Novel Design Approach and Architectures for Sinusoid Output Direct Digital Frequency Synthesis

Conception et Architectures Nouvelles pour Synthétiseurs Numériques de Fréquences

A Thesis Submitted

to the Department of Electrical and Computer Engineering

of the Royal Military College of Canada

by

Joseph Mathieu Pierre Langlois, CD, rmc, B.Eng., M.Eng., P.Eng.

Lieutenant-Commander

In Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

September 2002

0-612-71892-1

Acknowledgements remerciements

Let us state the obvious: this thesis could not have been completed without the assistance and support from several people. I would therefore like to express my sincere gratitude to all of them.

Dr. Dhamin Al-Khalili, for his constant involvement and interest in this research, even during his sabbatical leave, and for his outstanding ability at turning apparent dead ends into fruitful ideas.

Dr. Valek Szwarc, for suggesting research on Direct Digital Frequency Synthesis.

Dr. Gilles Labonté pour son introduction succincte aux algorithmes génétiques, qui ont permis de débloquer cette recherche. Dr. Robert Gervais, pour avoir révisé une partie des dérivations mathématiques. Dr. Lucien Haddad et Dr. Philippe Baille, pour les discussions sur de nouvelles idées. Dr. Robert Benesch et Dr. Jean Fugère, pour leur support constant depuis cinq ans.

Jean-Luc Derome, pour l'aide avec les outils Cadence™. Jennifer Mullan, pour la multitude d'articles obtenus d'obscures conférences. Mariusz Jarosz, for his help during hardware testing.

The Canadian Microelectronics Corporation, for the availability of design tools and for grants of fabrication area.

Mes parents, Mariette Drouin et Rodolphe Langlois.

Mon épouse et meilleure amie, Caroline Dion, pour avoir accepté que j'accorde autant de mon temps et de mon énergie à ce travail.

Cette dissertation est dédiée à mes enfants, Èvelyne et Étienne, et à mon épouse Caroline.

Abstract

Langlois, Joseph Mathieu Pierre, Royal Military College of Canada, May 2002, Novel Design Approach and Architectures for Sinusoid Output Direct Digital Frequency Synthesis

Supervisor: Dr. Dhamin Al-Khalili

Direct Digital Frequency Synthesis (DDFS) makes it possible to generate single phase or quadrature sinusoids over bandwidths measured in hundreds of megahertz, with sub-hertz frequency resolution, unparalleled frequency hopping rates, and phase continuity on hopping. These characteristics make DDF Synthesizers ideal in a wide array of applications, including spread-spectrum communications, radar, instrumentation and broadcasting. Design challenges include maintaining high spectral purity, high clock rates and low power consumption.

This thesis presents a novel approach to the design of sinusoid output DDF Synthesizers with Phase-to-Sine Amplitude Converters based on linear interpolation. For such synthesizers, the first quadrant of the sine function is approximated with piecewise-continuous linear segments. Simple control circuitry reconstructs a full sine wave by symmetry.

Basic principles of DDFS are presented, and a comprehensive review of existing work is made. A spectral analysis of the linear interpolation of the sine function is given, from which a design procedure is derived for the number of segments required and the value of system parameters to achieve a desired spectral purity. New hardware-optimized single phase and quadrature DDFS architectures are presented, and their validity is supported by several designs implemented with three different technologies. The new design approach and architectures are compared with existing work and are shown to produce designs with significantly reduced complexity and power consumption for equal performance.

résumé

Langlois, Joseph Mathieu Pierre, Collège militaire royal du Canada, juillet 2002, Conception et Architectures Nouvelles pour Synthétiseurs Numériques de Fréquences

Directeur de thèse: Dr. Dhamin Al-Khalili

Les synthétiseurs numériques de fréquences (SNF) génèrent des signaux sinusoïdaux en phase unique ou en quadrature sur des largeurs de bandes de plusieurs centaines de mégahertz, avec une résolution inférieure à un hertz, des taux de sauts en fréquences inégalés, en maintenant une continuité de phase lors d'un saut. Ces caractéristiques font des SNF les synthétiseurs de choix pour une vaste gamme d'applications, incluant les communications à dispersion de spectre, les systèmes radars, les instruments de mesure, et le domaine de la radiodiffusion. Cependant, il est souvent difficile de concevoir un SNF atteignant une haute pureté spectrale, une fréquence d'horloge élevée et une faible consommation de puissance.

Cette thèse présente une approche originale pour la conception de SNF à sortie sinusoïdale avec convertisseurs de phase à amplitude employant l'interpolation linéaire. Dans un tel cas, on fait l'approximation du premier quadrant de la fonction sinus par des segments de droite, et un circuit de contrôle permet de reconstruire une onde sinusoïdale complète par symétrie.

Les principes de base des SNF sont présentés, et une revue détaillée des travaux existants est faite. Une analyse spectrale de l'interpolation linéaire de la fonction sinus est donnée, menant à une relation entre la pureté spectrale désirée, le nombre de segments nécessaires, et la valeur des paramètres du système. De nouvelles architectures de SNF optimisées pour réalisation matérielle sont présentées, et leur validité est supportée par plusieurs exemples appliqués à trois différentes technologies. La nouvelle approche de conception et les nouvelles architectures sont comparées aux travaux existants, démontrant que la complexité et la consommation de puissance des synthétiseurs résultants est significativement moindre pour des performances égales.

Table of Contents

List of Figures xii

List of Tables xv

Acronyms xvi

Chapter 1 Introduction 1

1.1 Overview 1

1.2 Motivation 3

1.3 Research Objectives 4

1.4 Synopsis 4

Chapter 2 Principles of Direct Digital Frequency Synthesis 6

2.1 Direct Digital Frequency Synthesizer Architecture 6

2.1.1 General Description 6

2.1.2 Tuned Frequency and Effect of the Frequency Control Word 8

2.1.3 Frequency Resolution 8

2.1.4 Tuning Bandwidth 9

2.1.5 Tuning Latency 10

2.1.6 Phase Continuity upon Tuning 11

2.1.7 Frequency and Phase Modulation 11

2.1.8 Other Output Types 11

2.2 Phase to Sinusoid Amplitude Converter Complexity Reduction 12

2.2.1 Phase Angle Truncation 12

2.2.2 Exploiting Quadrant Symmetry 13

2.2.3 Quadrature Sinusoid Outputs Architecture 16

2.3 Output Noise 17

2.3.1 Noise Due to Amplitude Error 17

2.3.2 Noise Due to Phase Truncation 20

2.3.3 Phase Noise 20

2.3.4 Noise Reduction Techniques 21

2.4 Clock Output DDFS 22

2.5 System Performance Parameters and DDFS Complexity Metrics 23

2.5.1 Performance Parameters 23

2.5.2 Hardware Implementation Complexity Metrics 25

Chapter 3 State of the Art in Phase to Sinusoid Amplitude Conversion 27

3.1 Angular Decomposition 27

3.1.1 Tierney et al.: Sum of Angles and Multiplication 27

3.1.2 Hutchinson: Sum of Two Angles, Approximations and Addition 30

3.1.3 Sunderland et al.: Sum of Three Angles 30

3.1.4 Nicholas et al.: Exhaustive Search 32

3.1.5 Goldberg: Auxiliary Functions 33

3.2 Sine Amplitude Compression 33

3.2.1 Nicholas et al.: Sine-Phase Difference Algorithm 35

3.2.2 Yamagishi et al.: Double Trigonometric Approximation 35

3.2.3 Sodagar and Lahiji: Parabolic Approximation 37

3.2.4 Comparison 38

3.3 Angle Rotation Based Methods 40

3.4 Polynomial Approximations 41

3.4.1 Freeman's Architecture 42

3.4.2 Bellaouar et al.: 1st Degree Taylor Series 43

3.4.3 Liu et al.: Linear Segments of Unequal Lengths 45

3.4.4 Curticăpean et al.: Angular Decomposition Revisited 46

3.4.5 Weaver and Kerr: 2nd degree Taylor Expansion 47

3.4.6 Fanucci et al.: Quadratic Approximation 47

3.4.7 De Caro et al.: Optimized Polynomials, 2nd and 3rd Degree 48

3.4.8 Sodagar and Lahiji: Simplified 4th Degree Polynomial 49

3.4.9 Palomäki and Niitylahti: High Degree Chebyshev Polynomials and Taylor Series 50

3.5 PSAC-DAC Combinations 51

Chapter 4 New Amplitude Compression Technique 53

4.1 Review of Amplitude Compression 53

4.2 New Approximation and Architecture 54

4.2.1 Linear Interpolation Scheme 54

4.2.2 Design Example 56

4.2.3 New Architecture 57

4.3 Comparison of the New and Existing Techniques 58

4.4 ROM-less Performance 60

Chapter 5 Analysis and Design of DDFS Based on Linear Interpolation 61

5.1 General System Architecture 61

5.2 Output Spectral Analysis 63

5.3 Upper Bound on SFDR given the Number of Linear Segments Used 67

5.4 Linear Interpolation Coefficients Selection 73

5.4.1 Selection Criteria for h(n), n = 3, 5, 7, ..., 4s – 3 73

5.4.2 Acceptable Range of Values for g(1) Given a Desired SFDR 77

5.4.3 Selection Criteria for g(n), n = 1, 3, 5, 7, ..., 2s – 1 78

5.5 Design Procedure 79

Chapter 6 New DDFS Architectures Based on Linear Interpolation 82

6.1 New Architectures 82

6.1.1 General Description 82

6.1.2 Addend Alignment 85

6.1.3 Architecture Examples, Single Phase Output 88

6.1.4 Quadrature Outputs Architecture 90

6.2 High Level PSAC Complexity Metrics 93

6.2.1 System Coefficients and Parameters 94

6.2.2 Complexity Metrics 94

6.2.3 Number of ROM Bits 95

6.2.4 Number of 2:1 Multiplexers 95

6.2.5 Number of Full Adders 96

6.2.6 Total Cost 96
6.3 Design Process and Design Examples 97
6.3.1 Design Space Exploration and Selection of Coefficient and Parameter Sets 97
6.3.2 Candidate Slopes 97
6.3.3 Establishing a List of Groups of Acceptable Slopes 99
6.3.4 Exhaustive Search on M, L, C and D and Selection 102
6.3.5 Output Data and Spectral Plots 104
6.4 HDL System Description 112
6.4.1 Automated VHDL Code Generation 112
6.4.2 Module Compiler Language™ Description 113
6.5 Implementation Data and Results 114
6.5.1 Implementation in Xilinx® FPGA 114
6.5.2 Implementation Details for 0.18 µm and 0.35 µm CMOS 115
6.5.3 0.18 µm Implementation Results 115
6.5.4 0.35 µm Implementation Results 118
6.6 Fabricated Designs 121
6.6.1 60 dBc Single Phase Design Description 121
6.6.2 Test Strategy and Procedure 122
6.6.3 Test Results 123
6.6.4 84 dBc Design 129
Chapter 7 Interpretation and Discussion 130
7.1 Comparison with State of the Art 130

7.2 Complexity Increase with SFDR Performance 133

7.2.1 Quadratic Relationship 133

7.2.2 High Level Metrics versus Implementation Data 134

7.2.3 0.35 μm vs 0.18 μm Data 135

Chapter 8 Conclusion 136

8.1 Summary of the Contribution 136

8.2 Completion of Research Objectives 136

8.3 Recommendations for Future Work 137

References 140

Bibliography 148

Vita 151

List of Figures

Figure 2-1 - Single-phase sinusoid output DDFS architecture 7

Figure 2-2 - Varying the output frequency by skipping samples 9

Figure 2-3 - Quarter-wave symmetry (from [60]) 14

Figure 2-4 - Quadrature DDFS Architecture (from [90]) 16

Figure 3-1 - Sunderland approach 31

Figure 3-2 - PSAC architecture based on amplitude compression 34

Figure 3-3 - Nicholas et al. approach - sine-phase difference algorithm 36

Figure 3-4 - Yamagishi approach - double trigonometric approximation 37

Figure 3-5 - Sodagar and Lahiji approach - parabolic approximation 38

Figure 3-6 - Sine function compression comparison – approximated curves 39

Figure 3-7 - Sine function compression comparison - residual error 39

Figure 3-8 - Bellaouar et al.'s architecture (simplified) 45

Figure 3-9 - Curticăpean et al.'s architecture (simplified) 47

Figure 4-1 - New Amplitude Compression DDFS Architecture 57

Figure 4-2 - Comparison of new and existing architectures – approximating functions 59

Figure 4-3 - Comparison of new and existing architectures – residual error 59

Figure 5-1 - General architecture of single-phase DDFS using linear interpolation 61

Figure 5-2 - Jump in f at a discontinuity 65

Figure 5-3 - First quadrant of f defined by four linear segments 67

Figure 5-4 - Upper bound on SFDR as a function of the number of linear segments used........... 73

Figure 5-5 - Two limit conditions for acceptable intervals for g(n), with g(n) = g(4s – n)........... 75

Figure 6-1 Novel single-phase multiplier-less DDFS architecture, general format 83

Figure 6-2 - Decomposition of the phase angle word 85

Figure 6-3 - Addend alignment, general case... 86

Figure 6-4 - Addend alignment, example 1.. 87

Figure 6-5 - Addend alignment, example 2.. 87

Figure 6-6 - Addend alignment, pre-addition optimization.. 88

Figure 6-7 - Novel multiplier-less DDFS architecture, four segments, slopes with two digits..... 89

Figure 6-8 - Novel multiplier-less DDFS architecture, eight segments, slopes with two digits ... 89

Figure 6-9 - Novel multiplier-less DDFS architecture, eight segments, slopes with three digits . 90

Figure 6-10 - Coefficient conversion for quadrature DDFS... 92

Figure 6-11 - Novel multiplier-less QDDFS architecture, eight segments, slopes with two digits93

Figure 6-12 - Estimated PSAC complexity versus SFDR 104

Figure 6-13 - Output data and error, 8 segments, 60.1 dBc SFDR design 106

Figure 6-14 - Typical spectral plot, 8 segments, 60.1 dBc SFDR design.................................... 106

Figure 6-15 - Output data and error, 16 segments, 71.7 dBc SFDR design 107

Figure 6-16 - Typical spectral plot, 16 segments, 71.7 dBc SFDR design.................................. 107

Figure 6-17 - Output data and error, 32 segments, 80.0 dBc SFDR design 108

Figure 6-18 - Typical spectral plot, 32 segments, 80.0 dBc SFDR design.................................. 108

Figure 6-19 - Output data and error, 32 segments, 84.2 dBc SFDR design 109

Figure 6-20 - Typical spectral plot, 32 segments, 84.2 dBc SFDR design.................................. 109

Figure 6-21 - Output data and error, 64 segments, 91.9 dBc SFDR design 110

Figure 6-22 - Typical spectral plot, 64 segments, 91.9 dBc SFDR design................................. 110

Figure 6-23 - Output data and error, 64 segments, 96.2 dBc SFDR design 111

Figure 6-24 - Typical spectral plot, 64 segments, 96.2 dBc SFDR design................................. 111

Figure 6-25 - FPGA implementation statistics 114

Figure 6-26 - Power vs area, 0.18 μm CMOS worst case library... 117

Figure 6-27 - Area vs SFDR, 0.18 μm CMOS worst case library... 117

Figure 6-28 - Area vs clock rate, 0.18 μm CMOS worst case library... 118

Figure 6-29 - Area vs SFDR, 0.35 μm CMOS typical case library... 120

Figure 6-30 - Power vs area, 0.35 μm CMOS typical case library... 120

Figure 6-31 - Die picture, 60 dBc single phase DDFS in 0.18 μm CMOS 121

Figure 6-32 - Net core power consumption.. 126

Figure 6-33 - Net core power consumption versus FCW input (fixed FCW) 127

Figure 6-34 - I/O power consumption versus FCW input (fixed FCW)...................................... 128

Figure 7-1 - Complexity and performance summary 132

Figure 7-2 - Comparison of PSAC complexity estimates 134

List of Tables

Table 2-1 - Output noise sequences, M = 3 18

Table 3-1 - Compression factor for number of sub-tables and ROM address width 29

Table 3-2 - Sine function compression comparison – maximum residual error 38

Table 4-1 - Comparison of new and existing techniques 58

Table 6-1 - Candidate slopes sets 98

Table 6-2 - Selection of a slope set, 8 segments case 99

Table 6-3 - Selected sets of parameters and coefficients 103

Table 6-4 - Synthesis results, 0.18 μm CMOS worst case library 116

Table 6-5 - Synthesis results, 0.35 μm typical case library 119

Table 6-6 - Static core power consumption 123

Table 6-7 - Timing characteristics, chips 1 and 2 124

Table 6-8 - Net core power consumption, random FCW 125

Table 6-9 - I/O power consumption, random FCW 128

Table 6-10 - Power characteristics, two-FCW frequency hopping 129

Table 7-1 - Complexity and performance comparisons 131

Acronyms

ASIC	Application Specific Integrated Circuit
CLB	Configurable Logic Block
CMC	Canadian Microelectronics Corporation
CMOS	Complementary Metal Oxide Semiconductor
CORDIC	Coordinate Rotation Digital Computer
CRC	Communications Research Center
CSD	Canonical Signed Digit
DAC	Digital to Analog Converter
DC	Direct Current
DDF	Direct Digital Frequency
DDFS	Direct Digital Frequency Synthesizer *or* Direct Digital Frequency Synthesis
DIP	Dual In-line Package
DRDC	Defense Research and Development Canada
FCW	Frequency Control Word
FFT	Fast Fourier Transform
FPGA	Field Programmable Gate Array
GaAs	Gallium Arsenide
HDL	Hardware Description Language
HMCS	Her Majesty's Canadian Ship

IEEE	Institute of Electrical and Electronics Engineers
IO	Input / Output
IP	Intellectual Property
LFSR	Linear Feedback Shift Register
LPF	Low Pass Filter
LSB	Least Significant Bit
MAE	Maximum Amplitude Error
MCL	Module Compiler™ Language
MESFET	Metal Semiconductor Field Effect Transistor
MSB	Most Significant Bit
NA	Not Applicable *or* Not Available
NCO	Numerically Controlled Oscillator
PSAC	Phase to Sinusoid Amplitude Converter
QDDFS	Quadrature DDFS
RMC	Royal Military College
ROM	Read Only Memory
SFDR	Spurious Free Dynamic Range
SiGe	Silicon Germanium
SOI	Silicon On Insulator
TTL	Transistor-Transistor Logic
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit

Chapter 1

Introduction

1.1 Overview

This thesis concerns Direct Digital Frequency Synthesis[1] and Direct Digital Frequency Synthesizers (DDFS[2]). This type of synthesizer offers unique performance characteristics and advantages over designs based on Phase-Locked Loop (PLL) techniques. These include output bandwidths measured in hundreds of megahertz, frequency resolution in the sub-Hertz range, a tuning latency limited to one or a few clock cycles, the ability to maintain phase continuity when tuning, and excellent stability.

Frequency synthesizers generate a waveform of a desired shape at a specific output frequency. The waveform is most often a sinusoid, but it can also be a square wave or a sawtooth. There is normally one input signal of a fixed frequency, known as the reference frequency or standard frequency. The output frequency is usually a rational multiple of the reference frequency, i.e. $f_{out} = (p/q) \times f_o$, where p and q are positive integers, and f_{out} and f_o are the output and reference

[1] In the literature, the terms "Direct Digital Frequency Synthesis", "Direct Digital Synthesis", "Numerically Controlled Oscillator" and "Table Look-up Frequency Synthesis" are used by different authors to mean the same thing. The first term, which appears to be the most popular, will be used in this text.

[2] The single acronym "DDFS" will be used to mean either Direct Digital Frequency Synthesis or Direct Digital Frequency Synthesizer when the context fully specifies which term is meant. Otherwise, "DDF Synthesis" or "DDF Synthesizer" will be used as appropriate.

frequencies, respectively. In indirect frequency synthesis, the output signal is produced from a secondary oscillator that is locked in phase to the primary frequency reference. In direct frequency synthesis, the output signal is obtained from the standard frequency by a combination of operations such as mixing, filtering, multiplication and division. The output signal's frequency is a function of the reference frequency, without the use of internal oscillators or feedback loops. Direct Digital Frequency Synthesis is a special case of direct frequency synthesis, with digital components alone being used.

The first DDFS architecture was introduced by Tierney, Rader and Gold in 1971 [92]. Realized with 85 discrete TTL packages, it was clocked at just under 820 kilohertz and consumed 12 watts of power. The frequency resolution was 12.5 hertz for an output bandwidth of approximately 400 kilohertz. The severe technology limitations of the time prevented the architecture from gaining wide popularity, and few papers on the subject were published during the following decade [29][93]. The high power consumption, low clock rates and high spectral spurs were seen as major drawbacks of the approach.

A few important DDFS papers were published in the early eighties. Weathley and Phillips [102] observed that the high spurious noise in DDFS output was produced by the periodic nature of the device. They proposed a noise reduction technique that appeared counter intuitive: it was based on the addition of a random noise pattern inside the architecture. Merghardt [62] then provided a first analysis of the output spectra for DDFS. A major breakthrough occurred in 1984 with the publication, by Sunderland et al. [87], of a paper reporting on a first monolithic DDFS realized in 3.5 μm CMOS/SOS technology. The chip was to be used in a spread-spectrum frequency hopping satellite communications system. Capable of generating sinusoids from DC to about 3 megahertz in steps of 7.15 hertz, the synthesizer dissipated only 300 milliwatts.

In 1987, Nicholas and Samueli [66] made an important contribution to the understanding of the spurious levels in DDFS output. Joined by Kim in 1988 [67], the same authors then introduced an optimized DDFS architecture that was to endure as the reference against which all other designs were compared for several years.

Many DDFS patents were issued during this time, showing a growing interest for the technology [103][28][22][77][32][37][101]. These have mainly dealt with noise reduction and with architectures leading to reduced system complexity.

Between 1989 and this day, more than 100 conference and journal papers on DDFS have been published. Dozens of patents have been issued. A monograph by Goldberg [26] has also been published, and Kroupa [44] edited a collection of important DDFS papers in 1999. Novel DDFS designs that advance the limits of clock frequency, output spectral purity and power consumption are a regular feature at the IEEE International Symposium on Solid State Circuits, and in the IEEE Journal of Solid State Circuits.

Direct Digital Frequency Synthesizers have been used in a wide range of applications, including spread spectrum frequency hopping communications [87][11][78][79], and electronic warfare and radar systems [81][2][104][1][89]. Their versatility has also made them ideal candidates in such diverse areas as test instrumentation [19][12][9], broadcasting [96][99], hydrogen maser receivers [61], particle accelerators [35], and medical engineering [24].

1.2 Motivation

The phenomenal growth in interest for DDFS has been largely driven by the corresponding growth in wireless communications in the last 20 years, since the synthesis of a sinusoid of a desired frequency is a fundamental requirement in most communication systems. This requirement includes up- and down-conversion through frequency mixing, modulation and demodulation. In frequency-hopping spread-spectrum wireless communication systems, a frequency synthesizer with excellent performance must be used. Requirements include short tuning latency with high precision, excellent output spectral purity, low power consumption and low complexity. The shift to all-digital systems has also favored DDFS research.

There has been extensive research collaboration between the Royal Military College of Canada (RMC), Defense Research and Development Canada (DRDC) in Ottawa and the Communications Research Center (CRC) into the design of digital down converters for radio and electronic warfare receivers. This has included work on filtering and decimation [13], complex multipliers [54], and quadrature demodulation [108][46]. The last major down-converter building block that had not yet been addressed was the local oscillator. It was therefore proposed by CRC to study DDFS architectures, with the initial aim of integrating such a device together into existing down converter research. This proposal was the seed that led to the research presented in this thesis.

1.3 Research Objectives

The following goals and objectives were established for this research:

- to discover innovative sinusoid generation techniques for use in DDFS in order to reduce system complexity, when compared with prior work, while maintaining superior output spectral characteristics and high data rates;
- to apply these techniques to the development of novel DDFS architectures suitable for fast frequency hopping, low power wireless communications systems;
- to define metrics to evaluate DDFS system performance and costs early in the design cycle;
- to simulate, implement, and test several DDFS designs in order to compare the proposed techniques and architectures with prior work; and,
- to develop a behavioral synthesis capability for a DDFS Intellectual Property (IP) block for inclusion into a general-purpose or tailored communication system.

1.4 Synopsis

This thesis is divided into 8 chapters including this introduction. Chapter 2 presents an overview of the traditional DDFS architecture, and it discusses its main operating and performance parameters. Chapter 3 presents the state of the art in Phase to Sinusoid Amplitude Converter (PSAC) architectures for use in DDFS. Chapter 4 presents a new method and architecture for ROM data amplitude compression. Chapter 5 presents a generalized architecture based on linear interpolation, analyzes its output spectrum, and provides a systematic design approach. Chapter 6 presents new hardware-optimized DDFS architectures exploiting the analysis and design procedure described in Chapter 5. Several design examples are given, together with implementation data in FPGA, 0.35 μm CMOS and 0.18 μm CMOS. Chapter 7 contains an interpretation and discussion of the results and a comparison with existing work. Conclusions are found in Chapter 8.

Chapter 2

Principles of Direct Digital Frequency Synthesis

2.1 Direct Digital Frequency Synthesizer Architecture

2.1.1 General Description

Most Direct Digital Frequency Synthesizers are derived from the architecture introduced by Tierney, Rader and Gold [92] in 1971. This architecture can be divided into three blocks: a phase accumulator, a Phase-to-Sinusoid Amplitude Converter (PSAC), and a Digital-to-Analog Converter (DAC) with a Low Pass Filter (LPF). The system has two inputs: a reference clock of frequency f_0 and a frequency control word (FCW). A single phase version of the architecture is shown in Figure 2-1.

The phase accumulator is made up of an adder and a register, both N bits wide. On every clock cycle, the phase accumulator integrates the value of the FCW, hence the value stored in the register increases with time, eventually overflows past zero, and the cycle starts again. The phase accumulator output is therefore a ramp whose slope is directly proportional to the FCW. After n clock cycles, the value stored in the phase accumulator register is equal to

$$\langle n \times FCW + p_0 \rangle_{2^N} \quad n = 0,1,2,3,... \qquad \textit{(2-1)}$$

where $\langle \ \rangle_{2^N}$ denotes the modulo-2^N operation, and p_0 is the phase accumulator's initial contents upon reset. This value is generally 0.

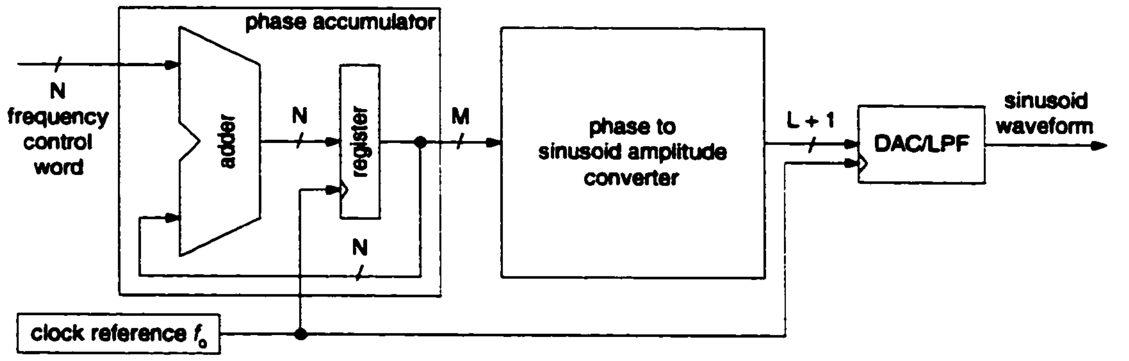

Figure 2-1 - Single-phase sinusoid output DDFS architecture

The phase accumulator's contents can be interpreted as a portion of a rotation around a unit circle. The equivalent phase angle is equal to

$$\varphi(n) = \frac{2\pi}{2^N}\langle n \times FCW + p_0 \rangle_{2^N} \qquad n = 0,1,2,3,... \tag{2-2}$$

The role of the PSAC is to calculate an approximation of the sine of this angle. The approximation is quantized with $L + 1$ bits, which can be viewed as L bits of amplitude and one of sign. Ideally, the PSAC output is equal to

$$A \times \sin \varphi(n) \tag{2-3}$$

where A is some amplitude that can be selected to maximize the synthesizer's output dynamic range. If the PSAC output is interpreted as a fraction, then the greatest value of A is $(2^L - 1) / 2^L$. In practice, the output of the PSAC includes a noise amplitude not indicated in Equation (2-3). This will be discussed in more detail below.

In its simplest form, the PSAC is implemented as a ROM Look-Up Table (LUT). The number of ROM words and the word width determine the angular and amplitude resolutions of the synthesizer, respectively. These in turn affect the synthesizer's output noise characteristics.

The role of the DAC and LPF is to convert the sequence of digital sinusoid samples into an analog waveform. These components are not present if the digital sequence is used directly in a digital system.

2.1.2 Tuned Frequency and Effect of the Frequency Control Word

As shown in Equation (2-2), the effective angle increment on every clock cycle is directly proportional to the ratio of the FCW to the number of states of the phase accumulator:

$$\Delta\varphi = \varphi(n) - \varphi(n-1) = 2\pi \times \frac{FCW}{2^N} \quad n = 1,2,3,\ldots \tag{2-4}$$

The angular frequency of the ramp output from the phase accumulator, which is also the synthesizer's output angular frequency, is therefore given by:

$$\omega_{out} = \frac{\Delta\varphi}{T_0} = f_0 \times \Delta\varphi = 2\pi f_0 \times \frac{FCW}{2^N} \tag{2-5}$$

in radians/second, where T_0 and f_0 are the period in seconds and the frequency in hertz of the clock reference, respectively. The synthesizer's output frequency is equal to:

$$f_{out} = \frac{\omega_{out}}{2\pi} = f_0 \times \frac{FCW}{2^N} \tag{2-6}$$

in hertz.

For small FCW (and hence small phase increment), more samples will be obtained from the PSAC per unit time, leading to a low frequency. For large FCW, the sine amplitudes will be scanned more quickly producing a higher frequency. This is shown in Figure 2-2, where it is assumed that the PSAC is implemented with a ROM LUT storing 16 samples of the sine function. Three curves are shown, corresponding to three values for the FCW. For the value 1, all samples are retrieved in order and the output frequency is the lowest possible above DC. For the value 2, every other sample is skipped. For the value 4, three samples are skipped per clock cycle.

2.1.3 Frequency Resolution

Equation (2-6) can be re-written as:

$$f_{out} = FCW \times \frac{f_0}{2^N} \tag{2-7}$$

The ratio of the clock reference to the number of states of the phase accumulator represents the frequency resolution of the synthesizer:

$$\Delta f = \frac{f_0}{2^N} \qquad \textbf{(2-8)}$$

This equation implies that the frequency resolution of the synthesizer can be made arbitrarily small by choosing as large an N as necessary. In practice, the width of the phase accumulator would be limited by system size and performance considerations. However, with a clock reference of 100 MHz and a typical phase accumulator width of 32 bits, the frequency resolution of the synthesizer is less than 0.025 Hz, which is adequate in most applications. Every additional bit in the phase accumulator improves the frequency resolution by a factor of two.

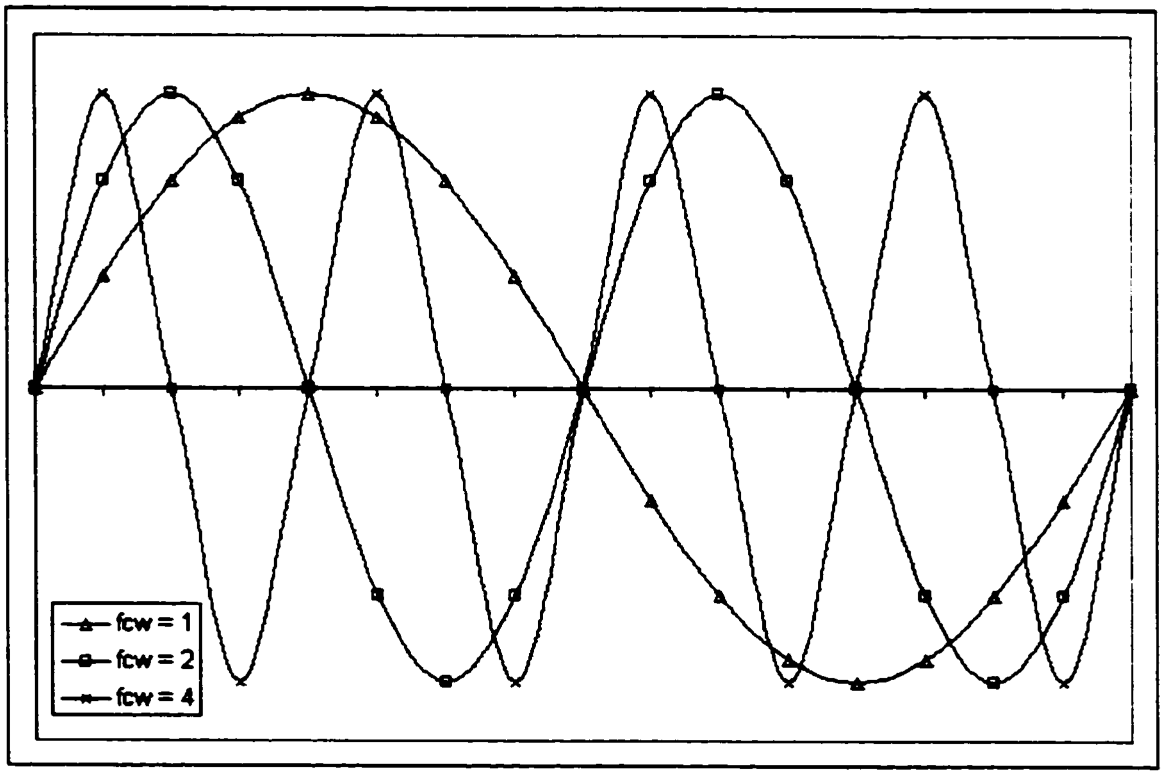

Figure 2-2 - Varying the output frequency by skipping samples

2.1.4 Tuning Bandwidth

When the FCW is equal to 0, the synthesizer's output is a constant equal to the sine amplitude of the present phase angle. The lowest non-zero frequency occurs when the FCW is set to one, and it is equal to the frequency resolution of the synthesizer:

$$f_{\min} = 1 \times \Delta f = \frac{f_0}{2^N} \qquad \textit{(2-9)}$$

The maximum theoretical frequency is dictated by the Nyquist criterion. Its upper bound is equal to half of the reference frequency:

$$f_{\max} < \frac{f_0}{2} = 2^{N-1} \times \Delta f \qquad \textit{(2-10)}$$

The value of the FCW must effectively be less than 2^{N-1}, and its width is limited to $N - 1$ bits. For quadrature output synthesizers (discussed below), however, the FCW may be expressed as an N-bit signed integer since output signals with negative frequencies are then possible.

In practice, for analog output DDF Synthesizers, the maximum frequency is limited to approximately $3f_0/8$ to simplify the design of the low-pass filters that follow the DACs. The tuning bandwidth therefore extends from DC to less than half of the reference frequency, as limited by practical filter design considerations.

2.1.5 Tuning Latency

The maximum rate at which the FCW can be changed and the time it takes for the output signal to stabilize on the new frequency are of prime importance in frequency-hopping spread-spectrum communications systems.

In a DDF Synthesizer, the tuning latency can be limited to a single clock cycle, which is extremely fast. The critical path is often placed in the phase accumulator, especially if it is very wide to accommodate fine frequency resolution (Equation (2-8)). Alternatively, the critical path may be in the PSAC. This can be the case for ROM architectures if the ROM is very large, and also for ROM-less architectures if the calculations inside the PSAC are complex.

In order to increase the synthesizer's tuning bandwidth, it is necessary to increase the reference clock's frequency. This may require pipeline registers to be included in the critical path. In such a case, the tuning latency of the synthesizer is increased by one clock cycle for every stage of pipelining added inside the synthesizer. Still, assuming that 10 levels of pipelining are used, the tuning latency is limited to 20 nanoseconds for a 500 megahertz clock.

2.1.6 Phase Continuity upon Tuning

DDF Synthesizers have the unique characteristic that phase continuity is maintained when changing the output frequency. This is a consequence of using a phase accumulator: when the FCW is changed, its new value is simply added to the existing sum in the phase accumulator. Thus, the present phase information is kept and the sinusoid at the new frequency starts from this same phase.

2.1.7 Frequency and Phase Modulation

An interesting aspect of DDF Synthesizers is that frequency and phase modulation can be easily imposed on the sinusoid output signal.

Since the FCW determines the instantaneous frequency of the output signal, frequency modulation is accomplished by simply varying the value of the FCW. For example, a "chirp" waveform can be easily synthesized by using a counter to generate a ramp as the FCW. The slope of this ramp determines the rate of frequency change within the chirp. Its bounds determine the chirp's lower and upper frequencies.

Phase modulation is accomplished by adding a modulating signal to the phase word prior to accessing the PSAC. For example, Quadrature Phase Shift Keying can be realized by adding one of four constant values to the phase word. The constant to be added is selected based on a control signal representing the data to be encoded.

2.1.8 Other Output Types

The architecture presented in Figure 2-1 can be modified so that the system output is not a sinusoid.

A popular form of DDF Synthesizer, often referred to in the literature as Numerically Controlled Oscillator (NCO), has a square wave (or clock) output. It can be realized simply by eliminating the PSAC, the DAC and LPF, and using the phase accumulator's MSB as the output signal.

A ramp can also be generated by passing a desired number of most significant bits from the phase accumulator directly to the DAC and LPF. Alternatively, a triangular wave can be generated by increasing and decreasing the phase accumulator contents on alternate half-periods. This can be realized by applying a one's complement operation to the phase accumulator output when the MSB is a '1'. The technique is further described in the following section, as it can also applied to sinusoid output synthesizers.

Finally, the PSAC can be replaced by a table storing samples from any periodic function. The synthesizer will then be able to reproduce this function with a varying frequency.

Unless explicitly specified, this dissertation only considers sinusoid output DDFS.

2.2 Phase to Sinusoid Amplitude Converter Complexity Reduction

The complexity of the PSAC has a significant impact on a DDF Synthesizer's performance and cost. This block has therefore been the subject of intense research. In this section, basic PSAC complexity reduction techniques are discussed. More advanced techniques will be described in Chapter 3.

2.2.1 Phase Angle Truncation

From the block diagram of Figure 2-1, three separate bus widths define the traditional DDF Synthesizer:

- N, the width of the adder, register and FCW;
- M, the width of the addressing lines to the PSAC; and
- $L + 1$ (for L bits of magnitude and one of sign), the width of the PSAC data.

From Equation (2-8), it is seen that increasing N improves the synthesizer's frequency resolution. In practice, it is advantageous to have a large N for fine frequency resolution. However, if the PSAC is implemented with a ROM Look-Up Table, using all N bits to address the ROM would lead to a very large memory. Accordingly, the phase value is normally truncated to retain only M

most significant bits. Memory requirements are reduced accordingly without sacrificing frequency resolution. However, this truncation causes a periodic amplitude error in the synthesized waveform which causes spurious noise in the output spectrum. This is discussed in greater detail in Section 2.3.2.

If the PSAC is implemented as a ROM Look-Up Table, total storage requirement are therefore equal to:

$$\text{ROM size (bits)} = (L+1)\times 2^{M} \tag{2-11}$$

This equation shows that the ROM size grows exponentially with the width of the truncated phase angle, and linearly with the amplitude resolution of the output data.

2.2.2 Exploiting Quadrant Symmetry

A very simple ROM size reduction technique for a DDF Synthesizer is based on exploiting the quarter-wave symmetry property of the sine function. The approach was first described by Tierney et al. [92], and was used by virtually every researcher thereafter. Notable exceptions are Saul and Mudd [80] and Saul and Taylor [81], for unexplained reasons, and Sodagar and Lahiji [83][85], for algorithmic reasons. The exploitation of quarter-wave symmetry normally has no effect on the selection of other ROM size reduction strategies, and should almost always be used.

As a first step, all angles must be mapped to the interval [0, 2π]. In DDFS, this is accomplished trivially by the modulo-2^N operation of the phase accumulator. The second step consists of mapping angles that lie in the third and fourth quadrants to angles in the second and first quadrants in order to calculate their sine amplitudes:

$$\sin(\theta) = \begin{cases} \sin(\theta) & 0 \le \theta < \pi \\ -\sin(\theta-\pi) & \pi \le \theta < 2\pi \end{cases} \tag{2-12}$$

The final step consists of mapping resulting angles that lie in the second quadrant to angles in the first quadrant:

$$\sin(\theta) = \begin{cases} \sin(\theta) & 0 \le \theta < \frac{\pi}{2} \\ \sin(\pi - \theta) & \frac{\pi}{2} \le \theta < \pi \end{cases} \qquad (2\text{-}13)$$

A very elegant way of implementing this process of reconstructing a full sinusoid from first quadrant samples alone was proposed by Tierney et al. [92]. It was illustrated concisely and clearly by Manassewitsch [60], Figure 1-20, and an adapted diagram is given in Figure 2-3. The method is based on the observation that the two MSBs of the phase word identify which quadrant an angle belongs to.

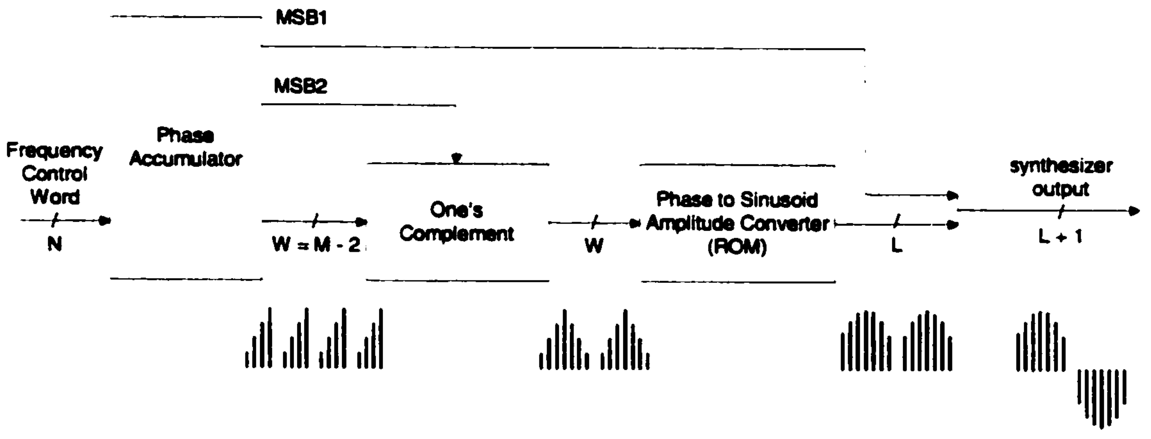

Figure 2-3 - Quarter-wave symmetry (from [60])

The phase angle MSB indicates whether an angles lies in the first two quadrants (a '0') or in the latter two quadrants (a '1'). Hence, the MSB readily provides the sign of the sine amplitude. Considering only the remaining portion of the phase angle word effectively maps an angle to the first and second quadrants (the operation θ – π in Equation (2-12)).

Inspecting the second MSB identifies whether the resulting angle is in the second (a '1') or first (a '0') quadrant. The operation π – θ in Equation (2-13) is implemented by 'counting down' from the phase angle π/2 instead of 'counting up' from 0. Hence, the second MSB controls a one's complementor that inverts the remaining phase angle bits as necessary.

The memory storage requirements are significantly reduced from what was specified by Equation (2-11). First, the number of necessary ROM words is divided by four. Further, the ROM word width is reduced by one bit, since only positive values must be generated and the sign information is provided directly by the phase angle MSB. For a LUT implementation, storage requirements are then:

$$\text{ROM size (bits)} = L \times 2^{M-2} \tag{2-14}$$

The so-called memory compression ratio is therefore equal to:

$$\frac{(L+1)\times 2^{M}}{L\times 2^{M-2}} = \frac{4\times(L+1)}{L} \tag{2-15}$$

There is a consequence to scanning first quadrant angles up or down with the help of the one's complementor. When calculating the data that must be stored into or calculated by the PSAC, a phase shift equal to $\pi/2^M$ must be introduced. This peculiarity was observed by Nicholas et al. [67] and described further by Vankka [97]. The reasoning is explained here.

Assume that the phase accumulator reset value is 0 and the FCW is 1. Also assume that the PSAC is implemented with a ROM. The first accessed ROM address is 0, and the last ROM address accessed in the first quadrant is $2^{M-2} - 1$. On the following clock cycle, the second MSB of the phase accumulator changes from 0 to 1 and all other bits are zero. The 1's complement block is activated and the $M - 2$ least significant phase accumulator bits are inverted, from zeros to ones. The accessed ROM address is therefore $2^{M-2} - 1$ twice in a row. A similar situation occurs when the contents of the phase accumulator change from the last angle of the second quadrant to the first angle of the third quadrant, but this time it is memory address 0 that is accessed twice in a row.

The data stored in address 0 should therefore be offset from $sin(0)$ by an angle equal to half of the angular resolution $\Delta\varphi$, which is the angle difference between the sinusoid samples of successive memory cells shown in Equation (2-4). Instead of the ROM storing the values of $\sin(0)$, $\sin(\Delta\varphi)$, $\sin(2 \times \Delta\varphi)$, ..., and $\sin(\pi/2 - \Delta\varphi)$, it should store the values of $\sin(\Delta\varphi / 2)$, $\sin(3 \times \Delta\varphi / 2)$, $\sin(5 \times \Delta\varphi / 2)$, ..., and $\sin(\pi / 2 - \Delta\varphi / 2)$.

2.2.3 Quadrature Sinusoid Outputs Architecture

The DDFS synthesizer introduced by Tierney et al. [92] had quadrature outputs. The authors noted that this allowed the bandwidth of the synthesizer to be doubled, as compared with a single phase case, for a given reference frequency. This is because the generation of quadrature outputs implies that it is possible to specify a negative frequency. Therefore, the FCW can be interpreted as an N-bit two's complement number. If the synthesizer's quadrature output is up-converted by some mixing process, then its bandwidth is effectively doubled from the single phase case, at no increase in reference clock frequency. However, Tierney et al.'s architecture required two accesses of the PSAC per clock cycle in order to compute the quadrature outputs. An alternative would have been to double memory and processing requirements.

The eighth-wave symmetry of the sine and cosine functions can be exploited instead, and two PSAC blocks are used. The blocks generate the sine and cosine for angles in the interval [0, π/4] only, and hence the complexity of each block is half of the complexity of the PSAC for the single-phase case. The overall computational complexity for calculating sinusoid amplitudes is therefore unchanged. This architecture was first proposed by Tan and Samueli [90], and it is shown in Figure 2-4.

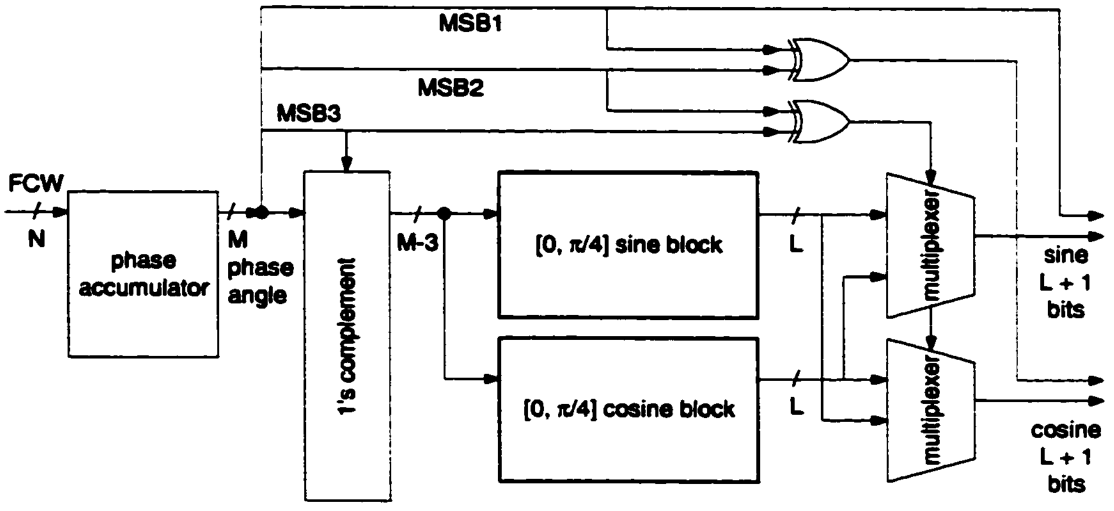

Figure 2-4 - Quadrature DDFS Architecture (from [90])

The three MSBs of the phase angle identify one of eight octants around the unit circle. The first MSB again provides the sign of the sine function, and, when XOR'ed with the second MSB, the result gives the sign of the cosine function. A second XOR gate accepts as inputs the second and third MSBs of the phase angle. Its output is a control signal for two 2:1 multiplexers which select a PSAC block output to reconstruct the sine and cosine waves. The third MSB controls the 1's complement block that implements the count-up or count-down process of the phase angle.

2.3 Output Noise

The noise performance of a DDF Synthesizer is normally specified by the Spurious Free Dynamic Range (SFDR), also known in the literature as Spectral Purity. The SFDR is defined as the ratio of the power in the desired frequency to the power in the greatest undesired frequency spur. It is normally expressed in decibels with respect to the carrier (dBc).

Several researchers provided descriptions of the noise spectra of DDF Synthesizers, including Tierney et al. [92], Mehrgardt [62], Kisenwether and Troxell [38], Nicholas and Samueli [66], Mattison & Coyle [61], Garvey and Babitch [23], Kroupa [40][41][42][43][45], and Torosyan and Willson [94].

In this section, we will consider the SFDR of the digital output sequence of the synthesizer. We are not considering analog noise introduced by the DAC or the LPF.

2.3.1 Noise Due to Amplitude Error

We will assume that the PSAC output sequence is invariant with time, i.e. the output amplitude is always the same for a given angle. Given the digital nature of DDFS, this is the default case. Without loss of generality, and to simplify the analysis, we will assume that quadrant symmetry is not used. The PSAC is therefore able to provide a sine amplitude estimate for 2^M distinct angles. Let $m \in \{0, 1, 2, \ldots, 2^M - 1\}$ be the M-bit phase angle that is passed to the PSAC. The PSAC output sequence, which we denote by $f(m)$, is equal to:

$$f(m) = A\sin\left(\frac{2\pi m}{2^M}\right) + \varepsilon(m) \quad m = 0,1,2,3,\ldots,2^M - 1 \qquad \textit{(2-16)}$$

where $\varepsilon(m)$ is the error sequence between the PSAC output and an ideal sine wave sequence of amplitude A expressed with infinite precision. The error sequence results from several factors, including the amplitude quantization error. Assuming that some processing is performed by the PSAC instead of using a ROM, any algorithmic error made in the calculation of $f(m)$ also affects the error sequence. The output frequency spectrum of the synthesizer therefore has two components: a pure sine wave of amplitude A, which is the desired system output, and some noise due to the amplitude error on the output samples. The noise spectrum can be calculated precisely by taking the Discrete Fourier Transform of one period of the noise sequence.

Let us assume that $N = M$, i.e. that no phase truncation is done. The case for $N > M$ is considered in the next section. The value of the FCW affects the order with which the noise sequence $\varepsilon(m)$ comes out of the synthesizer. For even values of FCW, not all noise sequence samples appear at the output. However, for odd values of the FCW, the repetition period of the noise samples will always be M, i.e. it will take exactly M changes in the phase accumulator before the sequence starts to repeat.

For example, assume that M = 3. There are four possible sequences for non-zero FCW values consistent with the Nyquist rate (i.e. FCW $\in$ {1, 2, 3}). They are listed in Table 2-1.

FCW	sequence	noise period
1	$\varepsilon(0), \varepsilon(1), \varepsilon(2), \varepsilon(3), \varepsilon(4), \varepsilon(5), \varepsilon(6), \varepsilon(7), \varepsilon(0), \varepsilon(1), \varepsilon(2), \ldots$	8
2	$\varepsilon(0), \varepsilon(2), \varepsilon(4), \varepsilon(6), \varepsilon(0), \varepsilon(2), \varepsilon(4), \ldots$	4
2	$\varepsilon(1), \varepsilon(3), \varepsilon(5), \varepsilon(7), \varepsilon(1), \varepsilon(3), \varepsilon(5), \ldots$	4
3	$\varepsilon(0), \varepsilon(3), \varepsilon(6), \varepsilon(1), \varepsilon(4), \varepsilon(7), \varepsilon(2), \varepsilon(5), \varepsilon(0), \varepsilon(3), \varepsilon(6),$	8

Table 2-1 - Output noise sequences, M = 3

We observe that for a value of FCW equal to 2, the noise sequence depends on whether the initial value of the phase accumulator is an even or odd number. In either case, only half of the noise samples corresponding to a FCW equal to one contribute to the output noise sequence. Consequently, the output spectrum is expected to vary with the initial phase accumulator value. We also observe that for values of FCW equal to 1 or 3, the noise sequence includes all noise samples but in a different order.

This observation has a fundamentally important consequence for the analysis and design of DDF Synthesizers. A change in FCW results in a change in the order of the noise samples out of the synthesizer and in a corresponding change in the position of discrete frequency components in the noise spectrum. However, the amplitudes and number of the noise frequency components do not change. Consequently, studying the noise spectrum corresponding to one value of FCW allows the exact prediction of the noise spectrum corresponding to all other values of FCW that produce noise sequences based on the same set of samples. The reader is referred to the papers by Nicholas and Samueli [66] and Vankka [97] for a full description of this principle and for a proof.

Consequently, in most cases it is advantageous to restrict the value of the FCW to odd numbers only. This way, the output spectrum of the synthesizer can be determined by generating a single output noise sequence and calculating its spectrum. It is therefore simple, during the design process, to evaluate and compare different architectures or architecture parameters. There is also a second advantage in the presence of phase truncation spurs, discussed in the following section.

Forcing an odd value of FCW is fairly simple, by hard-wiring a '1' as the LSB of the FCW. Instead of reducing the frequency resolution by a factor of 2, the phase accumulator can be made one bit wider to accommodate the hard-wired '1'. The effect on system complexity and timing is normally insignificant, especially for large N (i.e. greater than 20). A small frequency offset is introduced, however. The synthesizer's output frequency becomes:

$$f_{out} = \frac{\omega_{out}}{2\pi} = f_0 \times \frac{2FCW + 1}{2^{N+1}} = f_0 \times \frac{FCW + \frac{1}{2}}{2^N} \qquad (2\text{-}17)$$

in hertz.

Nicholas and Samueli [66] proposed an elegant technique to force an odd FCW. It consists of generating the sequence {0, 1, 0, 1, 0, ...} with the help of feedback loop composed of an inverter and a D-register, and feeding this sequence to the input carry port of the phase accumulator.

2.3.2 Noise Due to Phase Truncation

As discussed previously and shown in Equation (2-8), a large phase accumulator width is normally required to achieve fine frequency resolution. However, the complexity of the PSAC varies exponentially with the width of the phase word. Hence, as discussed in Section 2.2.1, the phase word is normally truncated from N to M bits, $N > M$. Thus, the frequency resolution of the synthesizer is unaffected. However, spurious noise is introduced in the synthesizer output.

This phenomenon has been well described by Nicholas and Samueli [66], Kroupa et al. [45], and, more recently, by Torosyan and Willson [94]. It has been shown that the worst case spurs are $6 \times M$ decibels down from the fundamental, provided that $N > M + 4$, i.e. that at least 5 bits are truncated.

2.3.3 Phase Noise

Phase noise is not often considered or reported for sine-output DDFS, for a number of reasons.

It is normally assumed that a DDFS behaves as a digital frequency divider, for which the output phase noise is equal to the input reference's phase noise, divided by the square of the relative output signal frequency (see the introductory paper for part VII, in [44]):

$$S_{\varphi,out} = S_{\varphi,in} \times \left(\frac{FCW}{2^N} \right)^2 \qquad \textbf{(2-18)}$$

where $S_{\varphi,out}$ and $S_{\varphi,in}$ are the output signal and reference frequency phase power spectral densities, respectively. In general, the phase noise of the reference frequency is low enough that the system performance is limited by the frequency spurs due to sample amplitude error and phase truncation. For an analog output DDFS, the limited slew-rate of the DAC is an important contributor to phase noise [61].

2.3.4 Noise Reduction Techniques

As discussed above, the digital nature of DDF Synthesizers implies that the error amplitude for any angle is invariant with time. For a fixed value of FCW, the same error samples are generated with a period that depends on the FCW, and the resulting sequence's frequency spectrum is generally not white. Again, if the FCW is restricted to an odd number, the noise sequence period is equal to the number of states of the phase accumulator.

In order to reduce the amplitude of the greatest spurs in the output frequency spectrum, the periodicity of the noise must be eliminated. There are two main methods of accomplishing this, and they consist of adding a pseudo-random sequence to the phase word before it is applied to the PSAC, or to the amplitude obtained from the PSAC. The pseudo-random sequence can be generated by a Linear Feedback Shift Register (LFSR) with a very long period. The effective error sequence period is therefore made very long, which results in a significant reduction in the amplitude of distinct frequency spurs. A drawback of the method, however, is that the overall noise floor is raised.

Several researchers have proposed variations and combinations of these two approaches. Wheatley et. al. [102] and Wheatley [103] were the first to propose this randomization process. Their work was directed to a clock-output DDFS instead of the sinusoid output type that has been discussed so far. In a clock-output (or square wave output DDFS), the most significant bit of the accumulator is used as a clock signal. In Wheatley's system, the pseudo-random sequence is added to the content of the phase accumulator. Jasper [32] applied Wheatley's technique to the sine output DDFS, with the consequence that the PSAC complexity, in terms of number of words and of word width, was reduced while maintaining the same level of SFDR. Kerr and Weaver [37], then Reinhardt et. al. [77], applied a random sequence to the output data from the PSAC, prior to driving the DAC. Finally, Flanagan and Zimmerman [21] combined the two approaches. These authors provided a detailed analysis of adding dithers to both the phase word and the sine amplitudes, and proposed a design achieving very good performance. The drawback, however, is the added system complexity due to the inclusion of two LFSRs (18 and 16 bits wide) and two 16-bit wide adders in the data path.

A mention should also be made of the work by O'Leary and Maloberti [72], who proposed a more simple approach. It consists of inserting a noise shaper, built by grouping an adder and a register, between the output of the phase accumulator and of the ROM. The adder's first input is the phase accumulator. The adder's high order output bits address the ROM, and the low order bits are registered and fed back to the adder's second input.

Vankka [98] presented a concise review of spur reduction techniques for DDFS. He then introduced three new methods that are variations of the work presented above. The first method involves the option of switching the extra '1' as the LSB of the phase accumulator on or off, depending on the value of the FCW. This way, for those values of the FCW that do not produce high spurious output, the noise floor is kept low. The second method involves high-pass filtering of the dither sequences prior to their addition to the phase word or to the sinusoid amplitude. The advantage of this approach is that while the noise floor is increased, as discussed above, most of the noise power is high in the frequency band. Consequently, a large portion of it lies in the stop band of the synthesizer's analog LPF, and is hence removed. The third method introduces filtering directly in the phase angle datapath or in the sine amplitude datapath.

2.4 Clock Output DDFS

In the literature, several papers claim to implement so called ROM-less DDF Synthesizers with very low complexity and high SFDR. However, when discussing ROM-less approaches, it is important to distinguish between sine-output and square-wave-output DDF Synthesizers. The latter are normally much simpler than the former.

The confusion in terminology can be traced back to some early DDFS papers [7][76]. When implementing a clock output DDFS, phase jitter tends to be a problem because, for most values of the FCW, the accumulator does not cycle to the zero value on every period. An early solution to this problem was to generate a sine wave with the help of a ROM LUT and a DAC/LPF block (i.e. the architecture discussed so far and shown in Figure 2-1), then to use a limiting circuit to produce a square wave. This was obviously a fairly expensive and complex way of generating a jitter-reduced clock, but it worked well since the phase noise performance predicted by Equation (2-18) could be attained provided that the DAC linearity was sufficiently high.

Several researchers have since published papers claiming high SFDR for ROM less approaches, but their architectures generate clock signals. Hence, the SFDR is only measured from DC to the third harmonic of the clock. Phase noise then becomes a much more important performance parameter than SFDR.

This dissertation is concerned only with sinusoid output DDF Synthesizers. Therefore, the interested reader is referred to several papers on jitter reduction for clock output DDFS: Nakagawa and Nosaka [64], Nosaka, Nakagawa, and Yamagishi [71] Yamagishi, Nosaka, Muraguchi and Tsukahara [107], Nieznanski [70], and Calbaza and Savaria [8].

2.5 System Performance Parameters and DDFS Complexity Metrics

In this section, we summarize the system performance parameters and complexity metrics that must be considered when comparing the relative merits of two DDF Synthesizers. The reader is also referred to Reinhardt et al. [76] and Manassewitsch [60] for a broader discussion on the performance of various types of synthesizers.

2.5.1 Performance Parameters

The following parameters define the performance of DDF Synthesizers:

1. Spurious Free Dynamic Range (SFDR): the power ratio between the greatest undesired frequency spur and the frequency of interest, normally expressed in decibels relative to the frequency of interest or carrier (dBc);

2. maximum clock rate: the maximum frequency of reference signal determines the synthesizer's tuning bandwidth, since it extends from DC to less than half of the reference signal's frequency;

3. frequency resolution: the frequency resolution is determined by the phase accumulator width and the reference signal's frequency; hence, the relative frequency resolution of two designs with different maximum clock rates can be compared by considering the widths of their phase accumulators only;

4. tuning latency: the tuning latency is determined by the number of pipeline stages in the data path; the latency is equal to $p + 1$ clock cycles, where p indicates the number of pipeline stages;

5. output type: the output is generally either a sinusoid or a square wave; for sinusoid output DDFS, single phase or quadrature outputs may be available;

6. output data format: exploiting quadrant symmetry, as discussed in Section 2.2.2, naturally favors the sign and magnitude format for the output data; however, another format is often required, such as unsigned representation for DACs; in low-SFDR DDFS, this conversion process can require a significant increase in system resources; and,

7. modulation and/or demodulation capability: modulation and demodulation capability, if present, must be well described as it adds a significant amount of complexity to the synthesizer.

There are other performance parameters that apply to all frequency synthesizers. For DDFS, however, they are secondary due to the nature of the system. For example, stability of a DDFS is normally a function of the reference clock only, although the DAC's performance could vary with time due to aging of its analog components. Other such parameters include the type of required references, the output level, the flatness, stability and accuracy of the output signal amplitude, and the possibility of synchronization with an external reference.

2.5.2 Hardware Implementation Complexity Metrics

When comparing two implemented DDF Synthesizers of equal or comparable performance, several metrics can be used. First, however, the technology must be fully specified in order for the comparison to be meaningful. For ASICs, this includes the nature of the manufacturing process (CMOS, GaAs, SiGe, SOI, etc.), the minimum feature size (0.6 µm, 0.35 µm, 0.18 µm, etc.) and the cell library used. For FPGA implementations, it includes the FPGA family and speed grade. In all cases, the operating voltage and temperature must also be specified as they have a significant impact on power consumption and maximum clock rate.

The metrics are:

1. <u>core area</u> (for ASIC technologies): when reporting the total core area used by the design, it is essential to specify whether the area includes only the total area occupied by the core cells, or if it is the total core area after cell placement and routing;
2. <u>core power consumption</u>: when reporting core power consumption, the results must be further qualified by specifying whether they have been obtained from simulation or by measurements of an implemented chip; in the case of simulation, it should also be specified whether the simulation includes interconnect parasitics extracted from placement and routing information;
3. <u>number of core cells or FPGA blocks</u>: the number of core cells or FPGA blocks used by the design can be readily obtained after synthesis; and,
4. <u>number of equivalent core gates and/or core transistors</u>: this popular design complexity metric can unfortunately be measured several ways; for FPGA implementations, the synthesis tool can report an estimate based on the number of used blocks and the relative utilization of each block's resources; for designs implemented in ASICs, there are four possibilities:
    a. the total area occupied by the core cells can be divided by the area of a reference cell, such as the smallest 2-input NAND gate, to obtain an equivalent gate count, and the equivalent number of transistors is obtained from multiplying this number by the number of transistors in the reference cell;

b. the list of core cells can be inspected, and the expected number of transistors for each cell can be estimated (2 for an inverter, 4 for a 2-input NAND gate, etc.); however, it must be noted that cell libraries typically include several cells realizing the same Boolean function albeit with different timing and power consumption characteristics;

c. the total core area can be divided by a design density parameter specified by the cell library vendor, providing an equivalent gate or transistor count; and,

d. some synthesis and placement tools provide the option of reporting the number of transistors in a design.

Chapter 3

State of the Art in Phase to Sinusoid Amplitude Conversion

It has already been stated that, in its simplest incarnation, the Phase to Sinusoid Amplitude Converter (PSAC) is realized as a ROM look-up table. However, its size tends to be large for good SFDR performance, increasing power consumption and system area, and decreasing the maximum clock rate. In Section 2.2, we considered two basic techniques to achieve ROM size reduction, to wit: phase accumulator truncation and exploiting the quadrant or octant symmetry of the sine function for single phase and quadrature output DDFS, respectively.

This chapter includes a review of the state of the art in PSAC architectures for DDFS. It also discusses techniques that have been proposed to implement this block with the least possible complexity for a given performance level.

3.1 Angular Decomposition

Significant memory savings can be made by segmenting the ROM into "coarse angle" and "fine angle" sub-tables, retrieving values from each on every clock cycle, and adding processing to derive a correct sine function approximation.

3.1.1 Tierney et al.: Sum of Angles and Multiplication

As a first step, Tierney et al. [92] and Tierney [93] suggested breaking the phase angle into a sum of sub-angles. A complex sample is kept in memory for the value of the complex exponential evaluated at each sub-angle. Since the exponential of a sum of values is the product of the exponentials of each value, a complex multiplication must be done on the ROM data. For example, if the phase accumulator angle θ is expressed as the sum

$$\theta = \sum_i \theta_i \qquad (3\text{-}1)$$

then the complex exponential corresponding to this angle can be calculated as

$$e^{j\theta} = e^{j\sum_i \theta_i} = \prod_i e^{j\theta_i} \qquad (3\text{-}2)$$

For the implementation, the partition of the phase accumulator word into sub-words is very simple. The phase accumulator word can be split into S slices of adjacent bits. As an example, assume that quadrant symmetry is used, the truncated phase word width is $M = 14$ bits and the ROM address bus is $W = 12$ bits wide. If W and S are not relatively prime, as in this case, then each slice can contain the same number of bits. Here, the ROM can be decomposed into 3 sub-tables, with the 4 most significant bits (bits 8 to 11) of the phase word addressing one table, bits 4 to 7 addressing a second table, and the 4 least significant bits (bits 0 to 3) addressing a third table. In this case, the most significant slice represents a "coarse" angle, the middle slice represents an angle with "middle" granularity, and the least significant slice represents a "fine" angle. The two complex multiplications represent "middle" and "fine" rotations from the coarse angle.

Following this approach, if the ROM is decomposed in S sub-tables, then $S - 1$ complex multiplications have to be performed. However, storage requirement can be significantly reduced. In the basic case, i.e. with no decomposition, the ROM contains 2^W words encoding complex sinusoid samples. By decomposing the ROM into S sub-tables, each sub-table can store as few as $2^{\lfloor W/S \rfloor}$ complex values, where $\lfloor\ .\ \rfloor$ indicates truncation of the fractional part. The total storage requirement is equal to $\sum_{i=1}^{S} 2^{w_i}$ complex values, where $W = \sum_{i=1}^{S} w_i$. The storage compression factor is equal to:

$$\frac{2^W}{\sum_{i=1}^{S} 2^{w_i}} \qquad (3\text{-}3)$$

Table 3-1 gives examples for different values of S and W.

It can be seen that the marginal improvement achieved from increasing the number of sub-tables provides an interesting advantage as the ROM address width *W* gets large. Considering the high cost of performing complex multiplication, the selection of the number of sub-tables must be made carefully. In fact, there are no examples in the literature of this method being applied, other than in the original paper by Tierney et al. [92] and the follow-on paper by Tierney [93].

		ROM address width W				
		8	9	10	11	12
number of sub-tables S	2	8.0	10.7	16.0	21.3	32.0
	3	12.8	21.3	32.0	51.2	85.3
	4	16.0	25.6	42.7	78.8	128.0
	5	16.0	28.4	51.2	85.3	146.3
	6	16.0	28.4	51.2	93.1	170.7
	7	16.0	28.4	51.2	93.1	170.7
	8	16.0	28.4	51.2	93.1	170.7

Table 3-1 - Compression factor for number of sub-tables and ROM address width

If quadrature outputs are not required, this method is less efficient. This is because complex values must be stored in all sub-tables, and complex multiplications are done. The result is therefore a complex value, good for quadrature outputs. For single phase output, it is not normally necessary to store complex values in the ROM.

Finally, it should be noted that storage of the fine sub-table need not be as large as that of the coarse sub-table. First, it is possible to reduce the number of samples by 2 by having positive and negative adjustment angles. Second, due to the finite precision of the calculations, many bits can be truncated without significant consequence on system performance. Details can be found in [92] and [93].

3.1.2 Hutchinson: Sum of Two Angles, Approximations and Addition

Hutchinson [29] proposed an improved coarse and fine ROM segmentation approach that does not require complex multiplication. All angles θ are split into a sum of a coarse angle *C* (the most significant bits of the phase accumulator) and a fine angle *F* (the least significant bits). The following trigonometric approximation is then made:

$$\begin{aligned}\sin(\theta) &= \sin(C+F)\\ &= \sin C\cos F + \sin F\cos C\\ &\cong \sin C + \sin F\cos C\end{aligned} \qquad (3\text{-}4)$$

The approximation is valid since *F* is small. For example, with (*M*, *C*, *F*) = (12, 8 , 4) bits, then the smallest value of cos*F* is cos(2 × π × 15 / 4096), which is approximately $(1-2^{-12})$.

The memory is partitioned into two sub-tables. One stores values of sin*C*, and the other stores values of the product sin*F* × cos*C*. All products are calculated in advance. Therefore, the only remaining operation is the addition of the retrieved values from the sub-tables. The paper includes an example that shows a ROM compression ratio of approximately 11.3.

3.1.3 Sunderland et al.: Sum of Three Angles

Sunderland et al. [87] extended Hutchinson's approach by splitting the phase word into three four-bit slices, *A*, *B*, and *C*, with *A* and *C* the most and least significant slices, respectively. An approximation for the sine of a sum of three angles is made as follows:

$$\begin{aligned}\sin(\theta) &= \sin(A+B+C)\\ &= \sin(A+B)\cos C + \cos(A+B)\sin C\\ &= \sin(A+B)\cos C + \cos A\cos B\sin C - \sin A\sin B\sin C\\ &\cong \sin(A+B) + \cos A\sin C\end{aligned} \qquad (3\text{-}5)$$

The approximation is valid given the relative values of *A*, *B*, and *C*. Figure 3-1 shows a simplified version of the resulting block diagram.

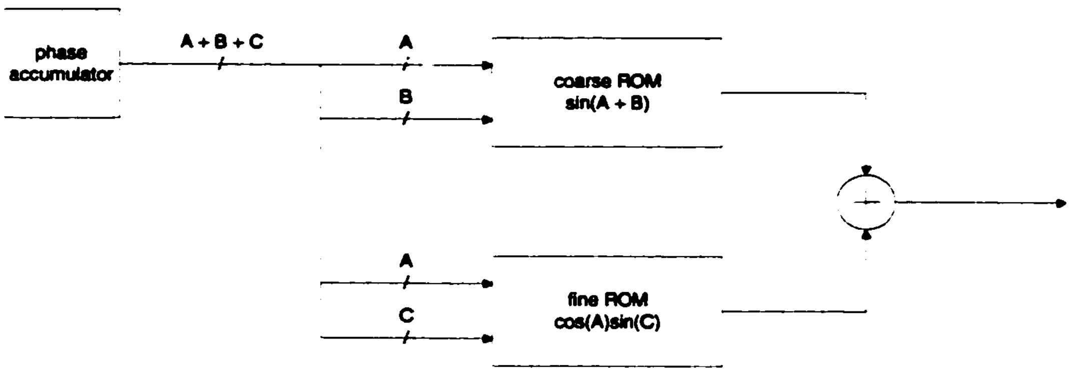

Figure 3-1 - Sunderland approach

Even though angles are expressed as a sum of three sub-angles, the ROM is divided into only two sub-tables. Address slices A and B address the coarse ROM which stores the value of $\sin(A + B)$. Address slices A and C address the fine ROM which stores the value of $\cos A \times \sin C$. The data from the two ROMs are added together to obtain the sine approximation.

In the paper, the authors present a specific design where the phase accumulator's 20 bits are truncated to 14 bits. Quadrant symmetry is used, leaving 12 bits to address the memory, and the slices A, B, and C are 4 bits wide each.

For this design, the authors introduced three minor corrections to reduce the total algorithm error further:

- $(\pi/2)(2^{-5})$, the average value of B, is added to A in the second term of Equation (3-5);
- the factor C includes a phase offset of $(\pi/2)(2^{-13})$ (initial phase angle discussed in Section 2.2.2); and,
- the coarse and fine terms are multiplied by $(1 - 2^{-11})$, the largest 11-bit fraction less than 1, to maximize the output dynamic range without having overflow.

These corrections are all pre-computed and the results are stored in the ROM. Finally, the authors state that they have verified that the maximum amplitude error is less than 1.5×2^{-11}. The compression ratio obtained is approximately 11.7. The overhead cost comes from the addition of the two ROM outputs.

3.1.4 Nicholas et al.: Exhaustive Search

Nicholas et al. [67] introduced an improvement over the Sunderland technique that does not involve trigonometric approximations. The ROM is again divided into coarse and fine sections, as shown in Figure 3-1, and they are addressed using the same scheme. The coarse ROM stores the best approximation of the sine function at each point. The fine ROM stores correction values to be added to each coarse ROM sample.

The fundamental difference of this approach resides in the selection of the fine ROM correction values. The authors propose that an exhaustive search be done to find values that minimize either the mean-square error or the maximum absolute error of the resulting approximations. Thus, Equation (3-5) is not followed. The authors observe that minimizing the mean-square error provides the lowest total spur energy, as predicted by theory, but that minimizing the maximum absolute error tends to reduce the value of the greatest spurs. An important advantage of this approach is that it can be applied to any function since it is not based on trigonometric identities.

The authors proposed a further refinement to their approach. Since the sine function is fairly linear over short distances, one can replace the addition of the coarse and fine ROM samples with addition or subtraction, as required. The symmetry in the fine ROM samples is therefore exploited, and the size of the fine ROM is halved. A consequence of this approach is that it involves greater complexity since a control signal must be derived to control the operation of the adder/subtracter. However, the authors state that this requires less than four logic gates for a 12-bit phase accumulator design.

During the coefficients search, Nicholas et al. derived some heuristics for the selection of the width of the segmented angles A, B and C. The authors also performed several simulations to select the lowest quantization width for the ROM data that met their performance requirement. Finally, it was observed that the amplitude of the "ideal" sine wave against which the approximation error was calculated for all angles could also be varied. The SFDR performance is improved by selecting an amplitude less than 1.0 with no consequence since only the nature of the ROM data is changed.

Finally, the principle of amplitude compression in DDFS was introduced in this paper. It is described in Section 3.2.

To summarize, this work by Nicholas, Samueli and Kim marked a fundamental change in the design of a PSAC for DDFS. The authors observed that the output SFDR could be optimized by performing an exhaustive search of the data to be stored in ROM. Several search parameters were varied during the search, and the coefficients achieving the desired SFDR with the least system cost were chosen. The same authors, as well as others from the same research group then applied these principles to implement several DDF synthesizers [68][69][91] and DDFS-based systems [90][78][79] that were the yardstick against which most other implementations have been compared.

3.1.5 Goldberg: Auxiliary Functions

Goldberg [26] proposed a general technique of storing "auxiliary functions" in small ROMs and adding their outputs together. The first ROM stores approximations of the sine function evaluated at the center of a reduced number of coarse intervals. The approximation can be made with a reduced number of bits. The other ROMs store progressively finer correction values to be added (or subtracted) to the first approximation. This approach can be viewed as a generalization of the Nicholas et al. technique. In his book, the author presents only one design example and he states that it achieves a lesser compression ratio than the one reported by Nicholas et al.. The main goal of the approach was to devise a way to avoid needing more than 8 bits storage per word for easier selection of off-the-shelf components. However, it offers interesting insight on how to best represent the sine function.

3.2 Sine Amplitude Compression

Even when angle decomposition is used, the dynamic range of the sine amplitude of the coarse angle is often the same as the dynamic range of the final approximation. By reducing this dynamic range, considerable memory savings can be obtained since fewer bits must be stored for each of the ROM samples.

Three techniques have been proposed to reduce the dynamic range of the ROM samples and hence the ROM size. They consist of calculating a sine amplitude approximation based on the value of the phase angle, then adding an amplitude correction stored in ROM. Depending on the accuracy of the approximation, the ROM data width is reduced by a certain number of bits, d, and the ROM size is reduced by a factor equal to $(L - d) / L$. This is illustrated in Figure 3-2.

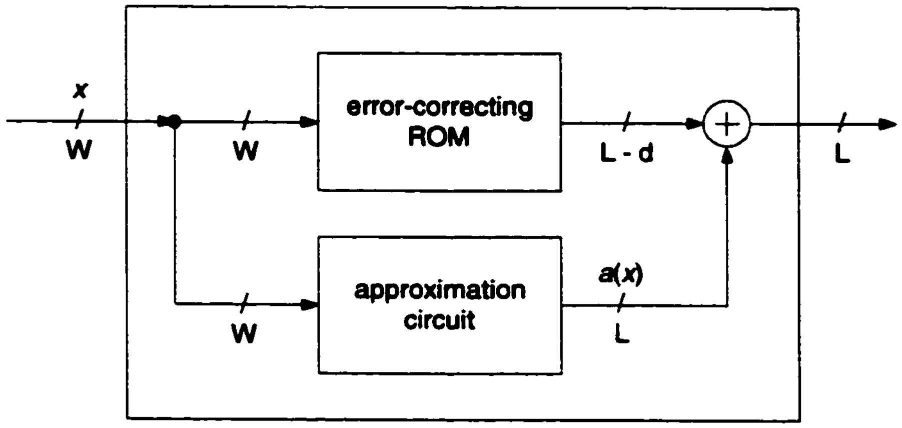

Figure 3-2 - PSAC architecture based on amplitude compression

Given a phase angle $x \in [0, 1[$, the error $\varepsilon(x)$ between an ideal sine wave of amplitude A and the output from the approximation circuit, $a(x)$, is given by:

$$\varepsilon(x) = A\sin(\frac{\pi x}{2}) - a(x) \qquad (3\text{-}6)$$

A typical value of A is $(2^L - 1) / 2^L$, which has the advantage of maximizing the synthesizer's output dynamic range.

The dynamic range of the error $\varepsilon(x)$ determines the number of amplitude bits in the ROM that can be saved compared to the case where no amplitude compression is used. For simplicity's sake, the approximation circuit is normally designed so that $\varepsilon(x)$ is strictly positive or negative. Then, to save 1 bit of storage per word, the maximum absolute value of $\varepsilon(x)$ must be less than 0.5. For 2 bits, it must be less than 0.25. For d bits, it must be less than 2^{-d}.

3.2.1 Nicholas et al.: Sine-Phase Difference Algorithm

In [67], Nicholas, Samueli and Kim introduced the "sine-phase difference algorithm". This "algorithm" exploits the monotonously increasing property of the sine function in the first quadrant. Function a in Equation (3-6) is the trivial identity function. Assuming an ideal sine amplitude A equal to 1, the ROM stores quantized approximations of

$$\varepsilon(x) = \sin(\frac{\pi x}{2}) - x \quad 0 \le x < 1 \tag{3-7}$$

The maximum absolute value of the error $|\varepsilon(x)|$ can easily be calculated. The phase angle x_m for which the error is maximal can be easily calculated by equating the first derivative of (3-7) to 0:

$$\begin{aligned} \frac{\partial \varepsilon(x_m)}{\partial x} &= 0 = \frac{\pi}{2}\cos(\frac{\pi x_m}{2}) - 1 \\ x_m &= \frac{2}{\pi}\cos^{-1}\left(\frac{2}{\pi}\right) \cong 0.5607 \end{aligned} \tag{3-8}$$

The maximum error can then be readily calculated:

$$\varepsilon(x_m) = \sin(\frac{\pi x_m}{2}) - x_m \cong 0.2105 \tag{3-9}$$

Hence, the amplitude compression is approximately equal to 4.75:1, so 2 bits of storage are saved for every ROM sample.

The overall block diagram of the Nicholas et al. approach is shown in Figure 3-3. It incorporates the sine-phase difference algorithm as well as the coarse/fine segmentation of the ROM described in Section 3.1.4.

3.2.2 Yamagishi et al.: Double Trigonometric Approximation

Yamagishi et al. proposed a simple yet effective improvement over the sine-phase difference algorithm, and called it a "double trigonometric approximation" [105][106]. Only one ROM is used to store samples, but its width is reduced by three bits. The technique consists of adding two terms to the ROM samples: the value of the phase accumulator (as for Nicholas et al.), and the positive phase of a triangular wave derived from the phase accumulator value. The synthesis of the triangular wave is very simple. Its value is equal to that of the phase accumulator, scaled by a factor of 0.25 (a right shift by two positions) with a bit-wise inversion depending on the third MSB of the phase accumulator. The approximation circuit of Figure 3-2 effectively implements the following equation defined by parts:

$$a(x)=\begin{Bmatrix} \frac{5}{4}\times x & 0\le x<\frac{1}{2} \\ \frac{3}{4}\times x+\frac{1}{4} & \frac{1}{2}\le x<1 \end{Bmatrix} \qquad (3\text{-}10)$$

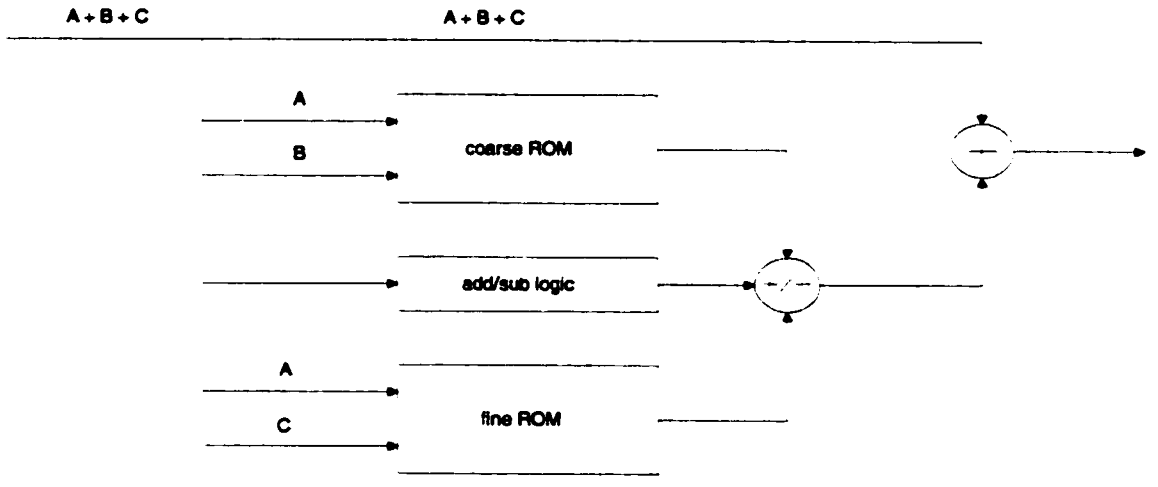

Figure 3-3 - Nicholas et al. approach - sine-phase difference algorithm

Calculation of the two straight lines requires a single adder and simple inversion logic, as shown in the block diagram of Figure 3-4. Two bits from the phase accumulator are used to control quadrant symmetry, leaving the 10-bit output shown in Figure 3-4. Nine of the 10 MSBs are retained and form a ramp. A triangular wave is added to this ramp signal. The triangular wave is derived from the 7 MSBs as follows. For the first half of the phase accumulator values, the MSB is 0 and the 7 MSBs are used as is. For the second half of the range, the MSB is one and the other 6 bits are inverted. Proper realization of Equation (3-10) is therefore implemented.

Following the process shown by Equations (3-8) and (3-9), it is easy to show that the maximum amplitude errors for the two linear segments are approximately equal to 0.0879 and 0.1163. The amplitude compression ratio is therefore approximately equal to 8.6:1, so three bits per sample are saved, which is one more than for Nicholas et al..

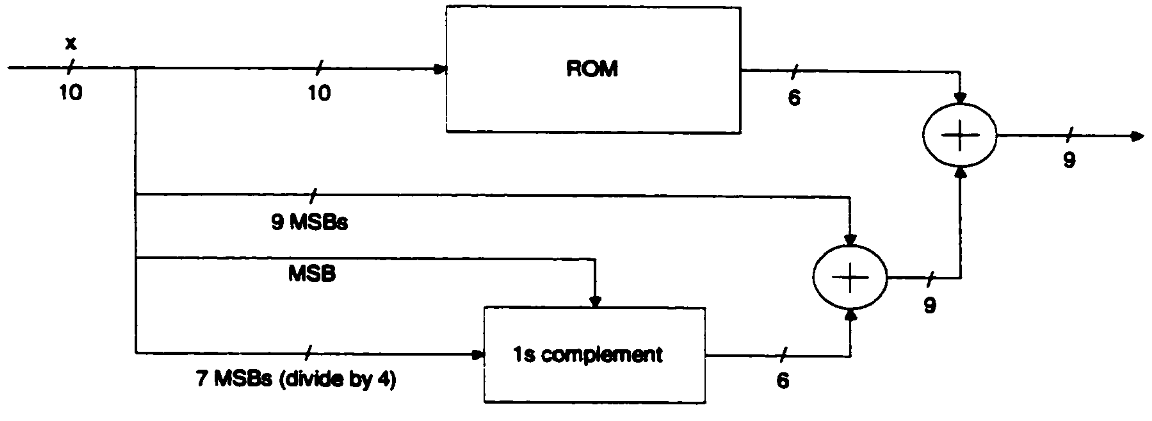

Figure 3-4 - Yamagishi approach - double trigonometric approximation

3.2.3 Sodagar and Lahiji: Parabolic Approximation

Sodagar and Lahiji suggested approximating the first two quadrants of the sine function with a parabola whose maximum and x-intercepts are the same as that of a sine half-period [83][85]. The approach is very simple, yet it provides accurate results. For a phase angle value $x \in [0, 2[$ representing an angle in the interval $[0, \pi[$, its sine is calculated according to:

$$\sin(\pi x) = x(2-x) + \varepsilon(x) \quad 0 \le x < 2 \tag{3-11}$$

where $\varepsilon(x)$ is the residual error from the parabolic approximation, and its value can be stored in a look-up table. The maximum value of the residual error is approximately 0.056, for an amplitude compression ratio of approximately 17.8:1, meaning that 4 bits per sample can be saved. The value of the maximum residual error must be calculated by numerical methods since no analytical approach can be taken to evaluate it.

The resulting approximation circuit architecture is shown in Figure 3-5. Half-wave rather than quarter-wave symmetry can be exploited in this version, and hence the one's complement block shown in Figure 2-3 is not necessary. However, if an error correcting ROM is used, then quarter-wave symmetry should be exploited to reduce the size of the ROM by half, and a one's complement block will also be required. The main disadvantage of this approach is its high computational cost due to the multiplier and to the two's complement block.

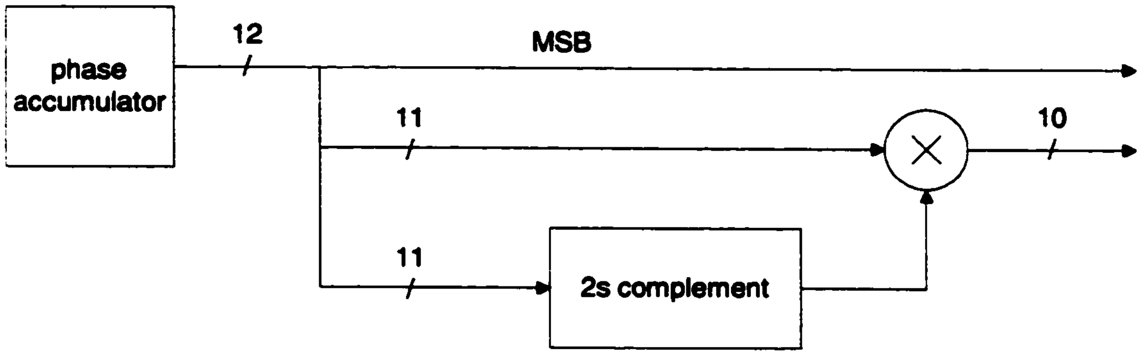

Figure 3-5 - Sodagar and Lahiji approach - parabolic approximation

3.2.4 Comparison

In this section, a comparison is made between the "sine-phase difference algorithm", the "double trigonometric approximation", and the "parabolic approximation". Table 3-2 compares the maximum residual error in each case. Figure 3-6 shows the three approximated curves together with the actual sine function for the first quadrant. Figure 3-7 shows the residual error that must be stored in the ROM for each case. For comparison purposes, the sign of the residual error has been inverted for the parabolic approximation. It can be seen that the maximum absolute residual error for the parabolic approximation is the least for first quadrant angles.

approach	maximum residual error
Nicholas et al. [67]: sine-phase difference algorithm	0.2105
Yamagishi et al. [105][106]: double trigonometric approximation	0.1163
Sodagar & Lahiji [83][85]: parabolic approximation	0.0560

Table 3-2 - Sine function compression comparison – maximum residual error

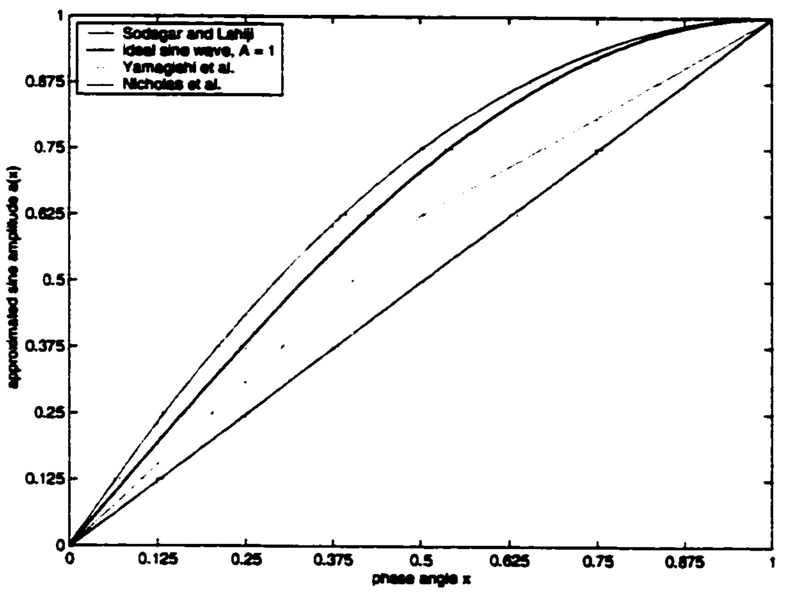

Figure 3-6 - Sine function compression comparison – approximated curves

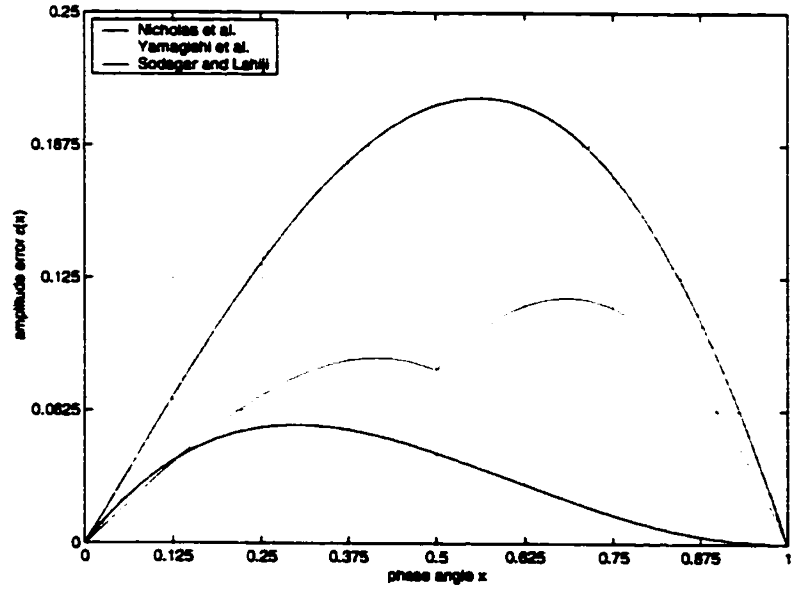

Figure 3-7 - Sine function compression comparison - residual error

3.3 Angle Rotation Based Methods

In the hope of eliminating the ROM table look-up altogether, several researchers have used a CORDIC [100] or other angle rotation-based algorithms for the PSAC. Many have succeeded in producing designs with high SFDR and reasonable complexity. An advantage of this approach is the ease with which the synthesizer can be converted to a synthesizer/mixer without the need for complex multiplication. The important disadvantage, however, is the increased tuning latency.

Madisetti et al. [57][58][59] reported a DDFS with 100 dBc of SFDR using a modified CORDIC. The angle-rotation algorithm is implemented as a multiplierless feedforward datapath, which allows easy pipelining and limits the accumulation of round-off errors. The modular architecture permits outputs of arbitrary precision by cascading enough angle rotation stages in the datapath. A fabricated design is presented with a tuning resolution of 0.0015 Hz at 100 MHz and a 100 dB spurious-free dynamic range. The complexity is estimated at 58000 transistors in 1.0 μm CMOS, and power consumption is measured at 14 mW/MHz. The tuning latency is 16 clock cycles.

Grayver and Daneshrad [27] noted that, in the traditional CORDIC, a very large number of stages is necessary to calculate the cosine and sine of an angle with good precision. However, the authors point out that once this calculation is finished, a recursive calculation can be done to obtain DDFS-equivalent operation. The authors therefore propose to use a traditional CORDIC structure to calculate the cosine and sine of the phase-increment angle, then to use a small recursive block with only two multipliers and two adders to generate the signal at the stable frequency. To improve frequency resolution an additional sub-rotation stage is used in the CORDIC line. This stage is controlled by the MSB of an accumulator, and adds a correction value every time the accumulator overflows. A correction circuit is also necessary to account for finite precision effects in the calculations, which cause unwanted effects in recursive architectures. An example architecture is given which provides 80 dBc SFDR, but the complexity of the implementation is not discussed.

Torosyan, Fu and Willson [95] presented a 300 MHz Quadrature DDF Synthesizer/Mixer based on a two-stage angle rotation algorithm. The implementation of the complex mixing process is embedded in the generation of the output frequency, and hence the overall design is very efficient.

CORDIC optimizations were suggested by Cardells-Tormo and Valls-Coquillat [10] in order to achieve higher spectral purity at lower hardware cost. A combination of a CORDIC and a LUT has also been suggested by Janiszewski, Hoppe and Meuth [30][31].

3.4 Polynomial Approximations

The sine amplitude compression techniques discussed in Section 3.2 effectively amount to polynomial approximations of the first quadrant of the sine function. The sine-phase difference algorithm and the double trigonometric approximation use 1st degree polynomials, i.e. linear segments, and the parabolic approximation uses a 2nd degree polynomial. Since the approximations made are not very precise, the residual error must be stored in a table or computed with the help of angular decomposition as explained previously.

Many successful PSAC design approaches have been based on making more complex polynomial approximations of the sine function for angles in the first quadrant. In such cases, acceptable SFDR performance can be attained without the need to store or calculate the residual error in a table. The general expression for such an approximation is

$$\sin\left(\frac{\pi x}{2}\right) \cong \begin{cases} \sum_{i=0}^{r} c_{0i}(x-x_0)^i & x_0 \le x < x_1 \quad (x_0 = 0) \\ \sum_{i=0}^{r} c_{1i}(x-x_1)^i & x_1 \le x < x_2 \\ \vdots & \\ \sum_{i=0}^{r} c_{ki}(x-x_k)^i & x_k \le x < x_{k+1} \\ \vdots & \\ \sum_{i=0}^{r} c_{(s-1)i}(x-x_{s-1})^i & x_{s-1} \le x < x_s \quad (x_s = 1) \end{cases} \tag{3-12}$$

where x is the phase angle, represented as a fractional number in the interval $[0, 1[$, r is the degree of the polynomial approximation, s is the number of piecewise continuous polynomial segments, the c_{ki}'s are the polynomial coefficients, and x_k is the lower bound of the k^{th} piecewise continuous segment.

In the literature, the PSAC design approaches that realize Equation (3-12) differ on four main points:

- the degree r, $r \geq 1$, of the polynomial approximation;
- the number s, $s \geq 1$, of piecewise-continuous polynomial segments;
- the position of the segment bounds x_k; and,
- the method by which the polynomial coefficients c_{ki} are calculated.

It is noted that, if the s segments are equal in length, then a segment's lower bound x_k is given by:

$$x_k = \frac{k}{s} \quad k \in \{0, 1, 2, ..., s\} \tag{3-13}$$

As a consequence, the subtraction $(x - x_k)$, for $x_k \leq x < x_{k+1}$ is accomplished trivially by truncating the $\log_2 s$ MSBs from x.

In this section, approaches based on polynomial approximations are reviewed. They are presented in increasing order of the highest polynomial degree r used.

3.4.1 Freeman's Architecture

To the best of the author's knowledge, the first description of a PSAC based on a polynomial approximation is the 1989 patent by Freeman of Rockwell International Corporation [22]. Reference to this patent is seldom made in the literature even though the proposed architecture was fundamentally different from all previous work. One conference paper by Kent and Sheng [36], of the same firm as Freeman, describes a DDFS implementation based on the patent. This paper was reprinted in the book edited by Kroupa [44], but is otherwise rarely cited by DDFS authors.

Freeman proposed a PSAC architecture decomposing the first quadrant of the sine function into $s = 16$ linear ($r = 1$) segments which are equal in length. The phase word, which is 12 bits wide, is decomposed into three parts: the first two MSBs are use for quadrant symmetry reconstruction, the following 4 MSBs (i.e. bits 9 down to 6) specify one of 16 segments, and the remaining 6 LSBs represent the result of the subtractions in Equation (3-12), which are angle offsets from the segment bounds x_k. One 16 words × 5 bits ROM stores the linear segment slopes c_{k1} and a second 16 words × 10 bits ROM stores the segment initial amplitudes c_{k0}. A 6 bits × 5 bits multiplier is used.

In addition to implementing Equation (3-12), an amplitude correction is applied for some angles. A third ROM stores correction values that reduce the Maximum Amplitude Error (MAE) between an ideal sine amplitude and the approximation made.

The segment coefficients are selected according to an approximation for the sine of a sum of two angles:

$$\begin{aligned}\sin\theta &= \sin(\theta_a + \theta_b) \\ &= \sin\theta_a \times \cos\theta_b + \sin\theta_b \times \cos\theta_a \\ &\cong \sin\theta_a + \theta_b \times \cos\theta_a + \varepsilon(\theta_a, \theta_b)\end{aligned} \qquad (3\text{-}14)$$

where $\sin\theta_a$ represents a segment's initial amplitude, and $\cos\theta_a$ represents a segment's slope. The correction values are represented by the function $\varepsilon(\theta_a, \theta_b)$.

The system was realized in a 0.6 μm GaAs MESFET technology, consuming 3 watts of power and occupying an area of 5.54 mm × 3.43 mm (218 × 135 mils). The post-DAC spectral purity was measured at –55 dBc up to a frequency of 245 MHz. The maximum system clock was 500 MHz, and the phase accumulator was 28 bits wide.

3.4.2 Bellaouar et al.: 1st Degree Taylor Series

University of Waterloo researchers Bellaouar, O'brecht, Fahim, and Elmasry proposed and patented a linear interpolation approach for the design of the PSAC [3][4][5]. It is similar to Freeman's approach, but differs in the following points. First, the authors presented a general architecture where the number s of linear segments and the design's bus widths can be selected to meet a desired SFDR. Second, their architecture facilitates the generation of quadrature outputs, exploiting the octant symmetry as proposed by Tan and Samueli [90] and discussed in Section 2.2.3. Finally, the selection of the polynomial coefficients is based on a Taylor series expansion of the sine function.

The Taylor series of a function f in the neighborhood of a point $x = a$ is given by:

$$\begin{aligned} f(x) &= \sum_{n=0}^{\infty} \frac{(x-a)^n f^{(n)}(a)}{n!} \\ &= f(a) + (x-a)f'(a) + \frac{(x-a)^2 f''(a)}{2} + \dots \end{aligned} \qquad (3\text{-}15)$$

The series converges for a certain convergence interval around point $x = a$, provided that all derivatives of the function exist at that point.

Bellaouar et al.'s architecture only computes the first two terms of Equation (3-15). The following approximation is made:

$$\sin\left(\frac{\pi x}{2}\right) \cong \sin\left(\frac{\pi x_k}{2}\right) + (x - x_k) \times \frac{\pi}{2} \cos\left(\frac{\pi x_k}{2}\right) \quad x_k \leq x < x_{k+1} \qquad (3\text{-}16)$$

where $x \in [0, 1[$ represents a phase angle in the first quadrant, and the x_k are segment bounds as described in the introduction to this section. The constant initial amplitude and slope coefficients in Equation (3-15) are stored in two small ROMs. The simplified resultant architecture is shown in Figure 3-8. The actual architecture is a variation of Figure 2-4 that allows the same core building blocks to be used for either a single phase or a quadrature output synthesizer.

A single phase output DDFS was implemented in a 0.8 µm CMOS process. It uses the equivalent of $s = 32$ segments and includes a 5 × 5 multiplier and an 8-bit wide adder/subtractor. The 9-bit output data is in sign and magnitude format, and the phase word is 12 bits wide. Although the PSAC can generate sinusoid sequences achieving 64 dBc of spectral purity, post-DAC measurements apparently indicated a 60 dBc limit, likely due to DAC non linearity and imperfect LPF. The authors claim to have achieved a memory compression ratio of 2.46:1 compared to the Nicholas architecture discussed previously [67], since only 416 bits of ROM storage are required.

However, processing overhead is higher, especially considering the 5 × 5 multiplier. Power consumption is significantly less than for previous implementations, at approximately 0.32 mW/MHz.

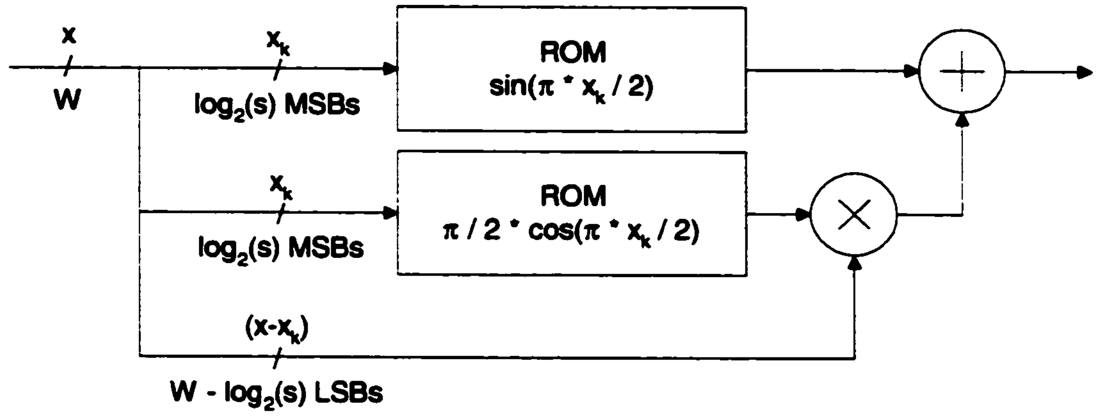

Figure 3-8 - Bellaouar et al.'s architecture (simplified)

3.4.3 *Liu et al.: Linear Segments of Unequal Lengths*

Liu, Yu and Tsao [55] proposed a PSAC architecture implementing the approximation of the first quadrant of the sine function with 12 piecewise-continuous linear segments of unequal lengths. The segments are divided into eight unequal sections, with the last two sections further divided into four sub-sections, for a total of 12 pieces.

In each section or sub-section, a first degree polynomial of the form $m \times x + b$ is used, where m is the slope and b is the y-intercept. The implemented equation is slightly different from the one shown in Equation (3-12) since angles are not processed according to their offset from segment bounds. This is a consequence of using unequal length segments. System complexity is thus increased since the simplification shown in Equation (3-13) cannot be applied.

A novelty in Liu et al.'s approach is that the architecture does not include multipliers. Instead, the segment slope are represented with at most four signed digits, and the multiplication is realized by adding shifted versions of the phase angle. The slopes include power-of-two factors as large as 2^0 and as small as 2^{-8}, implying a wide dynamic range for the additions and increased complexity.

The segment coefficients are selected to minimize the MAE, which is no greater than 7/4096 for all angles.

A single phase output DDFS was realized in 0.6 μm CMOS. Four levels of pipelining are used. The output is 10 bits wide in two's complement representation, and the phase word is 14 bits wide. Power consumption is approximately 1 mW/MHz and the SFDR is better than 67.6 dBc for a 10 MHz clock. The transistor count is low at 11721.

3.4.4 Curticăpean et al.: Angular Decomposition Revisited

Curticăpean and Niittylahti [14][15] and Curticăpean et al. [16] proposed an architecture based on angular decomposition, trigonometric identities and approximations for small angles. Although the angular decomposition is the same that was proposed by Hutchinson [29], described in Section 3.1.2, its implementation is analogous to a first degree polynomial interpolation and hence it will be discussed here. The architecture features high spectral purity, reduced ROM size, reasonable computational costs and reduced power consumption.

A simplified block diagram of the architecture is given in Figure 3-9. It realizes Equation (3-4). The actual architecture produces quadrature outputs, and is a variation of Figure 2-4 taking into account the particularities of the approximation that is made. For example, the ROM storing sin(F) is shared by the sine and cosine blocks.

A search is made to select the optimum angular separation between the coarse and fine angles C and F, and the bus widths throughout the system are also carefully selected to minimize system complexity while achieving high SFDR. As a final optimization, the multiplication is truncated to reduce its complexity. The resultant error is compensated by modifying the data stored in ROM.

The three papers show scaled variations of the same architecture, with SFDRs of 64, 85 and 96 dBc. Implemented in 0.35 μm CMOS, the complexities are approximately 4000, 7600 and 14000 transistors, and the power consumption is 0.10, 0.35 and 0.40 mW/MHz, respectively.

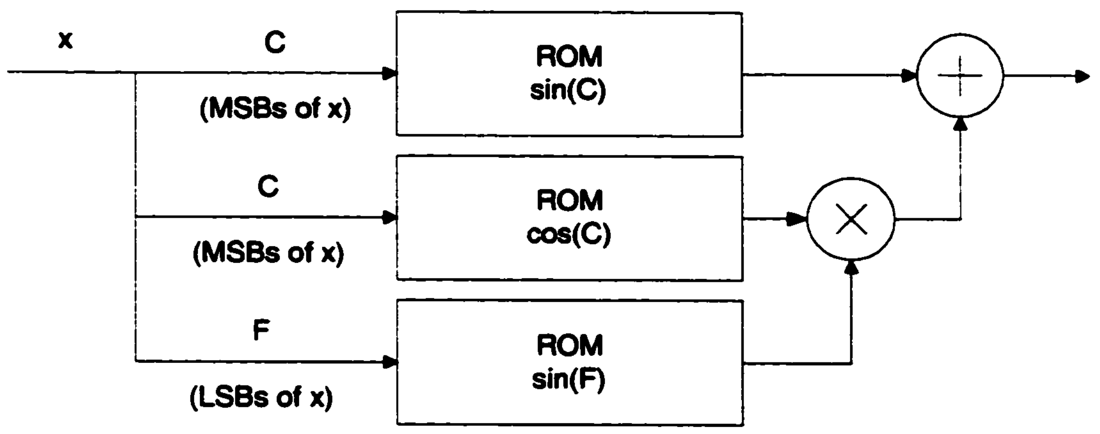

Figure 3-9 - Curticăpean et al.'s architecture (simplified)

3.4.5 Weaver and Kerr: 2nd degree Taylor Expansion

Weaver and Kerr [101] were the first to propose a Taylor series expansion of the sine function for DDFS. In the patent, the authors propose an architecture for calculating the sine of first quadrant angles by decomposing this quadrant in a number of polynomial segments. Over each segment, the architecture calculates a sine approximation using the first three terms of its Taylor series, as shown in Equation (3-15). The authors state that it is generally not necessary to calculate the fourth or other terms of the series, as their contributions would be negligible. However, the patent specifies that as many terms as necessary can be added if finer precision is desired.

A combination of ROMs are used. One ROM stores the amplitude, first derivative and second derivative of the sine function evaluated at the segment lower bounds x_k. The squaring and multiplication indicated in the third term of Equation (3-15) can be accomplished with the help of a look-up table implemented in a second ROM. In a variation of their approach, the authors propose reducing memory storage requirements by calculating the squaring operation and the multiplication in the third term of the equation, although this increases the computational burden.

3.4.6 Fanucci et al.: Quadratic Approximation

Fanucci, Roncella and Saletti [20] used four piecewise-continuous 2nd degree polynomials to approximate the first quadrant of the sine function, implementing Equation (3-12) directly for $r =$

2 and $s = 4$. Although the coefficients c_{ki} are given in the paper, their selection process is described in general terms only: the authors state that in order to maximize the SFDR, they selected polynomial coefficients that reduced the MAE. The coefficients are stored in a ROM, and two successive multiplications realize the approximation.

The coefficients are quantized with a large number of bits: 12 bits for c_{k0}, 9 bits for c_{k1}, and 5 bits for c_{k2}, resulting in ROM storage of $4 \times 26 = 104$ bits. The two multipliers' input ports are 11×5 and 11×9 bits wide. As a complexity measure of the architecture, the authors state that 175 full adders are required. The achieved SFDR is approximately 72 dBc. Finally, the paper contains a summary analysis comparing linear and quadratic approximations for the sine function.

3.4.7 De Caro et al.: Optimized Polynomials, 2nd and 3rd Degree

De Caro, Napoli and Strollo [18] introduced two designs implementing the quadrature architecture shown in Figure 2-4. The designs approximate the first octant of the sine and cosine functions with single ($s = 1$) 2nd and 3rd degree polynomials, respectively.

In each case, the selection of the polynomial coefficients was done to optimize spectral purity. Several selection processes were considered by the authors, including Taylor series, Chebyshev polynomials and Legendre polynomials. However, it was found that the best SFDR was achieved using non-linear Nelder-Mead simplex optimization [65]. The quantization of the coefficients is done with high precision, and hence the calculation of the polynomials is complex. For example, for the 2nd degree design, the sine function in the first octant is approximated by:

$$\sin(\frac{\pi x}{4}) \cong -0.001991 + 0.821689x - 0.109358x^2 \quad 0 \leq x < 1 \tag{3-17}$$

In order to realize these high precision calculations, the authors considered two architectures but selected a technique called CSD Hyper-folding as the most efficient.

The two designs were synthesized for a 0.35 μm CMOS library and a simulation was done including extracted post-layout parasitics. The 2nd degree design achieves an SFDR of 60 dBc with a 83 MHz clock and occupies 0.18 mm^2. Its power consumption is 0.187 mW/MHz. The 3rd degree design achieves an SFDR of 80 dBc at 88 MHz, occupies 0.44 mm^2, and consumes 0.436 mW/MHz. These designs are compared with CORDIC-based ones and shown to be much improved in terms of area and power consumption, for similar SFDR.

An interesting novelty of De Caro et al.'s approach is the specific goal of optimizing the spectral purity instead of minimizing the MAE by applying a standard interpolation process for the selection of the polynomial coefficients.

3.4.8 Sodagar and Lahiji: Simplified 4th Degree Polynomial

Sodagar and Lahiji [84][86] proposed an extension to the so-called parabolic approximation discussed in Section 3.2.3. They suggest using a single 4th degree polynomial by cascading two first order parabolic approximators shown in Equation (3-11). Defining the "first order parabolic approximation" by

$$pb_1(x) = 4x(1-x) \tag{3-18}$$

the "second order parabolic approximation" is given by

$$\begin{aligned} \sin(\pi x) &\cong pb_2(x) \\ &= pb_1(x) - 4K \times pb_1(x)(1 - pb_1(x)) \\ &= 64Kx^4 - 128Kx^3 + (80K-4)x^2 + (4-16K)x \end{aligned} \tag{3-19}$$

where $x \in [0, 1[$ represents a phase angle in the range $[0, \pi[$, and the constant K is approximately equal to 0.05591.

Implementing the last line of Equation (3-19) appears difficult in hardware. However, the second line can be implemented very efficiently. The second term in the second line is simply the output of a first order parabolic approximator, defined by Equation (3-18), multiplied by the constant K. Its input comes from a first order parabolic approximator whose input is the phase angle. Hence, the second line of Equation (3-19) is readily implemented in hardware by cascading two first order parabolic approximators, with a multiplication by K and a subtraction. Computational costs for the second order parabolic approximation are three multipliers, two two's complementors and a subtractor. In the paper, the authors note that multiplication by the factor K increases the complexity significantly, and hence they implement multiplication by this factor with 2 shifts and 2 additions (i.e. they quantize K with three signed digits).

The MAE for any angle is approximately equal to 9.81×10^{-4}, or close to 2^{-10}, giving 10 bits of amplitude precision for the sine amplitude. If no error-correcting ROM is used, the SFDR is approximately 64.8 dBc.

3.4.9 Palomäki and Niitylahti: High Degree Chebyshev Polynomials and Taylor Series

It is well known that any function, satisfying some very general conditions, can be approximated in the interval [-1, 1] by a linear combination of Chebychev polynomials. Palomäki and Niitylahti [74] used two single ($s = 1$) 4th degree Chebyshev polynomials to approximate the sine and cosine functions very accurately for angles in the interval [0, $\pi/4$]. Exploiting the first octant symmetry for sine and cosine, the architecture generates quadrature sinusoids (as discussed in Section 2.2.3). Restricting the polynomial interpolation to the interval [0, $\pi/4 \approx 0.79$] also greatly increases its precision, which would be degraded if it were applied to the interval [0, $\pi/2 \approx 1.57$]. Chebyshev polynomials therefore seem well adapted to this problem. Coefficients for the polynomials are calculated in advance from known equations.

The resulting PSAC architecture is fairly complex, including two squaring circuits, one multiplier, three two-operand adders and two 5-operand adders. Other multipliers are probably required for scaling operations, but the paper does not provide sufficient detail to assess this. Although the architecture is complex, a comparison table indicates that the design requires approximately 50% of the transistors of a design based on the Nicholas architecture for "similar" spectral purity (70 vs 84 dBc, which is in fact quite different), or 23 000 vs 58 000 transistors. The tuning latency is much improved, from 14 clock cycles to 5, mainly because of the absence of a ROM and its replacement with a regular data path.

A DDFS based on this architecture was realized in a 0.35 µm CMOS, 3.3 V, 4 metal, n-well process. Design complexity is estimated at 23000 transistors and the maximum clock frequency is 160 MHz. The spectral purity is better than 70 dBc.

The same authors also proposed using a single 5th degree Taylor series expansion [75]. The architecture is very similar to the one proposed in [74]. A 82.5 dBc SFDR design was implemented using a 0.35 μm CMOS library. The total area is 0.66 mm^2 and the power consumption is 0.109 mW/MHz. Eight pipeline levels allow a maximum clock of 320 MHz.

3.5 PSAC-DAC Combinations

Some researchers have had success combining the PSAC and the DAC in a single unit. The resulting post-DAC SFDR is similar to that achieved by more traditional architectures, but with a significant reduction in power consumption. However, this approach is only valid when a digital output is not desired.

Bjerede [6] patented a DDFS technique applicable when an analog sinusoid output is desired. The technique replaces the basic ROM and (linear) DAC combination by a non-linear DAC with no ROM. Spurious frequency components are suppressed by randomly dithering the rate at which the fine accumulator is incremented, as proposed by others and discussed in Section 2.3.4.

Mortezapour and Lee [63] also used a non-linear DACs to replace the ROMs and DACs in the traditional approach to DDFS. They refer to Bjerede's patent. Two quadrature designs were implemented in 0.5 μm CMOS technology, and they achieved better than 55 dBc for low synthesized frequencies (up to 500 KHz), with significant degradation past that point. However, power consumption is very low, at 0.16 mW/MHz. This is quite good, since it includes the DAC's power consumption as well. The main drawbacks of the approach is its low SFDR.

Jiang and Lee [33][34] also proposed similar designs, based on a non-linear DAC. The authors present a way of segmenting the DAC into sub-DACs, following angular decomposition discussed earlier. A design was implemented in 0.25 μm CMOS. The spectral purity is better than 50 dBc with a tuning bandwidth from DC to 113 MHz (300 MHz clock). Power consumption is low at 0.8 mW/MHz, including the DAC.

Shah and Colling [82] have also investigated a similar approach, based on an analogue memory array. With the addition of compensating circuitry, simulation results indicate that 59 dBc of SFDR could be attained for a 200 MHz clock with power consumption at 0.175 mW/MHz.

Chapter 4

New Amplitude Compression Technique

This chapter first summarizes existing work on amplitude compression as a means to reduce PSAC complexity for DDFS. A new sine approximation scheme is then introduced and a novel architecture is presented. A comparison between existing work and the new approach is then made, showing that the new architecture achieves significant ROM size reduction at low processing cost. Finally, the stand-alone performance of the approximation circuit is considered.

4.1 Review of Amplitude Compression

The three known sine amplitude compression techniques for DDFS were discussed in Section 3.2. In each case, the goal is to realize the PSAC in Figure 2-1 with the architecture shown in Figure 3-2 with the smallest possible ROM. A circuit calculates a coarse approximation to which a quantized error value, stored in ROM, is added. Equation (3-6) is re-stated here for clarity. Given a phase angle $x \in [0, 1[$, the error $\varepsilon(x)$ that must be quantized and stored in ROM is given by:

$$\varepsilon(x) = A\sin(\frac{\pi x}{2}) - a(x) \qquad \textit{(4-1)}$$

where $a(x)$ is the output from the approximation circuit and A is the amplitude of the ideal sine wave that needs to be reconstructed. A typical value of A is $(2^L - 1) / 2^L$, where L is the number of bits used to quantize positive sine amplitudes, and is typically the ROM width if the compression scheme is not used. This has the advantage of maximizing the output dynamic range.

The dynamic range of the error $\varepsilon(x)$ determines the number of amplitude bits in the ROM that can be saved compared to the case where no amplitude compression is used. To save 1 bit of storage per word, the maximum absolute value of $\varepsilon(x)$ must be less than 0.5. For 2 bits, it must be less than 0.25. For d bits, it must be less than 2^{-d}.

In order to keep the architecture as simple as possible, all error samples $\varepsilon(x)$ should have the same sign. Otherwise, the ROM data would need to be one bit wider, or a special control circuit would be required to control an adder/subtracter replacing the adder shown in Figure 3-2.

Finally, the approximation circuit should be kept as simple as possible. Obviously, if the complexity of the approximation circuit is increased sufficiently, then the maximum value of the error $\varepsilon(x)$ might be small enough that an error-correcting ROM is no longer required. This would result is a PSAC of one of the forms presented in Section 3.4. This progression is exemplified by the works of Sodagar and Lahiji ([83][85], then [84][86]).

4.2 New Approximation and Architecture

4.2.1 Linear Interpolation Scheme

The "sine-phase difference algorithm" [67] and "double trigonometric approximation" [105][106] are based on making a very coarse linear interpolation of the sine function with one and two piecewise continuous segments, respectively. These techniques can be seen as special cases of the implementation of Equation (3-12) with $r = 1$, with the output of the approximation circuit $a(x)$ given by:

$$a(x) = \begin{cases} y_0 + m_0(x - x_0) & x_0 \le x < x_1 \quad (x_0 = 0) \\ y_1 + m_1(x - x_1) & x_1 \le x < x_2 \\ \vdots & \\ y_k + m_k(x - x_k) & x_k \le x < x_{k+1} \\ \vdots & \\ y_{s-1} + m_{s-1}(x - x_{s-1}) & x_{s-1} \le x < x_s \quad (x_s = 1) \end{cases} \qquad (4\text{-}2)$$

where s is the number of linear segments, y_k and m_k are a segment's initial amplitude and slope, respectively, and x_k is a segment's lower bound.

As discussed in Section 3.4, the implementation of this equation is very efficient in hardware if the segments are equal in length and if the number of segments s is equal to a power of two. In such a case, the subtractions are done by a trivial truncation of $\log_2 s$ MSBs of x. However, when the goal is to reduce the dynamic range of the error $\varepsilon(x)$, it may be advantageous to consider unequal segments. This may increase the computational complexity, so a balance must be struck between the maximum value of $\varepsilon(x)$ and the complexity of the approximation circuit.

In the proposed linear interpolation scheme, the output of the approximation circuit $a(x)$ is given by:

$$a(x)=\begin{cases} y_0+m_0x & x_0\le x<x_1 \quad (x_0=0) \\ y_1+m_1x & x_1\le x<x_2 \\ \vdots & \\ y_k+m_kx & x_k\le x<x_{k+1} \\ \vdots & \\ y_{s-1}+m_{s-1}x & x_{s-1}\le x<x_s \quad (x_s=1) \end{cases} \tag{4-3}$$

where the symbols are as defined previously, except that y_k is now a segment's y-intercept.

The following heuristics can be used to guide the selection of the number of segments s and their slopes and y-intercepts:

- to simplify the design, $\varepsilon(x)$ must be positive for all x;
- the maximum value of $\varepsilon(x)$ in each segment must be below a certain target value corresponding to the number of bits d to be saved in the ROM;
- the least number of segments s should be used, to minimize control circuitry costs;
- the m_k coefficients should be represented as a sum of a small number of non-zero digits;
- x_k, the segment lower bounds, should be represented with as few most significant bits of the phase angle as possible, to reduce control circuitry costs; and,
- y_k, the segment y-intercepts, should be represented with as few most significant bits as possible, to reduce addition costs.

4.2.2 Design Example

As a first step in the present research, a goal of achieving a saving of 4 bits per ROM word was set. The resulting amplitude compression factor in this case is the same as the one achieved by the "parabolic approximation" [83][85]. Another goal was to achieve this amplitude compression factor at a much reduced computational cost.

The design heuristics listed above were used to select the number of segments and their coefficients. It was found that three linear segments were sufficient. The resulting implementation of Equation (4-3) is:

$$a(x) = \begin{cases} 0 + \dfrac{3x}{2} & 0 \le x < \dfrac{5}{16} \\ \dfrac{5}{32} + x & \dfrac{5}{16} \le x < \dfrac{3}{4} \\ \dfrac{1}{2} + \dfrac{x}{2} & \dfrac{3}{4} \le x < 1 \end{cases} \tag{4-4}$$

It is noted that this equation can be implemented with very low hardware cost since the approximation is calculated with a single addition in all three segments. Further, the partial product $x / 2$ can be shared between the first and third segments. The second segment's lower bound is expressed with four bits, and the third's is expressed with only two bits. The first segment's y-intercept is trivial, and the third's is expressed with a single non-zero digit.

Using the approach shown by Equations (3-8) and (3-9), and assuming that the sine waveform to be reproduced has amplitude $A \rightarrow 1$, it is easy to show that the maximum residual errors $\varepsilon(x)$ in the three segments are approximately equal to 0.0086, 0.0518 and 0.0477. These errors are less than 0.0625, corresponding to a compression by a factor of 16 and a reduction in storage of 4 bits per ROM word.

The total system complexity overhead of this scheme, including the error-correcting ROM, is equal to two additions and simple control circuitry to determine in which segment an angle x lies.

4.2.3 New Architecture

A DDFS architecture implementing Equation (4-4) is shown in Figure 4-1. In the Figure, it is assumed that $M - 2 \geq L$, that $L \geq 8$, and that quadrant symmetry is used. The approximation circuit consists of an adder whose inputs are selected by two multiplexers. Since there are three segments, one would expect 4:1 multiplexers to be used. However, since one partial product can be shared between two segments, and one of the operands for the first segment is zero, the multiplexers have only two inputs each.

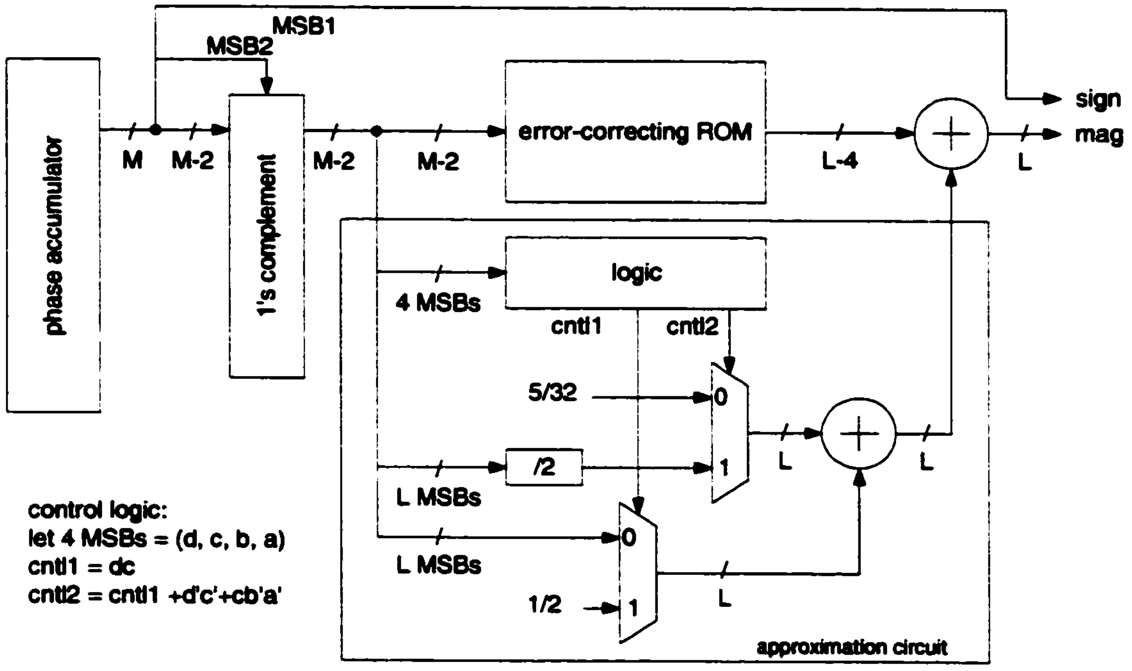

Figure 4-1 - New Amplitude Compression DDFS Architecture

A logic block generates the multiplexer control signals. Calculating the first control signal requires a simple two-input AND gate. The second control signal is a function of the first. The multiplexer inputs are either constants expressed with 1 and 5 bits, or trivial shifts of the phase accumulator. Two additions are required, including the one between the approximated sample and the error correcting ROM.

4.3 Comparison of the New and Existing Techniques

Figure 4-2 compares the approximation curve generated by the architecture of Figure 4-1 with that of existing techniques already reported. Figure 4-3 compares the residual error in each case. For comparison purposes, the error has been multiplied by -1 for the "parabolic approximation." It can be seen that the residual error remains below 0.0625, corresponding to an amplitude compression of 4 bits, both for the new technique and for the "parabolic approximation."

Table 4-1 gives a comparison of the new and existing amplitude compression techniques in terms of bits saved per ROM word and in terms of computational complexity. It can be seen that, although the number of bits saved is the same as for the parabolic approximation of Sodagar and Lahiji, the computational costs are considerably reduced.

technique	**$\varepsilon(x)_{max}$**	**bits saved**	**computational cost**
sine-phase difference algorithm [67]	$<.2080$	2	1 addition
double trigonometric approximation [105][106]	$<.1163$	3	2 additions, 1's complementor
parabolic approximation [83][85]	$<.0560$	4	multiplier, two's complementor
new technique	$<.0518$	4	2 additions, 2 multiplexers, and simple control circuitry

Table 4-1 - Comparison of new and existing techniques

From the data in Table 4-1, and assuming that the system output amplitude is quantized with L = 8 bits plus a sign bit, then the new technique realizes a reduction in ROM size of 33% compared with the sine-phase difference algorithm and of 20% with the double trigonometric approximation, at a modest increase in processing and control costs. The ROM size reduction is the same as for the parabolic approximation, at a significantly reduced processing cost. The parabolic approximation requires a full multiplier and a 2's complementor. The multiplier would be 8×8 for the case considered presently.

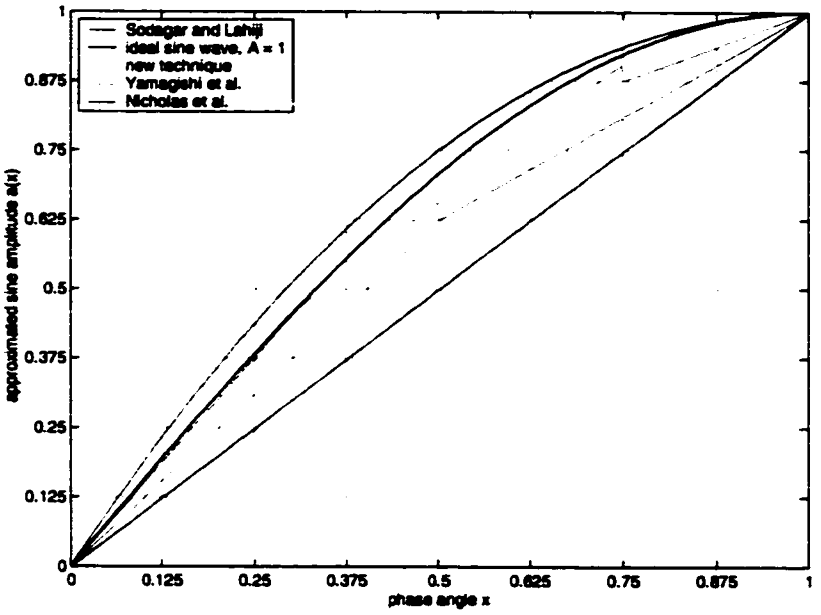

Figure 4-2 - Comparison of new and existing architectures – approximating functions

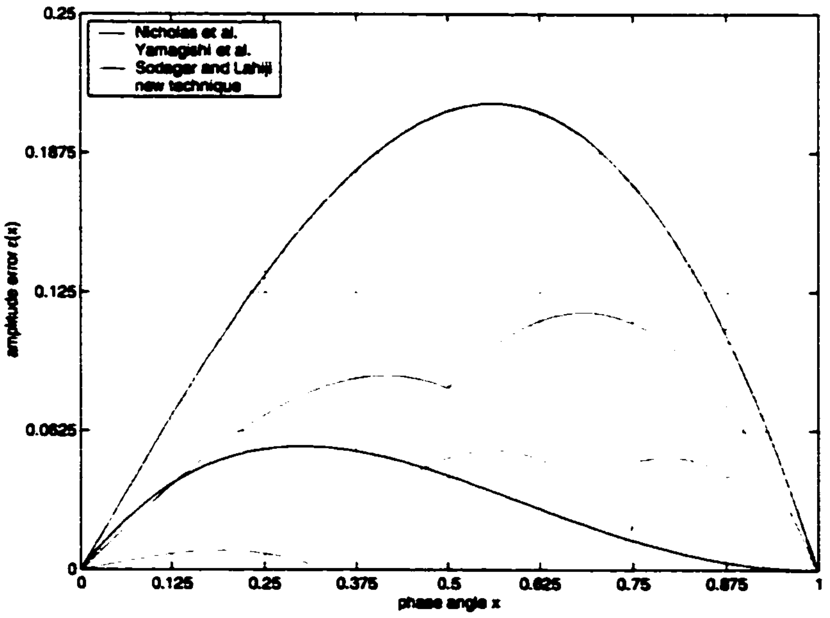

Figure 4-3 - Comparison of new and existing architectures – residual error

4.4 ROM-less Performance

Considering the architecture of Figure 4-1 with $M = 10$ and $L = 8$, and with no error-correcting ROM, a spectral analysis of the output sequence reveals that the SFDR is approximately 37.8 dBc. This is an improvement of more than 9 dB over the "parabolic approximation" architecture under the same conditions [85], which achieves approximately 28.7 dBc SFDR.

Eliminating the ROM also eliminates the requirement to keep the errors strictly positive or negative. It was observed that the SFDR could be improved to approximately 40.5 dBc by using the same architecture with a modification of the *y*-intercept coefficients of the 2nd segment [49]. The resulting approximation function is equal to:

$$a(x)=\begin{cases} 0+\dfrac{3x}{2} & 0\leq x<\dfrac{5}{16} \\ \dfrac{11}{64}+x & \dfrac{5}{16}\leq x<\dfrac{3}{4} \\ \dfrac{1}{2}+\dfrac{x}{2} & \dfrac{3}{4}\leq x<1 \end{cases} \qquad (4\text{-}5)$$

This improvement in SFDR leads to the following observation: when approximating the sine function with linear segments, the output SFDR can vary greatly with small modifications in the segment coefficients. Several questions can then be asked: given a required SFDR, what is the minimum number of linear segments that should be used? How should the segment parameters be selected to achieve this SFDR? Can the segment parameters be selected to optimize the hardware implementation also? These questions are considered in the following chapters.

Chapter 5

Analysis and Design of DDFS Based on Linear Interpolation

This chapter includes the first known attempt at a systematic analysis of the output spectrum of DDF Synthesizers based on linear interpolation. A generalized linear interpolation DDFS architecture is presented, and a detailed spectral analysis of its output sequence is provided. A closed form expression relating the achievable SFDR and the minimum number of linear segments that must be used is derived. A systematic coefficient selection process that maximizes the SFDR given a number of linear segments used is also derived, and a detailed design procedure is provided.

5.1 General System Architecture

A generic DDFS architecture with a phase to sine amplitude converter using linear interpolation is shown in Figure 5-1.

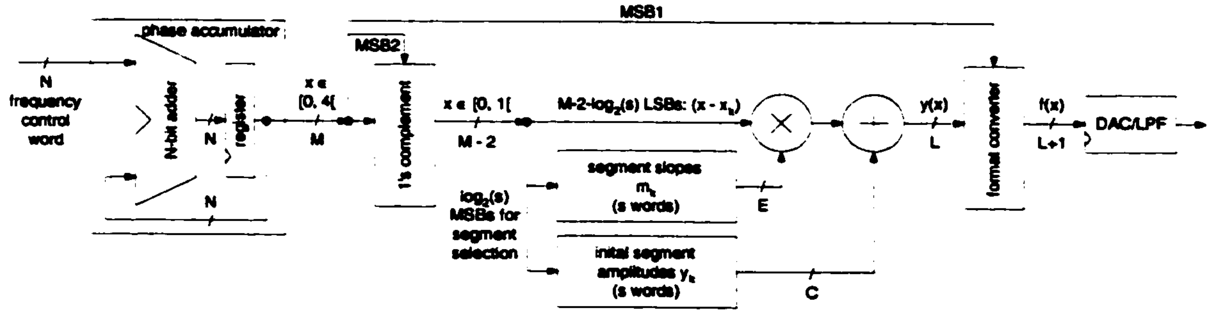

Figure 5-1 - General architecture of single-phase DDFS using linear interpolation

The least significant ($N - M$) bits from the phase accumulator are truncated, and the resultant M bits represent an angle in the interval [0, 2π[, scaled with a binary representation to a fractional number in the interval [0, 4[. Hence, the digital sequence f passed to the DAC is:

$$f(x) = A\sin(\frac{\pi x}{2}) + \varepsilon(x) \qquad (5\text{-}1)$$

where A is some amplitude and $\varepsilon(x)$ is an error sequence with period 4.

Quadrant symmetry is exploited. Therefore, the two MSBs of the truncated phase angle x indicate the quadrant in which this angle lies, and are used to control the format converter and the 1's complement operator. The remaining ($M - 2$) bits of x represent an angle in the interval [0, $\pi/2$[, quantized as a binary fraction in the interval [0, 1[. Signal y, quantized with L bits, is therefore the approximated sine amplitude for x in the first quadrant, and f is the approximated sine amplitude for any angle reduced to the scaled interval [0, 4[.

Function f has period 4, and even symmetry about points $x = 1$ and $x = 3$. Let s be the number of linear segments used to approximate the sine function in the first quadrant. Hence, f is defined in the interval [0, 1[by:

$$f(x) = \begin{cases} y_0 + m_0(x - x_0) & x_0 \le x < x_1 \quad (x_0 = 0) \\ y_1 + m_1(x - x_1) & x_1 \le x < x_2 \\ \vdots & \\ y_k + m_k(x - x_k) & x_k \le x < x_{k+1} \\ \vdots & \\ y_{s-1} + m_{s-1}(x - x_{s-1}) & x_{s-1} \le x < x_s \quad (x_s = 1) \end{cases} \qquad (5\text{-}2)$$

which is a special case of Equation (3-12) for $r = 1$. Assume that s is equal to an integer power of two. Hence, the $\log_2 s$ most significant bits of x can be used to select a slope m_k and an initial segment amplitude y_k, $k = 0, 1, 2, ..., s - 1$. Further, if the segments are equal in length, then the remaining $M - \log_2 s$ bits give the result of the subtraction ($x - x_k$), and

$$x_k = k / s \qquad (5\text{-}3)$$

5.2 Output Spectral Analysis

The spectral properties of function f will now be considered. If it exists, the Fourier series of function f is given by:

$$f(x)=a_0+\sum_{n=1}^{\infty}\left(a_n\cos\frac{2n\pi}{T}x+b_n\sin\frac{2n\pi}{T}x\right) \tag{5-4}$$

where the period T of f is 4 in the case considered here. The series converges if f is piecewise continuous and has left-hand and right hand derivatives at each point over the interval $[0, T]$. Its sum is f, except at any point x_k where f is discontinuous, and the sum of the series at such points is the average of the left-hand and right-hand limits of f at x_k. Since f has odd periodicity, it is represented by a Fourier sine series and all a_n coefficients are equal to 0. Assuming that f has quadrant symmetry, it can readily be shown that the b_n coefficients are given by:

$$b_n=\begin{cases}\dfrac{8}{T}\displaystyle\int_0^{T/4} f(x)\sin(\frac{2n\pi x}{T})dx & n\text{ odd}\\ 0 & n\text{ even}\end{cases} \tag{5-5}$$

The Fourier series coefficients of f can be found by using the definition of f from Equation (5-2) in Equation (5-5):

$$b_n=\begin{cases}\dfrac{8}{T}\displaystyle\sum_{k=1}^{s}\int_{x_{k-1}}^{x_k} f(x)\sin(\frac{2n\pi x}{T})dx & n\text{ odd}\\ 0 & n\text{ even}\end{cases} \tag{5-6}$$

The integral can be evaluated by parts :

$$\int_{x_{k-1}}^{x_k} f(x)\sin(\frac{2n\pi x}{T})dx=\frac{-T}{2n\pi}f(x)\cos(\frac{2n\pi x}{T})\Bigg|_{x_{k-1}}^{x_k}+\frac{T}{2n\pi}\int_{x_{k-1}}^{x_k} f'(x)\cos(\frac{2n\pi x}{T})dx \tag{5-7}$$

Since f may be discontinuous at x_k, the evaluation of the first term to the right of the equal sign in Equation (5-7) must be done on the *inside* of the interval. That is to say that the left-hand limit

$f(x_k^-)$ must be taken at x_k, and the right-hand limit $f(x_{k-1}^+)$ must be taken at x_{k-1}. This is the well-known method of integration by jumps [39].

Using the notation $C_k = \cos(\frac{2n\pi x_k}{T})$ and replacing (5-7) in (5-6) yields, for odd n,

$$\begin{aligned} b_n &= \frac{8}{T}\times\frac{-T}{2n\pi} \\ &\times\left[f(x_1^-)C_1 - f(x_0^+)C_0 + f(x_2^-)C_2 - f(x_1^+)C_1 + \ldots + f(x_s^-)C_s - f(x_{s-1}^+)C_{s-1}\right] \\ &+\frac{8}{T}\times\frac{T}{2n\pi}\sum_{k=1}^{s}\int_{x_{k-1}}^{x_k} f'(x)\cos(\frac{2n\pi x}{T})dx \end{aligned} \qquad (5\text{-}8)$$

The expression within brackets can be written as:

$$\begin{aligned} &-f(x_0^+)C_0 + (f(x_1^-) - f(x_1^+))C_1 + (f(x_2^-) - f(x_2^+))C_2 \\ &+\ldots + (f(x_{s-1}^-) - f(x_{s-1}^+))C_{s-1} + f(x_s^-)C_s \end{aligned} \qquad (5\text{-}9)$$

Noting that:

$$C_0 = \cos(0) = 1 \qquad (5\text{-}10)$$

$$C_s = \cos(\frac{n\pi}{2}) = 0 \qquad (n \text{ odd}) \qquad (5\text{-}11)$$

the expression is therefore equal to:

$$-(f(x_0^+) + j_1C_1 + j_2C_2 + \ldots + j_{s-1}C_{s-1}) \qquad (5\text{-}12)$$

where

$$j_k = f(x_k^+) - f(x_k^-) \qquad (5\text{-}13)$$

is the "jump" in f at x_k, as shown in Figure 5-2. If f is continuous at some x_k, then the corresponding j_k is obviously zero. It must be noted that j_k may take either positive or negative values.

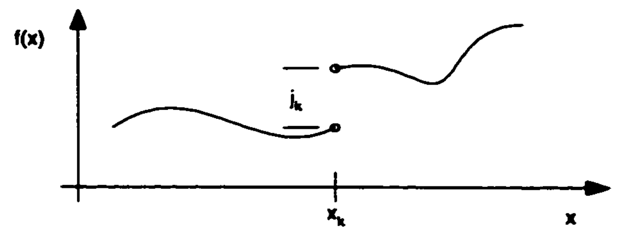

Figure 5-2 - Jump in f at a discontinuity

Therefore, for f satisfying the conditions given above, we have:

$$b_n = \frac{4}{n\pi}\left[f(x_0^+) + \sum_{k=1}^{s-1} j_k \cos(\frac{2n\pi x_k}{T}) + \sum_{k=1}^{s} \int_{x_{k-1}}^{x_k} f'(x)\cos(\frac{2n\pi x}{T})dx \right] \tag{5-14}$$

for odd n, and 0 otherwise.

The same procedure is now applied to the integral in (5-14). Integration by parts yields:

$$\int_{x_{k-1}}^{x_k} f'(x)\cos(\frac{2n\pi x}{T})dx = \frac{T}{2n\pi} f'(x)\sin(\frac{2n\pi x}{T})\Big|_{x_{k-1}}^{x_k} - \frac{T}{2n\pi}\int_{x_{k-1}}^{x_k} f''(x)\sin(\frac{2n\pi x}{T})dx \tag{5-15}$$

Again, since $f'(x)$ may be discontinuous at x_k, the evaluation of the first term to the right of the equal sign in Equation (5-15) must be done on the *inside* of the interval. We also note from Equation (5-2) that the second derivative of f is 0 and hence that the integral on the right of the equal sign in Equation (5-15) does not have to be evaluated. Using the notation $S_k = \sin(\frac{2n\pi x_k}{T})$ and the result from (5-15), the summed integral in (5-14) is equal to:

$$\sum_{k=1}^{s} \int_{x_{k-1}}^{x_k} f'(x)\cos(\frac{2n\pi x}{T})dx = $$
$$\frac{T}{2n\pi} \times \left[f'(x_1^-)S_1 - f'(x_0^+)S_0 + f'(x_2^-)S_2 - f'(x_1^+)S_1 + \ldots + f'(x_s^-)S_s - f'(x_{s-1}^+)S_{s-1} \right] \tag{5-16}$$

The expression in brackets can be written as:

$$\begin{aligned} &-f'(x_0^+)S_0 + (f'(x_1^-) - f(x_1^+))S_1 + (f'(x_2^-) - f'(x_2^+))S_2 \\ &+ \ldots + (f'(x_{s-1}^-) - f'(x_{s-1}^+))S_{s-1} + f'(x_s^-)S_s \end{aligned} \tag{5-17}$$

Noting that:

$$S_0 = \sin(0) = 0 \tag{5-18}$$

the expression is therefore equal to:

$$-(j_1'S_1 + j_2'S_2 + \ldots + j_{s-1}'S_{s-1} - f'(x_s^-)S_s) \tag{5-19}$$

where

$$j_k' = f'(x_k^+) - f'(x_k^-) \tag{5-20}$$

is the "jump" in the first derivative of f at x_k. These can be positive, null or negative values.

With $T = 4$, $x_0 = 0$ and $x_s = 1$ as discussed previously and shown in Equation (5-2), we obtain the following equation for the Fourier sine series coefficients of f:

$$b_n = \frac{4}{n\pi}\left[\left(f(0^+) + \sum_{k=1}^{s-1} j_k C_k\right) - \left(\frac{2}{n\pi}\right)^1 \left(\sum_{k=1}^{s-1} j_k' S_k - f'(1^-)S_s\right)\right] \tag{5-21}$$

for odd n, and 0 otherwise, where:

$$j_k = f(x_k^+) - f(x_k^-) \tag{5-22}$$

$$j_k' = f'(x_k^+) - f'(x_k^-) \tag{5-23}$$

$$C_k = \cos(\frac{n\pi x_k}{2}) = \cos(\frac{n\pi}{2s}k) \tag{5-24}$$

$$S_k = \sin(\frac{n\pi x_k}{2}) = \sin(\frac{n\pi}{2s}k) \tag{5-25}$$

It is also noted that, for equal length segments, $x_k = k/s$. Figure 5-3 shows an example of the first quadrant of a function f defined by four linear segments.

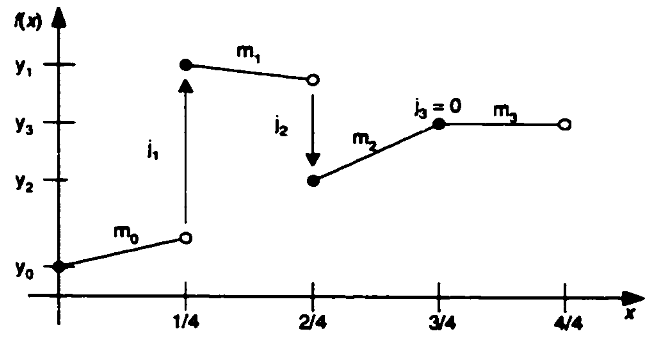

Figure 5-3 - First quadrant of f defined by four linear segments

From Figure 5-3, it can readily be seen that

$$f(0^{+}) = y_0 \tag{5-26}$$

$$j_k = y_k - y_{k-1} - \frac{m_{k-1}}{s} \qquad 1 \le k \le s-1 \tag{5-27}$$

$$f'(1^{-}) = m_{s-1} \tag{5-28}$$

$$j'_k = m_k - m_{k-1} \qquad 1 \le k \le s-1 \tag{5-29}$$

The Fourier sine series coefficients of f are therefore given by:

$$b_n = \frac{4}{n\pi}\left(y_0 + \sum_{k=1}^{s-1} (y_k - y_{k-1} - \frac{m_{k-1}}{s}) C_k \right) - \frac{8}{n^2\pi^2}\left(\sum_{k=1}^{s-1} (m_k - m_{k-1}) S_k - m_{s-1} S_s \right) \tag{5-30}$$

for odd n, and 0 otherwise.

5.3 Upper Bound on SFDR given the Number of Linear Segments Used

The SFDR is defined as the ratio of the power in the frequency of interest to the power in the strongest harmonic or spur. Without loss of generality, in this discussion it is assumed that the frequency of interest is for $n = 1$. For convenience's sake, the symbol P represents the SFDR, expressed as the ratio of the fundamental's amplitude to that of the greatest harmonic's. P is given by

$$P = \frac{|b(1)|}{\max(|b(n)|)} \quad n \neq 1 \tag{5-31}$$

The SFDR can be expressed in decibels as

$$10\log\left(\frac{b(1)^2}{\max(b(n)^2)}\right) = 20\log(P) \quad n \neq 1 \tag{5-32}$$

The amplitude of all harmonics is given by a version of Equation (5-30) stated in the more compact form

$$b(n) = \frac{4}{n\pi}g(n) - \frac{8}{n^2\pi^2}h(n) \tag{5-33}$$

for odd n, and 0 otherwise, where:

$$g(n) = \sum_{k=0}^{s-1}\alpha_k C_k(n) \quad n = 1, 3, 5, \ldots \tag{5-34}$$

$$h(n) = \sum_{k=1}^{s}\beta_k S_k(n) \quad n = 1, 3, 5, \ldots \tag{5-35}$$

$$C_k(n) = \cos\left(\frac{k\pi}{2s}n\right) \tag{5-36}$$

$$S_k(n) = \sin\left(\frac{k\pi}{2s}n\right) \tag{5-37}$$

The α_k and β_k are directly related to the jumps in amplitude and in the first derivative, respectively, of f. Using (5-34) and comparing Equations (5-30) and (5-33) yields

$$y_0 + \sum_{k=1}^{s-1}\left(y_k - y_{k-1} - \frac{m_{k-1}}{s}\right)C_k(n) = \sum_{k=0}^{s-1}\alpha_k C_k(n) \tag{5-38}$$

and hence the α_k can readily be calculated as:

$$\alpha_k = \begin{cases} y_k & k = 0 \\ y_k - y_{k-1} - \dfrac{m_{k-1}}{s} & k = 1,2,3,\ldots,s-1 \end{cases} \tag{5-39}$$

Similarly, using (5-35) and comparing Equations (5-30) and (5-33), we have

$$\sum_{k=1}^{s-1} (m_k - m_{k-1}) S_k(n) - m_{s-1} S_s = \sum_{k=1}^{s} \beta_k S_k(n) \tag{5-40}$$

and hence the β_k are given by:

$$\beta_k = \begin{cases} m_k - m_{k-1} & k = 1,2,3,\ldots,s-1 \\ -m_{s-1} & k = s \end{cases} \tag{5-41}$$

From (5-34) - (5-37), $g(n)$ and $h(n)$ are finite sums of sinusoids and therefore periodic in n. The longest period of any of the individual sinusoids that make up the sum is for $k = 1$, and the corresponding period can easily be found:

$$\frac{2\pi}{T} = \frac{\pi}{2s} \Rightarrow T = 4s \tag{5-42}$$

Thus, $g(n)$ and $h(n)$ are periodic in n with period $4s$, which is stated formally as:

$$g(n) = g(n + 4s) \tag{5-43}$$

$$h(n) = h(n + 4s) \tag{5-44}$$

The sine function has odd symmetry, and a sum of sines also has odd symmetry. The cosine function has even symmetry, and a sum of cosines also has even symmetry. Hence, $g(n)$ has even symmetry and $h(n)$ has odd symmetry. We therefore have:

$$g(n) = g(-n) \tag{5-45}$$

$$h(n) = -h(-n) \tag{5-46}$$

Combining the periodicity and symmetry of $g(n)$ and $h(n)$ yields:

$$g(n) = g(4s - n) \tag{5-47}$$

$$h(n) = -h(4s - n) \tag{5-48}$$

As stated above, it is desired that the fundamental frequency corresponds to $n = 1$, and that it be stronger than all harmonics. From (5-33)-(5-37), one way to achieve this goal is to have all α_k positive and all β_k negative. For $n = 1$, all angles in the summations of (5-34) and (5-35) will be in the first quadrant, and all summed terms will be positive. For odd values of n greater than 1 but less than $2s$, the angles for the summed sines and cosines will be uniformly distributed in the interval $[0, n\pi/2]$, with $n\pi/2s$ radians between each angle. Thus, there will be a mixture of positive and negative terms, leading to smaller sums. Symmetry and periodicity properties can be applied for odd values of n greater than $2s$ to realize that the same sums will be produced, with sign inversions in the case of the sum of sines. If all α_k are positive, then $g(1)$ will be greater than all other $g(n)$, since for other values of n at least one angle will be out of the first quadrant. Similarly, if all β_k are negative, then $|h(1)|$ will be greater than all other $|h(n)|$ for the same reason. Consequently, under these conditions $b(1)$ will be greater than all other $b(n)$, as desired.

Assume that the α_k and β_k are selected such that $b(1)$ has the greatest amplitude, and that $g(1) = 0$. Considering the periodicity of $h(n)$ shown by (5-44) and the symmetry of $h(n)$ shown by (5-48), then it is obvious that $-h(4s - 1) = h(4s + 1) = h(1)$. An upper bound for the SFDR can then be found if we assume $b(n) = 0$, $n = 2, 3, 4, \ldots, 4s - 2$. In that case, the SFDR would depend only on the ratio of the amplitudes of $b(1)$ to that of $b(4s - 1)$ and $b(4s + 1)$. Because of the $1/n^2$ factor in (5-33), the harmonic corresponding to $n = 4s - 1$ will be the greatest. Under these conditions, an upper bound for the SFDR is given by:

$$P \le \frac{\left|\frac{8}{\pi^2} h(1)\right|}{\left|\frac{8}{(4s-1)^2\pi^2} h(4s-1)\right|} = \frac{\left|\frac{8}{\pi^2} h(1)\right|}{\left|\frac{8}{(4s-1)^2\pi^2} h(1)\right|} = (4s-1)^2 = 16s^2 - 8s + 1 \tag{5-49}$$

The bound for SFDR can be improved from that given in (5-49) by exploiting the even and odd symmetries of $g(n)$ and $h(n)$, respectively. Consider (5-33) and assume that $h(1)$ is fixed and negative and $g(1)$ is variable and positive. We have

$$\begin{aligned} |b(1)| &= \left|\frac{4}{\pi}g(1) - \frac{8}{\pi^2}h(1)\right| \\ |b(4s-1)| &= \left|\frac{4}{(4s-1)\pi}g(1) + \frac{8}{(4s-1)^2\pi^2}h(1)\right| \\ |b(4s+1)| &= \left|\frac{4}{(4s+1)\pi}g(1) - \frac{8}{(4s+1)^2\pi^2}h(1)\right| \end{aligned} \qquad \textit{(5-50)}$$

Any increase in $g(1)$ also increases $|b(1)|$, which is desirable. Further, any increase in $g(1)$ decreases $|b(4s - 1)|$, which is also desirable since it decreases the ratio of the amplitude of that harmonic to the fundamental's. However, increasing $g(1)$ also increases $|b(4s + 1)|$, which is not desirable if that harmonic's amplitude gets larger than $|b(4s - 1)|$. Consequently, increasing $g(1)$ has a positive effect on the SFDR, but only to a point. Eventually, the amplitude of $|b(4s + 1)|$ reaches that of $|b(4s - 1)|$ and the SFDR starts to decrease again. The best case occurs when the amplitudes of these two harmonics are equal, corresponding to an optimum value for $g(1)$. This leads to a relation between $g(1)_{opt}$ and $h(1)$:

$$\begin{aligned} |b(4s-1)| &= |b(4s+1)| \\ -\frac{4}{(4s-1)\pi}g(1)_{opt} - \frac{8}{(4s-1)^2\pi^2}h(1) &= \frac{4}{(4s+1)\pi}g(1)_{opt} - \frac{8}{(4s+1)^2\pi^2}h(1) \\ g(1)_{opt}\frac{4}{\pi}\left(\frac{8s}{(4s-1)(4s+1)}\right) &= h(1)\frac{8}{\pi^2}\left(\frac{-16s}{(4s-1)^2(4s+1)^2}\right) \\ g(1)_{opt} &= -\frac{4h(1)}{\pi}\left(\frac{1}{16s^2-1}\right) \end{aligned} \qquad \textit{(5-51)}$$

Optimizing $g(1)$ will also optimize $b(1)$. Using (5-51) in (5-33) for $n = 1$ yields:

$$\begin{aligned} b(1)_{opt} &= \frac{4g(1)_{opt}}{\pi} - \frac{8h(1)}{\pi^2} \\ &= -\frac{16h(1)}{\pi^2(16s^2-1)} - \frac{8h(1)}{\pi^2} \\ &= -\frac{8h(1)}{\pi^2}\left(\frac{2}{16s^2-1}+1\right) \\ &= -\frac{8h(1)}{\pi^2}\left(\frac{16s^2+1}{16s^2-1}\right) \end{aligned} \tag{5-52}$$

As discussed above, setting $g(1) = g(1)_{opt}$ will make $|b(4s+1)|$ and $|b(4s-1)|$ equal, denoted here by the subscript *eq*:

$$\begin{aligned} b(4s+1)_{eq} &= \frac{4g(1)_{opt}}{(4s+1)\pi} - \frac{8h(1)}{(4s+1)^2\pi^2} \\ &= -\frac{16h(1)}{\pi^2(4s+1)(4s+1)(4s-1)} - \frac{8h(1)}{(4s+1)^2\pi^2} \\ &= -\frac{8h(1)}{\pi^2(4s+1)^2}\times\left(\frac{2}{(4s-1)}+1\right) \\ &= -\frac{8h(1)}{\pi^2(16s^2-1)} \end{aligned} \tag{5-53}$$

This yields the following upper bound for SFDR as a function of the number of segments s:

$$P_{max} \le \frac{|b(1)_{opt}|}{|b(4s+1)_{eq}|} = 16s^2+1 \tag{5-54}$$

or, in decibels,

$$20\log(P_{max}) \cong 20\log(16s^2) \cong 24 + 40\log s \quad s \ge 1 \tag{5-55}$$

A graph of this bound is shown in Figure 5-4. Equation (5-55) can also be used to determine the minimum number of linear segments necessary to achieve a desired SFDR.

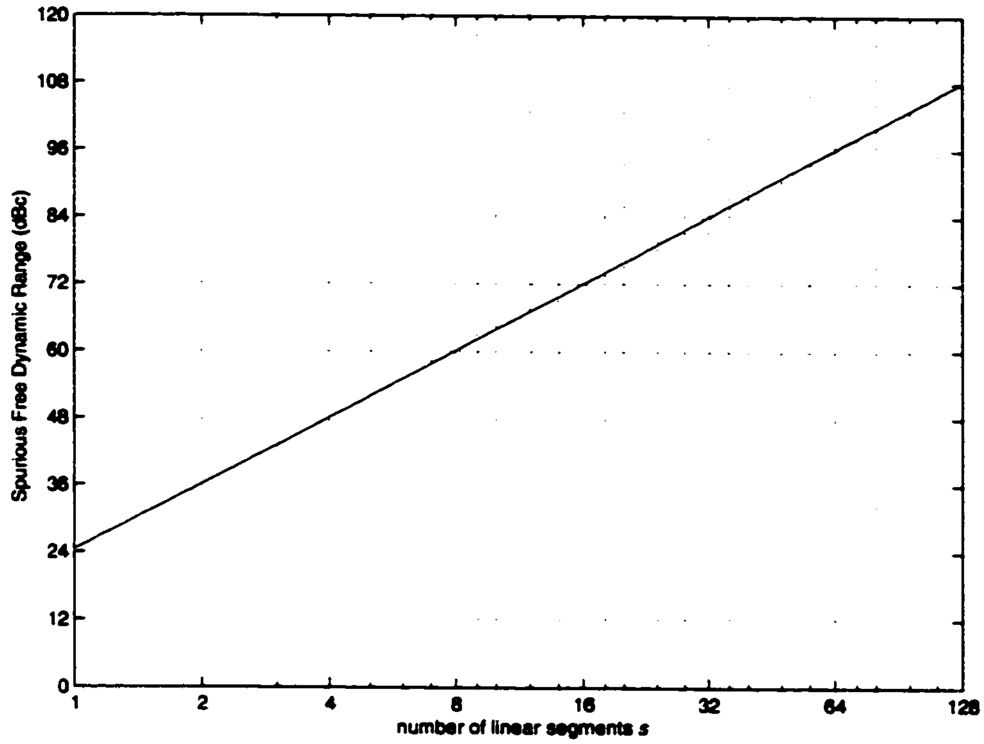

Figure 5-4 - Upper bound on SFDR as a function of the number of linear segments used

5.4 Linear Interpolation Coefficients Selection

5.4.1 Selection Criteria for h(n), n = 3, 5, 7, ..., 4s − 3

Equation (5-54) only considers harmonics for values of n equal to 1, $4s - 1$, $4s + 1$, $8s - 1$, $8s + 1$, etc. This section considers the criteria that must be met by function h in order to achieve a given SFDR P, for $n = 3, 5, 7, \ldots, 4s - 3$. From (5-31) and (5-33), we have

$$\left|\frac{4}{n\pi}g(n) - \frac{8}{n^2\pi^2}h(n)\right| \le \frac{b(1)}{P} \quad n = 3, 5, 7, \ldots, 4s - 3 \tag{5-56}$$

from which the following two linear inequalities are obtained:

$$g(n) \geq \frac{n\pi}{4}\left(-\frac{b(1)}{P} + \frac{8}{n^2\pi^2}h(n)\right) \tag{5-57}$$

and

$$g(n) \leq \frac{n\pi}{4}\left(\frac{b(1)}{P} + \frac{8}{n^2\pi^2}h(n)\right) \tag{5-58}$$

for n = 3, 5, 7, ..., $4s - 3$. Consequently, $h(n)$ must be chosen such that there exists at least one solution for $g(n)$ that satisfies inequalities (5-57) and (5-58), for n = 3, 5, 7, ..., $4s - 3$. We pose

$$g(n)_{\min} = \frac{n\pi}{4}\left(-\frac{b(1)}{P} + \frac{8}{n^2\pi^2}h(n)\right) \tag{5-59}$$

and

$$g(n)_{\max} = \frac{n\pi}{4}\left(\frac{b(1)}{P} + \frac{8}{n^2\pi^2}h(n)\right) \tag{5-60}$$

Using the odd symmetry property of $h(n)$ from (5-48) yields

$$\begin{aligned} g(4s-n)_{\min} &= \frac{(4s-n)\pi}{4}\left(-\frac{b(1)}{P} + \frac{8h(4s-n)}{(4s-n)^2\pi^2}\right) \\ &= \frac{(4s-n)\pi}{4}\left(-\frac{b(1)}{P} - \frac{8h(n)}{(4s-n)^2\pi^2}\right) \end{aligned} \tag{5-61}$$

and

$$\begin{aligned} g(4s-n)_{\max} &= \frac{(4s-n)\pi}{4}\left(\frac{b(1)}{P} + \frac{8h(4s-n)}{(4s-n)^2\pi^2}\right) \\ &= \frac{(4s-n)\pi}{4}\left(\frac{b(1)}{P} - \frac{8h(n)}{(4s-n)^2\pi^2}\right) \end{aligned} \tag{5-62}$$

Equation (5-47) combines the symmetry and periodicity of $g(n)$, and $g(n)$ must lie inside the two intervals defined by Equations pairs (5-59)(5-60) and (5-61)(5-62), for n = 3, 5, 7, ..., $4s - 3$. Figure 5-5 illustrates the two limit cases where this becomes no longer possible. The following can be stated formally:

$$g(n)_{\min} \le g(4s-n)_{\max} \tag{5-63}$$

and

$$g(n)_{\max} \ge g(4s-n)_{\min} \tag{5-64}$$

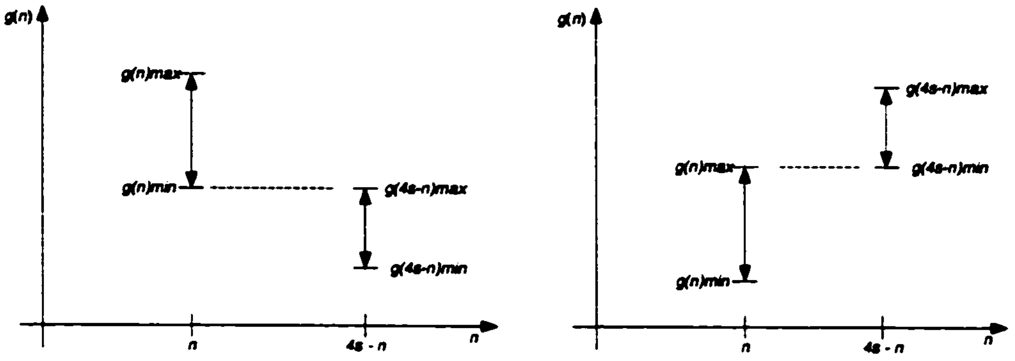

***Figure 5-5 - Two limit conditions for acceptable intervals for* g(n), *with* g(n) = g(4s – n)**

Expanding (5-63) yields

$$\begin{gathered}
g(n)_{\min} \le g(4s-n)_{\max} \\
\frac{n\pi}{4}\left(-\frac{b(1)}{P}+\frac{8h(n)}{n^2\pi^2}\right) \le \frac{(4s-n)\pi}{4}\left(\frac{b(1)}{P}-\frac{8h(n)}{(4s-n)^2\pi^2}\right) \\
-\frac{nb(1)}{P}+\frac{8h(n)}{n\pi^2} \le \frac{4sb(1)}{P}-\frac{nb(1)}{P}-\frac{8h(n)}{(4s-n)\pi^2} \\
-\frac{4sb(1)}{P} \le -\frac{8h(n)}{\pi^2}\left(\frac{1}{n}+\frac{1}{(4s-n)}\right) \\
\frac{4sb(1)}{P} \ge \frac{8h(n)}{\pi^2}\left(\frac{4s-n+n}{n(4s-n)}\right) \\
h(n) \le \frac{\pi^2 b(1) n(4s-n)}{8P}
\end{gathered} \tag{5-65}$$

and expanding (5-64) yields

$$
\begin{aligned}
g(n)_{\max} &\geq g(4s-n)_{\min} \\
\frac{n\pi}{4}\left(\frac{b(1)}{P}+\frac{8h(n)}{n^2\pi^2}\right) &\geq \frac{(4s-n)\pi}{4}\left(-\frac{b(1)}{P}-\frac{8h(n)}{(4s-n)^2\pi^2}\right) \\
\frac{nb(1)}{P}+\frac{8h(n)}{n\pi^2} &\geq -\frac{4sb(1)}{P}+\frac{nb(1)}{P}-\frac{8h(n)}{(4s-n)\pi^2} \\
\frac{4sb(1)}{P} &\geq -\frac{8h(n)}{\pi^2}\left(\frac{1}{n}+\frac{1}{(4s-n)}\right) \\
\frac{4sb(1)}{P} &\geq -\frac{8h(n)}{\pi^2}\left(\frac{4s-n+n}{n(4s-n)}\right) \\
-h(n) &\leq \frac{\pi^2 b(1)n(4s-n)}{8P}
\end{aligned}
\qquad (5\text{-}66)
$$

Combining (5-65) and (5-66) yields

$$
|h(n)| \leq \frac{\pi^2 b(1)n(4s-n)}{8P} \qquad (5\text{-}67)
$$

for $n = 3, 5, 7, \ldots, 2s - 1$, since $h(n)$ has symmetry about $n = 2s$, from (5-48).

During the design process, it can be advantageous to select the slope coefficients before the segment initial amplitude coefficients. This is done to optimize the system's hardware realization, which is greatly dependent on the cost of implementing the multiplication in Figure 5-1. Hence, the slopes m_k and the β_k, given by (5-41), should be set early on. The $h(n)$ are then determined from (5-35), and Equation (5-67) can be used to determine the achievable SFDR corresponding to these slopes:

$$
P \leq \frac{\pi^2 b(1)n(4s-n)}{8\times\max(|h(n)|)} \qquad (5\text{-}68)
$$

However, the value of b(1) may not be known at this time since the segment amplitudes y_k, and hence the α_k and $g(n)$, have not been selected yet. Inspection of (5-52) reveals that, for the case were the achieved SFDR is close to the bound predicted by (5-54), then $b(1) \rightarrow -8 \times h(1) / \pi^2$, especially for large s. Hence, with the goal of achieving an SFDR close to the bound predicted by (5-54), the following equation can be used to assess the suitability of a set of slopes m_k:

$$P \leq \frac{-h(1)n(4s-n)}{\max(|h(n)|)} \tag{5-69}$$

for $n = 3, 5, 7, \ldots, 2s - 1$.

5.4.2 Acceptable Range of Values for g(1) Given a Desired SFDR

Equation (5-54) provides an upper bound for the SFDR given a number of segments. This bound can only be achieved with a single value of $g(1)$, given by (5-51), and a corresponding value of $b(1)$. This severely restricts the choice of values for the α_k coefficients and the corresponding segment initial amplitudes y_k. It is therefore useful to be able to specify a range of acceptable values for $g(1)$ and $b(1)$ that satisfy a given P.

Consider cases where $|b(4s + 1)|$ and $|b(4s - 1)|$ are different, while maintaining

$$P \leq \frac{|b(1)|}{|b(4s+1)|} \text{ and } P \leq \frac{|b(1)|}{|b(4s-1)|} \tag{5-70}$$

where the desired SFDR P must obviously be less than that specified by (5-54). There are two cases to consider. First assume that $g(1) < g(1)_{opt}$. In this case, we will have $|b(4s - 1)| > |b(4s + 1)|$. Denoting the minimal value of $g(1)$ that will satisfy (5-70) by $g(1)_{min}$, we have:

$$\begin{aligned}
\left|\frac{4g(1)_{min}}{\pi} - \frac{8h(1)}{\pi^2}\right| &= \left|\frac{4g(1)_{min}}{(4s-1)\pi} + \frac{8h(1)}{(4s-1)^2\pi^2}\right| \times P \\
\frac{4g(1)_{min}}{\pi} - \frac{8h(1)}{\pi^2} &= \left(-\frac{4g(1)_{min}}{(4s-1)\pi} - \frac{8h(1)}{(4s-1)^2\pi^2}\right) \times P \\
\frac{4g(1)_{min}}{\pi}\left(1 + \frac{P}{(4s-1)}\right) &= \frac{8h(1)}{\pi^2}\left(1 - \frac{P}{(4s-1)^2}\right) \\
g(1)_{min} &= \frac{2h(1)}{\pi}\frac{\left(1 - P/(4s-1)^2\right)}{\left(1 + P/(4s-1)\right)}
\end{aligned} \tag{5-71}$$

Now consider the case where $g(1) > g(1)_{opt}$. In this case, we will have $|b(4s + 1)| > |b(4s - 1)|$. Denoting the maximum value of $g(1)$ that will satisfy (5-70) by $g(1)_{max}$, we have:

$$\begin{aligned}
\left|\frac{4g(1)_{\max}}{\pi}-\frac{8h(1)}{\pi^2}\right| &= \left|\frac{4g(1)_{\max}}{(4s+1)\pi}-\frac{8h(1)}{(4s+1)^2\pi^2}\right| \times P \\
\frac{4g(1)_{\max}}{\pi}-\frac{8h(1)}{\pi^2} &= \left(\frac{4g(1)_{\max}}{(4s+1)\pi}-\frac{8h(1)}{(4s+1)^2\pi^2}\right) \times P \\
\frac{4g(1)_{\max}}{\pi}\left(1-\frac{P}{(4s+1)}\right) &= \frac{8h(1)}{\pi^2}\left(1-\frac{P}{(4s+1)^2}\right) \\
g(1)_{\max} &= \frac{2h(1)}{\pi}\frac{\left(1-P/(4s+1)^2\right)}{\left(1-P/(4s+1)\right)}
\end{aligned} \qquad (5\text{-}72)$$

The corresponding values of $b(1)_{\min}$ and $b(1)_{\max}$ are readily obtained by replacing (5-71) and (5-72) in (5-33), giving:

$$b(1)_{\min} = -\frac{32h(1)sP}{\pi^2(4s-1)(P+4s-1)} \qquad (5\text{-}73)$$

and:

$$b(1)_{\max} = -\frac{32h(1)sP}{\pi^2(4s+1)(P-4s-1)} \qquad (5\text{-}74)$$

5.4.3 Selection Criteria for g(n), n = 1, 3, 5, 7, ..., 2s − 1

Equations (5-71) and (5-72) specified the bounds within which $g(1)$ must lie. Consider now the bounds set by Equations (5-57) and (5-58) for n = 3, 5, 7, ..., $4s$ − 3, together with Figure 5-5. From Equation (5-47), $g(n)$ has even symmetry about $n = 2s$. Therefore, the bounds for $g(n)$ need only be specified for one half of a period of n, and they are given by:

$$g(n)^{*}_{\min} = \begin{cases} g(1)_{\min} & n=1 \\ \max(g(n)_{\min}, g(4s-n)_{\min}) & n=3,5,7,\dots,2s-1 \end{cases} \qquad (5\text{-}75)$$

and

$$g(n)^{*}_{\max} = \begin{cases} g(1)_{\max} & n=1 \\ \min(g(n)_{\max}, g(4s-n)_{\max}) & n=3,5,7,\dots,2s-1 \end{cases} \qquad (5\text{-}76)$$

5.5 Design Procedure

From the derivation of the previous section, a systematic design approach for a DDF Synthesizer based on the architecture of Figure 5-1 can be obtained. The design procedure is as follows:

Part I: specification of SFDR and selection of the number of segments

1. Specify a target SFDR P.

2. From (5-54), determine the minimum number of segments to achieve this SFDR; in practice, this number will be an integer power of 2;

Part II: selection of a set of slopes

3. Select a set of quantized slopes m_k, and compute the corresponding set of coefficients β_k (Equation (5-41)). Compute $h(1)$ (Equation (5-35)).

4. Confirm that the set of quantized slopes allows the target SFDR to be achieved, according to Equation (5-69). Calculate and verify that $g(1)_{min} < g(1)_{max}$ (Equations (5-71) and (5-72)), and that $b(1)_{min} < b(1)_{max}$ (Equations (5-73) and (5-74)). If any of these conditions is not verified, return to step 3 and select a different set of slopes m_k.

Notes on Part II:

a. The slope coefficients m_k must be quantized with finite precision. This precision has a direct impact on the system complexity required for storing these coefficients and on the complexity of the multiplier in Figure 5-1. Real valued slopes can be calculated for each segment directly from the linear interpolation of the sine function over this segment:

$$m_k = \frac{\sin(\frac{\pi x_{k+1}}{2}) - \sin(\frac{\pi x_k}{2})}{x_{k+1} - x_k} = s \times \left(\sin(\frac{\pi x_{k+1}}{2}) - \sin(\frac{\pi x_k}{2}) \right) \qquad (5\text{-}77)$$

assuming that the s segments have equal length. These real values must then be quantized to the desired precision. It is these resulting quantized values that must satisfy the conditions set in steps 3 and 4 above. The remainder of the design procedure is based on using the quantized values of the slopes, not the values calculated by Equation (5-77).

b. The first derivative of sin($\pi x/2$) at $x = 0$ is $\pi/2 \approx 1.57$, and it is 0 at $x = \pi/2$. Hence, a reasonable design heuristic consists of limiting the range of acceptable values for m_k to the interval [0, 2].

c. For a small number of segments, i.e. 4 or 8, an exhaustive search can be performed to find an acceptable set of slopes m_k, according to the criteria listed in step 4., among choices expressed with a fixed number of bits. For large s, it was found that a Genetic Algorithm [25] is well suited to this task. Details are provided in Sections 6.3.2 and 6.3.3.

Part III: calculation of y_k coefficients

5. Obtain $g(n)^*_{\min}$ and $g(n)^*_{\max}$ from (5-75) and (5-76).

6. Pose 2 sets of s inequalities that can be expressed in matrix form as:

$$[C_k(n)][\alpha_k] \le \lfloor g(n)^*_{\max} \rfloor \tag{5-78}$$

$$[C_k(n)][\alpha_k] \ge \lfloor g(n)^*_{\min} \rfloor \tag{5-79}$$

and find a $s \times 1$ vector $[\alpha_k]$ that satisfies all of them. Equations (5-78) and (5-79) define the bounds of an s-dimension hyper-parallelogram, and a suitable vector $[\alpha_k]$ represents a point that lies inside it. A simple approach to solve this problem consists of finding one vertex of the parallelogram by solving Equation (5-78) with the equal sign, and finding its opposing vertex by similarly solving Equation (5-79). The center of the parallelogram can readily be found by taking the geometric mean of the coordinates of these two vertices.

7. From the α_k obtained in 6., calculate floating point values for the y_k using Equation (5-39).

Part IV: compilation of system parameters and simulation

8. Select appropriate values of M, L, C and D, where M is the number of phase bits, L is the number of bits used to quantize the output amplitude, C is the number of bits used to quantize the y_k, and D is the number of added columns in the final adder (other bits are truncated). Simulate the system taking finite precision arithmetic effects into account, and calculate one period of sinusoid samples with a value of FCW equal to 1. Confirm that the output amplitude can be

represented by a fraction in the interval [0, 1[, i.e. by a quantized binary integer in the interval [0, $2^L - 1$], otherwise reject this set of parameters.

9. Take the Discrete Fourier Transform of the sequence, and measure the SFDR. As discussed in Section 2.3.1, spectral components will vary in position but not in amplitude for all odd values of FCW, hence evaluating the SFDR for FCW = 1 is sufficient as long as the FCW is always odd.

10. Repeat for different values of M, L, C and D, and obtain high-level design metrics such as the required number of full adders and 2-to-1 multiplexers, and the number of stored bits. Select the set that meets SFDR requirements at the lowest cost.

Chapter 6

New DDFS Architectures Based on Linear Interpolation

This chapter presents new DDFS architectures with a PSAC based on linear interpolation. The architectures efficiently realize the system shown in Figure 5-1 with low hardware complexity. High-level complexity metrics are provided to allow rapid cost evaluation early into the design cycle. It is then shown how the design procedure described in Chapter 5 can be applied to these architectures, and chosen system coefficients and parameters are given that meet various levels of output SFDR. HDL description issues are discussed for VHDL and Synopsys® Module Compiler Language™. Implementation results in FPGA and for 0.35 μm and 0.18 μm CMOS libraries are listed for several designs. Finally, details concerning two designs submitted for fabrication are given, together with hardware test results for one of them.

6.1 New Architectures

6.1.1 General Description

A major contributor to system complexity in the architecture of Figure 5-1 is the multiplier. In Section 3.4, existing DDFS architectures based on a linear interpolation of the sine function were reviewed. Freeman's architecture [22] includes a 6×5 multiplier, while Bellaouar et al.'s includes a 5×5 multiplier. Liu et al. [55], however, proposed a multiplier-less architecture, and represented the slope coefficients with no more than 4 signed digits. They used linear segments of unequal lengths.

The present research has resulted in novel DDFS architectures with a multiplier-less PSAC based on linear interpolation with equal length segments [47][50][51][52][53]. To eliminate the requirement for multipliers, the segment slopes m_k are selected such that all can be represented with a limited number of signed digits. Multiplexers provide corresponding shifts of the LSBs of the phase accumulator to a multi-operand adder. A separate multiplexer acts as a small ROM and provides constants equal to segment initial amplitudes y_k. A generalized version of the novel architecture is shown in Figure 6-1.

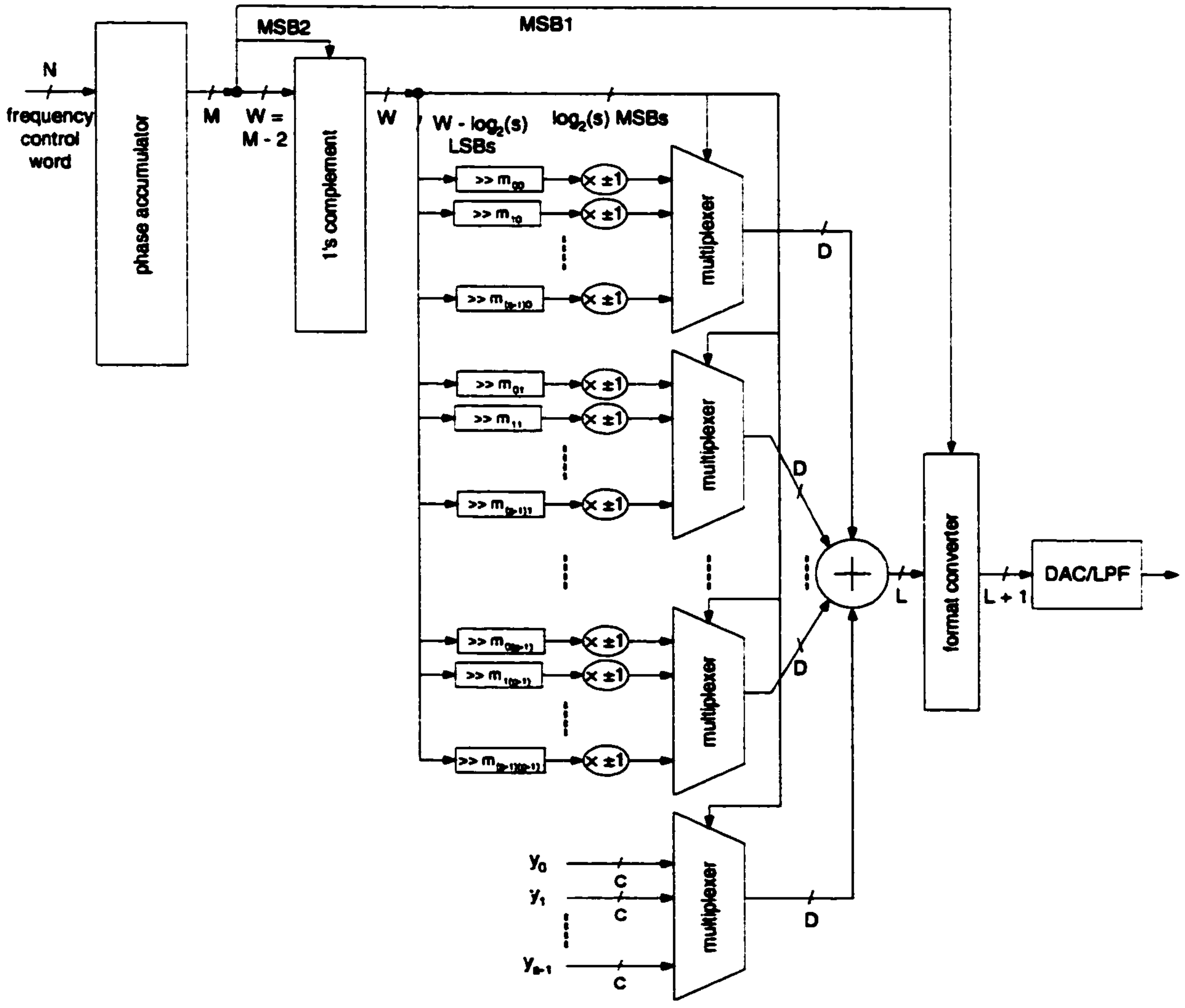

Figure 6-1 Novel single-phase multiplier-less DDFS architecture, general format

As before, N is the internal width of the phase accumulator and of the frequency control word. The first quadrant of the sine function is approximated with s equal length segments, following the definition given by Equation (5-2). The truncated phase word x is M bits wide. Quadrant symmetry is exploited, and $W = M - 2$ is the number of bits used to represent angles in the first quadrant. The segment slopes m_k are expressed as a sum of at most p powers of two selected in the interval $[-1, 1]$:

$$m_k = \sum_{j=0}^{p-1} m_{kj} \quad m_{kj} \in \left\{-2^0, -2^{-1}, -2^{-2}, \ldots, 0, \ldots, 2^{-2}, 2^{-1}, 2^0\right\} \qquad (6\text{-}1)$$

The multiplication of the offset angles $(x - x_k)$ by a single power of two is realized by a trivial shift. A total of p multiplexers with s inputs each select the correct partial products according to the phase angle segment. In Figure 6-1, the notation ">> m_{kj}" represents a right shift corresponding to the jth power of two used to represent the slope in the kth segment. The number of shifted positions is equal to $-\log_2(|m_{kj}|)$, where the absolute value is taken since the digits may be negative. If necessary, the sign inversion is realized by the ± 1 multipliers that follow each shifter block in Figure 6-1.

An additional multiplexer provides the segment initial amplitudes y_k, represented with C bits. This multiplexer and its coefficients are the equivalent of a ROM, but it can normally be realized with logic gates due to its small size. There are therefore a total of $p + 1$ multiplexers whose outputs are D bits wide. The multiplexer control signals are given by the $\log_2 s$ MSBs of the W-bit phase angle since the s segments are equal in length. The remaining $W - \log_2 s$ LSBs give the offset angle $(x - x_k)$.

A multi-operand adder sums the multiplexer outputs, and rounds the result to L bits. Sign information is added to the sine amplitudes in a format converter, and the $L + 1$ bits wide output sequence can be passed to a DAC and a LPF.

The complexity of this architecture is significantly less than one using a regular multiplier and a ROM to store the slope coefficients m_k. This is true only if the number of signed digits used to represent the slopes is small, e.g. 2 or 3.

6.1.2 Addend Alignment

In this section, the alignment of the multiplexer outputs prior to addition and the resulting implementation optimizations will be explained in detail.

The M-bit phase word x is expressed in fixed point notation with two bits for the integer part and $M-2$ bits for the fractional part. Let

$$x = X_{M-1}\times 2^{1} + X_{M-2}\times 2^{0} + X_{M-3}\times 2^{-1} + \ldots + X_{1}\times 2^{-(M-3)} + X_{0}\times 2^{-(M-2)} \qquad (6\text{-}2)$$

where X_i is the ith bit of x. The phase word can be decomposed into three parts: the quadrant, the segment, and the offset angle. The two MSBs of x, i.e. the integer portion, identify the quadrant. The following $\log_2 s$ bits identify the segment. The remaining LSBs, from $M-3-\log_2 s$ down to 0, give the offset angle equal to $(x-x_k)$ in Equation (5-2). This decomposition is shown in Figure 6-2.

quadrant	binary point	segment	$(x-x_k)$
$X_{M-1}\ X_{M-2}$	.	$X_{M-3}\ \ldots\ X_{M-3-\log 2(s)+1}$	$X_{M-3-\log 2(s)}\ \ldots\ X_2\ X_1\ X_0$

Figure 6-2 - Decomposition of the phase angle word

As discussed in the previous section, the multi-operand adder receives $p+1$ addends. Although the addends are D bits wide, several of these bits are zeros or, in the presence of a negative slope digit, ones. These include the $\log_2 s$ digits corresponding to the segment, and the following $-\log_2|m_{kj}|$ bits due to the shift operation. If a sufficient number of bits are shifted, then it is also possible that the resulting addend will be more than D bits wide. In such a case, the extra LSBs are discarded. The number of discarded bits is equal to $\max\{-\log_2|m_{kj}| + M - 2 - D, 0\}$. The effect of this truncation must be taken into account when selecting system parameters, as described in part IV of the design procedure of Section 5.5. Zeros can also be introduced in the case of the C-bit initial amplitudes, when $D > C$. The corresponding addend's $(D-C)$ LSBs will always be zeros.

Coefficients and parameters are selected such that all sums are restricted to the interval [0, 1[. Hence, no overflow occurs to the left of the binary point. After the operands are added, the sum is rounded to L bits. The rounding type is a simplified form of *rounding to nearest*, i.e. the $(L + 1)^{th}$ MSB is added as a carry to the L^{th} most significant column.

The correct alignment of the $p + 1$ addends is shown in Figure 6-3 below, for the case where all m_{kj} are positive.

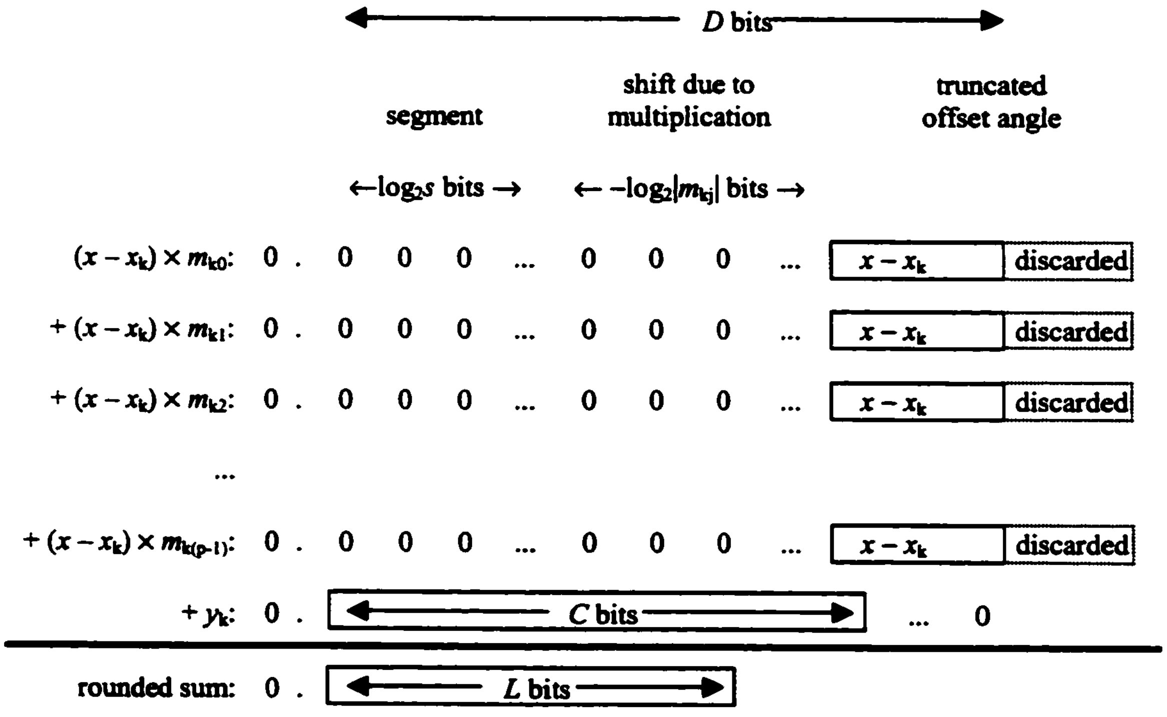
D bits
segment
shift due to multiplication
truncated offset angle
←log₂s bits →
← −log₂|m_kj| bits →
(x − x_k) × m_k0: 0 . 0 0 0 ... 0 0 0 ...
+ (x − x_k) × m_k1: 0 . 0 0 0 ... 0 0 0 ...
+ (x − x_k) × m_k2: 0 . 0 0 0 ... 0 0 0 ...
...
+ (x − x_k) × m_k(p-1): 0 . 0 0 0 ... 0 0 0 ...
x − x_k
discarded
+ y_k: 0 .
C bits
... 0
rounded sum: 0 .
L bits

Figure 6-3 - Addend alignment, general case

The process can be further illustrated by means of two examples. First, consider a DDFS with s = 8 segments and slopes m_k represented with at most two signed digits. Assume a typical case where the phase angle x is M = 13 bits wide, the output is expressed with L = 7 bits of amplitude, the segment initial amplitudes y_k are expressed with C = 10 bits, and the addition is done on D = 11 bits wide operands. The two MSBs of x, bits 12 and 11, identify the quadrant. Bits 10, 9 and 8 identify the segment, and bits 7 down to 0 give the offset angle.

For this case, there are three 11-bit addends. The first two addends are shifted versions of the offset angle $(x - x_k)$, with some bits possibly truncated, and the third addend is the initial amplitude y_k, expressed with 10 bits followed by a zero as the LSB. Consider a specific segment with slope m_k equal to $(1 + 1 / 8)$. The first three bits of the first addend and the first six bits of the second addend are zeros. The addition is then performed with the addends aligned as shown in Figure 6-4:

$$
\begin{array}{rrcccccccccccc}
 & (x-x_k)/1: & 0\,. & 0 & 0 & 0 & X_7 & X_6 & X_5 & X_4 & X_3 & X_2 & X_1 & X_0 \\
+ & (x-x_k)/8: & 0\,. & 0 & 0 & 0 & 0 & 0 & 0 & X_7 & X_6 & X_5 & X_4 & X_3 \\
+ & y_k: & 0\,. & Y_9 & Y_8 & Y_7 & Y_6 & Y_5 & Y_4 & Y_3 & Y_2 & Y_1 & Y_0 & 0 \\
\hline
 & \text{rounded sum:} & 0\,. & S_6 & S_5 & S_4 & S_3 & S_2 & S_1 & S_0 & & & &
\end{array}
$$

Figure 6-4 - Addend alignment, example 1

As a second example, consider the conditions of the previous example, but with a slope m_k including a negative digit, e.g. $m_k = (1 - 1 / 8)$. Consequently, the second addend's bits must be inverted prior to the addition. This is shown in Figure 6-5, where $\underline{X}_i$ denotes the complement of X_i. A '1' is not added in the LSB position of the second addend since it would be discarded. The impact of this approximation must be considered when selecting system parameters and coefficients. A '0' is shown to the left of the binary point for the second addend since that bit is not part of the addition and coefficients are selected such that the sum is less than one. Consequently, any overflow from the addition is discarded.

$$
\begin{array}{rrcccccccccccc}
 & (x-x_k)/1: & 0\,. & 0 & 0 & 0 & X_7 & X_6 & X_5 & X_4 & X_3 & X_2 & X_1 & X_0 \\
+ & (x-x_k)/8: & 0\,. & 1 & 1 & 1 & 1 & 1 & 1 & \underline{X}_7 & \underline{X}_6 & \underline{X}_5 & \underline{X}_4 & \underline{X}_3 \\
+ & y_k: & 0\,. & Y_9 & Y_8 & Y_7 & Y_6 & Y_5 & Y_4 & Y_3 & Y_2 & Y_1 & Y_0 & 0 \\
\hline
 & \text{rounded sum:} & 0\,. & S_6 & S_5 & S_4 & S_3 & S_2 & S_1 & S_0 & & & &
\end{array}
$$

Figure 6-5 - Addend alignment, example 2

As a final optimization, the ones in the MSB positions of the second addend can be pre-added to the segment initial amplitudes y_k, resulting in the final addition shown in Figure 6-6, where $Y_i^{\bullet}$ represents the original MSBs of y_k to which a series of '1' digits have been added.

$$
\begin{array}{rrcccccccccccccc}
 & (x-x_k)/1: & 0 & . & 0 & 0 & 0 & X_7 & X_6 & X_5 & X_4 & X_3 & X_2 & X_1 & X_0 \\
+ & (x-x_k)/8: & 0 & . & 0 & 0 & 0 & 0 & 0 & 0 & \underline{X}_7 & \underline{X}_6 & \underline{X}_5 & \underline{X}_4 & \underline{X}_3 \\
+ & y_k: & 0 & . & Y_9^{\bullet} & Y_8^{\bullet} & Y_7^{\bullet} & Y_6^{\bullet} & Y_5^{\bullet} & Y_4^{\bullet} & Y_3 & Y_2 & Y_1 & Y_0 & 0 \\
\hline
 & \text{rounded sum:} & 0 & . & S_6 & S_5 & S_4 & S_3 & S_2 & S_1 & S_0 & & & &
\end{array}
$$

Figure 6-6 - Addend alignment, pre-addition optimization

6.1.3 Architecture Examples, Single Phase Output

Three example architectures are shown in Figure 6-7, Figure 6-8 and Figure 6-9, with different cases of the number of linear segments and number of signed digits used to represent the slopes m_k.

In Figure 6-7, there are four linear segments, and the slopes are represented with no more than two positive digits. The 2 MSBs of the truncated phase identify one of four segments and are the multiplexer control signals. Each multiplexer has 4 inputs. For the two upper multiplexers, the inputs are shifted versions of the $W-2$ least significant bits of the truncated phase. The lowest multiplexer inputs are constants equal to the segment initial amplitudes y_k.

Figure 6-8 shows a version of the architecture with 8 linear segments and slopes m_k expressed as a sum of at most two positive powers of two. The 3 MSBs of the truncated phase identify one of eight segments and are the multiplexer control signals. Each multiplexer has 8 inputs, which are shifted versions of the $W-3$ least significant bits of the truncated phase or initial amplitudes in the case of the lowest multiplexer.

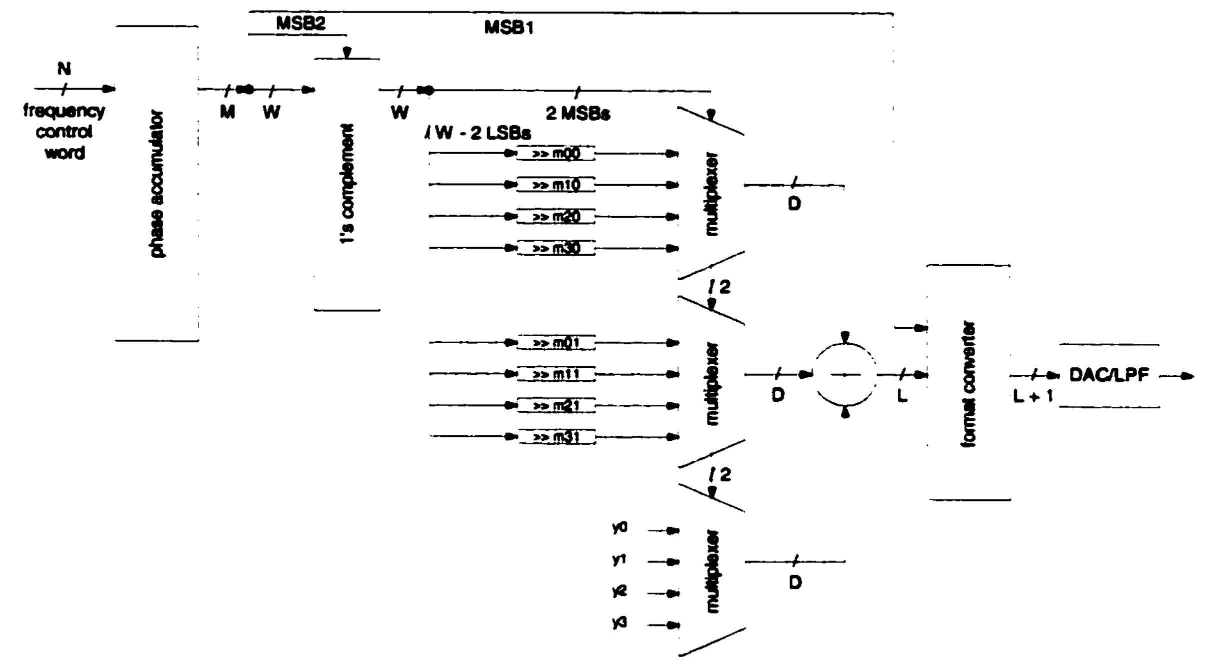

Figure 6-7 - Novel multiplier-less DDFS architecture, four segments, slopes with two digits

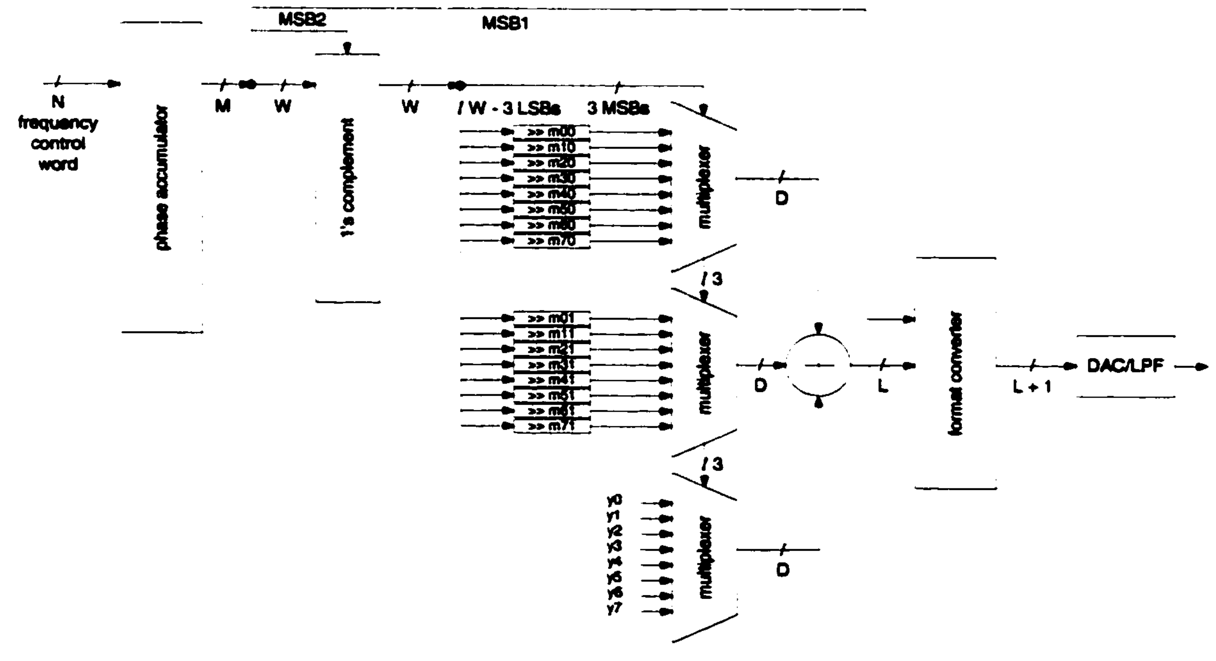

Figure 6-8 - Novel multiplier-less DDFS architecture, eight segments, slopes with two digits

Figure 6-9 shows a version of the architecture with 8 linear segments and slopes m_k represented with no more than three positive digits. For this case, the 3 MSBs of the truncated phase again identify one of eight segments and are the multiplexer control signals. Each multiplexer has 8 inputs which are shifted versions of the $W - 3$ least significant bits of the truncated phase or segment initial amplitudes.

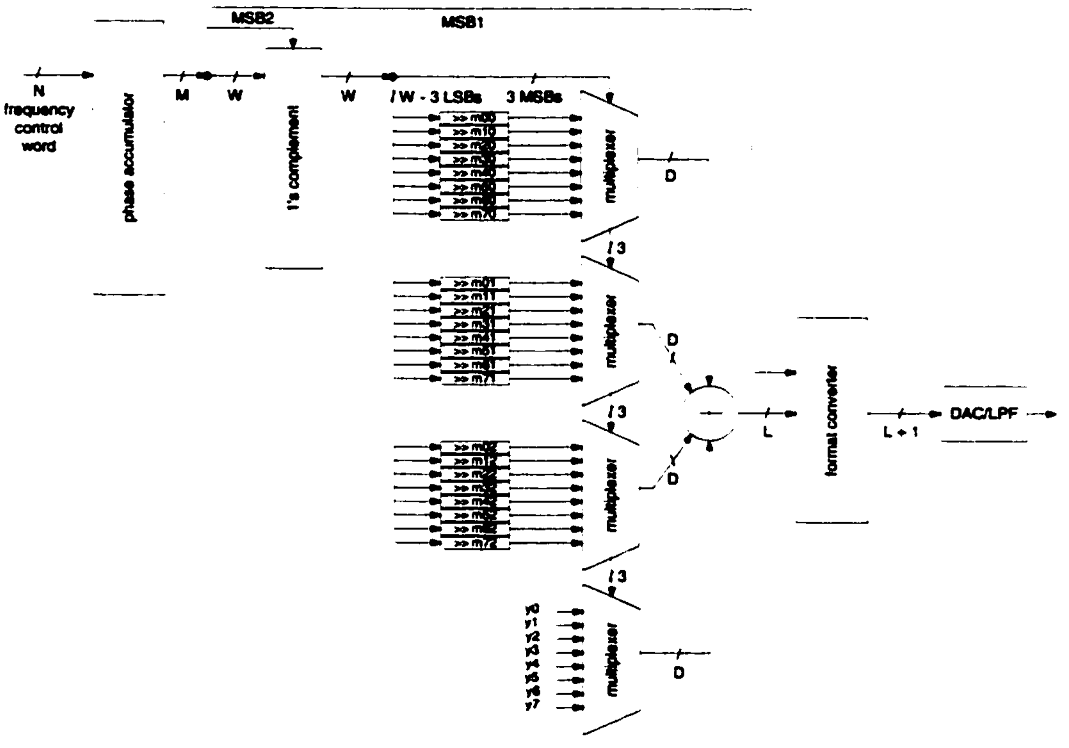

Figure 6-9 - Novel multiplier-less DDFS architecture, eight segments, slopes with three digits

6.1.4 Quadrature Outputs Architecture

The single phase output architecture can be adapted to produce quadrature outputs with a limited increase in hardware costs, following the general architecture discussed in Section 2.2.3, and as presented in [53]. In such a case, the PSAC is divided into two independent parts, each responsible for generating either sine or cosine amplitudes for angles in the first octant. The sine and cosine blocks implement the following equations:

$$\sin(\frac{\pi x}{2}) \cong \begin{cases} y_0 + m_0(x-x_0) & x_0 \le x < x_1 \quad (x_0 = 0) \\ y_1 + m_1(x-x_1) & x_1 \le x < x_2 \\ \vdots & \\ y_k + m_k(x-x_k) & x_k \le x < x_{k+1} \\ \vdots & \\ y_{\frac{s}{2}-1} + m_{\frac{s}{2}-1}(x-x_{\frac{s}{2}-1}) & x_{\frac{s}{2}-1} \le x < x_{\frac{s}{2}} \quad (x_{\frac{s}{2}} = \frac{1}{2}) \end{cases} \qquad (6\text{-}3)$$

$$\cos(\frac{\pi x}{2}) \cong \begin{cases} w_0 - n_0(x-x_0) & x_0 \le x < x_1 \quad (x_0 = 0) \\ w_1 - n_1(x-x_1) & x_1 \le x < x_2 \\ \vdots & \\ w_k - n_k(x-x_k) & x_k \le x < x_{k+1} \\ \vdots & \\ w_{\frac{s}{2}-1} - n_{\frac{s}{2}-1}(x-x_{\frac{s}{2}-1}) & x_{\frac{s}{2}-1} \le x < x_{\frac{s}{2}} \quad (x_{\frac{s}{2}} = \frac{1}{2}) \end{cases} \qquad (6\text{-}4)$$

for x in the interval [0, ½[, hence each block only calculates an approximation of the sine or cosine functions for first octant angles.

The coefficients m_k and y_k, $k = 0, 1, 2, ..., s-1$, for a single phase output design can be converted for use in a quadrature output design with simple linear geometry. The resulting output sequence must be rigorously identical for each design in order to guarantee the same SFDR.

The first half of the coefficients m_k and y_k, for $k = 0, 1, 2, ..., s / 2 - 1$, are used without modification to implement the sine block. For the cosine block, the slope coefficients and segment initial amplitudes are denoted by the symbols n_k and w_k, respectively. These coefficients can be calculated from the second half of the single phase coefficients, i.e. for $k = s / 2, s / 2 + 1, s / 2 + 2, ..., s-2, s-1$:

$$n_{s-k-1} = m_k \qquad (6\text{-}5)$$

$$w_{s-k-1} = y_k + m_k \times (x_{k+1} - x_k - 2^{-W}) \qquad (6\text{-}6)$$

where $W = M - 2$ is the width of the truncated phase angle x after removal of the two MSBs for quadrant symmetry. Equation (6-6) is illustrated in Figure 6-10. The distance between two samples within one segment is equal to 2^{-W}. For the segment shown on the left, the amplitude of the last sample that is calculated becomes the initial amplitude w_{s-k-1} of the mirrored segment, shown on the right of the figure, and its value is given by Equation (6-6).

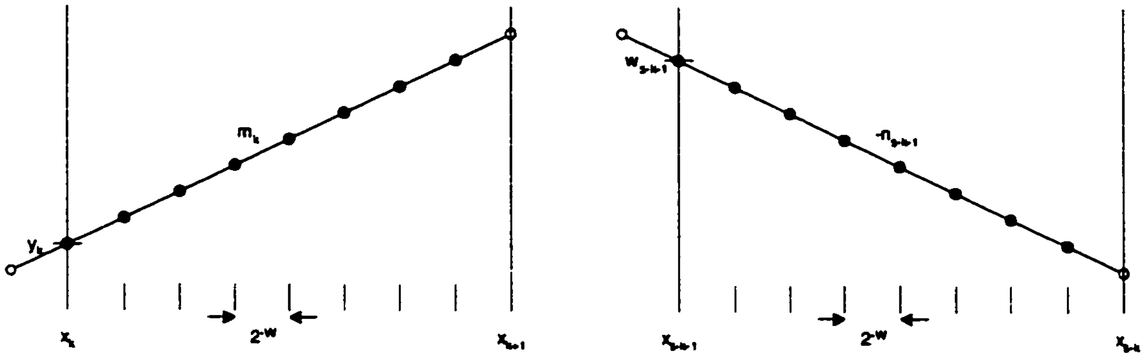

Figure 6-10 - Coefficient conversion for quadrature DDFS

If the segments are equal in length, then Equation (6-6) becomes

$$w_{s-k-1} = y_k + m_k \times (s^{-1} - 2^{-W}) \qquad (6\text{-}7)$$

for $k = s/2, s/2 + 1, \ldots, s - 2, s - 1$.

Applying this conversion process to the single-phase example shown in Figure 6-8 results in the architecture shown in Figure 6-11. The cost in multiplexers is unchanged, as compared with the single phase case, since the number of multiplexers is doubled but their number of inputs is halved. The complexity increase comes from an extra multi-operand adder, two 2:1 bus multiplexers and two XOR gates.

In Figure 6-11, it is assumed that the DAC/LPF can accept data in sign and magnitude format. If that is not the case, then two format converters must be added.

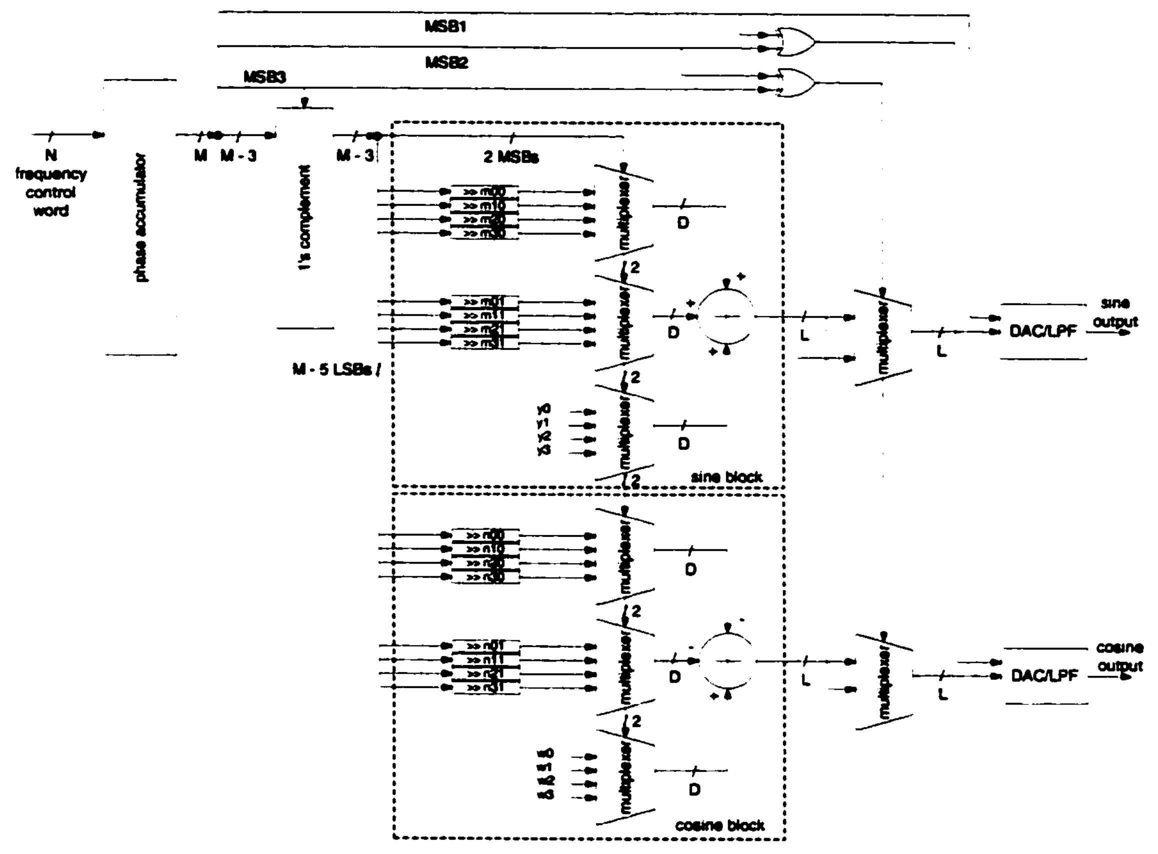

Figure 6-11 - Novel multiplier-less QDDFS architecture, eight segments, slopes with two digits

6.2 High Level PSAC Complexity Metrics

Step 10 of the design procedure detailed in Section 5.5 recommends that system complexity be evaluated once a set of system coefficients and parameters have been selected. Since several such sets may satisfy design requirements, it is useful to compare the resulting designs and to pick the one with the least complexity. One approach to compare the resulting designs would be to synthesize each one and extract complexity and performance parameters from the synthesis tool. However, experience has shown that there can easily be hundreds of coefficient sets for which the SFDR varies by less than 1 dB. It is therefore impractical to generate a Hardware Description Language (HDL) definition of each system, to synthesize them for a chosen library, and to obtain complexity metrics from the synthesis tool.

6.2.1 System Coefficients and Parameters

High level complexity metrics can be extracted directly from a set of coefficients and parameters. These coefficients and parameters are:

- the number of segments s;
- the segment slopes m_k;
- the segment initial amplitudes y_k;
- the number of truncated phase bits M;
- the number of bits used to quantize the sample amplitudes, L;
- the number of bits C used to quantize the initial amplitudes y_k; and,
- the number of added columns of bits, D.

From the segment slopes m_k, we obtain the number of powers of two used to represent them, p. The number of addends is therefore equal to $p + 1$, as described previously.

6.2.2 Complexity Metrics

The following high level complexity metrics can be easily calculated from a design's coefficients and parameters, based on the architecture described in Section 6.1. They are:

- the number of ROM bits;
- the number of 2:1 multiplexers that provide shifted versions of the phase value; and,
- the number of full adders.

In order to simplify the comparison between two designs, it is useful to combine these three metrics into a single number. One possibility consists of assigning a number of transistors to each block such as multiplexers and full adders. The number of transistors per block must be selected carefully, since it can vary depending on the technology used for the actual implementation.

In the present architecture, the 'ROM' is implemented with constant-input multiplexers, and its cost can be calculated as such.

The following general assumptions are made:

- a single phase output architecture is targeted;
- the cost of the phase accumulator is not considered;
- the cost of logic overhead to exploit quadrant symmetry is not considered;
- the cost of an output data format converter is not considered; and,
- the cost of pipelining registers is not considered.

6.2.3 Number of ROM Bits

The number of ROM bits that must be stored is equal to:

$$ROMbits = s \times C \tag{6-8}$$

A 2^s:1 multiplexer can be realized using (s − 1) 2:1 multiplexers. Assuming that the ROM is implemented with hardwired-input multiplexers, the corresponding number of 2:1 multiplexers is

$$muxROM = (s-1) \times C \tag{6-9}$$

6.2.4 Number of 2:1 Multiplexers

The number of multiplexers that provide shifted versions of the phase angle is equal to p. Each one has s inputs, and each input is $D - \log_2 s$ bits wide. Since a 2^s:1 multiplexer can be realized using (s − 1) 2:1 multiplexers, the corresponding number of 2:1 multiplexers is

$$muxShift = p \times (D - \log_2 s) \times (s-1) \tag{6-10}$$

The estimate provided by Equation (6-10) doesn't take into account the complexity reduction incurred when multiplexer inputs can be combined. For example, consider a 4:1 multiplexer whose input ports 0 and 1 are tied together. The number of 2:1 multiplexers needed to build this 4:1 multiplexer is equal to 2 instead of 3, if all four inputs were different.

The reduction in the number of multiplexers can be calculated precisely by a careful inspection of the powers of two that are summed to make up the slope coefficients m_k.

6.2.5 Number of Full Adders

The final number of full adders in the implemented system varies greatly with performance requirements, since ripple, carry select and carry look-ahead adders have different complexities. Therefore, for these high level metrics, it is assumed that the multi-operand addition is realized by a number of 3:2 compression stages, followed by a simple ripple carry adder.

The number of addends $p + 1$ is equal to the number of signed digits used to represent the slopes m_k, plus one. In order to compress $p + 1$ addends to 2, the number of full adders required is equal to $p - 1$. It is assumed that D columns of bits must be added. The final ripple carry adder is D bits wide, and a rounding operation must be done to L bits. The estimate is simplified by neglecting the initial half adder for the LSB and the rounding logic, and by assuming that the final adder is realized with D full adders. The total number of full adders is therefore equal to:

$$fullAdders = (p-1)\times D + D = p\times D \tag{6-11}$$

6.2.6 Total Cost

The total cost T (in transistors) can be estimated as:

$$T = fullAdders \times Cost_{FA} + (muxROM + muxShift)\times Cost_{mux2:1} \tag{6-12}$$

For this research, we were targeting a 0.18 μm CMOS library provided by the Canadian Microelectronics Corporation (CMC). A careful inspection of the standard full adder and multiplexer cells led to cost selections of 6 and 24 transistors for the 2:1 multiplexer and the full adder, respectively.

6.3 Design Process and Design Examples

In order to validate the architecture presented in this chapter as well as the coefficient selection procedure described in Section 5.5, several DDF Synthesizers achieving SFDRs in the range of 60 to 96 dBc were designed.

6.3.1 Design Space Exploration and Selection of Coefficient and Parameter Sets

Designs with $s \in \{8, 16, 32, 64\}$ segments were considered. In each case, the design space exploration procedure included the following steps:

1. Establishing a list of sets of acceptable segments slopes m_k (part II, Section 5.5);
2. Calculating the corresponding segment initial amplitudes y_k (part III, Section 5.5) for each set of segment slopes in the list;
3. Calculating the SFDR and high-level complexity metrics for reasonable system parameters M, L, C and D (part IV, Section 5.5); and,
4. Selecting one set of coefficients and parameters achieving approximately 60, 72, 80, 84, 92 and 96 dBc with the least high-level complexity metric estimate.

6.3.2 Candidate Slopes

A key aspect of the new architecture comes from the restrictions imposed on the segment slopes. There are three considerations: the number of signed digits used to represent a slope, the dynamic range of the signed digits used in a set of slopes, and whether or not to allow negative digits.

The number of signed digits used to represent the segment slopes m_k has an important impact on system complexity. Slopes with 1, 2 and 3 signed digits were considered. For designs with less than 72 dBc of SFDR, performances very close to the bound specified by Equation (5-54) were achieved with only two signed digits. Above that level, it was found that three signed digits were sufficient up to and including designs with 64 segments, the highest number investigated. Sub-optimal designs (32 segments with 80 dBc SFDR and 64 segments with 92 dBc SFDR) were produced using only two signed digits to represent the slopes.

A secondary consideration is the dynamic range of the signed digits, which corresponds to the width of the slopes (E in Figure 5-1). It has already been observed that, for first quadrant angles, the first derivative of the scaled sine function $\sin(\pi x/2)$ is in the interval $[0, \pi/2]$. Hence, the greatest digit used was 2^0. Again, it was found that the smallest necessary digit depended on the SFDR to be achieved. For 8-segment designs achieving 60 dBc SFDR, the smallest digit used was 2^{-3}. For 64-segment, 96-dBc SFDR designs, it was 2^{-5}.

A third consideration when restricting signed digit choices is whether to allow negative digits. The importance of this restriction depends on the impact on overall system complexity. For eight segments, 60 dBc SFDR designs, bit inversions have a small but noticeable impact. For 64 segments, 96 dBc SFDR designs, the impact is negligible.

Candidate slopes m_k were considered from the sets shown in Table 6-1.

slopes	max. # of digits	E	all digits >0?
{6, 5, 4, 3, 2, 1, 0} / 4	2	3	yes
{12, 10, 9, 8, 6, 5, 4, 3, 2, 1, 0} / 8	2	4	yes
{12, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0} / 8	2	4	no
{24, 20, 18, 17, 16, 12, 10, 9, 8, 6, 5, 4, 3, 2, 1, 0} / 16	2	5	yes
{24, 20, 18, 17, 16, 15, 14, 12, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0} / 16	2	5	no
{48, 40, 36, 34, 33, 32, 24, 20, 18, 17, 16, 12, 10, 9, 8, 6, 5, 4, 3, 2, 1, 0} / 32	2	6	yes
{48, 40, 36, 34, 33, 32, 31, 30, 28, 24, 20, 18, 17, 16, 15, 14, 12, 10,9, 8, 7, 6, 5, 4, 3, 2, 1, 0} / 32	2	6	no
{13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0} / 8	3	4	choice
{26, 25, 24, 22, 21, 20, 19, 18, 17, 16, 14, 13, 12, ..., 1, 0} / 16	3	5	yes
{26, 25, 24, ..., 1, 0} / 16	3	5	no
{52, 50, 49, 48, 47, 46, 44, 42, 41, 40, ..., 1, 0} / 32	3	6	no

Table 6-1 - Candidate slopes sets

6.3.3 Establishing a List of Groups of Acceptable Slopes

Using each of the candidate slopes sets in Table 6-1, a search was performed to find groups of slopes for which a "large" SFDR could be attained. Each group was evaluated according to the criteria described at step 4 in Section 5.5. The search process will be described with an example.

Consider the first quadrant of function sin($\pi x/2$), separated into 8 segments. The first three columns Table 6-2 below respectively show the segment lower bounds x_k, the function amplitude at those points, and the floating point value of linear segment slopes m_k connecting the points.

x_k	sin($\pi x/2$)	slope m_k	quantized m_k (choice 1)	quantized m_k (choice 2)	quantized m_k (choice 3)
0	0.0000 ...	12.486 ... / 8	12 / 8	12 / 8	12 / 8
0.125	0.1951 ...	12.006 ... / 8	12 / 8	12 / 8	12 / 8
0.25	0.3827 ...	11.065 ... / 8	12 / 8	10 / 8	10 / 8
0.375	0.5556 ...	9.698 ... / 8	10 / 8	9 / 8	10 / 8
0.5	0.7071 ...	7.959 ... / 8	8 / 8	8 / 8	8 / 8
0.625	0.8315 ...	5.914 ... / 8	6 / 8	6 / 8	6 / 8
0.75	0.9239 ...	3.642 ... / 8	4 / 8	4 / 8	3 / 8
0.875	0.9808 ...	1.230 ... / 8	1 / 8	1 / 8	1 / 8

Table 6-2 - Selection of a slope set, 8 segments case

The last three columns of Table 6-2 show three possible groups of choices of slopes m_k quantized according to the second candidate slopes set in Table 6-1, i.e. two non-zero digits, $E = 4$, and all digits positive. The choice of 12 / 8 for m_0 and m_1 appears obvious when inspecting the floating point values in the third column. However, the choice for m_2 is not so obvious, since 11 / 8 is not part of the candidate set being considered. Hence, there are several possibilities that must be evaluated. Using the criteria summarized at step 4 in Section 5.5, and specifically Equation

(5-69), it is found that groups of choices 1, 2 and 3 in Table 6-2 can achieve at most 60.22, 60.03 and 59.17 dBc of SFDR, respectively. Since Equation (5-54) predicts a maximum achievable SFDR of approximately 60.21 dBc for eight segments, these three groups should be considered as possible design candidates. Although group 3's maximum SFDR of 59.17 dBc may seem low, it is observed that the first four slopes can be split into two pairs, leading to an eventual efficient hardware realization with multiplexers.

This manual approach to the selection of slope sets is obviously very time consuming. Instead, a simple algorithm was implemented to perform an exhaustive search among candidate slope sets. The algorithm was used for designs with $s = 8$ and 16 segments, and those groups of slopes with the potential of achieving SFDRs close to the bound predicted by Equation (5-54) were retained.

For designs with 32 and 64 segments, or when considering slope sets with a dynamic range of 5 or 6 bits, an exhaustive search was not possible due to the large number of combinations. Instead, a Genetic Algorithm (GA) [25] was used yielding very good results in reasonable time.

GAs offer an attractive way to solve complex optimization problems. They are based on genetic principles of natural selection and evolution. An initial "population" of solutions that represent points in a multi-dimensional search space is first randomly selected. Each candidate solution in the population is called a "chromosome". The "fitness" of each chromosome is evaluated using an objective function. A new "generation" of chromosomes is obtained by selecting pairs of "parent" chromosomes from the present generation and intermixing parts of their "genes". The probability of selecting a chromosome for "reproduction" is proportional to its relative fitness in the population, hence better solutions have a higher chance of contributing to the next generation of solutions. "Mutations" can also take place during the reproduction process by introducing random variations within chromosomes. Several generations of solutions can quickly be computed until an acceptable solution is found.

The setup of a GA involves making several choices. These include the initial population size and the number of generations to go through. Two probability thresholds are also often selected. The first is the probability of crossover, P_c, which determines whether genes from two parents are intermixed to produce offspring. The second is the probability of mutation P_m. A high value of P_c, e.g. > 0.9, is normally selected. P_m is normally much smaller, e.g. < 0.01. Finally, "elitism" can be implemented, whereby a few best chromosomes are copied directly into the next generation. This prevents the best solutions from being discarded, whether or not they are selected for reproduction.

One common difficulty in applying GAs to a problem is the coding of the search space. A chromosome is encoded as a string of bits ("genes"), each representing a part of a solution. In the present case, however, this coding is straightforward. Each chromosome must specify s segment slopes m_k. Since we have chosen to restrict the slopes to a limited set of acceptable values, the chromosomes encode the m_k with their position in the set. As an example, the first four slopes sets in Table 6-1 contain no more than 16 elements. Hence, encoding any one slope in one of these sets requires only 4 bits. Since this provides a total of 16 possibilities, some values may not be used for those sets with fewer than 16 elements. When calculating the fitness function of a chromosome, the value 0 is automatically assigned if one of the disallowed possibilities occurs.

The selected fitness function was the achievable SFDR predicted by Equation (5-69).

Several values for the initial population size in the range from 50 to 5000 were considered. A small population size is sufficient for a small search space but leads to premature convergence for larger search space. However, a large population size increases the algorithm's running time significantly. The probability of crossover was maintained between 0.9 and 0.99, and the probability of mutation between 0.01 and 0.001. Elitism was used.

It was found that convergence times were significantly improved by carefully choosing the initial population of chromosomes. While randomness is essential to have a representative set of solutions across the search space, it is useful to "help nature" a little. This was done as follows. As demonstrated in Table 6-2, the floating point values of the segment slopes connecting sine amplitudes evaluated at the segment bounds were calculated. A 'base' chromosome was formed by selecting quantized slopes for each segment that minimized the error between the floating point values and an element in the selected slope set. The population of chromosomes was then established by taking the base chromosome and randomly varying the quantized slopes within the selected slope set. The amplitude of these variations was kept within a few positions inside the slope set.

6.3.4 Exhaustive Search on M, L, C and D and Selection

With acceptable sets of slopes selected for each of the number of segments $s \in \{8, 16, 32, 64\}$, and corresponding values of y_k represented with floating point precision, an exhaustive search of system parameters M, L, C, and D was then conducted. It was observed that an exhaustive search for reasonable values such as $L \in [7, 19]$, $M \in [10, 18]$, $C \in [9, 16]$ and $D \in [10, 16]$ was well within the capabilities of a desktop computer, i.e. with a search time in the order of one hour, even for designs with $s = 64$ segments.

For each resulting list of design parameters and coefficients, the SFDR and high level complexity were evaluated. A performance metric equal to the ratio of the SFDR to the estimated complexity was calculated for each design. For this metric, the SFDR was expressed in absolute format, i.e. not in dB. The designs were then placed in decreasing order of performance. Among the top performers, a single design was selected according to a comparison of all performance parameters in relation to similar designs reported in the literature. This included the SFDR, the number of ROM bits, and the truncated phase width.

Table 6-3 lists the final six coefficients sets that were selected. The third column, marked 'T', gives the high-level estimated cost in transistors according to Equation (6-12). The first three designs listed have slopes m_k represented with at most two signed digits. This is also true for the 91.9 dBc, 64 segments design. The other two designs, achieving 84.2 dBc and 96.2 dBc, have slopes represented with at most three signed digits.

Figure 6-12 plots the estimated cost in transistors T for each design as a function of the achieved SFDR. A second order polynomial curve is superimposed over the data, showing a very good match and a quadratic relation between complexity and SFDR.

s	SFDR	T	M	L	C_f	D	m_k	y_k
8	60.1	1294	13	7	10	11	{12, 12, 10, 9, 8, 6, 4, 1} / 8	{2, 191, 384, 552, 697, 819, 909, 971} / 1024
16	71.7	2116	13	9	10	11	{20, 20, 20, 20, 18, 17, 16, 15, 14, 12, 10, 9, 7, 5, 3, 1} / 16	{0, 81, 161, 238, 317, 391, 460, 525, 584, 640, 689, 729, 764, 792, 811, 823} / 1024
32	80.0	4594	15	11	12	13	{48, 48, 48, 48, 48, 48, 48, 48, 48, 48, 40, 40, 40, 40, 36, 36, 34, 31, 31, 28, 24, 24, 20, 20, 17, 15, 12, 10, 8, 7, 3, 0} / 32	{2, 198, 394, 588, 781, 971, 1159, 1345, 1527, 1704, 1894, 2063, 2226, 2384, 2543, 2690, 2833, 2970, 3095, 3219, 3336, 3437, 3537, 3621, 3702, 3773, 3836, 3887, 3930, 3961, 3988, 4004} / 4096
32	84.2	5734	16	11	14	14	{25, 25, 25, 25, 24, 24, 24, 23, 23, 22, 22, 21, 21, 20, 19, 18, 17, 16, 16, 15, 13, 12, 11, 10, 9, 8, 7, 6, 4, 3, 2, 1} / 16	{0, 800, 1598, 2391, 3192, 3971, 4737, 5508, 6248, 6988, 7696, 8399, 9065, 9723, 10359, 10971, 11552, 12106, 12616, 13108, 13585, 14013, 14408, 14767, 15088, 15375, 15624, 15834, 16022, 16156, 16250, 16306} / 16384
64	91.9	9520	18	13	15	16	{40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 36, 36, 36, 36, 36, 36, 36, 36, 34, 34, 33, 32, 33, 31, 31, 30, 30, 28, 28, 28, 28, 24, 24, 24, 24, 24, 20, 20, 20, 18, 18, 17, 15, 15, 14, 14, 12, 12, 10, 10, 8, 7, 7, 5, 4, 4, 3, 2, 0} / 32	{6, 659, 1311, 1963, 2612, 3258, 3905, 4547, 5187, 5826, 6459, 7087, 7713, 8333, 8949, 9590, 10191, 10788, 11378, 11962, 12539, 13105, 13665, 14234, 14776, 15316, 15850, 16356, 16876, 17372, 17863, 18335, 18814, 19264, 19701, 20127, 20576, 20978, 21365, 21741, 22105, 22486, 22822, 23143, 23467, 23760, 24046, 24326, 24577, 24820, 25041, 25262, 25450, 25640, 25800, 25958, 26093, 26206, 26316, 26404, 26468, 26526, 26567, 26599} / 32768
64	96.2	12040	18	13	15	16	{50, 49, 49, 49, 49, 49, 49, 49, 48, 48, 48, 48, 47, 47, 46, 46, 46, 44, 44, 44, 44, 42, 42, 41, 41, 40, 39, 39, 38, 37, 36, 36, 35, 34, 33, 32, 31, 30, 29, 28, 27, 26, 25, 24, 23, 22, 21, 19, 18, 17, 16, 15, 14, 12, 12, 10, 9, 8, 7, 5, 4, 3, 2, 0} / 32	{0, 794, 1585, 2374, 3162, 3947, 4729, 5509, 6293, 7066, 7834, 8596, 9361, 10113, 10866, 11605, 12335, 13074, 13790, 14498, 15195, 15899, 16578, 17255, 17914, 18568, 19212, 19836, 20455, 21062, 21657, 22230, 22798, 23351, 23890, 24415, 24926, 25420, 25900, 26364, 26811, 27243, 27658, 28055, 28437, 28801, 29146, 29482, 29795, 30087, 30362, 30618, 30855, 31083, 31275, 31464, 31627, 31770, 31894, 32007, 32092, 32159, 32205, 32241} / 32768

Table 6-3 - Selected sets of parameters and coefficients

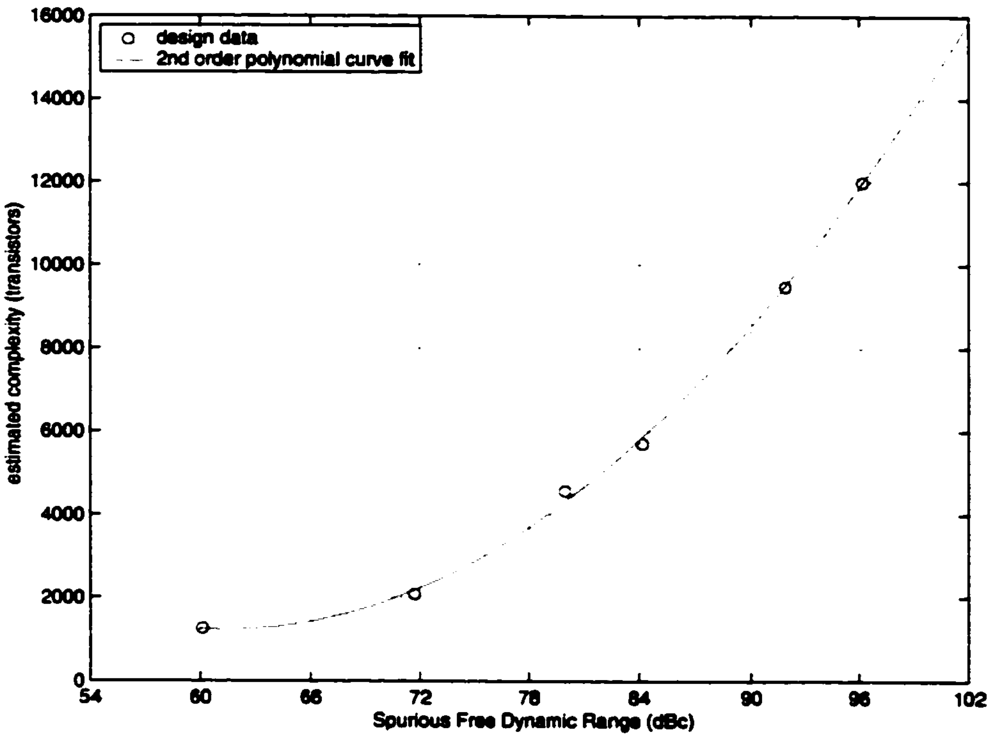

Figure 6-12 - Estimated PSAC complexity versus SFDR

6.3.5 Output Data and Spectral Plots

This section contains output data graphs showing error amplitudes, as well as spectral plots for each of the designs listed in Table 6-3.

Figure 6-13, corresponding to the 8 segments, 60 dBc SFDR design, shows the PSAC output sequence for angles corresponding to the first quadrant. It also shows the error between an ideal sinusoid and the output sequence, as defined by Equation (2-16). The amplitude A of the ideal sinusoid is approximately 123 / 128, and the MAE is approximately 0.01 or 1.29 LSB. The left axis of the figure is for the output sequence and the right axis is for the error amplitudes. The eight segments are clearly noticeable, as are the finite precision quantization effects.

Figure 6-14 shows the corresponding output spectrum for an odd value of FCW corresponding to an output frequency approximately equal to 0.22 times the reference frequency. As discussed in Section 2.3.1, this spectrum is representative of all other cases where the FCW would be odd. In all other cases, the position of noise spurs would change, but their number and amplitudes would not.

The remainder of the Figures show similar plots for the five other designs listed in Table 6-3.

Considering Figure 6-15 and Figure 6-21, it is noticed that the amplitude of the approximated sinusoid is approximately equal to 0.8, contrary to other designs where it approaches 1.0. As a consequence, the full dynamic range allotted by the output data width is underutilized. This could be seen as a problem for analog output designs, as the noise introduced by the DAC would then have a more important effect.

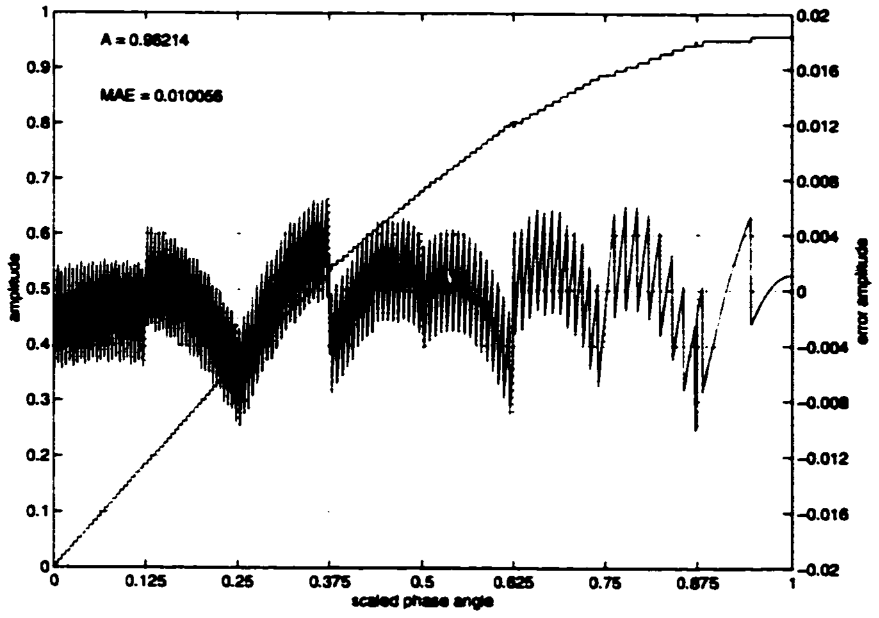

Figure 6-13 - Output data and error, 8 segments, 60.1 dBc SFDR design

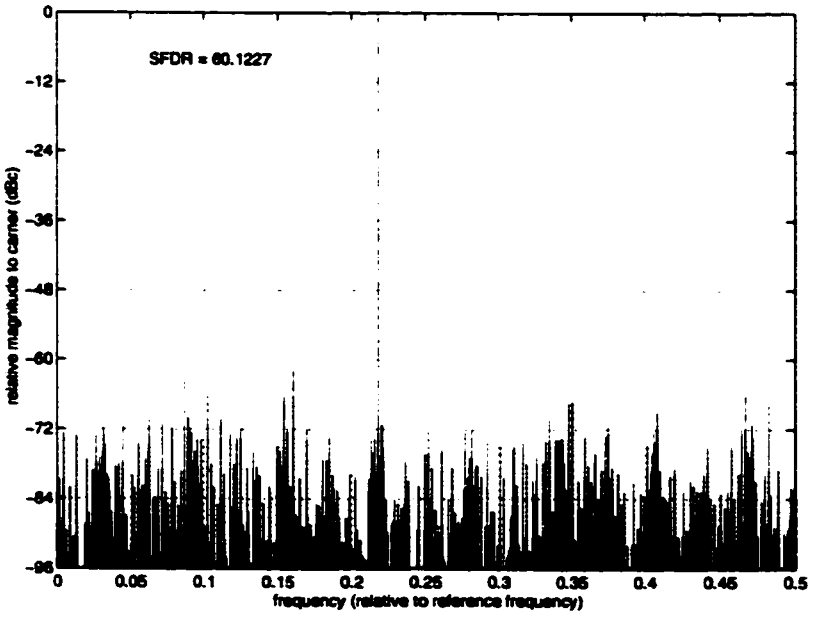

Figure 6-14 - Typical spectral plot, 8 segments, 60.1 dBc SFDR design

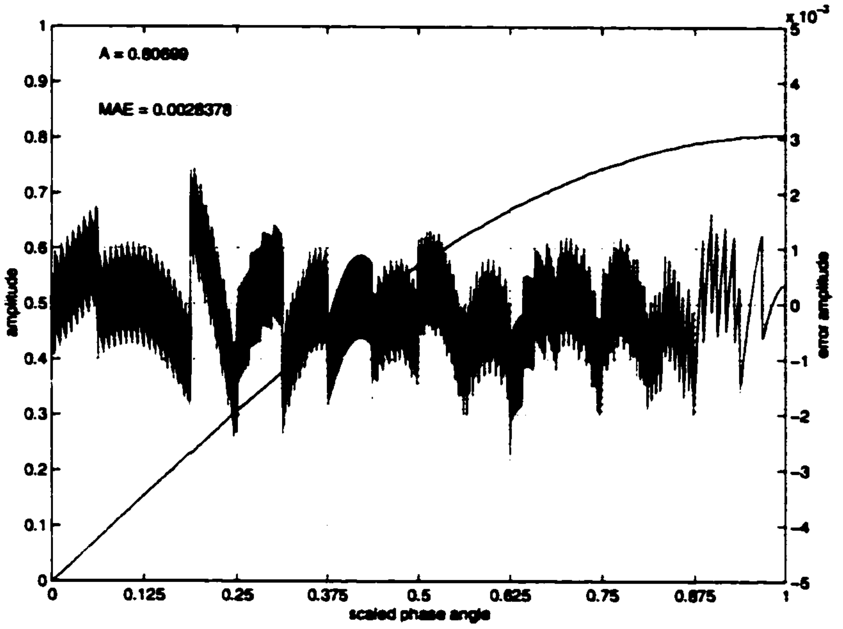

Figure 6-15 - Output data and error, 16 segments, 71.7 dBc SFDR design

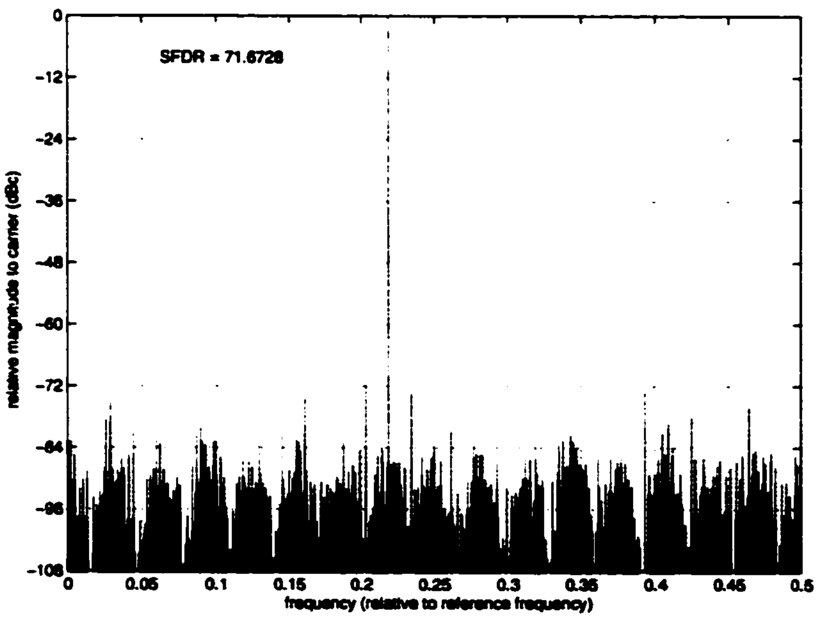

Figure 6-16 - Typical spectral plot, 16 segments, 71.7 dBc SFDR design

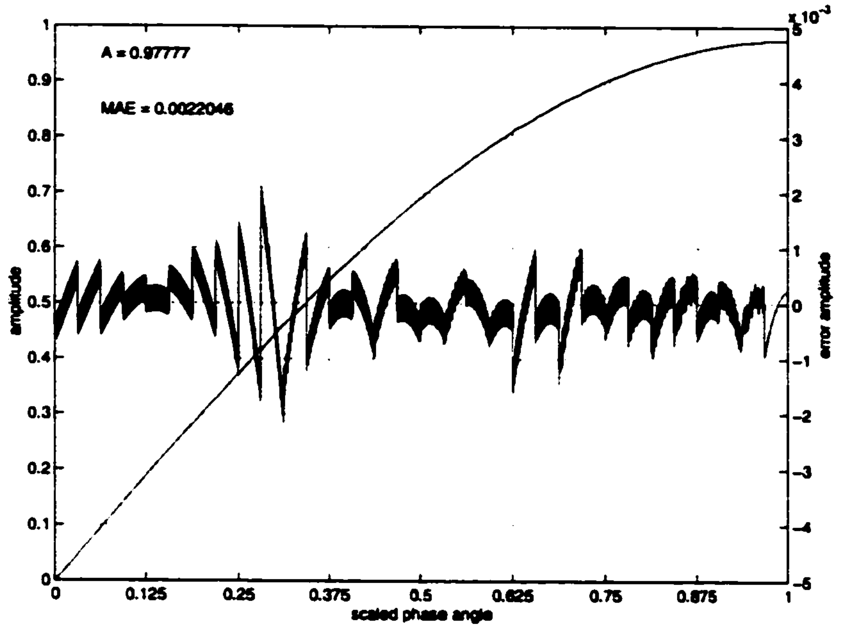

Figure 6-17 - Output data and error, 32 segments, 80.0 dBc SFDR design

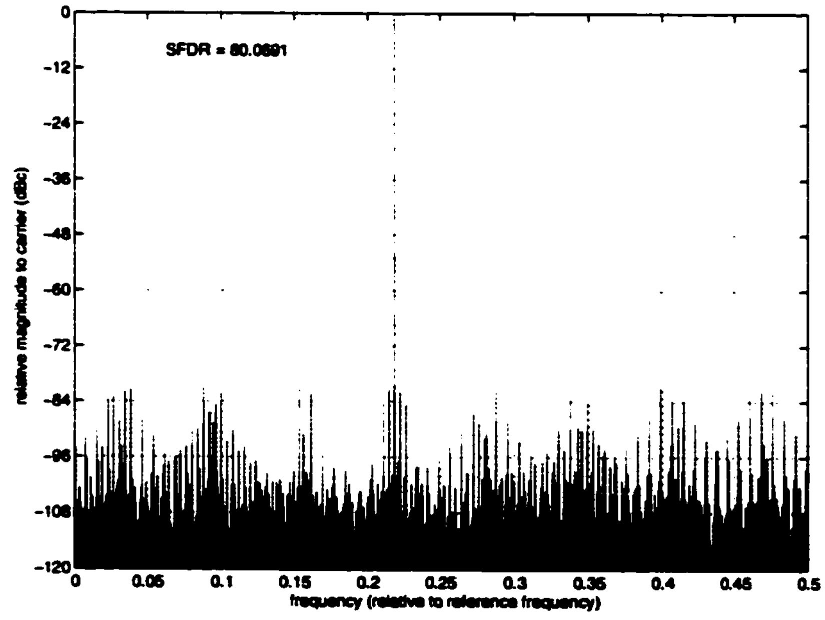

Figure 6-18 - Typical spectral plot, 32 segments, 80.0 dBc SFDR design

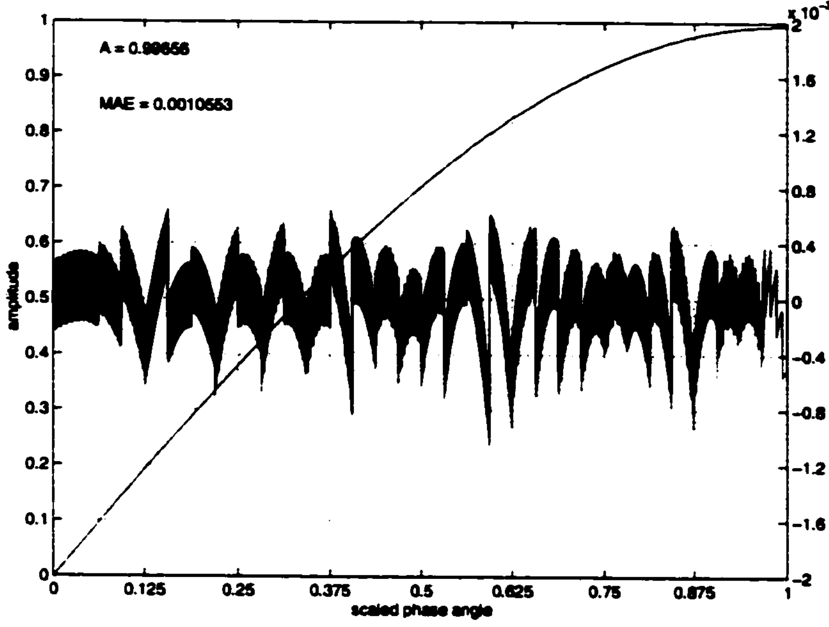

Figure 6-19 - Output data and error, 32 segments, 84.2 dBc SFDR design

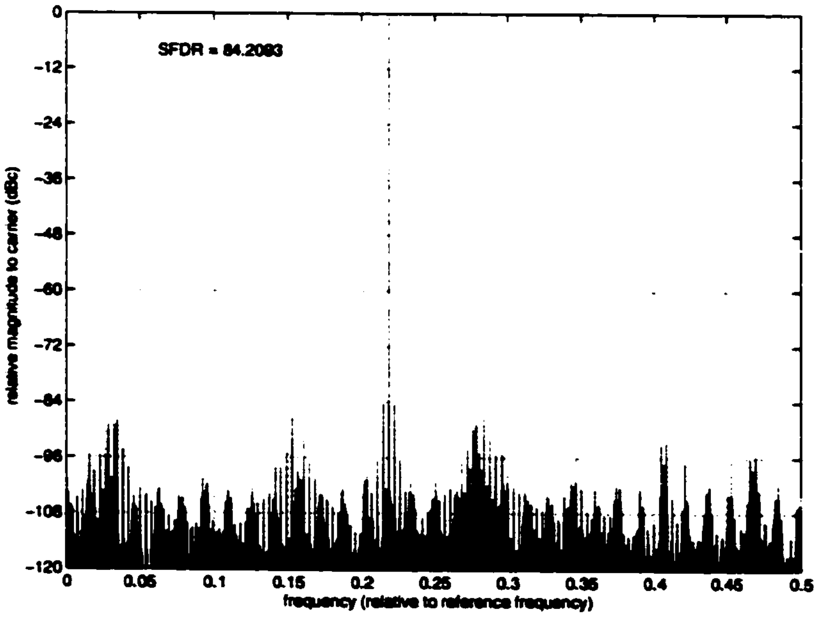

Figure 6-20 - Typical spectral plot, 32 segments, 84.2 dBc SFDR design

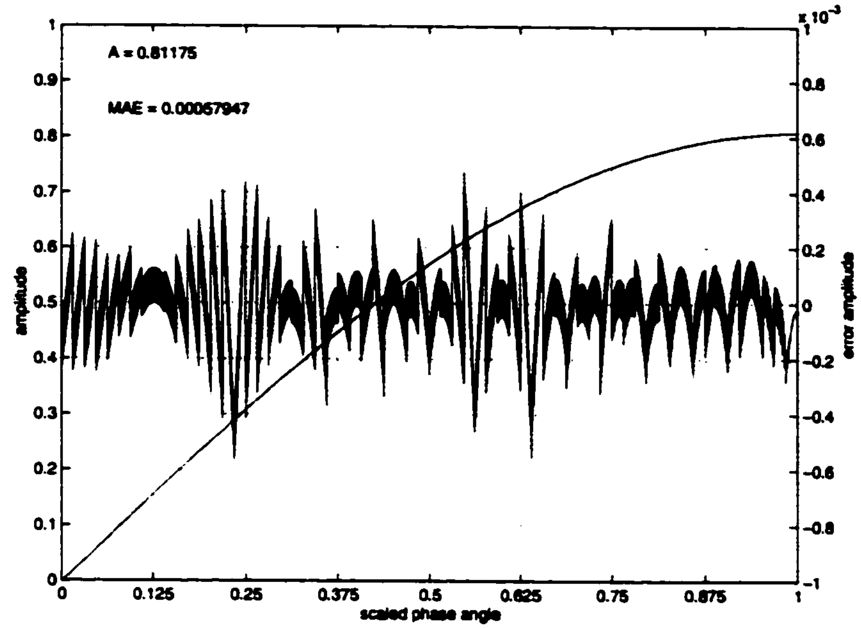

Figure 6-21 - Output data and error, 64 segments, 91.9 dBc SFDR design

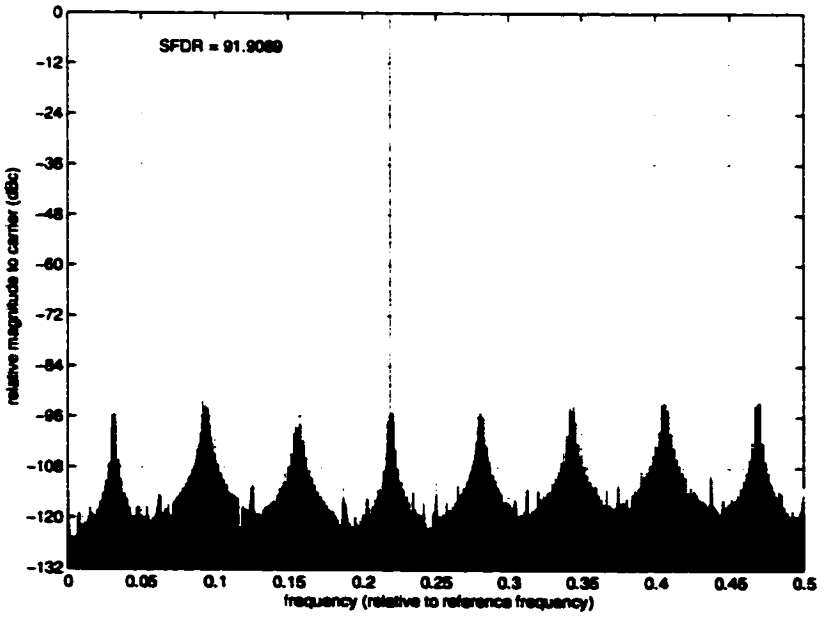

Figure 6-22 - Typical spectral plot, 64 segments, 91.9 dBc SFDR design

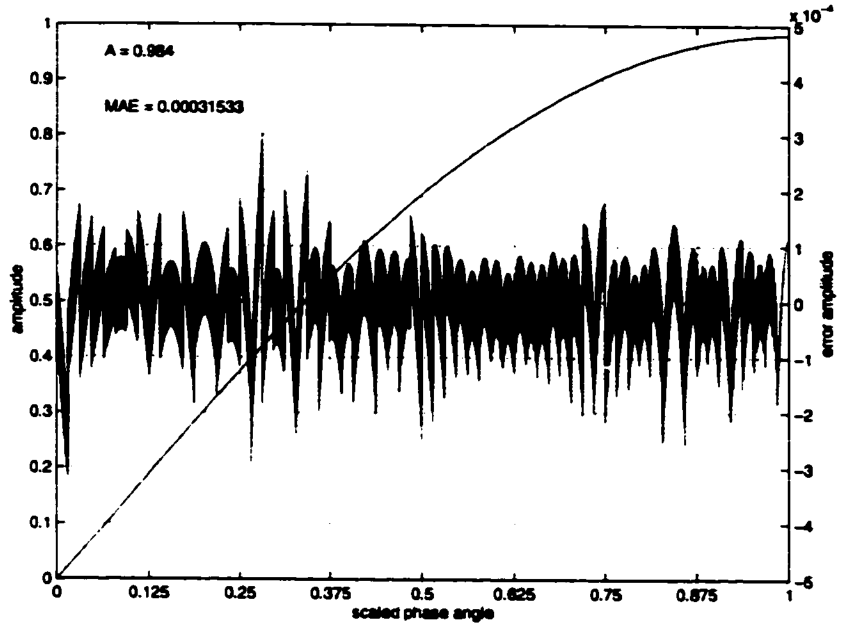

Figure 6-23 - Output data and error, 64 segments, 96.2 dBc SFDR design

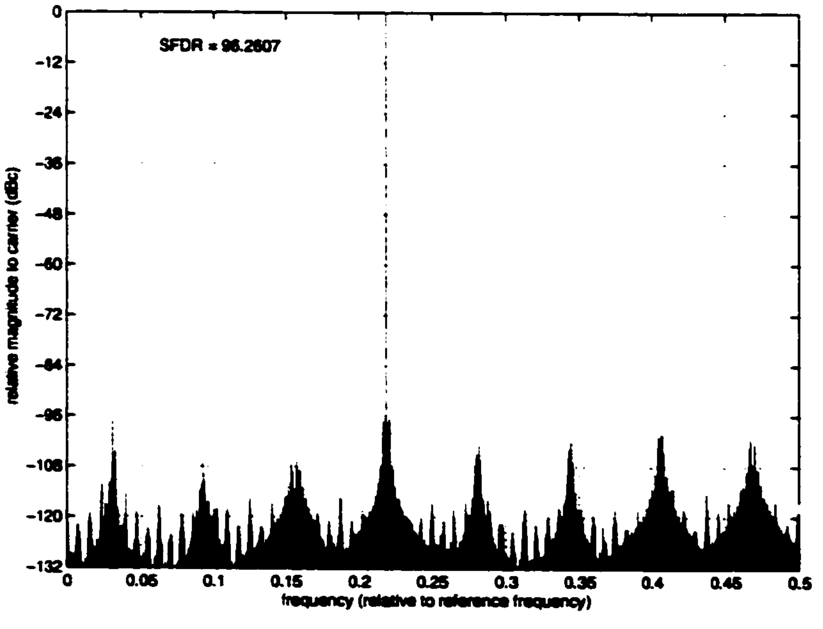

Figure 6-24 - Typical spectral plot, 64 segments, 96.2 dBc SFDR design

6.4 HDL System Description

6.4.1 Automated VHDL Code Generation

Inspection of Figure 6-1 reveals a regular structure that can be easily parameterized. The quadrature architecture of Figure 6-11, being a simple extension of the single phase case, can also be easily parameterized. Hence, these architectures lend themselves well to automated code generation.

A Matlab™ script was prepared to facilitate the generation of VHDL code for DDF Synthesizers based on the architecture introduced in this chapter. The code accepts, as inputs, vectors of slopes m_k and initial amplitudes y_k, values for the parameters M, L, C and D, the width of the phase accumulator N, and whether a single phase or a quadrature output design is desired. The output is a syntactically correct, synthesizable VHDL description of a DDFS that matches the specified coefficients and parameters. In the case of a quadrature DDFS, the m_k and y_k parameters supplied are for the single phase case. The procedure and equations described in Section 6.1.4 are followed to calculate the equivalent parameters for the quadrature architecture.

The script performs the following steps:

```
input the slope and initial amplitude coefficients
input system parameters s, N, M, L, C, D
input whether this is a quadrature design
decompose the slope coefficients as a sum of digits
generate the header and entity declarations
declare signals
describe the phase accumulator
describe the control signals
describe the PSAC multiplexers responsible for multiplication
describe the PSAC multiplexer responsible for the initial amplitudes
describe the PSAC final sum and rounding process
describe output data conversion and the output registers
```

The script contains approximately 650 lines of code. The generated VHDL file sizes vary between 170 lines of code for a single phase DDFS with 8 segments achieving 60 dBc of SFDR, to 500 lines of code for a quadrature DDFS with 64 segments achieving 96 dBc of SFDR. The generated VHDL also includes statements for writing output data to a binary file on disk, greatly simplifying testing and verification.

6.4.2 Module Compiler Language™ Description

Although a structural VHDL description is convenient in most cases, it was found that it does not lend itself well to design optimization.

Specifically, most synthesis tools (e.g., Synopsys® FPGA Express™ and Design Compiler™) do not automatically insert pipeline stages in order to increase the clock rate. These tools also do not offer alternative adder architectures to the traditional ripple adder, such as a carry select adder or carry save adder.

Consequently, the use of the Module Compiler™ tool from Synopsys® was investigated [88]. This tool greatly simplifies data-path design. It includes a library of several pre-defined sophisticated and parameterized logic and arithmetic blocks that can be instantiated directly in VHDL or Verilog code. It allows the description of custom blocks in a special HDL called Module Compiler™ Language (MCL), which is very similar in format and syntax to Verilog. These blocks can then be instantiated in other MCL code, or in VHDL or Verilog code.

Module Compiler™ accepts design constraints in terms of system clock rate, area or power consumption. It allows the specification of different types of adders, or will automatically select the best one based on design constraints. It also has the capability of automatically inserting pipeline registers in the data paths where required. From a MCL description and set of design constraints, the tool generates an optimized netlist of cells corresponding to a specified library. A detailed report of cell usage, total cell area, timing and power consumption estimation can be generated. Finally, the resulting netlist can be further optimized by Synopsys® Design Compiler™ automatically.

MCL descriptions were prepared for the quadrature DDFS designs achieving 60, 72, 84 and 96 dBc. The 60 dBc design requires approximately 150 lines of code, and the 96 dBc requires approximately 400.

6.5 Implementation Data and Results

6.5.1 Implementation in Xilinx® FPGA

The VHDL descriptions of the DDFS designs described in Table 6-3 were synthesized, mapped, placed and routed for implementation in a Xilinx® 4010XL FPGA, in order to obtain complexity statistics. Figure 6-25 shows the implementation statistics, in terms of Configurable Logic Block (CLB) usage and equivalent gates, as reported by the implementation tools. In all cases, the data is for a phase accumulator that is $N = 32$ bits wide.

Also shown on the figure are two second order polynomial fitting curves revealing a quadratic relationship between design complexity and SFDR.

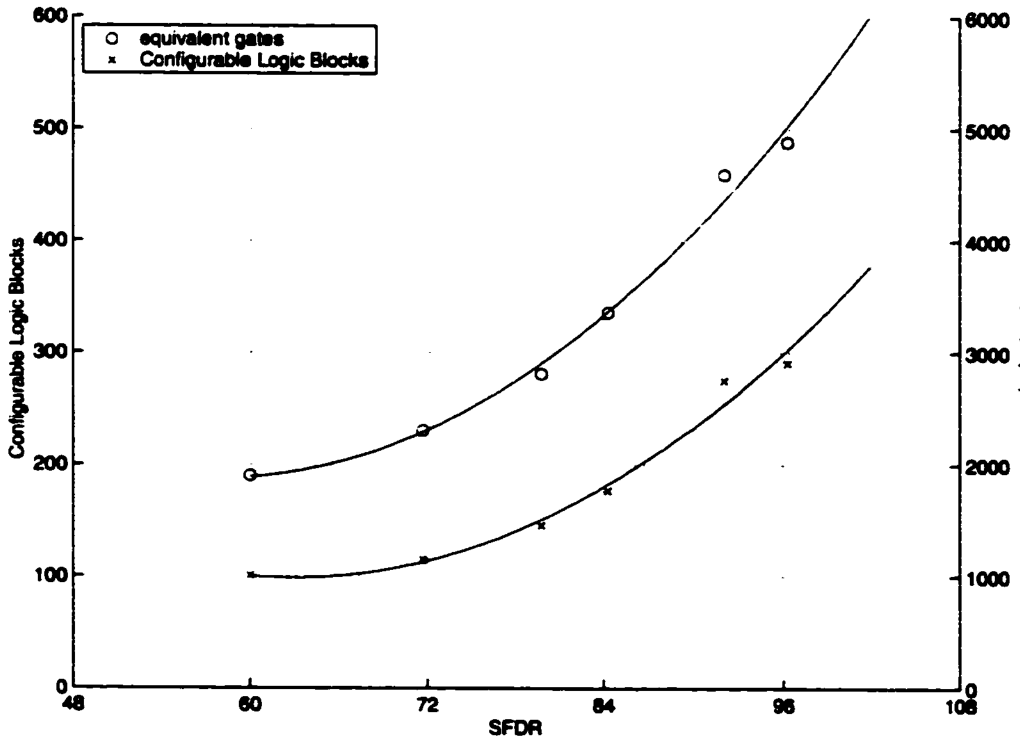

Figure 6-25 - FPGA implementation statistics

6.5.2 Implementation Details for 0.18 μm and 0.35 μm CMOS

From Table 6-3, the four designs achieving 60, 72, 84 and 96 dBc of SFDR were selected for implementation in 0.35 μm and 0.18 μm CMOS. Module Compiler™ Language descriptions were used, and various clock rate design constraints were imposed. The process was repeated for different types of adders and number of pipeline stages. All designs had quadrature outputs.

Implementation data was collected after synthesis and optimization by Synopsys® Design Compiler™. The data included the maximum clock rate, the number of cells, the total cell area and the estimated power consumption. As discussed in Section 2.5.2, the number of transistors was estimated by dividing the total cell area by the area of a basic NAND gate, and by multiplying this quotient by 4. The power consumption estimate assumed random FCW input with uniform distribution, and a load on the core output registers equal to the capacitance of a single output cell. This value is equal to 0.158 pF and 0.025 pF for the 0.35 μm and 0.18 μm libraries, respectively. The power consumption figures were normalized to units of mW/MHz.

Two different output data formats types were considered: sign and magnitude, which requires no extra processing after the PSAC, and unsigned format, which requires extra logic and an extra adder. This format is often required by DACs. Obviously, the impact of this extra adder on overall complexity is more significant for the low SFDR designs.

6.5.3 0.18 μm Implementation Results

Table 6-4 presents data for designs synthesized for a 0.18 μm CMOS library assuming worst case parameters, referred to as a "worst case library". All designs have quadrature outputs and a phase accumulator width of $N = 32$.

Figure 6-26 plots the power consumption as a function of design area. A linear best fit curve through the data shows a linear relationship between these two quantities, which is consistent with intuition.

Figure 6-27 plots the area as a function of SFDR for 12 designs extracted from Table 6-4, divided into 3 categories: designs optimized for area, designs with few pipelines, and designs optimized for high clock rates with several stages of pipelining. Three 2nd order best fit curves through the data again reveal a quadratic relationship between area and SFDR.

SFDR (dBc)	N	latency (cycles)	output type	clock (MHz)	cells	area (μm^2)	transistors	power (mW/MHz)
60.1	32	0	S&M	88	372	11262	3693	0.015
60.1	32	0	S&M	201	766	16791	5507	0.021
60.1	32	8	S&M	709	1443	73807	24205	0.127
71.7	32	0	unsigned	88	520	13961	4579	0.017
71.7	32	0	unsigned	150	920	19702	6461	0.023
71.7	32	9	unsigned	654	1753	88220	28932	0.147
84.2	32	0	unsigned	88	809	20698	6788	0.021
84.2	32	0	unsigned	125	1295	28138	9228	0.029
84.2	32	2	unsigned	250	1203	33899	11117	0.043
84.2	32	4	unsigned	340	1536	51401	16857	0.069
84.2	32	8	unsigned	500	2447	122920	40311	0.200
96.2	32	0	unsigned	70	1093	27223	8928	0.026
96.2	32	0	unsigned	120	1674	36647	12018	0.035
96.2	32	3	unsigned	265	1649	51832	16998	0.067
96.2	32	5	unsigned	340	2529	100579	32985	0.146
96.2	32	8	unsigned	508	3780	192141	63012	0.316

Table 6-4 - Synthesis results, 0.18 μm CMOS worst case library

Figure 6-28 presents a plot of area as a function of maximum clock rate for the four categories of SFDR. It can be seen that the inclusion of pipeline registers is very costly in terms of area, and, as shown by Figure 6-26, power consumption as well. Inspection of the 0.18 μm CMOS library shows that the flip-flops are among the larger cells of the library. Their effect on system complexity is more important for high SFDR designs because of their wider data paths and hence larger number of flip flops required per pipeline stage.

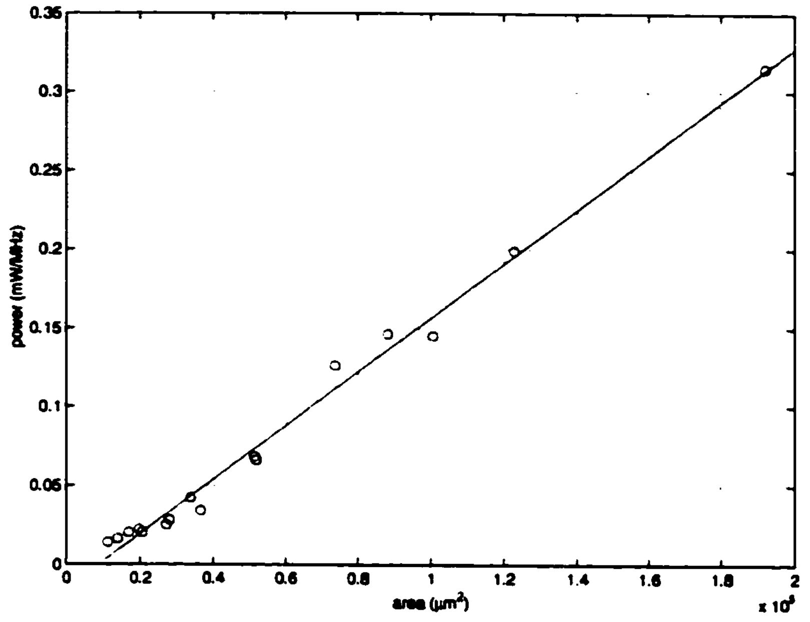

Figure 6-26 - Power vs area, 0.18 μm CMOS worst case library

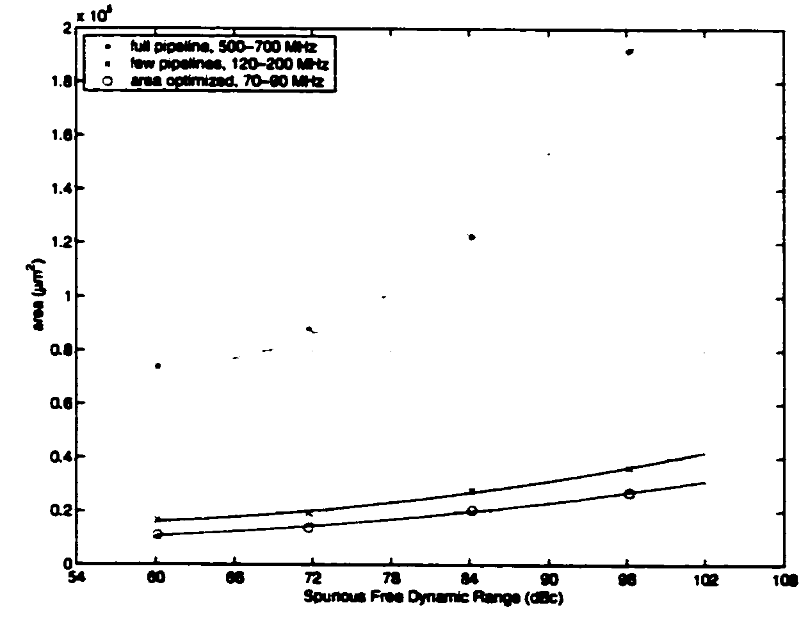

Figure 6-27 - Area vs SFDR, 0.18 μm CMOS worst case library

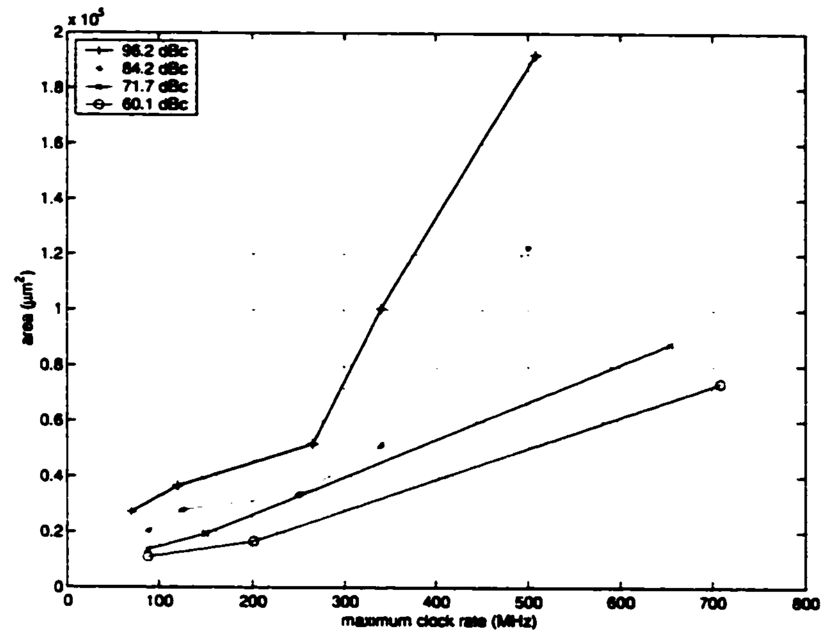

Figure 6-28 - Area vs clock rate, 0.18 μm CMOS worst case library

6.5.4 0.35 μm Implementation Results

A selected group of designs were synthesized for a 0.35 μm typical case library in order to provide a meaningful comparison of the new architecture with existing work. Table 6-5 gives design data and implementation results. The width of the phase accumulator *N* varies for each coefficient set, in order to match similar designs in previous work.

Figure 6-29 summarizes the data from Table 6-5. For greater emphasis, the fast 84.2 dBc designs are plotted with a different symbol.

The table includes an extra column showing whether regular ripple carry adders or a faster type of adder was specified for the Module Compiler™ tool. Excluding the 60.1 dBc design, it is noted that the linear relationship between area and power consumption does not hold as well as for the 0.18 μm design examples. For example, in the case of the two versions of the 84.2 dBc design with a clock rate of 100 MHz, the design with larger area actually consumes less power. Although it is very difficult to pinpoint the exact cause of this phenomenon, it is observed that the larger design includes one stage of pipelining. It has already been noted that in the libraries used, the flip-flops are fairly large cells, inflating the size of the design. However, the effective power consumption per area is less for these cells than for cells implementing logic functions. Hence, a design with no pipelining but with complex logic to implement a carry look-ahead adder may be smaller, yet it may consume more power. A similar observation occurs for the 84.2 dBc/320 MHz design, and for the 96.2 dBc designs. Figure 6-30 illustrates the relationship between power and area for these designs.

It is also observed from the 96.2 dBc design data that area and power consumption are increased when the output is converted to unsigned format.

SFDR (dBc)	N	latency (cycles)	output type	clock (MHz)	cells	area (μm^2)	transistors	power (mW/MHz)	adder type
60.1	20	0	S&M	107	277	35752	2724	0.089	ripple
60.1	20	0	S&M	108	317	38885	2963	0.098	fast
71.7	16	0	S&M	160	437	58608	4465	0.135	fast
71.7	16	0	S&M	160	381	59395	4525	0.115	ripple
84.2	28	0	S&M	100	703	79135	6029	0.175	fast
84.2	28	1	S&M	100	700	98840	7531	0.163	ripple
84.2	28	4	S&M	320	1374	282293	21508	0.545	fast
84.2	28	9	S&M	320	1542	368743	28095	0.516	ripple
96.2	32	0	S&M	100	1048	109918	8375	0.234	fast
96.2	32	1	S&M	100	1020	134400	10240	0.220	ripple
96.2	32	1	unsigned	100	1194	137060	10443	0.358	fast
96.2	32	1	unsigned	100	1110	143763	10953	0.239	ripple

Table 6-5 - Synthesis results, 0.35 μm typical case library

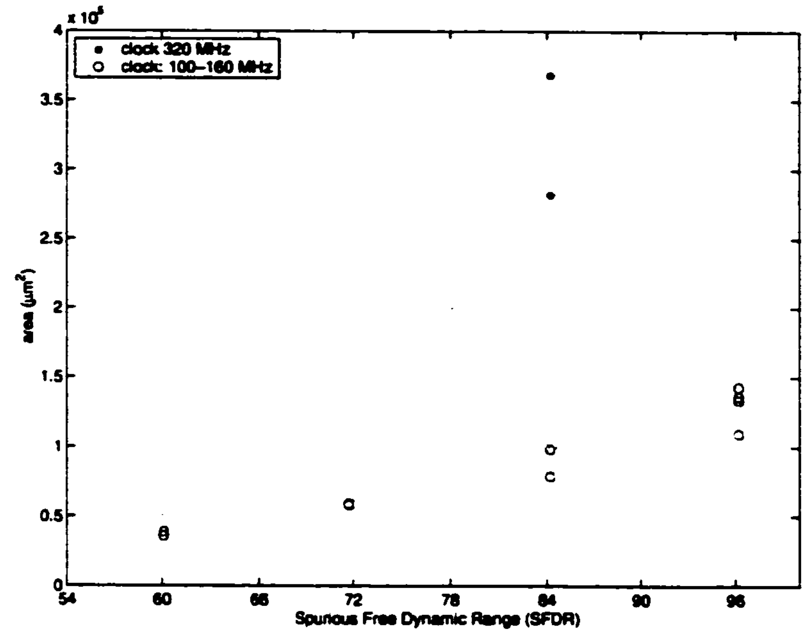

Figure 6-29 - Area vs SFDR, 0.35 μm CMOS typical case library

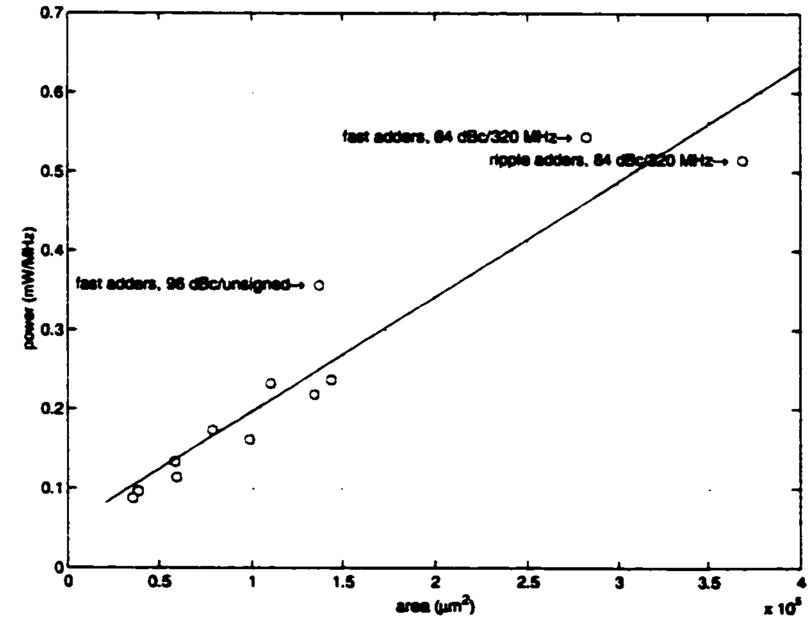

Figure 6-30 - Power vs area, 0.35 μm CMOS typical case library

6.6 Fabricated Designs

6.6.1 60 dBc Single Phase Design Description

A single-phase version of the 60 dBc design was fabricated in 0.18 µm CMOS from the Taiwan Semiconductor Manufacturing Company through CMC. A die picture is shown in Figure 6-31.

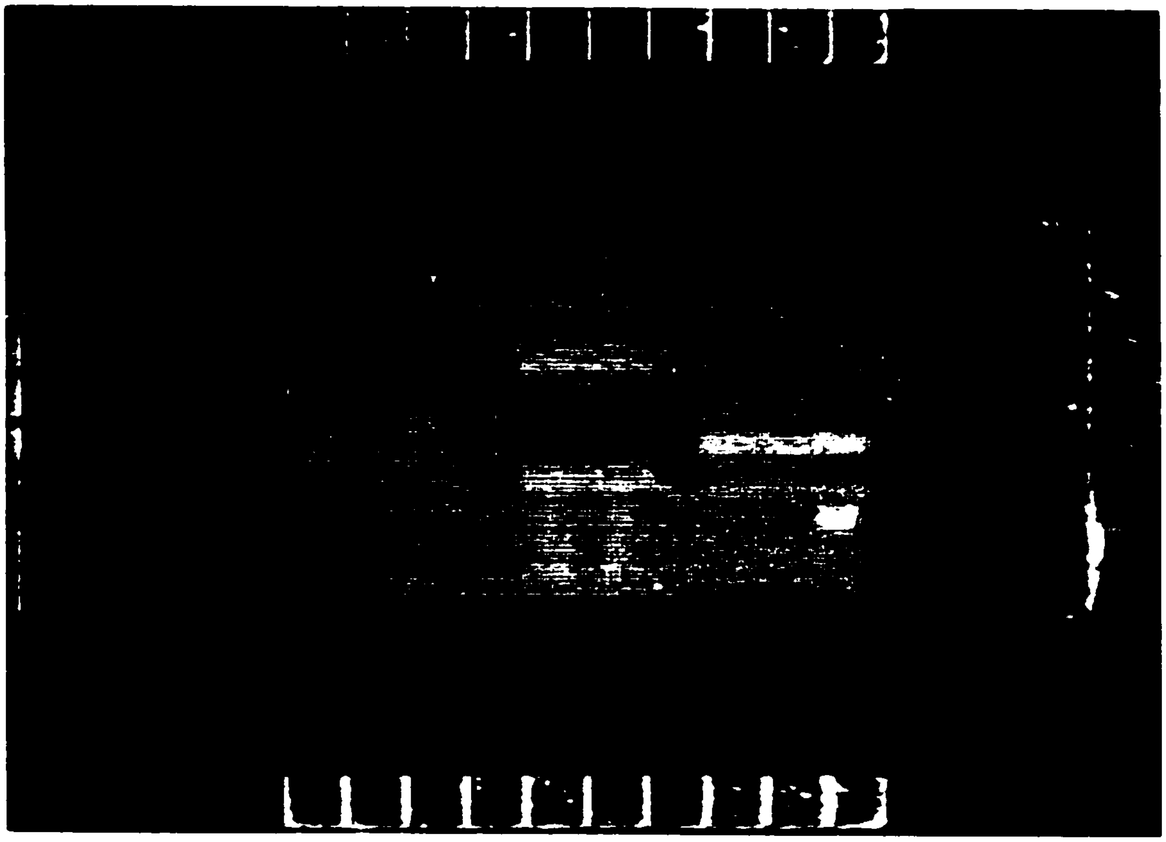

Figure 6-31 - Die picture, 60 dBc single phase DDFS in 0.18 µm CMOS

During placement and routing with automated tools, a clock constraint of 125 MHz was easily met without having to add pipelining registers. Core power consumption is estimated at 9.6 µW/MHz for a 1.8 V supply voltage, and 7.5 µW/MHz for a 1.6 V supply voltage. These figures were reported by the synthesis tool, assuming a load of 0.025 pF on the output registers (i.e. the load of one output I/O cell). The 8-bit output data is in unsigned format, compatible with most commercial DACs.

The FCW is 16 bits wide, yielding a frequency resolution of approximately 1526 Hz for a 100 MHz reference clock. As discussed in Section 2.3.1, however, the 60 dBc SFDR performance is guaranteed only for odd values of FCW. Hence, the "actual" frequency resolution for a 100 MHz clock is 3052 Hz. Although a wider FCW would have improved frequency resolution, it would also have increased the die area beyond that allowed under the fabrication grant. Serializing the FCW input would have alleviated that problem, but at an increased tuning latency. Since this design's critical path is in the phase accumulator, an increase in its width may have required the use of pipelining or fast adder architectures in order to maintain the 125 MHz clock rate target.

A total of 155 core cells are used and they occupy an area of 5773 μm^2. Since there was ample core area available, very loose placement was used and the total core area is approximately 11250 μm^2. The chip has 36 pins, including clock and reset signals, core power pins, I/O cell power pins, and ground pins. The core cells are rated at 1.8 V supply, and the I/O cells are rated at 3.3 V supply. The input I/O cells are 5 V tolerant. The final die dimensions, including I/O cells and pads, are 1321.6 $\mu m \times$ 1181.6 μm, for a total area approximately equal to 1.562 mm^2. As shown by Figure 6-31, this is a severely I/O bound chip.

6.6.2 Test Strategy and Procedure

Two dies were mounted in 40 pin DIP packages and tested with the IMS tester available in CMC's offices. Both chips were found to be functional, albeit with somewhat high leakage currents as detailed below. A basic functional test was first performed with a fixed FCW input of 64_{10}, a clock frequency of 1 MHz, and the rated 1.8 V core supply. This FCW results in the lowest output frequency whose period corresponds exactly to the period of the phase accumulator output, i.e. with no phase truncation effects. The sine output was collected for one period of data and compared with the expected sequence from the system's HDL description. Both chips passed this test.

The chips were then characterized with an FCW input sequence made of 20000 random values, uniformly distributed in the interval 0 to 65535_{10}. Again, the sine output data was collected for all these vectors and compared with the expected sequence from the system's HDL description. A "pass" was recorded when all 20000 output vectors were correct. In the absence of a standardized test sequence for DDF Synthesizers in the literature, changing the FCW on every clock cycle this way was found to be an acceptable approach. Although no application is foreseen where the FCW would change so frequently, it is expected that this test should provide pessimistic operating conditions for comparison with previous work.

The clock frequency and core supply voltage were varied for each chip to determine the range over which correct operation was maintained. Frequencies from 10 to 100 MHz and voltages from 1.2 to 1.8 V were considered. In the presence of a "pass" condition, the core power consumption was measured in each case. The setup time, hold time, output delay and data valid times were also measured.

Power consumption was also measured for fixed FCW operation for selected frequencies over the tuning range, and for varying FCW during frequency-hopping operation.

6.6.3 Test Results

Static power consumption was measured for different core supply voltages, and the results are presented in Table 6-6. Considering the small number of cells in this design, the leakage current for chip 1 is very high. It also appears high for chip 2. Simulations for a 1.8 V core supply at 25 °C indicate an expected leakage power of approximately 27 nW.

core supply (volts)	chip 1 (µwatts)	chip 2 (µwatts)
1.8	198	68
1.7	131	34
1.6	95	not measured
1.5	77	not measured
1.4	66	not measured
1.3	52	< 1
1.2	42	< 1

Table 6-6 - Static core power consumption

The timing characteristics of both chips are shown in Table 6-7. Empty table cells represent data that was not measured. This was due to a limitation of the test equipment at the higher frequencies, and, for chip 2, to its unfortunate destruction during testing. In the table, t_{su} indicates the setup time, and t_h the hold time, both for the input with respect to the clock. For the output, the measured timing characteristics were the propagation delay T_{delay}, which is the minimum time after a clock transition before the data was available on the output ports, and T_{valid}, the time during which the data remained valid and stable.

For a 1.8 V supply, the propagation delay was found to be approximately equal to 12.5 ns, restricting chip operation to approximately 80 MHz. However, operation was found to be unreliable, in general, for clock frequencies above 50 MHz. This is consistent with the expected performance of the chosen 40-pin DIP package.

clock (MHz)	**core supply (V)**	**FCW t_{su} (ns)**		**FCW t_h (ns)**		**sine T_{delay} (ns)**		**sine T_{valid} (ns)**	
		1	**2**	**1**	**2**	**1**	**2**	**1**	**2**
10	1.8	3.5	3.5	0.5	0.5	11.3	11.3	88.9	88.9
20	1.8	4.0	4.0	0.5	0.5	12.5	12.5	37.5	37.5
25	1.8	3.5		0.5		11.0		29.5	
50	1.8	5.5	5.5	8.7	8.7	12.0	12.0		
80	1.8	12.5				12.5			
10	1.2	6.0	5.0	0.5	4.5	42.5	34.5	40.0	66.0
20	1.2	6.0		0.5		37.0		12.0	
25	1.2	5.0		1.0		40.0		1.3	

Table 6-7 - Timing characteristics, chips 1 and 2

Net core power consumption is shown in Table 6-8 for both chips for the case of a random FCW input. "Net core power" means that the static power measured and reported in Table 6-6 was removed from the observed power consumption of the chip under the stated conditions. Some data is missing for chip 2, as it was inadvertently destroyed during testing. Figure 6-32 presents data from Table 6-8 corresponding to core supply voltages of 1.8 and 1.2 volts.

clock (MHz)	core supply (V)	chip 1 net core power (µW)	chip 2 net core power (µW)
10	1.8	194	196
20	1.8	475	470
25	1.8	612	not measured
50	1.8	1470	1488
80	1.8	1552	not measured
83	1.8	not functional	1659
87	1.8	not functional	1738
50	1.7	877	not measured
80	1.7	not functional	1385
20	1.3	291	288
50	1.3	not functional	469
10	1.2	122	105
20	1.2	243	150
25	1.2	302	not measured

Table 6-8 - Net core power consumption, random FCW

It is observed that, for a 1.8 V supply, the net core power varies linearly with the clock frequency, for frequencies below 50 MHz. After that point, however, the rate of change of power consumption decreases, which is contrary to standard power models. This may be attributed to reduced voltage swing of signals due to device bandwidth limitation. Furthermore, it was stated above that the rated operating frequency of the chosen package is 50 MHz, hence measurements above that limit are not reliable. The inconsistency may also be due to temperature effects that were not taken into account during testing. Finally, the high leakage currents of both chips for a supply voltage of 1.8 V cast doubts on their relative performances.

The difference between the estimated and measured power consumption for 1.8 V supply, ~10 μW/MHz versus 20 μW/MHz at 10 MHz and 30 μW/MHz at 50 MHz, is likely due to different assumptions on the dynamic characteristics of the input FCW data sequence.

The best performance for chip 1 was achieved for a 1.2 V supply, with 12.1 μW/MHz net power consumption. For chip 2, the best performance was achieved at 20 MHz and 1.2 V supply, with a net power consumption of only 7.5 μW/MHz. These results compare favorably with those of recent implemented designs in the same performance range, to wit: De Caro et al. [18] (190 μW/MHz, 60 dBc, 0.35 μm), and Bellaouar et al. [5] (320 μW/MHz, 64 dBc, 0.8 μm).

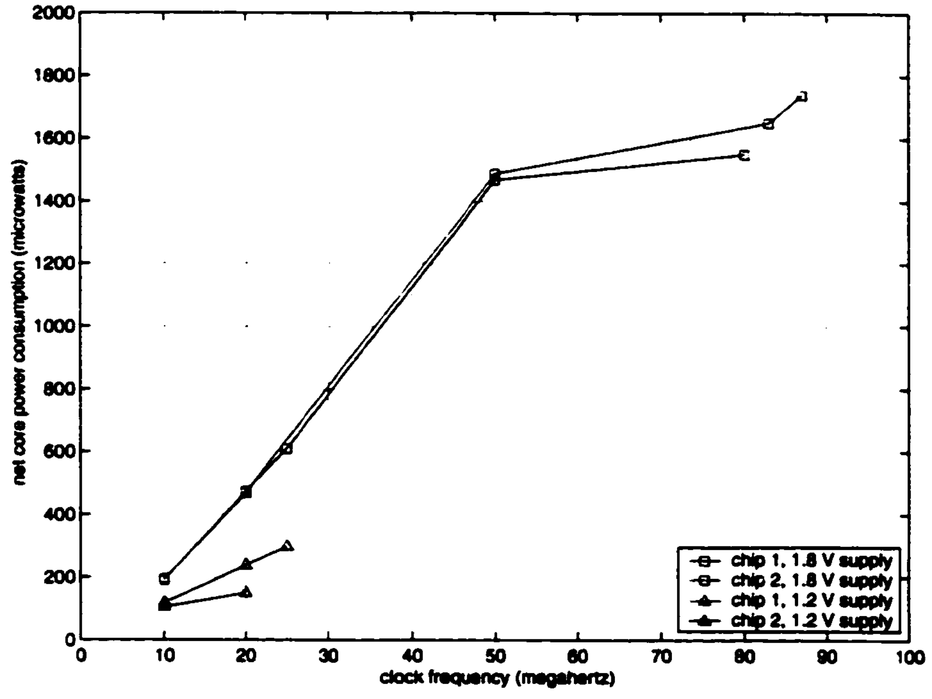

Figure 6-32 - Net core power consumption

Figure 6-33 plots net core power consumption as a function of FCW, for the case where the FCW is fixed. Again, these figures represent the net power after removal of the static power. For chip 1, the core supply voltage was 1.2 V and the clock frequency was 25 MHz. For chip 2, the figures were 1.3 V and 50 MHz, respectively.

Approximately two orders of magnitude separate these data with the case of random FCW input. Power consumption in chip 1 is approximately twice that of chip 2. Important variations are observed, even for very close FCW input values. They can be explained by the different toggle rates of internal registers corresponding to the different FCWs.

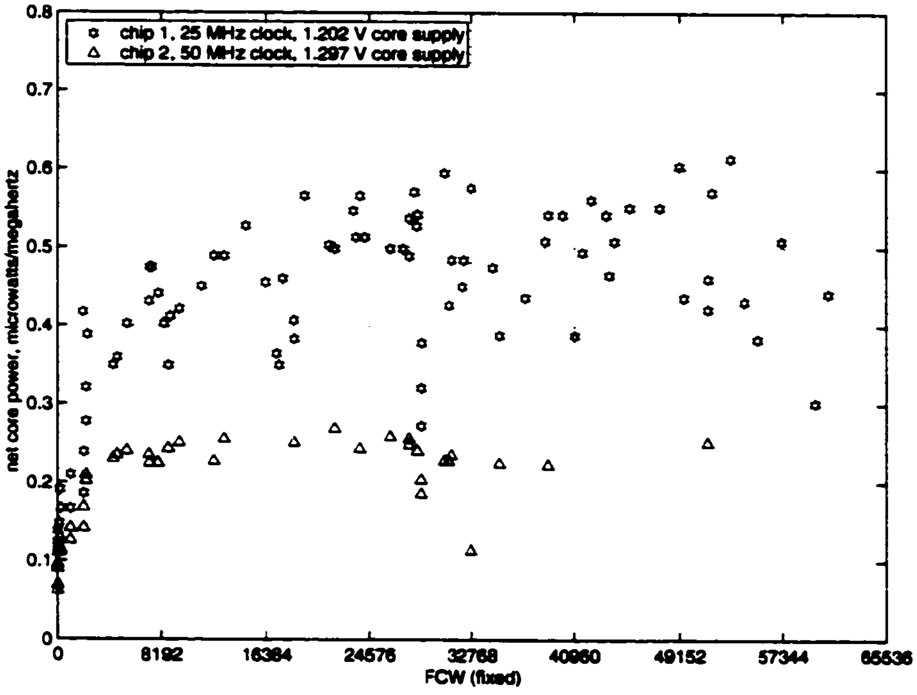

Figure 6-33 - Net core power consumption versus FCW input (fixed FCW)

Table 6-9 presents the I/O power consumption for random FCW, and Figure 6-34 plots the I/O power consumption for fixed FCW. In the fixed FCW case, the I/O power consumption is approximately three orders of magnitude greater than the core power. For the random FCW case, the difference is approximately two orders of magnitude. This fact and the obvious wasted silicon area in Figure 6-31 demonstrate that it is very inefficient to implement the new architecture as a single system in an advanced technology such as 0.18 μm CMOS. The small size and low power consumption of the core make it an ideal Intellectual Property (IP) building block for System On a Chip (SoC) applications.

I/O supply (volts)	chip 1 (milliwatts/MHz)	chip 2 (milliwatts/MHz)
3.3	2.43	2.26

Table 6-9 - I/O power consumption, random FCW

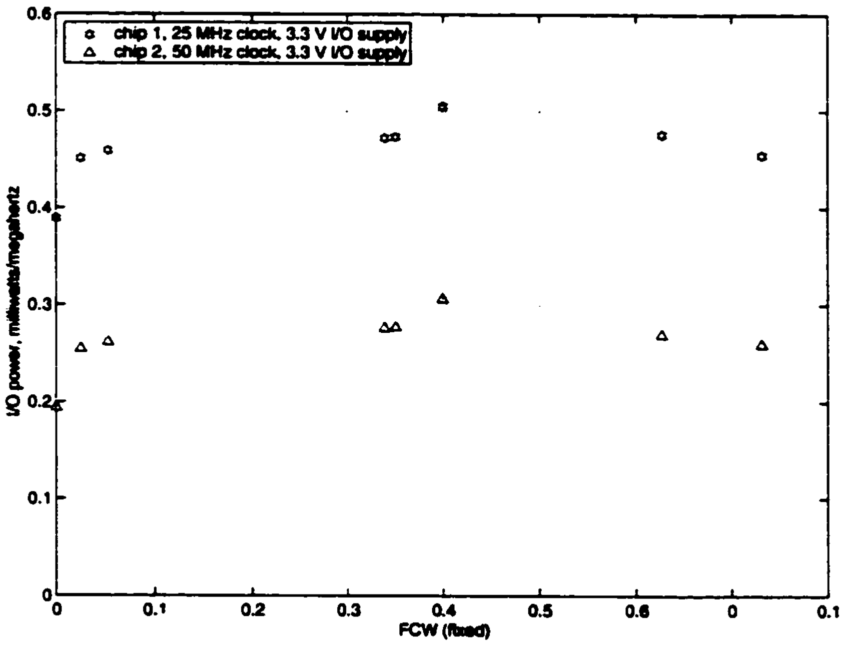

Figure 6-34 - I/O power consumption versus FCW input (fixed FCW)

Table 6-10 presents net core power consumption measurements for a frequency hopping condition with two FCWs. The core supply and clock frequency at which these measurements were made are indicated in the table. In each case, the FCW was maintained constant during 1000 clock cyles, then switched. For chip 1, it is observed that in two cases the power consumption is greater than for the random FCW test. For chip 2, the random test provided the worst case data. Still, it is likely that several combinations of FCWs during frequency hopping would result in power consumption figures higher and lower than those observed for the random FCW test.

FCW 1	FCW 2	chip 1 net core power @ 25 MHz, 1.2 V (μW/MHz)	chip 2 net core power @ 50 MHz, 1.3 V (μW/MHz)
59904	62720	9.2	6.6
1	26214	7.6	5.5
4660	65244	11.4	6.4
13107	30583	14.1	8.3
4369	30583	12.1	7.8
13107	38897	13.4	8.1

Table 6-10 - Power characteristics, two-FCW frequency hopping

6.6.4 84 dBc Design

A 84.2 dBc SFDR, 500 MHz DDFS was allotted fabrication area by CMC and was submitted for fabrication. Contrary to the data shown in Table 6-4, the typical case cell library data (e.g. 1.8 V, 25 °C) was used at all steps of the design process. The core contains 1234 cells occupying a total area of 47612 μm^2, and core power consumption is estimated at 88 μW/MHz with random FCW input and 0.025 pF load on the output registers. The estimated number of transistors is 16000. After placement and routing, the core area is approximately 90000 μm^2 and the die area is 3.8 mm^2. The chip has a total of 76 I/O pins, 18 of which are for power.

The quadrature outputs are in 12 bit unsigned format, and the FCW is 32 bits wide. The phase accumulator is 33 bits wide and its LSB is forced to oscillate between 0 and 1 every clock cycle. The effect is to force an odd value of FCW, thereby guaranteeing the 84.2 dBc performance for all output frequencies. The resulting frequency resolution is approximately 0.116 Hz for a 500 MHz clock.

The packaged chip is expected in October 2002.

Chapter 7

Interpretation and Discussion

7.1 Comparison with State of the Art

As discussed in Section 2.5, it is generally very difficult to compare two DDF Synthesizer designs because of the large number of performance parameters and design metrics involved. To make a proper comparison, two designs should achieve the same SFDR, have the same phase accumulator width, output type and output data format, achieve the same clock rate, and be implemented in the same technology. The relative merits of each design would then be measured by the core area, tuning latency and power consumption.

Table 6-5 presented data on several DDFS designs synthesized for a 0.35 μm CMOS library. The SFDR, phase accumulator width and clock rates were matched as much as possible to recent work published in the literature. The comparison data between the present 0.35 μm implementations and existing work is summarized in Table 7-1. The total number of ROM bits is included in this table since it is commonly reported in the literature. When reporting area, only the contribution of core cells is considered. Because it is impractical to compare the area of two designs implemented in different technologies, the table also includes the estimated number of transistors. This has been calculated by dividing the total core area by the area of a basic NAND gate and multiplying the quotient by four. Unless otherwise noted, table data is for quadrature output designs. Lastly, the table does not include designs with modulation capability.

design	SFDR (dBc)	clock (MHz)	N (bits)	ROM (bits)	transistors (#)	area (mm^2)	process (µm)	approach
[57] / 1995	100	100	36	0	58000	12.0	1.00	modified CORDIC
this work (64 segments)	**96.2**	**100**	**32**	**960**	**8400**	**0.110**	**0.35**	**optimized linear interpolation**
[16] / 2001	96.0	100	31	2304	14000	0.230	0.35	modified angular decomposition
[68] / 1991*	90.3	150	32	3072	35000	24.5	1.25	Nicholas architecture
[15] / 2001	84.3	100	28	832	7600	NA	0.35	modified angular decomposition
this work (32 segments, v. 1)	**84.2**	**100**	**28**	**448**	**6000**	**0.079**	**0.35**	**optimized linear interpolation**
this work (32 segments, v. 2)	**84.2**	**320**	**28**	**448**	**21500**	**0.282**	**0.35**	**optimized linear interpolation**
[75] / 2001	82.5	320	N/A	0	43600	0.660	0.35	5th degree Taylor series
[18] / 2002	80.0	88	N/A	0	NA **33000	0.440	0.35	optimized 3rd degree polynomial
[73] / 2000	78.0	160	N/A	0	39000	NA	0.35	angle rotation with feedback
this work (16 segments)	**71.7**	**160**	**16**	**160**	**4500**	**0.059**	**0.35**	**optimized linear interpolation**
[55] / 2001	67.6	80	16	0	11721	1.045	0.60	linear interpolation
[3] / 1999*	64.0	30	20	416	NA	0.900	0.80	1st degree Taylor series
[14] / 2000	64.0	80	20	304	4000	NA	0.35	modified angular decomposition
this work (8 segments)	**60.1**	**107**	**20**	**80**	**2800**	**0.035**	**0.35**	**optimized linear interpolation**
[18] / 2002	60.0	83	N/A	0	NA **13700	0.180	0.35	optimized 2nd degree polynomial

NA: data not available in the reference
* single phase output design;
** calculated estimate for 0.35 µm CMOS based on area: 76000 transistors per mm^2

Table 7-1 - Complexity and performance comparisons

Figure 7-1 summarizes the data in Table 7-1. It can be seen that the present architecture is superior to all previous work when using the number of transistors as a complexity metric. As can be seen from the table, the performance parameters have been fairly well matched. Comparing the number of transistors normalizes data that is greatly dependent on technology, such as area and power consumption, and hence it is felt that it is an appropriate comparison metric.

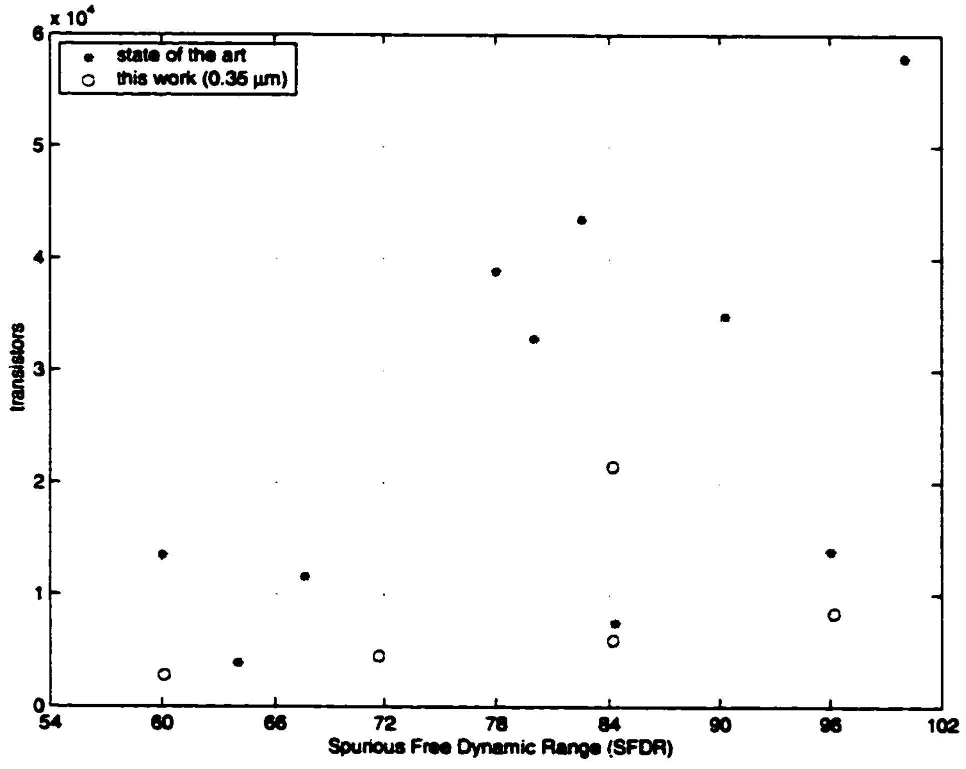

Figure 7-1 - Complexity and performance summary

The three designs by Curticăpean et al. [14][15][16] are the only ones whose performance is similar to this work's, yet the 96 dBc design requires 66% more transistors. The architecture on which they are based includes ROMs and multipliers. However, a detailed analysis and information disclosed in [17] reveals that the two architectures have several similarities. The multiplication operation performed in Curticăpean's architecture is optimized by limiting the operand width and by truncating several bits from the addition. Further, a localized search is performed to optimize parameter selection from initial search points established with trigonometric identities [17].

Care must be taken when considering the performance data for the designs by Madisetti et al. [57] and by Nicholas et al. [68], due to the design tradeoffs that were likely needed to achieve high clock rates in older technology. These clock rates were likely achieved with the help of several stages of pipelining and the use of fast cells, and hence these designs are not easily compared to the remainder of the designs in the table.

7.2 Complexity Increase with SFDR Performance

This section discusses the observed relationship between the complexity of the new architecture with the SFDR. The validity of the high level design metrics used during the design process is also compared to the observed complexity data obtained after implementation.

7.2.1 Quadratic Relationship

A quadratic relationship between system complexity and achieved SFDR was noted in the high-level complexity metrics (Figure 6-12) and in the implementations in FPGA (Figure 6-25), 0.18 μm (Figure 6-27) and 0.35 μm (Figure 6-29) CMOS. However, Figure 5-4 shows that the required number of linear segments, and hence the complexity in terms of 2:1 multiplexers, is doubled for each 12 dBc increase in output SFDR. One would therefore expect an exponential rather than a quadratic relationship between complexity and SFDR.

However, the cost in multiplexers is only a portion of overall system complexity. A significant contribution comes from the multi-operand adder. Inspection of Table 6-3 reveals that the progression in the width *D* of the multi-operand adder is fairly linear over the range of SFDRs considered. The same is true for parameters *M*, *L*, and *C*. Further, the number of addends increases only by one in the same range. Hence, the overall effect is not exponential, and the observed quadratic relation is expected to hold above 100 dBc SFDR.

7.2.2 High Level Metrics versus Implementation Data

The number of transistors estimated by the high level design metrics described in Section 6.2 and by the area method for 0.35 µm and 0.18 µm implementations can be compared. Selected data from Table 6-3, Table 6-4 and Table 6-5 is plotted in Figure 7-2 to highlight differences and similarities.

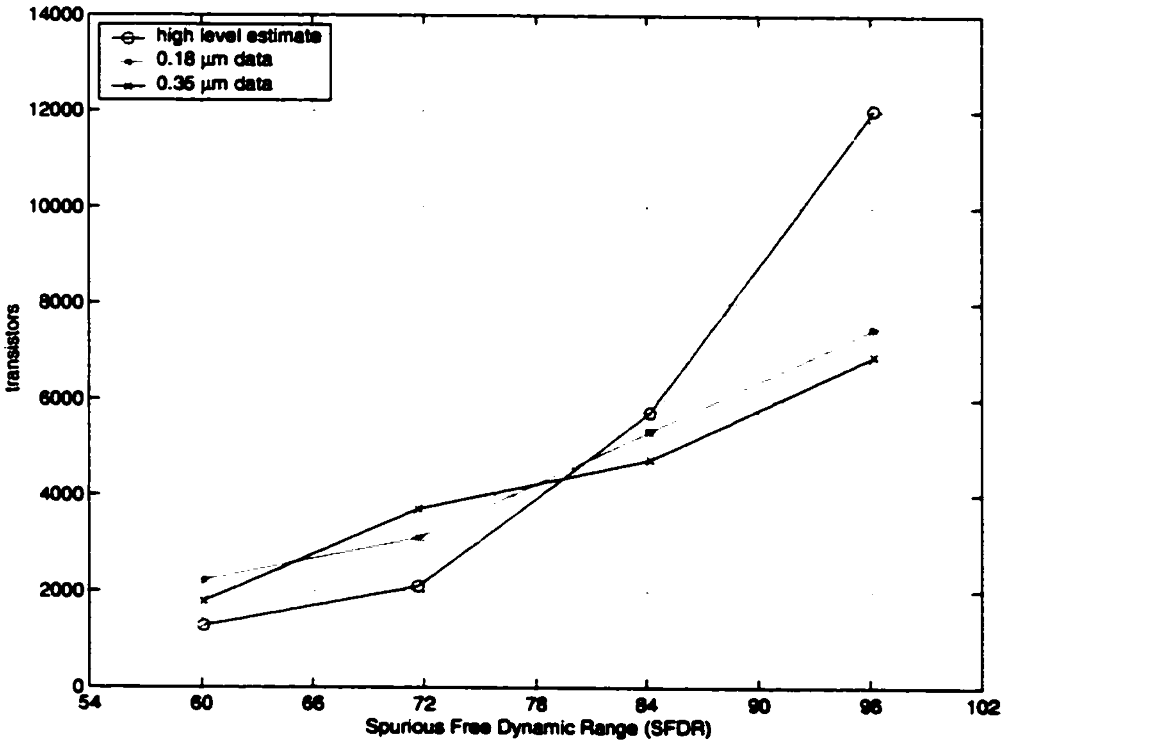

Figure 7-2 - Comparison of PSAC complexity estimates

The 0.18 µm and 0.35 µm implementation data comes from designs without pipelining and hence with lower maximum clock rates. Complexity data on designs with higher clock rates is not included, since the high-level complexity metrics do not include the contribution from pipelining registers or the use of fast cells. The number of transistors shown for these two technologies was adjusted to take into account the contribution of the phase accumulator. It was assumed that, on average, the phase accumulator contributes 45.3 transistors per bit. This average was calculated by adding the area occupied by one full adder and one D flip-flop, with respect to a basic two input NAND gate. Removing the contribution from the phase accumulator provides a better comparison with the high level metrics, which do not make provision for it. The goal of the high design metrics is to allow the comparison of various choices for the implementation of the PSAC, and not the remainder of the system.

The high-level estimate of the number of transistors appears to be too low at low SFDR and too high at high SFDR. At low SFDR, the difference can be explained by the assumption of single phase output in the case of the high-level metrics, and by the fact that neither quadrant symmetry circuitry nor output data conversion are considered. For high SFDR, the larger designs lend themselves well to optimization by the synthesis tool, exploiting redundancy in the multiplexer input ports and using efficient multiple-inputs cells. Multiplexer optimization is obviously not captured well by the high-level estimates. The optimization of the multi-operand adders is also not included in the high level metrics. As shown in 6.1.2, several bits from some operand may be zeros, simplifying the adder's implementation.

The 'hump' in 0.35 μm data at 72 dBc is explained by the fact that this design contains fast cells in order to achieve a 160 MHz clock rate.

Overall, Figure 7-2 shows that the high level complexity metrics provide an accurate method of comparing the relative complexity of coefficient sets and design parameters early in the design cycle.

7.2.3 0.35 μm vs 0.18 μm Data

Figure 7-2 shows a very close match in the number of required transistors to implement designs in either 0.35 μm and 0.18 μm technologies. The constant difference between the two can be explained by the fact that the 0.18 μm data is for a worst case library, while the 0.35 μm data is for a typical case library.

Chapter 8

Conclusion

8.1 Summary of the Contribution

This thesis has presented a new approach to the design of DDF Synthesizers. The approach is based on the approximation of a sinusoid with piecewise continuous linear segments whose slopes and initial amplitudes are selected to optimize the output SFDR. The selection of the segment slopes is constrained to keep hardware implementation costs low by eliminating the requirement for multipliers. A theoretical basis to the new approach was given, relating the maximum SFDR that could be attained and the number of linear segments used. A systematic design procedure was described. New DDFS architectures were introduced. Several design examples were given, showing that the maximum performance bound could be achieved in spite of restrictions imposed on the selection system parameters. The new approach and new architectures were validated with several implemented designs of varying complexities and performance characteristics, comparing favorably with existing work.

8.2 Completion of Research Objectives

The research objectives initially set out where fully met.

An innovative sinusoid generation technique based on a hardware-optimized linear interpolation technique was devised for use in Direct Digital Frequency Synthesizers. It was applied to the development of novel DDFS architectures achieving various levels of SFDR. The new technique reduces system complexity significantly, when compared with prior work, while maintaining superior output spectral characteristics, high data rates, and low tuning latency.

A proof relating the achievable SFDR to the number of linear segments used in the interpolation was presented. A major impact of this proof is that it validates the hardware optimizations made in the proposed architectures.

A systematic design procedure was presented and applied successfully.

High level design metrics applicable to the new technique were defined in order to facilitate the design space exploration early into the design cycle. These were applied successfully for the selection of the sample designs. A very good match to actual implementation data was observed.

An automated synthesis algorithm was developed, implemented as a Matlab™ script and used successfully.

Several DDFS designs achieving SFDRs in the range of 60 to 96 dBc were produced, simulated and implemented in three technologies: FPGA, 0.18 μm CMOS and 0.35 μm CMOS. Performance parameters and complexity metrics were extracted for each design. Two designs were submitted for fabrication following two successful fabrication grant requests.

Finally, the implementation data was used to compare the present work to the state of the art, and to verify the validity of the new approach.

8.3 Recommendations for Future Work

The work presented in this thesis could be extended in several ways, offering promising new research projects.

The linear segmentation approach analyzed in Chapter 5 was shown to be a special case of polynomial segmentation, described in general by Equation (3-12). The relationship found between the SFDR and the number of linear segments should be extended to polynomials of degree 2 and 3, and possibly to polynomials of arbitrary degree. Similarly, the design procedure should also be extended to high degree polynomials. This would lead to new corresponding architectures. Issues concerning the most efficient implementation of integer power functions would have to be resolved. Architectures with multipliers and architecture without multipliers would have to be investigated. It may still be possible to quantize multiplier coefficients with few signed digits. Design complexity metrics would have to be developed for each new architecture to assess the relative complexities of coefficient sets and system parameters during the design process.

A fundamental requirement in communications systems is the mixing process. It normally involves a complex multiplication, which is a costly operation in terms of hardware. It may be possible to design the DDF Synthesizer with this operation in mind, i.e. to reduce the overall system complexity of the synthesizer/mixer. One candidate approach would involve an alternative number representation system, such as the Logarithmic or Residue Number Systems (LNS and RNS). In the logarithmic case, the DDFS output may no longer be a sequence representing the sine function, but the logarithm of the sine function instead.

Although the new architecture that was presented does not include multipliers per se, it is suspected that at very high SFDR, e.g. greater than 120 dBc, the minimum number of signed digits required to represent linear segment slopes will be so great that it may be more efficient to use the traditional combination of a ROM and a multiplier. The breakeven point, if any, should be determined. If it exists, then issues concerning efficient multiplier implementation using rounding and/or error correction would have to be investigated.

The hardware implemented designs discussed in Chapter 6 were synthesized for general-purpose libraries of cells. It is believed that the new architectures would be well adapted to multiplexer-friendly techniques such as pass-transistor logic. Although this technique suffers from V_T drop, it is very efficient in terms of number of transistors necessary to implement multiplexers and in terms of power consumption. At the very least, the multiplexers described in the new architecture could be implemented from special pass-transistor logic cells to observe the effect on overall system complexity and performance.

This work has shown that approximate calculations can be "good enough" for data processing systems working with or producing analog signals. Thus, the linear interpolation scheme described herein may be applicable to areas not related to frequency synthesis. These include fast approximate FFT calculations and data compression.

In summary, it is believed that several chapters remain to be written on the topic of Direct Digital Frequency Synthesizer architectures.

References

[1] E. Adler, J. Clark, M. Conn, P. Phuong, and B. Scheiner, "Low-cost technology for multimode radar," *IEEE Aerospace and Electronics Systems Magazine*, vol. 14, June 1999, pp. 23-27.

[2] V. Andrews, C.T.M. Chang, J.D. Cayo, S. Sabin, W.A White, and M.P. Harris, "A monolithic digital chirp synthesizer chip with I and Q channels," *IEEE Journal of Solid State Circuits*, vol. 27, no. 10, October 1992, pp. 1321-1326.

[3] A. Bellaouar, M.S. O'brecht, A.M. Fahim, and M.I. Elmasry, "Low-power direct digital frequency synthesis for wireless communications," in *Proceedings of the IEEE 1999 Custom Integrated Circuits Conference*, 1999, pp. 593-596.

[4] A. Bellaouar, M.S. O'brecht, and M.I. Elmasry, "Low-power direct digital frequency synthesizer architecture," U.S. Patent 5 999 581, 7 December, 1999.

[5] A. Bellaouar, M.S. O'brecht, A.M. Fahim, and M.I. Elmasry, "Low-power direct digital frequency synthesis for wireless communications," *IEEE Journal of Solid State Circuits*, vol. 35, no. 3, March 2000, pp. 385-390.

[6] B.E. Bjerede, "Suppression of spurious frequency components in direct digital frequency synthesizer," U.S. Patent 5 073 869, 17 December, 1991.

[7] A.L. Bramble, "Direct digital frequency synthesis," in *Proceedings of the 35th Annual Frequency Control Symposium*, 1981, pp. 406-414.

[8] D.E. Calbaza and Y. Savaria, "Direct digital frequency synthesis of low-jitter clocks," *IEEE Journal of Solid State Circuits*, vol. 36, no. 3, March 2001, pp. 570-572.

[9] L. Callegaro and V. D'Elia, "A synchronized two-phase sinewave generator for AC metrology system compensations," *IEEE Transactions on Instrumentation and Measurement*, vol. 49, no. 2, April 2000, pp. 320-324.

[10] F. Cardells-Tormo and J. Valls-Coquillat, "Optimization of direct digital frequency synthesisers based on CORDIC," *Electronic Letters*, 11 October 2001, Vol. 37, No. 21, pp. 1278-1280.

[11] G. Chang, A. Rofougaran, M. Ku, A.A. Abidi, and H. Samueli, "A low-power CMOS digitally synthesized 0-13 MHz agile sinewave generator," in *IEEE International Solid State Circuits Conference Digest of Technical Papers*, May 1994, pp. 32-33.

[12] S. Ciglaric, D. Fefer, and A. Jeglic, "Special Considerations for Alternatively Designed Digital Phase Angle Standard," *IEEE Transactions on Instrumentation and Measurement*, vol. 47, no. 1, February 1998, pp. 199-203.

[13] M.P. Craig, "Design and VLSI implementation of the quadrature conversion, filtering, and decimation of narrow band signals," Master's of Engineering thesis, Royal Military College of Canada, 1992.

[14] F. Curticăpean and J. Niittylahti, "Low power direct digital frequency synthesizer," in *Proceedings of the 43rd IEEE Midwest Symposium on Circuits and Systems*, Lansing MI, 9-11 August 2000, pp. 822-825.

[15] F. Curticăpean and J. Niittylahti, "A hardware efficient direct digital frequency synthesizer," in *Proceedings of the IEEE International Conference on Electronics, Circuits and Systems*, Malta, 2-5 September 2001, pp. 51-54.

[16] F. Curticăpean, K.I. Palomäki and J. Niittylahti, "Direct digital frequency synthesiser with high memory compression ratio," *Electronic Letters*, 11 October 2001, Vol. 37, No. 21, pp. 1275-1277.

[17] F. Curticăpean, private communications, Winter and Spring 2002.

[18] D. De Caro, E. Napoli and A.G.M. Strollo, "Direct digital frequency synthesizers using high-order polynomial approximation," in *IEEE International Solid State Circuits Conference, Digest of Technical Papers*, San Diego, 3-7 February 2002, pp. 134-135.

[19] A. Edwin, "Direct-digital synthesis applications," *Microwave Journal*, vol. 33, no. 1, January 1990, pp. 149-151.

[20] L. Fanucci, R. Roncella and R. Saletti, "A sine wave digital synthesizer based on a quadratic approximation," in *Proceedings of the IEEE International Frequency Control Symposium*, 2001, pp. 806-810.

[21] M.J. Flanagan and G.A. Zimmerman, "Spur-reduced digital sinusoid synthesis," *IEEE Transactions on Communications*, vol. 43, no. 7, July 1995, pp. 2254-2262.

[22] R.A. Freeman, "Digital sine conversion circuit for use in direct digital synthesizers," U.S. Patent 4,809,205, 28 February 1989.

[23] J.F. Garvey and D. Babitch, "An exact spectral analysis of a number controlled oscillator based synthesizer," in *Proceedings of the 44th Annual Frequency Control Symposium*, 1990, pp. 511-521.

[24] R. Giannini, S. Tonkovic, and V. Kozina, "A DSP based method for audio stimuli generation," in *Proceedings of the 19th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 30 October 30 – 2 November 1997, vol. 5, pp. 1974-1977.

[25] D.E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, 1989.

[26] B.G. Goldberg, *Digital Frequency Synthesis Demystified*, LLH Technology Publishing, 1999.

[27] E. Grayver and B. Daneshrad, "Direct digital frequency synthesis using a modified CORDIC," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, 1998, pp. 241-244.

[28] R. Hassun and A.W. Kovalick, "Waveform synthesis using multiplexed parallel synthesizers," U.S. Patent 4 454 486, 12 June, 1984.

[29] B.H. Hutchinson, Jr., "Contemporary frequency synthesis techniques," *Frequency Synthesis: Techniques and Applications*, J. Gorski-Popiel, Ed., New York: IEEE Press, 1975, pp. 25-45.

[30] I. Janiszewski, B. Hoppe and H. Meuth, "Precision and performance of numerically controlled oscillators with hybrid function generators," in *Proceedings of the IEEE International Frequency Control Symposium*, Seattle, Washington, 6-8 June, 2001.

[31] I. Janiszewski, B. Hoppe and H. Meuth, "VHDL-based design and design methodology for reusable high performance direct digital frequency synthesizers," in *Proceedings of the Design Automation Conference*, Las Vegas, Nevada, 18-22 June, 2001.

[32] S.C. Jasper, "Frequency resolution in a digital oscillator," U.S. Patent 4 652 832, 24 March, 1987.

[33] J. Jiang and E.K.F. Lee, "A ROM-less direct digital frequency synthesizer using segmented nonlinear digital-to-analog converter," in *Proceedings of the IEEE Custom Integrated Circuits Conference*, San Diego, CA, May 6-9, 2001, pp. 165-168.

[34] J. Jiang and E.K.F. Lee, "Segmented sine wave digital-to-analog converters for frequency synthesizer," in *Proceedings of the IEEE Symposium on Circuits and Systems*, Sydney, Australia, May 6-9, 2001, pp. 520-523.

[35] M. Kanazawa et al., "Beam test of the acceleration system with the DDS in HIMAC synchrotron," in *Proceedings of the Particle Accelerator Conference*, 12-16 May 1997, vol. 2, pp. 2347 - 2349.

[36] G.W. Kent and N.-H. Sheng, "A high purity, high speed direct digital synthesizer," in *Proceedings of the IEEE International Frequency Control Symposium*, pp. 207-211, 1993.

[37] R.J. Kerr and L.A. Weaver, "Pseudorandom dither for frequency synthesis noise," U.S. Patent 4 901 265, 13 February 1990.

[38] E.C. Kisenwether and W.C. Troxell, "Performance analysis of the numerically controlled oscillator," in *Proceedings of the 40th Annual Symposium on Frequency Control*, 1986, pp. 373-378.

[39] E. Kreyszig, *Advanced Engineering Mathematics*, 6th Ed., John Wiley & Sons, 1988, Section 10.6.

[40] V.F. Kroupa, "Spectral purity of direct digital frequency synthesizers," in *Proceedings of the 44th Annual Symposium on Frequency Control*, 1990, pp. 498-510.

[41] V.F. Kroupa, "Discrete spurious signals and background noise in direct digital frequency synthesizers," in *Proceedings of the IEEE International Frequency Control Symposium*, 1993, pp. 242-249.

[42] V.F. Kroupa, "Spectral properties of DDFS: computer simulations and experimental verifications," in *Proceedings of the IEEE International Frequency Control Symposium*, 1994, pp. 613-623.

[43] V.F. Kroupa, "Phase and amplitude disturbances in direct digital frequency synthesizers," *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 46, no. 3, May 1999, pp. 481-486.

[44] V.F. Kroupa, Ed., *Direct Digital Frequency Synthesizers*, IEEE Press, 1999.

[45] V.F. Kroupa, V. Cizek, J. Stursa, and H. Svandova, "Spurious signals in direct digital frequency synthesizers due to the phase truncation," *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 47, no. 5, September 2000, pp. 1166-1172.

[46] J.M.P. Langlois, "Design and implementation of wide band quadrature demodulators on Field Programmable Gate Arrays," Master's of Engineering thesis, Royal Military College of Canada, May 1999.

[47] J.M.P. Langlois and D. Al-Khalili, "Low complexity direct digital frequency synthesizer with high spectral purity," U.S. Patent Provisional Application, August 2001.

[48] J.M.P. Langlois and D. Al-Khalili, "ROM size reduction with low processing cost for direct digital frequency synthesis," in *Proceedings of the IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, Victoria, BC, August 2001, pp. 287-290.

[49] J.M.P. Langlois and D. Al-Khalili, "Efficient sine amplitude computation for direct digital frequency synthesis," in *Proceedings of the IEEE International Symposium on Signal Processing and Information Technology*, Cairo, Egypt, December 2001.

[50] J.M.P. Langlois and D. Al-Khalili, "A low power direct digital frequency synthesizer with 60 dBc spectral purity," in *Proceedings of the ACM Great Lakes Symposium on VLSI*, New York, NY, April 2002, pp. 166-171.

[51] J.M.P. Langlois and D. Al-Khalili, "Hardware optimized direct digital frequency synthesizer architecture with 60 dBc spectral purity," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, Phoenix AZ, May 2002, pp. 361-364.

[52] J.M.P. Langlois and D. Al-Khalili, "A new approach to the design of low power direct digital frequency synthesizers," in *Proceedings of the IEEE International Frequency Control Symposium*, New Orleans, Louisiana, May 2002.

[53] J.M.P. Langlois and D. Al-Khalili, "A quadrature direct digital frequency synthesizer architecture using piecewise-continuous linear segments," in *Proceedings of the Queen's Biennial Symposium on Communications*, Kingston, Ontario, June 2002, pp. 463-467.

[54] J.C.H. Latour, "High-speed complex multiplier integrated circuit with emphasis on low power design," Master's of Engineering thesis, Royal Military College of Canada, 1995.

[55] S.-I. Liu, T.-B. Yu and H.-W. Tsao, "Pipeline direct digital frequency synthesiser using decomposition method," *IEE Proceedings on Circuits, Devices and Systems*, vol. 148, no. 3, June 2001, pp. 141-144.

[56] H.-C. Liu, J.S. Min, and H. Samueli, "A low-power baseband receiver IC for frequency-hopped spread spectrum communications," *IEEE Journal of Solid State Circuits*, vol. 31, no. 3, March 1996, pp. 384-394.

[57] A. Madisetti, A. Kwentus, and A.N. Willson Jr., "A sine/cosine direct digital frequency synthesizer using an angle rotation algorithm," in *IEEE International Solid State Circuits Conference Digest of Technical Papers*, 15-17 February 1995, pp. 262-263.

[58] A. Madisetti and A.Y. Kwentus, "Method and apparatus for direct digital frequency synthesizer," U.S. Patent 5 737 253, 7 April 1998.

[59] A. Madisetti, A.Y. Kwentus, and A.N. Willson, Jr., "A 100-MHz, 16-b, direct digital frequency synthesizer with a 100-dBc spurious-free dynamic range," *IEEE Journal of Solid State Circuits*, vol. 34, no. 8, August 1999, pp. 1034-1043.

[60] V. Manassewitsch, *Frequency Synthesizers, Theory and Design*, 2nd ed., New York: Wiley, 1989.

[61] E.M. Mattison and L.M. Coyle, "Phase noise in direct digital synthesizers," in *Proceedings of the 42nd Annual Frequency Control Symposium*, 1988, pp. 352-356.

[62] S. Mehrgardt, "Noise spectra of digital sine-generators using the table-lookup method," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 31, August 1983, pp. 1037-1039.

[63] S. Mortezapour and E.K.F. Lee, "Design of low-power ROM-less direct digital frequency synthesizer using nonlinear digital-to-analog converter," *IEEE Journal of Solid State Circuits*, vol. 34, no. 10, October 1999, pp. 1350-1359.

[64] T. Nakagawa and H. Nosaka, "A direct digital synthesizer with interpolation circuits," *IEEE Journal of Solid State Circuits*, vol. 32, May 1997, pp. 766-770.

[65] J.A. Nelder and R. Mead, "A simplex method for function minimization," *Computer Journal*, Vol. 7, 1965, pp. 308-313.

[66] H.T. Nicholas III and H. Samueli, "An analysis of the output spectrum of direct digital frequency synthesizers in the presence of phase-accumulator truncation," in *Proceedings of the 41st Annual Symposium on Frequency Control*, 1987, pp. 495-502.

[67] H.T. Nicholas III, H. Samueli and B. Kim, "The optimization of direct digital frequency synthesizer performance in the presence of finite word length effects," in *Proceedings of the 42nd Annual Symposium on Frequency Control*, 1988, pp. 357-363.

[68] H.T. Nicholas III and H. Samueli, "A 150-MHz direct digital frequency synthesizer in 1.25 μm CMOS with -90 dBc spurious performance," in *Proceedings of the IEEE International Solid State Circuits Conference*, 1991.

[69] H.T. Nicholas III and H. Samueli, "A 150-MHz direct digital frequency synthesizer in 1.25 μm CMOS with -90 dBc spurious performance," *IEEE Journal of Solid State Circuits*, vol. 26, no. 12, December 1991, pp. 1959-1969.

[70] J. Nieznanski, "An alternative approach to the ROM-less direct digital synthesis," *IEEE Journal of Solid State Circuits*, vol. 33, no. 1, January 1998, pp. 169-170.

[71] H. Nosaka, T. Nakagawa, and A. Yamagishi, "A phase interpolation direct digital synthesizer with a digitally controlled delay generator," in *Symposium on VLSI Circuits Digest of Technical Papers*, 1997, pp. 75-76.

[72] P. O'Leary and F. Maloberti, "A direct-digital synthesizer with improved spectral performance," *IEEE Transactions on Communications*, vol. 39, no. 7, July 1991, pp. 1046-1048.

[73] K.I. Palomäki, J. Niitylahti and Lehtinen, "A pipelined digital frequency synthesizer based on feedback," in *Proceedings of the 43rd IEEE Midwest Symposium on Circuits and Systems*, Lansing MI, 9-11 August 2000.

[74] K.I. Palomäki and J. Niitylahti, "Direct digital frequency synthesizer architecture based on Chebyshev approximation," in *Proceedings of the 34th Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, CA, 29 October – 1 November, 2000, pp. 1639-1643.

[75] K.I. Palomäki and J. Niittylahti, "A quadrature phase-to-amplitude converter based on Taylor series approximation," in *Proceedings of the 3rd International Conference on Information, Communications and Signal Processing*, Singapore, 15-18 October 2001.

[76] V. Reinhardt, K. Gould, K. McNab, and M. Bustamante, "A short survey of frequency synthesizer techniques," in *Proceedings of the 40th Annual Symposium on Frequency Control*, 1986, pp. 355-365.

[77] V.S. Reinhardt, K.V. Gould, and K.M. McNab, "Randomized digital/analog converter direct digital synthesizer," U.S. Patent 5 014 231, 7 May, 1991.

[78] A. Rofougaran et al., "A single-chip 900-MHz spread-spectrum wireless transceiver in 1-μm CMOS - Part I: Architecture and transmitter design," *IEEE Journal of Solid State Circuits*, vol. 33, no. 4, April 1998, pp. 515-534.

[79] A. Rofougaran et al., "A single-chip 900-MHz spread-spectrum wireless transceiver in 1-μm CMOS - Part II: Receiver design," *IEEE Journal of Solid State Circuits*, vol. 33, no. 4, April 1998, pp. 535-547.

[80] P.H. Saul and M.S.J. Mudd, "A direct digital synthesizer with 100-MHz output capability," *IEEE Journal of Solid State Circuits*, vol. 23, no. 3, June 1988, pp. 819-821.

[81] P.H. Saul and D.G. Taylor, "A high-speed digital frequency synthesizer," *IEEE Journal of Solid State Circuits*, vol. 25, no. 1, February 1990, pp. 215-220.

[82] S.S. Shah and S. Colling, "A 200 MHz analogue-ROM based direct digital frequency synthesizer with amplitude modulation," in *Proceedings of the IEEE International Symposium on VLSI Technology, Systems and Applications*, Hsinchu, Taiwan, April 18-20, 2001, pp. 53-56.

[83] A.M. Sodagar and G.R. Lahiji, "Parabolic approximation: a new method for phase-to-amplitude conversion in sine-output direct digital frequency synthesizers" in *Proceedings of the International Symposium on Circuits and Systems*, Switzerland, May 2000, pp. 515-518.

[84] A.M. Sodagar and G.R. Lahiji, "A novel architecture for ROM-less sine-output direct digital frequency synthesizers by using the 2nd-order parabolic approximation," in *Proceedings of the IEEE/IEA International Frequency Control Symposium*, Kansas City, Missouri, 7-9 June 2000, pp. 284-289.

[85] A.M. Sodagar and G.R. Lahiji, "Mapping from phase to sine-amplitude in direct digital frequency synthesizers using parabolic approximation," *IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing*, vol. 47, no. 12, December 2000, pp. 1452-1457.

[86] A.M. Sodagar and G.R. Lahiji, "A pipelined ROM-less architecture for sine-output direct digital frequency synthesizers using the second-order parabolic approximation," *IEEE Transactions on Circuits and Systems - II: Analog and Digital Signal Processing*, vol. 48, no. 9, September 2001, pp. 850-857.

[87] D.A. Sunderland, R.A. Strauch, S.S. Wharfield, H.T. Peterson and C.R. Cole, "CMOS/SOS frequency synthesizer LSI circuit for spread spectrum communications," *IEEE Journal of Solid State Circuits*, vol. 19, August 1984, pp. 497-505.

[88] Synopsys Inc., Module Compiler User Guide, version 2000.05, May 2000.

[89] H. Szu, R. Stapleton, and F. Willwerth, "Digital radar commercial applications," in *Proceedings of the IEEE International Radar Conference, Arlington VA*, May 7-12 2000, pp. 717-722.

[90] L.K. Tan and H. Samueli, "A 200 MHz quadrature digital synthesizer/mixer in 0.8 μm CMOS," *IEEE Journal of Solid State Circuits*, vol. 30, no. 3, March 1995, pp. 193-200.

[91] L.K. Tan, E.W. Roth, G.E. 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, December 1995, pp. 1463-1473.

[92] J. Tierney, C.M. Rader and B. Gold, "A digital frequency synthesizer," *IEEE Transactions on Audio and Electroacoustics*, vol. AU-19, 1971, pp. 48-57.

[93] J. Tierney, "Digital frequency synthesizers," *Frequency Synthesis: Techniques and Applications*, J. Gorski-Popiel, Ed., New York: IEEE Press, 1975, pp. 121-149.

[94] A. Torosyan and A.N. Willson, Jr., "Analysis of the output spectrum for direct digital frequency synthesizers in the presence of phase truncation and finite arithmetic precision," in *Proceedings of the 2nd International Symposium on Image and Signal Processing and Analysis*, pp. 458-463.

[95] A. Torosyan, D. Fu and A.N. Willson, Jr., "A 300 MHz quadrature direct digital synthesizer/mixer in 0.25 μm CMOS," in *IEEE International Solid State Circuits Conference, Digest of Technical Papers*, San Diego, 3-7 February 2002, pp. 132-133.

[96] S. Twelves, and C.J. Kikkert, "Performance of a digital stereo FM modulator with reduced output resolution," *IEEE Transactions on Broadcasting*, vol. 43, March 1997, pp. 104 - 113.

[97] J. Vankka, "Methods of mapping from phase to sine amplitude in direct digital synthesis," in *Proceedings of the IEEE Frequency Control Symposium*, 1996, pp. 942-950.

[98] J. Vankka, "Spur reduction techniques in sine output direct digital synthesis," in *Proceedings of the IEEE Frequency Control Symposium*, 1996, pp. 951-959.

[99] G.-J. van Rooyen and J.G. Lourens, "A quadrature baseband approach to direct digital FM synthesis," *IEEE Transactions on Broadcasting*, vol. 46, no. 3, September 2000, pp. 227-230.

[100] J. Volder, "The CORDIC trigonometric computing technique," *IRE Transactions on Electronic Computers*, vol. EC-8, no 3, September 1959, pp. 330-334.

[101] L.A. Weaver Jr. and R.J. Kerr, "High resolution phase to sine amplitude conversion," U.S. Patent 4 905 177, 27 February, 1990.

[102] C.E. Wheatley III and D.E. Phillips, "Spurious suppression in direct digital synthesizers," in *Proceedings of the 35th Annual Frequency Control Symposium*, 1981, pp. 428-435.

[103] C.E. Wheatley III, "Digital frequency synthesizer with random jittering for reducing discrete spectral spurs," U.S. Patent 4 410 954, 18 October, 1983.

[104] Y. Wu and J. Li, "The design of digital radar receivers," *IEEE Aerospace and Electronic Systems Magazine*, January 1998, pp. 35-41.

[105] 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," in *Proceedings of the IEEE Custom Integrated Circuits Conference*, 1995, pp. 319-322.

[106] 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, no. 2, February 1998, pp. 210-217.

[107] A. Yamagishi, H. Nosaka, M. Muraguchi, and T. Tsukahara, "A phase-interpolation direct digital frequency synthesizer with an adaptive integrator," *IEEE Transactions on Microwave Theory and Techniques*, vol. 48, no. 6, June 2000, pp. 905-909.

[108] G. Zhang, D. Al-Khalili, R. Inkol, R. Saper, "A novel approach to the design of I/Q demodulation filters," *IEE Proceedings on Vision, Image and Signal Processing*, vol. 141, no. 3, June 1994, pp. 154-160.

Bibliography

J.-C. Chih, J.-Y. Chou, S.-G. Chen, "An efficient direct digital frequency synthesizer based on two-level table lookup," in *Proceedings of the IEEE International Frequency Control Symposium*, 2001, pp. 824-827.

K.-H. Cho and H. Samueli, "A frequency-agile single-chip QAM modulator with beamforming diversity," *IEEE Journal of Solid State Circuits*, vol. 36, no. 3, March 2001, pp. 398-407.

W.A. Chren Jr., "RNS-based enhancements for direct digital frequency synthesis," *IEEE Transactions on Circuits and Systems II*, vol. 42, no. 8, August 1995, pp. 516-524.

W.A. Chren Jr., "Low delay-power product CMOS design using one-hot residue coding," in *Proceeding of the 1995 International Symposium on Low Power Design*, Dana Point, CA, 1995, pp. 145-150.

W.A. Chren Jr., "One-hot residue coding for low delay-power product CMOS design," *IEEE Transactions on Circuits and Systems II*, vol. 45, no. 3, March 1998, pp. 303-313.

B. De Smedt and G. Gielen, "Models for systematic design and verification of frequency synthesizers," *IEEE Transactions on Circuits and Systems –II: Analog and Digital Signal Processing*, vol. 46, no. 10, October 1999, pp. 1301-1308.

M.D. Ercegovac and T. Lang, "Fast cosine/sine implementation using on-line CORDIC," in *Proceedings ot the 21st Annual Asilomar Conference on Signals, Systems and Computers*, 2-4 November 1987, pp. 222-226.

R. Ertl and J. Baier, "Increasing the frequency resolution of NCO-systems using a circuit based on a digital adder," *IEEE Transactions on Circuits and Systems II*, vol. 43, March 1996, pp. 266-269.

B. Giebel, J. Lutz, and P. O'Leary, "Digitally controlled oscillator," *IEEE Journal of Solid State Circuits*, vol. 24, no. 6, June 1989, pp. 640-645.

R.P. Giffard and L.S. Cutler, "A low frequency, high resolution digital synthesizer," in *Proceedings of the 46th Annual Frequency Control Symposium*, 1992, pp. 188-192.

M.T. Hill and A. Cantoni, "An integrated high-frequency narrow-band high-resolution synthesizer," *IEEE Transactions on Circuits and Systems –II: Analog and Digital Signal Processing*, vol. 46, no. 9, September 1999, pp. 1171-1178.

M. Kosunen, J. Vankka, M. Waltari, L. Sumanen, K. Koli and K. Halonen, "A CMOS quadrature baseband frequency synthesizer/modulator," *Analog Integrated Circuits and Signal Processing*, vol. 18, no. 1, January 1999, pp. 55-67.

L.J. Kushner, "The composite DDS – a new direct digital synthesizer architecture," in *Proceedings of the 47th Annual Frequency Control Symposium*, 1993, pp. 255-260.

R. Larsen and S.-L. Lu, "Interpolation-based digital quadrature frequency synthesizer," in *Proceedings of the IEEE International ASIC/SOC Conference*, Arlington, VA, September 13-16, 2000, pp. 48-52.

L.K. Lau, R. Jain, H. Samueli, H.T. Nicholas and E.G. Cohen, "A silicon compiler for direct digital frequency synthesis," *Journal of VLSI Signal Processing*, vol. 4, no. 2-3, 1992, pp. 213-226.

L. Lo Presti and G. Cardamone, "A direct digital frequency synthesizer using an IIR filter implemented with a DSP microprocessor," in *Proceedings of the 1994 IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 3, April 1994, pp. 201-204.

L. Lo Presti, G. Cardamone, A De Marchi and E. Rubiola, "A new architecture for a sine-wave output DDS with high spectral purity," *Direct Digital Frequency Synthesizers*, V.F. Kroupa, Ed., IEEE Press, 1999, pp. 352-357.

F. Lu, H. Samueli, J. Yuan, and C. Svensson, "A 700-MHz 24-b pipelined accumulator in 1.2 μm CMOS for application as a numerically controlled oscillator," *IEEE Journal of Solid State Circuits*, vol. 28, no. 8, August 1993, pp. 878-886.

U. Meyer-Bäse, S. Wolf, and F. Taylor, "Accumulator-synthesizer with error-compensation," *IEEE Transactions on Circuits and Systems –II: Analog and Digital Signal Processing*, vol. 45, no. 7, July 1998, pp. 885-890.

J.S. Min and H. Samueli, "Analysis and design of a frequency-hopped spread-spectrum transceiver for wireless personal communications," *IEEE Transactions on Vehicular Technology*, vol. 49, no. 5, September 2000, pp. 1719-1731.

P. Nagvajara, T.R. Damarla, J. Wang, and S. Chansilp, "A phase-coherent numerically controlled oscillator based on pipeline architecture," in *11th annual IEEE International ASIC Conference*, 1998, pp. 355-358.

H. Nosaka, Y. Yamaguchi, and M. Muraguchi, "A non-binary direct digital synthesizer with an extended phase accumulator," *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 48, no. 1, January 2001, pp. 293-298.

H. Nosaka, Y. Yamaguchi, A. Yamagishi, H. Fukuyama and M. Muraguchi, "A low-power direct digital synthesizer using a self-adjusting phase-interpolation technique," *IEEE Journal of Solid State Circuits*, vol. 36, no. 8, August 2001, pp. 1281-1285.

V.S. Reinhardt, "Spur reduction techniques in direct digital synthesizers," in *Proceedings of the 47th Annual Frequency Control Symposium*, 1993, pp. 230-241.

R. Richter and H.-J. Jentschel, "A virtual clock enhancement method for DDS using an analog delay line," *IEEE Journal of Solid State Circuits*, vol. 36, no. 7, July 2001, pp. 1158-1161.

R.de J. Romero-Troncoso and G. Espinosa-Flores-Verdad, "Phase accumulator synthesis for DDS applications," *Electronic Letters*, vol. 35, no. 10, 13 May 1999, pp. 770-772.

H.-G. Ryu, Y.-Y. Kim, H.-M. Yu and S.-B. Ryu, "Design of DDFS-driven PLL frequency synthesizer with reduced complexity," *IEEE Transactions on Consumer Electronics*, vol. 47, no. 1, February 2001, pp. 194-198.

H.-G. Ryu, Y.-Y. Kim and H.-M. Yu, "Reduced-complexity design for triple-tunable frequency synthesizer," *Electronic Letters*, vol. 37, no. 8, 12 April 2001, pp. 482-484.

D.L. Schilling, R.L. Pickholtz, and L.B. Milstein, "Spread spectrum goes commercial," *IEEE Spectrum*, August 1990, pp. 40-45.

D.L. Schilling, L.B. Milstein, R.L. Pickholtz, M. Kullback, and F. Miller, "Spread spectrum for commercial communications," *IEEE Communications Magazine*, April 1991, pp. 66-79.

M.A. Taslakov, "Direct digital synthesizer with improved spectrum at low frequencies," in *Proceedings of the 2000 IEEE/IEA International Frequency Control Symposium and Exhibition*, Kansas City, Missouri, 7-9 June 2000, pp. 280-283.

M. Thompson, "Low-latency, high-speed numerically controlled oscillator using progression-of-states technique," *IEEE Journal of Solid State Circuits*, vol. 27, no. 1, January 1992, pp. 113-117.

R. Uusikartano, J. Niitylahti and M. Renfors, "Area-optimized FPGA implementation of a digital FM modulator" in *Proceedings of the 1999 International Symposium on Circuits and Systems (ISCAS 1999)*, 30 May - 2 June 1999, pp. 360-362.

R. Uusikartano and J. Niitylahti, "A compact digital frequency synthesizer for GSM IF up/down converter" in *Proceedings of the 2000 International Symposium on Circuits and Systems (ISCAS 2000)*, Switzerland, May 2000, pp. 113-116.

R. Uusikartano and J. Niitylahti, "A digital frequency synthesizer for a 2.4 GHz fast frequency hopping transceiver" in *Proceedings of the 43rd IEEE Midwest Symposium on Circuits and Systems, 2000*, 8 - 11 August 2000, pp. 420-423.

J. Vankka, M. Waltari, M. Kosunen and K.A.I. Halonen, "A direct digital synthesizer with an on-chip D/A converter," *IEEE Journal of Solid State Circuits*, vol. 33, no. 2, February 1998, pp. 218-227.

J. Vankka, M. Honkanen and K.A.I. Halonen, "A multicarrier GMSK modulator," *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 6, June 2001, pp. 1070-1079.

B.C. Wong and H. Samueli, "A 200 MHz all-digital QAM modulator and demodulator in 1.2 μm CMOS for digital radio applications," *IEEE Journal of Solid State Circuits*, vol. 26, December 1991, pp. 1970-1979.