ROBERTO SARTORI

Design, analysis and optimization of a dynamically reconfigurable regenerative comparator for ultra-low power 6-bit TC-ADCs in 90nm CMOS technology

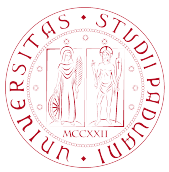
Design, analysis and optimization of a dynamically reconfigurable regenerative comparator for ultra-low power 6-bit TC-ADCs in 90nm CMOS technology

roberto sartori

Supervisor Prof.ssa Maria Elena Valcher

Assistant Supervisor Dr.D. Juan A. Montiel-Nelson

October 2013



UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Escuela de Ingeniería de
Telecomunicación y Electrónica
Facultad de Ingeniería
Universidad de Las Palmas de
Gran Canaria

Dipartimento di Ingegneria
dell'Informazione
Facoltà di Ingegneria
Università degli Studi di Padova

"While the pessimists complain about the wind,
and the optimists expect the wind to change,
we adjust the sails." — William Arthur Ward

To my family.

ABSTRACT

Analog-to-Digital Converters (ADCs) have always been a topic of intense research in electronics since the advances in digital processing and storage technologies enabled for the pre-processing, post-processing and storage of analog signals as large streams of digital values. Mixed signal integrated circuits, including data acquisition and conversion, experienced a real revolution in terms of the number of emerging applications in electronics and telecommunications.

Powerful processors appeared for mixed signal processing, allowing for longer word-length, higher operation frequencies and larger memory sizes. Research efforts were initially focused on increasing the sampling rate and the resolution of the analog-to-digital and digital-to-analog converters to meet real-time multimedia processing requirements. Several acquisition methodologies and architectures were presented to support a wide range of applications oriented to processing huge volumes of data [1].

However, nowadays there is a high number of emerging applications where it is still necessary to collect and process analog data, but they are subject to strong energy consumption restrictions. Mobile phones containing gyroscopes and accelerometers among other sensor assisted Global Positioning System (GPS) navigation applications, sensor networks supported by small batteries, or medical aids using remote sensing battery-less are typical ultra-low power applications requiring Analog-to-Digital Converter (ADC) subsystems. The conversion rate and resolution specifications are less restrictive than the power consumption restrictions in these applications. Typically, ultra-low power requirements must be met, but just some few samples or kilo-samples per second are needed for 4-8 bit resolutions. Therefore, among all ADC architectures in literature, the Successive Approximation Register (SAR) and $\Sigma\Delta$ converters are the more suitable architectures [2] for this purpose.

It is well known that SAR architectures are preferable in terms of area and power consumption in comparison with the other topologies [3] at medium and low conversion rates. Furthermore, due to the fact that high performance technologies are mainly focused on reducing the delay while maintaining low power consumption in the digital circuitry, the use of recent technlogies in ADC design is more effective as greater is the digital section in comparison with its analog one [4]. In this sense, several approaches have been presented to optimize the digital controller reducing the power and increasing the conversion speed on SAR ADC architectures by using comparator-based binary search [5, 6] and asynchronous controllers [7, 8, 9]. In a Comparator-

based Asynchronous Binary Search (CABS), the need for digital-to-analog conversion and digital controllers is completely removed by using a binary tree of comparators with built-in thresholds which are triggered to run the successive approximation algorithm [5]. But the circuit complexity of this approach follows the same exponential growth than flash ADCs.

Recently, a novel ADC architecture named Threshold-Configuring ADC (TC-ADC) was introduced [10]. TC-ADC is based on a single comparator and a programable array of transistors that allow to implement the ADC and Digital-to-Analog Converter (DAC) functionalities [11]. The TC-ADC authors demonstrate the usefulness of this new architecture when low power consumption is required at medium/low conversion rates.

In this work the threshold configurable regenerative comparator on which TC-ADCs are based is optimized to further reduce the power consumption for use in battery-less biomedical sensor applications. Moreover, the effect of device mismatches on the offset, gain and linearity errors of the ADC is analyzed by means of Monte Carlo simulations. This optimized comparator reduces the power consumption from $13\mu W$ to $3\mu W$ in the more power hungry section of the TC-ADC (65% of the overall TC-ADC power is dissipated in the regenerative comparator), while maintaining the same full scale range. The optimized comparator achieves also better performance (about 50% improvement) in terms of offset, gain and non-linearity errors when matched devices are given. In the presence of device mismatch just the gain error shows a significative improvement, although non-linearity error analysis indicates higher simmetry and predictability which can be further exploited to reduce the complexity of non-linearity cancellation circuits required for higher resolution applications.

RESUMEN

En este trabajo se optimiza el comparador regenerativo configurable de umbral en lo que el Threshold-Configuring SAR-ADC se basa para reducir aún más el consumo de energía, en aplicaciones de sensores biomédicos sin batería [1]. Por otra parte, el efecto de la dispersión de proceso de los dispositivos en características del conversor como el offset, la ganancia y los errores de linealidad del ADC se han de analizar de forma cualitativa y cuantitativa por medio de simulaciones de Monte Carlo. Resultados previos muestran que estos comparadores

optimizados permiten reducir el consumo de energía de $13\mu W$ hasta $3\mu W$ manteniendo constante el mismo fondo de escala.

En presencia de la dispersión de proceso "device mismatch", los mismos resultados previos muestran que sólo el error de ganancia muestra una mejora significativa, aunque el análisis de error de no linealidad indica una mayor simetría y la previsibilidad. Estas dos características pueden aprovecharse más para reducir la complejidad de los circuitos de cancelación de no linealidad requerida para aquellas aplicaciones de mayor resolución [12].

Por todo lo anterior, es objetivo de este trabajo obtener las relaciones cualitativas y cuantitativas entre dimensiones de dispositivos, potencia consumida, fondo de escala, errores de linealidad, offset y ganancia del conversor ADC basado en un comparador de umbral reconfigurable y regenerativo como el referenciado en [13]. Estas relaciones permiten dimensionar el comparador para una aplicacion de ultra bajo consumo de potencia. El proceso tecnologico es de UMC, Complementary Metal-Oxide-Semiconductor (CMOS) 90nm "Standard Process 1.0V CMOS 1P9M".

COMPENDIO

Il presente studio mira ad ottimizzare il comparatore di soglia di tensione regenerativo riconfigurabile su cui si basa il Threshold-Configuring SAR ADC, al fine di ridurre ulteriormente il consumo di energia in applicazioni quali sensori biomedici senza batteria. Vengono inoltre analizzati in forma qualitativa e quantitativa gli effetti della dispersione di processo dei dispositivi che costituiscono il convertitore, l'offset, il guadagno e gli errori di linearità del ADC, attraverso simulazioni Monte Carlo. Risultati precedenti mostrano come questi comparatori ottimizzati permettano di ridurre il consumo di energia da $13\mu W$ a $3\mu W$ mantenendo costante il fondo scala.

L'obiettivo finale è dunque quello di ottenere le relazioni qualitative e quantitative tra le dimensioni dei dispositivi, la potenza dissipata, il fondo di scala, gli errori di linearità, l'offset e il guadagno del ADC basato sul comparatore di soglia di tensione riconfigurabile e rigenerativo presentato in [13]. Tali relazioni permettono di dimensionare il comparatore per applicazioni ad ultra basso consumo di potenza. Il processo tecnologico con cui vengono sviluppate le simulazioni è UMC, CMOS 90nm "Standard Process 1.0V CMOS 1P9M".

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

ACRONYMS

ADC Analog-to-Digital Converter

AVG AVeRage

BS Binary Search

CABS Comparator-based Asynchronous Binary Search

CC Comparator-based Controller

CMOS Complementary Metal-Oxide-Semiconductor

CR Charge-Redistribution

CS Charge-Sharing

CtrlTC Threshold Configuration control

DAC Digital-to-Analog Converter

DNL Differential NonLinearity

DoE Design-of-Experiments

ENOB Effective Number Of Bits

EOC End Of the Conversion

ERP Effective Radiated Power

FOM Figure Of Merit

FS Full-Scale

FTL Fast FeedThrough Logic

GaAs Gallium Arsenide

GPS Global Positioning System

IDC Incremental Data Converter

INL Integral NonLinearity

LSB Least Significant Bit

MOM Metal-Oxide-Metal

MOS Metal-Oxide-Semiconductor

MOSFET Metal-Oxide-Semiconductor Field-Effect Transistor

MSB Most Significant Bit

RFID Radio-Frequency IDentification

RMS Root Mean Square

S/H Sample and Hold

SA Successive Approximation

SAR Successive Approximation Register

SC Switched Capacitor

SNDR Signal-to-Noise and Distortion Ratio

SNR Signal-to-Noise Ratio

T/H Track and Hold

TC Threshold-Configuring

TC-ADC Threshold-Configuring ADC

VLSI Very-Large-Scale-Integrated

Part I

ULTRA-LOW POWER ADC ARCHITECTURES

1 INTRODUCTION

Analog-to-digital and digital-to-analog converters provide the link between the analog world of systems where transducers are involved and the digital world of signal processing, computing, and other digital data collection or data processing systems. Numerous types of converters have been designed that use the best technology available at the time a design is made. High-performance sub-micron CMOS technologies result in high-resolution or high-speed ADCs and DACs that can be applied to digital audio, digital video, instrumentation and signal processing systems.

In many recent applications, such as in battery-operating medical devices or in habitat monitoring sensor networks, special data converters which can operate on battery or even harvested power are needed. The available power in these devices is very limited, often only a few tens of microwatts. In these situations data converters are used to convert only low-frequency signals with just low-to-medium accuracy. Low power consumption, however, is critical. Examples of such applications include battery-powered biomedical sensors for electrocardiogram or electroencephalogram signals, hearing aids and sensor networks for industrial or environmental applications. These applications stimulated novel algorithms, novel architectures and special circuit design strategies for both DACs and ADCs.

1.1 BASIC A/D CONVERTER FUNCTION

In a digital system the amplitude is quantized into discrete steps and at the same time the signal is sampled at discrete time intervals. A block diagram of an A/D converter is shown in Figure 1.1. A Sample

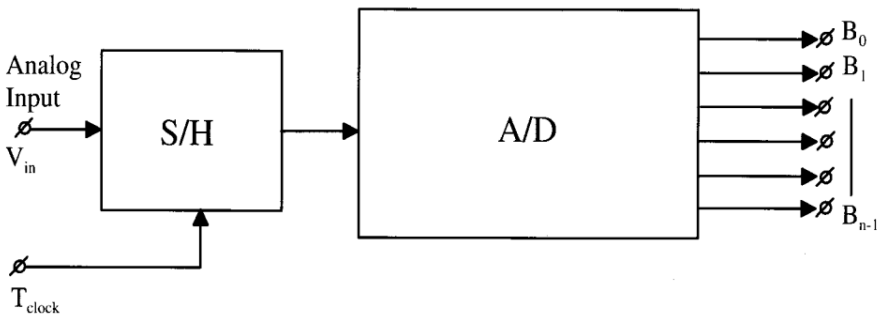


Figure 1.1: Block diagram of an ADC

and Hold (S/H) amplifier is added to sample the input signal and hold the signal information at the sampled value during the time in which

the conversion into a digital number is performed. The analog input value is converted into a digital value using the following equation

$$\frac{V_a}{R_{ref}} = D_{out} + q_e = \sum_{m=0}^{n-1} B_m 2_m + q_e \tag{1.1}$$

In this equation D_{out} represents the digitized value of the analog input signal V_a and q_e represents the quantization error. The quantization error represents the difference between the analog input signal V_a divided by R_{ref} and the quantized digital signal D_{out} when a finite number of quantization levels n is used [3].

1.1.1 *Conversion systems*

When an ADC or a DAC are applied in a complete system using digital signal processing, then extra components must be added. In Figure 1.2 an A/D converter system is shown. The filters at the input of

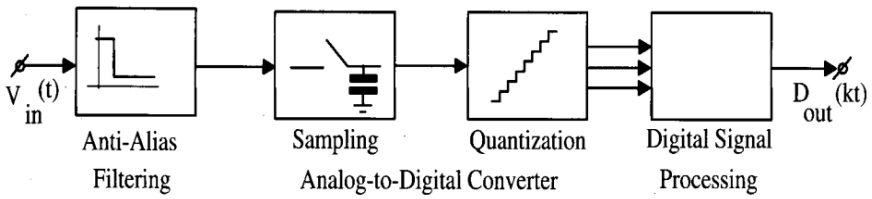


Figure 1.2: A/D converter system

the ADC limit the input signal band. In many cases this signal band is a low-pass band, but bandpass applications are possible too. In the bandpass case the high frequency band is converted into a low-pass frequency band and then converted into a digital value. Such an operation is called *sub-sampling*. When the S/H amplifier shows good high frequency performance, then the sub-sampling operation is performed with low distortion. Bandpass signals can be converted into low-pass signals.

The performance of the filters in A/D and D/A converter systems has a strong influence on the total performance of such systems.

1.2 SPECIFICATIONS OF CONVERTERS

To obtain insights into the design criteria for converters it is important to arrive at an unanimous definition of specifications. These specifications must include the application of converters into conversion systems. Dynamic specifications of converters are needed to obtain insights in the applicability of a certain converter in a digital signal processing system: for example, digital audio or digital video. In a conversion system the complete conversion from analog into digital or digital into analog information is performed. Such a system includes input or output amplification and anti-alias filtering.

A converter basically consists of an amplitude quantizer followed by a sampler (Figure 1.3). In an ideal converter the sequence of quan-

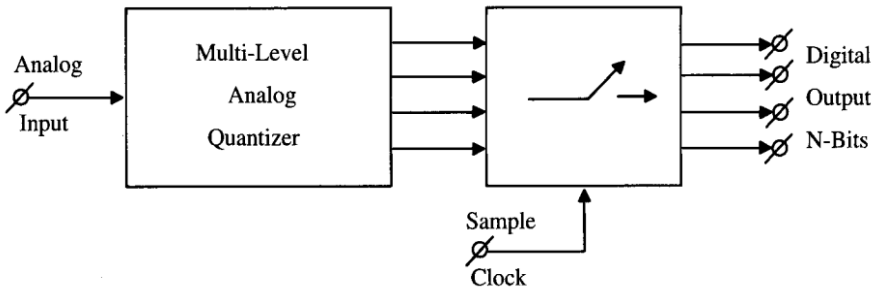


Figure 1.3: Ideal converter

tizer and sampler can be changed without having any influence on the performance or the operation of the system. In practical converters, electronic elements used to construct the converter show finite matching. Noise in active and passive elements reduces the maximum dynamic range of a system. Especially with low supply voltages the limiting effects of noise will occur in wide-band or high-resolution converters. The non-idealities introduced by component mismatch introduce errors in the operation of the converter. In general a designed converter meets a linearity specification set at about $\pm\frac{1}{2}$ Least Significant Bit (LSB). Such a specification will introduce errors that are far more significant than the errors introduced by quantization. It is therefore important to refer these errors to the quantization error. The non-ideality then results in a decrease of performance in Effective Number Of Bits (ENOB)s compared to the ideal converter [3].

1.2.1 *Absolute accuracy*

The *absolute accuracy* of a converter is the actual full-scale input or output (ADC or DAC) signal (voltage, current, or charge) referred to the absolute standard of *the National Bureau of Standards*. This absolute accuracy is mostly related to the reference source used in the converter. In integrated circuits this reference consists of an integrable source which in modern systems is based on the band-gap voltage of silicon. This reference source should have low-noise with respect to the resolution of the converter. Temperature coefficients in the ideal case should be so small that the accuracy of the reference over the specified temperature range stays within the resolution of the converter ($\pm\frac{1}{2}$LSB over the full temperature range).

1.2.2 *Relative accuracy*

The *relative accuracy* is the deviation of the output signal or output code of a converter from a straight line drawn through zero and full

scale. Output signals or output codes must be corrected from a possible zero offset. This relative accuracy is called Integral NonLinearity (INL).

The $\pm\frac{1}{2}$LSB INL definition (the boundaries for which the nonlinearity deviates not more than $\pm\frac{1}{2}$LSB from a straight line through zero and full scale) implies a *monotonic* behavior of the converter.

In an ADC *monotonicity* means that no missing codes can occur. It must be noted at this point that converters can be designed which are guaranteed monotonic but do not have the half LSB linearity specification. These converters are based on non-binary weighting of the bit currents.

In an ADC the INL definition including the quantization levels equal to

$$INL \leq \frac{1}{2}LSB \tag{1.2}$$

is valid. In case the Most Significant Bit (MSB) value is 2LSB values larger than the sum of all the smaller bits, then a missing code appears in the converter. To guarantee monotonicity of the converter, the sum of the errors must never exceed $\pm\frac{1}{2}$LSB value.

Note that the monotonicity specification does not automatically ensure that a converter has an INL error less than or equal to $\pm\frac{1}{2}$LSB, while the nonlinearity specification of less than or equal to $\pm\frac{1}{2}$LSB is sufficient to prove monotonicity of a binary-weighted converter.

1.2.3 *Differential nonlinearity*

The Differential NonLinearity (DNL) error describes the difference between two adjacent analog signal values compared to the step size (LSB weight) of a converter generated by transitions between adjacent pairs of digital code numbers over the full range of the converter.

The differential nonlinearity is zero if every transition to its neighbors equals 1LSB. In a monotonic binary-weighted converter an increase of the digital code value by 1LSB can result in an increase in the analog signal between 0 and 2LSBs. The maximum differential nonlinearity in this case is $\pm$1LSB.

Writing down the DNL for an ADC in a formula gives:

$$DNL = A_{input}(Q_{m+1}) - A_{input}(Q_m) - 1LSB \tag{1.3}$$

where Q_{m+1} and Q_m are two adjacent quantization levels. $A_{input}(Q_m)$ is the analog input voltage corresponding to the quantization level Q_m.

In Figure 1.4 the transfer curve of a 4-bit ADC is shown. The drawn line shows the ideal transfer characteristic, while a dashed line indicates the measured transfer curve of a practical converter.

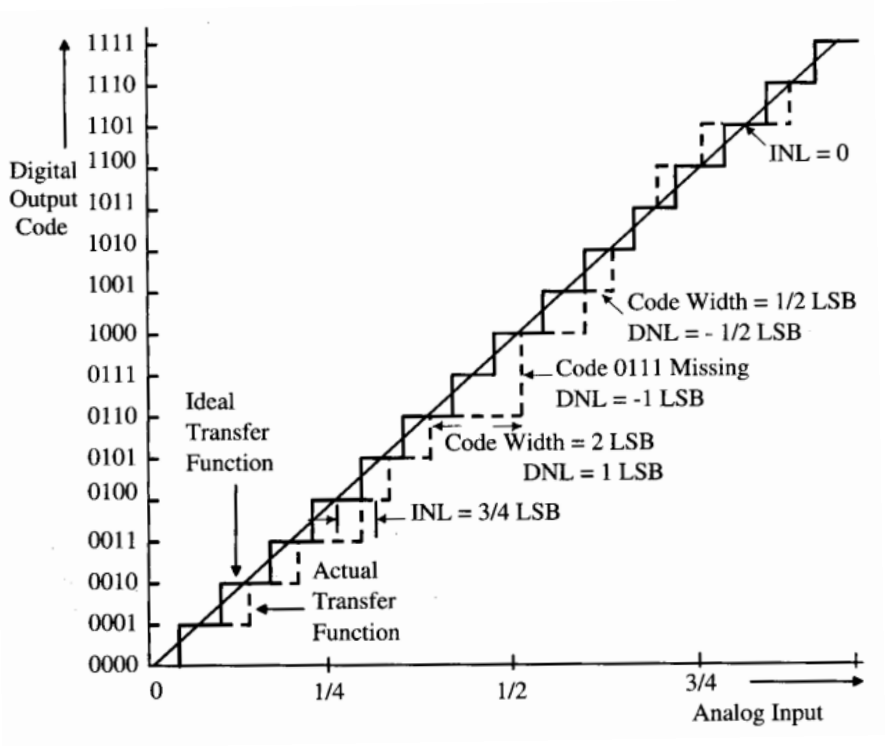


Figure 1.4: Transfer curve of a 4-bit ADC

1.2.4 *Offset*

Input amplifiers, output amplifiers, and comparators in practical circuits inherently have a built-in offset voltage and offset current. This offset is caused by the finite matching of components. The offset results in a non-zero input or output voltage, current or digital code although a zero signal is applied to the converter.

1.2.5 *Signal-to-Noise Ratio*

The Signal-to-Noise Ratio (SNR) depends on the resolution of the converter and automatically includes specifications of linearity, distortion, sampling time uncertainty, glitches, noise, and settling time. Over half the sampling frequency, the SNR must be specified and should ideally follow the theoretical formula

$$S/N_{max} = 6.02n + 1.76dB \tag{1.4}$$

The SNR is calculated for a sine wave input with the maximum amplitude allowed by the system. The ratio between the frequency of the sine wave and the sampling frequency should be irrational. In case input signals with a smaller amplitude are applied, then the SNR decreases in accordance to the input signal decrease.

1.2.6 *Effective Number of Bits*

To get a comparison method for converters, the ENOBs is measured under Nyquist conditions (see Section A.2). The dynamic range of the converter under comparison includes quantization errors, clock jitter errors, distortion errors and circuit noise. Then the ENOBs is defined as

$$ENOB = \frac{(SNDR_{measured})_{dB} - 1.76}{6.02}\, dB \tag{1.5}$$

with the Signal-to-Noise and Distortion Ratio (SNDR), measurement of the purity of a signal, defined [14] as

$$SNDR = \frac{P_{signal}}{P_{quantizationError} + P_{randomNoise} + P_{distortion}} \tag{1.6}$$

where P is the average power of the signal, quantization error, random noise and distortion components.

Using this definition of SNDR it is very easy to compare ADCs or DACs with the same number of bits, but, due to different circuit designs, having different performance.

1.2.7 *Figure of Merit*

To compare different architectures and performances of a converter the Figure Of Merit (FOM) has been defined. This index compares converters with respect to power consumption, ENOBs and maximum input signal frequency. The FOM is defined as

$$FOM = \frac{Power}{2f_{in}2^{ENOB}} \tag{1.7}$$

In this equation $2f_{in}$ can be defined as the Nyquist frequency (see Section A.3), however, in case the resolution is determined by the ENOB the input frequency can be different from the sampling frequency and in such a case will be smaller than the Nyquist frequency. With improved technology and advanced converter architectures the FOM drops about a factor every 10 years.

2

A/D CONVERTERS

According to the necessity of *high-speed* or *high-accuracy* ADCs, some converter architectures are more suitables than others.

Flash converter

The best-known architecture for high-speed ADC is the *flash converter* structure. In this structure an array of comparators compares the input voltage with a set of increasing reference voltages (Figure 2.1). The comparator outputs represent the input signal in a digital (ther-

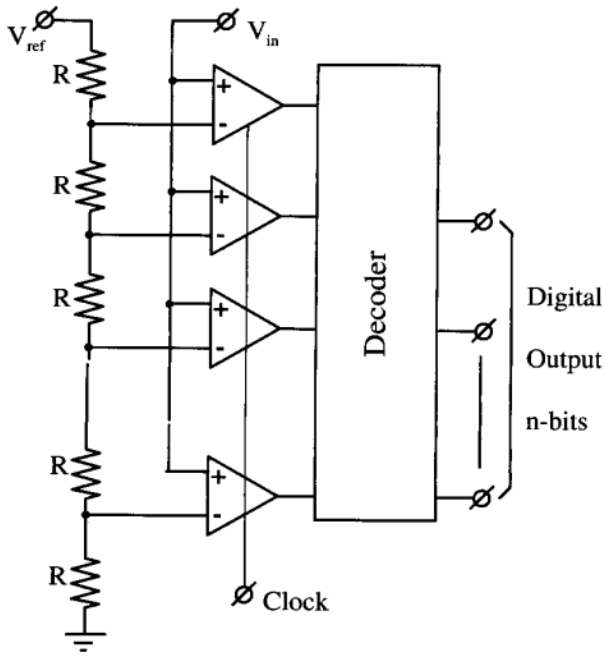


Figure 2.1: Full-flash ADC architecture

mometer) code which can be easily converted into a Gray or binary weighted output code. The flash architecture shows a good speed performance and it can easily be implemented in an integrated circuit as a repetition of simple comparator blocks and a (ROM) decoder structure. However, this architecture requires $2^N - 1$ comparators to achive an N-bit resolution. The parallel structure makes it difficult to obtain a high-resolution while maintaining at the same time a large bandwidth, a low power consumption, and a small die size. Interpolation between reference levels reduces the number of reference taps and input amplifiers resulting in a lower power consumption. The influence of offset voltages in the input amplifiers can be reduced by

using averaging between active amplifier stages. At the same time, SNR is improved without using more power.

Sub ranging converter

An alternative to the full-flash architecture is the multi-step A/D conversion or *sub ranging* principle. In high-speed converters the two-step architecture is the most popular because of the ease of implementation. However, in the sub ranging system no gain stage between the first converter stage and the second converter stage is used. Matching problems between gain stages and reference voltage of the second or subrange converter stage are avoided in this way. In Figure 2.2 a ba-

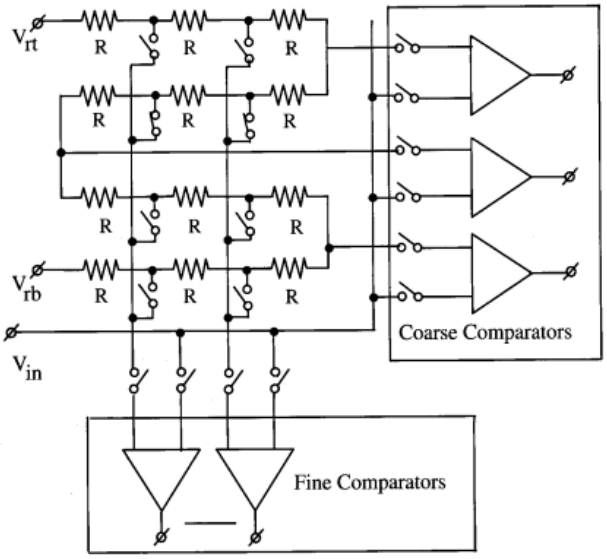


Figure 2.2: Sub ranging converter architecture

sic subrange system is shown. An ideal 8-bit sub ranging converter consists of a 4-bit coarse quantizer using 16 coarse comparators and a 4-bit fine quantizer using 15 fine comparators. Furthermore a reference ladder with $2^N - 1$ ladder taps is used. The coarse comparators are connected to 16 coarse ladder taps. These ladder taps have 15 fine taps in between of every coarse tap. When the coarse information is obtained, then the fine ladder taps in between of the coarse signal are addressed and the fine conversion can take place. It must be clear that at the input a S/H amplifier is needed to sample the analog input information. During the hold mode the coarse and fine conversion takes place. In general the coarse comparators have less gain than the fine comparators. Furthermore in a practical system an over and an under range is required for the fine converter to correct for small errors encountered in the coarse quantization. Linearity of this system is determined by the input S/H amplifier, the linearity of the reference ladder and the offset voltage of the fine comparators.

After the coarse quantization is performed, the digital signal is applied to a DAC to reconstruct the analog signal. This reconstructed signal is subtracted from the analog input signal which is held by the S/H amplifier. After subtraction has taken place the residue signal can be amplified and is then applied to the fine quantizer which performs the conversion into a digital value. The coarse plus fine output code with, in many cases, an error correction operation results in the final digital output word. A good balance between circuit complexity, power consumption, and die size is obtained in this type of converter. The final dynamic performance, however, depends substantially on the quality and dynamic performance of the S/H amplifier.

In Metal-Oxide-Semiconductor (MOS) technology circuits can be operated in a continuous-time mode or in a discrete-time mode. Most architectures in MOS use a discrete-time mode of operation. In such a solution the S/H operation is combined with the system function. Tipically the discrete-time operation includes an automatic offset cancelation technique in the comparator stages. In a full-flash system, $2^N - 1$ small S/H amplifier-comparators are therefore used to perform the conversion function. Because of the small value of the hold capacitor, offsets induced by switching transients and channel charges of the switching devices limit the resolution of the total system.

Pipeline converter

Pipeline converter architectures are very popular in CMOS technology. This architecture consists of a cascade of simple modular converter blocks performing between 1 and 5 bit conversions each. Each block consists of an ADC, a DAC for the reconstruction of the analog signal, a subtracter to determine the signal residue after the quantization and a gain stage. A S/H function is part of the converter block. Analog data is delayed by the S/H amplifier stage during the conversion resulting in an output code latency equal to the number of cascaded stages.

A high resolution at a high sampling frequency is possible using the pipeline architecture. Sharing of amplifiers in a pipeline converter is possible. This reduces power consumption and reduces die size.

An analysis of Low-Power Pipeline ADCs is presented in Section 2.1.1.

Folding converter

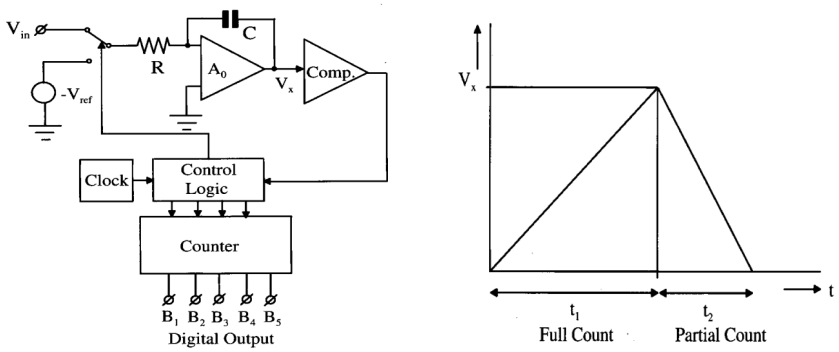
To overcome some problems of the S/H amplifier, design alternatives have been worked out that have the advantage of the digital sampling used in the full-flash converter and the die size of the two-step system but do not require a S/H amplifier. This architecture is called a *folding* architecture, which is capable of achieving a large analog bandwidth and high resolution without incurring in the power and area penalties

associated with the flash architectures (details can be found in [15, 16]).

High-resolution monolithic ADCs are subject to growing interest due to the rapidly expanding market for digital signal processing systems. Monolithic converters with such a high linearity are difficult to design and require special circuit configurations. When a low conversion speed is needed, integrating types of converters can be used. In integrating types of high-resolution ADCs basically the analog input signal is converted into a time which is proportional to the input signal. Time is measured using a counter with an accurate clock. These systems are relatively slow because of the counting operation in the time-to-number conversion cycle. A speed improvement is obtained by using a coarse and fine conversion cycle in the time-to-number counting operation. A well-known ADC based on this system is the dual slope converter.

Dual slope converter

This system consists of an input switch, an integrator with a comparator, a clock generator with control logic, and a counter (Figure 2.3a) . The operation of the system is as follows. Starting from a reseted



(a) Dual slope ADC system (b) Dual slope ADC timing

Figure 2.3: Dual slope ADC architecture

integrator, the input signal V_{in} is integrated during a time t_1 which corresponds to a full count of the counter. Then the input is switched to the reference voltage V_R having the opposite sign compared to the input signal. The integrator is now discharged. During the discharge time pulses are counted. Counting stops when the comparator detects zero. As a result, the counts in the counter represent the digital value of the input signal. The timing of the operation is shown in Figure 2.3b. A simple calculation shows

$$V_{in} = \frac{t_2}{t_1} V_R \tag{2.1}$$

Here t_2 is the time during which the integrator is discharged from the integrated input signal to zero. As it is shown in Equation 2.1 the clock is not critical; only the ratio between the charge and discharge times is important. A disadvantage of this system is the low conversion speed if a high resolution is required.

Successive approximation converter

In fast and highly accurate ADCs, the *successive approximation* method is commonly used. Accuracy and linearity in this system are determined by the DAC, while the conversion speed depends on the comparator response time and the settling time of the DAC. In a successive approximation system the analog input signal is approximated by the step-by-step built-up analog output voltage of the DAC, starting with the most significant bit. To obtain the high accuracy for the DAC needed to construct a 14- to 16-bit ADC, *Dynamic Element Matching* is used.

Due to the construction of the bit switches, which in the high-accuracy part consist of a diode-transistor configuration, the output voltage swing at the D/A current output must be small. This voltage swing, called output voltage compliance, reduces the bit switching accuracy of the DAC. To avoid problems in this system, a special comparator operation is needed. In general-purpose ADCs, a general nonlinear comparator circuit with high gain around the zero-crossing level is used.

SAR converters are discussed more in details in Section 2.2.

Cyclic converter

The ease with which a hold operation can be constructed allows the ADC implementation using a *cyclic converter* algorithm. In the cyclic converter the number of components is drastically reduced and consists of two S/H amplifier circuits with an accurate times two amplifier stage and a subtracter circuit. Per conversion step the remaining signal is compared with a reference signal. If the remainder is larger than the reference signal, a subtraction of the reference signal from the remainder is performed. The error signal that is then generated is amplified by two and compared with the reference signal again. This operation is repeated until the total number of bits that can be converted is obtained [3].

2.1 LOW-POWER ADC ARCHITECTURES

It is possible to classify ADCs as *Nyquist-rate* or *oversampling* ADCs.

Nyquist-rate ADCs are memoryless, and each analog sample is individually converted into its digital equivalent. By contrast, each digital output word of an *oversampled* ADC is derived from all preceding in-

put samples. This allows the digital output to have very short output words and, hence, it allows the use of simple internal quantizers.

2.1.1 *Nyquist-rate ADCs*

Algorithm	Conversion time	Latency (delay)	Resolution (typical)	Usual implementation
Parallel (flash)	T	T	5-9 bits	R string, comparators
Pipeline	T	nT	10-14 bits	SC stages+ T/H+ opamps.
Subranging (half-flash)	$2T$	$2T$	8-12 bits	R strings, comparators
Serial (Succ.appr)	nT	nT	7-12 bits	SC charge redistribution
Counting	2^nT	2^nT	16-24 bits	SC or CT integrator

Table 2.1: Classification of Nyquist-rate ADCs
(T = clock period, N = resolution in bits)

Table 2.1 shows a classification of Nyquist-rate ADCs by their algorithms. For micropower applications, the parallel and subranging converter architectures are usually unsuitable, and the counting converters may be too slow. Hence, the analysis will focus on pipeline and serial converters.

Low-Power Pipeline ADCs

The general architecture of a *pipeline converter* is shown in Figure 2.4 . In general a pipeline converter consists of a cascade of identical stages that are separated by a S/H amplifier. This S/H amplifier is part of the sub-converter stage. Tipically the converter is preceeded by a S/H amplifier. As it can be seen from Figure 2.4 the lower part of a converter stage consists of the already mentioned S/H amplifier followed by a p-bit analog-to-digital sub converter. This ADC drives directly a p-bit DAC to reconstruct the quantized analog signal. This quantized analog signal is subtracted from the sampled analog input signal of the stage. After subtraction of the quantized signal from the analog input signal this residue is amplified by the gain stage and then applied to the following sub-converter stage. By pipelining in the converter an optimization can be obtained between maximum sampling clock and the speed of the circuits used. In the first stage the maximum accuracy is required. This accuracy depends on the resolution the converter is designed for. After the first stage a reduced accuracy can be applied without influencing the overall converter ac-

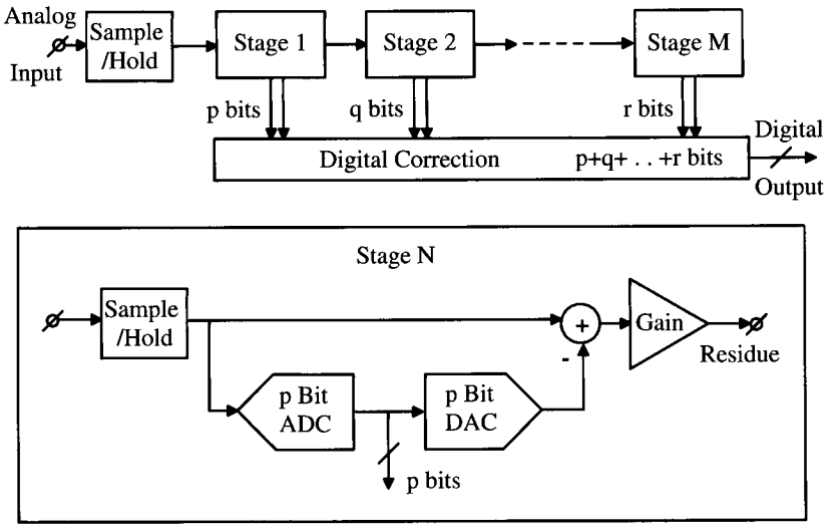


Figure 2.4: Pipeline converter architecture

curacy too much. What architecture and what resolution are used per stage depends on the overall resolution that is required and what a designer thinks he can achieve [3].

Most pipeline ADCs currently use the multiplied residue algorithm [17]. This requires the use of an internal ADC, a DAC and a residue amplifier in every stage. Although several of these functions can be combined by clever circuit design, the power dissipation of the circuit is usually too high for most micropower applications. An alternative pipeline configuration, based on the *divided reference principle* [18], is shown in Figure 2.5. Here, the analog input samples are held in

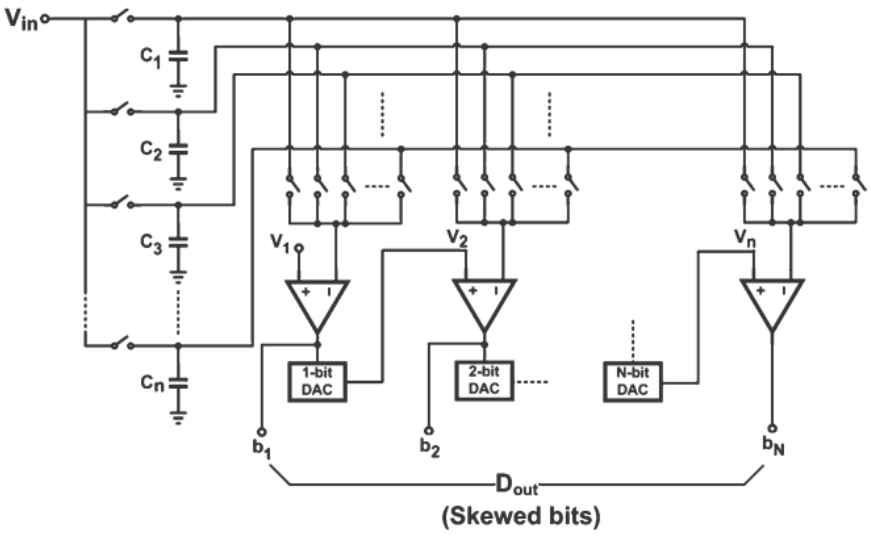


Figure 2.5: Low-power pipeline SC ADC

capacitors C_1 to C_n until all bits are determined by the combination of the DACs and comparators. The only active elements in the circuit are

the comparators, and hence, the operation requires very little power [1].

Serial ADCs

An N-bit *serial* ADC samples and holds the analog input for N clock periods and derives 1 bit of the digital output word in each period. The conversion may be based on the multiplied residue algorithm, also called *algorithmic conversion*, or on a divided reference principle (Successive Approximation (SA)). The former requires an accurate amplifier and a comparator, whereas the latter needs only Switched Capacitor (SC) circuitry [19] and a comparator. Hence, the SA ADC is more suitable for micropower converters.

Figure 2.6 shows the block diagram of a SA ADC (often, the whole ADC is called a SAR converter.) An SC realization is shown for $N = 3$ in Figure 2.7a [20]. The operation is performed in three steps.

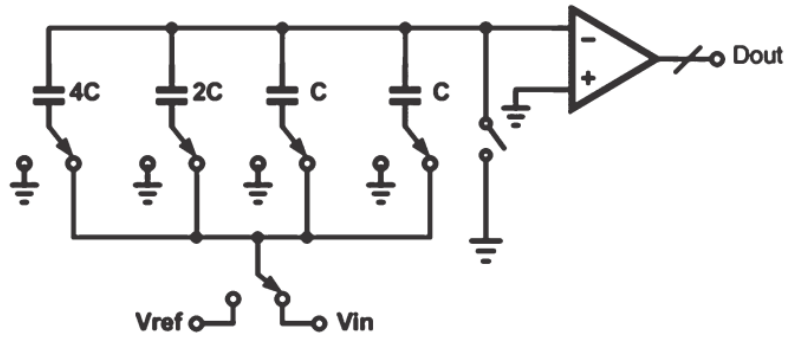
A. All capacitors are charged between V_{in} and ground.

B. The bottom plate of the largest (MSB) capacitor is switched to V_{ref} , generating a test voltage $V_{ref}/2 - V_{in}$ at the comparator input. The sign of this voltage determines the MSB.

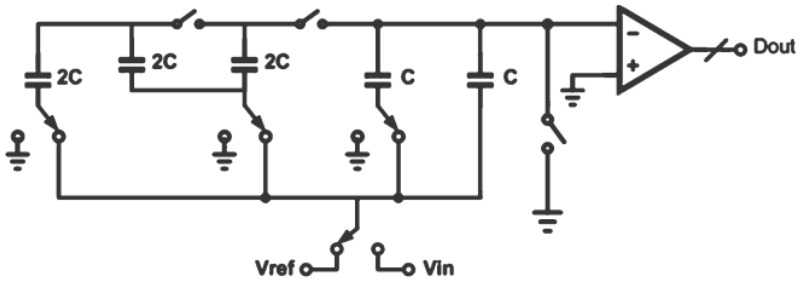
C. Depending on the MSB, a test voltage $(3V_{ref}/4 - V_{in})$ or $(V_{ref}/4 - V_{in})$ is generated at the comparator input. This determines the next bit.

Steps B. and C. are then repeated until all bits have been found. The spread between the largest and smallest capacitors is large: for N-bit resolution, their ratio is $2N$. Since the smallest capacitance has a minimum value determined by technological considerations, the largest capacitors often need to have high values. Thus, the dynamic power dissipated by charging and discharging them during the conversion can become significant. Charging a capacitor C to a voltage V requires an energy CV^2 , half of which is dissipated in the switches used. For the 3-bit ADC of Figure 2.7a, the energy needed to convert a small input voltage into the code 000 requires an energy $E = (49/8)CV^2$. Several papers discussed modifications of the basic circuit of Figure 2.7a to reduce the dynamic power dissipation. An effective method [21] is illustrated in Figure 2.7b. In this structure, the

Figure 2.6: Block diagram of an SAR ADC



(a) Circuit diagram of a SA ADC



(b) Low-power equivalent circuit

Figure 2.7: 3-bit SA ADC

conversion starts using the smallest, rather than the largest, capacitors, and additional capacitors are gradually switched into the circuit. As a result, the total power dissipation is significantly reduced; for the generation of the code 000, the energy required is only $(7/8)CV^2$. The average energy is reduced by about 75% compared with that in the original structure. A design issue arising for the structure of Figure 2.7b is that the required accuracy of the smallest capacitors C is increased, since now they directly affect the MSBs.

2.1.2 *Oversampled ADCs*

Sigma-delta ADCs

In Figure 2.8 a general form of a *sigma-delta* ($\Sigma\Delta$) ADC system is shown. The system uses a multi-bit quantizer (ADC) and a multi-bit DAC to reconstruct the analog signal. When multi-bit DACs are used to reconstruct the analog signal, then the linearity of such a converter is important. In case of high-resolution converters an accuracy problem in the D/A system is encountered. To overcome this accuracy problem a 1-bit system is used. In a 1-bit DAC the linearity is determined by the accuracy of switching between the reference signals. If a high switching accuracy can be guaranteed, then a very linear system is obtained. From the input signal the output signal of the 1-bit DAC is subtracted. The difference of these two signals is filtered by

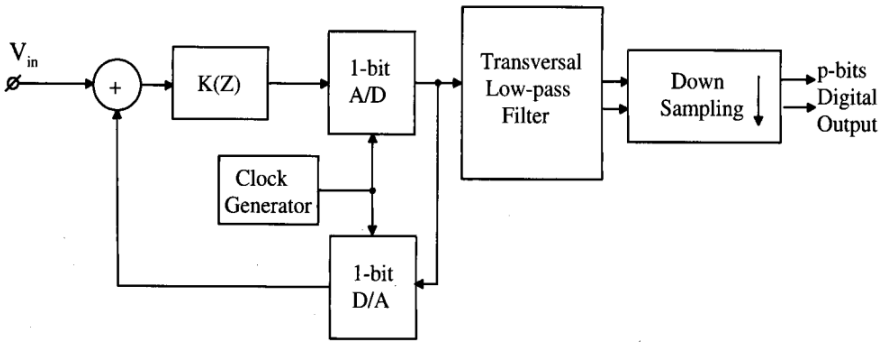


Figure 2.8: Sigma-delta ADC architecture

the loop filter, and the output signal of the loop filter is applied to the 1-bit quantizer or ADC. The clock frequency of the system is high compared to the maximum analog input frequency while the order of the loop filter determines the dynamic range of the system. The output of the 1-bit ADC is usually applied to a digital low-pass which rejects signals above the signal band of interest. Then sub-sampling or decimation is applied to obtain a multi-bit output code. The whole operation results in a binary-weighted digital output signal that can have a minimum sampling ratio equal to twice the signal bandwidth.

When the loop filter that is applied in this system consists of a continuous-time filter, then the analog signal band is filtered with the same filtering characteristic that is applied for noise-shaping. A cost-effective solution is obtained in this way. In the case of a discrete-time loop filter the anti-alias filtering must be performed before the analog signal enters the ADC. Discrete-time filters are mixing the high frequency input signals with the sampling clock, resulting in aliasing of signals which is not allowed [3].

ΣΔ ADCs are typically used for high-resolution applications and seldom in micropower systems, although by using a passive loop filter or noise coupling [22], the number of opamps needed, and hence the power requirements, may be reduced.

Incremental ADCs

A modified version of the ΣΔ ADC, that is the *incremental* ADC, is more often used in micropower applications [23]. It basically uses the same architecture as the ΣΔ ADC, but its operation is pulsed: the loop and the decimation filter are both powered for M clock periods and then reset. The final output words $d(M), d(2M), \ldots$ of the decimation filter form the output sequence. To illustrate the operation of the Incremental Data Converter (IDC), Figure 2.9 shows the analog portion of a third-order converter. Assuming that the loop is stable and scaled so that all integrator outputs satisfy

$$|v_i(n)| < V_{ref} \ (i = 1, 2, 3), \tag{2.2}$$

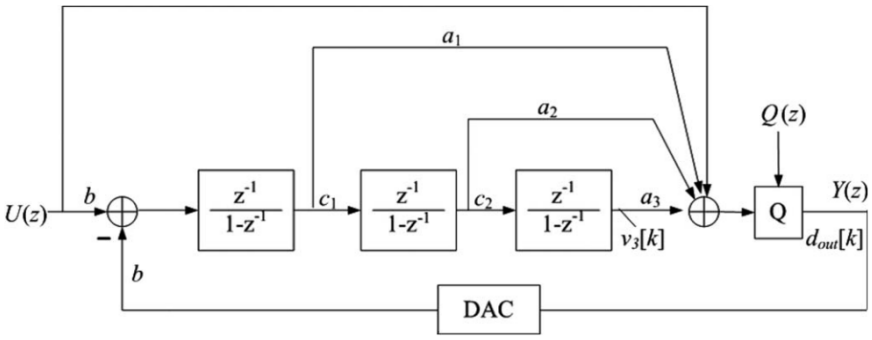


Figure 2.9: Incremental ADC architecture

analysis shows that

$$\left| u - \frac{6}{M(M-1)(M-2)} \sum_{m=0}^{M-1} \sum_{L=0}^{m-1} \sum_{k=0}^{L-1} d_{out}[k]\, V_{ref} \right| \leq \frac{6V_{ref}}{bc_1c_2M(M-1)(M-2)} \tag{2.3}$$

Hence, choosing the decimation filter as a scaled triple accumulator like that its output is

$$D = \frac{6}{M(M-1)(M-2)} \sum_{m=0}^{M-1} \sum_{L=0}^{m-1} \sum_{k=0}^{L-1} d_{out}[k]\, V_{ref} \tag{2.4}$$

the conversion error satisfies

$$|\varepsilon| = |u - D| \leq \frac{6V_{ref}}{bc_1c_2M(M-1)(M-2)} \tag{2.5}$$

Even for high-resolution converters, M needs to be only of the order of a few hundred clock periods. Hence, the IDC may spend a considerable time in the sleep mode between conversions, reducing its power dissipation. Alternatively, a single IDC may be multiplexed between many sensors or channels.

Besides, the power dissipation of the IDC may further be reduced by cascading it with a Nyquist-rate ADC, for example a SAR ADC [24, 25].

2.2 SUCCESSIVE APPROXIMATION REGISTER ADCS

The architecture of a SAR ADC is shown in Figure 2.10. The basic converter consists of a comparator stage (*Comp*), the successive approximation register (*SAR*) and the *DAC*. A *S/H* and an *anti-alias filter* are added to limit the maximum analog input frequency and to convert the continuous time input signal into a discrete time signal. At the beginning of the conversion the MSB is switched on and the input signal is compared to the output signal of the DAC. When the input signal is larger than the output signal of the DAC, then the MSB remains on, the next bit is switched on and a comparison will be performed. A *bit-by-bit* operation is performed in this system to bring

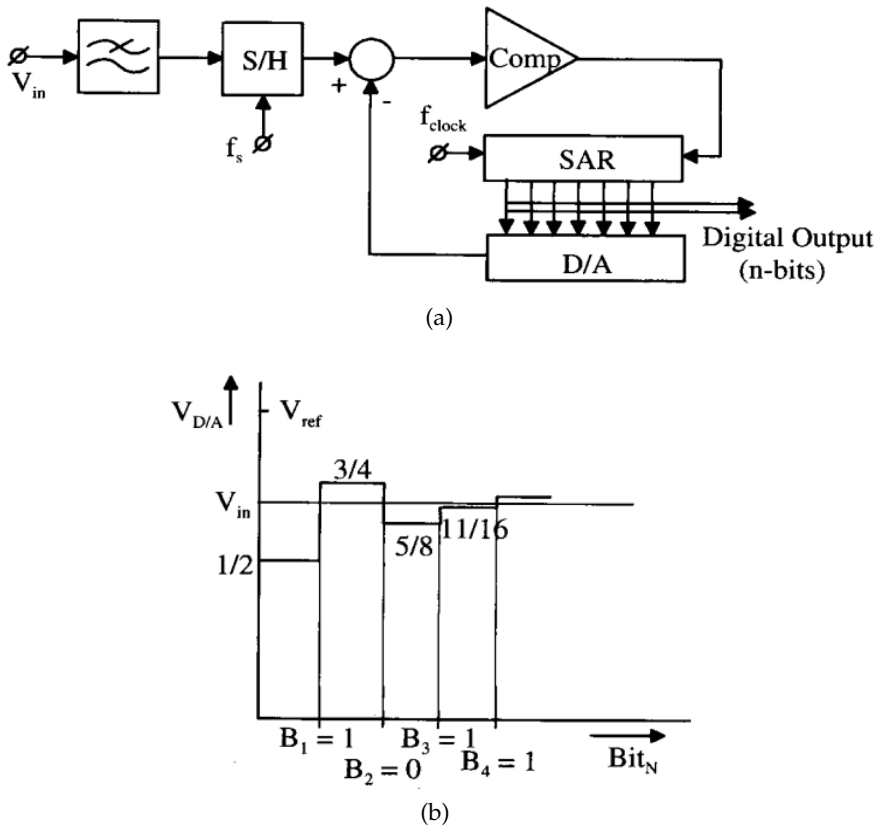


Figure 2.10: SAR ADC architecture

the D/A output signal within 1LSB to the time discrete input signal. In the lower part of Figure 2.10 the conversion procedure as a function of bit weighting is shown. The output value in the figure equals 1011. A complete conversion in this system requires N switchings and comparison operations to convert the input signal into a N-bit digital output value. The conversion time equals

$$T_{conversion} = N \cdot T_{settling} \tag{2.6}$$

The *settling time* is defined as the time required to settle within LSB of the DAC. The linearity and accuracy of this system depends on the DAC.

Different SAR architectures originate from different ways of implementing both the DAC and the digital controller.
An efficient way of implementing the binary search algorithm uses a Charge-Redistribution (CR) DAC with binary weighted capacitors. As originally proposed in [20], the S/H function can be realized by the DAC itself. The charge redistribution principle has several advantages. It is hardware efficient, since it requires a minimum amount of logic and simple analog circuits, with no need for "precision", high gain opamps. By leveraging both parallelism [7, 26] and redundancy [26], efficiency and yield of CR-SAR ADCs can be improved by interleaving slower and power-efficient ADC slices, at the cost of a larger area.

Also, more sophisticated switching techniques have allowed reaching record energy efficiencies [27], by charging and discharging the capacitances of CR-SAR ADCs in multiple steps or via a split capacitor bank [26, 28, 29].

A CR ADC can, however, require generating clock signals at a higher frequency than the sample rate to drive the comparator, or sizing the comparator for the worst case comparison time. Moreover, power-hungry active buffers may be needed for input and reference voltages to settle within the required time and accuracy, while driving high capacitive loads [30]. To solve the first issue, asynchronous processing can be successfully adopted [7, 29]. To alleviate the second issue, the input capacitance of the whole converter can be decoupled from the DAC capacitor array [27]. A DAC composed of unit capacitors (in fully differential topologies) for bits of resolution is, however, still necessary, with possibly additional hardware to support multi-step redistribution. A detailed analysis of the power consumption and the linearity of capacitive-array DACs employed in SAR ADCs can be found in [31]. At moderate resolutions, capacitor matching and insensitivity to (nonlinear) parasitics [27] provide the lower bound for the unit capacitor size, which is certainly the ultimate limitation to further area and power reductions in charge redistribution ADCs.

The need for more efficient and compact DAC implementations has motivated the use of series capacitive ladder networks [7], to make the input capacitance independent of the ADC resolution, or non-binary successive approximation DACs [30], to relax the settling requirements on DAC and buffers at the cost of additional conversion steps and digital processing. However, a series ladder structure can still be vulnerable to the parasitic capacitance, especially when the capacitor is implemented as a low-cost Metal-Oxide-Metal (MOM) capacitor available in a standard digital CMOS process. Furthermore, if the size limit of this ladder is pushed down, expensive digital post-processing and calibration becomes unavoidable to compensate for random errors [7]. As an alternative approach, asynchronous Charge-Sharing (CS) SAR ADCs [8, 9] with two-stage passive S/H have also proved to bring power savings on the fast settling circuitry that provides both the input and the reference voltages in the feedback DAC. [11]

2.2.1 *Low-power SAR ADCs*

An example of a micropower successive approximation ADC is described in [32]. The important part of this design is the DAC and the current subtracter circuit. The architecture is identical to the previous system. In Figure 2.11 the DAC architecture is shown. In the DAC segmentation in combination with an R/2R resistor network is shown. The segmented section is driven by the segment decoder. This seg-

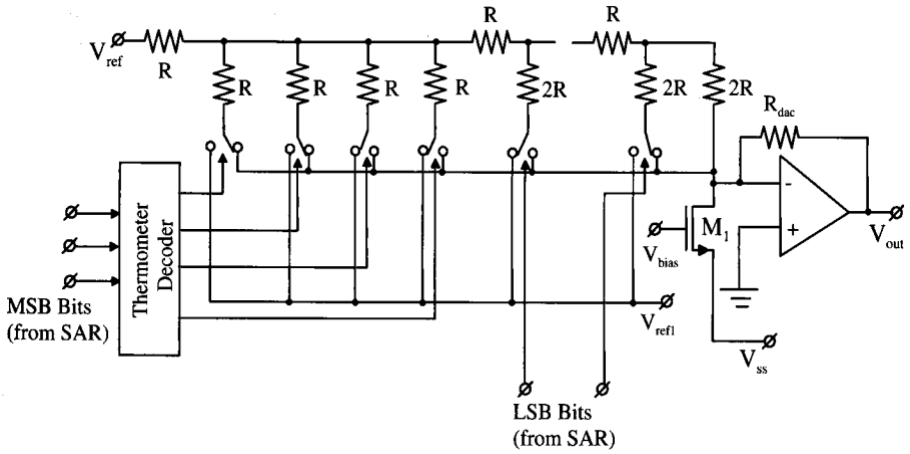


Figure 2.11: Low-power DAC architecture

mentation is used to ensure monotonicity of this converter, while the R/2R network reduces the total amount of elements to a practical applicable size. In this system a 3-bit segmentation is used. Note that not all segmented resistors have been drawn in this circuit diagram. Then a 5 to 7 bit binary weighted R/2R network completes the DAC. At the output of the network an operational amplifier is shown. This amplifier guarantees the output level after the bit switches to be at ground level. The bit switches have to be designed in such a manner that equal voltage drops across the switches are generated to avoid accuracy problems. Furthermore a DC bias current via transistor M1 is applied to the system to obtain a proper DC biasing adjustment. The R_{dac} determines part of the gain in the subtracter/comparator chain. The successive approximation logic drives the MSB and LSB bit switches. Because of the segmentation more clock pulses are needed to obtain a conversion. In a full binary weighted 10-bit system only 10 clock cycles are required while in this system with 3-bit segmentation the total amount of clock cycles increases to 7 + 7 = 14 clock cycles. This can be seen as a disadvantage of this system.

2.3 COMPARATOR-BASED BINARY SEARCH ADCS

By further pushing the quest for scalable and "digital" architectures, ADCs using *comparator-based asynchronous binary search* have been also proposed. Both DAC and digital controller can be completely avoided in a CABS ADC.

The operating principle of the CABS converter is based on binary search, the same principle that is used in a SA ADC. But instead of approximating the input signal, in the CABS architecture comparators with built-in thresholds are used to bracket the input signal (shown in Figure 2.12). Similarly to a flash converter, the sampled input signal is applied to all comparators, but unlike flash converters, not all

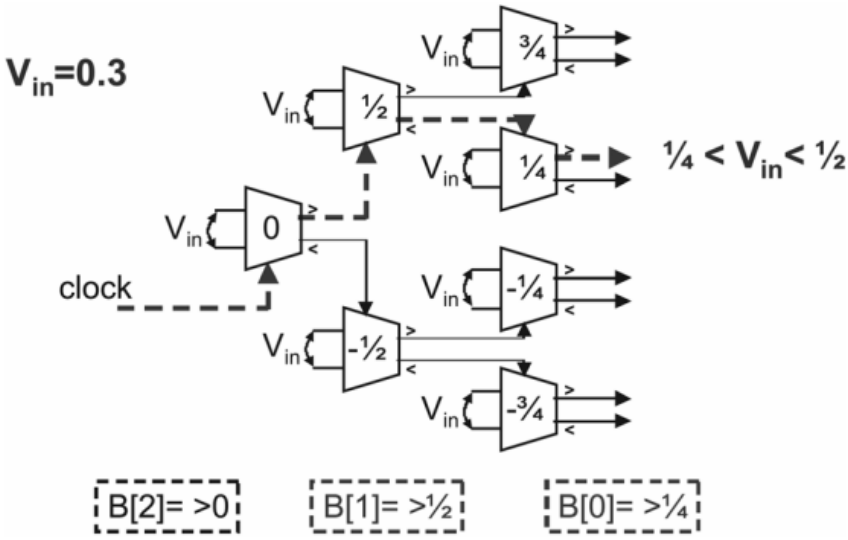


Figure 2.12: Operating principle of a CABS ADC, shown for a 3-bit ADC with 0.3 input on a 0 to range (dashed lines indicate active path)

comparators are clocked. Instead the comparators are connected in a binary tree, in which the root comparator compares the input signal with zero and based on its decision asynchronously triggers one of its children, comparators with threshold $1/2$ and $-1/2$. If the input signal is greater than zero the comparator with threshold $1/2$ is triggered, if smaller than zero, the comparator with threshold $-1/2$ is triggered. This second comparator in turn triggers one of its children in the third layer, closing in on the input signal. Based on the outputs of the activated comparators an unsigned binary code is derived: a logic 1 is encoded for "greater than" and a logic 0 for "smaller than" (Figure 2.12). In this manner the input value is converted into an unsigned binary representation.

In Figure 2.12 the operating principle is illustrated for 3 bits, but it can be extended to more bits. The number of comparators in the tree is for an n-bit converter equal to $2^n - 1$ as is the case in a (full) flash converter, but only comparators are triggered. So the power consumption is reduced to the power required to do a minimal binary search, hence much smaller than the power of a parallel search implemented in a standard flash converter. The quantization starts on the rising clock edge: successively, layer by layer, one comparator decides. A falling clock edge resets all the activated comparators following the same path through the tree that was followed during quantization. As a result this architecture does not need explicit controller circuitry.

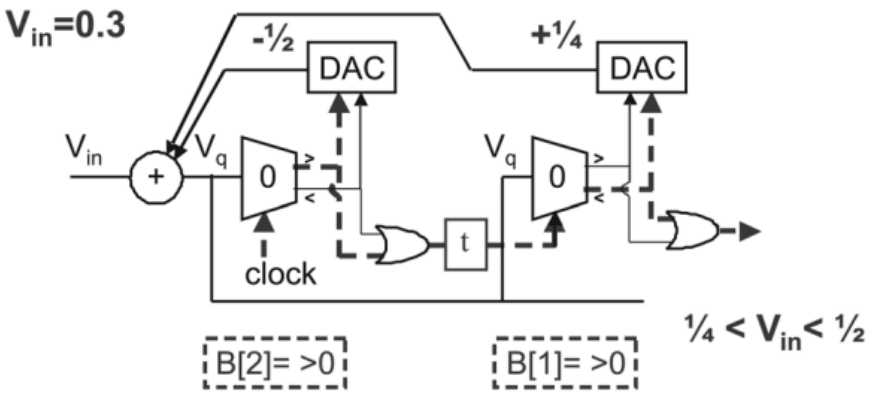


Figure 2.13: Operating principle of a SAR-CC ADC, after sampling the input every bit is determined by a comparator which controls a feedback DAC (dashed lines indicate active path)

2.3.1 *SAR ADCs using CC*

Although the CABS architecture reduces power consumption in an ADC, it does have the disadvantage of exponential complexity. This can be avoided by implementing a SAR ADC in which the comparators also implement the controller function (SAR using a Comparator-based Controller (CC)). The operating principle of this architecture is shown in Figure 2.13. Once again the sampled input signal is applied to all comparators in the architecture in parallel. But instead of a tree, a chain of comparators is implemented in which each one controls a feedback DAC that modifies the sampled input signal. The algorithm implemented on this architecture is also successive approximation. Normally this algorithm is implemented by a synchronous or asynchronous controller, in the presented approach the comparators serve as state machine of the algorithm: if a comparator is reset both "greater than" and "smaller than" outputs are zero, if it is activated one of them is logic 1 and its feedback DAC modifies the input signal. The output of a comparator however can not immediately trigger the next one in the chain, since the DAC feedback signal needs to settle. Hence, an appropriate delay block with delay τ is inserted, such that the next comparator is only triggered when the feedback signal of the DAC is stable. Also this architecture is edge triggered, a rising clock edge at the start of the chain starts the quantization, a falling clock edge resets the structure. [5]

Recently, a Binary Search (BS) ADC implementation has been proposed in [6], where a switched reference voltage network and a reference-prediction circuit are used to alleviate the calibration burden, and avoid the exponential growth in comparator count respectively. The number of comparator scales linearly with the ADC resolution, at

the expense of increased complexity in the switching network, which now tends to grow exponentially with n.

2.4 THRESHOLD-CONFIGURING ADCS

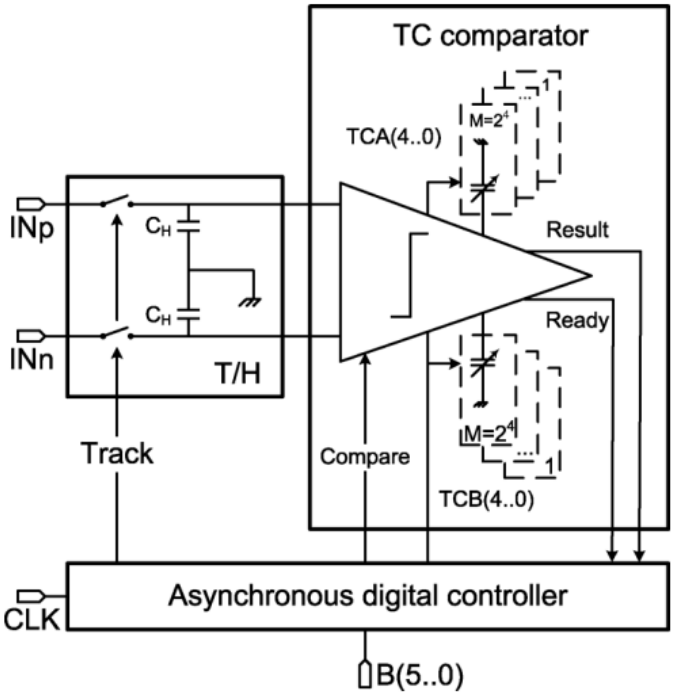
The evolution of ADCs in the last few years has been driven by the quest for both power and technology scalable architectures. Compatibility with advanced digital CMOS processes determines the success of highly digital architectures, such as SAR and $\Sigma\Delta$ ADCs for ultra-low power applications as wireless sensing nodes or implanted biomedical devices. High research effort has been done in the recent past to introduce highly digital architectures in ADCs for both high speed and low power applications. The use of digital regenerative comparators is a key factor in the design of these novel architectures. TC-ADC was introduced in [10] and [11] for medium resolutions and speed, and low power applications. It combines the principles of operation of SAR ADCs with the use of a regenerative comparator, a programmable array of threshold configuring transistors, and an asynchronous controller, allowing to implement together the ADC and DAC functionalities while increasing the power scalability.

2.4.1 *TC-ADC Principle of Operation*

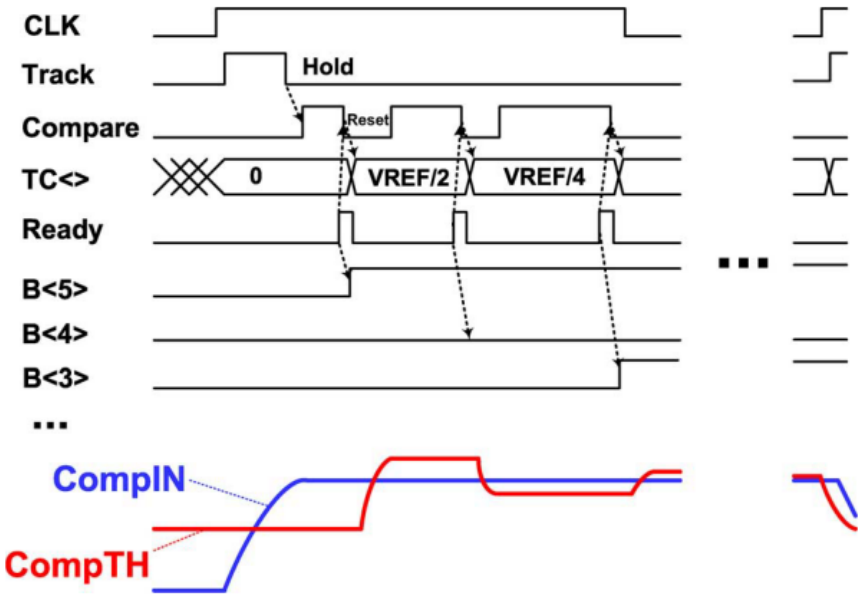
The simplified schematic diagram of a TC-ADC is shown in Figure 2.14a. The principle of operation of such architecture is as follows:

A. The Track and Hold (T/H) circuitry is responsible for sampling the analog signal – by means of its switching devices – and then keeping constant the input terminals of the comparator – by means of the holding capacitors – until the decision is completed. This funtion is done at each clock cycle, controlled by the track signal generated in the controller.

B. The asynchronous controller initializes the comparator threshold to $0.0V$, and generates the *Track* and *Compare* signals needed to perform the first decision, which is triggered at the falling edge of the *Track* signal, as shown in Figure 2.14b. As soon as the decision is taken and notified by the raising edge of the comparator signal *Ready*, the MSB of the output word, $B[5]$, is determined and the comparator is reset by the falling edge of the *Compare* signal. The threshold $CompTH$ can then be set to its next value, e.g., either $+V_{REF}/2$ (if $CompIN = (IN_p - IN_n)$ is larger than 0) or $-V_{REF}/2$ (if $CompIN$ is smaller than 0) based on the comparator decision. A new comparison is then triggered and the signal sequence described above repeats until all output bits, from the MSB to the LSB, are determined.



(a) TC-ADC schematic diagram



(b) Timing diagram of a TC-ADC conversion cycle. $CompIN = (IN_p - IN_n)$ is the input differential voltage at the comparator, $CompTH$ is the comparator threshold voltage as set by the threshold configuring devices at each conversion step

Figure 2.14: TC-ADC architecture

The asynchronous controller provides the *Track* (*Hold*), *Compare* (*Reset*) signals, and latches the output bits at the end of each conversion. Based on each comparison's result, the controller computes, using low complexity combinational logic, the necessary configuration words to adequately shift the threshold for the next cycle. To program comparator's thresholds, two binary-scaled arrays of switchable capacitors connected at the outputs of the latch are exploited. The digital words $TCA[4:0]$ and $TCB[4:0]$, shown in Figure 2.14a, control the amount of imbalance $\frac{\triangle C}{C}$ in the load of one of the two comparator outputs, which in turn generates a shift in the comparator trip point. Each switched capacitor in the arrays is implemented using PMOS transistors with short-circuited source and drain terminals, behaving as MOS varactors. [13]

2.4.2 *Architectural Details*

By restricting the focus to medium resolution and speed applications, it is possible to operate conversions with a minimal number of components.

A dedicated feedback DAC to generate the error signal between the sampled input and the ADC output is avoided, since the comparator itself implements both the A/D and D/A functionalities. In fact, since moderate resolutions are the target, the CR-DAC of a traditional SAR ADC is conveniently replaced by a TC-ADC. The TC-ADC will also consist of binary-scaled arrays of capacitors, thus presenting the same linearity and matching requirements as the CR-DAC, for a given resolution. However, while in CR-DACs capacitors need to be highly linear, since they provide the gain between charge and voltage, in a TC-ADC capacitors are switched *on* or *off* in a mostly "digital" fashion to trim the trip points. These capacitors can then be implemented using small transistors out of a standard digital technology. Moreover, since no capacitors are used for the MSB decision, $(2^n - 2)$ unit devices are required to realize an n-bit TC-ADC, which is one half of the total number of units (2^{n+1}) in fully differential CR converters.

Although the comparator threshold generation mechanism can be affected by supply voltage and temperature variations, it still brings area advantages and enough linearity for the target resolution [11]. Moreover, "reference" voltages (except, of course, for supply voltages) do not need to be externally generated and buffered by possibly power-hungry circuits. At the same time, the asynchronous operation based on a self-timed comparator [7, 33] avoids high-speed clock generation and buffering, as well as comparator design for the worst case conversion time.

Since the target is compact solutions at moderate speeds, the serial conversion process is performed out of only one comparator. All thresholds are, therefore, generated out of the same comparator,

which needs to be steered by a controller during its n conversion cycles. For this reason, a TC-ADC may be slower and less power efficient than a BS-ADC without controller. However, its area and complexity advantage can be substantial since the controller area, which scales approximately linearly with n, is generally negligible with respect to other components. On the other hand, comparator area in BS-ADC with built-in thresholds shows exponential complexity, and has been reduced to linear only with sophisticated reference switching schemes [6]. Finally, an offset compensated TC-ADC at moderate resolutions can basically operate without calibration, while $2^n - 1$ trip points need to be calibrated in a CABS-ADC with built-in thresholds.

2.4.3 *Circuit Implementation*

Here are presented analysis and design details for all the building blocks of a 6-bit TC-ADC.

Track-and-Hold

The T/H simply consists of two NMOS switches (with no bootstrap circuits) that sample the input signal onto low-cost MOM capacitors. These are the only passive components in the ADC. The *Track* signal is generated from an external clock through a digital buffer.

Threshold-Configurable Comparator

The fully dynamic threshold-configurable comparator at the heart of the TC-ADC originates from a modified PMOS version of the *StrongArm* architecture [34], which has been also used in high-speed low-power flash ADCs [35, 36].

The working principle it is based on is fully described in Chapter 3.

Digital Controller

The controller generates the *Track* (*Hold*), *Compare* (*Reset*) signals, and latches the output bits at the end of each conversion (Figure 2.14b). Asynchronous operation enables higher sampling rates for the same power consumption, since there is no need to generate a clock at approximately n times the sampling frequency f_s for an n-bit ADC. Moreover, a new conversion step can start as soon as the previous one is completed in a domino-fashion, so that the sampling time needs not be sized for the worst-case comparator's decision time.

A conceptual, simplified block diagram of the controller is represented in Figure 2.15 . The timing signals of the ADC are all derived from a single ended 50% duty cycle clock input. At the beginning of each conversion, the clock drives the output register to latch the result of previous conversion, and a pulse generator to provide the *Track* signal. Another pulse generator triggered by the end of the tracking

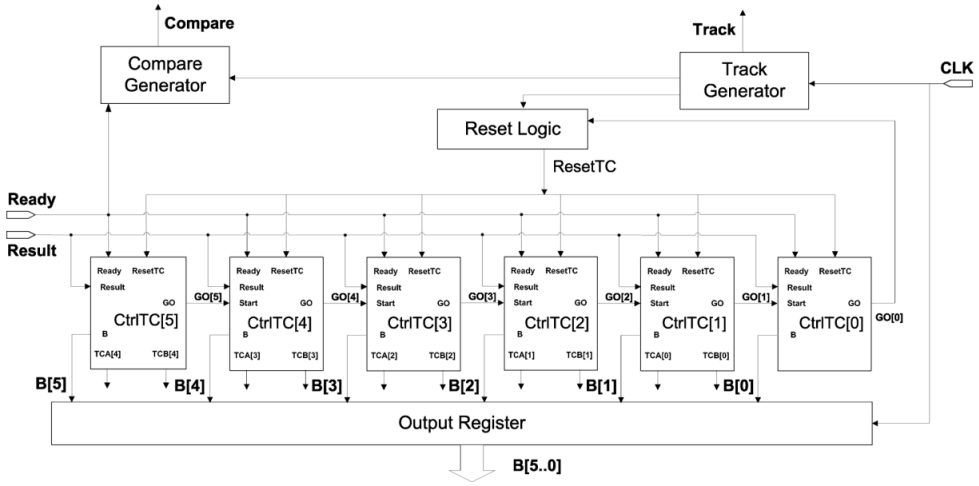


Figure 2.15: Simplified block diagram of the digital controller

phase drives the comparator and guarantees its proper reset. Finally, there are six cascaded Threshold Configuration control (CtrlTC) units, each of them includes registers and low-complexity combinational logic to store the conversion result into the output bit, compute the threshold configuration signals and activate the next unit in the cascade. All the units have the same structure, with the exception of the first and the last ones, which are adequately modified to accommodate the first and last comparator conversion of each cycle. The cascade of CtrlTC units build the SAR "shift-register", which operates as follows.

As soon as the input signal tracking starts, the *Track Generator* sends a reset pulse to all of the 6 CtrlTC units so that the threshold configuration signals $TCA[i]$ and $TCB[i]$ become low and no TC device is activated. The comparator is then ready for its first conversion. When triggered by the *Ready* signal from the comparator, the first unit $CtrlTC[5]$ will store the conversion result $B[5]$, and compute the configuration signals $TCA[4]$ and $TCB[4]$ to properly shift the comparator threshold for the next comparison. At the same time, the state variable $GO[5]$ is raised and activates the next unit $CtrlTC[4]$, which will now be the only one to be sensitive to the *Ready* signal from the comparator. After the end of the second conversion, the new bit value $B[4]$ is stored and the next configuration signals $TCA[3]$ and $TCB[3]$ are computed, while the control flow proceeds through $CtrlTC[3-0]$ until the last conversion occurs. At this point, the state variable $GO[0]$ will denote the end of conversion and will be used to trigger the reset of the TC logic for the next conversion cycle. If the whole cycle is not completed within one clock period, a new conversion will anyway start. The CLK signal is used to latch the current output B and reset the CtrlTC units.

In each CtrlTC unit, the configuration signals $TCA[i]$ and $TCB[i]$ are generated from the state variables GO and the outputs B using low complexity combinational logic as follows:

$$\begin{aligned} TCA\,[i] &= (\overline{B\,[5] + \overline{GO\,[i+1]}}) \cdot (\overline{B\,[i] \cdot GO\,[i]}) \\ TCB\,[i] &= (\overline{\overline{B\,[5]} + \overline{GO\,[i+1]}}) \cdot (\overline{\overline{B\,[i]} \cdot GO\,[i]}) \end{aligned} \tag{2.7}$$

After the first comparison cycle, when all the configuration arrays are deactivated, the MSB $B[5]$ determines which one of the two C_{TCA-B} arrays will be configured during the subsequent conversion cycles. For instance, when $B[5]$ is 0 and $GO[i+1]$ rises to 1, $B[i]$ and $GO[i]$ are still 0 and the $TCA[i]$ signal becomes high, thus activating the ith unit of the C_{TCA} array. As soon as the conversion step is completed, the signal $GO[i]$ is set and $TCA[i]$ will then evaluate its final logic level based on $B[i]$, i.e., the current comparison result. [11]

Part II

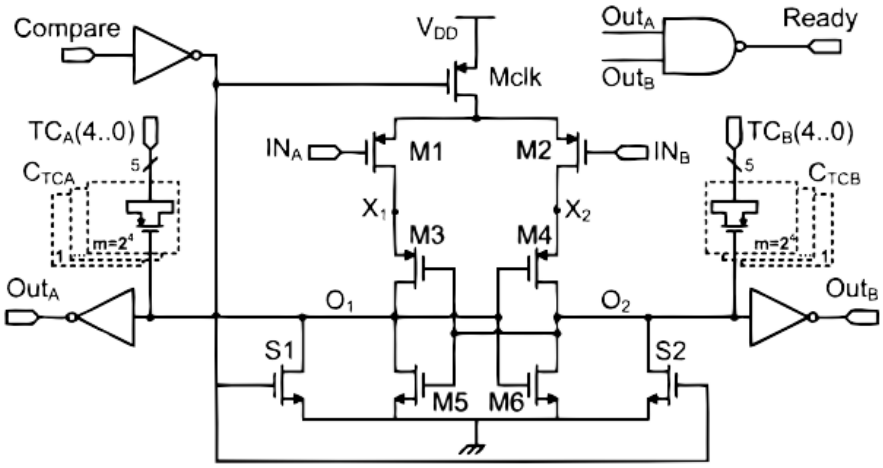
AN ULTRA-LOW POWER COMPARATOR

The main key in the TC-ADC operation is the understanding on how the digital comparator is influenced by the digitally configurable capacitor arrays. Regenerative comparators rely on the principle of operation of the well known latch–type voltage sense amplifiers used in digital circuits to read the contents of different memories [37]. Latch-type comparators are well suited for these applications as they achieve fast decisions due to strong positive feedback, but they are subject to static and dynamic random offset voltages as effect of the device mismatch [38, 39]. Moreover, regenerative comparators are based on the same principle of operation as novel Fast FeedThrough Logic (FTL) logic families for digital Gallium Arsenide (GaAs) [40] and CMOS [41, 42] circuits. The basic operation and design methodology are described in Chapter 3, while Chapter 4 presents the relationship between power consumption and full scale of the Threshold-Configuring (TC)-Comparator, compared with the radiated power vs distance characteristic of a Radio-Frequency IDentification (RFID) reader.

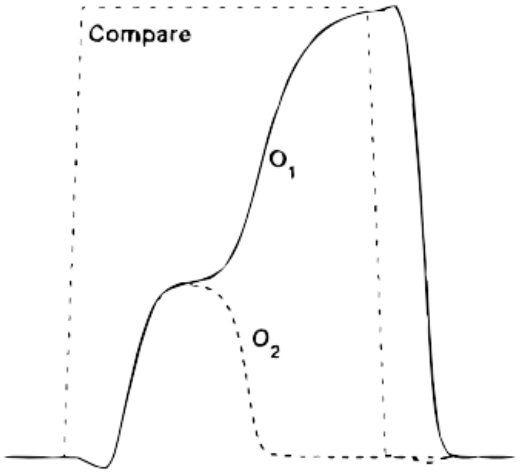
3

THRESHOLD-CONFIGURABLE COMPARATOR

As shown in Figure 3.1a[1], the core comparator topology consists of an input differential pair (M_1 and M_2) feeding a current into a regenerative back-to-back inverter pair (M_3 through M_6).



(a) Schematic diagram



(b) Waverforms at *Compare*, O_1 and O_2 nodes

Figure 3.1: Simplified Threshold Configuring regenerative Comparator

1 This *Simplified* version is derived from the TC-Comparator schematic originally presented in [11].

3.1 BASIC OPERATION

The *Compare* signal sets the operating phases. When *Compare* is low, switches NMOS S_1 and S_2 reset the comparator, pushing nodes O_1 and O_2 down to ground, so that identical initial charge conditions are put in both output nodes. In this phase, the comparator is inactive and cleared from the previous state. No current is drawn in the circuit as *Mclk* is off.

When *Compare* goes high, the comparator is activated: the Evaluation phase starts and *Mclk* is switched on, while *S*1 and *S*2 are switched off. In this way the input differential voltage is sensed and a decision is taken by the regenerative back-to-back inverter pair.

The Evaluation phase can be approximately divided into three sub-phases.

A. At first, M_1 and M_2 operate in saturation regime, nodes X_1 and X_2 are charged almost linearly while the cross-coupled inverters are off. In this sub-phase any voltage difference between IN_A and IN_B generates an imbalance between the drain currents I_1 and I_2 of M_1 and M_2 so that the charge rate of nodes X_{1-2} is not the same.

B. When the voltages of nodes X_{1-2} raise approximately to $|V_{Tp3}|$ (V_{Tp} is the threshold voltage of the PMOS transistors), a second sub-phase begins, in which M_{3-4} start conducting and produce a voltage difference between nodes O_1 and O_2 as well. In this sub-phase the common mode voltage at nodes X_{1-2} keeps growing approximately from $|V_{Tp3}|$ to V_{DD}, while nodes O_{1-2} charge up to V_{Tn5} (V_{Tn} being the threshold voltage of the NMOS transistors), still with an approximately constant slope.

C. In the third sub-phase, when M_{1-2} exit and M_{5-6} enter their saturation regions, the circuit mostly behaves as a cross-coupled inverter pair amplifying the initial output voltage difference to logic levels.

Finally, after the decision has been taken, no current is drawn into the circuit as either the n or the p transistor will be off in the inverters. A typical representation of the voltage waveforms at the internal nodes is shown in Figure 3.1b.

A NAND gate generates the *Ready* signal indicating when O_1 and O_2 assume opposite logic values and the comparison is done. This signal is used by the asynchronous SAR controller as a trigger to reset the comparator and continue the conversion process with another threshold configuration word. To avoid slowing down the conversion process, O_1 and O_2 are buffered by properly skewed inverters, as in [7]. For very small comparator inputs, the metastable voltage of nodes Out_A and Out_B is designed to fall below the NAND gate

threshold, to be interpreted as a "data ready" and pass the control to the next conversion step.

3.2 THRESHOLD GENERATION

If the implemented circuit was perfectly differential, the output would depend on the sign of the differential input, thus providing a reference threshold of 0. On the other side, any mismatch purposely introduced between the two half-circuits causes the comparator trip point to shift, which will result in a different "built-in" threshold. Differently from [35, 36], where the input pair device widths are balanced, a relative capacitance imbalance on nodes O_{1-2} is exploited to create the "reference" threshold.

A load difference $\triangle C_O = C_{O_1} - C_{O_2}$ at nodes O_{1-2} needs to be compensated by corresponding differences in both the charging currents of O_{1-2} and X_{1-2}, namely $\triangle I_3$ (flowing through M_{3-4}) and $\triangle I_1$ (flowing through M_{1-2}). Computing how these current differences translate into a trip point shift $\triangle V_{IN}$ is equivalent to analyzing the dynamic offset voltage due to output load mismatch in a latched comparator.

A few approaches have been proposed in the literature to analyze the input referred offset of dynamic comparators. As in the balanced method, described in [38], $\triangle V_{IN}$ is regarded as the compensation voltage needed at the input terminals, to cancel the mismatch effect and ensure that the comparator reaches its balanced status in spite of any load imbalance. This principle has been used, for instance, in [36], to evaluate the offset voltage induced by random MOS threshold voltage mismatch, current factor mismatch, and input pair device widths.

In this TC-Comparator, differently from [36] and [38], it is adopted a piece-wise linearization method, similar to the noise analysis technique proposed in [43], to achieve higher modeling accuracy. The bias point variations are tracked across the main comparator operation phases, and these are accounted for all the transient currents and voltages due to capacitance charge and discharge, as also demonstrated in [39] for a simple two-inverter latch structure. In particular, it is assumed that the circuit is solicited by a constant signal voltage $\triangle V_{IN}$, and a finite number of operating phases are defined as above (Section 3.1. BASIC OPERATION), where the circuit is linearized and analyzed. Transitions between phases are assumed instantaneous, and thus neglected. The evolution of the output voltage is obtained by solving the differential equations governing the equivalent circuit in each phase and combining results with continuity so that the starting point of each phase coincides with the final point of the previous one.

The input differential voltage $\triangle V_{IN}$ required to balance the cross-coupled inverters back in their metastable state during the latch re-

generation phase (sub-phase C), in spite of the preexisting imbalance, is calculated in [11] with a "pseudo-noise" analysis and simulation technique (recently advocated also in [44]), as a fast estimation method for transient performance variations due to device mismatch. It leads to the following formula:

$$\triangle V_{IN} \approx \frac{\left|V_{GS1} - V_{Tp1}\right|}{4} \frac{\left|V_{GS3} - V_{Tp3}\right|}{\left|V_{Tp3}\right|} \left(1 + \frac{V_{DD} - V_{Tinv}}{V_{Tn5}}\right) \frac{\triangle C_O}{C_O} \quad (3.1)$$

where V_{Tinv} is the metastable voltage of the latch M_{3-6} during sub-phase C, and V_{GSi}, V_{Tpi}, V_{Tni} denote the gate-source voltage, and the (PMOS or NMOS) threshold voltage of device i respectively. In particular, V_{Tinv} can be approximated as $V_{DD}/2$ for a non-skewed cross-coupled inverter pair. V_{GS1} is the "average" gate-source voltage of M_{1-2} in sub-phase A, and it can be approximated as $V_{CM} - V_{DD}$, V_{CM} being the input common mode voltage. V_{GS3} is the "average" gate-source voltage of M_{3-4} in sub-phase B, which can be approximately expressed as $\left(V_{Tn5} + V_{Tp3} - V_{DD}\right)/2$.

To create a capacitive delta, two binary-weighted arrays of digitally switchable capacitors, C_{TCA} and C_{TCB}, are implemented using small PMOS varactors, with drain and source short-circuited and gate connected to the comparator, on both sides, as shown in Figure 3.1a.

During the conversion process, capacitors are sequentially activated only on one side, based on the MSB value, to generate the closest threshold to the given input. Note that for ideally matched devices, the word $TCA = 0$, $TCB = 0$ implies a zero threshold comparator: any differential component in the analog signal will thus induce a delta between the M_{1-2} differential pair, which in turn will produce the latch decision. [11]

3.3 DESIGN METHODOLOGY

Both comparator and configuration devices must be codesigned to guarantee the global ADC performance, since capacitance at nodes O impacts the comparator speed. In fact, at the beginning of the evaluation phase, the linear charge time is mostly determined by the charging current I_0 set by the M_{clk} switch and the total capacitances at nodes X_1, X_2, as estimated in [37]. However, when regeneration starts, the time constant changes according to the transistors that are active at that time, being primarily determined by the approximate expression

$$\tau_R = \frac{(g_{ds1} + g_{m3})C_O}{g_{ds1}(g_{m3} + g_{m5}) + g_{m3}g_{m5}} \quad (3.2)$$

which reduces to $C_O/(g_{m3} + g_{m5})$ when $g_{ds1} \gg g_{m3} \cdot g_{m5}$, i.e., when M_{1-2} enter their linear region of operation. From Equation 3.1, the larger the PMOS device C_{on}/C_{off} ratio or input pair overdrive, the

larger the threshold shift. However, large threshold configuration devices may slow down the comparator, while a larger overdrive will also increase noise, as shown by the analysis in [43]. Since the maximum threshold shift determines the ADC built-in V_{REF}, C_{TCA-B} devices and input pair overdrive must be sized to provide the desired dynamic range. Based on estimations of the total decision time from [37] and Equation 3.2, in [11] it is derived a time-varying, nonlinear behavioral model of the comparator, which also includes non-stationary noise effects and nonlinearities in the threshold generation mechanism.

4

DESIGN SPECIFICATION REQUIREMENTS

The TC-Comparator performance optimization, object of this study, could not be properly elaborated whitout well-defined criteria to indentify the exact value of V_{th} for each digital code.

4.1 TC-COMPARATOR *power consumption* VS *full scale*

Looking into the best optimization methodology to adopt for the TC-Comparator elements sizing led to the plain necessity of being able to search and define the threshold voltage V_{th} related to each digital code included into the comparator full-scale. V_{th} is used indeed as "reference value" inside the range of v_{id} values corresponding to each digital code, due to the non-linear relationship between v_{id} and *digital codes* (Figure 7.3).

The first investigation and optimization method adopted was the *scanning mode* (described in Section 5.1.1; script of the corresponding `sheader.sp` file presented in Section A.7.1). This method of analysis by parameters variation turned out to be too slow for the main aim, that is the sizing optimization, but powerful for detecting the most precise value of V_{th} for each digital code included into the full-scale. In Figure 4.1 it is described how the scanning mode can detect each V_{th} value with the required precision, in a *performance* vs v_{id} output plot. Due to the finite precision provided by the calculator and the intrinsic delay time shown by the comparator before the output signals go to the High or Low level (Figure 5.5b), the *performance* vs v_{id} traces result non-asymptotic. Anyway, V_{th} values can be limited within a defined range with sufficient precision, by applying the following iterative algorithm:

1. Fix a digital code.
2. Set the region to a maximum (absolute) value. (Note that only half of all possible voltage values are swept. Note also that the limit value for the first iteration should be the largest possible, that is the power supply voltage value, in this case $-1.0V$).
3. Sweep inside the defined region.
4. Plot the output performance vs v_{id}.
5. Redefine the (narrower) sweeping region.
6. Go to 3.

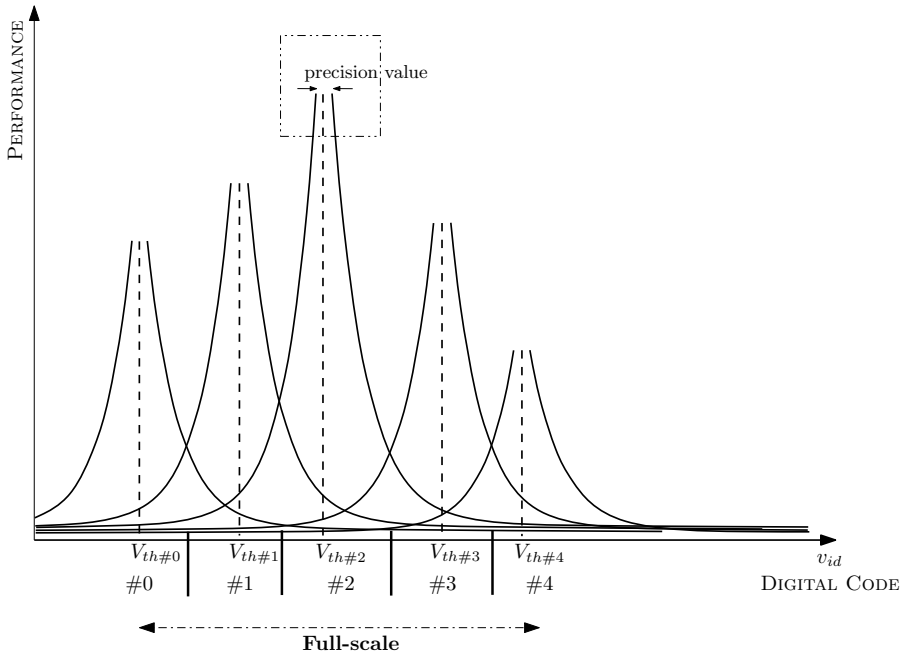


Figure 4.1: Qualitative output of *scanning* optimization method

Note that values of only half of the x-axys are provided. This choice is due to the inner simmetry of the two comparator branches, which allow to avoid useless data processing during simulations.

The v_{id} variation modality is provided to the scanning optimization method as a differential voltage obtained by varying v_{i1} and v_{i2} signals over time (Figure 4.2). The sweep trend is linear.

Actually, the main script contained into the `sheader.sp` file operates by iterating, for each digital code, two main steps:

A. Obtain the sweep region

B. Do the sweep

where point B refers to the entire algorithm described just above. The `sheader.sp` file is built by concatenating other three files:

`pre-header.sp` and `post-header.sp` containing fixed parameters (scanning resolution, ...);

`vid-header.sp` containing the varying parameters which define and redefine the sweeping region; updated parameters are obtained from the output measurements file, through a search and compare with the previous ones, by keeping the one which makes the region narrower. The process is iterated until the required precision is reached.

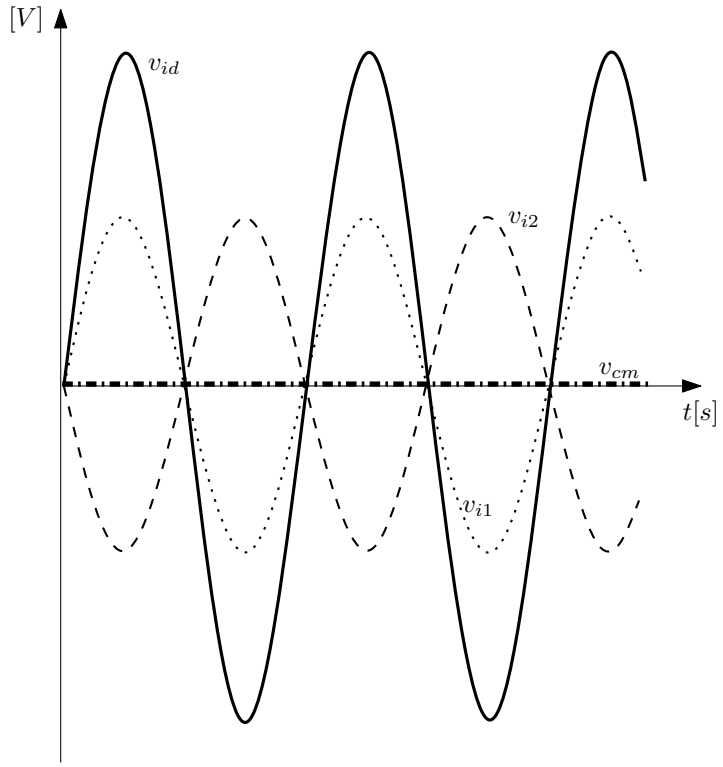


Figure 4.2: v_{id} values generation for scanning

4.1.1 *Results*

By applying the scanning optimization method iteratively, a precise relationship among transistors size, power consumption and full-scale (v_{id} range) was obtained:

device	Size	Power consumption	FS
M_{clk}	⇑	⇑	⇑
M_{1-2}	⇑	⇑	⇑
M_{3-4}	⇓	⇓	⇑
M_{5-6}	⇓	⇓	⇑

Table 4.1: *Size* vs *Power consumption* vs *Full-Scale* relationships

Note that in Table 4.1 relationships for each device size variation were obtained by keeping the other transistor size constant.

Note also that power consumption is calculated as an average power:

$$\langle P \rangle = \frac{1}{T} \int_0^T P_{comparator} \cdot dt \tag{4.1}$$

where $P_{comparator} = \sum P_{MOSFET}$, with $P_{MOSFET} = \sum V_i I_i$.

Referring to Figure 3.1a, if the M_{clk} (or M_{1-2}) transistor size is increased, its inner resistance will reduce, by leaving more current to

flow from the power supply V_{DD}, with an increment of the overall power consumption. An other consequence of the size increasing is also the change of the X_1 (or X_2, respectively) voltage node which, because of the smaller inner resistance of M_{clk} (M_{1-2}), will have a value closer to V_{DD}. For this reason, the voltage gap $\triangle V$ between GND and X_1 (X_2), that is the Full-Scale (FS) range, increases (Figure 4.3).

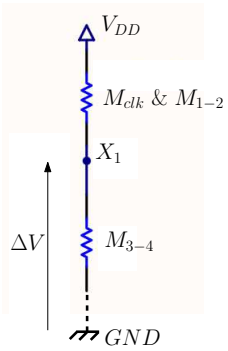


Figure 4.3: Inner resistance Model for the comparator left branch

The same *size* vs *power consumption* relationship is valid for the M_{3-4} (or M_{5-6}) transistor, where the size reduction of the chip area entails a bigger inner resistance, which leads to a smaller current flow from V_{DD} towards GND. On the other hand, there is a different reason behind the inverse trend of *size* vs *full-scale*. By considering a concentrated parameters model of the TC-Comparator (Figure 4.4) it is

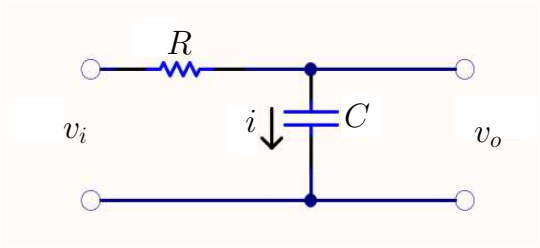


Figure 4.4: TC-Comparator concentrated parameters Model

easy to find a first order solution for the current $i(t)$ flowing through the circuit, in fact it is an RC circuit, whose well-known solution is given by

$$i(t) = \frac{dQ}{dt}(t) = -\frac{Q_0}{RC} \cdot e^{-\frac{t}{RC}} = I_o \cdot e^{-\frac{t}{RC}} \tag{4.2}$$

with $Q(t)$ the charge on the equivalent capacitor C at time t and $Q_0 = Q(t = 0)$.

If $i(t)$ is plotted (Figure 4.5), it is possible to observe that with a constant current flow (by fixing M_{clk} and M_{1-2} sizes), an increment of the inner resistance R of M_{3-4} and M_{5-6} due to their size reduction would necessary lead to an increment of the time t required to discharge the load capacitances. This situation is not possible because the TC-Comparator is designed to balance load differencies, so the

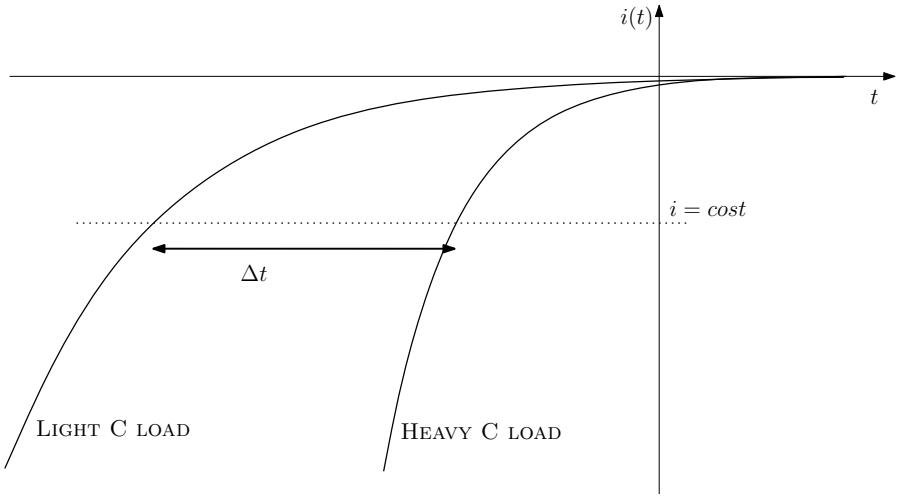


Figure 4.5: Qualitative analysis of the current i flowing through the comparator branches

only solution is a current increase, which once again implies a bigger FS range.

4.2 RFID READER *radiated power* VS *distance*

The DC chip power consumption (P_{cons}) and the efficiency in the RF into DC power conversion (η) are the factors that limit the operation range in a passive tag. For better understanding this statement, the Friis Transmission Equation (4.3) is introduced. This equation gives the power transmitted from one antenna to another, in free space:

$$\frac{P_r}{P_t} = G_r G_t \left(\frac{\lambda}{4\pi d}\right)^2 \tag{4.3}$$

Here P_t is the power fed into the transmitting antenna at its input terminals; P_r is the power available at the output terminal of the receiving antenna; G_r and G_t are the gains of the receiving and transmitting antenna, respectively; λ is the wavelength; d is the distance between the antennas (see Figure 4.6).

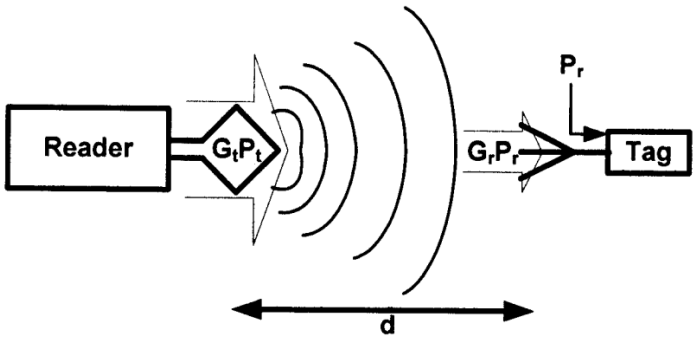


Figure 4.6: Illustration of the Friis Transmission Formula

For proper operation it is indispensable that

$$P_r \geq \eta P_{cons} \tag{4.4}$$

where P_{cons} is the upper bound of the power consumption on chip and η is the RF into DC power conversion efficiency. If condition (4.4) is not verified, there will not be enough power available to supply the chip. Assuming that $P_t G_t$ is limited by governmental regulations in Europe to $2W$ Effective Radiated Power (ERP), with a frequency of $868MHz$, a typical value for G_r is $2.5dBi$, with polarization losses of $3dB$; the function P_r vs d is illustrated in Figure 4.7.

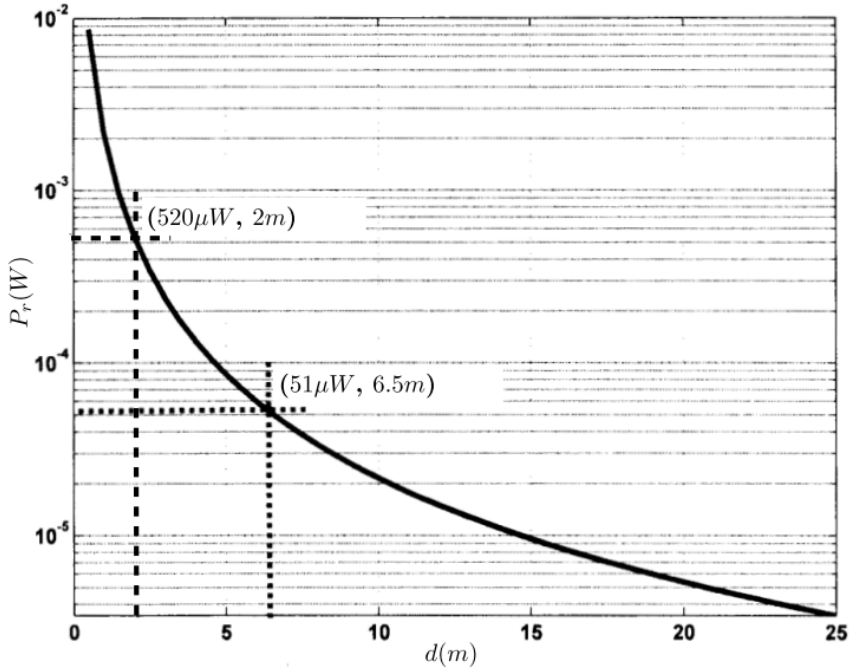


Figure 4.7: Available power at the output of the receiving antenna as a function of the distance

The figure shows the power that reaches the tag as a function of the distance, without considering obstacles and reflections. It is clear that tags consumption severely limits the operation range of the system. For an operation range of 2 meters, it will require a factor $\eta P_{cons} \leq 520\mu W$. Hence assuming $\eta = 20\%$, the upper bound of the power consumpion on chip with 2 meters between antennas is $P_{cons} = 2.6mW$ [45].

Part III

SIZING AND OPTIMIZATION

Simulations developed in this study aim at obtaining the best chip sizes for a power optimized design of a TC-Comparator. Wonder to attain this goal, a Simulation environment set-up must be preliminarily defined (Chapter 5). Afterwards, process corner and variation parameters are set into the simulator scripts for the selected technology process model (Chapter 6). Finally, results are obtained and analyzed in different linearity and mismatch conditions.

5 SIMULATION ENVIRONMENT SET-UP

To be able to evaluate the TC Comparator behaviour and which parameters are critical for its optimization, it is foundamental to develop a Simulation Environment where obtaining values related to the adopted technology, the UMC CMOS 90 nm "Standard Process 1.0V CMOS 1P9M".

5.1 HSPICE ENVIRONMENT STRUCTURE

HSPICE is a program that, starting from a circuit description and analysis options, outputs the analysis it has done on that circuit.

Doing an HSPICE simulation requires some input elements, as shown in Figure 5.1 ; in particular, it must be provided that:

- the *Technology File*, containing the reference parameters of the UMC CMOS 90 nm "Standard Process 1.0V CMOS 1P9M" technology;
- the *Netlist*, a set of files, typically with a `.sp` extension. Its structure is decribed in Figure 5.2 ;
- the *Measurements File*, where the parameters that it is necessary to monitor are defined;
- the *Corner of Process* with a defined *variance* σ, required for Monte Carlo Analysis (see Section A.4).

Although HSPICE produces many output files, the main one is the file with `.lis` extension, for example `circuit.lis`. This file contains all the important results from the HSPICE analysis: operating points, measurement results, error messages. Typically after simulating a circuit, it is better to check this file first in order to ensure there were no errors in the netlist. The other files that HSPICE generates are used by GnuPlot, a portable command-line driven graphic utility that allows to graphically plot HSPICE analysis results.

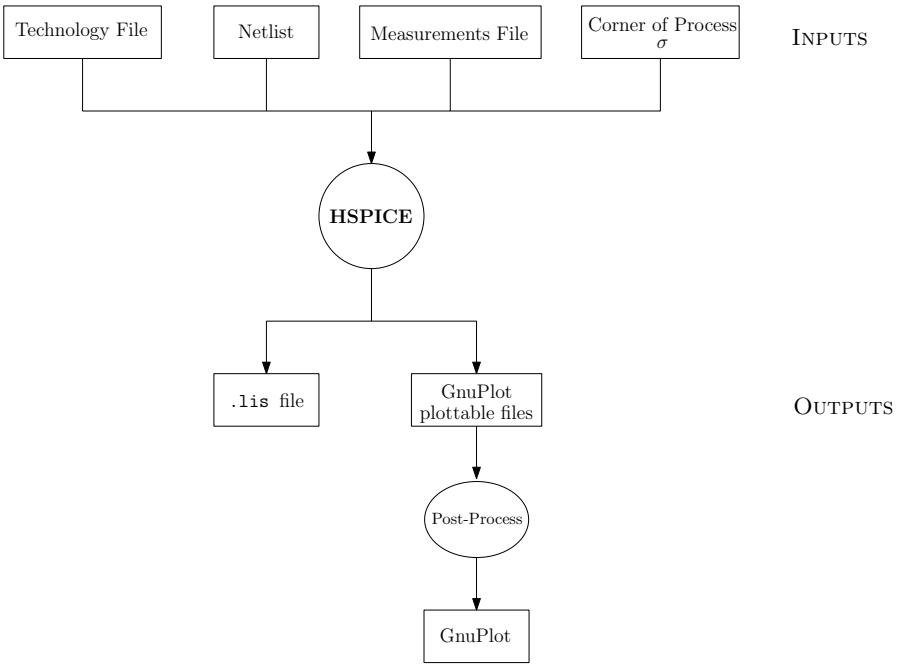


Figure 5.1: HSPICE Environment structure

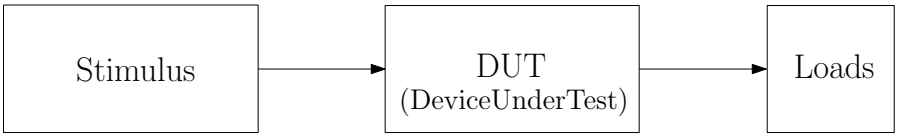


Figure 5.2: Netlist structure

5.1.1 *Files Hierarchy*

From a detailed analysis of the virtual workspace used during the simulation workflow, it appears what follows:

`bin/` It contains the main command files used to run the simulations; in particular:

`go` it works structurally *over* the *tcadc* command file, allowing to iterate the simulation with some parameters variations;

`tcadc` the properly called *main* file containing the preprocessing simulation directives (see Section A.6) . It presents the following internal structure:

TITLE 'Threshold Voltage Determination'

OPTIONS

LIB *L90_SP10_V031_MC_CORNER.lib*, where *CORNER* means that the *Typical* Corner Process Model is adopted

INCLUDES

- parameters
- global nets
- inputs/stimulus
- netlist

THE TOP NETLIST

MEASUREMENTS

PROBE POINTS

OPTIMIZATION MODELS contains variation values in order to evaluate performances in terms of [AVeRage (AVG) and Root Mean Square (RMS)] power dissipation, current and voltage (see Section A.7)

`sheader.sp` Scanning optimization mode (Section A.7.1). It contains a number of `.alter` commands, which modify any previous HSPICE sentence. In particular, these commands operate on x-axis parameters like

- v_{id}
- digital code

`oheader.sp` Bisection optimization mode (Section A.7.2)

`mheader.sp` Monte Carlo optimization mode (Section A.7.3)

`extractor` it is fundamental for data postprocessing, as it extracts data from the measurements output files and generates tables of data in the right format for GnuPlot graphic analysis.

`lib/` It contains the technology libraries for the UMC 90nm Process Models used by HSPICE. The version adopted is *L90_SP10_V031*.

`v7.6-v7.9/` Its contain the output files obtained by the simulations. The main one is

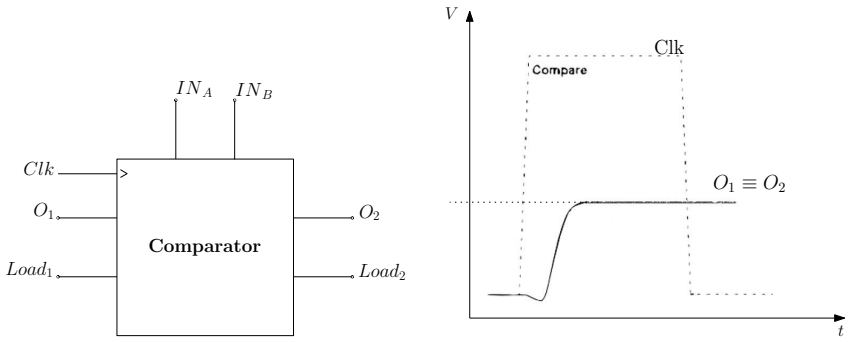
`circuits.inc` (see Section A.8)

- netlist of the components
- $cl0 \sim cl5$ capacitive loads initial values

Note that all the command files are written in Perl.

5.2 SIMULATION SETUP

By considering the Decision Phase of the TC-Comparator operating principle (Section 3.1, sub-phase c), it is possible to see how the circuit – behaving like a cross-coupled inverter pair – amplifies the initial output voltage difference to logic levels (Figure 3.1b). Therefore the key point in the analysis of the comparator operating principle comes down to the detection of the *Ideal Equilibrium Point* between the input differential voltage $v_{id} = (IN_A - IN_B)$ and the load capacitances connected (Figure 5.3).



(a) 7-pole TC-Comparator model

(b) Ideal waverforms at *Compare*, O_1 and O_2 nodes

Figure 5.3: Ideal Equilibrium Point *Inputs-Loads* in a TC-Comparator

Actually, this status of equilibrium cannot exist because of the presence of thermal noise (and mismatch) that randomly unbalances the relation between Imput and Load.

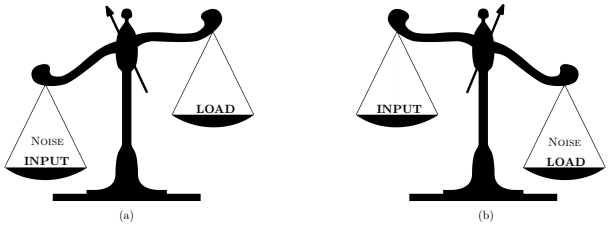


Figure 5.4: Imbalance between Input and Load due to thermal noise

The strategy adopted to pick out the searched threshold voltage V_{th} (for each digital code) can be described as follows.

Given a defined input load, for $IN_A \equiv IN_B$, outputs O_{1-2} are obtained. Subsequently, the minimum value of v_{id}

$$V_{th} = min(v_{id}) = min(IN_A - IN_B) \tag{5.1}$$

necessary for changing the outputs will be revealed (Figure 5.5).

This value can be numerically determined using an optimization procedure implemented in HSPICE, the *Bisection Methodology*, which uses a binary search method to find the value of an input variable (see Section A.5).

The Bisection Method requires the definition of a *precision variable*, used like decision parameter. The most suitable choice in this case is $\tilde{O}(T)$ measured at time T, defined as

$$\tilde{O}(T) = O_1(T) - O_2(T) \tag{5.2}$$

At this point, V_{th} is obtained by two successive simulations, which give:

$$\begin{aligned} V_{th}^{-} &= V_{th} - \overset{\varepsilon}{IN} \\ V_{th}^{+} &= V_{th} + \overset{\varepsilon}{IN} \end{aligned} \tag{5.3}$$

where

$$\begin{aligned} V_{th}^{-} &= v_{id}|_{O_1 > O_2} \\ V_{th}^{+} &= v_{id}|_{O_2 > O_1} \\ \overset{\varepsilon}{IN} &\simeq \varepsilon \cdot v_{id}, \quad \varepsilon \ll 1 \end{aligned}$$

By recursively repeating these simulations, it is possible to obtain a value for V_{th} with the required precision.

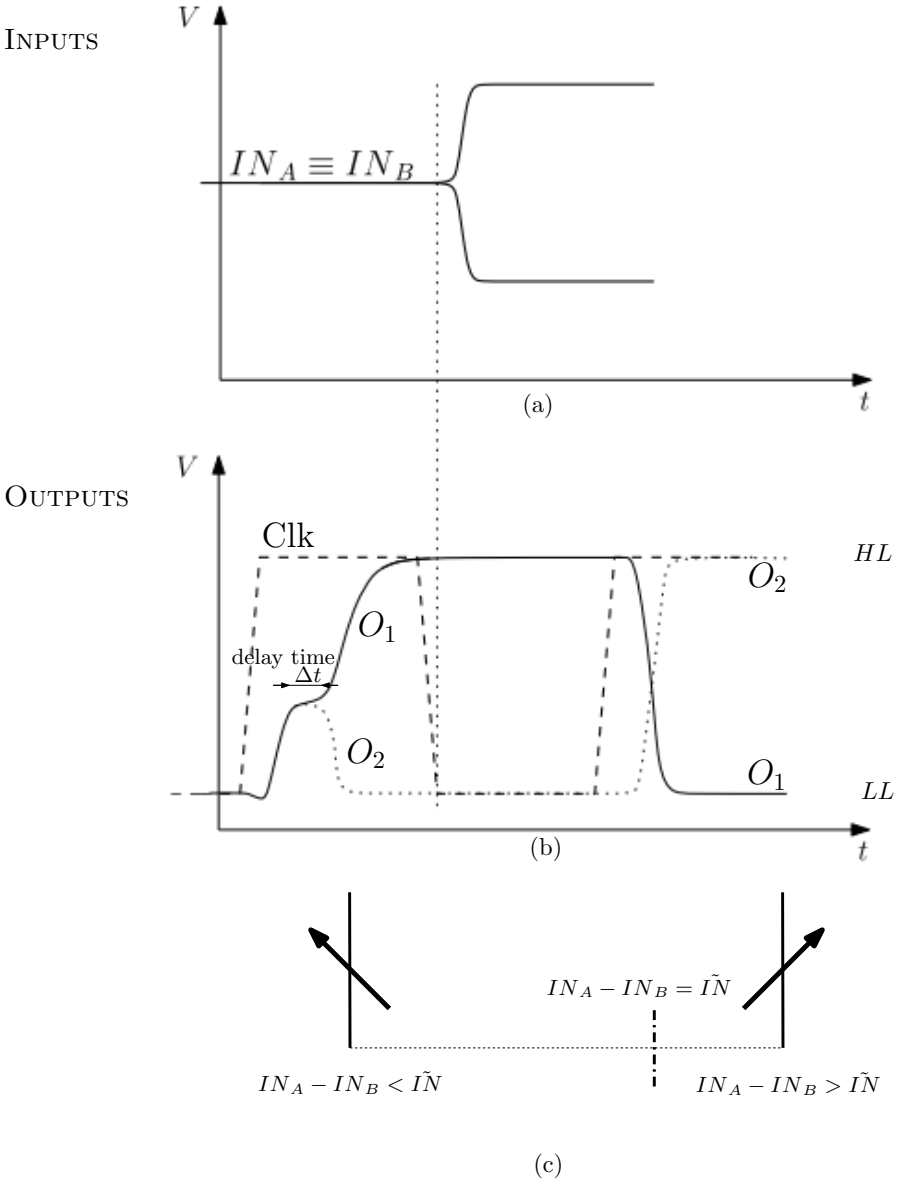


Figure 5.5: Threshold Voltage Detection for a TC-Comparator

6

SIZING FOR ULTRA-LOW POWER AND FS OPERATION

Defining Integrated Circuit performance without further qualification is somewhat subjective. However for the purpose of comparison, it is possible to quantify performance in terms of three parameters: power dissipation, functional throughput rate and communication limits. In computationally intensive structures, usually one has to address the best approach that can be pursued to partition a given system. For example, the question that may arise in this process is the choice as to whether one creates a very high speed single channel or to take the alternative approach of partitioning the system into parallel channels clocked at lower frequency. However, according to design specification of an ultra-low power ADC for blood pressure applications, the suitable technology needs to satisfy the following criteria:

- Very low propagation delay possibly less than $100ps/gate$;
- Low gate dissipation in order of $100\mu W/gate$;
- Very low dynamic switching energy, that is less than $0.1pJ$;
- Very high level of integration, greater than $50000\,gates$;
- High process yield.

6.1 CHARACTERISTICS OF CMOS DEVICES

The CMOS device is the key active device used in the Comparator circuits. Therefore a comparison of processes must consider the gain-bandwidth product f_t, transconductance g_m, parasitic capacitances and resistances of the Metal-Oxide-Semiconductor Field-Effect Transistor (MOSFET) to evaluate circuit performance.

6.1.1 *Threshold voltage*

Threshold voltage, V_{th}, is defined as the minimum gate electrode bias required to strongly invert the surface under the poly and form a conducting channel between the source and the drain regions. V_{th} is usually measured at a drain-source current of $250\mu A$. Common values are $2-4V$ for high voltage devices with thicker gate oxides, and $1-2V$ for lower voltage, logic-compatible devices with thinner gate oxides. With power MOSFETs finding increasing use in portable electronics and wireless communications where battery power is at

a premium, the trend is toward lower values of V_{th} and $R_{DS(on)}$, with$R_{DS(on)}$ the on-state resistance of a power MOSFET (defined in Section 6.1.3).

6.1.2 *Maximum drain-source Currents*

The drain-source current is the maximum current the MOSFET has available to drive the circuit. A large value implies the MOSFET is able to drive large capacitive loads, however the power dissipation directly increases. Sub-threshold current flows from drain to source when the gate to source voltage is below the pinch off voltage. This is when electrons are transported in the channel by diffusion and drift.

6.1.3 *Parasitic Capacitance and on-state Resistance*

The gate capacitance of a MOSFET is the major limiting factor of circuit speed if the interconnect lengths are short, as is the case in densely packed VLSI circuits. C_{gs0} and C_{gd0} are the gate-source and gate-drain zero-bias junction capacitances per micron gate width respectively. These are used to model the gate capacitance of MOSFET's.

The on-state resistance of a power MOSFET is made up of several components as shown in Figure 6.1 :

$$R_{DS(on)} = R_{source} + R_{ch} + R_A + R_J + R_D + R_{sub} + R_{wcml} \quad (6.1)$$

R_{source} =Source diffusion resistance
R_{ch} =Channel resistance
R_A =Accumulation resistance
R_J ="JFET" component - resistance of the region between the two body regions
R_D =Drift region resistance
R_{sub} =Substrate resistance
R_{wcml} = Sum of bond wire resistance, metallization and leadframe contributions

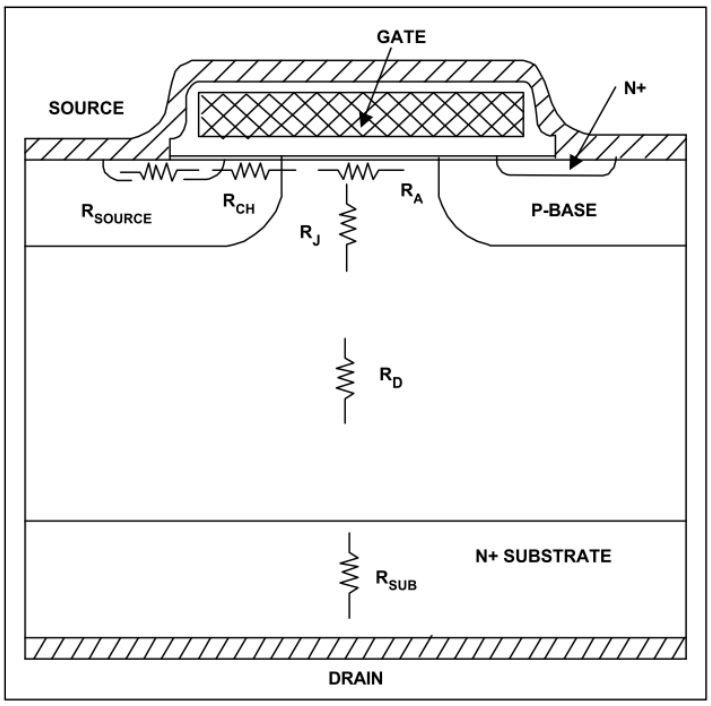


Figure 6.1: Origin of MOSFET Internal Resistance

6.1.4 *Transconductance and gain-bandwidth*

The transconductance of a MOSFET in digital circuits is related to the noise margin of a circuit through the gain in the transition region. The transconductance g_m is given by

$$g_m = \left.\frac{\partial I_{DS}}{\partial V_{GS}}\right|_{V_{DS}=cost} \tag{6.2}$$

It is also proportional to the transconductance parameter β. High g_m is desirable for voltage gain, however switching speed is also determined by the gain-bandwidth product f_t, given by

$$f_t = \frac{g_m}{2\pi\left(C_{gs} + C_{gd}\right)} \tag{6.3}$$

where C_{gs} and C_{gd} are small signal gate-source and gate-drain capacitances.

6.2 UMC CMOS 90NM PROCESS CHARACTERISTICS

UMC[1] provides a global model for design works of products manufactured using the 90nm LOGIC/MIXED_MODE 1P9M Process.

1 UMC is a leading global semiconductor foundry that provides advanced technology and manufacturing services for applications spanning every major sector of the IC industry. For more information, http://www.umc.com

To cope with the impact of process variations on circuit performance, worst-case models are provided along with the typical case and they are listed below. Note that "fast" here means lower threshold voltage, higher leakage and driving current, and "slow" means higher threshold voltage, lower leakage and driving current.

TT: Typical n-ch MOSFET and Typical p-ch MOSFET

FF: Fast n-ch MOSFET and Fast p-ch MOSFET

SS: Slow n-ch MOSFET and Slow p-ch MOSFET

FNSP: Fast n-ch MOSFET and Slow p-ch MOSFET

SNFP: Slow n-ch MOSFET and Fast p-ch MOSFET

The provided model is valid within the following geometry (Table 6.1), voltage (Table 6.2), and temperature (Table 6.3) ranges. Device behavior beyond these ranges may not be well described by the model and is not guaranteed.

1.0 V n-ch MOSFET	**1.0 V p-ch MOSFET**
$0.08\mu m \leq L_{DES} \leq 50\mu m$	$0.08\mu m \leq L_{DES} \leq 50\mu m$
$0.12\mu m \leq W_{DES} \leq 100\mu m$ (adopt multi-finger structures if $W_{DES} > 10\mu m$)	$0.12\mu m \leq W_{DES} \leq 100\mu m$ (adopt multi-finger structures if $W_{DES} > 10\mu m$)
$0.24\mu m \leq SA, SB$	$0.24\mu m \leq SA, SB$
$0.28\mu m \leq SD(NF > 1)$	$0.28\mu m \leq SD(NF > 1)$

Note: $SAREF = SBREF = 1.76\mu m$ is the reference dimension of the STI-stress effect.

Table 6.1: Geometry range for UMC CMOS 90nm Process Model

1.0 V n-ch MOSFET	**1.0 V p-ch MOSFET**
$0V \leq V_{GS} \leq 1.0V(*1.1)$	$0V \geq V_{GS} \geq -1.0V(*1.1)$
$0V \leq V_{DS} \leq 1.0V(*1.1)$	$0V \geq V_{GDS} \geq -1.0V(*1.1)$
$1.0V \leq V_{BS} \leq 0V$	$1.0V \geq V_{BS} \geq 0V$

Table 6.2: Voltage range for UMC CMOS 90nm Process Model

1.0 V n-ch MOSFET	**1.0 V p-ch MOSFET**
$-55°C \sim +125°C$	$-55°C \sim +125°C$

Table 6.3: Temperature range for UMC CMOS 90nm Process Model

6.2.1 *Electrical parameters*

The most important electrical parameters for the characterization of this device are summarized in Table 6.4 for NMOS and in Table 6.5 for PMOS, respectively.

Linear and saturation threshold voltages are extracted at

$$I_D = 300nA \cdot W_{DES}/(L_{DES} - 0.01\mu m)$$

for NMOS and

$$I_D = -70nA \cdot W_{DES}/(L_{DES} - 0.01\mu m)$$

for PMOS. All the normalized currents are divided with W_{DES}.

All parameters are given for $T = 25°C$, $V_{BS} = 0V$ unless specified otherwise.

Parameter	Condition	Unit	Target	Measured Value	Extracted Value	Centered Value
Long channel device (W_{DES}/L_{DES}= 10 μm / 1 μm)						
V_{TLIN}	V_{DS} = 0.05 V	V	--	0.164	0.154	0.150
I_{ON}	V_{DS} =V_{GS} = 1 V	μA/μm	--	141.83	138.46	143.33
Wide and short channel device (W_{DES}/L_{DES}= 10 μm / 0.08 μm)						
V_{TLIN}	V_{DS} = 0.05 V	V	0.330	0.336	0.338	0.331
V_{TSAT}	V_{DS} = 1 V	V	0.240	0.239	0.242	0.240
Body effect V_T shift	V_{DS} =1V, V_{BS}=0~-1V	V	--	0.048	0.057	0.056
DIBL V_T shift	V_{DS}=0.05~1 V	V	0.090	0.097	0.096	0.091
I_{ON}	V_{DS}=V_{GS}=1 V	μA/μm	655.00	641.31	649.15	651.71
I_{OFF}	V_{DS}=1 V, V_{GS}=0 V	A/μm	5.0e-09	6.8e-09	5.4e-09	5.8e-09
Narrow and long channel device (W_{DES}/L_{DES}= 1 μm / 0.08 μm)						
V_{TLIN}	V_{DS} = 0.05 V	V	--	0.340	0.331	0.326
V_{TSAT}	V_{DS} = 1 V	V	--	0.247	0.238	0.237
Body effect V_T shift	V_{DS} =1V, V_{BS}=0~-1V	V	--	0.051	0.057	0.057
DIBL V_T shift	V_{DS}=0.05~1 V	V	--	0.093	0.093	0.089
I_{ON}	V_{DS}=V_{GS}=1 V	μA/μm	--	648.09	676.55	676.20
I_{OFF}	V_{DS}=1 V, V_{GS}=0 V	A/μm	--	4.1e-09	5.9e-09	6.3e-09
Narrow and short channel device (W_{DES}/L_{DES}= 0.12 μm / 0.08 μm)						
V_{TLIN}	V_{DS} = 0.05 V	V	0.270	0.271	0.275	0.269
V_{TSAT}	V_{DS}=1 V	V	0.200	0.202	0.205	0.200
I_{ON}	V_{DS}=V_{GS}=1 V	μA/μm	830.00	835.75	852.00	833.10
I_{OFF}	V_{DS}=1 V, V_{GS}=0 V	A/μm	6.7e-08	8.9e-09	1.7e-08	1.7e-08
Capacitance						
CJS	V_J = 0 V	F/m^2	1.00e-03	1.11e-03	1.11e-03	1.07e-03
CJSWS	V_J = 0 V	F/m	1.50e-10	1.98e-10	2.08e-10	1.26e-10
CJSWGS	V_J = 0 V	F/m	2.30e-10	3.29e-10	5.80e-10	2.31e-10
C_{OVL}	V_{GS} = -0.5 V	F/m	3.40e-10	3.67e-10	3.16e-10	3.16e-10

Table 6.4: Electrical parameters for NMOS UMC 90nm Process Model

Parameter	Condition	Unit	Target	Measured Value	Extracted Value	Centered Value
Long channel device (W_{DES}/L_{DES}= 10 µm / 1 µm)						
V_{TLIN}	V_{DS} = -0.05 V	V	--	-0.124	0.123	0.121
I_{ON}	V_{DS} =V_{GS} = -1 V	µA/µm	--	-39.25	-39.58	-40.67
Wide and short channel device (W_{DES}/L_{DES}= 10 µm / 0.08 µm)						
V_{TLIN}	V_{DS} = -0.05 V	V	-0.270	-0.273	-0.280	-0.273
V_{TSAT}	V_{DS} = -1 V	V	-0.177	-0.180	-0.177	-0.174
Body effect V_T shift	V_{DS} =-1V, V_{BS}=0~1V	V	--	-0.064	-0.065	-0.064
DIBL V_T shift	V_{DS}=-0.05~-1 V	V	-0.093	-0.093	-0.103	-0.099
I_{ON}	V_{DS}=V_{GS}=-1 V	µA/µm	-280.00	-265.15	-271.91	-279.63
I_{OFF}	V_{DS}=-1 V, V_{GS}=0 V	A/µm	-1.0e-08	-8.3e-09	-9.8e-09	-1.1e-08
Narrow and long channel device (W_{DES}/L_{DES}= 1 µm / 0.08 µm)						
V_{TLIN}	V_{DS} = -0.05 V	V	--	-0.286	0.287	0.274
V_{TSAT}	V_{DS} = -1 V	V	--	-0.207	0.200	0.186
Body effect V_T shift	V_{DS} =-1V, V_{BS}=0~1V	V	--	-0.079	0.078	0.079
DIBL V_T shift	V_{DS}=-0.05~-1 V	V	--	-0.079	0.087	0.088
I_{ON}	V_{DS}=V_{GS}=-1 V	µA/µm	--	-232.86	-250.96	-269.50
I_{OFF}	V_{DS}=-1 V, V_{GS}=0 V	A/µm	--	-3.2e-09	-4.8e-09	-7.2e-09
Narrow and short channel device (W_{DES}/L_{DES}= 0.12 µm / 0.08 µm)						
V_{TLIN}	V_{DS} = -0.05 V	V	-0.240	-0.249	0.262	0.241
V_{TSAT}	V_{DS}=-1 V	V	-0.190	-0.196	0.206	0.186
I_{ON}	V_{DS}=V_{GS}=-1 V	µA/µm	-310.00	-292.45	-289.32	-310.77
I_{OFF}	V_{DS}=-1 V, V_{GS}=0 V	A/µm	-5.0e-09	-2.9e-09	-3.7e-09	-6.4e-09
Capacitance						
CJS	V_J = 0 V	F/m^2	1.00e-03	1.27e-03	1.28e-03	1.26e-03
CJSWS	V_J = 0 V	F/m	1.10e-10	1.97e-10	2.03e-10	1.29e-10
CJSWGS	V_J = 0 V	F/m	3.20e-10	6.35e-10	2.82e-10	2.45e-10

Table 6.5: Electrical parameters for PMOS UMC 90nm Process Model

6.2.2 *Model fitting accuracy*

The model fitting accuracy of parameters I_D, G_M and G_{DS} (Table 6.11) is quantified by using the following equation:

$$error_{\%} = 100 \cdot \frac{|meaasurement - simulation|}{\frac{measurement+simulation}{2}} \tag{6.4}$$

Different error criteria are defined for different parameters (I_D, G_M and G_{DS}) as well as for different operation region. These criteria are explained below.

A. I_D accuracy criteria

   a) I_D in sub-threshold region, i.e. $V_{GS} < V_T$;

   b) I_D in moderate inversion region, that is $V_T \leq V_{GS} < V_T + V_X$;

   c) I_D in strong inversion region, that is $V_T + V_X \leq V_{GS} \leq V_{CC}$; where $V_X = (V_{CC} - V_T)/2$

B. G_M and G_{DS} accuracy criteria

   a) G_M and G_{DS} in sub-threshold region, that is $V_{GS} < V_T$

   b) G_M and G_{DS} in inversion region, that is $V_T \leq V_{GS} \leq V_{CC}$

In-depth analysis about Worst Case model and other parameters trends can be retrieved in [46].

Target of Quality Check	$V_{GS} < V_T$ (sub-threshold)	$V_T \leq V_{GS} < V_T+V_X$ (moderate inversion)	$V_T+V_X \leq V_{GS} \leq V_{CC}$ (strong inversion)
I_D	$\leq 75\%$	$\leq 25\%$	$\leq 7\%$
G_M	$\leq 50\%$	$\leq 20\%$	$\leq 20\%$
G_{DS}	$\leq 100\%$	$\leq 50\%$	$\leq 50\%$

Table 6.11: Requirement for model fit accuracy

7 OPTIMIZATION FOR LINEARITY IMPROVEMENT

The main goal of this study is to further reduce the power consumption of the presented 6-bit TC-ADC circuit in [11] and estimate how the offset, gain and threshold calibrating networks are affected by such power optimization. Although random offset errors, yield and noise sensitivity of regenerative comparators have been studied [37, 39], the power optimization has not been considered at all. Therefore, traditional transistor sizing techniques are applied to optimize the power consumption. Additionally, it is analyzed the offset, gain and TC calibrating networks requeriments by means of Monte Carlo simulations.

7.1 ANALYSIS METHODOLOGY

The main power consumption in TC-ADC is generated by the threshold configuring comparator, so optimization is focused, in this section, on the circuit. The procedure for power optimization and mismatch analysis is sum up in the next items:

A. In order to guide the optimization, circuit operation is analyzed by looking at power, delay and full scale contributions of each device.

B. Circuit parameters that best match the available specifications in [11] are obtained. A reference cell comparator using such circuit parameters is considered for comparison purposes.

C. The circuit operation know-how is used to optimize the threshold configuring comparator in terms of power consumption, looking for no delay degradation while maintaining the same FS operation range (obtained through the simulation procedure described in Section 4.1).

D. Offset, gain and non-linearity errors are obtained by simulation of both reference and optimized regenerative comparators under perfect device matching condition.

E. Finally offset, gain and non-linearity errors are analyzed by means of Monte Carlo simulations, using mismatch models for the UMC CMOS 90nm "Standard Process 1.0V CMOS 1P9M" technology.

As shown in Section 4.1, to conduct the research about the maximum power consumption for each digital code generation (that is in correspondence of the threshold voltage V_{th}) by varying the sizes of the

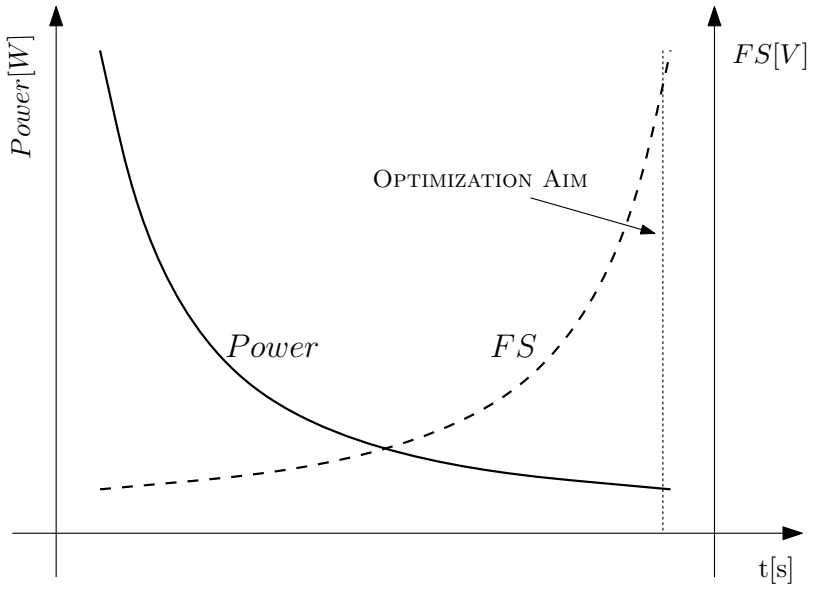


Figure 7.1: *Power consumption* and *FS* trends over time for a TC-Comparator

comparator transistors through the *scanning optimization* method, resulted unfeasible. For this reason, another optimization mode, based on the Bisection methodology (see Section A.5) – whose script is contained into the `oheader.sp` file (Section A.7.2) – was adopted.

The optimization aim can be described through the following condition:

$$min(P),\ max(FS) \equiv \begin{cases} \forall P, & max(FS) \\ \forall FS & min(P) \end{cases} \tag{7.1}$$

By observing the results presented in Table 4.1, it is clear that, once fixed the size for M_1 and M_2, the power consumption and FS trends for M_3 and M_4 sizing present a trade-off (Figure 7.1). The issue is formally described by the condition 7.1, that is obtaining the lowest overall power consumption and the largest FS range for a given size.

After the initial analysis about *size* vs *power consumption* vs *FS* described in Section 4.1.1, where trends for each transistor were obtained by keeping the size of the other ones fixed, the following investigation was about the effects in terms of mutual influence by modifying more than one transistor size at the same time. The identical behaviour noticed for M_{1-2} and M_{3-4} suggested to set a multiplying factor between sizes in order to reduce the number of possible combinations:

$$\begin{cases} M_2 = & m_{12} \cdot M_1 \\ M_4 = & m_{34} \cdot M_3 \end{cases} \tag{7.2}$$

In this way, given the M_1 size (dependent on the maximum current flowing out of V_{DD}) and the m_{34} size factor (fixed by the technology[1]), the search was limited to finding m_{12} in order to minimize the power consumption and m_{34} to maximize the FS.

7.1.1 *Optimization analysis by Bisection method*

Given a fixed digital code, the threshold voltage V_{th} is the input voltage which leads the converter to keep the output at the middle value, in a metastable equilibrium. So, to measure it, it is not possible to directly detect the exact value, while the Bisection approach can be used in the following way.

As shown in Figure 7.2 , it is always possible to find a time region

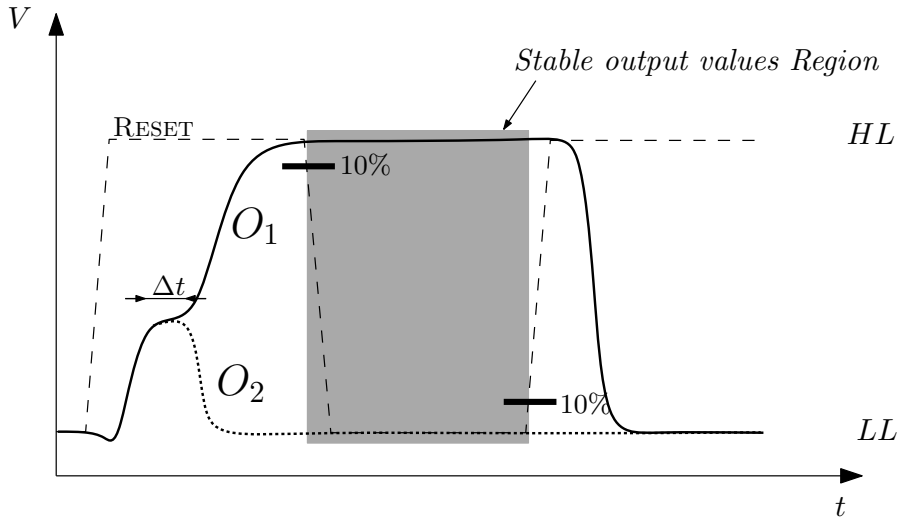


Figure 7.2: Stable region for Output values detection

where the two output signals O_1 and O_2 are stable, one at the High and the other one at the Low level. In this situation, it is possible to determine the correspondent v_{id} region, whose edges are

$$\begin{aligned} \exists v_{id}^{+}: \quad & v_{O_1} - v_{O_2} > 0 \\ \exists v_{id}^{-}: \quad & v_{O_1} - v_{O_2} < 0 \end{aligned} \tag{7.3}$$

$$\text{with } v_{id}^{-} < V_{th} < v_{id}^{+}$$

Through iterative application of Bisection method, this region becomes narrower, with v_{id}^{-} and v_{id}^{+} values increasingly closer to the searched V_{th}.

1 In particular, M_3 is a p-MOS transistor, while M_4 is a n-MOS transistor. Because of the known different performance of the two MOSFET types, an exact multiplying factor ($3 \leq m_{34} \leq 4$) can be calculated in order to balance the effect.

Following this procedure, a plot useful for calculating INL and DNL errors for each digital code can be quickly obtained and with the required precision.

7.2 DESIGN RESTRICTIONS ACHIEVEMENT

From the analysis of circuit operation, it can be concluded that sizing of each device has a relative effect on circuit performance as shown in Table 7.1 . First, by using data available at the reference design specification ($\triangle C_{LSB} = 0.5fF$, $C = 40fF$ and $W/L = 0.12\mu m/0.32\mu m$ for the TC unit device) it was obtained that just one TC unit device is used for the LSB of TC networks. Initially, M_3-M_6 transistors on the reference design are sized to obtain an inverter threshold near $V_{DD}/2$, and S_1-S_4 to minimum dimensions. Then, M_1 and M_2 devices are sized to obtain the desired FS for a larger enough *Mclk* device. Last, *Mclk* is sized to obtain the desired delay and power specification, while allowing some minor variation in the S_1-S_4 devices to get a fine matching. This will shift the obtained FS again, so the procedure is iterated by resizing the devices again until very small shifts are generated for the M_1-M_2 sizes. Finally, a last iteration is done to take into account the technology pitch.

Note that, if the reference specification does not correspond to an optimum power consumption point, there will be a non-unique solution which satisfies the specifation constrains. To avoid high dissipation in the latch transistors, the minimum M_3-M_6 area solution is selected, although other solutions could lower the offset and nonlinearity errors due to device mismatch. Also, note that the higher the FS condition is, the larger the *Mclk* transistors will be required in order to allow for higher current flow through the differential pair M_1-M_2, or the larger M_1 and M_2 transistors will be required to cause higher current imbalance for a fixed *Mclk* size.

Next, power optimization is accomplished by repeating a similar procedure as described before for power, delay and FS matching. But

Device	Power Consumption	Delay	Full Scale Range
Mclk	high	high	high
M_1-M_2	low	low	high
M_3-M_6	low	high	low
S_1-S_4	low	low	low
C_{TCA}-C_{TCB}	low*	low*	high

*assuming small devices for threshold configuring networks.

Table 7.1: Impact of device sizing in the performance of the threshold configuring comparator

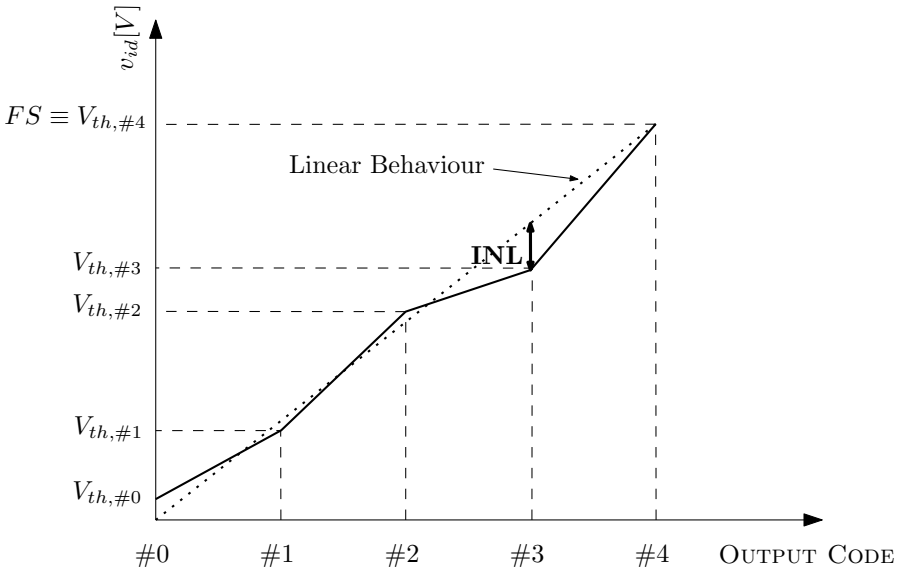
in this case power specification is reduced to its minimum value. Minimum power consumption can be determined by initially making M_1-M_2 devices large enough, and reducing the size of *Mclk* till delay and FS conditions are met. To avoid highly non-linear behavior, *Mclk* cannot be too much reduced in comparison with M_1 and M_2. On the other hand, as lower the M_3/M_5 (or M_4/M_6) ratio is, the smaller will be the required *Mclk* devices. Thus, optimum power sizing tends to reduce M_3/M_5 and M_4/M_6 ratios and slighlty increase the $M_1/Mclk$ and $M_2/Mclk$ ones.

The sizing of TC units in the reference design allows for increased C_{on}/C_{off} ratio (approximately 6 times) when compared with minimum size devices. On the one hand, high capacitive deltas in TC networks relax the sizing condition of *Mclk* to fit the FS specification and could reduce the power consumption. But on the other hand, it is preferrable to keep devices close to the minimum size in order to reduce the overall capacitive load, except for the case that reduced device mismatch is required. The obtained device sizing for TC units is considered a good compromise between C_{on}/C_{off} ratio, mismatch and impact on power consumption.

Both, in the reference design and in the optimized one, we assumed that the matching of power and delay constrains are met for input differential voltages at $\pm 10\mu V$ from the zero threshold condition ($TC_A = TC_B = 0$). A common mode input voltage of $0.3V$ is also assumed in the IN_A-IN_B terminals, because this is an optimum design point for the speed and yield [37] performance of latch-type voltage sense amplifiers.

7.3 NONLINEARITY ANALYSIS: DNL AND INL ERRORS

Once the threshold voltage $V_{th,\#n}$ has been individuated for each $\#n$ digital code (through Scanning optimization mode, as described in Section 4.1), it is possible to plot the real behavior of the TC-Comparator (Figure 7.3)



Note that ideal linear behaviour of comparator is traced by drawing a straight line through zero and FS.

Figure 7.3: Qualitative plot of non-linear relationship between Input voltage and Output digital code for a TC-Comparator

and compare it with the ideal linear working function. From this comparation it is obtained the INL error, that is the deviation of the output code from the linear trend (as defined in Section 1.2.2).

The same graph is also useful for obtaining and evaluating the DNL error of each digital code, calculated through Equation 1.3.

8 MISMATCH ANALYSIS

Regenerative comparators are very sensitive to device mismatch and noise. So adequate layout techniques must be used in order to reduce mismatch and allow for high symmetry in the designs [42, 12]. If small devices are assumed in threshold configuring and calibrating networks, capacitive load and minimum power consumption are highly related with the routing parasitic capacitances. Thus, to reduce routing complexity associated with offset, gain and TC calibrating networks will be a key factor in order to further reduce the overall power consumption in TC comparators and ADC.

The goal of Monte Carlo analysis is to obtain the design restrictions that must be met by calibrating networks (offset, gain and threshold configuring calibration) and the way they are affected when optimized power consumption cells are used. Additionally, conclusions of this analysis can guide the development of future TC-ADC architecture improvements for ultra-low power applications.

Unfortunately, the noise analysis is not a kind of error that can be compensated by the calibrating networks. Expected non-linearity values caused by the noise component must be inside the 0.2LSBs range for 90nm CMOS technologies [43].

Note that device mismatch around the typical process corner, which accounts for both, static and dynamic offset errors caused by all the devices in the circuit, is used in the simulations.

8.1 MISMATCH OPTIMIZATION THROUGH DNL ANALYSIS

After having implemented an effective algorithm (based on scanning and bisection optimization modes) to detect the INL error for each digital code, it was used time after time through Monte Carlo simulations, which permit to analyze the effects of process scattering over the observed parameters.

By plotting all the results on the same graph, it is easy to spot the biggest INL (absolute) value, which has to be smaller than $\frac{1}{2}$LSB (as specified in Equation 1.2):

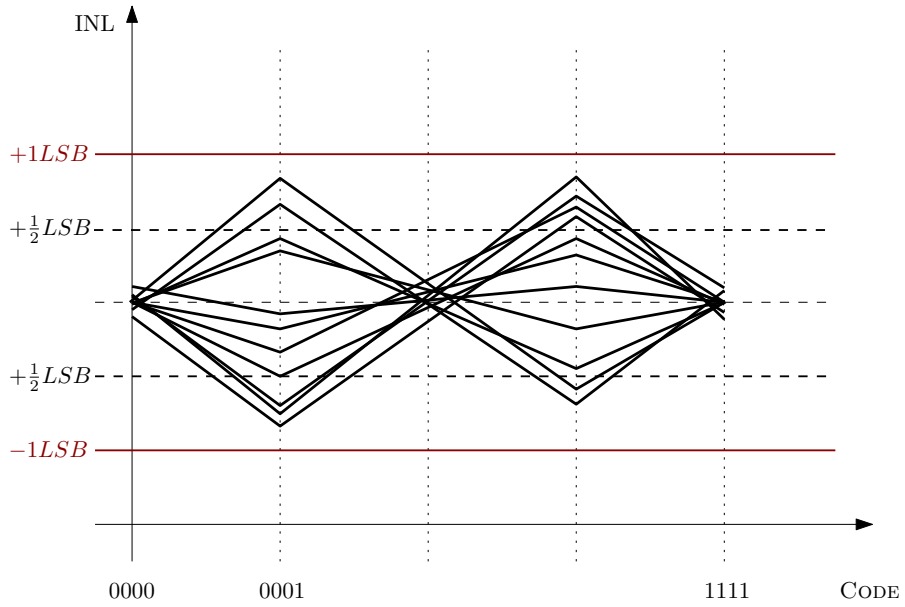


Figure 8.1: Qualitative plot of INL errors for each digital code generated through Monte Carlo simulations

9 RESULTS

All the circuit parameters and performance obtained are presented in Table 9.1 and Table 9.2 for the reference design (REF) and the power optimized one (OPT) respectively. Circuit parameters are shown in Table 9.1, in terms of transistor size, while circuit performance is shown in Table 9.2 in terms of delay, power and FS range, and the maximum absolute offset, gain and non-linearity (DNL, INL) errors for all the possible threshold configuring TC_A and TC_B words. As commented before, power and delay measurements are referred to a $\pm 10\mu V$ input signal for the $TC_A = TC_B = 0$ codes. A resolution step of $100\mu V$ (about 0.02 LSBs) is used for the voltage sweep and a common mode input voltage of $0.3V$ is assumed at the IN_A-IN_B terminals. The best-fit of the trip points to a straight line obtained by the least squares method is assumed for linearity parameters (offset, gain, DNL and INL). The sampling period is fixed at $40ns$, in order to avoid non-linear power behavior near the maximum sampling frequency ($50MSps$).

It can be observed that the optimized design reduces the power consumption and delay of more than 77% and 20% respectively, for a full scale range of $\pm 160mV$. Moreover, our power optimized design exhibits near 50% improvement for the linearity error parameters. The higher energy efficiency of the optimized solution delivers energy to the capacitor arrays in a more efficient way than the reference solution did. As lower energy is dissipated by the internal devices in comparison with the quantity distributed to capacitor arrays, lower non-linear effects are obtained. Non-linearity errors for each TC code are shown in Figure 9.1. Note that, due to perfect simmetry in the design, the INL error for $TC_A = TC_B = 0$ (code number 31) is null.

Finally, mismatch analysis results are presented in Table 9.3 . It can be observed that the maximum absolute offset errors of both designs are close to each other. But the optimized power consumption version achieves more than 30% gain and DNL errors reduction. The lower gain error can be directly used for decreasing the corresponding calibrating network size (and complexity) and to further reduce the power consumption. On the other hand the INL error is about 21% worse for the power optimized design. The INL error increase is due to the higher relative variations of the node currents in presence of a mismatch condition when reducing the device size.

Figure 9.2 plots the INL errors corresponding to each code and Monte Carlo run. 3D plots are shown in Figure 9.2a and Figure 9.2b. Overlayed 2D plots for each run are shown in Figure 9.2c and Fig-

Device	Size (REF)	Size (OPT)
M_{clk}	$41.13\mu m \times 0.08\mu m$	$2.69\mu m \times 0.08\mu m$
M_1, M_2	$2.79\mu m \times 0.08\mu m$	$4.18\mu m \times 0.08\mu m$
M_3, M_6	$0.13\mu m \times 0.08\mu m$	$0.40\mu m \times 0.08\mu m$
S_1, S_4	$1.22\mu m \times 0.08\mu m$	$1.70\mu m \times 0.08\mu m$
C_{TC_A}, $C_{TC_B}(LSB)$	$0.12\mu m \times 0.32\mu m$	$0.12\mu m \times 0.32\mu m$

Table 9.1: Device sizing parameters for reference (REF) and optimized (OPT) comparators

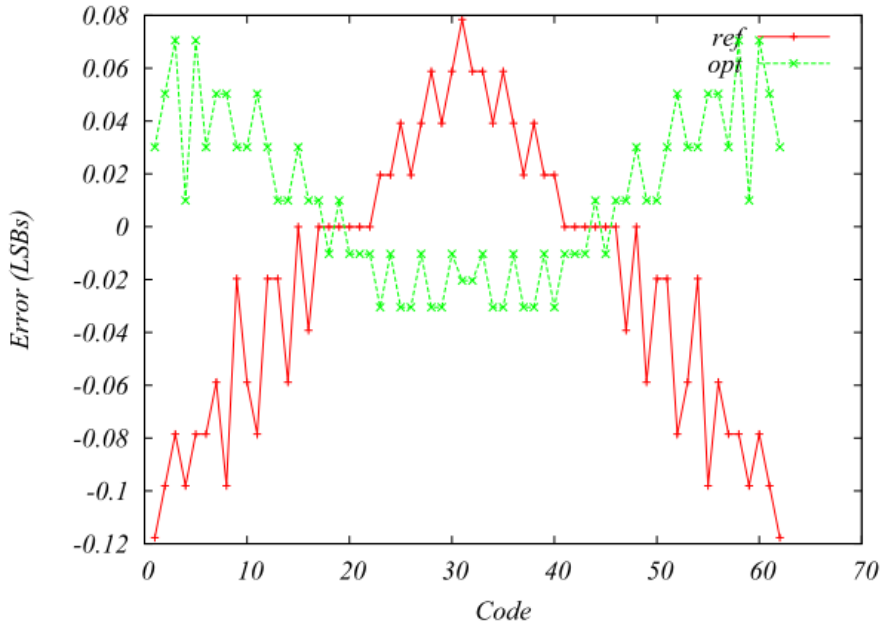
Device	REF	OPT
Power$[\mu W]$	13.00	2.92
Delay$[ns]$	1.65	1.31
FS$[mV]$	$\pm$159.95	$\pm$160.10
Offset$[mV]$	-3.11	-1.5
Gain Error[%]	$+2.00$	-0.98
DNL$[LSBs]$	$\pm$0.118	$\pm$0.071
INL$[LSBs]$	$\pm$0.639	$\pm$0.338

Table 9.2: Obtained performance for reference (REF) and optimized (OPT) comparators, without device mismatch

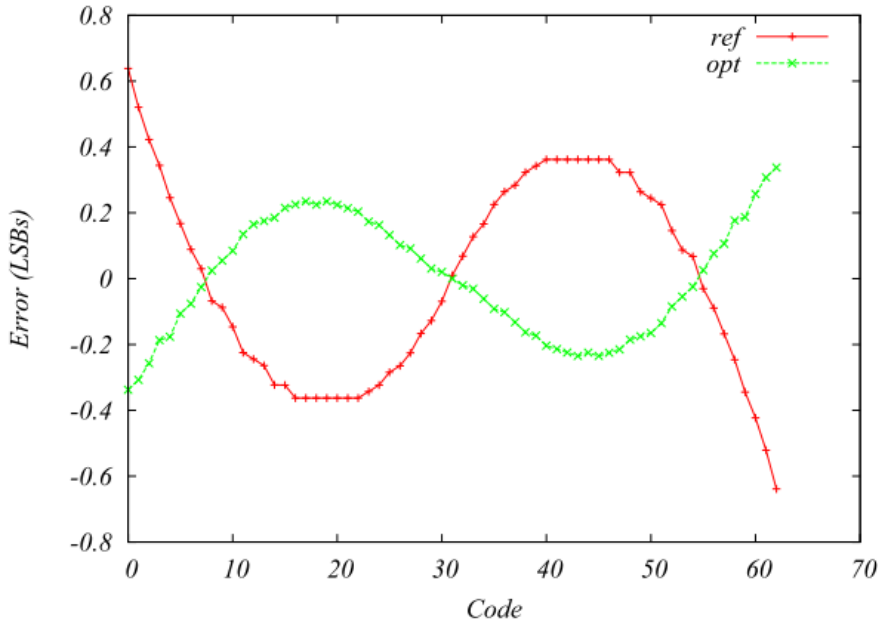
Device	REF	OPT
Offset$[mV]$	$\pm$93.60	$\pm$91.25
Gain Error[%]	$\pm$8.54	$\pm$5.40
DNL$[LSBs]$	$\pm$0.124	$\pm$0.085
INL$[LSBs]$	$\pm$0.956	$\pm$1.163

Table 9.3: Obtained performance for reference (REF) and optimized (OPT) comparators, with device mismatch

ure 9.2d. The offset and gain erros are depicted in Figure 9.3. Although the INL error is about 0.2 LSBs worse for the power optimized design it can be more easily modeled than the reference ones. This could be used to simplify the required calibration parameters in the case, for example, of a memory-based cancellation scheme.

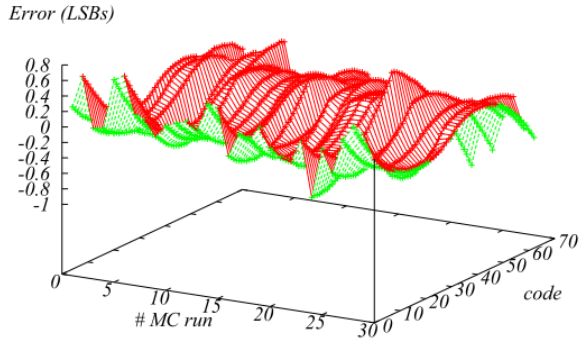


(a) DNL

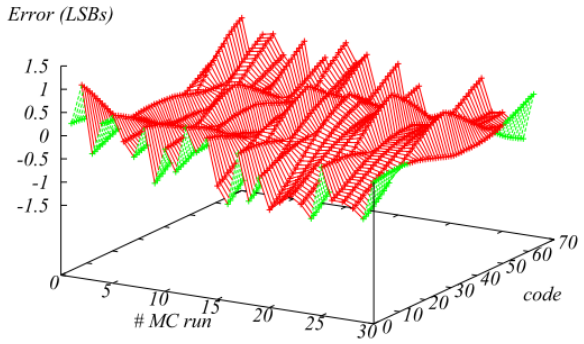


(b) INL

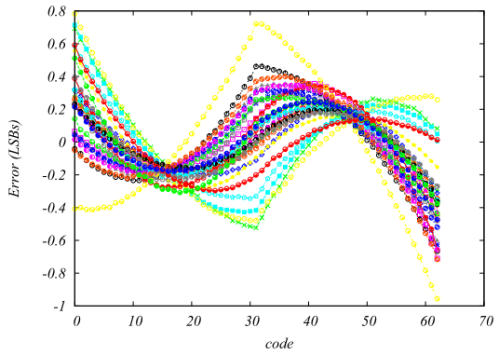
Figure 9.1: Non-linearity plots



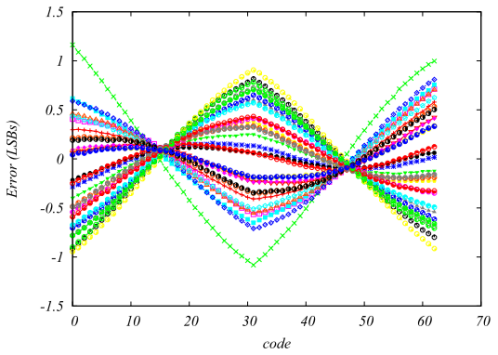
(a)



(b)

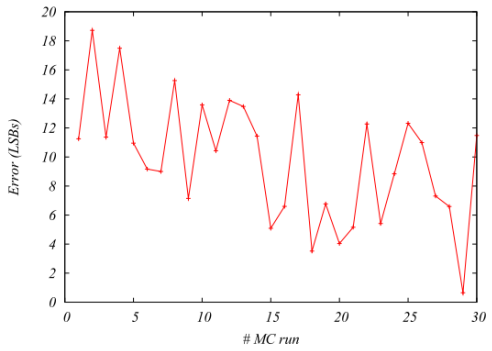


(c)

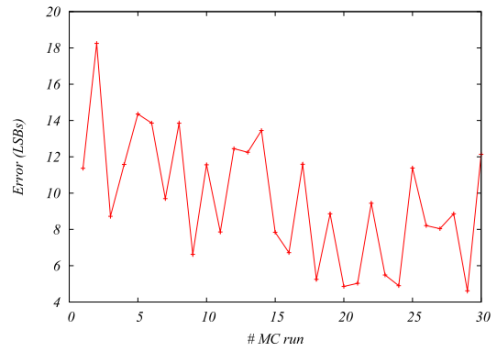


(d)

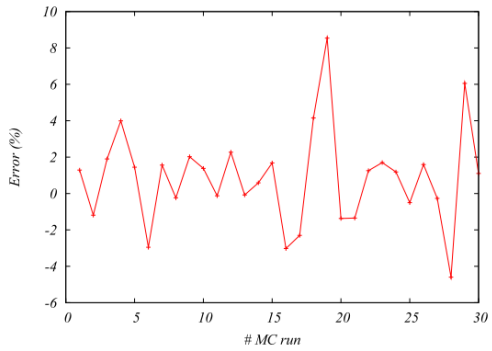
Figure 9.2: INL plots using mismatched devices: (a) and (b) are 3D-plots, and (c) and (d) are the corresponding 2D-plots, for the reference and power optimized comparators



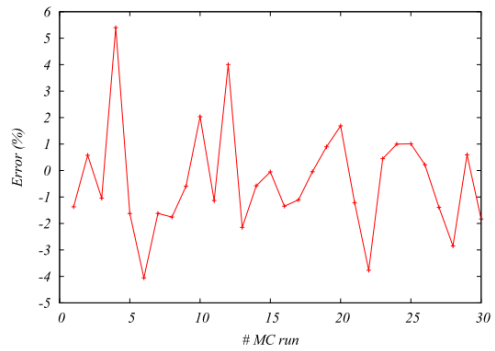
(a)



(b)



(c)



(d)

Figure 9.3: INL plots using mismatched devices: (a) and (b) are 3D-plots, and (c) and (d) are the corresponding 2D-plots, for the reference and power optimized comparators

9.1 OPTIMIZATION RESULTS – *scanning* METHOD

Plots presented in this section are generated by GnuPlot from the *scanning mode* simulation data.

Each device size is set to the optimal value (see Table 9.1).

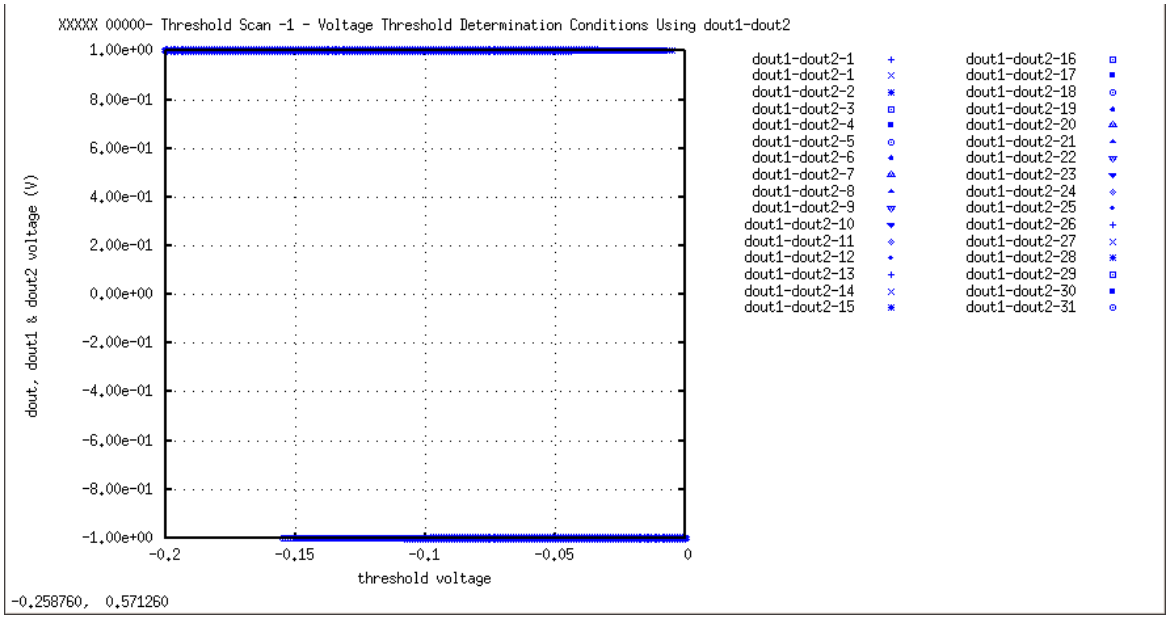


Figure 9.4: V_{th} Determination Conditions 1 - *scanning mode, opt.val.*

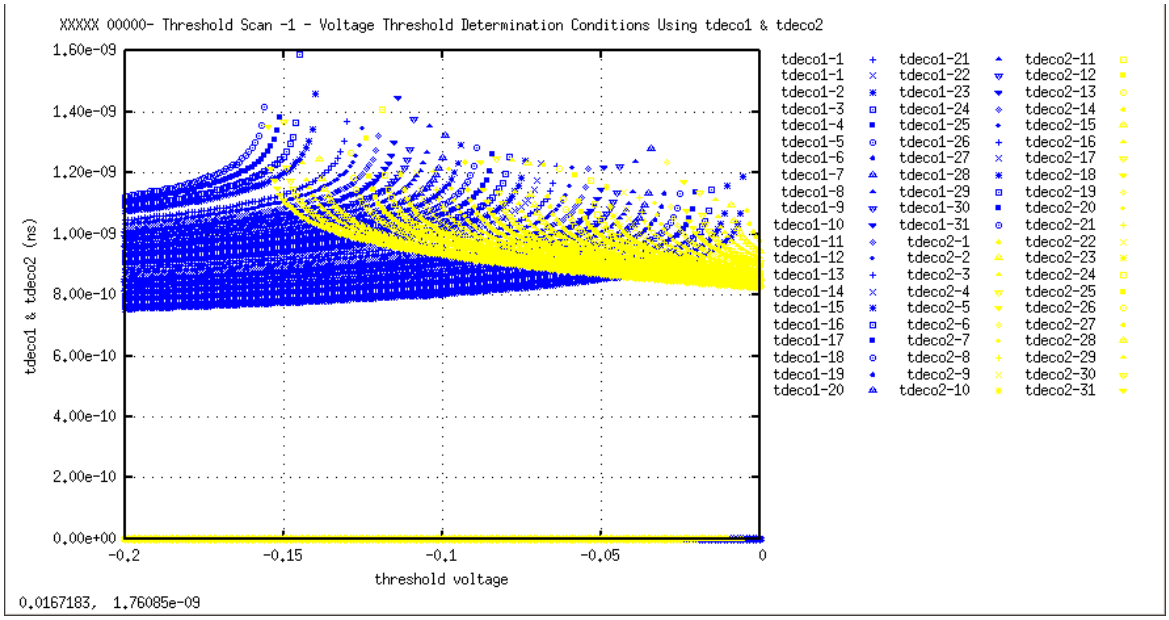


Figure 9.5: V_{th} Determination Conditions 2 - *scanning mode, opt.val.*

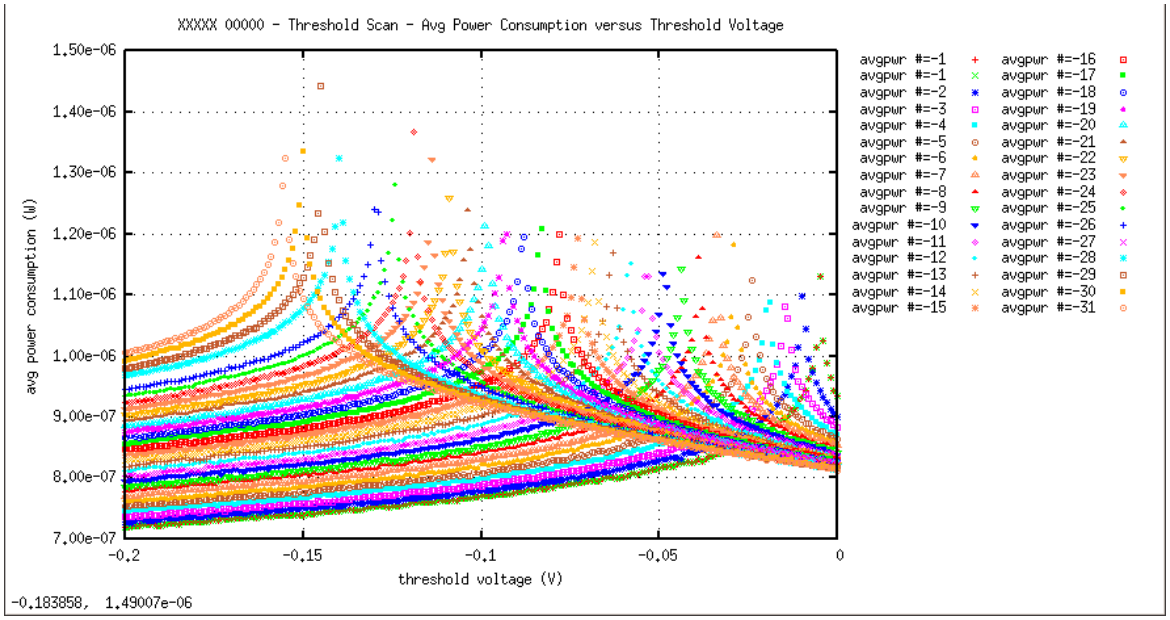


Figure 9.6: Average Power Consumption *vs* V_{th} *- scanning mode, opt.val.*

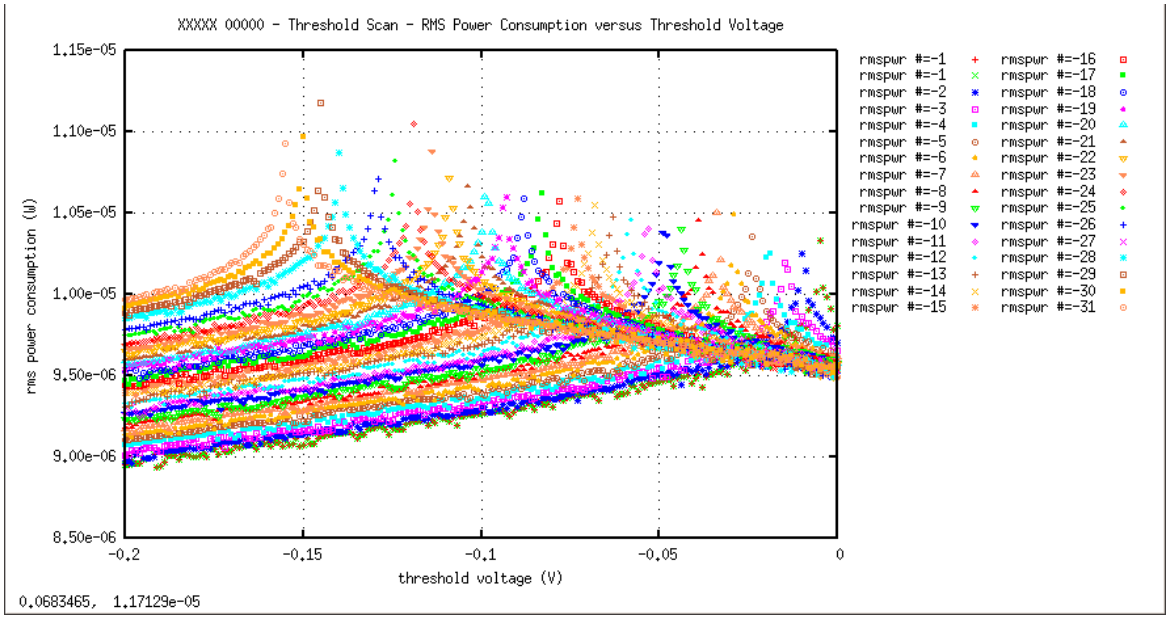


Figure 9.7: RMS Power Consumption *vs* V_{th} *- scanning mode, opt.val.*

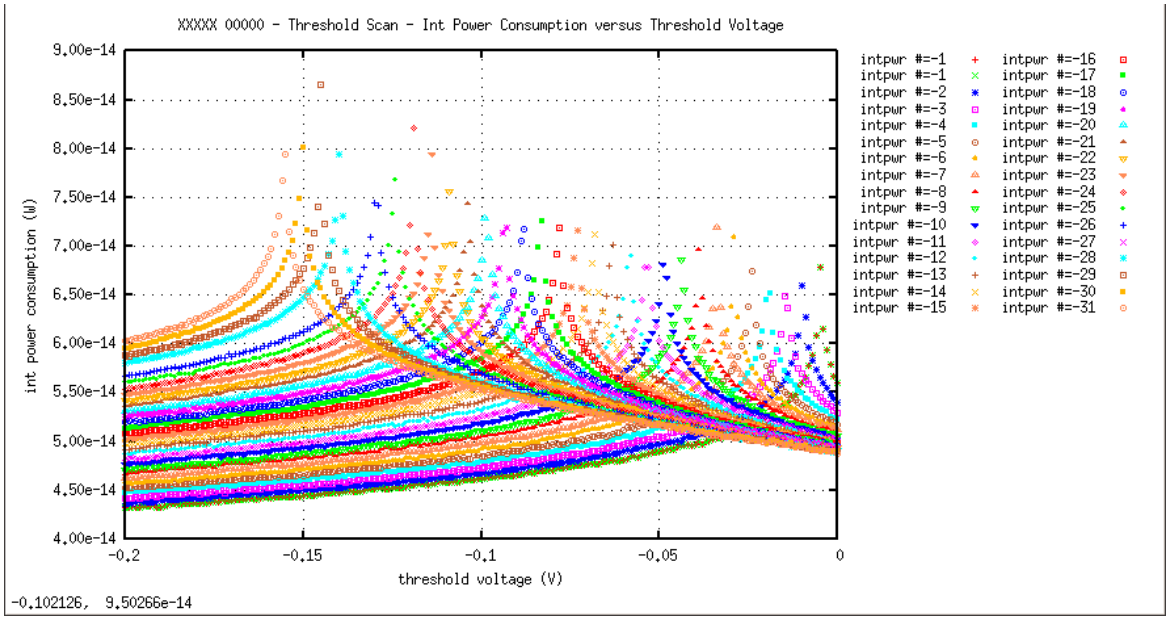


Figure 9.8: Int Power Consumption *vs* V_{th} - *scanning mode, opt.val.*

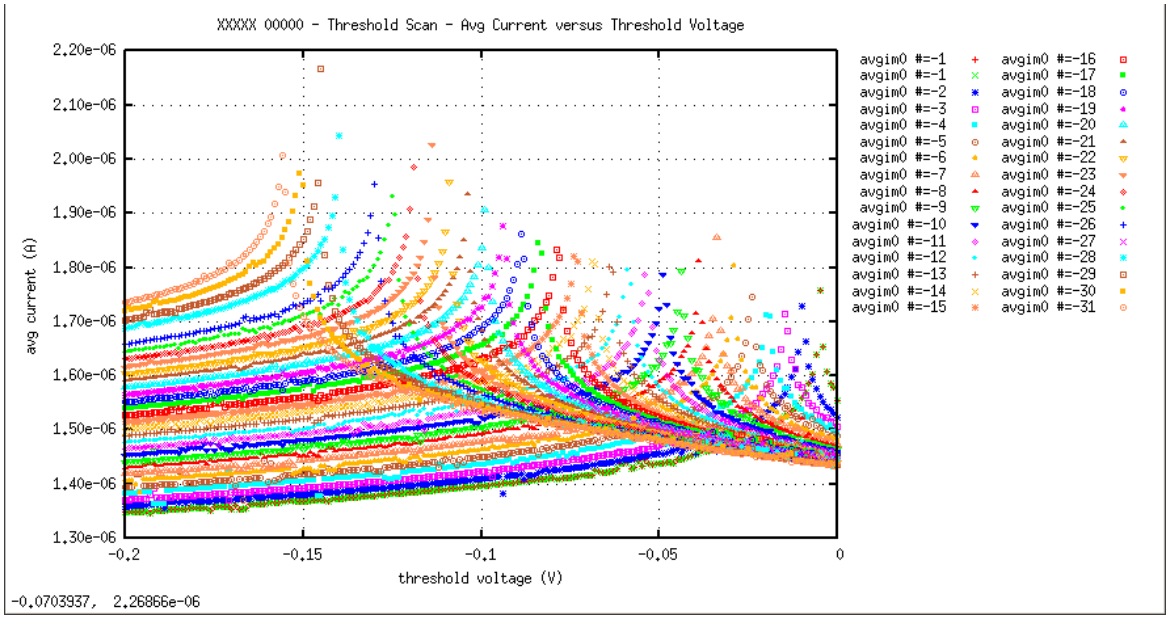


Figure 9.9: Average Current *vs* V_{th} 1 - *scanning mode, opt.val.*

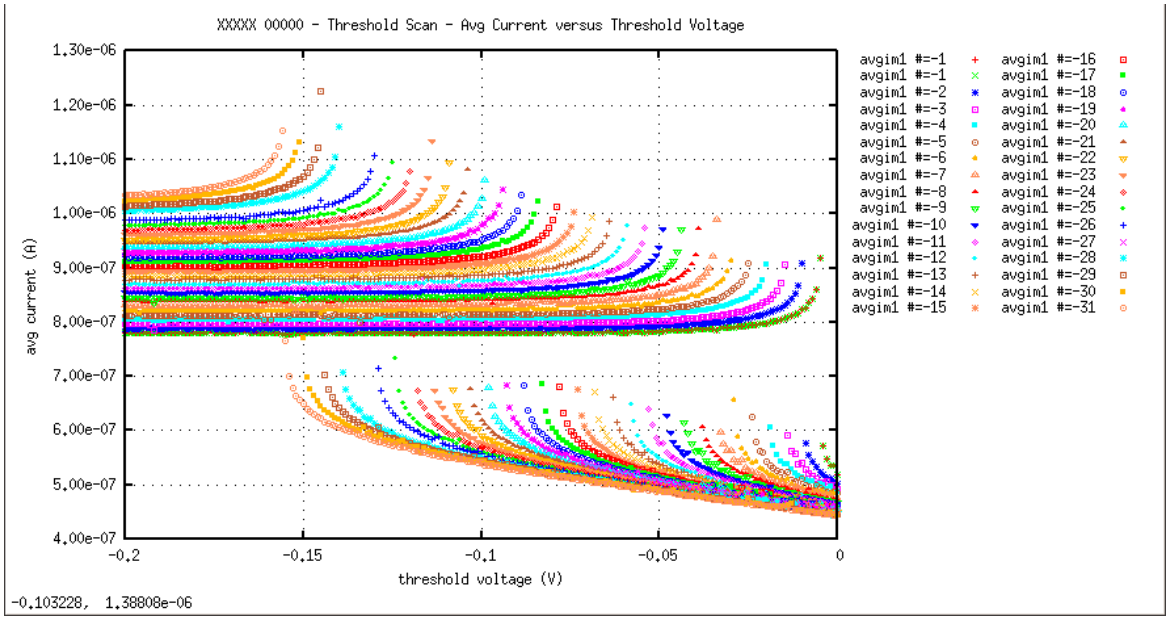


Figure 9.10: Average Current *vs* V_{th} 2 - *scanning mode, opt.val.*

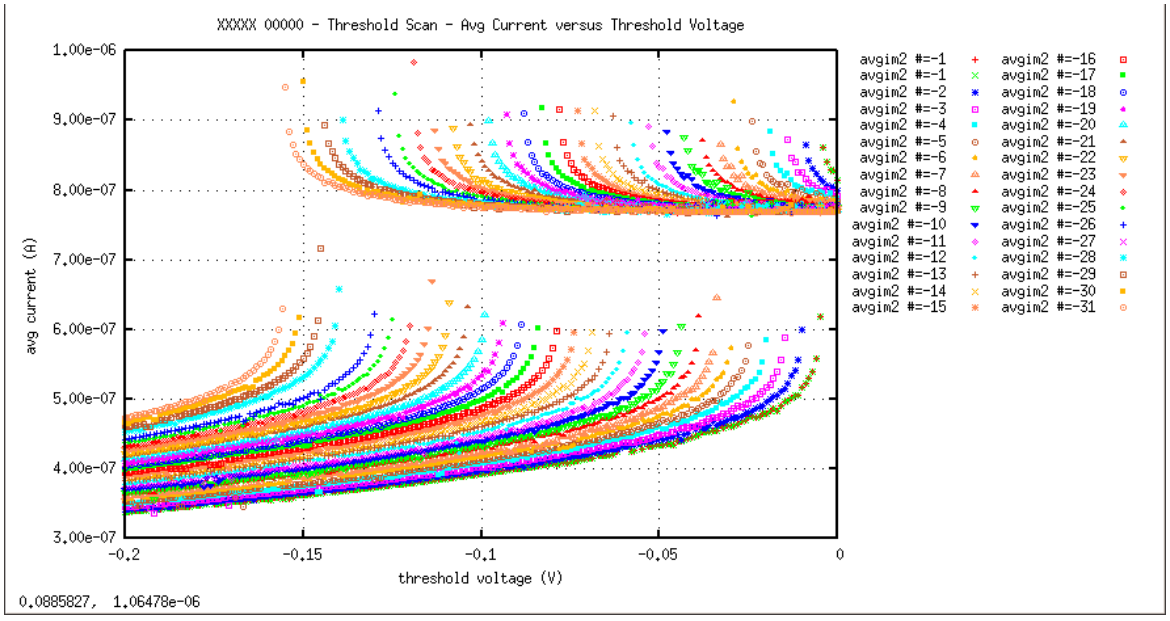


Figure 9.11: Average Current *vs* V_{th} 3 - *scanning mode, opt.val.*

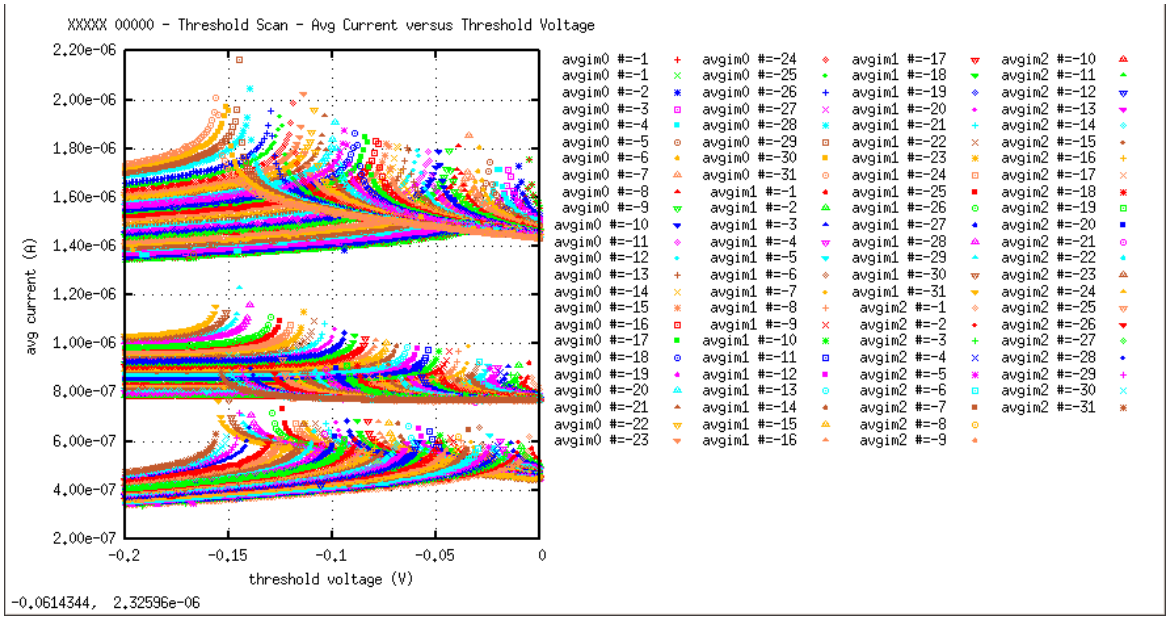


Figure 9.12: Average Current *vs* V_{th} 4 - *scanning mode, opt.val.*

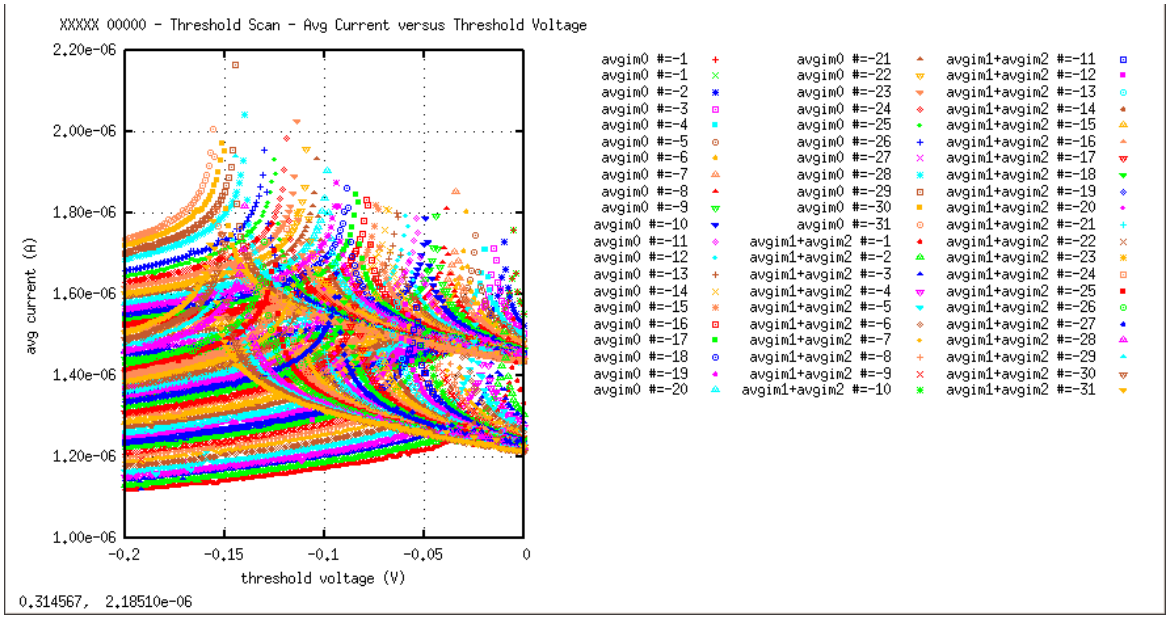


Figure 9.13: Average Current *vs* V_{th} 5 - *scanning mode, opt.val.*

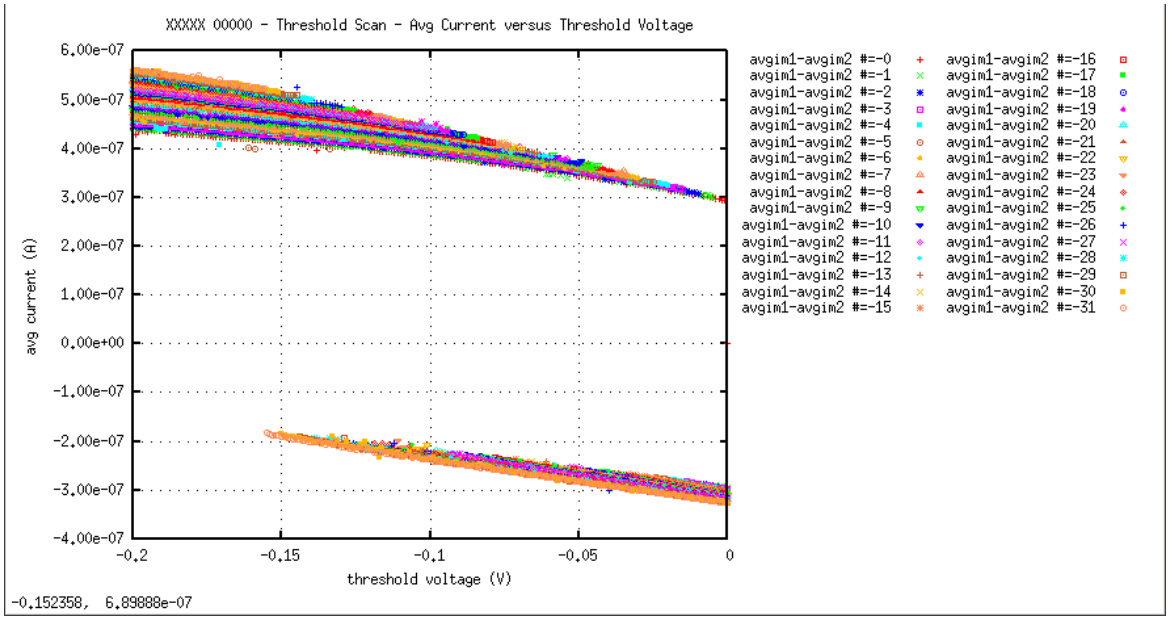


Figure 9.14: Average Current *vs* V_{th} 6 - *scanning mode, opt.val.*

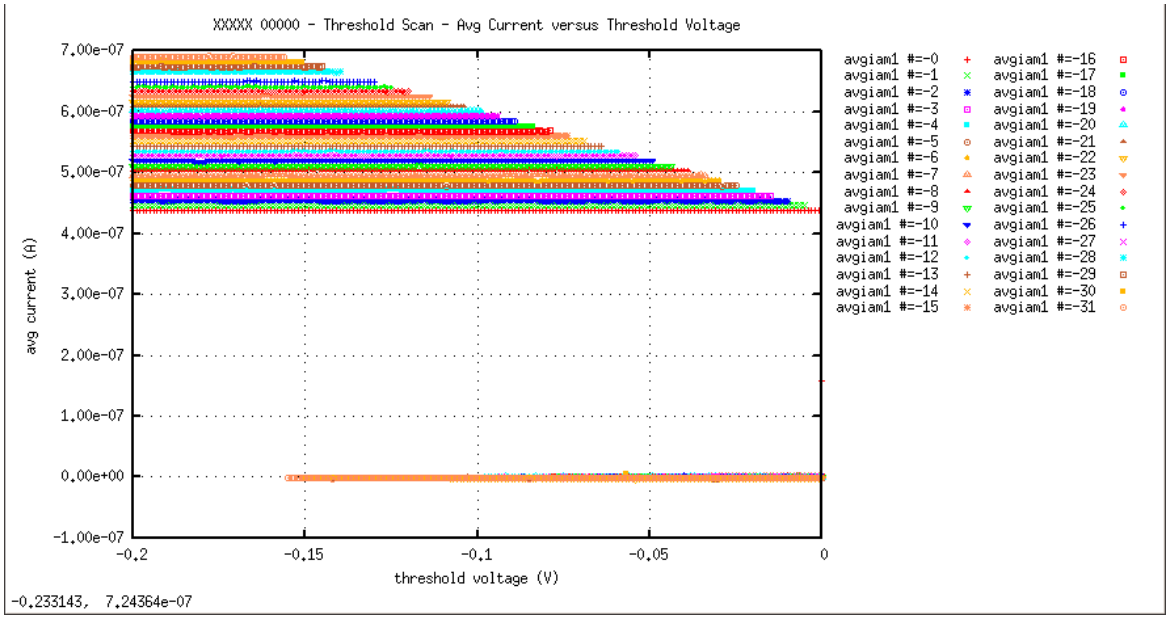


Figure 9.15: Average Current *vs* V_{th} 7 - *scanning mode, opt.val.*

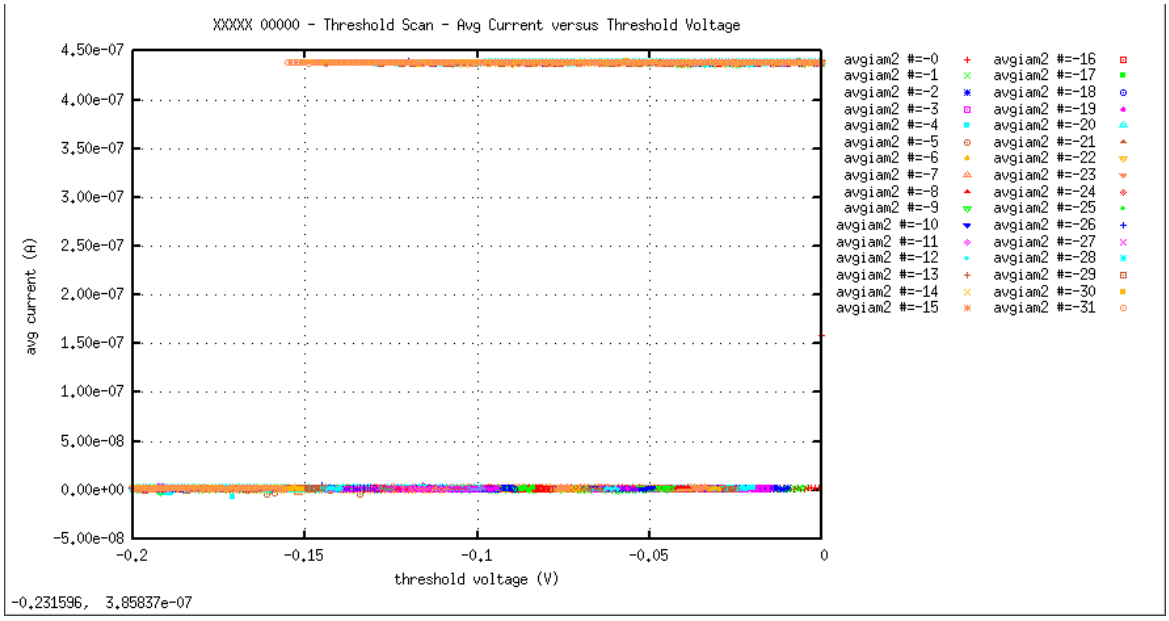


Figure 9.16: Average Current *vs* V_{th} 8 - *scanning mode, opt.val.*

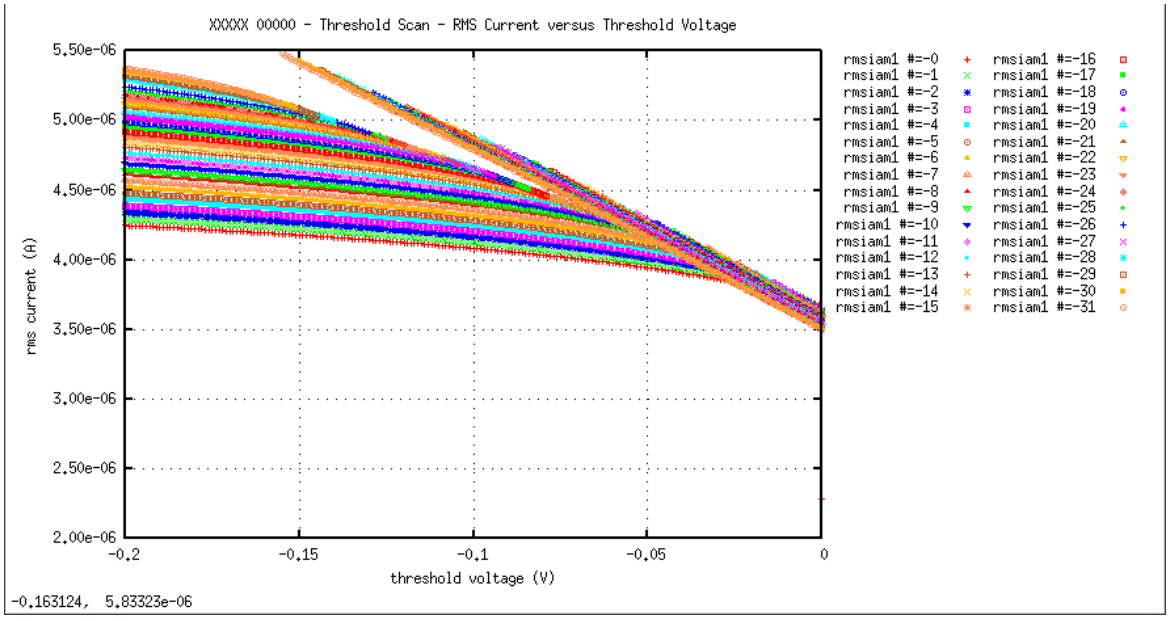


Figure 9.17: RMS Current *vs* V_{th} 1 - *scanning mode, opt.val.*

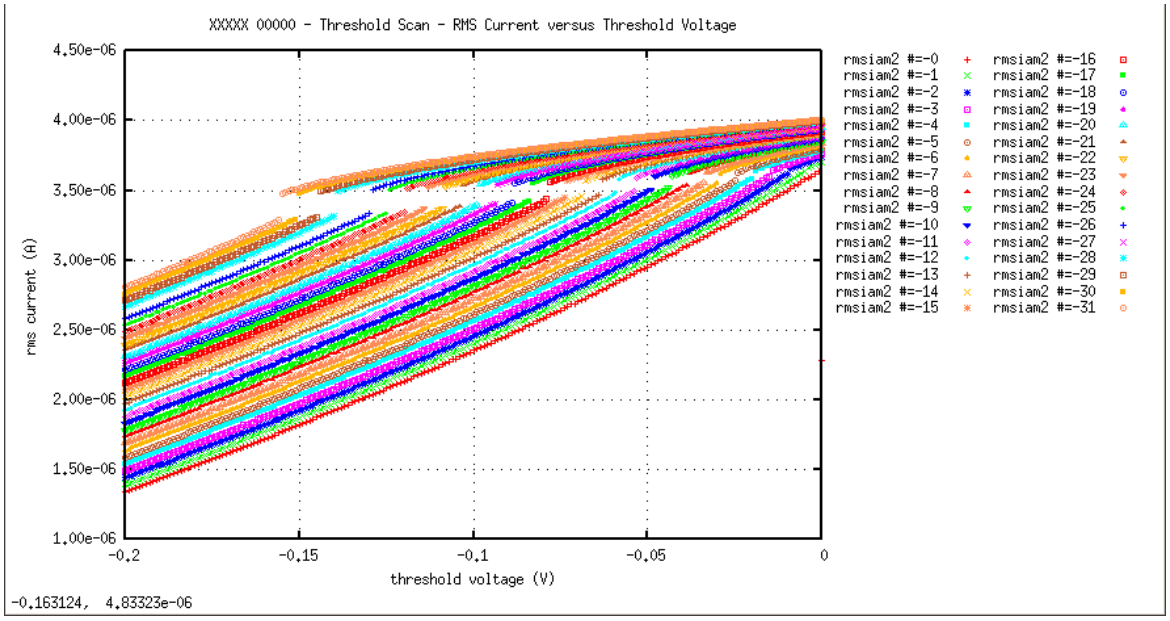


Figure 9.18: RMS Current *vs* V_{th} 2 - *scanning mode, opt.val.*

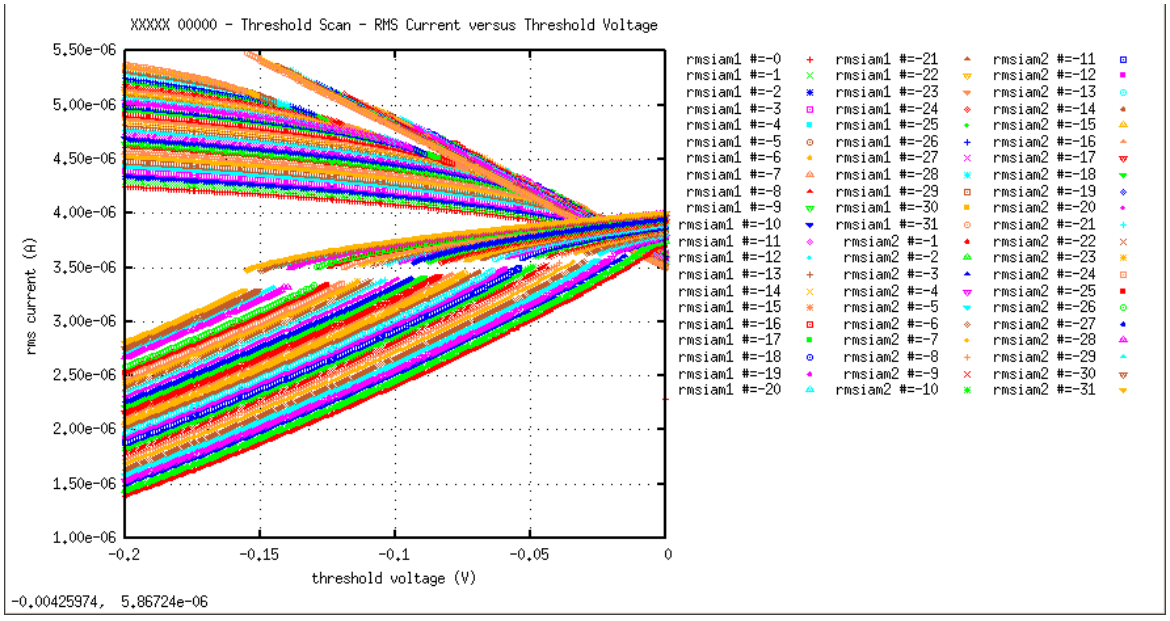


Figure 9.19: RMS Current *vs* V_{th} 3 - *scanning mode, opt.val.*

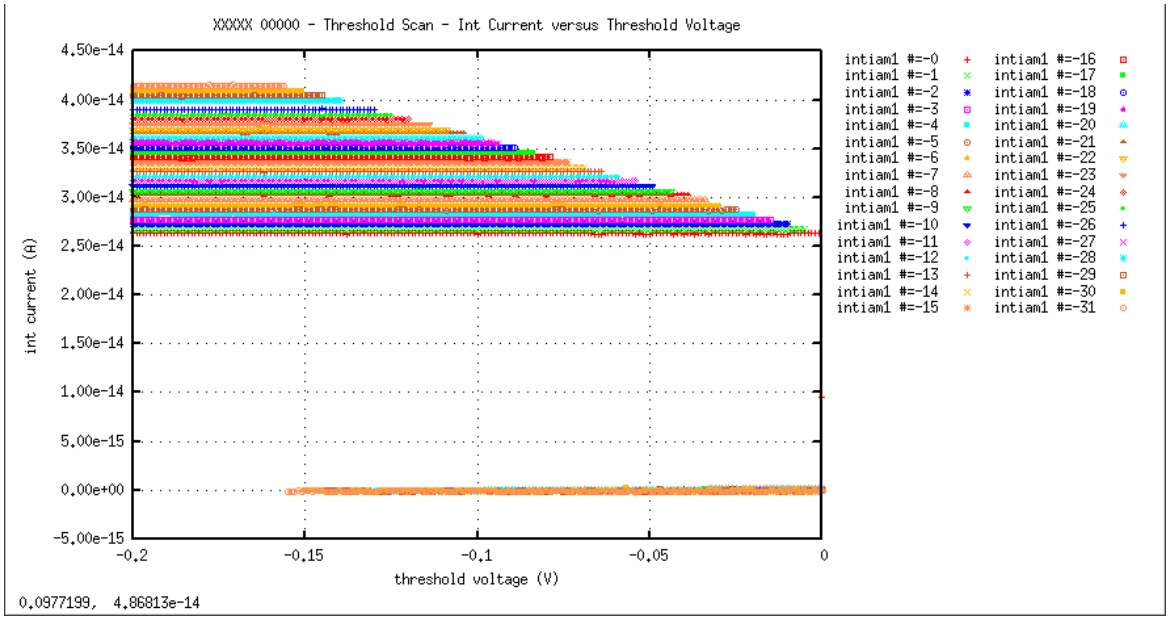


Figure 9.20: Int Current *vs* V_{th} 1 - *scanning mode, opt.val.*

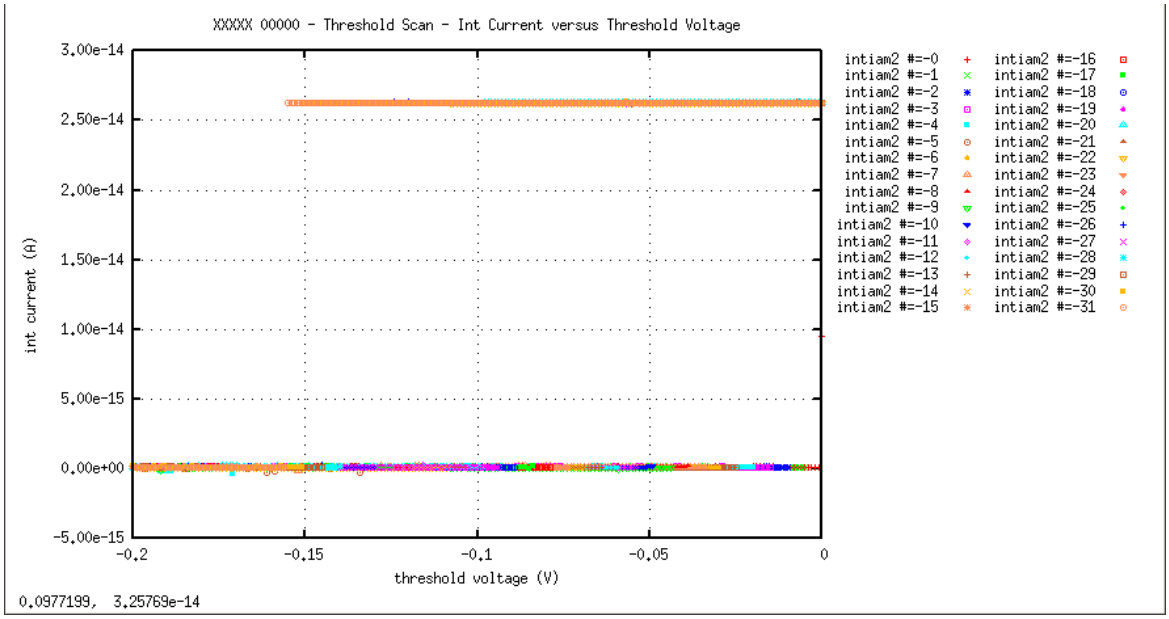


Figure 9.21: Int Current *vs* V_{th} 2 - *scanning mode, opt.val.*

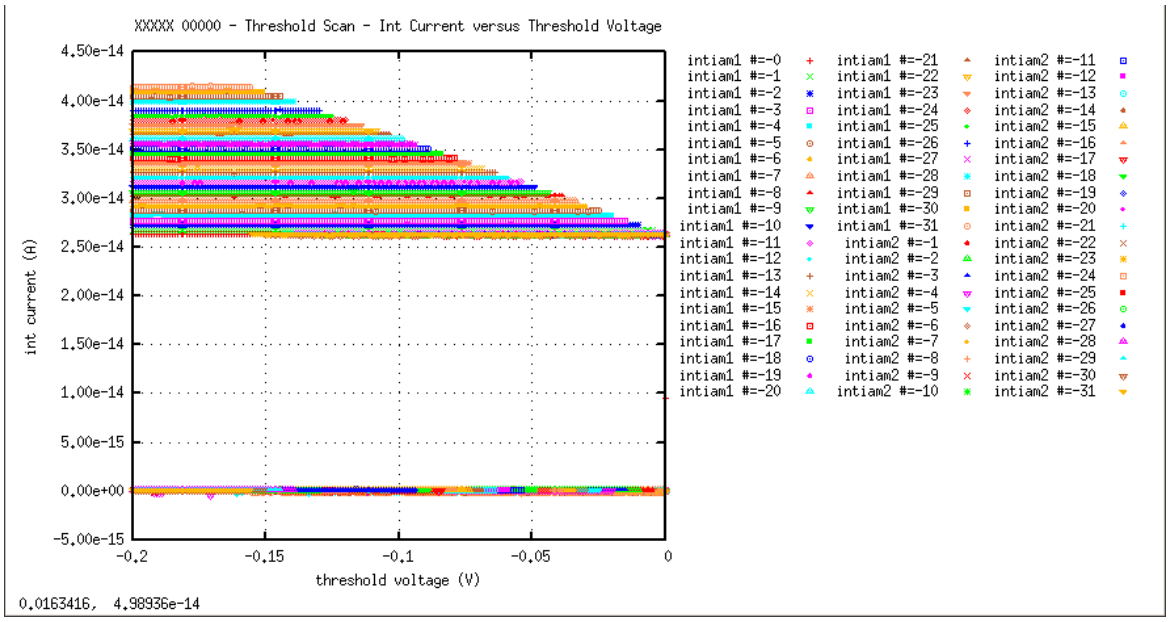


Figure 9.22: Int Current *vs* V_{th} 3 - *scanning mode, opt.val.*

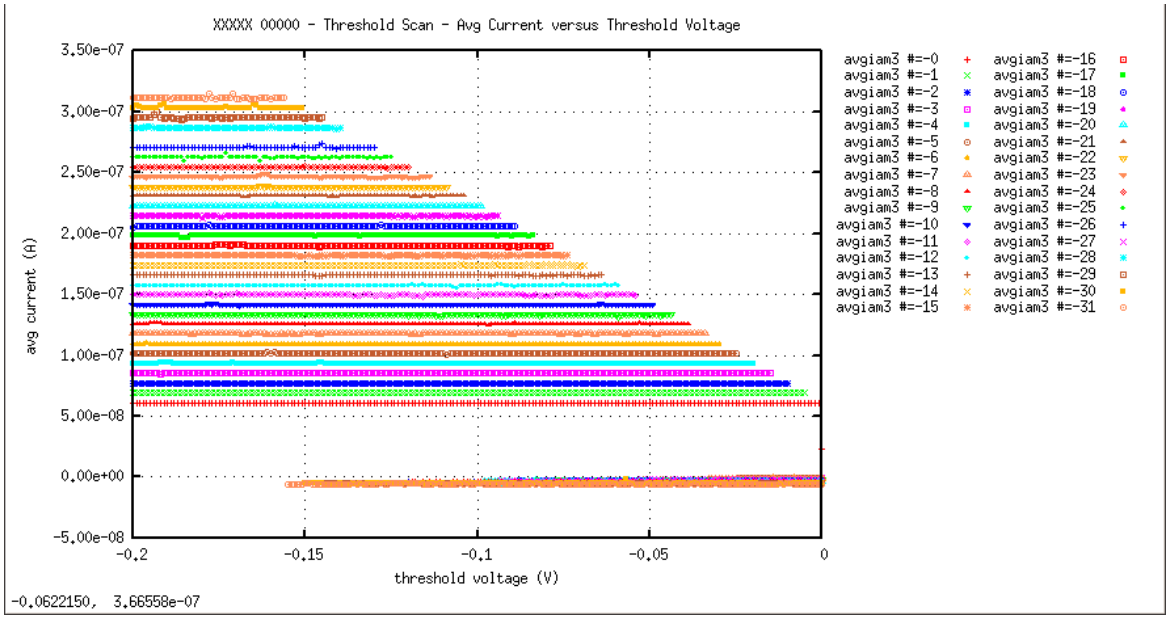


Figure 9.23: Average Current *vs* V_{th} 1 - *scanning mode, opt.val.*

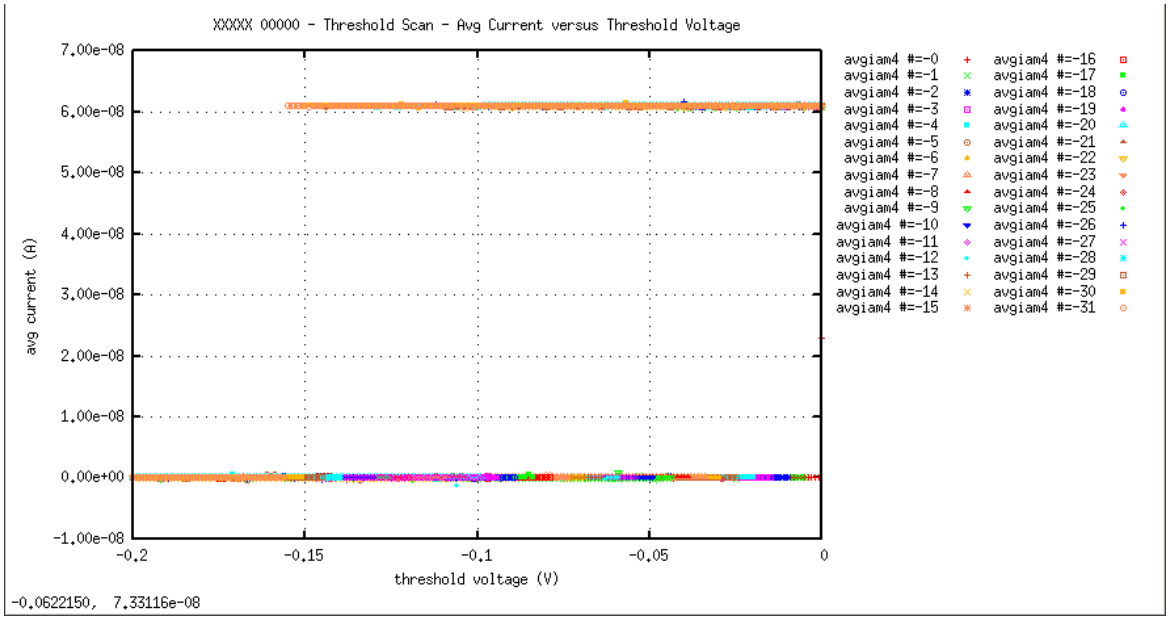


Figure 9.24: Average Current *vs* V_{th} 2 - *scanning mode, opt.val.*

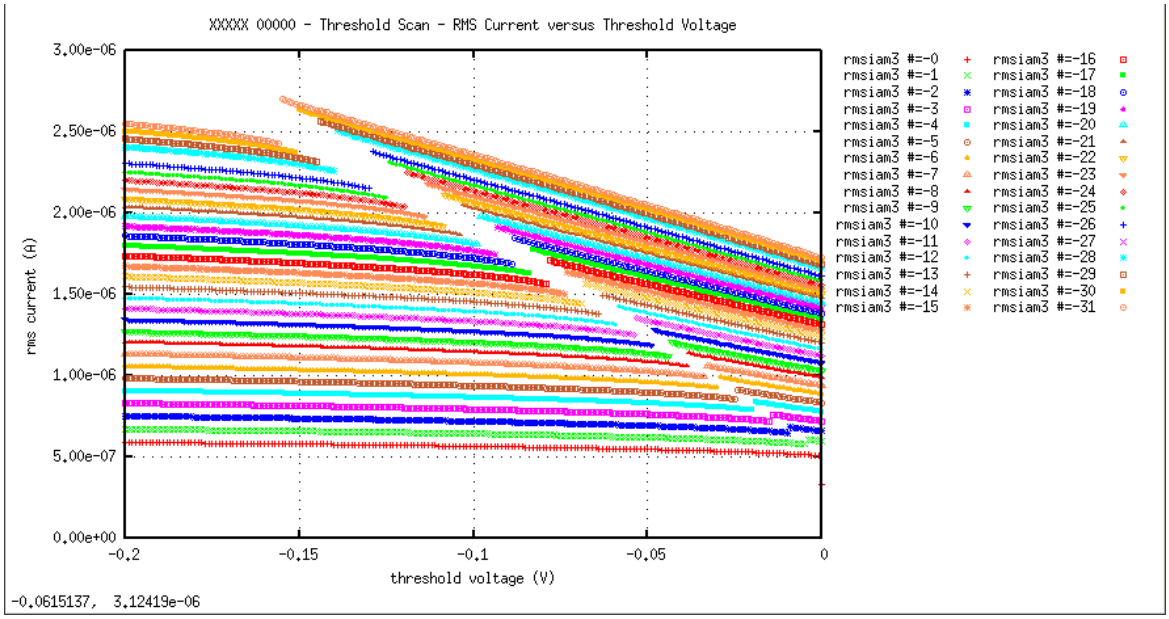


Figure 9.25: RMS Current *vs* V_{th} 4 - *scanning mode, opt.val.*

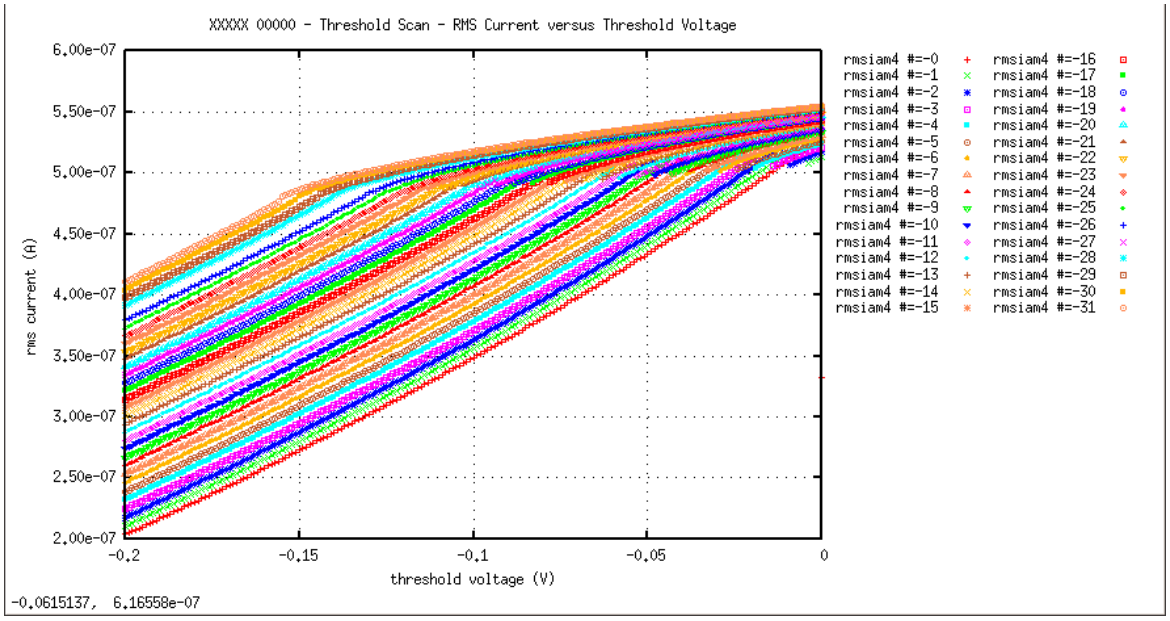


Figure 9.26: RMS Current *vs* V_{th} 5 - *scanning mode, opt.val.*

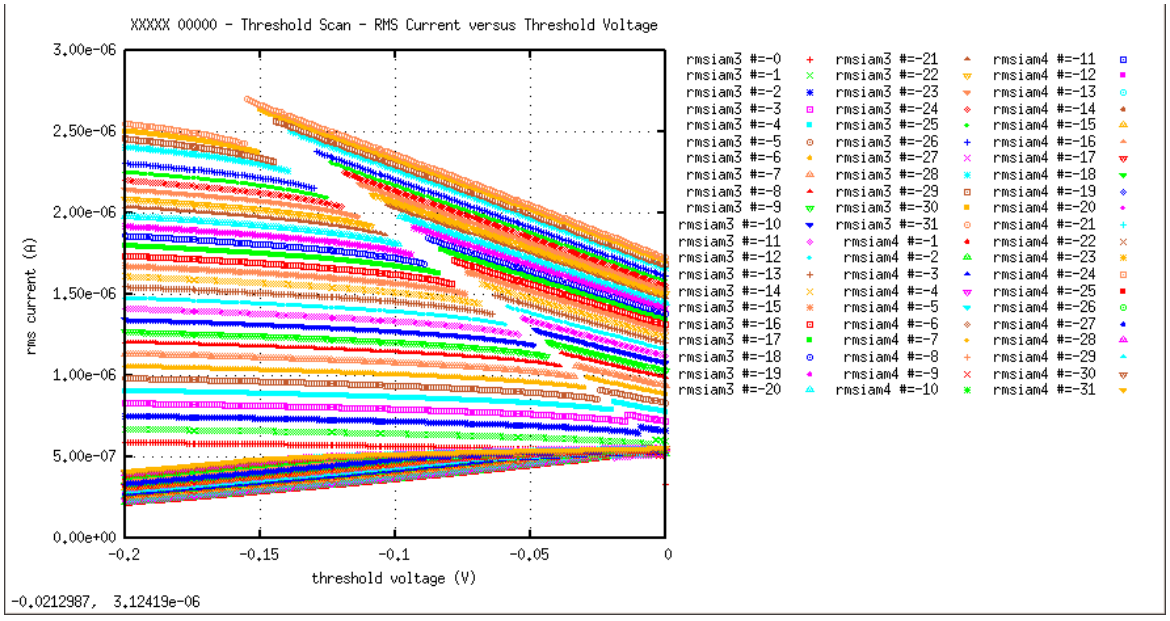


Figure 9.27: RMS Current *vs* V_{th} 6 - *scanning mode, opt.val.*

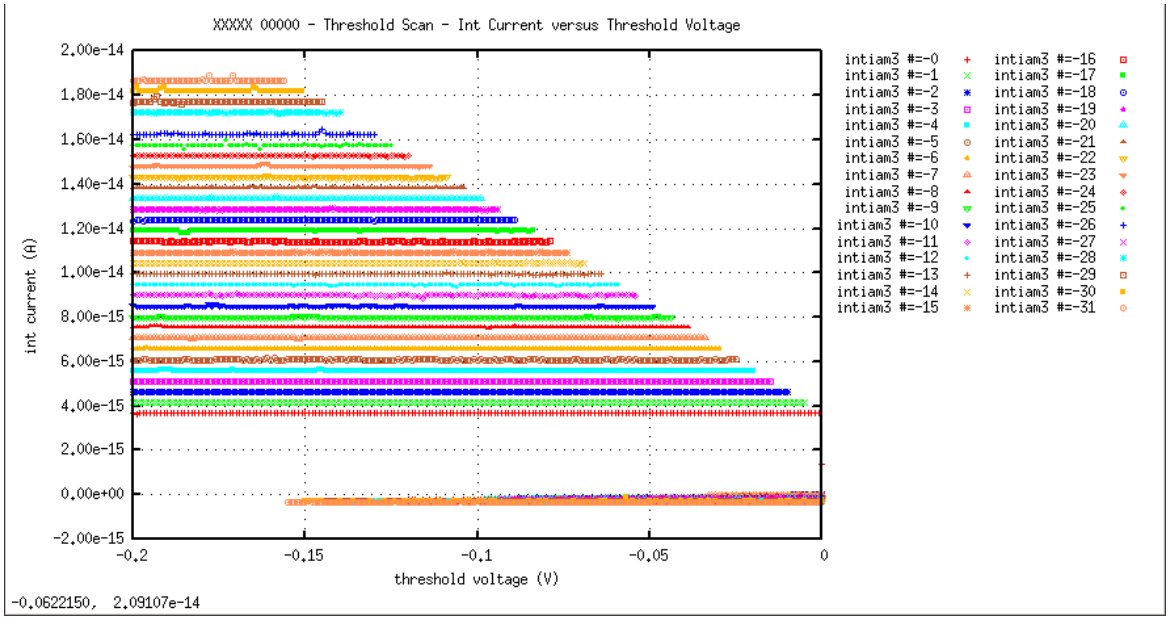


Figure 9.28: Int Current *vs* V_{th} 4 - *scanning mode, opt.val.*

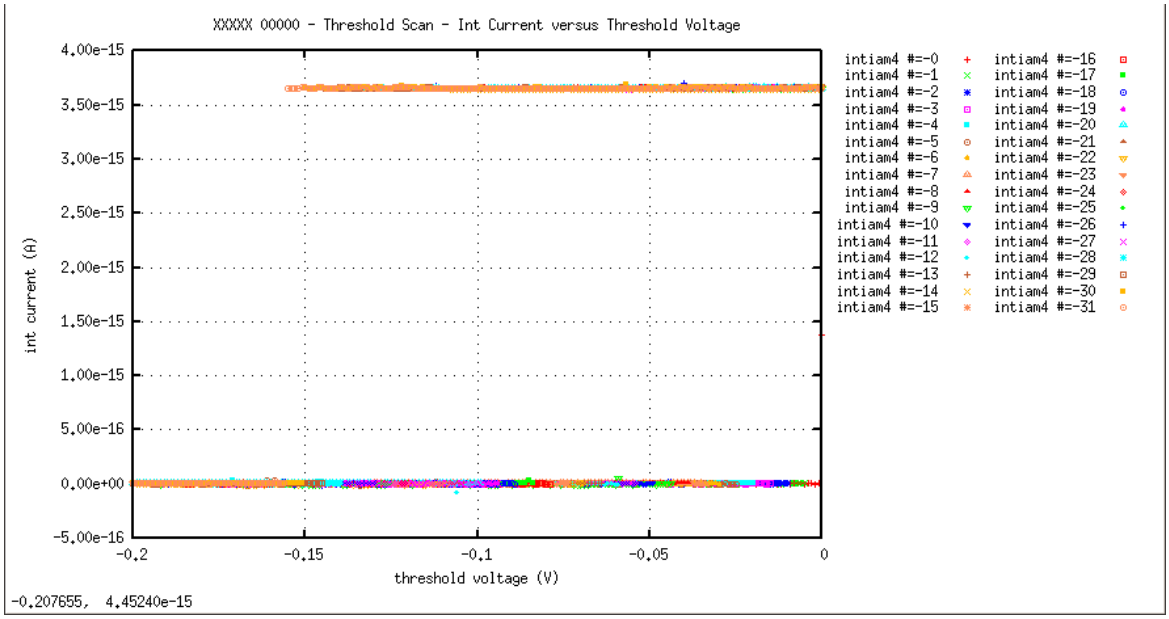


Figure 9.29: Int Current *vs* V_{th} 5 - *scanning mode, opt.val.*

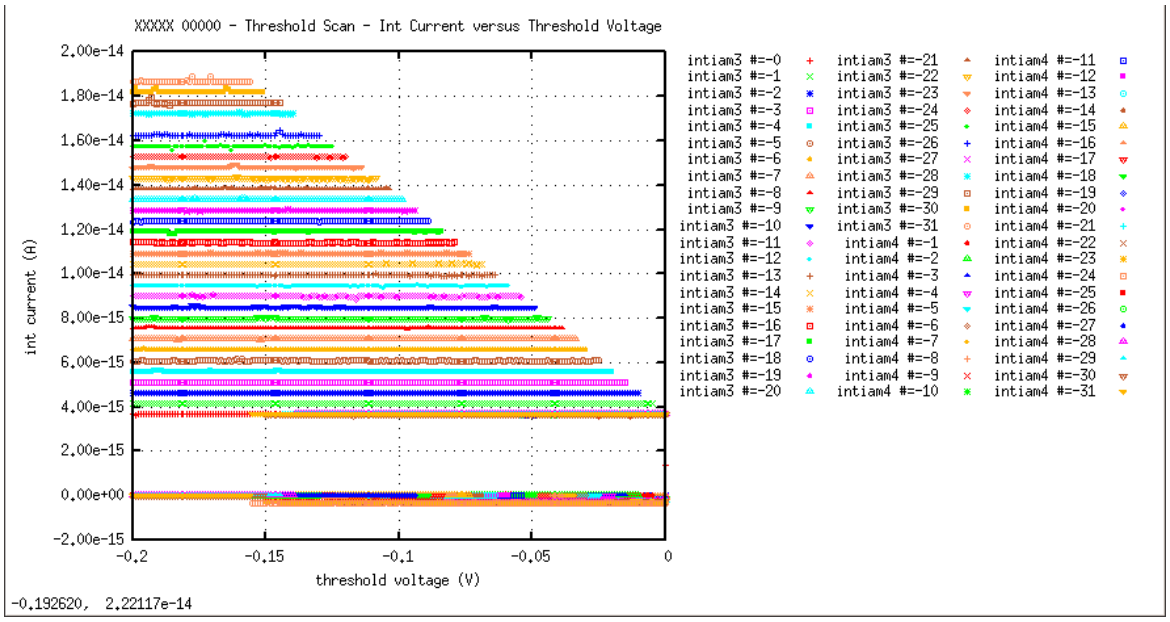


Figure 9.30: Int Current *vs* V_{th} 6 - *scanning mode, opt.val.*

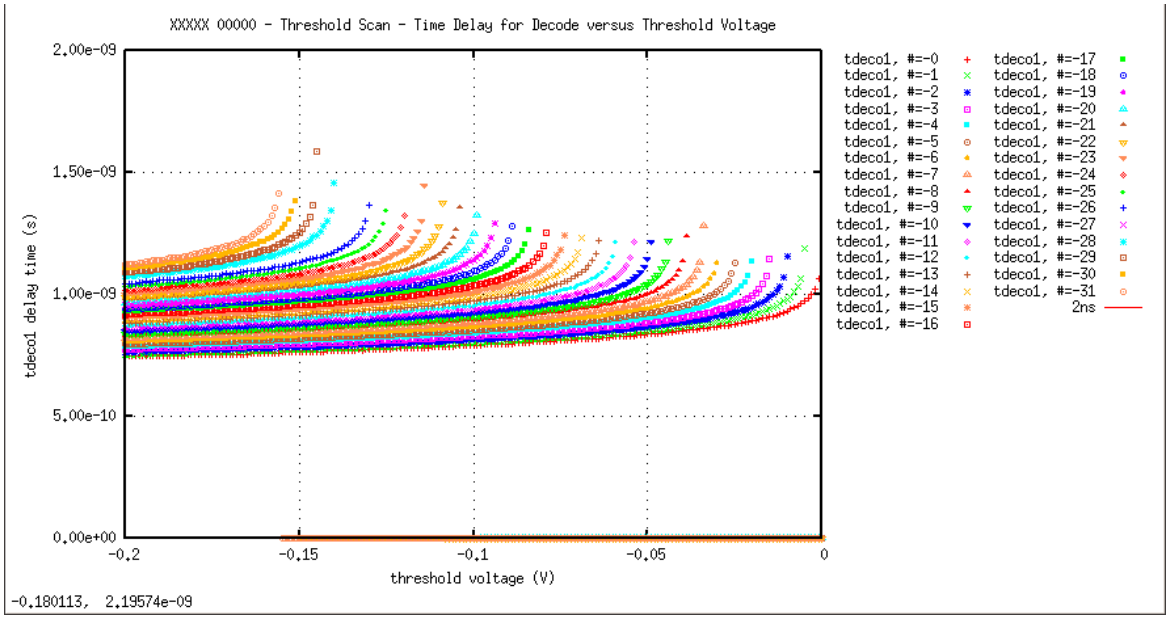


Figure 9.31: Time Delay for Decode *vs* V_{th} 1 - *scanning mode, opt.val.*

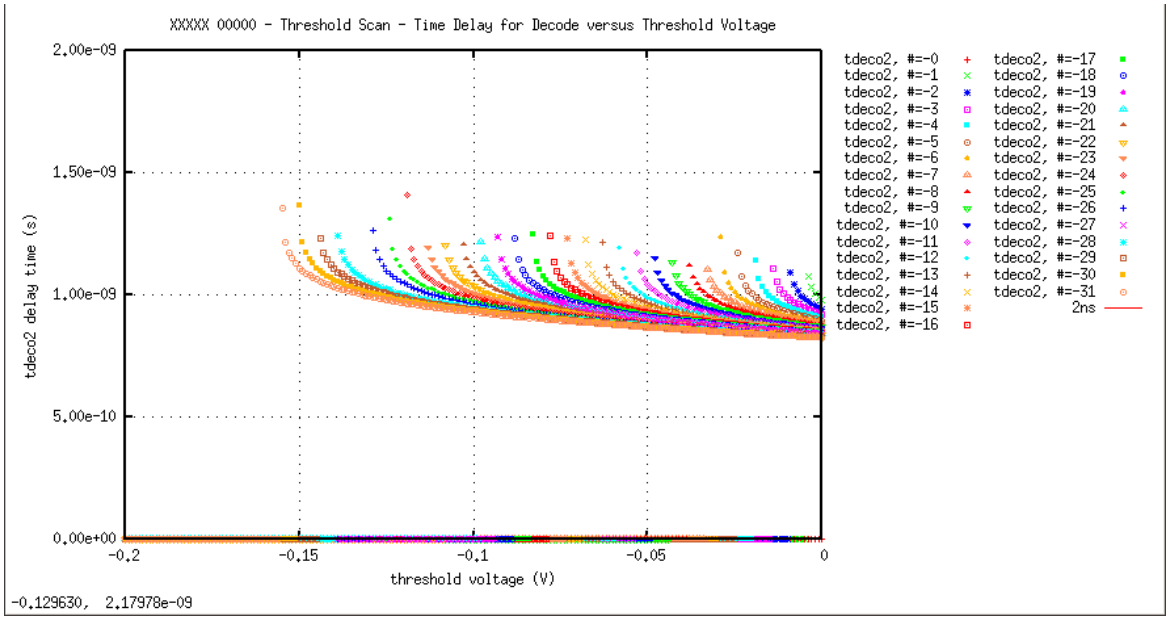


Figure 9.32: Time Delay for Decode *vs* V_{th} 2 - *scanning mode, opt.val.*

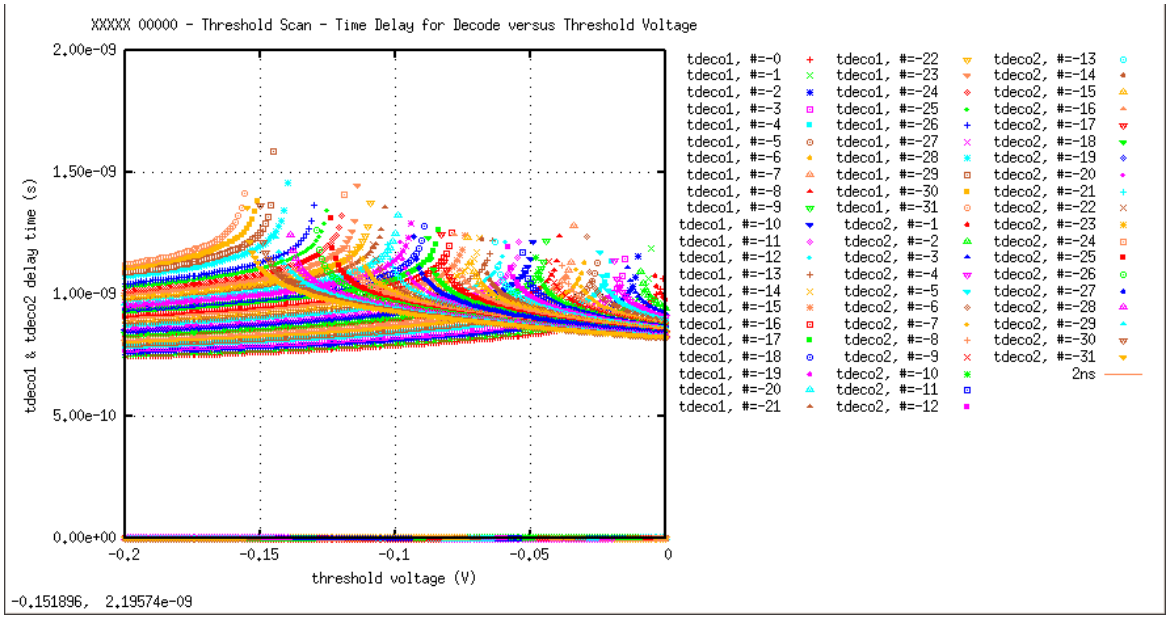


Figure 9.33: Time Delay for Decode *vs* V_{th} 3 - *scanning mode, opt.val.*

9.2 REFERENCE VALUES – *scanning* METHOD

Plots presented in this section are generated by GnuPlot from the *scanning mode* simulation data.

Each device size is set to the reference value (see Table 9.1).

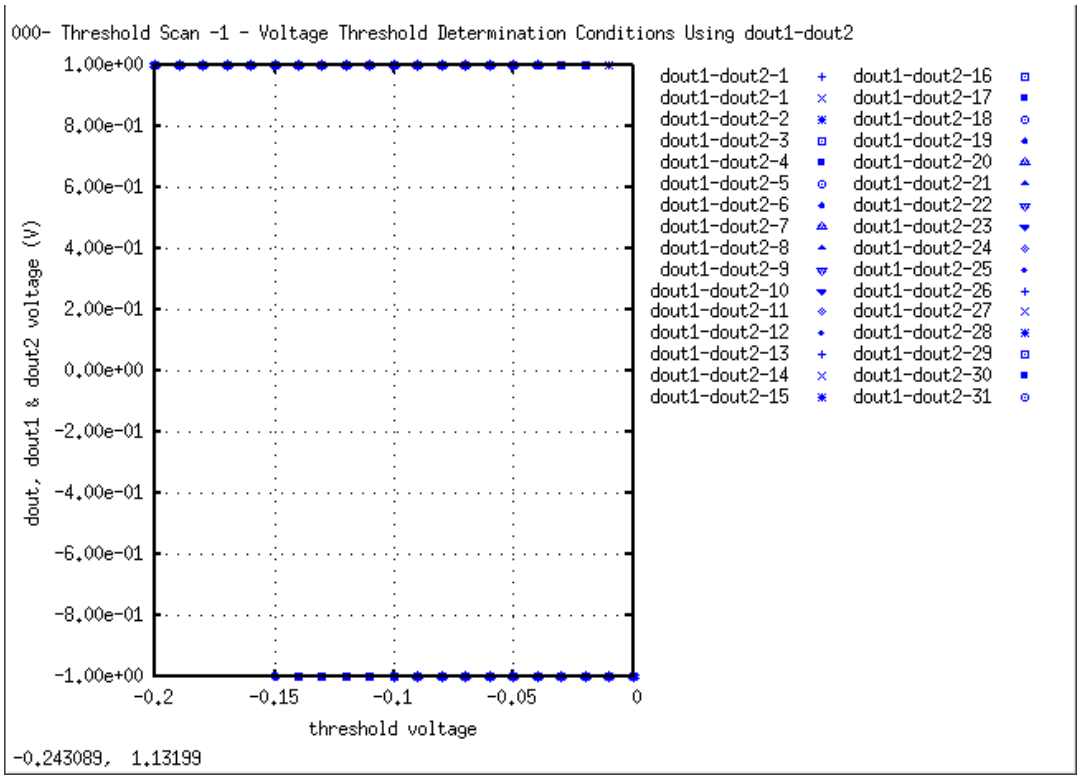


Figure 9.34: V_{th} Determination Conditions 1 - *scanning mode, ref.val.*

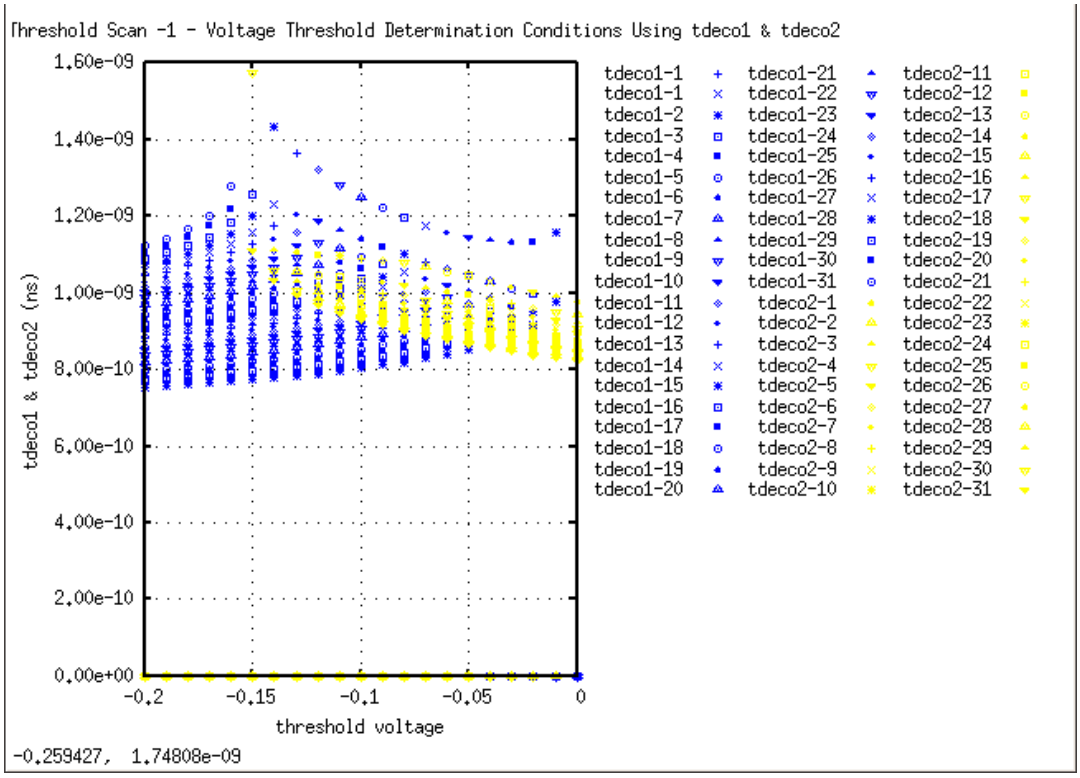


Figure 9.35: V_{th} Determination Conditions 2 - *scanning mode, ref.val.*

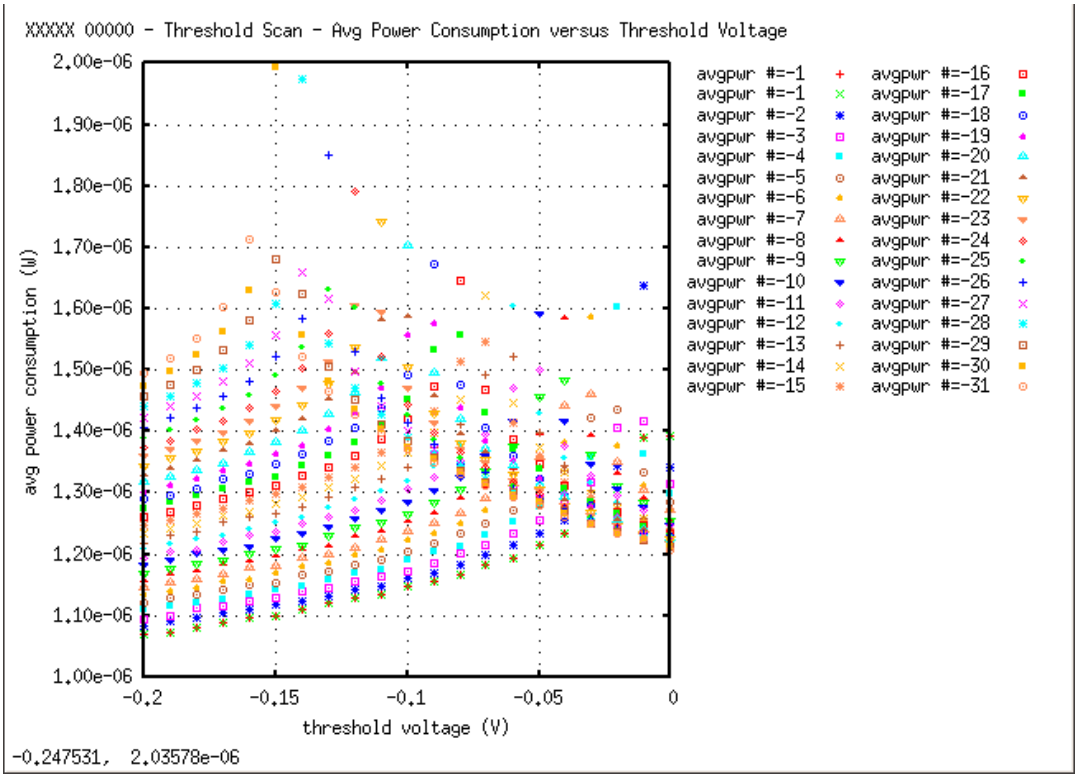


Figure 9.36: Average Power Consumption *vs* V_{th} *- scanning mode, ref.val.*

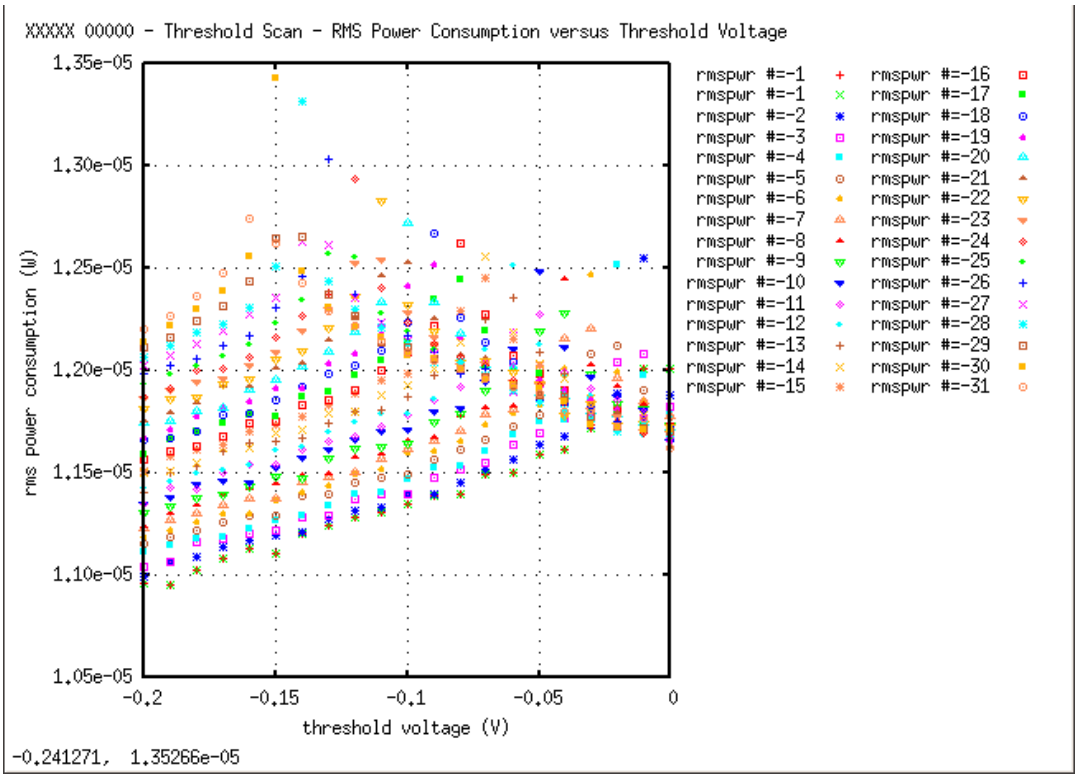


Figure 9.37: RMS Power Consumption *vs* V_{th} *- scanning mode, ref.val.*

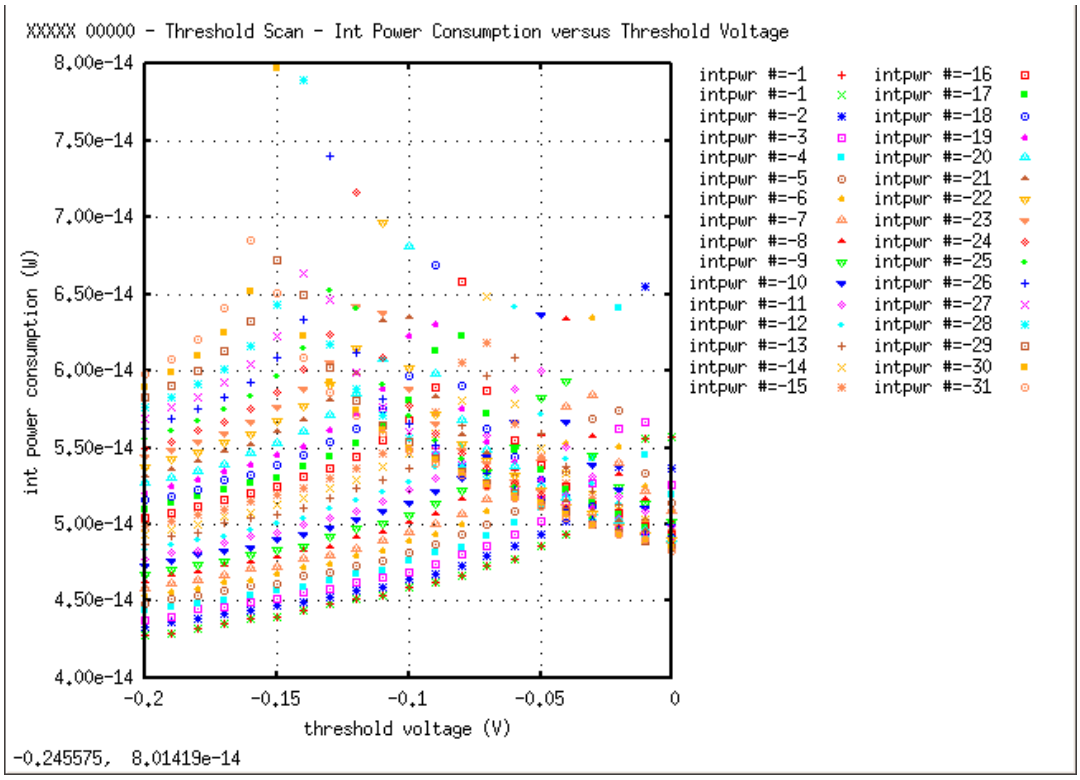


Figure 9.38: Int Power Consumption *vs* V_{th} - *scanning mode, ref.val.*

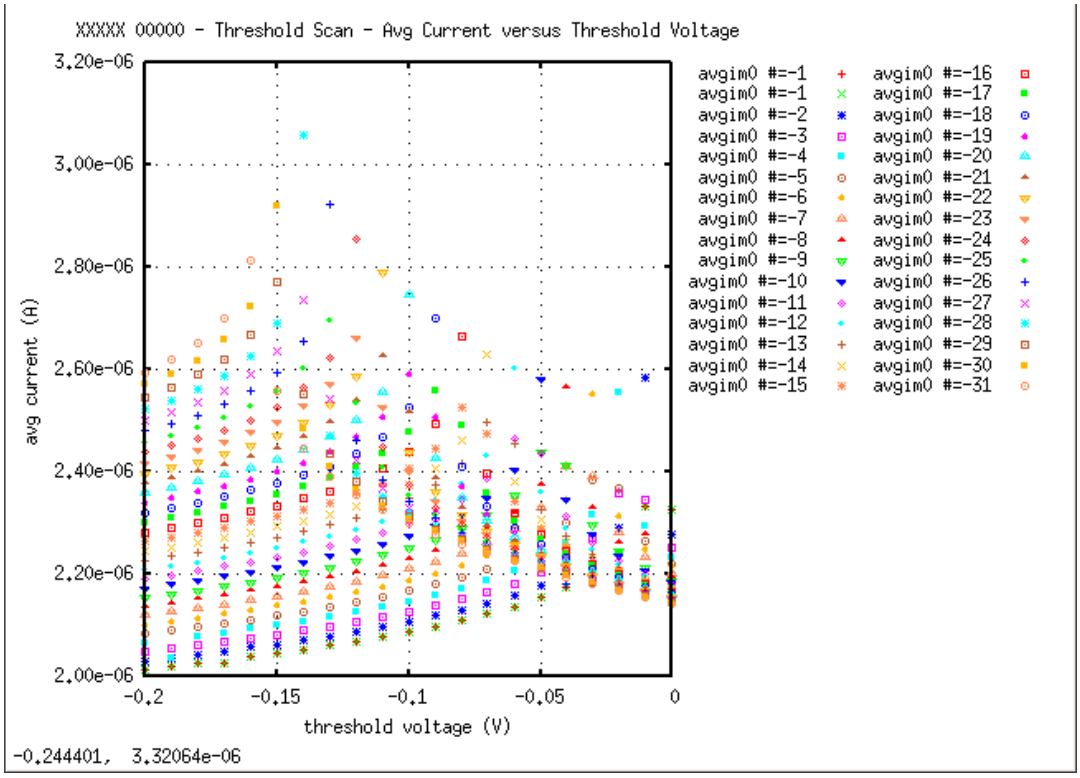


Figure 9.39: Average Current *vs* V_{th} 1 - *scanning mode, ref.val.*

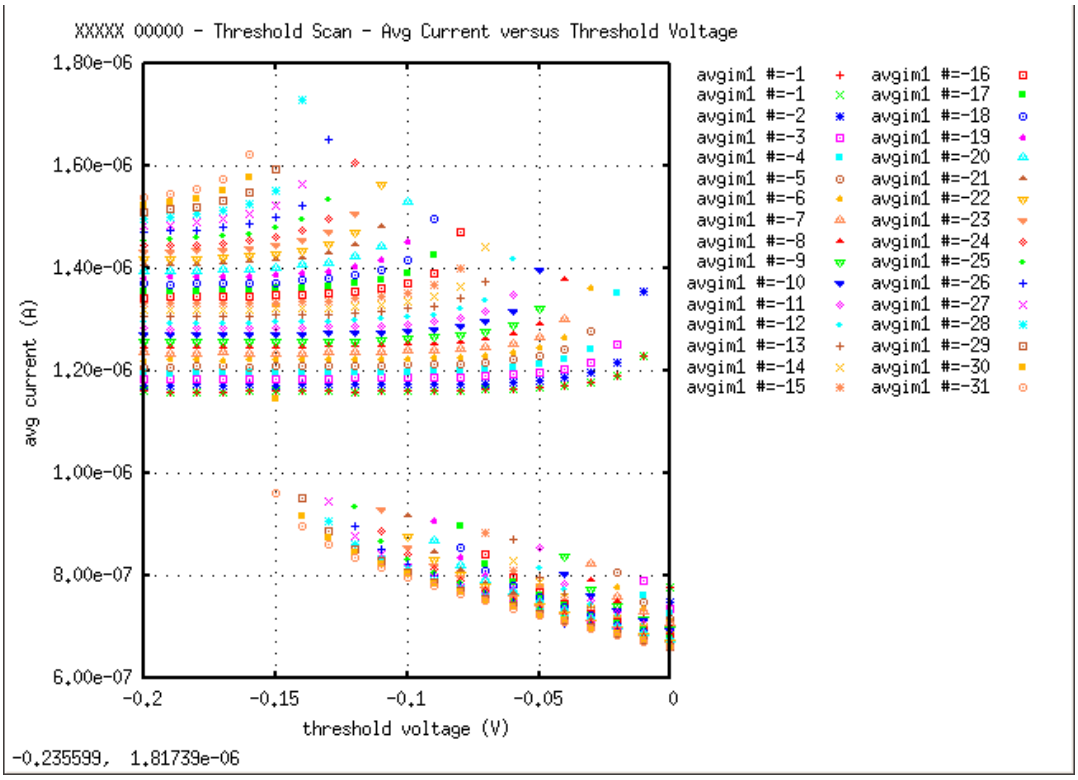


Figure 9.40: Average Current *vs* V_{th} 2 - *scanning mode, ref.val.*

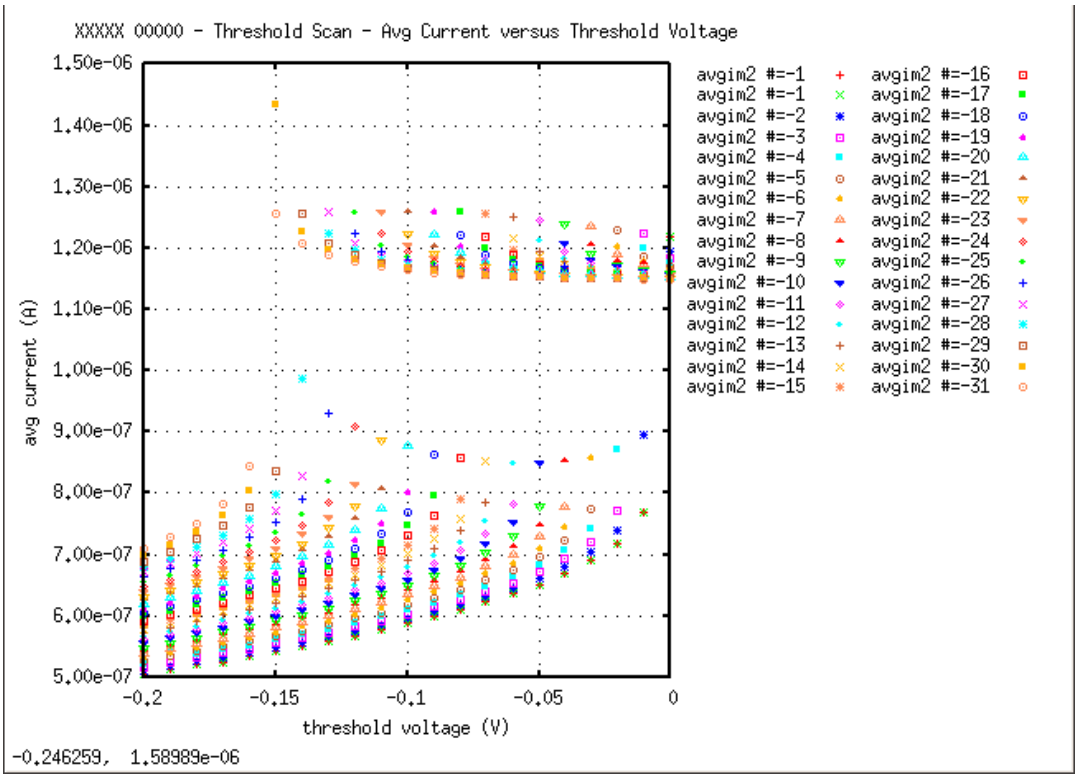


Figure 9.41: Average Current *vs* V_{th} 3 - *scanning mode, ref.val.*

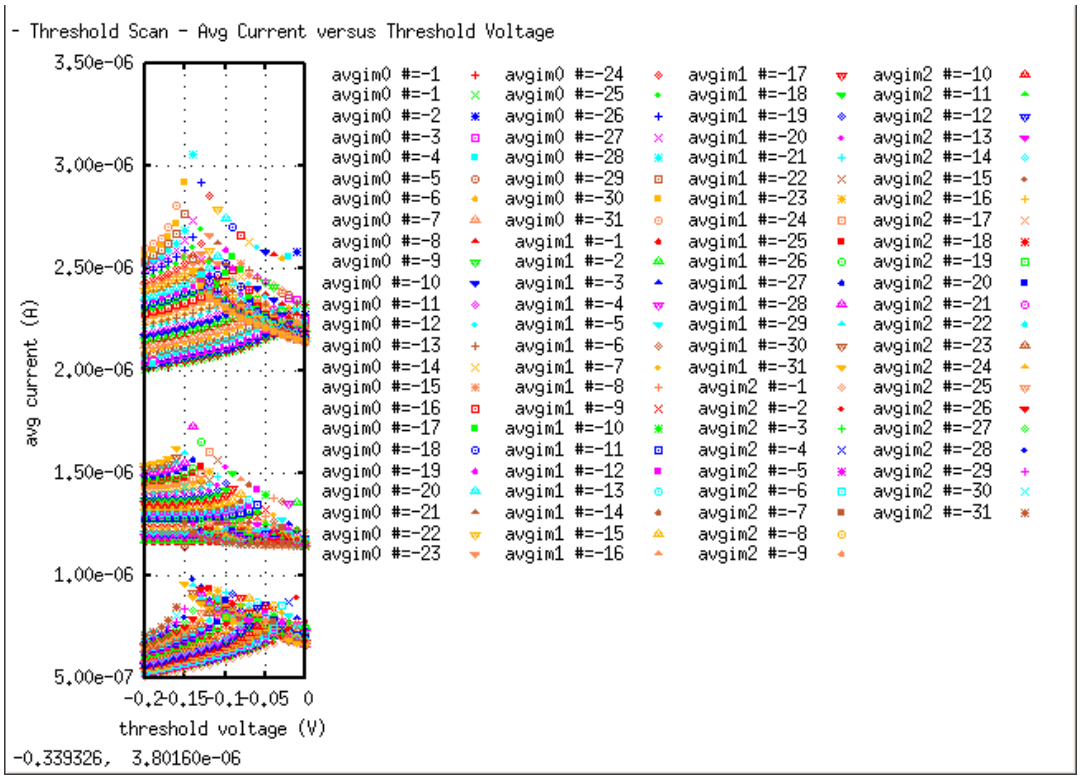


Figure 9.42: Average Current *vs* V_{th} 4 - *scanning mode, ref.val.*

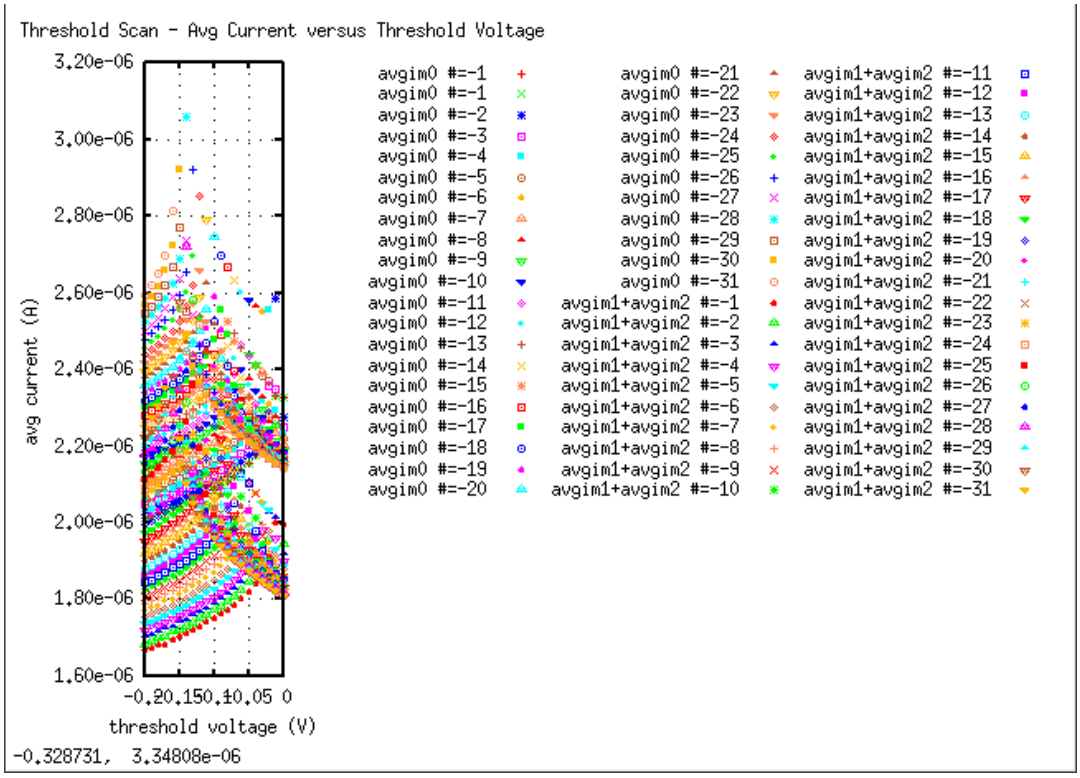


Figure 9.43: Average Current *vs* V_{th} 5 - *scanning mode, ref.val.*

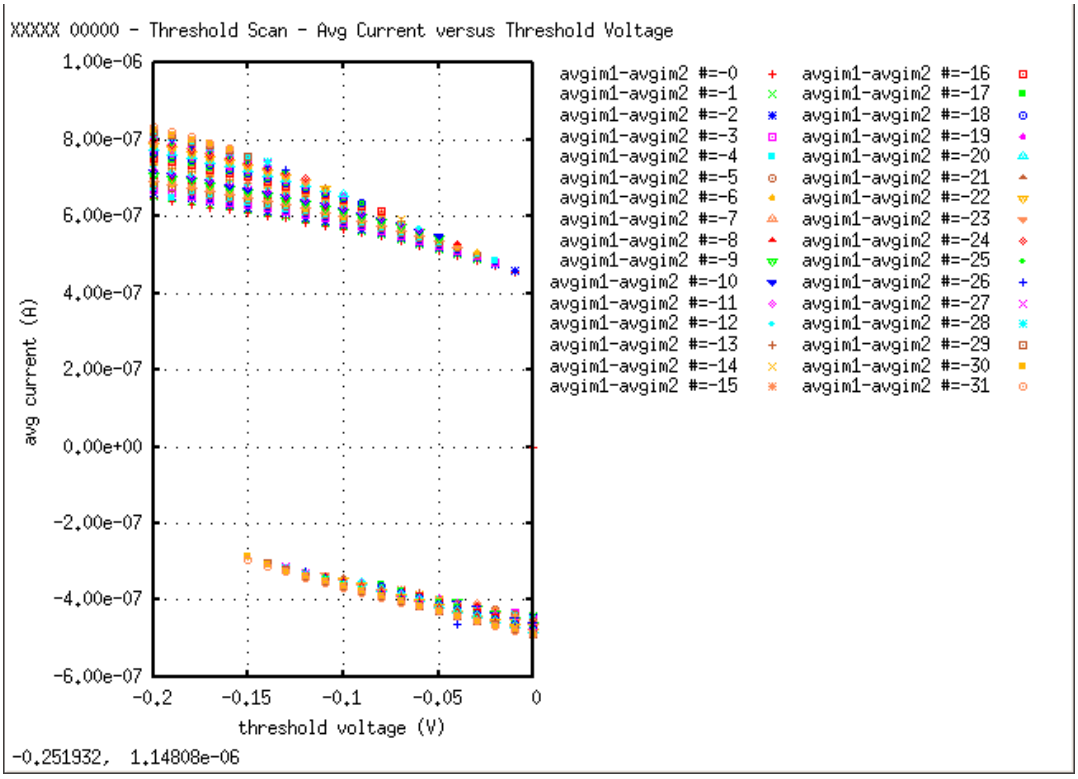


Figure 9.44: Average Current *vs* V_{th} 6 - *scanning mode, ref.val.*

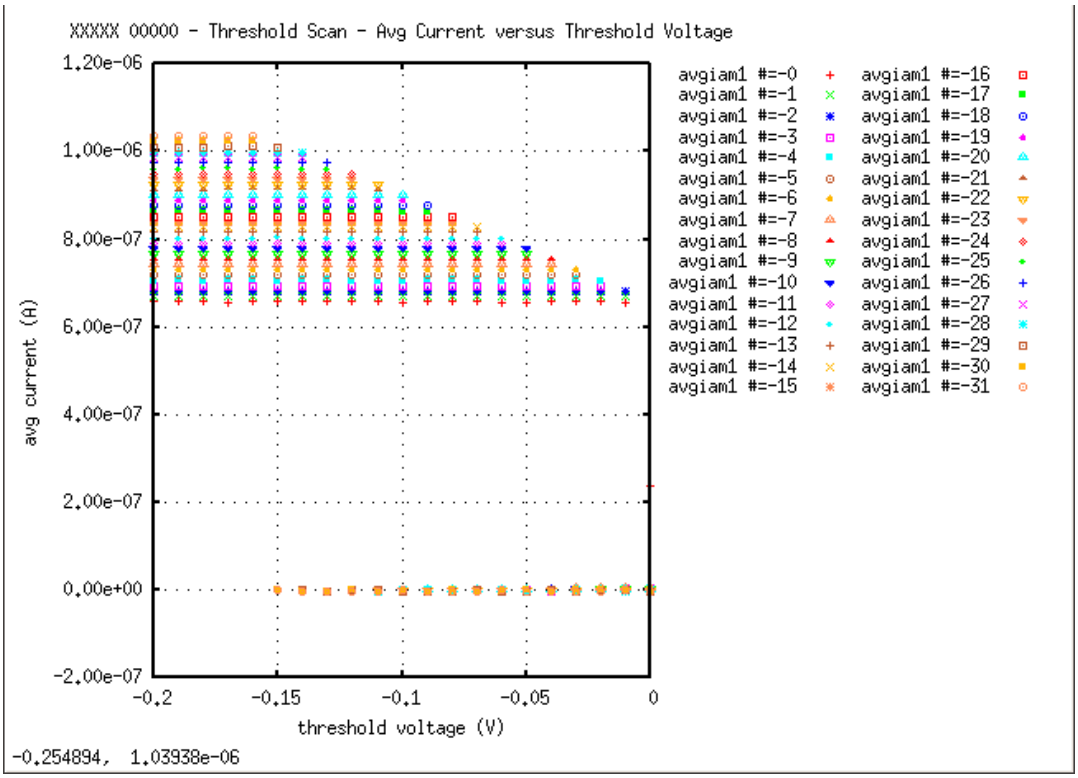


Figure 9.45: Average Current *vs* V_{th} 7 - *scanning mode, ref.val.*

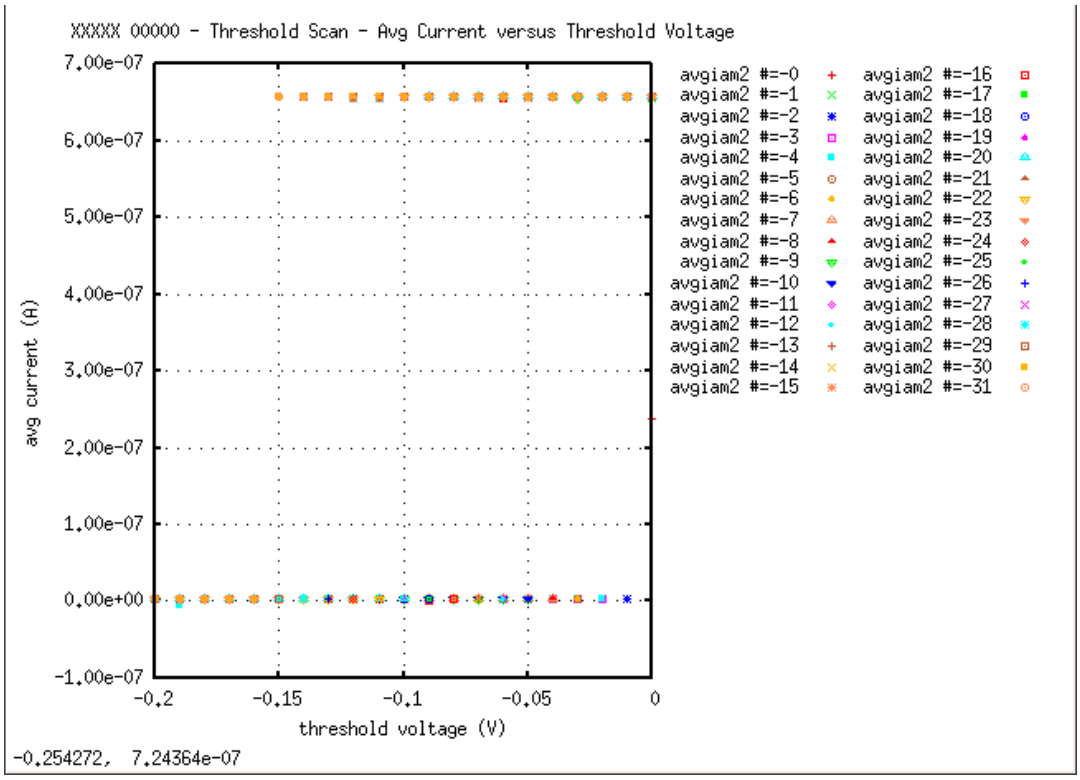


Figure 9.46: Average Current *vs* V_{th} 8 - *scanning mode, ref.val.*

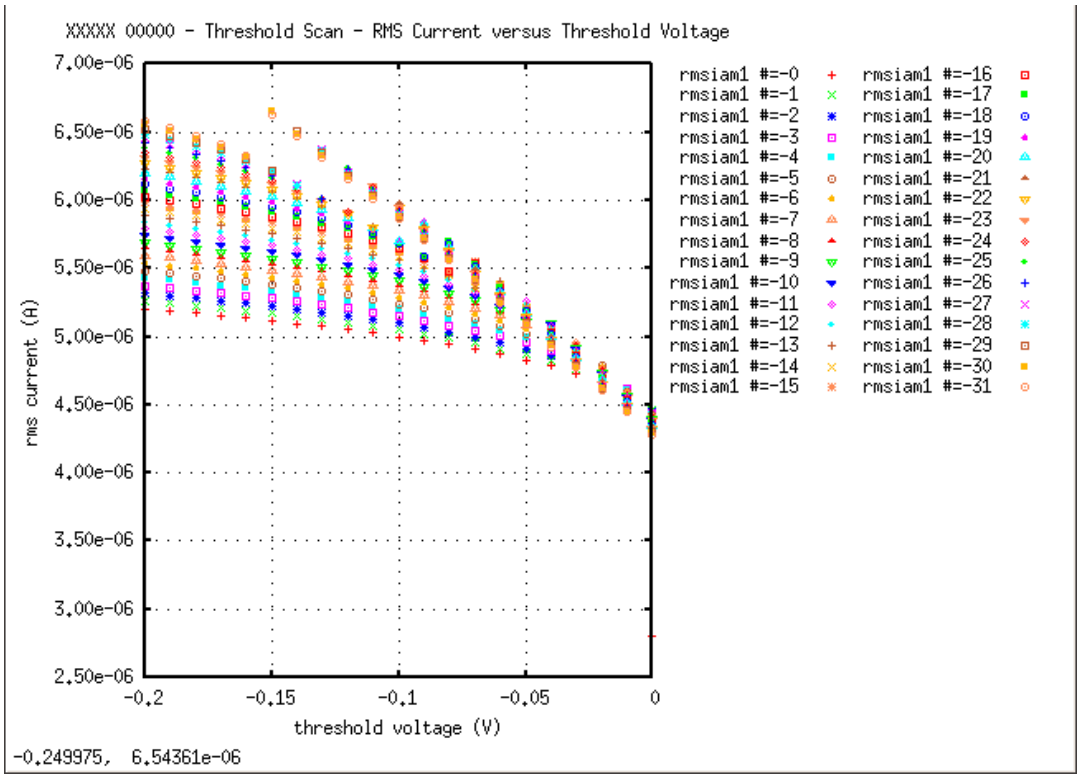


Figure 9.47: RMS Current *vs* V_{th} 1 - *scanning mode, ref.val.*

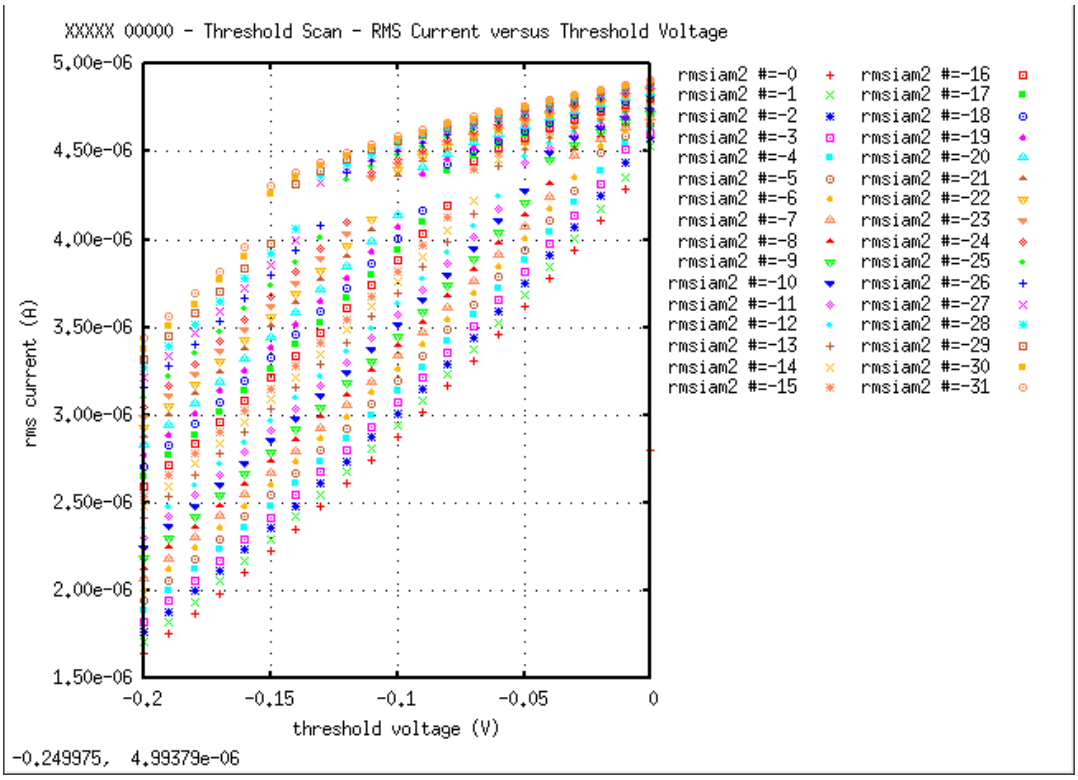


Figure 9.48: RMS Current *vs* V_{th} 2 - *scanning mode, ref.val.*

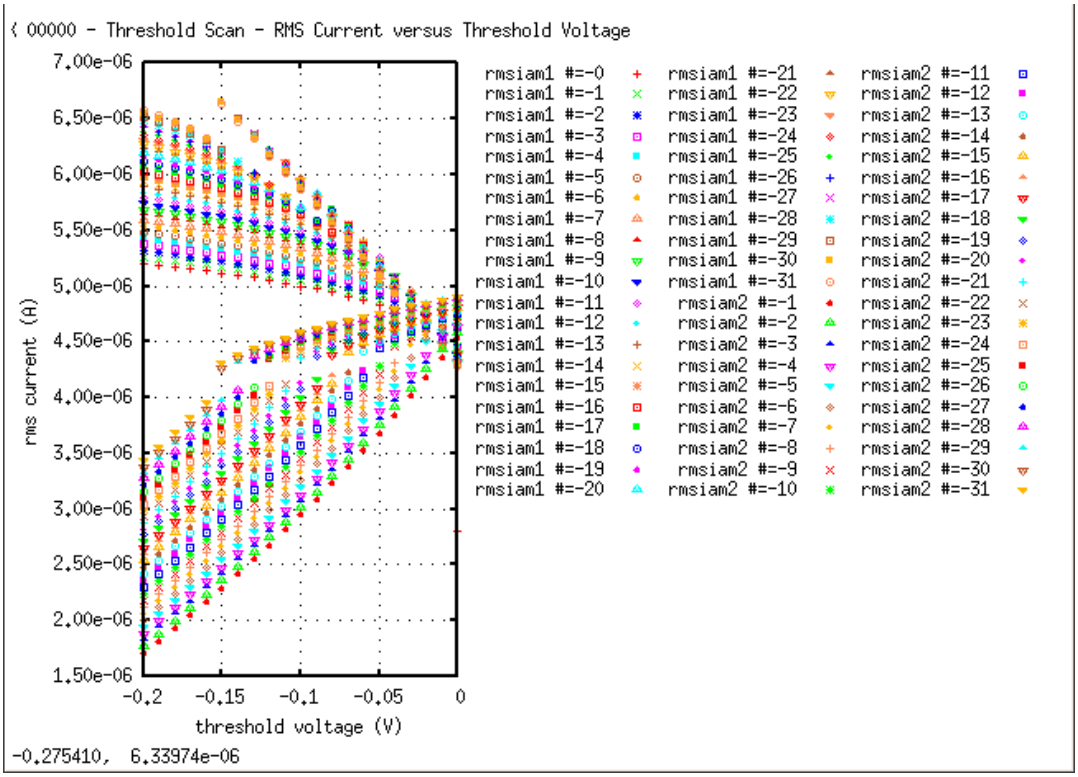


Figure 9.49: RMS Current *vs* V_{th} 3 - *scanning mode, ref.val.*

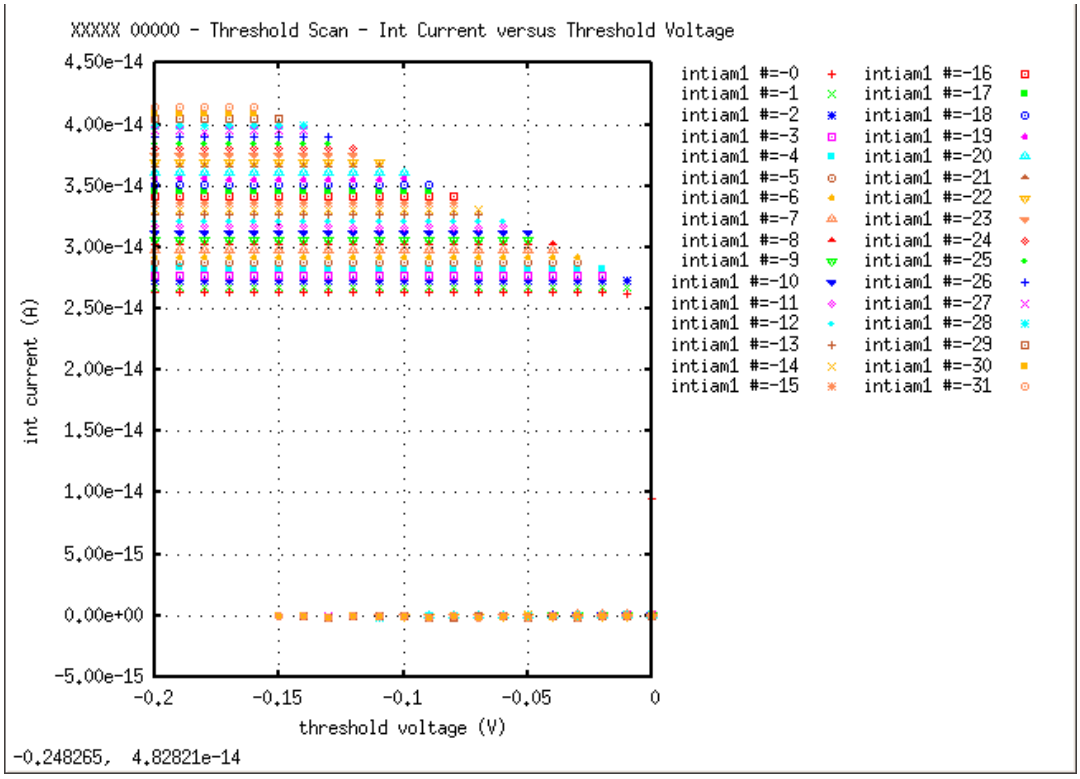


Figure 9.50: Int Current *vs* V_{th} 1 - *scanning mode, ref.val.*

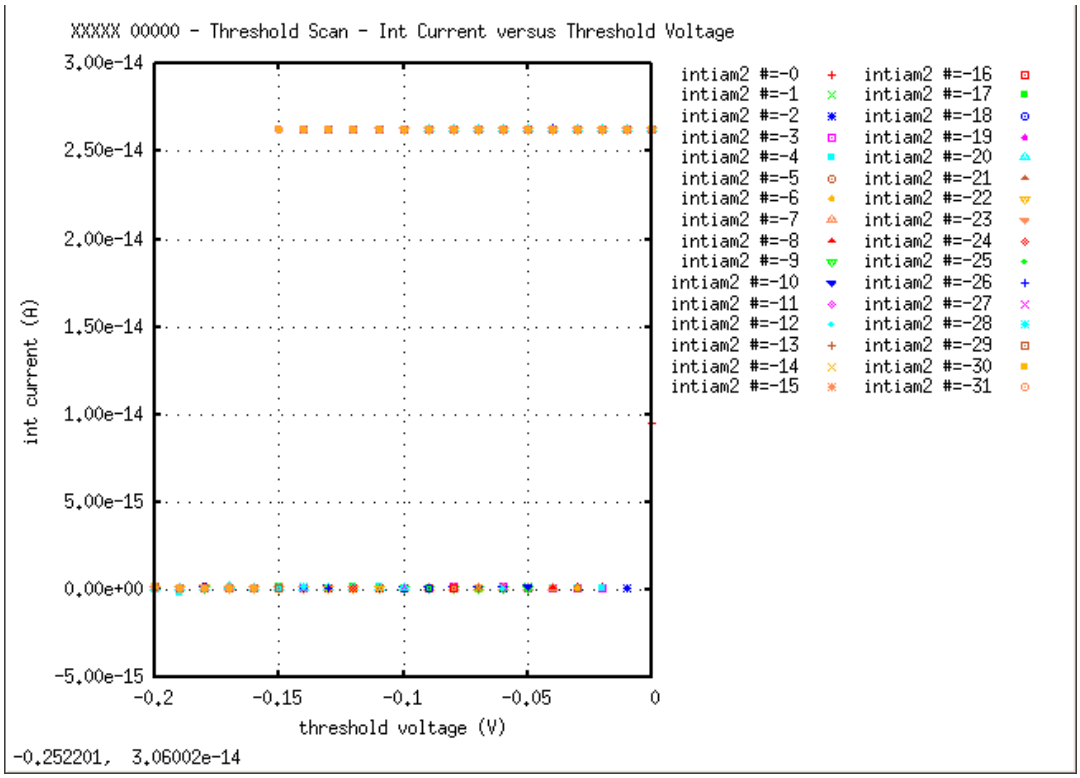


Figure 9.51: Int Current *vs* V_{th} 2 - *scanning mode, ref.val.*

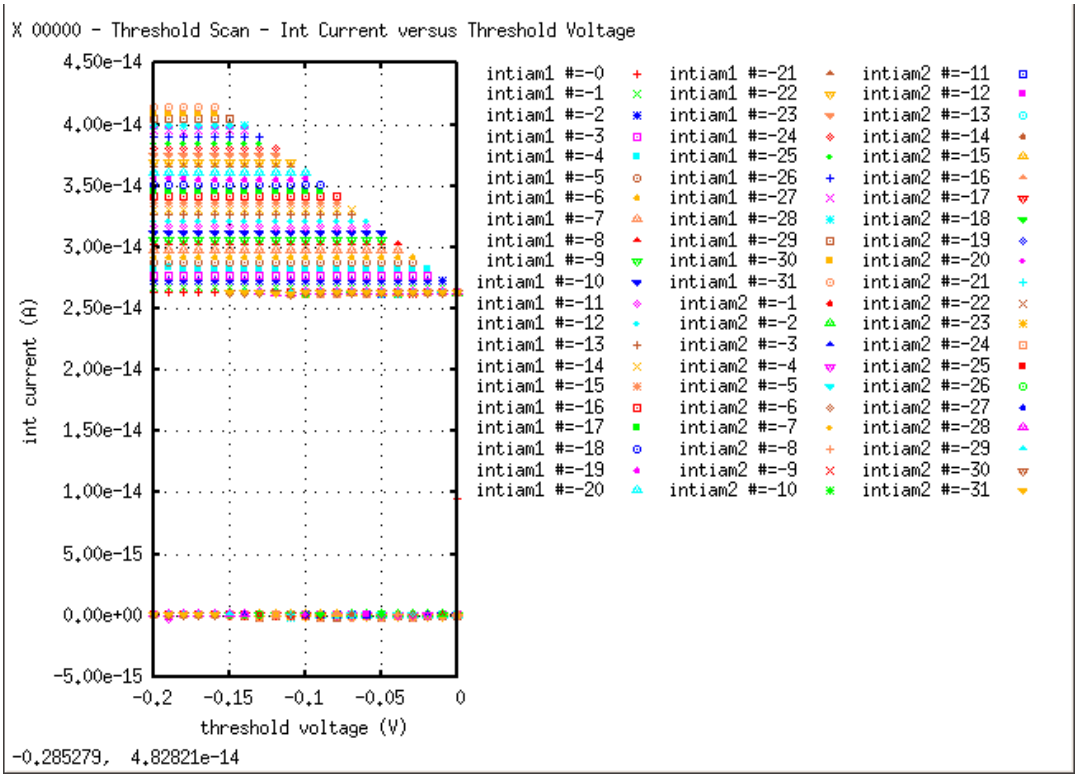


Figure 9.52: Int Current *vs* V_{th} 3 - *scanning mode, ref.val.*

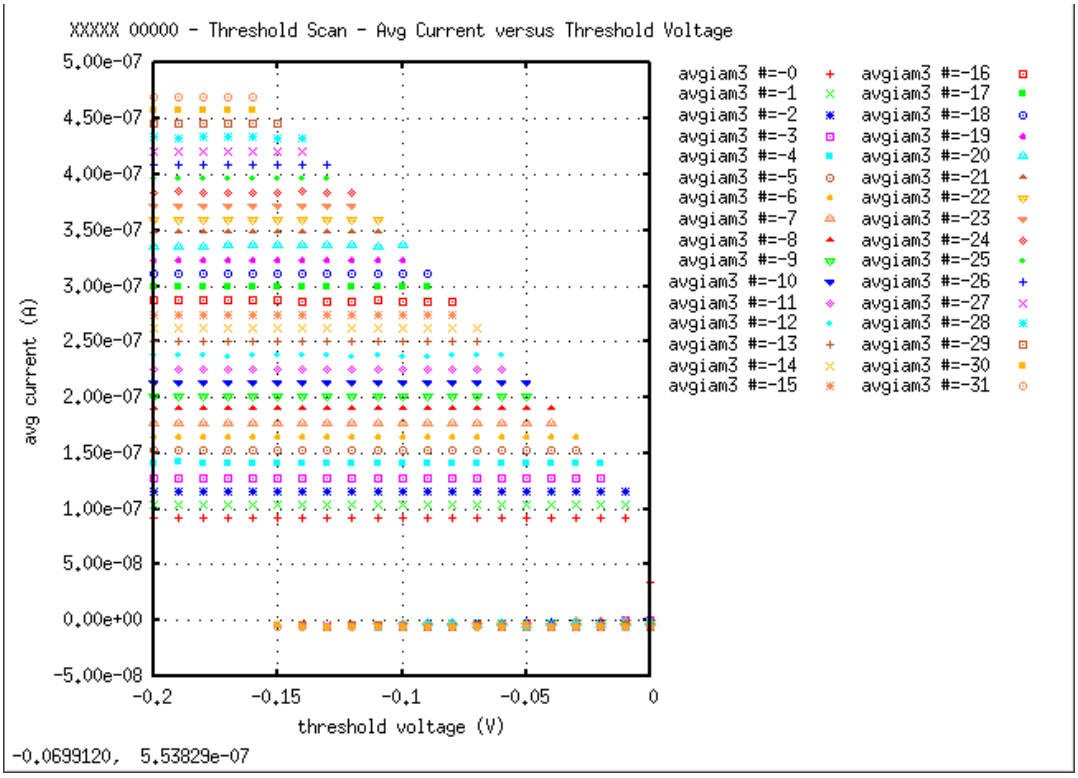


Figure 9.53: Average Current *vs* V_{th} 1 - *scanning mode, ref.val.*

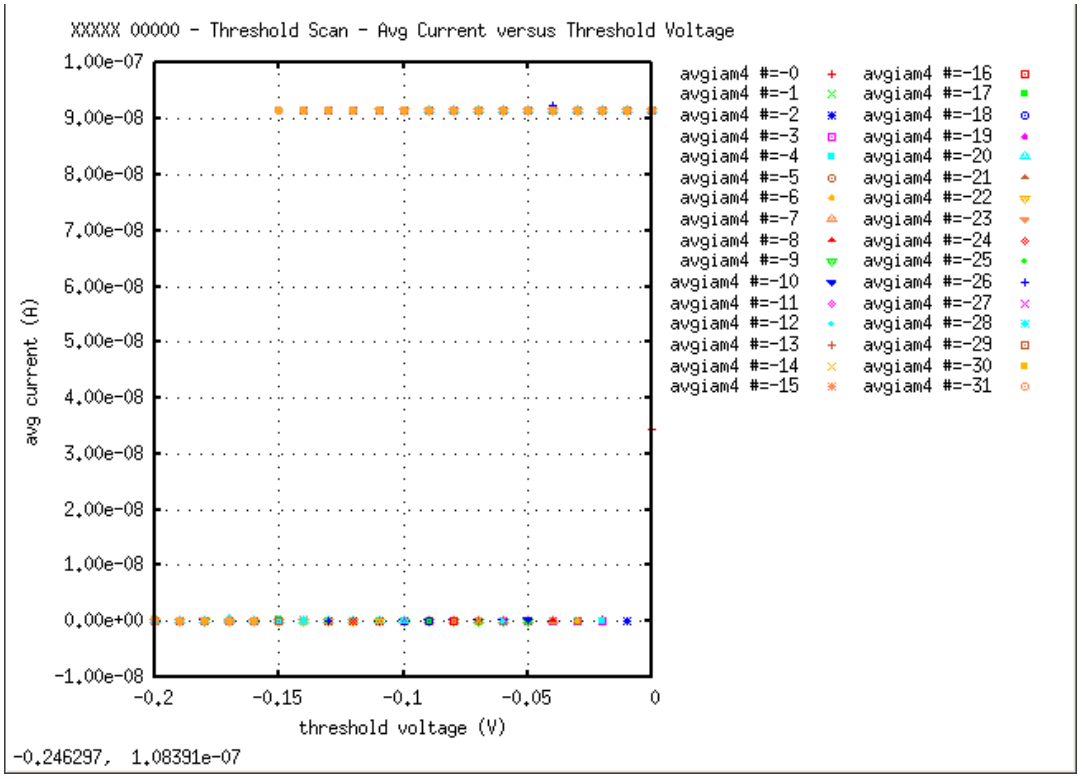


Figure 9.54: Average Current *vs* V_{th} 2 - *scanning mode, ref.val.*

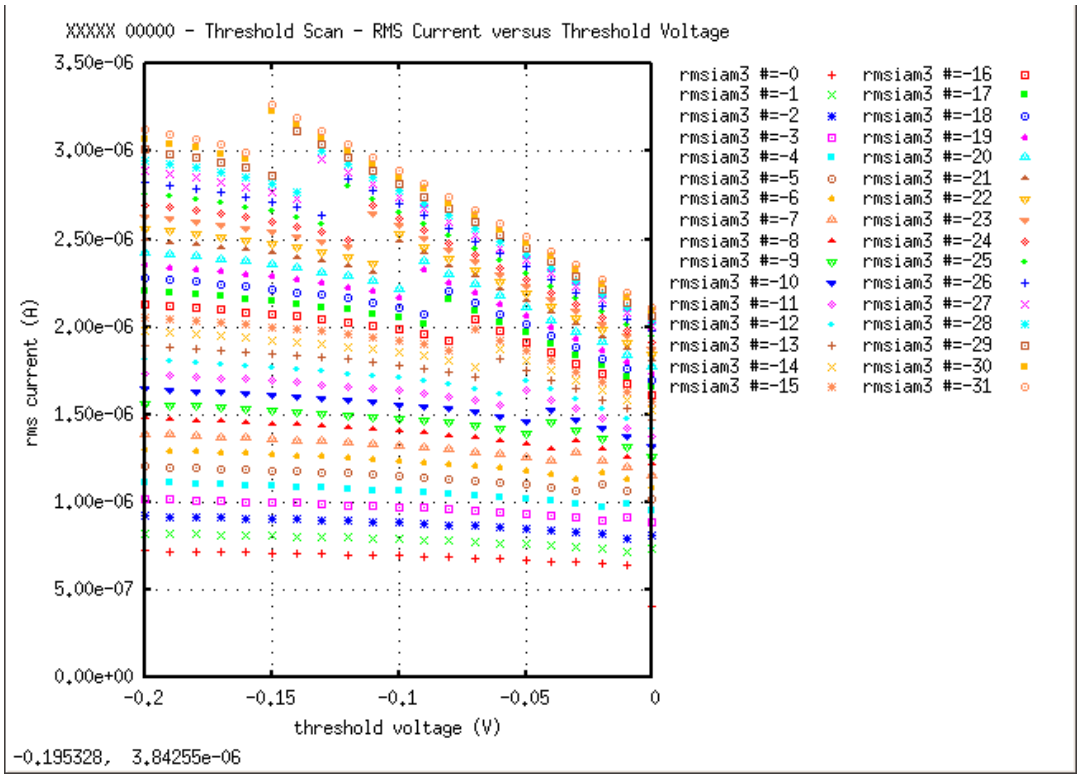


Figure 9.55: RMS Current *vs* V_{th} 4 - *scanning mode, ref.val.*

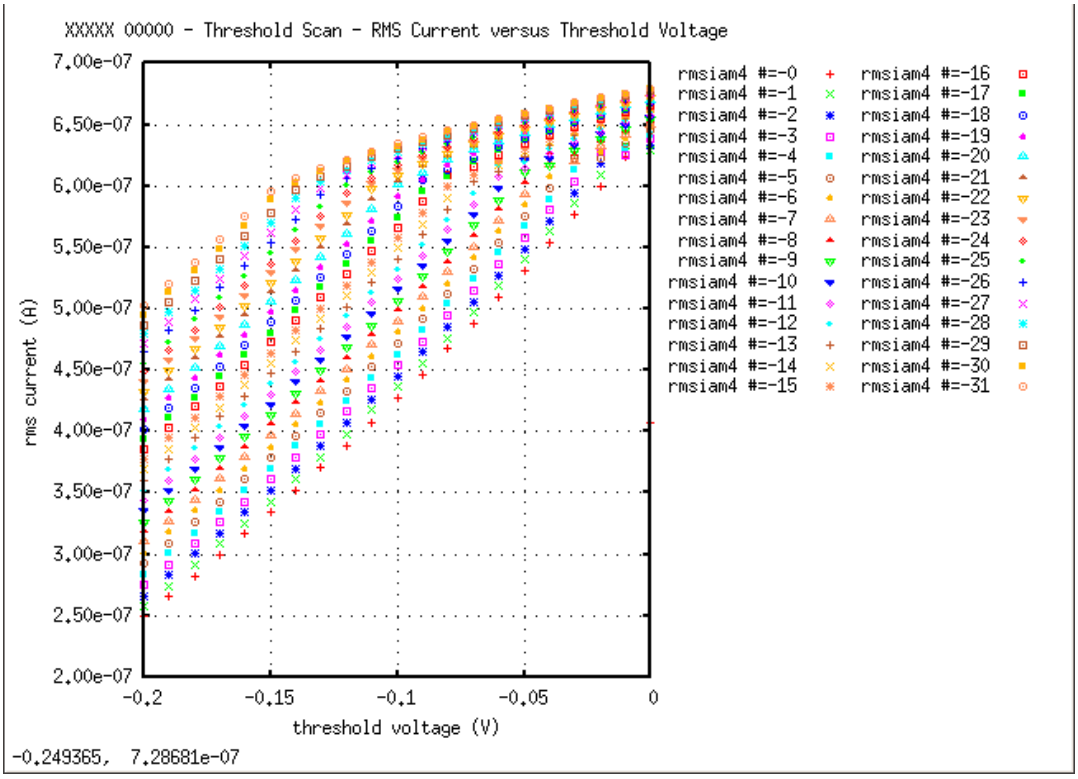


Figure 9.56: RMS Current *vs* V_{th} 5 - *scanning mode, ref.val.*

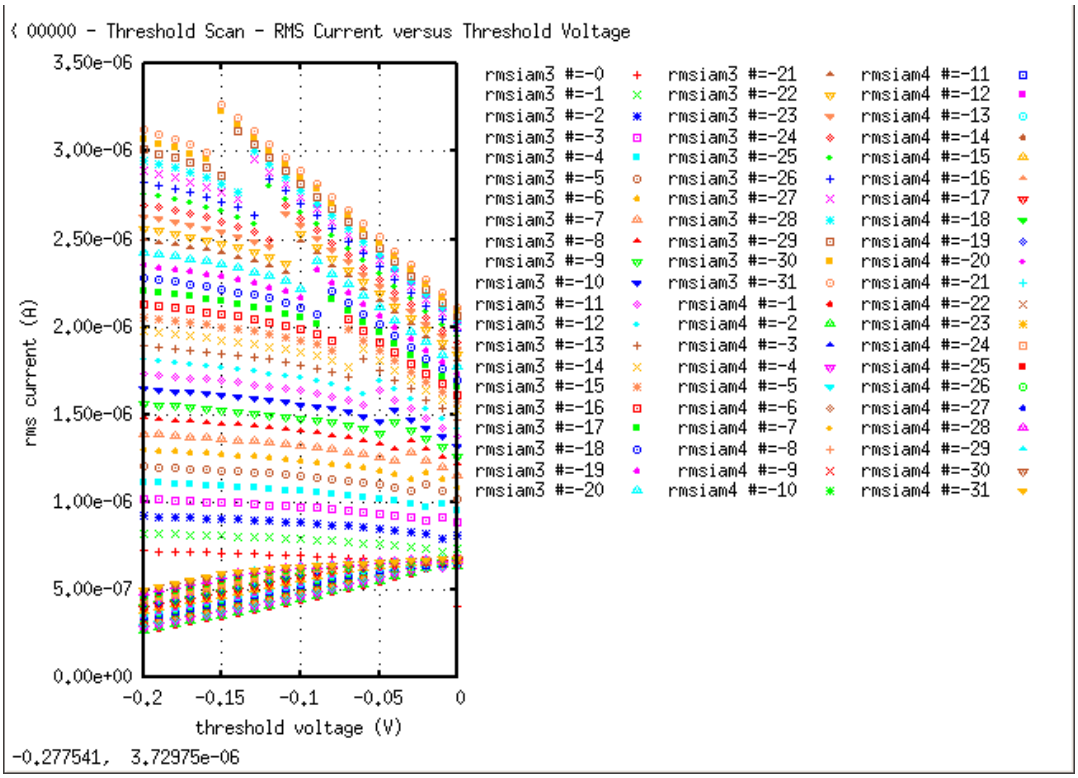


Figure 9.57: RMS Current *vs* V_{th} 6 - *scanning mode, ref.val.*

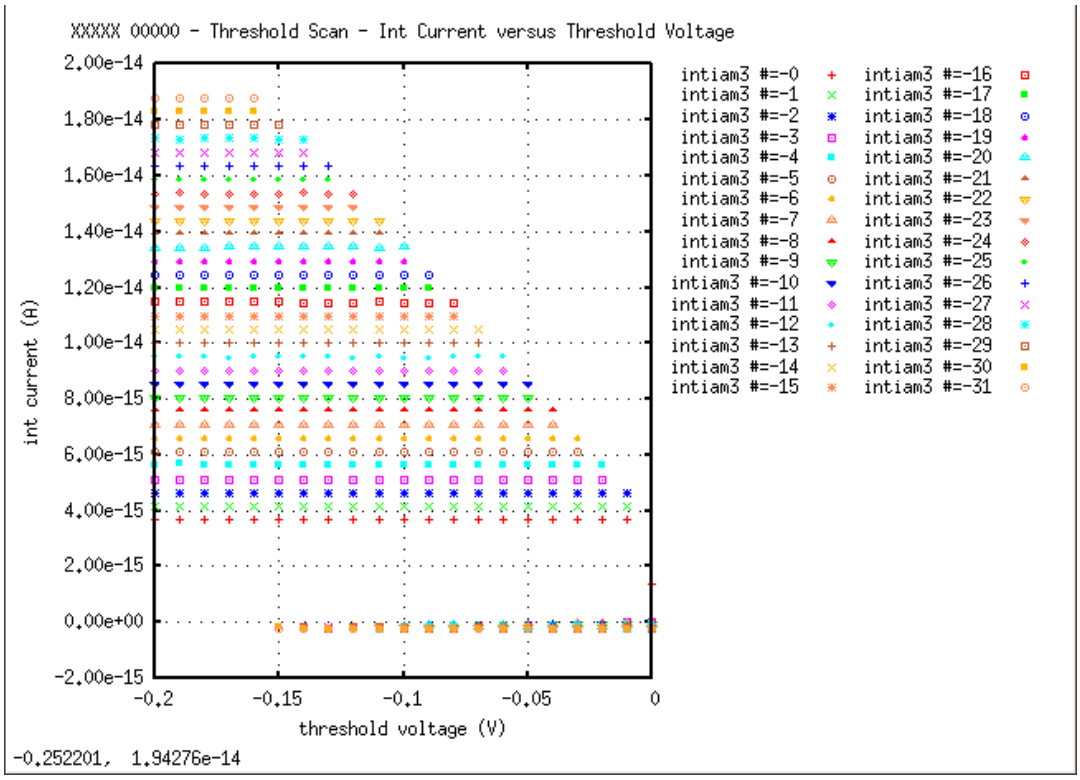


Figure 9.58: Int Current *vs* V_{th} 4 - *scanning mode, ref.val.*

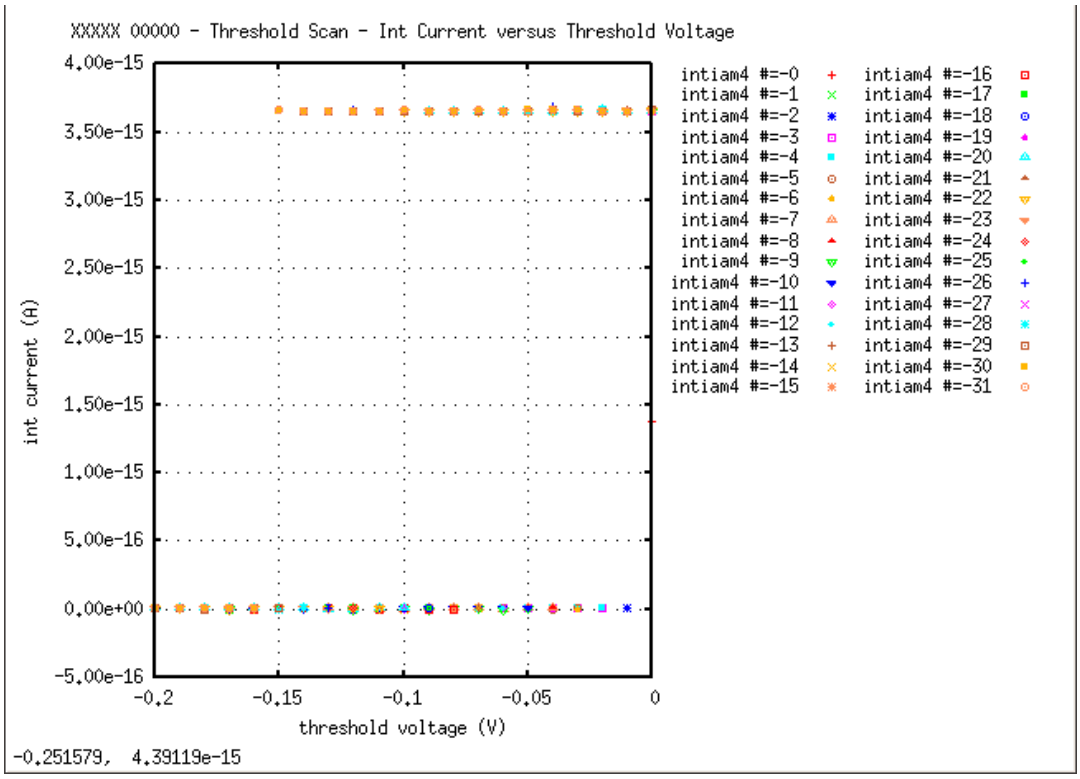


Figure 9.59: Int Current *vs* V_{th} 5 - *scanning mode, ref.val.*

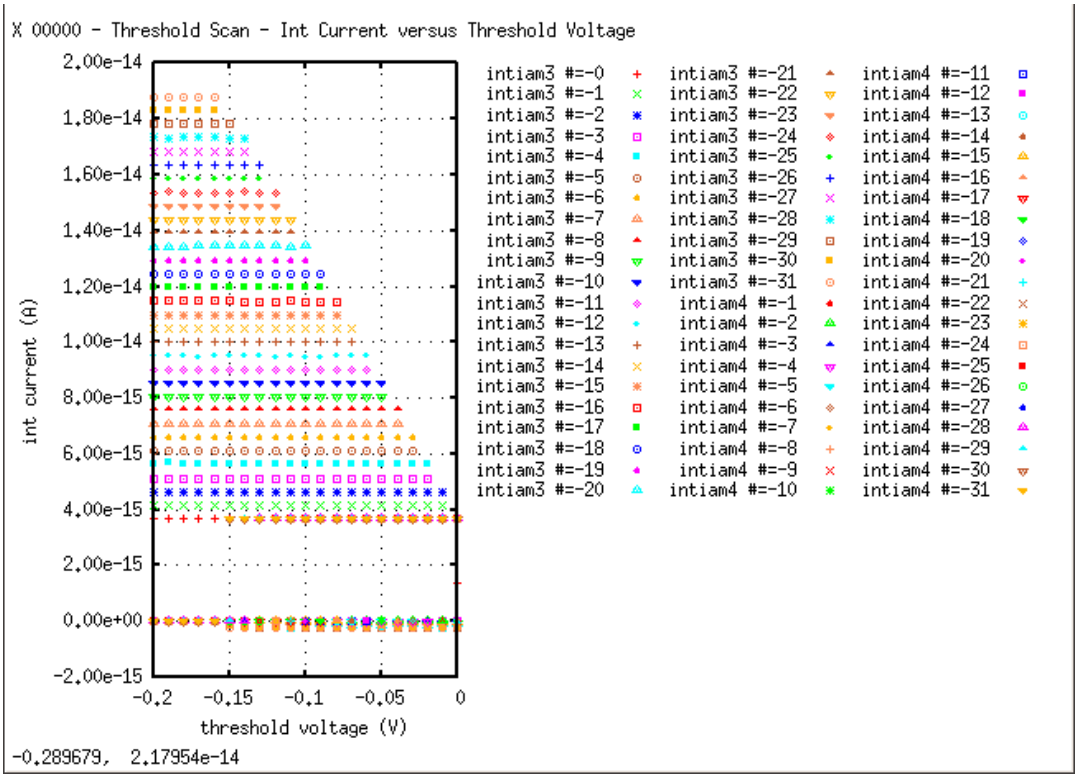


Figure 9.60: Int Current *vs* V_{th} 6 - *scanning mode, ref.val.*

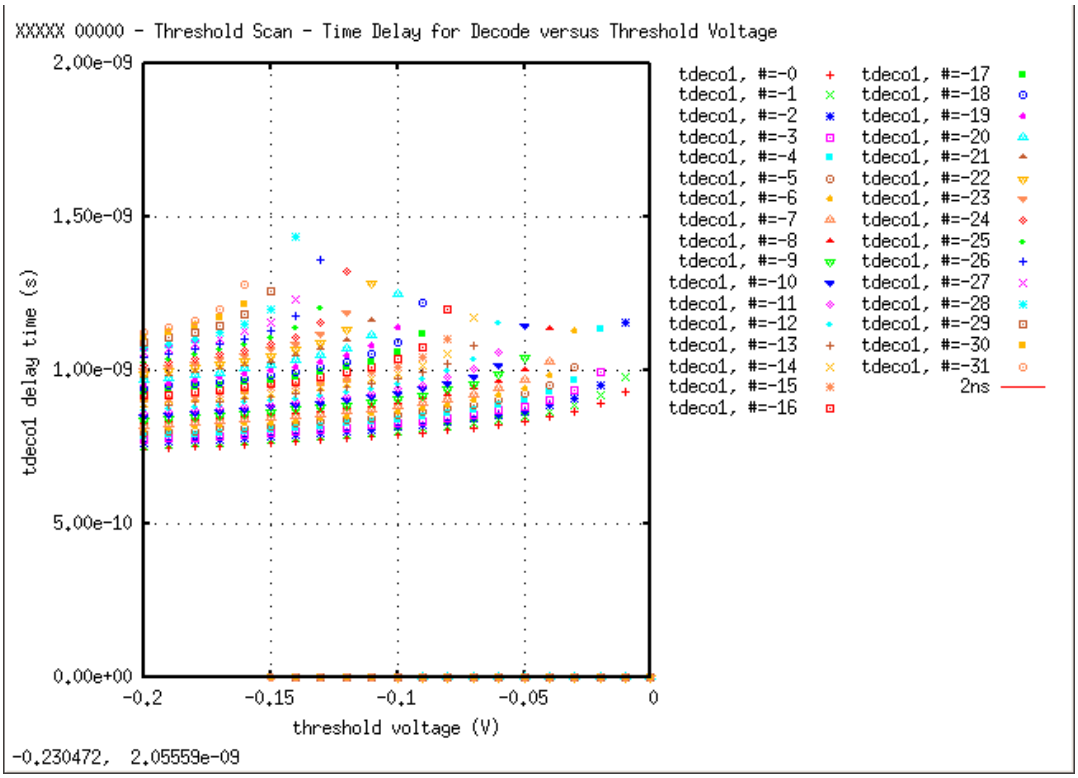


Figure 9.61: Time Delay for Decode *vs* V_{th} 1 - *scanning mode, ref.val.*

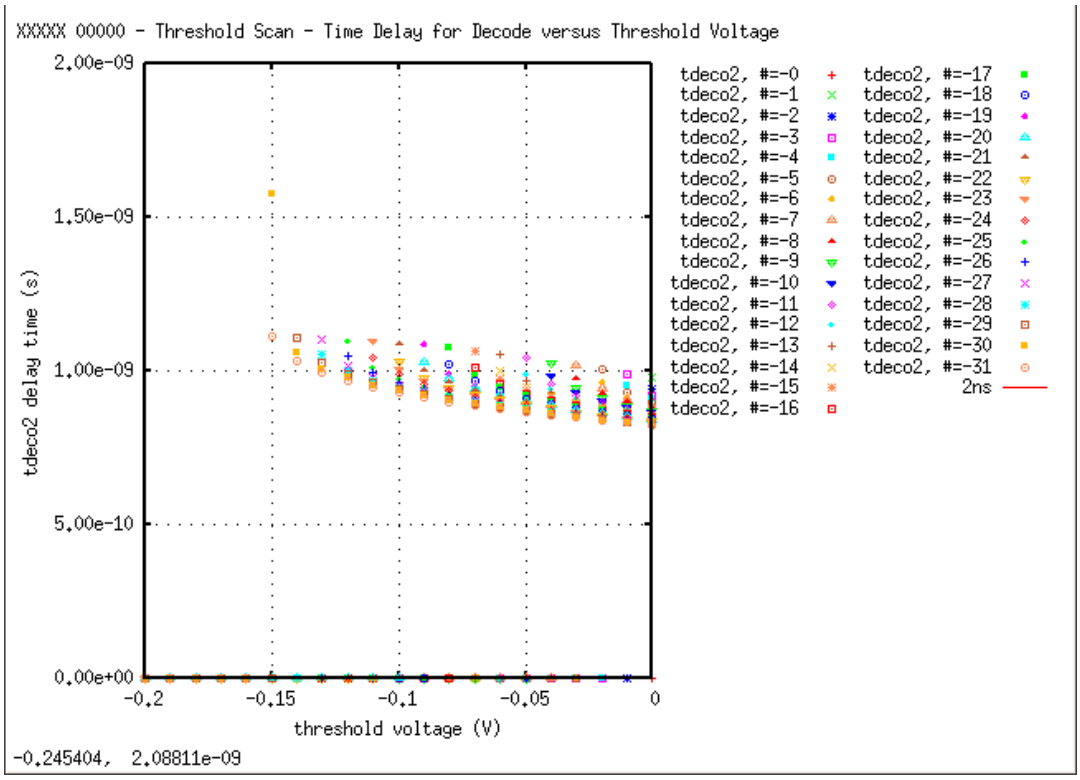


Figure 9.62: Time Delay for Decode *vs* V_{th} 2 - *scanning mode, ref.val.*

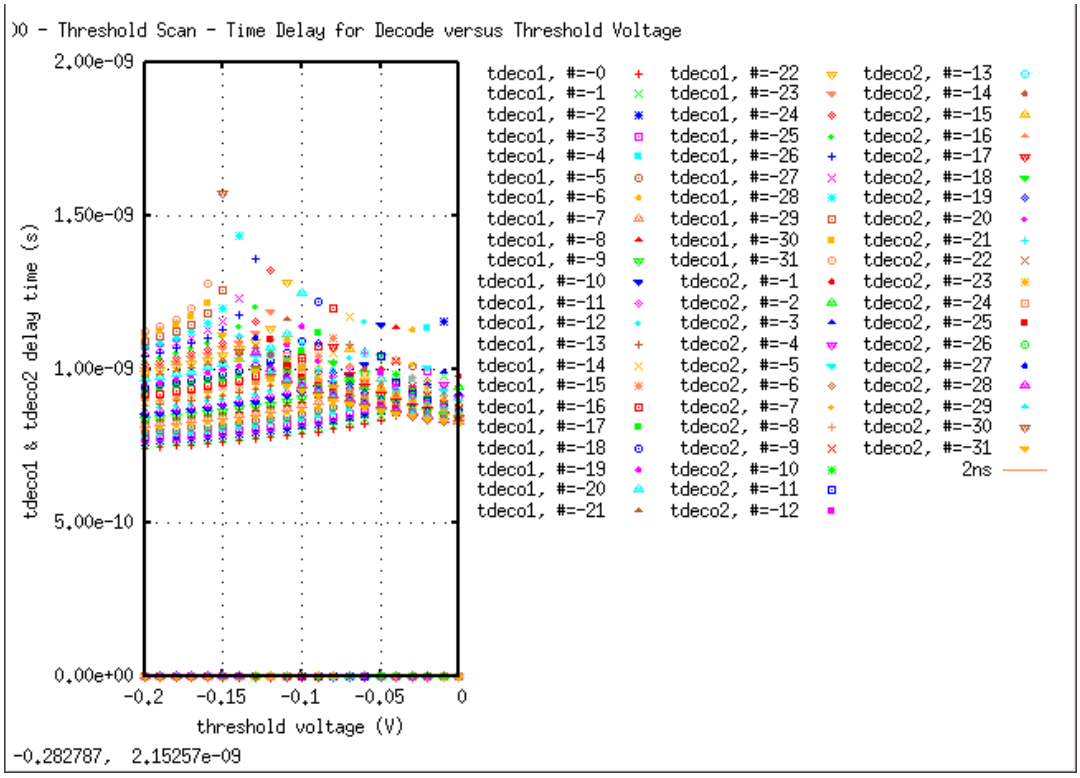


Figure 9.63: Time Delay for Decode *vs* V_{th} 3 - *scanning mode, ref.val.*

9.3 MISMATCH RESULTS – *monte carlo* SIMULATION

Plots presented in this section are generated by GnuPlot from *Monte Carlo simulation* data.

Each device size is set to the optimal value (see Table 9.1).

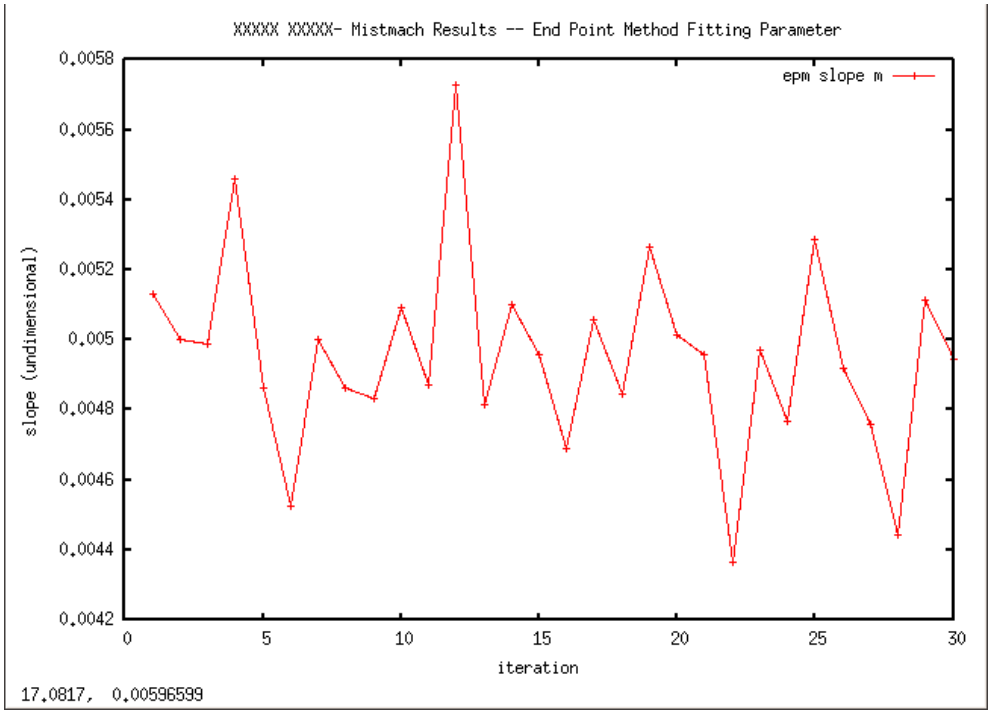


Figure 9.64: Slope - *mismatch, MonteCarlo EndPoint method*

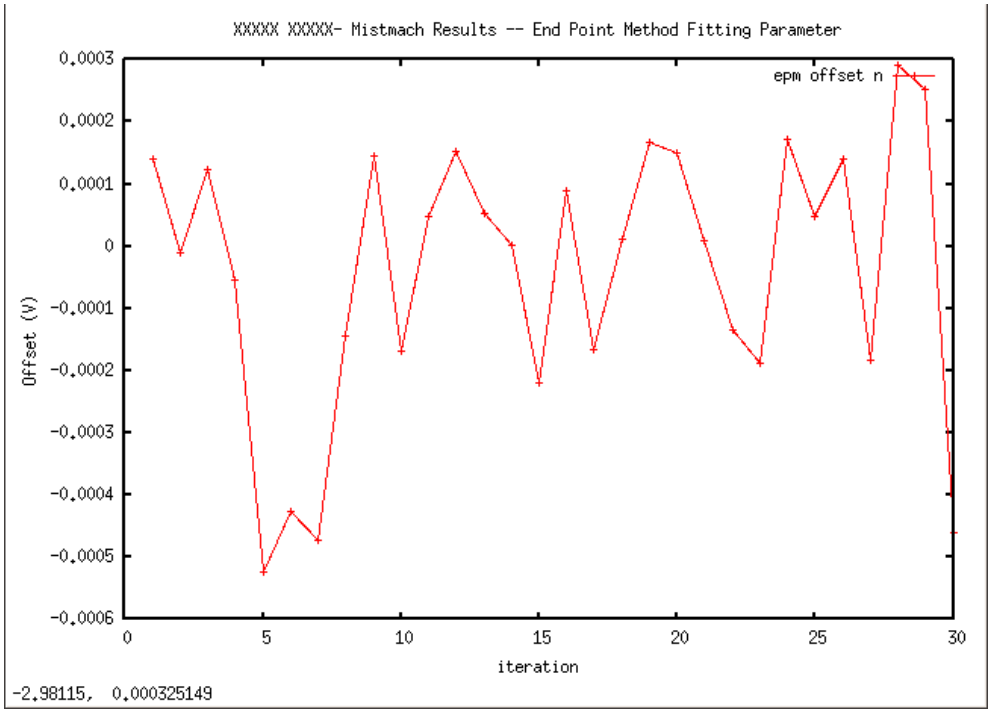


Figure 9.65: Offset - *mismatch, MonteCarlo EndPoint method*

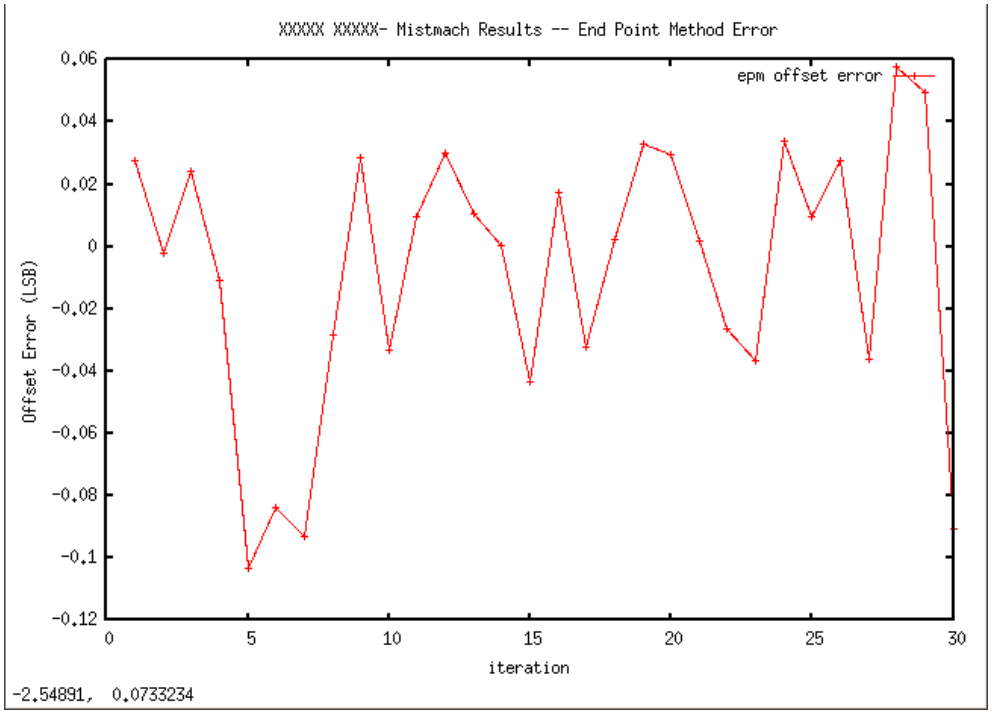


Figure 9.66: Offset Error (LSB) - *mismatch, MonteCarlo EndPoint method*

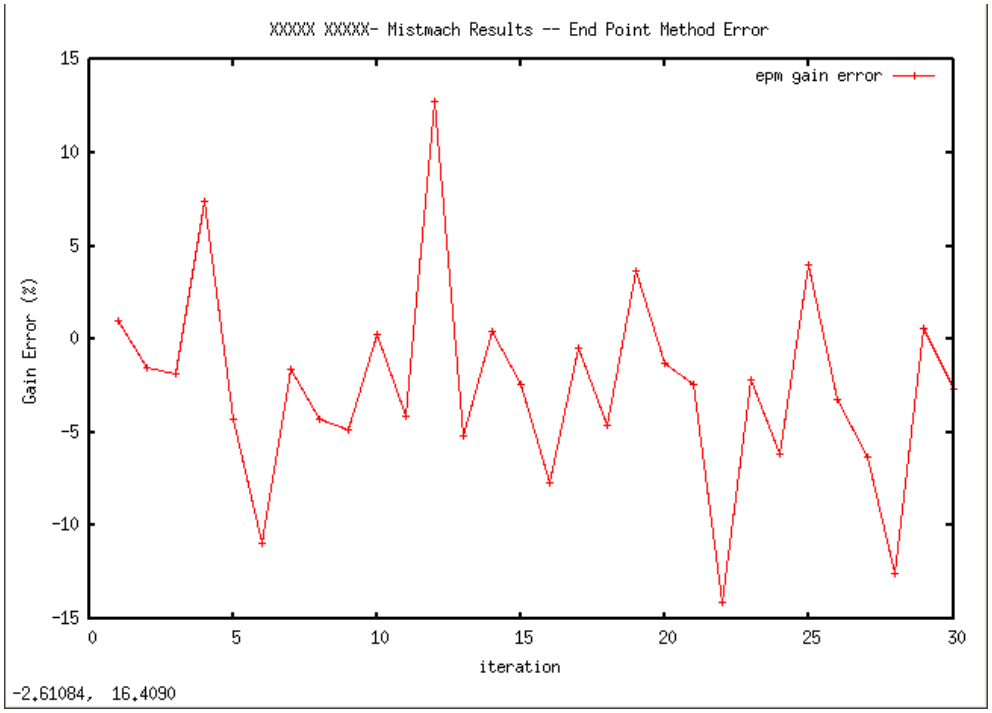


Figure 9.67: Gain Error % - *mismatch, MonteCarlo EndPoint method*

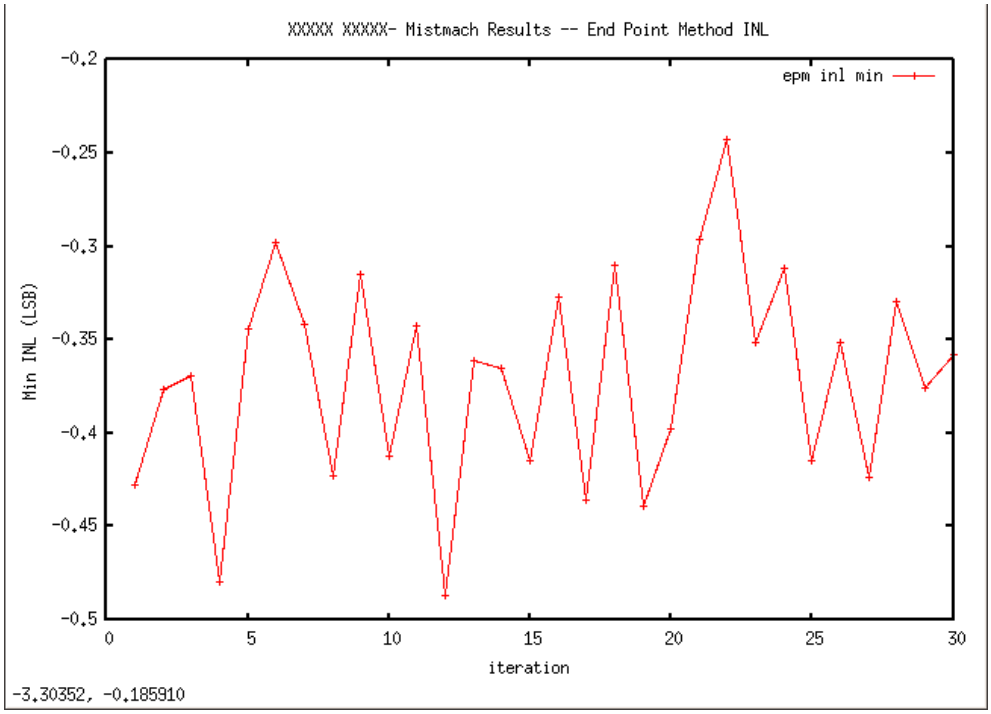


Figure 9.68: Min INL (LSB) - *mismatch, MonteCarlo EndPoint method*

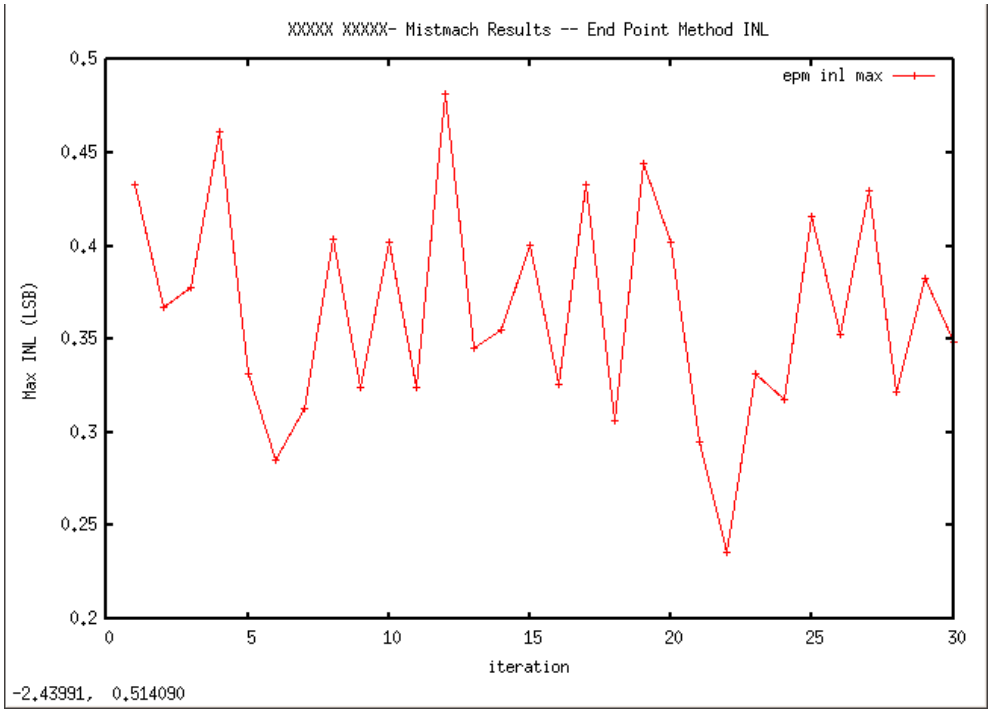


Figure 9.69: Max INL (LSB) - *mismatch, MonteCarlo EndPoint method*

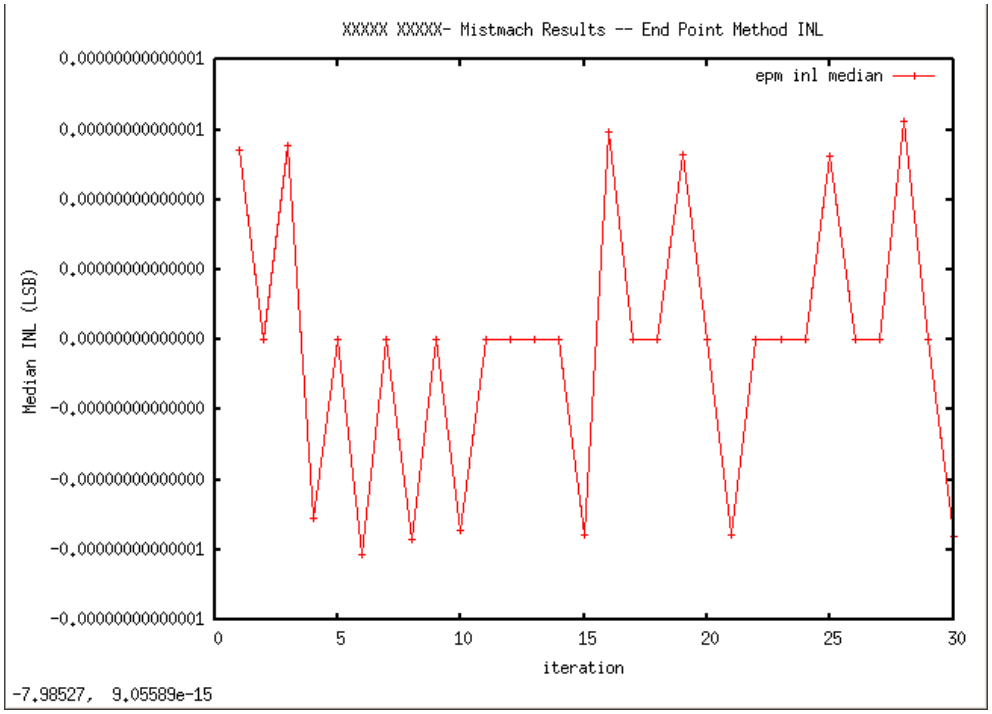


Figure 9.70: Median INL (LSB) - *mismatch, MonteCarlo EndPoint method*

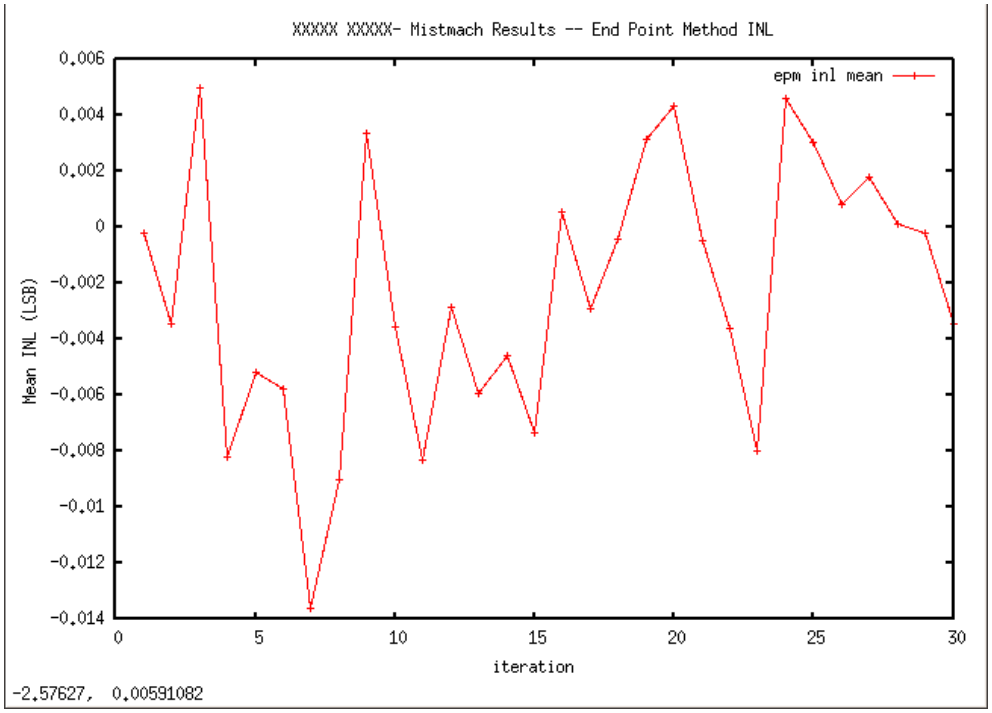


Figure 9.71: Mean INL (LSB) - *mismatch, MonteCarlo EndPoint method*

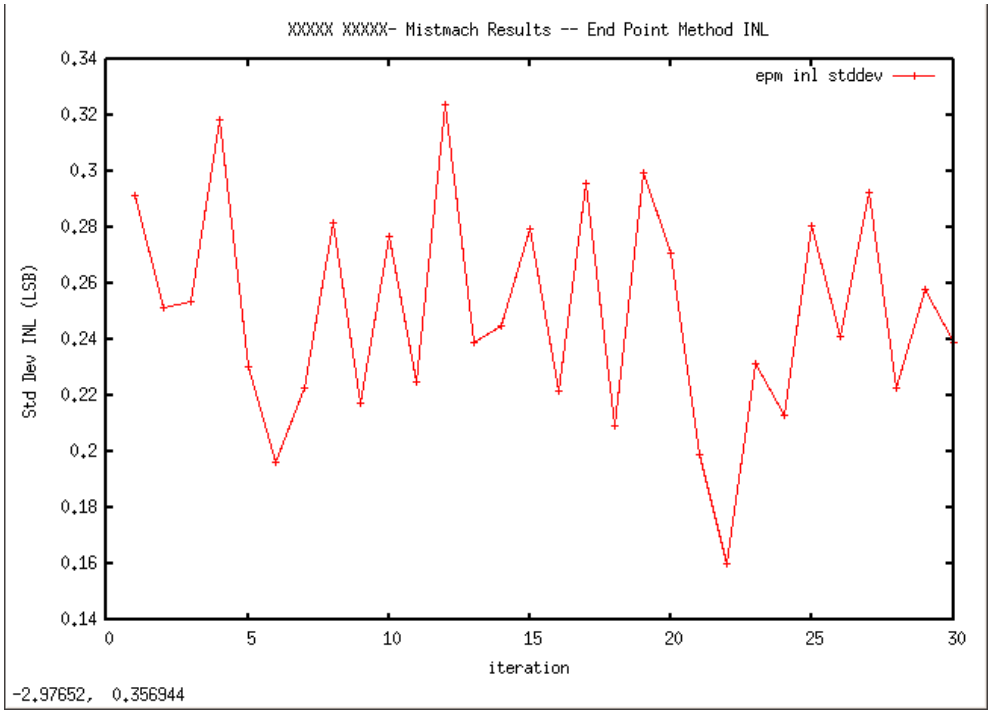


Figure 9.72: Standard Deviation INL (LSB) - *mismatch, MonteCarlo EndPoint method*

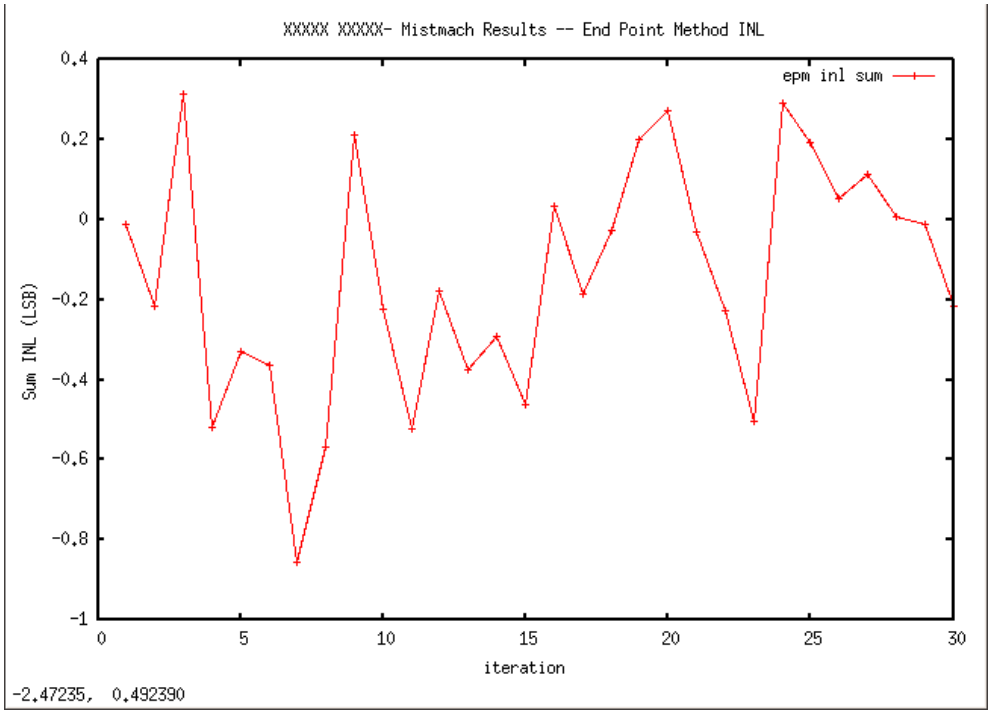


Figure 9.73: Sum INL (LSB) - *mismatch, MonteCarlo EndPoint method*

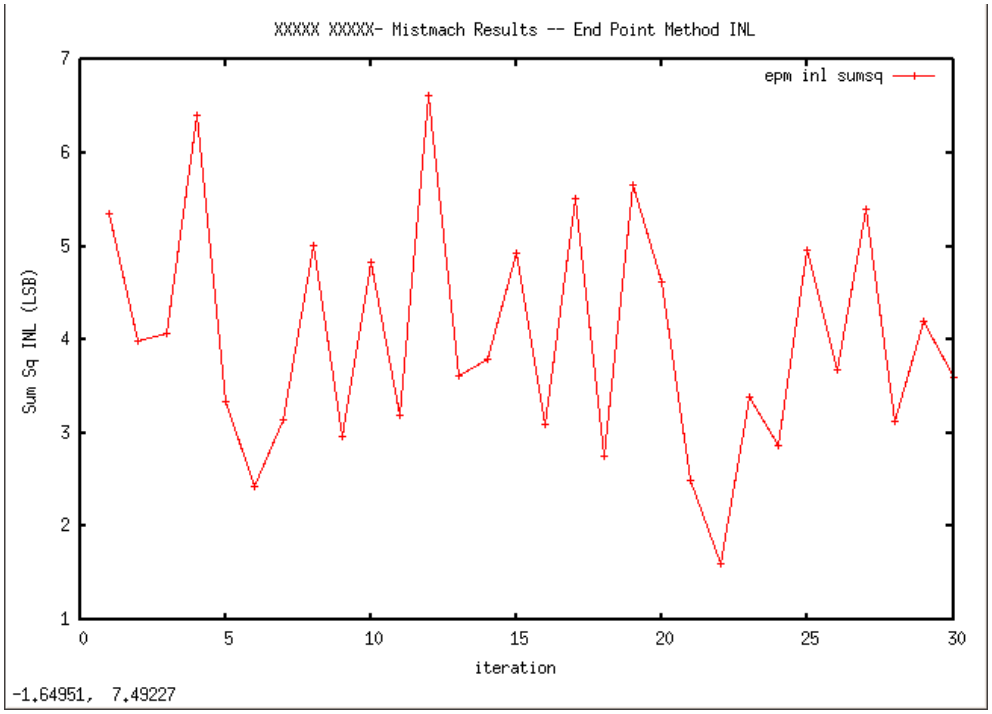


Figure 9.74: Sum Sq INL (LSB) - *mismatch, MonteCarlo EndPoint method*

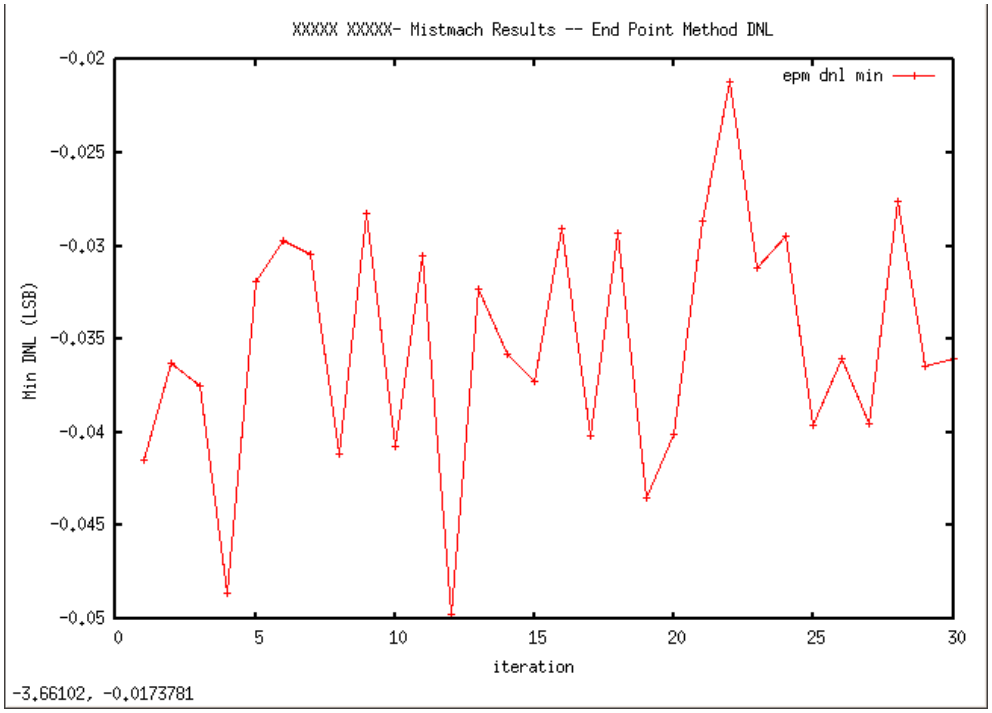


Figure 9.75: Min DNL (LSB) - *mismatch, MonteCarlo EndPoint method*

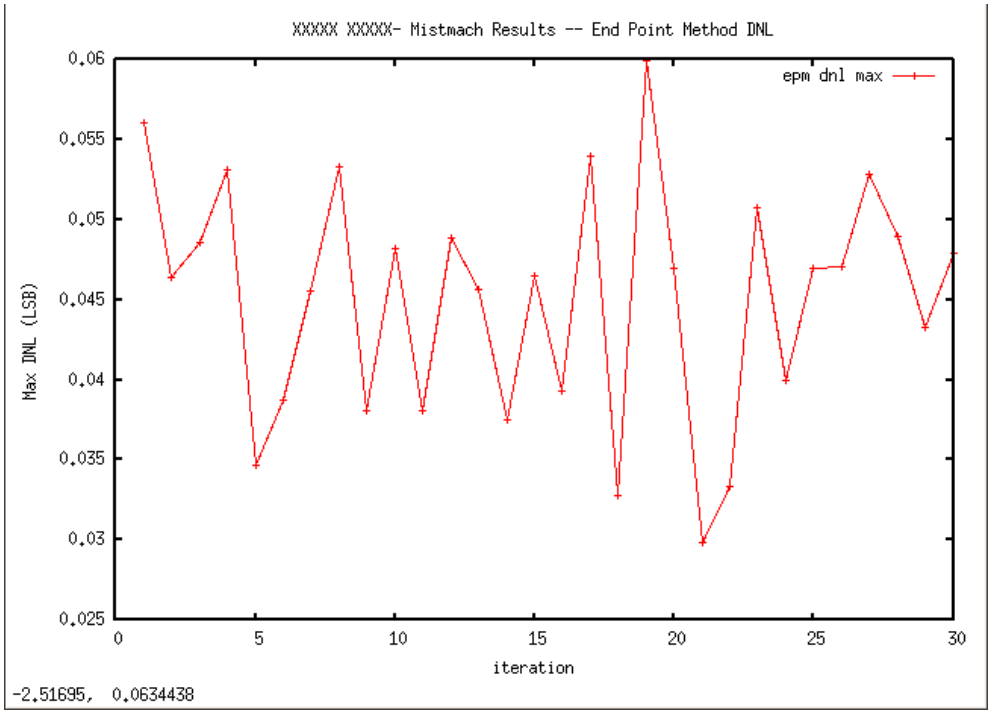


Figure 9.76: Max DNL (LSB) - *mismatch, MonteCarlo EndPoint method*

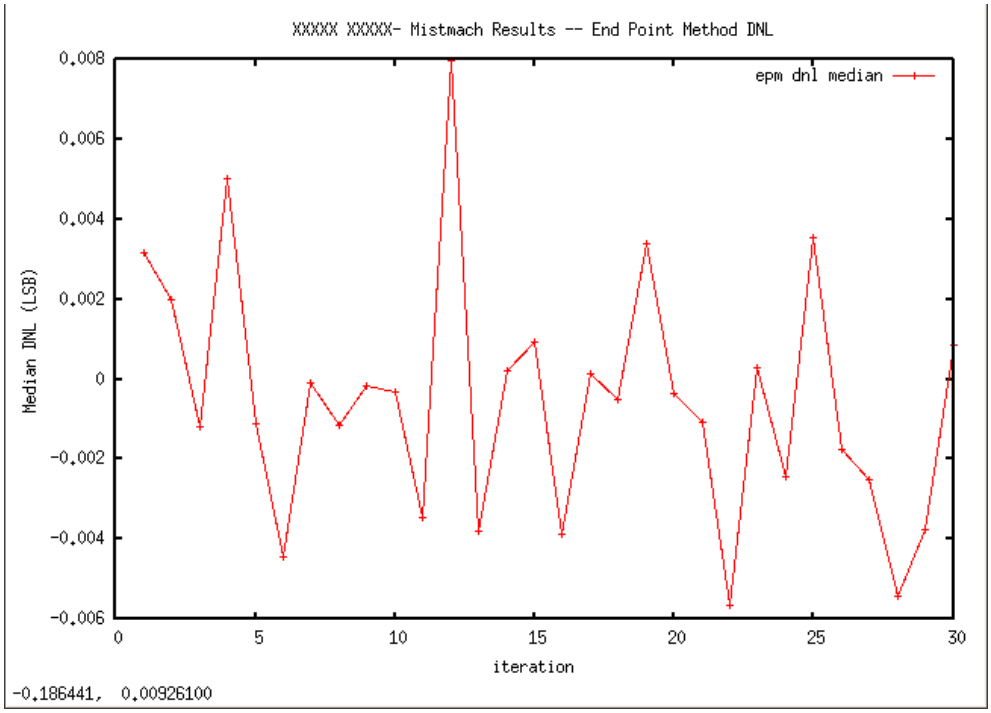


Figure 9.77: Median DNL (LSB) - *mismatch, MonteCarlo EndPoint method*

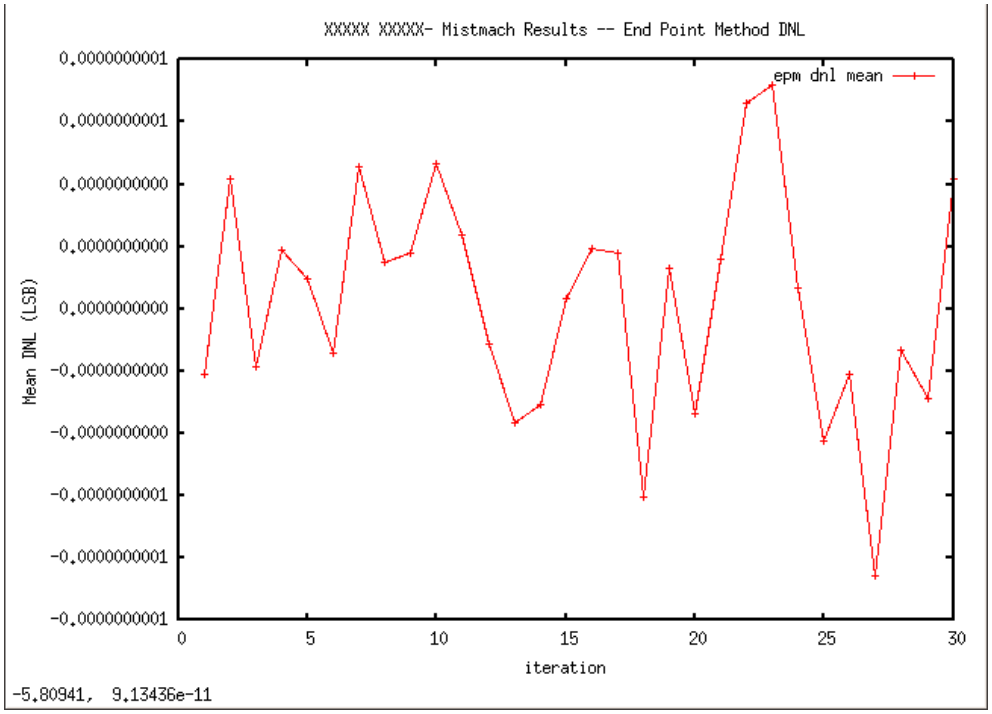


Figure 9.78: Mean DNL (LSB) - *mismatch, MonteCarlo EndPoint method*

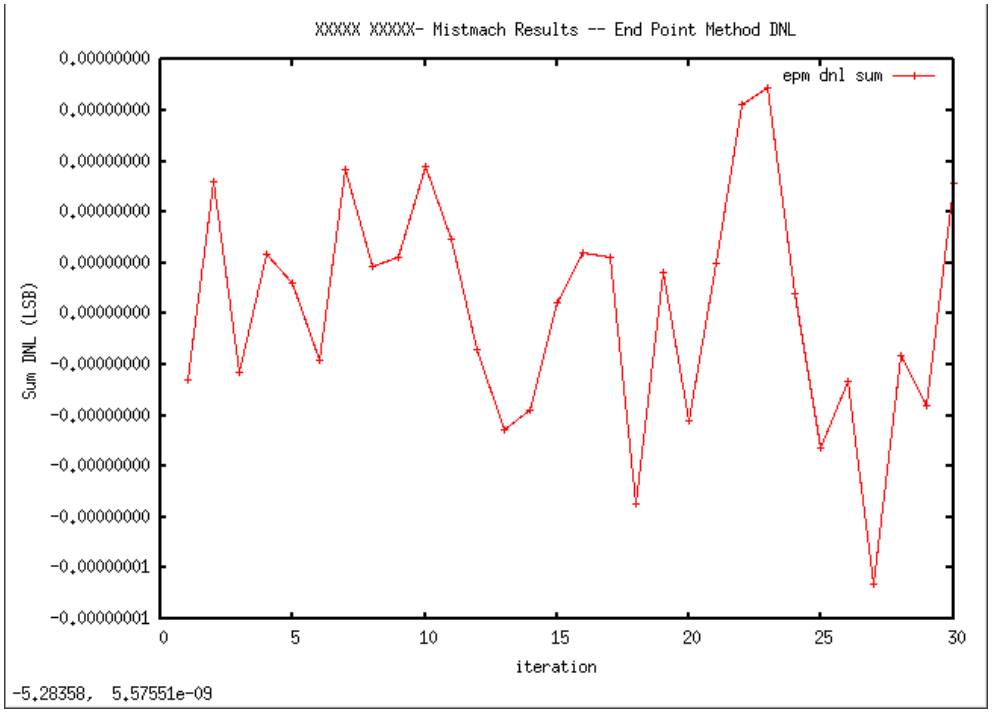


Figure 9.79: Sum DNL (LSB) - *mismatch, MonteCarlo EndPoint method*

Figure 9.80: Slope - *mismatch, MonteCarlo LSE method*

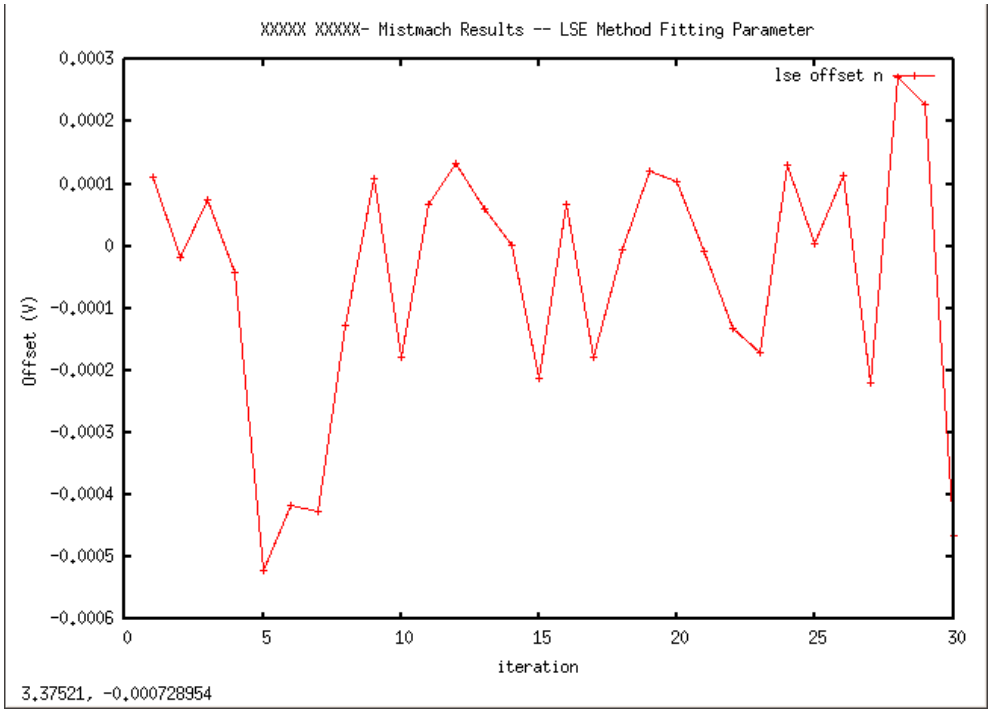


Figure 9.81: Offset - *mismatch, MonteCarlo LSE method*

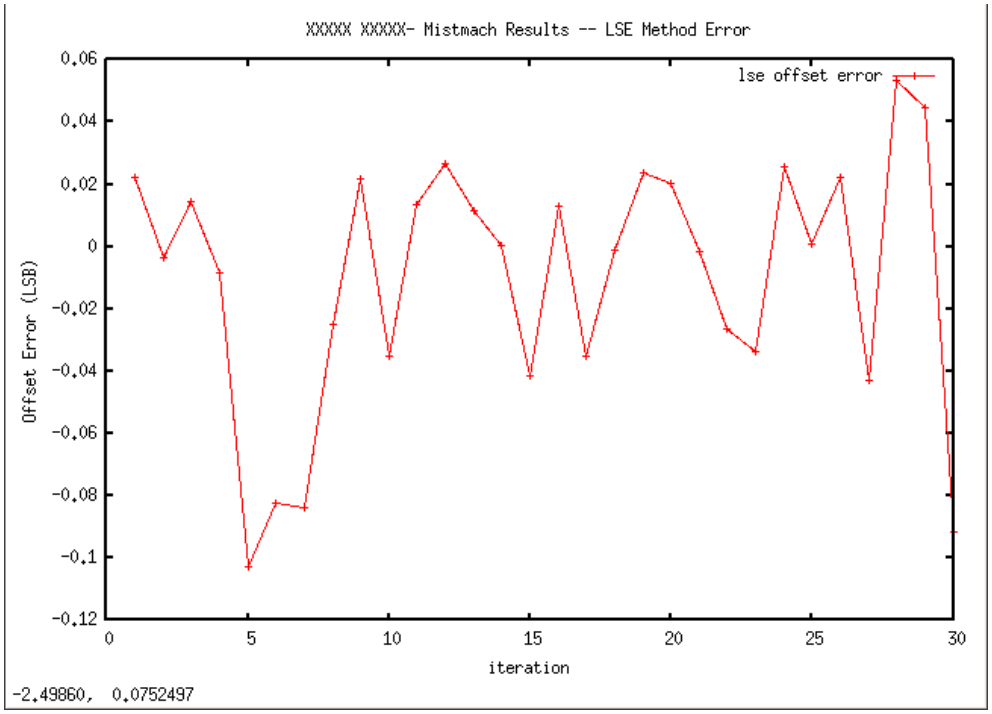


Figure 9.82: Offset Error (LSB) - *mismatch, MonteCarlo LSE method*

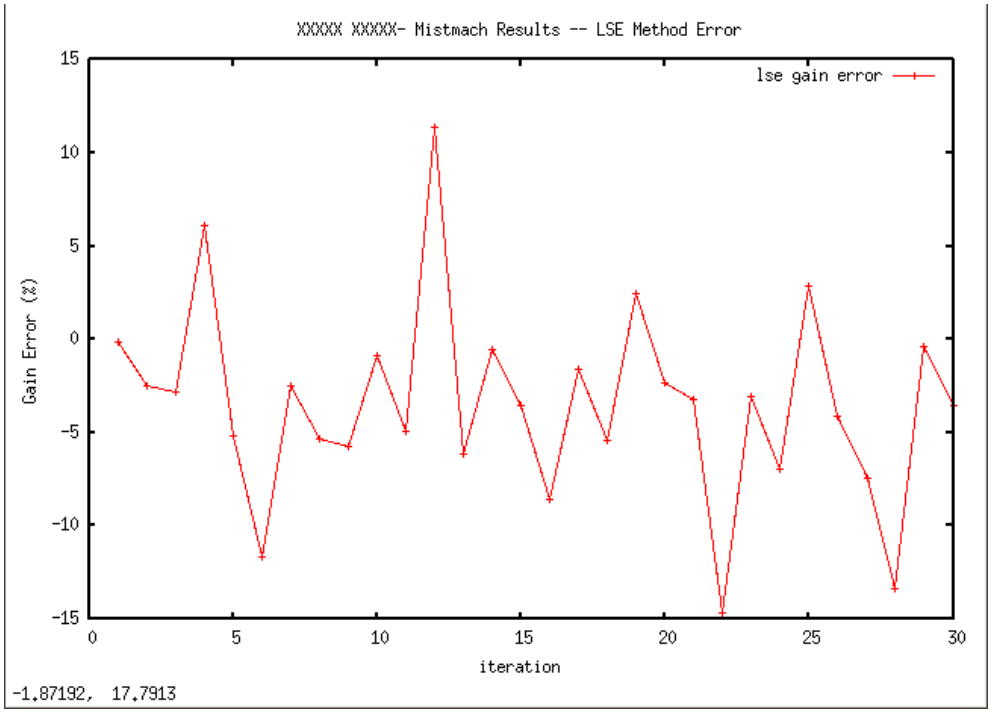


Figure 9.83: Gain Error % - *mismatch, MonteCarlo LSE method*

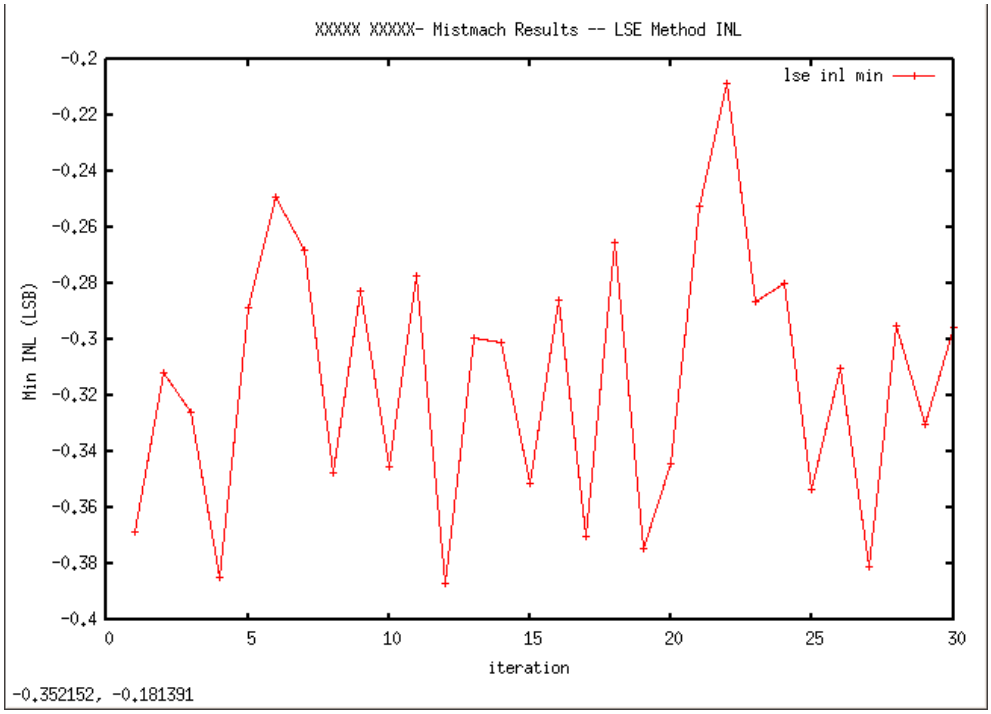


Figure 9.84: Min INL (LSB) - *mismatch, MonteCarlo LSE method*

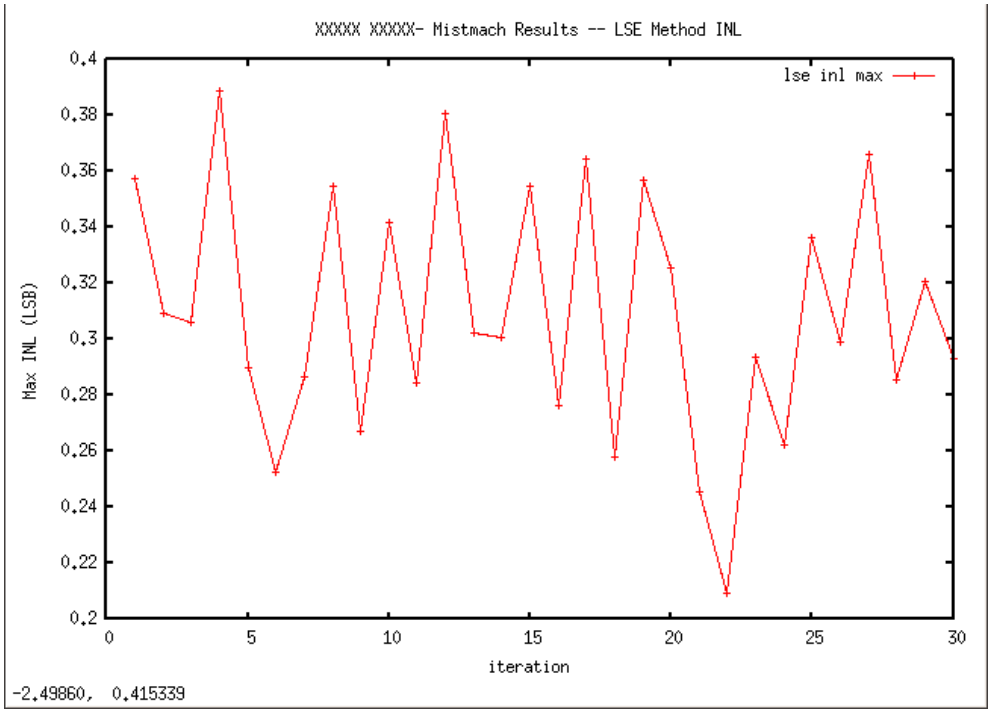


Figure 9.85: Max INL (LSB) - *mismatch, MonteCarlo LSE method*

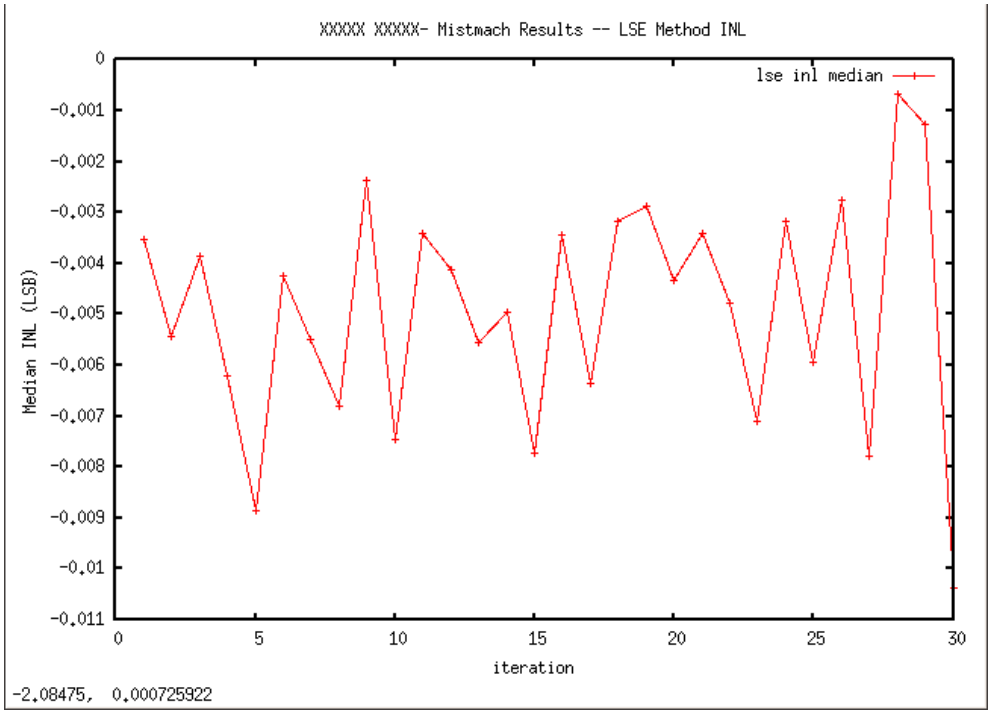


Figure 9.86: Median INL (LSB) - *mismatch, MonteCarlo LSE method*

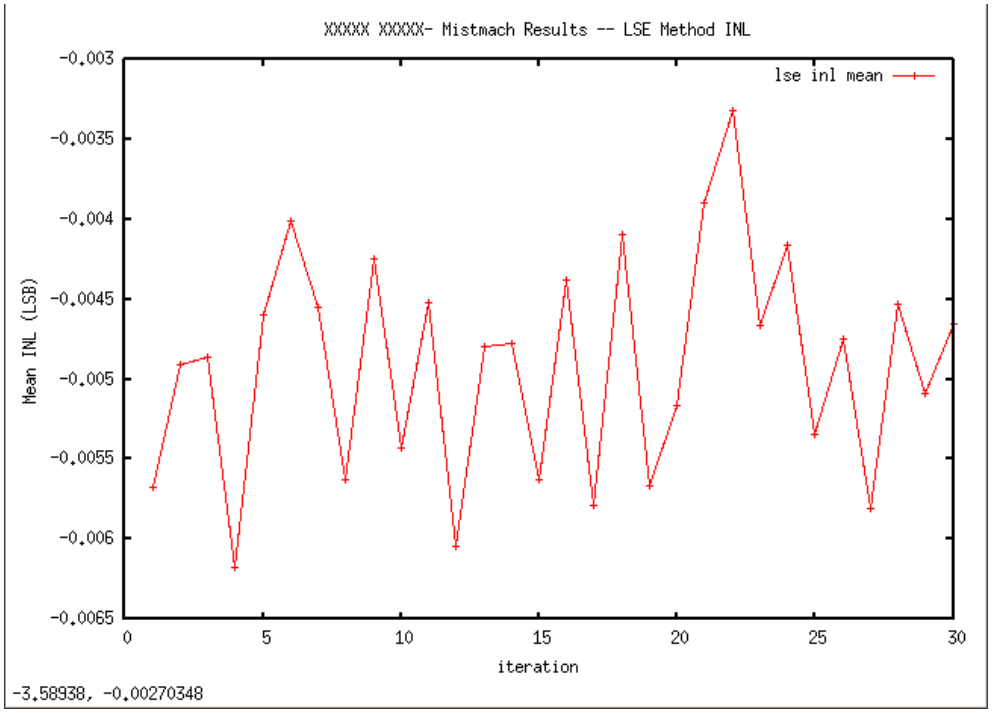


Figure 9.87: Mean INL (LSB) - *mismatch, MonteCarlo LSE method*

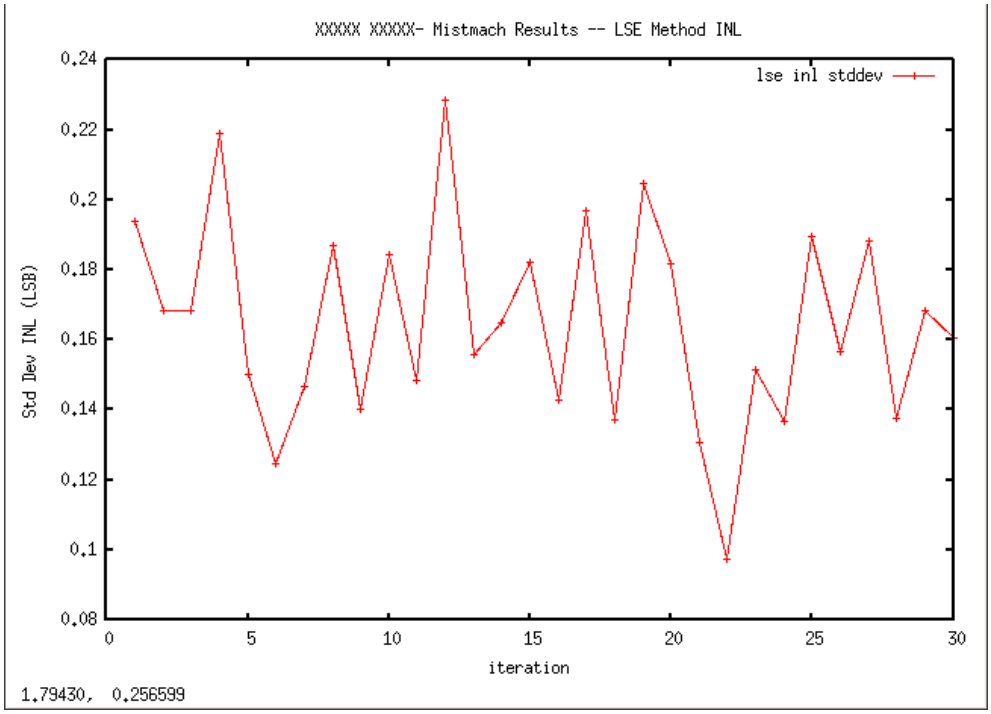


Figure 9.88: Standard Deviation INL (LSB) - *mismatch, MonteCarlo LSE method*

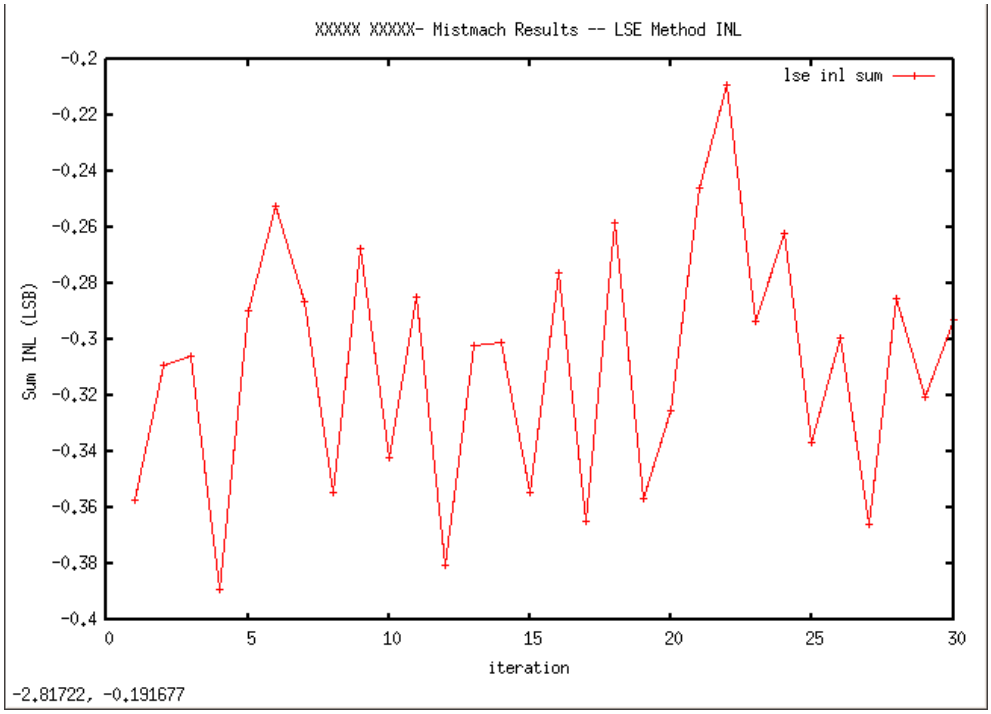


Figure 9.89: Sum INL (LSB) - *mismatch, MonteCarlo LSE method*

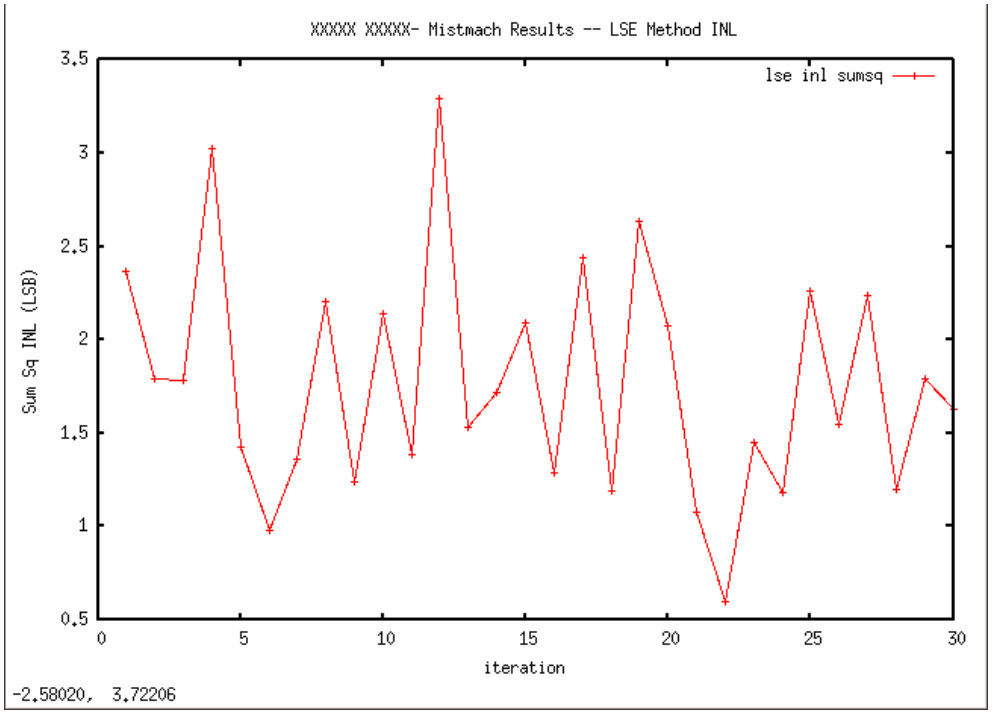


Figure 9.90: Sum Sq INL (LSB) - *mismatch, MonteCarlo LSE method*

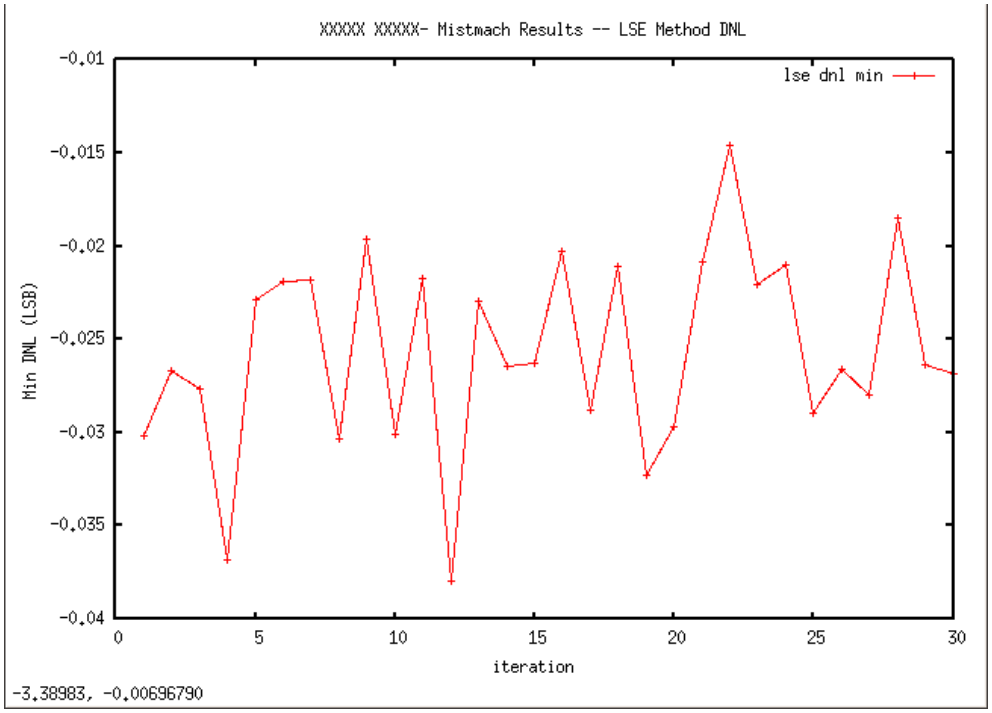


Figure 9.91: Min DNL (LSB) - *mismatch, MonteCarlo LSE method*

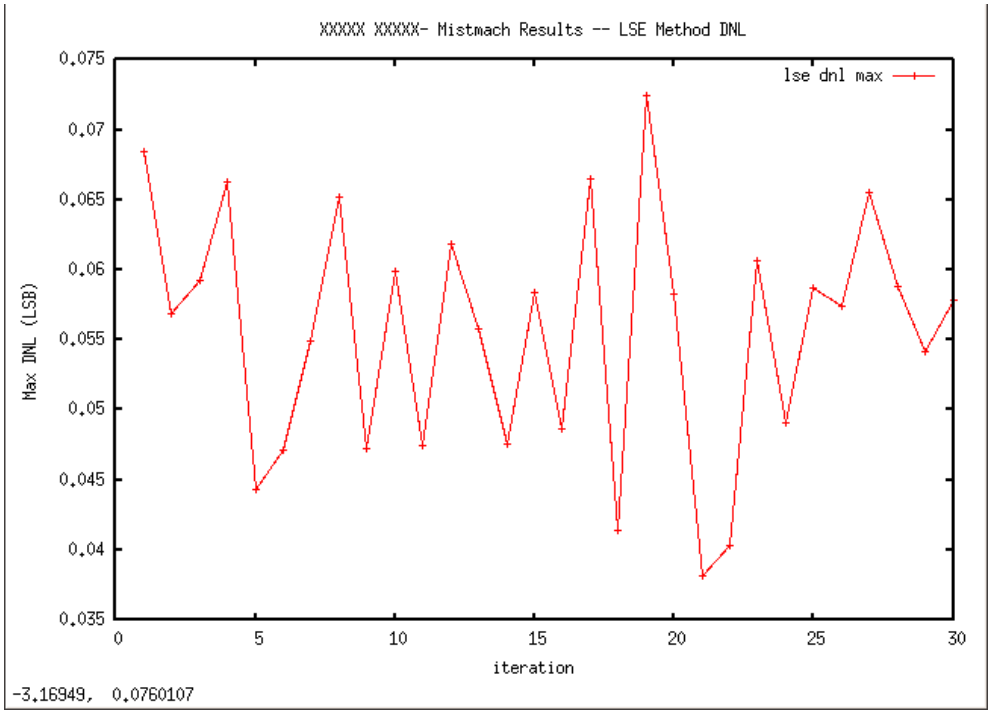


Figure 9.92: Max DNL (LSB) - *mismatch, MonteCarlo LSE method*

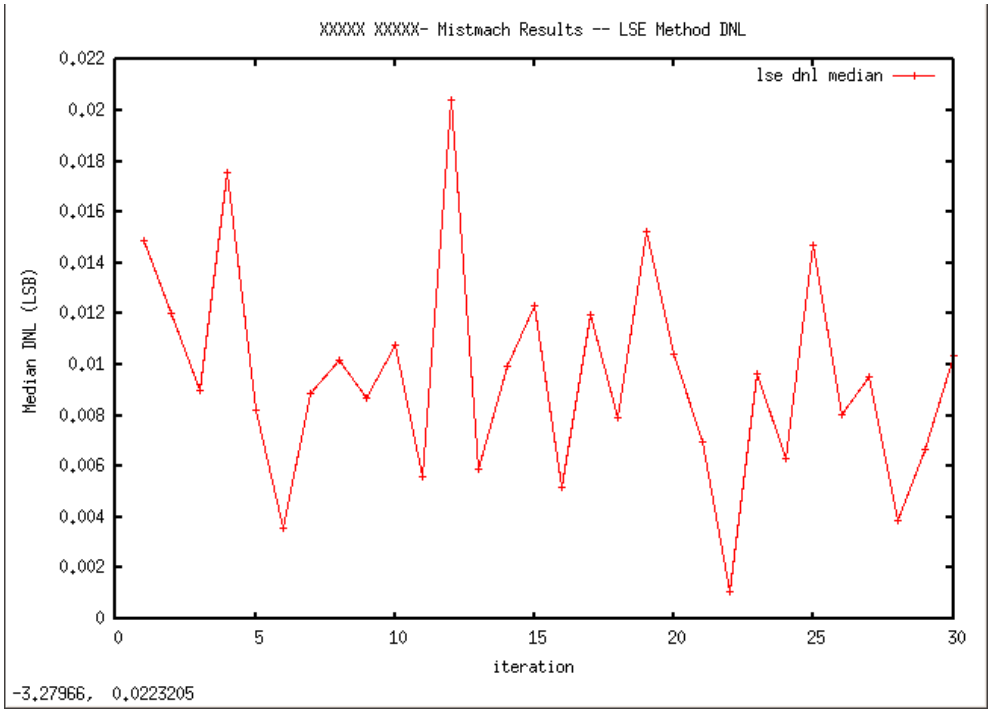


Figure 9.93: Median DNL (LSB) - *mismatch, MonteCarlo LSE method*

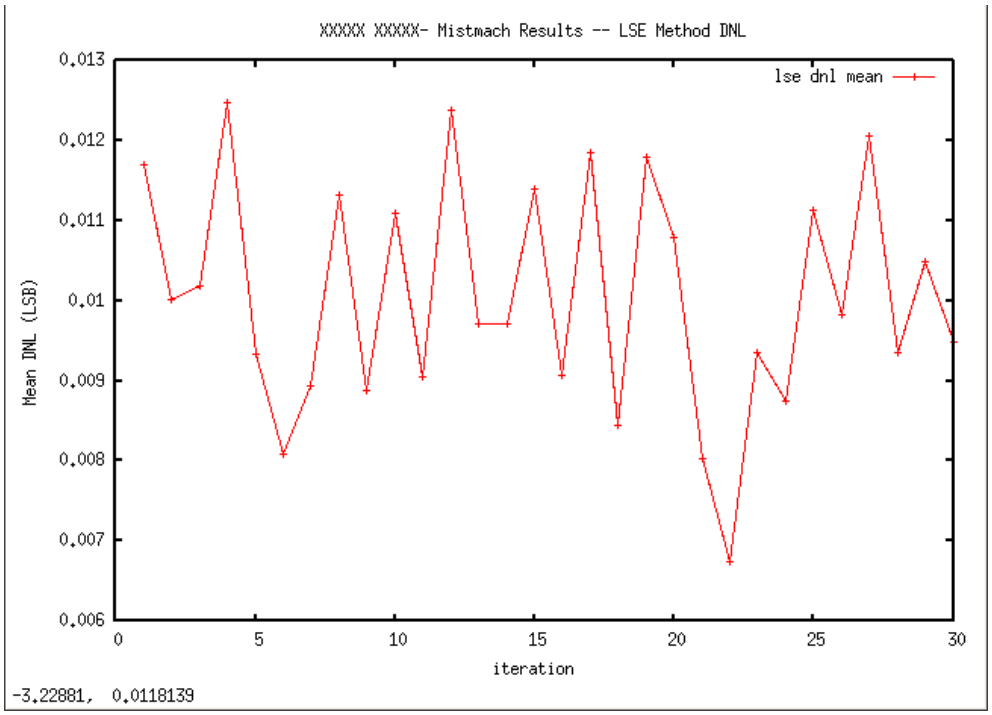


Figure 9.94: Mean DNL (LSB) - *mismatch, MonteCarlo LSE method*

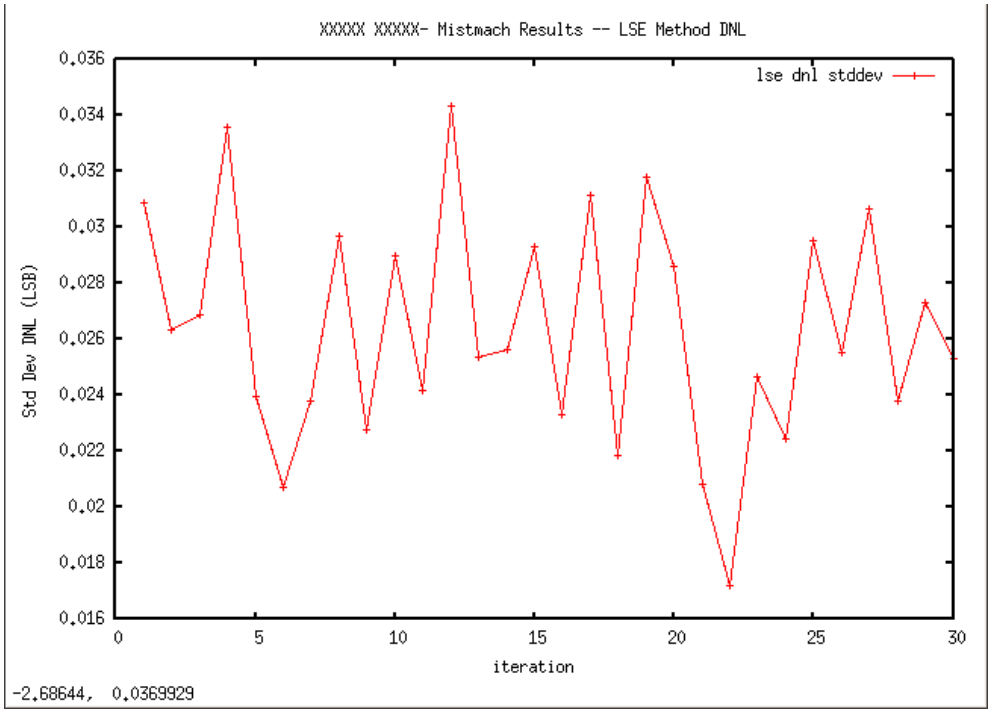


Figure 9.95: Standard Deviation DNL (LSB) - *mismatch, MonteCarlo LSE method*

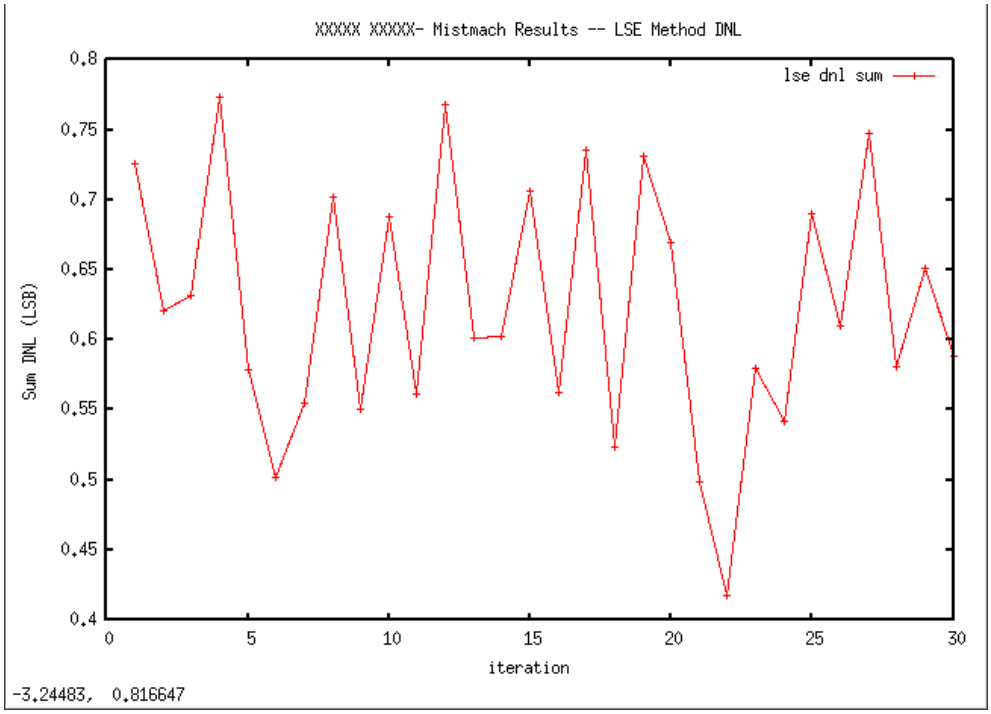


Figure 9.96: Sum DNL (LSB) - *mismatch, MonteCarlo LSE method*

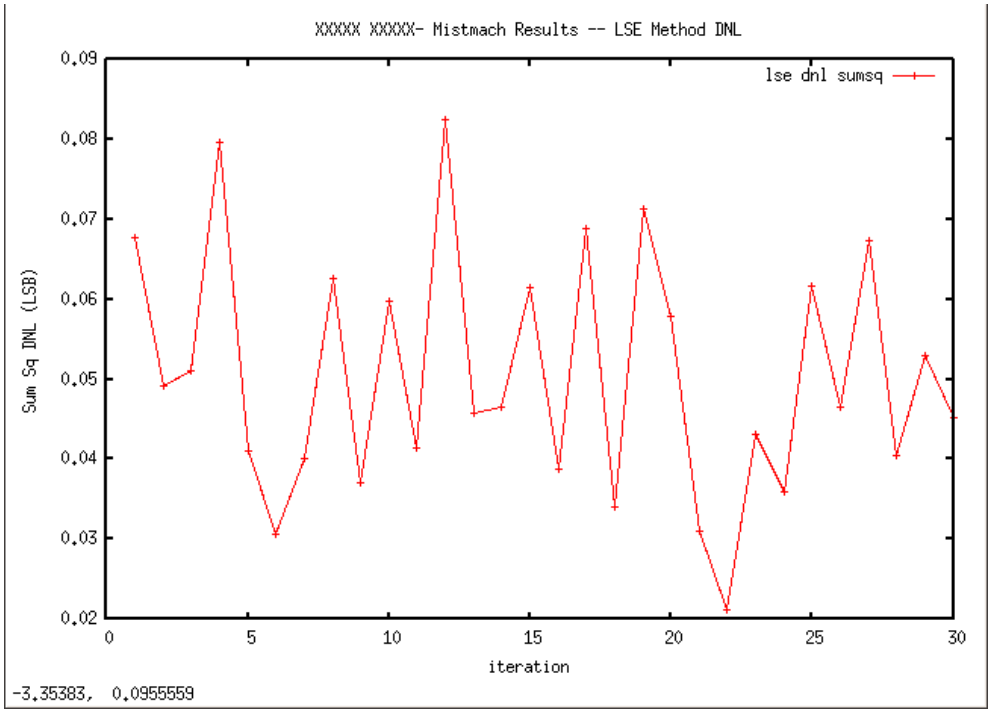


Figure 9.97: Sum Sq DNL (LSB) - *mismatch, MonteCarlo LSE method*

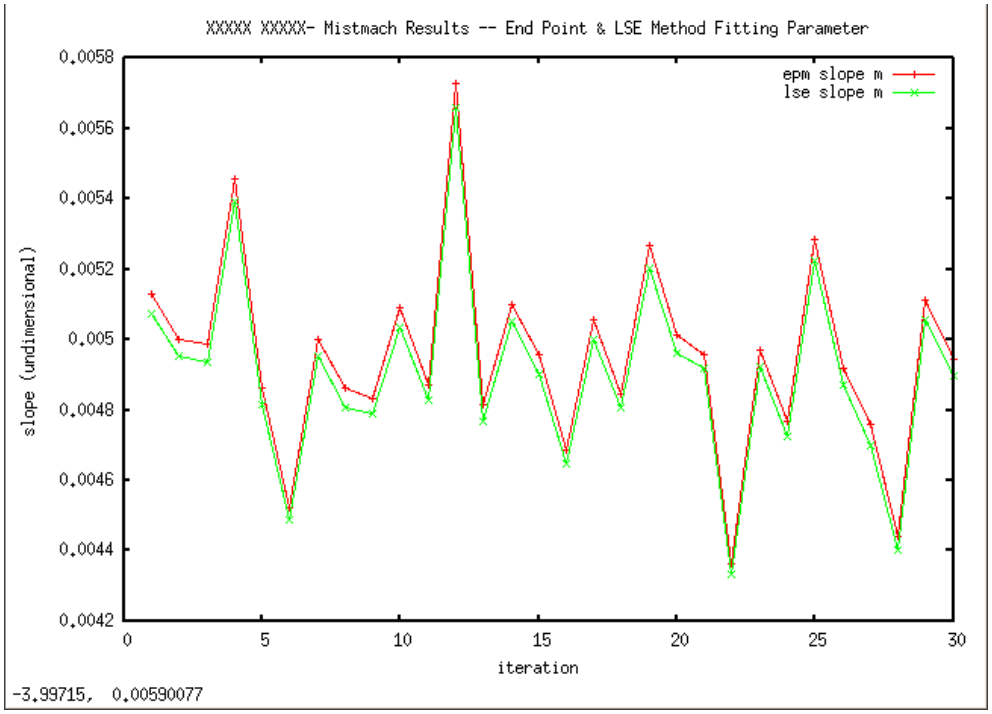


Figure 9.98: Slope - *mismatch, MonteCarlo EndPoint&LSE method*

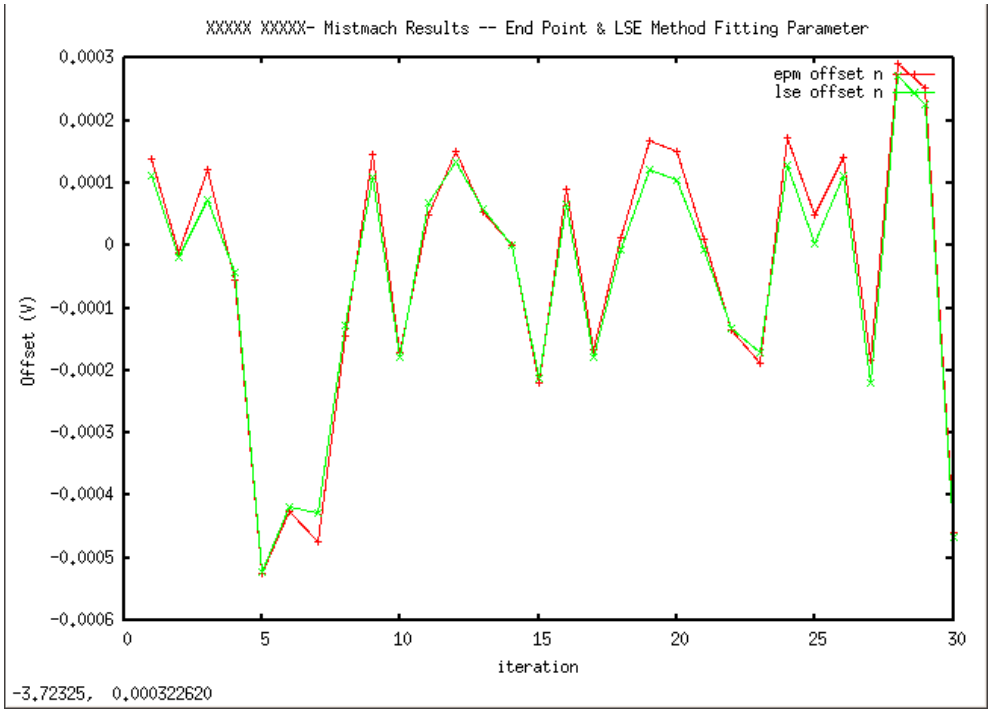


Figure 9.99: Offset - *mismatch, MonteCarlo EndPoint&LSE method*

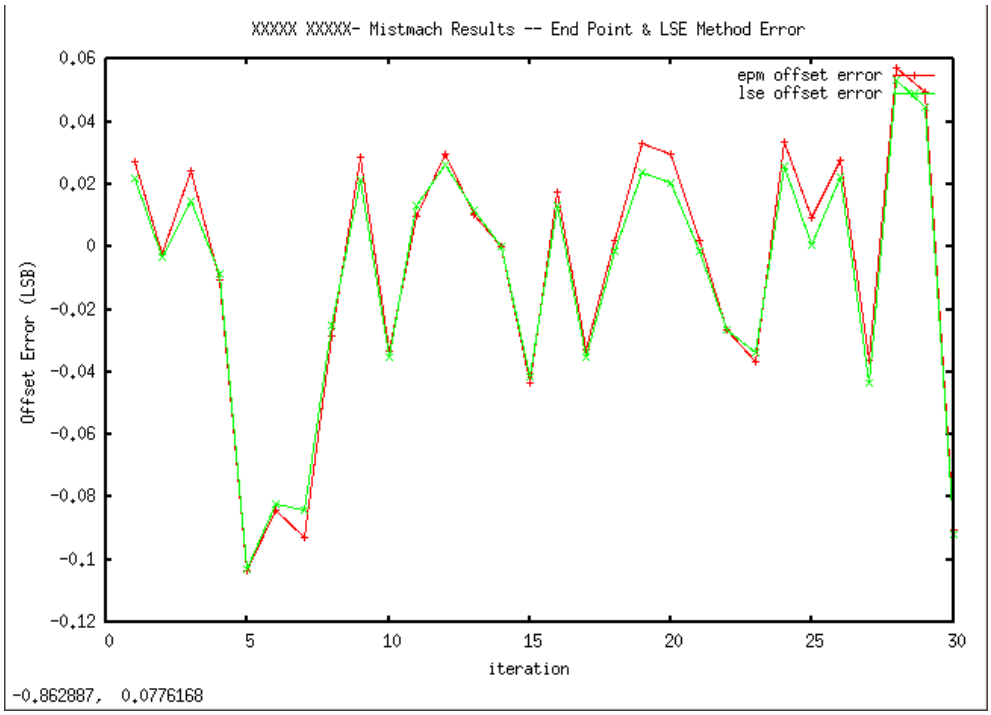


Figure 9.100: Offset Error (LSB) - *mismatch, MonteCarlo EndPoint&LSE method*

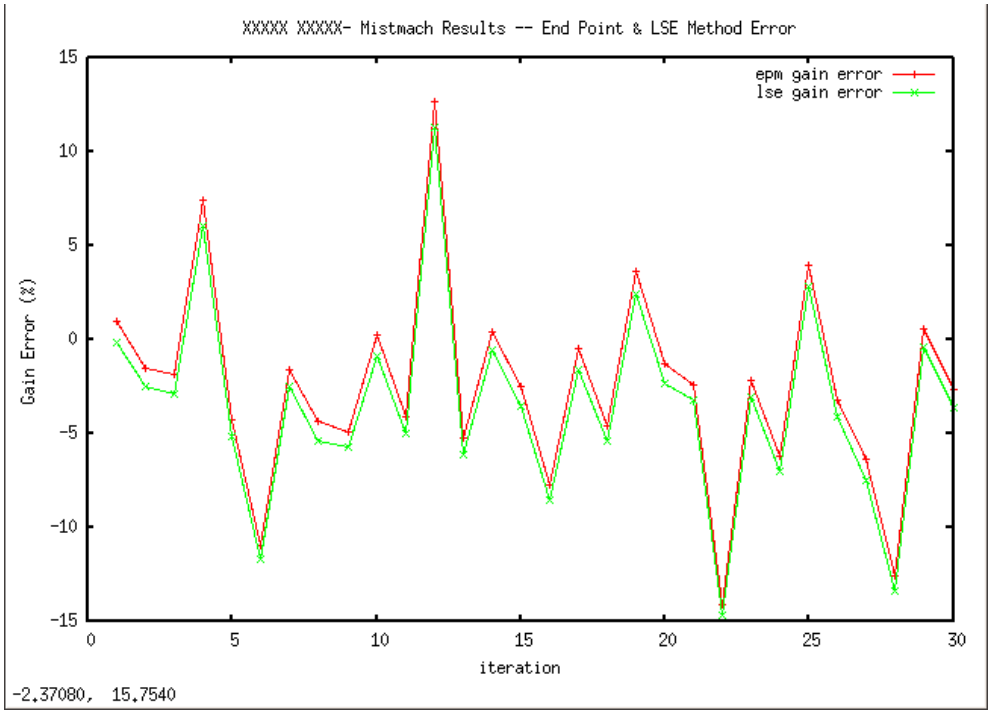


Figure 9.101: Gain Error % - *mismatch, MonteCarlo EndPoint&LSE method*

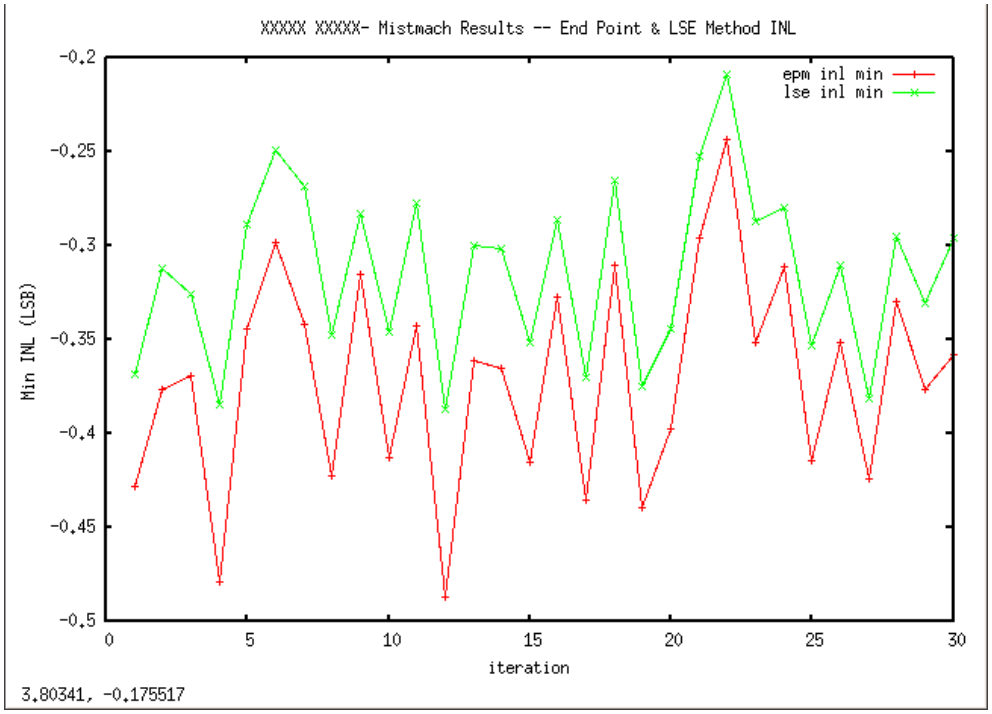


Figure 9.102: Min INL (LSB) - *mismatch, MonteCarlo EndPoint&LSE method*

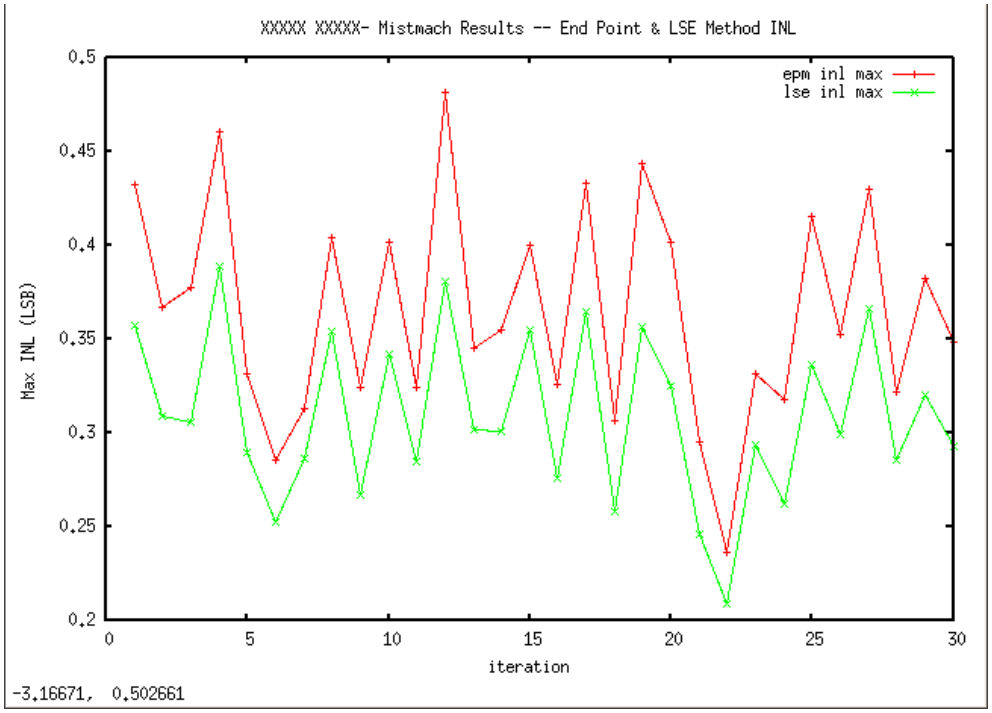


Figure 9.103: Max INL (LSB) - *mismatch, MonteCarlo EndPoint&LSE method*

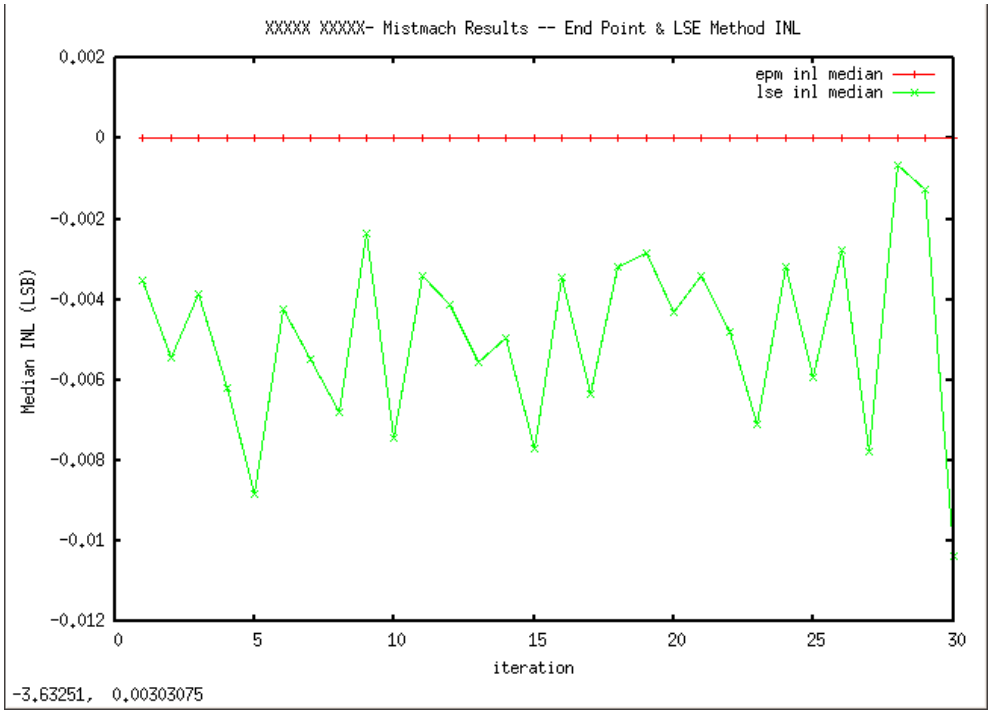


Figure 9.104: Median INL (LSB) - *mismatch, MonteCarlo EndPoint&LSE method*

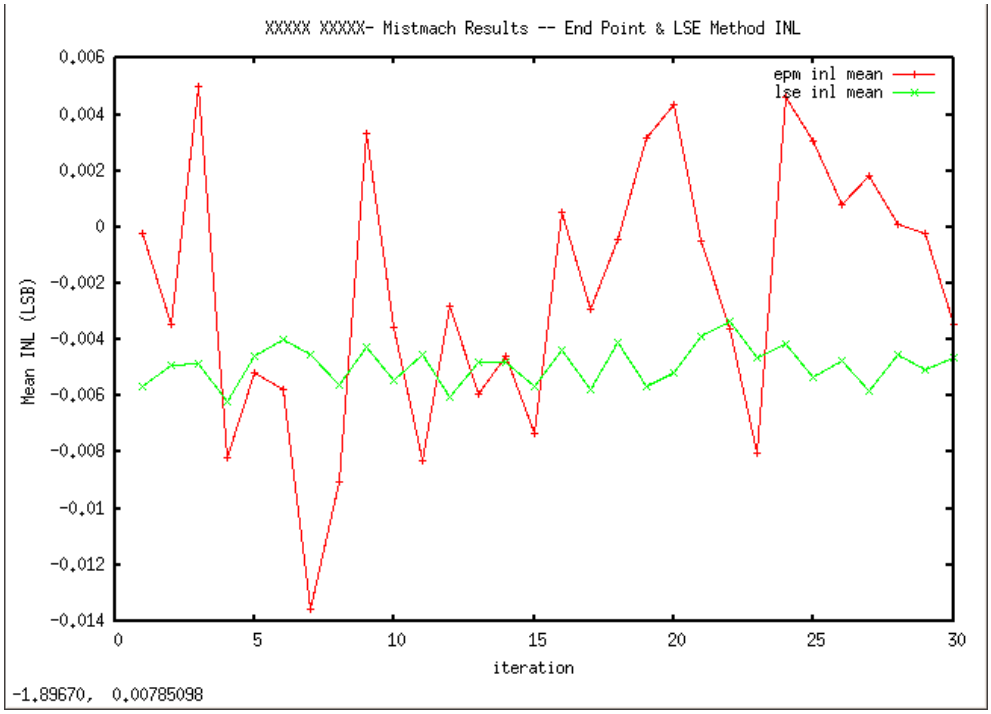


Figure 9.105: Mean INL (LSB) - *mismatch, MonteCarlo EndPoint&LSE method*

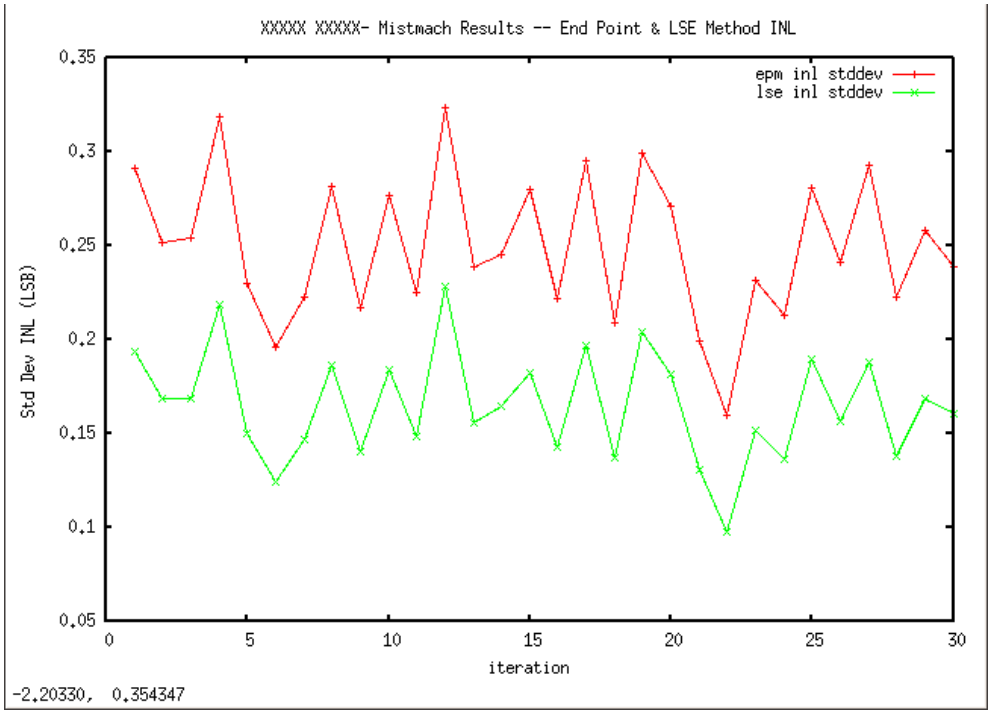


Figure 9.106: Standard Deviation INL (LSB) - *mismatch, MonteCarlo End-Point&LSE method*

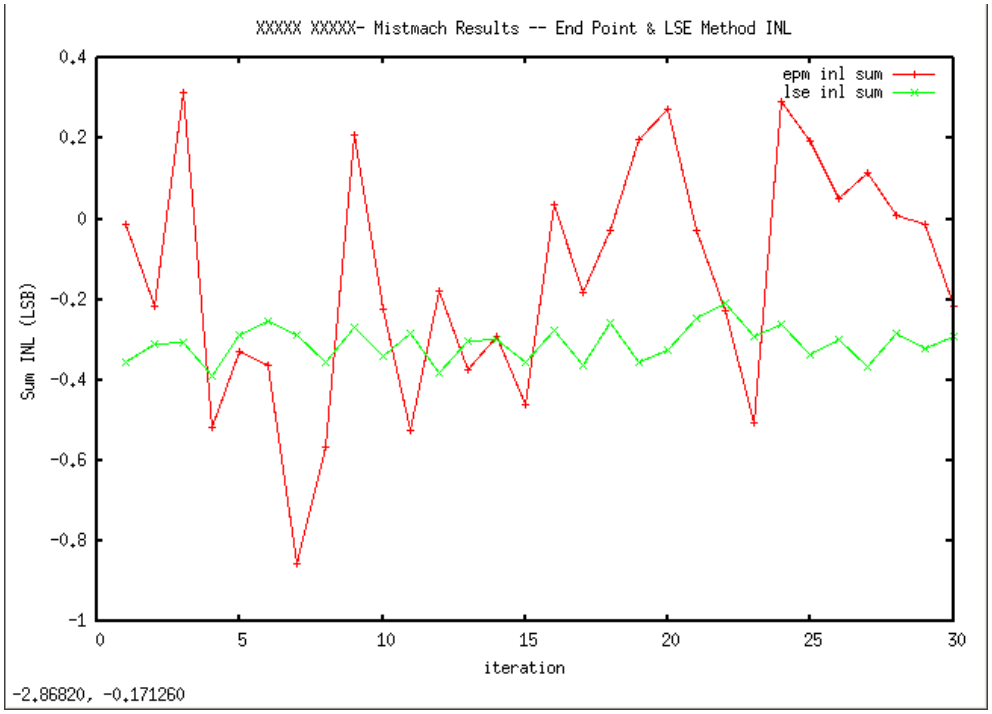


Figure 9.107: Sum INL (LSB) - *mismatch, MonteCarlo EndPoint&LSE method*

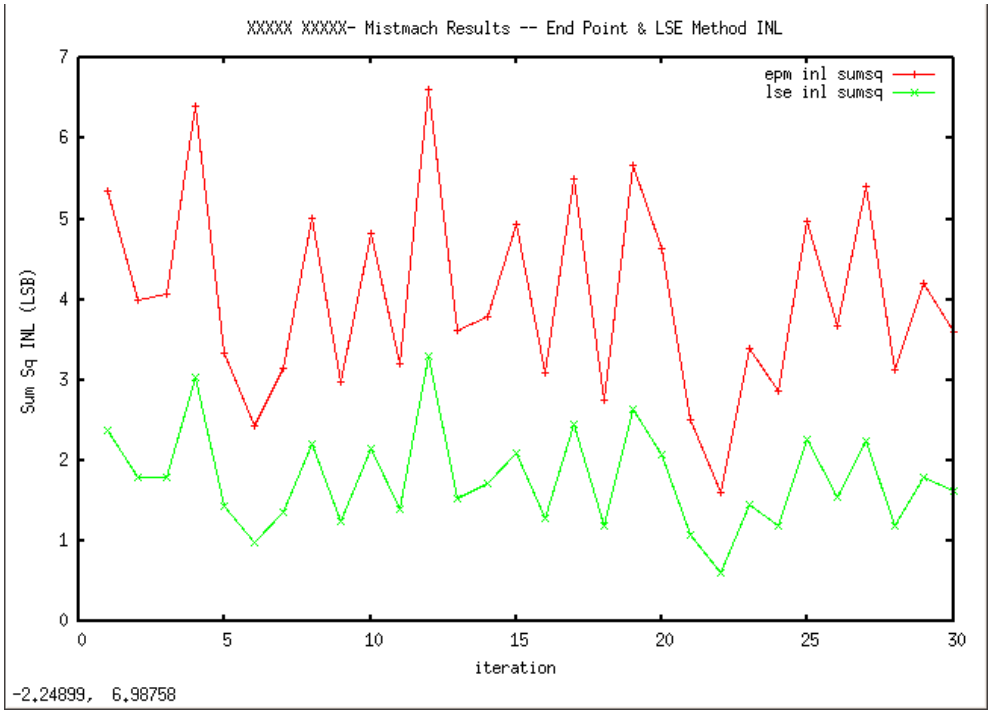


Figure 9.108: Sum Sq INL (LSB) - *mismatch, MonteCarlo EndPoint&LSE method*

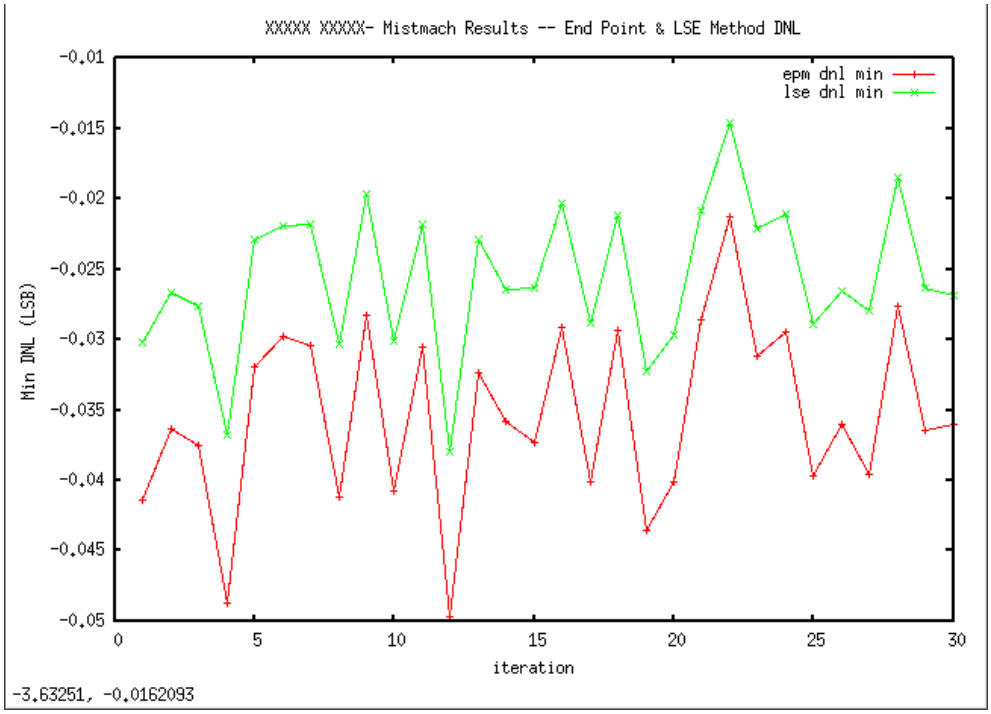


Figure 9.109: Min DNL (LSB) - *mismatch, MonteCarlo EndPoint&LSE method*

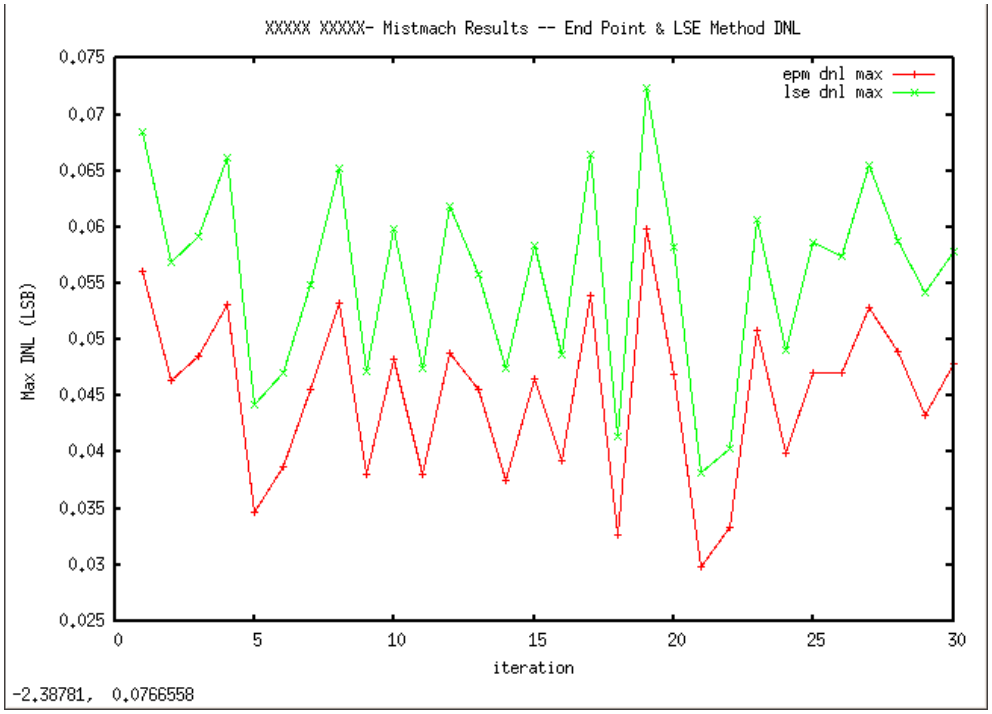


Figure 9.110: Max DNL (LSB) - *mismatch, MonteCarlo EndPoint&LSE method*

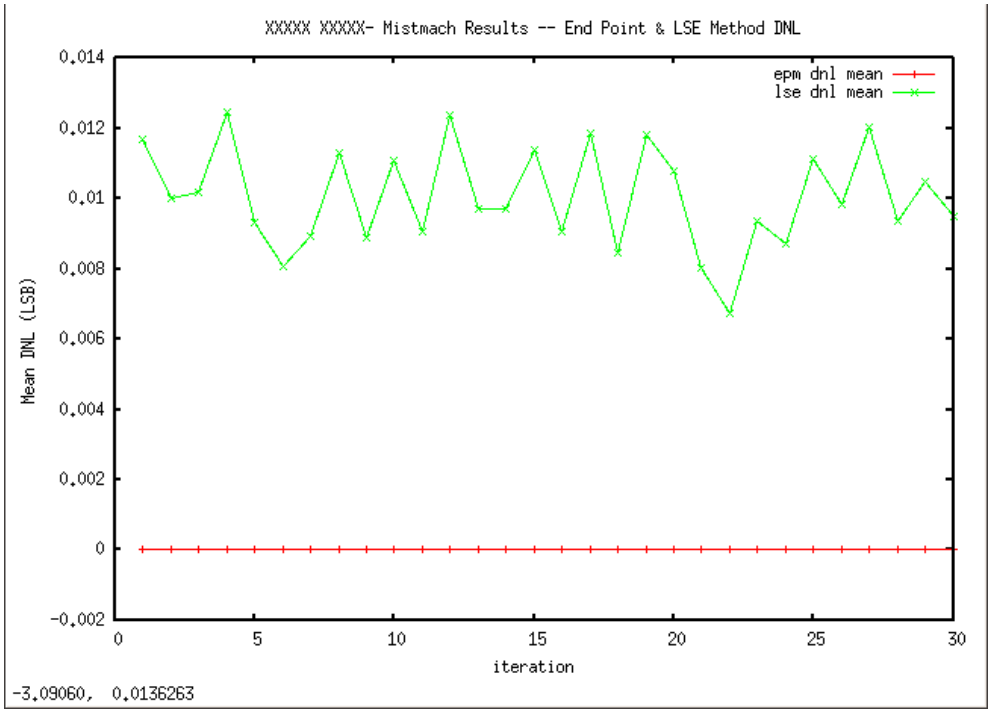


Figure 9.111: Mean DNL (LSB) - *mismatch, MonteCarlo EndPoint&LSE method*

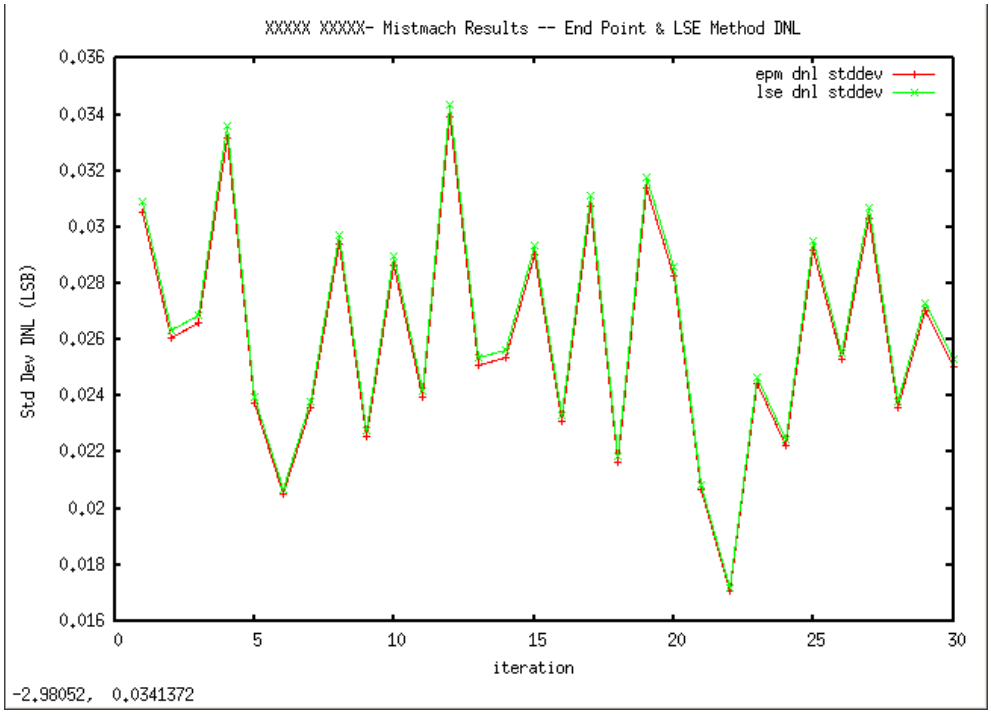


Figure 9.112: Standard Deviation DNL (LSB) - *mismatch, MonteCarlo End-Point&LSE method*

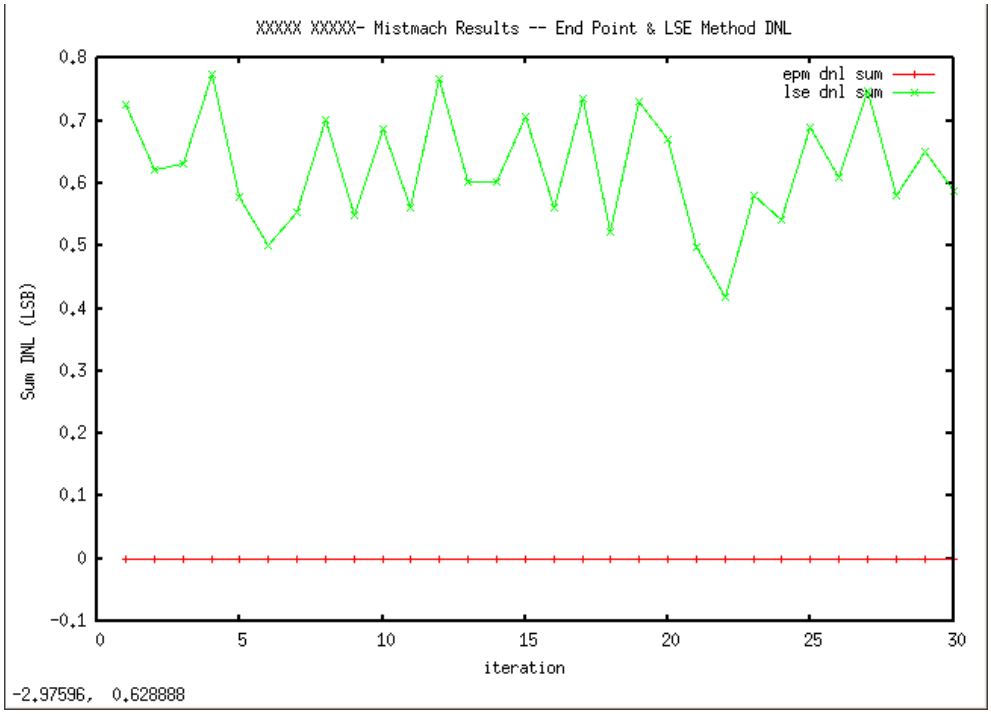


Figure 9.113: Sum DNL (LSB) - *mismatch, MonteCarlo EndPoint&LSE method*

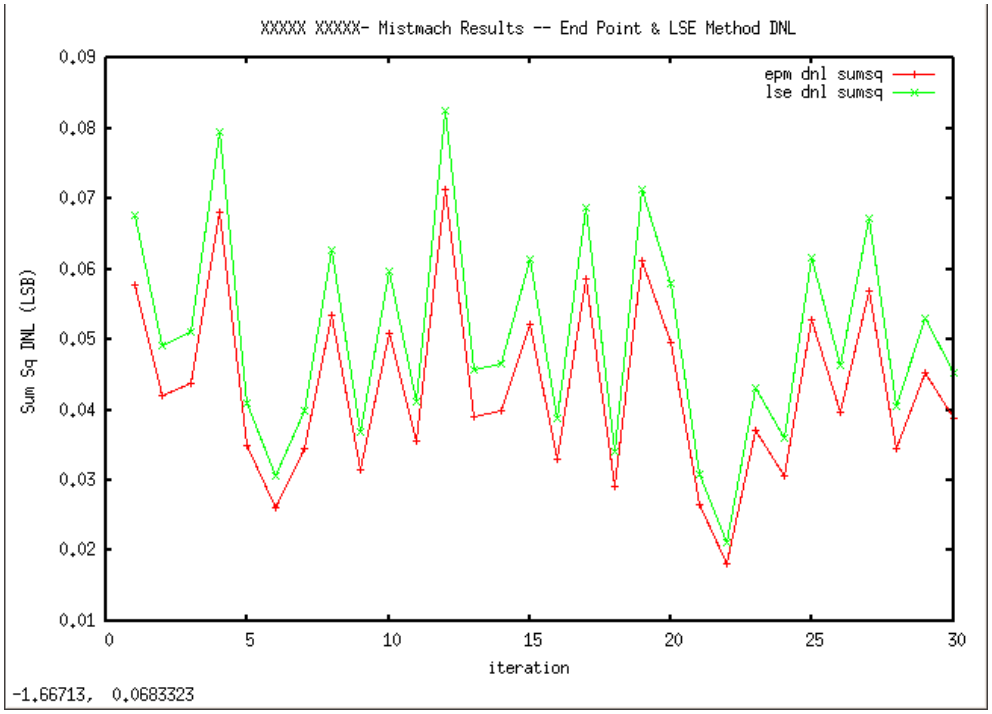


Figure 9.114: Sum Sq DNL (LSB) - *mismatch, MonteCarlo EndPoint&LSE method*

9.4 CONCLUSIONS

The aim of this work was the optimization of a threshold configurable regenerative comparator for ultra-low power battery-less applications and the analysis of the effect that this power reduction has over non-linearity in a 6-bit TC-ADC.

The power optimized comparator cuts down on 77% the power consumption and improves by a 20% the maximum sampling rate for a constant FS specification of $\pm 160mV$. The higher energy efficiency causes improved linearity (about 50% for offset, gain, DNL and INL errors) for perfectly matched devices, while it exhibits higher sensitivity to dynamic random errors (21% worse INL for mismatched devices). Gain error is significatively improved even in the presence of random errors.

Future work has to be done by using the higher predictability of INL errors obtained for the power optimized comparator in order to simplify the calibrating networks.

Part IV

APPENDIX

APPENDIX

A.1 THE SAR ALGORITHM

The successive approximation ADC circuit typically consists of four chief subcircuits (Figure A.1):

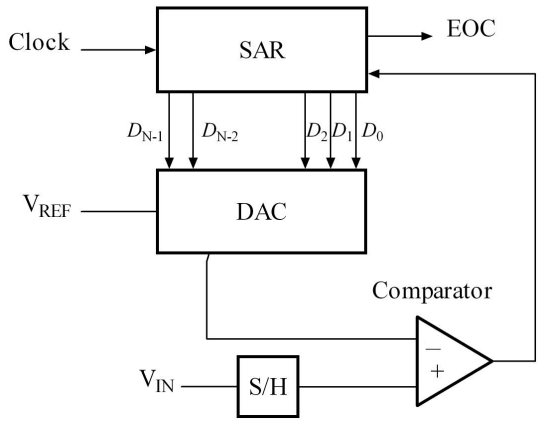


Figure A.1: Successive Approximation ADC Block Diagram

1. A S/H circuit to acquire the input voltage (V_{IN}).
2. An analog voltage comparator that compares V_{IN} to the output of the internal DAC and outputs the result of the comparison to the SAR.
3. A successive approximation register subcircuit designed to supply an approximate digital code of V_{IN} to the internal DAC.
4. An internal reference DAC that supplies the comparator with an analog voltage equivalent of the digital code output of the SAR for comparison with V_{IN}.

The SAR is initialized so that the MSB is equal to a digital 1. This code is fed into the DAC, which then supplies the analog equivalent of this digital code $\left(\frac{V_{ref}}{2}\right)$ into the comparator circuit for comparison with the sampled input voltage. If this analog voltage exceeds V_{IN} the comparator causes the SAR to reset this bit; otherwise, the bit is left to 1. Then the next bit is set to 1 and the same test is done, continuing this binary search until every bit in the SAR has been tested. The resulting code is the digital approximation of the sampled input voltage and is finally output by the DAC at the End Of the Conversion (EOC).

Mathematically, assume $V_{in} = x\, V_{ref}$, so that x in $[-1, 1]$ is the normalized input voltage. The objective is to approximately digitize x to an accuracy of $\frac{1}{2n}$. The algorithm proceeds as follows:

1. Initial approximation $x_0 = 0$.
2. ith approximation $x_i = x_{i-1} - \frac{s(x_{i-1}-x)}{2^i}$.

where $s\,(x)$ is the signum-function $sgn(x)$ ($+1$ for $x \geq 0$, -1 for $x < 0$). It follows, by mathematical induction, that $|xn - x| \leq \frac{1}{2n}$. [47]

As shown in the above algorithm, a SAR ADC requires:

A. An input voltage source V_{IN}.
B. A reference voltage source V_{ref} to normalize the input.
C. A DAC to convert the ith approximation x_i to a voltage.
D. A Comparator to perform the function $s\,(x_i - x)$ by comparing the DAC's voltage with the input voltage.
E. A Register to store the output of the comparator and apply $x_{i-1} - \frac{s(x_{i-1}-x)}{2^i}$.

A.2 NYQUIST SAMPLING CONDITION

The Nyquist–Shannon sampling theorem, after Harry Nyquist and Claude Shannon, in the literature more commonly referred to as the Nyquist sampling condition, is a fundamental result in the field of information theory.

Shannon's version of the theorem states:

> If a function $x(t)$ contains no frequencies higher than $B\,Hz$, it is completely determined by giving its ordinates at a series of points spaced $\frac{1}{2B}\,s$ apart.

In other words, a bandlimited function can be perfectly reconstructed from an infinite sequence of samples if the bandlimit B is no greater than $\frac{1}{2}$ the sampling rate (samples per second). The theorem also leads to a formula for reconstruction of the original function from its samples. When the bandlimit is too high (or there is no bandlimit), the reconstruction exhibits imperfections known as *aliasing*. The Poisson summation formula provides a graphic understanding of aliasing and an alternative derivation of the theorem, using the perspective of the function's Fourier transform. In practice of course, infinite sequences, perfect sampling, and perfect interpolation are all replaced by approximations that deviate from the mathematical ideal of perfect reconstruction. Moreover, the theorem is a sufficient, but not necessary, condition.

To formalize the statements above, let $X(f)$ be the Fourier transform of the bandlimited function $x(t)$:

$$X(f) \stackrel{def}{=} \int_{-\infty}^{\infty} x(t)e^{-i2\pi ft}dt \tag{A.1}$$

and assume that

$$X(f) = 0, \quad \forall\, |f| > B$$

When $x(t)$ is uniformly sampled at intervals of T seconds, the resulting sequence is denoted by $x(nT)$, for all integer values of n, and the sample-rate (samples per second) is:

$$f_s \stackrel{def}{=} \frac{1}{T}$$

A sufficient condition to reconstruct $x(t)$ from its samples is $f_s > 2B$, or equivalently $B < \frac{f_s}{2}$. The two thresholds, $2B$ and $\frac{f_s}{2}$, are respectively called the *Nyquist rate* and *Nyquist frequency*. And they are attributes of $x(t)$ and of the sampling equipment respectively. The condition described by these inequalities is called *the Nyquist condition* [48].

A.3 NYQUIST FREQUENCY

The Nyquist frequency is half the sampling frequency of a discrete signal processing system [49, 50]. It is sometimes known as the folding frequency of a sampling system [51].

When the continuous function to be sampled contains no frequencies equal or higher than the Nyquist frequency, all the aliases caused by sampling occur above the Nyquist frequency. The term aliasing usually refers to the case where some original frequency components have aliases below Nyquist. That often causes distortion when a continuous function is subsequently reconstructed from samples.[52]

A.4 CORNER OF PROCESS

In semiconductor manufacturing, a process corner is an example of a Design-of-Experiments (DoE) technique that refers to a variation of fabrication parameters used in applying an integrated circuit design to a semiconductor wafer. Process corners represent the extremes of these parameter variations within which a circuit that has been etched onto the wafer must correctly function (Figure A.2):

- SLOW CORNER. Everything goes wrong, that is all process variations deviate towards a device with reduced currents.
- FAST CORNER. Everything is better than planned, that is all process variations deviate towards a device with increased currents.

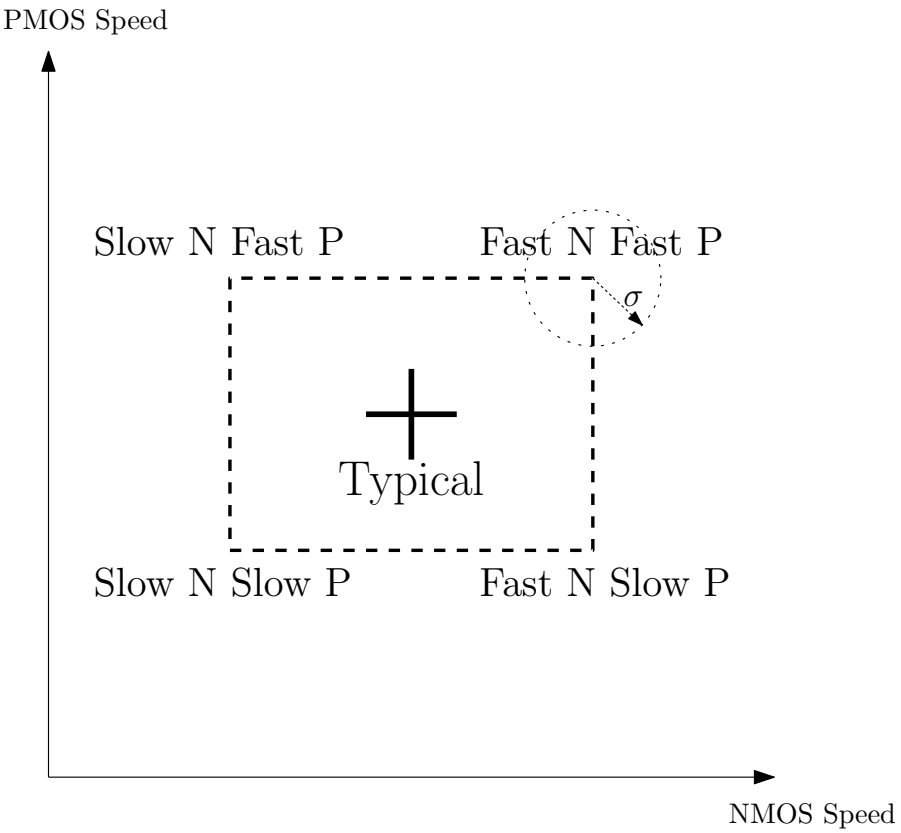


Figure A.2: Process corners in PMOS/NMOS fabrication parameters

- Cross corners. One device type has increased, the other type reduced currents.

A circuit running on devices fabricated at these process corners may run slower or faster than specified and at lower or higher temperatures and voltages, but if the circuit does not function at all at any of these process extremes the design is considered to have inadequate design margin [53].

In Very-Large-Scale-Integrated (VLSI) circuit microprocessor design and semiconductor fabrication, a process corner represents a three or six sigma variation from nominal doping concentrations (and other parameters) in transistors on a silicon wafer. This variation can cause significant changes in the duty cycle and slew rate of digital signals, and it can sometimes result in catastrophic failure of the entire system. Variation may occur for many reasons, such as minor changes in the humidity or temperature changes in the clean-room when wafers are transported, or due to the position of the die relative to the center of the wafer [54].

A.5 BISECTION METHODOLOGY IN HSPICE

Bisection is an optimization methodology that uses a binary search method to find the value of an input variable (*target value*). This variable is associated with a goal value of an output variable. The type of the input and output variables can be voltage, current, delay time, or gain, related by some transfer function. In general, it is used a binary search to locate the goal value of the output variable, within a search range of the input variable. Then iteratively halve that range, to rapidly converge on the target value. At each iteration, HSPICE compares the measured value of the output variable, with the goal value. Both the pass/fail method and the bisection method use bisection [55].

The bisection procedure involves two measurement and optimization steps:

A. Detects whether the output transition occurred.

B. Automatically varies the input parameter, to find the value for which the transition barely occurs.

Measurement

The `MAX` measurement function can be used to detect the success or failure of an output transition. For a low-to-high output transition, a `MAX` measurement produces *zero* on failure, or approximately the V_{dd} supply voltage on success. This measurement, using a goal of V_{dd} (minus a suitable small value to ensure a solution), is sufficient to drive the optimization.

Optimization

The bisection method is straightforward, if you specify a single measurement, with a goal, and known upper and lower boundary values, for the input parameter. The engineer must specify acceptable upper and lower boundary values.

Using Bisection

Before being able to use bisection, the following specifications must be given:

- A pair of values, for the *upper* and *lower boundaries* of the input variables. To find a solution, one of these values must result in an output variable `S |goal value|` and the other must result in `< |goal value|`.
- A `goal` value.

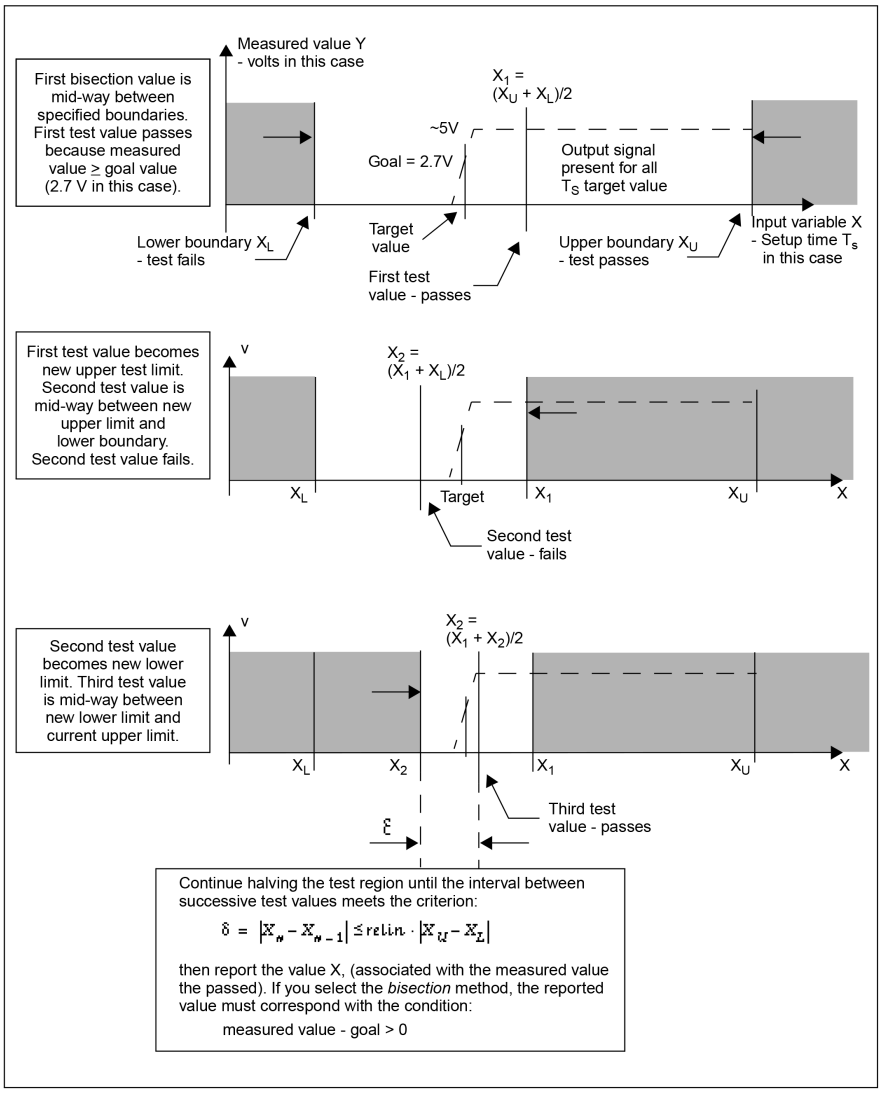


Figure A.3: Bisection Example for Three Iterations

- *Error tolerance* value. The bisection process stops when the difference between successive test values is minor or equal to the error tolerance. If the other criteria are also met, see below.
- *Related variables*. A monotonic transfer function to relate variables has to be used. A steadily-progressing time (increase or decrease) results in a single occurrence of the goal value at the target input variable value.

HSPICE includes the error tolerance in a relation, used as a process-termination criterion.

Figure A.3 shows an example of the binary search process, which the bisection algorithm uses. This example is the pass/fail type, and for example it is appropriate for a setup-time analysis that tests for the presence of an output transition. In this case:

1. A long setup time $T_S(= T_2 - T_1)$ results in a V_{OUT} transition (a pass).
2. A too-short setup time (where the latch has not stabilized the input data, before the clock transition) results in a fail.
3. For example, it is possible to define a pass time value as any setup time, T_S, that produces a V_{OUT} output equal to the minimum *High* logic output level ($2.7V$), which is the goal value.
4. The target value is a setup time that just produces the V_{OUT} value of $2.7V$. Finding the exact value is impractical, if not impossible, so specify an error tolerance, to calculate a solution arbitrarily close to the target value.
5. The bisection algorithm performs tests for each specified boundary value, to determine the direction in which to pursue the target value, after the first bisection. In this example, the upper boundary has a pass value, and the lower boundary has a fail value.
6. To start the binary search, specify the lower and upper boundaries. The program tests the point midway between the lower and upper boundaries.
    a) If the initial value passes the test, the target value must be less than the tested value (in this example). The bisection algorithm moves the upper search limit to the value that it just tested.
    b) If the test fails, the target value must be greater than the tested value. Bisection moves the lower limit to the value that it just tested.
7. The algorithm tests a value midway between the new limits.
8. The search continues in this manner, moving one limit or the other to the last midpoint, and testing the value midway between the new limits.
9. The process stops when the difference between the latest two test values is less than or equal to the specified error tolerance. To normalize this value, there is to multiply by the initial boundary range.

Command Syntax

The following syntax is used for bisection:

```
.MODEL <OptModelName> OPT METHOD=BISECTION ...
```

or

```
.MODEL <OptModelName> OPT METHOD=PASSFAIL ...
```

OptModelName is the model to be used. The METHOD keyword indicates which optimization method to use. The OPT keyword indicates that optimization is to be performed.

For bisection, the method can be one of the following:

- BISECTION. When the difference between the two latest test input values is within the error tolerance and the latest measured value exceeds the goal, bisection has succeeded and then ends. This process reports the optimized parameter that corresponded to the test value that satisfies this error tolerance and this goal (passes).
- PASSFAIL. When the difference between the two latest test input values is within the error tolerance and one of the values $\geq$ goal (passes) and the other fails, bisection has succeeded and then ends. The process reports the input parameter value associated with the "pass" measurement.

The parameters are passed in a normal optimization specification:

```
.PARAM <ParamName>=<OptParFun> (<Initial>, <Lower>, <Upper>)
```

In the BISECTION method, the measure results for <Lower> and <Upper> limits of <ParamName> must be on opposite sides of the goal value in the .MEASURE statement. In the PASSFAIL method, the measure must pass for one limit and fail for the other limit. The process ignores the value of the <Initial> field. The error tolerance is a parameter in the model being optimized. In both methods, bisectional search is applied to only one parameter.

When the OPTLST option is set (.OPTION OPTLST=1), the process outputs the following information for the BISECTION method:

```
bisec-opt iter = <num_iterations> xlo = <low_val> xhi = <high_val
    >
x = <result_low_val> xnew = <result_high_val>
err = <error_tolerance>
```

The x is the old parameter value and xnew is the new parameter value.

When .OPTION OPTLST=1, the process outputs the following information for the PASSFAIL method:

```
bisec-opt iter = <num_iterations> xlo = <low_val> xhi = <high_val
    >
```

```
x = <result_low_val> xnew = <result_high_val>
measfail = 1
```

In this syntax, `measfail = 0` for a test failure for the x value.

Performing Transient Analyses with Bisections

When performing transient analysis bisection with the `.TRAN` statement, use the following syntax:

```
.TRAN <TranStep> <TranTime> SWEEP OPTIMIZE=<OptParFun>
+ RESULTS=<MeasureNames> MODEL=<OptModelName>
```

When performing a transient analysis bisection with the `.MEASURE` statement, use the following syntax:

```
.MEASURE TRAN <MeasureName> <MeasureClause> GOAL=<GoalValue>
```

A.6 tcadc FILE CODE

It is presented the Perl code for the `tcadc` file, actually the *main* file which contains all the preprocessing simulation commands, parameters, stimulus, netlist, measurements and probe points that have to be run.

Listing 1: `tcadc`

```
#!/usr/bin/perl
$inputstr = 's';
$inputm = 1;
$dotest = 0;
$noiseprocess = 0;
if($#ARGV == 0){
        $inputstr = $ARGV[0];
        }
        elsif($#ARGV == 1) {
                $inputstr = $ARGV[0];
                $inputm = $ARGV[1];
                }
                elsif($#ARGV == 2) {
                        $inputstr = $ARGV[0];
                        $inputm = $ARGV[1];
                        }
                        elsif($#ARGV == 3) {
                                $inputstr = $ARGV[0];
                                $inputsts = $ARGV[1];
                                $inputm = $ARGV[2];
                                }
                                else {
```

```
                                        goto USAGE;
                                        }

START:
if ($inputstr eq 'C') {
        $unique_filename = "*";
        $unique_dir = $ARGV[1];
        goto RM;
        }
        elsif ($inputstr eq 'c') {
                $prefix = "tcadc";
                $sufix  = ".dat";
                $unique_filename = $ARGV[1];
                $unique_dir = $ARGV[1];
                goto CLEAN;
                }
                elsif ($inputstr eq 'r') {
                        $prefix = "tcadc";
                        $sufix  = ".dat";
                        $inputstr = $inputsts;
                        $unique_dir = $ARGV[3];
                        $unique_filename = $ARGV[3];
                        goto PROCESS;
                        }
                        elsif ($inputstr eq 'n') {
                                $prefix = "tcadc";
                                $sufix  = ".dat";
                                $inputstr = $inputsts;
                                $unique_dir = $ARGV[3];
                                $unique_filename = $ARGV[3];
                                $noiseprocess = 1;
                                goto NOISE;
                                }
        elsif ($inputstr eq 'u') {
                $prefix = "tcadc";
                $sufix  = ".dat";
                if(($inputsts ne 'o')&&($inputsts ne 'm')){
                goto USAGE;
                }
                $inputstr = $inputsts;
                $inputsts = 's';
                $unique_dir = $ARGV[3];
                $unique_filename = $ARGV[3];
                $noiseprocess = 2;
                goto NOISE;
                }
                elsif ($inputstr eq 't') {
                        $prefix = "tcadc";
                        $sufix  = ".dat";
                        $unique_dir = "test";
                        $unique_filename = "test";
                        $dotest=1;;
```

```
                        goto MKDIR;
                        }
                        elsif ($inputstr eq 'x') {
                                $prefix = "tcadc";
                                $sufix  = ".dat";
                                $inputstr = $inputsts;
                                $unique_dir = $ARGV[3];
                                $unique_filename = $ARGV[3];
                                goto RERUN;
                                }
        elsif (((($inputstr eq 'm')||($inputstr eq 'o')||(
            $inputstr eq 's'))&&($#ARGV == 2)){
                $prefix = "tcadc";
                $sufix  = ".dat";
                $unique_filename = $ARGV[2];
                $unique_dir = $ARGV[2];
                goto MKDIR;
                }
                elsif (((($inputstr eq 'm')||($inputstr eq 'o')||(
                    $inputstr eq 's'))&&($#ARGV == 1)){
                        $prefix = "tcadc";
                        $sufix  = ".dat";
                        $unique_filename = $prefix . "_" .
                            get_timestamp();
                        $unique_dir = $unique_filename;
                        goto MKDIR;
                        }

NOISE:
$mkvth = "../bin/mkvthsp ";
$vthfile = $unique_dir . "/" . $prefix . $sufix . $inputstr .
    $inputm . "-vth";
$spfile = $unique_dir . "/" . $prefix . $sufix . $inputstr .
    $inputm . "-vth-sp";
$command = join "", $mkvth, " ", $vthfile,  " > ", $spfile;
print "\n";
print "...processing command:  $command\n";
print "\n";
system($command);
$command = join "", "cat ", "pre-", $inputsts, "header.sp > ",
    $inputsts, "header.sp";
@args = ($command);
print "\n";
print "...generating header file: @args\n";
print "\n";
system(@args);
if(($inputsts eq 'm')){
        $command = join "", "printf \" monte=%d\\n\"", " ",
            $inputm, " >> ", $inputsts, "header.sp";
        @args = ($command);
        print "\n";
        print "...generating header file: @args\n";
```

```
        print "\n";
        system(@args);
        }
$command = join "", "cat ", $spfile ," >> ", $inputsts, "header.
    sp";
@args = ($command);
print "\n";
print "...generating header file: @args\n";
print "\n";
system(@args);
$command = join "", "cat ", "pos_", $inputsts, "header.sp >> ",
    $inputsts, "header.sp";
@args = ($command);
print "\n";
print "...generating header file: @args\n";
print "\n";
system(@args);
$inputstr = $inputsts;
$unique_filename = $prefix . "_" . get_timestamp();
$unique_dir = $unique_filename;
goto MKDIR;

MKDIR:
$command = join "", "mkdir ", $unique_dir;
@args = ($command);
print "\n";
print "...making simulation dir: @args\n";
print "\n";
system(@args);

MKMT:
if($dotest){
        for($iitem=0;
        $iitem<=62;
        $iitem++){
                $command = join "", "printf \"# \\n\"", " > ",
                    $unique_dir, "/", $unique_filename, ".mt",
                    $iitem;
                print "\n";
                print "...making gnuplot command:  $command\n";
                print "\n";
                system($command);
                $command = join "", "printf \"# Test File
                    Generated: %s\\n\"", " ", get_timestamp(),  "
                     >> ", $unique_dir, "/", $unique_filename, ".
                    mt", $iitem;
                print "\n";
                print "...making gnuplot command:  $command\n";
                print "\n";
                system($command);
                $command = join "", "printf \"index vid tdeco1
                    tdeco2 vdeco1 vdeco2 avgpt avgimo avgim1
```

```
                    avgim2 avgim1–avgim2 avgim1+avgim2 avgiam1
                    avgiam2 avgiam3 avgiam4 rmspt rmsimo rmsim1
                    rmsim2 rmsim1–rmsim2 rmsim1+rmsim2 rmsiam1
                    rmsiam2 rmsiam3 rmsiam4 intpt intimo intim1
                    intim2 intim1–intim2 intim1+intim2 intiam1
                    intiam2 intiam3 intiam4 vid_ dout1 dout2
                    dout1–dout2 vina_ vinb_ vx0 vx1 vx2 temper
                    alter#\\n\"",  " >> ", $unique_dir, "/",
                    $unique_filename, ".mt", $iitem;
                print "\n";
                print "...making gnuplot command:  $command\n";
                print "\n";
                system($command);
                $command = join "", "printf \"1 %g 03 04 05 06 07
                     08 09 10 11 12 13 14 15 16 17 18 19 20 21 22
                     23 24 25 26 27 28 29 30 31 32 33 34 35 36 37
                     38 39 40 41 42 43 44 45 46 %g\\n\"",  " ", -
                    $iitem + 31, " ", $iitem + 1, " >> ",
                    $unique_dir, "/", $unique_filename, ".mt",
                    $iitem;
                print "\n";
                print "...making gnuplot command:  $command\n";
                print "\n";
                system($command);
                }
        print "usage: tcadc 'r m 1 test'\n";
        print "and modify spec file'\n";
        goto EXIT;
        }

HEADER:
if(($noiseprocess == 0)){
        $command = join "", "cat ", "pre–", $inputstr, "header.sp
             > ", $inputstr, "header.sp";
        @args = ($command);
        print "\n";
        print "...generating header file: @args\n";
        print "\n";
        system(@args);
        if(($inputstr eq 'm')){
                $command = join "", "printf \" monte=%d\\n\"", "
                    ", $inputm, " >> ", $inputstr, "header.sp";
                @args = ($command);
                print "\n";
                print "...generating header file: @args\n";
                print "\n";
                system(@args);
                }
    $command = join "", "cat ", "vid–", $inputstr, "header.sp >>
        ", $inputstr, "header.sp";
        @args = ($command);
        print "\n";
```

```
        print "...generating header file: @args\n";
        print "\n";
        system(@args);
        $command = join "", "cat ", "pos_", $inputstr, "header.sp
             >> ", $inputstr, "header.sp";
        @args = ($command);
        print "\n";
        print "...generating header file: @args\n";
        print "\n";
        system(@args);
        }

SIMULATION:
$command = join "", "cat ", $prefix, ".sp ",  $inputstr, "header.
    sp > ", $unique_dir, "/", $unique_filename, ".hsp";
@args = ($command);
print "\n";
print "...generating simulation file: @args\n";
print "\n";
system(@args);
RERUN:
$command = join "", "hspice ", $unique_dir, "/", $unique_filename
    , ".hsp > ", $unique_dir, "/", "simulation.log";
@args = ($command);
print "\n";
print "...running hspice simulation: @args\n";
print "\n";
system(@args);
$command = join "", "mv ", $unique_filename, ".* ", $unique_dir,
    "/.";
@args = ($command);
print "\n";
print "...moving hspice output files to simulation dir: @args\n";
print "\n";
system(@args);

PROCESS:
$zero   = join "", $unique_dir, "/", $unique_filename;
$first  = join "", $unique_dir, "/", $unique_filename, ".mt";
$second = join "", $unique_dir, "/", $prefix, $sufix, $inputstr;
$extract = "../bin/extractor ";
$catcomm = "cat ";
if($inputstr eq 'm'){
        @letters = ("+31", "+30", "+29", "+28", "+27", "+26", "
            +25", "+24", "+23", "+22", "+21", "+20", "+19", "+18"
            , "+17", "+16", "+15", "+14", "+13", "+12", "+11", "
            +10", "+9", "+8", "+7", "+6", "+5", "+4", "+3", "+2",
             "+1", "-0", "-1", "-2", "-3", "-4", "-5", "-6", "-7"
            , "-8", "-9", "-10", "-11", "-12", "-13", "-14", "-15
            ", "-16", "-17", "-18", "-19", "-20", "-21", "-22", "
            -23", "-24", "-25", "-26", "-27", "-28", "-29", "-30"
            , "-31");
```

```
        }
        else {
                @letters = ("-0", "-1", "-2", "-3", "-4", "-5", "
                    -6", "-7", "-8", "-9", "-10", "-11", "-12", "
                    -13", "-14", "-15", "-16", "-17", "-18", "-19
                    ", "-20", "-21", "-22", "-23", "-24", "-25",
                    "-26", "-27", "-28", "-29", "-30", "-31");
                }
if(($inputstr eq 's')){
        $i = 0;     $iitem=1;
        foreach $stritem (@letters){
                $command = join "", $extract, $first, $i,  " > ",
                     $second, $iitem, $stritem;
                print "\n";
                print "...processing command:  $command\n";
                print "\n";
                system($command);
                $i++;
                }
        }
if(($inputstr eq 'o')||($inputstr eq 'm')){
        $i = 0;
        foreach $stritem (@letters){
                for ($iitem = 1;
                $iitem <= $inputm;
                $iitem++){
                        $command = join "", $extract, $first, $i,
                             " line ", $iitem,  " > ", $second,
                            $iitem, $stritem;
                        print "\n";
                        print "...processing command:  $command\n
                            ";
                        print "\n";
                        system($command);
                        }
                $i++;
                }
        }
if(($inputstr eq 'o')||($inputstr eq 'm')){
        for ($iitem = 1;
        $iitem <= $inputm;
        $iitem++){
                foreach $stritem (@letters){
                        $command = join "", $catcomm, $second,
                            $iitem, $stritem, " >> ", $second,
                            $iitem;
                        print "\n";
                        print "...processing command:  $command\n
                            ";
                        print "\n";
                        system($command);
                        }
```

```
            if($inputstr eq 'o'){
                    $command = join "", $catcomm, $second,
                        $iitem, "-o ", $second, $iitem, "-31
                        > ", $second, $iitem, "-o", "-31";
                    print "\n";
                    print "...processing command:  $command\n
                        ";
                    print "\n";
                    system($command);
                    }
                    else {
                            $command = join "", $catcomm,
                                $second, $iitem, "+31 ",
                                $second, $iitem, "-31 > ",
                                $second, $iitem, "+31", "-31"
                                ;
                            print "\n";
                            print "...processing command:
                                $command\n";
                            print "\n";
                            system($command);
                            }
                    }
            }
    if(($inputstr eq 'o')||($inputstr eq 'm')) {
            $j = 0;
            foreach $stritem (@letters){
                    $j++;
                    if($j==1){
                            $stritem1 = $stritem;
                            }
                            elsif($j==2){
                                    $stritem2 = $stritem;
                                    for ($iitem = 1; $iitem
                                        <= $inputm; $iitem++)
                                        {
                                            $command = join "
                                                ", $extract,
                                                $second,
                                                $iitem,
                                                $stritem1, "
                                                ", $second,
                                                $iitem,
                                                $stritem2, "
                                                > ", $second,
                                                 $iitem,
                                                $stritem1,
                                                $stritem2;
                                            print "\n";
                                            print "...
                                                processing
```

```
                                                    command:
                                                    $command\n";
                                                print "\n";
                                                system($command);
                                                }
                                        $stritem1 = $stritem2;
                                        $j--;
                                        }
                                }
                        }
                goto CPSPEC;

BISECOPT: #Jump this part of the code. This is to generate a
    check file for bisection opt.
$bisecopt = "../bin/bisecopt ";
$simlogfile = $unique_dir . "/simulation.log";
$bisecoptfile = $unique_dir . "/" . $prefix . $sufix . $inputstr
    . $inputm . "bisec-opt";
if($inputstr eq 'o'){
        $command = join "", $bisectopt, $simlogfile, " > ",
            $bisectoptfile;
        @args = ($command);
        print "\n";
        print "...obtaining bisec-opt file from output simulation
             log file: @args\n";
        print "\n";
        system(@args);
        }

CPSPEC:
$command = join "", "cp ", $prefix, $sufix, $inputstr, "-spec ",
    $unique_dir, "/.";
@args = ($command);
print "\n";
print "...copying spec file to simulation dir: @args\n";
print "\n";
system(@args);

GNU:
$command = join "", "printf \"gnuplot -e \\\"argv0='%s';
argv1=%g;
argv2='%s';
argv3=-1\\\" %s.gpl\\n\"", " ", $inputstr, " ", $inputm, " ",
    $unique_dir, " ", $prefix, " > ", $unique_filename, ".gnu";
print "\n";
print "...making gnuplot command:  $command\n";
print "\n";
system($command);
if(($inputstr eq 'o')||($inputstr eq 'm')) {
        $command = join "", "printf \"gnuplot -e \\\"argv0='%s';
        argv1=0;
        argv2='%s';
```

```
        argv3=-1\\\"%s.gpl\\n\"", " ", $inputstr, " ",
            $unique_dir, " ", $prefix, " >> ", $unique_filename,
            ".gnu";
        print "\n";
        print "...making gnuplot command:  $command\n";
        print "\n";
        system($command);
        }
$command = join "", "chmod +x ", $unique_filename, ".gnu",;
print "\n";
print "...modifying gnuplot command:  $command\n";
print "\n";
system($command);
goto EXIT;

RM:
$command = join "", "rm ", $unique_dir, "/", $unique_filename, ".
    hsp";
print "\n";
print "...processing command:  $command\n";
print "\n";
system($command);
$command = join "", "rm ", $unique_dir, "/", $unique_filename, ".
    mt*";
print "\n";
print "...processing command:  $command\n";
print "\n";
system($command);
$command = join "", "rm ", $unique_dir, "/", $unique_filename, ".
    tr*";
print "\n";
print "...processing command:  $command\n";
print "\n";
system($command);
$command = join "", "rm ", $unique_dir, "/", $unique_filename, ".
    st*";
print "\n";
print "...processing command:  $command\n";
print "\n";
system($command);
$command = join "", "rm ", $unique_dir, "/", $unique_filename, ".
    pa*";
print "\n";
print "...processing command:  $command\n";
print "\n";
system($command);
$command = join "", "rm ", $unique_dir, "/", $unique_filename, ".
    ic*";
print "\n";
print "...processing command:  $command\n";
print "\n";
system($command);
```

```
CLEAN:
$command = join "", "rm ", $unique_dir, "/", $prefix, $sufix, "*"
    ;
print "\n";
print "...processing command:  $command\n";
print "\n";
system($command);
$command = join "", "rm *tmp";
print "\n";
print "...processing command:  $command\n";
print "\n";
system($command);
$command = join "", "rm *.*~";
print "\n";
print "...processing command:  $command\n";
print "\n";
system($command);
$command = join "", "rm *~";
print "\n";
print "...processing command:  $command\n";
print "\n";
system($command);
EXIT: print "\n";
print "...That's all folks\n";
print "\n";
exit;

USAGE:
print "usage: tcadc 't'|{'s'|'o'|'m' index {tag}, index
    =1,2,3...}|'x'|'r'|'u'|'n' 's'|'o'|'m' {...} tag\n";
print "C: do a hard clean;
c: do a soft clean;
\n";
print "t: test;
s: scan mode;
o: bisection opt.;
m: montecarlo + bisection opt;
\n";
print "x: rerun simulation  (do a soft clean first);
r: pos-process simulation (do a soft clean first);
\n";
print "n: do noise analysis\n";
print "u: from 'o'|'m' to scan mode with noise analysis;
\n";
print "tag: directory tag;
\n";
exit;
sub get_timestamp {
        @abbr = qw( Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov
            Dec );
```

```
        ($sec,$min,$hour,$mday,$mon,$year,$wday,$yday,$isdst) =
            localtime(time);
        $year += 1900;
        return $mday . '_' . $abbr[$mon] . '_' . $year . '_' .
            $hour. '_' . $min. '_' . $sec;
        }
```

A.7 OPTIMIZATION MODELS FILES CODE

Here it is reported the code written in order to run different types of optimizing simulations for performance evaluation of TC-Comparator power dissipation, current and voltage best fitting values.

A.7.1 *Simulation script for `sheader.sp` file*

The `sheader.sp` file contains the directives for simulations based on the Scanning optimization mode.

Listing 2: `sheader.sp` - Scanning optimization mode

```
*************************************************************
**** Header for NO MonteCarlo Simulation
*************************************************************
**** Mismatch for MonteCarlo Simulation
******
.param sigma=0
.param process=0 mismatch=0
*************************************************************
.param vid=0 vid_start=0 vid_stop=0 vid_incr=0
*************************************************************
.param vid_noise=10.0uv vid_noise_incr=1.0uv vid_fs=-200mv
*************************************************************
.param vid_start_31='vid_noise+-8.6994776e-09' vid_stop_31='-
    vid_noise+-8.6994776e-09'
.param vid_start_30='vid_noise+-4.8675948e-03' vid_stop_30='-
    vid_noise+-4.8675948e-03'
.param vid_start_29='vid_noise+-9.7177363e-03' vid_stop_29='-
    vid_noise+-9.7177363e-03'
.param vid_start_28='vid_noise+-1.4564207e-02' vid_stop_28='-
    vid_noise+-1.4564207e-02'
.param vid_start_27='vid_noise+-1.9407156e-02' vid_stop_27='-
    vid_noise+-1.9407156e-02'
.param vid_start_26='vid_noise+-2.4253818e-02' vid_stop_26='-
    vid_noise+-2.4253818e-02'
.param vid_start_25='vid_noise+-2.9080763e-02' vid_stop_25='-
    vid_noise+-2.9080763e-02'
.param vid_start_24='vid_noise+-3.3941531e-02' vid_stop_24='-
    vid_noise+-3.3941531e-02'
.param vid_start_23='vid_noise+-3.8771286e-02' vid_stop_23='-
    vid_noise+-3.8771286e-02'
```

```
.param vid_start_22='vid_noise+-4.3659324e-02' vid_stop_22='-
    vid_noise+-4.3659324e-02'
.param vid_start_21='vid_noise+-4.8545729e-02' vid_stop_21='-
    vid_noise+-4.8545729e-02'
.param vid_start_20='vid_noise+-5.3428821e-02' vid_stop_20='-
    vid_noise+-5.3428821e-02'
.param vid_start_19='vid_noise+-5.8357190e-02' vid_stop_19='-
    vid_noise+-5.8357190e-02'
.param vid_start_18='vid_noise+-6.3295966e-02' vid_stop_18='-
    vid_noise+-6.3295966e-02'
.param vid_start_17='vid_noise+-6.8252648e-02' vid_stop_17='-
    vid_noise+-6.8252648e-02'
.param vid_start_16='vid_noise+-7.3236715e-02' vid_stop_16='-
    vid_noise+-7.3236715e-02'
.param vid_start_15='vid_noise+-7.8241682e-02' vid_stop_15='-
    vid_noise+-7.8241682e-02'
.param vid_start_14='vid_noise+-8.3272059e-02' vid_stop_14='-
    vid_noise+-8.3272059e-02'
.param vid_start_13='vid_noise+-8.8300299e-02' vid_stop_13='-
    vid_noise+-8.8300299e-02'
.param vid_start_12='vid_noise+-9.3413144e-02' vid_stop_12='-
    vid_noise+-9.3413144e-02'
.param vid_start_11='vid_noise+-9.8463285e-02' vid_stop_11='-
    vid_noise+-9.8463285e-02'
.param vid_start_10='vid_noise+-1.0359673e-01' vid_stop_10='-
    vid_noise+-1.0359673e-01'
.param vid_start_9='vid_noise+-1.0874526e-01' vid_stop_9='-
    vid_noise+-1.0874526e-01'
.param vid_start_8='vid_noise+-1.1384806e-01' vid_stop_8='-
    vid_noise+-1.1384806e-01'
.param vid_start_7='vid_noise+-1.1900904e-01' vid_stop_7='-
    vid_noise+-1.1900904e-01'
.param vid_start_6='vid_noise+-1.2420357e-01' vid_stop_6='-
    vid_noise+-1.2420357e-01'
.param vid_start_5='vid_noise+-1.2932084e-01' vid_stop_5='-
    vid_noise+-1.2932084e-01'
.param vid_start_4='vid_noise+-1.3459414e-01' vid_stop_4='-
    vid_noise+-1.3459414e-01'
.param vid_start_3='vid_noise+-1.3976896e-01' vid_stop_3='-
    vid_noise+-1.3976896e-01'
.param vid_start_2='vid_noise+-1.4495092e-01' vid_stop_2='-
    vid_noise+-1.4495092e-01'
.param vid_start_1='vid_noise+-1.5013130e-01' vid_stop_1='-
    vid_noise+-1.5013130e-01'
.param vid_start_0='vid_noise+-1.5523863e-01' vid_stop_0='-
    vid_noise+-1.5523863e-01'
**********************************************************
.title 'ooooo ooooo - Scan for Threshold Voltage Determination'
******
.param vtca4=0 vtca3=0 vtca2=0 vtca1=0 vtca0=0
.param vtcb4=0 vtcb3=0 vtcb2=0 vtcb1=0 vtcb0=0
******
```

```
.param vid_start=vid_start_31 vid_stop=vid_stop_31 vid_incr=
    vid_noise_incr
******
.tran 0.1p '2*T+2n' sweep vid vid_start vid_stop vid_incr
**********************************************************************
**********************************************************************
.alter .title '00001 00000 - Scan for Threshold Voltage
    Determination'
.param vtca4=0 vtca3=0 vtca2=0 vtca1=0 vtca0=1
.param vid_start=vid_start_30 vid_stop=vid_stop_30 vid_incr=
    vid_noise_incr
**********************************************************************
**********************************************************************
.alter .title '000010 00000 - Scan for Threshold Voltage
    Determination'
.param vtca4=0 vtca3=0 vtca2=0 vtca1=1 vtca0=0
.param vid_start=vid_start_29 vid_stop=vid_stop_29 vid_incr=
    vid_noise_incr
**********************************************************************
**********************************************************************
.alter .title '000011 00000 - Scan for Threshold Voltage
    Determination'
.param vtca4=0 vtca3=0 vtca2=0 vtca1=1 vtca0=1
.param vid_start=vid_start_28 vid_stop=vid_stop_28 vid_incr=
    vid_noise_incr
**********************************************************************
**********************************************************************
.alter .title '000100 00000 - Scan for Threshold Voltage
    Determination'
.param vtca4=0 vtca3=0 vtca2=1 vtca1=0 vtca0=0
.param vid_start=vid_start_27 vid_stop=vid_stop_27 vid_incr=
    vid_noise_incr
**********************************************************************
**********************************************************************
.alter .title '000101 00000 - Scan for Threshold Voltage
    Determination'
.param vtca4=0 vtca3=0 vtca2=1 vtca1=0 vtca0=1
.param vid_start=vid_start_26 vid_stop=vid_stop_26 vid_incr=
    vid_noise_incr
**********************************************************************
**********************************************************************
.alter .title '000110 00000 - Scan for Threshold Voltage
    Determination'
.param vtca4=0 vtca3=0 vtca2=1 vtca1=1 vtca0=0
.param vid_start=vid_start_25 vid_stop=vid_stop_25 vid_incr=
    vid_noise_incr
**********************************************************************
**********************************************************************
.alter .title '000111 00000 - Scan for Threshold Voltage
    Determination'
.param vtca4=0 vtca3=0 vtca2=1 vtca1=1 vtca0=1
```

```
.param vid_start=vid_start_24 vid_stop=vid_stop_24 vid_incr=
    vid_noise_incr
**********************************************************************
**********************************************************************
.alter .title '001000 00000 - Scan for Threshold Voltage
    Determination'
.param vtca4=0 vtca3=1 vtca2=0 vtca1=0 vtca0=0
.param vid_start=vid_start_23 vid_stop=vid_stop_23 vid_incr=
    vid_noise_incr
**********************************************************************
.alter .title '001001 00000 - Scan for Threshold Voltage
    Determination'
.param vtca4=0 vtca3=1 vtca2=0 vtca1=0 vtca0=1
.param vid_start=vid_start_22 vid_stop=vid_stop_22 vid_incr=
    vid_noise_incr
**********************************************************************
.alter .title '001010 00000 - Scan for Threshold Voltage
    Determination'
.param vtca4=0 vtca3=1 vtca2=0 vtca1=1 vtca0=0
.param vid_start=vid_start_21 vid_stop=vid_stop_21 vid_incr=
    vid_noise_incr
**********************************************************************
.alter .title '001011 00000 - Scan for Threshold Voltage
    Determination'
.param vtca4=0 vtca3=1 vtca2=0 vtca1=1 vtca0=1
.param vid_start=vid_start_20 vid_stop=vid_stop_20 vid_incr=
    vid_noise_incr
**********************************************************************
.alter .title '001100 00000 - Scan for Threshold Voltage
    Determination'
.param vtca4=0 vtca3=1 vtca2=1 vtca1=0 vtca0=0
.param vid_start=vid_start_19 vid_stop=vid_stop_19 vid_incr=
    vid_noise_incr
**********************************************************************
.alter .title '001101 00000 - Scan for Threshold Voltage
    Determination'
.param vtca4=0 vtca3=1 vtca2=1 vtca1=0 vtca0=1
.param vid_start=vid_start_18 vid_stop=vid_stop_18 vid_incr=
    vid_noise_incr
**********************************************************************
.alter .title '001110 00000 - Scan for Threshold Voltage
    Determination'
.param vtca4=0 vtca3=1 vtca2=1 vtca1=1 vtca0=0
.param vid_start=vid_start_17 vid_stop=vid_stop_17 vid_incr=
    vid_noise_incr
**********************************************************************
.alter .title '001111 00000 - Scan for Threshold Voltage
    Determination'
.param vtca4=0 vtca3=1 vtca2=1 vtca1=1 vtca0=1
.param vid_start=vid_start_16 vid_stop=vid_stop_16 vid_incr=
    vid_noise_incr
**********************************************************************
```

```
.alter .title '10000 00000 - Scan for Threshold Voltage
    Determination'
.param vtca4=1 vtca3=0 vtca2=0 vtca1=0 vtca0=0
.param vid_start=vid_start_15 vid_stop=vid_stop_15 vid_incr=
    vid_noise_incr
*****************************************************************
.alter .title '10001 00000 - Scan for Threshold Voltage
    Determination'
.param vtca4=1 vtca3=0 vtca2=0 vtca1=0 vtca0=1
.param vid_start=vid_start_14 vid_stop=vid_stop_14 vid_incr=
    vid_noise_incr
*****************************************************************
.alter .title '10010 00000 - Scan for Threshold Voltage
    Determination'
.param vtca4=1 vtca3=0 vtca2=0 vtca1=1 vtca0=0
.param vid_start=vid_start_13 vid_stop=vid_stop_13 vid_incr=
    vid_noise_incr
*****************************************************************
.alter .title '10011 00000 - Scan for Threshold Voltage
    Determination'
.param vtca4=1 vtca3=0 vtca2=0 vtca1=1 vtca0=1
.param vid_start=vid_start_12 vid_stop=vid_stop_12 vid_incr=
    vid_noise_incr
*****************************************************************
.alter .title '10100 00000 - Scan for Threshold Voltage
    Determination'
.param vtca4=1 vtca3=0 vtca2=1 vtca1=0 vtca0=0
.param vid_start=vid_start_11 vid_stop=vid_stop_11 vid_incr=
    vid_noise_incr
*****************************************************************
.alter .title '10101 00000 - Scan for Threshold Voltage
    Determination'
.param vtca4=1 vtca3=0 vtca2=1 vtca1=0 vtca0=1
.param vid_start=vid_start_10 vid_stop=vid_stop_10 vid_incr=
    vid_noise_incr
*****************************************************************
.alter .title '10110 00000 - Scan for Threshold Voltage
    Determination'
.param vtca4=1 vtca3=0 vtca2=1 vtca1=1 vtca0=0
.param vid_start=vid_start_9 vid_stop=vid_stop_9 vid_incr=
    vid_noise_incr
*****************************************************************
.alter .title '10111 00000 - Scan for Threshold Voltage
    Determination'
.param vtca4=1 vtca3=0 vtca2=1 vtca1=1 vtca0=1
.param vid_start=vid_start_8 vid_stop=vid_stop_8 vid_incr=
    vid_noise_incr
*****************************************************************
.alter .title '11000 00000 - Scan for Threshold Voltage
    Determination'
.param vtca4=1 vtca3=1 vtca2=0 vtca1=0 vtca0=0
```

```
.param vid_start=vid_start_7 vid_stop=vid_stop_7 vid_incr=
    vid_noise_incr
**********************************************************************
.alter .title '11001 00000 - Scan for Threshold Voltage
    Determination'
.param vtca4=1 vtca3=1 vtca2=0 vtca1=0 vtca0=1
.param vid_start=vid_start_6 vid_stop=vid_stop_6 vid_incr=
    vid_noise_incr
**********************************************************************
.alter .title '11010 00000 - Scan for Threshold Voltage
    Determination'
.param vtca4=1 vtca3=1 vtca2=0 vtca1=1 vtca0=0
.param vid_start=vid_start_5 vid_stop=vid_stop_5 vid_incr=
    vid_noise_incr
**********************************************************************
.alter .title '11011 00000 - Scan for Threshold Voltage
    Determination'
.param vtca4=1 vtca3=1 vtca2=0 vtca1=1 vtca0=1
.param vid_start=vid_start_4 vid_stop=vid_stop_4 vid_incr=
    vid_noise_incr
**********************************************************************
.alter .title '11100 00000 - Scan for Threshold Voltage
    Determination'
.param vtca4=1 vtca3=1 vtca2=1 vtca1=0 vtca0=0
.param vid_start=vid_start_3 vid_stop=vid_stop_3 vid_incr=
    vid_noise_incr
**********************************************************************
.alter .title '11101 00000 - Scan for Threshold Voltage
    Determination'
.param vtca4=1 vtca3=1 vtca2=1 vtca1=0 vtca0=1
.param vid_start=vid_start_2 vid_stop=vid_stop_2 vid_incr=
    vid_noise_incr
**********************************************************************
.alter .title '11110 00000 - Scan for Threshold Voltage
    Determination'
.param vtca4=1 vtca3=1 vtca2=1 vtca1=1 vtca0=0
.param vid_start=vid_start_1 vid_stop=vid_stop_1 vid_incr=
    vid_noise_incr
**********************************************************************
.alter .title '11111 00000 - Scan for Threshold Voltage
    Determination'
.param vtca4=1 vtca3=1 vtca2=1 vtca1=1 vtca0=1
.param vid_start=vid_start_0 vid_stop=vid_stop_0 vid_incr=
    vid_noise_incr
**********************************************************************
**********************************************************************
.end
```

A.7.2 *Simulation script for* `oheader.sp` *file*

The `oheader.sp` file contains the directives for simulations made by using the Bisection optimization mode.

Listing 3: `oheader.sp` - Bisection optimization mode

```
*****************************************************************
**** Header for NO MonteCarlo Simulation
*****************************************************************
**** Mismatch for MonteCarlo Simulation
******
.param sigma=0 .param process=0 mismatch=0
*****************************************************************
.param vid=0 vid_start=0 vid_stop=0 vid_incr=0
*****************************************************************
.param vid_noise=1.0mv vid_noise_incr=1.0mv
*****************************************************************
.param vid_start_31='vid_noise+-3.05e-06' vid_stop_31='-vid_noise
    +-3.05e-06'
.param vid_start_30='vid_noise+-4.87e-03' vid_stop_30='-vid_noise
    +-4.87e-03'
.param vid_start_29='vid_noise+-9.73e-03' vid_stop_29='-vid_noise
    +-9.73e-03'
.param vid_start_28='vid_noise+-1.46e-02' vid_stop_28='-vid_noise
    +-1.46e-02'
.param vid_start_27='vid_noise+-1.94e-02' vid_stop_27='-vid_noise
    +-1.94e-02'
.param vid_start_26='vid_noise+-2.42e-02' vid_stop_26='-vid_noise
    +-2.42e-02'
.param vid_start_25='vid_noise+-2.91e-02' vid_stop_25='-vid_noise
    +-2.91e-02'
.param vid_start_24='vid_noise+-3.39e-02' vid_stop_24='-vid_noise
    +-3.39e-02'
.param vid_start_23='vid_noise+-3.88e-02' vid_stop_23='-vid_noise
    +-3.88e-02'
.param vid_start_22='vid_noise+-4.37e-02' vid_stop_22='-vid_noise
    +-4.37e-02'
.param vid_start_21='vid_noise+-4.85e-02' vid_stop_21='-vid_noise
    +-4.85e-02'
.param vid_start_20='vid_noise+-5.34e-02' vid_stop_20='-vid_noise
    +-5.34e-02'
.param vid_start_19='vid_noise+-5.84e-02' vid_stop_19='-vid_noise
    +-5.84e-02'
.param vid_start_18='vid_noise+-6.33e-02' vid_stop_18='-vid_noise
    +-6.33e-02'
.param vid_start_17='vid_noise+-6.83e-02' vid_stop_17='-vid_noise
    +-6.83e-02'
.param vid_start_16='vid_noise+-7.32e-02' vid_stop_16='-vid_noise
    +-7.32e-02'
.param vid_start_15='vid_noise+-7.82e-02' vid_stop_15='-vid_noise
    +-7.82e-02'
```

```
.param vid_start_14='vid_noise+-8.33e-02' vid_stop_14='-vid_noise
    +-8.33e-02'
.param vid_start_13='vid_noise+-8.83e-02' vid_stop_13='-vid_noise
    +-8.83e-02'
.param vid_start_12='vid_noise+-9.34e-02' vid_stop_12='-vid_noise
    +-9.34e-02'
.param vid_start_11='vid_noise+-9.85e-02' vid_stop_11='-vid_noise
    +-9.85e-02'
.param vid_start_10='vid_noise+-1.04e-01' vid_stop_10='-vid_noise
    +-1.04e-01'
.param vid_start_9='vid_noise+-1.09e-01' vid_stop_9='-vid_noise
    +-1.09e-01'
.param vid_start_8='vid_noise+-1.14e-01' vid_stop_8='-vid_noise
    +-1.14e-01'
.param vid_start_7='vid_noise+-1.19e-01' vid_stop_7='-vid_noise
    +-1.19e-01'
.param vid_start_6='vid_noise+-1.24e-01' vid_stop_6='-vid_noise
    +-1.24e-01'
.param vid_start_5='vid_noise+-1.29e-01' vid_stop_5='-vid_noise
    +-1.29e-01'
.param vid_start_4='vid_noise+-1.34e-01' vid_stop_4='-vid_noise
    +-1.34e-01'
.param vid_start_3='vid_noise+-1.40e-01' vid_stop_3='-vid_noise
    +-1.40e-01'
.param vid_start_2='vid_noise+-1.45e-01' vid_stop_2='-vid_noise
    +-1.45e-01'
.param vid_start_1='vid_noise+-1.50e-01' vid_stop_1='-vid_noise
    +-1.50e-01'
.param vid_start_0='vid_noise+-1.55e-01' vid_stop_0='-vid_noise
    +-1.55e-01'
**********************************************************************
.title '00000 00000 - Scan for Threshold Voltage Determination'
******
.param vtca4=0 vtca3=0 vtca2=0 vtca1=0 vtca0=0
.param vtcb4=0 vtcb3=0 vtcb2=0 vtcb1=0 vtcb0=0
******
.param vid_start=vid_start_31 vid_stop=vid_stop_31 vid_incr=
    vid_noise_incr
******
.param vid=optadc(vid_start,vid_start,vid_stop)
******
.tran 0.1p '2*T+2n' sweep optimize=optadc results=dout1-dout2
    model=opt3
**********************************************************************
**********************************************************************
.alter .title '00001 00000 - Scan for Threshold Voltage
    Determination'
.param vtca4=0 vtca3=0 vtca2=0 vtca1=0 vtca0=1
.param vid_start=vid_start_30 vid_stop=vid_stop_30 vid_incr=
    vid_noise_incr
**********************************************************************
**********************************************************************
```

```
.alter .title '000010 00000 - Scan for Threshold Voltage
    Determination'
.param vtca4=0 vtca3=0 vtca2=0 vtca1=1 vtca0=0
.param vid_start=vid_start_29 vid_stop=vid_stop_29 vid_incr=
    vid_noise_incr
**********************************************************************
**********************************************************************
.alter .title '000011 00000 - Scan for Threshold Voltage
    Determination'
.param vtca4=0 vtca3=0 vtca2=0 vtca1=1 vtca0=1
.param vid_start=vid_start_28 vid_stop=vid_stop_28 vid_incr=
    vid_noise_incr
**********************************************************************
**********************************************************************
.alter .title '000100 00000 - Scan for Threshold Voltage
    Determination'
.param vtca4=0 vtca3=0 vtca2=1 vtca1=0 vtca0=0
.param vid_start=vid_start_27 vid_stop=vid_stop_27 vid_incr=
    vid_noise_incr
**********************************************************************
**********************************************************************
.alter .title '000101 00000 - Scan for Threshold Voltage
    Determination'
.param vtca4=0 vtca3=0 vtca2=1 vtca1=0 vtca0=1
.param vid_start=vid_start_26 vid_stop=vid_stop_26 vid_incr=
    vid_noise_incr
**********************************************************************
**********************************************************************
.alter .title '000110 00000 - Scan for Threshold Voltage
    Determination'
.param vtca4=0 vtca3=0 vtca2=1 vtca1=1 vtca0=0
.param vid_start=vid_start_25 vid_stop=vid_stop_25 vid_incr=
    vid_noise_incr
**********************************************************************
**********************************************************************
.alter .title '000111 00000 - Scan for Threshold Voltage
    Determination'
.param vtca4=0 vtca3=0 vtca2=1 vtca1=1 vtca0=1
.param vid_start=vid_start_24 vid_stop=vid_stop_24 vid_incr=
    vid_noise_incr
**********************************************************************
**********************************************************************
.alter .title '001000 00000 - Scan for Threshold Voltage
    Determination'
.param vtca4=0 vtca3=1 vtca2=0 vtca1=0 vtca0=0
.param vid_start=vid_start_23 vid_stop=vid_stop_23 vid_incr=
    vid_noise_incr
**********************************************************************
.alter .title '001001 00000 - Scan for Threshold Voltage
    Determination'
.param vtca4=0 vtca3=1 vtca2=0 vtca1=0 vtca0=1
```

```
.param vid_start=vid_start_22 vid_stop=vid_stop_22 vid_incr=
    vid_noise_incr
**********************************************************************
.alter .title '001010 00000 - Scan for Threshold Voltage
    Determination'
.param vtca4=0 vtca3=1 vtca2=0 vtca1=1 vtca0=0
.param vid_start=vid_start_21 vid_stop=vid_stop_21 vid_incr=
    vid_noise_incr
**********************************************************************
.alter .title '001011 00000 - Scan for Threshold Voltage
    Determination'
.param vtca4=0 vtca3=1 vtca2=0 vtca1=1 vtca0=1
.param vid_start=vid_start_20 vid_stop=vid_stop_20 vid_incr=
    vid_noise_incr
**********************************************************************
.alter .title '001100 00000 - Scan for Threshold Voltage
    Determination'
.param vtca4=0 vtca3=1 vtca2=1 vtca1=0 vtca0=0
.param vid_start=vid_start_19 vid_stop=vid_stop_19 vid_incr=
    vid_noise_incr
**********************************************************************
.alter .title '001101 00000 - Scan for Threshold Voltage
    Determination'
.param vtca4=0 vtca3=1 vtca2=1 vtca1=0 vtca0=1
.param vid_start=vid_start_18 vid_stop=vid_stop_18 vid_incr=
    vid_noise_incr
**********************************************************************
.alter .title '001110 00000 - Scan for Threshold Voltage
    Determination'
.param vtca4=0 vtca3=1 vtca2=1 vtca1=1 vtca0=0
.param vid_start=vid_start_17 vid_stop=vid_stop_17 vid_incr=
    vid_noise_incr
**********************************************************************
.alter .title '001111 00000 - Scan for Threshold Voltage
    Determination'
.param vtca4=0 vtca3=1 vtca2=1 vtca1=1 vtca0=1
.param vid_start=vid_start_16 vid_stop=vid_stop_16 vid_incr=
    vid_noise_incr
**********************************************************************
.alter .title '10000 00000 - Scan for Threshold Voltage
    Determination'
.param vtca4=1 vtca3=0 vtca2=0 vtca1=0 vtca0=0
.param vid_start=vid_start_15 vid_stop=vid_stop_15 vid_incr=
    vid_noise_incr
**********************************************************************
.alter .title '10001 00000 - Scan for Threshold Voltage
    Determination'
.param vtca4=1 vtca3=0 vtca2=0 vtca1=0 vtca0=1
.param vid_start=vid_start_14 vid_stop=vid_stop_14 vid_incr=
    vid_noise_incr
**********************************************************************
```

```
.alter .title '10010 00000 - Scan for Threshold Voltage
    Determination'
.param vtca4=1 vtca3=0 vtca2=0 vtca1=1 vtca0=0
.param vid_start=vid_start_13 vid_stop=vid_stop_13 vid_incr=
    vid_noise_incr
*****************************************************************
.alter .title '10011 00000 - Scan for Threshold Voltage
    Determination'
.param vtca4=1 vtca3=0 vtca2=0 vtca1=1 vtca0=1
.param vid_start=vid_start_12 vid_stop=vid_stop_12 vid_incr=
    vid_noise_incr
*****************************************************************
.alter .title '10100 00000 - Scan for Threshold Voltage
    Determination'
.param vtca4=1 vtca3=0 vtca2=1 vtca1=0 vtca0=0
.param vid_start=vid_start_11 vid_stop=vid_stop_11 vid_incr=
    vid_noise_incr
*****************************************************************
.alter .title '10101 00000 - Scan for Threshold Voltage
    Determination'
.param vtca4=1 vtca3=0 vtca2=1 vtca1=0 vtca0=1
.param vid_start=vid_start_10 vid_stop=vid_stop_10 vid_incr=
    vid_noise_incr
*****************************************************************
.alter .title '10110 00000 - Scan for Threshold Voltage
    Determination'
.param vtca4=1 vtca3=0 vtca2=1 vtca1=1 vtca0=0
.param vid_start=vid_start_9 vid_stop=vid_stop_9 vid_incr=
    vid_noise_incr
*****************************************************************
.alter .title '10111 00000 - Scan for Threshold Voltage
    Determination'
.param vtca4=1 vtca3=0 vtca2=1 vtca1=1 vtca0=1
.param vid_start=vid_start_8 vid_stop=vid_stop_8 vid_incr=
    vid_noise_incr
*****************************************************************
.alter .title '11000 00000 - Scan for Threshold Voltage
    Determination'
.param vtca4=1 vtca3=1 vtca2=0 vtca1=0 vtca0=0
.param vid_start=vid_start_7 vid_stop=vid_stop_7 vid_incr=
    vid_noise_incr
*****************************************************************
.alter .title '11001 00000 - Scan for Threshold Voltage
    Determination'
.param vtca4=1 vtca3=1 vtca2=0 vtca1=0 vtca0=1
.param vid_start=vid_start_6 vid_stop=vid_stop_6 vid_incr=
    vid_noise_incr
*****************************************************************
.alter .title '11010 00000 - Scan for Threshold Voltage
    Determination'
.param vtca4=1 vtca3=1 vtca2=0 vtca1=1 vtca0=0
```

```
.param vid_start=vid_start_5 vid_stop=vid_stop_5 vid_incr=
    vid_noise_incr
**********************************************************************
.alter .title '11011 00000 - Scan for Threshold Voltage
    Determination'
.param vtca4=1 vtca3=1 vtca2=0 vtca1=1 vtca0=1
.param vid_start=vid_start_4 vid_stop=vid_stop_4 vid_incr=
    vid_noise_incr
**********************************************************************
.alter .title '11100 00000 - Scan for Threshold Voltage
    Determination'
.param vtca4=1 vtca3=1 vtca2=1 vtca1=0 vtca0=0
.param vid_start=vid_start_3 vid_stop=vid_stop_3 vid_incr=
    vid_noise_incr
**********************************************************************
.alter .title '11101 00000 - Scan for Threshold Voltage
    Determination'
.param vtca4=1 vtca3=1 vtca2=1 vtca1=0 vtca0=1
.param vid_start=vid_start_2 vid_stop=vid_stop_2 vid_incr=
    vid_noise_incr
**********************************************************************
.alter .title '11110 00000 - Scan for Threshold Voltage
    Determination'
.param vtca4=1 vtca3=1 vtca2=1 vtca1=1 vtca0=0
.param vid_start=vid_start_1 vid_stop=vid_stop_1 vid_incr=
    vid_noise_incr
**********************************************************************
.alter .title '11111 00000 - Scan for Threshold Voltage
    Determination'
.param vtca4=1 vtca3=1 vtca2=1 vtca1=1 vtca0=1
.param vid_start=vid_start_0 vid_stop=vid_stop_0 vid_incr=
    vid_noise_incr
**********************************************************************
**********************************************************************
.end
```

A.7.3 *Simulation script for `mheader` file*

The `mheader.sp` file contains the directives for mismatch analysis led through Monte Carlo simulation technique.

Listing 4: `mheader.sp` - Monte Carlo optimization mode

```
**********************************************************************
**** Header for MonteCarlo Simulation
**********************************************************************
**** Mismatch for MonteCarlo Simulation
******
.param sigma=3 .param process=0 mismatch=1
**********************************************************************
****** Analysis conditions
```

```
******
.param vid_noise=0.0mv vid_noise_incr=1.0mv
************************************************************
.param vid=-100mv vid_start=+1000.0mv vid_stop=-1000.0mv vid_incr
    =-0.1mv
******
.param vid=optadc(vid_start,vid_start,vid_stop)
******
.tran 0.1p '2*T+2n' sweep optimize=optadc results=dout1-dout2
    model=opt3 monte=30
************************************************************
.param vid_noise=0.0mv vid_noise_incr=1.0mv vid_fs=200mv
************************************************************
.param vid_start_62='vid_noise+vid_fs' vid_stop_62='-vid_noise-
    vid_fs'
.param vid_start_61='vid_noise+vid_fs' vid_stop_61='-vid_noise-
    vid_fs'
.param vid_start_60='vid_noise+vid_fs' vid_stop_60='-vid_noise-
    vid_fs'
.param vid_start_59='vid_noise+vid_fs' vid_stop_59='-vid_noise-
    vid_fs'
.param vid_start_58='vid_noise+vid_fs' vid_stop_58='-vid_noise-
    vid_fs'
.param vid_start_57='vid_noise+vid_fs' vid_stop_57='-vid_noise-
    vid_fs'
.param vid_start_56='vid_noise+vid_fs' vid_stop_56='-vid_noise-
    vid_fs'
.param vid_start_55='vid_noise+vid_fs' vid_stop_55='-vid_noise-
    vid_fs'
.param vid_start_54='vid_noise+vid_fs' vid_stop_54='-vid_noise-
    vid_fs'
.param vid_start_53='vid_noise+vid_fs' vid_stop_53='-vid_noise-
    vid_fs'
.param vid_start_52='vid_noise+vid_fs' vid_stop_52='-vid_noise-
    vid_fs'
.param vid_start_51='vid_noise+vid_fs' vid_stop_51='-vid_noise-
    vid_fs'
.param vid_start_50='vid_noise+vid_fs' vid_stop_50='-vid_noise-
    vid_fs'
.param vid_start_49='vid_noise+vid_fs' vid_stop_49='-vid_noise-
    vid_fs'
.param vid_start_48='vid_noise+vid_fs' vid_stop_48='-vid_noise-
    vid_fs'
.param vid_start_47='vid_noise+vid_fs' vid_stop_47='-vid_noise-
    vid_fs'
.param vid_start_46='vid_noise+vid_fs' vid_stop_46='-vid_noise-
    vid_fs'
.param vid_start_45='vid_noise+vid_fs' vid_stop_45='-vid_noise-
    vid_fs'
.param vid_start_44='vid_noise+vid_fs' vid_stop_44='-vid_noise-
    vid_fs'
```

```
.param vid_start_43='vid_noise+vid_fs' vid_stop_43='-vid_noise-
    vid_fs'
.param vid_start_42='vid_noise+vid_fs' vid_stop_42='-vid_noise-
    vid_fs'
.param vid_start_41='vid_noise+vid_fs' vid_stop_41='-vid_noise-
    vid_fs'
.param vid_start_40='vid_noise+vid_fs' vid_stop_40='-vid_noise-
    vid_fs'
.param vid_start_39='vid_noise+vid_fs' vid_stop_39='-vid_noise-
    vid_fs'
.param vid_start_38='vid_noise+vid_fs' vid_stop_38='-vid_noise-
    vid_fs'
.param vid_start_37='vid_noise+vid_fs' vid_stop_37='-vid_noise-
    vid_fs'
.param vid_start_36='vid_noise+vid_fs' vid_stop_36='-vid_noise-
    vid_fs'
.param vid_start_35='vid_noise+vid_fs' vid_stop_35='-vid_noise-
    vid_fs'
.param vid_start_34='vid_noise+vid_fs' vid_stop_34='-vid_noise-
    vid_fs'
.param vid_start_33='vid_noise+vid_fs' vid_stop_33='-vid_noise-
    vid_fs'
.param vid_start_32='vid_noise+vid_fs' vid_stop_32='-vid_noise-
    vid_fs'
**********************************************************************
.param vid_start_31='vid_noise+vid_fs' vid_stop_31='-vid_noise-
    vid_fs'
**********************************************************************
.param vid_start_30='vid_noise+vid_fs' vid_stop_30='-vid_noise-
    vid_fs'
.param vid_start_29='vid_noise+vid_fs' vid_stop_29='-vid_noise-
    vid_fs'
.param vid_start_28='vid_noise+vid_fs' vid_stop_28='-vid_noise-
    vid_fs'
.param vid_start_27='vid_noise+vid_fs' vid_stop_27='-vid_noise-
    vid_fs'
.param vid_start_26='vid_noise+vid_fs' vid_stop_26='-vid_noise-
    vid_fs'
.param vid_start_25='vid_noise+vid_fs' vid_stop_25='-vid_noise-
    vid_fs'
.param vid_start_24='vid_noise+vid_fs' vid_stop_24='-vid_noise-
    vid_fs'
.param vid_start_23='vid_noise+vid_fs' vid_stop_23='-vid_noise-
    vid_fs'
.param vid_start_22='vid_noise+vid_fs' vid_stop_22='-vid_noise-
    vid_fs'
.param vid_start_21='vid_noise+vid_fs' vid_stop_21='-vid_noise-
    vid_fs'
.param vid_start_20='vid_noise+vid_fs' vid_stop_20='-vid_noise-
    vid_fs'
.param vid_start_19='vid_noise+vid_fs' vid_stop_19='-vid_noise-
    vid_fs'
```

```
.param vid_start_18='vid_noise+vid_fs' vid_stop_18='-vid_noise-
    vid_fs'
.param vid_start_17='vid_noise+vid_fs' vid_stop_17='-vid_noise-
    vid_fs'
.param vid_start_16='vid_noise+vid_fs' vid_stop_16='-vid_noise-
    vid_fs'
.param vid_start_15='vid_noise+vid_fs' vid_stop_15='-vid_noise-
    vid_fs'
.param vid_start_14='vid_noise+vid_fs' vid_stop_14='-vid_noise-
    vid_fs'
.param vid_start_13='vid_noise+vid_fs' vid_stop_13='-vid_noise-
    vid_fs'
.param vid_start_12='vid_noise+vid_fs' vid_stop_12='-vid_noise-
    vid_fs'
.param vid_start_11='vid_noise+vid_fs' vid_stop_11='-vid_noise-
    vid_fs'
.param vid_start_10='vid_noise+vid_fs' vid_stop_10='-vid_noise-
    vid_fs'
.param vid_start_9 ='vid_noise+vid_fs' vid_stop_9 ='-vid_noise-
    vid_fs'
.param vid_start_8 ='vid_noise+vid_fs' vid_stop_8 ='-vid_noise-
    vid_fs'
.param vid_start_7 ='vid_noise+vid_fs' vid_stop_7 ='-vid_noise-
    vid_fs'
.param vid_start_6 ='vid_noise+vid_fs' vid_stop_6 ='-vid_noise-
    vid_fs'
.param vid_start_5 ='vid_noise+vid_fs' vid_stop_5 ='-vid_noise-
    vid_fs'
.param vid_start_4 ='vid_noise+vid_fs' vid_stop_4 ='-vid_noise-
    vid_fs'
.param vid_start_3 ='vid_noise+vid_fs' vid_stop_3 ='-vid_noise-
    vid_fs'
.param vid_start_2 ='vid_noise+vid_fs' vid_stop_2 ='-vid_noise-
    vid_fs'
.param vid_start_1 ='vid_noise+vid_fs' vid_stop_1 ='-vid_noise-
    vid_fs'
.param vid_start_0 ='vid_noise+vid_fs' vid_stop_0 ='-vid_noise-
    vid_fs'
**********************************************************************
**********************************************************************
**********************************************************************
****** B Side Code ******
**********************************************************************
**********************************************************************
.param vtca4=0 vtca3=0 vtca2=0 vtca1=0 vtca0=0
.param vtcb4=0 vtcb3=0 vtcb2=0 vtcb1=0 vtcb0=0
**********************************************************************
**********************************************************************
.title 'ooooo 11111 - Bisection for Threshold Voltage
    Determination'
.param vtcb4=1 vtcb3=1 vtcb2=1 vtcb1=1 vtcb0=1
```

```
.param vid_start=vid_start_62 vid_stop=vid_stop_62 vid_incr=
    vid_noise_incr
**********************************************************************
**********************************************************************
.alter .title 'ooooo 11110 - Bisection for Threshold Voltage
    Determination'
.param vtcb4=1 vtcb3=1 vtcb2=1 vtcb1=1 vtcb0=0
.param vid_start=vid_start_61 vid_stop=vid_stop_61 vid_incr=
    vid_noise_incr
**********************************************************************
**********************************************************************
.alter .title 'ooooo 11101 - Bisection for Threshold Voltage
    Determination'
.param vtcb4=1 vtcb3=1 vtcb2=1 vtcb1=0 vtcb0=1
.param vid_start=vid_start_60 vid_stop=vid_stop_60 vid_incr=
    vid_noise_incr
**********************************************************************
**********************************************************************
.alter .title 'ooooo 11100 - Bisection for Threshold Voltage
    Determination'
.param vtcb4=1 vtcb3=1 vtcb2=1 vtcb1=0 vtcb0=0
.param vid_start=vid_start_59 vid_stop=vid_stop_59 vid_incr=
    vid_noise_incr
**********************************************************************
**********************************************************************
.alter .title 'ooooo 11011 - Bisection for Threshold Voltage
    Determination'
.param vtcb4=1 vtcb3=1 vtcb2=0 vtcb1=1 vtcb0=1
.param vid_start=vid_start_58 vid_stop=vid_stop_58 vid_incr=
    vid_noise_incr
**********************************************************************
**********************************************************************
.alter .title 'ooooo 11010 - Bisection for Threshold Voltage
    Determination'
.param vtcb4=1 vtcb3=1 vtcb2=0 vtcb1=1 vtcb0=0
.param vid_start=vid_start_57 vid_stop=vid_stop_57 vid_incr=
    vid_noise_incr
**********************************************************************
**********************************************************************
.alter .title 'ooooo 11001 - Bisection for Threshold Voltage
    Determination'
.param vtcb4=1 vtcb3=1 vtcb2=0 vtcb1=0 vtcb0=1
.param vid_start=vid_start_56 vid_stop=vid_stop_56 vid_incr=
    vid_noise_incr
**********************************************************************
**********************************************************************
.alter .title 'ooooo 11000 - Bisection for Threshold Voltage
    Determination'
.param vtcb4=1 vtcb3=1 vtcb2=0 vtcb1=0 vtcb0=0
.param vid_start=vid_start_55 vid_stop=vid_stop_55 vid_incr=
    vid_noise_incr
**********************************************************************
```

```
**********************************************************************
.alter .title 'ooooo 10111 - Bisection for Threshold Voltage
    Determination'
.param vtcb4=1 vtcb3=0 vtcb2=1 vtcb1=1 vtcb0=1
.param vid_start=vid_start_54 vid_stop=vid_stop_54 vid_incr=
    vid_noise_incr
**********************************************************************
**********************************************************************
.alter .title 'ooooo 10110 - Bisection for Threshold Voltage
    Determination'
.param vtcb4=1 vtcb3=0 vtcb2=1 vtcb1=1 vtcb0=0
.param vid_start=vid_start_53 vid_stop=vid_stop_53 vid_incr=
    vid_noise_incr
**********************************************************************
**********************************************************************
.alter .title 'ooooo 10101 - Bisection for Threshold Voltage
    Determination'
.param vtcb4=1 vtcb3=0 vtcb2=1 vtcb1=0 vtcb0=1
.param vid_start=vid_start_52 vid_stop=vid_stop_52 vid_incr=
    vid_noise_incr
**********************************************************************
**********************************************************************
.alter .title 'ooooo 10100 - Bisection for Threshold Voltage
    Determination'
.param vtcb4=1 vtcb3=0 vtcb2=1 vtcb1=0 vtcb0=0
.param vid_start=vid_start_51 vid_stop=vid_stop_51 vid_incr=
    vid_noise_incr
**********************************************************************
**********************************************************************
.alter .title 'ooooo 10011 - Bisection for Threshold Voltage
    Determination'
.param vtcb4=1 vtcb3=0 vtcb2=0 vtcb1=1 vtcb0=1
.param vid_start=vid_start_50 vid_stop=vid_stop_50 vid_incr=
    vid_noise_incr
**********************************************************************
**********************************************************************
.alter .title 'ooooo 10010 - Bisection for Threshold Voltage
    Determination'
.param vtcb4=1 vtcb3=0 vtcb2=0 vtcb1=1 vtcb0=0
.param vid_start=vid_start_49 vid_stop=vid_stop_49 vid_incr=
    vid_noise_incr
**********************************************************************
**********************************************************************
.alter .title 'ooooo 10001 - Bisection for Threshold Voltage
    Determination'
.param vtcb4=1 vtcb3=0 vtcb2=0 vtcb1=0 vtcb0=1
.param vid_start=vid_start_48 vid_stop=vid_stop_48 vid_incr=
    vid_noise_incr
**********************************************************************
**********************************************************************
.alter .title 'ooooo 10000 - Bisection for Threshold Voltage
    Determination'
```

```
.param vtcb4=1 vtcb3=0 vtcb2=0 vtcb1=0 vtcb0=0
.param vid_start=vid_start_47 vid_stop=vid_stop_47 vid_incr=
    vid_noise_incr
**********************************************************************
**********************************************************************
.alter .title 'ooooo o1111 - Bisection for Threshold Voltage
    Determination'
.param vtcb4=0 vtcb3=1 vtcb2=1 vtcb1=1 vtcb0=1
.param vid_start=vid_start_46 vid_stop=vid_stop_46 vid_incr=
    vid_noise_incr
**********************************************************************
**********************************************************************
.alter .title 'ooooo o1110 - Bisection for Threshold Voltage
    Determination'
.param vtcb4=0 vtcb3=1 vtcb2=1 vtcb1=1 vtcb0=0
.param vid_start=vid_start_45 vid_stop=vid_stop_45 vid_incr=
    vid_noise_incr
**********************************************************************
**********************************************************************
.alter .title 'ooooo o1101 - Bisection for Threshold Voltage
    Determination'
.param vtcb4=0 vtcb3=1 vtcb2=1 vtcb1=0 vtcb0=1
.param vid_start=vid_start_44 vid_stop=vid_stop_44 vid_incr=
    vid_noise_incr
**********************************************************************
**********************************************************************
.alter .title 'ooooo o1100 - Bisection for Threshold Voltage
    Determination'
.param vtcb4=0 vtcb3=1 vtcb2=1 vtcb1=0 vtcb0=0
.param vid_start=vid_start_43 vid_stop=vid_stop_43 vid_incr=
    vid_noise_incr
**********************************************************************
**********************************************************************
.alter .title 'ooooo o1011 - Bisection for Threshold Voltage
    Determination'
.param vtcb4=0 vtcb3=1 vtcb2=0 vtcb1=1 vtcb0=1
.param vid_start=vid_start_42 vid_stop=vid_stop_42 vid_incr=
    vid_noise_incr
**********************************************************************
**********************************************************************
.alter .title 'ooooo o1010 - Bisection for Threshold Voltage
    Determination'
.param vtcb4=0 vtcb3=1 vtcb2=0 vtcb1=1 vtcb0=0
.param vid_start=vid_start_41 vid_stop=vid_stop_41 vid_incr=
    vid_noise_incr
**********************************************************************
**********************************************************************
.alter .title 'ooooo o1001 - Bisection for Threshold Voltage
    Determination'
.param vtcb4=0 vtcb3=1 vtcb2=0 vtcb1=0 vtcb0=1
.param vid_start=vid_start_40 vid_stop=vid_stop_40 vid_incr=
    vid_noise_incr
```

```
**********************************************************************
**********************************************************************
.alter .title '00000 01000 - Bisection for Threshold Voltage
    Determination'
.param vtcb4=0 vtcb3=1 vtcb2=0 vtcb1=0 vtcb0=0
.param vid_start=vid_start_39 vid_stop=vid_stop_39 vid_incr=
    vid_noise_incr
**********************************************************************
**********************************************************************
.alter .title '00000 00111 - Bisection for Threshold Voltage
    Determination'
.param vtcb4=0 vtcb3=0 vtcb2=1 vtcb1=1 vtcb0=1
.param vid_start=vid_start_38 vid_stop=vid_stop_38 vid_incr=
    vid_noise_incr
**********************************************************************
**********************************************************************
.alter .title '00000 00110 - Bisection for Threshold Voltage
    Determination'
.param vtcb4=0 vtcb3=0 vtcb2=1 vtcb1=1 vtcb0=0
.param vid_start=vid_start_37 vid_stop=vid_stop_37 vid_incr=
    vid_noise_incr
**********************************************************************
**********************************************************************
.alter .title '00000 00101 - Bisection for Threshold Voltage
    Determination'
.param vtcb4=0 vtcb3=0 vtcb2=1 vtcb1=0 vtcb0=1
.param vid_start=vid_start_36 vid_stop=vid_stop_36 vid_incr=
    vid_noise_incr
**********************************************************************
**********************************************************************
.alter .title '00000 00100 - Bisection for Threshold Voltage
    Determination'
.param vtcb4=0 vtcb3=0 vtcb2=1 vtcb1=0 vtcb0=0
.param vid_start=vid_start_35 vid_stop=vid_stop_35 vid_incr=
    vid_noise_incr
**********************************************************************
**********************************************************************
.alter .title '00000 00011 - Bisection for Threshold Voltage
    Determination'
.param vtcb4=0 vtcb3=0 vtcb2=0 vtcb1=1 vtcb0=1
.param vid_start=vid_start_34 vid_stop=vid_stop_34 vid_incr=
    vid_noise_incr
**********************************************************************
**********************************************************************
.alter .title '00000 00010 - Bisection for Threshold Voltage
    Determination'
.param vtcb4=0 vtcb3=0 vtcb2=0 vtcb1=1 vtcb0=0
.param vid_start=vid_start_33 vid_stop=vid_stop_33 vid_incr=
    vid_noise_incr
**********************************************************************
**********************************************************************
```

```
.alter .title '00000 00001 - Bisection for Threshold Voltage
    Determination'
.param vtcb4=0 vtcb3=0 vtcb2=0 vtcb1=0 vtcb0=1
.param vid_start=vid_start_32 vid_stop=vid_stop_32 vid_incr=
    vid_noise_incr
*****************************************************************
*****************************************************************
******
**** 0 Code ******
*****************************************************************
*****************************************************************
.alter .title '00000 00000 - Bisection for Threshold Voltage
    Determination'
.param vtca4=0 vtca3=0 vtca2=0 vtca1=0 vtca0=0
.param vtcb4=0 vtcb3=0 vtcb2=0 vtcb1=0 vtcb0=0
.param vid_start=vid_start_31 vid_stop=vid_stop_31 vid_incr=
    vid_noise_incr
*****************************************************************
*****************************************************************
******
**** A Side Code ******
*****************************************************************
*****************************************************************
.alter .title '00001 00000 - Bisection for Threshold Voltage
    Determination'
.param vtca4=0 vtca3=0 vtca2=0 vtca1=0 vtca0=1
.param vid_start=vid_start_30 vid_stop=vid_stop_30 vid_incr=
    vid_noise_incr
*****************************************************************
*****************************************************************
.alter .title '000010 00000 - Bisection for Threshold Voltage
    Determination'
.param vtca4=0 vtca3=0 vtca2=0 vtca1=1 vtca0=0
.param vid_start=vid_start_29 vid_stop=vid_stop_29 vid_incr=
    vid_noise_incr
*****************************************************************
*****************************************************************
.alter .title '000011 00000 - Bisection for Threshold Voltage
    Determination'
.param vtca4=0 vtca3=0 vtca2=0 vtca1=1 vtca0=1
.param vid_start=vid_start_28 vid_stop=vid_stop_28 vid_incr=
    vid_noise_incr
*****************************************************************
*****************************************************************
.alter .title '000100 00000 - Bisection for Threshold Voltage
    Determination'
.param vtca4=0 vtca3=0 vtca2=1 vtca1=0 vtca0=0
.param vid_start=vid_start_27 vid_stop=vid_stop_27 vid_incr=
    vid_noise_incr
*****************************************************************
*****************************************************************
```

```
.alter .title '000101 00000 - Bisection for Threshold Voltage
    Determination'
.param vtca4=0 vtca3=0 vtca2=1 vtca1=0 vtca0=1
.param vid_start=vid_start_26 vid_stop=vid_stop_26 vid_incr=
    vid_noise_incr
**********************************************************************
**********************************************************************
.alter .title '000110 00000 - Bisection for Threshold Voltage
    Determination'
.param vtca4=0 vtca3=0 vtca2=1 vtca1=1 vtca0=0
.param vid_start=vid_start_25 vid_stop=vid_stop_25 vid_incr=
    vid_noise_incr
**********************************************************************
**********************************************************************
.alter .title '000111 00000 - Bisection for Threshold Voltage
    Determination'
.param vtca4=0 vtca3=0 vtca2=1 vtca1=1 vtca0=1
.param vid_start=vid_start_24 vid_stop=vid_stop_24 vid_incr=
    vid_noise_incr
**********************************************************************
**********************************************************************
.alter .title '001000 00000 - Bisection for Threshold Voltage
    Determination'
.param vtca4=0 vtca3=1 vtca2=0 vtca1=0 vtca0=0
.param vid_start=vid_start_23 vid_stop=vid_stop_23 vid_incr=
    vid_noise_incr
**********************************************************************
.alter .title '001001 00000 - Bisection for Threshold Voltage
    Determination'
.param vtca4=0 vtca3=1 vtca2=0 vtca1=0 vtca0=1
.param vid_start=vid_start_22 vid_stop=vid_stop_22 vid_incr=
    vid_noise_incr
**********************************************************************
.alter .title '001010 00000 - Bisection for Threshold Voltage
    Determination'
.param vtca4=0 vtca3=1 vtca2=0 vtca1=1 vtca0=0
.param vid_start=vid_start_21 vid_stop=vid_stop_21 vid_incr=
    vid_noise_incr
**********************************************************************
.alter .title '001011 00000 - Bisection for Threshold Voltage
    Determination'
.param vtca4=0 vtca3=1 vtca2=0 vtca1=1 vtca0=1
.param vid_start=vid_start_20 vid_stop=vid_stop_20 vid_incr=
    vid_noise_incr
**********************************************************************
.alter .title '001100 00000 - Bisection for Threshold Voltage
    Determination'
.param vtca4=0 vtca3=1 vtca2=1 vtca1=0 vtca0=0
.param vid_start=vid_start_19 vid_stop=vid_stop_19 vid_incr=
    vid_noise_incr
**********************************************************************
```

```
.alter .title '001101 00000 - Bisection for Threshold Voltage
    Determination'
.param vtca4=0 vtca3=1 vtca2=1 vtca1=0 vtca0=1
.param vid_start=vid_start_18 vid_stop=vid_stop_18 vid_incr=
    vid_noise_incr
*****************************************************************
.alter .title '001110 00000 - Bisection for Threshold Voltage
    Determination'
.param vtca4=0 vtca3=1 vtca2=1 vtca1=1 vtca0=0
.param vid_start=vid_start_17 vid_stop=vid_stop_17 vid_incr=
    vid_noise_incr
*****************************************************************
.alter .title '001111 00000 - Bisection for Threshold Voltage
    Determination'
.param vtca4=0 vtca3=1 vtca2=1 vtca1=1 vtca0=1
.param vid_start=vid_start_16 vid_stop=vid_stop_16 vid_incr=
    vid_noise_incr
*****************************************************************
.alter .title '10000 00000 - Bisection for Threshold Voltage
    Determination'
.param vtca4=1 vtca3=0 vtca2=0 vtca1=0 vtca0=0
.param vid_start=vid_start_15 vid_stop=vid_stop_15 vid_incr=
    vid_noise_incr
*****************************************************************
.alter .title '10001 00000 - Bisection for Threshold Voltage
    Determination'
.param vtca4=1 vtca3=0 vtca2=0 vtca1=0 vtca0=1
.param vid_start=vid_start_14 vid_stop=vid_stop_14 vid_incr=
    vid_noise_incr
*****************************************************************
.alter .title '10010 00000 - Bisection for Threshold Voltage
    Determination'
.param vtca4=1 vtca3=0 vtca2=0 vtca1=1 vtca0=0
.param vid_start=vid_start_13 vid_stop=vid_stop_13 vid_incr=
    vid_noise_incr
*****************************************************************
.alter .title '10011 00000 - Bisection for Threshold Voltage
    Determination'
.param vtca4=1 vtca3=0 vtca2=0 vtca1=1 vtca0=1
.param vid_start=vid_start_12 vid_stop=vid_stop_12 vid_incr=
    vid_noise_incr
*****************************************************************
.alter .title '10100 00000 - Bisection for Threshold Voltage
    Determination'
.param vtca4=1 vtca3=0 vtca2=1 vtca1=0 vtca0=0
.param vid_start=vid_start_11 vid_stop=vid_stop_11 vid_incr=
    vid_noise_incr
*****************************************************************
.alter .title '10101 00000 - Bisection for Threshold Voltage
    Determination'
.param vtca4=1 vtca3=0 vtca2=1 vtca1=0 vtca0=1
```

```
.param vid_start=vid_start_10 vid_stop=vid_stop_10 vid_incr=
    vid_noise_incr
*****************************************************************
.alter .title '10110 00000 - Bisection for Threshold Voltage
    Determination'
.param vtca4=1 vtca3=0 vtca2=1 vtca1=1 vtca0=0
.param vid_start=vid_start_9 vid_stop=vid_stop_9 vid_incr=
    vid_noise_incr
*****************************************************************
.alter .title '10111 00000 - Bisection for Threshold Voltage
    Determination'
.param vtca4=1 vtca3=0 vtca2=1 vtca1=1 vtca0=1
.param vid_start=vid_start_8 vid_stop=vid_stop_8 vid_incr=
    vid_noise_incr
*****************************************************************
.alter .title '11000 00000 - Bisection for Threshold Voltage
    Determination'
.param vtca4=1 vtca3=1 vtca2=0 vtca1=0 vtca0=0
.param vid_start=vid_start_7 vid_stop=vid_stop_7 vid_incr=
    vid_noise_incr
*****************************************************************
.alter .title '11001 00000 - Bisection for Threshold Voltage
    Determination'
.param vtca4=1 vtca3=1 vtca2=0 vtca1=0 vtca0=1
.param vid_start=vid_start_6 vid_stop=vid_stop_6 vid_incr=
    vid_noise_incr
*****************************************************************
.alter .title '11010 00000 - Bisection for Threshold Voltage
    Determination'
.param vtca4=1 vtca3=1 vtca2=0 vtca1=1 vtca0=0
.param vid_start=vid_start_5 vid_stop=vid_stop_5 vid_incr=
    vid_noise_incr
*****************************************************************
.alter .title '11011 00000 - Bisection for Threshold Voltage
    Determination'
.param vtca4=1 vtca3=1 vtca2=0 vtca1=1 vtca0=1
.param vid_start=vid_start_4 vid_stop=vid_stop_4 vid_incr=
    vid_noise_incr
*****************************************************************
.alter .title '11100 00000 - Bisection for Threshold Voltage
    Determination'
.param vtca4=1 vtca3=1 vtca2=1 vtca1=0 vtca0=0
.param vid_start=vid_start_3 vid_stop=vid_stop_3 vid_incr=
    vid_noise_incr
*****************************************************************
.alter .title '11101 00000 - Bisection for Threshold Voltage
    Determination'
.param vtca4=1 vtca3=1 vtca2=1 vtca1=0 vtca0=1
.param vid_start=vid_start_2 vid_stop=vid_stop_2 vid_incr=
    vid_noise_incr
*****************************************************************
```

```
.alter .title '11110 00000 - Bisection for Threshold Voltage
    Determination'
.param vtca4=1 vtca3=1 vtca2=1 vtca1=1 vtca0=0
.param vid_start=vid_start_1 vid_stop=vid_stop_1 vid_incr=
    vid_noise_incr
*********************************************************************
.alter .title '11111 00000 - Bisection for Threshold Voltage
    Determination'
.param vtca4=1 vtca3=1 vtca2=1 vtca1=1 vtca0=1
.param vid_start=vid_start_0 vid_stop=vid_stop_0 vid_incr=
    vid_noise_incr
*********************************************************************
*********************************************************************
.end
```

A.8 circuits.inc FILE CODE

For a correct interpretation of Perl code containing the Netlist of the TC-Comparator, device references are related to the ones of Figure 3.1a, to follow presented in Table A.1.

Figure 3.1a	Listing 5	Figure 3.1a	Listing 5
M_{clk}	xm0	S_2	xms2
M_1	xm1	X_1	x1
M_2	xm2	X_2	x2
M_3	xm3	$TC_A(4...0)$	tca(4...0)
M_4	xm4	$TC_B(4...0)$	tcb(4...0)
M_5	xm5	O_1	xam1
M_6	xm6	O_2	xam2
S_1	xms1		

Table A.1: Device name reference for HSPICE codes

Listing 5: circuits.inc

```
.subckt inv in out
*********************************************************************
xm0 vdd in out vbgp P_10_SP w=0.12u l=0.08u
xm1 out in gnd vbgn N_10_SP w=0.12u l=0.08u
.ends
*********************************************************************
.subckt tc_comp ina inb nreset tca0 tca1 tca2 tca3 tca4 tcb0 tcb1
     tcb2 tcb3 tcb4 o1 o2 x0 x1 x2
*********************************************************************
xamvdd    vdd    vddx1    amp
xamgnd    gnd    gndx1    amp
```

```
xamvbgp   vbgp   vbgpx1   amp
xamvbgn   vbgn   vbgnx1   amp
xamnreset nreset nresetx1 amp
xamx0     x0     x0x1     amp
xamx1     x1     x1x1     amp
xamx2     x2     x2x1     amp
xamina    ina    inax1    amp
xaminb    inb    inbx1    amp
xamo1     o1     o1x1     amp
xamo2     o2     o2x1     amp
xamtca0   tca0   tca0x1   amp
xamtca1   tca1   tca1x1   amp
xamtca2   tca2   tca2x1   amp
xamtca3   tca3   tca3x1   amp
xamtca4   tca4   tca4x1   amp
xamtcb0   tcb0   tcb0x1   amp
xamtcb1   tcb1   tcb1x1   amp
xamtcb2   tcb2   tcb2x1   amp
xamtcb3   tcb3   tcb3x1   amp
xamtcb4   tcb4   tcb4x1   amp
*****************************************************************
xm0  vddx1 nresetx1    x0 vbgpx1 P_10_SP w='0.12u*m1' l=0.08u
xm1   x0x1    inax1    x1 vbgpx1 P_10_SP w='0.12u*m2' l=0.08u
xm2   x0x1    inbx1    x2 vbgpx1 P_10_SP w='0.12u*m2' l=0.08u
xm3   x1x1      o22   o11 vbgpx1 P_10_SP w='0.12u*m3' l=0.08u
xm4   x2x1      o11   o22 vbgpx1 P_10_SP w='0.12u*m3' l=0.08u
xm5    o11      o22 gndx1 vbgnx1 N_10_SP w='0.12u*m4' l=0.08u
xm6    o22      o11 gndx1 vbgnx1 N_10_SP w='0.12u*m4' l=0.08u
*****************************************************************
xam1 o11 o1x1 amp
xam2 o22 o2x1 amp
*****************************************************************
xms1  o1x1 nresetx1 gndx1 vbgnx1 N_10_SP w='0.12u*m5' l=0.08u
xms2  o2x1 nresetx1 gndx1 vbgnx1 N_10_SP w='0.12u*m5' l=0.08u
xms3  x1x1 nresetx1 gndx1 vbgnx1 N_10_SP w='0.12u*m5' l=0.08u
xms4  x2x1 nresetx1 gndx1 vbgnx1 N_10_SP w='0.12u*m5' l=0.08u
*****************************************************************
xam3 o1x1 o111 amp
xam4 o2x1 o222 amp
*****************************************************************
xma0 tca0x1 o111 tca0x1 vbgpx1 P_10_SP w='0.12u' l='0.32u' mf=
    mxma0 m=mxma0
xma1 tca1x1 o111 tca1x1 vbgpx1 P_10_SP w='0.12u' l='0.32u' mf=
    mxma1 m=mxma1
xma2 tca2x1 o111 tca2x1 vbgpx1 P_10_SP w='0.12u' l='0.32u' mf=
    mxma2 m=mxma2
xma3 tca3x1 o111 tca3x1 vbgpx1 P_10_SP w='0.12u' l='0.32u' mf=
    mxma3 m=mxma3
xma4 tca4x1 o111 tca4x1 vbgpx1 P_10_SP w='0.12u' l='0.32u' mf=
    mxma4 m=mxma4
*****************************************************************
```

```
xmb0 tcb0x1 o222 tcb0x1 vbgpx1 P_10_SP w='0.12u' l='0.32u' mf=
    mxmb0 m=mxmb0
xmb1 tcb1x1 o222 tcb1x1 vbgpx1 P_10_SP w='0.12u' l='0.32u' mf=
    mxmb1 m=mxmb1
xmb2 tcb2x1 o222 tcb2x1 vbgpx1 P_10_SP w='0.12u' l='0.32u' mf=
    mxmb2 m=mxmb2
xmb3 tcb3x1 o222 tcb3x1 vbgpx1 P_10_SP w='0.12u' l='0.32u' mf=
    mxmb3 m=mxmb3
xmb4 tcb4x1 o222 tcb4x1 vbgpx1 P_10_SP w='0.12u' l='0.32u' mf=
    mxmb4 m=mxmb4
*****************************************************************
cl0     o1x1 gndx1 '25f-3.1f'
cl1     o2x1 gndx1 '25f-3.1f'
cl2     x1x1 gndx1 15f
cl3     x2x1 gndx1 15f
cl4     x0x1 gndx1 4f
cl5 nresetx1 gndx1 5f
*****************************************************************
*ci0 o1 vdd c='(v(tca0)+v(tca1)*2+v(tca2)*4+v(tca3)*8+v(tca4)*16)
    *0.5f' ctype=1
*ci1 o2 vdd c='(v(tcb0)+v(tcb1)*2+v(tcb2)*4+v(tcb3)*8+v(tcb4)*16)
    *0.5f' ctype=1
*****************************************************************
.ends
*****************************************************************
.subckt amp i1 i2
*****************************************************************
vamp i1 i2 dc 0.0
*****************************************************************
.ends
```

BIBLIOGRAPHY

[1] G. Temes. "Micropower data converters: A tutorial". *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 57(no. 6):pp. 405–410, 2010. (Cited on pages ix, x, and 16.)

[2] L.L. Lewyn, T. Ytterdal, C. Wulff, and K. Martin. "Analog Circuit Design in Nanoscale CMOS Technologies". *Proceedings of the IEEE*, vol. 97(no. 10):pp. 1687–1714, Oct. 2009. (Cited on page ix.)

[3] R. Van de Plassche. *"CMOS Integrated Analog–to–Digital and Digital–to–Analog Converters"*. Kluwer Academic, 2th edition, 2003. (Cited on pages ix, 4, 5, 13, 15, and 18.)

[4] S. Saxena, C. Hess, H. Karbasi, A. Rossoni, S. Tonello, P. McNamara, S. Lucherini, S. Minehane, C. Dolainsky, and M. Quarantelli. "Variation in Transistor Performance and Leakage in Nanometer–Scale Technologies". *Electron Devices, IEEE Transactions*, vol. 55(no. 1):pp. 131–144, Jan. 2008. (Cited on page ix.)

[5] G. Van der Plas and B. Verbruggen. "A 150 MS/s 133μW 7 bit ADC in 90 nm Digital CMOS". *IEEE J. Solid–State Circuits*, vol. 43 (no. 12):pp. 2631–2640, Dec. 2008. (Cited on pages ix, x, and 24.)

[6] Y.Z. Lin, S.J. Chang, Y.T. Liu, C.C. Liu, and G.Y. Huang. "A 5b 800MS/s 2 mW asynchronous binary–search ADC in 65 nm CMOS". *IEEE Int. Solid–State Circuits Conf. Dig. Tech. Papers*, pages pp. 80–81, 2009. (Cited on pages ix, 24, and 28.)

[7] M. Chen and R. Brodersen. "A 6b 600 MS/s 5.3 mW asynchronous ADC in 0.13". *IEEE J. Solid–State Circuits*, vol. 41(no. 12):pp. 2669–2680, Dec. 2006. (Cited on pages ix, 20, 21, 27, and 34.)

[8] Craninckx and G. Van der Plas. "a 65 fj/conversion–step 0–to–50 MS/s 0–to–0.7 mw 9b charge–sharing SAR ADC in 90 nm digital CMOS". *IEEE Int. Solid–State Circuits Conf. Dig. Tech. Papers*, pages pp. 246–600, Feb. 2007. (Cited on pages ix and 21.)

[9] V. Giannini, P. Nuzzo, V. Chironi, A. Baschirotto, G. Van der Plasand, and J. Craninckx. "An 820 μW 9b 40 MS/s noise–tolerant dynamic-SAR ADC in 90 nm digital CMOS". *IEEE Int. Solid–State Circuits Conf. Dig. Tech. Papers*, pages pp. 238–239, Feb. 2008. (Cited on pages ix and 21.)

[10] P. Nuzzo, C. Nani, C. Armiento, A. Sangiovanni Vincentelli, J. Craninckx, and G. Van der Plas. "A 6–bit 50–MS/s threshold

configuring SAR ADC in 90–nm digital CMOS". *VLSI Circuits, 2009 Symposium*, pages pp. 238–239, Jun. 2009. (Cited on pages x and 25.)

[11] P. Nuzzo, C. Nani, C. Armiento, A. Sangiovanni Vincentelli, J. Craninckx, and G. Van der Plas. "A 6–bit 50–MS/s Threshold Configuring SAR ADC in 90–nm Digital CMOS". *Circuits and Systems I: Regular Papers IEEE Transaction*, vol. 59(no. 1):pp. 80–92, Jan. 2012. (Cited on pages x, 21, 25, 27, 30, 33, 36, 37, and 63.)

[12] A. Hastings. *"The art of analog layout"*. Prentice Hall, Upper Saddle Riverand, NJ, 2001. (Cited on pages xi and 69.)

[13] V. Navarro-Botello, J. Sosa, and Juan A. Montiel-Nelson. "Analysis and optimization of dinamically reconfigurable regenerative comparators for ultra–low power 6–bit TC–ADCs in 90nm CMOS technologies". *in 27th. Intl. Conf. on Design of Circuits and Integrated Systems (DCIS 2012)*, Nov. 2012. (Cited on pages xi and 27.)

[14] "SNDR". `http://en.wikipedia.org/wiki/Sndr`, Retrieved August 2012. (Cited on page 8.)

[15] U. Fiedler and D. Seitzer. "A high-speed 8 bit A/D converter based on a Gray-code multiple forlding circuit". *IEEE Journal of Solid-State Circuits*, vol. SC-14:pp. 547–551, June 1979. (Cited on page 12.)

[16] R.E.J. van der Grift, W.J.M. Rutten, and M. van der Veen. "An 8-bit video ADC incorporating folding and interpolation techniques". *IEEE Journal of Solid-State Circuits*, vol. SC-22:pp. 944–953, Dec. 1987. (Cited on page 12.)

[17] D. A. Johns and K. Martin. *"Analog Integrated Circuit Design"*. Wiley, New York, 1997. (Cited on page 15.)

[18] G. C. Temes. "High-accuracy pipeline ADC configuration". *Electron. Lett.*, vol. 21(no. 17):pp. 762–763, Aug. 1985. (Cited on page 15.)

[19] "Switched Capacitor". `http://en.wikipedia.org/wiki/Switched_capacitor`, Retrieved August 2012. (Cited on page 16.)

[20] J.L. McCreary and P.R. Gray. "All-MOS charge distribution analog-to-digital conversion techniques–Part I". *IEEE J.Solid-State Circuits*, vol. SSC-10(no. 6):pp. 371–379, Dec. 1975. (Cited on pages 16 and 20.)

[21] J.-S. Lee and L.-C. Park. "Capacitor array structure and switch control for energy-efficient SAR ADCs". *Proc. IEEE Int. Symp. Circuits Syst.*, pages pp. 236–239, 2008. (Cited on page 16.)

[22] K. Lee, J. Chae, M. Aniya, K. Hamashita, K. Takasuka, S. Takeuchi, and G.C. Temes. "A noise-coupled time-interleaved delta-sigma ADC with 4.2 MHz bandwidth, -98 dB THD, and 79 dB SNDR". *IEEE J.Solid-State Circuits*, vol. 43(no. 12):pp. 2601–2612, Dec. 2008. (Cited on page 18.)

[23] J. Markus, J. Silva, and G. C. Temes. "Theory and applications of incremental delta-sigma converters". *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 51(no. 4):pp. 678–690, April 2004. (Cited on page 18.)

[24] A. Agah, K. Vleugels, P.B. Griffin, M. Ronaghi, J.D. Plummer, and B.A. Wooley. "A high-resolution low-power oversampling ADC with extended-range for bio-sensor arrays". *Proc. IEEE Int. Symp. VLSI Circuits*, pages pp. 244–245, June 2007. (Cited on page 19.)

[25] J. De Maeyer, P. Rombouts, and L. Weyten. "A double-sampling extended-counting ADC". *IEEE J. Solid-State Circuits*, vol. 39(no. 3):pp. 411–418, March 2004. (Cited on page 19.)

[26] B.P. Ginsburg and A.P. Chandrakasan. "Highly interleaved 5-bit, 250-MSample/s, 1.2-mW ADC with redundant channels in 65-nm CMOS". *IEEE J. Solid-State Circuits*, vol. 43(no. 12):pp. 2641–2650, Dec. 2008. (Cited on pages 20 and 21.)

[27] M. van Elzakker, E. van Tuijl, P. Geraedts, D. Schinkel, E. Klumperink, and B. Nauta. "A 1.9 μW 4.4 fJ/conversion-step 10b 1MS/s charge-redistribution ADC". *IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, 3-7:pp. 244–610, Feb. 2008. (Cited on page 21.)

[28] B.P. Ginsburg and A.P. Chandrakasan. "500-MS/s 5-bit ADC in 65-nm CMOS with split capacitor array DAC". *IEEE J. Solid-State Circuits*, vol. 42(no. 4):pp. 739–747, Apr. 2007. (Cited on page 21.)

[29] Y.-K. Cho, Y.-D. Jeon, J.-W. Nam, and J.-K. Kwon. "A 9-bit 80 MS/s successive approximation register analog-to-digital converter with a capacitor reduction technique". *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 57(no. 7):pp. 502–506, Jul. 2010. (Cited on page 21.)

[30] F. Kuttner. "A 1.2 V 10b 20 MSample/s non-binary successive approximation ADC in 0.13 μm CMOS". *IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, vol. 2:pp. 136–137, Feb. 2002. (Cited on page 21.)

[31] M. Saberi, R. Lotfi, K. Mafinezhad, and W.A. Serdijn. "Analysis of power consumption and linearity in capacitive digital-to-analog converters used in successive approximation ADCs". *IEEE Trans. Circuits Syst. I, Reg. Papers*, 2011. doi: 10.1109/TCSI.2011.2107214. (Cited on page 21.)

[32] S. Mortezapour and E.K.F. Lee. "A 1-V, 8-bit successive approximation ADC in standard CMOS process". *IEEE Journal of Solid-State Circuits*, vol. 35:pp. 642–646, April 2000. (Cited on page 21.)

[33] G. Promitzer. "12-bit low-power fully differential switched capacitor noncalibrating successive approximation ADC with 1MS/s". *IEEE Solid-State Circuits*, vol. 36(no. 7):pp. 1138–1143, Jul. 2001. (Cited on page 27.)

[34] T. Kobayashi, K. Nogami, T. Shirotori, and Y. Fujimoto. "A current-controlled latch sense amplifier and a static power-saving input buffer for low-power architectures". *IEEE J. Solid-State Circuits*, vol. 28(no. 4):pp. 523–527, Apr. 1993. (Cited on page 28.)

[35] G. Van der Plas, S. Decoutere, and S. Donnay. "A 0.16 pJ/conversion-step 2.5 mW 1.25 GS/s 4b ADC in a 90 nm digital CMOS process". *IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, pages pp. 566–567, Feb. 2006. (Cited on pages 28 and 35.)

[36] P. Nuzzo, G. Van der Plas, F. De Bernardinis, L. Van der Perre, B. Gyselinckx, and P. Terreni. "A 10.6 mW/0.8 pJ power-scalable 1 GS/s 4b ADC in 0.18 μm CMOS with 5.8 GHz ERBW". *Proc. IEEE/ACM Design Automation Conf.*, pages pp. 873–878, Jul. 2006. (Cited on pages 28 and 35.)

[37] B. Wicth, T. Nirschl, and D. Schmitt-Landsiedel. "Yeld and Speed Optimization of a Latch-Type Voltage Sense Amplifier". *Solid-State Circuits, IEEE Journal on*, vol. 39(no. 7):pp. 1148–1158, Jul. 2004. (Cited on pages 31, 36, 37, 63, and 67.)

[38] J. He, S. Zhan, D. Chen, and R. Geiger. "Analyses of static and dynamic random offset voltages in dynamic comparators". *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 56(no. 5):pp. 911–919, May 2009. (Cited on pages 31 and 35.)

[39] A. Nikoozadeh and B. Murmann. "An Analysis of Latch Comparator Offset Due to Load Capacitor Mismatch". *Circuits and Systems II: Express Briefs, IEEE Transactions on*, vol. 53(no. 12):pp. 1398–1402, Dec. 2006. (Cited on pages 31, 35, and 63.)

[40] S. Nooshabadi and J.A.Montiel-Nelson. "Fast Feedthrough Logic: A high performance Logic Family for GaAs". *Circuits and Systems I: Reg. Papers, IEEE Transactions on*, vol. 51(no. 11):pp. 2189–2203, Nov. 2004. (Cited on page 31.)

[41] V. Navarro-Botello, J.A. Montiel-Nelson, and S. Nooshabadi. "Analysis of high performance fast feedthorugh logic families in CMOS". *Circuits and Systems II: Express Briefs, IEEE Transactions on*, vol. 54(no. 6):pp. 489–493, Jun. 2008. (Cited on page 31.)

[42] V. Navarro-Botello, J.A. Montiel-Nelson, and S. Nooshabadi. "Feedthrough: An Energy Efficient CMOS Logic Family for Arithmetic Circuits". *CMOS Techonolgy, NOVA Publishers, Electrical and Engineering Developments*, 2010. (Cited on pages 31 and 69.)

[43] P. Nuzzo, F. De Bernardinis, P. Terreni, and G. Van der Plas. "Noise analysis of regenerative comparators for reconfigurable ADC architectures". *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 55(no. 6):pp. 1441–1454, Jul. 2008. (Cited on pages 35, 37, and 69.)

[44] J. Kim, K. Jones, and M. Horowitz. "Fast, non-Monte-Carlo estimation of transient performance variation due to device mismatch". *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 57(no. 7):pp. 1746–1755, Jul. 2010. (Cited on page 36.)

[45] Ricardo M. Ramos. *"Contributions to the design of the energy recovery and stabilization system for long range passive wireless sensors"*. Tesis Doctoral. Las Palmas de Gran Canaria, 2009. (Cited on page 45.)

[46] *"UMC 90nm LOGIC/MIXED_MODE Standard Performance 1.0V MOSFET SPICE Model (BSIM4V4.3.0 Global Model) Document"*. Version 0.6 Phase 1. April 2009. (Cited on page 62.)

[47] R. J. Baker. *"CMOS Circuit Design, Layout and Simulation"*. Wiley-IEEE, 3th edition, 2010. (Cited on page 140.)

[48] "Nyquist–Shannon sampling theorem". `http://en.wikipedia.org/wiki/Nyquist-Shannon_sampling_theorem`, Retrieved May 2013. (Cited on page 141.)

[49] Ulf Grenander. *"Probability and Statistics: The Harald Cramer Volume"*. Wiley, 1959. (Cited on page 141.)

[50] Harry L. Stiltz. *"Aerospace Telemetry"*. Prentice-Hall, 1961. (Cited on page 141.)

[51] T. Zawistowski and P. Shah. "An Introduction to Sampling Theory". `http://www2.egr.uh.edu/~glover/applets/Sampling/Sampling.html`, Retrieved August 2012. (Cited on page 141.)

[52] "Nyquist frequency". `http://en.wikipedia.org/wiki/Nyquist_frequency`, Retrieved August 2012. (Cited on page 141.)

[53] Weste Neil H.E. and Harris David. *"CMOS VLSI Design: A Circuits and Systems Perspective"*. Addison Wesley, 3th edition, 2005. ISBN 0-321-14901-7. (Cited on page 142.)

[54] "Process corners". http://en.wikipedia.org/wiki/Process_corners, Retrieved September 2012. (Cited on page 142.)

[55] *"HSPICE Application Manual"*. Version X-2005.09. Synopsys, USA, 2005. (Cited on page 143.)

Friendship.
Is not something
you learn in school.
But if you haven't learned
the meaning of friendship,
you really haven't learned anything.

— Muhammad Ali

ACKNOWLEDGMENTS

I would like to thank a lot the Professor Maria Elena Valcher of the Department of Information Engineering at University of Padova for her excellent support.

Heartfelt thanks to the Supervisor Dr.D. Juan A. Montiel-Nelson of the Department of Telecommunications and Electronic Engineering at University of Las Palmas de Gran Canaria for his valuable support and contribution to this work.

When you remove the impossible,
whatever remains,
however improbable,
must be the truth.

— Sherlock Holmes

Padova, October 2013

Roberto Sartori

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography *"The Elements of Typographic Style"*. `classicthesis` is available for both LaTeX and LyX:

http://code.google.com/p/classicthesis/

Final Version as of October 2, 2013 (`classicthesis` version 1.2).