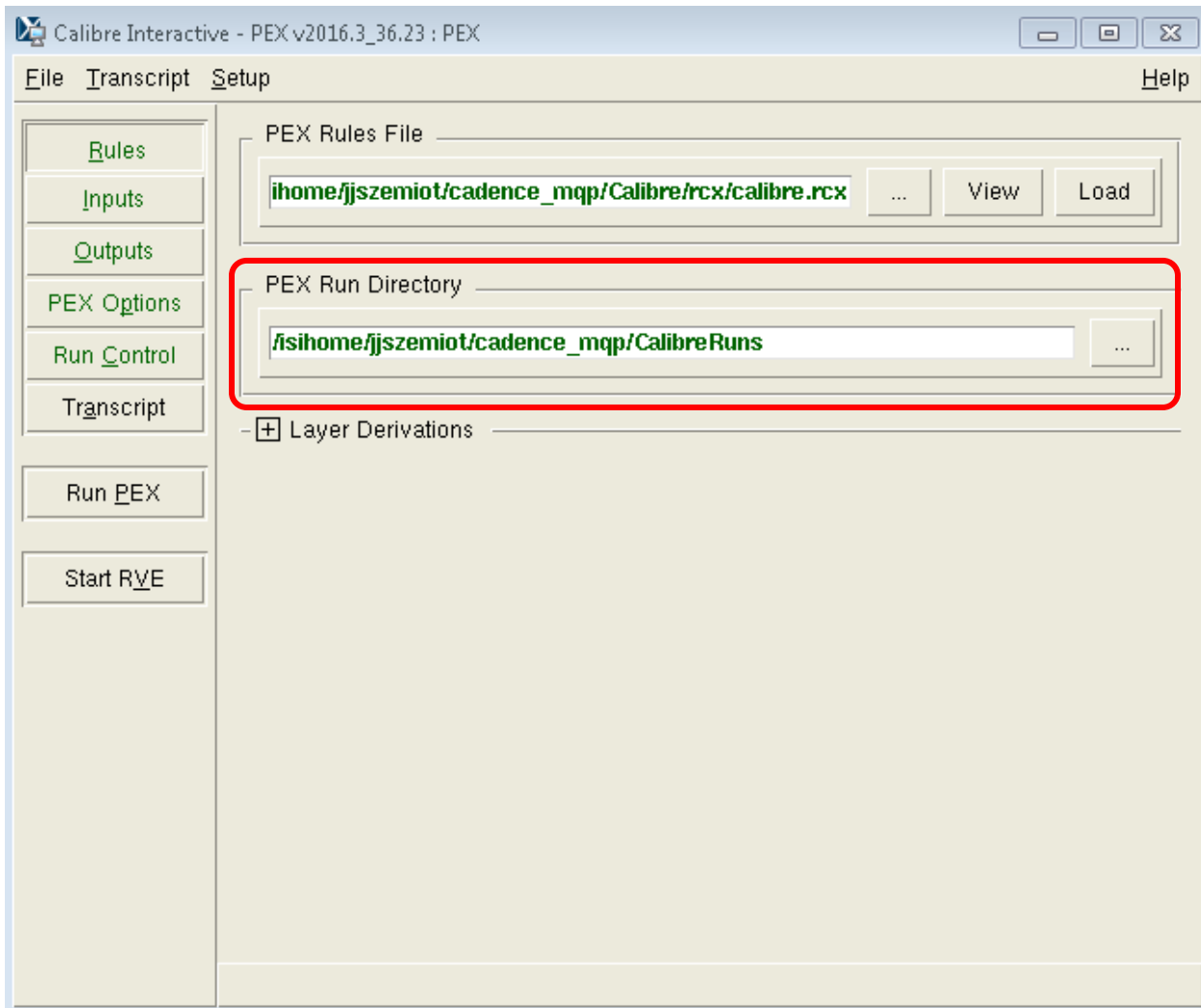
• • •



• • •

Parasitic Extraction

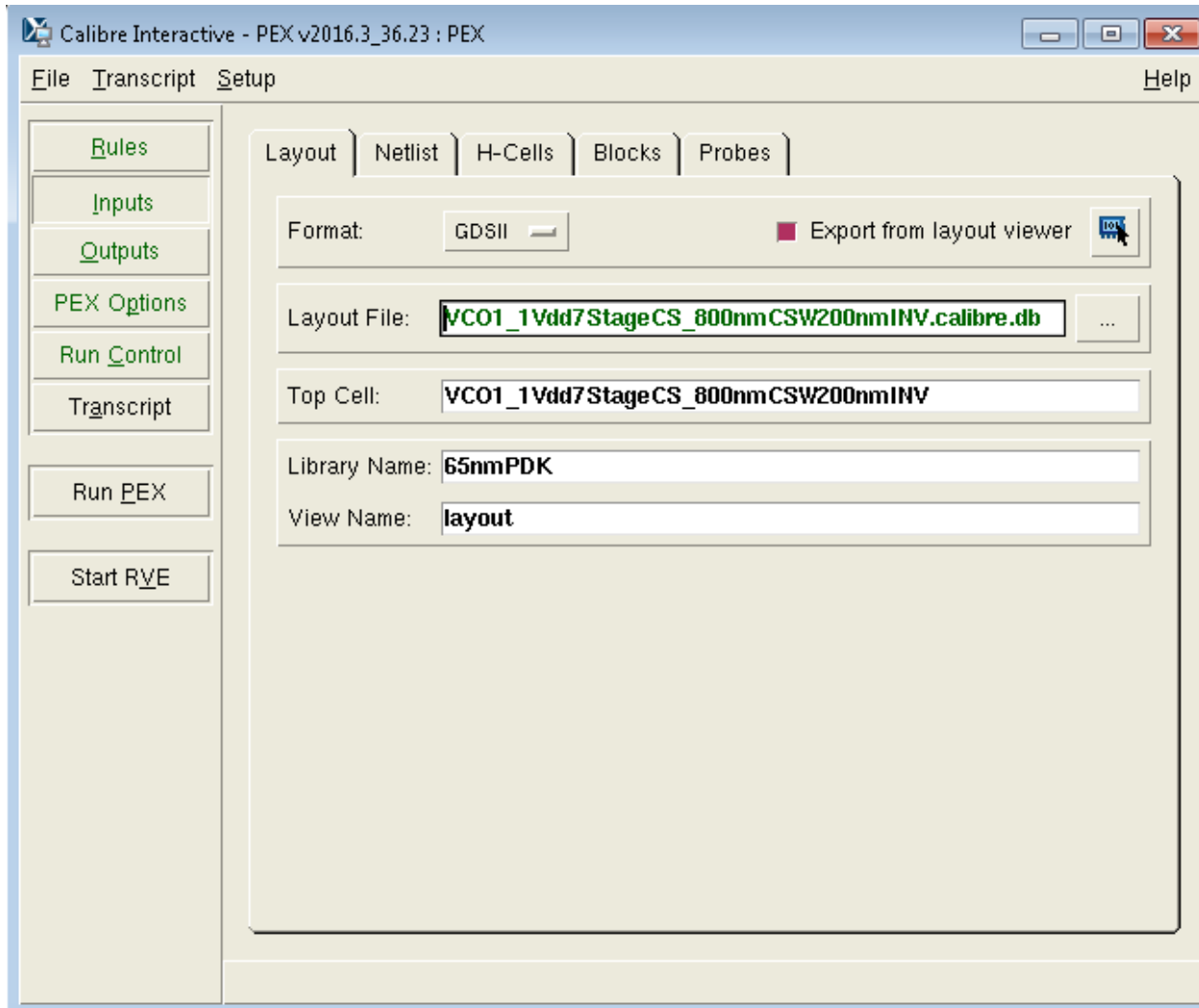
Now you can proceed to Parasitic Extraction (PEX) by selecting *Calibre* -> *Run PEX*. The Calibre Interactive – PEX window opens. Once again you need to identify the appropriate rules file and specify the inputs. The rules file was yet again in */share/apps/cadence/TSMC65gp/Calibre*, this time in the RCX folder. It should be called something similar to *calibre.rcx*. It is beneficial to use another directory, such as *CalibreRun*, for the PEX Run Directory where all of the reports will go.



This verification has more steps, and thus, much more could go wrong. By default, PEX Options are hidden so select *Setup* -> *PEX Options* to reveal them. There are a lot of other things to fill in, so follow the snips provided. The names and locations will be different, but the file type is key.

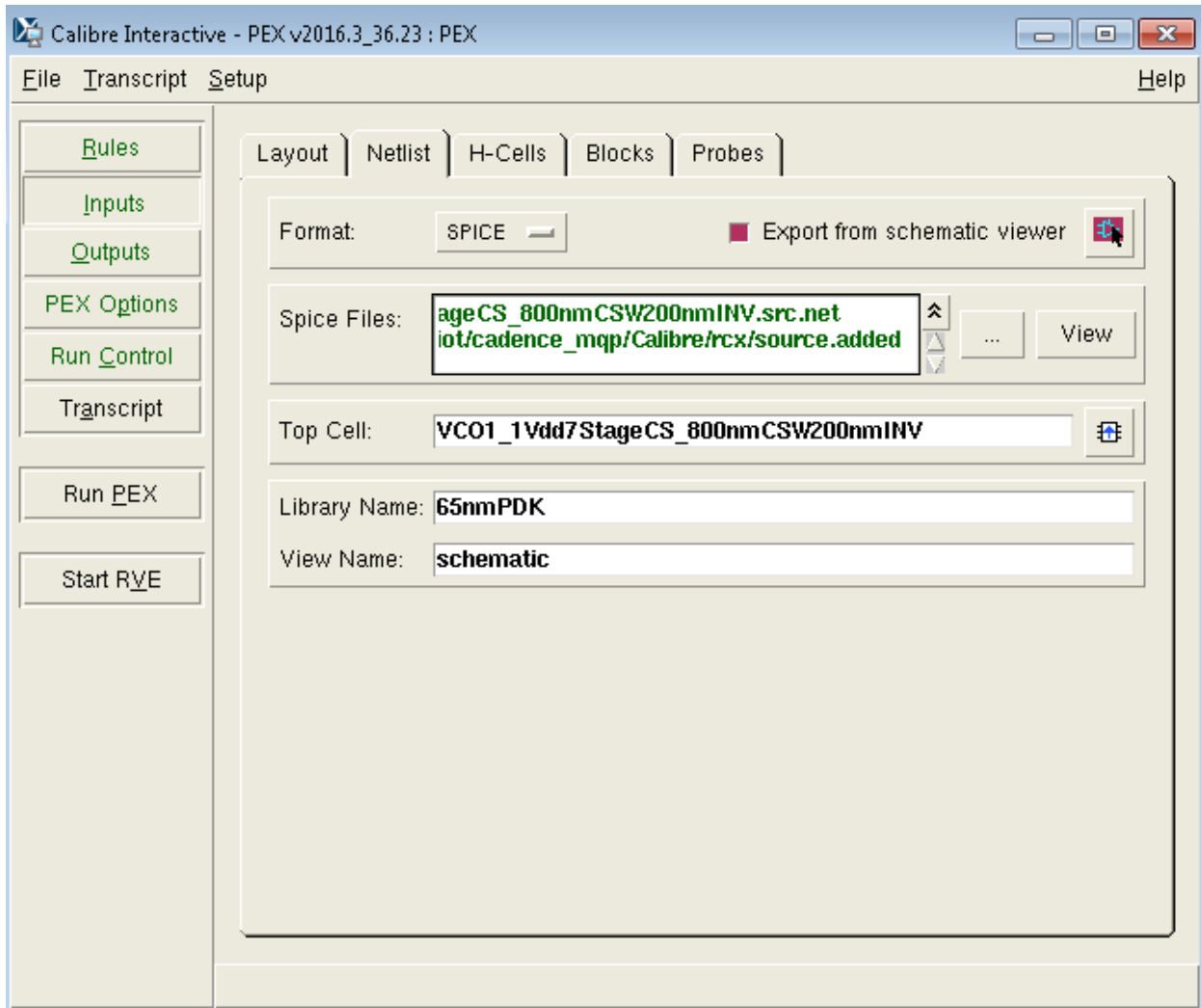
VCO-BASED ADC

...



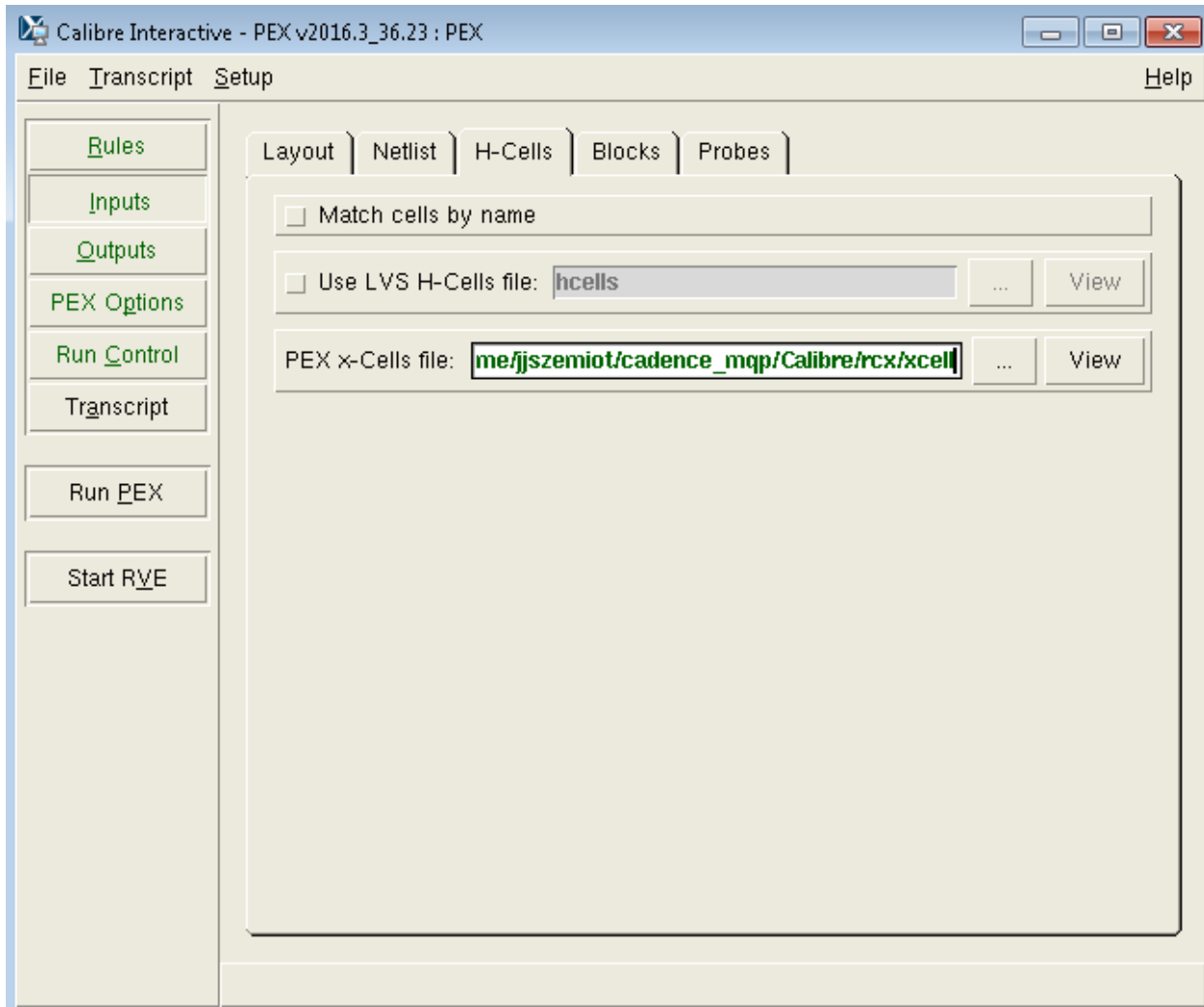
VCO-BASED ADC

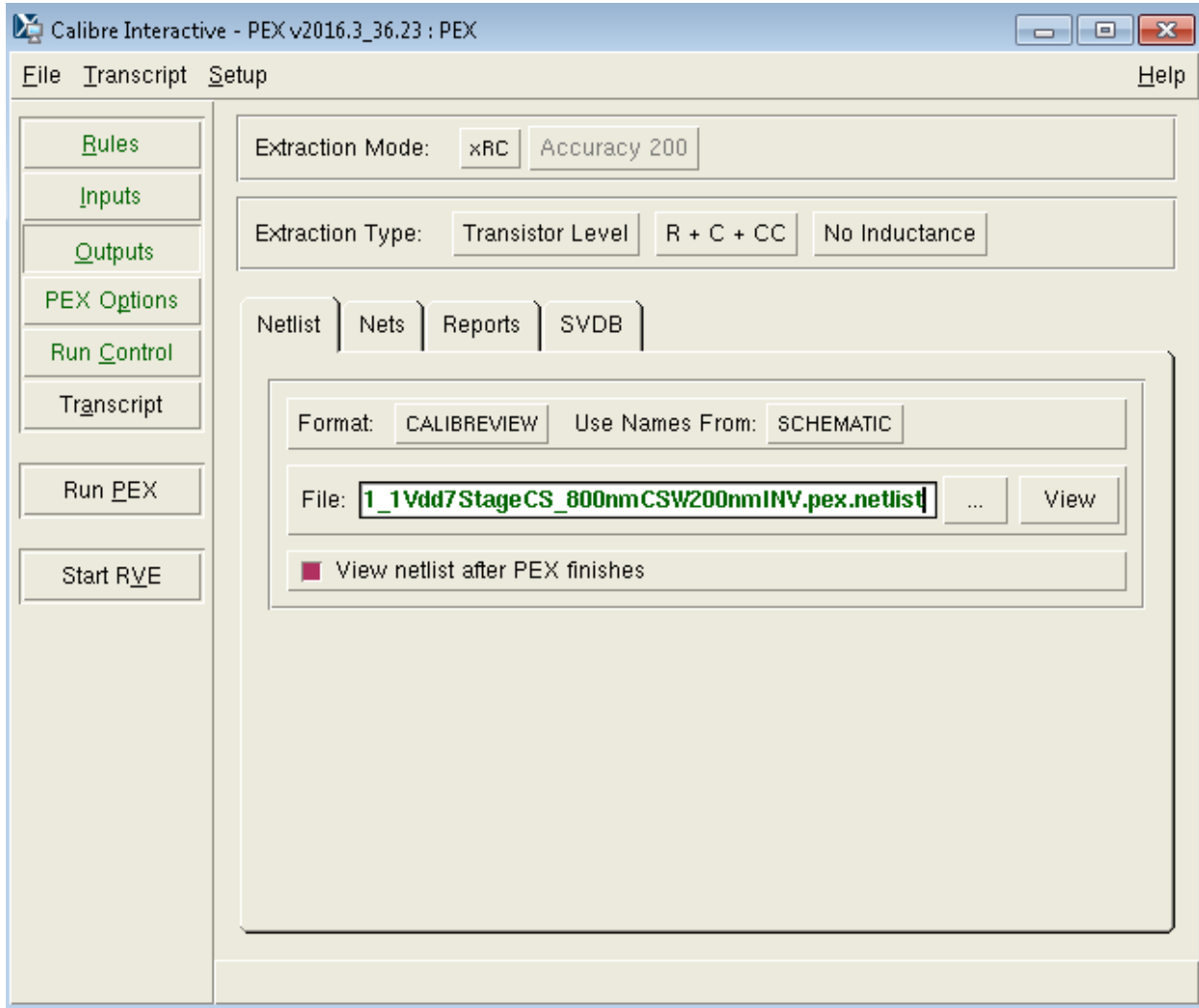
...



VCO-BASED ADC

...

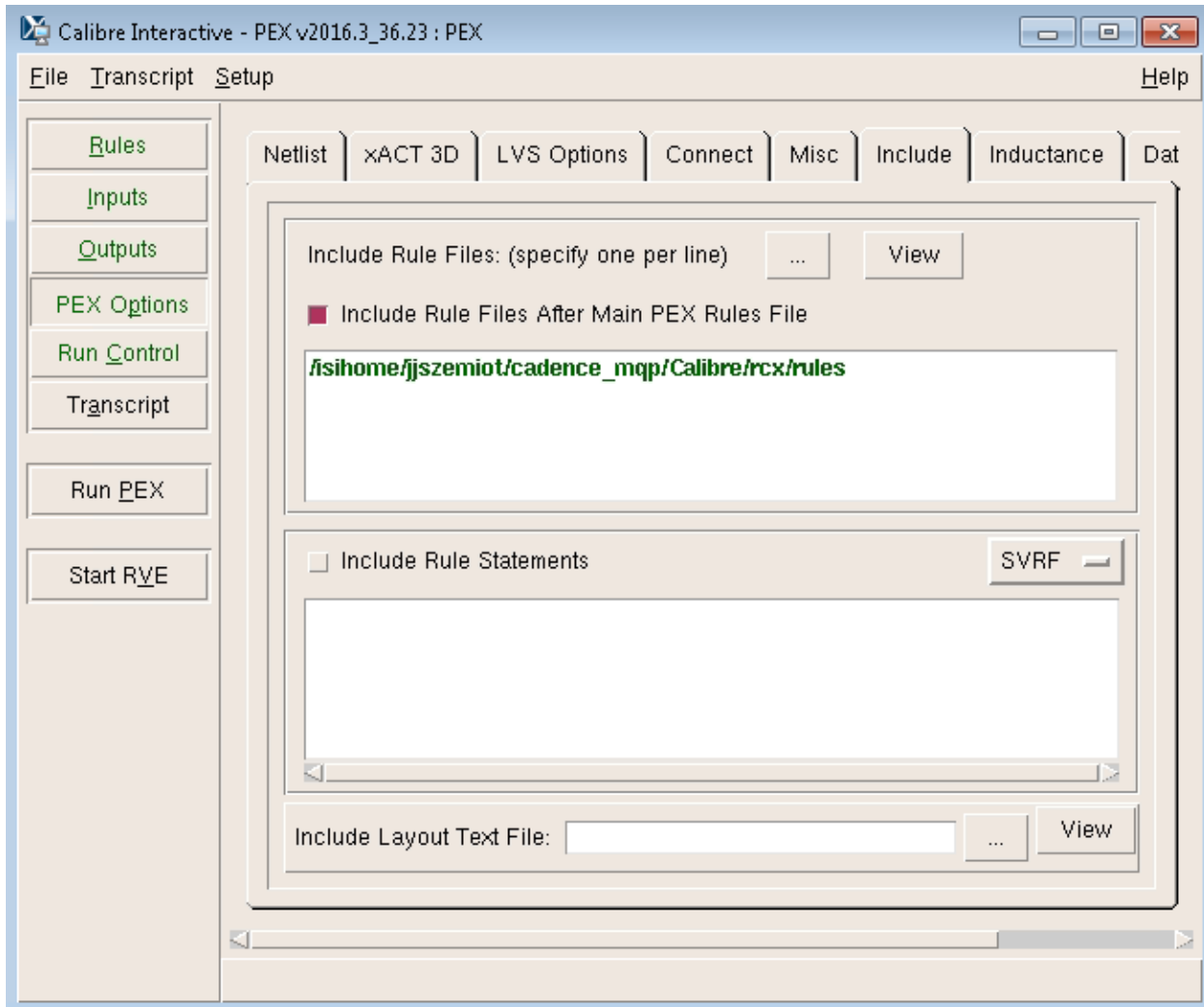




Try pressing *Run PEX*. Overwrite files as necessary. Wait. Warnings are negligible. If the rules file gives you an error regarding an *include ./rules*, edit the *calibre.rcx* file by commenting out this line and including *rules* separately, as below.

VCO-BASED ADC

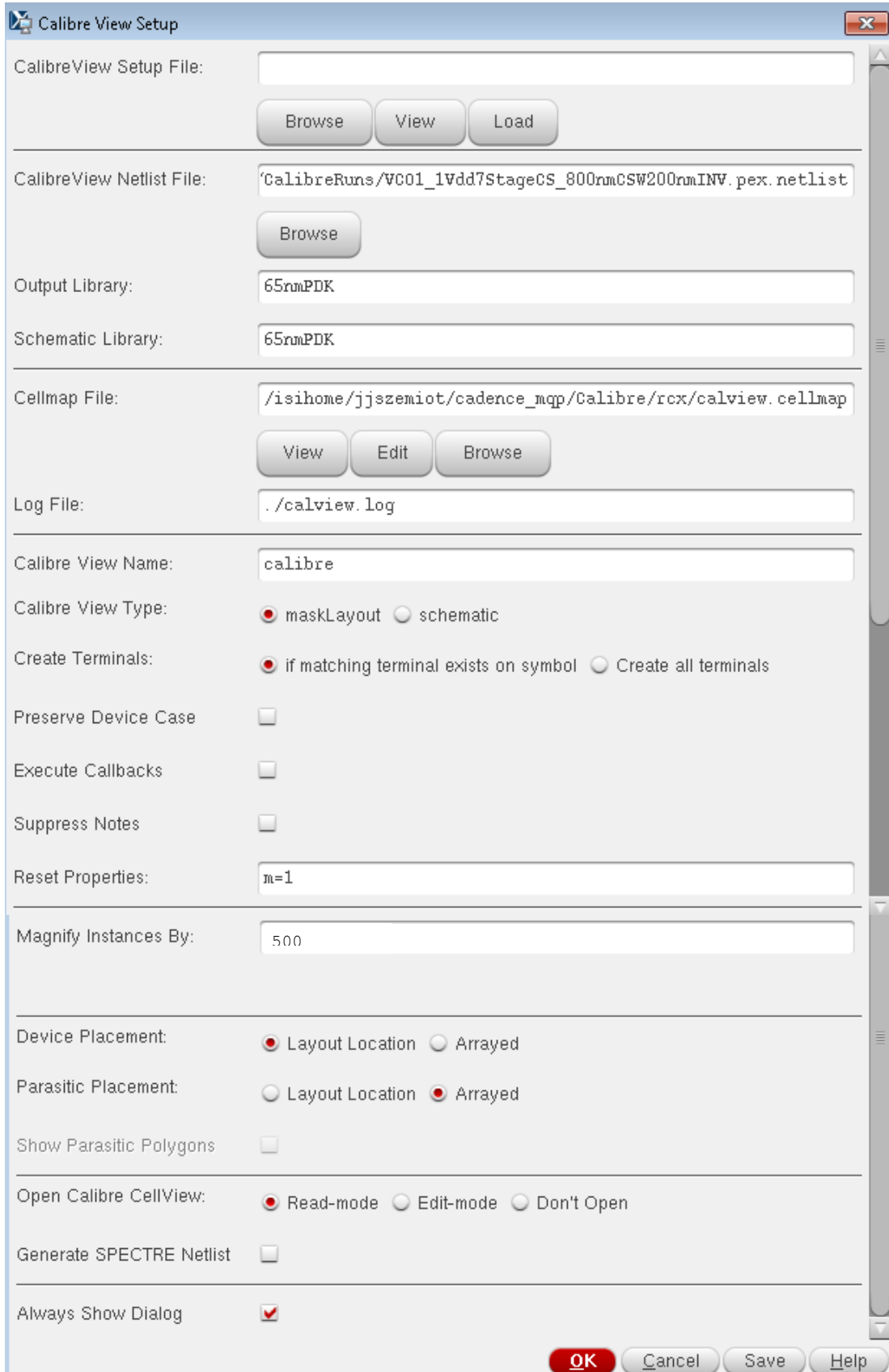
• • •



Once it runs successfully, Calibre View Setup will open. Fill it out as outlined below (magnification of 500 is determined by trial and error to eliminate all errors from the resulting calibre cellview).

VCO-BASED ADC

...



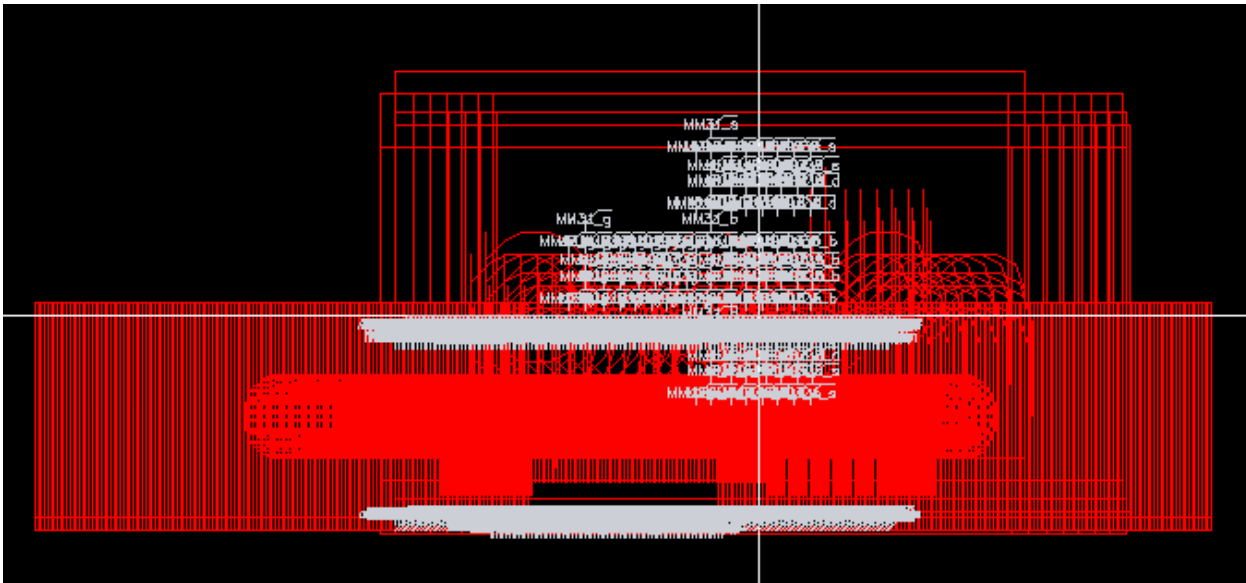
The image shows a 'Calibre View Setup' dialog box with various configuration options. The fields are as follows:

- CalibreView Setup File:** Empty text field with buttons: Browse, View, Load.
- CalibreView Netlist File:** 'CalibreRuns/VC01_1Vdd7StageCS_800nmCSW200nmINV.pex.netlist' with a Browse button.
- Output Library:** '65nmPDK'.
- Schematic Library:** '65nmPDK'.
- Cellmap File:** '/isihome/jjszemiot/cadence_mqp/Calibre/rcx/calview.cellmap' with buttons: View, Edit, Browse.
- Log File:** './calview.log'.
- Calibre View Name:** 'calibre'.
- Calibre View Type:** Radio buttons: ☒ maskLayout, ☐ schematic.
- Create Terminals:** Radio buttons: ☒ if matching terminal exists on symbol, ☐ Create all terminals.
- Preserve Device Case:** ☐.
- Execute Callbacks:** ☐.
- Suppress Notes:** ☐.
- Reset Properties:** 'm=1'.
- Magnify Instances By:** '500'.
- Device Placement:** Radio buttons: ☒ Layout Location, ☐ Arrayed.
- Parasitic Placement:** Radio buttons: ☐ Layout Location, ☒ Arrayed.
- Show Parasitic Polygons:** ☐.
- Open Calibre CellView:** Radio buttons: ☒ Read-mode, ☐ Edit-mode, ☐ Don't Open.
- Generate SPECTRE Netlist:** ☐.
- Always Show Dialog:** ☒.

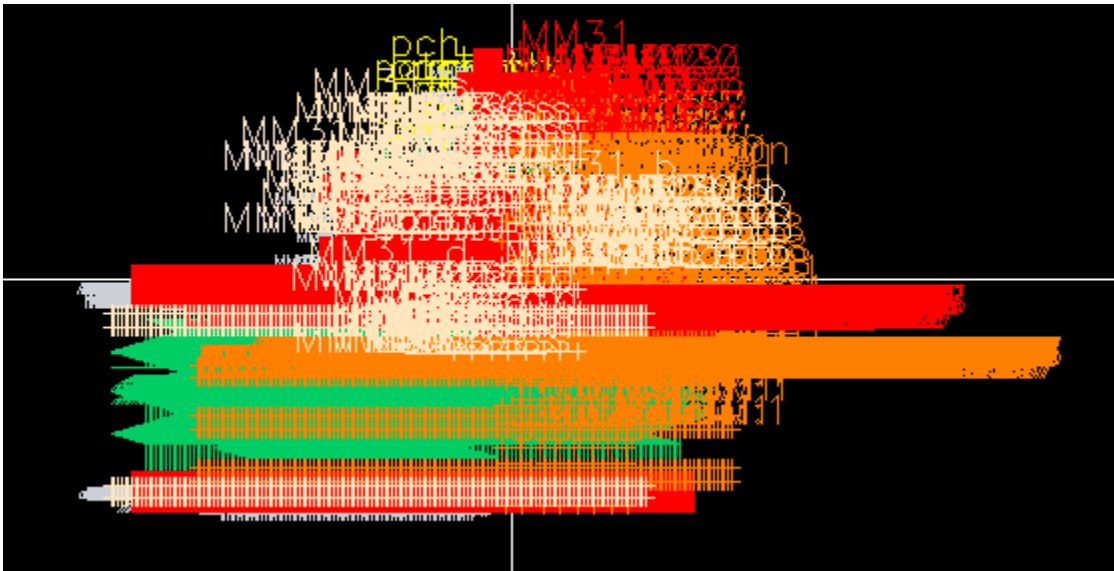
Buttons at the bottom: OK (highlighted in red), Cancel, Save, Help.

• • •

Once the Calibre cellview opens, it should look something like this. Some portions are hidden, so to reveal them, press shift+f.



After pressing shift+f, you will see something similar to the following. It includes all of the extracted parasitic capacitances and resistances. Now you can move on to post-layout simulation.



Post-Layout Simulation

The next step is making a configuration view (config). The instructions for this step are poorly explained. This allows you to use the extracted view for more accurate simulations prior to system level simulation.

Chapter 5: Digital Layout and Synopsys

In this chapter, the relative procedure of creating digital layout of this MQP is covered step by step. The digital part of the ADC consists of MUX and ripple counter. Even though, the components can be implemented using Analog circuits, it's far convenient to create digitally as cadence automatically translate the flip-flops etc. into analog circuitry, hence saving significant amount of time. The steps required generally to create a digital layout using Synopsys Design Vision and cadence encounter softwares are listed below:

- Create Verilog code of digital components
- Synthesize digital components using Synopsys Design Vision
- Creating the layout using cadence encounter

Create Verilog Code for digital components

This is the simplest step in the procedure of creating digital layout. The digital components of the circuit must be written using Verilog hardware description language. The code can be as simple as one module or more than one module with a top module. For instance, the digital components of this MQP include a 4-to-1 MUX and a ripple counter. The ripple counter consists of 11 D Flip-flops. The top module is named MQP_CNT which virtually connects the MUX to the Counter. The Verilog code for the digital components is included in **Appendix C**.

Synthesize digital components using Synopsys Design Vision

Next, after the Verilog code is created, Synopsys Design Vision is used to synthesize the digital circuitry. First a folder named "digital" is created on Rambo server and made the current directory. Following commands are typed in the terminal to open Synopsys Design Vision.

```
module load synopsys
```

```
design_vision &
```

Once the software is opened, we can now synthesize the digital circuitry. This procedure is based on our MQP project thus more detailed tutorial can be found here at:

http://coefs.uncc.edu/amukherj/files/2012/04/DesignVisionTutorial_f07.pdf

• • •

First click Setup under File tab in order to setup search path, link library and symbol library.

Search Path

```
/share/apps/cadence/tc65gplus/200a/Front_End/timing_power_noise/CCS
/share/apps/synopsys/syn/libraries/syn/share/apps/synopsys/syn/minpower/syn/share/apps/synopsys/
syn/dw/syn_ver /share/apps/synopsys/syn/dw/sim_ver
```

Link Library

```
*/share/apps/cadence/tc65gplus/200a/Front_End/timing_power_noise/CCS/tc65gplus_200a/tc65g
plustc_ccs.db
```

Target Library

```
/share/apps/cadence/tc65gplus/200a/Front_End/timing_power_noise/CCS/tc65gplus_200a/tc65g
plustc_ccs.db
```

Next Verilog code is analyzed and elaborated which functions can be found under File tab. When analyzing, add all the Verilog code of the design (not just top module). Then elaborate by selecting the top module. Then synthesize the design using default settings by clicking compile design under Design tab. The synthesized circuitry is saved as Verilog file with _syn.v added to the name (for example, "MQP_syn.v").

This Verilog code contains the components from the digital library associated to the process technology. The Figure below shows the digital components used from the digital library.

```
MUX4D0 U1 ( .I0(i[0]), .I1(i[2]), .I2(i[1]), .I3(i[3]), .S0(b),
.S1(a), .Z(q) );
endmodule

module DFLIPFLOP_0 ( clk, reset, d, q, qb );
    input clk, reset, d;
    output q, qb;

    DFCND1 q_reg ( .D(d), .CP(clk), .CDN(reset), .Q(q), .QN(qb) );
endmodule
```

The timing, power and area of the synthesized design can be measured as well. The following exemplary code is typed in the command line to create timing, area and power reports, using the input one since the MUX has 4 inputs.

VCO-BASED ADC

• • •

```
set design MUXCNT

current_design $design

set a 0;
set b 0;
create_clock -period 0.17 -waveform {0 0.15} [get_ports "in[0]"]

set_clock_uncertainty -setup 0.01 [get_clocks "in[0]"]
set_clock_uncertainty -hold 0.01 [get_clocks "in[0]"]
set_clock_transition 0.05 [get_clocks "in[0]"]

set_fanout_load 8 [all_outputs]

set_dont_touch_network [get_ports "in[0]"]

set auto_wire_load_selection true

set_fix_multiple_port_nets -all -buffer_constants
compile_ultra -retime -timing

compile_ultra -incremental -area

change_names -rules verilog -hierarchy
write -f verilog -hierarchy -output "netlists/${design}.v"

write_sdc "netlists/${design}.sdc"
redirect "reports/${design}.area0" { report_area }
redirect "reports/${design}.tim0" { report_timing }
redirect "reports/${design}.power0" { report_power }
```

The code for obtaining the reports of the rest of the inputs are included in **APPENDIX D**. Only run for one input at each time.

Creating the layout using Cadence Encounter

Next, open Cadence Encounter in the same directory. This can be done by typing the following commands in the terminal.

```
module load cadence
```

```
encounter &
```

Once the software is loaded, import the synthesized Verilog code by clicking import design under the file tab. The Verilog should be the synthesized Verilog code from Synopsys, for technology/physical libraries, select LEF files and fill the box with the following:

• • •

/share/apps/cadence/tc65gplus/200a/Back_End/lef/tc65gplus_200a/lef/tc65gplus_9lmT2.lef

This basically selects the digital library of the technology utilized for this MQP. Once again, this is specific instructions for this project and things might vary accordingly. Type **VDD** for power nets, and **VSS** for ground nets. The following tutorial provides the essential steps necessary to create digital layout using Encounter which the MQP team has greatly referred to progress with the project.

<https://www.youtube.com/watch?v=Z5WKIDbthdg&t=1006s>

Running DRC on MQP project gave via errors at first. These were eliminated by using “nano-route” under Route tab and selecting “Route” and “optimize via”. Another error was that when doing special route, the standard cells and power strips overlap each other, which results in shorting. Since the MQP digital circuitry is simple and minute, it was decided power strips would not be necessary since all cells can reach the power rails. This eliminated the DRC problems and finally the circuit is working.

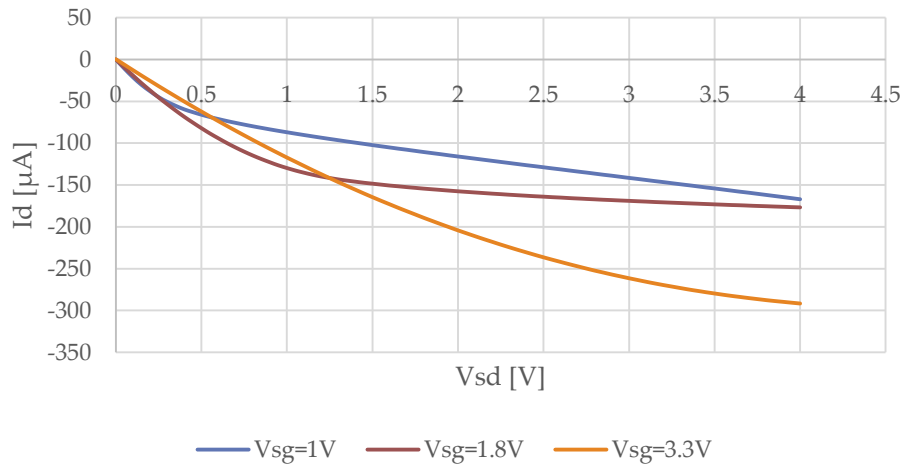
Finally, save the digital layout as encounter file as well as GDS, the latter is necessary to import the digital layout in Cadence Virtuoso.

APPENDIX B: Comprehensive Simulation Results

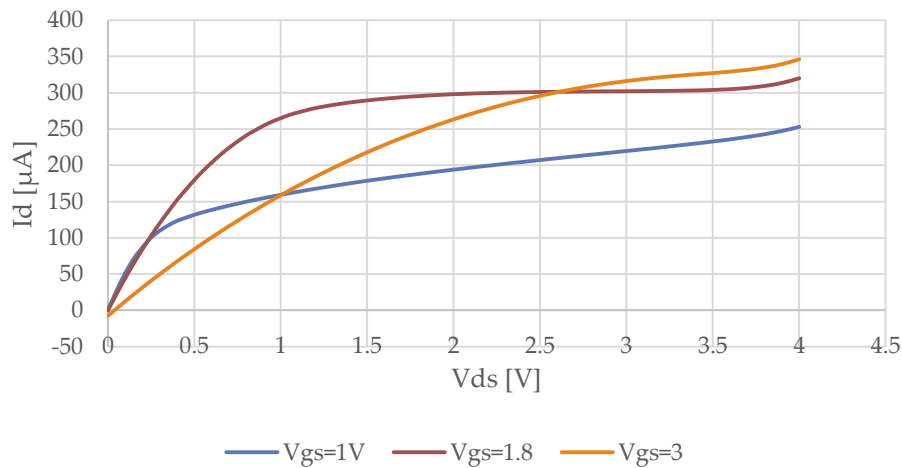
MOSFET V-I characteristics

The V-I characteristics of the MOSFETs were examined in Cadence in their standard PMOS and NMOS test circuits. The drain current was plot against the source-drain and drain source voltages to analyze the transconductance behavior of the PMOS and NMOS respectively.

PMOS baseline V-I Characteristics



NMOS baseline V-I characteristics



The transconductance of the MOSFETs measured in the triode region gave insight into some general behavior. For example, inverting the transconductance provided approximations of the general resistive behavior at varying gate-source voltages. The on resistance for each device was on the order of 1 k Ω .

• • •

There were unresolved problems encountered while measuring V-I characteristics on the NMOS; they relate to the general behavior past 5V drain-source voltage and a 3V gate-source voltage limit. When looking at the V-I characteristics at 5V and beyond, the behavior no longer looks like a MOSFET and more like a diode. The drain current proceeds to increase exponentially at around 5V instead of continuing to exhibit saturation behavior. The 3V gate-source voltage limit meant a failure to compute V-I characteristics greater than 3V on NMOS test circuit. Whenever the voltage limit was exceeded in testing, Cadence would send an error message of convergence failure.

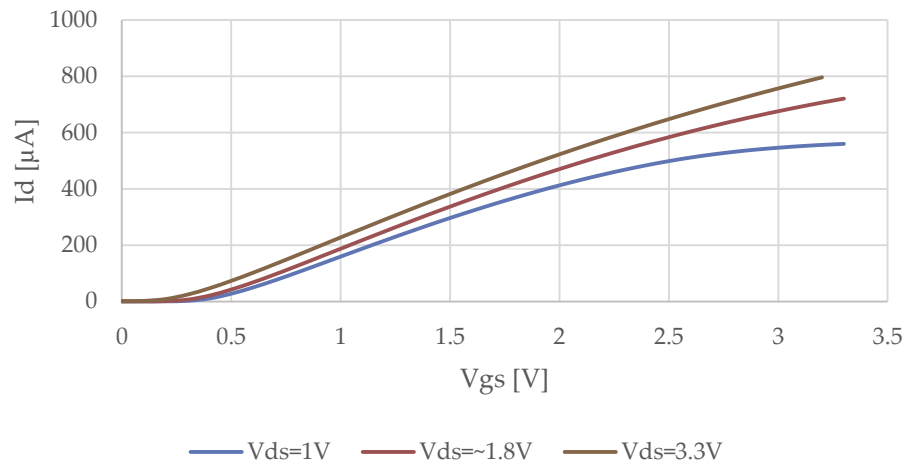
The PMOS was more cooperative in the simulations. While the PMOS had no issues computing the V-I characteristics at a source-gate voltage of 3.3V; its general behavior past 10V source-drain voltage was strange. At 10V, the PMOS show a similar issue to the NMOS past 5V; the drain current's magnitude increase exponentially instead of being linear in saturation.

The V_{gs} vs I_d behavior was as expected. The simulation on cadence is done by sweeping the gate source voltage while holding a fixed drain source voltage at 1V, 1.8V, 2.53V and 3.3V. The saturation behavior of the PMOS and NMOS matched up properly with their theoretical behavior. The threshold voltages appeared to be surprisingly low ($\sim 0.4V$) and the same for both types of MOSFETs.

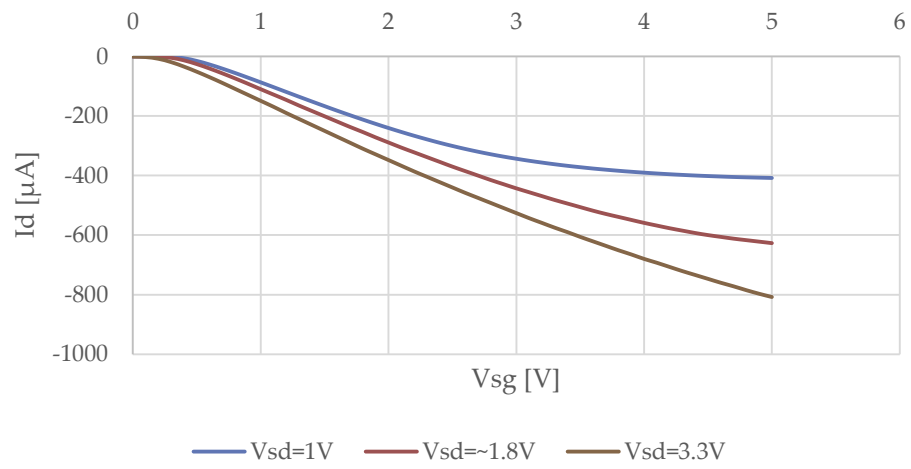
VCO-BASED ADC

• • •

NMOS V_{gs} vs I_d



PMOS V_{sg} vs I_d

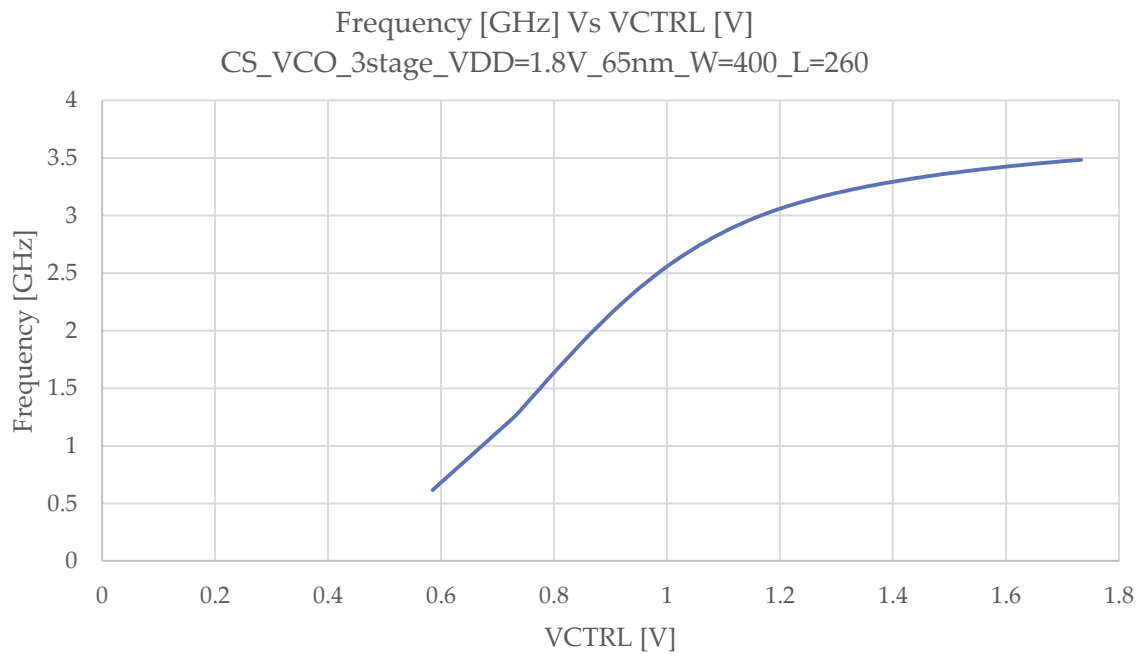


VCO-BASED ADC

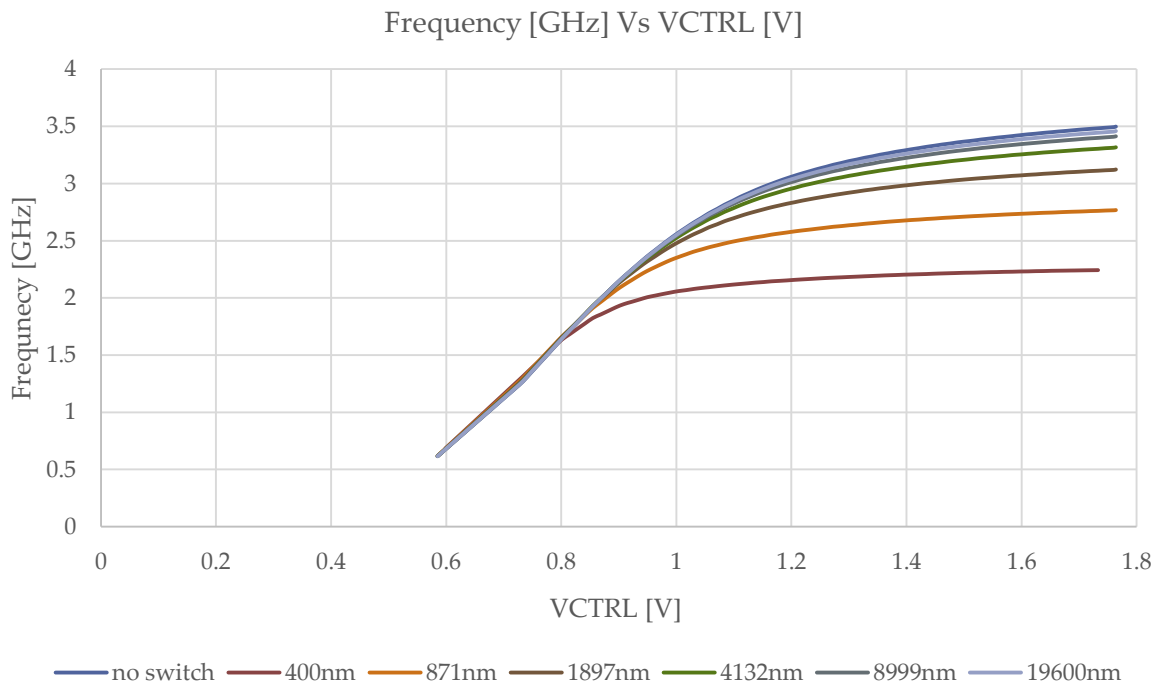
• • •

VCO with 1.8V Supply: 3 Stage

Current Starved VCO



Current Starved VCO with PMOS switch

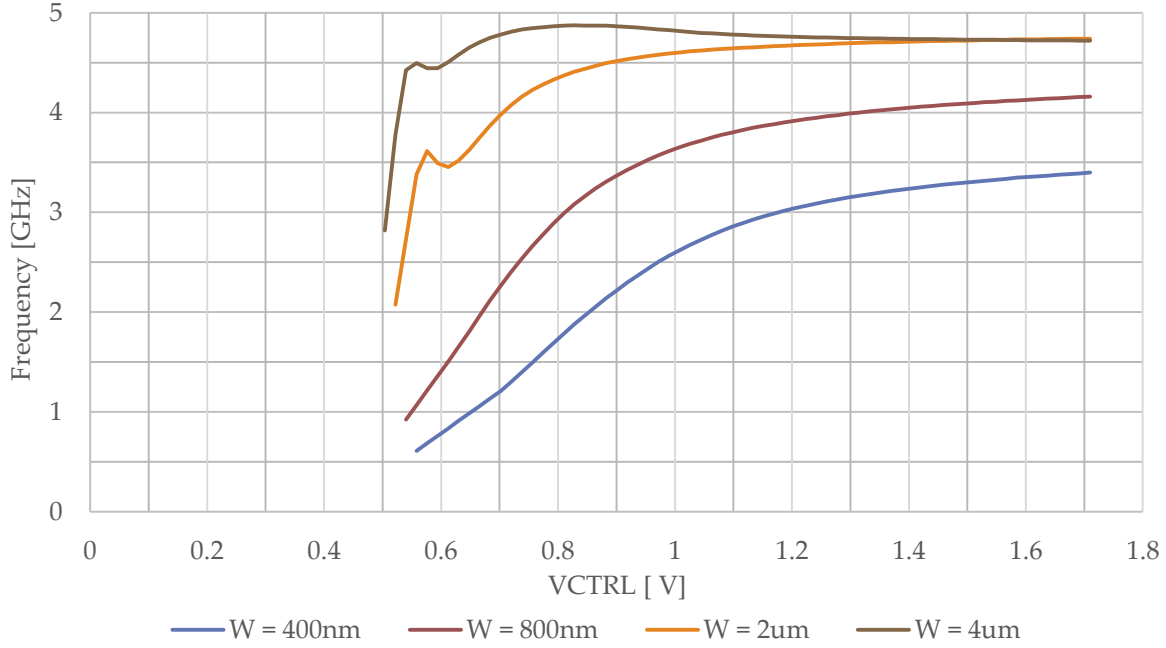


VCO-BASED ADC

• • •

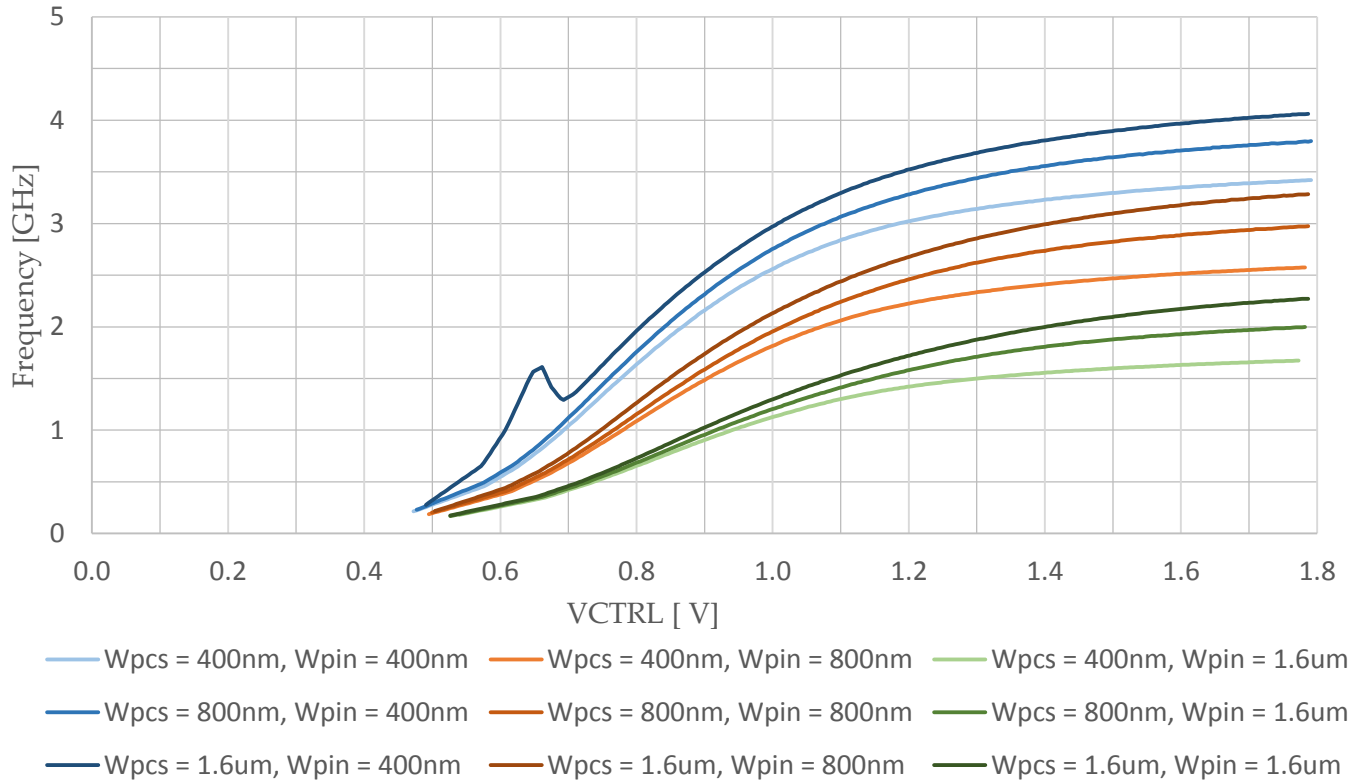
Current Starved VCO Incrementing Width of Current Sourcing and Starving MOS

V-f Characteristics plot of 3 stage, $V_{DD} = 1.8V$, $W [m] = 400n, 800n, 2\mu, 4\mu$



Current Starved VCO Incrementing Width of Current Sourcing and Inverter PMOS

V-f characteristics of 3 stage, $1.8V$ CSVCO with different widths of CS PMOS and inverter PMOS



• • •

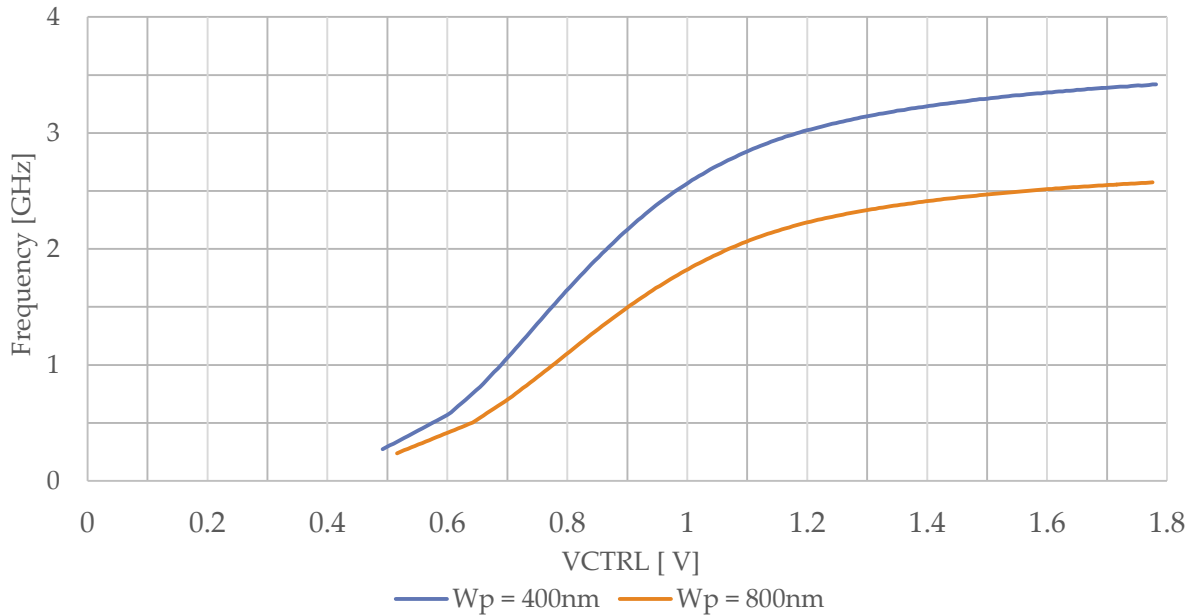
Rise and fall time of 1.8V 3 stage CSVCO

Rise Time (ps)	W _{pin}		
W _{cs}	400nm	800nm	1.6μm
400nm	114.6	155.8	249.6
800nm	99.7	133.4	212.7
1.6μm	90.3	122	193.1

Fall Time (ps)	W _{pin}		
W _{cs}	400nm	800nm	1.6μm
400nm	82	103.2	145.3
800nm	79.1	98.8	138.1
1.6μm	76.94	95.5	133.9

Current Starved VCO with Inverter PMOS Width twice that of Inverter NMOS Width

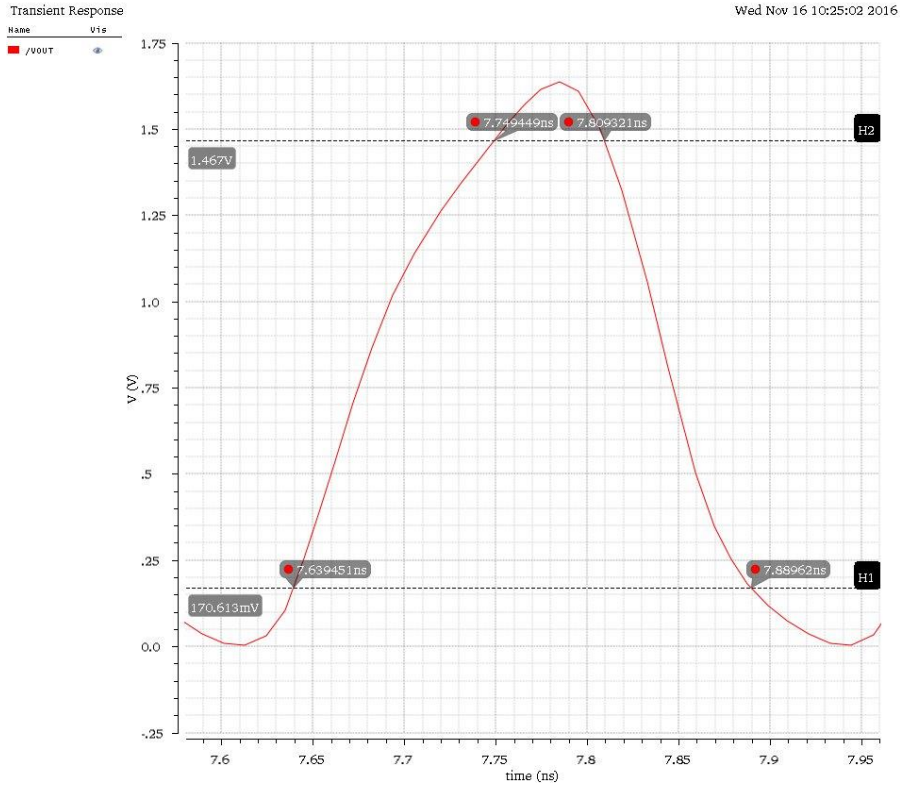
V-f Characteristics plot of 3 stage, VDD = 1p8V

Blue is baseline , Orange is when W_{pmos} (inverter) = 2W_{nmos} (inverter)

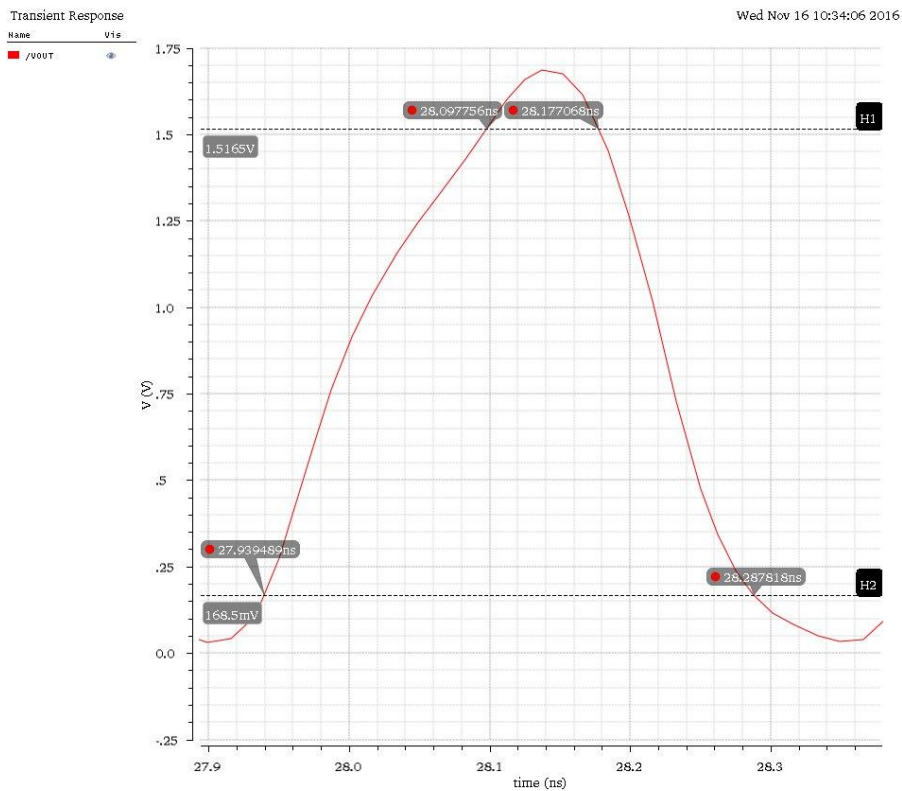
VCO-BASED ADC

• • •

W = 400nm , rise time = 100 ps, fall time = 80 ps

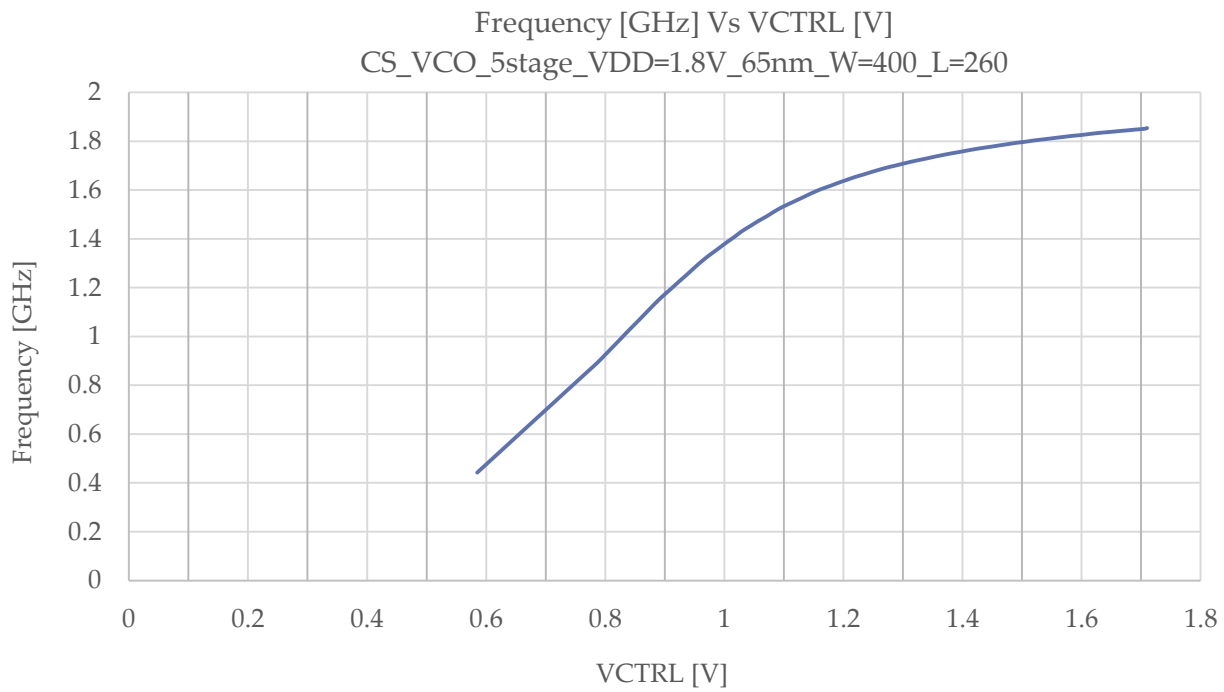


W = 800nm , rise time = 158.4 ps ,fall time = 110.7 ps

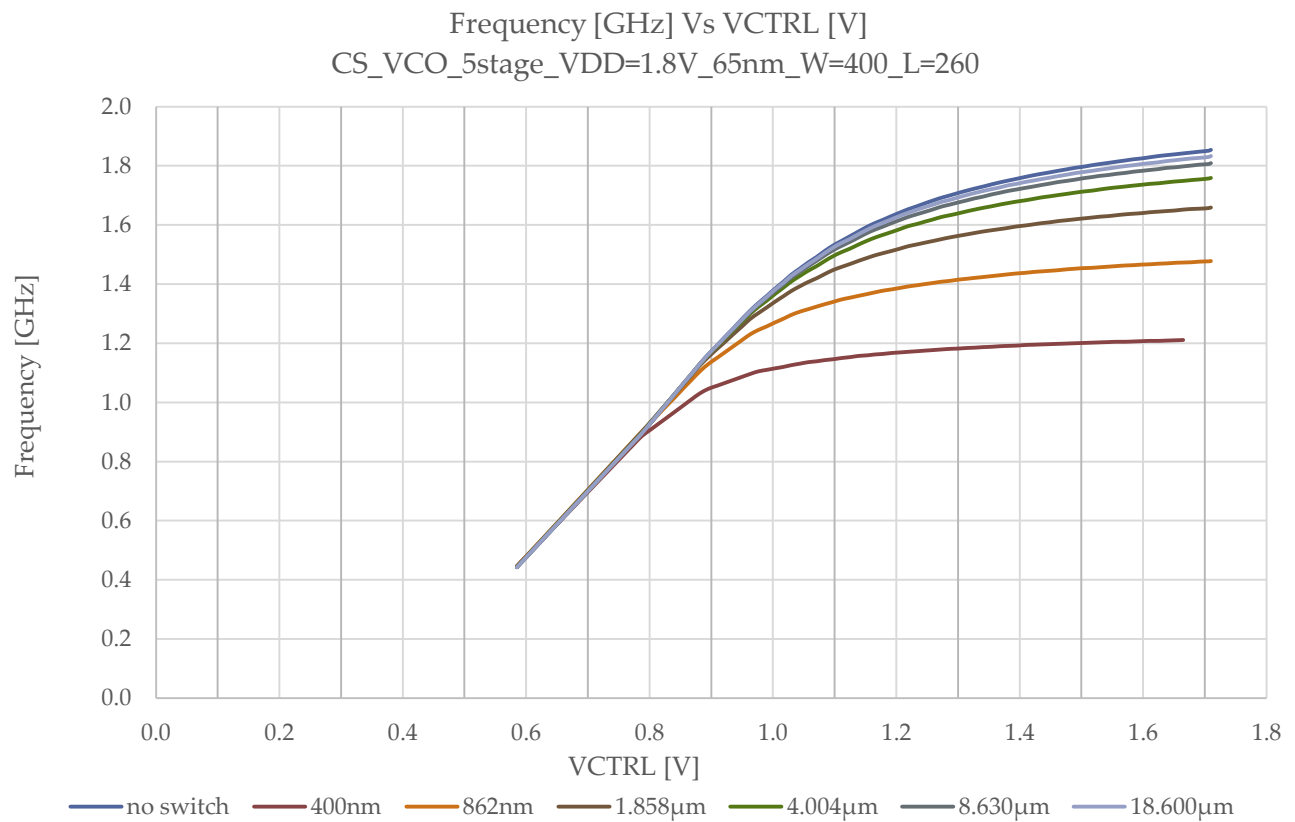


VCO with 1.8V Supply: 5 Stage

Current Starved VCO



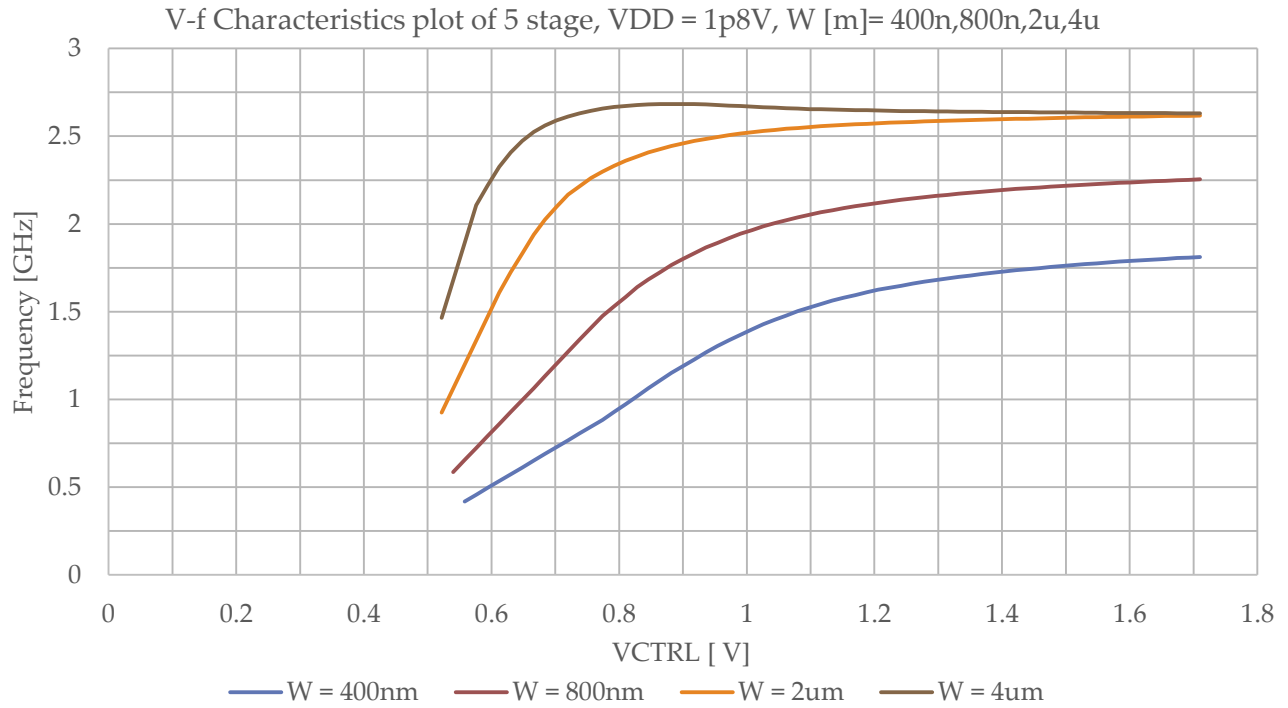
Current Starved VCO with PMOS switch



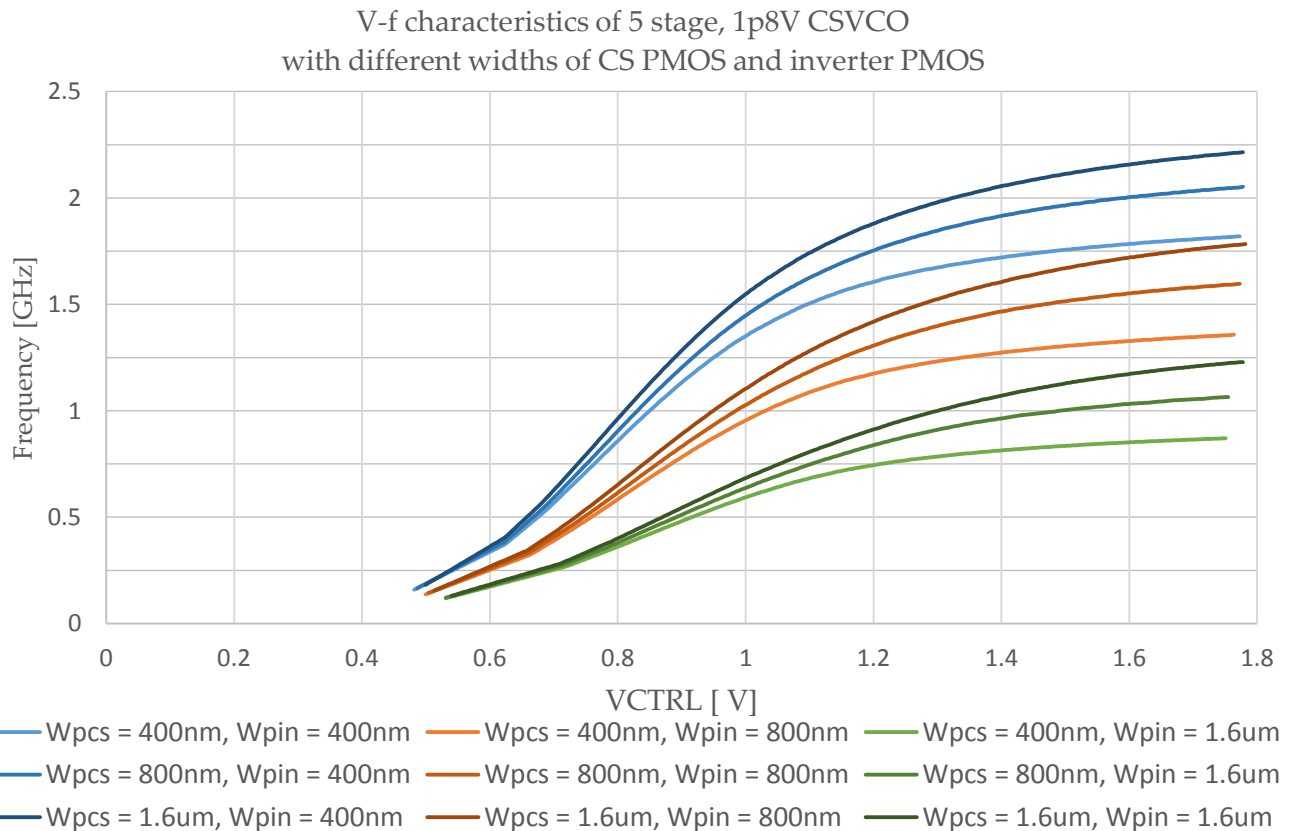
VCO-BASED ADC

• • •

Current Starved VCO Incrementing Width of Current Sourcing and Starving MOS



Current Starved VCO Incrementing Width of Current Sourcing and Inverter PMOS



VCO-BASED ADC

• • •

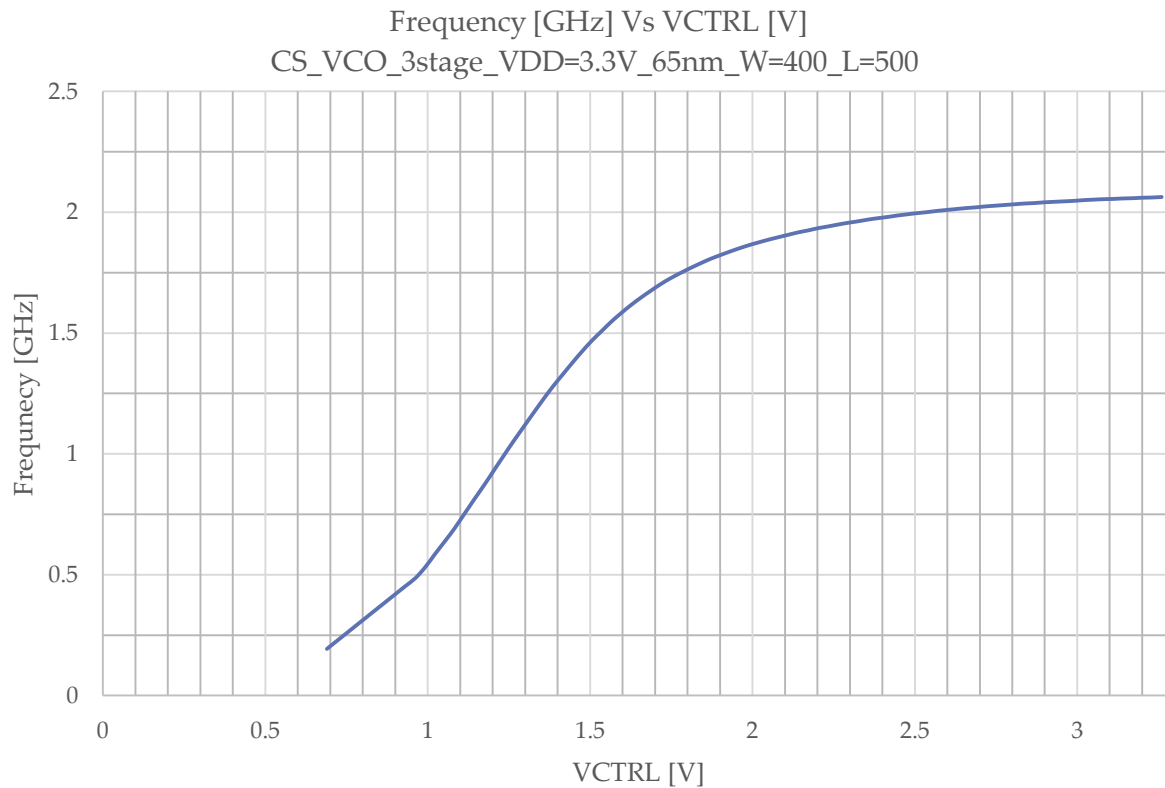
Rise and fall time of 1.8V 5 stage CSVCO

Rise Time (ps)	Wpin		
Wcs	400nm	800nm	1.6um
400nm	241.1	365.9	631.8
800nm	218.8	323.8	542
1.6um	202.7	295.1	481.3p

Fall Time (ps)	Wpin		
Wcs	400nm	800nm	1.6um
400nm	134.7	182.2	273.6
800nm	134.1	178.9	268.6
1.6um	134.3	179.7	268.1

VCO with 3.3V Supply: 3 Stage

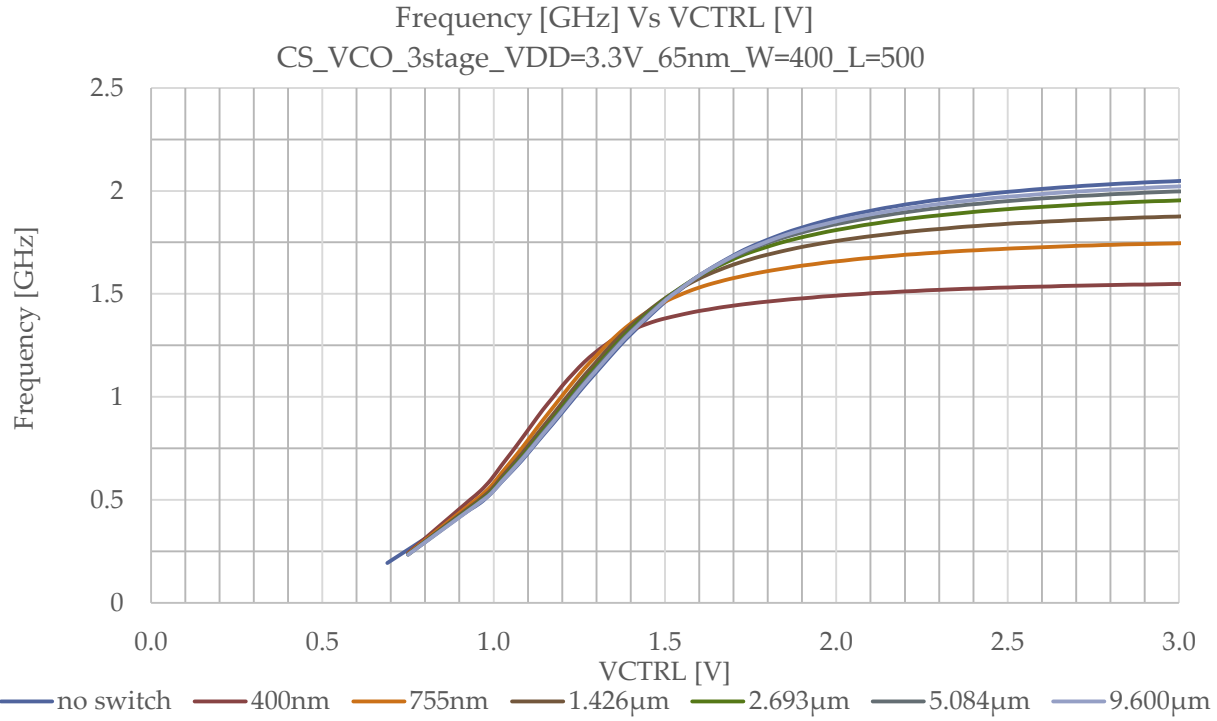
Current Starved VCO



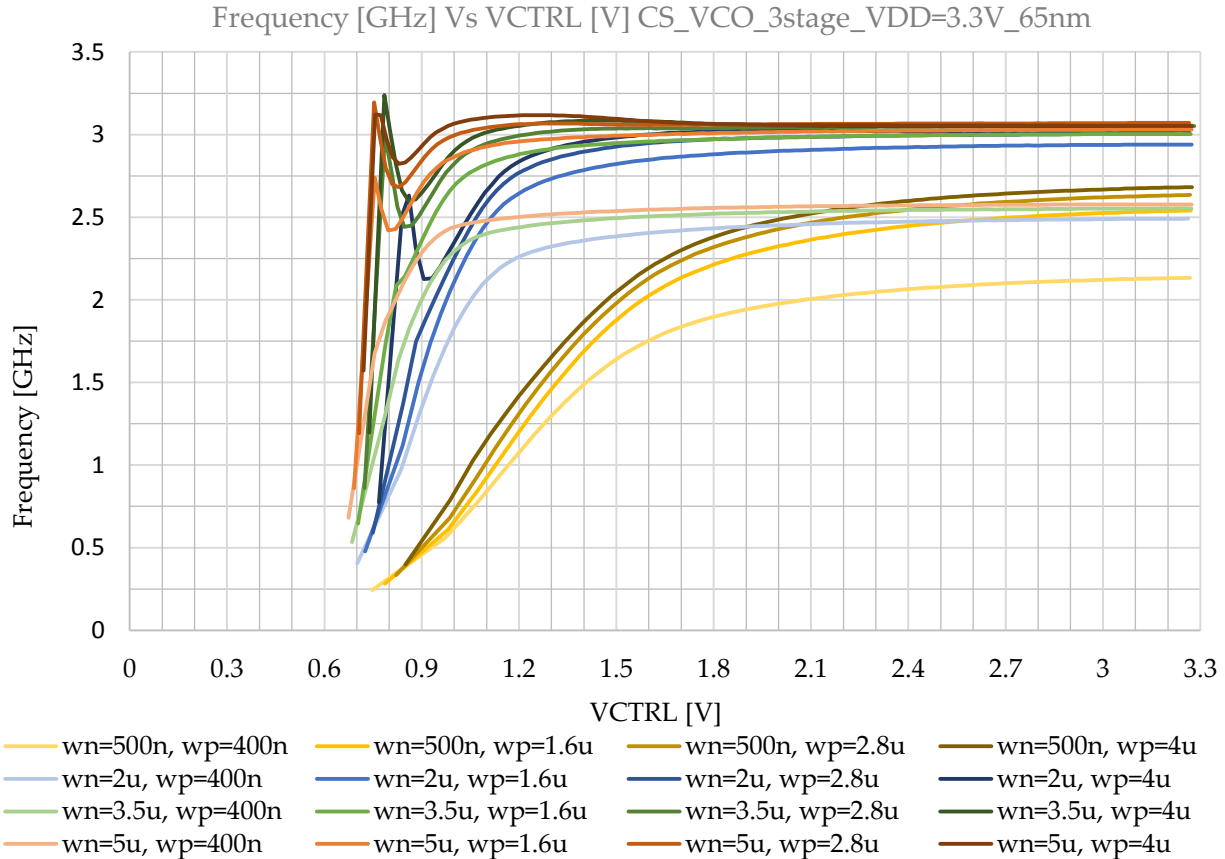
VCO-BASED ADC

• • •

Current Starved VCO with PMOS switch



Current Starved VCO Incrementing Width of Current Sourcing and Starving MOS



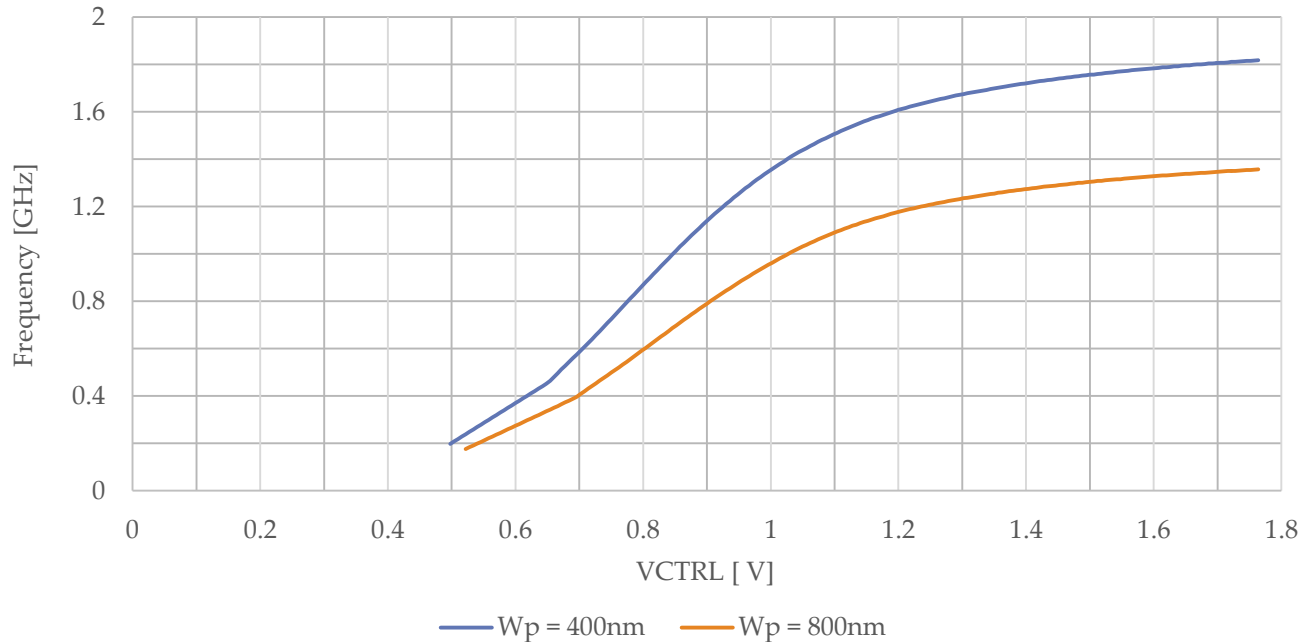
VCO-BASED ADC

• • •

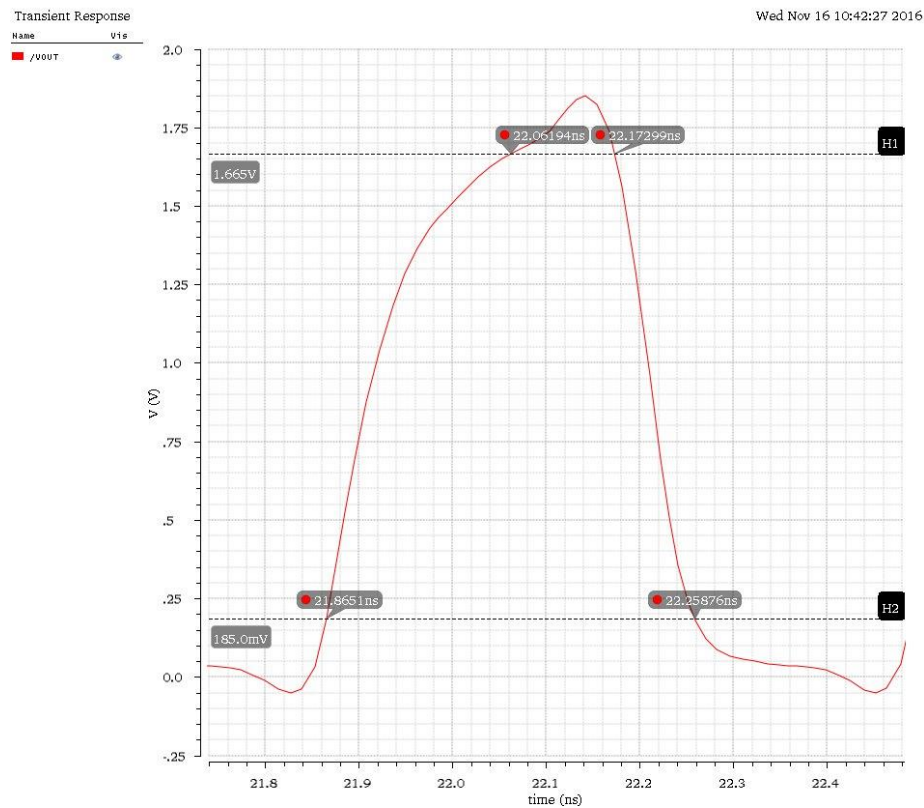
Current Starved VCO with Inverter PMOS Width twice that of Inverter NMOS Width

V-f Characteristics plot of 5 stage, VDD = 1p8V

Blue is baseline , Orange is when $W_{\text{pmos}}(\text{inverter}) = 2W_{\text{nmos}}(\text{inverter})$



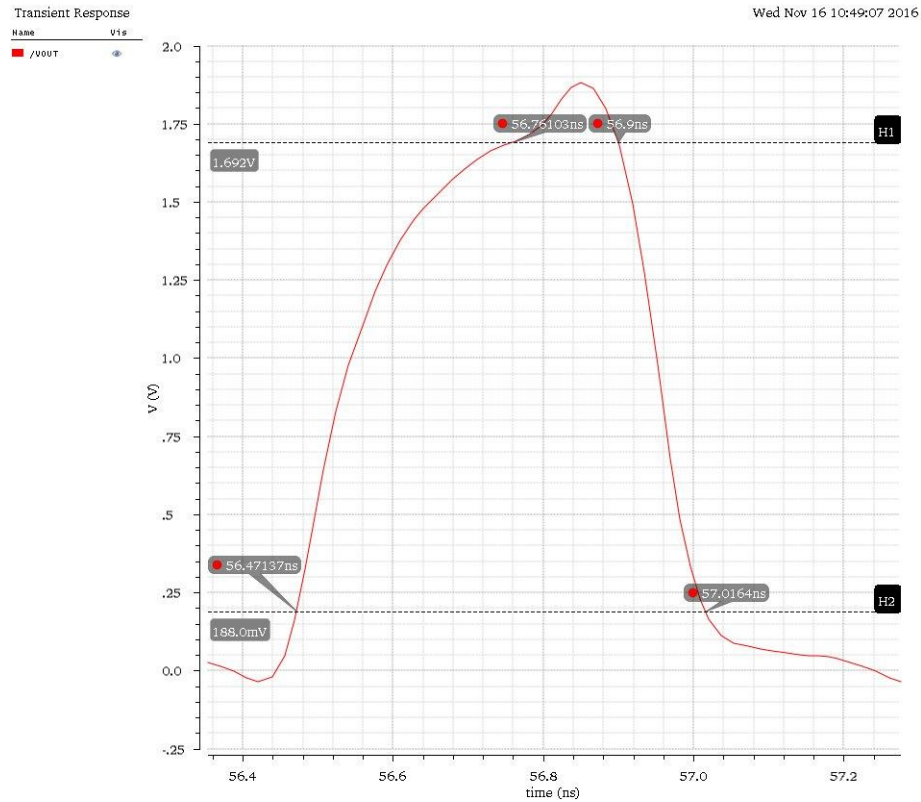
$W = 400\text{nm}$, rise time = 197ps, fall time = 85ps



VCO-BASED ADC

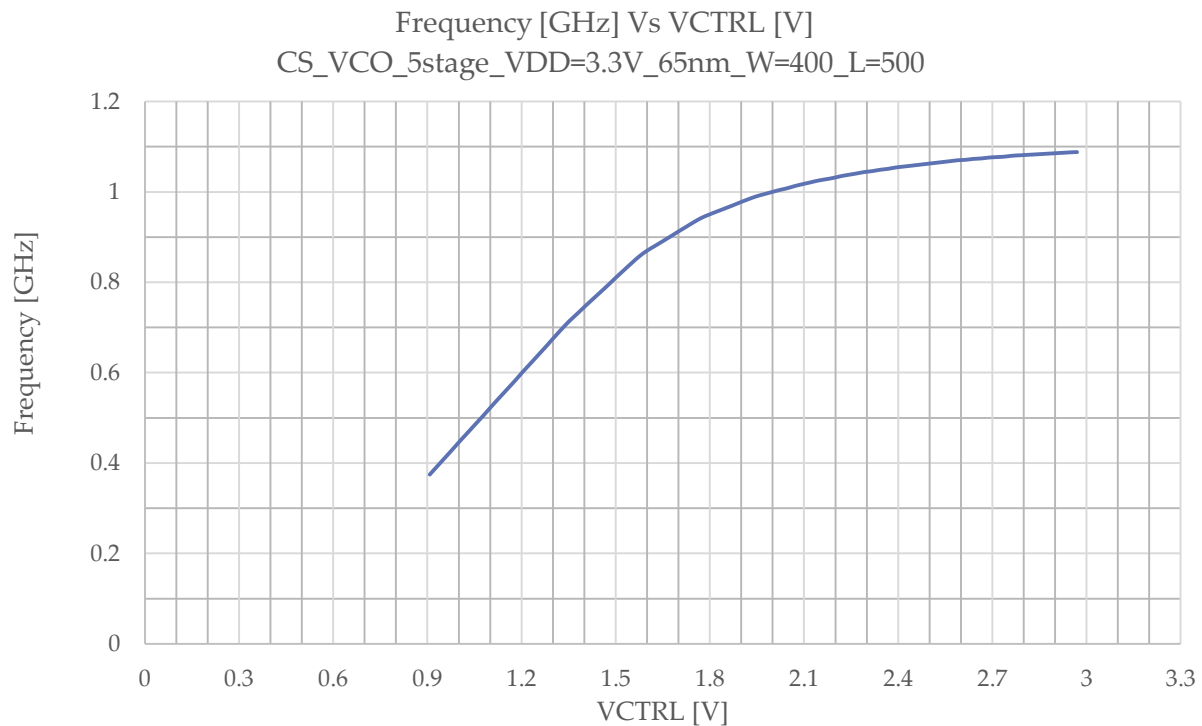
• • •

W = 800nm, rise time =289.7ps, fall time =116.4ps



VCO with 3.3V Supply: 5 Stage

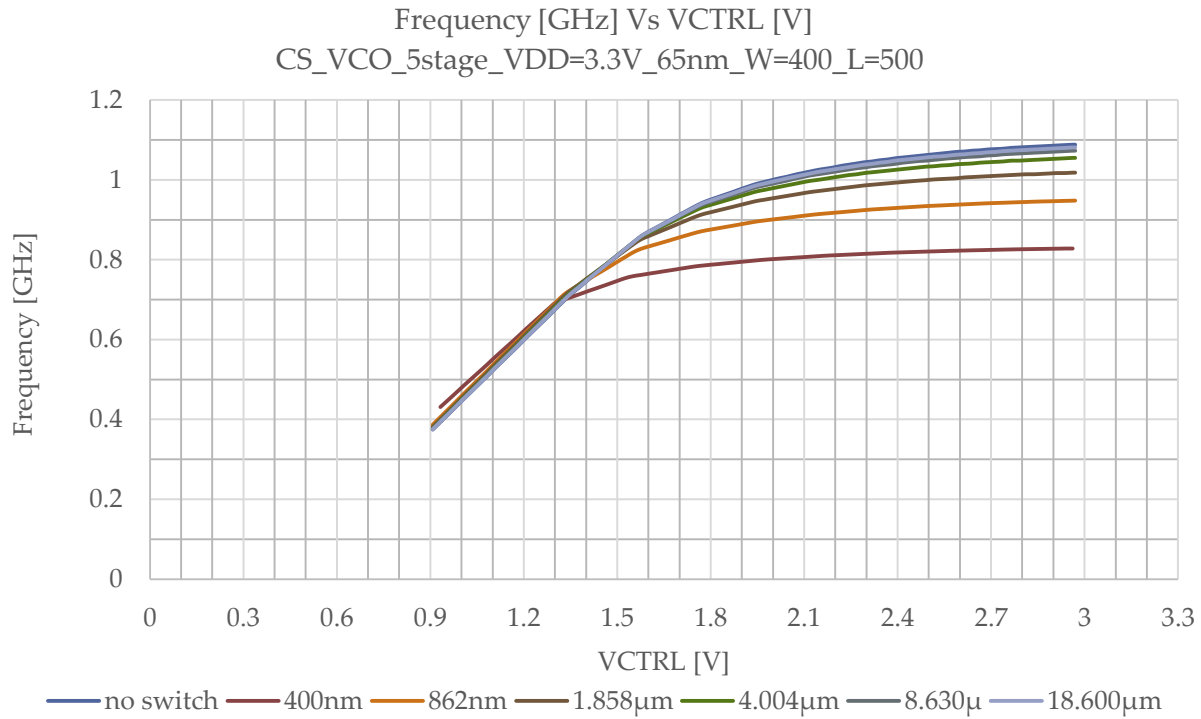
Current Starved VCO



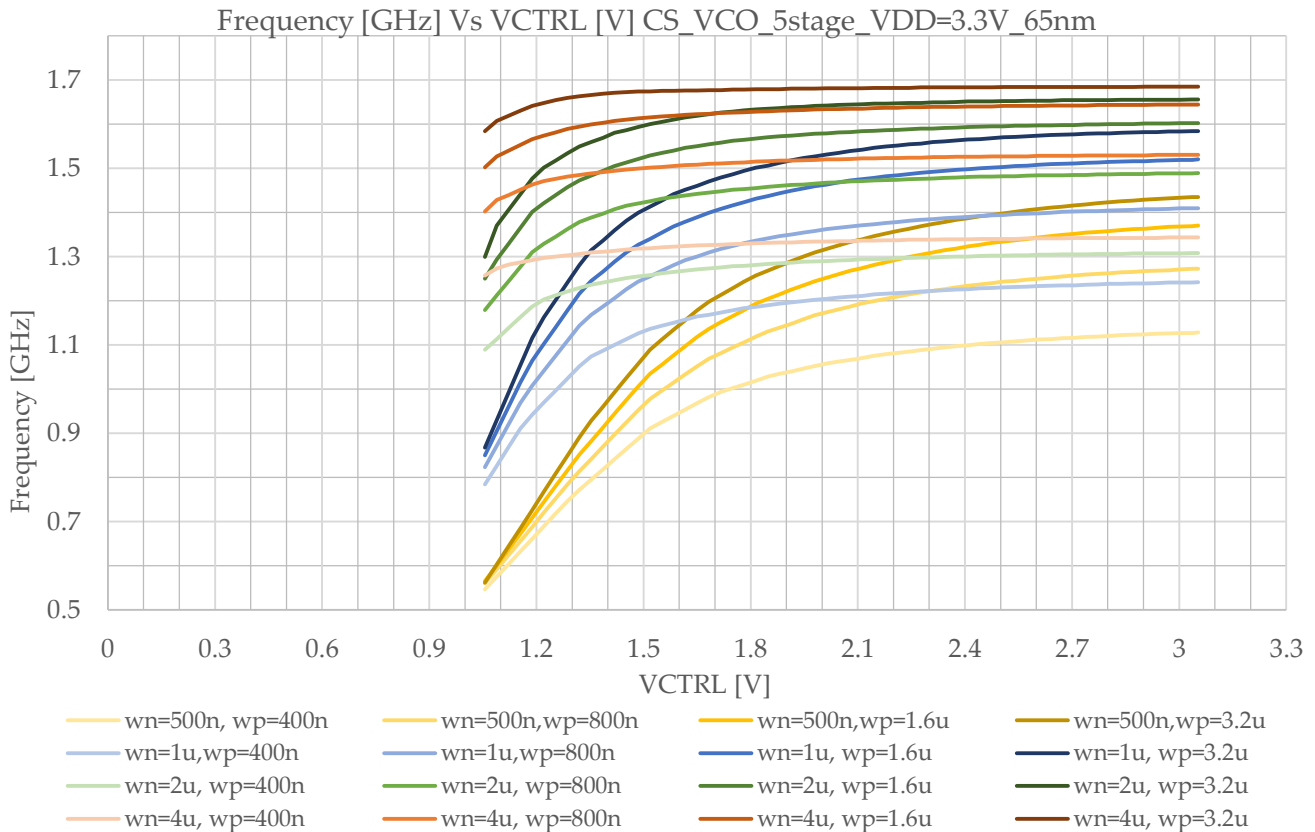
VCO-BASED ADC

• • •

Current Starved VCO with PMOS switch



Current Starved VCO Incrementing Width of Current Sourcing and Starving MOS

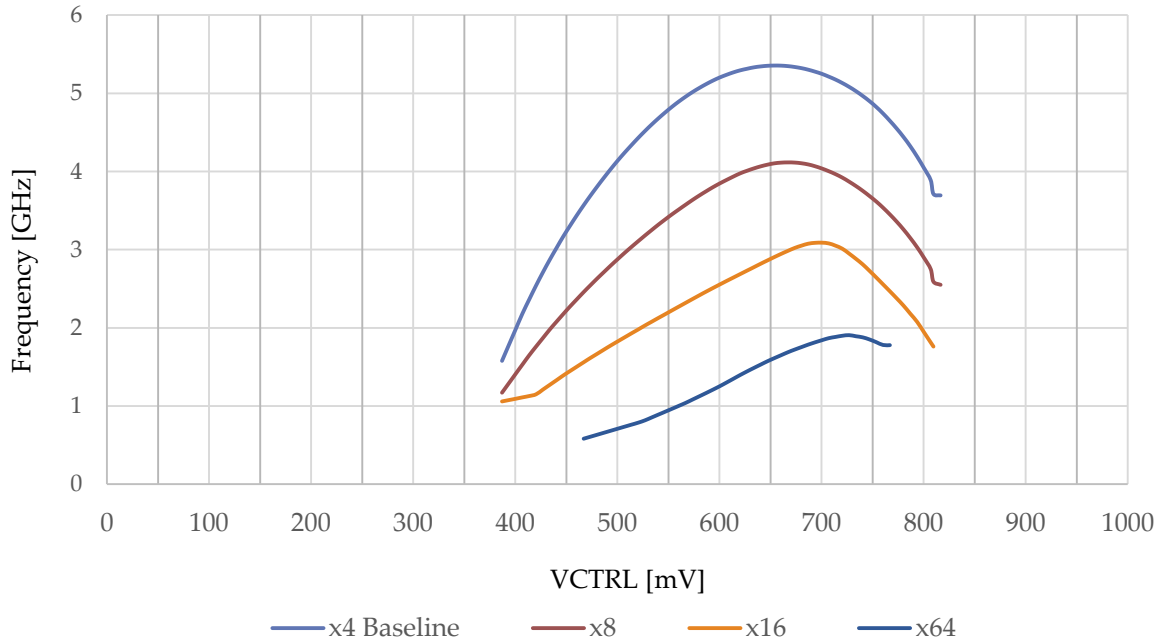


VCO-BASED ADC

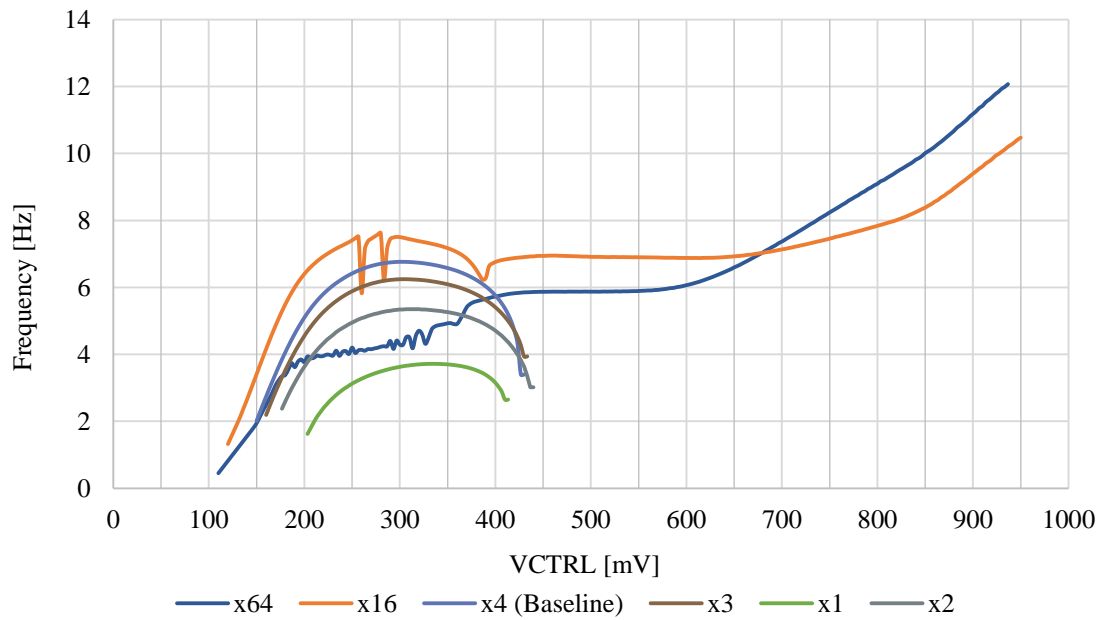
• • •

Pseudo-differential VCO

V-f characteristics of 5 stage, 1V CS PD VCO varying width of inverter rings with minimum width of inverter taps



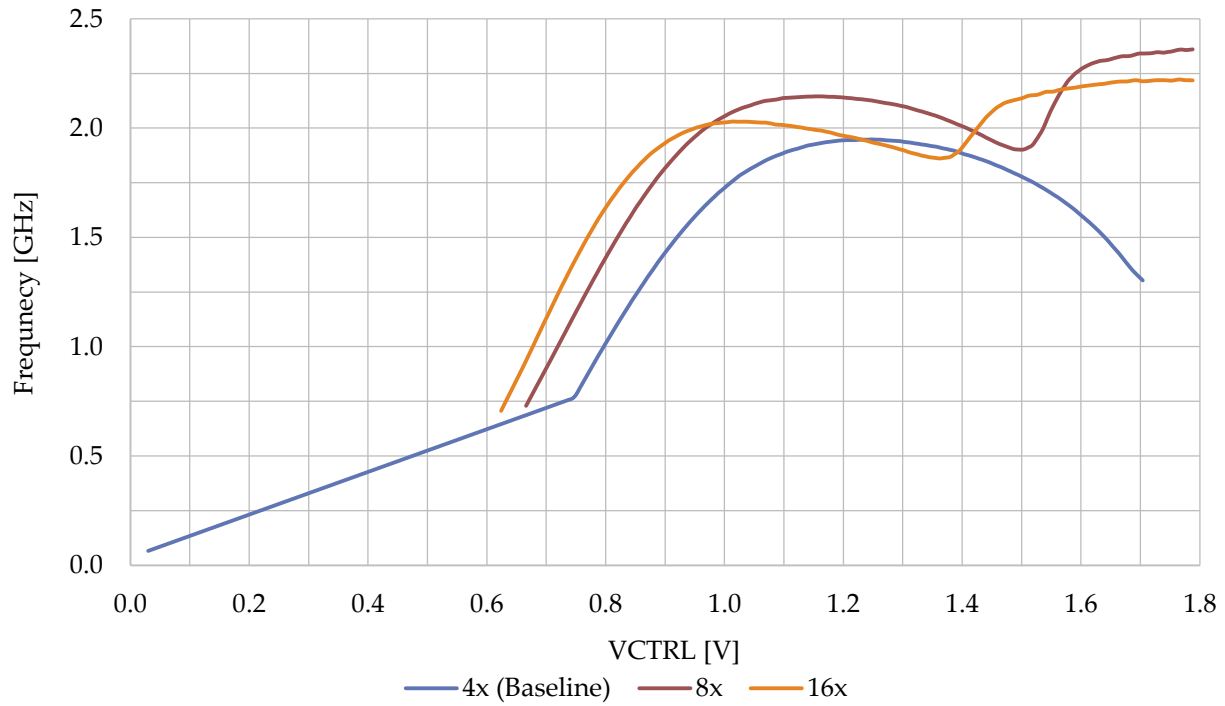
V-f characteristics of 5 stage, 1V CS PD VCO varying width of inverter rings with minimum width of inverter taps



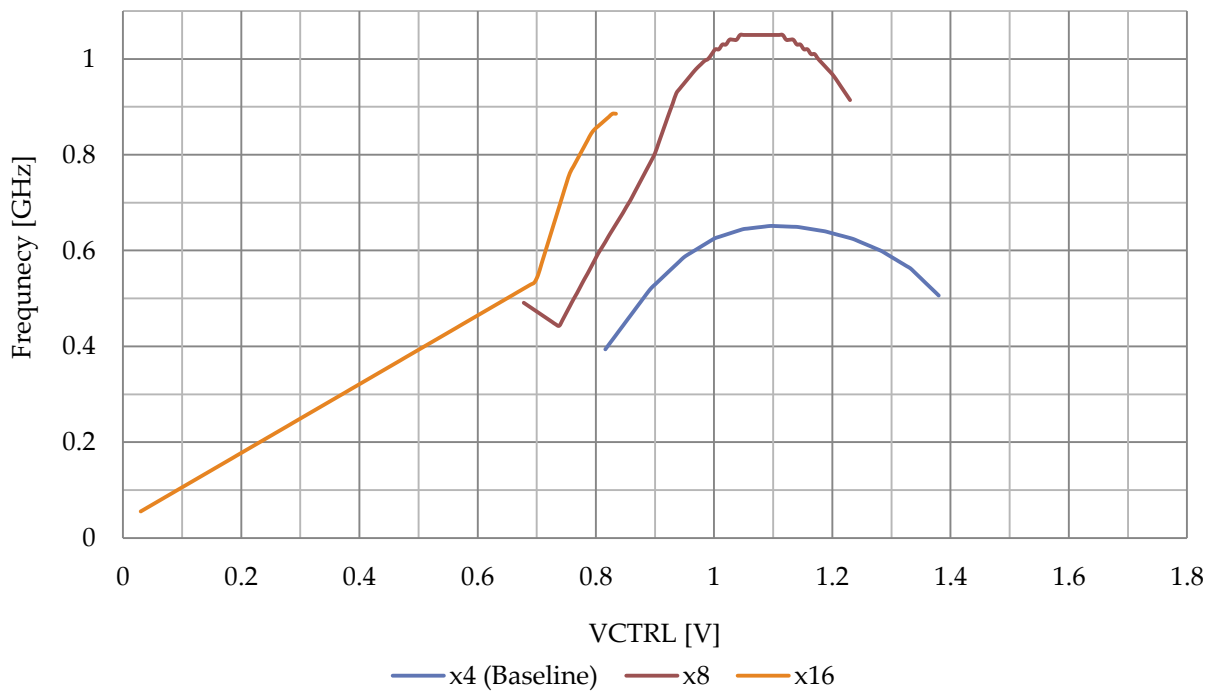
VCO-BASED ADC

• • •

V-f characteristics of 3 stage, 1.8V CS PD VCO varying width of inverter rings with minimum width of inverter taps



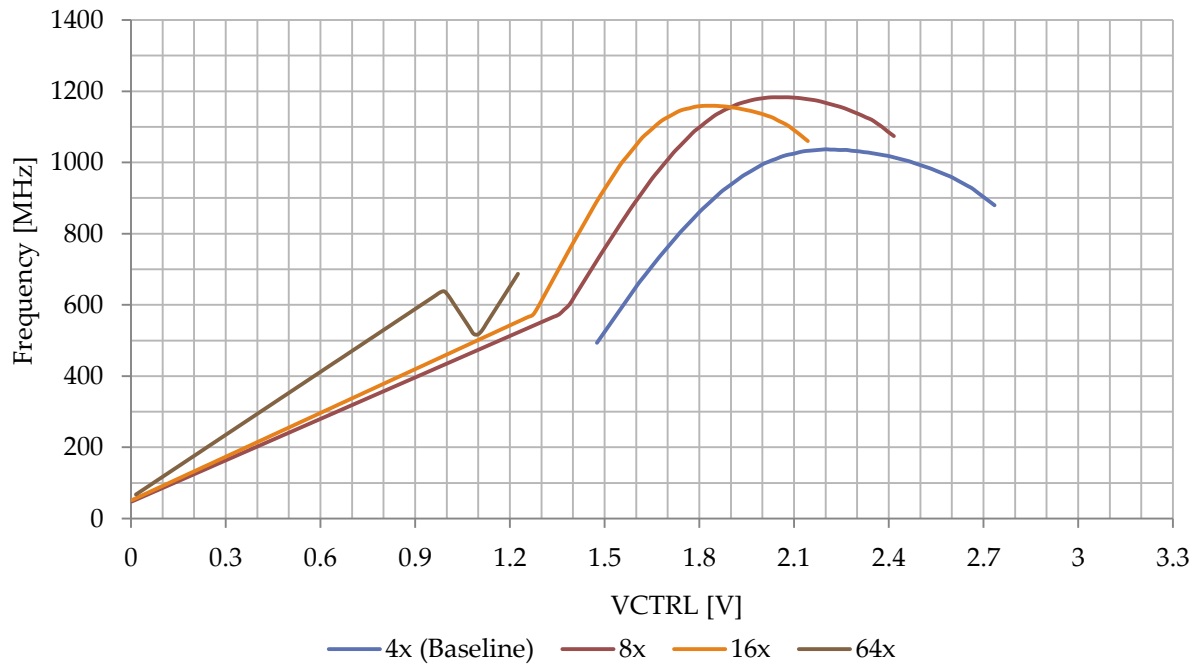
V-f characteristics of 5 stage, 1.8V CS PD VCO varying width of inverter rings with minimum width of inverter taps



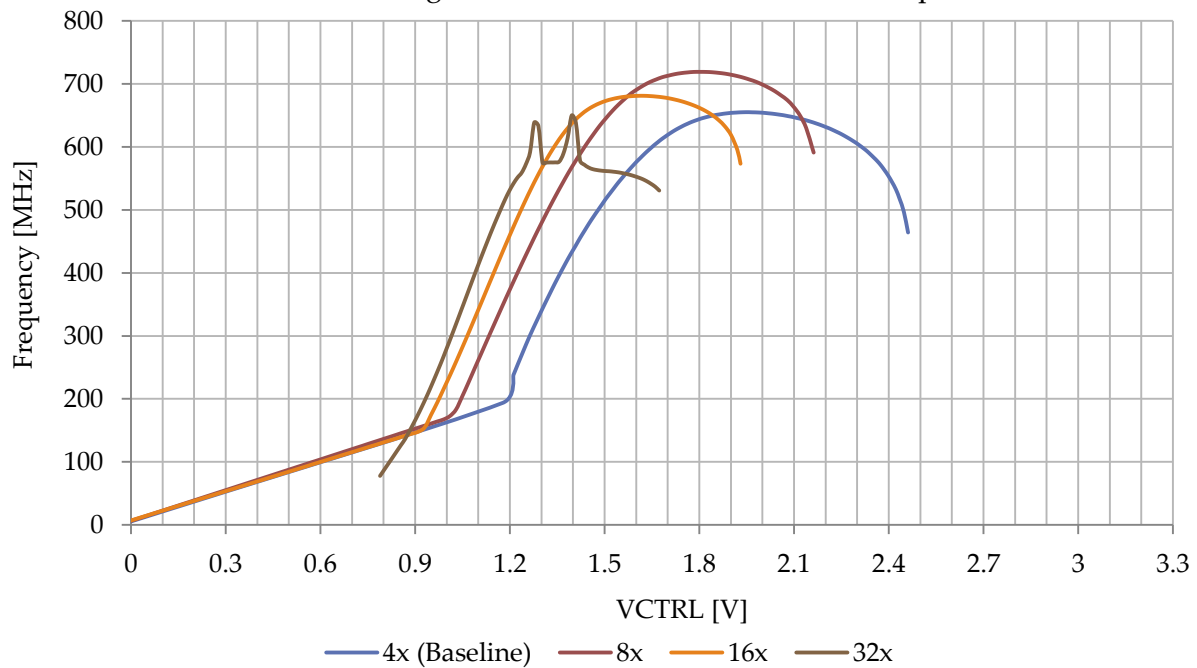
VCO-BASED ADC

• • •

V-f characteristics of 3 stage, 3.3V CS PD VCO varying width of inverter rings with minimum width of inverter taps



V-f characteristics of 5 stage, 3.3V CS PD VCO varying width of inverter rings with minimum width of inverter taps



• • •

Appendix C: Verilog Code

Top module: MQP_CNT

```

module MQP_CNT(

in,
sel,
input_rst,
output_bit

);

input [3:0] in;
input [1:0] sel; // select for 4-to-1 MUX
input input_rst;
output [11:0] output_bit;          // 12 bit output, MAX = 2^12 -1

// 2^12 * 22 = 1.375 * 2^17

wire mux_output;

MUX MUX(.i(in),.a(sel[0]),.b(sel[1]),.q(mux_output));

RPCNT RPCNT(.ripplecnt_clk(mux_output), .ripplecnt_rst(input_rst),
.ripplecnt_bit(output_bit));

endmodule

```

MUX module

```

module MUX(

i,
a,
b,
q

);

input [3:0] i;
input a;
input b;
output reg q;

wire [1:0] ba;

```


• • •

```
assign ba = {b,a};
```

```
always @(ba,i)
    case (ba)
        2'b00: q <= i[0];
        2'b01: q <= i[1];
        2'b10: q <= i[2];
        2'b11: q <= i[3];
    endcase
```

```
endmodule
```

Ripple counter module

```
module RPCNT(

    ripplecnt_clk,
    ripplecnt_rst,
    ripplecnt_bit

);

    input ripplecnt_clk, ripplecnt_rst;
    output [11:0] ripplecnt_bit;          // 12 bit output, MAX = 2^12 -1

    // 2^12 * 22 = 1.375 * 2^17

    wire qb0, qb1, qb2, qb3, qb4, qb5, qb6, qb7, qb8, qb9, qb10, qb11;

    DFLIPFLOP ripplecnt_dff0(.clk(ripplecnt_clk), .reset(ripplecnt_rst),
        .d(qb0), .q(ripplecnt_bit[0]), .qb(qb0));

    DFLIPFLOP ripplecnt_dff1(.clk(qb0), .reset(ripplecnt_rst), .d(qb1),
        .q(ripplecnt_bit[1]), .qb(qb1));

    DFLIPFLOP ripplecnt_dff2(.clk(qb1), .reset(ripplecnt_rst), .d(qb2),
        .q(ripplecnt_bit[2]), .qb(qb2));

    DFLIPFLOP ripplecnt_dff3(.clk(qb2), .reset(ripplecnt_rst), .d(qb3),
        .q(ripplecnt_bit[3]), .qb(qb3));

    DFLIPFLOP ripplecnt_dff4(.clk(qb3), .reset(ripplecnt_rst), .d(qb4),
        .q(ripplecnt_bit[4]), .qb(qb4));

    DFLIPFLOP ripplecnt_dff5(.clk(qb4), .reset(ripplecnt_rst), .d(qb5),
        .q(ripplecnt_bit[5]), .qb(qb5));

    DFLIPFLOP ripplecnt_dff6(.clk(qb5), .reset(ripplecnt_rst), .d(qb6),
        .q(ripplecnt_bit[6]), .qb(qb6));
```

• • •

```

DFLIPFLOP ripplecnt_dff7(.clk(qb6), .reset(ripplecnt_rst), .d(qb7),
.q(ripplecnt_bit[7]), .qb(qb7));

DFLIPFLOP ripplecnt_dff8(.clk(qb7), .reset(ripplecnt_rst), .d(qb8),
.q(ripplecnt_bit[8]), .qb(qb8));

DFLIPFLOP ripplecnt_dff9(.clk(qb8), .reset(ripplecnt_rst), .d(qb9),
.q(ripplecnt_bit[9]), .qb(qb9));

DFLIPFLOP ripplecnt_dff10(.clk(qb9), .reset(ripplecnt_rst), .d(qb10),
.q(ripplecnt_bit[10]), .qb(qb10));

DFLIPFLOP ripplecnt_dff11(.clk(qb10), .reset(ripplecnt_rst), .d(qb11),
.q(ripplecnt_bit[11]), .qb(qb11));

endmodule

```

D Flip-Flop module

```

module DFLIPFLOP(

clk,
reset,
d,
q,
qb

);

input clk, reset, d;
output reg q;
output qb;

/*
always@(posedge clk or posedge reset) begin          // caution: asynchronous
reset
    if(reset) begin
        q <= 1'b0;
        qb <= 1'b1;
    end

    else begin
        q <= d;
        qb <= ~d;
    end
end
*/

```

VCO-BASED ADC

• • •

```
always@(posedge clk or negedge reset) begin          // caution: asynchronous
reset
    if(~reset) begin
        q <= 1'b0;
    end

    else begin
        q <= d;
    end
end

assign qb = ~q;

endmodule
```

• • •

Appendix D: Performance Analysis on Digital circuitry

```

set design MQP_CNT

current_design $design

set a 1;
set b 0;
create_clock -period 0.22 -waveform {0 0.15} [get_ports "in[1]"]

set_clock_uncertainty -setup 0.01 [get_clocks "in[1]"]
set_clock_uncertainty -hold 0.01 [get_clocks "in[1]"]
set_clock_transition 0.05 [get_clocks "in[1]"]

set_fanout_load 8 [all_outputs]

set_dont_touch_network [get_ports "in[1]"]

set auto_wire_load_selection true

set_fix_multiple_port_nets -all -buffer_constants
compile_ultra -retime -timing

compile_ultra -incremental -area

change_names -rules verilog -hierarchy
write -f verilog -hierarchy -output "netlists/${design}.v"

write_sdc "netlists/${design}.sdc"
redirect "reports/${design}.area1" { report_area }
redirect "reports/${design}.tim1" { report_timing }
redirect "reports/${design}.power1" { report_power }

set design MQP_CNT

current_design $design

set a 0;
set b 1;
create_clock -period 0.22 -waveform {0 0.15} [get_ports "in[2]"]

set_clock_uncertainty -setup 0.01 [get_clocks "in[2]"]
set_clock_uncertainty -hold 0.01 [get_clocks "in[2]"]
set_clock_transition 0.05 [get_clocks "in[2]"]

set_fanout_load 8 [all_outputs]

set_dont_touch_network [get_ports "in[2]"]

```

VCO-BASED ADC

• • •

```
set auto_wire_load_selection true

set_fix_multiple_port_nets -all -buffer_constants
compile_ultra -retime -timing

compile_ultra -incremental -area

change_names -rules verilog -hierarchy
write -f verilog -hierarchy -output "netlists/${design}.v"

write_sdc "netlists/${design}.sdc"
redirect "reports/${design}.area2" { report_area }
redirect "reports/${design}.tim2" { report_timing }
redirect "reports/${design}.power2" { report_power }

set design MQP_CNT

current_design $design

set a 1;
set b 1;
create_clock -period 0.3 -waveform {0 0.15} [get_ports "in[3]"]

set_clock_uncertainty -setup 0.01 [get_clocks "in[3]"]
set_clock_uncertainty -hold 0.01 [get_clocks "in[3]"]
set_clock_transition 0.05 [get_clocks "in[3]"]

set_fanout_load 8 [all_outputs]

set_dont_touch_network [get_ports "in[3]"]

set auto_wire_load_selection true

set_fix_multiple_port_nets -all -buffer_constants
compile_ultra -retime -timing

compile_ultra -incremental -area

change_names -rules verilog -hierarchy
write -f verilog -hierarchy -output "netlists/${design}.v"

write_sdc "netlists/${design}.sdc"
redirect "reports/${design}.area3" { report_area }
redirect "reports/${design}.tim3" { report_timing }
redirect "reports/${design}.power3" { report_power }
```

VCO-BASED ADC

• • •

Report of total cell area of digital circuitry

Report : area

Design : MUXCNT

Version: I-2013.12-SP5-5

Date : Fri Feb 24 14:11:22 2017

Information: Updating design information... (UID-85)

Library(s) Used:

tcbn65gplustc_ccs (File:
/share/apps/cadence/tcbn65gplus/200a/Front_End/timing_power_noise/CCS/tcbn65
gplus_200a/tcbn65gplustc_ccs.db)

Number of ports:	19
Number of nets:	32
Number of cells:	13
Number of combinational cells:	1
Number of sequential cells:	12
Number of macros/black boxes:	0
Number of buf/inv:	0
Number of references:	2

Combinational area:	8.280000
Buf/Inv area:	0.000000
Noncombinational area:	103.680004
Macro/Black Box area:	0.000000
Net Interconnect area:	undefined (Wire load has zero net area)

Total cell area:	111.960004
Total area:	undefined

1

VCO-BASED ADC

• • •

Report of Power dissipation by digital circuitry

Loading db file

'/share/apps/cadence/tc65gplus/200a/Front_End/timing_power_noise/CCS/tc65gplus_200a/tc65gplustc_ccs.db'

Information: Propagating switching activity (low effort zero delay simulation). (PWR-6)

Warning: Design has unannotated primary inputs. (PWR-414)

Warning: Design has unannotated sequential cell outputs. (PWR-415)

Report : power

-analysis_effort low

Design : MUXCNT

Version: I-2013.12-SP5-5

Date : Fri Feb 24 14:11:28 2017

Library(s) Used:

tcbn65gplustc_ccs (File:
/share/apps/cadence/tc65gplus/200a/Front_End/timing_power_noise/CCS/tc65gplus_200a/tc65gplustc_ccs.db)

Operating Conditions: NCCOM Library: tcbn65gplustc_ccs

Wire Load Model Mode: segmented

Design	Wire Load Model	Library
MUXCNT	ZeroWireload	tcbn65gplustc_ccs

Global Operating Voltage = 1

Power-specific unit information :

Voltage Units = 1V

Capacitance Units = 1.000000pf

Time Units = 1ns

Dynamic Power Units = 1mW (derived from V,C,T units)

Leakage Power Units = 1nW

Cell Internal Power = 76.7843 uW (95%)

Net Switching Power = 3.8075 uW (5%)

Total Dynamic Power = 80.5918 uW (100%)

Cell Leakage Power = 475.4619 nW

VCO-BASED ADC

• • •

Total	Internal	Switching	Leakage
Power Group	Power	Power	Power
Power (%) Attrs			

io_pad	0.0000	0.0000	0.0000
0.0000 (0.00%)			
memory	0.0000	0.0000	0.0000
0.0000 (0.00%)			
black_box	0.0000	0.0000	0.0000
0.0000 (0.00%)			
clock_network	2.8995e-02	1.7617e-03	31.4905
3.0788e-02 (37.98%)			
register	2.9593e-02	1.1507e-03	41.4156
3.0785e-02 (37.97%)			
sequential	1.8196e-02	8.9503e-04	402.5558
1.9494e-02 (24.05%)			
combinational	0.0000	0.0000	0.0000
0.0000 (0.00%)			

Total	7.6784e-02 mW	3.8075e-03 mW	475.4619 nW
8.1067e-02 mW			
1			

Report of timing slack by digital circuitry (using input1)

Report : timing

-path full

-delay max

-max_paths 1

Design : MUXCNT

Version: I-2013.12-SP5-5

Date : Fri Feb 24 14:11:22 2017

Operating Conditions: NCCOM Library: tcbn65gplustc_ccs

Wire Load Model Mode: segmented

Startpoint: RPCNT_ripplecnt_dff0_q_reg

(rising edge-triggered flip-flop clocked by in[0])

Endpoint: RPCNT_ripplecnt_dff0_q_reg

(rising edge-triggered flip-flop clocked by in[0])

Path Group: in[0]

Path Type: max

VCO-BASED ADC

• • •

Des/Clust/Port	Wire Load Model	Library
MUXCNT	ZeroWireload	tcbn65gplustc_ccs
Point	Incr	Path
clock in[0] (rise edge)	0.00	0.00
clock network delay (ideal)	0.00	0.00
RPCNT_ripplecnt_dff0_q_reg/CP (DFCND1)	0.00	0.00 r
RPCNT_ripplecnt_dff0_q_reg/QN (DFCND1)	0.13	0.13 r
RPCNT_ripplecnt_dff0_q_reg/D (DFCND1)	0.00	0.13 r
data arrival time		0.13
clock in[0] (rise edge)	0.17	0.17
clock network delay (ideal)	0.00	0.17
clock uncertainty	-0.01	0.16
RPCNT_ripplecnt_dff0_q_reg/CP (DFCND1)	0.00	0.16 r
library setup time	-0.01	0.15
data required time		0.15
data required time		0.15
data arrival time		-0.13
slack (MET)		0.03

Appendix E: Matlab Code

vcofft1v11stage.m (Figure 57)

```

clear all;
filename = 'lv11stagefft.xlsx'; % extract data from excel file
time = xlsread(filename, 'A2:A501');
vcooutput = xlsread(filename, 'B2:B501');
stage11 = xlsread(filename, 'C2:C501');

%Perform Fast Fourier transform with default settings
Vcooutputfreq = fft(vcooutput);
Stage11freq = fft(stage11);

n=2e-10; %sampling speed

fs=1/n;
%Sampling at 2e-10
%so sampling at 5GHz
f=[];
for a= -249:1:250 %Finding unnormalized frequency

    if a==0
        f=[f 0]; % Can't divide 0
    else
        f=[f fs*a/500];
    end
end

figure;
subplot(2,2,2);
plot(time,vcooutput);
xlabel('time(s)');
ylabel('Voltage (V)');
title('/Vout(at end of 5stage buffer)');

subplot(2,2,1);
plot(time,stage11);
xlabel('time(s)');
ylabel('Voltage (V)');
title('/stage11');

subplot(2,2,3);
plot(f(250:500),abs(Stage11freq(250:500)));%only plot positive frequencies
xlabel('n, (Hz)');
ylabel('Magnitude of X[K]');
title('/stage11 frequency spectrum');

subplot(2,2,4);
plot(f(250:500),abs(Vcooutputfreq(250:500)));%only plot positive frequencies
xlabel('n, (Hz)');
ylabel('Magnitude of X[K]');
title('/Vout frequency spectrum');

```

• • •

vcofft3p3v11stage.m (Figure 58)

```

clear all;
filename = '3p3v11stagefft.xlsx'; % extract data from excel file
time = xlsread(filename, 'A2:A501');
vcooutput = xlsread(filename, 'B2:B501');
stage11 = xlsread(filename, 'C2:C501');

%Perform Fast Fourier transform with default settings
Vcooutputfreq = fft(vcooutput);
Stage11freq = fft(stage11);

n=2e-10; %sampling speed

fs=1/n;
%Sampling at 2e-10
%so sampling at 5GHz
f=[];
for a= -249:1:250 %Finding unnormalized frequency

    if a==0
        f=[f 0]; % Can't divide 0
    else
        f=[f fs*a/500];
    end
end

figure;
subplot(2,2,2);
plot(time,vcooutput);
xlabel('time(s)');
ylabel('Voltage (V)');
title('/Vout(at end of 5stage buffer)');

subplot(2,2,1);
plot(time,stage11);
xlabel('time(s)');
ylabel('Voltage (V)');
title('/stage11');

subplot(2,2,3);
plot(f(250:500),abs(Stage11freq(250:500)));%only plot positive frequencies
xlabel('n, (Hz)');
ylabel('Magnitude of X[K]');
title('/stage11 frequency spectrum');

subplot(2,2,4);
plot(f(250:500),abs(Vcooutputfreq(250:500)));%only plot positive frequencies
xlabel('n, (Hz)');
ylabel('Magnitude of X[K]');
title('/Vout frequency spectrum');

```

• • •

vcofft1v11stageNoAlias.m (Figure 59)

```

clear all;
filename = 'lv11stagefast.xlsx';% extract data from excel file
time = xlsread(filename,'A2:A1001');
vcooutput = xlsread(filename,'C2:C1001');
stage11 = xlsread(filename,'B2:B1001');

%Perform Fast Fourier transform with default settings
Vcooutputfreq = fft(vcooutput);
Stage11freq = fft(stage11);

n=1e-10;%sampling speed
fs=1/n;
%Sampling at 1e-10
%so sampling at 10GHz
f=[];
for a= -499:1:500 %Finding unnormalized frequency

    if a==0
        f=[f 0];% Can't divide 0
    else
        f=[f fs*a/1000];
    end
end

figure;
subplot(2,2,2);
plot(time,vcooutput);
xlabel('time(s)');
ylabel('Voltage (V)');
title('/Vout(at end of 5stage buffer)');

subplot(2,2,1);
plot(time,stage11);
xlabel('time(s)');
ylabel('Voltage (V)');
title('/stage11');

subplot(2,2,3);
plot(f(501:1000),abs(Stage11freq(501:1000)));%only plot positive frequencies
xlabel('n, (Hz)');
ylabel('Magnitude of X[K]');
title('/stage11 frequency spectrum');

subplot(2,2,4);
plot(f(501:1000),abs(Vcooutputfreq(501:1000)));
%only plot positive frequencies
xlabel('n, (Hz)');
ylabel('Magnitude of X[K]');
title('/Vout frequency spectrum');

```