# A 5GS/s 4-BIT TIME-BASED ADC

By

Assem Shoukry Hussein

Hady Mohamed said

Mahmoud Abd el-hamed Ahmed

Mahmoud Fawzy Zakrya

Mazen Mohamed Taha

Sherif Mohamed Hosny

Under the Supervision of

Dr.Hassan Mostafa

Dr.Mohamed Amin

A Graduation Project Report Submitted to
The Faculty of Engineering at Cairo University
In Partial Fulfillment of the Requirements for the
Degree of
Bachelor of Engineering
in
Electronics and Electrical Communications Engineering

Faculty of Engineering, Cairo University

Giza, Egypt

July 2014

# Table of Contents

## List of Tables

# List of Figures

# List of Symbols and Abbreviations

Please insert here any symbols and abbreviations that you commonly use in your report.

Example:

| | |
|---|---|
| ADC | Analog-to-digital converter |
| SNR | Signal-to-Noise Ratio |
| TADC | Time based analog to digital converter |
| DAC | Digital to analog converter |
| MSB | Most Significant Bit |
| LSB | Least Significant Bit |
| OSR | Oversampling ratio |
| DNL | Differential non-linearity |
| INL | Integral Non-Linearity |
| SNR | Signal-to-noise ratio |
| SINAD | Signal-to-noise and distortion ratio |
| ENOB | Effective number of bits |
| AFC | Analog voltage to frequency converter |
| FDC | Frequency to digital converter |
| VTC | Voltage to time converter |
| TDC | Time to digital converter |
| VDL | Vernier delay line |

## Acknowledgments

We would like to thank our supervisors Dr.Hassan Mostafa and Dr.Mohamed Amin and express our gratitude for their great support, encouragement, and guidance throughout this project. Finally, we would like to thank our families and our friends for being source of warmth, stability and encouragement not only during our study in the faculty but in all life aspects.

# Abstract

ADCs are very important electronic modules. Any communication receiver or even any device that needs an analog interface must contain an ADC. Also it's an important part of most embedded systems applications that communicate with analog interfaces.

The target of our designed ADC is to be integrated inside an SDR (Software Defined Radio).The time-based ADC is considered an essential block in designing software radio receivers because it exhibits higher speed and lower power consumption compared to the conventional ADCs, especially, at scaled down CMOS technologies. While CMOS technology continues to scale down very fast, conventional voltages to digital converters are facing challenging obstacles concerning accuracy, resolution and power. In particular, due to supply voltage reduction, the voltage domain is becoming noisier. In addition, the relatively high threshold voltage makes the available headroom very small for any sophisticated analog architectures. These challenges make the conventional ADCs incapable of providing the high speed required to adopt them at the UWB receivers front-end.

On the positive side of scaling, with decreased rising and falling times, the switching characteristics of MOS transistors offer excellent timing accuracy at high frequencies. Consequently, the time based ADCs (TADC) appears as the best solution to achieve the front-end ADCs high speed requirements.

The T-ADC is mainly composed of two blocks which are the VTC (Voltage-to-Time Converter) and the TDC (Time-to-Digital Converter). The T-ADC briefly converts the analog signal to time delay through the VTC. Then, the time-represented signal is converted to a digital one through the TDC.

Our project is to design a 5GS/s 4-bit nyquist rate time based ADC, which is the fastest achieved speed with this number of bits. We have no restriction on power consumption and any reasonable dynamic range is acceptable. After that, we will implement the layout of the ADC and it should be fabricated afterwards under the 65nm technology.

# Chapter 1: Introduction

An analog-to-digital converter (ADC) is a device that converts a continuous physical (analog) quantity representing real-world phenomenon e.g., light, sound, temperature or pressure to a digital number that represents the quantity's amplitude.

The input of an analog-to-digital converter (ADC) consists of a signal that can take theoretically infinite number of values (continuous amplitude). In contrast, the output of the ADC has defined number of levels or states (discrete amplitude). The number of states is almost always a power of two -- that is, 2, 4, 8, 16, etc.

The main goal of ADC is to digitize the analog signals, which means to record and store the analog signals in numbers so that the microprocessor will be able to read, understand and manipulate the data. Digital signals are more efficiently than analog signals, as digital impulses, which are well-defined and orderly, are easier for electronic circuits to distinguish from noise because in digital technology the translation of information is into binary format (zero or one) where each bit is representative of two distinct amplitudes. Analog to digital converter is an important block used in mixed analog/digital systems. For example, Analog-to-digital converters are integral to current music reproduction technology, Digital signal processing system, some scientific instruments like Digital imaging systems and radar. Any communication must contain an ADC. Also it’s an important part of most embedded systems applications that communicate with analog interfaces.

## 1.1 ADCs Types

ADCs are divided into two main types according to the sampling frequency. The first type is called the Nyquist rate ADCs. Like flash ADC and successive-approximation ADC .In this type the sampling frequency is equal to twice the maximum frequency in the input signal bandwidth (Nyquist frequency) which is suitable for applications that require high input signal frequency. The second type is called oversampling ADCs. Oversampling conversion technique uses sampling frequency much larger than twice the input signal frequency. This technique is used for applications with low input signal frequency and high resolution requirements. ADCs are divided into two main types according to the way of conversion. The first type converts the analog input signal into digital code directly (conventional ADC). In the second type the conversion is done indirectly by converting the analog input signal into an intermediate representation then converting the intermediate representation into digital code. Time based analog to digital converter (TADC) is an example on this type. It is important to know that the above two classifications are not exclusive. ADC can use direct or indirect conversion with sampling frequency equals to Nyquist frequency or larger sampling frequency [1].

### 1.1.1 Nyquist Rate-Conventional ADCs

#### *1.1.1.1 Digital Ramp ADC*

Known as the stair step-ramp, or simply counter ADC converter. The basic idea is to connect the output of a counter to the input of a digital to analog converter (DAC), then compare the analog output of the DAC with the analog input signal If Comparator output is high then continue counting else stop counting. The circuit's need to count all the way from 0 at the beginning of each count cycle, which is suitable only for relatively slow sampling of the analog signal. A diagram of ramp ADC is shown in figure 1-1.

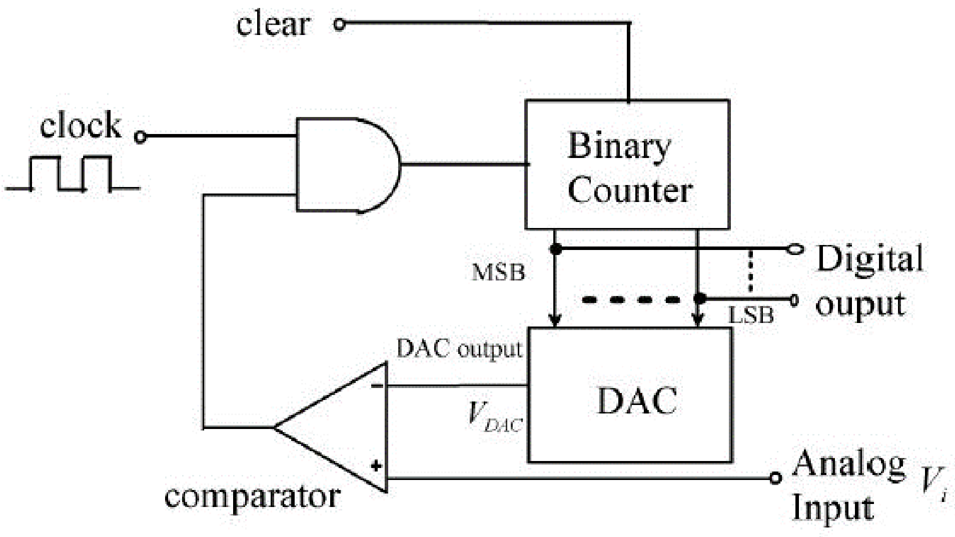


Figure 1-1 Ramp ADC

#### 1.1.1.2 Successive Approximation ADC

Instead of counting up in binary sequence, this Register counts by trying all values of bits starting with the most-significant bit and finishing at the least-significant bit. The most Significant Bit (MSB) set to one and remaining bits set to zero. Then the digital output is compared to the input signal. If Comparator output is high, MSB remains high. On the other hand, if comparator output is low, the MSB is set to zero. The conversion process continues for the next largest MSB in the same way [1]. Figure 1-2 shows the implantation of successive approximation ADC.

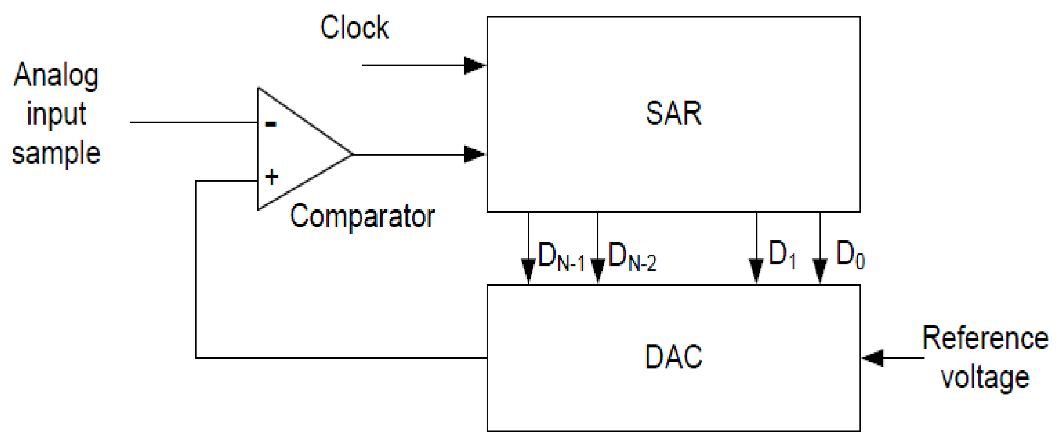


Figure 1-2 successive approximation ADC [1]

#### 1.1.1.3 Flash ADC

Also called the parallel ADC converter, it is the fastest type of ADCs. It consists of ladder of well-matched resistors connected to a reference voltage and $2^N - 1$ parallel comparators, each one comparing the input signal to a certain reference voltage. Each reference voltage is one LSB greater than the reference voltage immediately below it The comparator generates a logical '0' or '1' depending if the measured voltage is above or below the reference voltage producing a thermometer code. The thermometer code is then decoded to the appropriate digital output code using priority encoder [1]. The main disadvantage of this type of ADC is that it uses large number of comparators which results in large area and more power consumption. Figure 1-3 shows flash ADC diagram.

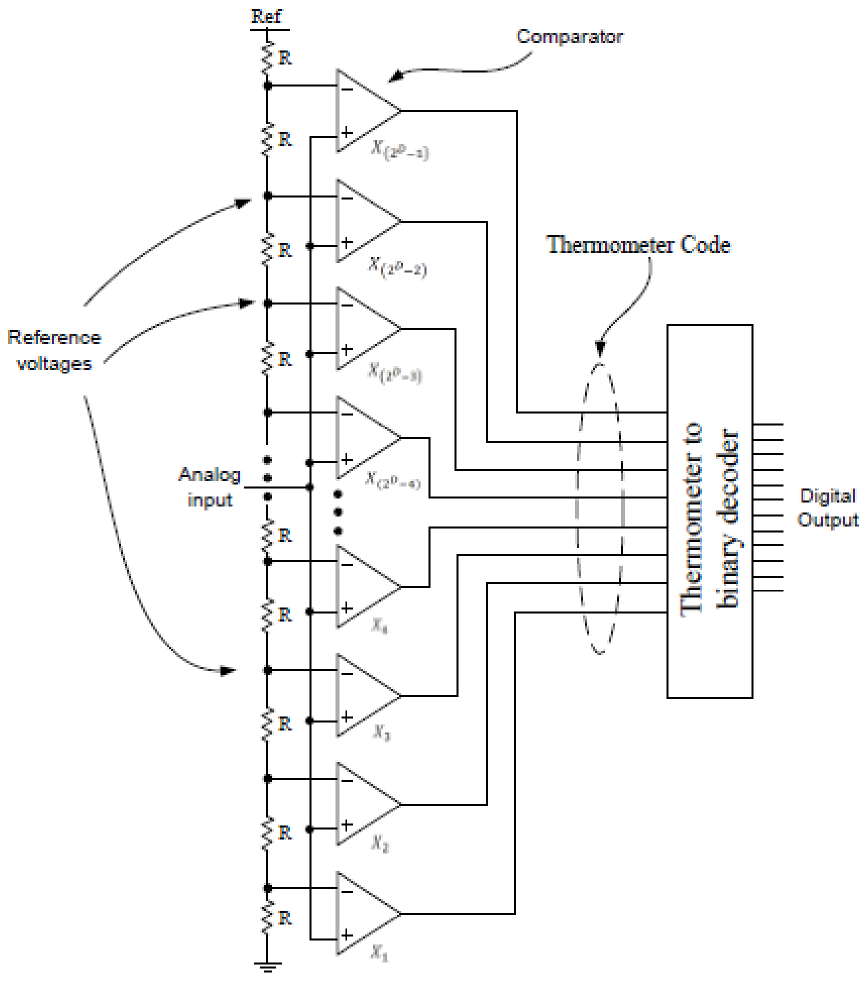


Figure 1-3 Flash ADC [1]

### 1.1.2 Over Sampling-Conventional ADCs

#### *1.1.2.1 Sigma-Delta Modulator*

The basic concept of the sigma-delta modulator is the use of high sampling rate and feedback for improving the effective resolution of the quantizer [1]. In a conventional ADC, an analog signal is integrated, or sampled, with a sampling frequency and subsequently quantized in a multi-level quantizer into a digital signal. This process introduces quantization error noise. The first step in a delta-sigma modulation is delta modulation. In delta modulation the change in the signal (its delta) is encoded, rather than the absolute value. The result is a stream of pulses, as opposed to a stream of numbers as is the case with PCM. In delta-sigma modulation, the accuracy of the modulation is improved by passing the digital output through a 1-bit DAC and adding (sigma) the resulting analog signal to the input signal, thereby reducing the error introduced by the delta-modulation. figure 1-4 shows sigma-delta Modulator diagram.

One of the most important sigma-delta modulator characteristics is the oversampling ratio (OSR) which defined as the ratio of the sampling frequency $fs$ to the Nyquist frequency.

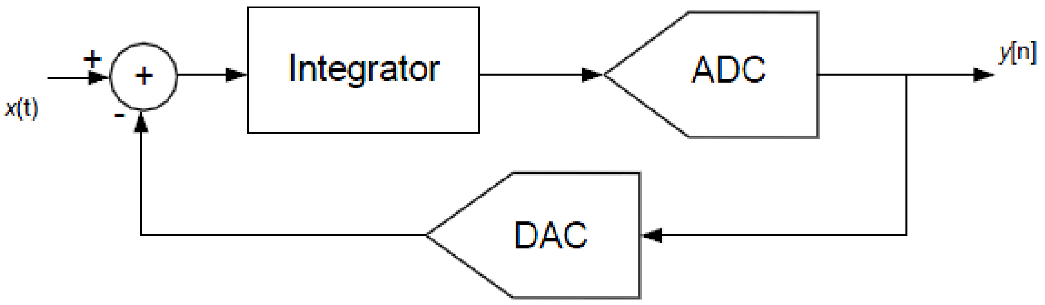


Figure 1-4 Sigma-delta modulator [1]

### 1.1.3 Indirect Conversion ADCs (Time-Based ADCs)

In this type the conversion is indirectly done by converting the analog input signal into an intermediate representation (time/frequency) then converting the intermediate representation into digital code [1].

The main advantage of time based ADC is to make use of CMOS technology scaling. While CMOS technology continues to scale down very fast, conventional voltages to digital ADCs are facing challenging obstacles concerning accuracy, resolution and power. In particular, due to supply voltage reduction, the voltage domain is becoming noisier. In addition, the relatively high threshold voltage makes the available headroom very small for any sophisticated analog architectures. On the positive side of scaling, with decreased rising and falling times, the switching characteristics of MOS transistors offer excellent timing accuracy at high frequencies [1].it will be discussed in the following chapters.

## 1.2 ADC Characteristics

There are many characteristics and parameters used to define the performance of ADCs which is useful to decide which type of ADCs is suitable for a certain application. These characteristics can be critical in some application. So, for these application we need to know the following ADC characteristics

### 1.2.1 Static Specifications

Static specifications for ADC define the performance of the ADC for a DC input signal, they represent errors due to non-idealities in circuit implementation of the ADC which do not depend on time

#### 1.2.1.1 Offset Error

Also referred as zero scale error, is defined as a constant difference between the ADC characteristic and the ideal one at zero input voltage [1] as shown in figure 1-5, this error is a result of the offset voltage of the operational amplifier and it can be easily overcome using calibration by measuring a reference point and subtracting that value from future samples.

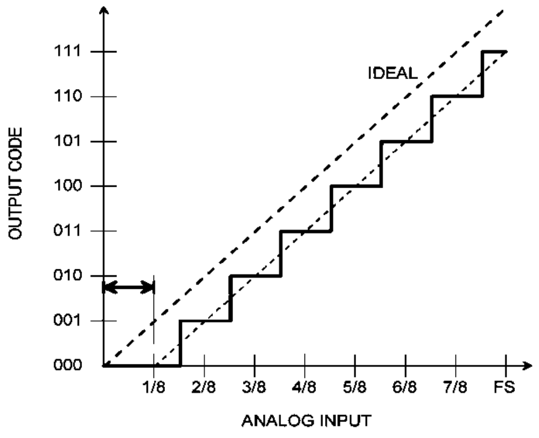


Figure 1-5 Offset error

#### 1.2.1.2 Gain Error

Defined as the difference of the slope of the actual output values and the ideal output values [1] as shown in figure 1-6.

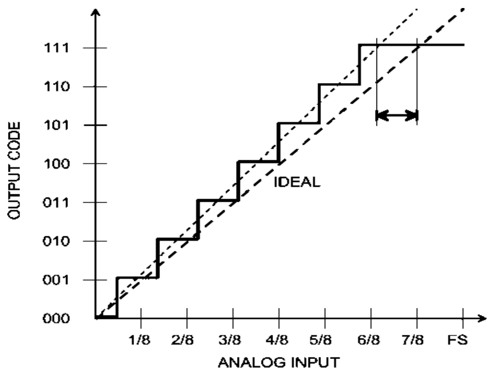


Figure 1-6 Gain error

Gain error can be removed be measuring a second reference point to determine the actual gain.

#### 1.2.1.3 *Differential And Integral Non-linearity*

For an ideal ADC the output is divided into $2^n$ uniform steps with the same width. The differential non-linearity (DNL) is defined as the deviation from the ideal step width and it is not possible to remove its effects with calibration. DNL errors accumulate to produce a total Integral Non-Linearity (INL). INL is defined as the deviation of an actual transfer function from the ideal one which is straight line [1]. Figure 1-7 illustrate these errors.

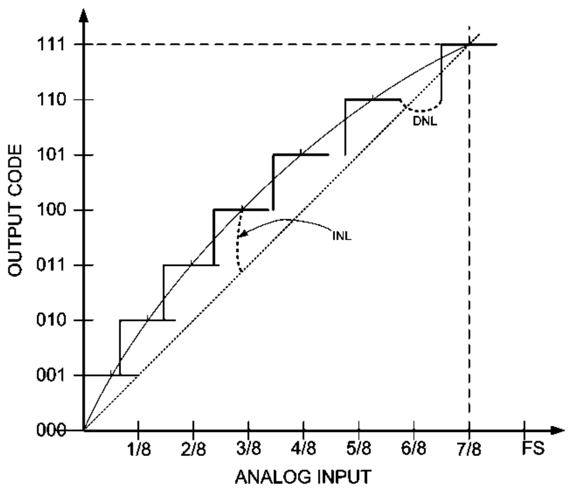


Figure 1-7 DNL&INL errors

#### 1.2.1.4 *Missing Codes*

ADC is said to have no missing codes when the input voltage is swept over its range (ramp input) and all possible output code combinations appear [1]. A DNL error of <±1LSB guarantees no missing codes. Figure 1-8 show missing codes error.

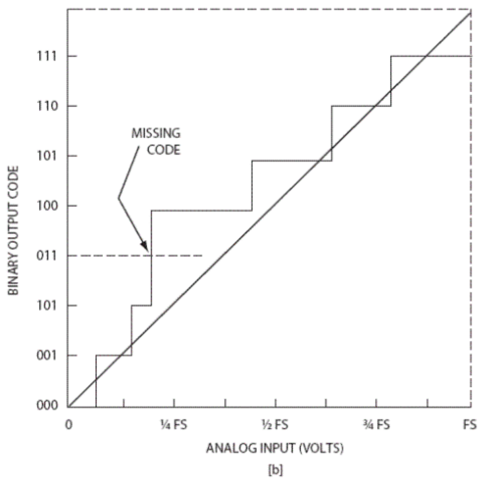


Figure 1-8 Missing code error

### 1.2.2 Dynamic Specifications

The specifications of the ADC that depend on time (dynamic)

#### *1.2.2.1 ADC Conversion Time and Sampling Rate*

The conversion time of the ADC is the time it takes for the analog signal to be converted to become a digital signal. Sampling rate is the reciprocal of the conversion time and it is defined as the speed at which the ADC can convert the analog input into digital bits continuously

#### *1.2.2.2 Signal to Noise Ratio*

Signal-to-noise ratio, often written S/N or SNR, is a measure of signal strength relative to background noise [2]. The ratio is usually measured in decibels (dB).

$$SNR = 10 * \log\left(\frac{Full\ scale\ input\ power}{Quantization\ noise\ power + circuit\ noise\ power}\right)\ dB \qquad (1\text{-}1)$$

For the quantization noise only (ignoring the circuit noise) and for n-bit integers with equal distance between quantization levels (uniform quantization) the SNR is given by

$$SNR = 6.02D + 1.76 \qquad (1\text{-}2)$$

#### *1.2.2.3 Signal to Noise and Distortion Ratio*

SINAD measurement is most widely used for measuring and specifying the sensitivity of a radio receiver. SINAD often defined as the ratio of the total received power to the noise-plus-distortion power [2]

$$SINAD = 10 * \log\left(\frac{Full\ scale\ input\ power}{noise\ power + \mathrm{distortion}\ power}\right)\ dB \qquad (1\text{-}3)$$

#### *1.2.2.4 Effective Number of Bits*

A measure of the signal-to-noise-and-distortion ratio used to compare actual analog to-digital converter (ADC) performance to an ideal ADC [3]. Since the ideal ADC SNR (due to the quantization noise only) is larger than the system SNDR, the actual number of bits will be less than the one got from equation (1-2). ENOB is the number of bits that if we substitute in the equation (1-2), the value of SNR will equal to the system SINAD value.

ENOB is obtained using the following formula

$$ENOB = \frac{SINAD - 1.76}{6.02} \qquad (1\text{-}4)$$

# Chapter 2: Time-Based ADCs

A time based analog to digital converter (TADC) is the type of ADCs that perform the analog to digital conversion in an indirect way by first converting analog signal to time (frequency) domain then converting the time representation into digital code.

This way of conversion helps us to eliminate the problems which faces us in traditional design ways of ADCs due to CMOS scaling and to make use of excellent timing characteristics of small MOS transistors.

There are several ADC architectures that we can use to design time-based ADC. In this chapter we will describe these architectures.

## 2.1 Integrating ADC

It is also called slope ADC. Integrating ADCs perform analog to digital conversion in time domain. Figure 2-1 shows diagram of a single slope ADC. The sampled input voltage (Vs) is stored on a capacitor. Then, Vs is discharged by a constant current source and this generates a ramp voltage at the capacitor output. A counter is start counting by the start of the ramp and stops when the ramp voltage is zero [4].

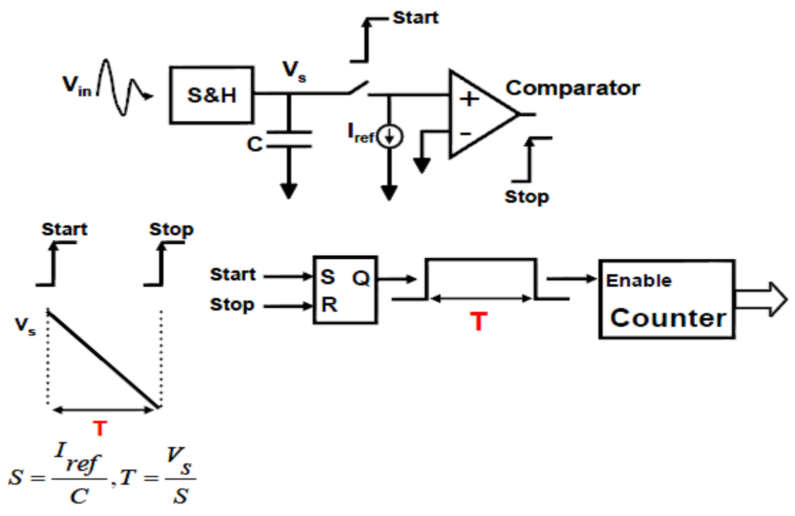


Figure 2-1Single slope ADC [4]

The counter digital output is proportional to the input signal. This architecture is simple and has low complexity but its sampling speed is low. In order to have a high speed some researchers replaced the counter with advanced time to digital conversion techniques.

## 2.2 AFC/FDC Based ADCs

Voltage-to-frequency-conversion based ADC consists of two blocks. The block is the analog voltage to frequency converter (AFC). It converts the input signal to frequency. Then the second block, frequency to digital converter (FDC) converts frequency to digital code [4]. Figure 2-2 shows the diagram of this ADC.

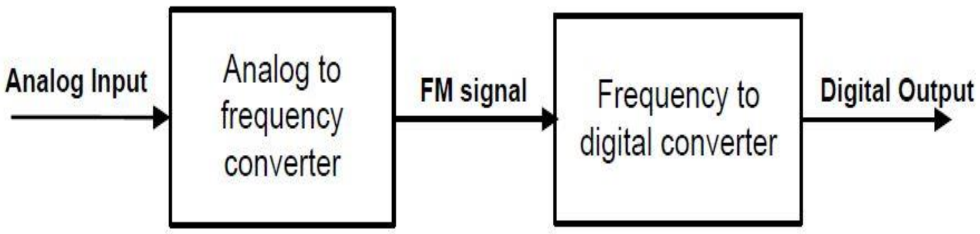


Figure 2-2 AFC/FDC based ADC [1]

### 2.2.1 Analog Voltage to Frequency Converter (AFC)

The voltage to frequency converter is based on using ring voltage controlled oscillator (ring VCO) [1].

As we can see from figure 2-3, a ring VCO generally consists of chain of delay cells connected in series, and a ring is formed by connecting the output of the last stage to the input of the first stage.

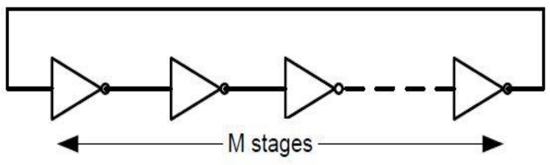


Figure 2-3 Ring VCO [1]

The delay cell can be implemented by using single-input-single-output and the number of delay cells ($M$) must be an odd number.

To understand how the ring works, assume that we have an odd number of cells (inverters) and the output of the first stage is one. As the output of the first stage is the input of the second stage. This results in making the output of the second stage to be zero, the same procedure continue in all the ring stages until we reach the last stage which output is one. The output of the last stage force the output of the first stage to switch to zero and this output start to propagate in the ring until the output of the first stage is forced again to be one and so on [1]. As we can see, it takes two loops through the ring to complete one period

We can write the relation between the period of oscillation $T$ and the delay of one delay cell $td$ as:

$$T = 2 \times M \times t_d \tag{2-1}$$

$$F = \frac{1}{T} \tag{2-2}$$

Where T is the period, td is the delay of one delay cell, M is the number of cells (odd number) and F is the frequency. Figure 2-4 illustrates the output of each cell.

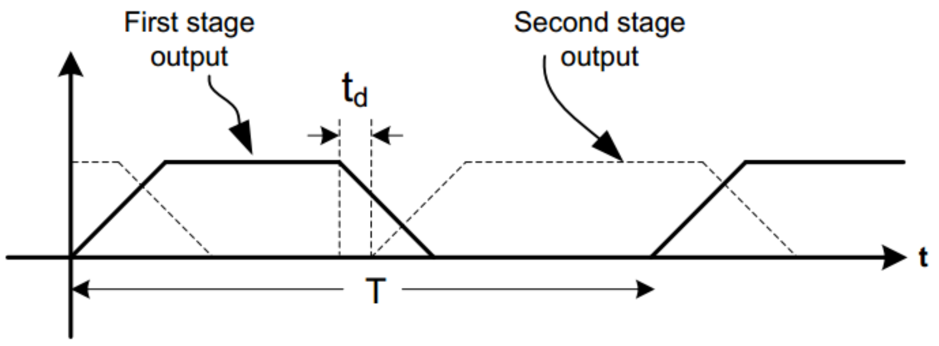


Figure 2-4 the output of the cell [1]

In addition to the condition on M and the above equations, we can control F by modulating $t_d$ using $V_{in}$ and choosing M an odd number.

Then, it is helpful to design a parameter which defines ratio of change in output frequency to change in input voltage, we can call it tuning parameter $k_{vco}$

$$K_{vco} = \frac{dW}{dVin} = 2\pi \frac{dF}{dVin} \tag{2-3}$$

### 2.2.2 Frequency to Digital Converter (FDC)

FDC is used to convert the frequency modulated signal into a digital code. FDC do its job by counting and quantizing the number of rising edges of the frequency modulated signal during the sampling interval [1] $Tref = 1/Fref$.

The simplest FDC may be implemented simply as a count and dump converter as shown in figure 2-5

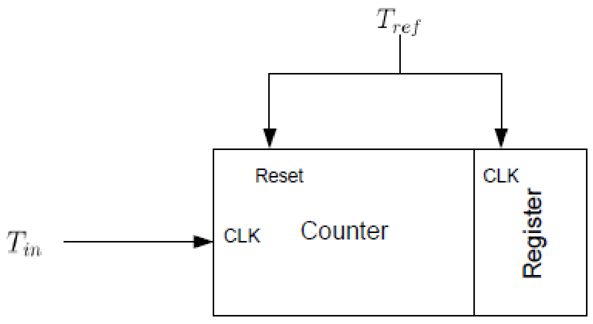


Figure 2-5 the simplest FDC [1]

Figure 2-6 shows the timing diagram of the input and output

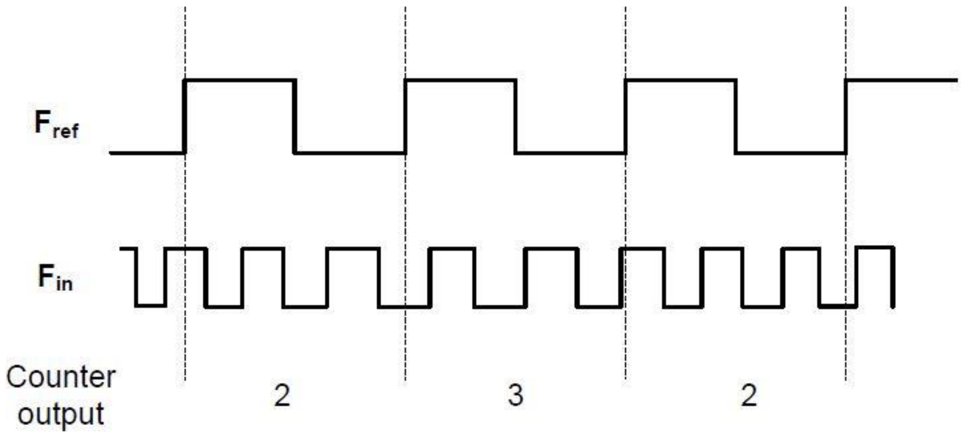


Figure 2-6 Timing diagram of simple FDC [1]

The drawback of this type is the counter resetting operation, which is a limiting factor for high speed operation [1]. We can overcome this drawback by using counter with no upper limit followed by digital differentiator as we do not need to reset the counter every $T_{ref}$.The non-limited counter can be realized as a modulo $2^n$ counter on the condition that the maximum number of received rising edges during $T_{ref}$ is smaller than the module of the counter to avoid signal aliasing. The differentiation is done by subtract two consecutive readings of the counter. Figure 2-7 shows this implementation.

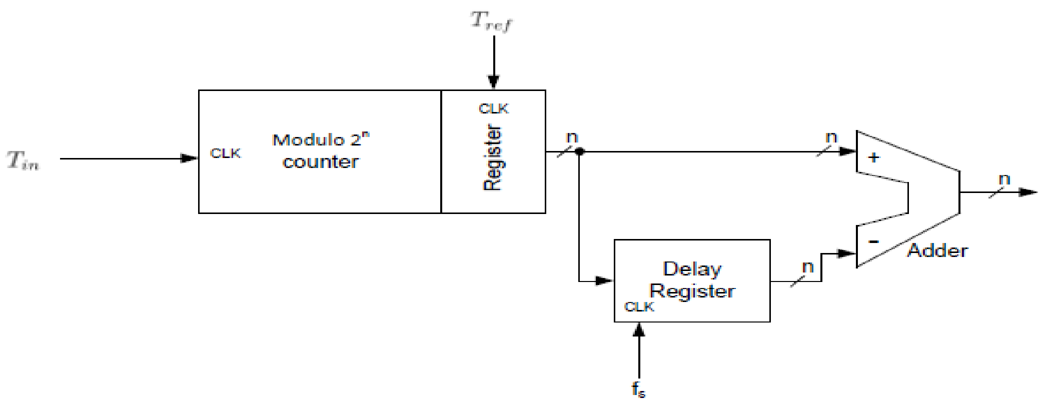


Figure 2-7 Non limited counter implementation of the FDC [1]

we can use modulo $2^1$ arithmetic (D-flip-flop), If the maximum received number of edges during $T_{ref}$ is smaller than two and the subtraction operation can be done using an XOR gate as shown in figure 2-8

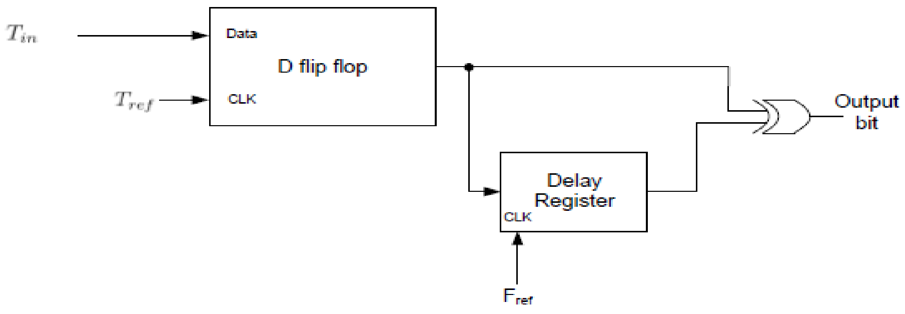


Figure 2-8 Modulo $2^1$diagram of FDC

Another kind of the FDC is based on phase version of the sigma-delta modulator [1] as shown in figure 2-9

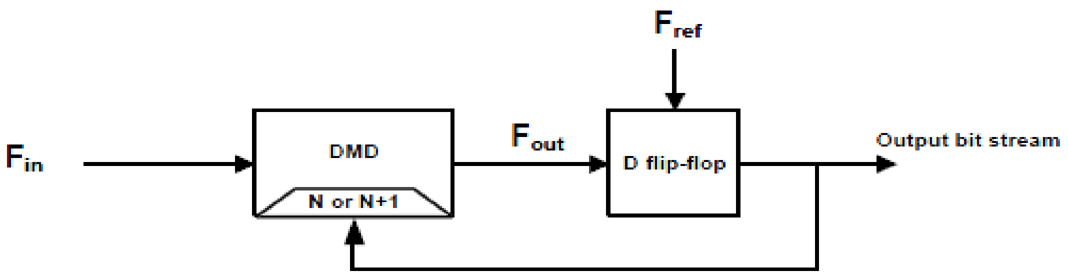


Figure 2-9 Sigma-delta modulator based FDC [1]

Figure 2-10 shows the timing diagram of the FDC signals. Simply we compare the phase of the signal $F_{out}$ with the reference signal $F_{ref}$ using D flip-flop. The output from the D-flip-flop can be seen to correspond to the sign of the phase difference, and thus the D-flip-flop acts as a phase quantizer, giving a one-bit approximation of the phase difference. This signal is feedback to control the divide ratio of the dual modulus frequency divider DMD. This forces the DMD to divide alternately by the higher and the lower modulus, causing the phase of the divided-down input to follow the phase of the reference signal, and causing the output bit stream from the comparator to possess a duty cycle that represent the input frequency and in turn the input voltage signal.

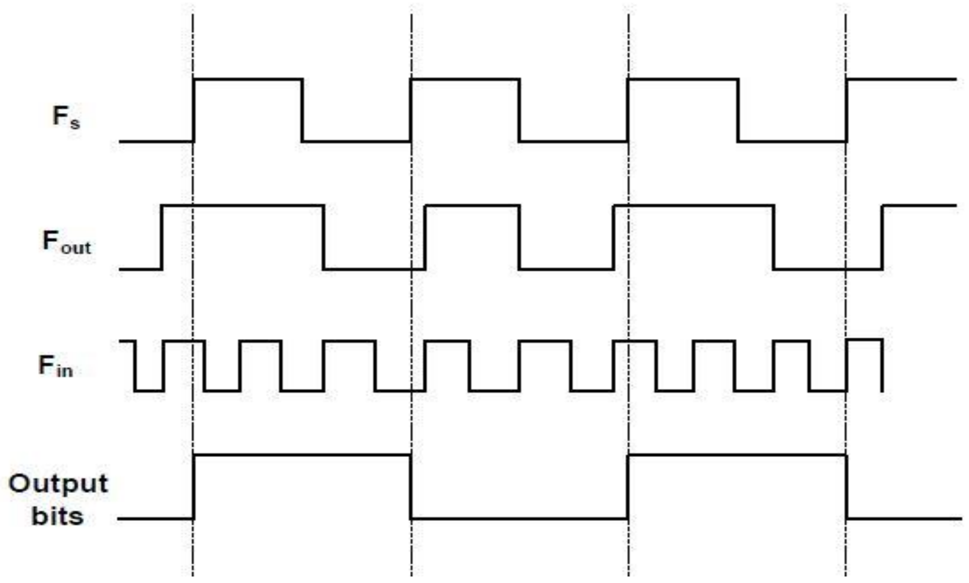


Figure 2-10 Timing diagram of FDC signals [1]

The discriminator can uniquely converts the input frequency into digital output if and only if we can maintain the input frequencies between $N \times fref$ and $(N + 1)\ fref$, where $N$ and $N + 1$ are the DMD divider ratios. If the input frequency doesn't fall in this region aliasing will occur.

## 2.3 VTC/TDC Based ADCs

Voltage-to-time-conversion based ADC consists of two blocks, voltage to time converter (VTC) and time to digital converter (TDC). We use the VTC block to convert the change in the voltage signal into a delay between two signals. Then we use TDC to convert this delay into digital code [1]. Figure 2-11 shows the block diagram of this ADC.

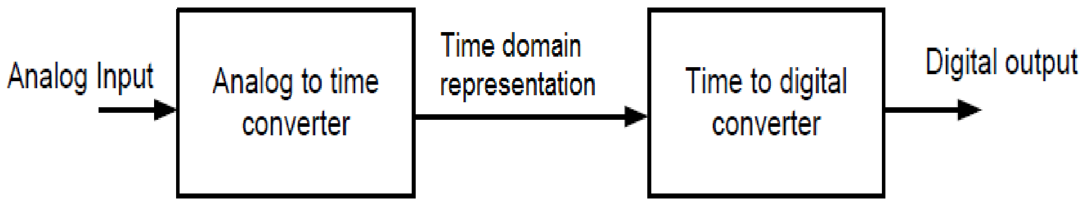


Figure 2-11VTC/TDC based ADC [1]

### 2.3.1 Voltage to Time Converter (VTC)

The VTC block, can be implemented using single-input-single-output inverter [1]. The conversion is done by controlling the current flows through the upper two transistors (M1, and M3) using the analog input signal thus we can control the charging or discharging rate of the capacitor connected to the output of the inverter is as shown in figure 2-12 So we control the time it takes $Vout$ to reaches a certain voltage.

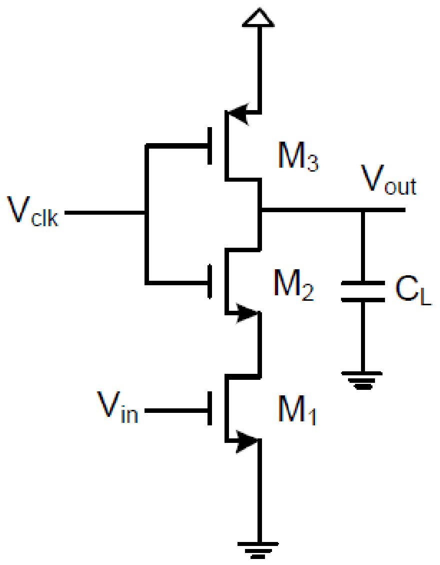


Figure 2-12 current starved inverter [1]

The main problem with this cell is the nonlinearity between the controlled voltage ($Vin$) and the delay value. This nonlinearity results in introducing distortion. Designers try to improve the linearity of this cell.

### 2.3.2 Time to Digital Converter (TDC)

The basic idea of the TDC is converting the amount of delay between the two signals that are named "The modulated signals" coming out of the VTC into digital thermometer code which is converted by the decoder into the desired number of number of bits of the ADC [1]. The two modulated signals coming from the VTC are named "Start" and "Stop" signals. There are several TDC architectures. We will illustrate them in the following pages.

#### *2.3.2.1 Counter Based Approach*

The first and the simplest way to implement the Time to digital converter is using a simple counter as shown in figure 2-13, the clock signal that is connected to the counter is the CP signal which is used to trigger the counter to start counting with the positive edge of this signal [5].

In order to measure the difference between the Start and Stop signals we have to count the number of rising edges of the clock and with the knowledge of the clock period which is equal to Tcp then we can conclude the difference ΔT easily.

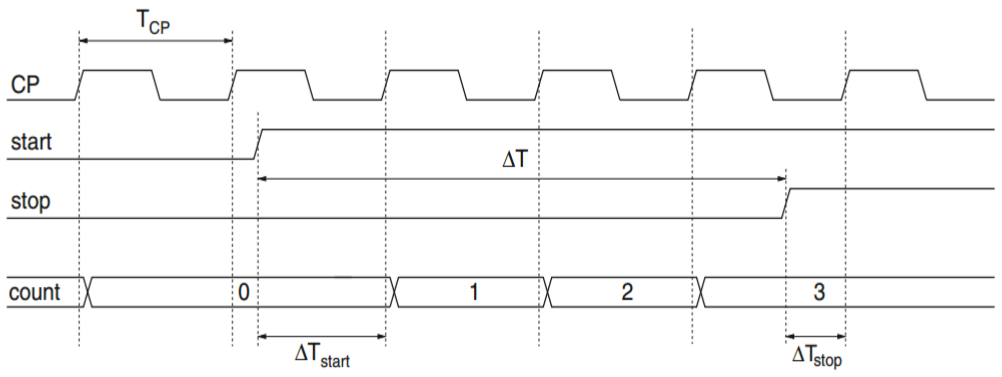


Figure 2-13 counter approach timing diagram [5]

The main problem of this technique is that the start and stop signal are not synchronized with the clock, which leads to the appearance of the measurement error ΔTstart which is equal to the time difference between the rising edge of the start signal and the succeeding rising edge of the clock, also similarly leads to the appearance of ΔTstop as shown in figure 3.1.

From figure 3.1 the interval ΔT could be calculated from these relations

$$\Delta T = N \times T_{cp} + \left(T_{cp} - \Delta T_{stop}\right) - (T_{cp} - \Delta T_{start})$$

$$= N \times T_{cp} - \Delta T_{stop} + \Delta T_{start}$$

$$= N \times T_{cp} + \varepsilon_T \tag{2-4}$$

$$\Delta T_{start} \in [0 \colon T_{cp}]$$

$$\Delta T_{stop} \in [0 \colon T_{cp}]$$

$$\varepsilon_T = \Delta T_{start} - \Delta T_{stop} \in [-T_{cp} \colon T_{cp}] \tag{2-5}$$

Where N is the number of counts, and εT is the quantization error; this error takes place due to the conversion of the voltage signal from the continuous voltage domain to the digital domain which contains certain number of levels defined by the number of bits of the ADC. Equ 3.3 illustrates that the quantization error is varying from –Tcp to Tcp and since our aim is to minimize the quantization error as much as we can in order to achieve better accuracy for the system, all we need is to minimize the period of the clock (Tcp) and so minimizing εT.

So it’s recommended to trigger the counter by a clock with very high frequency to achieve better performance. However, increasing the frequency will lead to higher power consumption. Also the CMOS oscillators are not available any more at high frequency, so we will need to use LC oscillators with high cost. The solution of this dilemma is to divide the clock into smaller time intervals; this in turn will lead to decrease the quantization error. Sooner or later the counter approach is not considered as an efficient solution for case of high frequencies.

#### 2.3.2.2 *Digital Delay Line Approach*

As mentioned earlier the solution of the problems of the previous technique, the clock signal must be divided into smaller time intervals which means that each counter clock cycle has to be sub-divided asynchronously by a time to digital converter [5] as shown in figure 2-14.

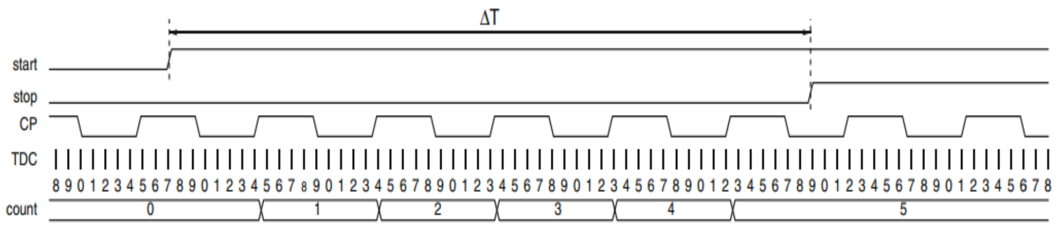


Figure 2-14 TDC Principle of subdivision timing diagram [5]

The interval ΔT calculations became more accurate due to dividing the interval into M subdivisions, leading to decreasing the quantization error and so better performance. In order to maintain the M subdivisions we can use a ring oscillator with M stages to generate M equally spaced versions of the clock signal. Achieving higher accuracy could be done by using a delay chain formed from delay cells in order to produce delayed versions of the original reference clock. In this case time the resolution will depend on the delay of the delay elements in the chain.

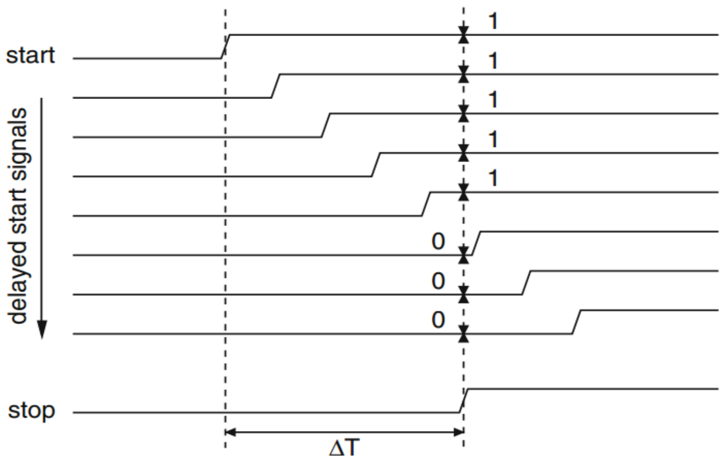


Figure 2-15 the delayed versions of the clock signal [5] figure 2-15 illustrates the basic idea of the TDC, assuming that the start signal is leading the stop by ΔT, the TDC will let the start signal pass through a chain containing delay elements with a known delay equals to τ that in turn will delay the start by N times. At the rising edge of the stop signal when N*τ = ΔT, there will be some delayed versions of start whose values are logic '1' and other delayed versions of start whose values are logic '0'. The number of delayed versions who have logic '1' value will aid in calculating the period ΔT by this relation

$$\Delta T = N \times \tau + \varepsilon \tag{2-6}$$

In order to implement the system to achieve this timing diagram, a chain of delay elements is used with a D flip-flop as shown in figure 2-16. The buffer is used to perform the job of the delay element in the chain. The input of the D of the flip-flop is connected to the output of the buffer and the clock is connected to the stop signal in order to make the flip flop acts as a comparator which will store the value of the start signal and make it available at its output at the rising edge of the stop signal. This technique is used to produce the thermometer code as explained previously.

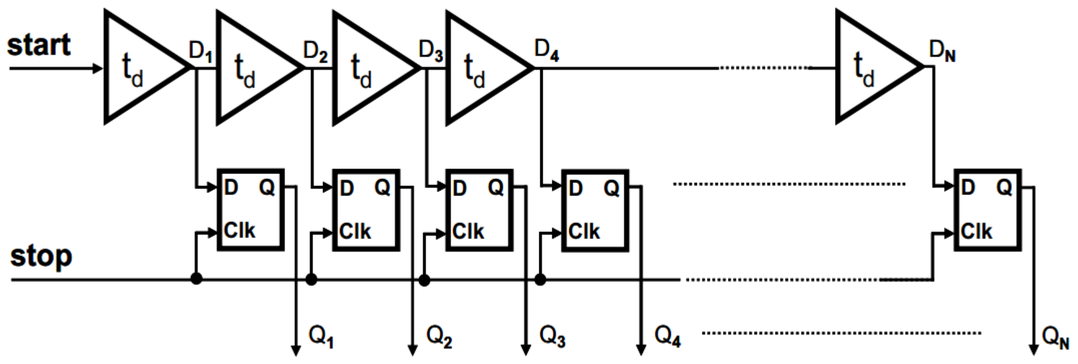


Figure 2-16 delay Line approach implementation [4]

The main disadvantage of this approach is the large size of the buffer as the designer is forced to use 2 stages of CMOS inverters that will lead to suffering from wasted area. More over the resolution of the ADC in this design is restricted by the delay of a single delay element that couldn't be smaller than τ.

### 2.3.2.3 Inverter Based Digital Delay Line Approach

The previous approach was depending on the delay chain technique that uses the buffer as a delay element, and as earlier we have revealed its pros and cons. In this technique the buffer is replaced by single CMOS inverter and so the resolution decreases resulting in better performance [6]. However, on the other hand there is some correlation due to common process steps during manufacturing of NMOS and PMOS devices which make this approach not preferable.

Sometimes the steps of these processes are systematic like the formation of the gate oxide and the gate lithography. However, there are also completely independent process steps such as the ion implantation for threshold voltage adjustment. This leads to systematic non-linearity of the converter characteristic that in turn imbalances the delays again and so the rising and falling edges are not the same any more.

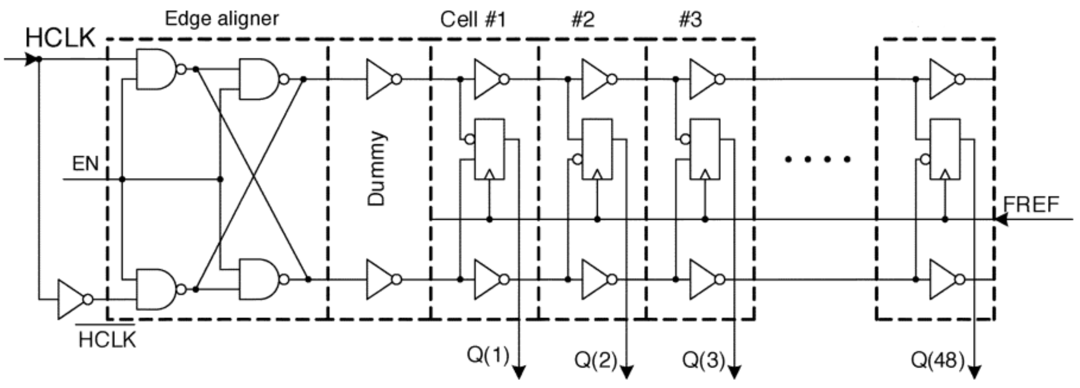


Figure 2-17 inverter based delay line implementation [6]

This approach is not recommended if the effect variations became quite obvious; so In order to overcome these problems a Symmetrical differential ended flip-flop is used. In this case we have two delay lines each is formed from CMOS inverters not buffers, one of them is for the original start signal and the other is for the inverted version of the start signal.

Both signals the original and the inverted are connected to the inputs of the differential ended flip-flop, and as usual the stop signal will be connected to the clock of the flip-flop. In order to over-come the inverting effect of the CMOS inverter the two signals must be twisted at the input of the flip-flop as illustrated in figure 2-17.

The placement of the edge aligner is due to the presence of variations in the CMOS process that leads to an inequality between the rising and falling edges of the original and inverted start signal, so it's placed to make them the same.

#### 2.3.2.4 Vernier Oscillator Approach

The previous implementation still suffers from many problems such as the restriction on the resolution as it depends on the delay of single delay element. In other words we can't achieve a certain delay smaller than the delay of the CMOS inverter/buffer. This approach has deviated from the usual path a little bit by replacing the delay line by slow, fast oscillators and phase detector [7]. This helped in enhancing the resolution of the ADC because it moved away from the single delay chain technique. Figure 2-18 shows the Vernier oscillator approach implementation

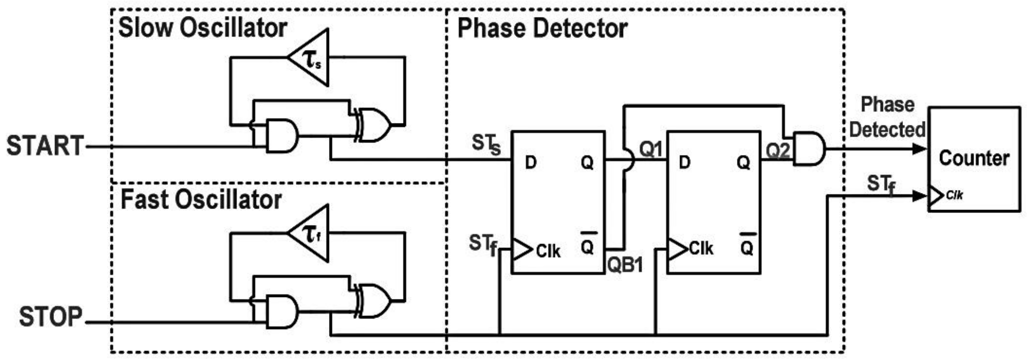


Figure 2-18 Vernier oscillator approach implementation [7]

The basic idea of this technique is that the start signal triggers the slow oscillator and the stop signal triggers the fast one. Since the start signal is leading the stop by ΔT, the two signals of the oscillators are connected to the input of phase detector which is responsible for subtracting the phase of the two signals till locking takes place. Locking takes place when the signal of the fast and slow oscillators became aligned on the same edge as shown in figure 2-19. Once locking took place it will disable the counter and it will stop counting. The difference between the two signals can calculate by the aid of the number of counts using this relation

$$\Delta T = N \times (T_s - T_f) \qquad (2\text{-}7)$$

Where N is the number of counts, Ts is the period of the slow oscillator, and Tf is the period of the fast oscillator.

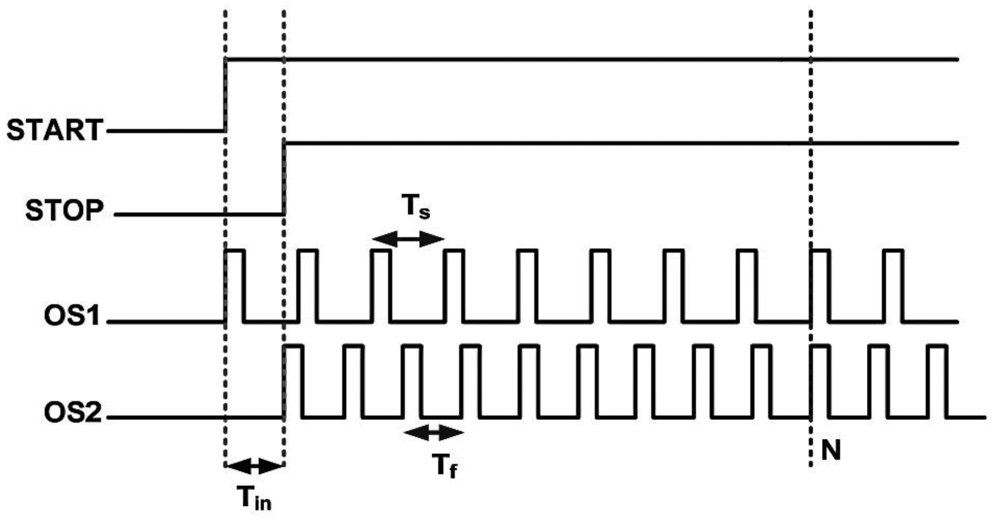


Figure 2-19 Vernier oscillator technique timing diagram [7]

Although this approach solved the problem of resolution in the delay line and the mismatch in the CMOS inverter, it's restricted by the delay of the phase detector. This is solved by modifying the design of the phase detector in order to achieve smaller resolution and so better performance.

#### *2.3.2.5 Vernier Delay Line Approach*

The Final technique that is considered the most efficient one as it compromises all the disadvantages of the above approaches and it also has most of their advantages is the vernier delay line technique.

In order to solve the resolution problem that is restricted by the delay of single delay element; this technique used two delay lines rather than using only one to achieve differential delay. The first delay line is used to delay the start signal; it consists of a chain of delay elements that have delay larger than the delay of the delay elements used in the second delay line which are used for delaying the stop signal [4] as shown in figure 2-20.

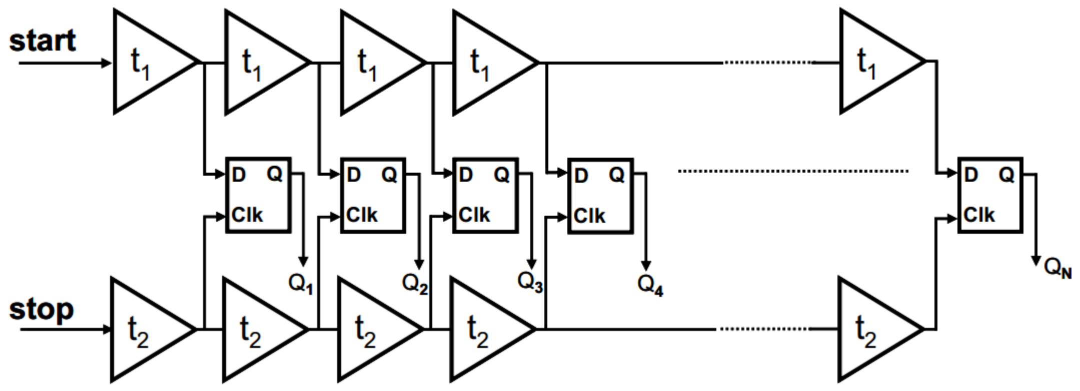


Figure 2-20 Vernier delay line technique core [4]

By using this idea the resolution will depend on the difference between the delays of the two cells not their pure delay which is the main advantage of this technique. The difference between the start and stop signal in this case could be given by the following relation

$$\Delta T = N \times (t_1 - t_2) \tag{2-8}$$

Where N is the number of stages of the delay line, t1 is the delay of the buffer in the delay line of the start signal, and similarly t2 is for the stop signal.

# Chapter 3: VTC Analysis

Our project is to design a 5GS/s 4-bit nyquist rate time based ADC, which is the fastest achieved speed with this number of bits. We will use 65nm technology to implement our design. We have no restriction on power consumption and any reasonable dynamic range is acceptable.

In this chapter will discuss the VTC block and its analysis including an explanation of the operation of the circuit, optimization of the output linearity and schematic diagram of the VTC. Also we will present the simulations results and layout of this block.

In chapter 4 we will discuss the TDC block and in chapter 5 we will present the simulation results and performance of the overall system.

## 3.1 Operation of VTC Block

The main role of VTC is to convert from analog signal (voltage) to time and this conversion is done by using inverter and control the rate which the capacitor connected to the output of it is charged or discharged by the analog input signal. This architecture is called current starved inverter [1].

### 3.1.1 Current Starved Inverter Cell

As shown in figure 3-1 it is an example of current starved inverter. The upper two transistors (M1 and M3) are the normal inverter, while M2 is to control the current flows in the inverter when the capacitor is discharging.

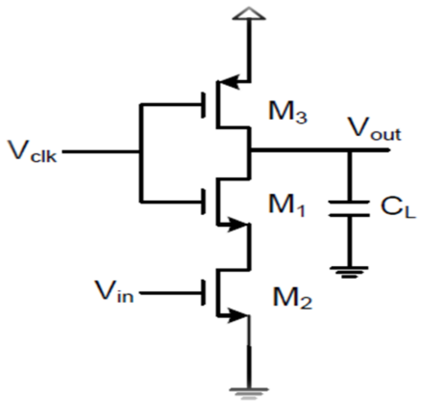


Figure 3-1current starved inverter [1]

By controlling the discharging current of $CL$ according to equation 3-1, through the varying of the input voltage $Vin$, we control the time it takes $Vout$ to reaches a certain voltage, say the threshold voltage of another inverter driven by $Vout$.

$$I = c\frac{dVout}{dt} \tag{3-1}$$

The main problem with this delay cell is the nonlinearity between the controlled voltage ($Vi$) and the delay value. There are 2 main reasons for nonlinearity we will discuss them.

#### 3.1.1.1 *Tracking error*

In real life, $Vin$ is not varying slowly, If $Vin$ is assumed to vary linearly, then $V$c falls down nonlinearly as shown in Figure 3-2 (solid line). Thus $V$c crosses $Vt$ at a time that is different than the case when $Vin$ is constant. If the $Vin$ is sampled using sample and hold block so $Vin$ will be constant while $V$c is falling (dotted line in Figure 3-2). If $Vin$ is increasing with time, this means that the current discharging the capacitor increases, and so the output voltage will cross the threshold earlier than the explicit S/H case. But if $Vin$ is decreasing with time, the output voltage will cross the threshold latter than the explicit S/H case [1].

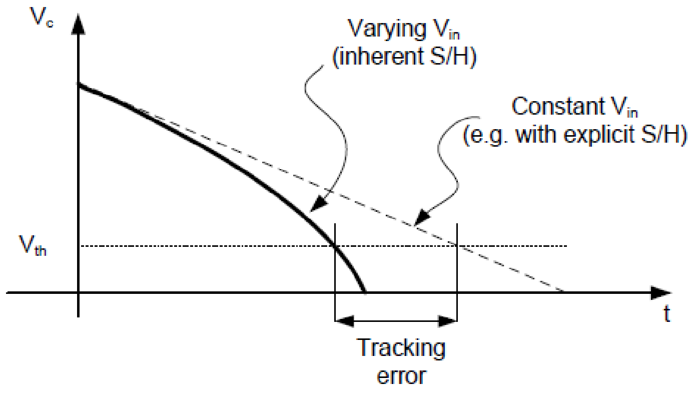
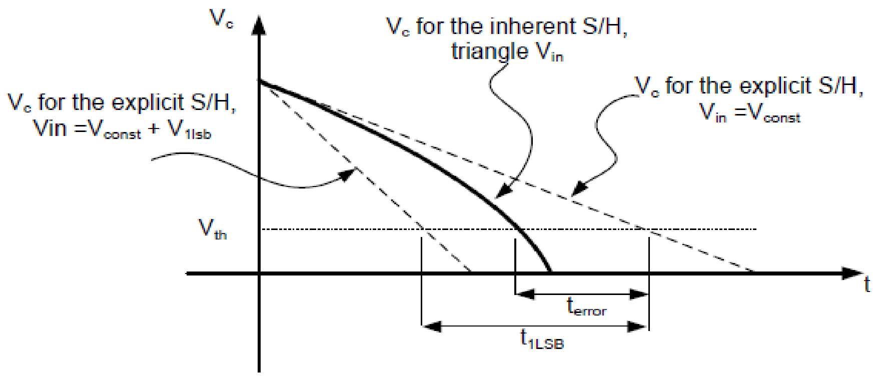


Figure 3-2 tracking error [1]

The difference between the two times (for implicit S/H and for constant $Vin$) is called the tracking error. It will limit the maximum frequency of the $Vin$ for a given resolution in terms of the number of bits.

The decision to use sample and hold block or not is depend on the tracking error, if it is less than the time corresponding to 1 LSB change in the $Vin$ ($t1_{LSB}$ ) (this 1 LSB change is calculated for 2 values of $Vin$. In each value, $Vin$ is assumed to be explicitly sampled and hence constant in the sampling window) so we don't need to use sample and hold block. This is shown in Figure 3-3. Note that because $Vin$ is constant, the two dotted lines have constant (but different) slopes. This is because $Vin$ increases from $Vconst$ to $Vconst$ + $V1_{LS}$, where $V1_{LSB}$ is the voltage corresponding to 1 LSB (= Full scale/$2^n$) where n is the number of bits.



$V_c$
$V_c$ for the inherent S/H, triangle $V_{in}$
$V_c$ for the explicit S/H, $V_{in} = V_{const}$
$V_c$ for the explicit S/H, Vin $= V_{const} + V_{1lsb}$
$V_{th}$
t
$t_{error}$
$t_{1LSB}$

Figure 3-3 relation between $t_{error}$ and $t1_{LSB}$ [1]

#### 3.1.1.2 Non ideality of transistor

As we explained above we control the discharging current of $CL$, through the varying of the input voltage $Vin$. And we need current to be constant so we will adjust transistor to operate is saturation mode. The relation between the current and $Vin$ is given by

$$I = K(V_{GS} - V_{th})^2 \tag{3-2}$$

And this relation shows that the current doesn't change linearly with $Vin$.

### 3.1.2 The Main Core of VTC

Figure 3-4 shows that M1-M4 make up a voltage-starved inverter, while M5-M6 form a standard CMOS inverter used to sharpen the edges of the signal Vout (t). The gate input to M3 is the input signal to the VTC (Vin) [2]. In this analysis, it will be assumed that Vin can be considered constant over a single VTC conversion cycle. The reason for this is that the VTC is only sensitive to the input for a short time during the switching process, so it effectively samples the input. The gate input to M4 is a DC bias voltage (Vconst) used to tune the linearity of the VTC.

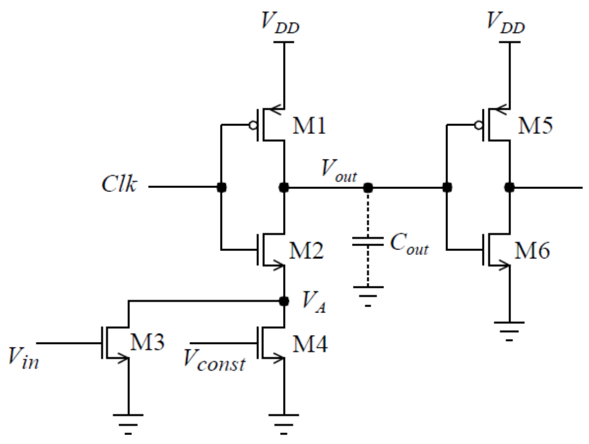


Figure 3-4VTC core circuit [2]

Since the M1-M2 inverter has starving devices between M2 and ground but not between M1 and VDD, the delay which falling edges of CLK take to be passed through VDD is minimized.  However, rising edges of CLK will be slowed down by the starved inverter, depending on the value of Vin. The delay on this edge, and how it varies with Vin, is what we are interested in analyzing.

A basic summary of the VTC operation is as follows: When a rising edge occurs on Clk(t), Cout starts to discharge so Vout(t) begins to decrease from VDD at a rate dependent on Vin. When this ramping signal reaches the threshold of the M5-M6 inverter, a rising edge is triggered on the inverter output.

## 3.2 Design Procedure

We need to design VTC operates with clock 5Gsample/s, good linearity between input voltage (Vin) and delay, fine dynamic range of input voltage ( difference between smallest input and largest input while VTC is working right ) and fine range of delay to release specifications on TDC block.

We will design range of delay to be 50 psec and dynamic range of input not to be less than 100 mv so it is hard to get these specifications from single cell of VTC as we operate at high frequency. The period of CLK is 200psec and we interested at rising edges only so we have only 100psec so it is hard to have delay range of 50 psec from it with fine linearity so we will use differential VTC.

### 3.2.1 VTC Half Cell

First we will design VTC half cell and then we will use two half cells to form differential VTC. Figure 3-5 shows that M1-M4 make up a voltage-starved inverter, while M5-M6 form a standard CMOS inverter used to sharpen the edges of the signal Vout(t) and M7-M10 form a standard CMOS buffer to avoid loading effect.

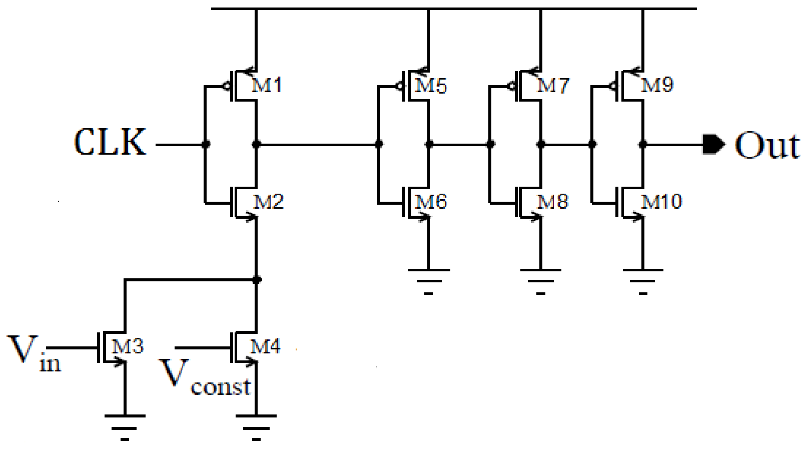


Figure 3-5 VTC half cell [2]

We need to have delay range of 25psec from the single cell in order to have delay range of 50psec from the final differential VTC.

#### *3.2.1.1 Duty-Cycle Adjustment*

Since we are interested in rising edges of CLK only, we have only 100p to change the delay of discharging of Cout. So we will adjust the duty cycle of the CLK to have more flexibility in the minimum delay and maximum delay so the period which the CLK is VDD will be 125 psec.

We can design circuit to change the duty cycle also we can do this from CLK generator.

#### *3.2.1.2 Biasing Adjustment*

We are interested to have high linearity between Vin and delay as well as fine dynamic range (not less than 100m). First Vconst will be the minimum voltage to have M4 in saturation mode, so if Vconst is equal Vth so any voltage of source of M2 will make M4 in saturation mode. Second we need the current of transistor M4 is enough to discharge the Cout to the threshold of the inverter (M5-M6) and to determine the smallest delay and largest delay we want as we can change them through increase or decrease the current of M4.

As well as VB (biasing voltage of M3) will be the minimum voltage to have M3 in saturation mode then we will sweep on Vin to have the best linearity and the proposed delay range.

### 3.2.2 Differential VTC

The VTC is a differential circuit composed of two half-cells. Each half cell is fed by the same clock and bias voltages with complementary RF inputs. So we will double delay range and dynamic range and improve the linearity so the delay range will be larger than 50psec and dynamic range will be larger than 100mv.

## 3.3 Schematic Simulation Results

We have designed various VTC blocks which have different specifications. We can control in the design parameters(dynamic range of the input signal and delay range position) by controlling the current passes in the transistor connected to Vin(control dynamic range) and by controlling the current passes in the transistor connected to Vconst (control delay range position)

In the following sections we will present the results of these different designs.

### 3.3.1 5 Giga Sample/s VTC

We have 2 designs achieve this speed but they are different in dynamic range and delay range position. The duty cycle of the CLK(200 ps) is 62.5 %.

#### *3.3.1.1 VTC with 200 mV Dynamic Range*

Figure 3-6 shows the delay between Vc and the CLK, Figure 3-7 shows the delay between the output of the VTC half cell and the CLK and figure 3-8 shows the delay between the two half cells.

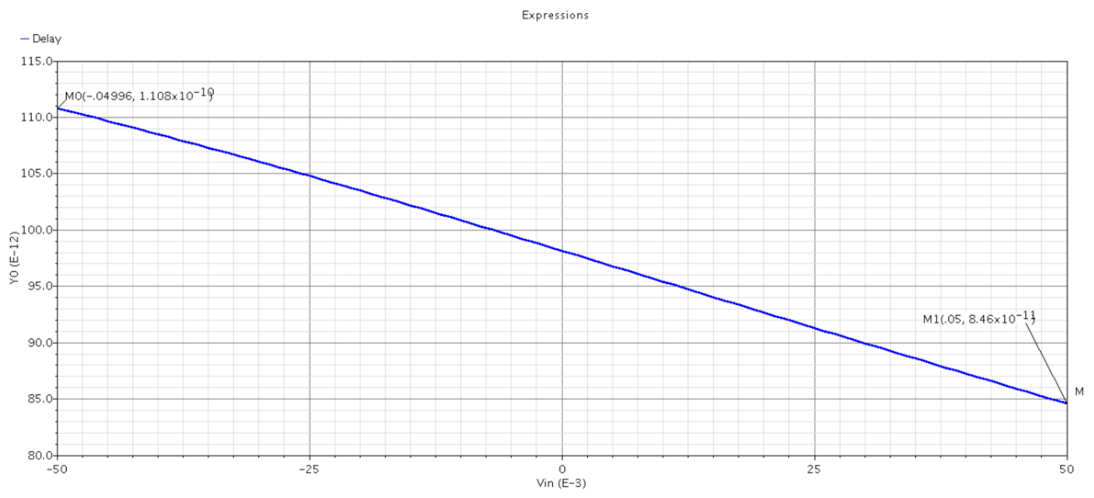


Figure 3-6 the delay between Vc and the CLK of 5 GSample/s VTC with 200mV dynamic range

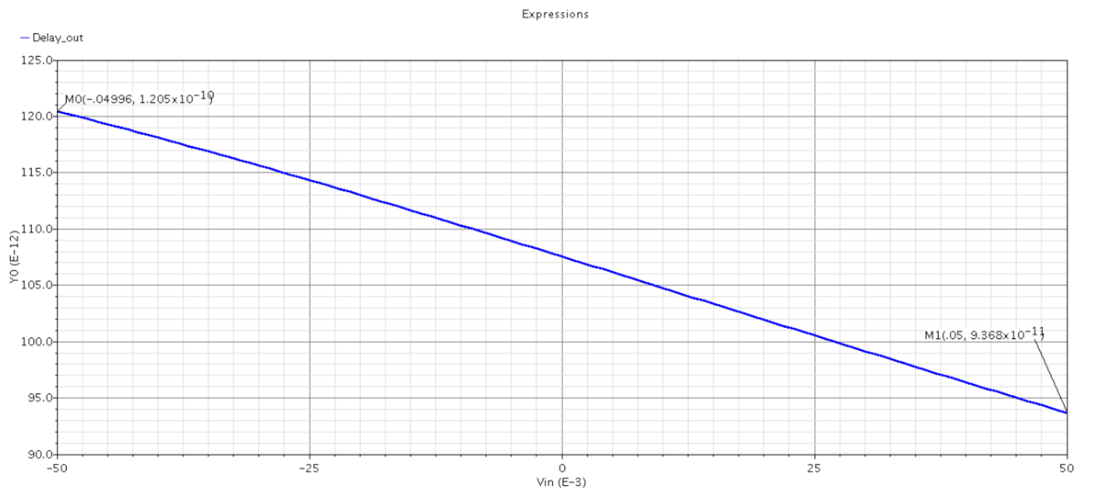


Figure 3-7 the delay between the output of the VTC half cell and the CLK 5 GSample/s VTC with 200mV dynamic range

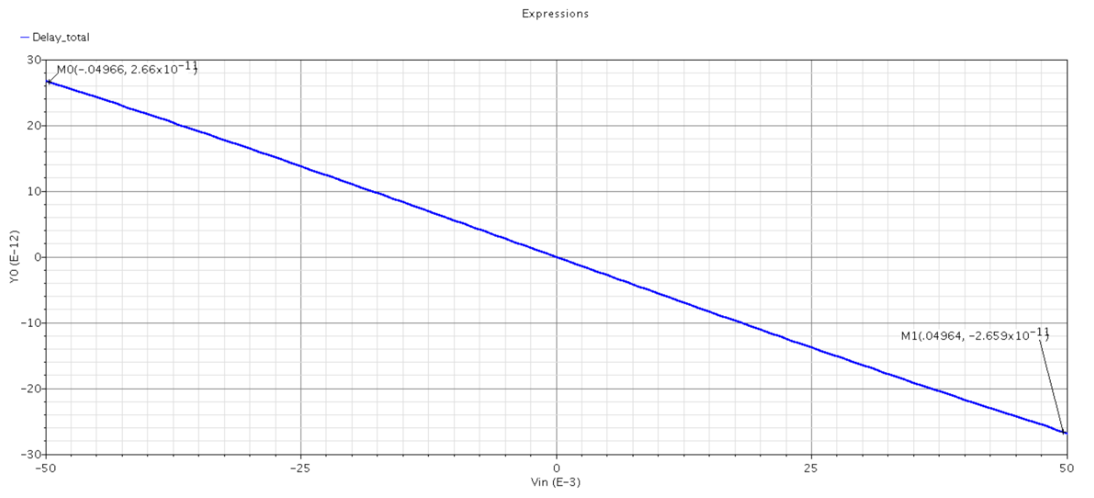


Figure 3-8 the delay between the two half cells 5 GSample/s VTC with 200mV dynamic range

Figures 3-6 and 3-7 show that the delay range of the half cell =25 ps while figure 3-8 shows that differential delay between 2 half cells =50 ps.

#### 3.3.1.2 VTC with 140 mV Dynamic Range

Figure 3-9 shows the delay between Vc and the CLK, Figure 3-10 shows the delay between the output of the VTC half cell and the CLK and figure 3-11 shows the delay between the two half cells.

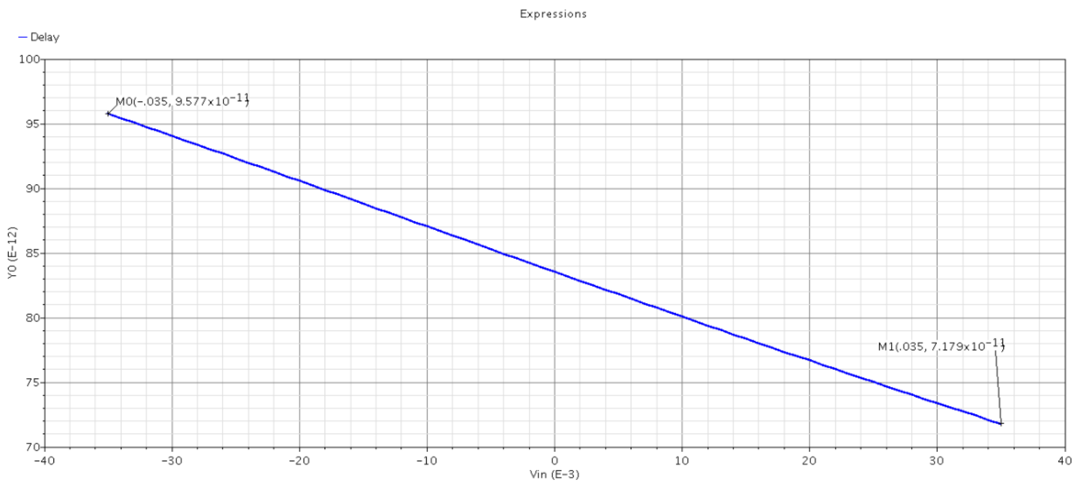


Figure 3-9 the delay between Vc and the CLK of 5 GSample/s VTC with 140mV dynamic range

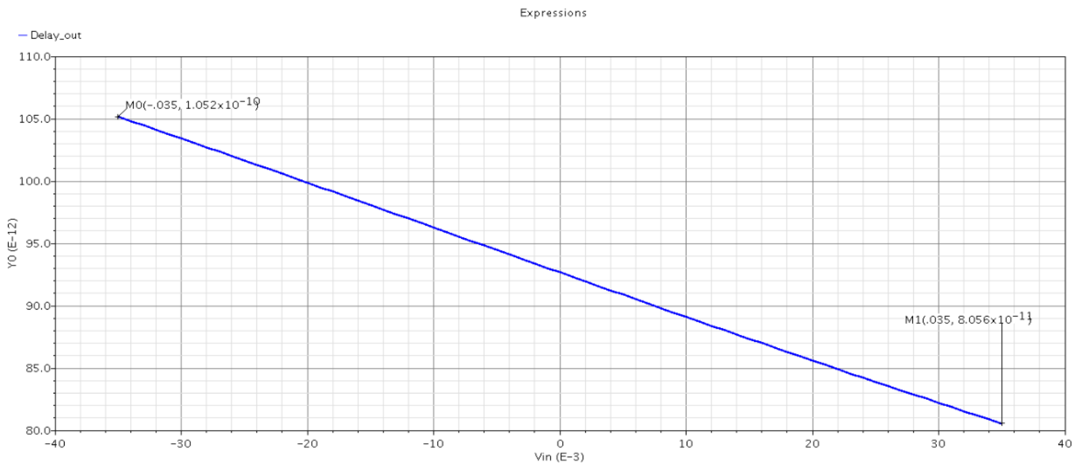


Figure 3-10 the delay between the output of the VTC half cell and the CLK 5 GSample/s VTC with 140mV dynamic range

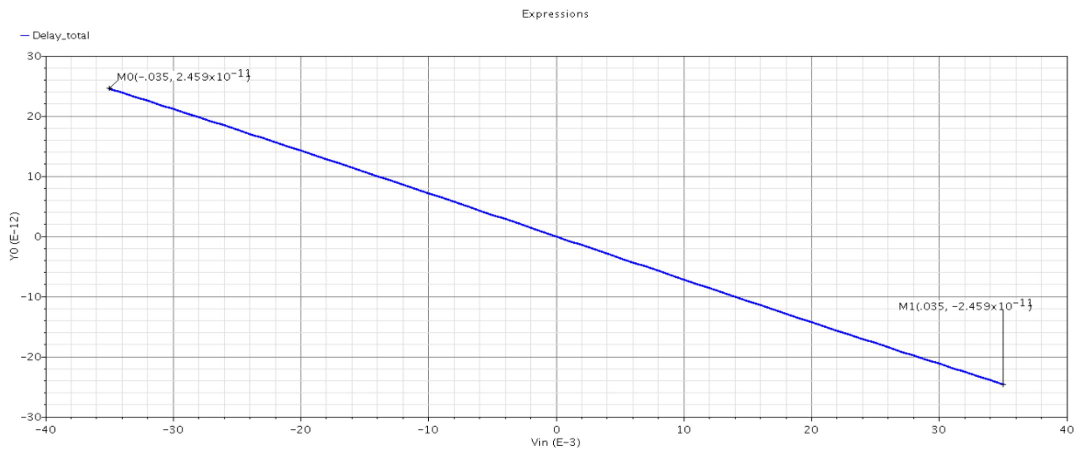


Figure 3-11 the delay between the two half cells 5 GSample/s VTC with 140mV dynamic range

Figures 3-9 and 3-10 show that the delay range of the half cell =25 ps while figure 3-11 shows that differential delay between 2 half cells =50 ps.

### 3.3.2 6.6 Giga Sample/s VTC

We have 2 designs achieve this speed but they are different in dynamic range and delay range position.. The duty cycle of the CLK(150 ps) is 66.7 %. These designs may help us in future to make ADC work on speed higher than 5 Giga sample/s

#### *3.3.2.1 VTC with 240 mV Dynamic Range*

Figure 3-12 shows the delay between Vc and the CLK, Figure 3-13 shows the delay between the output of the VTC half cell and the CLK and figure 3-14 shows the delay between the two half cells.

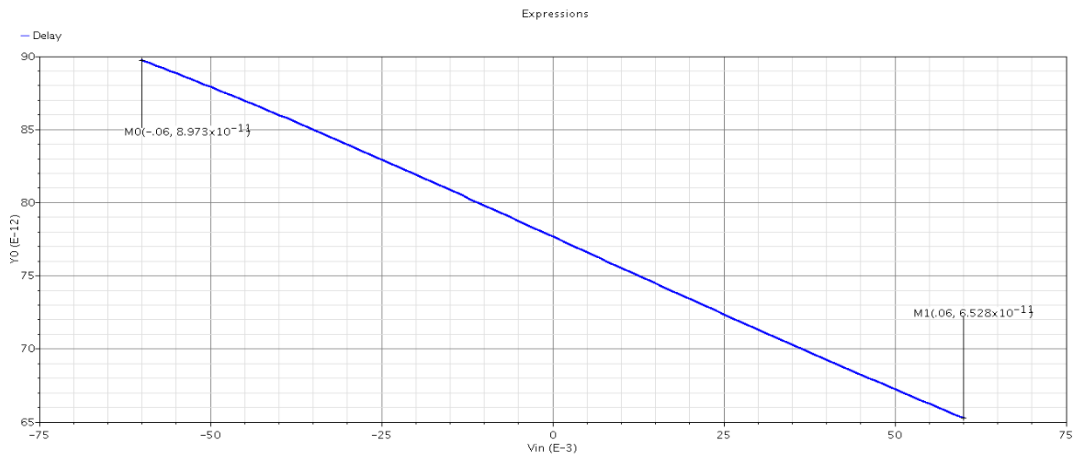


Figure 3-12 the delay between Vc and the CLK of 6.6 GSample/s VTC with 240mV dynamic range

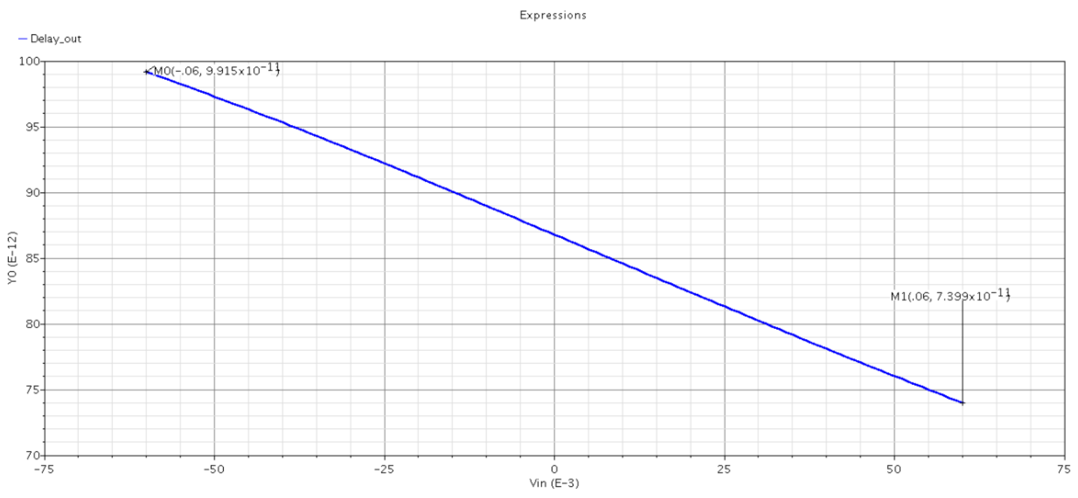


Figure 3-13 the delay between the output of the VTC half cell and the CLK 6.6 GSample/s VTC with 240mV dynamic range

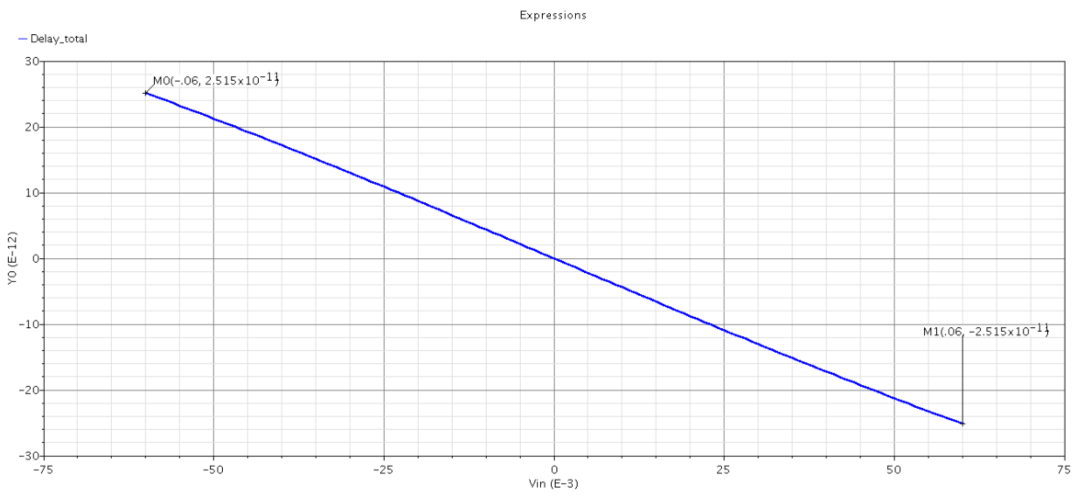


Figure 3-14 the delay between the two half cells 6.6 GSample/s VTC with 240mV dynamic range

Figures 3-12 and 3-13 show that the delay range of the half cell =25 ps while figure 3-14 shows that differential delay between 2 half cells =50 ps.

#### *3.3.2.2 VTC with 180 mV Dynamic Range*

Figure 3-15 shows the delay between Vc and the CLK, Figure 3-16 shows the delay between the output of the VTC half cell and the CLK and figure 3-17 shows the delay between the two half cells.

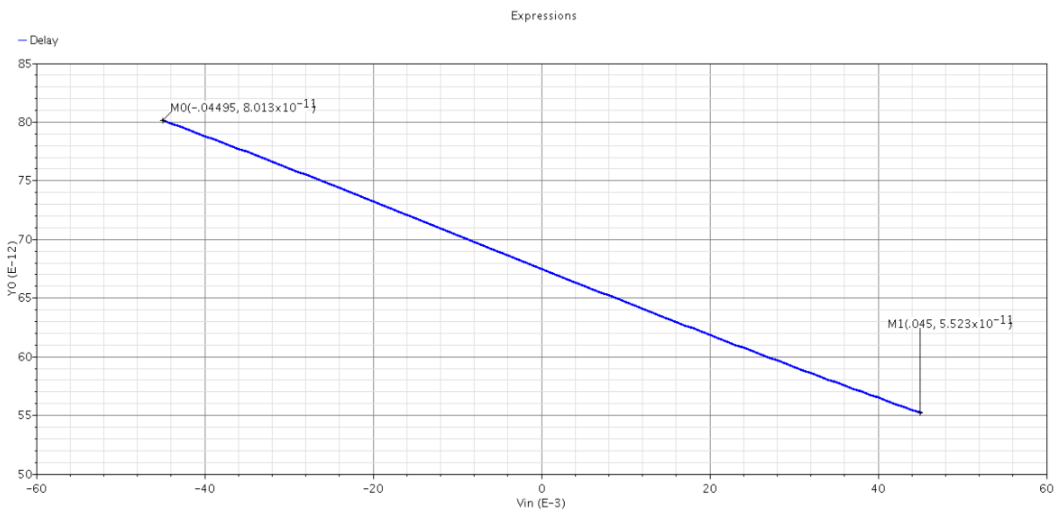


Figure 3-15 the delay between Vc and the CLK of 6.6 GSample/s VTC with 180mV dynamic range

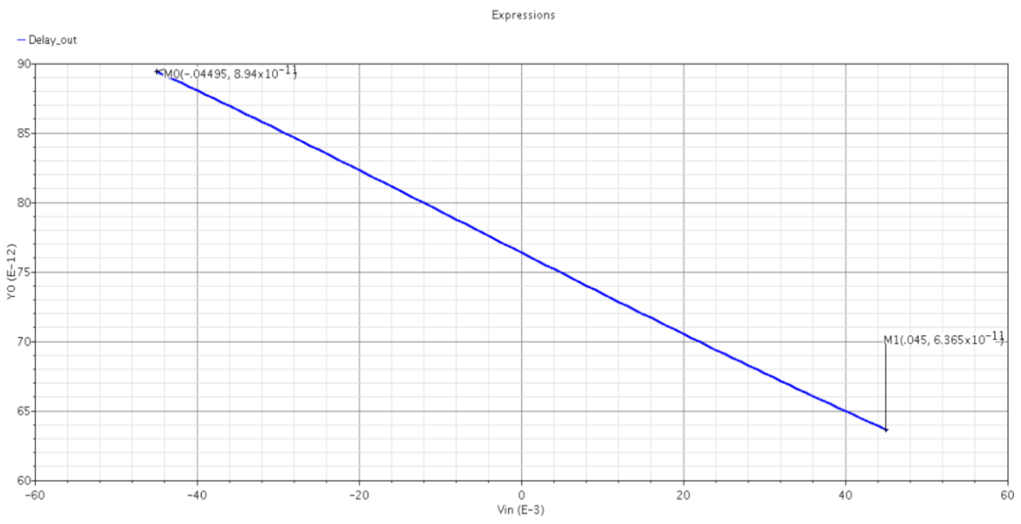


Figure 3-16 the delay between the output of the VTC half cell and the CLK 6.6 GSample/s VTC with 180mV dynamic range

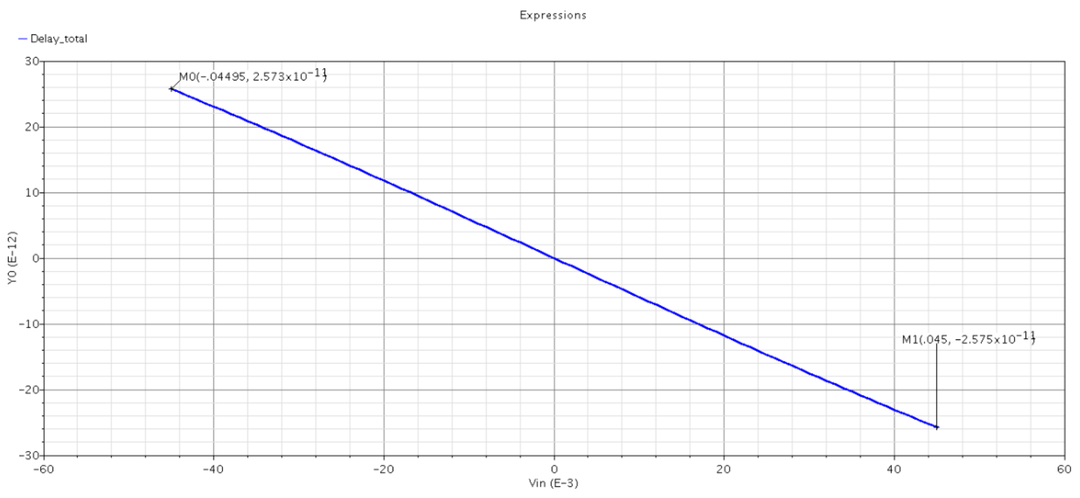


Figure 3-17 the delay between the two half cells 6.6 GSample/s VTC with 180mV dynamic range

Figures 3-15 and 3-16 show that the delay range of the half cell =25 ps while figure 3-17 shows that differential delay between 2 half cells =50 ps.

## 3.4 Layout and Post Layout Simulation Results

We have built only the lower dynamic range VTCs on layout level as they have better linearity.

### 3.4.1 5 Giga Sample/s VTC Layout

Figure 3-18 shows the layout of 5 Giga Sample/s VTC with dynamic range=140 mV

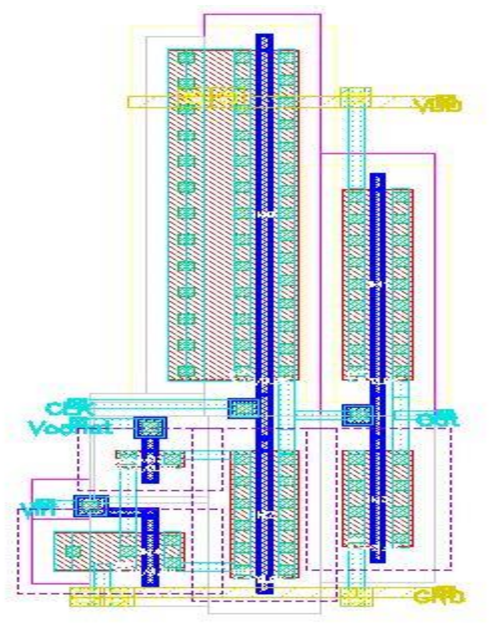


Figure 3-18 5 Giga Sample/s VTC layout

Figures 3-19 show that the delay range of the half cell =25 ps while figure 3-20 shows that differential delay between 2 half cells =50 ps.

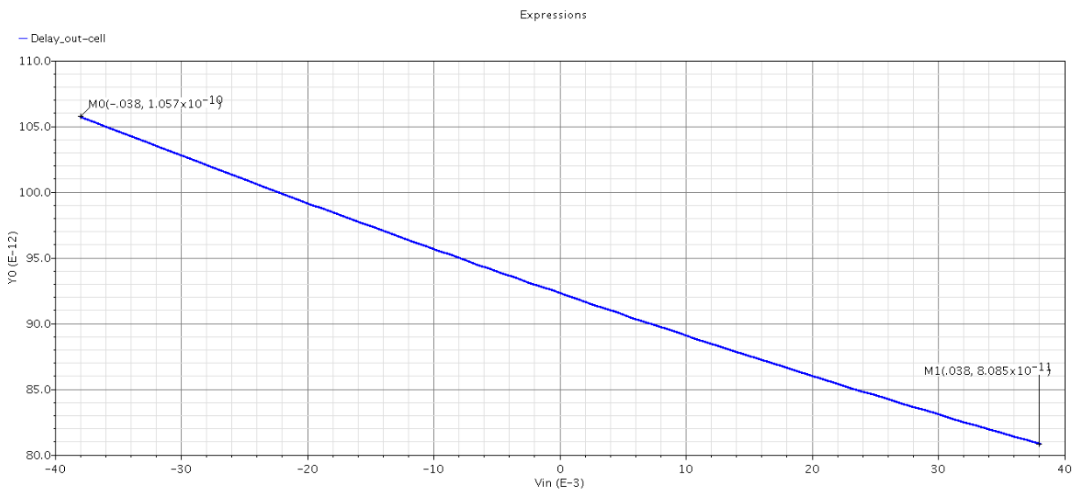


Figure 3-19 the delay between the output of the VTC half cell and the CLK 5 GSample/s VTC with 140mV dynamic range after the layout

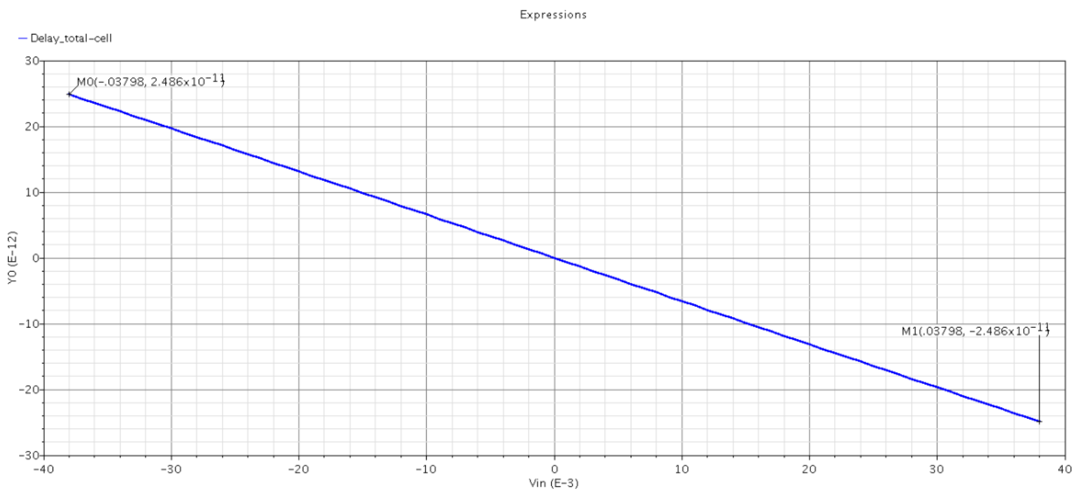


Figure 3-20 the delay between the two half cells 5 GSample/s VTC with 140mV dynamic range after the layout

After layout the dynamic range become 152 mv.

### 3.4.2 6.6 Giga Sample/s VTC

Figure 3-21 shows the layout of 6.6 Giga Sample/s VTC with dynamic range=180 mV

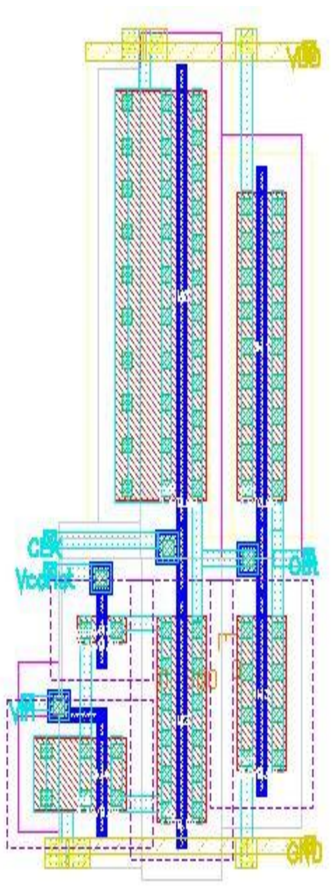


Figure 3-21 6.6 Giga Sample/s VTC layout

Figures 3-22 show that the delay range of the half cell =25 ps while figure 3-23 shows that differential delay between 2 half cells =50 ps.

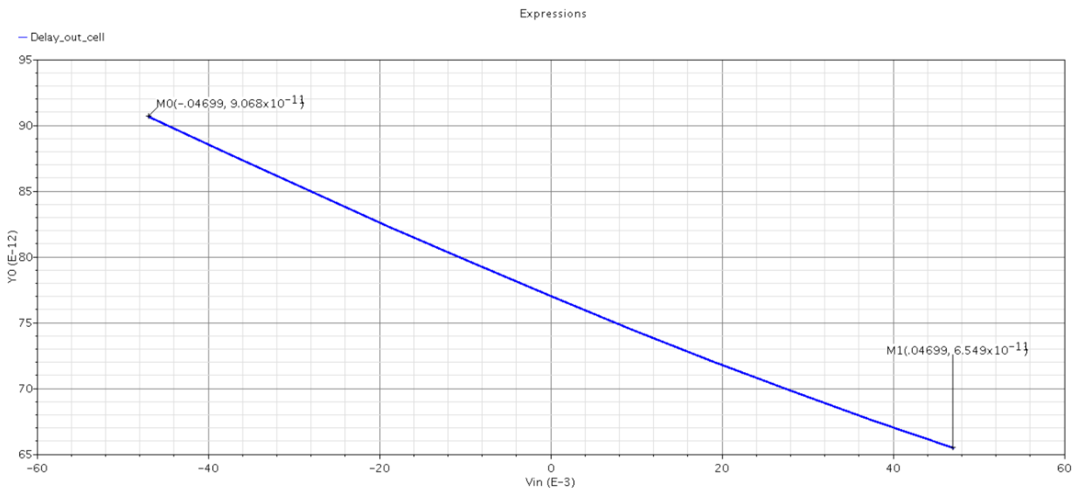


Figure 3-22 the delay between the output of the VTC half cell and the CLK 6.6 GSample/s VTC with 180mV dynamic range after the layout

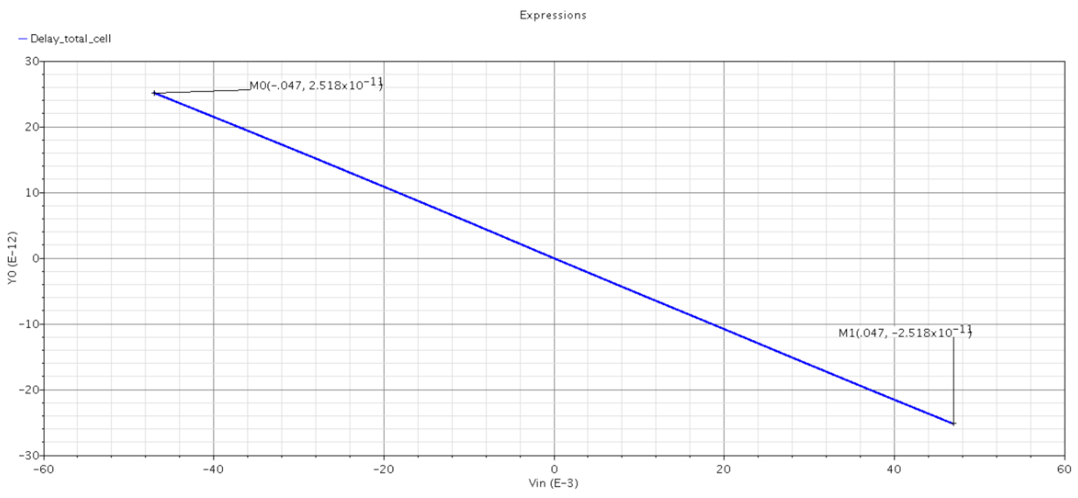


Figure 3-23 the delay between the two half cells 6.6 GSample/s VTC with 180mV dynamic range after the layout

After layout the dynamic range become 188 mv.

# Chapter 4: 4-bit Vernier Delay Line TDC Analysis

In This chapter we will illustrate the TDC block used in our design, which architecture we have used to achieve the specifications of our design, and the simulation results of the TDC block.

Figure 4-1 shows the TDC block. It consists from vernier delay line (VDL), decision flip flops and retiming circuit [2]. Also there is a encoder as a final block after retiming circuit which converts the 15-bit thermometer-coded output of the VDL (T1 through T15) to a 4-bit binary output (B0 through B3).

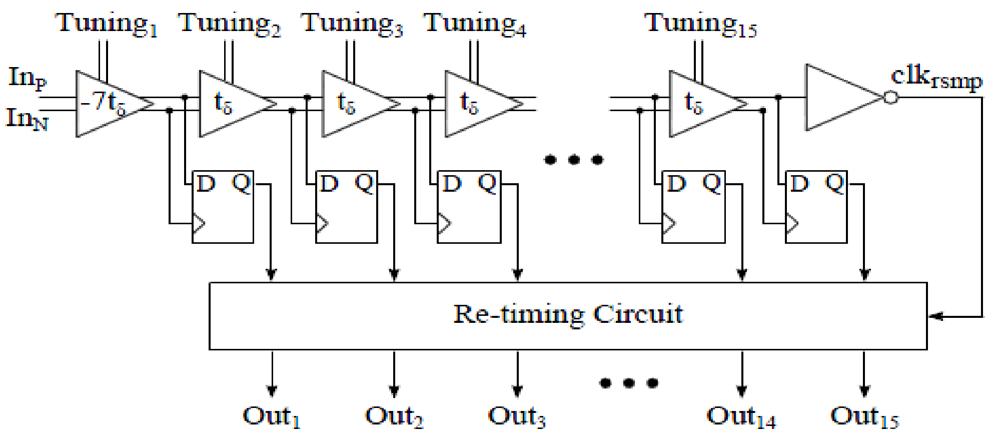


Figure 4-1TDC block [8]

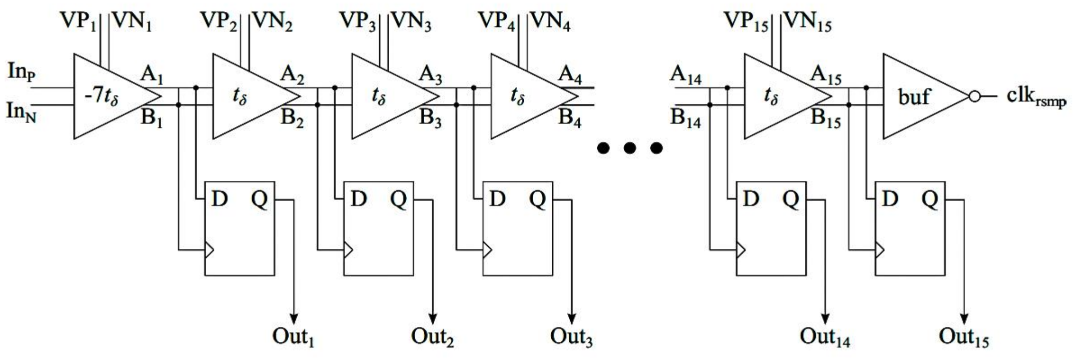
In the following sections we will illustrate each part of TDC block

## 4.1 Delay Line Structure

As mentioned in the above sections the pros and the cons of many techniques used in implementing the time to digital converter block. After comparing all the above approaches we concluded that the vernier delay line is the most appropriate technique that will fit our design due to the required high speed. The used design is quite similar to the vernier delay line approach but with a little modification. Since the difference between the 2 signals coming out from the VTC block is equal to 50psec and so this range is divided on the fifteen cells in the TDC and so the delay of each cell can be calculated from relation 4-1

$$T_\delta = \frac{50\ Psec}{2^4} = 3.125 \text{ psec} \qquad (4\text{-}1)$$

The first cell is used to delay the start and stop signals by -7tδ (or -21.875psec) and the other remaining cells each delay the two signals by tδ (or 3.125psec) till the end of the delay chain. The output of each delay cell is connected to the input of the flip-flop and similar to the vernier delay line; the cell shown in figure 4-2 is differentially ended so there are two delay chains one of them connected to the D and the other to the CLK input of the flip-flop.



$VP_1$ $VN_1$ $VP_2$ $VN_2$ $VP_3$ $VN_3$ $VP_4$ $VN_4$ $VP_{15}$ $VN_{15}$
$In_P$
$In_N$
$-7t_\delta$
$t_\delta$
$A_1$ $B_1$ $A_2$ $B_2$ $A_3$ $B_3$ $A_4$ $B_4$ $A_{14}$ $B_{14}$ $A_{15}$ $B_{15}$
buf
$clk_{rsmp}$
D Q
$Out_1$ $Out_2$ $Out_3$ $Out_{14}$ $Out_{15}$

Figure 4-2delay line core [2]

The flip-flop in turn will act as a comparator or decision block at the time of the rising edge of the stop signal. The flip-flop outputs make up the 15-bit thermometer code representation of the TDC output. The output of the flip-flop will be a group of ones and zeros representing the period between the start and stop signal. The number of ones in the code represents the number of the shifted versions of the start signal whose rising edges were leading the rising edge of the stop signal. This code will enter the encoder that will count the number of ones in the code and convert them to the desired four bits in the digital domain. The first delay stage, a delay of −7tδ (or -21.875ps) is introduced between the signals start and stop. Each subsequent delay stage (2 through 15) adds a positive tδ (or 3.125ps) to the delay between the two signals. It can be seen that the delay between the two signals sweeps through the range of -7tδ to +7tδ as the signals travel through the VDL.

The main reason of using the first cell to delay the two signals by -7tδ is to make the delay cells in the delay chain cover all the time difference between the two signals as explained in figure 4-3. Assuming the two signals entering the TDC block were aligned and so ΔT=0. In order to convert this value to binary code first the two signals are being shifted by -7tδ and then pass through 15 delay cell each delay tδ. So when reaching the 7th cell the difference between both signals will be -1tδ; and after reaching the 15th cell the difference will be 7tδ. So it's quite obvious that using the first cell enabled the system to sweep on all the period.

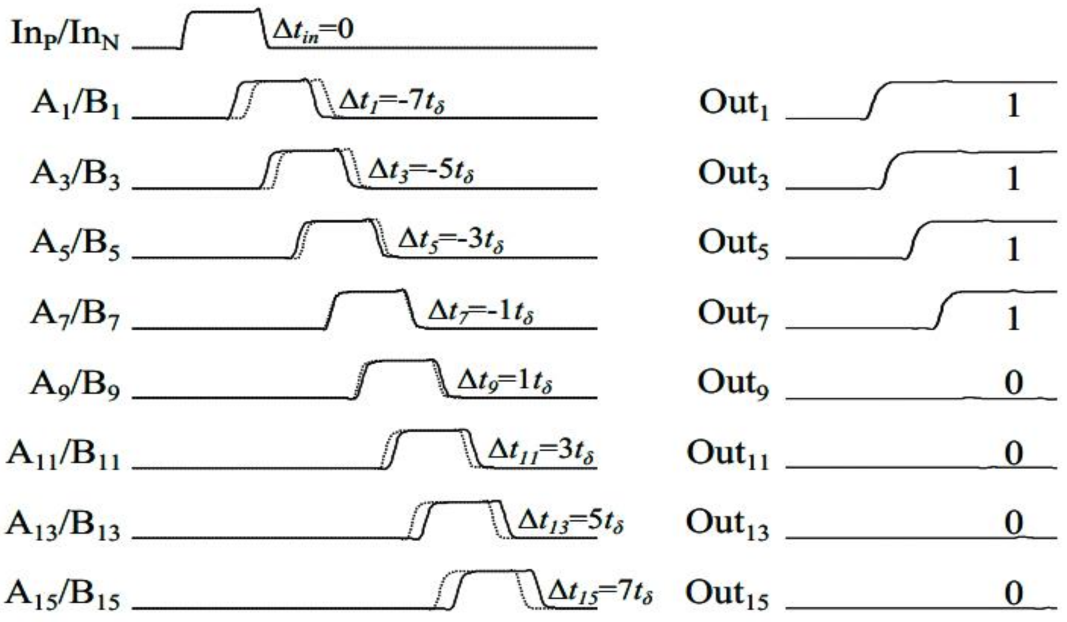


Figure 4-3 Delay line waveforms [2]

As mentioned before the flip-flop output will be '0' when its clock signal (Bi) arrives before its input signal (Ai), or in other words when Δti > 0. The output will be '1' when the data arrives before the clock, or when Δti < 0.

## 4.2 Delay Cell Structure

The delay line shown above in figure 4-2 consists of cascaded delay cells which is the core of the TDC.

In order to illustrate how the delay cell works, we are going to show the core of this block as shown in figure 4-4. Mainly it is implemented using starved inverter which consists of 6 transistors; the basic CMOS inverter is formed from the 2 transistors M4 and M5 that are followed by another inverter formed from the transistors M7 and M8.

For normal operation the transistors mentioned above must operate in linear (triode) region; however the devices M3 and M6 which control the amount of current passing through the inverter must operate in saturation region.

The transistors M1 and M2 are comparatively small as they have quarter the width of the remaining transistors, the purpose of using these devices will be illustrated in the calibration section but here we will only note that they are biased to ensure that a minimum amount of current is able to flow even if M3 and M6 enter cut-off mode, and to make sure that other transistors are functionally working if the threshold voltage had changed in fabrication.

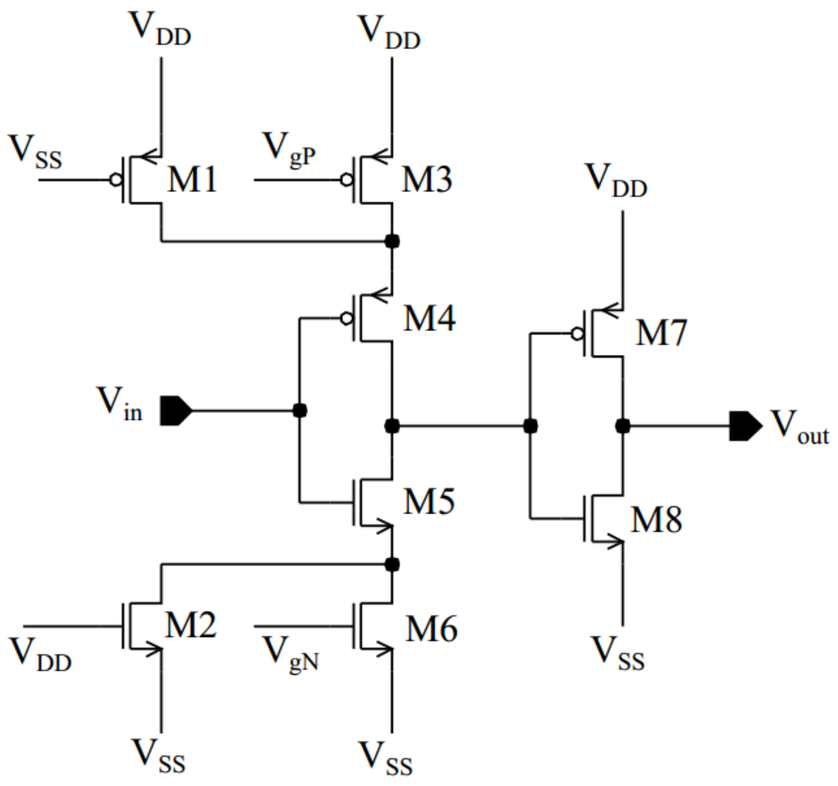


Figure 4-4Delay Cell Schematic [2]

The bias voltages Vgn and Vgp are used to control the rising and falling edges of the delay cell, this is done by controlling the amount of current passing through the starved inverter.

### 4.2.1 Delay Cell Tuning

As mentioned in the above section changing the value of the bias voltages Vgn and Vgp will control the rising and falling edges of the delay cell, now we will illustrate how this is done.

Knowing that the propagation time for the rising and falling edges tpLH and tpHL respectively are respectively are defined as the time taken by the output to reach 50% of the full scale are given by relations 4-2 and 4-3 [2].

$$\mathrm{tpLH} = \frac{\mathrm{VDD}*\mathrm{CL}}{2*\mathrm{IDP}} \tag{4-2}$$

$$\mathrm{tpHL} = \frac{\mathrm{VDD}*\mathrm{CL}}{2*\mathrm{IDN}} \tag{4-3}$$

Where VDD is the supply voltage, CL is the output load capacitance, and IDP and IDN are the drain currents of the PMOS and NMOS transistors respectively. In order to have equal propagation times for the rising and falling edges we have to make the drain currents of the PMOS and NMOS transistors equal as both transistors are charging or discharging the same load capacitance [2].

The drain currents in both transistors are given by the following equations

$$\mathrm{IDN} = \frac{\mu\mathrm{n}*\mathrm{Cox}*\mathrm{WN}}{2*\mathrm{L}} * (\mathrm{Vgs} - \mathrm{VTN})^2 \tag{4-4}$$

$$\mathrm{IDP} = \frac{\mu\mathrm{p}*\mathrm{Cox}*\mathrm{WP}}{2*\mathrm{L}} * (\mathrm{Vsg} - |\mathrm{VTP}|)^2 \tag{4-5}$$

where μn and μp are the electron and hole mobilities respectively, Cox is oxide capacitance per unit area, WN and WP are the NMOS and PMOS gate widths, L is the gate length, and VTN and VTP are the absolute NMOS and PMOS threshold voltages. So we have to adjust the device widths so that the ratio $\frac{\mathrm{Wp}}{\mathrm{WN}}$ is equal to $\frac{\mu\mathrm{n}}{\mu\mathrm{p}}$ and bias voltages Vgn and Vgp must be set together according to equation 4-6

$$\mathrm{Vgn} = \mathrm{VDD} - \mathrm{Vgp} \tag{4-6}$$

This results in equal propagation delays for the rising and falling edges.

So it’s quite obvious from equations 4-4 and 4-5 that changing the value of VgN and VgP will lead to changing the value of the drain current drawn from the NMOS and PMOS devices respectively, referring to equations 4-2 and 4-3 changing the drain currents will lead to change tpLH and tpHL and then giving the ability to control the delay of the signal; This is the main idea of the delay cell block.

Figure 4-5 shows the absolute delay produced by the delay as VgN is swept and Vgp is set to VDD-Vgn.

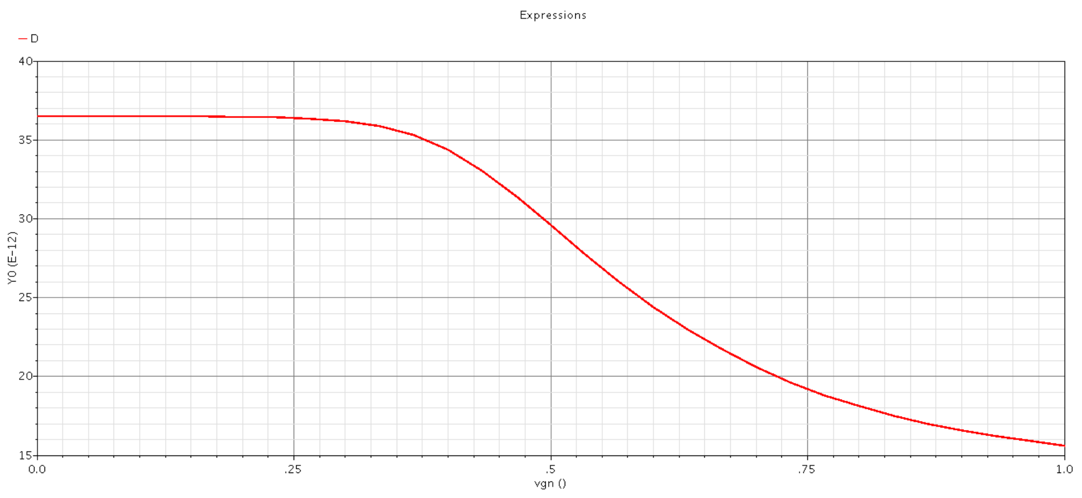


Figure 4-5 absolute delay produced by the delay

## 4.2.2 Differential Delay Cell

As shown in the previous figure no value of Vgn can produce the required delay (3.125ps)

Due to this problem one of the approaches mentioned above in the types of the TDC blocks was the vernier delay line approach. This approach depends on the placing 2 chains of delay cells one of them has higher delay than the other. So depending on this technique we can define the time tδ is the time difference between the 2 cells which can be easily achieved.

The delay cell will be differentially ended. To illustrate this on the above structure we just note that the delay cell of the start signal is considered as voltage controlled delay unit that could be controlled using the values of the bias voltages Vgn and Vgp. The delay cell of the stop signal has constant delay and has the same structure as the start; however the only difference is in replacing the Vgn by VDD and the Vgp by Vss. Figure 4-6 shows the block diagram of the start and stop Delay Cells

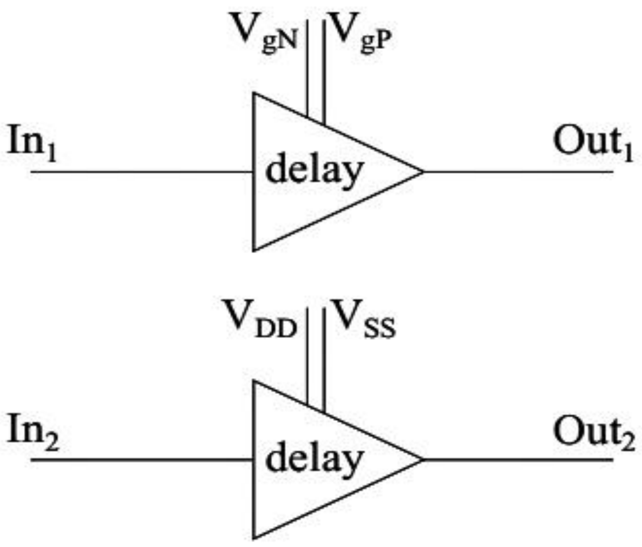


Figure 4-6 The start and stop Delay Cells [2]

Figure 4-7 shows the differential delay produced by the the delay as VgN is swept and Vgp is set to VDD-Vgn

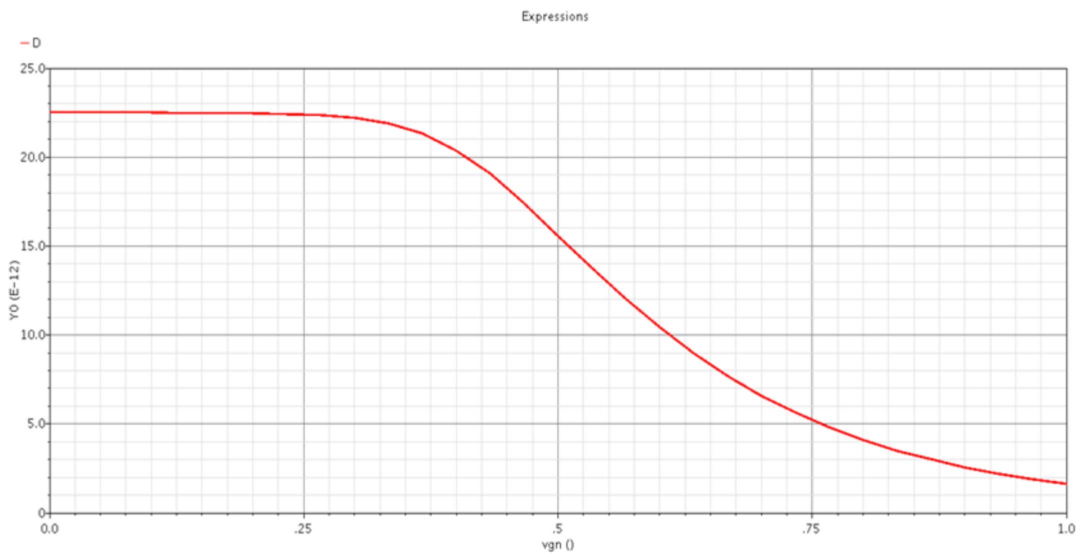


Figure 4-7 the differential delay produced by the the delay as VgN

For the delay of -7tδ, the configuration of figure 4-8 is used. The same delay cell that is shown in figure 4-6 is used, but with 2 elements in series each produces half the required delay. To produce a delay in the opposite direction of the tδ circuit, the top path introduces minimum delay while the bottom path delay is controlled using VgN and VgP.

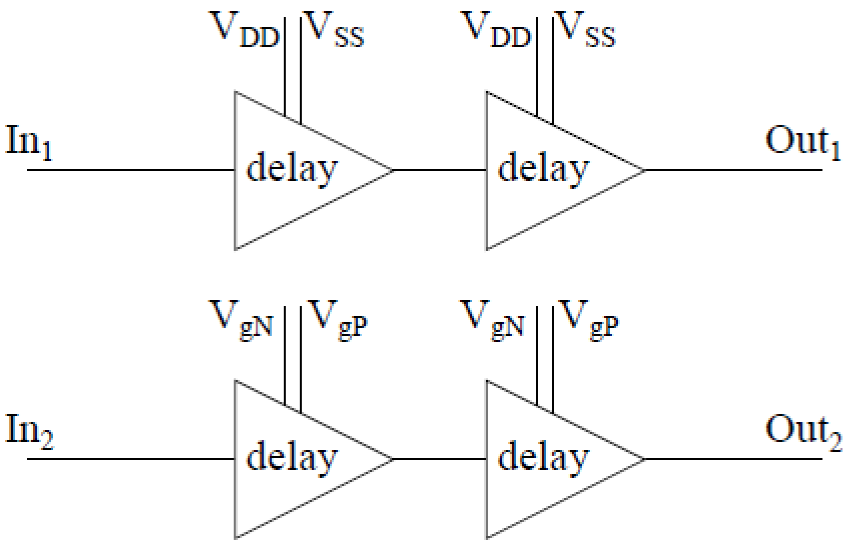


Figure 4-8 differential delay block for generating -7tδ [2]

Using 2 elements instead of 7 decreases the power consumption and layout area for the circuit. Although -7tδ (-21.875ps) is within the tuning range of the absolute delay (Figure 4-5) and it would be possible to save much more area by using a single delay element in the bottom path and no element at all in the top path, this was considered too risky as any process variation beyond what the simulator predicts could make it impossible to reach the desired delay.

### 4.2.3 Delay Cell Layout

The completed layout for the delay cell is shown in figure 4-9. The total area is 7.82um × 3um

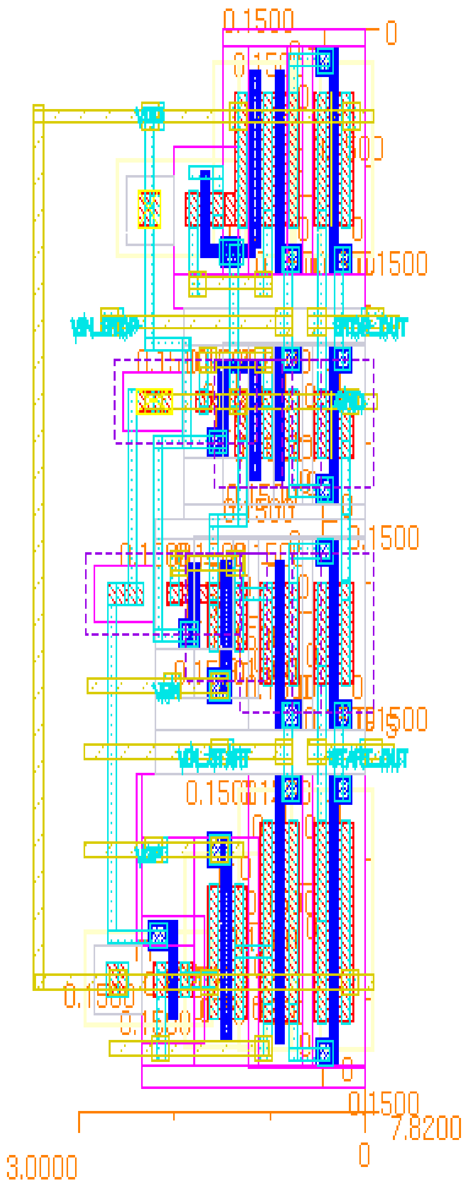


Figure 4-9 the layout of the delay cell

Figure 4-10 shows the absolute delay produced by the delay after layout as VgN is swept and Vgp is set to VDD-Vgn

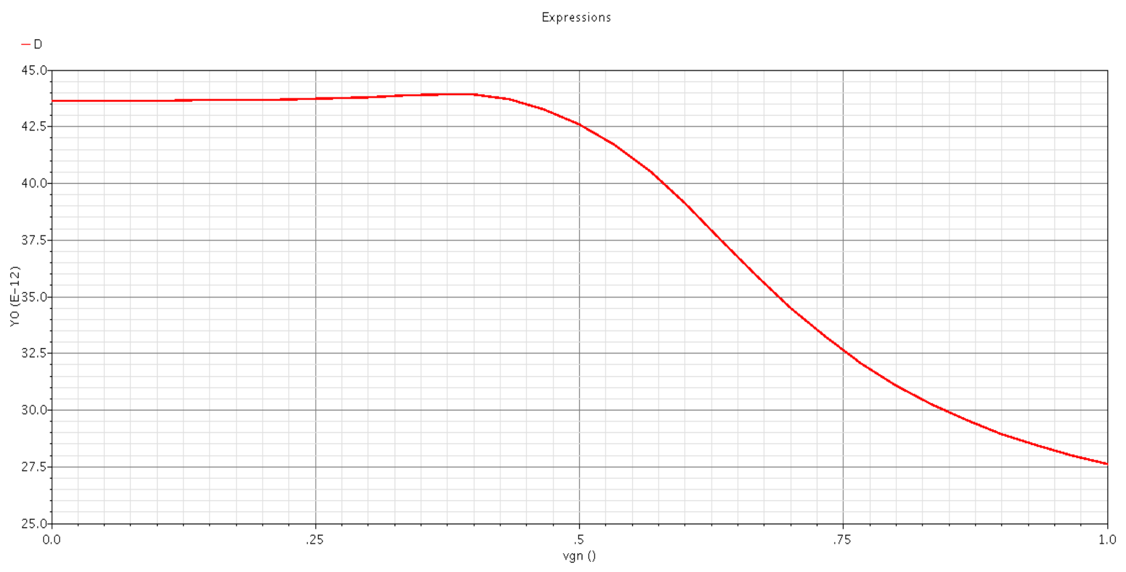


Figure 4-10 the absolute delay produced by the delay after layout

Figure 4-11 shows the differential delay produced by the delay after layout as VgN is swept and Vgp is set to VDD-Vgn

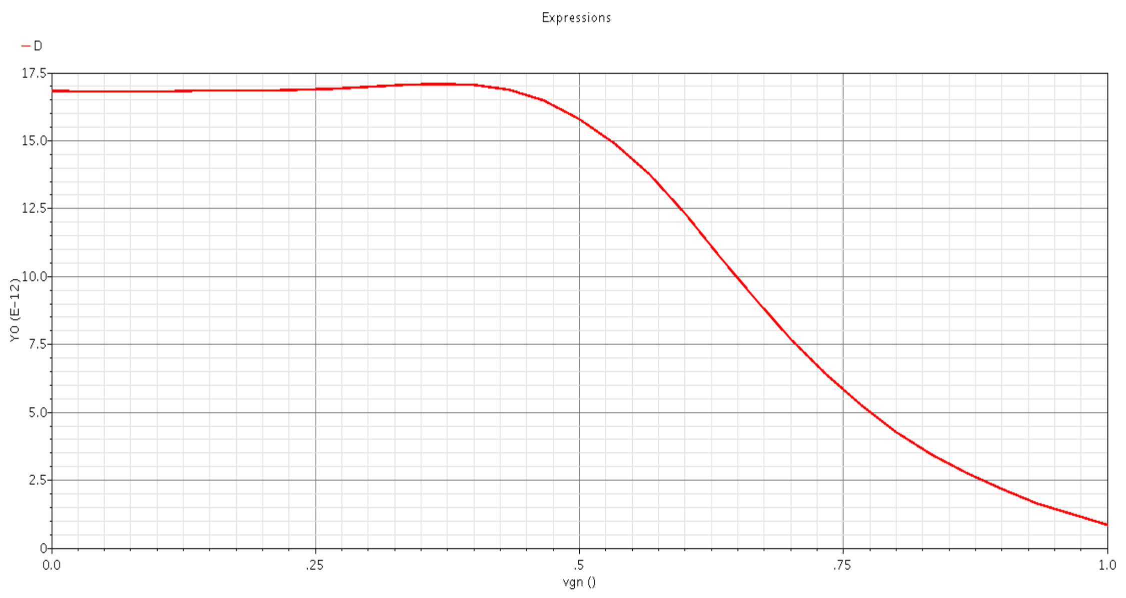


Figure 4-11 the differential delay produced by the delay after layout

## 4.3 TDC Buffer

### 4.3.1 TDC Buffer Structure

The second basic building block in the time to digital converter (TDC) is the buffer cell which acts as the connecting point between the delay cell and the flip-flop. The block of the buffer is also differential ended. One of them is used for the start signal coming out from the voltage controlled delay unit that is the one controlled by the bias sources Vgn and Vgp. The other one is used for the stop signal coming from the delay cell whose Vgn and Vgp are connected to VDD and GND respectively. Figure 4-12 shows the buffer cell of stop signal schematic

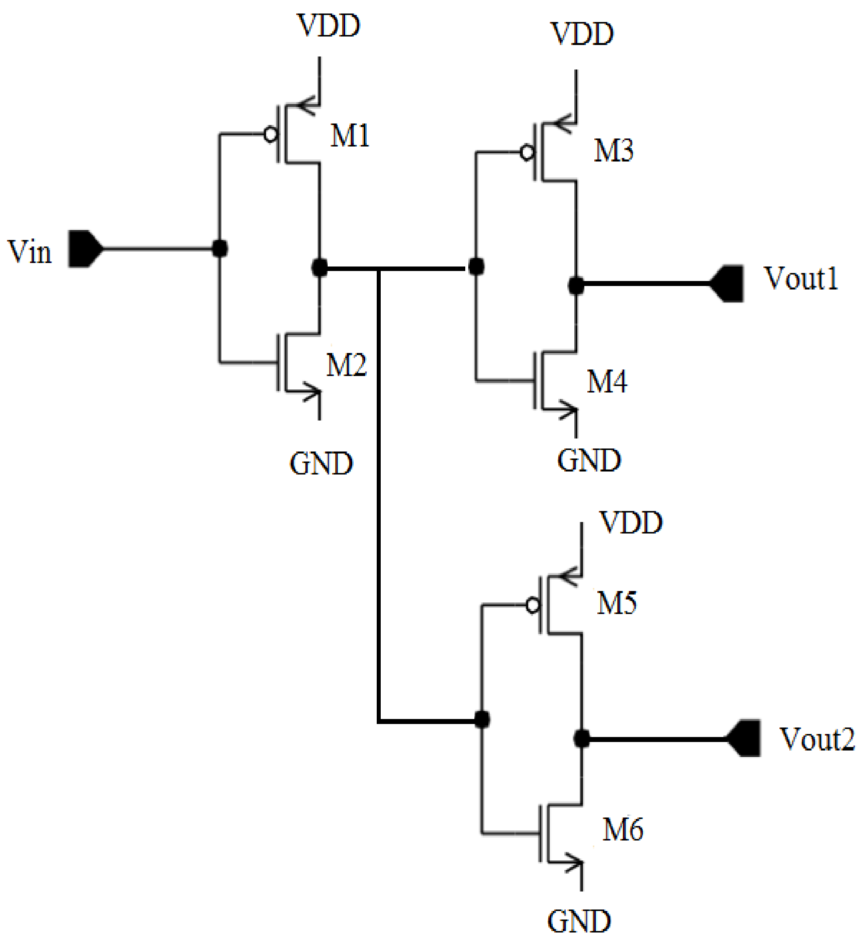


Figure 4-12 Buffer cell of stop signal schematic [2]

The main aim of using the buffer cell is to fasten the rising and falling edges of the input signal to the flip-flop. Due to the large load capacitance seen from the delay cell block due to the cascade stages of the delay chain, and the cascade stages of the flip-flops; this leads to degrade the fastness of the signal whether in the rising edge or in the falling edge of the input signal.

In order to solve this dilemma, it was a must to connect a buffer between the delay cell and the flip-flop to aid in sharpening the rising and falling edges of the signal entering the flip-flop.

Speaking about the buffer cell of the stop signal which is shown in figure 4-12. It consists of two cells of CMOS inverter placed in cascade. The extra inverter formed by the two devices M5 and M6 is used to deliver the signal to the re-clocking stage which is discussed in the next section.

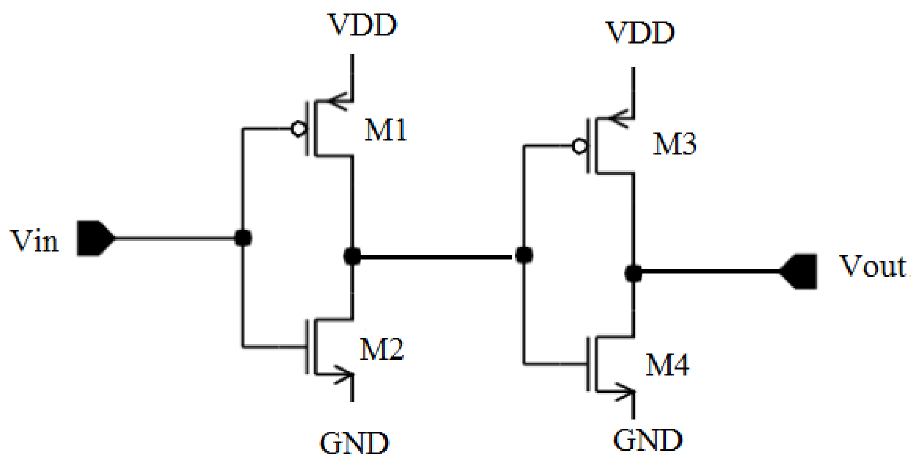


Figure 4-13 buffer cell of start signal schematic [2]

Considering the buffer cell of the start cell; it is the ordinary CMOS buffer which is formed from two inverters placed in cascade as shown in figure 4-13. The sizing of the devices in this block is large in order to make the output rising and falling edges fast.

### 4.3.2 TDC Buffer Layout

The completed layout for the buffer is shown in figure 4-14. The total area is9um × 4.325um.

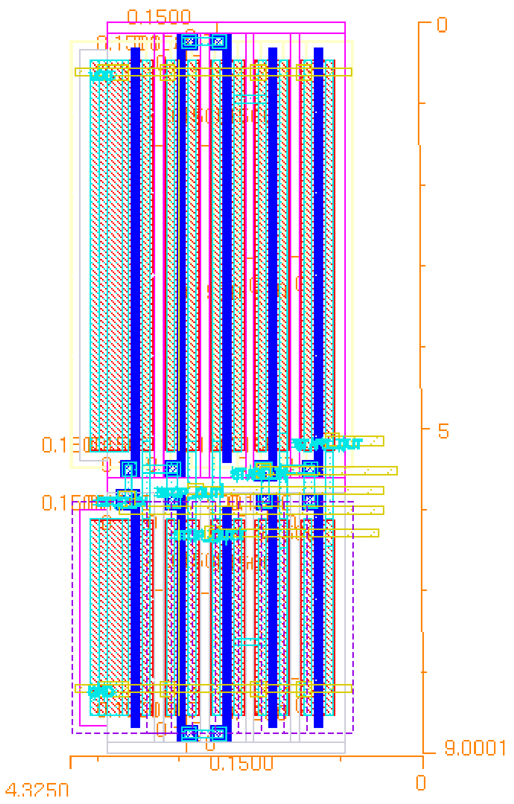


Figure 4-14 TDC buffer layout

## 4.4 TDC Decision Flip Flop

According to the delay cell outputs, we want a block to determine if the stop signal is preceding the start signal or not. If the stop signal was preceding the start signal, the output is one. If not, the output is zero. As described in the past section, each delay cell adds or subtracts a delay of 3.125ps. Accordingly, we have three critical states:

1. The two signals are coincident on each other. We want the output to be one.
2. The stop signal is preceding the start signal by 3.125ps. We want the output to be one)
3. The start signal is preceding the stop signal by 3.125ps. We want the output to be zero)

In the view of the flip-flop design, this is translated to a setup time of nearly -3ps, which is a tight restriction. We tried many types of flip-flops but they all failed to achieve the required performance like the SDFF and K-6, Dual Rail ETL. Finally, we reached this design in figure 4-15. It's somehow similar to the mC2MOS Latch but with many improvements.

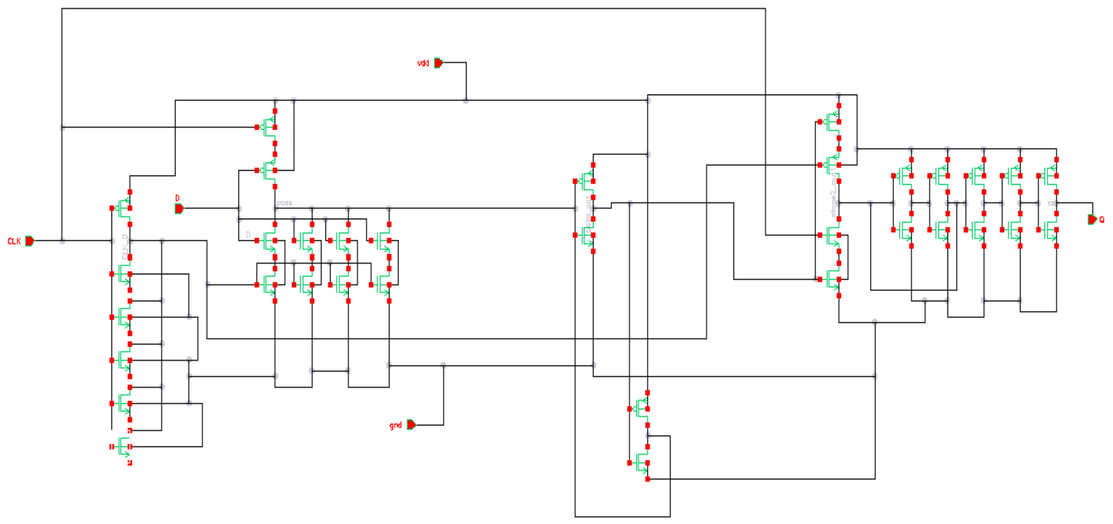


Figure 4-15 flip flop circuit diagram

### 4.4.1 Design Details

At first we need the inversion of the clock to be delayed by 3ps only to the clock signal which is considered a very fast inversion. And since we are interested only in the rising edge of the clock, we will enlarge the width of the NMOS transistor. However, the capacitance is exponentially increasing with the width of the transistor as in figure 4-16. Consequently we don't want to enlarge the width of the NMOS transistor above certain values to avoid the increase in the capacitance and so the delay of the output. The solution is to use several transistors in parallel to each other with a smaller size so the capacitance will increase linearly instead of exponentially. The same idea is used again in the design.

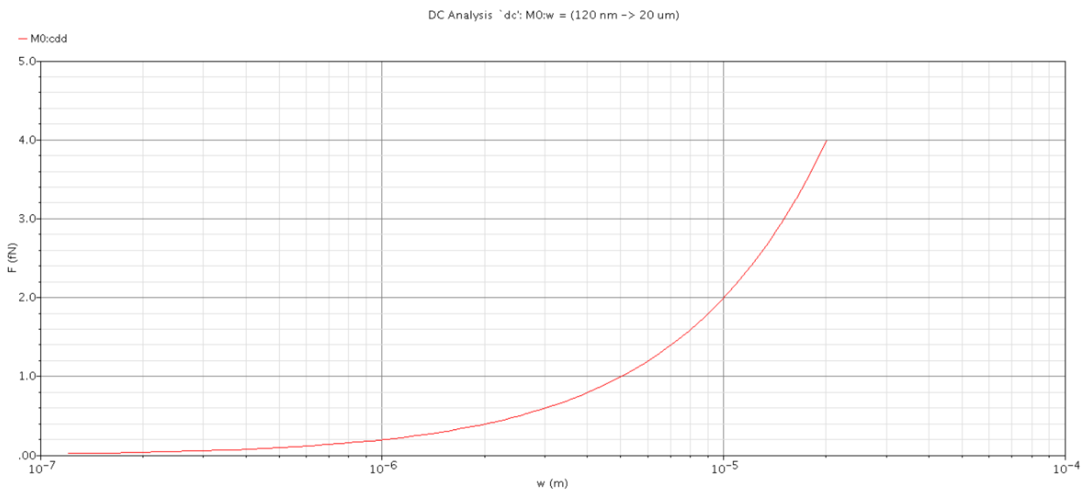


Figure 4-16 Cdd VS W

The idea of the design is based on capturing the value of the "D" input of the flip-flop during the negative half cycle of the clock till the critical time where the "CLK" and "D" are coincident on each other. In the positive half cycle, output is enabled presenting the last change occurred in "D" during the negative half cycle. A latch is added in the middle and at the end to keep the level of the signals especially in the critical cases demonstrated before.

The most challenging part of the design was catching any change in the input till the case of coincident inputs. We designed the flip-flop to achieve the cases we are required to face as in figure 4-17. For example we do not have to predict a "D" input changing from one to zero just before the "CLk" edge. Consequently, the transistors that we need them large are the NMOS transistors.

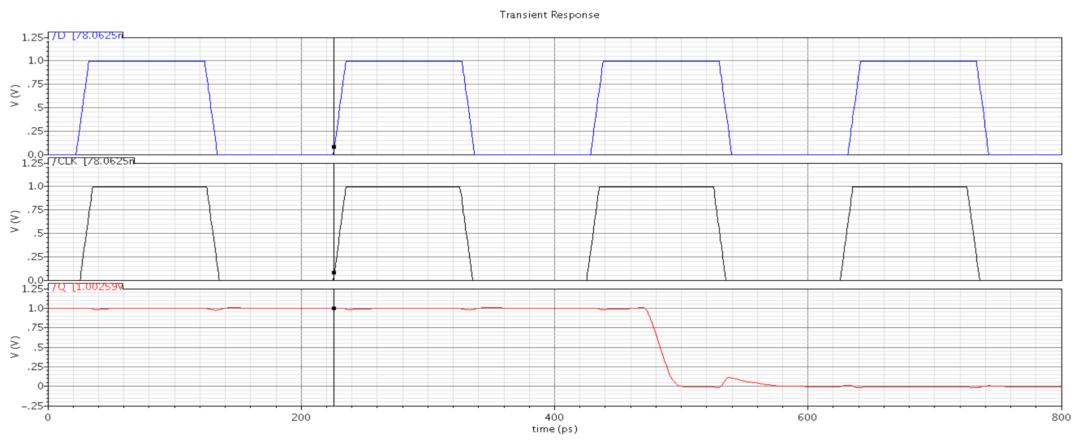


Figure 4-17 critical cases simulation results before layout

## 4.5 TDC Re-clocking and Pipelining

After designing the flip-flop block and before taking the output and connecting it to the encoder circuit; there was an essential block placed between those blocks known as the re-clocking circuit.

Due to the presence of delay line that is followed by stages of cascaded flip-flops; the output of the 1st flip-flop will not be at the same time of the 15th one so they have to be synchronized before encoded. More important point is that From simulations it is shown that the time delay between the first and last flip-flop outputs exceeds the 5GHz clock period of 200ps, directly encoding these outputs would result in errors

The main idea of the re-clocking is to make the outputs of all flip-flops become aligned on the same rising edge before they are being encoded by the encoder [2].

In order to implement the circuit we used two approaches; one of them is using chain of buffers to delay the signal, and the other is based on using chain of flip-flops.

### 4.5.1 Design Using Buffers

Speaking about the 1st approach it's based on CMOS inverter design that is used to form a buffer cell. As illustrated in figure 4-18 the first row is the original flip-flops of the TDC block and assuming that we have only 4 stages not fifteen for simplicity. The output of the 1st flip-flop to be aligned with the 4th flip-flop it will pass through 3 buffers to delay it by the same amount of time taken until the last signal come out from the 4th flip-flop and then their rising edges become aligned, similarly the output of the 2nd and 3rd flip flop.

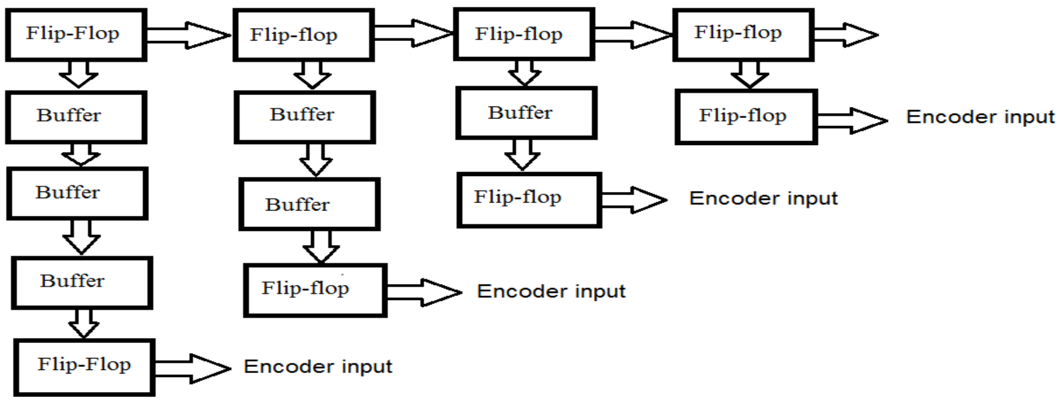


Figure 4-18 Re-clocking using buffer cells

As explained in figure 4-18 after all the signals path through the buffer chain they are connected again to the input of another flip-flops which are triggered by the same clock. In this case we can't use an external input from off chip to trigger these flip-flops because its phase would likely not be aligned with the data being re-clocked. In order to solve this dilemma the clock signal was taken from the end of the delay line and inverted in order to fit with the signals needed to be re-clocked because it will be subjected to the same process variations as the data. The clock signal is labeled "clkrsmp" in figure 4-2.

### 4.5.2 Design Using Flip-flops:

As mentioned before, we need the outputs of the decision flip flops to be ready together at the same time. However, this would not happen with this design. The delay between the output of the first flip-flop and the output of the last flip-flop is more than 200ps because the clock entering the 15th flip-flop is delayed by 15 delay stages more than the clock entering the first flip-flop. This means that when the output of the last flip-flop is ready, its corresponding correct output of the first-flop will be lost and replaced by another output corresponding to a new input as there is always a new input each 200ps. Consequently, we want to save the output of the flip-flops, synchronize them together and at the same time receive new inputs. This is done by a pipelined re-clocking system.

The idea is based on adding a series of flip-flops after the decision flip-flop to save the output of the flip- flop each 200ps. The ideal case is to save the output each 200ps and receive a new input correspondingly. However, we do not need to adjust the clock of the re-clocking flip-flop precisely on 200ps to avoid importing a new clock to the system and avoid the synchronization problems with the existing clock. The solution is that we can use the delayed versions of the clock to perform the re-clocking task. Fig 4-19 shows the delayed versions of clocks

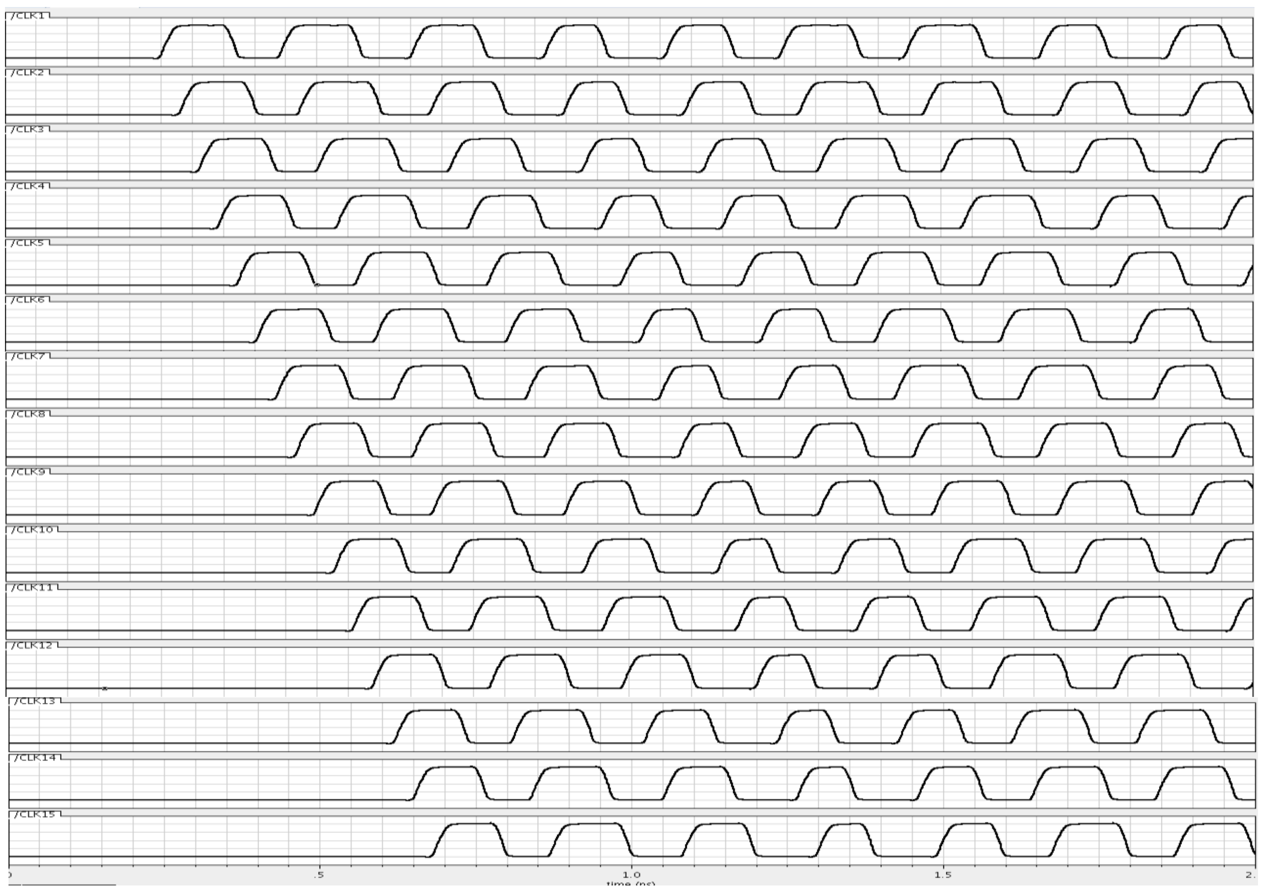


Figure 4-19 delayed versions of the stop signal in the delay cell

In the re-clocking part, the re-clocking clocks should be connected to several flip-flops beside the main decision flip-flops. This may result in an over loading effect. Consequently, the signals used in re-clocking will be generated from a separated output from the buffer.

Now we want to determine the number of re-clocking flip-flops after each decision flip-flop. To get a correct number, we should try on the worst case condition. This worst case condition happens when two consecutive edges entering the clock flip-flop are in a case where they are too close to each other.  As mentioned in the VTC section, the clocks are pulse modulated according to the input voltage. Consequently, this worst case condition would happen when entering the smallest possible value to the VTC followed by the largest possible value in the form of consecutive pulses. This would result in a series of consecutive clocks of narrow pulses then wide pulses.

In this case, the time between two consecutive rising edges would be the least. Here we can choose which clocks we should use in each stage of re-clocking to be in the safe side away from any errors.  Let’s take an example as in figure 4-20; we want to adjust the output of the first decision flip-flop.

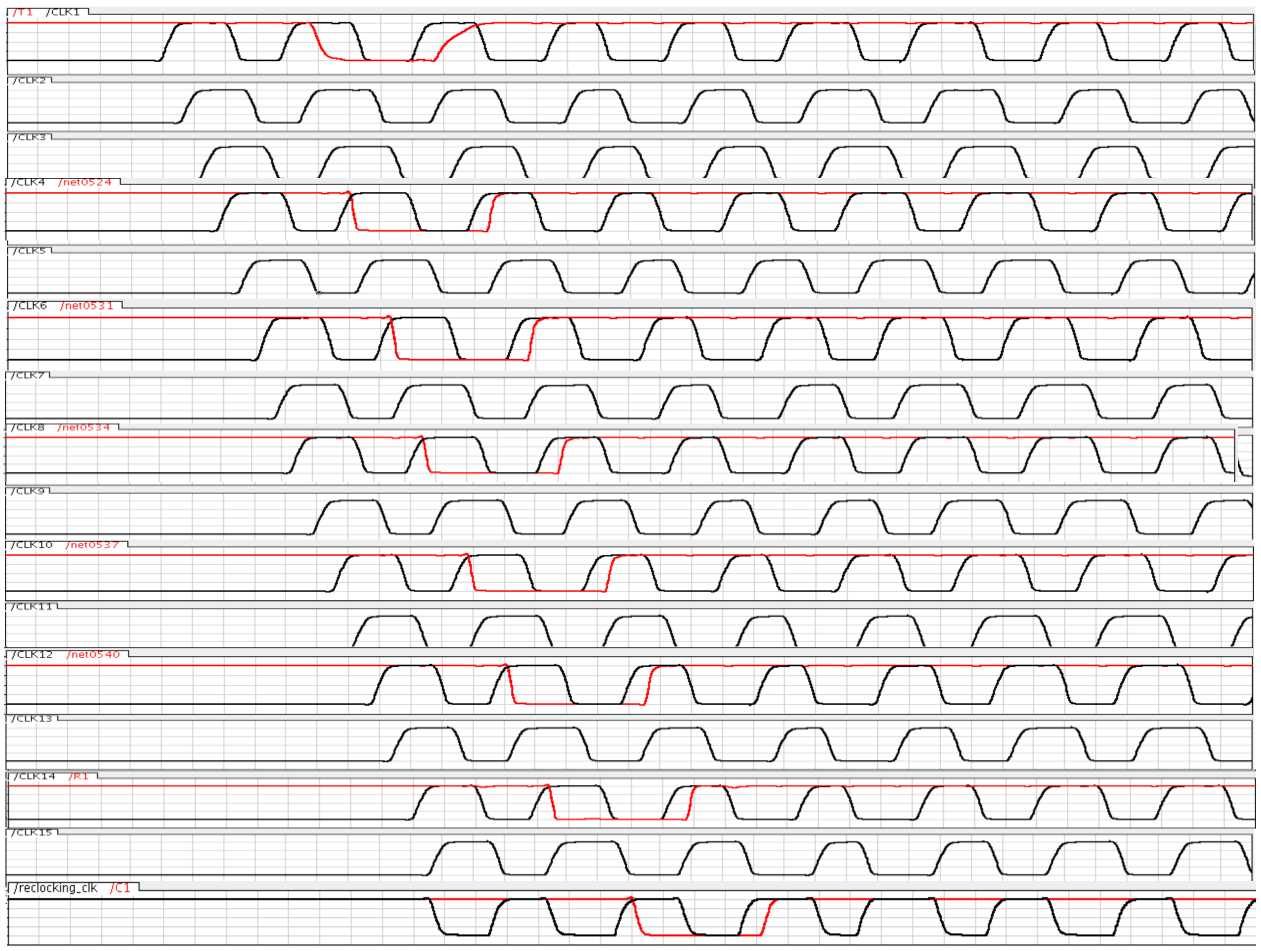


Figure 4-20 delayed versions of stop signal with decision flip flop outputs

As shown in figure 4-20, we will first draw all the 15 delayed versions of the stop signals in the delay cells. And then, we will draw the output of the first flip-flop on the first clock signal. Now we want to choose a delayed version of this clock to save this output. There are two conditions on the delay between the two chosen clock versions to correctly decide the number of flip-flops:

1) The rising edge of the delayed clock version should be delayed enough so that the output of the flip-flop is ready.

2) The rising edge of the delayed clock version should not be too far that it wouldn’t

come after the output of the decision flip-flop is changed. i.e: the delay between the rising edge of the main clock and the re-clocking clock should be less than 200ps. In the shown example I chose to re-clock using the 4th, 6th, 8th, 10th, 12th and the 14th stop signals from the delay cells. This means that 6 flip-flops are used to re-clock the first decision flip-flop outputs.

Similarly, the rest decision flip-flop outputs are re-clocked in the same way. At the end, the first flip-flop will have the largest number of flip-flops (6 flip-flops as in the example) and the number will decrease gradually till zero re-clocking flip-flops at the last decision flip-flop.

The number of flip-flops demonstrated in the example can be minimized greatly but it would be better to perform this minimization after the post layout simulations of the delay cell and decision flip-flops to get correct results.

After the demonstrated re-clocking stages, we want to register all the outputs entering the Encoder using the same clock and also all the outputs from the encoder. This clock must make the outputs ready at a rate of 5GS/sec precisely. We cannot use the rising edges of the delayed version of the clocks as we did before, it will not be true as the distance between these edges is varying according to the input voltage and we want a 200ps varied edges. Instead, we can use the falling edge in this final stage of re-clocking as the falling edge comes every 200ps precisely and is not pulse modulated like the rising edge. Consequently, we will use an inverter after the last delay cell to invert the stop signal and then, we will use this signal as the re-clocking signal.

Regarding the flip-flop used in re-clocking, it nearly doesn’t have any constraints like the decision flip-flop. As a result, we will use a simplified design of the decision flip-flop by removing all the unused transistors as shown in fig 4-21

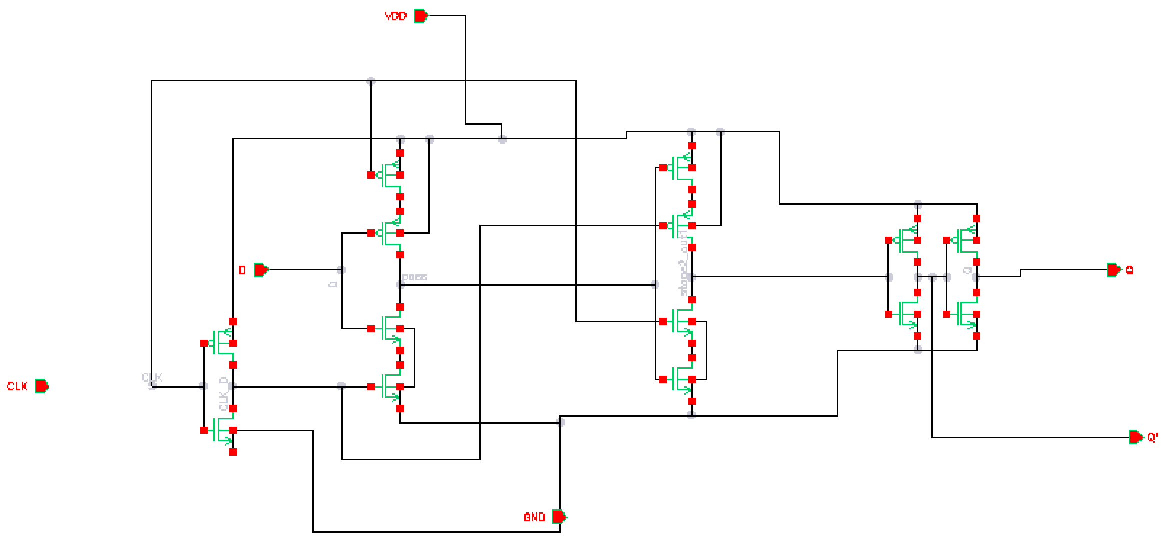


Figure 4-21 reclocking flip flop circuit diagram

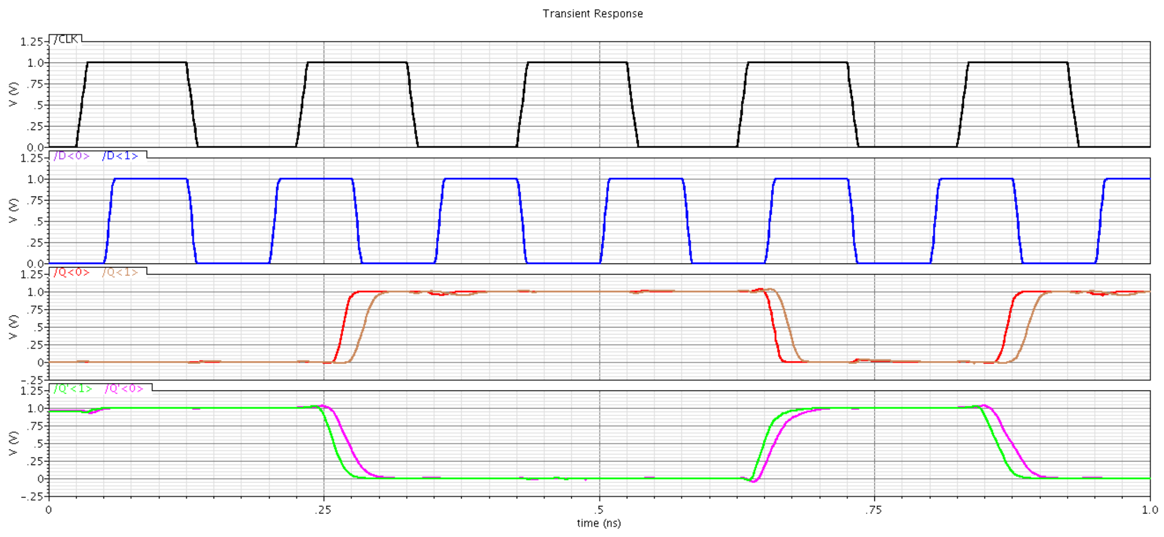


Figure 4-22 simulation results of the flip flop

Figure 4-22 shows the simulation results of the reclocking flip flop.

Re-clocking using flip-flops is better than re-clocking using buffers because any mismatch or error in the layout or the fabrication process may lead to different timings in the buffer results. However, these mismatches or errors in the flip-flops wouldn't affect the results as the case in the buffers because the output of each stage is observed at the delay cell clock edges only.

Let's take another example to figure out how is re-clocking done. For example, if the input was minimum then maximum, all the re-clocking outputs will change from zero to one and then from one to zero, Etc.

Figure 4-23 shows the reclocking of the output of the first decision flip-flop “T1” and the stages it passes by until it is out.

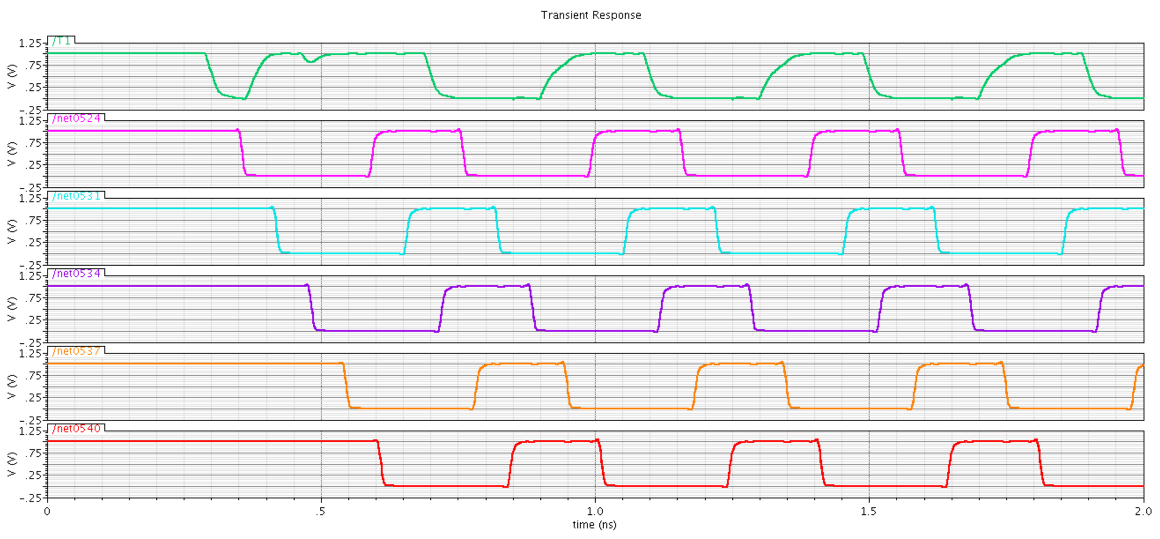


Figure 4-23 reclocking output stages

Figure 4-24 shows the output of the re-clocking circuit that are inputs to the decoder in this case.

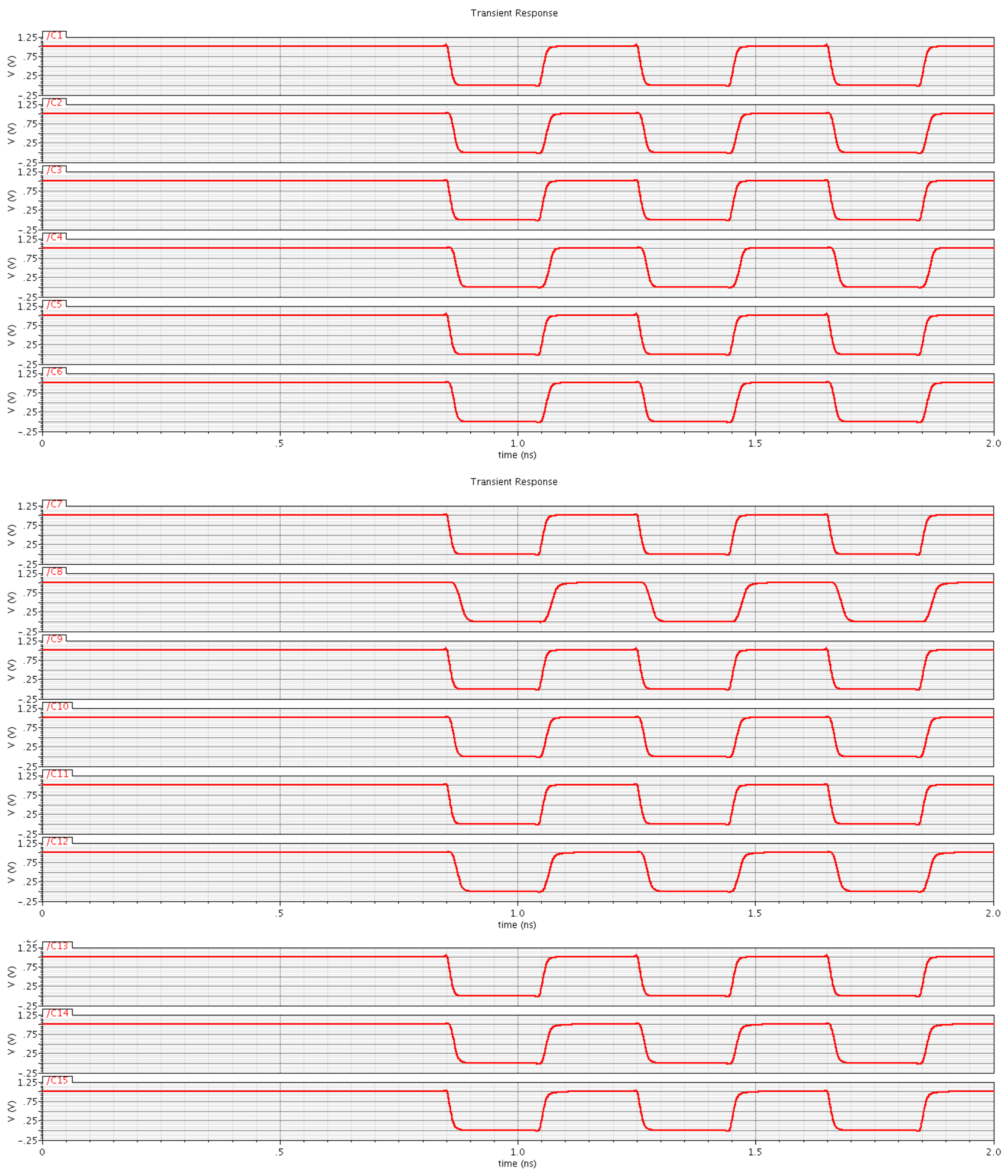


Figure 4-24 the output of the re-clocking circuit that are inputs to the decoder

## 4.6 TDC Encoder

The VDL output consists of 15 bit forming 16 possible combination, a series of n consecutive ones followed by (15-n) consecutive zeros are produced starting from 15 consecutive zeros then one followed by 14 consecutive zero and so on till 15 consecutive ones, the final block in the TDC is the thermometer encoder which converts the 15-bit thermometer-coded output of the VDL (T1 through T15) to a 4-bit binary output (B0 through B3).

According to the specific sequence described above the encoder in this case is a priority encoder in which the output bits depend on the position of the last '1' in the sequence as shown in the following table

| output of the VDL (T1 through T15) | binary output (B0 through B3) |
|---|---|
| 000000000000000 | 0000 |
| 100000000000000 | 1000 |
| 110000000000000 | 0100 |
| 111000000000000 | 1100 |
| 111100000000000 | 0010 |
| 111110000000000 | 1010 |
| 111111000000000 | 0110 |
| 111111100000000 | 1110 |
| 111111110000000 | 0001 |
| 111111111000000 | 1001 |
| 111111111100000 | 0101 |
| 111111111110000 | 1101 |
| 111111111111000 | 0011 |
| 111111111111100 | 1011 |
| 111111111111110 | 0111 |
| 111111111111111 | 1111 |

Table 1 15 to 4 priority encoder

### 4.6.1 TDC Encoder Architecture

Different architectures can be used for the encoder as ROM implementation or Tree implementation, however ROM are complex, area and power consuming and not fast enough for 5GS/s operation. Tree architecture is faster than ROMs. It also introduce uniform loading of the thermometer bits but requires high number of logic gates.

Using Karnaugh maps to directly map the thermometer-coded inputs to the binary outputs minimizes the amount of logic required, with comparable speed to the tree architecture. Using sum-of-product approach we can easily get the expression for each of the 4 output bits from the previous table while considering other compensations as a do not care state.

The expressions are as follows

$B0 = T15 + T13.\overline{T14} + T11.\overline{T12} + T9.\overline{T10} + T7.\overline{T8} + T5.\overline{T6} + T3.\overline{T4} + T1.\overline{T2}$

$B1 = T14 + T10.\overline{T12} + T6.\overline{T8} + T2.\overline{T4}$

$B2 = T12 + T4.\overline{T8}$

$B3 = T8$

This architecture requires a total of 11 2-input AND gates and 11 2-input OR gates. However the OR gate has bad performance at 5GS/s because of the lower mobility of holes as compared to electrons, The OR gate has two series PMOS (M1 and M2) in the pull-up network (PUN) as shown in the figure below. PMOS device M1 pull the output node up to VDD through the resistance of M2. Increasing the width of the device increased the drive current but also increased the parasitic capacitance at the gate and drain of M1, resulting in no net increase in switching speed. In the AND gate, PMOS transistors of the PUN are connected in parallel so each transistor switch quickly on its own [2] .figure 4-25 shows the Schematics for (a) AND gate and (b) OR gate

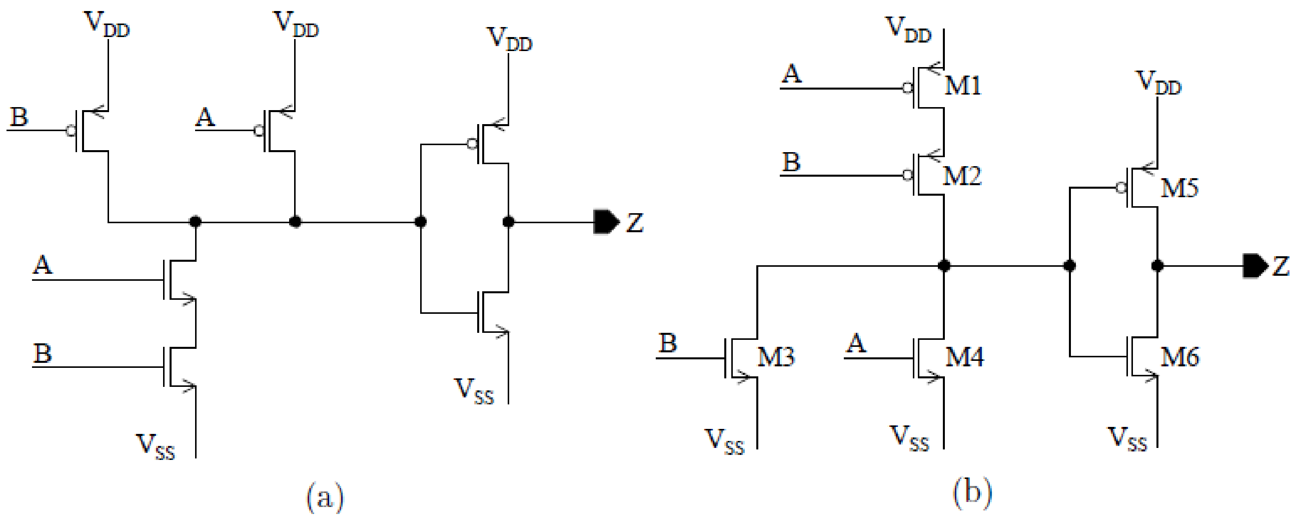


Figure 4-25 Schematics for (a) AND gate and (b) OR gate [2]

Because of the above problem, the encoder has to be designed using only AND logic. The conversion from OR gates to NAND gates was done using boolean algebra. Figure 4-26 below shows this conversion.

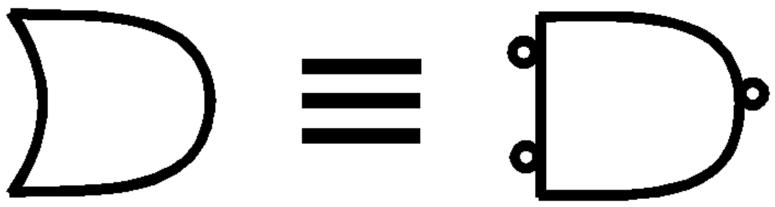


Figure 4-26 the conversion from OR gate to NAND gate

Thus the encoder is designed using ANDs and NANDs only. No additional inverters are needed to invert the thermometer outputs since the flip-flops produce both Q and $\overline{Q}$ (differential outputs).the Schematic for NAND gate is shown if figure 4-27

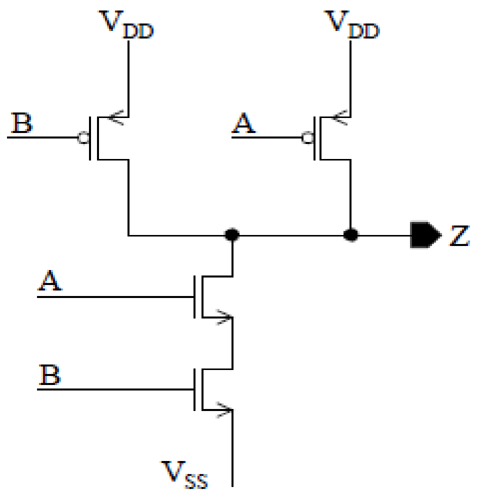


Figure 4-27 Schematic for NAND gate [2]

An observation can be made about this logic that there are no logic gates between the inputs and B3, while there are 4 gates between the inputs and B0. This results in a significant timing mismatch at 5GS/s. so, dummy logic gates have to be inserted so that each output has 4 gates between the input and output to have the same propagation delay in all outputs and avoid this mismatching. This implementation requires 15 AND gates and 17 OR gates. The final implementation is shown in the figure 4-28

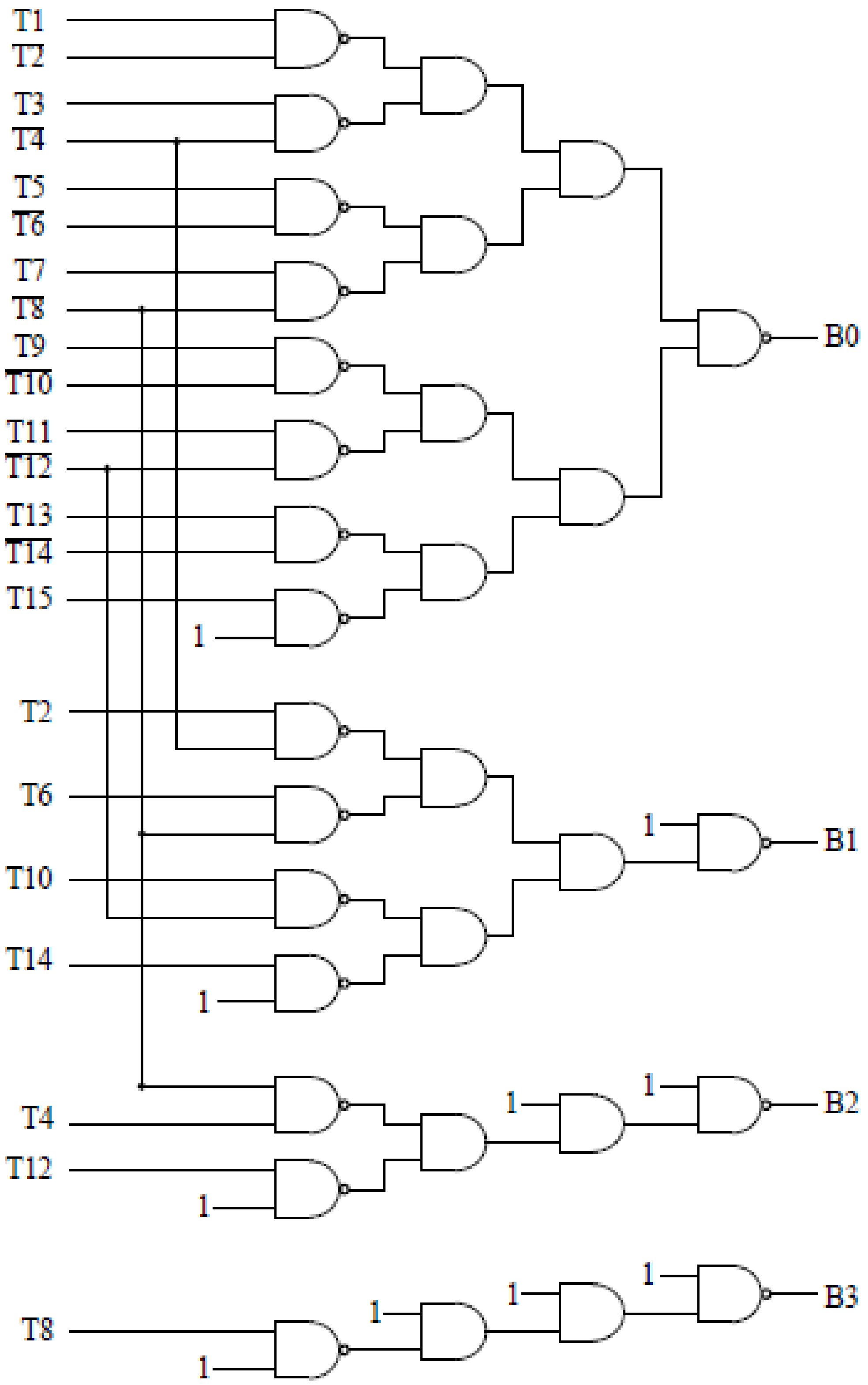


Figure 4-28 Final Minimum-Logic encoder Design [2]

The total delay of the encoder should be less than 200ps so it can encode outputs each cycle ,so the individual gate should be fast enough to withstand this speed,

As mentioned before the propagation delay decreases by increasing the sizing of transistor which lead to increase the current driven but at certain size the increase in the load capacitance due to increasing size starts to be significant and no net decrease in the propagation delay is achieved by increasing sizing above this point.

By sweeping on the sizing of the transistors while measuring the propagation delay the ratio $\frac{W}{L}$ is chosen to be 4um in nmos and 8um in pmos

Figure 4-29 shows the output of the encoder due to ramp input, the output starts from all zeros and increases monotonically till all ones

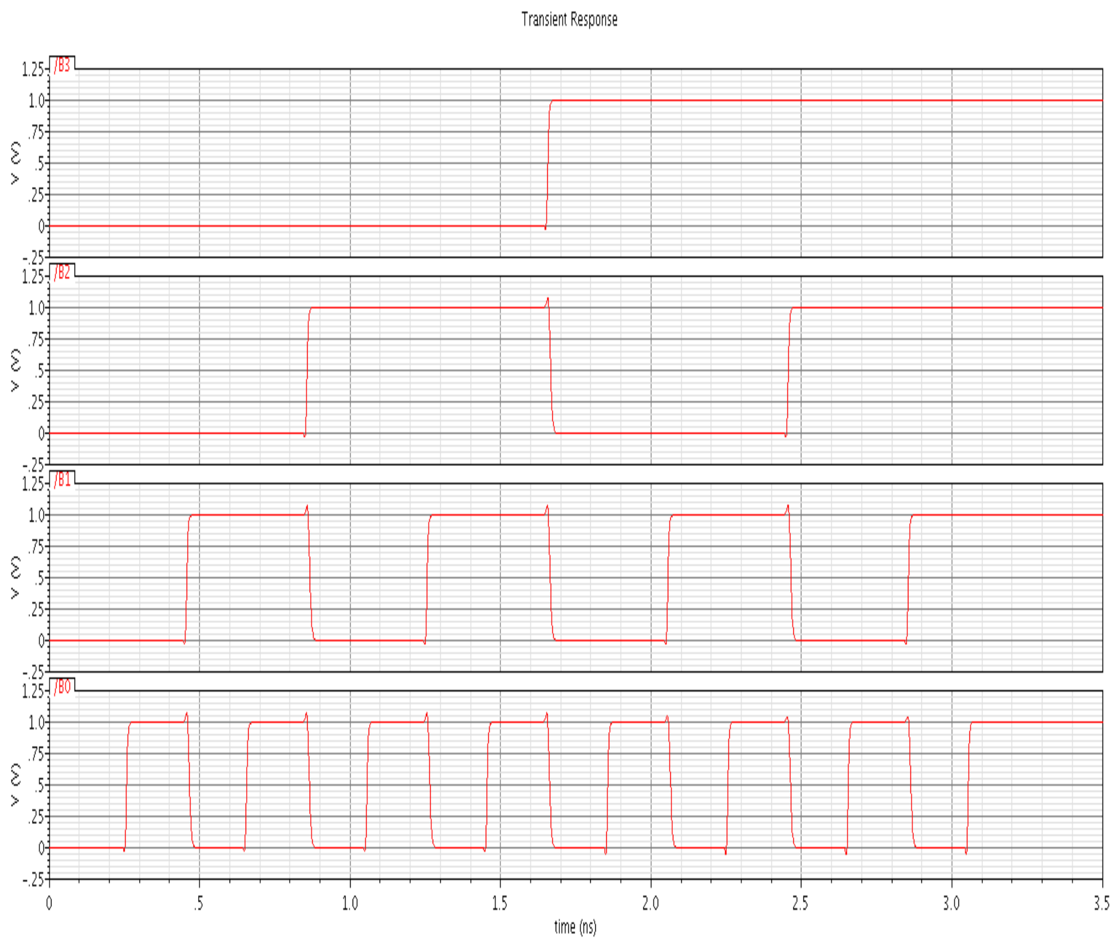


Figure 4-29 the output of the encoder due to ramp input

### 4.6.2 TDC Encoder Layout

The completed layout for the encoder is shown in figure 4-30. The total area is 96.8um × 7.3um

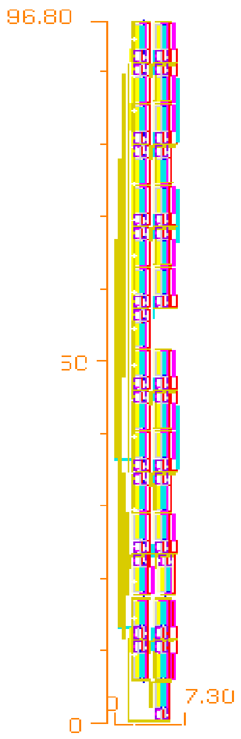


Figure 4-30 the layout of the encoder

Figure 4-31 shows the post layout results of the encoder due to ramp input

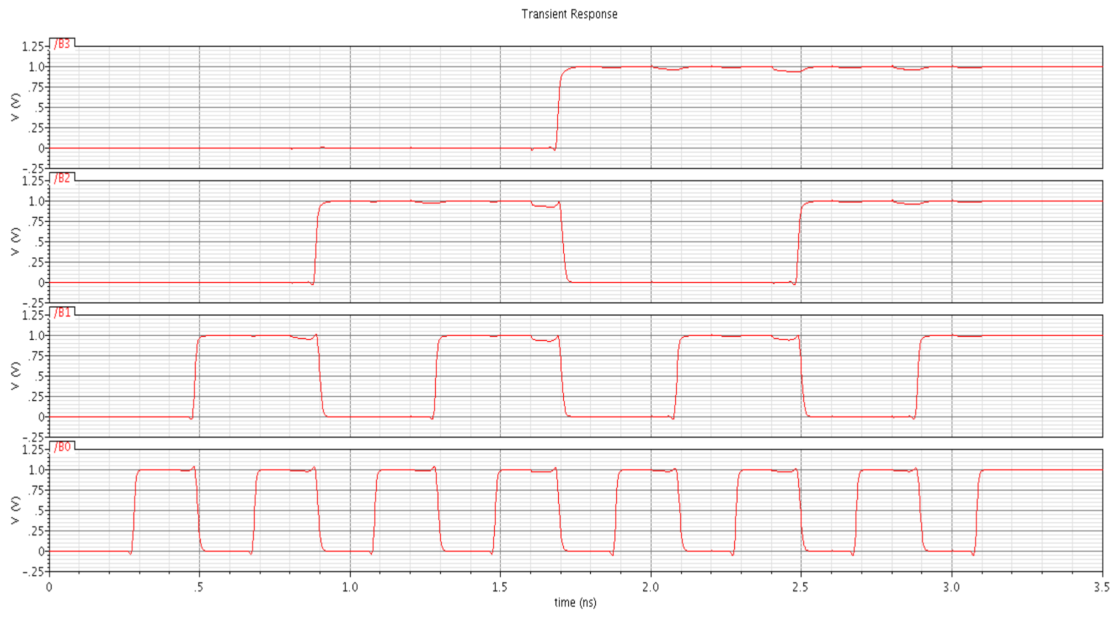


Figure 4-31post layout results of the encoder due to ramp input

# Chapter 5: ADC Simulation Results &Performance

To achieve the requirements of our time based ADC we have used the 5 Giga sample/s VTC block with dynamic range =140 mV which we have illustrated in chapter 3 and we have used the TDC block illustrated in chapter 4.

In this chapter we will present the simulation results and characteristics of our time based ADC.

## 5.1 Checking ADC Functionality

To check if the ADC works or not we will apply sin wave as input with reasonable frequency (for example 502 MHz) and convert the output to analog using DAC and check the output fit with the input sin wave or not. Figure 5-1 show the output if no sample and hold exists while figure 5-2 shows the output in presence of sample and hold circuit.

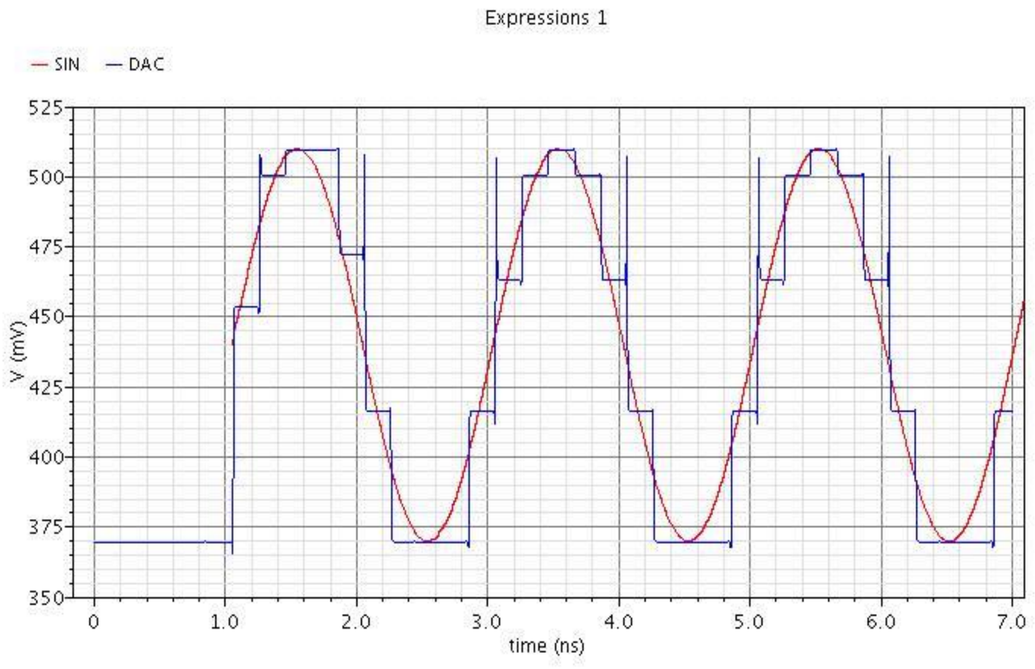


Figure 5-1output due to sin wave input without using sample and hold circuit

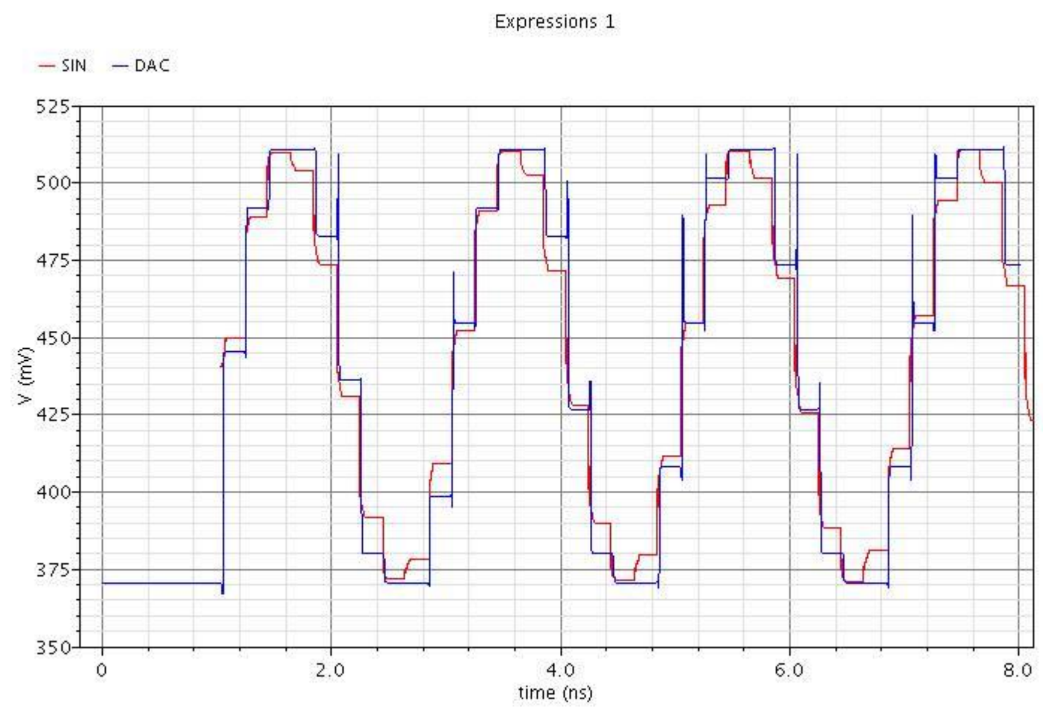


Figure 5-2 output due to sin wave input using sample and hold circuit

## 5.2 ADC Output Codes

Figure 5-3 shows the output codes due to different input voltages

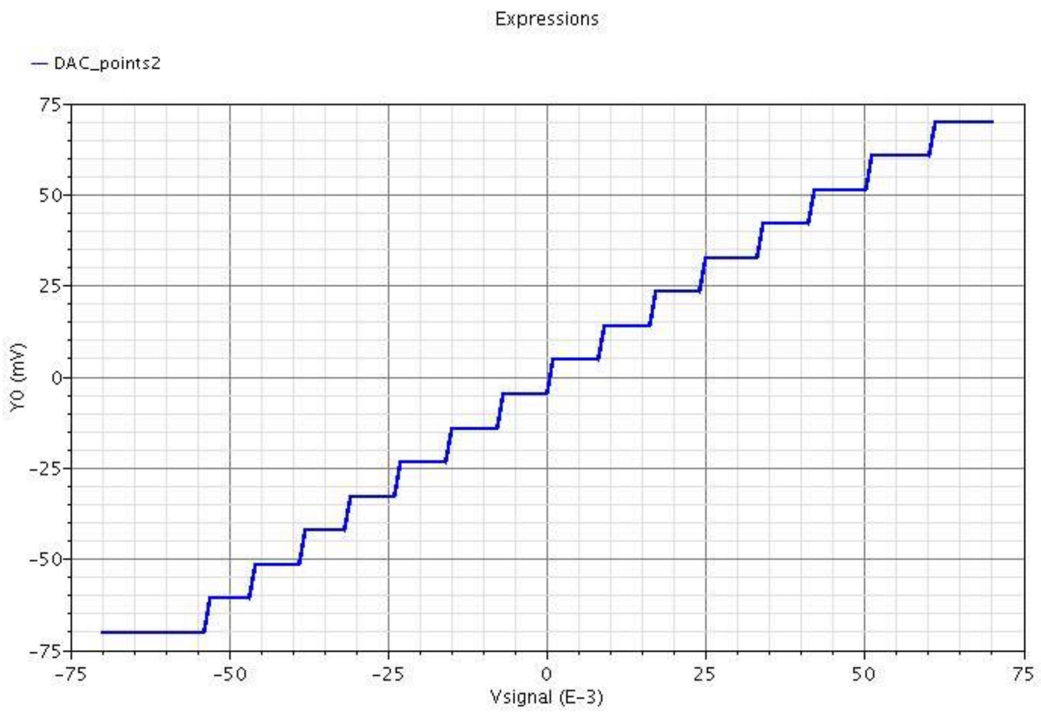


Figure 5-3 output codes VS input voltage

## 5.3 Effective Number of Bits ENOB

We calculated the ENOB at 3 frequencies with and without the sample and hold circuit.

### 5.3.1 Low Frequency 502 MHz

Figure 5-4 shows the frequency domain of the output due to input frequency =502 MHz without sample and hold and figure 5-5 shows the frequency domain of the output due to input frequency =502 MHz with sample and hold circuit.

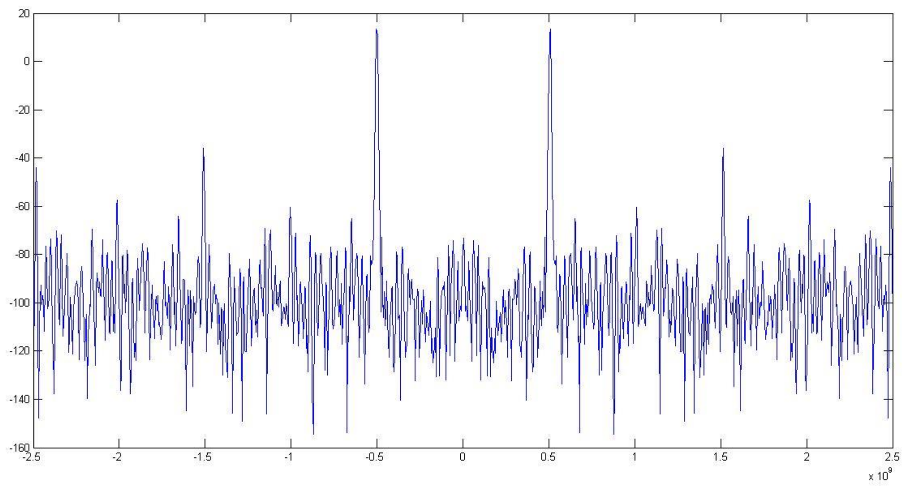


Figure 5-4 the frequency domain of the output due to input frequency =502 MHz without sample and hold circuit

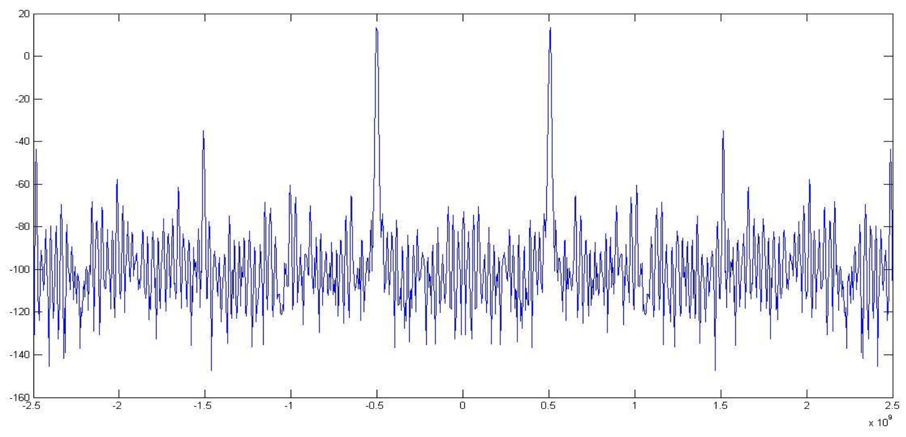


Figure 5-5 the frequency domain of the output due to input frequency =502 MHz with sample and hold circuit

ENOB without sample and hold circuit=3.25

ENOB using sample and hold circuit= 3.23

### 5.3.2 High Frequency 2457 MHz

Figure 5-6 shows the frequency domain of the output due to input frequency =2457 MHz without sample and hold and figure 5-7 shows the frequency domain of the output due to input frequency =2457 MHz with sample and hold circuit.

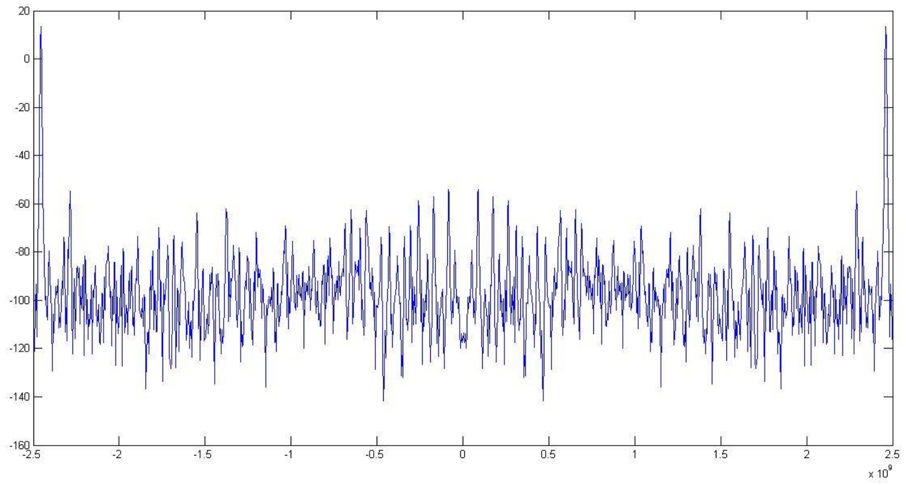


Figure 5-6 the frequency domain of the output due to input frequency =2457 MHz without sample and hold circuit

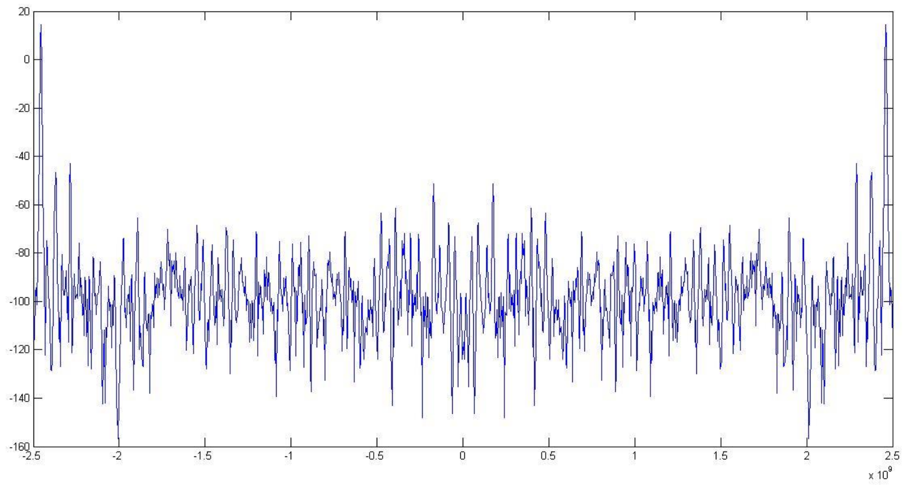


Figure 5-7 the frequency domain of the output due to input frequency =2457 MHz with sample and hold circuit

ENOB without sample and hold circuit=3.62

ENOB using sample and hold circuit= 3.53

### 5.3.3 Practical High Frequency 1098 MHz

Figure 5-8 shows the frequency domain of the output due to input frequency =1098 MHz with sample and hold circuit

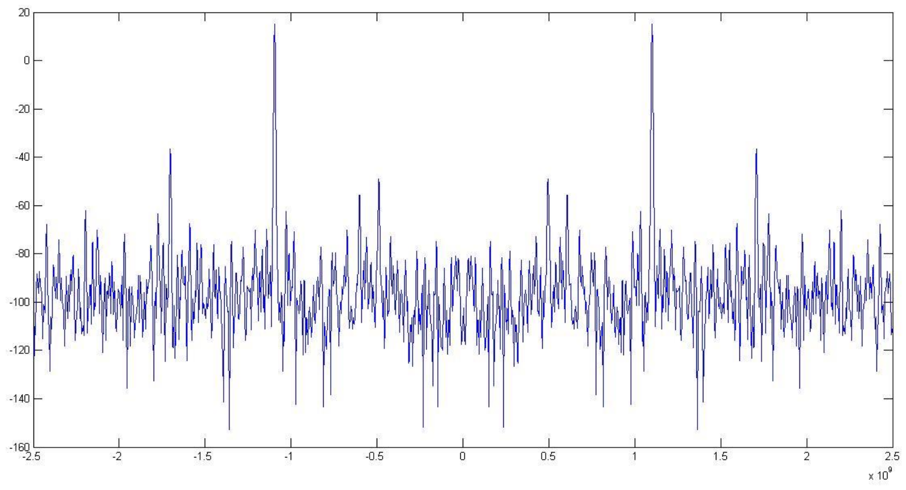


Figure 5-8 frequency domain of the output due to input frequency =1098 MHz with sample and hold circuit

ENOB using sample and hold circuit= 3.34

## 5.4 Layout and Simulated Power Consumption

As mentioned in previous chapters the layout of all blocks is done and tested using post-layout simulations .The completed layout for the T-ADC is shown in figure 5-9. The total active chip area is 68.3um x121.5 um. The simulated power consumption of the full TDC running at 5GS/s, is 13.34mw with a 1V supply.

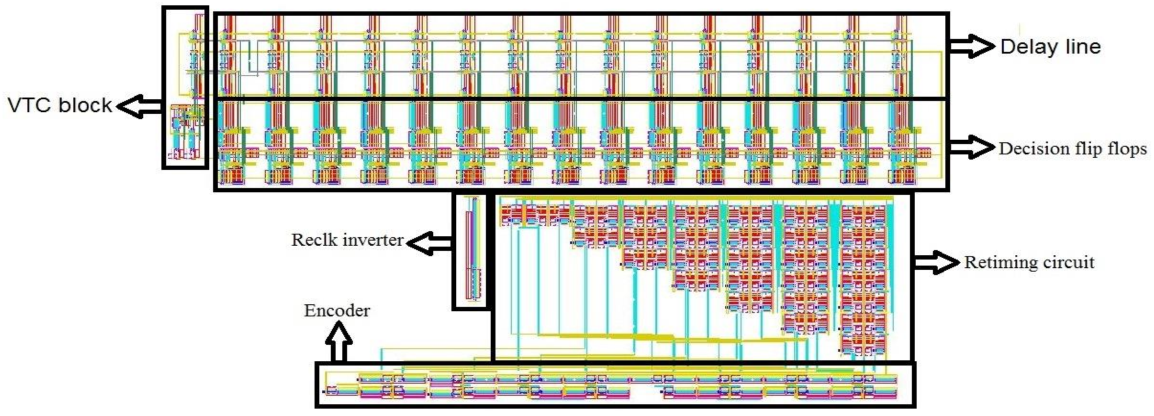


Figure 5-9 full layout of T-ADC

# Chapter 6: Conclusion and Future Work

In this thesis, we reviewed the concept of time-based analog-to-digital conversion using a VTC and a TDC block which employs a completely different technique from the conventional ADC and quantizes time at predefined amplitude intervals.

The main product of this work is a 4-bit, 5GS/s ADC and its layout and it should be fabricated afterwards under the 65nm technology. At the maximum input frequency of 2457, 1098 and 502 MHz, the ENOB is 3.6, 3.3 and 3.2 respectively.

This work has made several important contributions, including:

- This ADC uses the fastest achieved sampling speed (5GS/s) with this number of bits.
- Improving the re-clocking techniques used in other designs
- A tunable TDC against process variation with 3.125ps resolution
- A VTC with speed up to 6.6Gs/s , dynamic range 140mv and delay 50ps

## 6.1 Future Work

There is plenty of more work to be done to improve the performance of ADC. Some suggestions will be offered for future research stemming from the presented work.

### 6.1.1 Modifying the Layout

Since the performance of circuit after layout is not identical with schematic, the size of transistors could be modified for layout oriented. To reduce the parasitic effects, layout should be improved from a better floor plan.

### 6.1.2 Increase the ADC Speed to 6.6GS/s

As mentioned before the VTC sampling speed is up to 6.6GS/s so we can increase the system sampling speed by making some modifications to the TDC core.

### 6.1.3 Adding an On-chip TDC Automatic Calibration Algorithm

An automatic calibration algorithm tunes the delay tuning circuits. It would be possible to integrate this logic on-chip using a finite state machine. Instead using a code running on a PC to calibrate the chip.

The idea of calibration is simple. It is mainly based on testing a large number of samples and calculating the ENOB. Based on these samples, if the ENOB was small, we can count how many times each code was generated. Each code must be generated a certain number of times. By comparing the output results with the pre-saved results, we can calibrate the TDC through the voltage sources of the delay cells until maintaining an acceptable ENOB.

The merit of this way of calibration is that we don't need to calibrate the VTC, as this way of TDC calibration calibrates the VTC also in return.

### 6.1.4 Minimizing the Number of Flip-flops in Re-clocking

As proposed in the re-clocking part we can minimize the number of flip-flops greatly. For example instead of using six re-clocking flip-flops after the first decision flip-flop, we can only use two or three.

There are two ways of minimizing the number of flip-flops:

1) Using more delayed versions of clocks in re-clocking as proposed before. For example, as shown in figure 6-1 using clock 6 instead of clock 4 for the first re-clocking flip flop after the first decision flip-flop instead of clock 4.

2) Using the same clock of decision flip-flop in the re-clocking flip-flop:

This means that each re-clocking stage will be of a 200ps delay. For example, using clock 1 in all the re-clocking stage if the first decision flip-flop.

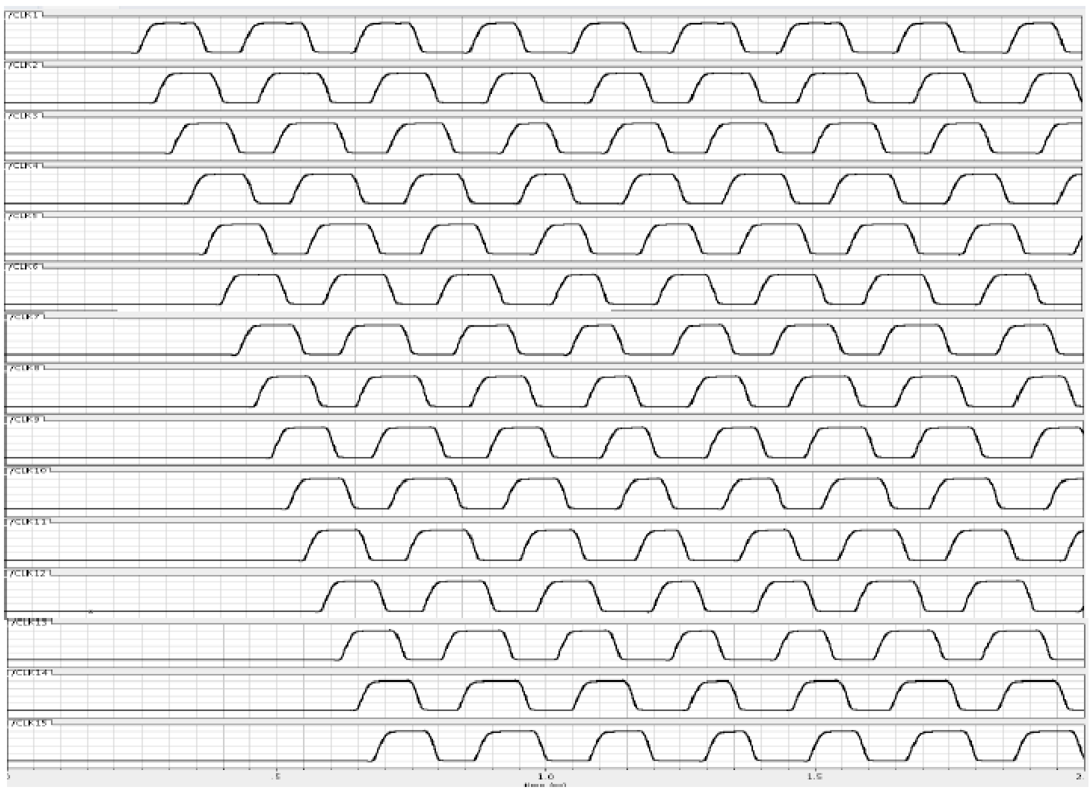


Figure 6-1 delayed versions of the clock signal

### 6.1.5 Optimize the Power Consumptions

In our design there is no limitation on power consumption. We can modify our blocks to maintain a certain level of power consumption

# References

[1] Mohamed Amin, *Design of a Time Based Analog to Digital Converter thesis*, Waterloo, Ontario, Canada, 2009

[2] Andrew Robert Macpherson, *A Time-Based 5GS/s CMOS Analog-to-Digital thesis*, CALGARY ALBERTA MAY, 2013

[3] “IEEE Standard for Terminology and Test Methods for Analog-to-Digital Converters,” 2011.

[4] Shahrzad Naraghi, *Time-Based Analog to Digital Converters dissertation*, Michigan, 2009

[5] Stephan Henzler, *Time-to-Digital Converters, Springer Series in Advanced Microelectronics,* 2010

[6] R. B. Staszewski, S. Vernulapalli, P.Vallur, et al., “1.3V 20ps Time-to-Digital Converter for Frequency Synthesis in 90-nm CMOS”, *IEEE Transactions on Circuits and Systems II, Vol. 53, No. 3, 2006, pp. 220-224*

[7]C. Lin and M. Syrzycki, "Single-Stage Vernier Time-to-Digital Converter with Sub-Gate Delay Time Resolution", *Circuits and Systems, Vol. 2 No. 4, 2011, pp. 365-371.*

[8] Andrew R. Macpherson, James W. Haslett and Leonid Belostotski, “A 5GS/s 4-bit Time-Based Single-Channel CMOS ADC for Radio Astronomy”, *Custom Integrated Circuits Conference (CICC), 2013 IEEE*

# Appendix

## Layout Tutorial

We are going to illustrate step by step how to implement the layout of a given circuit for example CMOS inverter using "tsmc65nm" technology and how to check the functionality of the circuit using DRC,LVS and PEX simulations.

### Building the layout

The first step is to open the Virtuoso window and select a certain schematic file that is implemented earlier or draw a new one. The input/output pins must be higher case as shown in figure 1.

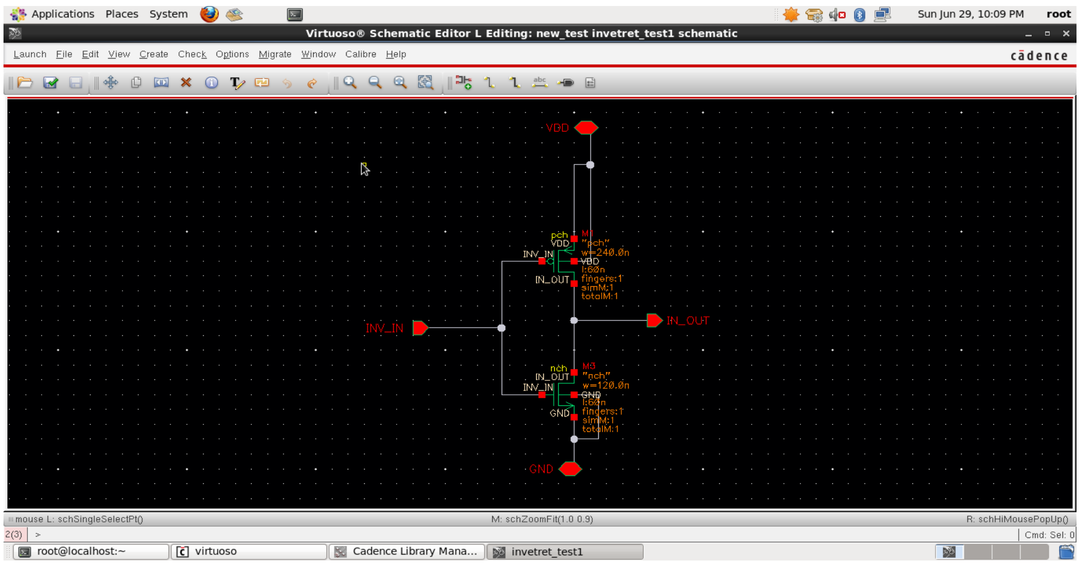


Figure 1: Circuit schematic

After finishing the design of the circuit, press check and save first to ensure that there are errors in the connections. In order to open the layout window go to the menu bar and hit "Launch>>Layout XL" as shown in figure 2.

A dialogue box will appear asking if you want to create a new layout for this design or you have already implemented it before and you just want to modify in it as shown in figure 3. If this is your first time to draw the layout then check the button "Create New", if not press "Open Existing" and press "OK".

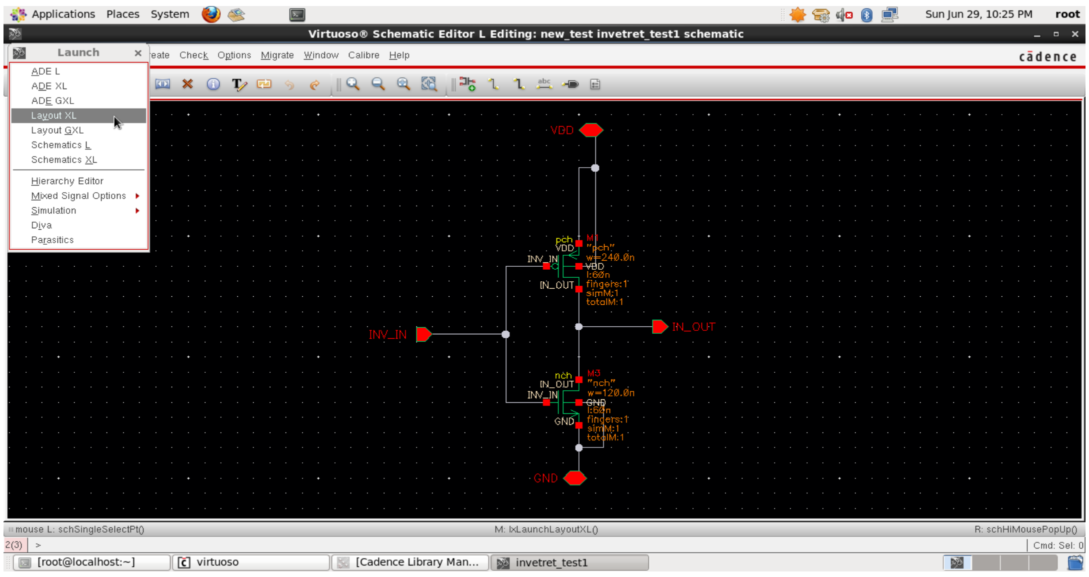
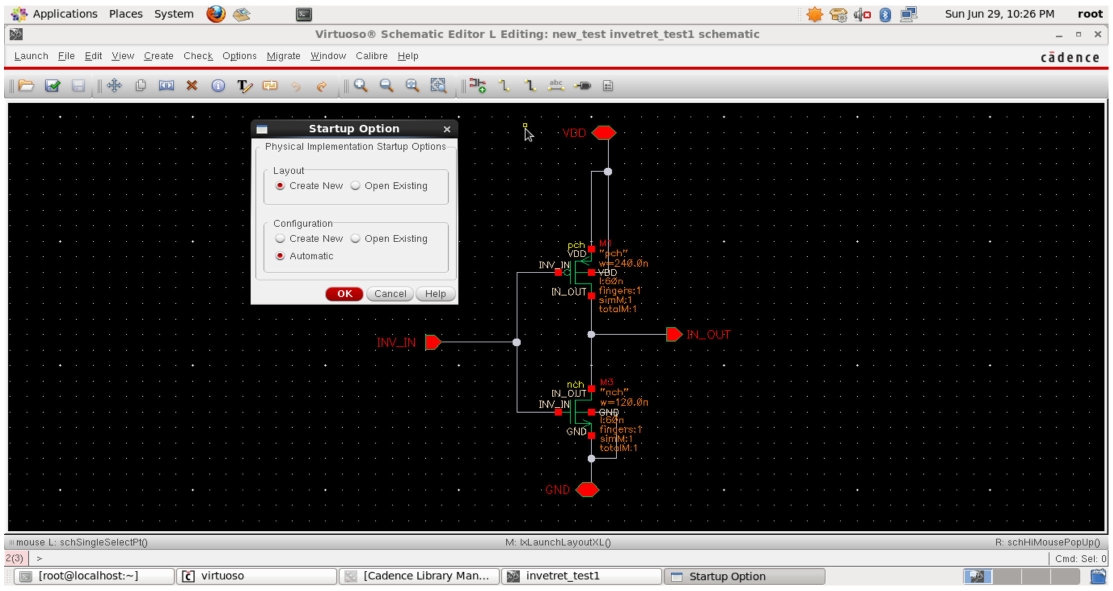


Figure 2: The launch window

Once you press "OK" another dialogue box will appear as illustrated in figure 4. Take care that the name of the layout file must be the same name of the schematic file and press "OK".


Startup Option

Physical Implementation Startup Options

Layout: [✓ Create New] Open Existing

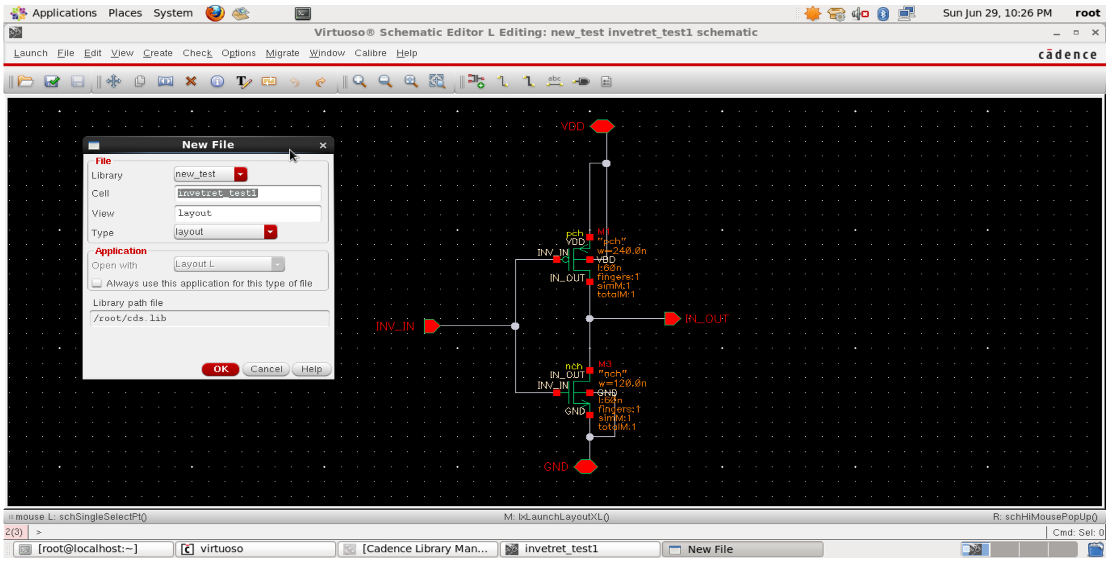
Configuration: Create New Open Existing [✓ Automatic]

OK
Cancel
Help
Applications Places System
Sun Jun 29, 10:26 PM root
Virtuoso® Schematic Editor L Editing: new_test invetret_test1 schematic
Launch File Edit View Create Check Options Migrate Window Calibre Help
cādence
VDD
INV_IN
IN_OUT
GND
pch
nch
mouse L: schSingleSelectPt()
M: lxLaunchLayoutXL()
R: schHiMousePopUp()
Cmd: Sel: 0
[root@localhost:~]
virtuoso
[Cadence Library Man...
invetret_test1
Startup Option

Figure 3: Startup option window

After closing this window; the layout window will appear joined by another window entitled "LSW" which contains the palette of metals, poly-silicon, thin-oxide, N-select, P-select…..etc.

Metals in "tsmc65nm" technology are named as "M1" for metal 1, "M2" for metal 2 and so on. Poly-silicon is named as "PO", the active (Thin-oxide) is named as "OD", the N-select is "NP" and similarly the P-select is "PP".


Virtuoso® Schematic Editor L Editing: new_test invetret_test1 schematic
Launch File Edit View Create Check Options Migrate Window Calibre Help
New File
OK
Cancel
Help
VDD
GND
INV_IN
IN_OUT
mouse L: schSingleSelectPt()
M: lxLaunchLayoutXL()
R: schHiMousePopUp()

File
Library: new_test
Cell: invetret_test1
View: layout
Type: layout

Application
Open with: Layout L
Always use this application for this type of file

Library path file
/root/cds.lib

Figure 4: The new file window

In order to bring the transistors automatically in the layout window go to the menu bar and press "Connectivity>>Generate>>All from sources" If this is the first time to create the layout as mentioned in Fig 5. But if the layout file exists already and you only want to modify something press "Connectivity>>Update>>Components and nets".

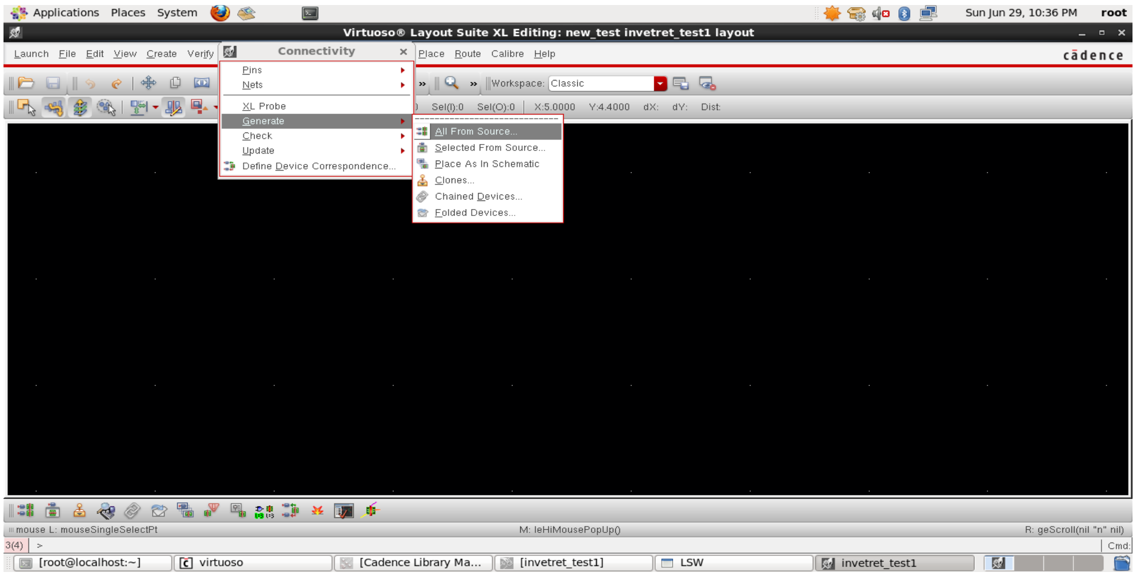
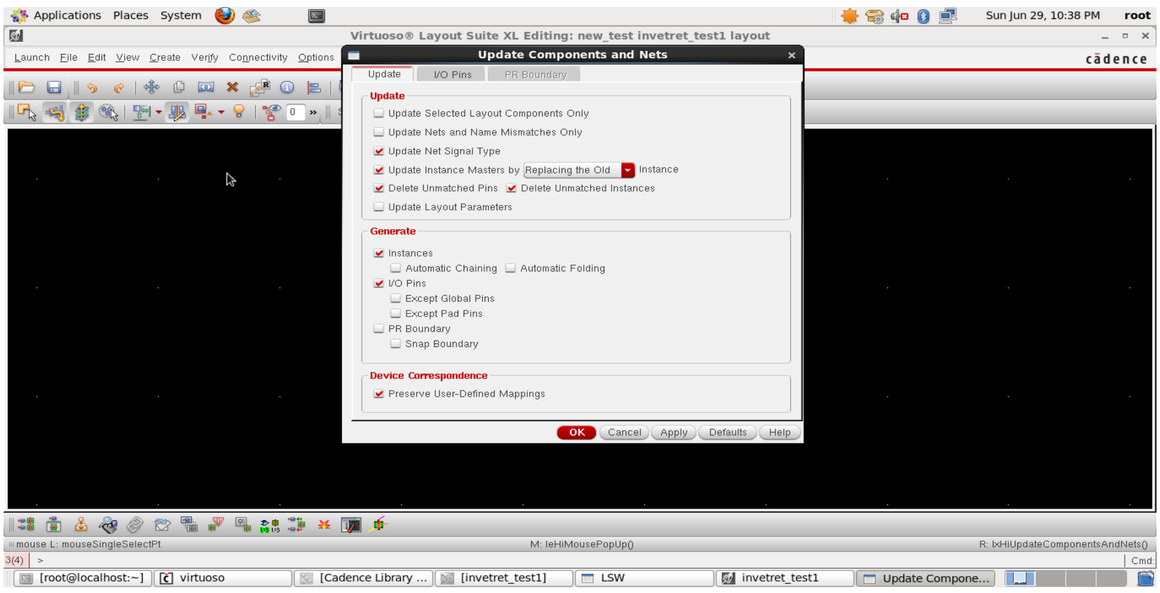


Figure 5: The connectivity window

For both cases the same dialogue box will appear like the one shown in figure 6. Considering the generate part, there are 3 check boxes "Instances" which is responsible for the placement of the transistors; "IO pins" which is responsible for placing the I/O pins; and "PR boundaries" which is responsible for placing the boundaries and we don't need it in this step so uncheck it as mentioned in figure 6.



Update Components and Nets

Update | I/O Pins | PR Boundary

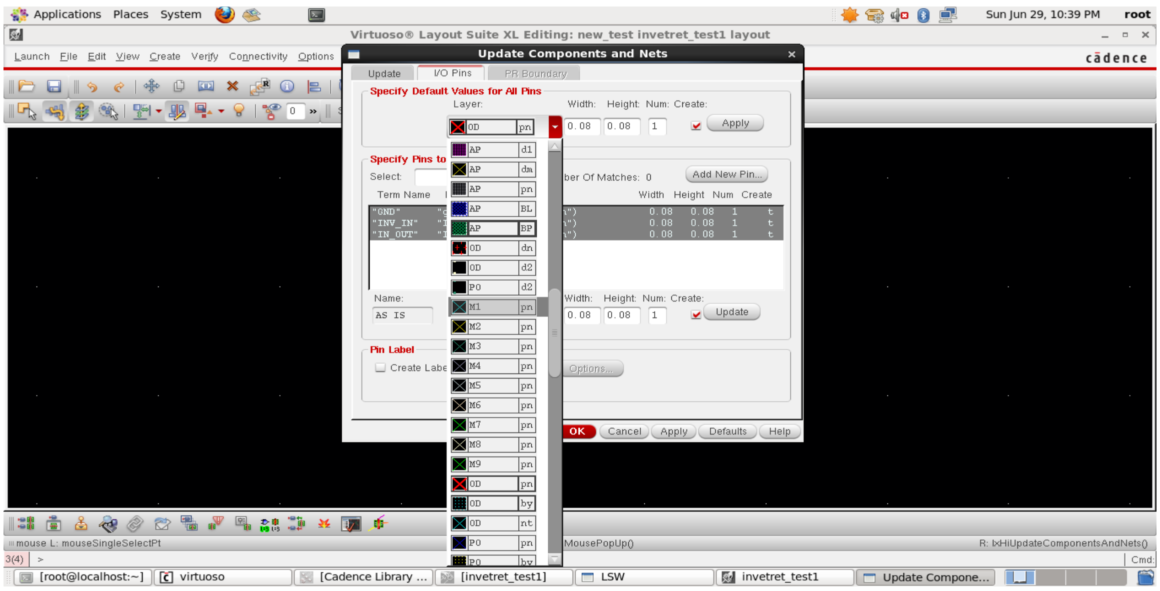
**Update**
- [ ] Update Selected Layout Components Only
- [ ] Update Nets and Name Mismatches Only
- [✓] Update Net Signal Type
- [✓] Update Instance Masters by Replacing the Old Instance
- [✓] Delete Unmatched Pins [✓] Delete Unmatched Instances
- [ ] Update Layout Parameters

**Generate**
- [✓] Instances
  - [ ] Automatic Chaining [ ] Automatic Folding
- [✓] I/O Pins
  - [ ] Except Global Pins
  - [ ] Except Pad Pins
- [ ] PR Boundary
  - [ ] Snap Boundary

**Device Correspondence**
- [✓] Preserve User-Defined Mappings

Figure 6: The generate/update window

Select the next tab "I/O pins" then select all the pins as shown in figure 7 then choose the type of the pin to be metal 1 and do the same for the 2 menus as illustrated in figure 7.



Update Components and Nets

Update | I/O Pins | PR Boundary

**Specify Default Values for All Pins**
Layer: OD pn | Width: 0.08 | Height: 0.08 | Num: 1 | Create: ✓ | Apply

**Specify Pins to ...**
Select: ... Number Of Matches: 0 | Add New Pin...

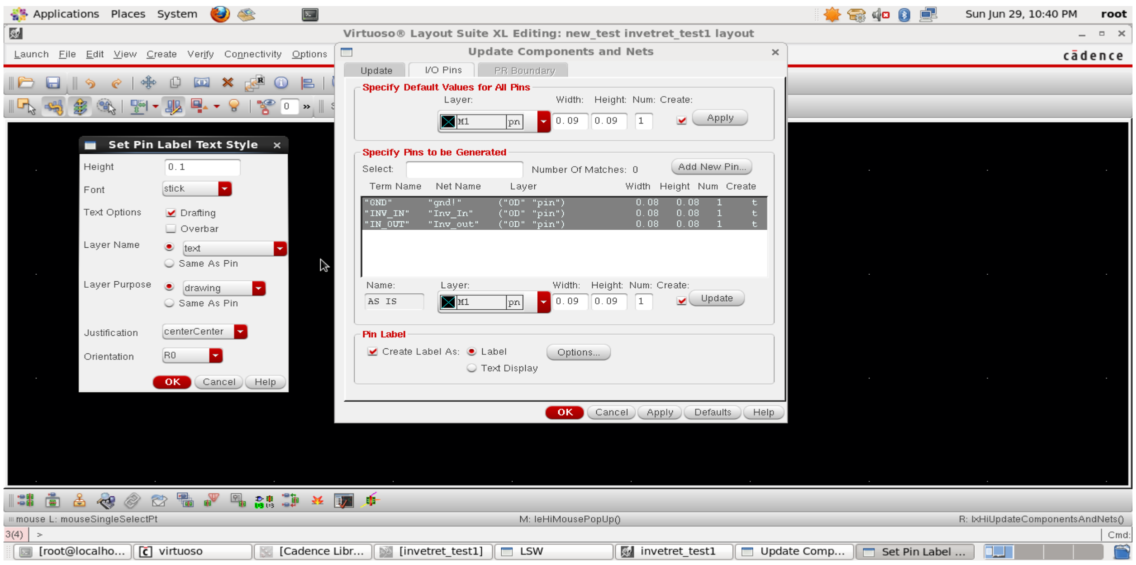
| Term Name | ... | Width | Height | Num | Create |
|---|---|---|---|---|---|
| "GND" | | 0.08 | 0.08 | 1 | t |
| "INV_IN" | | 0.08 | 0.08 | 1 | t |
| "IN_OUT" | | 0.08 | 0.08 | 1 | t |

Name: AS IS | Width: 0.08 | Height: 0.08 | Num: 1 | Create: ✓ | Update

**Pin Label**
- [ ] Create Label ... Options...

Figure 7: The I/O pin tab

After choosing M1 pin for both menus you have to check the box called "Create label as" and choose "label" then press options and change the "Height" to "0.1" instead of "1" as mentioned in figure 8. Press update and notice that the set of pins are all updated.



Set Pin Label Text Style

| | |
|---|---|
| Height | 0.1 |
| Font | stick |
| Text Options | [✓ Drafting] Overbar |
| Layer Name | [✓ text] Same As Pin |
| Layer Purpose | [✓ drawing] Same As Pin |
| Justification | centerCenter |
| Orientation | R0 |

OK Cancel Help

Update Components and Nets

Update | I/O Pins | PR Boundary

Specify Default Values for All Pins

| Layer: | Width: | Height: | Num: | Create: |
|---|---|---|---|---|
| M1 pn | 0.09 | 0.09 | 1 | ✓ |

Specify Pins to be Generated

Select: Number Of Matches: 0 Add New Pin...

```
Term Name    Net Name     Layer            Width   Height  Num  Create
"GND"        "gnd!"       ("OD" "pin")     0.08    0.08    1    t
"INV_IN"     "Inv_In"     ("OD" "pin")     0.08    0.08    1    t
"IN_OUT"     "Inv_out"    ("OD" "pin")     0.08    0.08    1    t
```

| Name: | Layer: | Width: | Height: | Num: | Create: |
|---|---|---|---|---|---|
| AS IS | M1 pn | 0.09 | 0.09 | 1 | ✓ |

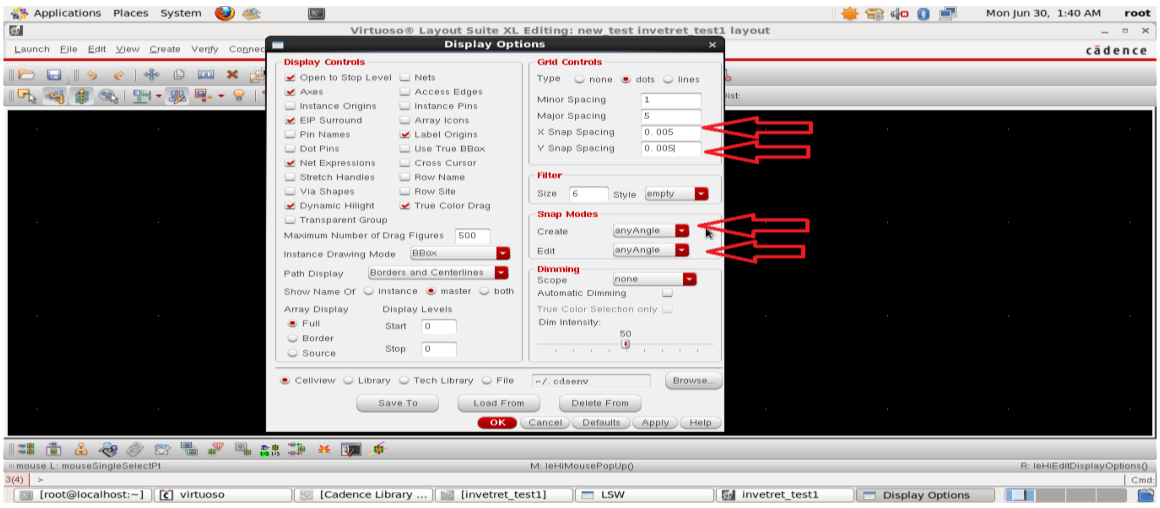
Pin Label

✓ Create Label As: [✓ Label] Text Display Options...

OK Cancel Apply Defaults Help

Figure 8: The set pin label dialogue box

Before starting to draw in the layout file; first you must make some steps every time you open the layout file whether creating a new one or editing an existing one. First you have to Press "Ctrl+F" in order to make the layout of the transistors appear. Then in order to make the movement of the transistors and components become easy press "E". The dialogue box shown in figure 9 will appear, choose the selected options as in the figure and press "OK".



Display Options

Display Controls

| | |
|---|---|
| ✓ Open to Stop Level | Nets |
| ✓ Axes | Access Edges |
| Instance Origins | Instance Pins |
| ✓ EIP Surround | Array Icons |
| Pin Names | ✓ Label Origins |
| Dot Pins | Use True BBox |
| ✓ Net Expressions | Cross Cursor |
| Stretch Handles | Row Name |
| Via Shapes | Row Site |
| ✓ Dynamic Hilight | ✓ True Color Drag |
| Transparent Group | |

Maximum Number of Drag Figures 500

Instance Drawing Mode BBox

Path Display Borders and Centerlines

Show Name Of instance [✓ master] both

Array Display: [✓ Full] Border Source

Display Levels: Start 0, Stop 0

Grid Controls

Type none [✓ dots] lines

| | |
|---|---|
| Minor Spacing | 1 |
| Major Spacing | 5 |
| X Snap Spacing | 0.005 |
| Y Snap Spacing | 0.005 |

Filter

Size 6 Style empty

Snap Modes

| | |
|---|---|
| Create | anyAngle |
| Edit | anyAngle |

Dimming

Scope none

Automatic Dimming

True Color Selection only

Dim Intensity: 50

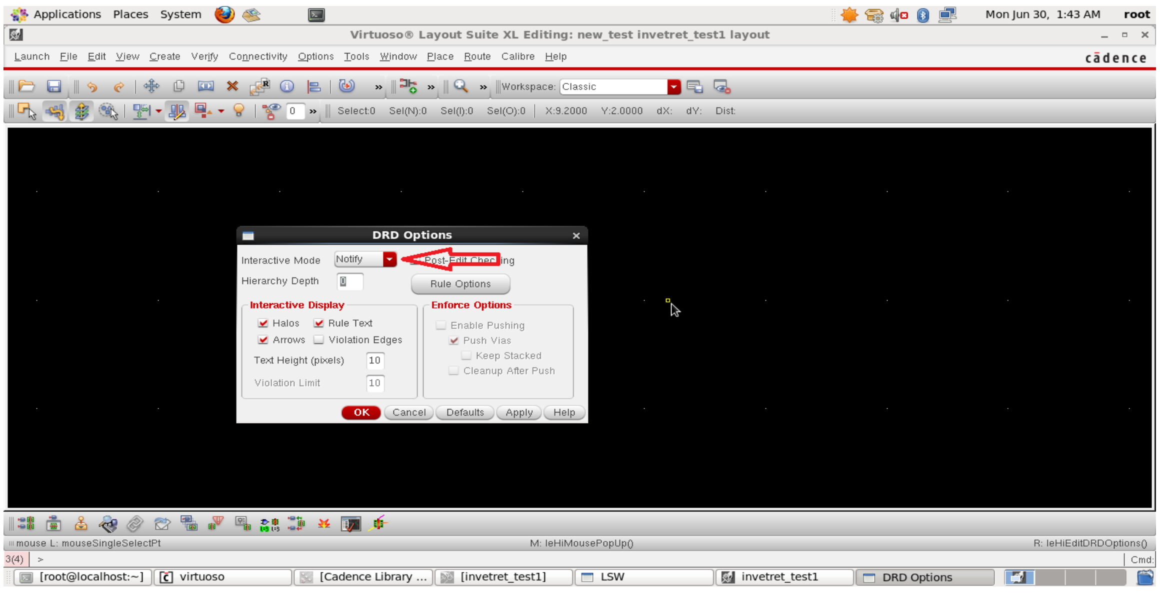
[✓ Cellview] Library Tech Library File ~/.cdsenv Browse...

Save To Load From Delete From

OK Cancel Defaults Apply Help

Figure 9: The display options window

In order to check the DRC while drawing the layout to make the drawing more easy; you have to go to the menu bar select "Options>>DRD edit". A dialogue box will appear you have to choose "Notify" as in figure 10.



Virtuoso® Layout Suite XL Editing: new_test invetret_test1 layout

DRD Options

Interactive Mode: Notify
Post-Edit Checking [ ]
Hierarchy Depth: 0

Interactive Display
- [✓] Halos
- [✓] Rule Text
- [✓] Arrows
- [ ] Violation Edges
- Text Height (pixels): 10
- Violation Limit: 10

Enforce Options
- [ ] Enable Pushing
- [✓] Push Vias
- [ ] Keep Stacked
- [ ] Cleanup After Push

Figure 10: The DRD options window

Now you shall start drawing the connections you need in the circuit using some shortcuts that is very helpful while drawing.

"Shift+Z": zoom out.

"Ctrl+Z": zoom in.

"M": move.

"S": stretching.

"R": drawing rectangle.

"P": drawing polygon.

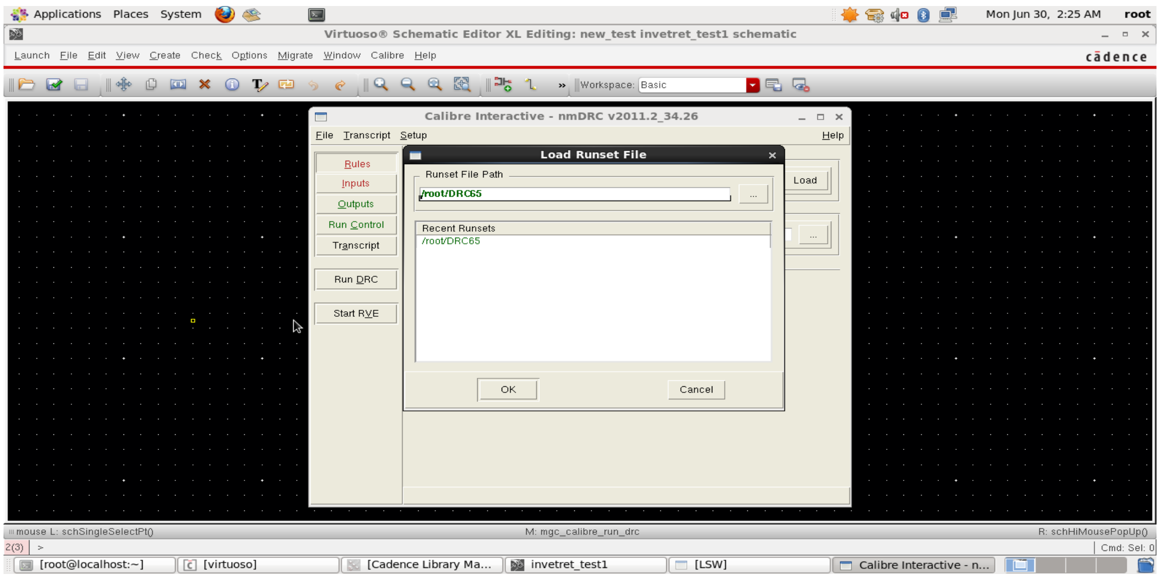
Once you finish the connections of the cell, you will need to connect the I/O pins. All you have to do is to press "Q" on the text of the pin for example the word "VDD" and choose metal 1 then connect it.

## Running DRC

"DRC" stands for "Design Rule Check". The main aim of this type of analysis is to check that the connections you have done in the layout design will work correctly and doesn't violate the standard design rules of the used technology.

In order to perform this type of analysis, you go to the menu bar and choose "Calibre>>Run DRC" a dialogue box will appear as shown in figure 11 asking you for the path of the run-set files. As you see in the figure; I have already included the path of the file then hit "OK". Press "Run DRC" button and wait until the error list appears.

Some errors will not be removed until the circuit is fabricated like density errors, floating gate errors, ESD errors, and R.1 errors so ignore these errors if there is nothing except them and move to the next step.



Virtuoso® Schematic Editor XL Editing: new_test invetret_test1 schematic

Calibre Interactive - nmDRC v2011.2_34.26

Load Runset File

Runset File Path: /root/DRC65

Recent Runsets: /root/DRC65

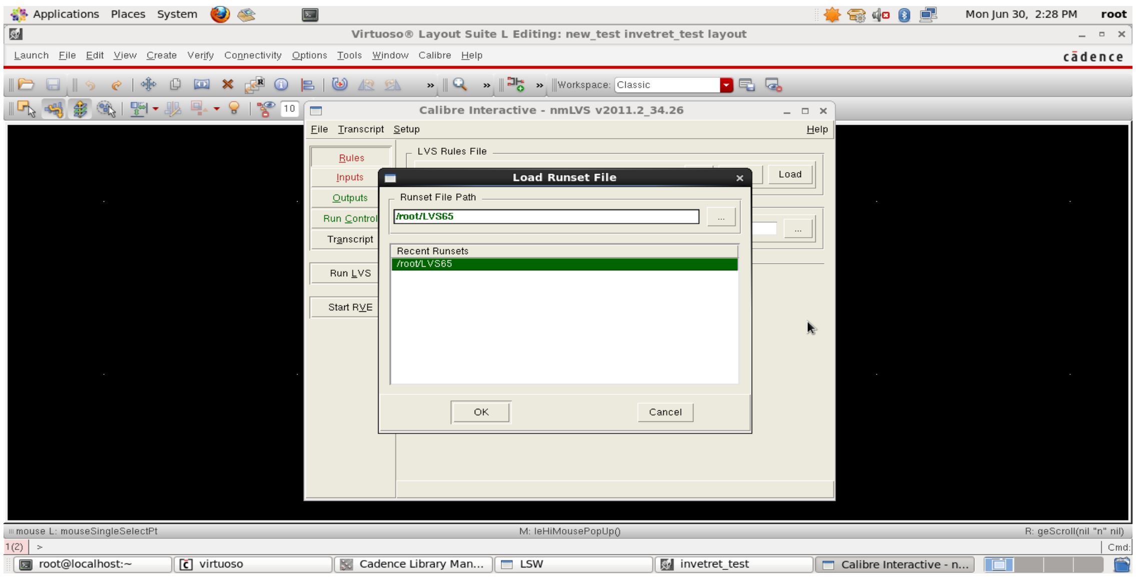
mouse L: schSingleSelectPt() M: mgc_calibre_run_drc R: schHiMousePopUp()

Figure 11: The DRC window

## Running LVS

"LVS" stands for "Layout Versus Schematic". This type of analysis checks the connections you have done in the layout design and compares it with the connections in the schematic design. If the connections were the same then the LVS simulation will pass, if not there will be some errors appearing in the LVS report.

In order to run this type of simulation, you have to go to the menu bar and choose "Calibre>>Run LVS" then a dialogue box will appear like the one shown in figure 12 asking you to attach the run-set files then press "OK". After this window disappears you have to press the "Run LVS" button.


Calibre Interactive - nmLVS v2011.2_34.26

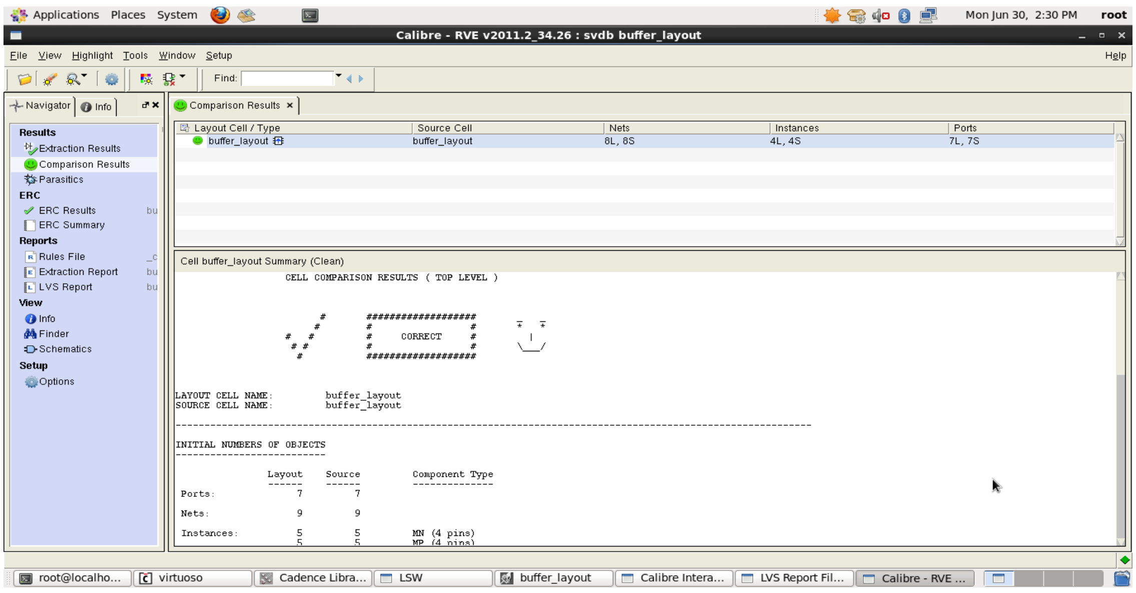
Load Runset File

Runset File Path: /root/LVS65

Recent Runsets: /root/LVS65

Figure 12: The LVS window

If your connections were correct the result of the simulation will be a green happy smile appearing at the left of the screen as shown in figure 13. If there was something wrong in the connection then open the LVS report and check the position of the error.


Calibre - RVE v2011.2_34.26 : svdb buffer_layout

| Layout Cell / Type | Source Cell | Nets | Instances | Ports |
|---|---|---|---|---|
| buffer_layout | buffer_layout | 8L, 8S | 4L, 4S | 7L, 7S |

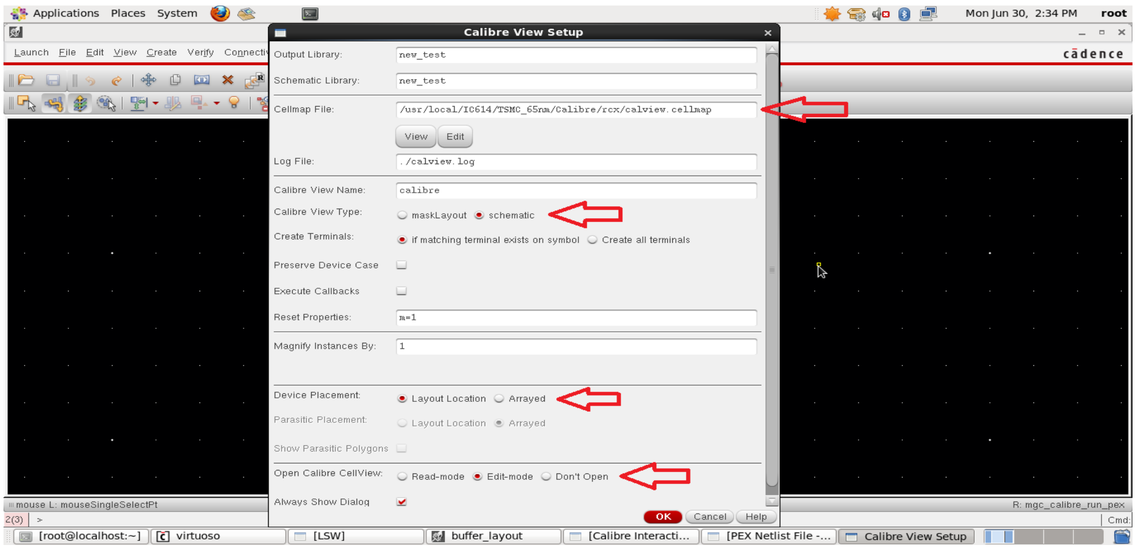
Cell buffer_layout Summary (Clean)

```
                CELL COMPARISON RESULTS ( TOP LEVEL )



                     #        ##################
                    #         #                #
                #   #         #     CORRECT    #
                 # #          #                #
                  #           ##################


LAYOUT CELL NAME:         buffer_layout
SOURCE CELL NAME:         buffer_layout

--------------------------------------------------------------------------------------------

INITIAL NUMBERS OF OBJECTS
--------------------------

                Layout    Source         Component Type
                ------    ------         --------------
 Ports:              7         7

 Nets:               9         9

 Instances:          5         5         MN (4 pins)
                     5         5         MP (4 pins)
```

Figure 13: The LVS result window

## Running PEX

"PEX" stands for "Parasitic Extraction". In this type of simulation the program extracts all the parasitic resistances and capacitances that can appear in your design and take it in consideration. This in turn will help you to perform post layout simulation which means that running the system after extracting the parasitic resistances and capacitances; and then comparing the results with the original results of the schematic.


Calibre View Setup

Output Library: new_test

Schematic Library: new_test

Cellmap File: /usr/local/IC614/TSMC_65nm/Calibre/rcx/calview.cellmap

Log File: ./calview.log

Calibre View Name: calibre

Calibre View Type: maskLayout [✓ schematic]

Create Terminals: [✓ if matching terminal exists on symbol] Create all terminals

Preserve Device Case ☐

Execute Callbacks ☐

Reset Properties: m=1

Magnify Instances By: 1

Device Placement: [✓ Layout Location] Arrayed

Parasitic Placement: Layout Location [✓ Arrayed]

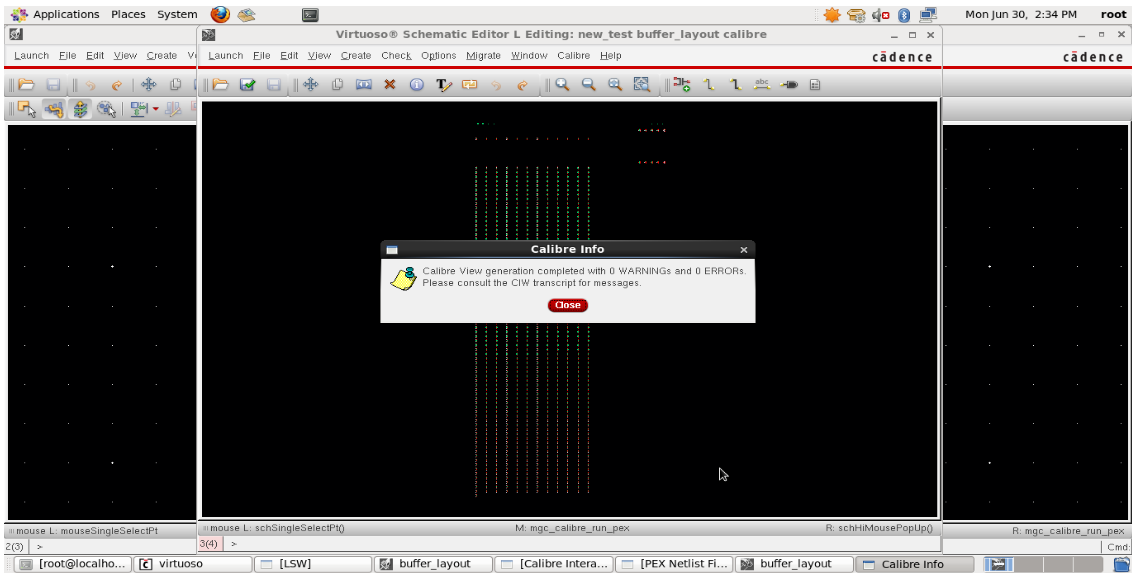
Show Parasitic Polygons ☐

Open Calibre CellView: Read-mode [✓ Edit-mode] Don't Open

Always Show Dialog ✓

Figure 14: The PEX options window

Running this type of simulation is done by heading to the menu bar and pressing "Calibre>>Run PEX". The same window asking for the run-set files that appeared in the previous two types will appear again and you will do the same steps then press "Run PEX" button. The window shown in figure 14 will appear and you have to choose the same options then press "OK" and wait until the window in figure 15 appears notifying you that the PEX simulation is finished with zero errors and zero warnings.


Virtuoso® Schematic Editor L Editing: new_test buffer_layout calibre
Calibre Info
Close
cādence

Calibre View generation completed with 0 WARNINGs and 0 ERRORs.
Please consult the CIW transcript for messages.

Figure 15: The PEX result window

After the appearance of this window you have to close the PEX, and the Layout window and save the state. Then open a new schematic file and build a test-bench for the circuit you have just built its layout like the one drawn in Figure 16.

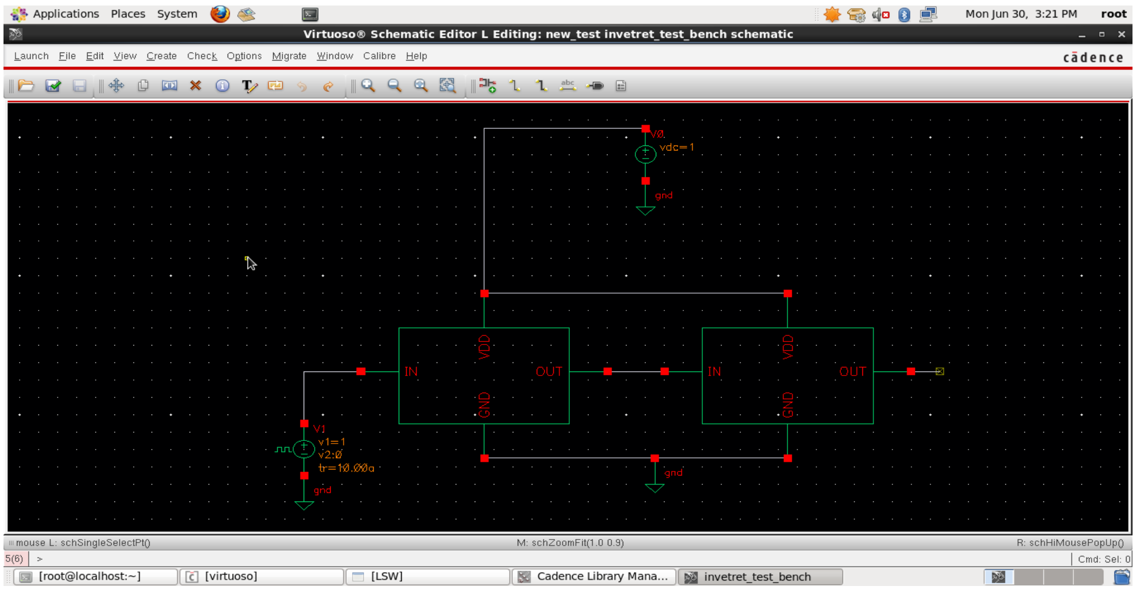

Virtuoso® Schematic Editor L Editing: new_test invetret_test_bench schematic
V0
vdc=1
gnd
IN
OUT
VDD
GND
IN
OUT
VDD
GND
V1
v1=1
v2:0
tr=10.00a
gnd
gnd
cādence

Figure 16: The test-bench schematic

Then open the ADE L and run the desired analysis you perform for your project. Before hitting the run button first you have to go to the menu bar select "Setup>>Environment" as mentioned in figure 17.

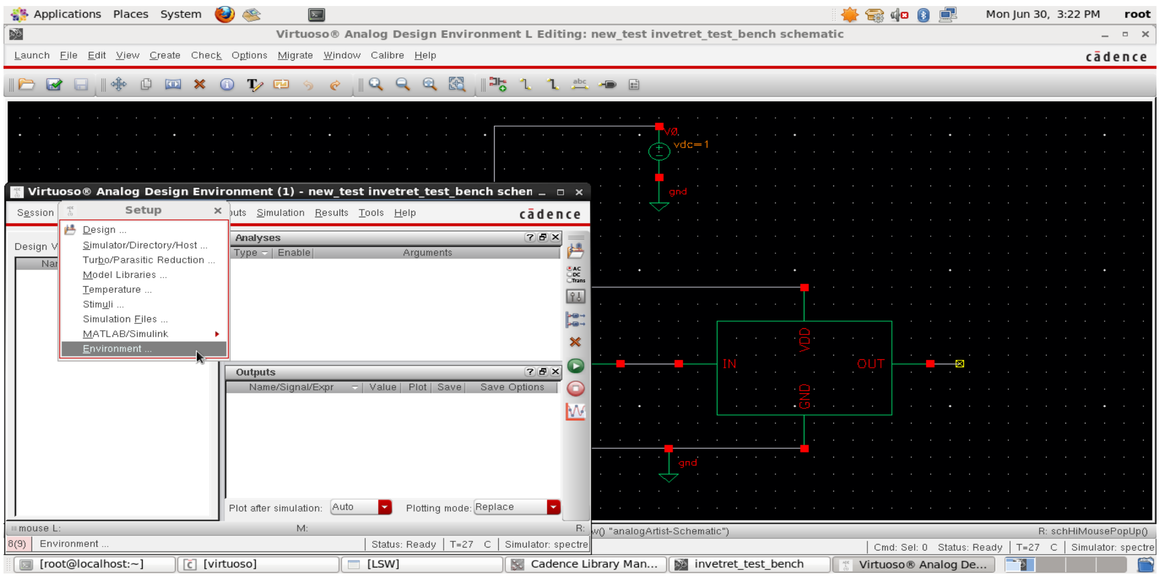
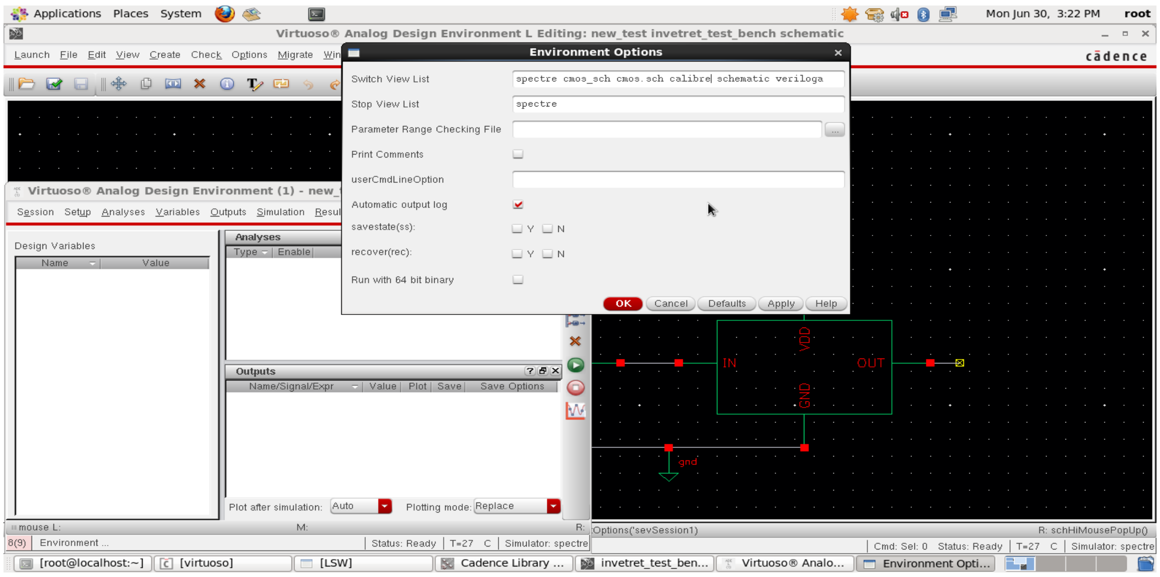

Virtuoso® Analog Design Environment L Editing: new_test invetret_test_bench schematic
Launch File Edit View Create Check Options Migrate Window Calibre Help
Virtuoso® Analog Design Environment (1) - new_test invetret_test_bench schen
Setup
Design ...
Simulator/Directory/Host ...
Turbo/Parasitic Reduction ...
Model Libraries ...
Temperature ...
Stimuli ...
Simulation Files ...
MATLAB/Simulink
Environment ...
Analyses
Outputs
Plot after simulation: Auto
Plotting mode: Replace
Status: Ready T=27 C Simulator: spectre

Figure 17: The ADE L window

A new window will appear showing you the types of simulations that the program is going to run. So in order to make the program run the circuit after extracting the parasitic resistances and capacitances you need to place the word "calibre" before the word "schematic" as in figure 18.


Virtuoso® Analog Design Environment L Editing: new_test invetret_test_bench schematic
Environment Options
OK
Cancel
Defaults
Apply
Help

| | |
|---|---|
| Switch View List | spectre cmos_sch cmos.sch calibre schematic veriloga |
| Stop View List | spectre |
| Parameter Range Checking File | |
| Print Comments | ☐ |
| userCmdLineOption | |
| Automatic output log | ✓ |
| savestate(ss): | ☐ Y ☐ N |
| recover(rec): | ☐ Y ☐ N |
| Run with 64 bit binary | ☐ |

Figure 17: The Environment options window

After setting these options you have to run the simulation and see the difference between the output in the case of schematic and case of layout. If the results aren't satisfying you have to go back to the layout and modify the connections in order to have small capacitance and resistance to decrease the delay then run LVS,PEX again.

## ENOB Calculation Tutorial

To calculate the ENOB we can use either the histogram method or the fast Fourier transform (FFT) method.in the following section we will describe the FFT method in details.

### FFT method

In the FFT method a specified length of data is recorded from the output of the ADC. For best results, the test must be arranged so that an exact integer number of input cycles occurs during the test period -this is known as coherent sampling. In other words, the following should be true

$$f_i = (\frac{J}{M})F_s$$

Where: J is an integer which is relatively prime to M, fs is the sampling frequency and M is the record length.

The condition of being relatively prime means that M and J have no common factors, i.e., their greatest common divisor is one. For the recommended frequency there are exactly J cycles in a record .If M is a power of two, then any odd value for J meets the relatively prime condition. That condition also guarantee that M uniformly distributed phases will be sampled.

#### *Steps of FFT method*

The following steps show how we can apply FFT method on the ADC

- Use a sin wave as an input signal for our system with an amplitude matching the full scale input span of the ADC.
- Calculate ENOB at different input frequencies (502, 1098, 2457…etc.). We choose numbers that Fs is not multiple from to obtain right results.
- If the ADC is nyquist one the practical Fin to calculate at is Fin=Fs/5
- Take 1024 samples from the output (or 512, 2048…etc.) and choose J that satisfy the condition $f_i = (\frac{J}{M})F_s$
- Run the simulation on the cadence and choose the transient stop time to achieve the required number of samples.
- Convert the output code of the ADC to analog signal by applying DAC equation on it
- Generate a table from the output graph. Table start by the first right output value from the ADC, end at the time achieve the require number of samples and step size =1/Fs

- Import this table to the MATLAB and run the code

## MATLAB code

This codes is obtained from sigma-delta ADC toolbox and we edited in it to be suitable for our design

```
Cadence_Output=data(:,2); %%to extract output from the table of the
Cadence
data_stream =Cadence_Output;
dc_offset = sum(data_stream)./length(data_stream);
data_stream = data_stream-dc_offset;
N = length(data_stream);    %Length of your timedomain data (remove
some samples for settling)
w=hann(N)/(N/4);    %Hann window %to adjust fft results
%plot(w)

bw=2500e6; %BW of your system %if ADC is nyquist BW=.5FS
Fs=5000e6; %sampling freq
Fin=2457e6;  %frequency of the input test sin wave
f=Fin/Fs;           % Normalized signal frequency
fB=N*(bw/Fs);       % Base-band frequency bins (the BW you are
looking at)
[snr,ptot,psig,pnoise,output_W]=calcSNR(data_stream,f,3,fB,w,N);
ENOB=(snr-1.76)/6.02    % Equivalent resolution in bits
L=length(data_stream);
 fx=linspace(-Fs/2,Fs/2,L); %frequency axis
 plot(fx,20*log10(fftshift(output_W)*1000))
```

-------------------------------------------functions used in the code---------------------------------

```
function [snrdB,ptotdB,psigdB,pnoisedB,output_W] =
calcSNR(vout,f,fBL,fBH,w,N)
% SNR calculation in the time domain (P. Malcovati, S. Brigati)
%
% [snrdB,ptotdB] = calcSNR(vout,f,fBL,fBH,w,N)
% [snrdB,ptotdB,psigdB] = calcSNR(vout,f,fBL,fBH,w,N)
% [snrdB,ptotdB,psigdB,pnoisedB] = calcSNR(vout,f,fBL,fBH,w,N)
%
% vout:         Sigma-Delta bitstream taken at the modulator output
% f:            Normalized signal frequency (fs = 1)
% fBL:          Base-band lower limit frequency bins
% fBH:          Base-band upper limit frequency bins
% w:            Windowing vector
% N:            Number of samples
%
% snrdB:        SNR in dB
% ptotdB:       Sigma-Delta modulator output power spectral density
(vector) in dB
% psigdB:       Extracted signal power spectral density (vector) in
dB
% pnoisedB:     Noise power spectral density (vector) in dB

fBL=ceil(fBL);
fBH=ceil(fBH);
signal=(N/sum(w)).*sinusx(vout(1:N).*w,f,N);    % Extracts sinusoidal
signal
noise=vout(1:N)-signal;                        % Extracts noise
components
stot=((abs(fft((vout(1:N).*w)'))).^2);         % Bitstream PSD
```

```
ssignal=(abs(fft((signal(1:N).*w)'))).^2;    % Signal PSD
snoise=(abs(fft((noise(1:N).*w)'))).^2;      % Noise PSD
pwsignal=sum(ssignal(fBL:fBH));              % Signal power
pwnoise=sum(snoise(fBL:fBH));            % Noise power
output_W=stot;
snr=pwsignal/pwnoise;
snrdB=dbp(snr);
norm=sum(stot(1:N/2));%/sum(vout(1:N).^2)*N;     % PSD normalization
if nargout > 1
    ptot=stot/norm;
    ptotdB=dbp(ptot);
end
if nargout > 2
    psig=ssignal/norm;
    psigdB=dbp(psig);
end
if nargout > 3
    pnoise=snoise/norm;
    pnoisedB=dbp(pnoise);
end

function y = dbp(x)
% Calculates the input value in dB dbp(x) = 10*log10(x)
% (by S. Brigati, P. Malcovati)
%
% y = dbp(x)
%
% x:    Input
%
% y:    Output in dB

y = -Inf*ones(size(x));
nonzero = x~=0;
y(nonzero) = 10*log10(abs(x(nonzero)));

function y = dbv(x)
% Calculates the input value in dB dbp(x) = 20*log10(x)
% (by S. Brigati, P. Malcovati)
%
% y = dbv(x)
%
% x:    Input
%
% y:    Output in dB

y = -Inf*ones(size(x));
nonzero = x~=0;
y(nonzero) = 20*log10(abs(x(nonzero)));

function outx = sinusx(in,f,n)
% Extracts of a sinusoidal signal (S. Brigati, P. Malcovati)
%
% outx = sinusx(in,f,n)
%
% in:       Input data vector
% f:        Normalized input signal frequency
% n:        Number of simulation points
%
% outx:     Sinusoidal signal
```

```
sinx=sin(2*pi*f*[1:n])';
cosx=cos(2*pi*f*[1:n])';
in=in(1:n);
a1=2*sinx.*in;
a=sum(a1)/n;
b1=2*cosx.*in;
b=sum(b1)/n;
outx=a.*sinx + b.*cosx;
```