Digital Self-Calibration Techniques for High-Accuracy, High-Speed Analog-to-Digital Converters

bу

Andrew Nicholas Karanicolas

Bachelor of Science Electrical Engineering and Computer Science Massachusetts Institute of Technology (1987)

Master of Science Electrical Engineering and Computer Science Massachusetts Institute of Technology (1990)

Submitted to the Department of Electrical Engineering and Computer Science In Partial Fulfillment of the Requirements For the Degree of

Doctor of Philosophy

at the

Massachusetts Institute of Technology

February 1994

©Massachusetts Institute of Technology
All Rights Reserved

| Signature of Author | <u> </u> |
|---------------------|---|
| | Department of Electrical Engineering and Computer Science |
| | 16 February 1994 |
| Certified By | |
| | Hae-Seung Lee |
| | Professor of Electrical Engineering and Computer Science |
| | Thesis Supervisor |
| Accepted By | |
| | Frederic R. Morgenthaler |
| | Chairman, Department Committee on Graduate Students |

ARCHIVES
MASSACHUSETTS INSTITUTE
OF TECHNICIONS

JUL 13 1994

Digital Self-Calibration Techniques for High-Accuracy, High-Speed Analog-to-Digital Converters

bу

Andrew Nicholas Karanicolas

Submitted to the
Department of Fiectrical Engineering and Computer Science
on 16 February 1994 in partial fulfillment of the requirements
for the Degree of Doctor of Philosophy

Abstract

This thesis presents digital self-calibration techniques for high-accuracy, high-speed Analogto-Digital Converters (ADCs). The digital self-calibration addresses the traditional limitations to pipeline ADCs implemented in monolithic IC processes. The digital self-calibration reported here automatically accounts for capacitor mismatch, capacitor nonlinearity leading to Differential Nonlinearity (DNL), charge injection, finite op-amp gain and comparator offset. The errors are directly measured under the same conditions as during the run mode. Thus, DNL is not contributed by circuit nonlinearity. No external reference data converters are needed in the technique and no high-precision matched components are required. Since the calibration algorithm is all digital, no extra analog hardware is required for the calibration mode or the run mode. The digital self-calibration can attain 15-bits of linearity with component matching no better than 7-bits. The digital self-calibration performs a simple calibration algorithm in the digital domain alone. As a result of the algorithm simplicity, no multipliers, microprocessors or other complex arithmetic systems are required. The digital calibration presented here may be applied to pipeline or cyclic ADC architectures employing a 1-bit-per-stage or multi-bit-per-stage design. An important advantage of a pipeline architecture is that it exhibits linear hardware growth, as a function of resolution, compared to exponential hardware growth for flash ADC architectures. Successive approximation and delta-sigma oversampled ADCs have lower throughput than a pipeline ADC because of an increased number of clock cycles needed for conversion. Additionally, pipeline ADCs can acquire Sample-and-Hold Amplifier (SHA) data or multiplexed SHA data, unlike deltasigma oversampled ADCs. A 15-bit 1 MS/s switched-capacitor pipeline ADC prototype is investigated. The 1-bit-per-stage, radix 1.93 design incorporates digital self-calibration. Total Harmonic Distortion (THD) was measured as -90 dB up to a 100 kHz sine-wave input frequency at a 1 MS/s sampling rate. Maximum DNL of ± 0.25 LSB at 15-bits and maximum Integral Nonlinearity (INL) of ± 1.25 LSB at 15-bits were measured at 1 MS/s. The chip was fabricated in an 11 V, 4 GHz, 2.4 μ m BiCMOS process.

Thesis Supervisor: Prof. Hae-Seung Lee

Title: Professor of Electrical Engineering and Computer Science

Acknowledgements

Upon reaching the destination of a fantastic journey, I would like to thank the incredibly diverse set of people who have been part of my endeavors at M.I.T. Having shared with them in the intense, cooperative and ultimately rewarding experiences at M.I.T. has taught all of us how to excel. Having helped each other through some difficult times has taught all of us how to accept each other and enjoy the wonderful times.

First, I want extend thanks to my thesis advisor, Prof. Hae-Seung Lee. His technical contributions have been invaluable in this work and it has been greatly rewarding to be part of his outstanding team in the Microsystems Technology Laboratory. His belief in me, and in my project, was instrumental to sustaining my drive in this project. It was an immense pleasure to have shared with him the excitement from the success of this work.

I am tremendously grateful to my thesis readers. Prof. Charles Sodini provided me important feedback during the evolution of my doctoral thesis. His advice in practical matters is greatly appreciated. As a teacher, and a professional role model, Prof. James Roberge was instrumental in my decision to pursue a doctoral thesis in the field of analog circuits.

There are many people at Harris Semiconductor who played a key role in the implementation of the chip in this work. Kanti Bacrania provided me the opportunity to design and layout the chip at Harris. His commitment to having the chips fabricated at Harris is greatly appreciated. Greg Fisher was helpful in explaining many aspects of the Fastrack design system. Thanks go to Akira Ito, who responded to my almost endless stream of process related questions. Mark Snowden was helpful in keeping the computer systems up and running. Thanks go to Deborah Michaud for helping me get Versatec plots. Jeanne Santell provided good conversation during her rounds as night watchman in the many early morning hours in Bldg. E11. Thanks also go to Dale Edwards for helping provide a great intern program at Harris.

Many people at Analog Devices were very helpful, and friendly, in arranging access to the P-CAD PCB layout tools. In particular, thanks go to Dave Robertson for providing me initial direction among the various PCB approaches. Peter Casparius was helpful, and patient, in explaining how to use the many features of the system necessary to complete the PCB. Thanks go to Stacy Ho and Hassan Malik for fun conversation. Additional thanks to Stacy for rides to Analog Devices on many cold, wet New England mornings. Mick Mueck was helpful in providing feedback on the PCB layout. I am grateful to Chris Mangelsdorf for informative conversations about ADCs and Analog Devices.

The students at MTL provided a great atmosphere to work in and were a lot of fun to be around. The many discussions with Kenneth O and Merit Hong spanned just about every topic possible. Many thanks go to Joe Lutsky, Shujaat Nadeem, Steve Decker and Jen Lloyd for the fun times. Additional thanks to Joe Lutsky for giving me rides home during terrible weather. Special thanks to Kathleen Early for sharing the TA experience and for being a fantastic friend. I enjoyed many conversations with Barry Thompson about audio, business and west coast life. Thanks go to David Martin and Paul Yu for interesting

conversations on a myriad of topics. Joe Kung and Chris Eugster provided a source of many colorful arguments about academics, business and music. Special thanks to Fritz Herrmann and Jeff Gealow for their outstanding computer expertise. Their efforts have been central to keeping our computers in both Harry's and Charlie's group up and running. Ken Szajda has been an interesting source of technical discussions, has provided a wide variance of humor and has been a great guy to talk to from the very beginning in this long path at M.I.T. John Bulzacchelli helped with his grasp on electrical engineering, and his patience in explaining some of it to me, as well as being a great friend. Special thanks to Tom Lee for many great conversations about technical matters and life, for great times as TAs up in the 5th floor lab of Bldg. 38 and for his excellence in teaching me about analog circuits. Many thanks to Craig Keast, and his wife Barbara, for providing a western outpost for many of my century bike rides that passed through Groton, Massachusetts, and to their Basset Hound, Burt, who no longer attacks me.

I want to thank Prof. Art Smith, Dean of Student Affairs, for giving me advice about a wide variety of issues that arise throughout a graduate student's life at M.I.T. Many thanks are due to Marilyn Pierce, Monica Bell, Alice Twohig and Lisa Bella in the EECS Graduate Administration offices who were always pleasant when asked a myriad of administrative questions. Special thanks go to Marilyn Pierce who was especially patient in explaining administrative deadlines, even when I forgot about them. Many thanks to Beth McDonald for always being cheerful to talk to. Special thanks to Kate Paterson for her excellent work in handling the SRC reviews.

Finally, I want to express my gratitude to some friends from my undergraduate years that I still stay in contact with at M.I.T. and across the U.S.A. Ivan Deutsch has been a great friend throughout my stay at M.I.T. and I thank him for listening to me when I was down, always remembering my birthday and being a lot of fun to do stuff with. Isako Hoshino has provided great conversation about everything possible and has been a lot of fun to be with. I would like to thank Bradford Tropea for an infinite source of great conversation and good times going all the way back to Atlantic City High School days. Many thanks to Anand Mehta for discussing the glories and trials of being undergraduate and graduate students at M.I.T. and for lots of good times, pizza and rides to movies. Last, but not least, special thanks go to George Xixis, Ken Ishii, Sheryl Shanks, Steve Rohall, Meilin Wong (Teela), John Rochlis, Chun The, Annette Leonard (Nemesis), Amy Lee, Carla Fermann (I'm Not Loud!), Paul Watts (Photon), Livia Jacunski, Karen Fu, Bill Gimbel and Andy Berlin.

This research was supported by the Semiconductor Research Corporation (91-SP-080) and by General Electric and Analog Devices. The prototype pipeline ADC was fabricated by Harris Semiconductor. MOSIS provided fabrication for the PCB used in the ADC evaluation. Some of the computer simulations were performed using the CRAY XMP resources at the M.I.T. Supercomputer Facility.

Dedication

This work would not have been possible without the continuous love and support from my parents, Nicholas and Zoi Karanicolas, and from my sister, Anna Maria. Their hard work and devotion are the source of my inspiration. I owe all of my success to them.

Biography

Andrew N. Karanicolas was born in Philadelphia, Pennsylvania, on July 28, 1965. He attended George Washington High School from 1979-1980 in Philadelphia. Upon moving to Atlantic City, New Jersey, he attended Atlantic City High School from 1980-1983. He received the S.B. degree from the Dept. of Electrical Engineering and Computer Science from the Massachusetts Institute of Technology, Cambridge, Massachusetts, in June 1987. The S.B. thesis was titled Feedback Control of Plasma Radius in the Versator Tokamak. From September 1987 to June 1988, he was a Teaching Assistant in the Dept. of EECS at M.I.T. for the undergraduate classes Electronic Devices and Circuits and Signals and Sustems. From June 1988 until February 1994, he has been a Research Assistant at the M.I.T. Microsystems Technology Laboratory under the direction of Prof. Hae-Seung Lee. He received the S.M. degree from the Dept. of EECS in September 1990. The S.M. thesis was titled A Switched-Capacitor Pipelined BiCMOS Analog-to-Digital Converter. He was a research intern at Harris Semiconductor, Melbourne, Florida, from July 1991 to December 1991. His present research interests include high-performance BiCMOS, CMOS and bipolar analog integrated circuits and systems. Since 1991, he has been a member of Sigma Xi and since 1985, a student member of the IEEE. He received the Schmeidler Foundation Scholarship from 1983-1987, the Papines Foundation Scholarship from 1983-1987, and the Rotary Club Scholarship during 1983.

Contents

| 1 | Intr | oducti | on | 16 |
|---|------|--------|---|----|
| | 1.1 | Descri | ption of the Problem | 16 |
| | | 1.1.1 | Overview | 16 |
| | | 1.1.2 | Pipeline ADC Architecture | 16 |
| | | 1.1.3 | Limitations to Switched-Capacitor ADCs | 17 |
| | | 1.1.4 | Digital Self-Calibration for Pipeline ADCs | 17 |
| | | 1.1.5 | Digital Error-Correction and Digital Self-Calibration | 18 |
| | 1.2 | Applic | eation Realms | 18 |
| | | 1.2.1 | General High-Accuracy High-Speed Data Acquisition | 18 |
| | | 1.2.2 | Transducer Based Signal Processing Systems | 19 |
| | | 1.2.3 | Communication Systems | 20 |
| | | 1.2.4 | Control Systems | 20 |
| | | 1.2.5 | Ranging Systems | 20 |
| | 1.3 | Organ | ization of the Thesis | 21 |
| 2 | Bac | kgrour | ad | 22 |
| | 2.1 | _ | ary of ADC Techniques | 22 |
| | | 2.1.1 | Flash ADCs | 22 |
| | | 2.1.2 | Subranging ADCs | 24 |
| | | 2.1.3 | Pipeline ADCs | 24 |
| | | 2.1.4 | Successive Approximation ADCs | 27 |
| | | 2.1.5 | Delta-Sigma Oversampled ADCs | 29 |
| | | 2.1.6 | Cyclic ADCs | 29 |
| | 2.2 | Previo | ous Studies in Pipeline ADC Calibration Techniques | 31 |
| | | 2.2.1 | A 12 bit 600 kS/s Digitally Self-Calibrated Pipelined Algorithmic ADC | 31 |
| | | 2.2.2 | Digital-Domain Calibration of Multistep Analog-to-Digital Converters | 32 |
| | | 2.2.3 | A 13 bit 2.5MHz Self-Calibrated Pipelined A/D Converter in 3μm | |
| | | | CMOS | 32 |
| | | 2.2.4 | Self-Calibrating Pipelined Subranging Analog-to-Digital Converter . | 33 |
| | | 2.2.5 | Architecture for High Sampling Rate, High Resolution Analog-to- | |
| | | | Digital Converter System | 34 |

8 CONTENTS

| | | 2.2.6 Digital Error Correction System for Subranging Analog-to-Digital Converters |
|---|------|--|
| | 2.3 | Converters |
| | 2.0 | Advantages of the Proposed Digital Bell Cambration |
| 3 | Pipe | eline ADC Architecture 39 |
| | 3.1 | General Pipeline Subranging Structure |
| | 3.2 | Radix 2 1-Bit-per-Stage Pipeline Subranging ADC Algorithm |
| | 3.3 | Radix 2 1-Bit-per-Stage Pipeline ADC Error Sources 47 |
| | 3.4 | Radix < 2 1-Bit-per-Stage Pipeline ADC Motivation |
| | 3.5 | Radix < 2 Pipeline ADC Properties |
| 4 | D:a | tal Self-Calibration 51 |
| 4 | 4.1 | Introduction |
| | 4.1 | Basic ADC Metrics |
| | 4.2 | - 111 - 1 - |
| | | |
| | | |
| | | 4.2.3 Differential Nonlinearity |
| | | 4.2.4 Integral Nonlinearity |
| | | 4.2.5 Examples of DNL and INL 54 |
| | 4.3 | Principles of Digital Self-Calibration for Pipeline ADCs |
| | | 4.3.1 Introduction |
| | | 4.3.2 MX2 Stage with $G=2$ |
| | | 4.3.3 MX2 Stage with $G > 2$ |
| | | 4.3.4 MX2 Stage with $G < 2$ |
| | | 4.3.5 MX2 Stage with $G < 2$ and Digital Calibration Applied 60 |
| | | 4.3.6 Interpretation of the Last K Stage Pipeline Section 60 |
| | 4.4 | Calibration Algorithm |
| | | 4.4.1 Introduction |
| | | 4.4.2 Derivation of Algorithm |
| | | 4.4.3 Algorithm |
| | 4.5 | Comparator Offset |
| | 4.6 | Pipeline Synthesis |
| | 4.7 | Determination of the Calibration Constants S_1 and S_2 |
| | 4.8 | Extension to Multi-Bit-per-Stage Pipeline ADCs |
| | 4.9 | ADC Digital Output Reduction |
| | 4.10 | Properties of Digital Self-Calibration |
| | D:-: | tal Self-Calibration Simulator 77 |
| 5 | _ | |
| | 5.1 | Software Strategy |
| | 5.2 | Simulator Structure |
| | 5.3 | Simulator Description |
| | 5.4 | Simulation Results |
| | | 5.4.1 MX2 Stage Gain $G = 1.9704$ |

| CONTENTS | 9 |
|----------|---|
| | |

| | | 5.4.2 | MX2 Stage Gain $G = 1.9550$ | | | | | | | | | , . | | | | | | | 79 |
|---|-----|--------|-------------------------------|---|---|---|---|---|-------|-------|--|------------|---|---|---|---|---|---|-----|
| | | 5.4.3 | MX2 Stage Gain $G = 1.9324$ | • | | • | | • | | • | | , • | • | • | • | • | • | • | 79 |
| 6 | Swi | ched- | Capacitor System Design | | | | | | | | | | | | | | | | 86 |
| | 6.1 | SC Pi | peline ADC | | | | | | | | | | | | | | | | 86 |
| | | 6.1.1 | Basic Structure | • | | | | | | • | | | | | | | | • | 86 |
| | | 6.1.2 | Timing | | | | | | | | | | | | | | | | 88 |
| | 6.2 | SC SH | [A | | | | | | | | | | | | | | | | 88 |
| | | 6.2.1 | Sample Mode | | | | | | | | | | | | | | | | 88 |
| | | 6.2.2 | Amplify Mode | | | | | | | | | | | | • | | | • | 88 |
| | 6.3 | SC M | X2 | | | | | | | | | | | | | | | | 91 |
| | | 6.3.1 | Sample Mode | | | | | | | | | | | | | | | | 91 |
| | | 6.3.2 | Amplify Mode | | | | | | | | | | | | | | | | 91 |
| | 6.4 | Limita | ations to the SC SHA and MX2 | | | | | | | | | | | | | | | • | 95 |
| | | 6.4.1 | Nonlinearity | | | | | | | | | | | | | | | | 95 |
| | | 6.4.2 | Residual Offset | | | | | | | | | | | | | | | | 98 |
| | | 6.4.3 | Noise | | | | | | | | | | | | | | | | 99 |
| | | 6.4.4 | Settling Time and Accuracy . | | | | | | | | | | | | | | | | 100 |
| | | 6.4.5 | Aperture Jitter | | | | | | | | | | | | | | | | 100 |
| | 6.5 | SC Co | omparator | | | | | | | | | | | | | | | | 101 |
| | | 6.5.1 | Basic Structure | | • | | | | | | | | | | | | | | 101 |
| | | 6.5.2 | Reset Mode | | | | | | | | | | | | | | | | 101 |
| | | 6.5.3 | Amplify Mode | | | | | | | | | | | | | | | | 101 |
| | | 6.5.4 | Latch | | | | | | | | | | | | | | | | 101 |
| | 6.6 | Limita | ations to the SC Comparator . | | | | | | | | | | | | | | | | 104 |
| | | 6.6.1 | Residual Offset | | | | | | | | | | | | | | | | 104 |
| | | 6.6.2 | Noise | | • | | • | | • | | | | | | | • | • | • | 104 |
| 7 | Ana | log In | tegrated Circuit Design | | | | | | | | | | | | | | | | 105 |
| | 7.1 | Opera | tional Amplifier | | | | | | | | | | | | | | | | 105 |
| | | 7.1.1 | Op-Amp Overview | | | | | | | | | | | | | | | | 105 |
| | | 7.1.2 | Frequency Response Issues . | | | | | | | | | | | | | | | | 105 |
| | | 7.1.3 | Transient Response Issues | | | | | | | | | • • | | | | | | | 108 |
| | | 7.1.4 | Common Mode Feedback Netv | | | | | | | | | | | | | | | | 109 |
| | 7.2 | Comp | arator Pre-Amplifier | | | | | | | | | | | | | | | | 109 |
| | 7.3 | _ | nic CMOS Latch | | | | | | | | | | | | | | | | 111 |
| | 7.4 | • | Distribution Network | | | | | | | | | | | | | | | | 111 |
| | 7.5 | | l Output Buffer | | | | | | | | | | | | | | | | 115 |
| | 7.6 | _ | ation Results | | | | | | | | | | | | | | | | 115 |

10 CONTENTS

| 8 | Mix | ed-Signal System Layout Design | 118 |
|----|------|---|-----|
| | 8.1 | Overall Layout Strategy | 118 |
| | 8.2 | SHA and MX2 Layout | 119 |
| | 8.3 | ADC Layout | 119 |
| | 8.4 | Layout Verification | 119 |
| 9 | Test | System | 123 |
| | 9.1 | Overall Test System Strategy | 123 |
| | 9.2 | Test System Structure | 123 |
| | 9.3 | PCB | 125 |
| | | 9.3.1 Overall Design | 125 |
| | | 9.3.2 Ground Plane Design | 127 |
| | 9.4 | Analog Inputs | 127 |
| | 9.5 | Pipeline Control Clocks | 127 |
| | 9.6 | Calibration Control Clocks | 130 |
| | 9.7 | Reference Voltage Circuit | 130 |
| | 9.8 | External Bias Circuit | 132 |
| | 9.9 | Analog and Digital Power Supplies | 132 |
| 10 | Exp | erimental Results | 135 |
| | - | Total Harmonic Distortion | 135 |
| | | 10.1.1 Overview | 135 |
| | | 10.1.2 Sine-Wave Input, 9.87560 kHz, -2 dB FS | 135 |
| | | 10.1.3 Sine-Wave Input, 98.7560 kHz, -2 dB FS | 136 |
| | | 10.1.4 Sine-Wave Input, 198.7560 kHz, -2 dB FS | 136 |
| | | 10.1.5 Sine-Wave Generator Distortion Test | 136 |
| | 10.2 | DNL and INL | 136 |
| | | Input Referred Noise | 141 |
| | | Dielectric Relaxation | 141 |
| | | Ambient Temperature | 141 |
| | | Power Dissipation and Die Area | 144 |
| 11 | Con | clusions | 146 |
| | | Thesis Summary | 146 |
| | | Future Work | 145 |
| A | Swi | tched-Capacitor Thermal Noise Analysis | 143 |
| | A.1 | | 148 |
| | A.2 | Direct and Sample-and-Held Thermal Noise | 149 |
| | A.3 | | 149 |
| | A.4 | | 149 |
| | 41.7 | A.4.1 Power Spectrum Density of a Discrete Time Stochastic Signal | 149 |
| | | A.4.2 Undersampled Broad-Band Noise FT | |

| CONTENTS | 11 |
|----------|----|
| | |

| | | A.4.3 Enumeration of Sample-and-Held Noise in SC Networks | 155 |
|--------------|------------|---|-----|
| | A.5 | SHA Sample-and-Held Noise Analysis | 156 |
| | | A.5.1 Nonsampling Transmission Gate Contribution | 156 |
| | | A.5.2 Op-Amp Contribution | 156 |
| | | A.5.3 Reset Switch Contribution | 156 |
| | | A.5.4 Total Output Referred Sample-and-Held Noise | 156 |
| | A.6 | MX2 Sample-and-Held Noise Analysis | 157 |
| | A.7 | Pipeline ADC Sample-and-Held Noise Analysis | 158 |
| | A.8 | Signal-to-Noise Ratio | 158 |
| | | | |
| \mathbf{B} | Pip | eline ADC Package | 159 |
| | B.1 | Thermal Resistance | 159 |
| | B.2 | Wire-bond Connection Diagram | 159 |
| | B.3 | | 159 |
| | B.4 | PGA Pin-out and Signal Table | 162 |
| C | Dig | ital Self-Calibration Simulator C Code | 166 |
| | _ | header.h | 166 |
| | C.2 | main.c | 166 |
| | C.3 | setbit.c | 202 |
| | | testbit.c | |
| | _ | | 004 |
| D | | t System C Code | 204 |
| | | acq.c | |
| | D.2 | cal.c | 209 |

List of Figures

| 2.1 | Flash ADC architecture |
|------|---|
| 2.2 | One stage of a subranging ADC architecture |
| 2.3 | Pipeline ADC architecture |
| 2.4 | Successive approximation ADC architecture |
| 2.5 | Delta-sigma oversampled ADC architecture |
| 3.1 | General pipeline subranging ADC |
| 3.2 | General cyclic subranging ADC |
| 3.3 | Radix 2 1-bit-per-stage pipeline subranging ADC 43 |
| 3.4 | Radix 2 1-bit-per-stage ADC algorithm in the sample mode |
| 3.5 | Radix 2 1-bit-per-stage ADC algorithm in the amplify mode 45 |
| 3.6 | Residue plot of a MX2 stage in a radix 2 1-bit-per-stage pipeline ADC 46 |
| 3.7 | Residue plot of a 2-bit-per-stage MX4 stage |
| 3.8 | Errors in radix 2 1-bit-per-stage ADCs. Missing codes and missing decision |
| | levels are present |
| 4.1 | Example of uniform decision level locations |
| 4.2 | Example of nonuniform decision level locations |
| 4.3 | Example of DNL |
| 4.4 | Example of INL |
| 4.5 | Ideal residue and transfer characteristic of a pipeline ADC 5 |
| 4.6 | Residue and transfer characteristic of a pipeline ADC with missing decision |
| | levels |
| 4.7 | Residue and transfer characteristic of a pipeline ADC with missing codes 59 |
| 4.8 | Pipeline ADC with digital calibration applied to the eleventh stage 6 |
| 4.9 | Residue plot of a MX2 stage in a 1-bit-per-stage radix < 2 pipeline ADC |
| | with $d(i+1)-d(i)\approx 2\Delta$ |
| 4.10 | Residue plot of a MX2 stage in a 1-bit-per-stage pipeline ADC with $d(i +$ |
| | $1)-d(i)\approx 0.$ |
| 4.11 | Residue plot of a MX2 stage with comparator offset V_{OS} 68 |
| 4.12 | Digital calibration of higher level stages |
| 4.13 | Determination of calibration constant S_1 |
| 4.14 | Determination of calibration constant S_2 |

LIST OF FIGURES

| 4.15 | Residue plot of a 2-bit-per-stage MX4 stage. Sub-ADC an are indicated | | | | | | 73 |
|--------------|---|------|-----|-------|---|-------|-------------|
| 4 16 | Comparison of decision level locations for the cases of unrec | | | | | | |
| 1.10 | ADC digital output. Ideal and actual locations are shown | | | | | | |
| | cases | | | | | | 75 |
| | | | | | | | |
| 5 .1 | Simulated DNL, $G = 1.9704$, $K = 6$ | | | | | • | 80 |
| 5.2 | Simulated INL, $G = 1.9704$, $K = 6$ | | | | | • | 81 |
| 5.3 | Simulated DNL, $G = 1.9550$, $K = 6$ | | | | | | 82 |
| 5.4 | Simulated INL, $G = 1.9550$, $K = 6$ | | | | | | 83 |
| 5.5 | Simulated DNL, $G = 1.9324$, $K = 6$ | | | | | | 84 |
| 5.6 | Simulated INL, $G = 1.9324$, $K = 6$ | | | | | | 85 |
| | | | | | | | ۰. |
| 6.1 | SC Pipeline ADC. | | | | | | 87 |
| 6.2 | Pipeline ADC schedule | | | | | | 89 |
| 6.3 | SHA in the sample mode | | | | | | 92 |
| 6.4 | SHA in the amplify mode | | | | | | 93 |
| 6.5 | MX2 in the sample mode | | | | | | 96 |
| 6.6 | MX2 in the amplify mode | | | | | | 97 |
| 6.7 | Comparator structure in the reset mode | | | | | | 102 |
| 6.8 | Comparator structure in the amplify mode | • | | • | • | • | 103 |
| <i>p</i> 7 1 | 0 | | | | | | 106 |
| 7.1 | Operational amplifier | | | | | | 110 |
| 7.2 | Common mode feedback schematic | | | | | | 112 |
| 7.3 | Comparator pre-amplifier | | | | | | 113 |
| 7.4 | Comparator latch. | | | | | | 114 |
| 7.5 | Bias distribution network | | | | | | |
| 7.6 | Digital output buffer design | • | • • | • | • | • | 116 |
| 8.1 | Pipeline ADC stage layout | | | | | | 120 |
| 8.2 | Pipeline ADC layout | | | | | | 121 |
| | | | | | | | |
| 9.1 | Test setup for the pipeline ADC | | | | | | 12 4 |
| 9.2 | Layout of the PCB | | | | | | 126 |
| 9.3 | Ground plane design for the PCB | | | | | | 128 |
| 9.4 | Analog input circuit for the pipeline ADC | | | | | | 129 |
| 9.5 | Pipeline control clock driver circuit | | | | | | 131 |
| 9.6 | Calibration control clock driver circuit | | | | | | 131 |
| 9.7 | Reference circuit for the pipeline ADC | | | | | | 133 |
| 9.8 | External bias circuit for the pipeline ADC | | | | | | 134 |
| 10 1 | M 1 DDM -1-4 0 07500 LH - : 0 lD : | D.C. | | | | | 105 |
| | Measured FFT plot, 9.87560 kHz sine-wave input, -2dB | | | | | | 137 |
| | Measured FFT plot, 98.7560 kHz sine-wave input, -2dB | | | | | | 138 |
| 10.3 | Measured FFT plot, 198.756 kHz sine-wave input, -2dB | r D. | • | • | | | 139 |

14 LIST OF FIGURES

| 10.4 | Measured DNL plot | 142 |
|-------------|--|-----|
| 10.5 | Measured INL plot | 143 |
| 10.6 | Pipeline ADC die photograph | 145 |
| A .1 | $ S_v(j\Omega) $ and $ S_x(j\Omega) $ plotted with the $S_v(j\Omega)$ replicas scaled and shifted in | |
| | frequency as a function of Ω | 151 |
| A.2 | $ S_x(j\Omega) $ and $ S_y(j\Omega) $ plotted as a function of Ω | 154 |
| B.1 | Wire-bond diagram for pipeline ADC IC | 160 |
| B.2 | Pin-out diagram for pipeline ADC IC | 161 |

List of Tables

| 3.1 | Gain parameters A_0 , A_1 and A_2 for pipeline and cyclic ADCs as a function of the number of bits resolved per stage, $M.$ | 42 |
|------|---|-----|
| 4.1 | Parameters of a pipeline section as a function of V_{in} . $G=2, K=6$ | 62 |
| 4.2 | Parameters of a pipeline section as a function of V_{in} . $G < 2$, $K = 6$ | 63 |
| 4.3 | Parameters of a pipeline section as a function of V_{in} . Digital calibration | |
| | applied. $G < 2$, $K = 6$ | 65 |
| 7.1 | Op-amp performance summary | 117 |
| 7.2 | Pre-amp performance summary | 117 |
| 7.3 | Latch performance summary | 117 |
| 10.1 | Harmonic analysis, 9.87560 kHz sine-wave input, -2 dB FS | 140 |
| 10.2 | Harmonic analysis, 98.7560 kHz sine-wave input, -2 dB FS. HD7 is aliased | |
| | into the baseband. | 140 |
| 10.3 | Harmonic analysis, 198.7560 kHz sine-wave input, -2 dB FS. HD5, HD4, | |
| | HD19, HD7 and HD3 are aliased into the baseband | 140 |
| 10.4 | Chip performance | 144 |

Chapter 1

Introduction

1.1 Description of the Problem

1.1.1 Overview

An increasing demand for high-performance Digital Signal Processing (DSP) systems motivates an increasing desire for high-resolution, high-speed Analog-to-Digital Converters (ADCs). These ADCs are used in a wide variety of environments, including industrial, communication, medical and aerospace. The use of ADCs in such systems permits so-phisticated DSP algorithms to be utilized, which alleviates complexity in the analog signal processing required otherwise. The analog signal presented to the input of high-resolution, high-speed ADC is frequently a zero-order hold type originating from an imaging array, multiplexer or Sample-and-Hold Amplifier (SHA). Because of the zero-order hold nature of such signals, delta-sigma ADC approaches can not be used to accomplish high-resolution quantization, independent of conversion speed.

Traditional high-resolution, high-speed ADCs rely on expensive, hybrid or discrete implementation and thus are not amenable to low cost manufacturing. The use of monolithic Integrated Circuit (IC) techniques has been repeatedly demonstrated in the digital and analog circuit domains to be economical for synthesizing complex systems. Furthermore, the integration of ADCs with other realms of monolithic systems has been an increasing trend to increase system versatility. The combination of ADCs with other signal processing, transducer, communication, control or ranging systems results in a higher value-added system. Thus, monolithic integration of high-accuracy, high-speed ADCs with other systems requires that the ADC architecture is amenable to the properties of the host processes. However, the desire to integrate high-speed, high-resolution ADCs in low cost, IC processes presents algorithmic, circuit and process design challenges.

1.1.2 Pipeline ADC Architecture

The pipeline ADC architecture is desirable for achieving high-speed, high-resolution ADCs. As a sampled-data system, a pipeline ADC can quantize either continuous or zero-order hold

inputs. One advantage of the pipeline ADC architecture is that it exhibits linear growth in hardware, compared to exponential growth with flash ADCs, for increasing resolution. In addition, pipeline ADCs require fewer clock cycles than successive approximation or oversampled ADCs, which is important for high-speed. However, pipeline ADCs are capable of achieving only 8-10 bits of linearity in most IC processes without the use of component trimming or self-calibration techniques.

1.1.3 Limitations to Switched-Capacitor ADCs

Switched-Capacitor (SC) circuit techniques are efficient for integration of sampled-data systems, such as pipeline ADCs, in Metal Oxide Semiconductor (MOS) based technologies. A SC integration platform combines discrete time signal processing, analog signal processing and algorithmic execution of signal processing functions using charge based signals [1, 2, 3, 4]. These functional attributes can be tedious to duplicate in non-MOS based technologies.

Typically, component values in ADC algorithms are fundamentally related to each other by ratios of the component values. Thus, ratiometric limitations are fundamental limitations of achievable ADC resolution. MOS based technologies share the traditional limitations common to IC processes including mask resolution limits, mask misalignment, nonuniform deposition and nonuniform etching.

Device related errors in MOS based SC systems include channel charge injection, clock feed-through and substrate coupling. MOS devices and other IC components contribute to wide-band thermal noise limitations. Other important aspects of MOS device physics that limit performance include threshold voltage relaxation [5, 6] and 1/f noise [7].

1.1.4 Digital Self-Calibration for Pipeline ADCs

In order to achieve precise ratios, high-precision component tolerances are required in traditional ADC approaches, which are not compatible with low cost, MOS IC processes. As a result, the use of calibration techniques is important to eliminate the need for high-precision matching.

Self-calibration refers to an algorithmic means for correcting ADC errors without the need for high-precision matched components and without the need for special trimming or alignment using mechanical components. This thesis will present a digital self-calibration technique developed for pipeline ADCs with a 1-bit-per-stage, or multiple-bits-per-stage design. The digital self-calibration technique can also be applied to cyclic ADCs. The technique removes ratiometric errors and is simultaneously tolerant to comparator errors. The technique described here accomplishes digital self-calibration without requiring external data converters, complex microprocessor based calibration engines or special calibration training signals. This approach uses innate properties of pipeline ADCs to an advantage for self-calibration. The use of digital self-calibration is important to minimize the quantity, and hence expense, of precision analog circuitry. In contrast, digital circuits are much cheaper to construct. The digital self-calibration will be presented in this thesis from the perspective of a synthesis strategy for a wide range of high-accuracy ADCs. A 15-bit, 1

MS/s prototype was integrated in a $2.4\mu m$ BiCMOS process to demonstrate the digital self-calibration technique.

1.1.5 Digital Error-Correction and Digital Self-Calibration

Digital error-correction refers to methods of correcting comparator errors that occur in pipeline subranging ADCs. These error-correction schemes do not account for ratiometric errors. Error-correction schemes for pipeline subranging ADCs generally assume certain errors that can occur due to comparator errors and then correct for those errors digitally, without measuring any errors. As a result, it is important to dissociate error-correction schemes, which do not measure errors, from self-calibration schemes, which involve measurement of errors. Some self-calibration schemes involve the use of error-correction principles so that both ratiometric and comparator errors are removed. The digital self-calibration presented in this thesis accounts for ratiometric errors and is tolerant to comparator errors, without additional action needed.

1.2 Application Realms

1.2.1 General High-Accuracy High-Speed Data Acquisition

Video Applications

With the advent of powerful, and cheap, integrated circuits that perform a variety of analog and digital functions, the electronic subsystem of a video camera oriented for television use has evolved into an efficient structure, in terms of physical size, power consumption and function. Even with these advances, video cameras still retain a substantial amount of complexity because of the frequent requirement for image transmission in natural color [8]. The reduction of user controls and the corresponding increase of automatic systems in consumer oriented video cameras has spawned an increase in electrical and mechanical complexity [9, 10]. However, the image sensor systems remains at the center of the camera complexity.

Semiconductor image sensors have evolved on a slower path compared to semiconductor circuits [11, 12]. However, advances in Very Large Scale Integration (VLSI) technology, combined with Charge Coupled Device (CCD) circuits, have resulted in practical implementation of silicon based image sensors for video cameras. The use of solid-state CCD image sensors, combined with VLSI circuit techniques, has permitted the overall cost of manufacture of video cameras to drop to the point where mass consumer production is viable [8].

Digital signal transmission, storage and processing schemes for television systems motivates the need for high-performance ADCs to digitize the output signal of the CCD imager. Since the data is already zero-order held, delta-sigma approaches are not appropriate. High-volume production of such systems requires low overall cost and high-performance simultaneously. Other digital video applications include high-speed electronic shuttering CCD

imager based cameras for motion analysis. In order to provide accurate data analysis for such an application, the resolution and speed requirements for the ADC are substantially higher compared to television applications. Overall, the use of monolithic ADCs in solid-state imager based digital cameras is important when the size, weight and power dissipation of the camera systems are critical.

Medical Applications

The medical advantages of noninvasive imaging techniques motivates the use of DSP to perform complex imaging functions. The use of X-Ray Computer Axial Tomography (CAT) requires that high-performance ADCs are used to digitize the output of the crystal and photomultiplier tube detector system [13]. ADC artifacts can impair the analytical capabilities of CAT [14].

The use of nuclear Magnetic Resonance Imaging (MRI) techniques has been motivated from the desire to avoid ionizing radiation for patient evaluation. The use of MRI requires high-sensitivity receivers and high-performance ADCs to digitize the attenuated RF signals after passing through tissues. Because of the extremely large dynamic range requirement of MRI techniques, the receiver and ADC design is particularly challenging. Even more challenging is to perform real-time imaging of tissues [13].

One advantage of ultrasonic imaging compared to CAT or MRI is the absence of ionizing radiation or powerful magnetic and RF fields [13]. Another advantage of ultrasonic imaging is that the stimulus and analysis hardware is intrinsically simpler and less costly than either CAT or MRI systems. However, the tomographic techniques used to reassemble the image are similar among ultrasound, CAT and MRI. The use of DSP to analyze the receiver transducer output requires a high-performance ADC for analytical capabilities. In order to produce a low cost, high-performance, portable ultrasound imaging instrument, low cost techniques for high-performance ADCs are required.

1.2.2 Transducer Based Signal Processing Systems

An important signal processing structure frequently used involves the use of a transducer and an ADC system to quantize the electrical signals so that digital processing can be utilized. The transducers derive electrical signals in response to physical quantities including, but not limited to, pressure, temperature, acceleration and position. Since manipulation of analog signals is challenging from a packaging viewpoint, it is desirable to perform the ADC function as close to the sensor as possible. The combination of silicon micro-machined sensors with ADCs and signal processing capabilities results in a more versatile system. The use of efficient ADC structures is important with silicon micro-machined sensors because of the additional constraints imposed by the host sensor process and the sensor design itself.

1.2.3 Communication Systems

The advantages of digital communication techniques are well established and are increasingly prevalent in high-capacity telecommunication and personal communication links. The combination of communication systems and transducer systems is powerful because once the transducer data is quantized, complex arrangements of transducers can be employed in a network. The ability to use large numbers of sensors to perform wide-area monitoring of physical phenomenon at low cost means that the ADC must be included with each sensor.

The use of ADCs in radio systems presents several potential advantages. Reduction of analog circuitry in combination with advanced DSP algorithms is increasingly important to allow design flexibility in chief radio system parameters such as modulation methods, transmitter power and antenna design. One use of ADCs in superheterodyne based radio systems is to convert the IF signal to digital by sampling at the IF frequency. An ADC used in this capacity can be thought of as a digital demodulator in the radio system. The ability of networked radio systems to change transmitter power depending on node distance is important to minimize interference effects for close proximity nodes yet achieve communication at long distance. As a result, the dynamic range of radio signals can be large and demand high-performance ADCs.

1.2.4 Control Systems

The use of DSP and algorithmic based signal processing methods for control systems provides an impetus for high-performance ADCs used to quantize wide dynamic range analog signals present in such systems. The use of a pipeline architecture presents advantages for realizing high-resolution ADCs used in such applications. As described above, flash ADCs present unfavorable hardware growth for increased resolution, regardless of speed. One of the disadvantages of a pipeline architecture used in control systems is that the latency of the pipeline presents delay. This delay may add excessive phase shift in a control system loop. However, the total delay depends on the number of stages employed so a pipeline architecture has design flexibility to reduce latency to comparable or lower level than successive approximation ADCs, for example. The use of delta-sigma ADCs for high-resolution applications in control systems normally presents greater latency than a pipeline ADC used in a high-performance environment. This latency originates from the long impulse response present in the high-order FIR filter used in the decimator.

1.2.5 Ranging Systems

Both RADAR and SONAR systems present challenging applications of DSP and algorithmic based signal processing. This setting demonstrates another application of high dynamic range ADCs. The use of hybrid construction techniques may be sufficient for traditional applications primarily oriented towards aerospace applications. However, the potential use of sophisticated ranging techniques for automotive collision avoidance systems would likely require a lower cost ADC solution.

1.3 Organization of the Thesis

This thesis is organized in a top-down fashion, overall. Chapter 1 presents the introduction to this thesis. The description of the problem will be stated and the direction of this thesis will be motivated. With this established, the design methodology that the digital self-calibration provides will be outlined. The scope of applications that rely on high-accuracy, high-speed ADCs will be presented.

Chapter 2 discusses the background in this research area. A summary of common ADC techniques, and the relative merits of each approach, are discussed with the focus placed on high-accuracy, high-speed ADCs. Previous studies relating to pipeline ADC calibration techniques are then covered. This includes both analog and digital calibration methods. The advantage of the digital self-calibration presented in this thesis concludes this chapter.

The specific pipeline architecture used in the ADC presented in this thesis is discussed in Chapter 3. The errors present in the pipeline architecture are discussed for 1-bit and multi-bit pipeline ADCs.

Chapter 4 is the central focus of this thesis and will present the digital self-calibration principle. The properties of the digital self-calibration will be discussed including a discussion of the design methodology that the digital self-calibration provides.

A discussion of the simulator used to evaluate the digital self-calibration is discussed in Chapter 5. Simulation results obtained from the simulator are presented.

The SC system design used to implement the Radix < 2 ADC is presented in Chapter 6. Limitations to the SC circuits are discussed.

The analog IC design is shown in Chapter 7. The op-amp, pre-amp and latch circuits are principally discussed.

The layout design is important for achieving a high-performance system and is discussed in Chapter 8.

The overall design of the test system is presented in Chapter 9. The test system design principally reflects the need to maintain signal integrity in the evaluation of a high-performance ADC.

Experimental results are shown in Chapter 10. The sine-wave Fast Fourier Transform (FFT) results are presented. The Differential Nonlinearity (DNL) and Integral Nonlinearity (INL) results are then shown.

Observations derived from this work are presented along with future directions in Chapter 11.

Various appendices follow to discuss specific, detailed issues arising in this thesis. Appendix A presents a detailed SC noise analysis for a pipeline ADC, Appendix B shows the PGA package employed for the ADC, Appendix C shows the digital self-calibration simulator C code programs and Appendix D shows the test system C code programs.

Chapter 2

Background

In this chapter, various ADC techniques are presented in order to establish a context for comparison and contrast to pipeline subranging ADC techniques. With this context established for pipeline subranging ADCs, prior studies in calibration schemes are discussed. The advantages of the proposed digital self-calibration technique are then outlined.

2.1 Summary of ADC Techniques

2.1.1 Flash ADCs

Full Flash ADCs

One of the most direct ADC approaches is the flash ADC [15]. Fig. 2.1 shows that a comparator is assigned for each decision level in the flash ADC. All of the comparator inputs of the same polarity are connected in parallel to the analog input V_{in} . A reference voltage ladder is used to provide the decision level spacing which each comparator compares against. The flash ADC operates by strobing all of the comparators simultaneously. The resulting output codes are usually termed a thermometer code because ideally all the comparator results above the analog signal level will result in 0 and below the analog signal level will result in 1. Combinational logic is then used to translate the thermometer code into a linear binary output, or other desired digital output form. Sophisticated logic error-detection schemes can be used to correct for various error scenarios that may occur.

The advantage of the flash ADC architecture is speed. In principle, no op-amps or SHAs are required. Only a single clock cycle is needed for a conversion cycle. Flash ADCs are used when conversion rate is a primary design consideration [16, 17, 18]. As a result of this architecture, for N bits of resolution, $2^N - 1$ comparators are required. Since each comparator must have a resolution that is consistent with the full converter accuracy, the comparator design dictates the primary complexity of the flash ADC. A primary limitation of this structure to high-accuracy is the exponential growth of hardware with resolution because the number of comparators required is proportional to 2^N . Related to this limitation, the input capacitance, the number of ladder taps, the size of the combinational logic

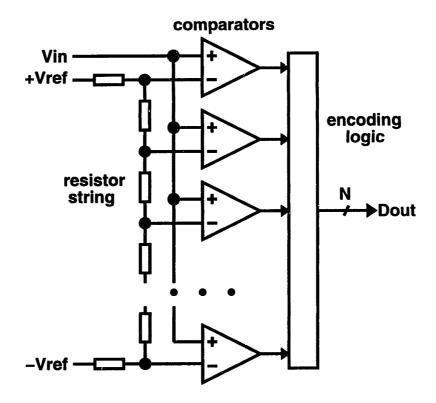


Figure 2.1: Flash ADC architecture.

and the total power dissipation increase exponentially for increased resolution. In practice, 8-bit resolution, corresponding to 255 comparators, is frequently a practical limit based on these collective considerations. The direct approach of the flash ADC leads to high-speed but not high-resolution for practical ADCs.

Folding and Interpolating Flash ADCs

A method for alleviating the comparator requirement in the flash ADC is the use of folding and interpolating structures [16, 19, 18, 20, 21, 22]. The main objective of these structures is to arrive at an ADC that has comparable speed to a full flash ADC, uses less power and area than a full flash ADC and avoids the speed limitation of subranging ADCs. Unfortunately, the hardware growth for folding and interpolating structures is still an exponential function of the number of bits needed, however the proportionality constant is reduced.

2.1.2 Subranging ADCs

The basic principle of a subranging ADC is to divide the quantization into ranges, beginning with an upper MSB section. Progressively lower bit weight sections are then quantized by succeeding stages [15]. This quantization in any particular stage may be performed using a variety of ADC methods. Once the upper MSB section bits are determined, a reconstruction level is produced using a DAC and a residue is produced corresponding to the difference between the analog input and the DAC output. This residue is then processed by another subranging stage. If desired, the subranging stages may be pipelined. When pipelining is not employed, interstage SHAs are eliminated but the throughput usually suffers as a result. The subranging structure is illustrated in Fig. 2.2 for a particular subranging stage.

A commonly used arrangement is a 2-step subranging ADC where 2 subranging stages are used in cascade [23, 24, 25, 26]. Medium resolution, high-speed ADCs typically employ a 2-step subranging architecture. For low-to-medium resolution, usually in the range of 8-bit to 10-bit, component mismatch properties in modern analog IC processes are favorable enough so that self-calibration is not required. In some cases, laser trimming can be used to calibrate the unit once. However, comparator errors can limit resolution. A digital error-correction algorithm can be used to correct the comparator errors, but not any ratiometric errors [27, 28, 29]. Thus, the final accuracy will be limited by the process matching properties.

2.1.3 Pipeline ADCs

Pipeline ADCs are special types of subranging ADCs where clock phases of adjacent stages are opposite. In this fashion, each stage processes the input to that stage and the result is sampled to the next stage. The throughput is improved, because each stage can process new input every two clock phases.

A generalized pipeline structure is shown in Fig. 2.3. The overall approach of a pipeline ADC is to provide a cascade of stages for processing the analog input signal V_{in} . Once the

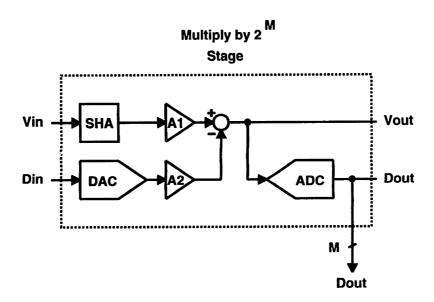


Figure 2.2: One stage of a subranging ADC architecture.

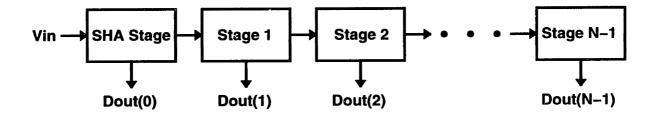


Figure 2.3: Pipeline ADC architecture.

first stage has processed a particular analog input, the result is passed to the next stage. A pipeline ADC is a sampled-data system so means for storing intermediate analog and digital results are necessary. This usually entails interstage SHAs and latches. As a result, the first stage can then acquire a new analog input. Because the data must pass through all of the stages before the answer is complete, the digital results need to be stored intermediately for final time alignment. A shift register function can be used in hardware, or software, in order to perform the time alignment. Once the pipeline and shift register functions are initially filled, the pipeline throughput is limited by the delay of a single processing stage, assuming each stage has the same delay. In sharp contrast to the flash ADC structure, pipeline ADCs only require linear growth in hardware for increasing resolution. Notice that adding another stage to the pipeline will increase the resolution by at least 1-bit, depending on the resolution of the stage. The latency of the pipeline is the summation of all of the processing stage delays. Notice that the latency only grows linearly for increased resolution. Many applications are not sensitive to latency that is on the order of N times larger than the throughput time. However, control systems are one example where latency may be important to consider. This is because of stability concerns arising from the increased phase shift in the loop transmission attributed to the latency.

Pipeline ADCs offer high-speed capability because the throughput of the ADC is determined by the delay of a single stage. The pipeline ADC architecture is general and various methods may be employed within the stages. A 1-bit-per-stage pipeline subranging ADC offers the combination of simplicity and speed in the circuit design for each stage [30, 31, 32, 33, 34, 35]. Multi-bit per stage pipeline subranging ADCs have been demonstrated [36, 37, 27, 28, 29]. This thesis will outline in Chapter 4 an all-digital self-calibration technique that is particularly amenable to pipeline subranging ADCs structures in general and will present a 1-bit-per-stage pipeline ADC with all-digital self-calibration technique implementation [31, 32].

2.1.4 Successive Approximation ADCs

The basic principle of a successive approximation ADC is to arrange a DAC circuit in a feedback system so that the error signal composed of the difference of the analog input and the DAC output is minimized [15]. A digital control system is used to schedule operations that lead to minimization of the error over a number of sequential operations. As the system state evolves, the output resolution improves, assuming that the error is reduced at each sequential operation. The overall structure of this ADC is shown in Fig. 2.4. Notice that this a feedback structure and is thus subject to stability concerns. For instance, the monotonicity of the DAC is very important. If the DAC output decreases for increasing input, then the digital controller will continue incrementing the DAC input until the error is minimized. This results in missing codes. Depending on the digital controller algorithm, limit cycles are possible if the DAC is not monotonic.

Successive approximation ADCs have been traditionally used for medium-resolution, medium-speed ADCs [38]. Digital error-correction schemes which employ faster settling strategies have been reported [39]. The successive approximation ADC is amenable to either

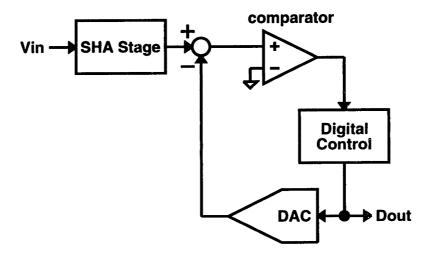


Figure 2.4: Successive approximation ADC architecture.

factory calibration [40, 41] or self-calibration resulting in a high-accuracy ADC [42, 43, 44]. The primary disadvantage to successive approximation ADC is that many clock cycles are needed to complete a single conversion. This presents a disadvantage when high-speed, high-accuracy ADCs are desired.

2.1.5 Delta-Sigma Oversampled ADCs

Delta-Sigma oversampled ADCs have been used successfully for high resolution signal conversion at low frequencies, such as in audio applications [45]. However, a large class of signals that need to be quantized are zero-order held. This includes video data signals originating from CCD imaging systems and multiplexed data signals. Delta-sigma oversampled ADCs can not be employed for these important classes of high-speed signals that in many cases require quantization at high-accuracy.

Fig. 2.5 shows a delta-sigma oversampled structure. A large oversampling ratio is employed in order to achieve high resolution when using low resolution quantizer and reconstruction circuits in a delta-sigma modulator. However, the use of such a delta-sigma modulator to accomplish high-resolution, high-speed with a low resolution quantizer in the delta-sigma results in unacceptably large oversampling ratios, and thus unacceptably large clock rates. In order to relax the oversampling ratio required, a multi-bit quantizer and reconstruction circuits in the delta-sigma modulator can be used. However, the advantages of 1-bit delta-sigma ADCs are less pronounced when multi-bit quantizer and reconstruction circuits are needed, thus limiting the attainable accuracy. The use of self-calibration techniques on the low resolution quantizer and reconstruction circuits is promising for medium to high-resolution at high-speed [46]. Even with a lower oversampling ratio compared to 1-bit delta-sigma ADC designs, the sampling rate required can becomes prohibitively high for present silicon implementations.

In addition, a decimation filter is required to attenuate the out-of-baseband quantization noise resulting from the delta-sigma modulator. One advantage of the oversampling and subsequent decimation approach is that the analog anti-alias filter requirements are relaxed. This advantage of oversampling and decimation can be exploited by any ADC by sampling at super-Nyquist. The particular need for a sharp anti-alias filter function, whether largely accomplished in the analog or digital domain, depends on the application. In situations where input filtering is not required, the delta-sigma oversampled ADC presents the disadvantage of requiring oversampling and decimation to attenuate out-of baseband quantization noise. In a limited set of situations that can tolerate the elevated sampling rate and the resulting data redundancy, the decimation can be waived. If the resulting data needs to be stored in memory or recorded on to media of finite size, then decimation is required, regardless of ADC type.

2.1.6 Cyclic ADCs

These ADCs are a class of algorithmic structures [15] similar to subranging ADCs except the same stage is reused. The digital self-calibration technique can be applied to a cyclic

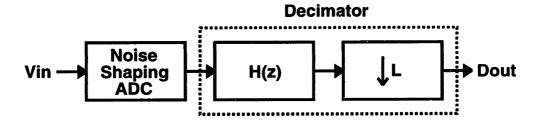


Figure 2.5: Delta-sigma oversampled ADC architecture.

ADC resulting in a compact structure capable of high-resolution. An advantage of the cyclic ADC is that the analog and digital hardware required is reduced. Thus, the digital self-calibration logic requirements collapse to a single accumulator, small memory and controller for the cyclic ADC. The cyclic ADC technique can potentially replace delta-sigma structures because of the simplicity of the digital self-calibration technique. If oversampling is elected to relax the input filtering requirements, then the comparison to be made is between the cyclic ADC employing digital self-calibration and a high-order delta-sigma modulator. If decimation is needed, then either approach would require the same decimator structure. The cyclic approach, with digital self-calibration applied, decouples the achievable resolution from oversampling ratio. Thus, if oversampling is chosen to relax the input filtering requirements, the oversampling ratio can be chosen independent of the resolution. In many cases, oversampling ratios between 2 and 8 are sufficient to greatly relax the input filtering requirements.

2.2 Previous Studies in Pipeline ADC Calibration Techniques

2.2.1 A 12 bit 600 kS/s Digitally Self-Calibrated Pipelined Algorithmic ADC

This paper [33] discusses a digital self-calibration scheme for a radix-2, 1.5-bit per stage pipeline ADCs. This scheme employs 2 comparators per stage in order to resolve 1.5 bits per stage. A digital error-correction scheme is used so that the pipeline ADC is tolerant to large comparator error. In the design presented, ± 0.25 Least Significant Bit (LSB) comparator offset can be tolerated per stage. This is an important advantage for high-speed, low cost implementation in MOS based processes.

Digital Error-Correction

Two comparators are used to determine which of three contiguous regions the residue output is within. The output of each stage is $V_{out} = 2V_{in}(i) - D(i)V_{ref}$ where i is the stage index and where D(i) = -1, 0, +1. The operation of this scheme is outlined in detail in [47]. The basic idea is to use a redundant coding scheme so that multiple digital output words can represent the same analog input, even in the presence of large comparator errors. Digital logic is then used to derive the correct digital output.

Digital Self-Calibration

The basic approach used is to measure the offset and gain error resulting from each stage by applying certain analog inputs to each stage and observing the digital output that results. In this scheme, each stage in the pipeline is measured without using results from stages deeper in the pipeline. The scheme depends on the offset and gain error being relatively small. However, one disadvantage of this digital self-calibration scheme is that it measures

the calibration data under different conditions in the calibration mode compared to the actual run mode so that the achievable accuracy is more limited compared to the digital self-calibration technique presented in this thesis [34].

2.2.2 Digital-Domain Calibration of Multistep Analog-to-Digital Converters

This work [48, 49] outlines a digital self-calibration technique for multi-step ADCs. Digital error-correction is used to relax comparator offset requirements. This digital error-correction makes use of 1-bit of overlap between the first and second stage of a two-step, subranging ADC. This scheme avoids the use of external DACs used for calibration in traditional approaches. As a result, only digital logic needs to be added to the basic, digital error corrected two-step ADC described in this paper.

Multi-Step Architecture

The primary disadvantage of this technique is that it employs a capacitor array in order to obtain accurate closed loop gain for the two-step implementation shown in this work. For pipeline ADCs with many stages, capacitor arrays consume large area and present large load capacitance to high-speed op-amps. The digital-domain calibration here also introduces accumulation of truncation error leading to interstage gain error for many stages employed in pipeline ADCs.

Segmented Transfer Characteristic

The principle used in this work is to consider the ADC transfer characteristic as a synthesis of segments. Ideally, the segments are collinear. However, for a two-step ADC, errors in the DAC reconstruction levels leads to disjoint segments. The calibration must span the entire transfer characteristic and the number of calibration points needed depends on how many segments are desired. It is desirable to use as many segments as possible in order to relax the linearity required for each segment. This is a disadvantage for high-resolution ADCs based on the large number of segments that would become necessary. A competing concern is the digital truncation error that this scheme introduces, which is minimized when the number of segments is minimized.

2.2.3 A 13 bit 2.5MHz Self-Calibrated Pipelined A/D Converter in $3\mu m$ CMOS

In this work [50, 37, 51], a self-calibrating pipeline ADC is described. This approach utilizes digital error-correction to account for comparator errors. Binary weighted capacitor arrays are used to correct for gain stage errors. This scheme avoids the use of external data converters for the calibration.

Analog Calibration

The basic idea in this work is to make use of binary weighted capacitor arrays employed in the interstage gain amplifiers. A search algorithm selects the correct capacitor value needed for the ADC. This procedure is followed for each of the interstage amplifiers used. After this calibration is completed, the ADC can then be used in the run mode, with the digital error-correction used to account for the comparator errors. One of the difficulties with this scheme is that the calibration data is obtained under different conditions than the run mode. This presents an ultimate limit to the resolution achievable with this scheme. Another difficulty is that capacitor arrays consume excess die area and present excess capacitive loading when calibrating many stages.

Digital Error-Correction

This work uses a digital error-correction scheme to account for errors in decision levels in the sub-ADCs in each of the stages. The digital error-correction scheme has been used in other subranging ADC contexts. The basic idea is to provide one extra bit of redundancy so that the residue output of a stage does not overrange the succeeding stage. Since increasing the range of the second stage is not practical, the gain of the next stage is reduced, instead. The digital result from the second stage is then incremented or decremented by one.

2.2.4 Self-Calibrating Pipelined Subranging Analog-to-Digital Converter

The authors in [52] present a scheme for improving the linearity of an ADC that is based on the piecewise linear approximation of the ADC transfer function. The scheme presented is not truly self-calibrating. The gain errors in the unity-gain buffers used between stages are not corrected. A high-precision "gold standard" converter is required for calibration; a delta-sigma ADC is employed for this purpose. Since there is no stage gain in this pipeline, the total noise-power referred to the input is the total noise power generated by the circuits used in the pipeline. The piecewise linear approximation method described in [52] can be applied to other types of ADCs.

System Philosophy.

In the Background of the Invention section, this technique identifies the pipeline ADC structure as a good candidate for potential high-speed and high-resolution. The technique states that the extreme gain matching requirement is identified as a principal linearity error source. Bulk CMOS processes present component matching limitations which limit the accuracy of SC pipeline ADC implementations. The technique further states in the Summary of the Invention section that high-resolution, low-speed ADCs can be used in combination with pipeline ADCs to arrive at a high-resolution, high-speed ADC through a calibration means.

Limitations of Piecewise Linear Approximation.

The calibration principle of this technique is to make use of a piecewise linear approximation of the pipeline ADC transfer characteristic. This method uses 2^{n1} segments, where n1 is the number of bits resolved by the first stage of the pipeline ADC. Typically, n1 will be limited to 8 or 9 because of process limitations. The error between the piecewise linear approximation at the turning-points and the external reference ADC transfer characteristic is measured for a given analog input. These errors are measured for each turning-point of the piecewise linear approximation and stored in memory for computation in accordance to an algorithm. The algorithm used requires multiplication. With n1 = 8, a 256 segment approximation is used. For a 12-bit ADC target resolution, each segment spans 16 codes. For a 16-bit ADC target resolution, each segment spans 256 codes. As a result, this technique becomes increasingly less effective for ADCs beyond the 12-bit level. This is because the approximation becomes increasingly coarse for high-resolution ADCs resulting in increased INL. This situation could be relieved by increasing n1 but that would then require an n1-bit accurate ADC in the first pipeline stage.

Calibration Mode Limitations

It is desirable to calibrate the entire output span of the ADC to eliminate dependence on input signal properties. The ADC in this technique can be run in a distinct calibration mode to accomplish this task. In this case, some means for spanning the entire converter input range must be provided. Thus, additional circuitry must be provided to generate a ramp or other suitable training signal. This training signal must not exhibit any momentary glitches or else some codes may not be calibrated. Because of the long calibration time for the ADC in this technique, a method for generating a slow ramp is necessary. This can prove to be difficult to accomplish using monolithic analog circuits because of the small capacitor values available without consuming enormous chip area. The use of a DAC to generate the ramp would only further compound the disadvantages of this technique in requiring both a high-precision ADC and DAC for calibration.

2.2.5 Architecture for High Sampling Rate, High Resolution Analog-to-Digital Converter System

The authors in [53] present an all digital self-calibration scheme for pipeline ADCs. The approach presented in [53] is specific to pipeline ADCs while the approach presented in [52] is more general and can be applied to other types of ADCs. The scheme in [53] makes use of a "gold standard" converter, namely a delta-sigma ADC. The authors claim that high-accuracy pipeline ADCs are not feasible because present day comparator circuits do not have sufficient precision to resolve against 1 LSB at high-precision. This particular scheme makes use of both the high-precision, low-speed ADC and the low-precision, high-speed pipeline ADC to arrive at calibration data needed to determine a high-precision, high-speed ADC. However, as with [52], the authors in [53] do not provide a method for synthesizing an intrinsically high-precision pipeline ADC.

System Philosophy.

In the Background of the Invention section, this technique identifies the pipeline ADC structure as a good candidate for potential high-speed and high-resolution. The technique states that the comparator resolution requirement of a pipeline is seen as one of the principal errors that cannot be addressed with present day technology. The technique further states in this section that high-resolution, low-speed ADCs, such as delta-sigma and integrating ADCs, typically do not have the comparator offset problem. Thus, while ignoring the properties of subranging pipeline ADCs, the notion that high-resolution, low-speed ADCs are required is reinforced, as was in [52]. The execution of the calibration method using a high-resolution, low-speed ADC in this technique results in a long calibration time because each output code of the pipeline ADC must be calibrated to the high-resolution, low-speed ADC.

Limitations to Calibration Mode

This technique can perform calibration using the input signal as the training signal. However, it is desirable to calibrate the entire output span of the ADC to eliminate dependence on input signal properties. In this case, some means for spanning the entire converter input range must be provided. Thus, additional circuitry must be provided to generate a ramp or other suitable training signal. This training signal must not exhibit any momentary glitches or else some codes may not be calibrated. Because of the long calibration time for the ADC in this technique, a method for generating a slow ramp is necessary. This can prove to be difficult to accomplish using monolithic analog circuits because of the small capacitor values available without consuming enormous chip area. The use of a DAC to generate the ramp would only further compound the disadvantages of this technique in requiring both a high-precision ADC and DAC for calibration.

2.2.6 Digital Error Correction System for Subranging Analog-to-Digital Converters

The use of an all-digital self-calibration scheme is applied to a pipeline ADC described in [54]. The digital self-calibration system incorporated in this technique involves the use of a Least Mean Squared (LMS) algorithm together with a look-up table to arrive at a calibrated output. In this scheme, a "gold standard" ADC or DAC is required with accuracy near an order of magnitude higher than the intended accuracy. An extra 2-bits is mentioned as sufficient to arrive at 12-bits resolution using 14-bit converters. The calibration data for this approach is typically determined at the factory and then the data is stored in a Read-Only-Memory (ROM). This calibration method in this scheme is not a true self-calibrating approach. The approach cannot be used to address the issue of how to arrive at the "gold standard" ADC. Because of the external data converters used, the ADC in this technique operates in the calibration mode under different conditions than during the run mode.

System Philosophy

In the Background of the Invention section, this technique identifies the pipeline ADC structure as a good candidate for potential high-speed and high-resolution. The technique states that practical circuit nonidealities such as component mismatching, nonlinerity and offset are some of the principal errors that afflict pipeline ADCs. In the Summary of Invention section, the technique describes that a LMS adaptation calibration algorithm is used to accommodate practical circuit nonidealities. In the Preferred Embodiment section, "gold-standard" data converters are used to derive the "exact" output for the LMS adaptation algorithm. Thus, while ignoring the properties of subranging pipeline ADCs, the notion that additional high-resolution data converters required is reinforced, similar to the notion that high-resolution ADCs were required in [52, 53]. Thus, the techniques [52, 53, 54] each use the same basic premise of requiring external high-resolution data converters for calibrating pipeline ADCs. The school of thought in this technique is to disregard properties of the pipeline ADC and to instead treat the calibration as an optimization problem.

Limitations to Calibration Mode

The LMS algorithm is an iterative algorithm that must converge in order to obtain accurate updates. The chief pacing element in the iterations needed for convergence is the slow, external reference data converter. To calibrate the entire output span of the converter in view of a pipeline with arbitrary errors, every code must be accessed. This is necessary because the LMS algorithm described is a computational procedure that is generally independent of principles involved with the ADC that is to be calibrated. However, the calibration time requirement can be reduced if certain errors can be parameterized. One limitation to the LMS algorithm from the point of view of low cost monolithic implementation is that multiplication is required for the update equation. The LMS scheme described incurs additional cost when using a digital or analog random signal generator to provide the necessary calibration training signals, as suggested.

2.3 Advantages of the Proposed Digital Self-Calibration

The digital self-calibration technique presented in this thesis presents an entirely different approach compared to the digital self-calibration schemes in [54, 53, 52] as a true self-calibration technique for pipeline ADCs. The digital self-calibration presented here does not require matching better than 7-bit of any components used in the ADC system. This is in sharp contrast to systems which require "gold standard" converters.

The chief advantage of the digital self-calibration technique presented here is that it permits high-resolution, high-speed pipeline ADCs without requiring high component tolerances that are incompatible with low cost MOS processes. Since no external data converters are required, the digital self-calibration technique achieves higher accuracy at lower cost. Only simple calibration input signals are needed, namely analog ground. This eliminates the cost of building circuits that generate special calibration training signals. An efficient

calibration algorithm permits much shorter calibration times enabling the ADC to be used in environments that do not tolerate long calibration delays. The digital processing required does not require complex processors or large memories. Thus, the digital processing circuits can be made smaller resulting in reduced die area and thus reduced cost.

The digital self-calibration technique presented here may be applied to pipeline or cyclic ADC architectures. The digital self-calibration technique can be used to synthesize a family of ADCs spanning a wide range of resolutions and speeds through the basic use of a pipeline or cyclic structure with subranging stages. Design effort, and hence cost, can be reduced by first designing a variety of subranging stages appropriate for the speed desired. By cascading a certain number of stages, a particular resolution is achieved. As an example, for pipeline ADCs, when higher resolution is desired, more stages are used and when less resolution is desired, fewer stages are used. The digital self-calibration logic construction shares this modular property. This modular property of the analog and digital hardware is not displayed in most other ADC architectures, although delta-sigma ADCs have modular attributes. In the remaining ADC cases, significant redesign for different resolution combinations is required. Finally, the digital self-calibration technique does not depend on any particular technology used for circuit implementation. This is an important property because the economics of a particular technology can be determined first and then the ADC can be implemented in that technology. With other ADC approaches, the reverse case is more typical, leading to high cost custom IC processes necessary for implementation.

The application of digital self-calibration to cyclic ADCs results in a family of high-resolution and low-to-medium speed ADCs. This ADC approach may improve overall performance with lower cost and less design effort compared to delta-sigma or successive approximation ADCs. Such ADCs are typically used in consumer applications. Digital self-calibrating cyclic ADCs exhibit much less analog complexity and much less digital complexity compared to delta-sigma ADCs while attaining similar or greater resolution. This is an important advantage for cost sensitive applications, such as consumer products.

This thesis presents a digital self-calibration technique based on a radix 1.93 and a one-comparator-per-stage conversion algorithm. A nonradix 2 conversion algorithm was previously employed in a successive approximation converter [40]. That technique required a precise external calibration source and hence could be factory-calibrated only. The technique described here is self-calibrating, simple and tolerant to comparator errors. The nonunity gain resulting from the approach here can be easily compensated elsewhere in the system.

The key advantage of the digital self-calibration technique reported in this thesis is that the errors at the carry transitions are directly measured under the same condition as during the normal conversion. Therefore, this technique offers potentially higher calibration accuracy than other calibration techniques [42, 49, 37] which measure the error at different conditions than the actual conversion. Another important aspect of this design is that the calibration is performed in the digital domain so no extra analog circuitry, such as weighted capacitor arrays, are needed and no extra clock cycles are needed during the conversion [55, 42]. The digital self-calibration technique reported here automatically accounts for

capacitor mismatch, capacitor nonlinearity leading to DNL, charge injection, finite op-amp gain and comparator offset. The nominal offsets of the op-amp and comparator are reduced by standard offset cancellation and subsequently eliminated with digital self-calibration techniques.

Unlike analog calibration, digital calibration does not correct or create analog decision levels. Therefore, the uncalibrated ADC must provide decision levels spaced no greater than 1 LSB at the intended resolution. In one-bit-per-stage pipeline ADCs, missing decision levels result when the input of any of the stages exceeds the full scale due to capacitor mismatch, capacitor nonlinearity, charge injection, finite op-amp gain and comparator offset. The missing decision levels cannot be removed by digital calibration alone. Missing decision levels can be eliminated, however, by using gain less than 2 and 2-3 more stages of pipeline which gives enough redundancy in the analog decision levels. With gain less than 2, missing codes are introduced rather than missing decision levels. The missing codes that result with a gain less than 2 are eliminated by the digital calibration. In contrast to an elaborate calibration reported previously using gain less than 2 [55], the calibration reported here is much simpler and more accurate. No multiplication is needed in the calibration algorithm and only a small digital memory is needed.

Chapter 3

Pipeline ADC Architecture

3.1 General Pipeline Subranging Structure

A general pipeline subranging structure is a cascade of stages, as shown in Fig. 3.1. The analog input V_{in} is first presented to the SHA stage. The output of the SHA stage then drives the remaining pipeline stages. Notice that each pipeline stage uses a SHA. Each stage resolves M(i) bits where i is the stage index, $0 \le i \le J - 1$. There is one SHA stage and J - 1 Multiply-by- $2^{M(i)}$ stages. The total number of bits resolved is $N = M(0) + M(1) + \cdots + M(J-1)$. An ADC and a DAC of M(i) bits each are used for each Multiply-by- $2^{M(i)}$ stage. An ADC of M(0) bits is used for the SHA stage. The output of each SHA in the Multiply-by- $2^{M(i)}$ pipeline stages is amplified by $A_1(i)$. The output of the DAC is then amplified by $A_2(i)$ and subtracted from the output of amplifier $A_1(i)$. The subtractor output is then passed on to the next stage. In addition, the subtractor output is quantized by an ADC in stage i and the digital output is also passed on to the next stage.

The cyclic subranging ADC structure is shown in Fig. 3.2. An input switch toggles between the sample and convert connections. The switch is first set to the sample mode. The analog input V_{in} is then sampled by the SHA stage. The switch is then set to the convert mode. The output of the Multiply-by- $2^{M(0)}$ stage is then passed back to the input SHA stage. This process repeats for a total of J-1 cycles. The total number of bits resolved is N=JM(0).

Table 3.1 shows the values assumed by A_0 , A_1 and A_2 for a given stage as a function of the number of bits resolved per stage. The gain A_0 is typically 1 for the pipeline or cyclic ADC structures. Table 3.1 assumes that M bits are resolved in the SHA and Multiply-by- 2^M stages for either the pipeline or cyclic ADC structures.

3.2 Radix 2 1-Bit-per-Stage Pipeline Subranging ADC Algorithm

A radix 2 pipeline ADC is of interest because of the simplicity of the components needed in the pipeline stages. The radix of a pipeline refers to the interstage gain that is used.

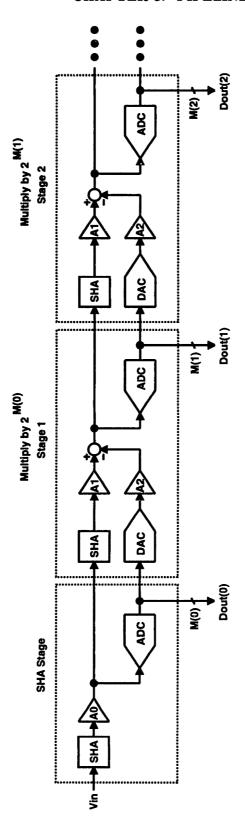


Figure 3.1: General pipeline subranging ADC.

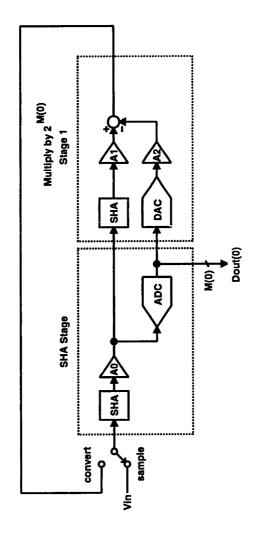


Figure 3.2: General cyclic subranging ADC.

| M | A_0 | A_1 | A_2 | |
|---|-------|-------|-------|--|
| 1 | 1 | 2 | 1 | |
| 2 | 1 | 4 | 3 | |
| 3 | 1 | 8 | 7 | |
| 4 | 1 | 16 | 15 | |

Table 3.1: Gain parameters A_0 , A_1 and A_2 for pipeline and cyclic ADCs as a function of the number of bits resolved per stage, M.

In the context of the generalized pipeline structure, the radix refers to gain A_1 . Thus, a radix 2 pipeline ADC uses an interstage gain of 2 and utilizes a 1-bit-per-stage architecture. Fig. 3.3 shows the basic architecture of such a pipeline. For the radix 2 pipeline ADC, the ADC and DAC in each pipeline stage each have a 1-bit resolution. A 1-bit ADC is simply a comparator providing one decision level. A 1-bit DAC is simply a pair of switches to ronnect to two different reconstruction levels.

The algorithm used for the radir 2 pipeline ADC is discussed next. Fig. 3.4 illustrates stage i in the sample mode. Notice that the 1-bit ADC, or comparator, in stage i is reset during this time and is not connected to other circuits. The analog signal $V_{in}(i)$ is presented to the sampling capacitors of stage i with an op-amp in the unity gain connection. The topplates of the capacitors are connected to the op-amp. A 1-bit ADC, or comparator, from the previous stage i-1 monitors the analog input $V_{in}(i)$. The digital input bit $D_{in}(i)=1$ if $V_{in}(i)>0$ and $D_{in}(i)=0$ if $V_{in}<0$. Fig. 3.5 illustrates stage i during the amplify mode. The op-amp in stage i is now connected in an inverting amplifier configuration. The bottom plates of one of the capacitors is connected to the op-amp output and the other bottom plate is connected to either $+V_{ref}$ or $-V_{ref}$, depending on the state of the input bit $D_{in}(i)$. The pair of switches to $+V_{ref}$ and to $-V_{ref}$ performs the 1-bit DAC function. The op-amp output is $2V_{in}(i)-V_{ref}$ if $D_{in}(i)=1$ and is $2V_{in}+V_{ref}$ if $D_{in}(i)=0$. The comparator in stage i monitors the output $V_{out}(i)$ and provides the bit decision to the next stage.

The algorithm that each stage i follows is now summarized. The quantities $D_{in}(i)$ and $D_{out}(i)$ refer to the digital input and output, respectively, of stage i. The quantities $V_{in}(i)$ and $V_{out}(i)$ refer to the the analog input and output, respectively, of stage i.

$$V_{out}(i) = 2V_{in}(i) + V_{ref}, if D_{in}(i) = 0,$$
 (3.1)

$$V_{out}(i) = 2V_{in}(i) - V_{ref}, if D_{in}(i) = 1,$$
 (3.2)

$$D_{out}(i) = 0, if V_{out}(i) < 0,$$
 (3.3)

$$D_{out}(i) = 1, if V_{out}(i) > 0$$
 (3.4)

A residue plot showing $V_{out}(i)$ plotted as a function of $V_{in}(i)$ is shown in Fig. 3.6 for the cases $D_{in}(i) = 0$ and $D_{in}(i) = 1$. Each stage i follows this algorithm on the data that the

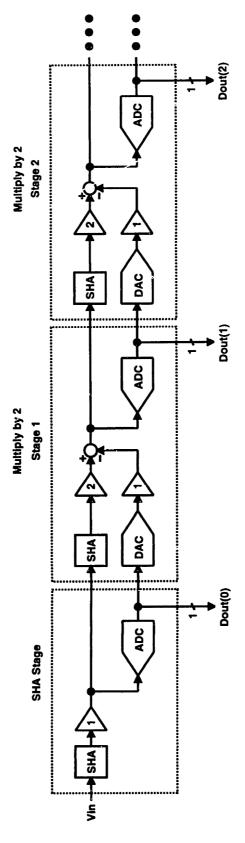


Figure 3.3: Radix 2 1-bit-per-stage pipeline subranging ADC.

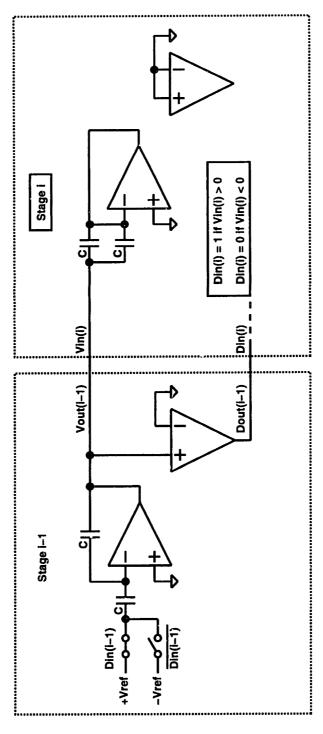


Figure 3.4: Radix 2 1-bit-per-stage ADC algorithm in the sample mode.

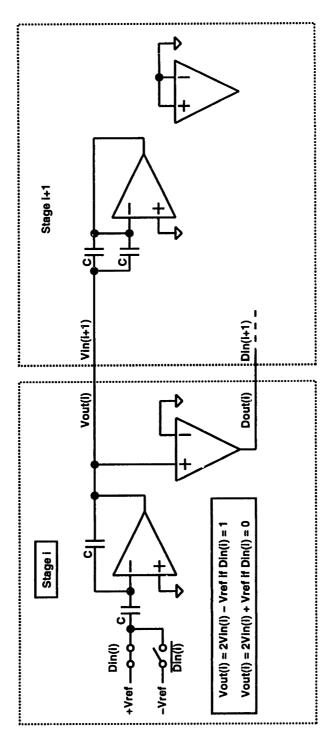
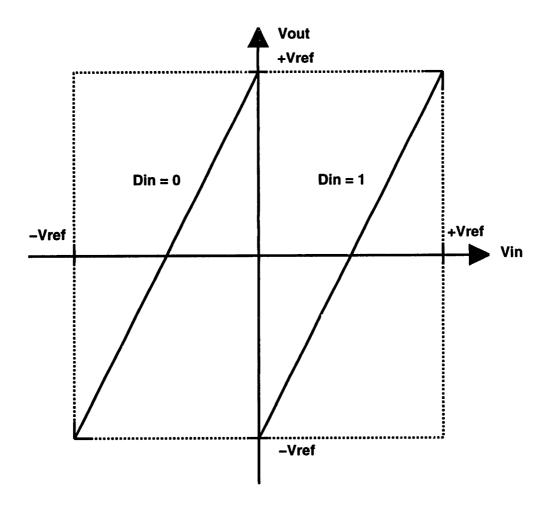


Figure 3.5: Radix 2 1-bit-per-stage ADC algorithm in the amplify mode.



$$V_{out} = 2V_{in} + V_{ref}, if D_{in} = 0,$$
 (3.5)

$$V_{out} = 2V_{in} - V_{ref}, if D_{in} = 1$$
 (3.6)

Figure 3.6: Residue plot of a MX2 stage in a radix 2 1-bit-per-stage pipeline ADC.

stage is presented and presents the output to the succeeding stage. As the data progresses through the pipeline, additional LSBs are determined. Once a pipeline stage has performed its function on a given set of input data, new data can be accepted subsequently. Thus, the throughput of the pipeline is the processing time T of a single stage while the latency of the pipeline is JT. This property of a pipeline permits high-speed and high-accuracy simultaneously.

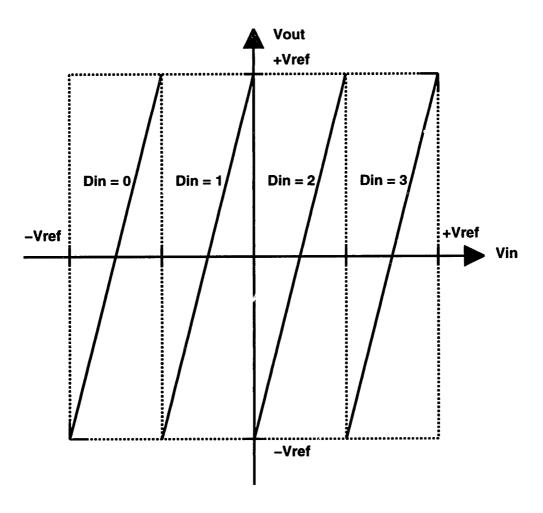
Fig. 3.7 shows a residue plot of a 2-bit-per-stage MX4 stage. The output $V_{out}(i)$ is plotted as a function of $V_{in}(i)$ for the cases $D_{in}(i) = 0$, $D_{in}(i) = 1$, $D_{in}(i) = 2$, $D_{in}(i) = 3$.

3.3 Radix 2 1-Bit-per-Stage Pipeline ADC Error Sources

Fig. 3.8 shows the ideal residue plot and the effects of principal errors on the residue plot. The dashed box represents the reference boundary that passes through coordinates $\pm V_{ref}$ along the V_{out} axis and $\pm V_{ref}$ along the V_{in} axis. Charge injection offset causes a vertical shift of the residue plot. Near the major carry transition point, the residue exceeds the reference boundary resulting in missing decision levels. This is because the remaining pipeline section is saturated so that the output code does not change for the corresponding range of analog input. Near the major carry transition point, the residue minimum does not extend to $-V_{ref}$ resulting in a gap from the minimum to the reference boundary. As a result, missing codes result. This is because the full input range of the remaining pipeline section is not accessed. Comparator offset causes a shift of the major carry transition point. This leads to the residue exceeding the reference boundary as well as leading to a gap to the reference boundary. Again, missing decision levels and missing codes result, respectively. Finally, capacitor mismatch as indicated causes the residue to exceed the reference boundary near the major carry transition point resulting in missing decision levels. Capacitor mismatch could also lead to a gap from the residue extrema to the reference boundary near the major carry transition point resulting in missing codes.

3.4 Radix < 2 1-Bit-per-Stage Pipeline ADC Motivation

As discussed above, when the output of any stage in a radix 2 pipeline ADC exceeds $\pm V_{ref}$, missing decision levels occur, which cannot be eliminated by digital calibration alone. The key is to use a nominal gain less than 2 such that the output of each stage never exceeds $\pm V_{ref}$. For the case of radix < 2, the principal errors affect the residue output in a similar manner as with the case of radix 2. Since the residue is maintained within the reference boundary, no missing decision levels result. The missing codes that result, due to the gaps from the residue extrema to the reference boundary, are eliminated with digital calibration.



$$V_{out} = 4V_{in} + 3V_{ref}, if D_{in} = 0,$$
 (3.7)

$$V_{out} = 4V_{in} + V_{ref}, if D_{in} = 1,$$
 (3.8)

$$V_{out} = 4V_{in} - V_{ref}, if D_{in} = 2,$$
 (3.9)

$$V_{out} = 4V_{in} - 3V_{ref}, if D_{in} = 3$$
 (3.10)

Figure 3.7: Residue plot of a 2-bit-per-stage MX4 stage.

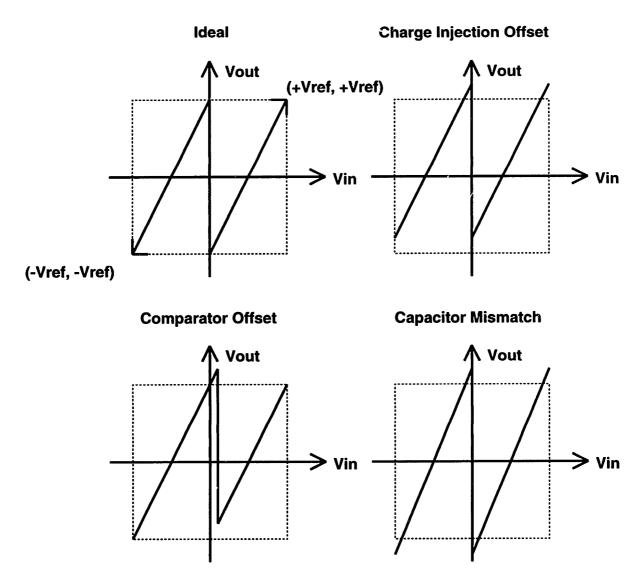


Figure 3.8: Errors in radix 2 1-bit-per-stage ADCs. Missing codes and missing decision levels are present.

3.5 Radix < 2 Pipeline ADC Properties

In order to synthesize a digital self-calibration scheme, some properties of a radix < 2 pipeline ADCs will be noted. It is important to view the pipeline ADC from the standpoint of producing output numbers that are not necessarily related by a power-of-2. The task of correct output code assignment is relatively simpler than the task of correct decision level assignment. The most important property of the radix < 2 pipeline ADC is that the decision level spacing is automatically maintained even though missing codes result. This will be seen in the next section. The digital self-calibration removes the missing codes. The resulting converter then has a transfer characteristic that may have a simple gain error. This can be compensated for in the analog domain, digital domain or both. In situations where the converter is used in a DSP system, multiplication of the ADC data by an appropriate constant is easily performed, if needed.

Chapter 4

Digital Self-Calibration

4.1 Introduction

The goal of the digital self-calibration technique is to establish a means for a high-accuracy, high-speed pipeline ADC. The digital self-calibration technique for pipeline ADCs incorporates several important concepts simultaneously. The essential structure of the individual ideas and the collective digital self-calibration technique will be presented in this chapter. The key points in this chapter are outlined first.

First, the primary function of an ADC is temporal and amplitude quantization of the analog input signal. An ideal ADC samples the analog input signal in equal time steps and in equal amplitude steps. Provision of equal decision level spacing in an ADC is one of the most difficult tasks for a high-accuracy, high-speed ADC. Ratiometric accuracy limitations of component values, charge injection and offsets can contribute missing decision levels. Thus, missing decision levels reflect analog circuit imperfections. Decision levels cannot be created by digital techniques alone. Provision of decision levels must be accomplished by the analog circuit design. In the previous chapter, a radix < 2 pipeline ADC structure was shown to have no missing decision levels but resulted in missing codes. In general, an important principle of digital self-calibration is that although digital techniques cannot be used to eliminate missing decision levels, such techniques can be used to eliminate missing codes. The task of eliminating missing codes is equivalent to code reassignment.

Second, the total number of decision levels that an ideal ADC provides is a nonzero, positive integer that may be based on a power-of-2, but not necessarily. Removing the restriction of a power-of-2 number of decision levels is a powerful concept because this then permits the use of a radix < 2 pipeline ADC structure. As was seen in the last chapter, a radix < 2 pipeline ADC can eliminate missing decision levels. The digital self-calibration removes missing codes that result from the radix < 2 pipeline ADC and is tolerant to comparator error. These properties will be shown in detail.

Third, since digital operations on the ADC outputs do not create or eliminate decision levels, the decision level spacing from the calibrated ADC must be no wider than 1 LSB at the intended resolution. By using a redundant number of output bits, this can be

guaranteed. With a redundant number of output bits, and hence a redundant number of decision levels, the decision level spacing from the calibrated ADC will be smaller than 1 LSB at the intended resolution.

4.2 Basic ADC Metrics

4.2.1 Scalar Quantization

An ideal ADC assigns outputs to a finite set of intervals determined from the analog input. This process is called scalar quantization [56, 57]. The analog input v is considered to be divided into intervals spaced by decision levels d(i). Thus, the intervals r(i) are:

$$r(i) \in \{d(i) < i < d(i+1)\}, \ 0 \le i \le 2^N - 1$$
 (4.1)

where N is the number of bits that the ADC resolves. Thus, there are a total of 2^N intervals. There are 2^N-1 decision levels not including d(0) and $d(2^N)$ which are used to delineate the input range of the ADC. Thus, $V_{FS} = d(2^N) - d(0)$. Uniform quantization occurs when the decision levels are spaced equally along the amplitude axis. Fig. 4.1 illustrates an example of uniform decision level locations and Fig. 4.2 illustrates an example of nonuniform decision level locations.

4.2.2 Coding

The intervals r(i) are each mapped to outputs w(i). A binary coding scheme for w(i) is desired. The simplest binary coding scheme results in:

$$w(i) = i, \ 0 \le i \le 2^N - 1$$
 (4.2)

Other binary coding schemes may be used depending on system constraints. For some applications, a Gray-code scheme is employed where only one bit changes at one time.

An N-bit binary word can be used to represent up to 2^N outputs. Notice that the total number of outputs, and consequently the total number of intervals and decision levels, does not necessarily have to be a power-of-2.

4.2.3 Differential Nonlinearity

Differential Nonlinearity (DNL) is a measure of how uniform decision levels are spaced in an ADC. A tedious method of showing how uniform the decision levels are spaced would be to plot the transfer characteristic of an ADC. The resulting staircase waveform would quickly become unreadable for a large number of decision levels. A simpler method is to express the difference between the actual decision level spacing and the ideal decision level spacing:

$$DNL(i) = \hat{A}(i) - A(i), \ 0 \le i \le 2^{N} - 1$$
 (4.3)

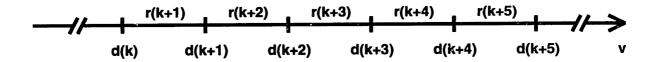


Figure 4.1: Example of uniform decision level locations.

$$\frac{r(k+1) + r(k+2) + r(k+3) + r(k+4) + r(k+5)}{d(k) + d(k+1) + d(k+2) + d(k+3) + d(k+4) + d(k+5)} \bigvee$$

Figure 4.2: Example of nonuniform decision level locations.

where $\hat{A}(i)$ is the actual decision level spacing and A(i) is the ideal decision level spacing, in general. As a result:

$$\hat{A}(i) = d(i+1) - d(i), \quad 0 \le i \le 2^{N} - 1 \tag{4.4}$$

For a uniform quantizer, the ideal decision level spacing is constant:

$$A(i) = \Delta, \ 0 \le i \le 2^N - 1$$
 (4.5)

where $\Delta = V_{FS}/2^N$ is the ideal decision level spacing for a uniform quantizer with Full-Scale (FS) range V_{FS} . For an ideal ADC, DNL(i) = 0 for all i. Notice that DNL(i) > 0 when the decision level spacing is larger than ideal and DNL(i) < 0 when the decision level spacing is smaller than ideal. A DNL(i) of -1 corresponds to a missing code.

4.2.4 Integral Nonlinearity

Integral Nonlinearity (INL) is a measure of how uniform the decision level spacings are over long ranges of the transfer characteristic. The INL characterizes the transfer characteristic completely to within a scale factor. The quantity INL(i) is the cumulative DNL(i):

$$INL(i) = \sum_{k=0}^{i} DNL(k), \ 0 \le i \le 2^{N} - 1$$
 (4.6)

The INL reflects the deviation of the actual ADC transfer characteristic from the ideal transfer characteristic. Notice that the INL is derived from the DNL. However, the DNL could also be derived from the INL. Thus, it is seen that:

$$DNL(i) = INL(i+1) - INL(i), \ 0 \le i \le 2^{N} - 1$$
 (4.7)

4.2.5 Examples of DNL and INL

Fig. 4.3 shows an example of DNL and Fig. 4.4 shows an example of INL. The DNL and INL are derived from the nonuniform decision level locations shown above. The ideal decision level spacing is derived from the uniform decision level locations shown above.

4.3 Principles of Digital Self-Calibration for Pipeline ADCs

4.3.1 Introduction

The pipeline synthesis of the digital self-calibration described here represents an important departure from other self-calibration techniques. The idea here is to use the pipeline itself during the self-calibration mode. This arises from the observation that any consecutive subsection of a pipeline ADC can be viewed itself as an ADC. The result is an all-digital self-calibration technique that requires no analog trimming, no external data converters and no special calibration signals.

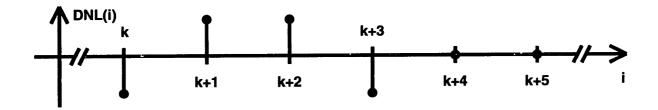


Figure 4.3: Example of DNL.

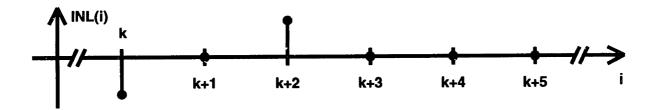


Figure 4.4: Example of INL.

The synthesis procedure begins with a last section of pipeline of K stages. Assuming that this last section is monotonic with no missing codes over K bits, this section can be used to measure the preceding stage during the calibration mode. This last K bit section is then combined with the preceding stage to yield a K+1 stage ADC. The total number of decision levels is almost doubled since the stage that was calibrated had a gain less than 2. The digital self-calibration aligns the transfer characteristic of the K bit section together at the major carry transition point. As a result, if the decision levels were correctly positioned in the K bit section as well.

4.3.2 MX2 Stage with G = 2

Fig. 4.5 shows an ideal pipeline section of K stages at the end of the pipeline with an ideal MX2 stage that has a gain G=2. This idealized structure is instructive in understanding the operation of the digital self-calibration. The ideal MX2 stage is shown with an analog input V_{in} , a digital input D_{in} , an analog output V_{out} and a digital output D_{out} . The output bit D_{out} has the K bits concatenated to result in the quantity X. This quantity X represents the quantization of the residue V_{out} . The digital input D_{in} has the quantity X concatenated to result in the quantity $D_{in}: X$. This quantity $D_{in}: X$ represents the quantization of the input V_{in} . The residue plot is shown for the two cases $D_{in}=0$ and $D_{in}=1$. The transfer characteristic is also shown with $D_{in}: X$ plotted against V_{in} . In this ideal pipeline ADC, the transfer characteristic is perfectly linear. In reality, the $D_{in}: X$ axis assumes only discrete values but the transfer characteristic is drawn continuously for simplicity in illustration.

4.3.3 MX2 Stage with G > 2

The same structure as before is now shown with a gain G > 2 in Fig. 4.6. The residue plot clearly indicates that the residue V_{out} exceeds the reference boundary box. The transfer characteristic is composed of two linear regions, corresponding to $D_{in} = 0$ and $D_{in} = 1$, respectively, that are disjoint near $V_{in} = 0$. Notice that the wide range of analog input near $V_{in} = 0$ results in no change of the output code. As a result, missing decision levels result in this region. The missing decision levels cannot be eliminated with digital calibration alone.

4.3.4 MX2 Stage with G < 2

The same structure as before is now shown with a gain G < 2 in Fig. 4.7. The residue plot clearly indicates that the residue V_{out} is contained within the reference boundary box. The transfer characteristic is composed of two linear regions, corresponding to $D_{in} = 0$ and $D_{in} = 1$, respectively, that are disjoint near $V_{in} = 0$. Notice that there is a sudden change in code near $V_{in} = 0$. As a result, missing codes result in this region. The missing codes can be eliminated with digital self-calibration.

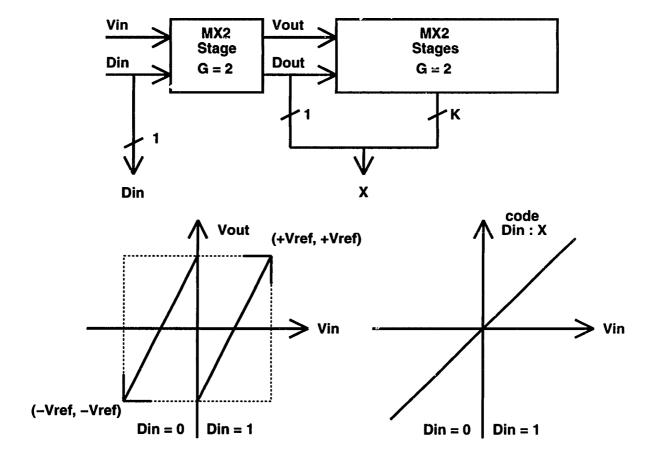


Figure 4.5: Ideal residue and transfer characteristic of a pipeline ADC.

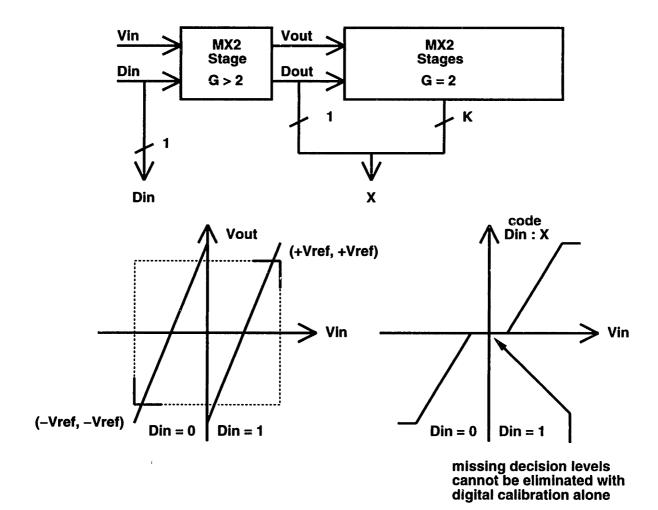


Figure 4.6: Residue and transfer characteristic of a pipeline ADC with missing decision levels.

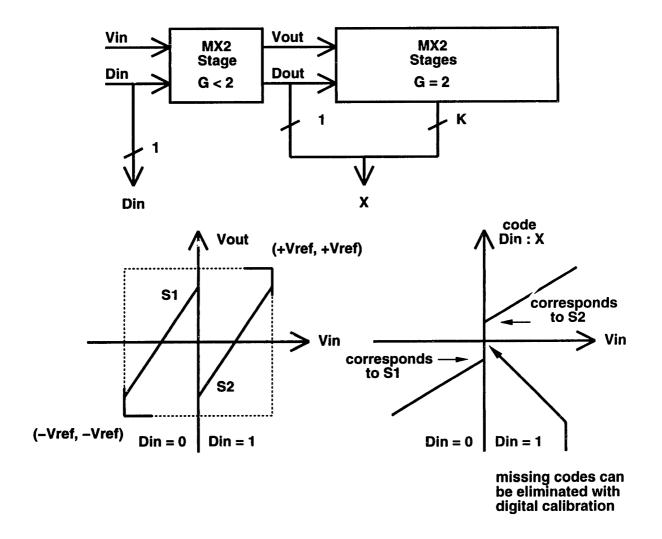


Figure 4.7: Residue and transfer characteristic of a pipeline ADC with missing codes.

4.3.5 MX2 Stage with G < 2 and Digital Calibration Applied

The same structure as before is now shown with a gain G < 2 and with digital calibration logic applied in Fig. 4.8. In this figure, the MX2 stage identified is the eleventh stage and the last section of pipeline is six stages, as actually used in the pipeline ADC. Thus, K = 6. The digital calibration logic accepts inputs D_{in} and X. The output Y is the calibrated output. The digital calibration logic accepts two inputs S_1 and S_2 which are digitized residue extrema indicated in Fig. 4.8. These parameters are shown on the residue plot also. It will be seen later that these parameters correspond to X when the pipeline ADC shown in Fig. 4.8 is used with certain input conditions. The transfer characteristic is shown with no missing codes, as desired.

4.3.6 Interpretation of the Last K Stage Pipeline Section

The last K stage pipeline section was shown to measure the preceding reduced radix MX2 stage. The resulting ADC has a resolution that is somewhat less than a radix 2 pipeline ADC of K+1 stages because fewer decision levels are available due to the reduced radix. As the calibrated pipeline is synthesized, the current last section of calibrated pipeline has an analogous role of measuring the next preceding reduced radix MX2 stage. This follows from the key concept that any contiguous section of a pipeline ADC is itself a pipeline ADC which can be used for calibration.

For a reduced radix stage, the maximum possible input range may extend beyond $\pm V_{ref}$ without exceeding the input range of the remaining pipeline section. However, the actual input range of a given reduced radix stage may be less than its maximum possible input range. This constraint originates from the last K stage section of pipeline that uses radix 2 stages. In order to avoid missing decision levels, the input to the last K stage section must not exceed $\pm V_{ref}$. As K is decreased, the maximum permissible input range of a given reduced radix stage increases. However, as K is decreased, the total number of decision levels decreases since there are then more stages with reduced radix, assuming a fixed total number of stages. Notice that the total number of decision levels that can be accessed is set by 2^N where N is the total number of stages. The actual number of decision levels accessible will always be less than 2^N for a radix < 2 pipeline ADC.

4.4 Calibration Algorithm

4.4.1 Introduction

One of the goals in arriving at a digital self-calibration algorithm was to avoid the need for complicated numerical operations, large look-up tables or specialized calibration training signals. The origins of the algorithm begins with the previous ADC pipeline structure used to introduce digital self-calibration.

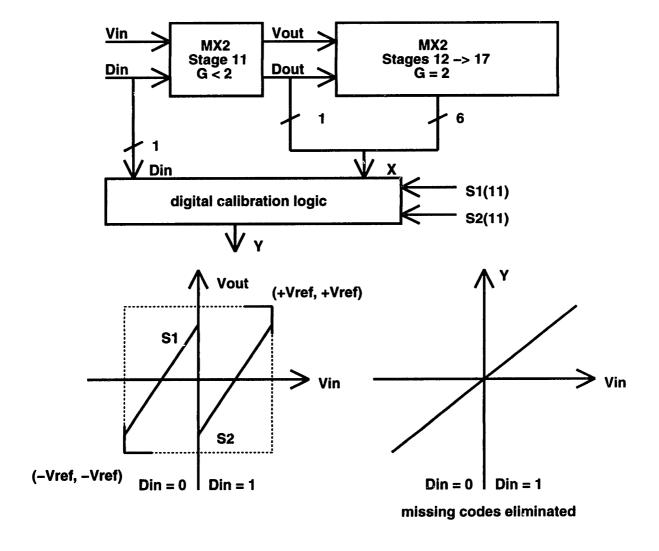


Figure 4.8: Pipeline ADC with digital calibration applied to the eleventh stage.

| V_{in} | V_{out} | D_{in} | X | $D_{in}:X$ | |
|------------|------------|----------|-----|------------|--|
| $+V_{ref}$ | $+V_{ref}$ | 1 | 127 | 255 | |
| • | • | 1 | 126 | 254 | |
| • | • | • | • | • | |
| • | • | 1 | 1 | 129 | |
| 0 | $-V_{ref}$ | 1 | 0 | 128 | |
| 0 | $+V_{ref}$ | 0 | 127 | 127 | |
| • | • | 0 | 126 | 126 | |
| • | • | • | • | • | |
| • | | 0 | 1 | 1 | |
| $-V_{ref}$ | $-V_{ref}$ | 0 | 0 | 0 | |

Table 4.1: Parameters of a pipeline section as a function of V_{in} . $G=2,\,K=6$.

4.4.2 Derivation of Algorithm

One of the first observations to be made is to examine the various quantities such as D_{in} , X, D_{in} : X and Y as a function of V_{in} , as appropriate.

Pipeline Section with MX2 Stage G=2

Table 4.1 illustrates the output parameters of an ideal pipeline section with G=2 as a function of V_{in} . This Table corresponds to Fig. 4.5 with G=2 and K=6. Notice that X repeats for $D_{in}=0$ and $D_{in}=1$. This is because X represents the quantization of the residue V_{out} . The key point is to notice that the output $D_{in}: X$ has no missing codes and counts from 0 to $2^8-1=255$.

Pipeline Section with MX2 Stage G < 2

Table 4.2 illustrates the output parameters of a pipeline section with MX2 Stage G < 2 as a function of V_{in} . This Table corresponds to Fig. 4.7 with G = 1.93 and K = 6. In this case, it is seen that X does not reach the maximum or minimum value possible because the gain is reduced in the leading MX2 stage. Thus $D_{in}: X$ has missing codes at the major carry transition point.

This observation is central to the digital self-calibration scheme. The quantity $D_{in}:X$ can be remapped to eliminate the missing codes without affecting decision levels. The interpretation of the output $D_{in}:X$ is that the MX2 stage structure joins the two halfs of the residue plot together. Thus, the decision level spacing is essentially not affected. The reduced gain of the MX2 stage means only that fewer than the maximum possible number of decision levels are accessed.

| V_{in} | V_{out} | D_{in} | X | $D_{in}:X$ |
|------------|-------------|----------|-----|------------|
| $+V_{ref}$ | $<+V_{ref}$ | 1 | 125 | 253 |
| • | • | 1 | 124 | 252 |
| • | • | • | • | • |
| • | • | 1 | 3 | 131 |
| 0 | $>-V_{ref}$ | 1 | 2 | 130 |
| 0 | $<+V_{ref}$ | 0 | 125 | 125 |
| • | • | 0 | 124 | 124 |
| • | • | • | • | • |
| • | • | 0 | 3 | 3 |
| $-V_{ref}$ | $>-V_{ref}$ | 0 | 2 | 2 |

Table 4.2: Parameters of a pipeline section as a function of V_{in} . G < 2, K = 6.

Pipeline Section with MX2 Stage G < 2 and Digital Calibration System

Table 4.3 illustrates the output parameters of a pipeline section with MX2 Stage G < 2 and digital self-calibration system employed as a function of V_{in} . This Table corresponds to Fig. 4.8 with G = 1.93 and K = 6. Notice that the calibrated output Y has no missing codes. This is performed by arranging the output Y to count with X for $D_{in} = 0$ and to continue counting for $D_{in} = 1$. This eliminates the missing codes.

4.4.3 Algorithm

The digital self-calibration algorithm can now be simply stated as:

$$Y = X, \quad if \quad D_{in} = 0 \tag{4.8}$$

$$Y = X + S_1 - S_2, if D_{in} = 1$$
 (4.9)

It is important to point out that no explicit digital multiplication is needed for calibration. Using the example shown in Table 4.3, it is seen that $S_1 = 125$ and that $S_2 = 2$. As a result, $S_1 - S_2 = 123$. With $D_{in} = 0$, the output Y counts from 2 to 125. When $D_{in} = 1$, the output Y becomes X + 123 which begins at 125. The output continues to 248. If desired, the offset can be removed at this stage by determining $S_0 = 2$ and subtracting S_0 from Y. This would result in at output ranging from 0 to 246.

The decision levels from a succeeding pipeline section are mapped from the residue output of a MX2, or equivalently the input to the pipeline section, to the input of the MX2 stage. Fig. 4.9 shows the case where the decision levels d(i) and d(i+1) were mapped from the V_{out} axis to the V_{in} axis of the MX2 stage in this manner. The ideal decision level spacing along the V_{in} axis is Δ . The decision level spacing and MX2 gain are exaggerated

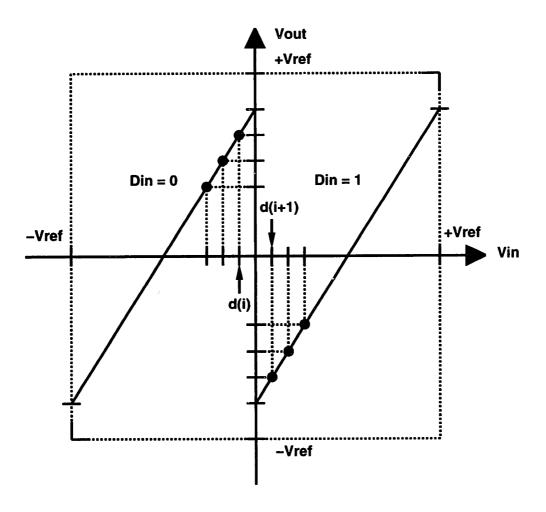


Figure 4.9: Residue plot of a MX2 stage in a 1-bit-per-stage radix < 2 pipeline ADC with $d(i+1)-d(i) \approx 2\Delta$.

| V_{in} | V_{out} | D_{in} | X | $D_{in}: X$ | Y |
|------------|-------------|----------|-----|-------------|-----|
| $+V_{ref}$ | $<+V_{ref}$ | 1 | 125 | 253 | 248 |
| • | • | 1 | 124 | 252 | 247 |
| • | • | • | | • | • |
| • | • | 1 | 3 | 131 | 126 |
| 0 | $>-V_{ref}$ | 1 | 2 | 130 | 125 |
| 0 | $<+V_{ref}$ | 0 | 125 | 125 | 125 |
| • | • | 0 | 124 | 124 | 124 |
| • | • | • | | • | • |
| • | • | 0 | 3 | 3 | 3 |
| $-V_{ref}$ | $>-V_{ref}$ | 0 | 2 | 2 | 2 |

Table 4.3: Parameters of a pipeline section as a function of V_{in} . Digital calibration applied. G < 2, K = 6.

for illustration purposes. It is assumed that the decision level spacings from the succeeding pipeline section are uniform. Also, it is assumed that the MX2 offset is zero so that the decision level magnitudes about the origin are equal:

$$|d(i+1)| = |d(i)| \tag{4.10}$$

This assumption is for determining the extreme cases. The decision level spacing across the major carry transition point is $d(i+1) - d(i) \approx 2\Delta$ for the case shown. If the gain of the MX2 stage were incremented a small amount, then new decision levels would be accessed and d(i+1) - d(i) would become approximately zero.

Fig. 4.10 shows the case where the decision level spacing $d(i+1)-d(i)\approx 0$. If the gain of the MX2 stage were decremented a small amount, then the decision levels d(i) and d(i+1) indicated would no longer be accessible and the new decision level spacing d(i+1)-d(i) would become approximately 2Δ .

The actual width depends on the precise gain and offset of the current MX2 stage and the decision level spacing of the succeeding pipeline stages. The average expected value of $d(i+1) - d(i) = \Delta$, the ideal decision level spacing. As a result, a single code should be mapped to this range, as indicated above. The digital self-calibration algorithm maps the interval d(i+1) - d(i) to a single code with an average decision level spacing of Δ . For instance, Table 4.3 illustrates how code 125 spans the major carry transition crossing, as desired. This property of the digital self-calibration algorithm is maintained in the presence of comparator offset.

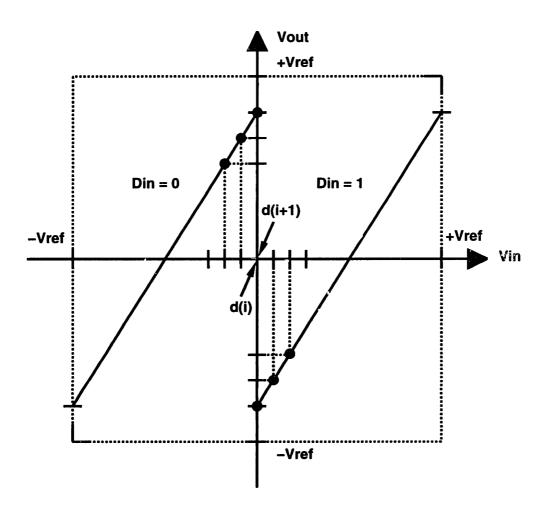


Figure 4.10: Residue plot of a MX2 stage in a 1-bit-per-stage pipeline ADC with $d(i+1) - d(i) \approx 0$.

4.5 Comparator Offset

As discussed above, comparator offset of a traditional pipeline ADC is a primary limitation to accuracy. The digital calibration is tolerant to comparator offset. Fig. 4.11 shows a residue plot of a MX2 stage with comparator offset from the previous stage. For the first MX2 stage in the pipeline ADC, the comparator offset may originate from the comparator in the SHA stage. The key feature of this residue plot is the equation:

$$S_1' - S_2' = S_1 - S_2 \tag{4.11}$$

What this means is that the difference in the constants $S_1 - S_2$ does not change so that the calibrated residue output is unchanged when there is comparator offset from any cause, whether static or dynamic in nature [58]. This remains true as long as the effective comparator offset maintains the residue within the reference boundary box.

This is an important result that distinguishes the digital self-calibration presented in this thesis from other digital calibration schemes applied to pipeline ADCs.

4.6 Pipeline Synthesis

The pipeline synthesis procedure is derived from the property that any contiguous pipeline section within a pipeline ADC is an ADC itself. The calibrated K+1 stage pipeline section described up to this point is now used as an ADC to measure the next preceding MX2 stage. The number of decision levels is almost again doubled. The synthesis procedure is repeated in the manner described above until all of the pipeline stages are calibrated. Fig. 4.12 illustrates the calibration being applied to stage 10 with K=6. The digital self-calibration aligns the transfer characteristic arising from the K+1 stage pipeline section at the major carry transition point for the K+2 stage pipeline. Notice that viewing the pipeline from a given stage, the decision level spacing is derived from the succeeding pipeline section since the digital self-calibration ensures that the transfer characteristic is connected together correctly.

The total number of decision levels accessible in general is F:

$$F \approx 2 \prod_{i=1}^{N-1} G(i) \tag{4.12}$$

where G(i) is the gain of MX2 stage i and N accounts for the total number of MX2 stages and the SHA stage. It is important to notice that the digital self-calibration permits G(i) to be different among the MX2 stages. This remains true as long as G(i), for a given MX2 stage i, is reduced sufficiently so that missing decision levels do not occur. In the simplified case where the MX2 stages in the K stage pipeline section each have gain of 2 and the MX2 stages to be calibrated each have gain of G, the total number of decision levels accessible is described as:

$$F \approx 2G^{N-K-1}2^K \tag{4.13}$$

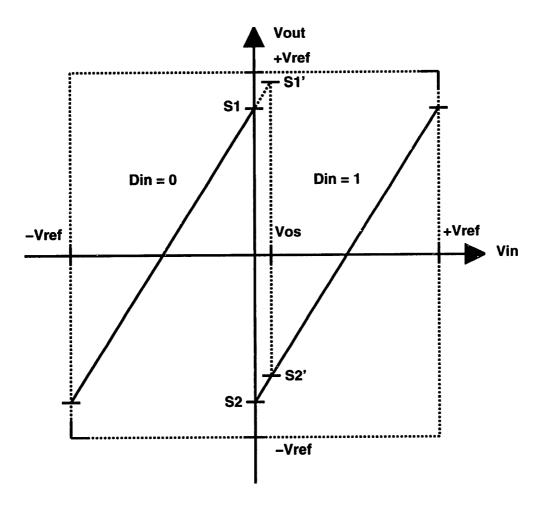


Figure 4.11: Residue plot of a MX2 stage with comparator offset V_{OS} .

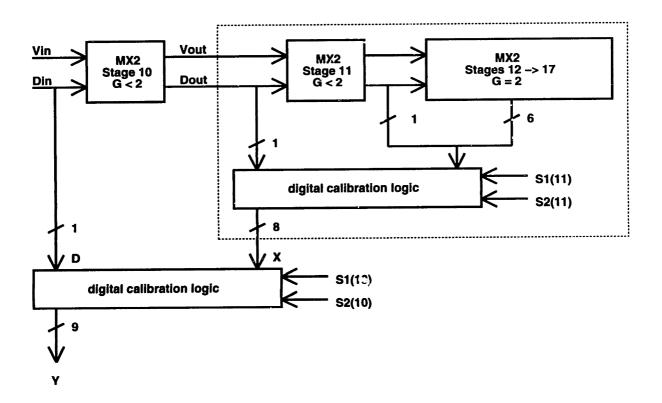


Figure 4.12: Digital calibration of higher level stages.

The synthesis procedure can be used to synthesize a radix < 2 pipeline with K = 1, if desired. Since fewer decision levels result in this case, it is desirable to use as large a value of K as is feasible. In this thesis, N = 18, K = 6 and G = 1.93 resulting in $F \approx 177150$.

4.7 Determination of the Calibration Constants S_1 and S_2

The calibration constants S_1 and S_2 are each obtained by operating a pipeline section with specific inputs forced at the input of the section. In order to obtain S_1 for a particular stage, the analog input is grounded and the input bit is forced to be 0 and the residue from the stage is quantized using the successive pipeline stages. Fig. 4.13 shows how calibration constant S_1 is obtained. In order to obtain S_2 for a particular stage, the analog input is grounded and the input bit is forced to be 1 and the residue from the stage is quantized uning the successive pipeline stages. Fig. 4.14 shows how calibration constant S_2 is obtained. The digital data output is averaged without truncation to avoid truncation errors. Since only the difference $S_1 - S_2$ is of importance, the analog input during measurement of S_1 and S_2 does not have to be precisely ground. The key is that S_1 and S_2 are measured with the same input voltage near the comparator threshold so that overranging of the succeeding pipeline stages does not result.

Typically, S_1 and S_2 are determined during a calibration mode. Once determined, the ADC is then used in the run mode. Since the calibration constants are averaged quantities, it is possible to determine samples of S_1 and S_2 continuously with interruption in the sampling periodicity. Several analog input samples can be skipped to permit determination of one sample of a calibration constant. Over a long period of time, a long term average of S_1 and S_2 for each stage can be assembled to maintain calibration over varying conditions.

Once determined, the calibration constants are stored in a digital memory. Since only S_1-S_2 is needed in the digital calibration algorithm, a 132-bit memory suffices for a pipeline ADC with one SHA stage and 17 MX2 stages.

4.8 Extension to Multi-Bit-per-Stage Pipeline ADCs

The digital self-calibration was presented from the standpoint of a 1-bit-per-stage implementation. The concept can be extended to multi-bit-per-stage pipeline ADCs. The concept is based on providing additional calibration constants $S_1, S_2, S_3, S_4, \dots, S_k, S_{k+1}$ such that differences $S_1 - S_2, S_3 - S_4, S_5 - S_6, \dots S_k - S_{k+1}$ are maintained when sub-ADC decision levels or when sub-DAC reconstruction levels are different from ideal.

Fig. 4.15 shows the residue plot of a 2-bit-per-stage MX4 stage with sub-ADC and sub-DAC errors. The sub-ADC error is caused by a decision level location error in the sub-ADC. The sub-DAC error is caused by a reconstruction level error in the sub-DAC. As an example, comparator error can contribute to sub-ADC error and a nonuniform resistor string used to assign reconstruction levels can contribute sub-DAC error. In Fig. 4.15, $S'_1 - S'_2 = S_1 - S_2$, $S'_3 - S'_4 = S_3 - S_4$ and $S'_5 - S'_6 = S_5 - S_6$.

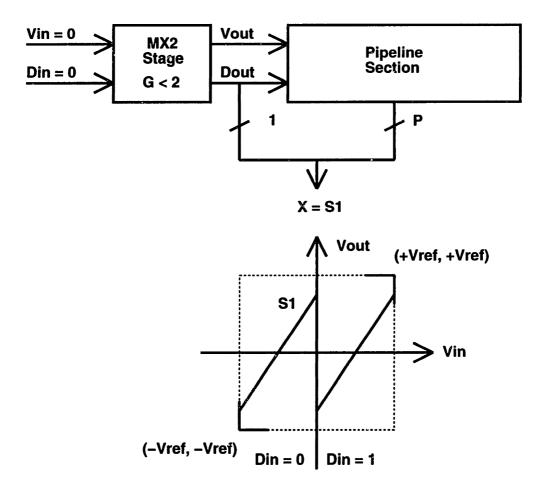


Figure 4.13: Determination of calibration constant S_1 .

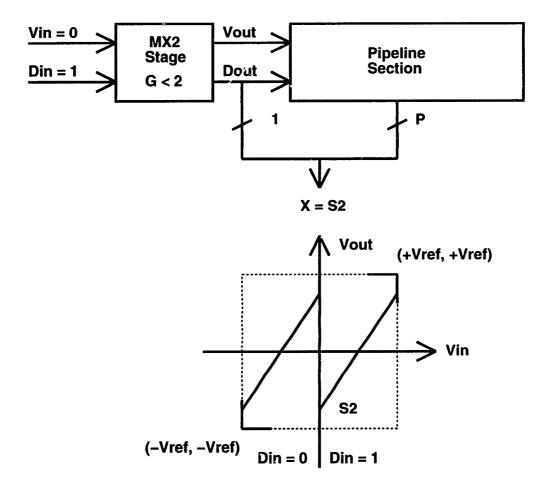


Figure 4.14: Determination of calibration constant S_2 .

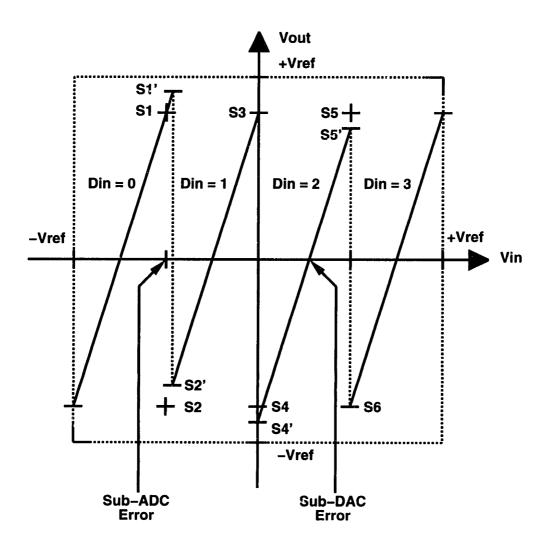


Figure 4.15: Residue plot of a 2-bit-per-stage MX4 stage. Sub-ADC and Sub-DAC errors are indicated.

4.9 ADC Digital Output Reduction

The digital self-calibration does not create or eliminate decision levels. Thus, the decision level spacing from the calibrated ADC must be no wider than 1 LSB at the intended resolution. By using a redundant number of stages, this can be guaranteed. In some cases, reduction of the redundancy in the digital output data is desired. By reducing the number of output codes produced, fewer corresponding decision levels are accessed. The final ADC resolution is lower but the linearity of the ADC at the lower resolution improves. The linearity of the ADC at the original resolution is not changed.

The concept of digital output reduction is illustrated in Fig. 4.16. A reduction factor of L=4 is shown. The ideal unreduced and actual unreduced decision level locations are shown first. The actual unreduced decision level locations are characterized by DNL(k). The ideal reduced and actual reduced decision level locations are shown next in Fig. 4.16. Notice how the actual reduced decision level spacing is more regular than the actual unreduced decision level spacing. The actual reduced decision level locations are characterized by DNL'(k). The quantities DNL'(k) and INL'(k) are related to DNL(k) and INL(k) as follows:

$$DNL'(k) = \frac{1}{L} \sum_{i=kL}^{(k+1)L-1} DNL(i)$$
 (4.14)

$$INL'(k) = \frac{1}{L}INL((k+1)L-1)$$
 (4.15)

Notice that DNL'(k) averages DNL(k) over L codes. The quantity INL'(k) reduces the peak value of INL(k) by a factor of L. It is not necessary for the reduction factor L to be a power-of-2.

4.10 Properties of Digital Self-Calibration

This thesis outlines a digital self-calibration technique that establishes a general method for digital self-calibration of high-accuracy, high-speed pipeline subranging ADCs. The thesis addresses the traditional limitations of high-resolution ADCs including ratiometric and offset errors, among others, that in arise in monolithic IC processes. The technique presents a novel pipeline architecture, not described in previous techniques, which disconnects the pipeline at a stage to be calibrated and permits special calibration inputs to be applied to the selected stage input. By first applying the calibration inputs at the input of an end section of the pipeline, a calibrated pipeline section is synthesized which in turn is used with the preceding pipeline stage to synthesize a higher resolution calibrated pipeline section. This hierarchical process is repeated until the beginning of the pipeline is reached, and thus the entire pipeline is calibrated. The result is an efficient digital self-calibration algorithm. The digital self-calibration can be applied to an arbitrary number of stages, each resolving an arbitrary number of bits. With a certain resolution ADC of any type provided, subranging stages can precede this ADC and then digital self-calibration can be used to arrive at a higher resolution ADC. The technique is amenable to cyclic ADCs as

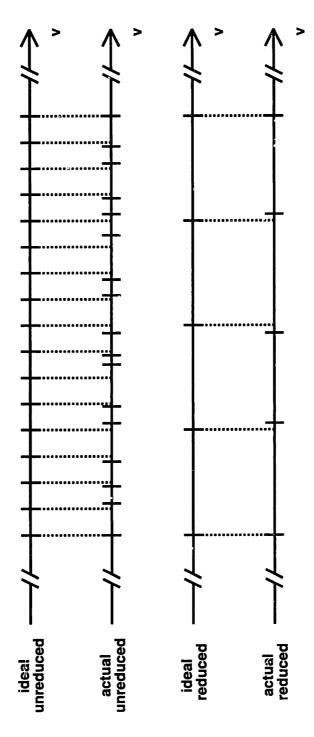


Figure 4.16: Comparison of decision level locations for the cases of unreduced and reduced ADC digital output. Ideal and actual locations are shown for the respective cases.

well as pipeline ADC structures. Cyclic ADCs tradeoff hardware reduction with decreased throughput relative to a pipeline. Thus, the advantage of cyclic ADCs is that they are small but require more clock cycles for a conversion. The digital self-calibration hardware scales down with cyclic ADC hardware reduction. Thus, the digital self-calibration can be used with a pipeline, cyclic or combination of these architectures with other ADC architectures to attain high-resolution ADCs at a wide range of conversion speeds.

The articulation of fundamental calibration principles of pipeline architectures permits direct synthesis of the digital self-calibration algorithm. One of the major outcomes of the articulation of these principles is that the pipeline converter itself is used in the calibration mode. Thus, no external reference data converters are needed in the technique. No highprecision components are required. The digital self-calibration can attain 15-bits of linearity from component matching no better than 7-bits. Thus, the scheme represents a true selfcalibration. The errors are directly measured under the same condition as during the run mode. The calibration data is obtained by use of a simple set of d-c calibration inputs that are indistinguishable from a previous stage output. The calibration data is obtained with the pipeline ADC operating at full speed. A small digital memory is used to store the calibration data. The calibration data is combined with the run mode raw data, in accordance with the digital self-calibration algorithm, to arrive at the calibrated output. The digital self-calibration performs a simple calibration algorithm in the digital domain alone. As a result of the algorithm simplicity, no multipliers, microprocessors or other complex arithmetic systems are required. Since the calibration algorithm is all digital, no extra analog hardware is required for calibration mode or run mode.

Chapter 5

Digital Self-Calibration Simulator

5.1 Software Strategy

A simulator program was developed in order to test various digital self-calibration algorithms and to specifically explore the radix < 2 pipeline ADC algorithm. An important feature of this program is that it is modular in nature. As a result, the program modules can also be used for the test system program used during evaluation of the pipeline ADC IC. In the case of the test system program, the core of the simulator program is used for the digital self-calibration evaluation with measured calibration data and measured run mode data provided from the actual ADC. An advantage of this program approach is that software development and debugging effort for the simulator and test system program is concurrent. This is an important strategy for minimizing test system development time and resources.

5.2 Simulator Structure

The goal of the simulator program is to provide a tool that can be used to determine the effectiveness of different digital self-calibration techniques. The present simulator accommodates a 1-bit-per-stage pipeline ADC design. Each stage of the pipeline is essentially modeled using a function. This permits each stage to be called for a result in a modular manner. For example, to determine a conversion in simulation, a SHA function call and N-1 MX2 function calls are needed for a total of N bits. A compact program can be devised by taking advantage of recursion present in the digital self-calibration synthesis procedure. However, the resulting code is then more complex and more difficult to modify. The simulator program here is structured so that the stages of the pipeline can be used in any desired order through simple modification of the code. As a result, the present simulator program is not optimized for minimum code size.

5.3 Simulator Description

The digital self-calibration scheme synthesizes the pipeline ADC by calibrating a given stage with the pipeline section that follows. This is a recursive concept which can be easily accommodated in the software using function calls. First, the calibration constants are determined for the entire pipeline ADC. Next, an analog input signal is presented to the input of the ADC. The analog input is then processed by each stage of the pipeline as the analog data propagates down the pipeline. As this occurs, the raw digital data is determined along the way. When the analog wavefront reaches the end of the pipeline, then the digital self-calibration algorithm, and the calibration constants determined above, can be used to determine the calibrated output beginning with the last pipeline section and then propagating back towards the SHA.

The analog signals are modeled using floating point variables and the digital signals are modeled using integer variables. The main simulator program output is DNL and INL. This data is determined by scanning the input with an analog ramp that increments by an amount much smaller than the LSB of the converter under consideration. This is important to make sure that the simulator program captures the correct decision level locations.

Various pipeline ADC nonidealities are modeled. These include primarily gain and offset error for the SHA and MX2 amplifiers and offset for the comparators. If desired, nonlinearity can also be modeled in the SHA and MX2 stages.

As mentioned above, the number of calibrated output codes is not necessarily a power-of-2. The actual ADC output can be determined by a simple division of the ADC output by a reduction factor. In the case where the reduction factor is a power-of-2, the division is especially simple to perform. In this case, a right-shift of the digital output data, by the appropriate number of bits, would perform the digital output data reduction. It is important to note that since many signal processing algorithms are not sensitive to a scale factor, digital output reduction may not be necessary.

The digital self-calibration simulator C code programs are shown in Appendix C.

5.4 Simulation Results

A pipeline ADC with 1 SHA stage, 11 G < 2 MX2 stages and 6 G = 2 MX2 stages was simulated for various values of G. The simulator was run to determine the DNL and INL using the digital self-calibration technique. For the cases shown, the DNL is within ± 0.25 LSB at 15-bits and the INL is within ± 0.75 LSB at 15-bits. The DNL and INL are plotted on the same respective scales for comparison. As G is decreased, the number of decision levels available decreases. No missing codes are present before the data reduction is applied. The output reduction factor is L = 5 for all of the cases shown.

5.4.1 MX2 Stage Gain G = 1.9704

Fig. 5.1 shows the DNL after calibration and Fig. 5.2 shows the INL after calibration for G = 1.9704. Notice that the DNL appears to have mostly negative excursions. The decision

levels near the major carry transition point for a MX2 stage are relatively close together, compared to 1 LSB, resulting in the negative DNL excursions. It should be pointed out that these negative excursions are balanced by positive DNL over longer calibrated output code duration. As a result, the INL does not deviate excessively.

5.4.2 MX2 Stage Gain G = 1.9550

Fig. 5.3 shows the DNL after calibration and Fig. 5.4 shows the INL after calibration for G=1.9550. Notice that the DNL appears more balanced in this case. The decision levels near the major carry transition point for a MX2 stage are nearly 1 LSB apart in this case.

5.4.3 MX2 Stage Gain G = 1.9324

Fig. 5.5 shows the DNL after calibration and Fig. 5.6 shows the INL after calibration for G=1.9324. Notice that the DNL appears to have mostly positive excursions. The decision levels near the major carry transition point for a MX2 stage are relatively far apart, compared to 1 LSB, resulting in the positive DNL excursions. It should be pointed out that these positive excursions are balanced by negative DNL over longer calibrated output code duration. As a result, the INL does not deviate excessively.

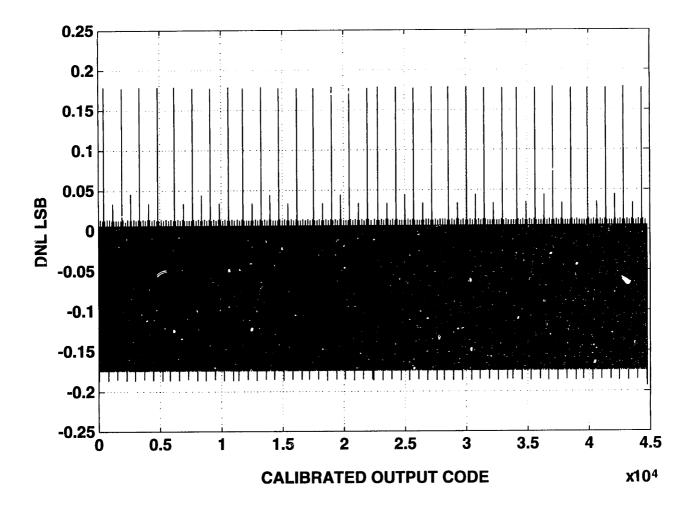


Figure 5.1: Simulated DNL, G = 1.9704, K = 6.

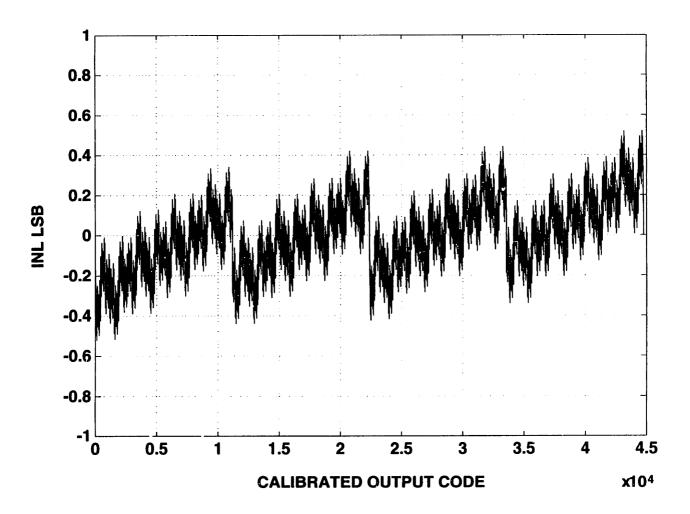


Figure 5.2: Simulated INL, G = 1.9704, K = 6.

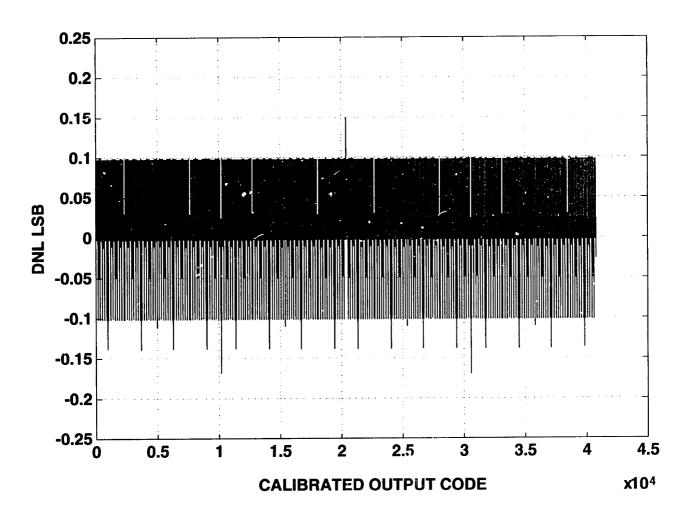


Figure 5.3: Simulated DNL, G = 1.9550, K = 6.

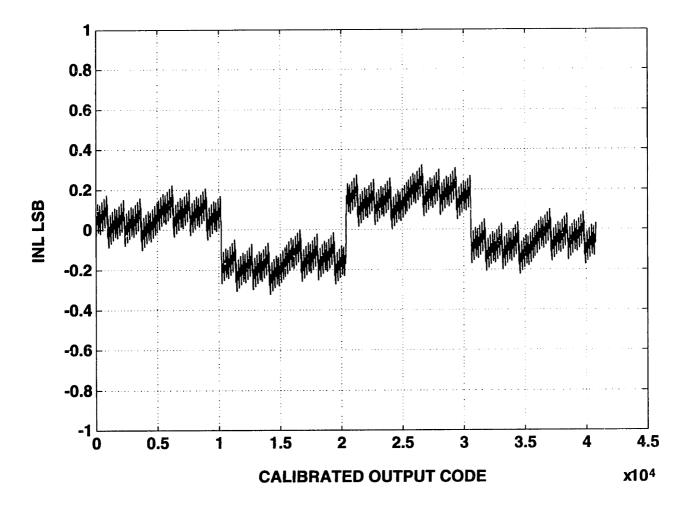


Figure 5.4: Simulated INL, G = 1.9550, K = 6.

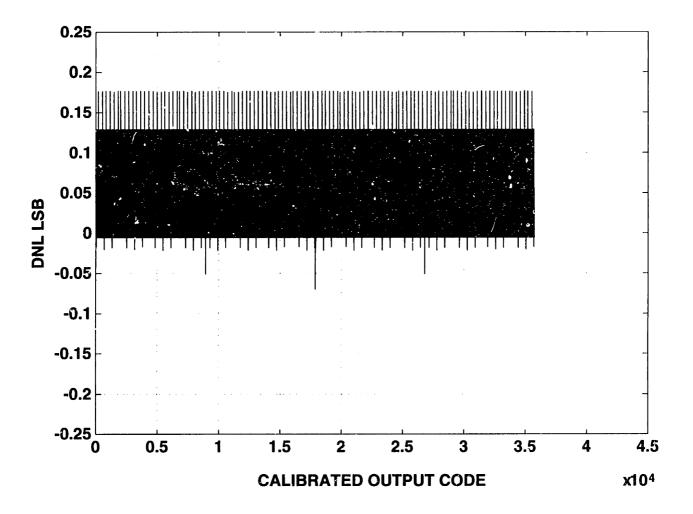


Figure 5.5: Simulated DNL, G = 1.9324, K = 6.

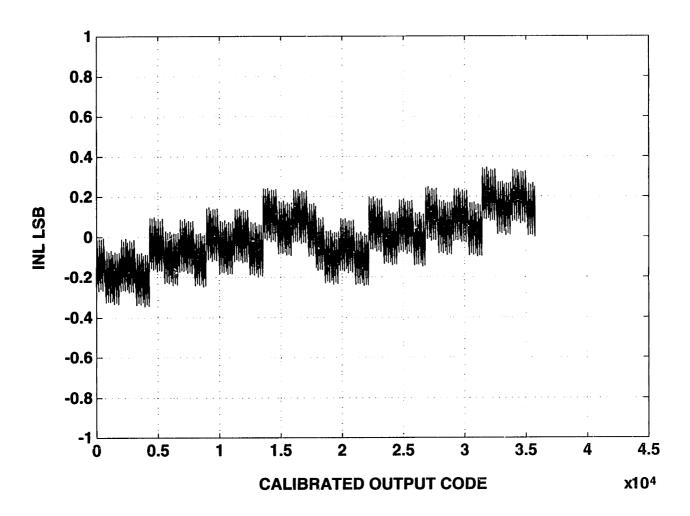


Figure 5.6: Simulated INL, G = 1.9324, K = 6.

Chapter 6

Switched-Capacitor System Design

This chapter will outline the SC system design for the 1-bit-per-stage digitally self-calibrated pipeline ADC. SC techniques are used in combination with MOS based technologies. This permits the synthesis of low cost systems for a variety of purposes. The use of BiCMOS provides additional design capability for building high-performance SC systems. Fully differential circuit topology is used to permit good high-frequency power supply rejection in addition to improving the dynamic range.

6.1 SC Pipeline ADC

6.1.1 Basic Structure

The SC pipeline ADC is shown in Fig. 6.1. As discussed above, the pipeline ADC is a cascade of processing stages. The pipeline begins with a SHA stage that is then followed by a succession of N-K-1, G<2 MX2 stages and K, G=2 MX2 stages. The analog differential output of a MX2 stage is connected to the analog differential input of a succeeding stage. In addition, the digital output of a stage is connected to the digital input of a succeeding stage. As can be seen from Fig. 6.1, provision for disconnecting the pipeline at the input of a G < 2 MX2 stage is present. This permits calibration of each G < 2 MX2 stage, as desired. When a stage is to be calibrated, the differential analog signal V_{p1} and V_{p2} is connected to the input of the desired stage. As described above, the necessary analog signal needed for calibration is ground. Thus, $V_{p1} = 0$ and $V_{p2} = 0$ are used. The digital input D_{p1} is connected to the stage to be calibrated to force the input bit of the selected stage. In general, provision of V_{p1} and V_{p2} as potentials different from ground permits determination of other calibration constants, namely S_0 and S_3 , when the program input difference is $-V_{ref}$ or $+V_{ref}$, respectively. Since only one stage is calibrated at one time, the signals V_{p1} , V_{p2} and D_{p1} are bussed. The provision of external V_{p1} and V_{p2} , or program inputs, provides flexibility for testing. Although the last section of G = 2 MX2pipeline is accessible at each stage input, the calibration data for these stages is not used. As discussed above, the digital calibration can synthesize a calibrated pipeline by beginning from the last stage of the pipeline, if desired.

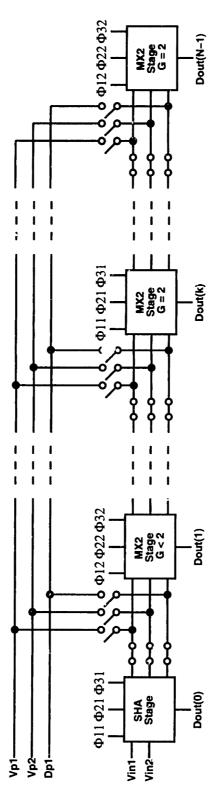


Figure 6.1: SC Pipeline ADC.

6.1.2 Timing

Fig. 6.2 shows the timing diagram for the pipeline ADC. Notice that the pipeline timing relies essentially on two nonoverlapping clocks ϕ_{11} and ϕ_{31} . There is an even and odd set of clocks used. Also, there is logical redundancy in the clocks shown, namely that $\phi_{11} = \phi_{32}$ and $\phi_{31} = \phi_{12}$. The redundancy is present because the clocks shown are independent, which is important for testing flexibility. The clocks ϕ_{11} , ϕ_{32} and ϕ_{31} , ϕ_{12} can be merged, respectively, to reduce the total number of clocks, but at the expense of testing flexibility. The basic timing of the pipeline ADC is now described. The analog input is connected to the SHA input capacitors during the time that ϕ_{11} is high. The input is sampled on the high-to-low transition of ϕ_{11} . A delayed clock ϕ_{21} is used to minimize charge injection in the SHA and will be discussed further later. When the clock ϕ_{31} is high, the SHA is put into the amplify mode to hold the sampled analog input. During this time, ϕ_{12} is high as the first MX2 stage samples the SHA output. A delayed clock ϕ_{22} is used to minimize charge injection in the MX2 and will be discussed further later. When the clock ϕ_{32} is high, the MX2 stage is put into the amplify mode. A analog input is again presented to the SHA when ϕ_{11} becomes high again. As the clocks progress, the analog inputs presented to the SHA stage propagate down the pipeline.

6.2 SC SHA

6.2.1 Sample Mode

The SHA employs SC networks, a fully differential op-amp and a fully differential comparator. Fig. 6.3 shows the SC SHA in the sample mode. The top plates of capacitors C_1 and C_2 are connected to the op-amp inputs. The SHA function begins by sampling the analog input on to the bottom plates of the capacitors during the sample mode. During this mode, the op-amp is configured in a unity-gain connection. The differential input charge sampled is:

$$Q_i = -(C_1 + C_2)(V_{in} - V_{os}) + C_3 V_{os}$$
(6.1)

Thus, when ϕ_1 goes low, the analog input is sampled. The sampling occurs with a constant gate-source voltage applied to the switch because the top plate of the capacitors are connected to the op-amp which is in the unity gain feedback connection during the sample mode. This is enforced by having ϕ_1 go low first. Once this is accomplished, the top-plate capacitor charge cannot be changed as this node is a cut set of capacitors. The clock ϕ_2 then disconnects the capacitors from the analog input.

6.2.2 Amplify Mode

Once the analog input is sampled, the SHA function continues to the amplify mode. Fig. 6.4 shows the SHA in the amplify mode. The capacitors are placed around the op-amp during this mode resulting in a held output. This occurs when ϕ_3 is high. The charge for

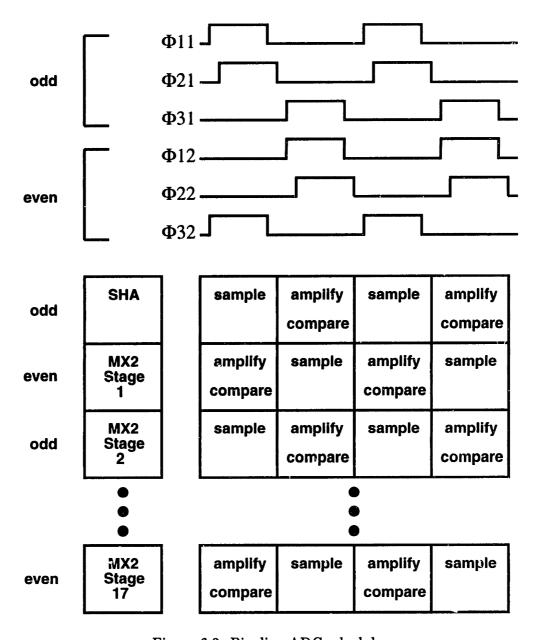


Figure 6.2: Pipeline ADC schedule.

the amplify mode is written as:

$$Q_{f} = -(C_{1} + C_{2} + C_{3})(V_{out} - V_{os})$$
(6.2)

The capacitors C_1 and C_2 are nominally equal. The mismatch is represented through a parameter α_1 such that:

$$C_2 = (1 + \alpha_1)C_1 \tag{6.3}$$

The capacitor C_3 is a small fraction of either C_1 or C_2 and is expressed as:

$$C_3 = \alpha_2 C_1 \tag{6.4}$$

The purpose of the capacitor C_3 is to obtain a gain that is slightly less than 1. The purpose of the SHA gain slightly less than 1 is to prevent the SHA output from overranging the succeeding stages so that the SHA can be calibrated. In this design, $\alpha_2 = 0.035$.

Using charge conservation, $Q_i = Q_f$. The presence of charge injection must be accounted for so the charge balance is then written as:

$$Q_i - Q_f = Q_e \tag{6.5}$$

As a result, the output V_{out} can be solved in terms of V_{in} and the SHA closed loop gain is found to be:

$$V_{out} = \frac{V_{in}}{1 + C_3/(C_1 + C_2)} + \frac{Q_e}{C_1 + C_2 + C_3}$$
 (6.6)

Using the expression for C_2 and C_3 in terms of C_1 , it is seen that:

$$V_{out} = \frac{V_{in}}{1 + \alpha_2/(2 + \alpha_1)} + \frac{Q_e}{C_1(2 + \alpha_1 + \alpha_2)}$$
(6.7)

During the amplify mode, the comparator is connected to the op-amp output. At the end of the amplify mode, the comparator will make a decision as to the sign of the op-amp output. The analog data and the digital data is presented to the succeeding stage at this time. When the analog input is being sampled to the SHA, the comparator is put into the reset mode.

It is important that the residual offset is minimized to the point where the residue of a MX2 stage to be calibrated does not exceed the reference boundary. The extrema of the residue plot occur when $V_{in} = -V_{ref}$ and $V_{in} = V_{ref}$. The extrema are denoted respectively as S_0 and S_3 :

$$S_0 = \frac{-V_{ref}(2+\alpha_1)}{2+\alpha_1+\alpha_2} + \frac{-Q_e/C_1}{2+\alpha_1+\alpha_2}$$
 (6.8)

$$S_3 = \frac{V_{ref}(2+\alpha_1)}{2+\alpha_1+\alpha_2} + \frac{-Q_e/C_1}{2+\alpha_1+\alpha_2}$$
 (6.9)

The digital self-calibration requires that the extrema of the transfer characteristic to be within $\pm V_{ref}$:

$$S_0 > -V_{ref} \tag{6.10}$$

$$S_3 < +V_{ref} \tag{6.11}$$

6.3. SC MX2

The constraints can now be expressed as:

$$Q_e > -C_1 V_{ref} \alpha_2 \tag{6.12}$$

$$Q_e < C_1 V_{ref} \alpha_2 \tag{6.13}$$

Combining the constraints above, a constraint can be simply written as:

$$|Q_e/C_1| < \alpha_2 V_{ref} \tag{6.14}$$

6.3 SC MX2

6.3.1 Sample Mode

The MX2 employs SC networks, a fully differential op-amp and a fully differential comparator. Fig. 6.5 shows the SC MX2 in the sample mode. The MX2 function begins by sampling the analog input on to the bottom plates of one of the capacitors during the sample mode. During this mode, the op-amp is configured in a unity-gain connection. The differential input charge sampled is:

$$Q_{i} = -(C_{1} + C_{2})(V_{in} - V_{os}) + C_{3}V_{os}$$
(6.15)

Thus, when ϕ_1 goes low, the analog input is sampled. The sampling occurs with a constant gate-source voltage applied to the switch because the top plate of the capacitors are connected to the op-amp which is in the unity gain feedback connection during the sample mode. This is enforced by having ϕ_1 go low first, as before. Once this is accomplished, the top-plate capacitor charge cannot be changed as this node is a cut set of capacitors. The clock ϕ_2 then disconnects the capacitors from the analog input.

6.3.2 Amplify Mode

Once the analog input is sampled, the MX2 function continues to the amplify mode. Fig. 6.6 shows the MX2 in the amplify mode. One capacitor is placed around the op-amp during this mode resulting in a held output. The other capacitor is connected to either $+V_{ref}$ or $-V_{ref}$, depending on the previous bit. The amplify mode thus performs the gain function and 1-bit DAC function needed. This occurs when ϕ_3 is high. The charge for the amplify mode is written as:

$$Q_f = -C_1 V_{ref} - (C_2 + C_3)(V_{out} - V_{os})$$
(6.16)

The capacitors C_1 and C_2 are nominally equal. The mismatch is represented through a parameter α_1 such that:

$$C_2 = (1 + \alpha_1)C_1 \tag{6.17}$$

The capacitor C_3 is a small fraction of either C_1 or C_2 and is expressed as:

$$C_3 = \alpha_2 C_1 \tag{6.18}$$

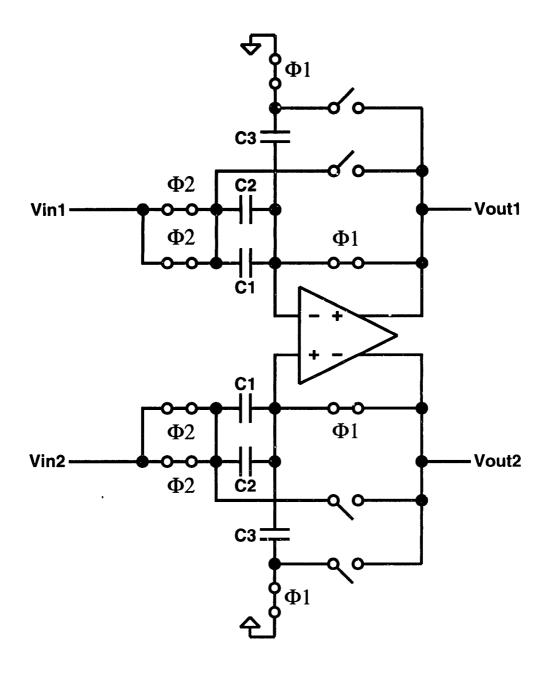


Figure 6.3: SHA in the sample mode.

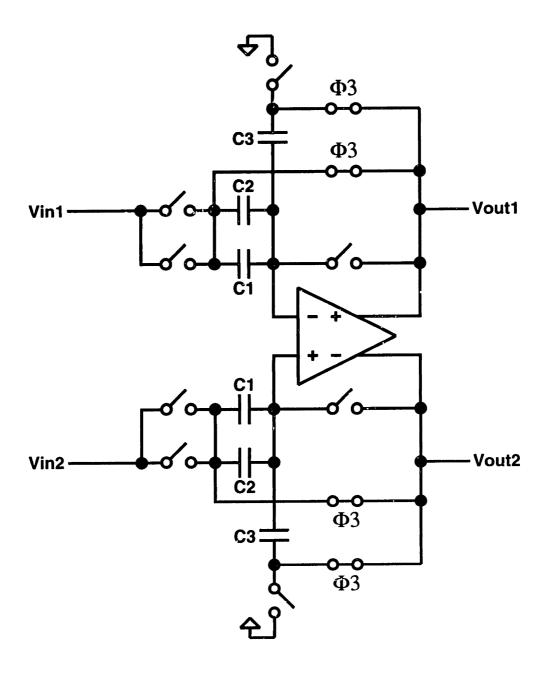


Figure 6.4: SHA in the amplify mode.

The purpose of the capacitor C_3 is to obtain a gain that is slightly less than 2 required in the radix < 2 pipeline ADC. In this design, $\alpha_2 = 0.035$.

Using charge conservation, $Q_i = Q_f$. The presence of charge injection must be accounted for so the charge balance is then written as:

$$Q_i - Q_f = Q_e \tag{6.19}$$

As a result, the output V_{out} can be solved in terms of V_{in} and V_{ref} using the expression for C_2 and C_3 in terms of C_1 :

$$V_{out} = \frac{(2 + \alpha_1)V_{in} - V_{ref}}{1 + \alpha_1 + \alpha_2} + \frac{Q_e}{C_1(1 + \alpha_1 + \alpha_2)}$$
(6.20)

During the amplify mode, the comparator is connected to the op-amp output. At the end of the amplify mode, the comparator will make a decision as to the sign of the op-amp output. The analog data and the digital data is presented to the succeeding stage at this time. When the analog input is being sampled to the SHA, the comparator is put into the reset mode.

It should be pointed out that the G < 2 MX2 amplifier can be easily modified to a G = 2 MX2 amplifier by disconnecting C_3 .

It is important that the residual offset is minimized to the point where the residue of a MX2 stage to be calibrated does not exceed the reference boundary. The extrema of the residue plot occur when $V_{in} = -V_{ref}$, $D_{in} = 0$; $V_{in} = 0$, $D_{in} = 0$; $V_{in} = 0$, $D_{in} = 1$; $V_{in} = V_{ref}$, $D_{in} = 1$. The extrema are denoted respectively as S_0 , S_1 , S_2 and S_3 :

$$S_0 = \frac{Q_e/C_1 - V_{ref}(1 + \alpha_1)}{1 + \alpha_1 + \alpha_2}$$
 (6.21)

$$S_1 = \frac{Q_e/C_1 + V_{ref}}{1 + \alpha_1 + \alpha_2} \tag{6.22}$$

$$S_2 = \frac{Q_e/C_1 - V_{ref}}{1 + \alpha_1 + \alpha_2} \tag{6.23}$$

$$S_3 = \frac{Q_e/C_1 + V_{ref}(1 + \alpha_1)}{1 + \alpha_1 + \alpha_2}$$
 (6.24)

The digital self-calibration requires that the extrema of the transfer characteristic to be within $\pm V_{ref}$:

$$S_0 > -V_{ref} \tag{6.25}$$

$$S_1 < +V_{ref} \tag{6.26}$$

$$S_2 > -V_{ref} \tag{6.27}$$

$$S_3 < +V_{ref} \tag{6.28}$$

The constraints can now be expressed as:

$$-C_1 V_{ref} \alpha_2 < Q_e < C_1 V_{ref} (\alpha_1 + \alpha_2)$$

$$\tag{6.29}$$

$$-C_1 V_{ref}(\alpha_1 + \alpha_2) < Q_e < C_1 V_{ref} \alpha_2 \tag{6.30}$$

Notice that $\alpha_2 \geq 0$ in all cases while α_1 can be zero or nonzero. It is needed to have $\alpha_1 + \alpha_2 \geq 0$. This can be accomplished by choosing α_2 to be large enough to swamp out α_1 so that $\alpha_1 + \alpha_2$ remains positive. Combining the constraints above, a worst case constraint can be simply written as:

$$|Q_e/C_1| < V_{ref}(|\alpha_1| + \alpha_2) \tag{6.31}$$

6.4 Limitations to the SC SHA and MX2

6.4.1 Nonlinearity

Nonlinearity contributed by capacitors and transmission gates will contribute INL, but not DNL. This is because the digital self-calibration measures the pipeline ADC during the calibration mode under the same condition as during the run mode. As a result, the transfer characteristic is aligned at the maj carry transition point at each MX2 stage that is calibrated.

Voltage Coefficient of Capacitors

Voltage coefficient of capacitors in the SHA and MX2 stages is an important limitation to the INL of the ADC. Voltage coefficient from the capacitor occurs because of depletion layer depth changing due to applied potential for a given doping. In the case of a double polysilicon capacitor, there are two depletion layers, one for each capacitor plate, to be concerned with. One advantage of fully differential circuits is that even order terms of a power series expansion representing a nonlinearity can be canceled [59]. Since the third order, and higher, terms are typically much smaller in the representative power series expansion, linearity can be improved.

Transmission Gates

Another important source of nonlinearity is signal dependent charge injection. It is shown in [60, 61, 62] that the charge injection is a function of the device geometry, external impedance, and applied device potentials.

The sampling switch in the SHA is the feedback reset switch connected around the opamp. The op-amp reset switches must usually have a small aspect ratio in order to help minimize charge injection. However, decreasing the aspect ratio increases the impedance of the reset switch and this can degrade settling characteristics of the op-amp. It is important to reduce charge injection in an effort to not only reduce the offset as much as possible but to also lessen the magnitude of any nonlinear charge injection originating from the sampling switches. Although the charge injection is nominally canceled by virtue of the differential circuit operation, the charge injection from the positive branch is different from the negative branch when the ADC input voltage is significantly different from zero.

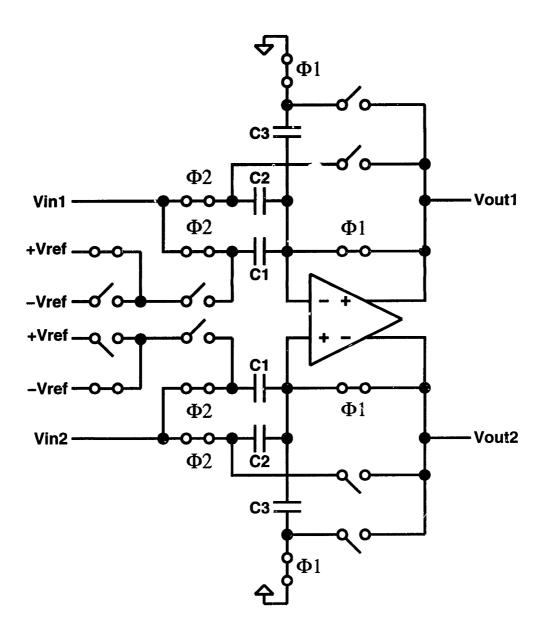


Figure 6.5: MX2 in the sample mode.

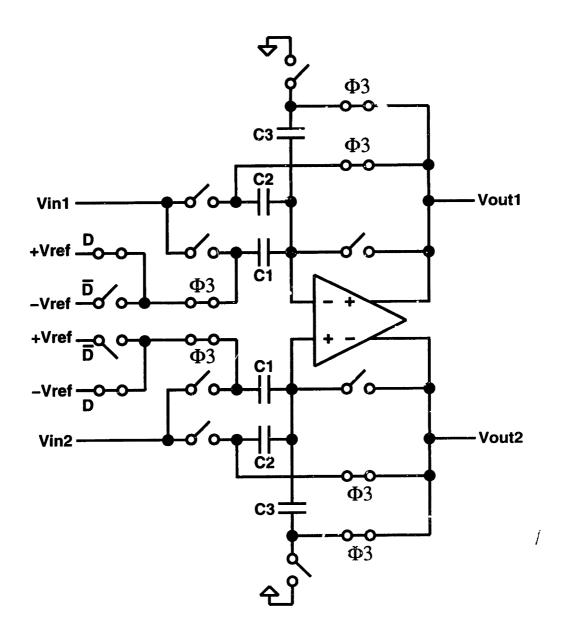


Figure 6.6: MX2 in the amplify mode.

The dominant external impedance connected to the MOS device terminals is the input capacitor and transmission gate branch. As a result, it is important that a large aspect ratio is used for nonsampling transmission gate switches which are used to access the switched capacitors. Large aspect ratio transmission gates provide low impedance.

Since the mobility and threshold voltages of NMOS and PMOS are usually different, the conductance in the ohmic region will differ for these devices when used in a transmission gate with a widely varying voltage. In one case, the conductance of the NMOS device will dominate while in the other case, the conductance of the PMOS device will dominate. Since the device parameters are different, it is important to scale the device widths such that the nominal impedance of a transmission gate is as small and constant as possible.

6.4.2 Residual Offset

The digital self-calibration removes the effects of amplifier offsets. It is important that the offset does not cause the residue output of a radix < 2 MX2 stage to exceed the reference boundary. An advantage of fully differential sampling is that offsets are reduced. In some cases, it is desirable to minimize the overall input referred offset of the ADC. Input referred offset may be canceled by applying zero ADC input and then digitally subtracting the result from the ADC output.

Charge Injection

Charge injection represents the dominant residual offset component. Offset in the SHA contributes to ADC input referred offset.

The MOS charge injection above was modeled as Q_{ϵ} . This parameter needs to now be discussed in terms of MOS device parameters. The channel charge of a transmission gate used for the op-amp reset switch can be written as:

$$Q_{trg} = |(WL)_{trn,eff}C_{ox}(V_{gn} - V_t)| + |(WL)_{trp,eff}C_{ox}(V_{gp} - V_t)|$$

$$(6.32)$$

The source potential is assumed to be zero in this expression. This is a worst case scenario for Q_{trg} because there is a partial cancellation of offset between n-channel and p-channel devices. This cancellation is difficult to predict because of the complexity in the terminal impedances looking out of the source-drain of the MOS devices [60]. The differential charge injection Q_e is modeled as:

$$Q_e = Q_{trg} \delta_s \delta_{trg} \tag{6.33}$$

where δ_s represents the splitting of charge to drain and source and δ_{trg} is the mismatch between the upper and lower transmission gates in a differential circuit. Thus, as a result:

$$Q_e = \delta_s \delta_{trg} [(WL)_{trn,eff} + (WL)_{trp,eff}] C_{ox} (V_g - V_t)$$
(6.34)

where $V_g - V_t = |V_{gn} - V_t| = |V_{gp} - V_t|$.

It is desirable to use a single MOS device for the reset switch in order to reduce the total amount of parasitic coupling present on an op-amp input node. The above analysis

will handle this case as it is a subset of using a transmission gate for the reset switch. Thus, for a single NMOS device used in each reset switch:

$$Q_e = \delta_s \delta_{trg}(WL)_{trn,eff} C_{ox}(V_g - V_t)$$
(6.35)

For the Harris HBC-10 process, the effective dimensions are calculated as $W_{eff} = W - 3.2$ and $L_{eff} = L - 0.6$ with units of μ m. The NMOS reset switch is drawn with an aspect ratio 60/2.4. For this process, the gate-oxide thickness is 200 Å. As a result, $C_{ox} \approx 2 \times 10^{-7} \mathrm{F/cm^2}$. The threshold voltage $V_t \approx 0.7$ V and the gate potential is assumed to be 5 V. Notice that the source potential is near 0V because the common mode feedback holds the common mode level near 0V. With the channel charge splitting assumed to be equal between source and drain, $\delta_s = 0.5$. Thus, Q_e is determined to be:

$$Q_e = 4.4744 \times 10^{-13} \delta_{trg} \tag{6.36}$$

With $C_1 = 1$ pF and $V_{ref} = 2.5$ V, the constraint above is evaluated as:

$$\delta_{trg} < 5.5878\alpha_2 \tag{6.37}$$

where it is assumed that α_2 will swamp out α_1 . Assuming a worst case of 10% charge injection mismatch, $\delta_{trg} = 0.1$. Thus, $\alpha_2 > 0.0179$. Notice that for lower V_{ref} , α_2 must be increased in order to tolerate the same charge injection mismatch. For this design, α_2 is set to 0.035 in order to provide sufficient design headroom for lower operating voltage conditions.

Op-Amp Input Referred Offset

Since the SHA and MX2 function performs correlated double sampling of the op-amp input referred noise and offset, offset contributed from nominal op-amp component mismatch is canceled.

6.4.3 Noise

Direct and Sample-and-Held Thermal Noise

Thermal noise represents one of the ultimate limits to the achievable resolution of the ADC, regardless of input frequency. Thermal noise is contributed by the op-amp input stage, reset switch and SC networks. Thus, it is important that the op-amp thermal noise Power Spectrum Density (PSD) is minimized. It is also important to use as large capacitors as possible to reduce the kT/C noise. However, this requires a quadratic increase in capacitance in order to make impact. In order to maintain speed, this results in increased power dissipation.

Complete charge transfer in SC circuits implies that the amplifier time constants are much smaller than the period T. As a result, severe aliasing occurs in the sample-and-held thermal noise when wide-band thermal noise is undersampled. It is shown in [63, 64, 65] that even slight undersampling causes aliased thermal noise to appear white. A detailed analysis of the SC thermal noise for the SHA, MX2 and ADC is shown in Appendix A.

Flicker (1/f) Noise

Flicker or 1/f noise is largely contributed in the SHA and MX2 by the PMOS transistors from the op-amp. The SHA and MX2 performs correlated double sampling which reduces the PSD magnitude of the 1/f noise. Since the sampling rate of the ADC is ≥ 1 MHz, considerably higher than the corner frequency of most 1/f thermal noise processes, the correlated double sampling is effective.

6.4.4 Settling Time and Accuracy

Achieving low settling time to high-accuracy requires a high unity-gain frequency op-amp with good magnitude and phase characteristics near crossover. The use of BiCMOS provides the ability to achieve high unity-gain frequency by virtue of the bipolar devices used in the op-amp. Sufficiently high slew-rate is needed from the op-amp output in order to meet the settling requirements of the pipeline ADC design.

The settling behavior of the SC networks is important. As a result, the use of large aspect ratio, nonsampling transmission gates helps to ensure that the SC networks are not dominating the settling time.

The op-amp d-c gain contributes to the final settling accuracy of the SHA and MX2 stage outputs. Although the digital self-calibration accounts for finite op-amp d-c gain, the op-amp d-c gain can drift due to environmental changes such as ambient temperature change. Large op-amp d-c gain is used to obtain the desired desensitivity to change in the d-c gain [66].

6.4.5 Aperture Jitter

The aperture jitter of the SHA primarily affects the aperture jitter properties of the pipeline ADC. Aperture jitter represents one of the ultimate limitations to the resolution of an ADC for high-frequency input signals. Notice that once the analog data is sampled into the pipeline ADC, then the aperture jitter of succeeding stages is not important. This is because the burden of obtaining the zero-order hold samples of the analog input rests with the SHA.

The extrinsic component of the aperture jitter arises primarily from the clock circuits and clock generator used external to the ADC chip. To a large extent, these components can be reduced through appropriate use of high-precision clock circuits. The intrinsic component of the aperture jitter may involve thermal noise of digital gates in the sampling clock signal path.

In this design, direct metal connection is established from the bond-pad to the gate of the MOS reset switch. Thus, no digital gates are used that can add aperture jitter. This scheme also permits a simple RC filter to be used at the SHA clock pin allowing the SHA clock waveform to be controlled externally, if desired.

6.5 SC Comparator

6.5.1 Basic Structure

The comparator employs SCs, pre-amplifiers and a latch. The system operates as an offset canceled pre-amplifier system that is then used to drive a latch with sufficient drive [67]. Offset cancellation is used in the pre-amplifiers to reduce the nominal offset of the pre-amplifiers so that the dominant comparator offset is the pre-amplifier input referred latch offset. The pre-amplifier system design employs two offset canceled pre-amplifiers so that the input referred offset is less than 1 LSB at the intended resolution. The reason for this is that the original digital self-calibration scheme corrected for ratiometric errors but not for comparator errors.

The comparator function begins by connecting the analog input to the first pre-amplifier. At the end of this mode, the pre-amplifiers have amplified the input so that a fraction of an LSB input can drive the maximum expected latch offset.

6.5.2 Reset Mode

Fig. 6.7 shows the comparator structure in the reset mode. During the reset mode, the analog input is disconnected and the inputs to each pre-amplifier are grounded. The offset referred error is then stored on the capacitors [68]. This performs correlated double sampling as well as offset cancellation. One concern of this approach is that the output referred voltage AV_{os} . It is important that this output voltage does not exceed the linear operating range of the pre-amplifier in order to ensure that complete offset cancellation is accomplished. For a given offset and linear operating range, an upper limit on the gain A is implied. Because of this constraint, two offset canceled pre-amplifier stages are employed in order to achieve the desired gain without leading to excessively large AV_{os} for each pre-amplifier stage.

6.5.3 Amplify Mode

Fig. 6.8 shows the comparator structure in the amplify mode. The offset canceled preamplifiers drive the latch which in turn is used to hold the state of the comparator.

6.5.4 Latch

The latch design is a cross coupled inverter CMOS latch. This structure is relatively simple and uses small area and power. The relatively high offset of the basic latch design may require at least one pre-amplifier for the 1-bit-per-stage digital self-calibration presented in this thesis. Optimization of the latch design can reduce the latch offset and thus relax overall pre-amplifier gain requirements.

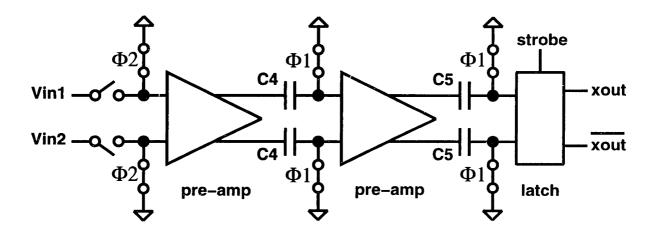


Figure 6.7: Comparator structure in the reset mode.

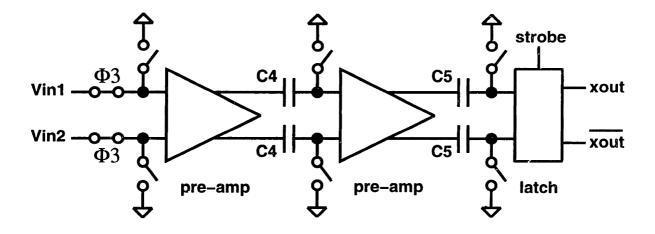


Figure 6.8: Comparator structure in the amplify mode.

6.6 Limitations to the SC Comparator

6.6.1 Residual Offset

Charge Injection

Offset in the comparator should be minimized so that the maximum dynamic range of the MX2 stage can be used. Charge injection represents the dominant residual offset component. The digital self-calibration can tolerate comparator offset which relaxes the comparator offset requirement.

Component Mismatch

Since the comparator function performs correlated double sampling of the pre-amp input referred noise and offset, offset contributed from nominal pre-amp component mismatch is canceled.

6.6.2 Noise

Noise in the comparator appears as a dynamic offset which is removed by the digital self-calibration. Since the calibration tolerates static or dynamic comparator error, comparator noise is unimportant.

Frequency Response

Achieving low settling time to high-accuracy requires a high-speed pre-amplifier design. The use of BiCMOS provides the ability to achieve high-frequency response by virtue of the bipolar devices used in the pre-amplifier.

The settling behavior of the SC networks is important. The use of large aspect ratio transmission gates helps to ensure that the SC networks are not limiting the settling time.

Chapter 7

Analog Integrated Circuit Design

7.1 Operational Amplifier

7.1.1 Op-Amp Overview

BiCMOS is used to provide n-p-n bipolar devices for high-speed, high-gain analog capability while providing CMOS devices for a SC environment. The operational amplifier is shown in Fig. 7.1. The op-amp is a 2-stage, fully-differential design with a 100 MHz unity-gain bandwidth and a 125 dB d-c gain [69, 70]. The main signal path is indicated in bold. A PMOS differential input stage, composed of devices M_1 and M_2 , is used for lower 1/f noise and lower threshold voltage relaxation, compared to NMOS [6]. Focus will be placed on the right half of the remainder of the op-amp signal path. An n-p-n bipolar second stage Q_2 is used to achieve a high-frequency nondominant pole. The second stage is cascoded with device Q_3 and then actively loaded with a cascoded current source, composed of devices M_9 and M_{10} , to obtain the high output resistance used to help achieve large op-amp d-c gain. This large d-c gain is desirable to avoid the need for recalibration due to gain drift arising, for instance, from a temperature change. Device Q_1 is used as an emitter follower to prevent the second stage from excessively loading the first stage output and thus reducing the d-c gain. Compensation capacitor C_c is used for pole-split compensation of the op-amp [71].

7.1.2 Frequency Response Issues

Pole-Splitting

The root locus of the feedback system present in the 2-stage op-amp is considered. With an ideal two-stage design, the root contour of the poles for the op-amp, as a function of the compensation capacitance C_c , will traverse a path that is along σ axis in the left-half s-plane [71, 72]. Thus, as C_c is increased from zero, a dominant pole moves towards the origin in the s-plane while a nondominant pole moves along the negative σ axis, hence pole-splitting.

One of the difficulties with a 2-stage op-amp design is that there is potential for failure of pole-splitting [71]. When additional poles are present in the op-amp second-stage, then

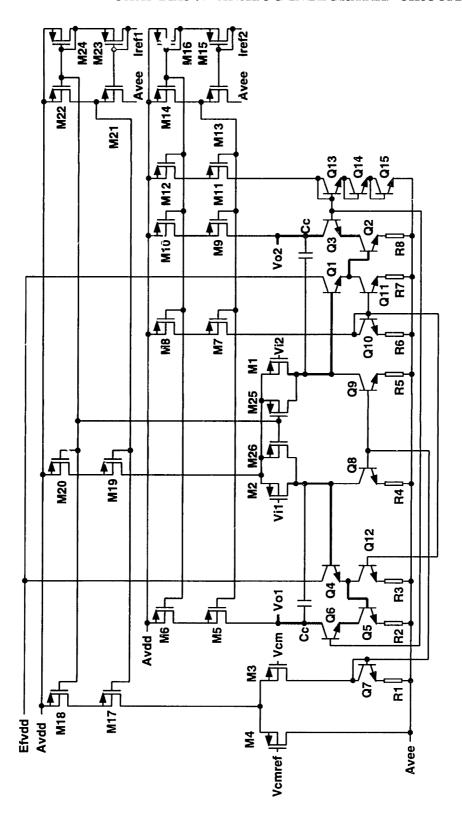


Figure 7.1: Operational amplifier.

the root contour no longer is confined to the σ axis. The addition of a third high-frequency pole results in a root contour that has a dominant pole evident but with two nondominant poles that split off of the σ axis. As is shown in [71], these two nondominant poles can approach the $j\Omega$ axis. The use of an emitter follower Q_1 between the first and second stages of the op-amp is one source of a high-frequency pole that can contribute to failure in pole-splitting. The presence of parasitic base resistance in the common-emitter amplifier Q_2 in the second stage is yet another potential source that can contribute to failure in pole splitting.

When failure of pole-splitting occurs, the op-amp frequency response can exhibit gain peaking which can cause instability in closed-loop amplifiers using the op-amp. This is caused by the nondominant poles approaching the $j\Omega$ axis, as discussed above. The manner in which the peaking effect is reduced is to degenerate the transconductance of device Q_2 . Resistor R_8 is used for this purpose. This permits a reduction of the loop transmission magnitude in the minor loop while maintaining the current level in the second stage needed to meet slew-rate requirements. One of the difficulties of introducing R_8 is that increasing the emitter potential can result in saturation when the temperature is increased. As a result, the cascode device Q_3 is biased with a diode stack composed of three diode connected bipolar devices, Q_{13} , Q_{14} and Q_{15} . This is necessary in order to retain device Q_2 out of saturation under the conditions of expected operating temperature and increased emitter potential due to R_8 . It is important to point out that the large transconductance of the n-p-n device is still needed to obtain the high-performance in the op-amp. However, the method described here decouples current level issues from stability issues.

Input Stage MOS Device Ω_T

The input stage employs MOS devices which appear in the signal path of the op-amp. The Ω_T of the MOS devices needs to be considered in order to achieve good phase-margin for a high-performance op-amp. This is because the phase response of the op-amp depends on the phase response of the MOS device. The phase response of the small-signal transconductance generator is of importance, as a result. It is shown in [73] that a nonquasi-static analysis of the MOS transistor is necessary in order to predict high-frequency response. The y-parameter description of the transconductance generator is shown to be:

$$y_m = \frac{g_m}{1 + j\Omega\tau_1 + \cdots} \tag{7.1}$$

where the time constant τ_1 is determined to be:

$$\tau_1 \approx \frac{4}{15} \frac{1}{\Omega_T} \tag{7.2}$$

Physically, this result reflects the fact that the inversion layer of the MOS device lags behind the change in the gate-source voltage V_{GS} .

The Ω_T of a MOS device is derived from first principles as [73]:

$$\Omega_T \approx \frac{\mu V_{DS,SAT}}{L_{eff}^2} \tag{7.3}$$

An important consequence of this result is that $V_{DS,SAT}$ should be as large as possible. Another important consequence of this result is that short channel length devices should be employed in the input stage as there is approximately quadratic dependence on the channel length L_{eff} . It is important to take the nonquasi-static analysis into account because some SPICE based circuit simulators do not account for this effect in the AC Analysis.

First Stage Miller Capacitance

The high d-c gain of the op-amp is partially due to the gain of the first stage. Because of the Miller effect in the first stage, the effective input capacitance contribution from the drain-gate overlap capacitance is magnified:

$$C_m = (1 + |A_1|)C_{ad} (7.4)$$

where A_1 is the gain of the first stage. When the op-amp is used in a closed loop-amplifier, the increased input capacitance at low frequencies reduces the available d-c loop transmission magnitude. The simulated d-c gain for this design is 135 dB determined by driving the op-amp with voltage sources connected to the op-amp input terminals. However, the actual op-amp design will never be used in this manner. Thus, the available d-c gain of the loop transmission for the MX2 in the amplify mode is approximately 125 dB.

7.1.3 Transient Response Issues

Slew Rate

In order to meet settling time requirements, the op-amp slew rate must be sufficiently high. One of the disadvantages of a two-stage op-amp design is that the op-amp is biased Class-A meaning that bias current is always present. As discussed above, the use of relatively large second-stage current and the associated potential instability caused by failure of pole-splitting is combated through the use of degeneration resistor R_8 .

Emitter Follower Collector Current

When the differential input pair is steered strongly during slew rate limiting conditions, the emitter follower Q_1 may temporarily experience a large base current drive. The resulting emitter current can exceed the collector current from current source Q_{11} . The resulting current difference is sustained by Q_2 and then passes out towards A_{VEE} . The large emitter current in Q_1 originates from the collector of Q_1 . The emitter follower collector is supplied by a separate power supply, EF_{VDD} . This power supply is joined to A_{VDD} externally to the chip where a lower source impedance is possible than from on-chip power supply connections. In this manner, any large current spikes from the op-amp emitter followers in the ADC system are kept out of sensitive power supply paths, such as A_{VDD} .

Bias Network Splitting

Since the slew-rate of the op-amp outputs can be large during amplifier settling or during reset operation, it is desired to separate the second-stage bias network from the first-stage bias network. As can be seen in the op-amp schematic, I_{REF1} and devices M_{17-24} provide bias current for the differential mode and common mode feedback differential pairs. In addition, I_{REF2} and devices M_{5-16} provide bias current for the second stage and other less sensitive bias needs. Thus, when the op-amp outputs have sharp changes, the relatively slow settling gate bias rails in the input stage bias network are not affected. This is an important strategy to help decouple second-stage transients from the sensitive first-stage bias. Another advantage of separate bias networks is that it provides the ability to change the op-amp operating conditions should process parameters differ from simulated values. This strategy is helpful from the standpoint of testing the ADC research vehicle.

7.1.4 Common Mode Feedback Network

A dynamic common mode feedback scheme is used to sense the output common mode level of the op-amp [70, 74]. The common mode feedback circuit is shown in 7.2. The voltage V_{CM} represents the output common mode level. A differential pair composed of devices M_3 and M_4 is used to steer common mode current to the first stage of the op-amp. This completes the common mode negative feedback loop that tends to drive V_{CM} equal to V_{CMREF} , thus stabilizing the output common mode level. The voltage V_{CMREF} is set between the power supply rails, or ground, for a positive and negative power supply scheme. Devices M_{25} and M_{26} are used to avoid a start-up difficulty in the output common mode level when the power supply is turned on. If the output common mode level is near the A_{VDD} power supply rail, then the input differential pair bias current source M_{19} could be cutoff and the op-amp output common mode level could remain at the A_{VDD} power supply rail. However, M_{25} and M_{26} turn on in this condition to drive the output common mode level away from A_{VDD} , as desired. Once the output common mode level is stabilized, devices M_{25} and M_{26} remain cutoff for normal op-amp operation.

7.2 Comparator Pre-Amplifier

The comparator pre-amplifier is shown in Fig. 7.3. A 2-stage, open loop design is used. The main signal path is emphasized in bold. A PMOS differential input pair, composed of devices M_1 and M_2 , is used for analogous reasons as for the op-amp. Focus will be placed on the right half of the remainder of the pre-amp signal path. Emitter follower Q_2 is used as a buffer between the first stage output and the second stage input. A bipolar differential pair, composed of devices Q_3 and Q_4 , is principally used to achieve the gain in the pre-amp. Emitter follower Q_6 is used to prevent loading of the second stage output. Devices Q_{7-10} are used to establish sufficient common mode input level to bases of Q_1 and Q_2 so that V_{CB} for Q_{13} remains positive. The comparator uses 2 pre-amplifiers in cascade, connected using coupling capacitors, which then drive a simple CMOS cross-coupled latch. Open loop offset

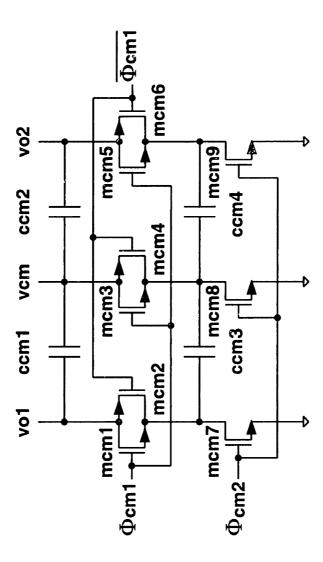


Figure 7.2: Common mode feedback schematic.

cancellation [68] is used to reduce the input referred offset of the pre-amplifiers well within the calibration range. The gain of each pre-amplifier is approximately 60. Since passive loads are used in the pre-amplifier design with low gain, a common mode feedback scheme is not required.

7.3 Dynamic CMOS Latch

The dynamic CMOS latch design used to hold the state of the comparator is shown in Fig. 7.4. The latch operates on nonoverlapping clocks ϕ_1 and ϕ_3 . In order to minimize "kickback" to the second stage SCs, source followers M_9 and M_{10} are used as buffers. When ϕ_1 is high, the source follower outputs are connected to the drains of the latch core devices M_{1-4} . Thus, the latch core is pre-charged to the input voltage during this time. The input switches M_7 and M_8 are turned off when ϕ_1 becomes low. When ϕ_3 becomes high, the latch core activates in positive feedback causing each of the outputs V_{o1} and V_{o2} to rapidly diverge to D_{VDD} or D_{VEE} , depending on the input pre-charge. The advantage of the dynamic CMOS latch design is overall circuit simplicity as well as easy generation of CMOS logic levels.

7.4 Bias Distribution Network

The bias distribution network has the major task of providing the necessary bias currents to various points on the IC. Since the pipeline ADC contains a multitude of op-amps and comparators, the bias network also provides a means to merge the large number of total bias currents needed down to a convenient number for interfacing at the IC pads. Fig. 7.5 illustrates the basic concept of the bias distribution network.

The basic design of the bias distribution scheme is to establish a low impedance voltage V_{DST} which can be distributed by a metal wire across the IC and tapped at any point where current will be needed. A simple voltage-to-current conversion is provided locally. The motivation for this approach is manifold. First, the low impedance drive of V_{DST} provides a means to diminish errors contributed by parasitic capacitance coupling high-frequency clock signals. Second, the use of a linear distribution wire reduces the amount of layout area needed to route bias wires. Third, the modularity of the layout is maintained with a linear distribution scheme since the linear distribution wire can be accessed wherever convenient, even if cell placement is altered.

The Widlar current mirror in the circuit begins with reference device Q_1 . An emitter follower Q_3 is used to drive the distribution wire V_{DST} . This emitter follower also sustains the bias currents accumulated from remote access points along V_{DST} . Device Q_3 is biased at a similar current level to I_{REF} in order to maintain reasonably high device f_T . An emitter resistor R_1 is used with the reference device Q_1 . The emitter resistor voltage is chosen to be large compared to ΔV_{BE} originating from device mismatch or environmental change. Thus, the current ratios are primarily determined by resistor ratios. The thermal tracking between input and output devices cannot be assured due to the relatively long

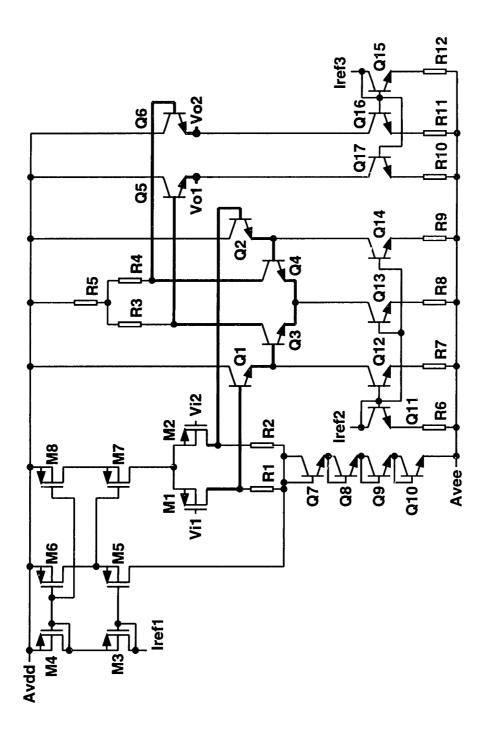


Figure 7.3: Comparator pre-amplifier.

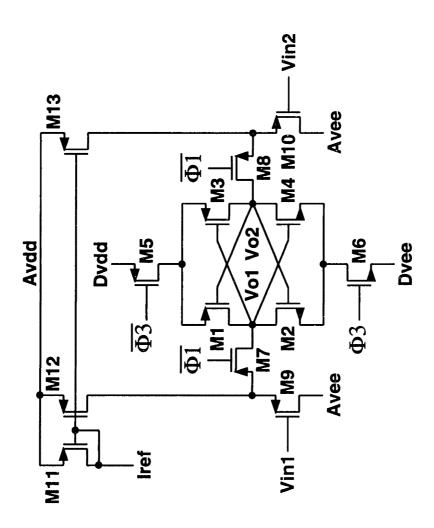


Figure 7.4: Comparator latch.

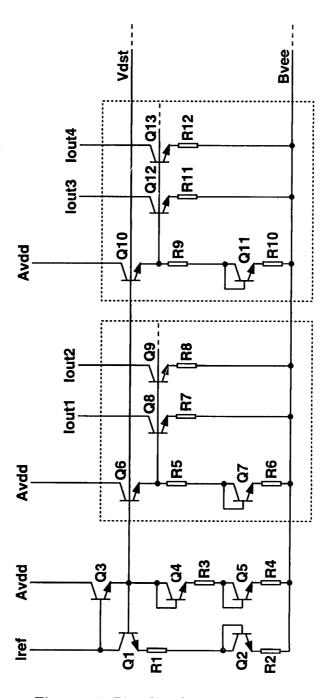


Figure 7.5: Bias distribution network.

distances that the devices are separated by. The use of emitter resistors to degenerate the common-emitter gain helps to ensure thermal stability.

The distribution wire is itself accessed with an emitter follower at the desired access point. This helps reduce coupling of noise between access points along the linear distribution wire. The return currents are in B_{VEE} which is intentionally separate from A_{VEE} on chip. This is an important consideration because the large d-c currents and transient current present in A_{VEE} can adversely affect the bias network otherwise. The potentials B_{VEE} and A_{VEE} are joined externally where a lower source impedance is possible than from on-chip power supply connections.

7.5 Digital Output Buffer

The digital output buffer design is shown in Fig. 7.6. This design employs a simple inverter structure but uses separate power supplies V_H and V_L . This permits the digital output level to be adjusted only as large as necessary. This helps to reduce noise coupling from the digital outputs to other parts of the system.

7.6 Simulation Results

The op-amp performance is summarized in Table 7.1. The comparator pre-amplifier performance summary is shown in Fig. 7.2. The latch performance is summarized in Table 7.3.

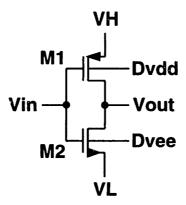


Figure 7.6: Digital output buffer design.

| Parameter | Simulated Value |
|-------------------------|--------------------------------|
| d-c gain | 135 dB |
| unity-gain frequency | 100 MHz |
| phase-margin | 68° |
| differential slew rate | $400~\mathrm{V}/\mu\mathrm{s}$ |
| 14-bit settling time | < 50 ns |
| cap. loading per output | 5 pF |
| input thermal noise PSD | $12.9~\mathrm{nV}/\sqrt{Hz}$ |
| positive PSRR | > 80 dB at 1 MHz |
| negative PSRR | > 80 dB at 1 MHz |
| power supply | ±5 V |
| power dissipation | 90 mW |

Table 7.1: Op-amp performance summary.

| Parameter | Simulated Value |
|---------------------------|----------------------------|
| d-c gain | 36 dB |
| -3 dB frequency | 80 MHz |
| 14-bit offset cancel time | < 50 ns |
| cap. loading per output | 2 pF |
| input thermal noise PSD | $9.5 \text{ nV}/\sqrt{Hz}$ |
| positive PSRR | > 50 dB at 1 MHz |
| negative PSRR | > 50 dB at 1 MHz |
| power supply | ±5 V |
| power dissipation | 30 mW |

Table 7.2: Pre-amp performance summary.

| Parameter | Simulated Value |
|-------------------------|-----------------|
| cap. loading per output | 0.2 pF |
| latch acquire time | 10 ns |
| latch reset time | 20 ns |
| power supply | ±5 V |

Table 7.3: Latch performance summary.

Chapter 8

Mixed-Signal System Layout Design

8.1 Overall Layout Strategy

The goal of a layout strategy for a mixed-signal IC is to provide efficient connection of signals without severely compromising signal integrity. This requires separation of signal types so that interference is minimized. It is desirable to have as many signals accessible as possible for a research vehicle. The relatively large pin count of the IC makes a Pin Grid Array (PGA) package attractive. The PGA package offers essentially four groups of pins, one for each side of the package. The pins nearest the corners are most susceptible to corruption between two adjacent groups. The goal of the layout strategy for the ADC in this thesis is to exploit the partial signal segregation offered by the PGA package. Another important goal is to accomplish as many digital functions as possible external to the ADC IC. This permits the most flexibility, and the shortest circuit and layout design time, while deliberately placing focus on the analog function of the IC.

The design of a large mixed-signal system, such as a pipeline ADC, requires that a hierarchical layout scheme is used to manage the design effectively. An advantage of the pipeline ADC is that it is amenable to hierarchical layout design. Once a single MX2 stage layout is complete, the pipeline ADC layout can proceed rapidly. In this thesis, the layout for the ADC IC uses 18 total stages. There are three types of stages, by function: SHA, G=1.93 MX2 and G=2 MX2. These are all based on a single stage design with minor changes made to convert to the desired function. Another important feature of the design that is exploited in the layout is the fully differential topology. This permits one half of the core op-amp or pre-amp circuit to be laid out. This saves effort and helps guarantee a symmetrical core layout for an op-amp or pre-amp. Thus, much of the layout effort is expended in the design of one stage.

The floor-plan arrangement of the pipeline ADC is largely guided by external constraints. Each pipeline stage is large enough in this design to prevent a linear cascade from being practical for an IC with less than $1 \text{ cm} \times 1 \text{ cm}$ dimensions. As a result, the 18 stages for this design are laid out in a serpentine manner with each leg being 6 stages long. This results in an overall layout that fits inside of a 9.3 mm \times 8.3 mm die area.

8.2 SHA and MX2 Layout

The SC circuits require attention in order to minimize clock noise. Fig. 8.1 shows the basic arrangement of circuitry in the SHA and MX2. Recall that minor circuit changes are accomplished in order to derive the SHA, G=1.93 MX2 and G=2 MX2 circuits. Notice that the clock lines are arranged so that the clock lines never cross op-amp summing nodes. This is very important because parasitic capacitive coupling to the op-amp summing node during the amplify mode, for the respective stage, will cause a clock noise error to appear at the residue output of that stage. The bias sections are laid out separate from the op-amp and comparator cores in order to keep the op-amp and comparator interstage wiring as simple as possible. Thus, a line of symmetry is maintained for the signal handling parts of the stage. The layout design of the stage permits relatively easy hierarchical concatenation of stages at the root layout level. This way, any changes made in a stage are immediately propagated to the entire ADC. The analog signals are daisy chained from stage to stage. Notice that the analog signals are as far away as possible from the clock lines.

8.3 ADC Layout

The ADC layout is shown in Fig. 8.2. The basic organization of the ADC is to provide 3 rows, each row with 6 stages. The arrows indicate the direction of the data propagation along the rows. The goal is to arrive at an IC with reasonable aspect ratio in order to ease packaging requirements. As can be seen from the floor-plan, the actual circuits within each section are surrounded with a multitude of distribution signals including: power supply, reference, pipeline control clocks, calibration control clocks and digital outputs. The goal of the layout design was to distribute these signals parallel to the sections so each section can obtain easy access. The second level metal layer (M2) in the Harris HBC-10 process has lower capacitance to substrate and is thicker than the first level metal (M1). Thus, M2 is preferred for the east-west signals. The north-south access wires are in M1 as a result. Notice that the east-west clocks lines terminate at the western side of the layout in order to permit the western bias lines to be free from clock crossings. The pipeline ADC uses a total of 94 pads. Approximately 20% of these pads are dedicated for power supply and analog ground. The pipeline ADC PGA package is discussed in greater detail in Appendix B.

8.4 Layout Verification

The layout verification is an important process of the layout design. The use of layout verification tools is intended to compare the layout net-list to a schematic net-list. However, in order to have a useful verification, it is necessary to establish the validity of the schematic. The circuit simulation is a powerful tool for this purpose as it permits evaluation of basic circuit function. Successful simulation, and correct interpretation of the simulation results, is critical to establish validity of the mixed-signal system schematic. The entire pipeline ADC transient response was simulated with power supply voltages and bias currents ramped

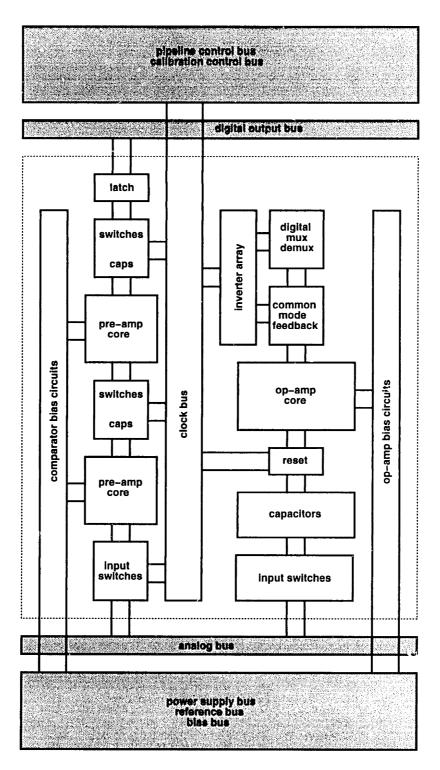


Figure 8.1: Pipeline ADC stage layout.

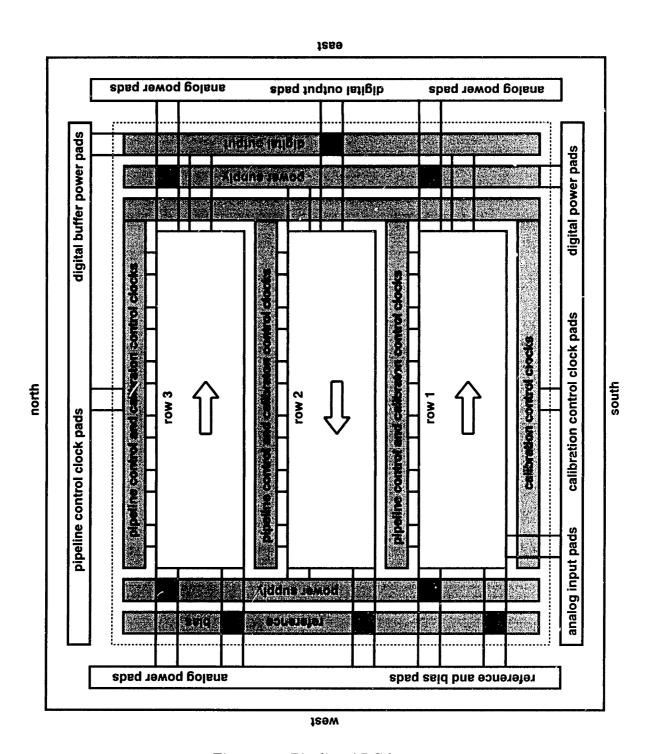


Figure 8.2: Pipeline ADC layout.

up from zero to normal values in order to determine correct power-up operation. With different d-c inputs applied to the ADC, the residue outputs and digital outputs were carefully examined to ensure that data correctly propagated down the pipeline. It was desirable to evaluate the ADC with more extensive test input signals, such as sine-waves. However, prohibitively long simulation time with such input signals limited the input test signals to d-c cases only. However, simulation of the SHA with a wide variety of test input signals was accomplished in significantly less time. The combination of careful physical layout design, schematic layout design, simulation and layout verification resulted in a single design cycle for the pipeline ADC IC in the Harris HBC-10 process.

Chapter 9

Test System

9.1 Overall Test System Strategy

The main purpose of the test system is to provide a means for evaluating the pipeline ADC under various test conditions. The test system is part of the ADC and requires extensive design and careful implementation to insure that performance of the ADC chip is not limited by the test system, to the extent that this is possible.

The strategy used in the test system is to accomplish as many functions with software as is possible since an ADC produces digital data as the primary output. The use of software functions permits rapid reconfiguration of digital self-calibration algorithms. This flexibility is particularly important for a research vehicle. In addition, the use of software control in the laboratory permits automation of experiments that may require tedious manipulations otherwise.

In an effort to minimize software development needed, the simulator program for the digital self-calibration is used to derive the test system program that performs the digital self-calibration for measured data. This provides a powerful tool in the laboratory for comparing the results of a simulation with measured data. The use of the C programming language adds additional power because of the wide availability of C libraries for different computer platforms.

An important strategy used in the test system is to store as much real time data as is reasonable. This is important because the real time data acquired from the laboratory is then easily recalled at a later time. All data acquired is compressed and backed-up to insure that it uses a minimum of disk space and that it is available at all times.

9.2 Test System Structure

The structure of the test system used to evaluate the performance of the pipeline ADC is shown in Fig. 9.1. The goal of the test system is to provide a platform for providing all necessary test signals needed for the ADC. A 4-layer Printed Circuit Board (PCB) is the central part of the test system. The ADC IC is inserted into a Zero Insertion Force (ZIF)

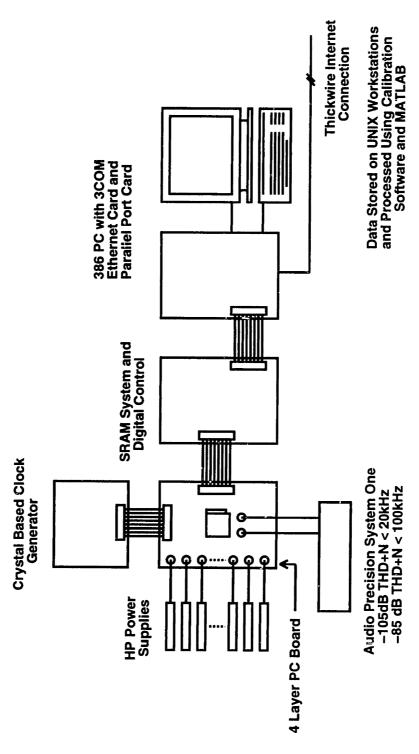


Figure 9.1: Test setup for the pipeline ADC.

9.3. PCB

socket. Power supply, clock, bias, input and output signals all converge to the PCB. The ADC voltage reference signals are generated on the PCB to insure minimum corruption of these signals.

A Personal Computer (PC) is used to control the test system. Once the ADC operating environment is established, the PC is used to acquire calibration mode data and then to acquire run mode data while different test input signals are provided to the ADC. All data obtained from the ADC is at a higher data rate than the PC is able to easily acquire. This requires that high-speed data is first acquired using a bank of Static RAM (SRAM) memory. As a result, a memory system is constructed to act as a buffer between the ADC and the PC. After the data is acquired by the memory, it is then loaded up to the PC.

The PC in this test setup is not convenient for large file and memory manipulations and does not employ a high-speed CPU. Thus, the data is then loaded up from the PC to networked workstations which provide a much more flexible environment for data processing. Another advantage of this approach is that data can then be accessed in the laboratory and processed on any computer on the network that is convenient. Once the data is located on a workstation platform, the data can be stored on disk drives with larger capacity and with back-up schedules. Also, a powerful workstation does not need to be located in the laboratory.

Pipeline ADCs require digital shift registers to perform a time alignment of the data. This is necessary because the analog data must propagate to the end of the pipeline before the conversion is complete. Thus, the bits determined along the way must be stored. The test system performs this operation in software to avoid the need for additional hardware on the research vehicle IC or PCB. If the entire digital self-calibration system is to be performed in hardware on the IC, then the shift registers would be necessary on the IC.

The test system C code programs are shown in Appendix D.

9.3 PCB

9.3.1 Overall Design

The PCB layout is shown in Fig. 9.2. The PCB was designed using the P-CAD PCB layout system. The general location of circuitry on the PCB is designed with consideration of signal types on the ADC IC pads. A four layer board design is chosen in order to obtain an effective ground plane while using component and solder side only for signals. A difficult design aspect of the PCB is the ground plane layout. Ideally, a single ground plane would be used for the entire system. A competing design approach would employ a star-ground approach. For high-accuracy, high-speed ADCs, design experience has favored a ground plane based design [75]. In either approach, it is important to identify where currents are present in the ground plane and what signal type they belong to. The use of surface mount ceramic bypass capacitors is important in order to provide a low impedance return current path to the ground plane. These capacitors are connected within several millimeters from the PGA pin in order to minimize impedance. All BNC connectors on the PCB are bypassed

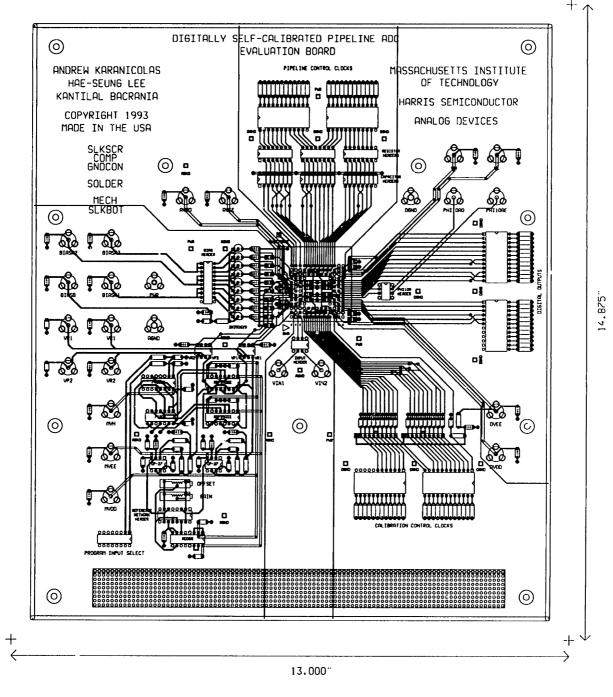


Figure 9.2: Layout of the PCB.

with large tantalum capacitors. The use of BNC cable to connect power supplies is helpful to shield transients in the power supplies.

127

9.3.2 Ground Plane Design

The ground plane design chosen is based on the principle that the ratio of copper conductivity to insulator conductivity is extremely large. As a result, it is possible to guide current by notching out the ground plane where it is not desirable to have a current path. The shaded regions in Fig. 9.3 indicate the copper ground plane and the light colored regions indicate the ground plane notches. Instead of relying on thin wires to connect digital and analog ground planes, a common ground is intended to occur underneath the ADC by using the ground plane layer itself. In this manner, analog and digital return currents always pass through relatively wide regions of ground plane, and thus through relatively low impedances. Observe that the analog input return currents are kept away from the reference, bias and calibration control clocks. The digital output return currents are kept away from the digital power supply and calibration control clocks. The analog power supply return currents are kept away from the pipeline control clocks, bias and reference. Thus, this design uses principles of star grounding techniques but employs wide ground plane material in order to be effective at relatively high-frequencies.

9.4 Analog Inputs

The analog input circuit used for the pipeline ADC is shown in Fig. 9.4. The use of an XLR connector cable permits a shielded and balanced cable to transmit the sine-wave from the sine-wave generator to the ADC. Although coaxial cable could be used, it has much higher capacitance per unit length than the XLR connector cable. This is an important consideration because the harmonic distortion of the sine-wave generator is adversely affected by excessive generator load capacitance [76]. For the relatively low frequencies of operation, the transmission line impedance seen at the generator side will appear primarily capacitive. A one-pole $R_{in}C_{in}$ filter is used at each ADC input. The use of high-quality components, such as mica, polystyrene or polypropylene capacitors and metal-film resistors, is essential to prevent the input filter from generating significant harmonic distortion. The metal-film resistor R_{in} is chosen with a mica capacitor $C_{in} \approx 1000$ pF so that the cutoff frequency of the filter is near the input sine-wave frequency.

9.5 Pipeline Control Clocks

The pipeline control clock driver circuit employed is shown in Fig. 9.5. The pipeline ADC uses a split power supply, nominally ± 5 V. The approach chosen to drive the pipeline control clocks is to provide an external circuit which derives the clocks needed for the ADC. The clock driver circuit employs a TTL level input. This is convenient as TTL compatible logic can then be used to derive the logical clocks before driving the pipeline ADC.

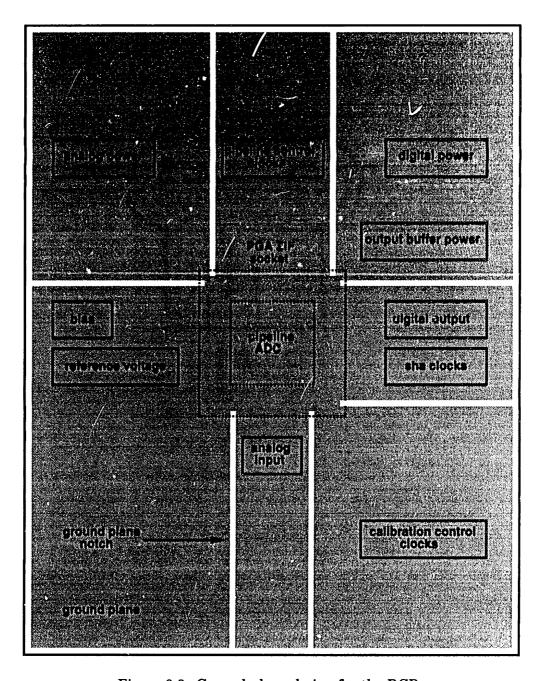


Figure 9.3: Ground plane design for the PCB.

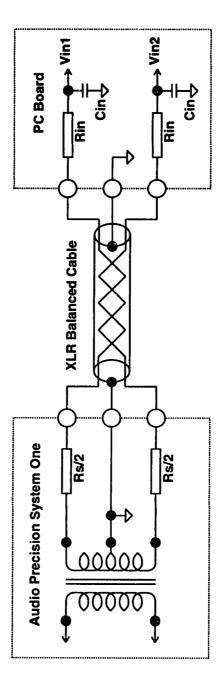


Figure 9.4: Analog input circuit for the pipeline ADC.

The signal path of the clock driver begins with a p-n-p transistor to accomplish level shift of the TTL input logic level to lower potentials. An open collector TTL logic gate is used to provide an easy means for generating up to ± 5 V output with relatively high current drive. This clock driver circuit is suitable for operating near 10 MHz. In this design, Q_1 is a 2N3906 p-n-p transistor, U_1 is a TTL 7407 open collector gate, $R_1 = 470 \Omega$, $R_2 = 470 \Omega$, $R_3 = 220 \Omega$.

A crystal clock generator is used to derive the pipeline control clocks used in the ADC. The SHA clocks are passed through a minimum of logic circuitry before being passed to a clock driver circuit. This helps minimize extrinsic contribution of aperture jitter to the SHA clock waveform.

9.6 Calibration Control Clocks

The calibration control clocks are normally static during the run mode operation of the ADC. They are active only one at a time during the calibration mode. Since calibration data is averaged, a particular calibration clock signal is active for a relatively long period of time. With a 1 MHz clock and 2048-pt averaging, a calibration control clock would remain active for 2.048 ms. Since these clocks operate at a low frequency, the external calibration control clock driver circuit is not critical.

The calibration control clocks are derived from a clock driver circuit based on a single p-n-p inverter. The clock driver circuit used for this purpose is shown in Fig. 9.6. Since high current drive is not needed for these clocks, the p-n-p inverter alone suffices for this case. The clock driver circuit employs a TTL level input. The TTL level calibration control clocks are derived from the PC. In this design, Q_1 is a 2N3906 p-n-p transistor, $R_1 = 470 \,\Omega$, $R_2 = 4.7 \,\mathrm{k}\Omega$.

9.7 Reference Voltage Circuit

The reference voltage circuit used to derive the reference voltages for the pipeline ADC is shown in Fig. 9.7. This reference voltage circuit is critical to the accuracy of the ADC. The basic idea of this circuit is to make use of a widely available band-gap voltage reference [77] that is then buffered by low-noise op-amps. Thermal noise from the band-gap reference circuit is filtered by a low-pass filter before the buffer. The buffer circuit permits a low impedance drive capability over a wide frequency range. The voltage reference PGA pins are bypassed with surface mount ceramic capacitors, located in close proximity to the PGA pins, for good high-frequency bypassing. These bypass capacitors are important because the SC circuits in the pipeline ADC need sharp impulses of current in the reference distribution system for short periods of time. Thus, it is critical that the reference potential does not change significantly during this time. In addition, large tantalum capacitors are located nearby for lower frequency bypassing of the voltage reference PGA pins. The voltage reference circuit shown in Fig. 9.7 is primarily derived from [78]. The values used

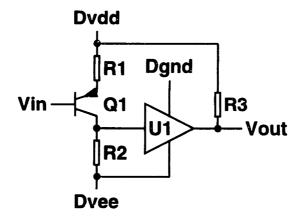


Figure 9.5: Pipeline control clock driver circuit.

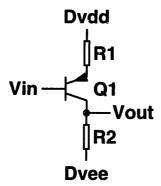


Figure 9.6: Calibration control clock driver circuit.

for the reference voltage circuit are $R_1 = 10 \text{ k}\Omega$, $R_2 = 10 \text{ k}\Omega$ (20T), $R_3 = 20 \text{ k}\Omega$ (20T), $R_4 = 47 \text{ k}\Omega$, $R_5 = 1 \text{ k}\Omega$, $R_6 = 22 \Omega$, $C_1 = 10 \mu\text{F}$, $C_2 = 10 \mu\text{F}$, $C_3 = 0.01 \mu\text{F}$, $C_4 = 10 \mu\text{F}$.

9.8 External Bias Circuit

There are three external bias currents needed for each row in the pipeline ADC. Each row has a comparator, first-stage op-amp and second-stage op-amp bias current. Thus, there are a total of nine bias signals that are needed. The external bias circuit design is shown in Fig. 9.8 The bias signals are arranged to be driven from 3 different power supplies V_{b1} , V_{b2} and V_{b3} , one for each pipeline ADC row. These means are convenient for setting the bias of each row independently. The resistor network can easily be configured to derive all 9 bias currents from a single bias supply V_{b1} . The V_{b0} potential is set sufficiently above A_{VEE} so that the external cascode p-n-p transistors are not saturated. The bias input pins on the ADC IC are bypassed with capacitors to A_{VEE} because the bias inputs are referenced to AVEE in the bias current mirror circuits in the ADC. Surface mount ceramic capacitors of value $0.1 \mu F$ are employed on the PCE for these bypass capacitors. The transistors used are p-n-p 2N3906. The resistors R_{1-9} are chosen to result in the desired bias currents for given operating voltages V_{b0} , V_{b1} , V_{b2} and V_{b3} . The designed bias currents are 0.3 mA for the comparator, 0.5 mA for the op-amp first stage and 1.5 mA for the op-amp second stage. Typical values of resistors for the first row are: $R_1 = 11.3 \text{ k}\Omega$, $R_2 = 6.8 \text{ k}\Omega$, $R_3 = 2.3 \text{ k}\Omega$. For the second row: $R_4 = 11.3 \text{ k}\Omega$, $R_5 = 6.8 \text{ k}\Omega$, $R_6 = 2.3 \text{ k}\Omega$. For the third row: $R_7 = 11.3 \text{ k}\Omega, R_8 = 6.8 \text{ k}\Omega, R_9 = 2.3 \text{ k}\Omega.$

9.9 Analog and Digital Power Supplies

Separate analog and digital power supplies are used to operate the pipeline ADC. The use of multiple power supplies helps minimize corruption that may arise from digital transient currents interfering with the analog power supply. Another advantage of separate power supplies is to maintain flexibility desirable for testing purposes. All power supplies are heavily bypassed with tantalum and ceramic capacitors as close as possible to the ADC.

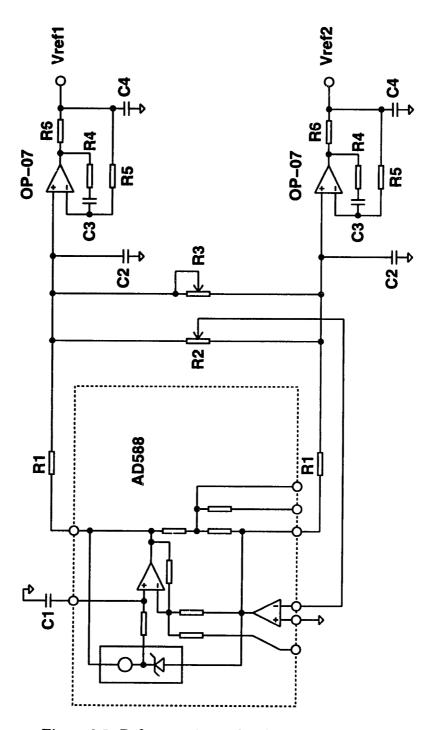


Figure 9.7: Reference circuit for the pipeline ADC.

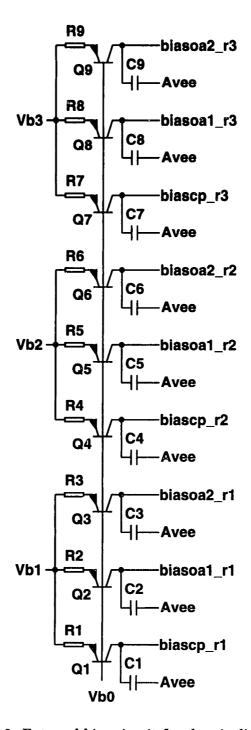


Figure 9.8: External bias circuit for the pipeline ADC.

Chapter 10

Experimental Results

The prototype ADC uses external logic circuits and software to perform the addition, subtraction and data storage for the digital self-calibration algorithm. The Matlab program was used to perform all DSP analysis. In order to operate the ADC, calibration data is first obtained. The converter is then run at the maximum possible speed. Calibration mode data and run mode data are obtained at 1 Ms/s. The chip was designed for a maximum sampling rate of 8 MS/s. However, the present experimental set-up is limited to 1 MS/s. It is surmised that the inductance and capacitance associated with the 121-pin PGA package utilized is the primary source of circuit settling degradation.

10.1 Total Harmonic Distortion

10.1.1 Overview

In order to evaluate the Total Harmonic Distortion (THD) performance, a sine-wave is presented to the ADC using an Audio Precision System One. The sine-wave data from the ADC is then acquired by the test system. A Blackman window is used to window the calibrated sine-wave data prior to performing the FFT [79]. The FFTs are computed with data lengths of 128K-pts. The indices are identified for the Fundamental (FND) sine-wave and Harmonic Distortion (HD) spectral components by examining the FFT magnitude about a local maximum and selecting the index corresponding to the local maximum. The THD is then computed from the FFT by evaluating the ratio of the total HD power related to the FND, not including the FND itself, to the FND power. The total HD power considered extends from d-c to the Nyquist rate, or half of the sampling rate, which is 500 kHz. For the harmonic analysis, all HD amplitudes are measured with respect to the FND amplitude.

10.1.2 Sine-Wave Input, 9.87560 kHz, -2 dB FS

Fig. 10.1 shows the FFT of the measured output data with a 9.87560 kHz sine-wave input at -2 dB FS input. The THD in Fig. 10.1 is measured as -90.5 dB. Table 10.1 presents the harmonic analysis for the FFT.

10.1.3 Sine-Wave Input, 98.7560 kHz, -2 dB FS

Fig. 10.2 shows the FFT of the measured output data with a 98.7560 kHz sine-wave input at -2 dB FS input. The input frequency of 98.7560 kHz is close to the limit of the sine wave generator, which is rated at approximately -85 dB THD+N up to 100 kHz. The THD in Fig. 10.2 is measured as -89.5 dB. Table 10.2 presents the harmonic analysis for the FFT.

10.1.4 Sine-Wave Input, 198.7560 kHz, -2 dB FS

Fig. 10.3 shows the FFT of the measured output data with a 198.7560 kHz sine-wave input at -2 dB FS input. The THD in Fig. 10.3 is measured as -83.19 dB. Table 10.3 presents the harmonic analysis for the FFT.

10.1.5 Sine-Wave Generator Distortion Test

The Audio Precision System One sine-wave generator was measured independently with a Hewlett-Packard HP3585B spectrum analyzer. Typically, a notch filter is needed in this test to remove the sine-wave so that the residual can be analyzed without exceeding the Spurious-Free Dynamic Range (SFDR) of the spectrum analyzer [76]. The Audio Precision System One provides an internal, self-tuning notch filter and measures the THD with an on-board DSP analyzer. However, the DSP unit in the Audio Precision System One has a maximum sampling rate of 192 kHz so an external spectrum analyzer is needed to examine the notch filter output beyond 96 kHz. The sine-wave generator distortion test described here is consistent with the Audio Precision System One THD analysis for frequencies less than 96 kHz.

The sine-wave generator output was set to the same amplitude used in the ADC FFT tests for all frequencies of interest. The HP3585B was then used to examine the spectral content of the Audio Precision System One notch filter output. To determine the harmonic amplitudes, the notch filter gain is accounted for as specified by Audio Precision [76]. A measured HD3 magnitude of -118.7 dB was dominant for a 9.87560 kHz FND. A measured HD3 magnitude of -97.31 dB were dominant for a 98.7560 kHz FND. A measured HD3 magnitude of -84.62 dB was dominant for a 198.7560 kHz FND. Notice that this magnitude is close to the -84.34 dB HD3 peak seen in the ADC FFT for the 198.7560 kHz FND.

10.2 DNL and INL

The DNL and INL were obtained using a sine-wave code density test [80] with a FS sine-wave input at 9.87560 kHz. The Audio Precision System One sine-wave generator was used to provide the input sine-wave for the DNL and INL tests. The sine-wave code density test was run for approximately 8 hours collecting a 32 million sample histogram.

Fig. 10.4 shows the plot of the measured DNL. The peak DNL is within ± 0.25 LSB at 15-bits. Fig. 10.5 shows the plot of the measured INL. The peak INL is within ± 1.25 LSB

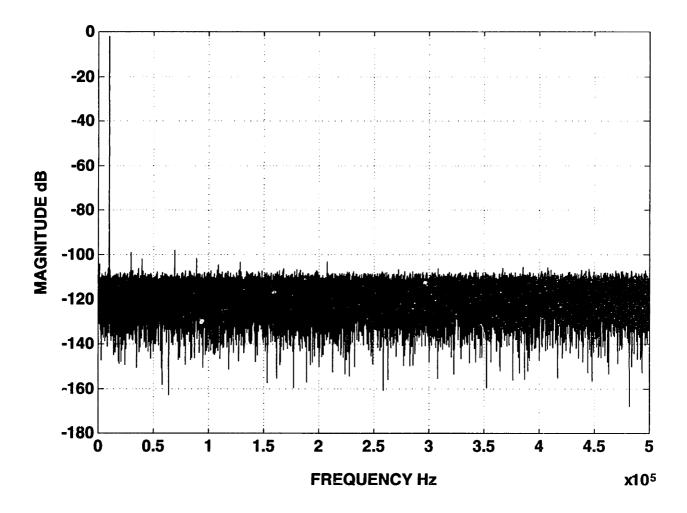


Figure 10.1: Measured FFT plot, 9.87560 kHz sine-wave input, -2dB FS.

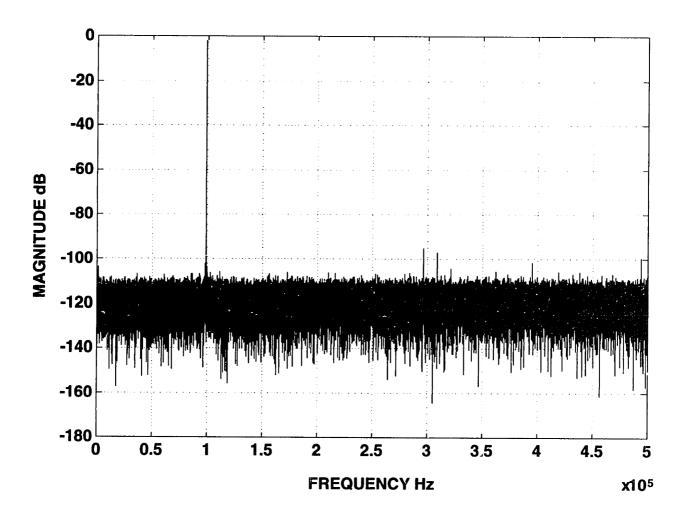


Figure 10.2: Measured FFT plot, 98.7560 kHz sine-wave input, -2dB FS.

10.2. DNL AND INL

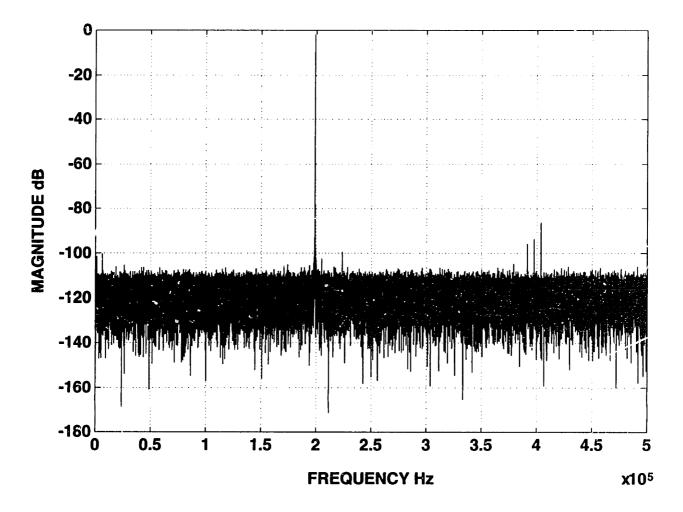


Figure 10.3: Measured FFT plot, 198.756 kHz sine-wave input, -2dB FS.

| Parameter | Frequency kHz | Magnitude dB |
|-----------|---------------|--------------|
| FND | 9.88 | 0 |
| HD3 | 29.64 | -96.83 |
| HD4 | 39.52 | -99.80 |
| HD7 | 69.14 | -95.96 |
| HD9 | 88.90 | -99.60 |
| HD11 | 108.65 | -102.51 |
| HD13 | 128.40 | -101.30 |
| HD21 | 207.41 | -101.12 |

Table 10.1: Harmonic analysis, 9.87560 kHz sine-wave input, -2 dB FS.

| Parameter | Frequency kHz | Magnitude dB |
|-----------|---------------|--------------|
| FND | 98.76 | 0 |
| HD3 | 296.28 | -93.06 |
| HD7 | 308.74 | -94.98 |
| HD4 | 395.03 | -99.49 |
| HD5 | 486.15 | -97.48 |

Table 10.2: Harmonic analysis, $98.7560 \, \text{kHz}$ sine-wave input, $-2 \, \text{dB}$ FS. HD7 is aliased into the baseband.

| Parameter | Frequency kHz | Magnitude dB |
|-----------|---------------|--------------|
| HD5 | 6.20 | -98.29 |
| FND | 198.78 | 0 |
| HD4 | 204.96 | -100.46 |
| HD19 | 223.52 | -97.43 |
| HD7 | 391.36 | -93.65 |
| HD2 | 397.54 | -91.67 |
| HD3 | 403.73 | -84.34 |

Table 10.3: Harmonic analysis, 198.7560 kHz sine-wave input, -2 dB FS. HD5, HD4, HD19, HD7 and HD3 are aliased into the baseband.

at 15-bits. The measured INL shown in Fig. 10.5 reflects contribution from cumulative drift of the calibrated transfer characteristic over the 8 hour period. In addition, the measured INL reflects contribution from the capacitor voltage coefficient of 62 ppm/V.

10.3 Input Referred Noise

The input referred noise of the ADC was measured as approximately 1.25 LSB Root Mean Square (RMS) at 15-bit resolution. Due to reduced power supply operation and a capacitor voltage coefficient of 62 ppm/V, the differential reference voltage was limited to 4 V.

The power supply of ± 5 V is reduced to ± 4 V on some parts. This is because of changes to the Harris HBC-10 ground rules made after the IC design was submitted for fabrication. For a ± 5 V power supply, the differential input range was designed to be 10 V. With a reduction in power supply voltage to ± 4 V, the permissible input range drops to 6 V. The input range drops further to 4 V in order to accommodate capacitor voltage coefficient. An increased input range leads to increased SNR, but at the expense of increased THD.

10.4 Dielectric Relaxation

The effects of capacitor dielectric relaxation were investigated by examining the real time calibration data before averaging. The calibration parameter S_3 was examined, corresponding to $V_{in} = V_{ref}$. The prior calibration parameter S_2 was obtained with $V_{in} = 0$. The calibration data is taken in sequence so that the data for S_3 is obtained shortly after the data for S_2 was collected. No slow settling components, indicative of a dielectric relaxation process, were detected.

10.5 Ambient Temperature

The ambient temperature of the ADC IC package was measured as 25 C. The ambient temperature was dropped to 7 C to investigate the change in calibration constants $S_1 - S_2$. At 25 C, $S_1 - S_2$ was measured for MX2 stage 1 as 122392.53. At 7 C, $S_1 - S_2$ was measured for MX2 stage 1 as 122387.86. Thus, the change in $S_1 - S_2$ is -4.67. With a reduction factor of 5 in the output data, the misalignment in the transfer characteristic due to the measured change in $S_1 - S_2$ would amount to less than 1 LSB INL contribution at 15-bit.

The ambient temperature was also raised to 43 C. At this temperature, the change in calibration constants was observed to be large, on the order of ≈ 1000 , indicating IC malfunction. It is surmised that the reduced power supply introduces op-amp biasing difficulties at elevated temperature.

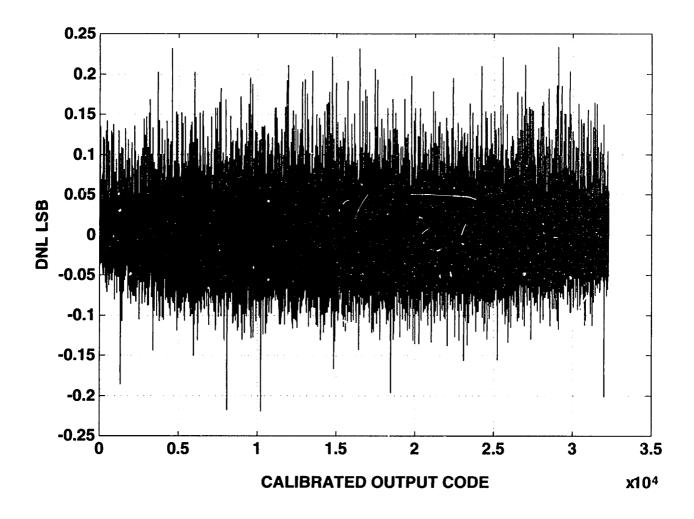


Figure 10.4: Measured DNL plot.

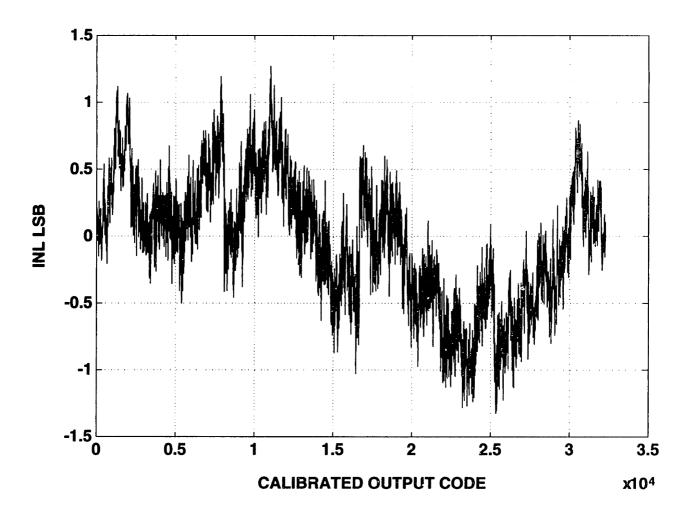


Figure 10.5: Measured INL plot.

| Parameter | Measured Value |
|----------------------|----------------------------|
| Resolution | 15-bits |
| Conversion Rate | 1 MS/s |
| THD | −90.5 dB |
| DNL | $<\pm 0.25~\mathrm{LSB}$ |
| INL | $< \pm 1.25 \text{ LSB}$ |
| Input Referred Noise | 1.25 LSB RMS |
| Input Range | 4 V |
| Power Supply | ±4 V |
| Power Dissipation | 1.8 W |
| IC Dimensions | 9.3 mm × 8.3 mm |
| Component Count | 5955 |
| Process | 11 V, 4 GHz, 2.4 μm BiCMOS |

Table 10.4: Chip performance.

10.6 Power Dissipation and Die Area

The power dissipation reflects two design parameters. First, the ADC operates on a ± 4 V power supply. Second, to achieve high-accuracy at high-speed, large capacitors and large aspect ratio transmission gates are used, resulting in over 7 pF of loading for each operational amplifier. To settle to greater than 14-bit resolution at the target sampling rate of 8 MS/s, relatively high power is necessary for the operational amplifier. Although capacitance and amplifier power can be scaled down for later stages of the pipeline, identical designs are employed for the SHA stage and all 17 MX2 stages for simplicity. No attempts were made to optimize the die area. The summary of performance parameters is listed in Table 10.4. Fig. 10.6 shows the die photograph of the ADC.

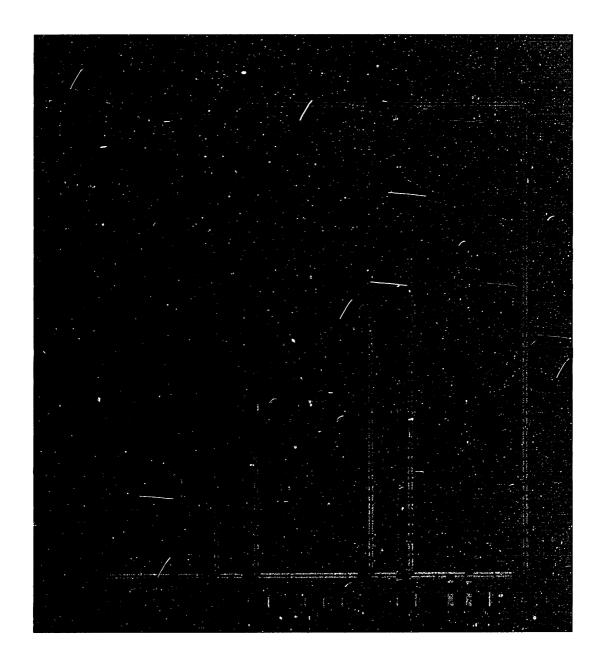


Figure 10.6: Pipeline ADC die photograph.

Chapter 11

Conclusions

11.1 Thesis Summary

The digital self-calibration techniques described in this thesis provide a methodology for designing high-accuracy, high-speed ADCs. The combination of simplified analog and digital hardware design provides a potentially powerful technique for synthesizing high-performance ADCs compared to other ADC approaches. Because of the flexibility present in the digital self-calibration scheme, a family of high-performance ADCs can be synthesized. This technique may be applied to pipeline or cyclic ADC architectures. A 1-bit-per-stage or multi-bit-per-stage design may be employed with either ADC architecture. The digital self-calibration permits flexibility in the desired resolution by simply adding or subtracting stages from the pipeline ADC or by modifying the number of cycles per sample in the cyclic ADC. These modular attributes of pipeline and cyclic ADCs are powerful concepts for the system designer because of the potential for the ADC to be quickly modified to suit the needs of a given application.

This thesis has demonstrated a digital self-calibration technique that is applied to a radix < 2, 1-bit-per-stage pipeline ADC. The digital self-calibration reported here automatically accounts for capacitor mismatch, capacitor nonlinearity leading to DNL, charge injection, finite op-amp gain and comparator offset. The errors are directly measured under the same conditions as during the run mode. Thus, DNL is not contributed by circuit nonlinearity. A 15-bit, 1 Ms/s pipeline ADC prototype was demonstrated utilizing an 11 V, 4 GHz, 2.4 μ m BiCMOS process. No component matching or offsets better than 7-bit accuracy are needed to attain 15-bit linearity with this technique. The digital self-calibration algorithm requires only addition and subtraction arithmetic functions and small data storage.

11.2 Future Work

This thesis has shown that the proposed digital self-calibration is viable for a high-accuracy, high-speed pipeline ADC employing a radix < 2, 1-bit-per-stage architecture. An important direction for future work is further exploration of the performance limits of high-accuracy,

11.2. FUTURE WORK

high-speed pipeline ADCs that employ digital self-calibration concepts. Another important direction is examination of cyclic ADCs with digital self-calibration applied. This is because of the potential for cyclic ADCs with digital self-calibration to improve ADC performance compared to other high-accuracy ADC approaches, such as delta-sigma ADCs, at lower power, less area and reduced complexity. A further direction to expand this work is in the realm of high-speed ADCs operating at moderate resolution with digital self-calibration applied. An optimization study of pipeline ADCs in a digital self-calibration context may yield insight for scaling such ADCs to meet particular accuracy and sampling rate goals while satisfying other design constraints.

An important result from this thesis is that digital self-calibration provides the potential to decouple accuracy issues from speed issues. This identifies an important strategy for employing algorithmic techniques to permit independent optimization of speed. Low-power and low-voltage operating environments are likely to continue placing analog design complexity issues at the forefront of mixed-signal design. In general, exploring how algorithmic techniques can be leveraged to alleviate analog design complexity may be important for realizing high-performance ADCs in such environments.

Appendix A

Switched-Capacitor Thermal Noise Analysis

The analytical procedures for SC thermal noise applied to the SHA and MX2 are reviewed in this Appendix. The analytical procedures shown in [63] use a SC integrator to motivate the procedure. Results derived from [64, 65] are used in [63] and these results will be reviewed to further strengthen the theoretical basis for the analytical procedure given in [63] when applied to the SHA and MX2 in question.

The goal of the analytical procedure is to arrive at a reasonably straightforward analytical expression that can be used to design for specific thermal noise requirements. The analysis considers thermal noise contributed by the op-amp and by the MOS devices used in the SC network.

A.1 Noise Analysis Overview

The goal of the noise analysis is to arrive at the RMS value σ_x^2 of the noise power of a random noise signal x(t). It is shown in [81, 56, 82] that the mean-square power is found by evaluating the integral of the Power Spectrum Density (PSD) $S_x(j\Omega)$:

$$\sigma_x^2 = \frac{1}{2\pi} \int_{-\infty}^{\infty} S_x(j\Omega) d\Omega \tag{A.1}$$

The PSD $S_x(j\Omega)$ is the Fourier Transform (FT) of $R_x(\tau)$, the autocorrelation of x(t). Hence, the autocorrelation is the convolution:

$$R_{\mathbf{x}}(\tau) = \mathbf{x}(\tau) * \mathbf{x}(-\tau) \tag{A.2}$$

Thus, the FT of the autocorrelation is:

$$S_x(j\Omega) = X(j\Omega)X(-j\Omega) \tag{A.3}$$

Since the random signal is real, the FT is conjugate symmetric so that $X(j\Omega) = X^*(-j\Omega)$. Thus,

$$\sigma_x^2 = \frac{1}{2\pi} \int_{-\infty}^{\infty} |X(j\Omega)|^2 d\Omega \tag{A.4}$$

Linear filtering of a PSD $S_x(j\Omega)$ by a filter $H(j\Omega)$ results in an output PSD $S_y(j\Omega)$ [81]:

$$S_{y}(j\Omega) = |H(j\Omega)|^{2} S_{x}(j\Omega) \tag{A.5}$$

A.2 Direct and Sample-and-Held Thermal Noise

The thermal noise sources in the SC network present direct and sample-and-held thermal noise components. The direct component arises from the continuous nature of the SC network and the sample-and-held component arises from the discrete time nature of the SC network.

It is desirable that SC systems accomply n nearly complete charge transfer between specific time slots in the system timing. For many systems, this means that the sampling rate $f_s = 1/T$ of the system is significantly lower than the bandwidth of the circuits used in the SC network, $1/\tau_{sc}$. This means that in terms of the noise autocorrelation function, the correlation time constant is much smaller than the sampling period T. Thus, it can be assumed that the broad-band and sample-and-held noise components in different time slots are decorrelated. Hence, their contributions can be evaluated separately.

Because the time constant $\tau_{sc} \ll T$, the sampling rate $1/T \ll 1/\tau_{sc}$. Thus, the noise signals will be *undersampled* because of the frequent need for nearly complete charge transfer in SC systems.

A.3 Direct Noise Analysis

The goal of the direct noise analysis is to determine an equivalent input referred noise resistance to model a weighted sum of white noise sources present in an amplifier network. The op-amp equivalent resistance is denoted R_{oa} . The direct noise analysis proceeds in an identical manner to classical analog circuit noise analysis as presented in [72, 1]. Thus, the particular procedure is not reviewed here.

It is shown in [1, 63] that the sample-and-held noise component is dominant in SC networks. As a result, the computation of R_{oa} is important primarily for evaluating the sample-and-held noise component that has dependence on R_{oa} .

A.4 Sample-and-Held Noise Analysis

A.4.1 Power Spectrum Density of a Discrete Time Stochastic Signal

Consider a continuous time stochastic signal v(t). A discrete time stochastic signal x(t) can be written as:

$$x(t) = v(t)\delta_T(t) \tag{A.6}$$

where $\delta_T(t)$ is a Dirac comb [64]. Thus, $\delta_T(t)$ can be expressed as:

$$\delta_T(t) = \sum_{n=-\infty}^{\infty} \delta(t - nT)$$
 (A.7)

Applying properties of Dirac unit impulse [83]:

$$x(t) = \sum_{n=-\infty}^{\infty} v(nT)\delta(t-nT)$$
 (A.8)

The autocorrelation $R_x(\tau)$ of the discrete time stochastic signal is of interest:

$$R_x(\tau) = E[q(\tau)q^*(t+\tau)] \tag{A.9}$$

Since q(t) is real, and since v(t) and $\delta_T(t)$ are independent, $R_x(\tau)$ is expressed as:

$$R_x(\tau) = E[v(\tau)v(t+\tau)\delta_T(\tau)\delta_T(t+\tau)]$$
 (A.10)

$$= E[v(\tau)v(t+\tau)]E[\delta_T(\tau)\delta_T(t+\tau)]$$
 (A.11)

This result is written as:

$$R_{x}(\tau) = R_{v}(\tau)R_{T}(\tau) \tag{A.12}$$

It is shown in [83] that:

$$R_T(\tau) = \frac{1}{T}\delta_T(\tau) \tag{A.13}$$

As a result, $R_x(\tau)$ becomes:

$$R_{x}(\tau) = \frac{1}{T}\delta_{T}(\tau)R_{v}(\tau) \tag{A.14}$$

Now the FT $S_x(j\Omega)$ can be considered:

$$S_{x}(j\Omega) = \int_{-\infty}^{\infty} \frac{1}{T} \delta_{T}(\tau) R_{v}(\tau) \exp(-j\Omega\tau) d\tau \qquad (A.15)$$

$$= \int_{-\infty}^{\infty} \frac{1}{T} \sum_{n=-\infty}^{\infty} R_{\nu}(nT) \delta(\tau - nT) \exp(-j\Omega\tau) d\tau \qquad (A.16)$$

Interchanging the order of integration and summation [83], the desired result emerges:

$$S_{x}(j\Omega) = \frac{1}{T} \sum_{n=-\infty}^{\infty} R_{v}(nT) \exp(-j\Omega nT)$$
 (A.17)

It is interesting to observe the FT of the sampled version of $R_v(\tau)$ which can be expressed as:

$$\sum_{n=-\infty}^{\infty} R_{v}(nT) \exp(-j\Omega nT) = \frac{1}{T} \sum_{n=-\infty}^{\infty} S_{v} \left[j \left(\Omega - \frac{2\pi n}{T} \right) \right]$$
 (A.18)

Thus:

$$S_{x}(j\Omega) = \frac{1}{T^{2}} \sum_{n=-\infty}^{\infty} S_{v} \left[j \left(\Omega - \frac{2\pi n}{T} \right) \right]$$
 (A.19)

which is the result found in [1]. Fig. A.1 shows $|S_v(j\Omega)|$ and $|S_x(j\Omega)|$ plotted with the $S_v(j\Omega)$ replicas scaled and shifted in frequency.

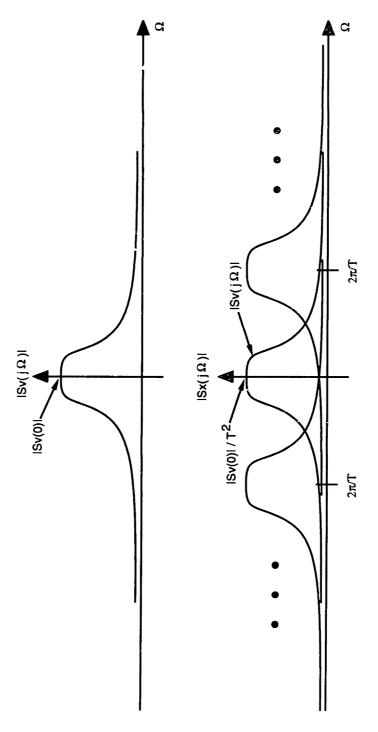


Figure A.1: $|S_v(j\Omega)|$ and $|S_x(j\Omega)|$ plotted with the $S_v(j\Omega)$ replicas scaled and shifted in frequency as a function of Ω .

A.4.2 Undersampled Broad-Band Noise FT

The sample-and-held noise analysis relies on consideration of the undersampled broad-band noise FT. The goal is to express a PSD that results when broad-band noise is filtered by a one-pole filter and then sampled periodically. In [64], the PSD in consideration is of the form:

$$S_{v}(j\Omega) = \frac{S_{n0}}{1 + (\Omega/\Omega_{c})^{2}}$$
 (A.20)

where S_{n0} is the two-sided PSD of the broad-band noise and Ω_c is the 3 dB-frequency of the one-pole filter. It is shown in [63] that the autocorrelation corresponding to the PSD above is:

$$R_{v}(\tau) = \frac{\Omega_{c}}{2} S_{n0} \exp(-|\tau|\Omega_{c})$$
 (A.21)

When the filtered noise v(t) is sampled, the sampled autocorrelation becomes:

$$R_{v}(nT) = \frac{\Omega_{c}}{2} S_{n0} \exp(-|n|\Omega_{c}T)$$
 (A.22)

Using equation A.17, the FT of $R_x(nT)$ is:

$$S_{x}(j\Omega) = \frac{1}{T} \sum_{n=-\infty}^{\infty} \frac{\Omega_{c}}{2} S_{n0} \exp(-|n|\Omega_{c}T) \exp(-j\Omega nT)$$
 (A.23)

Enumerating this summation yields:

$$S_{x}(j\Omega) = \frac{\Omega_{c}S_{n0}}{2T} \left(\frac{\sinh \Omega_{c}T}{\cosh \Omega_{c}T - \cos \Omega T} \right)$$
(A.24)

Since SC systems involve zero-order hold signals, the output PSD is a linear filtered version of $S_x(j\Omega)$. The linear filtered output $S_y(j\Omega)$ is determined to be:

$$S_{y}(j\Omega) = |H(j\Omega)|^{2} S_{x}(j\Omega)$$
 (A.25)

where in the case of a SC system:

$$|H(j\Omega)|^2 = \left(\frac{\sin\Omega\Delta/2}{\Omega/2}\right)^2 \tag{A.26}$$

The parameter Δ is the duty-cycle of the zero-order hold pulse. Thus, $0 < \Delta < T$. For many SC systems, $\Delta = T/2$. Expanding and simplifying the expression for $S_y(j\Omega)$ yields:

$$S_{y}(j\Omega) = S_{n0} \left(\frac{\Omega_{c}T}{2}\right) \left(\frac{\Delta}{T}\right)^{2} \left(\frac{\sin\Omega\Delta/2}{\Omega\Delta/2}\right)^{2} \left(\frac{\sinh\Omega_{c}T}{\cos^{2}\Omega_{c}T - \cos\Omega T}\right)$$
(A.27)

It is useful to define a ripple function $G(\Omega_c T)$ to evaluate change in $S_y(j\Omega)$ from $\Omega T = 0$ to $\Omega T = \pi$ [64]. Notice that $\Omega T = \pi$ corresponds to the Nyquist rate or half of the sampling

rate. The quantity $\Omega_c T$ corresponds to the ratio of undersampling since $\Omega_c T = 2\pi (f_c/f_s)$. The ripple function is now defined as:

$$G(\Omega_c T) = \frac{S_{\nu}(j0) - S_{\nu}(j\pi/T)}{S_{\nu}(j0)}$$
 (A.28)

Evaluating $S_y(j0)$ and $S_y(j\pi/T)$:

$$S_{y}(j0) = S_{n0} \left(\frac{\Omega_{c}T}{2}\right) \left(\frac{\Delta}{T}\right)^{2} \left(\frac{\sinh \Omega_{c}T}{\cosh \Omega_{c}T - 1}\right)$$
 (A.29)

$$S_y(j\pi/T) = S_{n0} \left(\frac{\Omega_c T}{2}\right) \left(\frac{\Delta}{T}\right)^2 \left(\frac{\sinh \Omega_c T}{\cosh \Omega_c T + 1}\right)$$
 (A.30)

This yields $G(\Omega_c T)$:

$$G(\Omega_c T) \approx 1 - \left(\frac{\cosh \Omega_c T - 1}{\cosh \Omega_c T + 1} \right)$$
 (A.31)

using the assumption that:

$$\left(\frac{\sin\Omega\Delta/2}{\Omega\Delta/2}\right)^2 \approx 1\tag{A.32}$$

For the case where $\Delta = T/2$ and $\Omega T = \pi$, it is seen that $\Omega \Delta/2 = \pi/4$. In this case, $(\sin(\pi/4)/(\pi/4))^2 \approx 0.8106$.

Letting $u = \Omega_c T$, a binomial expansion can be written for 1 - G(u) in terms of $\exp(-u)$, where $\exp(-u) \ll 1$:

$$\left(\frac{\cosh u - 1}{\cosh u + 1}\right) \approx 1 - 4\exp(-u), \quad u \gg 1$$
(A.33)

Thus, $G(\Omega_c T)$ is identified as:

$$G(\Omega_c T) \approx 4 \exp(-\Omega_c T)$$
 (A.34)

This demonstrates that $G(\Omega_c T)$ changes very little up to the Nyquist rate under even slight undersampling. For example, when $f_c/f_s=1$, which is slight undersampling, then $\Omega_c T=2\pi\approx 6.2832$. Thus, $G(\Omega_c T)=7.4698\times 10^{-3}$. Typically, $f_c/f_s\approx 10$ or $\Omega_c T\approx 62.832$ for nearly complete settling of high-accuracy SC circuits. Clearly, $G(\Omega_c T)$ is vanishingly small under this condition.

Fig. A.2 shows $|S_x(j\Omega)|$ and $|S_y(j\Omega)|$ plotted as a function of Ω . The approximation for 1 - G(u) is assumed. Notice that $[\sin(\Omega\Delta/2)/(\Omega\Delta/2)]^2$ is taken into account and acts to narrow-band $S_y(j\Omega)$.

The reason why this result is important is because the first-order filtered-and-sampled white noise can itself be considered to be white under the condition of strong undersampling. This greatly simplifies the general analysis of filtered-and-sampled white noise. Under the condition of strong undersampling, $S_y(j\Omega)$ becomes approximately:

$$S_{y}(j\Omega) = S_{n0} \left(\frac{\Omega_{c}T}{2}\right) \left(\frac{\Delta}{T}\right)^{2} \left(\frac{\sin\Omega\Delta/2}{\Omega\Delta/2}\right)^{2}$$
(A.35)

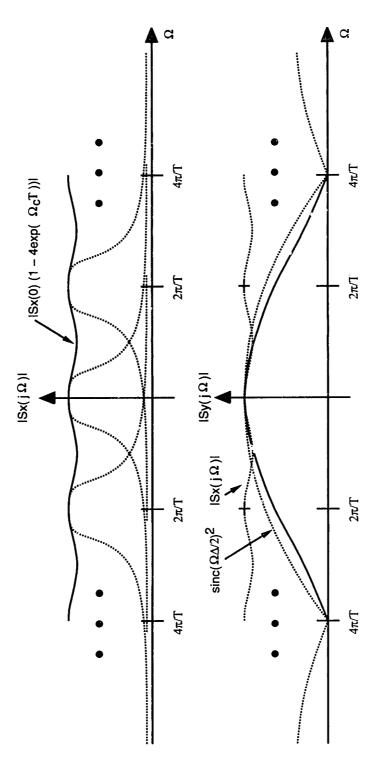


Figure A.2: $|S_x(j\Omega)|$ and $|S_y(j\Omega)|$ plotted as a function of Ω .

The factor $\sinh \Omega_c T/(\cosh \Omega_c T - \cos \Omega T)$ is near unity because both $\cosh u$ and $\sinh u$ are approximately $\exp(u/2)$ when $u \gg 1$. Since $\exp u \gg 1$ in this case, $\cosh \Omega_c T \gg \cos \Omega T$.

In order to determine the RMS power, the PSD $S_{\nu}(j\Omega)$ must be integrated:

$$\sigma_{\mathbf{y}}^{2} = \frac{1}{2\pi} \int_{-\infty}^{\infty} S_{\mathbf{y}}(j\Omega) d\Omega \tag{A.36}$$

Thus, using the approximate PSD $S_{\nu}(j\Omega)$, the RMS power is:

$$\sigma_y^2 = \frac{1}{2\pi} \left(\frac{S_{n0} \Omega_c T}{2} \right) \left(\frac{\Delta}{T} \right)^2 \int_{-\infty}^{\infty} \left(\frac{\sin \Omega \Delta / 2}{\Omega \Delta / 2} \right)^2 d\Omega \tag{A.37}$$

Letting $x = \Omega \Delta/2$, the integral in the expression above evaluates to $2\pi/\Delta$. The RMS power is thus simplified as:

$$\sigma_y^2 = \frac{S_{n0}\Omega_c}{2} \frac{\Delta}{T} \tag{A.38}$$

With a typical duty cycle, $\Delta/T = 1/2$:

$$\sigma_y^2 = \frac{S_{n0}\Omega_c}{4} \tag{A.39}$$

It will be seen in the next section that determining sample-and-held thermal noise is essentially reduced to applying Equation A.39.

An instructive case arises when the one-pole filter is composed of a resistor R generating noise and a capacitor C. In this case, $S_{n0} = 2k\theta R$ and $\Omega_c = 1/(RC)$. Thus, the RMS power is found to be:

$$\sigma_y^2 = \frac{1}{2} \frac{k\theta}{C} \tag{A.40}$$

A.4.3 Enumeration of Sample-and-Held Noise in SC Networks

In order to determine the sample-and-held noise component, the SC network is examined during the sample mode. The key is to identify the one-pole filtered noise for the switched-capacitors in the network. Depending on the noise source, the bandwidth is limited primarily by the op-amp or the network branches. In order to arrive at a useful result, a one-pole op-amp and a one pole network branch is assumed for each respective noise source. In this case, the analysis presented above can be used. Notice that the one-pole case is pessimistic since the integrated PSD with a two-pole, or greater, op-amp would yield a smaller RMS noise result.

It is important to observe that Ω_c is not necessarily related to each S_{n0} source, namely when the op-amp primarily acts as the one-pole filter. Since the loop bandwidth of the op-amp will be proportional to g_m/C_c for a two-stage op-amp design, σ_{oa}^2 will be:

$$\sigma_{oa}^2 \propto \frac{k\theta}{C_c} g_m R_{oa} \tag{A.41}$$

where R_{oa} is the op-amp equivalent input noise resistance. In the ADC presented here, R_{oa} is the same for each op-amp. Depending on the loop-transmission of the op-amp, the unity-gain frequency of the loop transmission may be lower than g_m/C_c . In that case, the RMS power will then be smaller compared to the unity-gain connection case.

Finally, when differential circuits are analyzed, each half-circuit of the differential circuit is analyzed and the PSD is then doubled to arrive at the final PSD.

A.5 SHA Sample-and-Held Noise Analysis

A.5.1 Nonsampling Transmission Gate Contribution

The op-amp is in the unity-gain connection during the sample mode and the summing node will be assumed to be an incremental ground. This is a reasonable assumption because for low frequencies, the d-c gain of the op-amp maintains a nearly incremental ground summing point while at frequencies beyond the unity gain frequency, the load capacitance of the op-amp becomes the dominant impedance. Thus, the voltage across the sampling capacitors is actually diminished further at high-frequencies so the assumption of an incremental ground summing node is pessimistic.

There are two capacitors in question, C_1 and C_2 , used in the sample mode. Thus, the capacitance switched is 2C, where $C = C_1 = C_2$. Consider two switches, each of resistance R, in parallel. Thus, $S_{n01} = k\theta R$, $\Omega_{c1} = 1/(RC)$. The RMS power is then:

$$\sigma_{y1}^2 = \frac{1}{4} \frac{k\theta}{C} \tag{A.42}$$

A.5.2 Op-Amp Contribution

The op-amp has a PSD with an equivalent R_{oa} . Thus, $S_{n02} = 2k\theta R_{oa}$. The op-amp is in the unity connection and filters the PSD S_{n02} . Thus, $\Omega_{c2} = g_m/C_c$. In this case, the RMS noise power contributed by the op-amp is:

$$\sigma_{y2}^2 = \frac{1}{2} \frac{k\theta}{C_c} g_m R_{oa} \tag{A.43}$$

A.5.3 Reset Switch Contribution

For low frequencies, the op-amp loop transmission will attenuate the reset switch contribution. The contribution at high-frequencies is ignored for simplicity.

A.5.4 Total Output Referred Sample-and-Held Noise

The SHA system function magnitude is $|H_{SHA}(z)| = 1$. Thus, the sample-and-held noise power on the sampling capacitors is transferred to the output with $|H_{SHA}(z)|^2 = 1$. The

total RMS output referred power is now found:

$$\sigma_{SHA}^2 = \frac{1}{4} \frac{k\theta}{C} + \frac{1}{2} \frac{k\theta}{C_c} g_m R_{oa} \tag{A.44}$$

The differential output power is simply twice the half-circuit power. Hence:

$$\sigma_{DSHA}^2 = \frac{1}{2} \frac{k\theta}{C} + \frac{k\theta}{C_c} g_m R_{oa} \tag{A.45}$$

A.6 MX2 Sample-and-Held Noise Analysis

The sample-and-held noise analysis of the MX2 is similar to the SHA. The primary difference is that the MX2 system function magnitude is $|H_{MX2}(z)| = 2$. Thus, the sample-and-held noise power on the sampling capacitors is transferred to the output with $|H_{MX2}(z)|^2 = 4$. The other difference is that wide-band noise present from the previous stage needs to be accounted for. Thus, $S_{n03} = 2k\theta R_{pr}$ where R_{pr} represents the equivalent noise resistance appearing at the output of the previous stage. Notice that this noise originating from a previous stage is always low pass filtered by the op-amp in the previous stage. Thus, $\Omega_{c3} = g_m/C_c$. The contribution from R_{pr} is found to be:

$$\sigma_{y3}^2 = \frac{1}{2} \frac{k\theta}{C_s} g_m R_{pr} \tag{A.46}$$

As a result, taking into account $|H_{MX2}(z)|^2$, the total RMS output referred power is now found:

$$\sigma_{MX2}^{2}(i) = \frac{k\theta}{C} + 2\frac{k\theta}{C_c}g_mR_{oa} + 2\frac{k\theta}{C_c}g_mR_{pr}(i)$$
(A.47)

where the index i indicates the specific MX2. The differential output power is simply twice the half-circuit power. Hence:

$$\sigma_{DMX2}^{2}(i) = 2\frac{k\theta}{C} + 4\frac{k\theta}{C_c}g_mR_{oa} + 4\frac{k\theta}{C_c}g_mR_{pr}(i)$$
(A.48)

The op-amp noise dominates the contribution in $R_{pr}(i)$. If the previous stage is a SHA, then $R_{pr} \approx R_{oa}$. However if the previous stage is a MX2, then $R_{pr} \approx 4R_{oa}$. This is because the input referred PSD of the previous op-amp is magnified by $|H_{MX2}(z)|^2$ corresponding to the previous MX2 stage. Thus, there are two principal cases for $\sigma_{DMX2}^2(i)$:

$$\sigma_{DMX2}^2(1) = 2\frac{k\theta}{C} + 8\frac{k\theta}{C_c}g_mR_{oa} \qquad (A.49)$$

$$\sigma_{DMX2}^2(2) = 2\frac{k\theta}{C} + 20\frac{k\theta}{C_c}g_mR_{oa} \qquad (A.50)$$

Notice that $\sigma^2_{DMX2}(i)$ is the same for $2 \le i \le N-1$. Observe further that:

$$\sigma_{DMX2}^{2}(2) = \sigma_{DMX2}^{2}(1) + 12 \frac{k\theta}{C_c} g_m R_{oa}$$
 (A.51)

A.7 Pipeline ADC Sample-and-Held Noise Analysis

The total differential ADC input referred sample-and-held noise is determined as:

$$\sigma_{DADC}^2 = \sigma_{DSHA}^2 + \frac{\sigma_{DMX2}^2(1)}{4} + \frac{\sigma_{DMX2}^2(2)}{16} + \dots + \frac{\sigma_{DMX2}^2(N-1)}{2^{2(N-1)}}$$
(A.52)

Using the results above:

$$\sigma_{DADC}^2 = \sigma_{DSHA}^2 + \frac{1}{4} \left(\sigma_{DMX2}^2(2) - 12 \frac{k\theta}{C_c} g_m R_{oa} \right) + \frac{\sigma_{DMX2}^2(2)}{16} + \dots + \frac{\sigma_{DMX2}^2(2)}{2^{2(N-1)}}$$
 (A.53)

Considering the limit $N \to \infty$, the summation simplifies to:

$$\sigma_{DADC}^{2} = \sigma_{DSHA}^{2} + \frac{\sigma_{DMX2}^{2}(2)}{3} - 3\frac{k\theta}{C_{c}}g_{m}R_{oa}$$
 (A.54)

Thus, the ADC noise expression becomes:

$$\sigma_{DADC}^2 = \frac{7}{6} \frac{k\theta}{C} + \frac{14}{3} g_m R_{oa} \frac{k\theta}{C_c} \tag{A.55}$$

The op-amp thermal noise PSD is simulated as 12.9 nV/ \sqrt{Hz} . Thus, the equivalent resistance $R_{oa} = 10.4 \text{ k}\Omega$. The unity-gain frequency of the op-amp is simulated as 100 MHz. The switched capacitor C and the compensation capacitor C_c are each 1 pF. Thus, the ADC noise is estimated to be:

$$\sigma_{DADC}^2 = 4.6667 \times 10^{-9} + 1.2198 \times 10^{-7} V^2 \tag{A.56}$$

As can be seen, the op-amp contribution is dominant in the ADC noise. The RMS noise is then 355.87 μ V.

A.8 Signal-to-Noise Ratio

A differential FS sine wave with amplitude A has a peak-to-peak amplitude of 2A so $V_{FS} = 2A$. Consider the operating condition of the ADC where $V_{FS} = 4$ V, A = 2 V. The signal power is determined as:

$$\sigma_{DSIG}^2 = \frac{1}{2}A^2 \tag{A.57}$$

The Signal-to-Noise Ratio (SNR) is defined in dB as:

$$SNR = 20\log_{10}\left(\frac{\sigma_{DSIG}}{\sigma_{DADC}}\right) \tag{A.58}$$

Thus, for $V_{FS} = 4$ V, SNR = 75.0 dB. This is near the measured value of about 79.0 dB. Notice that the designed value of the FS input range is $V_{FS} = 10$ V. Thus, the SNR predicted in this case would be 90.9 dB.

Appendix B

Pipeline ADC Package

B.1 Thermal Resistance

The operating temperature of the IC depends not only on the power dissipation of the IC but also on the thermal resistance of the package. The thermal resistance from junction to ambient is denoted Θ_{JA} , with units of K/W. For the PGA package used, $\Theta_{JA} \approx 35.7$ K/W. Thus, the temperature rise for 1.8 W power dissipation is expected to be approximately 64.2 K. As a result, at room temperature ambient of 25 C, the junction temperature would be 89.2 C.

B.2 Wire-bond Connection Diagram

The wire-bond connection for the pipeline ADC IC is shown in Fig. B.1. This PGA package provides a total of 121 pins. With the index mark triangle at the lower left, the 4 sides of the PGA package are referenced counter-clockwise as south, east, north and west. The south side is used to access the differential analog inputs at the southwest corner and the calibration control clocks along the south side. Notice that the differential analog inputs are kept as far away as possible from the relatively high-speed digital signals. The east side is used to access the digital outputs and power supplies. The north side is used to access the pipeline control clocks. The west side is used to access analog bias signals and power supply pins. The most sensitive bias signals correspond to the first row. These bias signals are located on the west side towards the southwest corner. Thus, these bias signals have the most distance from pipeline control clocks. The remaining bias signals are located on the western side with the third row bias signals located towards the northwest corner.

B.3 PGA Pin-out Diagram

Fig. B.2 shows the pin-out of the PGA package used for the pipeline ADC. Pins that are not used are connected to analog ground.

WIRE BONDING DIAGRAM

HBC10 MIT PIPELINE CHIP GHA 121 PIN GRID ARRAY

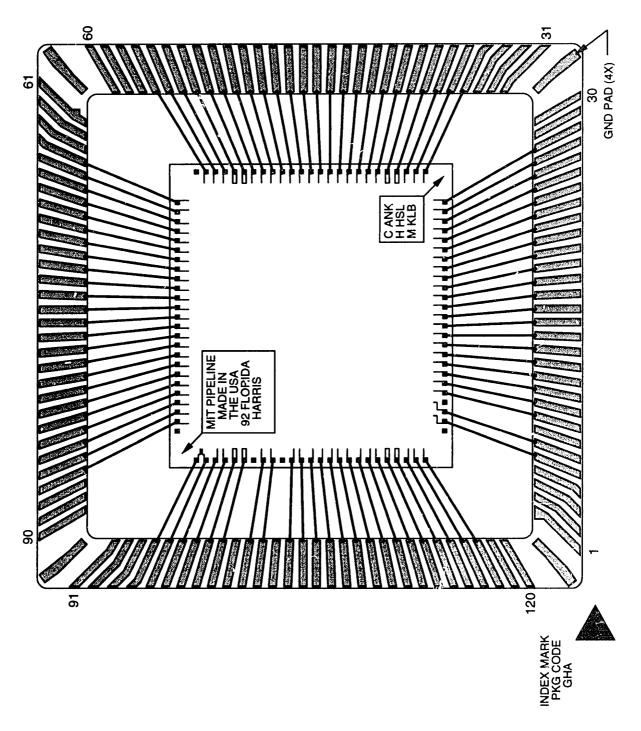


Figure B.1: Wire-bond diagram for pipeline ADC IC.

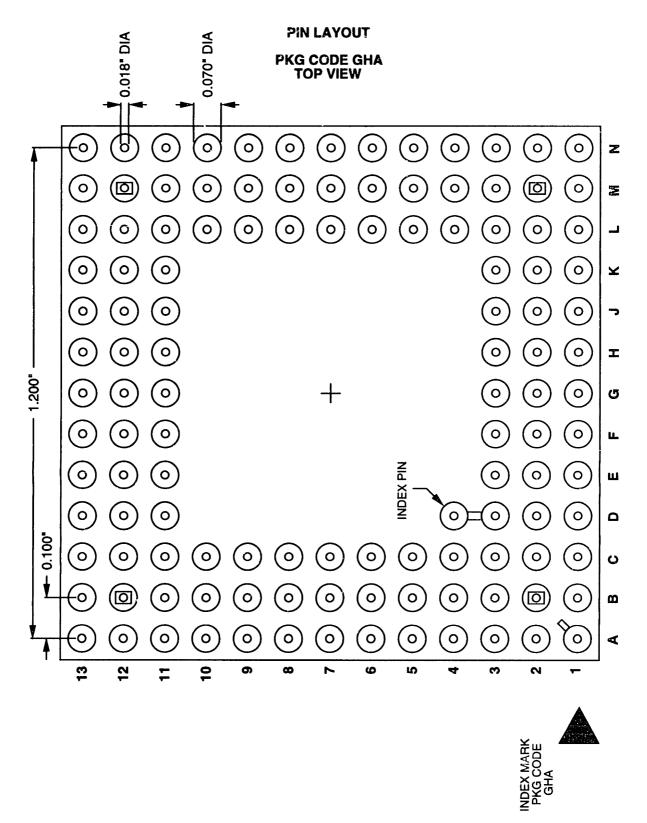


Figure B.2: Pin-out diagram for pipeline ADC IC.

B.4 PGA Pin-out and Signal Table

pga pin and pad assignment

pad numbering convention:

north

| | 90 | 61 | |
|---|-----|----|---|
| | 91 | 60 | |
| W | | | е |
| е | | | a |
| 8 | | | 8 |
| t | 120 | 31 | t |
| | 01 | 30 | |

south

pin and pad assignment table:

| pgapin | pgapad | a2dpad | a2dsignal | pgasignal |
|-----------|--------|--------|-----------|-----------|
| south | | | | |
| C3 | 1 | | | agnd |
| B2 | 2 | | | agnd |
| B1 | 3 | | | agnd |
| D3 | 4 | 1 | nc | agnd |
| C2 | 5 | 2 | vin1 | vin1 |
| C1 | 6 | 3 | vin2 | vin2 |
| D2 | 7 | 4 | nc | agnd |
| E3 | 8 | 5 | phic_s1 | phic_s1 |
| D1 | 9 | 6 | phic_s2 | phic_s2 |
| E2 | 10 | 7 | phic_s3 | phic_s3 |
| E1 | 11 | 8 | phic_s4 | phic_s4 |
| F3 | 12 | 9 | phic_s5 | phic_s5 |
| F2 | 13 | 10 | phic_s6 | phic_s6 |
| F1 | 14 | 11 | phic_s7 | phic_s7 |
| G2 | 15 | 12 | phic_s8 | phic_s8 |
| G3 | 16 | 13 | phic_s9 | phic_s9 |
| G1 | 17 | 14 | phic_s10 | phic_s10 |
| H1 | 18 | 15 | phic_s11 | phic_s11 |
| H2 | 19 | 16 | phic_s12 | phic_s12 |

| Н3 | 20 | 17 | phic_s13 | phic_s13 |
|------|----|----|--------------|--------------|
| J1 | 21 | 18 | phic_s14 | phic_s14 |
| J2 | 22 | 19 | phic_s15 | phic_s15 |
| K1 | 23 | 20 | phic_s16 | phic_s16 |
| J3 | 24 | 21 | phic_s17 | phic_s17 |
| K2 | 25 | 22 | phic_s18 | phic_s18 |
| L1 | 26 | 23 | phii | phii |
| M1 | 27 | 24 | dvdd | dvdd |
| КЗ | 28 | 25 | dvee | dvee |
| L2 | 29 | | | agnd |
| N1 | 30 | | | die_attach |
| | | | | |
| east | | | | |
| L3 | 31 | | | agnd |
| M2 | 32 | | | agnd |
| N2 | 33 | | | agnd |
| L4 | 34 | 26 | nd_s1 | nd_s1 |
| M3 | 35 | 27 | nd_s2 | nd_s2 |
| N3 | 36 | 28 | nd_s3 | nd_s3 |
| M4 | 37 | 29 | avee | 8008 |
| L5 | 38 | 30 | avdd | avdd |
| N4 | 39 | 31 | nd_s4 | nd_s4 |
| M5 | 40 | 32 | nd_s5 | nd_s5 |
| N5 | 41 | 33 | nd_s6 | nd_s6 |
| L6 | 42 | 34 | phi1_oa_even | phi1_oa_even |
| M6 | 43 | 35 | phi1_oa_odd | phi1_oa_odd |
| N6 | 44 | 36 | nd_s7 | nd_s7 |
| M7 | 45 | 37 | nd_s8 | nd_s8 |
| L7 | 46 | 38 | nd_s9 | nd_s9 |
| N7 | 47 | 39 | nd_s10 | nd_s10 |
| N8 | 48 | 40 | nd_s11 | nd_s11 |
| M8 | 49 | 41 | nd_s12 | nd_s12 |
| L8 | 50 | 42 | nd_s13 | nd_s13 |
| N9 | 51 | 43 | nd_s14 | nd_s14 |
| M9 | 52 | 44 | nd_s15 | nd_s15 |
| N10 | 53 | 45 | avee | avee |
| L9 | 54 | 46 | avdd | avdd |
| M10 | 55 | 47 | nd_s16 | nd_s16 |
| N11 | 56 | 48 | nd_s17 | nd_s17 |
| N12 | 57 | 49 | nd_s18 | nd_s18 |
| L10 | 58 | 50 | nc | agnd |
| M11 | 59 | | | agnd |
| | | | | |

| N13 | 60 | | | die_attach |
|------------|------------|--------------|---------------|---------------|
| north | | | | |
| L11 | 61 | | | agnd |
| M12 | 62 | | | agnd |
| M13 | 63 | | | agnd |
| K11 | 64 | 51 | vl | vl |
| L12 | 65 | 52 | vh | vh |
| L13 | 66 | 53 | phi2_oa_even | phi2_oa_even |
| K12 | 67 | 54 | nphi3_oa_even | nphi3_oa_even |
| J11 | 68 | 55 | phicm1_even | phicm1_even |
| K13 | 69 | 56 | phicm2_even | phicm2_even |
| J12 | 70 | 57 | phi1_pa_even | phi1_pa_even |
| J13 | 71 | 58 | phi2_pa_even | phi2_pa_even |
| H11 | 72 | 59 | phi3_pa_even | phi3_pa_even |
| H12 | 73 | 60 | phi4_pa_even | phi4_pa_even |
| H13 | 74 | 61 | phi1_la_even | phi1_la_even |
| G12 | 7 5 | 62 | phi2_la_even | phi2_la_even |
| G11 | 76 | 63 | phi3_la_even | phi3_la_even |
| G13 | 77 | 64 | phi2_oa_odd | phi2_oa_odd |
| F13 | 78 | 65 | nphi3_oa_odd | nphi3_oa_odd |
| F12 | 79 | 66 | phicm1_cdd | phicm1_odd |
| F11 | 80 | 67 | phicm2_odd | phicm2_odd |
| E13 | 81 | 68 | phi1_pa_odd | phi1_pa_odd |
| E12 | 82 | 69 | phi2_pa_odd | phi2_pa_odd |
| D13 | 83 | 70 | phi3_pa_odd | phi3_pa_odd |
| E11 | 84 | 71 | phi4_pa_odd | phi4_pa_odd |
| D12 | 85 | 72 | phi1_la_odd | $phi1_la_odd$ |
| C13 | 86 | 73 | phi2_la_odd | phi2_la_odd |
| B13 | 87 | 74 | phi3_la_odd | phi3_la_odd |
| D11 | 88 | 75 | Ii C | agnd |
| C12 | 89 | | | agnd |
| A13 | 90 | | | die attach |
| west | | | | |
| C11 | 9 î | | | agnd |
| B12 | 92 | | | agnd |
| A12 | 93 | | | agr.d. |
| C10 | 94 | · ? 6 | dvdd | dvdd |
| B11 | 95 | 77 | dvee | dvee |
| A11 | 96 | 78 | agnd | agnd |
| B10 | 97 | 79 | biasoa2_r3 | biasoa2_r3 |

| С9 | 98 | 80 | avdd | avdd |
|------------|-----|-----|------------|------------|
| A10 | 99 | 81 | avee | avee |
| B9 | 100 | 32 | nc | agnd |
| A9 | 101 | 83 | biasoa1_r3 | biasoa1_r3 |
| C8 | 102 | 84 | biascp_r3 | biascp_r3 |
| B8 | 103 | 85 | nc | agnd |
| 84 | 104 | 86 | efvdd | efvdd |
| B7 | 105 | 87 | biasoa2_r2 | biasoa2_r2 |
| C7 | 106 | 88 | vcmref | vcmref |
| ∆7 | 107 | 89 | vp1 | vp1 |
| A 6 | 108 | 90 | vp2 | vp2 |
| B6 | 109 | 91 | biasoa1_r2 | biasoa1_r2 |
| C6 | 110 | 92 | biascp_r2 | biascp_r2 |
| A 5 | 111 | 93 | vr1 | vr1 |
| B 5 | 112 | 94 | vr2 | vr2 |
| A4 | 113 | 95 | biasoa2_r1 | biasoa2_r1 |
| C5 | 114 | 96 | avdd | avdd |
| B4 | 115 | 97 | avee | avee |
| A 3 | 116 | 98 | agnd | agnd |
| A 2 | 117 | 99 | biasoa1_r1 | biasoa1_r1 |
| C4 | 118 | 100 | biascp_r1 | biascp_r1 |
| В3 | 119 | | | agnd |
| A1 | 120 | | | die_attach |
| | | | | |

Appendix C

Digital Self-Calibration Simulator C Code

C.1 header.h

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <math.h>
#include <malloc.h>

#define VREF 5.0

typedef struct
{
  int    output_word;
  double output_residue;
  int    sahold_output_word;
  double sahold_output_residue;
  int    stage_output_residue;
  int    stage_output_residue;
} OUTPUT;
```

C.2 main.c

```
analog-to-digital converter. Variables of length double are
   used to represent the analog variables.
   ----- */
#include "header.h"
static int d[32];
main()
{
FILE *file_output;
int i_max, i, j, k, m, n, grain_factor, *code, code_temp, temp;
int w, y, s[32][4], level, cal;
int reduce_step, kmax_dnl, kmax_inl;
int (*proc)();
int (*run_proc)();
double vin, error[32][4], epsilon, delta, *dlevel, *diff;
double *dnl, *inl;
double *rdnl, *rinl;
double r, dnl_sum;
extern int stage();
extern int sahold();
extern int calibrate();
extern int plev00();
extern int plev01();
extern int plev02();
extern int plev03();
extern int plev04();
extern int plev05();
extern int plev06();
extern int plev07();
extern int plev08();
extern int plev09();
extern int plev10();
extern int plev11();
extern int plev12();
extern int plev13();
extern int plev14();
extern int plev15();
extern int ptest00();
extern int ptest01();
extern int ptest02();
extern int ptest03();
extern int ptest04();
```

```
extern int ptest05();
extern int ptest06();
extern int ptest07();
extern int ptest08();
extern int ptest09();
extern int ptest10();
extern int ptest11();
extern int ptest12();
extern int ptest13();
extern int ptest14();
extern int ptest15();
OUTPUT out;
/* memory allocation and initialization */
     = calloc(524288, sizeof(int));
dlevel = calloc(524288, sizeof(double));
diff = calloc(524289, sizeof(double));
dnl = calloc(524289, sizeof(double));
     = calloc(524289, sizeof(double));
inl
rdnl = calloc(524289, sizeof(double));
rinl = calloc(524289, sizeof(double));
for(k=31; k>=0; k--)
    for(j=3; j>=0; j--)
{
error[k][j] = 0.0;
s[k][j] = 0;
}
   input
                 description
                 number of MX2 stages with G = 2.
   n
                 number of MX2 stages with G < 2.
   \mathbf{m}
                 (diagnostic) test analog input.
   grain_factor number of steps searched per LSB.
   reduce_step
                 output data reduction factor.
   d[k]
                 (diagnostic) number of raw codes to be mapped to same output
                 code around major carry transition point for respective MX2
                 stage k.
   error[k][j]
                 gain and offset errors modeled for SHA and MX2 stage k.
*/
```

```
scanf("%d", &n);
scanf("%d", &m);
scanf("%201f", &vin);
scanf("%d", &grain_factor);
scanf("%d", &reduce_step);
for(k=14; k>=0; k--)
    scanf("%d", &d[k]);
for(k=18; k>=1; k--)
    for(j=3; j>=0; j--)
scanf("%201f", &error[k][j]);
/* print out input values */
printf("n = %d\n", n);
printf("m = \frac{d}{n}, m);
printf("vin = %.8lf\n", vin);
printf("grain_factor = %d\n", grain_factor);
printf("reduce_step = %d\n", reduce_step);
for(k=14; k>=0; k--)
    printf("d[%d] = %d \n", k, d[k]);
for(k=18; k>=1; k--)
    for(j=3; j>=0; j--)
printf("error[%d][%d] = %.8lf \n", k, j, error[k][j]);
printf("\n");
/* select run_proc based on m */
switch( m )
    {
  case 1: run_proc = ptest01;
    break;
  case 2: run_proc = ptest02;
    break;
  case 3: run_proc = ptest03;
    break;
  case 4: run_proc = ptest04;
    break:
  case 5: run_proc = ptest05;
```

```
break:
  case 6: run_proc = ptest06;
    break;
  case 7: run_proc = ptest07;
    break:
  case 8: run_proc = ptest08;
    break:
  case 9: run_proc = ptest09;
    break;
  case 10: run_proc = ptest10;
    break;
  case 11: run_proc = ptest11;
    break;
  case 12: run_proc = ptest12;
    break:
  case 13: run_proc = ptest13;
    break:
  case 14: run_proc = ptest14;
    break:
  case 15: run_proc = ptest15;
    break;
  default:
    {
    printf("ERROR ERROR ERROR\n");
    printf("int _m_ out of range.\n");
    exit( 1 );
    }
    }
/* assign LSB size */
delta = 2*VREF/pow(2.0, (double) (n + m + 1));
/* assign increment per LSB */
epsilon = delta/grain_factor;
/* begin calibration mode */
level = 0;
proc = plev01;
calibrate(n, level, error, s, proc, &out);
printf("%d %d %d %d\n", s[0][0], s[0][1], s[0][2], s[0][3]);
if ((s[0][1] - s[0][0]) != (s[0][3] - s[0][2]))
```

171

```
ſ
    printf("ERROR ERROR ERROR");
    printf("(s[0][1] - s[0][0]) != (s[0][3] - s[0][2])");
    exit( 1 );
    }
cal = 0;
ptest01(n, vin, error, s, cal, &out);
printf("%d\n", out.output_word);
level = 1;
proc = plev02;
calibrate(n, level, error, s, proc, &out);
printf("%d %d %d %d\n", s[1][0], s[1][1], s[1][2], s[1][3]);
if ((s[1][1] - s[1][0]) != (s[1][3] - s[1][2]))
    printf("ERROR ERROR ERROR");
    printf("(s[1][1] - s[1][0]) != (s[1][3] - s[1][2])");
    exit( 1 );
    }
cal = 0;
ptest02(n, vin, error, s, cal, &out);
printf("%d\n", out.output_word);
level = 2;
proc = plev03;
calibrate(n, level, error, s, proc, &out);
printf("%d %d %d %d\n", s[2][0], s[2][1], s[2][2], s[2][3]);
if ((s[2][1] - s[2][0]) != (s[2][3] - s[2][2]))
    printf("ERROR ERROR ERROR");
    printf("(s[2][1] - s[2][0]) != (s[2][3] - s[2][2])");
    exit( 1 );
    }
cal = 0;
ptest03(n, vin, error, s, cal, &out);
printf("%d\n", out.output_word);
```

```
level = 3;
proc = plev04;
calibrate(n, level, error, s, proc, &out);
printf("%d %d %d %d\n", s[3][0], s[3][1], s[3][2], s[3][3]);
if ((s[3][1] - s[3][0]) != (s[3][3] - s[3][2]))
    -
   printf("ERROR ERROR ERROR");
    printf("(s[3][1] - s[3][0]) != (s[3][3] - s[3][2])");
    exit( 1 );
    }
cal = 0;
ptest04(n, vin, error, s, cal, &out);
printf("%d\n", out.output_word);
level = 4;
proc = plev05;
calibrate(n, level, error, s, proc, &out);
printf("%d %d %d %d\n", s[4][0], s[4][1], s[4][2], s[4][3]);
if ((s[4][1] - s[4][0]) != (s[4][3] - s[4][2]))
    ſ
   printf("ERROR ERROR ERROR");
    printf("(s[4][1] - s[4][0]) != (s[4][3] - s[4][2])");
    exit( 1 );
    }
cal = 0;
ptest05(n, vin, error, s, cal, &out);
printf("%d\n", out.output_word);
level = 5;
proc = plav06;
calibrato(n, level, error, s, proc, &out);
printf("%d %d %d %d\n", s[5][0], s[5][1], s[5][2], s[5][3]);
if ((s[5][1] - s[5][0]) != (s[5][3] - s[5][2]))
   printf("ERROR ERROR ERROR");
   printf("(s[5][1] - s[5][0]) != (s[5][3] - s[5][2])");
    exit( 1 );
```

```
}
cal = 0;
ptest06(n, vin, error, s, cal, tout);
printf("%d\n", out.output_word);
level = 6;
proc = plev07;
calibrate(n, level, error, s, proc, &out);
printf("%d %d %d %d\n", s[6][0], s[6][1], s[6][2], s[6][3]);
if ((s[6][1] - s[6][0]) != (s[6][3] - s[6][2]))
    ſ
    printf("ERROR ERROR ERROR");
    printf("(s[6][1] - s[6][0]) != (s[6][3] - s[6][2])");
    exit( 1 ):
    }
cal = 0;
ptest07(n, vin, error, s, cal, &out);
printf("%d\n", out.output_word);
level = 7;
proc = plev08;
calibrate(n, level, error, s, proc, &out);
printf("%d %d %d %d\n", s[7][0], s[7][1], s[7][2], s[7][3]);
if ((s[7][1] - s[7][0]) != (s[7][3] - s[7][2]))
    {
   printf("ERROR ERROR ERROR");
   printf("(s[7][1] - s[7][0]) != (s[7][3] - s[7][2])");
    exit( 1 );
cal = 0;
ptest08(n, vin, error, s, cal, &out);
printf("%d\n", out.output_word);
level = 8;
proc = plev09;
calibrate(n, level, error, s, proc, &out);
printf("%d %d %d %d\n", s[8][0], s[8][1], s[8][2], s[8][3]);
```

```
if ((s[8][1] - s[8][0]) != (s[8][3] - s[8][2]))
    printf("ERROR ERROR ERROR");
    printf("(s[8][1] - s[8][0]) != (s[8][3] - s[8][2])");
    exit( 1 );
    }
cal = 0;
ptest09(n, vin, error, s, cal, &out);
printf("%d\n", out.output_word);
level = 9;
proc = plev10;
calibrate(n, level, error, s, proc, &out);
printf("%d %d %d %d\n", s[9][0], s[9][1], s[9][2], s[9][3]);
if ((s[9][1] - s[9][0]) != (s[9][3] - s[9][2]))
    ſ
    printf("ERROR ERROR ERROR");
    printf("(s[9][1] - s[9][0]) != (s[9][3] - s[9][2])");
    exit( 1 );
    }
cal = 0;
ptest10(n, vin, error, s, cal, &out);
printf("%d\n", out.output_word);
level = 10;
proc = plev11;
calibrate(n, level, error, s, proc, &out);
printf("%d %d %d %d\n", s[10][0], s[10][1], s[10][2], s[10][3]);
if ((s[10][1] - s[10][0]) != (s[10][3] - s[10][2]))
    {
    printf("ERROR ERROR ERROR");
   printf("(s[10][1] - s[10][0]) != (s[10][3] - s[10][2])");
    exit( 1 );
    }
cal = 0;
ptest11(n, vin, error, s, cal, &out);
```

```
printf("%d\n", out.output_word);
level = 11:
proc = plev12;
calibrate(n, level, error, s, proc, &out);
printf("%d %d %d %d\n", s[11][0], s[11][1], s[11][2], s[11][3]);
if ((s[11][1] - s[11][0]) != (s[11][3] - s[11][2]))
    printf("ERROR ERROR ERROR");
    printf("(s[11][1] - s[11][0]) != (s[11][3] - s[11][2])");
    exit( 1 );
cal = 0;
ptest12(n, vin, error, s, cal, &out);
printf("%d\n", out.output_word);
level = 12;
proc = plev13;
calibrate(n, level, error, s, proc, &out);
printf("%d %d %d %d\n", s[12][0], s[12][1], s[12][2], s[12][3]);
if ((s[12][1] - s[12][0]) != (s[12][3] - s[12][2]))
    printf("ERROR ERROR ERROR");
   printf("(s[12][1] - s[12][0]) != (s[12][3] - s[12][2])");
    exit( 1 );
    }
cal = 0;
ptest13(n, vin, error, s, cal, &out);
printf("%d\n", out.output_word);
level = 13;
proc = plev14;
calibrate(n, level, error, s, proc, &out);
printf("%d %d %d %d\n", s[13][0], s[13][1], s[13][2], s[13][3]);
if ((s[13][1] - s[13][0]) != (s[13][3] - s[13][2]))
    ſ
   printf("ERROR ERROR ERROR");
```

```
printf("(s[13][1] - s[13][0]) != (s[13][3] - s[13][2])");
   exit( 1 );
   }
cal = 0;
ptest14(n, vin, error, s, cal, &out);
printf("%d\n", out.output_word);
level = 14;
proc = plev15;
calibrate(n, level, error, s, proc, &out);
printf("%d %d %d %d\n", s[14][0], s[14][1], s[14][2], s[14][3]);
if ((s[14][1] - s[14][0]) != (s[14][3] - s[14][2]))
   printf("ERROR ERROR ERROR");
    printf("(s[14][1] - s[14][0]) != (s[14][3] - s[14][2])");
    exit( 1 );
    }
cal = 0;
ptest15(n, vin, error, s, cal, &out);
printf("%d\n", out.output_word);
/* begin linearity search */
vin = -VREF;
cal = 0;
(*run_proc)(n, vin, error, s, cal, &out);
temp = out.output_word;
i = 0;
code_temp = temp;
code[i] = temp;
dlevel[i] = vin;
while (vin < VREF)
cal = 0;
(*run_proc)(n, vin, error, s, cal, &out);
temp = out.output_word;
if (temp != code_temp)
```

```
{
    i = i + 1;
    i_max = i;
    code_temp = temp;
    code[i] = code_temp;
    dlevel[i] = vin;
    diff[i - 1] = dlevel[i] - dlevel[i - 1];
vin = vin + epsilon;
diff[i_max] = VREF - dlevel[i_max];
/* recompute delta based on the maximum output code. */
delta = 2*VREF/(1.0 + (double) i_max);
/* now compute dnl[j], the true differential nonlinearity. */
for(j=0;j<=i_max;j++)</pre>
dnl[j] = (diff[j]/delta) - 1.0;
/* now compute inl[j], the true integral nonlinearity. */
inl[0] = dnl[0];
for(j=1;j<=i_max;j++)
inl[j] = inl[j-1] + dnl[j];
r = (double) reduce_step;
/* DNL output */
printf("DNL\n");
/* print out number of decision levels i_max */
printf("number of decision levels i_max: %d\n", i_max);
/* print out value of code[i_max] */
printf("value of code[i_max]: %d\n", code[i_max]);
/* reduce differential nonlinearity data length */
i = 0;
```

```
k = 0;
while(i <= i_max)</pre>
    {
    dnl_sum = 0;
    for(j = i; j <= i + reduce_step - 1; j++)
{
        dnl_sum = dnl_sum + dnl[j];
}
    rdnl[k] = dnl_sum / r;
    i = i + reduce_step;
    k = k + 1:
    }
kmax_dnl = k - 1;
printf("kmax_dnl = %ld\n", kmax_dnl);
/* INL output */
printf("INL\n");
/* print out number of decision levels i_max */
printf("number of decision levels i_max: %d\n", i_max);
/* print out value of code[i_max] */
printf("value of code[i_max]: %d\n", code[i_max]);
/* reduce integral nonlinearity length */
i = 0;
k = 0;
while(i <= i_max)</pre>
   rinl[k] = inl[i + reduce_step - 1] / r;
    i = i + reduce_step;
    k = k + 1;
    }
kmax_inl = k - 1;
printf("kmax_inl = %ld\n", kmax_inl);
/* write out DNL and INL data */
printf("begin writing dnl data \n");
file_output = fopen("dnl.adc", "w+");
```

```
for(i = 0; i <= i_max; i++)
   fprintf(file_output, "%.4lf \n", dnl[i]);
fclose(file_output);
printf("begin writing inl data \n");
file_output = fopen("inl.adc", "w+");
for(i = 0; i \le i_{max}; i++)
   fprintf(file_output, "%.4lf \n", inl[i]);
fclose(file_output);
printf("begin writing reduced dnl data \n");
file_output = fopen("rdnl.adc", "w+");
for(i = 0; i <= kmax_dnl; i++)</pre>
   fprintf(file_output, "%.4lf \n", rdnl[i]);
fclose(file_output);
printf("begin writing reduced inl data \n");
file_output = fopen("rinl.adc", "w+");
for(i = 0; i <= kmax_inl; i++)
   fprintf(file_output, "%.4lf \n", rinl[i]);
fclose(file_output);
end: exit( 0 );
}
Function stage(k, vin, xin, error, ptr)
  This function returns an int representing the kth-stage reference
  nonrestoring pipeline conversion of the input signal vin and state
  xin. Error coefficients are passed with the array _error_.
  output of the pipeline section is passed to a struct referenced by
  pointer ptr. Array _error_ map:
  error[j][3] <-> amplifier_offset[j]
  error[j][2] <-> comparator_offset[j]
```

```
error[j][1] <-> alpha_2[j]
  error[j][0] <-> a pha_1[j]
  int stage(k, vin, xin, error, ptr)
int k, xin;
double vin, error[32][4];
OUTPUT *ptr;
-{
int x;
double residue;
extern int testbit();
extern int setbit();
if (xin == 1)
   residue = (2 + \operatorname{error}[k][0])*\operatorname{vin} / (1 + \operatorname{error}[k][0] + \operatorname{error}[k][1]) -
VREF/(1 + error[k][0] + error[k][1]) + error[k][3];
   residue = (2 + error[k][0])*vin / (1 + error[k][0] + error[k][1]) +
VREF/(1 + error[k][0] + error[k][1]) + error[k][3];
x = 0;
if (residue >= errcr[k][2])
   x = 1;
else
   x = 0:
ptr->stage_output_word = x;
ptr->stage_output_residue = residue;
}
Function sahold(k, vin, error, ptr)
  This function returns an int representing a 1-bit conversion of the input
  signal vin with a sample-and-hold in the k-th stage of the pipeline.
  Error coefficients are passed with the array _error_. The output of the
  sample-and-hold is passed to a struct referenced by pointer ptr.
  int sahold(k, vin, error, ptr)
int k;
double vin, error[32][4];
```

```
OUTPUT *ptr;
int x;
double residue;
extern int testbit();
extarn int setbit();
x = 0;
if (vin >= error[k][2])
 x = 1;
else
 x = 0;
residue = vin * (1 + error[k][0]) + error[k][3];
ptr->sahold_output_word = x;
ptr->sahold_output_residue = residue;
}
Function calibrate(n, level, error, s, proc, ptr)
  int calibrate(n, level, error, s, proc, ptr)
int n, level, s[32][4];
double error[32][4];
int (*proc)();
OUTPUT *ptr;
int w;
double r;
r = -VREF;
w = 0;
(*proc)(n, r, w, error, s, 1, ptr);
s[level][0] = ptr->output_word;
r = 0;
w = 0:
(*proc)(n, r, w, error, s, 1, ptr);
```

```
s[level][1] = ptr->output_word;
r = 0:
w = 1;
(*proc)(n, r, w, error, s, 1, ptr);
s[level][2] = ptr->output_word;
r = VREF;
w = 1;
(*proc)(n, r, w, error, s, 1, ptr);
s[level][3] = ptr->output_word;
Function plev00(n, rin, xin, error, s, cal, ptr)
  int plev00(n, rin, xin, error, s, cal, ptr)
int n, xin, s[32][4], cal;
double rin, error[32][4];
OUTPUT *ptr;
int k, y, w;
double r;
r = rin;
w = xin;
y = 0;
for(k=n; k>-1; k--)
   stage(k, r, w, error, ptr);
   r = ptr->stage_output_residue;
   w = ptr->stage_output_word;
   y = setbit(y, k - 1, w);
ptr->output_word = y;
ptr->output_residue = r;
}
Function plev01(n, rin, xin, error, s, cal, ptr)
```

```
int plev01(n, rin, xin, error, s, cal, ptr)
int n, xin, s[32][4], cal;
double rin, error[32][4];
OUTPUT *ptr;
int y, w, tmp;
double r;
r = rin;
w = xin;
y = 0;
stage(n + 1, r, w, error, ptr);
r = ptr->stage_output_residue;
u = ptr->stage_output_word;
y = setbit(y, n, w);
plev00(n, r, w, error, s, 0, ptr);
tmp = ptr->output_word + y;
if (cal = 0)
if (xin == 0)
tmp = tmp - s[0][0];
tmp = tmp - s[0][2] - d[0] + s[0][1] - s[0][0] - d[0];
ptr->output_word = tmp;
Function plev02(n, rin, xin, error, s, cal, ptr)
  */
int plev02(n, rin, xin, error, s, cal, ptr)
int n, xin, s[32][4], cal;
double rin, error[32][4];
OUTPUT *ptr;
int y, w, tmp;
```

```
double r:
r = rin:
w = xin;
y = 0;
stage(n + 2, r, w, error, ptr);
r = ptr->stage_output_residue;
w = ptr->stage_output_word;
y = setbit(y, n + 1, w);
plev01(n, r, w, error, s, 0, ptr);
tmp = ptr->output_word;
if (cal == 0)
if (xin == 0)
tmp = tmp - s[1][0];
tmp = tmp - s[1][2] - d[1] + s[1][1] - s[1][0] - d[1];
ptr->output_word = tmp;
}
Function plev03(n, rin, xin, error, s, cal, ptr)
  int plev03(n, rin, xin, error, s, cal, ptr)
int n, xin, s[32][4], cal;
double rin, error[32][4];
OUTPUT *ptr;
int y, w, tmp;
double r;
r = rin;
w = xin;
y = 0;
stage(n + 3, r, w, error, ptr);
r = ptr->stage_output_residue;
w = ptr->stage_output_word;
y = setbit(y, n + 2, w);
```

```
plev02(n, r, w, error, s, 0, ptr);
tmp = ptr->output_word;
if (cal == 0)
if (xin == 0)
tmp = tmp - a[2][0];
tmp = tmp - s[2][2] - d[2] + s[2][1] - s[2][0] - d[2];
ptr->output_word = tmp;
Function plev04(n, rin, xin, error, s, cal, ptr)
  int plev04(n, rin, xin, error, s, cal, ptr)
int n, xin, s[32][4], cal;
double rin, error[32][4];
OUTPUT *ptr;
int y, w, tmp;
double r;
r = rin;
w = xin;
y = 0;
stage(n + 4, r, w, error, ptr);
r = ptr->stage_output_residue;
w = ptr->stage_output_word;
y = setbit(y, n + 3, w);
plev03(n, r, w, error, s, 0, ptr);
tmp = ptr->output_word;
if (cal == 0)
if (xin == 0)
tmp = tmp - s[3][0];
tmp = tmp - s[3][2] - d[3] + s[3][1] - s[3][0] - d[3];
```

```
ptr->output_word = tmp;
Function plev05(n, rin, xin, error, s, cal, ptr)
  int plev05(1., rin, xin, error, s, cal, ptr)
int n, xin, s[32][4], cal;
double rin, error[32][4];
OUTPUT *ptr;
int y, w, tmp;
double r;
r = rin;
w = xin;
y = 0;
stage(n + 5, r, w, error, ptr);
r = ptr->stage_output_residue;
w = ptr->stage_output_word;
y = setbit(y, n + 4, w);
plev04(n, r, w, error, s, 0, ptr);
tmp = ptr->output_word;
if (cal == 0)
if (xin == 0)
tmp = tmp - s[4][0];
else
tmp = tmp - s[4][2] - d[4] + s[4][1] - s[4][0] - d[4];
ptr->output_word = tmp;
}
Function plev06(n, rin, xin, error, s, cal, ptr)
```

```
int plev06(n, rin, xin, error, s, cal, ptr)
int n, xin, s[32][4], cal;
double rin, error[32][4];
OUTPUT *ptr;
int y, w, tmp;
double r;
r = rin;
w = xin;
y = 0;
stage(n + 6, r, w, error, ptr);
r = ptr->stage_output_residue;
w = ptr->stage_output_word;
y = setbit(y, n + 5, w);
plev05(n, r, w, error, s, 0, ptr);
tmp = ptr->output_word;
if (cal == 0)
if (xin == 0)
tmp = tmp - s[5][0];
else
tmp = tmp - s[5][2] - d[5] + s[5][1] - s[5][0] - d[5];
ptr->output_word = tmp;
}
Function plev07(n, rin, xin, error, s, cal, ptr)
  _______ */
int plev07(n, rin, xin, error, s, cal, ptr)
int n, xin, s[32][4], cal;
double rin, error[32][4];
OUTPUT *ptr;
-{
int y, w, tmp;
double r;
```

```
r = rin;
w = xin;
y = 0;
stage(n + 7, r, w, error, ptr);
r = ptr->stage_output_residue;
w = ptr->stage_output_word;
y = setbit(y, n + 6, w);
plev06(n, r, w, error, s, 0, ptr);
tmp = ptr->output_word;
if (cal == 0)
ſ
if (xin == 0)
tmp = tmp - s[6][0];
else
tmp = tmp - s[6][2] - d[6] + s[6][1] - s[6][0] - d[6];
ptr->output_word = tmp;
Function plev08(n, rin, xin, error, s, cal, ptr)
  int plev08(n, rin, xin, error, s, cal, ptr)
int n, xin, s[32][4], cal;
double rin, error[32][4];
OUTPUT *ptr;
{
int y, w, tmp;
double r;
r = rin;
w = xin;
y = 0;
stage(n + 8, r, w, error, ptr);
r = ptr->stage_output_residue;
w = ptr->stage_output_word;
y = setbit(y, n + 7, w);
plev07(n, r, w, arror, s, 0, ptr);
```

```
tmp = ptr->output_word;
if (cal == 0)
₹
if (xin == 0)
tmp = tmp - s[7][0];
else
tmp = tmp - s[7][2] - d[7] + s[7][1] - s[7][0] - d[7];
ptr->output_word = tmp;
Function plev09(n, rin, xin, error, s, cal, ptr)
  int plev09(n, rin, xin, error, s, cal, ptr)
int n, xin, s[32][4], cal;
double rin, error[32][4];
OUTPUT *ptr;
{
int y, w, tmp;
double r;
r = rin;
w = xin;
v = 0:
stage(n + 9, r, w, error, ptr);
r = ptr->stage_output_residue;
w = ptr->stage_output_word;
y = setbit(y, n + 8, w);
plev08(n, r, w, error, s, 0, ptr);
tmp = ptr->output_word;
if (cal == 0)
-{
if (xin == 0)
tmp = tmp - s[8][0];
else
tmp = tmp - s[8][2] - d[8] + s[8][1] - s[8][0] - d[8];
}
```

```
ptr->output_word = tmp;
Function plev10(n, rin, xin, error, s, cal, ptr)
  int plev10(n, rin, xin, error, s, cal, ptr)
int n, xin, s[32][4], cal;
double rin, error[32][4];
OUTPUT *ptr;
int y, w, tmp;
double r;
r = rin;
w = xin;
y = 0;
stage(n + 10, r, w, error, ptr);
r = ptr->stage_output_residue;
w = ptr->stage_output_word;
y = setbit(y, n + 9, w);
plev09(n, r, w, error, s, 0, ptr);
tmp = ptr->output_word;
if (cal == 0)
{
if (xin == 0)
tmp = tmp - s[9][0];
else
tmp = tmp - s[9][2] - d[9] + s[9][1] - s[9][0] - d[9];
ptr->output_word = tmp;
Function plev11(n, rin, xin, error, s, cal, ptr)
  */
int plev11(n, rin, xin, error, s, cal, ptr)
```

```
int n, xin, s[32][4], cal;
double rin, error[32][4];
OUTPUT *ptr;
int y, w, tmp;
double r;
r = rin;
w = xin:
y = 0;
stage(n + 11, r, w, error, ptr);
r = ptr->stage_output_residue;
w = ptr->stage_output_word;
y = setbit(y, n + 10, w);
plev10(n, r, w, error, s, 0, ptr);
tmp = ptr->output_word;
if (cal == 0)
{
if (xin == 0)
tmp = tmp - s[10][0];
else
tmp = tmp - s[10][2] - d[10] + s[10][1] - s[10][0] - d[10];
ptr->output_word = tmp;
Function plev12(n, rin, xin, error, s, cal, ptr)
  */
int plev12(n, rin, xin, error, s, cal, ptr)
int n, xin, s[32][4], cal;
double rin, error[32][4];
OUTPUT *ptr;
{
int y, w, tmp;
double r;
r = rin;
w = xin;
```

```
y = 0;
stage(n + 12, r, w, error, ptr);
r = ptr->stage_output_residue;
w = ptr->stage_output_word;
y = setbit(y, n + 11, w);
plev11(n, r, w, error, s, 0, ptr);
tmp = ptr->output_word;
if (cal == 0)
if (xin == 0)
tmp = tmp - s[11][0];
else
tmp = tmp - s[11][2] - d[11] + s[11][1] - s[11][0] - d[11];
ptr->output_word = tmp;
Function plev13(n, rin, xin, error, s, cal, ptr)
  int plev13(n, rin, xin, error, s, cal, ptr)
int n, xin, s[32][4], cal;
double rin, error[32][4];
OUTPUT *ptr;
{
int y, w, tmp;
double r;
r = rin;
w = xin;
y = 0;
stage(n + 13, r, w, error, ptr);
r = ptr->stage_output_residue;
w = ptr->stage_output_word;
y = setbit(y, n + 12, w);
plev12(n, r, w, error, s, 0, ptr);
tmp = ptr->output_word;
if (cal == 0)
```

```
if (xin == 0)
tmp = tmp - s[12][0];
else
tmp = tmp - s[12][2] - d[12] + s[12][1] - s[12][0] - d[12];
ptr->output_word = tmp;
Function plev14(n, rin, xin, error, s, cal, ptr)
  */----*
int plev14(n, rin, xin, error, s, cal, ptr)
int n, xin, s[32][4], cal;
double rin, error[32][4];
OUTPUT *ptr;
{
int y, w, tmp;
double r;
r = rin;
w = xin;
y = 0;
stage(n + 14, r, w, error, ptr);
r = ptr->stage_output_residue;
w = ptr->stage_output_word;
y = setbit(y, n + 13, w);
plev13(n, r, w, error, s, 0, ptr);
tmp = ptr->output_word;
if (cal == 0)
}
if (xin == 0)
tmp = tmp - s[13][0];
else
tmp = tmp - s[13][2] - d[13] + s[13][1] - s[13][0] - d[13];
ptr->output_word = tmp;
```

```
/* ----
  Function plev15(n, rin, xin, error, s, cal, ptr)
  int plev15(n, rin, xin, error, s, cal, ptr)
int n, xin, s[32][4], cal;
double rin, error[32][4];
OUTPUT *ptr;
-{
int y, w, tmp;
double r:
r = rin;
w = xin;
y = 0;
stage(n + 15, r, w, error, ptr);
r = ptr->stage_output_residue;
w = ptr->stage_output_word;
y = setbit(y, n + 14, w);
plev14(n, r, w, error, s, 0, ptr);
tmp = ptr->output_word;
if (cal == 0)
if (xin == 0)
tmp = tmp - s[14][0];
else
tmp = tmp - s[14][2] - d[14] + s[14][1] - s[14][0] - d[14];
ptr->output_word = tmp;
}
      1
Function ptest00(n, vin, error, s, cal, ptr);
  int ptest00(n, vin, error, s, cal, ptr)
int n, s[32][4], cal;
double vin, error[32][4];
OUTPUT *ptr;
```

195

```
ſ
int w;
double r;
sahold(n + 1, vin, error, ptr);
w = ptr->sahold_output_word;
r = ptr->sahold_output_residue;
plev00(n, r, w, error, s, cal, ptr);
Function ptest01(n, vin, error, s, cal, ptr);
  int ptest01(n, vin, error, s, cal, ptr)
int n, s[32][4], cal;
double vin, error[32][4];
OUTPUT *ptr;
{
int w:
double r;
sahold(n + 2, vin, error, ptr);
w = ptr->sahold_output_word;
r = ptr->sahold_output_residue;
plev01(n, r, w, error, s, cal, ptr);
Function ptest02(n, vin, error, s, cal, ptr);
  int ptest02(n, vin, error, s, cal, ptr)
int n, s[32][4], cal;
double vin, error[32][4];
OUTPUT *ptr;
ſ
int w;
double r;
```

```
sahold(n + 3, vin, error, ptr);
w = ptr->sahold_output_word;
r = ptr->sahold_output_residue;
plev02(n, r, w, error, s, cal, ptr);
Function ptest03(n, vin, error, s, cal, ptr);
  int ptest03(n, vin, error, s, cal, ptr)
int n, s[32][4], cal;
double vin, error[32][4];
OUTPUT *ptr;
-{
int w;
double r;
sahold(n + 4, vin, error, ptr);
w = ptr->sahold_output_word;
r = ptr->sahold_output_residue;
plev03(n, r, w, error, s, cal, ptr);
}
Function ptest04(n, vin, error, s, cal, ptr);
  int ptest04(n, vin, error, s, cal, ptr)
int n, s[32][4], cal;
double vin, error[32][4];
OUTPUT *ptr;
int w;
double r;
sahold(n + 5, vin, error, ptr);
w = ptr->sahold_output_word;
r = ptr->sahold_output_residue;
plev04(n, r, w, error, s, cal, ptr);
```

```
}
Function ptest05(n, vin, error, s, cal, ptr);
  int ptest05(n, vin, error, s, cal, ptr)
int n, s[32][4], cal;
double vin, error[32][4];
OUTPUT *ptr;
{
int w;
double r;
sahold(n + 6, vin, error, ptr);
w = ptr->sahold_output_word;
r = ptr->sahold_output_residue;
plev05(n, r, w, error, s, cal, ptr);
Function ptest06(n, vin, error, s, cal, ptr);
  int ptest06(n, vin, error, s, cal, ptr)
int n, s[32][4], cal;
double vin, error[32][4];
OUTPUT *ptr;
{
int w;
double r;
sahold(n + 7, vin, error, ptr);
w = ptr->sahold_output_word;
r = ptr->sahold_output_residue;
plev06(n, r, w, error, s, cal, ptr);
}
Function ptest07(n, vin, error, s, cal, ptr);
```

```
int ptest07(n, vin, error, s, cal, ptr)
int n, s[32][4], cal;
double vin, error[32][4];
OUTPUT *ptr;
int w:
double r;
sahold(n + 8, vin, error, ptr);
w = ptr->sahold_output_word;
r = ptr->sahold_output_residue;
plev07(n, r, w, error, s, cal, ptr);
Function ptest08(n, vin, error, s, cal, ptr);
  int ptest08(n, vin, error, s, cal, ptr)
int n, s[32][4], cal;
double vin, error[32][4];
OUTPUT *ptr;
{
int w:
double r;
sahold(n + 9, vin, error, ptr);
w = ptr->sahold_output_word;
r = ptr->sahold_output_residue;
plev08(n, r, w, error, s, cal, ptr);
Function ptest09(n, vin, error, s, cal, ptr);
  int ptest09(n, vin, error, s, cal, ptr)
```

```
int n, s[32][4], cal;
double vin, error[32][4];
OUTPUT *ptr;
Ł
int w;
double r;
sahold(n + 10, vin, error, ptr);
w = ptr->sahold_output_word;
r = ptr->sahold_output_residue;
plev09(n, r, w, error, s, cal, ptr);
}
Function ptest10(n, vin, error, s, cal, ptr);
  int ptest10(n, vin, error, s, cal, ptr)
int n, s[32][4], cal;
double vin, error[32][4];
OUTPUT *ptr;
{
int w:
double r;
sahold(n + 11, vin, error, ptr);
w = ptr->sahold_output_word;
r = ptr->sahold_output_residue;
plev10(n, r, w, error, s, cal, ptr);
}
Function ptest11(n, vin, error, s, cal, ptr);
  int ptest11(n, vin, error, s, cal, ptr)
int n, s[32][4], cal;
double vin, error[32][4];
OUTPUT *ptr;
{
```

```
int w:
double r;
sahold(n + 12, vin, error, ptr);
w = ptr->sahold_output_word;
r = ptr->sahold_output_residue;
plevi1(n, r, w, error, s, cal, ptr);
Function ptest12(n, vin, error, s, cal, ptr);
  ______*/
int ptest12(n, vin, error, s, cal, ptr)
int n, s[32][4], cal;
double vin, error[32][4];
OUTPUT *ptr;
ſ
int w;
double r;
sahold(n + 13, vin, error, ptr);
w = ptr->sahold_output_word;
r = ptr->sahold_output_residue;
plev12(n, r, w, error, s, cal, ptr);
}
Function ptest13(n, vin, error, s, cal, ptr);
  int ptest13(n, vin, error, s, cal, ptr)
int n, s[32][4], cal;
double vin, error[32][4];
OUTPUT *ptr;
{
int w;
double r;
sahold(n + 14, vin, error, ptr);
```

```
w = ptr->sahold_output_word;
r = ptr->sahold_output_residue;
plev13(n, r, w, error, s, cal, ptr);
Function ptest14(n, vin, error, s, cal, ptr);
  ______ */
int ptest14(n, vin, error, s, cal, ptr)
int n, s[32][4], cal;
double vin, error[32][4];
OUTPUT *ptr;
ſ
int w:
double r;
sahold(n + 15, vin, error, ptr);
w = ptr->sahold_output_word;
r = ptr->sahold_output_residue;
plev14(n, r, w, error, s, cal, ptr);
Function ptest15(n, vin, error, s, cal, ptr);
  int ptest15(n, vin, error, s, cal, ptr)
int n, s[32][4], cal;
double vin, error[32][4];
OUTPUT *ptr;
-
int w;
double r;
sahold(n + 16, vin, error, ptr);
w = ptr->sahold_output_word;
r = ptr->sahold_output_residue;
plev15(n, r, w, error, s, cal, ptr);
```

C.3 setbit.c

```
Andrew Karanicolas
  10-20-89
  Function: setbit(word, position, bit_value)
  This function returns an int that is the specified word with the
  specified bit position set to bit_value.
  int setbit(word, position, bit_value)
int word, position, bit_value;
int mask0, mask1, temp, output;
temp = 1;
mask0 = temp << position;</pre>
mask1 = ~mask0;
if (bit_value == 0)
 output = mask1 & word;
 output = mask0 | word;
return(output);
}
C.4 testbit.c
Andrew Karanicolas
  10-20-89
  Function: testbit(word, position)
  This function returns an int that is set according to the value
  of the bit at the specified position in the specified word. If
  the bit is set to 1, then 1 is returned, else 0 is returned.
  int testbit(word, position)
```

C.4. TESTBIT.C 203

```
int word, position;
{
int temp, output;
temp = 1;
output = temp & (word >> position);
return(output);
}
```

Appendix D

Test System C Code

D.1 acq.c

```
/* Program for Pipeline ADC Data Acquisition System
                                                                */
                                                                */
/* Andrew Karanicolas
/* Microsystems Technology Laboratory
                                                                */
                                                                */
/* 06-07-92
#include <stdio.h>
#include <conio.h>
#include <malloc.h>
                                                                */
/* base address for PXB-721 is 0x300
                                                                */
/* 8255A_3 is presently used
                                                                */
/* PORT_D3 is control register
#define PORT_A3 0x308
#define PORT_B3 0x309
#define PORT_C3 0x30A
#define PORT_D3 0x30B
/* mode 0x82 configures PORT_A3 as output, PORT_B3 as input
                                                                */
/* and PORT_C3 as output
                                                                */
#define MODE 0x82
/* set all instruction bits to 0 for full reset
                                                                */
#define RESET 0x00
                                                                */
/* instructions to read SRAM DATA_C, DATA_B, DATA_A
#define DATA_C 0x42
#define DATA_B 0x22
```

D.1. ACQ.C 205

#define DATA_A 0x02 /* data acquisition system instruction format LOADMEM 2 REQREAD 3 REQURITE 4 REQCOUNTMAR 5 REQCLRMAR 6 XO 7 X1 8 NC */ /* data length read from SRAM */ #define DATA_MAX 131072 /* data width from ADC */ #define DATA_WIDTH 18 /* margin required for time align to prevent array over-run */ #define MARGIN 36 main() **{** FILE *stream; long i, j, xc, xb, xa; long bit_value, k, m, n; extern long testbit(); extern long setbit(); int *a, *b, *c; long *xsum, *ysum, *zsum; a = calloc(DATA_MAX, sizeof(int)); b = calloc(DATA_MAX, sizeof(int)); c = calloc(DATA_MAX, sizeof(int)); xsum = calloc(DATA_MAX, sizeof(long)); ysum = calloc(DATA_MAX, sizeof(long)); zsum = calloc(DATA_MAX, sizeof(long)); /* set up PXB-721 mode */ outp(PORT_D3, MODE);

/* acquire data */

```
/* clear MAR */
outp(PORT_A3, 0x10);
/* reset */
outp(PORT_A3, RESET);
/* set LOADMEM high */
outp(PORT_A3, 0x01);
/* acquire data with LOADMEM and REQCOUNTMAR high */
outp(PORT_A3, 0x09);
/* wait a while */
for(i = 0; i \le 500000; i++)
/* read data from SRAM */
/* clear MAR */
outp(PORT_A3, 0x10);
for(j = 0; j \le DATA_MAX; j++)
{
/* read out DATA_C from current MAR address */
outp(PORT_A3, DATA_C);
/* get DATA_C */
c[j] = inp(PORT_B3);
/* read out DATA_B from current MAR address */
outp(PORT_A3, DATA_B);
/* get DATA_B */
b[j] = inp(PORT_B3);
/* read out DATA_A from current MAR address */
outp(PORT_A3, DATA_A);
/* get DATA_A */
a[j] = inp(PORT_B3);
```

D.1. ACQ.C 207

```
/* reset */
outp(PORT_A3, RESET);
/* increment MAR */
outp(PORT_A3, 0x08);
}
/* reset */
outp(PORT_A3, RESET);
/* construct composite data from c[j], b[j], a[j] */
for(j = 0; j \le DATA_MAX; j++)
{
xc = c[j];
xc = xc << 16;
xb = b[j];
xb = xb << 8;
xa = a[j];
xsum[j] = xc + xb + xa;
xsum[j] = xsum[j] >> 6;
ysum[j] = (~xsum[j]) & 262143;
}
/* time align data */
for(j = 0; j \le DATA_MAX - MARGIN; j++)
{
k = 0;
n = 0;
zsum[j] = 0;
for(m = DATA_WIDTH - 1; m \ge 0; m--)
    bit_value = testbit(ysum[j + k], m);
    zsum[j] = setbit(zsum[j], m, bit_value);
    if (n == 0)
n = 1;
    else
{
n = 0;
k = k + 1;
}
```

```
}
/* write data out to file */
stream = fopen("DATA.ADC", "w+");
for(j = 0; j \le DATA_MAX - MARGIN; j++)
fprintf(stream, "%ld\n", zsum[j]);
/* close file */
fclose(stream);
return 0;
}
/*
   Function: setbit(word, position, bit_value)
   This function returns a long that is the specified word with the
   specified bit position set to bit_value.
*/
long setbit(word, position, bit_value)
long word, position, bit_value;
long mask0, mask1, output, temp;
temp = 1;
mask0 = temp << position;</pre>
mask1 = ^mask0;
if (bit_value == 0)
  output = mask1 & word;
  output = mask0 | word;
return output;
}
/*
   Function: testbit(word, position)
```

```
This function returns a long that is set according to the value
   of the bit at the specified position in the specified word. If
   the bit is set to 1, then 1 is returned, else 0 is returned.
*/
long testbit(word, position)
long word, position;
ſ
long temp, output;
temp = 1;
output = temp & (word >> position);
return output;
}
D.2
       cal.c
/* Program for Pipeline ADC Calibration System
                                                                */
/* Andrew Karanicolas
                                                                */
/* Microsystems Technology Laboratory
                                                                */
/* 09-10-92
                                                                */
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>
#include <float.h>
#include <math.h>
/* base address for PXB-721 is 0x300
                                                                */
/* 8255A_3 is presently used
                                                                */
/* PORT_D3 is control register
                                                                */
#define PORT_A3 0x308
#define PORT_B3 0x309
#define PORT_C3 0x30A
#define PORT_D3 0x30B
/* mode 0x82 configures PORT_A3 as output, PORT_B3 as input
                                                                */
/* and PORT_C3 as output
                                                                */
#define MODE 0x82
/* set all instruction bits to 0 for full reset
                                                                */
#define RESET 0x00
```

```
/* instructions to read SRAM DATA_C, DATA_B, DATA_A
                                                               */
#define DATA_C 0x42
#define DATA_B 0x22
#define DATA_A 0x02
/* data acquisition system instruction format
1
      LOADMEM
2
      REQREAD
3
      REQURITE
      REQCOUNTMAR
5
      REQCLRMAR
6
      XO
7
      X 1
8
      NC
*/
/* calibration system instruction format
1
      CALIBRATE
2
      CO
3
      C1
4
      C2
5
      СЗ
6
      C4
7
      S0
      S1
*/
/* program reference instructions
   S1S0 = 11 VP = VR1 - VR2 PHII = 1
   S1S0 = 10
               VP = AGND
                                 PHII = 1
   S1S0 = 00
               VP = AGND
                                 PHII = 0
   S1S0 = 01 VP = VR2 - VR1 PHII = 0
*/
#define S3 0xC0
#define S2 0x80
#define S1 0x00
#define SO 0x40
/* data length read from SRAM */
/*
#define DATA_MAX 2048
```

```
*/
/* data width from ADC */
#define DATA_WIDTH 18
/* margin required for time align to prevent array over-run */
#define MARGIN 36
main()
ſ
FILE *stream:
double sum, num_points, temp;
int select_index, select_stage;
int cal_index, cal_stage, calibrate;
char cal_instruction;
long DATA_MAX;
long i, j, xc, xb, xa;
long bit_value, k, m, n;
long position;
long dim_i, dim_j, dim_k;
extern long testbit();
extern long setbit();
extern long extract();
int *a, *b, *c;
long *xsum, *ysum, *zsum;
long *e;
double **s;
/* get DATA_MAX */
printf("enter DATA_MAX: \n");
scanf("%ld", &DATA_MAX);
/* allocate memory */
a = calloc(DATA_MAX, sizeof(int));
b = calloc(DATA_MAX, sizeof(int));
c = calloc(DATA_MAX, sizeof(int));
xsum = calloc(DATA_MAX, sizeof(long));
ysum = calloc(DATA_MAX, sizeof(long));
zsum = calloc(DATA_MAX, sizeof(long));
e = calloc(DATA_MAX, sizeof(long));
/* allocate and initialize s[i][j] */
```

```
dim_i = DATA_WIDTH;
dim_j = 4;
s = calloc(dim_i, sizeof(double *));
for(i = 0; i <= dim_i - 1; i++)
   s[i] = calloc(dim_j, sizeof(double));
   }
/* set up PXB-721 mode */
outp(PORT_D3, MODE);
/* acquire calibration data from desired stage */
for(select_stage = 1; select_stage <= DATA_WIDTH; select_stage++)</pre>
   for(select_index = 0; select_index <= 3; select_index++)</pre>
      {
/* subtract 1 as hardware selection begins at 0 */
cal_stage = select_stage - 1;
/* shift cal_stage left so that cal_stage is aligned for instruction */
cal_stage = cal_stage << 1;</pre>
/* set cal_index depending on selected calibration data index */
switch( select_index )
 ſ
 case 0: cal_index = S0;
    break:
 case 1: cal_index = S1;
   break;
 case 2: cal_index = S2;
    break;
 case 3: cal_index = S3;
   break;
 default:
   ſ
     printf("ERROR ERROR ERROR \n");
     printf("int select_index out of range. \n");
     return 1;
   }
 }
```

```
/* build calibration instruction */
calibrate = 1;
cal_instruction = cal_index + cal_stage + calibrate;
/* clear PORT_C3 */
outp(PORT_C3, RESET);
/* send calibration instruction to PORT_C3 */
outp(PORT_C3, cal_instruction);
/* wait a while for VP to settle */
for(i = 0; i \le 100000; i++)
/* acquire data */
/* clear MAR */
outp(PORT_A3, 0x10);
/* reset */
outp(PORT_A3, RESET);
/* set LOADMEM high */
outp(PORT_A3, 0x01);
/* acquire data with LOADMEM and REQCOUNTMAR high */
outp(PORT_A3, 0x09);
/* wait a while */
for(i = 0; i \le 100000; i++)
/* read data from SRAM */
/* clear MAR */
outp(PORT_A3, 0x10);
for(j = 0; j \le DATA_MAX; j++)
/* read out DATA_C from current MAR address */
outp(PORT_A3, DATA_C);
```

```
/* get DATA_C */
c[j] = inp(PORT_B3);
/* read out DATA_B from current MAR address */
outp(PORT_A3, DATA_B);
/* get DATA_B */
b[j] = inp(PORT_B3);
/* read out DATA_A from current MAR address */
outp(PORT_A3, DATA_A);
/* get DATA_A */
a[j] = inp(PORT_B3);
/* reset */
outp(PORT_A3, RESET);
/* increment MAR */
outp(PORT_A3, 0x08);
}
/* reset */
outp(PORT_A3, RESET);
/* construct composite data from c[j], b[j], a[j] */
for(j = 0; j \le DATA_MAX; j++)
{
xc = c[j];
xc = xc \ll 16;
xb = b[j];
xb = xb \ll 8;
xa = a[j];
xsum[j] = xc + xb + xa;
xsum[j] = xsum[j] >> 6;
ysum[j] = (xsum[j]) & 262143;
/* time align data */
for(j = 0; j \leftarrow DATA_MAX - MARGIN; j++)
```

```
{
k = 0:
n = 0;
zsum[j] = 0;
for(m = DATA_WIDTH - 1; m >= 0; m--)
    {
    bit_value = testbit(ysum[j + k], m);
    zsum[j] = setbit(zsum[j], m, bit_value);
    if (n == 0)
n = 1;
    else
{
n = 0;
k = k + 1;
    }
}
/* extract output bits related to calibration of stage */
for(j = 0; j \leftarrow DATA_MAX - MARGIN; j++)
    position = DATA_WIDTH - select_stage;
    e[j] = extract(zsum[j], position);
    }
/* compute average */
temp = 0.0;
sum = 0.0;
num_points = DATA_MAX - MARGIN;
for(j = 0; j \le DATA_MAX - MARGIN; j++)
    {
    temp = e[j];
    sum = sum + temp;
sum = sum / num_points;
/* copy average to s[select_stage][select_index] */
s[select_stage - 1][select_index] = sum;
printf("s[%ld][%ld] = %.10lf \n", select_stage, select_index,
       s[select_stage - 1][select_index]);
```

```
}
/* write data out to file */
stream = fopen("CALDATA.ADC", "w+");
for(select_stage = 1; select_stage <= DATA_WIDTH; select_stage++)</pre>
     for(select_index = 0; select_index <= 3; select_index++)</pre>
     fprintf(stream, "%.10lf \n", s[select_stage - 1][select_index]);
fclose(stream);
/* reset */
outp(PORT_C3, RESET);
/* free allocated memory */
free(a);
free(b);
free(c);
free(xsum):
free(ysum);
free(zsum);
free(e);
free(s);
*/
return 0;
/*
   Function: setbit(word, position, bit_value)
   This function returns a long that is the specified word with the
   specified bit position set to bit_value.
*/
long setbit(word, position, bit_value)
long word, position, bit_value;
long mask0, mask1, output, temp;
temp = 1;
```

```
mask0 = temp << position;</pre>
mask1 = mask0;
if (bit_value == 0)
  output = mask1 & word;
else
  output = mask0 | word;
return output;
}
/*
   Function: testbit(word, position)
   This function returns a long that is set according to the value
   of the bit at the specified position in the specified word. If
   the bit is set to 1, then 1 is returned, else 0 is returned.
*/
long testbit(word, position)
long word, position;
ſ
long temp, output;
temp = 1;
output = temp & (word >> position);
return output;
}
   Function: extract(word, position)
   This function extracts a sub-word of word from bit 0 to bit position.
*/
long extract(word, position)
long word, position;
long mask0, mask1, temp, output, i;
mask1 = 0;
for(i = 0; i <= position; i++)</pre>
  {
```

```
temp = 1;
mask0 = temp << i;
mask1 = mask1 + mask0;
}
output = word & mask1;
return output;
}</pre>
```

Bibliography

- [1] R. Gregorian, G. C. Temes, Analog MOS Integrated Circuits for Signal Processing. New York: Wiley, 1986.
- [2] J. L. McCreary and P. R. Gray, "All-MOS charge redistribution analog-to-digital conversion techniques-part 1," *IEEE Journal of Solid-State Circuits*, vol. SC-10, no. 6, pp. 371-379, Dec. 1975.
- [3] R. E. Suarez and P. R. Gray and D. A. Hodges, "All-MOS charge redistribution analog-to-digital conversion techniques-part 2," *IEEE Journal of Solid-State Circuits*, vol. SC-10, no. 6, pp. 379-385, Dec. 1975.
- [4] D. A. Hodges, P. R. Gray, R. W. Brodersen, "Potential of MOS for Analog Integrated Circuits," *IEEE Journal of Solid-State Circuits*, vol. SC-13, no. 3, pp. 285-293, June 1978.
- [5] T. L. Tewksbury III, Relaxation Effects in MOS Devices due to Tunnel Exchange with Near-Interface Oxide Traps. PhD thesis, Massachusetts Institute of Technology, Dept. of EECS, June 1992.
- [6] T. L. Tewksbury, H.-Seung Lee, G. A. Miller, "The Effects of Oxide Traps on the Large-Signal Transient Response of Analog MOS Circuits," *IEEE Journal of Solid-State Circuits*, vol. 24, no. 2, pp. 542-544, Apr. 1989.
- [7] B. J. Gross, 1/f Noise in MOSFETS with Ultrathin Gate Dielectrics. PhD thesis, Massachusetts Institute of Technology, Dept. of EECS, June 1992.
- [8] R. E. Flory, "Image Acquisition Technology," Proc. IEEE, vol. 73, no. 4, pp. 613-637, Apr. 1985.
- [9] H. Toyoda, H. Harada, M. Onishi, S. Nishikawa, Y. Kitamura, "New Automatic Focusing System for Video Cameras," in *IEEE International Conference on Consumer Electronics*, 1986.
- [10] T. Maruyama, K. Sano, H. Inage, M. Masuda, Y. Sugi, Y. Arai, "A Zoom Lens with New Infrared Autofocus Mechanism," in *IEEE International Conference on Consumer Electronics*, 1986.

[11] A. Franken, N. V. Rao, "Television Camera Tubes and Solid-State Sensors for Broad-cast Applications," Society of Motion Picture and Television Engineers Journal, pp. 799-804, Aug. 1986.

- [12] L. Thorpe, E. Tamura, "A New 510-Element CCD Camcorder for ENG," Society of Motion Picture and Television Engineers Journal, pp. 518-526, June 1987.
- [13] P. N. T. Wells, ed., Scientific Basis of Medical Imaging. Edinburgh, Scotland: Churchill Livingstone, 1982.
- [14] J. J. Erickson, F. D. Rollo, ed., Digital Nuclear Medicine. Philadelphia, PA: J. B. Lippincott Company, 1983.
- [15] B. M. Gordon, "Linear Electronic Analog/Digital Conversion Architectures, Their Origins, Parameters, Limitations, and Applications," IEEE Trans. on Circuits and Systems, vol. CAS-25, no. 7, pp. 391-418, July 1978.
- [16] J. van Valburg, R. J. van de Plassche, "An 8-b 650-MHz Folding ADC," IEEE Journal of Solid-State Circuits, vol. 27, no. 12, pp. 1662-1666, Dec. 1992.
- [17] C. Schiller, "A 4-GHz 8-b ADC System," *IEEE Journal of Solid-State Circuits*, vol. 26, no. 12, pp. 1781-1789, Dec. 1991.
- [18] R. J. van de Plassche, P. Baltus, "An 8-bit 100-MHz Full-Nyquist Analog-to-Digital Converter," *IEEE Journal of Solid-State Circuits*, vol. 23, no. 6, pp. 1334-1344, Dec. 1988.
- [19] R. E. J. van de Grift, et al., "An 8-bit Video ADC Incorporating Folding and Interpolation Techniques," *IEEE Journal of Solid-State Circuits*, vol. SC-22, no. 6, pp. 944-953, Dec. 1987.
- [20] R. E. J. van de Grift, R. J. van de Plassche, "An Monolithic 8-bit 100MHz Full-Nyquist Analog-to-Digital Converter," *IEEE Journal of Solid-State Circuits*, vol. SC-19, no. 3, pp. 374-378, June 1984.
- [21] R. J. van de Plassche, R. E. J. van de Grift, "High-Speed 7-bit A/D Converter," *IEEE Journal of Solid-State Circuits*, vol. SC-14, no. 6, pp. 938-943, Dec. 1979.
- [22] U. Fiedler, D. Seitzer, "A High-Speed 8-bit A/D Converter Based on a Gray-Code Multiple Folding Circuit," *IEEE Journal of Solid-State Circuits*, vol. SC-14, no. 3, pp. 547-551, June 1979.
- [23] B. Razavi, B. A. Wooley, "A 12-b 5-MSample/s Two-Step CMOS A/D Converter," *IEEE Journal of Solid-State Circuits*, vol. 27, no. 12, pp. 1667-1678, Dec. 1992.
- [24] R. Petschacher, et al., "A 10-b 75-MSPS Subranging A/D Converter with Integrated Sample and Hold," *IEEE Journal of Solid-State Circuits*, vol. 25, no. 6, pp. 1339-1346, Dec. 1990.

[25] M. K. Mayes, S. W. Chin, "A Multistep A/D Converter Family with Efficient Architecture," IEEE Journal of Solid-State Circuits, vol. 24, no. 6, pp. 1492-1497, Dec. 1989.

- [26] M. P. V. Kolluri, "A 12-bit 500-ns Subranging ADC," IEEE Journal of Solid-State Circuits, vol. 24, no. 6, pp. 1498-1506, Dec. 1989.
- [27] S. H. Lewis, P. R. Cray, "A Pipelined 5-Msample/s 9-bit Analog-to-Digital Converter," *IEEE Journal of Solid-State Circuits*, vol. SC-22, no. 6, pp. 954-961, Dec. 1987.
- [28] S. H. Lewis, "Optimizing the Stage Resolution in Pipelined, Multistage, Analog-to-Digital Converters for Video-Rate Applications," IEEE Trans. on Circuits and Systems II: Analog and Digital Signal Processing, vol. 39, no. 3, pp. 516-523, Aug. 1992.
- [29] P. Real, D. H. Robertson, C. W. Mangelsdorf, T. L. Tewksbury, "A Wide-Band 10-b 20-Ms/s Pipelined ADC Using Current-Mode Signals," *IEEE Journal of Solid-State Circuits*, vol. 26, no. 8, pp. 1103-1109, Aug. 1991.
- [30] B.-S. Song, M. F. Tompset, K. R. Lakshmikumar, "A 12-bit 1-Msample/s Capacitor Error-Averaging Pipelined A/D Converter," *IEEE Journal of Solid-State Circuits*, vol. SC-23, no. 6, pp. 1324-1333, Dec. 1988.
- [31] A. N. Karanicolas, H.-Seung Lee, K. L. Bacrania, "A 15b 1Ms/s Digitally Self-Calibrated Pipeline ADC," in *IEEE International Solid-State Circuits Conference*, (San Francisco, CA), 1993.
- [32] A. N. Karanicolas, H.-Seung Lee, K. L. Bacrania, "A 15b 1Ms/s Digitally Self-Calibrated Pipeline ADC," *IEEE Journal of Solid-State Circuits*, vol. 28, no. 12, pp. n1-n2, Dec. 1993.
- [33] H.-Seung Lee, "A 12 bit 600 kS/s Digitally Self-Calibrated Pipelined Algorithmic ADC," in *IEEE Symposium on VLSI Circuits*, (Kyoto, Japan), 1993.
- [34] H.-Seung Lee, A. N. Karanicolas, "Accuracy Limitations and Calibration Techniques for Pipeline A/D Converters," in *IEEE International Conference on VLSI and CAD*, (Taejon, Korea), 1993.
- [35] S. D. E. Pomeroy, High Resolution, High Speed Analog-To-Digital Conversion Techniques. PhD thesis, Massachusetts Institute of Technology, Dept. of EECS, Aug. 1989.
- [36] S. Sutarja, P. R. Gray, "A Pipelined 13-bit, 250-ks/s, 5-V Analog-to-Digital Converter," *IEEE Journal of Solid-State Circuits*, vol. 23, no. 6, pp. 1316-1323, Dec. 1988.
- [37] Y.-M. Lin, B. Kim, P. R. Gray, "A 13b 2.5MHz Self-Calibrated Pipelined A/D Converter in 3μm CMOS," IEEE Journal of Solid-State Circuits, vol. SC-26, pp. 628-636, Apr. 1991.

[38] A. Fotouhi, D. A. Hodges, "High-Resolution A/D Conversion in MOS/LSI," IEEE Journal of Solid-State Circuits, vol. 14, no. 6, pp. 920-926, Dec. 1979.

- [39] K. Bacrania, "A 12-bit Successive-Approximation-Type ADC with Digital Error Correction," *IEEE Journal of Solid-State Circuits*, vol. SC-21, no. 6, pp. 1016-1025, Dec. 1986.
- [40] Z. G. Boyacigiller, B. Weir, P. D. Bradshaw, "An error-correcting 14b/20μs CMOS A/D converter," in IEEE International Solid-State Circuits Conference, 1981.
- [41] Z. G. Boyacigiller, et al., Aug. 1984. High Accuracy Digital-to-Analog Converter. United States Patent #4,465,996.
- [42] H.-Seung Lee, D. A. Hodges and P. R. Gray, "A self-calibrating 15-bit CMOS A/D converter," *IEEE Journal of Solid-State Circuits*, vol. SC-19, no. 6, pp. 813-819, Dec. 1984.
- [43] H.-Seung Lee, D. A. Hodges, "Accuracy Considerations in Self-Calibrating A/D Converters," IEEE Trans. on Circuits and Systems, vol. CAS-32, no. 6, pp. 590-597, June 1985.
- [44] H.-Seung Lee, D. A. Hodges, "Self-Calibration Technique for A/D Converters," *IEEE Trans. on Circuits and Systems*, vol. CAS-30, no. 3, pp. 188-190, Mar. 1983.
- [45] J. C. Candy, G. C. Temes, "Oversampling Methods for A/D and D/A Conversion," in Oversampling Delta-Sigma Data Converters (J. C. Candy, G. C. Temes, ed.), pp. 1-25, New York: IEEE Press, 1992.
- [46] J. W. Fattaruso, et al., "Self-Calibration Techniques for a Second-Order Multibit Sigma-Delta Modulator," in *IEEE International Solid-State Circuits Conference*, 1993.
- [47] B. Ginetti, et al., "A CMOS 13-b Cyclic RSD A/D Converter," IEEE Journal of Solid-State Circuits, vol. 27, no. 7, pp. 957-965, July 1992.
- [48] S.-Hoon Lee, Bang-Sup Song, "Digital-Domain Calibration of Multistep Analog-to-Digital Converters," *IEEE Journal of Solid-State Circuits*, vol. 27, no. 12, pp. 1679– 1688, Dec. 1992.
- [49] S.-Hoon Lee, Bang-Sup Song, "A Code-Error Calibrated Two-Step A/D Converter," in *IEEE International Solid-State Circuits Conference*, 1992.
- [50] Y.-M Lin, B. Kim, P. R. Gray, "A 13 bit 2.5MHz Self-Calibrated Pipelined A/D Converter in 3μm CMOS," in *IEEE Symposium on VLSI Circuits*, 1990.
- [51] Y.-M. Lin, Performance Limitations on High-Resolution Video-Rate Analog-Digital Interfaces. PhD thesis, University of California, Berkeley, Electronics Research Laboratory, College of Engineering, June 1990.

[52] J.-P. Hwang, et al., Jan. 1990. Self-Calibrating Pipelined Subranging Analog-to-Digital Converter. *United States Patent* #4,894,656.

- [53] J. J. Tiemann, et al., Feb. 1990. Architecture for High Sampling Rate, High Resolution Analog-to-Digital Converter System. United States Patent #4,903,026.
- [54] D. B. Ribner, Sept. 1991. Digital Error Correction System for Subranging Analog-to-Digital Converters. United States Patent #5,047,772.
- [55] R. H. McCharles, Charge Circuits for Analog LSI. PhD thesis, University of California, Berkeley, Electronics Research Laboratory, College of Engineering, Mar. 1981.
- [56] Jae S. Lim, Two-Dimensional Signal and Image Processing. Englewood Cliffs, N.J.: Prentice Hall, 1990.
- [57] A. Gersho, "Principles of Quantization," IEEE Trans. on Circuits and Systems, vol. CAS-25, no. 7, pp. 427-436, July 1978.
- [58] Hae-Seung Lee; Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology. Personal Communication.
- [59] P. W. Li, M. J. Chin, P. R. Gray, and R. Castello, "A Ratio Independent Algorithmic Analog-to-Digital Conversion Technique," *IEEE Journal of Solid-State Circuits*, vol. SC-19, no. 6, pp. 1138-1143, Dec. 1984.
- [60] G. Wegmann, E. A. Vittoz, F. Rahali, "Charge Injection in Analog MOS Switches," IEEE Journal of Solid-State Circuits, vol. SC-22, no. 6, pp. 1091-1097, Dec. 1987.
- [61] W. B. Wilson, H. Z. Massoud, E. J. Swanson, R. T. George, Jr., R. B. Fair, "Measurement and Modeling of Charge Feedthrough in n-channel MOS Analog Switches," IEEE Journal of Solid-State Circuits, vol. SC-20, no. 6, pp. 1206-1213, Dec. 1985.
- [62] D. MacQuigg, "Residual Charge on a Switched Capacitor," IEEE Journal of Solid-State Circuits, vol. SC-18, no. 6, pp. 811-813, Dec. 1983.
- [63] C.-A. Gobet, A. Knob, "Noise Analysis of Switched Capacitor Networks," *IEEE Trans. on Circuits and Systems*, vol. CAS-30, no. 1, pp. 37-43, Jan. 1983.
- [64] C. A. Gobet, "Spectral Distribution of a Sampled 1st-Order Lowpass Filtered White Noise," IEE Electronics Letters, vol. 17, no. 19, pp. 720-721, Sept. 1981.
- [65] Cl.-A. Gobet, A. Knob, "Noise Generated in Switched Capacitor Networks," IEE Electronics Letters, vol. 16, no. 19, pp. 734-735, Sept. 1980.
- [66] J. K. Roberge, Operational Amplifiers: Theory and Practice. New York: Wiley, 1975.
- [67] B. J. McCarroll, C. G. Sodini, H.-Seung Lee, "A High-Speed CMOS Comparator for use in an ADC," *IEEE Journal of Solid-State Circuits*, vol. SC-23, no. 1, pp. 159-165, Feb. 1988.

[68] A. B. Grebene, Bipolar and MOS Analog Integrated Circuit Design. New York: Wiley, 1984.

- [69] A. N. Karanicolas, K. K. O, J. Y. Wang, H.-Seung Lee, R. L. Reif, "A High Frequency Fully Differential BiCMOS Operational Amplifier," in *IEEE Custom Integrated Circuits Conference*, (Boston, MA), 1990.
- [70] A. N. Karanicolas, K. K. O, J. Y. Wang, H.-Seung Lee, R. L. Reif, "A High Frequency Fully Differential BiCMOS Operational Amplifier," *IEEE Journal of Solid-State Cir*cuits, vol. 26, no. 3, pp. 203-208, Mar. 1991.
- [71] J. E. Solomon, "The Monolithic Op Amp: A Tutorial Study," *IEEE Journal of Solid-State Circuits*, vol. SC-9, no. 6, pp. 314-332, Dec. 1974.
- [72] P. R. Gray, R. G. Meyer, Analysis and Design of Analog Integrated Circuits. New York: Wiley, 2nd ed., 1984.
- [73] Y. P. Tsividis, Operation and Modeling of the MOS Transistor. New York: McGraw Hill, 1987.
- [74] R. Castello, P. R. Gray, "A High Performance Micropower Switched-Capacitor Filter," IEEE Journal of Solid-State Circuits, vol. SC-20, no. 6, pp. 1122-1132, Dec. 1985.
- [75] Mick Mueck; Analog Devices, Inc. Personal Communication.
- [76] Audio Precision, Inc. Staff, ed., Audio Precision System One Manual. Beaverton, Oregon: Audio Precision, Inc., 1992.
- [77] Analog Devices Staff, ed., Data Converter Reference Manual Volume I. Norwood, MA: Analog Devices, Inc., 1992.
- [78] Crystal Semiconductor Staff, ed., Analog/Digital Conversion IC's. Austin, TX: Crystal Semiconductor Corporation, 1992.
- [79] F. J. Harris, "On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform," *Proc. IEEE*, vol. 66, no. 1, pp. 51-84, Jan. 1978.
- [80] J. Doernberg, H.-Seung Lee, D. A. Hodges, "Full-Speed Testing of A/D Converters," IEEE Journal of Solid-State Circuits, vol. SC-19, no. 6, pp. 820-827, Dec. 1984.
- [81] A. Papoulis, Probability, Random Variables, and Stochastic Processes. New York: McGraw Hill, 1965.
- [82] A. V. Oppenheim, R. W. Schafer, Discrete-Time Signal Processing. Englewood Cliffs, NJ: Prentice Hall, 1989.
- [83] W. McC. Siebert, Circuits, Signals, and Systems. Cambridge, MA: MIT Press, 1986.