AN ABSTRACT OF THE DISSERTATION OF

Allen Waters for the degree of Doctor of Philosophy in

Electrical and Computer Engineering presented on March 5, 2015.

Title:

Automated Verilog-to-Layout Synthesis of ADCs Using Custom Analog Cells.

Abstract approved: ______________________________

Un-Ku Moon

A procedure for automating the design and layout of analog-to-digital converters (ADCs) is presented. This procedure makes use of the existing synthesis and place-and-route tools that are common in digital circuit design. A method for adding rudimentary analog cells to the standard library is described, allowing the designer to synthesize mixed-signal designs from Verilog code. By using cells that are simple and highly scalable, the same Verilog code may be used to implement the design in any number of process nodes, for rapid portability and scalability. Two different ADC architectures are implemented as proofs of concept: first, a third-order MASH ADC is fabricated in 130nm and 65nm CMOS, taking advantage of the structure's tolerance to the mismatch introduced by the automated place-and-routing. Second, a Nyquist-rate pipeline ADC using the highly-scalable ring amplifier is fabricated in 65nm CMOS. The measurement results from these chips show that synthesized ADCs can achieve moderate performance with drastically reduced design time compared to manual layout.

Automated Verilog-to-Layout Synthesis of ADCs Using Custom Analog Cells

by

Allen Waters

A DISSERTATION

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

Presented March 5, 2015
Commencement June 2015

ACKNOWLEDGMENTS

I would like to first express my appreciation to Dr. Un-Ku Moon for his guidance as my research advisor. When I first joined his group, I did not fully understand how lucky I was to be working under such an experienced and accomplished professor. I'm very grateful for the kindness he has shown me the last five years. I'll do my best to make good use of his technical knowledge and insight in my future career.

Secondly, I give thanks to the other members of my graduate committee: Dr. Karti Mayaram, Dr. Gabor Temes, and Dr. Huaping Liu. I appreciate their time and the technical feedback they have given me along the way. Furthermore, I would like to thank them for the valuable knowledge they provided through their teaching and coursework. Many thanks to Dr. Karen Shell from the College of Earth, Ocean, and Atmospheric Sciences for serving as my Graduate Council Representative. I appreciate the time and energy she has dedicated to my exams and program meetings over the past years.

I would like to express my gratitude to the senior group members who assisted me during my first years of graduate study: Skyler Weaver, Omid Rajaee, Tawfik Musah, Nima Maghari, Ho-Young Lee, Ben Hershberg, Jon Guerber, Taehwan Oh, Manideep Gande, Hari Prasath Venkatram, and Yue Hu. It was a real privilege to learn from these guys, and my only regret is that I didn't ask more questions from the beginning. Matt Brown has also provided terrific support as the AMS lab manager.

I offer my gratitude and best wishes to the current (including recently graduated) group members, who have been great friends: Jason Muhlestein, Jerry Leung, Spencer Leuenberger, Hyuk Sun, Yang Xu, Brandilyn Coker, Praveen Ku-mar Venkatachala and Farshad Farahbahkshian. I will miss our time together in Dr. Moon's research group, and wish them the best in the remainder of their grad school years.

There are many other students, current and former, in the KEC 4130 grad office who I want to thank... I won't list all their names here. Being a grad student feels like a pretty raw deal sometimes, when you're putting in crazy hours working on a tapeout or preparing for an exam. Having other people in the office working the same crazy hours makes it a lot more bearable, and these guys have been a great support network.

Without going into too much detail, I'd like to acknowledge some other people and organizations that helped me get through my degree: the ARCS Foundation (Portland Chapter), the 26th Street Superette, Galen Wigg, Rex Goliath, the Coalition of Graduate Employees, Suds and Suds.

Finally, I want to recognize and thank the members of my family who made this possible. My parents Tom and Bonnie have been supportive and encouraging throughout all of my schooling. I'm thankful that I've finally reached the end and want to thank them for all the contributions they've made along the way, both large and small. I'd like to thank my brother Ben for being my closest friend the entire time, and my older brother Patrick for being my greatest role model.

There are too many other friends, family members and colleagues to list here. For all those people, I appreciate the individual contributions you've made that helped me along my way.

TABLE OF CONTENTS

Page

1 INTRODUCTION 1

1.1 Motivation 1

1.2 Thesis Structure 3

2 PRIOR WORK 6

2.1 Prior Work in Highly Digital ADCs 6

2.1.1 Highly Digital Flash 6

2.1.2 Basic Time-to-Digital Converters 8

2.1.3 Highly-Digital $\Delta\Sigma$ Modulator 12

2.2 Prior Work in Synthesized Analog Blocks 13

2.2.1 Synthesized Stochastic Flash ADC 14

2.2.2 Analog Design Automation 15

2.3 Summary 17

3 PROPOSED ANALOG SYNTHESIS TOOLCHAIN 18

3.1 Existing Digital Synthesis Toolchain 19

3.2 Library Information for Digital Standard Cells 22

3.2.1 Graphic Database System (GDS) Information 22

3.2.2 Library Exchange Format (LEF) Information 23

3.2.3 Liberty (LIB) Timing File Information 27

3.3 Creating Custom Analog Cell Information 27

3.3.1 Minimalist Set of Analog Cells 31

3.3.2 Creating .GDS File for Custom Cell 32

3.3.3 Creating .LEF File for Custom Cell 35

3.3.4 Creating .LIB File For Custom Cell 35

3.3.5 Controlling Floorplanning for Analog Cells 39

TABLE OF CONTENTS (Continued)

Page

3.4 Final Toolchain 40

3.4.1 New Automated Analog Design Toolchain 43
3.4.2 Capabilities and Limitations 43

3.5 Summary 48

4 A 1-1-1 MASH ADC SYNTHESIZED FROM VERILOG CODE 49

4.1 Background on ΔΣ Modulation 49

4.1.1 Nyquist-Rate SQNR Limit 49
4.1.2 Improving Resolution with Oversampling 52
4.1.3 First-Order Noise Shaping 54
4.1.4 Low Distortion Topology 57
4.1.5 Higher-Order Noise Shaping 61
4.1.6 Finite Integrator Gain and Gain Compression 66

4.2 Proposed System Architecture 69

4.2.1 Top-Level Design 69
4.2.2 Noise Sources 73
4.2.3 Capacitor Sizing 75

4.3 Minimalist Sub-Block Design 77

4.3.1 Operational Amplifier 77
4.3.2 Three-Level Quantizer Design 79
4.3.3 Feedback DAC Design 81
4.3.4 Final Simulation Results 81

4.4 Experimental Results 83

4.4.1 MASH ADC in 130nm CMOS 83
4.4.2 MASH ADC in 65nm CMOS 87
4.4.3 Comparison to State-of-the-Art 93

4.5 Summary 93

TABLE OF CONTENTS (Continued)

Page

5 RING AMPLIFIER-BASED PIPELINE ADC SYNTHESIZED FROM VERILOG CODE 95

5.1 Background on Ring Amplifier 95

5.2 Proposed System Architecture 98

5.2.1 Top-Level System Design 98
5.2.2 Noise Sources 101
5.2.3 Capacitor Sizing 102

5.3 Minimalist Sub-Block Design 103

5.3.1 Ring Amplifier Output Stage 103
5.3.2 Sub-ADC Reference Ladder 104
5.3.3 Final Simulation Results 104

5.4 Experimental Results 105

5.4.1 Pipeline ADC in 65nm CMOS 107
5.4.2 Comparison to State-of-the-Art 107

5.5 Summary 109

6 CONCLUSION 110

6.1 Summary 110

6.2 Future Work 111

6.2.1 Automating Architecture Design 111
6.2.2 Analog Cells with Fake Logic Functions 113
6.2.3 Analog Cells with Fake Timing Information 115

BIBLIOGRAPHY 117

APPENDICES 122

A Finite Gain in Switched-Capacitor Integrators 123

TABLE OF CONTENTS (Continued)

Page

B List of Acronyms 129

LIST OF FIGURES

Figure		Page
1.1	Scaling trend in CMOS process sizes.	2
1.2	Block diagram of ATMega128 microcontroller, highlighting ADC.	4
1.3	Survey of modern ADC performance, compared to ATMega128 example.	5
2.1	Modifying the (a) standard flash ADC to create a (b) redundant flash.	7
2.2	Highly digital dynamic comparator, with k input device stack of variable widths.	8
2.3	Toolchain used for digital circuit synthesis.	9
2.4	Effect of voltage noise with scaling supply voltage for (a) ADC and (b) TDC.	10
2.5	Examples of TDC architectures: (a) flash TDC, (b) Vernier TDC, (c) VCO-based TDC. Voltage-to-time operation displayed in gray.	11
2.6	TDC delay cells used in (a) multi-path GRO and (b) SRO.	12
2.7	Using a DSM loop to suppress VCO nonlinearity.	13
2.8	An analog comparator constructed from (a) two cross-coupled NAND3 logic gates, to achieve the (b) transistor-level schematic for a comparator.	14
2.9	Previous work in automating the (a) traditional analog design process, showing hightlighted blocks where automation has been achieved by (b) ASTRX, (c) KOAN, and (d) WREN.	16
3.1	Toolchain used for digital circuit synthesis.	20
3.2	Instantiations in Verilog can be (a) logical, (b) primitive gate-level, or (c) process-specific gate-level. All describe the same (d) schematic.	21
3.3	Example of how a standard cell might appear in (a) layout view, and (b) a hexdump of the binary GDS file describing it.	24
3.4	Example technology LEF layer definitions.	25

Figure Page

3.5 Example technology LEF contact definitions. 26

3.6 Example LEF macro statement. 28

3.7 Example LIB technology definitions. 29

3.8 Example LIB cell definition. 30

3.9 A minimal set of custom cells for analog design. 31

3.10 Example of multi-pitch custom cell layout, for a MIM capacitor. . 33

3.11 Streaming out a GDS file for a custom cell. 34

3.12 Example of describing the inverting integrator (a) schematic into (b) Verilog code using explicit instantiations of custom cells. 37

3.13 Using Design Compiler directives to explicitly instantiate analog custom cells without a logical description. 38

3.14 Example of controlling floorplanning with the "setObjFPlanBox" directive. The modules in (a) Verilog are floorplanned in (b) TCL to achieve a good starting point in the (c) layout. 41

3.15 Using PNR floorplanning, one level deeper. 42

3.16 Toolchain used for analog circuit synthesis. 44

3.17 A possible non-overlapping clock generator (a) schematic and (b) waveform. 47

4.1 Ideal ADC quantization error shown in (a) schematic and (b) transfer function. 50

4.2 Reducing quantization noise with oversampling. 52

4.3 A first-order $\Delta\Sigma$ ADC. 54

4.4 Power spectral density with first-order noise shaping. 56

4.5 A first-order $\Delta\Sigma$ ADC, using low-distortion architecture. 58

4.6 Switched capacitor implementation of low-distortion DSM topology. 59

4.7 DSM modulators with (a) second and (b) third-order noise shaping. 60

Figure Page

4.8 Theoretical SQNR limit of 1.5-bit modulators with various orders. 64

4.9 An n-stage MASH ADC. 64

4.10 Simulation results showing effect of finite gain and thermal noise. 67

4.11 Simulation results showing effect of gain compression in integrator. 68

4.12 Simulation results showing sensitivity of finite-gain calibration in 1-1-1 MASH, for a nominally 30dB amplifier. 69

4.13 Maximum SQNR for different MASH orders using 30dB opamps (perfectly calibrated). 71

4.14 Top-level architecture of 1-1-1 MASH ADC. 72

4.15 Switched-capacitor implementation of first-order DSM (shown single-ended for simplicity). 73

4.16 Location of noise sources (a) in the first-order DSM, and (b) an equivalent representation combining sources. 74

4.17 Minimalist single stage opamp design using (a) basic differential pair and (b) additional cross-couple active load. 78

4.18 DC gain of the simple opamp as a function of output swing, for (a) 130nm and (b) 65nm processes. 79

4.19 Harmonic distortion of the opamp transfer functions across output swing. 80

4.20 Opamp performance in 130nm process, showing (a) noise PSD and (b) integrated noise voltage. 80

4.21 Opamp performance in 65nm process, showing (a) noise PSD and (b) integrated noise voltage. 81

4.22 Switched-capacitor implementation of three-level quantizer, using a Strong-Arm latch as the quantizer. 82

4.23 Voltage-switched three-level DAC (shown single-ended for simplicity). 82

LIST OF FIGURES (Continued)

Figure Page

4.24 Final simulation results with transient noise, for (a) transistor-level and (b) post-layout extraction. . . . 84

4.25 Parasitic capacitance causing 2nd-order distortion, shown in (a) schematic and (b) the resulting FFT spectrum. . . . 85

4.26 Layout view of floorplanning for MASH ADC in 130nm process. . 86

4.27 Example output spectrum of 130nm MASH ADC. . . . 86

4.28 Performance of 130nm MASH ADC, sweeping across (a) input amplitude and (b) sample rate. Results use OSR = 40 in all cases. 88

4.29 Estimated power breakdown in 130nm MASH ADC. . . . 89

4.30 Die photo of synthesized MASH ADC in 130nm process. . . . 89

4.31 Example output spectrum of 65nm MASH ADC. . . . 90

4.32 Performance of 65nm MASH ADC, sweeping across (a) input amplitude and (b) sample rate. Results use OSR = 75 in all cases. . . 91

4.33 Estimated power breakdown in 65nm MASH ADC. . . . 92

4.34 Die photo of synthesized MASH ADC in 65nm process. . . . 93

4.35 Comparison of this work to ISSCC publications. . . . 94

5.1 Basic architecture of (a) stand-alone ring amplifier and (b) ring amplifier-based MDAC. . . . 96

5.2 Example ring amplifier output waveform showing the three phases of operation. . . . 97

5.3 (a) Top-level architecture of proposed pipeline ADC, and (b) schematic of MDAC stage (shown single-ended for simplicity). . . . 99

5.4 Pseudo-differential ring amplifier implementation, with added power-saving feature and switched-capacitor CMFB. . . . 100

5.5 Location of noise sources (a) in switched-capacitor MDAC stage, and (b) an equivalent block diagram combining sources for *i*-th MDAC stage. . . . 101

5.6 Output stage of ringamp, to be added as custom analog cell. . . . 104

LIST OF FIGURES (Continued)

Figure Page

5.7 Implementation of sub-ADC and resistive reference ladder (shown single-ended for simplicity). 105

5.8 Final simulation results with transient noise, for (a) transistor-level and (b) post-layout extraction. 106

5.9 Extracted simulation results across deadzone settings. 107

5.10 Die photo of synthesized pipeline ADC in 65nm process. 108

5.11 Comparison of this work to ISSCC publications. 109

6.1 Automating the creation of the Verilog code in future work. 111

6.2 Graphical representation of resolution/bandwidth look-up table used by automation toolbox. 112

A.1 Switched-capacitor integrator with a finite-gain amplifier. 124

A.2 Switched-capacitor integrator during (a) phase 1 and (b) phase 2. 125

A.3 Deviation in first-order NTF using finite gain approximation, for an amplifier with 20dB DC gain. 127

A.4 Reduction in theoretical SQNR from approximating $H(z)$ for a third-order DSM. 128

LIST OF TABLES

Table | Page

4.1 OptiMASH ADC Summary of Performance 87

4.2 OptiMASH Prime ADC Summary of Performance 92

5.1 Ringamp-based Pipeline ADC Summary of Performance 108

For my parents.

AUTOMATED VERILOG-TO-LAYOUT SYNTHESIS OF ADCS USING CUSTOM ANALOG CELLS

CHAPTER 1. INTRODUCTION

"Then as it was, then again it will be

And though the course may change sometimes

Rivers always reach the sea."

— Led Zeppelin (*Ten Years Gone*)

1.1 Motivation

Since the formulation of Moore's Law in the 1960's, the effects of semiconductor process scaling continue to reduce feature sizes and increase transistor densities [1]. The scaling trends (compiled from annual reports from the International Technology Roadmap for Semiconductors [2]) are shown in Figure 1.1. The effects on individual transistors are increased transition frequency (f_t), higher transconductance (g_m), lower output impedance (r_o), and lower supply voltages (necessary for device reliability). Digital circuits benefit greatly from these effects: they achieve increased operating speeds, and decreased power and area. Furthermore, the low sensitivity to physical layout allows digital design to be automatically synthesized from Verilog code. When implementing the same digital design in a smaller process, the same Verilog source may be reused for a fast time-to-market.

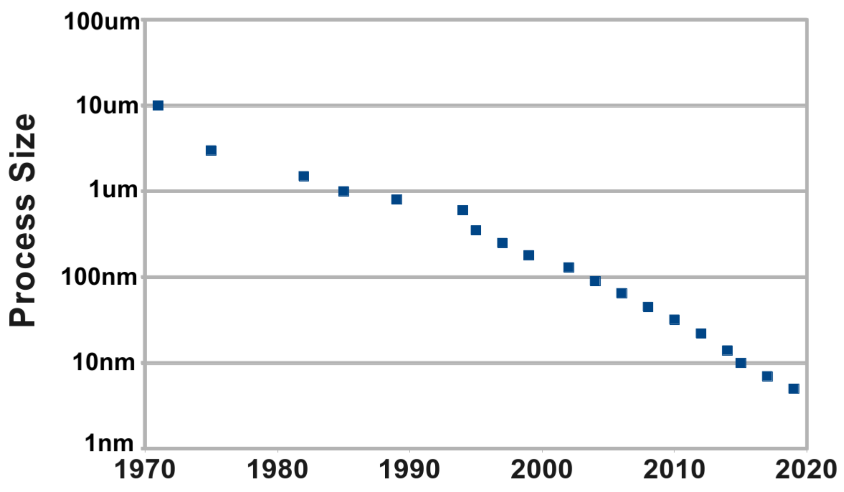


Figure 1.1: Scaling trend in CMOS process sizes.

While analog circuits benefit from faster speeds in smaller process nodes, they suffer from many other drawbacks. Since r_o decreases faster than g_m increases, these devices have reduced intrinsic gain ($g_m r_o$) [3]. Smaller supply voltages mean less available signal swing (reducing signal-to-noise ratio) and less headroom for cascodes. Furthermore, the designer faces laborious redesign and manual layout. The result is that process scaling provides no guarantee of better analog performance, *and* requires high design cost.

Mixed-signal design inherently contains a combination of analog and digital blocks, such that some parts may be re-synthesized from the same Verilog code while others require a costly custom design and layout. Consider as a case study the Atmel ATMega128 microcontroller shown in Figure 1.2. The microcontroller is almost entirely a digital design, which could be described in Verilog code and rapidly ported among different process nodes. However, consider the highlighted ADC... one pesky analog block within a sea of synthesizable digital logic. This will require manual redesign and layout in a new process, creating a bottleneck in the design. Furthermore, the performance specifications for this ADC are quite moderate: the two operating points are 10-bit resolution at 15kS/s, or 8-bit resolution at 76.9 kS/s [4]. As shown in Figure 1.3, these are far from state-of-the-art [5].

Considering the modest performance specifications, is it possible that by adding a few rudimentary analog components to the digital standard cell library, the ADC block could be synthesized along with the rest of the microcontroller? Certainly an automatically routed ADC layout would never outperform that of a manual design. However, for mixed signal designs with moderate speed and resolution requirements, the much faster design time may be worth the power/area overhead.

1.2 Thesis Structure

The remainder of this thesis will be structured as follows. Chapter 2 begins with a review of how synthesis and place-and-route tools are currently used for digital design, and offers a vision of how analog Verilog-to-layout synthesis could be implemented. Chapter 3 summarizes previous work in the synthesis of analog circuits. Chapter 4 presents the design and synthesis of a 1-1-1 MASH $\Delta\Sigma$ ADC, along with measurement results for prototypes in both 130nm and 65nm CMOS. Chapter 5 presents the design and synthesis of a Nyquist-rate pipeline ADC, with measurement results. Finally, Chapter 6 summarizes the thesis and offers suggestions for future work in this area.

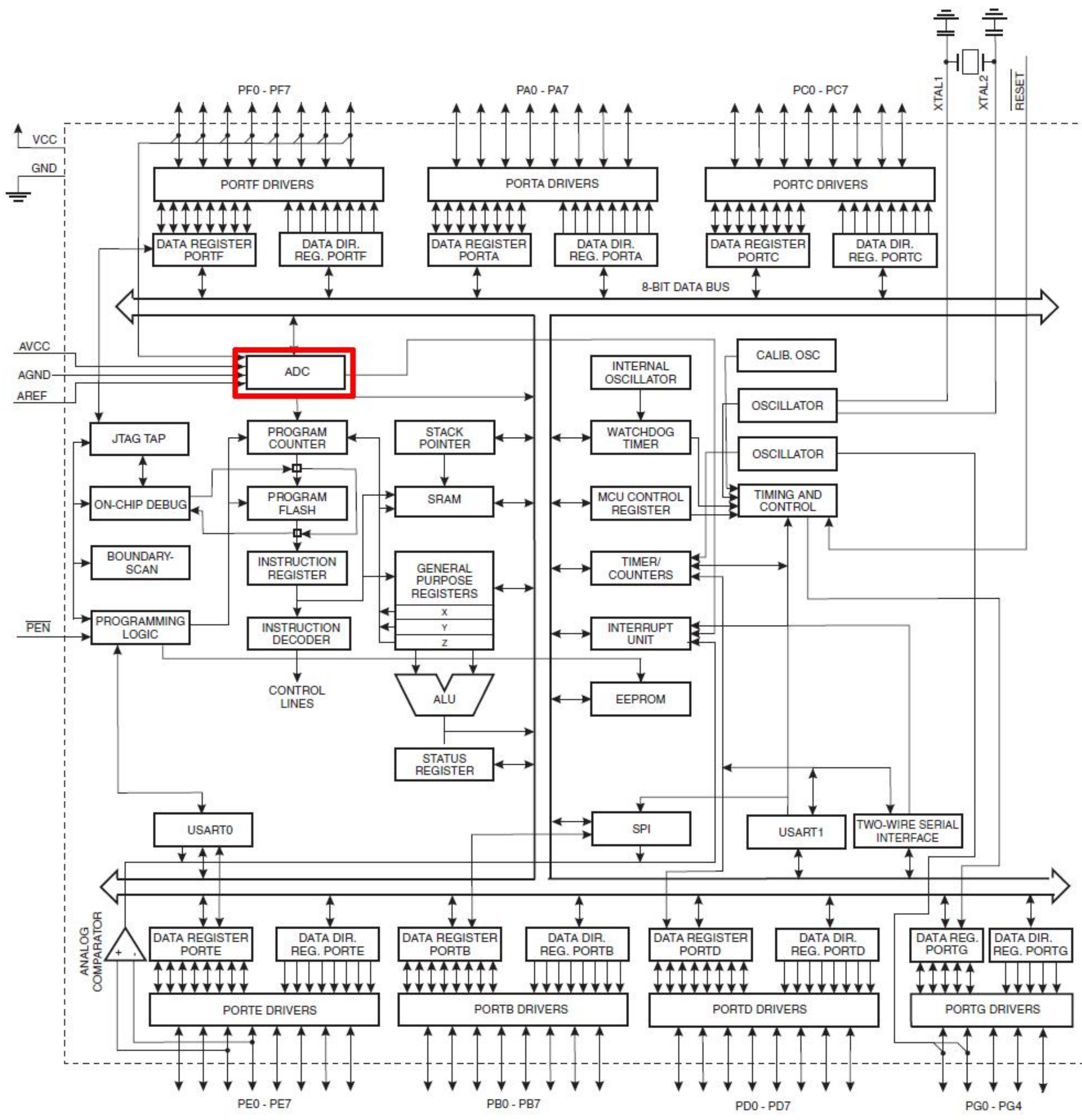


Figure 1.2: Block diagram of ATMega128 microcontroller, highlighting ADC.

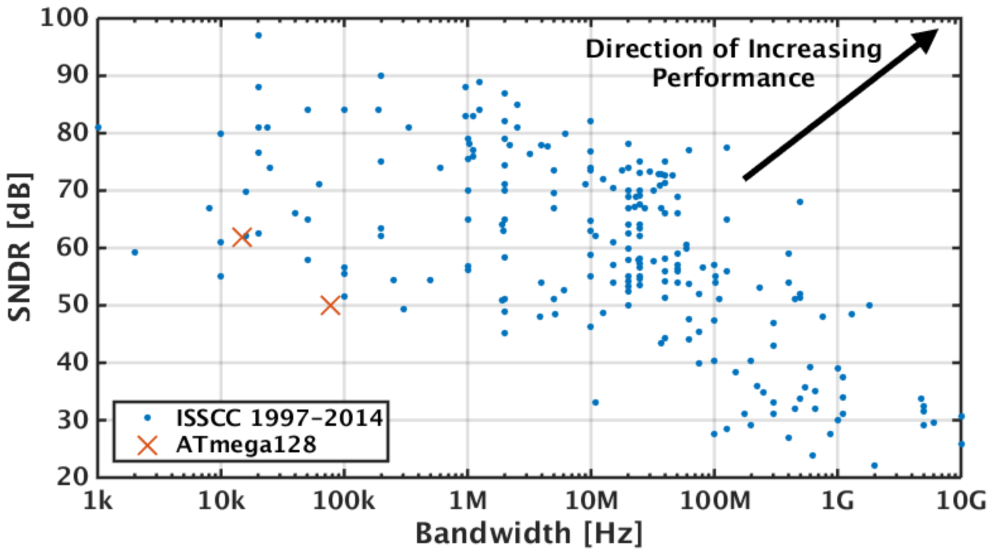


Figure 1.3: Survey of modern ADC performance, compared to ATMega128 example.

CHAPTER 2. PRIOR WORK

"The past, like the future, is indefinite and exists only as a spectrum of possibilities."

— Stephen Hawking

2.1 Prior Work in Highly Digital ADCs

2.1.1 Highly Digital Flash

In flash ADCs, effective resolution is limited by comparator offsets. Traditionally these offsets are controlled with analog techniques such as device sizing, trimming, offset nulling, and averaging. However, these have adverse effects on performance by increasing area and power consumption. Furthermore, these techniques do not scale easily to modern CMOS processes, so designers are forced to consider more scalable, digital-like solutions for controlling offsets.

Figure 2.1a shows a conventional flash ADC, for which there are strict input-referred offset requirements. The solution presented in [6] is to use redundancy and create four different comparators with ideal trip-points at each code transition. Figure 2.1b shows a digital calibration engine that selects the closest of the four trip-points to each ideal value, achieving good linearity and greatly reducing the individual comparator offset requirements. The work in [7] studies the design trade-offs of this approach with a generalized number R of redundant comparators at each trip-point.

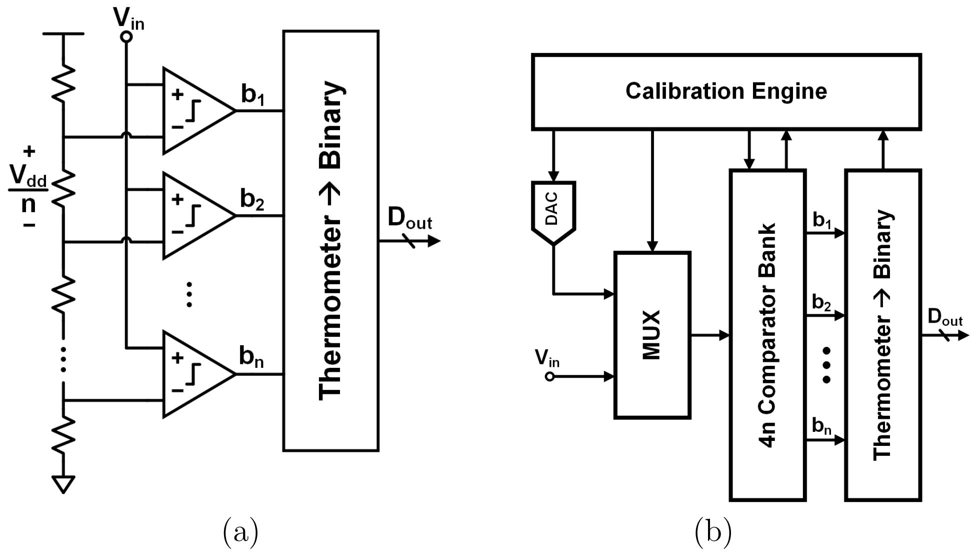


Figure 2.1: Modifying the (a) standard flash ADC to create a (b) redundant flash.

The flash ADC implemented in [8] uses a slightly different method of comparator redundancy to create a scalable flash ADC. It uses a large number of instances of the comparator shown in Figure 2.2. Rather than generate references using a resistor ladder, this comparator creates static voltage offsets by varying the number of minimum-sized input devices that are combined in series and in parallel. Since these thresholds will vary significantly, comparator redundancy is exploited and a digital background calibration scheme estimates the cumulative distribution function (CDF) in order to select a linear subset of the comparator bank. The redundancy and calibration allow the ADC to scale very well with supply voltage, but with two banks of 127 comparators achieves an effective number of bits (ENOB) of only 5.56-bits.

Similarly, [9] implements a highly digital stochastic ADC with comparators that are implemented from minimum-sized devices. It uses identical comparator instances and relies on the random mismatches in offset voltage, as shown in Figure

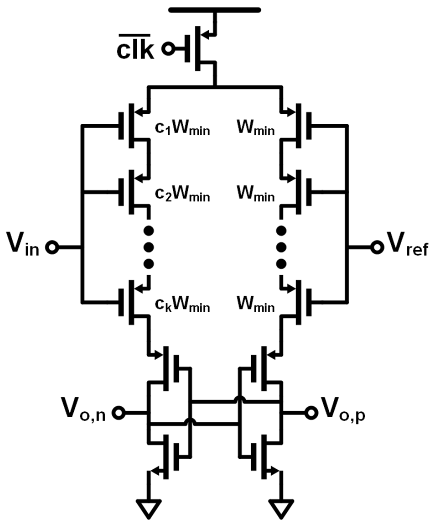


Figure 2.2: Highly digital dynamic comparator, with k input device stack of variable widths.

2.3. One downside of this approach is that the input signal swing is only 280 mV_{pp} (differential). Another drawback of the stochastic flash is that the resolution scales more poorly than a conventional flash ($n = 4^N$ as opposed to $n = 2^N$). For this reason, 1152 comparators achieve a limited 33.6-dB SNDR.

While comparator redundancy in flash ADCs is more scalable than conventional analog techniques, the inflated number of comparators is an expensive area requirement. The following subsections explore architectures other than flash that are digital-like and can scale well to smaller process nodes.

2.1.2 Basic Time-to-Digital Converters

As mentioned in Section 1.1, shrinking supply voltages in modern processes reduces SNR. As shown in Figure 2.4a, if resolution is kept constant while supply

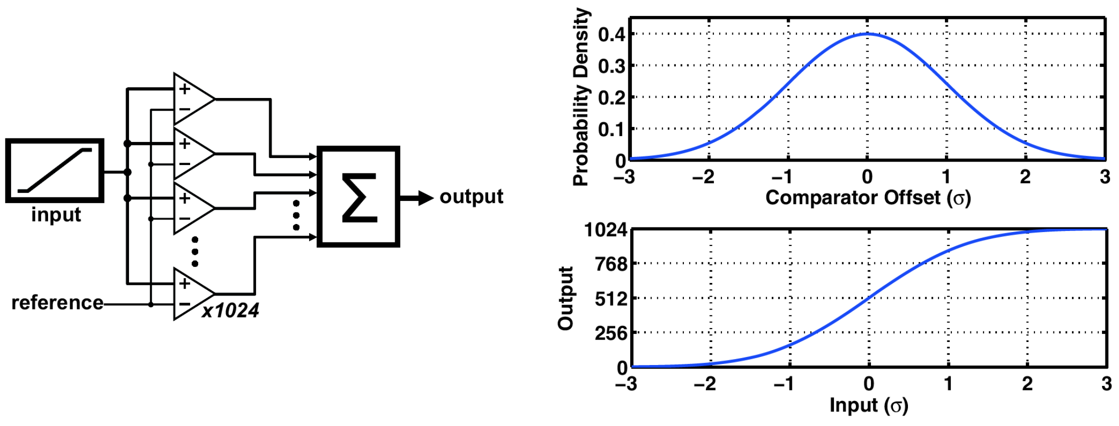
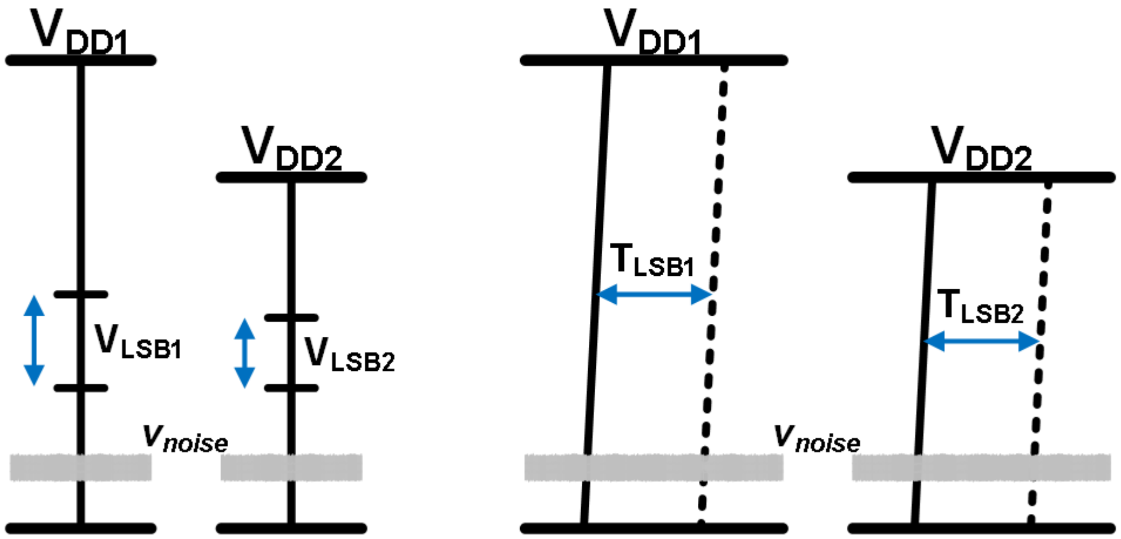


Figure 2.3: Toolchain used for digital circuit synthesis.

shrinks, then the size of a least-significant bit (LSB) decreases relative to the voltage noise. However, the time-domain LSB for a TDC suffers much less. While the voltage noise will translate into timing jitter, the effect is much less pronounced than for the ADC.

For this reason, a possible solution for process scaling is to use a front-end voltage-to-time converter (VTC) block followed by the scalable TDC. Overall this achieves an A/D operation. Some basic TDC architectures are shown in Figure 2.5. The basic flash TDC shown in 2.5a requires only digital cells and is highly synthesizable. This TDC counts the number of delay elements that the input time pulse propagates through until the reference clock edge arrives. However, the time resolution is limited to the minimum gate delay in the process [10, 11]. An improved structure is the Vernier TDC shown in Figure 2.5b, which adds a delay line to the T_{ref} path. The delay elements in the reference line have slightly shorter delays than those in the input chain ($\tau_1 > \tau_2$), such that the time resolution is now the difference between the individual delay elements. These Vernier TDCs overcome the limitation from the minimum gate delay in the process, yet still only



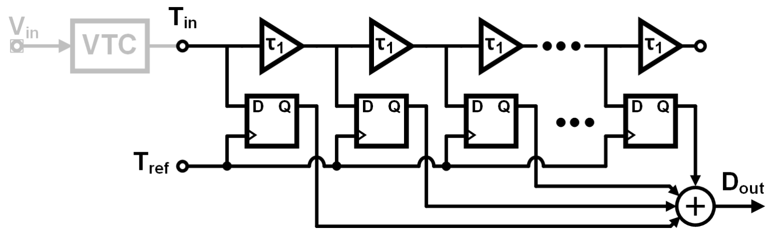
(a) (b)

Figure 2.4: Effect of voltage noise with scaling supply voltage for (a) ADC and (b) TDC.

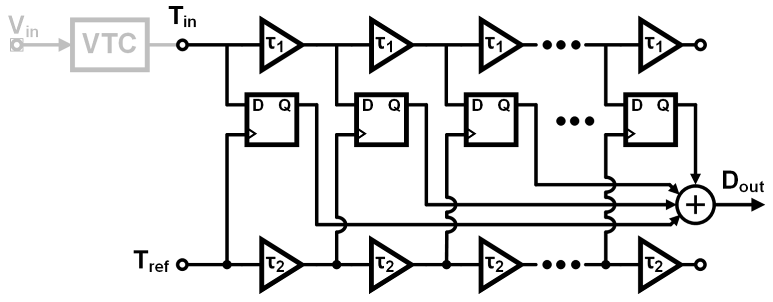
achieve moderate resolution.

$$\begin{aligned} T_{res,flash} &= \tau_1 \\ T_{res,vernier} &= \tau_1 - \tau_2 \end{aligned} \tag{2.1}$$

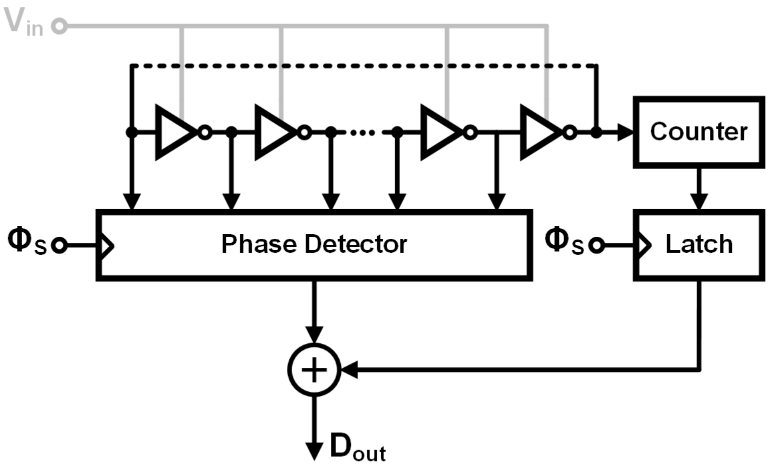
Other previous work [12, 13, 14, 15] has implemented the VCO-based quantizer shown in Figure 2.5b. This structure uses a ring oscillator delay chain to count the number of delays that occur within the sampling period. The VTC operation is inherent, by using the input voltage as the supply for the inverter ring. The benefit is that an analog VTC block is unnecessary and the structure is entirely standard cell elements, but the disadvantage is that the VTC is very nonlinear.



(a)

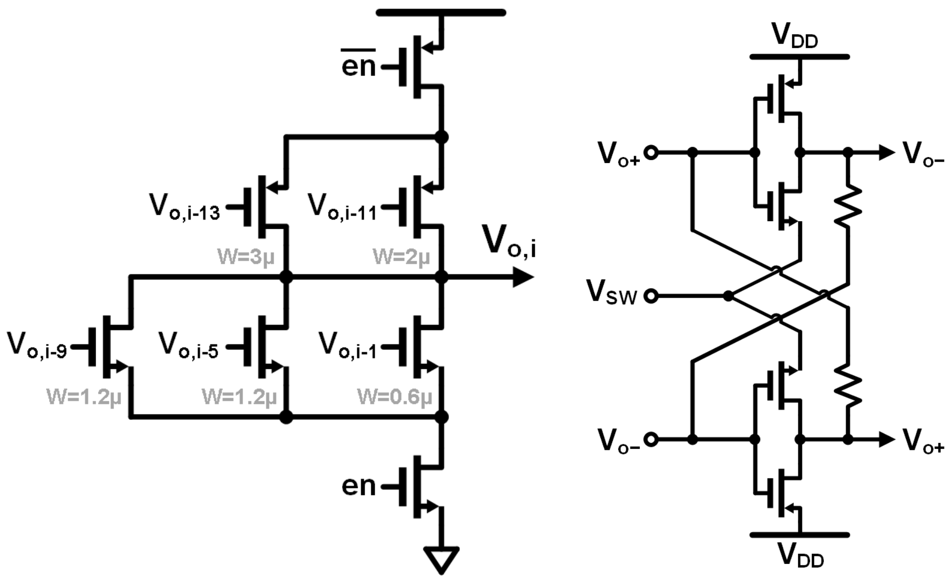


(b)



(c)

Figure 2.5: Examples of TDC architectures: (a) flash TDC, (b) Vernier TDC, (c) VCO-based TDC. Voltage-to-time operation displayed in gray.



(a) (b)

Figure 2.6: TDC delay cells used in (a) multi-path GRO and (b) SRO.

More recent work has used a multi-path gated ring oscillator (GRO) to achieve very fine time resolution [16], but the added complexity to the delay elements push this further from our goal of digital-like design. Similarly, [17] creates a switched-ring oscillator (SRO) that improves upon the GRO but still requires analog complexity in the delay cell and the front-end VTC. While many blocks in the GRO/SRO are highly digital, overall it is not attractive for synthesized ADC design.

2.1.3 Highly-Digital $\Delta\Sigma$ Modulator

One solution to the linearity problems in the VCO quantizer (see Figure 2.5c) is to use it as the quantizer within a $\Delta\Sigma$ modulator ADC. As shown in Figure 2.7, this pre-distorts the input to the VCO to suppress the quantizer nonlinearity

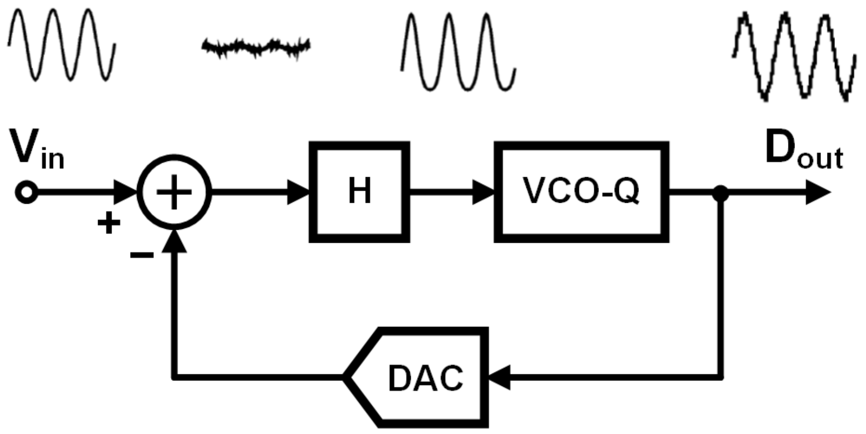


Figure 2.7: Using a DSM loop to suppress VCO nonlinearity.

and improve resolution [18, 19]. While this allows the highly digital VCO-based quantizer to achieve high linearity, it requires many analog blocks within the $\Delta\Sigma$ loop, such as integrators and feedback DACs.

The work in [20, 21] similarly uses a VCO-based quantizer in a $\Delta\Sigma$ ADC, but adds a digital calibration scheme and self-dithering technique that allow the ADC to achieve high resolution without analog integrators, comparators, voltage references or feedback DACs. The only analog requirement is an input V/I converter. By using mostly digital circuitry, it consumes less area and is better suited for technology scaling. These benefits are expected to improve even further with continued process scaling.

2.2 Prior Work in Synthesized Analog Blocks

While the work discussed in Section 2.1 are more compatible with process scaling, they still require a time-consuming custom layout. The motivation for this work is to find architectures that are amenable to both process scaling and to

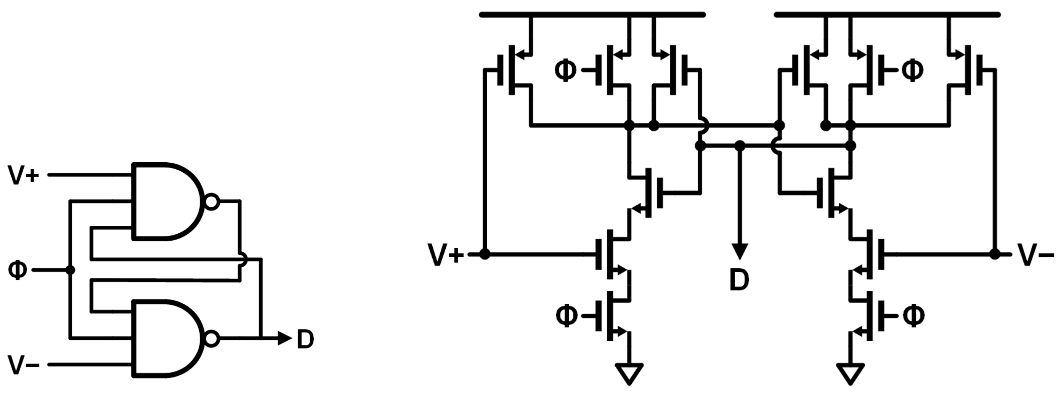


Figure 2.8: An analog comparator constructed from (a) two cross-coupled NAND3 logic gates, to achieve the (b) transistor-level schematic for a comparator.

automated layout. This section studies prior work that automates the layout of an ADC, as well as other design automation tools for analog circuit design.

2.2.1 Synthesized Stochastic Flash ADC

Building on concept described in Section 2.1.1, the work in [22] synthesizes a stochastic flash ADC using an all-digital comparator cell. As shown in Figure 2.8, the comparator is constructed from cross-coupled NAND logic cells. An ADC similar to the one shown in Figure 2.3 is fully synthesized from Verilog code to layout. This offers accelerated design time that is robust to the random mismatch of automated place-and-route, because the only analog signals are the inputs. However, the resolution is still extremely limited (peak is 35.9-dB SNDR at low input frequency). The ability to include some basic analog functionality in addition to the standard digital library would allow the synthesized ADC to achieve greater performance [23].

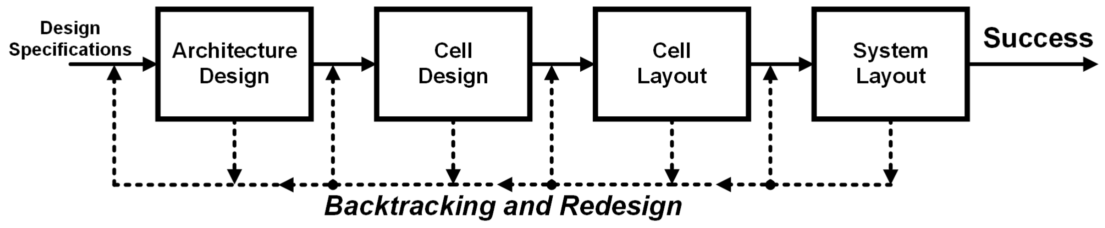
2.2.2 Analog Design Automation

In the past, many designers have made effort to automate portions of the analog circuit design process. The phrase “analog synthesis” has been used to mean many different things, causing confusion and skepticism within the mixed-signal design community. This section will summarize and differentiate between the different analog design automation that has been performed in the past, so that the new contribution of the work presented here will be apparent. Figure 2.9 shows the traditional analog design process and highlights the points at which other tools have provided some level of automation.

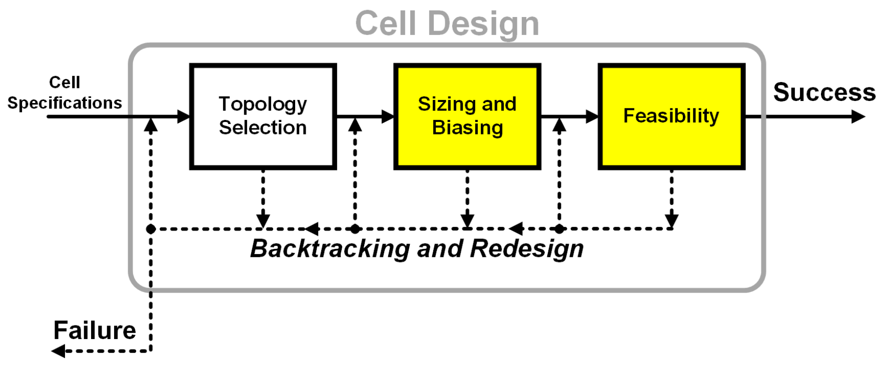
First, consider the ASTRX cell design automation tool [24]. This automates the steps highlighted in Figure 2.9b; given the designer’s description of the topology (in a format similar to SPICE) and a list of performance specifications, it automatically sizes and biases the circuit, then verifies whether the design will hold up to process variations.

A second example of previous analog design automation is the KOAN tool, for device-level placement and routing [25]. The KOAN algorithm successfully handles symmetries and other important analog layout concerns, but was demonstrated in an archaic 2μm CMOS technology, for which the design rules are much more simple than in modern nanoscale processes. It is demonstrated for a simple opamp and a comparator.

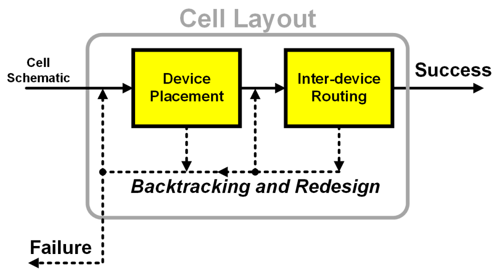
The final example of previous analog design automation is shown in Figure 2.9d by the WREN tool, which automates system-level routing. After the designer places all the sub-blocks, this tool optimizes the routing among the blocks to ensure coupling parasitics are within user-defined limits [26]. Other tools provide similar automation, but attempt to minimize area or wirelength. While this is a useful



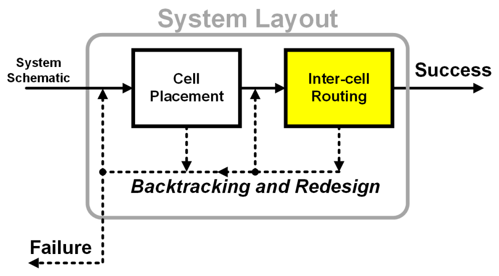
(a)



(b)



(c)



(d)

Figure 2.9: Previous work in automating the (a) traditional analog design process, showing hightlighted blocks where automation has been achieved by (b) ASTRX, (c) KOAN, and (d) WREN.

automation tool, it requires that the designer place the sub-blocks manually, which requires some knowledge of the optimal routing beforehand.

2.3 Summary

Since digital circuits scale better than their analog counterparts, some recent research has focused on creating highly-digital ADC architectures. Redundant and stochastic flash ADCs scale well with process but achieve low resolution. Combining a front-end VTC with a back-end TDC takes advantage of the very scalable time-domain circuits, and pushes all the analog complexity to the front-end VTC. However, achieving high resolution still requires a more complicated, analog-intensive delay cell. This is not in line with the goals of synthesized ADC design.

Previous analog design automation has provided “synthesis” in cell design, cell layout, and system-level routing. But in each case, taking an architectural description to layout still requires some manual and time-consuming layout. To avoid manual layout, the traditional analog design process must be replaced with a different process that is more conducive to automation. The next Chapter describes a vision for a new process, based on the highly-automated digital synthesis and place-and-route tools that already exist.

CHAPTER 3. PROPOSED ANALOG SYNTHESIS TOOLCHAIN

"I suppose it is tempting, if the only tool you have is a hammer, to treat everything as if it were a nail."
— Abraham Maslow

This Chapter describes a procedure for creating library information for analog cells and adding them alongside the standard digital cells, such that Verilog code describing analog functions can be synthesized into a layout. Using a minimal set of these rudimentary analog components (single-stage opamp, transmission gate, comparator, resistor and capacitor) ADCs may be described in Verilog code, then synthesized and automatically laid out. The same Verilog code may be used in different CMOS processes, demonstrating the rapid portability and scalability of this design technique. In each new process, like the logic gates in the digital standard library, layout and routing information for the custom analog cells must be created and then provided to the design automation tools. The creation of this library information is a one-time occurrence that can then be used in any number of analog designs within that same process node, just like the digital standard library.

3.1 Existing Digital Synthesis Toolchain

Figure 3.1 illustrates how digital circuits are automatically synthesized from Verilog code. First, the designer writes a register transfer level (RTL) Verilog file that describes the logic function of the circuit. Several examples are shown in Figure 3.2, each creating a 2:1 multiplexer with different levels of abstraction [27]. For example, the code snippet shown in Figure 3.2a uses a logical/Boolean expression for the multiplexing, at a high level of abstraction. While it very likely would arrive at the gate-level schematic shown in Figure 3.2d, it is possible that the logic synthesizer would instead use a different topology, but it is of no concern to the designer. The Verilog code in Figure 3.2b instead uses a primitive gate-level description. This lower level of abstraction specifies which gate functions should be used and how they should be wired, guaranteeing that the logic synthesizer arrives at the schematic shown in Figure 3.2d. However, the primitives are not process-specific, and this Verilog code could be reused in any number of different technologies. In contrast, the module in Figure 3.2c replaces the primitives with process specific logic gate names: for example, AND2X1_DN_QQD1 is a specific two-input AND gate with single (X1) drive strength. For a digital design this is not useful, it forces the Verilog to be rewritten to be used in a different process. However, it will be useful for the proposed analog circuit synthesis, as will be discussed in Section 3.3.4.

The second step in the toolchain uses the logic synthesizer tool (Cadence's "Design Compiler", in this example) to convert the RTL Verilog into gate-level Verilog. This requires information about the specific semiconductor process from the foundry, in order to determine which logic gates to use to achieve the RTL logic functions. This is provided in the .LIB file as shown in Figure 3.1. It is sometimes

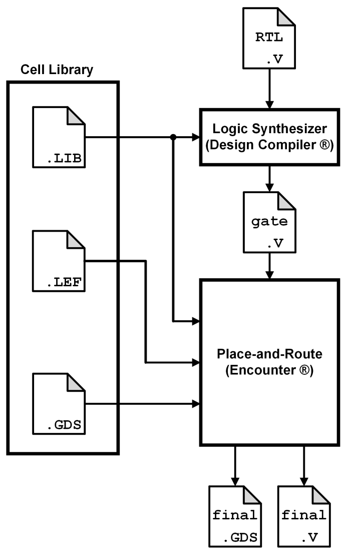


Figure 3.1: Toolchain used for digital circuit synthesis.

provided in .DB or .TLF format instead, but will be labeled as a .LIB file here. The contents of this file will be described in detail in Section 3.2.3.

There may be multiple versions of cells performing the same logic function (for example, a three-input NAND gate), providing different sizes of the cell that are capable of driving larger/faster outputs at the expense of power consumption. Given the RTL Verilog, the logic synthesizer will find many different combinations of digital standard cells that all achieve the desired logic function. The designer may provide constraints regarding timing, power consumption, and area that will guide the synthesizer in optimizing the gate-level Verilog code. For these reasons, a logical/Boolean Verilog description as in Figure 3.2a may be synthesized into different topologies or versions of the same topology with different gate sizes. Any process-specific gates (as in Figure 3.2c) are passed directly to the gate-level Verilog, unchanged.

The third and final step in Verilog-to-layout synthesis is to convert the gate-myMux.v

```
module multiplexer(input a, b, s, output w);
  assign w = s ?  b :  a;
endmodule
```

(a)

myMux.v

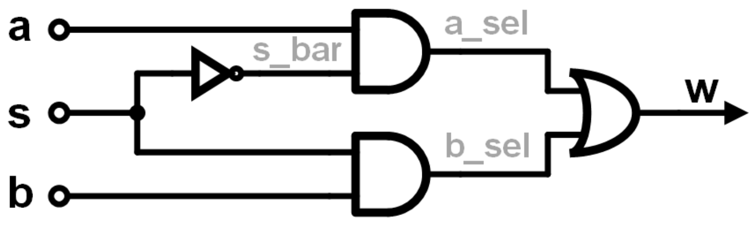
```
module multiplexer(input a, b, s, output w);
  wire a_sel, b_sel, s_bar;
  not U1 (s_bar, s);
  and U2 (a_sel, a, s_bar);
  and U3 (b_sel, b, s);
  or  U4 (w, a_sel, b_sel);
endmodule
```

(b)

myMux.v

```
module multiplexer(input a, b, s, output w);
  wire a_sel, b_sel, s_bar;
  INVX1_DN_QQD1  U1 (s_bar, s);
  AND2X1_DN_QQD1 U2 (a_sel, a, s_bar);
  AND2X1_DN_QQD1 U3 (b_sel, b, s);
  OR2X1_DN_QQD1  U4 (w, a_sel, b_sel);
endmodule
```

(c)



(d)

Figure 3.2: Instantiations in Verilog can be (a) logical, (b) primitive gate-level, or (c) process-specific gate-level. All describe the same (d) schematic.

level Verilog into a layout, using a place-and-route (PNR) tool. Cadence's "Encounter" is used in this work. This requires additional information about the standard digital library, as shown in Figure 3.1. The additional information is related to layout and routing of each standard cell, and will be described in Sections 3.2.1 and 3.2.2 respectively. It is possible that the routing will be sufficiently long that the PNR tool will deem it necessary to add digital buffers at certain points in the layout. These do not affect the logic function, but will cause the final Verilog file to be slightly different than the gate-level Verilog. The last output file is the top-level GDS layout.

3.2 Library Information for Digital Standard Cells

This Section provides information about each of the three library files used in digital synthesis. It explains what the files contain and how that information is used by the software tools. Code snippets of example library information are included to assist the reader. While examples are useful to illustrate the main points, it is impossible to include all the variations and subtleties here. References to more detailed and thorough User Guides are included for each file type, and will be indispensable for further work.

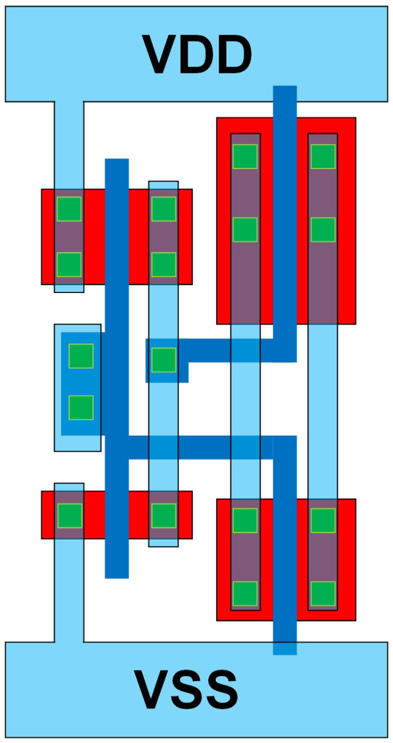
3.2.1 Graphic Database System (GDS) Information

The GDS or GDSII file format is a binary description of geometry of the integrated circuit (IC) layout. It is organized hierarchically and contains the layer, location, and size of each material in the layout. This includes any text (such as pin labels) added to the layout. It does not contain any information about the function of its contents, and there is no guarantee that the contents meet the Design Rule Check (DRC). Since it is a binary file it is not informative to go into detail here, but the interested reader should consult the Format Manual [28]. For our purposes, just consider that the GDS file describes the geometric layers of the layout, and can be visualized as the IC layout itself as shown in Figure 3.3.

3.2.2 Library Exchange Format (LEF) Information

Library Exchange Format (LEF) files contain information related to routing. It can be divided into two categories: technology information and individual cell information (called "macros"). The technology LEF contains definitions for each metal and via layer used in the process, including sizing, spacing and antenna information to meet DRC requirements. For metal layers, it describes the parasitic resistances and capacitances, and defines a routing direction. Finally, it defines different contacts that may be used by the place-and-route tool. Examples of the layer and contact definitions are shown in Figures 3.4 and 3.5, respectively.

The macro LEF file contains descriptions of each of the standard cells in the library. Another LEF file will contain macros for each of the digital standard cells. It identifies the dimension of each cell and the locations of its input/output pins. Each pin is identified by the appropriate metal layer(s) and areas. Diffusion and gate areas are specified for each pin to avoid antenna errors. Finally, "obstruction" areas are specified, to prevent the place-and-route tool from causing shorts with the metal routing. Examples of a cell macro and pin statement are shown in Figure 3.6. While this example gives the reader a feel for the information and organization of the file, there are many more options and variations that can be used. Cadence's



(a)

tgateX1.gds

```
Address  || Data
0000000  || 0600 0200 0500 1c00 0201 7000 0a00 0a00
0000010  || 0b00 0e00 1600 7000 0a00 0b00 0b00 2800
0000020  || 2600 0e00 0602 5953 414e 4344 442e 0042
0000030  || 1400 0503 413e 3789 c64b efa7 4439 2fb8
0000040  || 9ba0 515a 1c00 0205 4500 0c00 1f00 1000
0000050  || 0000 0000 7000 0a00 0a00 0a00 2300 0f00
0000060  || 0c00 0606 4c54 695f 766e 3158 0400 0008
0000070  || 0600 020d 0200 0600 020e 0000 7c00 0310
     ... || ...
0000a30  || 0007 0400 0004 0000 0000 0000 0000 0000
0000a40  || 0000 0000 0000 0000 0000 0000 0000 0000
```

(b)

Figure 3.3: Example of how a standard cell might appear in (a) layout view, and (b) a hexdump of the binary GDS file describing it.

tech.lef

```
LAYER metal1
  TYPE ROUTING ;
  DIRECTION HORIZONTAL ;
  PITCH 0.56 ;
  OFFSET 0.28 ;
  WIDTH 0.23 ;
  AREA 0.20 ;
  SPACING 0.23 RANGE 0 9.995 ;
  SPACING 0.6 RANGE 10.0 35.0 ;
  # Resistance is in ohms/square
  RESISTANCE RPERSQ 0.082 ;
  # Area cap is in pF/um^2 using C/A=epsilon/d
  CAPACITANCE CPERSQDIST 0.0000186 ;
  # Edge cap is in pF/um
  EDGECAPACITANCE 0.0001177 ;
  THICKNESS 0.52 ;
  ANTENNAAREARATIO 200 ;
  ANTENNASIDEAREARATIO 400 ;
END metal1
...
LAYER via1
  TYPE CUT ;
  SPACING 0.26 ;
  ANTENNAAREARATIO 20 ;
END via1
```

Figure 3.4: Example technology LEF layer definitions.

tech.lef

```
VIA M1_M2 DEFAULT
  RESISTANCE 10 ;
  LAYER metal2 ;
    RECT -0.14 -0.19 0.14 0.19 ;
  LAYER via1 ;
    RECT -0.13 -0.13 0.13 0.13 ;
  LAYER metal1 ;
    RECT -0.19 -0.14 0.19 0.14 ;
END M1_M2
...
VIA M1_POLY DEFAULT
  RESISTANCE 12.5 ;
  LAYER metal1 ;
    RECT -0.17 -0.115 0.17 0.115 ;
  LAYER contact ;
    RECT -0.11 -0.11 0.11 0.11 ;
  LAYER poly ;
    RECT -0.21 -0.21 0.21 0.21 ;
END M1_POLY
```

Figure 3.5: Example technology LEF contact definitions.

LEF reference manual provides more details [29].

3.2.3 Liberty (LIB) Timing File Information

Functional, area, timing and power information for each cell may be provided in either Timing Library Format (.TLF) or open-source Liberty (.LIB) file format. These ASCII representations may also be compiled into a binary database (.DB) format. Examples of the technology and individual cell information are shown in Figures 3.7 and 3.8, respectively. The logic synthesizer uses the functional information to first identify combinations of standard cells which achieve the desired RTL function, then to pick the optimal gate combination that achieves the timing constraints and area/power specifications provided by the designer.

While the examples here are useful to understand what information is provided, there is much more detailed and general descriptions of the file formatting available in the respective reference manuals [30, 31].

3.3 Creating Custom Analog Cell Information

With an understanding of how the digital toolchain synthesizes a final GDS layout from Verilog, we now move on to the task of adding analog functionality to the standard cell library. We begin by considering what “minimum effective dose” of analog cells will allow us to synthesize analog-to-digital converters. Then the procedure for creating the GDS, LEF and LIB information for these cells is described. This Section concludes with examples of instantiating and floorplanning analog cells in Verilog.

myCell.lef

```
MACRO invx1
  CLASS CORE ;
  FOREIGN invx1 0.000 0.000 ;
  ORIGIN 0.000 0.000 ;
  SIZE 1.980 BY 5.040 ;
  SYMMETRY x y ;
  SITE CORE ;
  PIN Y
    AntennaDiffArea 0.855 ;
    DIRECTION OUTPUT ;
    PORT
    LAYER metal1 ;
      RECT 1.365 1.220 1.805 3.600 ;
    END
  END Y
  PIN A
    AntennaGateArea 0.27 ;
    DIRECTION INPUT ;
    PORT
    LAYER metal1 ;
      RECT 0.175 1.870 1.055 2.435 ;
    END
  END A
  PIN VSS
    DIRECTION INOUT ;
    USE ground ;
    SHAPE ABUTMENT ;
    PORT
    LAYER metal1 ;
      RECT 0.000 -0.400 0.465 0.400 ;
      RECT 0.465 -0.400 0.885 1.500 ;
      RECT 0.885 -0.400 1.980 0.400 ;
    END
  END VSS
  [VDD pin statement]
END invx1
```

Figure 3.6: Example LEF macro statement.

myCells.lib

```
library(library_name) {
  delay_model : table_lookup ;
  time_unit : ``1ns'' ;
  voltage_unit : ``1V'' ;
  current_unit : ``1uA'' ;
  pulling_resistance_unit : ``1kohm'' ;
  capacitive_load_unit(1, pf) ;
  leakage_power_unit : ``1nW'';
  power_supply () { power_rail(VDD, 1.20) ; }

  operating_conditions(TYP) {
    process : 1.0 ;
    temperature : 60 ;
    voltage : 1.20 ; }
  wire_load(``500_CELLS'') {
    resistance : 0;
    capacitance : 0.000140 ;
    area : 0.01 ;
    slope : 36
    fanout_length(1,33) ;
    ...
    fanout_length(5,198) ; }
  [other wire load models]

  slew_lower_threshold_pct_rise : 10 ;
  slew_upper_threshold_pct_rise : 90 ;
  [other threshold definitions]

  input_voltage (I_VDD_120_Rail) {
    vil : 0.1 * VDD ;
    vih : 0.9 * VDD ;
    vimin : -0.5 ;
    vimax : VDD + 0.5 ; }
  output_voltage (O_VDD_120_Rail) {
    vol : 0.1 * VDD ;
    voh : 0.9 * VDD ;
    vomin : -0.1 * VDD ;
    vomax : 1.1 * VDD ; }
  lu_table_template(general_template_7_8) {
    variable_1 : total_output_net_capacitance
    index_1(``1, 2, 3, 4, 5, 6, 7'') ;
    variable_2 : input_net_transition
    index_2(``1, 2, 3, 4, 5, 6, 7, 8'') ; }
  [other LUT template]
  [cell definitions]
}
```

Figure 3.7: Example LIB technology definitions.

myCells.lib

```
cell(INVX1) {
  area :  6.65 ;
  cell_footprint :  inv ;
  pin(Y) {
    direction :  output ;
    output_voltage :  O_VDD_120_Rail ;
    output_signal_level :  VDD ;
    function :  ``(!A)''
    max_capacitance :  0.17 ;
    internal_power() {
      related_pin :  ``A'' ;
      fall_power(LUT size) {LUT}
      rise_power(LUT size) {LUT} }
    timing() {
      cell_rise(LUT size) {LUT}
      rise_transition(LUT size) {LUT}
      cell_fall(LUT size) {LUT}
      fall_transition(LUT size) {LUT}
      steady_state_current_high(LUT size) {LUT}
      noise_immunity_high(LUT size) {LUT}
      steady_state_current_low(LUT size) {LUT}
      noise_immunity_low(LUT size) {LUT}
      timing_sense :  negative_unate ;
      related_pin :  ``A'' ; }
    }
  pin(VDD) { direction :  input ; }
  pin(VSS) { direction :  input ; }
  pin(A) {
    direction :  input ;
    input_voltage :  I_VDD_120_Rail ;
    input_signal_level :  VDD ;
    capacitance :  0.0029 ;
    max_transition :  2.0000 ;
    rise_capacitance :  0.0029 ;
    fall_capacitance :  0.0029 ;
    rise_capacitance_range(0.0029, 0.0029) ;
    fall_capacitance_range(0.0029, 0.0029) ;
    }
  leakage_power () {
    when :  ``(!A)'' ;
    value :  ``9.199620'' ; }
  leakage_power () {
    when :  ``A'' ;
    value :  ``8.225110'' ; }
  cell_leakage_power :  8.712365
}
```

Figure 3.8: Example LIB cell definition.

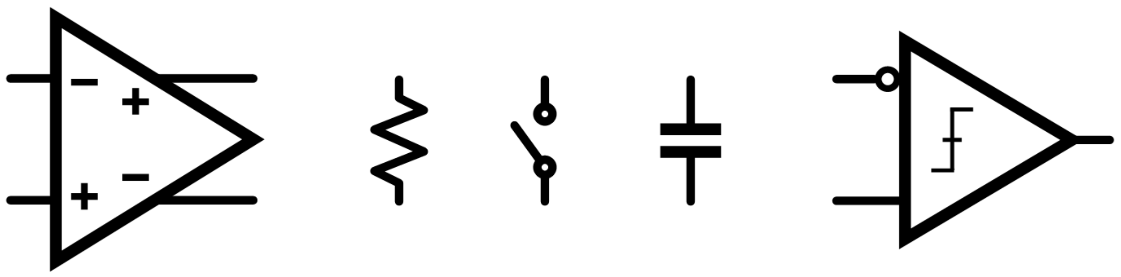


Figure 3.9: A minimal set of custom cells for analog design.

3.3.1 Minimalist Set of Analog Cells

Figure 3.9 shows set of analog cells that enable a designer to synthesize ADCs. The most obvious is a comparator cell, which is necessary for building a quantizer. Second, a unit capacitor and switch cell are needed to implement the switched-capacitor circuits that are fundamental to discrete-time ADCs. To maintain the "digital-like" nature of the synthesized data converters, bootstrapping will not be used. A simple transmission gate should suffice. Third, a unit resistor cell is useful for implementing reference ladders and other simple analog functions. Finally, some amplification technique will prove useful in many different architectures. This is the most analog of the custom cells and will prove the most difficult to design, because the traditional analog techniques like gain-boosting and compensation are too complex for synthesized ADCs. Other techniques like cascoding are not friendly with process scaling, and should be avoided. Simple and scalable amplification solutions will be described in later Chapters.

3.3.2 Creating .GDS File for Custom Cell

Creating a custom cell begins the same as in traditional analog design: a transistor-level schematic is created, then laid out. However, since the cell will be place-and-routed alongside standard digital cells, it must fit within the digital supply pitches. An easy way to start is to copy a filler cell from the digital standard library, and delete everything except the supply rails. If more room is necessary, the rails can be widened or the cell can span multiple supply pitches, as shown in Figure 3.10. To avoid DRC violations with adjacent cells, it is necessary to leave a gap between the layers and the right and left edges of the custom cell. The width of this gap will depend on the DRC rules for the given process.

In Cadence, the layout view will be saved as a Cadence DataBase (.CDB) file type. To stream it out in GDS format, open the Integrated Circuit Front-to-Back (ICFB) window. The default setup may result in a licensing error, so run the following command in ICFB terminal (Figure 3.11a):

```
>> hiSetMenuItemCallback(eval(ExportItem) 'StreamOut
     ``pipoDisplay(transStreamOutForm)'')
```

Similarly, the following command prevents licensing errors for the stream-in process that will be used later:

```
>> hiSetMenuItemCallback(eval(ImportItem) 'StreamIn
     ``pipoDisplay(transStreamInForm)'')
```

To complete the stream-out process, select *File* → *Export* → *Stream...* in the ICFB window. This will open the "Virtuouso Stream Out" menu shown in Figure 3.11b. Complete the menu with the library name and cell name you wish to export. Selected "OK" to begin the PIPO STRMOUT. When it is complete, the tool will notify you that the process was successful as shown in Figure 3.11c.

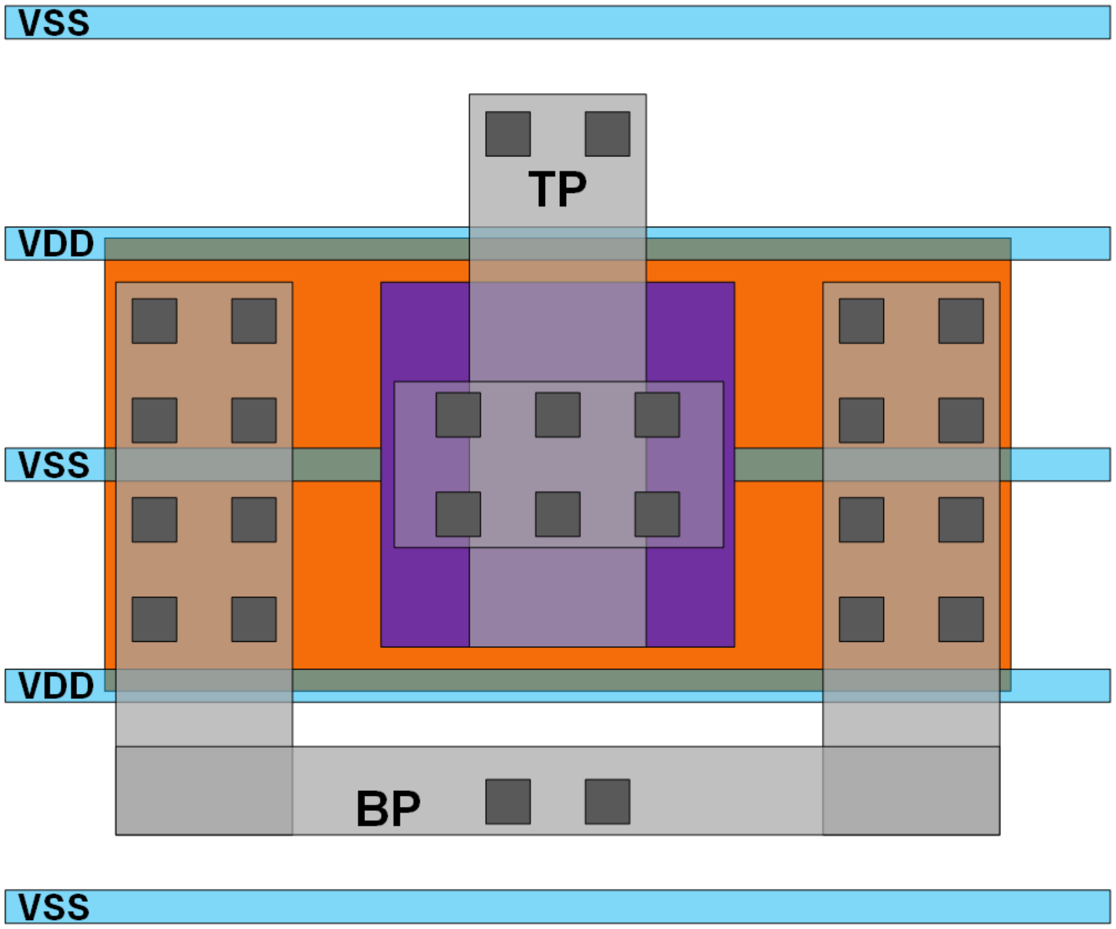
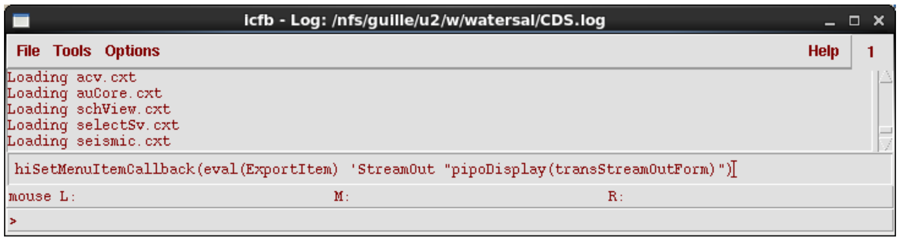
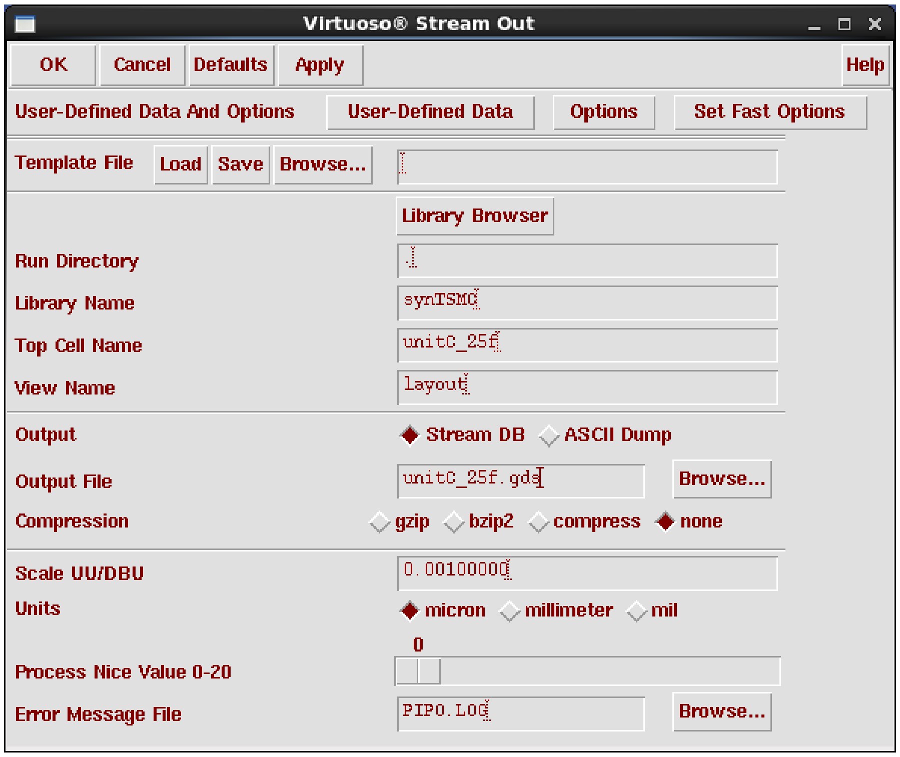


Figure 3.10: Example of multi-pitch custom cell layout, for a MIM capacitor.



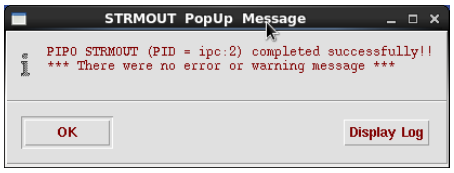
```
Loading acv.cxt
Loading auCore.cxt
Loading schView.cxt
Loading selectSv.cxt
Loading seismic.cxt
hiSetMenuItemCallback(eval(ExportItem) 'StreamOut "pipoDisplay(transStreamOutForm)")
mouse L:                    M:                    R:
>
```

(a)



Template File	
Run Directory	.
Library Name	synTSMC
Top Cell Name	unitC_25f
View Name	layout
Output	[✓ Stream DB] ASCII Dump
Output File	unitC_25f.gds
Compression	gzip bzip2 compress [✓ none]
Scale UU/DBU	0.00100000
Units	[✓ micron] millimeter mil
Process Nice Value 0-20	0
Error Message File	PIPO.LOG

(b)



```
PIPO STRMOUT (PID = ipc:2) completed successfully!!
*** There were no error or warning message ***
```

(c)

Figure 3.11: Streaming out a GDS file for a custom cell.

Details about the stream out process such as the type, layer and shape of each layout polygon will be saved in a logfile.

3.3.3 Creating .LEF File for Custom Cell

Given the completed layout view in Cadence, there is a one-time process for creating a LEF description of each cell to be used by the place-and-route tool. Using the trace cursor in the layout view, the designer may manually enter the pin names, layers and coordinates as seen in the template of Figure 3.6. Similarly, the designer may trace the active/diffusion layers to provide antenna information, and describe the overall cell size and symmetry. The Cadence Abstract tool automates this process of translating cell layouts into LEF files [32]. However, for analog cells such as opamps, the designer may want more control over the "obstruction" areas to prevent the PNR tool from routing above critical nodes. Therefore, the LEF file creation process may be done manually, automatically, or some combination of the two.

While manually writing the LEF information is a tedious process, it is a one-time procedure for which the custom cell library may then be used in any number of analog designs.

3.3.4 Creating .LIB File For Custom Cell

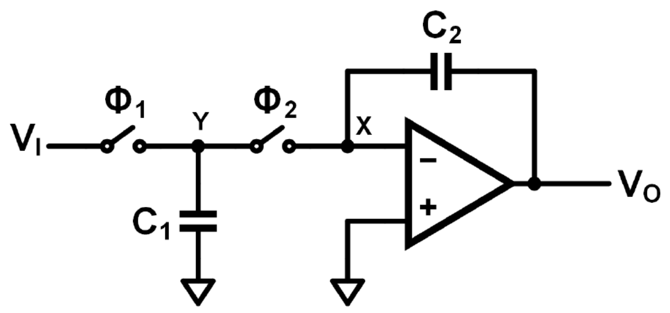
While the procedures described in Sections 3.3.2 and 3.3.3 for creating analog GDS and LEF information were very straightforward, creating analog LIB information is not. Recall the contents of a digital standard cell's LIB description from

Figure 3.8. The area, cell footprint, and leakage power can be determined. For each pin the direction, capacitance, signal levels, and power consumption could be described. But the cell function and pin timing information have no meaning for these analog custom cells. This Subsection will explore the consequences and various solutions to this problem.

The absence of a logical function for the analog cells affects the synthesis of RTL Verilog to gate-level Verilog. If any “fake” function is specified for the custom cells, then it is possible that the synthesizer will choose to use those cells to perform that function in place of real digital cells, and will not achieve the correct results. On the other hand, if no function is specified then the tool will never know to when to use the analog cells. The solution implemented here is to explicitly instantiate the analog custom cells when they are needed, which circumvents the need for any function in the LIB file. An example is shown in Figure 3.12.

Without functional information, the synthesizer will try to optimize the design by removing the function-less analog cells. The designer may prevent this by using the set_dont_touch directive [33]. An example of this is shown in Figure 3.13. This directive preserves a sub-design during optimization, meaning that it applies hierarchically to all cells, nets and sub-designs of the reference it is called upon. In the dc_syn.scr (shown in Figure 3.13b), line 1 applies the directive to only the “Flash” module in the first ADC stage. This could be either a single custom library cell, or a module defined elsewhere in the Verilog code. Line 2 applies set_dont_touch to both the “Flash” and “Opamp” references in the second stage, including whatever sub-blocks they may have. Finally, line 3 applies the directive to the entire third pipeline stage.

If the method shown in Figure 3.12 is used to explicitly instantiate analog cells, then the absence of timing information for the custom cells is no longer a



(a)

myADC.v

```
module invIntegrator(Vi, Vo, phi[1:2]);
  input Vi, phi[1:2];
  output Vo;
  wire VSS, X, Y;

  tieVSS T1( .Y(VSS)                         );
  tgate  S1( .IN(Vi),   .OUT(Y), .A(phi[1]) );
  unitC  C1( .BP(VSS),  .TP(Y)              );
  tgate  S2( .IN(Y),    .OUT(X), .A(phi[2]) );
  unitC  C2( .BP(X),    .TP(Vo)             );
  opamp  A1( .INP(VSS), .INN(X), .OUT(Vo)   );
endmodule ;
```

(b)

Figure 3.12: Example of describing the inverting integrator (a) schematic into (b) Verilog code using explicit instantiations of custom cells.

myADC.v

```
module synthPL(...);
  ...
  PLstage1 U1( ...  );
  PLstage2 U2( ...  );
  PLstage3 U3( ...  );
  PLstage4 U4( ...  );
  backend  U5( ...  );
endmodule ;

module PLstage1(...);
  ...
  Flash U1( ...  );
  DAC U2(...);
  Opamp U3(...);
  ...
endmodule ;
```

(a)

dc_syn.scr

```
set_dont_touch { U1/U1 }
set_dont_touch { U2/U1 U2/U3 }
set_dont_touch { U3 }
```

(b)

Figure 3.13: Using Design Compiler directives to explicitly instantiate analog custom cells without a logical description.

problem either. The synthesizer will use the custom cells if (and only if) they are instantiated in the input RTL Verilog file. Other sections of the Verilog code may still be Boolean or primitive logical descriptions, and will be synthesized from the digital standard cells as normal. A drawback of this approach, however, is that in the absence of timing information, the digital automation tools cannot determine the best way to route the pins of analog custom cells. For example, the synthesizer does not know the input capacitance of the pins to determine the drive-strength of preceding cells (particularly difficult with our new unit capacitor cell). Similarly, the PNR tool doesn't know how sensitive the routing of analog nets will be, so critical analog signal paths may be routed over unnecessarily long distances.

In order to improve the synthesis and place-and-route of analog circuits, some LIB information would be useful. By including *just* the pin capacitance for the analog cells, we ensure that the synthesizer will choose appropriate sizes if a digital standard cell is driving an analog custom cell. This is sufficient for the synthesizer but not for the PNR engine. Two options exist for controlling the automated routing of the PNR tool for analog cells: floorplanning and fake timing information. The first solution is described in Section 3.3.5 and was implemented in the designs described here. The second is proposed as a direction for future work in Section 6.2.

3.3.5 Controlling Floorplanning for Analog Cells

The implemented solution in this work used no timing information in the LIB file, only the pin capacitances. To implement some control over the automated routing, floorplanning was used to pre-place blocks within the chip area [34]. This gives the PNR a starting point to begin the routing optimization; a well-designed floorplan can speed up the run-time and provide better performance. Since our custom analog cells have no timing information to guide the PNR tool, floorplanning will provide a significant performance benefit. An example of floorplanning a pipeline ADC is shown in Figure 3.14.

While the floorplanning of the pipeline stages is useful, the PNR is still operating with little information when routing the blocks within each stage. For this reason, a second level of floorplanning can be done as shown in Figure 3.15. This not only allocates a specific area for each stage, but also pre-places the major functional blocks (flash quantizer, DAC, and opamp) within each stage. This way the designer can improve the routing of the critical analog signal paths, as one would do in a traditional analog layout.

The floorplanning could continue down to lower levels of the hierarchy, pre-placing the cells more and more specifically to achieve better matching and shorter routing between the analog cells. However, this floorplanning will change between designs and between process nodes. While many levels of floorplanning achieves better layout, it sacrifices the portability and rapid design time that is a goal of this project. For this reason, one or two levels of floorplanning seems to be a reasonable compromise between performance and portability.

3.4 Final Toolchain

Following the description of the existing digital synthesis toolchain and an explanation of how custom analog cells may be created and added to the standard library, this Section illustrates the new analog circuit synthesis toolchain. It discusses the capabilities and limitations of the tool, since the word "synthesis" has previously been applied to several different automation techniques for analog

myADC.v

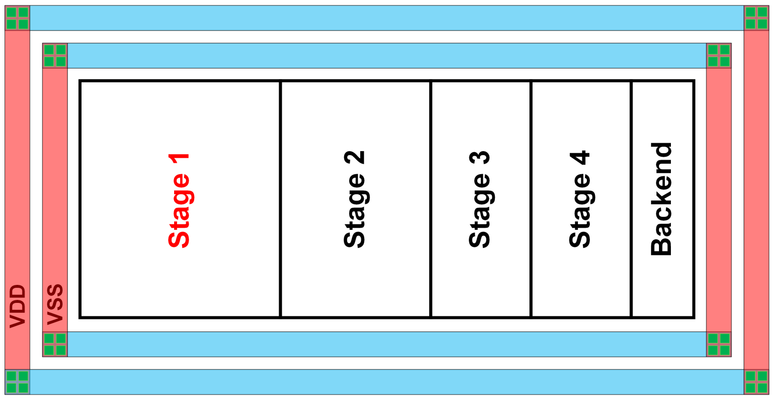
```
module synthPL(...);
  ...
  PLstage1 U1( ...  );
  PLstage2 U2( ...  );
  PLstage3 U3( ...  );
  PLstage4 U4( ...  );
  backend  U5( ...  );
endmodule ;
```

(a)

placeandroute.tcl

```
setObjFPlanBox Module U1 X1 Y1 X2 Y2
setObjFPlanBox Module U2 X1 Y1 X2 Y2
setObjFPlanBox Module U3 X1 Y1 X2 Y2
setObjFPlanBox Module U4 X1 Y1 X2 Y2
setObjFPlanBox Module U5 X1 Y1 X2 Y2
```

(b)



(c)

Figure 3.14: Example of controlling floorplanning with the “setObjFPlanBox” directive. The modules in (a) Verilog are floorplanned in (b) TCL to achieve a good starting point in the (c) layout.

myADC.v

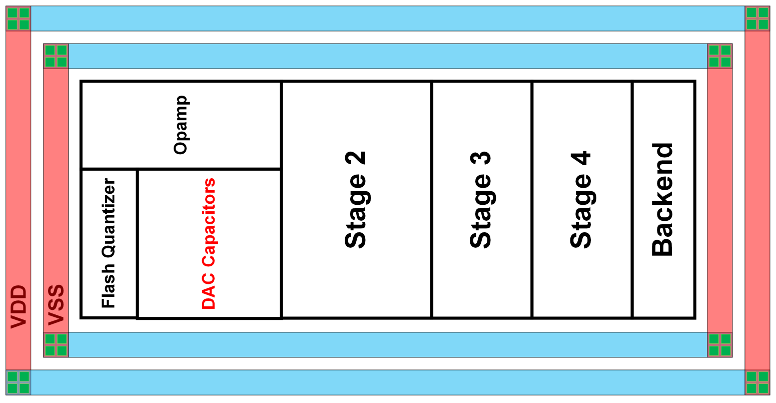
```
module synthPL(...);
   ...
   PLstage1 U1( ...  );
   ...
endmodule ;

module PLstage1(...);
   ...
   Flash U1( ...  );
   DAC U2(...);
   Opamp U3(...);
   ...
endmodule ;
```

(a)

placeandroute.tcl

```
setObjFPlanBox Module U1 X1 Y1 X2 Y2
...
setObjFPlanBox Module U1/U1 X1 Y1 X2 Y2
setObjFPlanBox Module U1/U2 X1 Y1 X2 Y2
setObjFPlanBox Module U1/U3 X1 Y1 X2 Y2
```

(b)



(c)

Figure 3.15: Using PNR floorplanning, one level deeper.

circuits.

3.4.1 New Automated Analog Design Toolchain

The newly created toolchain is shown in Figure 3.16. The similarity to the existing digital toolchain (from Figure 3.1) is a benefit of the automation method proposed here; it relies on software tools that are already commonly used and well-supported. The synthesis and place-and-route script files (.SCR and .TCL) are not new to this toolchain, they are part of the existing digital toolchain but were not shown in Figure 3.1 for simplicity. They are shown here in grayscale to indicate that the designer should add set_dont_touch directives and floorplanning within these files.

3.4.2 Capabilities and Limitations

As explained in Section 2.2.2, there are other analog circuit design automation tools that describe their function as "synthesis". To avoid confusion with these tools, this Section will explain precisely what capabilities our toolchain possesses. It will proceed in order of the traditional analog design flow shown in Figure 2.9a.

This tool *does not* automate the architecture design. The designer must choose the ADC architecture they wish to implement, and make decisions about the number of stages, sub-quantizer resolution, etc. Automating this phase of the design is discussed as a direction for future work in Section 6.2.

This tool *does not* automate the individual cell design or layout. However, the objective of this work is that the number of analog cells required and their

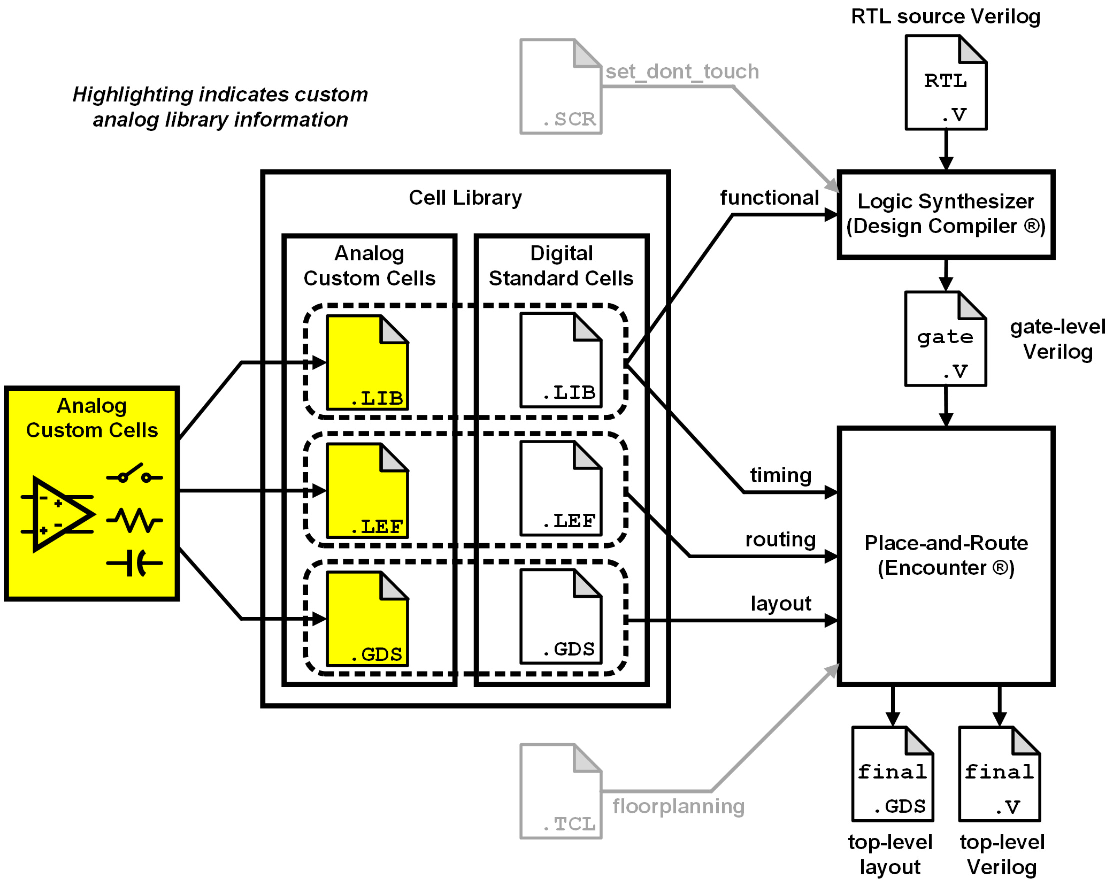


Figure 3.16: Toolchain used for analog circuit synthesis.

complexity will be minimal. Furthermore, the cell design/layout is a one-time occurrence that could then be used in any number of circuits. Ideally it would be performed at the same time as the creation of the standard digital cell library, requiring very little overhead in the context of the combined digital and analog cell creation.

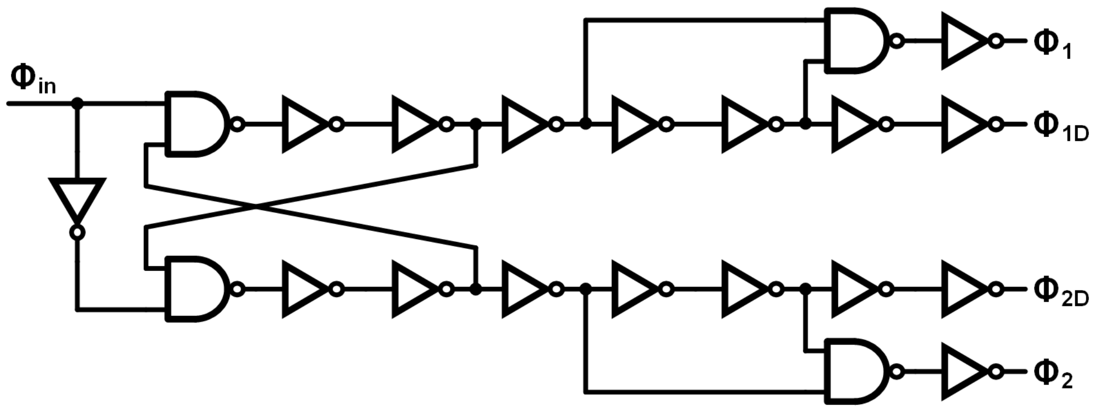
The tool *does* automate the system-level layout, from Verilog to GDS. The designer writes the Verilog description of the circuit, which at some points will contain digital circuit descriptions of Boolean and primitive logic, and at other points will describe the analog portions with a SPICE-like syntax. Given this Verilog file, the synthesis into gate-level Verilog and layout is automated. The same RTL Verilog code containing analog custom cells may be reused in other processes, as long as the naming conventions for the analog library cells are consistent.

One limitation of the toolchain in Figure 3.16 occurs in verification. With the lack of functional information for analog cells, Verilog simulators like ModelSim will be unable to provide any RTL verification [35]. Similarly, analog circuit simulators cannot process the RTL code. Two possible solutions are to rewrite the RTL portions in Verilog-A (which is compatible with many analog simulators), or to run the simulation in mixed-mode using Cadence's Spectre and Verilog-XL tools coupled together [36]. The drawback of this approach is that it requires the designer to create a schematic view of the circuit. The traditional analog designer may not find this bothersome, but it is much slower than being able to simulate directly from the RTL Verilog.

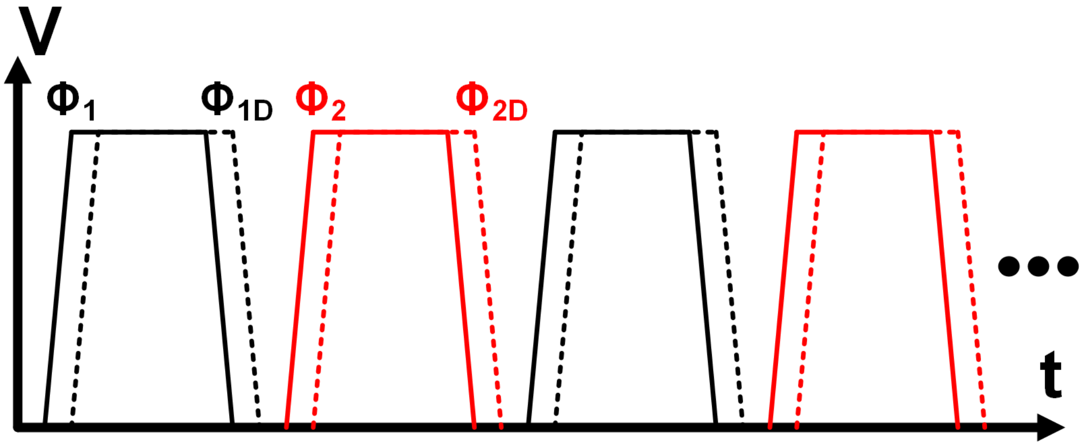
Similarly, if the designer wishes to run post-layout simulations for verification, the mixed-signal synthesis toolchain causes complications. Physical verification tools require that Layout-Versus-Schematic (LVS) be passed before parasitic extraction (RCX or PEX) takes place [37]. For a traditional analog layout, LVS compares the layout to the schematic view (which already exists). For a fully digital layout, the final Verilog-file can be streamed into Cadence's Design Framework II (DFII) software as a schematic view. Then LVS is run to compare the imported schematic to the GDS file[1]. Thus a mixed signal design may run LVS using the schematic imported from the final Verilog code, alongside the manually created schematic for the input/output (I/O) buffers, decoupling capacitors, pad ring, etc. Then extraction can be performed and finally post-layout simulation. Although complicated, the post-layout verification for the synthesized ADC is easier than pre-layout.

One final limitation is with the synthesis of the non-overlapping clock generator circuit that is common in ADC design [39]. A possible implementation with an example output waveform is shown in Figure 3.17. Note that this is constructed entirely from digital standard cells, and would be therefore expected to be easily synthesized from Verilog. The designer could specify timing constraints for the desired non-overlap period and delay period, and the synthesizer will size the inverter delay chains appropriately. Unfortunately, although composed of standard cells, the function of this circuit is far from digital. The use of inverter or buffer delays to intentionally create time delays, and the feedback path to the NAND inputs is non-standard for digital logic design. In this work, the clock phase generator was never successfully synthesized from Verilog code. Even when the topology of Figure 3.17a was explicitly coded in Verilog and input to the synthesizer, it would attempt to optimize the design by removing pairs of the inverters (logically each

[1]In some technologies, the schematic view of the standard cell library is provided. In others, a Circuit Description Language (CDL) file is provided instead to be used for LVS. CDL is a subset of the SPICE format [38] and is equally compatible with LVS.



(a)



(b)

Figure 3.17: A possible non-overlapping clock generator (a) schematic and (b) waveform.

pair has no effect) and disconnecting the feedback delay path. Despite the highly-digital nature of the clock generation circuit, a drawback of this work was that it had to be explicitly instantiated in Verilog and protected with the set_dont_touch directive.

3.5 Summary

In order to enable the automated Verilog-to-layout synthesis of analog circuits, this Chapter studied the existing digital synthesis toolchain. It describes what tools are used, what library information is needed, and how the tools use that information. Then a procedure for creating similar library information for analog custom cells is described. The GDS and LEF information is obtained easily, but the LIB file (functional and timing information) does not translate well for analog functions. In the absence of timing information, the synthesis tool can be manipulated by explicitly placing the custom cells in Verilog and then instructing the tool not to optimize them. The PNR tool can be manipulated using floorplanning, but future work could explore creating false timing information to better control the analog layout. The result is a new toolchain for automating the layout of analog circuits. Some limitations include the inability to directly simulate the RTL Verilog code, a more complicated LVS procedure, and the inability to synthesize the clock generator from Verilog.

CHAPTER 4. A 1-1-1 MASH ADC SYNTHESIZED FROM VERILOG CODE

"Never tell me the odds!"

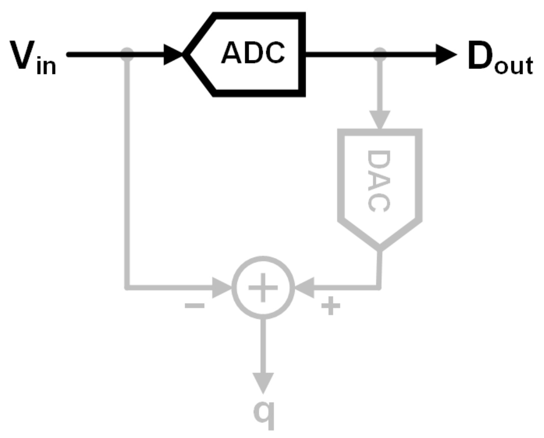
— Han Solo, *The Empire Strikes Back*

This Chapter describes the first demonstration of the proposed analog synthesis process using custom cells. It begins by providing some background information about $\Delta\Sigma$, in order to explain why they are an appropriate architecture to use for synthesized ADCs. It describes the top-level architecture of the implemented ADC, and provides some circuit design details for the custom analog blocks that need to be created. Measurement results are provided for test chips in two different processes (130nm and 65nm nodes) in order to demonstrate the portability of this design technique.

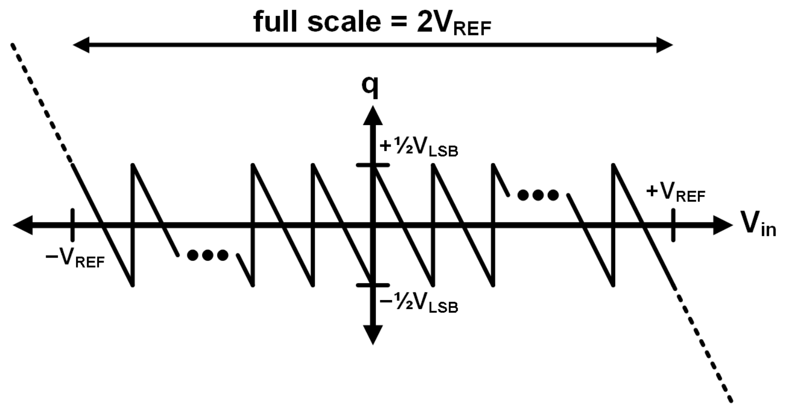
4.1 Background on $\Delta\Sigma$ Modulation

4.1.1 Nyquist-Rate SQNR Limit

Consider the ideal Nyquist-rate ADC shown in Figure 4.1a. The ADC converts an analog input voltage into a digital output, and introduces some quantization error. The transfer function is shown in Figure 4.1b; note that within the



(a)



(b)

Figure 4.1: Ideal ADC quantization error shown in (a) schematic and (b) transfer function.

full-scale range of the ADC, the error is uniformly distributed according to (4.1):

$$q \sim \mathrm{U}\left(-\frac{1}{2}V_{LSB}, +\frac{1}{2}V_{LSB}\right) \tag{4.1}$$

where $V_{LSB} = \frac{2V_{REF}}{2^N}$ for an N-bit ADC.

The theoretical signal-to-quantization noise (SQNR) limit can be calculated from this information. The maximum signal power will be from a full-scale sine wave input, which has amplitude $A = V_{ref}$. The power of this sine wave input is:

$$\sigma_{sig}^2 = \frac{1}{T}\int_0^T A^2 \sin^2(\omega t)dt = \frac{A^2}{2} = \frac{V_{REF}^2}{2} \tag{4.2}$$

Similarly, the power of the quantization noise is:

$$\begin{aligned} \sigma_q^2 &= \frac{1}{V_{LSB}}\int_{-V_{LSB}/2}^{+V_{LSB}/2} q^2 dq = \frac{V_{LSB}^2}{12} \\ &= \left(\frac{4\cdot V_{REF}^2}{2^{2N}}\right)/12 \\ &= \frac{V_{REF}^2}{3\cdot 2^{2N}} \end{aligned} \tag{4.3}$$

The maximum SQNR is the ratio between the results in (4.2) and (4.3):

$$\begin{aligned} SQNR_{max} &= \frac{\sigma_{sig}^2}{\sigma_q^2} \\ &= \frac{V_{REF}^2}{2}\Bigg/\frac{V_{REF}^2}{3\cdot 2^{2N}} \\ &= \frac{3}{2}\cdot 2^{2N} \end{aligned} \tag{4.4}$$

$$\begin{aligned} SQNR_{max}[dB] &= 10\log_{10}(SQNR_{max}) \\ &= 6.02N + 1.76 \end{aligned} \tag{4.5}$$

The final expression in (4.5) is commonly used to determine the effective number of bits (ENOB) of ADCs.

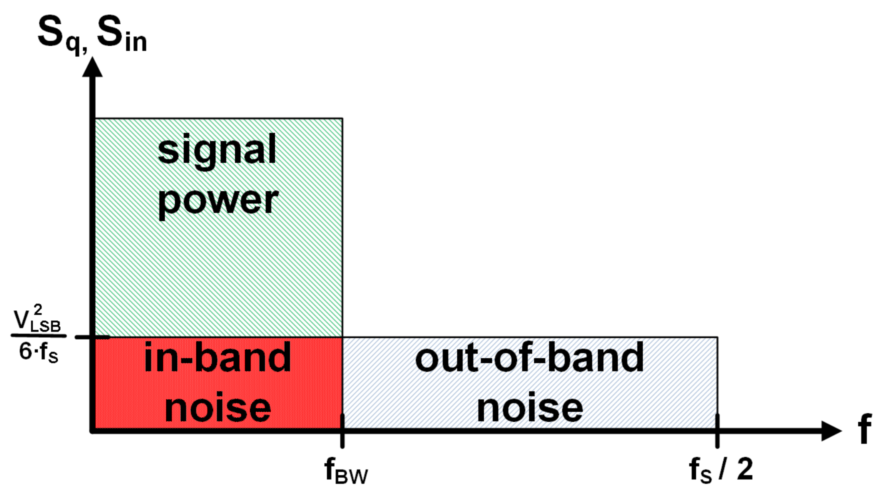


Figure 4.2: Reducing quantization noise with oversampling.

4.1.2 Improving Resolution with Oversampling

The quantization noise calculated in (4.3) has a constant power spectral density (PSD) over the full bandwidth of the ADC. The one-sided PSD is give as:

$$\begin{aligned} S_q &= \sigma_q^2 \cdot \frac{2}{f_S} \\ &= \frac{1}{f_S} \cdot \frac{V_{LSB}^2}{6} \end{aligned} \tag{4.6}$$

As shown in Figure 4.2, one method to improve the SQNR of the converter is to use a signal bandwidth that is less than Nyquist rate. Let the oversampling ratio (OSR) be defined as the ratio between Nyquist rate ($f_S/2$) and the new bandwidth (f_{BW}):

$$OSR = \frac{f_S}{2 \cdot f_{BW}} \tag{4.7}$$

or equivalently

$$f_{BW} = \frac{f_S}{2 \cdot OSR} = \frac{1}{2 \cdot T \cdot OSR} \tag{4.8}$$

The resulting in-band quantization noise is:

$$\begin{aligned} \sigma^2_{q,OSR} &= \int_0^{f_{BW}} S_q df = f_{BW} \cdot \frac{V^2_{LSB}}{6 \cdot f_S} \\ &= \frac{f_S}{2 \cdot OSR} \cdot \frac{V^2_{LSB}}{6 \cdot f_S} \\ &= \frac{\sigma^2_q}{OSR} \end{aligned} \tag{4.9}$$

The reduction of in-band quantization noise translates into an increase in the maximum achievable SQNR:

$$\begin{aligned} SQNR_{max,OSR} &= \frac{V^2_{REF}}{2} \bigg/ \frac{V^2_{REF}}{3 \cdot 2^{2N} \cdot OSR} \\ &= \frac{3}{2} \cdot 2^{2N} \cdot OSR \end{aligned} \tag{4.10}$$

$$\begin{aligned} SQNR_{max,OSR}[dB] &= 10 \log_{10}(SQNR_{max,OSR}) \\ &= 6.02N + 1.76 + 10 \log_{10}(OSR) \end{aligned} \tag{4.11}$$

Oversampling the ADC provides a benefit of 3.01-dB per octave (0.5-bit per octave). While it is useful to have control over this exchange between bandwidth and resolution, the increase in resolution does not match the reduction in bandwidth, so oversampling this ADC comes at a reduction of Figure-of-Merit as well.

$$FoM = \frac{Power}{f_{BW} \cdot 2^{ENOB}} \tag{4.12}$$

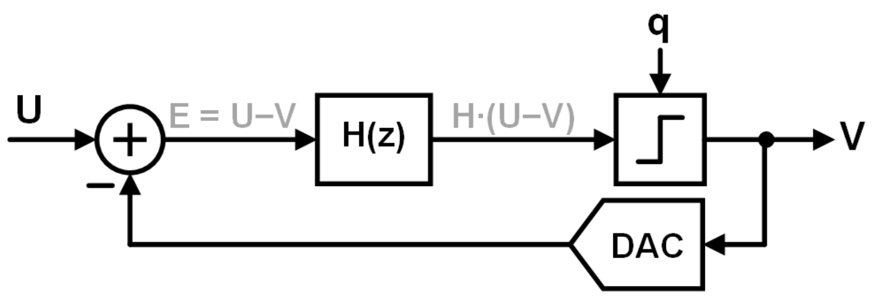


Figure 4.3: A first-order ΔΣ ADC.

4.1.3 First-Order Noise Shaping

The ΔΣ modulator shown in Figure 4.3 makes better use of oversampling than the previous example. The difference (i.e. error) between the input and output is passed through a loop filter and quantized. The loop filter transfer function is selected in order to drive the error towards zero, making the digital output an accurate representation of the analog input. To understand why this structure is so useful, we first derive an expression for the output (V) of the modulator:

$$\begin{aligned} V &= H(U-V)+q \\ V + H\cdot V &= H\cdot U + q \\ V &= \frac{H}{1+H}\cdot U + \frac{1}{1+H}\cdot q \end{aligned} \tag{4.13}$$

$$STF(z) = \frac{H}{1+H}, \qquad NTF(z) = \frac{1}{1+H} \tag{4.14}$$

There are two different transfer functions for the modulator: a signal transfer function (STF) from U to V, and a noise transfer function (NTF) from q to V. Ideally the STF should not alter the input signal, it should result in nothing more than a delay. The ideal NTF should reduce in-band quantization noise by pushing the noise power out to higher out-of-band frequencies. Supposing that the loop filter is an integrator:

$$H = \frac{z^{-1}}{1 - z^{-1}} \tag{4.15}$$

$$STF(z) = \frac{H}{1 + H} = z^{-1} \tag{4.16}$$

$$NTF(z) = \frac{1}{1 + H} = 1 - z^{-1} \tag{4.17}$$

The STF in (4.16) delays the input one clock cycle, but otherwise has no impact on the input signal. The effect of the NTF is less obvious. To understand how the quantization noise is shaped, solve for the squared magnitude of the NTF:

$$\begin{aligned}
|NTF|^2 &= |1 - z^{-1}|^2 \\
&= |1 - e^{-j2\pi fT}|^2 \\
&= |1 - [\cos(-2\pi fT) + j \cdot \sin(-2\pi fT)]\,|^2 \\
&= |1 - [\cos(2\pi fT) - j \cdot \sin(2\pi fT)]\,|^2 \\
&= |1 - \cos(2\pi fT) + j \cdot \sin(2\pi fT)|^2 \\
&= [1 - \cos(2\pi fT)]^2 + [sin(2\pi fT)]^2 \\
&= 1 - 2\cos(2\pi fT) + \cos^2(2\pi fT) + \sin^2(2\pi fT) \\
&= 2 - 2\cos(2\pi fT)
\end{aligned} \tag{4.18}$$

where $T = 1/f_S$. Figure 4.4 shows how this NTF pushes the in-band quantization noise power to higher frequencies. To quantify the improvement to SQNR, the shaped noise is integrated across the bandwidth. The new quantization noise power will be denoted $\sigma^2_{q,L1}$ to indicate that it is for a $\Delta\Sigma$ modulator with first-order loop filtering. Loop filter order will be discussed in more detail in Section

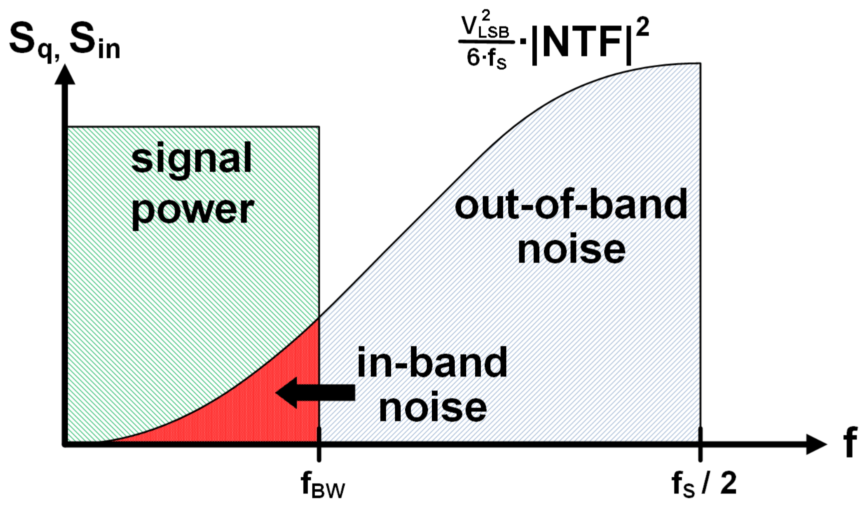


Figure 4.4: Power spectral density with first-order noise shaping.

4.1.5.

$$\begin{aligned}
\sigma_{q,L1}^2 &= \int_0^{f_{BW}} \left[S_q \cdot |NTF|^2\right] df \\
&= \int_0^{f_{BW}} \left[\frac{V_{LSB}^2}{6 f_S}\left(2 - 2\cos(2\pi f T)\right)\right] df \\
&= \frac{V_{LSB}^2}{6 f_S}\left[2f - \frac{1}{\pi T}\sin(2\pi f T)\right]_0^{f_{BW}} \\
&= \frac{V_{LSB}^2}{6 f_S}\left[2 f_{BW} - \frac{f_S}{\pi}\sin(2\pi f_{BW} T)\right]
\end{aligned} \tag{4.19}$$

Substituting using (4.8):

$$\begin{aligned}
\sigma_{q,L1}^2 &= \frac{V_{LSB}^2}{6 f_S}\left[\frac{f_S}{OSR} - \frac{f_S}{\pi}\sin(\frac{\pi}{OSR})\right] \\
&= \frac{V_{LSB}^2}{6}\left[\frac{1}{OSR} - \frac{1}{\pi}\sin(\frac{\pi}{OSR})\right]
\end{aligned} \tag{4.20}$$

Using the Taylor series expansion for the sine function:

$$\begin{aligned}\sigma_{q,L1}^2 &= \frac{V_{LSB}^2}{6}\left[\frac{1}{OSR} - \frac{1}{\pi}\left(\frac{\pi}{OSR} - \frac{\pi^3}{3!\cdot OSR^3} + \frac{\pi^5}{5!\cdot OSR^5} - \cdots\right)\right] \\ &= \frac{2\cdot V_{REF}^2}{3\cdot 2^{2N}}\left[\frac{\pi^2}{3!\cdot OSR^3} - \frac{\pi^4}{5!\cdot OSR^5} + \cdots\right]\end{aligned} \tag{4.21}$$

Using the same maximum signal power from before, the SQNR limit becomes

$$\begin{aligned}SQNR_{max,L1} &= \frac{V_{REF}^2}{2}\bigg/\left\{\frac{2\cdot V_{REF}^2}{3\cdot 2^{2N}}\cdot\left[\frac{\pi^2}{3!\cdot OSR^3} - \frac{\pi^4}{5!\cdot OSR^5} + \cdots\right]\right\} \\ &= \frac{\frac{3}{2}\cdot 2^{2N}}{2\cdot\left[\frac{\pi^2}{3!\cdot OSR^3} - \frac{\pi^4}{5!\cdot OSR^5} + \cdots\right]}\end{aligned} \tag{4.22}$$

$$\begin{aligned}SQNR_{max,L1}[dB] &= 6.02N + 1.76 - 10\log_{10}\left\{2\cdot\left[\frac{\pi^2}{3!\cdot OSR^3} - \frac{\pi^4}{5!\cdot OSR^5} + \cdots\right]\right\} \\ &\approx 6.02N + 1.76 - 10\log_{10}\left(\frac{\pi^2}{3\cdot OSR^3}\right) \\ &= 6.02N + 1.76 + 30\log_{10}(OSR) - \log_{10}\left(\frac{\pi^2}{3}\right)\end{aligned} \tag{4.23}$$

The approximation in (4.23) is very accurate for high OSR; even for $OSR = 2$ it causes an error of only 0.54-dB. Note that the SQNR now improves at a rate of 9.03-dB per octave, as a result of the noise shaping.

4.1.4 Low Distortion Topology

The preceding analysis relies on the assumption of a perfect integrator. In reality, the integrator will suffer from nonidealities including distortion, which causes the integrator output to contain harmonics of its input signal. For the

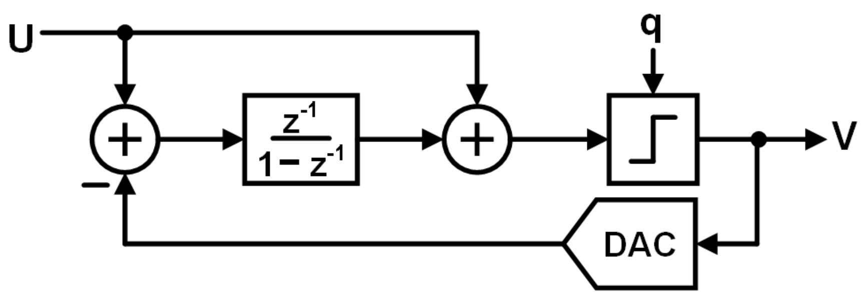


Figure 4.5: A first-order ΔΣ ADC, using low-distortion architecture.

topology studied in Figure 4.3, the integrator processes signal E:

$$\begin{aligned} E &= U - V \\ &= U - \left[z^{-1} \cdot U + (1 - z^{-1}) \cdot q\right] \\ &= (1 - z^{-1}) \cdot U - (1 - z^{-1}) \cdot q \end{aligned} \tag{4.24}$$

The integrator input contains a filtered version of the U that will create distortion. An alternative topology is shown in Figure 4.5, which uses a feed-forward path to avoid sending any residue of U into the integrator. Now the output of the modulator is given by:

$$\begin{aligned} V &= U + H \cdot (U - V) + q \\ (H + 1) \cdot V &= (H + 1) \cdot U + q \\ V &= U + \frac{1}{H + 1} \cdot q \\ V &= U + q \cdot z^{-1} \end{aligned} \tag{4.25}$$

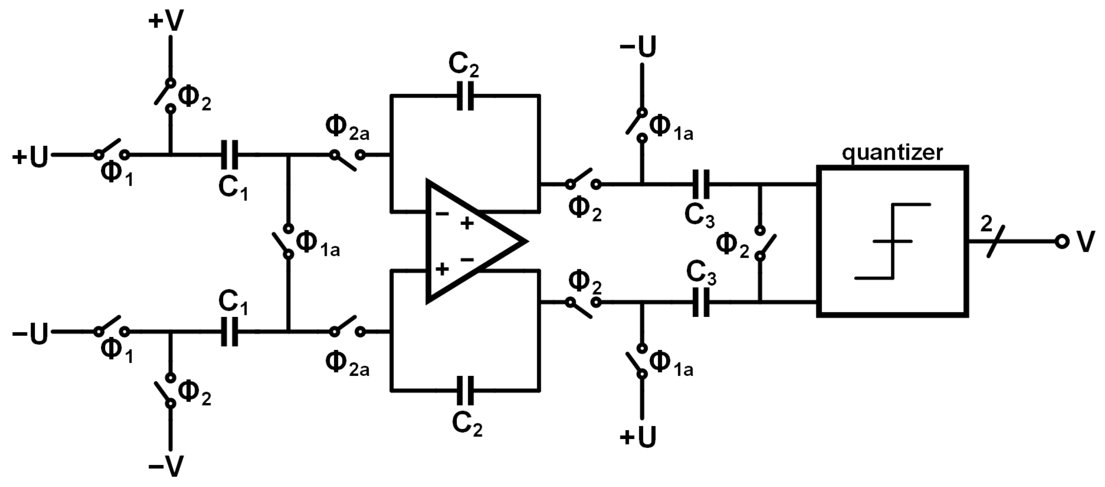
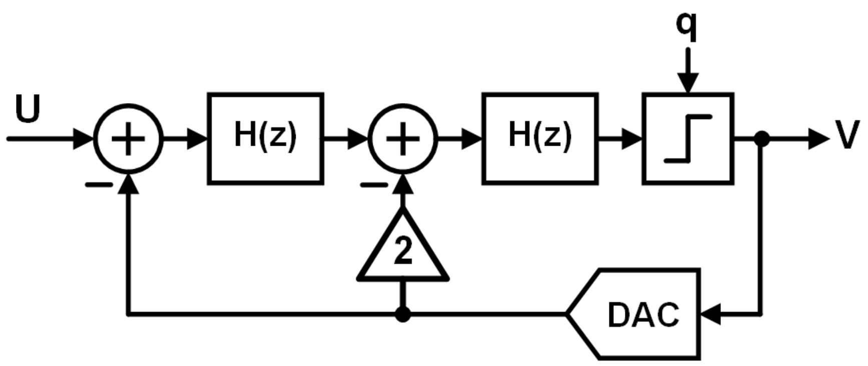


Figure 4.6: Switched capacitor implementation of low-distortion DSM topology.

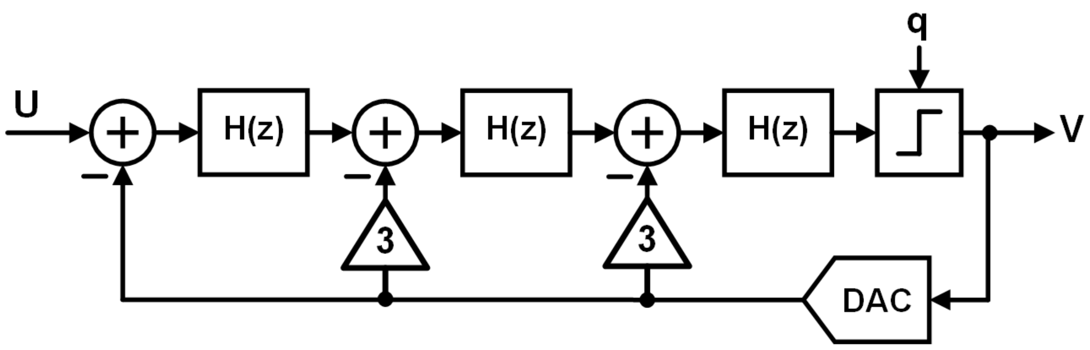
which means that the integrator processes:

$$\begin{aligned} E &= U - V \\ &= U - \left[U + q \cdot z^{-1}\right] \\ &= q \cdot z^{-1} \end{aligned} \tag{4.26}$$

With the integrators processing only quantization noise, no harmonics of U will be generated. Integrator nonlinearity is still a concern, but it is relaxed compared to the previous topology. The switched capacitor implementation of the low-distortion topology is shown in Figure 4.6. The circuit design details for the amplifier, quantizer, and feedback DAC will be provided in Section 4.3.



(a)



(b)

Figure 4.7: DSM modulators with (a) second and (b) third-order noise shaping.

4.1.5 Higher-Order Noise Shaping

More extreme noise shaping can be achieved by increasing the order of the NTF. Examples of second- and third-order noise shaping are shown in Figure 4.7. Analyzing these structures in the z-domain yields the following NTF:

$$NTF_L = \left(1 - z^{-1}\right)^L \tag{4.27}$$

where L is the loop filter order. The theoretical SQNR limit for an order L modulator is derived by repeating the steps in (4.18)-(4.23) for the generalized NTF:

$$\begin{aligned} |NTF_L|^2 &= \left|\left(1 - z^{-1}\right)^L\right|^2 \\ &= \left|1 - z^{-1}\right|^{2L} \\ &= \left[2 - 2\cos(2\pi fT)\right]^L \end{aligned} \tag{4.28}$$

The closed-form solution of this integral is a hypergeometric function, so for this general solution the Taylor series expansion and approximation will be performed earlier.

$$\begin{aligned} |NTF_L|^2 &= \left[2 - 2\left(1 - \frac{(2\pi fT)^2}{2!} + \frac{(2\pi fT)^4}{4!} - \frac{(2\pi fT)^6}{6!} + \cdots\right)\right]^L \\ &= \left[2\left(\frac{(2\pi fT)^2}{2!} - \frac{(2\pi fT)^4}{4!} + \frac{(2\pi fT)^6}{6!} - \cdots\right)\right]^L \\ &\approx (2\pi fT)^{2L} \end{aligned} \tag{4.29}$$

This approximation of the NTF is used to integrate the in-band quantization noise power:

$$\begin{aligned}\sigma_{q,L}^2 &= \int_0^{f_{BW}} \left[S_q \cdot |NTF_L|^2\right] df \\ &= \int_0^{f_{BW}} \left[\frac{V_{LSB}^2}{6f_S} \cdot (2\pi fT)^{2L}\right] df \\ &= \frac{V_{LSB}^2}{6f_S} \left[\frac{f \cdot (2\pi)^{2L} \cdot (fT)^{2L}}{2L+1}\right]_0^{f_{BW}} \\ &= \frac{V_{LSB}^2}{6f_S} \left[\frac{f_{BW} \cdot (2\pi)^{2L} \cdot (f_{BW}T)^{2L}}{2L+1}\right]\end{aligned} \tag{4.30}$$

Substituting using (4.8):

$$\begin{aligned}\sigma_{q,L}^2 &= \frac{\left(\frac{V_{LSB}^2}{6}\right) \cdot (2\pi)^{2L} \cdot \left(\frac{1}{2 \cdot OSR}\right)^{2L+1}}{2L+1} \\ &= \frac{\left(\frac{2 \cdot V_{REF}^2}{3 \cdot 2^{2N}}\right) \cdot (2\pi)^{2L} \cdot \left(\frac{1}{2 \cdot OSR}\right)^{2L+1}}{2L+1}\end{aligned} \tag{4.31}$$

Finally, the SQNR is calculated as the ratio of the maximum signal power (σ_{sig}^2) to newly derived quantization noise power:

$$\begin{aligned}SQNR_{max,L} &= \frac{V_{REF}^2}{2} \Bigg/ \left\{\frac{\left(\frac{2 \cdot V_{REF}^2}{3 \cdot 2^{2N}}\right) \cdot (2\pi)^{2L} \cdot \left(\frac{1}{2 \cdot OSR}\right)^{2L+1}}{2L+1}\right\} \\ &= \left(\frac{3}{2} \cdot 2^{2N}\right) \cdot \frac{(2L+1)(2 \cdot OSR)^{2L+1}}{2(2\pi)^{2L}} \\ &= \left(\frac{3}{2} \cdot 2^{2N}\right) \cdot \frac{(2L+1)(OSR)^{2L+1}}{\pi^{2L}}\end{aligned} \tag{4.32}$$

$$\begin{aligned} SQNR_{max,L}[dB] &= 10\log_{10}\left(2^{2N}\right) + 10\log_{10}\left(\frac{3}{2}\right) \\ &\quad + 10\log_{10}\left(OSR^{2L+1}\right) - 10\log_{10}\left(\frac{\pi^{2L}}{2L+1}\right) \\ &= 6.02N + 1.76 + (20L+10)\log_{10}(OSR) - 10\log_{10}\left(\frac{\pi^{2L}}{2L+1}\right) \end{aligned} \tag{4.33}$$

The result in (4.33) is frequently used to determine the SQNR limit of a $\Delta\Sigma$ ADC given the loop filter order (L), OSR, and quantizer resolution (N). Examples of higher-order noise shaping are shown in Figure 4.8. For first- and second-order modulators, the stability of the loop is guaranteed [40]. Unfortunately, for third-order and higher the stability of the loop becomes a problem (as demonstrated in root-locus plots). It is possible for the integrator outputs to saturate and disrupt the normal operation. There exist advanced techniques for solving this [41, 42, 43], but the increased complexity is counter-productive for our goal of a simple and synthesizable design.

A common method for achieving higher-order noise shaping without stability concerns is the Multi-stAge noise-SHaping (MASH) modulator [44, 45]. Shown in Figure 4.9, this cascades multiple $\Delta\Sigma$ stages such that the latter stages process the error residue from the preceding stage. Each stage output is filtered and added together to obtain:

$$D = F_1 \cdot V_1 + F_2 \cdot V_2 + \cdots + F_n \cdot V_n \tag{4.34}$$

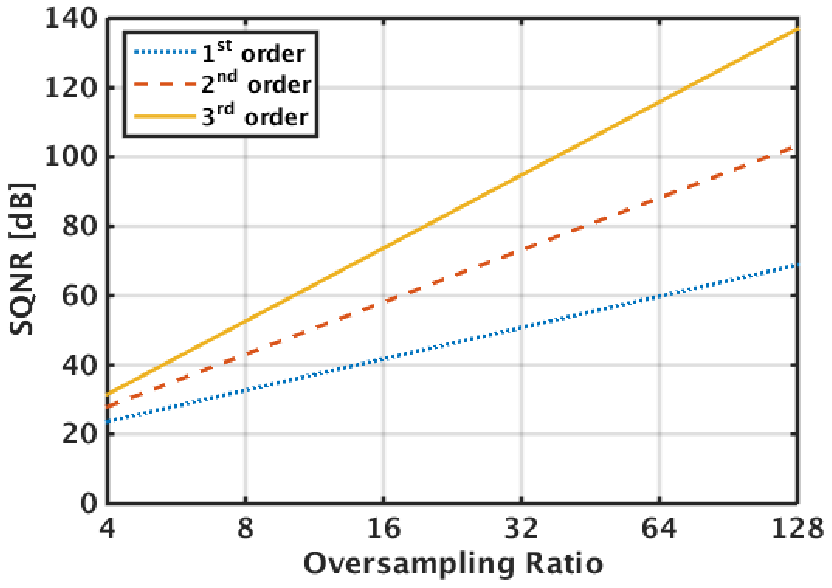


Figure 4.8: Theoretical SQNR limit of 1.5-bit modulators with various orders.

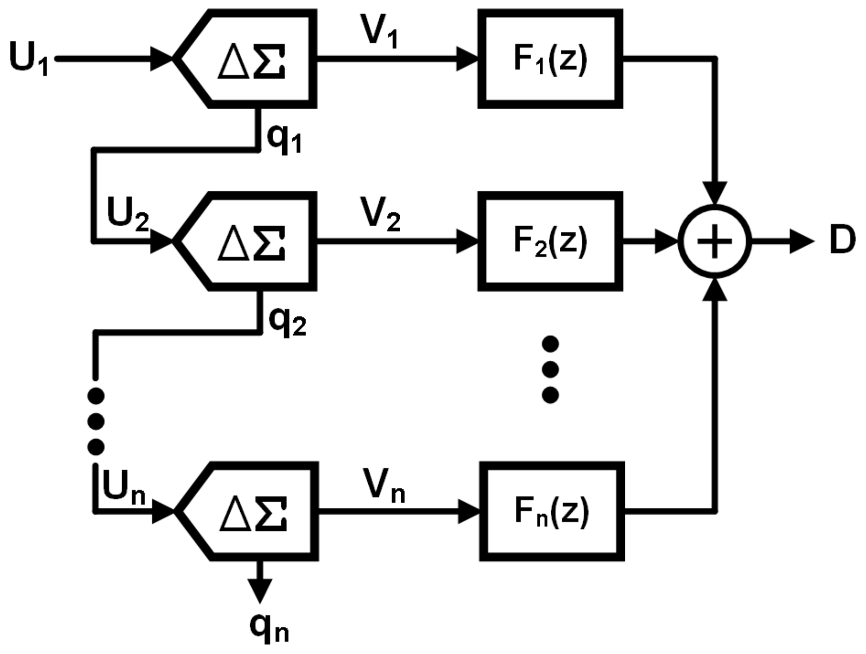


Figure 4.9: An n-stage MASH ADC.

Each V_i may be expressed in terms of the stage's STF and NTF:

$$\begin{aligned} D =& F_1[STF_1 \cdot U_1 + NTF_1 \cdot q_1] + F_2[STF_2 \cdot U_2 + NTF_2 \cdot q_2] \\ & + \cdots + F_n[STF_n \cdot U_n + NTF_n \cdot q_n] \\ =& U_1\left[F_1 \cdot STF_1\right] + q_1\left[F_1 \cdot NTF_1 + F_2 \cdot STF_2\right] \\ & + q_2\left[F_2 \cdot NTF_2 + F_3 \cdot STF_3\right] + \cdots + q_n\left[F_n \cdot NTF_n\right] \end{aligned} \tag{4.35}$$

In order to cancel the quantization noise from stages 1 through $n-1$, the filter is designed such that:

$$\begin{aligned} F_1 \cdot NTF_1 + F_2 \cdot STF_2 = 0 \quad &\Rightarrow \quad F_2 = -F_1 \cdot \frac{NTF_1}{STF_2} \\ F_2 \cdot NTF_2 + F_3 \cdot STF_3 = 0 \quad &\Rightarrow \quad F_3 = -F_2 \cdot \frac{NTF_2}{STF_3} \\ &\cdots \\ F_{n-1} \cdot NTF_{n-1} + F_n \cdot STF_n = 0 \quad &\Rightarrow \quad F_n = -F_{n-1} \cdot \frac{NTF_{n-1}}{STF_n} \end{aligned} \tag{4.36}$$

This simplifies the expression for the output signal:

$$\begin{aligned} D &= U_1\left[F_1 \cdot STF_1\right] + q_n\left[F_{n-1} \cdot \frac{-NTF_{n-1}}{STF_n} \cdot NTF_n\right] \\ &= U_1\left[F_1 \cdot STF_1\right] + q_n\left[F_{n-2} \cdot \frac{-NTF_{n-2}}{STF_{n-1}} \cdot \frac{-NTF_{n-1}}{STF_n} \cdot NTF_n\right] \\ &= U_1\left[F_1 \cdot STF_1\right] + q_n\left[(-1)^{n-1} \cdot F_1\left(\frac{NTF_1 \cdot NTF_2 \cdot NTF_3 \cdots NTF_n}{STF_2 \cdot STF_3 \cdots STF_n}\right)\right] \end{aligned} \tag{4.37}$$

Thus the MASH has signal and noise transfer functions equivalent to:

$$\begin{aligned} \widehat{STF} &= F_1 \cdot STF_1 \\ \widehat{NTF} &= (-1)^{n-1} \cdot F_1\left(\frac{NTF_1 \cdot NTF_2 \cdot NTF_3 \cdots NTF_n}{STF_2 \cdot STF_3 \cdots STF_n}\right) \end{aligned} \tag{4.38}$$

Assuming that each individual stage STF (and F_1, because it directly processes the input signal) have been designed with unity magnitude:

$$\begin{aligned} \left|\widehat{STF}\right| &= 1 \\ \left|\widehat{NTF}\right| &= NTF_1 \cdot NTF_2 \cdot NTF_3 \cdots NTF_n \end{aligned} \tag{4.39}$$

The result in (4.39) indicates that the MASH achieves an order of noise shaping equivalent to the sum of the orders of the individual stages. This allows a high-order $\Delta\Sigma$ ADC to be designed using only 1st- and 2nd-order loop filters, and thus removing the concerns about stability.

4.1.6 Finite Integrator Gain and Gain Compression

Any nonidealities in the circuit's NTF will reduce the performance below the SQNR limit expressed in (4.33). First, the gain of the opamps used in the integrators will be finite, making the integrator transfer function look like:

$$H(z) = \frac{\left(\frac{A}{A+2}\right) z^{-1}}{1 - \left(\frac{A+1}{A+2}\right) z^{-1}} \tag{4.40}$$

The derivation of (4.40) is included in Appendix A. This equation is commonly approximated as $H(z) = \frac{z^{-1}}{1-\left(1-\frac{1}{A}\right)z^{-1}}$, but for the extremely low gain opamps used in this work the approximation is not valid. The effect of the finite gain is that the in-band quantization noise cannot be fully suppressed, such that the low-frequency noise is no longer shaped as shown in Figure 4.10. This creates a cut-off point at which the modulator order effectively drops to zero-order, and SQNR is only improved by 3.01dB per decade again.

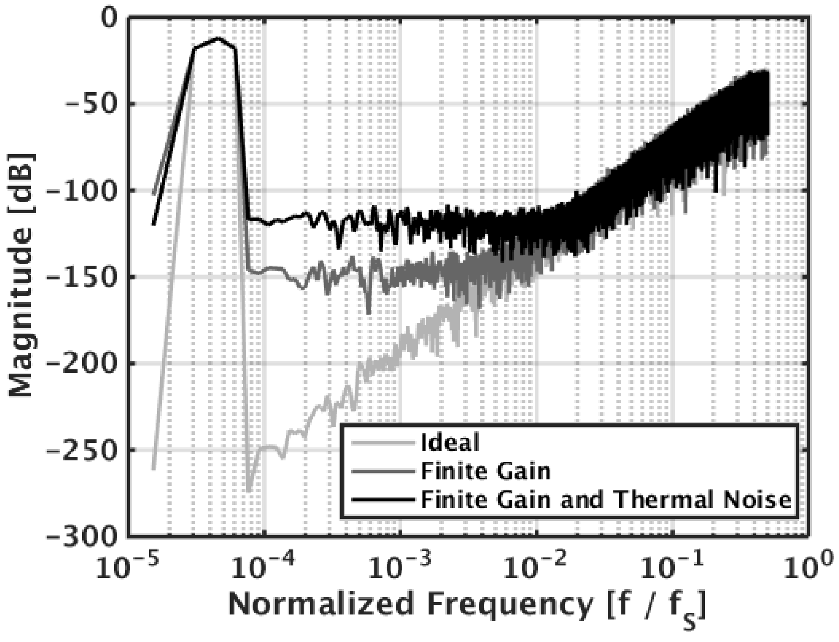


Figure 4.10: Simulation results showing effect of finite gain and thermal noise.

While the preceding discussions have focused on the SQNR limit due to quantization noise, another nonideality is the thermal noise associated with sampling capacitors, switches, and opamps. Thermal noise is shaped just like quantization noise, and the effect is just an increase in the noise floor as shown in Figure 4.10. A more detailed discussion of thermal noise requirements will be provided in Section 4.2.2.

The final nonideality discussed here is opamp gain compression. In addition to being finite, the opamp gain tends to decrease at larger output swings. This signal-dependent DC gain causes distortion as shown in Figure 4.11. For good spurious-free dynamic range (SFDR) the designer aims to minimize the compression of the gain at large output swings. Section 4.3.1 will simulate this performance for the amplifiers used in this work. Figure 4.12 shows the sensitivity of the per-

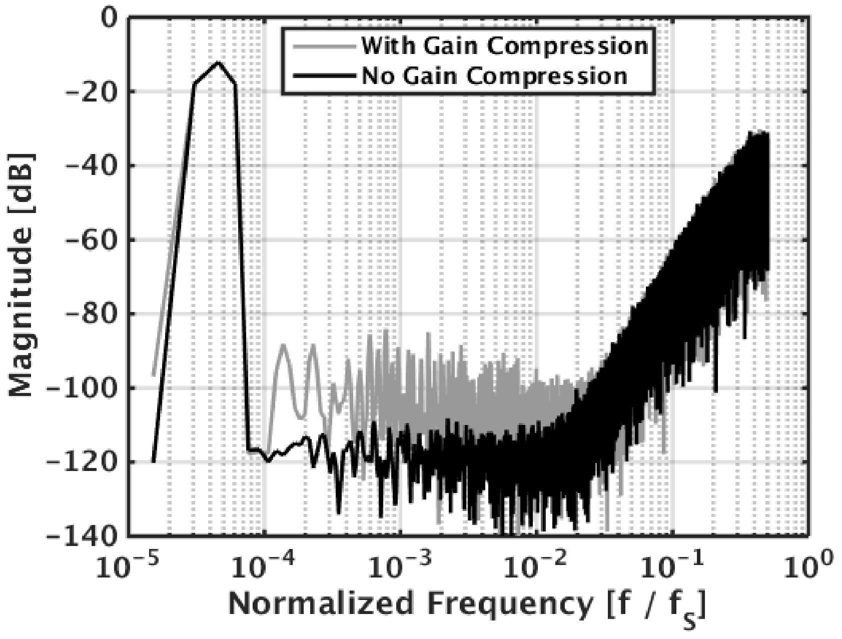


Figure 4.11: Simulation results showing effect of gain compression in integrator.

formance to matching the gain used in filtering the digital output to the true gain of the opamp. The addition of thermal noise reduces the peak SNDR as expected, but doesn't change the estimated gain value at which SNDR is maximized. However, including gain compression not only reduces peak SNDR due to distortion, but also shifts the point at which it occurs to a lower value. This can be viewed as a weighted-averaging of the nominal DC gain values with the compressed DC gain values, across the output swing of the integrators.

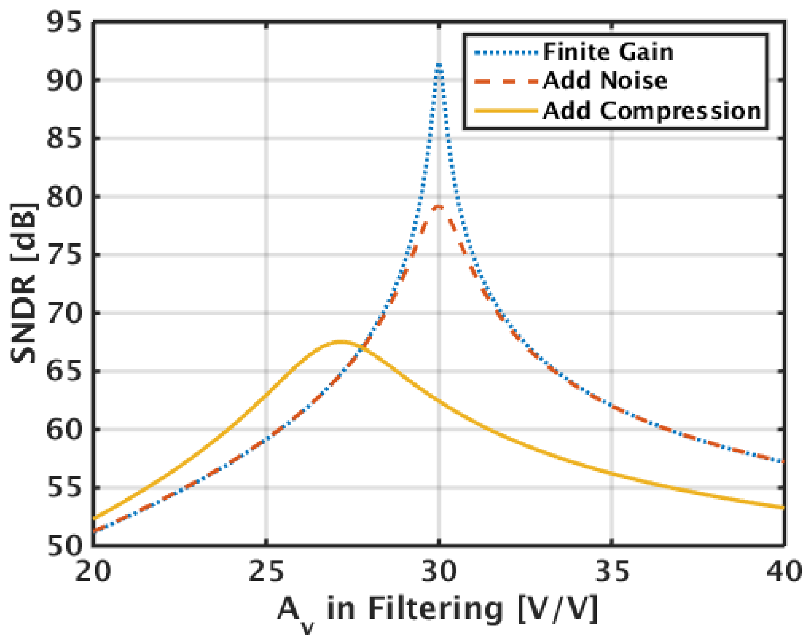


Figure 4.12: Simulation results showing sensitivity of finite-gain calibration in 1-1-1 MASH, for a nominally 30dB amplifier.

4.2 Proposed System Architecture

4.2.1 Top-Level Design

The $\Delta\Sigma$ ADC seems an appropriate architecture for synthesized ADC design. It trades increased digital complexity and higher operating speeds for relaxed analog matching requirements [40], making use of the existing digital standard cell library. The relaxed analog matching compensates for the rudimentary analog components and mismatch that is expected from the automated layout procedure. In particular, the MASH ADC composed of single-order stages is appealing because of the guaranteed stability and the simple trade-off between SQNR and power/area. Using a three-level quantizer provides N = 1.5-bit resolution while allowing the feedback DAC to switch among $\pm V_{REF}$ and V_{CM} with no need for a reference ladder. This allows for a very simple voltage-switching DAC that will be inherently linear in a differential structure.

In order to determine how many 1st-order DSM stages to cascade, Figure 4.13 plots the maximum SQNR achieved by different orders of MASH ADCs. The simulations use a 30dB opamp (a single intrinsic $g_m r_o$ that we may expect from our rudimentary amplifier) and are perfectly calibrated; no thermal noise or gain compression is considered. The SQNR should be significantly high than the target resolution, to allow for degradations from these other nonidealities. For a target of 62dB-SNDR (10 ENOB), designing with SQNR $>$ 70dB ensures that accuracy limitations will be from the performance of the rudimentary cells and from routing mismatch, rather than being a fundamental limitation of the architecture. The first- and second-order MASH in Figure 4.13 are rolling off towards 3.01dB / octave by the time that they reach 70dB-SQNR, showing that the finite gain beginning to dominate the results. On the other hand, for $L \geq 3$ the ideal noise shaping is maintained well past 70dB-SQNR.

The 1-1-1 (third-order) MASH ADC is chosen to be implemented in this work. Even with single-stage opamp gain, it achieves sufficiently high SQNR to ensure that the SNDR will be limited by sub-block performance and automated routing, rather than being an architectural limitation. While fourth-order or higher shows increased benefit in top-level simulation, it is possible that in implementation the nonideal effects would cause the additional stages to provide negligible benefit. Third-order is deemed sufficient for these proof-of-concept ADCs.

A top-level block diagram of the implemented ADC is shown in Figure 4.14. The low-distortion architecture described in Section 4.1.4 is used in all three stages; while the latter stages should be processing only white noise from quantization

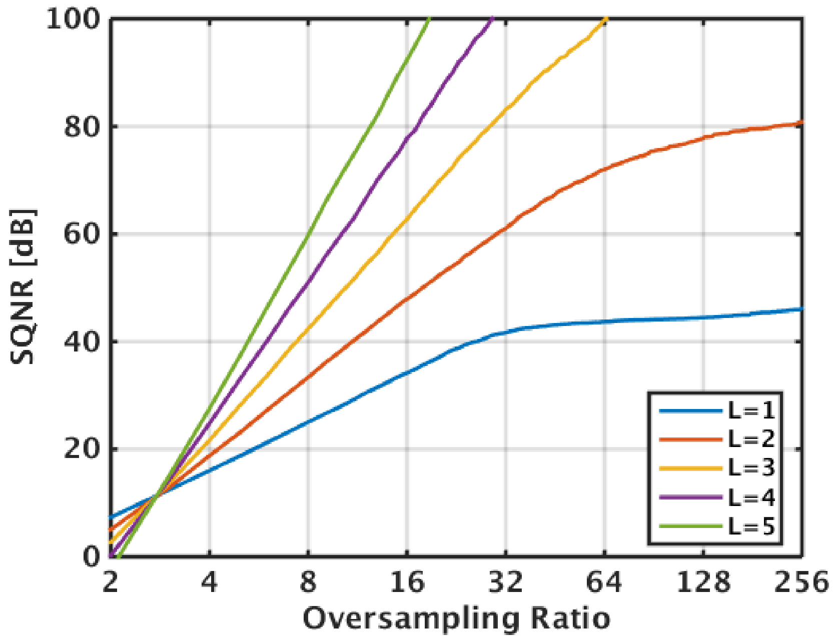


Figure 4.13: Maximum SQNR for different MASH orders using 30dB opamps (perfectly calibrated).

error (making the low-distortion topology unnecessary), it is convenient to use the same Verilog module for all stages. The error residue passed to the next MASH stage is:

$$\begin{aligned}
U_{n+1} &= H \cdot (U_n - V_n) \\
&= H \cdot \left\{ U_n - \left[U_n + \left(\frac{1}{H+1} \right) \cdot q_n \right] \right\} \\
&= -\left(\frac{H}{H+1} \right) \cdot q_n \\
&= -q_n \cdot \left[\frac{\left(\frac{A}{A+2}\right) \cdot z^{-1}}{1 - \left(\frac{1}{A+2}\right) \cdot z^{-1}} \right] \approx -q_n \cdot z^{-1}
\end{aligned} \tag{4.41}$$

Since the residue is a delayed version of the quantization noise, the analysis from Section 4.1.5 may be used with only a minor modification to the filter transfer

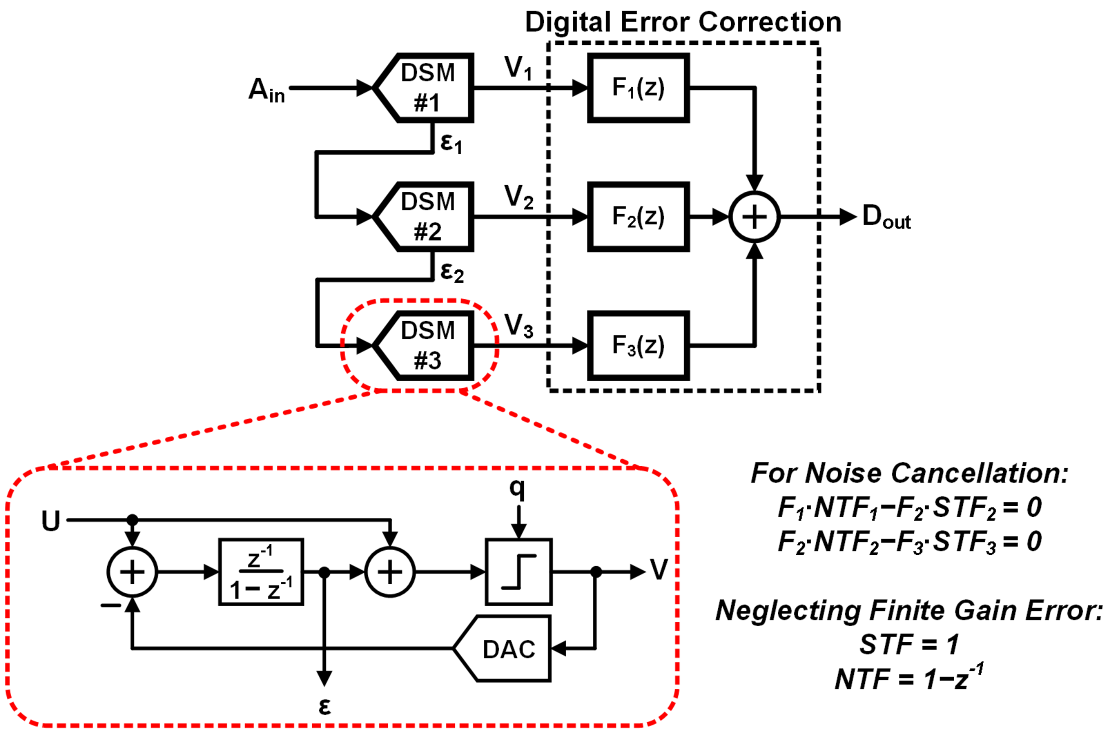


Figure 4.14: Top-level architecture of 1-1-1 MASH ADC.

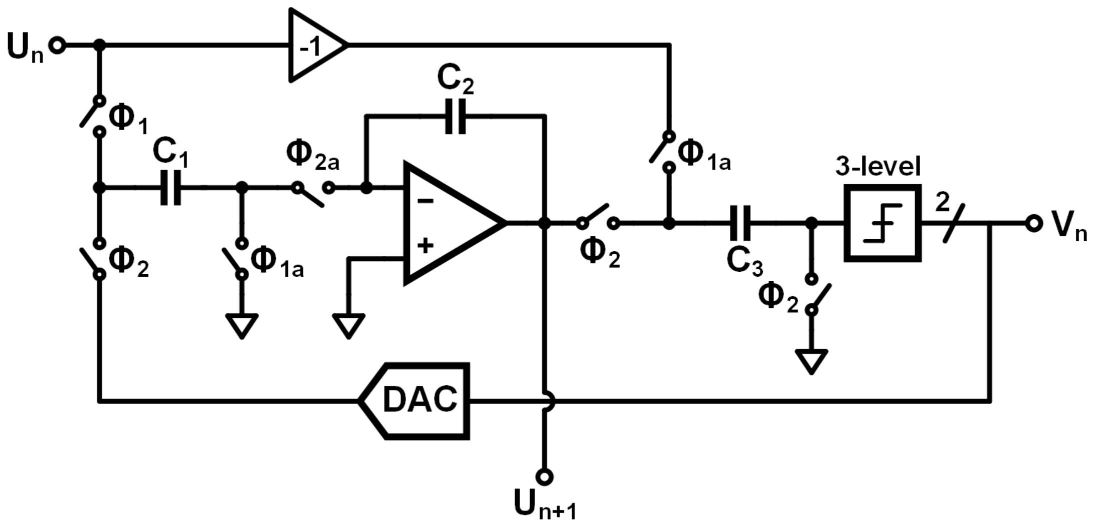


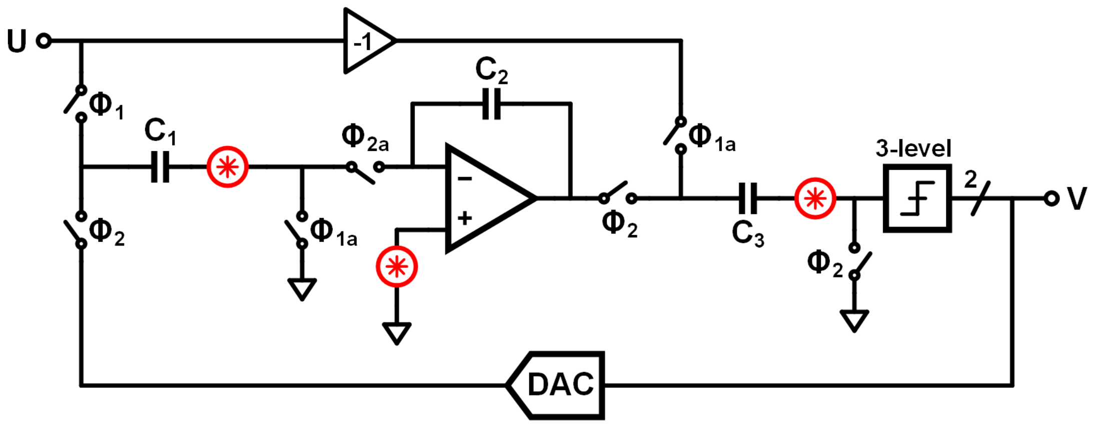
Figure 4.15: Switched-capacitor implementation of first-order DSM (shown single-ended for simplicity).

functions F_i. In this work the filtering and calibration was applied off-chip, but could be included in Verilog and synthesized in future implementations.

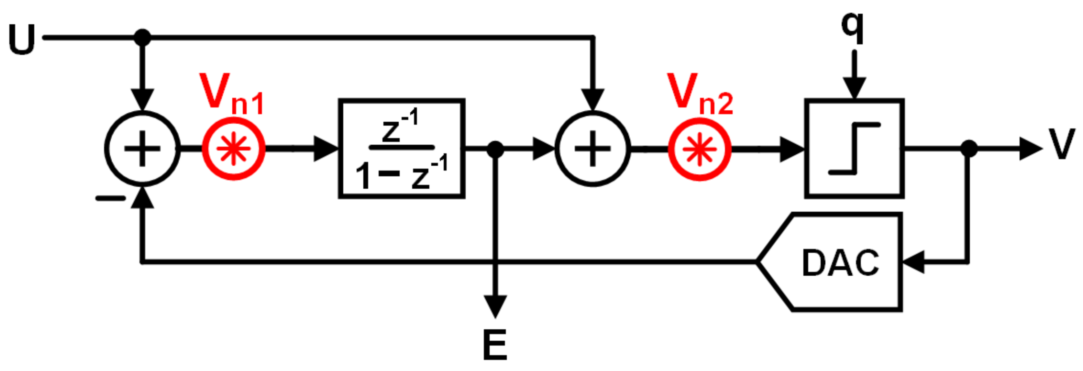
The switched-capacitor implementation of each individual stage of the $\Delta\Sigma$ modulator ADC is shown in Figure 4.15. The switched-capacitor adders are easily constructed from the transmission gate and unit capacitor in the custom library.

4.2.2 Noise Sources

Figure 4.16a illustrates sources of noise within each $\Delta\Sigma$ stage: kT/C switching noise for each of the switch-capacitor adders, and input-referred opamp noise (including thermal noise and flicker noise) for the integrator. Figure 4.16b shows the effect of these noise sources in the block diagram, combining the first switching noise source and the input-referred opamp noise into V_{n1}. Note that the location of



(a)



(b)

Figure 4.16: Location of noise sources (a) in the first-order DSM, and (b) an equivalent representation combining sources.

the second switching noise source (V_{n2}) makes it indistinguishable from quantization noise; a benefit of the MASH structure is that V_{n2} is canceled by the following stage for all but the final stage. Four total noise sources remain: those at the input of the integrator ($V_{n1,1}$, $V_{n1,2}$, and $V_{n1,3}$) and the second switched-capacitor adder noise source in the final stage ($V_{n2,3}$).

4.2.3 Capacitor Sizing

The thermal noise PSD of a switched-capacitor branch is given by:

$$S_C = \frac{2 \cdot 2}{2} \frac{kT}{C} = \frac{2kT}{C} \tag{4.42}$$

where the factors of 2 come from the capacitor being switched twice per sampling period and from the differential architecture, and the factor of $\frac{1}{2}$ coming from the sampling capacitance effectively being $2C$ for a differential structure [40]. For the proposed 1-1-1 MASH, the contributions of all capacitors except C_1 in the first stage are noise-shaped. Therefore the kT/C noise from C_1 in stage one will dominate. The remainder of this Section will derive how to size C_1 for the target resolution for the 130nm used in this work.

Given the supply voltage of 1.2V, assume a -3.01dBFS input in the interest of keeping distortion low. The corresponding voltage amplitude is:

$$A = (1.2) \cdot 10^{-3.01/20} = 0.8486 \; [V] \tag{4.43}$$

for an input signal power of:

$$\sigma_{sig}^2 = \frac{A^2}{2} = 0.36 \; [V^2] \tag{4.44}$$

For a target resolution of 10 ENOB, the total noise power must be 61.96dB below the signal power, at roughly -65dBFS. This corresponds to noise power of:

$$\sigma_n^2 = \frac{0.36}{10^{65/10}} = 1.138 \cdot 10^{-7} \; [V^2] \tag{4.45}$$

Previously, it was decided to operate with a theoretical SQNR significantly above 10-bit resolution. For the 3rd-order DSM, Figure 4.13 indicates that with an OSR of 20 the modulator achieves 70dB SQNR, such that it will not limit the overall resolution. This analysis will proceed by providing the entire noise budget to the kT/C sampling noise, to arrive at the bare minimum C_1 (assuming the contribution of opamp noise and other extrinsic effects are zero) for 10-bit SNR. Then C_1 will be sized significantly greater than this minimum value, to remove kT/C noise as the limiting factor and to shift the noise requirements to the opamp and extrinsic effects.

If C_1 is allocated the entire noise budget:

$$\begin{aligned} C_{1,min} &= \frac{2kT}{\sigma_n^2}\Big/ OSR \\ &= \frac{2 \cdot 4.142 \cdot 10^{-21}}{1.138 \cdot 10^{-7}}\Big/ 20 = 3.64\ [fF] \end{aligned} \tag{4.46}$$

Instead, using size $C_1 = 100fF$:

$$\begin{aligned} \sigma_{n,C1}^2 &= \frac{2kT}{C_1}\Big/ OSR \\ &= \frac{2 \cdot 4.142 \cdot 10^{-21}}{100 \cdot 10^{-15}}\Big/ 20 = 4.14 \cdot 10^{-9}\ [V^2] \end{aligned} \tag{4.47}$$

which corresponds to an SNR of:

$$\begin{aligned} SNR &= 10 \cdot \log_{10}\left(\frac{\sigma_{sig}^2}{\sigma_{n,C1}^2}\right) \\ &= 10 \cdot \log_{10}\left(\frac{0.36}{4.14 \cdot 10^{-9}}\right) = 79.4\ [dB] \end{aligned} \tag{4.48}$$

The 27.5x increase in the capacitor sizing improves the SNR by $\log_4(27.5) = 2.39$ bits. For the proof-of-concept ADC, this is useful because kT/C switching noise will not be the limiting factor in the performance. Instead, the accuracy requirements will be shifted onto the opamp design and matching within the automated layout, testing the capabilities of the proposed design toolchain.

4.3 Minimalist Sub-Block Design

In this section, the transistor-level design of the opamp and quantizer are described. The remainder of the circuit is composed only of transmission gates, unit capacitors, and standard digital library cells.

4.3.1 Operational Amplifier

Figure 4.17a shows a very simple differential opamp topology, consisting of a differential pair with diode-connected loads. Using current-source connected loads would provide higher gain, but at the expense of requiring another bias voltage. The diode-connected implementation requires only a single bias voltage, in the interest of reducing analog complexity. The small-signal gain of the opamp is given in (4.49), and is approximated by the ratio of transconductances for the input and load transistors. For the diode-connected loads, there is a direct trade-off between output voltage swing, input common-mode (CM) range, and voltage gain [46].

$$A_v = \frac{g_{m1}}{g_{m3} + \frac{1}{r_{o1}||r_{o3}}} \approx \frac{g_{m1}}{g_{m3}} \tag{4.49}$$

Consider the effects of adding cross-coupled loads as shown in Figure 4.17b. First, the cross-coupled path appears to be a negative transconductance, which

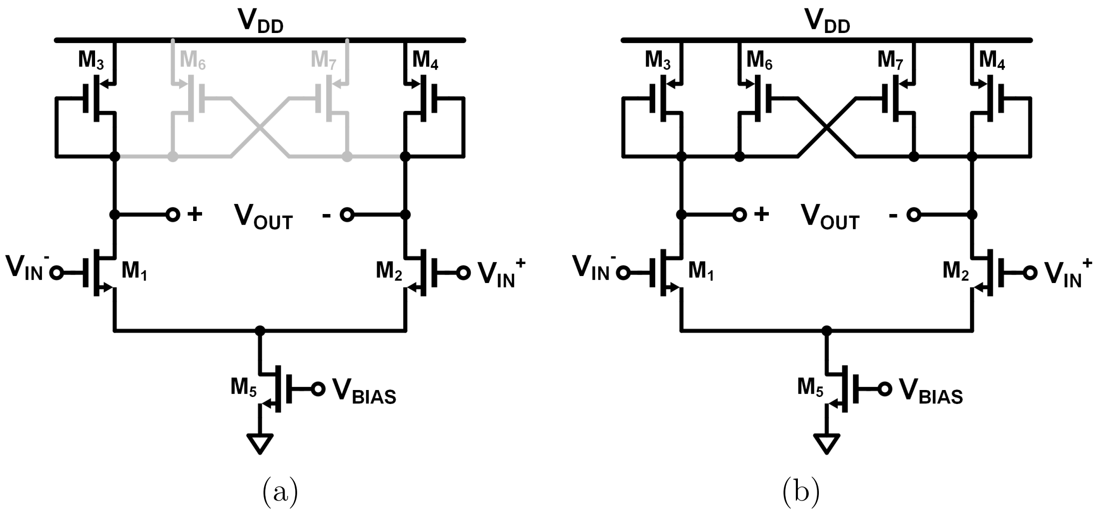
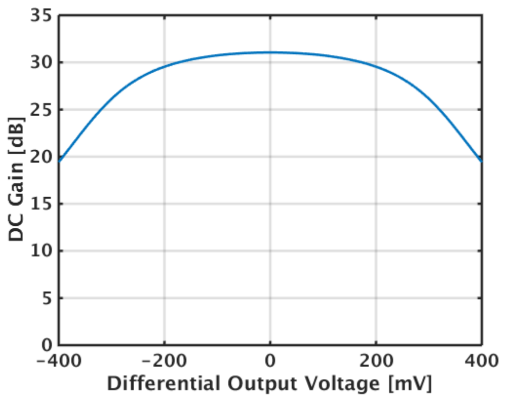


Figure 4.17: Minimalist single stage opamp design using (a) basic differential pair and (b) additional cross-couple active load.

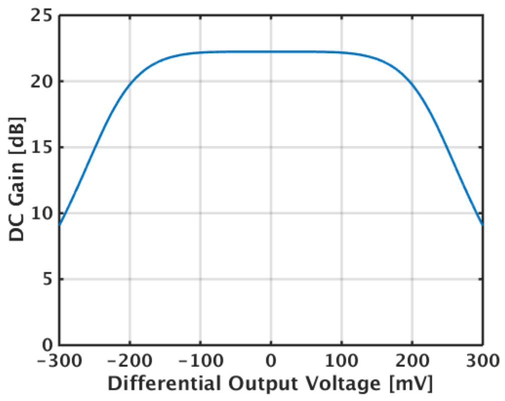
can be used to cancel g_{m3} and increase the voltage gain. Ideally, equally-sized M_3 and M_6 will perfectly cancel each other and the gain will be determined by the parallel combination of output resistances with the input transconductance g_{m1}. If there is sufficient mismatch such that $g_{m6} > g_{m3} + \frac{1}{r_{o1}||r_{o3}||r_{o6}}$, then the voltage gain could flip polarity. Therefore it may be necessary to size the diode-tied M_3 slightly larger than M_6, depending on the expected level of mismatch.

$$A_v = \frac{g_{m1}}{g_{m3} - g_{m6} + \frac{1}{r_{o1}||r_{o3}||r_{o6}}} \approx g_{m1}[r_{o1}||r_{o3}||r_{o6}] \tag{4.50}$$

The DC gain across output swing is plotted in Figure 4.18. The same topology was implemented in 130nm and 65nm, achieving a peak DC gain of 31.1 dB and 22.2 dB, respectively. The gain compression is also evident in the Figure, illustrating a trade-off between integrator output swing and distortion. This is portrayed in a different way in Figure 4.19, showing the harmonic distortion introduced by the amplifiers' gain dependency upon output swing. Note that because



(a)



(b)

Figure 4.18: DC gain of the simple opamp as a function of output swing, for (a) 130nm and (b) 65nm processes.

the amplifier noise and distortion are suppressed by the modulator loop, it does not appear directly at the output [47].

The opamp noise PSD and integrated noise voltage are plotted for the two amplifiers in Figures 4.20 and 4.21. For the 130nm version, even at an ambitious 1GHz the integrated opamp noise ($\sigma_{n,opamp}$) is only 242 μV, falling 67.9dB below the -3dB input signal. Similarly, the 65nm opamp noise voltage is 194 μV which falls (given the reduction in supply voltage to 1V) 68.2dB below the input signal. At more reasonable sample rates the contribution of opamp noise is even further below the target 10-bit SNDR.

4.3.2 Three-Level Quantizer Design

In order to construct a 3-level quantizer using only a single custom comparator cell and no reference ladder, the comparator offsets are instead embedded using switched-capacitors as shown in Figure 4.22. The comparator is implemented as a

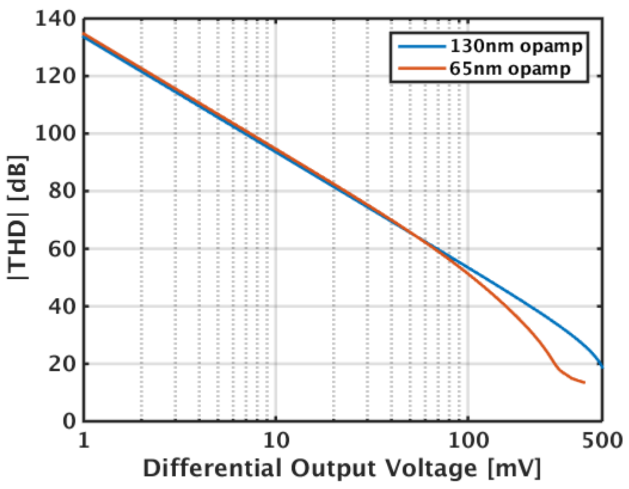
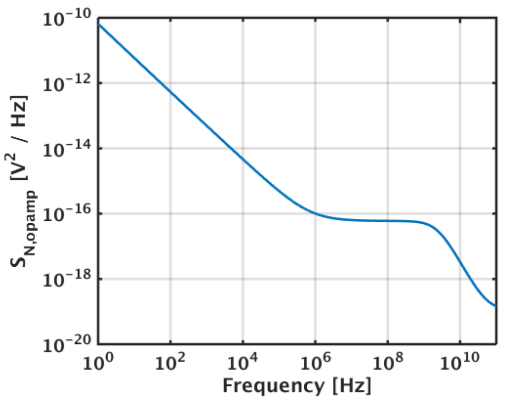
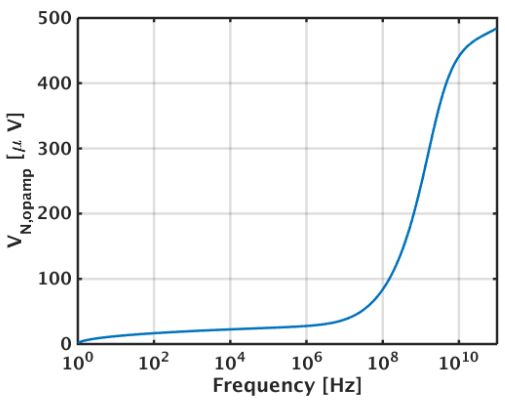


Figure 4.19: Harmonic distortion of the opamp transfer functions across output swing.

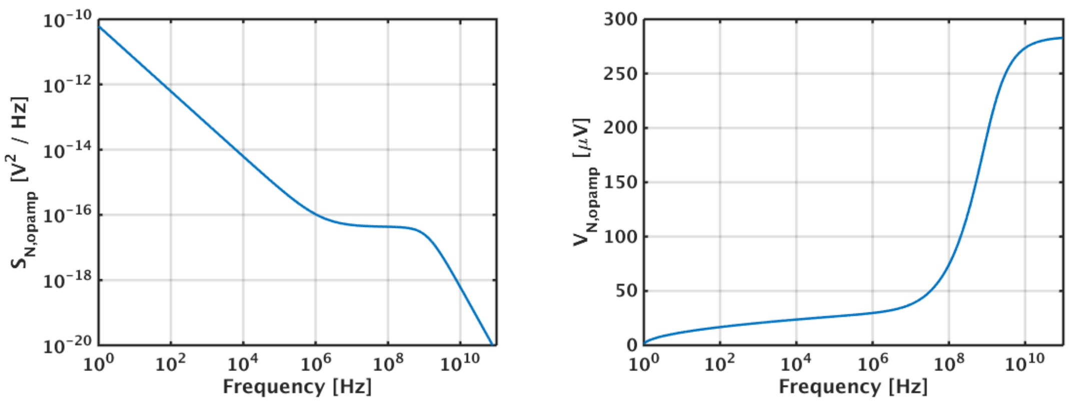


(a)



(b)

Figure 4.20: Opamp performance in 130nm process, showing (a) noise PSD and (b) integrated noise voltage.



(a) (b)

Figure 4.21: Opamp performance in 65nm process, showing (a) noise PSD and (b) integrated noise voltage.

Strong-Arm latch [48].

4.3.3 Feedback DAC Design

The schematic for the feedback DAC is shown in Figure 4.23. It is an extremely simple 3-level voltage-switching DAC. It is included here only in the interest of completeness, being that it is synthesized from custom analog cells.

4.3.4 Final Simulation Results

The final pre- and post-layout extraction simulations in the 130nm process are shown in Figure 4.24, including transient noise. The extracted results show strong second-order distortion, due to coupling through parasitic capacitances. Extensive simulation determined that a single parasitic capacitance (shown in Figure

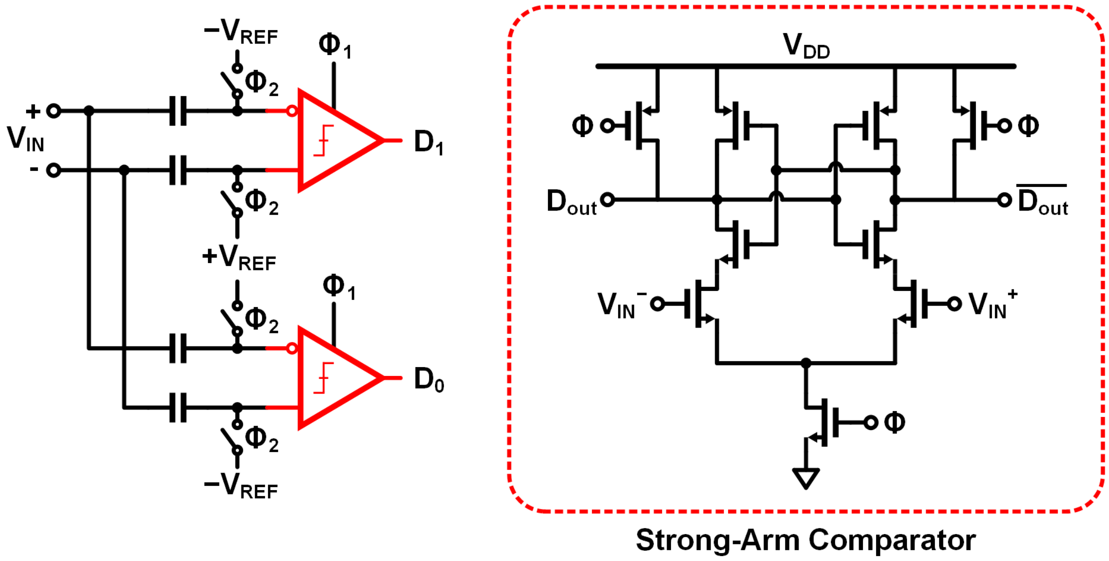


Figure 4.22: Switched-capacitor implementation of three-level quantizer, using a Strong-Arm latch as the quantizer.

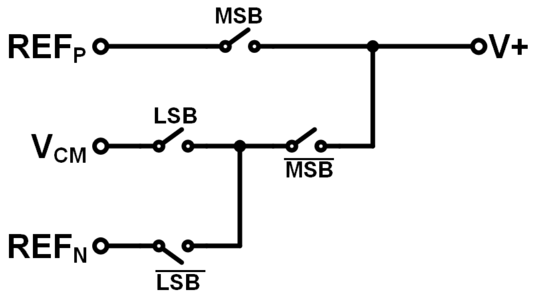


Figure 4.23: Voltage-switched three-level DAC (shown single-ended for simplicity).

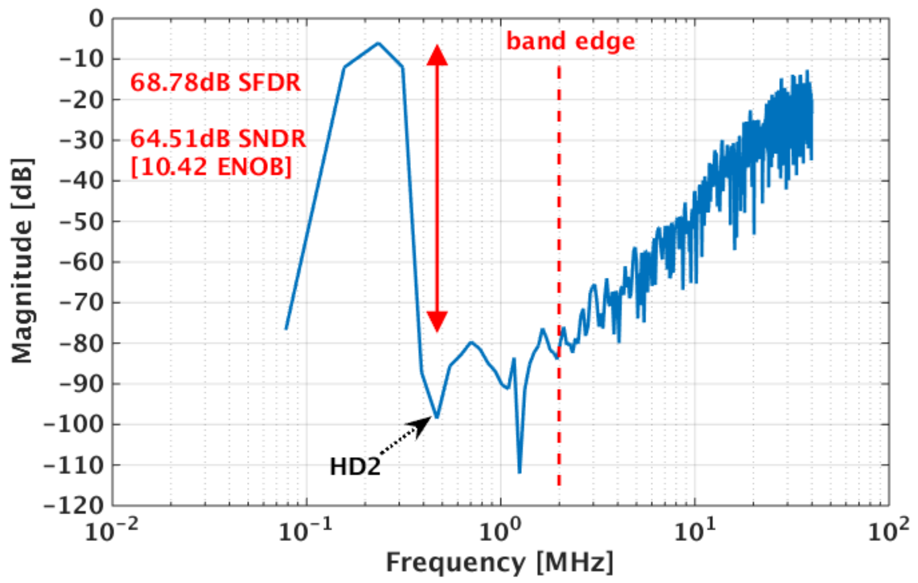
4.25a) accounts for the majority of the even-order distortion. The simulation results with and without this single parasitic capacitance are shown in Figure 4.25b. If the parasitic capacitance were matched on the opposite side of the differential structure then the effect would cancel; the automated layout causes the capacitive coupling to be different at the two opamp input nodes, creating even-order distortion.

4.4 Experimental Results

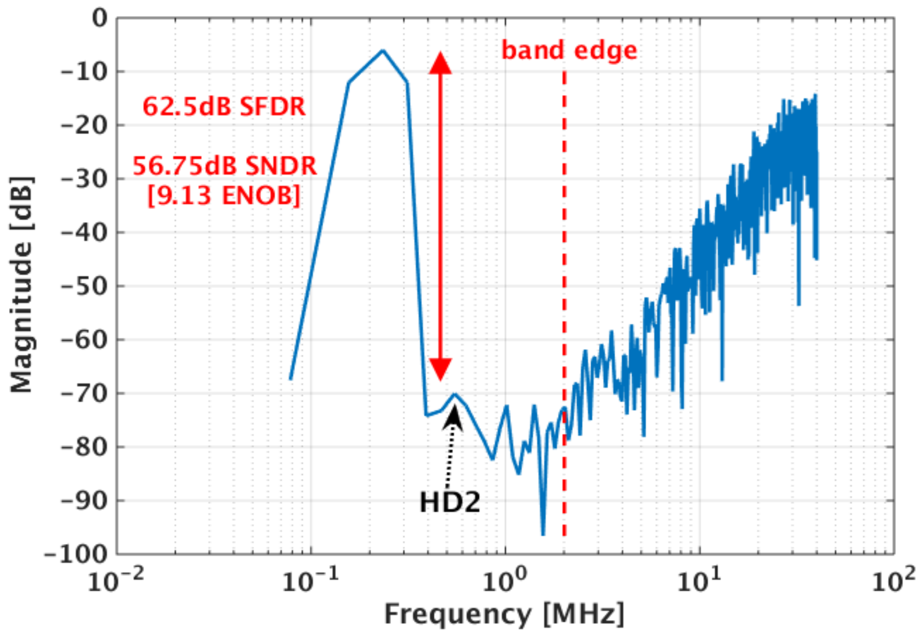
4.4.1 MASH ADC in 130nm CMOS

Manufactured in National Semiconductor's 130nm Optimos process, the 1-1-1 MASH ADC (project name "OptiMASH") floorplanning is shown in Figure 4.26. An example output spectrum is shown in Figure 4.27, demonstrating the 3rd-order noise shaping and the flattening of the noise spectrum due to finite gain in the amplifiers. An OSR of 20 places the band edge at approximately the corner of the noise shaping, such that increasing OSR beyond this point provides very little SNDR benefit. Figure 4.28 show the performance over different input amplitudes and sampling rates. These measurements are all recorded with a ≈100kHz input signal, but it was verified SNDR is sustained at higher f_{in} values up to 2MHz. The ADC maintains performance up to 80MS/s, at which point performance drops off as the Strong-Arm latches fail to resolve within the sampling period.

Operating at 80MS/s it achieves 56.2-dB SNDR at an OSR of 20. It consumes 983μW from a 1.2-V supply for a resulting FoM of 466fJ/conversion-step. This power consumption includes all integrators, quantizers, DACs, clock generation circuitry and reference voltages. A breakdown of the power consumption by

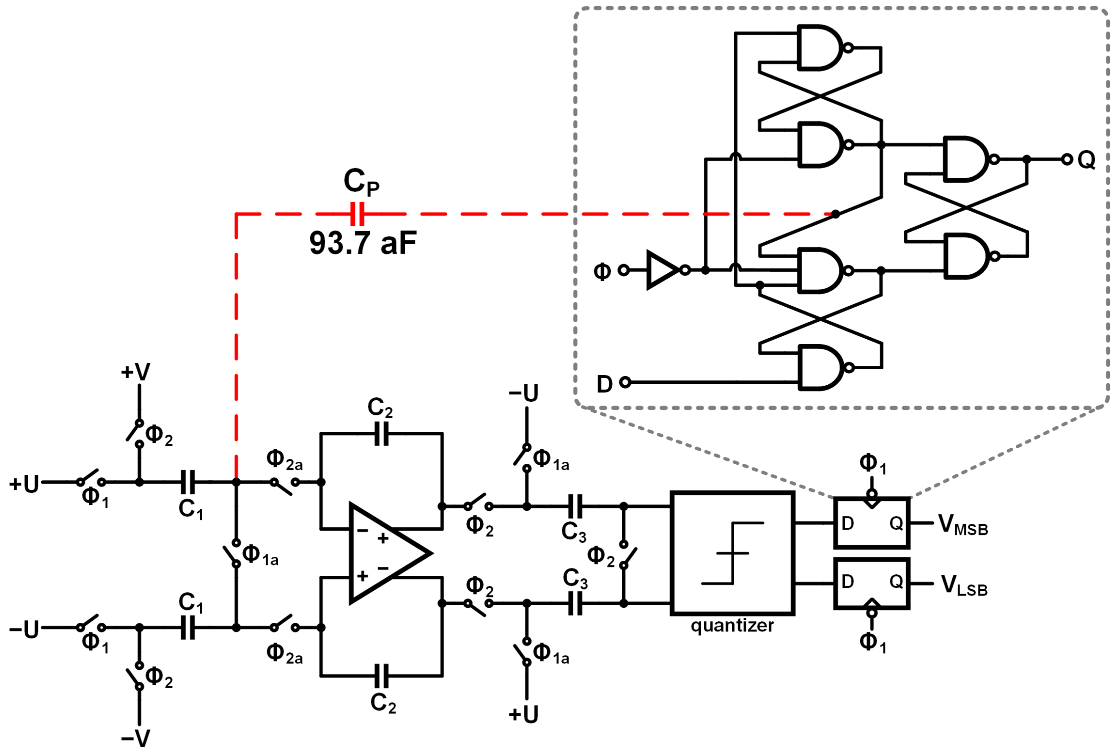


(a)

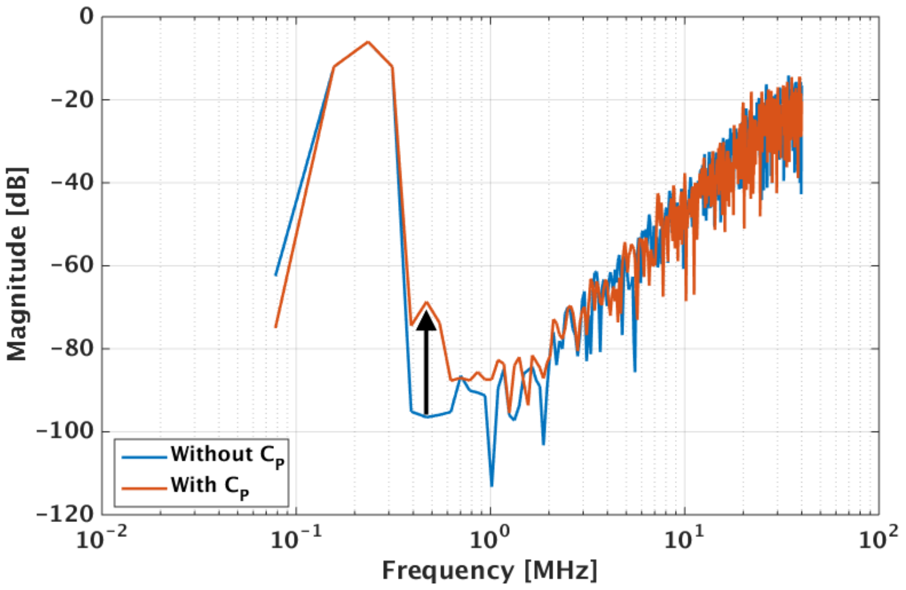


(b)

Figure 4.24: Final simulation results with transient noise, for (a) transistor-level and (b) post-layout extraction.



(a)



(b)

Figure 4.25: Parasitic capacitance causing 2nd-order distortion, shown in (a) schematic and (b) the resulting FFT spectrum.

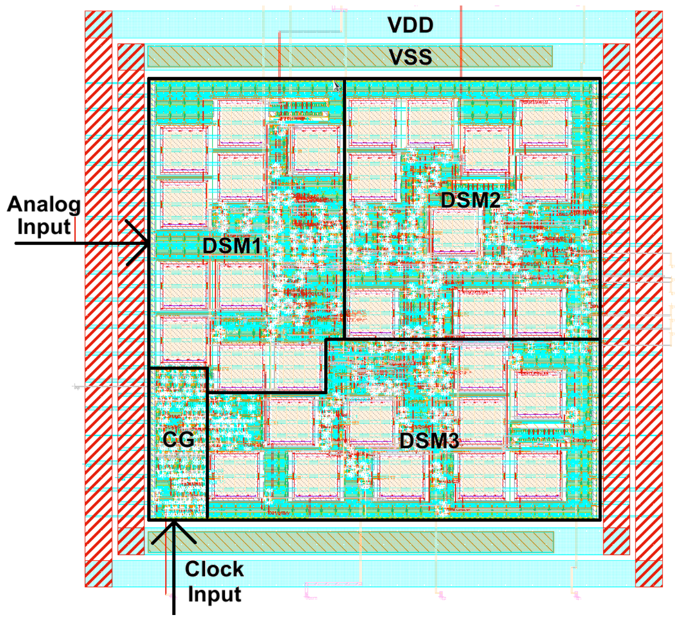


Figure 4.26: Layout view of floorplanning for MASH ADC in 130nm process.

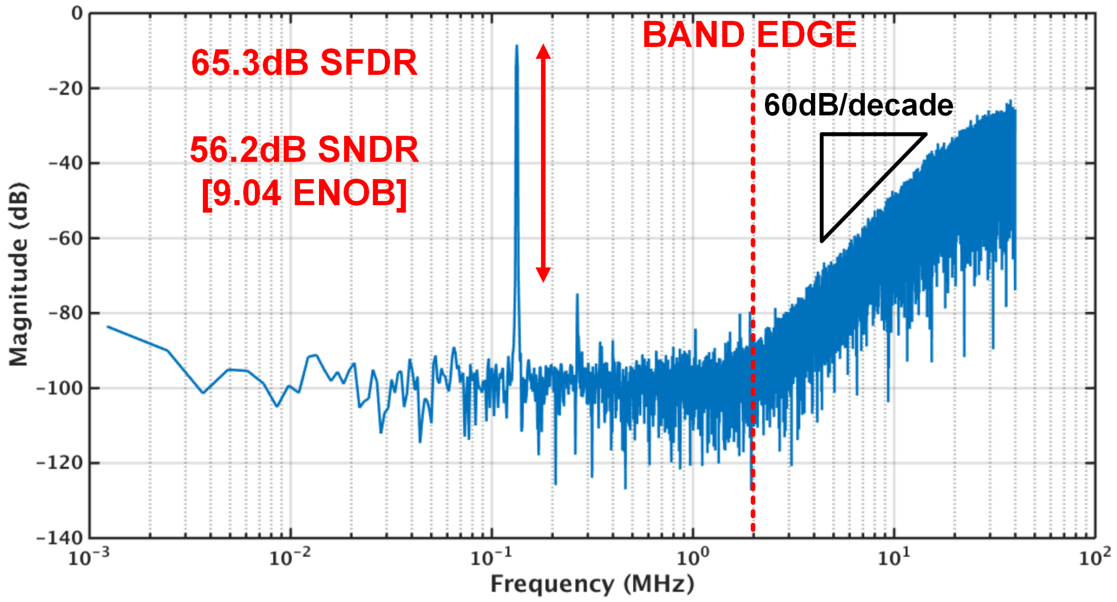


Figure 4.27: Example output spectrum of 130nm MASH ADC.

Table 4.1: OptiMASH ADC Summary of Performance

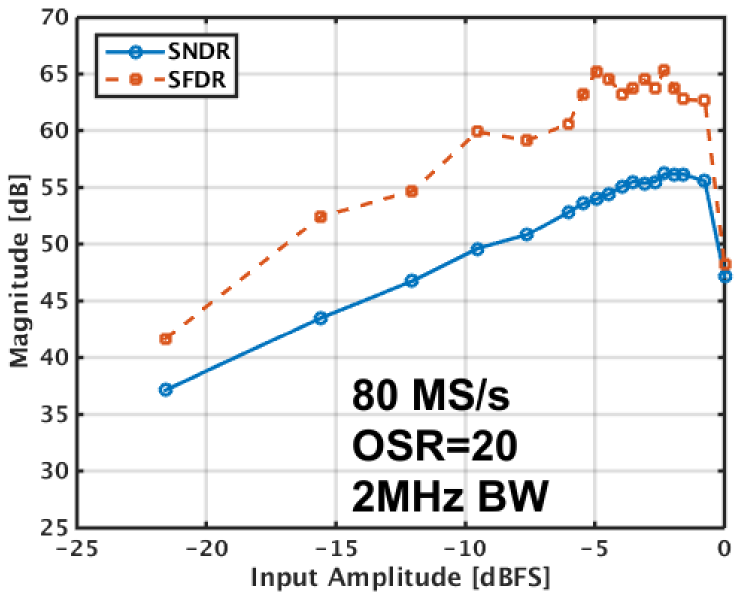
Technology	1P6M 130nm CMOS
Area	0.046mm^2 (0.215mm x 0.215mm)
Supply Voltage	1.2V
Sample Rate	80 MS/s
OSR	20
Bandwidth	2 MHz
SNDR	56.20 dB
ENOB	9.04 bits
Power	983.4 μW
FoM	466 fJ/conv-step

block (estimated from extracted layout simulations) is shown in Figure 4.29. Note that the latter MASH stages consume more power than the first, on account of processing the higher activity quantization error. In future designs the capacitor sizes could be scaled down in the latter stages to reduce their power consumption, but that was not implemented in this proof-of-concept ADC; all stages use a 100fF unit capacitor.

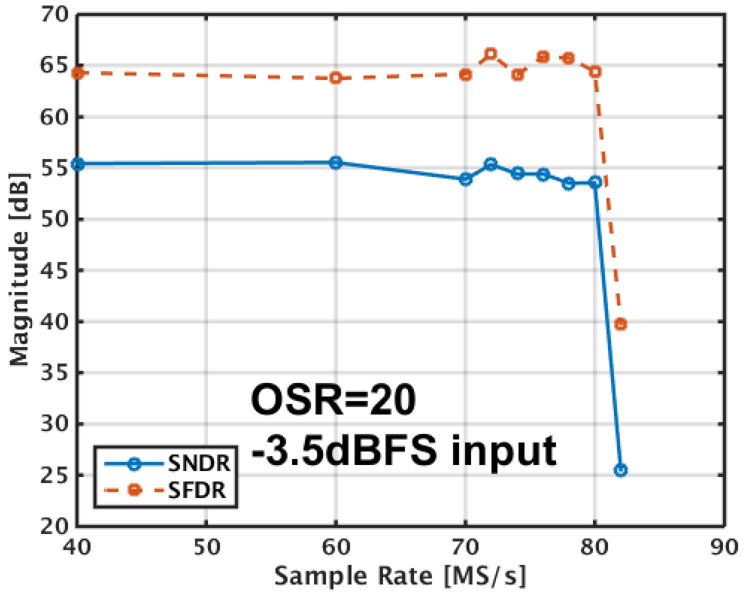
A performance summary of the ADC is included in Table 4.1, showing 9+ ENOB with 2MHz bandwidth. It occupies an active area of 0.046mm^2; a die photo is shown in Figure 4.30.

4.4.2 MASH ADC in 65nm CMOS

The second chip implements the ADC using the same Verilog code in a smaller process node, in order to demonstrate the scalability and portability of this proposed design automation toolchain. The new version (project name “OptiMASH Prime”) is implemented in a 65nm CMOS process provided by Taiwan



(a)



(b)

Figure 4.28: Performance of 130nm MASH ADC, sweeping across (a) input amplitude and (b) sample rate. Results use OSR = 40 in all cases.

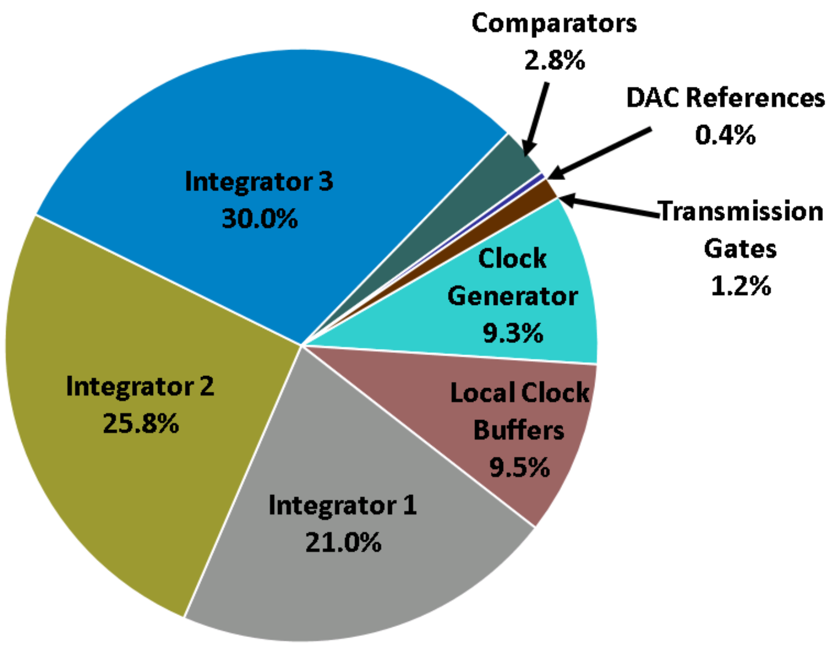


Figure 4.29: Estimated power breakdown in 130nm MASH ADC.

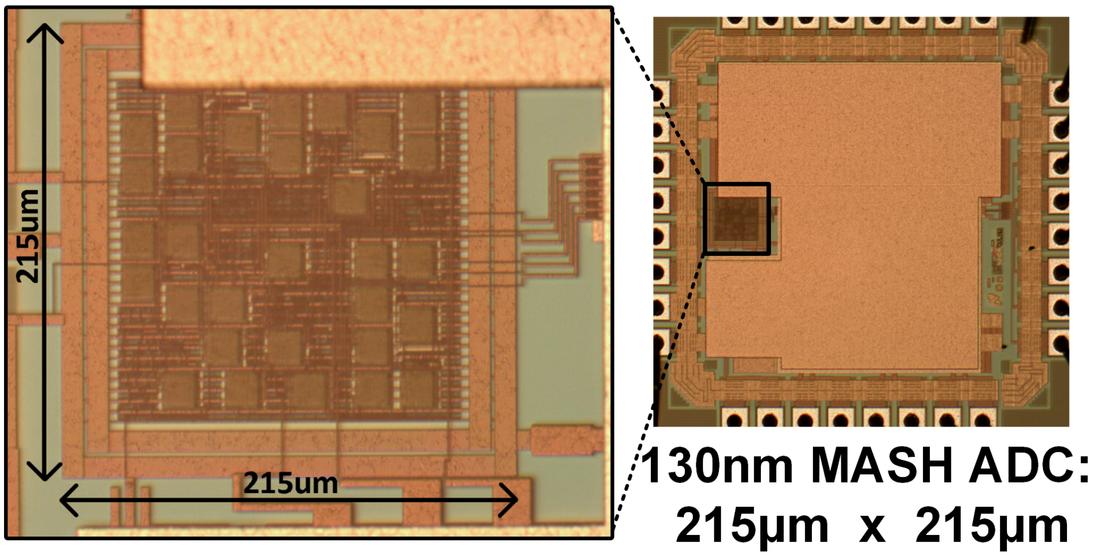


Figure 4.30: Die photo of synthesized MASH ADC in 130nm process.

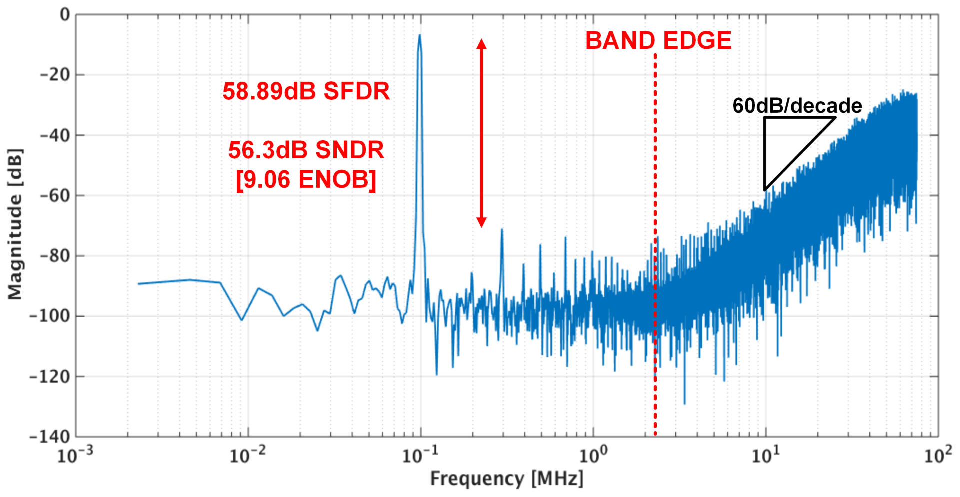
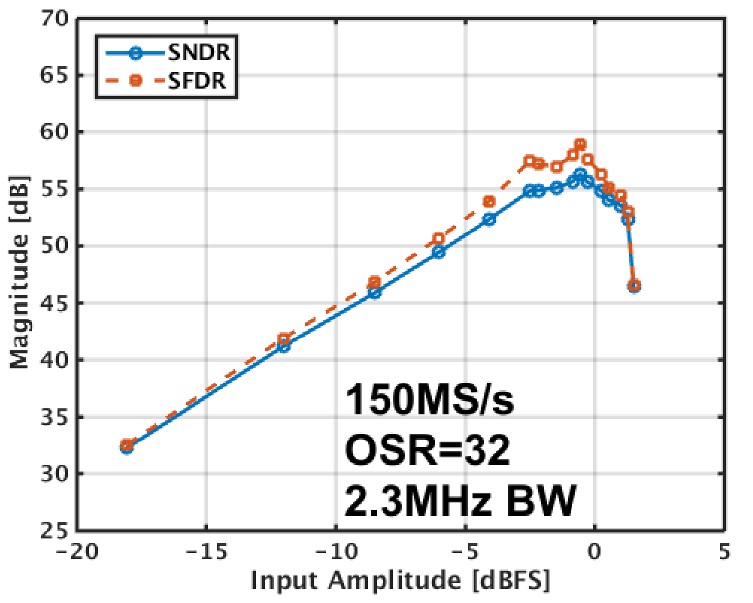


Figure 4.31: Example output spectrum of 65nm MASH ADC.

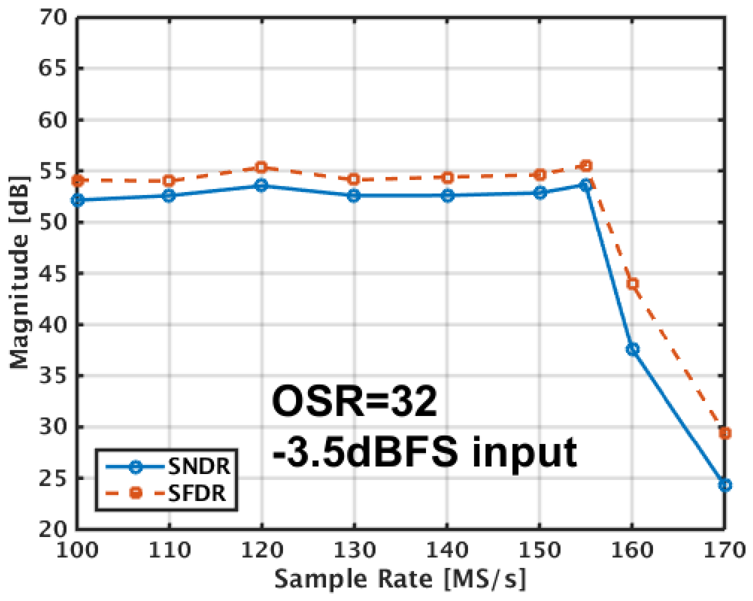
Semiconductor Manufacturing Company (TSMC). An example output spectrum is shown in Figure 4.31, and the performance over different input amplitudes and sampling rates are included in Figure 4.32.

Operating at 150MS/s it achieves 56.3-dB SNDR at an OSR of 32. It consumes 872μW from a 1-V supply for a resulting FoM of 348.6fJ/conversion-step. A breakdown of the power consumption by block (estimated from extracted layout simulations) is shown in Figure 4.33. Although all stages are identical, the power consumption in the latter two integrators are reduced because their input common mode signals are lower than the first stage. This design used a smaller unit capacitor size (50fF) than the previous version, but is still not noise-limited. Resolution is limited by the third harmonic.

A performance summary of the ADC is included in Table 4.2, showing 9+ ENOB with 2.3MHz bandwidth. It occupies an active area of 0.014mm^2; a die photo is shown in Figure 4.34.



(a)



(b)

Figure 4.32: Performance of 65nm MASH ADC, sweeping across (a) input amplitude and (b) sample rate. Results use OSR = 75 in all cases.

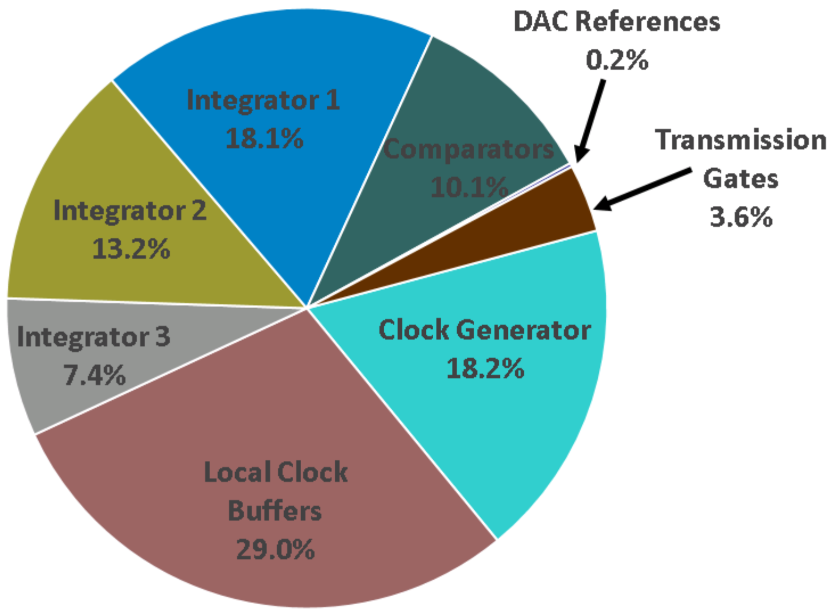


Figure 4.33: Estimated power breakdown in 65nm MASH ADC.

Table 4.2: OptiMASH Prime ADC Summary of Performance

Technology	1P9M 65nm CMOS
Area	0.014 mm^2 (0.108mm x 0.128mm)
Supply Voltage	1V
Sample Rate	150 MS/s
OSR	32
Bandwidth	2.34 MHz
SNDR	56.30 dB
ENOB	9.06 bits
Power	872.0 μW
FoM	348.6 fJ/conv-step

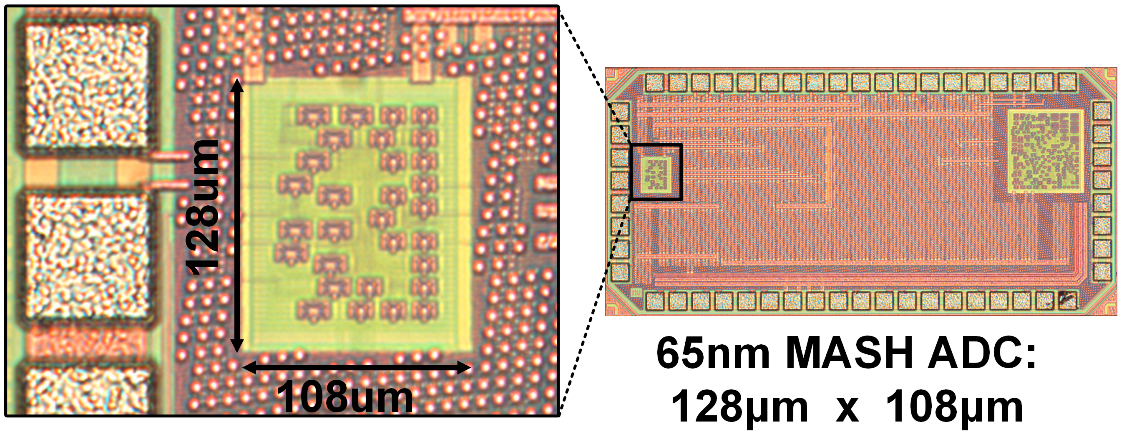


Figure 4.34: Die photo of synthesized MASH ADC in 65nm process.

4.4.3 Comparison to State-of-the-Art

Figure 4.35 compares the performance achieved in this work to the state-of-the-art work presented at ISSCC. Even with the automatically place-and-routing, this work reaches respectable speed and resolution. It should be noted that this performance comes with a power overhead, making FoM less favorable. However, the rapid Verilog-to-layout design time and portability are useful for mixed-signal design.

4.5 Summary

The work in [22] is the only other on record to achieve fully automated Verilog-to-layout synthesis, but the accuracy is limited to 35.9-dB SNDR (5.7 ENOB). The work described here is the second demonstration of fully automated Verilog-to-layout synthesis, and the first to incorporate rudimentary analog cells

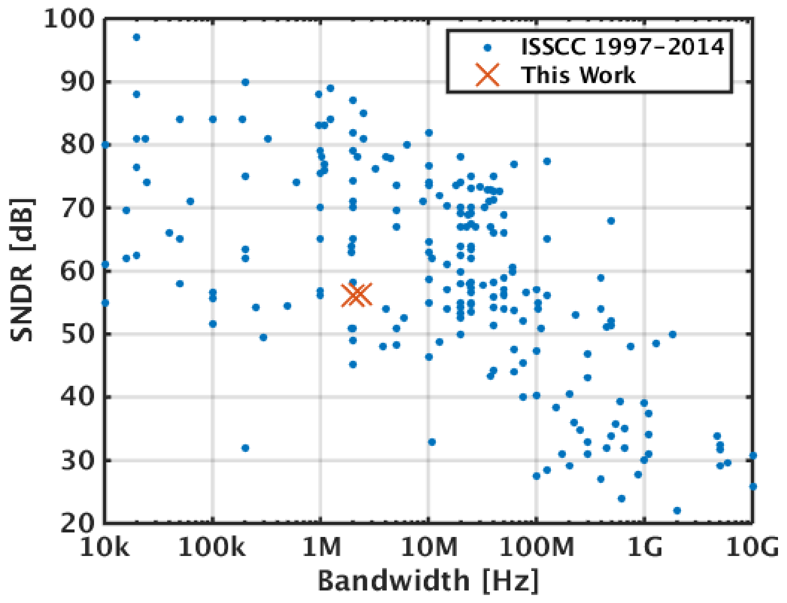


Figure 4.35: Comparison of this work to ISSCC publications.

into the “digital” library for Verilog. This results in improved resolution to above 9 ENOB. Expanding the library further to include more analog capabilities, as well as considering other Verilog-tolerant ADC architectures, will open the door to more design possibilities.

CHAPTER 5. RING AMPLIFIER-BASED PIPELINE ADC SYNTHESIZED FROM VERILOG CODE

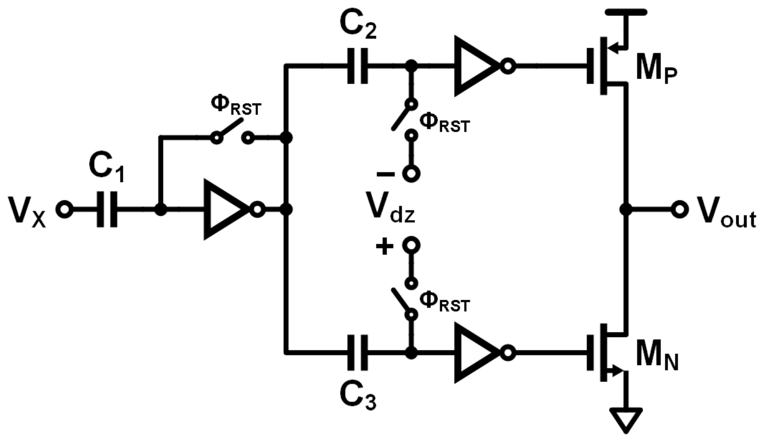
"Everyone has a plan until they get punched in the mouth."

— Mike Tyson

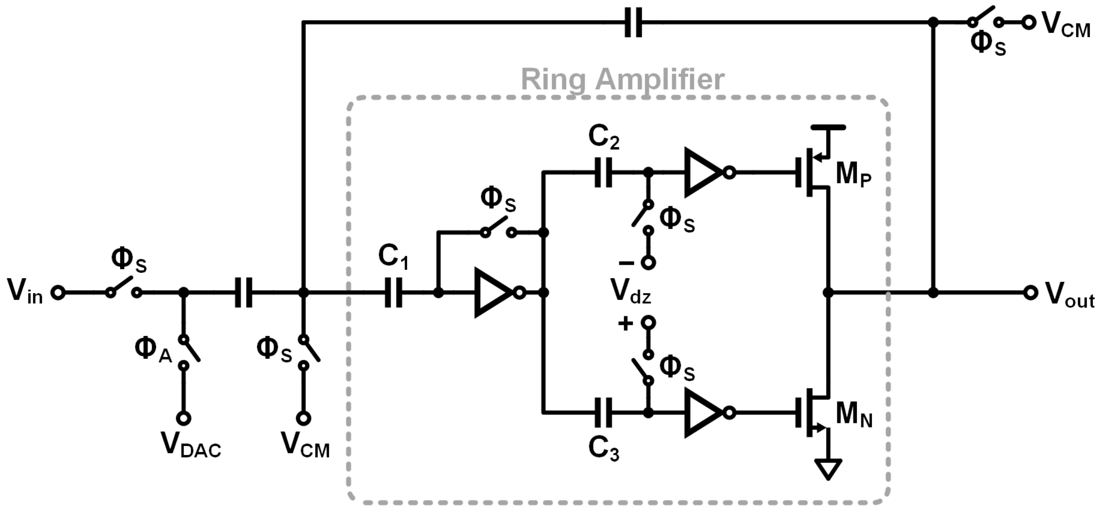
This Chapter describes the synthesis of a pipeline ADC, using the highly scalable ring amplification technique. It begins by giving some background on ring amplifiers to explain why they are so amenable to process scaling and to the design automation toolchain used in this work. The top-level architecture of the implemented ADC is presented, followed by circuit design details for the custom analog blocks. Measurement results are provided for a test chips in 65nm CMOS.

5.1 Background on Ring Amplifier

In light of the scaling challenges described in Chapter 1, the ring amplifier (abbreviated "ringamp" or "RAMP") was proposed as an efficent amplification technique in modern CMOS processes. The basic structure introduced in [49] is shown in Figure 5.1a. It is created by splitting a three-stage ring oscillator into two signal paths with different offsets embedded onto capacitors C_2 and C_3. These offsets create a range of input values for which neither of the output devices (M_P and M_N) conduct. It is this non-conducting deadzone that allows the ring oscillator-like structure to stabilize and behave like an amplifier. Figure 5.1b shows an example of the ringamp in a multiplying DAC (MDAC), for which the ringamp



(a)



(b)

Figure 5.1: Basic architecture of (a) stand-alone ring amplifier and (b) ring amplifier-based MDAC.

reset phase takes place while the input is being sampled. Capacitor C_1 cancels the difference between the input common mode and the trip-point of the first inverter, ensuring that the final settled value of the input will be V_{CM} regardless of the precise inverter threshold.

One fundamental advantage of the ring amplifier is the initial slew-based charging shown in Figure 5.2. A conventional opamp charges its capacitive load

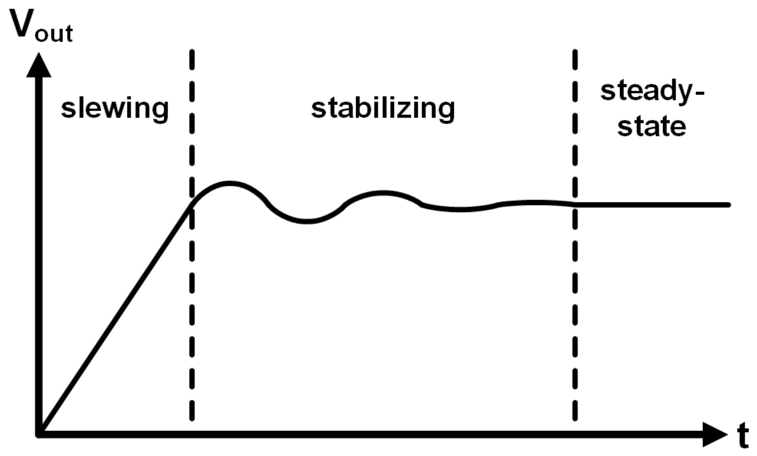


Figure 5.2: Example ring amplifier output waveform showing the three phases of operation.

with some form of RC-based settling. The slew-based charging here allows even large loads to be charged efficiently with small ringamp transistor sizes. This is an advantage for scalability because it helps decouple the ringamp's internal device sizing and power requirements from the size of the load capacitors. Along with the immunity to output compression, this makes the ringamp an attractive amplification solution for synthesizable ADC design.

While originally implemented as the coarse-path amplifier in a split correlated-level-shifting (CLS) pipeline ADC (the fine amplifier being a double-cascode telescopic opamp), later work demonstrated a ringamp-only pipeline ADC [50]. Further work has created high-precision ring amplifiers [51] and self-biased ring amplifiers [52] by adding more analog complexity to the structure. These recent publications have also explored different methods of how and where to embed the deadzone, as well as dynamically changing V_{DZ} during each amplification phase. For synthesized ADC design, it is preferable to use the highly-digital basic structure shown in Figure 5.1.

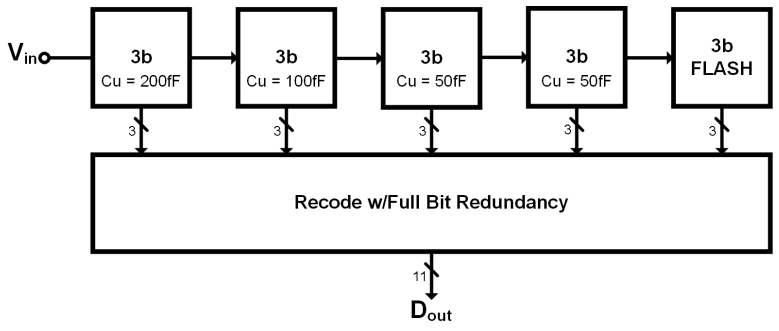
For the interested reader, a more extensive and detailed analysis of ring amplifier design may be found in [53] or [54].

5.2 Proposed System Architecture

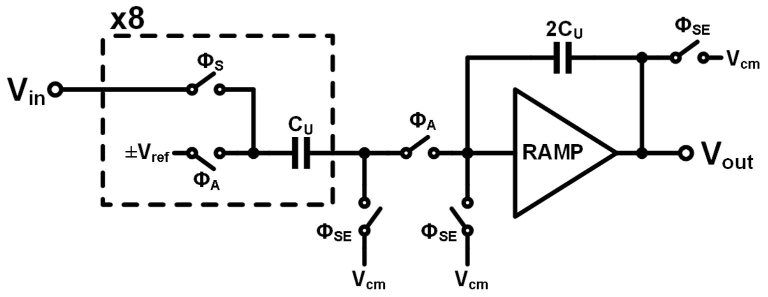
5.2.1 Top-Level System Design

The implemented pipeline ADC is shown in Figure 5.3a, using four 3-bit MDAC stages and a 3-bit backend flash quantizer. With a full bit of redundancy between each stage, the total resolution is 11 bits. The multiply-by-4 MDAC shown in Figure 5.3b is used in each stage. The schematic of the pseudo-differential implementation is shown in Figure 5.4, and includes switched-capacitor common-mode feedback (CMFB) as well as additional enable switches to save power. The "reset" phase takes place during the MDAC's sampling phase, and the "enable" phase takes place during the amplification period.

Sizing for the MDAC capacitors (C_U) is determined by thermal noise requirements and will be discussed in Section 5.2.3. The CMFB capacitors are then sized to ensure that the CMFB path gain is several times smaller than the gain of the primary MDAC feedback path. The ringamp output devices (M_P and M_N) are sized according to slewing requirements, based on the load capacitance and the desired bandwidth (this design targeted 30MHz sample rate to mimic the ringamp characterization ADC presented in [49]). The input devices are sized based on amplifier noise requirements, but the second stage is allowed to be sized very small.



(a)



(b)

Figure 5.3: (a) Top-level architecture of proposed pipeline ADC, and (b) schematic of MDAC stage (shown single-ended for simplicity).

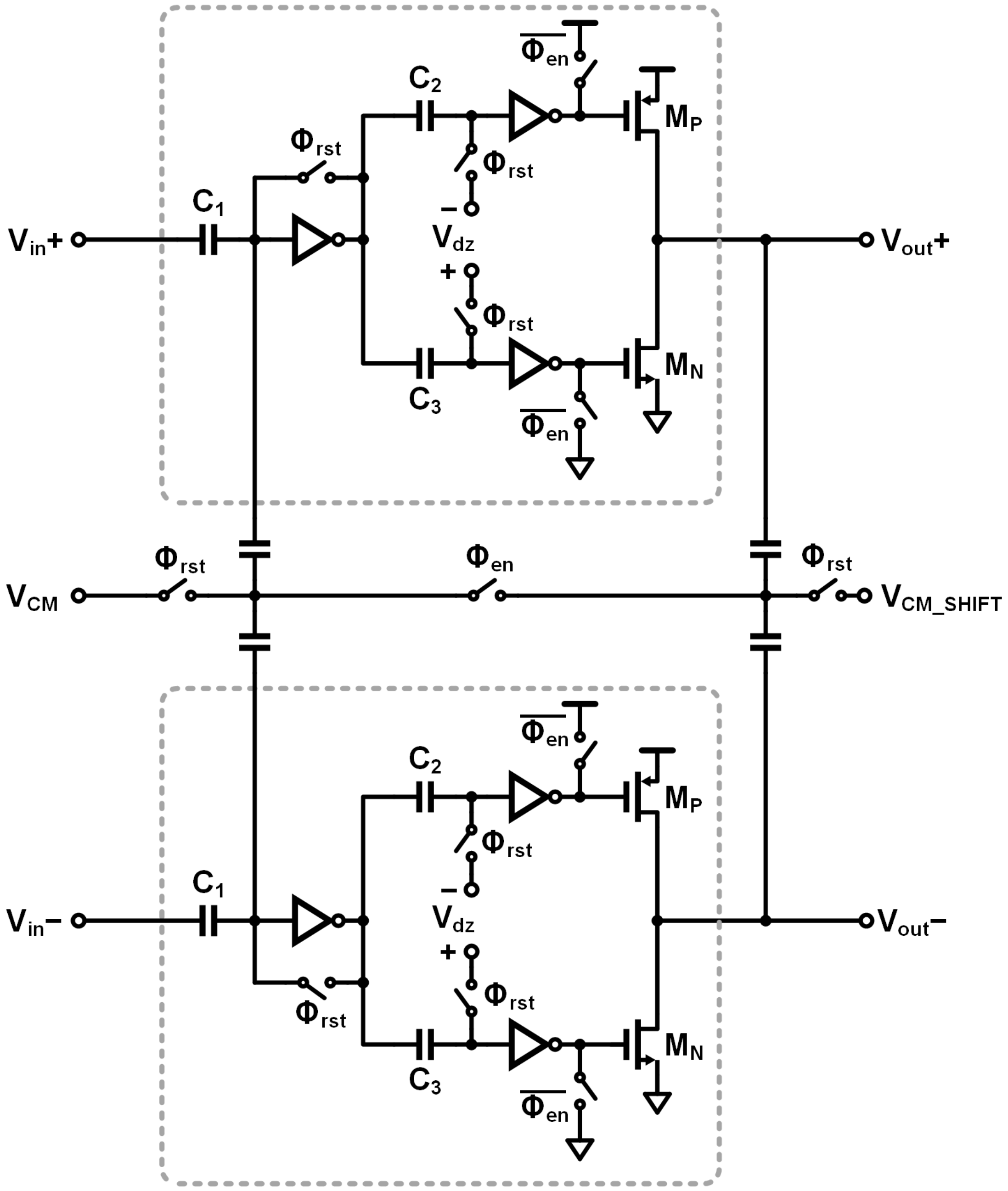
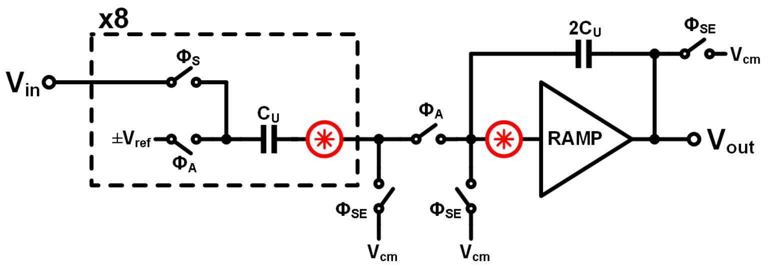
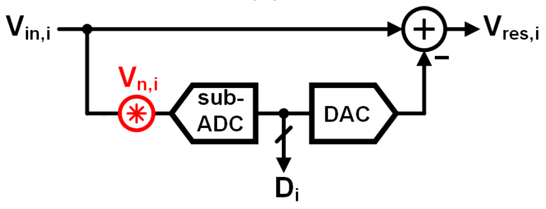


Figure 5.4: Pseudo-differential ring amplifier implementation, with added power-saving feature and switched-capacitor CMFB.



(a)



(b)

Figure 5.5: Location of noise sources (a) in switched-capacitor MDAC stage, and (b) an equivalent block diagram combining sources for i-th MDAC stage.

5.2.2 Noise Sources

Figure 5.5a highlights the noise contributions in a switched-capacitor MDAC stage: the kT/C noise from the sampling capacitance, and the input-referred opamp noise from the ring amplifier. The two are additive and can be combined into an equivalent source as shown in Figure 5.5b. Then the total input-referred noise for the pipeline ADC will be the sum of the noise in each individual stage, divided by the appropriate amount of interstage gain:

$$\sigma_n^2 = \sigma_{n,1}^2 + \frac{\sigma_{n,2}^2}{G} + \frac{\sigma_{n,3}^2}{G^2} + \cdots + \frac{\sigma_{n,k}^2}{G^{k-1}} \tag{5.1}$$

where k is the total number of MDAC stages and G is the interstage gain.

5.2.3 Capacitor Sizing

Just like in the previous Chapter, this analysis will begin by supposing that the entire noise budget is allocated to the sampling capacitance. It will proceed to calculate the minimum capacitor sizing to achieve the full resolution of the pipeline. Then the capacitors will be sized well above this value, in order to ensure that kT/C noise is not the limiting factor. As a simplification, assume that the multiply-by-4 MDAC stage shown in 5.3b is used and that the capacitor sizes are scaled by two between each stage. Then the total noise from the four MDAC stages is:

$$\begin{aligned}\sigma_n^2 &= \sigma_{n,1}^2 + \frac{2 \cdot \sigma_{n,1}^2}{4} + \frac{2^2 \cdot \sigma_{n,1}^2}{4^2} + \frac{2^3 \cdot \sigma_{n,1}^2}{4^3} \\ &= 1.875 \cdot \sigma_{n,1}^2\end{aligned} \tag{5.2}$$

Substituting the PSD for a switched-capacitor branch (as explained in Section 4.2.3):

$$\sigma_n^2 = 1.875 \cdot \frac{2kT}{C_S} \tag{5.3}$$

and then determining the SNR:

$$\begin{aligned}SNR &= 10 \cdot \log_{10}\left(\frac{\sigma_{sig}^2}{\sigma_n^2}\right) \\ &= 10 \cdot \log_{10}\left(\frac{A^2/2}{1.875 \cdot \frac{2kT}{C_S}}\right)\end{aligned} \tag{5.4}$$

and solving for the sampling capacitance:

$$C_S = 1.875 \cdot \left(\frac{2kT}{A^2/2}\right) \cdot 10^{SNR/10} \tag{5.5}$$

For the full 11-bit effective resolution, this noise should fall 68dB below a full-scale input signal ($A = 1$):

$$C_S \geq 193.2 \;\; [fF] \tag{5.6}$$

A sampling capacitance of 1.6 pF (a factor of 8 larger) is used instead, to ensure that kt/C noise won't limit the ADC resolution. The ringamp is tolerant to this increased loading because of the slew-based charging described in Section 5.1. It will, however, increase power consumption, but that is deemed acceptable in this proof-of-concept ADC. The thermal noise limit becomes:

$$\begin{aligned} SNR &= 10 \cdot \log_{10}\left(\frac{0.5}{1.875 \cdot \frac{2kT}{1.6 \cdot 10^{-12}}}\right) \\ &= 77.2 \;\; [dB] \end{aligned} \tag{5.7}$$

This will reduce the maximum SNR only slightly below the 11-bit quantization error limit.

5.3 Minimalist Sub-Block Design

5.3.1 *Ring Amplifier Output Stage*

The ringamp is fortunately composed almost entirely of inverters (from the standard digital library), switches and capacitors (both added to our custom analog library in Chapter 4). The only new cell needed to implement the ringamp is the output stage highlighted in Figure 5.6. This new custom analog cell has the advantage of being nearly identical to an inverter. The schematics and layout are readily available by removing the connection between the gates of the two transistors in the inverter.

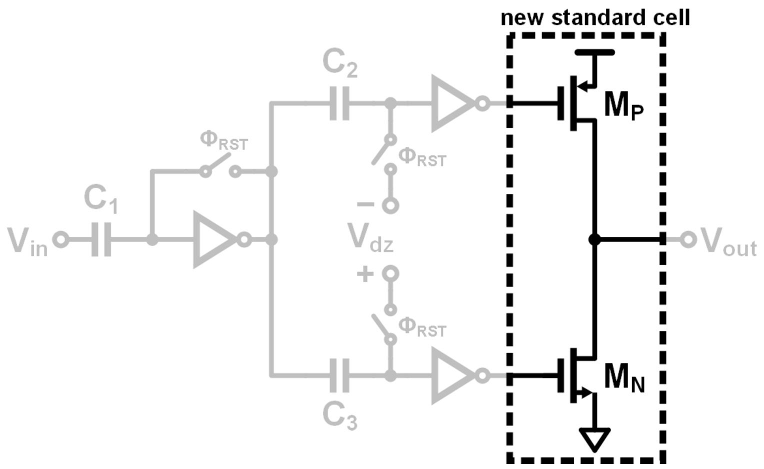


Figure 5.6: Output stage of ringamp, to be added as custom analog cell.

5.3.2 *Sub-ADC Reference Ladder*

The schematic for the sub-ADC is shown in Figure 5.7. It uses the same switched capacitor quantizer as described in Chapter 4, only with more references generated from a resistor ladder. The resistor ladder uses the final analog custom cell, the unit resistor. This sub-block is included here only in the interest of completeness, being that it is synthesized from custom analog cells.

5.3.3 *Final Simulation Results*

The final pre- and post-layout extraction simulation results are shown in Figure 5.8, including transient noise. Extracted results show increased distortion due to routing mismatch between the pseudo-differential signal paths. Figure 5.9 shows the performance across different deadzone settings, showing a wide range of values at which quantization noise limits the resolution rather than amplifier

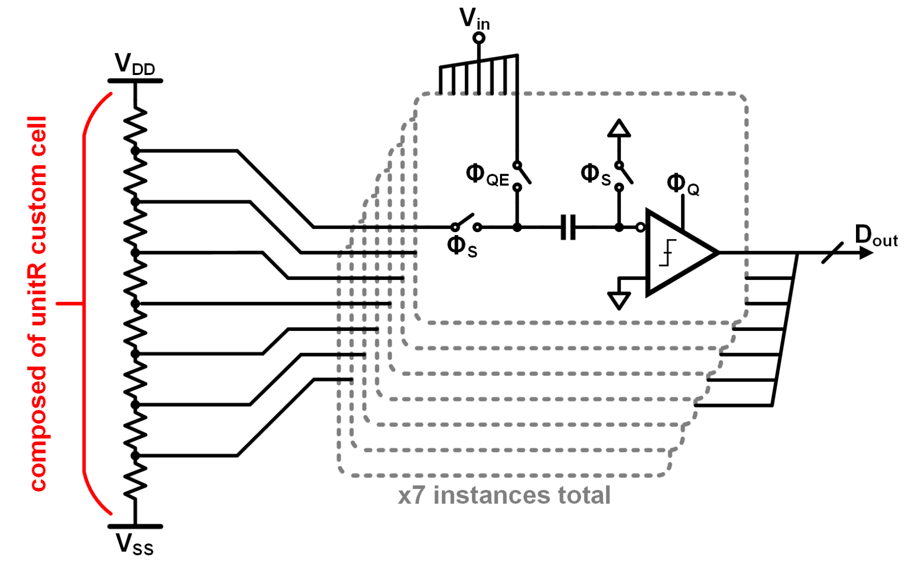
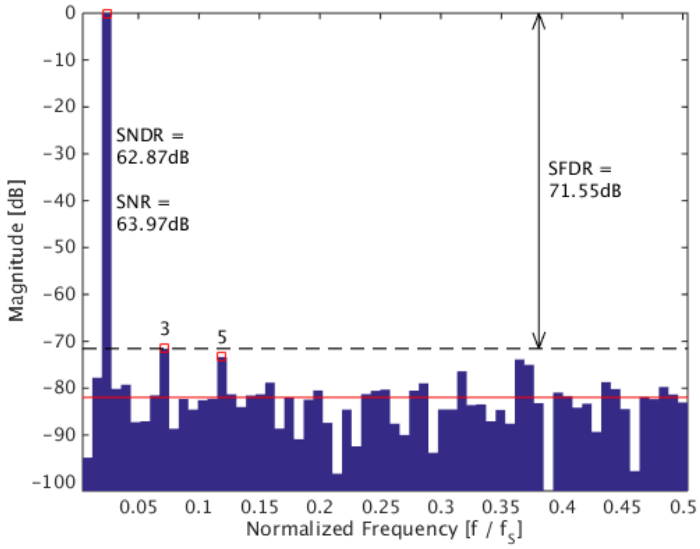


Figure 5.7: Implementation of sub-ADC and resistive reference ladder (shown single-ended for simplicity).

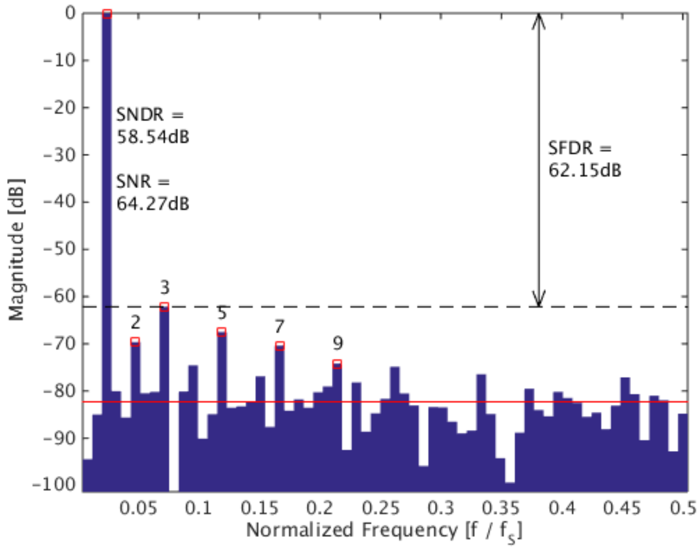
performance. However, within this plateau the ringamp settling time (and consequently power consumption) are still dependent on the deadzone voltage. For a given sample rate it is optimal to choose the deadzone voltage corresponding to the far right side of the plateau, to achieve the maximum resolution without wasting power consumption due to extra ringing behavior.

5.4 Experimental Results

Experimental results are in the process of being collected.



(a)



(b)

Figure 5.8: Final simulation results with transient noise, for (a) transistor-level and (b) post-layout extraction.

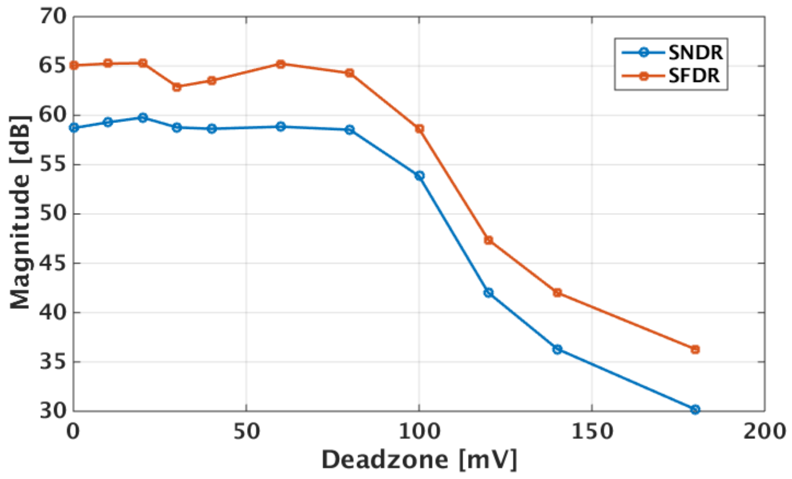


Figure 5.9: Extracted simulation results across deadzone settings.

5.4.1 *Pipeline ADC in 65nm CMOS*

A performance summary of the ADC is included in Table 5.1. While measurement results are being refined, this table includes final extracted simulation results. The ADC achieves 9.43 ENOB with 15MHz bandwidth; operating with 3.23 mW power consumption this corresponds to an FoM of 156 fJ per conversion step. It occupies an active area of 0.058mm^2; a die photo is shown in Figure 5.10.

5.4.2 *Comparison to State-of-the-Art*

Figure 5.11 compares the performance achieved in this work to the state-of-the-art work presented at ISSCC. Like the work presented in Chapter 4, it reaches respectable speed and resolution despite being automatically routed. Please note that this is based on extracted simulation results, pending further measurements.

Table 5.1: Ringamp-based Pipeline ADC Summary of Performance

Technology	1P9M 65nm CMOS
Area	0.058mm^2 (0.230mm x 0.250mm)
Supply Voltage	1.0V
Resolution	11 bits
Input Range	2.0 V_{p-p} differential
Sample Rate	30 MHz
SNDR	58.54 dB
ENOB	9.43 bits
Power	3.23 mW
FoM	156 fJ/conv-step

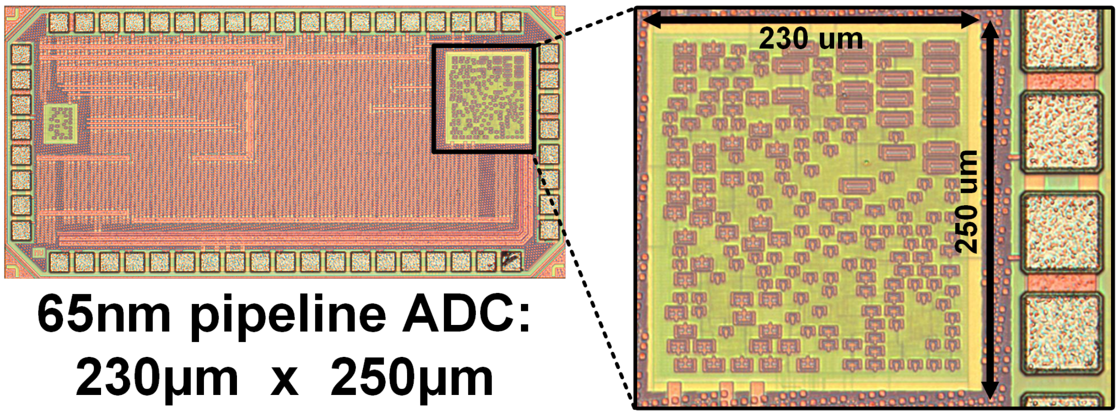


Figure 5.10: Die photo of synthesized pipeline ADC in 65nm process.

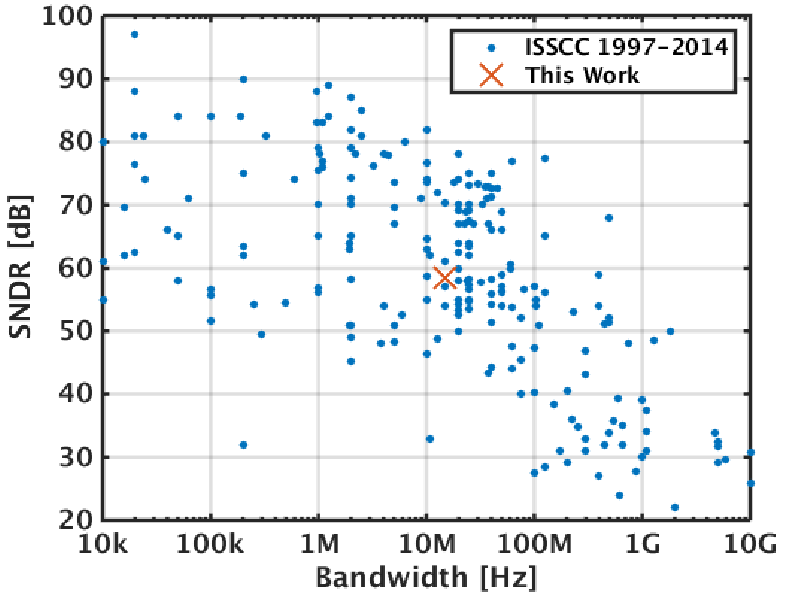


Figure 5.11: Comparison of this work to ISSCC publications.

5.5 Summary

The work presented in this Chapter extends the possible applications of synthesized Verilog-to-layout ADCs to higher conversion speeds. It makes use of the highly digital, highly scalable ring amplification technique. The slew-based charging provides an easily portable amplification solution that is beneficial for synthesized ADC design. Furthermore, because the ringamp consists entirely of inverters, switches and capacitors it is easily constructed from our set of analog custom cells.

CHAPTER 6. CONCLUSION

“So we beat on, boats against the current, borne back ceaselessly into the past.”

— F. Scott Fitzgerald (*The Great Gatsby*)

6.1 Summary

A procedure for automating the layout of ADCs is presented. This procedure makes use of the existing synthesis and place-and-route tools that are common in digital circuit design. A method for adding rudimentary analog cells to the standard library is described, allowing the designer to synthesize mixed-signal designs from Verilog code. By using cells that are simple and highly scalable, the same Verilog code may be used to implement the design in any number of process nodes, for rapid portability and scalability. Two different ADC architectures were implemented as proofs of concept: first, a third-order MASH ADC is fabricated in 130nm and 65nm CMOS, taking advantage of the structure’s tolerance to the mismatch introduced by the automated place-and-routing. These chips achieved 9+ ENOB with bandwidths of 2 MHz and 2.34 MHz, respectively. Second, a Nyquist-rate pipeline ADC using the highly-scalable ring amplifier is fabricated in 65nm CMOS. It achieves 9.43 ENOB with a bandwidth of 15 MHz. The measurement results from these chips show that synthesized ADCs can achieve moderate performance with drastically reduced design time compared to manual layout.

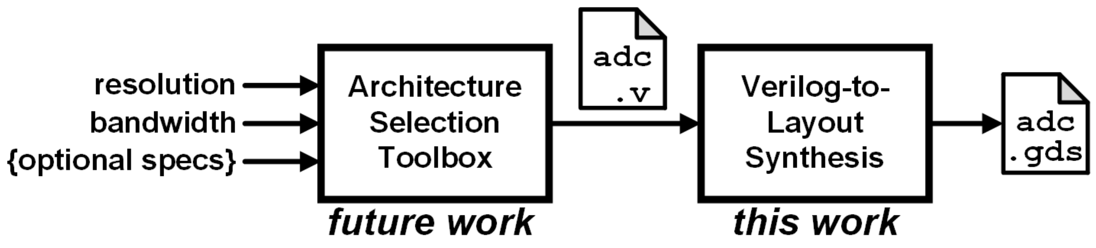


Figure 6.1: Automating the creation of the Verilog code in future work.

6.2 Future Work

6.2.1 Automating Architecture Design

The work described in this thesis automates the layout of ADCs from Verilog code. One direction for future work is to automate the creation of Verilog code, from the performance specifications provided by the designer. As shown in Figure 6.1, this tool would allow the designer to provide the desired bandwidth, resolution, and any other target specifications and decide upon the optimal (or at least satisfactory) ADC architecture.

This would rely on having process-dependent information about a set of different ADC architectures, such as the MASH and pipeline converters presented here. For example, the Architecture Selection Toolbox would need know the SNDR achieved by different order MASH ADCs across OSR (like the results shown in Figure 4.13). Similarly, the toolbox would need information about the capacitor sizing requirements to achieve a specific SNDR with different numbers of 1.5-bit pipeline ADC stages. Although not implemented here, it would be helpful for the toolbox to include information about SAR ADC performance as well. The result would appear similar to Figure 6.2, with variations inside each architecture type

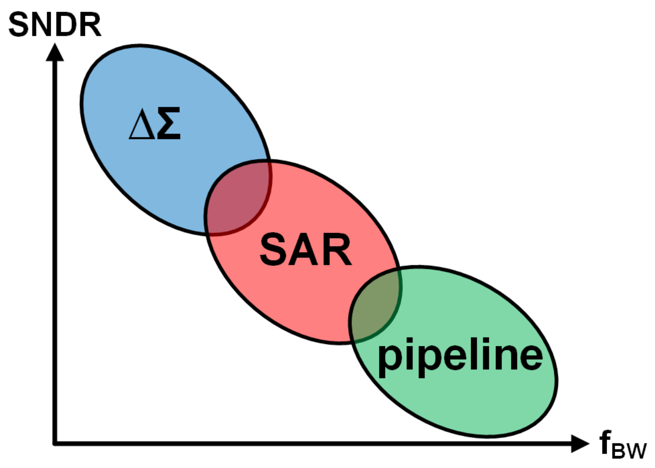


Figure 6.2: Graphical representation of resolution/bandwidth look-up table used by automation toolbox.

due to the number of stages and capacitor sizing. The toolbox could compile a list of possible design solutions from these pre-determined relationships, eliminating those which do not meet the user's specifications. Of the remaining architectures, the user could instruct the tool to choose the option which minimizes area, power, or some other parameter.

Given the selected architecture, a Verilog file could be automatically created describing the circuit. Automating this code generation may be the most difficult part of this future work. The Verilog code can then be processed by the Verilog-to-layout automation toolchain presented in this thesis, arriving at the final layout from only the desired performance specifications.

6.2.2 Analog Cells with Fake Logic Functions

In Section 3.3.4, it was explained that portions of the standard cell library information in the LIB file has no meaning for analog cells. The cells do not have logical functions that can be described in Boolean logic, and the digital timing information (for example, rise time and fall time) does not apply. The absence of logical function forced the custom analog cells to be instantiated explicitly in this work, and the absence of timing information left the PNR tool without guidance for routing connections. Floorplanning was used to improve the routing, but it is not an ideal solution.

The first problem (lack of logical function) causes a problem because the designer must decide the sizing of the analog cells when they are explicitly instantiated: for example, he must select either a 1X- or 2X-sized transmission gate, or a 50fF or 200fF unit capacitor cell. If there were a way to describe the analog function instead, then it would open the door for the logic synthesizer to somehow determine device sizing the same way that it does for digital cells. The logic function must be some Boolean expression that can be expressed in Verilog. Whatever function is chosen leaves some risk (however unlikely) that the digital portion of the circuit will synthesize to the same function, and will *mistakenly* select this custom analog cell in place of the true digital logic function. Therefore the function should be something complex and unlikely to occur in the true digital logic. For example, suppose that the LIB information for each opamp cell (possibly multiple sizes) lists a logic function equivalent to a 17-input AND gate. When the designer wishes to use an opamp, he uses Boolean logic for a 17-input AND operation in Verilog. The synthesizer finds in the LIB information that the opamp cell “performs” this function. There are three types of errors that may occur are summarized as follows:

Risk #1:

The designer uses the “logic” function for the custom analog cell, and the synthesizer finds that using a combination of digital standard cells achieves lower power than the custom analog cell. This is avoided by specifying a relatively low power consumption for the custom cell in the LIB file.

Risk #2:

The real digital logic synthesizes to the same function that was specified for the analog custom cell. The probability of this happening is reduced by making the analog “logic” function very complicated. This function can be implemented as a macro to improve the readability of the Verilog code.

Risk #3:

The real digital logic synthesizes to a function that *could* be implemented with the analog cell’s “logic” function. For example, the synthesizer wants a 2-input AND gate, which could be implemented using a 17-input AND gate with 15 pins connected to VDD. Since the power consumption of the analog custom cell was made very low (to reduce Risk #1 above), the synthesizer opts to use the custom cell instead of a real AND gate. This is reduced by specifying a relatively high power consumption for the custom cell.

Clearly, the solutions to Risks #1 and #3 are conflicting and leave the synthesizer in a delicate balance. There is no guarantee that the desired circuit will be synthesized. However, it is a direction worth pursuing in future work, in order to allow the synthesizer to select among different sizes of custom analog cells rather than hardcoding the device sizes.

6.2.3 Analog Cells with Fake Timing Information

The second problem (lack of timing information) causes a problem because the PNR tool has no way to identify which nodes are part of the critical signal path. The same transmission gate cell may sometimes be used in a critical path needing very short routing lengths, and elsewhere in a non-critical path for which routing length is less important. In this work, floorplanning was used to guide the PNR tool towards a more optimal solution. A more robust solution is to provide falsified timing information for analog cells in the LIB file, manipulating the tool into optimizing the routing.

For example, suppose that two versions of a custom cell are created: myCell_A and myCell_B (refer to Figure 3.8 as an example of these cell definitions). The schematic and layout for the cells are identical. For myCell_A, all input pins are described in the LIB file as having relatively large capacitances and slow transition times, meaning that the preceding cell must be routed only a very short distance to satisfy timing requirements. All output pins are described as having slow rise and fall times, meaning that the following cell should also be placed very close. On the other hand, myCell_B is specified to have small pin capacitances (easy to drive) and relaxed timing requirements, meaning that it can be routed a long distance while still satisfying the timing constraints. Clearly myCell_A is ideal for critical paths, like those connecting to the virtual ground node of an amplifier. The alternative myCell_B could be used in sub-blocks for which matching requirements are relaxed.

By itself, this could be used instead of (or in addition to) floorplanning to improve the routing of analog signals in the synthesized ADC. In combination with the other future work described in Section 6.2.2, the goal is to allow the synthesis tool to identify what sizes of analog cells to use (capacitor size, transmission gate size, opamp drive strength) and further instruct the PNR tool which portions of the circuit are most critical for routing.

BIBLIOGRAPHY

[1] G. E. Moore, “Cramming More Components onto Integrated Circuits,” *Electronics*, pp. 114–117, April 1965.

[2] “2013 Overall Roadmap Technology Characteristics (ORTC) Table,” International Technology Roadmap for Semiconductors, 2013.

[3] B. Murmann, P. Nikaeen, D.J. Connelly, and R.W. Dutton, “Impact of Scaling on Analog Performance and Associated Modeling Needs,” *IEEE Trans. Electron Devices*, pp. 2160–2167, Sep. 2006.

[4] “ATmega128 Datasheet: 8-bit Atmel Microcontroller with 128KBytes In-System Programmable Flash,” Atmel Corporation, 2011.

[5] B. Murmann, “ADC Performance Survey 1997-2014,” [Online]. Available: http://web.stanford.edu/∼murmann/adcsurvey.html.

[6] C. Donovan and M.P. Flynn, “Digital Calibration Incorporating Redundancy of Flash ADCs,” *IEEE J. Solid-State Circuits*, vol. 37, no. 3, pp. 432–437, March 2002.

[7] M.P. Flynn, C. Donovan, and L. Sattler, “Digital Calibration Incorporating Redundancy of Flash ADCs,” *IEEE Trans. Circuits Syst. II*, vol. 50, no. 5, pp. 205–213, May 2003.

[8] D. C. Daly and A. P. Chandrakasan, “A 6b 0.2-0.9 V Highly Digital Flash ADC with Comparator Redundancy,” in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2008, pp. 554–555.

[9] S. Weaver, B. Hershberg, and U. Moon, “Stochastic Flash Analog-to-Digital Conversion,” *IEEE Trans. Circuits Syst. I*, vol. 57, no. 11, pp. 2825–2833, Nov. 2010.

[10] P. Dudek, S. Szczepanski, J.V. Hatfield, “A High-Resolution CMOS Time-to-Digital Converter Utilizing a Vernier Delay Line,” *IEEE J. Solid-State Circuits*, vol. 35, no. 2, pp. 240–247, Feb 2000.

[11] T.E. Rahkonen and J.T. Kostamovaara, “The Use of Stabilized CMOS Delay Lines for the Digitization of Short Time Intervals,” *IEEE J. Solid-State Circuits*, vol. 28, no. 8, pp. 887–894, Aug 1993.

[12] T. Watanabe, T. Mizuno, and Y. Makino, “An All-Digital Analog-to-Digital Converter With 12-μV/LSB Using Moving-Average Filtering,” *IEEE J. Solid-State Circuits*, vol. 38, no. 1, pp. 120–125, Jan 2003.

[13] M.A. Farahat, F.A. Farag, and H.A. Elsimary, "Only Digital Technology Analog-to-Digital Converter Circuit," in *Circuits and Systems, 2003 IEEE 46th Midwest Symposium on*, Dec 2003, pp. 178–181.

[14] H. Farkhani, M. Meymandi-Nejad, and M. Sachdev, "A Fully Digital ADC Using a New Delay Element with Enhanced Linearity," in *Proc. IEEE Int. Symp. on Circuits and Syst.*, May 2008, pp. 2406–2409.

[15] M. Negreiros, L. Carro, and G. Cassel, "All Digital ADC with Linearity Correction and Temperature Compensation," in *Instrumentation and Measurement Technology Conference (I2MTC), 2010 IEEE*, May 2010, pp. 147–152.

[16] M. Straayer and M. Perrott, "A Multi-Path Gated Ring Oscillator TDC With First-Order Noise Shaping," *IEEE J. Solid-State Circuits*, vol. 44, no. 4, pp. 1089–1098, Apr. 2009.

[17] A. Elshazly, S. Rao, B. Young, and P.K. Hanumolu, "A Noise-Shaping Time-to-Digital Converter Using Switched-Ring Oscillators — Analysis, Design, and Measurement Techniques," *IEEE J. Solid-State Circuits*, vol. 49, no. 5, pp. 1184–1197, May 2014.

[18] M. Straayer and M. Perrott, "A 12-Bit, 10-MHz Bandwidth, Continuous-Time $\Sigma\Delta$ ADC With a 5-Bit, 950-MS/s VCO-Based Quantizer," *IEEE J. Solid-State Circuits*, vol. 43, no. 4, pp. 805–814, Apr. 2008.

[19] M. Park and M. Perrott, "A 78 dB SNDR 87 mW 20 MHz Bandwidth Continuous-Time $\Delta\Sigma$ ADC With VCO-Based Integrator and Quantizer Implemented in 0.13 μm CMOS," *IEEE J. Solid-State Circuits*, vol. 44, no. 12, pp. 3344–3358, Dec. 2009.

[20] G. Taylor and I. Galton, "A Mostly Digital Variable-Rate Continuous-Time ADC $\Delta\Sigma$ Modulator," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2010, pp. 298–299.

[21] G. Taylor and I. Galton, "A Mostly Digital Variable-Rate Continuous-Time Delta-Sigma Modulator ADC," *IEEE J. Solid-State Circuits*, vol. 45, no. 12, pp. 2634–2646, Dec. 2010.

[22] S. Weaver, B. Hershberg, and U. Moon, "Digitally Synthesized Stochastic Flash ADC Using Only Digital Standard Cells," in *Symp. VLSI Circuits Dig. Tech. Papers*, Jun. 2011, pp. 266–267.

[23] S. Weaver, *Automated Synthesis of Analog to Digital Conversion*, Ph.D. thesis, Oregon State University, 2010.

[24] E.S. Ochotta, T. Mukherjee, R.A. Rutenbar, and L.R. Carley, *Practical Synthesis of High-Performance Analog Circuits*, Kluwer Academic Publishers, Norwell, MA, first edition, 1998.

[25] J.M. Cohn, D.J. Garrod, R.A. Rutenbar, and L.R. Carley, "KOAN/ANAGRAM II: New Tools for Device-Level Analog Placement and Routing," *IEEE J. Solid-State Circuits*, vol. 26, no. 3, pp. 330–342, Mar 1991.

[26] S. Mitra, S.K. Nag, R.A. Rutenbar, and L.R. Carley, "System-Level Routing of Mixed-Signal ASICs in WREN," in *Computer-Aided Design, 1992. ICCAD-92. Digest of Technical Papers., 1992 IEEE/ACM International Conference on*, Nov 1992, pp. 394–399.

[27] Z. Navabi, *Verilog Digital System Design: Register Transfer Level Synthesis, Testbench, and Verification*, McGraw-Hill, New York, NY, second edition, 2006.

[28] "GDSII Stream Format Manual (Revision 6.0)," Calma Company, Feb. 1987.

[29] "LEF/DEF Language Reference (Revision 5.7)," Cadence Design Systems, Nov. 2009.

[30] "Timing Library Format Reference (Revision 4.3)," Cadence Design Systems, Oct. 2000, [Online]. Available: http://www.ee.virginia.edu/~mrs8n/soc/SynthesisTutorials/ct_tlfref.pdf.

[31] "Liberty User Guides and Reference Manual Suite (Version 2013.03)," Open Source Liberty, 2013, [Online]. Available: http://www.eecs.berkeley.edu/~alanmi/publications/other/liberty13_03.pdf.

[32] "Abstract Editor Reference (Revision 5.0)," Cadence Design Systems, July 2002.

[33] "Design Compiler User Guide (Version C-2009.06)," Synopsys Incorporated, June 2009, Chapter 8: Optimizing Your Design.

[34] "Encounter User Guide (Version 4.1.5)," Cadence Design Systems, May 2005, Chapter 8: Floorplanning the Design.

[35] "ModelSim User's Manual (Version 10.1c)," Mentor Graphics Incorporated, 2012.

[36] "Spectre Circuit Simulator User Guide (Version 5.0)," Cadence Design Systems, January 2004, Chapter 1: Introducing the Spectre Circuit Simulator.

[37] "Assura Physical Verication User Guide (Revision 3.1.7)," Cadence Design Systems, Feb. 2008, Chapter 1: Introduction to Assura Physical Verification.

[38] "Design Data Translator's Reference (Version 5.0)," Cadence Design Systems, July 2002, Chapter 5: Translating CDL Files.

[39] D.A. Johns and K. Martin, *Analog Integrated Circuit Design*, John Wiley and Sons, New York, NY, first edition, 1997.

[40] R. Schreier and G.C. Temes, *Understanding Delta-Sigma Data Converters*, John Wiley and Sons, Inc, Hoboken, NJ, first edition, 2005.

[41] K.C.H. Chao, S. Nadeem, W.L. Lee, and C.G. Sodini, "A Higher Order Topology for Interpolative Modulators for Oversampling A/D Converters," *IEEE Trans. Circuits Syst.*, vol. 37, no. 3, pp. 309–318, Mar 1990.

[42] T. Ritoniemi, T. Karema, and H. Tenhunen, "Design of Stable High Order 1-bit Sigma-Delta Modulators," in *Proc. IEEE Int. Symp. on Circuits and Syst.*, May 1990, pp. 3267–3270.

[43] O. Rajaee, T. Musah, N. Maghari, S. Takeuchi, M. Aniya, K. Hamashita, U. Moon, "Design of a 79 dB 80 MHz 8X-OSR Hybrid Delta-Sigma/Pipelined ADC," *IEEE J. Solid-State Circuits*, vol. 45, no. 4, pp. 719–730, April 2010.

[44] T. Hayashi, Y. Inabe, K. Uchimura, and T. Kimura, "A Multistage Delta-Sigma Modulator without Double Integration Loop," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb 1986, pp. 182–183.

[45] Y. Matsuya, K. Uchimura, A. Iwata, T. Kobayashi, M. Ishikawa, and T. Yoshitome, "A 16-bit Oversampling A-to-D Conversion Technology Using Triple-Integration Noise Shaping," *IEEE J. Solid-State Circuits*, vol. 22, no. 6, pp. 921–929, Dec 1987.

[46] B. Razavi, *Design of Analog CMOS Integrated Circuits*, McGraw-Hill, New York, NY, first edition, 2000.

[47] I. Fujimori, L. Longo, A. Hairapetian, K. Seiyama, S. Kosic, J. Cao, and S. Chan, "A 90-dB SNR 2.5-MHz Output-Rate ADC Using Cascaded Multibit Delta-Sigma Modulation at 8$\times$ Oversampling Ratio," *IEEE J. Solid-State Circuits*, vol. 35, no. 12, pp. 1820–1828, Dec 2000.

[48] J. Kim, B.S. Leibowitz, J. Ren, and C.J. Madden, "Simulation and Analysis of Random Decision Errors in Clocked Comparators," *IEEE Trans. Circuits Syst. I*, vol. 56, no. 8, pp. 1844–1857, Aug 2009.

[49] B. Hershberg, S. Weaver, K. Sobue, S. Takeuchi, K. Hamashita, and U. Moon, "Ring Amplifier for Switched-Capacitor Circuits," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb 2012, pp. 460–461.

[50] B. Hershberg, S. Weaver, K. Sobue, S. Takeuchi, K. Hamashita, and U. Moon, "A 61.5dB SNDR Pipelined ADC Using Simple Highly-Scalable Ring Amplifiers," in *Symp. VLSI Circuits Dig. Tech. Papers*, June 2012, pp. 32–33.

[51] B. Hershberg and U. Moon, "A 75.9dB-SNDR 2.96mW 29fJ/conv-step Ringamp-Only Pipelined ADC," in *Symp. VLSI Circuits Dig. Tech. Papers*, June 2013, pp. 94–95.

[52] Y. Lim and M.P. Flynn, "A 100MS/s 10.5b 2.46mW Comparator-less Pipeline ADC Using Self-Biased Ring Amplifiers," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb 2014, pp. 202–203.

[53] B. Hershberg, *Ring Amplification for Switched Capacitor Circuits*, Ph.D. thesis, Oregon State University, 2012.

[54] B. Hershberg, S. Weaver, K. Sobue, S. Takeuchi, K Hamashita, and U. Moon, "Ring Amplifiers for Switched Capacitor Circuits," *IEEE J. Solid-State Circuits*, vol. 47, no. 12, pp. 2928–2942, Dec 2012.

APPENDICES

A Finite Gain in Switched-Capacitor Integrators

Consider the integrator in Figure A.1. The output voltage is determined by:

$$V = A \cdot (0 - X) = -A \cdot X \tag{A.1}$$

and the voltage across capacitor C_2 (denoted V_2) is given by:

$$\begin{aligned} V_2 &= V - X = -A \cdot X - X \\ &= -X(A+1) \end{aligned} \tag{A.2}$$

During clock phase Φ_1, the input is sampled as shown in Figure A.2a. Let this be time $N-1$ in discrete time. The charge across the two capacitors at the end of this phase are:

$$\begin{aligned} q_{1,1} &= C_1 \cdot V_1 \\ &= C_1 \cdot U[N-1] \\ q_{2,1} &= C_2 \cdot V_2 \\ &= -C_2 \cdot (A+1) \cdot X[N-1] \end{aligned} \tag{A.3}$$

for a total charge of:

$$\begin{aligned} Q_{TOT,1} &= q_{1,1} + q_{2,1} \\ &= C_1 \cdot U[N-1] - C_2 \cdot (A+1) \cdot X[N-1] \end{aligned} \tag{A.4}$$

During phase Φ_2, the sampled input voltage is integrated and the charges

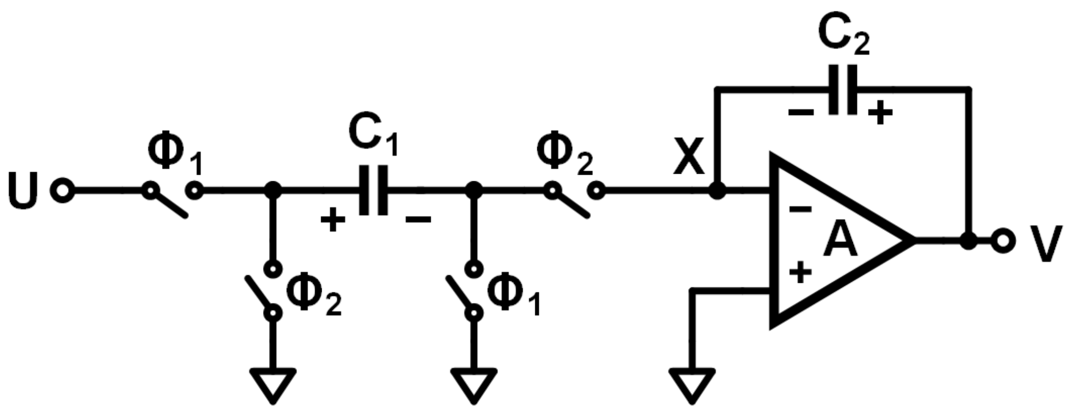


Figure A.1: Switched-capacitor integrator with a finite-gain amplifier.

become:

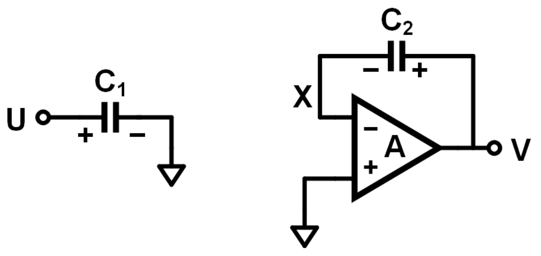
$$\begin{aligned} q_{1,2} &= C_1 \cdot (0 - X[N]) \\ &= -C_1 \cdot X[N] \\ q_{2,2} &= -C_2 \cdot (A+1) \cdot X[N] \end{aligned} \tag{A.5}$$

for a total charge of:

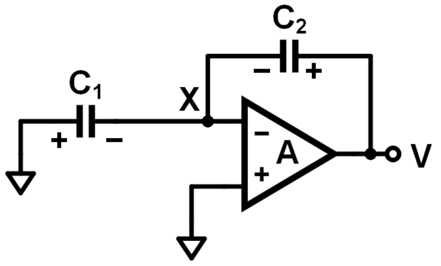
$$Q_{TOT,2} = -C_1 \cdot X[N] - C_2 \cdot (A+1) \cdot X[N] \tag{A.6}$$

Due to conservation of charge during the integration, the total charges in (A.4) and (A.6) must be equal. The two are equated:

$$\begin{aligned} Q_{TOT,2} &= Q_{TOT,1} \\ -C_1 \cdot X[N] - C_2 \cdot (A+1) \cdot X[N] &= C_1 \cdot U[N-1] - C_2 \cdot (A+1) \cdot X[N-1] \\ X[N] \cdot \{-C_1 - C_2 \cdot (A+1)\} &= C_1 \cdot U[N-1] - C_2 \cdot (A+1) \cdot X[N-1] \end{aligned} \tag{A.7}$$



(a)



(b)

Figure A.2: Switched-capacitor integrator during (a) phase 1 and (b) phase 2.

and then solved for $X[N]$:

$$\begin{aligned} X[N] &= \frac{C_1 \cdot U[N-1] - C_2 \cdot (A+1) \cdot X[N-1]}{-C_1 - C_2 \cdot (A+1)} \\ &= \frac{C_2 \cdot (A+1) \cdot X[N-1] - C_1 \cdot U[N-1]}{C_2 \cdot (A+1) + C_1} \end{aligned} \tag{A.8}$$

Substituting this result into (A.2):

$$\begin{aligned} V[N] &= -A \cdot X[N] \\ &= -A \cdot \left\{ \frac{C_2 \cdot (A+1) \cdot X[N-1] - C_1 \cdot U[N-1]}{C_2 \cdot (A+1) + C_1} \right\} \\ &= V[N-1] \left(\frac{(A+1)C_2}{(A+1)C_2 + C_1} \right) + U[N-1] \left(\frac{A \cdot C_1}{(A+1)C_2 + C_1} \right) \end{aligned} \tag{A.9}$$

Assuming that $C_1 = C_2$:

$$V[N] = V[N-1]\left(\frac{A+1}{A+2}\right) + U[N-1]\left(\frac{A}{A+2}\right) \tag{A.10}$$

which can be represented in the z-domain as:

$$\begin{aligned} V(z) &= V(z)\cdot z^{-1}\left(\frac{A+1}{A+2}\right) + U(z)\cdot z^{-1}\left(\frac{A}{A+2}\right) \\ V(z)\left\{1-\left(\frac{A+1}{A+2}\right)\cdot z^{-1}\right\} &= U(z)\cdot\left(\frac{A}{A+2}\right)\cdot z^{-1} \\ V(z) &= U(z)\left\{\frac{\left(\frac{A}{A+2}\right)\cdot z^{-1}}{1-\left(\frac{A+1}{A+2}\right)\cdot z^{-1}}\right\} \end{aligned} \tag{A.11}$$

This results in the final transfer function $H(z)$ below:

$$H(z) = \frac{\left(\frac{A}{A+2}\right)\cdot z^{-1}}{1-\left(\frac{A+1}{A+2}\right)\cdot z^{-1}} \tag{A.12}$$

Commonly, people makes the approximation that for very large A, the charge in (A.5) becomes:

$$q_{1,2} = -C_1 \cdot X[N] = C_1 \cdot \frac{V[N]}{A} \approx 0 \tag{A.13}$$

which (continuing through the derivation) results in an approximation of the transfer:

$$H(z) \approx \frac{z^{-1}}{1-\left(\frac{A}{A+1}\right)\cdot z^{-1}} \tag{A.14}$$

and is sometimes further approximated as:

$$H(z) \approx \frac{z^{-1}}{1-\left(1-\frac{1}{A}\right)\cdot z^{-1}} = \frac{z^{-1}}{1-\alpha\cdot z^{-1}} \tag{A.15}$$

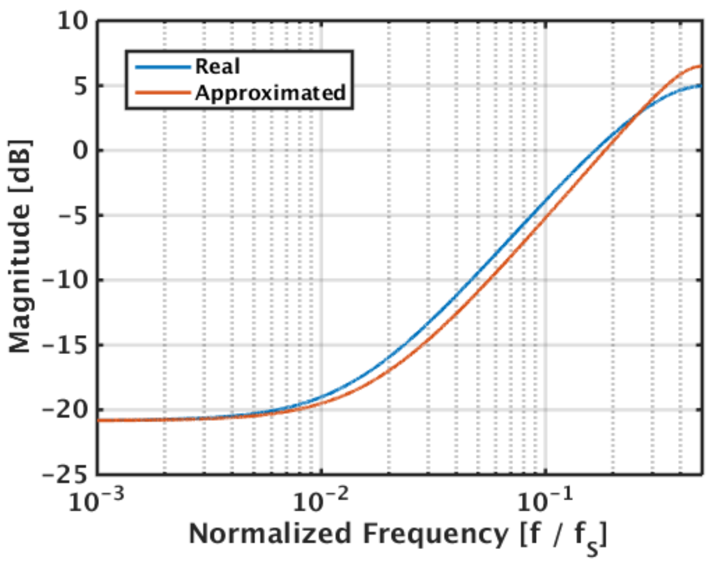


Figure A.3: Deviation in first-order NTF using finite gain approximation, for an amplifier with 20dB DC gain.

The approximated transfer function deviates from the real $H(z)$ as shown in Figure A.3. The resulting reduction in theoretical SQNR is shown in Figure A.4. As the opamp gain increases, there is less error due to the approximated transfer function. For these high gain amplifier the reduction only becomes significant at high OSR, at which point the theoretical SQNR is extremely high and would be limited by factors such as thermal noise or component mismatch instead. For extremely low-gain amplifiers the SQNR quickly becomes limited by finite gain and the error in the transfer function becomes irrelevant.

On the other hand, for the 30dB amplifier the maximum SQNR with $OSR = 32$ is reduced from 82dB to 68dB by the approximation. After accounting for thermal noise and mismatch, this could certainly affect the final SNDR for the circuit. The results demonstrate that for the low-gain single-stage opamps used in this work, the common approximation of the finite-gain integrator transfer function

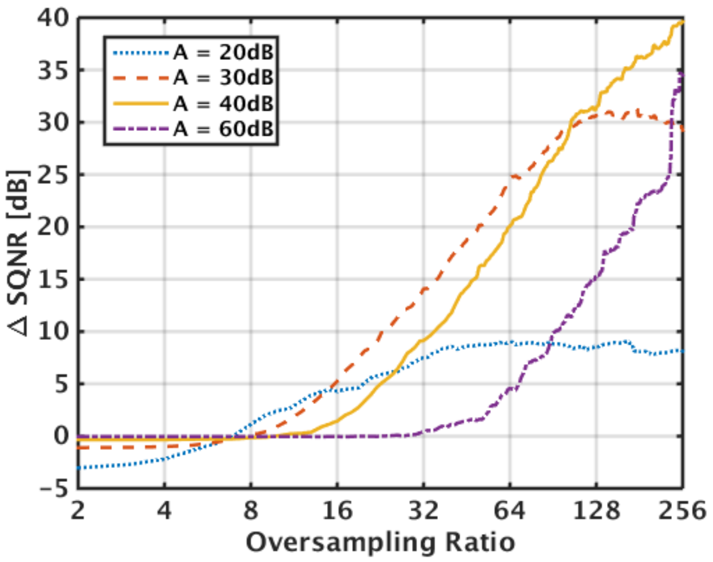


Figure A.4: Reduction in theoretical SQNR from approximating $H(z)$ for a third-order DSM.

in (A.15) will cause issues in digital calibration that will limit the maximum SQNR below the target resolution. The more accurately derived result in (A.12) must be used instead.

B List of Acronyms

The work presented in this thesis relies on acronyms for software tools, library file names, and mixed-signal circuit concepts. This list of acronyms is provided in the hope that it will reduce confusion for the reader.

ADC - analog-to-digital converter

ASTRX

A design automation tool for device sizing in analog circuits, created at Carnegie Mellon University.

BW - bandwidth

CDB - Cadence DataBase file

This file contains the layout view in a format used by the Virtuoso layout editor.

CDF - cumulative distribution function

CDL - Circuit Description Language

A SPICE-like syntax that is sometimes used for digital standard cell libraries, instead of schematics.

CLS - correlated level-shifting

CMFB - common-mode feedback

CMOS - complementary metal-oxide-semiconductor

DAC - digital-to-analog converter

DB - database file

A variation of the LIB file.

DC - Design Compiler

A logic synthesizer software tool provided by Synopsys.

DFII - Design Framework II

A suite of Cadence software tools used for design, simulation, and verification.

DRC - design rules check

Verification that the layout satisfies physical rules for manufacturing.

DSM - delta-sigma modulator

DZ - deadzone

ENOB - effective number of bits

FET - field effect transistor

FFT - fast Fourier transform

GDS - Graphic Database System

A binary file describing the layout of an integrated circuit.

GRO - gated ring oscillator

IC - integrated circuit

ICFB - Integrated Circuit Front-to-Back

Cadence's software tool that provides access to schematic and layout editors.

ISSCC - International Solid-State Circuits Conference

KOAN

A design automation tool for device-level placement and routing, created at Carnegie Mellon University.

LEF - Library Exchange Format

Contains library information related to routing.

LIB - Liberty Timing File

Containing library information related to function, area, timing and power consumption.

LSB - least-significant bit

LUT - look-up table

LVS - layout versus schematic

Verification that the circuit layout matched the schematic.

MASH - Multi-stAge noise SHaping

MDAC - multiplying DAC

MIM - metal-insulator-metal capacitor

NS - noise shaping

NTF - noise transfer function

OSR - oversampling ratio

PEX - parasitic extraction

PNR - place and route

PSD - power spectral density

RAMP - ring amplifier

RCX - resistor/capacitor extraction

RTL - register-transfer level

Process-independent Verilog code that uses logical and Boolean expressions.

SCR - script

A scripting language used by Design Compiler

SFDR - spurious-free dynamic range

SNDR - signal-to-noise-and-distortion ratio

SNR - signal-to-noise ratio

SPICE - Simulation Program with Integrated Circuit Emphasis

A common analog circuit simulation program.

SQNR - signal-to-quantization-noise ratio

SRO - switched ring oscillator

STF - signal transfer function

TCL - Tool Command Language

A scripting language used by the Encounter PNR software tool.

TDC - time-to-digital converter

TLF - Timing Library Format

A variation of the LIB file.

TSMC - Taiwan Semiconductor Manufacturing Company

VCO - voltage-controlled oscillator

VTC - voltage-to-time converter

WREN

A design automation tool for system-level routing, created at Carnegie Mellon University.