Digital Phase Accumulation for Direct Digital Frequency Synthesis

by

Joseph Dominic Cali

A dissertation submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Auburn, Alabama
May 5, 2013

Keywords: DDS, DDFS, DCDO, DAC, Phase Truncation Errors, CORDIC

Approved by

Fa Dai, Chair, Professor of Electrical and Computer Engineering
Richard Jaeger, Ginn Distinguished Professor of Electrical and Computer Engineering
Robert Dean, Associate Professor of Electrical and Computer Engineering
Stanley Reeves, Professor of Electrical and Computer Engineering

Abstract

This work explores direct digital frequency synthesis (DDFS) theory and design and its application in radar systems. Though there is nothing particularly novel about DDFS in general, recent designs have been revolutionized with the advancements in CMOS processes and SiGe BiCMOS integration from 2000 to the current day. Many of the performance limitations highlighted in early literature, such as the area and power of the sinusoidal read-only memory (ROM), no longer apply to designs in modern integrated circuit (IC) processes. The digitally-controlled digital oscillator (DCDO) of the DDFS can now produce signals with spectral purity far beyond the capabilities of the digital to analog converter (DAC). CMOS miniaturization allows for high dynamic range sinusoids to be generated with CORDICs instead of lossy compressed sine and cosine ROMs. Parallelization in the accumulator and modulation paths eliminate the need for power hungry, current mode logic (CML) pipeline accumulators. Noise shaping is better understood than at any point prior to this moment, which allows us to mitigate quantization noise that arises from phase or amplitude truncation.

However, alarmingly few DDFS designs published in the past five years have taken note of the radical shift in the design landscape. Of equal importance are the new challenges that have arisen in small feature size geometries. In a way, this document is an attempt to consolidate the state of the art in DDFS design and propose improvements from the study. To this end, the dissertation is organized into two distinct sections, the DCDO and the DAC. Digital phase accumulation and sinusoid generation are approached from number theory and real analysis respectively. An exact computation of the spurs generated through phase truncation is developed that results in closed form expressions for the DCDO spectrum. Current switches and architectures for improved DAC performance is presented qualitatively.

Acknowledgments

Journeying down the path of higher education can rarely be attributed to the will power or foresight of the individual in pursuit. In recent years, I have appreciated the support of the faculty and staff of Auburn University who have guided me through a challenging five years of graduate school. In addition, I benefitted from the assistance of my fellow graduate students with whom all my designs have interfaced in some manner. I acknowledge my major advisor, Dr. Fa Dai for taking me on as a graduate student and funding eight integrated circuit designs through my stint as a graduate student. I also must mention the members of my committee Dr. Dean, Dr. Jaeger and Dr. Reeves for their specialized assistance through many challenging design problems. I cannot fail to mention Dr. Niu, as his passionate and skilled teaching of semiconductor physics from his deep knowledge of the subject has proven helpful dozens of times on the job in my short time in the workforce.

There are countless teachers who from kindergarten through my undergraduate degree at Louisiana State University (LSU) have devoted their energy and time to teaching me and putting up with my relentless questions with regard to the "how's" and the "why's" of this world. Without the prodding of my professors at LSU, I may have never considered an advanced degree. Above these teachers stand the two greatest teachers in my life, my mother and father, who have patiently raised me and provided emotional and financial support throughout my academic journey. They sacrificed many conveniences for me to attend a private school in preparation for college.

Lastly, I must thank my wife, Alison, for supporting me through the endless nights of class work, the many weekends of research, my late night existential crises (now why am I in graduate school again?), and tough medical challenges. She has certainly done more to shape the outcome of this work than any other person in my life.

Table of Contents

List of Figures

List of Tables

List of Symbols

List of Theorems

List of Abbreviations

BIST Built-In Self-Test

BPSK Binary Phase Shift Keying

BTM Bipartite Table Method

CDMA Code Division Multiple Access

CML Current Mode Logic

CORDIC COordinate Rotation DIgital Computer

CS current steering

CW Continuous Wave

DAC Digital-to-Analog Converter

DDFS Direct Digital Frequency Synthesis

DEM Dynamic Element Matching

DFF D-Flip-Flop

DFT Discrete Fourier Transform

DNL Differential Non-Linearity

DSP Digital Signal Processing

Emitter Coupled Logic

ENOB Effective Number of Bits

FCW Frequency Control Word

FIR Finite Impulse Response

GCD Greatest Common Divisor

IDE Integrated Development Environment

INL Integral Non-Linearity

KCL Kirchoff Current Law

KVL Kirchoff Voltage Law

LFM Linear Frequency Modulation

LSB Least Significant Bit

MSB Most Significant Bit

MTM Multipartite Table Method

NRTZ Non-Return-to-Zero

ORA Output Response Analyzer

PLL Phase-Locked Loop

PM Phase Modulation

QAM Quadrature Amplitude Modulation

QPSK Quadrature Phase Shift Keying

RAM Random Access Memory

ROM Read-Only Memory

RTZ Return-to-Zero

SCMF Sine or Cosine Mapping Function

SFDR Spurious Free Dynamic Range

SINAD Signal to Noise Ratio and Distortion

SNDR Signal to Noise and Distortion Ratio

SNR Signal to Noise Ratio

Source Coupled Logic

SPI Serial Peripheral Interface

SSM Static Mismatch Shaping

SSPA Switching Sequence Post Adjustment

THD Total Harmonic Distortion

TSPC True Single Phase Clock

WLAN Wireless Local Area Network

Chapter 1

Introduction to Phase Accumulators

In this chapter, Direct Digital Frequency Synthesis (DDFS) is introduced as an important component in modern 21st century communication systems, and its fundamental operating principles are presented. Wireless cellular communication techniques such as code division multiple access (CDMA) and spread spectrum wireless local area networks (WLAN) [2] require fast frequency switching, an attribute in which DDFS excels over conventional analog frequency synthesis approaches. As integrated circuit processes advance, DDFS is also emerging as a critical component in commercial radar systems, agile clock synthesizers [3] and high speed testing equipment [4] opening up new opportunities in industries outside of telecommunications, the automotive industry being one of the more exciting [5],[6].

In DDFS systems, the amplitude, frequency, and phase of synthesized waveforms can be modulated digitally and nearly instantaneously, which depending upon the operating frequency of the technology and the level of pipelining in the digital core could mean less than a few nanoseconds. The lock time of a standard analog phase-locked loop (PLL) can be on the order of several hundred microseconds as a result of the slow settling time of the loop filter [7]. The ability to directly modulate the signal also allows for arbitrary, high-bandwidth waveform synthesis varying from simple phase-shift keying used in low cost signal data transmission systems to complex non-linear frequency sweeps used in radar systems [8]. One of the better published results of an arbitrary waveform generator is presented by Van de Sande *et al.* [4] the year of this writing, indicating that research in the field remains active.

The DDFS operates not by digitally controlling an analog oscillator component but by numerically computing a complex digital signal and directly converting it to a physical electrical quantity through a digital-to-analog converter (DAC). The phase, frequency, and

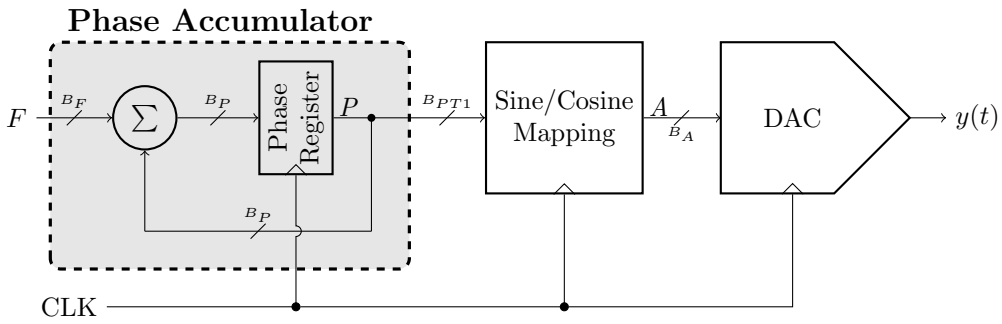


Figure 1.1: Basic DDFS Block Diagram

amplitude of a DDFS system are themselves digital codes that can be modulated in the digital domain through simple multiplication and addition operations. These operations, excluding quantization, are completely linear and thus superior to the analog equivalent operations that apply unwanted harmonic distortion and spurious mixing to the signal.

The earliest implementation of a circuit in an academic publication that resembles the modern DDFS appears in 1971 by Joseph Tierney *et al.* [9]. Figure 1.1 shows the architecture of the DDFS proposed in [9], exluding the quadrature sine and cosine outputs and the analog reconstruction filter after the DAC, which is the basic architecture of modern DDFS devices. The focus of this dissertation is explaining how to generate spectrally pure sinusoids with such a device in an efficient manner by gathering published results for the various components comprising the device and inserting mathematical explanations when necessary or helpful. Care is taken in an attempt to place the analysis of DDFS systems on a clear mathematical foundation and perhaps to illuminate some of the more difficult concepts such as the rise of spurs through phase truncation.

For readers not familiar with the terminology, a spur is unwanted coherent spectral energy, harmonically related to the intended synthesized signal or otherwise. This implies that attaching a spectrum analyzer to the capture the waveform generated, one would see a distinct tone that did not decrease in power when averaged over time, hence the use of coherency in the definition. In the digital domain, one would find that increasing the length

of the Discrete Fourier Transform also had no influence over the magnitude or phase of the unwanted tones.

The fundamental components of the DDFS are a clock receiver and distribution tree, an overflowing accumulator, a sine and/or cosine mapping function (SCMF), one or more DACs, and a reconstruction filter at the DAC(s) output. The overflowing accumulator is often called the *phase accumulator*, as its cyclic overflowing is analogous to the phase of a sinusoid. The accumulator is incremented by a value known as the *frequency control word* (FCW). The reconstruction filter is not studied in detail, but several DAC clocking methodologies to reduce the stringent requirements of the filter are presented during the DAC architecture survey (Section 7.5).

The term SCMF is used instead of the more common read-only-memory (ROM) or "lookup table" (LUT). The terminology is borrowed from Torosyan in his dissertation and publications [10] and is general enough to encompass the wide range of techniques available for sinusoidal phase to amplitude conversion. The name choice also separates the functional behavior of the component from its realization on silicon. The cost of digital memory has become so remarkably inexpensive both in area and power that some designs use random access memory (RAM) as a opposed to a ROM to implement the SCMF function. Bipartite and multipartite table methods , BTM [11] and MTM [12] respectively, and the COrdinate Rotation DIgital Computer (CORDIC) [13] are implemented and studied in this work as effective techniques for implementing the SCMF.

1.1 Explanation of Notation

The conventions used in this document are described in this section for reference. This is particularly important in mixed-signal systems such as a DDFS, as many of the analyses of the behavior of the device transition between discrete-time and continuous-time representations of the signal. The same conceptual entity crosses several processing domains. In order to clearly denote when a digital variable, or some non-digitized discrete time sequence,

is intended, an upper case English letter glyph is used. For instance, P represents the phase state of the phase accumulator and A represents the amplitude output of the SCMF. An immense effort was put into the writing attempting

- To provide consistency in notation. The author wants avoid incessant flipping between this section and subsequent sections.
- To avoid collisions with important variables in literature. For instance, repeatedly using a variable Q in a text about passive filters in a manner unrelated to the quality-factor of an inductor or energy storage tank can be confusing.

The n^{th} element in the sequence P is denoted with square brackets, $P[n]$ being the state of the phase accumulator at clock cycle n. Parentheses are used to denote continuous functions, where $y = x(t)$ is the value of the function x corresponding to the argument t. The phase accumulator at time nT_s is given as $P(nT_s)$, where T_S is the period of the clock driving the DDFS. Some mathematics texts [14] use a more general and formal notation to represent the same function concept $x : t \mapsto y$, where $t \in S_T$ and $y \in S_Y$ and S_T and S_Y are sets. The notation $x \in S$ means that the element x is contained in the set S.

There are some commonly used sets in mathematics that appear frequently in DDFS analysis. Instead of repeatedly listing the elements that form the set, a list of all sets used

in this document are presented below (many used by [15] in his Modern Algebra text):

$$\emptyset = \text{The empty set (i.e. that set containing no elements)} \tag{1.1}$$

$$\mathbb{B} = \text{The set containing only 0 and 1} \tag{1.2}$$

$$\mathbb{P} = \text{The set of all positive integers, also known as the natural numbers} \tag{1.3}$$

$$\mathbb{P}_0 = \text{The set of all positive integers including zero} \tag{1.4}$$

$$\mathbb{Z} = \text{The set of all integers} \tag{1.5}$$

$$\mathbb{Z}_n = \text{The set: } \{x \in \mathbb{Z} : 0 \leq x < n\} \tag{1.6}$$

$$\mathbb{R} = \text{The set of all real numbers including } \infty \text{ and } -\infty \tag{1.7}$$

$$\mathbb{C} = \text{The set of all complex numbers} \tag{1.8}$$

Several small proofs will be derived in the text to lay a foundation for some of the fundamental behavior of DDFS systems. Some readers may find this excessive, but the author believes that the best explanations to the spurious behavior of phase truncation come from number theory used in conjunction with real and complex analysis. While Torosyan felt it sufficient to state in a single line that "this result follows from a fundamental result from number theory" [16] citing a large text on number theory, this work attempts to pull the information from various works on abstract algebra and number theory to supplement the material. If completed to the level of detail intended by the author, purchasing a mathematics textbook to understand a key statement in this work should not be necessary.

The digital signals of the DDFS have a finite word length representation. The number of bits of resolution for a variable is denoted by a capital B with the variable name as the subscript. Thus the number of bits in the phase accumulator is denoted B_P and the number of bits of amplitude resolution is denoted as B_A. The number of unique states that can be represented by the finite bit length word is denoted by a capital N with the variable name as the subscript. Going back the phase accumulator as the example, a B_P-bit phase accumulator has $N_P = 2^{B_P}$ states.

The least period of sequences, of which discussions begin in Section 1.2, are denoted using Λ. The length of the least period of the phase accumulator with N_P states would be Λ_P. In later sections, it becomes clear that Λ_P is a function of the frequency control word driving the phase accumulator. The reduced frequency control word (Section 1.2) is denoted with Γ. If the sequence is modulo N_P with frequency control word F, then Γ_P denotes the reduced frequency control word (Equation 3.26).

The multiplicative inverse of a number or variable is denoted as the name of variable with -1 in the superscript. In this work, the multiplicative inverses under analysis are integer numbers a^{-1} that such $\langle aa^{-1} \rangle_N = 1$. This will be used quite frequently in Chapter 4 where the inverse of the reduced frequency control word is frequently used in computing spectrum permutations.

1.1.1 Number Theory Axioms and Notation

In this section, several axioms and definitions that are used in proofs in Section 1.2 and Chapter 3 are supplied. The principle of mathematical deduction is defined as follows [15]:

Principle 1.1 (Mathematical Induction)**.** *Suppose S is a subset of $\mathbb{P}$ such that the following two properties hold:*

1. $1 \in S$.
2. *For all $k \in \mathbb{P}$, if $k \in S$, then $k + 1 \in S$.*

then $S = \mathbb{P}$.

Mathematical induction is the last of Peano's five axioms for natural numbers [15], and thus it is sufficient to treat it as an axiom in this work. The first step, showing that $1 \in S$ is called the basis step of mathematical induction. The second step is called the induction step. This will be used to show that any integer can be represented by a binary number.

Principle 1.2 (Well-Ordering Principle)**.** *Every nonempty set of non-negative integers has a smallest element.*

The mathematical constructs necessary to develop the well-ordering principle from the axiom of choice are beyond the scope of this work. In some mathematical systems the well-ordering principle is treated as an axiom itself, underlying the subtleties of a mathematical principle that appears trivial at first glance. The presentation of the division algorithm (Theorem 3.1) in Section 3.1 makes use of this principle. As a majority of the derivations in the work make use of the division algorithm, the Well-Ordering Principle provides an underpinning for the entire text.

Most of the arithmetic in hardware implementations is modulo arithmetic attributable to the finiteness of the physical components. The notation $m \mid b$ is read m *divides* b. This notation appears in the explanation of the modulo behavior of the phase accumulator in Chapter 3.

Definition 1.1 (Divides)**.** *An integer m divides an integer b, or symbolically $m \mid b$, if and only if there exists an integer d such that $md = b$.*

If no such integer d exists, then m does not divide b, or symbolically $m \nmid b$. Many of the proofs in Chapter 3 use this definition. Also in Chapter 4, it is shown that the analysis of the spectrum of a digitally controlled digital oscillator (DCDO) can be dramatically simplified whenever the number of error states divides the number of phase accumulator states. The last definition necessary before starting the analysis is the least common multiple (LCM) of two integers.

Definition 1.2 (Least Common Multiple)**.** *The least common multiple of two integers a and b, symbolically denoted as $LCM(a, b)$, is the smallest positive integer c such that $a \mid c$ and $b \mid c$.*

The least common multiple of two integers a and b is unique, as two integers cannot both be *least* from the well-ordering theorem (Principle 1.2). All other definitions and theorems useful for understanding the behavior of a DDFS are presented or derived as necessary.

1.1.2 Binary Arithmetic

A majority of the digital signals are implemented in hardware as binary numbers. This flows from the efficiency and robustness of which two state logic can be implemented in electronic circuits [17]. The arithmetic operations on these digital signals are then binary arithmetic operations. This text, unless explicitly noted in the section, assumes the value of a digital signal represented by bits is unsigned. The binary number $B = b_{N-1}b_{N-2}\cdots b_0$ in its unsigned representation has the base-10 value given in Equation 1.9.

$$v_B = \sum_{i=0}^{N-1} 2^i b_i \tag{1.9}$$

where b_i takes either the value of 0 or 1 (i.e. $b_i \in \mathbb{B}$) and b_i is called the i^{th} *bit*. From Equation 1.9, it is clear the maximum unsigned value of an N-bit word is obtained when all $b_i = 1$.

$$\max\{v_B\} = \sum_{i=0}^{N-1} 2^i = 1 + 2 + \cdots + 2^{N-2} + 2^{N-1} = 2^N - 1 \tag{1.10}$$

The summation is evaluated using an analysis typical for geometric series (Lemma 1.2), multiplying both sides by $(r-1)$, where $r = 2$ in this case and rearranging. The minimum possible unsigned value for B is obtained by setting all $b_i = 0$. Thus $\min\{v_B\} = 0$.

Theorem 1.1 (Binary Number Representation)**.** *Any non-negative integer can be represented by an unsigned binary number.*

Proof. Now let us show that any $x \in \mathbb{P}_0$ can be represented by an unsigned binary number. First we have already noted that $x = 0$ can be represented by Equation 1.9 by setting all $b_i = 0$. Now we need only show that $x \in \mathbb{P}$ can be represented by an unsigned binary number. We will do so by using strong induction. Assume that S is a non-empty subset of $\mathbb{P}$. For the basis step, we note that for $x = 1$, then Equation 1.9 can be made to equal 1 by

setting $b_i = 0$ for $i > 0$ and $b_0 = 1$.

$$x = \sum_{n=0}^{\infty} 2^i b_i = 2^0 b_0 = 1, \;\; b_i = 0, i > 0 \tag{1.11}$$

Thus $1 \in S$. Now we take the induction step. Assume that $x \in S$ and every positive integer from 1 to x can be represented by a binary number. We must now show that $x+1 \in S$. We can do this by finding a binary representation of $x+1$, or equivalently, finding c_i such that

$$x + 1 = \sum_{n=0}^{\infty} 2^i c_i \tag{1.12}$$

Let us briefly consider some of the properties of x. By our induction step, we know that x can be written as an unsigned binary number:

$$\begin{aligned} x &= \sum_{n=0}^{\infty} 2^i b_i = 2^0 b_0 + 2^1 b_1 + \cdots \\ &= 2^0 b_0 + 2\left(b_1 + 2^1 b_2 + \cdots\right) \end{aligned} \tag{1.13}$$

Clearly, the second term is a multiple of 2 and is therefore an even number by definition. If $b_0 = 1$, then the first term evaluates to 1, and adding 1 to an even number yields an odd number. So for x to be even $b_0 = 0$ otherwise x is odd. Now let us tackle the case of $x+1$ assuming x is even.

$$\begin{aligned} x + 1 = \sum_{n=0}^{\infty} 2^i c_i &= \sum_{n=0}^{\infty} 2^i b_i + 1 \\ 2^0 c_0 + 2^1 c_1 + \cdots &= \left(2^0 b_0 + 2^1 b_1 + \cdots\right) + 1 \end{aligned} \tag{1.14}$$

Since x is assumed even, $b_0 = 0$. Applying this knowledge and rearranging we get

$$2^0 c_0 - 1 = \left(2^1 b_1 + 2^2 b_2 + \cdots\right) - \left(2^1 c_1 + 2^2 c_2 + \cdots\right) \tag{1.15}$$

Setting $c_0 = 1$ and setting $b_i = c_i$ for $i = \{1, 2, \cdots\}$ sets both the left and right hand sides of the equation to zero and the equality holds. Thus $x + 1 \in S$ whenever x is even. Now let us consider the case when x is odd. If x is odd then $x + 1$ is even.

Since $x + 1$ is even, $2|(x + 1)$ by definition and there exists a number d such that $2d = (x + 1)$, in this case d is a positive integer since $x + 1$ is an even positive integer. If we can show that d can be written as an unsigned binary number then $x + 1$ can be written as an unsigned binary number. We can show this by proving that $d \leq x$ and thus by our induction hypothesis, i.e. every positive integer from 1 to x can be represented by a binary number, $d \in S$.

$$d \leq x \Rightarrow \frac{(x+1)}{2} \leq x \Rightarrow (x+1) \leq 2x \tag{1.16}$$

Since the least integer in our set S is 1, it is clear that the previous inequality holds (if this is not satisfactory, then apply induction to the inequality). Since $d \in S$, we can now find the binary representation of $x + 1$.

$$\begin{aligned}
(x+1) &= 2d \\
\sum_{n=0}^{\infty} 2^i c_i &= 2\left(\sum_{n=0}^{\infty} 2^i b_i\right) \\
\left(2^0 c_0 + 2^1 c_1 + 2^2 c_2 + \cdots\right) &= \left(2^1 b_0 + 2^2 b_1 + \cdots\right) \\
\left(2^1 c_1 + 2^2 c_2 + \cdots\right) &= \left(2^1 b_0 + 2^2 b_1 + \cdots\right)
\end{aligned} \tag{1.17}$$

Since $x + 1$ is even, $c_0 = 0$. It is clear from Equation 1.17 that setting $c_i = b_{i-1}$ for all $i > 0$ causes the equality to hold. Therefore $x + 1 \in S$ whenever x is odd. Since the induction step holds for all $x + 1$, the set $S = \mathbb{P}$ and we have shown that all positive integers can be represented by an unsigned binary number. □

In two's complement representation, the value of B is

$$v_B = -2^{N-1}b_{N-1} + \sum_{i=0}^{N-2} 2^i b_i. \tag{1.18}$$

The minimum and maximum values of the two's complement representation can be computed similarly to the unsigned binary case. Using Equation 1.18, the maximum value is obtained by setting $b_{N-1} = 0$ and b_{N-2} down to b_0 to 1. The minimum value is obtained by setting $b_{N-1} = 1$ and b_{N-2} down to b_0 to 0.

$$\max\{v_B\} = \sum_{i=0}^{N-2} 2^i = 2^{N-1} - 1 \tag{1.19}$$

$$\min\{v_B\} = -2^{N-1} \tag{1.20}$$

The conversion of an unsigned full-scale binary number to the two's complement number system such that the zero from the unsigned representation maps to the lowest two's complement value and the maximum valued in unsigned representation maps to the maximum two's complement value involves only inverting the most significant bit (MSB) of the unsigned number.

A technique used commonly in DDFS designs to approximate the negation of the value of an integer is to take a one's complement of the two's complement binary representation of the number. In one's complement, all the bits of B are inverted. This operation is popular because of its efficient hardware implementation, as only N XOR gates are required for the inversion of an N-bit word. The architectures presented in Chapter 6 utilize this technique in the sinusoidal compression algorithm. Figure 1.2 is a gate level block diagram of a conditional one's complement operation. The bit a inverts all the b_i bits when asserted high but does not affect the value of b_i when asserted low.

One must carefully evaluate the approximation of negation using one's complement in the system. So consider the effect of one's complement on a word B in a two's complement

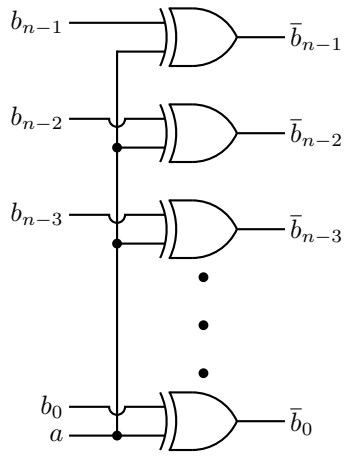


Figure 1.2: Gate Logic for One's Complement

binary system. The resulting one's complement value v_{B1} is given in Equation 1.21.

$$v_{B1} = -2^{N-1}\overline{b}_{N-1} + \sum_{i=0}^{N-2} 2^i \overline{b}_i \tag{1.21}$$

where $\overline{b}_i$ is the complement of b_i, meaning that if $b_i = 0$ then $\overline{b}_i = 1$ and if $b_i = 1$ then $\overline{b}_i = 0$. From this one can see that the one's complement does not negate v_B, which is to say $-v_B \neq v_{B1}$.

$$\begin{aligned} v_{B1} + v_B &= -2^{N-1}\left(b_{N-1} + \overline{b}_{N-1}\right) + \sum_{i=0}^{N-2} 2^i \left(b_i + \overline{b}_i\right) \\ &= -2^{N-1} + \sum_{i=0}^{N-2} 2^i \\ &= -1 \end{aligned}$$

since $b_i + \overline{b}_i = 1$ using the definition previous supplied by the definition of a complement in a binary number system. From the previous equation, we see that $v_{B1} = -v_B - 1$.

1.2 Overview of Direct Digital Frequency Synthesis

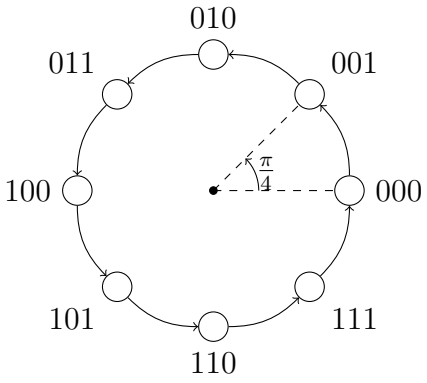
The DDFS of Figure 1.1 operates by incrementing an accumulator at the clock frequency f_{clk} by the value F, where $F \in \mathbb{Z}_{N_P}$ and $\mathbb{Z}_{N_P} = \{0, 1, 2, \ldots, N_P - 1\}$ is the set of all integers between 0 and $N_P - 1$ inclusive. F is generally constrained between zero and the maximum value of the phase accumulator, though there are exceptions to this rule when reducing area overhead of the adder and control logic by a minuscule margin is critical [13]. F represents the FCW and will be used as its symbol in mathematical notation. The accumulator has a finite bit resolution B_P and therefore the accumulator will overflow periodically as F is continually added to the previous accumulator state. The rate of overflow of the phase accumulator is thus dependent on F. F thereby controls the frequency of the synthesized DDFS waveform, suggesting that both the phase accumulator and FCW are aptly named.

The periodic overflowing of the accumulator is a remarkably efficient technique for implementing the periodic phase behavior of a sinusoid. The phase accumulator maps $[0, N_P)$ to $[0, 2\pi)$ when driving a lookup table that maps $P \in [0, N_P)$ to $\sin(2\pi P/N_P)$. Referring to Equation 1.10, $N_P = 2^{B_P} - 1$ is the maximum integer value that can be stored in the phase accumulator when treating the phase accumulator value as an unsigned integer. With additional hardware, the number of phase states can be set to any positive integer less than $2^{B_P} - 1$. One of the new achievements of this work is finding closed form equations for the spectrum of a DCDO for any N_P. The relationship between the current integer phase state of the accumulator, P, and the analogous normalized phase value, θ, is then

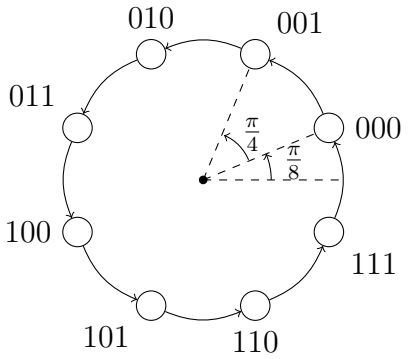
$$\theta[n] = \frac{2\pi}{N_P} P[n] \tag{1.22}$$

where θ is in radians. This maps the phase states uniformly across $[0, 2\pi)$ in N_P steps. Figure 1.3a demonstrates the mapping between the phase states of a 3-bit accumulator and the corresponding phase in radians. But this particular mapping need not be the case. The phase accumulator could, in fact, map to any closed interval in $\mathbb{R}$. This feature will prove

useful in implementing sinusoidal quarter-wave compression in Section 6.1.1. Figure 1.3b shows a 3-bit phase accumulator with a 1/2 least significant bit (LSB) offset in which case $P \in [0, N_P)$ maps to $[\pi/8, 2\pi + \pi/8)$. Note that in both the figures, the phase step remains



(a) Phase Mapping Circle



(b) Phase Mapping Circle (1/2 LSB Offset)

Figure 1.3: Phase Accumulator State Plots (Circle)

uniform. The adjustment in the mapping happens through the SCMF and will be discussed in more detail in Chapter 6.

The SCMF of the DDFS takes the value of P (Equation 3.4) and maps it to the appropriate sine or cosine value A. Equation 1.23 shows the SCMF mapping for an untruncated phase accumulator word to an ideal sine function.

$$\begin{aligned} A[n] &= \sin\left(\frac{2\pi}{N_P} P[n]\right) \\ &= \sin\left(\frac{2\pi}{N_P} \langle nF + P_0 \rangle_{N_P}\right) \\ &= \sin\left(\frac{2\pi F}{N_P} n + \frac{2\pi}{N_P} P_0\right) \end{aligned} \tag{1.23}$$

The modulo N_P arithmetic in the argument can be dropped since the period of the sinusoid is 2π and implicitly executes the modulo operation. This can be demonstrated through the following simple lemma.

Lemma 1.1 (Dropping Modulo Operation in Sinusoids)**.** *The modulo operator can be dropped within the sine and cosine functions with argument $2\pi P[n]/N_P$, where $P[n]$ is given by Equation 3.1. Equivalently,*

$$\sin\left(\frac{2\pi}{N_P}\left\langle nF + P_0\right\rangle_{N_P}\right) = \sin\left(\frac{2\pi}{N_P}\left(nF + P_0\right)\right) \tag{1.24}$$

Proof. Using the definition of the modulo operation described in Section 3.1, for arbitrary $n \in \mathbb{Z}$ and $P_0 \in \mathbb{Z}$ there exists an integer d such that

$$\left\langle nF + P_0\right\rangle_{N_P} = nF + P_0 - dN_P = r \tag{1.25}$$

where $0 \leq r < N_P$. Plugging $nF + P_0 - dN_P$ for $P[n]$ and applying the trigonometric difference identity for sine yields:

$$\begin{aligned}\sin\left(\frac{2\pi}{N_P}\left(nF + P_0 - dN_P\right)\right) = \sin\left(\frac{2\pi}{N_P}\left(nF + P_0\right)\right)\cos\left(\frac{2\pi}{N_P}\left(dN_P\right)\right) \\ - \sin\left(\frac{2\pi}{N_P}\left(dN_P\right)\right)\cos\left(\frac{2\pi}{N_P}\left(nF + P_0\right)\right)\end{aligned} \tag{1.26}$$

This can be further reduced by observing that

$$\cos\left(\frac{2\pi}{N_P}\left(dN_P\right)\right) = \cos\left(2\pi d\right) = 1 \tag{1.27}$$

$$\sin\left(\frac{2\pi}{N_P}\left(dN_P\right)\right) = \sin\left(2\pi d\right) = 0 \tag{1.28}$$

Finally, substituting Equation 1.27 and Equation 1.28 back into Equation 1.26,

$$\sin\left(\frac{2\pi}{N_P}\left(nF + P_0 - dN_P\right)\right) = \sin\left(\frac{2\pi}{N_P}\left(nF + P_0\right)\right) \tag{1.29}$$

□

The spectrum of such a sinusoid (Equation 1.23) can calculated by executing a discrete Fourier transform over one period of the waveform, which is N_P in this particular case. The discrete Fourier transform is defined later in Section 4.3.

$$\begin{aligned}\mathcal{F}\{A\}[k] &= \sum_{n=0}^{N_P-1} \sin\left(\frac{2\pi F}{N_P}n\right) e^{-j2\pi kn/N_P},\ 0 \le k \le N_P - 1 \\ &= \sum_{n=0}^{N_P-1} \frac{1}{2j}\left[e^{j2\pi Fn/N_P} - e^{-j2\pi Fn/N_P}\right] e^{-j2\pi kn/N} \\ &= \sum_{n=0}^{N_P-1} \frac{1}{2j}\left[e^{j2\pi n(F-k)/N_P} - e^{-j2\pi n(F+k)/N_P}\right] \end{aligned} \tag{1.30}$$

Euler's formula, given in Equation 1.31, was applied to the sine function to simplify the expression.

$$e^{jx} = \cos(x) + j\sin(x) \tag{1.31}$$

Euler's formula can be used to write sine and cosine function as the sum of two complex exponentials as follows (to verify, substitute Equation 1.31 for each of the complex exponential terms in the equations below):

$$\sin(x) = \frac{1}{2j}\left[e^{jx} - e^{-jx}\right] \tag{1.32}$$

$$\cos(x) = \frac{1}{2}\left[e^{jx} + e^{-jx}\right]. \tag{1.33}$$

Equation 1.30 is the difference of two geometric series. A technique for finding the closed form solution of a geometric series is presented here.

Lemma 1.2 (Geometric Series)**.** *A summation series of the form*

$$\sum_{n=0}^{N-1} r^n$$

is called a finite geometric series and can be written as the ratio of two numbers,

$$\sum_{n=0}^{N-1} r^n = \frac{1-r^N}{1-r} \tag{1.34}$$

if $r \neq 1$.

Proof. Let $r \neq 1$. The problem is solved by expanding the summation

$$\sum_{n=0}^{N-1} r^n = 1 + r + r^2 + \cdots + r^N$$

and multiplying both sides by $1-r$ and using the distributive property of multiplication over addition.

$$\begin{aligned}(1-r)\sum_{n=0}^{N-1} r^n &= (1-r)\left(1 + r + r^2 + \cdots + r^{N-1}\right) \\ &= \left(1 + r + r^2 + \cdots + r^{N-1}\right) - \left(r + r^2 + r^3 + \cdots + r^N\right) \\ &= 1 - r^N\end{aligned}$$

Then dividing both sides of previous equation by $1-r$, which can be done since $r \neq 1$, gives

$$\sum_{n=0}^{N-1} r^n = \frac{1-r^N}{1-r} \tag{1.35}$$

and the proof is complete. □

The summation of $e^{j2\pi an/N_P}$ for $a \in \mathbb{Z}$ over one period is zero if $N_P \nmid a$. This can be shown by directly computing the summation using Equation 1.34,

$$\begin{aligned}\sum_{n=0}^{N_P-1} e^{j2\pi an/N_P} &= \sum_{n=0}^{N_P-1} \left(e^{j2\pi a/N_P}\right)^n \\ &= \frac{1-e^{j2\pi aN_P/N_P}}{1-e^{j2\pi a/N_P}} \\ &= \frac{1-e^{j2\pi a}}{1-e^{j2\pi a/N_P}} = \frac{1-1}{1-e^{j2\pi a/N_P}} = 0\end{aligned}$$

The summation of a geometric series works because $e^{j2\pi a/N_P} \neq 1$ when $N_P \nmid a$.

Lemma 1.3 (When the Complex Exponential Equals 1)**.**

$$e^{j2\pi a/N_P} = 1 \tag{1.36}$$

if and only if $N_P \mid a$.

Proof. The forward proof is simple, as $N_P \mid a$ means that there exists an integer d such that $dN_P = a$. Substituting this value for a in Equation 1.36 yields:

$$e^{j2\pi dN_P/N_P} = e^{j2\pi d}. \tag{1.37}$$

Applying Euler's formula (Equation 1.31) to Equation 1.37

$$e^{j2\pi d} = \cos(2\pi d) + j\sin(2\pi d) = 1 + j0 = 1 \tag{1.38}$$

Now one must show that if Equation 1.36 holds then $N_P \mid a$. Applying Euler's formula to Equation 1.36 yields

$$e^{j2\pi a/N_P} = \cos\left(\frac{2\pi a}{N_P}\right) + j\sin\left(\frac{2\pi a}{N_P}\right) = 1 \tag{1.39}$$

The right hand side of Equation 1.39 only equals 1 when the cosine term equals 1 and the sine term equals 0. Cosine only equals 1 when the argument is $2\pi n$ for $n \in \mathbb{Z}$. Then

$$2\pi n = \frac{2\pi a}{N_P} \tag{1.40}$$

$$a = N_P n \tag{1.41}$$

Then $N_P \mid a$. Plugging the solved value of a into the sine argument gives 0 and the equation holds. Therefore, Equation 1.36 is true if and only if $N_P \mid a$. □

Finally, the value of the summation for when $F - k = 0$ must be computed

$$\sum_{n=0}^{N_P-1} \frac{1}{2j} e^{j2\pi n(F-k)/N_P} = \frac{1}{2j} \sum_{n=0}^{N_P-1} e^{j2\pi n0/N_P} \tag{1.42}$$

$$= \frac{1}{2j} \sum_{n=0}^{N_P-1} 1 = \frac{N_P}{2j} \tag{1.43}$$

Gathering together results of the analysis gives the final DFT result.

$$\mathcal{F}\{A\}[k] = \begin{cases} \frac{N_P}{2j} & k = F \\ -\frac{N_P}{2j} & k = -F \\ 0 & \text{otherwise} \end{cases} \tag{1.44}$$

The results of the DFT indicate that all the spectral energy generated by a DDFS assuming no phase truncation and an ideal SCMF is located at a single frequency bin F, which is precisely the FCW. This leads to what should perhaps be referred to as the fundamental equation for DDFS systems. Noting the relationship between the DFT and CTFT for uniform sampling interval T_{clk}, an equation for the output frequency of the DDFS can be formulated

$$f_0 = \frac{F}{N_P} f_{clk} \tag{1.45}$$

where f_0 is the frequency of the fundamental tone generated by the DDFS system.

1.3 Advantages of DDFS

As is the custom for Auburn University dissertations and publications concerning DDFS devices, a brief explanation of the advantages of DDFS over traditional PLLs is supplied. The benefits ultimately reduce to the benefits of operating on signals in the digital domain over the analog domain:

- Straight-forward efficient implementation of complex modulation schemes.

- Direct manipulation of parameters that are often difficult to precisely control in the analog equivalent circuit.
- Digital signals are not corrupted by noise as easily as their analog domain counterparts.
- Arithmetic operations are highly linear and tolerant to device mismatch.

In addition to these, several other benefits of DDFS already mentioned earlier in the introduction are summarized.

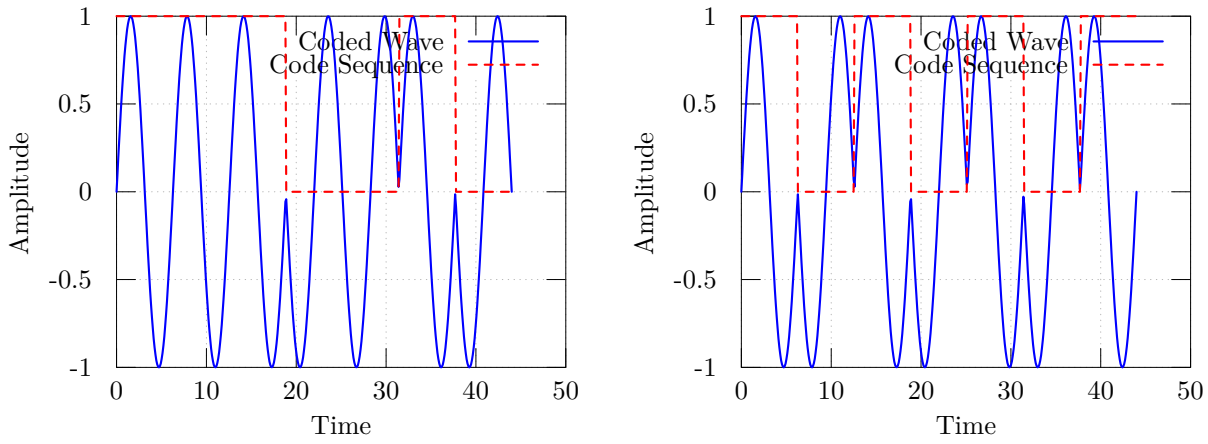
- Very fast frequency switching, several orders of magnitude faster than that of a traditional PLL [7].
- Fine frequency resolution (see Equation 1.45).
- Broadband frequency synthesis, as the same device with a sufficiently fast input clock can synthesize signals on the order of several Kilohertz to the order of several Gigahertz.

1.3.1 Digital Phase Modulation

Phase modulation (PM) is the process of adding, subtracting, multiplying or otherwise affecting the output of the phase accumulator. The phase signal after phase modulation is

$$P_m[n] = P[n] + M[n] \quad \text{or} \quad P_m[n] = M[n]P[n] \tag{1.46}$$

where $P[n]$ is the phase accumulator, $M[n]$ is the modulation sequence and $P_m[n]$ is the resulting modulated phase word. Typically this is used to efficiently generate various phase shift keying techniques such as Binary Phase Shift Keying (BPSK) and Quadrature Phase Shift Keying (QPSK). By controlling the phase of the synthesized waveform, information can be encoded in the synthesized signal for transmission over noisy mediums. Figures 1.4a and 1.4b show the transient waveforms of continuous wave (CW) signals modulated with BPSK sequences. This is not to be confused with frequency modulation, which is discussed next in Section 1.3.2.



(a) BPSK for 1110010 Sequence

(b) BPSK for 1010101 Sequence

Figure 1.4: BPSK Waveforms

In the radar architecture described in Chapter 6, Barker codes are hardwired into the PM circuitry for testing. Table 1.1 shows the list of Barker codes [8] built into the DDFS. These and other more complex codes were implemented in the radar system for "short-range", low-power pulse compression operating modes. The phase is flipped every k clock

Table 1.1: Built-in Barker Codes

Number Of Bits	Code
3	110
5	11101
7	1110010
11	11100010010
13	1111100110101

cycles by 180 degrees if a binary "1" is encountered and is not flipped if a "0" is encountered. The hardware implementation is as simple as toggling the MSB of the phase accumulator, which only requires an XOR gate.

1.3.2 Digital Frequency Modulation

Linear frequency modulation (LFM), also known as *chirp* modulation, can be implemented by adding a frequency accumulator before the phase accumulator. Figure 1.5 shows a block diagram implementation of the modification that should be made to the accumulator to allow for LFM. A full derivation of LFM is provided in Section 5.2.2, but in short, the

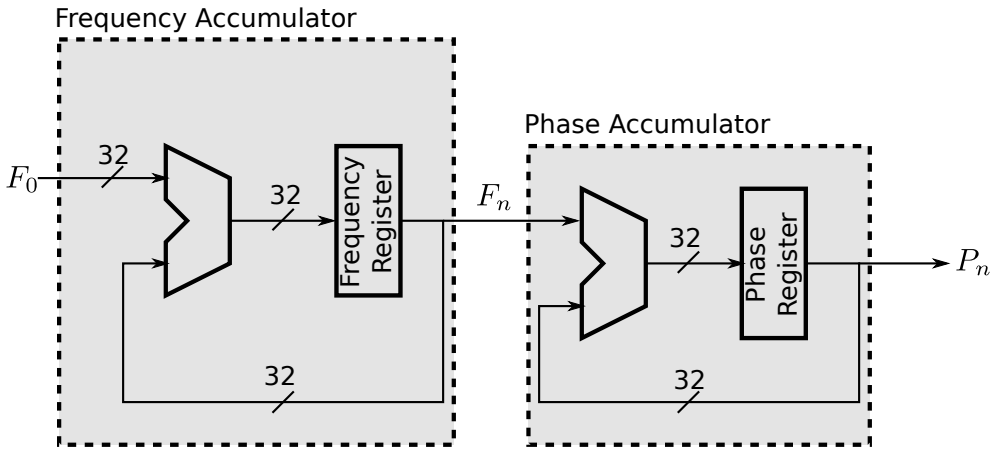


Figure 1.5: Simple Chirp Accumulator Diagram

frequency of the DDFS increments by F_0 every clock cycle. While not shown explicitly in the figure, the frequency register is typically initialized to some start frequency and other control circuitry watches the frequency register value to issue a stop command. The chirp rate is controlled with the same precision and agility as the frequency of a traditional phase accumulator. Figure 1.6 shows a 10 ns duration chirp waveform with frequency accelerating from around 500 MHz to 5 GHz.

The chirp waveform is well suited for radar application requiring pulse compression. In Chapter 6, LFM is implemented in a stretch processing pulse compression radar. Figure 6.14a shows several conceptual chirp waveforms in the explanation of stretch processing. The fine control of the chirp rate allows the radar system to dynamically tailor its output waveform based on the distance of the target under investigation.

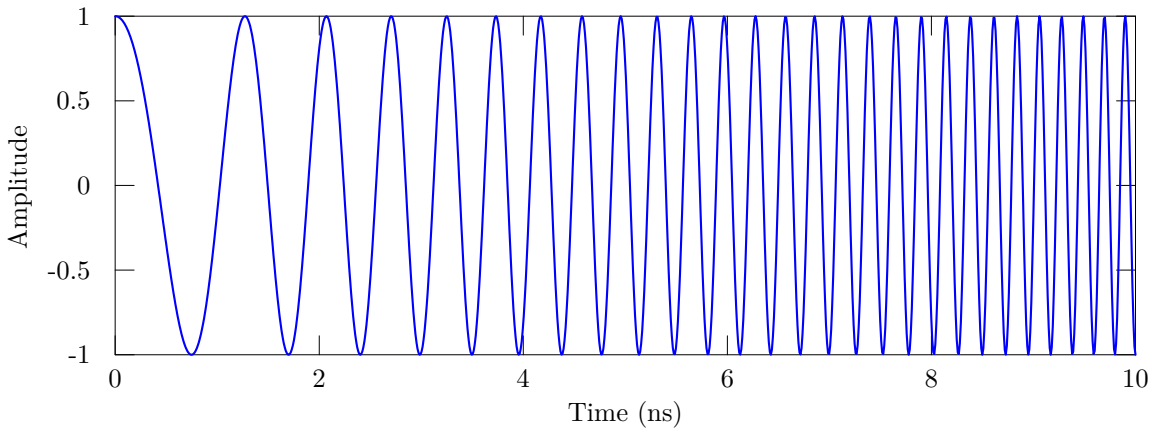


Figure 1.6: 10ns Chirp Waveform

1.3.3 Digital Amplitude Modulation

By placing a multiplier at the output of the SCMF, the amplitude of the output can be digitally modulated. This operation is important for implementing quadrature amplitude modulation (QAM) schemes such as QAM16 and QAM64, which are commonly used in digital communications systems such as fiber and cable internet [18]. The operation is performed in the digital domain, so implementing higher order (such as QAM 256) or more complex modulation schemes is feasible without much added overhead to the DDFS system, though the DAC and filter requirements become increasingly difficult.

In more complex systems, a finite impulse response (FIR) filter can be added to the output of the SCMF. This filter can be used to pre-distort the signal before driving the DAC to compensate for the non-idealities of the following analog circuitry, or of the non-linearity of the DAC itself. In Section 6.5.1, the design of an inverse sinc filter is shown that is applied to the radar system of Chapter 6. The inverse sinc FIR compensates for the zero-order hold operation of a traditional current steering DAC.

1.3.4 Fine Frequency Resolution and Fast Switching

One of the most cited benefits to DDFS designs is fine frequency tuning [7]. Equation 1.45 provides the frequency of a DDFS given an FCW F. Informally, the resolution of a DDFS device is the difference in synthesized frequency between two adjacent FCWs. This can be calculated by setting $F = 1$

$$f_0 = \frac{F}{N_P} f_{\text{clk}} = \frac{f_{\text{clk}}}{N_P} \tag{1.47}$$

A 32-bit phase accumulator is a value commonly found in commercial DDFS parts [19],[20]. For a 1 GHz clock frequency, this results in a frequency resolution of $10^9/2^{32} \approx 0.23\,\text{Hz}$. This also implies that the LFM discussed in Section 1.3.2 is capable of generating remarkably smooth chirp waveforms when given enough bits of resolution.

The DDFS can also rapidly switch between different output frequencies. This is of critical importantance in spread spectrum applications, where the speed of the frequency switching directly impacts the performance of the system. Using the DDFS as a local oscillator allows one to quickly switch to a different band, as quickly as the analog filter can respond to the changes. Combining the fine frequency resolution and fast frequency switching makes for a versatile solution for demanding problems from a host of fields [19], ranging from biomedical to military.

1.4 Summary of Contributions and Chapter Breakdown

This section summarizes the contributions of the author to the state of the art in the analysis of phase truncation spurs and to DDFS design in general. In Chapter 3, a complete approach for calculating the least period of the output sequence of a phase accumulator with truncation is derived. This approach is completely general and does not depend on the number of states in the phase accumulator to be a power of two. To the author's knowledge, there are no publications that perform this analysis. In Chapter 2, the most frequently cited

published analyses on phase truncation spurs are presented in chronological order. The notation between the analysis is unified. Again, the author is not aware of any such analysis available in published literature. By comparing the methodologies, it becomes clear what can and what cannot be computed using each methodology and how strongly analyses are dependent upon their predecessors' research.

In Chapter 4, an exact and fully general analysis of phase truncation spurs is developed. The technique generalizes and builds upon Torosyan's work on phase truncation spurs and is not subject to the same limitations. Furthermore, the approach on computing the closed form expression for phase truncation spurs is completely quantitative and does not depend on any approximations or intuition to arrive at the results. This is to say that the analysis is a direct computation of the discrete Fourier transform on the output of a DCDO. Here it is further shown how the spectrum of the quantized amplitude waveform is mostly independent of the phase truncation spurs. Application of the theory is applied to the author's previous published design [21] with suggestions for improvements. In Chapter 5, a novel approach to parallelizing a phase accumulator with frequency modulation is presented. It covers several patents on the topic, as there is not much in the way of academic publications.

In Chapter 6, the DDFS designs at Auburn University and their application in a simple single-chip radar system is explored [22]. The design attempted the challenging task of co-locating a radar transmitter and receiver onto the same silicon substrate. A quadrature DDFS generated the radar waveform and pulse compression sequence. Lastly, in Chapter 7, a survey of literature highlights many of the problems in the DAC designs at Auburn University and offers a suggestions for improving DAC designs based on observations made during the survey. Some of the analysis and observations in the chapter have not been published to the author's knowledge.

Chapter 2

Background of Phase Truncation Analysis

In this chapter, an overview of the literature surrounding the analysis of the performance of DDFS devices is presented. Despite widely cited publications by Nicholas [23], Jenq [24] and Torosyan [10] on the analysis of phase truncation spurs in DDFS, papers are still published (or submitted for publishing) with either old approximations of the spurious behavior for which concise closed-form equations exist, or, worse yet, incorrect reasoning related to the location, magnitude and optimization of such spurs.

Oftentimes early, widely cited papers continue to propagate while newer analyses go unnoticed. Two authors, Jenq and Torosyan, through two separate mathematical techniques, provide a closed-form solution to the spurs generated from phase truncation. Jenq's analysis can be elegantly used to compute the signal-to-noise ratio in the presence of phase truncation but does not compute the location of the spurs. Torosyan goes a step further, presenting an elegant algorithm for efficiently calculating *all* of the spurs that results from phase truncation in the order of magnitude.

An attempt is also made in this chapter to unify several of the previous published techniques by using a consistent notation between techniques. The publications on phase truncation errors span nearly three decades and use various mathematical approaches for deriving the spurious content. The techniques are presented chronologically from publication date.

2.1 Mehrgardt's Analysis (1983)

Mehrgardt [25] attempts to explain the non-intuitive spectral output of a signal generated from a finite length sinusoidal lookup table. The critical observation in the analysis is

that the phase word can be written as the sum of two sawtooth waveforms, the truncated phase word and kept phase word.

$$\hat{\theta}[n] = \theta[n] - \Delta\theta[n] \tag{2.1}$$

where $\hat{\theta}[n]$ is the phase word after truncation, $\theta[n]$ is given by Equation 1.22 and $\Delta\theta[n]$ is the value of the bits truncated after mapping. The truncated phase word drives an N_Q entry sinusoidal lookup table with values

$$A[n] = \sin\left(\frac{2\pi}{N_Q}n\right), n \in \{0, \ldots, N_Q - 1\} \tag{2.2}$$

where $N_Q = N_P/N_E$ as in Chapter 3. Note that no amplitude truncation is applied to the table values, meaning the analysis uses an ideal SCMF. The spurious response will be generated completely from phase truncation.

Mehrgardt decides to tackle the problem by considering an analogous system in the continuous time domain. Consider the function below

$$\begin{aligned} S(t) &= \sin\left(\frac{2\pi}{N_P}\left[N_P f t - N_E x_{sw}(t)\right]\right) \\ &= \sin\left(2\pi f t - \frac{2\pi}{N_Q}x_{sw}(t)\right) \end{aligned} \tag{2.3}$$

where $x_{sw}(t)$ is a sawtooth waveform of amplitude 1 and frequency $N_Q f$ where $f = \frac{F}{N_P} f_{\text{clk}}$ is the desired frequency of the synthesized tone. N_E is the number of error states and N_P is the number of phase states, as discussed in Chapter 3. The sawtooth waveform can be represented mathematically as

$$x_{sw}(t) = N_Q f t - \lfloor N_Q f t \rfloor \tag{2.4}$$

where $\lfloor \cdot \rfloor$ is the real domain truncation operation that maps an real number r to the nearest integer that has a value less than r. That this is a reasonable approximation for the behavior of the phase accumulator can be shown without too much analysis. As an example, let $T_{\text{clk}} = 10^{-9}$ s, $F = 3$, $N_P = 2^7$ and $N_E = 2^5$. Figure 2.1 is a plot of the phase accumulator values along with the continuous time approximation of the phase from Equation 2.3. The phase accumulator overflows approximately every $T_{clk} N_P / F$ seconds or at a rate of $F / (T_{clk} N_P)$. Notice the actual digital error values look like samples of the continuous time function used

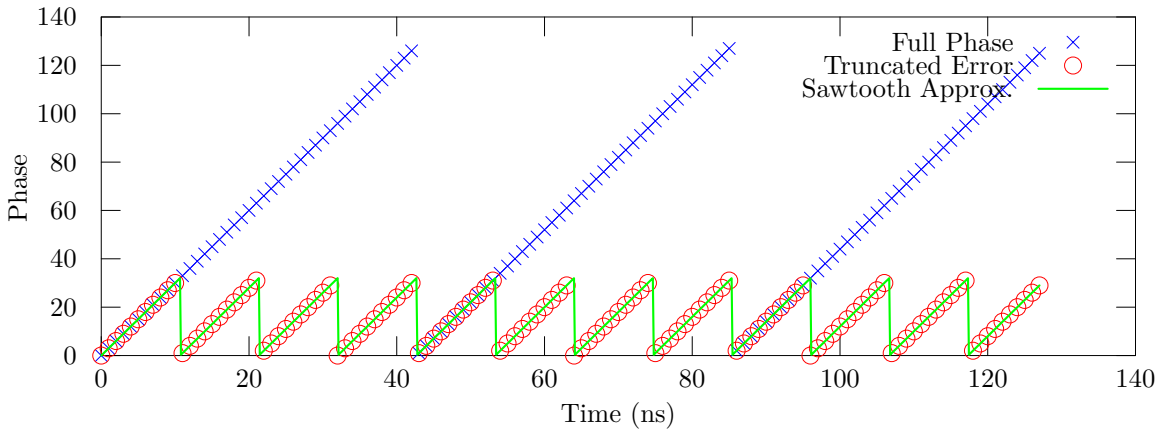


Figure 2.1: Sawtooth Approximation

in the analysis.

Applying the difference trigonometric identity for sine, Equation 2.3 can be rewritten as

$$S(t) \approx \sin(2\pi f t) \cos\left(\frac{2\pi}{N_Q} x_{sw}(t)\right) - \cos(2\pi f t) \sin\left(\frac{2\pi}{N_Q} x_{sw}(t)\right) \tag{2.5}$$

Now in general, $N_E \ll N_P$ making N_Q large and therefore the small angle approximation for sine and cosine can be applied on Equation 2.5.

$$S(t) \approx \sin(2\pi f t) - \cos(2\pi f t) \left[\frac{2\pi}{N_Q} x_{sw}(t)\right] \tag{2.6}$$

Since $x_{sw}(t)$ is periodic, the Fourier series can be computed but it must be noted that $x_{sw}(t)$ is not a Dirichlet sawtooth because it does not take the value of the mid-point at discontinuities. Consequently,the Fourier series does not actually exist for $x_{sw}(t)$ as described by Equation 2.4. This oversight is corrected in Nicholas's analysis by adding a periodic pulse train that forces the sawtooth to take the value of the midpoint at discontinuities. Regardless, $x_{sw}(t)$ is *almost* a Dirichlet sawtooth and thus the Fourier series of the sawtooth is used to represent it. First, the definition of the Fourier series is presented.

Definition 2.1 (Fourier Series of Real-Valued Function)**.** *The Fourier series of a real valued function is defined as*

$$\mathcal{F}_s\left\{f(x)\right\} = \sum_{n=0}^{\infty}\left[a_n\cos\left(nx\right) + b_n\sin\left(nx\right)\right] \tag{2.7}$$

where the coefficients of Equation 2.7 are computed using the inner product described below

$$a_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f\left(x\right)\cos\left(nx\right)dx, \quad n \geq 0 \tag{2.8}$$

$$b_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f\left(x\right)\sin\left(nx\right)dx, \quad n \geq 0 \tag{2.9}$$

Without performing the full calculation, the Fourier series of a sawtooth waveform is

$$x_{sw}(t) \approx \sum_{k=1}^{\infty}(-1)^k\frac{\sin\left(2\pi k N_Q f t\right)}{\pi k} \tag{2.10}$$

Equation 2.10 is an approximation because the constant component has been ignored, as it does not contribute to the spurious response of the waveform. Substituting the Fourier series of $x_{sw}(t)$ back into Equation 2.6, the derivation of the spectrum of the continuous time analogy is complete. Note that the product to sum identity was used in the derivation.

$$S(t) \approx \sin\left(2\pi f t\right) - \sum_{k=1}^{\infty}\frac{\left(-1\right)^k}{N_Q k}\left[\sin\left(2\pi\left(kN_Q+1\right)ft\right) + \sin\left(2\pi\left(kN_Q-1\right)ft\right)\right] \tag{2.11}$$

At this point, $S(t)$ must be sampled to get back to the discrete time DCDO case. Mehrgardt does this is stages, starting with sawtooth waveform. The sampling process is executed by replacing f with its discrete value and substituting n/f_{clk} for t. Applying the sampling process to $x_{sw}(t)$ first, the following equation is derived.

$$x_{sw}[n] = \sum_{k=1}^{\infty}(-1)^k \frac{\sin\left(2\pi k N_Q \left(\frac{F}{N_P} f_{clk}\right)\left(\frac{n}{f_{clk}}\right)\right)}{\pi k} \tag{2.12}$$

$$= \sum_{k=1}^{\infty}(-1)^k \frac{\sin\left(2\pi k F n/N_E\right)}{\pi k} \tag{2.13}$$

Now F and N_E can be reduced by removing common factors, using the modular arithmetic of the previous chapter such that $F/N_E = \Gamma_E/\Lambda_E$ (Lemma 3.2). If following along using Mehrgardt's publication, one will notice the analysis in this work has diverged. In particular, he makes the statement that the finite precision frequency can be written as $2\pi a/b$ where a and b are not defined but exist and places the final results in these terms. In this analysis, the introduction of the extra terms are not necessary because the meaning of the symbols has been carefully tracked. Note that the sinusoid in the sawtooth equation is periodic in k with Λ_E. After several summations and trigonometric identities, which are left as an exercise to the reader in the original publication from Mehrgardt and so will also be done so here, a final result is achieved,

$$S[n] = \sin\left(2\pi \frac{F}{N_P} n\right) - \frac{\pi}{N_Q} \sum_{m=1}^{(\hat{N_P}-1)/2} \alpha_{k_m} \cdot$$
$$\left\{\sin\left(2\pi\left(\frac{m}{\hat{N_P}} + \frac{F}{N_P}\right)n\right) + \sin\left(2\pi\left(\frac{m}{\hat{N_P}} - \frac{F}{N_P}\right)n\right)\right\} \tag{2.14}$$

where k_m is calculated through the following relation

$$k_m = \langle m k' \rangle_{\hat{N_P}} \tag{2.15}$$

and where k' is the solution for k in the linear congruence relation

$$\left\langle k\hat{N}_Q F \right\rangle_{\hat{N}_P} = 1 \tag{2.16}$$

Lastly, the coefficients are of the form

$$\alpha_k = \begin{cases} \frac{(-1)^k}{\hat{N}_P \sin\left(\pi k/\hat{N}_P\right)} & \text{for } \hat{N}_P \text{ odd} \\ \frac{(-1)^k}{\hat{N}_P \tan\left(\pi k/\hat{N}_P\right)} & \text{for } \hat{N}_P \text{ even} \end{cases} \tag{2.17}$$

The result is never related back to the FCW to characterize the behavior of the DCDO with phase truncation or calculate the SFDR or SNR. The result here shows the relationship of the FCW and spurs because of the changes made in symbolic notation in the derivation. Only qualitative observations such as the number of spurs, the magnitude of spurs and that such spurs should be expected are presented.

As a summary, this technique provides:

- the number of phase truncation spurs and
- the magnitude of the phase truncation spurs.

It misses some spurs because of its failure to use a Dirichlet sawtooth waveform in its analysis. The author makes the observation of the discrepancy between his analysis and simulation in the later paragraphs of his paper.

2.2 Nicholas's Analysis (1985)

H. T. Nicholas provides [23] one of the most well-known analyses of phase truncation spurs. The analysis provides equations for the location, phase and amplitude of the spurs generated through phase truncation. The general idea is similar Mehrgardt's analysis described in Section 2.1, in that the phase error is thought of as a sawtooth waveform. Nicholas takes a more formal mathematical approach and finds several clever trigonometric reduction

techniques to bring about a final result that is concise, accurate and efficient in implementation. The results are summarized by the following theorems which are presented without proofs.

To summarize the steps taken by Nicholas:

1. Find the analogous continuous time representation of the phase and the truncation error sequence. This takes the form of a pulse train

$$p_e(t) = \frac{N_E}{\Lambda_E} \sum_{k=1}^{\infty} \frac{\Lambda_E}{\pi k} \sin\left(\frac{\pi k}{\Lambda_E}\right) \cos\left(2\pi k \frac{F}{N_E} t\right) + \frac{1}{2} \tag{2.18}$$

   and a sawtooth that meets Dirichlet conditions

$$x_{sw}(t) = \sum_{k=1}^{\infty} \frac{N_E}{\pi k} \sin\left(2\pi k \frac{F}{N_E} t\right) + \frac{N_E}{2} \tag{2.19}$$

   Note that these waveforms are slightly different than those shown in the publication. If plotting the waveforms exactly from the publication, one will not get the correct error sequence. This is because Nicholas removed the DC term from both the pulse train, which is mentioned in the publication, **and** the sawtooth waveform. However, in the plots in the publication, the DC term is added back.

2. Sample the continuous time representation by the DDFS sample rate, which is performed in the same manner described in Section 2.1:

$$p_e[n] = \frac{N_E}{\Lambda_E} \sum_{k=1}^{\infty} \frac{\Lambda_E}{\pi k} \sin\left(\frac{\pi k}{\Lambda_E}\right) \cos\left(2\pi k \frac{F}{N_E} n\right) + \frac{1}{2} \tag{2.20}$$

   and the Dirichlet sawtooth waveform

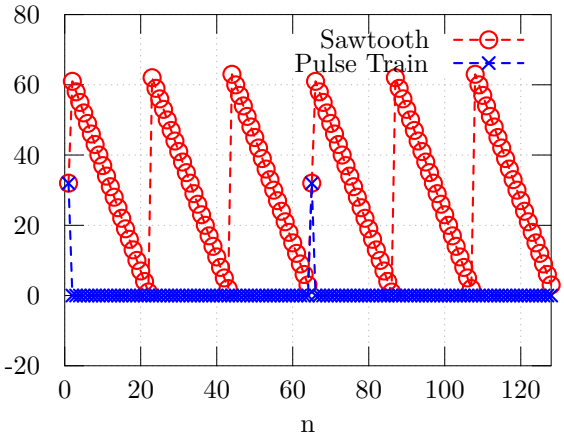
$$x_{sw}[n] = \sum_{k=1}^{\infty} \frac{N_E}{\pi k} \sin\left(2\pi k \frac{F}{N_E} n\right) + \frac{N_E}{2} \tag{2.21}$$

The truncation error sequence can then be reconstructed by subtracting the pulse train from the sawtooth waveform

$$e_p[n] = x_{sw}[n] - p_e[n] \tag{2.22}$$

Figure 2.2a shows the sawtooth waveform and pulse train waveform for $N_E = 64$ and $F = 3$. Figure 2.2b shows the subtraction of the pulse train from the sawtooth waveform, yielding the the very familiar phase truncation error sequence plotted in Figure 2.1.

(a) Nicholas' Sawtooth and Pulse Train



(b) Nicholas' Phase Error Sequence

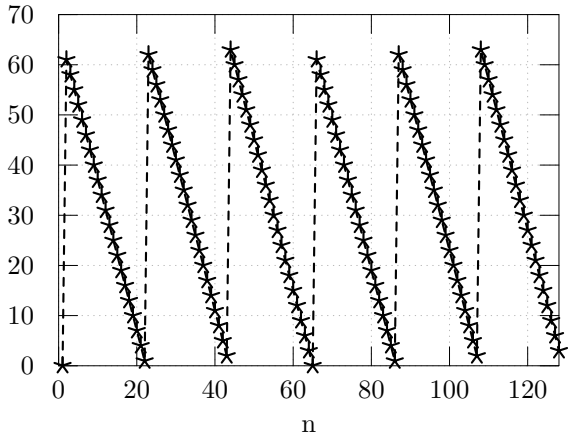


Figure 2.2: Error Sequence Waveform Components

3. Perform an spectacularly complex set of trigonometric manipulations found in mathematics texts dedicated to the task, Nicholas arrives at his final error sequence equation

$$e_p[n] = \frac{-N_E}{\Lambda_E} \sum_{k=1}^{\Lambda_E/2} \left[\cot\left(\frac{\pi k}{\Lambda_E}\right) \sin\left(2\pi k \frac{F}{N_E} n\right) - \cos\left(2\pi k \frac{F}{N_E} n\right) \right] \tag{2.23}$$

4. Use number theory to relate the spurs in k to the frequency control word.

5. Plug the error sequence into the original sine expression and use the small angle approximation. This is where Nicholas' approach becomes an approximation to the spurious behavior (albeit a very good one).

6. Perform an enormous amount of trigonometric manipulations and number to theory to arrive at the final answer.

The results from Nicholas' work are summarized in the following set of theorems. The number of spurs due to phase truncation is provided in Theorem 2.1

Theorem 2.1 (Nicholas Number of Spurs)**.** *An accumulator with B_P bits whose least B_T bits are truncated has $(\Lambda_E/2-1)$ spurs. Or written in notation used previously in the document, an accumulator with N_P phase states and N_E error states such that $N_E \mid N_P$ has $(\Lambda_E/2-1)$ spurs.*

The mapping from the κ (spur index) value to the frequency control word is provided by Theorem 2.2.

Theorem 2.2 (Nicholas Spur Index)**.** *The **spur index**, κ, for the spur located at the DFT frequency bin k is, if $2 \mid (k - \Lambda_E/2)$*

$$\kappa = \left\langle \frac{k - \Gamma_E}{2^{B_P - B_T}} \Gamma_E^{(\Lambda_E/2-1)} \right\rangle_{\Lambda_E} \tag{2.24}$$

where Γ_E is the reduced frequency control word for the error sequence (Lemma 3.3)

$$\Gamma_E = \frac{F}{GCD\,(F, N_E)} \tag{2.25}$$

and $N_E = 2^{B_T}$ is the modulo number for the error sequence. If $2 \mid (-k - \Lambda_E/2)$,

$$\kappa = \left\langle \frac{-k - \Gamma_E}{2^{B_P - B_T}} \Gamma_E^{(\Lambda_E/2-1)} \right\rangle_{\Lambda_E} \tag{2.26}$$

The magnitude of the spurs given a spur index is provided by Theorem 2.3.

Theorem 2.3 (Nicholas Spur Magnitude)**.** *The magnitude of the spur at spur index κ is*

$$m_\kappa = \frac{\pi 2^{B_T - B_P}}{\Lambda_E} \operatorname{cosec}\left(\frac{\kappa\pi}{\Lambda_E}\right) \tag{2.27}$$

Theorem 2.4 (Nicholas Spur Phase)**.** *The phase of the spur at spur index κ is*

$$\phi_\kappa = -\cot\left(\frac{\kappa\pi}{\Lambda_E}\right) \tag{2.28}$$

As mentioned in above, the analysis starts in a similar vein as Mehrgardt, but with one critical difference in the specification of the error sawtooth waveform. Nicholas notes that the Dirichlet conditions for the Fourier series of a function with discontinuities require that the function take on the average value of the function at the point of discontinuity. He thus adds a separate pulse train function to create a sawtooth error function that can be analyzed by using the Fourier series. In Mehrgardt's publication, he notes that there are extra terms in the result that resulted from insufficiencies of the sawtooth approximation to the actual error function behavior. Nicholas avoids those terms by creating a pair of continuous time functions that have convergent Fourier Series.

To demonstrate the importance of this dissertation, outside of avoiding trigonometric small-angle approximations, the final discrete time equation in Nicholas' analysis is

$$\begin{aligned} A[n] = \sin\left(2\pi\frac{F}{N_P}n\right) - \sum_{k=1}^{\Lambda_E/2}\left(\frac{\pi}{\Lambda_E N_Q}\operatorname{cosec}\left(\frac{k\pi}{\Lambda_E}\right)\right)\cdot \\ \left(e^{j2\pi\mathrm{GCD}(F,N_E)(\Gamma_E - k\Gamma_E N_Q)n/N_P} + e^{-j2\pi\mathrm{GCD}(F,N_E)(\Gamma_E - k\Gamma_E N_Q)n/N_P}\right)\cdot \\ e^{-j\cot(k\pi/\Lambda_E)} \end{aligned} \tag{2.29}$$

Compare this equation with the closed form solution from this document in Chapter 4 under Theorem 4.6. The small angle approximation prevents simplification to an elegant final form.

Lastly, the worst case spur magnitude is predicted by the following equation. This flows from the observation that the spur magnitude is a strictly decreasing function of the spur

index. So selecting a spur index of 1 yields

$$m_{wc} = \frac{1}{N_Q} \frac{\frac{\pi}{\Lambda_E}}{\sin\left(\frac{\pi}{\Lambda_E}\right)} \tag{2.30}$$

2.3 Jenq's Analysis (1988)

Y. C. Jenq published a series of papers on analyzing the spectrum of non-uniformly sampled signals [26, 24, 27, 28]. Re-deriving the analysis as it pertains to modern DCDO's is worthwhile, as the notation used in [24] is decidedly different than that commonly used in DDFS literature. There is one critical difference between Jenq's analysis and the following derivation: Jenq uses a continuous time domain representation for the output of the DCDO. Using the notation for a uniformly sampled discrete time signal, the analysis is more easily accessible to one familiar with digital signal processing (DSP) without having to concern one's self with integrals and the Dirac delta function.

Before presenting the theorem, the discrete time Fourier transform (DTFT) is given in Equation 2.31.

$$X(\omega) = \sum_{n=-\infty}^{\infty} x[n] e^{-j\omega n} \tag{2.31}$$

The DTFT can be derived from the continuous time Fourier transform, or just Fourier transform, described in Section 7.2. The DTFT is used to compute the spectrum of waveforms discretized in the time domain, which is precisely the types of waveforms considered in this work.

Theorem 2.5 (Jenq's Non-Uniform Sampling Theorem)**.** *The DTFT of a signal formed by non-uniformly sampling a waveform $g(t)$ that is band-limited between $\left(\frac{-1}{2T}, \frac{1}{2T}\right)$ in such a way that the **overall** sampling process repeats with period $M \cdot T$ is given by*

$$G(\omega) = \frac{1}{MT} \sum_{m=0}^{M-1} \left[\sum_{k=-\infty}^{\infty} G_0\left(\omega - k\left(\frac{2\pi}{MT}\right)\right) e^{j2\pi\omega k t_m/(MT)} \right] e^{jm\omega T} \tag{2.32}$$

where t_m is the uniform sampling rate of a sub-sequence of the sampled $g(t)$ by taking every Mth sample (it is uniform because the sampling process is periodic with MT), ω is the angular frequency of the waveform in radians and G_a is the CTFT of $g(t)$.

This theorem is not difficult to show. The reason that Jenq gets cited in DDFS literature is an interesting observation made about the behavior of the truncated phase accumulator output.

2.3.1 Jenq's Observation

The key observation made by Jenq in his analysis (Section 2.3) is that in the presence of phase truncation the phase code sent to the SCMF is not uniformly spaced. Effectively this "looks like" a non-uniform sampling operation on the generated sinusoid. But the phase accumulator is of finite length, so regardless of phase truncation, it is periodic and therefore the truncated phase word is also periodic. In Section 1.2, the periodicity of the phase accumulator given a frequency control word F is given in Equation 3.12.

Now consider a four bit phase accumulator incremented by $F = 3$ that is truncated to three bits and fed into an ideal SCMF (i.e. assume there is no quantization in the amplitude value stored in the SCMF). Thus in the example, the number of bits truncated from the phase word is $B_T = 1$, the number of bits in the phase accumulator is $B_P = 4$ and the number of kept bits after truncation is $B_{PT} = 3$. Table 2.1 shows the state of the phase accumulator, the truncated phase word, and the phase step between adjacent phase words.

Note that the leftmost column of the table matches the predicted untruncated phase sequence of Theorem 3.2 and also the second column shows the predicted truncated phase sequence of Theorem 3.4. While the proofs should be sufficient, working through an example provides a helpful check on the result and some intuition into the behavior of the devices modeled by the mathematical analysis. There are a few observations that can be made from the sequence.

Table 2.1: Table of Truncated Phase States (4-bit)

Phase Accumulator	Truncated Phase	Truncated Phase Step
0000 (00)	000 (0)	-
0011 (03)	001 (1)	1
0110 (06)	011 (3)	2
1001 (09)	100 (4)	1
1100 (12)	110 (6)	2
1111 (15)	111 (7)	1
0010 (02)	001 (1)	2
0101 (05)	010 (2)	1
1000 (08)	100 (4)	2
1011 (11)	101 (5)	1
1110 (14)	111 (7)	2
0001 (01)	000 (0)	1
0100 (04)	010 (2)	2
0111 (07)	011 (3)	1
1010 (10)	101 (5)	2
1101 (13)	110 (6)	1
0000 (00)	000 (0)	2

- As is the case in Figure 3.1b, the phase accumulator is periodic with $N_P = 2^4 = 16$ clock cycles.
- The truncated phase sequence is also periodic with the phase accumulator. While already stated, working through a simple example helps visualize the periodicity.
- The truncated phase step is not uniform. It varies between the values of 1 and 2.

An interesting feature follows from this analysis. Since the truncated phase sequence is periodic, the delta phase cycle is also periodic and the periodicity of the delta phase cycle is the same as the periodicity of the phase error sequence from truncation. From Lemma 3.5, if $2^{B_T} | N_P$, which is does in our example, then

$$\Lambda_E = \frac{2^{B_T}}{\text{GCD}\,(F, 2^{B_T})} = \frac{2}{\text{GCD}(1, 2)} = 2 \tag{2.33}$$

2.3.2 Jenq's Results

Jenq does not deal with spur locations, magnitudes or phases in any of his publications. Instead the non-uniform sampling theorem is applied and then Parseval's relation is used to find a noise power boundary. This allows for a completely general calculation of the signal to noise ratio due to phase truncation spurs. He describes the problem in a manner differently than any of the authors, by thinking of the phase accumulator value as an integer value plus a coprime rational number. Variables W, L and M are introduced to describe the output of the phase accumulator, where L and M are coprime.

$$d = W + L/M \tag{2.34}$$

Looking at Equation 3.38 the expression can be rewritten in the notation used in this work. M is periodicity of the error sequence (Λ_E), as multiplying d by M yields an integer value. L is the reduced FCW over M, that is to say $L = \Gamma_E$. Applying the non-uniform sampling theorem gives the an answer in the time domain.

$$G(\omega) = \frac{2\pi}{T\Lambda_E} \sum_{k=-\infty}^{\infty} \left(\sum_{m=0}^{\Lambda_E - 1} e^{-j2\pi r_m f_0 / f_{clk}} e^{-j2\pi km/\Lambda_E} \right) \delta \left[\omega - \omega_0 - k \left(\frac{2\pi}{\Lambda_E T} \right) \right] \tag{2.35}$$

The amplitude of the spurs only considered at this point. Sampled the waveform yields

$$A(k) = \frac{1}{\Lambda_E} \sum_{m=0}^{\Lambda_E - 1} e^{-j2\pi \langle m\Gamma_E \rangle_{\Lambda_E} / (\Lambda_E N_Q)} e^{-j2\pi mk/\Lambda_E} \tag{2.36}$$

Definition 2.2 (Parseval's Relation)**.** *Parseval's relation for the sequence $g[n]$ of length N*

$$\sum_{n=0}^{N-1} |g[n]|^2 = \frac{1}{N} \sum_{k=0}^{N-1} |G[k]|^2 \tag{2.37}$$

Applying Parseval's relationship yields the final result for the signal to noise ratio

$$\mathrm{SNR} = 10\log_{10}\left[\frac{|A(k)|^2}{1-|A(k)|^2}\right] \tag{2.38}$$

Several observations can be made about how the function increases and decreases with various values of Λ_E and Γ_E. Using this information, the best and worst case signal to noise ratio is computed. The worst case is calculated as

$$\mathrm{SNR}_{wc} = 10\log_{10}\left[\frac{\left[\sin\left(\frac{\pi}{N_Q}\right)\left(\frac{N_Q}{\pi}\right)\right]^2}{1-\left[\sin\left(\frac{\pi}{N_Q}\right)\left(\frac{N_Q}{\pi}\right)\right]^2}\right] \tag{2.39}$$

and the best case is derived as

$$\mathrm{SNR}_{bc} = 20\log_{10}\left[\cot\left(\frac{\pi}{2N_Q}\right)\right] \tag{2.40}$$

For the details of the derivation, consider reading Jenq's series of non-uniform sampling papers.

2.4 Torosyan's Analysis (2001)

Although an excellent analysis of the location and magnitude of phase truncation spurs existed at the time of his publications, Arthur Torosyan and Alan Willson of UCLA provided an **exact**, **clear** and **practical** means for an analytical understanding of phase truncation spurs [29, 10]. Instead of working from analogous, time domain functions such as Nicholas and Mehrgardt, Torosyan approaches the problem using elementary number theory. The critical observation is that any two FCWs that generate periodic phase sequences of the same length, the resulting sequences can be related through a simple rearrangement and that the frequency responses of sequences related in this manner are also simple rearrangements of each other. The exact same observation is actually made by Nicholas in [23] but is not used in his analysis.

The implication is enormous for DCDO analysis, as the DFT need only be run by number of prime factors of N_P times and all other frequency domain responses can be generated by permuting the frequency response. Consider the case of a $B_P = 32$ bit accumulator. If every FCW should be tested, then the DFT would need to run $N_P = 2^{32} = 4294967296$ times on sequences of at least half are periodic with N_P. The current state of computer technology simply cannot handle the computational requirements of such a task in a manageable amount of time (i.e. if each DFT of such a sequence took one second, 4294967296 s is equal to approximately 136 years). It is great news then that mathematicians have shown that such computations are not required to fully characterize DCDOs.

Much of the work in Chapter 4 will be used in this analysis, so reading that chapter before continuing may be helpful. The analysis begins by showing that FCWs of the same period generate sequences that are simple permutations of each other (Theorem 3.9). Then it will be shown that the frequency domain representation of such sequences are also simple permutations of each other (Theorem 4.8). Then the DFT of the $\Gamma = 1$ case is computed (Theorem 4.6 with $\Gamma = 1$). Since the all sequences can be generated as permutations of $\Gamma = 1$, the analysis is complete. To summarize the results:

1. Only one frequency control word for each possible phase accumulator least period needs to be considered. Let this FCW be chosen such that after reducing the word modulo the number of phase accumulator states, the resulting number is 1 (i.e. $\Gamma = 1$).

2. All other frequency control words for given least period are permutations of the previously described FCW.

3. The DFT happens to commute with a set of vectors for a given least period and therefore the frequency spectrums are also permutations. This leads to the "window function" after simplifying the result.

4. An interesting simplification can be made when observing the reduced word 1, allowing the grouping of repeating terms and resulting in a simple final expression for the phase truncation spurs.

Chapter 3

Phase Accumulator Sequences from Number Theory

In this chapter, a complete analysis of the sequences generated by a phase accumulator is developed from elementary number theory. The analysis begins by proving the well-known untruncated phase accumulator state equation using basic modulo arithmetic (Section 3.1). The number theoretic techniques may seem excessive at this point, but the motivation for approaching the problem from a mathematical standpoint will eventually become apparent in Chapter 4. Several concepts from number theory are presented to support the analysis from the previous section and will be used in later proofs as the work progresses. After determining the expected phase accumulator sequence, the periods of such sequences are explored (Section 3.2). Several Greek letters are given special meaning, such as the reduced FCW Γ and the least period length Λ.

The effects of truncation on the phase accumulator sequence are explored in Section 3.3. The tools developed for deriving the period of an untruncated phase sequence are applied to the truncation problem resulting in a general theory for the periodicity of both the truncated phase sequence and the error sequence. Developing a pure mathematical framework for analyzing the phase accumulator prevents common mistakes in calculating the periodicity of DDFS waveforms. The relationship between both untruncated and truncated phase sequences of different FCWs is established in Section 3.4. Lastly, some comments on abstract algebraic structures that can be used to fully describe the operation of the phase accumulator are presented in Section 3.5.

3.1 Phase Accumulator Sequence

The state of the phase accumulator at clock cycle n can be written as a function of the FCW and the initial phase,

$$P[n] = \langle nF + P_0 \rangle_{N_P} \tag{3.1}$$

where P_0 is the initial state of the phase accumulator, F is the FCW and $\langle a \rangle_m$ represents the smallest non-negative integer remainder when dividing a by m (sometimes called the least residue). This is commonly referred to as the modulo m operator on the integer a. The modulo operation is clearly intimately related to integer division by definition. The division algorithm for positive integers is stated below [30]:

Theorem 3.1 (The Division Algorithm)**.** *Given non-negative integers a and b, $b \neq 0$, there exist unique integers q and r, with $0 \leq r < b$ such that*

$$a = bq + r \tag{3.2}$$

Proof. Let $a \in \mathbb{P}_0$ and $b \in \mathbb{P}$. Let $S = \{a - bn : n \in \mathbb{Z} \text{ and } a - nb \geq 0\}$. That S is non-empty can be shown by setting $n = 0$ since $a \geq 0$ by selection. S has a least element by the well-ordering principle (Principle 1.2). Let $r = a - bq$ be least element of S where $q \in \mathbb{P}$ is a selection of n that yields the least element of S. $r \geq 0$ from the definition of S. That $r < b$ comes from the selection of the least element. If $r \geq b$, then $a - b(q + 1) \geq 0$ and would be less that $a - bq$, meaning that the least element of S was not selected. So $0 \leq r < b$ and $a = bq + r$.

To show that q and r are unique, assume $a = bq_0 + r_0$, $0 \leq r_0 < b$. Setting the equations equal

$$\begin{aligned} bq + r &= bq_0 + r_0 \\ b\,(q - q_0) &= r_0 - r \end{aligned} \tag{3.3}$$

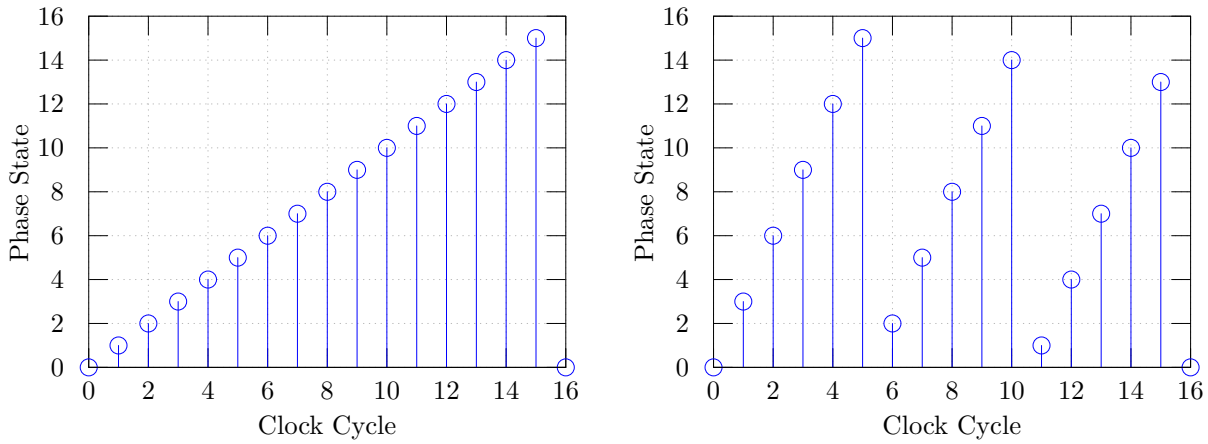
Then $b \mid (r_0 - r)$ which implies that $c_0 b = r_0 - r$, $c_0 \in \mathbb{Z}$. Since $0 \leq r_0 < b$ and $0 \leq r < b$, it follows that $-b < r_0 - r < b$. From the original selection $b \neq 0$, and the only integer choice of c_0 that produces a value between $-b$ and b is 0. Therefore $0 = r_0 - r$ and $r_0 = r$. Plugging back into Equation 3.3, $b(q - q_0) = 0$ and since $b \neq 0$, $q = q_0$. □

In summary, $\langle a \rangle_m$ returns the unique r of the previous stated division algorithm. More formally, the $\langle a \rangle_m$ operator yields the smallest positive integer r such that $a \equiv r \,(\mathrm{mod}\, m)$, or a congruent to r modulo b [30]. The latter expression forms a binary relationship on the integers described by the definition below:

Definition 3.1 (Congruence)**.** *a is congruent to b modulo m (i.e. $a \equiv b\,(mod\, m)$) if and only if $m \mid (a - b)$*

In much of this work, the initial phase is assumed to be zero as the results from analysis will follow without loss of generality. Any analysis dependent on the value of P_0 will explicitly allude to that requirement. Plotting Equation 3.1 for several clock cycles provides a clear visual explanation of the behavior of the phase accumulator and FCW. Figure 3.1a shows the progression of states versus clock cycle in a 4-bit phase accumulator with $F = 1$ and Figure 3.1b shows the progression of states for $F = 3$. Looking back at the phase circle of Figure 1.3a, one can visualize a higher FCW value "jumping over more circles" at each step (speaking in terms of the graphic).

The author is unaware of any explanation given as to why Equation 3.1 fully describes the phase state at clock cycle n. Clearly the phase accumulator has a finite number of states (N_P), so adding any F to $P[n]$, regardless of its value, must map back to a value in the



(a) 4-Bit Phase Accumulator State ($F = 1$) (b) 4-Bit Phase Accumulator State ($F = 3$)

Figure 3.1: Phase Accumulator State Plots

set $\mathbb{Z}_{N_P}$. Certainly the equation holds for Figure 3.1a and Figure 3.1b as well. But the reason the equation should hold for all F and an accumulator of any B_P is not trivially apparent (at least to the author). *All* of the phase truncation analyses in literature make use of Equation 3.1, which is not surprising since it describes the fundamental behavior of the phase accumulator.

Theorem 3.2 (Phase Accumulator Sequence)**.** *The non-negative integer value of a phase accumulator with N_P states at clock cycle n with frequency control word F and initial phase P_0 is*

$$P[n] = \langle nF + P_0 \rangle_{N_P} \tag{3.4}$$

Proof. Recall the proof that any non-negative integer can be represented by an unsigned binary number (Section 1.1.2). From the description of the phase accumulator and FCW, the repeated addition operation must also yield a non-negative number. So $nF + P_0$ is a non-negative number that can be represented by an unsigned binary number.

$$nF + P_0 = \sum_{n=0}^{\infty} 2^i b_i \tag{3.5}$$

But the phase accumulator has a finite number of bits B_P, by the description of the accumulation operation itself, the least B_P bits are kept, so

$$\begin{aligned} nF + P_0 &= \sum_{n=B_P}^{\infty} 2^i b_i + \sum_{n=0}^{B_P-1} 2^i b_i \\ &= \left(2^{B_P} b_{B_P} + 2^{B_P+1} b_{B_P+1} + \cdots\right) + \sum_{n=0}^{B_P-1} 2^i b_i \\ &= 2^{B_P}\left(b_{B_P} + 2^1 b_{B_P+1} + \cdots\right) + \sum_{n=0}^{B_P-1} 2^i b_i \end{aligned} \tag{3.6}$$

Subtracting the second summation term from both sides of the previous equation,

$$(nF + P_0) - \sum_{n=0}^{B_P-1} 2^i b_i = 2^{B_P}\left(b_{B_P} + 2^1 b_{B_P+1} + \cdots\right) \tag{3.7}$$

It is clear that 2^{B_P} divides the expression above, and we can say from the previous description of congruences that

$$(nF + P_0) \equiv \left[\sum_{n=0}^{B_P-1} 2^i b_i\right] \left(\mathrm{mod}\ 2^{B_P}\right) \tag{3.8}$$

Now to arrive at Equation 3.1, we must show that $\sum_{n=0}^{B_P-1} 2^i b_i$ is the smallest non-negative integer such that the congruence relationship above holds (additionally we should like to show that it is unique). From the definition of the modulo operator, this is equivalent to showing that the residue, r, when dividing $nF + P_0$ by 2^{B_P} is equal to $\sum_{n=0}^{B_P-1} 2^i b_i$ and that $r < 2^{B_P}$.

$$nF + P_0 = q\left(2^{B_P}\right) + r \tag{3.9}$$

Note that using Equation 3.6 immediately provides a potential r and q for the division algorithm:

$$r = \sum_{n=0}^{B_P-1} 2^i b_i \tag{3.10}$$

$$q = b_{B_P} + 2^1 b_{B_P+1} + \cdots \tag{3.11}$$

From Equation 1.10, we know that the maximum value of the r term is $2^{B_P} - 1$, so $r < 2^{B_P}$. When storing $nF + P_0$ in the finite bit phase accumulator, all the bits greater than or equal to B_P are discarded, or equivalently $b_i = 0$ for $i \geq B_P$ and therefore $q = 0$. From the division algorithm, we know that q and r must be unique. So r is the smallest non-negative integer of $nF + P_0$ divided by N_P and thus $r = \langle nF + P_0 \rangle_{N_P}$ by definition. □

Phase accumulators can also, with additional hardware, directly compute a number modulo any integer value of N_P. In fact, Jenq uses $N_P = 1000$ in his DDFS analysis [24], a number which neither Nicholas [23] or Torosyan [29] can accurately predict spurious phase truncation behavior from.

3.2 Phase Accumulator Period

Looking carefully at Figure 3.1b, an interesting observation about the periodicity of a phase accumulator is readily apparent. Though the phase accumulator overflows three times when $F = 3$, it does not return to its original starting state until the 16th clock cycle. That is to say, the phase accumulator sequence for both $F = 1$ and $F = 3$ is periodic with 16 clock cycles. The question then arises as to when the exact same phase accumulator state will repeat for an arbitrarily chosen FCW. Every explanation of the origin and nature of phase truncation spurs uses this information in its formulation. Equation 3.12 is the periodicity of the phase accumulator state.

$$\Lambda_P \triangleq \frac{N_P}{\mathrm{GCD}\,(F, N_P)} \tag{3.12}$$

where $\text{GCD}(a, b)$ is notation for the greatest common divisor (GCD) of a and b. Λ_P is the length of the sequence generated by Equation 3.1 before the phase state first repeats, or the smallest positive integer Λ_P such that $P[n + \Lambda_P] = P[n]$ for all $n \in \mathbb{Z}$. The greatest common divisor is formally defined below [15] and informally as the largest integer that divides both a and b where both a and b cannot simultaneously be zero.

Definition 3.2 (Greatest Common Divisor)**.** *Given $a, b \in \mathbb{Z}$, and both a and b not simultaneously zero, $GCD(a, b) = d$ if and only if*

1. *$d|a$ and $d|b$, and*
2. *If $c \in \mathbb{Z}$ and if $c|a$ and $c|b$, then $c|d$.*

One case, $\text{GCD}(a, b) = 1$, comes up frequently in developing the theory of linear congruences, and so a name is given to two such integers.

Definition 3.3 (Relatively Prime)**.** *Two integers a and b are said to be relatively prime, or coprime, if $GCD(a, b) = 1$.*

Let us now consider the origin of Equation 3.12. The author, again, is not aware of any derivation for this expression in the case of phase accumulators. We begin by introducing a few theorems that flow from our earlier definitions.

Lemma 3.1 (GCD Divisibility)**.** *If $GCD(a, b) = d$, then*

$$GCD\left(\frac{a}{d}, \frac{b}{d}\right) = 1 \tag{3.13}$$

Proof. Let $\text{GCD}(a, b) = d$. Assume that

$$\text{GCD}\left(\frac{a}{d}, \frac{b}{d}\right) = k, \; k > 1. \tag{3.14}$$

Then by Definition 3.2, $k \mid \frac{a}{d}$ and $k \mid \frac{b}{d}$. By Definition 1.1, there exists $c_0, c_1 \in \mathbb{Z}$ such that

$$c_0 k = \frac{a}{d} \Rightarrow c_0 k d = a \tag{3.15}$$

$$c_1 k = \frac{b}{d} \Rightarrow c_1 k d = b \tag{3.16}$$

Clearly then $kd \mid a$ and $kd \mid b$, but since $k > 1$, $kd > d$ making it a common divisor larger than d, which leads to a contradiction since d by choice is the greatest common divisor of a and b. Assuming $k < 1$ and not equal to zero also leads to a contradiction in a similar manner. Therefore $k = 1$. □

Now we show another helpful theorem [30] that will prove useful in the derivation of the phase accumulator least period.

Lemma 3.2 (Linear Modulo Normalization)**.** *If* $ac \equiv bc \,(mod\, m)$ *and* $GCD(c, m) = d$*, then* $a \equiv b \left(mod\, \frac{m}{d}\right)$.

Proof. Let $ac \equiv bc \,(\text{mod}\, m)$ and $\text{GCD}(c, m) = d$. From the definition of congruence given above, $m \mid (ac - bc) \Rightarrow m \mid c(a - b)$. Since d divides c and m (from the definition of GCD), $\frac{m}{d} \mid \frac{c}{d}(a - b)$. Now from Lemma 3.1, we know that $\text{GCD}\left(\frac{c}{d}, \frac{m}{d}\right) = 1$. Since $\frac{m}{d} \nmid \frac{c}{d}$, $\frac{m}{d} \mid (a - b)$. From Definition 3.1,

$$a \equiv b \left(\text{mod}\, \frac{m}{d}\right) \tag{3.17}$$

□

Here we note that nothing in the theorem or proof prevents $d = 1$, so it immediately follows that

$$ac \equiv bc \,(\text{mod}\, m) \Rightarrow a \equiv b \,(\text{mod}\, m) \tag{3.18}$$

if $\text{GCD}(c, m) = 1$.

As the DDFS is designed to generate spectrally pure signals, it is of critical importance to determine the periods of the sequences generated by the DCDO. Unwanted periodic behavior

generates deterministic spurs and will become a major topic in the discussions of Chapter 4. Now consider, yet again, the fundamental phase accumulator expression Equation 3.1. As an example of the case when $F = 1$, the the sequence generated is

$$P = [0, 1, 2, \ldots, N_P - 1, 0, 1, 2, \ldots] \tag{3.19}$$

Clearly then the length of the period for $F = 1$ is N_P. Now, in general, Equation 3.12 can be shown to be true for arbitrary F and P_0.

Theorem 3.3 (Phase Accumulator Periodicity)**.** *The least period of the sequence generated by applying the FCW F to a phase accumulator with N_P states without phase truncation is*

$$\Lambda_P \triangleq \frac{N_P}{GCD(F, N_P)} \tag{3.20}$$

Proof. Recall that the state of the phase accumulator at clock cycle n is

$$P[n] = \langle Fn + P_0 \rangle_{N_P} \tag{3.21}$$

If $\mathrm{GCD}(F, N_P) = 1$, then the period of the generated sequence is N_P and this can be shown by noting that for $n = 0$ that $P[0] = P_0$ and finding $k > 0$ such that $P[k] = P_0$. Equivalently, $P[k] = P[0] = \langle P_0 \rangle_{N_P}$, which means that

$$kF + P_0 \equiv P_0 \,(\mathrm{mod}\ N_P) \Rightarrow N_P \mid (kF + P_0 - P_0) \Rightarrow N_P \mid kF \tag{3.22}$$

But $N_P \nmid F$ since $\mathrm{GCD}(F, N_P) = 1$ and therefore $N_P \mid k$ and an integer d exists such that $dN_P = k$. The simplification was made using Lemma 3.4. Then we wish to find the smallest

positive integer d such $dN_P = k$ is true. Let $d = 1$, then $k = N_P$.

$$\begin{aligned} P[N_P] &= \langle N_P F + P_0 \rangle_{N_P} \\ &= \left\langle \langle N_P F \rangle_{N_P} + \langle P_0 \rangle_{N_P} \right\rangle_{N_P} \\ &= \langle P_0 \rangle_{N_P} \end{aligned} \tag{3.23}$$

so $d = 1$ satisfies the equality and the period of the sequence for $\text{GCD}(F, N_P) = 1$ is N_P. Now consider F such that $\text{GCD}(F, N_P) = d$. We are again looking for k such that

$$[kF + P_0] \equiv P_0 \,(\text{mod } N_P) \Rightarrow nF \equiv 0 \,(\text{mod } N_P) \tag{3.24}$$

as $P[0] = P_0$ still for this F. From Lemma 3.2 we know that

$$nF \equiv 0 \,(\text{mod } N_P) \Rightarrow n\frac{F}{d} \equiv 0 \left(\text{mod } \frac{N_P}{d}\right) \tag{3.25}$$

From Lemma 3.1, $\text{GCD}\left(\frac{F}{d}, \frac{N_P}{d}\right) = 1$ and from our previous analysis for the periodicity of the sequence for relatively prime F and N_P, it follows that the period is $\frac{N_P}{d}$. So we have shown the origin of Equation 3.12 and proved it to be true for all F. □

It is interesting to note that typically N_P is a power of 2, so $\text{GCD}(F, N_P)$ will be a power of 2 if F is not relatively prime to N_P. Thus looking at the position of first "1" in the bit string counting from the LSB of F gives the period of the sequence generated by the phase accumulator. Consider an 4-bit accumulator as a simple example. If $F = 6$, then the binary representation of $F = 0110$. Noting that the first 1 appears in the second bit position, meaning that the largest power of 2 divisor of F is 2^1 and the period of the phase accumulator is $2^4/2^1 = 2^3 = 8$.

3.3 Truncated Phase Sequences

Applying Lemma 3.2 to Theorem 3.2, it is possible to write the phase accumulator sequence Equation 3.4 in a "reduced" form. The reduced form is useful when deriving techniques for minimizing the number of required computations for a DFT of a sequence.

Lemma 3.3 (Alternative Phase Accumulator Expression)**.** *The phase accumulator Equation 3.4 with N_P states and FCW F can be written as:*

$$P[n] = d\left\langle \Gamma_P n \right\rangle_{\Lambda_P} \tag{3.26}$$

where $d = GCD\left(F, N_P\right)$, $\Gamma_P = F/d$ and Λ_P is the least period of $\left\langle Fn \right\rangle_{N_P}$ and can be calculated from Theorem 3.3.

Proof. Let $d = \mathrm{GCD}\left(F, N_P\right)$ and $\left\langle Fn \right\rangle_{N_P} = r_0$. Then by Definition 3.2, $d \mid F$ and $d \mid N_P$ and there exist integers $\Gamma_P = F/d$ and $\Lambda_P = N_P/d$. By the definition of the modulo operator, there exists $c_0 \in \mathbb{Z}$ such that

$$\left\langle Fn \right\rangle_{N_P} = r_0 \Rightarrow Fn - c_0 N_P = r_0 \tag{3.27}$$

$$\Rightarrow d\Gamma_P n - c_0 d\Lambda_P = r_0 \tag{3.28}$$

$$\Rightarrow \Gamma_P n - c_0 \Lambda_P = \frac{r_0}{d} \tag{3.29}$$

where $0 \leq r_0 < N_P$. Note that $0 \leq \frac{r_0}{d} < \frac{N_P}{d}$ must also be true and therefore $\left\langle \Gamma_P n \right\rangle_{\Lambda_P} = \frac{r_0}{d}$. Multiplying both sides by d yields $d\left\langle \Gamma_P n \right\rangle_{\Lambda_P} = r_0 = \left\langle Fn \right\rangle_{N_P}$. □

The next theorem provides a method for manipulating the sum of multiple integers modulo N. The technique is powerful at reducing the complexity of the summation in modulo arithmetic. As will be demonstrated soon, the truncation operation can be written concisely as the difference of two modulo sequences.

Lemma 3.4 (Sum of Two Integers Modulo N)**.**

$$\langle a+b\rangle_N = \langle \langle a\rangle_N + \langle b\rangle_N \rangle_N \tag{3.30}$$

Proof. From the definition of the modulo operator, the left-hand side of Equation 3.30 can be written as

$$r_0 = \langle a+b\rangle_N = (a+b) - c_0 N \tag{3.31}$$

where $0 \leq r_0 < N$ and $c_0 \in \mathbb{Z}$. The right-hand side of Equation 3.30 can be written as

$$\begin{aligned} r_a &= \langle a\rangle_N = a - c_1 N \\ r_b &= \langle b\rangle_N = b - c_2 N \\ r_1 &= (a - c_1 N) + (b - c_2 N) - c_3 N \\ &= (a+b) - (c_1 + c_2 + c_3)\, N \end{aligned} \tag{3.32}$$

where $0 \leq r_1 < N$ and $c_1, c_2, c_3 \in \mathbb{Z}$. From the division algorithm (Theorem 3.1), we know that for integer $(a+b)$ and N there exists a unique q and r such that

$$(a+b) = Nq + r, \quad 0 \leq r < N \tag{3.33}$$

As already stated, $0 \leq r_0 < N$ and $0 \leq r_1 < N$. Now from the previous equations, it is clear that

$$(a+b) = c_0 N + r_0 \tag{3.34}$$

$$(a+b) = (c_1 + c_2 + c_3)\, N + r_1 \tag{3.35}$$

By the definition of uniqueness through application of the division algorithm, $c_0 = c_1 + c_2 + c_3$ and $r_0 = r_1$. Therefore

$$\langle a + b \rangle_N = \langle \langle a \rangle_N + \langle b \rangle_N \rangle_N \tag{3.36}$$

and the proof is complete. □

The size of the SCMF is exponentially related, in the case of a ROM, to the number of bits of the phase accumulator used in addressing. The frequency resolution is dependent on the number of bits in the phase accumulator (Section 1.3.4). This creates a design trade-off decision point, where the area and speed of the ROM is juxtaposed with the frequency resolution of the DDFS. For this reason, the phase word from the phase accumulator is *truncated* so that a smaller address is required. The study of DDFS devices then requires the study of truncated phase sequences. Theorem 3.4 provides a mathematical expression for the truncated phase sequence. But first truncation should be clearly defined.

Definition 3.4 (Truncation)**.** *Truncation is defined as "ignoring" the least significant B_T digits of a number. The sequence generated by truncating the least B_T-digits of an N_P state accumulator driven by FCW F is given by*

$$P_{trunc}[n] = \langle Fn \rangle_{N_P} - \left\langle \langle Fn \rangle_{N_P} \right\rangle_{N_E} \tag{3.37}$$

where $N_E = r^{B_T}$ and r is the radix of the number system.

The truncated phase sequence is introduced as a definition in the general form above instead of a derived theorem to allow for truncation in non-radix 2 number systems. The purpose of truncation is to allow fewer digits to represent a number at the expense of some amount of error, which is denoted *truncation error*. Theorem 3.4 shows the reduced sequence generated through truncation in the phase accumulator. Intuitively for hardware engineers, one can view phase truncation of the accumulator as shifting the least B_T bits "right" (in the direction of the LSB) while simultaneously shifting zeros in at the MSB position.

Theorem 3.4 (Truncated Phase Sequence)**.** *N_E divides every element of the sequence generated by Equation 3.37, such that the truncated phase sequence can be written in a normalized form,*

$$P_T[n] = \frac{1}{N_E}\left[\langle Fn\rangle_{N_P} - \left\langle \langle Fn\rangle_{N_P}\right\rangle_{N_E}\right] \tag{3.38}$$

Proof. Consider the truncated sequence defined in Definition 3.4,

$$P_{\text{trunc}} = \langle Fn\rangle_{N_P} - \left\langle \langle Fn\rangle_{N_P}\right\rangle_{N_E} \tag{3.39}$$

From the definition of the modulo operator,

$$\langle Fn\rangle_{N_P} = Fn - c_0 N_P \tag{3.40}$$

$$\left\langle \langle Fn\rangle_{N_P}\right\rangle_{N_E} = \langle Fn\rangle_{N_P} - c_1 N_E \tag{3.41}$$

where $c_0, c_1 \in \mathbb{Z}$ for some fixed value of n (though such integer pairs exist for every value of n). Substituting Equations 3.40 and 3.41 into Equation 3.39,

$$(Fn - c_0 N_P) - [(Fn - c_0 N_P) - c_1 N_E] = c_1 N_E \tag{3.42}$$

Clearly $N_E | c_1 N_E$ and Equation 3.38 holds. □

Equation 3.43 shows the behavior in binary number systems, which better reflects hardware implementations. Setting $r = 2$, the number of error states for B_T digits (bits) of truncation is $N_E = 2^{B_T}$. Applying Theorem 3.4 with these values,

$$P_T[n] = \frac{1}{2^{B_T}}\left[\langle Fn\rangle_{N_P} - \left\langle \langle Fn\rangle_{N_P}\right\rangle_{2^{B_T}}\right] \tag{3.43}$$

A further simplification can be made when the number of accumulator states is a power of two, which is the case when the carry-out of the accumulator is simply ignored on an overflow. This type of phase accumulator is the focus of the analyses of Nicholas and Torosyan and

their results are in fact dependent on the special case. Assuming that $N_P = 2^{B_P}$ where B_P is the number of bits in the accumulator,

$$P_T[n] = \frac{1}{2^{B_T}} \left[\langle Fn \rangle_{2^{B_P}} - \langle Fn \rangle_{2^{B_T}} \right] \tag{3.44}$$

The reduction is made through application of Lemma 3.5 on the $2^{B_T} \mid 2^{B_P}$ case. The theorem immediately follows and will be used to derive the least period of the truncation error.

Lemma 3.5 (Least Period of the Modulo of a Modulo Sequence)**.** *The sequence*

$$f[n] = \langle \langle Fn \rangle_N \rangle_M \tag{3.45}$$

has least period Λ_N *if* $M \nmid N$ *or period* Λ_M *if* $M \mid N$*.*

Proof. First consider $M \mid N$, then there exists an integer d such that $dM = N$ (from Definition 1.1). From the definition of the modulo operator,

$$\langle Fn \rangle_N \Rightarrow Fn - c_0 N = r_0 \tag{3.46}$$

where $c_0 \in \mathbb{Z}$ and $0 \leq r_0 < 0$. Applying the second modulo operation,

$$\langle Fn - c_0 N \rangle_M \Rightarrow Fn - c_0 N - c_1 M = r_1 \tag{3.47}$$

where $c_1 \in \mathbb{Z}$ and $0 \leq r_1 < M$. Rearranging the previous equation and substituting dM for N,

$$Fn = (c_0 d + c_1) M + r_1 \tag{3.48}$$

But $0 \leq r_1 < M$, so from the definition of the modulo operation

$$Fn = (c_0 d + c_1) M + r_1 \Rightarrow \langle Fn \rangle_M \tag{3.49}$$

The period of $\langle Fn \rangle_M$ can be calculated from Theorem 3.2,

$$\Lambda_M = \frac{M}{\text{GCD}\,(F, M)} \tag{3.50}$$

Thus we have shown the case for $M \mid N$. Now consider the case where $M \nmid N$. From Theorem 3.2, we know the period of the sequence $\langle Fn \rangle_N$ is Λ_N, so the sequence $\langle \langle Fn \rangle_N \rangle_M$ is also periodic with Λ_N. □

The previous result is important in the analysis of the least period of truncated phase sequences. Lemma 3.5 can directly be used to calculate the least period of the truncation error sequence, where truncation is defined in Definition 3.4.

Truncation generally happens on a bit boundary as the operation is free in hardware (i.e. the bits are simply ignored). Therefore the general result of Theorem 3.5 can be made more specific to the typical DCDO accumulator use case.

Theorem 3.5 (Periodicity of Phase Truncation Error Sequence)**.** *The truncated error sequence of an N_P state phase accumulator driven by FCW F with N_E states in the truncated sequence has least periodicity*

$$\Lambda_E = \begin{cases} \Lambda_P & \text{if } N_E \nmid N_P \\ \frac{N_E}{GCD(F, N_E)} & \text{if } N_E \mid N_P \end{cases} \tag{3.51}$$

Proof. The error sequence of the phase accumulator is (Theorem 3.4)

$$P_E[n] = \left\langle \langle Fn \rangle_{N_P} \right\rangle_{N_E} \tag{3.52}$$

Let $N_E \nmid N_P$. Then applying Lemma 3.5, we know that $P_E[n]$ has least period Λ_P. Now let $N_E \mid N_P$, then again applying Lemma 3.5 $P_E[n]$ has least period

$$\frac{N_E}{\text{GCD}\,(F, N_E)} \tag{3.53}$$

and the proof is complete.

□

As will be discussed in Chapter 4, in most real hardware implementations $N_E \mid N_P$. Thus the variable $N_Q = \frac{N_P}{N_E}$ is introduced and represents the number of unique values in the truncated phase word. This value becomes the number of entries (address domain) of the SCMF.

Now the least period of the difference between two modulo sequences will be derived. Since the truncated error sequence was shown to be of this form in Theorem 3.4, the following derivation can be directly used to compute the periodicity of the sequence.

Theorem 3.6 (Periodicity of the Difference of Two Modulo Sequences)**.** *The least period of the sequence generated by*

$$f[n] = \langle Fn \rangle_N - \langle Fn \rangle_M \tag{3.54}$$

is the least common multiple of the least periods of the two sequences

$$f_0[n] = \langle Fn \rangle_N \tag{3.55}$$

$$f_1[n] = \langle Fn \rangle_M \tag{3.56}$$

Proof. Let $L = \mathrm{LCM}\,(\Lambda_N, \Lambda_M)$, where

$$\Lambda_N = \frac{N}{\mathrm{GCD}\,(F, N)} \tag{3.57}$$

$$\Lambda_M = \frac{M}{\mathrm{GCD}\,(F, M)} \tag{3.58}$$

are the least periods of $f_0[n]$ and $f_1[n]$ from Theorem 3.3. From Lemma 3.3, the sequences can be written as

$$f_0[n] = \mathrm{GCD}\,(F, N)\,\langle \Gamma_N n \rangle_{\Lambda_N} \tag{3.59}$$

$$f_1[n] = \mathrm{GCD}\,(F, M)\,\langle \Gamma_M n \rangle_{\Lambda_M} \tag{3.60}$$

where $\Gamma_N = F/\mathrm{GCD}\,(F, N)$ and $\Gamma_M = F/\mathrm{GCD}\,(F, M)$. First we check that L is a period for $f[n]$.

$$f[n + L] = \mathrm{GCD}\,(F, N)\,\langle \Gamma_N n + L \rangle_{\Lambda_N} - \mathrm{GCD}\,(F, M)\,\langle \Gamma_M + L \rangle_{\Lambda_M} \tag{3.61}$$

Note that the multiplication of the GCD terms does not influence periodicity of the sequence. From Definition 1.2, $\Lambda_N \mid L$ and $\Lambda_M \mid L$. Therefore $\langle L \rangle_{\Lambda_N} = 0$ and $\langle L \rangle_{\Lambda_M} = 0$. Applying Lemma 3.4 to the previous equation

$$f[n + L] = \mathrm{GCD}\,(F, N)\,\langle \Gamma_N n \rangle_{\Lambda_N} - \mathrm{GCD}\,(F, M)\,\langle \Gamma_M n \rangle_{\Lambda_M} = f[n] \tag{3.62}$$

Now we must show that L is the least period of $f[n]$. Assume that there is a positive integer $K < L$ such that

$$f[n + K] = f[n] \tag{3.63}$$

Then

$$f[n + K] = f[n] \tag{3.64}$$

$$\langle Fn + K \rangle_N - \langle Fn + K \rangle_M = \langle Fn \rangle_N - \langle Fn \rangle_M \tag{3.65}$$

$$\langle \langle Fn \rangle_N + \langle K \rangle_N \rangle_N - \langle \langle Fn \rangle_M + \langle K \rangle_M \rangle_M = \langle Fn \rangle_N - \langle Fn \rangle_M \tag{3.66}$$

which implies that

$$\langle K \rangle_N = 0 \tag{3.67}$$

$$\langle K \rangle_M = 0 \tag{3.68}$$

From the definition of the modulo operation, $K = c_0 N = c_1 M$ for some $c_0, c_1 \in \mathbb{P}$. So K must be a multiple of both N and M that is less than L. But this is a contradiction, since L is the least common multiple of Λ_N and Λ_M. □

Theorem 3.7 (Truncated Phase Sequence Period)**.** *The length of the least period of the truncated phase accumulator sequence (Equation 3.38) with N_P phase states and N_E error states is*

$$\Lambda_P = \frac{N_P}{GCD\,(F, N_P)} \tag{3.69}$$

Proof. The truncated phase sequence is given by Equation 3.38,

$$P_T[n] = \frac{1}{N_E}\left[\langle Fn \rangle_{N_P} - \left\langle \langle Fn \rangle_{N_P} \right\rangle_{N_E}\right] \tag{3.70}$$

From Theorem 3.6, the period of $P_T[n]$ is equal to the least common multiple of least period of $\langle Fn \rangle_{N_P}$ and $\left\langle \langle Fn \rangle_{N_P} \right\rangle_{N_E}$. From Theorem 3.3, the least period of $\langle Fn \rangle_{N_P}$ is

$$\Lambda_1 = \Lambda_P = \frac{N_P}{\mathrm{GCD}\,(F, N_P)} \tag{3.71}$$

From Theorem 3.5, the least period of

$$\Lambda_2 = \begin{cases} \Lambda_P \text{ if } \Lambda_E \nmid \Lambda_P \\ \Lambda_E, \text{ if } \Lambda_E \mid \Lambda_P \end{cases} \tag{3.72}$$

Now consider the least common multiple of Λ_1 and Λ_2 when $\Lambda_E \nmid \Lambda_P$. Then $\Lambda_2 = \Lambda_P$ and $\mathrm{LCM}(\Lambda_1, \Lambda_2) = \mathrm{LCM}(\Lambda_P, \Lambda_P) = \Lambda_P$.

Now consider the other case where $\Lambda_E \mid \Lambda_P$ which from Definition 1.1 there exists $c_0 \in \mathbb{Z}$ such that $c_0\Lambda_E = \Lambda_P$. Then $\text{LCM}(\Lambda_1, \Lambda_2) = \text{LCM}(\Lambda_P, \Lambda_E) = \text{LCM}(c_0\Lambda_E, \Lambda_E) = c_0\Lambda_E = \Lambda_P$. Therefore Λ_P is the least period of $P_T[n]$. □

The periodicities for arbitrary phase accumulator sequence, truncated phase sequence, and truncated error sequence have been derived along with the exact mathematical formulations of the sequences themselves. The analysis will prove useful in re-deriving Nicholas's [23], Torosyan's [29] and Jenq's [24] phase truncation papers of Chapter 2 as well as the fully general theorem presented in this work in Chapter 4.

3.4 Relationships Between Sequences

From Section 3.2, it is clear that different FCWs can have the same least period. Since there are only a finite number of states that can be taken in modulo arithmetic, it would intuitively seem that these sequences of the same period would be related in some manner. This section aims to develop how these sequences are related. Before beginning, however, it is helpful to show when a multiplicative inverse exists in a modulo number system.

Lemma 3.6 (GCD and Linear Diophantine Equations)**.** *If* $GCD\,(a, b) = d$ *and* $a \in \mathbb{P}_0$ *and* $b \in \mathbb{P}$*, then there exist* $x, y \in \mathbb{Z}$ *such that* $ax + by = d$*. Equations of the form* $ax + by = c$*, where only integer solutions are allowed are called linear Diophantine equations.*

Proof. The popular approach for proving this lemma is applying Euclid's GCD algorithm in reverse until an x and y are found such that $ax + by = \text{GCD}\,(a, b)$. Here a more compact approach will be used. Let $a \in \mathbb{P}_0$ and $b \in \mathbb{P}$. Let $S = \{ax + by : x, y \in \mathbb{Z} \text{ and } ax + by \geq 0\}$. That S is non-empty can be shown by choosing x and y equal to zero. S is well-ordered by Principle 1.2. Select the smallest element of S, say d, and write it as $ax_0 + by_0 = d$. Now if $d = \text{GCD}\,(a, b)$, then the proof is complete.

Assume that $d \nmid a$, then from the Theorem 3.1, $a = dq + r$, $0 < r < d$ (if $r = 0$ then d would divide a) and therefore $dq = a - r$. Multiplying $ax_0 + by_0 = d$ by q and substituting

for dq for $a - r$ yields the following

$$\begin{aligned} q(ax_0 + by_0) &= dq \\ q(ax_0 + by_0) &= a - r \\ a - qax_0 - by_0 &= r \\ a(1 - qx_0) - by_0 &= r \end{aligned}$$

Letting $x_1 = (1 - qx_0)$ and $y_1 = -y_0$, then $ax_1 + by_1 = r$ and $r \in S$. But this leads to a contradiction because $r < d$ but d is the least element of S. So $d \mid a$. The same technique can then be used to show that $d \mid b$.

Now to show that d is the greatest common divisor of a and b. By Definition 3.2, any divisor of a and b must also divide d for this to be true. Let c be a common divisor of a and b.

$$\begin{aligned} d &= ax + by \\ &= c\left(\frac{a}{c}\right)x + c\left(\frac{b}{c}\right)y \\ &= c\left(\frac{a}{c}x + \frac{b}{c}y\right) \end{aligned}$$

Clearly then $c \mid d$ and $\mathrm{GCD}(a, b) = d$. □

This provides a powerful technique for working with equations involving the greatest common divisor of two numbers.

Theorem 3.8 (Multiplicative Inverse in Modulo Arithmetic)**.** *A unique multiplicative inverse $F^{-1} \in \mathbb{Z}_{N_P}$ exists such that*

$$\left\langle F^{-1}F \right\rangle_{N_P} = \langle 1 \rangle_{N_P} \tag{3.73}$$

if and only if $GCD\,(F, N_P) = 1$,

Proof. Let $\text{GCD}\,(F, N_P) = 1$. From Lemma 3.6 there exist $x, y \in \mathbb{Z}$ such that $Fx + N_P y = 1$.

$$Fx + N_P y = 1 \tag{3.74}$$

$$Fx - 1 = -N_P y \tag{3.75}$$

Then $N_P \mid (Fx - 1)$ and from Definition 3.1, $Fx \equiv 1 \pmod{N_P}$ and x is the multiplicative inverse of F modulo N_P. Now to show that it is unique, assume that $x_0, x_1 \in \mathbb{Z}_{N_P}$ are distinct solutions to the linear congruence relation $Fx \equiv 1 \pmod{N_P}$. Then

$$\begin{aligned} Fx_0 &\equiv 1 \pmod{N_P} \\ Fx_1 &\equiv 1 \pmod{N_P} \end{aligned} \tag{3.76}$$

Through the transitive property, $Fx_0 \equiv Fx_1 \pmod{N_P}$. Then $N_P \mid (Fx_0 - Fx_1)$ and through the distributive property $N_P \mid F(x_0 - x_1)$. Since $\text{GCD}(F, N_P) = 1$, $N_P \nmid F$ and therefore $N_P \mid (x_0 - x_1)$. Again from Definition 3.1, $x_0 \equiv x_1 \pmod{N_P}$. But both x_0 and x_1 are less than N_P by definition and $x_0 = x_1$. Therefore the multiplicative inverse is unique by contradiction. □

The relationship between the sequences of FCWs can now be derived as all the tools necessary for the derivation have been developed.

Theorem 3.9 (FCW Time Sequence Permutation Relationship)**.** *The sequences generated by two different FCWs F_0 and F_1 driving an accumulator with N_P states are permutations of each other if GCD(F_0, N_P) = GCD(F_1, N_P).*

Proof. Let $\text{GCD}(F_0, N_P) = \text{GCD}(F_1, N_P) = d$. We wish to show that the following two sequences

$$f_0[n] = \langle F_0 n \rangle_{N_P} \tag{3.77}$$

$$f_1[n] = \langle F_1 n \rangle_{N_P} \tag{3.78}$$

are permutations of each other, and more particularly that,

$$f_0[k_1 n] = f_1[n] \quad \text{and} \quad f_1[k_0 n] = f_0[n] \tag{3.79}$$

for some $k_0, k_1 \in \mathbb{Z}_{N_P}$. First consider the case where $d = 1$ (i.e. F_0 and F_1 are relatively prime to N_P). Then from Theorem 3.8, there exist multiplicative inverses for F_0 and F_1 such that $F_0 F_0^{-1} = 1$ and $F_1 F_1^{-1} = 1$. Then

$$f_0[F_0^{-1} F_1 n] = \left\langle F_0 F_0^{-1} F_1 n \right\rangle_{N_P} = \langle F_1 n \rangle_{N_P} = f_1[n] \tag{3.80}$$

So $k_1 = F_0^{-1} F_1$ and $k_0 = F_1^{-1} F_0$. Now consider the case where $d \neq 1$. Then from Lemma 3.2, we get

$$\frac{f_0[n]}{d} = \left[\left(\frac{F_0}{d} \right) n \right] \left(\text{mod } \left(\frac{N_P}{d} \right) \right) \tag{3.81}$$

$$\frac{f_1[n]}{d} = \left[\left(\frac{F_1}{d} \right) n \right] \left(\text{mod } \left(\frac{N_P}{d} \right) \right) \tag{3.82}$$

But now $\text{GCD}\left(\frac{F_0}{d}, \frac{N_P}{d}\right) = \text{GCD}\left(\frac{F_1}{d}, \frac{N_P}{d}\right) = 1$ from Lemma 3.1 and multiplicative inverses exist for both $\frac{F_0}{d}$ and $\frac{F_1}{d}$ from Theorem 3.8. Then the logic for when $d = 1$ readily applies and $k_1 = \left(\frac{F_0}{d}\right)^{-1} \left(\frac{F_1}{d}\right)$ and $k_0 = \left(\frac{F_1}{d}\right)^{-1} \left(\frac{F_1}{d}\right)$ are the multipliers that generate the permutations.

□

Theorem 3.9 forms a cornerstone in the analysis of Chapter 4 and is the most important observation of this chapter. A similar observation is made by Torosyan in [29] through citing a number theory text. Here the relevant analysis from several texts have been collated to arrive at the same observation.

3.5 Comments on Mathematical Structure

Lastly, we note that the equivalent mathematical structure for the phase accumulator is a *commutative ring with identity*. The following paragraphs define and describe the properties of rings. First, the definition of a group is presented [15]

Definition 3.5 (Groups)**.** *The set P with a binary operation $+$ is called a group if*

1. *$a + (b + c) = (a + b) + c$ for $a, b, c \in P$, i.e. associativity holds.*
2. *There exists an element $e \in P$ such that $a + e = e + a = a$ for all $a \in P$. The element e is called the identity for $+$.*
3. *For every $a \in P$, there exists an element $b \in P$ such that $a + b = b + a = e$. The element b is called the inverse of a with respect to $+$.*

This can be further refined to a *Abelian* group in the phase accumulators case, as the *commutative* property holds ($a + b = b + a$, whenever $a, b \in P$). A ring is a set R with two binary operations $+$ and $\cdot$ on R such that

- The $+$ operation forms a commutative group on R, and
- The $\cdot$ operation forms a semigroup.
- For $a, b, c \in R$, then $a \cdot (b + c) = a \cdot b + a \cdot c$ and $(a + b) \cdot c = a \cdots c + b \cdot c$, i.e. the $\cdot$ operation is *distributive* over $+$.

Note that the definition of the phase accumulator at state n makes use of multiplication (or "$\cdot$" in the definition of the the commutative ring). Notice that a multiplicative inverse need not exist for an element $\alpha \in P$.

The author believes that using more advanced properties of Abelian rings could be used to arrive at a general solution more quickly than the detailed analysis used in this document. However, the solution derived in this work is exact and with merit. It also makes use of more

elementary properties of number theory such that an undergraduate student in Computer Engineering (or an engineering discipline that provides some discrete mathematics) could fully follow.

Chapter 4

Spectrum of Truncated Phase Sequences

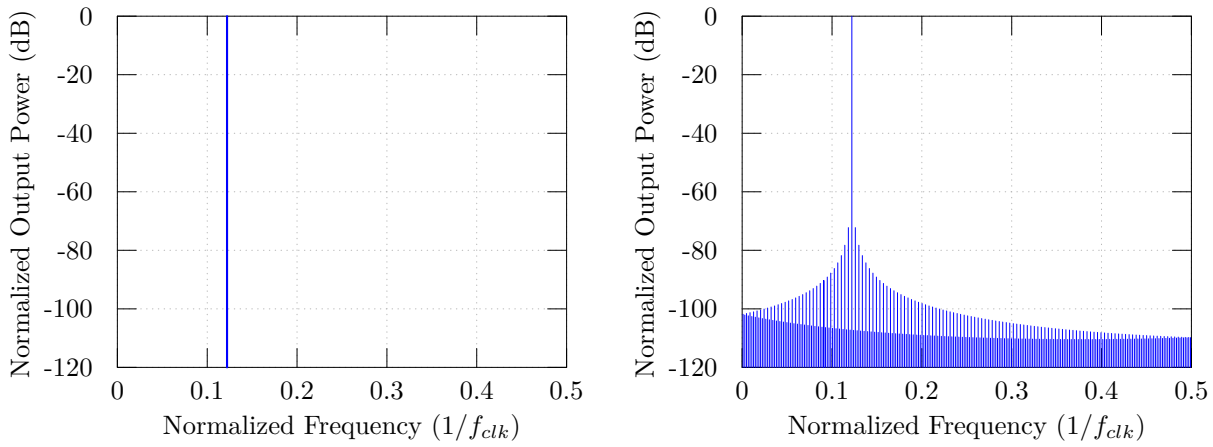
The spectrum of truncated phase sequences is now investigated. To visualize the peculiar behavior of the spurs generated by phase truncation, consider Figure 4.1a and Figure 4.1b. Figure 4.1a shows the output spectrum from a single period of a $B_P = 20$ bit phase accumulator, truncated to $B_Q = 12$ bits and driven by FCW $F = 2^8 \cdot 500$. Figure 4.1b shows the output spectrum when $F = (2^8 \cdot 500) + 1$. Though it has no effect on the analysis, assume 1 GHz clock for the above example. This means that two tones approximately $1\,\text{GHz}/2^{20} \approx 1\,\text{kHz}$ apart produce radically different output spectrums. Figure 4.1a shows only one spectral line at the intended synthesized frequency, while Figure 4.1b shows several hundred spurious tones in addition to the main tone. The truncation error sequence is (Theorem 3.4)

$$P_E[n] = \Big\langle \langle Fn \rangle_{N_P} \Big\rangle_{N_E} \tag{4.1}$$

where $N_P = 2^{B_P}$ and $B_E = 2^{B_P - B_Q} = 2^8$. From Lemma 3.5, the period of the error sequence is 2^8. In this chapter, it will be shown how this information can be used to predict the exact location, magnitude and phase of each individual spur that result from phase truncation.

4.1 Intuitive Understanding

Early analysis of the magnitude of the spurs generated by phase truncation uses trigonometric approximations. The simplest argument is that the error from phase truncation ($\Delta\theta$)



(a) $B_P = 20$, $F = 2^8 \cdot 500$, $B_Q = 12$ (b) $B_P = 20$, $F = 2^8 \cdot 500 + 1$, $B_Q = 12$

Figure 4.1: Spectrums from Two Adjacent FCWs

is relatively small with respect to the phase variable (θ). Equation 4.2 shows the approximation.

$$\sin(\theta + \Delta\theta) = \sin(\theta)\cos(\Delta\theta) + \cos(\theta)\sin(\Delta\theta) \tag{4.2}$$

$$\approx \sin(\theta) + \cos(\theta)\,\Delta\theta, \quad \text{whenever } \Delta\theta \ll 1 \tag{4.3}$$

The technique applies a product-to-sum trigonometric identity followed by a small angle approximation. The quality of the small angle approximation can be evaluated through applying a Taylor series expansion on sine.

Definition 4.1 (Taylor Series)**.** *Let $f(x)$ be an infinitely differentiable function. The Taylor series of $f(x)$ taken about a value* a *is defined by Equation 4.4.*

$$f(x) \approx \sum_{n=0}^{n=\infty} \left[\frac{d^n}{dx^n}\left(f(x)\right)\right] \frac{(x-a)}{n!} \tag{4.4}$$

The Maclaurin series expansion for sine, which is a special case of the Taylor series taken about $a = 0$, yields:

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \cdots \tag{4.5}$$

Note that for $x \ll 1$, the third order and fifth order terms become negligible. Equation 4.6 shows the small angle approximation for sine.

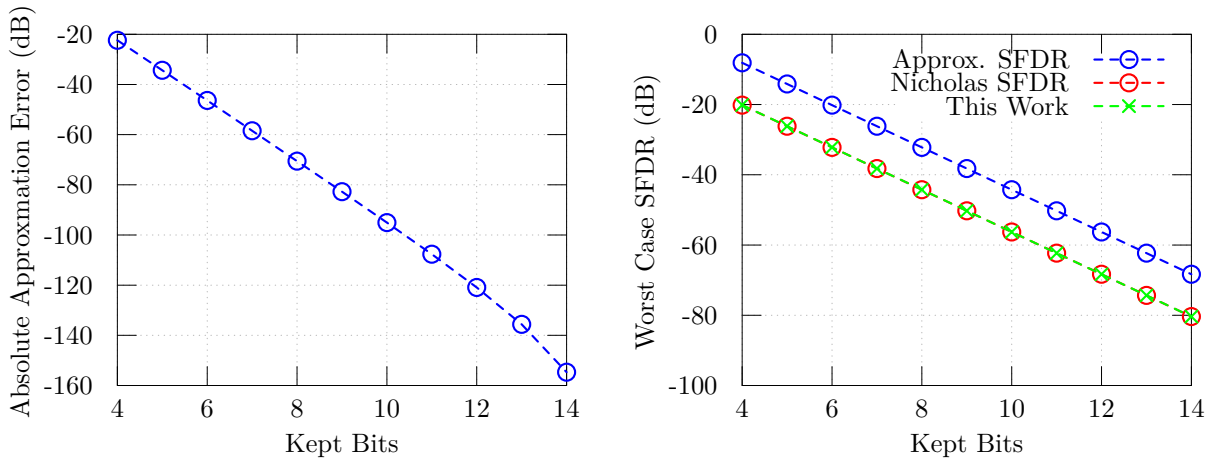
$$\sin(x) \approx x, \quad x \ll 1 \tag{4.6}$$

Ignoring the shape of the error $\Delta\theta$, i.e. assuming it constant, the approximation can be used to roughly approximate an upper bound on the largest spur from the phase truncation process. This is because keeping $\Delta\theta$ constant at its maximum value concentrates all the spurious energy into a single tone. In reality, $\Delta\theta$ varies with time and thus the cosine term is amplitude modulated by a "sawtooth wave-like" error function. This spreads the error energy into multiple frequency bins, reducing the maximum power at the spurious frequency.

Consider a phase accumulator with B_P-bits in which the bottom B_T bits are truncated, leaving a kept phase word of Q of B_Q-bits. Remember that the phase accumulator values are mapped uniformly from $[0, 2\pi)$ in $N_P = 2^{B_P}$ steps.

$$\max\{\Delta\theta\} = 2\pi\frac{N_{B_T}}{N_P} = 2\pi\frac{2^{B_T}}{2^{B_P}} = \frac{2\pi}{2^{B_P - B_T}} = \frac{2\pi}{2^{B_Q}} \tag{4.7}$$

It is clear from the previous equation that the phase error decreases exponentially with B_Q (halved for each kept bit added). Note also that the magnitude of the error depends on the relationship between number of bits truncated to the number of bits kept and not the number truncated bits alone. The loose upper bound for the SFDR in decibels from phase



(a) Small Angle Estimate

(b) Worst Case SFDR due to Phase Truncation

Figure 4.2: Simple Estimates for Worst Case SFDR due to Phase Truncation

truncation using this approximation is

$$20\log_{10}\left(\frac{2\pi}{2^{B_Q}}\right) = 20\log_{10}(2\pi) - 20B_Q\log_{10}(2) \approx 16 - 6.02B_Q \tag{4.8}$$

However, an approximation has been made in the derivation itself, so the "upper bound" is no upper bound in any formal sense of the phrase (or at least it has not been proven to be so yet in this work). The error in the trigonometric approximation of Equation 4.2 is shown in Figure 4.2a. Clearly for 10 bits or more, the approximation of Equation 4.2 to sine is greater than 90 dB, which will be much larger than the SNR and SFDR of the systems discussed in this work.

Figure 4.2b shows the actual worst case spur against the approximation using Nicholas's technique. Qualitatively it is clear that the estimate is a gross overestimate, approximately 10 dB, to the actual value of the worst case SFDR. But this is expected, as the phase error is a sawtooth like function against time and thus the spurious energy is spread across many bins.

There are several problems with relying on this analysis when designing a DDFS system.

1. The analysis provides no insight into the number of spurs generated by phase truncation.

2. The analysis provides no insight into the *location* of the phase truncation spurs.

3. The analysis provides no way to distinguish phase truncation spurs from quantized amplitude spurs.

4.2 Characteristics of Truncated Phase Sequences

Since the delta phase cycle is periodic with Λ_E, the difference between $P_T[n]$ and $P_T[n+\Lambda_E]$ should be the equal for all $n \in \mathbb{Z}$ and have the value F modulo N. An example demonstrating this can be found in Section 2.3.1.

Theorem 4.1 (Delta Phase Steps)**.** *The truncated phase difference separated by Λ_E steps is equal to the reduced FCW for F*

$$\langle P_T[n] - P_T[n+\Lambda_E] \rangle_{N_P} = \left\langle \frac{F}{GCD\,(F, N_E)} \right\rangle_{N_P} \tag{4.9}$$

if $N_E \mid N_P$ or is zero otherwise. N_P is the number of states in the phase accumulator and N_E is the number of states in the truncated error sequence.

Proof. Recall from Equation 3.38,

$$P_T[n] = \frac{1}{N_E}\left[\langle Fn \rangle_{N_P} - \left\langle \langle Fn \rangle_{N_P} \right\rangle_{N_E}\right] \tag{4.10}$$

$$P_T[n+\Lambda_E] = \frac{1}{N_E}\left[\langle Fn + F\Lambda_E \rangle_{N_P} - \left\langle \langle Fn + F\Lambda_E \rangle_{N_P} \right\rangle_{N_E}\right] \tag{4.11}$$

First consider the common case where $N_E \mid N_P$. Then the previous equations reduce, through application of Lemma 3.5, to

$$P_T[n] = \frac{1}{N_E}\left[\langle Fn\rangle_{N_P} - \langle Fn\rangle_{N_E}\right] \tag{4.12}$$

$$P_T[n+\Lambda_E] = \frac{1}{N_E}\left[\langle Fn + F\Lambda_E\rangle_{N_P} - \langle Fn + F\Lambda_E\rangle_{N_E}\right] \tag{4.13}$$

Next consider $\langle F\Lambda_E\rangle_{N_E}$. If $N_E \mid F\Lambda_E$ then $\langle F\Lambda_E\rangle_{N_E} = 0$ through the definition of the modulo operator. Let $d = \mathrm{GCD}\,(F, N_E)$. Then $d \mid F$ and $d \mid N_E$ by Definition 3.2 and there exist $c_0, c_1 \in \mathbb{Z}$ such that $c_0 d = F$ and $c_1 d = N_E$. $\Lambda_E = \frac{N_E}{d}$ from Equation 3.12. Then $F\Lambda_E = c_0 d\left(\frac{N_E}{d}\right) = c_0 N_E$. Clearly then $N_E \mid F\Lambda_E$ by Definition 1.1. Now the result follows through application of Lemma 3.4 that $\langle Fn + F\Lambda_E\rangle_{N_E} = \langle Fn\rangle_{N_E}$.

$$P_T[n] - P_T[n+\Lambda_E] = \frac{1}{N_E}\left[\left(\langle Fn\rangle_{N_P} - \langle Fn\rangle_{N_E}\right) - \left(\langle Fn + F\Lambda_E\rangle_{N_P} - \langle Fn\rangle_{N_E}\right)\right] \tag{4.14}$$

$$= \frac{1}{N_E}\left[\langle Fn\rangle_{N_P} - \langle Fn + F\Lambda_E\rangle_{N_P}\right] \tag{4.15}$$

Applying the modulo operator to both sides of the last equation yields the final result

$$\begin{aligned}\langle P_T[n] - P_T[n+\Lambda_E]\rangle_{N_P} &= \left\langle \frac{1}{N_E}\left[\langle Fn\rangle_{N_P} - \langle Fn + F\Lambda_E\rangle_{N_P}\right]\right\rangle_{N_P} \\ &= \left\langle \frac{1}{N_E}\langle Fn - Fn + F\Lambda_E\rangle_{N_P}\right\rangle_{N_P} \\ &= \left\langle \frac{1}{N_E}\langle F\Lambda_E\rangle_{N_P}\right\rangle_{N_P}\end{aligned} \tag{4.16}$$

Using the knowledge that $N_E \mid F\Lambda_E$ and substituting for Λ_E,

$$\begin{aligned}\langle P_T[n] - P_T[n+\Lambda_E]\rangle_{N_P} &= \left\langle \frac{F\Lambda_E}{N_E}\right\rangle_{N_P} \\ &= \left\langle \frac{F}{\mathrm{GCD}\,(F, N_E)}\right\rangle_{N_P}\end{aligned} \tag{4.17}$$

Now consider the other case where $N_E \nmid N_P$, then $\Lambda_E = N_P$ from Theorem 3.5. But $P_T[n]$ is periodic with N_P as well from Theorem 3.6. So $P_T[n] = P_T[n + \Lambda_E]$ and $P_T[n] - P_T[n + \Lambda_E] = 0$. □

The implication is that if $N_E \mid N_P$, then the sequence P_T can be divided into Λ_E sub-sequences that sum together to provide the original truncated phase sequence. First, it is important to show that any finite length sequence can be split into a sum of different sub-sequences by utilizing the Kronecker delta function (Definition 4.2).

Definition 4.2 (Kronecker Delta Function)**.** *The Kronecker delta is defined as a function on* $\mathbb{Z}$ *such that*

$$\delta[n] = \begin{cases} 0, & n \neq 0 \\ 1, & n = 0 \end{cases} \tag{4.18}$$

Theorem 4.2 (Sub-Sequences of a Finite Sequence)**.** *Any finite sequence* $f[n]$ *with length* Λ *such that* Λ *is a composite positive integer (i.e.* $\Lambda = aN$*) can be decomposed into the sum of* a *sub-sequences.*

$$f[n] = \sum_{q=0}^{a-1} \sum_{r=0}^{N-1} \delta\left[n - (Nq + r)\right] f[Nq + r], \quad 0 \leq n < \Lambda \tag{4.19}$$

where $\delta[n]$ *is the Kronecker Delta Function (Definition 4.2).*

Proof. Let $f[n]$ is a finite sequence of length Λ where $\Lambda = aN$ is a composite positive integer. It is clear that

$$f[n] = \sum_{m=0}^{\Lambda-1} \delta[n - m] f[m] \tag{4.20}$$

where $\delta[n]$ is the Kronecker delta function by the definition of the function. This operation is sometimes called substitution and proves helpful in analysis. From the division algorithm, for a given m and divisor N,

$$m = Nq + r \tag{4.21}$$

where $0 \leq r < N$ and q and r are unique. This information can be used to compose an equivalent double summation by substituting $Nq + r$ for m,

$$f[n] = \sum_{q=0}^{a-1} \sum_{r=0}^{N-1} \delta\left[n - (Nq + r)\right] f[Nq + r] \tag{4.22}$$

since $Nq + r \in \{0, 1, \cdots, \Lambda - 1\}$. □

While not dealing with finite sequences explicitly in the analysis thus far, the behavior of a periodic sequence can be understood by viewing a single period of the sequence. Under certain cases where the period of the error sequence divides the full truncated phase sequence period, Theorem 4.2 can be applied to develop an efficient algorithm for characterizing a DCDO. Let $\Lambda_E \mid \Lambda_P$ for a DCDO implementation, then the truncated phase sequence P_T can be reconstructed from the subsequences using Theorem 4.2.

$$P_T[n] = \sum_{j=0}^{(\Lambda_P/\Lambda_E - 1)} \sum_{m=0}^{\Lambda_E - 1} \delta[n - (\Lambda_E j + m)] P_T[n] \tag{4.23}$$

$$= \sum_{j=0}^{(\Lambda_P/\Lambda_E - 1)} \delta[n - \Lambda_E j] P_T[n] + \cdots + \sum_{j=0}^{(\Lambda_P/\Lambda_E - 1)} \delta[n - (\Lambda_E j + \Lambda_E - 1)] P_T[n] \tag{4.24}$$

Before analyzing the spectrum of such sequences, the interchangeability of the summations of finite sequences is considered.

Theorem 4.3 (Interchanging Summations for Finite Sequences)**.** *Two finite summations can be interchanged for arbitrary sequence $f[n]$.*

$$\sum_{n=0}^{N} \sum_{m=0}^{M} f[n, m] = \sum_{m=0}^{M} \sum_{n=0}^{N} f[n, m] \tag{4.25}$$

Proof. Since addition is commutative, $f[i,j]$ can be arranged into any order and return the same sum

$$\sum_{n=0}^{N}\sum_{m=0}^{M} f[n,m] = f[0,0] + f[0,1] + \cdots + f[0,M] + f[1,0] + \cdots + f[N,M] \tag{4.26}$$

$$= f[0,0] + f[1,0] + \cdots + f[N,0] + f[0,1] + \cdots + f[N,M] \tag{4.27}$$

$$= \sum_{m=0}^{M}\sum_{n=0}^{N} f[n,m] \tag{4.28}$$

□

The results seems self-evident, but it is important to note that the order of infinite summations can not be arbitrarily changed in every case. Thus pointing out that it can be done so with finite sequences prevents any gaping holes in the analysis that follows. The next step is to find the relationship between truncated phase output codes of the same value. To begin, the case of a FCW of one is analyzed.

Theorem 4.4 (Adjacent Truncated Phase Elements)**.** *For $F = 1$,*

$$P_T[\Lambda_E n] = P_T[\Lambda_E n + m] \tag{4.29}$$

for all $n \in \mathbb{Z}$ and $0 \leq m < \Lambda_E$ if $N_E \mid N_P$.

Proof. Plugging in the left hand side of Equation 4.29 into Equation 3.38 (with $N_E \mid N_P$ simplification)

$$P_T[\Lambda_E n] = \frac{1}{N_E}\left[\langle F\Lambda_E n\rangle_{N_P} - \langle F\Lambda_E n\rangle_{N_E}\right] \tag{4.30}$$

From the proof of Theorem 4.1, $N_E \mid F\Lambda_E$, and therefore $\langle F\Lambda_E n\rangle_{N_E} = 0$. Applying this knowledge results in

$$P_T[\Lambda_E n] = \frac{1}{N_E}\left[\langle F\Lambda_E n\rangle_{N_P}\right] \tag{4.31}$$

Next plugging the right hand side of Equation 4.29 into Equation 3.38 yields

$$P_T\left[\Lambda_E n + m\right] = \frac{1}{N_E}\left[\left\langle F\Lambda_E n + m\right\rangle_{N_P} - \left\langle F\Lambda_E n + m\right\rangle_{N_E}\right] \tag{4.32}$$

Applying Lemma 3.4 to $\left\langle F\Lambda_E n + m\right\rangle_{N_E}$ yields

$$\left\langle F\Lambda_E n + m\right\rangle_{N_E} = \left\langle \left\langle F\Lambda_E n\right\rangle_{N_E} + \left\langle m\right\rangle_{N_E}\right\rangle_{N_E} \tag{4.33}$$

$$= \left\langle \left\langle m\right\rangle_{N_E}\right\rangle_{N_E} = m \tag{4.34}$$

since $m < \Lambda_E \leq N_E$. Now it only needs to be shown that

$$\left\langle F\Lambda_E n + m\right\rangle_{N_P} - \left\langle F\Lambda_E n\right\rangle_{N_P} = m \tag{4.35}$$

By the division algorithm (Theorem 3.1), $F\Lambda_E n = N_P q_0 + r_0$, where $q_0 \in \mathbb{Z}$ and $0 \leq r_0 < N_P$. Also from the division algorithm, $F\Lambda_E n + m = N_P q_1 + r_1$ and $0 \leq r_1 < N_P$. Subtracting the previous two equations from each other

$$F\Lambda_E n + m - F\Lambda_E n = N_P q_1 + r_1 - N_P q_0 - r_0 \tag{4.36}$$

$$m = N_P(q_1 - q_0) + (r_1 - r_0) \tag{4.37}$$

Since $m < \Lambda_E \leq N_P$ and $0 \leq r_0, r_1 < N_P$, it is clear that $q_1 - q_0 = 0$ and $q_0 = q_1$. Consequently, $m = r_1 - r_0$. Plugging back into Equation 4.35,

$$\left\langle N_P q_1 + r_1\right\rangle_{N_P} - \left\langle N_P q_0 + r_0\right\rangle_{N_P} = r_1 - r_0 = m \tag{4.38}$$

□

From Theorem 3.9, it was noted that two frequency control words of the same period are simple permutations of each other. This leads to an interesting relationship that takes advantage of Theorem 4.4.

Theorem 4.5 (When Truncated Values Repeat)**.** *For any frequency control word F,*

$$P_T\left[\Lambda_E n\right] = P_T\left[\Lambda_E n + \Gamma_P^{-1} m\right] \tag{4.39}$$

for all $n \in \mathbb{Z}$ and $0 \leq m < \Lambda_E$ where Γ_P^{-1} is the multiplicative inverse of Γ_P modulo Λ_P and $N_E \mid N_P$.

Proof. This is easily shown by expanding $P_T\left[\Lambda_E n + \Gamma_P^{-1} m\right]$.

$$P_T\left[\Lambda_E n + \Gamma_P^{-1} m\right] = \frac{1}{N_E}\left[\left\langle F\Lambda_E n + F\Gamma_P^{-1} m\right\rangle_{N_P} - \left\langle\left\langle F\Lambda_E n + F\Gamma_P^{-1} m\right\rangle_{N_P}\right\rangle_{N_E}\right] \tag{4.40}$$

First consider the sequence $\left\langle F\Gamma_P^{-1} m\right\rangle_{N_P}$. Applying Lemma 3.3, it is easily shown that $\left\langle F\Gamma_P^{-1} m\right\rangle_{N_P} = d\left\langle \Gamma_P \Gamma_P^{-1} m\right\rangle_{\Lambda_P}$ where $d = \mathrm{GCD}\left(F, N_P\right)$. But Γ_P^{-1} is the multiplicative inverse of Γ_P modulo Λ_P by definition, that the multiplicative inverse exists comes from Theorem 3.8. Then $\left\langle F\Gamma_P^{-1} m\right\rangle_{N_P} = d\left\langle m\right\rangle_{\Lambda_P}$.

From the proof of Theorem 4.1, it was shown that $N_E \mid F\Lambda_E$ and therefore $\left\langle F\Lambda_E n\right\rangle_{N_E} = 0$. Then Equation 4.40 becomes

$$P_T\left[\Lambda_E n + \Gamma_P^{-1} m\right] = \frac{d}{N_E}\left[\left\langle \Gamma_P \Lambda_E n + m\right\rangle_{\Lambda_P} - \left\langle m\right\rangle_{\Lambda_E}\right] \tag{4.41}$$

Since $0 \leq m < \Lambda_E$, the same logic in the proof of Theorem 4.4 immediately applies and

$$P_T\left[\Lambda_E n\right] = P_T\left[\Lambda_E n + \Gamma_P^{-1} m\right] \tag{4.42}$$

□

Lemma 4.1 (Special Sub-Sequence Arrangement for Periodic Sequences)**.** *Any finite periodic sequence $f[n]$ with length Λ such that Λ is a composite positive integer (i.e. $\Lambda = aN$) can be decomposed into the sum of sub-sequences*

$$f[n] = \sum_{q=0}^{a-1} \sum_{r=0}^{N-1} \delta\left[n - \langle Nq + br \rangle_\Lambda\right] f\left[\langle Nq + br \rangle_\Lambda\right], \quad n \in \mathbb{Z} \tag{4.43}$$

where b is coprime to Λ.

Proof. Let $b \in \mathbb{Z}$ be coprime to Λ, then $\mathrm{GCD}(b, \Lambda) = 1$. Then $\langle br \rangle_\Lambda = \langle r \rangle_\Lambda$ by Lemma 3.2. The summation of Equation 4.43 then becomes

$$f[n] = \sum_{q=0}^{a-1} \sum_{r=0}^{N-1} \delta\left[n - \langle Nq + r \rangle_\Lambda\right] f\left[\langle Nq + r \rangle_\Lambda\right], \quad n \in \mathbb{Z} \tag{4.44}$$

which is the same as that shown in Theorem 4.2, and thus by the proof of that same theorem, $f[n]$ can be written equivalently as the sum of sub-sequences of the form given in Equation 4.43.

□

4.3 Spectrum in the Presence of Phase Truncation

Now we are ready to calculate the spectrum of a truncated phase sequence. All the work from Chapter 3 and the previous section of this chapter provide the tools necessary to compute the discrete Fourier transform (DFT) of an arbitrary DCDO with phase truncation. For completeness, the DFT is defined below

Definition 4.3 (Discrete Fourier Transform)**.** *If $x[n]$ is a discrete function that is periodic with N, then the DFT is*

$$X[k] = \sum_{n=0}^{N-1} x[n]\, e^{-j2\pi kn/N}, \quad 0 \le k \le N-1 \tag{4.45}$$

The DFT is typically used to describe the frequency content of the discrete waveform. The DFT is a reversible transformation and thus the inverse DFT is also defined

Definition 4.4 (Inverse Discrete Fourier Transform)**.**

$$X\left[k\right]=\frac{1}{N}\sum_{k=0}^{N-1}X\left[k\right]e^{j2\pi kn/N} \tag{4.46}$$

Theorem 4.6 (Spectrum of Truncated Phase Sequence)**.** *The DFT of a truncated phase sequence driving an ideal SCMF (i.e. no amplitude quantization) for when $N_E \mid N_P$ is*

$$S_T[k]=\frac{R_P V[k]}{2j}\left(\delta\left[\left\langle\Gamma_P-k\right\rangle_{R_P}\right]-\delta\left[\left\langle\Gamma_P+k\right\rangle_{R_P}\right]\right) \tag{4.47}$$

where $R_P \triangleq \frac{\Lambda_P}{\Lambda_E}$, $\delta[n]$ is the Kronecker delta function and

$$V\left[k\right]\triangleq\frac{1-e^{-j2\pi k\Lambda_E\Gamma_P^{-1}/\Lambda_P}}{1-e^{-j2\pi k\Gamma_P^{-1}/\Lambda_P}} \tag{4.48}$$

when $\Lambda_P \nmid k$ or $V[k]=\Lambda_E$.

Proof. Let the period of the error sequence Λ_E divide the period of the untruncated phase sequence Λ_P, then the truncated phase sequence has period Λ_P (Theorem 3.6). Now apply the reduced SCMF function to P_T

$$A_T[n]=\sin\left(\frac{2\pi}{N_Q}P_T[n]\right) \tag{4.49}$$

$A_T[n]$ has the same periodicity of $P_T[n]$ (which is Λ_P and calculated in Theorem 3.7), with the same error sequence periodicity (calculated in Theorem 3.5), so from Lemma 4.1 it can be written as the summation of sub-sequences as follows

$$A_T[n]=\sum_{l=0}^{\frac{\Lambda_P}{\Lambda_E}-1}\sum_{m=0}^{\Lambda_E-1}\delta\left[n-\left\langle\Lambda_E l+\Gamma_P^{-1}m\right\rangle_{\Lambda_P}\right]\sin\left(\frac{2\pi}{N_Q}P_T[n]\right) \tag{4.50}$$

That Γ_P^{-1} is coprime to Λ_P is certainly not difficult to show using the knowledge that Γ_P is coprime to Λ_P by definition. Now apply the Λ_P-point DFT to $A_T[n]$ to compute the spectral response

$$S_T[k] = \sum_{n=0}^{\Lambda_P-1} \sum_{l=0}^{\frac{\Lambda_P}{\Lambda_E}-1} \sum_{m=0}^{\Lambda_E-1} \delta\left[n - \left\langle \Lambda_E l + \Gamma_P^{-1} m \right\rangle_{\Lambda_P}\right] \sin\left(\frac{2\pi}{N_Q} P_T[n]\right) e^{-j2\pi kn/\Lambda_P} \tag{4.51}$$

From Theorem 4.3, the summations can be interchanged and we will do so for better readability. Interchanging the summation and applying the sifting property of the Kronecker delta,

$$S_T[k] = \sum_{l=0}^{\frac{\Lambda_P}{\Lambda_E}-1} \sum_{m=0}^{\Lambda_E-1} \sum_{n=0}^{\Lambda_P-1} \delta\left[n - \left\langle \Lambda_E l + \Gamma_P^{-1} m \right\rangle_{\Lambda_P}\right] \sin\left(\frac{2\pi}{N_Q} P_T[n]\right) e^{-j2\pi kn/\Lambda_P} \tag{4.52}$$

$$= \sum_{m=0}^{\Lambda_E-1} \sum_{l=0}^{\frac{\Lambda_P}{\Lambda_E}-1} \sin\left(\frac{2\pi}{N_Q} P_T\left[\left\langle \Lambda_E l + \Gamma_P^{-1} m \right\rangle_{\Lambda_P}\right]\right) e^{-j2\pi k\left\langle \Lambda_E l + \Gamma_P^{-1} m \right\rangle_{\Lambda_P}/\Lambda_P} \tag{4.53}$$

Since $0 \leq m < \Lambda_E$, Theorem 4.5 applies and $P_T[\Lambda_E l + \Gamma_P^{-1} m] = P_T[\Lambda_E l]$. Furthermore, $e^{-j2\pi k\left\langle \Lambda_E l + \Gamma_P^{-1} m \right\rangle_{\Lambda_P}/\Lambda_P}$ is periodic with Λ_P and the modulo operator can be dropped. Applying both these observations yields

$$S_T[k] = \sum_{m=0}^{\Lambda_E-1} \sum_{l=0}^{\frac{\Lambda_P}{\Lambda_E}-1} \sin\left(\frac{2\pi}{N_Q} P_T[\Lambda_E l]\right) e^{-j2\pi k\Lambda_E l/\Lambda_P} e^{-j2\pi k\Gamma_P^{-1} m/\Lambda_P} \tag{4.54}$$

$$= \left(\sum_{m=0}^{\Lambda_E-1} e^{-j2\pi k\Gamma_P^{-1} m/\Lambda_P}\right) \sum_{l=0}^{\frac{\Lambda_P}{\Lambda_E}-1} \sin\left(\frac{2\pi}{N_Q} P_T[\Lambda_E l]\right) e^{-j2\pi k\Lambda_E l/\Lambda_P} \tag{4.55}$$

Notice the critical observation that the exponential m term can be factored out of the inner summation. Applying the geometric series computation, Lemma 1.2, on the m summation yields

$$\sum_{m=0}^{\Lambda_E-1} e^{-j2\pi k\Gamma_P^{-1} m/\Lambda_P} = \frac{1 - e^{-j2\pi k\Gamma_P^{-1}\Lambda_E/\Lambda_P}}{1 - e^{-j2\pi k\Gamma_P^{-1}/\Lambda_P}} \tag{4.56}$$

Of course this is only for the case where $\Lambda_P \nmid k$, otherwise the summation evaluates to Λ_E as the exponential term evaluates to 1. Borrowing notation from Torosyan for consistency, we define

$$V[k] \triangleq \frac{1 - e^{-j2\pi k \Gamma_P^{-1} \Lambda_E / \Lambda_P}}{1 - e^{-j2\pi k \Gamma_P^{-1} / \Lambda_P}} \tag{4.57}$$

Notice however that this expression is different than derived in Torosyan's dissertation [31]. In particular, only the case for $F = 1$ is evaluated and the knowledge that the spectrum of sequences of equal least periods are simple rearrangements of each other. Here a completely general expression for any FCW has been calculated. Now from the proof of Theorem 4.4, it was shown that when $N_E \mid N_P$, as it does in the currently considered case

$$P_T[\Lambda_E n] = \frac{1}{N_E} \langle F \Lambda_E n \rangle_{N_P} \tag{4.58}$$

With this knowledge, consider the sine component of Equation 4.55,

$$\sin\left(\frac{2\pi}{N_Q} P_T[\Lambda_E l]\right) = \sin\left(\frac{2\pi}{N_P} \langle F \Lambda_E l \rangle_{N_P}\right) \tag{4.59}$$

$$= \sin\left(\frac{2\pi}{\Lambda_P} \langle \Gamma_P \Lambda_E l \rangle_{\Lambda_P}\right) \tag{4.60}$$

$$= \sin\left(\frac{2\pi}{\Lambda_P} (\Gamma_P \Lambda_E l)\right) \tag{4.61}$$

Reducing the modulo sequence was possible through Lemma 3.3 and by noting that $\Lambda_P = N_P / \mathrm{GCD}(F, N_P)$. The modulo operation was dropped because the sine function naturally performs the operation (Lemma 1.1). Then using Euler's formula 1.31,

$$\sin\left(\frac{2\pi}{\Lambda_P} (\Gamma_P \Lambda_E l)\right) = \frac{1}{2j}\left[e^{j2\pi \Gamma_P \Lambda_E l / \Lambda_P} - e^{-j2\pi \Gamma_P \Lambda_E l / \Lambda_P}\right] \tag{4.62}$$

Plugging back into Equation 4.55

$$S_T[k] = \frac{V[k]}{2j} \sum_{l=0}^{\frac{\Lambda_P}{\Lambda_E}-1} \left[e^{j2\pi \Gamma_P \Lambda_E l / \Lambda_P} - e^{-j2\pi \Gamma_P \Lambda_E l / \Lambda_P}\right] e^{-j2\pi k \Lambda_E l / \Lambda_P} \tag{4.63}$$

Consider now just the summation

$$\sum_{l=0}^{\frac{\Lambda_P}{\Lambda_E}-1} \left[e^{j2\pi\Gamma_P\Lambda_E l/\Lambda_P} - e^{-j2\pi\Gamma_P\Lambda_E l/\Lambda_P}\right] e^{-j2\pi k\Lambda_E l/\Lambda_P} = \sum_{l=0}^{\frac{\Lambda_P}{\Lambda_E}-1} e^{j2\pi\Lambda_E l(\Gamma_P-k)/\Lambda_P} - e^{-j2\pi\Lambda_E l(\Gamma_P+k)/\Lambda_P} \tag{4.64}$$

Now the exponential $e^{j2\pi\Lambda_E l(\Gamma_P-k)/\Lambda_P}$ and $e^{-j2\pi\Lambda_E l(\Gamma_P+k)/\Lambda_P}$ are periodic over k with Λ_P/Λ_E, which can easily be shown by replacing k with $k+\Lambda_P/\Lambda_E$ as shown below

$$e^{j2\pi\Lambda_E l(\Gamma_P-k-(\Lambda_P/\Lambda_E))/\Lambda_P} = e^{j2\pi\Lambda_E l(\Gamma_P-k)/\Lambda_P} e^{-j2\pi\Lambda_E l(\Lambda_P/\Lambda_E)/\Lambda_P} \tag{4.65}$$

$$= e^{j2\pi\Lambda_E l(\Gamma_P-k)/\Lambda_P} \tag{4.66}$$

So when $k = \Gamma_P + a\frac{\Lambda_P}{\Lambda_E}$, for $a \in \mathbb{Z}$ then the summation is equal to Λ_P/Λ_E otherwise the summation is equal to zero. This was shown in Section 1.2 by applying Lemma 1.2 but is shown here again as an example

$$\sum_{l=0}^{\frac{\Lambda_P}{\Lambda_E}-1} e^{j2\pi\Lambda_E l(\Gamma_P-k)/\Lambda_P} = \frac{1-e^{j2\pi(\Gamma_P-k)}}{1-e^{j2\pi\Lambda_E(\Gamma_P-k)/\Lambda_P}} \tag{4.67}$$

By setting $a=1$, it becomes clear that the summation is non-zero when $k-\Gamma_P \mid \frac{\Lambda_P}{\Lambda_E}$ which from the definition of the modulo operation $k = \Gamma_P \pmod{\frac{\Lambda_P}{\Lambda_E}}$. Therefore

$$\sum_{l=0}^{\frac{\Lambda_P}{\Lambda_E}-1} e^{j2\pi\Lambda_E l(\Gamma_P-k)/\Lambda_P} = \delta\left[\langle\Gamma_P - k\rangle_{(\Lambda_P/\Lambda_E)}\right] \tag{4.68}$$

Likewise, the other exponential term, $e^{-j2\pi\Lambda_E l(\Gamma_P+k)/\Lambda_P}$, is periodic over k with Λ_P/Λ_E, and therefore when $k = -\Gamma_P + a\frac{\Lambda_P}{\Lambda_E}$ then the summation is equal to Λ_P/Λ_E otherwise the summation is equal to zero. Using the analysis directly above, the final spectral response of

the truncated sequence is

$$S_T[k] = \frac{\Lambda_P V[k]}{\Lambda_E 2j} \left(\delta \left[\langle \Gamma_P - k \rangle_{(\Lambda_P/\Lambda_E)} \right] - \delta \left[\langle \Gamma_P + k \rangle_{(\Lambda_P/\Lambda_E)} \right] \right) \tag{4.69}$$

Letting $R_P = \frac{\Lambda_P}{\Lambda_E}$ and defining

$$S_E[k] = \frac{R_P}{2j} \left(\delta \left[\langle \Gamma_P - k \rangle_{R_P} \right] - \delta \left[\langle \Gamma_P + k \rangle_{R_P} \right] \right) \tag{4.70}$$

Then the final equation can be written as

$$S_T[k] = V[k] S_E[k] \tag{4.71}$$

□

Applying the logic of the previous proof to a cosine mapping function, as opposed to sine, yields

$$C_T[k] = V[k] C_E[k] \tag{4.72}$$

where $V[k]$ is the same window function of Theorem 4.6 and $C_E[k]$ is

$$C_E[k] = \frac{R_P}{2} \left(\delta \left[\langle \Gamma_P - k \rangle_{R_P} \right] + \delta \left[\langle \Gamma_P + k \rangle_{R_P} \right] \right) \tag{4.73}$$

For a quadrature system, where the output is represented as $CS_T[k] = C_T[k] + jS_T[k]$, the truncated output spectrum simplifies even further without any loss of generality.

$$CS_T[k] = \frac{R_P V[k]}{2} \delta \left[\langle \Gamma_P - k \rangle_{R_P} \right] \tag{4.74}$$

An even more powerful generalization can be made at this point about the relationship between phase truncation spurs and amplitude spurs from approximations and truncation.

This allows the analysis extend beyond single tone generation and be applied to arbitrary waveform generators or single tone generation with lossy compression.

Theorem 4.7 (DCDO Spectrum with Phase Truncation and Arbitrary ROM)**.** *The spurious response of an arbitrary ROM with an overflowing phase accumulator in the presence of truncation is*

$$S_G[k] = V[k]M[k], \quad k \in \{0, 1, 2, \ldots, \Lambda_P\} \tag{4.75}$$

where

$$M[k] = \mathcal{F}_{R_P}\left\{A[\Gamma_P n]\right\}[k], \tag{4.76}$$

$V[k]$ is the windowing function from Theorem 4.6, $\mathcal{F}_{R_P}\{\cdot\}$ is the R_P-point discrete Fourier transform and $A[\Gamma_P n]$ is the ROM sequence generated by the reduced frequency control word without phase truncation.

Proof. If one closely observes Equation 4.63, it is clear that the R_P-point DFT of the ROM values multiplied by the window function $V[k]$ yields the complete spectrum. In the analysis a sinusoid was used, but this certainly need not be the case. The ROM values repeat because of the characteristics of the truncated phase sequence, not the SCMF and thus $V[k]$ is **independent** from the choice of ROM values. Thus making A_T arbitrary, the contribution of the output spectrum from the ROM is

$$M[k] \triangleq \mathcal{F}_{R_P}\left\{A_T[\Gamma_P \Lambda_E n]\right\}[k] = \sum_{n=0}^{R_P-1} A_T[\Gamma_P \Lambda_E n] e^{-j2\pi kn/R_P} \tag{4.77}$$

$$= \sum_{n=0}^{R_P-1} A[\Gamma_P n] e^{-j2\pi kn/R_P} \tag{4.78}$$

$$= \mathcal{F}_{R_P}\left\{A[\Gamma_P n]\right\}[k], \tag{4.79}$$

Note that this is simply the discrete Fourier transform of the ROM **without** phase truncation. Finally, in its most general form, the output spectrum of a DCDO with an arbitrary

ROM is

$$S_G[k] = V[k]M[k] \tag{4.80}$$

where the subscript G is intended to represent "general." □

An interesting observation made by Torosyan in his dissertation is that the DFT commutes with sequences "of the form" of those generated by the overflowing phase accumulator. The proof is rather qualitative and "of the form" is never formally identified. Since the DFT commutes, the spectrums of two FCWs with the same least period for a DCDO are linear permutations of each other. This is not too surprising, since the time sequences in Theorem 3.9 have already been shown to be linear permutations of each other. Using the general closed form equation presented in Theorem 4.7, it is straightforward to prove the spectrums of the FCWs are simple linear permutations of each other.

Theorem 4.8 (FCW Frequency Sequence Permutation Relationship)**.** *The frequency response (DFT) of two different FCWs F_0 and F_1 driving an accumulator with N_P states are permutations of each other if $GCD(F_0, N_P) = GCD(F_1, N_P)$.*

Proof. Let $\text{GCD}(F_0, N_P) = \text{GCD}(F_1, N_P) = d$. Then from Theorem 4.7, the corresponding frequency domain representations of a full period for F_0 and F_1 are

$$S_{G0}[k] = V_0[k]M_0[k] \tag{4.81}$$

$$S_{G1}[k] = V_1[k]M_1[k] \tag{4.82}$$

If a coefficient α exists such that $V_0[\alpha k] = V_1[k]$ and $M_0[\alpha k] = M_1[k]$, then $S_{G0}[\alpha k] = S_{G1}[k]$ and the proof would be complete. Consider the respective window functions,

$$V_0[k] = \frac{1 - e^{-j2\pi k\Lambda_E\Gamma_{P0}^{-1}/\Lambda_P}}{1 - e^{-j2\pi k\Gamma_{P0}^{-1}/\Lambda_P}} \tag{4.83}$$

$$V_1[k] = \frac{1 - e^{-j2\pi k\Lambda_E\Gamma_{P1}^{-1}/\Lambda_P}}{1 - e^{-j2\pi k\Gamma_{P1}^{-1}/\Lambda_P}} \tag{4.84}$$

where Γ_{P0} and Γ_{P1} are multiplicative inverses modulo Λ_P of F_0 and F_1 reduced as per Lemma 3.3 respectively. Let $\alpha = \Gamma_{P1}^{-1}\Gamma_{P0}$. Recall that the complex exponential operation performs an equivalent modulo operation, then

$$V_0[\Gamma_{P1}^{-1}\Gamma_{P0}k] = \frac{1 - e^{-j2\pi\Gamma_{P1}^{-1}\Gamma_{P0}k\Lambda_E\Gamma_{P0}^{-1}/\Lambda_P}}{1 - e^{-j2\pi k\Gamma_{P0}^{-1}/\Lambda_P}} \tag{4.85}$$

$$= \frac{1 - e^{-j2\pi k\Lambda_E\Gamma_{P1}^{-1}/\Lambda_P}}{1 - e^{-j2\pi k\Gamma_{P0}^{-1}/\Lambda_P}} \tag{4.86}$$

$$= V_1[k] \tag{4.87}$$

Clearly then an α has been found that satisfies the window function rearrangement. Now consider $M_0[k]$ and $M_1[k]$.

$$M_0[k] = \sum_{n=0}^{R_P-1} A_{T0}[\Lambda_E n]e^{-j2\pi kn/R_P} \tag{4.88}$$

$$M_1[k] = \sum_{n=0}^{R_P-1} A_{T1}[\Lambda_E n]e^{-j2\pi kn/R_P} \tag{4.89}$$

Recall from Chapter 3 that A_{T0} and A_{T1} are

$$A_{T0}[n] = A[P_{T0}[n]] = A\left[\frac{d}{N_E}\left\langle\Gamma_{P0}\Lambda_E n\right\rangle_{\Lambda_P}\right] \tag{4.90}$$

$$A_{T1}[n] = A[P_{T1}[n]] = A\left[\frac{d}{N_E}\left\langle\Gamma_{P1}\Lambda_E n\right\rangle_{\Lambda_P}\right] \tag{4.91}$$

The simplification of the truncated phase sequence is shown in the proof of Theorem 4.6. Now check to see if $M_0[\alpha k] = M_1[k]$ by plugging Equation 4.90 into Equation 4.88.

$$M_0\left[\Gamma_{P1}^{-1}\Gamma_{P0}k\right] = \sum_{n=0}^{R_P-1} A\left[\frac{d}{N_E}\left\langle\Gamma_{P0}\Lambda_E n\right\rangle_{\Lambda_P}\right] e^{-j2\pi\Gamma_{P1}^{-1}\Gamma_{P0}kn/R_P} \tag{4.92}$$

$$= \sum_{j=0}^{R_P-1} A\left[\frac{d}{N_E}\left\langle\Gamma_{P0}\Lambda_E\Gamma_{P0}^{-1}\Gamma_{P1}j\right\rangle_{\Lambda_P}\right] e^{-j2\pi\Gamma_{P1}^{-1}\Gamma_{P0}k\Gamma_{P1}\Gamma_{P0}^{-1}j/R_P} \tag{4.93}$$

$$= \sum_{j=0}^{R_P-1} A\left[\frac{d}{N_E}\left\langle\Gamma_{P1}\Lambda_E j\right\rangle_{\Lambda_P}\right] e^{-j2\pi kj/R_P} = M_1\left[k\right] \tag{4.94}$$

The change of variable in the summation from n to j by setting $n = \Gamma_{P0}^{-1}\Gamma_{P1}j$ while keeping the summation limits the same is not trivially apparent. It is permissible in this instance because Γ_{P1} and Γ_{P0}^{-1} are both coprime to Λ_P and A and the exponential are simply rearrangements of the original sequence as both are effectively modulo Λ_P or $R_P = \Lambda_P/\Lambda_E$ (see Theorem 3.9 for phase accumulator sequence permutation explanation). By multiplying by a coprime number the least period of the sequence remains the same and all the values of A and the exponential are summed, albeit in a different order.

The same operations can also be used to show that $M_1[\Gamma_{P0}^{-1}\Gamma_{P1}k] = M_0[k]$. □

Since the spectrums of FCWs with the same least period are permutations of each other, it is important to determine the maximum possible number of least periods. It turns out that in the case of the phase accumulator, it is equivalent to deriving how many possible greatest common divisors are possible between N_P and F. This observation follows observing that the least period Λ_P is computed from N_P and F with $\Lambda_P = N_P/\gcd(F, N_P)$. Deriving the number of greatest common divisors between an arbitrary number F and a fixed number N_P is equivalent to deriving the total number distinct factors that can be formed from the prime factors of N_P. That the integer N_P has a unique prime factorization comes from the Unique Factorization Theorem [30].

Theorem 4.9 (Number of Phase Accumulator Least Periods)**.** *Let the prime-power decomposition of N_P be $p_n^{i_n}p_{n-1}^{i_{n-1}}\dots p_1^{i_1}$ where p_j is a prime factor of N_P and i_j is number of occurrences of p_j in N_P. The number of possible least periods with $F = 1, 2, \dots, N_P - 1$ is then $(i_n + 1)(i_{n-1} + 1)\dots(i_1 + 1) - 1$.*

Proof. Let the prime-power decomposition of N_P be $p_n^{i_n}p_{n-1}^{i_{n-1}}\dots p_1^{i_1}$. F will have some combination of prime factors in common with N_P, if not then F and N_P are coprime. So first we decide how many copies of p_n are common between F and N_P if F is allowed to equal N_P. The choices are 0, 1 up to i_n and thus there are $i_n + 1$ such choices. Next we decide

how many copies of p_{n-1}, and so forth to p_1. Thus there are

$$(i_n + 1)(i_{n-1} + 1) \dots (i_1 + 1) \tag{4.95}$$

possible combinations. But $F < N_P$ so the combination $\text{GCD}(F, N_P) = N_P$ is discarded, hence the subtraction of one. Note that $F = 0$ and $F = N_P$ are the same case since $P[n + N_P] = P[n]$. The final equation for the number of meaningful least periods is then

$$(i_n + 1)(i_{n-1} + 1) \dots (i_1 + 1) - 1 \tag{4.96}$$

□

As a quick sanity check, Torosyan and Nicholas claim that there are B_P such combinations for a B_P-bit accumulator. In their analyses both N_P and N_E are fixed to powers of two. Applying Theorem 4.9 with $N_P = 2^{B_P}$, it is clear that $p_1 = 2$ and $i_1 = B_P$ and thus there are $B_P + 1 - 1 = B_P$. Thus the specific case chosen works correctly.

To summarize the critical observations made in this chapter:

1. A closed form equation for the spectrum of a DCDO with an arbitrary mapping function in the presence of phase truncation is developed in Theorem 4.6 and Theorem 4.7.
   (a) The special case of sinusoids is explicitly solved.
   (b) The spectrum of the ROM and the window function can be computed independently and later used to quickly generate the exact expected spectrum.
2. The tones generated by amplitude mapping non-idealities are disjoint from the spurs from phase truncation.
3. The spectrums of FCWs with the same least period of simple permutations of each other which is shown in Theorem 4.8.

(a) This implies that a DCDO can be completely characterized by analyzing only FCWs that correspond to distinct least periods, which is the number of prime factors in N_P.

4.4 Interpreting Results

To gain an intuition into the results from Theorem 4.6, the periodicity of the terms of $S_G[k]$ is derived. This is accomplished by calculating the period of each multiplication term of $S_T[k]$. It has already been noted that the least period of $A_T[n]$ is Λ_P in the proof of Theorem 4.6, and thus an Λ_P-point DFT is taken to prevent any spectral leakage or aliasing. An N-point DFT is periodic with N, which is shown in the following lemma

Lemma 4.2 (DFT Periodicity)**.** *The N-point Discrete Fourier Transform is periodic with N.*

Proof. Let $X[k]$ be the DFT of a sequence $x[n]$ of length N. We wish to show that $X[k+N] = X[k]$.

$$X[k+N] = \sum_{n=0}^{N-1} x[n]e^{-j2\pi(k+N)/N} \tag{4.97}$$

$$= \sum_{n=0}^{N-1} x[n]e^{-j2\pi k/N}e^{-j2\pi N/N} \tag{4.98}$$

$$= \sum_{n=0}^{N-1} x[n]e^{-j2\pi k/N} = X[k] \tag{4.99}$$

The exponential is simplified by applying Euler's formula.

$$e^{-j2\pi} = \cos(-2\pi) + j\sin(-2\pi) = 1 + 0j = 1 \tag{4.100}$$

□

The first term of $S_G[k]$ under investigation is the window function $V[k]$. Recall again that under analysis is the case for when $N_E \mid N_P$. The least period of $V[k]$ is derived by calculating a such that $V[k+a] = V[k]$.

Lemma 4.3 (Window Function Periodicity)**.** *The window function $V[k]$ has least period Λ_P.*

Proof. As stated already, we are search for the smallest $a \in \mathbb{P}$ such that both

$$V[k+a] = \frac{1 - e^{-j2\pi(k+a)\Gamma_P^{-1}\Lambda_E/\Lambda_P}}{1 - e^{-j2\pi(k+a)\Gamma_P^{-1}/\Lambda_P}} \tag{4.101}$$

$$= \frac{1 - e^{-j2\pi k\Gamma_P^{-1}\Lambda_E/\Lambda_P} e^{-j2\pi a\Gamma_P^{-1}\Lambda_E/\Lambda_P}}{1 - e^{-j2\pi k\Gamma_P^{-1}/\Lambda_P} e^{-j2\pi a\Gamma_P^{-1}/\Lambda_P}} \tag{4.102}$$

Clearly if the newly added exponential terms equal one, then the equation holds. So we are searching for a such that

$$e^{-j2\pi a\Gamma_P^{-1}\Lambda_E/\Lambda_P} = 1 \tag{4.103}$$

$$e^{-j2\pi a\Gamma_P^{-1}/\Lambda_P} = 1 \tag{4.104}$$

The smallest such a that works for Equation 4.104 is $a = \Lambda_P$ because Γ_P^{-1} is coprime with Λ_P (from the proof of Theorem 4.6). The smallest such a for Equation 4.103 is $R_P = \Lambda_P/\Lambda_E$, which is less than Λ_P. If Λ_P is a period of Equation 4.103, then the least period for the window function has been found.

$$e^{-j2\pi\Lambda_P\Gamma_P^{-1}\Lambda_E/\Lambda_P} = e^{-j2\pi\Gamma_P^{-1}\Lambda_E} = 1 \tag{4.105}$$

Therefore $a = \Lambda_P$ is the least period of $V[k]$. □

The next term of $S_G[k]$ is the ROM spectrum $M[k]$. The following lemma derives the *typical* least period of $M[k]$. It does not hold for cases where the ROM itself has a least period less than the length of the ROM. The author knows of no cases where this would

occur in practice, as ROMs consume valuable chip real-estate and periodicity in the ROM implies that the size can be trivially compressed.

Lemma 4.4 (Period of Amplitude Spectrum with Phase Truncation)**.** *The function $M[k]$ from Theorem 4.7 is periodic with $R_P = \Lambda_P/\Lambda_E$.*

Proof. From Theorem 4.7, $M[k]$ is shown to be

$$M[k] = \mathcal{F}_{R_P}\left\{A_T[\Lambda_E n]\right\}[k] \tag{4.106}$$

From Lemma 4.2, an R_P-point DFT is periodic with R_P. There may be a smaller period for $M[k]$ for different A_T (e.g., consider a constant zero), but typically this is not the case. □

How many periods of $M[k]$ are in $S_G[k]$? Since $S_G[k]$ has least period Λ_P (from Lemma 4.2) and $M[k]$ has period $R_P = \Lambda_P/\Lambda_E$, the answer is clearly Λ_E. Practically, this means that the spectrum of the ROM is repeated Λ_E times across the spectrum while being modulated by the window function.

It is clear then that the spurs from phase truncation are "aliases" of the desired spectrum whose magnitude and phase are **solely** determined by the window function. Now we derive the shape of the function $V[k]$. Firstly, the expression can be rewritten in such a way as to clearly separate the magnitude of the window function from the phase.

$$\begin{aligned} V[k] &= \frac{1 - e^{-j2\pi k\Lambda_E\Gamma_{P0}^{-1}/\Lambda_P}}{1 - e^{-j2\pi k\Gamma_{P0}^{-1}/\Lambda_P}} \\ &= \left(\frac{e^{-j\pi k\Lambda_E\Gamma_{P0}^{-1}/\Lambda_P}}{e^{-j\pi k\Gamma_{P0}^{-1}/\Lambda_P}}\right)\frac{e^{j\pi k\Lambda_E\Gamma_{P0}^{-1}/\Lambda_P} - e^{-j\pi k\Lambda_E\Gamma_{P0}^{-1}/\Lambda_P}}{e^{j\pi k\Gamma_{P0}^{-1}/\Lambda_P} - e^{-j\pi k\Gamma_{P0}^{-1}/\Lambda_P}} \\ &= e^{-j\pi k(\Lambda_E-1)\Gamma_{P0}^{-1}/\Lambda_P}\left(\frac{\sin\left(\pi k\Lambda_E\Gamma_{P0}^{-1}/\Lambda_P\right)}{\sin\left(\pi k\Gamma_{P0}^{-1}/\Lambda_P\right)}\right) \end{aligned} \tag{4.107}$$

The conversion from the sum of two complex exponentials to the sine function is an application of Euler's formula (Equation 1.31). Clearly then for a given k, the phase and amplitude

of the truncation spur can be immediately, directly computed.

$$|V[k]| = \frac{\sin\left(\pi k \Lambda_E \Gamma_{P0}^{-1}/\Lambda_P\right)}{\sin\left(\pi k \Gamma_{P0}^{-1}/\Lambda_P\right)} \tag{4.108}$$

Now another interesting observation can be made. The numerator of $V[k]$ is periodic with R_P as shown in the proof of Lemma 4.3. This can also be trivially shown to be true for the numerator of $|V[k]|$. Recall that $M[k]$ is also periodic with R_P. This implies the magnitude of the ROM spectrum replicas within $M[k]$ are identical after multiplying by the numerator of the window function, which furthermore implies that the magnitude of the spurs are **completely** determined by the denominator of the window function. To reiterate, the magnitudes of the spurs from phase truncation are completely determined by analyzing the properties of a single cosecant term.

$$\frac{1}{\sin\left(\pi k \Gamma_P^{-1}/\Lambda_P\right)} = \csc\left(\frac{\pi k \Gamma_P^{-1}}{\Lambda_P}\right) \tag{4.109}$$

4.4.1 Ideal SCMF Example

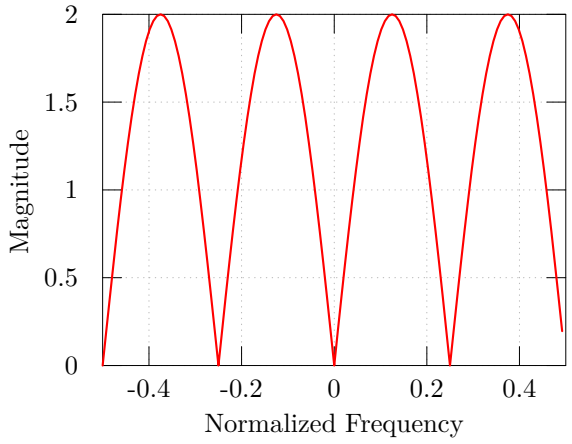
A visual explanation of the preceding will assist in understanding the conclusions. An SCMF must be selected and the size of the phase accumulator must be selected. Let us analyze a sine-only SCMF with no amplitude truncation. The spectrum, given by $S_T[k]$, of a such setup was derived in Theorem 4.6. Next example values for the phase accumulator must be selected in order to populate the data required for the plots. Let us choose $\Gamma_P = 1$. This is not an arbitrary choice, as it has some interesting properties that allows for the direct computation of the N largest phase truncation spurs. Next we choose $\Lambda_E = 4$ and $\Lambda_P = 128$, and these selections are arbitrary other than that $\Lambda_E \mid \Lambda_P$ (a requirement in the current analysis). Λ_E is also chosen small to make the resulting plots more readable (recall from the previous section that Λ_E replicas of the ROM spectrum are present in $S_G[k]$).

First the multiplicative inverse of Γ_P modulo Λ_P is computed. The selection of $\Gamma_P = 1$ makes such a computation obvious, as $1 \cdot 1 = 1$ and thus $\Gamma_P^{-1} = 1$. The magnitude of this particular case is then found by plugging the selected values into Equation 4.108

$$V[k] = \frac{\sin(\pi k/32)}{\sin(\pi k/128)}, \quad k = \{0, 1, \ldots, 127\} \tag{4.110}$$

As Torosyan in his dissertation, the numerator and denominator of the above window function are plotted independently. Figure 4.3a shows the magnitude of window function numerator and Figure 4.3b shows the magnitude of the window function denominator. Note that the numerator has $\Lambda_E = 4$, which matches the theory from Section 4.4.

(a) Window Function Numerator



(b) Window Function Denominator

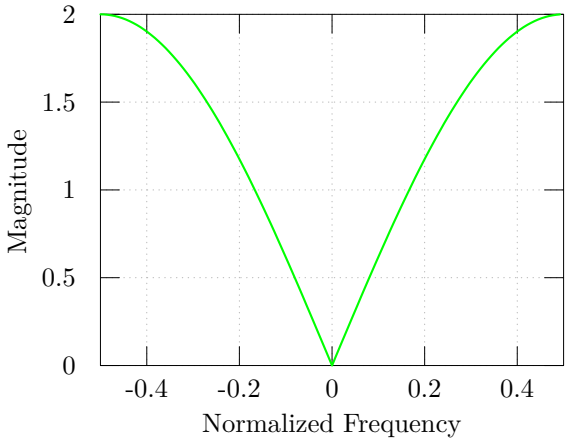


Figure 4.3: Window Function from Example

Now $S_E[k]$ (Equation 4.70) can also be computed given the selected values.

$$S_E[k] = \frac{16}{j}\left(\delta\left[\langle 1 - k\rangle_{32}\right] - \delta\left[\langle 1 + k\rangle_{32}\right]\right) \tag{4.111}$$

Thus between $k = \{-64, -63, \ldots, 63\}$, tones appears at $k = \{-64 + 1, -32 + 1, 1, 32 + 1\}$ and $k = \{-32 - 1, -1, 32 - 1, 64 - 1\}$. Figure 4.4a is a plot of $S_E[k]$. If only considering the the spectrum from 0 to $f_{ny}/2$, there are 4 unique tones, only 1 of which is the desired

ton and the others are spurs. This can be generalized to the ideal SCMF case, there are $\Lambda_E - 1$ spurs, which is to say that only one replica is desired, the others are spurs. From the previous discussion, it was noted that the magnitude of the numerator of the window function is periodic with the $S_E[k]$ and Figure 4.4b shows the multiplication of the two terms.

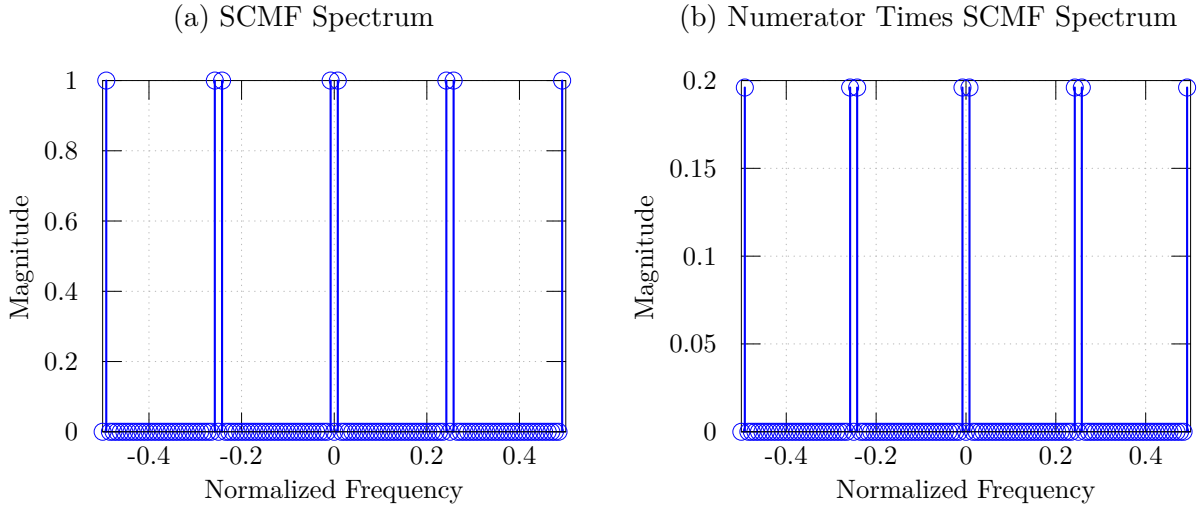


Figure 4.4: Window Function from Example

Which of the replicas are spurs and which of these are the intended generated tone? Consider the untruncated case to act as a guide to the discussion. In Section 1.2, no number theory reductions are made in the analysis and a N_P-point DFT is performed on the output of the ideal SCMF. It was noted in the proof of Theorem 4.6 that the SCMF output is actually periodic with Λ_P, so a Λ_P-point DFT should be sufficient to capture the data.

$$S[k] = \sum_{n=0}^{\Lambda_P - 1} \sin\left(\frac{2\pi}{N_P} \langle Fn \rangle_{N_P}\right) e^{-j2\pi kn/\Lambda_P} \tag{4.112}$$

$$= \sum_{n=0}^{\Lambda_P - 1} \sin\left(\frac{2\pi}{\Lambda_P} \langle \Gamma_P n \rangle_{\Lambda_P}\right) e^{-j2\pi kn/\Lambda_P} \tag{4.113}$$

Applying Euler's formula and computing the summation,

$$S[k] = \frac{1}{2j}\left[\delta\left[\langle \Gamma_P - k \rangle_{\Lambda_P}\right] - \delta\left[\langle \Gamma_P + k \rangle_{\Lambda_P}\right]\right] \tag{4.114}$$

This lead provides enough information to hypothesize that the fundamental tones for $S_T[k]$ are located at $k = \Gamma_P$ and $k = -\Gamma_P$ are the fundamental tones. Back to the particular problem under consideration, the desired tone is at $k = -1, 1$.

The two portions of information left to consider are the SFDR and SNR due to phase truncation. This is addressed in Section 4.6. But in the next section, a more complex example is provided with the full $S_T[k]$ spectrum computed.

4.5 Numerical Verification of Theory

This section provides several examples to demonstrate the developed phase truncation. The theory can be fully verified without any measured results, since the analysis applies to the digital portion of a DDFS. Here complete visibility into the behavior of the device can be achieved through simulation. One important note is that long FFT computations will start to show noticeable computational error. This results from the finite precision of the floating point precision calculations. The direct computation from Theorem 4.6 requires fewer operations, for a 32-bit accumulator over a billion fewer mathematical operations, and results in a more precise answer.

Figure 4.5a shows the absolute error between the theory and full simulation of a 12-bit accumulator with the least 4 bits truncated with frequency control word 7. The error is presented in this manner because plotting the spectrums of simulation versus theory on the same plot causes the results to be overlaid in such as way as to be indistinguishable (as shown in Figure 4.5b).

4.6 SFDR and SNR in the Presence of Phase Truncation

In this section, the theory from Section 4.3 is used to understand the impact of truncation on the system performance metrics such as spurious-free dynamic range (SFDR) and signal to noise ratio (SNR) . When writing about "performance metrics", care is taken to be precise as to the meaning of the terminology. The SFDR is defined as the ratio of the power

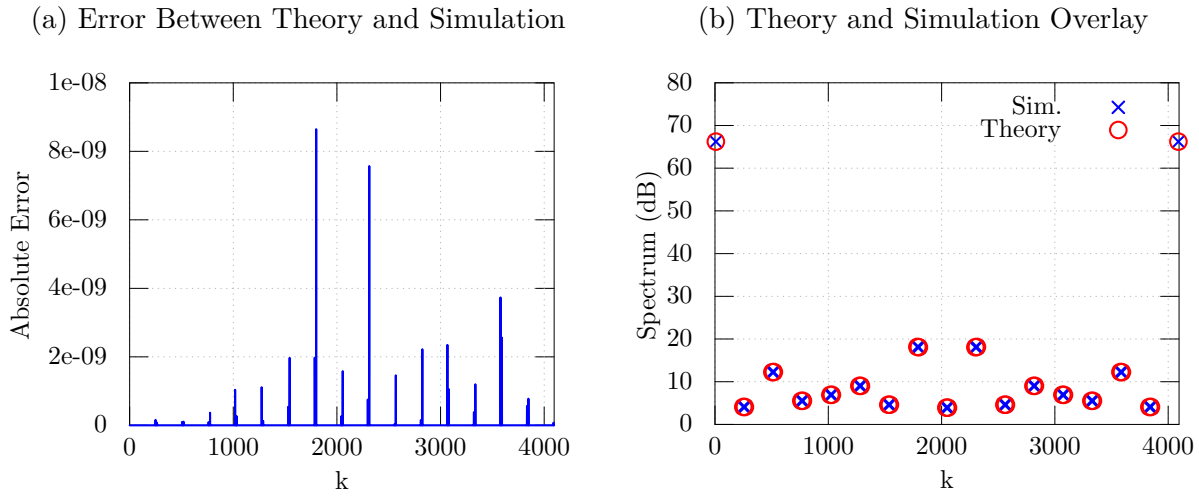


Figure 4.5: Numerical Validation

of a single fundamental tone and the power of the largest spurious tone in the spectrum [32]. Equation 4.115 expresses the relationship in a mathematical form,

$$\text{SFDR} = \frac{P_{\text{fund}}}{P_{\text{spur-max}}} \tag{4.115}$$

where P_{fund} is the power of the single fundamental tone at the output port and $P_{\text{spur-max}}$ is the power of the largest spur measured at the output port. SFDR is regularly reported in decibels,

$$\text{SFDR}_{\text{db}} = 10\log_{10}\left(\frac{P_{\text{fund}}}{P_{\text{spur-max}}}\right) \tag{4.116}$$

The SNR is defined as the ratio of the power of the fundamental tone to all non-harmonically related spur and noise power. As with SFDR, SNR is most often reported in decibels.

$$\text{SNR} = 10\log_{10}\left(\frac{P_{\text{fund}}}{\sum P_{\text{spur}} - \sum P_{\text{harm}}}\right) \tag{4.117}$$

Of interest in the analysis is whether the phase truncation generates spurs harmonically related to the fundamental tone. The total harmonic distortion (THD) is defined as the ratio

of the power of the single fundamental tone and the summed power all tones harmonically related to the fundamental tone.

$$\text{THD} = 10\log_{10}\left(\frac{P_{\text{fund}}}{\sum_{k=2}^{\infty} P_k}\right) \tag{4.118}$$

where P_k is power of the k^{th} harmonic relative to the fundamental tome P_{fund}. Again, the result is generally reported in decibels.

4.6.1 SFDR

In Section 4.4, it was noted that the denominator of the window function completely determines the relationship between ROM spectrum replicas. In this section, the SFDR of the ideal SCMF in the presence of phase truncation is derived. To begin the analysis, consider Figure 4.3b from the example. Keep in mind that this term is in the denominator, thus *lower* values corresponds to a *larger* tones. For the $\Gamma_P = 1$ case, the magnitude of the denominator of the window function monotonically increases with increasing k from zero to $\Lambda_P/2 - 1$ and monotonically increases with decreasing k from zero to $-\Lambda_P/2$.

This is always the case for $\Gamma_P = 1$, regardless of the value of Λ_P. This can be seen by plugging $\Gamma_P^{-1} = 1$ into the denominator of the window function (Equation 4.109).

$$\frac{1}{\sin\left(\frac{\pi k}{\Lambda_P}\right)} = \csc\left(\frac{\pi k}{\Lambda_P}\right) \tag{4.119}$$

Plugging in at the extreme values, $\sin(0) = 0$ and $|\sin(\pi(-\Lambda_P/2)/\Lambda_P)| = \sin(\pi/2) = 1$ and lastly $|\sin(\pi(\Lambda_P/2)/\Lambda_P)| = \sin(\pi/2) = 1$. Therefore the largest replicas for $\Gamma_P = 1$ case occur where the the denominator is least or the nearest tone to the right of $k = 1$ and to the left of $k = -1$. Going back to the example, the worst case spur is located at

$$k_{\text{spur}} = R_P - 1 \tag{4.120}$$

The next worst spur would occur at R_P+1 then $2R_P-1$ and so forth such that the locations of the spurs are

$$k_{\text{spur}} \in \{nR_P - 1, nR_P + 1\}, \quad n \in \{0, 1, \ldots, \Lambda_E/2\} \tag{4.121}$$

and the magnitudes of the spurs are arranged from largest to smallest with increasing value of k and consequently increasing value of n. Recall that only the relationship of the denominator matters, therefore the SFDR for the $\Gamma_P = 1$ case can be written as the ratio of the denominators of two window functions (writing them as the ratio of two window functions would work as well, but the numerators cancel out anyway).

$$\frac{\csc(\pi k_{\text{fund}}/\Lambda_P)}{\csc(\pi k_{\text{spur}}/\Lambda_P)} = \frac{\csc(\pi/\Lambda_P)}{\csc(\pi R_P/\Lambda_P - \pi/\Lambda_P)} \tag{4.122}$$

$$= \frac{\sin(\pi R_P/\Lambda_P - \pi/\Lambda_P)}{\sin(\pi/\Lambda_P)} \tag{4.123}$$

Recalling that $R_P = \Lambda_P/\Lambda_E$,

$$\frac{\sin(\pi(\Lambda_P/\Lambda_E)/\Lambda_P - \pi/\Lambda_P)}{\sin(\pi/\Lambda_P)} = \frac{\sin(\pi/\Lambda_E - \pi/\Lambda_P)}{\sin(\pi/\Lambda_P)} \tag{4.124}$$

Using the argument difference trigonometric identity for sine, the equation reduces further to

$$\frac{\sin(\pi/\Lambda_E)\cos(\pi/\Lambda_P) - \cos(\pi/\Lambda_E)\sin(\pi/\Lambda_P)}{\sin(\pi/\Lambda_P)} = \sin(\pi/\Lambda_E)\cot(\pi/\Lambda_P) - \cos(\pi/\Lambda_E) \tag{4.125}$$

It is now important to reiterate how the least periods are related to the number of states in the phase accumulator and the FCW. Both Λ_P and Λ_E are functions of F, so

Equation 4.125 can be rewritten in terms of the original design parameters.

$$\Lambda_P = \frac{N_P}{\text{GCD}\,(F, N_P)} \tag{4.126}$$

$$\Lambda_E = \frac{N_E}{\text{GCD}\,(F, N_E)} \tag{4.127}$$

Substituting back into Equation 4.125

$$\text{SFDR} = \sin\left(\frac{\pi \text{GCD}(F, N_E)}{N_E}\right) \cot\left(\frac{\pi \text{GCD}(F, N_P)}{N_P}\right) - \cos\left(\frac{\pi \text{GCD}(F, N_E)}{N_E}\right) \tag{4.128}$$

An alternative expression could be created substituting back into Equation 4.124 as well

$$\text{SFDR} = \frac{\sin\left(\frac{\pi \text{GCD}(F,N_E)}{N_E} - \frac{\pi \text{GCD}(F,N_P)}{N_P}\right)}{\sin\left(\frac{\pi \text{GCD}(F,N_P)}{N_P}\right)} \tag{4.129}$$

Recall that $N_E \mid N_P$ in the current analysis which by definition implies there is an integer c such that $cN_E = N_P$. But $N_Q = N_P/N_E$ has already been defined as the number of address states in the SCMF. $N_E \mid N_P$ for there to be any realizable optimization in hardware due to truncation. So there is further simplification to the expression yet.

$$\text{GCD}(F, N_P) = \text{GCD}(F, N_Q N_E) = \text{GCD}(F, N_E)\text{GCD}(\Gamma_E, N_Q) \tag{4.130}$$

where $\Gamma_E = F/\text{GCD}(F, N_E)$. Substituting this into Equation 4.129

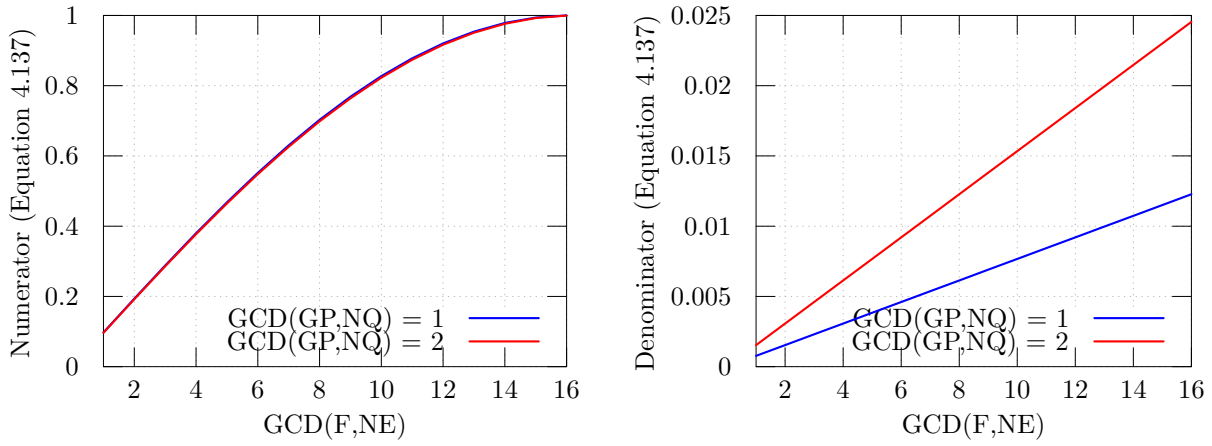
$$\text{SFDR} = \frac{\sin\left(\pi \text{GCD}(F, N_E)\left[\frac{1}{N_E} - \frac{\text{GCD}(\Gamma_E, N_Q)}{N_P}\right]\right)}{\sin\left(\frac{\pi \text{GCD}(F,N_E)\text{GCD}(\Gamma_E,N_Q)}{N_P}\right)} \tag{4.131}$$

If $\text{GCD}(F, N_E) = N_E$ then there is no truncation and infinite SFDR. Therefore, for any interesting SFDR value, the argument $\text{GCD}(F, N_E)/N_E$ has a minimum value of $1/N_E$ and maximum possible value of $1/p_l$ where p_l is the least prime factor of N_E. Without loss of generality, the smallest prime integer is 2 and therefore the maximum possible value of the

term is 1/2. Now consider the argument $\text{GCD}(\Gamma_E, N_Q)/N_P$, it has a minimum value of $1/N_P$ and a maximum possible value of $(N_Q/2)/N_P = 1/2N_E$.

4.6.2 Worst Case SFDR

To gain an intuition for how the SFDR changes with N_P, N_E and F, the numerator and denominator of Equation 4.131 are analyzed. First we fix $\text{GCD}(\Gamma_E, N_Q)$ and see how the numerator and denominator change with increasing $\text{GCD}(F, N_E)$. $N_E = 32$ and $N_P = 4096$ in the plotted figures. From the plot of the numerator in Figure 4.6a and the denominator in Figure 4.6b, it appears both increase monotonically with increasing $\text{GCD}(F, N_E)$. Closely inspecting the arguments in the numerator and denominator reveal this to be the case as well, as the arguments increasing monotonically with increasing (recall that the maximum value of $\text{GCD}(\Gamma_E, N_Q)/N_P = 1/(2N_E)$ so $1/N_E - 1/(2N_E)$ is always a positive quantity). Since sine monotonically increases from 0 to π and the limits of the arguments are computed in the previous paragraph and shown to be within that interval, it is always the case that both increase monotonically.



(a) Numerator of SFDR Function

(b) Denominator of SFDR Function

Figure 4.6: Numerical Validation

Now it must be determined whether the Equation 4.131 decreases or increases with increasing GCD(F, N_E). Figure 4.7 certainly seems to indicate that the SFDR decreases with increasing values of GCD(F, N_E). There is another way to show prove this to be the

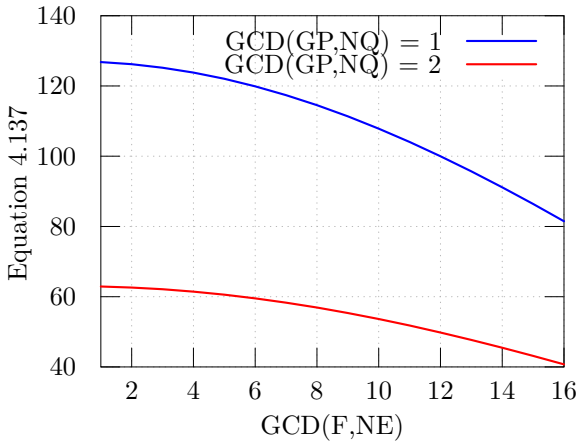


Figure 4.7: SFDR Function (Magnitude)

case for arbitrary N_E and N_P. Instead of computing the derivative and dealing with difficult trigonometry, several values of GCD(F, N_E) can be computed. First let GCD$(F, N_E) = 1$,

$$\text{SFDR}_1 = \frac{\sin\left(\pi\left(\frac{1}{N_E} - \frac{\text{GCD}(\Gamma_E, N_Q)}{N_P}\right)\right)}{\sin\left(\frac{\pi\text{GCD}(\Gamma_E, N_Q)}{N_P}\right)} \tag{4.132}$$

The next prime number after 1 is 2, so now GCD$(F, N_E) = 2$ is evaluated

$$\text{SFDR}_2 = \frac{\sin\left(2\pi\left(\frac{1}{N_E} - \frac{\text{GCD}(\Gamma_E, N_Q)}{N_P}\right)\right)}{\sin\left(\frac{2\pi\text{GCD}(\Gamma_E, N_Q)}{N_P}\right)} \tag{4.133}$$

Applying the double angle formula to the both the sine term in the numerator and denominator yields

$$\text{SFDR}_2 = \frac{2\sin\left(\pi\left(\frac{1}{N_E} - \frac{\text{GCD}(\Gamma_E, N_Q)}{N_P}\right)\right)\cos\left(\pi\left(\frac{1}{N_E} - \frac{\text{GCD}(\Gamma_E, N_Q)}{N_P}\right)\right)}{2\sin\left(\frac{\pi\text{GCD}(\Gamma_E, N_Q)}{N_P}\right)\cos\left(\frac{\pi\text{GCD}(\Gamma_E, N_Q)}{N_P}\right)} \tag{4.134}$$

Note that the SFDR_1 term appears as a multiplication term in SFDR_2. And thus the previous equation reduces to

$$\mathrm{SFDR}_2 = \mathrm{SFDR}_1 \left[\frac{\cos\left(\pi\left(\frac{1}{N_E} - \frac{\mathrm{GCD}(\Gamma_E, N_Q)}{N_P}\right)\right)}{\cos\left(\frac{\pi \mathrm{GCD}(\Gamma_E, N_Q)}{N_P}\right)} \right] \tag{4.135}$$

Recall that cosine monotonically decreases from zero to $\pi/2$. From the previous discussions, it was noted that $2N_E \leq N_P$ and typically $N_E \ll N_P$. For the smallest value of $\mathrm{GCD}(\Gamma_E, N_Q)$, i.e. 1, the denominator clearly dominates since $\pi/N_P < \pi/(1/N_E - 1/N_P)$ and thus $\mathrm{SFDR}_2 < \mathrm{SFDR}_1$. For the largest possible value of $\mathrm{GCD}(\Gamma_E, N_Q)$, i.e. $N_Q/2$, $\pi/(2N_E) = \pi/2N_E$, in which case $\mathrm{SFDR}_2 = \mathrm{SFDR}_1$. Thus for every possible value of $\mathrm{GCD}(\Gamma_E, N_Q)$ except for one case the SFDR *decreases* with increasing $\mathrm{GCD}(F, N_E)$ and in that case, the SFDR is flat.

This means that for a given number of phase and error states, the worst case SFDR can be directly calculated. It occurs for the largest possible $\mathrm{GCD}(F, N_E)$ less than N_E and the largest possible $\mathrm{GCD}(\Gamma_E, N_Q)$ less than N_Q. First set the largest possible $\mathrm{GCD}(F, N_E) = N_E/p_e$, where p_e is the least prime factor of N_E. Substituting this expression into Equation 4.131

$$\frac{\sin\left(\pi\left(\frac{N_E}{p_e}\right)\left(\frac{1}{N_E} - \frac{\mathrm{GCD}(\Gamma_E, N_Q)}{N_P}\right)\right)}{\sin\left(\pi\left(\frac{N_E}{p_e}\right)\left(\frac{\mathrm{GCD}(\Gamma_E, N_Q)}{N_P}\right)\right)} = \frac{\sin\left(\frac{\pi}{p_e} - \frac{\pi \mathrm{GCD}(\Gamma_E, N_Q)}{p_e N_Q}\right)}{\sin\left(\frac{\pi \mathrm{GCD}(\Gamma_E, N_Q)}{p_e N_Q}\right)} \tag{4.136}$$

$$= \sin\left(\frac{\pi}{p_e}\right)\cot\left(\frac{\pi \mathrm{GCD}(\Gamma_E, N_Q)}{p_e N_Q}\right) - \cos\left(\frac{\pi}{p_e}\right) \tag{4.137}$$

Now the we simply insert the worst case value for $\mathrm{GCD}(\Gamma_E, N_Q) = N_Q/c_q$ to arrive at the final value. Here c_q ranges from N_Q if Γ_E and N_Q are coprime to p_q, the smallest prime factor that is common to Γ_E and N_Q, if they are not coprime. Finally, this yields

$$\mathrm{SFDR}_{wc} = \sin\left(\frac{\pi}{p_e}\right)\cot\left(\frac{\pi}{p_e c_q}\right) - \cos\left(\frac{\pi}{p_e}\right) \tag{4.138}$$

Note that Equation 4.138 is an exact expression for the worst possible SFDR due to phase truncation spurs for all possible FCW combinations under the assumptions presented in this section.

An expression was developed for the special case where N_P and N_E are powers of two in Torosyan's dissertation. If the analysis above is correct, then Equation 4.138 should reduce to the following:

$$\mathrm{SFDR}_{wc} = \cot\left(\frac{\pi}{2N_Q}\right). \tag{4.139}$$

Which is a surprising result considering all the complicated analysis required to arrive here. This is one of the few times where the final solution to the exact analysis provides a simpler result than the solution arrived at by approximations. Let N_P and N_E be powers of two. Then the smallest prime factor of N_E is $p_e = 2$. The largest value of $\mathrm{GCD}(\Gamma_E, N_Q)$ is 1. This is because N_Q is a power of two as well. But for truncation spurs to exist, $\mathrm{GCD}(F, N_E)$ must be less than N_E (or else there is nothing to truncate, because the truncation error sequence is always 0). $\sin(\pi/2) = 1$ and $\cos(\pi/2) = 0$, so the final result is indeed

$$\sin\left(\frac{\pi}{2}\right)\cot\left(\frac{\pi}{2N_Q}\right) - \cos\left(\frac{\pi}{2}\right) = \cot\left(\frac{\pi}{2N_Q}\right) \tag{4.140}$$

An interesting observation can be made here that unites Jenq's analysis of SNR with the SFDR analysis in this section. Equation 2.40 is exactly the same as Equation 4.139, which means the worst case SFDR is equal to the best case SNR. This is because all of the harmonic energy is stored in a single spur at the worst case SFDR. The SNR for a spectrum with a single spur is always equal to the SFDR for the same spectrum. Considering that the two techniques arrived to the same conclusion through very different paths.

4.6.3 Spur Locations

Using Theorem 4.8, the spectrums of frequency words with the same least period are simple linear permutations of each other and thus the SFDR does not change for a given Λ_P. Let Γ_{P1} be an arbitrary reduced FCW with same Λ_P as the $\Gamma_P = 1$ example. Applying the permutations to spur locations, a closed form solution for the locations of spurs can be derived,

$$k_{\text{spur}}\Gamma_{P1}^{-1}\Gamma_P = k_{\text{spur}}\Gamma_{P1}^{-1} \tag{4.141}$$

$$= \left\{n\Gamma_{P1}^{-1}R_P - 1, n\Gamma_{P1}^{-1}R_P + 1\right\}, \quad n \in \{0, 1, \ldots, \Lambda_E/2\} \tag{4.142}$$

where the magnitudes of the spurs are still sorted by increasing n, though no longer increasing k. To find the order of spur magnitude, simply apply the frequency permutation to find its location for the $\Gamma_P = 1$ case.

Note that the number of spurs and spur locations is only for a *pure sinusoidal* case. From Theorem 4.7, it is clear that the *entire* spectrum of the ROM is copied Λ_E times over the full spectrum. Thus if there are hundreds of amplitude quantization spurs or compression spurs, then **all** of the quantization spurs are copied in the spectrum. Thus phase truncation can result in thousands of spurs being introduced into the spectrum. This observation does not fall out of previous DDFS analysis to the author's knowledge.

4.6.4 SNR

The signal to noise ratio can also be computed from the closed form analysis of phase accumulators. The first closed form analysis of this was performed by Jenq in his work and is given in Chapter 2. To calculate the SNR using this technique, all the spurs must be calculated directly. Since the spurs can be computed directly without computing the full DFT, this calculation is not as resource intensive as it may seem. Also, the calculation need only be calculated for the total number of possible least periods of the phase accumulator

(Theorem 4.9). Jenq's technique only provides an upper bound and may not be used in the case of a lossy sinusoidal compression technique. The full spectrum calculation in this work actually uses the the values from the SCMF (Theorem 4.7).

4.7 Architecture Changes for Improved Spurious Response

From the observations in Section 4.3 and Section 4.6, simple modifications to phase accumulator architecture can yield interesting results. The analysis has shown that small changes to the FCW result in dramatically different output spectrums. The system can be made agnostic to the FCW choice producing the same SFDR and SNR.

4.7.1 Force Coprime FCWs

One technique for improving the worst case SFDR of the DCDO in the presence of phase truncation is to force an FCW that is coprime to the number of states (N_P) in the phase accumulator. This technique, though not described in such terms, is originally proposed by Nicholas in [23]. The computational complexity of the technique is rather small, as only a single bit needs to be added to the phase accumulator. This is accomplished by inserting a toggle flip-flop to the carry-in of the first full-adder of the phase accumulator adder. Conceptually, half of the available phase accumulator states are being discarded, meaning that an unmodified phase accumulator that only accepted odd-valued FCWs would produce the same results. The difference in Nicholas' suggestion is that the hardware toggles a 1 on the carry-in of LSB full-adder to make every FCW coprime (i.e. an efficient hardware realization). Figure 4.8 shows a block diagram of the modification necessary for the implementation.

The drawback of this implementation is that the SNR of the is degraded and spurs are introduced in FCWs that do not experience phase truncation. Using Equation 4.137 and

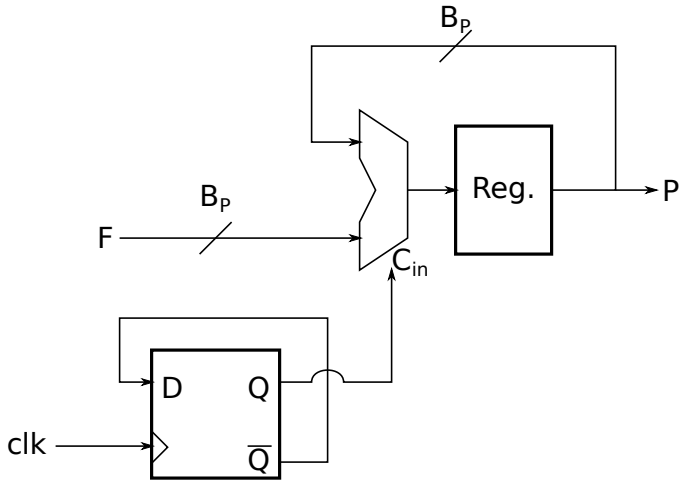


Figure 4.8: Forcing Coprime FCWs

Equation 4.139, the improvement to the worst case SFDR without the modification is

$$\text{SFDR}_{wc1} = \cot\left(\frac{\pi}{2N_Q}\right) = \cot\left(\frac{\pi N_E}{2N_P}\right) \tag{4.143}$$

The worst case SFDR with the modification is the SFDR of a coprime FCW driving a conventional phase accumulator. Let $\text{GCD}(F, N_E) = 1$ and $\text{GCD}(F, N_P) = 1$, which is what it means to force a coprime FCW. Using Equation 4.125 and noting that $\Lambda_P = N_P$ and $\Lambda_E = N_E$ for this case,

$$\text{SFDR}_{wc2} = \sin\left(\frac{\pi}{N_E}\right)\cot\left(\frac{\pi}{N_P}\right) - \cos\left(\frac{\pi}{N_E}\right) \tag{4.144}$$

Now it simply needs to be shown that this is an improvement over Equation 4.143. This can be done by dividing SFDR_{wc2} by SFDR_{wc1}, which yields

$$\frac{\text{SFDR}_{wc2}}{\text{SFDR}_{wc1}} = \frac{\sin\left(\frac{\pi}{N_E}\right)\cot\left(\frac{\pi}{N_P}\right) - \cos\left(\frac{\pi}{N_E}\right)}{\cot\left(\frac{\pi N_E}{2N_P}\right)} \tag{4.145}$$

$$= \csc\left(\frac{\pi}{N_P}\right)\tan\left(\frac{\pi N_E}{2N_P}\right)\sin\left(\frac{\pi}{N_E} - \frac{\pi}{N_P}\right) \tag{4.146}$$

Figure 4.9 numerically shows that SFDR_{wc2} is always better than SFDR_{wc1} by 1.88 dB, which is the extreme case where $N_P - 2$ bits are truncated. In general the improvement of the worst case SFDR is better than 3.9 dB but not by much. The expression asymptotically approaches a value around 3.922 dB which prevents any further improvement with this methodology.

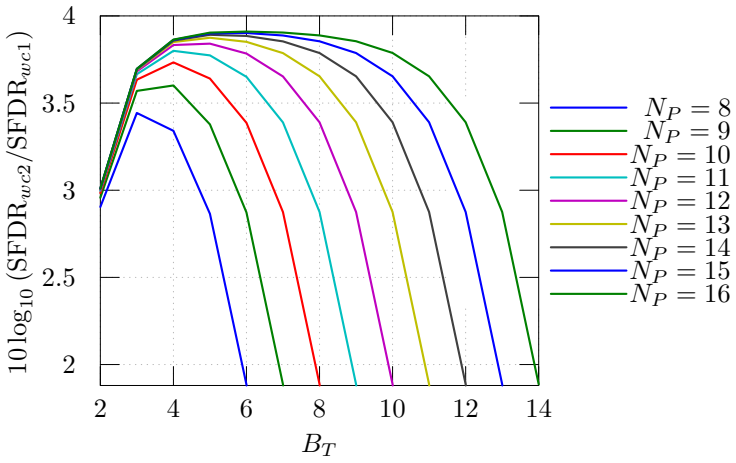


Figure 4.9: Modification SFDR Improvement

The technique by Nicholas impacts the frequency output of the DDFS. It effectively adds an extra bit to the accumulator but forces the FCW to 1.

$$f_0 = \frac{F}{N_P} f_{\text{clk}} + \frac{1}{2N_P} f_{\text{clk}} \tag{4.147}$$

Another simple modification can be made to prevent spurs from being generated in cases where phase truncation does not occur. This technique is novel to this work to the author's knowledge. Applying a logic "or" to each of the truncated bits detects the case of $\text{GCD}(F, N_E) = N_E$, in which case no phase truncation spurs occur. Sending the indicator to a logic "and" operation on the output of the toggle flip-flop addresses the issue. Figure 4.10 shows the necessary modification.

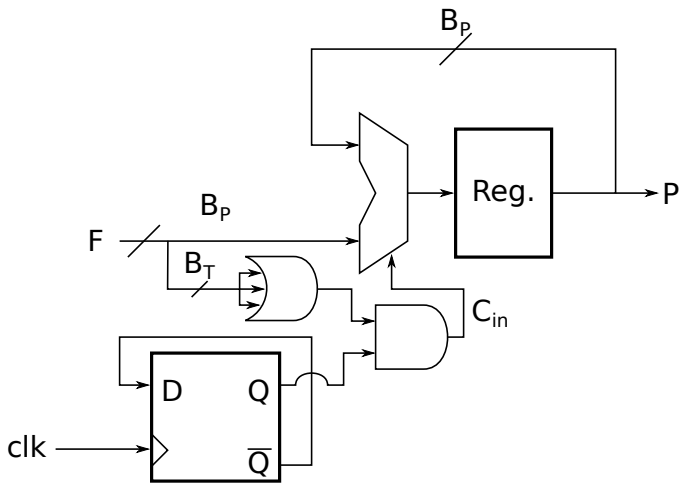


Figure 4.10: Forcing Coprime FCWs (Modification)

4.7.2 Phase Accumulator with Prime Number of States

Another technique for improving the worst case SFDR of the DCDO in the presence of phase truncation is to make the number of states in the phase accumulator a prime number. One special set of prime numbers that takes advantage of every state achievable by a B_P bit register minus one is a Mersenne Prime, which is a prime number of that form $M_P = 2^p - 1$. Table 4.1 provides four Mersenne prime number that can be easily implemented without losing many states. The implementation is also inexpensive (though not as efficient as the forced coprime FCW technique in Section 4.7.1).

Table 4.1: List of Mersenne Primes for Phase Accumulation

p	M_P
13	8191
17	131071
19	524287
31	2147483647

This technique works because every FCW is coprime to a phase accumulator with a prime number of states. The worst case SFDR for $B_P = 17$ and $B_T = 5$ is 68.325 dBc, which is computed using Equation 4.124 and converting to decibels. Now consider the

improvement by forcing $N_P = 2^{31} - 1$. The major theorems from Chapter 4 cannot be used because $N_E \nmid N_P$, but fortunately, the number of least periods analysis does work. This means that the SFDR needs to be calculated only once for all FCW (there is only one possible least period since all the FCW are prime to a prime-valued accumulator). The spectrum for $N_P = 2^{17} - 1$ and $F = 1441$ is plotted in Figure 4.11 and the worst case SFDR is found to be 72.12 dBc. This is an improvement of approximately 3.8 dB, so the same limit that existed for modification in Section 4.7.1 exists here.

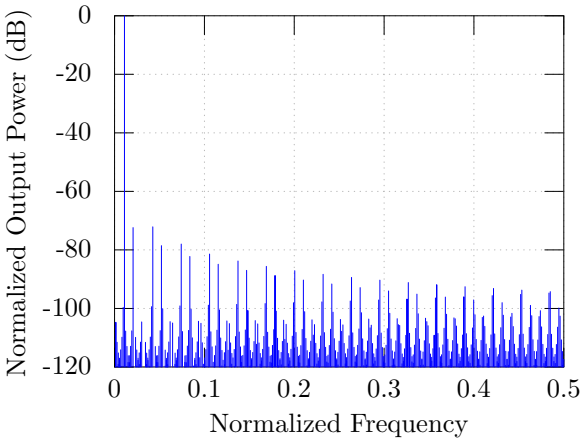


Figure 4.11: Mersenne Prime (17) Spectrum

This solution is not as elegant as forcing coprime FCWs because the spectrum is always polluted with spurs despite the 3.8 dB improvement in SFDR. With the modification shown in Figure 4.10, a direct 3.9 dB improvement is achieved when phase truncation spurs are present otherwise the spectrum is clean when there is no phase truncation.

Chapter 5

Parallelization of Phase Accumulator

A core component of the DCDO is the overflowing accumulator, whose behavior has been the target of the theory presented in the preceding chapters. The accumulator generates the periodic sequence analogous to the periodic phase of a sinusoid. The mathematics surrounding the sequences generated by a phase accumulator has already been discussed in Section 1.2 and Chapter 3. In this section, the hardware implementations of phase accumulators are presented. Figure 5.1 provides a block diagram of an accumulator with LFM capabilities.

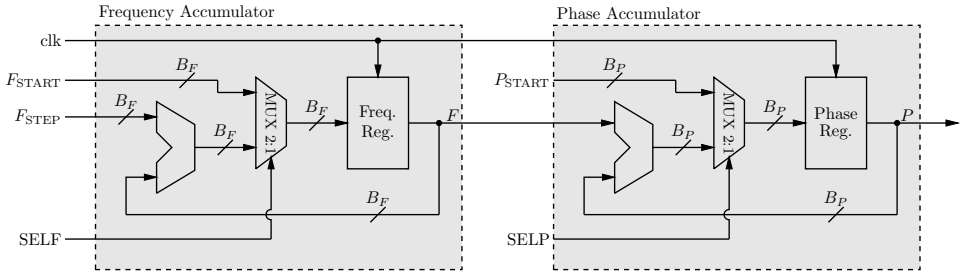


Figure 5.1: Phase Accumulator with LFM

B_F is the number of frequency control word bits, B_P is the number of phase accumulator bits, F_{START} is the starting FCW when LFM is enabled, F_{STEP} is the linear frequency increment, P_{START} is the starting phase of the accumulator, f_{clk} is the frequency at which the accumulators are clocked, F is the value of the frequency register and P is the value of the phase register. This matches the naming conventions used thus far in this document.

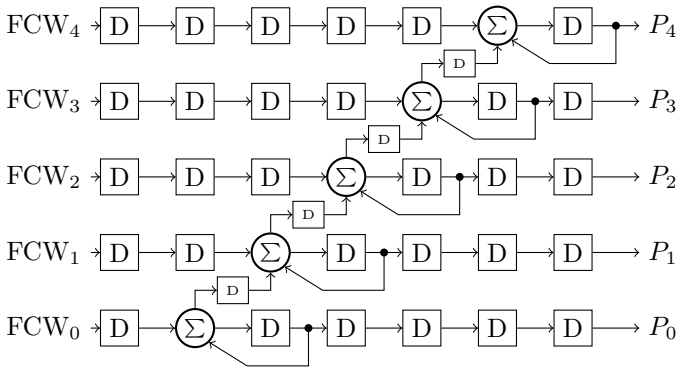


Figure 5.2: Block Diagram of Pipeline Accumulator

5.1 Pipelined Accumulator

The most common accumulator architecture chosen to address the problem of achieving high frequency operation is the pipeline accumulator [33],[34],[35],[36]. Figure 5.2 presents a conventional pipeline phase accumulator for five bits of phase accumulation. "High frequency" is a subjective term but implies operational frequencies relatively high with respect to the unity-gain band-width product (f_T) of target technology. At these frequencies static rail-to-rail CMOS fails to operate correctly.

The term pipelining refers to inserting a synchronous delay to break large combinatorial logic into smaller logical blocks with less propagation time. Each "D" in the figure represents a D-Flip-Flop (DFF) and each capital Greek Σ character is a full adder. In many cases this allows the overall clock rate of the system to increase dramatically over carry-save and carry look-ahead architectures. In the most extreme incarnation, every bit of the accumulator is pipelined, thus the longest delay between two system clocks is a full adder. Using current mode logic (CML), also called emitter coupled logic (ECL) when bipolar transistors are used or source coupled logic (SCL) when NFETs are used, extremely high frequencies can be obtained. The change to CML is not without drawbacks, since the static power consumption is nearly equivalent to the dynamic power consumption, an issue not present with rail-to-rail CMOS logic. Figure 5.2 demonstrates the traditional implementation of a pipeline phase (or frequency) accumulator using 5-bits.

A different architecture is necessary for obtaining LFM at multi-GHz clock speeds when a pipelining architecture is chosen. Figure 5.3 is a novel pipeline architecture that merges the backend DFFs of the frequency accumulator with the frontend DFFs of the phase accumulator, thus saving a significant amount of area and power. Additional savings can be made

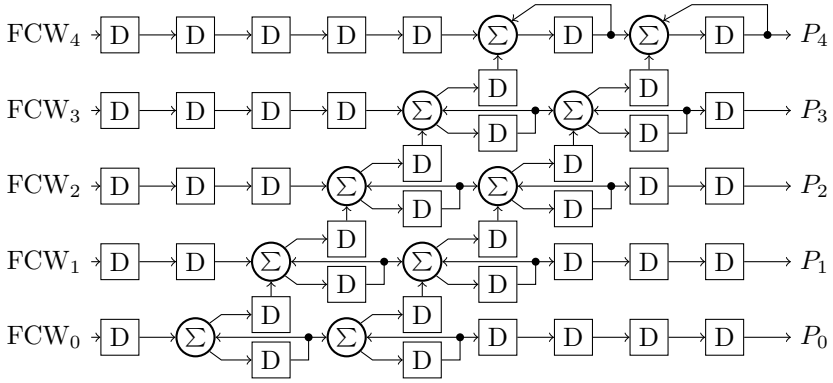


Figure 5.3: Block Diagram of Pipeline Accumulator with LFM

by removing the front registers entirely. Clearly this is possible since the FCW is generally held for many clock cycles and thus there is no need to pipeline. Using this technique causes undesirable phase jumps when changing FCWs. However, even with the power savings, the accumulator still requires a large number of DFFs. This leads to a discussion on the benefits of parallel accumulation.

5.2 Parallel Accumulator

A parallel accumulator is proposed to address the limitations encountered with the CML full pipeline accumulator. The phase accumulator is an excellent candidate for parallelization as the next phase state can be precisely predicted using a closed form equation (see Section 5.2.2). With this knowledge, the next phase state can be computed at the same time the current phase state is computed. The high speed circuitry is limited to the upconverting multiplexer logic needed to interleave the parallel computed phase values. Crafting high speed True Single Phase Clock (TSPC) logic in the upconverting multiplexer logic allows

static CMOS to carrying the data to 2-4 times a typical standard cell library in the target technology [37].

5.2.1 Prior Art

Parallelizing digital accumulation to increase the throughput of a DDFS has been proposed in a prior IEEE journal article [1] as well as prior patents [38][39][40]. In the first such patent of its type from Hassun in 1984 [38], the technique places N_P identical DCDO circuits parallel to each other and then multiplexes the output into a single DAC. In the patent by Goldberg [39], the entire calculation path from accumulator to ROM is duplicated N_P times and then multiplexed at N_P times the core digital clock frequency. Thus there are N_P accumulators, N_P ROMs, and N_P phase and frequency modulation paths. The accumulators are offset in such as way as to produce an output that when combined sequentially produces the same output as a DDFS that operates at N_P the clock frequency of the individual DDFS units.

Another interesting approach by Tan [1] uses a single phase accumulator that accumulates at a FCW four times (or more generally N_P times) that of the desired FCW. An additional set of multipliers and adders are used to predict the remaining phase values in parallel. Figure 5.4 is a block diagram of the architecture proposed by Tan. The authors of [1] claim frequency modulation support by externally varying the FCW but provide no measured results. The author of this work believes that several limitations of the architecture should be noted for the LFM case. Firstly, using an off-chip input for the frequency step word (FSW) limits the speed at which the LFM operates to the speed at which the off-chip circuitry can update the signal the digital core. This information often comes from a serial peripheral interface (SPI). Furthermore, in Subsection 5.2.2, we will demonstrate that the Tan architecture correctly supports LPM but only supports LFM at data rates lower than the clock frequency even if the FSW can change at the clock rate.

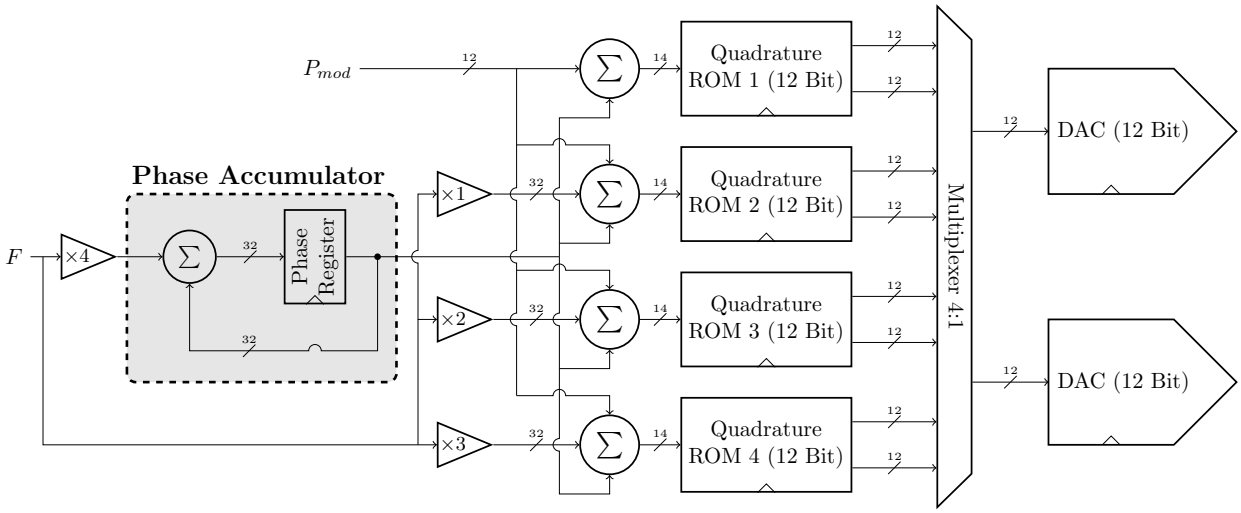


Figure 5.4: [1] Architecture

The architectures become more exciting, as well as complex, in more recent patents [40] where a finite state machine is used to initialize N_P LFM accumulators such that the output produced can be interleaved to the correct result. Figure 5.5 shows eight parallel traditional LFM components similar to the one shown in Figure 5.1. The wires controlling the initialization of the frequency and phase states are not drawn to prevent an overly busy block diagram.

The central observation of the patent is that if proper initial conditions are set for each LFM core, then the multiplexed signal from several paths is equivalent to the a single high speed LFM accumulator output. The mathematics involved in the derivation is similar to that in Section 5.2.2, and thus the derivation will be postponed for a section. The required initial values for each phase accumulator path is:

$$P_i[0] = P_{\text{START}} + iF_{\text{START}} + \frac{1}{2}F_{\text{STEP}}\left(i^2 - i\right) \tag{5.1}$$

where P_i is the phase of the ith accumulator path. Figure 5.6 shows the state machine from the patent that performs necessary computations to initialize each component. By controlling its clocking, the LFM accumulators can be initialized before running. Once the

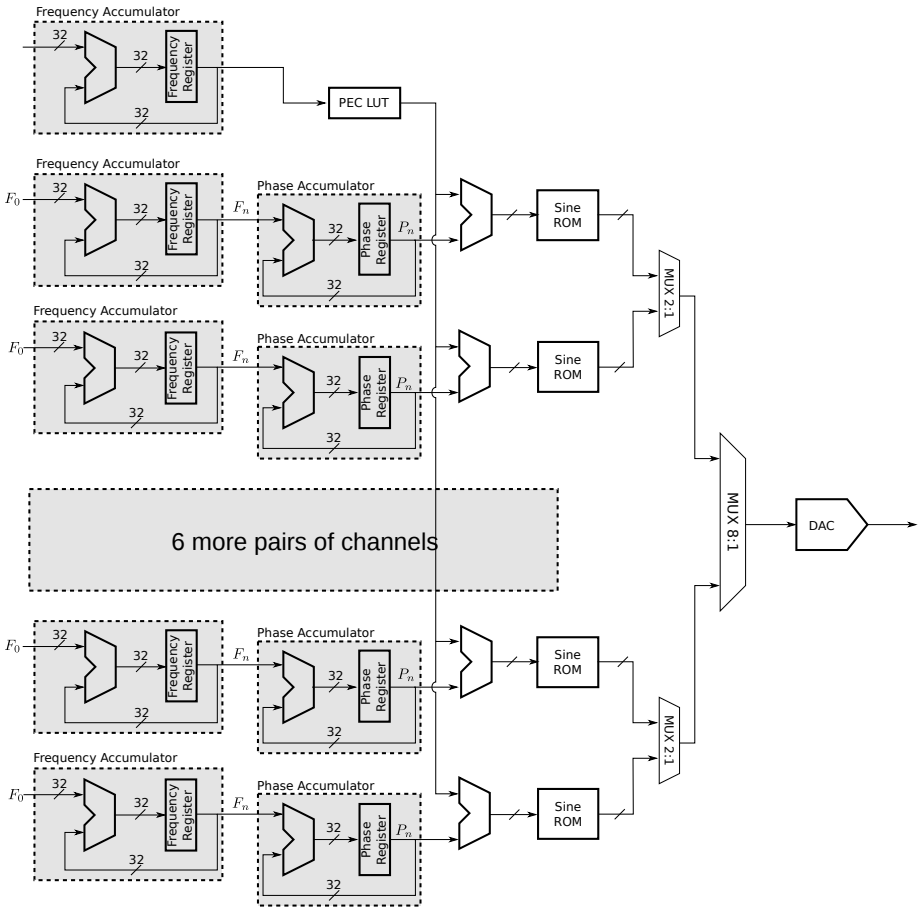


Figure 5.5: FSM Chirp-Enabled DDFS with Parallel Processing Path

relationship between the accumulators is correctly setup, the accumulators operate at the core clock frequency.

The author has developed a different, more direct, approach to parallel phase accumulation. Figure 5.7 shows a DDFS using the alternative architecture. Note that this is not remarkably different at this level from any of the other parallel accumulator implementations. The portion of the design that improves upon its predecessors and achieves novelty is the implementation of the parallel LFM phase accumulator itself.

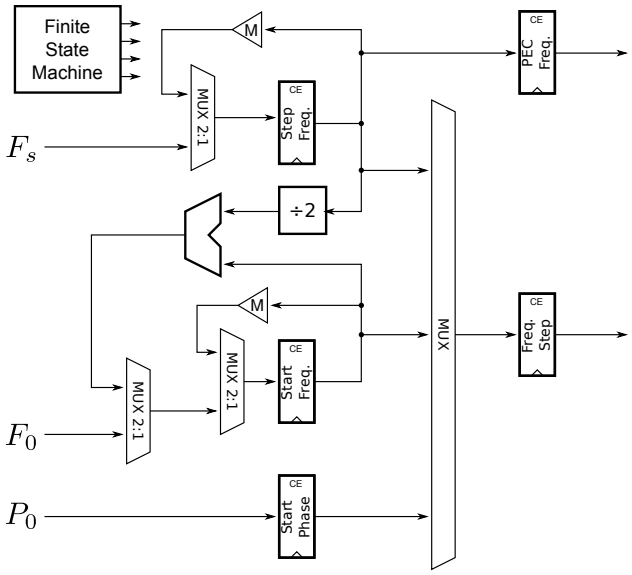


Figure 5.6: Finite State Machine for Parallel Processing Path

5.2.2 Derivation of LFM Enabled Architecture

As claimed in Section 5.2, phase accumulation is a deterministic process for which simple closed form equations can be derived. Equation 5.2 expresses the state of the phase accumulator at the n^{th} clock cycle:

$$P_n = P_{n-1} + F \tag{5.2}$$

where F is the frequency control word (FCW), or linear phase step size, given to the accumulator and P_n is the phase state at clock cycle n. Recursively evaluating Equation 5.2, we can predict the future values of the phase state at m clock cycle advanced from n.

$$\begin{aligned} P_{n+1} &= P_n + F \\ P_{n+2} &= P_{n+1} + F = P_n + 2F \\ P_{n+m} &= P_{n+m-1} + F = P_n + mF \end{aligned} \tag{5.3}$$

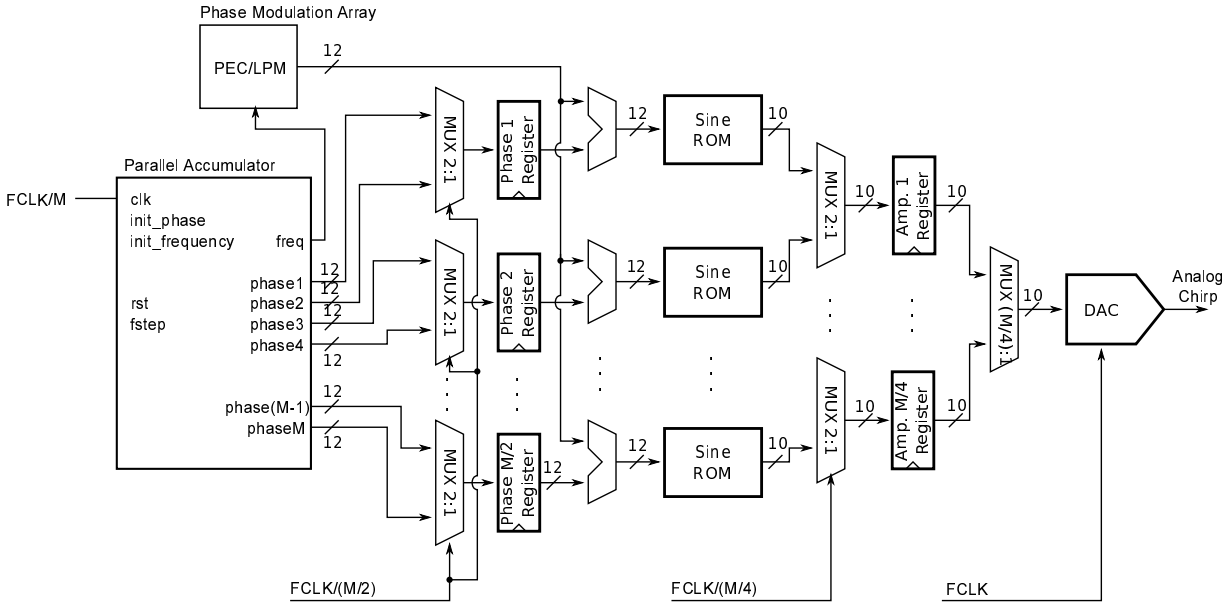


Figure 5.7: Proposed DDFS Using Novel Parallel Accumulator

The previous closed form expression gives the Tan architecture in Figure 5.4 when setting $m = 4$. Note however that F is assumed constant. This assumption must be relaxed to accurately predict the phase under LFM conditions.

Equations 5.4 and 5.5 express the state of the phase variable at the n^{th} clock cycle.

$$F_n = F_{n-1} + F_0 \tag{5.4}$$

$$P_n = P_{n-1} + F_n \tag{5.5}$$

where F_0 is the frequency step word (FSW), F_n is the FCW at clock cycle n and P_n is again the phase state at clock cycle n. Evaluating the Equation 5.5 recursively, we can predict the

future values of the phase variable.

$$P_{n+1} = P_n + F_{n+1} = P_n + F_n + F_0 \tag{5.6}$$

$$P_{n+2} = P_{n+1} + F_{n+2} = P_n + 2F_n + 3F_0 \tag{5.7}$$

$$P_{n+3} = P_{n+2} + F_{n+3} = P_n + 3F_n + 6F_0 \tag{5.8}$$

$$P_{n+m} = P_n + mF_n + \left(\frac{m^2 + m}{2}\right) F_0 \tag{5.9}$$

Likewise the future value of the frequency variable can be computed as

$$F_{n+1} = F_n + F_0 \tag{5.10}$$

$$F_{n+2} = F_{n+1} + F_0 = F_n + 2F_0 \tag{5.11}$$

$$F_{n+m} = F_{n+m-1} + F_0 = F_n + mF_0 \tag{5.12}$$

Now we can now analyze the Tan architecture of Figure 5.4 assuming that F is not a constant but instead changes over time (as required for LFM support). Note that there is not frequency prediction for F but instead it can only change at the rate of the low speed accumulator clock frequency. To achieve the correct phase value in the parallel computation, Equation 5.13 must be observed.

$$P_{n+m} = P_n + \sum_{k=1}^{m} F_{n+k} \tag{5.13}$$

Thus frequency modulation appears to have a zero order hold on signal and introduces undesirable modulation on the synthesized output.

5.2.3 Area and Power Growth Analysis

Without much effort, the number of DFFs and full adders required to implement a pipeline accumulator can be derived. Equation 5.14 expresses the number of full adders in

an accumulator with the architecture shown in Figure 5.2 as a function of the number of bits in the accumulator, B_P.

$$N_{fa}(B_P) = B_P \tag{5.14}$$

The number of DFF cells for the same architecture can also be expressed as a function of B_P.

$$N_{ff}(B_P) = 2\left[\frac{B_P^2 + B_P}{2}\right] + B_P \tag{5.15}$$

For a pipeline accumulator with on-chip LFM (Figure 5.3) the number of full adders is given by Equation 5.16.

$$N_{fa}(B_F, B_P) = B_P + B_F \tag{5.16}$$

Likewise, the number of DFF cell also becomes a function of B_P and B_F as shown in Equation 5.17

$$N_{ff}(B_F, B_P) = 2\left[\frac{B_P^2 + B_P}{2}\right] + 2B_F + B_P \tag{5.17}$$

Note that both solutions grow quadratically with the number of bits in the phase accumulator. Assuming a modest 3 mW per DFF, which is a reasonable assumption for an SiGe HBT CML architecture operating at the desired clock frequency, the accumulator alone will consume more than 3 W of power.

Table 5.1: Comparison of Accumulators

	Process	Architecture	V_{DD}	f_{clk} (GHz)	Power	Bits
[1]	CMOS	Parallel	5 V	0.8	N/A	32
[35]	InP	Full Pipeline	N/A	9.2	N/A	8
[41]	InP	Ripple Carry	N/A	13	2.13 W	8
[34]	InP	Full Pipeline	3.6 V	32	4.9 W	8
[36]	BiCMOS	Full Pipeline	3.3 V	6.2	0.825 W	9
[33]	BiCMOS	Full Pipeline	3.3 V	8.6	N/A	11
[42]	BiCMOS	Ripple Carry	3.3 V	5	N/A	24
This Work	BiCMOS	Parallel	1.2V	6.4	0.3 W	32

5.2.4 Hardware Implementation

The equations from Section 5.2.2 have been realized in HDL. The chip never made it out for fabrication, but the simulations appear to work correctly. Figure 5.8 shows the frequency and phase predictive steps. It may not be immediately apparent why this methodology is an improvement over [40].

1. No finite state machine logic required. This reduces logical complexity in the implementation.

2. No initialization sequence required when changing the chirp rate. This reduces latency between generating signals.

3. Extra frequency and phase modulation only needs to be inserted at the frequency and phase accumulators. Instead of the 8 adders required in Figure 5.5, only a single additional adder is required. The diagram in Figure 5.7 also indicates that using the 8 adders external to the parallel phase accumulator also works.

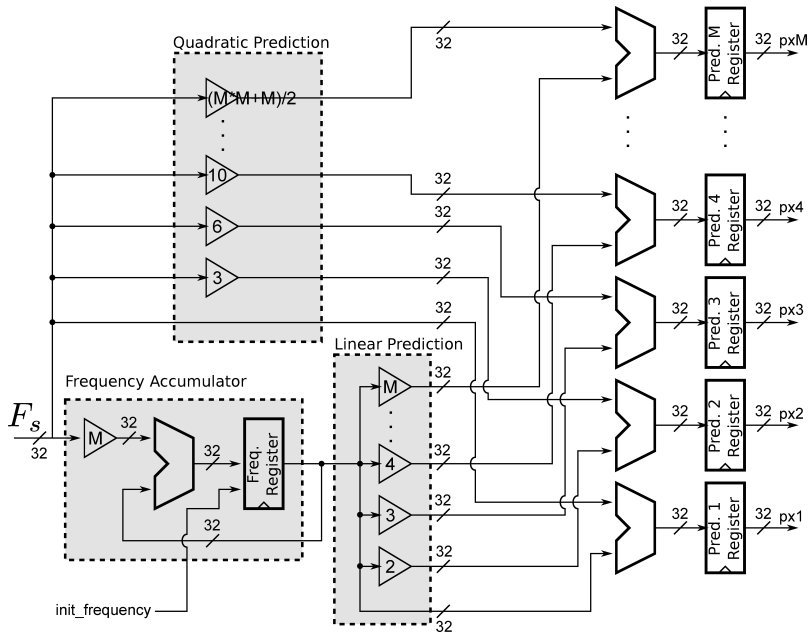


Figure 5.8: Frequency and Phase Predictive Step

The proposed architecture has an apparent drawback in that multiplications are required. These are *constant* multiplications, and thus can be implemented efficiently as shift and add circuits. Any multiplication by a power of two is simply a bit shift to the left, which means the operation is completely free. If the chirp rate does not need to change in one fast clock cycle, and most likely it does not, then the multipliers can be computed at a much slower rate (though this influences FSW switching latency). Figure 5.9 shows how the signals from the predictive circuitry are fed into the phase accumulator portion of the design.

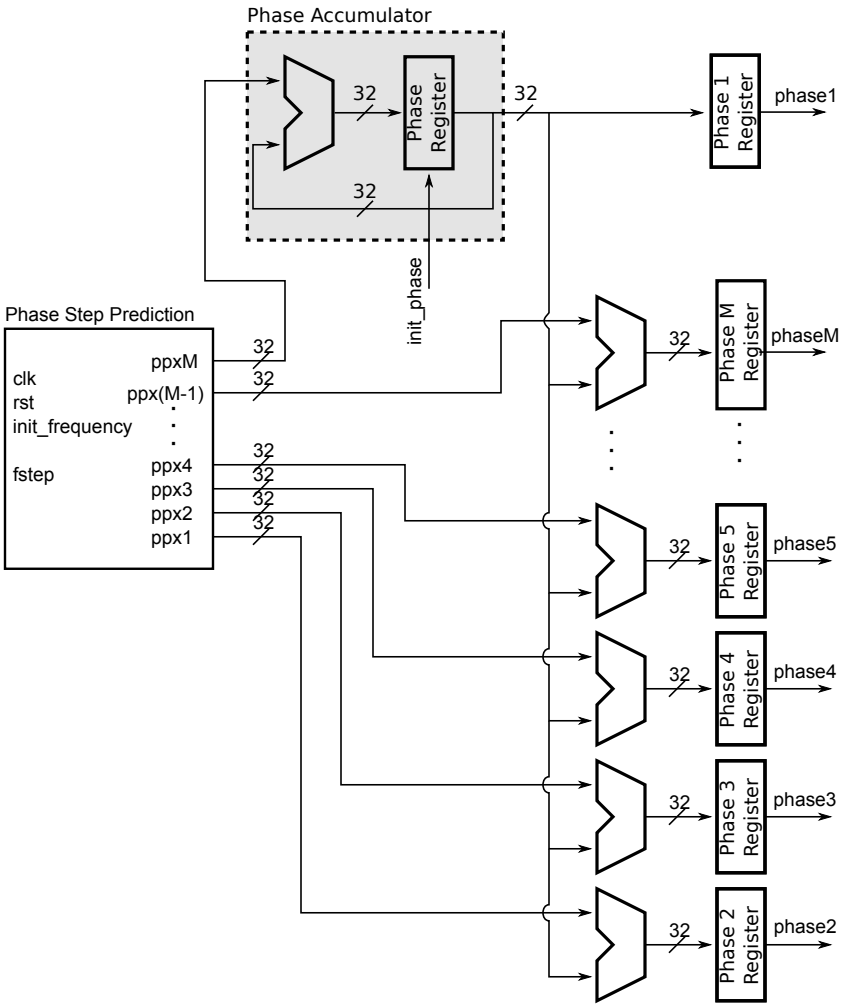


Figure 5.9: Parallel Phase Accumulator using Predictive Step

5.3 Multiplexer Upconversion Analysis

The four-to-one multiplexer shown at the output of Figure 5.4 is a two stage synchronous parallel to sequential data converter. Similar data multiplexing structures can be found in commercial DACs [43]. This is because the state-of-the-art FPGAs from Xilinx (the Virtex 7 at the time of this writing) and similarly Altera cannot output a data stream at the data rate of modern high-speed DACs. Figure 5.10 shows a symbolic diagram of the upconverting multiplexer.

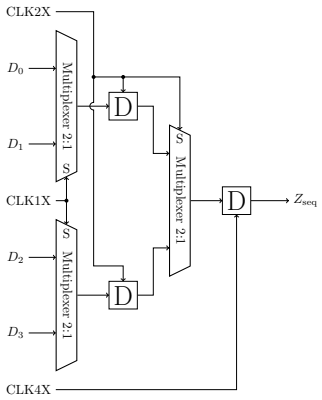


Figure 5.10: 4-to-1 Upconverting Multiplexer

Consider an N parallel data paths of B_A bits each. Let the clock at the lowest level be f_{c0}. Let the clock of the DAC be f_{cM}. Generally, the clock rate doubles and bus width halves at each upconverting multiplexer stage. Let f_{cml} be frequency at which a technology must switch from CMOS rail to rail logic cells to CML cells.

There are five unit cells in the design, ignoring the clock distribution tree, that are required to construct the multiplexer tree.

- CMOS DFF for lower speed synchronization.
- CMOS Multiplexer for lower speed data multiplexing.
- CMOS to CML Converter for taking the CMOS to CML for higher speed operation.
- CML DFF for higher speed synchronization.

- CML Multiplexer for higher speed data multiplexing.

The CMOS standard cell libraries provided vendors, such as IBM or ARM, do not operate at the maximum achievable operating frequency for CMOS rail-to-rail logic obtainable in a given process. Custom CMOS design may be used to obtain an extra stage of upconversion before switching to CML. Since the bus width is halved at each stage, the number of required stages can be calculated by

$$N_S = \log_2(N) \tag{5.18}$$

where N_S is the number of upconversion stages and N is the number of parallel data paths. Using Figure 5.10, it clearly follows that the number of multiplexer cell is

$$N_{\text{mux}} = \sum_{n=0}^{N_S-1} 2^n B_A \tag{5.19}$$

where N_{mux} is the number of multiplexer cells. Each multiplexer of Figure 5.10 is composed of B_A multiplexer cells. Similarly the number of DFFs is

$$N_{\text{dff}} = \sum_{n=0}^{N_S-1} 2^n B_A \tag{5.20}$$

where N_{dff} is the number of DFFs required in the multiplexer tree. It may be possible to eliminate some of the DFFs is algorithms to control the multiplexing and gating logic over temperature and process variation are implement. But for the purposes of this analysis, DFFs are assumed to capture the data at each new upconverted data rate. The frequency at the n^{th} stage is

$$f_{cn} = 2^n f_{c0} \tag{5.21}$$

At the stage where $f_{cn} < f_{\text{cml}}$, the CMOS to CML converter must convert the signal from a rail-to-rail operating voltage to CML voltage levels. Let this conversion happen at stage S_C. The conversion operation in some instances can be integrated into a hybrid CML

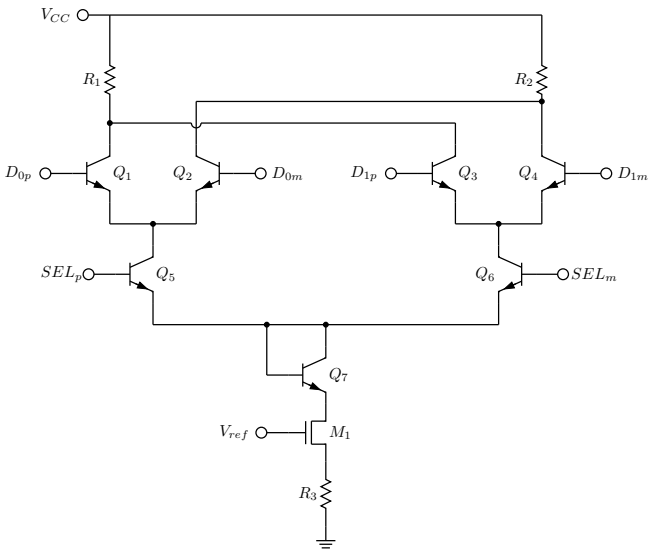


Figure 5.11: CML Multiplexer

multiplexer cell. Figure 5.11 shows a traditional CML multiplexer built from HBTs. The CMOS used in the current source helps reduce the required voltage supply, giving the design a 10 to 20 percent power savings.

Several publications report a threshold at which CM4L outpaces CMOS performance in terms of power and operating frequency [7]. This should be independently investigated for the target technology for proper optimization of the upconverting multiplexer tree. Some analysis in literature is naive in that the static power draw of CML is not incorporated into the power analysis. Low activity bits may benefit from CMOS rail-to-rail implementations over CML at a given frequency if the dynamic power consumption is somewhat higher. Furthermore, CML generally requires a higher supply rail and additional passive resistors that can be large in comparison to the active devices.

5.4 Behavioral HDL Synthesis

One of the contributions of this thesis is an improved technique for dynamically generating synthesizable HDL code and the corresponding test structures required to verify the synthesized result. These techniques flowed from previous work as a Computer Science undergraduate student at Louisiana State University (LSU).

5.4.1 Problems with Existing Techniques

A common approach to realizing reconfigurable HDL designs is nesting hundreds of `printf` or `echo` statements in programming language control structures [44]. This technique, whose commonality flows from the obviousness of its implementation, is error prone and inefficient in several ways. Firstly, the target language code is not properly highlighted during design since it is nested between quotation marks and parentheses and preceded by source language functions. The target language design tools are therefore not utilized during the design of the module. In contrast, templating allows syntax highlighting to function properly *while designing the module* because a majority of the template is written in the target language. With `printf` the code is placed in strings and is likely highlighted as if the code is for the *source* language. The difference here should not be understated. Proper spacing, syntax highlighting and other integrated development environment (IDE) features foster a more friendly, productive design environment.

Secondly, since every line of code is written in a `printf` statement, designers often forego proper code comments. Should a new designer take over the work or an independent design reviewer engaged, he will find a massive body of strings written in some source language with little idea of what is actually happening. The original author might choose to write the comments in the source language to address this issue, but the produced code may still not have sufficient documentation. Since the template language is written in the target language, comments flow as naturally as when coding entirely in that target language. In fact, the

comments themselves can take advantage of the templating system and produce dynamic content based on template settings.

The last problem highlighted here is failure to properly separate design concerns. With the `printf` approach, the target language code is written in the source language formatting and oftentimes in the same file. There are many difficulties that arise from this scenario. Version control becomes problematic, as any change to the structure of the target language design create a modification to the source language program that generates the code. With templating, the template and the code that generates the values used for the template are tracked effectively and correctly as separate entities. Reusability also takes a significant hit when using embedding the target language within the source language. While carefully creating libraries of `printf` type statements can arguably accomplish reusability similar to using a template, the reuability in the template solution is inherently accomplished. Templates are themselves practically defined as reusable code objects.

5.4.2 A Simple Example

The following code creates a combinatorial block that implements the `erfc` function assuming the input and output integers are normalized values between zero and one. The setup uses Python with the `numpy` [45] package to provide array math functionality similar to MATLAB and the Python `mako` [46] package provides the template engine. The first step is to write a template for handling combinatorial cases in Verilog. Listing 5.1 shows the template required for the example.

Listing 5.1: Combinatorial Logic Template (comb.v)

```
module ${name}(address, value);
        /* Parameters */
        parameter integer BX = ${BX};
        parameter integer BY = ${BY};

        /* Declare input ports */
        input wire [BX-1:0] address;
```

```
        /* Declare output ports */
        output reg [BY-1:0] value;

        always@(address) begin
        case(address):
% for i in range( len(values) ):
${address[i]}: value <= ${values[i]};
% endfor
        endcase
        end
endmodule
```

The template requires a variable that specifies the name of the module (`name`), one that holds the number of bits in the x word (`BX`), one that holds the number of bits in the $y = \text{erfc}(x)$ word (`BY`), an array that holds the possible combinatorial logic addresses (`address`) and an array that holds the `erfc` values (`values`). Anything with a dollar sign ($) preceding a word enclosed in curly brackets is evaluated to a string and replaced during template evaluation. The '%' starts a templating construct. In this instance a `for` loop is used to populate the Verilog `case` statement by traversing through all the address and `erfc` values. Note that any Verilog combinatorial lookup table can be generated by using the `comb.v` template.

Next the logic for generating addresses and values is written in the source language. The first part of Listing 5.2 shows the necessary code for calculating the `erfc` function in Python. The second part of Listing 5.2 demonstrates how to evaluate the template and generate the Verilog code.

Listing 5.2: Template Evaluation Using Python

```
# Import modules
from mako.template import Template
from numpy import linspace, floor, array
from scipy.special import erfc

# Perform the erfc calculations
BX = 8
BY = 12
x = linspace(0, 1, 2**BX, endpoint=False)
y = erfc(x)

# Convert to integer values
```

```
values = array(floor(y * 2**BY - 1),
        dtype=int)
address = array(x * 2**BX, dtype=int)

# Evaluate the template
name = 'CombErfc'
print Template(filename='comb.v').render(
        name='CombErfc',
        BX=BX,
        BY=BY,
        address=address,
        values=values
)
```

A fragment of the resulting output is provided in Listing 5.3. Obviously this particular examples is easily realized using `printf`, though perhaps not quite as elegantly. However, Section 6.5.1 shows the results of templating used to produce code for an inverse sinc FIR filter.

Listing 5.3: Template Output

```
module CombErfc(address, value);
    /* Parameters */
    parameter integer BX = 8;
    parameter integer BY = 12;

    /* Declare input ports */
    input wire [BX-1:0] address;

    /* Declare output ports */
    output reg [BY-1:0] value;

    always@(address) begin
    case(address):
    0: value <= 4095;
    1: value <= 4076;
    2: value <= 4058;
    3: value <= 4040;
    4: value <= 4022;
    5: value <= 4004;
    6: value <= 3986;
```

5.4.3 EDA Scripts

Successful implementations of complex System-on-Chip (SoC) designs require rigorous design methodologies and tool flows. Oftentimes circumstances, whether cost driven or foundry driven or feature drive, dictate that tools from different vendors be used at various stages of the design process. As an example, the digital simulation and verification may be done using Mentor Graphic's Questa, the digital RTL synthesis may be done by Synopsys' Design Compiler, analog design may be performed in Cadence's Virtuoso tool suite and Mentor's Calibre might be used for physical verification. Even when using the same vendor for the entire tool flow, integration between tools may be lacking.

Generally, the glue between stages of the design process are command scripts. These might be written in Skill, TCL, Scheme or any number of other languages and they act as bridges to prepare the export artefacts of one tool for insertion into the next. As an example, the GDSII of digital marco generated by digital synthesis may need to be imported into an custom analog integrated circuit suite for integration. Automating the bridging code is important in reducing the risk of making mistakes by having a person manually edit these files for each design entity that must pass through the flow. This also provides tractability and reproducibility of results.

Listing 5.4 is an excerpt from a file required by Cadence's `ihdl` command for importing a Verilog netlist as a schematic.

Listing 5.4: ihdl Import Example

```
-- The target library
dest_sch_lib := ${libname}

-- Reference libraries (where to search to
-- resolve external references)
ref_lib_list := ${ref_lib_list}

-- Do not overwrite a module with the same
-- name
import_if_exists := ${import_override}

-- Makes a symbol for the module
```

```
import_cells := ${import_cells}
```

The template generation code can be in the same file that generated the HDL originally. Instead of remembering hundreds of commands and language syntaxes, the instructions on how to perform a task can be looked up one time and placed into a template. The only required language experience then is the source language, which in this case is Python.

Another important feature to note is that the templating actually provides a layer of abstraction. If a vendor changes the syntax of a command file, the only change necessary is a new or modified template. The source language implementation for generating the bridging command code does not change.

5.4.4 Optimization

Optimization is another area that text templating proves itself as a useful tool. Languages such Cadence's `spectre` provide some degree of parametrization in designs. Verilog also continues to improve its generation and parametrization constructs. In most cases however, the parameters in the language allow for minor changes in the structure of the design, such as the width and length of a MOSFET. With templating, one could just as easily sweep through different MOSFET models as the parameters of a single model. The versatility of text templating allows for the introduction of interesting optimization algorithms, such as mixed-integer optimization.

Yet another benefit of driving optimization through a general purpose programming environment is the capability of adding new optimization algorithms. Some vendors hide optimization features behind expensive licenses, and even then, the designer is limited to the algorithms chosen by the vendor. With templating, one could use MATLAB's optimization toolbox to drive an optimization, making adjustments of design variables through text templating.

Chapter 6

Radar Application

Although DDFS is used in a wide array of applications, one application that is particularly well suited for the technique is radar. Firstly, complex waveforms can be directly generated in the digital domain. DDFS allows one to quickly respond to environmental changes by adjusting the waveform to different frequencies. The DDFS is trivially phase continuous and with some simple modifications can be made phase coherent. This chapter demonstrates how DDFS was incorporated in designs at Auburn University for radar and BIST applications.

6.1 Previous DDFS Designs

The original target application for a highly efficient DDFS architecture was an analog built-in self-test (BIST) system, described in detail by Qin in his dissertation [13]. Briefly summarized, the BIST circuit generated two sinusoidal test tones, summed them together and sent the signal to a DAC. The analog signal is then mixed to the frequency of the devices that requires self-healing and then allowed to propagate through an ADC receiver path. A third, high fidelity DCDO is used to extract in-phase and quadrature phase at a certain frequency from the samples of the ADC. The extracted information is then processed through an algorithm to determine the magnitude and phase of the signal at that frequency. This allows for the linearity measurements, spur searching and various other techniques.

6.1.1 Sine Wave Symmetry

All of the MTM, BTM and CORDIC SCMF analysis compyte only one quarter of the sinusoid that is to be synthesized. Sinusoidal functions naturally have a convenient symmetry

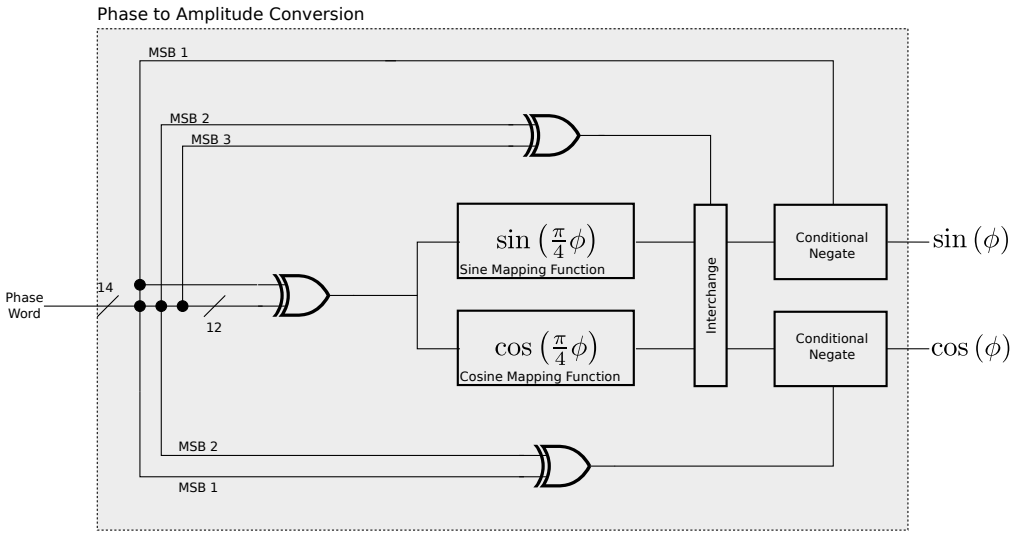


Figure 6.1: Quadrature, Quarter Sine Compression

that allows one to compress the size of the ROM in a lossless manner. Figure 6.1 shows the logic necessary to implement quarter sine compression in the quadrature case. Many authors use a 1's complement technique to invert the phase, which is also used in Figure 6.1. The 1's complement technique was dicussed in more detail in Section 1.1.2 and Figure 1.2.

Mathematically it is also easy to demonstrate the quarter wave symmetry using trigonometric identities. Equations 6.1 and 6.2 show how the amplitude of the sine and cosine are related to the sign of the phase. These alone allow for a half-wave sinusoidal compression. Adding Equations 6.3 and 6.4 to the mix allow the half-wave to be reduced by another factor of two, yielding quarter-wave sinusoidal compression.

$$\cos(\theta) = \cos(-\theta) \tag{6.1}$$

$$\sin(\theta) = -\sin(-\theta) \tag{6.2}$$

$$\cos\left(\frac{\pi}{2} - \theta\right) = \sin(\theta) \tag{6.3}$$

$$\sin\left(\frac{\pi}{2} - \theta\right) = \cos(\theta) \tag{6.4}$$

6.1.2 MTM DDFS

The first DDFS design by the author at Auburn University was conceived as an experiment to push Register Transfer Level (RTL) code through synthesis and place-and-route using the Cadence digital tool flow. None of the students in the Auburn University Radio Frequency research group had used digital synthesis tools at the time, but a flow needed to be developed for upcoming designs of increasing complexity. A symmetric multipartite table method (MTM) DDFS borrowing ROM values from [47] with corrections from the published result was written in Verilog. Figure 6.2 shows the block level implementation of the MTM DDFS. The design implemented dynamic element matching (DEM) at the output of the SCMF to randomize mismatch errors in the thermometer coded portion of the DAC.

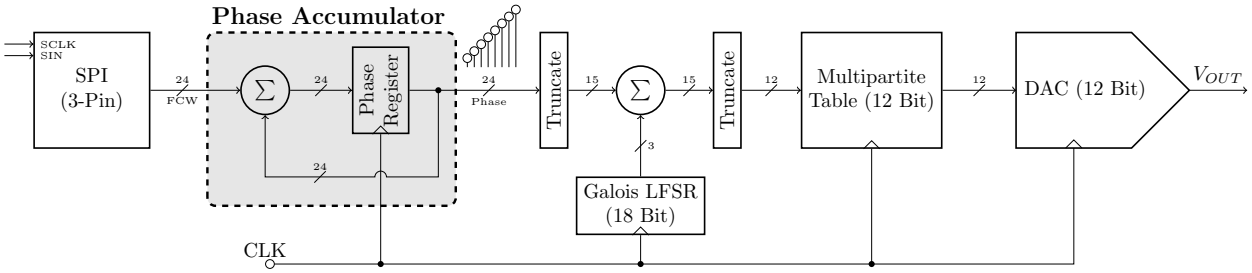


Figure 6.2: MTM DDFS Block Diagram

Table 6.1 shows the initial value table for the MTM DCDO. These values provide an accurate starting point from which smaller offset tables are used to construct the remainder of the sine. Both MTM and BTM are piece-wise linear approximations of transcendental functions and thus both require accurate initial value tables. Typically the Table of Initial Values (TIV) dominates the ROM size, as it stores the most amplitude resolution information.

Table 6.2a is one of the offset tables (TO) used in the MTM design and Table 6.2b is the second offset table used in the MTM design. Arguably the clearest explanation of multipartite table methods is supplied by [12]. To summarize, in the bipartite method

Table 6.1: Table of Initial Values

Addresses	Values
0, 1, 2, 3	12, 37, 61, 86
4, 5, 6, 7	110, 134, 158, 182
8, 9, 10, 11	206, 230, 254, 278
12, 13, 14, 15	301, 324, 347, 370
16, 17, 18, 19	393, 415, 437, 459
20, 21, 22, 23	481, 502, 523, 544
24, 25, 26, 27	564, 584, 604, 623
28, 29, 30, 31	642, 660, 679, 696
32, 33, 34, 35	714, 730, 747, 763
36, 37, 38, 39	778, 793, 808, 822
40, 41, 42, 43	836, 849, 861, 873
44, 45, 46, 47	885, 896, 906, 916
48, 49, 50, 51	926, 934, 943, 950
52, 53, 54, 55	958, 964, 970, 975
56, 57, 58, 59	980, 984, 988, 991
60, 61, 62, 63	993, 995, 996, 997

(Section 6.1.3) the offset tables compute the line segments.

$$y = m_i x \tag{6.5}$$

Here x can be decomposed into smaller n sub-components

$$x = \beta_0 x_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_{m-1} x_{n-1} \tag{6.6}$$

Plugging Equation 6.6 back into Equation 6.5

$$y = m_i \left(\beta_0 x_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_{m-1} x_{n-1}\right) \tag{6.7}$$

$$= \sum_{k=0}^{n-1} \beta_k m_i x_k \tag{6.8}$$

If the β_k values are chosen as powers of two, then the hardware multiplication operation for β_k is free. The size of x_k determines the number of entries in the TO and is exponentially

related to the value of x_k. This is clearly seen in the analysis of bipartite table in the following section. MTM therefore exchanges smaller TO tables for more additions. If the number of bits is sufficiently large, this translates into *significant* area improvement. The other downside to the the MTM technique is that quantization occurs in each table, resulting in a slight degradation in overall performance with respect to the BTM technique.

(a) Offset Table 1

Addresses	Values
0, 1, 2, 3	3, 9, 3, 9
4, 5, 6, 7	3, 9, 3, 8
8, 9, 10, 11	3, 8, 2, 7
12, 13, 14, 15	2, 7, 2, 6
16, 17, 18, 19	2, 6, 2, 5
20, 21, 22, 23	2, 5, 1, 4
24, 25, 26, 27	1, 3, 2, 0
28, 29, 30, 31	0, 1, 0, 0

(b) Offset Table 2

Addresses	Values
0, 1, 2, 3	0, 1, 1, 1
4, 5, 6, 7	2, 2, 2, 3
8, 9, 10, 11	0, 1, 1, 1
12, 13, 14, 15	2, 2, 2, 3
16, 17, 18, 19	0, 0, 1, 1
20, 21, 22, 23	1, 2, 2, 2
24, 25, 26, 27	0, 0, 1, 1
28, 29, 30, 31	1, 2, 2, 2
32, 33, 34, 35	0, 0, 1, 1
36, 37, 38, 39	1, 1, 2, 2
40, 41, 42, 43	0, 0, 0, 1
44, 45, 46, 47	1, 1, 1, 1
48, 49, 50, 51	0, 0, 0, 0
52, 53, 54, 55	0, 1, 1, 1
56, 57, 58, 59	0, 0, 0, 0
60, 61, 62, 63	0, 0, 0, 0

Figure 6.3 shows a block diagram of a two TO table MTM ROM. The diagram could be extended to any number of offset tables with little imaginative effort. B_Q are the bits of the address words used in the line calculate. B_{TIV} are the bits sent to the TIV table. B_{Q1} are the most significant q_1 bits of B_Q and B_{Q2} are the next most significant q_2 bits (or the remainder of the bits in the example). The only thing not shown explicitly in the diagram is the bit shift operation performed on the word amplitude word of the first offset table, $ATO1$. It should be multiplied by 2^{q_2}, or bit shifted that amount. In the diagram is implied as part of the "lookup" table operation.

The amount of compression of the MTM technique is difficult to quantify without implementation. In particular the size of the adders can quickly begin to dominate the area

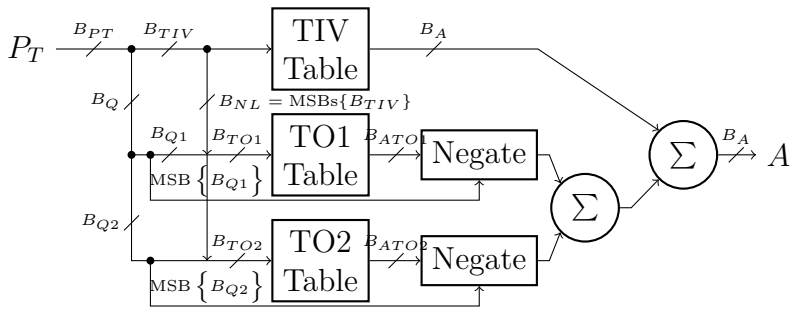


Figure 6.3: MTM Block Diagram

and power when the offset tables have a very small number of entries. Also, the extra additions may also require pipelining, which would dramatically impact the effective compression boost. The authors of [47] and [12] both showed better than 2X overall compression improvement in the implementations over published symmetric BTM ROMs.

Figure 6.4 shows a portion of the GDSII submitted for fabrication. The design was realized in a 0.130 μm BiCMOS process. The rectangular circuit on the left is the MTM DCDO and the rectangular circuit on the right is a 12-bit DAC with DEM. In the design, the DEM logic had a mistake that caused the design to operate poorly, 45 dBc SFDR Nyquist with 1 GHz clock.

6.1.3 BTM DDFS

The second DDFS design, and the one used for the first realization of the ORA BIST system, was a symmetric bipartite table method (BTM) DDFS [48]. The BTM technique is first described with respect to DDFS designs in [11] with the first observation that taking advantage of the line symmetry in hardware for additional area, speed and power savings reported in [48]. The BTM is mathematically a piece-wise linear approximation technique and hence can be applied in the approximation of any function. In this work, elementary transcendental functions such as sine and cosine are the targeted functions. Figure 6.5 shows the DDFS system that used the BTM ROM for sine.

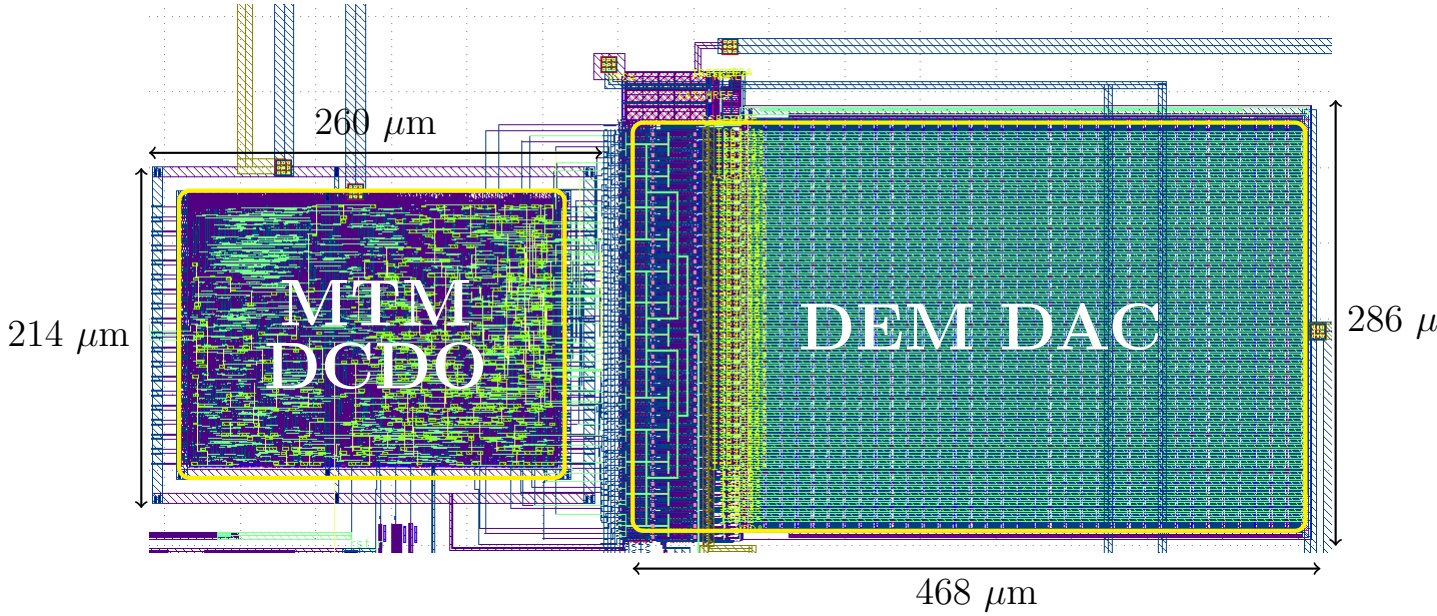


Figure 6.4: MTM DDFS GDSII (130 μm BiCMOS)

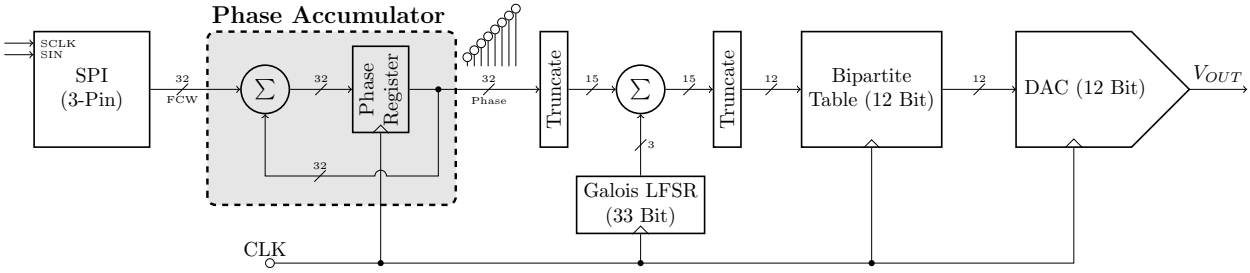


Figure 6.5: BTM DDFS Block Diagram

The compression methodology is most easily described through an example. Assume that a quarter sine ROM with $D = 8$ address bits needs to be compressed. Then the number of ROM entries is $N_D = 2^8 = 256$. The blue dashed line in Figure 6.6a shows the ideal values for the sine function plotted from 0 to $\pi/2$. For the moment, no amplitude quantization is applied in the analysis. In a *conventional* piece-wise linear (PWL) approximation of the sine function, the function is split into N lines. Equation 6.9 provides a PWL approximation of

the function $y(x)$.

$$\hat{y}(x) = \begin{cases} m_0 x + b_0, & a_0 \leq x < a_1 \\ m_1 x + b_1, & a_1 \leq x < a_2 \\ & \cdots \\ m_{N-1} x + b_{N-1} & a_{N-1} \leq x < a_N \end{cases} \tag{6.9}$$

The values of m_i and b_i are generally chosen to either minimize mean of the square of the error, which is commonly referred to as the mean squared error, between a line and the function or to minimize the maximum error. m_i is the slope of the approximating line and b_i is the y-intercept of the line. a_i forms the starting and ending points of the domain of the line segment. The mean squared error is generally minimized using a first order linear least squares regression fit on the function over that interval. The mean squared error between two continuous functions is given by Equation 6.10.

$$\text{MSE}_c = \frac{1}{a_{i+1} - a_i} \int_{a_i}^{a_{i+1}} \left(\hat{y}_i(x) - y_i(x)\right)^2 dx \tag{6.10}$$

Since the DCDO is actually approximating a discrete number of points, one can minimize the discrete mean squared error. This is performed trivially using a computer, as most math software, including open source software such as Octave [49], provide a method for applying a linear regression fit to data. The discrete mean squared error is shown in Equation 6.11.

$$\text{MSE}_d = \frac{1}{n} \sum_{k=0}^{N_D/N-1} \left(\hat{y}_i(x_k) - y_i(x_k)\right)^2 \tag{6.11}$$

Let $r = N_D/N$ be the number of points in a line segment. For the quarter sine approximation, $x_k = \frac{k\pi}{2N_D}$. For the first line between a_0 and a_1 the values of k range from 0 to $r-1$. To find

$\hat{y}_i$, the following over-determined system of equations must be solved:

$$m_0 x_0 + b_0 = y_i(x_0)$$
$$m_0 x_1 + b_0 = y_i(x_1)$$
$$\cdots$$
$$m_0 x_{r-1} + b_0 = y_i(x_{r-1})$$

This can be written in matrix notation as

$$\begin{bmatrix} x_0 & 1 \\ x_1 & 1 \\ \cdots & \cdots \\ x_{r-1} & 1 \end{bmatrix} \begin{bmatrix} m_0 \\ b_0 \end{bmatrix} = \begin{bmatrix} y_i(x_0) \\ y_i(x_1) \\ \cdots \\ y_i(x_{r-1}) \end{bmatrix} \Longrightarrow \mathbf{X}_0 \begin{bmatrix} m_0 \\ b_0 \end{bmatrix} = \mathbf{Y}_0 \tag{6.12}$$

The linear least squares best solution to the problem is [50]

$$\begin{bmatrix} m_0 \\ b_0 \end{bmatrix} = \left(\mathbf{X}_0^{\mathbf{T}} \mathbf{X}_0\right)^{-1} \mathbf{X}_0^{\mathbf{T}} \mathbf{Y}_0 \tag{6.13}$$

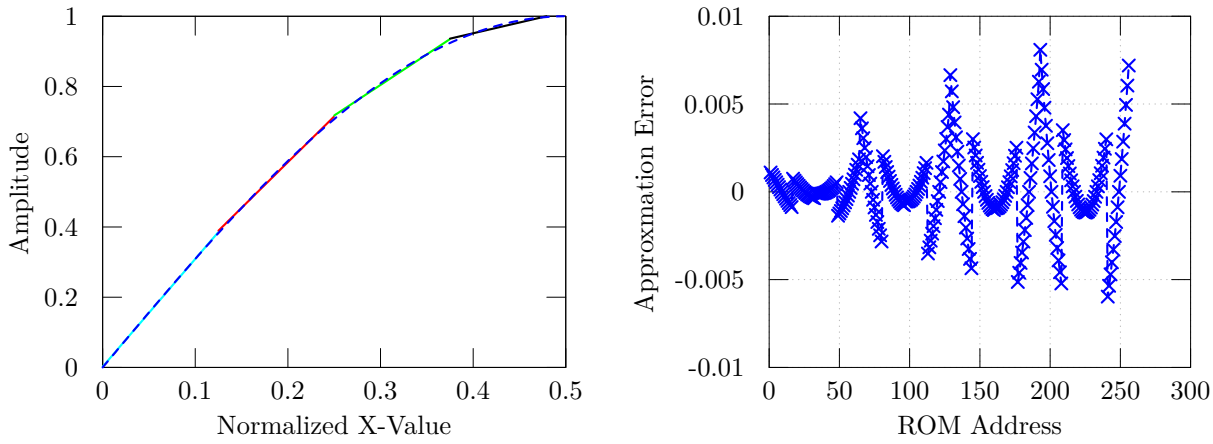
The bold characters are used to represent matrices, with the bold superscript "T" meaning the transpose operation. Performing the matrix product yields

$$b_0 = \frac{\sum_{k=0}^{r-1} y_i(x_k) \sum_{k=0}^{r-1} x_k^2 - \sum_{k=0}^{r-1} x_k \sum_{k=0}^{r-1} x_k y_i(x_k)}{r \sum_{k=0}^{r-1} x_k^2 - \left(\sum_{k=0}^{r-1} x_k\right)^2} \tag{6.14}$$

$$m_0 = \frac{r \left(\sum_{k=0}^{r-1} x_k y_i(x_k)\right) - \sum_{k=0}^{r-1} x_k \sum_{k=0}^{r-1} y_i(x_k)}{r \sum_{k=0}^{r-1} x_k^2 - \left(\sum_{k=0}^{r-1} x_k\right)^2} \tag{6.15}$$

The adjacent best fit line is calculated by changing the summations from 0 and $r-1$ to r and $2r-1$. This calculation is made for each of line segments in the approximation. To perform this operation in MATLAB or Octave, use of the built-in `polyfit` command is sufficient.

The multicolored line in Figure 6.6a shows the piece-wise linear approximation superimposed upon the targeted sine function. Each line is represented with a slope and a offset. The table of offset values is called the TO ROM in the same manner as the MTM table. Multipliers are expensive to implement in hardware, so the multiplication linear offset values are also stored in a ROM (the TO ROM).



(a) PWL Sine

(b) Approximation Error of BTM

Figure 6.6: Phase Accumulator State Plots

The conventional PWL implementation offers some compression over the ideal sine function. To develop concrete numbers for a comparison, assume 10-bits of amplitude resolution in the ROM. An uncompressed quarter sine ROM would required $2^8 \cdot 10 = 2560$ bits to represent the implementation. Assume 4 lines are used to represent the function. The initial value table has one entry for each line in this approximation. Thus there are $2^2 \cdot 10 + 2^7 \cdot 9 = 1192$ (see Equation 6.16). This is only a modest amount of compression. The important observation of the BTM technique is that one line segment is oftentimes good enough for several initial value points.

The number of bits required for the ROM is

$$2^{B_{TIV}} \cdot B_A + 2^{B_{TO}-1} \cdot B_{ATO} \tag{6.16}$$

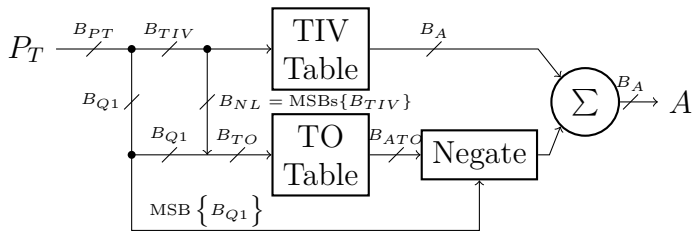


Figure 6.7: BTM ROM Block Diagram

B_A is the amplitude resolution of the ROM, B_{TIV} are the number of bits used to address the TIV table, B_{TO} are the number of bits used to address the TO table and B_{ATO} are the number of amplitude resolution bits required by the TO table. The minus 1 in the exponent of the number of TO entries comes from using line symmetry. Figure 6.7 shows a block diagram of a symmetric BTM ROM. $B_{Q1} = B_{PT} - B_{TIV}$ are the LSB of the phase word that addresses the ROM. B_{NL} are the number of different line approximations to in the offset table. The BTM approximation improves with an increasing number of initial value points and increasing number of line approximations.

The author is not aware of any direct closed form compression equations for the BTM method. Most authors performs a computer optimization over the various segmentation configurations to calculate the size. By observing Figure 6.6a it is clear that the widest range of amplitude values that must be stored in the TO table is from the first line segment. The range of the subsequent lines in the approximation is smaller (for a formal proof, observe that cosine is the derivative of sine and follow where the line . The domain of the first line segment is determined by the number of initial value points. For the quarter sine ROM, this is equal to $\frac{\pi}{2N_{TIV}}$. Using small angle approximation, $\sin(x) \approx x$. Then maximum value stores in the TO ROM is approximately $\frac{\pi}{2N_{TIV}}$. Using Figure 6.6a yet again, the maximum value of the cyan curve stops roughly at 0.4. $N_{TIV} = 4$ in that example so the expected maximum value is $\frac{\pi}{8}$ $approx 0.39$. It takes B_A bits to represent the maximum amplitude of

Table 6.2: Example BTM Compression

B_{PT}	B_{TIV}	B_{NL}	Bits	Size After Compression
8	2	2	1192	46.5%
8	3	2	592	23.1%
8	4	2	384	15.0%
8	5	2	416	16.3%
8	6	2	680	26.6%

the sine, so it takes

$$B_{ATO} = \left\lceil \log_2 \left(2^{B_A} \frac{\pi}{2N_{TIV}} \right) \right\rceil \tag{6.17}$$

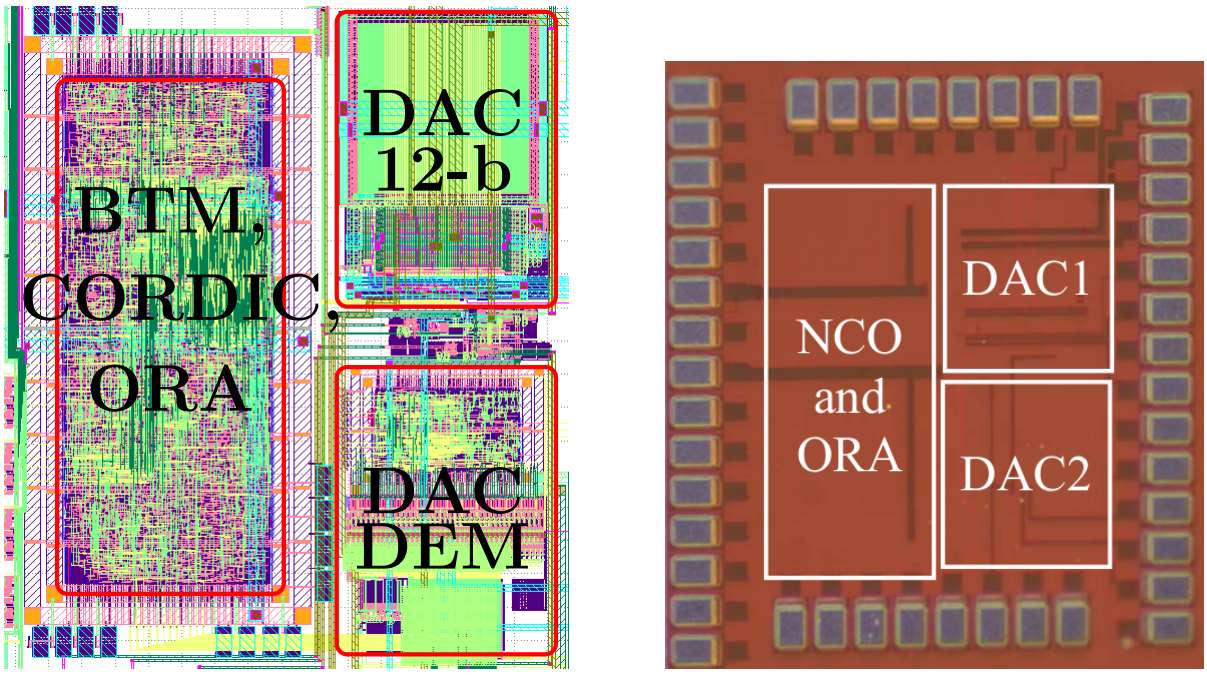
bits of amplitude resolution to represent the offsets. Then an excellent estimate of the number of bits required is easily developed:

$$2^{B_{TIV}} \cdot B_A + 2^{B_{TO}-1} \cdot \left\lceil \log_2 \left(2^{B_A} \frac{\pi}{2N_{TIV}} \right) \right\rceil \tag{6.18}$$

Note that all the variables are known at the outset of the computation, making it a direct computation. Table 6.2 shows various compression ratios with different segmentation choices.

Figure 6.6b shows the approximation error of the BTM with $B_{TIV} = 4$, $B_{NL} = 2$ and $B_{PT} = 0$. The jumps in error occur at the transition points between different line approximations. Notice the error is less than 0.1%. Depending on the quality of the DAC that follows, this error may not even show up in the output spectrum of the DDFS. Now let us observe the effect of the approximation error on the output of the spectrum. Figure 6.8a is an image of the GDSII file submitted for fabrication in a 130 μm BiCMOS process. The large block of circuitry on the left contains the BTM, ORA, SPI, control logic and CORDIC, which will discussed in more detail in Section 6.4. The two blocks to the right of the circuit are DACs. The upper one is a 12-bit CMOS DAC discussed in more detail in Section 6.6

and the lower one is an experimental DEM DAC that was refined from the MTM DDFS design in Section 6.1.2.



(a) Image of GDSII (b) Die Photo

Figure 6.8: BTM, CORIDC, ORA and DACs (130 μm BiCMOS)

One technique for reducing the spurious response that results from phase truncation is to add a small random signal to the truncated portion of the phase word and sum the overflow bit of that addition into the main phase word [51]. This technique is called additive dithering and is effective at reducing spurs generated from both phase and amplitude truncation. Figure 6.9 shows an example of a Galois linear feedback shift register (LFSR) . These allows for power and area efficient means of generating random sequences that can be used in the dithering process.

At this point, applying the theory from Chapter 4 would serve as an excellent verification of the theory and example of its application. Let the BTM have 8-bits of quarter sine address space. Thus the entire ROM, decompressed takes 10-bits of phase resolution. To keep the spectrum a manageable yet sufficiently detailed for use in verification, assume an accumulator of 12-bits ($N_P = 2^{12} = 4096$) and thus 2 bits are truncated and $N_E = 2^2 = 4$.

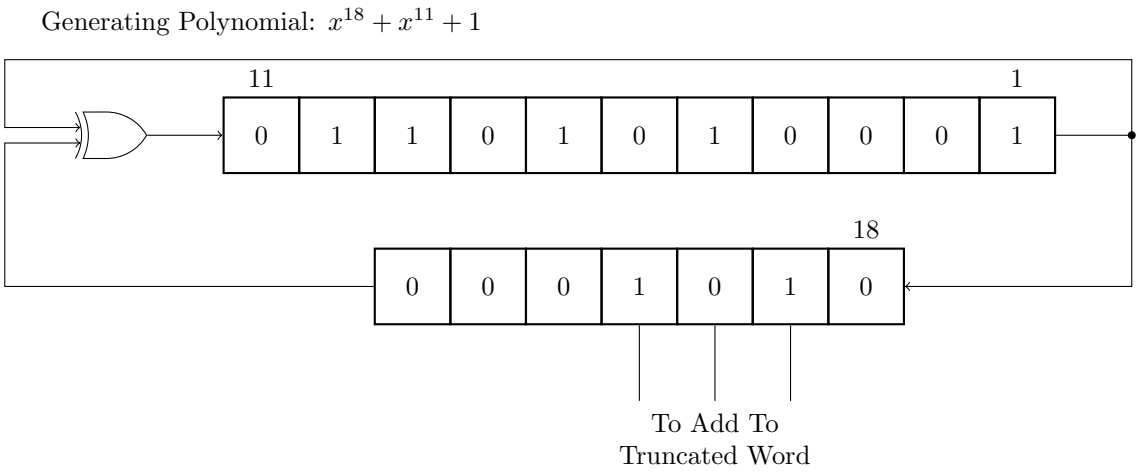


Figure 6.9: Galois 18-Bit LFSR

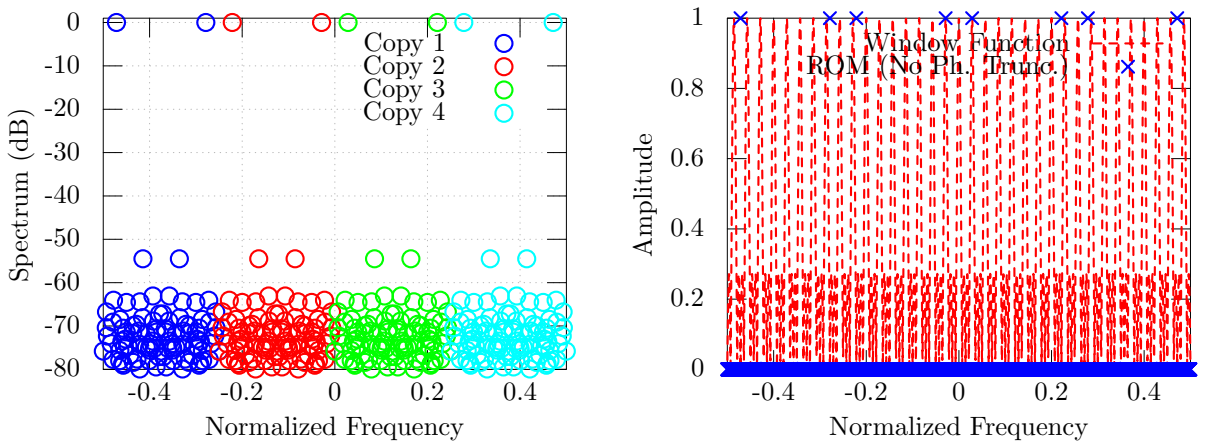
Choose $B_{NL} = 2$ and $B_{TIV} = 5$ for the BTM decomposition. Lastly let $F = 117$ (chosen completely arbitrarily other than to guarantee phase truncation).

Now apply Theorem 4.7. $\Gamma_P = 117$, $\Lambda_P = 4096$ and $\Lambda_E = 4$. The spectrum of the BTM without phase truncation driven by $\Gamma_P n$ should be copied Λ_E times then multiplied by the window function of Equation 4.48. Figure 6.10a shows the spectrum of BTM replicated Λ_E times. Figure 6.10b shows the window function plotted alongside the replicated BTM spectrum. It is not that insightful since it plotted in magnitude (and thus 50 dB down is not visually detectable).

Lastly, Figure 6.11 shows a plot of a Verilog simulation of the specified BTM against the direct computation of the spectrum through theory. This highlights the statements from Section 4.6.3 by providing an example. All of the BTM spurs are copied 4 times and hundreds of spurs are spread as a result of phase truncation.

6.1.4 Output Response Analyzer

One of the first DDFS implementations by the author was for an Output Response Analyzer (ORA) BIST circuit [21] [52]. Both the BTM and MTM designs described targetted the BIST system. Figure 6.13 shows the basic architecture of an ORA unit. The BIST system



(a) BTM Spectrum Copy

(b) BTM Spectrum with Window Function

Figure 6.10: Phase Accumulator State Plots

is designed to measure the performance of an RF receiver. Two test tones are generated using a DDFS that are then upconverted and sent into the receiver. Figure 6.12 shows the block diagram of the two tone test pattern generation circuit. The DCDO used as the test pattern generator is described in Chapter 6. The signal y is sampled by an ADC at the end of the receiver chain. The cosine and sine terms are generated by a single quadrature DCDO. The detailed derivations of ORA operation are discussed in [21], but, in summary, the system effectively computes a single DFT bin that contains the spectral energy at frequency f. The results of the derivation are shown in the following equations:

$$A(f) = \sqrt{DC_1^2 + DC_2^2} \tag{6.19}$$

$$\phi(f) = -\tan^{-1}\left(\frac{DC_2}{DC_1}\right) \tag{6.20}$$

In [21], the authors note that behavior of the ORA unit appear somewhat unpredictable and highly dependent on when the ORA accumulation is stopped. The authors originally view the "zero" crossing point of the phase accumulator as a period of the generated sinusoid. Later it was realized that the output "looked" best at an "integer multiple period" of the phase accumulator sequence. Terms such as fake integer multiple period (FIMP) and

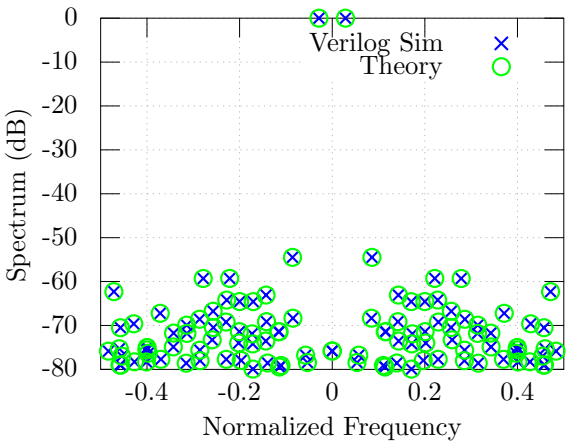


Figure 6.11: BTM Simulation Versus Prediction

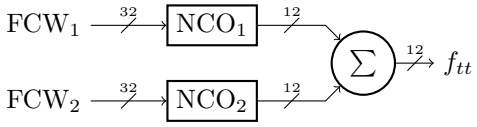


Figure 6.12: Two Tone Generation

good integer multiple period (GIMP) were used in explaining the behavior in subsequent publications.

Using the analysis from Chapter 3 and Chapter 4 a more formal description of the behavior can be specified. Firstly, the FIMP is not an integer multiple period at all nor is it related to the period of the digital waveform in any way. Thus the output at a FIMP suffers from extreme spectral leakage by stopping the accumulation before a full period of the fundamental waveform. The GIMP mentioned in the text is the least period of the phase accumulator given a f requency control word F, or

$$\text{GIMP} = \Lambda_P = \frac{N_P}{\text{GCD}(F, N_P)} \tag{6.21}$$

As discussed in [21] and [13] it is important to reduce to the ORA measurement time. Stopping on an least period provides an excellent measurement since the large power mixing tone from the DCDO does not suffer from spectral leakage. But the implementation of the

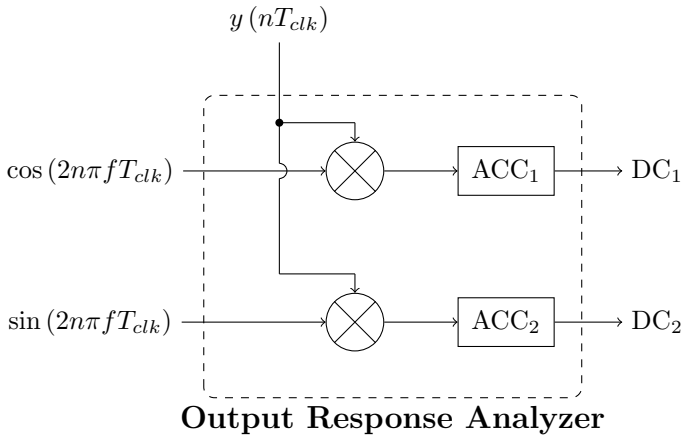


Figure 6.13: ORA Block Diagram

DCDO has a power-of-two N_P number of phase states. This implies that the test length increases by a factor of 2 for each frequency step. To address this issue a simple modification can be made to the DCDO that allows it to change N_P to an integer less than N_P but still satisfying $N_E \mid N_P$. This is equivalent to adding a programmable modulo operation to the phase accumulator output.

This allows Λ_P to be modified to the desired run length and no spectral leakage from stopping the accumulation at a non-period. This obviously changes the frequency step slightly. Let $N_P = 2^16$ and $N_E = 2^4$. Assume a 1 GHz clock frequency for the ORA and corresponding DCDO. Say want to measure a 123 MHz tone location with the ORA. The closest FCW corresponding to this tone is derived using Equation 1.45.

$$F = \frac{f_0 N_P}{f_{clk}} = 8060.9 \approx 8061 \tag{6.22}$$

But note that 8061 is coprime to N_P and thus $\Lambda_P = N_P$ and 2^{16} clock cycles are required for a measurement without spectral leakage. First note that changing $N_P = 2^{16} - 2^4$ allows one to precisely hit 123 MHz with the ORA. But one could choose a smaller N_P to reduce the measurement time and still precisely land on the frequency. Let $N_P = 2^{15} - 17 \cdot 2^4$. Then $F = 3997$ and the measurement time has been reduced by a factor of 2 and still suffers no

spectral leakage. Thus the analysis from this work was used to develop a simple modification that provides a significant amount of flexibility to the ORA BIST system.

6.2 Overview of Basic Radar Theory

The basic radar equation is generally presented in the first or second chapter of an undergraduate radar textbook [8]. While the fundamental equation appears in various forms depending which variables are chosen as function parameters for the power received, Equation 6.23 serves as a rough starting place for the discussion in this thesis.

$$P_r = \frac{P_t G_t A_e \sigma}{(4\pi)^2 R^4} \tag{6.23}$$

where P_r is the receiver power at the radar receiver, P_t is the transmitted power from the radar transmitter, A_e is the receiving antenna aperture, σ is the radar cross section of the target, G_t is the transmission antenna gain and R is the distance of the target from the radar system. This assumes that the transmitting and receiving antenna are placed in close enough proximity such that the distance that the signal travels from the radar to the target and back to the radar after reflecting from the target are approximately equal (or that the transmitting and receiving antenna are the same antenna). There are certainly cases when this assumption will not hold true, but for this work this assumption is used.

By expanding the antenna aperature term, Equation 6.23 can be written in a form [8] used in many standard textbooks.

$$P_r = \frac{P_t G_t G_r \lambda^2 \sigma}{(4\pi)^3 R^4} \tag{6.24}$$

Radar operates by using the propagation speed of an electromagnetic wave and time as a ruler for distance. A wave is radiated from the radar system and energy reflected from a target is received by the radar. Taking into account this round trip of travel, the distance

of a target from the radar antenna is given by

$$R = \frac{cT}{2} \tag{6.25}$$

where T is the time between transmission and reception of the electromagnetic wave. Oftentimes, ranges of interest are given and the equation is rearranged to calculate the necessary time interval for travel.

When multiple target detection is required, the transmitted pulse width of the radar must be reduced. The radar range resolution for traditional pulse radar is given by

$$\rho = \frac{c\tau}{2} = \frac{c}{2B} \tag{6.26}$$

where τ is the pulse period and B is the signal bandwidth. Immediately an inherent trade-off between transmitted energy and range resolution is apparent. By using pulse compression we can increase the transmitted signal power while maintaining a high bandwidth. We discuss this is a later section.

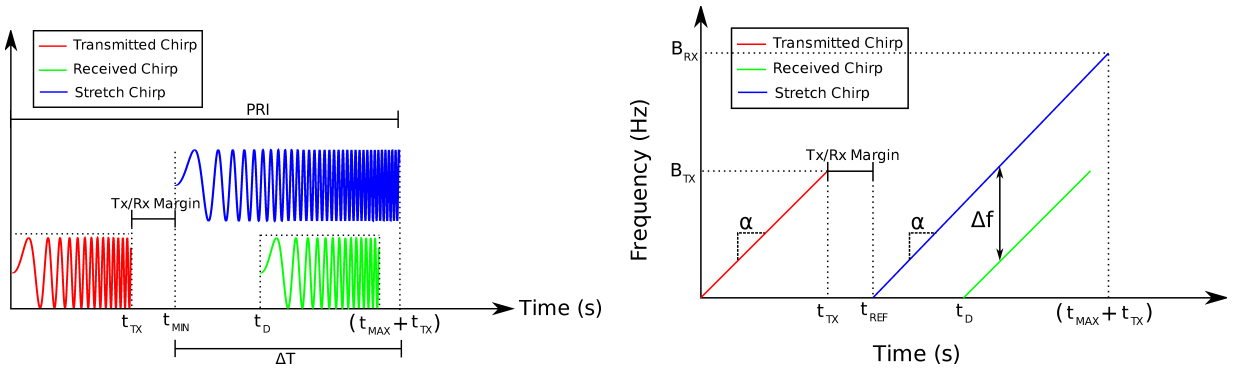
6.3 Overview of Stretch Processing

Equation 6.24 can be rearranged and solved for the distance R.

$$R = \sqrt[4]{\frac{P_t G_t G_t \lambda^2 \sigma}{(4\pi)^3 P_r}} \tag{6.27}$$

Assuming the antenna and transmitted wavelength are fixed, to double the radar range, the transmit power must be increased by a factor of $2^4 = 16$. When this is combined with the decreasing pulse width requirement for range resolution, the power requirements of a radar will quickly become unmanageable for smaller systems. Pulse compression is used to address the power, bandwidth and range issues.

Stretch processing is a pulse compression technique used in some pulse compression radar systems [8]. It combines the power benefits of having a long pulse duration with the range resolution of a high bandwidth signal. This effectively increases the range resolution of the radar without raising the bandwidth requirements of the receiver. The technique proves useful when a system needs a high range resolution but can only implement lower frequency digital signal processing hardware. Figure 6.14a shows a simplified time domain representation of stretch processing and Figure 6.14b shows the corresponding frequency domain representation of the same signal.



(a) Stretch Processing (Time)

(b) Stretch Processing (Frequency)

Figure 6.14: Example of Stretch Processing Signals

The technique works by transmitting a high bandwidth FM signal from the radar. Generally the bandwidth of this signal exceeds the capabilities of the ADC in the radar receiver chain. Upon receive, a higher bandwidth receiver chirp is mixed with the return transmit signal. The bandwidth of the receiver chirp, often called the dechirp signal, is set by the range interval (or range coverage) requirements of the radar. The following enumerated list roughly presents the mathematics of the technique:

1. A chirp signal is generated by our transmitter and has a characteristic equation of the form:

$$y_{tx}(t) = A \sin\left[2\pi f_0 t + \pi \alpha t^2\right] \tag{6.28}$$

where f_0 is the starting frequency of the waveform, α is the frequency slope and t is the time.

2. The signal received from a target has the following characteristic equation:

$$y_{rx}(t) = B\sin\left[2\pi(f_0 + f_d)(t - t_D) + \pi\alpha(t - t_D)^2 + \phi\right] \tag{6.29}$$

where t_D is the time delay from the electromagnetic wave propagation, f_d is the Doppler shift and ϕ is an unknown phase shift. t_D contains the distance information.

3. The received signal has a reference chirp wave mixed with it.

$$y_{ref}(t) = C\sin\left[2\pi f_{ref}(t - t_{ref}) + \pi\alpha_{ref}(t - t_{ref})^2\right] \tag{6.30}$$

where α_{ref} is the frequency slope of the reference chirp and t_{ref} is the time that the chirp reference is started. The multiplication of two sine waves gives

$$\sin(x_0)\sin(x_1) = \frac{1}{2}\left[\cos(x_0 - x_1) - \cos(x_0 + x_1)\right] \tag{6.31}$$

4. The mixed signal, after low pass filtering is given by

$$y_{out} = \frac{BC}{2}\cos\left[2\pi(f_0 + f_d - f_{ref})(t - t_D) + \pi(\alpha - \alpha_{ref})(t - t_D)^2\right. \tag{6.32}$$

$$\left. + 2\pi\alpha_{ref}(t_{ref} - t_D)(t - t_D) + \phi\right] \tag{6.33}$$

5. The final equation for the distance of the target (assuming $\alpha = \alpha_{ref}$, $f_0 = f_{ref}$ and $\beta = \frac{BC}{2}$) becomes

$$y_{out} = \beta\cos\left[2\pi(f_d + \alpha(t_{ref} - t_D))(t - t_D)\right] \tag{6.34}$$

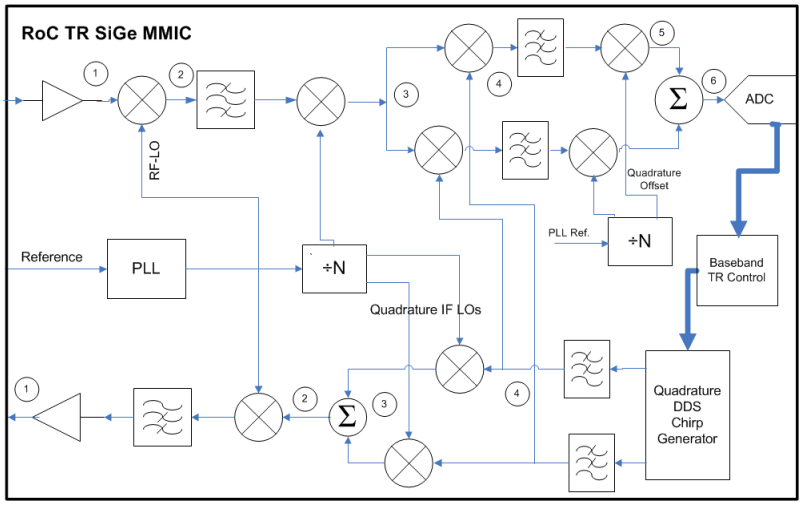


Figure 6.15: Radar-On-Chip Block Diagram

Performing spectrum analysis yields a frequency pulse at

$$f_{stretch} = f_d + \alpha(t_{ref} - t_D) \tag{6.35}$$

we know α and t_{ref}, so solving for t_D and applying Equation 6.25 we get

$$R = \frac{c}{2}\left(\frac{f_d}{\alpha} - t_D\right) \tag{6.36}$$

The range-Doppler coupling is inherent to stretch processing. The Doppler information must be extracted using another method.

6.3.1 Single Chip Radar

A single chip radar solution was developed using the stretch processing technique. Figure 6.15 shows a simplified block diagram of the single chip radar solution. The design was fabricated in a 130 nm BiCMOS process in 2011. The quadrature DDFS chirp generator shown in the figure was implemented using a partial dynamic rotation CORDIC. The detailed block diagram of the component is shown in Figure 6.24.

Figure 6.16 shows the die photograph of the RoC chip. The components from the block diagram are labelled on the silicon. The DACs marked in the figure are described in Section 6.6. The DCDO (labelled NCO in the figure) is a PDR CORDIC described in Section 6.4.

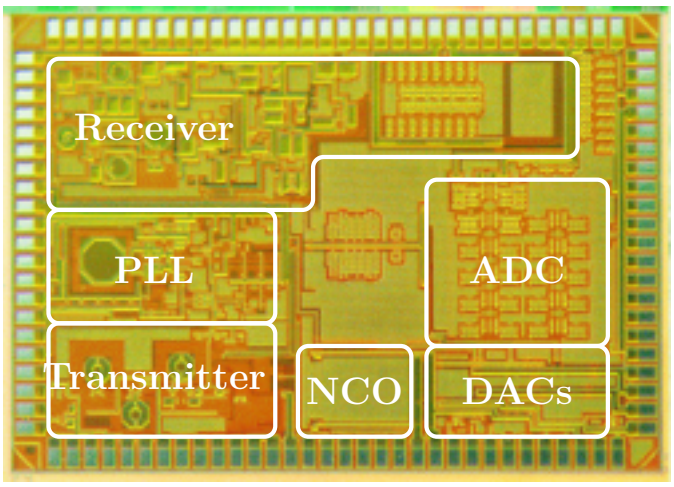


Figure 6.16: Die Photograph of RoC

6.4 CORDIC

For ROMs to achieve compression, approximations must be introduced into the phase to sine/cosine mapping function (SCMF). Another technique that has been widely used for accurate elementary function generation, such as trigonometric functions, is the Cordinate Rotation Digital Computer (CORDIC) algorithm. As the name implies, the CORDIC algorithm works by performing iterative rotations to a vector. Most of the early research revolved around digital computing and audio synthesis [53], but as CMOS processes have advanced it has found a place in high speed frequency synthesis [21]. The algorithm is relevant to this work, as a modified version of the CORDIC algorithm is sufficiently fast enough to be used in high speed DDFS. First however, a brief history of the CORDIC algorithm, along with the basic theory required to understand the algorithm, will be presented.

6.4.1 Basic Theory

The mathematical theory behind the CORDIC algorithm is quite old; however, the first observation that the technique could compute trigonometric equations in digital computers was made by Volder [53] in 1959. In his seminal paper, Volder notes that CORDIC can solve equations of the following two forms:

$$Y' = K\left(Y\cos\left(\lambda\right) + X\sin\left(\lambda\right)\right) \tag{6.37}$$

$$X' = K\left(X\cos\left(\lambda\right) - Y\sin\left(\lambda\right)\right) \tag{6.38}$$

and

$$R = K\sqrt{X^2 + Y^2} \tag{6.39}$$

$$\theta = \tan^{-1}\left(\frac{Y}{X}\right) \tag{6.40}$$

The two sets of equations also describe two conceptual modes of operation. These modes are *rotation* mode and *vectoring* mode and correspond to the solutions of Equation 6.37 and Equation 6.39 respectively. The rotation mode calculates the vector coordinates given a starting vector and an angle of rotation (i.e. it finds the x and y coordinates of a known vector after rotation). Vectoring mode calculates the magnitude and angle of a vector given the coordinates of the vector.

Both rotation mode and vectoring mode CORDICs are used in the implementation of the BIST system described at the end of Chapter 4. The rotation mode CORDIC is used to compute sine and cosine in a DDFS and consequently will be the focus of this section. To understand how CORDIC works and how the partial dynamic rotation CORDIC of the following sections is derived, it is helpful to derive the CORDIC algorithm itself.

In the conclusion of Volder's paper, he notes that other algorithms that use the fundamental concept of CORDIC can be used to solve different computing problems. It should

come as no surprise then that the CORDIC algorithm as described by Volder would eventually be generalized in a mathematical sense. The generalized (i.e. "unified") algorithm was introduced by Walther [54] in 1971.

There are two methodologies for arriving at the CORDIC algorithm. Muller [55] uses a non-restoring decomposition algorithm on the angle θ to derive the necessary recursive sequence to perform CORDIC. The other approach, whose variant is used in the derivation of the CORDIC recursive sequence in this chapter, is used by Walther [54]. Let $\mathbf{v}_i = (x_i, y_i)$ be a coordinate in the Cartesian coordinate system ($\mathbb{R}^2$). The bold characters in mathematical notation are used to represent vectors and matrices. Let $\mathbf{v}_{i+1}$ be generated from $\mathbf{v}_i$ using the following relationship:

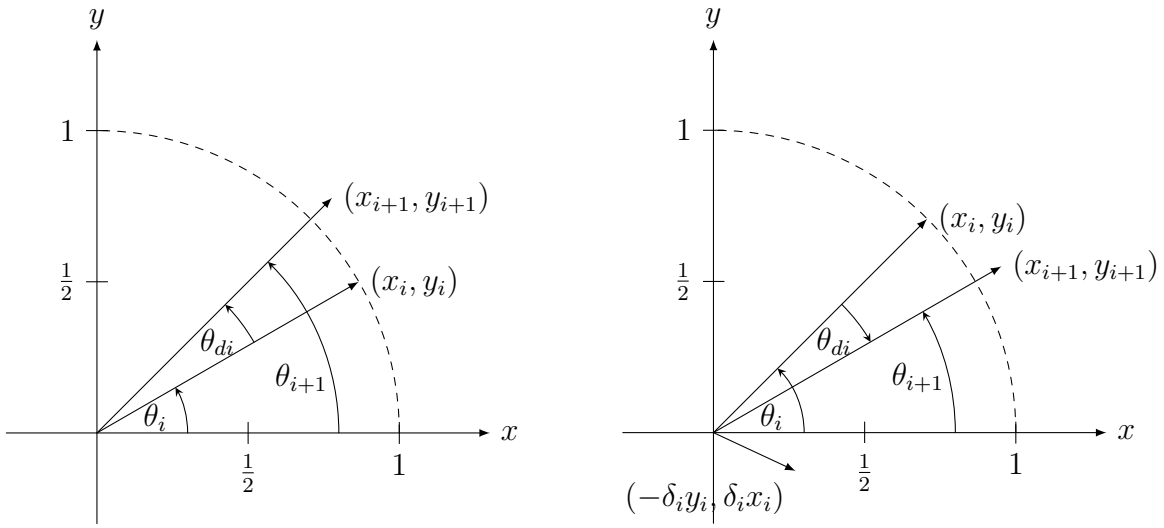
$$x_{i+1} = x_i - \delta_i y_i \tag{6.41}$$

$$y_{i+1} = y_i + \delta_i x_i \tag{6.42}$$

where $\delta_i \in \mathbb{R}$. Figure 6.17a shows an example of the effect of applying Equations 6.41 and 6.42 with positive δ_i to arbitrarily chosen coordinates (x_i, y_i). Notice the transformation equations can be equivalently thought of as adding a transformation vector $\mathbf{v}_{di} = (-\delta_i y_i, \delta_i x_i)$ to $\mathbf{v}_i$. Figure 6.17b shows the effect of a negative δ_i on the subsequent vector. Notice that the magnitude of the vector in both figures is scaled and no longer sits on the unit circle.

To begin to understand the behavior of the iterative equations, consider how the magnitude of $\mathbf{v}_{i+1}$ is related to $\mathbf{v}_i$.

$$\begin{aligned}
|\mathbf{v}_{i+1}| &= \sqrt{x_{i+1}^2 + y_{i+1}^2} \\
&= \sqrt{(x_i - \delta_i y_i)^2 + (y_i + \delta_i x_i)^2} \\
&= \sqrt{(x_i^2 - 2\delta_i x_i y_i + \delta_i^2 y_i^2) + (y_i^2 + 2\delta_i x_i y_i + \delta_i^2 y_i^2)} \\
&= \left(\sqrt{1 + \delta_i^2}\right) \sqrt{x_i^2 + y_i^2} = \sqrt{1 + \delta_i^2}\, |\mathbf{v}_i|
\end{aligned} \tag{6.43}$$



(a) Rotation with Positive δ_i (b) Rotation with Negative δ_i

Figure 6.17: CORDIC Vector Rotations

Let $\mathbf{v}_0 = (x_0, y_0)$ be the initial starting vector. Then the magnitude of the vector $\mathbf{v}_n$ is

$$|\mathbf{v}_n| = |\mathbf{v}_0| \prod_{i=0}^{n-1} \left(\sqrt{1 + \delta_i^2} \right) = |\mathbf{v}_0| \, K_n \tag{6.44}$$

where K_n is used in traditional CORDIC literature, though often without the subscript, to denote the product of magnitudes of iterative rotations [53] [54].

It is clear from Figure 6.17a and Figure 6.17b that a positive δ_i results in a positive θ_{di} (i.e. counter-clockwise rotation) and a negative δ_i results in a negative θ_{di} (i.e. clockwise rotation). The analysis that follows will show this mathematically. Calculate the relationship between the angle of $\mathbf{v}_i$ and $\mathbf{v}_{i+1}$. This is the sum of the angle of $\mathbf{v}_i$, denoted θ_i, and the angle between the vectors $\mathbf{v}_{i+1}$ and $\mathbf{v}_i$, denoted θ_{di}.

$$\theta_{i+1} = \theta_i + \theta_{di} \tag{6.45}$$

The following dot product relationship is used to calculate θ_{di}:

$$\cos(\theta_{di}) = \frac{\mathbf{v}_{i+1} \cdot \mathbf{v}_i}{|\mathbf{v}_{i+1}|\,|\mathbf{v}_i|} \tag{6.46}$$

$$= \frac{x_i(x_i - \delta_i y_i) + y_i(y_i + \delta_i x_i)}{(x_i^2 + y_i^2)\sqrt{1+\delta_i^2}} \tag{6.47}$$

$$= \frac{1}{\sqrt{1+\delta_i^2}} \tag{6.48}$$

Now using the cosine to tangent trigonometric identity

$$\cos(\theta_{di}) = \pm\frac{1}{\sqrt{1+\tan^2(\theta_{di})}} \tag{6.49}$$

The plus or minus arises from the fact the cosine is a even function and tangent is an odd function. In DDFS systems, only the first quadrant, where both tangent and cosine are positive, is of interest. In this case only the positive solution is valid. The solution for θ_{di} is then found by taking the inverse tangent:

$$\theta_{di} = \tan^{-1}(\delta_i) = \sigma_i \tan^{-1}(|\delta_i|) \tag{6.50}$$

where σ_i as the sign of δ_i. The inverse tangent function has odd symmetry, so for positive δ_i, θ_{di} is positive and for negative δ_i, θ_{di} is negative. This is in full agreement with the qualitative assesment from the previous figures. The σ_i is chosen to rotate the vector in the correct direction. Then the angle of $\mathbf{v}_n$ is

$$\theta_n = \theta_0 + \sum_{i=0}^{n-1} \sigma_i \tan^{-1}(|\delta_i|) = \theta_0 + \theta_{sn} \tag{6.51}$$

where θ_{sn} is the sum of the delta angles. At this point, it is interesting derive the coordinates after n such operations, x_n and y_n. Firstly x_n is computed using the cosine sum of angles

identity and the geometric observations that $\cos(\theta_0) = x_0$ and $\sin(\theta_0) = y_0$.

$$\begin{aligned} x_n &= |\mathbf{v}_n| \cos(\theta_0 + \theta_{sn}) \\ &= K_n |\mathbf{v}_0| [\cos(\theta_0)\cos(\theta_{sn}) - \sin(\theta_0)\sin(\theta_{sn})] \\ &= K_n |\mathbf{v}_0| [x_0 \cos(\theta_{sn}) - y_0 \sin(\theta_{sn})] \end{aligned} \tag{6.52}$$

Likewise for y_n, while not derived as explicitly

$$y_n = K_n |\mathbf{v}_0| [x_0 \sin(\theta_{sn}) + y_0 \cos(\theta_{sn})] \tag{6.53}$$

Depending on the initial value $\mathbf{v}_0$ chosen, different functions can be generated. Since this work concerns itself with the generation of sinusoids, it is paramount for correct operation. Using Equation 6.52 and Equation 6.53, it is clear that setting $x_0 = 1$ and $y_0 = 0$ generates both a sine and cosine output. Let $x_0 = 1$ and $y_0 = 0$, then $|\mathbf{v}_0| = 1$ and

$$x_n = K_n \cos(\theta_{sn}) \tag{6.54}$$

$$y_n = K_n \sin(\theta_{sn}). \tag{6.55}$$

Then the recursive equations so far have yielded a means to compute sine and cosine of θ_{sn} using only iterative multiplications and additions. δ_i has not been assigned a value, but from Equation 6.51 it is clear that θ_{sn} is a function of δ_i. One wants the sum of θ_{di} to "converge" on a desired angle α. Let α be the angle for whose sine and cosine value we wish to compute by this technique. A formal definition of convergence is required to proceed [14]:

Definition 6.1 (Convergent Series (Real))**.** *Let* $\{x_1, x_2, \cdots\} \in \mathbb{R}$ *be an infinite sequence of numbers, then the series of this sequence of numbers is said to converge to* $X \in \mathbb{R}$ *if given*

any $\varepsilon > 0$, $\varepsilon \in \mathbb{R}$ *there exists an integer* $N > 0$ *such that*

$$\left| X - \sum_{i=1}^{n} x_i \right| < \varepsilon \tag{6.56}$$

for all $n > N$.

Convergence is sometimes written as a single limit statement for conciseness:

$$X = \lim_{n \to \infty} \sum_{i=1}^{n} x_i. \tag{6.57}$$

For the algorithm to be useful for computing sine and cosine, θ_{sn} must converge to any arbitrary angle between 0 and $\pi/2$. Or more precisely, for any arbitrary $\alpha \in [0, \pi/2)$ and $\varepsilon > 0$ there exists an integer $N > 0$ such that

$$\left| \alpha - \sum_{i=1}^{n} \sigma_i \tan^{-1}\left(|\delta_i|\right) \right| < \varepsilon \tag{6.58}$$

for some $n > N$. The properties of series that converge are set forth using the Cauchy convergence criterion [14], which is

Theorem 6.1 (Cauchy Convergence Criterion (Real))**.** *A series*

$$\sum_{i=1}^{\infty} x_i \tag{6.59}$$

converges if and only if, given any $\varepsilon > 0$, $\varepsilon \in \mathbb{R}$ *there exists an integer* N *such that*

$$|x_{m+1} + \cdots + x_n| < \varepsilon \tag{6.60}$$

for all $n > m \geq N$.

Let us consider the sequences for which the series converges. This can be inferred from the Cauchy convergence criterion or proved simply as below:

Lemma 6.1 (Sequences for Convergent Series)**.** *If the sum of* $\{x_1, x_2, \cdots\} \in \mathbb{R}$ *converges, then*

$$\lim_{n\to\infty} x_n = 0 \tag{6.61}$$

Proof. Let $X = \lim_{n\to\infty} \sum_{i=1}^{n} x_i$. Note that $X = \lim_{n\to\infty} \sum_{i=1}^{n+1} x_i$ also. Then

$$\lim_{n\to\infty} \left(\sum_{i=1}^{n+1} x_i - \sum_{i=1}^{n} x_i \right) = \lim_{n\to\infty} x_{n+1} = 0 \tag{6.62}$$

since the limits of both series are the same. □

Note that the converse of Lemma 6.1 does not guarantee convergence for the series but Theorem 6.1 does. The harmonic series is a good example of series whose elements approach zero toward infinity, but whose series diverges. From the previous lemma,

$$\lim_{n\to\infty} \sigma_n \tan^{-1}\left(|\delta_n|\right) = \lim_{n\to\infty} \tan^{-1}\left(|\delta_n|\right) = 0 \tag{6.63}$$

The sign σ_n has no impact on the convergence of the sequence to zero. So δ_i must be chosen such that the inverse tangent of its sequence has a limit of zero. There are several ways to easily show this, but perhaps the easiest is to note the Taylor series expansion of inverse tangent leads to a small angle approximation similar to that of sine (i.e. $\tan^{-1}(x) \approx x$ for $x \ll 1$.

$$\tan^{-1}(x) = \sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1} \left(x^{2n+1}\right) = x - \frac{x^3}{3} + \frac{x^5}{5} - \cdots \tag{6.64}$$

Note the higher order x terms rapidly become negligible for small x.

There are multiple tests that can be used to determine convergence in a series, one such test is the *ratio test* [14]. If

$$\lim_{n\to\infty}\left|\frac{\tan^{-1}(\delta_{n+1})}{\tan^{-1}(\delta_n)}\right| < 1 \tag{6.65}$$

then the series converges absolutely. It has already been shown that it is required that $\tan^{-1}(\delta_i)$ approach zero for large values of i. Using the small angle approximation as n tends towards infinity, Equation 6.65 can be rewritten as

$$\lim_{n\to\infty}\left|\frac{\delta_{n+1}}{\delta_n}\right| < 1 \tag{6.66}$$

Now it need only be shown that the algorithm described in Definition 6.2 can converge to any angle between some convergence limits $\theta_{\max}$ and $\theta_{\min}$. This would complete the description of an algorithm capable of simultaneously computing a scaled sine or cosine to arbitrary precision given enough iterations. In order to cover the entire interval, a second less obvious property must be met.

$$|\delta_n| \leq \sum_{i=n+1}^{\infty} |\delta_i| \tag{6.67}$$

For the series of δ_i to take any value on the interval, the size of the previous step must be less than or equal to the sum of the remaining steps. This is more obvious when shown graphically, as in Figure 6.18. Unless the sum of the remaining steps is equal or greater than the size of the previous step, then there is a gap in the obtainable values from the algorithm. A gap in δ_i would result in a gap in $\tan^{-1}(\delta_i)$ and there would be unobtainable angles of rotation.

The CORDIC algorithm works by keeping track of the angle of rotation at each iteration to determine if the rotations have passed the desired target angle. In CORDIC literature, the variable z_i is used to denote the angle information at iteration i. One could start z_0

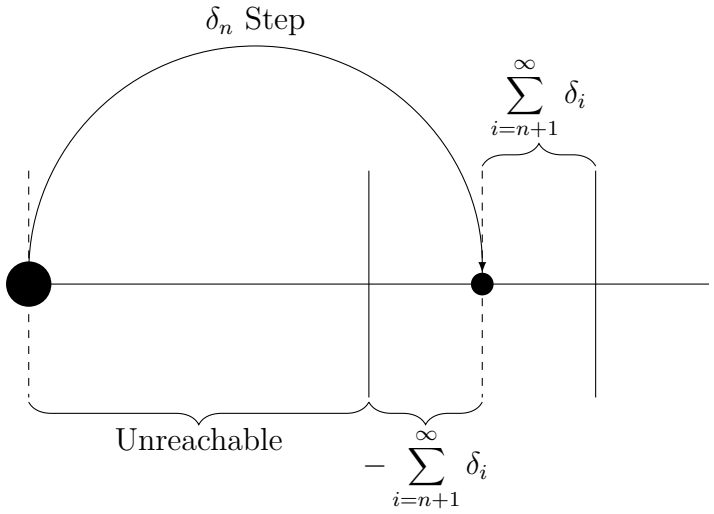


Figure 6.18: CORDIC Coverage Requirement

at zero and sum the θ_{di} components and σ_i could be computed by subtracting α from the current θ_i to determine the direction of rotation. A more efficient approach would be to start $z_0 = \alpha$ and subtract the θ_{di} while checking the sign of the residual angle at each iteration. This is how σ_i is selected. Define the iterative relationship:

$$z_{i+1} = z_i - \theta_{di} = z_i - \sigma_i \tan^{-1}(|\delta_i|) \tag{6.68}$$

where $\sigma_i = 1$ if $z_i \geq 0$ or $\sigma_i = -1$ otherwise. At the nth iteration,

$$z_n = z_0 - \sum_{i=0}^{n-1} \sigma_i \tan^{-1}(|\delta_i|) = z_0 - \theta_{sn} \tag{6.69}$$

If z_n converges to zero at the limit for n as it approaches infinity, then $\theta_{sn} = z_0$ and from Equations 6.54 and 6.55 one sees that the algorithm computes the sine and cosine of θ_{sn}. Clearly then, proving that z_n converges to zero from a given $z_0 = \alpha$ is sufficient to prove convergence for the CORDIC algorithm.

Theorem 6.2 (CORDIC Convergence Theorem)**.** *Let* $\tan^{-1}(\delta_1), \tan^{-1}(\delta_2), \ldots$ *be a sequence of real numbers whose series is convergent by the Cauchy convergence criterion for*

series (Theorem 6.1). If

$$\left|\tan^{-1}(\delta_n)\right| \leq \sum_{k=n+1}^{\infty} \left|\tan^{-1}(\delta_k)\right| \tag{6.70}$$

for any $n \in \mathbb{P}$, then for any $\alpha \in \mathbb{R}$ constrained by

$$-\sum_{i=0}^{\infty} \left|\tan^{-1}(\delta_i)\right| \leq \alpha \leq \sum_{i=0}^{\infty} \left|\tan^{-1}(\delta_i)\right| \tag{6.71}$$

The CORDIC z recurrence relation when seeded with $z_0 = \alpha$ converges, i.e.

$$\lim_{n\to\infty} z_n = 0 \tag{6.72}$$

where z_n is defined by Equation 6.68.

Proof. Let α be chosen according to Equation 6.71. Let S be a non-empty subset of $\mathbb{P}_0$. Let us show that

$$-\sum_{k=n}^{\infty} \left|\tan^{-1}(\delta_k)\right| \leq z_n \leq \sum_{k=n}^{\infty} \left|\tan^{-1}(\delta_k)\right| \tag{6.73}$$

for all n by induction. Since the series

$$\sum_{i=1}^{\infty} \tan^{-1}(\delta_i) \tag{6.74}$$

converges by the Cauchy criterion (Theorem 6.1), for any arbitrary $\varepsilon > 0$, $\varepsilon \in \mathbb{R}$ there exists an integer $N > 0$ such that

$$\left|\tan^{-1}(\delta_{m+1}) + \cdots + \tan^{-1}(\delta_n)\right| < \varepsilon \tag{6.75}$$

for all $n > m \geq N$. This implies

$$\lim_{n\to\infty} \sum_{k=n}^{\infty} \left|\tan^{-1}(\delta_k)\right| = 0 \tag{6.76}$$

and thus showing Equation 6.73 to be true would prove convergence by making the upper and lower limit of z_n equal to zero. First let us check that $0 \in S$ by setting $n = 0$ and using Equation 6.73.

$$-\sum_{k=0}^{\infty} \left|\tan^{-1}(\delta_k)\right| \leq z_0 \leq \sum_{k=0}^{\infty} \left|\tan^{-1}(\delta_k)\right| \tag{6.77}$$

Since $z_0 = \alpha$ and α has been chosen according to Equation 6.71, $0 \in S$. Now we take the induction step. Assume $x \in S$. It must be shown that $x + 1 \in S$ by showing that

$$-\sum_{k=x+1}^{\infty} \left|\tan^{-1}(\delta_k)\right| \leq z_{x+1} \leq \sum_{k=x+1}^{\infty} \left|\tan^{-1}(\delta_k)\right|. \tag{6.78}$$

z_{x+1} is computed as follows:

$$z_{x+1} = z_x - \sigma_x \tan^{-1}(|\delta_x|) \tag{6.79}$$

Since we have assumed $x \in S$,

$$-\sum_{k=x}^{\infty} \left|\tan^{-1}(\delta_k)\right| \leq z_x \leq \sum_{k=x}^{\infty} \left|\tan^{-1}(\delta_k)\right| \tag{6.80}$$

There are two cases to examine, $z_x \geq 0$ for which $\sigma_x = 1$, $\delta_x \geq 0$ and $z_x < 0$ for which $\sigma_x = -1$, $\delta_x < 0$. Assume $z_x \geq 0$, then $\sigma_x = 1$ and

$$z_{x+1} = z_x - \tan^{-1}(\delta_x) = z_x - \left|\tan^{-1}(\delta_x)\right| \tag{6.81}$$

The upper bound is found by substituting the upper bound of Equation 6.80 into Equation 6.81 and adjusting the equality to an inequality.

$$z_{x+1} \leq \sum_{k=x}^{\infty} \left|\tan^{-1}(\delta_k)\right| - \left|\tan^{-1}(\delta_x)\right| \tag{6.82}$$

$$\leq \sum_{k=x+1}^{\infty} \left|\tan^{-1}(\delta_k)\right| \tag{6.83}$$

The lower bound is found by substituting Equation 6.70 into Equation 6.81 and adjusting the equality to an inequality.

$$z_{x+1} = z_x - \left|\tan^{-1}(\delta_x)\right| \Rightarrow z_{x+1} \geq z_x - \sum_{k=x+1}^{\infty} \left|\tan^{-1}(\delta_k)\right| \tag{6.84}$$

But z_x is positive or zero, so

$$z_{x+1} \geq - \sum_{k=x+1}^{\infty} \left|\tan^{-1}(\delta_k)\right| \tag{6.85}$$

and the z_{x+1} is bounded for the case $z_x \geq 0$. Now consider $z_x < 0$, then $\sigma_x = -1$.

$$z_{x+1} = z_x + \left|\tan^{-1}(\delta_x)\right| \tag{6.86}$$

The lower bound is computed by substituting the lower bound of Equation 6.80 into Equation 6.86

$$z_{x+1} \geq - \sum_{k=x}^{\infty} \left|\tan^{-1}(\delta_k)\right| + \left|\tan^{-1}(\delta_x)\right| \tag{6.87}$$

$$\geq - \sum_{k=x+1}^{\infty} \left|\tan^{-1}(\delta_k)\right| \tag{6.88}$$

The upper bound is computed by substituting Equation 6.70 into Equation 6.86

$$z_{x+1} = z_x + \left|\tan^{-1}(\delta_x)\right| \Rightarrow z_{x+1} \leq z_x + \sum_{k=x+1}^{\infty} \left|\tan^{-1}(\delta_k)\right| \tag{6.89}$$

But z_x is negative, so

$$z_{x+1} \leq \sum_{k=x+1}^{\infty} \left|\tan^{-1}(\delta_k)\right| \tag{6.90}$$

Thus $x+1 \in S$ and the proof is complete. □

Note that this proof for the CORDIC algorithm is different from Walther's proof (perhaps a little more formal) [54]. As far is the author is aware, this is an original proof. The conventional CORDIC algorithm has been fully derived at this point. Definition 6.2 summarizes the results of the analysis thus far.

Definition 6.2 (Conventional CORDIC Iteration)**.** *The conventional CORDIC iteration is defined as*

$$x_{i+1} = x_i - \sigma_i \delta_i y_i \tag{6.91}$$

$$y_{i+1} = y_i + \sigma_i \delta_i x_i \tag{6.92}$$

$$z_{i+1} = z_i - \sigma_i \tan^{-1}(\delta_i). \tag{6.93}$$

where $\sigma_i = 1$ if $z_i \geq 0$ or $\sigma_i = -1$ otherwise. Performing successive CORDIC iterations to compute some desired value is called the CORDIC algorithm.

There are infinitely many such δ_i that can satisfy Equation 6.66 and Equation 6.67. What is desired is an efficient hardware implementation of Equation 6.41 and Equation 6.42. As has already been mentioned in the BTM and MTM sections, a multiplication or division by 2 is merely a shift operation, which is remarkably efficient in hardware (no combinatorial logic is required). Let $|\delta_i| = 2^{-i}$, a shift by i operation in hardware. Now let us calculate whether this series converges using the ratio test.

$$\lim_{n\to\infty} \left|\frac{2^{-(n+1)}}{2^{-n}}\right| = \lim_{n\to\infty} \frac{1}{2} = \frac{1}{2} < 1 \tag{6.94}$$

Clearly, the series then converges. Now let us calculate the interval of convergence. The maximum angle of rotation would be

$$\theta_{\max} = \sum_{i=0}^{\infty} \tan^{-1}\left(2^{-i}\right) \tag{6.95}$$

$$= \tan^{-1}(1) + \tan^{-1}\left(\frac{1}{2}\right) + \tan^{-1}\left(\frac{1}{4}\right) + \cdots \tag{6.96}$$

$$= 1.7432865\ldots \approx 0.55\pi \tag{6.97}$$

The minimum obtainable rotation angle would be $\theta_{\min} = -\theta_{\max} = -1.7432865\ldots$. Since quarter sine compression is typically used in a DDFS, the 0.55π is more than sufficient for implementing the SCMF. From this point forward δ_i will assumed to be a power of 2.

6.4.2 Conventional CORDIC

Figure 6.19 shows the hardware implementation of the conventional CORDIC iteration. Thus three full adders are required. Since z_i is a two's complement number, the sign detection block is merely a check on the MSB of z_i. The hard-wired shifts are "free" operations, requiring no additional hardware. The multiplication by -1 can be implemented using the one's complement technique described in Figure 1.2. Overall this is rather low hardware overhead for a technique that can compute a sinusoid to arbitrary precision.

One of the drawbacks of conventional CORDIC implementations is the scalar K from the iterative rotations. As n tends towards infinity,

$$K_\infty = \prod_{i=0}^{\infty} \left(\sqrt{1 + 2^{-2i}}\right) = 1.64676\ldots \tag{6.98}$$

But since K_n is a function of the number of iterations, the conventional CORDIC typically has a fixed number of iterations. One brute force method to is to initalize $x_0 = 1/K_n$ and thus the final computation results in $\cos(\alpha)$ and $\sin(\alpha)$. Other methods to correct for this unwanted scaling, and when the scaling can be ignored, will be discussed later in this chapter.

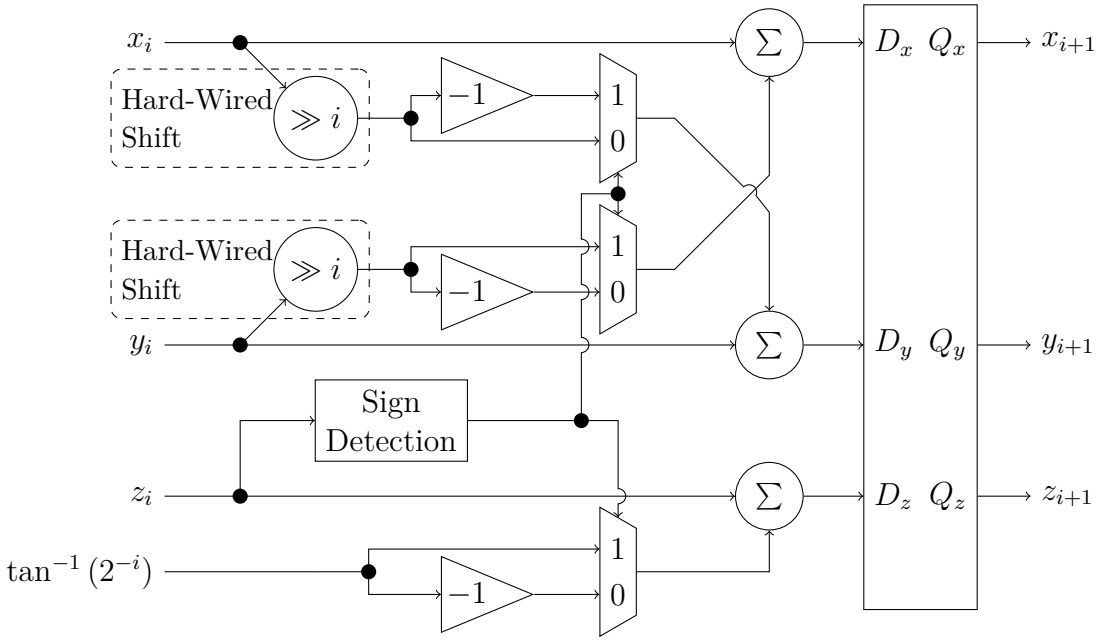
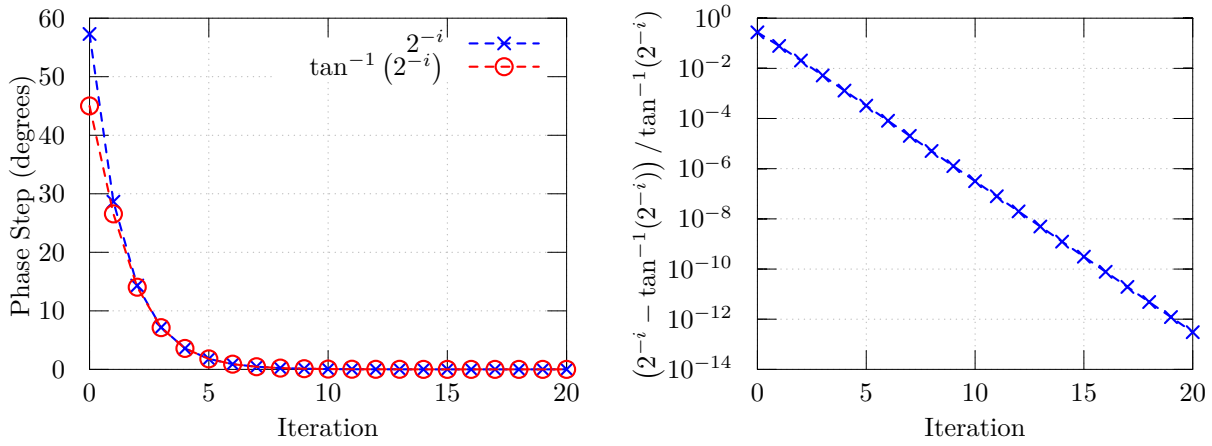


Figure 6.19: Conventional CORDIC Stage

Note that the upper bound of the remaining phase error, assuming vectoring mode operations with $z_0 = \alpha$, after n rotations is

$$\sum_{i=n+1}^{\infty} \left|\tan^{-1}\left(2^{-i}\right)\right| \tag{6.99}$$

The angle remaing phase error is bounded by the small angle approximation for arctangent. Figure 6.20a shows the value of arctangent against each CORDIC iteration as well as the small angle approximation of arctangent at each value. The better question is how the quality of the small angle approximation changes with each iteration. Figure 6.20b shows this information by taking the ratio of the magnitude of error of the small-angle approximation for arctangent at a certain iteration and dividing it by the arctangent value of that iteration. This will prove helpful in calculating upper error bounds on the phase and amplitude in the CORDIC algorithm in the following section.



(a) arctan Versus Iteration

(b) Quality of Approximation

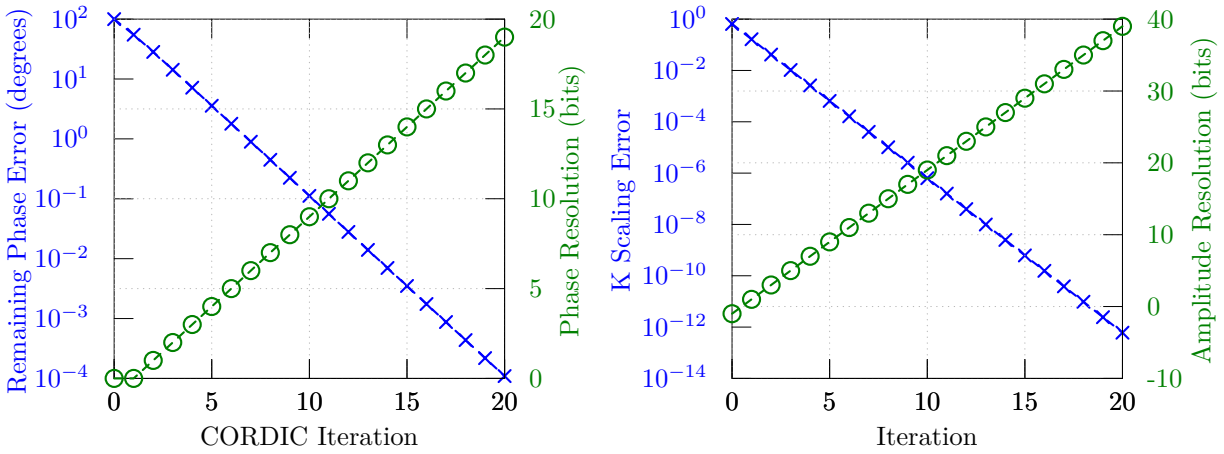
Figure 6.20: arctan Small Angle

6.4.3 Optimizing the CORDIC Algorithm for DDFS

The conventional CORDIC algorithm when started at zero phase and initialized such that the scaling factor is normalized covers a phase range of -0.55π to 0.55π. Since DDFS systems use quarter wave sinusoidal compression, this is over twice the required range for convergence. We only wish to compute angles in the interval $[0, \pi/2)$. This means that the CORDIC algorithm always takes the same first step, since $z_0 = \alpha$ is always greater than or equal to zero. Figure 6.21a shows the upper bound of the "remaining phase error" at iteration i (computed using Equation 6.99). Starting at iteration 0, $\theta_0 = 0$, the phase error upper boundary is the entire $\pi/2$, or 90°, range. After one iteration, the maximum phase error is 54.9°.

The question remains of how y_0, x_0 and z_0 should be initialized for the optimization to work correctly. Using Equation 6.52 and 6.53 is clear that $x_0 = \cos(\theta_0) = \cos(\pi/4) = \sqrt{2}/2$ and $y_0 = \sin(\pi/4) = \sqrt{2}/2$. These values can be pre-computed and stored in the a ROM for initialization. Using Equation 6.68, it is clear that z must be set to

$$z_0 = \alpha - \frac{\pi}{4} \tag{6.100}$$



(a) Phase Error Versus Iteration

(b) K Versus Iteration

Figure 6.21: CORDIC Bit Resolution

A brute force approach would be to directly compute this number and use it as a seed for the z-path of the CORDIC. Starting at $\theta_0 = \pi/4$ means that the CORDIC algorithm only needs to cover the interval $[-\pi/4, \pi/4]$ or $[-0.25\pi, 0.25\pi]$. Skipping the first CORDIC stage (i.e. starting $i = 1$) converges over the interval $[-0.305\pi, 0.305\pi]$ which is sufficient to cover the required interval.

The optimization of starting the vector at an angle of $\pi/4$ and skipping the first CORDIC iteration effectively eliminates one CORDIC iteration. The number of rotations can be further reduced by extending this idea to a general purpose look-up table that initializes the CORDIC algorithm to the m^{th} iteration. The green line of Figure 6.21a shows the numbers of bits of phase resolution obtained for a given upper bound of phase error. From Figure 6.20a and Figure 6.20b, along with Equation 6.64 for the small angle approximation, it shows that each CORDIC iteration gain a single bit of phase accuracy. This can be directly related to an amplitude error and consequently an effective number of bits. Let θ_r be the remaining phase error. The sine difference formula, which has been used in nearly every chapter of this

document, yields

$$\sin(\alpha - \theta_r) \approx \sin(\alpha) + \theta_r \cos(\alpha) \qquad \text{for } \theta_r \ll 1 \tag{6.101}$$

The error is then $\theta_r \cos(\alpha)$, but $\alpha \in [0, \pi/2)$ and therefore the error is bounded by the maximum value of cosine over that interval, or $\cos(0) = 1$. The maximum error is then θ_r, which is precisely the remainin phase error. So the green line of Figure 6.21a also predicts the number of bits of amplitude resolution provided by the CORDIC after i iterations *without error*. Figure 6.22 shows how x_0 and y_0 are derived by sending the MSBs of the DDFS phase word to a look-up table.

Assume that B_{LUT} bits are used for the lookup table. Then the $\pi/2$ CORDIC search range is reduced to:

$$\theta_{\text{ROT}} = \frac{\pi}{2^{B_{\text{LUT}}+1}} \tag{6.102}$$

If an offset is introduced into the LUT in the same manner as described in the $\pi/4$ optimization offset discussion, an extra iteration can be reduced. Thus a six bit LUT with a half-LSB offset elminates 7 CORDIC operations. That is to say, the first iteration would be $i = 8$ using Figure 6.21a.

An interesting phenomenon happens with the scaling factor when initializing the CORDIC algorithm. Figure 6.21b shows the scaling factor value of the i^{th} iteration. The green line shows the number of amplitude bits of resolution required at the output before the K_n scaling factor to introduce an error above quantization. From Equation 6.44, it should come as no surprise that the amplitude error decreases at a rapid rate, particularly with the 2^{-2i} rate of decrease in error per term.

6.4.4 Partial Dynamic Rotation CORDIC

The generalized PDR CORDIC architecture with support for conventional CORDIC stages is shown in Figure 6.22. This component is used the radar DDFS system described in this chapter. Consider a conventional CORDIC with $\theta_0 = 0$ and let $\alpha = 5°$. The first step would be $\tan^{-1}(2^0) = 45°$. This step far overshoots the desired angle of rotation. There is enough information to avoid this overshoot because the starting angle is known, the desired angle is known and the amount of rotation for a given 2^{-i} is known. Keep in mind that α is a binary word. Looking at the MSBs of word give an idea of how much rotation is required. In this case, starting at iteration $i = 3$ yields $\tan^{-1}(2^{-3}) = 7.125°$.

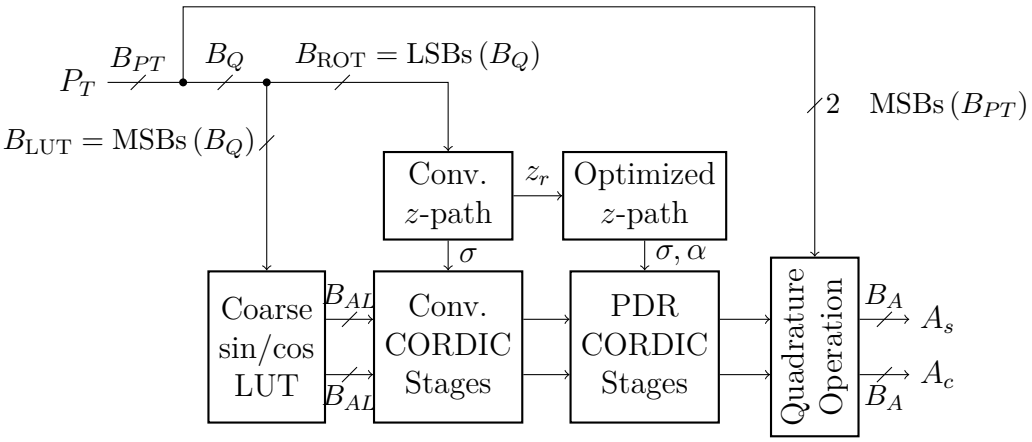


Figure 6.22: PDR CORDIC Architecture

One of the major drawbacks of the PDR CORDIC is that K_n for a fixed number of stages changes based on the requested angle α. It has already been shown in Figure 6.21b that if the CORDIC is initalized before rotations begin, the impact of K_n on the magnitude of the output is negligible. Figure 6.23 shows the block diagram for a PDR CORDIC rotation stage.

The dynamic rotation selection (DRS) logic shown in Figure 6.23 becomes quite simple if the CORDIC stages are seeded with enough resolution. Recall from the small angle approximation in Figure 6.20a that $\tan^{-1}(2^{-i}) \approx 2^{-i}$ for large i. Then the check to find the appropriate rotation angle is merely a check on the MSB position of the remaining phase

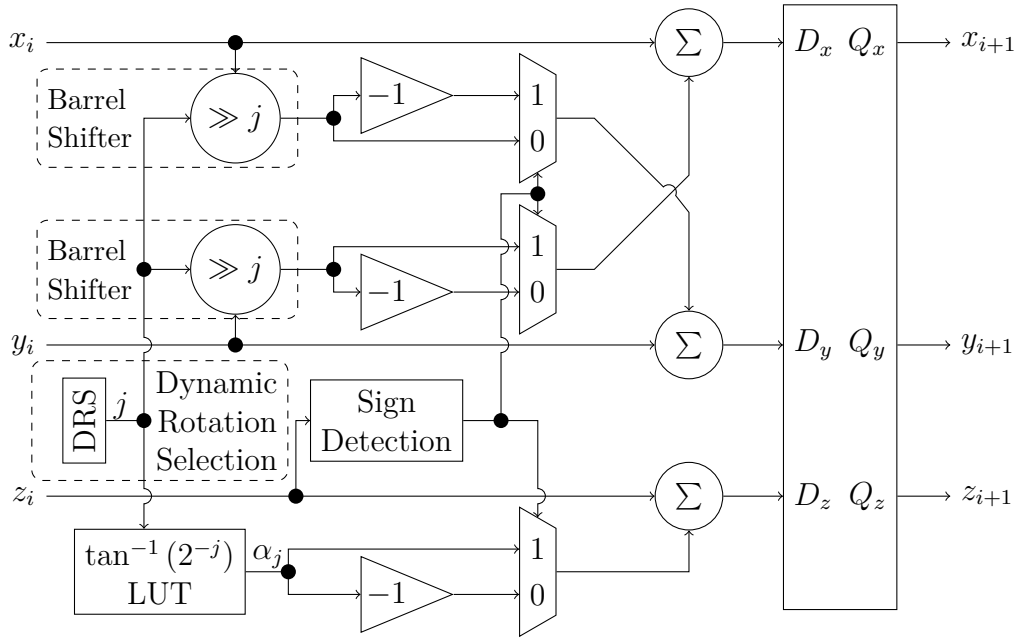


Figure 6.23: PDR CORDIC Stage

Design	B_A	B_P	SINAD	SFDR (W.C)	Frequency	Area
BTM	12	32	66	70 dBc	680 MHz	0.013 mm^2
MTM	11	32	N/A	58 dBc	1.0 GHz	0.008 mm^2
CORDIC (RoC)	12	32	64	66 dBc	1.1 GHz	0.011 mm^2
CORDIC (ORA)	12	32	73	78 dBc	680 MHz	0.054 mm^2

Table 6.3: Summary of DDFS Designs

(z_i). This is because we are comparing a 2^{-i} number, which is a value represented by a single bit.

Table 6.3 summarizes the size of the various lookup tables implemented and fabricated. In all cases, the SFDR and SNR of the DAC where several orders of magnitude worse than the digital code word produced by the DCDO. Therefore the frequency is a measured operating frequency, but the SINAD and SFDR are HDL simulated DCDO outputs.

6.5 Stretch Processing DDFS Architecture

One pulse compression technique for which a DDFS with LFM is well suited is stretch processing [42]. In fixed chirp rate stretch processing, a high bandwidth linear chirp of chirp rate (α) with a fixed time length (T_{TX}) is transmitted into the environment. The transmit period times the chirp rate yields the effective bandwidth of the transmitted chirp (β_{TX}). This bandwidth sets the range resolution (i.e. the radars ability to uniquely distinguish closely spaced targets). A first order approximation of the range resolution (R_{RES}) of a stretch processing system is given in Equation 6.25.

During reception of the signal reflected from a target, a "destretch" signal of the same chirp rate α as the transmitted signal, but with a longer time duration (T_{RX}) and consequently wider bandwidth, is used to demodulate the signal. The difference in the time duration between the transmit and receive chirps sets the range interval of the radar. The range interval is the "window" through which the radar can detect objects. It is the bandwidth of the destretch signal (β_{RX}), and not the transmitted pulse, that sets the system bandwidth requirement for the DDFS.

Figure 6.24 shows the top level DDFS architecture used for the radar system. The components of radar directly related to this dissertation are the DDFS and corresponding control circuitry. Two DACs for in-phase and quadrature phase sinusoid generation are implemented in the DDFS system.

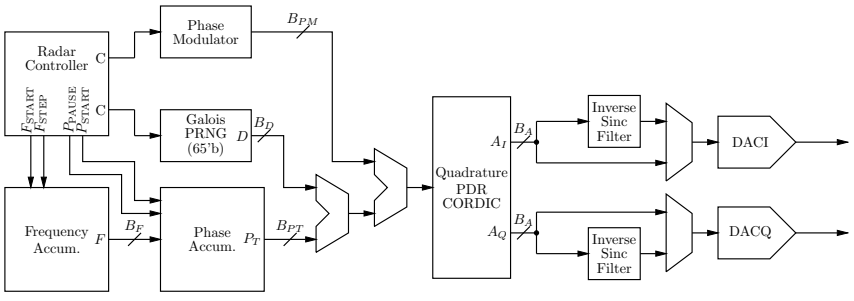
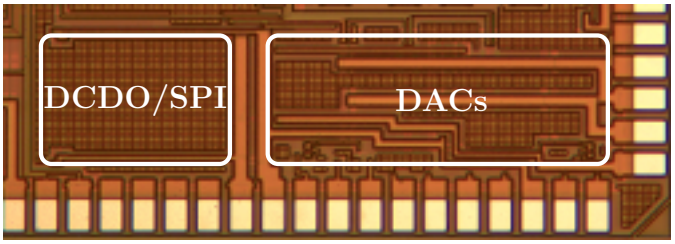


Figure 6.24: Block Diagram for Radar DDFS

Figure 6.25: Die Photograph of RoC (DDFS Zoomed)



The additive dithering technique that was used in both the MTM DDFS and BTM DDFS was also employed for the RoC DDFS. However, the phase truncation spurs rested so far beneath the noise floor of the DAC, that no spurious improvement was detected. Figure 6.25 is the die photo of the RoC chip zoomed around the DDFS. The DAC and DCDO/SPI are labelled to help make sense of the silicon.

6.5.1 Inverse Sinc Filter

The high speed DAC implementation used in the DDFS inherently applies a zero order hold (ZOH) operation on the output waveform. The ZOH transfer function is actually a sinc function in the frequency domain and its reason for existence is described in Section 7.2. As stated in Section 6.5, the DDFS generates a wide bandwidth "destretch" chirp signal that performs the pulse compression step of the radar. It is important that the amplitude of the generated chirp not fluctuate with frequency (and hence distort the radar measurement). One solution [56] is to apply an inverse sinc operation using a finite impulse response (FIR) filter to shape the waveform before sending it to the DAC.

In the DDFS, two FIR filters with 9-bit coefficient resolution, one for the I-path and one for the Q-path, were implemented after the PDR CORDIC. Pipelining the FIR filter was essential to allow it to reach 1 GHz operation. Figure 6.26 shows a block diagram of the inverse sinc filter component as implemented, where $c_0 = -1$, $c_1 = 4$, $c_2 = -16$, and $c_3 = 192$ for nine bit coefficient resolution. Note that after each addition the results are stored in a

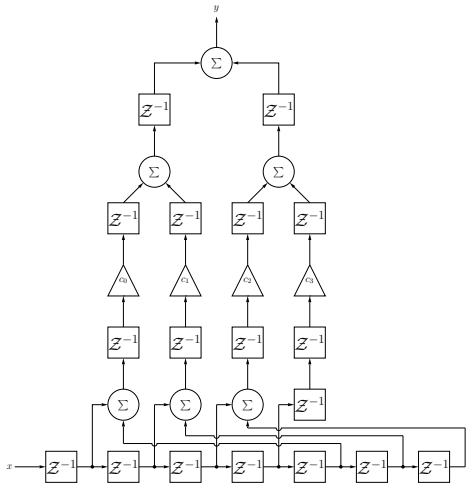


Figure 6.26: Inverse Sinc FIR Filter (Block Diagram)

pipeline register. The coefficients from Samueli's work were verified to be optimal in a min-max sense using a linear programming (LP) algorithm. An LP algorithm for finding optimal coefficients was developed with Python programming. The coefficients where found to be in full agreement with previously published work [56].

The measured results match the theory quite well (Figure 6.31a). To test the filters, a 90 MHz sweep with the clock frequency at 200 MHz was generated using the DDFS. Four markers are placed equally across the waveform and the amplitude was measured. With the inverse sinc filter deactivated, the measured values across the spectrum were -23.60 dBm, -24.03 dBm, -24.87 dBm and -26.37 dBm. This roll off indicates that the DACs do indeed exhibit ZOH behavior. Next, the same 90 MHz sweep with the clock frequency at 200 MHz was generated with the inverse sinc filter active. The four markers now read -27.24 dBm, -27.17 dBm, -27.09 dBm and -27.30 dBm. Here the gain variation is less than 0.21 dBm.

6.5.2 Radar Controller

The DDFS is tightly integrated with a digital radar controller. Several default modes of operation are programmable for the DDFS depending on the radar operating environment. The default mode is stretch processing mode for longer range target acquisition. A BPSK

mode is available for detecting objects close in to the radar with built-in Barker code modulation schemes. This mode is required because the RoC chip must operate in half-duplex mode from a single antenna and thus a long chirp would prevent detection of close in targets. There are also QPSK and general LPM modes for experimenting with different detection techniques in the lab. Several device characterization and test modes for the DACs, filters and DCDO were also implemented such as a single tone mode.

The basic operation of the radar transceiver in stretch processing mode and the algorithm is described as follows:

1. Initialize common analog components such as the PLL and bandgaps.
2. Deactivate the analog receiver circuitry.
3. Activate the analog transmitter circuitry.
4. Load the transmitter frequency control words into the start frequency, stop frequency and step frequency DDFS registers.
5. Clock the start frequency state into the frequency accumulator. Clock the start phase state into the phase accumulator.
6. Run the DDFS until the transmitter stop frequency control word is reached.
7. Load the transmitter timer, store the old receiver wait time and start waiting.
8. While waiting, activate the analog receiver circuitry.
9. Deactivate the analog transmitter circuitry.
10. Load the receiver frequency control words into the start frequency, stop frequency and step frequency DDFS registers.
11. Run the DDFS until the receiver stop frequency control word is reached.
12. Load the receiver timer, store the previous transmitter wait time, and start waiting.

13. Proceed to state (2).

The LPM modes operate with a similar algorithm except all the LFM behavior is deactivated.

6.6 Design of 12-bit CMOS DAC

Two fully differential, current steering 12-bit, 1 GHz CMOS DACs convert the digital output of the DCDO to a voltage. The DACs use a segmented architecture with 6-bits of thermometer coding for the MSB and 6-bits of binary coding for the LSB. Figure 6.27 is a block diagram of the DAC. The output of the DAC has 20 dB of digitally controlled, programmable gain. The gain is programmed by modifying the values of current reference and thus reducing the magnitude of the current through the current steering switches. This reduces the DACs operating frequency, which is described in Section 7.5.1.

The DAC uses a triple-centroid switching scheme made popular in [57], that randomizes spurs due to current cell mismatch. The clock tree is a balanced H-tree in an attempt to minimize clock skew along the wire paths. A single base transistor size was chosen such that

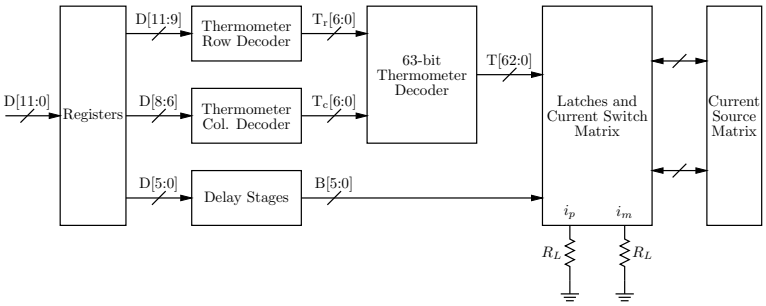


Figure 6.27: Block Diagram of 12-Bit CMOS DAC

the variation exhibited through Monte Carlo simulations kept the DAC DNL within bounds. Figure 6.28 demonstrates how the single unit transistor is used to build the current source network. [ht] The thermometer coded current sources have a cascade transistor added to increase the output impedance of the DAC. The DAC also implements custom high speed latches that convert the single ended digital input to a differential signal. These latches aid

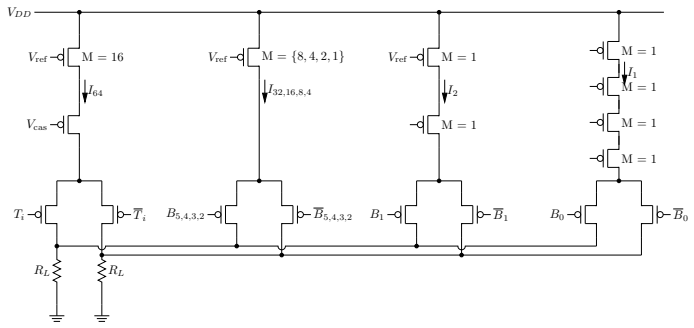


Figure 6.28: DAC Current Source Sizing

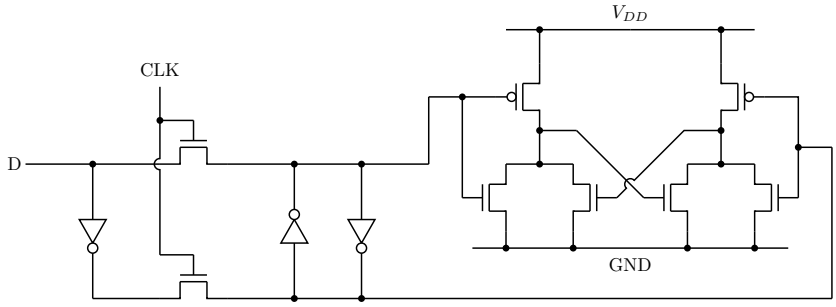


Figure 6.29: Synchronization Circuit for 12-Bit CMOS DAC

in synchronizing the the digital bits sent to the DAC from the synthesized digital component. The synchronization reduces timing mismatch and consequently the improves the spurious response of the DAC. The total area of the DAC, including the digital front-end, is 400 μm $\times$ 500 μm. The SFDR of the DAC is approximately 55 dBc (better than 60 dBc at certain frequencies) through about two thirds of the Nyquist frequency. The measured narrowband noise, where narrowband is measured before the third harmonic of the fundamental tone, is approximately 90 dBc.

The DAC high speed clock distribution tree makes use of an H-tree network as shown in Figure 6.30. This technique is used to equalize the amount of static delay between current steering cells on the clock distribution network. Any statis mismatch will directly translate into deterministic spurs when a periodic varying signal drives the DAC.

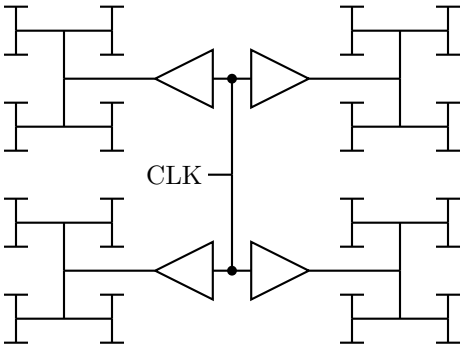


Figure 6.30: Clock Tree for 12-Bit CMOS DAC

6.7 Measurements

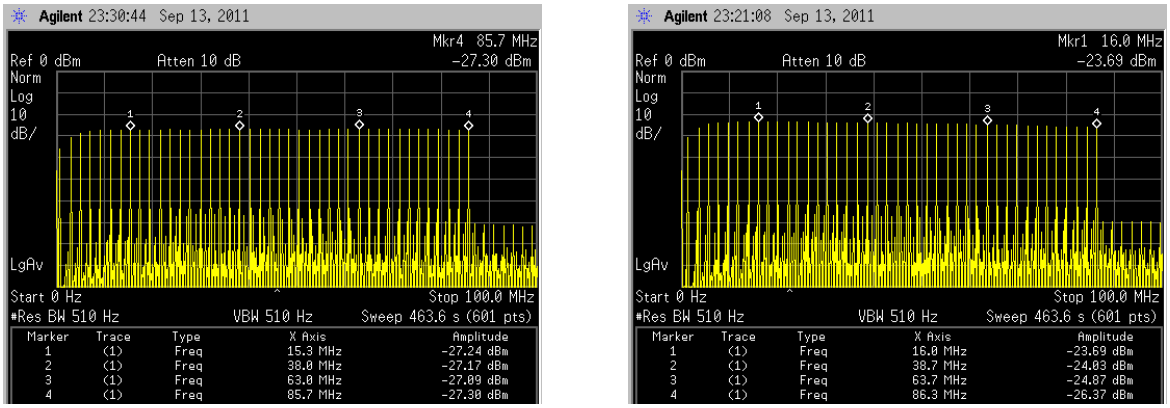
The performance of DDFS is summarized in Table 6.4. From a previous implementation, the research has shown that the digital component can run without errors up to 1.1 GHz. However, due to process manufacturing issues with this particular run, the system could only run at 650 MHz. Another version will be resubmitted without modification that should allow it to reach the proper operating frequency. The static DNL and INL of DAC cannot

Table 6.4: DDFS Performance Summary

Parameter	Value
f_{clk}	650 MHz
SFDR (low)	55 dBc (at 1.26 MHz)
SFDR (mid)	60 dBc (at 88 MHz)
SNR (narrowband)	91 dBc
Power (Analog)	150 mW
Power (Digital)	700 mW
DAC Area	400 μm $\times$ 500 μm (2X)
Digital Area (inc. SPI/control logic)	400 μm $\times$ 800 μm

be measured as the inputs of the DAC are not accessible from the pad frame of the chip. The SFDR however indicates that the DAC performs poorly in INL. This is likely due to the programmable gain stage of the DAC, as the third order harmonic was strongly dependent on the gain state.

Lastly, Figure 6.31a shows a 100 MHz chirp with the inverse sinc filter activated. The attenuation at low frequencies is from the test setup, in which the output of the packaged RoC chip is AC coupled to the spectrum analyzer. Figure 6.31b shows the waveform without the inverse sinc filter activated. Because of the scale of the waveform it is difficult visually determine the impact of the filter on the waveform. However, looking at the readings from the markers, the impact of the inverse sinc filter is more easily understood. The output spectrum when the inverse sinc filter is activated does not attenuate at higher frequencies. The deviation between the waveforms becomes even more pronounced as the output of the DAC approaches the Nyquist frequency.



(a) Chirp with Inverse Sinc On

(b) Chrip with Inverse Sinc Off

Figure 6.31: Inverse Sinc Filter

Figure 6.32 shows the DDFS operating in single tone mode. The main tone is located at 85.8 MHz and the largest spur is the fourth order harmonic of the main tone located at 343.2 MHz and is 57 dB down. This tone is measured before the low pass filter. This work briefly describes a fully functional DDFS for stretch processing radar applications. The performance of the digital logic of the DDFS is competitive with other published DDFS implementations given the feature size of the technology, almost certainly when frequency resolution and features are considered.

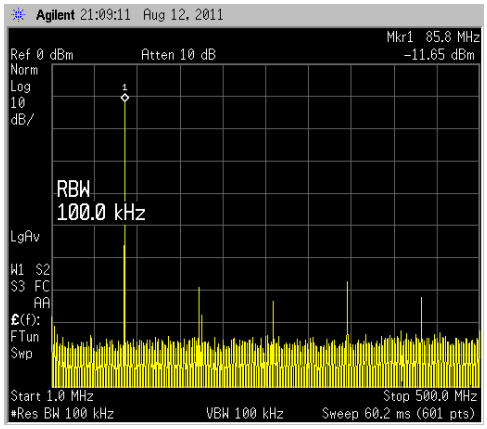


Figure 6.32: DDFS with Single Tone Output

Chapter 7

Digital-To-Analog Converters (DAC)

As digital circuitry evolves to smaller geometry nodes that result in higher speeds, lower power and smaller area, more functionality is relegated to the digital processing domain. The transition from the discrete-time digital domain to the continuous-time analog domain is performed by a Digital-to-Analog Converter (DAC). In modern designs, the performance of the DAC dominates the performance of the DDFS [4], since small feature size CMOS allows spectrally pure digital sinusoids to be generated with little overhead at sufficiently high speeds. It is not uncommon to have a digital SFDR and SNR that are 10 to 20 dB better than what can be achieved by a DAC in the same technology.

The term DAC is general and includes small devices that tune static circuit parameters to massive RF DACs as those designed by Analog Devices [58] or e2v [43]. The DACs discussed in this thesis are > 1 GSample/s current steering (CS) designs. The design issues discussed include clock and data timing errors and frequency dependent non-linearities that do not plague DC DACs. However, the discussion on current source mismatch, static INL, static DNL and segmented architectures are relevant to DC CS DACs as well as high speed CS DACs.

7.1 Basic Sampling Theory

The DDFS is a sampled-data system, and thus a brief explanation of the sampling process is beneficial in understanding the behavior of the device. It will also benefit later analysis in Section 7.2 in which different DAC switching schemes are discussed. An important operator must be defined to aid in the discussion of sampling theory, namely, the Dirac

Delta "function." Here the required mathematical theory to formalize the Dirac delta as a distribution is ignored and a more axiomatic treatment is provided [59].

Definition 7.1 (Dirac Delta)**.** *The* ***Dirac delta*** *in a one-dimensional real space can be defined as a heuristic function, symbollicaly denoted as* $\delta(x)$*, such that*

$$\delta(x) \triangleq \begin{cases} +\infty, & x = 0 \\ 0, & x \neq 0 \end{cases} \tag{7.1}$$

and satisfies the identity given in Equation 7.2

$$\int_{-\infty}^{\infty} \delta(x)\, dx = 1 \tag{7.2}$$

Of course, the following "proofs" about the properties of the Dirac delta all leave something to be desired, as the definition of the Dirac delta used in this work cannot be considered mathematically formal. However, if one assumes that the Dirac delta operates similar to a function within an integral, then these "proofs" hold. Now consider the behavior of the Dirac delta when used in an integral, as its usefulness in mathematically describing the sample operation becomes important.

$$\int_{-\infty}^{\infty} x(t)\delta(t)dt = \int_{-\infty}^{\infty} x(0)\delta(t)dt = x(0)\int_{-\infty}^{\infty} \delta(t)dt = x(0) \tag{7.3}$$

Equation 7.3 holds assuming, of course, that $x(t)$ is defined at zero. The integral of a function multiplied by the Dirac delta takes on the value of the function at which the Dirac delta is non-zero. Ideal sampling "captures" the value of a continuous function at an instant in time, which certainly sounds similar to the mathematical operation of the Dirac delta.

If one wishes to acquire the value of $x(t)$ once every T seconds, or sample the signal $x(t)$ with an interval of T, then a series of Dirac deltas equally spaced in time is needed. Let us

then define a "pulse train" of Dirac deltas as follows,

$$\Delta_T(t) \triangleq \sum_{k=-\infty}^{\infty} \delta(t - kT) \tag{7.4}$$

where T is the period of sampling and k is an integer. It is interesting to note that Δ_T is periodic with T. This can be shown as by proving that $\Delta_T(t) = \Delta_T(t + nT)$.

$$\Delta_T(t + nT) = \sum_{k=-\infty}^{\infty} \delta(t + nT - kT) \tag{7.5}$$

$$= \sum_{k=-\infty}^{\infty} \delta(t + (n - k)T) = \Delta_T(t) \tag{7.6}$$

The last step is possible because the summation limits tend to infinity. So shifting the sequence a finite number of T in any direction results in the same funtion. Since $\Delta_T(t)$ is periodic with T, it can represented by its Fourier series. The real valued Fourier series was described by Definition 2.1. Here the complex Fourier series is used to simplify notation.

$$\Delta_T(t) = \sum_{n=-\infty}^{\infty} \left(\frac{1}{T} \int_{-T/2}^{T/2} \sum_{k=-\infty}^{\infty} \delta(t - kT)\, e^{-j2\pi nt/T} dt \right) e^{j2\pi nt/T} \tag{7.7}$$

$$= \sum_{n=-\infty}^{\infty} \left(\frac{1}{T} e^{0} \right) e^{j2\pi nt/T}$$

$$= \frac{1}{T} \sum_{n=-\infty}^{\infty} e^{j2\pi nt/T} \tag{7.8}$$

The summation over k in Equation 7.7 was dropped because the Dirac delta is only zero for $k = 0$ as t varies from $-T/2$ to $T/2$. Now the Fourier transform of $\Delta_T(t)$ can be computed in a straight-forward manner. This leads to one of the most interesting results in sampling

theory.

$$\begin{aligned}\mathcal{F}\left\{\Delta_T(t)\right\} &= \int_{-\infty}^{\infty}\left(\frac{1}{T}\sum_{n=-\infty}^{\infty}e^{j2\pi nt/T}\right)e^{-j2\pi ft}dt \\ &= \frac{1}{T}\int_{-\infty}^{\infty}\sum_{n=-\infty}^{\infty}e^{-j2\pi t(f-n/T)} \\ &= \frac{1}{T}\sum_{n=-\infty}^{\infty}\delta\left(f-n/T\right) \end{aligned} \tag{7.9}$$

So the Fourier transform of the Dirac comb function is another Dirac comb but in the frequency domain. The spacing of the impulses is the sampling frequency $(1/T)$ of the original sampling operation.

One of the theorems related to sampling, fundamentally important to DAC design and universally taught to electrical engineering students is the Nyquist-Shannon Sampling Theorem. The first publication of this powerful theorem as it relates to the field of communication is provided by Shannon in 1948 [60]. The Nyquist-Shannon sampling theorem as given by Bernard Widrow [61] is described in Theorem 7.1.

Theorem 7.1 (Nyquist-Shannon Sampling Theorem)**.** *If the sampling radian frequency Ω_s is high enough so that*

$$\left|X\left(j\omega\right)\right| = 0 \quad for \quad |\omega| \geq \frac{\Omega_s}{2} \tag{7.10}$$

where $X(j\omega)$ is the CTFT of $x(t)$ then the sampling condition is met, and $x(t)$ is perfectly recoverable from its samples.

In more common parlance, Theorem 7.10 states that a bandlimited signal $x(t)$ can be perfectly reconstructed from its samples if the sample rate is at least twice the bandwidth. In the following section, the foundations of the Nyquist-Shannon sampling theorem will be built.

7.2 DAC Fundamentals

As part of this work, we will briefly discuss the fundamentals of DAC behavior. A DAC transforms a digital code word into a physical, electrical quantity. Typically this electrical quantity is a voltage (i.e. low impedance output) or a current (i.e. high impedance output). After the DAC generates the physical quantity, it is filtered by a low pass analog reconstruction filter or something that approximates such a filter. This need not be the case however, as output signals with frequencies higher than the first Nyquist zone can be synthesized by applying a bandpass filter to DACs with certain types of responses.

The following analysis makes heavy use of convolution. So in keeping with the spirit of this thesis, it is presented here along with one of its more important properties (Theorem 7.2) [59].

Definition 7.2 (Convolution)**.** *The convolution of $f(t)$ and $g(t)$, denoted $(f \star g)(t)$, is defined mathematically as*

$$(f \star g)(t) = \int_{-\infty}^{\infty} f(\tau)\, g(t-\tau)\, d\tau \tag{7.11}$$

Theorem 7.2 (Fourier Convolution Theorem)**.** *Let $x(t)$ and $y(t)$ be continuous function of t, then*

$$\mathcal{F}\{(x \star y)(t)\} = \mathcal{F}\{x(t)\} \cdot \mathcal{F}\{y(t)\} \tag{7.12}$$

$$\mathcal{F}\{x(t) \cdot y(t)\} = \mathcal{F}\{x(t)\} \star \mathcal{F}\{y(t)\} \tag{7.13}$$

This is generally referred to as the convolution theorem.

Proof. Let $x(t)$ and $y(t)$ be continuous functions whose Fourier transform exists. Then the Fourier transform of the convolution of $x(t)$ and $y(t)$ is

$$\begin{aligned}\mathcal{F}\left\{\left(x \star y\right)(t)\right\} &= \int_{-\infty}^{\infty} \left(x \star y\right)(t) e^{-j\Omega t} dt \\ &= \int_{-\infty}^{\infty} \left(\int_{-\infty}^{\infty} x\left(\tau\right) y\left(t-\tau\right) d\tau \right) e^{-j\Omega t} dt \end{aligned} \tag{7.14}$$

The order of integral operations can be rearranged provided that Fubini's theorem [62] is satisfied by the double integral of Equation 7.14 to yield

$$\mathcal{F}\left\{\left(x \star y\right)(t)\right\} = \int_{-\infty}^{\infty} x\left(\tau\right) \left(\int_{-\infty}^{\infty} y\left(t-\tau\right) e^{-j\Omega t} dt \right) d\tau \tag{7.15}$$

Substituting $r = t + \tau$ for t into Equation 7.15 and noting that $dt = dr$ gives the final result

$$\begin{aligned}\mathcal{F}\left\{\left(x \star y\right)(t)\right\} &= \int_{-\infty}^{\infty} x\left(\tau\right) \left(\int_{-\infty}^{\infty} y\left(r\right) e^{-j\Omega r + -j\Omega \tau} dr \right) d\tau \\ &= \left(\int_{-\infty}^{\infty} x\left(\tau\right) e^{-j\Omega \tau} d\tau \right) \left(\int_{-\infty}^{\infty} y\left(r\right) e^{-j\Omega r} dr \right) \\ &= \mathcal{F}\left\{x(t)\right\} \cdot \mathcal{F}\left\{y(t)\right\} \end{aligned} \tag{7.16}$$

□

The ideal DAC response is a series of weighted impulses [63]

$$y_{\text{IDEAL}}(t) = \sum_{n=-\infty}^{\infty} \alpha x\left[n\right] \cdot \delta\left(t - nT\right) + \beta \tag{7.17}$$

where $\delta(t)$ is the Dirac delta function defined in Equation 7.1, α is the gain of the DAC (see Section 7.3.1) and β is the offset of the DAC (also see Section 7.3.1). Here $x[n]$ is understood to be the value of the signal x at time nT and thus $x(nT) = x[n]$. As the DAC metrics considered in this work are measures of the effects of non-linearity on the input signal, linear terms α and β can be ignored by setting $\alpha = 1$ and $\beta = 0$. Note that Equation 7.17 can

be derived from the multiplication of the Dirac comb of Equation 7.4 in Section 7.1 and ignoring the DC offset β and linear gain.

$$\begin{aligned} x(t) \cdot \Delta_T(t) &= x(t) \sum_{n=-\infty}^{\infty} \delta\left(t - nT\right) \\ &= \sum_{n=-\infty}^{\infty} x(t)\delta\left(t - nT\right) \end{aligned} \tag{7.18}$$

In communications, the spectrum of the generated signals is critical and as DACs are the central actuators for such systems, it is also critical in the analysis of this chapter. Thus the mathematical tool for computing spectrum of continuous functions is presented. The continuous time Fourier transform (CTFT) is the given in Equation 7.19,

$$X\left(\Omega\right) = \int_{-\infty}^{\infty} x\left(t\right) e^{-j\Omega t} dt \tag{7.19}$$

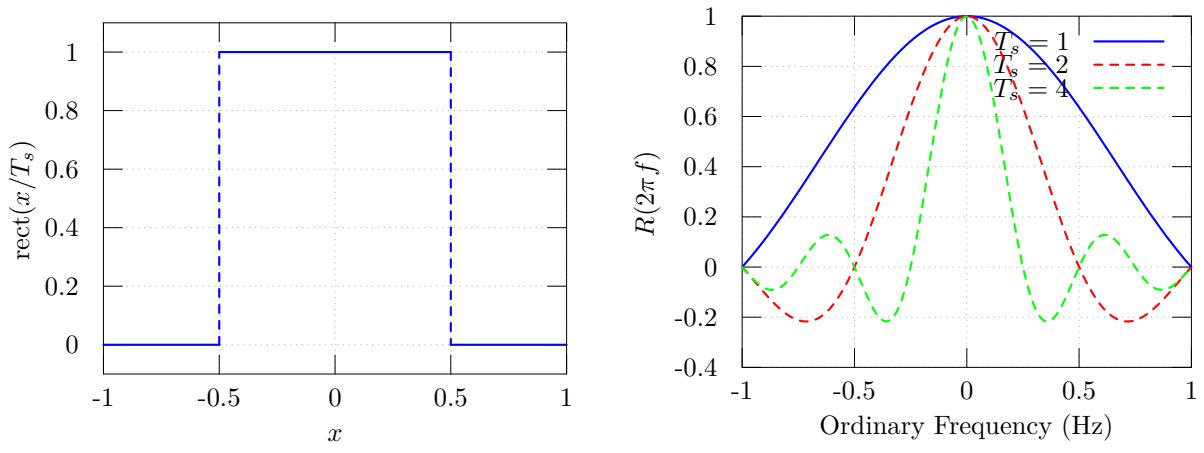
$$X\left(f\right) = \int_{-\infty}^{\infty} x\left(t\right) e^{-j2\pi f t} dt \tag{7.20}$$

where $x(t)$ is a complex-valued function of time, $\Omega \in \mathbb{R}$ is the continuous-time angular frequency and $X\left(\Omega\right)$ is the transformed signal and is generally complex. The second equation describes the Fourier transform as a function of the ordinary frequency, f. The CTFT, as with other variants of the Fourier transform, is invertible. The inverse (backwards) transform is given by Equation 7.21.

$$x\left(t\right) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X\left(\Omega\right) e^{j\Omega t} d\Omega \tag{7.21}$$

As an example, consider the following rectangle function (Equation 7.22, also known as the normalized box car function. Figure 7.1a shows the time domain response of the the rectangle function.

$$\text{rect}(x) = \begin{cases} 0 & |x| > 0.5 \\ 0.5 & |x| = 0.5 \\ 1 & |x| < 0.5 \end{cases} \tag{7.22}$$



(a) Rectangle Function (Time)

(b) Rectangle Function (Spectrum)

Figure 7.1: Rectangle Function Plots

Since we will soon be discussing sampling, we will scale the rectangle function to a single sample period T_s. We can then apply the continuous time Fourier Transform (7.19) on the modified rectangle function.

$$\begin{aligned} R(\Omega) &= \int_{-\infty}^{\infty} \text{rect}\left(\frac{t}{T_s}\right) e^{-j\Omega t} dt \\ &= \int_{-0.5T_s}^{0.5T_s} 1 e^{-j\Omega t} dt \\ &= \frac{1}{-j\Omega}\left(e^{-j\Omega t}\right)\Big|_{-0.5T_s}^{0.5T_s} \\ &= \frac{1}{-j\Omega}\left[e^{-0.5j\Omega T_s} - e^{0.5j\Omega T_s}\right] \end{aligned} \tag{7.23}$$

Applying Euler's formula (Equation 1.31) to Equation 7.23, the expression simplifies to a scaled *sinc* function.

$$\begin{aligned} R(\Omega) &= \frac{1}{-j\Omega}\left[e^{-0.5j\Omega T_s} - e^{0.5j\Omega T_s}\right] \\ &= \frac{1}{-j\Omega}\left[\cos\left(-T_s\frac{\Omega}{2}\right) + j\sin\left(-T_s\frac{\Omega}{2}\right) - \cos\left(T_s\frac{\Omega}{2}\right) - j\sin\left(T_s\frac{\Omega}{2}\right)\right] \\ &= \frac{1}{-j\Omega}\left[-2j\sin\left(T_s\frac{\Omega}{2}\right)\right] \\ &= \frac{\sin\left(T_s\frac{\Omega}{2}\right)}{\frac{\Omega}{2}} \end{aligned} \tag{7.24}$$

This analysis will become important in following paragraphs when the output spectrum of various DACs are considered. The unnormalized sinc function is defined as

$$\operatorname{sinc}(x) \triangleq \frac{\sin(x)}{x} \tag{7.25}$$

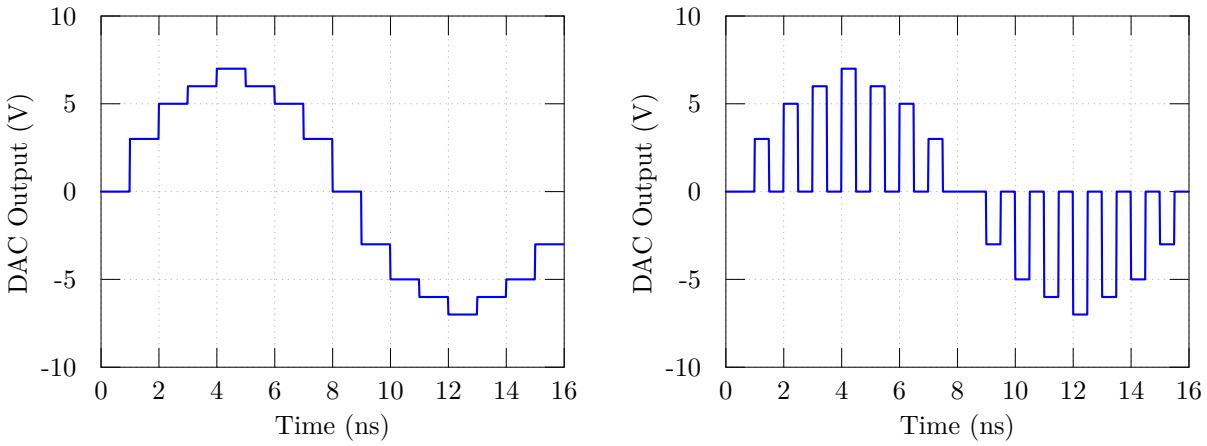
and is both non-causal and infinite. Setting $T_s = 1$, it follows that $R(\Omega) = \operatorname{sinc}(\Omega/2)$. Figure 7.1b shows the frequency response of the rectangular function for various T_s.

Now we compute the Fourier transform of the more complex $y_{\text{IDEAL}}(t)$ of Equation 7.17.

$$\begin{aligned} \mathcal{F}_{\text{CTFT}}\{y_{\text{IDEAL}}(t)\} &= \mathcal{F}\left\{\sum_{n=-\infty}^{\infty} x[n]\delta(t-nT)\right\} \\ &= \int_{-\infty}^{\infty}\sum_{n=-\infty}^{\infty} x[n]\delta(t-nT)\,e^{-j2\pi ft}dt \\ &= \int_{-\infty}^{\infty}\left[x[0]\delta(t) + x[1]\delta(t-T) + x[2]\delta(t-2T)\cdots\right]e^{-j2\pi ft}dt \\ &= x[0]e^{-j2\pi f} + x[1]e^{-j2\pi fT} + \cdots \\ &= \sum_{n=-\infty}^{\infty} x[n]e^{-j2\pi fnT} \end{aligned} \tag{7.26}$$

Notice that Equation 7.26, normalizing $T = 1$, is exactly the same as the DTFT shown in Equation 2.31. Thus what was stated as sampling in the time domain in words now has a mathematical representation.

The conventional CS DAC updates the data output with a new code word once every sampling interval and holds the value until the DAC is updated again. This is sometimes called a *zero-order hold* or a *sample and hold*. A DAC that implements such an hold is called a non-return-to-zero (NRTZ) DAC. Figure 7.2a provides an example of the time domain output of an NRTZ DAC. If the DAC returns to zero (RTZ) after waiting T_0 seconds, then the DAC is referred to an RTZ DAC. Figure 7.2b shows the output of an RTZ DAC with a 50% duty cycle, which is equivalent to setting $T_0 = T_s/2$ in Equation 7.33.



(a) Non-Return-to-Zero DAC Output (4 Bits) (b) Return-To-Zero DAC Output (4 Bits)

Figure 7.2: INL Curves for Thermometer-Coded DAC Models with Finite Output Impedance Current Sources

There are two methods for describing how the hold effect shapes the response of the DAC. As noted by Doris *et al.* [64], the NRTZ DAC response can be formulated as the convolution of the unit step response and a variation of the DAC response given in Equation 7.17,

$$y_{\text{NRTZ}}(t) = u(t) \star \sum_{n=-\infty}^{\infty} (x[n] - x[n-1])\, \delta\,(t - nT) \tag{7.27}$$

where $u(t)$ is the unit step response, also known as the Heaviside step function, and is defined in Equation 7.28.

$$u(t) = \begin{cases} 1, \ t \geq 0 \\ 0, \ t < 0 \end{cases} \tag{7.28}$$

An alternative representation noted by the author of this work is the convolution of a rectangular function (Equation 7.22) of the width the sample period with the ideal DAC response

$$y_{\text{NRTZ}}(t) = \text{rect}\left(\frac{t}{T_s}\right) \star \sum_{n=-\infty}^{\infty} x(t)\delta\left(t - nT_s\right) \tag{7.29}$$

Referring back to Theorem 7.2, convolution in the time domain is equivalent to multiplication in the frequency domain (Fourier transform domain). The Fourier transform of $\text{rect}(t)$ was calculated in Equation 7.24. The right-hand term is simply the ideal output spectrum of an ideal DAC (or the DTFT of the sequence synthesized by the DAC).

$$\mathcal{F}_{\text{CTFT}}\left\{y_{\text{NRTZ}}(t)\right\} = \mathcal{F}\left\{\text{rect}\left(\frac{t}{T_s}\right)\right\} \cdot \mathcal{F}\left\{y_{\text{IDEAL}}(t)\right\} \tag{7.30}$$

$$= \left(\frac{\sin\left(\frac{T_s\Omega}{2}\right)}{\frac{\Omega}{2}}\right) \cdot \mathcal{F}\left\{y_{\text{IDEAL}}(t)\right\} \tag{7.31}$$

The output then has a weight sinc filtered output response. This attentuation was the reason for the inverse sinc filter of radar DDFS, described in Section 6.5.1.

Likewise, an RTZ DAC output response can be formulated as the convolution of the unit step and two time shifted Dirac delta functions [64]

$$y_{\text{RTZ}}(t) = u(t) \star \sum_{n=-\infty}^{\infty} x[n]\left(\delta\left(t - nT\right) - \delta\left(t - T_0 - nT\right)\right) \tag{7.32}$$

or, again, as the convolution of a rectangular function of width $T_0 < T_s$ (typically $T_s/2$ in literature).

$$\mathcal{F}_{\text{CTFT}}\left\{y_{\text{RTZ}}(t)\right\} = \mathcal{F}\left\{\text{rect}(t/T_0)\right\} \cdot \mathcal{F}\left\{y_{\text{IDEAL}}(t)\right\} \tag{7.33}$$

$$= \left(\frac{\sin\left(\frac{T_0\Omega}{2}\right)}{\frac{\Omega}{2}}\right) \cdot \mathcal{F}\left\{y_{\text{IDEAL}}(t)\right\} \tag{7.34}$$

From Figure 7.1b, the *shorter* the pulse width, the *less* attenuation of the output spectrum due to value holding. This means that the inverse sinc filter requirement can be removed or the order of the filter reduced by changing the DAC output characteristic.

To quantify the effect, let $T_0 = T_s/2$,

$$\frac{\mathcal{F}\left\{y_{\text{RTZ}}(t)\right\}}{\mathcal{F}\left\{y_{\text{NRTZ}}(t)\right\}} = \frac{\sin\left(T_0\frac{\Omega}{2}\right) y_{\text{IDEAL}}(t)}{\frac{\Omega}{2}} \cdot \frac{\frac{\Omega}{2}}{\sin\left(T_s\frac{\Omega}{2}\right) y_{\text{IDEAL}}(t)} = \frac{\sin\left(T_0\frac{\Omega}{2}\right)}{\sin\left(T_s\frac{\Omega}{2}\right)} \tag{7.35}$$

Performing a Taylor series expansion (Definition 4.1) on the numerator and denominator

$$\frac{\sin\left(T_0\frac{\Omega}{2}\right)}{\sin\left(T_s\frac{\Omega}{2}\right)} = \frac{\frac{T_0\Omega}{2} - \frac{T_0^3\Omega^3}{8\cdot 3!} + \frac{T_0^5\Omega^5}{32\cdot 5!} - \cdots}{\frac{T_s\Omega}{2} - \frac{T_s^3\Omega^3}{8\cdot 3!} + \frac{T_s^5\Omega^5}{32\cdot 5!} - \cdots} \tag{7.36}$$

$$= \frac{T_0 - \frac{T_0^3\Omega^2}{4\cdot 3!} + \frac{T_0^5\Omega^4}{16\cdot 5!} - \cdots}{T_s - \frac{T_s^3\Omega^2}{4\cdot 3!} + \frac{T_s^5\Omega^4}{16\cdot 5!} - \cdots} \tag{7.37}$$

Now we substitue $T_s = 2T_0$ into the previous equation and compute the final result.

$$\frac{\mathcal{F}\left\{y_{\text{RTZ}}(t)\right\}}{\mathcal{F}\left\{y_{\text{NRTZ}}(t)\right\}} = \frac{T_0 - \frac{T_0^3\Omega^2}{4\cdot 3!} + \frac{T_0^5\Omega^4}{16\cdot 5!} - \cdots}{2T_0 - \frac{8T_0^3\Omega^2}{4\cdot 3!} + \frac{32T_0^5\Omega^4}{16\cdot 5!} - \cdots} \tag{7.38}$$

$$= \frac{1 - \frac{T_0^2\Omega^2}{4\cdot 3!} + \frac{T_0^4\Omega^4}{16\cdot 5!} - \cdots}{2 - \frac{2T_0^2\Omega^2}{3!} + \frac{2T_0^4\Omega^4}{5!} - \cdots} \tag{7.39}$$

But T_0 and T_s are generally much less than one, a 1 GHz DAC would has a 1 ns clock period. Therefore the attenutation is approximately 1/2 when returning to zero. Note that if T_0 becomes sufficiently small higher Nyquist zones of the DAC can be used.

7.3 DAC Performance Metrics

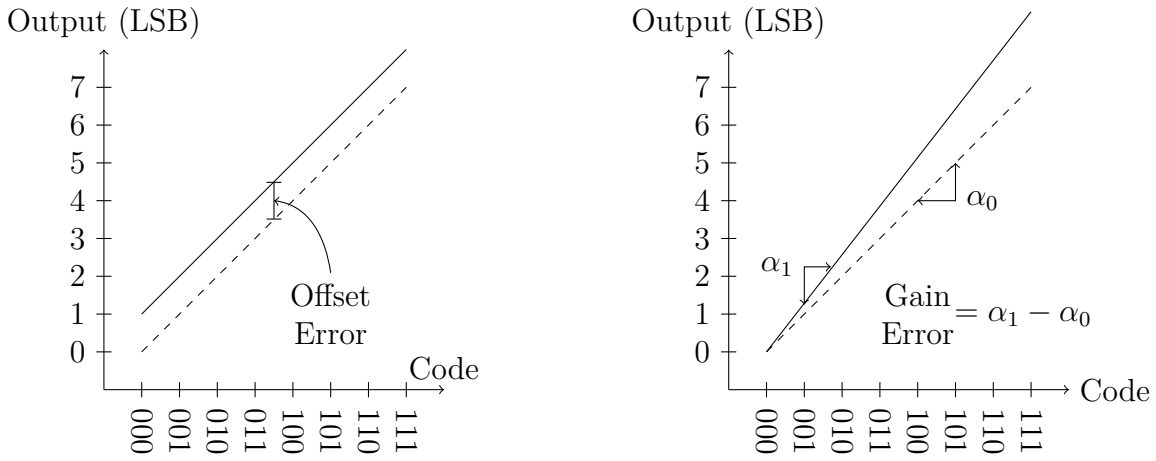
DACs operate in a wide range of environments with an equally wide range of requirements. A control DAC for a microelectromechanical system (MEMS) may need only operate at a few kilohertz sample frequency but may require a large output voltage and monotinicity over process variation. A DAC for a high-speed communications link may only require a few hundred millivolts of output swing but may have to operate up to several gigahertz. The qualities of both these DACs can be described by their *static* and *dynamic* performance. The remaining analysis of this chapter aims to aid designers in creating high performance DACs.

When writing about "high performance" devices, we want to be precise in the describing the measure of that performance. For the purposes of this dissertation, the static measures of concern are integral non-linearity, differential non-linearity, and static power consumption. The dynamic measures of concern are spurious free dynamic range, signal to noise ratio, sample frequency and total harmonic distortion. These performance metrics are influenced by a wide variety of effects.

7.3.1 Static DAC Performance

Static errors are both the simplest DAC design errors to understand and the simplest to correct. They therefore serve as an adequate starting place for DAC performance analysis. It is entirely possible to degrade the overall performance of the DAC by failing to weight individual DAC elements such that the contribution of static errors to the output are zero. This can be particularly bad in high speed CS DAC designs, where device sizing is small and device mismatch becomes significant. The two static performance number discussed in this chapter are Integral non-linearity (INL) and Differential non-linearity (DNL). Both these errors are direct causes for harmonic distortion in the DACs.

While not nearly as important in high-speed CS DACs as non-linear errors, two linear errors are mentioned for completeness. *Offset error* is defined as the linear deviation of the DAC output from the intended output applied to every DAC code. Figure 7.3a presents



(a) Offset Error (3 Bits)

(b) Gain Error (3 Bits)

Figure 7.3: Graphical Explanation of Gain and Offset Errors

a graphical explanation of offset error in a DAC. *Gain error* is defined as the deviation in the gain of the DAC versus the intended design target. Figure 7.3b provides a graphical explanation for gain error. Note that neither gain or offset errors contribute to the spurious performance of the DAC.

7.3.2 INL

Integral non-linearity (INL) is a measure of the deviation of the *static* transfer function of the DAC from some ideal linear transfer function. The measure is generally normalized to the LSB value of the DAC for reporting in publications. The two most widely used methods for determining the ideal linear transfer characteristic are end-point to end-point and least-squares linear ("best") fit. Both of these techniques compensate for linear errors, typically gain and offset error, as required by the IEEE definition for INL [65]. The following is the official description of INL for an N-bit DAC.

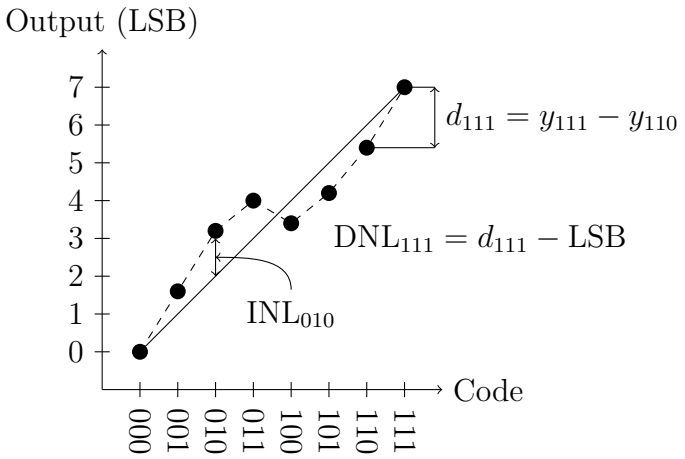
$$\mathrm{INL}[k] = \frac{I_{\mathrm{out}}[k] - k \cdot I_{\mathrm{lsb}}}{I_{\mathrm{lsb}}}; \quad I_{\mathrm{lsb}} = \frac{I_{\mathrm{out}}[2^N - 1]}{(2^N - 1)} \tag{7.40}$$

where $I_{\text{out}}[2^N - 1]$ is the maximum output of the DAC (i.e. assumes that the DAC output increases in magnitude with increasing k) and I_{lsb} is the LSB step of the DAC using the end-point to end-point line approximation. For the equation of a line, we use:

$$y = mx + b \tag{7.41}$$

where m is the slope of the line and b is the y-intercept of the line. For the DAC, x is the code word of the DAC, m is the LSB value of DAC used in our INL and DNL equations and b is the offset error. Figure 7.4 provides a graphical explanation of INL using a 3-bit DAC. The solid line is an end-point to end-point fitted line, the solid dots are the actual DAC output values, the x-axis is the DAC code input. So the solid line from Equation 7.41

Figure 7.4: Graphical Explanation of INL and DNL



can be rewritten in a discrete form as:

$$I_{\text{out}}[n] = I_{\text{lsb}}A[n] + b \tag{7.42}$$

where $A[n]$ is the DAC input code word at the n^{th} sample. This is in keeping with the output code from the DCDO in previous chapters. The end-point to end-point method takes the difference of the output of the DAC at the maximum and minimum output code words and

then adjusts out the offset error. The linear least squares fit, described in detail in Section 6.1.3, finds the line that minimizes the mean square error between itself and the actual DAC output data points.

Several common sources of static INL, and DNL for that matter, in CS DACs have been described in literature.

- Finite output impedance of the DAC current sources [66], [67].
- Mismatch in current sources due to local process variation, transistor and resistor mismatch, etc. [68]
- Voltage and temperature dependent resistive load variation (particularly if a polycrystalline silicon resistor [69],[66] is used).

Static INL degrades the spectral purity of the generated signal. This can be demonstrated by applying a sinusoid through the DAC transfer characteristic. This will be performed in subsequent sections. In order to start quantifying the static error for current steering DACs, static DAC models must be created.

7.3.3 DAC Models

Performing a first order analysis of a current steering DAC provides insight into a few of the major error sources that arise from the basic architecture. Figure 7.5 shows a single-ended, binary weighted current-mode DAC architecture with quite a few assumptions and simplifications. There are no output frequency dependent impedances as no load or source capacitance is considered, for the moment ignore C_L is the diagram. The switches are ideal and switch instantaneously. The on resistance of the current source, r_o, is assumed to scale linearly with the increasing current. We will see that even some of the simplest models of the CS DAC have a non-linear transfer function and thus produce a spurious response when driven with a periodically repeating input. In Figure 7.5, R_L is the load resistance, I_u is the LSB value of the current source, b_i is the ith bit of the code word A that is fed to the DAC,

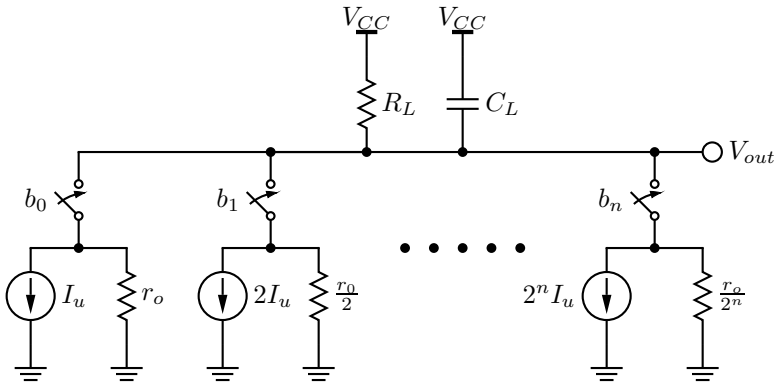


Figure 7.5: Simple Single-Ended Binary-Weighted Model

b_0 is the least significant bit (LSB) and b_n, where $m = B_A - 1$, is the most significant bit (MSB). We note that this model can be transformed to an equivalent thermometer coded DAC (Figure 7.6) and the transfer function analysis will remain the same. This is possible since the current sources add in parallel and the resistances scale in the binary model. For

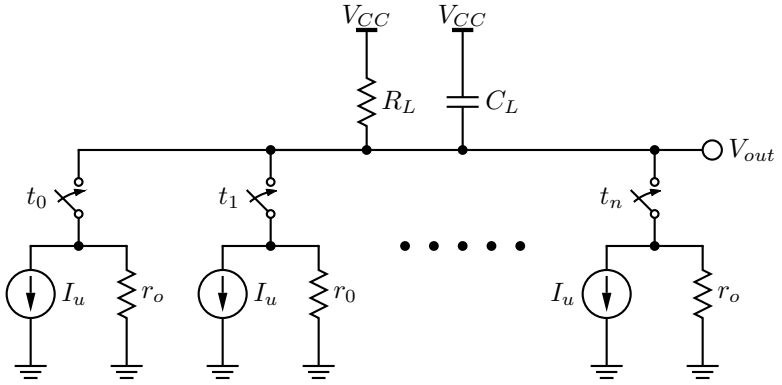


Figure 7.6: Simple Single-Ended Thermometer Model

Figure 7.6, the number of switches closed at sample n is equal to the value of $A[n]$ from Equation 7.42. The total number of switches is $N_A = 2^{B_A} - 1$, where B_A is the number of bits required to represent in the DAC input code. Using this model, we can calculate the transfer function of the DAC model. When $A[n] = 0$, all of the bits, t_i are zero. For this exercise, a bit value of zero indicates an open switch. In that case, no current is drawn and the output $V_{out} = V_{CC}$.

Now consider the scenario when $A[n] = 1$. In this case, $t_0 = 1$ and $t_2 = \cdots = t_n = 0$. So the circuit in Figure 7.6 reduces to Figure 7.7. This is effectively one output impedance in

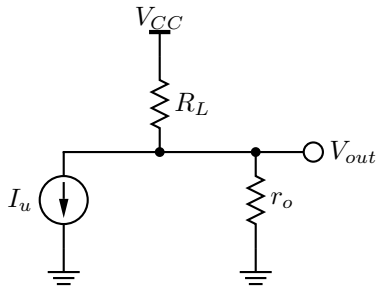


Figure 7.7: Single-Ended Single Bit Active

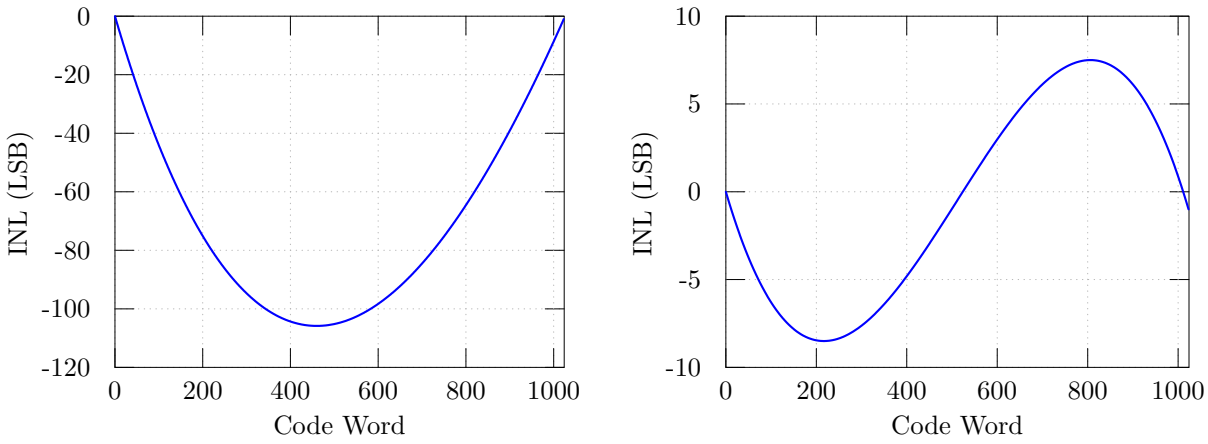
parallel with the resistive load. Now we note that as the switches close, the resistors combine in parallel. From circuit theory, the parallel combination of k resistors of the same value is R/k.

$$\begin{aligned} I_{out}[k] &= kI_u + \frac{V_{out}[k]}{r_o/k} \\ \frac{V_{CC} - V_{out}[k]}{R_L} &= kI_u + \frac{kV_{out}[k]}{r_o} \\ V_{out}[k] &= \frac{r_o V_{CC}}{r_o + kR_L} - \frac{kR_L r_o I_u}{r_o + kR_L} \end{aligned} \tag{7.43}$$

Figure 7.8a shows the INL of Figure 7.6 using Equation 7.43 for values of $B_A = 10$, $R_L = 50\,\Omega$, $r_0 = 100\,\mathrm{k}\Omega$, $I_u = 20\,\mu\mathrm{A}$ and $V_{CC} = 2\,\mathrm{V}$. Mathematically, the INL can be computed from Equation 7.43 as

$$\mathrm{INL_{SE}}\,[k] = \frac{I_u R_L^2 k\,(k - N_A)}{r_0} \tag{7.44}$$

This is equivalent to the single-ended INL derivation used by several authors and whose derivation can be found in Razavi's popular converter design book [66] *Principles of Data*



(a) Single-Ended Thermometer-Coded INL (b) Differential Thermometer-Coded INL

Figure 7.8: INL Curves for Thermometer-Coded DAC Models with Finite Output Impedance Current Sources

Conversion System Design. The worst case INL using Equation 7.44 is:

$$\mathrm{INL}_{\mathrm{SE,max}} = \frac{I_{\mathrm{u}} R_L^2 N^2}{4 r_o} \tag{7.45}$$

Fortunately, the situation can be improved by using a differential DAC architecture. Current steering designs inherently differential so a close look at the INL of the architecture is important. Figure 7.9 shows a simple thermometer-coded DAC with a differential output. In the architecture a switch closes the V_{outp} wire when the t_i bit value is one and closes to V_{outm} value when the t_i bit value is zero.

Using Equation 7.43 for the single ended analysis independently,

$$V_{outp}[k] = \frac{r_o V_{CC}}{r_o + kR_L} - \frac{kR_L r_o I_u}{r_o + kR_L} \tag{7.46}$$

$$V_{outm}[k] = \frac{r_o V_{CC}}{r_o + (N_A - k)R_L} - \frac{(N_A - k)R_L r_o I_u}{r_o + (N_A - k)R_L} \tag{7.47}$$

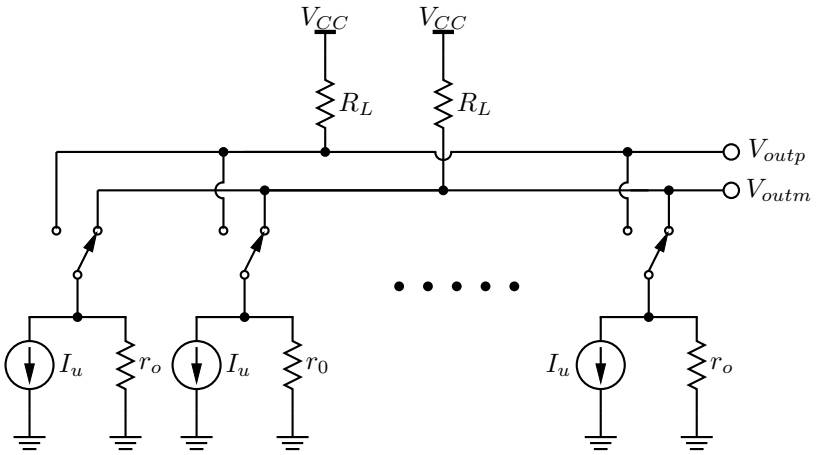


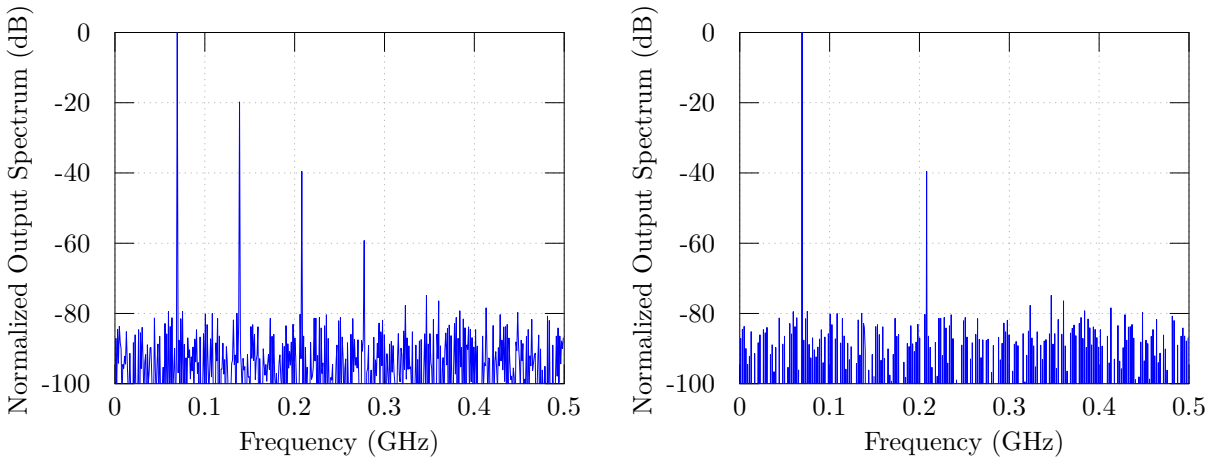
Figure 7.9: Simple Differential Thermometer Model

The output is taken as the difference between V_{outp} and V_{outm}. Therefore, after algebraic manipulation, the output voltage is found to be:

$$V_{out}[k] = V_{outp}[k] - V_{outm}[k] \tag{7.48}$$

$$= \frac{(N_A - 2k)\left(r_o R_L V_{CC} + I_u r_o^2 R_L\right)}{(kN_A - k^2)\, R_L^2 + r_o N_A R_L + r_o^2} \tag{7.49}$$

The INL can then be computed using Equation 7.48. Figure 7.8b shows the differential INL. Notice the improvement is significant. As stated in the previous section, the effect of INL on the output spectrum can be shown by driving a sinusoid through the transfer function. Figure 7.10a shows the effect on the single-ended DAC INL and Figure 7.10b shows the effect using the differential DAC INL. Note that the differential DAC has no even order harmonic distortion, whereas the single-ended DAC suffers from a large second order spur. From this analysis it is clear that DAC current source architectures should be chosen such that the output impedance is large. Some publications refer to the INL/DNL degradation due to changing output impedance from the DAC input code state as *code dependent load variation* (CDLV) [70].



(a) Single-Ended Thermometer-Coded Spectrum (b) Differential Thermometer-Coded Spectrum

7.4 Dynamic DAC Performance

One of the earliest papers dealing specifically identifying the causes of dynamic performance degradation from Van den Bosch *et al.* [71] captures many of the dynamic problems.

1. *The imperfect synchronization of the control signals of the current switches.*
2. *The digital signal feed-through via the C_{gd} of the switch transistors.*
3. *The voltage variation at the drain of the current source transistors.*
4. *The variation in the output impedance of the current sources.*

In addition to these, one of the other major issues is inter-symbol interference. Each of these will be briefly described before offering suggested solutions.

Non-linearities from the finite output impedance of CS DACs have been carefully analyzed by several authors. One of the better works discussing the problems arising from the dynamic effects of a frequency dependent finite output impedance are provided by Lin *et al.* [72]. Lin designed a 2.9 GS/s DAC in a 65 nm CMOS process with excellent linearity. Small feature size CMOS generally does not provide a large output impedance at high frequencies,

when compared to the latest SiGe or InP bipolar devices and thus the design is remarkably interesting. If the r_o from Section 7.3.3 is replaced by an complex impedance and the C_L of the load is not ignored, then the same non-linearities experienced with static output impedance mismatches apply to the dynamic case. If the frequency of the synthesized tone is large, then Z_o and Z_L become small and there is a significant degradation in the performance of the DAC [71].

Mismatches from process variation, or out-right nominal static timing errors, in delay of a signal path can cause harmonic distortion and spurs. The mismatch creates a glitch at the output of the DAC from an off-timing transition. If the DAC is generating a periodic signal, then this mismatch occurs in a periodic manner, generating a spur.

Intersymbol inteference (ISI) describes the phenomenon of a previous DAC code word (symbol) affecting the output characteristics of the current DAC code word. Ideally, the output of the DAC would only be dependent on the current code word. Three significant ISI causes are as follows:

1. The data value of a current switch is dependent on previous states due to the switches themselves not recovering to a memoryless state in a given amount of time.

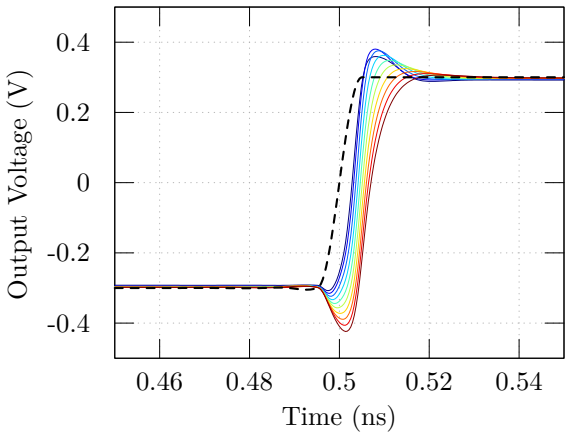
2. Dependency of the connected bias circuitry on the DAC code word.

3. Dependency on the output voltage of the DAC on the switching.

An example of when (2) becomes an issue is when the current source transistors are influenced by the switching be action of the current steering transistors. If the tail current does not return to its nominal operating state before transitioning to the next state, then an code dependent effect will be observed. This effect might be observed when operating near the frequency limits of the technology (i.e. the current source is simply not "fast" enough) or through an improperly designed current source (e.g. the switching pushes transistors into saturation, which can take a significant amount of time to recover).

Figure 7.10 shows an output glitch dependent on the device size of the switches of the current source. This is related to the charge feedthrough described in [64], but it is also simply a function of not adjusting the driving cells of the current switches for scaling.

Figure 7.10: Glitch Versus Device Size (1 μm to 10 μm)



Now that the main static and dynamic sources of error have been presented, DAC and current steering architectures will be presented that help mitigate these errors.

7.5 DAC Architectures

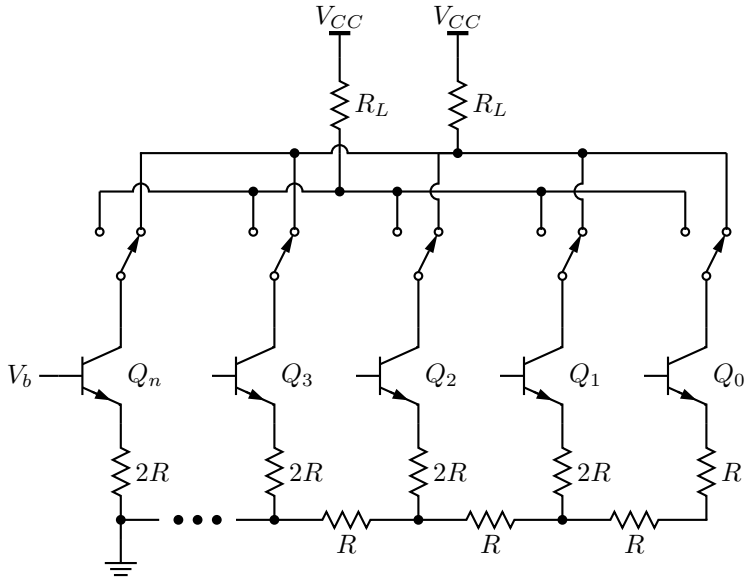
Current steering DACs can be divided into categories based on the architecture chosen in the design. This section provides a brief overview of several important DAC architectures that should be considered when designing a CS DAC. The two main types of DACs considered for CS are binary-weighted DACs and thermometer-coded DACs. An B_A-bit binary weighted CS DAC consists of B_A scaled current sources. The simplest such current steering design scales the transistors through in DAC by a binary weight.

7.5.1 R-2R DACs

The classic R-2R DAC is generally presented in an operational amplifier configuration, but an analog exists for CS architectures [73][74]. In [73], an R-2R ladder network is used in

the design of a 10-bit DAC in a bipolar process. The main motivation for the architecture is to avoid the challenging issue of scaling resistors to achieve binary weighting. Figure 7.11 shows a differential version (as [73] drove the signal single-ended into an operation amplifier) of the architecture. Note that the resistor network is formed at the emitters of the current sources. Also observe that the emitter area of the devices must be scaled with

Figure 7.11: R-2R with Binary Scaling (Emitter Network)



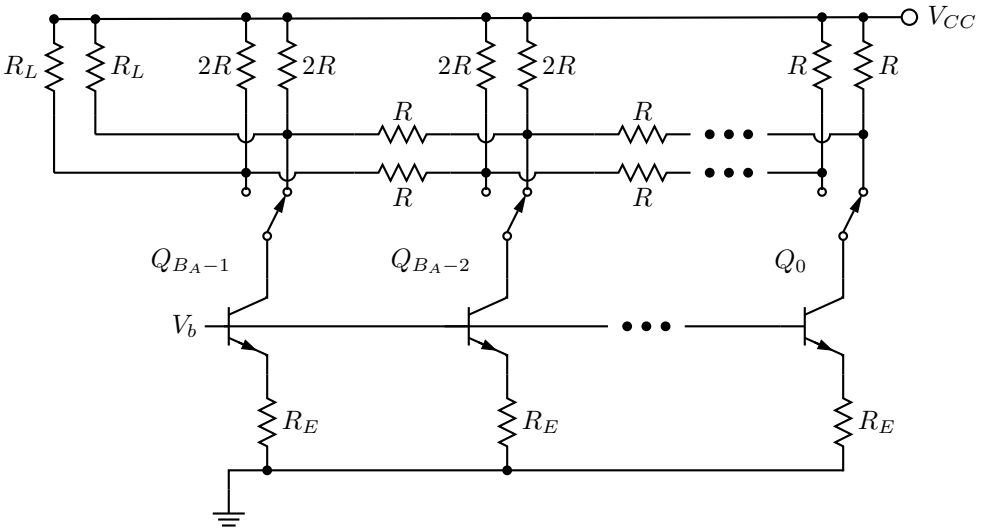
the binary weighting. So this architecture, while it relaxes the resistor sizing requirements, still suffers from a difficult NPN device scaling problem as the number of bits in the design grows large. This particularly a problem at high speeds, as the minimum size device must have a sufficient amount of current to operate at the target frequency. The metric used as a measure of the operating speed of the device at a specific current density is the unity-gain bandwidth product. For an HBT or bipolar transistor, this is (to the first order) [75]:

$$f_T = \frac{1}{2\pi} \frac{g_m}{C_\pi + C_\mu} \tag{7.50}$$

where C_π and C_μ, the Miller capacitance, can be found in the hybrid-π model of a bipolar transistor. g_m is the transconductance of the transistor and is linearly related to the current through the device (Equation 7.82).

The authors [74] introduce an alternative R-2R. The language used to distinguish between the two types of R-2R circuits are *binary attenuation* (the architecture proposed by the author) and *binary scaling*, the previously described R-2R architecture. The naming convention is borrowed for this work. In the binary attenuation architecture, the devices

Figure 7.12: R-2R with Binary Attenuation (Collector Network)



need not be scaled to achieve current scaling at the output. The currents driven by the sources are attenuated through a resistor network to achieve binary weighting. The R-2R ladder is located at the output of the DAC, which are the collectors of the transistor current cell switches, and divides the output current down as shown in Figure 7.12.

The advantages of the binary scaling architecture are:

- The architecture requires half the number of resistors when compared against the binary attenuation architecture in a differential DAC setting. This is because the R-2R division must occur for both outputs.

- Matching between resistors in the network result in common mode INL distortion in a differential architecture.

- The current through the R-2R network is mostly constant and therefore does not suffer from temperature changes based on DAC state. Modern high speed DAC designs rarely mention this, as the temperature time constant is much, much less than the switching speed of the DAC. The mismatch from device mismatch and timing will likely produce error long before that generated by temperature gradients. We also note that small feature sizes allow components to be placed in close proximity to each other, thus allowing a more uniform heat distribution across the R-2R ladder network.

The benefits of a fewer number of resistors is lessened because the devices must be scaled with increasing weight. The number and size of the devices can dominate the area of the resulting DAC. The advantages of the binary attenuation architecture are:

- The devices do not need to be scaled with binary weighting. This is significant, as larger or more devices result in higher parasitics which is of critical importance in high speed designs.

- A simpler emitter generation that is more easily matched between current sources. In small geometry designs, the metal routing can cause significant (where significant depends on the resolution of the DAC) voltage drops.

- Scaling the current through an active current source decreases the output impedance. Finite output impedance of current sources in major contributor to [72],[67],[76]. In Section 7.3.3, the effects of finite output impedance was looked at more closely.

A pure R-2R DAC, regardless of the architecture chosen, is a binary-coded DAC.

7.5.2 Thermometer Coded and Segmented DACs

In binary-coded DACs, each control source is weighted by a factor of two, as discussed in Section 7.5. The implementation of such a DAC is efficient in that only as many active

components as necessary are switched when a code word changes. However, binary DACs are highly susceptible to mismatch errors. Binary DACs also suffer from non-monotinicity in the presence of device mismatch. The jump typically happens on the bit boundary to the next power of two bit, for instance from code 7 (3'b0111) to code 8 (3'b1000).

To address this issue, designers have introduced thermometer-coded DAC architectures [77] [78] [79]. In fact, it would be more suprising to find a modern CS DAC architecture that *did not* have thermometer coded DAC as part of the design. In these designs, the value of the input code word $A[n]$ determines the number of switches to close.

Ideally, a designer would like all the benefits of thermometer-coded and binary-coded DACs simultaneously without suffering any of the drawbacks. One way to balance the area and speed benefits of binary-coded DACs with thermometer-coded DACs is to segment the design into portions. Figure 7.13 shows a generic segmented architectures using thermometer-coded current switches for the MSBs and a binary attenutation R-2R ladder for the LSBs.

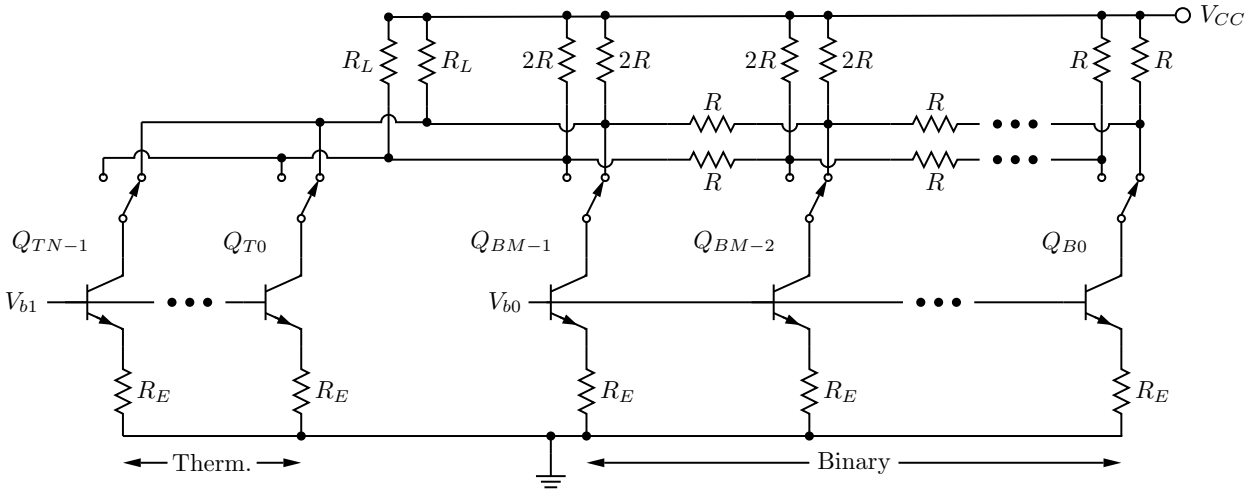


Figure 7.13: Segmented R-2R Binary with Thermometer MSBs

7.5.3 Return-to-Zero (RTZ)

A common technique used to mitigate the impact of ISI is and sometimes to reduce in the impact of sinc roll-off at higher frequencies. The technique can also be used to take signals from higher Nyquist zones. Figure 7.2a shows a NRTZ DAC output and Figure 7.2b shows an RTZ DAC output with 50% duty cycle. Though the RTZ technique has been used extensively in DAC design for a significant period of time, it surprisingly does not show up in many *academic* DAC publications. Table 7.1 is a small collection of RTZ DACs in literature.

Table 7.1: Published RTZ DACs

Publication	Year	Frequncy (GHz)	SFDR (dBc)
[80]	2005	1.6	70
[81]	2011	1.6	66
[4]	2012	7.2	80
[4]	2012	12	67

Compare these results to the NRTZ DAC listed in the Table 7.2 below. Outside of several low frequency DACs (where one would not have issues with ISI), there is a clear performance improvement over a majority of the NRTZ cases. The requirements for an inverse sinc filter are also relaxed as has already been discussed. RTZ addresses ISI by forcing the output of the DAC to a memoryless state (i.e. zero) before applying the next code word. This can clearly be seen in Figure 7.2b. This prevents the DAC from transitioning from a code dependent state, which is obviously have CDLV effects. RTZ also address the charge feedthrough problem from data switching. This is because the data is allowed to change while the output is at a zero state. This is illuminated further in Section 7.6.

7.5.4 Translinear Output Buffers and Non-Linear DACs

Though the issues of implementing a high speed phase accumulator has been addressed in Chapter 5, oftentimes the one wishes to avoid the ROM compression circuitry or the

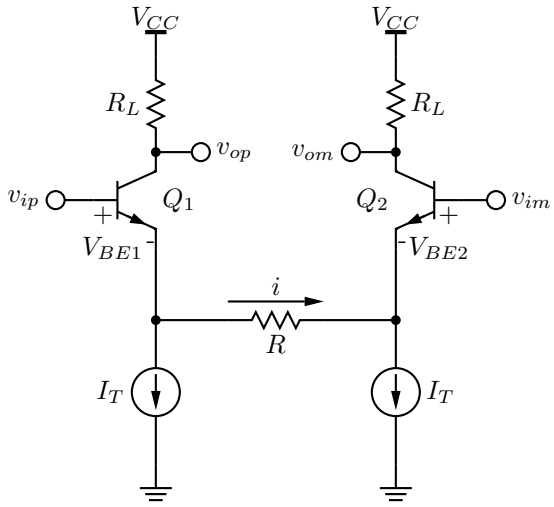
Source	SFDR (Low)	SFDR (Nyq.)	Area (mm^2)	Power (mW)	f_s (MHz)
[82]	58 (9.6 MHz)	N/A	5.000	730	1000
[83]	56 (3.9 MHz)	N/A	1.800	150	125
[84]	49 (8.0 MHz)	N/A	1.220	140	75
[78]	87 (2.0 MHz)	71	16.00	650	100
[85]	73 (8.0 MHz)	55	0.600	125	500
[86]	71 (1.0 MHz)	55	3.200	320	300
[57]	61 (5.0 MHz)	<50	13.10	300	150
[87]	70 (100.0 MHz)	61.2	0.350	110	1000
[64]	78	62	1.130	216	500
[88]	75	63	30.60	6000	1200
[72]	74	52	0.310	188	2900
[89]	76	61	1.000	97	200
[90]	67	N/A	2.500	400	1400
[91]	95 (1 MHz)	<59	0.440	82	320
[92]	71.68 (1 MHz)	43	0.800	25	250
[93]	64 (1 MHz)	<40	1.000	20	100
[94]	82	72	11.83	180	100
[95]	98 (10 MHz)	74	1.950	400	400
[96]	60 (1 MHz)	N/A	0.230	N/A	800
[79]	80.7 (1 MHz)	80.7	0.280	N/A	10
[97]	47.3 (30 MHz)	36.2	0.200	29	3000
[33]	50 (91.7 MHz)	45	4.200	4800	8600

Table 7.2: SFDR of NRTZ DACs

complex multiplexer tree. Directly generating the high speed phase with a pipeline phase accumulator avoids the multiplexer tree entirely. The size and area requirements for ROMs that operate at ultra-high frequencies proved prohibitive. This has lead to the creation of non-linear DACs [98, 33]. In these DACs, the current sources are generally "sine-weighted", such that a linear ramp through the DACs bits generates a sinusoidal output.

An alternative technique is run the phase output through a linear DAC that then drives a translinear device for sinusoidal generation. The idea of using a non-linear device to transform a linear output to a sinusoid is not new in DDFS literature; however, a recent DDFS by Yang *et al.* demonstrates remarkably good high speed performance at low powers [99]. Before delving into the more recent implementation, a review of earlier literature assists

Figure 7.14: Differential Pair



in the development. In 1976, Meyer *et al.* [100] used a differential pair as a triangle to sine wave converter. Figure 7.14 shows the architecture used by Meyer for his triangle-to-sine conversion analysis. The goal is to approximate a sinusoidal output at the terminals of the differential pair given an triangular input using the physical properies a bipolar transistor, i.e.

$$v_{od} = v_{op} - v_{om} = a_1 \sin\left(a_2 v_{id}\right) \tag{7.51}$$

where v_{id} represent the differential input voltage $v_{ip} - v_{im}$ where a_1 and a_2 are two linear coefficients that do not affect the spectral purity of the generated signal. Observing Figure 7.14, becomes clear that the output can be written as a function of the current i through the resistor R. Firstly, Ohm's law is used to find the relationship of the collector current through Q_1 and Q_2 to the output of the differential pair.

$$v_{op} = V_{CC} - R_L I_{C1} \tag{7.52}$$

$$v_{om} = V_{CC} - R_L I_{C2} \tag{7.53}$$

$$v_{od} = R_L \left(I_{C2} - I_{C1}\right) \tag{7.54}$$

where R_L the resistive load and V_{CC} is the supply voltage. The emitter current is related to the collector current of the transistor through the relationship

$$I_E = I_C + \frac{I_C}{\beta_F} = I_C \left(\frac{\beta_F}{1+\beta_F} \right) = \alpha_F I_C \tag{7.55}$$

where β_F is the forward gain of a bipolar transistor and $\alpha_F = \beta_F/(1+\beta_F)$ is commonly used in microelectronics texts [101]. If β_F is sufficiently high, as is the case in SiGe HBTs, then $I_C \approx I_E$ as $\alpha_F \approx 1$. In this particular analysis, α_F is kept throughout the analysis, which differentiates it from Meyer's analysis. This analysis also applies a differential input voltage, as opposed to driving the differential pair single-ended.

$$v_{od} = \frac{1}{\alpha_F} \left(I_{E2} - I_{E1} \right) \tag{7.56}$$

Applying Kirchoff's Current Law (KCL), it is clear that $I_{E1} - i - I_T = 0$ and $I_{E2} + i - I_T = 0$. Adding I_T to both sides of the equality yields Equation 7.57.

$$I_{E1} = I_T + i \tag{7.57}$$

$$I_{E2} = I_T - i \tag{7.58}$$

Substituting Equation 7.57 into Equation 7.56,

$$v_{od} = \frac{1}{\alpha_F} \left[(I_T - i) - (I_T + i) \right] = -\frac{2}{\alpha_F} i \tag{7.59}$$

Thus the output of the differential pair is a linear function i. Consider Kirchoff's Voltage Law (KVL) about the base-emitter pairs for solving for i.

$$-v_{ip} + V_{BE1} + iR - V_{BE2} + v_{im} = 0 \tag{7.60}$$

$$v_{id} = V_{BE1} + iR - V_{BE2} \tag{7.61}$$

The base-emitter voltage can be written as a function of the collector current as shown in Equation 7.62

$$V_{BE} = V_T \ln\left(\frac{I_C}{I_S}\right) \tag{7.62}$$

where I_S is the transport saturation current of the Gummel-Poon model [75] and V_T is the thermal voltage defined in Equation 7.83. The equation assumes that the forward Early voltage, V_A, of the device is infinite (i.e. the bipolar transistors have infinite output impedance, which is certainly not a valid assumption for high output frequencies). Using this relationship, the large signal transfer function of the differential pair can be derived. Substituing Equation 7.62 into Equation 7.60 yields the following equation

$$v_{id} = iR + V_T\left[\ln\left(\frac{I_{C1}}{I_S}\right) - \ln\left(\frac{I_{C2}}{I_S}\right)\right] \tag{7.63}$$

$$= iR + V_T \ln\left(\frac{I_{C1}}{I_{C2}}\right) \tag{7.64}$$

where the subtraction (addition) property of logarithms is used, $\ln(a) - \ln(b) = \ln(a/b)$. Substituting Equation 7.57 and Equation 7.55 into Equation 7.63

$$\frac{v_{id}}{V_T} = \frac{iR}{V_T} + \ln\left[\left(\frac{I_T + i}{\alpha_F}\right)\left(\frac{\alpha_F}{I_T - i}\right)\right] \tag{7.65}$$

$$= \frac{iR}{V_T} + \ln\left(\frac{I_T + i}{I_T - i}\right) \tag{7.66}$$

Applying the Taylor series in the neighborhood of $i = 0$ yields the series

$$\ln\left(\frac{I_T + i}{I_T - i}\right) = 2\left[\frac{i}{I_T} + \frac{i^3}{3I_T^3} + \frac{i^5}{5I_T^5} + \cdots\right] \tag{7.67}$$

$$= 2\sum_{n=0}^{\infty} \frac{i^{2n+1}}{(2n+1)I_T^{2n+1}} \tag{7.68}$$

Now the desired transfer function of i as a function of v_{id} is:

$$i = b_1 \sin\left(b_2 v_{id}\right) \tag{7.69}$$

where b_1 and b_2 are some constants. Thus applying the inverse sine operation to both sides yields

$$b_2 v_{id} = \sin^{-1}\left(\frac{i}{b_1}\right) \tag{7.70}$$

Applying the Taylor series expansion on the inverse sine function for i in the neighborhood of $i = 0$ yields:

$$b_2 v_{id} = \frac{i}{b_1} + \frac{1}{6}\left(\frac{i}{b_1}\right)^3 + \frac{3}{40}\left(\frac{i}{b_1}\right)^5 + \cdots \tag{7.71}$$

Finding b_1 and b_2 in Equation 7.71 such that the error between it and Equation 7.65 is minimized yields the final result

$$b_1 = I_T \tag{7.72}$$

$$b_2 = \frac{1}{V_T}\left(\frac{1}{I_T R/V_T + 2}\right) \tag{7.73}$$

The resulting output is a triangle wave (or sine wave with large odd harmonic terms). But one can outperform a single differential pair with a few more transistors.

Using the Padé approximant, one can generate a rational function of low degree polynomials that approximates transcendental functions such as sine or cosine quite well [102]. Equation 7.74 defines the Padé polynomial approximation of a real function f.

$$f(x) \approx \hat{f}_p(x) = \frac{\sum_{j=0}^{m} a_j x^j}{1 + \sum_{k=1}^{n} b_k x^k} \tag{7.74}$$

where the first $m+n$ derivatives of the function f are equal to the approximation $\hat{f}_p$,

$$f(0) = \hat{f}_p(0) \tag{7.75}$$

$$f'(0) = \hat{f}'_p(0) \tag{7.76}$$

$$f^{(m+n)}(0) = \hat{f}_p^{(m+n)}(0) \tag{7.77}$$

Note that this approximation is closely related to the Maclaurin series of the function f and in fact the Padé approximant often uses the Taylor series during its derivation.

Now consider the following approximations for sinusoidal function.

1. Using the Padé technique to approximate the sin function, we get

$$\alpha \sin(\pi x) \approx \frac{x\left(1-x^2\right)}{1+x^2} \tag{7.78}$$

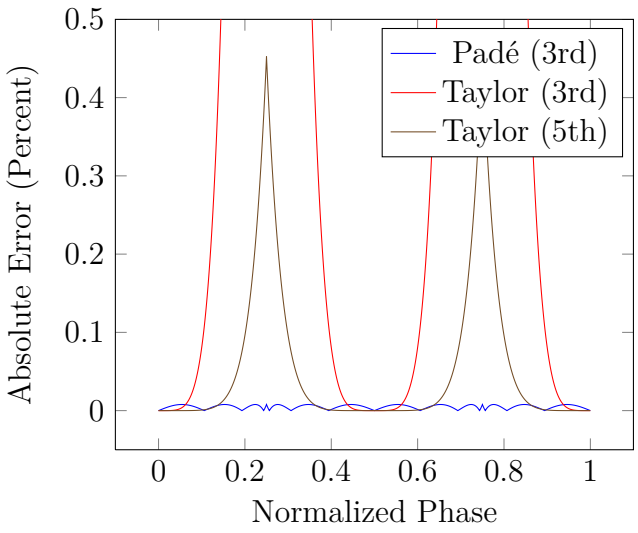
2. Using the Padé technique to approximate the cosine function, we get

$$\cos(\pi x) \approx \frac{\left(1-4x^2\right)\left(2-x^2\right)}{2+x^2} \tag{7.79}$$

Equations 7.78 and 7.79 are versions of the Padé approximation with coefficients rounded to the nearest integer. Figure 7.15 is to help visualize the performance of the Padé approximation against a more commonly used function the Taylor series approximation. The Padé approximant is interesting for sinusoidal approximation for the following two reasons:

- Division is more easily implemented in a translinear circuit than a high order polynomial [102].
- A third order Padé approximant is roughly as complex to implement as a third order Taylor Series approximant with translinear circuit [102].

Figure 7.15: Padé Sine Approximation

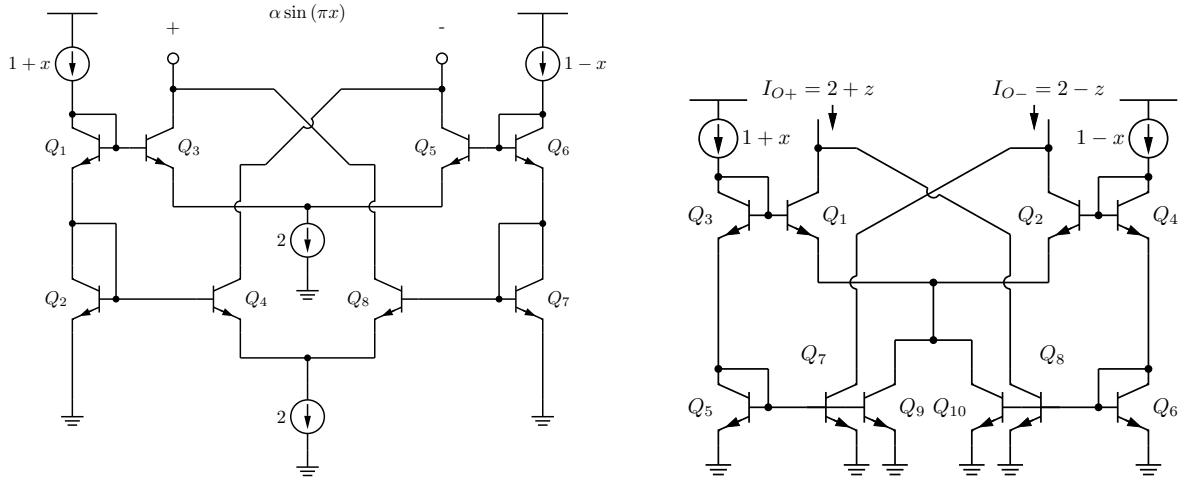


Using the synthesis techniques described in [102], two translinear implementations of the Padé approximations were realized. Figure 7.16a and Figure 7.17 show ideal translinear cicuits for the sine and cosine approximations respectively. Figure 7.16b shows a full transistor implementation of the translinear sine operation.

A novel quadrature DDFS architecture has been proposed to take advantage of the translinear output buffers described thus far in this section. Figure 7.18 provides a block diagram of the proposed DDFS. The design requires a current after the output of of the DAC on the cosine path, since the transfer function of the Padé approximations have been normalized. In practice, the output magnitude of the sine circuit of Figure 7.16a and the cosine circuit of Figure 7.17 are different.

7.6 Current Steering Cell Architectures

One of the critical decisions for designing a current steering DAC is selecting an architecture for the current steering cells that comprise the DAC. The output impedance of the DAC, ISI and sampling rate strongly depend on the performance of this single cell. In this section, several current steering cell architectures are analyzed and the problems addressed,



(a) Ideal Current Sources (b) Transistor Implementation

Figure 7.16: Translinear Sine Implementations

or raised, by each architecture are presented. The design decisions of this component center on trade-offs between performance, complexity, area and power.

The most primitive cell for a differential current steering cell is a three transistor differential pair. The current source transistor has no degeneration and there is no cascoding at any level. Figure 7.19a shows a schematic for the simple current steering cell. Q_{SW1} and Q_{SW2} are the current steering transistors, R_L is the load resistor of the DAC and Q_{CS} is the current source transistor. The current sourced by Q_{CS} is steered through Q_{SW1} and Q_{SW2} depending on which transistor is switched on. The value V_{sp} and V_{sm} are the differential data driving signals with quick transistion times. The differential signals transistion in such a way as to keep the time that both transistors are active relatively small in comparison to the time the data is held.

The simple architecture has several advantages:

- Low power since the power supply voltage can be low.
- Small area since the current cell only requires three transistors to implement.

The drawbacks for this architecture are quite significant unfortunately.

1. The current source output impedance is low and susceptible data changes.

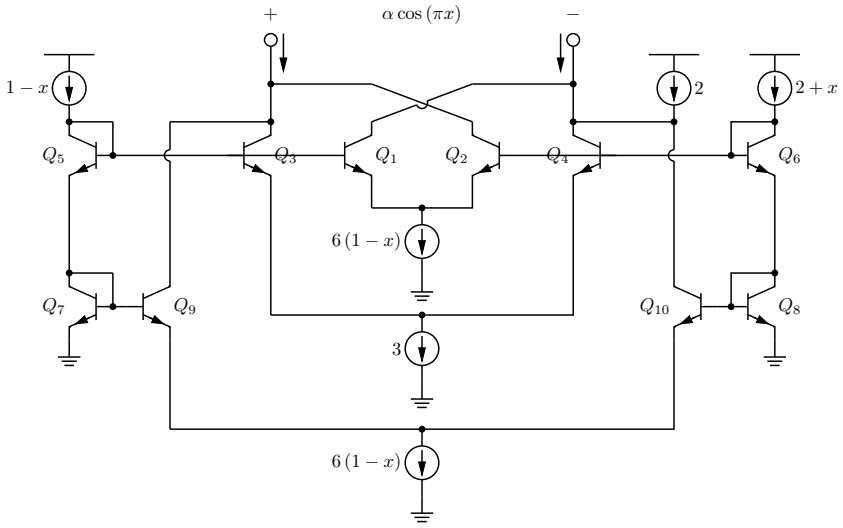


Figure 7.17: Differential Translinear Cosine Implementation (Ideal Current Sources)

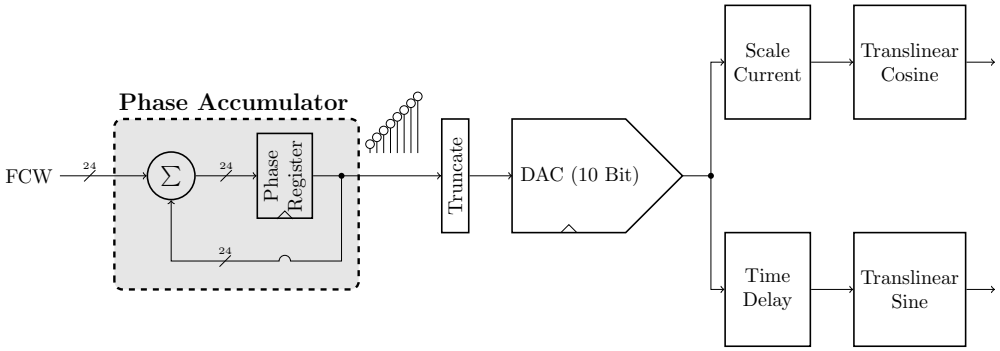
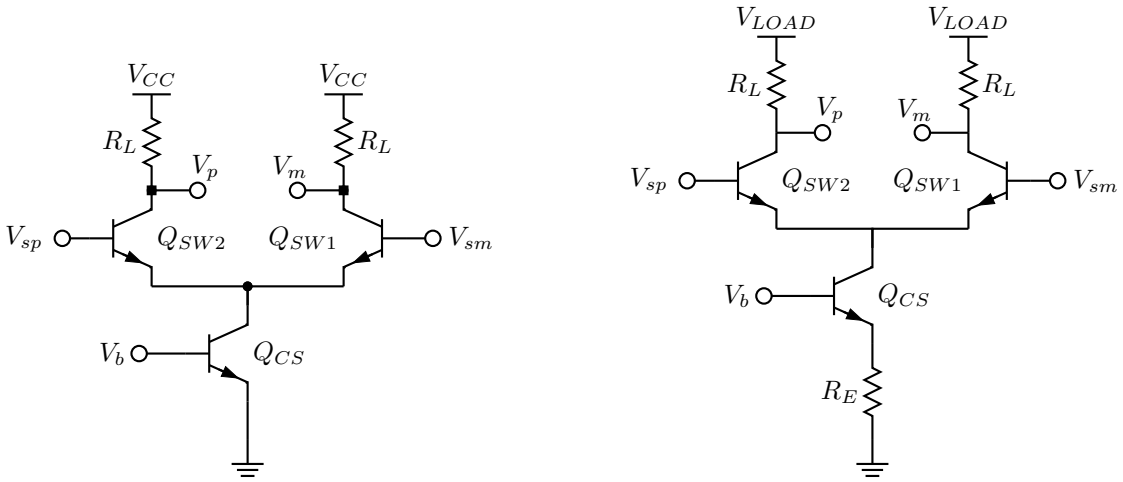


Figure 7.18: Quadrature Translinear DDFS

2. The voltage across the steering transistors is dependent on the output voltage of the DAC.

3. Glitches that capacitively feed through the switching pair are data dependent.

The high speed data switching signal causes the tail current through Q_{CS} to change, since Q_{CS} has a finite output impedance. At first glance, this may appear to be a common-mode effect and therefore eliminated when looking at the output differentially. However, with this configuration, this is not the case. When the glitch in the tail current occurs, it is reflected disproportionately through the active transistor of the current steering pair. The deactivated side is still in the process of activating and thus a majority of the tail current



(a) Simple Current Steering Cell

(b) Simple Current Steering Cell with Degeneration

Figure 7.19: Simple Current Steering Cells

fluctuation appears on a single side of the differential pair. This particular drawback is only important when the glitch power becomes significant with respect to the output of the DAC. For instance, if the DAC is clock slowly for a given process, the glitch will only appear for a tiny fraction of the output code. Significance is also dictated by the required effective number of bits (ENOB) for the DAC. For a DAC that operates near the limits of a technology, we will argue that this is important.

While we are not concerned with the bias structure of the DAC at this point, large fluctuations in the tail current of Q_{CS} will also propagate on the bias line V_b. If multiple current cells are tied to the same bias node, then the current cells have a negative, code dependent interaction. As rate of code changes are related to the signal being converted, this produces a non-linear, output frequency dependent distortion. Improving the output impedance of the current source mitigates this concern. As a reference, the small-signal output impedance of the current source of Q_{CS} is approximately

$$r_o \approx \frac{V_A}{I_C} \tag{7.80}$$

where V_A is the Early voltage of the transistor and I_C the collector current. The size of the glitch at the emitters of the switching resistors is dependent on this value.

A simple step that may be used to improve the finite output impedance of a the DAC is adding a resistor to the emitter of Q_{CS} as shown in Figure 7.19b. This resistor is called a *degeneration* resistor and improves the output impedance of the DAC. Equation 7.81 approximately

$$r_{o,dg} = r_o \left(1 + g_m R_E\right) \tag{7.81}$$

where R_E is the value of the degeneration resistor and g_m is the transconductance of Q_{CS}. The transconductance g_m is:

$$g_m = \frac{I_C}{V_T} \tag{7.82}$$

where I_C is the collector bias current through the transistor and V_T is the thermal voltage and is defined as

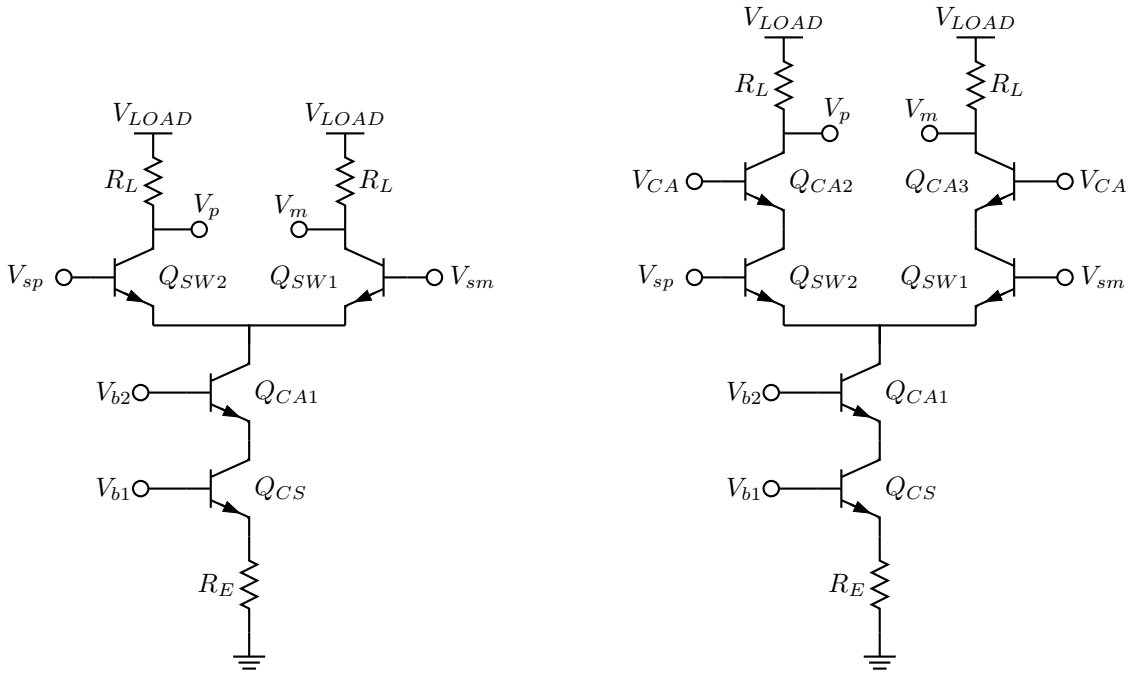
$$V_T = \frac{kT}{q} \tag{7.83}$$

where $k \approx 1.3896593 \times 10^{-23}$ J/K is Boltzmann's constant and $q \approx 1.602176565 \times 10^{-19}$ C is the elementary charge constant. At room temperature, $T_r = 27^\circ$ C, thermal voltage is roughly $V_T \approx 0.026$ V. As $I_C = 1$ mA is a reasonable value for biasing the current source and $R_E = 200\ \Omega$ is a reasonable value for the degeneration resistor, the output impedance of the current source is improved by ≈ 7.5 times.

The drawback, though small, is that the supply voltage of the DAC must be increased to account for the voltage drop across the resistor. Resistors also require a non-negligible amount of area. The output impedance can be improved further by adding a cascode transistor to the current source. Figure 7.20a introduces the cascode transistor Q_{CA1}. We first consider adding the cascode without the degeneration transistor. In that case,

$$r_{o,ca} = r_o \left(1 + \frac{\beta_0 g_m r_o}{\beta_0 + g_m r_o}\right) \approx \beta_0 r_o,\ g_m r_o \gg \beta_0 \tag{7.84}$$

where β_0 is the current gain of Q_{CA1}. In SiGe HBT processes, the current gain can be on the around 200 [103]. Using $I_C = 1$ mA as the standard, this yields $r_{o,ca} \approx 200 r_o$. Adding the degneration resistor only improves this situation further by replacing r_o with Equation 7.81. Using the same 200 Ω resistor and 1 mA collector current, this results in $r_{o,ca,dg} \approx 1400 r_o$.



(a) Current Steering Cell with Cascode Current Source (b) Current Steering Cell with Cascode Output

Figure 7.20: Current Steering Cells with Cascoding

Another dramatic improvement to the current source architecture can be achieved by adding a cascode transistor to the output of the switching transistors. Figure 7.20b provides an example of such a configuration. This particular configuration allows the switching transistors to drive a low impedance output, the emitter of the cascode transistors Q_{CA2} and Q_{CA3}. As shown in some of the DAC architectures of Section 7.5, in particular thermometer coded segments, can have the load of tens of transistors tied to the same node. This load impedance slows the performance of the DAC switches dramatically, but by adding the cascode transistors, the impedance is isolated from the switching transistors. Furthermore, the transistors Q_{CA2} and Q_{CA3} *fix* the voltage variation across the switches across all DAC code

choices to a few hundred millivolts. Otherwise the DAC output voltage is directly applied across the terminals of the switching transistors.

A new problem arises though from adding the cascode at the top of the transistors shown in Figure 7.20b. When the current is steered away from the cascode transistor, the bias current through the non-active switch cut the current off the cascode. This causes a delay from the switch to the output of the DAC, as the cascode must change from operating in an inactive region to being fully biased. To address this shortcoming, adding *keep-alive* transistors to the output cascode as shown in Figure 7.21 will keep those transistors from shutting off completely. The drawback of course is higher output power. This trade-off is very often worth the increased performance. Consider the performance of [72] or [4], which show some of the best published DAC results to date.

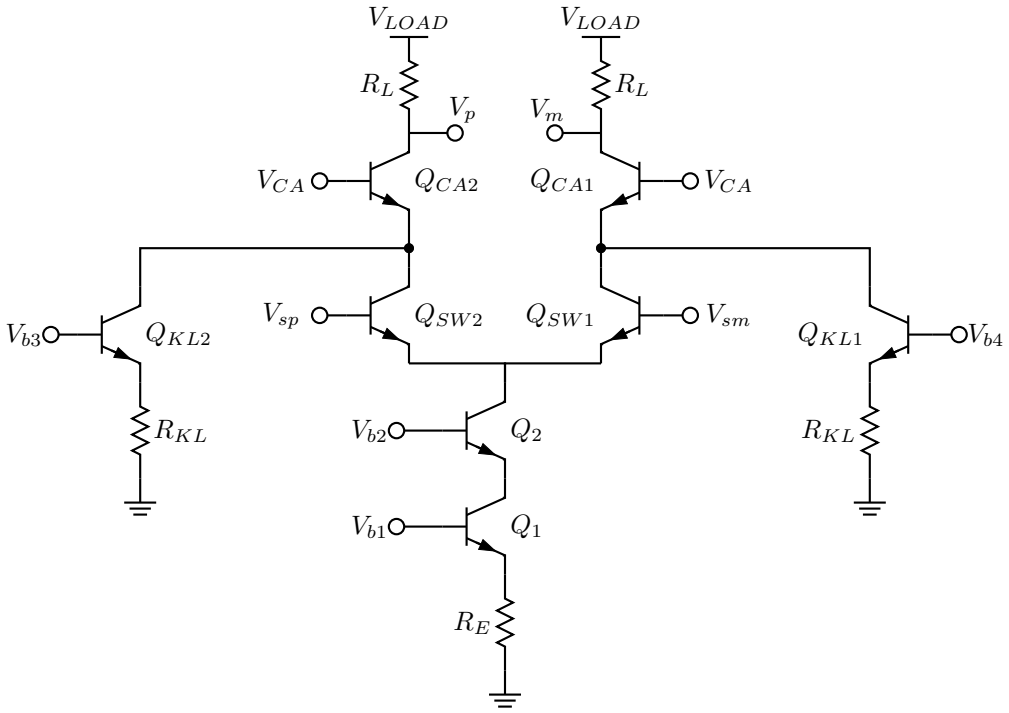


Figure 7.21: Current Steering Cell with Cascode Output and Keep Alive

All the techniques described thus far have done little to improve the non-linear glitching from data switching or intersymbol interference. Both of these become considerable concerns as the frequency of the process extends higher. As has already been discussed in Section

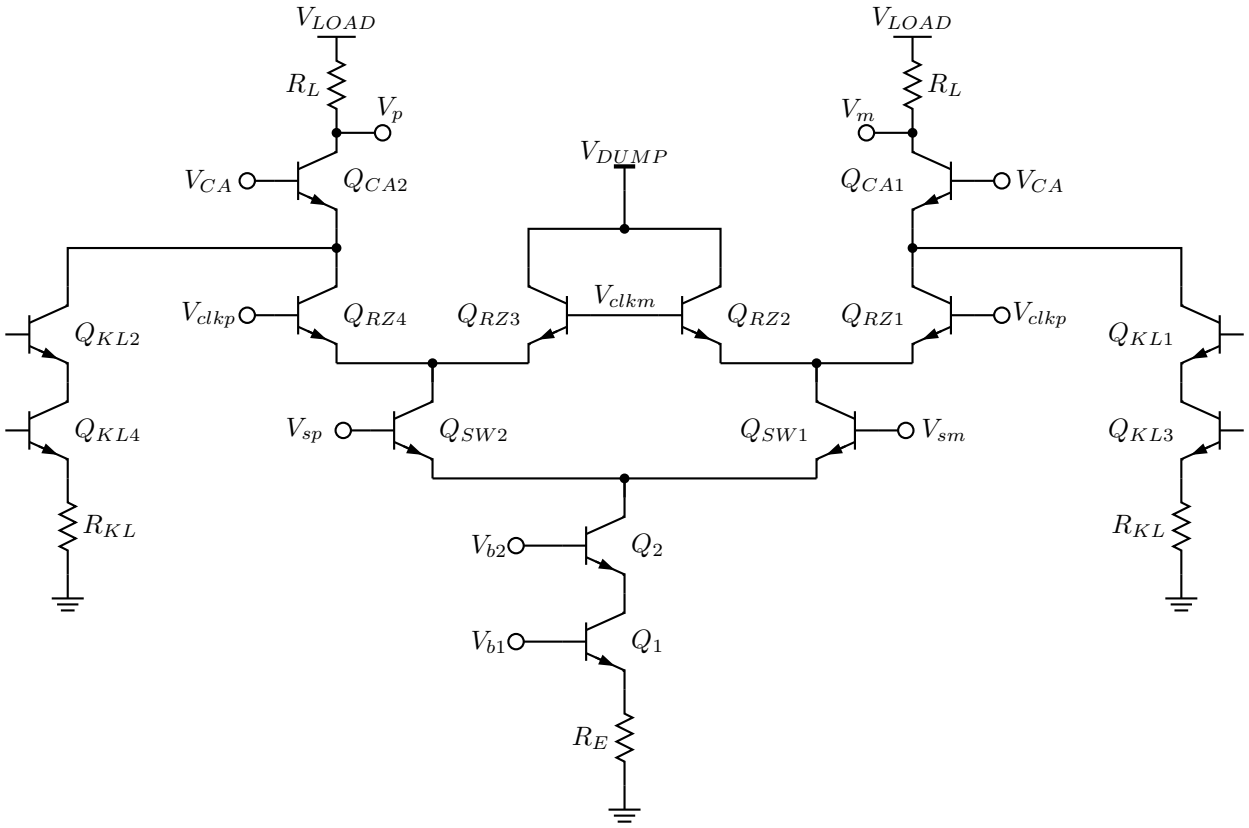


Figure 7.22: Current Steering Cell with Cascode, Keep Alive and RTZ

7.5.3, using an RTZ architecture addresses both problems. Figure 7.22 Combines all of the techniques discussed thus far, including an RTZ switching quad. This RTZ pair happens to incidentally further improve the isolation of the output from the data switches by acting as an extra cascode stage to the data switches.

The author believes that constructing a DAC using segmented R-2R binary attenutation architecture with the current switch shown in Figure 7.22 would dramatically improve the dynamic performance of the DACs being designed at Auburn University. Many of the DACs at Auburn, including the CMOS design discussed in Section 6.6, have dramatic decreases in performance when synthesizing high frequency signal, where high is relative to the sampling frequency of the DAC. The techniques described in this section provide a path to mitigate dynamic degradation effects before even considering calibration.

Chapter 8

Conclusions

In this work, an exact derivation for the spurs generated by phase truncation error in a phase accumulator were calculated using elementary number theory. The spectral theory was developed from binary unsigned arithmetic to the final computations of the discrete Fourier transform of the truncated phase sequence. The theory replaces the commonly cited work by Nicholas [23] and the less commonly cited work from Torosyan [31]. The particular derivation is well suited for teaching DDFS engineers both qualitatively and quantitatively the origin of phase truncation spurs and would fit well in a textbook on the topic.

A novel parallel phase accumulator with linear frequency modulation was introduced and its analysis on the size of the DCDO was presented. It was compared to other parallel accumulators in patent literature that perform a similar operation. In processes CMOS and BiCMOS processes with feature sizes less than or equal to 130 nm, the author argues that *every* DDFS design should be parallelizing the phase accumulator. This approach removes the need for non-linear DAC implementations entirely and allows designers to focus on the components that are actually limiting the performance of DDFS systems (i.e. DACs).

Lastly the DDFS systems designed at Auburn University by the author are presented, culminating in the quadrature DDFS used in the X-band radar-on-a-chip design that was fabricated in a 130 nm BiCMOS process. A revision of the system correctly the errors found during testing was developed to final GDSII form but the team at Auburn University has since taken jobs making testing impractical. The design has not been submitted for fabrication at the time of this writing.

There is a significant opportunity for future work derived from this thesis. Firstly, implementing the modified accumulators described in Section 4.7.1 and Section 4.7.2 with low-cost

FPGA from Xilinx feeding a low-cost DAC demonstration board from Analog Devices would allows for physical verification of the theory, since the theory was only numerically verified in this work. Secondly, use of the theory to modify fully explain the spectral analysis behavior of the output response analyzer and subsequently implementing a new variable state phase accumulator would prove interesting. The new accumulator described could also be used to develop a very fine frequency resolution DDFS. The math fully developing the list of all acquirable frequencies also makes for exciting analysis. Either one of these tasks, if properly built from the work described in this dissertation would be feasible for a master's student to perform. The first could even be accomplished by a senior project for an undergraduate student if proper components were supplied.

An exact analysis of the spectrum of a partial dynamic rotation CORDIC would require significant undertaking but could also lead insights into the devices behavior (and potentially techniques to improve it). The author believes that the CORDIC output stages can actually be used as an "error correction" stage at the output of a highly compressed ROM. To the author's knowledge, no one has ever taken a BTM or MTM LUT as the seed for a partial dynamic rotation CORDIC.

Lastly, the theory developed in Chapter 4 should be used in the analysis of other systems where truncation occurs, such as a fractional-N synthesizer. Some of the theory used in calculating the original phase truncation sequences property may also be used in analyzing the sequences generated by LFSR or $\Delta\Sigma$ modulators. Also, a more abstract, compact analysis producing the same results as this work would also be instrumental in the field.

Bibliography

[1] L. K. Tan, E. Roth, G. Yee, and H. Samueli, "An 800-MHz quadrature digital synthesizer with ECL-compatible output drivers in 0.8 μm CMOS," *IEEE Journal of Solid-State Circuits*, vol. 30, no. 12, pp. 1463 –1473, Dec. 1995.

[2] A. Yamagishi, M. Ishikawa, T. Tsukahara, and S. Date, "A 2-V, 2-GHz low-power direct digital frequency synthesizer chip-set for wireless communication," *IEEE Journal of Solid-State Circuits*, vol. 33, pp. 210–217, 1998.

[3] B.-D. Yang, J.-H. Choi, S.-H. Han, L.-S. Kim, and H.-K. Yu, "An 800-MHz low-power direct digital frequency synthesizer with an on-chip D/A converter," *IEEE Journal of Solid-State Circuits*, vol. 39, no. 5, pp. 761–774, 2004.

[4] F. Van de Sande, N. Lugil, F. Demarsin, Z. Hendrix, A. Andries, P. Brandt, W. Anklam, J. S. Patterson, B. Miller, M. Rytting, M. Whaley, B. Jewett, J. Liu, J. Wegman, and K. Poulton, "A 7.2 GSa/s, 14 Bit or 12 GSa/s, 12 Bit Signal Generator on a Chip in a 165 GHz f_T BiCMOS process," *IEEE Journal of Solid-State Circuits*, vol. 47, no. 4, pp. 1003–1012, 2012.

[5] T. Nagasaku, K. Kogo, H. Shinoda, H. Kondoh, Y. Muto, A. Yamamoto, and T. Yoshikawa, "77GHz low-cost single-chip radar sensor for automotive ground speed detection," in *Proc. IEEE Compound Semiconductor Integrated Circuits Symp. CSIC '08*, 2008, pp. 1–4.

[6] Y.-A. Li, M.-H. Hung, S.-J. Huang, and J. Lee, "A fully integrated 77GHz FMCW radar system in 65nm CMOS," in *Proc. IEEE Int. Solid-State Circuits Conf. Digest of Technical Papers (ISSCC)*, 2010, pp. 216–217.

[7] J. Rogers, C. Plett, and F. Dai, *Integrated Circuit Design for High-Speed Frequency Synthesis*. Artech House, 2006.

[8] M. Skolnik, *Radar Handbook*, 3rd ed. McGraw Hill, 2008.

[9] J. Tierney, C. Rader, and B. Gold, "A digital frequency synthesizer," *IEEE Transactions on Audio and Electroacoustics*, vol. 19, no. 1, pp. 48–57, 1971.

[10] A. Torosyan and A. N. Willson, "Exact analysis of DDS spurs and SNR due to phase truncation and arbitrary phase-to-amplitude errors," in *Proc. IEEE Int. Frequency Control Symp. and Exposition*, 2005.

[11] D. D. Sarma and D. W. Matula, "Faithful bipartite rom reciprocal tables," in *Proceedings of the 12th Symposium on Computer Arithmetic*, 1995, p. 17.

[12] F. de Dinechin and A. Tisserand, "Multipartite table methods," *IEEE Transactions on Computers*, vol. 54, no. 3, pp. 319–330, 2005.

[13] J. Qin, "Selective spectrum analysis and numerically controlled oscillator in mixed-signal built-in self-test," Ph.D. dissertation, Auburn University, December 2010.

[14] G. E. Shilov, *Elementary Real and Complex Analysis.* Dover Publications, Inc., 1973.

[15] R. F. Lax, *Modern Algebra and Discrete Structures.* Addison-Wesley Educational Publishers Inc., 1991.

[16] A. Torosyan, "Direct digital frequency synthesizers: Complete analysis and design guidelines," Ph.D. dissertation, University of California, Los Angeles, 2003.

[17] J. F. Wakerly, *Digital Design Principles and Practices*, 3rd ed. Prentice Hall, 2001.

[18] J. R. Barry, E. A. Lee, and D. G. Messerschmitt, *Digital Communication.* Springer, 2003.

[19] A. Devices, "1 GSPS, 14-Bit, 3.3V CMOS direct digital synthesizer," 2012.

[20] ——, "3.5 GSPS direct digital synthesizer with 12-bit DAC," 2012.

[21] J. Qin, J. D. Cali, B. F. Dutton, G. J. Starr, F. F. Dai, and C. E. Stroud, "Selective Spectrum Analysis for Analog Measurements," *IEEE Transactions on Industrial Electronics*, vol. 58, no. 10, pp. 4960–4971, October 2011.

[22] J. Yu, F. Zhao, J. Cali, D. Ma, X. Geng, F. F. Dai, J. D. Irwin, and A. Aklian, "A Single-Chip X-band Chirp Radar MMIC with Stretch Processing," in *CICC*, 2012, pp. 1–4.

[23] H. T. Nicholas and H. Samueli, "An analysis of the output spectrum of direct digital frequency synthesizers in the presence of phase-accumulator truncation," in *Proc. 41st Annual Symp. Frequency Control. 1987*, 1987, pp. 495–502.

[24] Y. C. Jenq, "Digital spectra of nonuniformly sampled signals. ii. Digital look-up tunable sinusoidal oscillators," *IEEE Transactions on Instrumentation and Measurement*, vol. 37, no. 3, pp. 358–362, 1988.

[25] S. Mehrgardt, "Noise spectra of digital sine-generators using the table-lookup method," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 31, no. 4, pp. 1037–1039, 1983.

[26] Y.-C. Jenq, "Digital spectra of nonuniformly sampled signals: fundamentals and high-speed waveform digitizers," *IEEE Transactions on Instrumentation and Measurement*, vol. 37, no. 2, pp. 245–251, 1988.

[27] Y. C. Jenq, "Digital spectra of nonuniformly sampled signals: theories and applications-measuring clock/aperture jitter of an A/D system," *IEEE Transactions on Instrumentation and Measurement*, vol. 39, no. 6, pp. 969–971, 1990.

[28] Y.-C. Jenq, "Digital spectra of nonuniformly sampled signals: a robust sampling time offset estimation algorithm for ultra high-speed waveform digitizers using interleaving," *IEEE Transactions on Instrumentation and Measurement*, vol. 39, no. 1, pp. 71–75, 1990.

[29] A. Torosyan and J. Willson, A. N., "Analysis of the output spectrum for direct digital frequency synthesizers in the presence of phase truncation and finite arithmetic precision," in *Proc. 2nd Int. Symp. Image and Signal Processing and Analysis ISPA 2001*, 2001, pp. 458–463.

[30] U. Dudley, *Elementary Number Theory*. Dover Publications, Inc., 1978.

[31] A. Torosyan, D. Fu, and J. Willson, A. N., "A 300-MHz quadrature direct digital synthesizer/mixer in 0.25-μm CMOS," *IEEE Journal of Solid-State Circuits*, vol. 38, no. 6, pp. 875–887, 2003.

[32] K. Doris, A. van Roermund, and D. Leenaerts, *Wide-Bandwidth High Dynamic Range D/A Converters*. Springer, 2010.

[33] X. Geng, F. Dai, J. Irwin, and R. Jaeger, "An 11-bit 8.6 GHz direct digital synthesizer MMIC with 10-bit segmented sine-weighted DAC," *Solid-State Circuits, IEEE Journal of*, vol. 45, no. 2, pp. 300–313, feb 2010.

[34] S. Turner and D. Kotecki, "Direct Digital Synthesizer with Sine-Weighted DAC at 32-GHz Clock Frequency in InP DHBT technology," *IEEE Journal of Solid-State Circuits*, vol. 41, no. 10, pp. 2284–2290, oct 2006.

[35] A. Gutierrez-Aitken, J. Matsui, E. Kaneshiro, B. Oyama, D. Sawdai, A. Oki, and D. Streit, "Ultrahigh-speed direct digital synthesizer using inp dhbt technology," *IEEE Journal of Solid-State Circuits*, vol. 37, no. 9, pp. 1115 – 1119, sep 2002.

[36] X. Yu, F. F. Dai, J. David Irwin, and R. Jaeger, "A 9-bit Quadrature Direct Digital Synthesizer Implemented in 0.18-μm SiGe BiCMOS Technology," *Microwave Theory and Techniques, IEEE Transactions on*, vol. 56, no. 5, pp. 1257–1266, may 2008.

[37] S. Pellerano, S. Levantino, C. Samori, and A. Lacaita, "A 13.5-mw 5-ghz frequency synthesizer with dynamic-logic frequency divider," *Solid-State Circuits, IEEE Journal of*, vol. 39, no. 2, pp. 378–383, 2004.

[38] R. H. A. W. Kovalick, "Waveform synthesis using multiplexed parallel synthesizers," USA Patent 4,454,486, June, 1984.

[39] B.-G. Goldberg, "Digital frequency synthesizer having multiple processing paths," USA Patent 4,958,310, nov, 1990.

[40] P. A. D. B. L. Tise, "Multiplexed chirp waveform synthesizer," USA Patent 6,614,813, September, 2003.

[41] S. Turner and D. Kotecki, "Direct digital synthesizer with ROM-Less architecture at 13-GHz clock frequency in InP DHBT technology," *IEEE Microwave and Wireless Components Letters*, vol. 16, no. 5, pp. 296–298, may 2006.

[42] X. Geng, F. Dai, J. Irwin, and R. Jaeger, "24-bit 5.0 GHz direct digital synthesizer RFIC with direct digital modulations in 0.13 μ m sige bicmos technology," *IEEE Journal of Solid-State Circuits*, vol. 45, no. 5, pp. 944–954, may 2010.

[43] e2v, "Low Power 12-bit 3 GSps DAC with 4/2:1 MUX," October 2011.

[44] G. J. Starr, J. Qin, B. F. Dutton, C. E. Stroud, F. F. Dai, and V. P. Nelson, "Automated generation of built-in self-test and measurement circuitry for mixed-signal circuits and systems," in *Proc. 24th IEEE Int. Symp. Defect and Fault Tolerance in VLSI Systems DFT '09*, 2009, pp. 11–19.

[45] "Scientific computing tools for python - numpy," Apr. 2012.

[46] "welcome to mako," Apr. 2012. [Online]. Available: http://www.makotemplates.org/

[47] D. D. Caro, N. Petra, and A. G. M. Strollo, "Reducing lookup-table size in direct digital frequency synthesizers using optimized multipartite table method," *IEEE Transaction on Circuits and Systems*, vol. 55, no. 7, pp. 2116–2127, Aug. 2008.

[48] M. J. Schulte and J. E. Stine, "Approximating Elementary Functions with Symmetric Bipartite Tables," *IEEE Transactions on Computers*, p. 842, 1999.

[49] J. W. Eaton, D. Bateman, and S. Hauberg, *GNU Octave Manual Version 3*. Network Theory Limited, 2008.

[50] S. Axler, *Linear Algebra Done Right*, 2nd ed. Springer, 1997.

[51] R. M. Gray and J. Stockham, T. G., "Dithered quantizers," *IEEE Transactions on Information Theory*, vol. 39, no. 3, pp. 805–812, 1993.

[52] T. E. C. III, G. M. Flewelling, D. S. Jansen, J. D. Cali, D. A. Chan, J. Freedman, M. Anthony, T. Dresser, F. Dai, and E. Gebarra, "Self-Healing in SiGe BiCMOS ICs for Low-SWAP Electronic Warfare Receivers," in *38th Annual GOMACTech Conference*, March 11-14 2013.

[53] J. E. Volder, "The cordic trigonometric computing technique," *IRE Transactions on Electronic Computers*, no. 3, pp. 330–334, 1959.

[54] J. S. Walther, "A Unified Algorithm for Elementary Functions," in *Proc. of Spring Joint Computer Conf.*, 1971, pp. 379–385.

[55] J.-M. Muller, *Elementary Functions: Algorithms and Implementation*, 2nd ed. Birkh auser, 2006.

[56] H. Samueli, "The design of multiplierless fir filters for compensating d/a converter frequency response distortion," *IEEE Transactions on Circuits and Systems*, vol. 35, no. 8, pp. 1064–1066, 1988.

[57] 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^2 random walk CMOS DAC," *IEEE Journal of Solid-State Circuits*, vol. 34, no. 12, pp. 1708–1718, 1999.

[58] A. Devices, "Ad9737a: RF Digital-to-Analog Converters," 2012.

[59] S. K. Mitra, *Digital Signal Processing: A Computer-Based Approach*, 3rd ed. McGraw Hill, 2006.

[60] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, pp. 379–423, 1948.

[61] B. Widrow and I. Kollár, *Quantization Noise*. Cambridge University Press, 2008.

[62] V. I. Bogachev, *Measure Theory: Volume 1*. Springer, 2007.

[63] D. Duttweiler and D. Messerschmitt, "Analysis of Digitally Generated Sinusoids with Application to A/D and D/A Converter Testing," *IEEE Transactions on Communications*, vol. 26, no. 5, pp. 669–675, 1978.

[64] K. Doris, J. Briaire, D. Leenaerts, M. Vertreg, and A. van Roermund, "A 12b 500MS/s DAC with $>$ 70dB SFDR up to 120MHz in 0.18μm CMOS," in *Proc. Digest of Technical Papers Solid-State Circuits Conf. ISSCC. 2005 IEEE Int*, 2005, pp. 116–588.

[65] *IEEE Standard 746-1984: Performance Measurements of A/D and D/A Conversion Techniques and Their Applications*, IEEE Std., 1984.

[66] B. Razavi, *Principles of Data Conversion System Design*, J. B. Anderson, Ed. New York: Wiley-IEEE Press, 1995.

[67] S. Luschas and H.-S. Lee, "Output impedance requirements for DACs," in *Proc. Int. Symp. Circuits and Systems ISCAS '03*, vol. 1, 2003.

[68] G. I. Radulov, M. Heydenreich, R. W. van der Hofstad, J. A. Hegt, and A. H. M. van Roermund, "Brownian-Bridge-Based Statistical Analysis of the DAC INL Caused by Current Mismatch," *IEEE Transactions on Circuits and Systems—Part II: Express Briefs*, vol. 54, no. 2, pp. 146–150, 2007.

[69] N. C.-C. Lu, L. Gerzberg, C.-Y. Lu, and J. D. Meindl, "Modeling and optimization of monolithic polycrystalline silicon resistors," *IEEE Transactions on Electron Devices*, vol. 28, no. 7, pp. 818–830, 1981.

[70] W.-H. Tseng, C.-W. Fan, and J.-T. Wu, "A 12-Bit 1.25-GS/s DAC in 90 nm CMOS With $>$ 70 db SFDR up to 500 MHz," *IEEE Journal of Solid-State Circuits*, vol. 46, pp. 2845–2856, 2011.

[71] A. Van den Bosch, M. Steyaert, and W. Sansen, "SFDR-bandwidth limitations for high speed high resolution current steering CMOS D/A converters," in *Electronics, Circuits and Systems, 1999. Proceedings of ICECS '99. The 6th IEEE International Conference on*, vol. 3, 1999, pp. 1193–1196 vol.3.

[72] C.-H. Lin, F. M. I. van der Goes, J. R. Westra, J. Mulder, Y. Lin, E. Arslan, E. Ayranci, X. Liu, and K. Bult, "A 12 bit 2.9 GS/s DAC with IM3 < -60 dBc beyond 1 GHz in 65 nm CMOS," *IEEE Journal of Solid-State Circuits*, vol. 44, pp. 3285–3293, 2009.

[73] D. J. Dooley, "A Complete Monolithic 10-b D/A Converter," *IEEE Journal of Solid-State Circuits*, vol. 8, no. 6, pp. 404–408, 1973.

[74] G. Kelson, H. H. Stellrecht, and D. S. Perloff, "A Monolithic 10-b Digital-to-Analog Converter Using Ion Implantation," *IEEE Journal of Solid-State Circuits*, vol. 8, no. 6, pp. 396–403, 1973.

[75] P. R. Gray, P. J. Hurst, S. H. Lewis, and R. G. Meyer, *Analysis and Design of Analog Integrated Circuits*, 4th ed. John Wiley and Sons, Inc., 2001.

[76] Y. Tang, J. Briaire, K. Doris, R. van Veldhoven, P. C. W. van Beek, H. J. A. Hegt, and A. H. M. van Roermund, "A 14 bit 200 MS/s DAC with SFDR > 78 dBc, IM3 < -83 dBc and NSD < -163 dBm/Hz Across the Whole Nyquist Band Enabled by Dynamic-Mismatch Mapping," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 6, pp. 1371–1381, 2011.

[77] A. Van Den Bosch, M. Borremans, M. Steyaert, and W. Sansen, "A 12 b 500 MSample/s current-steering CMOS D/A converter," in *Proc. Digest of Technical Papers Solid-State Circuits Conf. ISSCC. 2001 IEEE Int*, 2001, pp. 366–367.

[78] B. J. Tesch and J. C. Garcia, "A low glitch 14-b 100-MHz D/A converter," *IEEE Journal of Solid-State Circuits*, vol. 32, no. 9, pp. 1465–1469, 1997.

[79] D.-H. Lee, T.-H. Kuo, and K.-L. Wen, "Low-Cost 14-Bit Current-Steering DAC with a Randomized Thermometer-Coding Method," *IEEE Transactions on Circuits and Systems—Part II: Express Briefs*, vol. 56, no. 2, pp. 137–141, 2009.

[80] M.-J. Choe, K.-H. Baek, and M. Teshome, "A 1.6-GS/s 12-bit return-to-zero GaAs RF DAC for multiple Nyquist operation," *IEEE Journal of Solid-State Circuits*, vol. 40, pp. 2456–2468, 2005.

[81] W.-H. Tseng, J.-T. Wu, and Y.-C. Chu, "A CMOS 8-Bit 1.6-GS/s DAC with Digital Random Return-to-Zero," *IEEE Transactions on Circuits and Systems—Part II: Express Briefs*, vol. 58, 2011.

[82] P. Vorenkamp, J. Verdaasdonk, R. van de Plassche, and D. Scheffer, "A 1 GS/s, 10b digital-to-analog converter," in *Proc. IEEE Int. Solid-State Circuits Conf. Digest of Technical Papers. 41st ISSCC*, 1994, pp. 52–53.

[83] S.-Y. Chin and C.-Y. Wu, "A 10-b 125-MHz CMOS digital-to-analog converter (DAC) with threshold-voltage compensated current sources," *IEEE Journal of Solid-State Circuits*, vol. 29, no. 11, pp. 1374–1380, 1994.

[84] T.-Y. Wu, C.-T. Jih, J.-C. Chen, and C.-Y. Wu, "A low glitch 10-bit 75-MHz CMOS video D/A converter," *IEEE Journal of Solid-State Circuits*, vol. 30, no. 1, pp. 68–72, 1995.

[85] C.-H. Lin and K. Bult, "A 10-b, 500-MSample/s CMOS DAC in 0.6 mm^2," *IEEE Journal of Solid-State Circuits*, vol. 33, no. 12, pp. 1948–1958, 1998.

[86] J. Bastos, A. M. Marques, M. S. J. Steyaert, and W. Sansen, "A 12-bit Intrinsic Accuracy High-speed CMOS DAC," *IEEE Journal of Solid-State Circuits*, vol. 33, pp. 1959–1969, 1998.

[87] A. Van den Bosch, M. Borremans, M. Steyaert, and W. Sansen, "A 10-bit 1-GSample/s Nyquist current-steering CMOS D/A converter," in *Proc. CICC Custom Integrated Circuits Conf the IEEE 2000*, 2000, pp. 265–268.

[88] B. Jewett, J. Liu, and K. Poulton, "A 1.2GS/s 15b DAC for precision signal generation," in *Proc. Digest of Technical Papers Solid-State Circuits Conf. ISSCC. 2005 IEEE Int*, 2005, pp. 110–587.

[89] Q. Huang, P. A. Francese, C. Martelli, and J. Nielsen, "A 200MS/s 14b 97mW DAC in 0.18μm CMOS," in *Proc. Digest of Technical Papers Solid-State Circuits Conf. ISSCC. 2004 IEEE Int*, 2004, pp. 364–532.

[90] B. Schafferer and R. Adams, "A 3V CMOS 400mW 14b 1.4GS/s DAC for multi-carrier applications," in *Proc. Digest of Technical Papers Solid-State Circuits Conf. ISSCC. 2004 IEEE Int*, 2004, pp. 360–532.

[91] K. O'Sullivan, C. Gorman, M. Hennessy, and V. Callaghan, "A 12b 320 MSample/s current-steering CMOS D/A converter in 0.44mm^2," in *Proc. 29th European Solid-State Circuits Conf. ESSCIRC '03*, 2003, pp. 89–92.

[92] J.-H. Chi, S.-H. Chu, and T.-H. Tsai, "A 1.8-v 12-bit 250-ms/s 25-mw self-calibrated DAC," in *Proc. ESSCIRC*, 2010, pp. 222–225.

[93] M. P. Tiilikainen, "A 14-bit 1.8-v 20-mw 1-mm2 CMOS DAC," *IEEE Journal of Solid-State Circuits*, vol. 36, no. 7, pp. 1144–1147, 2001.

[94] A. R. Bugeja and B.-S. Song, "A self-trimming 14-b 100-MS/s CMOS DAC," *IEEE Journal of Solid-State Circuits*, vol. 35, no. 12, pp. 1841–1852, 2000.

[95] W. Schofield, D. Mercer, and L. S. Onge, "A 16b 400MS/s DAC with −80dBc IMD to 300MHz and −160dBm/Hz noise power spectral density," in *Proc. Digest of Technical Papers Solid-State Circuits Conf. ISSCC. 2003 IEEE Int*, 2003, pp. 126–482.

[96] M. Borremans, A. Van den Bosch, M. Steynaert, and W. Sansen, "A low power, 10-bit CMOS D/A converter for high speed applications," in *Proc. IEEE Conf Custom Integrated Circuits*, 2001, pp. 157–160.

[97] X. Wu, P. Palmers, and M. Steyaert, "A 130 nm CMOS 6-bit Full Nyquist 3 GS/s DAC," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 11, pp. 2396–2403, 2008.

[98] Z. Zhou and G. S. La Rue, "A 12-bit Nonlinear DAC for Direct Digital Frequency Synthesis," *IEEE Transactions on Circuits and Systems—Part I: Regular Papers*, vol. 55, no. 9, pp. 2459–2468, 2008.

[99] C.-Y. Yang, J.-H. Weng, and H.-Y. Chang, "A 5-GHz Direct Digital Frequency Synthesizer Using an Analog-Sine-Mapping Technique in 0.35-μm SiGe BiCMOS," *IEEE Journal of Solid-State Circuits*, vol. 46, no. 9, pp. 2064–2072, 2011.

[100] R. G. Meyer, W. M. C. Sansen, and S. Peeters, "The Differential Pair as a Triangle-Sine Wave Converter," *IEEE Journal of Solid-State Circuits*, vol. 11, no. 3, pp. 418–420, 1976.

[101] R. C. Jaeger and T. N. Blalock, *Microelectronic Circuit Design*, 3rd ed. McGraw Hill Science, Engineering and Math, 2007.

[102] E. Seevinck, *Analysis and Synthesis of Translinear Integrated Circuits*. Booksurge Publishing, 1988.

[103] J. D. Cressler and G. Niu, *Silicon-Germanium Heterojunction Bipolar Transistors*. Artech House, 2003.