Dynamic element matching techniques for data converters

Jerry Wayne Bruce
University of Nevada, Las Vegas

Follow this and additional works at: https://digitalscholarship.unlv.edu/rtds

Repository Citation
https://digitalscholarship.unlv.edu/rtds/3093

This Dissertation is brought to you for free and open access by Digital Scholarship@UNLV. It has been accepted for inclusion in UNLV Retrospective Theses & Dissertations by an authorized administrator of Digital Scholarship@UNLV. For more information, please contact digitalscholarship@unlv.edu.
INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

UMI®
DYNAMIC ELEMENT MATCHING TECHNIQUES

FOR DATA CONVERTERS

by

Jerry Wayne Bruce, II

Bachelor of Science in Engineering
University of Alabama in Huntsville
1991

Master of Science in Electrical Engineering
Georgia Institute of Technology
1993

A dissertation submitted in partial fulfillment
of the requirements for the

Doctor of Philosophy Degree
Department of Electrical and Computer Engineering
Howard R. Hughes College of Engineering

Graduate College
University of Nevada Las Vegas
May 2000

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
The Dissertation prepared by

Jerry Wayne Bruce, II

Entitled

Dynamic Element Matching Techniques for Data Converters

is approved in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Examination Committee Chair

Dean of the Graduate College

Examination Committee Member

Graduate College Faculty Representative
ABSTRACT

Dynamic Element Matching Techniques for Data Converters

by

Jerry Wayne Bruce, II

Dr. Peter Stubberud, Examination Committee Chair
Associate Professor of Electrical and Computer Engineering
University of Nevada Las Vegas

Analog to digital converter (ADC) circuit component errors create nonuniform quantization code widths and create harmonic distortion in an ADC's output. In this dissertation, two techniques for estimating an ADC's output spectrum from the ADC's transfer function are determined. These methods are compared to a symmetric power function and asymmetric power function approximations. Standard ADC performance metrics, such as SDR, SNDR, SNR, and SFDR, are also determined as a function of the ADC's transfer function approximations. New dynamic element matching (DEM) flash ADCs are developed. An analysis of these DEM flash ADCs is developed and shows that these DEM algorithms improve an ADC's performance. The analysis is also used to analyze several existing DEM ADC architectures.

Digital to analog converter (DAC) circuit component errors create nonuniform quantization code widths and create harmonic distortion in a DAC's output. In this dissertation, an exact relationship between a DAC's integral nonlinearity (INL) and its
output spectrum is determined. Using this relationship, standard DAC performance metrics, such as SDR, SNDR, SNR, and SFDR, are calculated from the DAC's transfer function. Furthermore, an iterative method is developed which determines an arbitrary DAC's transfer function from observed output magnitude spectra. An analysis of DEM techniques for DACs, including the determination of several suitable metrics by which DEM techniques can be compared, is derived. The performance of a given DEM technique is related to standard DAC performance metrics, such as SDR, SNDR, and SFDR. Conditions under which DEM techniques can guarantee zero average INL and render the distortion due to mismatched components as white noise are developed. Several DEM circuits proposed in the literature are shown to be equivalent and have hardware efficient implementations based on multistage interconnection networks. Example DEM circuit topologies and their hardware efficient VLSI implementations are also presented.
# TABLE OF CONTENTS

**ABSTRACT** ............................................................................................................... iii

**LIST OF FIGURES** .................................................................................................. vii

**LIST OF TABLES** ..................................................................................................... xi

**ACRONYMS** ............................................................................................................ xii

**ACKNOWLEDGMENTS** ........................................................................................... xiii

**CHAPTER 1**  **INTRODUCTION** ............................................................................... 1

**CHAPTER 2**  **BACKGROUND** ............................................................................... 5
  2.1 Analog to digital conversion and flash ADCs ............................................... 6
  2.2 Mismatched components in ADCs .......................................................... 14
  2.3 Digital to analog conversion and flash DACs ......................................... 19
  2.4 Mismatch component errors in DACs .................................................... 22
  2.5 Data converter metrics and terminology ............................................. 29
  2.6 Principles of DEM in ADCs ............................................................... 41
  2.7 Principles of DEM in DACs ............................................................... 45

**CHAPTER 3**  **ANALOG TO DIGITAL CONVERTER TRANSFER FUNCTION ERRORS AND HARMONIC DISTORTION** .................................... 50
  3.1 Power function approximations of an ADC's transformation .............. 51
  3.2 Fourier series representation of an ADC's transformations .............. 59
  3.3 Example ........................................................................................... 73
  3.4 Summary ........................................................................................... 78

**CHAPTER 4**  **DYNAMIC ELEMENT MATCHING TECHNIQUES FOR ANALOG TO DIGITAL CONVERTERS** .......................................................... 79
  4.1 Principles of DEM in ADCs ............................................................... 82
  4.2 An analysis of DEM ADCs .............................................................. 85
  4.3 Resistor string flash ADCs ............................................................. 102
  4.4 Current steering flash ADCs .......................................................... 115
  4.5 Conclusions .................................................................................... 125

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
LIST OF FIGURES

2.1 Ideal three bit ADC transfer function ................................................................. 6
2.2 System level illustration of analog to digital conversion ....................................... 7
2.3 Sample and hold amplifier model ......................................................................... 7
2.4 Example illustrating SHA operation ..................................................................... 8
2.5 ADC block diagram ............................................................................................ 9
2.6 Three bit resistor string flash ADC ..................................................................... 10
2.7 Additive dither model ....................................................................................... 13
2.8 A B bit current steering flash ADC block diagram ........................................... 17
2.9 Current steering flash th unit ADC cell ............................................................ 18
2.10 CMOS current comparator .............................................................................. 18
2.11 Ideal unipolar three bit DAC transfer function ............................................. 20
2.12 DAC block diagram ....................................................................................... 20
2.13 A B bit flash DAC architecture ....................................................................... 22
2.14 Three bit resistor string flash DAC architecture ............................................. 23
2.15 Three bit charge scaling flash DAC architecture .......................................... 25
2.16 Three bit current steering flash DAC architecture .......................................... 27
2.17 ADC transfer function terminology .................................................................. 31
2.18 DAC transfer function terminology .................................................................. 32
2.19 The quantization error characteristic for an ideal three bit ADC ....................... 35
2.20 Nonlinearity metrics for ADCs ......................................................................... 38
2.21 Nonlinearity metrics for DACs ......................................................................... 39
2.22 Voltage divider demonstration of clocked averaging DEM ............................... 42
2.23 A B bit dynamic element matching DAC architecture ...................................... 45
3.1 An example three bit ADC transformation ......................................................... 52
3.2 The fluctuating portion of the example three bit ADC transformation shown in Figure 3.1................................................................. 53
3.3 Three bit ADC transformations ......................................................................... 60
3.4 Three bit ADC transformations ......................................................................... 61
3.5 Block diagram of ADC operation ...................................................................... 62
3.6 Sampled three bit ADC transformations from Figure 3.3 ..................................... 66
3.7 Sampled three bit ADC transformations from Figure 3.4 .................................... 67
3.8 Power spectral density plot of the linear part of the Example's nonlinear ADC output. ........................................................................................................ 74
3.9 Power spectral density plot of the Example's nonlinear ADC output .................. 75
3.10 Power spectral density plot of the quantization error part of the Example's nonlinear ADC output ................................................................. 76
3.11 Power spectral density plot of the harmonic distortion part of the Example's nonlinear ADC output .............................................................. 77
4.1 A B bit dynamic element matching flash ADC architecture .............................. 83
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.2</td>
<td>Power spectral density of the output of a six bit ideal ADC.</td>
<td>95</td>
</tr>
<tr>
<td>4.3</td>
<td>Power spectral density of the output of a six bit ADC.</td>
<td>96</td>
</tr>
<tr>
<td>4.4</td>
<td>Expected distortion transformation for the barrel shift network DEM ADC with linear gradient mismatch errors.</td>
<td>97</td>
</tr>
<tr>
<td>4.5</td>
<td>Expected distortion squared transformation for the barrel shift network DEM ADC with linear gradient mismatch errors.</td>
<td>98</td>
</tr>
<tr>
<td>4.6</td>
<td>Power spectral density of the output of a six bit DEM ADC.</td>
<td>99</td>
</tr>
<tr>
<td>4.7</td>
<td>Power spectral density of the output of a six bit DEM ADC.</td>
<td>100</td>
</tr>
<tr>
<td>4.8</td>
<td>Expected distortion transformation for the generalized cube network DEM ADC with linear gradient mismatch errors.</td>
<td>101</td>
</tr>
<tr>
<td>4.9</td>
<td>Expected distortion transformation and distortion squared transformation for the generalized cube network DEM ADC with linear gradient mismatch errors.</td>
<td>102</td>
</tr>
<tr>
<td>4.10</td>
<td>Power spectral density of the output of a six bit DEM ADC.</td>
<td>103</td>
</tr>
<tr>
<td>4.11</td>
<td>Power spectral density of the output of a six bit DEM ADC.</td>
<td>104</td>
</tr>
<tr>
<td>4.12</td>
<td>Block diagram of a $B$ bit resistor string flash ADC.</td>
<td>105</td>
</tr>
<tr>
<td>4.13</td>
<td>The $n$th resistor string unit ADC cell.</td>
<td>108</td>
</tr>
<tr>
<td>4.14</td>
<td>Barrel shift DEM flash ADC architecture.</td>
<td>109</td>
</tr>
<tr>
<td>4.15</td>
<td>The $n$th resistor string unit ADC cell from [13].</td>
<td>111</td>
</tr>
<tr>
<td>4.16</td>
<td>The $n$th resistor string unit ADC cell from [70].</td>
<td>113</td>
</tr>
<tr>
<td>4.17</td>
<td>A $B$ bit current steering flash ADC block diagram.</td>
<td>116</td>
</tr>
<tr>
<td>4.18</td>
<td>The $n$th current steering flash unit ADC cell.</td>
<td>117</td>
</tr>
<tr>
<td>4.19</td>
<td>CMOS current comparator.</td>
<td>117</td>
</tr>
<tr>
<td>4.20</td>
<td>$B$ bit DEM current steering flash ADC architecture.</td>
<td>120</td>
</tr>
<tr>
<td>4.21</td>
<td>Current steering flash unit ADC cell.</td>
<td>120</td>
</tr>
<tr>
<td>4.22</td>
<td>Signal processing model of the unit ADC cell in Figure 4.21.</td>
<td>122</td>
</tr>
<tr>
<td>5.1</td>
<td>Power spectral density plot of the linear part of the Example's nonlinear DAC output.</td>
<td>142</td>
</tr>
<tr>
<td>5.2</td>
<td>Power spectral density plot of the Example’s nonlinear DAC output.</td>
<td>143</td>
</tr>
<tr>
<td>5.3</td>
<td>Power spectral density plot of the harmonic distortion part of the Example’s nonlinear DAC output.</td>
<td>144</td>
</tr>
<tr>
<td>5.4</td>
<td>Power spectral density plot of the Example's nonlinear DAC output.</td>
<td>145</td>
</tr>
<tr>
<td>5.5</td>
<td>Power spectral density plot of the Example’s nonlinear DAC output.</td>
<td>146</td>
</tr>
<tr>
<td>6.1</td>
<td>A $B$ bit dynamic element matching DAC architecture.</td>
<td>151</td>
</tr>
<tr>
<td>6.2</td>
<td>Four example six bit DAC transformations.</td>
<td>157</td>
</tr>
<tr>
<td>6.3</td>
<td>Four example six bit DAC integral nonlinearity transformations.</td>
<td>158</td>
</tr>
<tr>
<td>6.4</td>
<td>Power spectral density of the output of a six bit linear DAC.</td>
<td>165</td>
</tr>
<tr>
<td>6.5</td>
<td>Power spectral density of the output of a six bit DAC.</td>
<td>166</td>
</tr>
<tr>
<td>6.6</td>
<td>Power spectral density of the output of a six bit DEM DAC.</td>
<td>168</td>
</tr>
<tr>
<td>6.7</td>
<td>Power spectral density of the output of a six bit DEM DAC.</td>
<td>169</td>
</tr>
<tr>
<td>6.8</td>
<td>Power spectral density of the output of a six bit DEM DAC.</td>
<td>170</td>
</tr>
<tr>
<td>6.9</td>
<td>Power spectral density of the output of a six bit DEM DAC.</td>
<td>171</td>
</tr>
<tr>
<td>6.10</td>
<td>Power spectral density of the output of a six bit DEM DAC.</td>
<td>172</td>
</tr>
<tr>
<td>6.11</td>
<td>Power spectral density of a six bit DEM DAC's distortion output.</td>
<td>173</td>
</tr>
<tr>
<td>6.12</td>
<td>Power spectral density of the output of a six bit DAC.</td>
<td>174</td>
</tr>
<tr>
<td>6.13</td>
<td>Power spectral density of the output of a six bit DEM DAC.</td>
<td>175</td>
</tr>
<tr>
<td>6.14</td>
<td>Power spectral density of the output of a six bit DEM DAC.</td>
<td>177</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>6.15</td>
<td>Power spectral density of the output of a six bit DEM DAC</td>
<td>178</td>
</tr>
<tr>
<td>6.16</td>
<td>Power spectral density of the output of a six bit DEM DAC</td>
<td>179</td>
</tr>
<tr>
<td>6.17</td>
<td>Power spectral density of the output of a six bit DEM DAC</td>
<td>180</td>
</tr>
<tr>
<td>6.18</td>
<td>Power spectral density of a six bit DEM DAC’s distortion output</td>
<td>181</td>
</tr>
<tr>
<td>7.1</td>
<td>An example of a graph</td>
<td>184</td>
</tr>
<tr>
<td>7.2</td>
<td>Two input, two output interchange switch</td>
<td>185</td>
</tr>
<tr>
<td>7.3</td>
<td>Example graph and corresponding interconnection networks</td>
<td>187</td>
</tr>
<tr>
<td>7.4</td>
<td>An example interconnection network</td>
<td>188</td>
</tr>
<tr>
<td>7.5</td>
<td>Eight input crossbar graph</td>
<td>192</td>
</tr>
<tr>
<td>7.6</td>
<td>Eight input crossbar network</td>
<td>192</td>
</tr>
<tr>
<td>7.7</td>
<td>Three bit Benes graph</td>
<td>193</td>
</tr>
<tr>
<td>7.8</td>
<td>An eight input Benes network</td>
<td>194</td>
</tr>
<tr>
<td>7.9</td>
<td>An eight input Waksman permutation network</td>
<td>195</td>
</tr>
<tr>
<td>7.10</td>
<td>Three stage butterfly graph</td>
<td>197</td>
</tr>
<tr>
<td>7.11</td>
<td>A three stage IBCN</td>
<td>198</td>
</tr>
<tr>
<td>7.12</td>
<td>Example of a permutation in a three bit IBCN</td>
<td>199</td>
</tr>
<tr>
<td>7.13</td>
<td>Three stage generalized cube graph</td>
<td>201</td>
</tr>
<tr>
<td>7.14</td>
<td>A three stage GCN</td>
<td>202</td>
</tr>
<tr>
<td>7.15</td>
<td>Construction of a r+1 bit GCN</td>
<td>203</td>
</tr>
<tr>
<td>7.16</td>
<td>Example of input-output permutation in a three bit GCN</td>
<td>204</td>
</tr>
<tr>
<td>7.17</td>
<td>Three stage barrel shift graph</td>
<td>205</td>
</tr>
<tr>
<td>7.18</td>
<td>Transmission gate construction of a three stage barrel shift network</td>
<td>206</td>
</tr>
<tr>
<td>7.19</td>
<td>Example of a permutation in a three bit BSN</td>
<td>207</td>
</tr>
<tr>
<td>7.20</td>
<td>A three bit omega network</td>
<td>208</td>
</tr>
<tr>
<td>7.21</td>
<td>A B bit dynamic element matching flash DAC architecture</td>
<td>209</td>
</tr>
<tr>
<td>7.22</td>
<td>A three bit Benes network DEM DAC architecture</td>
<td>212</td>
</tr>
<tr>
<td>7.23</td>
<td>Three bit IBCN DEM DAC architecture</td>
<td>214</td>
</tr>
<tr>
<td>7.24</td>
<td>Three bit reduced complexity IBCN DEM DAC</td>
<td>215</td>
</tr>
<tr>
<td>7.25</td>
<td>Three bit GCN DEM DAC architecture</td>
<td>216</td>
</tr>
<tr>
<td>7.26</td>
<td>A three bit “Full Randomization” DEM DAC architecture</td>
<td>218</td>
</tr>
<tr>
<td>7.27</td>
<td>The “Full Randomization” DEM network’s switching block</td>
<td>219</td>
</tr>
<tr>
<td>7.28</td>
<td>A one bit “Full Randomization” DEM network</td>
<td>220</td>
</tr>
<tr>
<td>7.29</td>
<td>Construction of a k+1 bit “Full Randomization” DEM network</td>
<td>220</td>
</tr>
<tr>
<td>7.30</td>
<td>A three bit BTN DEM DAC architecture</td>
<td>221</td>
</tr>
<tr>
<td>7.31</td>
<td>Construction of a k+1 bit BTN</td>
<td>222</td>
</tr>
<tr>
<td>7.32</td>
<td>One bit BTN</td>
<td>224</td>
</tr>
<tr>
<td>7.33</td>
<td>VLSI implementation of a two bit BTN</td>
<td>226</td>
</tr>
<tr>
<td>7.34</td>
<td>VLSI implementation of a three bit BTN</td>
<td>228</td>
</tr>
<tr>
<td>7.35</td>
<td>VLSI implementation of a five bit BTN</td>
<td>230</td>
</tr>
<tr>
<td>7.36</td>
<td>Three bit BSN DEM DAC architecture</td>
<td>231</td>
</tr>
<tr>
<td>7.37</td>
<td>Three bit reduced complexity BSN DEM DAC architecture</td>
<td>233</td>
</tr>
<tr>
<td>A.1</td>
<td>Regulated cascode current sink</td>
<td>240</td>
</tr>
<tr>
<td>A.2</td>
<td>Example One’s floorplan</td>
<td>241</td>
</tr>
<tr>
<td>A.3</td>
<td>Six bit current steering DEM DAC using BTN network</td>
<td>242</td>
</tr>
<tr>
<td>A.4</td>
<td>Regulated cascode current sink</td>
<td>243</td>
</tr>
<tr>
<td>A.5</td>
<td>Example Two’s floorplan</td>
<td>244</td>
</tr>
</tbody>
</table>
A.6 Five bit switched capacitor DEM DAC using BT network ........................................ 245
A.7 Example Three’s floorplan ................................................................................ 247
A.8 Six bit current steering DEM DAC using barrel shift network ......................... 248
# LIST OF TABLES

2.1 Correspondence among natural binary, gray, and thermometer codes. .......................... 30
3.1 Example’s ideal and nonideal ADC quantization decision levels................................. 74
3.2 Summary of Example’s results.................................................................................... 77
5.1 Nonlinear DAC transfer function................................................................................ 142
6.1 Comparison of Example One’s SDR........................................................................ 174
6.2 Comparison of Example One’s SNDR ...................................................................... 175
6.3 Comparison of Example Two’s SDR......................................................................... 181
6.4 Comparison of Example Two’s SNDR ...................................................................... 182
7.1 Number of transmission gates in construction of the generalized cube DEM network implementations ........................................................................................................... 227
7.2 Characteristics of several DAC DEM networks......................................................... 235
A.1 Example One’s major subsystems............................................................................... 241
A.2 Example Two’s major subsystems ............................................................................. 246
A.3 Example Three’s major subsystems .......................................................................... 249
# ACRONYMS

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Term</th>
</tr>
</thead>
<tbody>
<tr>
<td>AC</td>
<td>Alternating current</td>
</tr>
<tr>
<td>A/D</td>
<td>Analog to digital</td>
</tr>
<tr>
<td>ADC</td>
<td>Analog to digital converter</td>
</tr>
<tr>
<td>BSN</td>
<td>Barrel shift network</td>
</tr>
<tr>
<td>BTN</td>
<td>Binary tree network</td>
</tr>
<tr>
<td>CMOS</td>
<td>Complementary metal oxide semiconductor</td>
</tr>
<tr>
<td>D/A</td>
<td>Digital to analog</td>
</tr>
<tr>
<td>DAC</td>
<td>Digital to analog converter</td>
</tr>
<tr>
<td>DC</td>
<td>Direct current</td>
</tr>
<tr>
<td>DEM</td>
<td>Dynamic element matching</td>
</tr>
<tr>
<td>DNL</td>
<td>Differential nonlinearity</td>
</tr>
<tr>
<td>DSP</td>
<td>Digital signal processing</td>
</tr>
<tr>
<td>DTCA</td>
<td>Discrete time, continuous amplitude</td>
</tr>
<tr>
<td>ENOB</td>
<td>Effective number of bits</td>
</tr>
<tr>
<td>FET</td>
<td>Field effect transistor</td>
</tr>
<tr>
<td>FRDEM</td>
<td>Full randomization dynamic element matching</td>
</tr>
<tr>
<td>GCN</td>
<td>Generalized cube network</td>
</tr>
<tr>
<td>IBCN</td>
<td>Indirect binary cube network</td>
</tr>
<tr>
<td>INL</td>
<td>Integral nonlinearity</td>
</tr>
<tr>
<td>LFSR</td>
<td>Linear feedback shift register</td>
</tr>
<tr>
<td>LSB</td>
<td>Least significant bit</td>
</tr>
<tr>
<td>MOS</td>
<td>Metal oxide semiconductor</td>
</tr>
<tr>
<td>MSB</td>
<td>Most significant bit</td>
</tr>
<tr>
<td>NFET</td>
<td>N-channel field effect transistor</td>
</tr>
<tr>
<td>PDF</td>
<td>Probability density function</td>
</tr>
<tr>
<td>PFET</td>
<td>P-channel field effect transistor</td>
</tr>
<tr>
<td>PSD</td>
<td>Power spectral density</td>
</tr>
<tr>
<td>RMS</td>
<td>Root mean square</td>
</tr>
<tr>
<td>SBS</td>
<td>Stochastic barrel shifter</td>
</tr>
<tr>
<td>SDR</td>
<td>Signal to distortion ratio</td>
</tr>
<tr>
<td>SFDR</td>
<td>Spurious free dynamic range</td>
</tr>
<tr>
<td>SHA</td>
<td>Sample and hold amplifier</td>
</tr>
<tr>
<td>SNDR</td>
<td>Signal to noise plus distortion ratio</td>
</tr>
<tr>
<td>SNR</td>
<td>Signal to noise ratio</td>
</tr>
<tr>
<td>SQNR</td>
<td>Signal to quantization plus noise ratio</td>
</tr>
<tr>
<td>THD</td>
<td>Total harmonic distortion</td>
</tr>
<tr>
<td>VLSI</td>
<td>Very large scale integration</td>
</tr>
</tbody>
</table>
ACKNOWLEDGMENTS

This dissertation is dedicated to the memory of my father. His encouraging words were always a source of inspiration, and his love of knowledge proved an excellent example. I know that he would have considered this dissertation to be one of the most prized volumes in his library.

Words cannot express my gratitude to my loving wife, Lori. She has supported me in so many ways. I am forever indebted to her for the emotional, professional, and technical guidance which she alone is capable of giving. Her unwavering love and much needed encouragement is a large reason that this dissertation is finished. I look forward to repaying the many missed hours I owe her.

I would like to thank Dr. Peter Stubberud for the time and energy he invested. He was an excellent advisor and will always be a good friend. Our daily conversations (on every possible topic) will be sorely missed.

Many others provided expertise, encouragement, and guidance. I would like to thank my committee members for their time and advice. The ECE department faculty and staff provided a positive and enjoyable environment in which to work. I would like to acknowledge the students who gave assistance in my research. Specifically, I would like to thank Bret Steadman for his thorough literature search and excellent thesis, Elias Dagher for his analog design on the switched capacitor DAC, and Long Huong and Alec Wong for their work on current steering DACs.
CHAPTER I

INTRODUCTION

Analog to digital conversion and digital to analog conversion are essential operations in many digital systems, and many of these digital systems require high performance data converters. For example, digital instrumentation applications, such as digital spectral analysis, require high speed, high accuracy analog to digital converters (ADCs) [13]. Wireless communication systems require an extremely low harmonic distortion, moderate resolution digital to analog converter (DAC) for direct digital synthesis [34].

Many ADC and DAC architectures rely on matched components to perform their tasks of data conversion. In practice, perfectly matched components are impossible to fabricate, and mismatch errors, which are defined as the difference between the designed and actual component values, are inevitable. In VLSI circuits, static mismatch errors are caused by process variations such as mask misalignment, nonuniform oxide thickness, and nonuniform doping densities. Additional mismatch errors can be generated by temperature gradients across the circuit, component aging and component noise [40], [59]. In ADCs and DACs, mismatch errors cause nonuniform code widths which cause errors in the converter's transfer function. As a result, the converter's performance is diminished.

For data converters that use matched components, self-calibrating circuits, which are circuits that periodically adjust their output, can reduce the effects of component mismatch errors. Self-calibrating circuits can correct both analog and digital circuit errors [52].
Analog error correction, an analog quantity, such as voltage, current, or capacitance, is digitally controlled to obtain the required accuracy. In digital error correction, a digital signal is modified to correct an error resulting from analog inaccuracies. Although self-calibrating designs are effective, they can be significantly complex [6]. Another method used to reduce mismatched component errors is error cancellation which operates such that the unknown error enters the circuit twice with opposite polarities and is cancelled. An error cancellation technique must be designed specifically into a data converter's architecture, and therefore, it cannot be easily applied to existing data converter architectures [65].

For data converters that use matched resistors, several techniques, including special VLSI layout techniques, laser trimming, self-calibration, error cancellation and dynamic element matching, can reduce the effects of component mismatch errors. Special VLSI layout techniques can reduce resistor mismatch errors caused by highly varying contact resistances by connecting resistors without leaving the resistor layer and avoiding current path contacts [78]. When special VLSI layout techniques and bulk CMOS processes cannot produce sufficiently matched resistors, special fabrication processes, such as those that use silicon chromium, nickel chromium, or tantalum nitride, can be used to fabricate more precisely matched resistors. However, these processes are expensive, and the fabricated resistors lack thermal stability and are susceptible to corrosion [30]. Laser trimming is another technique that can reduce the mismatch error in resistors. Resistor values can be changed by removing resistor material with a highly focused laser beam at the wafer stage. Although laser trimming the physical dimensions of a resistor string after fabrication can reduce mismatch errors, the procedure is expensive [3]. Furthermore, in
some circuits, trimming one component can negatively alter the behavior of the remaining
circuitry [65].

Another method that can reduce the effects of component mismatch errors is dynamic
element matching (DEM). DEM is a dynamic process that reduces the effects of
component mismatches in electronic circuits by rearranging dynamically the
interconnections of mismatched components so that the time averages of the equivalent
components at each of the component positions are equal or nearly equal [38]. By
appropriately varying the mismatched components' virtual positions, the effects of
mismatched components can be reduced, eliminated, or frequency shifted.

DEM has been used to reduce the effects of component mismatch errors in both ADCs
and DACs. Currently, the only DEM ADC technique reported in the open literature is the
stochastic barrel shifting flash ADC [13], [70]. The stochastic barrel shifting flash ADC
has been shown to have perfect quantization levels on average. However, the distortion
power resulting from ADC mismatch errors using this DEM technique is not known.

Several deterministic DEM techniques and stochastic DEM techniques have been
developed for DACs. Deterministic DEM techniques, such as clocked level averaging
[72], [73], data weighted averaging [6], and individual level averaging [16], [46], shape
harmonic distortion caused by the mismatched components in certain frequency bands.
Stochastic DEM techniques, such as stochastic level averaging, randomly permute DAC
circuit elements each sample. As a result, stochastic DEM techniques generate noise from
the mismatch errors and spreads this noise across the spectrum [15], [23].

DEM in ADCs and DACs has been implemented using several circuit topologies of
differing complexities, and to date, performance analyses of these DEM techniques have
been circuit topology specific and based primarily on simulation results [10], [15], [71]. In this dissertation, a method of analyzing DEM techniques quantitatively is developed, and the relationship between component mismatch errors and the signal to noise ratio (SNR), signal to distortion ratio (SDR), signal to noise plus distortion ratio (SNDR), and spurious free dynamic range (SFDR) is derived. The effects of a non ideal ADC's transformation on its output are estimated by three methods. A generalized analysis of the quantization threshold levels of DEM flash ADCs, including determination of several suitable metrics by which DEM techniques can be compared, is presented. The distortion power resulting from mismatched components for ADCs using DEM techniques is determined, and several DEM voltage division and current steering ADC architectures are analyzed. The relationship between a DAC's INL and its output spectrum is determined. Furthermore, an iterative method is presented to determine a DAC's transfer function from observed output magnitude spectrum. A generalized analysis of DEM techniques for DACs is derived, including the determination of several suitable metrics by which DEM techniques can be compared. The analysis relates the performance of a given DEM technique to standard DAC performance metrics, such as SDR, SNDR and SFDR. Several DEM circuits proposed in the literature are shown to be equivalent and have hardware efficient implementations based on well understood multistage interconnection networks. Finally, VLSI implementations of several DAC DEM circuits are presented.
CHAPTER 2

BACKGROUND

Analog to digital converters (ADCs) and digital to analog converters (DACs) have been implemented using a number of different architectures, including counter ramp, successive approximation, multistep, subranging, delta-sigma and flash [31], [59], [65]. Because flash converters convert signals quickly but grow exponentially in size as the number of bits increases, applications that require flash converters are typically high speed and have short word lengths. Also, because flash converters convert signals quickly, they are used as components in many other converter architectures [7].

Flash ADCs and flash DACs rely on matched components to perform their conversion tasks. However, perfectly matched components are impossible to fabricate. Because of variations in circuit fabrication processes, temperature gradients across the circuit, component aging, and component noise, circuit component values differ from their design values. As a result of these variations, called mismatch errors, ADC and DAC code widths are functions of their inputs and have degraded performance.

In this chapter, flash ADC architectures, flash DAC architectures, and sources of component mismatch errors are reviewed. Data converter terminology and metrics which quantify the performance of data converters are defined. Finally, a brief review of data converter DEM algorithms is presented. Although much of this chapter discusses flash...

5

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
converters, many of the developments can be applied to other converter architectures that use flash converters.

2.1 Analog to digital conversion and flash ADCs

An ideal $B$ bit ADC transforms an analog input signal, $x(nT)$, into a $B$ bit digital signal, $y[n]$, where $T$ is the ADC's sampling period and $n$ is an integer which indexes the sequence $y$. To convert signals, an ADC divides its input range into $2^B$ segments, and assigns an unique $B$ bit digital code to each of these segments. At each sample time, $nT$, the ADC converts the analog input signal into the digital code corresponding to the segment in which the analog input is located. Figure 2.1 shows an ideal three bit ADC transfer function.

Analog to digital conversion is often performed by a sample and hold amplifier (SHA) followed by an ADC. The SHA relaxes the timing requirements of the ADC by sampling a continuous time signal and holding its value until the next sample instant. The ADC quantizes the SHA's output and represents the quantity with the appropriate digital code.

![Ideal three bit ADC transfer function](image)

Figure 2.1. Ideal three bit ADC transfer function.
Figure 2.2 shows a system level illustration of such an A/D conversion process where the SHA and ADC are controlled by an external clock that has a period of $T$. The SHA samples the analog or continuous time input signal, $x_a(t)$, every $T$ seconds and holds the amplitude constant for the length of the sampling period. Thus, the SHA output signal, $x_o(t)$, is a continuous staircase type signal, such that

$$x_o(t) = x(nT) \quad nT \leq t \leq (n+1)T.$$ 

The ADC converts $x_o(t)$ into the digital signal, $x_B[n]$.

Figure 2.3 depicts a system model of an ideal SHA where the analog input, $x_a(t)$, is multiplied by an impulse train, $s(t)$, where

$$s(t) = \sum_{n=-\infty}^{\infty} \delta_o(t-nT),$$

where $\delta_o(t)$ is the Dirac delta function.

Figure 2.2. System level illustration of analog to digital conversion.

Figure 2.3. Sample and hold amplifier model.
and \( \delta_a(t) \) is a Dirac delta function. This operation produces another impulse train, \( x_s(t) \), where

\[
x_s(t) = x_a(t) \sum_{n=-\infty}^{\infty} \delta_a(t-nT) = \sum_{n=-\infty}^{\infty} x_a(nT)\delta_a(t-nT).
\]

By processing the impulse train, \( x_s(t) \), with the linear time-invariant zero order hold system that has impulse response, \( h_{\text{zoh}}(t) \), where

\[
h_{\text{zoh}}(t) = \begin{cases} 
1 & 0 < t < T \\
0 & \text{otherwise}
\end{cases}
\tag{2.1}
\]

the SHA's output, \( x_o(t) \), can be written as \( x_o(t) = x_a(t) * h_{\text{zoh}}(t) \).

Figure 2.4 illustrates the operation of the SHA in Figure 2.3. Figure 2.4(a) shows the sinusoidal SHA input, \( x_a(t) \). If the sampling period, \( T \), that is 32 times greater than the

![Figure 2.4](image-url)

**Figure 2.4.** Example illustrating SHA operation. (a) Analog input, \( x_a(t) \). (b) Sampled input, \( x_s(t) \). (c) SHA staircase signal output, \( x_o(t) \).
period of the sine wave, then the SHA generates the signals, $x_j(t)$ and $x_o(t)$, shown in Figure 2.4(b) and Figure 2.4(c), respectively.

An ADC can be modeled as a series connection of an analog to discrete-time continuous-amplitude (DTCA) converter, a quantizer and a digital encoder as shown in Figure 2.5. The analog to discrete-time continuous amplitude converter transforms the analog input signal, $x(t)$, into the DTCA signal, $x_c[n]$, such that

$$x_c[n] = x(nT),$$

where $T$ is the ADC's sampling period. The quantizer in Figure 2.5 transforms the DTCA signal, $x_c[n]$, into the digital signal, $x_q[n]$, by truncating or rounding $x_c[n]$'s amplitude to the appropriate digital code. The quantization transformation is nonlinear and can be written as

$$x_q[n] = Q(x_c[n])$$

where $Q$ represents the quantization transformation. The coder in Figure 2.5 transforms the digital signal, $x_q[n]$, into a digital code representation.

The flash ADC, also called the parallel ADC, converts an analog signal to a digital signal by comparing the analog signal to a set of references. For example, a typical $B$ bit flash ADC that converts an analog voltage signal contains approximately $2^B$ reference voltages and the same number of voltage comparators. Digital encoder logic determines the ADC output from the comparison results. In particular, Figure 2.6 shows an example

$$x(t) \rightarrow \text{A to DTCA converter} \rightarrow x_c[n] \rightarrow \text{quantizer} \rightarrow x_q[n] \rightarrow \text{digital encoder} \rightarrow x_B[n]$$

Figure 2.5. ADC block diagram.
of a commonly used three bit resistor string flash ADC architecture. This ADC's resistor string generates seven reference voltages, $V_0, V_1, ..., V_6$, where

$$V_i = (i+1) \frac{V_{\text{REF}}}{2^i + 1} \quad i = 0, 1, ..., 6.$$  

The $i$th comparator compares reference voltage, $V_i$, to the analog input voltage, $x(nT)$. If $V_i > x(nT)$, then $t_i$ is logical true else $t_i$ is logical false. The comparators' outputs form the signal $T[n]$, where

$$T[n] = [t_0[n], t_1[n], t_2[n], ..., t_{2^3-1}[n]].$$  

![Three bit resistor string flash ADC](image_url)  

**Figure 2.6.** Three bit resistor string flash ADC.
The encoder converts $T[n]$ into a $B$-bit digital code, such as natural binary, offset binary and two's complement or an application specific code.

2.1.1 Quantization error

In general, an ADC's quantized output, $x_q[n]$, differs from the true sample value, $x[n]$. This difference, called quantization error, denoted $e[n]$ and can be written as

$$e[n] = x_q[n] - x[n].$$

If the quantizer input, $x[n]$, is within range and quantization is performed with rounding, the quantization error is bounded by

$$-\frac{q}{2} \leq e[n] \leq \frac{q}{2},$$

where $q$ is the step size, or code width, of the quantizer. Although the range of quantization error is known, the exact quantization error is typically unknown. However, quantization error can be modelled statistically using following assumptions:

- The distribution of quantization errors is uniform over its range.
- The quantization error sequence, $e[n]$, is a sample sequence of a stationary white random process.
- The quantization error sequence, $e[n]$, is uncorrelated with the quantizer's input sequence.

Although these assumptions are not always valid, these assumptions often lead to accurate predictions of the statistics of the quantization error [55], [56]. It has been shown that the statistical model does not accurately model quantization errors when the quantization code...
width is large or the quantizer’s input sequence is not sufficiently complex, e.g. periodic or constant [29], [77].

If a quantizer performs truncation and the distribution of quantization errors is assumed uniform, then the mean of the quantization error

$$E\{e[n]\} = \frac{q}{2},$$

and quantization error variance, $$\sigma_e^2$$, is

$$\sigma_e^2 = \frac{q^2}{12}.$$  

If a quantizer rounds to the nearest quantization level and the distribution of quantization errors is assumed uniform, then the quantization error is zero mean, i.e.

$$E\{e[n]\} = 0,$$

and quantization error variance, $$\sigma_e^2$$, is

$$\sigma_e^2 = \frac{q^2}{12}.$$  

2.1.2 Dither

To improve the accuracy of the quantization error’s statistical model for periodic deterministic inputs, a dither signal sequence can be added to the ADC’s input. A dither signal is a white noise signal that when added to a periodic deterministic input causes the quantization error to behave like white noise. Figure 2.7 shows an additive dither model.

The quantizer’s input signal, $$x_c[n]$$, is the sum of the discrete time input signal, $$x[n]$$, and the discrete time dither signal, $$v[n]$$, that is, $$x_c[n] = x[n] + v[n]$$. The error, $$e[n]$$, due to quantization and dither is
\[ e[n] = x_q[n] - x[n] = x_c[n] + e[n] - x[n] = v[n] + e[n]. \] (2.3)

Therefore, the total error is the sum of the quantization error and the dither noise. By selecting a dither signal, \( v[n] \), which is uncorrelated from the discrete time input signal, \( x[n] \), the total quantization and dither error power, \( \sigma_e^2 \), can be expressed as

\[ \sigma_e^2 = \sigma_e^2 + \sigma_v^2 = \sigma_e^2 + \frac{q^2}{12}, \]

where \( \sigma_v^2 \) is the variance of the dither signal. To ensure that quantization error can be modelled as additive stationary white noise, an uncorrelated dither signal is added to the quantizer's input. Therefore, the total quantization error is whitened at the expense of adding a small amount of additional noise [27], [47].

Dither signals that have rectangular, triangular and Gaussian probability density functions (PDFs) can eliminate quantization distortion and render the total quantization and dither error in (2.3) equivalent to white noise [27], [47]. To accomplish this, a dither signal should have an amplitude large enough so that it whitens the total quantization and dither error, but not so large that excessive noise is added. Dither signals with amplitudes on the order of a quantization step size are typically sufficient [27].

Detailed analyses of dithered quantizers are provided in [27] and [47], and three commonly used dither signals are mentioned here for convenience. A rectangular PDF dither signal supported on \([-q/2, q/2]\) has a zero mean and a variance given by

Figure 2.7. Additive dither model.
\[ \sigma_v^2 = \frac{q^2}{12}. \]

A triangular PDF dither signal supported on \([-q, q]\) has a zero mean and a variance

\[ \sigma_v^2 = \frac{q^2}{6}. \]

A Gaussian PDF dither signal that has a root-mean-square (RMS) value of \(q/2\) has a variance

\[ \sigma_v^2 = \frac{q^2}{4}. \]

Quantization of full scale sinusoids using the rectangular, triangular and Gaussian dithers described above generate total quantization and dither error equivalent to white noise and add 3 dB, 4.8 dB and 6 dB, respectively, of noise to the quantization error power.

2.2 Mismatched components in ADCs

Many ADC architectures use matched references, comparators and switches to perform their task of signal conversion. ADC typically generate matched references, such as voltages or currents, using matched components such as resistors or transistors. Fabrication process limitations, temperature gradients across the circuit, component aging, and component noise cause circuit component values to differ from their design values. These variations, called mismatch errors, cause inaccurate reference levels that degrade the ADC’s performance. This section briefly discusses some common sources of mismatch errors in ADCs. A more thorough discussion of mismatch component errors can be found in [59] and [70].
2.2.1 Voltage division flash ADCs

A $B$ bit voltage division ADC typically uses matched circuit elements to generate $2^B$ reference voltages across the ADC's input range. The ADC's input voltage, $x(nT)$, is compared with each reference voltage, $V_k$ for $k = 0, 1, \ldots, 2^B - 2$. A comparator's output is logical zero if the ADC's input voltage is less than the comparator's reference voltage, and is a logical one if the ADC's input voltage is greater than the comparator's reference voltage. Digital logic encodes the comparators' outputs, $t_k[n]$ for $k = 0, 1, \ldots, 2^B - 2$, into a digital output signal, $x_d[n]$. For example, the three bit resistor voltage division ADC in Figure 2.6 uses eight resistors to generate seven references voltages, $V_k$ for $k = 0, 1, \ldots, 6$. The ADC's input voltage is compared with all seven reference voltages and digital logic generates the ADC's three bit digital output.

When fabricated with integrated circuit technology, the ADC's voltage divider resistors do not have identical resistance values. Integrated circuit resistors exhibit linear, nonlinear and random mismatch errors. Linear mismatch errors, such as linear gradient errors, are caused by linear variations in doping or resistor widths that occur one end of the voltage divider network to the other. Also, linear mismatch errors are caused by linear thermal gradients across the circuit. Appropriate design and layout techniques can minimize linear gradient mismatch errors caused by process variations [7], [48]. One type of nonlinear mismatch error in diffused resistors is caused by a voltage dependent depletion layer thickness. The depletion layer thickness under diffused resistors is voltage dependent causing diffused resistor resistance to be a function of the square root of the voltage divider voltage [40]. Random mismatches are caused by various random phenomenon, including uncertainties in geometry introduced in fabrication and
processing, random variation in contact resistances, component aging, component noise and thermal noise. Random mismatches due to fabrication variations can be reduced by maximizing the dimensions of the resistor [32], [78]. However, larger dimensions lead to higher parasitic capacitances between the substrate and the resistor and a larger chip area [59].

Voltages along a voltage division ADC’s reference ladder “bow” due to imperfect CMOS and bipolar comparators. Bipolar comparator input stages typically draw a small, constant base current from the voltage divider network, thereby lowering each resistor string tap voltage. Furthermore, base-emitter capacitances provide a high frequency signal path between the ADC input voltage and the tap voltage. ADCs with a large number of comparators have tap voltages which deviate significantly from their designed values [59]. CMOS comparators can also exhibit a conduction path between the tap voltages and the ADC input introducing bowing along the voltage divider. Furthermore, parasitic capacitances between the comparator’s “sampling” capacitor and the substrate aggravate the voltage bowing problem [18].

Both bipolar and CMOS voltage comparators have nonzero input offset voltages due to geometry mismatch in identically drawn transistor pairs. Traditionally, laser trimming and fuse techniques have been used in bipolar comparators to cancel offsets [20]. However, these methods are not easily applied to CMOS circuits. The input offset voltages in CMOS comparators are typically several times larger than those of bipolar comparators. Input offset voltages can be reduced or eliminated with circuits which periodically sense, store, and add the comparator’s offset to the input such that the offset is cancelled [4], [19], [60].
2.2.2 Current mode flash ADCs

A $B$ bit current mode flash ADC typically uses matched components to generate $2^B$ reference currents. Figure 2.8 shows a three bit current mode flash ADC, where a circuit schematic of an unit ADC cell is shown in Figure 2.9. In Figure 2.9, the ADC’s input current, $x(nT)$, is compared with the reference current, $kI_{\text{REF}}$, and the comparators’ output, $t_k[n]$, is

$$t_k[n] = \begin{cases} 1, & x(nT) \geq kI_{\text{REF}} \\ 0, & x(nT) < kI_{\text{REF}} \end{cases}.$$

The schematic for a CMOS current comparator is shown in Figure 2.10 [53]. A comparator’s output is logical zero if the ADC’s input current is less than the comparator’s reference current, and is logical one if the ADC’s input current is greater than the

![Figure 2.8. A $B$ bit current steering flash ADC block diagram.](image-url)
comparator's reference current. Digital logic encodes the comparators' outputs, \( t_k[n] \) for \( k = 0, 1, \ldots, 2^B - 2 \), into a digital output signal, \( x_B[n] \).

Because, matched components are impossible to fabricated transistors do not have identical electrical properties. Matched transistors may exhibit linear gradient and random mismatch errors similar to matched resistors. Mismatches in bipolar current sources are

![Figure 2.9. Current steering flash kth unit ADC cell.](image)

![Figure 2.10. CMOS current comparator.](image)
caused by current gain mismatch errors and emitter resistance mismatch errors. In small geometry bipolar devices, the emitter resistance is large and can vary considerably between devices. Although increasing the size of bipolar transistors reduces device mismatches, large collector-substrate and collector-base capacitances are created. These large capacitances slow the ADC’s switching speed [59]. Similar to bipolar current sources, mismatches in CMOS current sources can be reduced by increasing the transistors’ widths and lengths. However, larger transistor widths increase drain-substrate and gate-drain capacitances which, in turn, slow the ADC’s operation. Large transistor lengths require higher gate-source voltages to obtain a given current. Short channel device mismatches do not respond to changes in device length. Source degeneration resistors can be used in MOS current sources, but require matched resistors comparable with the inverse of the device’s transconductance [58].

The integrating nature of the current comparator in Figure 2.10 minimizes the comparator’s input offset [53]. However, careful circuit design and layout techniques must be used to minimize transistor geometry mismatch errors. Also, current comparator autozeroing techniques comparable to those of voltage comparators do not currently exist and are an open area of research.

2.3 Digital to analog conversion and flash DACs

An ideal $B$ bit DAC transforms a $B$ bit digital signal, $x[n]$, into an analog signal, $y(t)$, such that $y(nT) = qx[n]$ where $q$ is a constant, $T$ is the DAC’s sampling period and $n$ is an integer that indexes the sequence $x$. Figure 2.11 shows an example of an ideal
unipolar three bit DAC transfer function. Unlike an ideal ADC, an ideal DAC is piece wise linear.

An DAC can be modeled by the system shown in Figure 2.12. The input to the DAC in Figure 2.12 is a sequence of finite values, \( x[n] \), e.g. quantized samples obtained from a ADC. In Figure 2.12, the "convert to impulses" block transforms the digital input signal, \( x[n] \), into a train of Dirac delta functions, \( \delta_a(t) \), such that

\[
x_y(t) = \sum_{n=-\infty}^{\infty} x(nT) \delta(t-nT)
\]

The signal, \( x_y(t) \), is an analog function that is nonzero only at the sample times \( nT \). The zero order hold maintains the value of its input between samples instances. By processing

![Figure 2.11. Ideal unipolar three bit DAC transfer function.](image)

![Figure 2.12. DAC block diagram.](image)
the impulse train, \( x_c(t) \), with the linear time-invariant zero order hold system that has the impulse response in (2.1), the SHA's output, \( x_o(t) \), can be written as \( x_o(t) = x_a(t) * h_{zoh}(t) \).

Because the zero order hold is a linear time-invariant system,

\[
x_{zoh}(t) = \sum_{n=-\infty}^{\infty} x[n] h_{zoh}(t-nT).
\]

It can be shown that the zero order hold in (2.1) has the frequency response, \( H_{zoh}(j\Omega) \), where

\[
H_{zoh}(j\Omega) = \frac{2\sin(\Omega T/2)}{\Omega} e^{-j\Omega T/2}.
\]  \quad (2.4)

The reconstruction filter, also called the interpolation filter, is a lowpass filter providing frequency support below \( \Omega = \pi/T \). The nonunity gain frequency response in (2.4) can be corrected digitally or in the reconstruction filter. For example, the ideal reconstruction filter that compensates for the zero-order hold in (2.1) is

\[
H_{r}(j\Omega) = \begin{cases} 
\frac{\Omega T/2}{\sin(\Omega T/2)} e^{j\Omega T/2} & |\Omega| < \pi/T \\
0 & |\Omega| > \pi/T
\end{cases}.
\]

The system model in Figure 2.12 has been implemented using architectures which include the counter ramp, successive approximation, multistep, delta-sigma modulators and flash [31], [59], [65]. An ideal \( B \) bit flash, or parallel, DAC performs its conversion by encoding the input, \( x[n] \), into a thermometer code, \( r[n] \). Figure 2.13 shows an example \( B \) bit flash DAC. The \( 2^B \) thermometer coded bits control the \( 2^B \) unit DACs which have an output of \( q \) when activated and an output of zero when deactivated. The unit DAC outputs, \( y_k(nT) \) for \( 1 \leq k \leq 2^B \), are summed to produce the DAC output, \( y(nT) \).
2.4 Mismatch component errors in DACs

Many DAC architectures use matched references, amplifiers and switches to perform signal conversion. DAC references are typically voltages or currents generated by matched components, such as resistors, transistors, and capacitors. Fabrication process variations, temperature gradients across the circuit, component aging, and component noise cause circuit component values to differ from their design values. These variations, or mismatch errors, cause inaccurate output levels such that the DAC’s output contains harmonic distortion. This section briefly introduces some common sources of mismatch errors in DACs. A more thorough discussion of mismatch component errors can be found in [59] and [70].

2.4.1 Voltage division flash DACs

A $B$ bit voltage division DAC typically uses $2^B$ or more matched circuit elements to generate $2^B$ voltages. One of these voltages is selected as the DAC’s analog output by the DAC’s digital input signal or some function of the DAC’s digital input signal. Resistor string DACs and charge scaling DACs use resistors and capacitors, respectively, to perform voltage division and are the most common voltage division DACs [31], [59].

![Figure 2.13. A $B$ bit flash DAC architecture.](image-url)
2.4.1.1 Resistor string flash DACs

A $B$ bit resistor string flash DAC typically uses $2^B$ or more matched resistors to generate $2^B$ equally spaced voltages, $V_k$ for $k = 0, 1, \ldots, 2^B - 1$, where

$$V_k = \frac{k+1}{2^B} V_{\text{REF}}.$$

Figure 2.14 shows a three bit resistor voltage division flash DAC architecture. In Figure 2.14(a), $2^B$ switches connect the appropriate voltage to the DAC output, $y(t)$. The switch control signals, $S_k$ for $k = 0, 1, \ldots, 2^B - 1$, are generated by a $B:2^B$ decoder (not

Figure 2.14. Three bit resistor string flash DAC architecture (a) requiring a decoder to generate switch control signals. (b) Three bit resistor string flash DAC architecture using tree structured switches providing inherent decoding.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
shown). As the DAC’s word length increases, the DAC’s output has a large parasitic capacitance, which limits the DAC’s operating speed [31]. Figure 2.14(b) shows an alternative resistor string DAC architecture that arranges the switches into a binary tree structure. This architecture does not require a dedicated decoder and uses the DAC’s digital input bits, $x_k[n]$, and their compliments, $\bar{x}_k[n]$, for $k = 0, 1, \ldots, 2^B - 1$, to control the switches. Furthermore, parasitic capacitances are reduced since the output is connected to at most $B$ closed switches and $B$ open switches. Thus, the DAC architecture in Figure 2.14(b) can operate at higher speeds.

The resistor string flash DAC architecture in Figure 2.14 requires resistor matching to within the resolution of the DAC. This matching requirement is difficult to achieve for DAC’s larger than eight bits [59]. The resistor string flash DAC architecture in Figure 2.14 cannot drive resistive loads without a buffer. If appreciable current is drawn from the voltage divider network, additional errors will be introduced due to the nonlinearity of the DAC’s analog switches. Therefore, the output of a voltage division DAC is usually buffered by an amplifier with high input impedance. However, amplifier nonlinearities are introduced directly into the DAC’s output [59].

The resistor string DAC architecture in Figure 2.14 is susceptible to the same component matching errors, i.e. linear gradient errors due to variations in doping density or fabricated resistor widths, nonlinear errors in diffused resistors from nonuniform depletion layer thickness, random errors due to geometry uncertainties, random contact resistances, component noise, and component aging, as the ADC voltage divider networks in Section 2.2.1.
2.4.1.2 Charge scaling flash DACs

Charge scaling flash DACs perform their task of signal conversion by dividing a DAC's reference voltage, $V_{\text{REF}}$, using $B$, $2^B$, or more matched capacitors. Figure 2.15 shows a three bit charge scaling DAC architecture. Initially, each capacitor is discharged by the "reset" switch. After the capacitors have been discharged, each capacitor is connected to either $V_{\text{REF}}$ or ground. This causes the DAC output voltage, $y(t)$, to be a function of the voltage division between the capacitors. Figure 2.15(a) shows a charge scaling flash DAC architecture that uses $2^B$ switches to connect the appropriate number of equally sized capacitors to $V_{\text{REF}}$ and the remaining capacitors to ground. The switch control signals, $t_k$ for $k = 0, 1, \ldots, 2^B-1$, are generated by a thermometer encoder (not

Figure 2.15. Three bit charge scaling flash DAC architecture (a) with unary weighted capacitors and (b) binary weighted capacitors.
shown). Figure 2.15(b) shows a charge scaling flash DAC architecture that uses \( B \) switches to connect the appropriate combination of binary weighted capacitors to \( V_{\text{REF}} \), thereby creating the DAC output voltage, \( y(t) \). Unlike the DAC in Figure 2.15(a) that uses a thermometer encoder, the architecture in Figure 2.15(b) uses the DAC's digital input bits, \( x_k[n] \), and their compliments, \( *x_k[n] \), for \( k = 0, 1, ..., 2^B - 1 \), to control the switches.

Because the capacitor arrays in Figure 2.15 cannot supply current, both architectures in Figure 2.15 require buffers to drive loads [59]. Other limitations of the charge scaling DAC architectures are the large transient currents drawn from \( V_{\text{REF}} \) during switching [59] and the need for precisely matched capacitors [7]. Furthermore, parasitic capacitances at the buffer input limit the resolution of the charge scaling DAC architectures [7].

Mismatches between capacitors, capacitor voltage dependence, and top plate parasitic capacitances cause charge scaling DACs to be nonlinear. Capacitor geometry mismatch errors can be vary linearly across the array and randomly. Geometric mismatches are functions of capacitor width, length and oxide thickness. Oxide thickness is a function of the fabrication process and oxide thickness gradients can become significant for large capacitors. Therefore, increasing capacitor dimensions does not necessarily reduce mismatch error indefinitely. Minimum capacitor mismatch error is obtained by enlarging the capacitors to a size which is process specific [68]. Also, common centroid layout techniques can be used to improve the capacitor matching [51].

Capacitor voltage dependence originates from the variation of the dielectric constant across capacitors and the depletion region thickness of each capacitor plate [51]. Derivations of the capacitor voltage dependence for binary and unary weighted capacitor arrays are given in [44] and [59], respectively.
The top plate of the capacitor array has an appreciable parasitic capacitance to the substrate. This parasitic capacitance introduces a gain error over the full scale range of the DAC. While the gain error is easily ignored or corrected in stand-alone DACs, it creates differential nonlinearities in multistep ADCs or sigma-delta modulators [59].

2.4.2 Current steering flash DACs

A $B$ bit current steering flash DAC typically uses matched circuit elements to create $B$, $2^B$ or more reference currents. These reference currents are directed by analog switches and summed to create the DAC's output. Figure 2.16 shows a three bit current steering DAC architecture. The DAC architecture in Figure 2.16(a) uses $2^B$ switches to

![Diagram of three bit current steering flash DAC architecture with unary weighted current sources and binary weighted current sources.]

Figure 2.16. Three bit current steering flash DAC architecture (a) with unary weighted current sources and (b) binary weighted current sources.
connect the identical current sources to the DAC output. These currents are summed creating the DAC output, \( y(t) \). The switch control signals, \( t_k \) for \( k = 0, 1, \ldots, 2^B - 1 \), are generated by a thermometer encoder (not shown). Figure 2.16(b) shows a current steering flash DAC architecture that uses \( B \) switches to connect binary weighted reference currents to the DAC output. By selecting the appropriate combination of currents, the DAC output current, \( y(t) \), is formed. Although the DAC architecture in Figure 2.16(a) requires a thermometer encoder, the architecture in Figure 2.16(b) uses the DAC’s digital input bits, \( x_k[n] \) for \( k = 0, 1, \ldots, 2^B - 1 \), to control the switches.

A major advantage of the current steering DAC architecture in Figure 2.16 is their inherent high current drive and high speed [59]. However, these current steering DAC architecture creates “glitches” when the switches do not change states in unison. Since the current sources are in parallel, if one source is switched off and another source switched on, a “glitch” occurs if the timing is such that both sources are off or both sources are on at the same instant. This error is most significant at the DAC’s midscale when the largest number of sources are switching [59].

Current steering flash DACs accuracy is limited by mismatch error between current sources [7]. DAC current replication exhibits the same matching errors, i.e. linear gradient errors, random errors due to geometry uncertainties, component aging and component noise, as the current mode ADCs in Section 2.2.2. Additional sources of error in current steering DACs are finite output impedance of the current sources and the DAC’s load resistor nonlinearity. As the DAC output varies over its full scale range, different impedances are connected to the DAC output changing the load resistance and introducing nonlinearity. Furthermore, many current steering DACs convert the current output to a
voltage by connecting the DAC's output node to an integrated circuit resistor. Polysilicon resistors have a hyperbolic sine current-voltage characteristic which becomes more linear as the resistor length increases [48]. Integrated circuit diffusion resistors are nonlinear because their depletion region thickness is a function of voltage [40].

2.5 Data converter metrics and terminology

Because of the diverse nature of the data converter community, definitions and interpretations of many data converter specifications are not well standardized. Therefore, data converter metrics and terminology used in this dissertation are defined in this section.

2.5.1 Code

The code is the format and number system used to represent values digitally.

Popular linear codes include natural binary, binary coded decimal, offset binary, two's complement, gray, and thermometer codes. Complementary versions of these codes are also commonly used. Other special purpose codes, such as companding codes and mu-law codes, are used in application specific converters, such as speech coding data converters. Table 2.1 shows examples of a three bit natural binary, gray, and thermometer codes.

Typically, thermometer codes are used internally in many ADC and DAC architectures. The thermometer code is so named because its operation resembles a mercury thermometer. For example, consider Figure 2.6 when $V_{j-1} \geq x(t) \geq V_j$. For this case, the outputs of the comparators $t_0$, $t_1$, ..., $t_{j-1}$, $t_j$ are logical ONEs, while the outputs $t_{j+1}$, $t_{j+2}$, ..., $t_{2^n-1}$, are logical ZEROs. Therefore, as $x(t)$ increases and
Table 2.1. Correspondence among natural binary, gray, and thermometer codes.

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Binary</th>
<th>Gray</th>
<th>Thermometer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X3 X2 X1</td>
<td>G3 G2 G1</td>
<td>T7 T6 T5 T4 T3 T2 T1</td>
</tr>
<tr>
<td>0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>1</td>
<td>0 0 1</td>
<td>0 0 1</td>
<td>0 0 0 0 0 0 0 1</td>
</tr>
<tr>
<td>2</td>
<td>0 1 0</td>
<td>0 1 1</td>
<td>0 0 0 0 0 1 1 1</td>
</tr>
<tr>
<td>3</td>
<td>0 1 1</td>
<td>0 1 0</td>
<td>0 0 0 0 1 1 1 1</td>
</tr>
<tr>
<td>4</td>
<td>1 0 0</td>
<td>1 1 0</td>
<td>0 0 0 1 1 1 1 1</td>
</tr>
<tr>
<td>5</td>
<td>1 0 1</td>
<td>1 1 1</td>
<td>0 0 1 1 1 1 1 1</td>
</tr>
<tr>
<td>6</td>
<td>1 1 0</td>
<td>1 0 1</td>
<td>0 1 1 1 1 1 1 1</td>
</tr>
<tr>
<td>7</td>
<td>1 1 1</td>
<td>1 0 0</td>
<td>1 1 1 1 1 1 1 1</td>
</tr>
</tbody>
</table>

decreases, the comparators’ output thermometer code rises and falls, similar to a mercury thermometer.

2.5.2 Full scale reference

An data converter’s full scale reference is the constant, typically a voltage or current generated by an external source, which determines the converter’s maximum operating point.

2.5.3 Zero scale reference

An data converter’s zero scale reference is the constant, typically a voltage or current generated by an external source, which determines the converter’s minimum operating point.
2.5.4 Full scale input

An ADC's full scale input is the maximum analog input which can be applied to the ADC without clipping.

For many ADCs, full scale input is one quantization step size less than the ADC's full scale reference. In an ADC with natural binary digital output, a full scale input generates a digital output of all ones. Figure 2.17 shows the full scale input for a three bit ideal ADC.

2.5.5 Full scale output

A DAC's full scale output is its maximum analog output.

For many DACs with a natural binary coded digital input, full scale output occurs when the digital input is all ones. In many DACs, full scale output is one quantization step size less than the the DAC's full scale reference. Figure 2.18 shows the full scale output for a three bit ideal DAC.

![Figure 2.17. ADC transfer function terminology.](image)
2.5.6 Zero scale input

The zero scale input, or minimum scale input, of an ADC is its minimum analog input which can be applied to the ADC without clipping.

Zero scale input is ideally ground for unipolar ADCs and negative full scale input for bipolar ADCs. In an ADC with natural binary digital output, zero scale input generates a digital output of all zeros. Figure 2.17 shows the zero scale input for a three bit ideal ADC.

2.5.7 Zero scale output

The zero scale output, or minimum scale output, of a DAC is its minimum analog output.

For a DAC with natural binary coded digital input, zero scale output occurs when the digital input is all zeros. Ideally, zero scale output is ground for unipolar DACs and negative full scale for bipolar DACs. Figure 2.18 shows the zero scale output for a three bit ideal DAC.

![Figure 2.18. DAC transfer function terminology.](image)
2.5.8 Full scale range

The full scale range, input range or span, of an ADC is the difference between its full scale and zero scale inputs.

The full scale range, also called output range and span, of a DAC is the difference between its full scale and zero scale outputs.

Figure 2.17 shows the full scale range for a three bit ideal ADC. Figure 2.18 shows the full scale range for a three bit ideal DAC.

2.5.9 Resolution

Resolution is the number of partitions into which the full scale range is divided.

The resolution of a $B$ bit ADC or a $B$ bit DAC is $2^B$. Resolution is sometimes reported as the base two logarithm of the number of analog levels, that is the data converter's word length. For example, a 10 bit ADC has a resolution of 1024 levels or 10 bits.

2.5.10 Code width

ADC code width, or quantization step size, is the change in the ADC's analog input which causes a change in the ADC's digital output.

DAC code width, or quantization step size, is the difference in the DAC's analog output when the DAC's digital input changes from one code to an adjacent code.

For an ideal $B$ bit ADC, the code width is the ADC's full scale range divided by the number of digital input codes, $2^B$. Figure 2.17 shows the code width for a three bit ideal ADC. A DAC's code width is the DAC's full scale range divided by the number of digital
input codes. For example, a 10 bit ADC with a zero scale input of zero volts and a full scale input of four volts has a code width of \((4\text{V} - 0\text{V})/1024 = 3.9062\text{mV}\). Figure 2.18 shows the code width for a three bit ideal DAC.

2.5.11 Dynamic range

A data converter’s dynamic range is the ratio of its full scale range to its code width.

An ideal \(B\) bit converter has dynamic range equal to \(2^B\). Dynamic range is typically expressed in decibels (dB). For example, a 10 bit ADC with a zero scale input of zero volts and a full scale input of four volts has a dynamic range of \((4\text{V} - 0\text{V})/3.9062\text{mV} = 1024\), or equivalently in decibels as \(20\log_{10}1024 = 60\text{dB}\).

2.5.12 Quantization error

Quantization error in ADCs is defined as the difference between the ADC’s analog input value and the ADC’s ideal transfer point, where the ideal transfer point is defined as the midpoint of the analog values represented by a particular digital output code.

In ADCs, the analog full scale range is partitioned into \(2^B\) discrete ranges for that are represented by the ADC’s \(2^B\) digital output codes. Any analog value within a particular partition produces the same output code. Thus, the ADC analog input value which causes a particular digital output code is not known exactly. The difference between the ADC’s input value and the midpoint of the analog values represented by the input’s digital code is the quantization error. An ADC with an infinite number of bits can represent its analog input value perfectly and has no quantization error. Figure 2.19 shows an ideal three bit ADC transfer function and the corresponding quantization error.
2.5.13 ADC full scale error

An ADC's full scale error, also called scale error or gain error, is the difference between the ideal full scale input, i.e. one quantization step size less than the ADC's full scale reference, and the ADC's actual analog input.

Many ADCs have full scale error adjustment, so that full scale error can be made zero. Full scale ADC error is caused by comparators offset errors, amplifier gain errors, and errors in ADC references [31], [59].

![Figure 2.19. The quantization error characteristic for an ideal three bit ADC.](image)
2.5.14 DAC full scale error

A DAC's full scale error is the difference between the DAC's ideal full scale output value, i.e. one quantization step size less than the the DAC's full scale reference, and the DAC's actual full scale output.

Full scale DAC error is caused by amplifier gain error, nonideal switches, and errors in DAC references [31], [59].

2.5.15 ADC zero offset error

ADC zero offset error, zero scale error or zero code error, is the difference between the ADC's ideal zero scale, i.e. ground for unipolar ADCs and negative full scale input for bipolar ADCs, and the ADC's actual zero scale input.

Many ADCs have zero offset error adjustment so that zero offset error can be made zero. ADC zero offset errors are usually caused by nonzero input-offset voltage or input-offset currents in the ADC's amplifiers or comparators [31], [59].

2.5.16 DAC zero offset error

DAC zero offset error is the difference between the DAC's ideal zero scale output, i.e. ground for unipolar DACs and negative full scale for bipolar DACs, and the DAC's actual zero scale output.

DAC zero offset errors are usually caused by nonideal switches and nonzero input-offset voltages or input-offset currents in the DAC's amplifiers. [31], [59]
2.5.17 Signal to noise ratio

A converter's signal to noise ratio (SNR) is defined as the ratio of the converter's output signal power to its output noise power.

A converter's SNR depends on its resolution, i.e. higher resolution implies more quantization levels which reduces the quantization error and the quantization error power. For a single sinusoidal input, the output signal power is the root-mean-square (RMS) magnitude of the fundamental and the noise is the RMS sum of all non fundamental signals. The theoretical SNR for an ADC with full-scale sinusoidal input is given by

\[ \text{SNR} = 6.02B + 1.76 \text{ dB}, \]

where \( B \) is the number of bits [55].

2.5.18 Effective number of bits

A converter's effective number of bits (ENOB) is defined as the apparent number of bits that the converter possesses based on an observation of the converter's SNR.

For a single sinusoidal input, the SNR can be expressed in terms of the number of bits, \( B \). Rewriting in terms of the number of bits, it is possible to get a measure of performance expressed in terms of the ENOB, i.e.

\[ \text{ENOB} = \frac{\text{SNR} - 1.76}{6.02} \] [31].

2.5.19 Linearity error

Linearity error is a measure of the difference between an ideal transfer function and the converter's actual transfer function.

ADC linearity error does not include quantization, zero offset, gain offset, or scale errors. DAC linearity error does not include zero offset, gain offset, or scale errors. A
The converter's linearity can be measured against one of two ideals, endpoint straight line fit and best fit straight line. Each ideal is a different perfectly linear transfer function.

2.5.20 Straight line transfer functions

An endpoint straight line fit compares the converter's actual transfer function with an ideal transfer function determined by the converter's zero and full scale inputs. The endpoint straight line transfer function is the straight line between the midpoints of the converter's zero scale and full scale values.

The best fit straight line fit compares the actual converter transfer curve with an ideal transfer function found by minimizing a cost function, typically RMS error. Therefore, the best fit straight line transfer function represents straight line which "best" approximates the converter's transfer function. An endpoint straight line transfer function for a nonideal ADC is shown in Figure 2.20.

![Figure 2.20. Nonlinearity metrics for ADCs.](image-url)
2.5.21 Integral nonlinearity

A converter's integral nonlinearity (INL), also called integral linearity error, is defined as the difference between the converter's transfer function and the converter's straight line transfer function.

INL is the single most important measure of a converter's linearity. Many data converter data sheets only report the maximum value of the INL curve. Figure 2.20 shows the maximum INL of a nonideal three bit ADC. Figure 2.21 shows the maximum INL of a nonideal three bit DAC.

2.5.22 Differential nonlinearity

A converter's differential nonlinearity (DNL), also called differential linearity error, is defined as the difference between the converter's ideal code widths and the converter's actual code widths.

![Figure 2.21. Nonlinearity metrics for DACs.](image-url)
Adjacent digital codes should be associated with analog values which differ by an ideal code width, i.e. full scale range divided by the number of codes. The difference between the converter's actual code widths and the ideal code widths is DNL. Many data converter data sheets only report the maximum value of the DNL curve. Figure 2.20 shows the maximum DNL for a nonideal three bit ADC. Figure 2.21 shows the maximum DNL for a nonideal three bit DAC.

2.5.23 Harmonic distortion

A converter's harmonic distortion is the ratio of the power of a second and higher harmonics in the converter's output to the power of the fundamental of the converter's sinusoidal input.

When an ideal converter's input is a sinusoid, the output spectrum contains energy at the input's fundamental frequency and possibly at DC. When a nonideal converter's input is a sinusoid, spectral energy is also located at frequencies other than that found in the output of a linear DAC. Harmonic distortion is another measure of converter nonlinearity.

2.5.24 Spurious free dynamic range

When an ideal converter's input is a sinusoid, the output spectrum contains energy at the input's fundamental frequency and possibly at DC. When a nonideal converter's input is a sinusoid, spectral energy is also located at frequencies other than that found in the output of an ideal converter. The difference in amplitude between the largest of these spectral components, called spurs, and the amplitude of the input's fundamental is called spurious free dynamic range (SFDR).
2.6 Principles of DEM in ADCs

Analog DEM algorithms were introduced in 1974 by Klaas Klaassen who used DEM to obtain a constant division ratio from a voltage divider consisting of a mismatched resistors [38]. Early DEM was performed using mechanical or analog switches [38], [72], [73]. Klaassen's algorithm improved reference voltage accuracy in measurement instruments. Today, the algorithm is called clocked averaging DEM. Clocked averaging DEM is so named because the resistors in a voltage divider network are cyclically shifted into each resistor position. Although early DEM algorithms were used to reduce the effects of resistor mismatch errors, DEM algorithms work equally well at reducing or eliminating mismatch errors of other circuit elements, such as transistors and capacitors.

2.6.1 Analog DEM algorithms in ADCs

Many ADC architectures use comparators, switches, and matched references for signal conversion. Matched references in ADCs typically generate voltages or currents using matched components, such as resistors, capacitors or transistors. Current manufacturing processes cannot generate components that have identical electrical properties. Differences between identical circuit components, called mismatch errors, cause inaccurate quantization levels such that the ADC's output, $y[n]$, is $y[n] = x[n] + e[n]$, where $x[n]$ is the ideal ADC output and $e[n]$ is a sequence representing ADC conversion errors. The conversion error sequence, $e[n]$, contains harmonic distortion that reduces an ADC's SFDR, SDR, SNDR and ENOB.

An $N$ element clocked averaging DEM network cyclically shifts each element in the network chain one position in the same direction every time interval, $T$ [13], [38], [70].
After $NT$ cycles, each mismatched element will have occupied each of the $N$ network positions. Rotating the mismatched elements creates a dynamic network where the time average of each element value is equal to the average of the mismatched components’ values.

To illustrate, consider the clocked averaging voltage divider in Figure 2.22(a). The voltage divider has $N$ resistor positions labeled by $j = 1, 2, \ldots, N$ occupied by $N$ resistors, $R_i$, for $i = 1, 2, \ldots, N$. The reference voltage, $V_{\text{REF}}$, in Figure 2.22 is fixed. Using electronic switches which are not shown in Figure 2.22, the voltage divider resistors are sequentially rotated from position to position. After one rotation, the resistor chain in Figure 2.22(a) will appear as illustrated in Figure 2.22(b). After $NT$ cycles, each resistor will have occupied each of the $N$ branches of the chain.

![Diagram](image)

Figure 2.22. Voltage divider demonstration of clocked averaging DEM. (a) voltage divider network with $N$ elements, (b) voltage divider network after one shift and (c) virtual voltage divider network constructed with average resistance, $\bar{R}$. Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
The voltage, $V_{j,i}$, across the resistor in the $j$th resistor location when the resistor $R_i$ is occupying it is

$$V_{j,i} = IR_i$$

where $I$ is the current flowing through the resistors. After $NT$ cycles, each resistor has occupied each of the $N$ resistor positions and the average voltage, $\overline{V}_j$, at the $j$th resistor position is

$$\overline{V}_j = \frac{1}{N} \sum_{i=1}^{N} V_{j,i} = I\left(\frac{1}{N} \sum_{i=1}^{N} R_i\right)$$

(2.5)

Because the voltage divider’s reference voltage, $V_{\text{REF}}$, and the values of the resistors are constant, (2.5) becomes

$$\overline{V}_j = \frac{1}{N} \sum_{i=1}^{N} V_{j,i} = I\overline{R}$$

(2.6)

where $\overline{R}$ is the average resistance of the mismatched resistors, i.e. $\overline{R} = \frac{1}{N} \sum_{i=1}^{N} R_i$. On average, the voltage divider in Figure 2.22(a) is equivalent to the virtual static voltage divider shown in Figure 2.22(c) where each resistor has resistance $\overline{R}$ and $V_1 = V_2 = \ldots = V_N$. Thus, on average, clocked averaging DEM creates a perfect virtual voltage divider from a voltage divider with mismatched resistors. Clocked averaging DEM performs a similar operation on other circuit elements, such as transistors and capacitors.

In practice, analog DEM algorithms, such as the clocked averaging algorithm, will not create perfect components because their switches will not be matched [38]. For example, FET switches exhibit a small amount of resistance, $r_o$, when they are conducting. Due to process variations, these resistances will vary from FET to FET resulting in switch
mismatches. Other types of switches manifest similar problems. Errors due to switch mismatches can be minimized with careful switch designs and the use of dummy switches.

Another source of error that can be attributed to analog DEM results from imperfect averaging caused by imperfect clocks. Typically, the switches in an analog DEM network are controlled by digital control signals. If each resistor does not occupy each position for an equal length of time, the average resistance for each resistor position will not be equal. However, this error is largely avoidable, and resistor mismatch errors can be significantly reduced if the digital control circuitry is carefully designed.

Voltage spikes, which occur whenever analog DEM networks are dynamically reconfigured, are another source of error. However, voltage spikes are often reduced by parasitic capacitances or by explicit capacitors added by the circuit designer [38].

2.6.2 Digital DEM algorithms in ADCs

Analog DEM algorithms rearrange mismatched elements by using electronic or mechanical switches. Digital DEM algorithms rearrange mismatched elements by reordering the bits of the digital inputs to the mismatched elements. Digital DEM algorithms use signal processing algorithms and interconnection networks to virtually permute the mismatched components. To date, digital DEM algorithms in ADCs have been used in multibit oversampling ADCs, and only in the multibit oversampling ADC's feedback path DACs [6], [15], [22], [34], [46]. When multibit DACs are used in multibit oversampling ADCs, the multibit oversampling ADCs have fewer stability problems and lower oversampling requirements when compared to single bit oversampling ADCs [28]. Without the digital DEM algorithms, feedback path DACs in multibit oversampling
converters would not have the needed accuracy for oversampling converter architectures [15], [22].

2.7 Principles of DEM in DACs

Many DAC architectures use switches and matched components to convert signals. In practice, perfectly matched components are not possible due to fabrication limitations, thermal gradients, component aging, and component noise. The difference between the actual component value and the ideal component value is called mismatch error. In DACs, mismatch errors cause conversion errors in the DAC's analog output signal, \( y(t) \), such that

\[
y(nT) = qx[n] + e[n]
\]

where \( e[n] \) is a sequence representing DAC conversion errors. This conversion error sequence, \( e[n] \), contains the DAC's distortion that reduces the DAC's SFDR, SDR, SNDR and ENOB. Thus, a DAC's distortion can be reduced and its SFDR, SDR and SNDR increased by decreasing component mismatch errors in the DAC circuitry.

To demonstrate the principles of DEM in DACs, consider the \( B \) bit DEM DAC topology in Figure 2.23 [15]. This topology performs DEM by mapping a \( B \) bit input

![Figure 2.23](image-url)

**Figure 2.23.** A \( B \) bit dynamic element matching DAC architecture [15].

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
signal, $x[n]$, to $2^B$ single bit DACs through a $2^B$ line interconnection network. In this topology, a thermometer encoder converts the $B$ bit binary coded signal, $x[n]$, into a $2^B$ bit modified thermometer coded signal, $t[n]$. Without the interconnection network, the modified thermometer coded signal, $t[n]$, activates $x[n]$ particular single bit DACs. Ideally, each deactivated single bit DAC generates an analog signal of amplitude zero, and each activated single bit DAC generates an analog signal of amplitude $q$. The outputs of all the single bit DACs are summed to produce the DAC's output, $y(nT)$, where $y(nT) = qx[n]$. In practice, mismatched unit DAC component values cause deterministic conversion errors that cause harmonic distortion and reduce the DAC's SFDR, SDR, and SNDR. Using the interconnection network to randomize the mapping between the thermometer coded signal, $t[n]$, and the array of unit DACs, the positions of mismatched unit DACs can be virtually altered. With a deterministic DEM interconnection network, the shuffled thermometer coded signal, $g[n]$, activates $x[n]$ single bit DACs chosen according to a deterministic algorithm. As a result of this virtual rearrangement of unit DACs, the mismatched components generate harmonic distortion in particular frequency bands. This distortion is reduced or removed in subsequent processing, typically lowpass filtering in $\Delta\Sigma$ architectures. With a stochastic DEM interconnection network, the shuffled modified thermometer coded signal, $g[n]$, activates $x[n]$ single bit DACs chosen at random. As a result of this virtual rearrangement of unit DACs, the mismatched components generate noise instead of harmonic distortion. Regardless of the DEM algorithms, the mismatched errors, and the interconnection network, the DEM DAC's SFDR, SDR and SNDR can be increased.
Presently, the term “dynamic element matching” includes the concept of element randomization as well as element shifting. While early DEM algorithms were designed to shift resistors, DEM algorithms can be used to reduce the mismatch error of many different elements, including capacitors, transistors, and even complex structures, including current sources. Several DAC DEM algorithms have been proposed, including clocked level averaging [72], [73], individual level averaging [16], [46], data weighted averaging [6], and stochastic level averaging or stochastic DEM [15], [23].

2.7.1 Clocked level averaging DEM algorithm

Clocked level averaging rotates or flips the order of matched components in a periodic fashion. Clocked level averaging in DACs is often implemented with barrel shift registers. In flash DEM ADC and DAC architectures, clocked level averaging causes harmonic distortion at beat frequencies which are a function of the sampling frequency and the element rotation frequency [70]. When a clocked level averaging DEM DAC is used in an multibit oversampling ADC, the correlation between the clocked level averaging and the averaging action inherent to the ΔΣ modulator causes harmonic distortion [16]. It has been shown that clocked level averaging DEM improved the accuracy of a 12 bit DAC to less than 1/4 LSB. Furthermore, clocked level averaging also reduced the DAC’s sensitivity to process variations and component aging [72]. Similar results have been obtained in a 14 bit clocked level averaging DEM DAC [73].
2.7.2 Individual level averaging DEM algorithm

Individual level averaging DEM algorithm is essentially clocked level averaging performed on a per-level basis. Individual level averaging rotates or flips circuit elements in a periodic fashion, but a separate rotation state is maintained for each digital level. The advantage of the individual level averaging algorithm over the clock level averaging algorithm is that distortion is moved into higher frequency bands than clocked level averaging while preserving the noise shaping characteristics of the modulator in ΔΣ converters [16], [46].

2.7.3 Data weighted averaging DEM algorithm

The data weighted averaging DEM algorithm also rotates circuit elements, but does so at the maximum possible rate while insuring that each element is used the same number of times. The circuit elements are selected sequentially from the array starting with the next available unused element. The circuit components are used at the maximum possible rate, causing the mismatch errors to sum to zero more quickly. Simulations have shown that the distortion is shifted to higher frequencies than both clocked level averaging and individual level averaging algorithms. The data weighted averaging DEM algorithm preserved the noise shaping characteristics of the modulator in ΔΣ converters [6].

2.7.4 Stochastic level averaging DEM algorithms

While the deterministic DEM algorithms, such as clocked level averaging, individual level averaging and data weighted averaging, concentrates the harmonic distortion in higher frequency bands, stochastic level averaging spreads the mismatch error energy
across the spectrum [15], [23]. Stochastic level averaging, or stochastic DEM, randomly permutes all unit DAC elements each sample, creating noise instead of harmonic distortion.
CHAPTER 3

ANALOG TO DIGITAL CONVERTER TRANSFER FUNCTION ERRORS AND HARMONIC DISTORTION

ADC errors reduce an ADC's performance. Errors can be caused by timing uncertainties, called clock jitter, and circuit errors that create ADC transfer function errors. An ADC that has errors caused by clock jitter can be modelled as an ADC that has nonuniform or random sampling. Using these models, it has been shown that clock jitter creates noise in the ADC output [33], [69]. As a result, clock jitter errors are typically treated as additive white noise, much like quantization errors. The increase in the noise floor due to ADC clock jitter is proportional to the variance of the timing uncertainty [5], [74].

Unlike clock jitter errors that are effectively random, ADC transfer function errors are static and cause distortion in the ADC output. Furthermore, at slower clock speeds, ADC transfer function errors due to mismatched components are significantly larger than ADC clock jitter errors [37]. Increasingly, bulk CMOS processes are used to fabricate ADCs; bulk CMOS processes create circuit components which have values that vary significantly from their design values [40], [51], [58], [68]. Many ADC architectures use switches and matched components to convert signals. Error between matched components cause the ADC to have nonuniform quantization step sizes which causes transfer function errors. As
a result, mismatch errors cause distortion in the ADC's output [15], [31], [59], [65]. This reduces an ADC's performance.

Because an ADC's transfer function is inherently nonlinear, analog to digital conversion creates harmonic distortion. Harmonic distortion resulting from the quantization operation can be effectively eliminated by adding a small random signal, called dither, to the ADC's input [14], [26], [27], [77]; however, it does not eliminate harmonic distortion due to mismatched circuit components. Many techniques, including special layout techniques, laser trimming, self-calibration, error cancellation, and dynamic element matching, exist which reduce or relocate this harmonic distortion. To evaluate the performance of these techniques, a relationship between nonuniform quantization step size and the ADC output is developed. In particular, two methods are developed which determine the ADC's output harmonic distortion from the ADC's transfer function. Using this relationship, ADC performance metrics, such as signal to distortion ratio (SDR) and signal to noise plus distortion ratio (SNDR), can be calculated from an ADC's transfer function.

3.1 Power function approximations of an ADC's transformation

To develop a relationship between an ADC's transfer function and its frequency spectrum, let \( x(t) \) denote the ADC's analog input and \( y[n] \) denote the nonideal ADC's digital output. Thus, the ADC's output can be written as

\[
y[n] = T\left\{ x(t) \right\}_{t=nT_s},
\]

(3.1)
where $T$ is a transformation which maps a continuous function to a continuous function and represents the ADC's operation, and $T_x$ is the ADC's sampling period. Figure 3.1 shows an example transformation for a nonideal three bit ADC.

Because the analog circuit elements used in many ADC architectures can be accurately modelled with power functions, the transformation $T$ has been modelled using power functions [37]. For example, insulating materials around conductors retain past polarization due to the dielectric absorption. This memory characteristic which causes missing codes and nonlinearities has been modelled by a power function [25]. Figure 3.1 shows a power curve function, $T_a$, that models an ADC's transfer function. Figure 3.2 shows the difference between this power curve function and the nonideal ADC transfer function in Figure 3.1. This fluctuating transformation, the transformation in Figure 3.2,

Figure 3.1. An example three bit ADC transformation. The characteristic staircase ADC transfer function, $T\{x(t)\}$, and the smoothed transfer function modelled by a power function, $T_a\{x(t)\}$.
can be modelled as additive noise and handled in the same manner as quantization errors [74]. It has been shown that the power curve function model of a nonideal ADC transfer function is accurate when the fluctuation transformation is relatively insignificant compared to the distortion due to the nonlinear power function [37].

3.1.1 Symmetric power function approximation

In [37], ADC transformations were modelled by the power functions

\[
T_a\{x(t)\} = \begin{cases} 
    x(t)^{1+\nu} & x(t) > 0 \\
    0 & x(t) = 0 \\
    -[-x(t)]^{1+\nu} & x(t) < 0
\end{cases}
\]  

(3.2)

Figure 3.2. The fluctuating portion of the example three bit ADC transformation shown in Figure 3.1.
where $v$ is a constant that represents the amount of nonlinearity in the transformation. To calculate the ADC’s harmonic distortion using the nonlinear transformation in (3.2), let the input be

$$x(t) = \sin \Omega t,$$  \hspace{1cm} (3.3)

which implies that the approximate ADC output, $y(t)$, is

$$y(t) = T_a \{\sin \Omega t\},$$  \hspace{1cm} (3.4)

where $\Omega$ is the ADC’s input frequency in radians/sec. The ADC’s harmonic distortion can be estimated using a Fourier series expansion of the power function’s response to the ADC input. The $k$th Fourier series coefficient, $F(k)$, of $y(t)$ is given by

$$F(k) = \frac{\Omega}{2\pi} \int_{-\pi/2}^{\pi/2} T_a \{\sin \Omega t\} e^{jk\Omega t} dt,$$  \hspace{1cm} (3.5)

where $T = 2\pi/\Omega$ is the period of the ADC’s sinusoidal input.

To calculate the Fourier series coefficients in (3.5), note that

$$\int_0^\pi (\sin^b x)(\sin ax) dx = \frac{\pi \sin a\pi}{2} \frac{2^b(b+1)\beta \left( \frac{a+b+2}{2}, \frac{b-a+2}{2} \right)}{2^{b+1}}$$  \hspace{1cm} (3.6)

and

$$\int_0^\pi (\sin^b x)(\cos ax) dx = \frac{\pi \cos a\pi}{2} \frac{2^b(b+1)\beta \left( \frac{a+b+2}{2}, \frac{b-a+2}{2} \right)}{2^{b+1}}$$  \hspace{1cm} (3.7)

where $\beta(z, w)$ is the Beta function, which is written

$$\beta(z, w) = \int_0^1 t^{z-1}(1-t)^{w-1} dt,$$

for $\text{Re}\{z\} > 0$ and $\text{Re}\{w\} > 0$ [1]. Using (3.6) and (3.7),
Using (3.8), the Fourier series coefficients, \( F(k) \), in (3.5) can be written as

\[
F(k) = \frac{(1 - e^{jk\pi})e^{jk\pi/2}}{2^{1+v(v+2)}\beta\left(\frac{k+v+3}{2}, \frac{v-k+3}{2}\right)}
\]

which implies that

\[
F(2m) = 0 \quad m = 0, 1, 2, ...
\]

(3.9)

and

\[
F(2m+1) = \frac{(-1)^m 2j}{2^{1+v(2+v)}\beta\left(\frac{v+2m+4}{2}, \frac{v-2m+2}{2}\right)} \quad m = 0, 1, 2, ...
\]

(3.10)

Although exact Fourier series coefficients can be computed using (3.9) and (3.10), the nonlinearity parameter, \( v \), is typically very small which implies that \( 2^{1+v} \approx 2 \) and \( 2+v \approx 2 \).

Substituting these approximation into (3.10),

\[
F(1) = \frac{2j}{4\beta\left(\frac{v+4}{2}, \frac{v+2}{2}\right)} = \frac{2j}{4\beta(2,1)} = j,
\]

(3.11)

since \( \beta(z,w) = \Gamma(z)\Gamma(w)/\Gamma(z+w) \), where \( \Gamma(z) = \int_0^\infty t^{z-1}e^{-t}dt \) which is the gamma function [1]. Similarly

\[
F(3) = \frac{-j}{4\beta\left(3 + \frac{v}{2}, \frac{v}{2}\right)} = \frac{iv}{4},
\]

(3.12)

since \( \beta(3+v/2,v/2) \approx 2/v \) for small \( v \). Also, it can be shown that the higher odd coefficients are insignificantly small when compared to \( F(1) \) and \( F(3) \) [37].
3.1.2 Asymmetric power function approximation

Often, a nonideal ADC's transfer function is asymmetric about midscale, and the ADC model in (3.2) cannot adequately model the ADC's nonlinearity [13], [37]. A transformation that can more accurately model the nonlinearity is asymmetric about midscale and is written

\[
T_a\{x(t)\} = \begin{cases} 
  x(t)^{1+v} + \mu x^2(t) & x(t) > 0 \\
  0 & x(t) = 0 \\
  -[−x(t)]^{1+v} + \mu x^2(t) & x(t) < 0 
\end{cases} \quad (3.13)
\]

where \(v\) is the nonlinearity parameter similar to the parameter in (3.2), and \(\mu\) is a constant that represents the amount of asymmetry in the transformation [37].

Using a derivation similar to the derivation in Section 3.1.1, the Fourier series coefficients of the ADC output with the transformation given in (3.13) can be estimated. Because the nonlinearity parameter, \(v\), is typically very small and (3.11) and (3.12) can be used to approximate the nonlinear ADC's transfer function, the asymmetry parameter, \(\mu\), affects only the zeroth and second Fourier coefficients. Therefore, the Fourier series coefficients of the ADC output with the transformation given in (3.13) is

\[
F(0) = \mu \quad (3.14)
\]

\[
F(1) = j \quad (3.15)
\]

\[
F(2) = \frac{\mu}{2} \quad (3.16)
\]

\[
F(3) = \frac{j \mu}{4} \quad (3.17)
\]

\[
F(m) = 0 \quad m = 4, 5, ... \quad (3.18)
\]
As expected, the asymmetric component of the transformation in (3.13) affects DC and the second harmonic. The symmetric component of the transformation in (3.13) mostly affects the third harmonic like the transformation in (3.2). These ADC output Fourier coefficients in (3.14)-(3.18) show that the asymmetric transformation model in (3.13) generates harmonic distortion mostly in the second and third harmonics. Higher order Fourier series coefficients are insignificant.

3.1.3 ADC performance criteria

Three criteria used to measure an ADC’s performance are SFDR, SDR, and SNDR. For the symmetric power function approximation of an ADC’s transfer function, an ADC’s SFDR can be approximated using (3.11) and (3.12), and for the asymmetric power function approximation of an ADC’s transfer function, an ADC’s SFDR can be approximated using (3.15)-(3.18).

To calculate an ADC’s SDR and SNDR using the two power function approximations, the ADC’s output fundamental is assumed to be undistorted signal information. Because DC energy is typically not considered when computing ADC performance metrics [31], [59], [65], an ADC’s SDR can be approximated as the ratio of the power of the undistorted signal information and the power of the distortion due to nonuniform step sizes. For an ADC modelled by the symmetric power function in (3.2), an estimate of an ADC’s SDR is

\[
\text{SDR} = \frac{|F(1)|^2}{|F(3)|^2} = \frac{|j|}{|jv/4|^2} = \frac{16}{v^2},
\]

where \(F(1)\) and \(F(3)\) are the fundamental and third harmonic Fourier series coefficients in (3.11) and (3.12). For an ADC modelled by the asymmetric power function in (3.13), an estimate of an ADC’s SDR is
where $F(1)$, $F(2)$, and $F(3)$ are the fundamental, second, and third harmonic Fourier series coefficients in (3.15)-(3.17).

To calculate an ADC's SNDR, consider an ADC input, $x(t)$, which can be written as

$$x(t) = s(t) + w(t)$$

(3.21)

where $s(t)$ is the signal component of the ADC's input and $w(t)$ is an independent zero mean white noise component of the ADC's input. When applicable, this noise component, $w(t)$, of the ADC input can represent the quantization error and the additive dither noise. If quantization and dither error are modelled as additive white noise and the ADC can be modelled by the symmetric power function in (3.2), the ADC's SNDR can be estimated by

$$\text{SNDR} = \frac{|F(1)|^2}{|F(3)|^2 + \sigma_w^2} = \frac{16}{v^2 + \sigma_w^2}$$

(3.22)

where $F(1)$ and $F(3)$ are the fundamental and third harmonic Fourier series coefficients in (3.11) and (3.12), and $\sigma_w^2$ is the power of $w(t)$. If the quantization and dither error are modelled as additive white noise and the ADC can be modelled by the asymmetric power function in (3.13), the ADC's SNDR can be estimated by

$$\text{SNDR} = \frac{|F(1)|^2}{|F(2)|^2 + |F(3)|^2 + \sigma_w^2} = \frac{16}{v^2 + \mu^2 + \sigma_w^2}$$

(3.23)

where $F(1)$, $F(2)$, and $F(3)$ are the fundamental, second, and third harmonic Fourier series coefficients in (3.15)-(3.17), and $\sigma_w^2$ is the power of $w(t)$. 

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
3.2 Fourier series representation of an ADC's transformations

To derive an exact relationship between an ADC's transfer function and its output, consider a $B$ bit ADC that can be represented by the transformation, $T$, such that

$$y[n] = T\left\{ x(t) \right\}_{t=nT},$$

(3.24)

where $x(t)$ is the ADC's analog input, and $y[n]$ is the ADC's digital output. Figure 3.3(a) shows a nonideal three bit ADC transfer function. Also, let the linear transformation, $T_l$, represent an ideal infinite bit ADC transfer function, that is,

$$y_l[n] = T_l\left\{ x(t) \right\}_{t=nT} = q^{-1}x(nT),$$

where $x(t)$ is the ADC's analog input, $q$ is a constant that represents the ADC's code width, and $y_l[n]$ is the linear transformation's result. This linear transformation, $T_l$, represents the straight line transfer function for the ADC with code widths, $q$. Figure 3.3(b) shows the linear transformation of an ADC that has uniform code widths $q$, where $q = 0.25$. If the nonlinear transformation, $T_p$, represents an ADC with uniform code widths, then the output, $y(nT_p)$, of the perfect ADC can be written as

$$y[n] = T_p\left\{ x(t) \right\}_{t=nT} = T_l\left\{ x(t) \right\}_{t=nT} + T_q\left\{ x(t) \right\}_{t=nT} = y_l[n] + y_q[n]$$

(3.25)

where $T_q$ is a nonlinear transformation that maps $x(t)$ to the ADC's quantization errors, $y_q[n]$, caused by truncation or rounding. Figure 3.3(c) shows an example of a quantization error transformation for an ADC with uniform code widths $q$, where $q = 0.25$. Figure 3.4(a) shows the corresponding ideal ADC transformation, $T_p$, that is obtained by

---

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Figure 3.3. Three bit ADC transformations for $q = 0.25$. (a) Nonideal ADC transformation, $T$. (b) Linear transformation, $T_l$. (c) Quantization transformation, $T_q$. 

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
adding the curves in Figure 3.3(b) and Figure 3.3(c). Using (3.25) and defining a transformation, $T_d$ which is a nonlinear transformation that maps $x(t)$ to the ADC's conversion errors, $y_d[n]$, caused by nonuniform code widths, the output, $y[n]$, of the nonideal ADC can be written

$$y[n] = T\left\{ x(t) \right\}_{t=nT_s} = \left( T_p \left\{ x(t) \right\} + T_d \left\{ x(t) \right\} \right)_{t=nT_s} = y_p[n] + y_d[n].$$

Figure 3.4. Three bit ADC transformations for $q = 0.25$. (a) Ideal transformation, $T_p$. (b) Distortion transformation, $T_d$. 

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Figure 3.4(b) shows a distortion error transformation for the ADC transformation shown in Figure 3.3(a). The total ADC transformation, $T$, in Figure 3.3(a), can be obtained by adding the curves in Figure 3.3(b), Figure 3.3(c), and Figure 3.4(b).

3.2.1 A continuous Fourier series representation of an ADC's transformations

An ADC's output can be calculated by modelling an ADC as a continuous transformation followed by ideal sampling. Figure 3.5 shows a block diagram of the ADC's operation. To calculate the harmonic distortion of an ADC with the nonlinear transformation in (3.1), let the ADC input, $x(t)$, be periodic with period $T$. The ADC's harmonic distortion can be calculated using a Fourier series expansion of transformation of the ADC's input. The $k$th Fourier series coefficient, $F(k)$, of $T\{x(t)\}$ is

$$F(k) = \frac{1}{T} \int_{-T}^{T} T\{x(t)\} e^{-jk\Omega_0 t} dt,$$

(3.26)

where the fundamental frequency, $\Omega_0$, of the ADC input is $\Omega_0 = 2\pi/T$. Neglecting dynamic ADC errors, the ADC transformation is time invariant and can be represented by a finite duration function. Therefore, $T\{x(t)\}$ can be represented by a Fourier series expansion, that is

$$T\{x(t)\} = \sum_{l=-\infty}^{\infty} A(l) e^{j\Omega_0 x(t)},$$

(3.27)

![Figure 3.5. Block diagram of ADC operation.](image-url)
where \( \omega_o = 2\pi/m(R) \),

\[
A(l) = \frac{1}{m(R)} \int_{x \in R} T\{x\} e^{-j\omega_o x} dx,
\]

(3.28)

\( m(\cdot) \) is the Lebesgue measure, the set \( R \) of points in the ADC’s full scale range is written \( R = \{x: x_{2S} \leq x \leq x_{2S} + \text{FSR}\} \), \( x_{2S} \) is the ADC’s zero scale input, and \( \text{FSR} \) is the ADC’s full scale range. Substituting (3.27) into (3.26) and rearranging terms,

\[
F(k) = \frac{1}{m(R)} \sum_{l=-\infty}^{\infty} A(l) \int_{l_0}^{l_0+T} e^{j\omega_o lx(t)} e^{j k \Omega_o t} dt.
\]

(3.29)

Defining

\[
C(l,k) = \int_{x \in R} e^{j\omega_o lx(t)} e^{j k \Omega_o t} dt,
\]

(3.30)

the Fourier series coefficients in (3.29) are

\[
F(k) = \frac{1}{m(R)} \sum_{l=-\infty}^{\infty} A(l) C(l,k).
\]

(3.31)

If the ADC’s input is a sinusoid of frequency \( \Omega_o \), that is, \( x(t) = \sin(\Omega_o t) \), (3.30) can be written

\[
C(l,k) = \int_{l_0}^{l_0+T} e^{j\omega_o l \sin \Omega_o t} e^{j k \Omega_o t} dt = J_n(\omega_o l),
\]

(3.32)

where \( J_n(z) \) denotes the Bessel function of the first kind of order \( n \) and argument \( z \).

Therefore, if \( x(t) = \sin(\Omega_o t) \), (3.31) becomes

\[
F(k) = \frac{(-1)^k}{m(R)} \sum_{l=-\infty}^{\infty} A(l) J_n(\omega_o l).
\]

(3.33)

Using (3.31) or (3.33), the Fourier series coefficients of \( T\{x(t)\} \) can be determined. The spectrum of \( T\{x(t)\} \) is

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Assuming ideal sampling in Figure 3.5, the Fourier transform, \( Y(e^{j\omega}) \), of \( y[n] \) is

\[
Y(e^{j\omega}) = \frac{1}{T_s} \sum_{m=-\infty}^{\infty} F\left( j\left( \frac{\omega}{T_s} - \frac{2\pi m}{T_s} \right) \right).
\]

The Fourier series coefficients of an arbitrary ADC's output given a sinusoidal input can be found using (3.33). Replacing \( T \) in (3.27) with \( T_q \) and \( T_d \) and inserting the result into (3.33) gives the Fourier series coefficients of the ADC's output due to quantization errors and mismatch error distortion, respectively.

### 3.2.2 A discrete approximation of an ADC's transformations

Because a closed form expression of (3.33) does not exist, an analytical expression for an ADC's SDR, SNDR, and SFDR is not easily found. By using a discrete approximation of the transformations \( T_l, T_q, T_d, T_p, \) and \( T \), an estimate of the transformations' effect can be determined.

The ADC's quantization transformation, \( T_q \), maps an analog ADC input, \( x(t) \), to an analog output, \( y_q(t) \). The ADC's distortion transformation, \( T_d \), ideal transformation, \( T_p \), and transformation, \( T \), map an analog ADC input, \( x(t) \), to continuous time, discrete amplitude output signals. Each of the transformations, \( T_l, T_q, T_d, T_p, \) and \( T \), can be sampled by \( \eta \) samples and maps \( \eta \) input values to \( \eta \) output values. Therefore, each of the five sampled transformations \( \hat{T}_l, \hat{T}_q, \hat{T}_d, \hat{T}_p, \) and \( \hat{T} \) can be represented by a sequence. For example, the ADC's nonideal transformation, \( \hat{T} \), that maps \( \eta \) analog ADC input values to \( \eta \) outputs taking on \( 2^B \) unique values can be represented by the sequence...
\[ \tilde{T} = \left\{ \tilde{T}\left[ x_{ZS} + \frac{k(x_{FS} - x_{ZS})}{\eta} \right] : 0 \leq k \leq \eta - 1 \right\}, \] (3.35)

where \( x_{ZS} \) is the ADC's zero scale input and \( x_{FS} \) is the ADC's full scale input. To simplify the notation in (3.35), define the sequence, \( z[k] \), as

\[ z[k] = \left\{ x_{ZS} + \frac{k(x_{FS} - x_{ZS})}{\eta} : 0 \leq k \leq \eta - 1 \right\}, \]

which implies that (3.35) can be written

\[ \tilde{T} = \left\{ \tilde{T}[z[k]] : 0 \leq k \leq \eta - 1 \right\}. \] (3.36)

Therefore, the sequences in (3.35) and (3.36) are equivalent to the nonideal ADC's output response to the \( \eta \) equally spaced points between the ADC's zero scale and full scale inputs. The transformations \( \tilde{T}_l, \tilde{T}_q, \tilde{T}_p, \) and \( \tilde{T}_d \) can be described by similar sequences. In illustration, Figure 3.6 and Figure 3.7 show the sampled ADC transformations in Figure 3.3 and Figure 3.4, respectively, each sampled with 64 points.

Because \( \tilde{T}_l, \tilde{T}_q, \tilde{T}_d, \) and \( \tilde{T} \) are discrete models, the ADC's analog input, \( x(t) \), must be sampled and quantized. The discrete time, quantized ADC input, \( \tilde{x}(nT) \), is

\[ \tilde{x}(nT) = \left\{ Q_{z[k]}[x(nT)] : 0 \leq k \leq \eta - 1, 0 \leq n \leq N - 1 \right\}, \] (3.37)

where \( Q_{z[k]}[x(nT)] \) represents the quantization of the sequence \( x(nT) \) to the nearest point in \( z[k] \).

Because each of the sampled transformations, \( \tilde{T}_l, \tilde{T}_q, \tilde{T}_d, \) and \( \tilde{T} \), can be represented by a finite length sequence, they can be represented by a linear combination of a complete set of mutually orthogonal sequences. Therefore, the sampled transformation, \( \tilde{T} \), which
Figure 3.6. Sampled three bit ADC transformations from Figure 3.3. (a) Sampled nonideal ADC transformation, $\tilde{T}$. (b) Sampled linear transformation, $T_l$. (c) Sampled quantization transformation, $\tilde{T}_q$. 

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
maps the sampled and quantized ADC input, $\tilde{x}(nT)$, to the approximate nonideal ADC output, $\tilde{y}[n]$, can be written as

$$\tilde{y}[n] = \tilde{T}[\tilde{x}(nT)] = \frac{1}{\eta} \sum_{l=0}^{\eta-1} a_l e^{\frac{2\pi j l}{\eta} \tilde{x}(nT)}$$

where

$$a_l = \sum_{k=0}^{\eta-1} \tilde{T}[z[k]] e^{-\frac{2\pi j l}{\eta} z[k]} .$$

Figure 3.7. Sampled three bit ADC transformations from Figure 3.4. (a) Sampled ideal ADC transformation, $\tilde{T}_p$ (b) Sampled distortion transformation, $\tilde{T}_d$
If the signal \( \tilde{x}(nT) \) in (3.38) is a finite length sequence of length \( N \), then each of the mutually orthogonal sequences, \( e^{j2\pi l\tilde{x}(nT)/\eta} \), for \( 0 \leq l \leq M - 1 \), is a finite length sequence of length \( N \), and can be expressed as the linear combination of a complete set of mutually orthogonal sequences. Similarly, if the signal \( \tilde{x}(nT) \) in (3.38) is periodic with period \( N \), then each of the mutually orthogonal sequences, \( e^{j2\pi l\tilde{x}(nT)/\eta} \) for \( 0 \leq l \leq M - 1 \), is periodic with period \( N \), and can be expressed as the linear combination of a complete set of mutually orthogonal periodic sequences that have a period \( N \). Therefore, if \( \tilde{x}(nT) \) is a finite length sequence of length \( N \) or a periodic sequence with period \( N \),

\[
e^{\frac{2\pi}{\eta} \tilde{x}(nT)} = \frac{1}{N} \sum_{n=0}^{N-1} c_{k,l} e^{\frac{2\pi}{N} kn}
\]

(3.39)

for \( 0 \leq l \leq M - 1 \), where

\[
c_{k,l} = \frac{1}{\eta} \sum_{n=0}^{N-1} e^{\frac{2\pi}{\eta} \tilde{x}(nT)} e^{-\frac{2\pi}{N} kn}.
\]

Substituting (3.39) into (3.38),

\[
\hat{y}[n] = \hat{T}[\tilde{x}(nT)] = \frac{1}{\eta N} \sum_{l=0}^{\eta-1} a_l \sum_{n=0}^{N-1} c_{k,l} e^{\frac{2\pi}{N} kn}
\]

\[
- \frac{1}{\eta N} \sum_{l=0}^{\eta-1} \sum_{n=0}^{N-1} c_{k,l} a_l e^{\frac{2\pi}{N} kn}
\]

(3.40)

or

\[
\hat{y}[n] = \hat{T}[\tilde{x}(nT)] = \frac{1}{N} \sum_{k=0}^{N-1} Y_k e^{\frac{2\pi}{N} kn},
\]

(3.41)

where

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
By defining the column vector, \( \mathbf{Y} \), such that \( Y_k \) is the \( k \)th element of \( \mathbf{Y} \), the Fourier coefficients in (3.42) can also be calculated using

\[
\mathbf{Y} = \mathbf{C} \mathbf{A}
\]

(3.43)

where \( c_{k,l} \) is the element in the \( k \)th row and \( l \)th column in the matrix \( \mathbf{C} \), and \( a_l \) is the \( l \)th element in the column vector \( \mathbf{A} \). The matrix \( \mathbf{C} \) in (3.43) can also be calculated by

\[
\mathbf{C} = \begin{bmatrix}
\text{DFS}_N \left( \frac{1}{N} \right) \\
\text{DFS}_N \left( \frac{1}{N} e^{\frac{2\pi}{N} i(nT)} \right) \\
\vdots \\
\text{DFS}_N \left( \frac{1}{N} e^{\frac{2\pi}{N} (2^{\eta-1}) i(nT)} \right)
\end{bmatrix}
\]

(3.44)

where \( \text{DFS}_N \{ z[k] \} \) generates a column vector containing the \( N \) point discrete Fourier series (DFS) coefficients of the sequence \( z[k] \). Likewise, the matrix \( \mathbf{A} \) in (3.43) can be calculated by

\[
\mathbf{A} = \begin{bmatrix}
\text{DFS}_\eta \{ \tilde{T}[z[k]] : 0 \leq k \leq \eta - 1 \}
\end{bmatrix}
\]

(3.45)

where \( \text{DFS}_\eta \{ \tilde{T}[z[k]] \} \) generates a column vector containing the \( \eta \) point DFS coefficients of \( \tilde{T}[z[k]] \) for \( 0 \leq k \leq \eta - 1 \).

Using (3.43)-(3.45), the DFS coefficient of the nonideal ADC's output estimate is determined using the transformation estimate, \( \tilde{T} \), and samples of the ADC's analog input.

Replacing \( \tilde{T} \) with \( \tilde{T}_q \) in (3.45), yields

\[
\mathbf{A}_q = \begin{bmatrix}
\text{DFS}_\eta \{ \tilde{T}_q[z[k]] : 0 \leq k \leq \eta - 1 \}
\end{bmatrix}
\]

(3.46)

and

\[
\mathbf{Y}_q = \mathbf{C} \mathbf{A}_q
\]

(3.47)
where \( Y_q \) is the column vector of the ADC quantization error estimate's DFS coefficients.

Replacing \( \hat{T} \) with \( \hat{T}_p \) in (3.45),

\[
A_p = \left[ \text{DFS}_\eta \{ \hat{T}_p[k] : 0 \leq k \leq \eta - 1 \} \right],
\]

(3.48)

and

\[
Y_p = CA_p
\]

(3.49)

where \( Y_p \) is the column vector of ADC output estimate's DFS coefficients. Likewise, replacing \( \hat{T} \) by \( \hat{T}_d \) in (3.45), yields

\[
A_d = \left[ \text{DFS}_\eta \{ \hat{T}_d[k] : 0 \leq k \leq \eta - 1 \} \right],
\]

(3.50)

and

\[
Y_d = CA_d
\]

(3.51)

where \( Y_d \) is the column vector of the ADC output distortion error estimate's DFS coefficients.

### 3.2.3 ADC performance criteria

Three criteria that are used to measure an ADC's performance are SFDR, SDR, and SNDR. An ADC's SFDR can be estimated using (3.43). To estimate an ADC's SDR, consider the ADC's estimated average output information signal plus quantization error power, \( P_{y_{i+q}} \), where

\[
P_{y_{i+q}} = \frac{1}{N^2} (Y_l + Y_q)^H(Y_l + Y_q)
\]

\[
= \frac{1}{N^2}(A_l + A_q)^H C^H C(A_l + A_q)
\]

\[
= \frac{1}{N^2} A_l^H C^H C_l A_l + 2 \text{Re}(A_q^H C^H C_l A_l) + A_q^H C^H C_q A_l,
\]

(3.52)
and the superscript $H$ denotes the complex conjugate transpose. Since $\tilde{T}_l$ is a linear transformation, the ADC's estimated average output information signal power, $P_{y_l}$, can be written as

$$P_{y_l} = \frac{1}{N^2} A_l^H C^H C A_l = \frac{1}{N^2} q^{-2} X^H X,$$  \hspace{1cm} (3.53)

where $X$ is a column vector of the DFS coefficients of $\tilde{x}(nT)$. An estimate of the ADC's signal to quantization plus noise ratio (SQNR) is

$$\text{SQNR} = \frac{q^{-2} S^H S}{q^{-2} W^H W + A_q^H C C A_q + \frac{2}{N^2} q^{-1} \text{Re} \left\{ S^H C A_q + q^{-1} S^H W + W^H C A_q \right\}}.$$  \hspace{1cm} (3.54)

To calculate an ADC's SDR, consider an estimate, $P_{y_d}$, of the ADC's output distortion error power where

$$P_{y_d} = \frac{1}{N^2} Y_d^H Y_d = \frac{1}{N^2} A_d^H C^H C A_d.$$  \hspace{1cm} (3.55)

Using (3.53) and (3.55), the ADC's SDR estimate is

$$\text{SDR} = \frac{P_{y_l}}{P_{y_d}} = \frac{q^{-2} X^H X}{A_d^H C^H C A_d}.$$  \hspace{1cm} (3.56)

To estimate an ADC's SNDR, consider the sampled ADC input sequence, $\tilde{x}(nT)$, written as

$$\tilde{x}(nT) = \tilde{s}(nT) + \tilde{w}(nT)$$  \hspace{1cm} (3.57)

where $\tilde{s}(nT)$ is the ADC's input signal and $\tilde{w}(nT)$ is the sequence representing the ADC's input noise. Because $\tilde{T}_l$ is a linear transformation, the ADC's estimated output, $\tilde{y}[n]$, in (3.41) can be written as
\[
\tilde{y}(nT) = \tilde{T}_f \left\{ \tilde{s}(nT) + \tilde{w}(nT) \right\}
\]
\[
= T_f \left\{ \tilde{s}(nT) + \tilde{w}(nT) \right\} + T_q \left\{ \tilde{s}(nT) + \tilde{w}(nT) \right\} + \tilde{T}_d \left\{ \tilde{s}(nT) + \tilde{w}(nT) \right\}
\]
\[
= T_f \left\{ \tilde{s}(nT) \right\} + T_f \left\{ \tilde{w}(nT) \right\} + T_q \left\{ \tilde{s}(nT) + \tilde{w}(nT) \right\} + \tilde{T}_d \left\{ \tilde{s}(nT) + \tilde{w}(nT) \right\},
\]

and the Fourier series coefficients in (3.43) can be written as
\[
Y = q^{-1}S + q^{-1}W + Y_q + Y_d, \quad (3.58)
\]
where \(S\) and \(W\) are vectors that contain the DFS coefficients of \(\tilde{s}(nT)\) and \(\tilde{w}(nT)\), respectively, and \(Y_q\) and \(Y_d\) are the Fourier series coefficients of the estimated ADC quantization error and distortion due to mismatch errors, respectively. To estimate an ADC's SNDR, consider the ADC's average signal plus noise plus distortion power estimate, \(P_y\), where
\[
P_y = \frac{1}{N^2} Y^H Y
\]
\[
= \frac{1}{N^2} q^{-2} S^H S + \frac{1}{N^2} (q^{-2} W^H W + A_q^H CCA_d + A_q^H CCA_q)
\]
\[
+ \frac{2}{N^2} \text{Re} \left\{ q^{-1} S^H C A_q + q^{-2} S^H W + q^{-1} W^H C A_d + q^{-1} S^H C A_d + q^{-1} W^H C A_d + A_q^H C C A_d \right\}
\]
where the first term, \(\frac{1}{N^2} q^{-2} S^H S\), is the ADC's average output signal power, and the remaining terms are the ADC's average noise plus distortion power. Therefore, the ADC's estimated SNDR is
3.3 Example

Consider a three bit bipolar ADC that has the ideal and nonideal quantization decision levels in Table 3.1. Figure 3.3(a) and Figure 3.4(a) show the ADC’s ideal transformation, $T_i$, and nonideal transformation, $T$, respectively. Table 3.1 contains the seven values describing the transformations. For this particular example, the ADC’s input sequence is a full scale dithered sinusoid that has a frequency of $157\pi/1024$ radians/sample. The dither sequence is a strictly white sequence with a triangular probability distribution function with support on $(-q,q)$. The ADC’s ideal transformation, $T_p$, quantization error transformation, $T_q$, distortion transformation, $T_d$, and nonideal transformation, $T$, are sampled with 512 points to create the sampled ideal transformation, $\hat{T}_p$, quantization error transformation, $\hat{T}_q$, distortion transformation, $\hat{T}_d$, and nonideal transformation, $\hat{T}$, respectively.

The ADC input estimate, $\tilde{x}(nT)$, was created by quantizing the dithered sinusoidal input sequence to nine bits, and (3.49) is used to calculate the ideal ADC’s power spectral density (PSD) which is shown in Figure 3.8. Figure 3.9 shows the simulated nonideal ADC’s output PSD. The nonideal ADC’s SFDR and SDR obtained through simulation are 38.5 dB and 16.8 dB, respectively. To calculate the ADC’s SNDR, assume that the ADC’s input, $x(t)$, has the form, $s(t)+w(t)$, where $s(t)$ is the unquantized sinusoidal input.

\[
\text{SNDR} = \frac{q^{-2}S^H S}{q^{-2}W^HW + A^H_CC_A_d + A_q^HCCA_q + \frac{2}{N^2} \text{Re}\left\{A_q^HCCA_q\right\} + \frac{2}{N^2} q^{-1} \text{Re}\left\{S^HCA_q + q^{-1}S^HW + W^HCA_q + S^HCA_d + W^HCA_d\right\}}.
\]
Table 3.1. Example's ideal and nonideal ADC quantization decision levels.

<table>
<thead>
<tr>
<th>Ideal quantization decision levels</th>
<th>Nonideal quantization decision levels</th>
</tr>
</thead>
<tbody>
<tr>
<td>-0.75</td>
<td>-0.7379</td>
</tr>
<tr>
<td>-0.50</td>
<td>-0.4598</td>
</tr>
<tr>
<td>-0.25</td>
<td>-0.1656</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0.25</td>
<td>0.1656</td>
</tr>
<tr>
<td>0.50</td>
<td>0.4598</td>
</tr>
<tr>
<td>0.75</td>
<td>0.7379</td>
</tr>
</tbody>
</table>

Figure 3.8. Power spectral density plot of the linear part of the Example's nonlinear ADC output. The ADC's input is a full scale dithered sinewave that has a frequency of $157\pi/1024$ rad/sample.
without dither and \( w(t) \) is the signal that includes quantization error and dither noise. The simulated nonideal ADC's SNDR is 9.9 dB.

The power functions in (3.2) and (3.13) can be used to approximate the nonideal ADC transfer function in Figure 3.1. Minimizing the mean square error over the entire ADC transfer function in Figure 3.1, the nonlinearity parameter and asymmetry parameter are found to be \( v = -0.1454 \) and \( \mu = 0 \), respectively. Using (3.20) and (3.23), the nonideal ADC's SDR and SNDR are estimated to be 12.7 dB and 8.6 dB, respectively.

The spectrum, \( Y_q \), of the ADC's quantization error estimate, \( \hat{T}_q(\tilde{x}(nT)) \), can be calculated by (3.47). Figure 3.10 shows the PSD of the ADC's quantization error obtained

![Figure 3.9. Power spectral density plot of the Example's nonlinear ADC output. The ADC's input is a full scale dithered sinewave that has a frequency of 157\( \pi \)/1024 rad/sample.](image)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
from (3.47). The spectrum, $Y_d$, of the ADC's distortion estimate, $\tilde{T}_d\{\tilde{x}(nT)\}$, can be calculated by (3.51). Figure 3.11 shows the PSD of the simulated ADC's distortion obtained from (3.51). Using (3.56), the ADC's SDR is 16.8 dB. To calculate the ADC's SNDR, assume that the ADC's input estimate, $\hat{x}(nT)$, has the form, $\tilde{s}(nT) + \tilde{w}(nT)$, where $\tilde{s}(nT)$ is the unquantized sinusoidal input estimate without dither and $\tilde{w}(nT)$ is the signal that includes estimate of quantization error and dither noise. Then using (3.60), the ADC's SNDR estimate is 10.0 dB.

A summary of the Example's results are found in Table 3.2. The power function approximation of the ADC's transfer function provides only a rough estimate of the ADC's

![Figure 3.10](image_url)

**Figure 3.10.** Power spectral density plot of the quantization error part of the Example's nonlinear ADC output. the ADC's input is a full scale dithered sinewave that has a frequency of $157\pi/1024$ rad/sample.
Table 3.2. Summary of Example's results.

<table>
<thead>
<tr>
<th>Method of estimation</th>
<th>SDR</th>
<th>SNDR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power function approximation (3.20) and (3.23)</td>
<td>12.7 dB</td>
<td>8.6 dB</td>
</tr>
<tr>
<td>Fourier analysis (3.56) and (3.60)</td>
<td>16.8 dB</td>
<td>10.0 dB</td>
</tr>
<tr>
<td>Simulation results</td>
<td>16.8 dB</td>
<td>9.9 dB</td>
</tr>
</tbody>
</table>

SDR and SNDR. The power function approximation represents only the most general trends of the ADC's transfer function, while modelling quantization as additive white noise. In this example, the nonlinearity parameter is significant and the assumption leading to (3.11) and (3.12) introduces additional error. The estimation of the ADC's output

![Power spectral density plot](image)

Figure 3.11. Power spectral density plot of the harmonic distortion part of the Example's nonlinear ADC output. The ADC's input is a full scale dithered sinewave that has a frequency of $157\pi/1024$ rad/sample.
spectrum, SDR, and SNDR with (3.43), (3.56), and (3.60), respectively, provides more accurate results but requires significantly more computations. Reducing the number of transfer function samples reduces the computational burden at the expense of accuracy.

3.4 Summary

In this chapter, an ADC's transfer function is approximated by a power function and related to the ADC performance criteria SDR and SNDR. In particular, determining the nonlinearity and asymmetry parameters in (3.13) for a given ADC allows the SDR and SNDR to be calculated with (3.20) and (3.23), respectively. Also, a Fourier analysis technique was developed that estimates its output frequency spectrum from the ADC's transfer function. In particular, (3.42) and (3.43) calculate an ADC's output frequency spectrum given an ADC's transfer function and a particular input signal. Using an estimate of the ADC's quantization transformation and distortion transformation, (3.47) and (3.51) can also be used to calculate an ADC's harmonic distortion. The results from (3.42) and (3.43) can also be used to determine an ADC's SFDR. Also for a given transfer function and a particular input signal, an ADC's SDR and SNDR can be calculated using (3.56) and (3.60).
CHAPTER 4

DYNAMIC ELEMENT MATCHING TECHNIQUES FOR
ANALOG TO DIGITAL CONVERTERS

Many analog to digital converter (ADC) architectures use matched references, comparators, and switches for signal conversion. ADCs typically generate matched references using matched components such as resistors, capacitors, or transistors. Manufacturing processes cannot generate components which have identical electrical properties. The component value deviations, called mismatch errors, create inaccurate ADC references and quantization levels such that the ADC's output sequence, \( y[n] \), can be written as \( y[n] = x[n] + e[n] \), where \( x[n] \) is the ideal ADC output sequence and \( e[n] \) is a sequence representing ADC conversion errors. This conversion error sequence, \( e[n] \), contains harmonic distortion that reduces an ADC's SFDR, SDR, SNDR and ENOB.

Some ADC designs reduce component mismatches by using special fabrication processes or by laser trimming components; however, component mismatches cannot be completely eliminated. Other techniques, such as digital error correction algorithms [42], [63] and self calibration techniques [67], can also reduce the effects of component mismatches. Digital error correction modifies the ADC's digital output so that the analog inaccuracies are corrected. Self calibration techniques reduce mismatch errors by periodically inspecting the ADC's references and adjusting them to the required accuracy.
Another technique, called dynamic element matching (DEM), can reduce the effects of components mismatches by varying the components' interconnections [13], [70].

DEM algorithms dynamically rearrange the interconnections of mismatched components so that the time averages of the equivalent components at each of the component positions are nearly equal. If the mismatched components' virtual positions are appropriately varied, the harmonic distortion caused by the mismatched components can be reduced, frequency shifted or eliminated. As a result, DEM algorithms can increase an ADC’s SFDR, SDR, SNDR and ENOB.

Deterministic DEM algorithms rearrange the mismatched components according to a deterministic algorithm which in practice is periodic with a relatively small period. ADCs that use deterministic DEM algorithms have quantization interval errors that vary deterministically. Therefore, the ADC input signal is deterministically related to the nonuniform quantization intervals and this deterministic relationship causes distortion at particular frequencies in the ADC’s output. As a result, the mismatched components create distortion in particular frequency bands that are determined by the ADC’s input. In some applications, this distortion can be reduced or removed by subsequent processing, e.g. lowpass filtering in oversampling data converter architectures. Thus, the DEM ADC’s SFDR, SDR and SNDR can be increased [6], [46].

Stochastic DEM algorithms rearrange the mismatched components according to a stochastic algorithm or an algorithm which is periodic with a very large period. ADCs that use stochastic DEM algorithms have quantization intervals errors that vary stochastically. Therefore, if the stochastic algorithm is uncorrelated with the ADC’s input signal, the ADC’s input signal is uncorrelated with the nonuniform quantization intervals that cause
distortion in the ADC's output. As a result, the mismatched components cause the ADC conversion error sequence, $e[n]$, to be an uncorrelated, random signal rather than distortion correlated with the ADC's input. In some applications, this random signal, or noise, can be reduced or removed by subsequent processing, e.g. lowpass filtering in oversampling data converter architectures. Thus, the DEM ADC's SFDR, SDR and SNDR can be increased [15].

Digital DEM algorithms have been used in multibit oversampling ADC feedback path DACs to correct the DAC's nonuniform quantization intervals [6], [15], [22], [34], [46]. Without these digital DEM algorithms, multibit DACs in multibit oversampling ADC feedback paths do not have the needed accuracy. The only analog DEM algorithm reported in the open literature is the stochastic barrel shifting flash resistor string ADC [13], [70].

The architectures introduced in [13] and [70] produce uniform quantization intervals on average. Simulations of the stochastic barrel shifting analog DEM algorithm show no detectable harmonic distortion in the ADC output.

In this chapter, the effects of DEM on ADC code widths are analyzed. The DEM algorithm's effect on the ADC's SDR, SNDR, SNR and SFDR are determined. Finally, the effects that DEM algorithms have on two flash ADC architectures, resistor string ADCs and current steering ADCs, are analyzed. Circuit implementations of these two flash ADC architectures are presented.
4.1 Principles of DEM in ADCs

In Chapter 3, an ideal ADC's transformation was represented by the sum of a linear transformation and a nonlinear transformation representing quantization. Specifically, an ideal ADC's output, $y_p[n]$, is

$$y_p[n] = T_p\{x(nT)\} = T_l\{x(nT)\} + T_q\{x(nT)\} = y_l[n] + y_q[n]$$

where $T_p$ is an ideal ADC transformation, $T_l$ is a linear transformation mapping the ADC input, $x(nT)$, to the linear portion of the ADC's output signal, $y_l[n]$, and $T_q$ is a nonlinear transformation mapping the ADC input, $x(nT)$, to the quantization errors, $y_q[n]$. The ADC's quantization error sequence, $y_q[n]$, can contain harmonic distortion which reduces the ADC's SFDR, SDR and SNDR [26]. However, adding an appropriate analog dither signal to the ADC's input signal before sampling can make the quantization error appear as noise in the ADC's output spectrum [27].

The transformation of an ADC containing mismatched circuit components can also be represented by a nonlinear transformation. This nonlinear transformation can be written as the sum of the transformation representing an ideal ADC and a nonlinear transformation representing the distortion due to mismatched components. Specifically, the ADC's output, $y[n]$, can be written as

$$y[n] = T\{x(nT)\} = T_p\{x(nT)\} + T_d\{x(nT)\} = y_p[n] + y_d[n],$$

where $T_d$ is a nonlinear transformation mapping the ADC's input, $x(nT)$, to the conversion errors, $y_d[n]$, caused by mismatch errors. The nonlinear conversion error, $y_d[n]$, due to nonuniform quantization step sizes contains harmonic distortion that reduces the ADC's SFDR, SDR and SNDR. Thus, an ADC's harmonic distortion can be...
reduced and the ADC's SFDR, SDR and SNDR increased by decreasing component mismatch errors in the ADC circuitry.

Figure 4.1 shows a $B$ bit DEM flash ADC topology which performs DEM by mapping $2^B$ or more reference sources to $2^B$ comparators through an interconnection network. The reference quantities are combined to generate the $2^B$ decision levels which are compared to the ADC's analog input, $x(nT)$, by the $2^B$ comparators. The comparator outputs, \{ $y_0[n], y_1[n], \ldots, y_{2^B-1}[n]$ \}, represent a $2^B$ bit scrambled thermometer coded signal, i.e. a $2^B$ bit thermometer coded signal with a nonstandard, possibly random, bit ordering. The comparator outputs are summed to form the ADC's digital output, $y[n]$. In practice, the summing operation in Figure 4.1 is performed by digital logic which also generates the ADC's appropriate digital output code.

Without the interconnection network, the $l$th decision level is determined by particular references and a particular comparator. With perfectly matched references, the ADC's code widths are a constant, $q$, and the reference level applied to the $l$th comparator is $lq$.

![Figure 4.1. A $B$ bit dynamic element matching flash ADC architecture.](image)
The outputs of all the comparators are summed to produce the ADC's output, \( y[n] \), where
\[
y[n] = Q\{x(nT)\}
\]
and \( Q\{\cdot\} \) is the quantization operation. In practice, mismatch errors create reference source errors and comparator errors. Both of these errors cause nonuniform code widths. The nonideal ADC transfer function caused by nonuniform code widths causes deterministic conversion errors and harmonic distortion in the ADC's output spectrum. Therefore the ADC's SFDR, SDR, and SNDR are reduced.

With the interconnection network, the \( l \)th decision level is determined by references and a comparator based on interconnection network’s topology and the interconnection network’s control signal, \( c[n] \). Therefore, the positions of mismatched components are virtually altered. If the interconnection network’s control signal is uncorrelated with the ADC’s input signal, the ADC’s input signal is uncorrelated with the nonuniform quantization intervals which cause distortion in the ADC’s output. As a result, the mismatched components cause the ADC conversion error sequence, \( e[n] \), to be an uncorrelated, random signal rather than distortion correlated with the ADC’s input.

Because DEM algorithms dynamically rearrange components, deterministic DEM ADCs can be classified as nonlinear time varying systems, and stochastic DEM ADCs can be classified as nonlinear random systems. For a DEM ADC that uses a DEM algorithm that is controlled deterministically, the erroneous decision levels are applied to the comparators according a deterministic, typically periodic, algorithm. As a result of the virtual rearrangement of reference sources and comparators, the mismatched components generate distortion in particular frequency bands. If the frequency bands are known a priori or can be controlled, the distortion can be reduced or removed in subsequent processing, e.g. lowpass filtering in delta-sigma architectures. For a DEM ADC that uses a
DEM algorithm that is controlled stochastically, the erroneous decision levels are applied
to the comparators randomly. As a result of this virtual rearrangement of reference sources
and comparators, the mismatched components generate distortion across the spectrum and
not in particular frequency bands. Regardless of the DEM algorithm, the mismatched
component errors and the interconnection network, the DEM ADC's SFDR, SDR and
SNDR can be increased.

4.2 An analysis of DEM ADCs

Two DEM algorithms for ADCs were introduced in [13], [70]. These two techniques,
both of which are based on barrel shifting mismatched components, perform the task of
data conversion without introducing harmonic distortion in the ADC's output. No metrics
exist by which these and other DEM algorithms for ADCs can be compared. In this
section, a DEM ADC architecture is analyzed and criteria are developed for comparing
this architecture's performance when various DEM algorithms are applied to it.

4.2.1 Performance criteria for DEM ADCs

If the interconnection network's control signal, \( c[n] \), is stochastic, the transformation
that maps the unit references and the comparators to quantization levels is stochastic. This
implies that the distortion transformation, \( T_d \), and thus, the ADC transformation, \( T \), are
stochastic transformations. Similarly, if \( c[n] \) is deterministic, the transformation that maps
the unit references and the comparators to the quantization levels is deterministic. This
implies that \( T_d \), and thus \( T \), are deterministic transformations. As a result, stochastic
DEM ADC performance criteria are calculated using probabilistic means and variances,
and deterministic DEM ADC performance criteria are calculated using arithmetic means and variances. In this section, the ADC's mean quantization reference values, the ADC's mean INL, the ADC's SDR, and the ADC's SNDR are determined for a stochastic DEM ADC. These performance criteria can also be applied to deterministic DEM ADCs by replacing the probabilistic means and variances with arithmetic means and variances, respectively.

4.2.1.1 Expected quantization decision levels

One performance criterion that can be used to evaluate DEM ADCs is the expected value of the ADC's quantization decision levels. A DEM ADC with quantization decision levels which are perfect on average exhibits less harmonic distortion than the same ADC without DEM [13]. To determine the DEM ADC's average quantization decision levels, define the ADC's $k$th quantization decision level, $L_k(n)$, at time $nT$ as

$$L_k[n] = k\Delta + T_C^T[k, c[n]]M + T_R^T[k, c[n]]R$$

where $\Delta$ is the average code width, $M=[\mu_0 \mu_1 \ldots \mu_{2^r-1}]^T$, $\mu_i$ is the $i$th comparator's input offset, $R=[r_0 r_1 \ldots r_{N_R-1}]^T$, $r_i$ is the $i$th unit reference's deviation from the unit references' mean value, $N_R$ is the number of ADC unit references, $T_C[k, c[n]]$ is the transformation which determines which comparator is used for the $k$th quantization level for control signal, $c[n]$, and $T_R[k, c[n]]$ represents the selection of unit references to create the $k$th quantization level reference level for control signal, $c[n]$. As will be shown in subsequent sections, the physical interpretation and value of $N_R$ in (4.1) varies depending on the converter's implementation. Regardless of the DEM ADC's implementation, the expected value of the ADC's $k$th quantization level reference level is
To calculate the mean of the DEM ADC reference levels for a deterministic ADC, replace the probabilistic means in (4.2) with arithmetic averages.

The output, \( y[n] \), of the DEM ADC can be expressed in terms of the quantization decision levels in (4.1) and is given by

\[
y[n] = \sum_{k=0}^{2^q-1} u\left(x(nT) - L_k[n]\right)
\]

where \( u(t) \) is the Heaviside unit step function defined as

\[
u(t) = \begin{cases} 1, & t \geq 0 \\ 0, & t < 0 \end{cases}
\]

4.2.1.2 Expected output and INL

Another performance criterion for DEM ADCs is the expected DEM ADC output conditioned on the input value. As shown in Section 3.2, an ADC's output can be written as the sum of three transformations

\[
y[n] = T_l[x(nT)] + T_q[x(nT)] + T_d[x(nT)] = y_l[n] + y_q[n] + y_d[n]
\]

where \( T_l \) represents a linear transformation, \( T_q \) is the quantization transformation, \( T_d \) is the ADC's distortion transformation due to mismatched circuit components, \( y_l[n] \) is the ADC's output due to the transformation, \( T_l \), \( y_q[n] \) is the ADC's output due to the transformation, \( T_q \), and \( y_d[n] \) is the ADC's output due to the transformation, \( T_d \). The output of an ADC can also be written as the sum of a linear transformation and a transformation, \( T_{\text{INL}} \), which represents the ADC's INL, that is

\[
E\{L_k|k\} = k\Delta + E\left\{T_C^T|k\right\}M + E\left\{T_R^T|k\right\}R.
\]
\[ y[n] = T_i[x(nT)] + T_{\text{INL}}[x(nT)]. \] (4.4)

To calculate the mean of a stochastic DEM ADC's output, the expectation operator conditioned on the input signal, \( x(nT) \), is applied to the ADC's output in (4.3), that is,

\[ E[y[n]|x(nT)] = T_i[x(nT)] + T_q[x(nT)] + E[T_d|x(nT)]. \] (4.5)

The first term, \( T_i[x(nT)] \), in (4.5) represents the linear part of the ADC's output and the remaining two terms in (4.5) are the stochastic DEM ADC's expected INL for a particular input \( x(nT) \), that is

\[ E[T_{\text{INL}}|x(nT)] = T_q[x(nT)] + E[T_d|x(nT)]. \] (4.6)

Because the two terms, \( T_i[x(nT)] + T_q[x(nT)] \), in (4.5) represent the transformation, \( T_p \), of a perfect or ideal ADC, (4.5) can also be written as

\[ E[y[n]|x(nT)] = T_p[x(nT)] + E[T_d|x(nT)]. \] (4.7)

To calculate the mean ADC output conditioned on \( x(nT) \) and mean of the ADC INL for a deterministic ADC, replace the probabilistic means in (4.5), (4.6) and (4.7) with arithmetic averages.

### 4.2.1.3 Signal to distortion ratio and signal to noise plus distortion ratio

Two other criteria that can be used to evaluate an ADC's performance are signal to distortion ratio (SDR) and signal to noise plus distortion ratio (SNDR). The SDR of an ADC is the ratio of the power of the signal portion of the ADC's output to the power of the distortion in the ADC's output. The SNDR of an ADC is the ratio of the power of the signal portion of the ADC's output to the power of the noise and the distortion in the
ADC's output. Because $y[n]$ is a real stochastic stationary signal, the ADC's average signal plus distortion power, $P_y$, is

$$P_y = E\left[y^2[n]\right] = E\left[ E\left[y^2[n]|x(nT)\right]\right]. \quad (4.8)$$

Using (4.5) and (4.8), the ADC's average signal plus distortion power is

$$P_y = E\left[T_i^2 + T_q^2 + E\left(T_d^2|x\right) + 2T_iT_q + 2T_tE\left(T_d|x\right) + 2T_qE\left(T_d|x\right)\right]. \quad (4.9)$$

Rearranging the terms in (4.9),

$$P_y = E\left[T_i^2\right] + E\left[T_q^2\right] + 2E\left[T_iT_q\right] + E\left[T_d^2\right] + 2E\left[T_iE\left[T_d|x\right]\right] + T_qE\left[T_d|x\right] \quad (4.10)$$

$$= P_I + P_q + P_d$$

In (4.10), the first term, $E\{T_i^2\}$, is the average power, $P_I$, of the output of a linear transformation. $E\{T_q^2\} + 2E\{T_iT_q\}$ represents the average power, $P_q$, of the ADC's quantization error signal and the remaining terms represent the average power, $P_d$, of the ADC's conversion error, or distortion, signal.

Typically, the signal portion, $T_i[x(nT)]$, of an ADC's digital output is defined as an infinite precision representation of the ADC's analog input. The power of the ADC output's signal component is the power of the ADC output due to the linear transformation, i.e. $P_s = P_I$. Therefore, the ADC's SDR is

$$\text{SDR} = \frac{P_s}{P_d} = \frac{E\left[T_i^2\right]}{2E\left[T_iE\left[T_d|x\right]\right] + E\left[T_d^2|x\right]}, \quad (4.11)$$
and the ADC’s SNDR is

$$\text{SNDR} = \frac{P_s}{P_{n+d}}$$

$$= \frac{E\left\{ T_f^2 \right\}}{E\left\{ T_q^2 \right\} + E\left\{ T_d^2 \right\} + 2E\left\{ T_fT_q \right\} + 2E\left\{ T_fE\left\{ T_d|x \right\} \right\} + T_qE\left\{ T_d|x \right\} \right\} \right\}. \quad (4.12)$$

Because $T_f$ is a linear transformation, (4.11) can be written

$$\text{SDR} = \frac{q^{-2}E\left\{ x^2 \right\}}{2E\left\{ T_fE\left\{ T_d|x \right\} \right\} + E\left\{ T_d^2|x \right\}} \right\}, \quad (4.13)$$

and (4.12) can be written

$$\text{SNDR} = \frac{q^{-2}E\left\{ x^2 \right\}}{E\left\{ T_q^2 \right\} + E\left\{ T_d^2 \right\} + 2E\left\{ T_fT_q \right\} + 2E\left\{ T_fE\left\{ T_d|x \right\} \right\} + T_qE\left\{ T_d|x \right\} \right\} \right\}, \quad (4.14)$$

where $q$ is the quantization step size of the ideal ADC transformation $T_f$.

To calculate the SDR and the SNDR for a deterministic ADC, replace the probabilistic means in (4.11) and (4.12) with arithmetic averages.

4.2.1.4 Performance criteria in the presence of dither

In some applications, an independent, zero mean analog dither signal is added to the ADC’s input so that the quantization error appears as white noise instead of harmonic distortion. The addition of dither is done in the same spirit as DEM. Dither added to the ADC input effectively changes all of the ideal quantization decision levels by that dither
value. Since the quantization reference values change each sample interval and are uncorrelated to each other, the quantization error is rendered as noise instead of harmonic distortion. Therefore, adding a dither signal to the ADC input has an effect similar to applying DEM to an infinite "collection" of mismatched components.

When an additive dither signal is used, the ADC's input can be written

\[ x(nT) = s(nT) + w(nT) \]

where \( s(nT) \) is the information signal and \( w(nT) \) is the dither signal. To match the DC power of the ADC's input and output, the ADC's input can be normalized, i.e. \( x(nT) \in [-1,1] \). Therefore, the ADC output, \( y_i[n] \), due to the linear transformation is

\[ y_i[n] = T_i[x(nT)] = T_i[s(nT) + w(nT)] = \bar{q}^{-1}s(nT) + \bar{q}^{-1}w(nT). \]  

(4.15)

where \( \bar{q} \) is the ADC's average quantization step size. Using (4.15), the power, \( P_i \), of \( y_i[n] \) is given by

\[ P_i = E\{T_i^2[x(nT)]\} = \bar{q}^{-2}E\{s^2(nT)\} + \bar{q}^{-2}E\{w^2(nT)\} + 2\bar{q}^{-2}E\{s(nT)w(nT)\}. \]  

(4.16)

Because, the additive dither signal, \( w(nT) \), is independent and zero mean, (4.16) can be written

\[ P_i = \bar{q}^{-2}E\{s^2(nT)\} + \bar{q}^{-2}E\{w^2(nT)\}. \]  

(4.17)

The term, \( \bar{q}^{-2}E\{s^2(nT)\} \), in (4.17) represents the power, \( P_s \), of the ADC output's information signal component and the term, \( \bar{q}^{-2}E\{w^2(nT)\} \), in (4.17) represents the power of the ADC output's dither signal component. Therefore, the ADC's SDR is

\[ \text{SDR} = \frac{P_s}{P_d} = \frac{\bar{q}^{-2}E\{s^2(nT)\}}{E\{T_d^2\} + 2E\{T_iE\{T_d|x\}\}}. \]  

(4.18)
Assuming that the quantization transformation, $T_q$, implements rounding and the ADC's input, $x(nT)$, satisfies the conditions in Section 2.1.1, the power of the quantization transformation is $q^2/12$ [26], [55] and the ADC's SNDR is

$$\text{SNDR} = \frac{P_s}{P_{n+d}}$$

$$= \frac{\bar{q}^{-2}E\left\{s^2(nT)\right\}}{\frac{1}{12} + \bar{q}^{-2}E\left\{w^2(nT)\right\} + E\left\{c^2\right\} + 2E\left\{cT_q\right\} + 2E\left\{T_qE\left\{T_d|x\right\}\right\} + T_qE\left\{T_d|x\right\}}$$

(4.19)

To calculate the SDR and SNDR for a deterministic ADC in the presence of dither, replace the probabilistic means in (4.18) and (4.19) with arithmetic averages, respectively.

4.2.2 SFDR of stochastic DEM ADCs

Another criterion that can be used to evaluate a DEM ADC's performance is SFDR. Nonideal deterministic ADCs generate periodic distortion in the ADC's output, and this periodic distortion appears as spurs in the frequency spectrum of the ADC's output. Unlike nonideal deterministic ADCs, nonideal stochastic DEM ADCs do not generate periodic distortion in the ADC's output, but instead generate stochastic distortion in the ADC's output. This stochastic distortion appears as noise in the ADC output's frequency spectrum. If an appropriate dither has been added to the information signal, the quantization error appears as noise and not harmonic distortion in the ADC output's frequency spectrum [27]. If the stochastic DEM ADC's quantization error and the distortion due to mismatched components appear as white noise, the stochastic DEM ADC's SFDR can be estimated using its SNDR.
To show that an ADC's SFDR can be estimated by its PSD noise floor, consider the stochastic DEM ADC's output autocorrelation, $\phi_{yy}[n,k]$, where

$$\phi_{yy}[n,k] = E\left\{E\{y[n]y^*[n+k]\}\right\}. \quad (4.20)$$

Since the ADC's output is real,

$$\phi_{yy}[n,k] = E\left\{E\{y[n]y[n+k]\}\right\}. \quad (4.21)$$

Substituting (4.3) into (4.21), the ADC's output autocorrelation can be written as

$$\phi_{yy}[n,k] = E\left\{(y_p[n]+E\{y_d[n]\})(y_p[n+k]+E\{y_d[n+k]\})\right\} \quad (4.22)$$

where $y_p[n]$ represents the output of a perfect ADC and $y_d[n]$ represents the part of the ADC output due to mismatched components. Assuming that the ADC input signal and the DEM algorithm control signal are stationary random processes, (4.22) can be written as

$$\phi_{yy}[n-k] = \phi_{pp}[n-k] + 2E\left\{y_p[n]y_d[n-k]\right\} + \phi_{dd}[n-k] \quad (4.23)$$

where

$$\phi_{pp}[n-k] = E\left\{y_p[n]y_p[n-k]\right\},$$

which is the autocorrelation of the output of a perfect ADC and

$$\phi_{dd}[n-k] = E\left\{y_d[n]y_d[n-k]\right\}. $$
which is the autocorrelation of the distortion signal due to mismatched components. If the DEM algorithm has no memory and is controlled by an independent, zero mean, white stochastic signal, $y_p$ and $y_d$ are uncorrelated, $E\{y_d[n]\} = 0$, and (4.23) can be written as

$$\phi_{yy}[n-k] = \phi_{pp}[n-k] + \sigma_d^2 \delta[n-k]$$  \hspace{1cm} (4.24)

where $\delta[k]$ is the Kroneker delta and $\sigma_d^2$ is the variance of the ADC's output distortion. If an analog dither signal is added to the ADC's input such that the quantization error signal is white noise, the autocorrelation of the output of a perfect ADC can be written as

$$\phi_{pp}[n-k] = \frac{\phi_{ss}[n-k]}{q^2} + \frac{(\sigma_q^2 + \sigma_d^2)}{q^2} \delta[n-k],$$  \hspace{1cm} (4.25)

where $\sigma_q^2$ is the power of the quantization error, $\sigma_d^2$ is the power of the dither signal, $q$ is the quantization step size of the perfect ADC and

$$\phi_{ss}[n-k] = E\left\{ s(nT)s((n-k)T) \right\},$$

which is the autocorrelation of the information signal component of the ADC's input. When quantization is performed by rounding, the quantization signal power is $\sigma_q^2 = q^2/12$ [26], [55]. When the dither signal is a zero mean signal with a Gaussian PDF with RMS value equal to half of a LSB, the dither power is $\sigma_d^2 = q^2/4$. When the dither signal is a zero mean signal with a rectangular PDF which has a width equal to an LSB, the dither power is $\sigma_d^2 = q^2/12$. When the dither signal is a zero mean signal with a triangular PDF that has a width equal to two LSBs, the dither power is $\sigma_d^2 = q^2/6$ [27].

The results in (4.24) and (4.25) show that the autocorrelation of a stochastic ADC is that of a linear ADC plus an independent zero mean white noise. Therefore, if the ADC's input has an additive dither which whitens quantization errors, the DEM algorithm is

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
controlled by an independent zero mean white stochastic signal and the DEM algorithm has no memory, the SFDR of a stochastic DEM ADC can be estimated using the SNDR in (4.19).

4.2.3 Comparison of four DEM flash ADCs

Consider a six bit ADC that has a full scale dithered sinusoidal input with a frequency of $1251\pi/8192$ radians/sample, unit references which have linear gradient errors that vary linearly from $+5\%$ to -$5\%$ of the LSB and ideal comparators. The dither sequence is a strictly white sequence with a triangular probability distribution function supported on $(-q, q)$. Figure 4.2 shows the power spectral density (PSD) of the simulated ADC's output.

![Figure 4.2 - Power spectral density of the output of a six bit ideal ADC.](image-url)
where the ADC has uniform quantization steps of size \( q \). The PSD was obtained by averaging 40 periodograms each corresponding to \( 2^{14} \) samples of the dithered sinusoid input sequence. Figure 4.3 shows the PSD of the nonideal ADC's output. This PSD was also obtained by averaging 40 periodograms each corresponding to \( 2^{14} \) samples. In this example, four DEM algorithms are applied to this nonideal ADC. The first two DEM algorithms use a hardware efficient interconnection network, called a barrel shift network. The first barrel shift network is controlled by deterministic signal and the second is controlled by a white uniformly distributed stochastic signal. The last two DEM algorithms use a hardware efficient interconnection network, called a generalized cube network. The first generalized cube network is controlled by a white uniformly distributed

![Figure 4.3](image.png)

**Figure 4.3.** Power spectral density of the output of a six bit ADC containing mismatched components.
stochastic signal and the second is controlled by a colored uniformly distributed stochastic signal.

The first DEM algorithm is a $B$ bit barrel shift network controlled by the output of a $B$ bit binary counter. This deterministic DEM algorithm is similar to the clocked level averaging (CLA) algorithm for DEM DACs in [6] and [46]. The second DEM algorithm is a $B$ bit barrel shift network controlled by a white uniformly distributed stochastic signal [13], [70]. Since both algorithms use the barrel shift network and their control signals are uniformly distributed, $E\{T_d|x(nT)\}$ and $E\{T_d^2|x(nT)\}$ are the same for both algorithms. Figure 4.4 and Figure 4.5 show the expected distortion transformation and expected

![Figure 4.4](image)

Figure 4.4. Expected distortion transformation for the barrel shift network DEM ADC with linear gradient mismatch errors.
distortion squared transformation, respectively. Figure 4.6 and Figure 4.7 show the PSD of the output of the nonlinear ADC using the CLA DEM algorithm and the stochastic barrel shift network DEM algorithm, respectively. Using the data in Figure 4.4 and Figure 4.5 and the statistics of the input signal, the power of the CLA DEM ADC's distortion signal and the stochastic barrel shift network DEM ADC's distortion signal are 0.232 and 0.232, respectively. Using the data in Figure 4.4 and Figure 4.5, (4.11), (4.12), and the statistics of the input signal, the SDR and SNDR of the CLA DEM ADC are 32.7 dB and 31.4 dB, respectively. Because the stochastic barrel shift DEM ADC has the same distortion mean and power as the CLA DEM ADC, the SDR and SNDR of the stochastic barrel shift

![Graph](image_url)

**Figure 4.5.** Expected distortion squared transformation for the barrel shift network DEM ADC with linear gradient mismatch errors.
network DEM ADC are also 32.7 dB and 31.4 dB, respectively. The CLA DEM ADC was simulated using $40 \times 2^{14}$ samples and its experimental distortion power, SDR and SNDR were 0.232, 32.7 dB and 31.4 dB, respectively. The stochastic barrel shift network DEM ADC was also simulated using the same $40 \times 2^{14}$ samples and the experimental distortion power, SDR, and SNDR were 0.230, 32.7 dB and 31.4 dB, respectively.

The third DEM algorithm is a $B$ bit generalized cube network controlled by a white uniformly distributed stochastic signal, and the fourth DEM algorithm is a $B$ bit generalized cube network controlled by a colored uniformly distributed stochastic signal. Since both algorithms use the generalized cube network and their control signals are

![Figure 4.6. Power spectral density of the output of a six bit DEM ADC using a deterministic clocked level averaging algorithm.](image-url)
uniformly distributed, the $E\{T_d\}$ conditioned on $x(nT)$ and $E\{T^2_d\}$ conditioned on $x(nT)$ are the same for both algorithms. Figure 4.8 and Figure 4.9 show the expected distortion transformation and expected distortion squared transformation, respectively. Using the data in Figure 4.8 and Figure 4.9, and the statistics of the input signal, the power of the generalized cube network DEM ADC’s distortion signal when the DEM network is controlled by white and colored stochastic signals are 0.332 and 0.332, respectively. Using Figure 4.8 and Figure 4.9, (4.11), (4.12), and the statistics of the input signal, the SDR and SNDR of the generalized cube network DEM ADC with a white control signal are 31.2 dB and 30.2 dB, respectively. Because the generalized cube network DEM ADC controlled by

![Figure 4.7](image)

**Figure 4.7.** Power spectral density of the output of a six bit DEM ADC using a stochastic barrel shift network controlled by a white uniformly distributed stochastic control signal.
a white stochastic signal has the same distortion mean and power as the generalized cube network DEM ADC controlled by a colored uniformly distributed stochastic signal, the SDR and the SNDR of the generalized cube network DEM ADC controlled by a colored uniformly distributed control signal are also 31.2 dB and 30.2 dB, respectively. The generalized cube network DEM ADC controlled by a white uniformly distributed stochastic signal was simulated using $40 \times 2^{14}$ samples and its experimental distortion power, SDR, and SNDR were 0.332, 31.2 dB and 30.2 dB, respectively. The generalized cube network DEM ADC controlled by a colored uniformly distributed stochastic signal was also simulated using the same $40 \times 2^{14}$ samples and its experimental distortion power,

Figure 4.8. Expected distortion transformation for the generalized cube network DEM ADC with linear gradient mismatch errors.
SDR, and SNDR were 0.332, 31.2 dB and 30.2 dB, respectively. Figure 4.10 and Figure 4.11 show the PSD of the output of the nonlinear ADC using the generalized cube network with white and colored control signals, respectively.

4.3 Resistor string flash ADCs

Resistor string flash ADCs, sometimes called voltage division flash ADCs, rely on matched resistors and comparators to perform their tasks of data conversion. In practice, perfectly matched resistors and ideal comparators are impossible to fabricate. These errors
due to component mismatches, cause nonuniform code widths and create harmonic distortion in the output of resistor string flash ADCs.

In this section, a resistor string flash ADC architecture is reviewed. When the resistors in the voltage divider network are not exactly matched, the ADC will have nonuniform quantization step sizes and harmonic distortion results. A resistor string DEM flash ADC architecture is also reviewed. The architecture utilizes DEM algorithms to reduce the voltage reference errors due to mismatched resistors, thereby reducing the harmonic distortion. The Dem algorithm randomizes the virtual positions of the voltage divider resistors and the comparator used for a given quantization level. This virtual repositioning

Figure 4.10. Power spectral density of the output of a six bit DEM ADC using a stochastic generalized cube network controlled by a white uniformly distributed stochastic control signal.
of resistors and comparators occurs at each sample instant. Requirements are derived which guarantee the ADC has ideal mean quantization reference levels apart from a constant determined by the comparators’ input offsets. Furthermore, conversion errors due to mismatched components can be uncorrelated from the input, and noise is created instead of harmonic distortion. Two resistor string flash ADC implementations are examined and shown to have ideal mean quantization levels apart from a constant.

Figure 4.11. Power spectral density of the output of a six bit DEM ADC using a stochastic generalized cube network controlled by a colored uniformly distributed stochastic control signal.
4.3.1 Resistor string flash ADC architecture

Figure 4.12 shows a block diagram of a $B$ bit resistor string flash ADC. In the figure, the resistor string voltage divider creates $2^B$ reference voltages, $L_k$ for $k = 0, 1, \ldots, 2^B - 1$. The ADC’s input voltage, $x(nT)$, is compared with each reference voltage by the corresponding comparator. In general, the $k$th comparator has a nonzero input offset, $\mu_k$. The outputs of the comparators collectively form a thermometer coded representation of the ADC's output. Digital logic converts the thermometer code into the ADC output, typically a $B$ bit binary number.

![Diagram of a B bit resistor string flash ADC](image)

Figure 4.12. Block diagram of a $B$ bit resistor string flash ADC.
Without a loss of generality, the ADC is assumed to be unipolar, i.e. the full scale reference is \(V_{\text{DD}}\) and the zero scale reference is ground. If \(\bar{R}\) is defined as the average voltage divider resistor value, that is

\[
\bar{R} = \frac{1}{2^B} \sum_{k=0}^{2^B-1} R_k, \tag{4.26}
\]

then \(R_k = \bar{R} + r_k\), where \(r_k\) is the \(k\)th resistor’s deviation from the nominal resistance. The voltage drop, \(\delta_k\), across the \(k\)th resistor is

\[
\delta_k = \frac{V_{\text{DD}} R_k}{2^B \bar{R}} = \frac{V_{\text{DD}} \bar{R} + r_k}{2^B \bar{R}} = \frac{V_{\text{DD}}}{2^B} + \frac{r_k V_{\text{DD}}}{2^B \bar{R}} = \Delta + \xi_k,
\]

where \(\Delta = \frac{V_{\text{DD}}}{2^B}\) is the voltage drop across a single \(\bar{R}\) ohm resistor and the \(k\)th resistor’s voltage drop deviation, \(\xi_k\), from ideal voltage, \(\Delta\), is

\[
\xi_k = \frac{r_k}{2^B \bar{R}} V_{\text{DD}}. \tag{4.27}
\]

In the context of an ADC with ideal comparators, \(\Delta\) corresponds to the quantization step size of an ideal ADC and \(\xi_k\) is the DNL at the \(k\)th quantization step. If the \(k\)th comparator has a nonzero input offset, \(\mu_k\), the \(k\)th reference voltage, \(L_k\), is

\[
L_k = \sum_{i=0}^{k-1} \delta_i + \mu_k = k\Delta + \mu_k + \sum_{i=0}^{k-1} \xi_i \tag{4.28}
\]

If the voltage divider network’s resistors are matched, that is \(R_0 = R_1 = \ldots = R_{2^B-1}\), then \(\xi_0 = \xi_1 = \ldots = \xi_{2^B-1} = 0\). Also if the comparators have zero input offset errors, that is \(\mu_0 = \mu_1 = \ldots = \mu_{2^B-1} = 0\), then (4.28) becomes

\[
L_k = \sum_{i=0}^{k-1} \delta_i = k\Delta \tag{4.29}
\]
Equation (4.29) shows that the resistor string flash ADC's reference voltages are perfect when the resistors are perfectly matched. Therefore, the ADC's output will not exhibit any distortion due to mismatched components. Otherwise, it is possible to find a periodic input signal such that tones, or harmonic distortion, due to component mismatches will appear in the ADC's output spectrum.

4.3.2 Resistor string DEM flash ADC architecture

In practice, the perfectly matched resistors and ideal comparators in Figure 4.12 are impossible to fabricate. As shown in (4.28), component mismatch errors cause nonuniform code widths which create harmonic distortion in the output of resistor string flash ADCs. Resistor string DEM ADC techniques randomize the mismatched resistors' ordering at each conversion instant and reduce the effects of the mismatched resistors. Furthermore, resistor string DEM ADC techniques must also be designed to randomize the positions of the comparators. To illustrate, assume that the DEM algorithm randomizes the positions of the mismatched resistors but does not randomize the positions of the nonideal comparators. The $j$th quantization level is improved by the DEM algorithm but the $j$th quantization level's comparison is always made by the $j$th comparator with a nonzero input offset error, $\mu_j$. Therefore, the ADC's output will change at an imperfect voltage due to the $j$th comparator's fixed, nonzero input offset. This results in a deterministic error sequence added to the ideal ADC's output, thereby creating harmonic distortion.

In order to randomize the positions of the voltage divider resistors and the nonideal comparators in Figure 4.12, each resistor and its corresponding comparator can be combined into a resistor string unit ADC cell. Figure 4.13 shows such a resistor string unit.
ADC cell. These resistor string unit ADC cells must be randomized if DEM is to reduce the effects of errors due to resistor mismatches and comparator mismatches [70].

Resistor string flash ADCs using DEM algorithms typically require complex routing. The quantization reference levels rearranged by the DEM algorithm are analog and susceptible to errors due to integrated circuit layout. The barrel shift DEM algorithms introduced in [13] and [70], respectively, are regular and hardware efficient. Figure 4.14 shows the architecture that implements both algorithms. These barrel shift DEM ADC architecture can be implemented with both $R-R-R$ and $R-2R-3R$ resistor strings which perform the truncating and rounding quantization operations, respectively.

4.3.2.1 Analysis of resistor string DEM flash ADCs

This section derives the average quantization reference levels of the resistor string DEM flash ADC that randomizes the unit ADC cells shown in Figure 4.12. The effects of DEM algorithms on resistor string flash ADCs can be determined by finding the expected quantization decision levels. The DEM algorithms rearranges the unit ADC cells, and as a result, the comparator offset and quantization level reference value are functions of the

\[ L_k \]
\[ R_k \]
\[ V_{in} \]
\[ \mu_k \]
\[ t_k \]

Figure 4.13. The $k$th resistor string unit ADC cell.
DEM algorithms control signal, \( c(n) \). Using (4.28), the \( k \)th quantization level reference value, \( L_k(n) \), at time \( nT \) can be written as

\[
L_k(n) = k\Delta + \mu_k[c(n)] + \sum_{l=0}^{k-1} \xi_l[c(n)].
\]  

(4.30)

Using matrix form and (4.27), (4.30) can be written as

\[
L_k(n) = k\Delta + T_M^T[k, c(n)]M + \frac{V_{DD}}{R2^B}T_R^T[k, c(n)]R
\]  

(4.31)

where \( M = [\mu_0 \mu_1 \ldots \mu_{2^B-1}]^T \), \( R = [r_0 r_1 \ldots r_{2^B-1}]^T \), \( T_M[k,c(n)] \) maps \( k \) and \( c(n) \) into a \( 2^B \times 1 \) vector that selects the comparator for the \( k \)th quantization level, and \( T_R[k,c(n)] \) maps \( k \) and \( c(n) \) into a \( 2^B \times 1 \) vector that selects unit references which create the \( k \)th

![Figure 4.14. Barrel shift DEM flash ADC architecture.](image-url)
quantization level reference voltage. The comparator selection transformation,

\[ T_C[k, c(n)] \in \{0, 1\}^{2^k} \]

selects a single comparator to implement the \( k \)th quantization level. This implies that \( T_M^T[k, c(n)]T_M[k, c(n)] = 1 \). The unit cell selection transformation,

\[ T_R[k, c(n)] \in \{0, 1\}^{2^k} \]

selects the \( k \) resistors which construct the \( k \)th quantization level reference voltage. This implies that \( T_R^T[k, c(n)]T_R[k, c(n)] = k \). Using (4.31) and taking the expected value, the expected value of the ADC’s \( k \)th quantization level reference voltage is

\[
E\{L_k\} = k\Delta + E\left( T_M^T[k]M + \frac{V_{DD}}{R2B}E\left( T_R^T[k]\right)R \right). \tag{4.32}
\]

The derivation leading to (4.32) assumes the unit ADC cells use ideal analog switches. If the unit ADC cells use non-ideal analog switches with fixed “on” resistances, a similar result is obtained.

4.3.2.2 Carbone’s implementation

One implementation of the resistor string DEM flash ADC uses the architecture in Figure 4.14 with the unit ADC cells in Figure 4.15 [13]. The DEM algorithm’s control signal, \( c(n) \), where \( c(n) \in \{0, 1, \ldots, 2^k - 1\} \), is used to control a pair of switches added to each unit ADC cell as shown in Figure 4.15. At time \( n \), the switches in the \( c(n) \)th unit ADC cell are connected to ground and \( V_{DD} \), and all other pairs of switches are kept in the position that closes the current path between adjacent resistors. With the control signal \( c(n) \), \( R_{c(n)} \) and \( \mu_{c(n)} \) are used to determine the first quantization decision level. The second quantization decision level is determined by \( R_{[c(n)+1] \mod 2^k} \) and \( \mu_{[c(n)+1] \mod 2^k} \). For example, \( R_5 \) and \( \mu_5 \) are used to determine the first quantization
decision level when \( c(n) = 5 \). The second quantization decision level is determined by 
\( R_5, R_{6 \mod 2^\alpha} \) and \( \mu_{6 \mod 2^\alpha} \). Likewise, the pattern is continued for the remaining 
quantization decision levels. At the next sampling instant, a new value is chosen for \( c(n) \) 
and the process is repeated. Therefore, using the unit ADC cell in Figure 4.15 in the 
ariculture in Figure 4.14 virtually repositions the resistors and their corresponding 
comparators.

The DEM ADC constructed with the cell in Figure 4.15 in the architecture in 
Figure 4.14 has the reference selection transformation

\[
T_R[k, c(n)] = \text{rot}(A_k, c(n)),
\]
and the comparator selection transformation

\[
T_M[k, c(n)] = \text{rot}(B, c(n)),
\]
where \( A_k \) is the \( 2^B \times 1 \) vector with ones in the first \( k \) rows and \( 2^B - k \) zeros in the 
remaining rows, \( B \) is the \( 2^B \times 1 \) vector with a one in the first rows and \( 2^B - 1 \) zeros in the 
remaining rows, and \( \text{rot}(*, n) \) is the \( n \) position rotation operator. If the control signal,

![Diagram of the kth resistor string unit ADC cell from [13]](image-url)

Figure 4.15. The \( k \)th resistor string unit ADC cell from [13].
c(n), has an uniform probability density function then the expected reference selection transformation conditioned on \( k \) is

\[
E\{T_R|k\} = \frac{1}{2^B} \sum_{l=0}^{2^B-1} \text{rot}\{A_k, l\} = \frac{k}{2^B} \mathbf{1} ,
\]

(4.33)

where \( \mathbf{1} \) is a \( 2^B \times 1 \) vector of ones. Similarly if the control signal, \( c(n) \), has an uniform probability density function then the expected comparator selection transformation conditioned on \( k \) is

\[
E\{T_M|k\} = \frac{1}{2^B} \sum_{l=0}^{2^B-1} \text{rot}\{B, l\} = \frac{1}{2^B} \mathbf{1} .
\]

(4.34)

Because \( R_k = \overline{R} + r_k \), (4.26) can be written as

\[
\overline{R} = \frac{1}{2^B} \sum_{k=0}^{2^B-1} R_k = \frac{1}{2^B} \sum_{k=0}^{2^B-1} \overline{R} + r_k = \overline{R} + \frac{1}{2^B} \sum_{k=0}^{2^B-1} r_k
\]

which implies that

\[
\sum_{k=0}^{2^B-1} r_k = \mathbf{1}^T \overline{R} = 0 .
\]

(4.35)

Substituting (4.33), (4.34) and (4.35) into (4.32),

\[
E\{L_k\} = k\Delta + \frac{1}{2^B} \mathbf{1}^T \overline{M} = k\Delta + \overline{\mu}
\]

(4.36)

where

\[
\overline{\mu} = \frac{1}{2^B} \mathbf{1}^T \overline{M} = \frac{1}{2^B} \sum_{k=0}^{2^B-1} \mu_k .
\]

(4.37)

Therefore, the reference voltage values of a DEM ADC are equal to those of an ideal ADC plus the constant \( \overline{\mu} \), which is the average comparator input offset error.
4.3.2.3 Steadman’s implementation

One implementation of the resistor string DEM flash ADC uses the architecture in Figure 4.14 with the unit ADC cells in Figure 4.16 [70]. The DEM algorithm’s control signal, $c(n)$, where $c(n) \in \{0, 1, ..., 2^b - 1\}$, is used to control the switches in each of the unit ADC cells. In Figure 4.16, the arrows indicate that the switch is closed when the corresponding signal is a logical one. For example, if $c = 1$ in Figure 4.16, the two switches controlled by the signal $c$ are closed, and the other four switches are open. Using the unit ADC cell in Figure 4.16 in the architecture in Figure 4.14 causes the resistors and the corresponding comparators to be virtually repositioned. The signal $c(n)$ determines which unit ADC cell has switch signals $c = 1$ or $d = 1$. All other unit ADC cells have $c = d = 0$ which closes the current path between adjacent resistors and utilizes the corresponding comparator. If $c = 0$ and $d = 1$, $V_{\text{DD}}$ is connected to cell’s lower output and ground to the cell’s upper output. All other cells in the voltage divider have $c = d = 0$.

![Diagram of the kth resistor string unit ADC cell from [70].](image-url)
This condition connects adjacent resistors, thereby building a voltage divider network. If the switch signal $c$ is high and switch signal $d$ is low, ground is connected to cell’s lower output and $V_{DD}$ to the cell’s upper output. All other cells in the voltage divider have $c = d = 0$. The second voltage divider differs from the first in that the positions of $V_{DD}$ and ground are reversed and the resistors are used in reverse order.

Unlike a standard resistor string flash ADC, the resistor string DEM flash ADC voltage reference corresponding to the $j$th quantization level does not depend uniquely on the first $j$ resistors, $\{R_0, R_1, \ldots, R_{j-1}\}$, and the $j$th comparator offset. The resistor string DEM ADC's $j$th quantization level is determined $j$ consecutive resistors on one side of the $c(n)$th resistor, that is,

$$\{R_{(u-1)\text{mod}2^b}, R_{(u-2)\text{mod}2^b}, \ldots, R_{(u-j)\text{mod}2^b}\} \text{ or } \{R_{u}, R_{(u+1)\text{mod}2^b}, \ldots, R_{(u+j-1)\text{mod}2^b}\} \tag{4.38}$$

where the $u$th unit ADC cell is connected to the power supplies. The first set of resistors in (4.38) are used if $c = 1$, and the second set of resistors in (4.38) are used if $d = 1$.

The DEM ADC constructed with the cell in Figure 4.16 in the architecture in Figure 4.14 can be analyzed by interpreting the $B + 1$ bit control signal, $c(n)$, as follows: The $B$ LSBs of $c(n)$ determine which unit ADC cell is attached to the power supplies and the MSB of $c(n)$ determines the orientation of the resistor string, i.e. it determines whether $c = 1$ or $d = 1$. Thus, the proposed architecture has two possible reference selection transformations given by

$$T_R[k, c(n)] = \begin{cases} \text{rot}\{A_k, c(n)\}, & \text{if } c = 1 \\ \text{rot}\{A_k, 2^B - c(n)\}, & \text{if } d = 1 \end{cases}$$

and a comparator selection transformation given by
\[ T_M[k, c(n)] = \text{rot}\{B, c(n)\}, \]

where \( A_k \) is the \( 2^B \times 1 \) vector with ones in the first \( k \) positions and \( 2^B - k \) zeros in remaining positions, \( B \) is the \( 2^B \times 1 \) vector with a one in the first position and \( 2^B - 1 \) zeros in remaining positions, and \( \text{rot}\{\cdot, n\} \) is the \( n \) position rotation operator. If the control signal, \( c(n) \), has an uniform probability density function then the expected reference selection transformation conditioned on \( k \) is

\[
E\left\{ T_R|k \right\} = \frac{2}{2^B + 1} \sum_{l=0}^{2^B-1} \text{rot}\{A_{k,l}\} = \frac{k}{2^B} \mathbf{1}. \tag{4.39}
\]

Similarly if the control signal, \( c(n) \), has an uniform probability density function then the expected comparator selection transformation conditioned on \( k \) is

\[
E\left\{ T_M|k \right\} = \frac{2}{2^B + 1} \sum_{l=0}^{2^B-1} \text{rot}\{B, l\} = \frac{1}{2^B} \mathbf{1}. \tag{4.40}
\]

Substituting (4.39), (4.40) and (4.35) into (4.32),

\[
E\{L_k\} = k\Delta + \bar{\mu} \tag{4.41}
\]

where \( \bar{\mu} \) is given by (4.37). Therefore, the DEM ADC's reference voltage values are equal to those of an ideal ADC plus a constant \( \bar{\mu} \), which is the average comparator input offset error.

4.4 Current steering flash ADCs

In this section, a current steering flash ADC architecture and a current steering DEM flash ADC architecture are introduced. Using the transistor's inherent nonlinear \( I-V \) relationship, representing ADC signals with current can reduce voltage swings and
increase operating speeds [53]. Because transistors cannot be matched exactly, current mismatch errors occur between transistors. These errors cause nonuniform code widths and create harmonic distortion in the output of current steering flash ADCs. The current steering DEM flash ADC architecture introduced in this section randomizes the mismatched transistors' ordering at each conversion instant and reduces the effects of the mismatched transistors.

4.4.1 Current steering flash ADC architecture

Figure 4.17 shows a block diagram of a B bit current steering flash ADC. Similar to the resistor string flash ADC, the current steering flash ADC is constructed using $2^B$ unit ADC cells, where Figure 4.18 shows the current steering unit ADC cell. In this

![Block Diagram of a B bit Current Steering Flash ADC](image)

**Figure 4.17.** A B bit current steering flash ADC block diagram.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
architecture, the reference current sources generate a set of \(2^B\) reference currents, denoted by \(I_k\) for \(k = 0, 1, \ldots, 2^B - 1\). A current comparator compares the ADC's input current, \(x(nT)\) with each reference current. Figure 4.19 shows a schematic for a CMOS current comparator [53]. In general, the \(k\)th nonideal comparator has a nonzero input offset, \(\mu_k\). Therefore, the output of the \(k\)th comparator can be written as

\[
I_k = kI_{\text{REF}} + \frac{I_{\text{REF}}}{2} + \mu_k
\]

Figure 4.18. The \(k\)th current steering flash unit ADC cell.

Figure 4.19. CMOS current comparator.
The output of the comparators forms a thermometer coded representation of the ADC's output. Digital logic converts the thermometer coded signal into the coded ADC output, \( x_{th}[n] \), which is typically a \( B \) bit binary number.

In the unit ADC cell in Figure 4.18, device mismatches in the PFETs create a comparator reference current of \( g_k I_{REF} \), where \( g_k \) is the current gain of the \( k \)th unit ADC's reference current mirror. If \( \bar{g} \) is defined as the average current gain of the unit ADCs due to PFET mismatches, that is

\[
\bar{g} = \frac{1}{2^B} \sum_{k=0}^{2^B-1} g_k,
\]

then \( g_k = \bar{g} + e_k \), and the comparator's reference current, \( I_k \), is \( (\bar{g} + e_k)(kI_{REF} + I_{REF}/2) \), where \( e_k \) is the reference current gain deviation in the \( k \)th unit ADC. Similarly, device mismatches between the NFET in the unit ADC cell in Figure 4.18 and the NFET in Figure 4.17 create the comparator's input current to be \( h_k x(nT) \), where \( h_k \) is the current gain of the \( k \)th unit ADC's input current mirror. If \( \bar{h} \) is defined as the average current gain due to NFET mismatches, that is

\[
\bar{h} = \frac{1}{2^B} \sum_{k=0}^{2^B-1} h_k,
\]

then \( h_k = \bar{h} + f_k \), and the comparator's reference current is \( (\bar{h} + f_k)x(nT) \), where \( f_k \) is the input current gain deviation in the \( k \)th unit ADC. Therefore, the output of the \( k \)th comparator can be written as

\[
t_k[n] = \begin{cases} 
1, & x(nT) < kI_{REF} + \frac{I_{REF}}{2} + \mu_k \\
0, & x(nT) > kI_{REF} + \frac{I_{REF}}{2} + \mu_k
\end{cases}
\]
4.4.2 Current steering DEM flash ADC architecture

To apply DEM to the current steering flash ADC architecture, all devices creating, copying, or comparing the unit ADC currents must be randomized. In other words, the FETs creating the ADC’s reference currents, the unit ADC’s reference current mirrors, the unit ADC’s input current mirrors, and the current comparators must be randomized. Rearranging the current mirror FETs can reduce the effects due to current mirror mismatches, but will not reduce the conversion errors due to the comparator offset errors. If the comparators’ positions are not randomized, the fixed comparator input offset errors add a deterministic error sequence to the ADC’s output and create harmonic distortion.

Figure 4.20 shows a DEM current steering flash ADC architecture that randomizes the FETs creating the ADC’s reference currents, the unit ADC’s reference current mirrors, the unit ADC’s input current mirrors, and the current comparators. Figure 4.21 shows the unit ADC cell used in Figure 4.20. The ADC’s input, $x(nT)$, is mirrored in each of the unit ADC cells. The ADC’s master reference current is copied by

$$N_R = \sum_{k=0}^{2^B-1} 2k + 1 = 4^B - 2^B + 1$$

current mirrors using NFETs. All $N_R$ NFETs are connected to the switches in the unit ADCs. At a sample instant, the switch control signals, $t_{k,l}$ for $k=0,1,\ldots,2^B-1$ and $l=1,2,\ldots,N_R$, are closed such that each unit ADC implements an unique comparison.
level. To illustrate, a random unit ADC closes one of its switches mirroring $I_{REF}/2$, and this generates the first quantization reference level. A different random unit ADC closes three different switches to mirror $3I_{REF}/2$, and thus generates a second quantization reference level.

Figure 4.20. $B$ bit DEM current steering flash ADC architecture.

Figure 4.21. Current steering flash unit ADC cell.
reference level. A third random unit ADC closes five switches, all of which are different from the switches in the previous two unit ADC cells, to mirror $5I_{REF}/2$, and generates a third quantization reference level. Each unit ADC continues in this manner to create all $2^B$ quantization reference levels which are compared with the input current. At the next sample instant, the unit ADCs obtain new switch signals and the process repeats. Therefore, each quantization decision level is implemented with a different comparator, different copies of the unit reference current and a different copy of the ADC’s input current each sample.

4.4.2.1 Analysis

To analyze the operation of this DEM ADC architecture, consider the $m$th unit ADC that implements the $k$th quantization level comparison. The $m$th unit ADC’s reference current is the sum of $2k+1$ of the $N_R$ reference currents. The $l$th reference current can be written as $I + \epsilon_l$, where the average reference current, $I$, is $I = I_{REF}/2$, and $\epsilon_l$ is the $l$th reference current’s deviation from the average. Due to mismatch errors between the two PFETs, the reference current input of the comparator is

$$(\bar{g} + e_m) \left[ (2k+1)I + \sum_{l=1}^{2k+1} \epsilon_l \right]$$

where $\bar{g}$ is the average current gain of the PFET current mirrors in the $2^B$ unit ADCs and $e_m$ is the gain deviation in the $m$th unit ADC. Similarly, the current mirror that mirrors the input current has a gain due to transistor mismatches. Therefore, the comparator’s input current can be written as

$$(\bar{h} + f_m)x(nT)$$
where \( \bar{h} \) is the average current gain of the NFET current mirrors in the \( 2^B \) unit ADCs and \( f_m \) is the gain deviation in the \( m \)th unit ADC. Figure 4.22 shows the signal processing model for \( m \)th unit ADC cell that implements the \( k \)th quantization level.

The comparator's output, \( t_k \), is given by

\[
    t_k = \begin{cases} 
        1, & \left( \bar{g} + T_{C}[k, c(n)]E \right) \geq \bar{h}x(nT) + x(nT)T_{M}[k, c(n)]\hat{F} + T_{M}[k, c(n)]\hat{M}, \\
        0, & \text{else} 
    \end{cases} \tag{4.44}
\]

where \( E = [e_0, e_1, \ldots, e_{2^B-1}]^T \), \( T_{M}[k, c(n)] \) is the transformation that selects the ADC cell that performs the \( k \)th quantization level when selected by the control signal, \( c(n) \), \( T_{C}[k, c(n)] \) is the transformation that selects the \( k \) currents that create the unit ADC's current reference, \( R = [e_0, e_1, \ldots, e_{2^B-1}]^T \), \( F = [f_0, f_1, \ldots, f_{2^B-1}]^T \) and \( M = [\mu_0, \mu_1, \ldots, \mu_{2^B-1}]^T \). Defining \( \alpha = \bar{g}/\bar{h} \) and rearranging (4.44) gives

![Diagram of signal processing model](image.png)

Figure 4.22. Signal processing model of the unit ADC cell in Figure 4.21.
From (4.45), the expected value of the ADC’s $k$th reference current can be written as

$$E \{ L_k \} = x(nT) \leq \alpha k I + \alpha T_R^T[k, c(n)]R + (1/\eta) T_M^T[k, c(n)]E +$$

$$- \frac{x(nT)T_M^T[k, c(n)]F}{\eta} + \frac{T_M^T[k, c(n)]M}{\eta}.$$  

(4.46)

4.4.2.2 Stochastic barrel shifting DEM implementation

As an example, consider the DEM flash ADC architecture that uses the ADC cell in Figure 4.21 and a stochastic DEM algorithm that performs stochastic barrel shifting. For this DEM ADC, the reference selection transformation is

$$T_R[k, c(n)] = \text{rot}\{A_k, c(n)\},$$

and the comparator selection transformation is

$$T_M[k, c(n)] = \text{rot}\{B, c(n)\},$$

where $A_k$ is the $N_R \times 1$ vector has ones in $k$ consecutive rows starting at the row given by $\sum_{l=1}^{k} \ell$ and $2^B - k$ zeros in the remaining rows, $B$ is the $2^B \times 1$ vector that has a one in the first row and $2^B - 1$ zeros in the remaining rows, and $\text{rot}\{*, n\}$ is the $n$ position rotation operator. If the control signal, $c(n)$, has an uniform probability density function then the expected reference selection transformation conditioned on $k$ is

$$E \{ T_R | k \} = \frac{1}{N_R} \sum_{l=0}^{N_R-1} \text{rot}\{A_k, l\} = \frac{k}{N_R} \mathbf{1}.$$  

(4.47)
Similarly if the control signal, $c(n)$, has an uniform probability density function then the expected comparator selection transformation conditioned on $k$ is

$$E\{ T_M | k \} = \frac{1}{2^B} \sum_{l=0}^{2^B-1} \text{rot}\{ B, l \} = \frac{1}{2^B} 1.$$  \hfill (4.48)

Because $g_k = \bar{g} + e_k$, the average reference current gain in Figure 4.21 can be written as

$$\bar{g} = \frac{1}{2^B} \sum_{k=0}^{2^B-1} \bar{g} + e_k$$

which implies that

$$\sum_{k=0}^{2^B-1} e_k = 1^T E = 0.$$  \hfill (4.49)

Similarly, it can be shown that

$$\sum_{k=0}^{2^B-1} f_k = 1^T F = 0.$$  \hfill (4.50)

and

$$\sum_{k=0}^{N_k-1} e_k = 1^T R = 0.$$  \hfill (4.51)

Substituting (4.47), (4.48), and (4.49) into (4.46) implies

$$E\{ L_k \} = \alpha k T + \alpha \frac{k}{N_R} 1^T R + \frac{1}{h} + \frac{k}{N_R} 1^T R \frac{1}{2^B} 1^T E + \frac{x(nT)}{2^B} \frac{1}{h} + \frac{1}{h} + \frac{1}{2^B} 1^T M \frac{1}{h} + \frac{1}{h},$$  \hfill (4.52)

and substituting (4.50) and (4.51) into (4.52),

$$E\{ L_k \} = \alpha k T + \bar{\mu}/\bar{h}$$  \hfill (4.53)

where
\[
\mu = \frac{1}{2^B} \sum_{k=0}^{2^B-1} \mu_k.
\]

(4.54)

Therefore, DEM ADC’s reference voltage values are equal to those of an ideal ADC plus a constant \(\mu/\bar{n}\), which is due to the average comparators input offset error and the average current gain of the NFET current mirrors.

### 4.5 Conclusions

An ideal ADC can be linearized by adding a dither signal with a proper probability density function and amplitude to its input. No harmonic distortion is present in an ideal ADC’s output under these conditions. However, practical ADCs do not possess uniform quantization intervals, and consequently, harmonic distortion is present in the ADC’s output even if an appropriate dither signal is employed. In this chapter, a generic ADC architecture using dynamic element matching techniques was analyzed. The architecture can reduce ADC output errors due to component mismatch errors, such as inaccurate reference sources and nonideal comparator. The ADC’s SDR and SNDR can be used to compare different DEM algorithms. Computer simulations of DEM algorithms, existing and new, have been presented that support the theoretical findings. Finally, analysis and circuit implementations of resistor string and current steering DEM flash ADCs were presented.
CHAPTER 5

DIGITAL TO ANALOG CONVERTER TRANSFER FUNCTION ERRORS AND HARMONIC DISTORTION

A linear digital to analog converter (DAC) transforms a $B$-bit digital signal, $x[n]$, into an analog output signal, $y(nT)$, such that $y(nT) = qx[n]$ where $x[n]$ is the DAC's digital input signal, $T$ is the DAC's sampling period, $q$ is a constant that represents the DAC's quantization step sizes or code widths, and $n$ is an integer that indexes the sequence $x$. Many DAC architectures use matched components to convert signals. Because of variations in circuit fabrication processes, temperature gradients across the circuit, component aging, and component noise, circuit component values differ from their design values. As a result of these variations, called mismatch errors, DAC code widths are functions of $x[n]$ and not constants [15], [31], [59], [65]. Variable code widths that are a function of $x[n]$ add a nonlinear transformation, referred to as integral nonlinearity (INL), to the DAC's linear transformation. Nonzero INL generates harmonic distortion in a DAC's output. This harmonic distortion reduces a DAC's spurious free dynamic range (SFDR), signal to distortion ratio (SDR), and signal to noise plus distortion ratio (SNDR).

In this chapter, a Fourier analysis technique is developed that relates a DAC's transfer function to its frequency spectrum. This technique can also be used to relate a DAC's INL to its harmonic distortion. Using these analyses, a DAC's SFDR, SDR and SNDR can be
calculated from its transfer function. In this chapter, a technique is also developed that determines a DAC's transfer function from measured spectra.

5.1 A generalized Fourier series representation of a DAC's output

If the transformation $T$ represents a $B$ bit DAC's transfer function, the DAC's output, $y(nT)$, can be written as

$$y(nT) = T[x[n]]$$

where $T$ is a nonlinear transformation that can be described as

$$T = \{ T[x[n]] = \bar{q}x[n] + \text{INL}_{x[n] - x_0} : 0 \leq x[n] - x_0 \leq 2^B - 1 \}$$

(5.1)

$\bar{q}$ is the DAC's average code width, $\bar{q}x_0$ is the DAC's zero input offset, $x[n]$ is the DAC's digital input sequence that has been amplitude shifted so that its minimum value equals $x_0$, and $\text{INL}_{x[n] - x_0}$ is the DAC's INL at $x[n] = k$. The DAC's output, $y(nT)$, can also be written as

$$y(nT) = T[x[n]] = T_l[x[n]] + T_e[x[n]]$$

(5.2)

where $T_l$ is the linear transformation that maps $x[n]$ to the DAC's linear output, $y_l(nT)$, and $T_e$ is the nonlinear transformation that maps $x[n]$ to the output's conversion errors, $y_e(nT)$. The DAC's linear and nonlinear transformations can be written

$$T_l = \{ T_l[x[n]] = \bar{q}x[n] : 0 \leq x[n] - x_0 \leq 2^B - 1 \}$$

$$= \{ \bar{q}x_0, \bar{q}(x_0 + 1), ..., \bar{q}(x_0 + 2^B - 1) \}$$

and
respectively.

For a $B$-bit DAC, each of the three transformations, $T$, $T_I$, and $T_e$, maps $2^B$ inputs to $2^B$ outputs, and therefore each of the transformations can be represented by a finite length sequence of length $2^B$. Because each of these transformations can be represented by a finite length sequence, they can be represented by a linear combination of a complete set of mutually orthogonal sequences. In particular, the DAC's output, $y(nT)$, and nonlinear transformation, $T$, can be written as

$$y(nT) = T[x[n]] = \sum_{l=0}^{M-1} A[l]\varphi_{l}[x[n]] - x_0$$

where

$$A[l] = \frac{\sum_{z[n]=0}^{2^B-1} T[z[n] + x_0]\varphi_{l}^{*}[z[n]]}{\sum_{z[n]=0}^{2^B-1} \varphi_{0}[z[n]]\varphi_{l}^{*}[z[n]]}$$

$\varphi_{l}[z[n]]$ is a complete set of $M$ mutually orthogonal sequences such that

$$\sum_{z[n]=0}^{2^B-1} \varphi_{r}[z[n]]\varphi_{l}^{*}[z[n]] = 0 \text{ for } r \neq l$$

and $\varphi_{l}^{*}[z[n]]$ is the complex conjugate of $\varphi_{l}[z[n]]$.

If the input signal, $x[n]$, is a finite length sequence of length $L$, then each of the $\varphi_{l}[x[n] - x_0]$ for $0 \leq l \leq M-1$ is a finite length sequence of length $L$, and each of the $\varphi_{l}[x[n] - x_0]$ for $0 \leq l \leq M-1$ can be expressed as the linear combination of a complete set
of mutually orthogonal sequences. Similarly, if the input signal, $x[n]$, is periodic with period $L$, then each of the $\varphi_i[x[n]-x_0]$ for $0 \leq i \leq M-1$ is periodic with period $L$, and each of the $\varphi_i[x[n]-x_0]$ for $0 \leq i \leq M-1$ can be expressed as the linear combination of a complete set of mutually orthogonal periodic sequences that have a period $L$. Therefore, if $x[n]$ is a finite length sequence of length $L$ or a periodic sequence with period $L$, each of the $\varphi_i[x[n]-x_0]$ for $0 \leq i \leq M-1$ can be written as

$$\varphi_i[x[n]-x_0] = \sum_{k=0}^{N-1} C[k, i]Y_k[n]$$  \hspace{1cm} (5.6)

where

$$C[k, i] = \frac{\sum_{n=0}^{L-1} \varphi_i[x[n]-x_0]Y_k[n]}{\sum_{n=0}^{L-1} Y_k[n]Y_k^*[n]},$$  \hspace{1cm} (5.7)

$Y_k[n]$ is a complete set of $N$ mutually orthogonal sequences such that

$$\sum_{n=0}^{L-1} Y_r[n]Y_k^*[n] = 0 \quad r \neq k$$

and $Y_k^*[n]$ is the complex conjugate of $Y_k[n]$. Substituting (5.6) into (5.3),

$$y(nT) = \sum_{l=0}^{M-1} A[l] \sum_{k=0}^{N-1} C[k, l]Y_k[n] = \sum_{k=0}^{N-1} \sum_{l=0}^{M-1} C[k, l]A[l]Y_k[n].$$  \hspace{1cm} (5.8)

By defining $Y[k]$ as

$$Y[k] = \sum_{l=0}^{M-1} C[k, l]A[l].$$  \hspace{1cm} (5.9)

the DAC's output in (5.8) can be written as
where the summation in (5.10) has the form of a generalized Fourier series. By defining the column vector, $Y$, such that $Y[k]$ is the $k$th element of $Y$, the generalized Fourier coefficients in (5.9) can also be calculated using

$$Y = CA$$

(5.11)

where $C[k, l]$ is the element in the $k$th row and $l$th column in the matrix, $C$, and $A[l]$ is the $l$th element in the column vector, $A$. Similar arguments can be used to determine generalized Fourier coefficients for $y_j(nT)$. Because $T_j$ is a linear transformation, the generalized Fourier coefficients for $y_j(nT)$ can be determined in the usual manner.

5.2 The relationship between a DAC's transformation and its output's frequency spectrum

To determine the frequency spectrum of the DAC's output using (5.10), the complete set of mutually orthogonal sequences, $y_k[n]$ for $0 \leq k \leq N-1$, are chosen to correspond to desired frequency information. Therefore, let

$$y_k[n] = \frac{1}{N} e^{\frac{2\pi j kn}{N}}$$

(5.12)

for $0 \leq k \leq N-1$, which implies that (5.10) can be written as

$$y(nT) = T[x[n]] = \frac{1}{N} \sum_{k=0}^{N-1} Y[k] e^{\frac{2\pi j kn}{N}}.$$  

(5.13)

Although $\phi_i[x[n] - x_0]$ can be selected as any complete set of mutually orthogonal sequences, it is convenient to let

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
\[ \varphi_i[x[n] - x_0] = \frac{1}{2B} e^{\frac{2\pi i x_0}{2B}} e^{\frac{2\pi i n}{N}} \]

which implies that

\[ C[k, l] = \frac{1}{2B} \sum_{n=0}^{N-1} e^{\frac{2\pi i x_0}{2B}} e^{-\frac{2\pi i n}{N}} e^{-j\frac{2\pi k n}{N}}. \]  \hspace{1cm} (5.14)

and

\[ A[l] = \sum_{z[n]=0}^{2^g - 1} T[z[n] + x_0] e^{-\frac{2\pi i z[n]}{2^g}}. \]  \hspace{1cm} (5.15)

Substituting (5.14) and (5.15) into (5.9),

\[ Y[k] = \frac{1}{2^B} \sum_{l=0}^{2^g - 1} \sum_{n=0}^{N-1} e^{\frac{2\pi i x_0}{2B}} e^{-\frac{2\pi i n}{N}} \sum_{z[m]=0}^{2^g - 1} T[z[m] + x_0] e^{-\frac{2\pi i z[m]}{2^g}}. \]  \hspace{1cm} (5.16)

The Fourier coefficients, \( Y[k] \) for \( 0 \leq k \leq N - 1 \), can also be calculated using (5.11) where

\[ C = \begin{bmatrix} DFS_N \{ \varphi_0[x[n] - x_0] \} \\ DFS_N \{ \varphi_1[x[n] - x_0] \} \\ \vdots \\ DFS_N \{ \varphi_{2^g-1}[x[n] - x_0] \} \end{bmatrix} \]

\[ = \begin{bmatrix} DFS_N \{ \frac{1}{2B} e^{\frac{2\pi i x_0}{2B}} e^{\frac{2\pi i n}{N}} \} \\ DFS_N \{ \frac{1}{2B} e^{\frac{2\pi i x_0}{2B}} e^{\frac{2\pi i (2^g - 1) n}{N}} \} \end{bmatrix} \]

where \( DFS_N \{ \varphi_0[x[n] - x_0] \} \) generates a column vector containing the \( N \) point discrete Fourier series (DFS) coefficients of \( \varphi_0[x[n] - x_0] \), and

\[ A = [DFS_{2^g} \{ T[z[n]] : 0 \leq z[n] - x_0 \leq 2^B - 1 \}] \]  \hspace{1cm} (5.17)

where \( DFS_{2^g} \{ T[z[n]] \} \) generates a column vector containing the \( 2^B \) point DFS coefficients of \( T[z[n]] : 0 \leq z[n] - x_0 \leq 2^B - 1 \). Using (5.11) or (5.16), the frequency spectrum of the nonlinear DAC's output can be determined.
By defining the column vector, $Y_f$, as the DFS coefficients of $y_f(nT)$ and the column vector, $Y_e$, as the DFS coefficients of $y_e(nT)$, the DFS coefficients vectors, $Y_f$ and $Y_e$, can be calculated using $Y_f = CA_f$ and $Y_e = CA_e$, where

$$A_f = \left[\text{DFS}_2^T(\tau_n[n]:0 \leq n \leq 2^B - 1)\right]$$

and

$$A_e = \left[\text{DFS}_2^T(\tau_e[n]:0 \leq n \leq 2^B - 1)\right].$$

The DFS coefficients vector, $Y_f$, describes the spectrum of the DAC's output signal and the DFS coefficients vector, $Y_e$, describes the spectrum of the DAC's harmonic distortion. Because $A = A_f + A_e$, (5.11) can be written as

$$Y = C(A_f + A_e) = Y_f + Y_e. \quad (5.19)$$

Because $T_f$ is a linear transformation, $Y_f$ can also be written as

$$Y_f = \bar{q}X, \quad (5.20)$$

where $X$ is a column vector that contains the DFS coefficients of $x[n]$. Substituting (5.20) into (5.19), $Y$ can also be written as

$$Y = \bar{q}X + CA_e. \quad (5.21)$$

5.3 Performance criteria for DACs

Three criteria that are used to measure a DAC's performance are SFDR, SDR, and SNDR. A DAC's SFDR can be calculated directly using (5.11) or (5.16). To calculate a DAC's SDR, consider the DAC's average signal plus distortion power, $P_y$, where

$$P_y = \frac{1}{N^2}Y^HY \quad (5.22)$$
and the superscript $H$ denotes the complex conjugate transpose. Substituting (5.21) into (5.22),

$$P_y = \frac{1}{N^2} \bar{q}^2 X^H X + \frac{1}{N^2} (2\bar{q} \text{Re}(X^H CA_e) + A_e C^H CA_e)$$

where the first term, $\frac{1}{N^2} \bar{q}^2 X^H X$, is the DAC's average output signal power, $P_1$, and the second term, $\frac{1}{N^2} (2\bar{q} \text{Re}(X^H CA_e) + A_e C^H CA_e)$, is the DAC's average distortion power, $P_d$. Therefore, the DAC's SDR is

$$\text{SDR} = \frac{\bar{q}^2 X^H X}{2\bar{q} \text{Re}(X^H CA_e) + A_e C^H CA_e}$$

(5.23)

To calculate a DAC's SNDR, consider an input, $x[n]$, which can be written as

$$x[n] = s[n] + w[n]$$

where $s[n]$ is the DAC's input signal and $w[n]$ is the DAC's input noise. Because $T_i$ is a linear transformation, the DAC's output, $y(nT)$, in (5.2) can be written as

$$y(nT) = T[s[n] + w[n]] = T_i[s[n]] + T_i[w[n]] + T_e[s[n]] + w[n],$$

and (5.21) can be written as

$$Y = \bar{q}S + \bar{q}W + Y_e.$$  

(5.24)

where $S$ and $W$ are vectors that contain the DFS coefficients of $s[n]$ and $w[n]$, respectively. To calculate a DAC's SNDR, consider the DAC's average signal plus noise plus distortion power, $P_y$. Substituting (5.24) into (5.22),

$$P_y = \frac{1}{N^2} \bar{q}^2 S^H S + \frac{1}{N^2} (\bar{q}^2 W^H W + A_e C^H CA_e) + \frac{1}{N^2} 2\bar{q} \text{Re}(\bar{q} S^H W + S^H CA_e + W^H CA_e)$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
where the first term, $\frac{1}{N^2}q^2s^Hs$, is the DAC's average output signal power, and the remaining terms are the DAC's average noise plus distortion power. Therefore, the DAC's SNDR is

\[
\text{SNDR} = \frac{q^2s^Hs}{q^2W^HW + A^HCGCA_e + 2q\Re\{q\overline{s^H}W + s^HCA_e + W^HC^A_e\}}.
\] (5.25)

5.4 Relating a DAC output's frequency spectrum to its INL

A DAC's transfer function can be determined from measured spectrum of its output. To determine a DAC's transformation, $T$, from its frequency spectrum, $Y$, pre multiply (5.11) by $(C^HC)^{-1}C^HY$ which implies that

\[
A = (C^HC)^{-1}C^HY
\] (5.26)

and perform an inverse DFS (IDFS) of the result. In other words,

\[
T[x[n]] = \text{IDFS}(A) = \text{IDFS}\{(C^HC)^{-1}C^HY\}.
\] (5.27)

To calculate a DAC's transformation, $T$, from the output frequency spectrum, $Y$, (5.27) assumes that both magnitude and phase information of the DAC's output frequency spectrum, $Y$, are known. In many circumstances, only magnitude information is measured and available. In such cases, (5.11) can be written as

\[
\begin{bmatrix}
|Y[0]|e^{j\angle Y[0]} \\
|Y[1]|e^{j\angle Y[1]} \\
|Y[2]|e^{j\angle Y[2]} \\
\vdots \\
|Y[N-1]|e^{j\angle Y[N-1]}
\end{bmatrix}
- C
= \begin{bmatrix}
|A\{0]\|e^{j\angle A[0]} \\
|A\{1]\|e^{j\angle A[1]} \\
|A\{2]\|e^{j\angle A[2]} \\
\vdots \\
|A\{2^b-1\}||e^{j\angle A[2^b-1]}
\end{bmatrix}
\] (5.28)
where $|Y[k]|$ and $\angle Y[k]$ are the measured magnitude and the unknown phase of the output's $k$th frequency sample, respectively, and $|A[l]|$ and $\angle A[l]$ are the measured magnitude and the unknown phase of the output's $l$th frequency sample when $x[n] = n \mod 2^B$, respectively. (If $x[n] = n \mod 2^B$, $A$ contains the DFS coefficients of the DAC's transfer function.) To determine the DAC's transformation, the phases, $\angle A[l]$ for $0 \leq l \leq 2^B - 1$, of $A$ must be determined from the measured magnitudes, $|Y[k]|$ for $0 \leq k \leq N-1$ and $|A[l]|$ for $0 \leq l \leq 2^B - 1$.

An iterative algorithm that can determine the phase information of $Y$ and $A$ from the measured magnitudes, $|Y[k]|$ for $0 \leq k \leq N-1$ and $|A[l]|$ for $0 \leq l \leq 2^B - 1$, consists of four steps. To start, the algorithm initially requires an estimate, $\tilde{A}$, of $A$. To generate this initial estimate, let $|\tilde{A}[l]| = |A[l]|$ for $0 \leq l \leq 2^B - 1$ where $|A[l]|$ for $0 \leq l \leq 2^B - 1$ are measured magnitudes. The phases, $\angle Y[k]$ for $0 \leq k \leq N-1$, can be set to any real number, or they can be calculated assuming that the DAC is ideal. The algorithm's first step calculates an estimate, $\hat{Y}$, of $Y$ using $\tilde{A}$ and (5.11). The algorithm's second step generates an improved estimate, $\tilde{Y}$, of $Y$ by replacing the magnitudes in $\hat{Y}$ with the measured magnitudes, $|Y[k]|$ for $0 \leq k \leq N-1$. Using $\tilde{Y}$ and (5.26), the algorithm's third step calculates an estimate, $\hat{A}$, of $A$. The fourth step generates an improved estimate, $\tilde{A}$, of $A$ by replacing the magnitudes in $\hat{A}$ with the measured magnitudes, $|A[l]|$ for $0 \leq l \leq 2^B - 1$. Using $\tilde{A}$ and (5.11), the algorithm iterates by calculating a new estimate, $\hat{Y}$, of $Y$. Mathematically, the $i$th iteration of this algorithm can be written as

$$\hat{Y}_i = C\tilde{A}_{i-1} \quad \text{where} \quad \hat{Y}_i[k] = |\hat{Y}_i[k]|e^{j\hat{\angle}Y_i[k]} \quad \text{for} \quad 0 \leq k \leq N-1$$

(5.29)

$$\tilde{Y}_i[k] = |Y[k]|e^{j\tilde{\angle}Y_i[k]} \quad \text{for} \quad 0 \leq k \leq N-1$$

(5.30)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
\( \hat{A}_i = (C^H C)^{-1} C^H \tilde{Y}_i \) where \( \tilde{A}_i[l] = |\tilde{A}_i[k]| e^{j\tilde{\lambda}_i[k]} \) for \( 0 \leq l \leq 2^B - 1 \) \( (5.31) \)

\( \tilde{A}_i[l] = |A[l]| e^{j\lambda_i[l]} \) for \( 0 \leq l \leq 2^B - 1 \) \( (5.32) \)

These four steps are repeated until the error in the estimates of \( A \) and \( Y \) are within a desired tolerance.

To show that the iterative algorithm described in (5.29)-(5.32) converges, define \( \varepsilon_{Y,i}^2 \) as

\[
\varepsilon_{Y,i}^2 = \frac{1}{N^2} \sum_{k=0}^{N-1} |\tilde{Y}_i[k] - \hat{Y}_i[k]|^2,
\]

which is the average power of the error between the \( i \)th estimate, \( \hat{Y}_i \), of \( Y \) and the \( i \)th improved estimate, \( \tilde{Y}_i \), of \( Y \), and define \( \varepsilon_{A,i}^2 \) as

\[
\varepsilon_{A,i}^2 = \frac{1}{2^{2B}} \sum_{l=0}^{2^B - 1} |\tilde{A}_i[l] - \hat{A}_i[l]|^2
\]

which is the average power of the error between the \( i \)th estimate, \( \hat{A}_i \), of \( A \) and the \( i \)th improved estimate, \( \tilde{A}_i \), of \( A \). If it can be shown that

\( \varepsilon_{Y,i+1}^2 \leq \varepsilon_{A,i}^2 \leq \varepsilon_{Y,i}^2 \),

both of the \((i+1)\)st mean squared errors are less than or equal to both of the \( i \)th mean squared errors, and the iterative algorithm described in (5.29)-(5.32) converges to a solution.

Lemma 1: Let \( \Phi_i = [\varphi_i[x[0]-x_0] \varphi_i[x[1]-x_0] \ldots \varphi_i[x[N-1]-x_0]^T \) where \( \varphi_i[x[n]-x_0] \) is defined in (5.5) and the superscript \( T \) denotes the transpose operation. If the DAC's input, \( x(n) \), has the property that
Proof: If \( 0 \leq b_0 - x_0 \leq 2^B - 1 \) and \( P(x[n] = b_0) = 1/2^B \), the elements of the sequence \( x[n] \) take on every possible valid DAC input an equal number of times, and thus, the length, \( N \), of \( x(n) \) is \( N = \alpha 2^B \) where \( \alpha \) is an integer. Therefore,

\[
\phi_r^H \varphi_l = \sum_{n=0}^{N-1} \phi_r^* [x[n] - x_0] \phi_l [x[n] - x_0] = \alpha \sum_{z[n]=0}^{2^B-1} \phi_r^* [z[n]] \phi_l [z[n]]. \quad (5.35)
\]

If \( r \neq l \) (5.5) implies that \( \phi_r^H \varphi_l = 0 \). Q.E.D.

**Result 1:**
If \( \phi_l [x[n] - x_0] = \frac{1}{2^B} e^{j 2\pi r [x[n] - x_0]} \) for \( 0 \leq l \leq 2^B - 1 \),

\[
\phi_r^H \varphi_l = \begin{cases} 
N/2^B & r = l \\
0 & r \neq l
\end{cases}
\]

As shown in Lemma 1, \( \phi_r^H \varphi_l = 0 \) when \( r \neq l \) and \( \alpha = N/2^B \). Therefore, when \( r = l \), (5.35) implies that

\[
\phi_r^H \varphi_l = \frac{\alpha}{2^{2B}} \sum_{z[n]=0}^{2^B-1} e^{-j \frac{2\pi}{2^B} z[n]} e^{j \frac{2\pi}{2^B} r z[n]} = \frac{N}{2^{2B}}.
\]

**Lemma 2:** If \( C \) is the \( 2^B \times N \) matrix described by (5.7) and (5.11) where \( \gamma_k[n] = \frac{1}{N} e^{j 2\pi k n} \) for \( 0 \leq k \leq N-1 \), and the DAC's input, \( x(n) \), has the property that \( P(x[n] = b_0) = 1/2^B \) where \( 0 \leq b_0 - x_0 \leq 2^B - 1 \) then \( C^H C \) is a diagonal matrix.

Proof: Again, let \( \varphi_l = [\phi_l [x[0] - x_0] \phi_l [x[1] - x_0] \ldots \phi_l [x[N-1] - x_0]]^T \) where \( \phi_l [x[n] - x_0] \) is defined in (5.5), and define

\[
\gamma_k = \frac{1}{N} \left[ e^{j \frac{2\pi}{N} k} e^{j \frac{2\pi}{N} k} \ldots e^{j \frac{2\pi}{N} k(N-1)} \right]^T. \quad \text{If the matrix, } V, \text{ is defined as}
\]
\[ \mathbf{V} = \begin{bmatrix} 
\gamma_0 & \gamma_1 & \cdots & \gamma_{N-1} \\
\gamma_0^H & \gamma_1^H & \cdots & \gamma_{N-1}^H 
\end{bmatrix} = N \begin{bmatrix} 
\gamma_0 & \cdots & \gamma_{N-1} 
\end{bmatrix}, \quad (5.36) \]

the matrix \( \mathbf{C} \) can be written as \( \mathbf{C} = \begin{bmatrix} 
\mathbf{V}^H \phi_0 \mathbf{V} & \mathbf{V}^H \phi_1 & \cdots & \mathbf{V}^H \phi_{2^s-1} 
\end{bmatrix} \), and \( \mathbf{C}^H \mathbf{C} \) can be written as

\[
\mathbf{C}^H \mathbf{C} = \begin{bmatrix} 
\phi_0 \mathbf{V}^H \mathbf{V} \phi_0 & \phi_0 \mathbf{V}^H \mathbf{V} \phi_1 & \cdots & \phi_0 \mathbf{V}^H \mathbf{V} \phi_{2^s-1} \\
\phi_1 \mathbf{V}^H \mathbf{V} \phi_0 & \phi_1 \mathbf{V}^H \mathbf{V} \phi_1 & \cdots & \phi_1 \mathbf{V}^H \mathbf{V} \phi_{2^s-1} \\
\vdots & \vdots & \ddots & \vdots \\
\phi_{2^s-1} \mathbf{V}^H \mathbf{V} \phi_0 & \phi_{2^s-1} \mathbf{V}^H \mathbf{V} \phi_1 & \cdots & \phi_{2^s-1} \mathbf{V}^H \mathbf{V} \phi_{2^s-1} 
\end{bmatrix}
\]

Using (5.36), \( \mathbf{V} \mathbf{V}^H = N^2 \sum_{k=0}^{N-1} \gamma_k \gamma_k^H \) which implies that \( \mathbf{V} \mathbf{V}^H \), the \( r \)-th row and \( c \)-th column of \( \mathbf{V} \mathbf{V}^H \), can be written as

\[
\mathbf{V}_{rc} = \sum_{k=0}^{N-1} e^{j\frac{2\pi kr}{N}} e^{-j\frac{2\pi kc}{N}} = \begin{cases} 
N & r = c \\
0 & r \neq c 
\end{cases}
\]

Therefore, \( \mathbf{V} \mathbf{V}^H = N \mathbf{I}_N \) where \( \mathbf{I}_N \) is the \( N \times N \) identity matrix. Using this result and Lemma 1,

\[
\mathbf{C}^H \mathbf{C} = N \text{diag} \left\{ \phi_0^H \phi_0, \phi_1^H \phi_1, \cdots, \phi_{2^s-1}^H \phi_{2^s-1} \right\}
\]

where \( \text{diag}\{x\} \) is a diagonal matrix that has the elements of the vector \( x \) along its diagonal. Q.E.D.

**Result 2:** If \( \phi_l[x[n] - x_0] = \frac{1}{2^B} e^{j\frac{2\pi}{2^B}[x[n] - x_0]} \) for \( 0 \leq l \leq 2^B - 1 \) and \( \gamma_k[n] = \frac{1}{N} e^{j\frac{\pi}{N} kn} \) for \( 0 \leq k \leq N-1 \), which implies that \( \mathbf{C} \) is the matrix described by (5.17), then \( \mathbf{C}^H \mathbf{C} = \frac{N^2}{2^B} \mathbf{I}_{2^B} \). This result can be obtained using Lemma 2 and Result 1.
Theorem 1: If \( \Phi[x[n] - x_0] = \frac{1}{2^B} e^{j\frac{2\pi}{2^B}[x[n] - x_0]} \) for \( 0 \leq i \leq 2^B - 1 \), \( \gamma_k[n] = \frac{1}{N} e^{j\frac{2\pi}{N}kn} \) for \( 0 \leq k \leq N - 1 \), and the DAC's input, \( x[n] \), has the property that \( P[x[n] = b_0] = 1/2^B \) where \( P \) is the probability function and \( 0 \leq b_0 - x_0 \leq 2^B - 1 \), then \( \varepsilon_{\gamma,i+1}^2 \leq \varepsilon_{A,i}^2 \leq \varepsilon_{\gamma,i}^2 \).

Proof: In matrix form, (5.33) can be written as

\[
\varepsilon_{\gamma,i}^2 = \frac{1}{N^2} \| \tilde{Y}_i - \tilde{Y}_i \|^2 ,
\]

and (5.34) can be written as

\[
\varepsilon_{A,i}^2 = \frac{1}{2^{2B}} \| \tilde{A}_i - \tilde{A}_i \|^2
\]

where \( \| \cdot \| \) denotes the Euclidean, or Frobenius, vector-matrix norm [41].

Substituting (5.29) into (5.37) and noting that (5.31) implies that 
\( \tilde{Y}_i = C \tilde{A}_i \),

\[
\varepsilon_{\gamma,i}^2 = \frac{1}{N^2} \| \tilde{Y}_i - \tilde{Y}_i \|^2 = \frac{1}{N^2} \| C(\tilde{A}_{i-1} - \tilde{A}_i) \|^2 = \frac{1}{N^2} (\tilde{A}_{i-1} - \tilde{A}_i)^H C^H C(\tilde{A}_{i-1} - \tilde{A}_i)
\]

Substituting Result 2 into (5.39),

\[
\varepsilon_{\gamma,i}^2 = \frac{1}{2^{2B}} (\tilde{A}_{i-1} - \tilde{A}_i)^H (\tilde{A}_{i-1} - \tilde{A}_i) = \frac{1}{2^{2B}} \| \tilde{A}_{i-1} - \tilde{A}_i \|^2
\]

Because the elements of \( \tilde{A}_i \) and \( \tilde{A}_i \) have identical phases and the elements of \( \tilde{A}_i \) and \( \tilde{A}_{i-1} \) have identical magnitudes, \( |\tilde{A}_i[l] - \tilde{A}_i[l]| \leq |\tilde{A}_{i-1}[l] - \tilde{A}_i[l]| \) for \( 0 \leq l \leq 2^B - 1 \) which implies that

\[
\| \tilde{A}_i - \tilde{A}_i \|^2 \leq \| \tilde{A}_{i-1} - \tilde{A}_i \|^2
\]

Applying (5.41) to (5.38) and (5.40),
\[ e_{\text{A},i}^2 \leq e_{\text{Y},i}^2. \] (5.42)

Similarly, because the elements of \( \hat{Y}_{i+1} \) and \( \tilde{Y}_{i+1} \) have identical phases and the elements of \( \hat{Y}_{i+1} \) and \( \tilde{Y}_i \) have identical magnitudes,

\[ \| \hat{Y}_{i+1}[k] - \tilde{Y}_{i+1}[k] \| \leq \| \hat{Y}_{i+1}[k] - \tilde{Y}_i[k] \| \text{ for } 0 \leq k \leq N - 1 \]

which implies that

\[ \| \hat{Y}_{i+1} - \tilde{Y}_{i+1} \| \leq \| \hat{Y}_{i+1} - \tilde{Y}_i \| \] (5.43)

Applying (5.43) and (5.29) to (5.37),

\[ e_{\text{Y},i+1}^2 = \frac{1}{N^2} \| \hat{Y}_{i+1} - \tilde{Y}_{i+1} \|^2 \leq \frac{1}{N^2} \| \hat{Y}_{i+1} - \tilde{Y}_i \|^2 = \frac{1}{N^2} \| C(\tilde{A}_i - \hat{A}_i) \|^2 \]

which implies that

\[ e_{\text{Y},i+1}^2 \leq \frac{1}{N^2} (\tilde{A}_i - \hat{A}_i)^H C^H C (\tilde{A}_i - \hat{A}_i) \] (5.44)

Substituting Result 2 into (5.44)

\[ e_{\text{Y},i+1}^2 \leq \frac{1}{22B} (\tilde{A}_i - \hat{A}_i)^H (\tilde{A}_i - \hat{A}_i) = \frac{1}{22B} \| \tilde{A}_i - \hat{A}_i \|^2 = e_{\text{A},i}^2. \] (5.45)

Equations (5.42) and (5.45) imply that \( e_{\text{Y},i+1}^2 \leq e_{\text{A},i}^2 \leq e_{\text{Y},i}^2 \). Q.E.D.

Theorem 1 proves that the iterative algorithm described in (5.29)-(5.32) converges; however, the solution to which it converges is not a unique solution because the DC frequencies, \( Y[0] \) and \( A[0] \), are real numbers and their phases, \( \angle Y[0] \) and \( \angle A[0] \), can be either zero or \( \pi \). As a result, the iterative algorithm described in (5.29)-(5.32) can converge to a result which generates the correct transfer function shifted by \( 2|A[0]|/2^B \). In the author’s experience, converging to the wrong solution can be avoided by improving the initial estimate of \( |A| \).
5.5 Example

Consider a six bit DAC that has linear gradient and uniformly distributed random unit DAC errors where the linear gradient errors vary linearly from +5% to -5% of an LSB and the random errors are uniformly distributed between +2% and -2% of an LSB. Table 5.1 contains the 64 points that describe the DAC's nonlinear transformation, T. For this particular example, the DAC's input sequence is a full scale dithered sinusoid that has a frequency of $\frac{7\pi}{64}$ radians/sample. The dither sequence is a strictly white sequence with a triangular probability distribution function with support on $(-q,q)$.

Quantizing the dithered sinusoidal input sequence to six bits, (5.20) can be used to calculate the ideal DAC's power spectral density (PSD) which is shown in Figure 5.1. Because $Y = qX$, the PSD in Figure 5.1 is identical to a scaled PSD of the DAC's digital input sequence. Using the quantized input and the transfer function in Table 5.1, (5.11) or (5.16) can be used to determine the DAC's output PSD, which is plotted in Figure 5.2. Using Figure 5.2, the DAC's SFDR is measured to be 38.5 dB. The spectrum, $Y_e$, of the DAC's distortion, $y_e(nT)$, can be calculated by substituting $A_e$ for $A$ in (5.11) or substituting $T_e$ for $T$ in (5.16). Figure 5.3 shows the PSD of the DAC's distortion. Using (5.23), the DAC's SDR is 37.5 dB. To calculate the DAC's SNDR, assume that the DAC's input, $x[n]$, has the form, $s[n] + w[n]$, where $s[n]$ is the unquantized sinusoidal input without dither and $w[n]$ is the signal that includes quantization and dither noise. Then using (5.25), the DAC's SNDR is 33.9 dB. Using the data in Figure 5.3, the DAC's signal to noise ratio (SNR) is calculated to be 36.9 dB. The difference between SNR and SNDR is attributed to the distortion energy at the fundamental frequency as seen in Figure 5.3.
Table 5.1. Nonlinear DAC transfer function.

Nonlinear DAC output values for digital inputs 0-63

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-0.557</td>
<td>1.451</td>
<td>3.466</td>
<td>5.477</td>
<td>7.521</td>
<td>9.533</td>
<td>11.549</td>
<td>13.549</td>
</tr>
<tr>
<td></td>
<td>31.913</td>
<td>33.950</td>
<td>36.012</td>
<td>38.080</td>
<td>40.116</td>
<td>42.164</td>
<td>44.213</td>
<td>46.262</td>
</tr>
<tr>
<td></td>
<td>48.366</td>
<td>50.445</td>
<td>52.520</td>
<td>54.608</td>
<td>56.714</td>
<td>58.789</td>
<td>60.912</td>
<td>63.000</td>
</tr>
</tbody>
</table>

Figure 5.1. Power spectral density plot of the linear part of the Example’s nonlinear DAC output. The DAC’s input is a full scale dithered sinewave that has a frequency of $7\pi/64$ rad/sample.
To calculate this DAC's transfer function using magnitude spectra, the iterative algorithm described in (5.29)-(5.32) requires the PSD of the DAC's output response to a periodic ramp input and the PSD of the DAC's output response to an input that takes on every possible valid DAC input an equal number of times. Figure 5.4 shows the PSD of the DAC's output response to a periodic ramp input, and Figure 5.5 shows the PSD of the DAC's output response to a random signal that takes on every possible valid DAC input an equal number of times. Using the iterative algorithm described in (5.29)-(5.32), 113 iterations were required to obtain the data in Table 5.1.

Figure 5.2. Power spectral density plot of the Example’s nonlinear DAC output. The DAC’s input is a full scale dithered sinewave that has a frequency of $7\pi/64$ rad/sample.
5.6 Conclusions

In this chapter, a Fourier analysis technique was developed that relates a DAC's transfer function to its output frequency spectrum. In particular, (5.11) and (5.16) calculate a DAC's output frequency spectrum given a DAC's transfer function and a particular input signal. Furthermore, a DAC's performance metrics, including SFDR, SDR, and SFDR, can also be determined given a DAC's transfer function and a particular input signal. Also, a DAC's nonzero INL was shown to create harmonic distortion in the DAC output, and the distortion power was calculated exactly. Furthermore, DACs with different INL will

![Power spectral density plot](image)

Figure 5.3. Power spectral density plot of the harmonic distortion part of the Example's nonlinear DAC output. The DAC's input is a full scale dithered sinewave that has a frequency of $7\pi/64$ rad/sample.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Figure 5.4. Power spectral density plot of the Example's nonlinear DAC output. The DAC's input is $x(n) = x_0 + n \mod 2^B$. 

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Figure 5.5. Power spectral density plot of the Example's nonlinear DAC output. The DAC's input is a random signal that takes on every possible valid DAC input an equal number of times. PSD samples are connected by straight lines for convenience in plotting.
produced harmonic distortion in different frequency locations. Finally, an iterative technique given by (5.29)-(5.32) was developed that determines a DAC’s transfer function from measured spectra.
CHAPTER 6

DYNAMIC ELEMENT MATCHING TECHNIQUES FOR DIGITAL TO ANALOG CONVERTERS

Many practical DAC architectures use switches and matched components to convert signals. In practice, perfectly matched components are not possible due to fabrication limitations, thermal gradients, component aging and component noise. The difference between the actual component value and the actual component value is called mismatch error. In DACs, mismatch errors cause conversion errors in the DAC's analog output signal, \( y(t) \), such that \( y(nT) = qx[n] + e[n] \) where \( e[n] \) is a sequence representing DAC conversion errors. This conversion error sequence, \( e[n] \), contains the DAC's harmonic distortion that reduces the DAC's SFDR, SDR, SNDR and ENOB.

Some DAC designs reduce component mismatch errors by using special fabrication processes or by laser trimming components; however, component mismatch errors cannot be completely eliminated. As an alternative to special fabrication processes, techniques, such as digital error correction [42], [63] and digital self-calibration [67], have been used to reduce the effects of component mismatch errors. Another alternative to laser trimming and special fabrication processes, dynamic element matching (DEM) techniques reduce the effects of component mismatch errors by varying the matched components' interconnections.
DEM techniques dynamically rearrange the interconnections of mismatched components so that the time averages of the equivalent components at each of the component positions are nearly equal. If the interconnections are varied such that the mismatched components' virtual positions are sufficiently varied, the harmonic distortion caused by the mismatched components can be reduced, frequency shifted or eliminated. Therefore, DEM techniques can increase an DAC's SFDR, SDR, SNDR and ENOB.

Deterministic DEM techniques and stochastic DEM techniques have been developed for DACs. Deterministic DEM techniques, such as clocked level averaging [72], [73], data weighted averaging [6], and individual level averaging [16], [46], improve DAC performance by locating the distortion caused by the component mismatch errors in certain frequency bands. This distortion is reduced or removed in subsequent processing such as lowpass filtering in ΔΣ architectures. Stochastic DEM techniques, such as stochastic level averaging, randomly permute DAC circuit components each sample [15], [23]. Unlike deterministic DEM techniques which generate harmonic distortion, stochastic DEM techniques can spread the mismatch error energy across the spectrum generating white or colored noise in the DAC output.

Because DEM algorithms dynamically rearrange components, deterministic DEM DACs can be classified as time varying systems, and stochastic DEM DACs can be classified as random systems. Also, because DACs that contain mismatched components are nonlinear systems, deterministic and stochastic DEM DACs can be classified as nonlinear time varying systems or nonlinear random systems, respectively. As a result, analyses of DEM DACs have been specific to a DAC's DEM algorithm even though many DEM DACs have identical architectures. In this section, a particular DEM DAC
architecture is analyzed and criteria are developed for comparing this architecture’s performance when various DEM algorithms are applied to it.

6.1 Analysis of DEM DACs

As shown in Chapter 5, a DAC containing mismatched circuit components can be represented by a nonlinear transformation which can be written

\[ y(nT) = T[x[n]] = T_l[x[n]] + T_e[x[n]] \]  

(6.1)

where \( T \) is the DAC’s nonlinear transformation, \( x[n] \) is the DAC’s digital input sequence, \( T_l \) is the linear transformation that maps \( x[n] \) to the DAC’s linear output and \( T_e \) is the nonlinear transformation that maps \( x[n] \) to the output’s conversion errors. The nonlinear conversion error, \( T_e[x[n]] \), which is caused by component mismatch errors, contains the DAC’s harmonic distortion that reduces the DAC’s SFDR, SDR and SNDR. To reduce a DAC’s harmonic distortion and thus, its SFDR, SDR and SNDR, the effects of mismatch errors in the DAC circuitry must be reduced or removed.

6.1.1 A DEM flash DAC architecture

To demonstrate the principles of DEM, consider the \( B \) bit DEM DAC topology in Figure 6.1, which shows the block diagram of a \( B \) bit flash DAC that performs DEM by mapping a \( B \) bit binary input signal, \( x(n) \), where \( x_0 \leq x(n) \leq x_0 + 2^B - 1 \) to \( 2^B \) single bit DACs [15]. In Figure 6.1, the natural binary converter transforms the input signal, \( x(n) \), where \( x_0 \leq x(n) \leq x_0 + 2^B - 1 \) into the \( B \) bit natural binary signal, \( \chi(n) \), where \( \chi(n) = x(n) - x_0 \) which implies that \( 0 \leq \chi(n) \leq 2^{B} - 1 \). The modified thermometer coder converts the natural binary coded signal, \( \chi(n) \), into a \( 2^B \) bit modified thermometer coded...
The interconnection network connects the $2^B$ bits of the modified thermometer coded signal, $t(n)$, to the $2^B$ unit DACs. Regardless of the interconnection network's control signal, $c(n)$, the interconnection network's output, $g(n)$, activates $\chi(n)$, or $x(n)+x_0$, unit DACs and deactivates the remaining $2^B - \chi(n)$, or $2^B - x(n)+x_0$, unit DACs. If each activated unit DAC generates an analog signal, $a$, and each deactivated unit DAC generates an analog signal, $d$, the DAC's quantization step sizes or code widths, $q$, are the difference between $a$ and $d$, that is $q = a - d$ which is a constant. Also, the DAC's output, $y(nT)$, which is the sum of all of the unit DAC outputs, can be written as

$$y(nT) = a\chi(n) + d[2^B - \chi(n)] = (a-d)\chi(n) + d2^B = q[2^B - \chi(n)] + d2^B \quad (6.2)$$

In practice, mismatched components between each of the unit DACs prevent the analog outputs of the activated unit DACs from being identical. Similarly, the analog outputs of the deactivated unit DACs are not identical. As a result, the DAC's quantization step sizes or code widths are not constant, the DAC's transfer function is nonlinear, and the DAC's performance is degraded as shown in Chapter 5. To improve the DAC's performance, the DEM DAC's interconnection network dynamically alters the mapping.

Figure 6.1. A $B$ bit dynamic element matching DAC architecture.
between the input signal, \( x(n) \), and the mismatched unit DACs so that the time averages of the activated unit DAC outputs are nearly equal and the time averages of the deactivated unit DAC outputs are nearly equal. If the interconnection network's control signal, \( c(n) \), is deterministic, the mapping between the DAC's input signal, \( x(n) \), and the \( 2^B \) unit DACs is deterministic, and the DAC is referred to as a deterministic DEM DAC. Similarly, if the interconnection network's control signal, \( c(n) \), is stochastic, the mapping between the DAC's input signal, \( x(n) \), and the \( 2^B \) unit DACs is stochastic, and the DAC is referred to as a stochastic DEM DAC.

Although the DEM DAC in Figure 6.1 has been analyzed for specific interconnection networks and specific control signals, a general analysis of this DEM DAC and criteria for comparing the performance for different interconnection networks and control signals does not appear in the literature. In this chapter, a general analysis of the DEM DAC in is developed. Using this analysis, the DAC's mean INL, the variance of the DAC's INL, the DAC's SDR, the DAC's SNDR, and the DAC's SFDR can be calculated and used as criteria to compare DEM DACs using various interconnection networks and control signals.

### 6.1.2 An analysis of the DEM flash DAC architecture

In Figure 6.1, the natural binary coder and the modified thermometer coder transform the DAC's digital input, \( x(n) \), where \( x_0 \leq x(n) \leq x_0 + 2^B - 1 \) into the \( 2^B \) bit thermometer coded signal, \( t(n) \), where \( t(n) = x(n) - x_0 \) which implies that \( 0 \leq t(n) \leq 2^B - 1 \). The interconnection network maps the \( 2^B \) bits of the signal, \( t(n) \), to the \( 2^B \) unit DACs, and the DAC's output, \( y(nT) \), is the sum of the outputs of the unit DACs. Therefore, the DEM

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
DAC's output, $y(nT)$, is a function of $t(n)$ which is a function of $x(n)$, the interconnection network, the interconnection network’s control signal, $c(n)$, and the activated and deactivated analog outputs generated by each of the unit DACs.

To express the thermometer coded signal, $t(n)$, as a function of the DAC’s input signal, $x(n)$, let $\chi_k(n)$ represent $\chi(n)$’s $k$th bit where $\chi_1(n)$ and $\chi_B(n)$ are $\chi(n)$’s least significant bit (LSB) and most significant bit (MSB), respectively. Also, let the vector, $T(n)$, represent the modified thermometer coded signal, $t(n)$, where

$$T(n) = \begin{bmatrix} t_1(n) & t_2(n) & t_3(n) & \ldots & t_{2^B}(n) \end{bmatrix}^T,$$

the superscript $^T$ denotes transpose, and $t_1(n)$ and $t_{2^B}(n)$ are $t(n)$’s LSB and MSB, respectively. In this paper, the modified thermometer code is defined such that $t_1(n) = 0$ and $t_{2^B+1}(n) = \ldots = t_{2^B}(n) = \chi_k(n)$. For example, if $B = 3$,

$$T(n) = \begin{bmatrix} 0 & \chi_1(n) & \chi_2(n) & \chi_3(n) & \chi_3(n) & \chi_3(n) \end{bmatrix}^T.$$

Although an ordinary thermometer coder or another type of modified thermometer coder can be used, the modified thermometer coder defined above is typically used in DEM DACs because this modified thermometer coder typically simplifies the hardware design. Regardless of the thermometer code, $t(n)$ and $x(n)$ can be related by

$$T^T(n)T(n) = \chi(n) = x(n) - x_0$$

where the vector, $T(n)$, represents $t(n)$.

To express the interconnection network’s $2^B$ bit output signal, $g(n)$, as a function of the input signal, $x(n)$, let the vector, $G(n)$, where

$$G(n) = \begin{bmatrix} g_1(n) & g_2(n) & g_3(n) & \ldots & g_{2^B}(n) \end{bmatrix}^T$$

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
and \( g_1(n) \) and \( g_2(n) \) are \( g(n) \)'s LSB and MSB, respectively, represent the interconnection network's \( 2^B \) bit output signal, \( g(n) \). Because the interconnection network's output, \( G(n) \), is a function of the interconnection network's control signal, \( c(n) \), and the thermometer coded signal, \( t(n) \), which is a function of the DAC's input, \( x(n) \), the interconnection network can be represented by the transformation \( T_G \) such that

\[
G(n) = T_G[x(n), c(n)].
\]

Regardless of the transformation \( T_G \) and the control signal, \( c(n) \),

\[
G^T(n)G(n) = T^T(n)T(n) = x(n) - x_0,
\]

and

\[
[1 - G(n)]^T[1 - G(n)] = [1 - T(n)]^T[1 - T(n)] = 2^B - x(n) = 2^B \cdot x(n) + x_0.
\]

Therefore, the interconnection network activates \( G^T(n)G(n) \), or \( x(n) - x_0 \), unit DACs and deactivates the remaining \( [1 - G(n)]^T[1 - G(n)] \), or \( 2^B \cdot x(n) + x_0 \), unit DACs.

To express the DAC's analog output, \( y(nT) \), as a function of the interconnection network's output, \( G(n) \), define the output, \( y_k(nT) \), of the \( k \)th unit DAC as

\[
y_k(nT) = \begin{cases} a_k & \text{if } g_k[n] = 1 \\ d_k & \text{if } g_k[n] = 0 \end{cases}
\]

where \( a_k \) and \( d_k \) are the values of the activated and deactivated \( k \)th unit DAC, respectively. If \( \bar{a} \) and \( \bar{d} \) are defined as the average values of the activated and deactivated unit DACs, respectively, that is

\[
\bar{a} = \frac{1}{2^B} \sum_{k=1}^{2^B} a_k
\]

and
\[ \bar{d} = \frac{1}{2^B} \sum_{k=1}^{2^B} d_k \tag{6.6} \]

then \( \bar{q} = (\bar{a} - \bar{d}) \) where \( \bar{q} \) is the DAC's average code width, and the output, \( y_k(nT) \), of the \( k \)th unit DAC can also be written as

\[
y_k(nT) = \begin{cases} \bar{a} + h_k & g_k[n] = 1 \\ \bar{d} + l_k & g_k[n] = 0 \end{cases}
\]

where \( h_k = a_k - \bar{a} \) and \( l_k = d_k - \bar{d} \). Thus, if \( H = [h_1(n) \ h_2(n) \ h_3(n) \ \ldots \ h_{2^B}(n)]^T \), \( L = [l_1(n) \ l_2(n) \ l_3(n) \ \ldots \ l_{2^B}(n)]^T \) and \( \mathbf{1} \) is a \( 2^B \times 1 \) vector of ones, the DAC's output, \( y(nT) \), can be written as

\[
y(nT) = \sum_{k=1}^{2^B} y_k(nT) = \bar{a}G^T(n)G(n) + G^T(n)H + \bar{d}[1 - G(n)]^T[1 - G(n)] + [1 - G(n)]^T L
\]

Substituting (6.4) and (6.5) into (6.7),

\[
y(nT) = \bar{a}[x(n) - x_0] + G^T(n)H + \bar{d}(2^B - x(n) + x_0) + [1 - G(n)]^T L
\]

\[
= (\bar{a} - \bar{d})[x(n) - x_0] + \bar{d}2^B + G^T(n)(H - L) + \mathbf{1}^T L
\]

Because \( d_k = \bar{d} + l_k \), (6.6) can be written as

\[
\bar{d} = \frac{1}{2^B} \sum_{k=1}^{2^B} \bar{d} + l_k
\]

which implies that

\[
\sum_{k=1}^{2^B} l_k = \mathbf{1}^T L = 0 \tag{6.9}
\]

Similarly, because \( a_k = \bar{a} + h_k \),
\[ \sum_{k=1}^{2^B} h_k = 1^T H = 0 \quad (6.10) \]

Substituting (6.3) and (6.9) into (6.8), the DAC’s output can be written as

\[ y(nT) = \bar{q}[x(n)-x_0] + \bar{d}2^B + T_G^T[x(n), c(n)](H-L). \quad (6.11) \]

In (6.11), the term, \( \bar{q}[x(n)-x_0] + \bar{d}2^B \), is similar to the ideal DAC’s output in (6.2), and represents the DAC’s output when all of the unit DAC’s activated and deactivated analog outputs are identically \( \bar{a} \) and \( \bar{d} \), respectively. Therefore, the last term, \( T_G^T[x(n), c(n)](H-L) \), in (6.11), is the DAC’s nonlinear transformation that describes the DAC’s INL.

To properly calculate performance criteria for the DEM DAC in Figure 6.1, the DC power in the DAC’s digital input, \( x(n) \), and the DC power of DAC’s undistorted output, \( \bar{q}[x(n)-x_0] + \bar{d}2^B \), must be equal. Therefore, this analysis assumes that \( x_0 = \bar{d}2^B/\bar{q} \) which implies that

\[ y(nT) = T[x(n), c(n)] - \bar{q}x(n) + T_{\text{INL}}[x(n), c(n)] \quad (6.12) \]

where \( T \) represents the DAC’s transformation and \( T_{\text{INL}} \) represents the transformation of the DAC’s INL, that is,

\[ T_{\text{INL}}[x(n), c(n)] = T_G^T[x(n), c(n)](H-L) = \text{INL}_{\chi(n)} \]

where \( \text{INL}_{\chi(n)} \) is the INL for the DAC’s \( \chi(n) \)th code, that is, the DAC’s INL when \( x(n) = \chi(n) + x_0 \). Figure 6.2 and Figure 6.3 show four example DAC transformations and INL transformations, respectively.
6.1.3 Performance criteria for DEM DACs

If the interconnection network's control signal, $c(n)$, is a stochastic signal, the transformation $T_G$ is a stochastic transformation which implies that $T_{INL}$, and thus $T$, are stochastic transformations. Similarly, if $c(n)$ is a time varying, deterministic signal, the transformation $T_G$ is a time varying, deterministic transformation which implies that $T_{INL}$, and thus $T$, are time varying, deterministic transformations. As a result, stochastic

![Four example six bit DAC transformations.](image)

Figure 6.2. Four example six bit DAC transformations.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
DEM DAC performance criteria are calculated using probabilistic means and variances, and deterministic DEM DAC performance criteria are calculated using arithmetic means and variances. In this section, the mean of the DAC's INL, the variance of the DAC's INL, the DAC's SDR, and the DAC's SNDR are determined for a stochastic DEM DAC. These performance criteria can also be applied to deterministic DEM DACs by replacing the probabilistic means and variances with arithmetic means and variances, respectively.

Figure 6.3. Four example six bit DAC integral nonlinearity transformations.
To calculate the mean of a stochastic DEM DAC's INL, apply the expectation operator conditioned on the input signal, \( x(n) \), to the DAC's output in (6.12) that is,

\[
E\left[ y(nT)|x(n) \right] = \bar{q} x(n) + E\left\{ T_G^T[x(n), c(n)] | x(n) \right\} (H - L). \tag{6.13}
\]

In (6.13), the term, \( \bar{q} x(n) \), represents the DAC's output when all of the DAC's code widths are identically \( \bar{q} \), and therefore, the second term, \( E\left\{ T_G^T[x(n), c(n)] | x(n) \right\} (H - L) \), in (6.13) is the stochastic DEM DAC's expected INL for a particular input, \( x(n) \), that is,

\[
E\left\{ T_G^T[x(n), c(n)] | x(n) \right\} (H - L) = E\{ \text{INL}_x \} | x(n) \} \tag{6.14}
\]

Using (6.14), a stochastic DEM DAC's INL variance, \( \sigma_{\text{INL}|x}^2 \), conditioned on \( x(n) \) can be calculated as

\[
\sigma_{\text{INL}|x}^2 = E\{ T_G^T[x(n)] \} - E\{ T_G^T[x(n)] \} \frac{(H - L)^T E\{ T_G^T[x(n)] \} (H - L)}{H - L} \tag{6.15}
\]

which is also the DAC's output variance, \( \sigma_{y|x}^2 \), conditioned on \( x(n) \), that is,

\[
\sigma_{y|x}^2 = \sigma_{\text{INL}|x}^2 = \sigma_{\text{INL}_x|x}^2. \tag{6.16}
\]

Two other criteria used to measure a DAC's performance are SDR and SNDR. To calculate a stochastic DEM DAC's SDR, assume that the DAC's analog output, \( y(t) \), is an impulse train weighted by \( y(nT) \), that is,

\[
y(t) = \sum_{n=-\infty}^{\infty} y(nT) \delta(t-nT) \]
where $\delta(r)$ is the Dirac delta function. As a result, the DAC's output spectrum, $Y(j\Omega)$, can be written as

$$Y(j\Omega) = \sum_{n=-\infty}^{\infty} y(nT)e^{-j\omega nT} = Y(e^{j\omega T}) \quad |\omega| \leq \pi$$

where $\Omega$ represents frequency in radians per second and $\omega$ represents frequency in radians per sample. Therefore, the DAC's average signal plus distortion power, $P_y$, is

$$P_y = \frac{1}{2\pi} \int_{-\pi}^{\pi} |Y(e^{j\omega T})|^2 d\omega T$$

Because $y(nT)$ is a real stochastic stationary signal,

$$P_y = E[y^2(nT)] = E\left[ E[y^2(nT)|x(n)] \right]. \quad (6.17)$$

Substituting (6.12) into (6.17),

$$P_y = \frac{1}{2} E\left[ E\left[ q^2 x^2(n) + 2q x(n) T_{\text{INL}}[x(n), c(n)] x(n) + T_{\text{INL}}^2[x(n), c(n)] x(n) \right] \right]$$

$$= \frac{1}{2} E\left[ q^2 x^2(n) + 2q x(n) E\left[ T_{\text{INL}}[x(n), c(n)] x(n) \right] + \sigma_{T_{\text{INL}}}^2 + E^2\left[ T_{\text{INL}}[x(n)] \right] \right] \quad (6.18)$$

Rearranging the terms in (6.18) gives

$$P_y = \frac{1}{2} E\left[ x^2(n) \right] + 2q E\left[ x(n) E\left[ T_{\text{INL}}[x(n)] \right] \right] + \sigma_{T_{\text{INL}}}^2 + E^2\left[ T_{\text{INL}}[x(n)] \right] \quad (6.19)$$

In (6.19), the first term, $\frac{1}{2} E\left[ x^2(n) \right]$, is the average power, $P_y$, of the output of a linear DAC with code widths $q$, and the remaining terms represent the average power, $P_{y_d}$, of the output's conversion errors or distortion. Therefore, the DAC's SDR is
To calculate a stochastic DEM DAC's SNDR, consider an input, \( x(n) \), which can be written as

\[
x(n) = s(n) + w(n)
\]  

(6.21)

where \( s(n) \) is the signal component of the DAC's input and \( w(n) \) is an independent zero mean white noise component of the DAC's input. Substituting (6.21) into (6.19), the DAC's average signal plus noise plus distortion power, \( P_y \), is

\[
P_y = P_{y_s} + P_{y_{s+d}}
\]  

(6.22)

where

\[
P_{y_s} = q^2 E\left[ s^2(n) \right]
\]

and

\[
P_{y_{s+d}} = q^2 E\left[ w^2(n) \right] + q^2 E\left[ s(n)w(n) \right] + 2q E\left\{ x(n)E\left[ T_{INL}|x(n) \right] \right\} + \sigma_{T_{INL}}^2 + E^2\left[ T_{INL}|x(n) \right]
\]

In (6.22), the first term, \( P_{y_s} \), is the average power of the output of a linear DAC with code widths \( q \) and the second term, \( P_{y_{s+d}} \), is the average power of the output's noise plus distortion. Since \( w(n) \) is an independent, zero mean random signal, the DAC's average noise plus distortion power is

\[
P_{y_{s+d}} = q^2 E\left[ w^2(n) \right] + 2q E\left\{ x(n)E\left[ T_{INL}|x(n) \right] \right\} + \sigma_{T_{INL}}^2 + E^2\left[ T_{INL}|x(n) \right].
\]

Therefore the DAC's SNDR is
SNDR = \frac{P_{y_1}}{P_{y_{\text{ref}}}}

\begin{align*}
\bar{q}^2 E\left[ s^2(n) \right] \\
\bar{q}^2 E\left[ w^2(n) \right] + 2\bar{q} E\left[ x(n) E\left[ T_{\text{INL}}|x(n)\right] \right] + \sigma^2_{T_{\text{INL}}} + E^2\left[ T_{\text{INL}}|x(n)\right]
\end{align*}

(6.23)

To calculate the mean of the INL, the variance of the INL, the SDR, and the SNDR for a deterministic DAC, replace the probabilistic means and variances in (6.14), (6.15), (6.20) and (6.23) with arithmetic averages and variances, respectively.

### 6.1.4 An analysis of stochastic DEM flash DACs

For stochastic DEM DACs, interconnection networks are often chosen and controlled such that the DAC’s mean transformation represents a linear DAC, that is

\[ E\left[ y(nT)|x(n)\right] = \bar{q} x(n). \tag{6.24} \]

which implies that the stochastic DEM DAC’s expected INL for a particular input, \( x(n) \), is zero, that is,

\[ E\left[ T_{\text{INL}}[x(n, c(n)]|x(n)\right] = E\left[ T_G^T[x(n, c(n)]|x(n)\right](H - L) = 0. \tag{6.25} \]

For these types of stochastic DEM DACs, the performance criteria described by (6.15), (6.20) and (6.23), can be simplified by substituting (6.25) into these equations. Substituting (6.25) into (6.15),

\[ \sigma^2_{T_{\text{INL}}} = \sigma^2_{y|x} = (H - L)^T E\left[ T_G^T T_G\right] x(n)\right](H - L). \tag{6.26} \]
which is also the DAC’s mean squared INL for a particular \( x(n) \). Substituting (6.25) into (6.20), the DAC’s SDR can be written as

\[
\text{SDR} = \frac{P_{y_i}}{P_{y_d}} = \frac{q^2 E\left[x^2(n)\right]}{\sigma_{\text{INL}}^2}, \quad (6.27)
\]

and substituting (6.25) into (6.23), the DAC’s SNDR can be written as

\[
\text{SNDR} = \frac{P_{y_i}}{P_{y_{\text{ref}}}} = \frac{q^2 E\left[s^2(n)\right]}{q^2 E\left[w^2(n)\right] + \sigma_{\text{INL}}^2}. \quad (6.28)
\]

Another criterion used to measure a DAC’s performance is SFDR. Nonlinear deterministic DACs generate periodic distortion in the DAC’s output, and this periodic distortion appears as spurs in the frequency spectrum of the DAC’s output. Unlike nonlinear deterministic DACs, nonlinear stochastic DEM DACs do not generate periodic distortion in the DAC’s output, but instead generate stochastic distortion in the DAC’s output. This stochastic distortion appears as noise in the DAC’s output frequency spectrum. As a result, a stochastic DEM DAC’s SFDR can be estimated by estimating the noise floor of the power spectral density (PSD) of the DAC’s output. Thus, if the DAC’s input noise, \( w(n) \), is an independent zero mean white stochastic signal and the DAC’s stochastic DEM whitens the DAC’s output distortion, the DAC’s SFDR can be estimated using the DAC’s SNDR.

To determine a transformation, \( T_G[x(n), c(n)] \), that can whiten the DAC’s output distortion, consider the stochastic DEM DAC’s output autocorrelation, \( \phi_{yy}(n, k) \), where

\[
\phi_{yy}(n, k) = E\{y[n]y[n+k]\}. \quad (6.29)
\]
Substituting (6.12) into (6.29), the DAC’s output autocorrelation can be written as

\[
\phi_{yy}(n, k) = E \left[ \left( \bar{q}s(n) + \bar{q}w(n) + T_{\text{INL}}[x(n), c(n)] \right) \right.
\]
\[
\times \left( \bar{q}s(n + k) + \bar{q}w(n + k) + T_{\text{INL}}[x(n + k), c(n + k)] \right) \right]
\]

where \( x(n) = s(n) + w(n) \). Assuming that \( w(n) \) is an independent zero mean white stochastic signal, (6.30) can be written as

\[
\phi_{yy}(n, k) = \bar{q}^2 \phi_{ss}(n, k) + \bar{q}^2 \sigma_w^2 \delta(k)
\]
\[
+ \bar{q} E \{ x(n + k) E \{ T_{\text{INL}}[x(n), c(n)] | x(n) \} \}
\]
\[
+ \bar{q} E \{ x(n) E \{ T_{\text{INL}}[x(n + k), c(n + k)] | x(n + k) \} \}
\]
\[
+ E \left\{ T_{\text{INL}}[x(n), c(n)] T_{\text{INL}}[x(n + k), c(n + k)] \right\} \quad (6.31)
\]

where \( \delta(k) \) is the Kronecker delta function. If the interconnection network is chosen and controlled such that (6.25) is satisfied, (6.31) can be written as

\[
\phi_{yy}(n, k) = \bar{q}^2 \phi_{ss}(n, k) + \bar{q}^2 \sigma_w^2 \delta(k) + E \left\{ T_{\text{INL}}[x(n), c(n)] T_{\text{INL}}[x(n + k), c(n + k)] \right\} \quad (6.32)
\]

If the interconnection network also has no memory and the control signal, \( c(n) \), is an independent zero mean white stochastic signal, (6.32) can be written as

\[
\phi_{yy}(n, k) = \bar{q}^2 \phi_{ss}(n, k) + \bar{q}^2 \sigma_w^2 \delta(k) + \sigma_{\text{INL}}^2 \delta(k) \quad (6.33)
\]

The result in (6.33) shows that the expected output of a stochastic DAC is that of a linear DAC plus an independent zero mean white noise that has power \( \bar{q}^2 \sigma_w^2 + \sigma_{\text{INL}}^2 \). Therefore, if the DAC’s input noise, \( w(n) \), and the control signal, \( c(n) \), are independent zero mean white stochastic signals, the interconnection network has no memory, and the
interconnection network is chosen such that (6.25) is satisfied, the SFDR of a stochastic DEM DAC can be estimated using the SNDR in (6.28).

6.2 A comparison of five DEM flash DACs

Consider a six bit linear DAC that has a full scale dithered sinusoidal input with a frequency of \(313\pi/2048\) radians/sample. The dither sequence is a strictly white sequence with a triangular probability distribution function supported on \((-q, q)\). Figure 6.4 shows the power spectral density (PSD) of the simulated linear DAC's output. The PSD was obtained by averaging 40 periodograms each corresponding to \(2^{12}\) samples of the dithered sinusoid input sequence.

![Power spectral density of the output of a six bit linear DAC.](image)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
6.2.1 Example One

For this example, the unit DACs in Figure 6.1 have linear gradient errors that vary linearly from +5% to -5% of the LSB and uniformly distributed random errors that are uniformly distributed between +2% and -2% of the LSB. Figure 6.5 shows the PSD of the nonlinear DAC's output. This PSD was also obtained by averaging 40 periodograms each corresponding to $2^{12}$ samples.

In this example, five DEM algorithms are applied to this nonlinear DAC. The first algorithm is a Benes network controlled by a white uniformly distributed stochastic signal [23]. The Benes network is capable of generating all $2^n - 1$ unique vectors for $G(n)$; however, the Benes network contains a large number of switches and control signals [45].

![Figure 6.5. Power spectral density of the output of a six bit DAC with linear gradient and uniformly distributed unit DAC errors.](image)
The second and third DEM algorithms use a more hardware efficient interconnection network, called a barrel shift network, which is capable of generating $2^B - 2^B + 1$ of the $2^B - 1$ unique vectors for $G(n)$. One of the barrel shift networks is controlled by a deterministic signal and the other is controlled by a white uniformly distributed stochastic signal. The fourth and fifth DEM algorithms use a hardware efficient interconnection network, called a generalized cube network, which is capable of generating $\frac{1}{3}(2^B - 2)^2 + 1$ of the $2^B - 1$ unique vectors for $G(n)$. The generalized cube network has several equivalent circuit implementations of varying complexity as will be shown later. One of the generalized cube networks is controlled by a white uniformly distributed stochastic signal and the other is controlled by a colored uniformly distributed stochastic signal.

For the first DEM algorithm which is a Benes network controlled by a white uniformly distributed random signal, it can be shown that

$$E\{T_G|x(n)\} = \frac{\chi(n)}{2^B} I$$

(6.34)

and

$$E\{T_G^T|x(n)\} = \frac{\chi(n)[\chi(n)-1]}{2^B(2^B-1)} I_{2^B} + \frac{\chi(n)[2^B-\chi(n)]}{2^B(2^B-1)} I_{2^B}.$$

(6.35)

where $I_{2^B}$ is the $2^B \times 2^B$ ones matrix and $I_{2^B}$ is the $2^B \times 2^B$ identity matrix [23].

Substituting (6.9), (6.10), and (6.34) into (6.14),

$$E\{T_G|x(n)\}(H-L) = \frac{\chi(n)}{2^B} I^T (H-L) = 0,$$

(6.36)

and thus the DACs' average transformation can be described by (6.25). Substituting (6.35) into (6.26), the variance of this DEM DAC's INL is 0.122. Substituting (6.35) into (6.27),
this DEM DAC's SDR is 41.9 dB. Assuming the DAC's input, \( x(n) \), has the form, \( s(n) + w(n) \), where \( s(n) \) is the unquantized sinusoidal input without dither and \( w(n) \) is the signal that includes quantization and dither noise, the stochastic Benes DEM DAC's SNDR is calculated to be 31.7 dB using (6.28). This DEM DAC was simulated using \( 40 \times 2^{12} \) samples and its experimental INL variance, SDR and SNDR were 0.121, 41.9 dB and 31.7 dB, respectively. Figure 6.6 shows the experimental PSD of the output of the nonlinear DAC using this stochastic Benes DEM algorithm.

The second DEM algorithm is a \( B \) bit barrel shift network controlled by the output of a \( B \) bit binary counter. This deterministic DEM algorithm is the clocked level averaging (CLA) algorithm described in [6] and [46]. The third DEM algorithm applied to the

![Figure 6.6. Power spectral density of the output of a six bit DEM DAC using a stochastic Benes network controlled by a white uniformly distributed stochastic control signal.](image)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
nonlinear DAC is a $B$ bit barrel shift network controlled by a white uniformly distributed stochastic signal. For both algorithms, it can be shown that the expected transformation of $T_G$ conditioned on $x(n)$ is given by (6.34), and thus the DACs' average transformation can be described by (6.25). Furthermore, it can be shown that the columns of $E\{T_G T_G^T \mid x(n)\}$ are $2^B$ point circular convolutions of the modified thermometer code, $t(n)$, that is, $v_{r,c}$, the element in the $r$th row and $c$th column of $E\{T_G T_G^T \mid x(n)\}$ is

$$v_{r,c} = \frac{1}{2^B} \sum_{l=0}^{2^B-1} t_{(r-l) \mod 2^B} t_{(c-l) \mod 2^B},$$

(6.37)

where $t_i$ is the $i$th bit of the thermometer code, $t(n)$, corresponding to input $x(n)$. Using (6.15), the variances of the CLA DEM DAC's transformation and the stochastic barrel

![Figure 6.7. Power spectral density of the output of a six bit DEM DAC using a deterministic clocked level averaging algorithm.](image)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
shift network DEM DAC's transformation are 1.182 and 1.182, respectively. Using (6.20) and (6.23), the SDR and SNDR of the CLA DEM DAC are 32.0 dB and 29.0 dB, respectively. Because the stochastic barrel shift DEM DAC has the same INL mean and INL variance as the CLA DEM DAC, the SDR and the SNDR of the stochastic barrel shift network DEM DAC are also 32.0 dB and 29.0 dB, respectively. The CLA DEM DAC was simulated using $40 \times 2^{12}$ samples and its experimental INL variance, SDR and SNDR were 1.182, 32.0 dB and 29.0 dB, respectively. The stochastic barrel shift network DEM DAC was also simulated using the same $40 \times 2^{12}$ samples and the experimental INL variance, SDR and SNDR were 1.183, 32.0 dB and 29.0 dB, respectively. The PSD of the

![Figure 6.8](image)

**Figure 6.8.** Power spectral density of the output of a six bit DEM DAC using a stochastic barrel shift network controlled by a white uniformly distributed stochastic control signal.
output of the nonlinear DAC using the CLA DEM algorithm and the stochastic barrel shift network DEM algorithm are shown in Figure 6.7 and Figure 6.8, respectively.

The fourth DEM algorithm is a $B$ bit generalized cube network controlled by a white uniformly distributed stochastic signal, and the fifth DEM algorithm is a $B$ bit generalized cube network controlled by an colored uniformly distributed stochastic signal. For both algorithms, it can be shown that the expected transformation of $T_G$ conditioned on $x(n)$ is given by (6.34), and thus the DACs’ average transformation can be described by (6.25). Using (6.26), the variances of the generalized cube network DEM DAC’s transformation controlled by white and colored stochastic signals are 2.348 and 2.348, respectively. Using

![Power spectral density](image)

**Figure 6.9.** Power spectral density of the output of a six bit DEM DAC using a stochastic generalized cube network controlled by a white uniformly distributed stochastic control signal.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
(6.27) and (6.28), the SDR and SNDR of the generalized cube network DEM DAC with a white control signal are 29.0 dB and 27.3 dB, respectively. Because the generalized cube network DEM DAC controlled by a white stochastic signal has the same INL mean and INL variance as the generalized cube network DEM DAC controlled by a colored uniformly distributed stochastic signal, the SDR and the SNDR of the generalized cube network DEM DAC controlled by a colored uniformly distributed control signal are also 29.0 dB and 27.3 dB, respectively. The generalized cube network DEM DAC controlled by a white uniformly distributed stochastic signal was simulated using $40 \times 2^{12}$ samples and its experimental INL variance, SDR and SNDR were 2.350, 29.0 dB and 27.3 dB.

![Figure 6.10](image.png)

**Figure 6.10.** Power spectral density of the output of a six bit DEM DAC using a stochastic generalized cube network controlled by a colored uniformly distributed stochastic control signal.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
respectively. The generalized cube network DEM DAC controlled by a colored uniformly distributed stochastic signal was also simulated using the same $40 \times 2^{12}$ samples and its experimental INL variance, SDR and SNDR were 2.349, 29.0 dB and 27.3 dB, respectively. The PSD of the output of the nonlinear DAC using the generalized cube network with white and colored control signals are shown in Figure 6.9 and Figure 6.10, respectively. The PSD of the output distortion of the nonlinear DAC using the generalized cube network with a colored control signal is shown in Figure 6.11.

This example's results are summarized in Table 6.1 and Table 6.2.

![Figure 6.11](image_url)  
*Figure 6.11. Power spectral density of a six bit DEM DAC's distortion output. The DEM DAC uses a stochastic generalized cube network controlled by a colored uniformly distributed stochastic control signal.*
Table 6.1. Comparison of Example One’s SDR

<table>
<thead>
<tr>
<th>DEM technique</th>
<th>Derived SDR</th>
<th>Simulated SDR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stochastic Benes network</td>
<td>41.9 dB</td>
<td>41.9 dB</td>
</tr>
<tr>
<td>Clocked level averaging</td>
<td>32.0 dB</td>
<td>32.0 dB</td>
</tr>
<tr>
<td>Stochastic barrel shifting</td>
<td>32.0 dB</td>
<td>32.0 dB</td>
</tr>
<tr>
<td>Stochastic GCN (white)</td>
<td>29.0 dB</td>
<td>29.0 dB</td>
</tr>
<tr>
<td>Stochastic GCN (colored)</td>
<td>29.0 dB</td>
<td>29.0 dB</td>
</tr>
</tbody>
</table>

6.2.2 Example Two

For this example, the unit DACs in Figure 6.1 have random errors that are normally distributed between +7% and -7% of the LSB. Figure 6.5 shows the PSD of the nonlinear

![Figure 6.12. Power spectral density of the output of a six bit DAC with normally distributed unit DAC errors.](image)
Table 6.2. Comparison of Example One's SNDR

<table>
<thead>
<tr>
<th>DEM technique</th>
<th>Derived SNDR</th>
<th>Simulated SNDR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stochastic Benes network</td>
<td>31.7 dB</td>
<td>31.7 dB</td>
</tr>
<tr>
<td>Clocked level averaging</td>
<td>29.0 dB</td>
<td>29.0 dB</td>
</tr>
<tr>
<td>Stochastic barrel shifting</td>
<td>29.0 dB</td>
<td>29.0 dB</td>
</tr>
<tr>
<td>Stochastic GCN (white)</td>
<td>27.3 dB</td>
<td>27.3 dB</td>
</tr>
<tr>
<td>Stochastic GCN (colored)</td>
<td>27.3 dB</td>
<td>27.3 dB</td>
</tr>
</tbody>
</table>

DAC's output. This PSD was also obtained by averaging 40 periodograms each corresponding to $2^{12}$ samples.

In this example, the five DEM algorithms in Section 6.2.1 are applied to this nonlinear DAC. For the first DEM algorithm which is a Benes network controlled by a white

![Power spectral density](image)

Figure 6.13. Power spectral density of the output of a six bit DEM DAC using a stochastic Benes network controlled by a white uniformly distributed stochastic control signal.
uniformly distributed random signal, and the DACs’ average transformation can be described by (6.25). Substituting (6.35) into (6.26), the variance of this DEM DAC’s INL is 0.327. Substituting (6.35) into (6.27), this DEM DAC’s SDR is 37.6 dB. Assuming the DAC’s input, \(x(n)\), has the form, \(s(n) + w(n)\), where \(s(n)\) is the unquantized sinusoidal input without dither and \(w(n)\) is the signal that includes quantization and dither noise, the stochastic Benes DEM DAC’s SNDR is calculated to be 31.9 dB using (6.28). This DEM DAC was simulated using \(40 \times 2^{12}\) samples and its experimental INL variance, SDR and SNDR were 0.327, 37.5 dB and 31.9 dB, respectively. Figure 6.6 shows the experimental PSD of the output of the nonlinear DAC using this stochastic Benes DEM algorithm.

The second DEM algorithm is a \(B\) bit barrel shift network controlled by the output of a \(B\) bit binary counter. This deterministic DEM algorithm is the clocked level averaging (CLA) algorithm described in [6] and [46]. The third DEM algorithm applied to the nonlinear DAC is a \(B\) bit barrel shift network controlled by a white uniformly distributed stochastic signal. For both algorithms, the expected transformation of \(T_G\) conditioned on \(x(n)\) is given by (6.34), and thus the DACs’ average transformation can be described by (6.25). Furthermore, the element in the \(r\)th row and \(c\)th column of \(E\{T_GT_G^T|x(n)\}\) is given by (6.37). Using (6.15), the variances of the CLA DEM DAC’s transformation and the stochastic barrel shift network DEM DAC’s transformation are 0.241 and 0.241, respectively. Using (6.20) and (6.23), the SDR and SNDR of the CLA DEM DAC are 38.9 dB and 32.2 dB, respectively. Because the stochastic barrel shift DEM DAC has the same INL mean and INL variance as the CLA DEM DAC, the SDR and the SNDR of the stochastic barrel shift network DEM DAC are also 38.9 dB and 32.2 dB, respectively. The CLA DEM DAC was simulated using \(40 \times 2^{12}\) samples and its experimental INL
variance, SDR and SNDR were 0.241, 38.9 dB and 32.2 dB, respectively. The stochastic barrel shift network DEM DAC was also simulated using the same $40 \times 2^{12}$ samples and the experimental INL variance, SDR and SNDR were 0.240, 38.9 dB and 32.2 dB, respectively. The PSD of the output of the nonlinear DAC using the CLA DEM algorithm and the stochastic barrel shift network DEM algorithm are shown in Figure 6.7 and Figure 6.8, respectively.

The fourth DEM algorithm is a $B$ bit generalized cube network controlled by a white uniformly distributed stochastic signal, and the fifth DEM algorithms is a $B$ bit generalized cube network controlled by an colored uniformly distributed stochastic signal.

Figure 6.14. Power spectral density of the output of a six bit DEM DAC using a deterministic clocked level averaging algorithm.
For both algorithms, it can be shown that the expected transformation of $T_G$ conditioned on $x(n)$ is given by (6.34), and thus the DACs’ average transformation can be described by (6.25). Using (6.26), the variances of the generalized cube network DEM DAC’s transformation controlled by white and colored stochastic signals are 0.184 and 0.184, respectively. Using (6.27) and (6.28), the SDR and SNDR of the generalized cube network DEM DAC with a white control signal are 40.0 dB and 32.5 dB, respectively. Because the generalized cube network DEM DAC controlled by a white stochastic signal has the same INL mean and INL variance as the generalized cube network DEM DAC controlled by a colored uniformly distributed stochastic signal, the SDR and the SNDR of the generalized

![Figure 6.15](image.png)

**Figure 6.15.** Power spectral density of the output of a six bit DEM DAC using a stochastic barrel shift network controlled by a white uniformly distributed stochastic control signal.
cube network DEM DAC controlled by a colored uniformly distributed control signal are also 40.0 dB and 32.5 dB, respectively. The generalized cube network DEM DAC controlled by a white uniformly distributed stochastic signal was simulated using $40 \times 2^{12}$ samples and its experimental INL variance, SDR and SNDR were 0.184, 40.0 dB and 32.5 dB, respectively. The generalized cube network DEM DAC controlled by a colored uniformly distributed stochastic signal was also simulated using the same $40 \times 2^{12}$ samples and its experimental INL variance, SDR and SNDR were 0.184, 40.0 dB and 32.5 dB, respectively. The PSD of the output of the nonlinear DAC using the generalized cube network with white and colored control signals are shown in Figure 6.9 and Figure 6.10.

Figure 6.16. Power spectral density of the output of a six bit DEM DAC using a stochastic generalized cube network controlled by a white uniformly distributed stochastic control signal.
respectively. The PSD of the output distortion of the nonlinear DAC using the generalized cube network with a colored control signal is shown in Figure 6.18.

This example's results are summarized in Table 6.3 and Table 6.4.

6.3 Summary

In this chapter, the DEM DAC architecture in Figure 6.1 is analyzed and described by the transformation in (6.12). Using (6.12), expressions for the DAC's mean INL, the variance of the DAC's INL, the DAC's SDR, and the DAC's SNDR were developed and are described by (6.14), (6.15), (6.20) and (6.23), respectively. The expressions in (6.15),

![Figure 6.17. Power spectral density of the output of a six bit DEM DAC using a stochastic generalized cube network controlled by a colored uniformly distributed stochastic control signal.](image)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Table 6.3. Comparison of Example Two's SDR

<table>
<thead>
<tr>
<th>DEM technique</th>
<th>Derived SDR</th>
<th>Simulated SDR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stochastic Benes network</td>
<td>37.6 dB</td>
<td>37.6 dB</td>
</tr>
<tr>
<td>Clocked level averaging</td>
<td>38.9 dB</td>
<td>38.9 dB</td>
</tr>
<tr>
<td>Stochastic barrel shifting</td>
<td>38.9 dB</td>
<td>38.9 dB</td>
</tr>
<tr>
<td>Stochastic GCN (white)</td>
<td>40.0 dB</td>
<td>40.0 dB</td>
</tr>
<tr>
<td>Stochastic GCN (colored)</td>
<td>40.0 dB</td>
<td>40.0 dB</td>
</tr>
</tbody>
</table>

(6.20) and (6.23) were further simplified in (6.26), (6.27) and (6.28) for DAC's that have zero mean INL. Using these criteria, one deterministic DEM DAC and four stochastic

Figure 6.18. Power spectral density of a six bit DEM DAC's distortion output. The DEM DAC uses a stochastic generalized cube network controlled by a colored uniformly distributed stochastic control signal.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Table 6.4. Comparison of Example Two’s SNDR

<table>
<thead>
<tr>
<th>DEM technique</th>
<th>Derived SNDR</th>
<th>Simulated SNDR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stochastic Benes network</td>
<td>31.9 dB</td>
<td>31.9 dB</td>
</tr>
<tr>
<td>Clocked level averaging</td>
<td>32.2 dB</td>
<td>32.2 dB</td>
</tr>
<tr>
<td>Stochastic barrel shifting</td>
<td>32.2 dB</td>
<td>32.2 dB</td>
</tr>
<tr>
<td>Stochastic GCN (white)</td>
<td>32.5 dB</td>
<td>32.5 dB</td>
</tr>
<tr>
<td>Stochastic GCN (colored)</td>
<td>32.5 dB</td>
<td>32.5 dB</td>
</tr>
</tbody>
</table>

DEM DACs were analyzed, and the theoretical results are compared to experimental simulated results.
CHAPTER 7

DYNAMIC ELEMENT MATCHING NETWORK TOPOLOGIES
FOR DIGITAL TO ANALOG CONVERTERS

Many DEM algorithms for DACs rearrange mismatched circuit elements by reordering the bits of the digital inputs to the mismatched elements. These digital DEM algorithms often use complex interconnection network to virtually permute the mismatched components [6], [10], [15], [23], [34], [36], [64], [70], [71], [73], [79]. Several DEM DAC circuits in the literature are analyzed and shown to be well understood interconnection networks. Furthermore, several DEM specific, hardware efficient interconnection networks are derived from these standard interconnection networks. Finally, CMOS implementations of these hardware efficient networks are presented.

7.1 Interconnection networks

Interconnection networks, also referred to as circuit switching networks, are circuits that establish a connection between inputs and outputs through either a direct electrical connection or a logical path via logic gates.

Early interconnection networks were used for telephony applications and used direct electrical connections controlled by electromechanical relays. Today's interconnection networks typically utilize solid state switches or logic gates. Although logic gate interconnection networks do not have a direct electrical connection between their inputs...
and outputs, the analogy is maintained by the logical data path. Much interconnection network research currently focuses on their application for computer networks and component interconnections in parallel processing applications [9], [39], [49], [50].

Because interconnection networks are often based on graphs and analyzed by graph theory, several fundamental graph theory terms are reviewed in this chapter.

7.1.1 Node

A node is a point at which two or more elements have a common connection.

A node is sometimes called a vertex. Graphically, nodes are represented by small circles as shown in Figure 7.1.

7.1.2 Edge

An edge is a path between two nodes in a graph.

An edge is also referred to as a link, an arc, or a branch. Graphically, edges are represented by the line segments as shown in Figure 7.1.

![Figure 7.1. An example of a graph. Circles represent nodes and are labelled $n_1, ..., n_6$. The line segments represent edges and are labelled $e_1, ..., e_5$.](image-url)
7.1.3 Graph

A graph is mathematical structure that describes a set of elements and the connections between them.

A graph is completely characterized by two sets, a node set $N$ and an edge set $E$. Figure 7.1 shows a graph where $N = \{n_1, n_2, n_3, n_4, n_5, n_6\}$ and $E = \{e_1, e_2, e_3, e_4, e_5\}$.

7.1.4 Interchange switch

An interchange switch is a two input, two output switching cell that has two legitimate states, straight\(^1\) and exchange\(^2\).

Figure 7.2(a) shows the block diagram of the interchange switch. Figure 7.2(b) shows the interchange switch in the straight setting, which means the upper input is connected to the upper output and the lower input is connected to the lower output. Figure 7.2(c) shows the interchange switch in the exchange setting, which means the upper input is connected to the lower output and the lower input to the upper output. The control signal required to select the straight and exchange settings are traditionally logical zero and logical one,

\[ c = 0 \]
\[ c = 1 \]

Figure 7.2. Two input, two output interchange switch (a) block diagram, (b) straight setting, and (c) exchange setting.

\(^1\) The straight setting is also called *pass-through*.

\(^2\) The exchange setting is also called *swap*.
respectively. The meaning of the interchange switch's control signal states can be reversed where the control signal required to select the straight and exchange settings are logical one and logical zero, respectively. One meaning can be assumed without a loss of generality. If the meanings are reversed, identical results are obtained and the theory still holds. In this dissertation, the interchange switch's operation is defined as shown in Figure 7.2.

7.1.5 Relationship between graphs and interconnection networks

An interconnection network can be derived from a graph by applying rules which assign specific circuit elements to edges and nodes.

An interconnection network is a circuit topology based a graph. A graph can be the basis for several different interconnection networks depending on the rules which assign circuit elements to edges and nodes. For example, Figure 7.3(a) shows a graph with eight nodes and eight edges. Figure 7.3(b) shows an interconnection network constructed when the graph edges in Figure 7.3(a) are replaced with interchange switches. In Figure 7.3(a), node 0 connects to node 4 via a horizontal edge and node 6 via a diagonal edge. Node 2 connects to node 4 via a diagonal edge and node 6 via a horizontal edge. These four edges are replaced by the upper interchange switch in Figure 7.3(b). Similarly, the edges between nodes 1, 3, 5, and 7 are replaced with the lower interchange switch in Figure 7.3(b). The interchange switches implement the horizontal edge connectivity when set to the straight setting and the diagonal edge connectivity when set to the exchange setting. Figure 7.3(c) shows an interconnection network constructed when the nodes in
Figure 7.3(a) are replaced with interchange switches. Both interconnection networks in Figure 7.3(b) and Figure 7.3(c) are based on the graph in Figure 7.3(a).

Figure 7.3. Example graph and corresponding interconnection networks. (a) Four input graph. (b) Interconnection network generated by replacing edges with interchange switches. (c) Interconnection network generated by replacing nodes with interchange switches.
7.1.6 Full access network

An interconnection network is full access if the network can make a connection between any input and any output.

Figure 7.4 shows a full access network. For example, input 7 on the left can be connected to outputs 0 and 1 by setting the interchange switches DFI to 111 and 110, respectively. Input 7 can be connected to outputs 2 or 3 when the interchange switches DFJ to 101 and 100, respectively. Input 7 can be connected to outputs 4 or 5 when the interchange switches DHK to 011 and 010, respectively. Input 7 can be connected to outputs 6 and 7 when the interchange switches DHL to 001 and 000, respectively. Connections from the other inputs to all outputs can be made similarly.

Figure 7.4. An example interconnection network. The interchange switches have been labelled and their control signals have been omitted for compactness.
7.1.7 Partitionability

An interconnection network is said to be partitionable if it can be divided into independent subnetworks, possibly of different sizes.

A partitionable network can be characterized by any limitation on the way in which it can be subdivided [66]. For example, the interconnection network in Figure 7.4 can be partitioned in many different ways. The interconnection network in Figure 7.4 can be divided into two subnetworks using interchange switches ABEFIJ and CDGHKL, respectively. The interconnection network in Figure 7.4 can also be divided into three subnetworks using interchange switches ABCD, EFGH, and IJKL, respectively.

7.1.8 Recursion

An interconnection network is recursive if it contains two identical independent subnetworks of the same type as the larger network.

Each subnetwork must have all of the interconnection capabilities of a network of its type. Recursion is a more restrictive form of network partitioning. In other words, recursive networks are always partitionable, but partitionable networks are not necessarily recursive. Many switching networks based on the interchange switch are recursive. For example, the interconnection network in Figure 7.4 is recursive. The two smaller networks using interchange switches EFIJ and GHKL, respectively, have the same form as the complete interconnection network in Figure 7.4. The interchange switches ABCD direct the interconnection inputs to the upper or lower smaller networks.
7.1.9 Permutations

A permutation is a bijection⁴ that is from one set onto the same set.

For example, one permutation of the ordered list \( \{0,1,2,3,4,5,6,7\} \) is \( \{0,3,2,5,1,6,4,7\} \). If the input to the interconnection in Figure 7.4 is \( \{0,1,2,3,4,5,6,7\} \), the permutation, \( \{0,3,2,5,1,6,4,7\} \), can be achieved at the output of the interconnection network by setting the interchange switches A, C, D, E, H, I, J, and L to straight and the interchange switches B, F, G, and K to exchange.

7.1.10 Combinatorial power

A switching network's combinatorial power is defined as the number of permutations the network can perform divided by the total number of possible permutations.

For example, a network which can perform all possible permutations has a combinatorial power of one. If a particular switching network with eight inputs and eight outputs can only perform circular shifts of its inputs, the networks combinatorial power is \( 8/(8!) = 0.00019841 \).

7.2 Interconnection networks classes

By classifying interconnection networks according to their ability to connect inputs to outputs, interconnection networks can be divided into three classes, nonblocking, rearrangeable and blocking.

---

³ A bijection is a one-to-one and onto mapping.
7.2.1 Nonblocking networks

A network is said to be nonblocking if any input can be connected to any output without interference from the existing connections. A network is said to be strictly nonblocking if any free input can be connected to any free output without interference from the existing connections. If any input can be connected to any output without interference from the existing connections provided that the connections have been allocated due to a network specific algorithm, then the network is said to be wide sense nonblocking.

7.2.1.1 Crossbar network

Crossbar based networks were developed for early telephone switching applications. Although the physical crossbar switches are no longer used, the term is commonly applied to networks of this form. The square crossbar network is a strictly nonblocking network. The crossbar network consists of \( N^2 \) controllable switches arranged in an \( N \times N \) array. The square crossbar network grows impractically large for large values of \( N \). Figure 7.5 shows an eight signal crossbar graph. Replacing each edge in Figure 7.5 with a switch generates the eight signal crossbar network in Figure 7.6. As illustrated in Figure 7.6, the crossbar network can connect any input to any output, regardless of the existing connections. The control of input/output connections is straightforward. For example, setting all third row switches to close the horizontal path except for the switch in the sixth column, which is set to the vertical path, will connect the third input to the sixth output.
Figure 7.5. Eight input crossbar graph.

Figure 7.6. Eight input crossbar network. The switches’ control signals have been omitted for compactness.
7.2.1.2 Benes network

The Benes network is a recursive, full access interconnection network, and has an algorithm for allocating paths in the network [8]. Therefore, the Benes network is wide sense nonblocking. The $r$ bit Benes graph has $2^r$ rows, $2r$ stages, $2r+1$ levels, and $2^r(2r+1)$ nodes. For example, Figure 7.7 shows the three bit Benes graph which has eight rows, six stages, seven levels, and 56 nodes.

Two pairs of nodes in the Benes graph which are connected by both horizontal and diagonal edges form a partition. By replacing these partitions with an interchange switch, an interconnection network based on the Benes graph can be constructed. The small number above each diagonal edge in Figure 7.7 shows the location of interchange switches in the three bit Benes network in Figure 7.8. For example, consider the four nodes in the

![Figure 7.7. Three bit Benes graph. The number represent the location of interchange switches.](image)

---

4 Another network can be constructed by placing an interchange box at each node [70]. The resulting network will have twice as many inputs and outputs as the Benes network used in this dissertation.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
fourth and sixth rows at levels one and two. The connectivity between these four nodes can be implemented with the interchange switch labelled 19 in Figure 7.8.

The $r$ bit Benes network contains $r2^r$ interchange switches. The three bit Benes network in Figure 7.8 contains 24 interchange switches, four in each stage. Furthermore, the Benes network in Figure 7.8 contains two four input Benes networks. The upper four input Benes network is constructed with interchange switches 5, 6, 9, 10, 13, 14, 17, and 18. The lower four input Benes network is constructed with interchange switches 7, 8, 11, 12, 15, 16, 19, and 20.

![Figure 7.8](image)

Figure 7.8. An eight input Benes network. The network is redrawn to explicitly show the connections between the interchange switches which are numbered to demonstrate the relationship with Figure 7.7. The interchange switches' control signals have been omitted for compactness.
7.2.1.3 Other nonblocking networks

Other known nonblocking networks include Clos' network [17] and Cantor's network [12]. These networks have potential use in digital DEM algorithms for DACs. The details of which are not discussed in this dissertation and remain an open area of research.

7.2.2 Rearrangeable networks

A network is said to be rearrangeable if any input can be connected to any output, even though some existing connections might require rerouting.

For example, the Waksman network is a recursive, rearrangeable network that is related to the Benes network in structure [75]. Like the Benes network, the Waksman network has \(2^r\), where \(r\) is an integer, inputs and an identical number of outputs. The Waksman network is a Benes network where unnecessary interchange switches are

![Figure 7.9. An eight input Waksman permutation network. The interchange switches' control signals have been omitted for compactness.](image)

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
omitted. The omitted interchange switches are not needed to obtain all possible permutations. The omission of switches results in the Waksman network being rearrangeable and not wide sense nonblocking like the Benes network. The $r$ bit Waksman network has a maximum of $2r-1$ levels of switches. The number of interchange switches from input to output is not constant and is path dependent. Figure 7.9 shows an eight input Waksman network. Waksman’s network has not been used in digital DEM algorithms for DACs. The details of using Waksman’s network for digital DEM algorithms remain an open area of research.

7.2.3 Blocking networks

A network is said to be blocking if there exists a connection set that prevents some additional desired connections from being established between unused inputs and outputs.

In general, nonblocking and rearrangeable networks are large and require complex control algorithms. Blocking networks require much less hardware and have much simpler control algorithms compared to nonblocking and rearrangeable networks. Most blocking networks have only a single connection path between any pair of inputs and outputs and are full access networks with combinatorial power much less than one [66]. Although, many different blocking networks exist, only the indirect binary cube, generalized cube and barrel shift networks have been used for digital DEM algorithms in DACs.

7.2.3.1 Indirect binary cube network

The indirect binary cube network (IBCN) is a full access network that is based on the butterfly graph and the computation of the fast Fourier transform [45]. An $r$ stage butterfly
graph has $2^r$ rows, $2^r(r+1)$ nodes and $r2^{r+1}$ edges. Each node in the butterfly graph can be addressed with the ordered pair, $(w,i)$, where $w$ denotes the node's row and $i$ for $0 \leq i \leq r$ is the node's level. Two nodes, $(w,i)$ and $(w',i')$, in an $r$ stage butterfly graph are linked by an edge if and only if

1. $i' = i+1$,

and

2. $w = w'$ or $w$ and $w'$ differ in the $i$th bit when $w$ and $w'$ are interpreted as $r$ bit binary numbers.

For example, Figure 7.10 shows a three stage butterfly graph that has 8 rows, 32 nodes, and 48 edges. Node $(5,3)$ in the three bit butterfly graph connects to node $(5,2)$ since $i' = i+1$ and $w = w'$. Node $(5,3)$ also connects to node $(4,2)$ because $5 = (101)_2$ and $4 = (100)_2$ differ in the third bit and $i' = i+1$.

![Figure 7.10. Three stage butterfly graph. The numbers represent the location of interchange switches.](image-url)
Two pairs of nodes in the butterfly graph which are connected by both horizontal and diagonal edges form a partition. By replacing these partitions with an interchange switch, an interconnection network based on the butterfly graph can be constructed using \( r2^{r-1} \) interchange switches\(^5\). The small number above the diagonal edges in Figure 7.10 shows the location of the interchange switches for the three bit case. For example, Figure 7.11 shows the interconnection network, called the indirect binary cube network, based on the butterfly graph in Figure 7.10. This interconnection network requires twelve interchange switches, four in each stage. The IBCN is recursive [57]. In illustration, the three stage IBCN in Figure 7.11 contains two smaller IBCNs. The upper two bit IBCN is constructed

![Diagram of a three stage IBCN](image)

Figure 7.11. A three stage IBCN. The network is redrawn to explicitly show the connections between the interchange switches which are numbered to demonstrate the relationship with Figure 7.10. The interchange switches’ control signals have been omitted for compactness.

\(^5\) The butterfly graph is the basis for another interconnection network constructed by placing an interchange box at each node. This network, called the SW-Banyan network, has twice as many inputs and outputs as the IBCN [9], [24].

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
with interchange switches 1, 2, 5, and 6; the lower two bit IBCN is constructed with interchange switches 3, 4, 7, and 8.

An interchange switch in stage \( i \), for \( 0 \leq i \leq B - 1 \) is connected to two input edges, if and only if the edge labels differ in precisely the \( i \)th bit position. Furthermore, if the interchange switch is set to exchange, the edge labels' \( i \)th bit will be complemented. Thus, if an input \( a = a_{r-1}a_{r-2}...a_1a_0 \) is to be routed to an output \( b = b_{r-1}b_{r-2}...b_1b_0 \), then the interchange switches should be set to straight setting for the stages given by \( a \oplus b \), where \( \oplus \) is the binary exclusive-or operator. The product, \( a \oplus b \), gives the bit positions where the input, \( a \), and the output, \( b \), differ in their respective labels. The positions in which bits should be complemented gives the stage of the IBCN which is to perform the complementation. For example, consider connecting the input with label 1 to the output with label 2 in the three stage IBCN in Figure 7.11. To connect input \( 1 = (001)_2 \) to output \( 2 = (010)_2 \), the stages to set to exchange are given by the exclusive-or operation on the

![Diagram](image)

Figure 7.12. Example of a permutation in a three bit IBCN.

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
input and output labels, e.g. \(1 \oplus 2 = (001)_2 \oplus (010)_2 = (011)_2\). Therefore, the interchange switches in the stages zero and one are set to exchange, while the interchange switches in the stage two are set to the straight.

The IBCN is blocking. For example, if input 1 is connected to output 2 as shown in Figure 7.11, then input 0 cannot be connected to the outputs 0, 4 or 6 because the stage zero interchange switches are set to exchange. Furthermore, since the network connects input 1 to output 2, all interchange switches in stages zero and one are set to exchange and all interchange switches in stage two are set to straight. Therefore all connections are established such that the input pattern 01234567 is mapped to 32107654.

7.2.3.2 Generalized cube network

The generalized cube network (GCN), also called the multistage cube network \([66]\), is a full access, blocking network based on the structure of the hypercube. The generalized cube graph is a reversed butterfly graph. As a result, the GCN and the IBCN are closely related. An \(r\) stage generalized cube graph has \(2^r\) rows, \(2^r(r+1)\) nodes and \(r2^{r+1}\) edges. Each node in the generalized cube graph can be addressed with the ordered pair, \((w,i)\), where \(w\) denotes the node’s row and \(i\), for \(0 \leq i \leq r\), is the node’s level. Two nodes, \((w,i)\) and \((w',i')\), in an \(r\) stage generalized cube graph are linked by an edge if and only if

1. \(i' = i - 1\),

and

2. \(w = w'\) or \(w\) and \(w'\) differ in the \(r-i\)th bit when \(w\) and \(w'\) are interpreted as \(r\) bit binary numbers.
For example, consider the three stage generalized cube graph in Figure 7.13. Node \((5, 3)\) is connected to node \((5, 2)\) since \(i' = i - 1\) and \(w = w'\). Also, node \((5, 3)\) in the generalized cube graph is connected to node \((1, 2)\) because \(w = 5 = (101)_2\) and \(w' = 1 = (001)_2\) differ in the first bit and \(i' = i - 1\).

In the generalized cube graph, two pairs of nodes which are connected by both horizontal and diagonal edges form a partition. By replacing these partitions with an interchange switch, an interconnection network based on the generalized cube graph can be constructed using \(r2^{r-1}\) interchange switches\(^6\). A \(2^r\) input GCN requires \(r\) independent control signals and contains \(r\) stages, each of which have \(2^{r-1}\) interchange switches. Because each switch in the GCN has two different states, the network can

---

\(^6\) Like the Benes and butterfly graphs, the generalized cube graph is the basis for another interconnection network constructed by placing an interchange box at each node. This network has twice as many inputs and outputs as the GCN.

---

Figure 7.13. Three stage generalized cube graph. The numbers represent the location of interchange switches.
generate $2^r$ different permutations of the input signals. The number of realizable permutations can be increased to $2^r 2^{r-1}$, if all of the binary switches are operated independently. Each permutation is guaranteed to be unique because the network is a full access network. For example, Figure 7.14 shows a three stage GCN. The network in Figure 7.13 can be constructed with twelve interchange switches, four in each stage. The small number above each cross edge enumerates the interchange switches used in the construction of a GCN. The three bit generalized cube graph in Figure 7.13 is redrawn to explicitly show the interchange switches and their connections in Figure 7.14. The GCN is a recursive network because one $r$ stage network contains two $(r-1)$ stage GCNs as subgraphs as shown in Figure 7.15.

![Figure 7.14. A three stage GCN. The network is redrawn to explicitly show the connections between the interchange switches which are numbered to demonstrate the relationship with Figure 7.13. The interchange switches' control signals have been omitted for compactness.](image-url)
An interchange switch in stage $i$ for $0 \leq i \leq B - 1$ is connected to two input edges, if and only if the edge labels differ in precisely the $i$th bit position. Furthermore, if the interchange switch is set to exchange, the edge labels' $i$th bit is complemented. Thus, if an input $a = a_{r-1}a_{r-2}...a_1a_0$ is to be routed to an output $b = b_{r-1}b_{r-2}...b_1b_0$, then the interchange switches should be set to straight setting for the stages given by $a \oplus b$, where $\oplus$ is the binary exclusive-or operator. The product, $a \oplus b$, gives the bit positions where the input, $a$, and the output, $b$, differ in their respective labels. The bit positions which are complemented indicate the stage of the GCN which performs the complementation. In illustration, consider connecting the input with label 7 to the output with label 2 in the three stage GCN shown in Figure 7.14. To connect input $7 = (111)_2$ to output $2 = (010)_2$, the stages to set to exchange are given by the exclusive-or operation on the input and output labels, e.g. $7 \oplus 2 = (111)_2 \oplus (010)_2 = (101)_2$. Therefore, the interchange switches in the 0th and 2nd stage are set to exchange, while the interchange

![Diagram](image)

**Figure 7.15.** Construction of a $r+1$ bit GCN.
switches in the 1st stage are set to the straight. Because all of the stage two interchange switches in Figure 7.16 are set to exchange, certain permutations cannot be obtained. For example, input zero cannot be connected to outputs zero through three, and therefore, the GCN in Figure 7.16 is a blocking network. In addition, if input 7 is connected to output 2, the interchange switches in stages 1 are set to pass through and the interchange switches in stage 0 are set to exchange. Therefore all connections are established such that the input pattern 01234567 is mapped to 54761032.

7.2.3.3 Barrel shift network

The barrel shift network (BSN) is a full access network which is so named because it performs operations similar to a circular shift register. An $r$ stage barrel shift graph has $2^r$ rows, $2^r(r+1)$ nodes and $r2^{r+1}$ edges. Each node in the barrel shift graph is identified
with the ordered pair, \((w, i)\), where \(w\) denotes the node's row and \(i\), for \(0 \leq i \leq r\), is the node's level. Two nodes, \((w, i)\) and \((w', i')\), in an \(r\) stage BSN are linked by an edge if and only if

1. \(i' = i - 1\),

and

2. \(w = w'\) or \(w\) and \(w'\) differ by \(2^i\) in modulo \(2^r\) arithmetic.

For example, Figure 7.17 shows a three stage barrel shift graph which has eight rows, 32 nodes, and 48 edges. Node \((6, 2)\) in the three bit barrel shift graph is connected to node \((6, 1)\) since \(i' = i - 1\) and \(w = w'\). Also, node \((6, 2)\) in the three bit barrel shift graph is connected to node \((0, 1)\) because \(0 = (6 + 2^1) \mod 8\) and \(i' = i - 1\).

It is easily seen that four node, four edge partitions which are found in the butterfly and generalized cube graphs are not present in the barrel shift graph. Thus, the BSN cannot be constructed with the interchange switch in Figure 7.2. The BSN can be constructed by placing transmission gates on the horizontal and diagonal edges of the barrel shift graph.

![Three stage barrel shift graph](image)

**Figure 7.17.** Three stage barrel shift graph.
All horizontal edges transmission gates in a stage are open and all diagonal edge transmission gates in a stage are closed when a shift is performed. Conversely, all horizontal edge transmission gates are closed and the diagonal edge transmission gates are open when a shift is not performed. Figure 7.18 shows the locations of transmission gates in a three bit BSN.

Because pairs of edges are incident on an interchange switch in stage \( i \), for \( 0 \leq i \leq B - 1 \), if and only if their labels differ by precisely \( 2^n \) in modulo \( 2^r \) arithmetic, it is readily seen that if the interchange switch is set to exchange, the edge labels' will incremented by \( 2^n \) in modulo \( 2^r \) arithmetic. Thus, to route an input with label \( a = a_{r-1}a_{r-2}...a_1a_0 \) to an output with label \( b = b_{r-1}b_{r-2}...b_1b_0 \), the interchange switches should be set to straight setting for the stages where \( (b-a) \mod 2^r \) is nonzero. The modulo difference, \( (b-a) \mod 2^r \), indicates the stages of the BSN which are to perform the addition. For example, Figure 7.19 shows the connection between the input 7 and output 2 in the three stage BSN. To connect input 7 = (111)\(_2\) to output 2 = (010)\(_2\), the stages to set

![Diagram](image_url)
to exchange are given by the modulo difference of the input and output labels, e.g. 
\[(2-7) \mod 8 = 3 = (011)_2\]. Therefore, the cross edges in the stages one and two are 
traversed, and the straight edges in the stage zero are used. Because all of the stage two 
straight edges in Figure 7.19 are used, certain permutations cannot be obtained. For 
extample, input 0 cannot be connected to the outputs 4, 5, 6 or 7. It is easily seen that the 
BSN in Figure 7.19 is highly blocking. In addition, since it is specified to connect input 7 
to output 2, all straight edges in stage two and all cross edges in stages zero and are used. 
Therefore all connections are established such that the input pattern 01234567 is mapped 
to 56701234.

7.2.3.4 Other blocking networks

The omega network, also called the shuffle-exchange network, is a full access blocking 
network [43]. An \(r\) stage omega network has \(2^r\) rows, \(2^r(r+1)\) nodes, \(r2^{r+1}\) edges and 
is constructed with \(r2^{r-1}\) interchange switches. The omega network inputs are labeled

![Diagram of an Omega Network](image)

Figure 7.19. Example of a permutation in a three bit BSN. The BSN is presented in 
graph representation for clarity.
from 0 to $2^r - 1$. The connectivity of the omega network is described by the following rule: an edge is made between the rows whose number differs by a left circular shift when the row numbers are represented by a $r$ bit binary number. For example, in the three stage omega network in Figure 7.20, input $5 = (101)_2$ is connected to row $3 = (011)_2$ at the first stage which is connected to row $6 = (110)_2$ in the second stage which is connected to row $5 = (101)_2$ in the third stage.

The omega network is topologically equivalent, i.e. equal combinatorial power, to many other networks, including the IBCN and the GCN. These topologically equivalent networks are known as omega class networks [76]. All $r$ stage omega class networks have $2^r$ rows, $2^r(r+1)$ nodes, $r2^r+1$ edges and can be constructed with $r2^{r-1}$ interchange switches. All omega class networks are full-access blocking networks. Typically, different omega class networks implement a different set of input-output paths when each switch is

![Figure 7.20. A three bit omega network. The interchange switches' control signals have been omitted for compactness.](image)
controlled independently. When the switches common to a stage are controlled by the same signal, all omega class networks perform the same permutations. Interestingly, a nonblocking network can be constructed by connecting \( r \) omega class networks end-to-end. However, the number of stages which the signal will pass through will be much greater than the comparable Benes network.

Nonblocking networks are suitable for use in digital DEM algorithms for DACs even though they have combinatorial power much less than one. As will be seen in Section 7.3.3 and Section 7.3.4, the IBCN and GCNs have been used in digital DEM algorithms for DACs. It is expected that the other omega class networks and other nonblocking networks will provide similar results.

### 7.3 Interconnection networks for DEM in DACs

Interconnection networks are well suited for DEM algorithms in DACs [15]. Figure 7.21 shows a \( B \) bit DEM flash DAC architecture which uses a \( 2^B \) line interconnection network to randomized the unit DAC inputs. Because nonblocking

---

**Figure 7.21.** A \( B \) bit dynamic element matching flash DAC architecture.
interconnection networks possess full combinatorial capability, they can virtually rearrange circuit components into every possible permutation. However, nonblocking networks require a larger number of switching elements when compared to rearrangeable and blocking interconnection networks. Rearrangeable networks have not been used in DEM architectures for DACs thus far. Because they have full combinatorial capability with fewer switching elements than nonblocking networks, rearrangeable networks could be used to perform DEM; however, rearrangeable network switches require more complex control signals than nonblocking and blocking networks. Although blocking networks cannot realize all input-output permutations, they are commonly used in DEM architectures because they require fewer switching elements compared to nonblocking and rearrangeable networks. Also, blocking networks are often recursive which allows for hardware efficient and regular VLSI implementations.

In this section, nonblocking and blocking networks are examined for use in the DEM DAC architecture in Figure 7.21. Several blocking networks are shown to be equivalent and hardware efficient versions are introduced. Finally, the hardware efficient networks are examined for VLSI implementation.

7.3.1 Terminology

Traditionally, metrics for comparing interconnection networks assume that each input, output and path through the network is distinguishable. Because the interconnection network’s inputs are boolean, many input-output permutations are indistinguishable when used in digital DEM algorithms. Therefore, two new terms, logical permutations and
logical combinatorial power, are introduced for comparing the capabilities of DEM interconnection network.

7.3.1.1 Logical permutation

When switching network inputs are binary, several different network input patterns can generate identical output patterns. Each distinguishable mapping from an input bit pattern to an output bit pattern is called a logical permutation. For example, an eight input, eight output interconnection network with input \{1,1,1,1,1,1,1,1\} would generate an output \{1,1,1,1,1,1,1,1\} regardless of the interconnection network's number of permutations. Thus, for DEM interconnection networks that rearrange digital signals, the number of logical permutations is a measure of a network's number of distinguishable input-output mappings.

7.3.1.2 Logical combinatorial power

A switching network's logical combinatorial power is defined as the number of logical permutations the network can perform divided by the total number of possible logical permutations.

For example, if a particular switching network with eight inputs and outputs can only perform circular shifts of its inputs, the network has one logical permutation for the thermometer coded input for zero, eight logical permutations for the thermometer coded inputs for one through seven and one logical permutation for the thermometer coded input for eight. Thus, the total number of bit permutations is 58. The total number of logical
permutations for eight binary output is $2^8$, and the network's logical combinatorial power is $58/256 = 0.2265625$.

7.3.2 Benes DEM network

In the $B$ bit DEM DAC architecture in Figure 7.21, the $2^B$ unit DAC control signals are rearranged by the $B$ bit Benes network. The $2^B$ bit signal is created by appending a logical zero to the $2^B - 1$ bit thermometer coded representation of the $B$ bit DAC input, $x(n)$. The Benes network's outputs control the $2^B$ unit DACs. Figure 7.22 shows an example of a three bit Benes network used in the DEM DAC architecture Figure 7.21. If the Benes network's control signals, $c_k(n)$ for $k = 0, 1, ..., B2^B - 1$, are random bit sequences, the input signal, $x(n)$, activates $x(n)$ unit DACs randomly each sample. For a $B$ bit DEM DAC, the Benes network requires $B2^B$ control signals and $B2^B$ interchange

![Figure 7.22. A three bit Benes network DEM DAC architecture. The interchange switch control signals have been omitted for compactness.](image)

\[\text{This example assumes that the thermometer coded bits are ordered such that the maximum number of logical permutations are achieved.}\]
switches to generate \((2^B)!\) input-output permutations \([8], [45], [54]\). The number of logical permutations for the Benes network given an input code, \(x\), is the number of ways to set \(x\) bits of the possible \(2^B\) output lines, or \(\binom{2^B}{x} = \frac{(2^B)!}{(2^B-x)!x!}\). Using the binomial theorem \([61]\),

\[
(x + y)^n = \sum_{k=0}^{n} \binom{n}{k} x^k y^{n-k},
\]

the total number of logical permutations for the Benes network is

\[
\sum_{k=0}^{2^B} \binom{2^B}{k} = (1 + 1)^{2^B} = 2^{2^B}.
\]

Because many Benes network inputs are tied together when used in DEM algorithms, a more hardware efficient network exists which is equivalent to the Benes network. Because a Benes network can implement all possible input-output permutations and many of these permutations are indistinguishable when the network inputs are binary, the Benes network generates many input-output mappings unnecessary for DEM algorithms in DACs.

A DEM DAC using a Benes network has been simulated and analyzed \([23]\); however, the DEM DAC has not be manufactured because the Benes network requires a large number of interchange switches and independent control signals.

### 7.3.3 Indirect binary cube DEM network

In the \(B\) bit DEM DAC architecture in Figure 7.21, the \(2^B\) unit DAC control signals are rearranged by the \(B\) bit IBCN. The \(2^B\) bit signal is created by appending a logical zero to the \(2^B-1\) bit thermometer coded representation of the \(B\) bit DAC input, \(x(n)\). The
IBCN outputs control the $2^B$ unit DACs. Figure 7.23 shows an example of a three bit IBCN used in the DEM DAC architecture shown in Figure 7.21. If the IBCN’s control signals, $c_k(n)$ for $k = 0, 1, \ldots, B-1$, are random bit sequences, the input signal, $x(n)$, activates $x(n)$ unit DACs randomly each sample. For a $B$ bit DEM DAC, the IBCN requires $B$ control signals and $B2^{B-1}$ interchange switches to generate $2^B$ input-output permutations. If the interchange switches in a stage are not switched together, the IBCN requires $B$ control signals and $B2^{B-1}$ interchange switches to generate $2^{B2^{B-1}}$ input-output permutations.

Because many IBCN inputs are tied together when used in DEM algorithms, a more hardware efficient network exists which is equivalent to the IBCN. Consider the three bit IBCN in Figure 7.11 or the three bit IBCN DEM DAC architecture in Figure 7.23. The IBCN’s symmetry and the thermometer code bit replication render a number of the

![Three bit IBCN DEM DAC architecture](image)

**Figure 7.23.** Three bit IBCN DEM DAC architecture. The interchange switch control signals have been omitted for compactness.
interchange switches redundant. For example, at the end of the first two stages in Figure 7.23, the four copies of the DAC input's MSB are adjacent. Thus, the four interchange switches can be removed without affecting the operation of the network. Similarly, the two copies of the second MSB of the DAC input are still adjacent after the first interchange switch, so it can be removed. Figure 7.24 shows a three bit DEM DAC architecture using the hardware efficient IBCN. For a $B$ bit DEM DAC, the hardware efficient IBCN requires $B$ control signals and $2^{B-1}$ interchange switches. The number of logical permutations and logical combinatorial power of the hardware efficient IBCN is the same as the IBCN in Figure 7.23. The hardware efficient IBCN in Figure 7.24 has the same number of switches the interconnection network introduced in [35].

Figure 7.24. Three bit reduced complexity IBCN DEM DAC. The interchange switch control signals have been omitted for compactness.
A three bit DAC has been implemented using three cascaded IBCN networks which provide full combinatorial capability in a three bit converter. This DAC was used as the feedback DAC in a multibit oversampling ADC [15].

7.3.4 Generalized cube DEM network

In the $B$ bit DEM DAC architecture in Figure 7.21, the $2^B$ unit DAC control signals are rearranged by the $B$ bit GCN. The $2^B$ bit signal is created by appending a logical zero to the $2^B-1$ bit thermometer coded representation of the $B$ bit DAC input, $x(n)$. The GCN outputs control the $2^B$ unit DACs. Figure 7.25 shows an example of a three bit GCN used in the DEM DAC architecture shown in Figure 7.21. If the GCN's control signals, $c_k(n)$ for $k = 0, 1, ..., B-1$, are random bit sequences, the input signal, $x(n)$, activates $x(n)$ unit DACs randomly each sample. For a $B$ bit DEM DAC, the GCN requires $B$

![Figure 7.25. Three bit GCN DEM DAC architecture. The interchange switch control signals have been omitted for compactness.](image-url)
control signals and $B2^{B-1}$ interchange switches to generate $2^B$ input-output permutations.

To determine the number of logical permutations in a $B$ bit GCN DEM DAC where all interchange switches in a single stage are switched together, consider Figure 7.15 and Figure 7.25. If $C_{g}(n) = 0$, then regardless of the other control signals, the $2^{B-1}$ copies of the most significant bit of $x(n)$ are routed to the $2^{B-1}$ upper network outputs. Similarly, if $C_{g}(n) = 1$, then regardless of the other control signals, the $2^{B-1}$ copies of the most significant bit of $x(n)$ are routed to the $2^{B-1}$ lower network outputs. Because the GCN is recursive, the number of logical permutations for the DAC input $2k$, where $k$ is an integer, in a $B$ bit GCN DEM DAC, is equal to the number of logical permutations for the DAC input $k$ in the $B-1$ bit GCN DEM DAC. For odd DAC inputs in a $B$ bit GCN DEM DAC, the logical zero paired with a logical one can be placed in $2^B$ locations. Thus, the number of logical permutations for an odd DAC inputs in a $B$ bit GCN DEM DAC is $2^B$. Therefore, the total number of logical permutations that a $B$ bit GCN DEM DAC can generate is

$$2 + \sum_{k=1}^{B} 2^{2k-1} = 2 + \frac{2(4^B-1)}{3}.$$  

If the interchange switches are switched independently, the GCN DEM DAC requires $B$ interchange switch control signals and $B2^{B-1}$ interchange switches, while generating $2^{B2^{B-1}}$ input-output permutations.

Because many GCN inputs are tied together in Figure 7.25, a more hardware efficient network exists which is equivalent to the GCN. Two such networks, the "Full
Randomization" network and the binary tree network, are discussed the following sections.

7.3.4.1 "Full Randomization" DEM network

In [34], the thermometer coder and the $2^B$ line interconnection network shown in Figure 7.21 were implemented using a single multistage interconnection network named the Full Randomization\(^8\) dynamic element matching (FRDEM) network.

Figure 7.26 shows a three bit FRDEM DAC architecture. The FRDEM DAC architecture in Figure 7.26 contains a series of switching blocks, $S_{k,r}$, where $S_{k,r}$

---

\(^8\) The authors in [34] named their network the "Full Randomization" DEM network because it produces results similar to nonblocking networks. However, the FRDEM network has combinatorial power much less than one.
represents the r th switching block in the k th layer. Figure 7.27 shows the detail of the switching block, $S_{k,r}$. The internal switch in each switching block operates according to the interchange switch operation shown in Figure 7.2 except that the switching block in Figure 7.27 operates on k bits. The switching block’s input, $b(n)$, is a k+1 bit signal. When the control signal, $c_{k-1}(n)$, is a logical zero, $S_{k,r}$ sends $b(n)$'s k LSBs to the lower outputs and k copies of $b(n)$'s MSB to the upper outputs. When the control signal, $c_{k-1}(n)$, is a logical one, these outputs are exchanged, i.e. $S_{k,r}$ sends $b(n)$'s k LSBs to the upper outputs and k copies of $b(n)$'s MSB to the lower outputs. Figure 7.28 shows the switching block, $S_{0,1}$, which is an interchange switch and a one bit FRDEM network. In general, a k+1 bit FRDEM network can be constructed by connecting two k bit FRDEM networks with the switching block, $S_{k,1}$. Figure 7.29 shows the construction of a k+1 bit FRDEM network.

Figure 7.27. The “Full Randomization” DEM network’s switching block.
In general, a $B$ bit FRDEM DAC architecture requires a FRDEM network with $B$ layers, $2^{B-1}$ switching blocks, $B$ control signals, and $2^B$ unit DAC elements. Also, Layer $k$ contains $2^{B-k}$ switching blocks, each of which have one $(k+1)$ bit input signals and two $k$ bit output signals. To use a $B$ bit FRDEM network in a $B$ bit FRDEM DAC, a logical zero is appended to the LSB of the $B$ bit input signal, $x(n)$, and this $B+1$ bit signal is used by the $B$ bit FRDEM network. The FRDEM network outputs control the $2^B$ unit DACs. If the FRDEM network's control signals, $c_k(n)$ for $k = 0, 1, ..., B-1$, are random bit sequences, the input signal, $x(n)$, activates $x(n)$ unit DACs randomly each sample.

Figure 7.28. A one bit “Full Randomization” DEM network.

Figure 7.29. Construction of a $k+1$ bit “Full Randomization” DEM network.
7.3.4.2 Binary tree DEM network

Another hardware efficient implementation of the GCN is the binary tree network (BTN). The BTN is so named because it resembles a binary tree. Similar to the FRDEM network, the BTN implements the operations of both the thermometer coder and the $2^B$ line interconnection network shown in Figure 7.21. Figure 7.30 shows a three bit BTN. In general, a $k+1$ bit BTN can be constructed by connecting two $k$ bit BTNs as shown in Figure 7.31. When the control signal, $c_k(n)$, is a logical zero, the network sends $2^k$ copies of $b(n)$'s MSB to the upper outputs and $b(n)$'s $k$ LSBs to the lower $k$ bit BTN which generates $2^k$ outputs from these LSBs. When the control signal, $c_k(n)$, is a logical one, these outputs are exchanged. The BTN's high impedance output can be pulled to an appropriate logic level by external circuitry.

![Figure 7.30. A three bit BTN DEM DAC architecture.](image-url)
In general, a $B$ bit BTN DEM DAC architecture requires a BTN with $B$ layers, $2^{B+1} + B2^B - 2B - 2$ transmission gates, $B$ control signals, and $2^B$ unit DAC elements. To use a $B$ bit BTN in a $B$ bit BTN DEM DAC, the $B$ bit DAC input signal, $x(n)$, is applied to a $B$ bit BTN. The BTN's single high impedance output is pulled to a logical zero. The resulting $2^B$ outputs control the $2^B$ unit DACs. If the BTN's control signals, $c_k(n)$ for $k = 0, 1, \ldots, B-1$, are random bit sequences, the input signal, $x(n)$, activates $x(n)$ unit DACs randomly each sample.

7.3.4.3 Equivalence of generalized cube DEM networks

In this section, the FRDEM network and the BTN are formally shown to be equivalent to an appropriately connected $B$ bit GCN.

*Theorem 2:* The $n$ bit FRDEM network is equivalent to an appropriately connected $n$ bit GCN.

---

**Figure 7.31.** Construction of a $k+1$ bit BTN.
Proof: The switching block, $S_{0,1}$, shown in Figure 7.28 is a one bit FRDEM network and is equivalent to a one bit GCN. To illustrate, when $c_0(n)$ is a logical zero, both networks route $b_1(n)$ to the upper output and $b_0(n)$ to the lower output. When $c_0(n)$ is a logical one, both networks exchange these outputs.

Assume that the $k$ bit FRDEM network and the $k$ bit GCN are equivalent. The $k+1$ bit FRDEM network can be partitioned into two $k$ bit FRDEM networks connected by switching block, $S_{k,1}$, as shown in Figure 7.29. Switching block, $S_{k,1}$, performs the cube$_{k}$ operation. As shown in Figure 7.15, the $k+1$ bit GCN can be partitioned into two $k$ bit GCNs by setting all of the most significant stage interchange switches to operate in unison. When operated in this manner, the first stage of a GCN performs the cube$_{k}$ operation. Therefore, if the $k$ bit FRDEM network and $k$ bit GCN are equivalent, the $k+1$ bit FRDEM network and $k+1$ bit GCN are equivalent. Thus by induction, the $n$ bit FRDEM network and $n$ bit GCN are equivalent for $n \geq 1$.

Theorem 3: An appropriately connected $n$ bit GCN is equivalent to an $n$ bit BTN when the BTN's high impedance output is set to the LSB of the GCN's input.

Proof: In Theorem 2, it was shown that the switching block, $S_{0,1}$, in Figure 7.28, is a one bit FRDEM network and a one bit GCN. Furthermore, it is equivalent to the one bit BTN shown in Figure 7.32 when the BTN's high impedance output is set to the LSB of the GCN's input. To illustrate,
when \( c_0(n) \) is a logical zero, the GCN and the BTN route \( b_1(n) \) to the upper output, the GCN routes \( b_0(n) \) to the lower output, and the BTN generates a high impedance node, which is set to \( b_0(n) \), at the lower output. Similarly, when \( c_0(n) \) is a logical one, both networks exchange these outputs.

Assume that the \( k \) bit GCN and the \( k \) bit BTNs are equivalent when the BTN’s high impedance output is set to the LSB of the GCN’s input. As shown in Figure 7.15, the \( k+1 \) bit GCN can be partitioned into two \( k \) bit GCNs by setting all of the most significant stage interchange switches to operate in unison. If the control signal, \( c_k(n) \), is a logical zero, the input to the upper \( k \) bit GCN is \( k \) copies of the MSB of \( b(n) \), and the input to the lower \( k \) bit GCN is the \( k \) LSBs of \( b(n) \). Therefore, when \( c_k(n) \) is low, the upper \( k \) bit GCN outputs \( 2^k \) copies of the MSB and the lower GCN generates \( 2^k \) outputs from the LSBs. Similarly, these outputs are exchanged when \( c_k(n) \) is a logical one. As illustrated by Figure 7.31, the \( k+1 \) bit BTN functions identically. Therefore, if the \( k \) bit BTN and \( k \) bit GCN are equivalent, the \( k+1 \) bit BTN and \( k+1 \) bit GCN are equivalent. Thus, by induction, the \( n \) bit GCN and the \( n \) bit BTN are equivalent for

\[ x_0(n) \rightarrow \begin{array}{c} c_0(n) \hfill \end{array} \rightarrow g_1(n) \]
\[ \begin{array}{c} c_0(n) \hfill \end{array} \rightarrow g_0(n) \]

Figure 7.32. One bit BTN.
\[ n \geq 1 \text{ when the BTN's high impedance output is set to the LSB of the GCN's input.} \]

7.3.4.4 Comparison of GCN implementations

For a MOS implementation of the networks, binary switches can be implemented using transmission gates. For a MOS implementation of a \( B \) bit DEM DAC, the FRDEM network, the GCN and the BTN require \( 2^{B+3} - 4B - 8, B2^{B+1}, \) and \( 2^{B+1} + 2B - 2B - 2 \) transmission gates, respectively. Because \( B2^{B+1} > 2^{B+3} - 4B - 8 \) when \( B > 2 \) and \( B2^{B+1} > 2^{B+1} + 2B - 2B - 2 \) when \( B > 0 \), both the FRDEM and the BTN implementations require fewer transmission gates than the GCN. When compared to the FRDEM network, the BTN requires fewer transmission gates when \( 2^{B+3} - 4B - 8 < 2^{B+1} + 2B - 2B - 2 \) or \( B < 5.67 \). Table 7.1 compares the hardware requirements of the three equivalent network topologies. The results in Table 7.1 show that the BT implementation requires approximately the same number of transmission gates as the FRDEM implementation for six bits. However, the physical layout of a six bit BTN has more geometric regularity, simpler routing and is more compact than the physical layout of a six bit FRDEM network. Thus, of the three networks, the BTN has the lowest hardware complexity for two to six bit DACs, and the FRDEM network has the lowest hardware complexity for DACs with seven bits or more.

7.3.4.5 VLSI layout considerations

The FRDEM network and BTN\(S \) have a regular structure which is easily adapted to VLSI layout. Figure 7.33 shows the double-metal, N-well CMOS layout of a two bit BTN.
Figure 7.33. VLSI implementation of a two bit BTN.
Table 7.1. Number of transmission gates in construction of the generalized cube DEM network implementations

<table>
<thead>
<tr>
<th>bits</th>
<th>GCN</th>
<th>FRDEM</th>
<th>BTN</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>48</td>
<td>44</td>
<td>32</td>
</tr>
<tr>
<td>4</td>
<td>128</td>
<td>104</td>
<td>86</td>
</tr>
<tr>
<td>5</td>
<td>320</td>
<td>228</td>
<td>212</td>
</tr>
<tr>
<td>6</td>
<td>768</td>
<td>480</td>
<td>498</td>
</tr>
<tr>
<td>7</td>
<td>1792</td>
<td>988</td>
<td>1136</td>
</tr>
<tr>
<td>8</td>
<td>4096</td>
<td>2008</td>
<td>2542</td>
</tr>
</tbody>
</table>

The two bit BTN requires ten transmission gates. To simplify the routing of power, ground and the transmission gate control signals, a transmission gate cell and a complementary transmission gate cell were designed. The former cell acts as a short when its control signal is logical one and an open when its control signal is logical zero. The complementary transmission gate's operation is reversed. The DAC's input data bits are labelled \( x_0[n] \) and \( x_1[n] \) and applied to the metal lines in the upper right portion of Figure 7.33. The network's two control signals, \( c_0[n] \) and \( c_1[n] \), and their respective complementary signals, \(*c_0[n]\) and \(*c_1[n]\), are applied to busses running horizontally across the network. The unlabeled horizontal busses in Figure 7.33 conduct power and ground to the cells. The network's scrambled thermometer coded outputs are labelled at the bottom of Figure 7.33.

The three bit BTN is constructed using two bit BTNs connected by a bank of transmission gates as shown in Figure 7.31. Figure 7.34 shows the double-metal, N-well CMOS layout of a three bit BTN. The four transmission gates in a horizontal row in the middle of the network are the four transmission gates controlled by \( c_2[n] \) and \(*c_2[n]\) on the left side of the diagram in Figure 7.30. The four transmission gates in a vertical column
Figure 7.34. VLSI implementation of a three bit BTN. The time index has been omitted for clarity.
in the upper left corner of the network in Figure 7.34 are the four transmission gates controlled by $c_2[n]$ closest to the unit DAC in Figure 7.30. The four transmission gates in a vertical column in the lower left corner of the network in Figure 7.34 are the four transmission gates controlled by $c_2[n]$ closest to the unit DAC in Figure 7.30. The remainder of the transmission gates in Figure 7.34 represent two copies of the network in Figure 7.33. Because the BTN is recursive, the pattern can be continued to construct larger BTNs. For example, Figure 7.35 shows the double-poly, double-metal, N-well CMOS layout of a five bit BTN. The two-, three- and four bit BTNs contained within the five bit BTN can be seen in the layout. An example of each is highlighted in Figure 7.35.

7.3.5 Barrel shift DEM network

In the $B$ bit DEM DAC architecture in Figure 7.21, the $2^B$ unit DAC control signals are rearranged by a $B$ bit BSN. The $2^B$ bit signal is created by appending a logical zero to the $2^B - 1$ bit thermometer coded representation of the $B$ bit DAC input, $x(n)$. The BSN outputs control the $2^B$ unit DACs. Figure 7.36 shows an example of a three bit BSN DEM DAC. If the BSN's control signals, $c_k(n)$ for $k = 0, 1, ..., 2^B - 1$, are random bit sequences, the input signal, $x(n)$, activates $x(n)$ unit DACs randomly each sample. The BSN implements the $2^B$ circular shifts of its input lines, thereby generating $2^B$ input-output permutations. When the BSN inputs are all zero or all one, only one logical permutation exists. The number of logical permutations for all other network inputs is $2^B$. Therefore, the total number of logical permutations for the BSN is $2^{2B} - 2^B + 2$.

Because the BSN is not recursive, it cannot be constructed using interchange switches. The network is instead constructed using transmission gates for each horizontal and
Figure 7.35. VLSI implementation of a five bit BTN. The recursive structure is demonstrated by labelling three of the smaller BTN\(s\) within.
diagonal edger in the barrel shift graph. Therefore, for a $B$ bit DEM DAC, the BSN requires $B$ control signals and $B2^{B+1}$ transmission gates.

In the generalized cube DEM networks, a number of interchange switches can be eliminated because the GCN is recursive and many of the GCN inputs are copies of the same bit. The BSN is not recursive or constructed with the interchange switch in Figure 7.2. Nonetheless, the BSN has many redundant transmission gates since many of its inputs are copies of the same bit.

If the rows of the $r$ bit BSN in a $r$ bit DEM DAC are labelled with $r$ bit binary numbers, the location of the most significant one denotes which DAC input bit is connected to that row. Consider the three bit case in Figure 7.36. The logical zero is connected to the BSN row labelled $(000)_2$ and $x_1$ is connected to row $(001)_2$. Similarly, $x_2$ is connected to rows $(010)_2$ and $(011)_2$, while $x_3$ drives any row of the form $(1dd)_2$, where $d$ denotes a "don't care" bit.

Figure 7.36. Three bit BSN DEM DAC architecture. The BSN is presented in graph representation for clarity.
If the leading stage's control signal in a $r$ bit BSN is a logical zero, the leading stage does not rearrange the inputs. If the leading stage's control signal in a $r$ bit BSN is a logical one, the leading stage rearranges the inputs by adding $2^{r-1}$ in modulo $2^r$ arithmetic to the line labels. Similarly, the second stage of the $r$ bit BSN in a $r$ bit DEM DAC does not rearrange the inputs or rearranges the inputs by adding $2^{r-2}$ in modulo $2^r$ arithmetic to the line labels if the second stage's control signal is logical zero or logical one, respectively. The pattern is repeated at each stage of the BSN where the last stage does not rearrange the inputs or rearranges the inputs by adding 1 in modulo $2^r$ arithmetic to the line labels if the last stage's control signal is logical zero or logical one, respectively. Therefore, at the $k$th most significant stage of the BSN, the row labels have possibly been changed only in $k$ most significant bits of the row labels. It is easily seen from the initial location of the row labels and the DAC inputs which are connected to each $r$ bit BSN input that $2^{r-2}$ pairs of lines are redundant at the third to last stage. Also, $2^{r-3}$ pairs of lines and $2^{r-3}$ groups of four lines are redundant at the next most significant stage of a BSN. The pattern of redundant lines continue to the most significant stage where two pairs, two groups of four lines, two groups of eight lines, etc. are redundant. Each redundancy at each stage of the BSN can be replaced with only a single pair of transmission gates and the appropriate replication of the signal at a later point in the network. Figure 7.37 shows an example of a three bit hardware efficient BSN DEM DAC.

The number of transmission gates in the hardware efficient $B$ bit barrel shift DEM network is given by

$$2^B+2 + \sum_{i=0}^{B-3} 2^{i+2}(B-i) = 2^B+3 - 4B - 8.$$
This is the same number of transmission gates in the FRDEM network.

7.4 Conclusions

In general, digital DEM algorithms for DACs rearrange the unit DAC inputs in the architecture shown in Figure 7.21. As shown in Chapter 6, the combinatorial capability of the DEM DAC's interconnection network is important in determining the distortion power is created in the DAC's output. Thus, a desirable DEM DAC interconnection network would be capable of rearranging the unit DAC inputs into all possible permutations. The Benes network, in particular, can execute all possible permutations when each of its switches has separate and independent control signals. However, the number of control signals becomes prohibitive as the resolution of the DAC increases. A more practical DEM technique uses the same control signal for all switches common to the same stage of the interconnection network. Fewer control signals are needed, but the network implements far fewer permutations. Because the network's input is a thermometer coded value, many

Figure 7.37. Three bit reduced complexity BSN DEM DAC architecture. The BSN is presented in graph representation for clarity.
permutations are indistinguishable. As demonstrated in [15], [23], [34], [70], and Chapter 6, these less powerful interconnection networks can provide a sufficient number of permutations for DEM in DACs. This chapter has presented a comprehensive survey of the DEM algorithms found in the literature and introduced several new hardware efficient digital DEM algorithms. All of the digital DEM architectures proposed in the literature, with the exception of register-based barrel shifting DEM, were shown to be equivalent to well understood multistage interconnection networks. Properties of these interconnection networks can be used to obtain equivalent networks with minimal hardware complexity. Metrics for comparing the relative combinatorial capability and hardware complexity were provided. A summary of these metrics are presented in Table 7.2. Finally, several networks were examined and considerations for VLSI implementations presented.
Table 7.2. Characteristics of several DAC DEM networks.

<table>
<thead>
<tr>
<th>Network</th>
<th>interchange switches</th>
<th>transmission gates</th>
<th>control signals</th>
<th>I/O permutations</th>
<th>logical permutations</th>
<th>combinatorial power</th>
</tr>
</thead>
<tbody>
<tr>
<td>crossbar</td>
<td>N/A</td>
<td>$2^{B} + 2$</td>
<td>$2^{B}$</td>
<td>$(2^{B})!$</td>
<td>$2^{B}$</td>
<td>1.0</td>
</tr>
<tr>
<td>Benes(^1)</td>
<td>$B2^{B}$</td>
<td>$B2^{B} + 2$</td>
<td>$B2^{B}$</td>
<td>$(2^{B})!$</td>
<td>$2^{B}$</td>
<td>1.0</td>
</tr>
<tr>
<td>IBCN(^2)</td>
<td>$B2^{B} - 1$</td>
<td>$B2^{B} + 1$</td>
<td>$B$</td>
<td>$2^{B}$</td>
<td>$2 + \frac{2(4^{B} - 1)}{3}$</td>
<td>$(2^{B} - 1)!^{-1}$</td>
</tr>
<tr>
<td>IBCN(^3)</td>
<td>$2^{B} - 1$</td>
<td>$2^{B} + 2 - 4$</td>
<td>$B$</td>
<td>$2^{B}$</td>
<td>$2 + \frac{2(4^{B} - 1)}{3}$</td>
<td>$(2^{B} - 1)!^{-1}$</td>
</tr>
<tr>
<td>GCN(^2)</td>
<td>$B2^{B} - 1$</td>
<td>$B2^{B} + 1$</td>
<td>$B$</td>
<td>$2^{B}$</td>
<td>$2 + \frac{2(4^{B} - 1)}{3}$</td>
<td>$(2^{B} - 1)!^{-1}$</td>
</tr>
<tr>
<td>FRDEM</td>
<td>$2^{B+1} - B - 2$</td>
<td>$2^{B+3} - 4B - 8$</td>
<td>$B$</td>
<td>$2^{B}$</td>
<td>$2 + \frac{2(4^{B} - 1)}{3}$</td>
<td>$(2^{B} - 1)!^{-1}$</td>
</tr>
<tr>
<td>BTN</td>
<td>N/A</td>
<td>$2^{B+1} + B2^{B} - 2B - 2$</td>
<td>$B$</td>
<td>$2^{B}$</td>
<td>$2 + \frac{2(4^{B} - 1)}{3}$</td>
<td>$(2^{B} - 1)!^{-1}$</td>
</tr>
<tr>
<td>BSN</td>
<td>N/A</td>
<td>$B2^{B} + 1$</td>
<td>$B$</td>
<td>$2^{B}$</td>
<td>$2^{2B} - 2^{B} + 2$</td>
<td>$(2^{B} - 1)!^{-1}$</td>
</tr>
<tr>
<td>BSN(^3)</td>
<td>N/A</td>
<td>$2^{B+3} - 4B - 8$</td>
<td>$B$</td>
<td>$2^{B}$</td>
<td>$2^{2B} - 2^{B} + 2$</td>
<td>$(2^{B} - 1)!^{-1}$</td>
</tr>
</tbody>
</table>

1. Each interchange switch is operated independently.
2. All interchange switches in a stage are operated in unison.
3. Hardware efficient implementation.
CHAPTER 8

CONCLUSIONS

Five important contributions have been made in this dissertation. First, the effects of a non ideal ADC’s transformation on its output have been estimated. Second, a generalized analysis of flash DEM ADCs has been developed and several ADC DEM architectures analyzed. Third, the relationship between a DAC’s INL and its output spectrum has been determined. Fourth, a generalized analysis of DEM techniques for DACs has been derived. Finally, several DEM circuits proposed in the literature have been shown to be equivalent and have hardware efficient VLSI implementations.

In this dissertation, three methods that estimate the effects of a non ideal ADC’s transformation on its output have been developed. The first method approximates an ADC’s transfer function by a symmetric or asymmetric power function. The power function approximation is not always valid and thereby limits the method’s usefulness. The second method uses the infinite Fourier series expansion of the ADC’s transfer function to estimate the ADC’s output spectrum. Truncation of the infinite Fourier series introduces appreciable artifacts and the analysis is only useful for purely sinusoidal ADC inputs. The third method estimates the ADC’s output spectrum by a sampled ADC transfer function, and can be made more accurate by sampling the transfer function more finely. Also, the analyses can be used to estimate standard ADC performance metrics, such as SDR, SNDR, and SFDR, as a function of the ADC’s transfer function approximation.
A generalized analysis of the quantization threshold levels in DEM flash ADCs, including determination of several suitable metrics by which DEM techniques can be compared, has been derived. The distortion power resulting from mismatched components for ADCs using DEM techniques has been determined. Typically, the distortion power due to mismatched components is small compared to quantization error power. However, the distortion is concentrated in a few frequencies unless a stochastic DEM techniques is used. Several DEM voltage division and current steering ADC architectures were analyzed, simulated, and shown capable of removing all harmonic distortion when stochastically controlled. Furthermore, an example of shaping the distortion due to mismatched components was presented. Currently, DEM network control signals which shape the distortion due to mismatched components must be determined experimentally. An analytical method of determining a suitable control signal to shape distortion due to mismatched components remains an open research topic.

The relationship between a DAC's INL and its output spectrum has been found. A DAC's output spectrum can be found in response to an arbitrary periodic or finite input. Standard DAC performance metrics, such as SDR, SNDR and SFDR, were determined as a function of DAC's transfer function. Furthermore, an iterative method was presented to determine a DAC's transfer function from its observed output magnitude spectrum.

A generalized analysis of DEM techniques for DACs has been derived, including the determination of several suitable metrics by which DEM techniques can be compared. The analysis relates the performance of a given DEM technique to standard DAC performance metrics, such as SDR, SNDR and SFDR. The conditions on the DEM technique which guarantee that a DAC has zero average INL were determined. Using these DEM
techniques with a zero mean, uniformly distributed, independent white control signal was shown to render the distortion due to mismatched components as white noise. Furthermore, the distortion due to mismatched components can be shaped using DEM techniques and a non white stochastic control signal. Additional research is needed to adequately understand how the DEM network control signal directs the DEM technique to shape the distortion due to mismatched components.

Several DEM circuits proposed in the literature have been shown to be equivalent and have hardware efficient implementations based on multistage interconnection networks. Many more DEM circuits exists which satisfactorily permute the circuit elements and remain to be discovered. The design of the DEM circuit with minimal hardware complexity with all of the required properties determined in this dissertation remains an open question.
EXAMPLE DEM DAC IMPLEMENTATIONS

This appendix describes the implementation of three dynamic element matching DAC fabricated in American Microsystems, Inc.'s (AMI's) N-well, double-poly, double-metal 1.2μm bulk CMOS process. The first DAC is a six bit current steering DEM DAC that uses the binary tree network described in Section 7.3.4.2. The second DAC is a five bit switched capacitor DEM DAC that uses the binary tree network described in Section 7.3.4.2. The third DAC is a six bit current steering DEM DAC that uses the barrel shift network described in Section 7.3.5.

A.1 Example DAC One

A six bit differential current steering DEM DAC that uses the binary tree (BT) network was fabricated using AMI's N-well, double-metal 1.2μm CMOS process. The BT network's control signals are generated off chip and latched in six static registers. The DAC's input data is also latched by six static registers. The BT network output is the thermometer code bits reordered according to the externally applied control data. The BT network's 64 bit output is latched by array of static logic bit registers. These registers drive control analog switches that steering the unit DAC's current to summing nodes. An unit DAC's output current is directed to one output pin if its corresponding thermometer bit is logical one or directed to another output pin if the thermometer bit is logical zero. The unit

239

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
DACs' currents are generated by an array of current mirrors based on a regulated cascode current sink. The regulated cascode current sink circuit schematic is shown in Figure A.1. The regulated cascode current mirror has a very large output impedance and can maintain a constant output current, $I_o$, even if $V_o$ is small [7], [62]. The regulated cascode current sink is ideally suited for use in current steering DACs since it allows the DAC output voltage swing to be nearly rail-to-rail.

Figure A.2 shows Example One's floorplan. Table A.1 lists the major DAC subsystems and Figure A.3 shows the DAC's layout in a 1.2μm N-well, double-metal CMOS process. The total die size is approximately 2mm by 2mm.

A.2 Example DAC Two

A five bit differential switched capacitor DEM DAC that uses the BT network was fabricated using AMI's N-well, double-poly, double-metal 1.2μm CMOS process. The BT network's control signals are generated off chip and latched in five static registers. A

![Bias circuit and current mirror](image-url)  

Figure A.1. Regulated cascode current sink.
Table A.1. Example One’s major subsystems

<table>
<thead>
<tr>
<th>Subsystem</th>
<th>Marker</th>
</tr>
</thead>
<tbody>
<tr>
<td>input latches</td>
<td>A</td>
</tr>
<tr>
<td>clock generation and drivers</td>
<td>B</td>
</tr>
<tr>
<td>pad frame</td>
<td>C</td>
</tr>
<tr>
<td>six bit BT network</td>
<td>D</td>
</tr>
<tr>
<td>BT network output latches</td>
<td>E</td>
</tr>
<tr>
<td>unit DAC bias circuits</td>
<td>F</td>
</tr>
<tr>
<td>unit DAC arrays (current mirrors)</td>
<td>G</td>
</tr>
</tbody>
</table>

Figure A.2. Example One’s floorplan. Analog subsystems are shaded and digital subsystems are not.
Figure A.3. Six bit current steering DEM DAC using BT network.
five bit pseudorandom number is generated on chip with five linear feedback shift registers (LFSRs). The LFSRs have mutually prime periods. The LFSRs were constructed with dynamic logic shift registers to reduce layout area. A five line MUX allows an external line to select between the on chip pseudorandom control signal or the externally generated control signal. The DAC's input data is latched by five static registers. The BT network output is the thermometer code bits reordered according to the control data. The BT network’s 32 bit output is latched by array of static logic bit registers. These registers drive control analog switches that steering the unit DAC’s current to summing nodes. An unit DAC’s output current is directed to one output pin if its corresponding thermometer bit is logical one or directed to another output pin if the thermometer bit is logical zero. The unit DACs’ currents are generated by an array of current mirrors based on a regulated cascode current sink.

Figure A.5 shows Example Two’s floorplan. Table A.2 lists the major DAC subsystems, and Figure A.6 shows the DAC’s layout in a a 1.2μm N-well, double-poly, double-metal CMOS process. The total die size is approximately 2mm by 2mm.

Figure A.4. Regulated cascode current sink.
A.3 Example DAC Three

A six bit differential current steering DEM DAC that uses the barrel shift network was fabricated using AMI's N-well, double-metal 1.2μm CMOS process. The barrel shift network's control signals are generated off chip and latched in six static registers. A six bit pseudorandom number is generated on chip with six linear feedback shift registers (LFSRs). The LFSRs have mutually prime periods. The LFSRs were constructed with dynamic logic shift registers to reduce layout area. A six line MUX allows an external line

Figure A.5. Example Two's floorplan. Analog subsystems are shaded and digital subsystems are not.
Figure A.6. Five bit switched capacitor DEM DAC using BT network.
Table A.2. Example Two’s major subsystems

<table>
<thead>
<tr>
<th>Subsystem</th>
<th>Marker</th>
</tr>
</thead>
<tbody>
<tr>
<td>input latches</td>
<td>A</td>
</tr>
<tr>
<td>clock generation and drivers</td>
<td>B</td>
</tr>
<tr>
<td>pad frame</td>
<td>C</td>
</tr>
<tr>
<td>five bit LFSR (pseudorandom number generator)</td>
<td>D</td>
</tr>
<tr>
<td>MUX</td>
<td>E</td>
</tr>
<tr>
<td>five bit BT network</td>
<td>F</td>
</tr>
<tr>
<td>BT network output latches</td>
<td>G</td>
</tr>
<tr>
<td>capacitor array switches</td>
<td>H</td>
</tr>
<tr>
<td>differential output opamp</td>
<td>I</td>
</tr>
<tr>
<td>unit capacitor arrays</td>
<td>J</td>
</tr>
<tr>
<td>voltage divider</td>
<td>K</td>
</tr>
<tr>
<td>opamp bias voltage follower</td>
<td>L</td>
</tr>
<tr>
<td>differential opamp (test structure)</td>
<td>M</td>
</tr>
</tbody>
</table>

to select between the on chip pseudorandom control signal or the externally generated control signal. The DAC’s input data is latched by six static registers. The barrel shift network output is a set of thermometer coded bits reordered according to the control data. The barrel shift network’s 64 bit output is latched by an array of static registers. These registers control analog switches that steer the unit DAC’s current to a summing node. An unit DAC’s output current is directed to one output pin if its corresponding thermometer bit is logical one or directed to another output pin if the thermometer bit is logical zero.

The unit DAC currents are generated by an array of current mirrors based on a regulated cascode current sink. Figure A.1 shows schematic of the regulated cascode current sink circuit.
Figure A.7 shows Example Three’s floorplan. Table A.3 lists the major DAC subsystems, and Figure A.8 shows the DAC’s layout is shown in a 1.2μm N-well, double-metal CMOS process. The total die size is approximately 2mm by 2mm.

Figure A.7. Example Three’s floorplan. Analog subsystems are shaded and digital subsystems are not.
Figure A.8. Six bit current steering DEM DAC using barrel shift network.
Table A.3. Example Three’s major subsystems

<table>
<thead>
<tr>
<th>Subsystem</th>
<th>Marker</th>
</tr>
</thead>
<tbody>
<tr>
<td>input latches</td>
<td>A</td>
</tr>
<tr>
<td>clock generation and drivers</td>
<td>B</td>
</tr>
<tr>
<td>pad frame</td>
<td>C</td>
</tr>
<tr>
<td>six bit LFSR (pseudorandom number generator)</td>
<td>D</td>
</tr>
<tr>
<td>MUX</td>
<td>E</td>
</tr>
<tr>
<td>six bit barrel shift network</td>
<td>F</td>
</tr>
<tr>
<td>barrel shift network output latches</td>
<td>G</td>
</tr>
<tr>
<td>unit DAC bias circuit</td>
<td>H</td>
</tr>
<tr>
<td>unit DAC arrays (current mirrors)</td>
<td>I</td>
</tr>
</tbody>
</table>
REFERENCES


Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
VITA

Graduate College
University of Nevada Las Vegas

Jerry Wayne Bruce, II

Address:
2050 W. Warm Springs Road, Unit #3323
Henderson, NV 89014

Degrees:
Bachelor of Science in Engineering, 1991
University of Alabama in Huntsville

Master of Science in Electrical Engineering, 1993
Georgia Institute of Technology

Special Honors and Awards:
Audio Engineering Society Graduate Fellowship, 1997-1999

Publications:


256


Dissertation Title:
Dynamic Element Matching Techniques for Data Converters

Dissertation Examination Committee:
Chairperson, Peter Stubberud, Ph.D.
Committee Member, Rahim Khoie, Ph.D.
Committee Member, Eugene McGaugh, Ph.D.
Committee Member, Douglas Reynolds, Ph.D.
Graduate Faculty Representative, Evangelos Yfantis, Ph.D.