TECHNISCHE UNIVERSITÄT DRESDEN

**FAKULTÄT ELEKTROTECHNIK
UND INFORMATIONSTECHNIK**

**Institut für Feinwerktechnik
und Elektronik-Design**

MASTER THESIS

Topic: Design of an ultra-low-power current steering DAC in a modern SOI technology

Presented by: Shishira Subbarao Venkatesha

Born on: 24.07.1994 in: Chikmagalur, India

to
achieve the academic degree

MASTER OF SCIENCE

(M.Sc.)

Supervisor: Prof. Dr.-Ing. habil. Jens Lienig (IFTE, TU Dresden)
Dr.-Ing. Jeongwook Koh (Fraunhofer IIS/EAS)
First Reviewer: Prof. Dr.-Ing. habil. Jens Lienig (IFTE, TU Dresden)
Second Reviewer: Dr.-Ing. Alfred Kamusella (IFTE, TU Dresden)
Submission Date: 10.02.2020

Statement of Authorship

I hereby certify that I have authored this Master Thesis entitled *Design of an ultra-low-power current steering DAC in a modern SOI technology* independently and without undue assistance from third parties. No other than the resources and references indicated in this thesis have been used. I am aware that violations of this declaration may lead to subsequent withdrawal of the degree.

Dresden, 10.02.2020 Shishira Subbarao Venkatesha

Acknowledgments

I am indebted to few people who supported me for the successful completion of my master thesis. First and foremost I would like to thank my parents for giving me continuous support throughout my academics and sports. Without their love, encouragement and trust nothing would have been possible.

I convey my special thanks to Prof. Dr.-Ing.habil. Jens Lienig, director of IFTE, TU Dresden. He gave me an opportunity and had been guiding me during my entire thesis, his co-operation is highly commendable. Without his encouragement and guidance this research thesis would not have been successful. Further I would like thank Dr.-Ing. Alfred Kamusella in the reviewing committee for his precious time and contribution.

I would like to express my sincere gratitude towards my supervisor Dr.-Ing. Jeongwook Koh, for having shown utmost concern and support towards my thesis. Much of my learning and growing as a circuit designer in the last few months has been tremendously influenced by him. His tireless quest for perfecting ideas with critical thinking continuously motivated me during my thesis. I would like to thank my group manager Dr-Ing. Torsten Reich, for providing me an opportunity to do my research thesis at Fraunhofer IIS/EAS, Dresden. My special thanks to one of my dearest colleague M.Sc. Sunil Satish Rao, who was always ready to reach out with valuable ideas and discussions.

Dr. Koh, Mr. Satish Rao and I have brainstormed several number of times on various ideas regarding the thesis work. I sincerely appreciate their patience, confidence and trust they had in me. Further, my interactions with another colleague M.Sc. Marcel Jotschke were also helpful in the development of test benches and simulations. I thank him for his valuable insights and time.

Lastly, I would like to thank all my friends and well wishers throughout my journey.

Shishira Subbarao Venkatesha

Abstract

Despite the tremendous advancement in innovations on digitising and processing signals over the last century, real world signals are inevitably analog in nature. A digital-to-analog converter (DAC) serves in translating these digitised signals into different analog quantities like voltage, current or charges. We mainly focus on a Nyquist-rate current-steering digital-to-analog converter (CS-DAC) with resolution scalability from 8 to 12 *bit*, based upon the required output current for the application.

The proposed CS-DAC has a conversion rate of 10 Mega Samples per second *(MSps)* and adopts a segmented architecture for 8 *bit* to 12 *bit* resolution, where optimization is made for achieving a good performance with an area restriction. Especially, a new mode selection decoder is proposed and implemented for the resolution scalability of CS-DAC to 12 *bit*, 10 *bit* and 8 *bit*.

The CS-DAC is implemented in a 22 nm Fully Depleted Silicon on Insulator (FDSOI) process and only 0.8 V digital transistors were used for the design. For a typical case of 8 *bit* resolution, the measured integral non-linearity (INL) is between $\pm$ 0.05 LSB and the measured differential non-linearity (DNL) lies between -0.06 and 0.01 LSB providing 7.9 *bit* accuracy. The 10 *MSps* conversion rate has been obtained by transistor level designed binary-to-thermometer decoder and synchronisation circuit. A spurious-free dynamic range (SFDR) of 57 *dB* has been obtained for an input signal frequency in the interval from direct current (DC) to Nyquist-rate. The scalable CS-DAC is designed for ultra-low-power application. It has a total DC power consumption of 0.21 mW at 8 bit operation, 0.51 mW at 10 bit operation and 2.4 mW at 12 *bit* operation.

To the best of our knowledge, this is the first demonstration of a DAC capable of scaling its resolution.

Contents

List of Figures

List of Tables

Acronyms

IoT	Internet of Things
DAC	Digital-to-Analog Converter
ADC	Analog-to-Digital Converter
NMOS	Negative-channel Metal-Oxide Semiconductor
PMOS	Positive-channel Metal-Oxide Semiconductor
CMOS	Complementary Metal-Oxide Semiconductor
SOI	Silicon On Insulator
FDSOI	Fully Depleted Silicon On Insulator
I/O	Input and Output
CS-DAC	Current- Steering Digital-to-Analog Converter
BIT	Binary Digit
LSB	Least Significant Bit
MSB	Most Significant Bit
DR	Dynamic Range
FSR	Full Scale Range
Sps	Samples Per Second
KSps	Kilo Samples Per Second
MSps	Mega Samples Per Second
DNL	Differential Non-Linearity
INL	Integral Non-Linearity
CFT	Clock-Feedthrough
SNR	Signal-to-Noise Ratio
SNDR	Signal-to-Noise and Distortion Ratio
ENOB	Effective Number Of Bits
SFDR	Spurious-Free Dynamic Range

Op-Amp	Operational Amplifier
BOX	Buried Oxide
UTBB	Ultra Thin Body and Buried Oxide
RBB	Reverse Body Bias
FBB	Forward Body Bias
RVT	Regular Voltage Threshold
HVT	High Voltage Threshold
LVT	Low Voltage Threshold
SLVT	Super Low Voltage Threshold
p-p	Peak to Peak
2D	Two Dimensional
DFF	Delay Flip Flop
MS-DFF	Master-Slave Delay Flip Flop
CSA	Current Source Array
PVT	Process Voltage Temperature
MC	Monte-Carlo

1. Introduction

1.1. Digital to Analog Converter (DAC)

Over the past few years, the modern Internet of Things (IoT) have encountered drastic growth in the market and people require a bunch of latest hardware/software developments. IoT often includes software, hardware, or both that exchange data with the manufacturer, the operator and other connected devices. The data is an indispensable focus of IoT, hence it is important to ensure their accuracy and timing. As it's always said, the world we live in is analog and human need ways to interact with digital technologies [1]. Therefore, data converters are the essential blocks in signal processing circuitry providing digital-to-analog (D/A) and analog-to-digital (A/D) signal conversions. In electronic signal processing, a digital-to-analog converter (DAC, D/A, D2A, or D-to-A) is a system that converts a digital signal into an analog signal. Signal processing is a method of analysis, modification and interpretation of input signals like sound, images, time varying measurement values and many others. The processing of a signal helps to estimate characteristic parameters and also to transform the signal into the desired form. In modern electronics industry, DAC serves for various applications like audio amplifier, video encoder, motor control, sensor system, etc. In some applications, the function of DAC may not be straightforward but rather used for calibrations. For example few parameters like voltage offset, gain or current bias may require some timely adjustments or corrections. The ability to adjust these parameters dynamically is very important to ensure consistent results from the system [2].

A sensor-actuator system as illustrated in figure 1.1 used in IoT, is a relevant application. Sensors and actuators are credible factors for maintaining the precision of IoT data. A sensor collects information from the outside world which is sent to analog-to-digital converter (ADC) for analog to digital conversion. The digital data is then routed to signal processor where the decision is made and a correction signal is generated. A DAC receives this signal from the processor and produces a corresponding response to drive an actuator. Since, this response signal is often analog in nature, DAC becomes a suitable candidate for applications where calibration is necessary. As shown in figure 1.1, DAC serves as a conversion interface to reconstruct the analog signal from the digital output of a micro-controller or software [3]. The DACs are basically classified based on the reference quantity used, which will be elaborated in the next chapter.

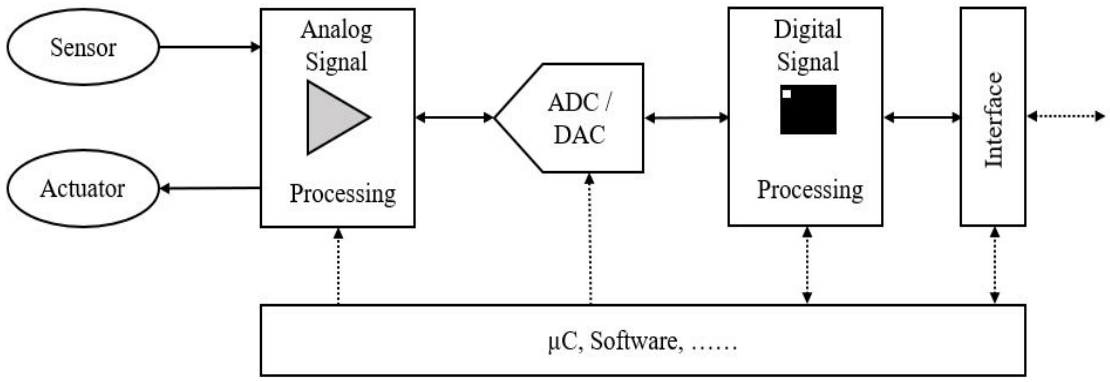

Figure 1.1.: Block diagram of a typical sensor-actuator system

1.2. Problem Statement

In selecting a DAC architecture, the first step is to determine the necessary resolution. The resolution of a DAC in a sensor-actuator system is selected based upon the maximum output signal's strength required to drive an actuator/load. In real application of a sensor-actuator system, there are many cases where the full scale current from a DAC needs to be varied to adjust sensor/actuator interface in the system. A low resolution DAC is necessary for low power consumption and a high resolution DAC for high power consumption. This poses the need of 'resolution scalability' in a DAC. In this work, a DAC is to be designed to operate for 8 *bit*, 10 *bit* and 12 *bit* resolutions.

The evolution towards the modern technology like Fully Depleted Silicon on Insulator (FDSOI) with smaller nodes makes it possible to further integrate several millions of transistors on a single chip. On the other hand, this brings several considerations with respect to circuit design. The chosen 22nm FDSOI technology offers two main transistor variants under its base platform, one is core devices with a supply voltage of 0.8 V and the other is I/O (input/output) devices with a supply voltage of 1.8 V. A suitable transistor type among them is to be selected for the design that is feasible for low power applications. The design considerations like high speed and high linearity along with low power requirements makes the design of DAC challenging. In this work, an ultra-low-power scalable DAC is to be implemented in a 22 nm FDSOI technology. During the design, a trade-off between power consumption and performance should be considered to meet the requirements.

1.3. Possible Solution

It has been keenly noted that the idea of resolution scalability yields a 'Smart and flexible' DAC for low power sensor applications. A detailed analysis and investigation on DAC

architectures has to be made to select a suitable topology for ultra-low-power application. The current steering architecture (subsection 2.2.3), is at the moment regarded as best suitable DAC architecture for high resolution and low power applications. The main highlight of this work is resolution scalability which has to be addressed from the initial design steps. The idea of resolution scalability is new and unique which makes the design even more challenging. A possible approach would be designing a CS-DAC to operate on three different modes (8 *bit*, 10 *bit* and 12 *bit*), by assigning each mode to each resolution. A new logic is to be developed that performs the necessary mode selection operation to achieve scalability. The 12 *bit* CS-DAC forms the top level design with maximum power dissipation, from which the resolution can be scaled down to 10 *bit* and 8 *bit*.

In 22nm FDSOI technology core devices of 0.8 V supply voltage are feasible for low power applications and they are further classified based on their threshold voltages (section 4.3). In a CS-DAC, the current source is the main design block responsible for D/A conversion. In designing a current source, the current mirror topology that fits for low voltage and low power requirements should be analysed and implemented. The transistor mismatches among these current mirrors limits the accuracy and influences the performance of a converter. These mismatches can be minimised by designing the current mirrors based on suitable statistical mismatch model and then verified by Monte-Carlo simulations.

1.4. Scope

The portrayed work will focus on the design steps taken for the implementation of an ultra-low-power DAC in a modern FDSOI technology, with an option for resolution scalability.

Chapter 2 discuses the performance metrics of a DAC, state of the art and remarks on selected architecture.

Chapter 3 describes the short summary on the idea taken to achieve the goal.

Chapter 4 describes the general overview of 22 nm FDSOI technology that is used in this work.

Chapter 5 gives an overview of basic requirements and design of blocks involved in CS-DAC are discussed with simulation.

Chapter 6 an approach to resolution scalability is explained and tested. Further, the performance metrics are measured with appropriate simulations.

Chapter 7 conclusion about the thesis and ideas for possible future work.

2. State of the Art

In this chapter the functionality of a DAC will be shortly discussed with the specifications that describe its static and dynamic performances, further we discuss about the available different DAC architectures. The remainder of this chapter discusses about the current steering architecture and their three possible implementations with advantages and disadvantages.

The motive of using a DAC is to transform the digital word to its corresponding analog representation with respect to a reference signal. In other words, it produces a multiple of certain reference quantity for each digital input code. As shown in figure 2.1 , a DAC has n bit digital input, a reference signal, clock input and analog output (single ended or differential) . The analog output is the weighted summation of the product of reference signal and digital input codes.

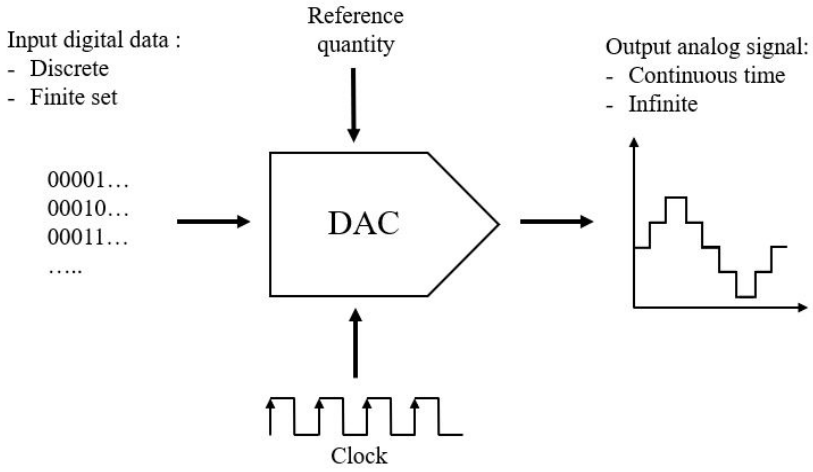

Figure 2.1.: The D/A converter

2.1. Performance Metrics

In order to characterise and analyse the performance of DACs, number for performance metrics have been introduced over the past years. These metrics are divided into three main measures to highlight their importance on the intended application. For example, DACs used

for instrumentation purposes should have high static performance, whereas in telecommunication application it should have excellent dynamic performance [4]. Some important metrics describing general, static and dynamic behaviour of a DAC will be discussed in this section.

2.1.1. General Specification

The general specification of a converter contains the basic properties like resolution, full scale range and conversion rate. These metrics serve as the initial requirements of a converter.

Resolution The resolution of a DAC is expressed as the number of bits in its digital input code, it is expressed in *bits*. Resolution is the number of bits used to generate the analog output levels.
For a DAC of resolution *N bit*, it will generate 2^N of analog output levels.

Least Significant Bit The height of one step is defined as 1 least significant bit (LSB), it is often used as the reference unit for certain quantities in the specification.

Full Scale Range It is the maximum value of output signal, which is 1 LSB below the reference. It is always smaller than the number of output levels (2^N) because of the finite resolution of DACs.
The relation between LSB and FSR can be written as :

$$1LSB = \frac{FSR}{(2^N - 1)} \tag{2.1}$$

Dynamic Range It is defined as the ratio of the output signal's maximum level to its minimum level. The dynamic range (DR) is associated to the resolution of the DAC and is expressed in *dB*.

$$DR = 20log\frac{(2^N - 1)}{1} \tag{2.2}$$

Conversion Rate It is the speed at which a DAC can operate and produce repetitious conversions at the output. They are be expressed in samples per second *Sps*.

2.1.2. Static Performance

Static performance can be discussed by regarding the DAC as a discrete-time circuit, where the analog output values are only viewed at discrete time intervals known as static (settled)

values. This static behavior is also referred as direct current (DC) behaviour which give rises to some errors in the system. These errors are signal-independent [5], they affect the accuracy of the converter by imposing non-linearity that limits the overall performance .

Offset and Gain Error An offset error is defined as the shift in the converter's transfer characteristic by a constant DC offset. It can be measured by setting the digital input code to all zeros (0).
A gain error is defined as the deviation in the slope of the actual characteristics with its ideal one. After removing the offset, gain error can be measured by applying an input code of all ones (1). Figure 2.2a visualises offset and gain errors.

Differential Non-Linearity Error (DNL) It is the worst case deviation between the actual and ideal step height (1 LSB) between corresponding two adjacent codes. They are expressed in terms of LSBs.

$$DNL_k^{norm} = \frac{A_{k+1}^{actual} - A_k^{actual} - 1LSB}{1LSB} \tag{2.3}$$

where, DNL_k^{norm} is the DNL normalised to 1 LSB, A_{k+1}^{actual} and A_k^{actual} are adjacent analog outputs.

Integral Non-Linearity Error (INL) The integral non-linearity error is defined as the maximum deviation of the actual transfer characteristics from the ideal straight line and is expressed in LSBs. It is also addressed as the running sum of DNL, i.e. INL at any point in transfer characteristics is the accumulation of all previous DNL errors.

$$INL_k^{norm} = \frac{A_k^{actual} - A_k^{ideal}}{1LSB} = \sum_{j=1}^{k} DNL_j^{norm} \tag{2.4}$$

where, INL_k^{norm} is the INL normalised to 1 LSB, A_k^{actual} and A_k^{ideal} are the actual and ideal analog outputs respectively.
Figure 2.2b illustrates DNL and INL errors using the transfer characteristics of a DAC.

Monotonicity A DAC is said to be monotonic when its output increases or remains unchanged as the digital input code is increased. In other words, the output should not decrease with an increasing input code.
If the DNL is less than or equal to 1 LSB, the characteristic of DAC is always monotonic [6].

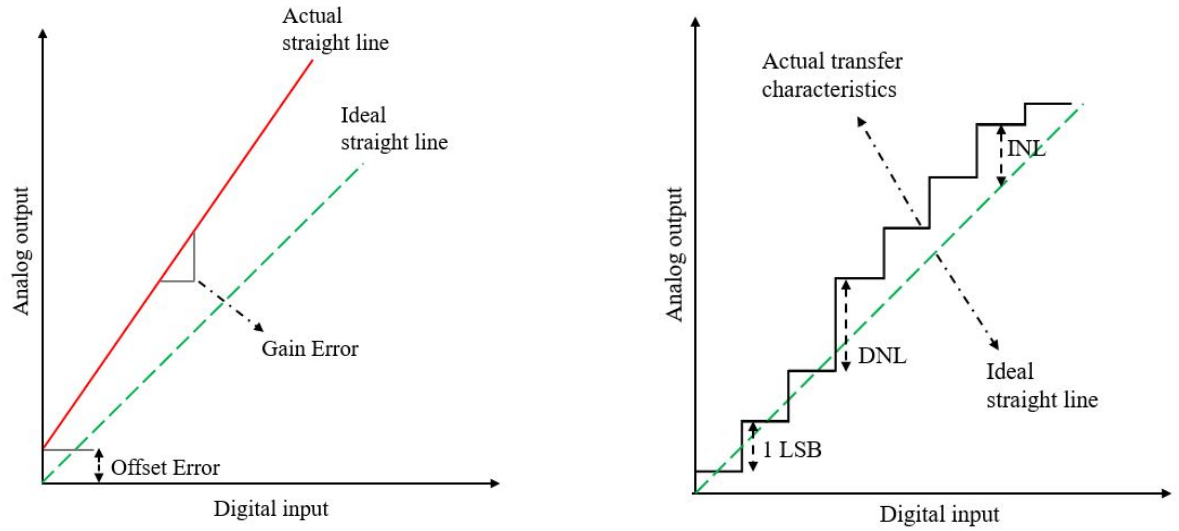

Figure 2.2.: Static performances of a DAC (a) Offset and gain errors (b) DNL and INL errors

2.1.3. Dynamic Performance

In this section, the DAC is analysed by considering it as continuous-time circuit, using measures in both time and frequency domain. The influence of input signal frequency and switching of analog elements are reflected in amplitude and transient behaviour of the analog output waveform [7]. These transients are signal dependent and hence dynamic in nature [5]. The dynamic metrics such as settling time, glitch and clock-feedthrough are based on the time-domain response of the DAC. On the other hand, metrics like signal-to-noise ratio (SNR), effective number of bits (ENOB) and spurious-free dynamic range (SFDR) are based on the frequency-domain analysis.

Glitch A glitch is a short uncontrolled variation in the output signal that is generated due to the unmatched switching time of different bits. As a result, a false code could appear at the output for a short period of time (figure 2.3a).

Clock-Feedthrough The issue of clock-feedthrough (CFT) is caused by the parasitic capacitive coupling in complementary metal-oxide semiconductor (CMOS) switches. Due to this coupling, the clock or digital switching signals will affect the analog output signal as shown in figure 2.3a. The CFT can be seen both at rising and falling edge of the clock signal.
For a CS-DAC (section 2.2.3) the CFT error can be viewed in a similar way as glitches [5].

Settling time In general, settling time of a DAC is defined as the time taken by the analog output signal to settle within a certain accuracy of the final value. During a full scale transition (figure 2.3a), the settling time should be kept as small as possible to reduce the distortion on the analog output signal [7].

Signal-to-noise ratio (SNR) The signal-to-noise ratio (SNR) is defined as the ratio of the fundamental signal power to the total noise power at output within a certain frequency band.

$$SNR_{dB} = 10log_{10}(\frac{P_s}{P_n}) \approx 6.02N + 1.76 \tag{2.5}$$

where P_s is the signal power, P_n is the noise power and N is the resolution of the converter. The SNR is increased by approximately 6 dB for each additional bit in the converter.

Signal-to-noise and distortion ratio (SNDR) The signal-to-noise and distortion ratio (SNDR) is defined as the ratio of the fundamental signal power to the sum of total noise power and distortion power within a certain frequency band.

$$SNDR_{dB} = 10log_{10}(\frac{P_s}{P_n + P_d}) \tag{2.6}$$

where P_s is the signal power, P_n is the noise power and P_d is the distortion power due to harmonics.

Effective number of bits (ENOB) The converters in real world will have a lot of impairments like noise, harmonics and non-linearity that limit their actual resolution. The resultant effective resolution corresponding to measured SNDR is known as effective number of bits.

$$ENOB = \frac{SNDR_{dB} - 1.76}{6.02} \tag{2.7}$$

Spurious-free dynamic range (SFDR) The spurious-free dynamic range (SFDR) is the ratio of the fundamental signal power to the power of the largest distortion or spurious component within a certain frequency band (figure 2.3b). The equation for SFDR in a CS-DAC (section 2.2.3) is derived in [4] :

$$SFDR_{dB} = 20log_{10}(\frac{Z_{imp}}{Z_L}) - 6.02(N-2) \tag{2.8}$$

where Z_L is the load impedance and Z_{imp} is parallel impedance.

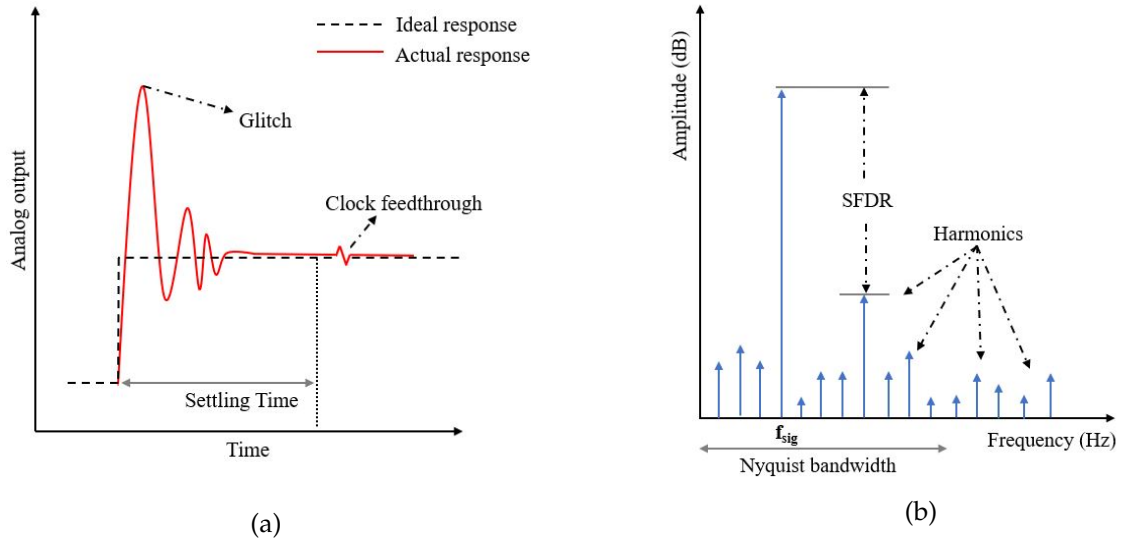

Figure 2.3.: Dynamic performances of a DAC (a) Time domain response (b) Frequency domain measure

2.2. DAC Architectures

The D/A converters are classified into serial and parallel converters based on their conversion speed. A serial DAC will perform only one conversion per clock cycle, hence they need N clock cycles for a resolution of N bit and are slower [6]. In contrast, parallel DAC is faster as it can convert all bits in the same clock cycle. As shown in figure 2.4, they are further classified based on the mode of conversion, i.e. voltage scaling, charge scaling and current scaling DACs. Three architectures, one for each conversion mode is shortly discussed in this section. The current steering architecture which is of interest, is further discussed more in detail with its three possible implementations.

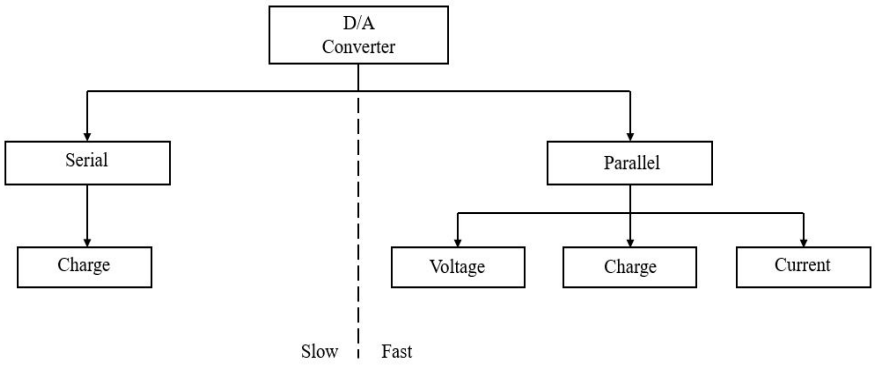

Figure 2.4.: The classification of DAC architectures

2.2.1. Voltage Scaling DACs

A voltage scaling DACs is the basic architecture, which comprises of a serial array of resistors scaling the voltage from voltage reference, V_{REF}, to ground. The elementary component is a resistor and the mode of conversion is voltage. R-2R resistor ladder is an classic example, that divides a V_{REF} into a number of different voltage levels.

R-2R ladder DAC is the oldest and still the 'cleanest' conversion method, it is an improvisation of binary weighted resistor DAC [3]. As shown in figure 2.5, it is designed using resistors of only two values R and 2R. Digital bits control the switches to connect one end of 2R resistor to one input of operational amplifier (op-amp) or the ground. The output voltage, V_{OUT}, of the ladder is a proper weighted sum of digital input codes.

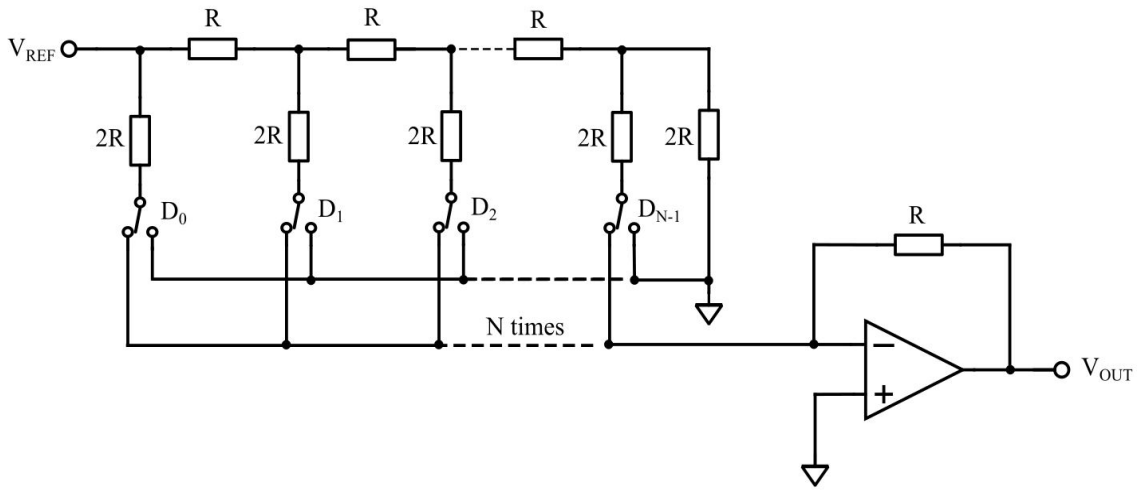

Figure 2.5.: The R-2R ladder DAC architecture

2.2.2. Charge Scaling DACs

The charge scaling DACs are also known as charge-redistribution architecture, they deal with sharing the charges stored in capacitor array [8]. In this case, the elementary component is a capacitor and the mode of conversion is charge. Figure 2.6 shows a charge scaling DAC of *N bit*, it contains a parallel array of binary weighted capacitors connected to an op-amp. If *C* is an unit capacitance, then the total number of capacitors in the array is $2^N C$.

Generally the circuit is operated under two non-overlapping clock phases, the reset phase and the charge sharing phase. The digital input codes controls the charging and discharging of capacitors through switches, causing V_{OUT} to be a function of the voltage division between the capacitors. They are one of the popular architecture in CMOS technology.

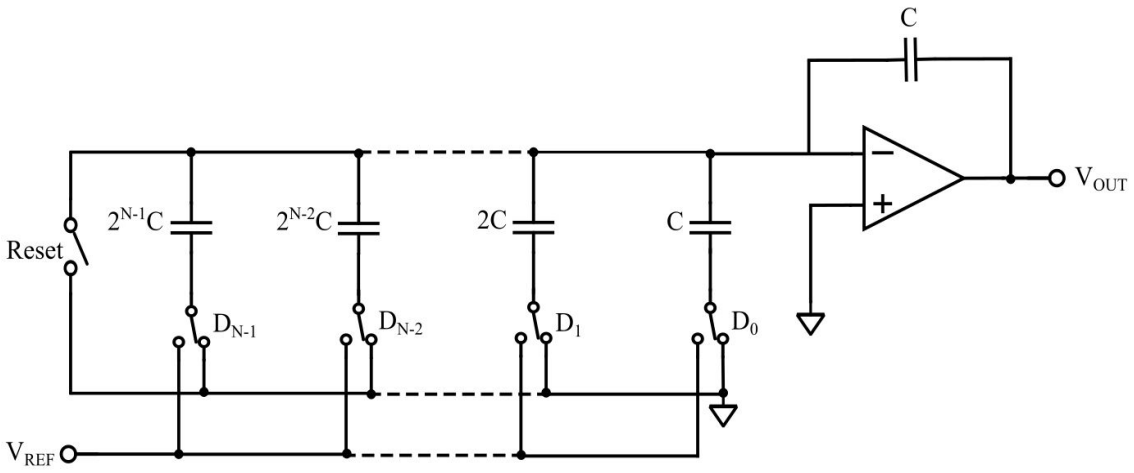

Figure 2.6.: The charge scaling DAC architecture

2.2.3. Current Scaling DACs

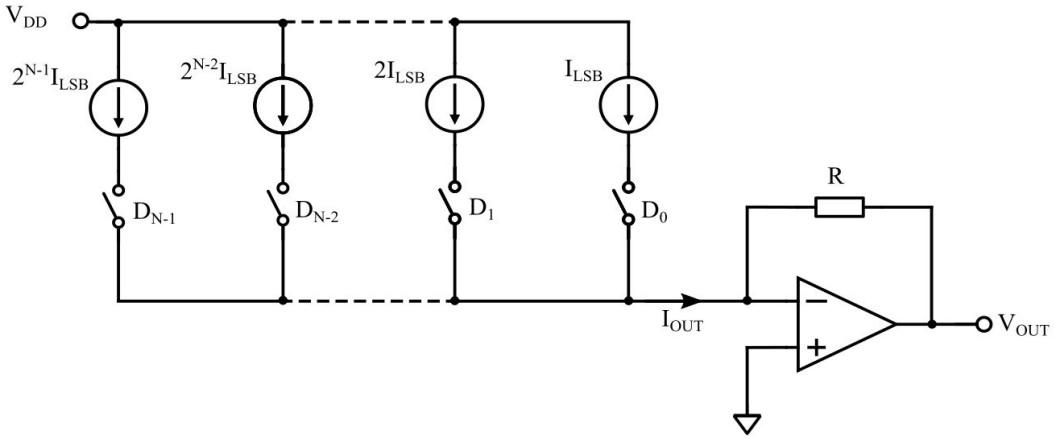

Figure 2.7.: Th current-steering DAC architecture

The next class of DACs are current scaling architecture, popularly known as current-steering DACs. It uses current source as an elementary component and the mode of conversion is current. Figure 2.7 depicts a CS-DAC that converts the output current into a voltage using an op-amp. In many applications just a load resistor used instead of op-amp. The main advantage is the use of transistors to implement current sources, which makes this architecture consume much smaller silicon area and offers higher speed than the voltage or charge scaling DACs [3].

The current sources are biased by a current mirror and matching behaviour of these elements determines the resolution of the CS-DAC. The current-steering architecture is further classified based on the coding schemes, known as binary-weighted and unary-weighted

implementations.

Binary Implementation In this implementation, a binary-weighted current source array is used which are directly controlled by binary inputs. As shown in figure 2.8, each current source in the array steers a current value twice as large as its previous element. The digital input codes directly switches a current to the output using a differential switch pair for each current source.

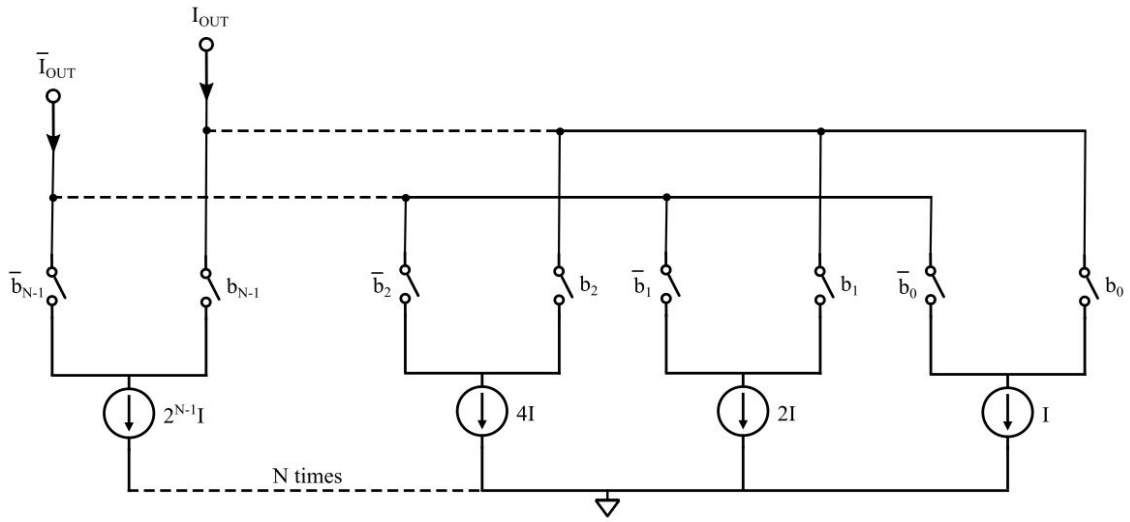

Figure 2.8.: Binary-weighted current steering DAC implementation

A *N bit* binary-weighted CS-DAC employs N number of current sources and its output is given by :

$$I_{OUT} = I_{LSB} \sum_{i=0}^{N-1} b_i 2^i \tag{2.9}$$

where I_{LSB} is the current value steered from least significant bit and b is the corresponding binary bit. Binary implementation is the simplest in current steering architecture, the advantages are small area and less power consumption. The drawbacks are, it suffers from large DNL errors, poor dynamic performance due to large glitches and the converter can exhibit non-monotonous behaviour [7].

Unary Implementation It is also termed as thermometer-coded implementation, which uses unary-weighted or identical current source array and each current source is addressed separately (figure 2.9). The digital binary input is applied to a binary-to-thermometer decoder and its output is fed to the differential switch pair.

A *N bit* unary-weighted CS-DAC employs $M = (2^N - 1)$ number of current sources

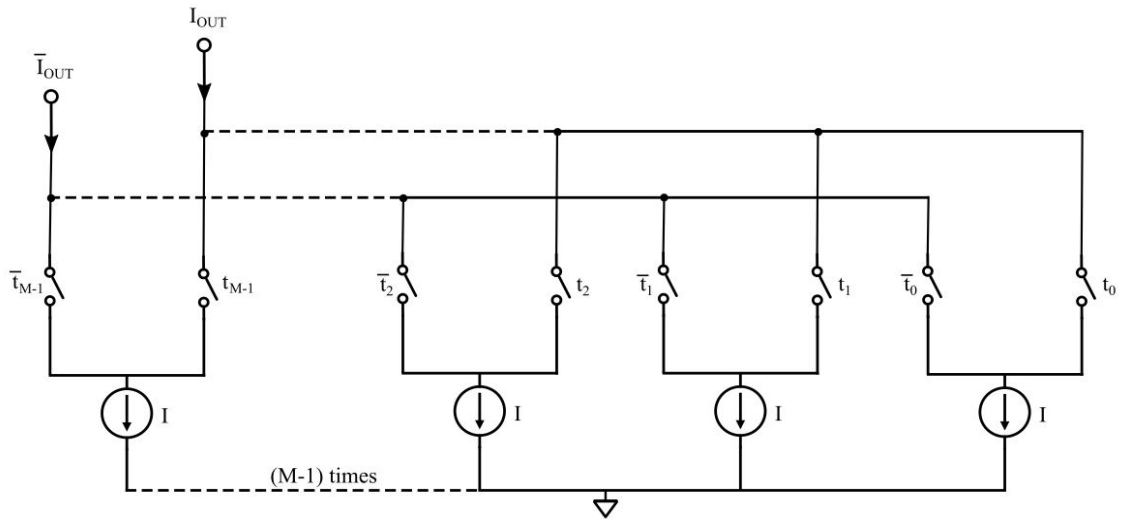

Figure 2.9.: Unary-weighted current steering DAC implementation

and its output current is given by :

$$I_{OUT} = I \sum_{i=0}^{M\text{-}1} t_i \tag{2.10}$$

where I is the identical current value and t is the corresponding thermometer coded bit. The unary implementation is a suitable alternative for binary implementation because, it guarantees monotonicity, good DNL and very low glitches [7]. The disadvantages are increased complexity (decoding logic), large area and higher power consumption.

Segmented Architecture As discussed in the previous paragraphs, binary and unary implementations have their own advantages and disadvantages. To get the best out of these two implementations, both are employed in a single architecture known as segmented current steering DAC. The basic idea of this architecture is to deliver a hybrid implementation of binary and unary implementations, where the DAC is divided into two sub-DACs based on the segmentation of digital inputs. By optimal segmentation, this architecture provides a balance between static and dynamic performances for a reasonable complexity and area [4].

Figure 2.10 shows a *N bit* binary input data, which is segmented into *B* least-significant bits switching the binary-weighted current sources and (*N-B*) most-significant bits that are thermometer-decoded to switch $2^{(N\text{-}B)} - 1$ unary current sources. The output current is then given by :

$$I_{OUT} = I \sum_{i=0}^{B-1} b_i 2^i + 2^B I \sum_{i=0}^{M-1} t_i \tag{2.11}$$

where $M = 2^{(N-B)} - 1$ is the number of unary current sources, $I = I_{LSB}$ is the LSB current value, b and t are the corresponding binary and thermometer bits respectively.

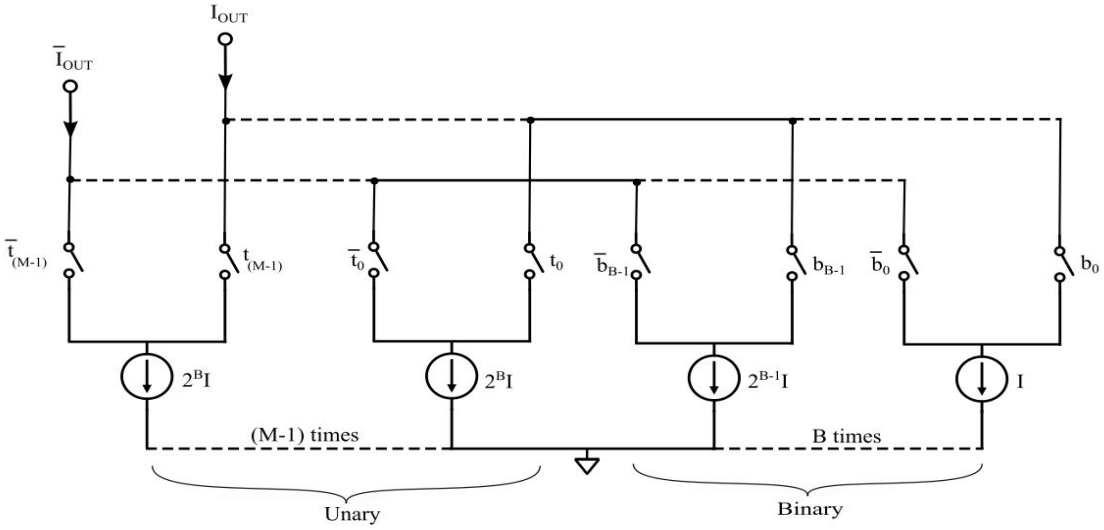

Figure 2.10.: Segmented current steering DAC implementation

2.2.4. Comparison

In this chapter, different architectures of DAC were shortly discussed that are presently available in the literature. The performance metrics (section 2.1) are explained to understand the basic functionality of each architecture. The advantages and disadvantages of these architectures are highlighted in table 2.1 for a clear comparison. The Current-steering architecture is selected for this work as they provide best results regarding speed, area, accuracy and mainly low power consumption which is of our utmost interest. The next step would be the selection of suitable implementation among the current-steering architectures (section 2.2.3). Figure 2.11 gives a brief insight on the comparison among three possible implementations of current steering architectures, with an example of 4 *bit* DAC in each case. The segmented current-steering architecture is further opted for this work, which brings a good trade off with respect to area, performance and power combining two implementations.

2.3. Remarks on Segmentation

Several works are done in the past on modelling of Current steering architecture emphasising on their static and dynamic behaviours. One of the important factor to be analysed is

Table 2.1.: Advantages and disadvantages DAC architectures

DAC Architectures	Advantages	Disadvantages
1.Voltage-Scaling DAC	• Easy implementation • Faster response time • Inherently monotonic & un-buffered	• Matching between wide range of resistors • Limited update rate • Large silicon area
2.Charge-Scaling DAC	• Good linearity • Occupies less area • Low power applications in CMOS	• Sensitive to parasitic • Matching of capacitors • Complex layout
3.Current-Scaling DAC	• High speed • High resolution • High accuracy & low power • Less silicon area; suitable for modern SOI tech.	• Challenging to design • Prone to mismatch errors • Limited output impdence

the degree of segmentation level. In the previous section, the advantages of segmented implementation was discussed and now it's important to analyse the optimal segmentation level to be chosen for the design. An area approach explained in [10], provides a starting point in selecting the segmentation level based on the silicon area consumption of the CS-DAC. The figure 2.12 shows the different area constraints depicted for a 10 *bit* implementation. A fully binary-weighted design is referred as 0 % segmented and a fully unary-coded design refers to 100 % segmented [10]. As discussed in subsection 2.2.3, fully unary-coded design consumes more area due to $2^N - 1$ identical current sources and binary-thermometer decoder. Whereas, a truly binary-weighted design consumes considerably small area. The graph of normalised chip area versus percentage of segmentation aids in the investigation of optimal segmentation level [10]. From figure 2.12 it can be observed that,

- The DNL performance is dependent on the segmentation level. Based on DNL performance only, if A_{unit} is the minimum analog area for 100 % segmentation, then the minimum analog area for 0 % segmentation is $2^N * A_{unit}$. On a logarithmic scale, analog area as a function of segmentation forms a straight line connecting these two points. (*line 1*)

- The digital area which indicates the area of decoding logic increases with the increase

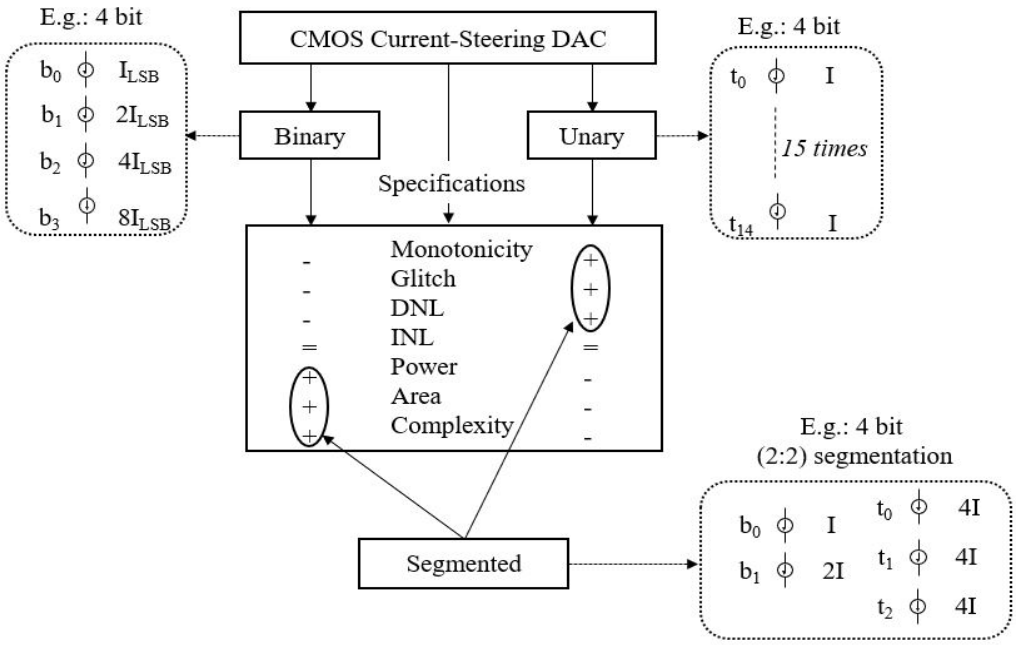

Figure 2.11.: The comparison of three current steering implementations [9]

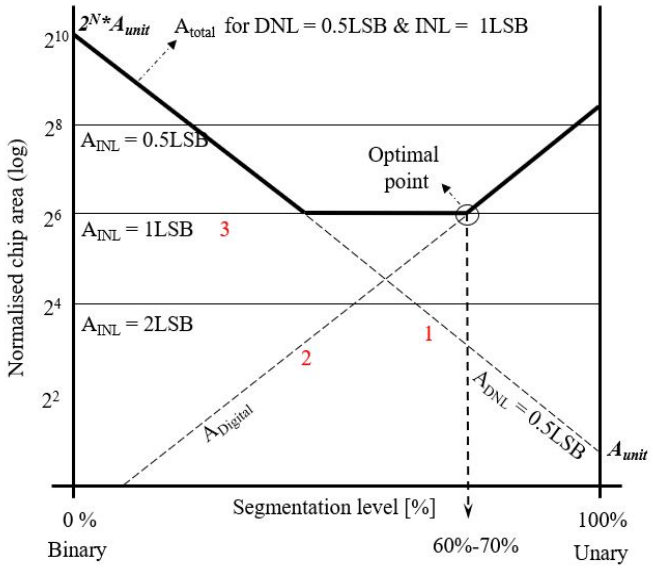

Figure 2.12.: Normalised required area versus segmentation level [10]

in segmentation level. (*line 2*)

- The INL behaviour in a CS-DAC is independent of segmentation level, but is dependent

only on the total analog area [4]. (*line 3*)

- As the segmentation level increases, the required total analog area is first dominated by the DNL specification, then by the INL specification and finally by the decoder logic.

In a CS-DAC, often the INL specification is more relaxed compared to DNL. As a conclusion, for minimal chip area the flat part in the figure 2.12 are the possible segmentation levels. The highest possible segmentation level in the flat part will have lower glitch energy and yields higher dynamic performance of the converter [4]. Hence, 60 % to 70 % is the optimal segmentation level to be selected for designing a segmented current steering DAC.

Some of the recent works on CS-DAC for low power applications are tabulated in table 2.2, that could be closely appropriate for comparison. Three works for each case 8 *bit*, 10 *bit* and 12 *bit* resolution are tabulated along with their chosen segmentation level and few available specifications. The conclusion that can be drawn is: the increase of binary level seems to be more advantageous in terms of low power, but the dynamic performance is mainly dependent on unary level. Therefore, the segmentation level should be wisely chosen as a good trade of between available chip area and required performance of the CS-DAC. The table 2.2 substantiates that the total power consumption of the CS-DAC decreases with the scaling down of resolution.

Table 2.2.: Recent works on current-steering DACs

Cite	Resolution (*bit*)	Seg. (U:B)	Rate (*MS/s*)	I_{LSB} (μA)	DNL (*LSB*)	INL (*LSB*)	Supply (*V*)	Power (*mW*)	Tech. (*nm*)
[11]	12	7:5	320	4.88	0.3	0.4	3.3	82	180
[3]	12	8:4	25	4.88	-0.003	-0.69	3.3	$\leq$ 32.8	180
[12]	12	8:4	200	2.93	0.2	-0.85	1.2	22	90
[13]	10	0:10	250	9.75	0.1	0.1	3.3	22	180
[14]	10	0:10	30	2.5	0.2	0.2	3.3	7.8	350
[15]	10	2:8	2	0.1	0.42	0.42	3.3	0.6	350
[16]	8	6:2	50	7.8	0.04	0.025	3.3	$\leq$ 0.3	350
[17]	8	4:4	2.6	0.04	0.4	0.5	1.8	0.1	180
[18]	8	4:4	80	4	0.14	0.35	1	$\leq$ 0.1	40

3. Research Objectives

Aim of this work is to implement an unique resolution scalable DAC for a sensor-actuator system. Since the choice of DAC architecture and its resolution for the system depends on how it can interface with the actuator, the application of DAC in sensor-actuator system was previously discussed. A segmented current steering DAC architecture is selected for the resolutions 8 *bit*, 10 *bit* and 12 *bit*.

The primary research is on the implementation of resolution scalability on CS-DAC with internal mode control operation. As shown in table 3.1, two digital control signals S_1 and S_0 will be used to select the desired mode of operation. Since we have three resolutions to be considered for optimal segmentation, the segmentation levels are selected based on the remarks from section 2.3. The idea is to keep the binary segmentation level same for all three cases and scale the unary level with respect to resolution. This is because unary implementation uses identical current sources and it would be much easier to group and scale them. Therefore, the binary segmentation of 4 *LSBs* is kept constant and the unary level is scaled as shown in table 3.1. The 12 *bit* CS-DAC of 8 : 4 segmentation forms the top level design, from which the resolution can be scaled down to 10 *bit* and 8 *bit* controlled by S_1 and S_0. As the resolution scales down, the number of active current cells required is reduced. The corresponding non-active current cells should be powered off, because keeping them floating or power on would not allow us to achieve scalability. Hence during scaling down, the non-active current cells have to be grouped with respect to selected resolution and powered off by additional logic control.

Table 3.1.: Resolution mode control

Modes S_1 S_0	Resolution (*bit*)	Segmentation (U : B)
0 0	-	-
0 1	8	4 : 4
1 0	10	6 : 4
1 1	12	8 : 4

The next aspect is the design of an ultra-low-power current steering DAC. To have a low power application from the chosen core devices in 22 nm FDSOI technology, a brief study on core devices is done in chapter 4. Scaling down of technology node is not beneficial to analog circuits, as transistor mismatch becomes more prominent by the reduction of voltage headroom and transistor sizes. Current source is the essential analog block in a CS-DAC, its accuracy directly reflects on static and dynamic performances of the converter. A low-voltage cascode current mirror topology is selected for biasing, which provides good accuracy and high output swing [8]. To further improve the accuracy, the current source transistor is dimensioned using Pelgrom's mismatch equations [19]. The dimensions of the current source transistor is dependent on the full scale current of the CS-DAC and the technology in which the design will be implemented.

First, the basic requirements and the blocks to be designed for a low power CS-DAC are analysed. Then, the approach and steps necessary to achieve resolution scalability is elaborated from top level.

4. FDSOI 22 nm Technology

This chapter contains a brief study on 22nm FDSOI technology from Global Foundries. The core devices in 22 nm technology with a supply voltage of 0.8 V is considered for this work. Hence, it is important to understand the transistor physics behind it before designing.

4.1. Overview

The surge of scaling down the CMOS technology nodes with high-performance and low-cost have made tremendous growth in the semiconductor industry. The main aim of CMOS scaling is to provide smaller and faster transistors from one node to the next lower node, best described by Moore's Law [20]. The continuous shrinking of bulk CMOS technology brings a limitation to its ideal functionality due to short channel effects and manufacturing complexities [21]. For advanced technology nodes like 28 nm and beyond, it has a serious effect on power consumption due to the leakage current [22]. On contrary, Fully Depleted Silicon on Insulator (FDSOI) technology allows further scaling of CMOS by offering good analog performance and reusing 90% of process steps used in 28 nm bulk technology. Figure 4.1 shows the cross section of a typical bulk CMOS technology.

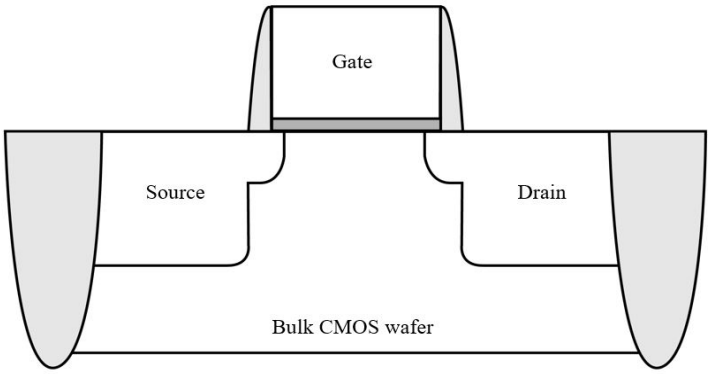

Figure 4.1.: Cross section of bulk CMOS transistor

FDSOI is also a planar technology and it does not completely change the fundamental geometry of the MOS transistor. The innovation in FDSOI lies in an addition of a thin layer

of insulator just below the channel as shown in figure 4.2 , called the Buried Oxide (BOX). This eliminates the need of injecting dopants into the channel layer and thus making it fully depleted. The insulator layer effectively confines the electron flow from source to the drain and drastically reduces the leakage current from the channel to the substrate [23].

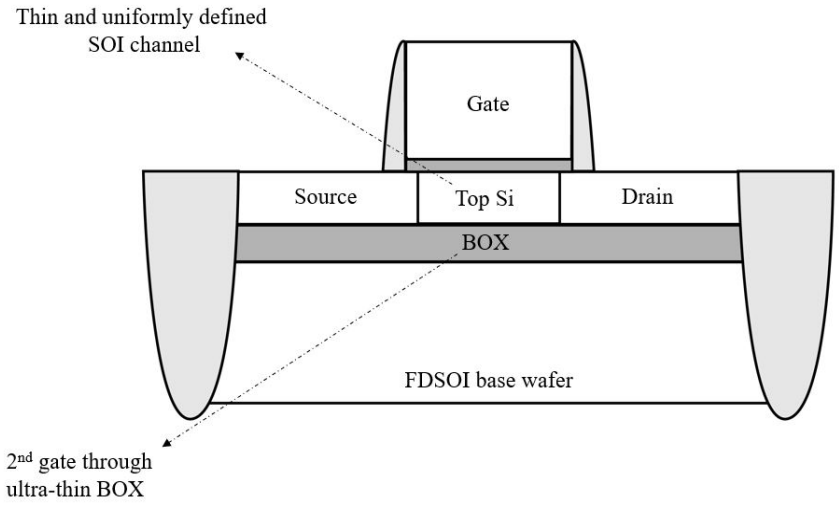

Figure 4.2.: Cross section of FDSOI transistor

In FDSOI, dopants usage are greatly minimised thereby limiting the variability within the process. As a result the characteristics of the each transistor in a chip are closer to average. A very thin film of silicon implements the transistor channel unlike the conventional bulk technology, this layer together with BOX is known as Ultra Thin Body and Buried Oxide (UTBB) [23]. The addition of BOX layer provides two major advantages, firstly the transistor encounters negligible capacitance between source-drain region and substrate, this leads to an improvement in the switching speed of a transistor [22]. Next is the greater ability for body biasing, the biasing creates a buried gate below the channel which makes the FDSOI act like a vertical double gate transistor.

4.2. Back Gate Biasing

In order to improve the performance of the transistor a voltage can be applied to it's substrate, this method is called back gate biasing. It facilitates the early formation of an inversion layer between the source and the drain. The threshold voltage of a transistor is the minimum gate to source voltage required to form an inversion layer that allows a current to flow between drain and source. Equation 4.1 known as Shichman-Hodges equation represents the threshold

voltage V_{th} for a n-channel device :

$$V_{thn} = V_{th0} + \gamma(\sqrt{|V_{SB} + 2\phi_F|} - \sqrt{|2\phi_F|}) \tag{4.1}$$

where, V_{SB} is the potential difference between source and body, V_{th0} is the threshold voltage when source is connected to bulk ($V_{SB} = 0$), γ is the body-effect coefficient and ϕ_F is the Fermi potential [22]. The idea of back gate bias, also known as body bias is to apply a voltage to the substrate, as the potential changes the back-bias term $\sqrt{|V_{SB} + 2\phi_F|}$ in equation 4.1 changes and hence the threshold voltage. The body terminal is exploited as a 'back gate', lower the value of back-bias term lower is the threshold voltage.

In bulk technology, the ability of back gate biasing is very limited due parasitic current leakage. In FDSOI due to the thin insulator layer, back gate biasing creates a buried gate below the channel. The buried gate further prevents any leakage of current into the substrate, this allows a much higher voltage on the body leading to significant boost in the performance. It gives an opportunity to control the behaviour of transistor not only through top gate but also through the substrate (back gate). Hence, body bias provides great design flexibility for circuit designers with active trade-off for performance vs power [24].

4.3. FDSOI Architectures

The core devices in 22nm FDSOI technology offers a wide range of transistors based on MOS architecture and their effective body-biasing. There are two MOS architectures, one is regular well device and the other is flipped well device. In figure 4.3 PMOS is over an n-well and NMOS over p-well, this is a conventional style structure referred to as *regular well* [25]. The Regular Voltage Threshold (RVT) and High Voltage Threshold (HVT) devices in 22nm FDx are fabricated using regular well architecture. A Reverse Body Bias (RBB) is used with conventional well devices where the back gate 'works against' gate voltage, thus increasing the V_{th} effectively. The RBB is used to reduce leakage current and save power.

In a *flipped well* architecture as show in figure 4.4, PMOS is placed over p-well and NMOS over n-well which is absolutely in contrast with conventional structures. The BOX layer plays a great role in this architecture by confining electron flow to form a depletion mode at very low voltage. A Forward Body Bias (FBB) with flipped well architecture effectively decrease the V_{th}, as the back gate bias here 'works with' the gate voltage. The FBB accelerates the performance of a MOS making it very useful in ultra low voltage and power applications. The flipped well architecture is used to fabricate Low Threshold Voltage (LVT) and Super Low Threshold Voltage (SLVT) devices.

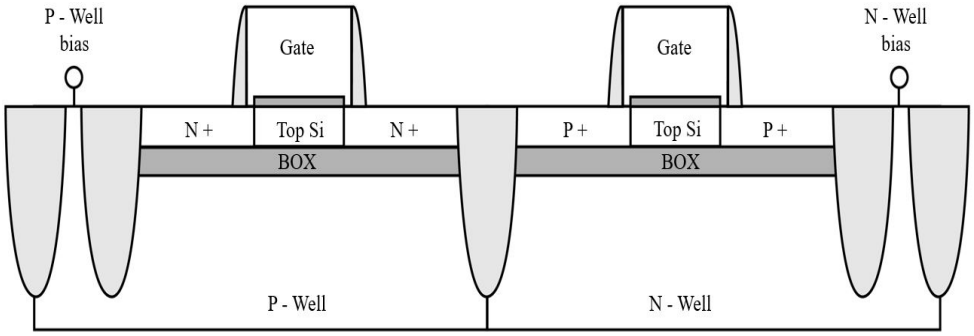

Figure 4.3.: Regular well FDSOI architecture

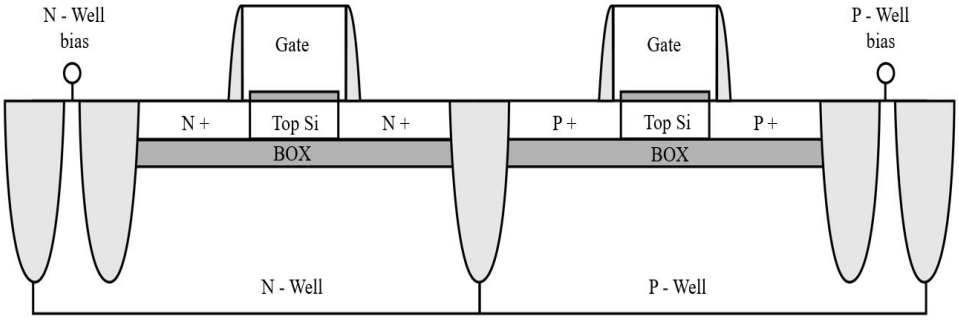

Figure 4.4.: Flipped well FDSOI architecture

The SLVT device is selected for this work to design both analog and digital blocks, because it can provide good performance with very low threshold voltage for low power applications.

5. Design Considerations

Before starting with the design of CS-DAC, basic requirements have to be defined. For 22 nm FDSOI technology, a power supply of 0.8 V with $\pm$ 5 % tolerance is used. The conversion rate of the CS-DAC is 10 MSps and the input is a digital signal of frequency of 5 *MSps*. The output of a CS-DAC is a differential current output pair, terminated by load resistors to generate a voltage signal. For temperature specification, the typical temperature range for consumer electronics from - 40 °C up to + 80 °C is used. Table 5.1 summarizes the basic requirements for the design.

Table 5.1.: Basic requirements of the CS-DAC

Parameter	Min.	Typ.	Max.	Unit
Supply	0.76	0.8	0.84	V
Temperature	-40	27	85	$°C$
Conversion rate	10	10	10	*MSps*
Input data-width	8	10	12	*bit*
Diff. V_{OUT} (p-p)	1	1	1	V

5.1. Architecture

The current steering DAC implementation provides high-speed and high-accuracy with respect to other alternatives. The DAC's output current can quickly charge parasitic capacitances at the output and hence achieve high speeds [26]. Figure 5.1 shows the block diagram of a segmented *N bit* CS-DAC. The digital input is further split into *B bits* LSB and *(N-B) bits* MSB parts. The MSB bits are encoded into $M = 2^{(N-B)} - 1$ *bits* of thermometer/unary code by a binary-to-thermometer decoder. The LSB *bits* are fed to binary delay elements which can be buffers or inverters in order to maintain same delay as that of the decoder. The drivers are used next to provide better driving capability for the decoded signals, as they need to drive huge number of flip-flops.

The decoder usually works asynchronously and thus the data coming out of it needs to

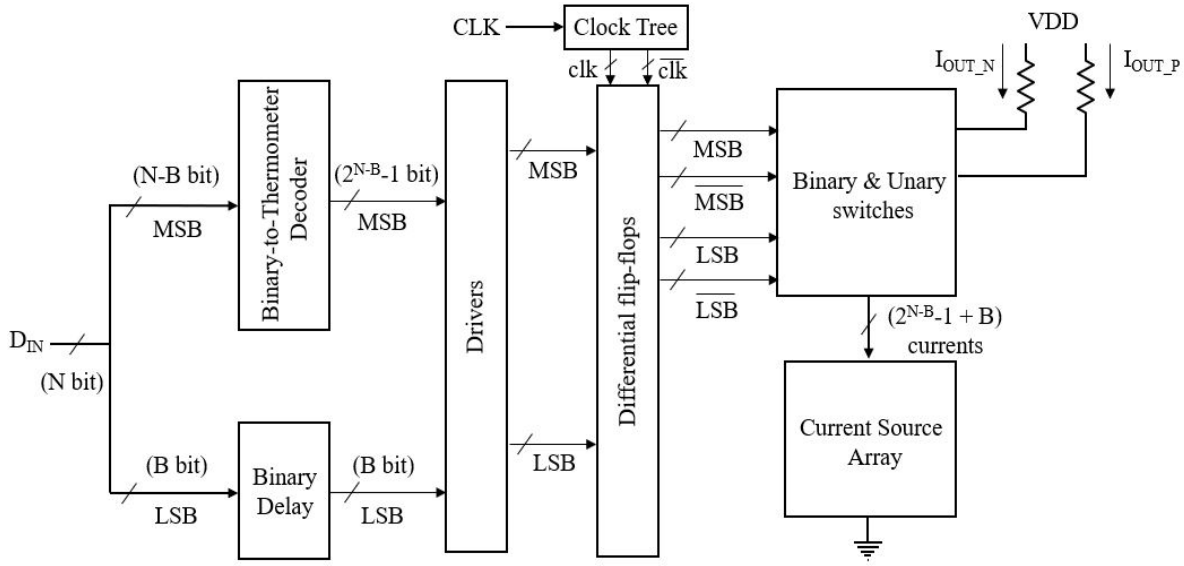

Figure 5.1.: Block diagram of a Current-Steering DAC

be synchronized [26]. Next, after the drivers, the signals are directed to differential flip-flops, usually implemented as Master-slave delay flip-flop (MS-DFF). The use of MS-DFF is not only for synchronisation but to also produce differential signals for the binary and unary switches. The number of MS-DFF required is fairly high and hence a clock tree is proposed in this work to avoid any clock related latency issues. The binary and unary switches are controlled by the differential signals from the flip-flops. The main function of the switching cells is to direct the flow of current from current sources and combine them appropriately. This operation generates an analog current output for the input digital word being converted. Depending on the speed requirements, current sources are usually abutted with the switches or placed separately as current source array (figure 5.1).

5.2. Current Source Array

The current source array (CSA), is a pool of MSB and LSB current sources with biasing circuit placed separately from the switching matrix. The current sources generate the required current, which are then properly switched to the differential output nodes. This current at the output node is linearly converted into a voltage by load resistors.

5.2.1. Current Source

Static and dynamic performances of the CS-DAC discussed in section 2.1 are mostly determined by the current source's accuracy, output impedance and switching time [27]. The

transistor mismatches often known as random errors limits the accuracy of current sources. These random errors are modelled in [9], using a normal distribution with mean zero and a relative standard deviation $\sigma(I)/I$. The statistical relationship between the relative standard deviation and the resolution of a CS-DAC is given in [4] :

$$\sigma(I)/I \leq \frac{1}{2C\sqrt{2^N}} \tag{5.1}$$

where C is a function of INL yield. This is the standard formula used to calculate the required accuracy of the current sources. In this work, the current sources are designed to fit in 8 *bit*, 10 *bit* and 12 *bit* resolutions. As a rule of thumb, from the equation 5.1, a CS-DAC of 8 *bit* resolution will have highest standard deviation on comparison with 12 *bit*. Hence, for a resolution of 8 *bit* to achieve a 3σ or 99.7% INL yield requirement (C = 3.1 from [4]), relative standard deviation is given by :

$$\sigma(I)/I \leq \frac{1}{2 \times 3.1\sqrt{2^8}} \leq 1\% \tag{5.2}$$

The LSB current of $500nA$ is chosen have low power consumption, for a N *bit* CS-DAC this leads to a full scale current of :

$$I_{FS} = I_{LSB}(2^N - 1) \tag{5.3}$$

Using the mismatch model derived in [19], the minimum area of the LSB current source transistor is given by :

$$WL_{min} = \frac{1}{2(\sigma(I)/I)^2}(A_\beta^2 + \frac{4A_{VT}^2}{(V_{gs} - V_{th})^2}) \tag{5.4}$$

where, A_β and A_{VT} are the transistor mismatch parameters given by the foundry. For the SLVT transistors from 22 nm FDSOI technology, the mismatch parameters are estimated to be A_β = 0.432% μm and A_{VT} = 1.35 $mV\mu m$ [28].

Using equation 5.4 and 5.2, the minimum area of current source transistor versus $\sigma(I)/I$ is plotted. It can be observed from figure 5.2, that the area requirement is very high as $\sigma(I)/I$ goes below 0.5%. The $\sigma(I)/I$ for a 12 *bit* CS-DAC, from equation 5.2 gives 0.25 % and it would consume very large area. This is also one of the reason for selecting $\sigma(I)/I$ as 1 % , it is a good trade-off between area and performance between three resolutions. Figure 5.3 shows the dependency of area with respect to gate overdrive voltage. Increasing the $(V_{gs} - V_{th})$ of the current source transistor relaxes the area requirement. But, the value of $(V_{gs} - V_{th})$ is limited by the required output voltage swing and the operation of transistor in saturation region. For a supply voltage of 0.8 V, the gate overdrive voltage of 100 mV is a reasonable choice. After

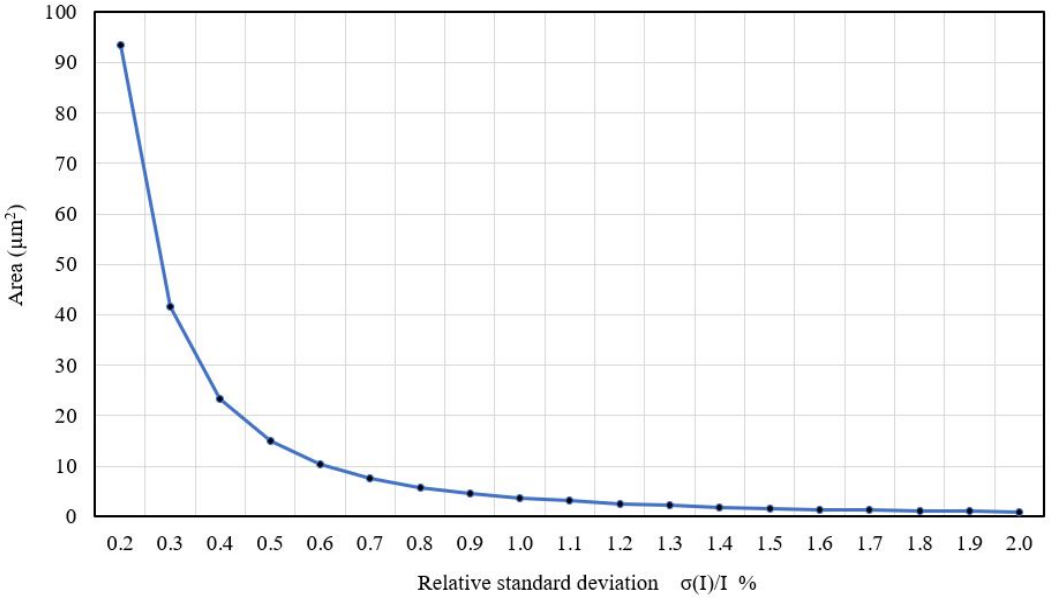

Figure 5.2.: Area vs relative standard deviation

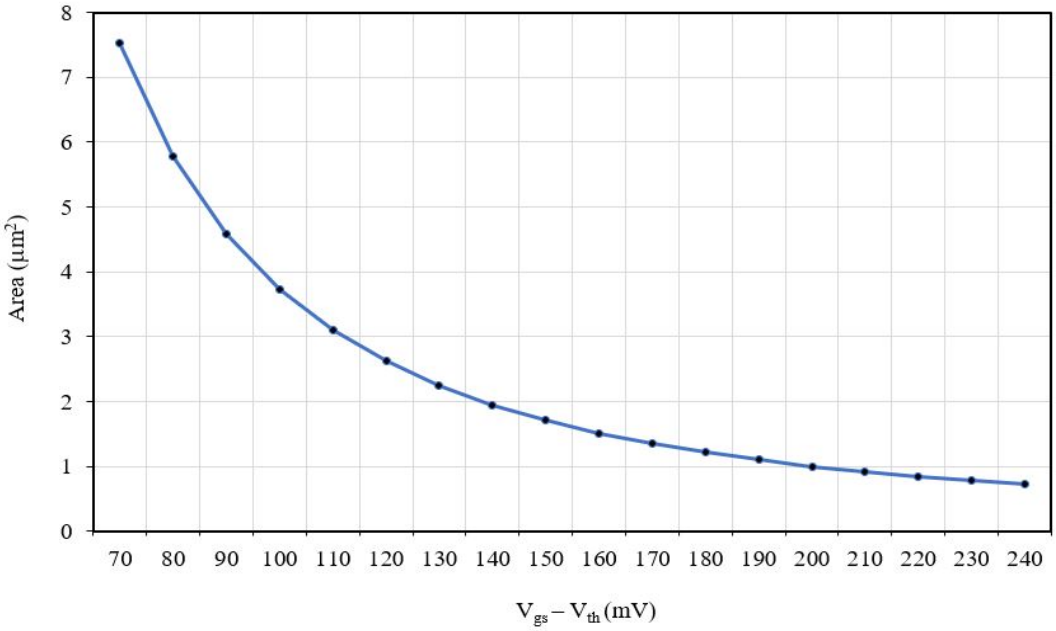

Figure 5.3.: Area vs (V_{DS})

analysing the parameters from the plots, minimum area of the LSB current source transistor is calculated :

$$WL_{min} = 3.7\mu m^2 \approx 4\mu m^2 \tag{5.5}$$

The basic current source circuit includes a current source transistor and two complementary switches (figure 5.4 (*left*)). Using a single transistor would have limited output impedance and its drain-source voltage (V_{DS}) would be prone to variations. This can be solved by adding a cascode transistor.

Cascoded Current Source In some cases use of single current source transistor is not sufficient, because the output impedance (Z_{imp}) seen at the drain of the switch transistor is small. This leads to poor accuracy in mirroring current and limits the performance of the converter. Therefore, we implement an additional cascode transistor to :

- Improve the node isolation of the current source transistor to maintain fairly constant V_{DS} which promises better accuracy of current source.
- Increase the Z_{imp} of the current source, to deliver highly saturated current values.

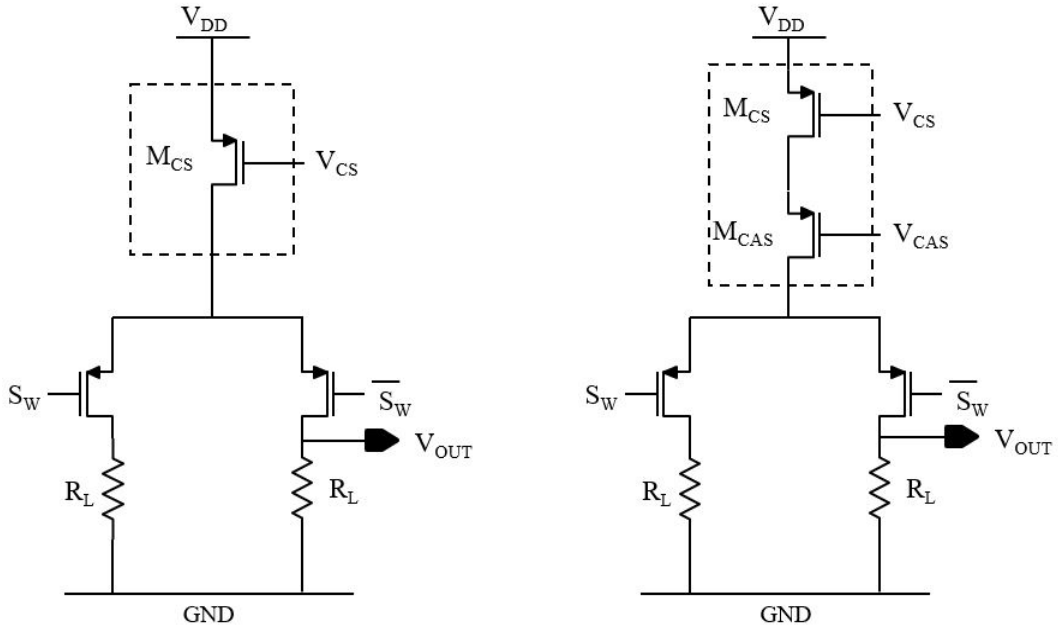

Figure 5.4.: Simple current source (*left*) and cascoded current source (*right*)

The two current source toplogies are shown in figure 5.4, the use of cascode transistor will increase the area of current source and reduces the output swing. On the other hand it provides better accuracy, which is of high priority. The current source transistor (M_{CS}) is dimensioned with area requirements derived from equation 5.4 and with low aspect ratio to have better accuracy. The cascode transistor (M_{CAS}) is sized by the fact that same current

flows through it and the gate-length is selected larger than the minimal value to provide higher Z_{imp}.

5.2.2. Current Source Biasing

As discussed in chapter 3, to have low power dissipation and high output swing, a low voltage cascode current mirror topology is used. Figure 5.5 shows the biasing circuit designed for the cascoded current sources. The NMOS sections are labelled as global biasing while the PMOS sections are labelled as local biasing. The labelling for biasing sections are done in accordance to the placement of biasing circuit in the layout. In order to have better DNL and INL performance in post-layout simulations, the realisation of two biasing sections can be done with suitable layout techniques. The biasing circuit is designed to bias the M_{CS} with a

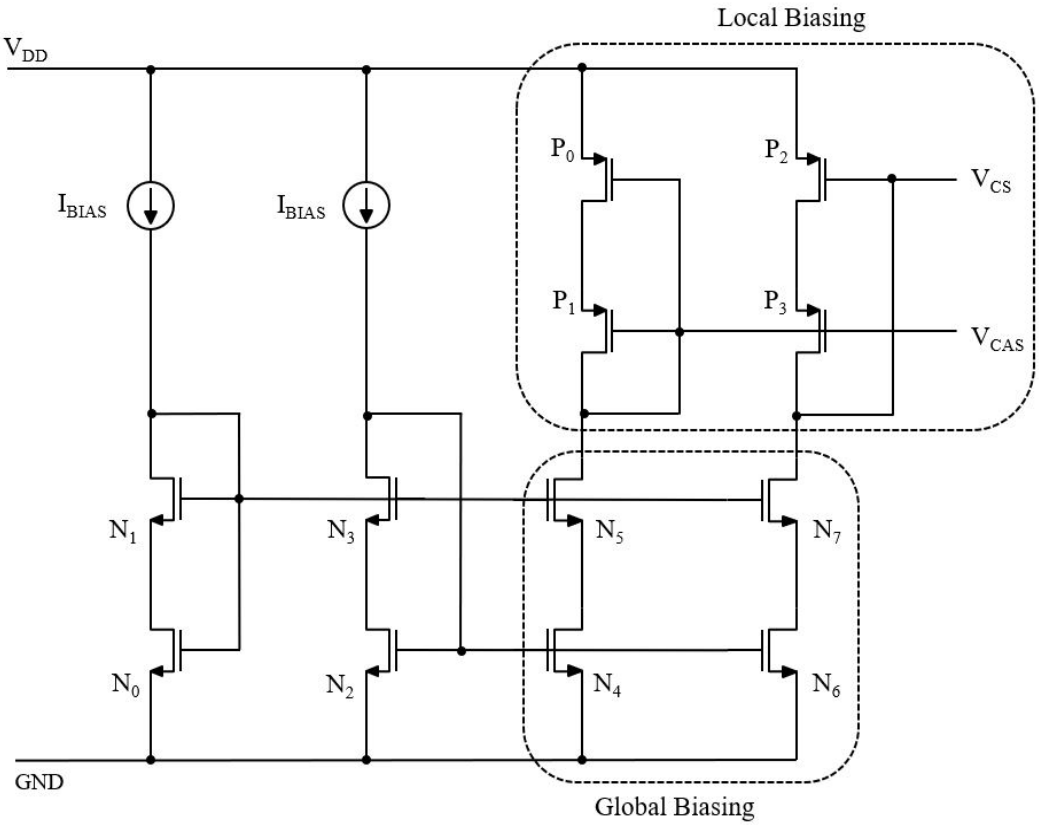

Figure 5.5.: Low voltage cascode biasing circuit

V_{DS} of $50mV$ above V_{DSsat} to ensure that it operates in saturation region for all PVT corners.

5.2.3. Simulations

The current source units are designed for four binary-weighted *LSBs*: 1*X*, 2*X*, 4*X* and 8*X* times that of I_{LSB} and unary-weighted *MSB* is 16*X* to that of I_{LSB}. The design is verified using a simple test bench and by varying the voltage at the node of each current sources. The figure 5.6, shows the simulation of four *LSB* current sources and one *MSB* current source. It is observed in each case, that the current saturates exactly to its scaled values for a minimum voltage of $|2V_{DS}|$.

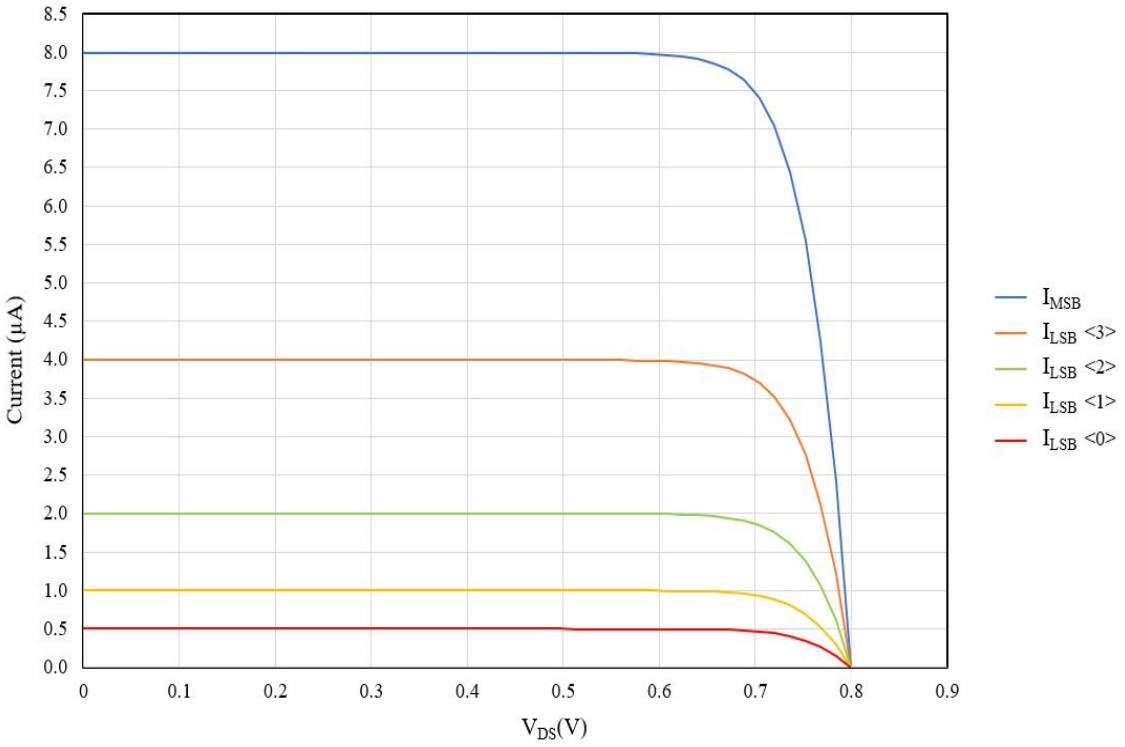

Figure 5.6.: Simulation of one unary and four binary current sources

After determining the expected final values of current sources through direct current (DC) simulations, additional simulations have to be done. Further verification of the designed current sources have to be done based on Monte-Carlo (MC) simulations. The MC simulation uses the 'Monte-Carlo algorithm' to distribute specified parameters randomly between the defined region [6]. The simulation is performed over 500 different samples distributing the process and mismatch values for all the corners mentioned in table 5.1. The output of this simulation would be a histogram of normal distribution with a mean and standard deviation for the given number of samples. In this case, the MC simulation is performed only for unary (*MSB*) current source. This is because in a segmented CS-DAC, the DNL/INL errors are highly dependent on unary coarse variations. A specification range of - 250 *nA* to + 250 *nA* is set to check whether unary current source mismatch is below $\pm$ 0.5 *LSB*.

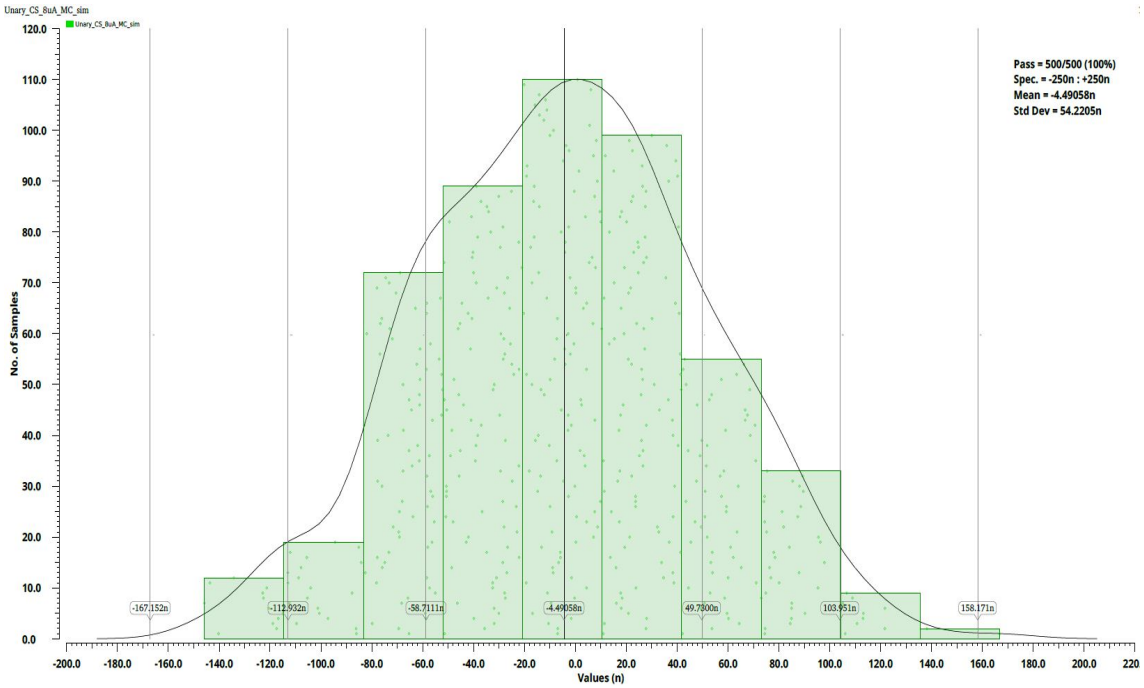

Figure 5.7.: Monte-Carlo simulation for unary current source

Diagram 5.7 represents the histogram of MC simulation performed on unary current source for the typical corner. The resulting standard deviation σ is about 54 nA. In accordance to industry standards, this is an acceptable value, because for a 3σ design a value of about 163 nA can be calculated. This value is lesser than 0.5 LSB that is 250 nA. This simulation ensures that the DNL/INL of a converter would not go beyond $\pm$ 0.5 LSB for all samples. An improvement of this value can only be achieved by enlarging the current source unit [6]. However, this will result in more chip area and is not acceptable for economic reasons.

5.3. Binary-to-Thermometer Decoder

The binary-to-thermometer decoder is the essential digital block in a segmented CS-DAC. Designing a decoder for M inputs provides $(2^M - 1)$ number of outputs, and undoubtedly it consumes very large area. A sophisticated solution was proposed in [29], it suggests using the row-column switching matrix, which would require two identical decoders of M/2 inputs. This idea reduces the large decoder area and complexity, but needs additional decoding logic in the row-column matrix (section 5.5). Since we are aiming to design a scalable CS-DAC, the decoder is designed considering the requirements of highest resolution. For a 12 *bit*, with unary segmentation of 8 *MSB* we need to use two identical 4 *bit* (M/2) binary-to-thermometer decoders. The truth table of a 4 *bit* binary-to-thermometer decoder is shown in table 5.2 (4 inputs and 15 outputs).

Table 5.2.: Truth table of a 4 *bit* Binary-to-Thermometer Decoder

Input				Output													
B_3	B_2	B_1	B_0	T_{14}	T_{13}	T_{12}	T_{11}	T_{10}	T_9	T_8	T_7	T_6	T_5	T_4	T_3	T_2	T_1	T_0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
0	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1
0	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
1	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
1	0	0	1	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
1	0	1	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
1	0	1	1	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1
1	1	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1
1	1	0	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

The decoder logic can be implemented either by using the standard cell libraries or by a full custom manually designed logic. In both cases, special care has to be taken for optimizing the timing constrains to improve the operating speed of the decoder [3]. The decoder is first implemented in Verilog-A, after checking its functionality with scalable CS-DAC, it is then realised using standard cells from the digital library.

5.4. Master-Slave Delay Flip Flop

The Master-slave delay flip flop (MS-DFF), is designed by cascading two delay flip flops (DFF) and complementing one of its enable input. Usually an inverter is used to generate the complementary clock signal. As shown in the figure 5.8, MS-DFF takes only two inputs, the D (data) input and the clock signal as enable input. The MS-FF generates two outputs Q and Q_N, which are complementary to each other. This gives us two synchronised differential signals to control the switches.

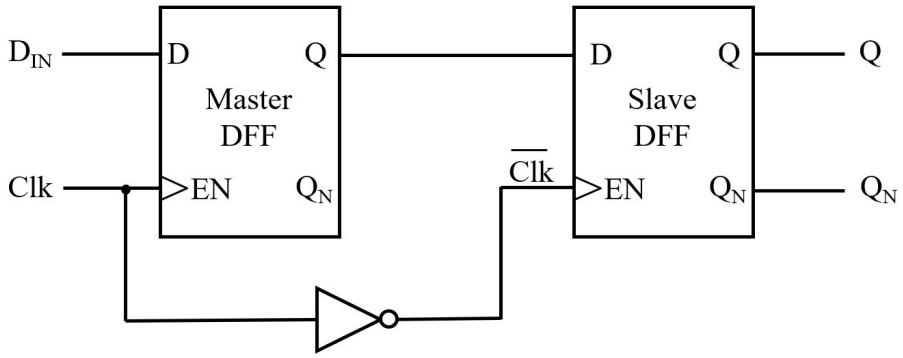

Figure 5.8.: The Master-slave delay flip-flop

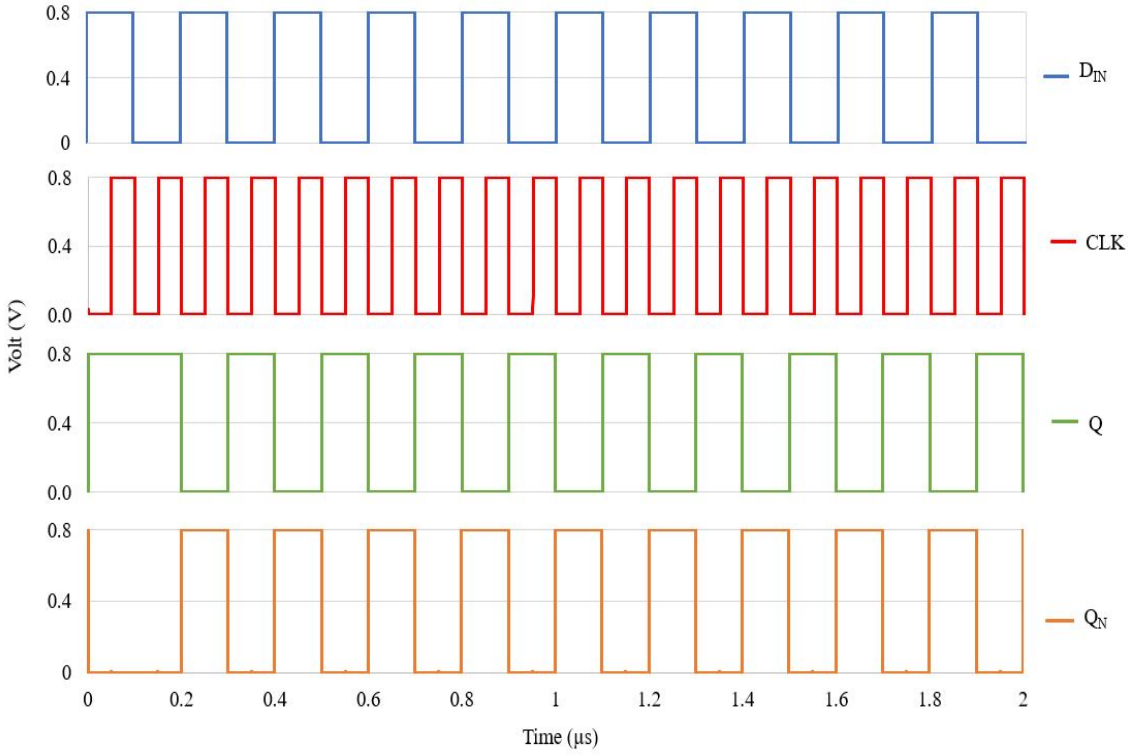

Figure 5.9.: Timing diagram of designed MS-DFF

It is important to know that these two flip flops are never enabled at the same time, in simple words, the slave DFF just follows the master DFF when the clock is inverted. The MS-DFF synchronise the data and shape their physical signals in a proper way for the next block i.e., the current switching cell. The delay of MS-FF is matched with the delay of binary-to-thermometer decoder, this brings better synchronisation between the binary and unary input signals. In this work MS-DFF is designed using two DFF from the standard cell library and figure 5.9 shows its simulation waveforms.

Each current switching cell requires one MS-DFF, so for a 12*bit* CS-DAC of 8:4 segmentation the number of flip-flops required is 255 + 4. Driving such a huge number of flip flops by one clock source would bring skewing problems during real implementation. Clock skew is the maximum difference in the arrival time of a clock signal at two different components. In order to make sure that the clock gets distributed evenly to all sequential elements (MS-DFF) in CS-DAC, a clock tree is implemented.

5.5. Switching Matrix and Decoding Logic

In a segmented CS-DAC, the most common problem is the arrangement of large number of current switching cells, distribution of data and clock to the respective cells. In this section, we only focus on arrangement of unary switching cells, as the arrangement of binary switching cells is simple and straightforward. The 2D matrix architecture proposed in [29], is an elegant approach that arranges switching cells in a compact matrix, making the distribution much more reasonable than in a linear way.

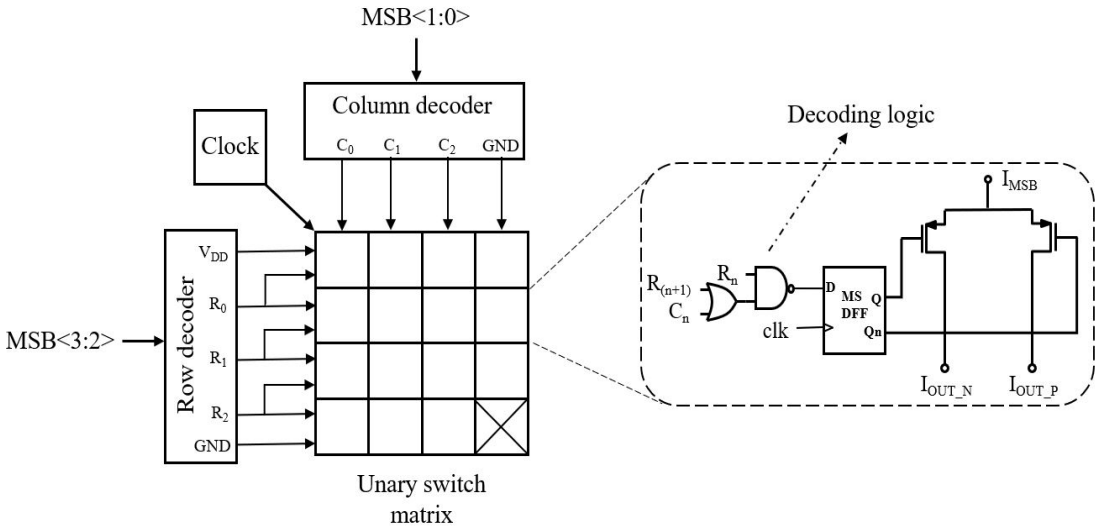

Figure 5.10.: Example of 2D matrix architecture for a 4:4 segmentation level

Figure 5.10 shows the 2D matrix arrangement, which consists of a row decoder, a column decoder and $2^M - 1$ switching cells. Each cell consists of a decoding logic, a MS-DFF and a switching pair. The unary switch matrix consists of three types of rows. They are [29]:

- Rows in which all of the current cells are turned on.
- Rows in which all of the current cells are turned off.
- A certain row in which current cells are turned on depending upon the column decoder signal.

One extra row before R_0 and one extra column after C_2 are added by connecting them to V_{DD} and GND respectively (figure 5.10). These extra cells are to match the number of unary units to that of linear architecture. The decoding logic, senses the thermometer codes of two consecutive rows (R_{n+1}, R_n) and one column C_n to determine which switch in the differential pair should be turned on. The decoding logic shown in figure 5.10 is for PMOS switches. A complete architecture of 12 *bit* CS-DAC for 8:4 segmentation is shown in figure 5.11 and it includes all the design blocks discussed in this chapter.

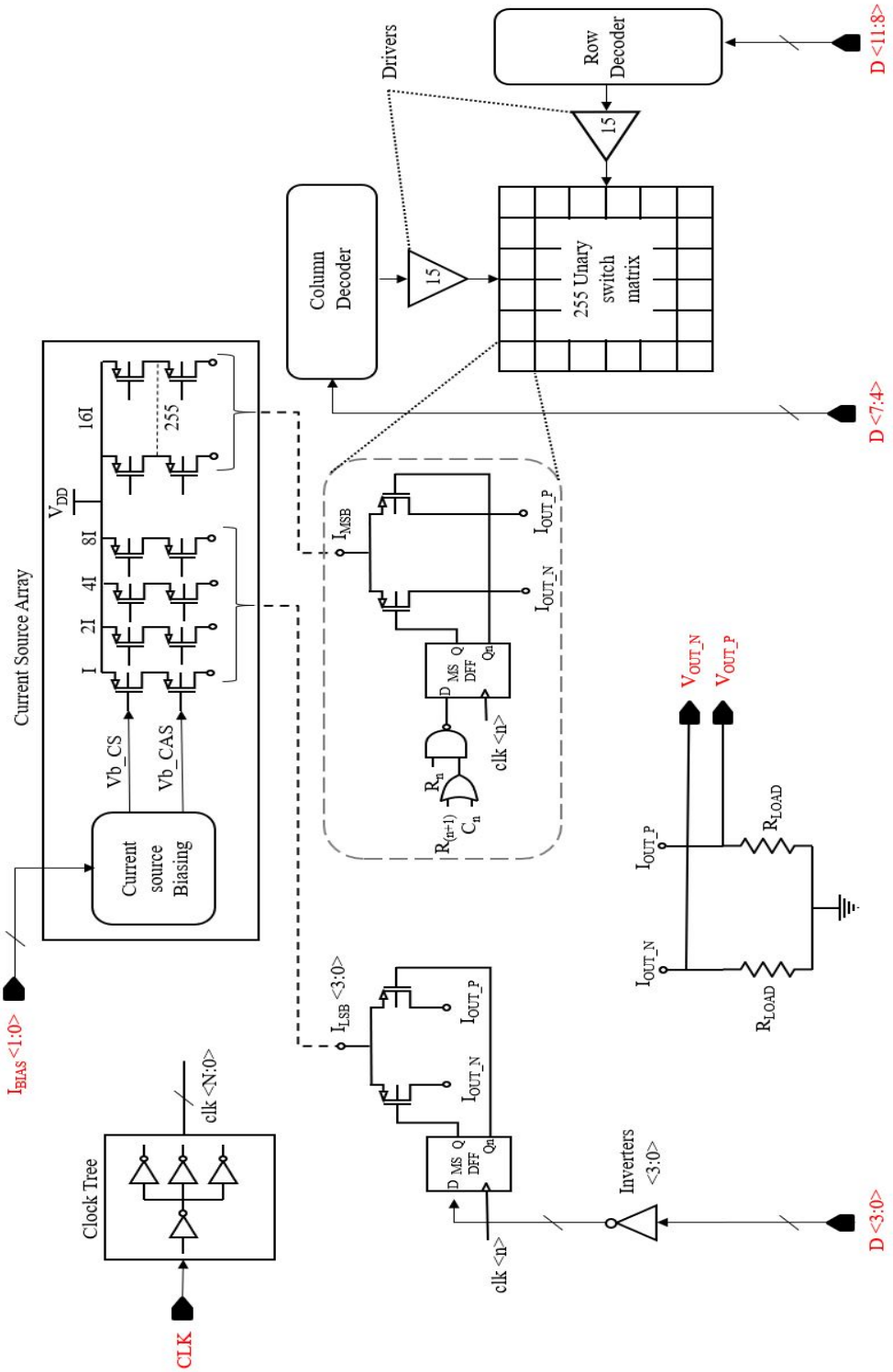

Figure 5.11.: The 12 bit current-steering DAC of 8:4 segmentation level

6. Implementation of Scalable CS-DAC

In this chapter, the implementation of resolution scalability for the CS-DAC is explained in detail. This chapter is divided into two main parts, the first part covers the steps taken to achieve scalability and the next part discuses about the results of designed scalable CS-DAC for all three resolutions.

6.1. Resolution Scalability

During the implementation of segmented CS-DAC, blocks are often divided into binary and unary blocks. In chapter 3, we selected the segmentation levels for the three resolutions and was tabulated in table 3.1. Since the binary segmentation is maintained same for all three resolutions, we concentrate only on scaling the unary block. The 12 *bit* CS-DAC forms the top level design from which the resolution can be scaled down. In unary block, the sub-blocks that need to be scaled are thermometer decoders, switching matrix and CSA. A logic is proposed to achieve control signals from modes S_1 and S_0 (table 6.1). A 2 *bit* binary-to-thermometer decoder is used as a mode selection decoder to generate the control signals.

Table 6.1.: Truth table for control signals

Modes		Control signals			Resolution
S_1	S_0	X_2	X_1	X_0	(*bit*)
0	0	1	1	1	-
0	1	0	1	1	8
1	0	0	0	1	10
1	1	0	0	0	12

6.1.1. Scalability for Thermometer Decoder

In chapter 5, we discussed about implementation of 4 *bit* binary-to-thermometer decoder. It is not necessary to scale the entire decoder itself, but there need to be a proper assignment of data at the inputs of row and column decoders. When the input data is directly fed to the

input of decoders, during scaling they would be assigned to decoders as shown in figure 6.1. In this case, the input assignment to the row and column decoders would not be symmetric for 10 *bit* and 8 *bit* modes. No matrix exists in 8 *bit* as there are no inputs assigned to the row decoder. This would completely affect the operation of matrix architecture and the unary switching would no longer be reliable.

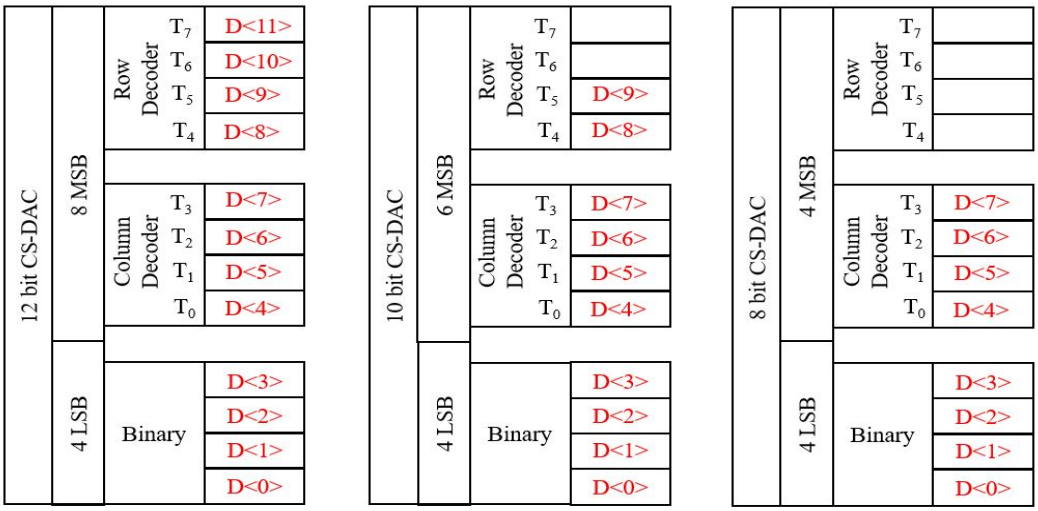

Figure 6.1.: Direct assignment of input to row and column decoders

A new set of enable signals are used to control the input assigning at the row and column decoders. The truth table is given in table 6.2 for the logic expressions given below.

$E_{12} = S_0$ & S_1
$E_{10} = \overline{S_0}$
$E_8 = S_0$ & $\overline{S_1}$

Table 6.2.: Truth table for enable signals

Modes		Enable signals			Resolution
S_1	S_0	E_{12}	E_{10}	E_8	(*bit*)
0	0	-	-	-	-
0	1	0	0	1	8
1	0	0	1	0	10
1	1	1	0	0	12

With the help of enable signals an additional logic is proposed for proper input assignment. These additional logic expressions are derived for each input of two decoders ($T_7 - to - T_0$), using simple Karnaugh mapping technique [30]. Now the assignment of digital data at the

inputs of decoders during scalability will be exactly as illustrated in figure 6.2. In schematic, they can be placed inside the mode selection decoder (figure 6.3), for a reason that they are driven by same signals S_1 and S_0.

$T_7 = E_{12}$ & D_{11}
$T_6 = E_{12}$ & $D_{10} \vee E_{10}$ & D_9
$T_5 = E_{12}$ & $D_9 \vee E_{10}$ & $D_8 \vee E_8$ & D_7
$T_4 = E_{12}$ & $D_8 \vee E_{10}$ & $D_7 \vee E_8$ & D_6

$T_3 = E_{12}$ & D_7
$T_2 = E_{12}$ & $D_6 \vee E_{10}$ & D_6
$T_1 = D_5$
$T_0 = D_4$

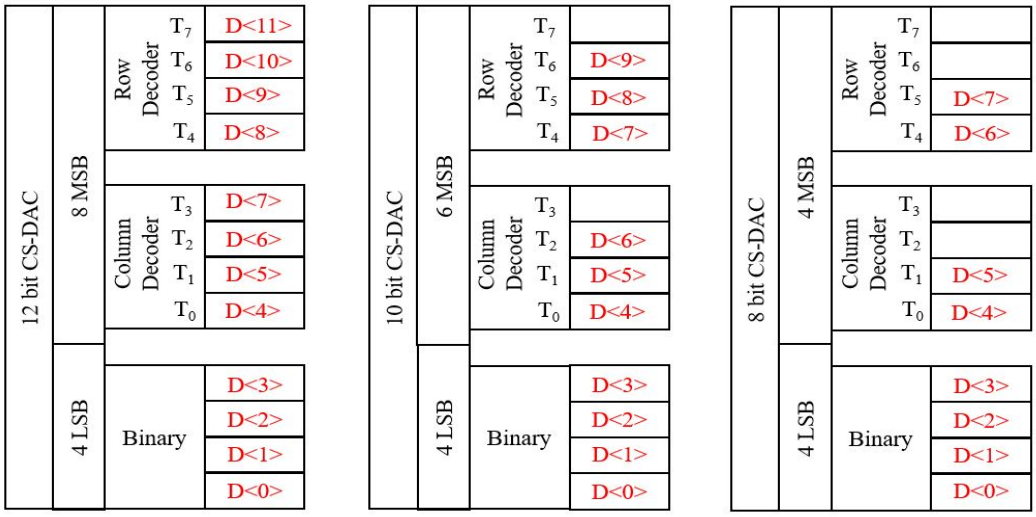

Figure 6.2.: Proper assignment of inputs by additional logic control

6.1.2. Scalability for Switching Matrix

The next sub-block to be scaled is switching matrix. For a 12 *bit* CS-DAC of 8:4 segmentation, the number of switching cells required is $2^M - 1 = 255$. In the same way, the number of switching cells required for 10 *bit* and 8 *bit* are 63 and 15 respectively. Scalability in switching matrix can be achieved by grouping the necessary cells for 10 *bit* and 8 *bit* in the top level 255 switch matrix (12 *bit*). After grouping them, each group is provided with a control signal as shown in figure 6.3. The idea is to turn off each and every cell in the group by just one control signal. For example, when the CS-DAC needs to be operated for 10 *bit* resolution, then the control signal X_0 turns off all the cells associated with group C.

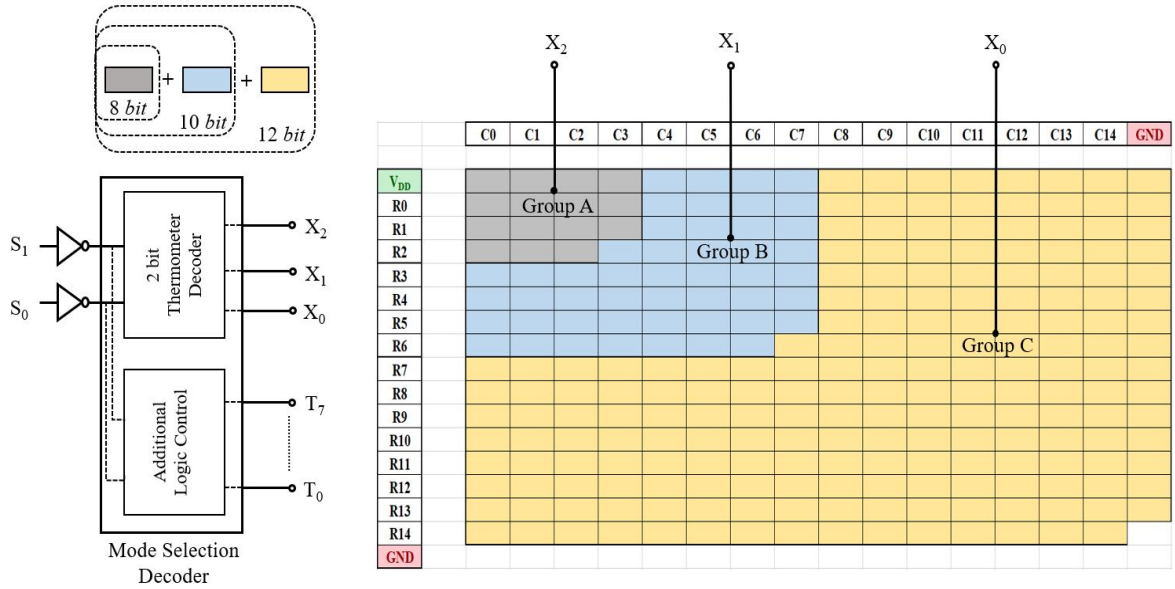

Figure 6.3.: Implementation of scalabilty in switching matrix

Table 6.3.: Grouping of cells in switching matrix

Modes		Control signals			Group A	Group B	Group C	Resolution
S_1	S_0	X_2	X_1	X_0				(*bit*)
0	0	1	1	1	-	-	-	-
0	1	0	1	1	ON	OFF	OFF	8
1	0	0	0	1	ON	ON	OFF	10
1	1	0	0	0	ON	ON	ON	12

The table 6.3 gives the overview of grouping cells in the matrix architecture. In order to have such a functionality, the individual operation of each cell is to be analysed :

- The control signal X_N should act as an enable signal for each cell, hence conventional decoding logic mentioned in section 5.5 needs to be modified.
- Another salient step is to power off the whole switching pair (both switches), when the corresponding group is turned OFF.

Therefore, a new idea for switching cell is proposed to support scalability. First, an OR gate is added just after the conventional decoding logic and one of its input is driven by control signal X_N. Second, the switching architecture is modified with new additional switches. The additional switches are used to switch the gate connection of main switching

pair to MS-DFF or V_{DD}. As shown in figure 6.4, when the gates of M_1 and M_2 are connected to MS-DFF the operation of switching pair is same as before. When the control signal decides to turn OFF the entire group, the gates of M_1 and M_2 are connected to V_{DD}, as a result no current flows.

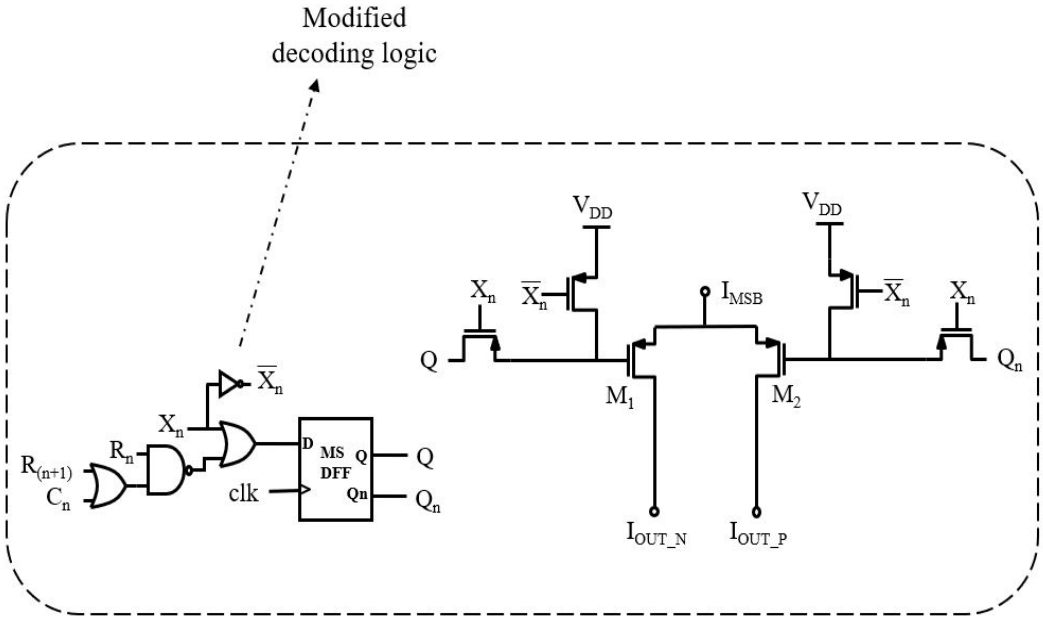

Figure 6.4.: The proposed switching cell architecture

6.1.3. Scalability for Current Source Array

The steps for scalability in CSA is similar to that of switching matrix. The 255 unary current sources are grouped into three groups as shown in figure 6.5. The three groups are monitored by a pair of switches and respective control signal. In each group, the gates of current source transistors are connected to either its bias voltage or to V_{DD}. The use of switches is to power off the entire group during scaling. It is important note that, here the current sources are grouped and powered off, whereas in switching matrix the cells are powered off individually.

The placement of 255 unary current sources in CSA with respect to layout is of most important. The 16 unary current sources (1 dummy) of group A at the center is surrounded by 48 current sources of group B, which is then surrounded by 192 current sources of group C. As illustrated in figure 6.5, the current sources in each group are separated into four quadrants spreading uniformly across the center. Such an arrangement would guarantee symmetrical layout, not only for 12 *bit*, but also when the CS-DAC is scaled down to 10 *bit* or 8 *bit*. It is important to use similar switch pairs for the gates of cascode transistors as well and figure 6.5 shows only the gate connections of current sources for illustration purpose.

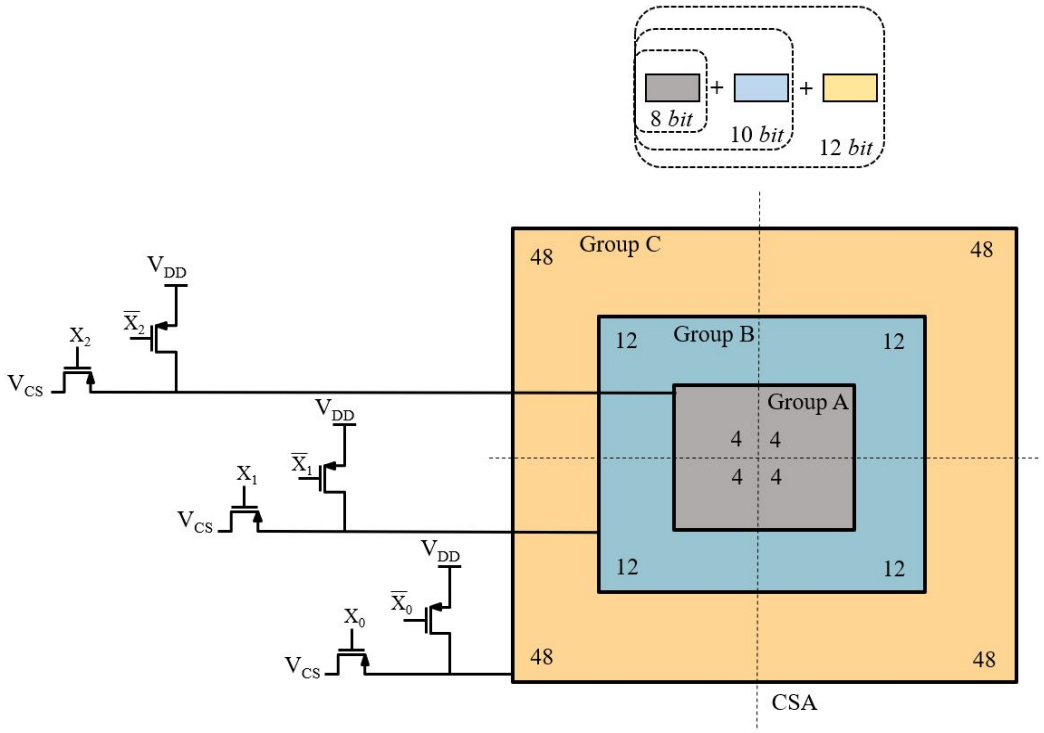

Figure 6.5.: Implementation of scalabilty in current source array

6.2. Design Results

In this section, the functionality and performance of the designed CS-DAC is analysed through measurements. The first step is to test the scalability feature of the CS-DAC. The next step is to analyse the static and dynamic performance for all three resolution modes: 8 *bit*, 10 *bit* and 12 *bit* separately. The simulations have been performed on the designed CS-DAC using the spectre simulator in the Cadence design tool.

6.2.1. Resolution Scalability Test

In order to check the working of scalability, three different test benches are designed. For these simulations, the binary-to-thermometer decoders are implemented using Verilog-A and the value $R_{LOAD} = 250\Omega$ remains same. From table 6.1, for modes S_1 and S_0 three cases are to be used and each case is assigned to each test bench respectively. Measuring the I_{FS} in each case determines the operating resolution of the CS-DAC and theoretically it is calculated using equation 5.3.

For 8 *bit*, 10 *bit* and 12 *bit* resolution, the I_{FS} should be :

$$I_{FS\_8bit} = 0.5\mu A \times (2^8 - 1) = 127.5\mu A \tag{6.1}$$

$$I_{FS\_10bit} = 0.5\mu A \times (2^{10} - 1) = 511.5\mu A \tag{6.2}$$

$$I_{FS\_12bit} = 0.5\mu A \times (2^{12} - 1) = 2.0475mA \tag{6.3}$$

A transient simulation is performed and two differential currents I_{OUTP} and I_{OUTN} are plotted. Figure 6.6 shows the output current when the CS-DAC was operated in 8 *bit* mode. The CS-DAC delivers a current of 127 μA which is nearly same as in equation 6.1. In the same manner, when the CS-DAC was operated in 10 *bit* and 12 *bit* modes, it delivered an output current of 501 μA and 1.8 mA respectively. Figures 6.7 and 6.8 shows the output current when the CS-DAC was operated in 10 *bit* and 12 *bit* mode respectively. The achieved simulation values of I_{FS} ensures the proper functioning of the proposed resolution scalability feature.

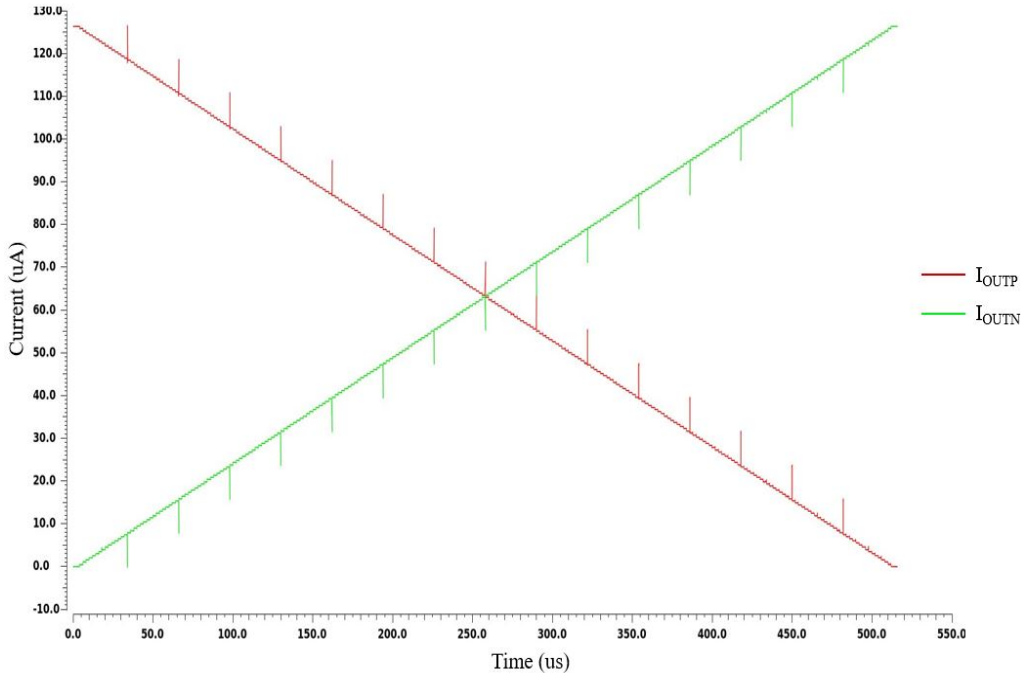

Figure 6.6.: The I_{FS} of CS-DAC in 8 bit resolution mode

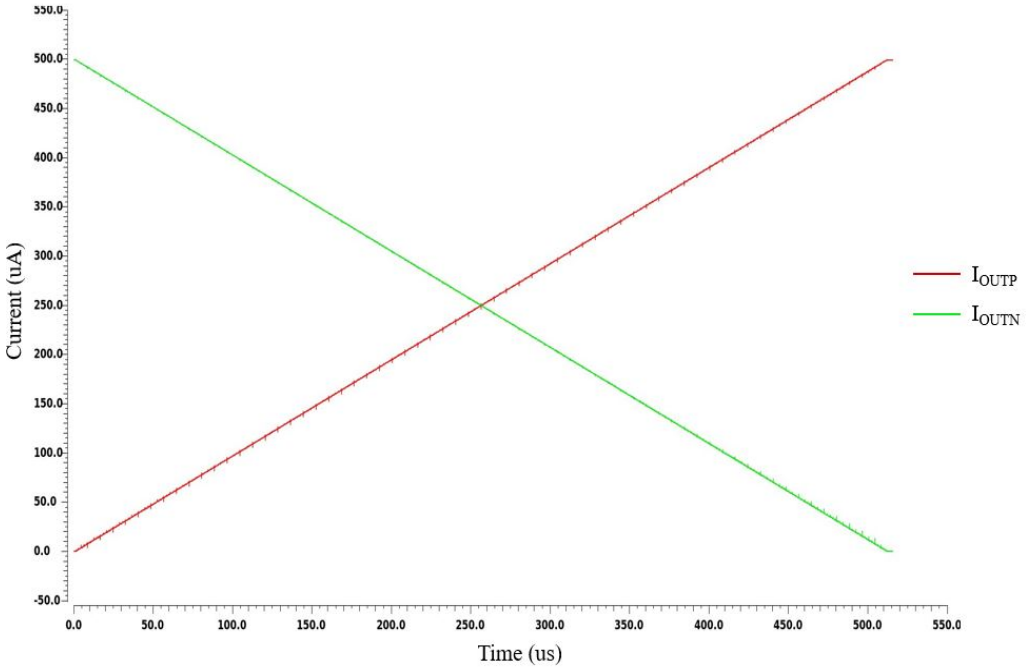

Figure 6.7.: The I_{FS} of CS-DAC in 10 bit resolution mode

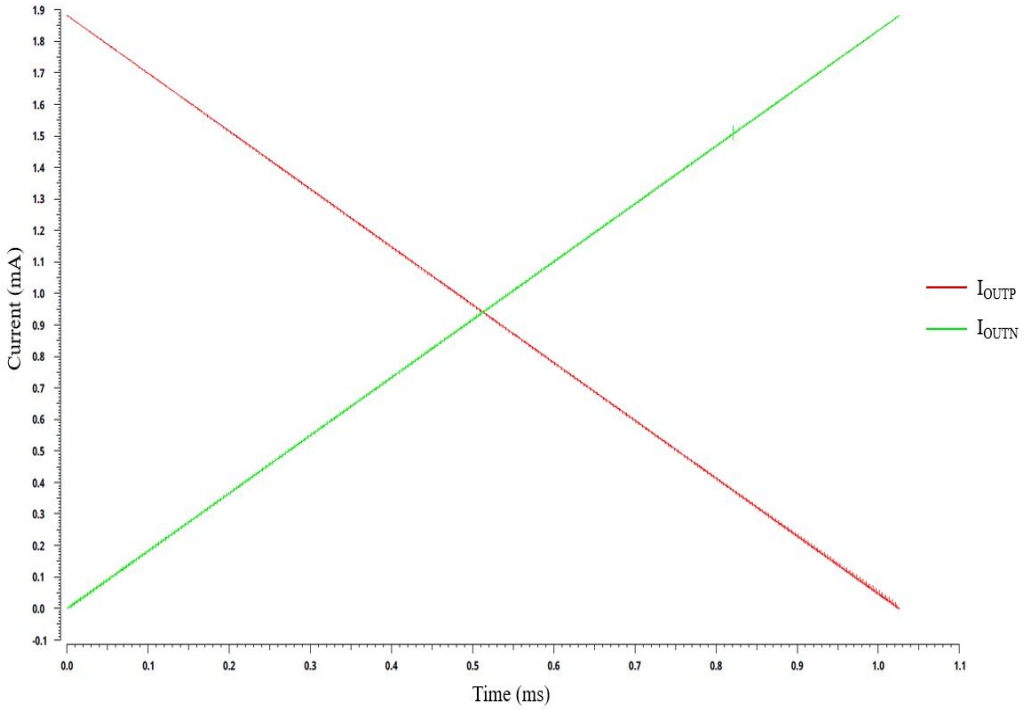

Figure 6.8.: The I_{FS} of CS-DAC in 12 bit resolution mode

The difference between the theoretical and simulation value of the I_{FS} is due to the gain error. Since a single biasing circuit is used to bias all the current sources, it experiences a huge DC load which slightly reduces the I_{LSB} value than expected. It is important to note that, the gain error was increased as the operating resolution was increased. This is because of the grouping in CSA (section 6.1.3), as the biasing circuit sees different number of current sources as the resolution is varied.

6.2.2. Static Performance of Scalable CS-DAC

To get the real schematic simulations for static and dynamic measurements, we now realise the binary-to-thermometer decoders with logic gates from standard cell library. To measure the static performances such as DNL and INL, a ramp input is fed into an ideal ADC (Verilog-A model) to generate the required digital codes. These digital codes are applied at the input of CS-DAC in the test bench, which then generates a ramp output signal with a frequency same as that of digital codes. One complete set of simulation data (ramp output) is necessary to evaluate DNL and INL.

The CS-DAC for selected 8 *bit* mode, is simulated with a digital input code of frequency 500 *KSps* at sampling rate of 10 *MSps* to have better settling condition. The figure 6.9 shows the DNL profile and indicates that the maximum DNL is 0.06 *LSB*. A small analysis that can be made to get an insight on the DNL profile, is to determine which transitions are responsible for the peaks. It can be observed from figure 6.9, that the peak DNL error is at every unary transition. In this work, when the CS-DAC is operating at 8 *bit* mode, it will have 15 unary transitions which is clearly shown in the DNL profile. The INL at any point in the transfer characteristics is the running sum or the accumulation of all previous DNL errors. Figure 6.10 shows the INL profile of range $\pm$ 0.05 LSB. Note that the results are almost close to an ideal DAC, this is due to the fact that layout matching issues and parasitics are not considered in schematic simulations.

Further, figures 6.11 and 6.12 shows the DNL and INL profiles of 10 *bit* mode respectively. A digital input code of frequency 1 *MSps* is applied for this case. Figures 6.13 and 6.14 shows the DNL and INL profiles of 12 *bit* mode, for the digital input code of frequency 5 *MSps*. The reason for selecting different input frequency is to save the simulation time, because for 10 and 12 *bit* resolutions the run time to cover all digital codes would be too long. The analysis done for 8 *bit* mode on the DNL and INL profiles can be applied here too. The complete summary of maximum DNL and INL values for the designed scalable CS-DAC is tabulated in table 7.1.

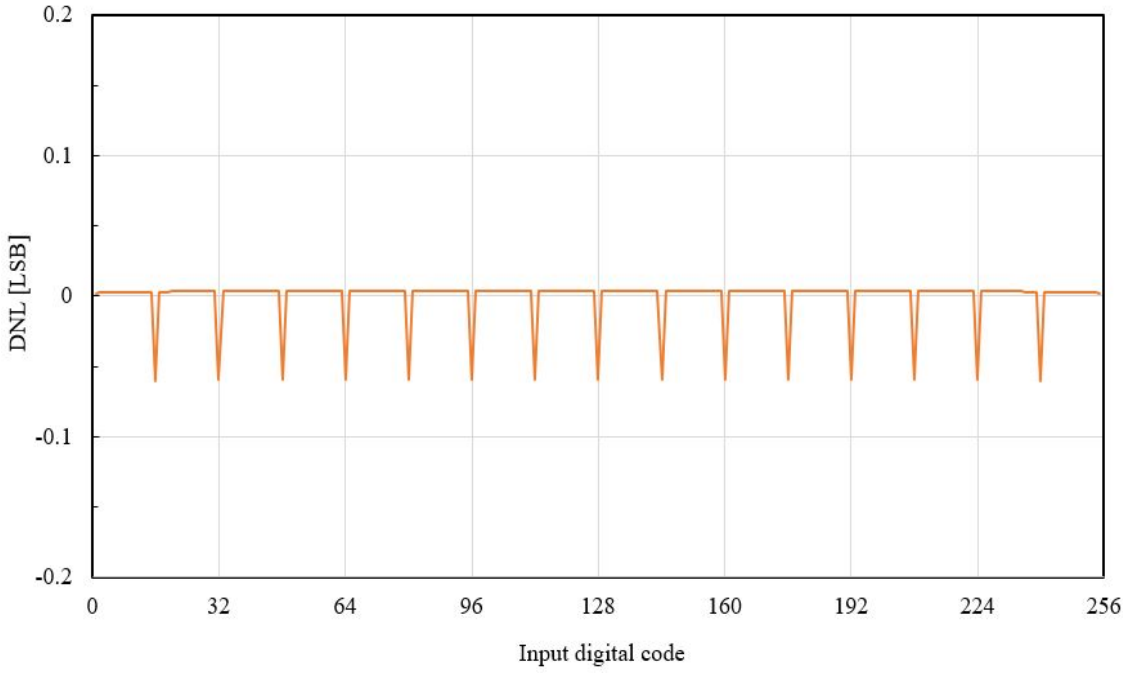

Figure 6.9.: Measured DNL profile of CS-DAC in 8 *bit* mode ($F_{IN} = 500KSps$)

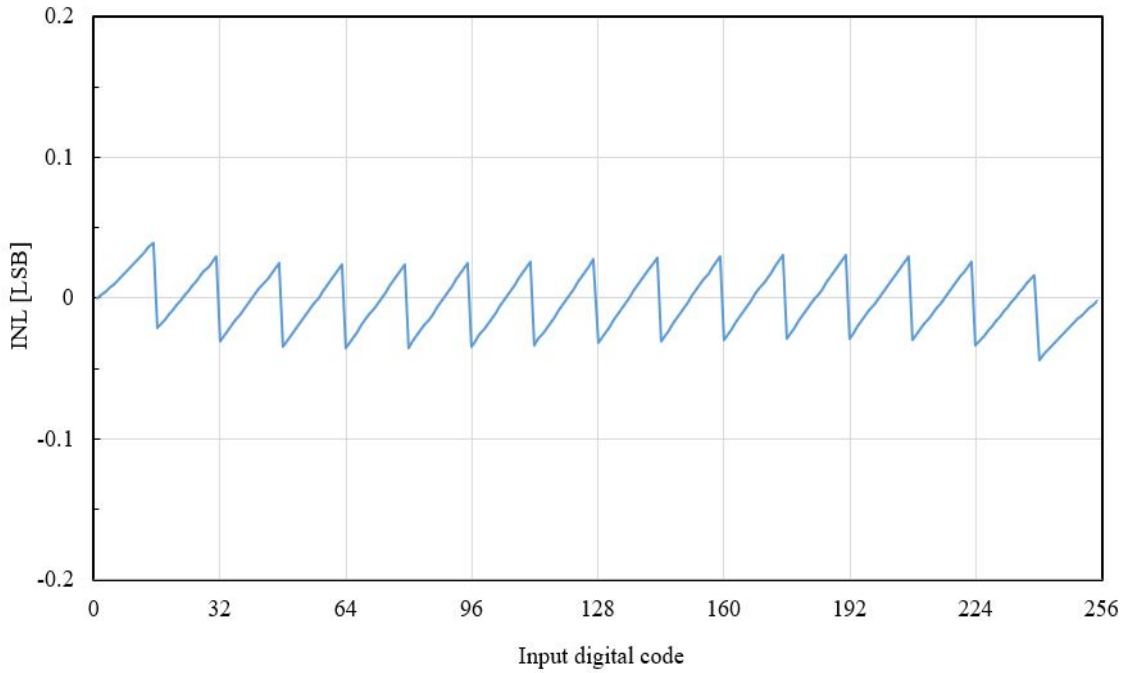

Figure 6.10.: Measured INL profile of CS-DAC in 8 *bit* mode ($F_{IN} = 500KSps$)

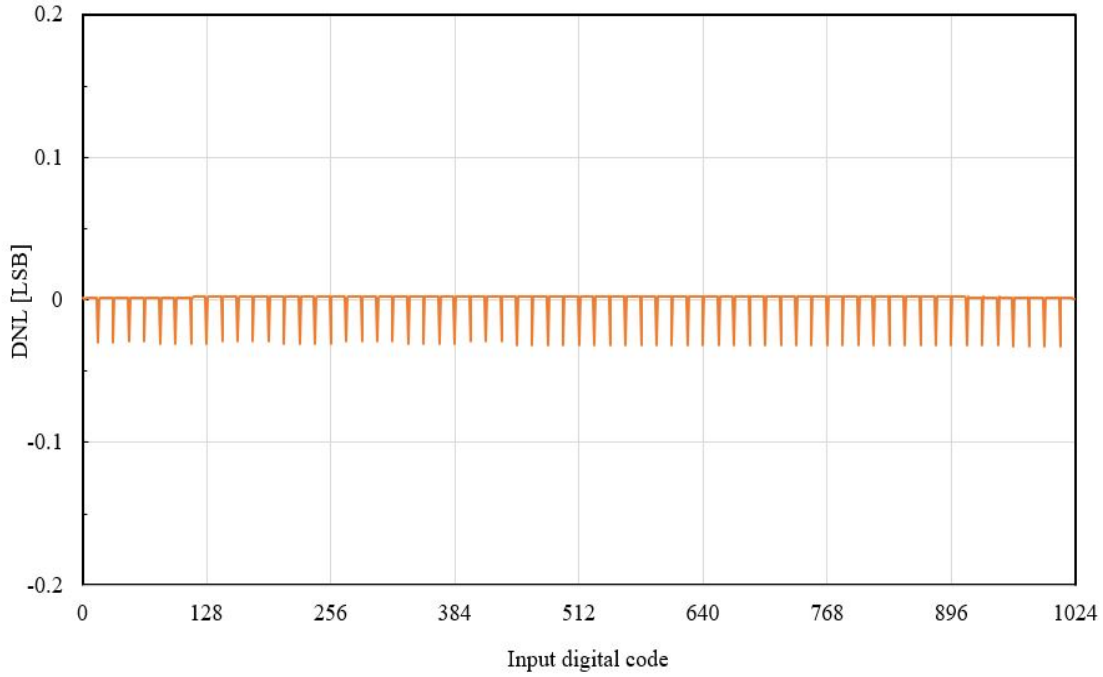

Figure 6.11.: Measured DNL profile of CS-DAC in 10 *bit* mode ($F_{IN} = 1MSps$)

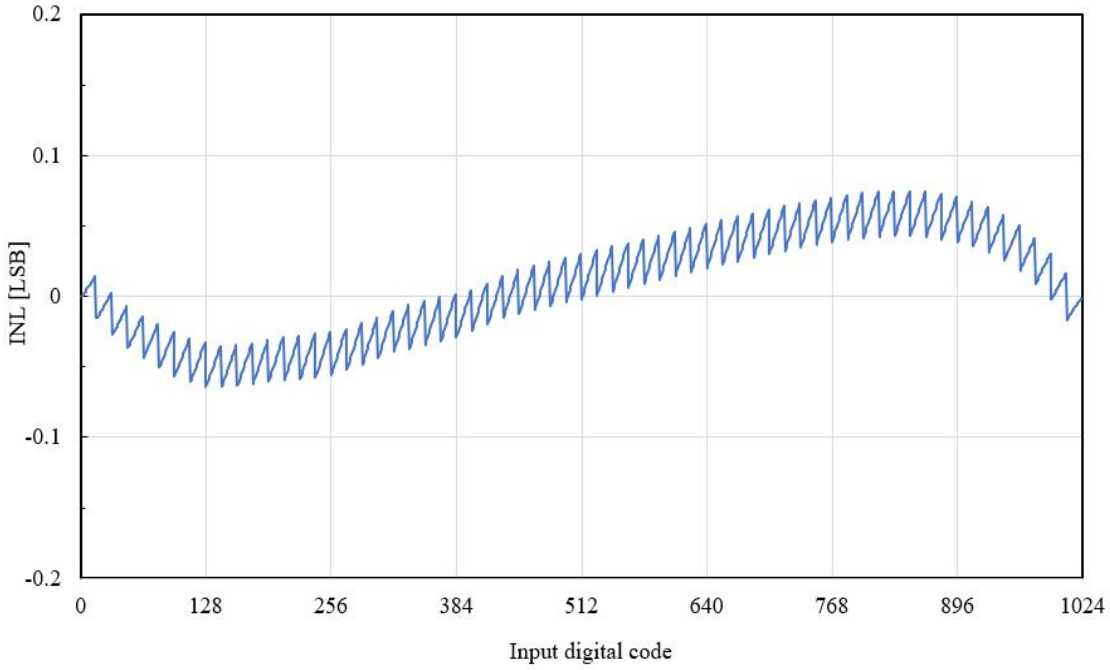

Figure 6.12.: Measured INL profile of CS-DAC in 10 *bit* mode ($F_{IN} = 1MSps$)

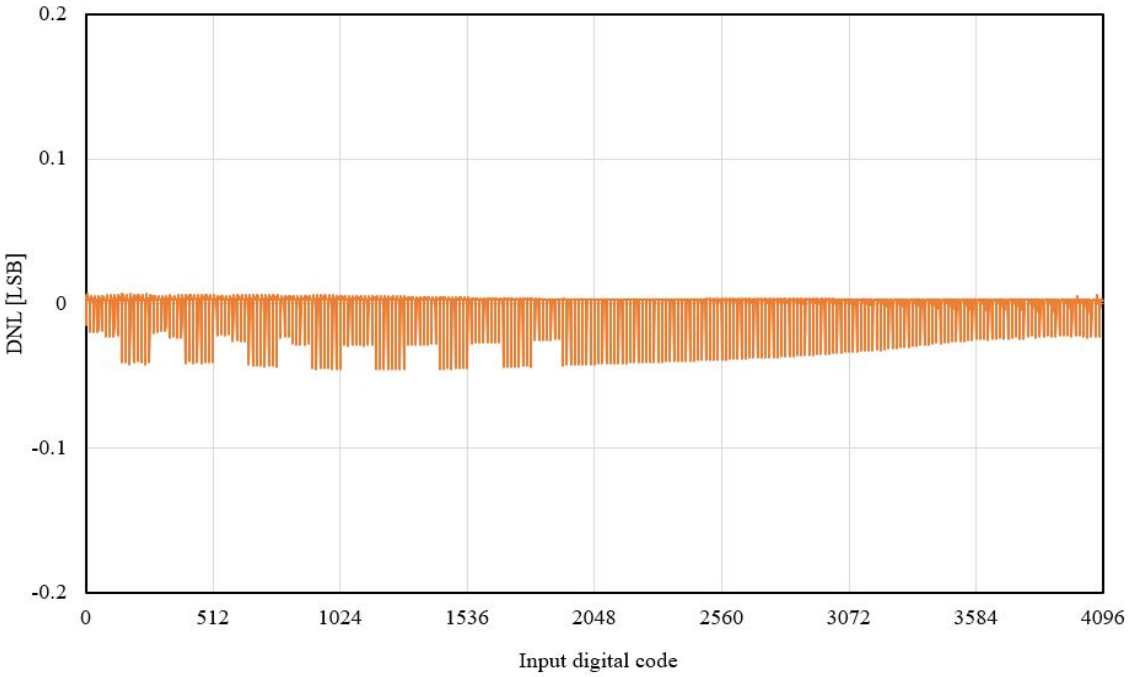

Figure 6.13.: Measured DNL profile of CS-DAC in 12 *bit* mode ($F_{IN} = 5MSps$)

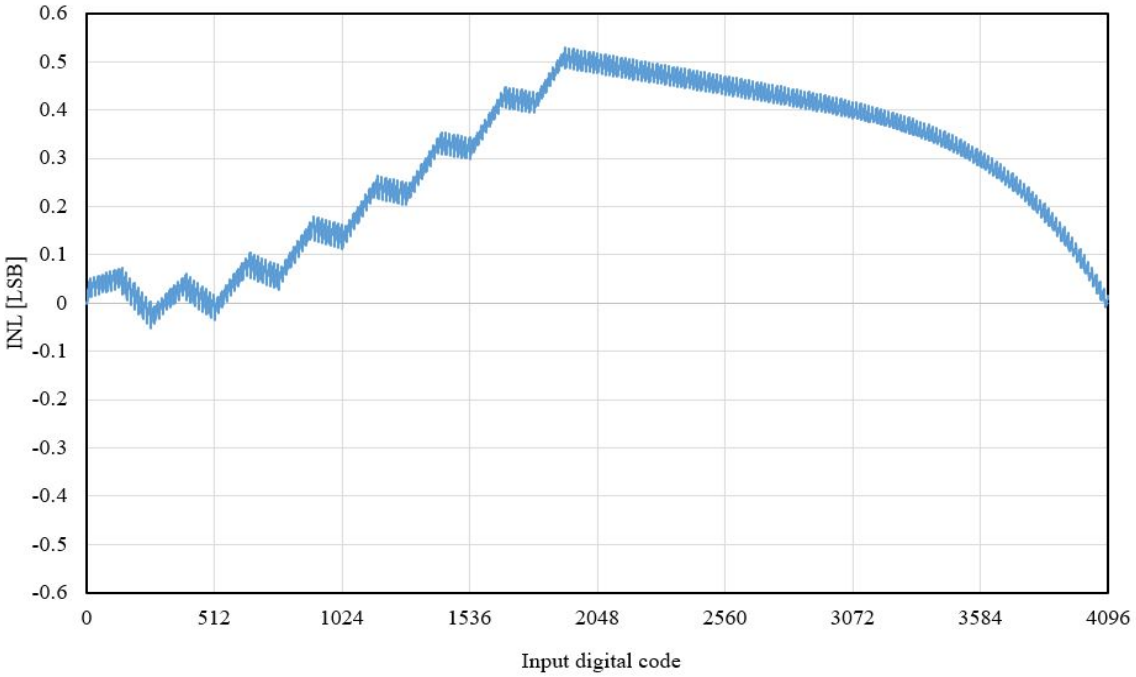

Figure 6.14.: Measured INL profile of CS-DAC in 12 *bit* mode ($F_{IN} = 5MSps$)

6.2.3. Dynamic Performance of Scalable CS-DAC

Dynamic performances like SFDR and ENOB are measured by setting a similar test bench to that of static measurements. But, in this case a sinusoidal digital sequence is required for CS-DAC in place of ramp signal. A coherent sampling technique is used to evaluate the output spectrum of a converter. According to this technique, input for ideal ADC is a sinusoidal wave of frequency :

$$F_{IN} = \frac{tone}{num\_points} \times Fclk_{ADC} \tag{6.4}$$

where, $Fclk_{ADC}$ is the sampling frequency of ADC, *tone* is a prime number so that F_{IN} and $Fclk_{ADC}$ are not harmonically related, and *num_points* is an integer of value 2^N (N is DAC's resolution).

To measure the SFDR of the CS-DAC in 8 *bit* mode, a sinusoidal wave of amplitude 400 mV and frequency (F_{IN}) 566.4 kHz is sampled at a rate of 10 $MSps$. The sinusoidal output samples from the CS-DAC are loaded into spectrum analyser to evaluate the SFDR and ENOB. Figure 6.15 shows the measured spectrum output for 8 *bit* mode using a rectangular window from the spectrum analyser. The figure 6.15 provides some information, which can be analysed for better understanding. The signal amplitude in terms of dB can be calculated as follows:

$$Amp_{dB} = 20log(Amp_V) \tag{6.5}$$

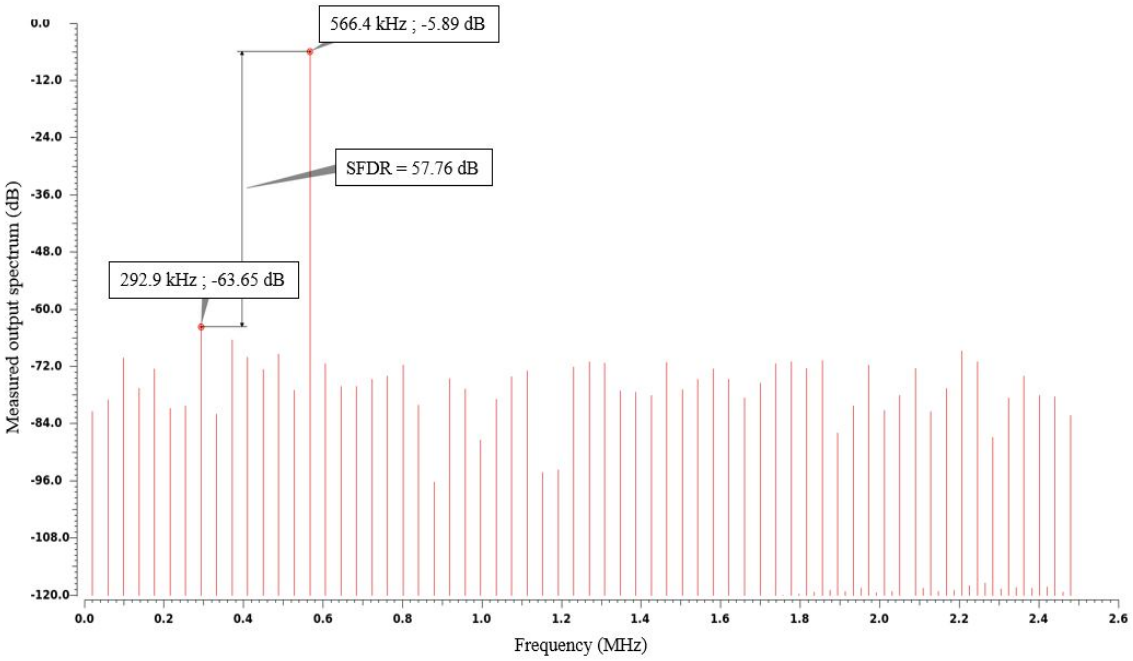

Figure 6.15.: Measured output spectrum of CS-DAC in 8 *bit* mode ($F_{IN} = 566.4KSps$)

The amplitude in volts of the output signal for the current in equation 6.1 and R_{LOAD} of 4K Ω is 508 mV. Using this value in equation 6.5 gives the power of output signal to be - 5.89 dB approximately. SFDR for the signal of frequency F_{IN}, can be calculated from the difference in amplitude between the fundamental and the largest harmonic spur in the window, that is :

$$SFDR_{8bit} = 63.65 - 5.89 = 57.76dB \tag{6.6}$$

Further, figures 6.16 and 6.17 shows the measured dynamic performances for the 10 *bit* and 12 *bit* modes respectively. The steps taken for analysing the output spectrum of 8 *bit* mode can be applied here too and the summary is given in table 7.1.

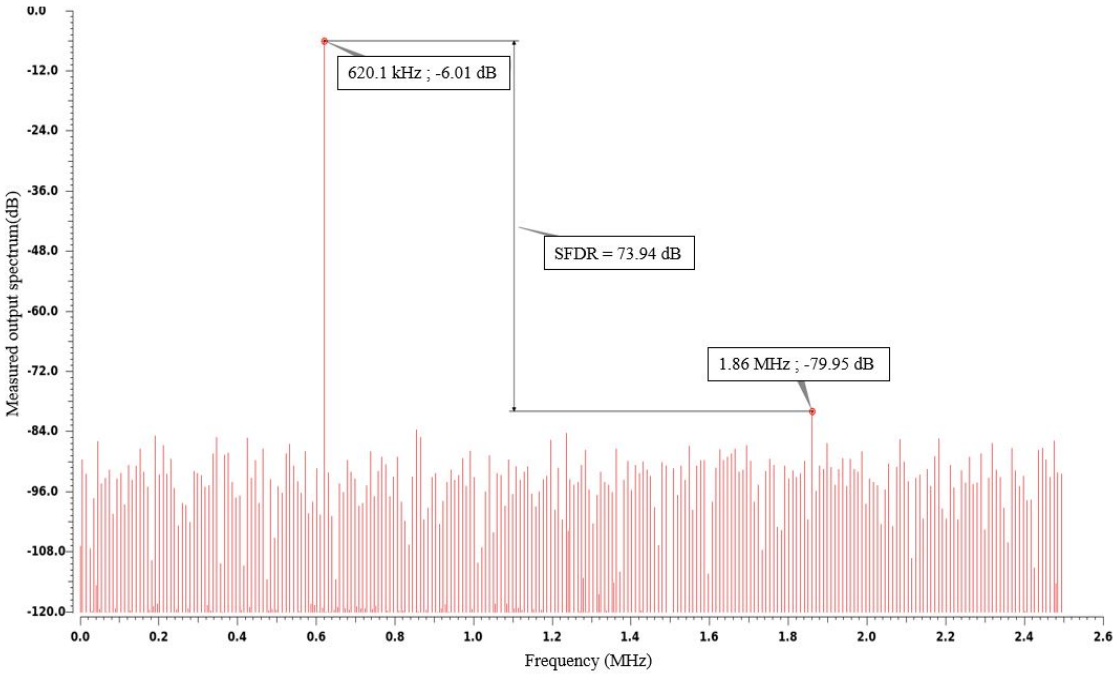

Figure 6.16.: Measured output spectrum of CS-DAC in 10 *bit* mode ($F_{IN} = 620.1KSps$)

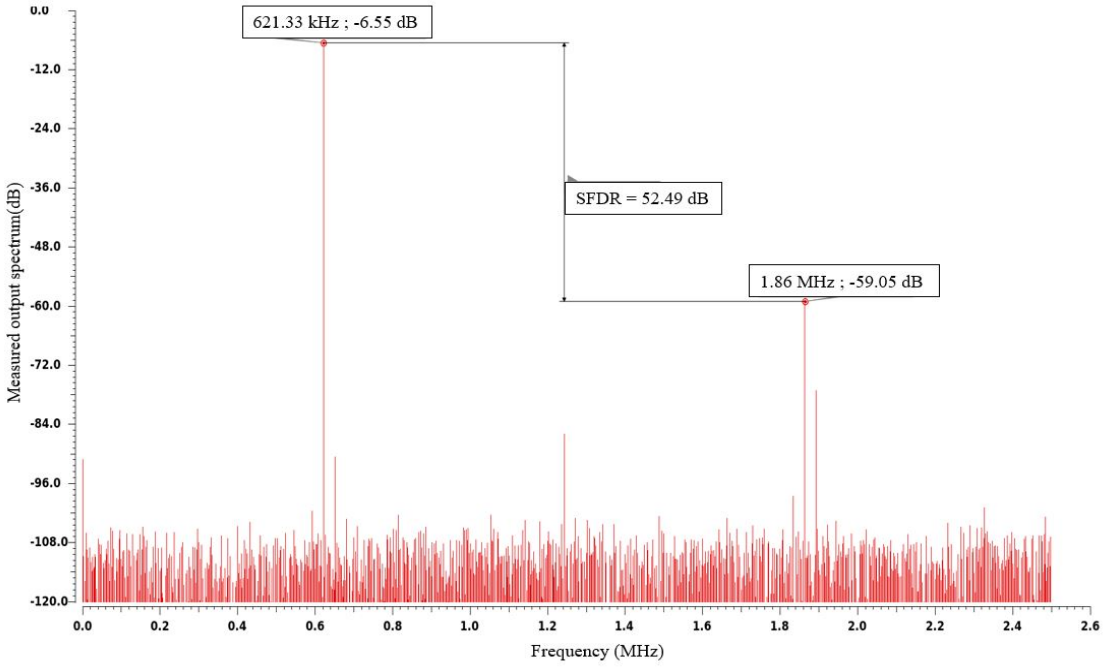

Figure 6.17.: Measured output spectrum of CS-DAC in 12 *bit* mode ($F_{IN} = 621.3KSps$)

7. Conclusion and Future Work

7.1. Conclusion

A comprehensive analysis and research of resolution scalable CS-DAC is presented throughout this thesis work. The recent design trends, converter's static and dynamic behaviours, and parameter dependencies are discussed in a detailed manner. A scalable CS-DAC with a conversion rate of 10 *MSps* and its associated control blocks were designed in transistor level using 22nm FDSOI technology for a supply voltage of 0.8V.

The proposed architecture consist of several sub blocks such as current sources, flip flops, decoders, clock tree and biasing circuit, all these blocks were designed in cadence environment. The architecture also includes a new mode selection decoder block, which is designed to control the corresponding blocks in CS-DAC to achieve resolution scalability. A detailed analysis was done on system level to come up with resolution scalability and the thesis discusses the steps taken to attain it. Further, the current sources were designed using statistical mismatch model and its accuracy was satisfactory. Low voltage cascoded current mirror topology was chosen for biasing purpose and it proved as a suitable choice for SLVT transistors by giving better output swing. The impact of PVT corners and statistical mismatch on the designed current sources were verified by MC simulations to promise better performance of the converter.

The implementation of resolution scalability in CS-DAC was justified with appropriate test bench and simulations. Further, the designed CS-DAC presents satisfactory static and dynamic performances as concluded in table 7.1. The SLVT transistors from the modern 22nm FDSOI technology reveals its excellent feasibility for each designed block. The complexities and non-idealities of the transistors increases with the scaling down of node and supply voltage. Despite of this, the modern 22 nm FDSOI technology emerges as a good choice for high performance design for ultra-low-power applications. The thesis work is concluded as, 'The implementation of ultra-low-power CS-DAC with scalable resolutions 8, 10 and 12 *bit* was achievable and the design can be used for future low power sensor applications'.

Table 7.1.: Outlook of the designed scalable CS-DAC

Parameters	8 *bit*	10 *bit*	12 *bit*
$Segmentation(U:B)$	4:4	6:4	8:4
$R_{LOAD}(\Omega)$	4K	1K	250
$I_{FS}(A)$	127 μ	501 μ	1.88 m
$V_{OUT\_diff}(V)$	1.016	1.002	0.94
$Power(mW)$	0.214	0.513	2.407
$DNL(LSB)$	0.06	0.04	0.05
$INL(LSB)$	0.04	0.06	0.53
$Rate(MSps)$	10	10	10
$SFDR$ @ F_{IN}	57.76 dB @ 566 KHz	73.94 dB @ 620 KHz	52.49 dB @ 621 KHz

7.2. Future Works

The steps taken for achieving resolution scalability was successful and justified. Since this is the first demonstration of designing a scalable DAC, there could other approaches or ideas in terms of reduced chip area. The presented thesis work focuses only on schematic design, but some considerations with respect to layout are given below.

- In this work, a general sequential switching scheme is used. This would introduce some linear and/or symmetrical gradient errors in layout. Several switching schemes, to compensate these errors are proposed in literature and a suitable one can be adapted.
- As explained in section 6.1.3, the current sources in CSA can be grouped and separated in four quadrants. This would provide symmetry and additional advantage of lowering the influence of distance matching effects in layout.
- In biasing circuit (section 5.2.2), the global biasing can be realised using a common-centroid layout pattern to reduce effects of gradients. The local biasing can be separated into four quadrants, this will improve the static performance of the CS-DAC [31].
- The CSA is sensitive to digital coupling through the substrate [4], hence placing a guard ring around CSA would avoid this coupling .
- Different power supply lines are used in schematic for the analog and the digital blocks, same has to be followed during layout.

A. Appendix

A.1. Verilog-A Code

A.1.1. 4-15 bit Binary-to-Thermometer Decoder

```
// VerilogA for 4 bit Binary-to-thermometer decoder

`include "constants.vams"
`include "disciplines.vams"
`define THEMO_BITS      15


module Bin2Ther_4bit( B, thermo);
input [0:3] B;
output [0:14] thermo;
electrical [0:3] B;
electrical [0:14] thermo;

parameter real vlogic_high = 0.8;
parameter real vlogic_low = 0;
parameter real vtrans = 0.4;
parameter real tdel = 0 from [0:inf);
parameter real trise = 500p from (0:inf);
parameter real tfall = 500p from (0:inf);
real Varthermo[0:`THEMO_BITS-1];
integer i;
genvar n;

analog begin
 @ ( initial_step ) begin
for (i = 0; i <= `THEMO_BITS-1 ; i = i + 1) begin
        Varthermo[i] = vlogic_low;
end
end
@ ((cross(V(B[0]) -vtrans,0)) or (cross(V(B[1]) -vtrans,0)) or
(cross ( V(B[2]) -vtrans , 0)) or (cross ( V(B[3]) -vtrans , 0)))
    begin
    if(V(B[3]) < vtrans && V(B[2]) < vtrans && V(B[1])
     < vtrans && V(B[0]) < vtrans) begin
        for (i = 0; i <= `THEMO_BITS-1 ; i = i + 1) begin
```

```
36          Varthermo[i] = vlogic_low;
37        end
38        end
39     else if(V(B[3]) < vtrans && V(B[2]) < vtrans && V(B[1])
40      < vtrans && V(B[0]) > vtrans) begin
41        for (i = 0; i < 1 ; i = i + 1) begin
42           Varthermo[i] = vlogic_high;
43        end
44        for (i = 1; i <= `THEMO_BITS-1 ; i = i + 1) begin
45           Varthermo[i] = vlogic_low;
46        end
47        end
48      else if(V(B[3]) < vtrans && V(B[2]) < vtrans && V(B[1])
49       > vtrans && V(B[0]) < vtrans)  begin
50        for (i = 0; i < 2 ; i = i + 1) begin
51           Varthermo[i] = vlogic_high;
52        end
53        for (i = 2; i <= `THEMO_BITS-1 ; i = i + 1) begin
54           Varthermo[i] = vlogic_low;
55        end
56        end
57       else if(V(B[3]) < vtrans && V(B[2]) < vtrans && V(B[1])
58        > vtrans && V(B[0]) > vtrans)  begin
59        for (i = 0; i < 3 ; i = i + 1) begin
60                Varthermo[i] = vlogic_high;
61        end
62        for (i = 3; i <= `THEMO_BITS-1 ; i = i + 1) begin
63                Varthermo[i] = vlogic_low;
64        end
65        end
66       else if(V(B[3]) < vtrans && V(B[2]) > vtrans && V(B[1])
67        < vtrans && V(B[0]) < vtrans)  begin
68        for (i = 0; i < 4 ; i = i + 1) begin
69                Varthermo[i] = vlogic_high;
70        end
71        for (i = 4; i <= `THEMO_BITS-1 ; i = i + 1) begin
72                Varthermo[i] = vlogic_low;
73        end
74        end
75       else if(V(B[3]) < vtrans && V(B[2]) > vtrans && V(B[1])
76        < vtrans && V(B[0]) > vtrans)  begin
77        for (i = 0; i < 5 ; i = i + 1) begin
78                Varthermo[i] = vlogic_high;
79        end
80        for (i = 5; i <= `THEMO_BITS-1 ; i = i + 1) begin
81                Varthermo[i] = vlogic_low;
82        end
83        end
```

```
84          else if(V(B[3]) < vtrans && V(B[2]) > vtrans && V(B[1])
85            > vtrans && V(B[0]) < vtrans)  begin
86            for (i = 0; i < 6 ; i = i + 1) begin
87                    Varthermo[i] = vlogic_high;
88            end
89            for (i = 6; i <= `THEMO_BITS-1 ; i = i + 1) begin
90                    Varthermo[i] = vlogic_low;
91            end
92            end
93          else if(V(B[3]) < vtrans && V(B[2]) > vtrans && V(B[1])
94           > vtrans && V(B[0]) > vtrans)  begin
95            for (i = 0; i < 7 ; i = i + 1) begin
96                    Varthermo[i] = vlogic_high;
97            end
98            for (i = 7; i <= `THEMO_BITS-1 ; i = i + 1) begin
99                    Varthermo[i] = vlogic_low;
100           end
101           end
102         else if(V(B[3]) > vtrans && V(B[2]) < vtrans && V(B[1])
103         < vtrans && V(B[0]) < vtrans)  begin
104           for (i = 0; i < 8 ; i = i + 1) begin
105                   Varthermo[i] = vlogic_high;
106           end
107           for (i = 8; i <= `THEMO_BITS-1 ; i = i + 1) begin
108                   Varthermo[i] = vlogic_low;
109           end
110           end
111         else if(V(B[3]) > vtrans && V(B[2]) < vtrans && V(B[1])
112           < vtrans && V(B[0]) > vtrans)  begin
113           for (i = 0; i < 9 ; i = i + 1) begin
114                   Varthermo[i] = vlogic_high;
115           end
116           for (i = 9; i <= `THEMO_BITS-1 ; i = i + 1) begin
117                   Varthermo[i] = vlogic_low;
118           end
119           end
120         else if(V(B[3]) > vtrans && V(B[2]) < vtrans && V(B[1])
121           > vtrans && V(B[0]) < vtrans)  begin
122           for (i = 0; i < 10 ; i = i + 1) begin
123                   Varthermo[i] = vlogic_high;
124           end
125           for (i = 10; i <= `THEMO_BITS-1 ; i = i + 1) begin
126                   Varthermo[i] = vlogic_low;
127           end
128           end
129         else if(V(B[3]) > vtrans && V(B[2]) < vtrans && V(B[1])
130           > vtrans && V(B[0]) > vtrans)  begin
131           for (i = 0; i < 11 ; i = i + 1) begin
```

```
132                 Varthermo[i] = vlogic_high;
133           end
134           for (i = 11; i <= `THEMO_BITS-1 ; i = i + 1) begin
135                 Varthermo[i] = vlogic_low;
136           end
137           end
138         else if(V(B[3]) > vtrans && V(B[2]) > vtrans && V(B[1])
139           < vtrans && V(B[0]) < vtrans)  begin
140           for (i = 0; i < 12 ; i = i + 1) begin
141                 Varthermo[i] = vlogic_high;
142           end
143           for (i = 12; i <= `THEMO_BITS-1 ; i = i + 1) begin
144                 Varthermo[i] = vlogic_low;
145           end
146           end
147         else if(V(B[3]) > vtrans && V(B[2]) > vtrans && V(B[1])
148           < vtrans && V(B[0]) > vtrans)  begin
149           for (i = 0; i < 13 ; i = i + 1) begin
150                 Varthermo[i] = vlogic_high;
151           end
152           for (i = 13; i <= `THEMO_BITS-1 ; i = i + 1) begin
153                 Varthermo[i] = vlogic_low;
154           end
155           end
156        else if(V(B[3]) > vtrans && V(B[2]) > vtrans && V(B[1])
157          > vtrans && V(B[0]) < vtrans)  begin
158           for (i = 0; i < 14 ; i = i + 1) begin
159                 Varthermo[i] = vlogic_high;
160           end
161           for (i = 14; i <= `THEMO_BITS-1 ; i = i + 1) begin
162                 Varthermo[i] = vlogic_low;
163           end
164           end
165       else if(V(B[3]) > vtrans && V(B[2]) > vtrans && V(B[1])
166         > vtrans && V(B[0]) > vtrans)  begin
167           for (i = 0; i < 15 ; i = i + 1) begin
168                 Varthermo[i] = vlogic_high;
169           end
170           for (i = 15; i <= `THEMO_BITS-1 ; i = i + 1) begin
171                 Varthermo[i] = vlogic_low;
172           end
173           end
174       end
175           for (n = 0; n <= `THEMO_BITS-1 ; n = n + 1) begin
176           V(thermo[n]) <+ transition( Varthermo[n], tdel, trise, tfall);
177           end
178       end
179       endmodule
```

A.1.2. Ideal 12 bit ADC

```
// VerilogA for 12 bit analog to digital converter
`include "discipline.h"
`include "constants.h"

module adc_12bit(vd11, vd10, vd9, vd8, vd7,vd6,vd5,vd4,vd3,vd2,vd1,
vd0,vin,vclk);
electrical vd11, vd10, vd9, vd8, vd7, vd6, vd5, vd4, vd3, vd2, vd1,
vd0, vin, vclk;
parameter real trise = 0 from [0:inf);
parameter real tfall = 0 from [0:inf);
parameter real tdel = 0 from [0:inf);
parameter real mismatch_fact=0 from [0:inf);
parameter real vlogic_high = 5;
parameter real vlogic_low  = 0;
parameter real vtrans_clk     = 2.5;
parameter real vref        = 1.0;


`define NUM_ADC_BITS    12
`define MAXINT  2_147_483_647.0


`define FRAC_MM(I)(1.0 + mismatch_fact*(dist_range*abs($random(I)/`MAXINT)- \
                    half_dist_range))

   real dist_range, half_dist_range;
   real comp_var[0:`NUM_ADC_BITS-1];
   real unconverted;
   real halfref;
   real comp_vref;
   real vd[0:`NUM_ADC_BITS-1];
   integer i;
   integer iseed;

   analog begin
      @ ( initial_step ) begin
         dist_range = 0.02;
         half_dist_range = 0.01;
         generate j ( 0, `NUM_ADC_BITS-1 ) begin
           iseed = j;
           comp_var[j] = `FRAC_MM(iseed);
         end
         halfref = vref / 2;
      end

      @ (cross(V(vclk) - vtrans_clk, 1, 1.0, vclk.potential.abstol)) begin
         unconverted = V(vin);
```

```
48          for (i = `NUM_ADC_BITS-1; i >= 0 ; i = i - 1) begin
49             vd[i] = 0;
50             comp_vref = halfref * comp_var[i];
51             if (unconverted > comp_vref) begin
52                vd[i] = vlogic_high;
53                unconverted = unconverted - comp_vref;
54             end else begin
55                vd[i] = vlogic_low;
56             end
57             unconverted = unconverted * 2;
58          end
59       end
60
61       V(vd11) <+ transition( vd[11], tdel, trise, tfall );
62       V(vd10) <+ transition( vd[10], tdel, trise, tfall );
63       V(vd9) <+ transition( vd[9], tdel, trise, tfall );
64       V(vd8) <+ transition( vd[8], tdel, trise, tfall );
65       V(vd7) <+ transition( vd[7], tdel, trise, tfall );
66       V(vd6) <+ transition( vd[6], tdel, trise, tfall );
67       V(vd5) <+ transition( vd[5], tdel, trise, tfall );
68       V(vd4) <+ transition( vd[4], tdel, trise, tfall );
69       V(vd3) <+ transition( vd[3], tdel, trise, tfall );
70       V(vd2) <+ transition( vd[2], tdel, trise, tfall );
71       V(vd1) <+ transition( vd[1], tdel, trise, tfall );
72       V(vd0) <+ transition( vd[0], tdel, trise, tfall );
73
74 `undef NUM_ADC_BITS
75    end
76 endmodule
```

The Verilog-A code for 10 and 8*bit* ideal ADC are implemented by simple modifications in the above code.

A.2. Cadence Schematics

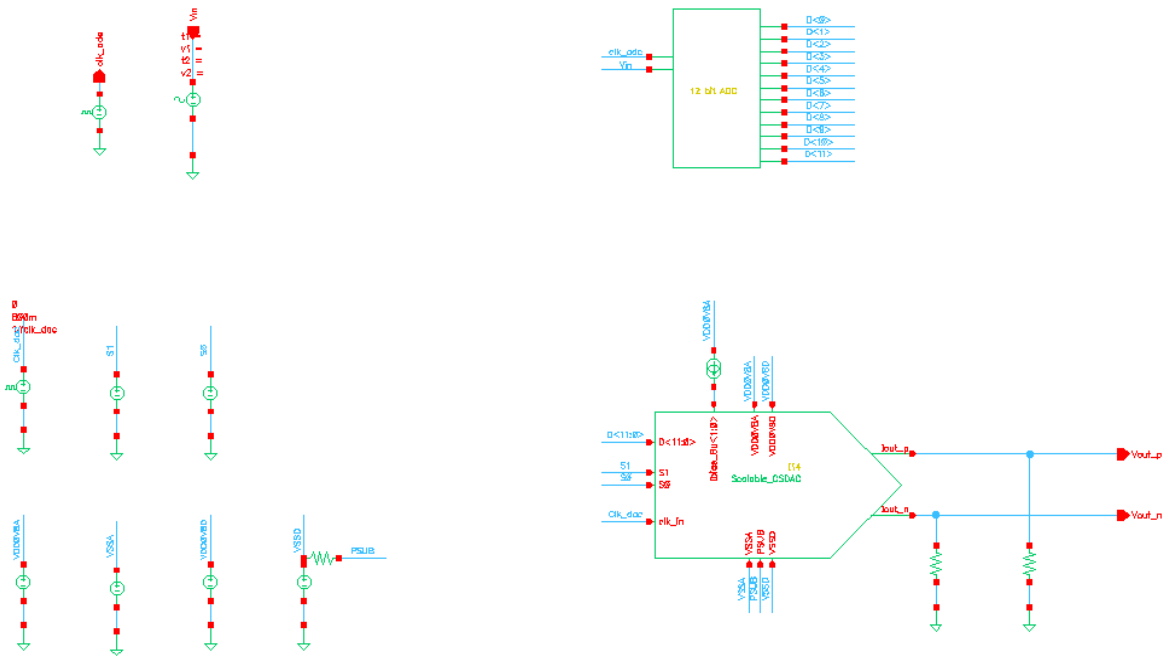

Figure A.1.: Test bench for designed scalable CS-DAC

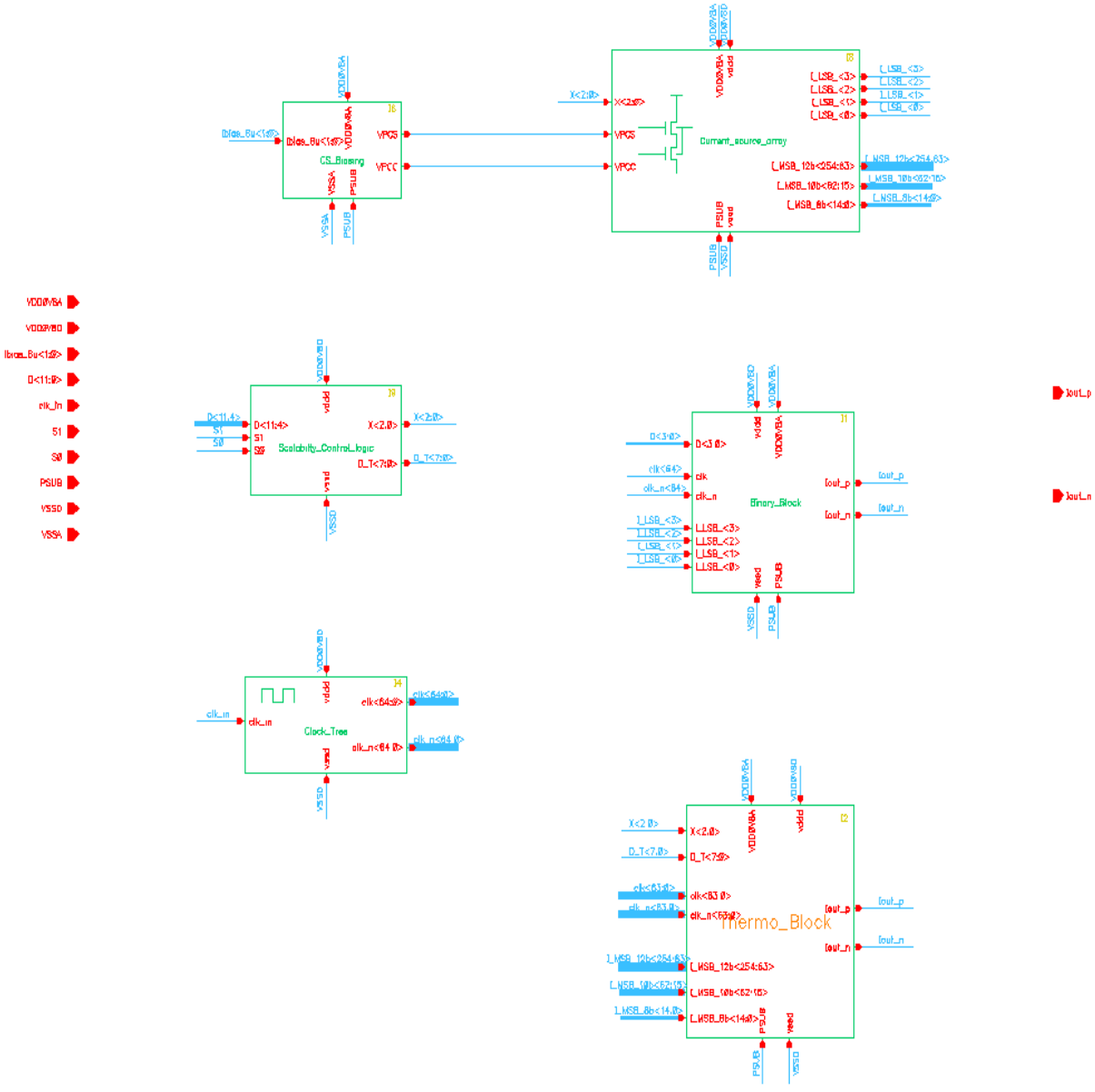

Figure A.2.: Top level of CS-DAC

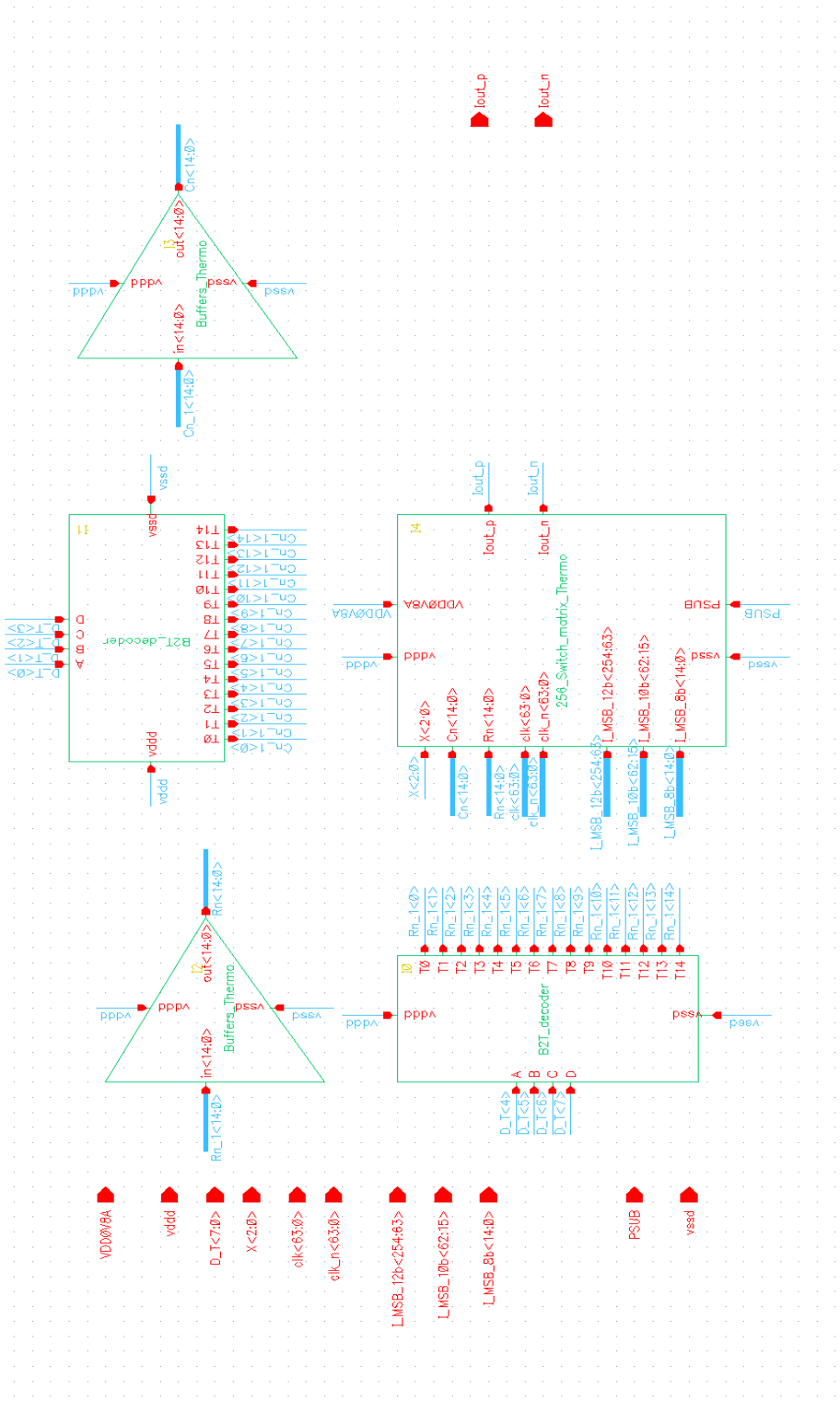

Figure A.3.: Scalable Thermometer Block

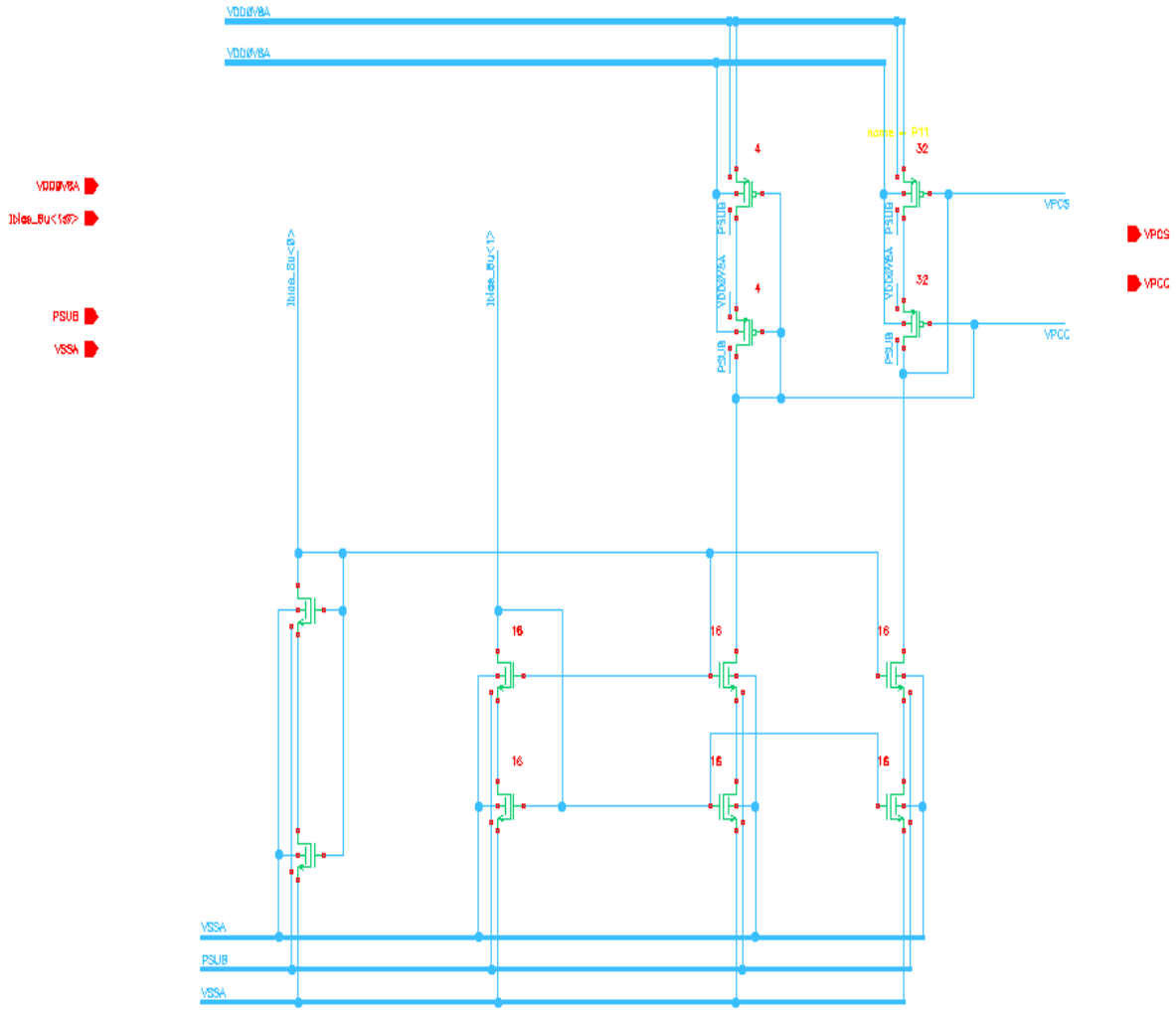

Figure A.4.: Current mirror biasing circuit

Bibliography

[1] Peter Kinget. "The World Is Analog". In: Circuit Cellar Staff (2014). http://www.ee.columbia.edu/~kinget/WhyAnalog/circuitcellar_The_World_Is_Analog_201410.pdf (Last retrieved on 08.02.2020).

[2] Bill McCulley. *Bridging the Divide: A DAC Applications Tutorial*. Tech. rep. http://www.ti.com/lit/an/snaa129/snaa129.pdf (Last retrieved on 08.02.2020).

[3] Tarek H. Zaki. *A Compact 12-bit Current-Steering D/A Converter for HDRCR Camera Systems*. 2010.

[4] A.V. den Bosch, Michiel Steyaert, and W. Sansen. *Static and dynamic performance limitations for high speed D/A converters*. 2004. DOI: 10.1007/978-1-4757-6579-3.

[5] J Jacob Wikner. *Studies on CMOS Digital-to-Analog Converters*. http://cc.ee.nchu.edu.tw/~aiclab/public_htm/DAC/Theses/2001Wikner.pdf (Last retrieved on 08.02.2020).

[6] Günther Langmann. *Low Power 10 bit DAC Design for Video-Encoder*. https://diglib.tugraz.at/download.php?id=576a78d110f1f&location=browse (Last retrieved on 08.02.2020).

[7] I. Myderrizi and A. Zeki. "Current-Steering D/A Converters: Functional specifications, Design Basics, and Behavioral Modelling". In: IEEE Antennas and Propagation Magazine, Vol. 52, No. 4 (2010). DOI: 10.1109/MAP.2010.5638288.

[8] R. Jacob Baker. *CMOS Circuit Design, Layout, and Simulation*. IEEE Press Series on Microelectronic Systems. 2010. ISBN: 978-0-470-88132-3.

[9] Anne Van den Bosch, A. F. Borremans, and Michel S. J. Steyaert. "A 10-bit 1-GSample/s Nyquist Current-Steering CMOS D/A Converter". In: IEEE Journal of Solid-State Circuits, vol. 36, No. 3. 2001. DOI: S0018-9200(01)01478-0.

[10] Chi-Hung Lin and Klaas Bult. "A 10-b, 500-MSamples/s CMOS DAC in 0.6 mm^2". In: IEEE Journal of Solid-State Circuits, Vol. 33, No. 12. 1998. DOI: 10.1109/4.735535.

[11] Kevin O'Sullivan, Michael Hennessy, and Vincent Callaghan. "A 12-bit 320-MSample/s Current-Steering CMOS D/A Converter in 0.44 mm^2". In: IEEE Journal of Solid-State Circuits, Vol. 39, No. 7. 2004. DOI: 10.1109/JSSC.2004.829923.

[12] Takeshi Ueno, Takafumi Yamaji, and Tetsuro Itakura. "A 1.2-V, 12-bit, 200 MSample/s Current-Steering D/A Converter in 90-nm CMOS". In: IEEE Custom Integrated Circuits Conference. 2005. DOI: 10.1109/CICC.2005.1568776.

[13] Jurgen Deveugele and Michiel S. J. Steyaert. "A 10-bit 250-MS/s Binary-Weighted Current-Steering DAC". In: IEEE Journal of Solid-State Circuits, Vol. 41, No. 2. 2006. DOI: 10.1109/JSSC.2005.862342.

[14] M. Borremans et al. "A Low Power, 10-bit CMOS D/A Converter for High Speed Applications". In: IEEE Custom Integrated Circuits Conference. 2001. DOI: 10.1109/CICC.2001.929746.

[15] Dominik Przyborowski and Marek Idzik. "A 10-bit Low-Power Small-Area High-Swing CMOS DAC". In: IEEE Transactions on Nuclear Science, Vol. 57, No. 1, 2010. DOI: 10.1109/TNS.2009.2038054.

[16] Hugo Hernandez, W.V. Noije, and Elkim Roa. "Design strategy of current source in Current-steering CMOS DAC". In: https://pdfs.semanticscholar.org/2352/36e004d9b5d28a6afdbfecee6aec1e6cb573.pdf (Last retrieved on 08.02.2020).

[17] G. Bertotti, A. Laifi, and E. Di Gioia. "An 8 bit current steering DAC for offset compensation purposes in sensor arrays". In: Advances in Radio Science, 10, 201–206. 2012. DOI: 1:10.5194/ars-10-201-2012.

[18] Ankush Shingade and Harshada Gadge andRabinder Henry. "Design and verification of Current steering segmented Digital to Analog Converter". In: Global Journal of Trends in Engineering,Vol(2)-Issue(4). https://www.researchgate.net/publication/275302756_Design_and_verification_of_Current_steering_segmented_Digital_to_Analog_Converter (Last retrieved on 08.02.2020).

[19] M. Pelgrom, A. Duinmaijer, and A. Welbers. "Matching Properties of MOS Transistors". In: IEEE Journal of Solid-State Circuits, vol. SC-24, pp. 1433-1439. 1989. DOI: 10.1109/JSSC.1989.572629.

[20] Moore G E. "Cramming more components onto integrated circuits." In: Electronics (1965). https://www.cs.csub.edu/~melissa/cs350-f15/notes/Doc/gordon_moore_1965_article.pdf (Last retrieved on 08.02.2020).

[21] Kangguo Cheng and Ali Khakifirooz. *Fully depleted SOI (FDSOI) technology*. Tech. rep. 2016.

[22] Mohammad Ashraful Alam. *Incremental Delta-Sigma ADCs in 22nm-FDX Technology*. https://www.researchgate.net/publication/320596980_Incremental_Delta-Sigma_S_ADCs_in_22nm-FDX_Technology (Last retrieved on 08.02.2020).

[23] *Learn More About FD-SOI*. https://www.st.com/content/st_com/en/about/innovation---technology/FD-SOI/learn-more-about-fd-soi.html (Last retrieved on 08.02.2020).

[24] Ramya Srinivasan and Tamer Ragheb. "Body-biasing scaling for GLOBALFOUNDRIES 22FDx". In: New Dimension to Explore the Design. https://www.globalfoundries.com/sites/default/files/articles/body-bias-scaling-for-globalfoundries-22fdx-technology-new-dimension-to-explore-the-design.pdf (Last retrieved on 08.02.2020).

[25] Don Blackwell and Timothy Miller. "Analog Design Workshop for 22FDX 22nm FD-SOI Technology part I". In: GLOBALFOUNDRIES webinar. https://www.globalfoundries.com/resources/technical-webinar-series/analog-design-workshop-22fdx-22nm-fd-soi-technology-part-one (Last retrieved on 08.02.2020).

[26] G. Radulov and P. Quinn. *Smart and Flexible Digital-to-Analog Converters*. Analog Circuits and Signal Processing, Springer. 2010. DOI: 10.1007/978-94-007-0347-6.

[27] Miquel Albiol, José Luis González, and Eduard Alarcón. "Mismatch and Dynamic Modeling of Current Sources in Current-Steering CMOS D/A Converters: An Extended Design Procedure". In: IEEE Transactions on Circuits And Systems—I: Regular Papers, Vol. 51, No. 1. 2004. DOI: 10.1109/TCSI.2003.821287.

[28] Timothy S. Miller. "22FDX Analog Design Best Practices and Guidelines". In: Fraunhofer Workshop Erlangen, Global Foundries. 2018.

[29] Takahiro Miki et al. "An 80-MHz 8-bit CMOS D/A Converter". In: IEEE Journal of Solid-State Circuits, vol. Sc-21, No. 6. 1986. DOI: 10.1109/JSSC.1986.1052639.

[30] J.H. Tucker and M.A. Tapia. "Using Karnaugh maps to solve Boolean equations". In: Proceedings IEEE Southeastcon '92. 1992. DOI: 10.1109/SECON.1992.202260.

[31] Peiman Aliparast et al. "A 12-Bit 1-Gsample/s Nyquist Current-Steering DAC for Wireless Transmitter". In: Circuits and Systems, 2011, 2, 74-84. 2011. DOI: 10.4236/cs.2011.22012.