SUCCESSIVE APPROXIMATION REGISTER WITH CONTINOUS DIS-ASSEMBLY ALGORITHM (SAR-CD) AND CIRCUIT DESIGN FOR TIME-BASED ANALOG TO DIGITAL CONVERTERS (TADC)

by
Karim Osama Ragab Mahmoud

A Thesis Submitted to the
Faculty of Engineering at Cairo University
in Partial Fulfillment of the
Requirements for the Degree of
MASTER OF SCIENCE
in
ELECTRONICS AND ELECTRICAL COMMUNICATIONS ENGINEERING

FACULTY OF ENGINEERING, CAIRO UNIVERSITY
GIZA, EGYPT
2017

SUCCESSIVE APPROXIMATION REGISTER WITH CONTINOUS DIS-ASSEMBLY ALGORITHM (SAR-CD) AND CIRCUIT DESIGN FOR TIME-BASED ANALOG TO DIGITAL CONVERTERS (TADC)

by
Karim Osama Ragab Mahmoud

A Thesis Submitted to the
Faculty of Engineering at Cairo University
in Partial Fulfillment of the
Requirements for the Degree of
MASTER OF SCIENCE
in
ELECTRONICS AND ELECTRICAL COMMUNICATIONS ENGINEERING

Under the Supervision of

Associate Prof. Ahmed Emira
Principal Advisor

Assistant Prof. Hassan Mostafa
Advisor

FACULTY OF ENGINEERING, CAIRO UNIVERSITY
GIZA, EGYPT
2017

SUCCESSIVE APPROXIMATION REGISTER WITH CONTINOUS DIS-ASSEMBLY ALGORITHM (SAR-CD) AND CIRCUIT DESIGN FOR TIME-BASED ANALOG TO DIGITAL CONVERTERS (TADC)

by
Karim Osama Ragab Mahmoud

A Thesis Submitted to the
Faculty of Engineering at Cairo University
in Partial Fulfillment of the
Requirements for the Degree of
MASTER OF SCIENCE
in
ELECTRONICS AND ELECTRICAL COMMUNICATIONS ENGINEERING

Approved by the examining Committee

Associate Prof. Ahmed Emira, Thesis Main Advisor

Prof. Muhammed Riad El Gonamy, Internal Examiner

Associate Prof. Yahia Ghallab, External Examiner
(Zewail City for Science and Technology)

FACULTY OF ENGINEERING, CAIRO UNIVERSITY
GIZA, EGYPT
2017

© Karim Osama Ragab Mahmoud 2017
All Rights Reserved

Engineer's Name: Karim Osama Ragab
Date of Birth: 8/9/1989
Nationality: Egyptian
E-mail: engkarimosama@gmail.com
Phone: 01002393426
Address: Electronics and Communications Engineering Department, Cairo University, Giza 12613, Egypt

Registration Date: 01/10/2012
Awarding Date: dd/mm/yyyy
Degree: Master of Science
Department: Electronics and Communications Engineering

Supervisors:
Associate prof. Ahmed Emira
Assistant prof. Hassan Mostafa

Examiners:

Associate prof. Ahmed Emira	(Thesis main advisor)
Prof. Muhammed Riad El Gonamy	(Internal examiner)
Associate prof. Yahia Ghallab	(External examiner)

Title of Thesis:
Successive Approximation register with Continuous Dis-assembly Algorithm (SAR-CD) and circuit design for Time-based Analog to Digital Converters (TADC)

Key Words:
Time to Digital Converter (TDC); Voltage to Time Converter (VTC); Analog to Digital Converter (ADC); Biomedical circuits

Summary:
This work proposes a novel algorithm for analog to digital conversion. The algorithm is a modified version of the successive approximation algorithm in which binary sub-weights of the input maximum are used to evaluate the corresponding digital words in a cyclic manner. The proposed algorithm moves the conditioning between the evaluated bits from the analog domain to the digital domain. In folded versions of the successive approximation ADC circuits, in which bits are evaluated in an iterative fashion, digital to analog converters may not be needed anymore. This major advantage promises for reduction in fabrication area and power consumption. A full mathematical proof for the algorithm is also introduced. A new circuit design is developed to utilize the algorithm benefits. Results show competent power and reduction with state-of-art designs.

Acknowledgment

In the name of Allah the most merciful the most gracious; all thanks to Allah the Lord of the Heavens and Earth and peace be upon Mohamed and his companions. I am also very gratefull to have Associate Prof. Ahmed Emira as my principle supervisor. I wish to express my gratitude to my adviser, Prof. Hassan Mostafa who was helpful and offered invaluable assistance, support and guidance. I am also genuinely blessed to have Associate Prof. ————- as a member of the supervisory committee, for his great efforts and constant care.

Many thanks to my family and friends for their support and help through the duration of this work.

My deepest gratitude to my family. Without their encouragement, I would not have gone this far.

Karim.

Abstract

ANAlog to Digital Converters(ADC) and Digital to Analog Converters(DAC) define the boundaries between the analog and digital blocks in every machine. Optimizing powerful data converters means that more analog blocks can be digitized. Digital blocks are much easier in design and manipulation. Also they provide robust performance against noise with less power consumption and fabrication area benefitting every new fabrication technology.

Many ADC architectures have been proposed to serve wide range of applications. Choosing the right ADC architecture is based on many factors such as sampling rate, power consumption, fabrication area, resolution, robustness towards Process, Voltage and Temperature (PVT) changes. Sigma-delta, flash, pipelined and Successive Approximation Register (SAR) present a wide range of architectures each.

Time-based ADC (TADC) is a special category of ADCs in which part of the data converter is desired to be digital in nature. The conversion between an analog quantity and the corresponding digital representation is done in two steps. The first step is to convert the analog quantity from voltage amplitude change into time change. In the second step, the signal representation in time change is converted to the corresponding digital binary representation. Most of the conversion effort is done in the second step. As the job is now simplified by the first step, the second one is expected to be implemented by digital components.

This work proposes a novel algorithm for analog to digital conversion. The algorithm is a modified version of the successive approximation algorithm in which binary sub-weights of the input maximum are used to evaluate the corresponding digital words in a cyclic mannar. The proposed algorithm moves the conditioning between the evaluated bits from the analog domain to the digital domain. In folded versions of the successive approximation ADC circuits, in which bits are evaluated in an iterative fashion, digital to analog converters may not be needed

anymore. This major advantage promises for reduction in fabrication area and power consumption. A full mathematical proof for the algorithm is also introduced. A new circuit design is developed to utilize the algorithm benefits. Results show competent power and reduction with state-of-art designs.

Contents

List of Tables

List of Figures

List of Symbols and Abbreviations

Abbreviations

ADC Analog to digital converter.

DAC Digital to analog converter.

DFF D–flip-flop.

DNL Differential non-linearity.

ENOB Effective number of bits.

FFT Fast fourier transform.

INL Integral non-linearity.

LSB Least significant bit, the index of the bit of smallest weight in binary word.

MSB Least significant bit, the index of the bit of largest weight in binary word.

MUX Multiplexer.

PVT Process-Voltage-Temperature , relating to their effects to circuit operation.

S&H Sample and hold.

SAR Successive approximation register algorithm.

SAR-CD Successive approximation register with continuous dis-assembly algorithm.

SNDR Signal to noise and distortion ratio.

SNR Signal to noise ratio.

SOC System on chip.

SQNR Signal to quantization noise ratio.

TADC Time-based analog to digital converter.

TDC Time to digitial converter.

UWB Ultra wide band.

VFC Voltage to frequency converter.

VTC voltage to time converter.

This thesis is dedicated to my father (may allah forgive him).

Chapter 1

Introduction

Due to technology scaling, there is a great demand to replace the analog designs with their digital counterparts. In Time-based Analog to Digital Converters (TADC), the input voltage is first converted into an intermediate change in frequency, pulse position or pulse width. This is performed by the first block of the TADC, which is denoted by the Voltage to Time Converter (VTC). This intermediate change is digitized by the second block of the TADC which is denoted by Time to Digital Converter (TDC). Most of the digital processing is conducted by the TDC block taking all the advantages of the CMOS technology scaling.

Pulse width modulation-based TADC utilizes the difference between the positive edges of two signals or the width of a given input pulse and compares it to the full scale. This kind of TDC has several applications such as: collecting on-chip measurements [11], replacing phase or frequency detector in frequency synthesis [12] and sensors interfacing [13]. Flash-like TDC promotes the usage of delay lines [14] which can have resolution as low as unit delay elements [15]. However, increasing the number of bits means exponentially increasing the area and power [14, 15]. Several methods to increase the number of bits or the dynamic range are investigated. In [16], the Most Significant Bits (MSBs) are calculated by counting the number of complete reference clock cycles inside the measured interval. Following, The Least Significant Bits (LSBs) are calculated using Vernier delay lines.

On the other hand, implementations based on binary search algorithms take place in applications that require low power and area with high resolution. In [17, 18], the well-known Successive Approximation Register (SAR) is applied in a cyclic manner. However, the existence of two signal branches imposes a critical condition to keep the signals synchronized along the conversion, and the error in

one branch is accumulated each iteration. This error accumulation becomes worse due to the conditioning on each path of the two signal paths.

In this work a new algorithm is introduced to move this dependency from the time domain to the digital domain. In addition, the algorithm needs only one signal path instead of two which relaxes the strict synchronization requirements. Two circuit design approaches are presented to demonstrate the advantages of the new algorithm.

This thesis is arranged as follows: Chapter II presents a brief review to the main analog to digital conversion approaches and measures. Also, Time-based analog to digital conversion concepts and circuit architectures are presented in the same chapter. The proposed SAR-CD algorithm with derived proof is introduced in Chapter III. Chapter IV proposes two circuit design architectures using the new algorithm with detailed results and analysis. Chapter V introduces calibration algorithm for the second circuit architecture to compete for the Process Voltage Temperature (PVT) changes.

Chapter 2

Background

Analog to digital conversion is the process in which the input analog quantity is approximated to a digital number to be stored in a machine. This analog quantity can be a voice signal or a signal from a given sensor. The number of digits allocated for each sample in the machine presents the accuracy of the approximation, or what is called, quantization. Figure 2.1 presents a 3-bits ADC. Which means that every sample of the input is stored in a bits with possible value of 8 levels. The Digital to Analog Converter (DAC) performs the opposite function, which is restoring the analog signal using the saved digital words.

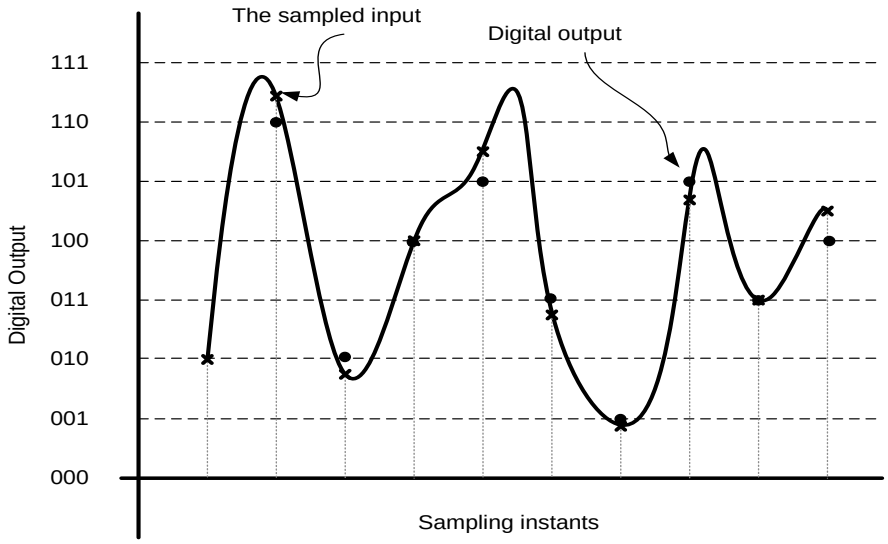


Figure 2.1: Example of 3-bits quantization [1]

2.1 ADC functions

2.1.1 Sampling

Sampling is the process of selecting periodic instances from the original signal to be quantized and saved to the target machine. The sampling frequency should be high enough to successfully present the original signal when restored using the digital to analog converter. The sampling frequency should follow:

$$fs \geq 2 * f_{BW} \tag{2.1}$$

Which is known as Nyquist rate. As fs is the sampling rate and f$_{BW}$ is the input signal bandwidth. 2.2 (a) Presents an example of a continuous signal in the frequency domain of a given bandwidth (2B). The signal is sampled with sampling frequency (f$_s$) higher than the Nyquist rate (b) and with a frequency smaller than the Nyquist rate (c). The signal in (b) can be used to re-constructed the original signal by applying a low pass filter to extract the signals below B. However, the signal in (c) cannot be used to re-constructed the original signal again as lower sampling frequency could not successfully present the signal.

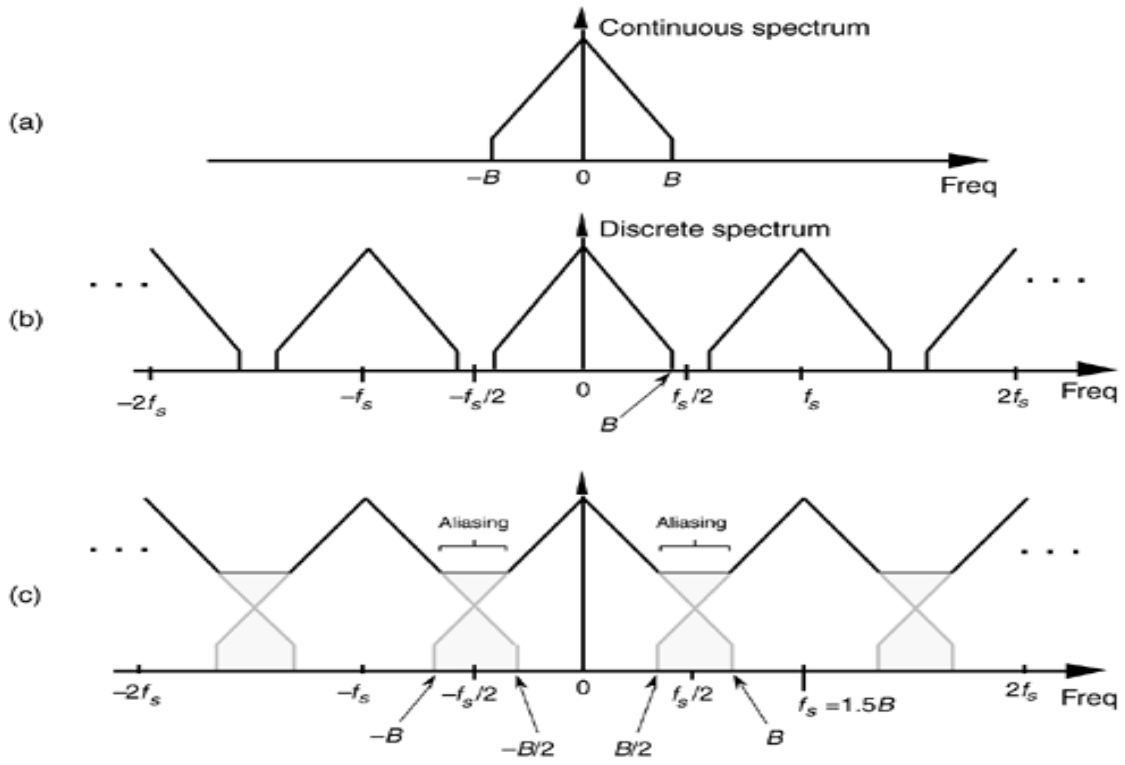


Figure 2.2: Nyquist rate sampling [2]

2.1.2 Quantization

Quantization is the process in which the analog sample is approximated to one of the available values to be stored in the target memory with digital representation. In figure 2.3, the input analog sinewave (with scaled amplitude and DC offset of 4 for clarity purpose) is quantized into 8 level so that each sample would need 3 bits to be stored ($2^3 = 8$). The quantization error is defined by the maximum difference between the original sample and the quantized (approximated) one and is found to be the half of the difference between each two levels; or in other words +/-LSB/2, as LSB stands for Least Significant Bit, which is the minimum resolution of the ADC. Figure 2.4 shows the quantization error for the signal in figure 2.3. As the amplitude is scaled to 8, the LSB equals one amplitude unit. And the maximum error in this case is (+\-) 1/2 = (+/-) LSB/2. In general:

$$LSB = \frac{Vfs}{2^n} \tag{2.2}$$

As Vfs is the full scale value and n is the number of ADC bits.

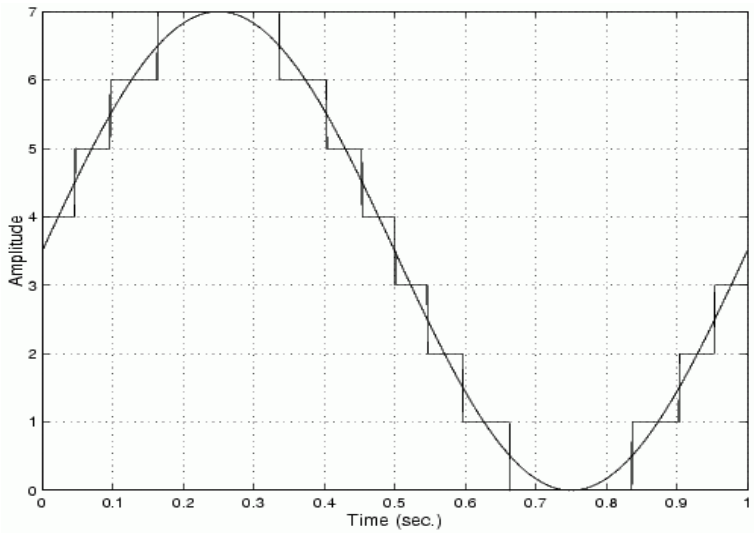


Figure 2.3: Quantization of analog sinewave to 3-bit [3]

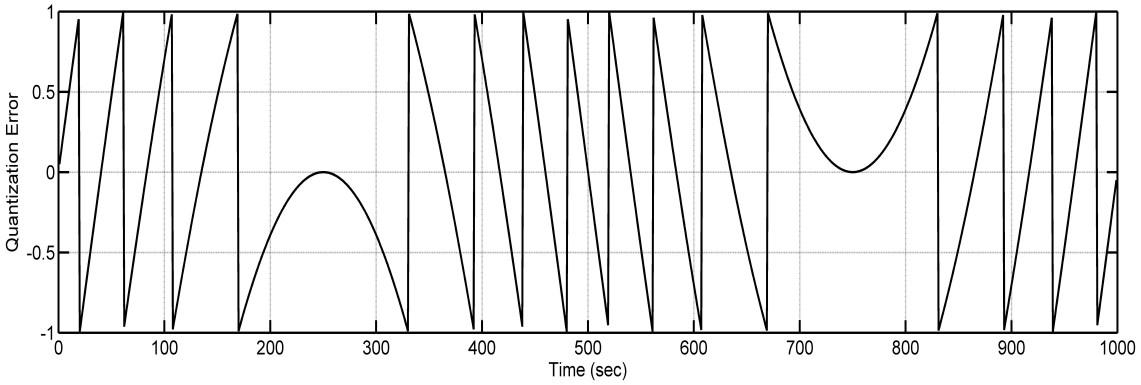


Figure 2.4: Quantization Error of +/-(LSB/2) [3]

Increasing the number of bits for each sample means increasing the available quantization levels by a power of 2, hence, increasing the accuracy.

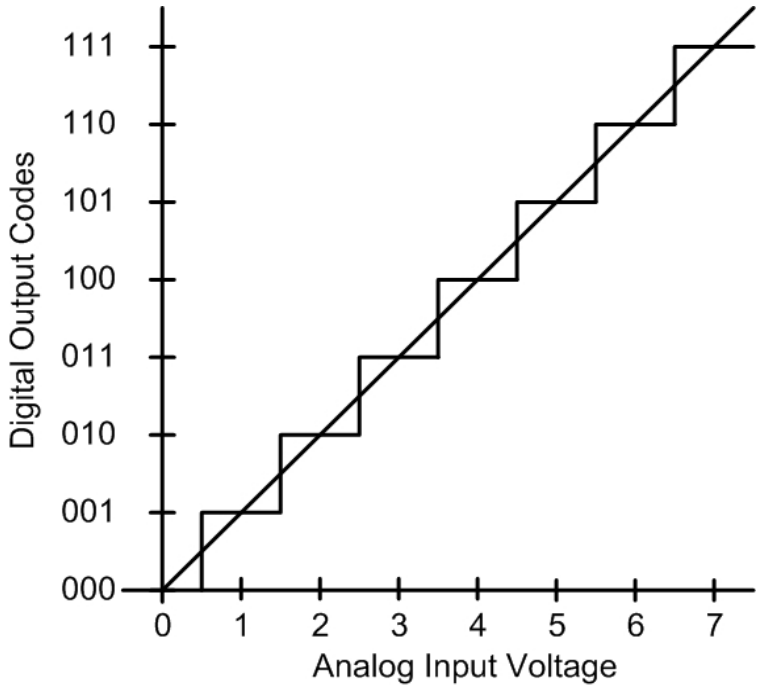


Figure 2.5: Ideal ADC input-output transfer function

2.2 ADC static characteristics

A linear ADC is the one in which the change in output digital word depends only on the amount of change of the input voltage along the designed input range. The characteristic is found to be as a linear relation between the input and the output. Figure 2.5 presents an ideal input-output transfer function for a 3-bits ADC.

Deviations from the ideal transfer function in real circuits are characterized in 4 main static characteristics [5]: 1- Offset error, 2- Gain error, 3-Differential Non-Linearity and 4-Integral non linearity. The next sub-sections reviews all the four charaterization measures.

2.2.1 Offset error

The offset error is defined and the deviation of the input-output characteristic line from the ideal one as shown in figure 2.6. This offset can be positive or negative. Usually, minor run time calibration can fix error.

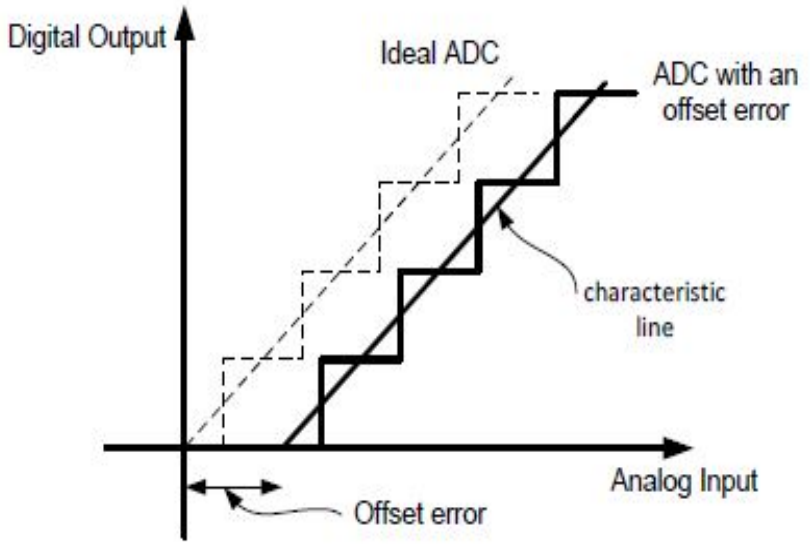


Figure 2.6: Offset error for a non-ideal ADC [1]

2.2.2 Gain error

The Gain error is defined as the deviation of the input output graph slope from the ideal case. Figure 2.7 shows an example of a gain error which causes a maximum error of 1 LSB. The error can be positive or negative. This type of error may require more complicated calibration techniques in comparison to the offset error.

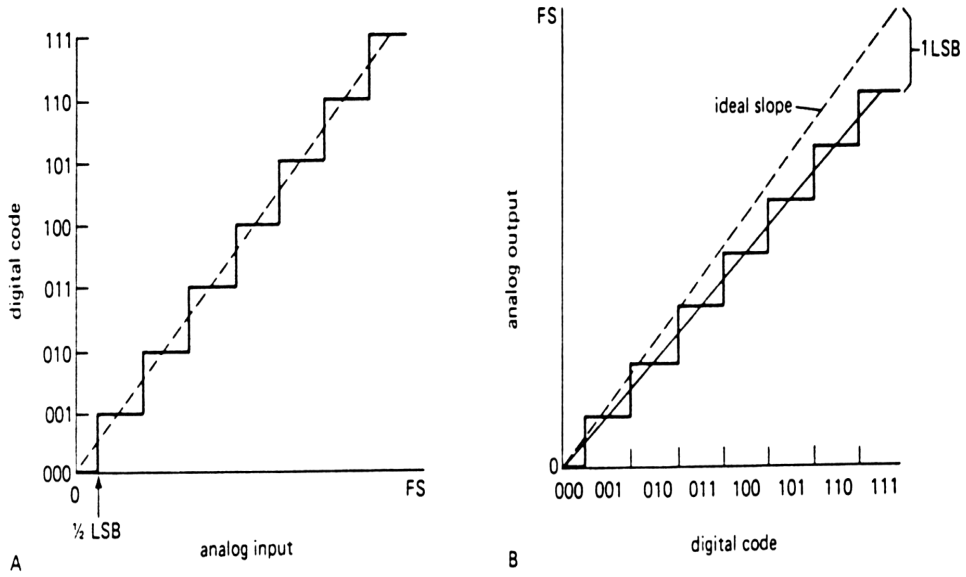


Figure 2.7: Gain error for a non-ideal ADC with 1 LSB maximum Error

2.2.3 Differential Non Linearity

It is defined as the deviation of the step size of the data converter from the ideal width of the bins. Assuming $X_k$is the transition point between successive codes $k-1$and k, then the width of the bin k is :

$$\triangle_r(k) = X_{K+1} - X_K$$

Then the differential non-linearity is:

$$DNL(k) = \frac{\triangle_r(k) - \triangle}{\triangle} \tag{2.3}$$

as $\triangle$ is the ideal step size.

The maximum differential nonlinearity is the maximum of $|DNL(k)|$ for all k. Usually, DNL is a measure that can be given the ADC datasheet in which it simply referred to the maximum differential non linearity.

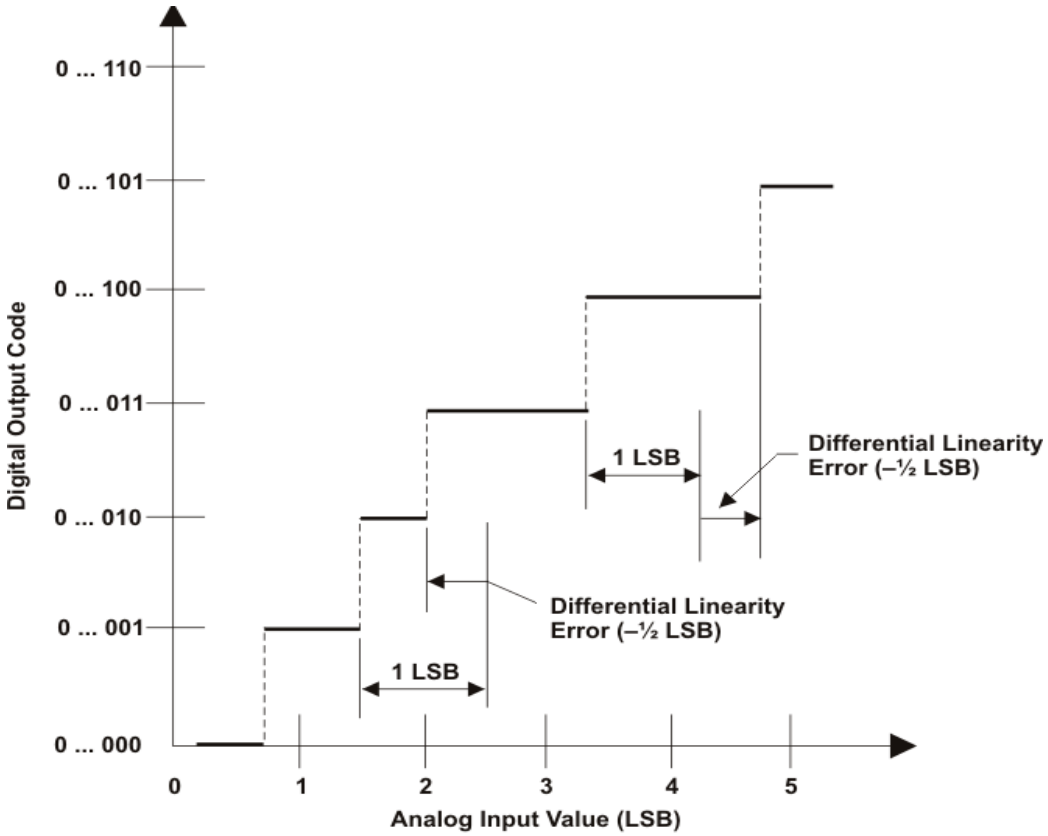


Figure 2.8: Differential non-linearity example [4]

2.2.4 Integral Non-Linearity

The integral non-linearity is the deviation of the ADC transfer function from the ideal transfer function. It also can be expressed as the summation of all the DNL across all the input output transfer function:

$$INL(k) = \sum_{i=0}^{k} DNL \tag{2.4}$$

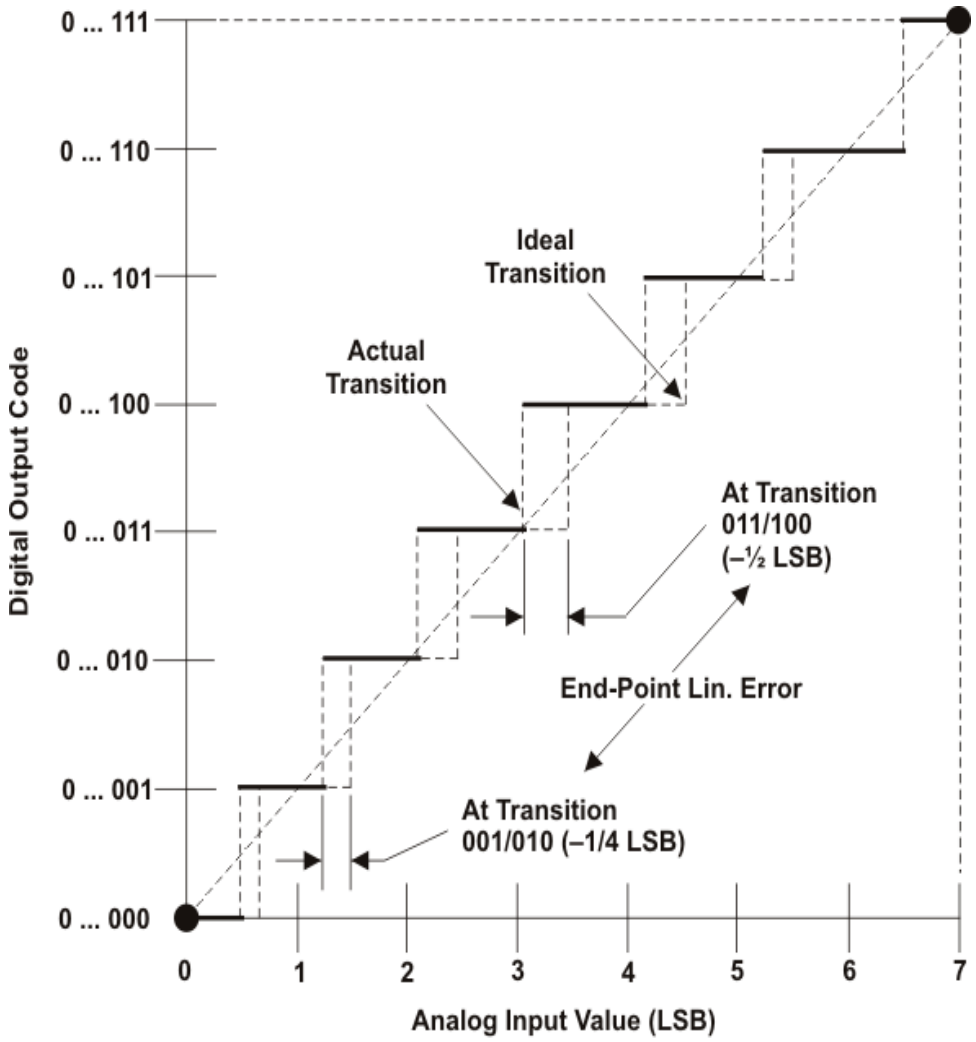


Figure 2.9: Integral non-linearity example [4]

2.2.5 Missing Codes

The ADC can skip some codes from the designed input-output transfer function. This can be indicated spotted by a sudden increase in the DNL for more than 1 LSB. A maximum DNL less than 1 DNL and INL value close to this number guarantees that this error cannot happen.

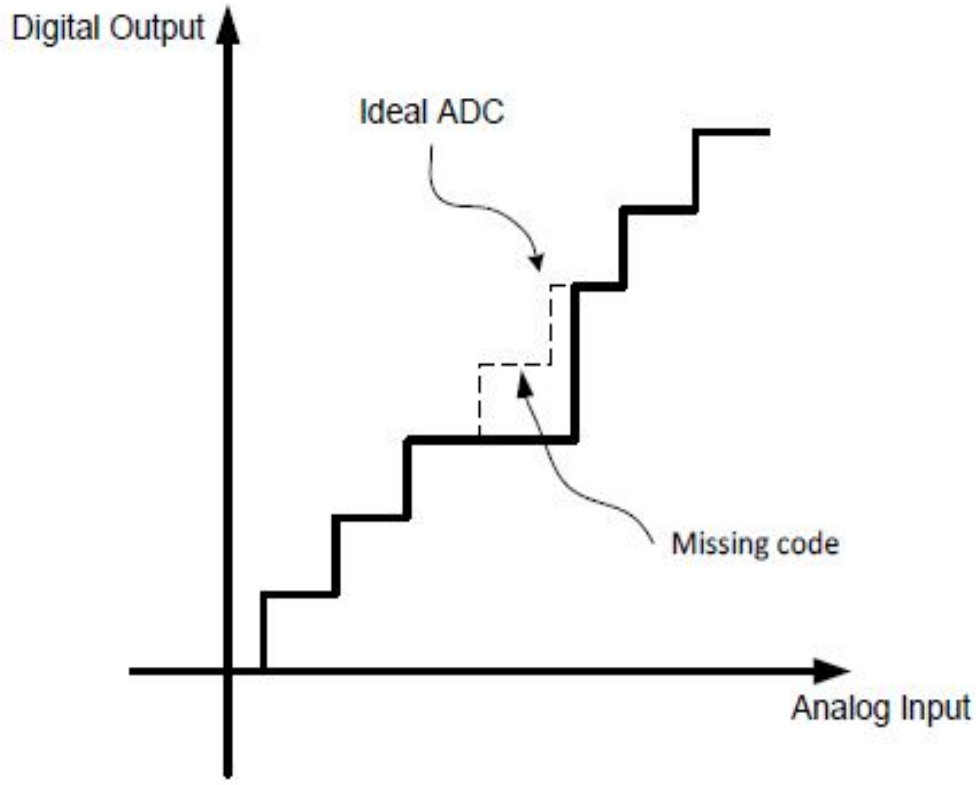


Figure 2.10: example of a missing code when a sudden DNL greater than 1 LSB occurs [1]

2.3 ADC Dynamic characteristics

The dynamic performance determines the frequency response and speed of the analog components of data converters. The performance is a concern when the input bandwidth and the conversion rate are high. The quality indicator of good dynamic feature is its capability to remain unchanged within the entire target range of operation. In the next subsections, some of the dynamic specifications are presented; [5] includes many other measure that may be an interest for the designer.

2.3.1 Analog Input Bandwidth:

The analog input bandwidth specifies the frequency at which a full scale ADC input leads to an output 3db below its low frequency value.

2.3.2 Input Impedance

The input impedance specifies the impedance between the input terminals of the ADC. At low frequency the input impedance is a resistance: ideally, it is infinite for voltage inputs and zero for current inputs (thus leading to an ideal measure of voltage or current.) At high frequency the input impedance is dominated by its capacitive component. Often, a switched capacitance structure performs the input sampling. In this case the specification provides the equivalent load at the input pin. At very high frequency the input impedance of the ADC must be the matched termination of the input connection.

2.3.3 Equivalent input referred noise

It is a measure of the electronic noise produced by the ADC circuit components. The result is that for a constant dc input the output is not fixed but there is a distribution of codes centered around the output code nominally encoding the input. For large number of samples, the noise distribution is approximately Gaussian. The standard deviation of the distribution defines the equivalent input referred noise. It is normally expressed in terms of LSBs or rms voltage. Figure 2.11 shows the histogram at the output of a possible data converter [5].

2.3.4 Maximum sampling frequency and conversion time

The maximum sampling frequency defines the maximum rate the input can be sampled and converted to the corresponding digital form. The conversion time is the time required for the ADC to sample and convert one sample. Usually, the conversion time is the inverse of the sampling rate unless the architecture is pipelined. Both measures are important and define the application the ADC can serve.

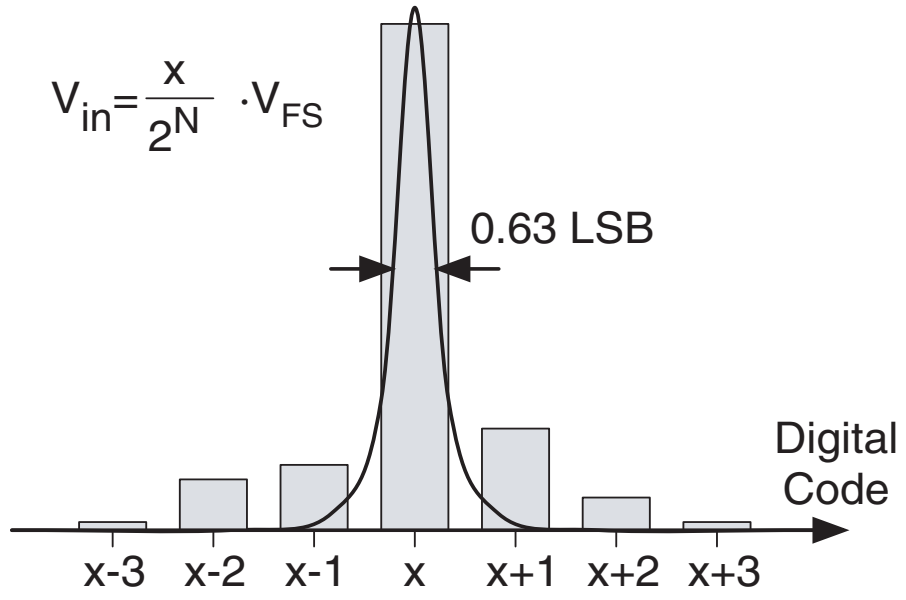


Figure 2.11: Estimation of the input referred noise using the histogram method for dc input (Figure 2.11 [5])

2.3.5 Signal to Noise Ratio (SNR)

is the ratio between the power of the signal (normally a sinewave) and the total noise produced by quantization and the noise of the circuit. The SNR accounts for the noise in the entire Nyquist interval. The SNR can depend on the frequency of the input signal and it decreases proportional to the input amplitude. Figure 2.12shows the SNR of a hypothetical 12-bit data converter with 50 MHz sampling frequency. The SNR for a −0.5 dB input is 67 dB. The loss in the SNR shows that the noise caused by the electronics is larger than the quantization noise. When the input signal is −20 dB then, as expected, the SNR is 48 dBs. Observe that the SNR performances versus frequency are good: the SNR is almost constant in the entire Nyquist range. Also, it only drops a few dB for frequencies in the second Nyquist zone. Therefore, the hypothetical converter of figure 2.12 is suitable for under-sampling a signal whose spectrum is in the second Nyquist zone.

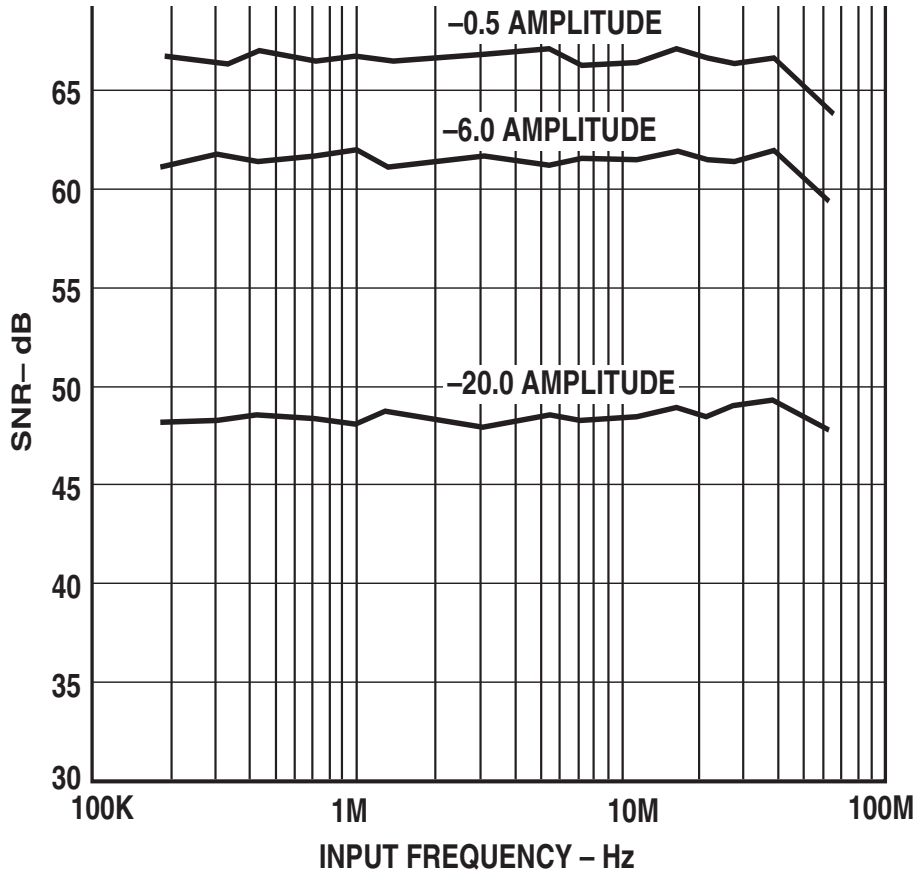


Figure 2.12: Example for hypothetical ADC SNR values for different input signal levels [5]

2.3.6 Signal to Noise and Distortion Ration (SNDR)

Is similar in definition to the SNR except that nonlinear distortion terms are counted for. Usually and input sine wave can easily spot the nonlinear effect. The SINAD is the ratio between the root mean square of the signal and the root sum square of the harmonic componentss plus noise (dc component is not considered). Since static and dynamic limitations cause a nonlinear response, the SINAD is dependent on both the amplitude and frequency of the input sine wave. Figure 2.13 shows the SNDR of a hypothetical 12-bit data converter with 50 MHz sampling frequency. The SNDR for a -0.5 dB input is 67 dB. The loss in the SNDR shows that the noise caused by the electronics is larger than the quantization noise. When the input signal is -20 dB then, as expected, the SNDR is 48 dBs. It is good to note that the SNDR performances are good (almost constant) for the Nyquist range. It is also shown that the harmonic terms in the SNDR are negligible if the input is -20 dB_{FS}or less. Larger input amplitudes bring about distortion especially at high

frequencies. Notice that the SNDR significantly degrades in the second Nyquist zone.

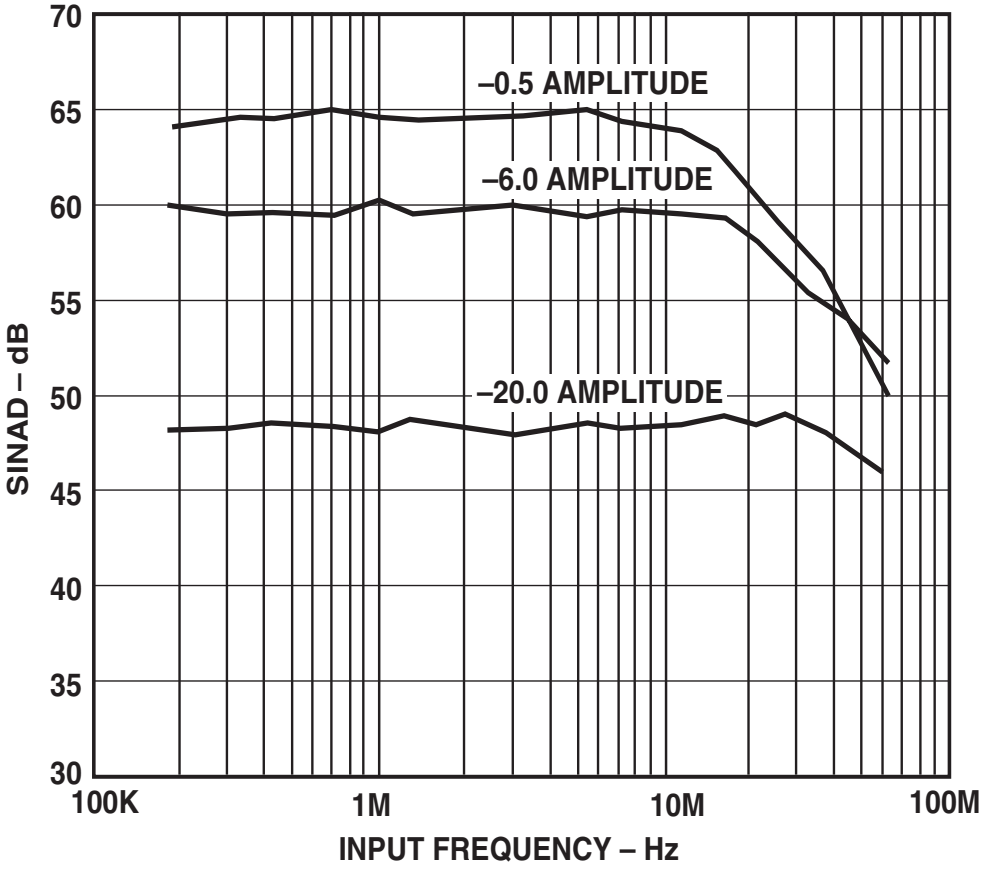


Figure 2.13: Signal to noise and distortion ratio versus input amplitude and input frequency (f_{ck}=50 MHz) [5]

2.3.7 Dynamic range

Is the input amplitude at which the SNR (or SNDR) is 0 dB. Usually it is useful for the types of ADC that does provide information about the SNR (or SNDR) at 0 dB_{FS}(like sigma-delta converters). Figure 2.14 shows a typical plot of the SNR versus the input amplitude for a sigma-delta ADC. The dynamic range is amost 80db while the peak SNR is at -6 dB_{FS} of the full scale.

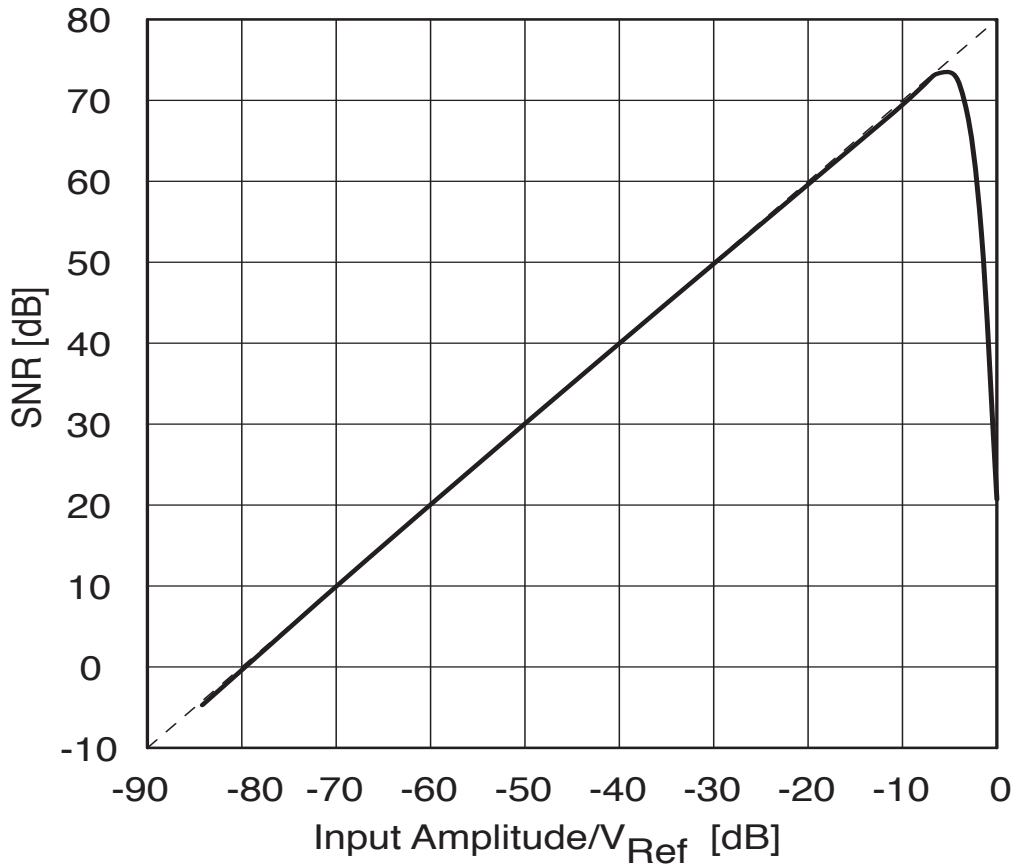


Figure 2.14: example of SNR for segma-delta converter versus input amplitude, reference voltage-ratio [5]

2.3.8 Effective Number Of Bits (ENOB)

This measure is a mirror to the Signal to Noise Ratio (SNR). It presents how much the system can deliver information about the input signal. As noise has many sources and is affected by many measures, we will focus only on the error from the quantization noise, Signal to Quantization Noise Ratio (SQNR) [5].

As shown in figure 2.4, the quantization noise is in the range of +-LSB, then the probability distribution function can be expressed as:

$$\begin{aligned} P(\varepsilon_q) &= \frac{1}{\triangle}, for\ -\frac{\triangle}{2} < \varepsilon_q < \frac{\triangle}{2} \\ P(\varepsilon_q) &= 0 \quad , otherwise \end{aligned} \tag{2.5}$$

As $\varepsilon_q$presents the error, and $\triangle$ presents the maximum error, which equals one LSB in our case.

To calculate quantization noise power, an average integration for ε_q should be performed as

$$P_Q = \int_{-\infty}^{\infty} \varepsilon_q^2 . P(\varepsilon_q)\, d\varepsilon_q = \int_{-\triangle/2}^{\triangle/2} \frac{\varepsilon_q^2}{\triangle}\, d\varepsilon_q = \frac{\triangle^2}{12} \tag{2.6}$$

This relation indicates that the number of bits for the system affects the quantization noise; as the number of bits increases, the quantization noise decreases, as $\triangle$ is decreased too.

Assuming an input sinewave signal of maximum amplitude A_{mx}, the signal power can be expressed as

$$P_{sin} = \frac{1}{T} \int_0^T \frac{A_{mx}^2}{4} sin^2(2.\pi.f.t)\, dt \;=\; \frac{A_{mx}^2}{8} \;=\; \frac{(\triangle.2^n)^2}{8} \tag{2.7}$$

As f is the sinewave operating frequency, n is the number of bits and T is the signal period.

From eq.2.7 and eq.2.6, the ratio between the signal and quantization noise power unveils:

$$SQNR_{sine,db} \;=\; (6.02.n + 1.78) \tag{2.8}$$

eq.2.8 presents a quick and useful relation between the ideal number of bits for an ideal ADC that can presents the given SQNR (or ideal SNR) of the system.

Estimating the ideal number of bits, or in other words ENOB, indicates how far the target ADC system is from the ideal one. When the SQNR is replaced by the general SNR, which counts for the quantization noise and all the noise in the system, when n presents the ENOB, eq.2.8 can be re-phrased as

$$ENOB_{sine} = \frac{SNR_{tot,db} - 1.78}{6.02} \tag{2.9}$$

This relation is a good and accurate estimate for sinewave ENOB value when SNR is known as in section A.2.

%

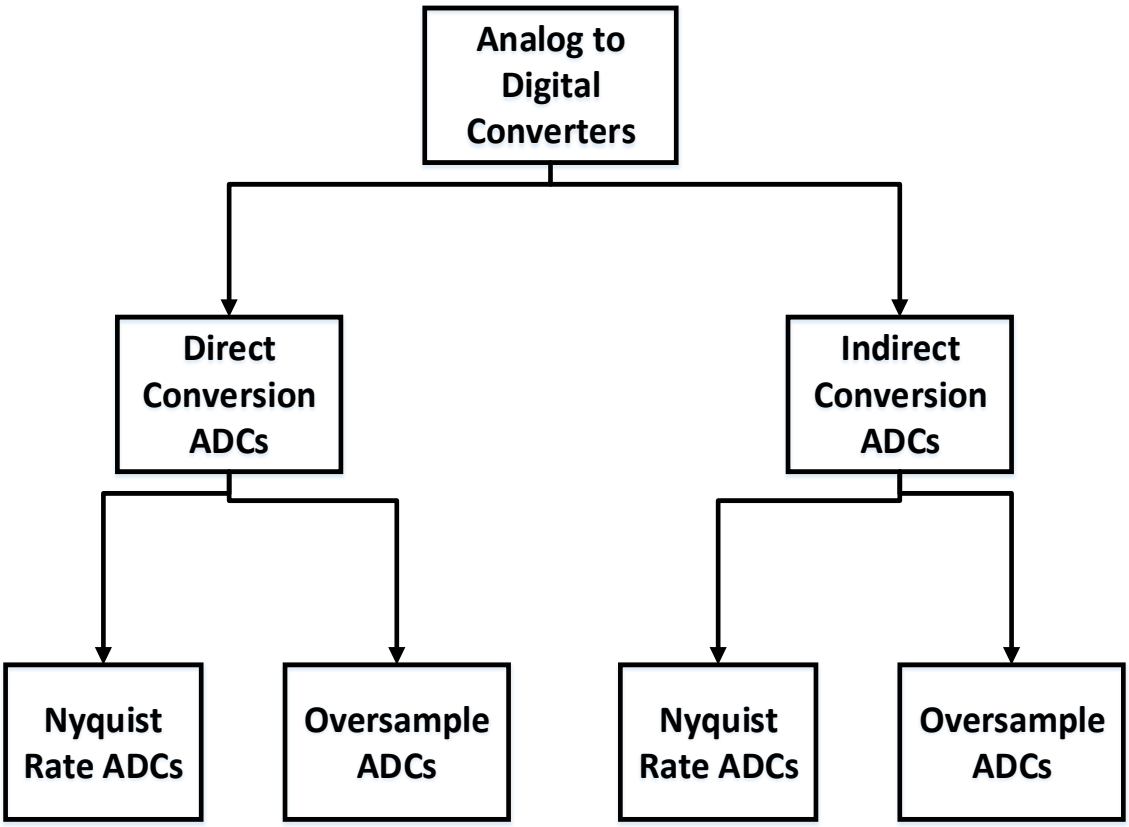


Figure 2.15: ADC types

2.4 Types of Analog to Digital Converters (ADC)

Analog to digital converters are divided in to two categories, direct conversion ADCs and indirect conversion ADCs. A direct conversion ADC converts the input analog voltage to the corresponding digital representation directly in one step. However, an indirect conversion ADC converts the input analog voltage to an intermediate form presented as a change in the frequency or characteristic in a pulse like the pulse position or width.

Each category can be divided into two sub-categories. A direct conversion or an indirect conversion ADC can be a Nyquist rate ADC or and oversampling ADC. In Nyquist rate ADCs, the input analog voltage is sampled at a rate higher, but close to, than double the input maximum frequency, which is called the Nyquist

rate. In contrast, oversampling ADCs samples the input signal at a rate much higher than the Nyquist rate, 4-10 times.

Oversampling ADCs can reject more noise from the band of operation so it can deliver more SNR. However, the high sampling requirement, relative to the input signal rates, makes it best fitted to the applications when high resolution is required with acceptable low sampling rates. On the other hand, Nyquist rate ADCs can deliver higher sampling rates in comparison to the oversampling ADCs.

Successive Approximation Register ADC (SAR-ADC) and flash ADC are examples of the Nyquist rate direct conversion ADC. Sigma-Delta modulator ADC is an example for oversampling direct conversion ADCs. Dual slope ADC is an example of Nyquist rate-indirect conversion ADCs. Voltage controlled oscillator ADC is an example of oversampling-indirect conversion ADCs [1]. Next, some examples of ADC types are presented.

2.4.1 Nyquist rate ADCs

2.4.1.1 flash ADC

Flash ADC is one of the fastest types of ADC [19]. In this type of ADC, the analog voltage input sample is compared with $2^D - 1$ reference values using $2^D - 1$ comparators, as D is the ADC number of bits. As we can see from figure 2.16, the reference voltages are generated using a resistive divider with 2^D-1 resistors. Each reference voltage is one LSB greater than the reference voltage immediately below it. Each reference voltage is connected to one of the two inputs of each comparator, while the other input is connected to the analog voltage input sample. Each comparator produces a "1" when the analog input voltage sample is higher than the reference voltage connected to it. Otherwise, the comparator produces "0". The comparators produce thermometer code. The thermometer code is then decoded to the appropriate digital output code.

It can be noticed that this type of ADCs requires big circuit and high power consumption. For each step, one LSB, there should be a comparator and a resistor. This drawback makes flash converters typically impractical for resolution greater than 8 bits (255 comparators). Moreover, the large number of comparators connected to input voltage results in a large parasitic capacitance that load the input terminal and limit the speed of the converter and requires a power-hungry buffer at the input terminal

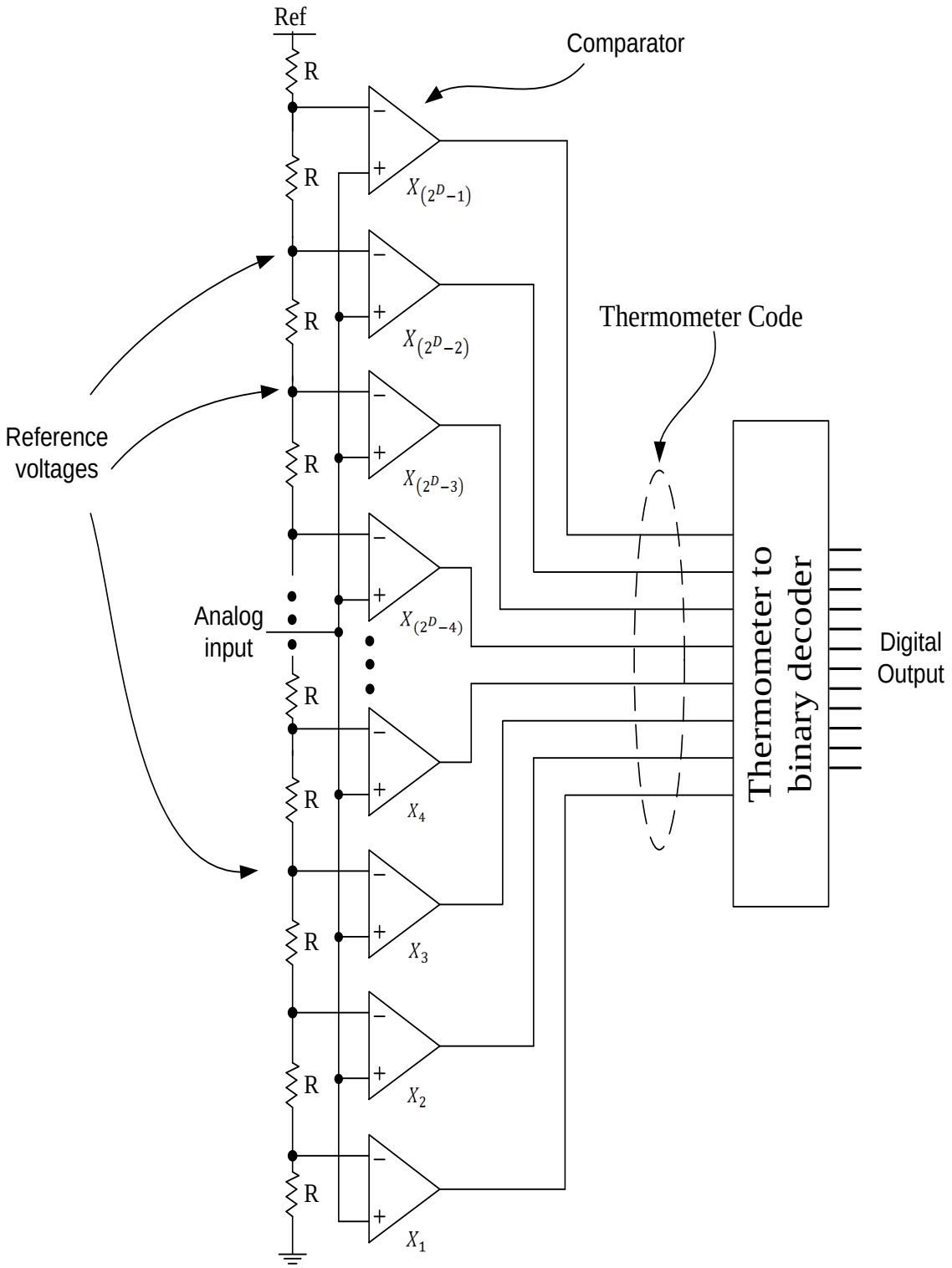


Figure 2.16: Flash ADC [1]

2.4.1.2 Piplelined ADC

Pipelined ADC divides the conversion job into simple tasks which can be pipelined in many stages. Usually, each stage resolves for one or few digital bits. Each stage can be a small ADC of the flash type. The architecture throughput is restricted by the delay of one stage only (maximum stage delay), hence, the throughput is comparable to the flash ADC sampling. However, a typical pipelined architecture consumes less power compared to the corresponding flash ADC.

As shown in figure 2.17, pipelined ADC consists in general of K stages. Each stage, except the last stage, consists of n bits flash ADC, n bits DAC, S&H, subtractor, and amplifier. The last stage is a flash ADC. In the first stage the S&H block samples and hold the analog input. Then, this input is fed to the n-bits flash ADC to convert it to n-bit code. Afterwards, the n-bits are fed to n-bits DAC to get intermediate analog voltage. Then, the intermediate analog voltage is subtracted from the original analog input to this stage to produce a residual voltage. This residual voltage is then amplified to the double and is fed to the next stage to get the next n-bits resolution and so on. While stage number two is working on the first sample, stage number one samples new analog input; this pipelining is the main reason for the high throughput. Finally, all the N-bits have to be aligned because they were generated at different times [1, 20].

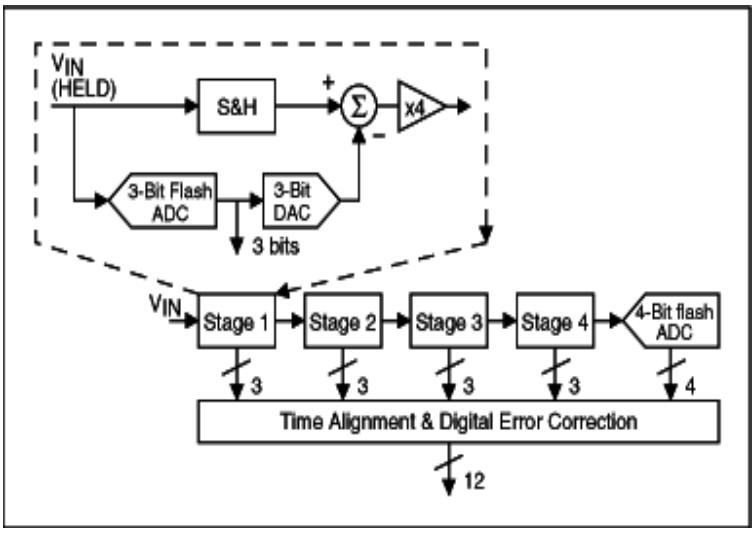


Figure 2.17: Pipline ADC [6]

2.4.1.3 Successive approximation ADC

Successive Approximation Register (SAR-ADC) depends on the successive comparison of the input quantity to the binary weights of the digital representation [20]. Figure 2.18 shows the basic SAR ADC. It consists of a comparator, DAC, and Successive Approximation Register (SAR) logic block. SAR-ADC takes an iterative approach to determine the input voltage. One bit is calculated for each iteration starting from the Most Significant Bit (MSB). First, the Most Significant Bit (MSB) in SAR starts with one "logic 1". The DAC converts this value to its corresponding voltage, which is the mid-scale voltage, then the comparator compares the input voltage with the reference voltage. If the comparator produces zero, input is smaller than the reference voltage, then the MSB will be erased. Otherwise, the MSB will remain one. After that, the next bit to the MSB will be one and the whole process is repeated, until all bits in SAR are known and the End Of Conversion (EOC) signal becomes one [1].

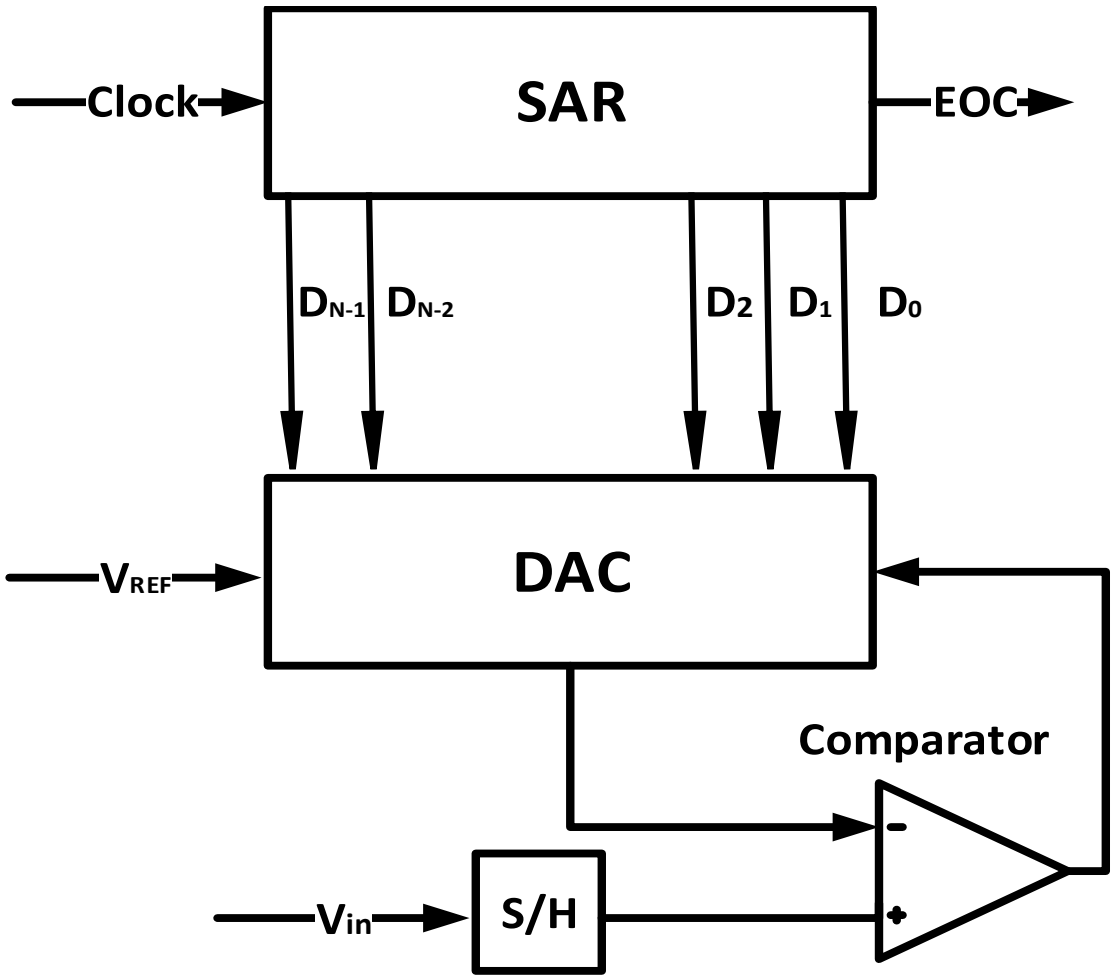


Figure 2.18: Basic circuit diagram of the successive approximation algorithm

For a given dynamic range 0 – VFS the MSB distinguishes between input signals that are below or above the limit VFS/2. Therefore, comparing the sampled input with VFS/2 obtains the first MSB bit as illustrated by the timing scheme of Figure 2.19 (a). The knowledge of the MSB restricts the search for the next bit to either the upper or lower half of the 0 – VFS interval. Consequently, the threshold for determining the second bit is either VFS/4 or (as it is for the case of the figure) 3VFS/4. After this, a new threshold is chosen and the next bit can be estimated. The timing diagram of figure 2.19 describes the operation for three bits but, obviously, the search can continue for additional clock cycles to determine more bits. The voltages used for the comparisons are generated by a DAC under the control of a logic system known as the successive approximation register (SAR) as shown in figure 2.19 (b). Notice that the input common mode range of the comparator must equal the dynamic range of the converter [5].

The method uses one clock period for the S&H and one clock period for the determination of every bit thus requiring (n + 1) clock intervals for an n-bit conversion. Sometimes, if the S&H settling period is significantly longer than the time required for each comparison, then it can be convenient to use two clock periods for the sampling and one per every bit totaling (n + 2) clock intervals for an n-bit conversion.

2.4.2 Oversampling ADCs

Oversampling conversion technique have become popular as it avoids many of the difficulties encountered with conventional method for analog-to-digital conversion, such as the use of anti-aliasing analog filters [21].

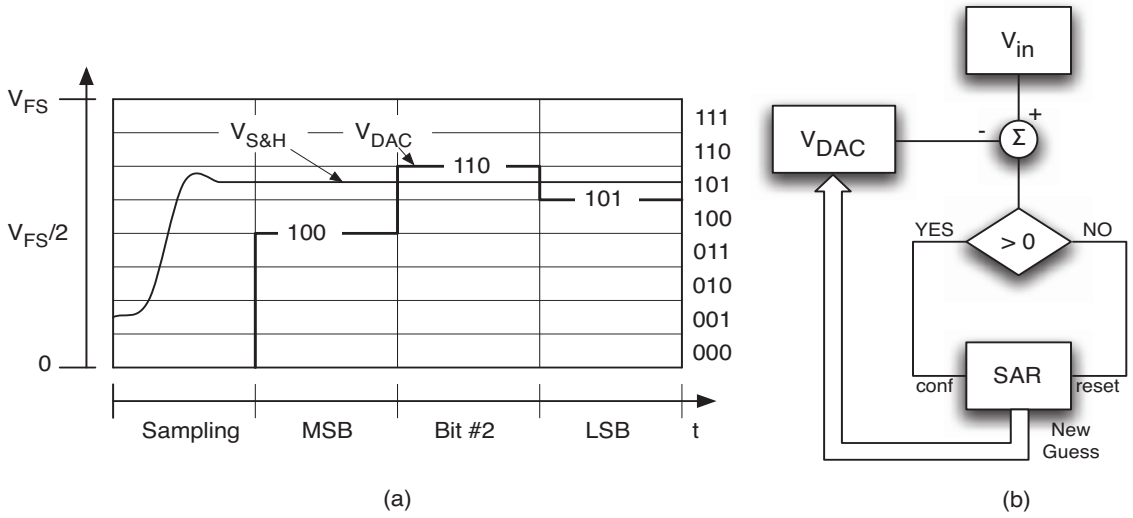


Figure 2.19: Timing (a) and flow diagram (b) of the successive approximation technique

2.4.2.1 Sigma Delta ADCs

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 [21]. Sigma-delta modulator modulates the analog signal into a digital code, usually single-bit code, at a frequency much higher than the Nyquist rate. The use of high frequency modulation and demodulation eliminates the need for sharp cutoffs in the analog anti-aliasing filter at the input of the ADC. Figure 2.20 shows the simplest sigma-delta modulator, the first-order sigma-delta modulator. The input to the circuit

feeds to the quantizer via an integrator, and the quantized output is fed back to be subtracted from the input signal [22].

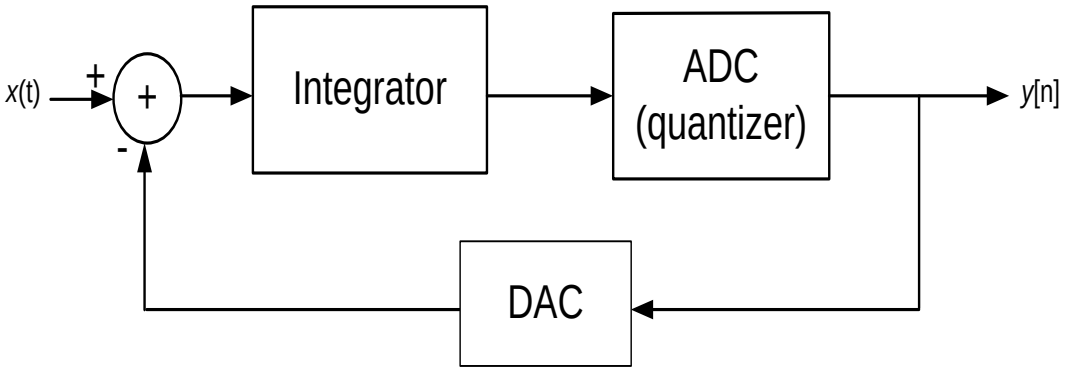


Figure 2.20: First-order sigma-delta modulator

2.5 Time-based ADC (TADC)

There is a global need to convert the analog designs with the digital counter parts. One of the many reasons is the noise resistance and the flexibility to move from one technology to another, easily. Also, it is becoming an important target for Systems On Chip (SOC) designs to consume less power and chip area for various applications. This increases the need for Ultra-Wide-Band (UWB) ADCs with higher sampling rates in which we can push more analog blocks towards the digital domain. Time Based ADC is a special kind of data converters in which the input voltage is first transformed into intermediate change in frequency, pulse position or pulse width using Voltage to Time Converter (VTC) circuit, and then Time to Digital Converter (TDC) circuits follows by making the rest of the conversion to the corresponding digital representation.

2.5.1 TADC based on frequency modulation

In this type, the voltage is converted to change in frequency of an oscillator. In this type the VTC is called VFC converter. The TDC then converts the modulated frequency to the corresponding digital words by a mean that detects the rate of frequency change. A simple criterion is to make a counter counts the number of positive edges of the oscillating signal in the sample period. One major disadvantages of this type is the power consumption.

2.5.2 TADC based on pulse position modulation

It is very similar to the TADC based on frequency modulation except that in the pulse position-based converters use the pulse position inside the sample period as the time signal. It is rarly used though because it is very similar in design to the TADCs based on pulse width modulation.

2.5.3 TADC based on pulse width modulation

TADCs based on the pulse width modulation are the most popular type of the time based ADCs and is the target for this work. In this type a VTC is used to convert the signal voltage to changes in pulse width of a reference signal. In the second block, the TDC, the pulse width change is converted to the corresponding digital words. Many designs for VTC and TDC of this type exists. Examples of each part follow.

2.5.3.1 Pulse-width VTC examples

Dual slope VTC Figure 2.21 illustrates dual slope TADC. Dual slope ADC consists of an integrator, comparator, counter, and control logic. First, the analog input,V_{in}, is integrated to produce the output voltage,Vout, in the first part of the clock cycle, with T_1period2.10. V_{out} value depends on the analog input voltage as in equation. After time T_1, the switch converts source from $-V_{in}$ to V_{ref} and at this point the discharging starts until the Vout reach zero voltage (detected by the comparator). The discharging time depends on Vout which depends on V_{in}. Then, the discharging time will depend on Vin as in 2.13.

$$V_{out} = \int_0^{T_1} -\frac{-V_{in}}{RC} dt = \frac{V_{in} t}{RC}|_{0 \to T_1} \tag{2.10}$$

$$V_{out} = \frac{V_{in} T_1}{RC} \tag{2.11}$$

$$V_{out} = -\int_{T_1}^{t} \frac{V_{ref}}{RC} dt + \frac{V_{in} T_1}{RC} \tag{2.12}$$

$$V_{out} = \frac{V_{ref}(T_1 - T_2)}{RC} + \frac{V_{in} T_1}{RC} \tag{2.13}$$

The discharging will continue until V_{out} becomes zero. Substituting V*out* by zero in 2.13 leads to equation 2.14 where T2 = t−T1.

$$T_2 = \frac{V_{in} T_1}{V_{ref}} \tag{2.14}$$

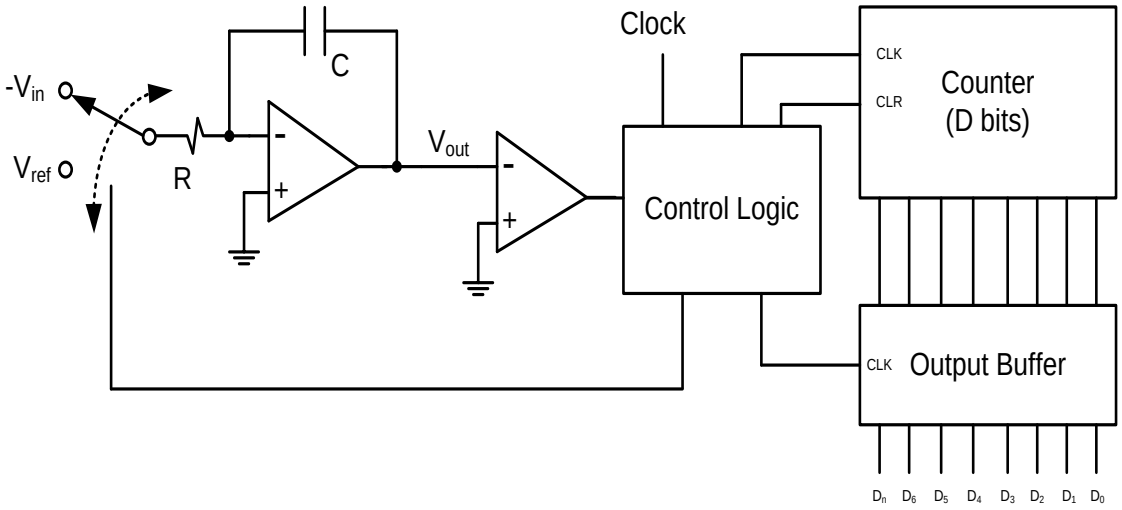


Figure 2.21: Dual Slope VTC [7]

The generated pulse width depends on the input signal. The counter in the second part of the system (counter and the output buffer) is used to converts the output pulse to the corresponding digital words performing the operation of a TDC. The designer can add the TDC of his choice instead. Dual slope TB-ADC is used in high resolution requirements, but in applications with low data rates. It has small offset and gain errors comparable to other ADC types [7]

Current starved-based VTC In figure 2.22, the input voltage, Vin, controls the delay of the falling edge of the clock signal, Vclk, through the inverter (Transistors M1 and M2) by controlling the discharging current of transistor M3 [23]. Increasing V_{in}increases the discharging current from $C_L$and reduces the negative edge transaction delay and vice versa. The main drawback of this scheme is that it controls only the discharging current and the negative edge delay only. In [24] a new design to control both the positive and negative edges is introduced. The design consists of pull up and pull down networks with an XNOR gate are used for better linearity.

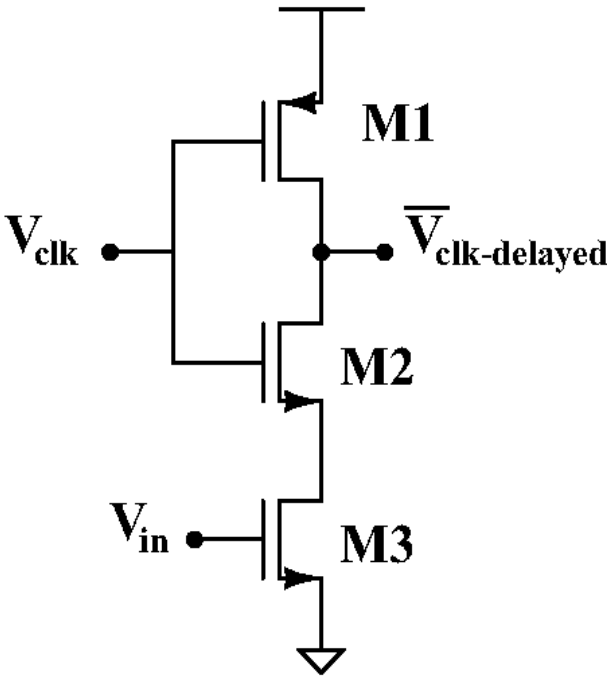


Figure 2.22: Current Starved VTC

2.5.3.2 Pulse-width TDC

Vernier Delay line-based TDC Delay line based TDC is a widely used approach for high speed time to digital conversion ([8]). 2.23 presents the main block diagram. It consists of 2^N stages, as N is the number of target bits. Each stage consists of a DFF with 2 buffers of $\tau d1$ and $\tau d2$ delays. TDC Delay line based TDC can measure a time interval between two events where the first event is marked by a Start signal and the second event is marked by a Stop signal (the target pulse is first converted into two pulses the positive edge of each one presents an edge transaction; Start and Stop signal positive edges presents the positive and negative edges of the original signal respectively). The Start signal propagates through series of buffers (delay line), each buffer delay the start signal by $\tau d2$ .Similarly, the Stop signal propagates through another delay line, each buffer delay the stop signal by $\tau d1$. The Stop signal samples the delayed version of the Start signal after each buffer. D-FF output is a thermometer code where each D-FF out is logic one as long as the Start signal leads the stop signal at the target DFF or logic zero otherwise (thermometer code). The signals traveling through each stage suffers from a phase difference in the direction that promotes the Stop signal to lead the Start signal. The difference is τd_1-τd_2. The output thermometer code is then converted to binary code through a thermometer to binary encoder ([8]).

This type of TDC is a good candidate for high resolution conversion because the resolution achieved can be smaller than the delay of an inverter. However, this comes at the cost of the area an power consumption which is not affordable for low power applications.

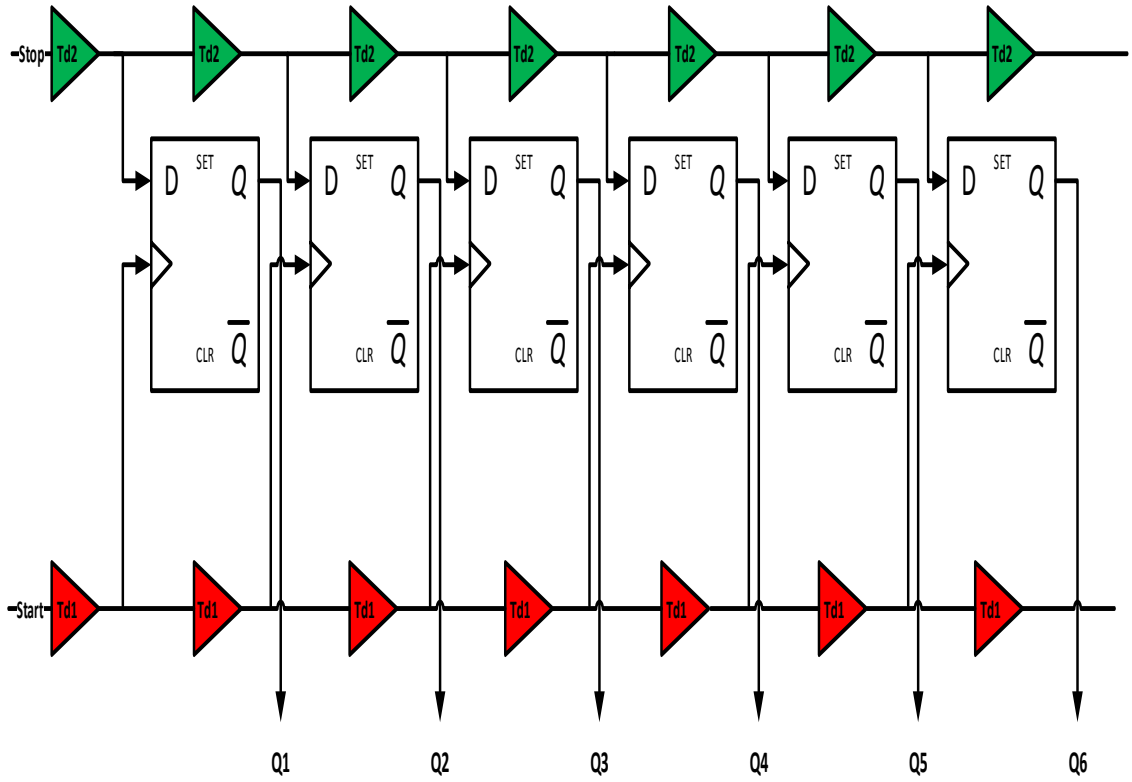


Figure 2.23: Vernier delay line-based TDC [8] [9]

Successive Approximation-based TDC Successive Approximation (SA) is a favorable alternative for low power applications. The operation is based on comparing the input quantity to the reference binary weighted quantities to directly calculate the digital bit (in each word). A full sample is resolved be iterating all the bits for each word. Each of a different binary weight.

In [18], a TDC circuit design based on successive approximation is introduced. The un-folded version of the design is presented in figure 2.24. It consists of a buffer of $T_{fs}/2$ (half the full scall time) delay followed by N_bits (number of bits) stages of the same type except the reference time to compare with ($T_{fs}/2^i$,as $i := 1 : N$). It is required to convert the analog phase difference between the Start and Stop signal.

Figure figure 2.25 presents the basic cell architecture (a) and the timing diagram (b) of the inputs and outputs for a general stage N. The decision works on the positive edge of the input signal (Start signal) and compares it with the postive edge of the reference signal (Stop signal). The positive edge of the reference signal is assumed to be shifted with the $Tfs/2^N$.The DFF checks if the input signal still leads the reference signal. If condition is valid and the input leads the reference pulse, the input pulse is delayed with an amount equivalent to DnRef as the reference pulse is always delayed with an amount of $D_{nin} = Tfs/2^{N+1}$. If the condition is false, then the input signal is not delayed and the difference between the input and referrence pulse is shortened by delaying the reference pulse and keeping the input pulse with the same phase. A multiplexer is used to select between the original input signal or the delayed version depending on the comparison result.

The main problem in this architecture is the propagation delays for the different components in the cell. The time signal imposes the need to insert delay elements to compensate for the propagation delay of each component (figure 2.26). The condition in the circuit path decides which analog positive edge to pass (for the input signal) making the synchronization difficult as the analog phase difference between the input and reference pulse depends on the condition result!. It will be shown in this work how this dependency is moved from the analog domain to the digital domain and the analog signal path is not affected by the condition result.

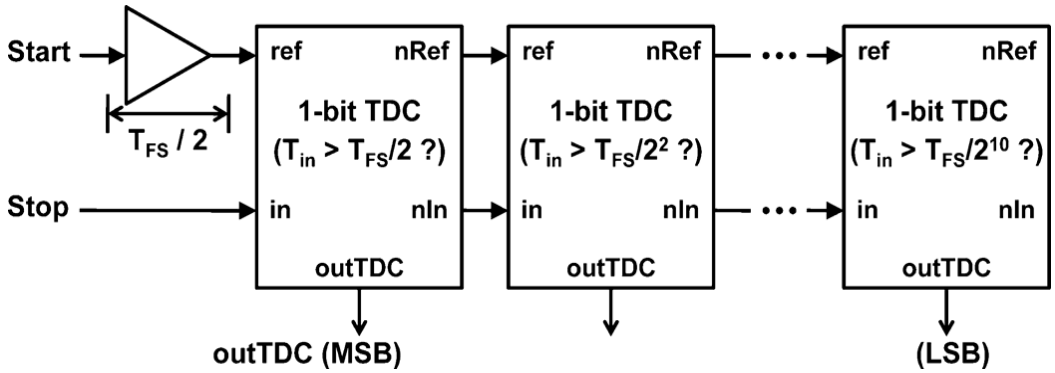


Figure 2.24: Decision select SA TDC circuit

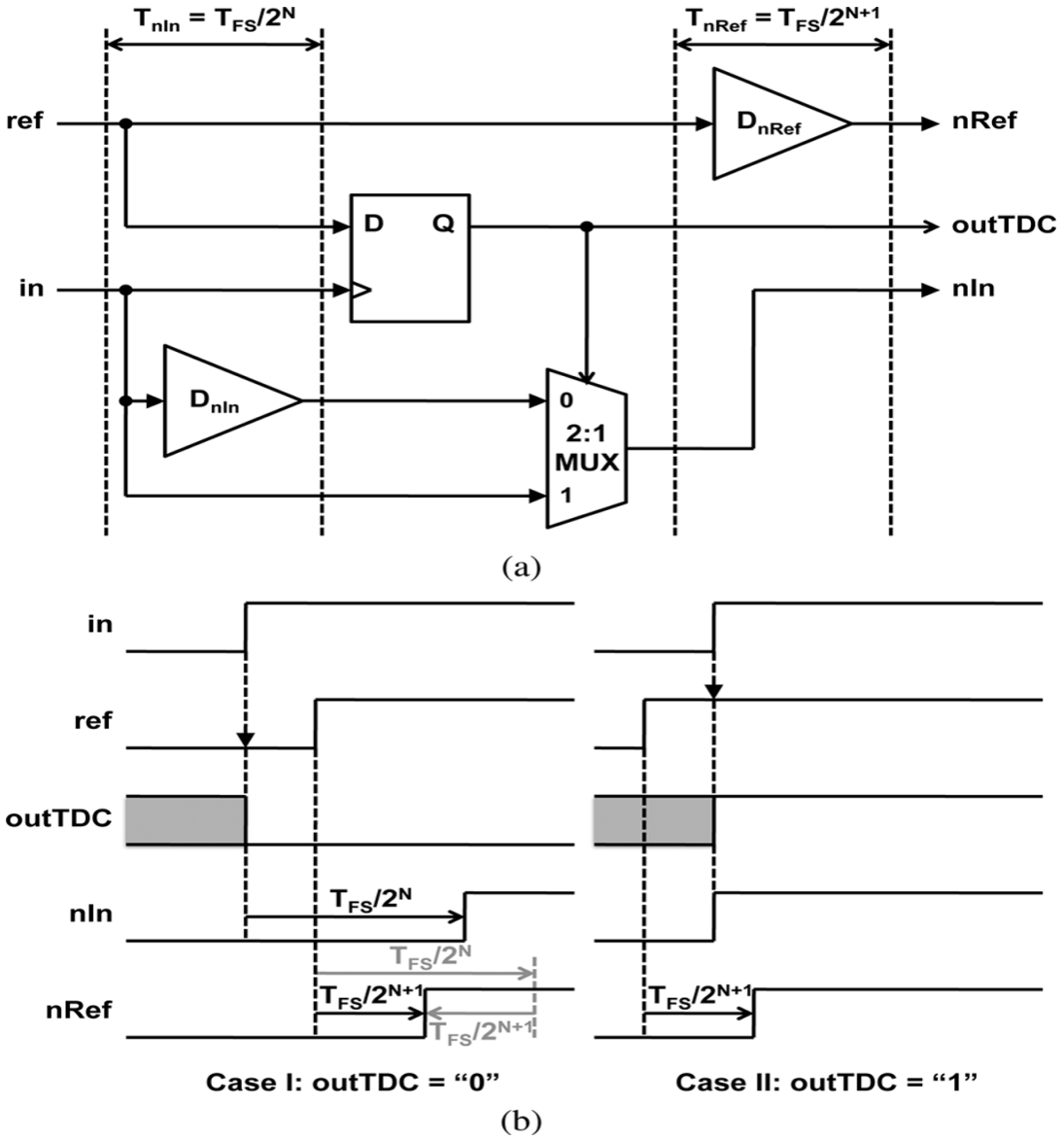


Figure 2.25: Decision select SA TDC cell (a) and timing diagram (b)

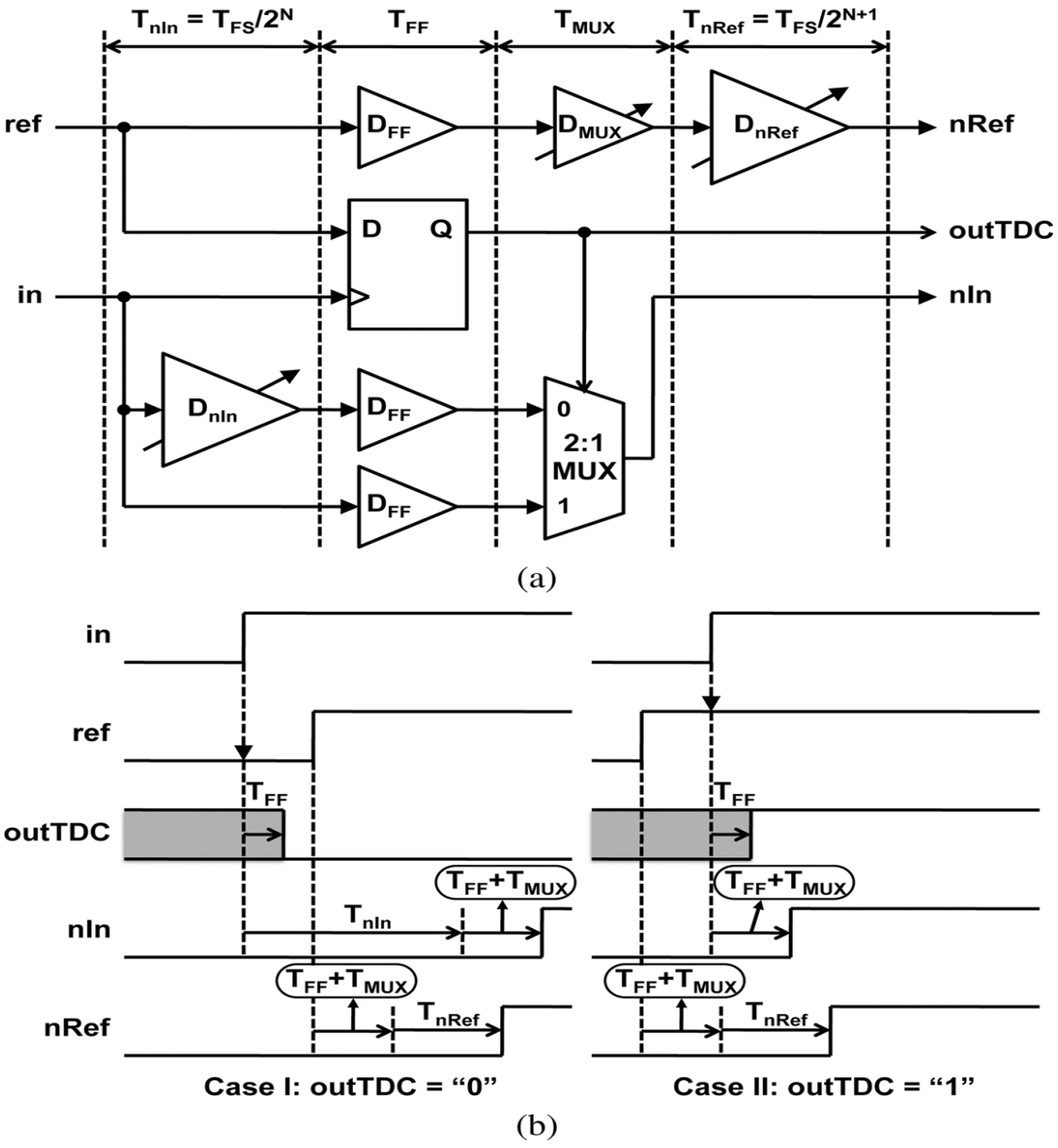


Figure 2.26: Compensation delays for decession-selection SA circuit

Chapter 3

Introducing SAR-CD algorithm

Implementations based on binary searching algorithms take place in applications that require low power and area with high resolution. In [17, 18], the well-known Successive Approximation Register (SAR) is applied in a cyclic manner. However, the existence of two signal branches imposes a critical condition to keep the signals synchronized along the conversion, and error in one branch is accumulated in each iteration. This error accumulation becomes worse due to the conditioning on each path of the two signal paths. In this work a new algorithm is introduced to move this dependency from the time domain to the digital domain. In addition, the algorithm needs only one signal path instead of two which relaxes the strict synchronization requirements.

3.1 SAR verses SAR-CD algorithm

In the conventional SAR algorithm, for a given decimal value, starting from the MSB and for each successive bit, the weight of the next bit is subtracted from the current input value. If the result is a positive number, then the digital bit is '1', otherwise it is '0'. For example, when converting the decimal number '9' to 4 binary weighted digits (i.e., 8, 4, 2, and 1). Starting from the MSB, the weight 8 is subtracted from the input 9. As the result is positive (i.e., 9-8 = +1), then the MSB is '1' and the subtraction result (i.e., +1) is the next iteration input. The second iteration weight 4 is subtracted from +1. As the result is the negative value -3, then the second bit is '0' and the input 1 is passed to the next iteration. After that, the

third iteration weight 2 is subtracted from the input +1, and etc. This process is valid as long as the passed decimal value is the result of a successful subtraction (i.e., a positive value).

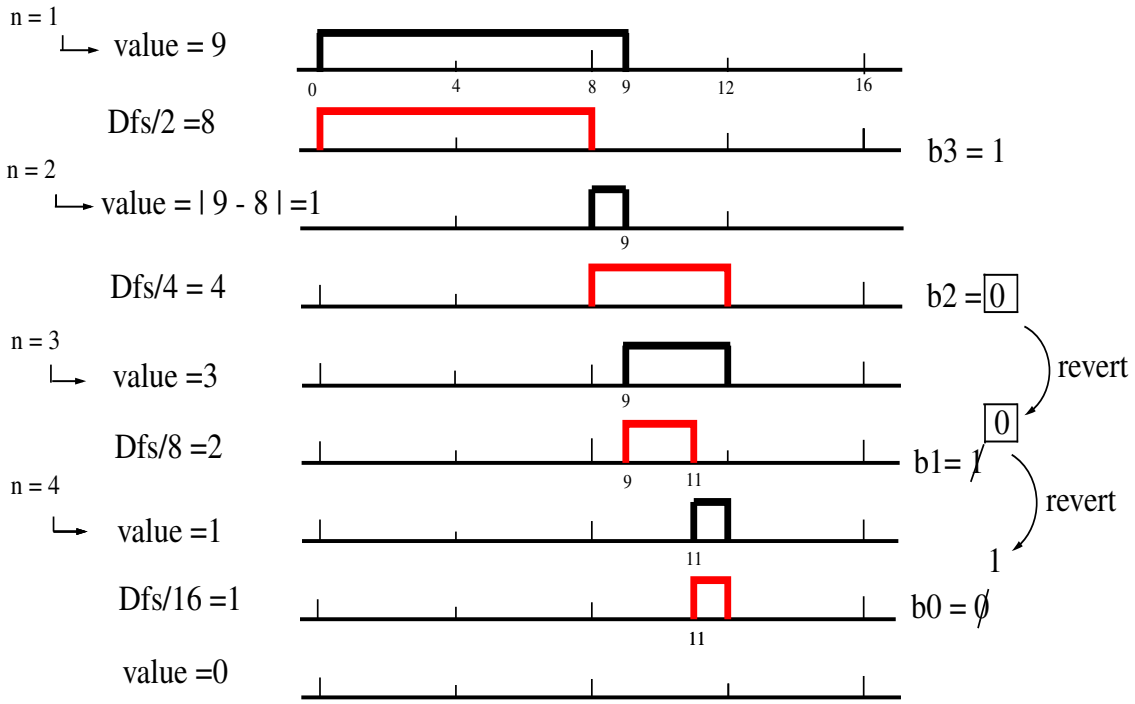


Figure 3.1: Evaluation of the binary "b3 b2 b1 b0" for the number ’9’ using the new algorithm

In this conventional SAR algorithm, the result of the unsuccessful subtraction (i.e., -3) is neglected, and in particular the modulus of this result (i.e., +3). This modulus value holds all the information needed to evaluate the next bits. In the new proposed SAR-CD algorithm, the input to each stage is the absolute difference between the input and the weight of the previous stage. Also, the value of each bit is used to correct the value of the next one.

In the previous example, the absolute difference between the second stage input +1 and the weight 4 (i.e., +3) is the input to the third stage weighted 2. This modulus value is used to evaluate the third bit. As 3 is greater than the weight 2, then the third bit is initially set to 1. However, this value should be inverted because the previous bit (second bit) is ’0’. This inverts the third bit value from ’0’ to ’1’. The full example to convert 9 to the binary form “b3 b2 b1 b0” using the new SAR-CD algorithm is shown in figure 3.1.

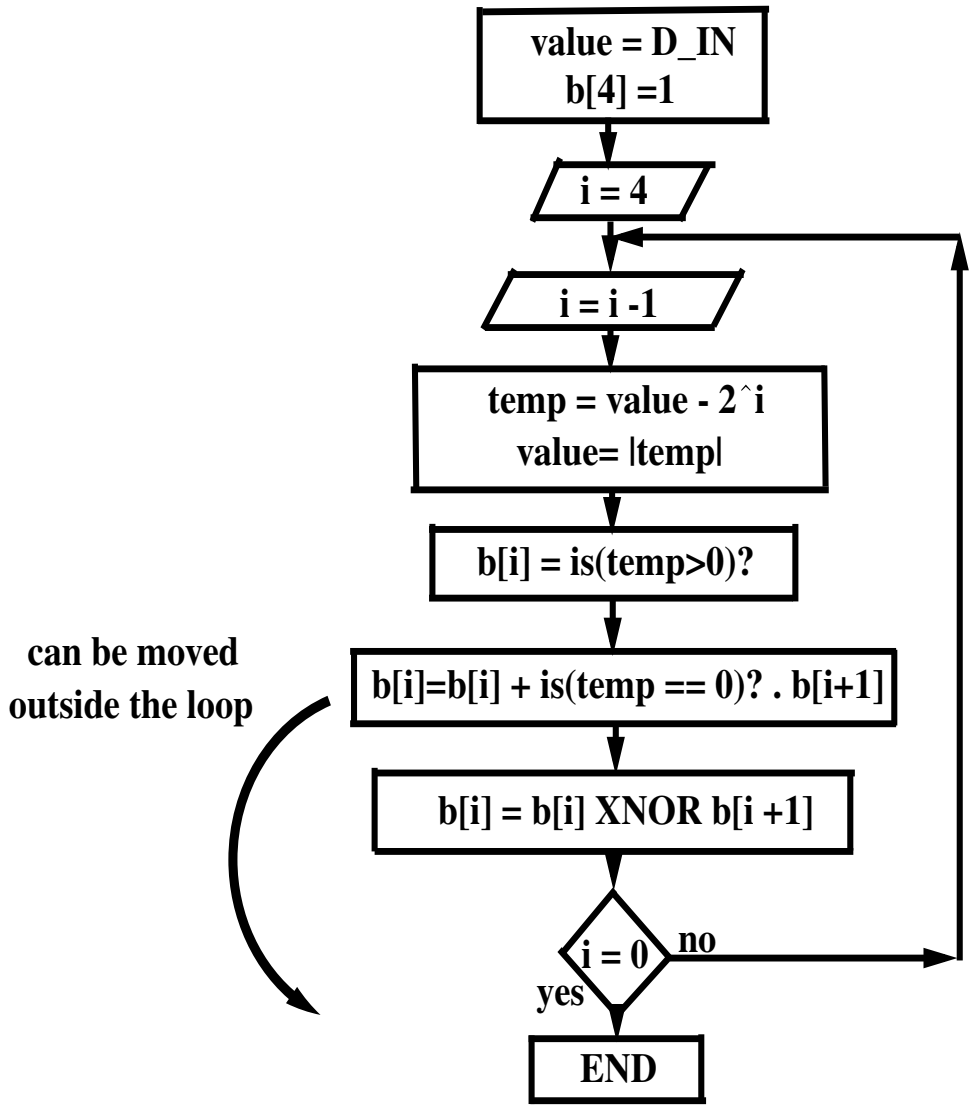


Figure 3.2: Flow chart for the proposed algorithm

The variable “value” is initially set to 9. As 9 is bigger than 8, then b3=1. For the next iteration, “value” equals abs(9-8). In general, for each successive stage, the input equals abs(value - Dfs/(2^n)), where Dfs is the full scale value (i.e., 16 for the 4-bit case) and ’n’ ranges from 1 to 4 iterating all the 4 stages. The stage output (i.e., b3, b2, b1, and b0) is initiated to ’1’ when the input is greater than the stage weight or ’0’ otherwise. However, this value is inverted if the previous bit is ’0’ or kept unchanged otherwise. Bits “b1” and “b0” are inverted after the subtraction (value - Dfs/(2^n)) because their previous bits (i.e., b2 and b1) equal ’0’ after correction, respectively.

3.2 SAR-CD general algorithm

The new proposed SAR-CD algorithm flowchart is depicted in figure 3.2. The digital output to compute is “b[3] b[2] b[1] b[0]” as ’b[3]’ is the MSB and ’b[0]’ is the LSB. The input number “D_IN” is assigned to the variable “value”. For each iteration, “value” is updated with the absolute difference between the old value and the loop reference weight (2^i). The bit value, in each iteration, depends on the sign of the result and the previous bit. ’b[i]’ is initially ’1’ if the difference is bigger than ’0’ (’is(temp>0)’ is true) or ’0’ otherwise (’is(temp>0)’ is false). The equality check depends on the previous bit (“is(temp==0)?.b[i+1]”). As ’+’ is arithmetic OR operation and ’.’ is arithmetic AND operation. The process ends when “i” equals ’0’. ’b[4]’ is initiated to ’1’ so that bits correction does not change ’b[3]’ value.

It should be noted that in the proposed SAR-CD algorithm, the new value is passed to the next iteration, denoted by "value", is independent of the current bit evaluation. Also, the bit correction (i.e., which is implemented by using an XOR operation) is a simple digital logic operation. Fortunately, this bit correction can be performed after the pulse dis-assembly (to a separate loop in figure 3.2). The new proposed SAR-CD algorithm is very useful for Time-Based Analog to Digital converters because the parameter “value” is an analog quantity. Analog quantities manipulation is prone to errors and may cost extra circuitry (e.g DAC).

In this work, two case study circuits are proposed to adopt the new SAR-CD algorithm to digitize the input variable pulse widths. The digital number corresponding to the pulse width is computed by comparing the input pulse to the full scale reference pulses, for example, pulses of width Pfs/2, Pfs/4, Pfs/8...., as Pfs is the full scale pulse. The equivalence of the abs(value – 2^i) is a simple XOR operation between the 2 pulses. The output is directly passed to the next iteration irrespective of the currently evaluated bit. In general, power-sensitive applications like in [25] exhibit significant reduction in the SAR-CD logic implementation complexity compared to the conventional SAR. This is because in the SAR-CD algorithm, most of the conditioning needed to produce the reference voltage is removed. Absolute comparison is simpler in implementation than the conventional SAR algorithm. For example, designs based on comparing voltage values held on two capacitors would need no capacitor discharging before each bit evaluation. This increases the maximum supported frequency and minimize the error from discharging circuits.

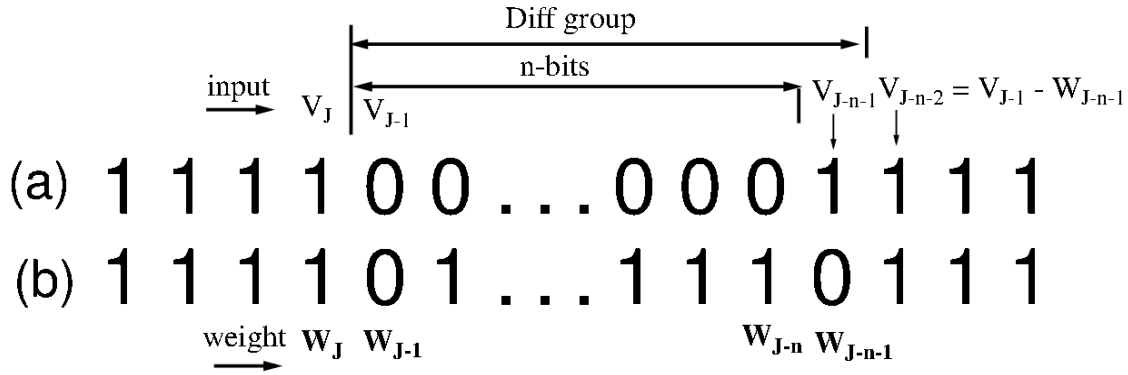


Figure 3.3: Digital output of zero's run intersecting one's run using standard algorithm (a) and using proposed algorithm (b) before bits correction

3.3 SAR-CD algorithm proof

The new proposed SAR-CD algorithm should satisfy the following two conditions to ensure that it is working correctly and similar to the conventional algorithm. These two conditions are: (1) correct solution and (2) same boundary conditions. For example, figure 3.3(a) presents a general digital output for a given input value. As the input to bit 'J' is V_J (which is denoted by "value" in figure 3.2), the weight of the bit 'J' is $W_J = 2^J$, 'J' ranges from "Nbits-1" to '0', where "Nbits" is the total number of bits. The input corresponds to an n-bits zero's run intersecting one's run at index 'J'.

From the previous comparison between the conventional SAR and the proposed SAR-CD algorithms, it is found that as long as the input to each bit is greater than the weight then both algorithms will behave the same. The two algorithms are different only when the input is lower than the weight, which should give '0' for this bit. Figure 3.3(b) presents the desired output from the second algorithm before bits correction. When bits correction is applied (XNOR operation between each two successive bits as depicted in figure 3.2) the correct output in figure 3.3(a) is achieved.

It is now desired to prove that the proposed SAR-CD algorithm produces the group of bits "Diff group" in figure 3.3(b) and using the same input, which is "V_{J-1}", the input to the bit of index "J-n-2", "V_{J-n-2}", is the same as when following the conventional SAR algorithm. At such a point the second algorithm will behave again like the conventional SAR algorithm. Starting from the condition on "V_{J-1}", implied by the example in (1), we calculate the inputs to the bits indexed

from “J-1” to “J-n” . Processing bit “J-1“ by making absolute comparison between the bit weight “W_{J-1}” from the input “V_{J-1}” results in the inequality in (2), Such that “W_{J-1}> V_{J-1}” as implied by the example. The middle term in equ.(2) presents the input to bit “J-2”.

$$W_{J-n-1} < V_{J-1} < W_{j-n} \tag{3.1}$$

$$W_{J-1} - W_{J-n-1} > W_{J-1} - V_{J-1} > W_{J-1} - W_{J-n} \tag{3.2}$$

For each bit, the difference between the bit weight and the summation of all the LSB bits is W_0:

$$W_{J-1} = \sum_{k=J-2}^{0} W_k + W_0 \tag{3.3}$$

And then moving the term $W_{j-n-1}$to the left hand side results in:

$$W_{J-1} - W_{J-n-1} = \sum_{k=J-2}^{J-n} W_k + W_{J-n-1} = \sum_{k=J-2}^{J-n-1} W_k \tag{3.4}$$

Substituting from equ.(4) by “W_{J-1}-W_{J-n-1}” and similarly by “W_{J-1}-W_{J-n}” to the left and right sides of equ.(2) respectively we find that:

$$\sum_{k=J-2}^{J-n-1} W_k > W_{J-1} - V_{J-1} > \sum_{k=J-2}^{J-n} W_k \tag{3.5}$$

The middle term implies that the input to bit “J-2” is greater than the summation of all the weights from “J-2” to “J-n”. This concludes that processing all the bits from “J-2” to “J-n” will give the uncorrected ones expected in figure 3.3(b) (ones run from index “J-2” to “J-n”). After the processing, the value entering bit “J-n-1” is then (middle term):

$$\sum_{k=J-2}^{J-n-1} W_k - \sum_{k=J-2}^{J-n} W_k > W_{J-1} - V_{J-1} - \sum_{k=J-2}^{J-n} W_k > 0 \tag{3.6}$$

Which then becomes:

$$W_{J-n-1} > W_{J-1} - V_{J-1} - \sum_{k=J-2}^{J-n} W_k > 0 \tag{3.7}$$

This relation indicates that processing bit “J-n-1” results in ’0’ as expected in figure 3.3(b). Hence, the input to “J-n-2” bit is in the form (middle term)

$$0 < W_{J-n-1} - (W_{J-1} - V_{J-1} - \sum_{k=J-2}^{J-n} W_k) < W_{J-n-1} \tag{3.8}$$

From equ.(4), it is found that:

$$\sum_{k=J-2}^{J-n} W_k = W_{J-1} - 2W_{J-n-1} \tag{3.9}$$

Substituting in equ.(8), it is proved that the middle term that represents the input to bit “J-n-2” equals to “$V_{J-1} - W_{J-n}$”, the same value that would have been obtained when following the conventional algorithm.

3.4 SAR-CD algorithm examples

In the following examples, the demonstration of the new algorithm is presented. The variable names follow the algorithm presented in the flow chart 3.2. For each stage, the absolute difference between the input and the stage reference is calculated. The result of the operation (“temp” in figure3.2) is presented in the forth coulomb. The digital bit before correction is located in the fifth coulomb, which is the result of the condition “is(temp>0)”. The case when “is(temp==0)?” is TRUE is marked by highlighting the absolute difference result with yellow (in second and third examples). The case when the result bit is zero, “is(temp>0) is FALSE, is marked in red to indicate that the next bit will be toggles in the bit correction operation. The final digital binary output is shown in the last coulomb.

3.4.1 Example 1 to convert “10.1” analog input to “1010”

The next table shows how the new algorithm, SAR-CD, can be used to convert the analog quantity 10.1 to the 4 bits-binary approximation b[3]b[2]b[1]b[0] = “1010” using SAR-CD algorithm depicted in the flow chart 3.2). The example is shown in figure 3.4.

Step/ stage	Bit to calculate	Absolute difference (input– 2^(N_bits-stage))	Abs difference out |-|	Digital bit before correction	Digital bit after correction (b[i] XNOR b[i+1])
1	b[3]	|10.1 - 8 |	2.1	1	1
2	b[2]	|2.1 - 4|	1.9	0	0
3	b[1]	|1.9 - 2|	0.1	0	1 (toggled)
4	b[0]	|0.1 - 1|	0.9	0	0

Figure 3.4: Example 1 to convert analog quantity 10.1 to binary “1010” using SAR-CD algorithm

The first iteration\stage starts with an input 10.1. The input is compared to the stage weight, ’8’, through the analog absolute difference operation which calculates the absolute difference between them. The differences |10.1-8| = 2.1 is the input to the second stage. As 10.1 is bigger than 8, then the first stage ends with digital b[3] is 1 (no bit correction is needed for the first iteration according to figure 3.2).. The second stage compares the input 2.1 and the weight 4. As 2.1 is not bigger than 4 then the digital bit for this stage (b[2]) before bit correction is 0. As b[3] is 1 then b[2] is kept with 0. The absolute difference (|2.1-4| = 1.9) is the input to the third stage. In the third stage, b[1] is initially set to 0 as 1.9 is not bigger than 2. However, bit correction toggles b[1] from 0 to 1 as the previous bit, b[2], equals to 0. The absolute difference out for the third stage, 0.1, is the input to the forth and last stage. In the forth stage, b[0] is initially 0 because 0.1 is not bigger than 1. And it is kept the same because b[1] has a new value of 1. The digital 4 bit binary representation for 10.1 is correctly presented by 1010 using the SAR-CD algorithm.

3.4.2 Example 2 to convert “20” analog input to “10100”

Step/ stage	Bit to calculate	Absolute difference (input– 2^(N_bits-stage))	Abs difference result |-|	Digital bit before correction	Digital bit after correction (b[i] XNOR b[i+1])
1	b[4]	|20 - 16 |	4	1	1
2	b[3]	|4 - 8 |	4	0	0
3	b[2]	|4 - 4 |	0	0	1 (toggled)
4	b[1]	|0 - 2 |	2	0	0
5	b[0]	|2 - 1 |	1	1	0 (toggled)

Figure 3.5: Example 2 to convert analog quantity 20 to binary “10100” using SAR-CD algorithm

In this example, it is desired to convert the analog input 20 to the digital 5 bits binary representation b[4]b[3]b[2]b[1]b[0] = “10100” using SAR-CD algorithm depicted in the flow chart figure 3.2. The example is presented in figure3.5.

The first iteration\stage starts with an input 20. The input is compared to the stage weight, ’16’, through the analog absolute difference operation which calculates the absolute difference between them. The differences |20-16| = 4 is the input to the second stage. As 20 is bigger than 16, then the first stage ends with digital b[4] is 1 (no bit correction is needed for the first iteration according to 3.2.The second stage compares the input 4 to the weight 8. As 4 is not bigger than 8 then the digital bit for this stage (b[3]) before bit correction is 0. As b[4] is 1 then b[3] is kept with 0. The absolute differnce (|4-8| = 4) is the input to the third stage. In the third stage, b[2] is initially set to 0 as 4 is not bigger than 4. Before we do bit correction (b[2] XNOR b[3]), it is marked that the absolute difference in this case is 0 (the condition “is(temp==0)?” in figure 3.2 is TRUE). Thus b[2] should have the value “ is(temp==0)? . b[3]” added. However as b[3] is zero, then no change happens to b[2] before bit correction. Then, bit correction for b[2] (b[2] XNOR b[3]) results toggling b[2] from 0 to 1. In the fourth stage, b[1] is initially 0 because the input is zero. Also, b[1] is kept the same because b[2] is resolved to 1 after bit correction. This value toggles b[0] from 1 to 0 in the fifth and last stage. The final digital output is “10100”.

3.4.3 Example 3 to convert “20” analog input to “10100”

Step/ stage	Bit to calculate	Absolute difference (input– 2^(N_bits- stage))	Abs difference result |-|	Digital bit before correction	Digital bit after correction (b[i] XNOR b[i+1])
1	b[4]	|28 - 16 |	12	1	1
2	b[3]	|12 - 8 |	4	1	1
3	b[2]	|4 - 4 |	0	0	1 (toggled)
4	b[1]	|0 - 2 |	2	0	0
5	b[0]	|2 - 1 |	1	1	0 (toggled)

Figure 3.6: Example 3 to convert analog quantity 28 to binary “11100” using SAR-CD algorithm

In this example, it is desired to convert the analog input 28 to the digital 5 bits binary representation b[4]b[3]b[2]b[1]b[0] = “11100” using SAR-CD algorithm depicted in the flow chart 3.2. The example is presented in figure3.6.

This example is very similar to the second example. However, the difference is spotted in the second and third stages. The result of the third stage now is one because the input 12 is higher than the reference 8. Thus, when calculating b[2] in the third stage, and as “is(temp==0)?” is TRUE, b[2] will be effected by the operation “b[i] = b[i] + is(temp==0)? . b[i+1]” and will not be affected by the bit correction “b[i] = b[i] XNOR b[i+1]” in figure 3.2.

For more examples, the reader can use the Matlab function sar_cd.m in appandex section A.1.2. This function presents the functionality of the novel Successive Approximation Algorithm with Continuous dis-assembly.

Chapter 4

Circuit design

Two circuit designs are presented in this work to demostrate the proposed algorithm. The first design presents the initial domenstration for the algorithm ([26]). The second circuit design presents a new architecture with many enhancements that solve many of the challenges found in the first design at the cost of circuit complexity([27]). It is recommended for the reader to go through both algorithms to give more insight to the new developed techniques.

4.1 First circuit design

The proposed circuit is a 4-bit system. It consists of 4 stages and each stage is responsible for evaluating the current bit value and correcting it by the bit evaluated from the previous stage; which is an input to this stage. Figure 4.1 shows the unit cell. In a general n-bit circuit, there should be 'n' successive stages of the same cell type (with different refernce pulse as will be shown).

For a general K^{th} cell, the cell is triggered once an input pulse from the previous stage (k-1) is detected. The input pulse "Pin" presets the digital value "b[k]" to '1' and triggers a pulse generator to generate a pulse of width proportional to Vfs/2^k; as Vfs is the full scale voltage. The input signal is delayed until the reference signal is high, and is compared to it by a simple XOR gate. The result of this comparison is a pulse with the difference-width. This pulse is an output for this stage and used to trigger the next stage. Also, the output pulse triggers the comparator by a small pulse to indicate the larger pulse. The inputs to the comparator are delayed versions of the input pulse "Pin" and the generated reference pulse to compensate for the XOR gate delay. The output of the comparator is the

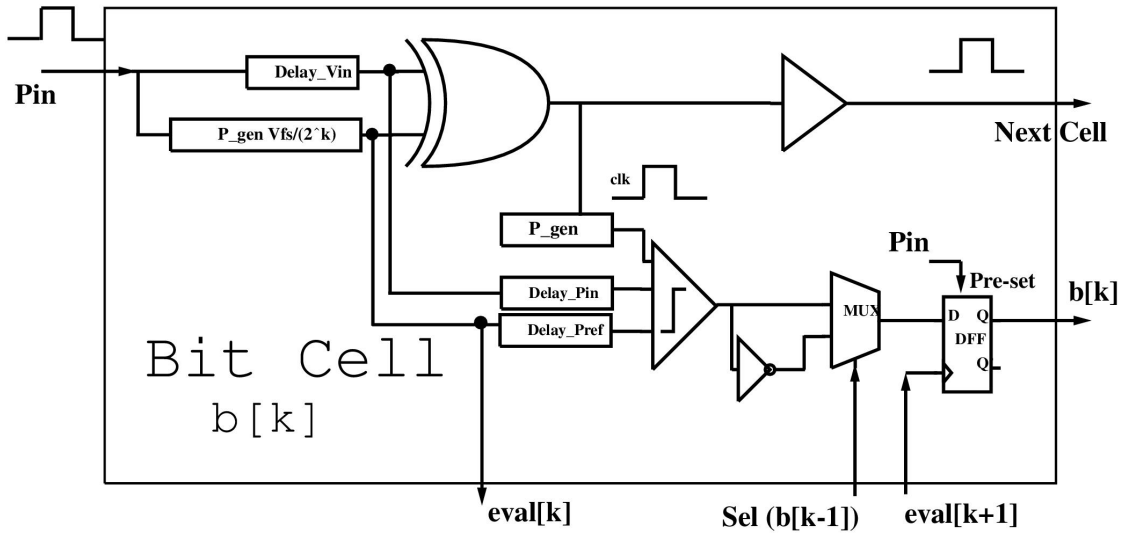


Figure 4.1: Bit unit cell

correct value for the bit “b[k]” after it passes through two conditions. The first condition is the previous bit value (b[k-1]). As stated in the algorithm, if the previous bit is ’0’, then the value of the current bit should be reverted. This is done using the 2*1 multiplexer with selection “Sel (b[k-1]))” such that the comparator output is selected when b[k-1] is ’1’ or the inverse is selected when b[k-1] is ’0’. The second condition resolve the problem of the XOR gate defined resolution as explained next.

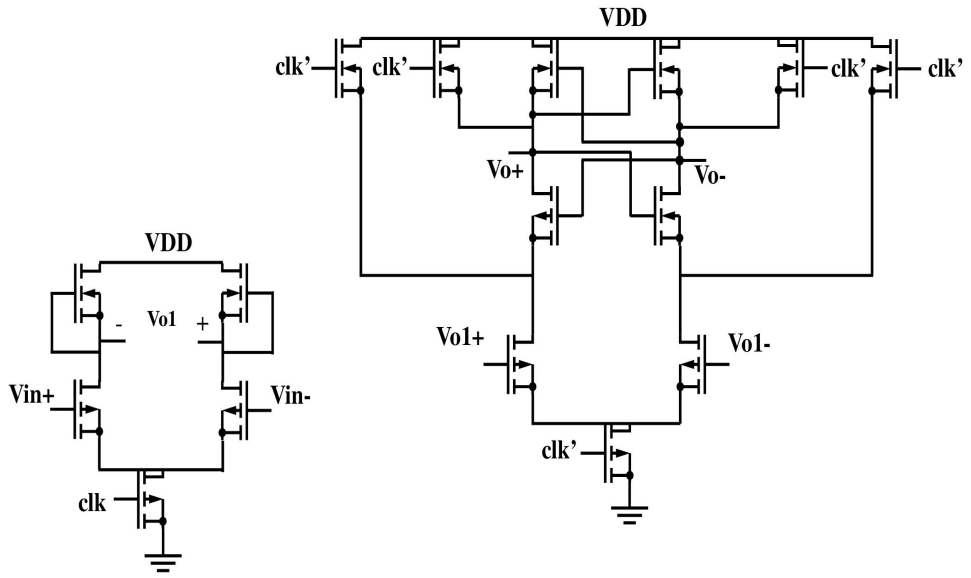
When the XOR inputs are very close in length, the XOR may not produce an output sufficient to trigger the next stage because of the limited resolution. In this case, which are more likely to happen for high sampling rates, there will be an error comparable to the reference pulse the input is compared to. In an example of 4-bit system, if this problem is encountered for the MSB, the output will be ‘0000’ instead of ‘0111’, which is a great loss. To solve this problem, we preset the digital bit of the current cell (“pre-set”). As long as this cell is triggered and no output from the XOR gate, and of course successive stages won’t be triggered, then the input pulse is very close to the reference pulse and the current bit should be forced to ’1’. However, in normal operation, when the XOR output is sufficient to trigger the next stage, the output of the comparator should be considered the correct value. This is done by the “eval[k+1]” feedback signal taken from the reference pulse generated by the next stage. This indicates that the signal survived to the next stage and the comparator output is selected instead of the preset value

'1'. The error in this case is defined by the resolution of the XOR gate. In other words, the resolution of the XOR gate defines the resolution of the system for this architecture.

This solution imposes a correlation between the signal path and the comparison path, when the 2 signals are compared for each bit. This correlation adds conditions to the design that requires time and power budget to fix. These conditions can be summarized in 2 points. The first one is fitting the timing requirements for the comparator of the early stage. When this signal is back from stage $k+1$, the comparator output should be ready with the correct comparison result. As shown in figure 4.6, a typical failure case happens when the signal path starting from the reference pulse output through the XOR gate, the buffer and the pulse generator of the next stage takes smaller delay than the comparator delay (in figure 4.1, eval[k+1] samples the output of the comparator before it is ready). One solution, which is used in this design, is to delay the signal "eval" till the comparator output is ready. This delay is found to be relatively high reaching more than 120ps (for the comparator used), which is a loss of area and also a loss of power that can reach up to 30% of the circuit power for a high speed delay line. The second point is meeting a condition that states that the delay summation of the XOR, buffer and the reference pulse generator should be greater than the delay summation of the DFF access time, DFF setup time and the multiplexer setup time. This exposes another main function for the buffer; which is to impose some delay to satisfy this condition. This condition and the problem are addressed in the simulation results section.

4.1.1 Design components

The comparator used is depicted in figure 4.2. It consists of 2 stages. When the clock signal is high, the first stage is activated and stores the input values for the small high period of the triggering clock. At the same time, the PMOS transistors in the second stage pre-charge the output to VDD preparing the coupled transistors for the evaluation phase. The evaluation phase starts once the clock goes low again when the second stage is activated and the first stage is deactivated holding the captured input values to be the input to the second stage. The evaluation is completed by this stage and the output remains on-hold till the next clock edge. Choosing the clock pulse width is critical because it should be long enough for pre-charging phase through the PMOS transistor, and not very long as the input of logic 1 (the input or the reference pulse) may go down quickly weakening the



.

Figure 4.2: The comparator circuit, stage 1 (left) and stage 2 (right)

differential signal held in the first stage. The comparator is triggered using a pulse generated from a pulse generator. This pulse generator is triggered by the XOR output.

The XOR gate used is a simple CMOS circuit. The error in the CMOS circuit is almost fixed along different input combinations and can be compensated in the next stage by changing the width of the generated reference pulse. Using XOR gate designs based on pass transistors add distortion to the output signal so it is not recommended even if they consume less power.

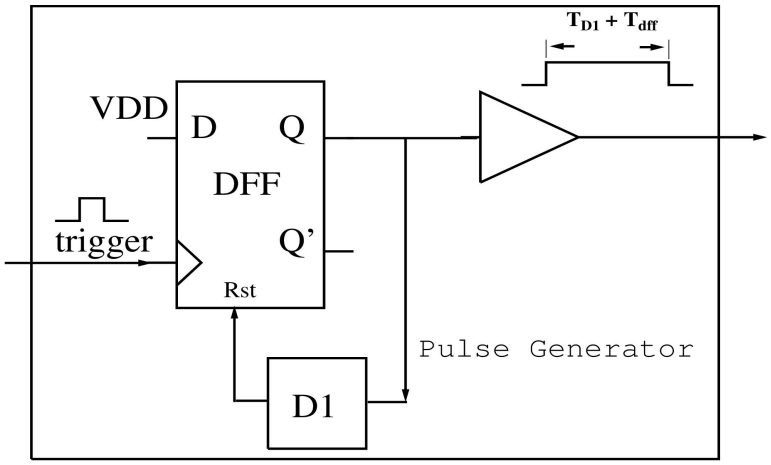


Figure 4.3: Pulse generator circuit

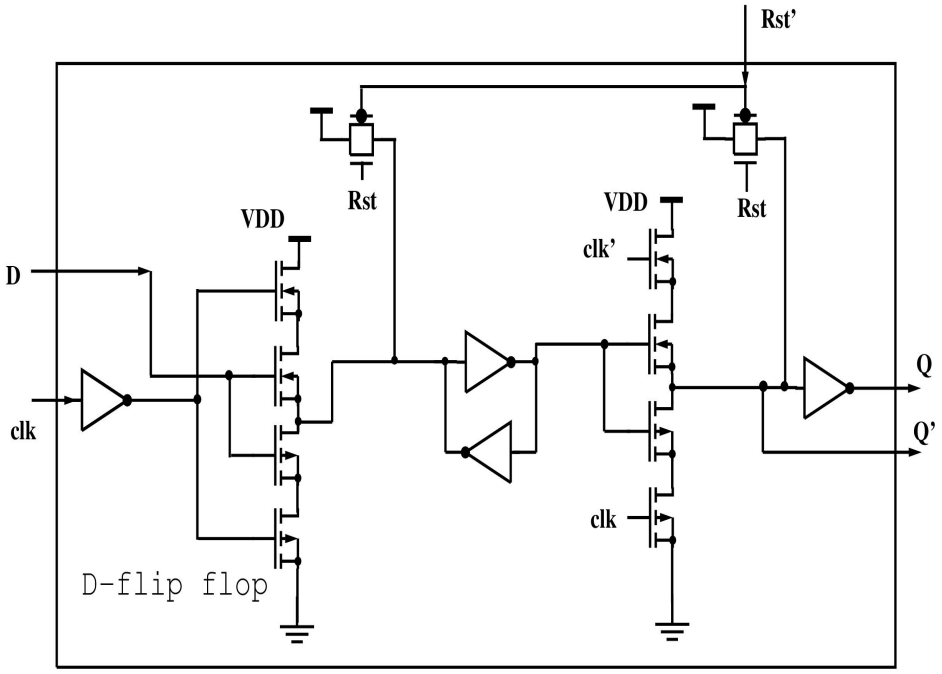


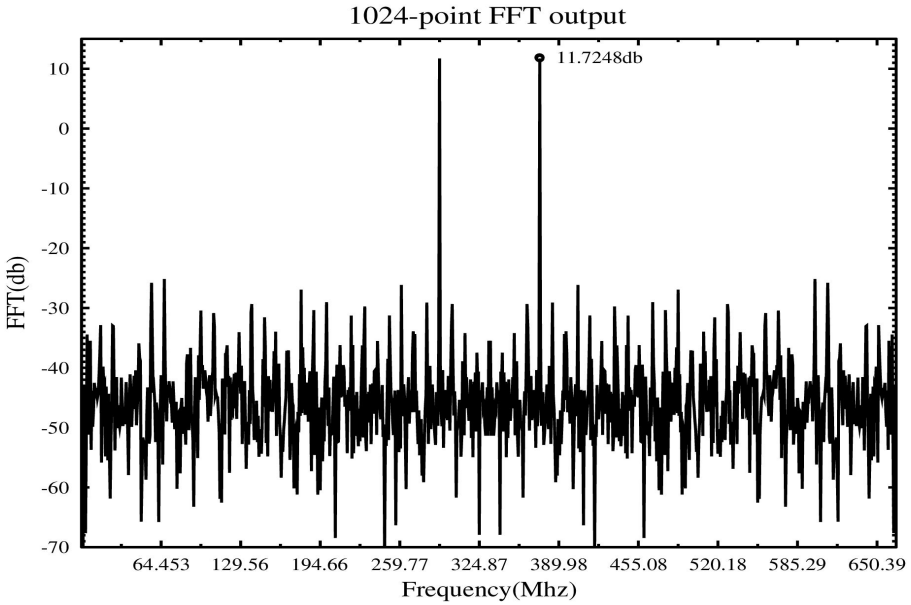
Figure 4.4: DFF based on SDFF

The pulse generator developed is depicted on figure 4.3. It consists of a DFF and a delay element. The input to the DFF is the supply and the output is connected to a delay array ending with the DFF asynchronous reset. Once a clock

trigger is detected, the output is high until the feedback signal reaches the “Rst” pin when the output goes low again. The width of the generated pulse is the summation of the array delay and the DFF loop delay. The delay of the DFF defines the minimum pulse width that can be generated imposing another resolution limit beside the XOR gate resolution. The DFF used is depicted in figure 4.4. This architecture is based on edge-triggered SDFF in [[10]] (Fig.1) for higher speeds over the transmission gate-based DFF.

4.1.2 Simulation results and analysis for the first design

The simulation is done for input pulses of width range from an offset of 10ps to a full scale width of 750ps. The input is mapped from a sine wave with a frequency of 41 MHz with a sampling rate of 666.7 MHz (1.5ns sampling period). The resolution is about 46ps ((750-10) / 16) The offset is optional and is compensated in the first stage by increasing the reference pulse with the same amount. As mentioned, this goes for any possible error from the XOR operation as the reference pulse in the next stage can be modified by design.



.

Figure 4.5: FFT output

The FFT output for the system is depicted in figure 4.5. The SNR corresponds to 3.67 ENOB. The circuit is running on 1.5 ns sampling period to resolve a pulse of 750ps maximum width. Ideally, the two numbers should be very close, however fixed delay in the circuit cause deviation from this value which can be summarized in four main delay sources. The first delay source is from the XOR gate which

consumes 43ps for each bit. The second one is the pulse generator that consumes a delay of 35ps for each bit. The third one is the feedback delay deployed at the signal "eval" path early mentioned to compensate for the comparator delay time ("Dcomp" in figure 4.6). The delay added is the one in the last bit stage only to take the final digital word; as this delay is consumed parallel to the signal path. This delay can be totally removed if the digital word is taken after the start of the new sample, when this stage is not used (caution should be taken when resetting the digital bits for the new sample). The last delay is a forced 20ps delay presented by the XOR gate buffer for each bit to satisfy the condition pointed out in section III and is explained in the next paragraph.

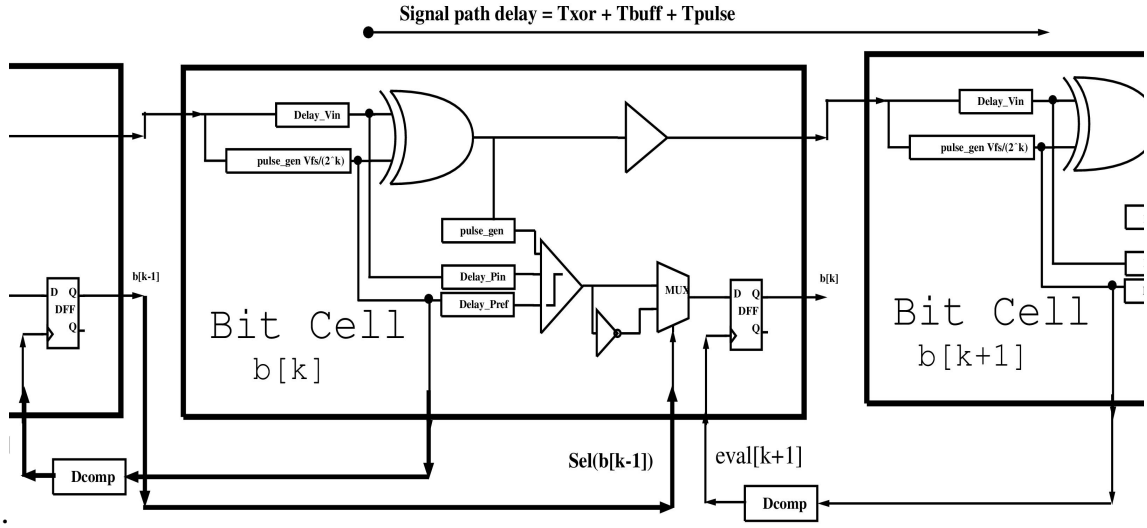


Figure 4.6: Feed back synchronization problem

To explain the reason for the condition in section III which states that the delay summation of the XOR, buffer and the reference pulse generator should be greater than the delay summation of the DFF access time, DFF setup time and the multiplexer setup time, consider figure 4.6. It depictss a capture of 3 stages; indexed from "K-1" to "k+1" (stages "k-1" and "k+1" are clipped for fitting purpose). Starting from stage "K", consider the signal at the output of the reference pulse generator output which is ready to enter the XOR gate and also to be fed-backed to stage "K-1". This signal takes 2 paths. The first path passes through the XOR gate and the buffer of stage "k", then through the reference pulse generator of stage "k+1", then through the feedback "Dcomp" (eval[k+1]). The summation of the path delay is:

$$T_1 = T_{XOR} + T_{common} + T_{buff} + T_{PulseGen} + T_{Dcomp} \tag{4.1}$$

As $Txor$ is the delay of the XOR gate and $Tcommon$ is the common part in the signal pulse and the reference pulse at which both of the two signal is high (the XOR gate output is zero in this period). The second path is through "Dcomp" delay element, DFF for stage "k-1", then through the MUX and the input of the DFF of stage "k", the summation of the path delay is:

$$T_2 = T_{Dcomp} + T_{DFFaccess} + T_{mux} + T_{DFFsetup} \tag{4.2}$$

For proper operation, the input to the MUX of the stage "K" should be ready before the clock signal "eval[k+1]" comes. This means that T1 > T2 is a condition that should be satisfied. All of these delays are fixed except the Tcommon which can take a value from '0' to Tfs2. As Tfs2 is the width of Pvfs2. Considering the worst case when Tcommon is equal to '0', the condition for proper operation becomes:

$$T_{XOR} + T_{buff} + T_{PulseGen} > T_{DFFaccess} + T_{mux} + T_{DFFsetup} \tag{4.3}$$

It is now clear that the buffer used after the XOR gate not only strengthens the signal but also it synchronize the internal signals for proper operation.

4.2 Second circuit design- All Digital TDC

The second circuit is considered a modified all-digital version of the First design. Figure 4.7 depicts a general N-bit architecture. The architecture in this work is a 10-bit version introduced as a case study. For each clock cycle, each cell performs two tasks. The first task is comparing the input pulse to the corresponding reference pulse to generate uncorrected bit "bu[k]" for the current sample, where 'k' ranges from '9' to '0'. The second task is to make bit correction (as presented in the algorithm in figure 3.2) to generate B[k]. Figure 4.8 depicts a general k^{th} unit cell. The input pulse Pin triggers a pulse generator to generate the reference pulse which corresponds to the binary weight of the cell bit.

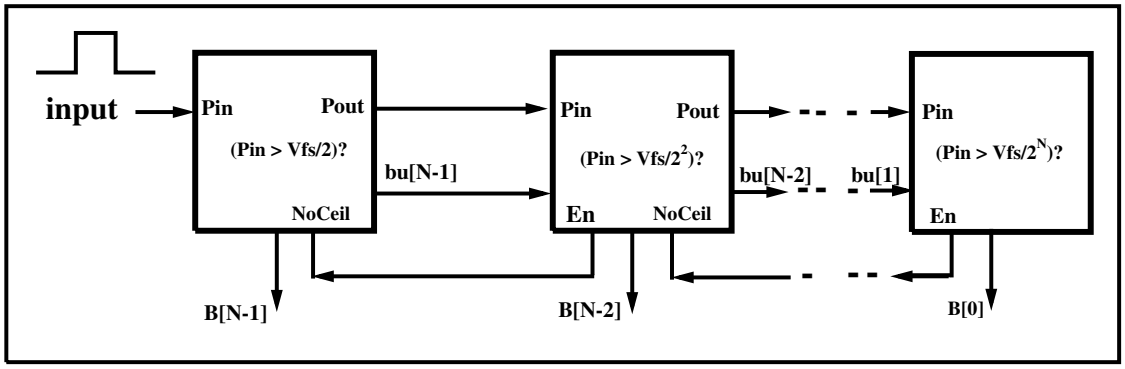


Figure 4.7: Successive approximation with continuous disassemble algorithm system architecture.

4.2.1 Circuit description

The triggering input pulse Pin[k] is delayed to compensate for the pulse generator propagation delay such that the XOR inputs (i.e., Pind[k] and Vr[k]) arrive simultaneously. The MSB stage generates a pulse with width Pfs/2, as Pfs is the full scale pulse. The delayed version of the input, Pind[k], and the reference pulses Vr[k] are applied to the XOR gate input ports to generate the absolute difference-pulse, which is then buffered to the next stage. In the mean time, inverted versions of the input and the reference pulses enter the clock and the input of a DFF respectively. The output of the DFF, X[k], indicates which pulse is longer. The DFF resolves value '1' when the input pulse is longer than the reference pulse and '0' otherwise. The delay unit circuit consists of two cascaded current-starved inverters. The delay of each unit is adjusted by sizing the current-starved inverter.

Figure 4.9shows the timing diagram for simulating the first 2 stages (k = 9,8) for 2 complete clock cycles (i.e., three clock edges). The 2 cycles resolve one input pulse longer (Pin9_1) and one shorter (Pin9_2) than half of the full scale pulse (Vr9). Pin9, Vr9, X9, bu[9] and B[9] represents the signals for the MSB cell (Pin9 and Vr9 share the voltage axis). At the first sample, the absolute difference between the input and reference pulse is the input to next stage (Pin8) which will be compared to the stage reference pulse (Vr8). The first sample resolves X9 and X8 to '1' and '0'. For normal operation, when both signals "En" and "NoCeil" (figure 4.8) are high, the uncorrected bits bu[9] and bu[8] are the output of the XNOR gate after bits correction. In this sample, bits correction does not change bu[8] value as bu[9] equals one according to the algorithm in figure 3.2 . This is shown at the vertical trace "T1" when the initialization value '1' for bu[8] is

changed to ’0’ in accordance to changing bu[9] from ’0’ to ’1’. However, for the second sample, when the input pulse is shorter than the Vr9 pulse, bu[9] resolves to ’0’ which keeps bu[8] to its initial value. This can be shown at vertical trace “T2” when the initial value for bu[8] does not change. The digital output of the second sample is ready at the third clock positive edge “T3” as B[9]B[8] are set to ’01’.

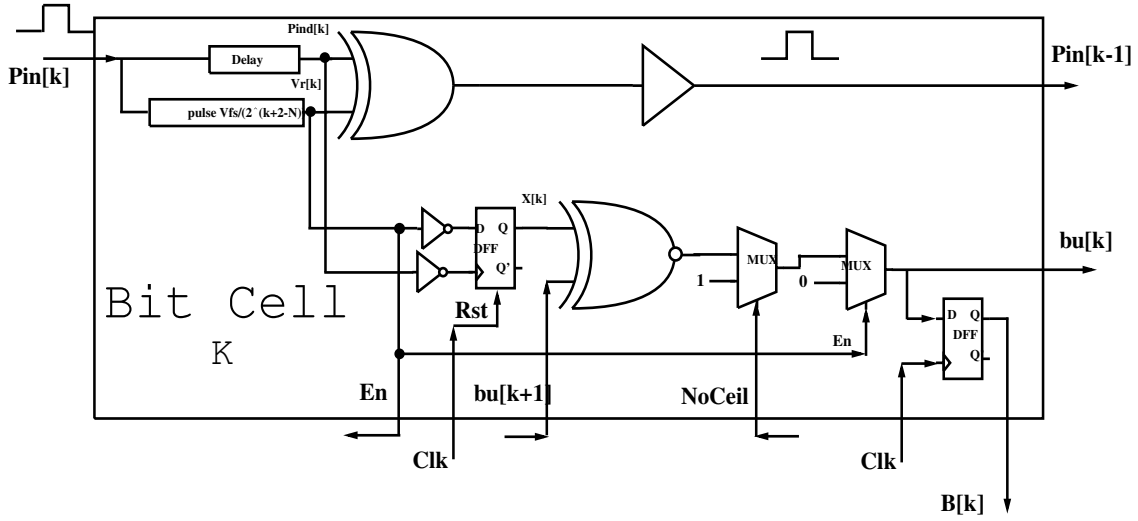


Figure 4.8: System unit bit cell

The enable signal “En” indicates the cell is triggered and the output value is correct. This flag is taken from the reference pulse and is held on a DFF to pass the result (This DFF is not shown in the diagram for simplicity). Due to the limited resolution of the XOR gate, the XOR gate might not produce an output sufficient to trigger the next stage, when the XOR inputs edges are very close. In this case, which are more likely to occur for high sampling rates, there will be an error. In an example of 4-bit system, the output will be ‘0000’ instead of ‘0111’ if this occurs for the first stage, which is a significant error. To solve this problem, a “pre-set” of the digital bit for the current cell is done. As long as this cell is triggered and no output from the XOR gate, and the input pulse is very close to the reference pulse, the current bit in this case is forced to ’1’. This is performed through the “NoCeill” feedback signal taken from the reference pulse generated in the next stage (similar to the eval[k+1] signal in figure 4.1). This indicates that the signal survived to the next stage and the comparison output is considered. The resolution of the XOR gate defines the resolution of the system. The resolution that can be resolved by the DFF to generate the ’X[k]’ signal should be designed higher than the resolution of the XOR gate.

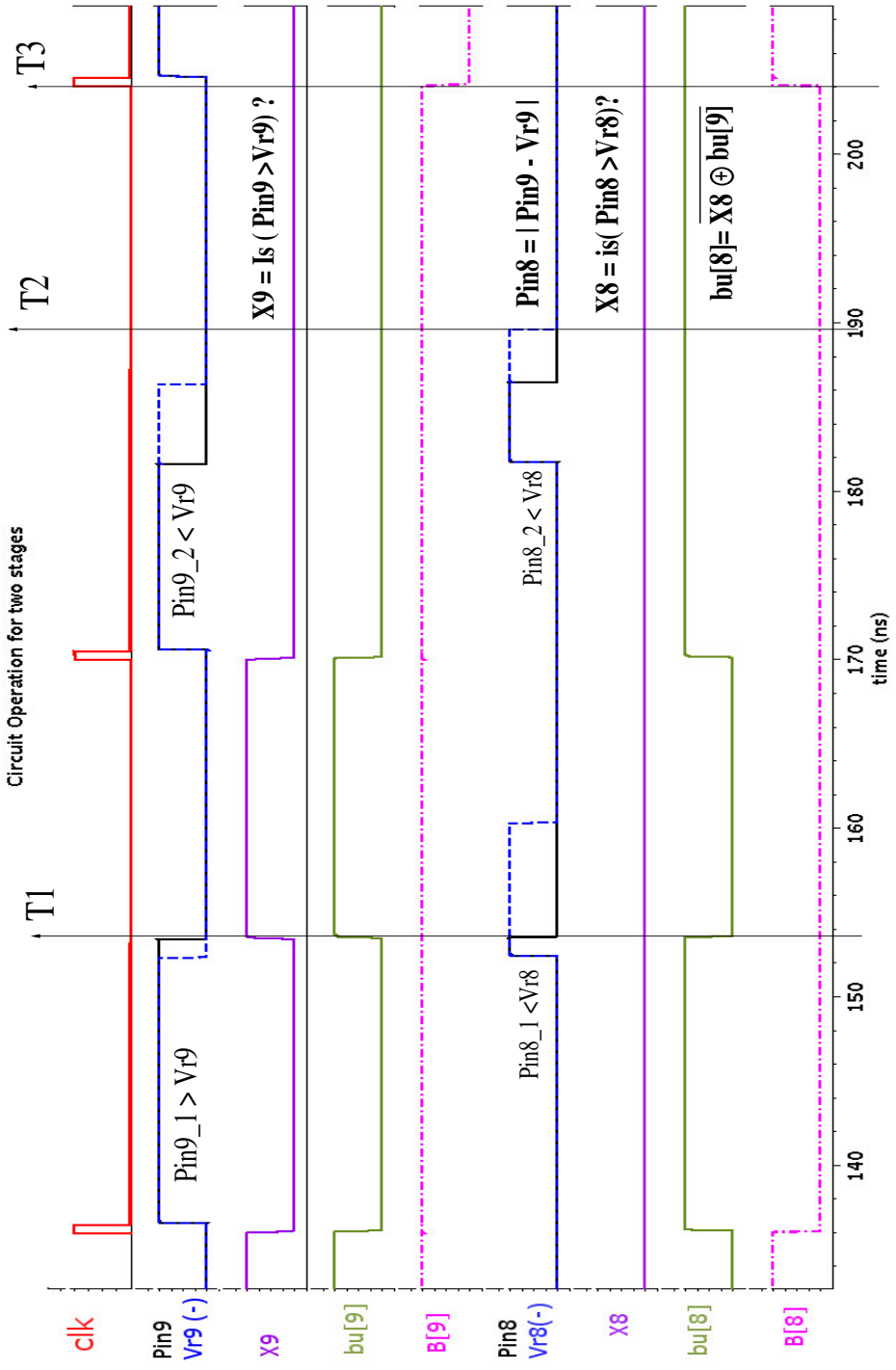


Figure 4.9: Simulation capture for the two MSB stages signals (landscape view)

4.2.2 Circuit components

The XOR gate used is implemented using CMOS logic. The error in the CMOS circuit is almost fixed along different input combinations and can be compensated

in the next stage by changing the width of the generated reference pulse. The pulse generator used is depicted on figure 4.10. It consists of two DFF and a delay element. The DFF used is based on edge-triggered SDFF introduced in [10] and is shown in figure 4.11. The generation is done in two steps. The cell input pulse triggers the first DFF producing an intermediate fixed length-pulse to trigger the second DFF. The second DFF and the delay element generate the desired pulse with custom width. For each DFF, the output is connected to the DFF asynchronous reset. Once a clock trigger is detected, the output is high until the feedback signal arrives at the "Rst" terminal and the output goes low again. Adding extra delay element "D1" increases the width of the generated pulse. The width of the generated pulse is the summation of the delay element and the DFF loop delay.

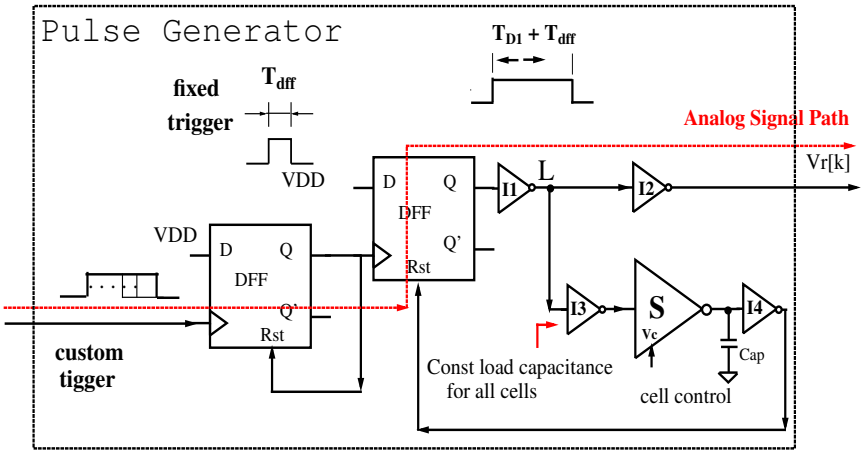


Figure 4.10: Pulse generator circuit

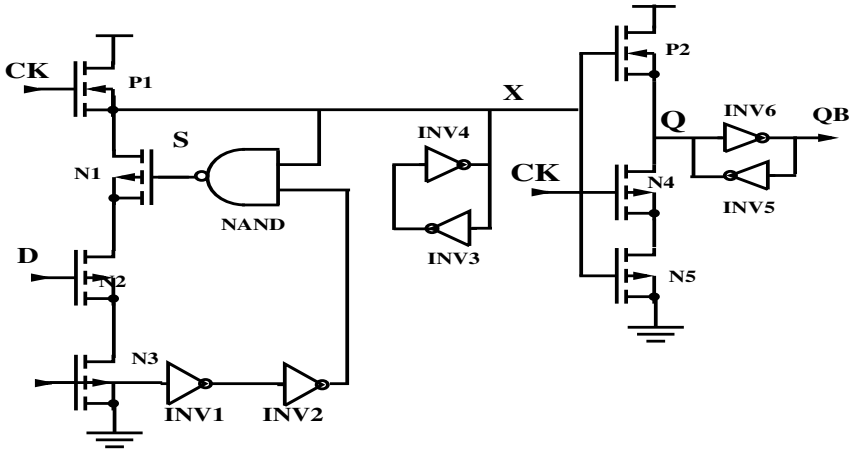


Figure 4.11: SDFF introduced in [10]

The pulse generation is done in two steps using two DFFs to eliminate the dependency of the generated pulse width on the input pulse width. The pulse generation trigger is done by the input signal, which can vary in width, affects the width of the generated signal. In figure 4.11, it is shown that the clock used for transistor “N4” works against the ’Rst’ operation by discharging the node ’Q’ to ground level while the reset signal (implemented by a PMOS with source connected to the VDD and gate connected inverted Rst and drain connected to ’Q’, not shown for simplecity) charges it to VDD. This introduces pulses with two lengths. One long pulse produced when the clock pulse is longer than “TD1+ D1”, and smaller pulse when clock signal is shorter. The difference between the two pulse widths is up to tens of picoseconds. This introduced the need to generate an intermediate pulse with short fixed length to trigger the actual pulse generator, the second DFF. The error after this addition is reduced to less than 1ps. The delay of the DFF defines the minimum pulse that can be generated.

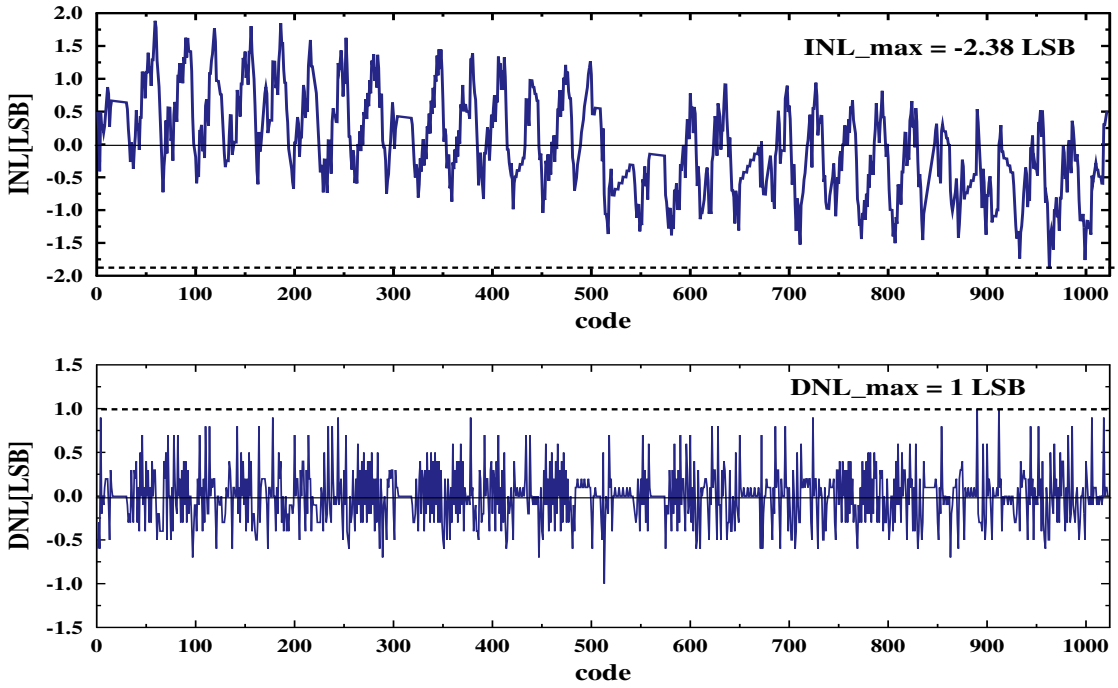


Figure 4.12: Simulated DNL and INL for the second design

4.2.3 Simulation results and analysis for the second design

The simulation is done for input pulses of width range from an offset of 10ps to a full scale width of 31.5ns. The simulation setup takes an input sine wave of frequency 1.75MHz and samples it at 29.4 MHz (chosen 34ns sampling period). The signal is converted using an ideal VTC to a pulse width-modulated signal with 31.5ns full scale time range. The resolution is around 30.7ps (31.5ns /1024) The output of the VTC is then applied to the target TDC circuit. A 1024-FFT processing is applied and the circuit achieved an Effective Number Of Bits (ENOB) value of 8.63bits. The circuit consumes 2.8mW from a 1V supply. Linearity test is done by simulating 10,240 point sweeping the full scale range. The results show maximum 1 LSB and -2.38 LSB for Differential Non Linearity (DNL) and Integral Non Linearity INL respectively as shown in figure 4.12. Figure 4.1 draws comparison to [18] and [28]. The theoretical maximum frequency is governed by the XOR gate resolution. Simulation results show that CMOS-based XOR gate on 65nm technology provides around 30ps LSB which supports about 33 Mhz maximum frequency for a 10-bit version of this architecture.

	This work	[28]	[18]
Archeticture	SAR-CD	SA	DSSA
Sampling (Ms/s)	29.4	12.5	80
Number of bits	10	13	10
Power(mw)	2.769	63.3	9.6
Supply(V)	1	3	1.2
Technology	65nm	350nm	65nm
DNL/INL max(LSB)	1/-2.38	-/0.74	1.35/2
Simulated/Measured	Simulated	Simulated	Measured

Table 4.1: Performance comparison

The fact that the proposed algorithm promises for one signal path instead of two eliminates the synchronization requirements along the conversion stages like in [17, 18]. This eliminates many delay compensators like the DFF and MUX delays in Fig. 6(a) in [18]. In addition, there is no conditioning in the signal path like in [18] as the MUX selects signal to pass depending on the current comparison result. These superior advantages obtained by the new algorithm make the design simpler and provide significant reduction in power and area.

The second design also showed enhancements over the first design in many points:

A- Eleminating the need to the comparator, making the design an all-digital design and as a result:

B- Simpler design eleminating some issues such as the one spotted in figure 4.6.

C- Higher linearity by more accurate reference pulse generation.

D- More robustness to PVT changes through lower dependency of the operation to the element delays.

It also should be noted that both circuits needs calibration addition to overcome the PVT changes. For example, generating accurate reference Plus for each stage is very critical. The width of the each should present the corresponding reference pulse width irrespective of the operation temprature or supply voltages. Also it should compete for the process variation. Another point is the need to make the inputs for the XOR gate for each stage start simultaneously. Deviation for this condition affects the linearity directly and will cause the loss of part of the signal. This spots the need for digital calibration (similar to many time-based circuits) as will be shown in the next section.

4.3 Digital Calibration for SAR-CD time-based TDC

The calibration logic should tune the pulse synchronization and the pulse generator delays in figure4.10 for proper operation. The pulse generation calibration for all the stages is first introduced. Calibration of the synchronization delay follows the same manner and is described after.

Figure 4.13 depicts the calibration circuit connection diagram. The whole connection consists of: 1-The target SAR_TDC circuit, 2-The calibration logic, 3-The VTC circuit and, 4- A multiplexer (2*1). The multiplexer chooses between either the circuit external input voltage or the calibration circuit input. The VTC converters the chosen input voltage input a modulated pulse. The VTC used here is an ideal component which should be replaced with real one in a complete ADC circuit. The target SAR-CD TDC circuit converters the selected input pulse to the corresponding digital of N bit (Out[N-1:0]), as N is the number of the ADC bits. The calibration logic controls the delay of each stage pulse generator (Vc in figure 4.10) to tune the reference pulses generated. This is presented by VC[N-1:0] in figure 4.13. The calibration logic also tune the current starved inverter used in signal synchronization ("Vc_synch"). It will be shown how a single control pin is used for all the stages without considerable degradation in system linearity.

Calibration delay tuning should be for the control signal Vc for each delay. Though there is 2 analog controls needed for each stage with total of 2*N bins, as N is the number of the TDC bits, the structure properties for the delay pulse generator make it possible for almost constant signal propagation delay across all the stages. Which make it possible for one control bin for all the synchronization delays ("Delay" in figure 4.10) (assuming all the remaining circuit components are identical for all the stages). In figure 4.10, the signal traveling through the circuit will always see the same path with the same loading. This is achieved by isolating the main current starved inverter "S" and the loading capacitor from the signal path through "I1" and "I3" inverters, which have the same size for all the stages. Different pulse widths are done though control voltage of "S" and the loading capacitor. This may impose symmetry constraints on the realized layout. However, it will be shown in the results that such differences are of minor effect.

Figure 4.14 portrays the main algorithm for the pulse generator calibration using the analog control voltage Vc[N-1:0], as Vc[N-1] is the analog control voltage for the first stage pulse generator which should produce a pulse of half of

the full scale-width. The control signal ranges from 0.5 to 1.2 ('Vcfs').'Vsigfs' is the input signal Vsig full scale range which is from 0.4V to 0.6V in this simulation environment. The algorithm logic tests the digital output "Out[N-1:0]" while changing the corresponding Vc[bitAdc], as "bitAdc" is the index of the stage currently calibrated. Vcfs is quantized into 256 level and is presented in 8 bits. "bitCalib" presents the current calibration bit index inside Vc[bitAdc] word. The algorithm loops 8*N times before each calibration bit in each control voltage is evaluated. Calibrating the synchronization delays, through VC_synch, almost follows the same manner. Loading the maximum input voltage (Vsig is maximum), Vc_synch is tuned for maximum digital output "Out[N-1:0]".

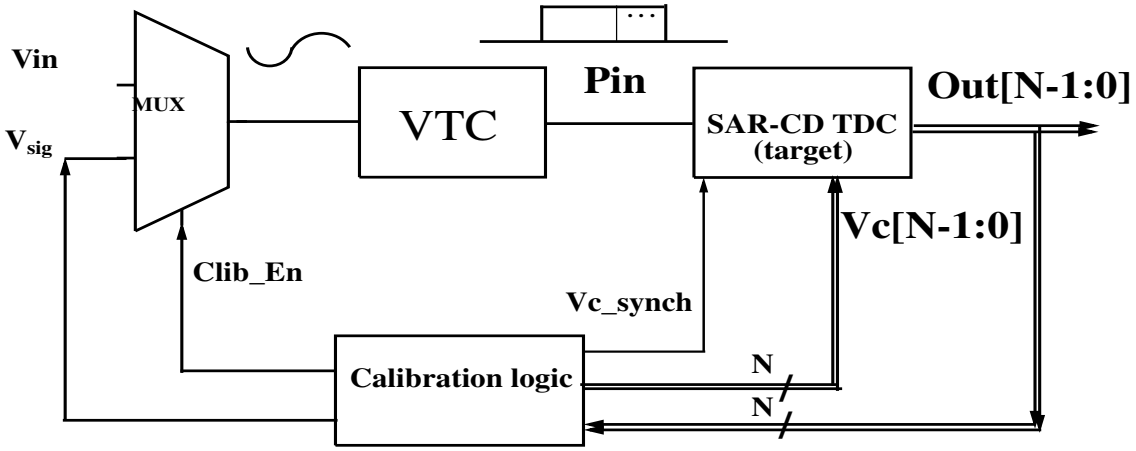


Figure 4.13: Calibration circuit connection diagram

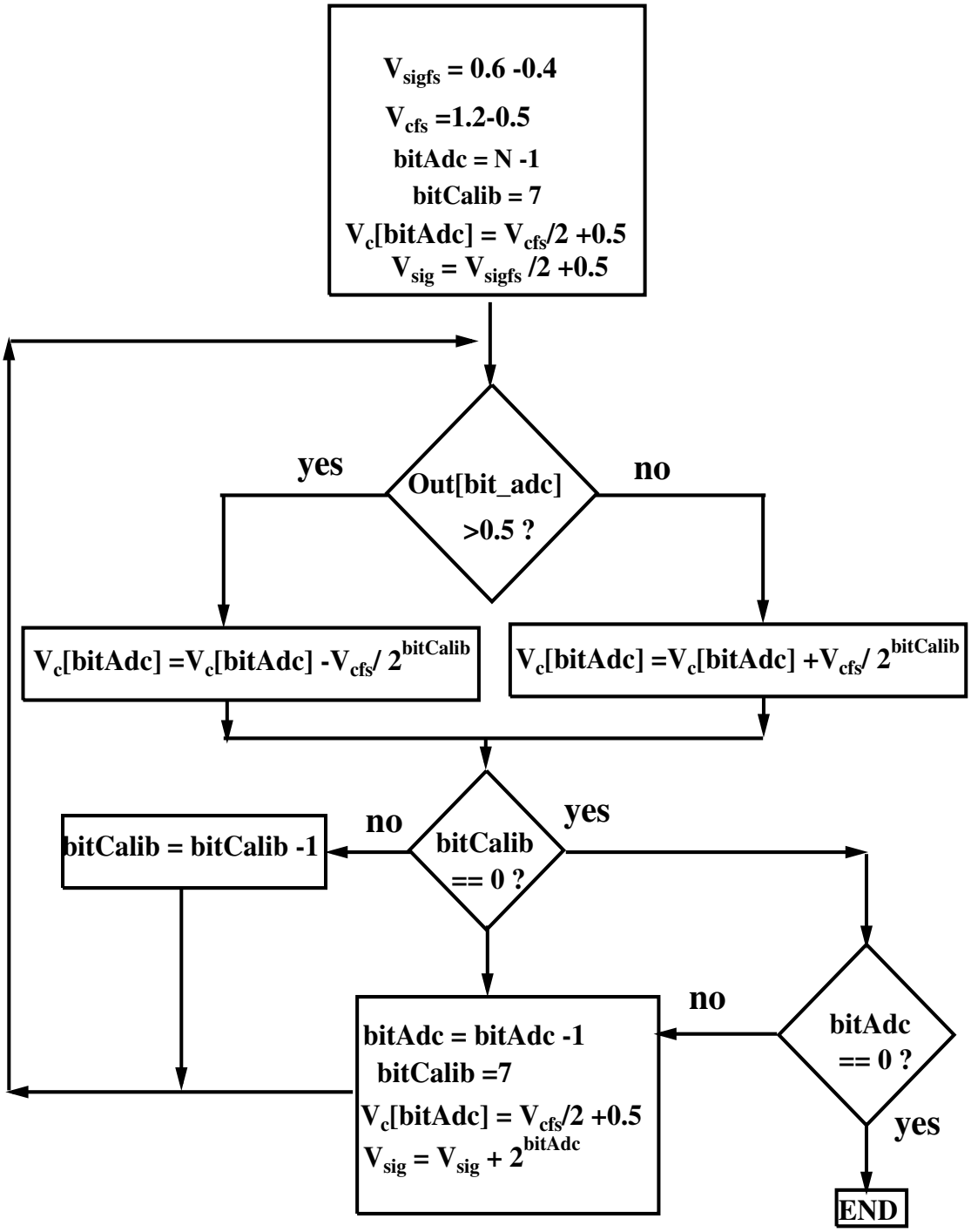


Figure 4.14: Calibration algorithm diagram

The algorithm starts with Vc[N-1] and Vsig are in the middle of the control voltage and the input dynamic range respectively. The main loop of the algorithm loops over Vc[N-1] by updating its value in a binary fashion. The update direction, increasing or decreasing, depends on the corresponding monitored Out[N-1] bit. If Out[N-1] is logic 1 (“Out[bit_adc]>0.5?”), then the generated pulse width of the first stage is longer than the input corresponding pulse width; which corresponds

to half the full scale. Hence, Vc[N-1] should be decreased to make the generated pulse longer. When the algorithm finishes the calibration of the MSB (“bitCalib == 0” is true) the algorithm starts the calibration for the next MSB (“Vc[N-2]”), by decreasing “bitAdc” and re-initializing “bitCalib“ to 7. In this version of the algorithm, the control voltage is quantized into 2^8 levels by initializing “bitCalib” to 7. The designer may increase the voltage resolution more by increasing the initialization value of “bitCalib”. The TDC circuit calibration ends when both “bitAdc” and “bitCalib” are zero values.

Figure 4.15 shows the simulation graphs for calibration of the first MSB, bit ’7’. For a time full scale of 31.5ns (check the simulation results section), the first MSB stage reference pulse should be adjusted to a pulse length of 15.75ns (31.5/2 ns). The graph shows the trials of the calibration algorithm to reach the desired value using the feedback signal “Out[7]”. As long as Out[7] signal is high, the calibration algorithm controls the reference pulse width through the Vc[7] signal (not shown for clarity) in the direction which increases Vr[7] as spotted in figure 4.13. Similarly, when Out[7] signal is low, the calibration logic reduces Vr[7] signal in the same manner.

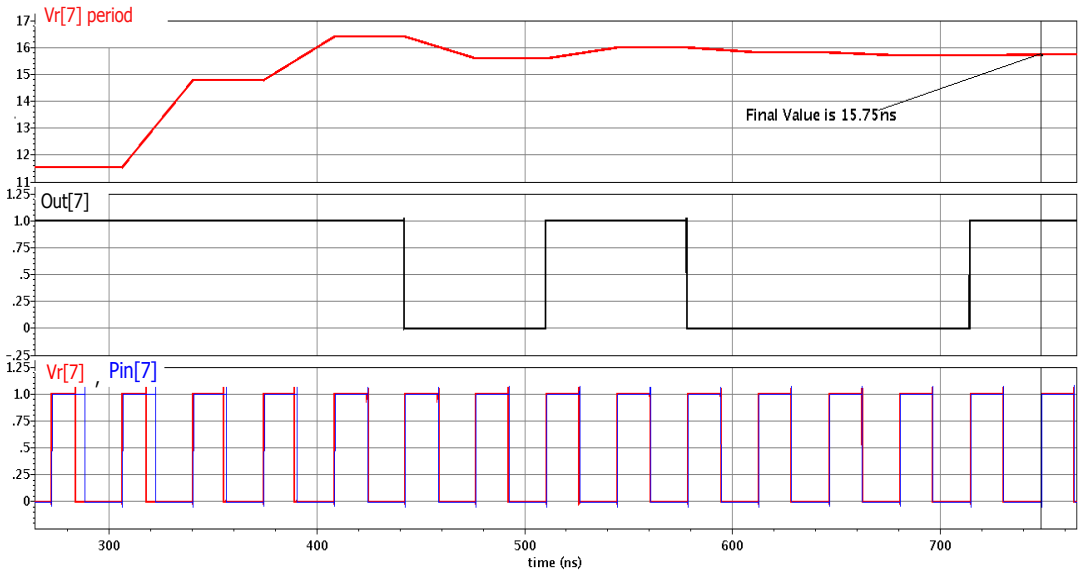


Figure 4.15: Calibration for first MSB

4.4 Calibration results

The calibration enhancement for the circuit operation is showed by Effective Number Of Bits (ENOB) as a measure for the system SQNR. The target circuit is a 9-bit version of the one presented in [27]. With the same input setup parameters presented. The input sine wave is of frequency 1.75MHz and is sampled at 29.4 MHz (34ns sampling period). The signal is converted using an ideal VTC to a pulse width-modulated signal with 31.5ns full scale time range. The input signal The circuit is tested for different operation temperatures and fabrication corners.

The left graph of 4.16 shows ENOB values for 27, 60 and 120 Celsius degrees. As the circuit is designed for 27 degree, increasing the temperature changes the internal propagation delays of the circuit and, hence, degrades the circuit performance. At 60 degree, simulation results showed the calibration enhanced the ENOB value from 6.7 to 7.3. The right graph of 4.16shows the calibration enhancements of the system performance for different fabrication corners; nominal, Fast Fast (FF) and Slow Slow (SS).

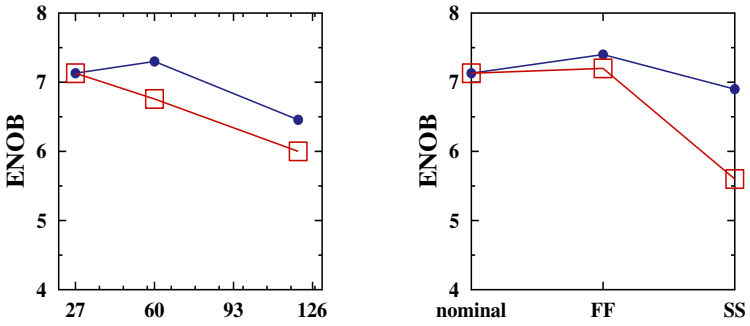


Figure 4.16: ENOB for the original circuit before calibration (square marker) and after calibration (dot marker). Simulation is for different temperature degrees (left) and different fabrication corners (right)

List of Publications

1-Ragab, O. K., H. Mostafa, and A. Eladawy, "TDC SAR Algorithm with Continuous Disassembly (SAR-CD) for Time-Based ADCs", *IEEE International Conference on Energy Aware Computing Systems and Applications (ICEAC 2015)*, Cairo Egypt, IEEE, pp. 1-4, 2015.

2-K. O. Ragab; H. Mostafa; A. Eladawy, "A Novel 10-bit 2.8mW Time-to-Digital Converter Design using SAR with Continuous Dis-assembly Algorithm", *IEEE Transactions on Circuits and Systems II: Express Briefs* , vol.PP, no.99, pp.1-1

References

[1] M. Amin, *Design of a Time Based Analog to Digital Converter*. PhD thesis, University of Waterloo, 2012.

[2] R. G. Lyons, *Understanding Digital Signal Processing, 3/E*. Pearson Education India, 2004.

[3] "Matlab data aquazition toolbox analog subsystem." accessed 18 May 2017.

[4] "Tms570ls3137 ep datasheet section 7.2.4.1." Accessed 19-5-2017.

[5] F. Maloberti, *Data converters*. Springer Science & Business Media, 2007.

[6] "Understanding piplelined adcs, tutorial 1023." https://www.maximintegrated.com/en/app-notes/index.mvp/id/1023. accessed 19-5-2017.

[7] *Analog Integrated Circuit Design*. John Wiley and Sons, 1997.

[8] S. Naraghi, *Time-based analog to digital converters*. PhD thesis, The University of Michigan, 2009.

[9] M. Wagih, "Design of time-based analog to digital converter (tb-adc) new design methodology for voltage-to-time (vtc) circuits," mathesis, Electronics and Communication department- Cairo University, 2015.

[10] F. Klass, "Semi-dynamic and dynamic flip-flops with embedded logic," in *VLSI Circuits, 1998 Symposium on*, pp. 108–109, June 1998.

[11] T. Hashimoto, H. Yamazaki, A. Muramatsu, T. Sato, and A. Inoue, "Time-to-digital converter with vernier delay mismatch compensation for high resolution on-die clock jitter measurement," in *VLSI Circuits, IEEE Symposium on*, pp. 166–167, June 2008.

[12] M. Zanuso, P. Madoglio, S. Levantino, C. Samori, and A. Lacaita, "Time-to-digital converter for frequency synthesis based on a digital bang-bang DLL," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 57, pp. 548–555, March 2010.

[13] T. Watanabe and T. Terasawa, "An all-digital ADC/TDC for sensor interface with TAD architecture in 0.18-μm digital CMOS," in *Electronics, Circuits, and Systems, 2009. ICECS 2009. 16th IEEE International Conference on*, pp. 219–222, Dec. 2009.

[14] P. Dudek, S. Szczepanski, and J. Hatfield, "A high-resolution CMOS time-to-digital converter utilizing a vernier delay line," *Solid-State Circuits, IEEE Journal of*, vol. 35, pp. 240–247, Feb. 2000.

[15] J. Yu, F. F. Dai, and R. Jaeger, "A 12-bit vernier ring time-to-digital converter in 0.13 CMOS technology," *Solid-State Circuits, IEEE Journal of*, vol. 45, pp. 830–842, April 2010.

[16] J.-P. Jansson, A. Mantyniemi, and J. Kostamovaara, "A CMOS time-to-digital converter with better than 10 ps single-shot precision," *Solid-State Circuits, IEEE Journal of*, vol. 41, pp. 1286–1296, June 2006.

[17] A. Mantyniemi, T. Rahkonen, and J. Kostamovaara, "A CMOS time-to-digital converter (TDC) based on a cyclic time domain successive approximation interpolation method," *Solid-State Circuits, IEEE Journal of*, vol. 44, pp. 3067–3078, Nov. 2009.

[18] H. Chung, H. Ishikuro, and T. Kuroda, "A 10-bit 80-MS/s decision-select successive approximation TDC in 65-nm CMOS," *Solid-State Circuits, IEEE Journal of*, vol. 47, pp. 1232–1241, May 2012.

[19] P. A. D. Holberg, *CMOS Analog Circuit Design*. Oxford University press, 1987.

[20] P. R. G. R. G. Meyer, *Analysis and design of analog integrated circuits*. John Wiley & Sons, 1990.

[21] S. R. N. G. C. Temes, S. R., *Delta-Sigma Data Converters :Theory, design, and simulation*. IEEE press, 1997.

[22] B. Leung, *DVLSI for wireless communication*. Prentice Hall Electronics and VLSI Series, 2002.

[23] H. Pekau, A. Yousif, and J. W. Haslett, "A cmos integrated linear voltage-to-pulse-delay-time converter for time based analog-to-digital converters," in *Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on*, pp. 4–pp, IEEE, 2006.

[24] H. Mostafa and Y. I. Ismail, "Highly-linear voltage-to-time converter (vtc) circuit for time-based analog-to-digital converters (t-adcs)," in *Electronics, Circuits, and Systems (ICECS), 2013 IEEE 20th International Conference on*, pp. 149–152, Dec 2013.

[25] S. Mahmoud, H. Salem, and H. Albalooshi, "An 8-bit, 10 KS/s, 1.87μW Successive Approximation Analog to Digital Converter in 0.25μm CMOS Technology for ECG Detection Systems," *Circuits, Systems, and Signal Processing*, pp. 1–21, January 2015.

[26] K. Ragab, H. Mostafa, and A. Eladawy, "TDC SAR algorithm with continuous disassembly (SAR-CD) for time-based ADCs," in *Energy Aware Computing Systems & Applications (ICEAC), 2015 International Conference on*, pp. 1–4, March 2015.

[27] K. O. Ragab, H. Mostafa, and A. Eladawy, "A novel 10-bit 2.8mw time-to-digital converter design using sar with continuous dis-assembly algorithm," 2016. IEEE Transactions on Circuits and Systems II: Express Briefs.

[28] S. Alahdab, A. Mantyniemi, and J. Kostamovaara, "A time-to-digital converter (TDC) with a 13-bit cyclic time domain successive approximation interpolator with sub-ps-level resolution using current DAC and differential switch," in *Circuits and Systems (MWSCAS), 2013 IEEE 56th International Midwest Symposium on*, pp. 828–831, Aug. 2013.

Appendix A

Appendix

A.1 SAR-CD Matlab code

In this section, 3 main functions are discussed:

1- Old SAR algorithm: presenting the old SAR algorithm which use the condition for the analog reference quantity. It is presented so that comparison to the new SAR-CD (next) is more clear. The functionality is presented in function "sar_standard(in, N_bits)".

2- New SAR-CD algorithm: presenting the new SAR-CD algorithm which moves the conditioning to the digital domain. The functionality is presented in fuction "sar_proposed(in , N_bits)".

3- A test bench which uses both the algorithm and compares the result for equal digital output. The functionality is presented next as Matlab code.

next we present

A.1.1 Traditional SAR Algorithm- Matlab function

%%%%%%%%%%%%% sar_standard.m %%%%%%%%%%%%%%%

```
function [ binary ] = sar_standard( in , N_bits )
%SAR_STANDARD Summary of this function goes here
% This function calculates the binary form for the input using the
% standard algorithm, Successive Approximation Register (SAR).
% inputs:
% in : input real value.
% N_bits : number of binary bit enough to represent the input.
% binary : output binary form. "binary(1:N_bits)"
```

```
% binary register for the standared algorithm binary = zeros(N_bits,1);
% input value for the standared algorithm value = in;
%% calculation for the binary form using the standared algorithm
for i=N_bits-1 :-1:0
if(value >= power(2,i)) % update the value for the next stage only if the input is bigger
value = value - power(2,i);
binary(i+1,1) = 1;
else
binary(i+1,1) = 0;
end
end
end
```

A.1.2 Noval SAR-CD Algorithm - Matlab function

```
%%%%%%%%%%%%% sar_cd.m %%%%%%%%%%%%%
function [ binary ] = sar_cd( in , N_bits )
% sar_cd Summary of this function goes here
% This function calculates the binary form for the input using the
% proposed algorithm, Successive Approximation Register with Continuous
% Dis-asssembly (SAR-CD).
% inputs: % in : input real value.
% N_bits : number of binary bit enough to represent the input.
% binary : output binary form. "binary(1:N_bits)"
% binary register for the proposed algorithm binary = zeros(N_bits+1,1);
% initialization value so that the bit correction for the MSB is correct binary(N_bits+1,1) =1; value = in;
for i=N_bits-1 :-1:0 % difference betweeen the input and the stage reference value
temp = (value - power(2,i));
% pass the absolute to the next stage irrespectively value = abs(temp)proposed
% performing initial bit evaluation using the input
binary(i+1,1)= (temp > 0);
% equality comparison depends on the previous bit result binary(i+2,1)
% (temp == 0) produce '1' if temp equal '0'or '0' otherwise
binary(i+1,1)= binary(i+1,1)+( (temp == 0) & binary(i+2,1) );
```

```
% performing bit correction to the calcualted bit
binary(i+1,1) = ~xor(binary(i+1,1), binary(i+2,1));
end
%Setting the bit register most bit to zero again after being used in bit
%correction for the MSB
binary(N_bits+1,1) =0;
end
```

A.1.3 Noval SAR-CD Algorithm - Matlab function

```
%%%%%%%%%%%%%% test bench %%%%%%%%%%%%%%
% This test bench is indended to verify the new algorithm, Successive Approximation Register with Continuous
% Dis-asssembly (SAR-CD) Vs the standard SAR algorithm. The is used side
% by side with the proof in the thesis to demostrate the
% correctness of the algorithm for all the numbers, real and integers.
% The test sweep the numbers between selected boundries. Then is calculates the
% binary form of each using the standard and the proposed algortihm. At last it
% checks the equivelance of both results produced by the proposed and the standard
% algorithm.
% select the minimum number to check
Min = 10000;
% select the maximum number to check
Max = 50000;
%%sweeping the range and compare both algorithms
for j = Min: Max value = j;
%calculating the minimum number of binary bits to present the input.
N_bits = ceil(log2(value) );
% Binary value using the standard algorithm
Binary_standard = sar_standard (value, N_bits);
% Binary value using the proposed algorithm
Binary_proposed = sar_standard (value, N_bits);
%% verifing the result by comparison
TRUE = 1;
```

```
FALSE =0;
equal = TRUE;
% initialize the comparison to TRUE as long as no bits are different
for i =1:N_bits
if(Binary_standard(i) ~= Binary_proposed(i))
equal = FALSE;
break;
end
end
%% Printing the result
if (equal == FALSE)
disp([ 'Proposed algorithm Fails in value: ' num2str(value)] );
else
disp([ 'Proposed algorithm succeeded: ' num2str(value)] );
end
end
```

A.2 Effective Number Of Bits (ENOB)

To calculate the effective number of bits for a given ADC, a sinusoidal signal can be used as an input. The sinesoidal signal amplitude should cover the dynamic range. As a sinusoidal signal is simply an impulse in the frequency domain, the signal can be extracted easily for the spectrum. The output digital stream should be first saved from Cadence Virtuoso then a Matlab code is used to calculate the IFFT and estimate the number of bits from the signal power.

To save the output, open the results browser from the ADE as shown:

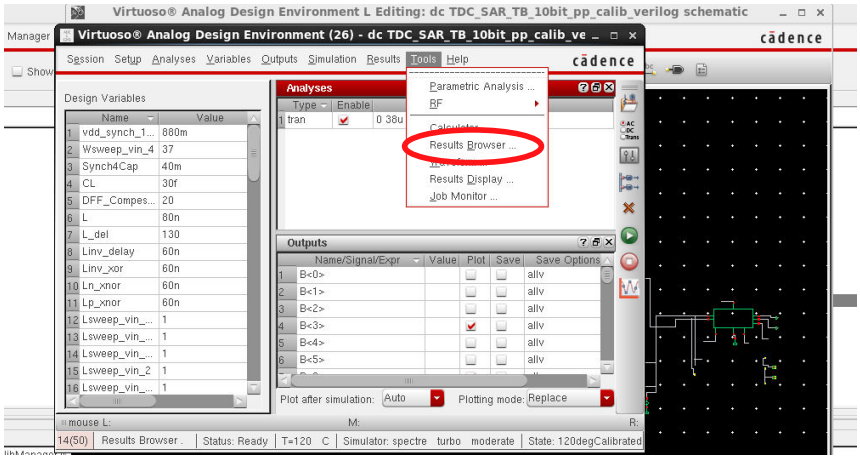


Figure A.1:

Then, right click the ADC output, “Dig_ot” (multiple signals can be selected by holding the Ctrl key) in our case, and select “Export” as shown:

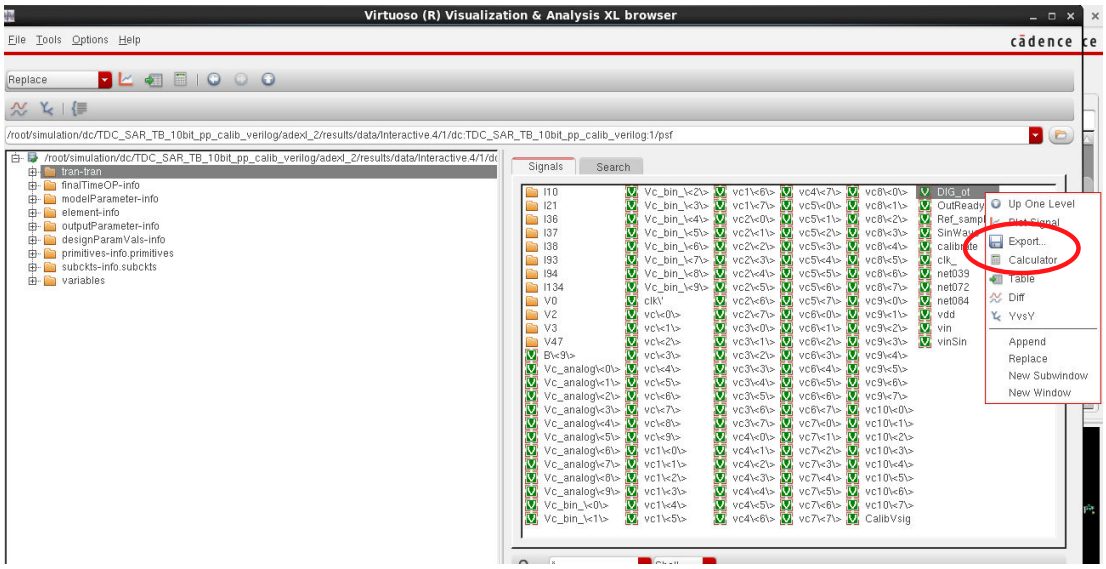


Figure A.2: Exporting the ADC output from the results browser

As Spactre simulator calculates continuous waveforms, the output should be sampled with the operation sampling frequency. The “Sampled Data” check box should be checked and the samping period corresponding to the mentioned sampling frequency should be witten in the “Step Size” box. In the shown figure the time period is 34ns which corresponds to 29.41Mhz. The “Start” point should be selected such that the circuit is expected to be in a stable operation (for example all the internal capacitances are charged). The “End” time is selected to be less than

the last simulation time and should be selected such that enough number of sampled are expected (in the shown example a 11us corresponds only to 323 sample). The selected format is “Matlab” (“.VCSV” format, Excel format, can be selected instead but different importing function to Matlab should be used then).

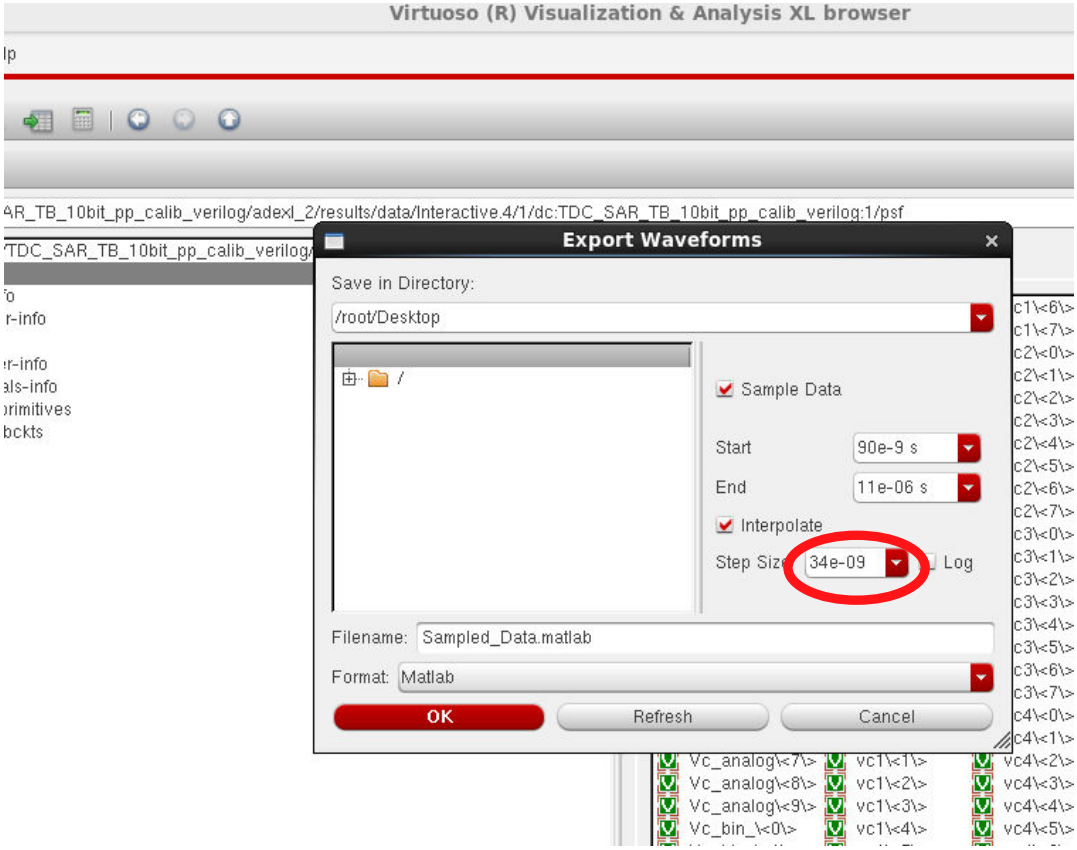


Figure A.3: Data should be sampled with the sampling frequency of operation

The following Matlab command can be used to imported the saved data file:

>> data=importdata('Sampled_Data.matlab');

Then “data” is a cell with 3 containers, “data” and “textdata” and “colheaders”. The latter two (“data.textdata” and “data.colheaders”) contains the name of the X and Y axis entries each variable and “data.data” contains the X and Y axis data for each variable. The following capture illustrates the contents of “data”, “data.data” and “data.textdata” respectively.

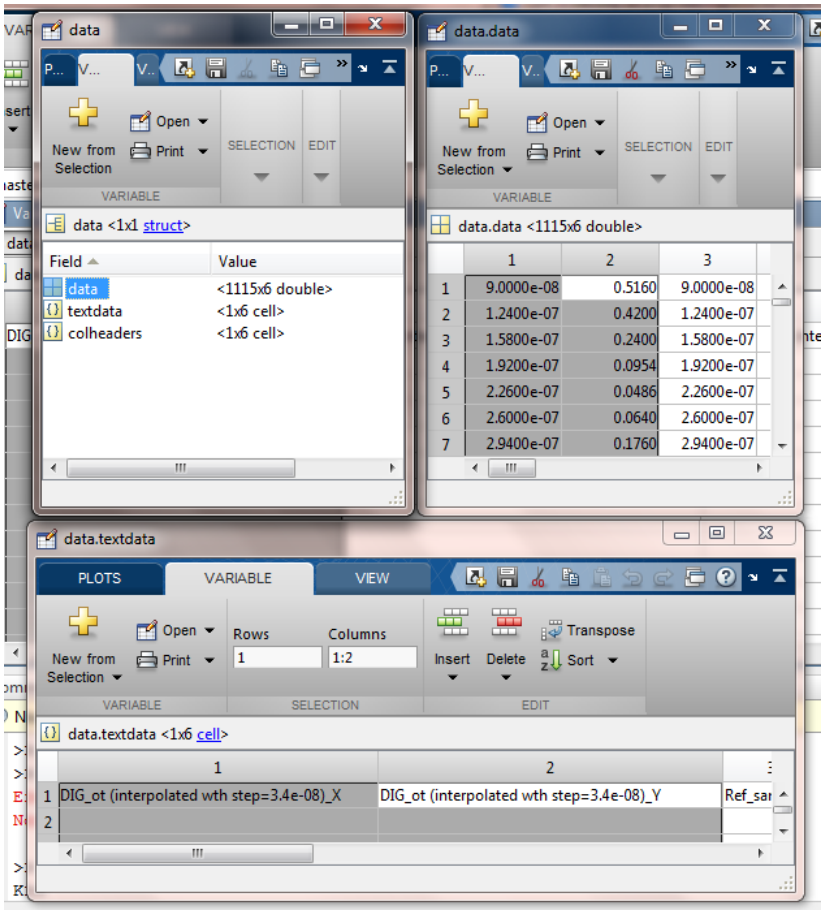


Figure A.4: The Matab data fomate cell contists of 3 elements “data”, “textdata” and “colheaders”

The desired samples stream are the Y axis of the firt variable, which is stored in the second row.

>> samples = data.data(:,2);

The time line or the X axis can be selected by:

>> A_axis = data.data(:,1);

The following Matlab Function is used to calculate the ENOB value based on signal specturm on the frequency domain.

%%%%%%%%%%%%% enob.m %%%%%%%%%%%%%

function [ENOB , SQNR , freq_db] = enob(samples ,Nbits)

% calculating the FFT

% the mean is subtracted to remove the DC component from the FFT output

% the "fftshift" function centers the FFT around Fs/2 figure;

>>freq = fftshift(abs(fft(samples - mean(samples) ,Nbits)));

>>freq_db = 20*log10(freq); figure; h =stem(freq_db);

% The signal is assumed to be presented in one peak (one peak)

>>Spwr = max(freq)^2;

% Spwr is multipled by 2 because there is 2 peaks centered arount Fs /2

>>Npwr = sum(freq.^2) - Spwr *2;

>>SQNR = 10*log10(Spwr / Npwr);

>>set(get(h,'BaseLine'),'BaseValue',-100);

% roughly, each 6db increase in the signal power means 1 bit increase in

% the ENOB value

>>ENOB = SQNR /6;

>>title([' FFT ',num2str(Nbits) , ' in db, SQR = ' ,num2str(SQNR) , 'ENOB ='',num2str(ENOB)]);

>>xlabel('Frequency (Hz)'); ylabel('signal(db) ');

>>end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

To use the enob funciton it is called simply by:

>>[ENOB , SQNR , freq_db] = enob(samples, N_fft);

Figure A.5 shows the target sine wave and figure A.6 shows the function output for a sine wave input with 63/N_FFT of the Sampling frequency, as N_FFT is 1024.

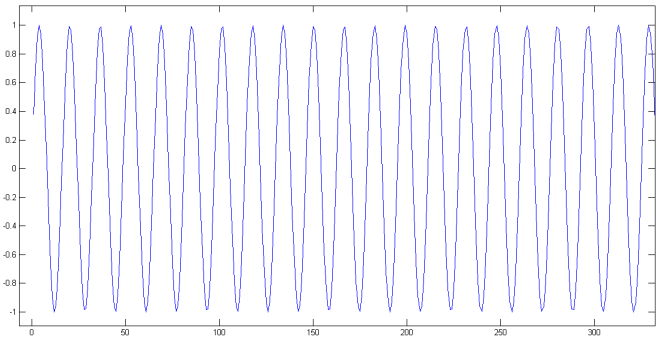


Figure A.5: Target Sine signal

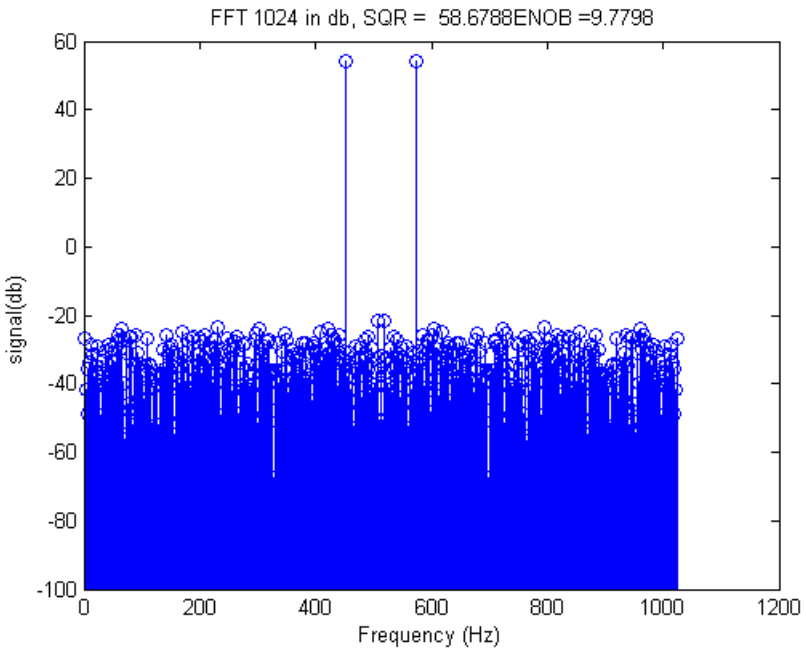


Figure A.6: Effective number of bits for a 10-bit quantized sine signal

A.3 Differential Non-Linearity (DNL) and Integral Non-Linearity (INL)

Calculating the ADC linearity includes examining the ADC output for each possible input. When examining every possible analog input signal is not possible, the input sweeping should choose the most reasonable input step that can speculate the system linearity. Figure A.7 shows a possible input ramp signal (blue) and the expected ideal ADC output (red) for a 4-bit system. Analog step for input sample is 0.05 of the full scale 16. Which means that there are 20 points for each digital step in the output staircase. In this tutorial ideal signals are presents (not from real ADC), however, the designer should replace the input and the output by real data from the target circuit simulation (as shown in figures A.1, A.2 and A.3). Matlab function is then used to calculate the DNL and INL.

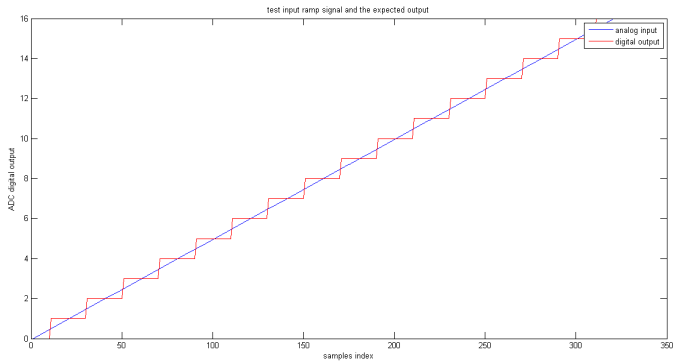


Figure A.7: Effective number of bits for a 10-bit quantized sine signal

The criteria (and parts of the Matlab code) is adopted from the EE247 class lab materials in Berkely university. Calculation of the DNL is based on calculating the number of samples for each step and making sure that the probability of locating a sample in a given digital step is equal for all the steps in the output staircase. The next Matlab function is used to calculate the DNL and INL. The Matlab function contains Matlab comments that describe the function.

%%%%%%%%%%%%% incldnl.m %%%%%%%%%%%%%

function [inl,dnl] = inldnl(x, delta)

% INLDNL compute INL and DNL from converter output x

% x output from ADC % delta spacing between codes. Default: 1

%

```
% Assumptions & limitations:
% - uniform quantizer
% - Input x is linear ramp
% compute histogram for the data, returns vectors n and xout containing
% the frequency counts and the bin locations.
[counts,centers] = hist(x, min(x):delta:max(x));
% eliminate end bins
counts(1) = [];
counts(end) = [];
% The mean presents the expected number of counts for each pin
dnl = counts/mean(counts) - 1;
% the INL is the accumulative sum of the DNL
inl = cumsum(dnl);
% Generate an equal spaced row vector from the start and end values of inl
inl = inl - linspace(inl(1), inl(end), length(inl));
   % plot result
   N = length(dnl);
   if N > 16
      fmt = 'r-';
   else
      fmt = 'ro:';
   end
   subplot(2,1,1);
   plot(1:N, dnl, fmt, [1 N], [1 -1; 1 -1], 'b:');
   fixfig;
   xlabel('bin');
   ylabel('DNL [in LSB]');
   maxdnl = ceil(max(dnl));
   axis([1 N floor(min(dnl)) maxdnl+1]);
   text(0.1*N+1, maxdnl+0.2, ...
   sprintf('avg=%.2g, std.dev=%.2g, range=%.2g', ... mean(dnl), std(dnl),
max(dnl)-min(dnl)));
   %title(sprintf('DNL and INL of %.1g Bit converter (from histogram test-
ing)', ...
   title(sprintf('DNL and INL ' ));
   subplot(2,1,2);
   %hold on;
```

```
    % Removing the offset from the inl value
    inl =inl-mean(inl);
    plot(1:N, (inl), 'b–', [1 N], [1 -1; 1 -1], 'b:');
    fixfig; xlabel('bin');
    ylabel('INL [in LSB]');
    maxinl = ceil(max(inl));
    axis([1 N floor(min(inl)) maxinl+1]);
    text(0.1*N+1, maxinl+0.2, ...
      sprintf('avg=%.2g, std.dev=%.2g, range=%.2g', ...
      mean(inl), std(inl), max(inl)-min(inl)));
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

To examine the algorithm, figure A.8 contains an example of and ADC output that contains nonlinear defects. The Matlab code is used to generate and plot the signal:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Generate a ramp signal from 0 to 16 with step 0.05
% Which means that each step contains 20 samples
samples = 0:.05:16;
```

% quantize(samples,bits) is a private funtion that quantize the input analog samples to the given number of bits.

% in this example then number of bits is 4. This function is a simple function that the designer can implement.

```
samples_q= quantize(samples',4);
plot(samples,'b'); hold on; plot(samples_q,'r');
title([' test input ramp signal and the expected output ']);
legend('analog input','digital output'); xlabel('samples index');
ylabel('ADC digital output');
```

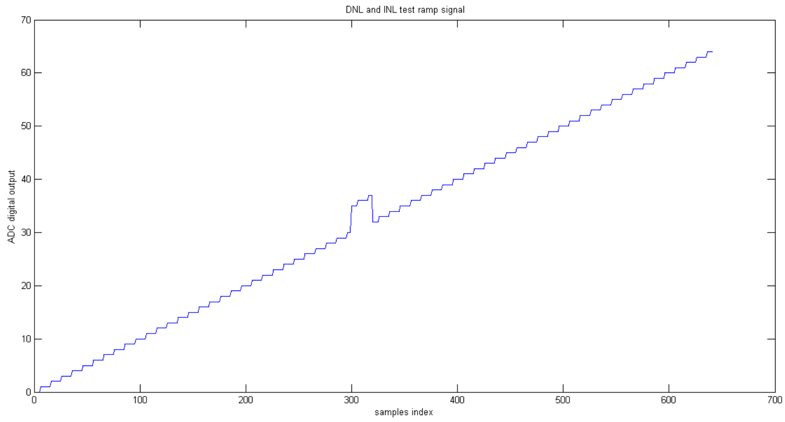


Figure A.8: example of ADC output with non-linear defect

% Introducting non-linear behavior is done by assigning the samples from 300 into 319

%incorrect values copied from samples 350 to 369 respectively

for i = 0: 19,

samples_q(300+i) = samples_q(300+i+50);

end; figure; plot(samples_q);

% plotting the DNL and INL

title([' DNL and INL test ramp signal']);

xlabel('samples index');

ylabel('ADC digital output');

%%

The following plot portrays the generated DNL and INL showing the non-linear behavior starting from the ruined samples.

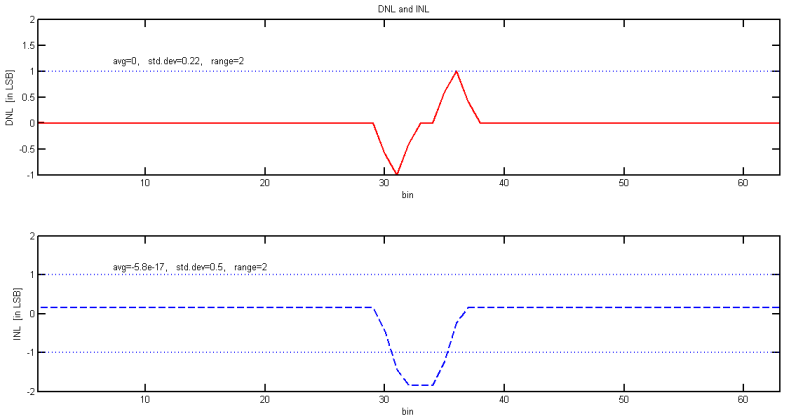


Figure A.9: The DNL (red) and INL(blue) plots for the example signal

A.4 Ideal VTC using Verilog-A

It is required to design a component to mimic the operation of a VTC to complete a full TADC system in adjacent with the target TDC. This compoenent should be used in simulation as cell view in Cadence Virtuoso. The required VTC should convert the input voltage to a modulated pulse for each sample. The modulation is done for the pulse width. The modulated pulses is then the input to the next stage, the target TDC.

The following Verilog-A code is used to convert input signal with range 0.4:0.6V to pulses of width ranges from 0:31.5ns.

/*****************************VTC_ideal.va*****************************/

module vtc_ideal(clk,in , out);

// The offset of the pulse width, can be changed to mimic real VTC of arbitrary minimum pulse width.

parameter real Poffset = 0*1e-12;// 65 LSB for 1024Fule scale

// ratio between the output pulse width in ps and the input value in volts.

parameter real Pslobe = 157.5 *1e-12; // 31500 =ps =31.5n for VFS=200mv

// minimum pulse in ps

parameter real LSB = 30.7e-12;

// used to speed up the simulation be increasing the simulation maximum step size when applicable

```
parameter real MaxStepOrder = 2000.0 ; // 50 times of the LSB
// The minimum value in mV
parameter integer Voffset = 400;// for VF =200mv
input in;// the component analog input
input clk; // The system clock
output out; // the component digital output
electrical in,clk,out;// the node type of of the pins of analog properties
integer start_time ;// the start time of the current sample pulse
integer out_v;
real end_time; // the start time
real time_t; // used for debugging
real stepSim; // current maximum simulation step
real stepMx; // maximum sinulation step == MaxStepOrder * LSB
analog begin //analog
@(initial_step)
begin //@(initial_step) , initialization assignemnts at the beginning of the simulation only
out_v =0;
end_time =0;
start_time =0;
stepMx = MaxStepOrder * LSB;
stepSim = stepMx;
end //@(initial_step)
$bound_step(stepSim);
@(cross(V(clk) -0.5,+1)) // at every ve+ edge of the clock do the folowing
begin //@(cross(V(clk) -0.5,+1))
//$display("step time now is :",stepSim);
// lower the maximum simulation time to curfully moitor the termination condition if(end_time <= $abstime) for accurate genration of the pulse width.
stepSim = LSB/2;
out_v=1;
time_t = $abstime; // save the current simulation time starting when the signal is assigned to 1
// end_time is the next time for the pulse to be low after going high for time depending on V(in)
end_time = ( V(in)*1000 - Voffset ) * Pslobe + Poffset;
```

```
//$display("Vin :%r, delay:%r, time now is:%r",V(in),end_time,time_t); // uncomment for debug
end_time = $abstime +end_time;
//$display("end_time :%r, Mx step%r",end_time,stepSim);// uncomment for debug
end //@(cross(V(clk) -0.5,+1))
if(end_time <= $abstime)
begin //if(end_time <= $abstime)
// increase the simulation maximum step to boost the simulation when no pulse is generated
stepSim=stepMx;
out_v =0;
end //if(end_time <= $abstime)
V(out) <+ transition(out_v ,0,10p,10p);
end //analog
endmodule
/*****************************************************************************/
```

The operation starts when a positive clock edge is detected when the pulse is assigned a high value. The current simulation time is saved (start_time) and the future simulation time when the signal is expected to be low is saved too (end_time). A forever “if“ condition is executed to monitor the real time simulation time when it reaches the value of “end_time” when the pulse is assigned low again.

It can be noted from the above code that is it optimized for maximum performance by changing the maximum simulation step when it is possible. This dramatically boots the simlation. Using an internal timer to calculate the period when the signal is high is not recommended as this increase the simulation time.

A.5 Calibration code Verilog

```
/*************************Calibration_SAR_TDC.v*************************/
//Verilog HDL for "dc", "calibration_ADC" "verilog"
module calibration_ADC_verilog_test(
clk,
DigIn,
Vsig,
// Analog control voltage for each stage
```

```
Vc1,Vc2,Vc3,Vc4,Vc5,Vc6,Vc7,Vc8,Vc9,Vc10,
// digital representation for the control voltage for each stage each of 2^(n_bits_calib-1) Full scale
Vc_bin1,Vc_bin2,Vc_bin3,Vc_bin4,Vc_bin5,Vc_bin6,Vc_bin7,Vc_bin8,Vc_bin9,Vc_bin10
);
// Begin the calibration after waitClocks untill all the internal capacitances are charged
parameter waitClocks = 5 ;
// The quantization number of bits defining the resolution of the control signal/voltage.
parameter n_bits_calib = 8 ;
parameter n_bits_ADC = 10 ;// number of bits of the target ADC
parameter pVc_start = 0.5 ; // control voltage offset in Volt
parameter pVc_end = 1.2 ; // Maximum control voltage in Volt
parameter pVsig_start = 0.4 ; // ADC input signal offset in Volt
parameter pVsig_end = 0.6 ;// ADC input signal Maximum in Volt
input clk;// System clock
input [9:0] DigIn; // The monitored ADC output. It is an input for the calibration component
output Vsig; // The ADC input signal
output Vc1; // control voltage for stage 1
output Vc2; // control voltage for stage 2
output Vc3;// control voltage for stage 3
output Vc4; // control voltage for stage 4
output Vc5;// control voltage for stage 5
output Vc6; // control voltage for stage 6
output Vc7;// control voltage for stage 7
output Vc8; // control voltage for stage 8
output Vc9;// control voltage for stage 9
output Vc10;// control voltage for stage 10
output Vc_bin1;// control voltage digital form for stage 1
output Vc_bin2;// control voltage digital form for stage 2
output Vc_bin3;// control voltage digital form for stage 3
output Vc_bin4;// control voltage digital form for stage 4
output Vc_bin5;// control voltage digital form for stage 5
output Vc_bin6;// control voltage digital form for stage 6
output Vc_bin7;// control voltage digital form for stage 7
```

```
output Vc_bin8;// control voltage digital form for stage 8
output Vc_bin9;// control voltage digital form for stage 9
output Vc_bin10;// control voltage digital form for stage 10
wire [7:0]Vc1;wire [7:0]Vc2;wire [7:0]Vc3;wire [7:0]Vc4;wire [7:0]Vc5;
wire [7:0]Vc6;wire [7:0]Vc7;wire [7:0]Vc8;wire [7:0]Vc9;wire [7:0]Vc10;
wire [7:0]Vc_bin1;wire [7:0]Vc_bin2;wire [7:0]Vc_bin3;wire [7:0]Vc_bin4;wire
[7:0]Vc_bin5;
wire [7:0]Vc_bin6;wire [7:0]Vc_bin7;wire [7:0]Vc_bin8; wire [7:0]Vc_bin9;wire
[7:0]Vc_bin10;
wire [7:0]Vsig;
integer out_v;
integer tempCeil;// temp ingeter used for ceiling operations
real Vc_fs;// local variable to calculate the full scale of the control voltage
real Vsig_fs;
real temp;
real Vsig_value;
real Vc_num[9:0];
integer Vc_num_intg[9:0];
real DigIn_num[9:0];
real Vc_value[9:0];
integer BitCalib_idx ,BitADC_idx, Vsig_num,clkDiv3,idx;
initial begin clkDiv3 =0 ;
BitCalib_idx = n_bits_calib-1;
BitADC_idx =n_bits_ADC-1;
Vc_fs = pVc_end -pVc_start;
// calculating the input signal full scale
Vsig_fs = pVsig_end -pVsig_start ;
// assigned half the control voltage in the middle of the full scale range
Vc_num[9] = 2**(n_bits_calib-1);Vc_num[8] = 2**(n_bits_calib-1);
Vc_num[7] = 2**(n_bits_calib-1);
Vc_num[6]=2**(n_bits_calib-1);Vc_num[5] = 2**(n_bits_calib-1);Vc_num[4]
= 2**(n_bits_calib-1);
Vc_num[3] = 2**(n_bits_calib-1);Vc_num[2] = 2**(n_bits_calib-1);
Vc_num[1] = 2**(n_bits_calib-1);
Vc_num[0] = 2**(n_bits_calib-1);
Vc_num_intg[9] = Vc_num[9]; Vc_num_intg[8] = Vc_num[8]; Vc_num_intg[7]
= Vc_num[7];
```

```
    Vc_num_intg[6] = Vc_num[6]; Vc_num_intg[5] = Vc_num[5]; Vc_num_intg[4]
= Vc_num[4];
    Vc_num_intg[3] = Vc_num[3]; Vc_num_intg[2] = Vc_num[2]; Vc_num_intg[1]
= Vc_num[1];
    Vc_num_intg[0] = Vc_num[0];
    Vc_value[9] = (Vc_num[9] * Vc_fs) / (2**n_bits_calib) +pVc_start;
    Vc_value[8] = (Vc_num[8] * Vc_fs) / (2**n_bits_calib) +pVc_start;
    Vc_value[7] = (Vc_num[7] * Vc_fs) / (2**n_bits_calib) +pVc_start;
    Vc_value[6] = (Vc_num[6] * Vc_fs) / (2**n_bits_calib) +pVc_start;
    Vc_value[5] = (Vc_num[5] * Vc_fs) / (2**n_bits_calib) +pVc_start;
    Vc_value[4] = (Vc_num[4] * Vc_fs) / (2**n_bits_calib) +pVc_start;
    Vc_value[3] = (Vc_num[3] * Vc_fs) / (2**n_bits_calib) +pVc_start;
    Vc_value[2] = (Vc_num[2] * Vc_fs) / (2**n_bits_calib) +pVc_start;
    Vc_value[1] = (Vc_num[1] * Vc_fs) / (2**n_bits_calib) +pVc_start;
    Vc_value[0] = (Vc_num[0] * Vc_fs) / (2**n_bits_calib) +pVc_start;
    $write("Vc_value[%d] init is %e\n",9,Vc_value[9]);
    $write("Vc_value[%d] init is %e\n",8,Vc_value[8]);
    $write("Vc_value[%d] init is %e\n",7,Vc_value[7]);
    $write("Vc_value[%d] init is %e\n",6,Vc_value[6]);
    $write("Vc_value[%d] init is %e\n",5,Vc_value[5]);
    $write("Vc_value[%d] init is %e\n",4,Vc_value[4]);
    $write("Vc_value[%d] init is %e\n",3,Vc_value[3]);
    $write("Vc_value[%d] init is %e\n",2,Vc_value[2]);
    $write("Vc_value[%d] init is %e\n",1,Vc_value[1]);
    $write("Vc_value[%d] init is %e\n",0,Vc_value[0]);
    Vsig_num = Vsig_num +(2**BitADC_idx);
    Vsig_value = (Vsig_num * Vsig_fs)/(2**n_bits_ADC) + pVsig_start;
    $write("Vsig_value is initialized to :%e\n",Vsig_value);
    end
    always @ (posedge(clk))
    begin
    clkDiv3 = clkDiv3 +1;
    $write("clkDiv3 is :%d",clkDiv3);
    DigIn_num[9] = (DigIn[9]);
    DigIn_num[8] = (DigIn[8]);
    DigIn_num[7] = (DigIn[7]);
    DigIn_num[6] = (DigIn[6]);
```

```
DigIn_num[5] = (DigIn[5]);
DigIn_num[4] = (DigIn[4]);
DigIn_num[3] = (DigIn[3]);
DigIn_num[2] = (DigIn[2]);
DigIn_num[1] = (DigIn[1]);
DigIn_num[0] = (DigIn[0]);
if (clkDiv3 >= waitClocks)
begin
if(DigIn_num[BitADC_idx] > 0.5)
begin
$write("DigIn high detected at :%e \n",$abstime);
tempCeil = Vc_num[BitADC_idx] - (2**(BitCalib_idx-1)) ;
$write("decrease Vc_num to :%d\n",tempCeil);
end
else
begin
tempCeil = Vc_num[BitADC_idx] + (2**(BitCalib_idx-1)) ;
$write("DigIn is low at :%e \n",$abstime);
$write("increase Vc_num to :%d\n",tempCeil);
end
Vc_num[BitADC_idx] =tempCeil;
if (BitCalib_idx == 0 )
begin
if (BitADC_idx == 0)
begin // end of calibration at time
$abstime (BitCalib_idx==0) &&(BitADC_idx == 0)
$write("end of calibration at time %e \n",$abstime);
$write("output Vc[%d] : %e ,which corresponds to: %e volt\n",9,Vc_num[9],
Vc_value[9]);
$write("output Vc[%d] : %e ,which corresponds to: %e volt\n",8,Vc_num[8],
Vc_value[8]);
$write("output Vc[%d] : %e ,which corresponds to: %e volt\n",7,Vc_num[7],
Vc_value[7]);
$write("output Vc[%d] : %e ,which corresponds to: %e volt\n",6,Vc_num[6],
Vc_value[6]);
$write("output Vc[%d] : %e ,which corresponds to: %e volt\n",5,Vc_num[5],
Vc_value[5]);
```

```
    $write("output Vc[%d] : %e ,which corresponds to: %e volt\n",4,Vc_num[4],
Vc_value[4]);
    $write("output Vc[%d] : %e ,which corresponds to: %e volt\n",3,Vc_num[3],
Vc_value[3]);
    $write("output Vc[%d] : %e ,which corresponds to: %e volt\n",2,Vc_num[2],
Vc_value[2]);
    $write("output Vc[%d] : %e ,which corresponds to: %e volt\n",1,Vc_num[1],
Vc_value[1]);
    $write("output Vc[%d] : %e ,which corresponds to: %e volt\n",0,Vc_num[0],
Vc_value[0]);
    end
    else
    begin //(BitCalib_idx==0) &&(BitADC_idx != 0)
    BitADC_idx = BitADC_idx -1;
    clkDiv3 = waitClocks-3; // wait 1 clock till the next bit is evaluated
    Vsig_num = Vsig_num + (2**BitADC_idx);
    Vsig_value = (Vsig_num * Vsig_fs)/(2**n_bits_ADC) + pVsig_start;
    end
    BitCalib_idx = n_bits_calib-1;
    $write("output Vc for bit %d : %e ,which corresponds to: %e
volt\n",BitADC_idx,Vc_num[BitADC_idx], Vc_value[BitADC_idx]);
    Vc_num[BitADC_idx] = (2**BitCalib_idx);
    end
    else
    begin //(BitCalib_idx!=0) BitCalib_idx = BitCalib_idx -1;
    $write("Calibration for ADC bit %d ,Calibration bit %d \n",BitADC_idx,BitCalib_idx);
    end
    Vc_value[BitADC_idx] = (Vc_num[BitADC_idx]/(2**n_bits_calib) * Vc_fs)
+pVc_start;
    end
    else
    begin// if (clkDiv3 >= waitClocks)
    $write("calibration is starting in %d clocks\n",(waitClocks-clkDiv3)) ;
    end // if (clkDiv3 >= waitClocks)
    Vc_num_intg[BitADC_idx] = Vc_num[BitADC_idx];
    end // end of @(cross(V(clk) -0.5,+1))
    assign Vc_bin1 = Vc_num_intg[0];
```

```
assign Vc_bin2 = Vc_num_intg[1];
assign Vc_bin3 = Vc_num_intg[2];
assign Vc_bin4 = Vc_num_intg[3];
assign Vc_bin5 = Vc_num_intg[4];
assign Vc_bin6 = Vc_num_intg[5];
assign Vc_bin7 = Vc_num_intg[6];
assign Vc_bin8 = Vc_num_intg[7];
assign Vc_bin9 = Vc_num_intg[8];
assign Vc_bin10= Vc_num_intg[9];
assign Vc1 = Vc_num_intg[0];
assign Vc2 = Vc_num_intg[1];
assign Vc3 = Vc_num_intg[2];
assign Vc4 = Vc_num_intg[3];
assign Vc5 = Vc_num_intg[4];
assign Vc6 = Vc_num_intg[5];
assign Vc7 = Vc_num_intg[6];
assign Vc8 = Vc_num_intg[7];
assign Vc9 = Vc_num_intg[8];
assign Vc10= Vc_num_intg[9];
endmodule
```

A.6 Corners Simulation and analysis

In the following tutorial, the simulation steps for temperature and fabrication corners are discussed. The simulation can be used to target one or more simulation fabrication corners. The following targets 3 different tempratures degrees; 27, 60 and 120 celecuis degree. And it targets 4 Fabrication corners; SS (slow-slow), FF (Fast-Fast) , SF (Slow-Fast), FS (Fast-Slow) and normal. It will be shown how the simulation setup is performed and how the results are extracted to be analyzed in Matlab.

1- Launch the ADE XL from Virtuoso schematic Editor for the target circuit.

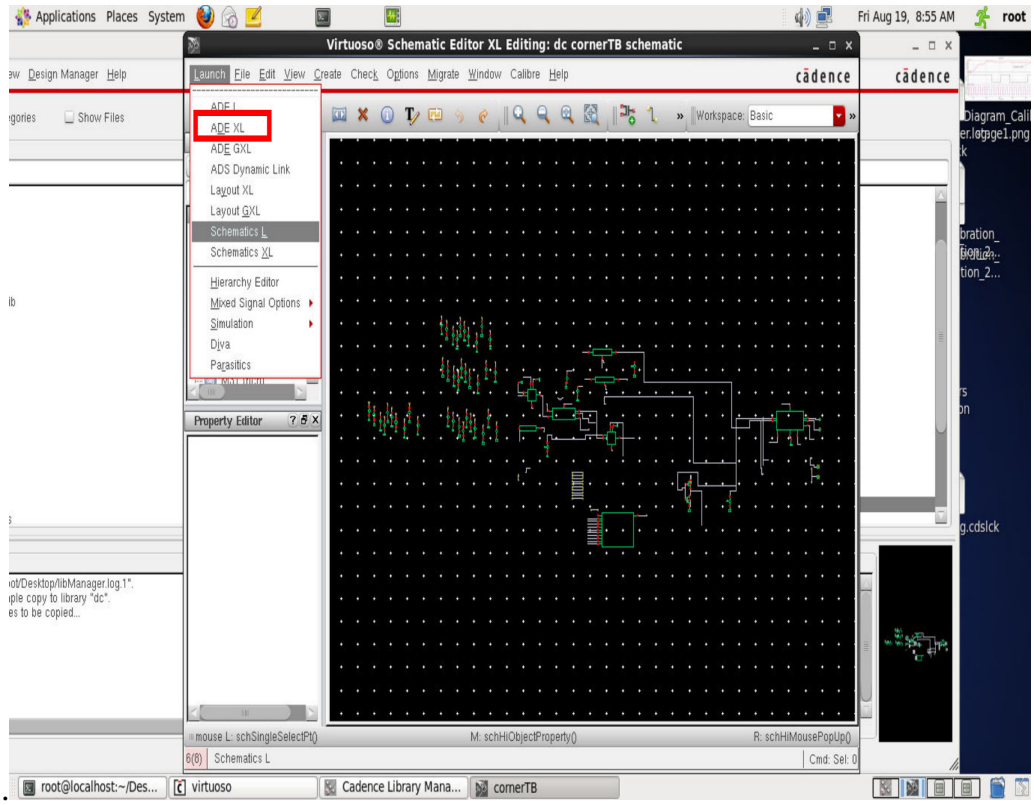


Figure A.10: Running ADE XL from the target circuit schematic.

The following window will appear. The “Data View” window shows the current test, global variables and parameters and the corners to be simulated. The “Run Summary” window shows the current running simulation and number of corners.

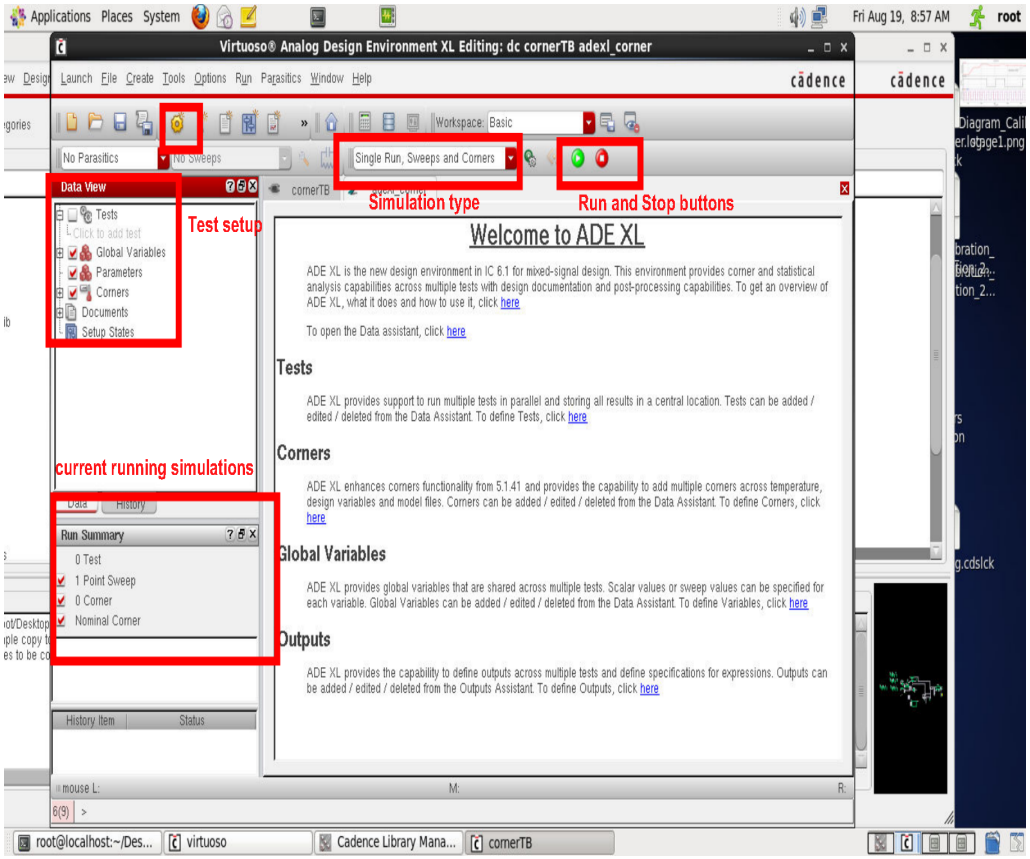


Figure A.11: Running ADE XL from the target circuit schematic.

2-To create a new test, choose “create->test”, choose the yellow icon or simply click “click to add test” from “Tests” in the Data View window. Choose the target circuit to be opened.

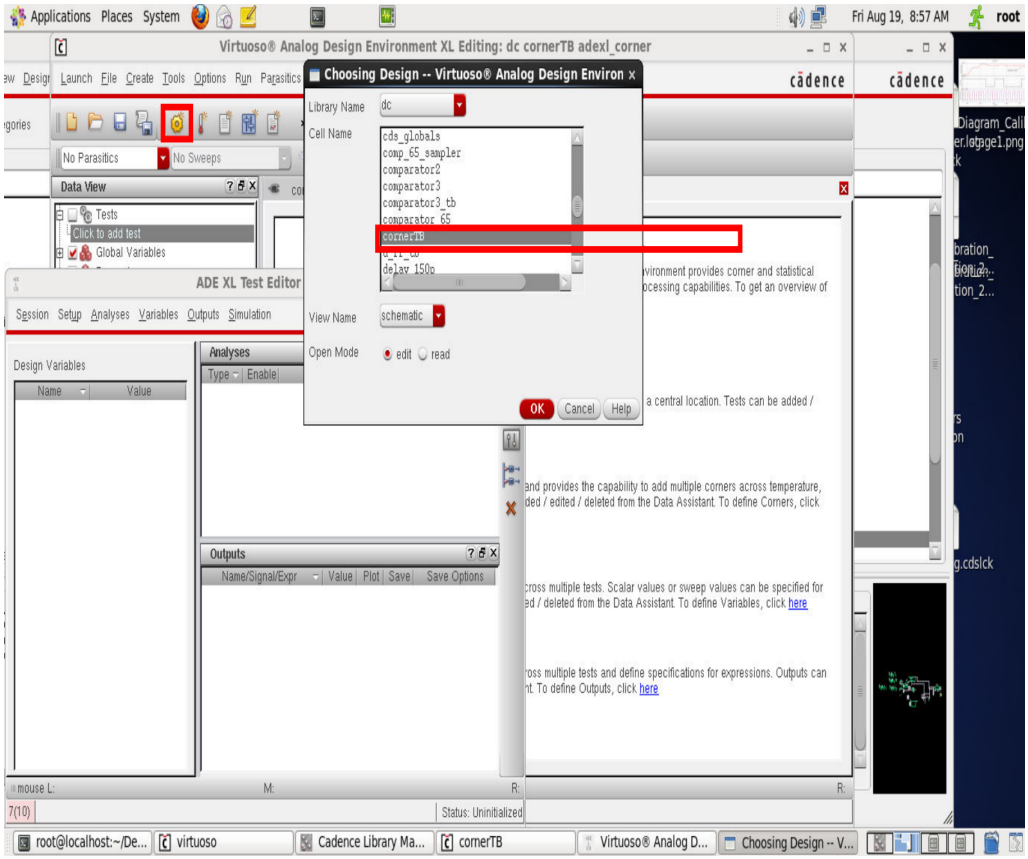


Figure A.12: Creating a new test for corners simulation

3- The ADE XL window will appear. You may configure a new simulation of load a saved state from “Session->Load State”. Make sure that the simlation configurations are correct as used to be in a regular simulation setup except the global variables which should be configured from “Global variables” in “Data View” window as changs to Variables in the ADE XL window have no effect.

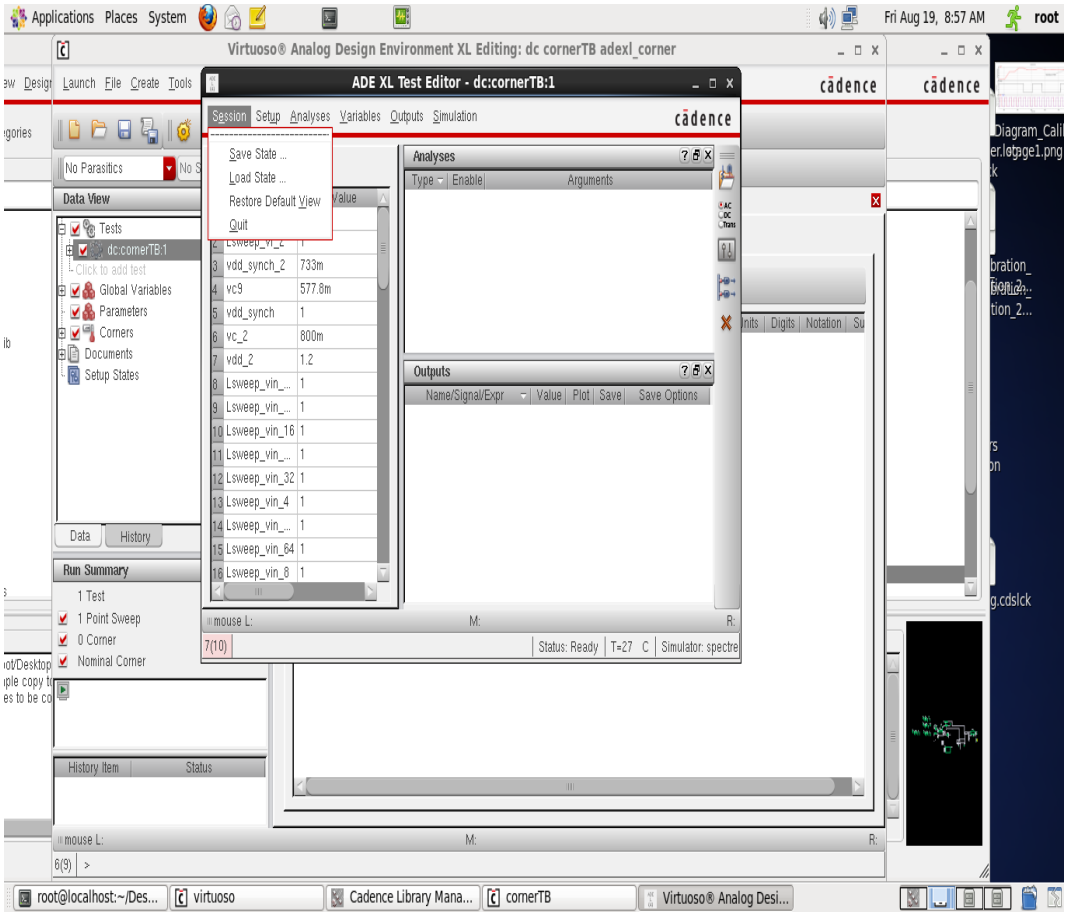


Figure A.13: Creating a new test for corners simulation

Each corner is defined as a set of different configurations for the design kit. To show how the model library defines them, from Setup tab in the ADE XL window, choose “Model Library Setup”. One may see a view like the one shown. It can be noticed that all the models are from the same file, however, specific sections are only selected for the current simulation setup. Each corner simulation chose different Section from the model file.

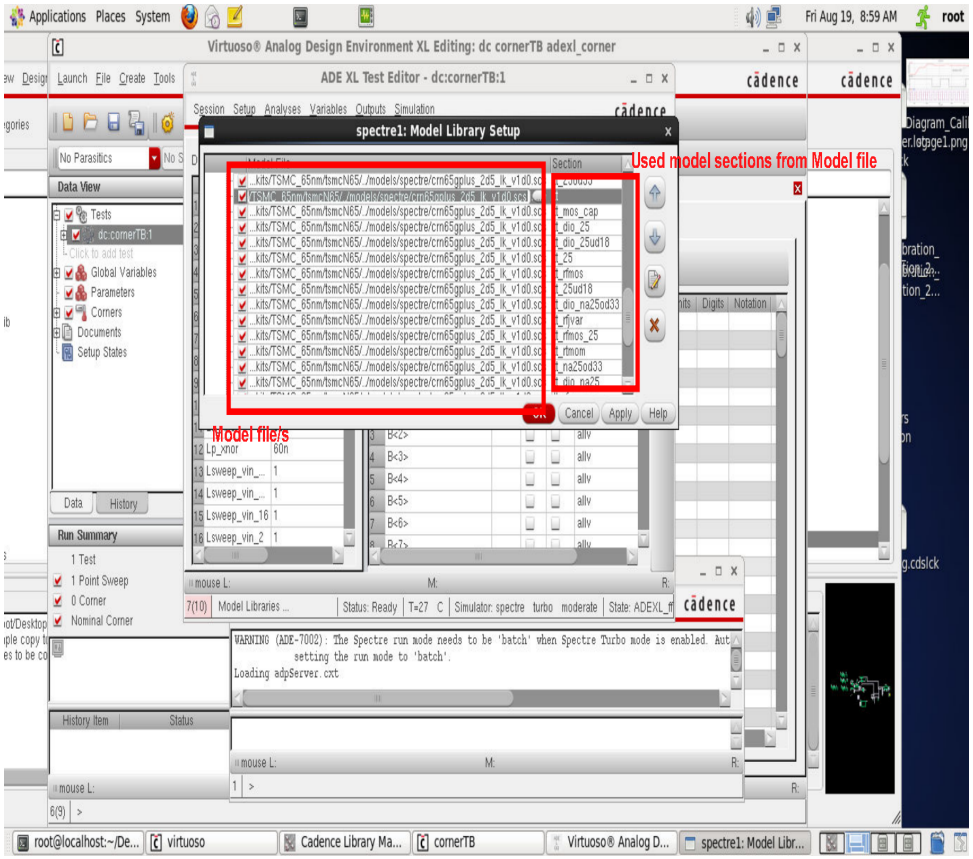


Figure A.14: Model library setup

Locating the library model setup from the file system, “section tt” can be found in the Model file.

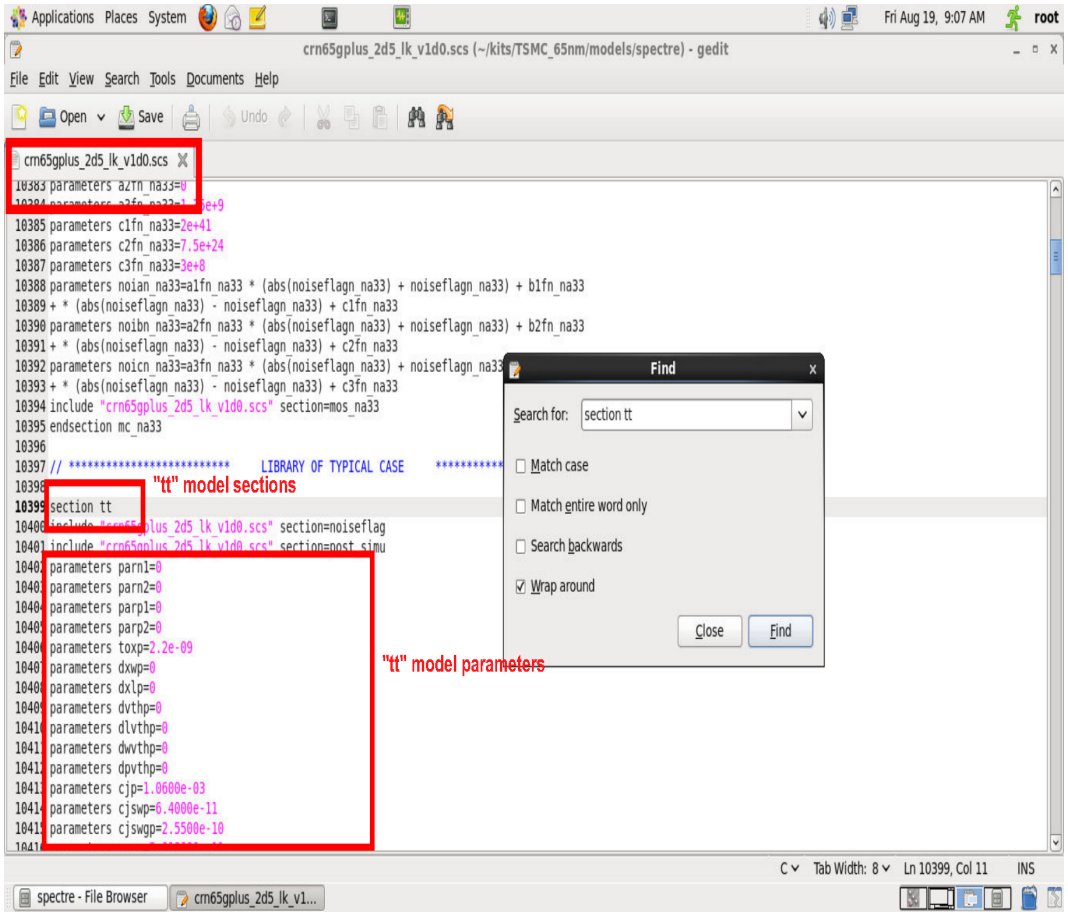


Figure A.15: Example of model file for TSMC13rf design kit for “tt” corner configuration (nominal)

4- After the previous step, the common simulation setup configurations are ready. Now, to chose the corners to be simulated, click “Click to add corner” from “Corners” in the Data View window. The following window will appear. Name a new corner in “Corner Name”. The different target temprature can be selected from “Temprature” in “Varables/Prameters”.

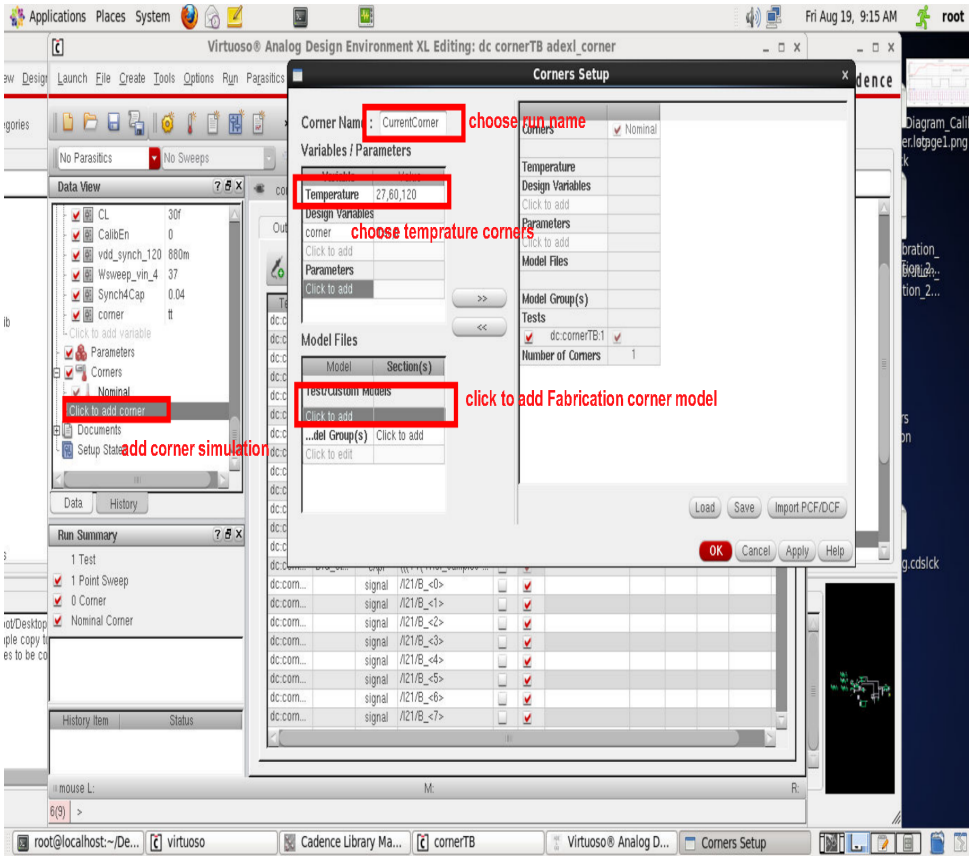


Figure A.16: Selecting the target temprature and fabrication corners

5- From “Mode Files” window click “click to add” under “Test/Custom Models”. The following window will appear. Click “import from Tests” to load all the models.

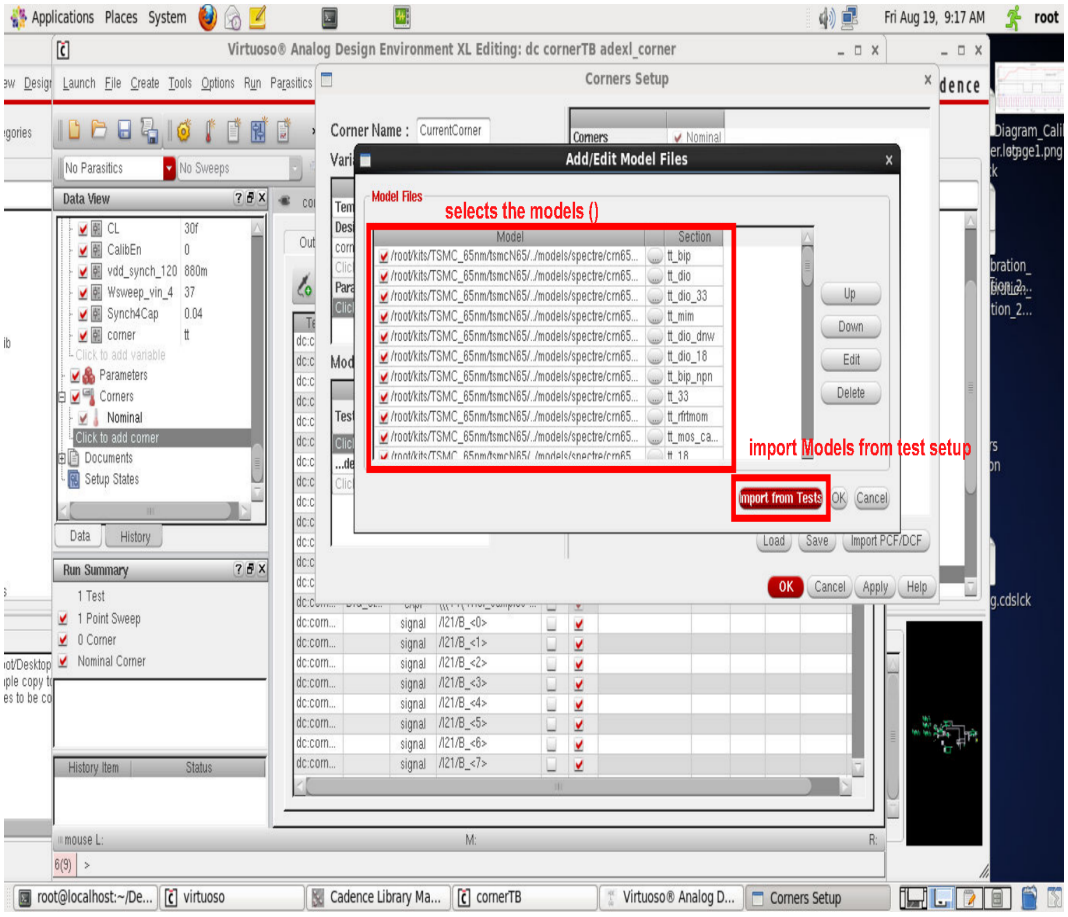


Figure A.17: loading the simulation models from the test setup

choose only the ones needed in the current simulation.

5- Create one or more corner setup. In this tutorials, 5 fabrication corners are selected; “tt”, “ff”, “ss”, “sf” and “fs”. Each with 3 different tempratures; 27, 60 and 120 Celecuis degrees. The total number of simulation runs are 5*3= 15 simulation run. When finished press Ok.

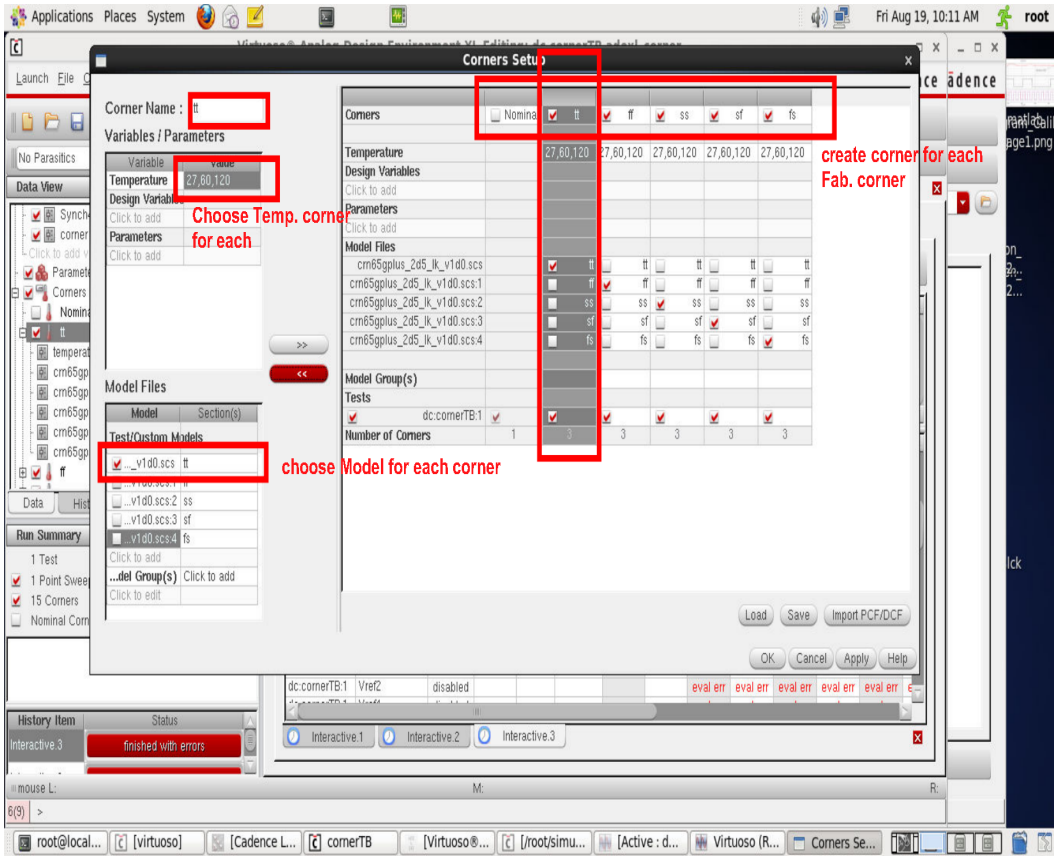


Figure A.18: Configuring the simulation temprature and fabrication corners

6- After the previous step, the target corners are shown in the Data View window. We are now ready to run the simulation.

To run the simulation, press the green play button. The simulation progress is shown under “History Item” window.

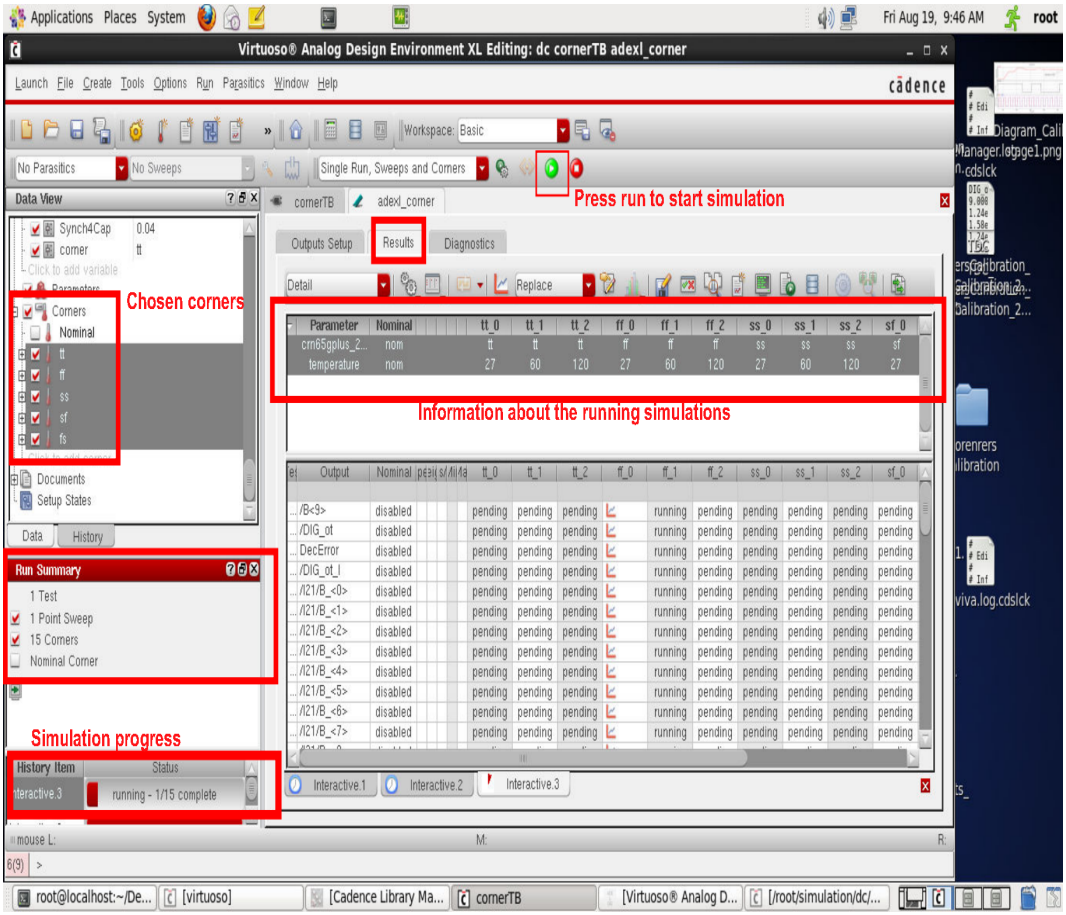


Figure A.19: Start and monitor the simulation

7- When the simulation is finished, plotting one or more of the output results signal can be selected from the “Results” tab and right click to the target signal and choose “Plot All Corners”. Another way is to select results icon and choose the signal to plot.

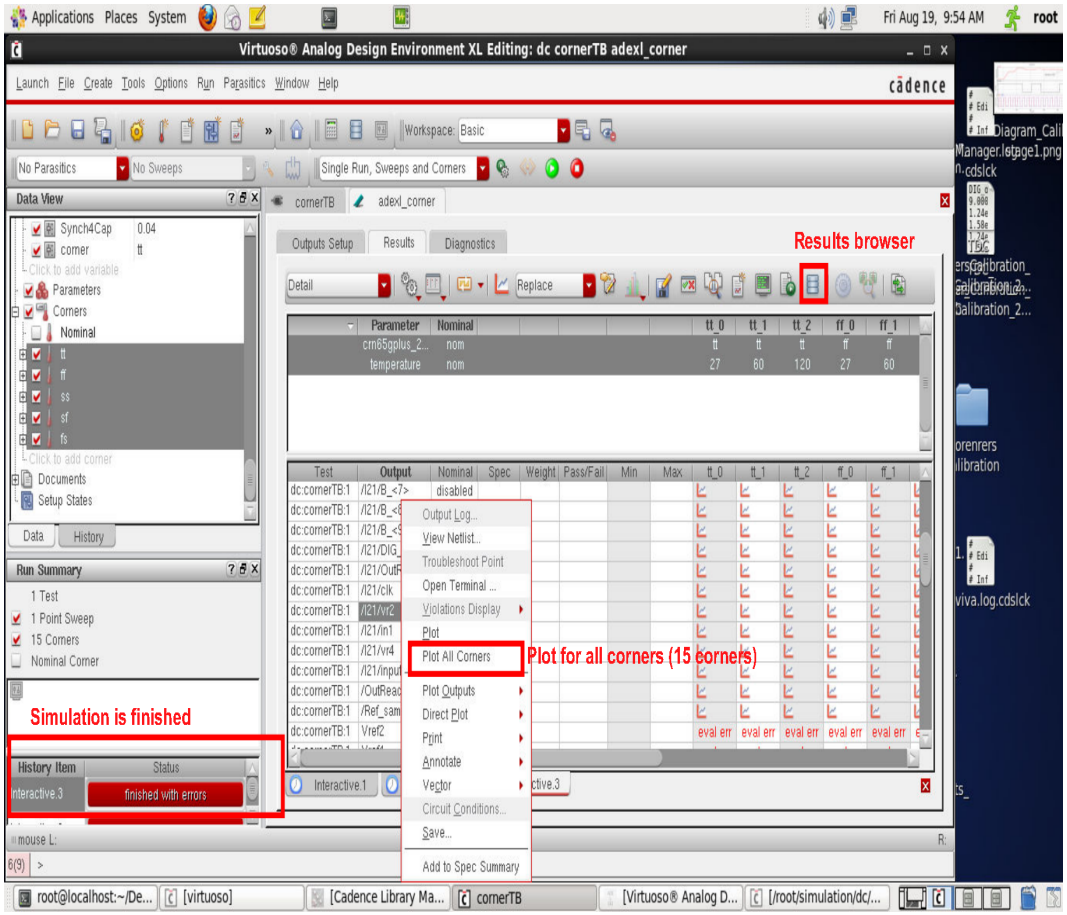


Figure A.20: Plotting the simulation results for all the corners

The following graph shows the results for the signal “/I21/vr2” for all the target simulation corners. It is shown how each corner affects the signal rising shape.

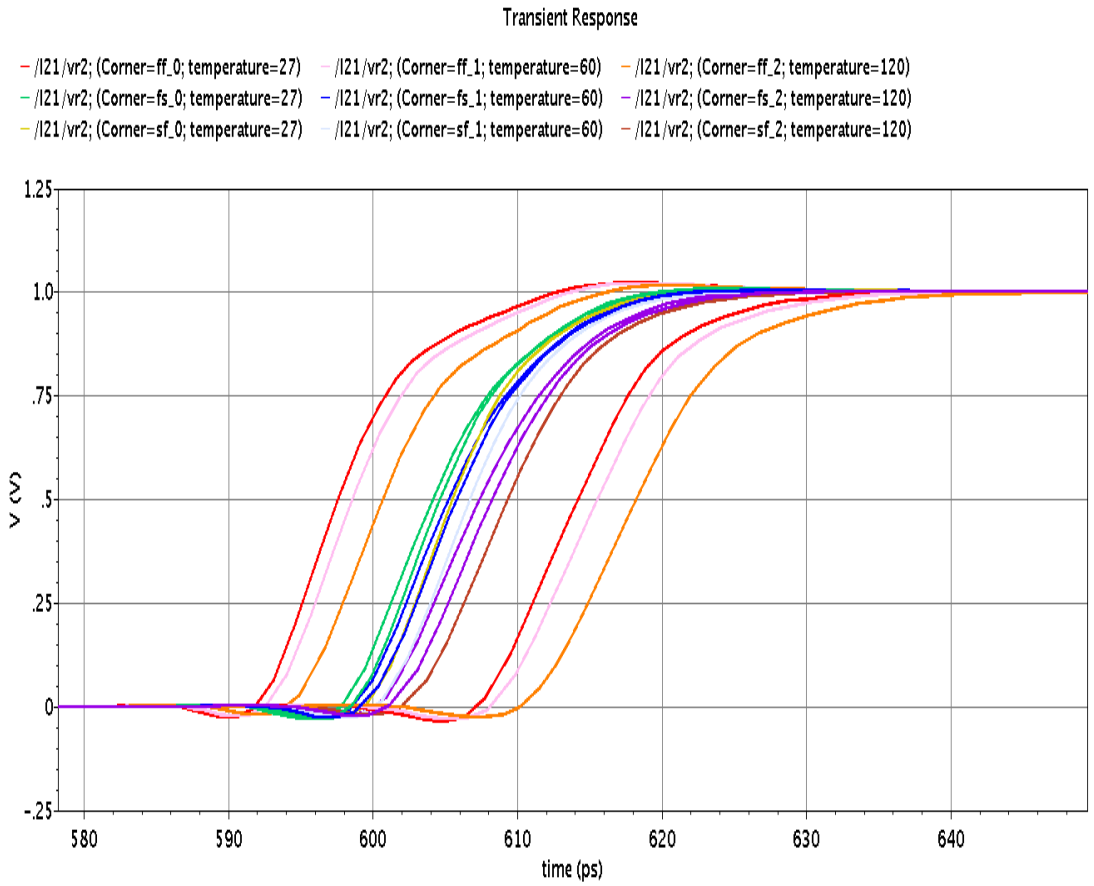


Figure A.21: Plotting the simulation results for all the corners

5- To export the results for Matlab, open the results from and right click on the target signal and choose “Export”.

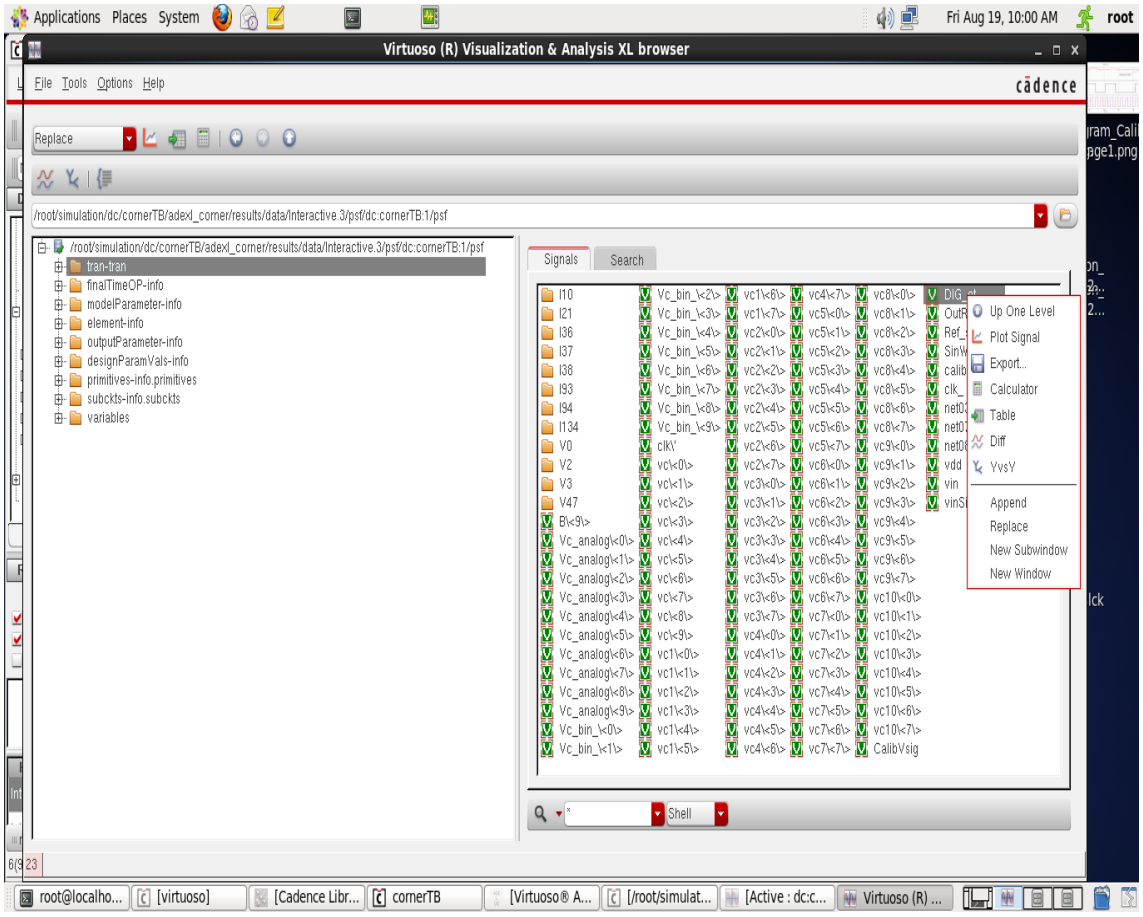


Figure A.22: Exporting signals to Matlab

From the Export Waveforms, the target signal can be sampled. Choose the "start" and "end" for the exported part. Then choose the sampling period "Step Size". Name the file with ".matlab" extension.

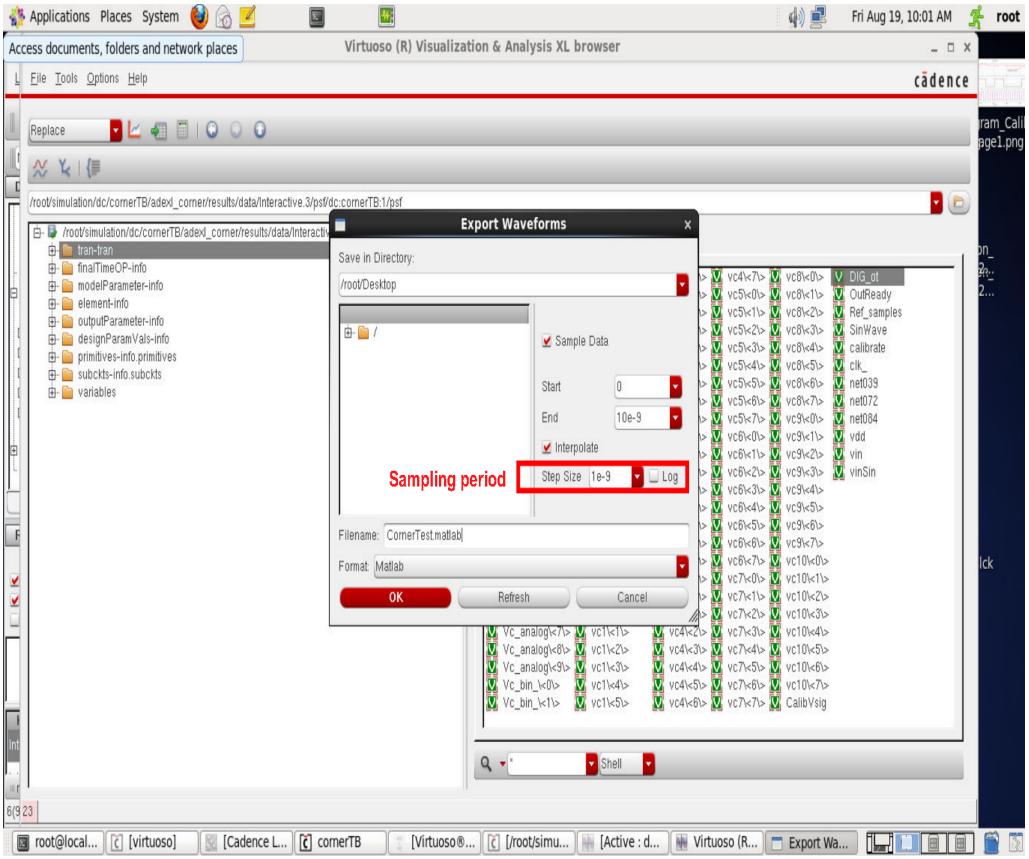


Figure A.23: Exporting signals to Matlab

The following is the exported file view to save 10 samples for all the corners. The highlighted part presents the 'X' and 'Y' for signal "DIG_ot" for corner "tt" and temprature 27 degree.

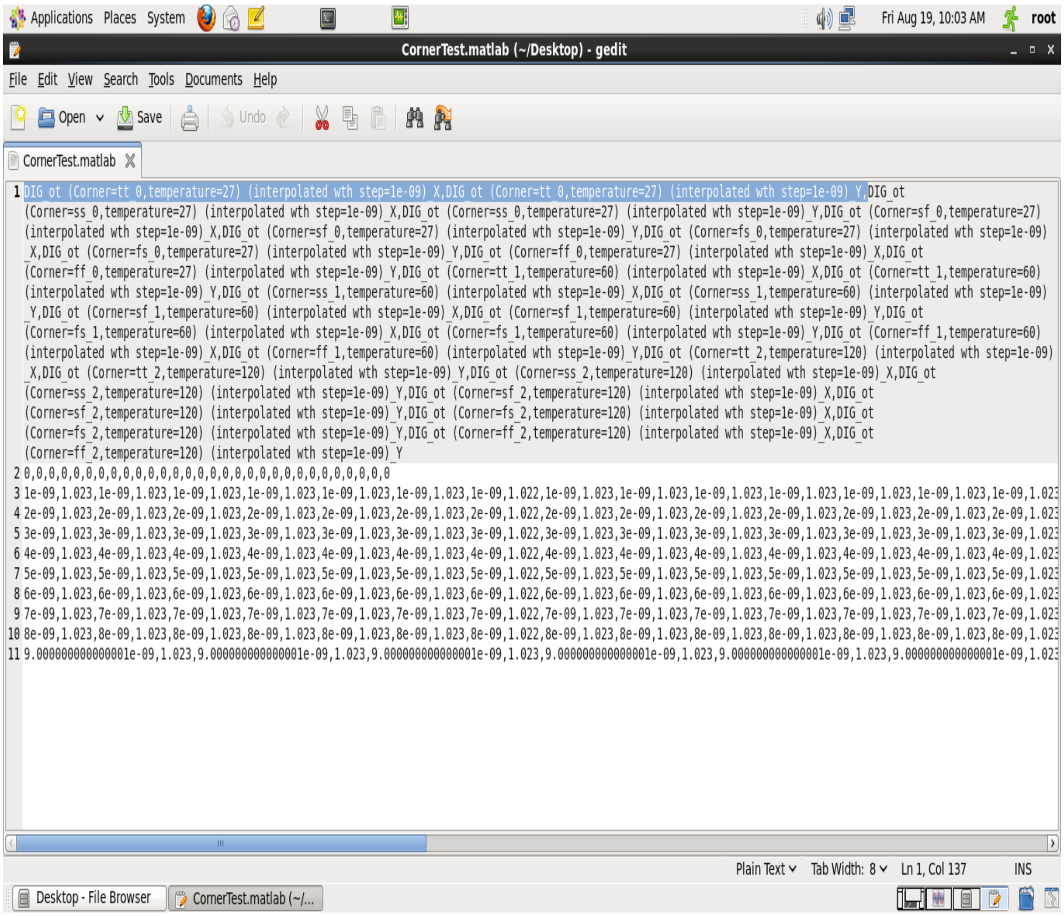


Figure A.24: Exported Matlab file view

To import the data to Matlab, use the following command

>> dataAll=importdata('CornerTest.matlab');

"dataAll.data(:,1:2)" will contain the time and volt for DIG_ot for "tt" simulation and 27 Celsius degree. The designer can now do the analysis for the target signal and corner.

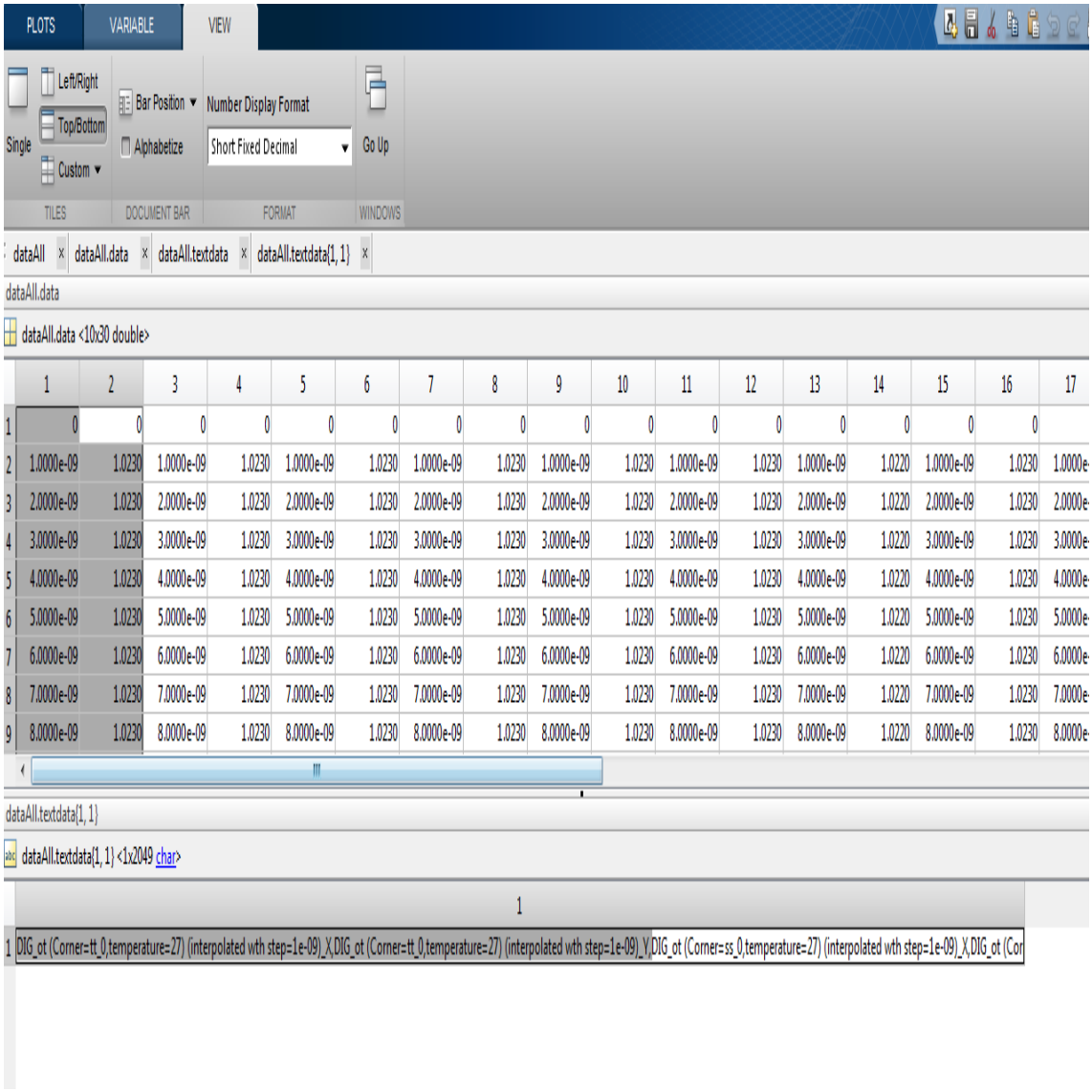


Figure A.25: Exported Matlab file view

ملخص الرسالة

محولات البيانات بين التمثيل التناظرى و التمثيل الرقمى تعّرف الحدود الفاصلة بين المكونات الرقمية و التناظرية فى اى آله. تحسين محولات بيانات قوية يعنى ان مزيدا من المكونات التناظرية يمكن ان تتحول لشكل رقمى. المكونات الرقمية اسهل فى)التصميم و التعديل. و هى أيضا تقدم قدرة فعالة فى مقاومة الضوضاء (الألكترونية .مع استهلاك اقل للطاقة و مساحة التصنيع مستفيدة من اى تكنولوجيا تصنيع جديدة
عديد من تصميمات محولات البيانات من التمثيل التناظرى للتمثيل الرقمى قد تم إقتراحها لخدمة العديد من التطبيقات. إختيار التصميم الأمثل يكون بناء على العديد من العوامل مثل معدل أخد العينات, مقدار إستهلاك الطاقة الكهربية و مساحة التصنيع و الدقة و القدرة على مقاومة تأثير تغيرات عملية التصنيع و الجهد الكهربى المستخدم و و درجة الحرارة. , سيجما-دلتا ومحول البيانات اللحظى المحول الأنبوبو المسجل .التقاربى المتتالى
ومحولات البيانات ذات الطابع الزمنى هى تصنيف خاص من محولات البيانات التى القيمةيفضل فيها ان يتم التحويل بإستخدام مكونات رقمية. التحويل بين القيمة التناظرية يتم التحويل منالقيمة الرقمية المناظرة يتم على مرحلتين. فى المرحلة الأولى يتم تحويل يتم فى المرحلةالتناظرية من تغير فى قيمة الجهد إلى تغير زمنى. فى المرحلة الثانية المرحلة الأولى فإنالتغير الزمنى إلى القيمة الرقمية الثنائية المناظرة. اغلبية التحويل بإستخدام مكونات رقميةالثانية. و حيث أن المطلوب إنجازة قد تم تبسيطة عن طريق للتحويل من الإشارات التناظرية الى الإشارات.المرحلة الثانية من المتوقع أن يتم تمثيلها هى نسخة معدلة من خوارزمية التقريب المتتابع و التىفى هذا العمل نقدم خوارزمية جديدة ثنائية فرعية من القيمة القصوى للمدخلات لايجاد القيمة الرقميةالرقمية. هذه الخوارزمية بشكل متكرر. الخوارزمية المقترحة تنقل الاشتراطات اللازمة بين القيميستخدم فيها نسب المحسوبة من التمثيل التناظرى الى التمثيل الرقمى. فى النسخ المطوية منالمناظرة الدوائر المعتمدة على خوارزمية التقريب المتتابع (التى يتم إيجاد القيم الثنائية فيهاالثنائية بشكل تكرارى) قد لا نحتاج إلى التحويل من القيم الرقمية الى القيم التناظرية بعد الآن و هو ما يحقق تقدم واعد من حيث مساحة التصنيع و الطاقة الكهربية المستهلكة. مقدم فىايضا إثبات رياضى كامل للخوارزمية الجديدة كما تم تطوير تصميم دائرة كهربية المتواجدةجديدة للاستفادة من فوائد هذة الخوارزمية. طبقاً للنتائج المرفقة فإن التحسين استهلاك الطاقة و التوفير فى مساحة التصنيع ينافس أحدث التصميماتالأن

مهندس: كريم اسامة رجب
تاريخ الميلاد: ١٩٨٩/٩/٨
الجنسية: مصري
تاريخ التسجيل: ٢٠١٢/١٠/٠١
تاريخ المنح: yyyy/mm/dd
الدرجة: الماجستير
القسم: هندسة الإلكترونيات والاتصالات الكهربية

المشرفون: د.أحمد عميرة

د. حسن مصطفى

الممتحنون:

د.أحمد عميرة	(المشرف الرئيسي)
محمد رياض الغنيمى	(الممتحن الداخلي)
----------	(الممتحن الخارجي)

عنوان الرسالة:

خوارزمية و تصميم دائرة بإستخدام تقنية التقريب المتتابع
بالتقطيع المتواصل لمحولات البيانات ذات الطابع الزمنى

الكلمات الدالة:
محول الوقت إلى رقمى ، محول تناظرى رقمى، نظم الراديو المعرفة برمجيا ، محول الوقت لرمز

ملخص الرسالة:

فى هذا العمل نقدم خوارزمية جديدة للتحويل من الإشارات التناظرية الى الإشارات الرقمية. هذه الخوارزمية هى نسخة معدلة من خوارزمية التقريب المتتابع و التى يستخدم فيها نسب ثنائية فرعية من القيمة القصوى للمدخلات لايجاد القيمة الرقمية المناظرة بشكل متكرر. الخوارزمية المقترحة تنقل الاشتراطات اللازمة بين القيم الثنائية المحسوبة من التمثيل التناظرى الى التمثيل الرقمى. فى النسخ المطوية من الدوائر المعتمدة على خوارزمية التقريب المتتابع (التى يتم إيجاد القيم الثنائية فيها بشكل تكرارى) قد لا نحتاج إلى التحويل من القيم الرقمية الى القيم التناظرية بعد الآن و هو ما يحقق تقدم واعد من حيث مساحة التصنيع و الطاقة الكهربية المستهلكة. مقدم ايضا إثبات رياضى كامل للخوارزمية الجديدة كما تم تطوير تصميم دائرة كهربية جديدة للاستفادة من فوائد هذة الخوارزمية. طبقاً للنتائج المرفقة فإن التحسين فى استهلاك الطاقة و التوفير فى مساحة التصنيع ينافس أحدث التصميمات المتواجدة الآن.

خوارزمية و تصميم دائرة بإستخدام تقنية التقريب المتتابع بالتقطيع المتواصل لمحولات البيانات ذات الطابع الزمنى

إعداد

كريم أسامة رجب

رسالة مقدمة إلي
كلية الهندسة – جامعة القاهرة
كجزء من متطلبات الحصول علي درجة
الماجستير
في
هندسة الإلكترونيات و الاتصالات الكهربية

يعتمد من لجنة الممتحنين:

د. أحمد عميرة – المشرف الرئيسي

– الممتحن الداخلي

عين شمس – الممتحن الخارجي

كلية الهندسة – جامعة

كلية الهندسة – جامعة القاهرة
الجيزة – جمهورية مصر العربية
٢٠١٧

خوارزمية و تصميم دائرة بإستخدام تقنية التقريب المتتابع بالتقطيع المتواصل لمحولات البيانات ذات الطابع الزمنى

إعداد

كريم أسامة رجب

رسالة مقدمة إلي
كلية الهندسة – جامعة القاهرة
كجزء من متطلبات الحصول علي درجة
الماجستير
في
هندسة الإلكترونيات و الاتصالات الكهربية

تحت إشراف

د. أحمد عميرة
الأستاذ المتفرغ
قسم هندسة الإلكترونيات والاتصالات الكهربية
كلية الهندسة – جامعة القاهرة

د. حسن مصطفى حسن مصطفى
المدرس
قسم هندسة الإلكترونيات والاتصالات الكهربية
كلية الهندسة – جامعة القاهرة

كلية الهندسة – جامعة القاهرة
الجيزة – جمهورية مصر العربية
٢٠١٧

خوارزمية و تصميم دائرة بإستخدام تقنية التقريب المتتابع بالتقطيع المتواصل لمحولات البيانات ذات الطابع الزمنى

إعداد

كريم أسامة رجب

رسالة مقدمة إلى
كلية الهندسة - جامعة القاهرة
كجزء من متطلبات الحصول على درجة
الماجستير
فى
الإلكترونات و الإتصالات الكهربية

كلية الهندسة – جامعة القاهرة
الجيزة – جمهورية مصر العربية
٢٠١٧