A 10-bit Linear Interpolation DAC with Tree-structured DEM for WiFi 6 Application

WENXUAN ZENG
MASTER´S THESIS
DEPARTMENT OF ELECTRICAL AND INFORMATION TECHNOLOGY
FACULTY OF ENGINEERING | LTH | LUND UNIVERSITY

A 10-bit Linear Interpolation DAC with Tree-structured DEM for WiFi 6 Application

Wenxuan Zeng
we1871ze-s@student.lu.se

Department of Electrical and Information Technology
Lund University

Supervisor: Nikola Ivanisevic, Nordic Semiconductor
Iman Ghotbi, Lund University

Examiner: Pietro Andreani

September 3, 2025

Printed in Sweden
Tryckeriet i E-huset, Lund

Acknowledgement

First of all, I would like to express my heartfelt thanks to my supervisor, Dr. Ivanisevic, and my manager, Mats Erixon, at Nordic Semiconductor, for their guidance, patience, encouragement, and kindness throughout this journey. Dr.Ivanisevic, with his expertise and patience, has provided invaluable insights and countless guidance to this project, for which I am deeply grateful. I would also like to thank all my colleagues at Nordic Semiconductor for their help and kindness. And a special thanks goes to my supervisor at Lund University, Dr.Ghotbi, who has provided me with invaluable help and guidance during my studies there.

My deepest gratitude goes to my family. To my parents, who have worked so hard and made countless sacrifices to give me a better life. Their unwavering support has allowed me to see the world I have always longed for. To my sister, who has always listened to my complaints and encouraged me when I was down. They have sacrificed so much, and now it is my turn to take on more responsibility.

Finally, I would like to thank the friends I met along the way, for their help and the joy they have brought into my life.

Abstract

This thesis focuses on two primary parts: the design of a 10-bit medium high-speed, high-linearity current-steering digital-to-analog converter (DAC), and the implementation of a linear analog interpolation technique aimed at suppressing images at the DAC output.

Design considerations for the current-steering DAC are discussed comprehensively, and the circuit is designed in 22nm Fully-Depleted Silicon-On-Insulator (FDSOI) CMOS technology. A tree-structured Dynamic Element Matching (DEM) scheme is employed to mitigate nonlinearity arising from current source mismatch. The principles and circuit implementation of DEM are analyzed in detail.

To replace the conventional low-pass filter (LPF) typically used after the DAC in transmitters, a four-fold analog linear interpolation technique in conjunction with an RC filter is proposed. This approach effectively attenuates images to meet system requirements. The theory and feasibility of linear interpolation are thoroughly discussed.

At a sampling rate of 128MHz and an input signal frequency of 10MHz, the interpolated DAC achieves 800mV peak-to-peak output swing, an Effective Number of Bits (ENOB) of 9.63 bits, a Spurious-Free Dynamic Range (SFDR) of 72.7dB, and a maximum image level of –48.7dBc under typical process corner, at $50°C$ and 0.83V supply voltage. The DAC core consumes an average current of 2.13mA.

Popular Science Summary

Transceiver is one of the most essential components in wireless communication system, responsible for receiving and transmitting communication signals. On the transmitter side, digital-to-RF signal path composes of a large number of analog blocks. The DAC converts the baseband digital signal into an analog signal. And the DAC is followed by LPF that supresses unwanted spectral images at DAC output. The filtered in-phase (I) and quadrature (Q) signals are then upconverted to the IF or RF band through mixing with a local oscillator signal.

Large number of analog blocks in conventional transmitter add various source of mismatch and distortion. And they occupy large silicon area, which doesn't decrease greatly as CMOS technology scales down. Transmitter structure with higher integration-level and lower cost is desirable.

An intuitive idea is that we can combine the functionality of different analog blocks together while maintain the same performance. The complexity of the transmitter can be lower and higher integration can be achieved. An example that utilizes this idea is RFDAC, which integrates DAC and mixer together and dismiss the LPF[1][2]. Digital signal can be converted to RF band directly with only RFDAC. In term of the system simplicity and flexibility, RFDAC is a promising choice for modern communication system such as 5G and 5G-Advanced. However, the Nyquist frequency of RFDAC are required to exceed the RF band to avoid aliasing. It would add huge pressure on the design of DAC and consume high power.

In WiFi IoT applications, power consumption is a key consideration for customers. The additional design complexity and high power requirements of RFDAC make it less desirable for this scenario. Instead, a high integration-level solution with low power consumption is more practical and better suited for WiFi IoT application. This thesis presents such a solution without compromising performance.

Table of Contents

List of Figures

List of Tables

Acronyms

ADC: Analog to Digital Converter.

DAC: Digital to Analog Converter.

ENOB: Effective Number of Bits.

SFDR: Spurious-Free Dynamic Range.

SNDR: Signal to Noise and Distortion Ratio.

DNL: Differential Non-linearity.

INL: Integral Non-linearity.

MSB: Most Significant Bits.

LSB: Least Significant Bits.

LPF: Low Pass Filter.

DEM: Dynamic Element Matching.

Chapter 1

Introduction

1.1 Background

DAC is a key component in communication systems, serving as the bridge between the digital and analog domains. It converts digital signals into analog signal for transmission in the analog world. Typically, a DAC performs two fundamental operations: conversion and holding. In the conversion phase, the DAC converts discrete digital codes into corresponding analog values; in the holding phase, it hold the analog value for one sampling period.

During the conversion process, errors can be introduced and cause the analog output to deviate from its ideal value. These errors can originate from various circuit blocks within the DAC and induce static nonlinearity. In the holding operation, the inherent zero-order hold behavior introduces spectral images, which needs to be filtered out using a low-pass filter. Notably, the power and area consumption of the LPF can be comparable to those of the DAC itself. Additionally, imperfections during the holding phase such as glitches can lead to dynamic nonlinearity, which becomes increasingly problematic at higher speed.

While high-speed DACs exceeding Gsps are becoming popular in 5G and 5G-Advanced systems, WiFi IoT applications prioritize cost-efficiency and low power consumption over extreme speed. Medium high-speed DACs that offer a balance between performance, power, and integration are more suitable in this application. Thus, optimizing DAC design and the transmitter chain by extension to reduce cost and power while maintaining performance is a valuable area for investigation.

Common methods to reduce static nonlinearity include calibration, DEM, or increasing the area allocated to current sources for better matching. Additionally, selecting an appropriate topology for the current cell and driver circuit can significantly enhance dynamic linearity. Regarding power and area, replacing the traditional analog low-pass filter with a more efficient technique could offer promising improvements in overall system efficiency.

1.2 Basic of DAC

1.2.1 DAC structure

DAC can be broadly classified into Nyquist-rate DACs and oversampling DACs based on their sampling rate. Nyquist-rate DACs are further categorized into capacitor-based, resistor-based, and current source-based architectures [3] by their structure. In contrast, oversampling DACs are typically realized using Sigma-Delta modulation techniques.

A comparison of DACs with different architectures is provided in Table 1.1. Among these, the current-steering DAC stands out for its ability to achieve both high speed and high resolution. In this project, the DAC is aimed for 10-bit resolution and 128 MHz sampling rate, which is kind of medium solution and medium-high sampling rate. Current steering DAC is a better option to start with in this scenerio.

Table 1.1: Comparison of different DAC architectures.

DAC Type	Speed	Resolution	Typical Use
R-2R Ladder	Medium	Medium	Audio, general-purpose
Current-Steering	High	High	Communications
Capacitor-based	Medium	Low	SAR-based DACs
Sigma-Delta	Low	Very High	Audio, sensors

Figure 1.1 illustrates the architecture of a binary current-steering DAC. For an N-bit DAC, there are N current cells, each providing a current that is proportional to the bit weight. Each current cell is controlled by a digital bit that turns the corresponding switch on or off. When a digital bit is high ('1'), the switch is activated, allowing the current to flow into the output. In this way, the digital input code is accurately converted into an analog current. Detail about design consideration in current steering DAC will be discussed in Chapter 2.

1.2.2 Performance Metrics of DAC

Differential Non-linearity(DNL)

With the successive code input, the ideal DAC output is a staircase-like waveform and the output difference between two successive code is always l least significant bit(LSB). However, due to the mismatch of the current source, each step may differ from 1 LSB, inducing nonlinearity. This kind of nonlinearity is called Differential Nonlinearity. And the definition of DNL for code k is

$$DNL(k) = \frac{\Delta_k}{\Delta} - 1 \tag{1.1}$$

Where Δ_k is output difference between input code k-1 and k, and Δ is the ideal 1 LSB output.

Integral Non-linearity(INL)

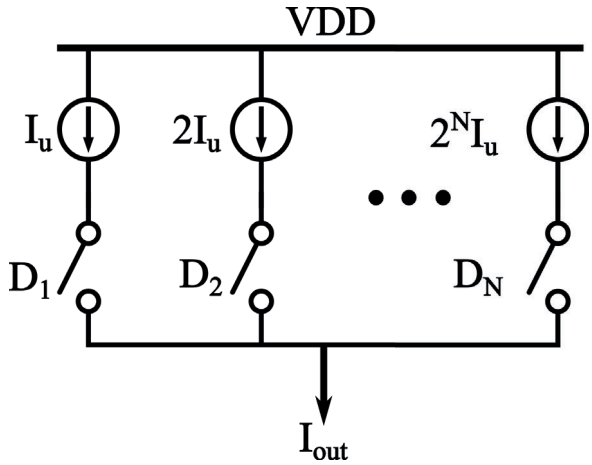


Figure 1.1: Diagram of a binary current steering DAC.

INL is the difference between real output and ideal output with the same input code. And it can be written as the summary of the DNL

$$INL(k) = \sum_{i=1}^{k} DNL(k) \tag{1.2}$$

Where $INL(k)$ is the INL for code k.

Signal to Noise Ratio (SNR)

SNR is the ratio between the signal power and noise power and its definition is

$$SNR(dB) = 20log(\frac{P_{sig}}{P_{noise}}) \tag{1.3}$$

Where P_{sig} is the signal power and P_{noise} is the noise power.

Signal to Noise and distortion Ratio (SNDR)

SNDR is the ratio between the signal power and the summary of noise power and distortion power. Its definition is

$$SNDR(dB) = 20log(\frac{P_{sig}}{P_{noise} + P_{distortion}}) \tag{1.4}$$

Where $P_{distortion}$ is the distortion power.

Effective Number of Bit (ENOB)

ENOB can be calculated from SNDR by

$$ENOB = \frac{SNDR(dB) - 1.76}{6.02} \tag{1.5}$$

Spurious Free Dynamic Range (SFDR)

SFDR is the ratio between the signal power and highest in-band spur power and its definition is

$$SFDR(dB) = 20log(\frac{P_{sig}}{P_{highest\ spur}}) \tag{1.6}$$

Where $P_{highest\ spur}$ is highest in-band spur power.

1.3 Outline of the thesis

Chapter 1: provides a brief introduction to the background and basics of the DAC.

Chapter 2: discusses the design considerations of current-steering DACs in detail.

Chapter 3: introduces the concept of tree-structured DEM used to eliminate mismatches in the current sources, and presents the implementation details of an 8-level tree-structured DEM.

Chapter 4: introduces the concept and theory of linear interpolation, and discusses how the linear interpolation technique can suppress images to the required level.

Chapter 5: presents the design details of the DAC circuits.

Chapter 6: presents the simulation results of the linear interpolation DAC. Various simulations are performed to characterize the DAC.

Chapter 7: provides the conclusion and outlines future work.

Chapter 2

Design Considerations in Current Steering DAC

2.1 Segmentation of the DAC

In the design of current steering DAC, two common schemes used to encode DAC are binary and thermometer. In the binary-weighted scheme, each current cell is controlled by one bit, so only N cells are needed in N bit DAC. The digital logic and mapping of current cells are simple to implement. However, since multiple cells have to be switched when digital code changes by one, the mismatch between the cells will accumulate, contributing to bad linearity. Therefore, the matching requirement on the current source cells is strict. In thermometer weighted scheme, only one cell is switched when the digital code changes by one, hence the linearity will be much better than that in the binary-weighted scheme. However, since $2^N - 1$ current cells are needed for a N bit DAC, the area of digital circuit will explode exponentially and mapping of current cells will be troublesome.

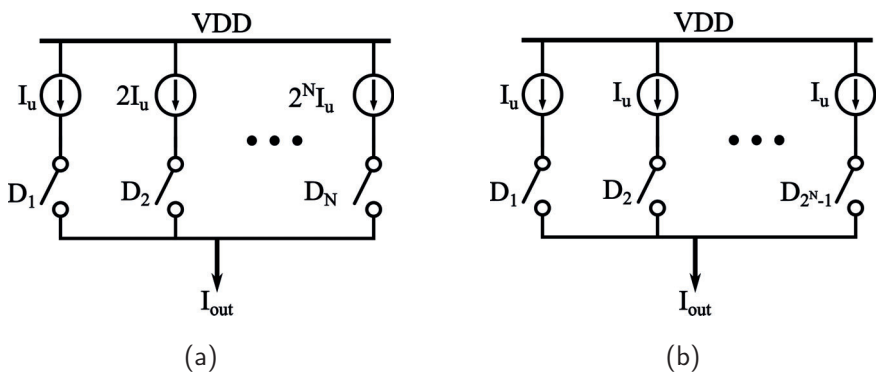


Figure 2.1: (a) Binary-weighted DAC(b) Thermometer-weight DAC

Considering the advantages and disadvantages of two schemes, nowadays segmentation of DAC is commonly used in both industry and academia. Usually the MSB part of the current steering DAC is encoded in thermometer-weighted scheme, while the LSB part is encoded in the binary-weighted scheme. In this way, good linearity can be achieved, while having moderate complexity in the digital

part of the DAC circuit. In [4], a method to estimate optimal segmentation ratio is given. On the premise of meeting the requirement of DNL and INL, required area vs segmentation ratio is plotted. The segmentation that gives the lower area can be chosen as a starting point in design.

2.2 Current Source Mismatch

Mismatch in the current source cell can be categorized into system mismatch and random mismatch. System mismatch can come from the gradient effect and edge effect. In thermometer-weighted architecture, column-row selection scheme is commonly used to switch thermometer cell, which switches adjacent cells one by one. Therefore, mismatch from the gradient effect will accumulate and degrade the linearity.A Q^2 random walk scheme[5], is proposed to solve this problem. Each thermometer cell is split into multiple units and spread across the array. The current cells are switched in a sequence such that the system mismatch doesn't accumulate any more.

However, the change of switch sequence will not help with the random mismatch. The random mismatch of current source can be estimated through Pelgrom model[6].

$$\frac{\sigma I_u}{I_u} = \frac{A_\beta^2 + \frac{4A_{vt}^2}{(V_{GS}-V_t)^2}}{2(WL)_{min}} \tag{2.1}$$

where I_u is the unit current, σI_u is the standard deviation of the unit current, A_β is the current factor mismatch parameter, A_{vt} is the threshold voltage mismatch parameter. Both of A_β, A_{vt} are process-dependent variables. Since $\frac{\sigma I}{I}$ is reversely proportional to the area, usually larger area for the current source is required to reduce the mismatch.

2.3 Finite impedance of current cell

In a current cell, the current is commuted by differential switches. Ideally, the output impedance of current source is infinite and all the current will flow into the load. However, the output impedance of a current source is finite due to channel-length modulation. The finite output impedance of a current source will introduce nonlinearity on the output.

An equivalent circuit of a current steering DAC is shown in Figure 2.2, where the output impedance of each unit cell is represented with Z_o and the load impedance is Z_L, while d is the digital input code ranging from 0 to N. All current flow into the positive output node with d = N while all current flow into the negative node with d = 0.

The current flowing into positive node and negative node is

$$I_p = I_u \cdot d, I_n = I_u \cdot (N - d) \tag{2.2}$$

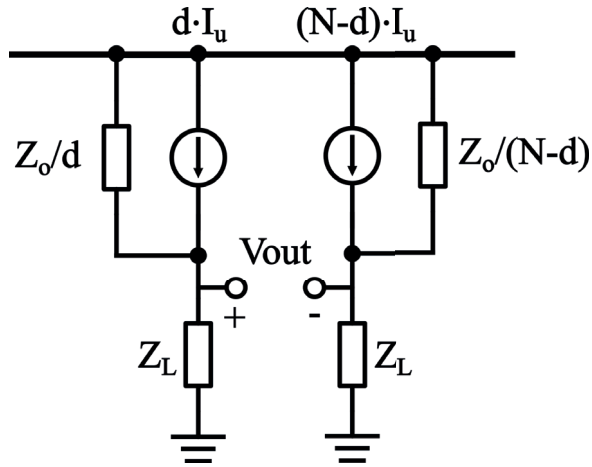


Figure 2.2: Equivalent circuit of DAC.

And the equivalent output impedance of turned-on current cell at each node can be written as

$$Z_p = \frac{Z_o}{d} // Z_L, Z_n = \frac{Z_o}{N-d} // Z_L \tag{2.3}$$

The output is differential and can be written as

$$V_{out} = V_p - V_n = I_u \cdot d \cdot (\frac{Z_o}{d} // Z_L) - I_u \cdot (N-d) \cdot (\frac{Z_o}{N-d} // Z_L) \tag{2.4}$$

Z_p can be decomposed into Taylor series as

$$Z_p = \frac{Z_L}{1 + \frac{Z_L}{Z_o} \cdot d} = Z_L \left[1 - \frac{Z_L}{Z_o} \cdot d + (\frac{Z_L}{Z_o})^2 \cdot d^2 + ...\right] (\frac{Z_L}{Z_o} \cdot d << 1) \tag{2.5}$$

Similarly, Z_n can be rewritten as

$$\begin{aligned} Z_n &= \frac{Z_L}{1 + \frac{Z_L}{Z_o} \cdot (N-d)} \\ &= Z_L \left[1 - \frac{Z_L}{Z_o} \cdot (N-d) + (\frac{Z_L}{Z_o})^2 \cdot (N-d)^2 + ...\right] (\frac{Z_L}{Z_o} \cdot (N-d) << 1) \end{aligned} \tag{2.6}$$

Substituting equation (2.5) (2.6) into equation (2.4), V_{out} can be simplified as

$$\begin{aligned} V_{out} &= -I_u \cdot Z_L \cdot N + 2I_u \cdot Z_L \cdot d + 2 \cdot \frac{Z_L^3}{Z_0^2} \cdot d^3 \cdot I_u \\ &= 2Z_L \cdot N \cdot I_u \left[-\frac{1}{2} + \frac{d}{N} + (\frac{Z_L \cdot N}{Z_o})^2 \cdot (\frac{d}{N})^3\right] \end{aligned} \tag{2.7}$$

The transfer function between digital input and discrete output is give by equation (2.7). This indicates that the third-order harmonic is the dominant source of distortion, and its magnitude is proportional to $(\frac{Z_L \cdot N}{Z_o})^2$. Increasing the

output impedance of a current source can improve the linearity. The expression of the output impedance is

$$Z_o = ro//\frac{1}{j\omega C_p} = \frac{r_o}{1 + j\omega r_o C_p}, \tag{2.8}$$

where r_o is the output resistance of the current source and C_p is the parasitic capacitance at the output node. A cascode is commonly used in the current cell to achieve higher output resistance. However, as the signal frequency increases, the output impedance decreases due to parasitic capacitance, which significantly affects high-speed designs. In [7], a cascode stage is added after the switches rather than before switches. And bleeding current is utilized to keep that cascode stage always on. In this way, the switched capacitance is reduced and higher linearity can be achieved over wider frequencies.

2.4 Disturbance from switching activities

2.4.1 Disturbance on common source node of switches

Ideally, during switching, the current from current source should remain constant. However, as illustrated in Figure 2.3, voltage fluctuations at tail node X can occur, introducing current glitches at the output. Since the switching activity is code-dependent, these glitches can generate unwanted harmonics in the output spectrum.

To mitigate these glitches, it is crucial to ensure that at least one switch remains on during the switching transition. If both switches turn off simultaneously, the voltage at node X may be pulled toward the supply voltage or ground, resulting in no current flow into the output and causing a significant glitch.

To prevent this, the crossing point of the switch control signals must be carefully tuned to avoid simultaneous turn-off. For instance, when using a PMOS current source, a lower crossing point of the control signals is preferred to ensure one switch stays on throughout the transition

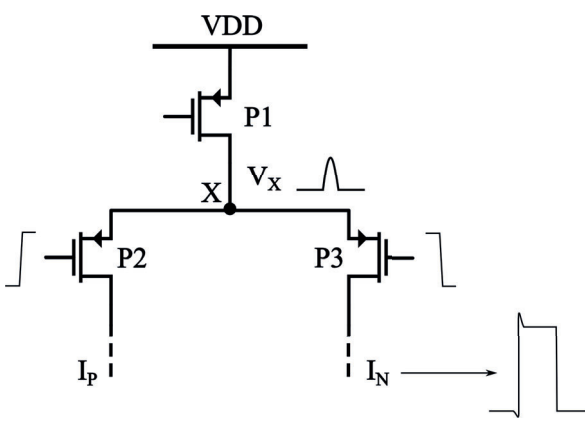


Figure 2.3: Disturbance on tail node.

Usually, dimension of switches is minimized to reduce the loading on the driver circuit. Meanwhile, with smaller switches, the coupling between gate node and tail node can be reduced, contributing to lower glitches. The glitches on the output can also be attenuated by lowering the swing range of switch control signal, which can attenuate the disturbance on common source node. However, lower sweeping range of switch control requires more effort on the design of driver and the driver will consume more power.

2.4.2 Disturbance on bias node

Current sources in the DAC are usually biased by a current mirror, which mirrors the reference current to the current source. Ideally, the bias voltage should remain constant when DAC is working. However, the source impedance of bias circuit is not zero. As shown in Figure 2.4, the disturbance on the common source node of switches will be coupled to bias node through parasitic capacitance. If a global bias is used to provide bias voltage for all the current cells, the disturbance on the bias node will induce current fluctuation on all the cells. Since switching is code-dependent , the disturbance on bias node is code-dependent as well and can introduce distortion at output current.

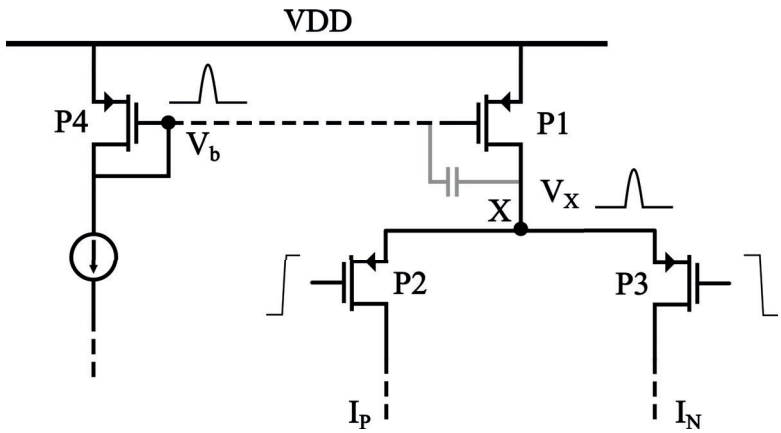


Figure 2.4: Disturbance on bias node.

2.4.3 Disturbance on the supply

Ideally, the source impedance of supply is zero and the supply voltage should be constant. However, since there are parasitic resistance from the metal line and parasitic inductance from the package, the supply voltage will fluctuate during switching since current glitches will flow into or out supply node. The disturbance on the supply is code-dependent and will induce distortion. In [8], quad switches are proposed to make the disturbance on the supply code-independent. As shown in Figure 2.5, there are two switches on each side. The control logic is generated in a way such that the current will be toggled between two switches on the same side when digital code doesn't change. Now, switching happens at every clock edge,

turning out to be code-independent. However, quad switches transfer switching from code-dependent to code-independent at the cost of more power and circuit.

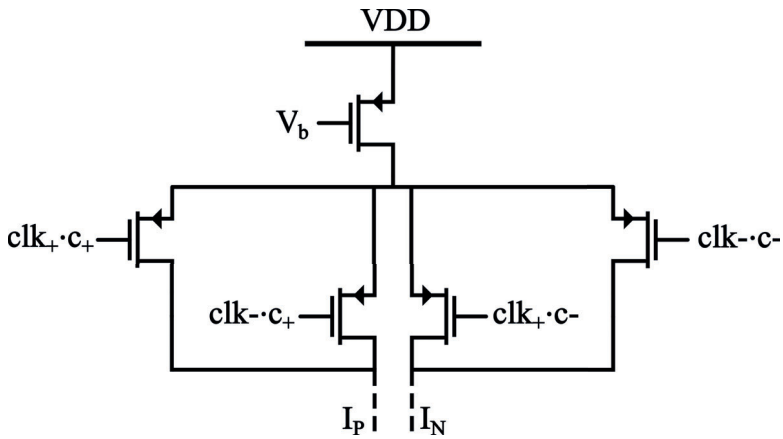


Figure 2.5: Quad switch

2.5 Time skew

There is mismatch between time at which digital signal transitions for each current cell. And this time skew between different cells can induce dynamic error in the DAC output. There are multiple resources of time skew, including the clock delay mismatch, digital logic delay mismatch, latch mismatch[9]. To mitigate the delay mismatch of digital signal, latch is usually used to synchronize the digital signal.

However, the mismatch between latches and switches can still result in transition mismatch between different cell, which can't be eliminated by the latches. Moreover, the layout mapping can induce delay mismatch in clock path and output signal path. Usually layout tree for clock and timing distribution is utilized to minimize the delay mismatch between distant current cells[9].

Chapter 3
Dynamic Element Matching

As discussed in Chaper 2, mismatch between different current cells will add error to the output. Since these errors are nonlinearly code-dependent , they will induce spurious tones in the output spectrum, thereby degrading the linearity. One way to mitigate the nonlinearity resulted from mismatch is calibration. Different calibration approach has been proposed in [10][11][12]. However, an ADC is usually needed in calibration of the DAC, which adds to the complexity and cost of the circuit. Another approach to linearize the DAC is to apply dynamic element matching(DEM) technique, which is widely used in $\Sigma\Delta$ ADC [13][14]. The basic idea of DEM is the set of unit elements selected will change dynamically even though input code is the same. In this way, the errors at output is nondeterministic with the input code and linear on average. The distortion induced by mismatch is scrambled into random noise.

DEM has also been applied to high-resolution Nyquist-rate DACs to improve the linearity [15]. And tree-structured DEM has gained its popularity because it can be applied to the DAC with arbitrary output levels[16]. In this chapter, the principles of tree-structured DEM is explained in detail and a 8-level tree-structured DEM logic is implemented for the DAC in this project.

3.1 Unary weighted DAC

A unary-weighted DAC with N+1 output levels composes N unary cells. And each cell can been considered as a 1-bit DAC. Figure 3.1 shows a unary weighted DAC with N cells. $x[n]$ is the digital input during n-th sampling period while $y(t)$ is the analog output. Assuming the output of a 1-bit DACs is one quantization interval Δ when input is 1, $y(t)$ can be denoted as the function of $x[n]$

$$y(t) = a(t - nT) \cdot x[n] \cdot \Delta \tag{3.1}$$

Where $a[t]$ is 1 if $0 < t < T$ and 0 otherwise. If the mismatch in each 1-bit DAC is taken into account, the output of i th 1-bit DAC during nth sampling period is

$$y_i(t) = \begin{cases} a(t - nT) \cdot x_i[n] \cdot \Delta + e_i(t - nT), & x_i[n] = 1 \\ 0, & x_i[n] = 0 \end{cases} \tag{3.2}$$

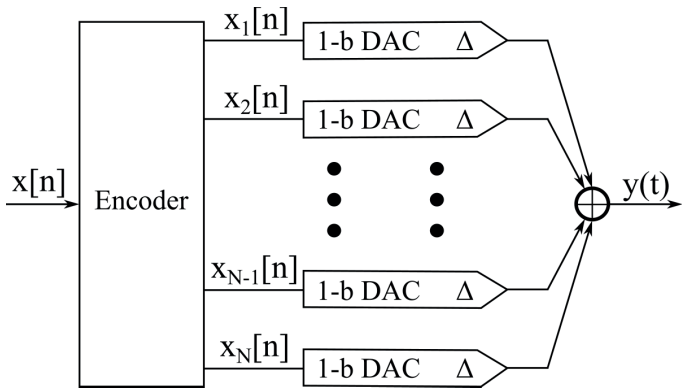


Figure 3.1: Block Diagram of Unary weighted DAC

Where $e_i(t - nT)$ is the error of ith 1-bit DAC. Adding the output of all 1-bit DAC together, output $y(t)$ is

$$y(t) = a(t - nT) \cdot x[n] \cdot \Delta + \epsilon(t) \tag{3.3}$$

And the expression of $\epsilon(t)$ is

$$\epsilon(t) = \sum_{i=1}^{N} x_i(n) \cdot e_i(t - nT) \tag{3.4}$$

If $\epsilon(t)$ is a constant or linearly dependent on $x[n]$. The output $y(t)$ is still linear and $\epsilon(t)$ won't induce distortion. However, only when $x[n] = 0\,, N$, $\epsilon(t)$ can be denoted as a linear function of $x[n]$.

$$\epsilon(t) = p(t - nT) \cdot x[n] \tag{3.5}$$

And

$$p(t) = \frac{1}{N} \cdot \sum_{i=1}^{N} e_i(t) \tag{3.6}$$

In general case, $\epsilon(t)$ can be written as

$$\epsilon(t) = p(t - nT) \cdot x[n] + e_{DAC}(t) \tag{3.7}$$

And $y(t)$ can be written as

$$y(t) = [a(t - nT) + p(t - nT)] \cdot x[n] \cdot \Delta + e_{DAC}(t) \tag{3.8}$$

Where $e_{DAC}(t)$ is deterministic error and nonlinearly dependent on the $x[n]$. We can denote

$$\alpha_i(t) = a(t) + e_i(t) \tag{3.9}$$

Then

$$\alpha(t) = \frac{1}{N} \sum_{i=1}^{N} \alpha_i(t) = a(t) + p(t) \tag{3.10}$$

$y(t)$ can be rewritten as

$$y(t) = \alpha(t - nT) \cdot x[n] \cdot \Delta + e_{DAC}(t) \tag{3.11}$$

If the expectation of $e_{DAC}(t)$ is 0 for any of input $x[n]$, then $y(t)$ is linearly dependent on x[n] on average. Distortion induced by $e_{DAC}(t)$ is scrambled into broadband noise. The goal of DEM algorithm in the next section is to design random selection sequnce that make expectation of $e_{DAC}(t)$ 0.

3.2 Tree-structured DEM

The principles of tree-structured DEM are discussed in [16] thoroughly. In the tree-structured DEM, the input code is split by digital blocks stage by stage, with randomization added in each stage. The topology of DEM block looks like a tree structure, that's where the name comes from. Before discussing the principle of tree-structure DEM, some definitions need to be proposed first.

$DAC^{(u,w)}$: For any integer u, w that satisfy $0 < u < w < N + 1$, $DAC^{(u,w)}$ is composed of the uth through wth 1-bit DACs and an encoder[16]. The digital input is written as

$$x^{(u,w)}[n] = \sum_{i=u}^{w} x_i[n] \tag{3.12}$$

The analog output is

$$y^{(u,w)}(t) = \alpha^{(u,w)}(t - nT) \cdot x^{(u,w)}[n] \cdot \Delta + e_{DAC}(t) \tag{3.13}$$

Where

$$\alpha^{(u,w)}(t) = \frac{1}{w - u + 1} \sum_{i=u}^{w} \alpha_i(t) \tag{3.14}$$

When $u = w$, $e_{DAC}(t) = 0$.

$S^{(u,v,w)}$: For any integer u, w that satisfy $0 < u < v < w < N + 1$, $S^{(u,v,w)}$ is a switching block that splits digital input $x^{(u,w)}$ into $x^{(u,v)}, x^{(v+1,w)}$ as the digital input of $DAC^{(u,v)}$ and $DAC^{(v+1,w)}$ respectively. Figure 3.2 shows how $x^{(u,w)}$ is decomposed by switching block $S^{(u,v,w)}$.

$x^{(u,v)}, x^{(v+1,w)}$ are calculated by

$$x^{(u,v)}[n] = (1 - G^{(u,v,w)}[n]) \cdot x^{(u,w)}[n] + s^{(u,v,w)}[n] \tag{3.15}$$

$$x^{(v+1,w)}[n] = G^{(u,v,w)}[n] \cdot x^{(u,w)}[n] - s^{(u,v,w)}[n] \tag{3.16}$$

Where $s^{(u,v,w)}[n]$ is a switching sequence. And $G^{(u,v,w)}[n]$ is

$$G^{(u,v,w)}[n] = \frac{w - v}{w - u + 1} \tag{3.17}$$

$S^{(u,v,w)}$ decomposes $DAC^{(u,w)}$ into $DAC^{(u,v)}$ and $DAC^{(v+1,w)}$. The analog output of $DAC^{(u,w)}$ can be written as

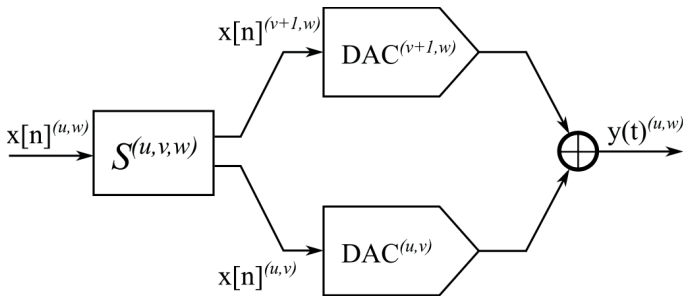


Figure 3.2: Block Diagram of decomposition

$$y^{(u,w)}(t) = y^{(u,v)}(t) + y^{(v+1,w)}(t) \tag{3.18}$$

Substituting equation(3.11), $y^{(u,w)}(t)$ can be rewritten as

$$\begin{aligned} y^{(u,w)}(t) = \alpha^{(u,v)}(t - nT) \cdot x^{(u,v)}[n] \cdot \Delta + e_{DAC}^{(u,v)}(t) \\ + \alpha^{(v+1,w)}(t - nT) \cdot x^{(v+1,w)}[n] \cdot \Delta + e_{DAC}^{(v+1,w)}(t) \end{aligned} \tag{3.19}$$

With equations (3.14)(3.15)(3.16), $y^{(u,w)}(t)$ can rewritten as

$$\begin{aligned} y^{(u,w)}(t) = \alpha^{(u,w)}(t - nT) \cdot x^{(u,w)}[n] \cdot \Delta + s^{(u,v,w)}[n]\Delta^{(u,v,w)}(t - nT) \\ + e_{DAC}^{(u,v)}(t) + e_{DAC}^{(v+1,w)}(t) \end{aligned} \tag{3.20}$$

Where

$$\Delta^{(u,v,w)}(t) = \left[\alpha^{(u,v)}(t) - \alpha^{(v+1,w)}(t)\right] \Delta \tag{3.21}$$

The decomposition process starts from the switching block $S^{(1,v,N)}$ and can be continued recursively until $S^{(u,u,w)}(u = 1, 2, 3, ..., N-1), w > u$, which drives the 1-bit DAC directly. From equation(3.20), $e_{DAC}^{(u,w)}(t)$ can be written as

$$e_{DAC}^{(u,w)}(t) = s^{(u,v,w)}[n]\Delta^{(u,v,w)}(t - nT) + e_{DAC}^{(u,v)}(t) + e_{DAC}^{(v+1,w)}(t) \tag{3.22}$$

$e_{DAC}^{(1,N)}(t)$ can be obtained from the recursion step

$$e_{DAC}^{(1,N)}(t) = \sum_{u,v,w}^{w} s^{(u,v,w)}[n]\Delta^{(u,v,w)}(t - nT) + \sum_{1}^{N} e_{DAC}^{(u,u)}(t) \tag{3.23}$$

Since $e_{DAC}^{(u,u)}(t) = 0$ for all u, equation above can be rewritten as

$$e_{DAC}^{(1,N)}(t) = \sum_{u,v,w} s^{(u,v,w)}[n]\Delta^{(u,v,w)}(t - nT) \tag{3.24}$$

Where the sum in equation(3.24) includes all the value of u, v, w used in the whole decomposition process. As defined in equation(3.21), $\Delta^{(u,v,w)}(t - nT)$ is

a pulse with fixed value, the statistical properties of $e_{DAC}^{(1,N)}(t)$ are dependent on $s^{(u,v,w)}[n]$. As discussed in Section 3.1, the necessary condition to mitigate the distortion induced by $e_{DAC}^{(1,N)}(t)$ is making the expectation of $e_{DAC}^{(1,N)}(t)$ be zero for all input $x[n]$. Switching sequence $s^{(u,v,w)}[n]$ need to be designed properly to achieve this condition.

For any switching block $S^{(u,v,w)}$, if $x^{(u,w)} \in \{0, w-u+1\}$, switching sequence $s^{(u,v,w)}$ is 0 since there is only one choice for $x^{(u,v)}, x^{(v+1,w)}$. Otherwise, there are at least two choices for $s^{(u,v,w)}$. In the tree-structured DEM, the following two choices are used.

$$s^{(u,v,w)}[n] = \left\langle \frac{w-v}{w-u+1} x^{(u,w)} \right\rangle \text{ or } \left\langle \frac{w-v}{w-u+1} x^{(u,w)} \right\rangle - 1 \tag{3.25}$$

Where $< a >= a - \lfloor a \rfloor$, $\lfloor a \rfloor$ denotes the largest integer which is less than or equal to a. The selection of $s^{(u,v,w)}[n]$ ensures that $x^{(u,v)}, x^{(v+1,w)}$ is integer.

The switching sequence can have zero value or two non-zero values with opposite signs. For each switching block, they can select switching sequence from those values dynamically in such a way that the expectation of each $s^{(u,v,w)}$ is zero. In the next section, a design example of 8-level tree-structured DEM is shown to present how the switching sequence can be selected.

3.3 Design case of a 8-level tree-structured DEM

In this section, an 8-level tree-structure DEM for use in a 10-bit DAC is described. 10-bit DAC is segmented into 3-bit unary part, 3-bit unary part, 4-bit binary part. The tree-structure DEM is applied to the most significant 3-bit for the purposes of improving the linearity. The tree-structure DEM decoder driving 7 1-bit DACs in MSB part is shown in Figure. Each switching block needs to select appropriate switching sequence with an expectation of zero.

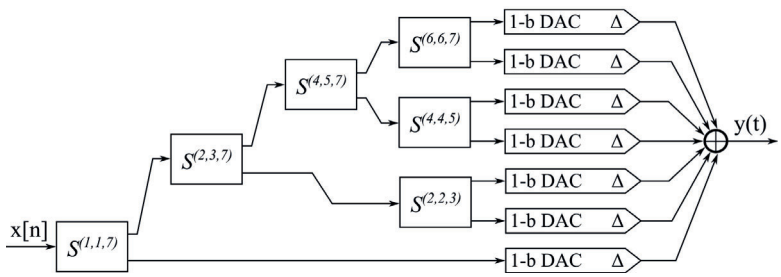


Figure 3.3: Block Diagram of 8-level tree-structured DEM

Let's discuss $s^{(1,1,7)}[n]$ first. For each case of digital input $x[n]$, $s^{(1,1,7)}[n]$ can be expressed as

$$s^{(1,1,7)}[n] = \begin{cases} 0, & x^{(1,7)}[n] \in \{0,7\} \\ \frac{6}{7}, -\frac{1}{7}, & x^{(1,7)}[n] \in \{1\} \\ \frac{5}{7}, -\frac{2}{7}, & x^{(1,7)}[n] \in \{2\} \\ \frac{4}{7}, -\frac{3}{7}, & x^{(1,7)}[n] \in \{3\} \\ \frac{3}{7}, -\frac{4}{7}, & x^{(1,7)}[n] \in \{4\} \\ \frac{2}{7}, -\frac{5}{7}, & x^{(1,7)}[n] \in \{5\} \\ \frac{1}{7}, -\frac{6}{7}, & x^{(1,7)}[n] \in \{6\} \end{cases} \tag{3.26}$$

To meet the requirement that the expectation of $s^{(1,1,7)}[n]$ should be zero, the probability distribution of $s^{(1,1,7)}[n]$ must satisfy the following equations

$$\begin{cases} P(s^{(1,1,7)}[n]=0)=1, & x^{(1,7)}[n] \in \{0,7\} \\ P(s^{(1,1,7)}[n]=\frac{6}{7})=\frac{1}{7}, P(s^{(1,1,7)}[n]=-\frac{1}{7})=\frac{6}{7}, & x^{(1,7)}[n] \in \{1\} \\ P(s^{(1,1,7)}[n]=\frac{5}{7})=\frac{2}{7}, P(s^{(1,1,7)}[n]=-\frac{2}{7})=\frac{5}{7}, & x^{(1,7)}[n] \in \{2\} \\ P(s^{(1,1,7)}[n]=\frac{4}{7})=\frac{3}{7}, P(s^{(1,1,7)}[n]=-\frac{3}{7})=\frac{4}{7}, & x^{(1,7)}[n] \in \{3\} \\ P(s^{(1,1,7)}[n]=\frac{3}{7})=\frac{4}{7}, P(s^{(1,1,7)}[n]=-\frac{4}{7})=\frac{3}{7}, & x^{(1,7)}[n] \in \{4\} \\ P(s^{(1,1,7)}[n]=\frac{2}{7})=\frac{5}{7}, P(s^{(1,1,7)}[n]=-\frac{5}{7})=\frac{2}{7}, & x^{(1,7)}[n] \in \{5\} \\ P(s^{(1,1,7)}[n]=\frac{1}{7})=\frac{6}{7}, P(s^{(1,1,7)}[n]=-\frac{6}{7})=\frac{1}{7}, & x^{(1,7)}[n] \in \{6\} \end{cases} \tag{3.27}$$

Different $s^{(1,1,7)}[n]$ corresponds to different $x^{(1,1)}[n]$ and $x^{(2,7)}[n]$. Take $x^{(1,7)}[n] = 1$ as an example, $x^{(1,1)}[n]$ should be 0 or 1 and the probability should be $\frac{6}{7}$ and $\frac{1}{7}$ respectively. The digital logic of switching block $S^{(1,1,7)}$that generates $x^{(1,1)}[n]$ and $x^{(2,7)}[n]$ is shown is in Figure 3.4

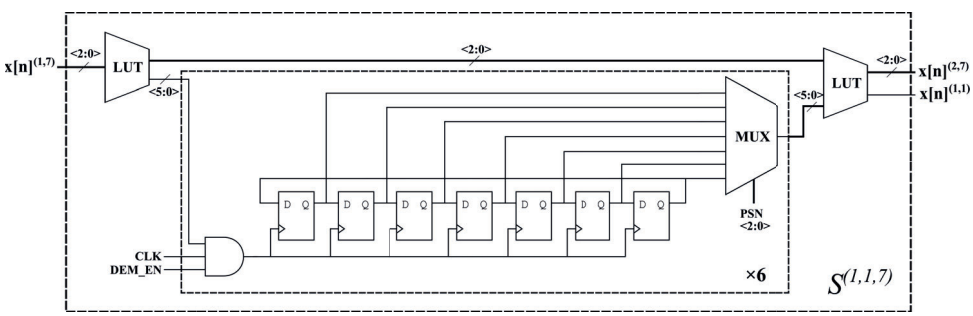


Figure 3.4: Digital logic of switching block $S^{(1,1,7)}$

LUT in Figure 3.4 will search case for each input code and choose output from two sets depending on the output code of MUX. The output code of MUX is called selection code and denoted as $SEL[n] < 5 : 0 >$, with 6 bits for 6 input cases. For example, if $x^{(1,7)}[n] = 1$, LUT will output $x^{(1,1)}[n] = 0, x^{(2,7)}[n] = 1$ when $SEL[n] < 0 >= 0$ and $x^{(1,1)}[n] = 1, x^{(2,7)}[n] = 0$ when $SEL[n] < 0 >= 1$. Figure 3.4 also shows the digital logic to generate selection code for each case. 7 D flip-flops form a recirculating shift register initialized with the predefined sequence. The MUX will select one D flip-flop output to pass depending on selection signal

$PSN < 2:0 >$, which is pseudo random number. Condition (3.27) can achieved by defining corresponding sequence in shift register. For example, for $x^{(1,7)}[n] = 1$, recirculating shift register can initialized with the sequence 1, 0, 0, 0, 0, 0, 0.

If $PSN < 2:0 >$ is fixed, the output of MUX will be periodic, with a period of 7 sampling periods. In this case, though (3.27) is satisfied, the switching sequence is still deterministic and will induce harmonics. So pseudo-random number is provided to determine which D flip-flops output will be selected as the the output of MUX. The pseudo random number generator is implemented by a 4-bit linear feedback shift register(LFSR) shown in Figure 3.5 , which is clocked at $\frac{1}{7}$ sampling rate. MUX will select a new output node according to pseudo random number every 7 sampling clocks. The output sequence of LFSR is periododic as well. But the period is 16 sampling period, which makes the switching sequence random enough.

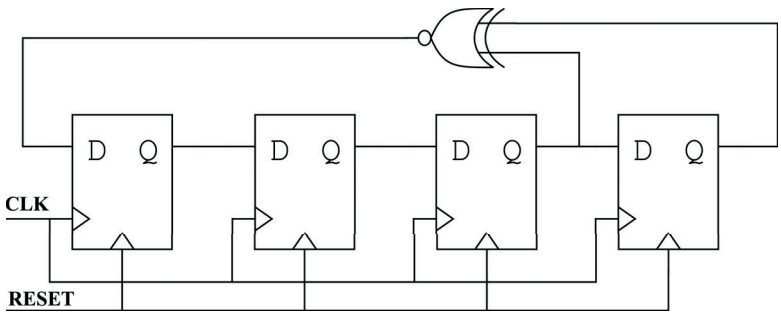


Figure 3.5: 4-bit Linear feedback shift register

Similarly, the switching sequence of $s^{(2,3,7)}[n]$ for all value of $x^{(2,7)}[n]$ is listed as

$$s^{(2,3,7)}[n] = \begin{cases} 0, & x^{(2,7)}[n] \in \{0,6\} \\ 0,-1, & x^{(2,7}[n] \in \{3\} \\ \frac{1}{3},-\frac{2}{3}, & x^{(2,7)}[n] \in \{2,5\} \\ \frac{2}{3},-\frac{1}{3}, & x^{(2,7)}[n] \in \{1,4\} \end{cases} \tag{3.28}$$

The probability distribution of $s^{(1,2,6)}[n]$ need to satisfy

$$\begin{cases} P(s^{(2,3,7)}[n]=0)=1, & x^{(2,7)}[n] \in \{0,3,6\} \\ P(s^{(2,3,7)}[n]=\frac{1}{3})=\frac{2}{3}, P(s^{(2,3,7)}[n]=-\frac{2}{3})=\frac{1}{3}, & x^{(2,7)}[n] \in \{2,5\} \\ P(s^{(2,3,7)}[n]=\frac{2}{3})=\frac{1}{3}, P(s^{(2,3,7)}[n]=-\frac{1}{3})=\frac{2}{3}, & x^{(2,7)}[n] \in \{1,4\} \end{cases} \tag{3.29}$$

For the other switching sequence $s^{(u,v,w)}[n]$, the expression of them can be generalize as

$$s^{(u,v,w)}[n] = \begin{cases} 0, & x^{(u,w)}[n]\ is\ even \\ \frac{1}{2},-\frac{1}{2}, & x^{(u,w)}[n]\ is\ odd \end{cases} \tag{3.30}$$

The probability distribution of $s^{(2,3,7)}[n]$ need to satisfy

$$\begin{cases} P(s^{(u,v,w)}[n]=0)=1, & x^{(u,w)}[n]\ is\ even \\ P(s^{(u,v,w)}=\frac{1}{2})=\frac{1}{2}, P(s^{(u,v,w)}=-\frac{1}{2})=\frac{1}{2}, & x^{(u,w)}[n]\ is\ odd \end{cases} \tag{3.31}$$

The switching blocks to generate switching sequence $s^{(2,3,7)}[n]$ and the others are similar to Figure 3.4, but has lower complexity. All the switching blocks and pseudo random number generator is implemented in verilog. An example verilog code for switching block $S^{(1,1,7)}$ is given in **Appendix**.

Chapter 4

Interpolation

4.1 Background

The conventional DAC is a zero order hold circuit and the output is a staircase-like waveform. The output of the DAC can be represented as the convolution of discrete output with a rectangular pulse as shown in Figure 4.1, where T_s is the sampling period.

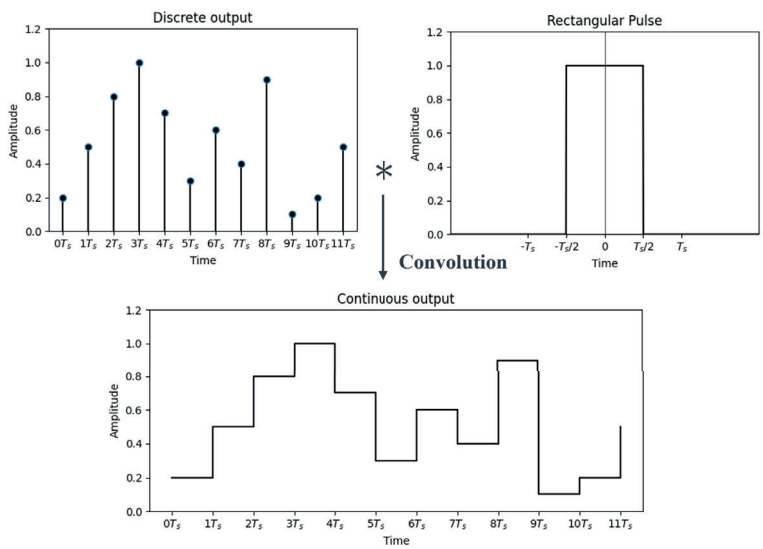


Figure 4.1: Convolution of discrete output and rectangular pulse

The data sampling frequency f_s in Figure 4.1 is $\frac{1}{T_s}$. Assuming the signal frequency of a sinusoidal input is f_{in}, the spectrum of a discrete-time output is periodic with f_s, with signal image at $n * f_s + f_{in}$ and $n * f_s - f_{in}$. The frequency response of the rectangular wave in Figure 4.1 can be expressed as

$$H(f) = sinc(f/f_s) = \frac{sin(\pi * f/f_s)}{\pi * f/f_s} \tag{4.1}$$

According to the theory of Fourier transformation, the output spectrum is the

multiplication of discrete-time output spectrum and rectangular pulse spectrum. Figure 4.2 shows the multiplication of the spectrum. Continuous output spectrum is the spectrum of discrete-time output attenuated by the sinc frequency response of rectangular pulse. Now images are distortion in the the spectrum rather than exact copy of signal tone.

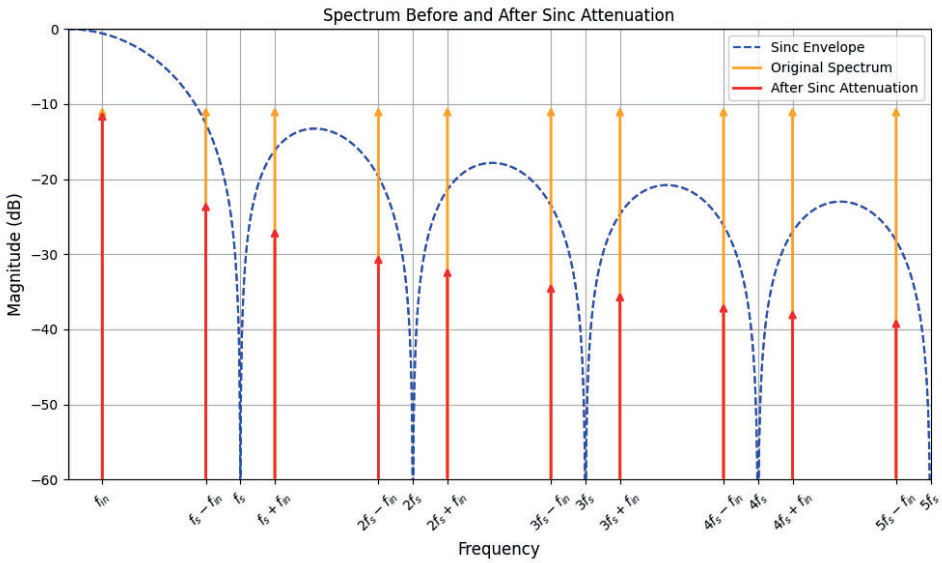


Figure 4.2: Spectrum before and after sinc attenuation

A low-pass filter (LPF) is typically required in the transmitter to suppress the spectral images. The design requirements for the LPF depend on the signal bandwidth and the sampling rate. A lower ratio of sampling rate to signal bandwidth demands a sharper filter response, which significantly increases the complexity of the LPF design and can be challenging to implement. While increasing the sampling rate can ease the filter requirements, the LPF still tends to occupy considerable chip area and consumes substantial power It will be promising if LPF can be replaced by more power-efficient and area-efficient approach.

A novel way to do filtering in analog domain without using LPF is analog interpolation, proposed in [17]. In the conventional DAC, the output transition occurs in each clock circle and completes in one step. But in the interpolation DAC, each current cell is divided into multiple units and controlled by clocks with different phases. In this way, one transition step is divided into multiple steps and the output is interpolated. A 4× linear interpolation is shown in Figure 4.3. According to the theoretical analysis in the next section, linear interpolation can provide higher attenuation on the images. If the interpolation order is high enough, the images can be attenuated to a low enough level to meet system requirements.

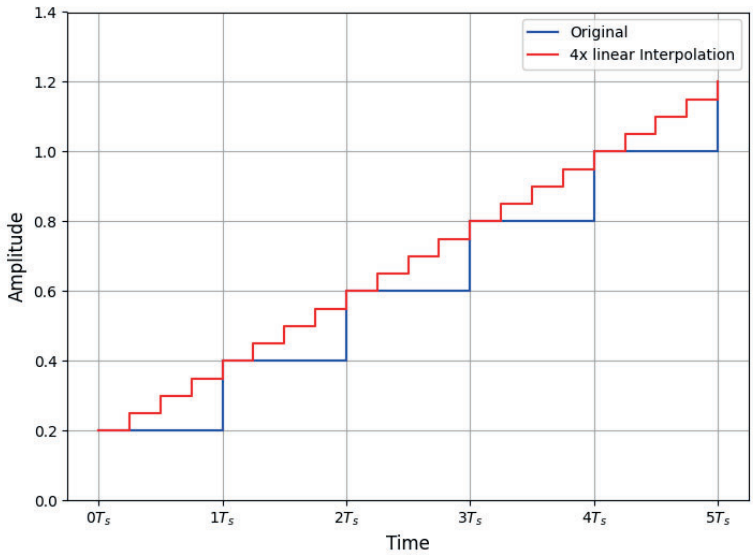


Figure 4.3: 4 fold Linear Interpolation

4.2 Theory analysis

Mathematical analysis of interpolation is discussed in depth in [18]. Figure 4.4 shows the diagram of 2-fold interpolation DAC. Both DACs are clocked with a sampling frequency f_s but the clock of the second DAC has a delay of td. $x[n]$ is the digital input of the DAC while $y_1(t)$ and $y_2(t)$ are the analog output of two DACs respectively.

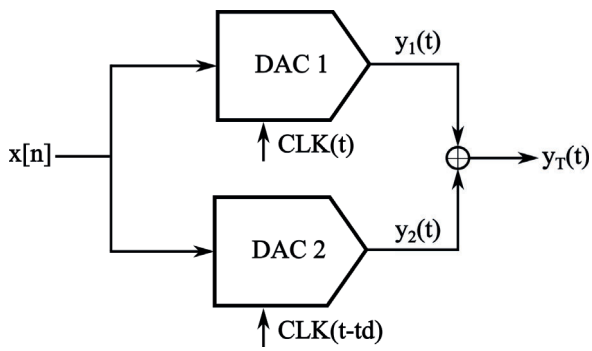


Figure 4.4: A two phase interpolation DAC

Output signal $y_1(t)$ can be written as

$$y_1(t) = \left[\sum_{n=-\infty}^{\infty} x[n] \cdot \delta(t - nT_s) \right] * rect(\frac{t}{s}) \tag{4.2}$$

Where $rect(t)$ is the rectangular pulse function and $*$ denotes convolution. The value of $rect(t)$ is 0 only if $-\frac{1}{2} < t < \frac{1}{2}$. Using

$$x[n] \cdot \delta(t - nT_s) = x(t) \cdot \delta(t - nT_s) \tag{4.3}$$

We can write $y_1(t)$ as

$$y_1(t) = x(t) \cdot \left[\sum_{n=-\infty}^{\infty} \delta(t - nT_s) \right] * rect(\frac{t}{T_s})$$

The Fourier transform of $y_1(t)$ is

$$Y_1(f) = X(f) * \left[\frac{1}{T_s} \sum_{n=-\infty}^{\infty} \delta(f - \frac{n}{T_s}) \right] \cdot (T_s \cdot \frac{sin(\pi f \cdot T_s)}{\pi f \cdot T_s}) \tag{4.4}$$

We can denote

$$A_1(f) = X(f) * \left[\sum_{n=-\infty}^{\infty} \delta(f - \frac{n}{T_s}) \right], \tag{4.5}$$

and

$$H(f) = \frac{sin(\pi f \cdot T_s)}{\pi f \cdot T_s}, \tag{4.6}$$

where $A_1(f)$ is the spectrum of discrete points train, with a period of T_s. And $H(f)$ is the frequency response of the rectangular pulse, which can be regarded as a filtering function.

The output of the second DAC can be written as

$$y_2(t) = y_1(t - td) \tag{4.7}$$

The Fourier transform of $y_2(t)$ is

$$Y_2(f) = Y_1(f) \cdot e^{-j2\pi \cdot f \cdot td} \tag{4.8}$$

The sum of $y_1(t)$ and $y_2(t)$ can be written as

$$Y_T(f) = Y_1(f) \cdot (1 + e^{-j2\pi \cdot f \cdot td}) = A_1(f) \cdot H(f) \cdot (1 + e^{-j2\pi \cdot f \cdot td}) \tag{4.9}$$

Total filtering frequency response from sinc function and interpolation is denoted by

$$H_T(f) = H(f) \cdot (1 + e^{-j2\pi \cdot f \cdot td}) = \frac{sin(\pi f \cdot T_s)}{\pi f \cdot T_s} \cdot (1 + e^{-j2\pi \cdot f \cdot td}), \tag{4.10}$$

which can be derived further as

$$H_T(f) = \frac{sin(\pi f \cdot T_s)}{\pi f \cdot T_s} \cdot e^{-j\pi \cdot f \cdot td} \cdot 2cos(\pi f \cdot td) \tag{4.11}$$

Only the amplitude of $H_T(f)$ is of interest and it is

$$|H_T(f)| = \left| \frac{sin(\pi f \cdot T_s)}{\pi f \cdot T_s} \cdot 2cos(\pi f \cdot td) \right| \tag{4.12}$$

Apart from sinc function, there is an additional filtering function $cos(\pi f \cdot td)$ which comes from analog interpolation. Assuming that $td = T_s/2$, filtering function from interpolation can be written as

$$|H_{itp}(f)| = \left| 2cos(\frac{\pi \cdot f}{2 \cdot f_s}) \right| \tag{4.13}$$

$|H_{itp}(f)|$ has zeros at $(2n-1) * f_s, n = 1, 2, 3...$ while it is equal to 1 when $f = 2n * f_s, n = 1, 2, 3....$ It means that the images near $(2n-1) * f_s$ will be attenuated further by $|H_{itp}(f)|$ while the image near $2n * f_s$ will stay almost the same.

The discussion above derives the frequency response of a 2-fold interpolation DAC. It can be generalized for a 2^N-fold interpolation DAC. Assuming $t_d[n] = \frac{n*T_s}{2^N}, n = 1, 2.., 2^N - 1$, the total filtering function from sinc function and interpolation can be written as

$$\begin{aligned} H_T(f)_{2^N} &= \frac{\sin(\pi f \cdot T_s)}{\pi f \cdot T_s} \cdot \left(1 + \sum_{n=1}^{2^N-1} e^{-j2\pi f \cdot \frac{nT_s}{2^N}} \right) \\ &= \frac{\sin(\pi f \cdot T_s)}{\pi f \cdot T_s} \cdot e^{-j\pi f \cdot T_s \cdot \frac{2^N-1}{2^N}} \cdot 2^N \cdot \prod_{k=1}^{N} \cos\left(\frac{\pi f}{2^k f_s} \right) \end{aligned} \tag{4.14}$$

If the amplitude of $H_T(f)_{2^N}$ at different interpolation order is normalized, it can be written as

$$|H_T(f)_{2^N}| = \frac{sin(\pi f/f_s)}{\pi f/f_s} \prod_{k=1}^{N} cos(\frac{\pi f}{2^k f_s}) \tag{4.15}$$

The frequency response $|H_T(f)_{2^N}|$ with different interpolation orders is plotted in Figure 4.5. Cosine terms in $|H_T(f)_{2^N}|$ give more attenuation, which can be utilized to filter the images. And with higher interpolation order, higher attenuation is achieved. Taking 4× interpolation as an example, $cos(\frac{\pi f}{4f_s}) \cdot cos(\frac{\pi f}{2f_s}) = 0$ when $f = f_s, 2f_s, 3f_s$. The image near $f_s, 2f_s, 3f_s$ will be attenuated further. When f $= 4f_s$, $cos(\frac{\pi f}{4f_s}) \cdot cos(\frac{\pi f}{2f_s}) = 1$ and it gives negligible attenuation on image near $4f_s$. The level of images near $4f_s$ will stay almost the same with the one without interpolation.

Additional filtering from interpolation makes it possible that images can be attenuated to required level without using LPF. The role that LPF plays can be replaced by the analog linear interpolation. The feasibility of this idea needs to be verified in the application case of this project. As discussed before, DAC in this project is used for WiFi 6 application, with 128MHz sampling rate and 20MHz baseband packet.Input frequency of 10 MHz is chosen as the worse case to explore how much linear interpolation could attenuate images in this transmitter

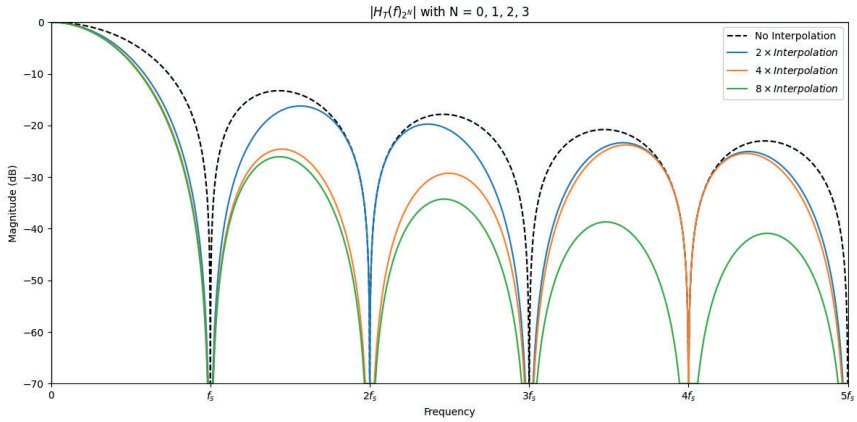


Figure 4.5: Frequency response with different interpolation order

system. With 10MHz input tone and 128MHz sampling rate, images will appear at $128MHz \pm k * 10MHz, k = 1, 2, 3....$ In Table 4.1, the level of 2nd-5th order images with different interpolation order is listed. Since images at $128MHz - k * 10MHz$ are usually higher than images at $128MHz + k * 10MHz$, only images at $128MHz - k * 10MHz$ are listed in the Table 4.1.

Table 4.1: Images level with different interpolation order

Images@	$1 \times Interp.$	$2 \times Interp.$	$4 \times Interp.$	$8 \times Interp.$
118MHz	-21.4dBc	-39.6dBc	-42.2dBc	-42.7dBc
246MHz	-27.8dBc	-27.9dBc	-52.1dBc	-54.9dBc
374MHz	-31.5dBc	-49.7dBc	-53.3dBc	-61.0dBc
502MHz	-34dBc	-34.1dBc	-34.1dBc	-64.3dBc

Figure 4.6 demonstrates the spectral mask for 20 MHz channel in WiFi 6 system. The transmit spectrum shouldn't exceed -40 dBr(dB relative to the maximum spectral density of the signal) at 30MHz offset and above [19]. With 7.5 dB margin, the images are required to be lower than -47.5 dBc.

As shown in Table 4.1, lower images can be achieved through higher interpolation order. For example, it's promising that all images level can be below -47.5 dBc if interpolation order above 8 is used. However, higher interpolation order will make the clock mapping more troublesome and the generation of multiphase clocks will need extra effort. It is desirable that the interpolation DAC can provide extra filtering needed, meanwhile, has acceptable complexity. It can be found that with 4-fold interpolation, images near $2f_s$ and $3f_s$ are below -47.5 dBc while images near $f_s, 4f_s$ still violate the requirement. However, since images near f_s is only 5.5 dB higher than the requirement. It's possible that a simple RC filter can provide additional attenuation needed. Assuming a capacitor C is added in parallel with resistor load R in each differential end, the frequency response of RC filter can be written as

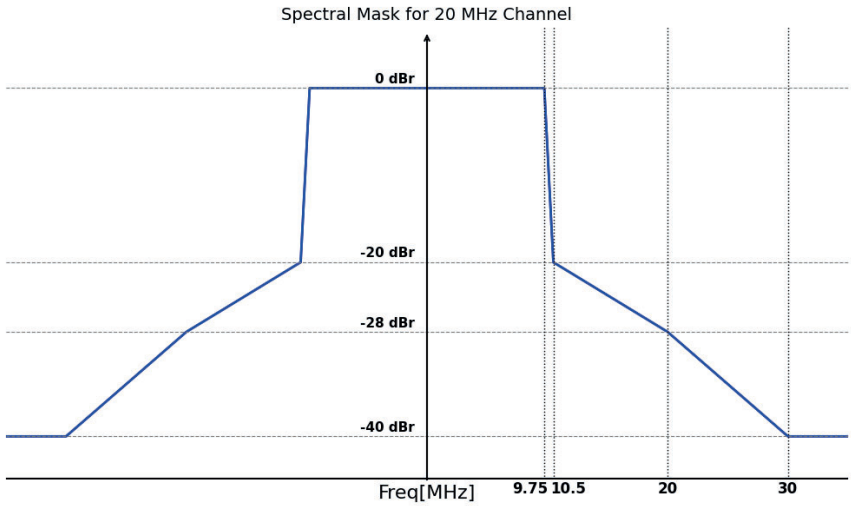


Figure 4.6: Spectral mask for 20MHz channel

$$H_{RC}(f) = \frac{R}{1 + j \cdot 2\pi \cdot f \cdot RC} \tag{4.16}$$

The amplitude of $H_{RC}(f)$ can be denoted as

$$|H_{RC}(f)| = \sqrt{\frac{R^2}{1 + (2\pi \cdot f \cdot RC)^2}} \tag{4.17}$$

The images at $f_s - f_{in}, 4 * f_s - f_{in}$ should be below -47.5 dBc with additional filtering from RC filter. The requirement can be written as

$$20log\left[\frac{|H_{RC}(f_{in})|}{|H_{RC}(f_s - f_{in})|}\right] > 47.5 - 42.2 \tag{4.18}$$

$$20log\left[\frac{|H_{RC}(f_{in})|}{|H_{RC}(4 * f_s - f_{in})|}\right] > 47.5 - 34.1 \tag{4.19}$$

It can be derived that $RC_1 > 2.12ns$, and $RC_2 > 1.46ns$ from the two equations above respectively. With a load resistor of 200Ω, the minimum capacitor needed is 10.6 pF. If a capacitor is added across the differential output rather than at each differential node, the capacitor needed is 5.3 pF equivalently, which is more acceptable.

4.3 Implementation

In theory section, the 4-fold linear interpolation DAC composes In the theory section, the 4-fold linear interpolation DAC is described as consisting of four identical DACs, each driven by a different clock phase with a 90-degree phase shift between

them. To realize this architecture, a single DAC is partitioned into four sub-DACs. As shown in Figure 4.7, the 10-bit DAC is segmented into a 6-bit thermometer code and a 4-bit binary code. All thermometer bits, along with the two binary MSBs, correspond to at least four unit current each. This allows them to be evenly divided into four groups, each driven by one of the four clock phases.

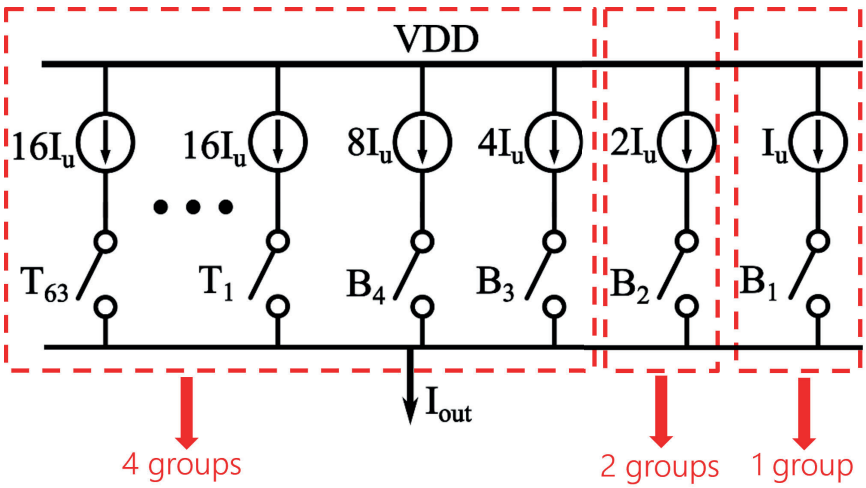


Figure 4.7: DAC partition for 4-fold interpolation

However, the two binary LSBs represent only two unit current and one unit current, respectively. These cannot be split evenly into four groups. As a result, the LSB (bit 0) is assigned to a single clock phase, while the next bit (bit 1) is divided into two groups and driven by two clock phases. Consequently, the DAC is partitioned into four sub-DACs with slightly different weights.

The output of this interpolation DAC therefore deviates slightly from an ideal 4-fold linear interpolation, leading to a small loss in resolution. Nonetheless, this implementation offers a simple and practical solution, and the resolution trade-off remains acceptable.

Chapter 5
Circuit Design

5.1 Current cell

Considering trade-off between linearity, area and complexity, 6 MSBs-4 LSBs segmentation is chosen for the 10-bit current steering DAC where the 6 MSBs are thermometer-weighted while the 4 LSBs are binary-weighted. And the 6 MSBs are segmented into two thermometer parts further, with 3 bits each. DEM logic is added in 3 MSBs. With the 3-3-4 segmentation, the maximum DNL in statistic appears at the transition: $0001111111 -> 0010000000$. The relationship between the mismatch ratio of the least significant bit current and DNL_{max} is

$$\sigma(DNL_{max}) = \sqrt{2^8 - 1} * \frac{\sigma Iu}{Iu} \tag{5.1}$$

Where Iu is the current of LSB. According to [4], the relationship between the mismatch ratio of least significant bit and INL is independent on segmentation and can be expressed by

$$\sigma(INL_{max}) = \sqrt{2^{N-2} - 1} * \frac{\sigma Iu}{Iu} \tag{5.2}$$

where N is the total bits of the DAC. The requirement on the DNL and INL is

$$3\sigma(INL_{max}) < 1LSB,\ 3\sigma(DNL_{max}) < 1LSB. \tag{5.3}$$

It can be derived that the requirement on the $\frac{\sigma Iu}{Iu}$ is

$$\frac{\sigma Iu}{Iu} < 2.1\% \tag{5.4}$$

With the requirement on $\frac{\sigma Iu}{Iu}$, the dimension of current source need to be sized properly to achieve low enough mismatch ratio. And with the design purpose of low power consumption, the unit current for LSB in binary part is selected to be 2uA initially. The total current of all current cells is 2.05mA, and the supply voltage for the DAC core is 0.83V.

PMOS is used for the design of current cell for its better matching and noise performance. Cascode transistor is added to improve the output impedance. Two different cascode structure of current cell is shown in Figure 5.1. The topology in Figure 5.1(a) has a transistor between current source transistor and switches while

the cascode transistor in Figure 5.1(b) is added at the output. Considering static linearity, both topology can provide high enough output impedance. However, the dynamic linearity is different between two topologies. In Figure 5.1(a), the drain of switch is connected with output node. The voltage variation of output node is large and code-dependent, which has significant impact on the switching activity. And it will induce distortion at the output signal. In the contrast, the drain voltage of switches in Figure 5.1(b) is more stable, and thus switches contribute less nonlinearity to output. Simulation results show that topology in Figure 5.1(b) gives better linearity to DAC and is selected eventually.

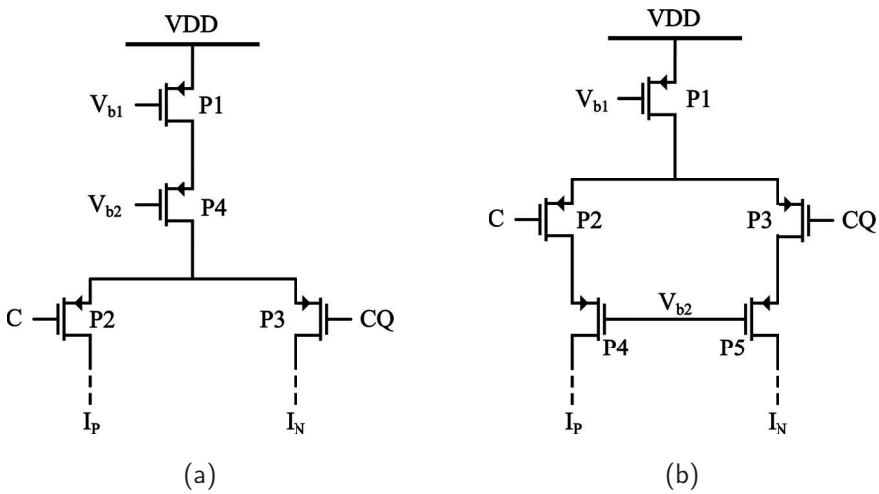


Figure 5.1: (a) cascode transistor between current source and switch (b) cascode transistor at output

The current source transistor is biased at 420mV while the cascode transistor is biased at 230mV. Current source transistor is sized as W = 800nm, L = $1\mu m$ to meet the requirement of mismatch. And it is decomposed into four transistors in series. The dimension of switches and cascode transistor is iterated to give better linearity. Sizing of the current cell is summarized in Table 5.1. Current cells in higher bit has the same topology but the number of transistors increases proportionally.

Table 5.1: Sizing of current cell.

	Current source	Cascode	Switch
W	800nm	600nm	120nm
L	$1\mu m$	100nm	40nm

5.2 Bias circuit

Current cell with bias circuit is shown in Figure 5.2. An external reference current of 20 μA is mirrored into a diode-connected PMOS transistor, which generate the bias voltages. Diode-connected PMOS transistors have the same W/L raio but different number of cells with current source transistor. They constitute a current mirror that copies the reference current to the current cell. The bias voltage for the cascode transistor is generated by mirroring a 20 µA current across an 11.5 $k\Omega$ resistor. Since variations in the cascode bias voltage have a negligible effect on the current value and noise performance, the main concern in the bias circuit design lies in the primary bias voltage.

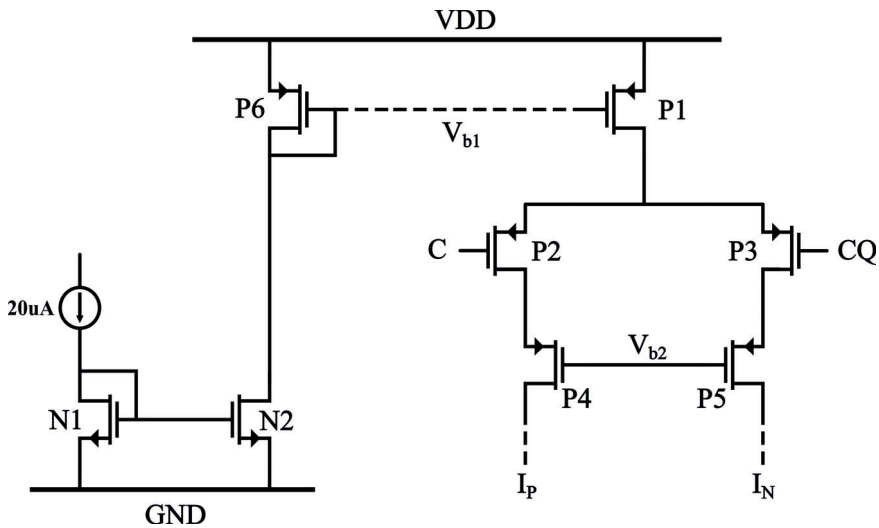


Figure 5.2: Current cell with bias circuit

One important design consideration of bias circuit is the source impedance. As the source impedance is not zero, a dip on common source node of switch pairs will be coupled to the bias node through parasitic capacitance of current source, contributing to disturbance on bias node. Since the bias circuit provides global bias voltage for all the current cells, the disturbance on bias node will be translated into disturbance on the output of each cell and induce dynamic nonlinearity. A bypass capacitor can be added in the bias node to filter out the disturbance. However, a large bypass capacitor is usually needed and the large area it occupies is unacceptable.

To attenuate disturbance on bias node, the strategy used in this design is trying to reduce the source impedance of bias circuit. The source impedance is mainly dependent on the gm of diode connected PMOS. PMOS is sized to be larger to lower the source impedance. The mirror ratio between diode-connected PMOS and LSB current source transistor is chosen to be 10:1. From the simulation result, the source impedance of bias circuit is about $3.72K\Omega$

Another significant issue on the design of bias circuit is the noise performance. The noise from the bias circuit will be amplified by the current cells and added

on the output of DAC. It's important to reduce the output noise of bias circuit to achieve high SNDR of the DAC. The N1 and N2 in bias circuit contributes most noise.

A common approach to attenuate the noise of a current mirror is to insert an RC filter between N1 and N2. However, in our case, a large load capacitance is already present due to the bias node being connected to all the current cells. Simulation results show that the equivalent load capacitance of the current source array is approximately 10 pF. Together with the source impedance, this forms an effective RC filter with a cutoff frequency of about $f_{-3dB} = 4.28$ MHz, which significantly attenuates flicker noise.

5.3 Master-Slave Latch

In each current cell, switch pair is preceded by a driver, which generates the switch control signal. A low-crossing point of positive/negative control signal is desirable to ensure switches don't turn off at the same time. It can lower the glitch power at the output and then improve the linearity. A latch shown in Figure 5.3 [3]can serve this purpose.

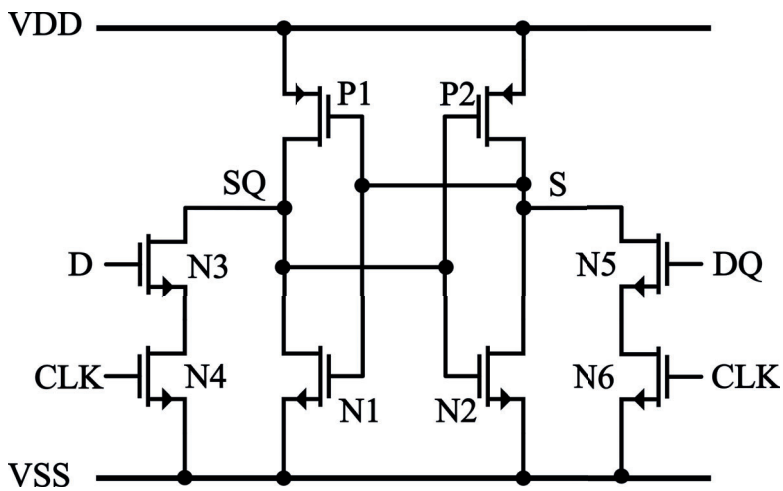


Figure 5.3: A latch driving the switching pair

In Figure 5.3, D and DQ are the input digital signals, while S and SQ are the differential outputs that drive the switches in the current cell. Transistors P1, P2, N1, and N2 form two head-to-tail connected inverters . When CLK is low, N4 and N6 are turned off, and the output state is latched, remaining unchanged regardless of the input. When CLK transitions high, N4 and N6 turn on, enabling the circuit to respond to input changes.

Assuming the SQ node is initially high and S is low, a rising edge on D turns on N3, pulling the SQ node low. Due to the positive feedback loop at the output, SQ is eventually driven to ground while S is pulled up to VDD. Since SQ starts to fall before S begins to rise, the crossing point of the two signals occurs below

VDD/2. The same mechanism applies in the opposite direction, where S is pulled low and SQ rises. This crossing level is primarily determined by the delay between the falling and rising edges of S and SQ.

The crossing point can be tuned by adjusting the transistor dimensions of N3, N4, N5, and N6. Increasing the W/L ratio allows the output node to be pulled down more quickly when N3 or N5 is turned on, increasing the delay between the falling and rising edges and resulting in a lower crossing point. A typical output waveform of the latch is shown in Figure 5.4, demonstrating full rail-to-rail swing and a crossing point around 80mV.

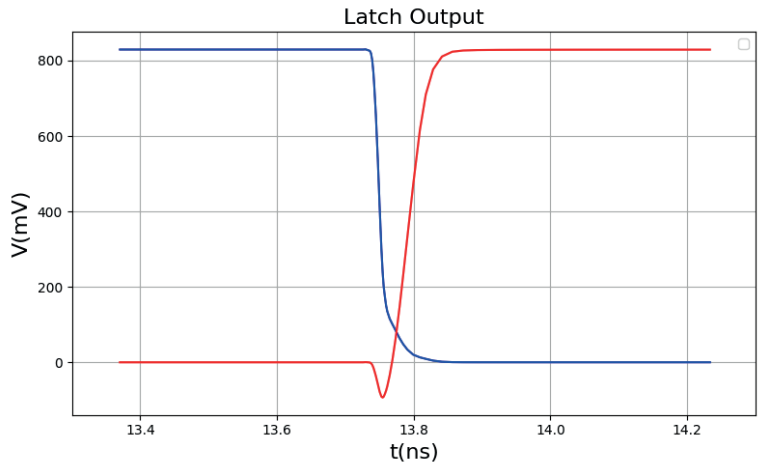


Figure 5.4: Output waveform of the latch

As discussed in Section 2.5, time skew from the input signal will induce distortion. To synchronize the input signal, two latches shown in Figure 5.3 are used to form a master-slave latch. Figure 5.5 is the block diagram of master-slave latch. When CLK is low, digital signal is passed to the output of the first latch. When CLK is high, the output of the first latch will keep unchanged and will be passed to the output of the second latch at rising edge of the clock. In this way, transition only happens in the rising edge of the clock and timing skew of input signal is reduced significantly.

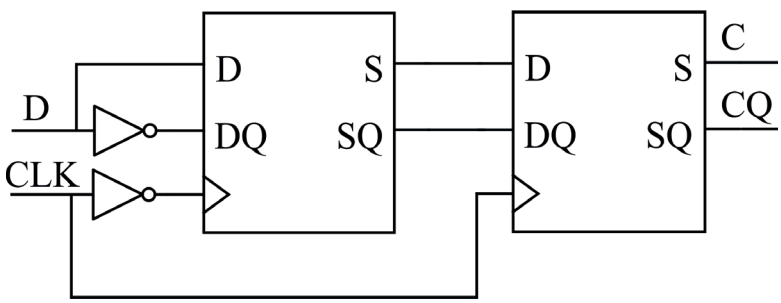


Figure 5.5: Master-slave Latch

5.4 Binary to Thermometer decoder

The 10-bit DAC is segmented in 3T-3T-4B scheme. There are two thermometer parts with 3 bits and binary to thermometer decoder is needed. Since the decoder for MSB thermometer part is implemented by DEM logic, only one decoder is required to convert 3 binary bits to 7 thermometer bits. Boolean equations between 3 bits binary code and 7 bits thermometer code is shown in Table 5.2. B_3 is the MSB of the bianry code and T_7 is the MSB of thermometer code.

To remove the redundant logic, the equations can be rearranged like in the Table 5.3. In this way, NOR and NAND gate is used rather than AND, OR, and the number of transistors can be reduced. The full binary to thermometer decoder design is shown in Figure 5.6.

Table 5.2: Boolean equations of decoder.

Output	Equation
$T1$	$B_1 + B_2 + B_3$
$T2$	$B_2 + B_3$
$T3$	$B_3 + (B_2 \cdot B_1)$
$T4$	B_3
$T5$	$B_3 \cdot (B_2 + B_1)$
$T6$	$B_3 \cdot B_2$
$T7$	$B_3 \cdot B_2 \cdot B_1$

Table 5.3: New Boolean equations of decoder.

Output	Equation
$T1$	$\overline{\overline{B_1} \cdot \overline{B_2} \cdot \overline{B_3}}$
$T2$	$\overline{\overline{B_2} \cdot \overline{B_3}}$
$T3$	$\overline{\overline{B_1 \cdot B_2} \cdot \overline{B_3}}$
$T4$	$\overline{\overline{B_3}}$
$T5$	$\overline{\overline{B_1 + B_2} + \overline{B_3}}$
$T6$	$\overline{\overline{B_2} + \overline{B_3}}$
$T7$	$\overline{\overline{B_1 \cdot B_2} + \overline{B_3}}$

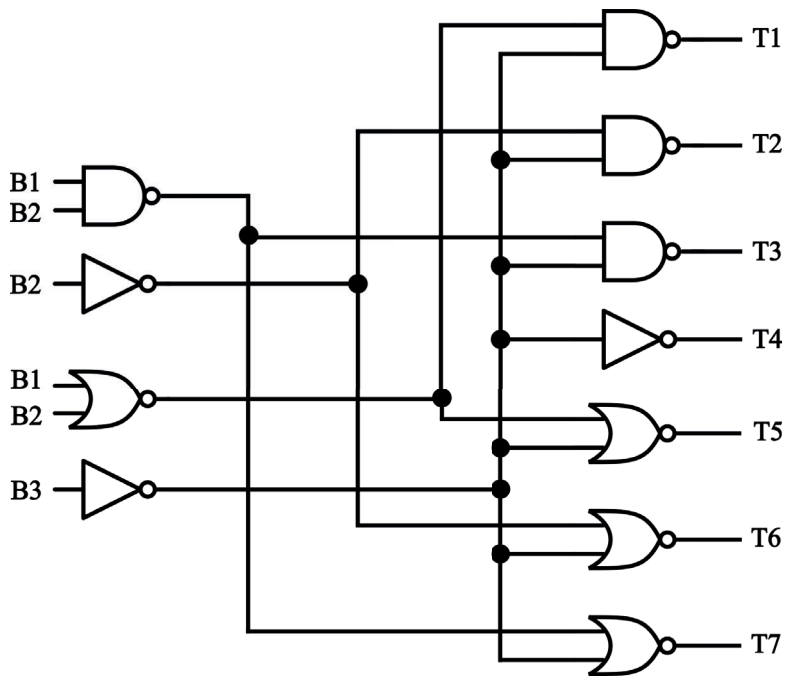


Figure 5.6: Binary to thermometer decoder

Chapter 6

Simulation Results and Discussion

6.1 DAC Testbench

As discussed in Chapter 3, 4-fold linear interpolation is the optimal choice that gives filtering function needed while maintains acceptable complexity. The testbench to simulate the performance of the linear interpolation DAC is shown in Figure 6.1

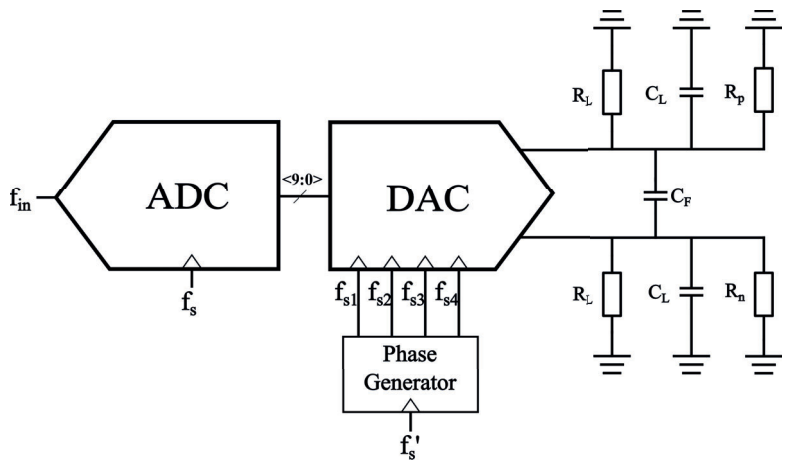


Figure 6.1: Testbench of linear interpolation DAC.

At the sampling rate of f_s, a 10-bit ADC model converts sinusoid input signal with frequency of f_{in} into digital code , which serves as the digital input of the DAC. A phase generator model generates four clock phases with 90° phase shift between each other. Four clock phases will drive four groups of cells in DAC, achieving 4-fold linear interpolation. R_p, R_n is the resistor load at differential output end and has a resistance of 297Ω. R_L and C_L model the resistive and capacitive load at next stage and their value are $750\Omega, 700fF$ respectively. Since our interpolation DAC is expected to replace the existing DAC+LPF, R_L and C_L are modeling the load of LPF. A $6pF$ capacitor C_F is added to form a RC filter with resistor load, which can provide extra filtering. Samping frequency fs is 128MHz while fin is set to $\frac{81}{1024} * fs = 10.125MHz$, which is close to 10MHZ

and meets the coherent sampling condition.

6.2 Performance of the DAC without linear interpolation

Before exploring the performance of the linear interpolation DAC, the DAC without interpolation is characterized. C_F in Figure 6.1 is dismissed and four clocks driving the DAC are set to have the same phase. Transient simulation without noise is performed and the differential output waveform of the DAC is shown in Figure 6.2.

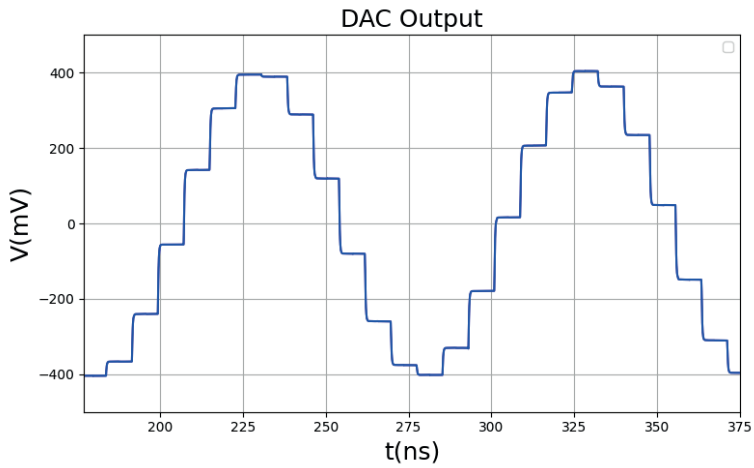


Figure 6.2: Differential output waveform of the DAC without interpolation.

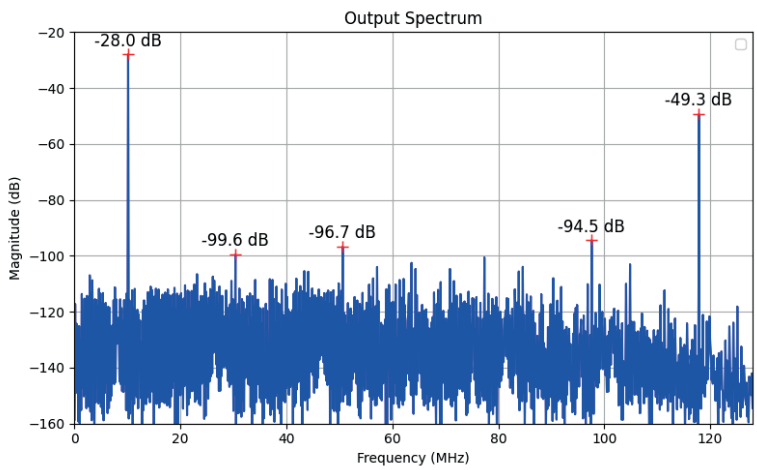


Figure 6.3: Output spectrum of the DAC without interpolation.

The full scale of differential output is 800mV peak-to-peak. And thanks to the technique used in the latch and current cell, the glitch at the output is ignorable. The spectrum of the output within $fs = 128MHz$ is displayed in Figure 6.3. Signal tone, third-order harmonic, fifth-order harmonic and image are marked in the spectrum. Referred to signal tone level, image at 117.825 MHz is -21.3dBc. The fifth-order harmonic is the highest harmonic within the Nyquist band and the level of it is -68.8dBc, which decides the value of SFDR. It's noticeable that there is a image of third-order harmonic in the spectrum, which has higher power level than third-order harmonic. Considering the noise and harmonic within Nyquist band, the ENOB and SFDR of the DAC is 9.8 bits and 68.8 dB respectively.

To investigate how the performance changes with the input frequency, fin is swept from 6.375MHz to 50.125MHz while fs stays 128MHz. Figure 6.4 plots the relationship between ENOB, SFDR and input frequency. Both ENOB and SFDR degrade significantly when the input frequency exceeds 20MHz. For instance, the ENOB decreases from 9.66 bits to 9.22 bits when input frequency increases from 20MHz to 25MHz.

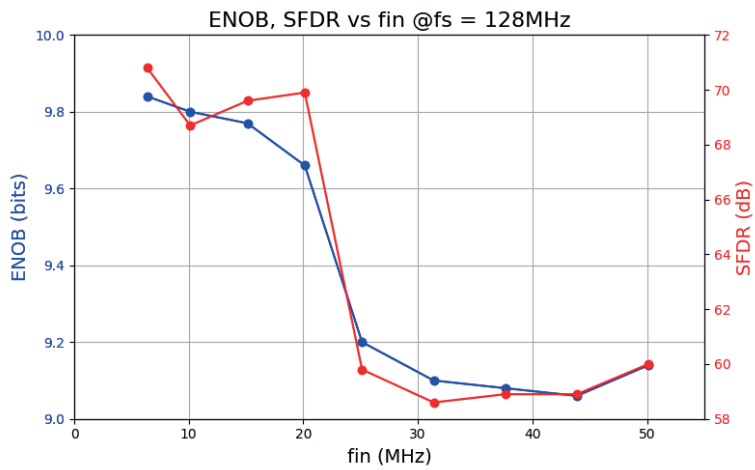


Figure 6.4: ENOB, SFDR vs input frequency.

The sampling frequency fs is swept as well to explore how performance changes with fs while fin stays 10MHz. The relationship between ENOB, SFDR and sampling rate is displayed in Figure 6.5. The performance of the DAC shows high sensitivity to the sampling rate. When fs increasing from 128MHz to 768MHz, ENOB degrades from 9.8 bits to 8.16 bits. The severe performance degradation at high sampling rate can comes from higher distortion induced by switching activity as discussed in Chapter 2.

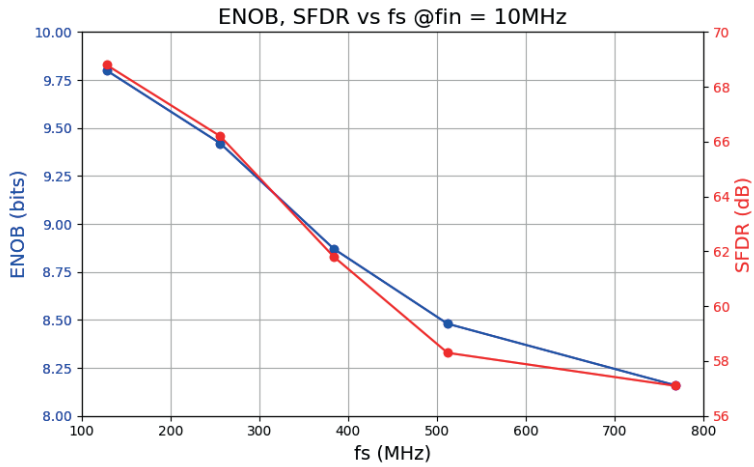


Figure 6.5: ENOB, SFDR vs sampling rate.

6.3 Performance of the linear interpolation DAC

6.3.1 Levels of harmonics and images in interpolation DAC output

According to Section 6.1, the linear interpolation DAC is characterized through transient simulation and transient noise isn't added in the beginning. The output waveform of the interpolation DAC is shown in Figure 6.6. Compared with the output of the DAC without interpolation, the waveform displayed in Figure 6.6 is smoothed by linear interpolation and RC filter.

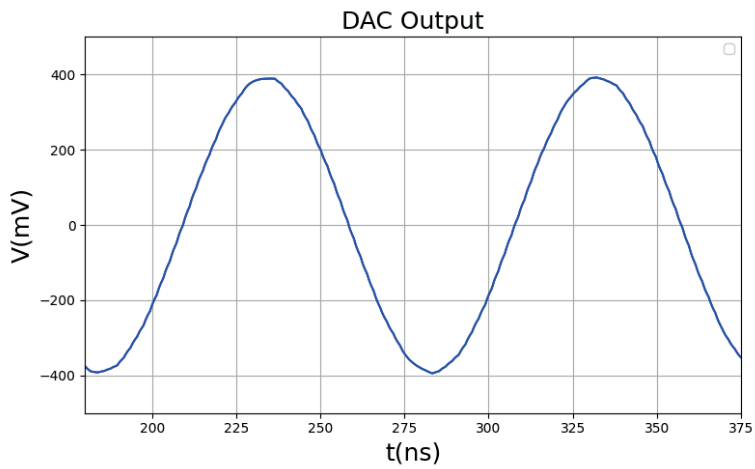


Figure 6.6: Differential output waveform of the interpolation DAC.

Figure 6.7 shows the output spectrum of interpolation DAC within 128MHz. Compared with Figure 6.3, the image at 117.825 MHz dropped from -21.3dBc to

-48.6dBc due to the filtering from interpolation and RC filter. And the image of third-order harmonic is attenuated to a negligible power level. Considering the noise and harmonic within Nyquist band, ENOB and SFDR of the DAC is 9.98 bits and 72.5 dB respectively.

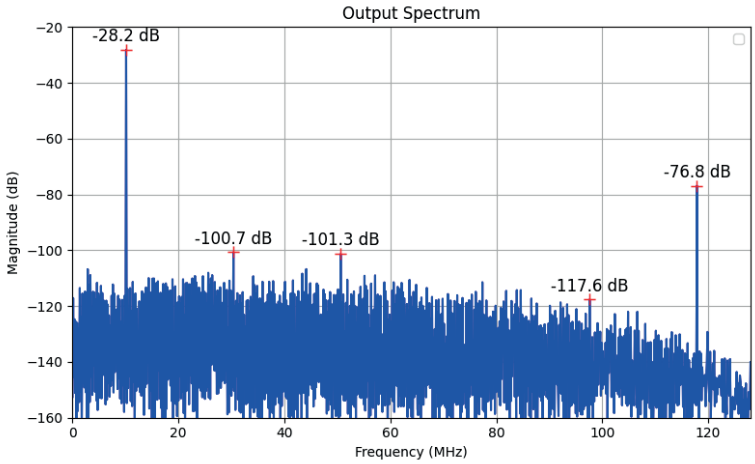


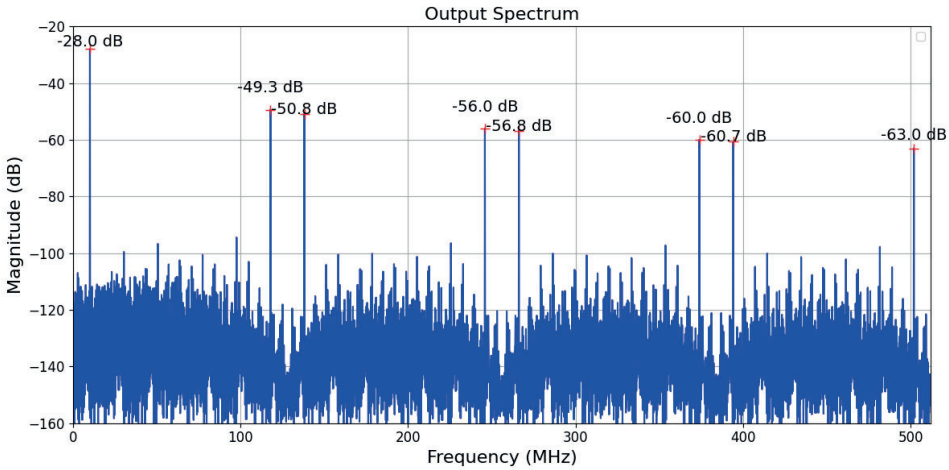
Figure 6.7: Output spectrum of the interpolation DAC.

To further compare the image power levels in DAC outputs with and without interpolation, Figure 6.8 presents the output spectrum of both DACs within the range of $4f_s$. A significant reduction in image levels is observed in Figure 6.8(b) compared to Figure 6.8(a). The image levels at 117.875MHz, 245.875MHz, 373.875MHz, and 501.875MHz are summarized in Table 6.1. Notably, the dominant image at 117.875MHz decreases from –21.3dBc to –48.6dBc with 4× interpolation. This suppression is sufficient to meet typical spectral mask requirements

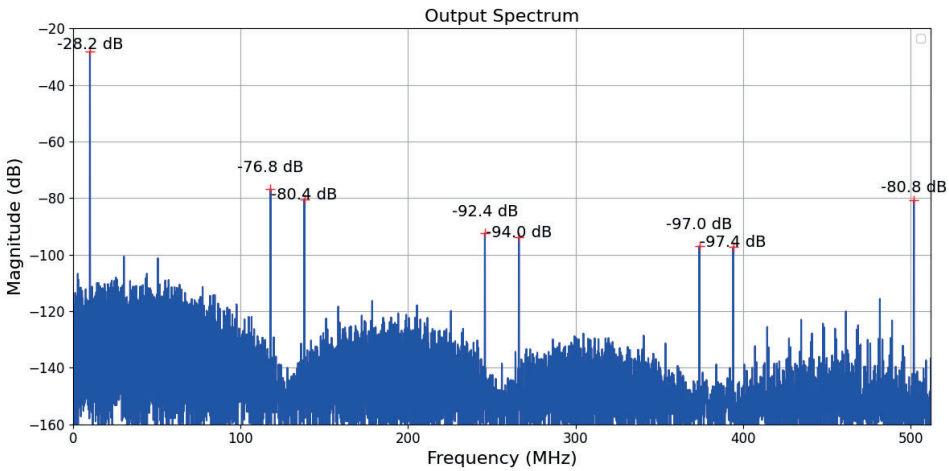
Table 6.1: Image levels with and without interpolation.

Images@	**No interpolation**	**4× Interpolation**	Drop
117.875MHz	-21.3 dBc	-48.6 dBc	27.3 dB
245.875MHz	-28 dBc	-64.2 dBc	36.2 dB
373.875MHz	-32 dBc	-68.8 dBc	36.8 dB
501.875MHz	-35 dBc	-52.6 dBc	27.6 dB

To demonstrate that the output spectrum of the interpolation DAC meets the spectral mask requirements, a baseband signal is used as the digital stimulus. The baseband signal is modulated using a 1024-QAM scheme, with a sampling rate of 128 MHz and a 20 MHz bandwidth packet. The output spectrum of the interpolation DAC, along with the spectral mask, is shown in Figure 6.9. As illustrated, output spectrum of the DAC remains within the bounds of the spectral mask, with image components well below the mask floor. Now the output of the interpolation DAC can be mixed with LO signal without the filtering of LPF.



(a)



(b)

Figure 6.8: (a) Output spectrum of the DAC without interpolation. (b) Output spectrum of the DAC with $4\times$ interpolation.

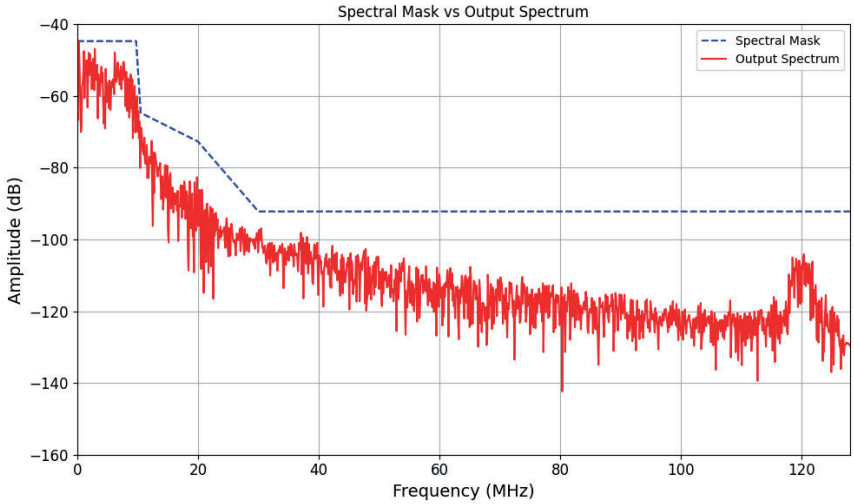


Figure 6.9: Spectral Mask vs Output spectrum.

6.3.2 Impact of clock jitter

Random clock jitter would add white noise in the output spectrum of the DAC, degrading ENOB. To investigate the impact of jitter on ENOB, gaussian jitter is added in f_s' shown in Figure 6.1. The relationship between ENOB and jitter is plotted in Figure 6.10. As the standard deviation of the jitter increases from zero to 31.25ps, ENOB drops from 9.98 bits to 8.80 bits. And the ENOB is around 9.5 bits when jitter is 15ps. It means that the clock jitter should be lower than 15ps to ensure the noise from jitter less than quantization noise.

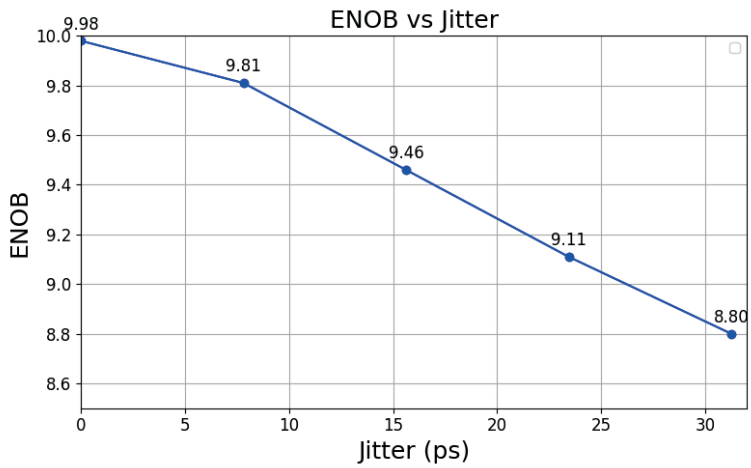


Figure 6.10: ENOB vs Jitter.

6.3.3 Transient noise analysis

There are many noise sources in current steering DAC such as bias circuit, load resistor and current source. And noise from bias circuit deserves special attention. Bias circuit provides a global bias for current source array. The noise from bias circuit will add a voltage noise at bias node and be converted into current noise at output current. So the noise at positive and negative output node is proportional to the output current. In other words, the noise at output is code-dependent. Since the output is differential, noise at differential output is zero at mid-scale input and reach maximum at zero or full scale input. To investigate the relationship between the noise power at the differential output and the bias node, a brief analysis is presented below.

Assuming the noise at bias node is $v_{n,bias}(t)$, it's amplified to maximum value $v_{n,max}(t)$ at output when input is at full scale. The relationship between $v_{n,max}(t)$ and $v_{n,bias}(t)$ is

$$v_{n,max}(t) = N \cdot g_m \cdot R_L \cdot v_{n,bias}(t) \tag{6.1}$$

Where g_m is the transconductance of LSB current cell and N is the total number of LSB cells. Assuming differential output is V_o at full-scale input, the general expression of the voltage at positive and negative output node can be written as

$$\begin{aligned} V_p'(t) &= V_p(t) + \frac{V_p(t)}{V_o} \cdot v_{n,\max}(t) \\ V_n'(t) &= V_n(t) + \frac{V_n(t)}{V_o} \cdot v_{n,\max}(t) \end{aligned} \tag{6.2}$$

Where $V_p(t), V_n(t)$ is output voltage without noise and $V_p'(t), V_n'(t)$ is output voltage with noise. The summary of V_p and V_n is V_o. The differential output can be written as

$$V_{out}(t)' = V_p(t) - V_n(t) + \frac{V_p(t) - V_n(t)}{V_o} \cdot v_{n,\max}(t) \tag{6.3}$$

The second term in (6.3) is the noise term at output and the power of it can be written as

$$P_{n,out} = E\left\{\left[\frac{V_p(t) - V_n(t)}{V_o} \cdot v_{n,\max}(t)\right]^2\right\} \tag{6.4}$$

Where E denotes the expectation. Since $V_p(t) - V_n(t)$ and $v_{n,\max}(t)$ are uncorrelated, (6.4) can be written as

$$P_{n,out} = E\left\{\left[\frac{V_p(t) - V_n(t)}{V_o} \cdot v_{n,\max}(t)\right]^2\right\} = \frac{E\left\{\left[\frac{V_p(t) - V_n(t)}{V_o}\right]^2\right\}}{V_o^2} \cdot E\left[v_{n,\max}^2(t)\right] \tag{6.5}$$

In the interpolation DAC, $V_p(t) - V_n(t)$ can be approximated as a sinusoidal signal with amplitude of V_o. Equation (6.5) can be rewritten as

$$P_{n,out} = \frac{1}{2} P(v_{n,\max}) \tag{6.6}$$

Substituting(6.2), (6.6) can be rewritten further as

$$P_{,outn} = \frac{1}{2} \cdot (N \cdot g_m \cdot R_L)^2 \cdot P(v_{n,\text{bias}}) \tag{6.7}$$

Where $P(v_{n,\text{bias}})$ is noise power at the bias node.(6.7) describes the relationship between the noise power at output and the noise power at bias node.

To analyze the noise contributions from different circuit components, transient noise analysis is enabled in the transient simulation with $f_{\max}$ set to 50GHz. Two simulation runs are conducted to quantify the noise originating from different sources. In the first run, only the noise from the bias circuit is enabled. In the second run, all noise sources are turned on. The resulting ENOB and SFDR from both simulation are summarized in Table 6.2.

Table 6.2: Transient noise simulation results

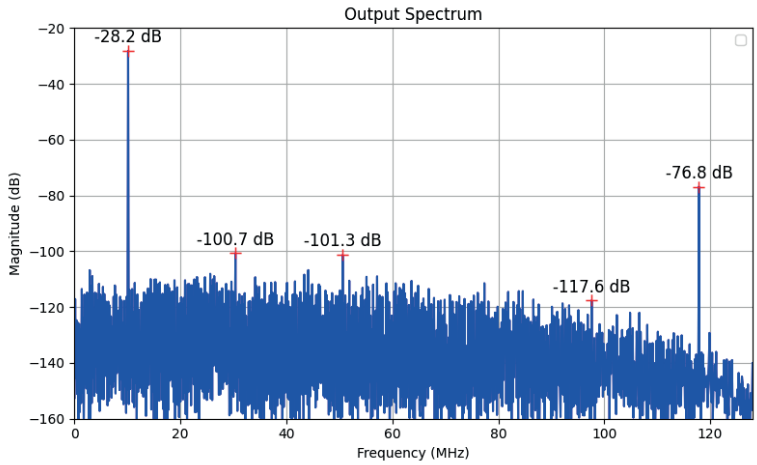
	ENOB(bits)	**SFDR(dB)**
Noise contribution from Bias	9.66	71.0
Noise contribution from all circuits	9.63	72.7

When only noise contribution from bias circuit is activated, ENOB decreases from 9.98 bits to 9.66 bits. And ENOB stays almost the same when other noise contribution is enabled. It indicates that the noise from bias circuit is the dominant noise source in the DAC circuit. Referred to the definition of ENOB and SNDR, the noise power contributed from bias circuit can be calculated.

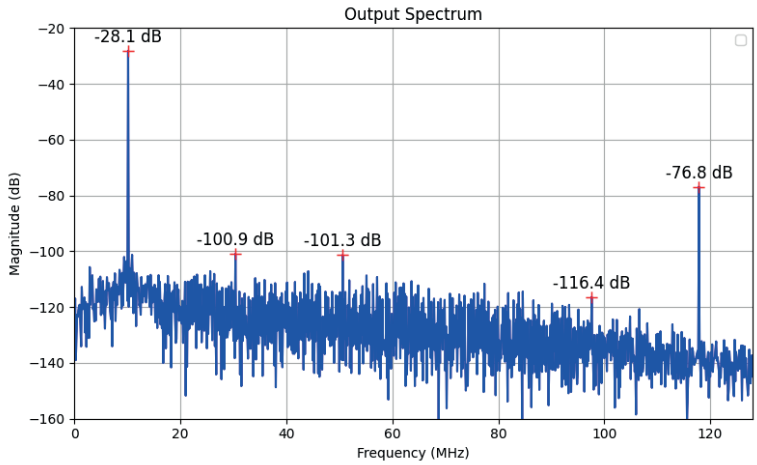
$$P_{n,bias} = 0.56 \cdot P_Q \tag{6.8}$$

Where P_Q is the quantization noise power.

Figure 6.11 compared the output spectrum with and without transient noise. It can found that the noise floor near signal tone is higher in the spectrum with transient noise. It can be explained by equation(6.3). The noise term at the output $\frac{V_p(t)-V_n(t)}{V_o} \cdot v_{n,\max}(t)$ is the multiplication of signal and noise from bias in time domain. In frequency domain, it is the convolution of signal spectrum and bias noise spectrum. Due to flicker noise from bias circuit, the spectrum of bias noise is higher in lower frequency. Convoluted with signal spectrum, it contributes to the higher noise floor near signal tone in Figure 6.11(b).



(a)



(b)

Figure 6.11: (a) Output spectrum of the interpolation DAC without transient noise. (b) Output spectrum of the interpolation DAC with transient noise

6.3.4 Monte Carlo Simulation

Monte Carlo simulation is performed to evaluate the impact of circuit mismatch on the performance. Mismatch can come from the current source, latches and the load resistors. Initially, the bias circuit is replaced with an analogLib model to eliminate its influence. Monte Carlo simulation results with 50 points are listed in Table 6.3. Considering the mean value, ENOB drops to 9.77 bits and SFDR drops to 68.7 dB due to the circuit mismatch .

Table 6.3: Monte Carlo simulation results

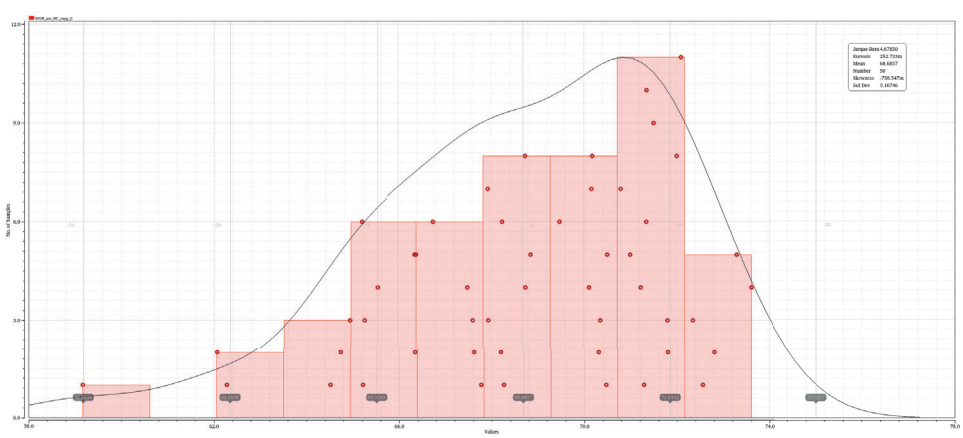
	min	**max**	**mean**	**Std dev**
ENOB(bits)	9.15	9.98	9.77	0.16
SFDR(dB)	59.17	73.6	68.7	3.2

Tree-structured DEM is used to scramble the distortion induced by the current cell mismatch. Monte Carlo simulation results with DEM enabled is listed in Table 6.4. Compared with results in Table 6.3, mean value of SFDR increases from 68.7dB to 73.2dB, demonstrating the effectiveness of DEM in reducing distortion. And the histogram of SFDR in two simulation run is shown in Figure 6.12.

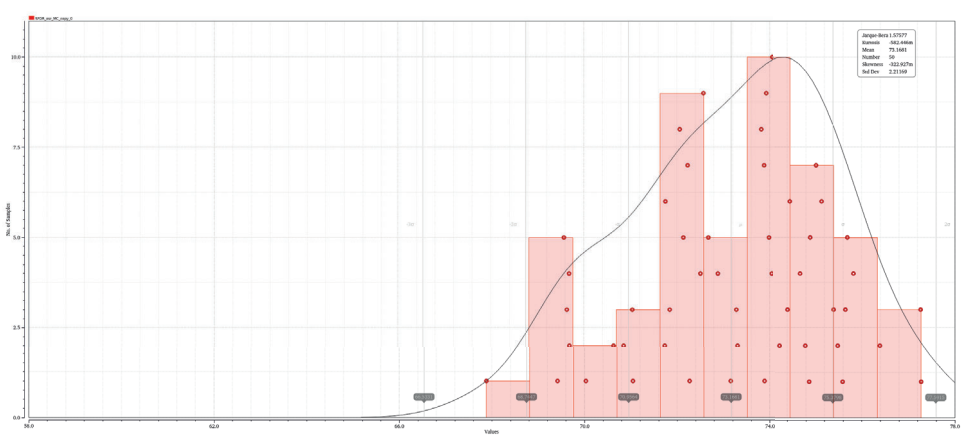
Table 6.4: Monte Carlo simulation results with DEM enabled

	min	**max**	**mean**	**Std dev**
ENOB(bits)	9.34	9.90	9.64	0.13
SFDR(dB)	67.9	77.3	73.2	2.2

The output spectrum of one Monte Carlo point with and without DEM is displayed in Figure 6.13. When DEM is activated, the third harmonic in the spectrum drops from -96.5dB to -105.2dB. It demonstrates that the harmonics induced from current cell mismatch are scrambled into broadband noise.

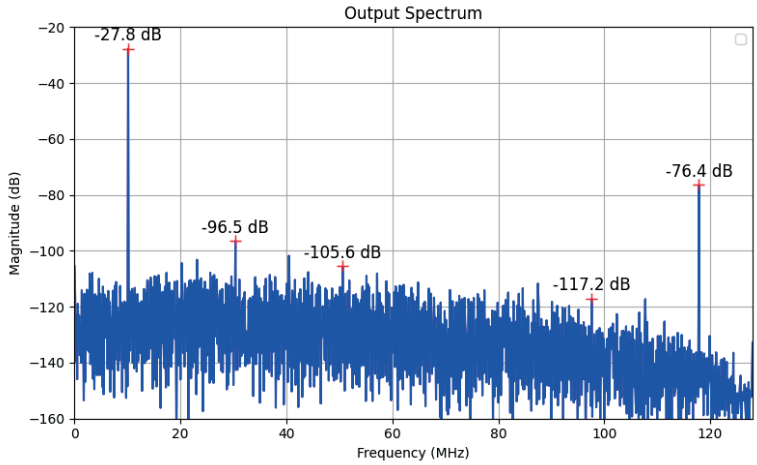


(a)

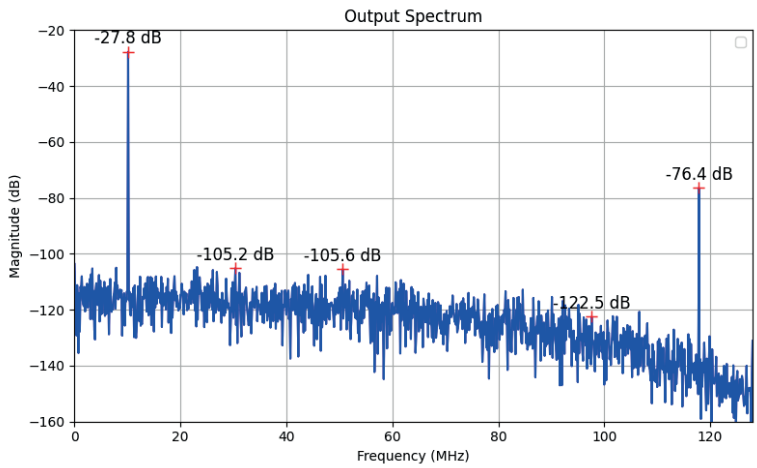


(b)

Figure 6.12: (a) Histogram of SFDR without DEM. (b) Histogram of SFDR with DEM



(a)



(b)

Figure 6.13: (a) Output spectrum without DEM. (b) Output spectrum with DEM

Then analoglib model for bias is replaced by bias circuit and mismatch from bias circuit is activated. The Monte Carlo simulation results with and without DEM are listed in Table 6.5 and Table 6.6 respectively. DEM continues to effectively improve the minimum and standard deviation of SFDR; however, its impact on the mean SFDR is negligible. This indicates that mismatch from bias circuit is the primary bottleneck limiting SFDR.

Table 6.5: Monte Carlo simulation results with bias circuit

	min	**max**	**mean**	**Std dev**
ENOB(bits)	9.08	9.92	9.69	0.14
SFDR(dB)	58.8	72.5	68.1	2.5

Table 6.6: Monte Carlo simulation results with bias circuit and DEM

	min	**max**	**mean**	**Std dev**
ENOB(bits)	9.32	9.78	9.57	0.11
SFDR(dB)	65.8	71.2	68.7	1.1

6.3.5 Performance across Process, Voltage, Temperature(PVT)

To evaluate the robustness of the interpolation DAC across process, voltage, and temperature (PVT) variations, transient noise simulations are performed under different process corners, supply voltages, and temperatures. The resulting SFDR and ENOB values across PVT conditions are shown in Figure 6.14 and Figure 6.15. Both ENOB and SFDR degrade with increasing temperature, particularly beyond $90°C$. As temperature rises, the load resistance increases, leading to a larger output swing. This increased swing pushes the current source transistors closer to the triode region, thereby introducing more nonlinearity. Additionally, the PVT simulation results indicate that both ENOB and SFDR improve with higher supply voltage, due to the increased headroom available for the current source transistors. ENOB has the worst value of 9.01 bits at ss process corner, 0.80V supply and $125°C$. And SFDR has its lowest value of 59.4dB at fs process corner, 0.80V supply and $125°C$.

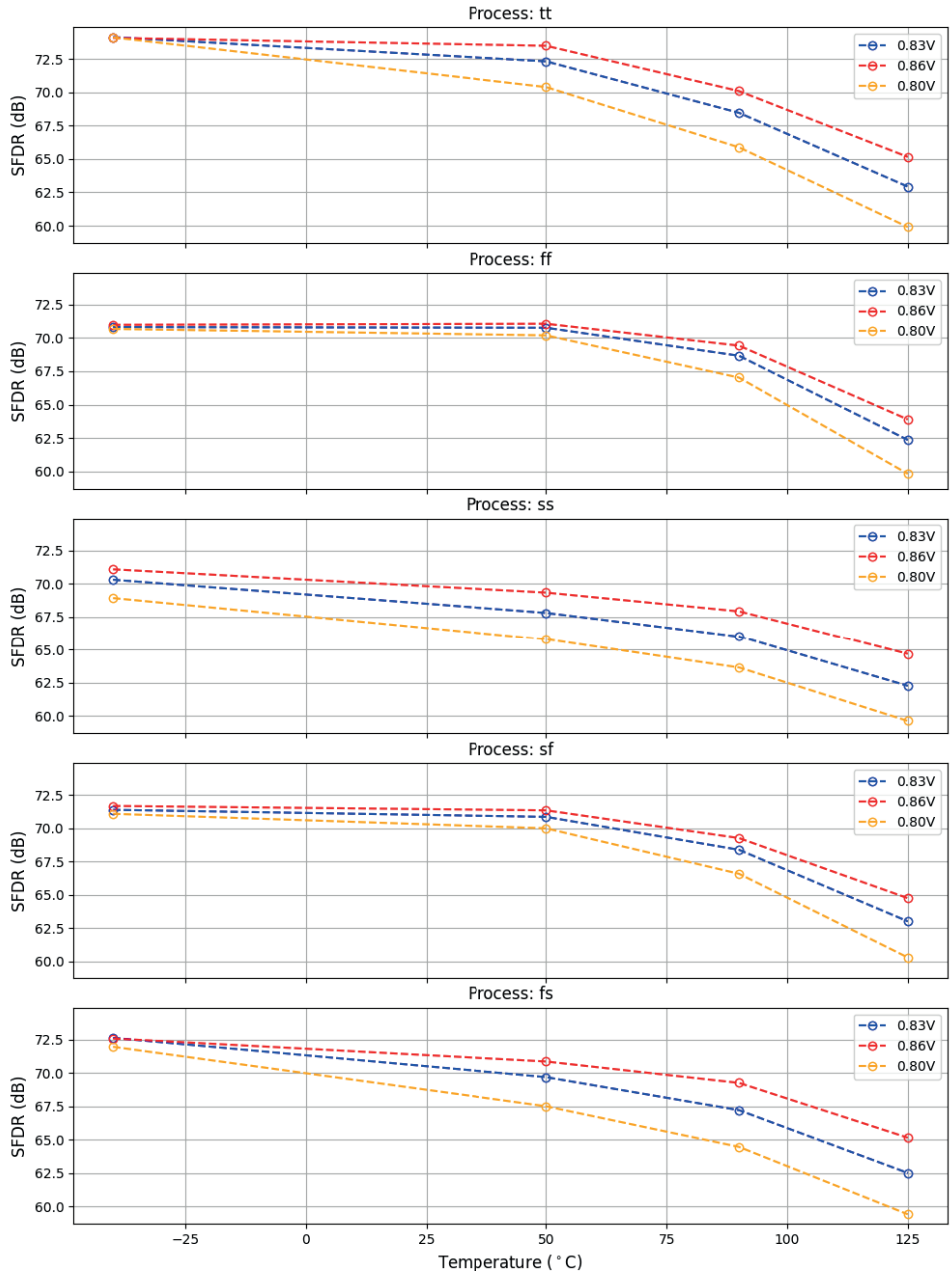


Figure 6.14: SFDR over PVT.

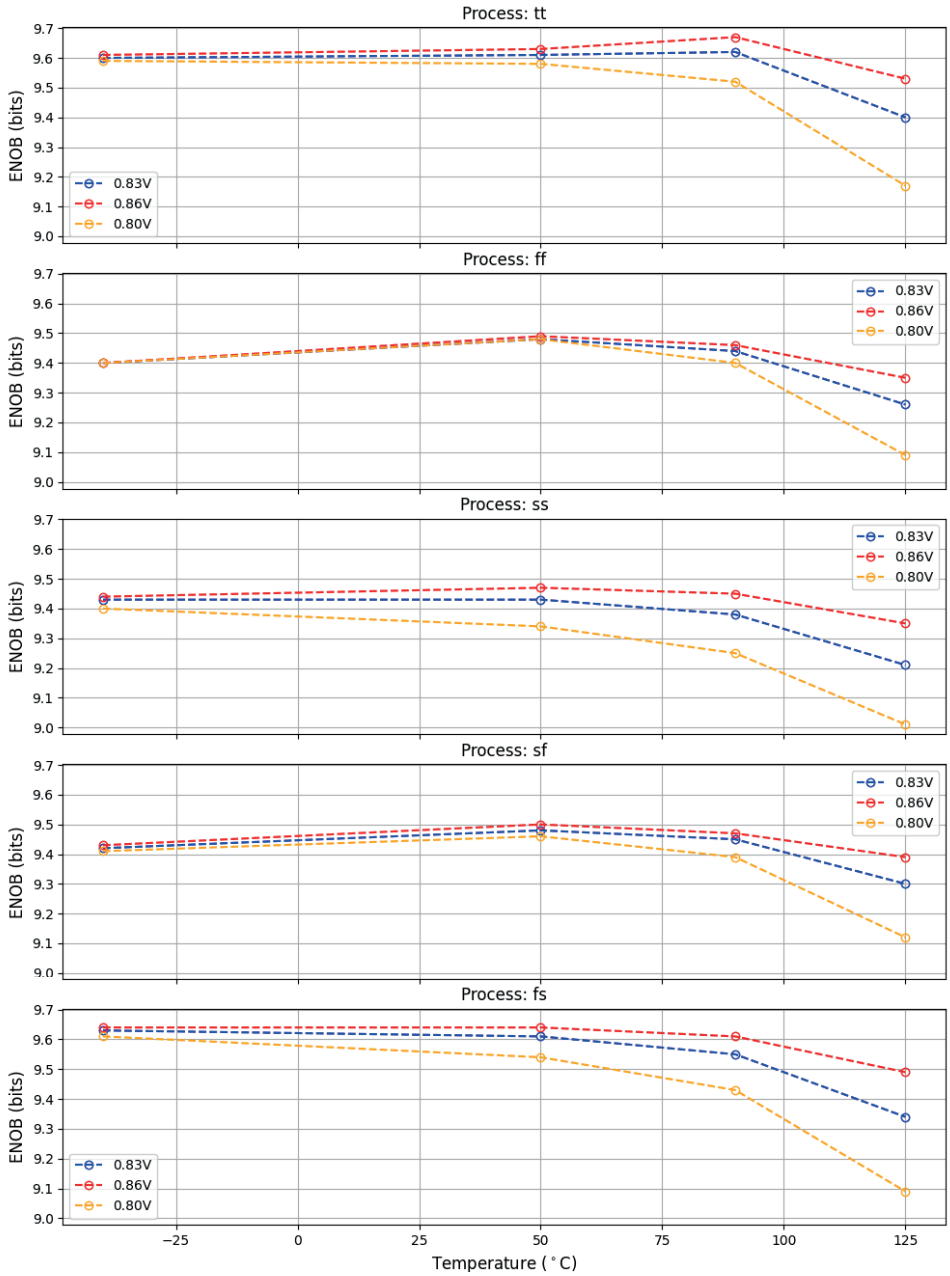


Figure 6.15: ENOB over PVT.

In addition to the standard PVT corners, transient noise simulations are also performed under four specific corners that account for variations in resistor, capacitor, and noise process parameters. The resulting SFDR and ENOB are listed in Table 6.7,where Rh,Nh,Ch,Vh,Th means high resistance, high noise, high cap, high supply voltage, high temperature respectively. High resistance and high temperature lead to higher out swing, contributing to worst ENOB and SFDR.

Table 6.7: Performance across 4 special corners

Corners	ENOB(bits)	SFDR(dB)
ff,Rl,Nh,Cl,Vh,Tl	9.31	69.23
ff,Rl,Nh,Cl,Vh,Th	9.45	65
ss,Rh,Nh,Ch,Vl,Tl	9.41	65.6
ss,Rh,Nh,Ch,Vl,Th	8.58	55

6.4 Performance summary

The operating conditions and performance of the interpolation DAC used in this thesis are summarized in Table 6.8. With a sampling rate of 128 MHz and an input frequency of 10 MHz, the DAC achieves an SFDR of 72.7 dBc and an ENOB of 9.63 bits, demonstrating excellent linearity. More importantly, the highest image level is only –48.5 dBc, fully satisfying the Wi-Fi 6 spectral mask requirements. In addition, the DAC core consumes an average current of just 2.13 mA at a supply voltage of 0.83 V, highlighting its low power consumption.

Table 6.8: Performance Summary

	This work
$Resolution(bits)$	10
$Process$	22nm FDSOI
$Supply(V)$	0.83
$f_s(MS/s)$	128
$f_{in}(MHz)$	10
$I_{avarage}(mA)$	2.13
$P_{core}(mW)$	1.77
$Core\ area(mm^2)$	0.028
$ENOB(bits)$	9.63
$SFDR(dB)$	72.7
$Image_{highest}(dBc)$	-48.5

Chapter 7

Conclusion and future work

7.1 Conclusion

The 10-bit linear interpolation DAC in this thesis employs a current-steering architecture to support medium high-speed operation. The cascode current source, combined with a rail-to-rail latch driver used in this design, manages to lower power consumption while maintain good linearity. And the tree-structure DEM employed in this design demonstrates its effectiveness in mitigating nonlinearity caused by current source mismatch. Operating at a sampling rate of 128 MHz with a 10 MHz input signal, the proposed DAC achieves an ENOB of 9.66 bits and a spurious-free SFDR of 72.7 dB, demonstrating excellent linearity.

Moreover, with linear interpolation technique proposed in this design, images at the DAC output are suppressed from -21.3 dBc to -48.6 dBc, meeting the requirements of the Wi-Fi 6 spectral mask. This enables the elimination of the conventional low-pass filter following the DAC, offering a more power-efficient and area-efficient transmitter solution.

However, simulation results also indicate that there is room for improvement in the design. The linearity and noise performance are primarily limited by the bias circuit, and the overall performance shows significant degradation under certain conditions, particularly at high temperature.

7.2 Future work

This design presents a promising solution for Wi-Fi 6 transmitter applications. However, several aspects deserve further investigation and optimization:

Phase Generator: The four clock phases required by the linear interpolation DAC are currently generated using a Verilog-A model, which does not account for non-ideal effects. In a practical implementation, a frequency divider with a 256 MHz input clock can be used to generate four 128 MHz clock phases. However, such dividers may introduce additional jitter and phase mismatches, which could have impact on the DAC.

Bias Circuit: Simulation results indicate that the bias circuit is the dominant contributor to output noise. Furthermore, after DEM effectively mitigates current

source mismatch, the bias circuit becomes the primary limitation on SFDR. Therefore, optimizing the bias circuit is essential to improving both noise performance and linearity.

Robustness Across Corners: The DAC exhibits significant linearity degradation under high-temperature (125°C) and high-resistance process conditions, primarily due to increased output swing. To address this, tunability can be introduced in the load resistance and output current, allowing dynamic adjustment of output swing and better corner robustness.

Tree-Structured DEM: In this design, the tree-structured DEM is implemented in Verilog code. For future work, it can be synthesized and implemented in CMOS digital circuit, enabling accurate estimation of circuit complexity and power consumption.

References

[1] S. Luschas, R. Schreier and Hae-Seung Lee, "Radio frequency digital-to-analog converter," in IEEE Journal of Solid-State Circuits, vol. 39, no. 9, pp. 1462-1467, Sept. 2004, doi: 10.1109/JSSC.2004.829377.

[2] P. Eloranta, P. Seppinen, S. Kallioinen, T. Saarela and A. Parssinen, "A Multimode Transmitter in 0.13 m CMOS Using Direct-Digital RF Modulator," in IEEE Journal of Solid-State Circuits, vol. 42, no. 12, pp. 2774-2784, Dec. 2007, doi: 10.1109/JSSC.2007.908749.

[3] Franco Maloberti. Data Converters. Springer US, 2007.

[4] Chi-Hung Lin and K. Bult, "A 10-b, 500-MSample/s CMOS DAC in 0.6 mm/sup 2/," in IEEE Journal of Solid-State Circuits, vol. 33, no. 12, pp. 1948-1958, Dec. 1998, doi: 10.1109/4.735535.

[5] G. A. M. Van Der Plas, J. Vandenbussche, W. Sansen, M. S. J. Steyaert and G. G. E. Gielen, "A 14-bit intrinsic accuracy Q/sup 2/ random walk CMOS DAC," in IEEE Journal of Solid-State Circuits, vol. 34, no. 12, pp. 1708-1718, Dec. 1999, doi: 10.1109/4.808896.

[6] M. J. M. Pelgrom, A. C. J. Duinmaijer and A. P. G. Welbers, "Matching properties of MOS transistors," in IEEE Journal of Solid-State Circuits, vol. 24, no. 5, pp. 1433-1439, Oct. 1989, doi: 10.1109/JSSC.1989.572629.

[7] C. -H. Lin et al., "A 12 bit 2.9 GS/s DAC With IM3 60 dBc Beyond 1 GHz in 65 nm CMOS," in IEEE Journal of Solid-State Circuits, vol. 44, no. 12, pp. 3285-3293, Dec. 2009, doi: 10.1109/JSSC.2009.2032624.

[8] Sungkyung Park, Gyudong Kim, Sin-Chong Park and Wonchan Kim, "A digital-to-analog converter based on differential-quad switching," in IEEE Journal of Solid-State Circuits, vol. 37, no. 10, pp. 1335-1338, Oct. 2002, doi: 10.1109/JSSC.2002.803056.

[9] G. Manganaro, "An Introduction to High Data Rate Current-Steering Nyquist DACs: Fasten your seat belts," in IEEE Solid-State Circuits Magazine, vol. 14, no. 3, pp. 24-40, Summer 2022, doi: 10.1109/MSSC.2022.3175676.

[10] Yonghua Cong and R. L. Geiger, "A 1.5-V 14-bit 100-MS/s self-calibrated DAC," in IEEE Journal of Solid-State Circuits, vol. 38, no. 12, pp. 2051-2060, Dec. 2003, doi: 10.1109/JSSC.2003.819163.

[11] G. Manganaro, “Analog calibration of a current source array at low supply voltage,” U.S. Patent 7 161 412, Jan. 9, 2007.

[12] D. J. Stoops, J. Kuo, P. J. Hurst, B. C. Levy and S. H. Lewis, "Digital Background Calibration of a Split Current-Steering DAC," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 66, no. 8, pp. 2854-2864, Aug. 2019, doi: 10.1109/TCSI.2019.2901626.

[13] R. T. Baird and T. S. Fiez, "Linearity enhancement of multibit /spl Delta//spl Sigma/ A/D and D/A converters using data weighted averaging," in IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, vol. 42, no. 12, pp. 753-762, Dec. 1995, doi: 10.1109/82.476173.

[14] I. Galton, "Spectral shaping of circuit errors in digital-to-analog converters," in IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, vol. 44, no. 10, pp. 808-817, Oct. 1997, doi: 10.1109/82.633435.

[15] K. L. Chan, N. Rakuljic and I. Galton, "Segmented Dynamic Element Matching for High-Resolution Digital-to-Analog Conversion," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 55, no. 11, pp. 3383-3392, Dec. 2008, doi: 10.1109/TCSI.2008.2001757.

[16] N. Rakuljic and I. Galton, "Tree-Structured DEM DACs with Arbitrary Numbers of Levels," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 57, no. 2, pp. 313-322, Feb. 2010, doi: 10.1109/TCSI.2009.2023931.

[17] Yijun Zhou and Jiren Yuan, "A 10-bit wide-band CMOS direct digital RF amplitude modulator," in IEEE Journal of Solid-State Circuits, vol. 38, no. 7, pp. 1182-1188, July 2003, doi: 10.1109/JSSC.2003.813290.

[18] P. T. M. van Zeijl and M. Collados, "On the Attenuation of DAC Aliases Through Multiphase Clocking," in IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 56, no. 3, pp. 190-194, March 2009, doi: 10.1109/TCSII.2009.2015365.

[19] IEEE Standard for Information Technology—Telecommunications and Information Exchange between Systems—Local and Metropolitan Area Networks—Specific Requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications—Amendment 1: Enhancements for High-Efficiency WLAN, IEEE Std 802.11ax-2021, Feb 2021.

Appendix A

Some extra material

A.1 Verilog Code for DEM switching block

```
1  //Verilog HDL for switching block "S_1_1_7"
2
3  module S_1_1_7 (Data, DEM_en, clk, reset, outup, outdown
       , PSN);
4
5      input [2:0] Data;
6      input clk, reset;
7      input DEM_en;
8      input [2:0] PSN;
9
10     output reg[2:0] outup;
11     output reg outdown;
12
13     reg [6:0] seq1,seq2,seq3,seq4,seq5,seq6;
14     reg SW1, SW2, SW3, SW4, SW5, SW6;
15     reg [5:0] enable_seq;
16     wire [6:0] Q1,Q2,Q3,Q4,Q5,Q6;
17 //   reg[3:0] outup,outdown;
18
19     assign Q1 = seq1;
20     assign Q2 = seq2;
21     assign Q3 = seq3;
22     assign Q4 = seq4;
23     assign Q5 = seq5;
24     assign Q6 = seq6;
25
26     always @(Data, SW1, SW2, SW3, SW4, SW5, SW6)
27         begin
28             case(Data)
29             //output that don't depend on sw when input
       is 0,7;
30                 3'b000:// input 0
```

```
31                      begin
32                          enable_seq = 6'b000000;
33                          outup = 3'b000;
34                          outdown = 1'b0;
35                      end
36
37                  3'b111:// input 7
38                      begin
39                          enable_seq = 6'b000000;
40                          outup = 3'b110;
41                          outdown = 1'b1;
42                      end
43
44              // output that depend on sw1 when input is
    1;
45                  3'b001://input 1
46                      begin
47                          enable_seq = 6'b000001;
48                          case(SW1 && DEM_en)
49                              0: begin outup = 3'b001;
    outdown = 1'b0; end
50                              1: begin outup = 3'b000;
    outdown = 1'b1; end
51                          endcase
52                      end
53
54                  3'b010://input 2
55                      begin
56                          enable_seq = 6'b000010;
57                          case(SW2 && DEM_en)
58                              0: begin outup = 3'b010;
    outdown = 1'b0; end
59                              1: begin outup = 3'b001;
    outdown = 1'b1; end
60                          endcase
61                      end
62
63                  //output that depend on sw2 when input
    is 3;
64                  3'b011://input 3
65                      begin
66                          enable_seq = 6'b000100;
67                          case(SW3 && DEM_en)
68                              0: begin outup = 3'b011;
    outdown = 1'b0; end
69                              1: begin outup = 3'b010;
    outdown = 1'b1; end
```

```
70                          endcase
71                      end
72
73                  3'b100://input 4
74                      begin
75                          enable_seq = 6'b001000;
76                          case(SW4 && DEM_en)
77                              0: begin outup = 3'b100;
    outdown = 1'b0; end
78                              1: begin outup = 3'b011;
    outdown = 1'b1; end
79                          endcase
80                      end
81
82                  3'b101://input 5
83                      begin
84                          enable_seq = 6'b010000;
85                          case(SW5 && DEM_en)
86                              0: begin outup = 3'b101;
    outdown = 1'b0; end
87                              1: begin outup = 3'b100;
    outdown = 1'b1; end
88                          endcase
89                      end
90
91                  3'b110://input 6
92                      begin
93                          enable_seq = 6'b100000;
94                          case(SW6 && DEM_en)
95                              0: begin outup = 3'b110;
    outdown = 1'b0; end
96                              1: begin outup = 3'b101;
    outdown = 1'b1; end
97                          endcase
98                      end
99
100             endcase
101         end
102
103 // Mux
104
105     always @(PSN,Q1, Q2, Q3, Q4, Q5, Q6)
106         begin
107
108             case(PSN)
109                 3'b000:
110                     begin SW1 = Q1[0];SW2 = Q2[0];SW3 =
```

```
    Q3[0];SW4 = Q4[0];SW5 = Q5[0];SW6 = Q6[0]; end
111                 3'b001:
112                     begin SW1 = Q1[1];SW2 = Q2[1];SW3 =
    Q3[1];SW4 = Q4[1];SW5 = Q5[1];SW6 = Q6[1]; end
113                 3'b010:
114                     begin SW1 = Q1[2];SW2 = Q2[2];SW3 =
    Q3[2];SW4 = Q4[2];SW5 = Q5[2];SW6 = Q6[2]; end
115                 3'b011:
116                     begin SW1 = Q1[3];SW2 = Q2[3];SW3 =
    Q3[3];SW4 = Q4[3];SW5 = Q5[3];SW6 = Q6[3]; end
117                 3'b100:
118                     begin SW1 = Q1[4];SW2 = Q2[4];SW3 =
    Q3[4];SW4 = Q4[4];SW5 = Q5[4];SW6 = Q6[4]; end
119                 3'b101:
120                     begin SW1 = Q1[5];SW2 = Q2[5];SW3 =
    Q3[5];SW4 = Q4[5];SW5 = Q5[5];SW6 = Q6[5]; end
121                 3'b110:
122                     begin SW1 = Q1[6];SW2 = Q2[6];SW3 =
    Q3[6];SW4 = Q4[6];SW5 = Q5[6];SW6 = Q6[6]; end
123             endcase
124         end
125
126 // shift register 1
127
128 always @(posedge clk, posedge reset)
129     begin
130         if (reset==1'b1)
131             seq1 <= 7'b1000000;
132         else begin
133             case(enable_seq[0])
134                 0:seq1 <= seq1;
135                 1: begin
136                     seq1[6] <= seq1[0];
137                     seq1[5] <= seq1[6];
138                     seq1[4] <= seq1[5];
139                     seq1[3] <= seq1[4];
140                     seq1[2] <= seq1[3];
141                     seq1[1] <= seq1[2];
142                     seq1[0] <= seq1[1];
143                 end
144             endcase
145         end
146     end
147
148 // shift register 2
149
150 always @(posedge clk, posedge reset)
```

```
151     begin
152         if (reset==1'b1)
153             seq2 <= 7'b1001000;
154         else begin
155             case(enable_seq[1])
156                 0:seq2 <= seq2;
157                 1: begin
158                     seq2[6] <= seq2[0];
159                     seq2[5] <= seq2[6];
160                     seq2[4] <= seq2[5];
161                     seq2[3] <= seq2[4];
162                     seq2[2] <= seq2[3];
163                     seq2[1] <= seq2[2];
164                     seq2[0] <= seq2[1];
165                 end
166             endcase
167         end
168     end
169
170 // shift register 3
171
172 always @(posedge clk, posedge reset)
173     begin
174         if (reset==1'b1)
175             seq3 <= 7'b1001001;
176         else begin
177             case(enable_seq[2])
178                 0:seq3 <= seq3;
179                 1: begin
180                     seq3[6] <= seq3[0];
181                     seq3[5] <= seq3[6];
182                     seq3[4] <= seq3[5];
183                     seq3[3] <= seq3[4];
184                     seq3[2] <= seq3[3];
185                     seq3[1] <= seq3[2];
186                     seq3[0] <= seq3[1];
187                 end
188             endcase
189         end
190     end
191
192 // shift register 4
193 always @(posedge clk, posedge reset)
194     begin
195         if (reset==1'b1)
196             seq4 <= 7'b1101001;
197         else begin
```

```
198             case(enable_seq[3])
199                 0:seq4 <= seq4;
200                 1: begin
201                     seq4[6] <= seq4[0];
202                     seq4[5] <= seq4[6];
203                     seq4[4] <= seq4[5];
204                     seq4[3] <= seq4[4];
205                     seq4[2] <= seq4[3];
206                     seq4[1] <= seq4[2];
207                     seq4[0] <= seq4[1];
208                 end
209             endcase
210         end
211     end
212
213
214 // shift register 5
215 always @(posedge clk, posedge reset)
216     begin
217         if (reset==1'b1)
218             seq5 <= 7'b1101011;
219         else begin
220             case(enable_seq[4])
221                 0:seq5 <= seq5;
222                 1: begin
223                     seq5[6] <= seq5[0];
224                     seq5[5] <= seq5[6];
225                     seq5[4] <= seq5[5];
226                     seq5[3] <= seq5[4];
227                     seq5[2] <= seq5[3];
228                     seq5[1] <= seq5[2];
229                     seq5[0] <= seq5[1];
230                 end
231             endcase
232         end
233     end
234
235 // shift register 6
236 always @(posedge clk, posedge reset)
237     begin
238         if (reset==1'b1)
239             seq6 <= 7'b1111011;
240         else begin
241             case(enable_seq[5])
242                 0:seq6 <= seq6;
243                 1: begin
244                     seq6[6] <= seq6[0];
```

```
245                         seq6[5] <= seq6[6];
246                         seq6[4] <= seq6[5];
247                         seq6[3] <= seq6[4];
248                         seq6[2] <= seq6[3];
249                         seq6[1] <= seq6[2];
250                         seq6[0] <= seq6[1];
251                     end
252                 endcase
253             end
254         end
255
256 endmodule
```

Printed by Tryckeriet i E-huset, Lund 2025

LUND
UNIVERSITY

Series of Master's theses
Department of Electrical and Information Technology
LU/LTH-EIT 2025-1091
http://www.eit.lth.se