AN ABSTRACT OF THE DISSERTATION OF

Skyler Weaver for the degree of Doctor of Philosophy in
Electrical and Computer Engineering presented on September 15, 2010.
Title: Automated Synthesis of Analog to Digital Conversion.

Abstract approved: _____

Un-Ku Moon

This thesis describes circuit architectures and techniques that facilitate the automatic synthesis and fabrication of analog-to-digital converters (ADCs). Since automated synthesis already exists for digital circuits and is part of the digital circuit design flow, this work demonstrates the feasibility of ADC synthesis with little or no modification to presently existing software tools. In the end, it is demonstrated that an ADC can be implemented from synthesizable Verilog code, making it highly portable from one process technology to another. Moreover, by demonstrating how to use existing standard digital gates to generate analog functions (e.g. an analog comparator), the physical implementation of an ADC can be as automatic and straightforward as a standard digital circuit.

Automated Synthesis of Analog to Digital Conversion

by

Skyler Weaver

A DISSERTATION

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

Presented September 15, 2010
Commencement June 2011

Doctor of Philosophy dissertation of Skyler Weaver presented on September 15, 2010

APPROVED:

Major Professor, representing Electrical and Computer Engineering

---

Director of the School of Electrical Engineering and Computer Science

---

Dean of the Graduate School

I understand that my dissertation will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my dissertation to any reader upon request.

Skyler Weaver, Author

ACKNOWLEDGMENTS

First of all, I would like to thank Professor Un-Ku Moon for being my advisor and helping me become a circuit master. His guidance and wisdom in the realm of analog circuits and life in general is forever appreciated. As I embark into my future career, it feels good to know that I have been equipped with the tools I will need on my journey, and I acknowledge that it was Dr. Moon that helped me discover my own stoneshiner.

I would also like to thank Professors Pavan Kumar Hanumolu, Karti Mayaram, and Gabor Temes for their advice and instruction. Their expert council have kept me thinking in the right direction regarding research. My success is in part their responsibility.

My fellow group members have all been good friends and colleagues, especially Benjamin Hershberg, Peter Kurahashi, and Nima Maghari. Being able to bounce ideas off each other and share a laugh has been a wonderful support. Our lunchtime brainstorming sessions were always enjoyable, even if we spent most of our time coming up with clever acronyms and ways to get to the moon.

My parents and family have always given me their love and have always supported and encouraged me in my passion for knowledge. I know they are proud of me for adding this work to human knowledge and unlocking some of the secrets of the Universe.

Lastly, I am grateful for Shauna, my wife. Her unquestioning support and love is always there: whether bringing me snacks and energy drinks for late nights in the lab, or accompanying me on transcontinental trips to conferences. I am truly lucky to have found someone whose companionship I cherish and whom I love so much.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

Page

LIST OF FIGURES

LIST OF FIGURES (Continued)

Figure Page

LIST OF FIGURES (Continued)

Figure Page

LIST OF FIGURES (Continued)

Figure | Page

LIST OF TABLES

To myself.

I really couldn't have done it without me.

AUTOMATED SYNTHESIS OF ANALOG TO DIGITAL CONVERSION

CHAPTER 1. INTRODUCTION

The invention of CMOS integrated circuits has led to the creation of technology only dreamed of a few decades ago. Consider the technology found on the original series of *Star Trek*, which at the time represented what was imagined to eventually be the pinnacle of human achievement. Today much of this technology has been surpassed and considered commonplace. For example, Captain Kirk would gladly trade his bulky communicator in for an iPhone, and Uhura's earpiece looks clunky next to an off-the-shelf Bluetooth headset. Integrated circuits make this technology possible; device scaling and synthesis that makes each generation of technology better and cheaper.

In digital circuit design there is a fairly straightforward trade-off between power and speed. For a digital gate to switch faster, it must source more current to its output load, i.e. the MOS gates of the successive digital block. Thus, the power-delay product is a good metric to compare digital circuits. By scaling the physical dimensions of transistors to smaller geometries, there is a innate reduction in the power-delay product in addition to an overall area reduction for a given digital design [1]. By merely shrinking the size of transistors, a circuit consumes less power for the same speed or it can achieve higher speed for the same power. In effect, scaling digital circuits is a win-win situation. There are certain reliability issues with scaled transistors that can cause transistors to fail [2]–[4], but

these issues can be remedied by changing process parameters and innovative device engineering [5][6].

Since there are practically only benefits to porting a digital design to a scaled-down process, we can expect this trend to continue [7][8]. Due to the low sensitivity of digital circuits to their physical layout, they are usually synthesized: that is, their physical layout is automatically generated. This is a huge benefit in the form of dramatically reduced design cost. Digital synthesis allows designers to describe digital circuits by writing code in either VHDL or Verilog programming languages and processing it through automation tools. The digital design can then be effortlessly ported to the next generation process by simply reprocessing already existing VHDL or Verilog code.

So far this is nothing but good news. All digital circuits must eventually interface to our analog world by the use of analog-to-digital conversion (ADC) and digital-to-analog conversion (DAC) circuits. To get the most out of higher performing digital circuits, there is a need for ADCs and DACs that can be easily integrated on scaled technology. Although scaling inherently improves digital circuits, scaling analog circuits tends to gives increased speed yet decreased gain and not necessarily lower power or area [9]. This means that every time an analog circuit is ported to a new process it must be meticulously and expensively redesigned. Moreover, classic analog circuits are very sensitive to their physical layout making synthesis practically impossible. Because of this, as scaling trends continue, the analog portion of mixed-signal systems tend to consume proportionally more power, area, and have a higher design cost than the digital counterparts. This thesis will address how to use existing digital synthesis tools to synthesize analog circuits and describe new ADC architectures that are synthesizable and portable to future processes.

CHAPTER 2. ANALOG CIRCUIT SYNTHESIS USING STANDARD DIGITAL SYNTHESIS TOOLS

Large digital circuits can be very complex, so they are frequently synthesized; that is, many of the circuit specifics and the physical layout are automatically generated by computer software. This allows the digital circuit to be designed and simulated in a very high-level way. Imagine trying to give driving directions to your house to a robot that only understands simple commands such as "turn the steering wheel to the left six degrees," and "increase the pressure on the accelerator by two percent." Trying to describe "drive down the interstate for 56 miles and take exit 231" using only these simple commands would be very tedious and it would be very easy to make a catastrophic mistake; however, if there is a deterministic way to convert a complex command into simple commands, software can be created that does the conversion. This the idea behind digital synthesis: it frees a designer from coding each individual digital logic gate to describe a circuit in a human-readable way.

A diagram of the standard digital circuit design flow can be seen in Fig. 2.1. A digital circuit is first described at register transfer level (RTL) in a hardware description language such as Verilog or VHDL. At this high level, the code very much resembles the C programming language. Once the digital circuit is described in RTL Verilog, it can be simulated to verify its functionality. The semiconductor foundry that will fabricate the physical circuit will provide a digital library of the standard digital cells. All digital libraries tend to have the same functional blocks, but the size, speed, and area of each block depends on the fabrication process. Once

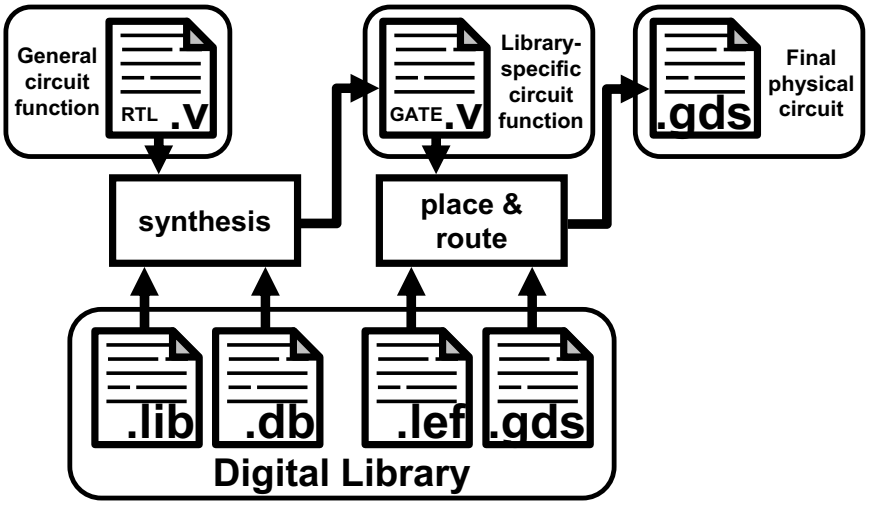


Figure 2.1: The digital circuit synthesis design flow.

the digital library has been obtained, the synthesis tool will take the RTL code and translate it into gate-level code that is optimized and specific to the process. The gate-level code can also be simulated to verify that the functionality is still correct. The final step to run the gate-level code through the place-and-route tool, which optimally places the digital blocks, physically, and generates the wiring to connect all of the blocks. This final output file from the place-and-route tool can also be simulated and will ultimately be constructed into a physical circuit.

As an example, see Fig. 2.2. This is a piece of RTL Verilog that is human-readable and takes in three inputs, A, B, and C. The output Z is true when two or more of the inputs are true (i.e. $Z = (A\&B)|(B\&C)|(A\&C)$). There is no standard digital cell that performs this function, so the synthesizer will need to create the same digital function out of simpler logic gates that are available in the digital library. Typically there are two types of files in the digital library: timing information (`.lib`, `.tlf`, or `.db` files) and physical information (`.lef` and `.gds` files). The synthesizer is only interested in the timing information which will have

```
1  module circuit(A, B, C, Z);
2   output Z;
3   input A, B, C;
4
5   assign Z = (A&B)|(B&C)|(A&C);
6
7  endmodule
```

Figure 2.2: RTL Verilog code.

```
1  module circuit(A, B, C, Z);
2   output Z;
3   input A, B, C;
4
5   wire n1, n2, n3;
6   nand2x2 U1( .A(A), .B(B), .Y(n1) );
7   nand2x2 U2( .A(B), .B(C), .Y(n2) );
8   nand2x2 U3( .A(A), .B(C), .Y(n3) );
9   nand3x2 U4( .A(A), .B(B), .C(C), .Y(Z) );
10
11 endmodule
```

```
1  module circuit(A, B, C, Z);
2   output Z;
3   input A, B, C;
4
5   wire n1, n2;
6   nand2x2 U1( .A(A), .B(B), .Y(n1) );
7   nor2x2 U2( .A(A), .B(B), .Y(n2) );
8   muxix2 U3( .IN0(n1), .IN1(n2), .SELECT(C), .Y(Z) );
9
10 endmodule
```

Figure 2.3: Gate-level Verilog code of two functionally equivalent digital circuits.

a list of all of the digital cells and their associated power, area, and speed. The designer can instruct the synthesizer to minimize any of these or to try and meet specific goals regarding the power, area, and speed. Fig. 2.3 is an example of two possible gate-level Verilog code outputs from the synthesizer. Both have the same digital function, so the synthesizer would choose a solution that optimally meets the targeted requirements.

There are two main advantages of synthesizing a digital circuit. First, synthesis allows a designer to spend more time on designing the function of the circuit instead of tediously optimizing the circuit at the gate level. Second, the RTL code can be re-synthesized onto a different process with relative ease, allowing a quick turnaround time when porting a circuit to a new technology.

Currently there are no automatic synthesis tools for analog circuits that are as proven and ubiquitous as conventional synthesis is for digital circuits. This is due to the fact that digital circuits are not very sensitive to their physical layout. Digital circuits are generally only sensitive to their timing. Many people have claimed to have created an "all-digital" ADC [10]–[13], but at the time of writing this thesis there has been no successful attempt at synthesizing an ADC using standard digital synthesis tools. The key limiting factor is that an ADC requires a circuit block that has an analog input and a digital output, and incorporating this into the constraints of the standard digital cell design space is nontrival.

2.1 Using Custom Analog Cells

One way to incorporate analog functionality into a digitally synthesized circuit is to create a custom "digital" cell and add it to the standard cell library. There is no way for the synthesis and place-and-route tools to distinguish between a custom analog cell and a standard digital cell. In order for a custom cell to be placed into the final circuit it must be added to the library (timing and physical) and then instantiated explicitly.

In order to add the physical layout of a custom analog cell to the digital library, the cell must first be designed and simulated and laid out in such a way that it matches certain characteristics of the other cells in the library. All digital

cells in the library have the same supply pitch and usually have all routing on a single metal layer, the closest to the substrate. In effect, all of the cells have the same height (set by the supply pitch) but can have varying widths. In order to make a custom cell match, it is easiest to start with the physical layout of a cell that is already in the library and alter it to make a custom cell. Once the physical layout is complete it can be added to the library. One section of the `.lef` file lists the input and output pin names, their location, and size with respect to the cell origin. If this is not set properly, the tool will not route to the cell correctly.

Once the physical layout has been added, the timing information needs to be added to the timing files as well or it will not be recognized by the synthesizer. Precaution must be taken when using a custom cell, because it requires "tricking" the synthesizer. In the library timing file, for each cell there is a description of its digital function and a list of the time delays from each of its inputs to each of its outputs. In the case of an analog cell where the input is analog, there is no way to describe the function in a way that the synthesizer will understand. It would be acceptable to use dummy data copied from another standard cell to make an entry into the library to at least make the custom cell valid; however, it is paramount that the function and timing entries make the cell appear to be a poor choice in implementing any digital circuit. For example, if a custom analog cell was created with two inputs and one output and the timing information was copied from a NAND gate cell, there is a risk that the synthesizer may instantiate the custom cell in a place where it needs to use a NAND gate in the digital portion of the circuit. Since the custom analog cell is not actually a NAND gate, the result will not work; however, the gate-level Verilog simulation would *verify that the circuit works.* This is because the digital simulation would use the timing and functional information from the copied NAND gate in order to simulate. To be safe, it is

```
1  module myCell(A, B, Z);
2   output Z;
3   input A, B;
4
5   customCell U1( .A(A), .B(B), .Z(Z) );
6
7  endmodule
```

Figure 2.4: Example instantiation of a custom cell 'customCell' into a Verilog module 'myCell.'

wise to use the `set_dont_use` *`cell_name`* command. This synthesizer directive will inform the synthesis tool that it should not select the cell *`cell_name`* from the library while synthesizing. This command will not remove the cell from the entire design if the cell is already part of a module that the tool will not synthesize (e.g. it was already synthesized or hand-designed).

With a custom analog cell added to the library physically, and functionally designed to prevent the synthesizer from choosing it automatically, the final step is to explicitly instantiate the cell and force the synthesizer to allow the cell to remain into the final circuit. This is done by calling the custom library cell by name in the RTL Verilog exactly how it will appear in the gate-level Verilog code. An example of this can be seen in Fig. 2.4. The example is of a custom cell that has been added to the library named *`customCell`* and is placed inside a module called *`myCell`* with the same number of inputs and outputs. The reason this is done is that when the synthesizer attempts to synthesize the circuit it will try and replace *`customCell`* with a cell that the synthesizer decides is "more optimum" based off of what is listed in the timing file. To avoid this, the command `set_dont_touch` *`myCell`* will direct the synthesizer to not optimize the internals of the module *`myCell`* in any way. This command must be given to both the synthesizer and the place-and-route tool. Note that `set_dont_touch` supersedes `set_dont_use`.

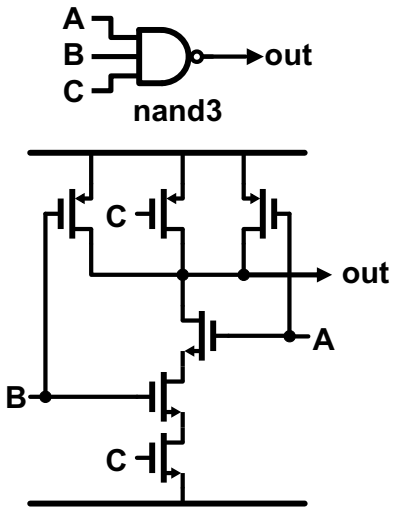


Figure 2.5: A standard digital CMOS NAND3 gate and its internal transistor schematic.

The final important detail about using a custom analog cell is that since some or all of the input and output nets are analog, the synthesizer command `dont_touch_network` *`netname`*, where *`netname`* is the name of the analog net, is required. Without this command the software tool may decide that the net has too high of a fanout or fanin and place digital buffers between the signal and the load. As digital buffers block analog signals, this command is important and must be given to both the synthesizer and the place-and-route tool.

2.2 Using Standard Cells for Analog Circuits

An even more elegant solution to using custom analog cells is to use the standard digital cells that come with the library to generate analog functions. In order to see how this is possible, look at the internal transistor implementation of a standard 3-input NAND (or NAND3) digital gate (Fig. 2.5). This circuit is a

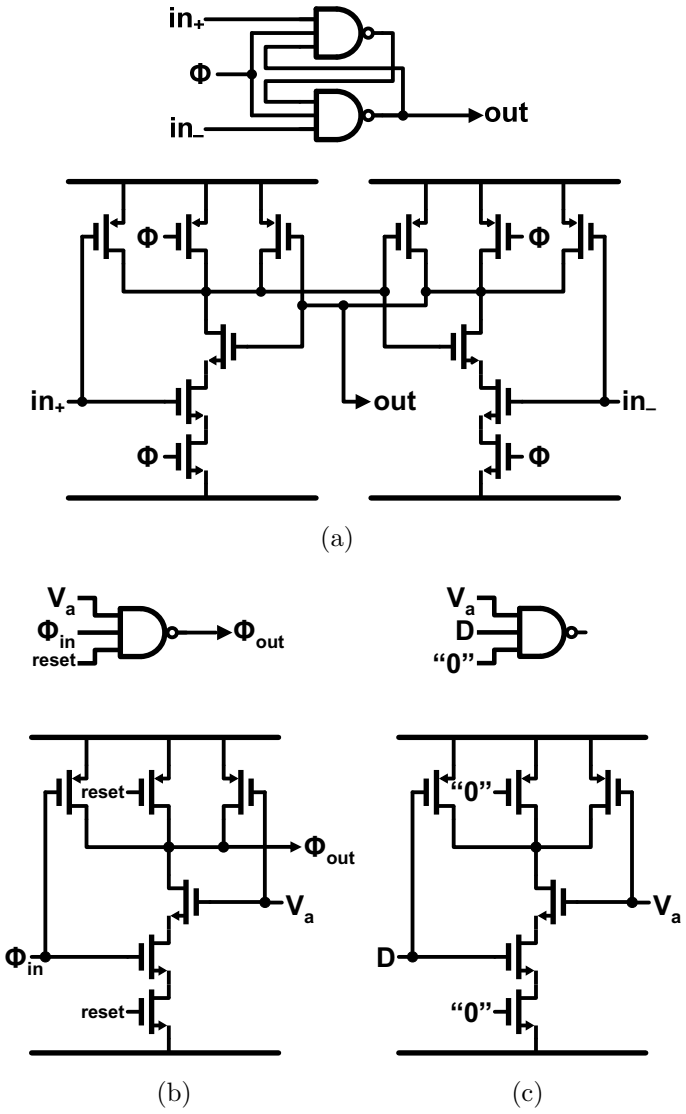


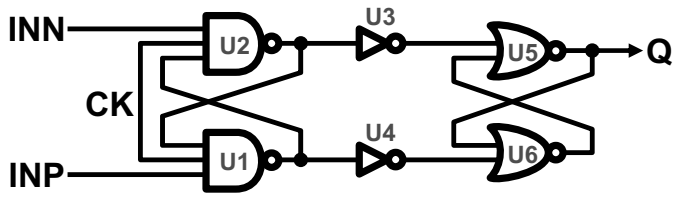
Figure 2.6: a) An analog comparator made from standard digital NAND3 cells. b) A analog voltage controlled delay cell with asynchronous reset using a standard digital NAND3 cell. c) A standard digital NAND3 cell used as a DAC cell.

digital logic gate performing a NAND function for three inputs, or $out = \overline{A\&B\&C}$, when the inputs are digital rail-to-rail inputs.

Upon observation, the schematic of the transistors inside a CMOS NAND3 gate closely resembles half of a clocked analog comparator. By cleverly connecting

```
1  module comparator(INP, INN, CK, Q);
2   output Q;
3   input INP, INN, CK;
4
5   wire op, on, opn, onn, qn;
6   nand3x1 U1 ( .A(op), .B(INP), .C(CK), .Y(on) );
7   nand3x1 U2 ( .A(on), .B(INN), .C(CK), .Y(op) );
8   invx1 U3 ( .A(op), .Y(opn) );
9   invx1 U4 ( .A(on), .Y(onn) );
10  nor2x2 U5 ( .A(qn), .B(opn), .Y(Q) );
11  nor2x2 U6 ( .A(Q), .B(onn), .Y(qn) );
12
13  endmodule
```

(a)



(b)

Figure 2.7: a) Verilog module 'comparator' which implements a NAND3 based comparator (lines 6-11). b) Gate-level schematic representation of the code.

two NAND3 gates together as in Fig. 2.6(a), an analog-input comparator is created if the common-mode of the input is high enough to ensure that the PMOS transistors connected to the input are in the cutoff region of operation. When the clock Φ is low, both outputs are reset to the positive supply rail. When the clock goes high, the outputs will begin to discharge through the three series NMOS devices. The discharge rate depends on the capacitance on the output node and the current through the three series devices. Since one of the series devices is connected to the analog input, the discharging current is related to the input. Once an output discharges to below a PMOS threshold voltage, the cross-coupled connection creates positive feedback that causes the comparator to force the outputs all the way to the supply rails. Implementing such a comparator can be done by explicitly

referencing the standard library cells in the RTL Verilog code as in Fig. 2.7(a). In this example, a static SR-latch is added to the output of the comparator. The SR-latch holds the output data valid while the comparator is reset. The SR-latch input is buffered with inverters to reduce a memory-effect on the comparator due to the SR-latch. Although this circuit is inherently compatible with digital synthesis, the synthesizer will assume that the circuit is actually a digital one, and will try and optimize it by replacing some of the gates or changing the circuit entirely while maintaining the same digital function. This digital optimization may render the circuit nonfunctional from an analog perspective, so here the command `set_dont_touch` *`comparator`* would prevent the synthesizer from altering the comparator module.

In the NAND3 comparator, the capacitance at the output nodes discharges as a rate that is dependent on an analog input. Using the same input-dependent discharging function, an analog voltage controlled delay cell can be created as seen in Fig. 2.6(b). When the clock input Φ goes high, the output is discharged to low at a rate that is dependent on the analog input V_a. The analog input voltage should be high enough as to keep the connected PMOS in the cutoff region of operation. In the figure, a NAND3 is used because this allows one input to be an asynchronous reset for the cell. Creating a large chain of delay cells can be used to create an voltage controlled delay line and can also be used to make an ADC as will be discussed in Section 4.

Finally, a NAND3 cell can also be used to implement a digital-to-analog (DAC) cell when used as in Fig. 2.6(c). Parasitic overlap capacitance creates a coupling effect between the input of one device and the input of an adjacent device. First, sample an analog voltage V_a onto the parasitic capacitance of a NAND3 gate input and hold it there at high-impedance. Changing the digital voltage D will

then cause a small voltage step on V_a due to the coupling capacitance and charge injection. This same DAC cell can be created out of a 2-input NAND gate, but a NAND3 gives better performance since an input can be held at digital zero to cause the NAND3 output to never change, saving power.

2.3 Summary

Analog circuit components can be integrated into standard digital circuit design if certain precautions are taken. A good candidate architecture for synthesis is one that is based on repeated and modular blocks that are not very sensitive to their physical location and routing. Designing and creating custom analog cells gives a designer a great deal of flexibility, but requires being very careful in integrating the custom cell into the standard cell library. Using standard cells to create analog circuits removes one layer of complexity by eliminating the need to integrate a custom cell into the library, but the design is less flexible since the design is limited to using already existing cells.

CHAPTER 3. STOCHASTIC FLASH ADC

All comparators have some input-referred offset due to random device mismatch. In a flash ADC, minimizing comparator offset is critical to the overall accuracy of the converter. This requires that each comparator consume a large area footprint in an effort to reduce device mismatch, or offset-canceling circuit techniques such as autozeroing or output offset storage must be implemented as described in [14]. The latter technique requires storing offset values on capacitors at the output of gain stages. Due to low intrinsic device gain multiple cascaded gain stages are typically used [15]–[17]. Instead of suppressing comparator offset, it is possible to use the random nature of the offset as part of a stochastic ADC.

Flash ADCs typically use a reference ladder to generate the comparator trip points that correspond to each digital code. First proposed in [18], a stochastic ADC uses comparators' inherent input-referred offset due to device mismatch as the trip-points. It has been proposed in [19] that by determining the offset of each comparator, it is possible to choose comparators with offsets that correspond to a desired transfer function. Choosing only the best of redundant comparators was also performed in the past in [20]. This solution requires a computationally expensive foreground calibration to generate a transfer function. In [19], the calibration logic consumed more area than the rest of the ADC combined; not including the computation engine which was off-chip. If comparator offset follows a distribution that is nearly linear, then the resulting comparator offsets can be used as the transfer function and none of this calibration hardware is required.

In a basic flash ADC, an input signal is connected to the inputs of a group

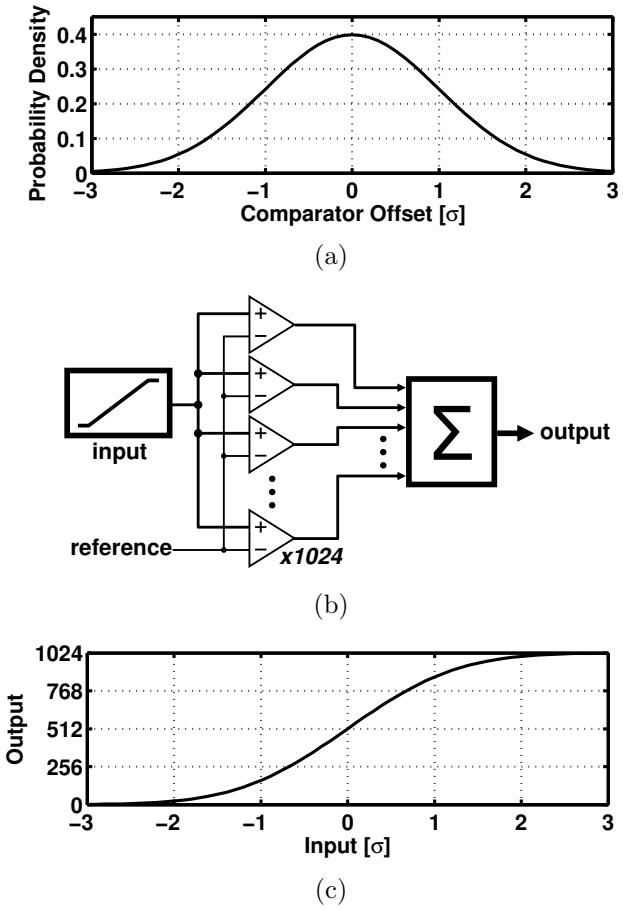


Figure 3.1: a) Probability density function of comparator offset in terms of standard deviation, σ, assuming Gaussian distribution. b) This is the basic stochastic flash ADC. c) Idealized output of the basic stochastic flash ADC with ramp input in terms of σ.

of comparators. The threshold of each comparator is set precisely, usually by a resistor string, such that all comparator thresholds are equally spaced by 1 LSB. In reality there is also a random offset in each comparator that, in effect, readjusts each comparator threshold by a random amount. This random offset, due to device mismatches will be assumed to be a Gaussian distribution with a mean (μ) of zero and variance (σ^2) inversely proportional to comparator area.

In a stochastic flash ADC, an input signal is also connected to the inputs of

a group of comparators. However, the comparator thresholds are not precisely set by design, but rather are allowed to be random. In the case of a standard flash, the comparator outputs after a conversion can be expected to be a thermometer code since the comparator thresholds are monotonically increasing by design. If each comparator threshold is random, however, then comparator outputs can not be expected to have any order. The total number of comparators that evaluate high will still be monotonically increasing with an increase in the input, so a ones adder is required to decode the output. This basic architecture with a group of comparators with random offsets followed by a ones adder is the basic stochastic flash ADC (Fig 3.1(b)).

The probability density function (PDF) of random comparator offset is influenced by many factors such as random variation of threshold voltage and current factor [21]. The Central Limit Theorem [22] indicates that since comparator offset is a sum of independent random variables with finite mean and variance the PDF will be approximately Gaussian (Fig. 3.1(a)). When a ramp signal is applied to the input of a basic stochastic flash ADC, the output will follow the cumulative distribution function (CDF) of comparator offset; therefore, the voltage transfer function of a basic stochastic flash ADC is the CDF of the random comparator offset (Fig. 3.1(c)). The number of comparators in the stochastic flash ADC must be enough such that the actual transfer function resembles the comparator offset CDF to the desired degree.

What make a stochastic flash ADC very interesting is that if the comparators are made to be digital cells, then the entire design can be synthesized like a digital circuit, as no analog references are required.

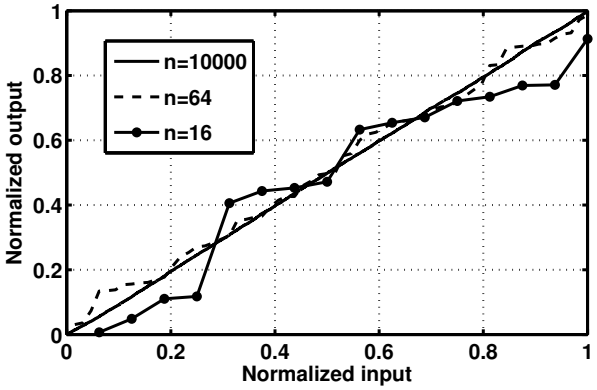


Figure 3.2: Normalized transfer function of a basic stochastic flash ADC with uniformly distributed comparator offsets for three cases where n is the number of comparators.

3.1 Statistical Analysis of a Uniform Distribution Stochastic Flash ADC

In a standard flash ADC, the number of comparators required to obtain N bits of quantization is $2^N - 1$. Since in a stochastic flash ADC the comparator levels are not set deliberately, but allowed to be random, a designer needs to know how many random comparator levels are required to obtain a desired accuracy. To analyze this, this section will only consider the case where comparator offset is random with a uniform PDF. The transfer function for a near infinite number of comparators will merely be the CDF of this random offset, which is a perfect line. Due to the random placement of each comparator level, a typical set of a smaller number of comparators will not give perfect linearity (Fig. 3.2). Actual comparator offset has a Gaussian PDF, but it will be demonstrated later that it is possible to approximate a uniform distribution, so this analysis is valid.

3.1.1 Number of Comparators Required

Before analyzing a stochastic flash ADC, let us first revisit how to obtain that for a conventional flash ADC, $2^N - 1$ comparators are required for N bits. The number of bits N can be calculated by determining the signal-to-quantization-noise ratio (SQNR) by the relationship,

$$N = \log_2(SQNR), \tag{3.1}$$

or when SQNR is expressed in decibels,

$$N = \frac{SQNR}{20\log_{10}(2)} \approx \frac{SQNR}{6.02}. \tag{3.2}$$

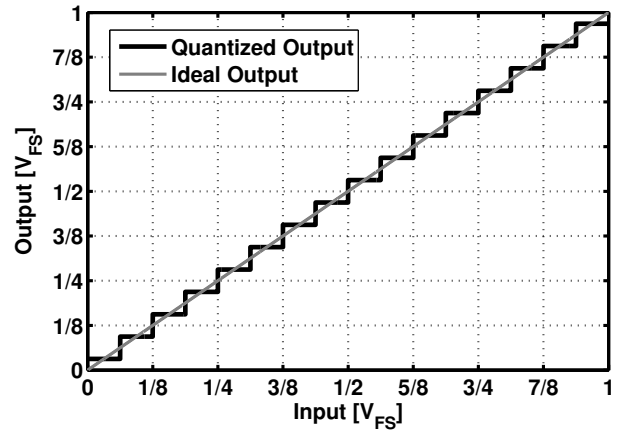
To calculate SQNR, the signal power and quantization noise power must be determined. First, consider the signal as being uniformly distributed between 0 and 1, the normalized range of the theoretical ideal ADC in Fig. 3.3(a). This signal can be described as a random variable with a uniform (PDF). The variance of a random variable is equivalent to its mean-square power, and since the variance of a uniform PDF is found to be

$$\mathrm{Var}(PDF_{uniform}) = \frac{\Delta^2}{12}, \tag{3.3}$$

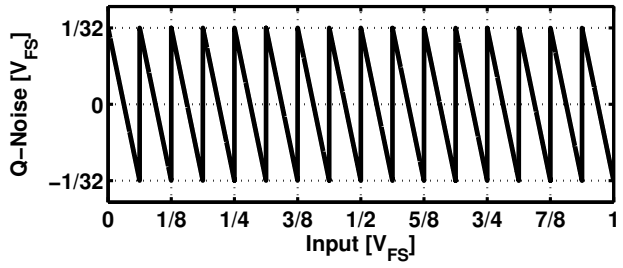
where Δ is the full range of the PDF. In the case of the signal, Δ=1 thus signal power is

$$P_{signal} = \frac{1}{12}. \tag{3.4}$$

To calculate quantization noise power, it must first be observed from Fig. 3.3(b)



(a)



(b)

Figure 3.3: a) An ideal 4-bit ideal ADC. b) The quantization noise voltage of an ideal 4-bit ADC, where quantization noise is the input subtracted from the output.

that quantization noise is uniformly distributed between $\pm 1/2$ LSB where

$$LSB = \Delta = \frac{1}{n+1}, \tag{3.5}$$

and n is number of comparators. Therefore,

$$P_{noise} = \frac{(\frac{1}{n+1})^2}{12}. \tag{3.6}$$

Now that signal and quantization noise power (mean-square) are known, SQNR in terms of voltage can be found by taking the ratio of the rms values of

each, i.e.

$$\begin{aligned} SQNR_{ideal} &= \frac{\sqrt{P_{signal}}}{\sqrt{P_{noise}}} \\ SQNR_{ideal} &= \frac{\sqrt{\frac{1}{12}}}{\sqrt{\frac{(\frac{1}{n+1})^2}{12}}} \\ SQNR_{ideal} &= n+1, \end{aligned} \tag{3.7}$$

or in decibels,

$$SQNR_{ideal,dB} = 20\log(SQNR_{ideal}). \tag{3.8}$$

From (3.1) and (3.7) the resulting number of comparators n required for a given number of bits N is

$$\begin{aligned} N &= \log_2(n+1) \\ n &= 2^N - 1, \end{aligned} \tag{3.9}$$

which is the expected result.

Now consider an ADC whose comparator thresholds are randomly and uniformly distributed along full-scale from 0 to 1. A possible example of a 3-comparator ADC of this type is shown in Fig. 3.4. Calling this a 2-bit ADC would be a misnomer, since quantization noise is actually much higher than an ideal 2-bit ADC, as will be demonstrated in this section. In order to bound quantization noise to zero at the extremes of the input (as in Fig. 3.4), let

$$LSB = \frac{1}{n}. \tag{3.10}$$

Fig. 3.5 is a visualization of how to describe the random comparator placement as a random variable. If it is known that there are n comparator thresholds

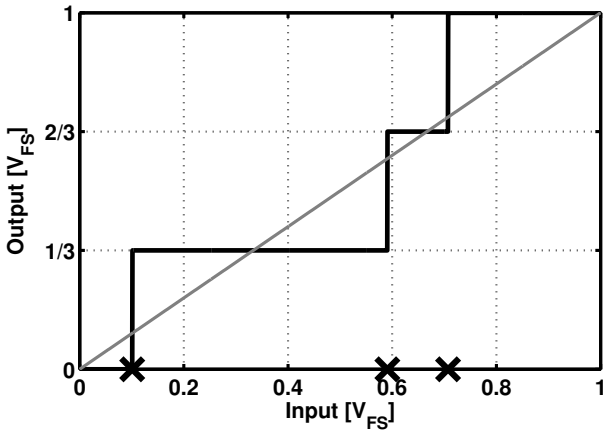


Figure 3.4: A 3-comparator ADC with random comparator placement. The marks along the x-axis represent the comparator thresholds.

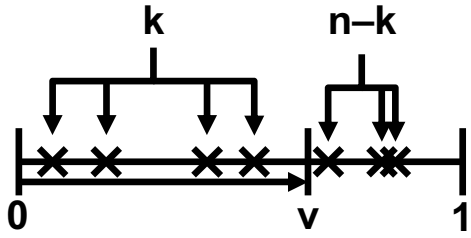


Figure 3.5: Graphical representation of a binomial random variable k – the number of n comparators between 0 and v.

within the range 0 to 1, and the number of thresholds between 0 and an arbitrary point v is called k, then the remaining thresholds $(n-k)$ must exist between v and 1. Since these comparator thresholds are uniformly distributed, the random variable k is a binomial distribution [23] with a probability mass function (PMF) of

$$\mathrm{PMF}_k(n,v) = \binom{n}{k}(v)^k(1-v)^{n-k}. \tag{3.11}$$

For any given value of input v, the output is merely k/n since $1LSB$ was

defined as $1/n$. The mean of the output k/n as a function of v is calculated to be

$$\begin{aligned} \text{Mean}(\frac{k}{n}(v)) &= \sum_{k=0}^{n} \left(\frac{k}{n}\right) \binom{n}{k} (v)^k (1-v)^{n-k} \\ \text{Mean}(\frac{k}{n}(v)) &= v. \end{aligned} \tag{3.12}$$

This result means that, for example, in the range 0 to 30% of full-scale there will on average be 30% of the total comparator thresholds. Any variation from this mean value results in quantization noise, therefore the variance of k/n is quantization noise power as a function of the input v, and is calculated to be

$$\begin{aligned} \text{Var}(\frac{k}{n}(v)) &= \sum_{k=0}^{n} \left(\frac{k}{n} - v\right)^2 \binom{n}{k} (v)^k (1-v)^{n-k} \\ \text{Var}(\frac{k}{n}(v)) &= \frac{v - v^2}{n}. \end{aligned} \tag{3.13}$$

Now to calculate SQNR for the uniform stochastic ADC, the signal power for a uniformly distributed input is the same as in the ideal case, only the quantization noise will be different, i.e.

$$P_{signal} = \frac{1}{12}. \tag{3.14}$$

Quantization noise power as a function of input was found in (3.13), and integrating this function over the signal range gives total quantization noise power;

$$P_{noise} = \int_0^1 \frac{v - v^2}{n} dv = \frac{1}{6n}. \tag{3.15}$$

This result can be verified numerically as shown in the example in Fig. 3.6.

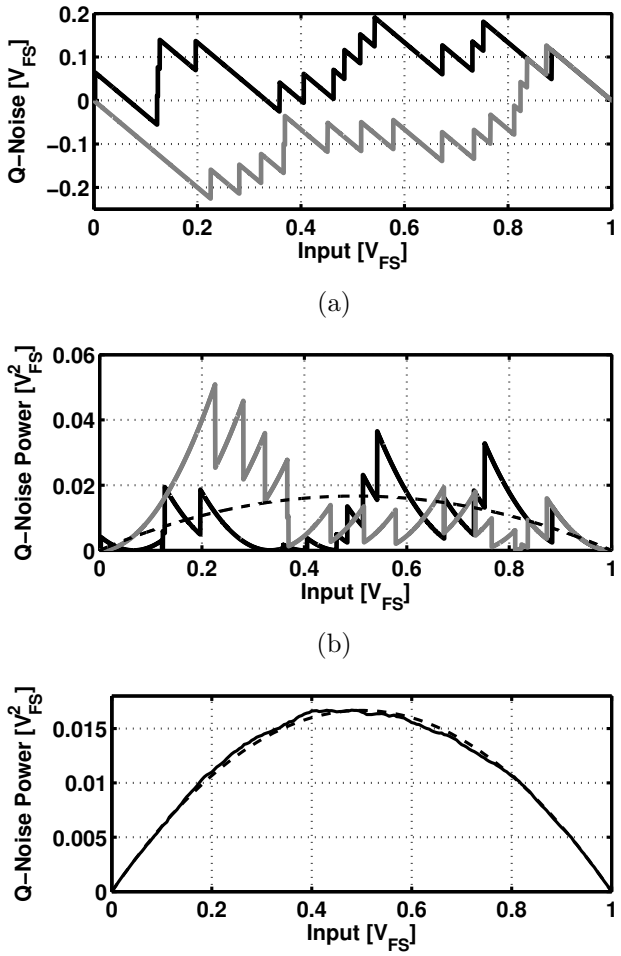


Figure 3.6: a) The quantization noise of two random 15-comparator ADCs with uniformly distributed comparator thresholds ($1LSB$=1/15). b) The square of quantization noise of the same two random ADCs. The dashed line is a plot of (3.13). c) The square of quantization noise of 5000 random ADCs averaged together. The dashed line is a plot of (3.13).

SQNR is finally calculated to be

$$\begin{aligned} SQNR_{stochastic} &= \frac{\sqrt{P_{signal}}}{\sqrt{P_{noise}}} \\ SQNR_{stochastic} &= \frac{\sqrt{\frac{1}{12}}}{\sqrt{\frac{1}{6n}}} \\ SQNR_{stochastic} &= \sqrt{\frac{n}{2}}. \end{aligned} \tag{3.16}$$

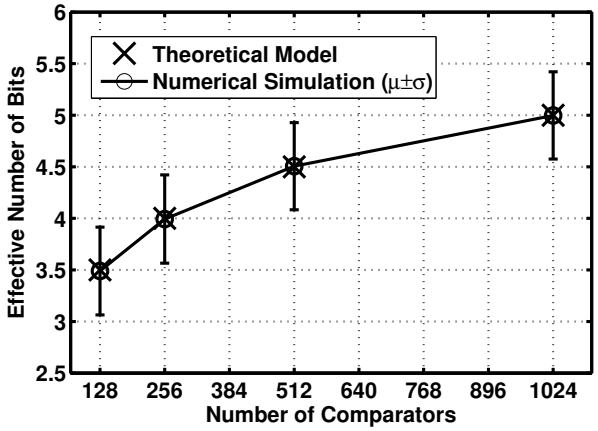


Figure 3.7: Effective number of bits as a function of number of comparators, where comparator thresholds are uniformly distributed across the input range.

From (3.1) and (3.16) the resulting number of comparators n required for a given number of bits N is

$$\begin{aligned} N &= \log_2(\sqrt{\frac{n}{2}}) \\ n &= 2 \cdot 4^N. \end{aligned} \tag{3.17}$$

This analysis assumes not canceling any DC offset of the ADC; if this offset is removed (as is usually the case) rms quantization noise will be decreased by 3dB [24], finally yielding,

$$n = 4^N. \tag{3.18}$$

This result can be easily verified with numerical simulation by taking n samples of a uniform random variable and using these values as the references for an simulated flash ADC. After applying a full-scale ramp input, the rms quantization noise can be calculated empirically, finally giving the resulting number of bits N. Repeating this test many times, as to satisfy the Law of Large Numbers [25],

allows us to find the average number of bits N and its standard deviation for a given number of comparators n. Plots of both the theoretical result from (3.18) and the numerical simulation result can be seen in Fig. 3.7. It is also relevant to note that the standard deviation of the number of bits N is approximately 0.42 bits regardless of the number of comparators.

To compare to a conventional flash ADC, the number of comparators must be increased by a factor of 2 to obtain 1 additional bit of accuracy. This analysis shows that to increase the accuracy of a uniformly random stochastic ADC by 1 bit, the number of comparators must be increased by a factor of 4.

3.1.2 Calculating ENOB from SNDR of a Sine-Wave Test

The value of the effective number of bits (ENOB) describes the overall accuracy of an ADC. The most widely used method to determine ENOB is to apply a sine-wave input and create a coherent FFT-plot. Signal-to-noise-and-distortion ratio (SNDR) is easily calculated by taking the power from the signal histogram bin and comparing it the the total remaining power from the remaining bins (after removing the DC bin). Creating an accurate, low-noise (through the use of band-pass filters) sine-wave input is very easy, making this method popular. Moreover, this method captures both static and dynamic sources of error. The accepted conversion to ENOB from SNDR from this sine-wave test is defined as

$$ENOB = \frac{SNDR - 1.76}{6.02}, \tag{3.19}$$

where SNDR has units of dB and ENOB is measured in bits. In this section it will be demonstrated that while (3.19) holds true for ADCs with quantization noise

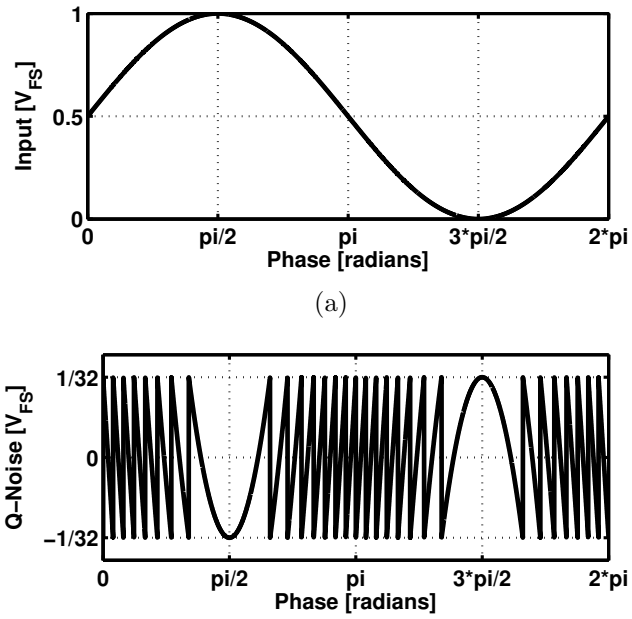


Figure 3.8: a) A full-scale sine-wave input. b) The associated quantization noise voltage for an ideal 4-bit ADC.

that is mostly uncorrelated to the input, if quantization noise is correlated to the input a different conversion of SNDR to ENOB should be considered.

To see where (3.19) comes from, first consider the ideal ADC from Section 3.1.1, and calculate the SQNR for when the input is a sine-wave. For a sine-wave input such as the waveform seen in Fig. 3.8(a), the observed quantization noise (Fig. 3.8(b)) still appears to be roughly uniformly distributed and, for the most part, uncorrelated to the input signal. Therefore the noise power in the sine-wave input case is identical to (3.6),

$$P_{noise,sine} = \frac{(\frac{1}{n+1})^2}{12}. \tag{3.20}$$

The signal, however, is no longer a uniform distribution, but a sine-wave

distribution whose PDF is

$$PDF_{sine}(v) = \frac{1}{\pi\sqrt{\frac{1}{4} - (v - \frac{1}{2})^2}} \tag{3.21}$$

over the full-scale range from 0 to 1. The variance of (3.21) is

$$\mathrm{Var}(PDF_{sine}) = \frac{1}{8}, \tag{3.22}$$

and thus the signal power is

$$P_{signal,sine} = \frac{1}{8}. \tag{3.23}$$

Calculating SQNR yields

$$\begin{aligned} SQNR_{ideal,sine} &= \frac{\sqrt{P_{signal,sine}}}{\sqrt{P_{noise,sine}}} \\ SQNR_{ideal,sine} &= \frac{\sqrt{\frac{1}{8}}}{\sqrt{\frac{(\frac{1}{n+1})^2}{12}}} \\ SQNR_{ideal,sine} &= \sqrt{\frac{3}{2}}(n+1). \end{aligned} \tag{3.24}$$

The result from (3.24) is not the same as (3.7), but the relationship between the two is

$$SQNR_{ideal} = \sqrt{\frac{2}{3}} SQNR_{ideal,sine}, \tag{3.25}$$

or in decibels,

$$\begin{aligned} SQNR_{ideal,dB} &= SQNR_{ideal,sine,dB} + 20\log(\sqrt{\frac{2}{3}}) \\ SQNR_{ideal,dB} &= SQNR_{ideal,sine,dB} - 1.76. \end{aligned} \tag{3.26}$$

It is from (3.26) and (3.2) that we derive (3.19) to make sure that ENOB is

in terms of a uniformly distributed input, repeated here,

$$ENOB = \frac{SQNR_{ideal,sine,dB} - 1.76}{6.02}. \tag{3.27}$$

Following the same procedure for the uniform distribution stochastic flash ADC, the difference between SQNR from a uniform input and a sine-wave input can be obtained. As in the case of an ideal ADC, the signal power of a sine-wave input is

$$P_{signal,sine} = \frac{1}{8}. \tag{3.28}$$

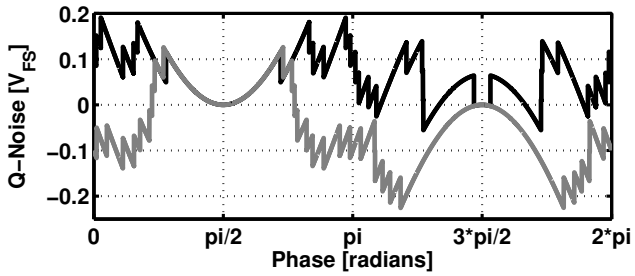
The quantization noise power, however, changes due to the input. Recall that from (3.13) (shown graphically in Fig. 3.6(c)), quantization power is smaller toward the extremes of the input range, and that a sine-wave spends more of its time at these regions. Integrating (3.13) multiplied by the PDF of a sine-wave weights the quantization noise power by the non-uniform distribution of the input and yields

$$P_{noise,sine} = \int_0^1 (\frac{v - v^2}{n}) \left(\frac{1}{\pi\sqrt{\frac{1}{4} - (v - \frac{1}{2})^2}} \right) dv = \frac{1}{8n}. \tag{3.29}$$

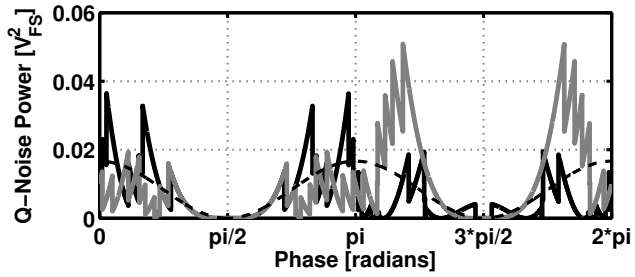
Another way of finding the same result is to substitute the actual sine-wave signal for v in (3.13) and integrate over the time period of the signal, yielding

$$\begin{aligned} P_{noise,sine} &= \frac{1}{2\pi} \int_0^{2\pi} \frac{(\frac{1}{2}\sin(t) + \frac{1}{2}) - (\frac{1}{2}\sin(t) + \frac{1}{2})^2}{n} dt \\ P_{noise,sine} &= \frac{1}{8n}. \end{aligned} \tag{3.30}$$

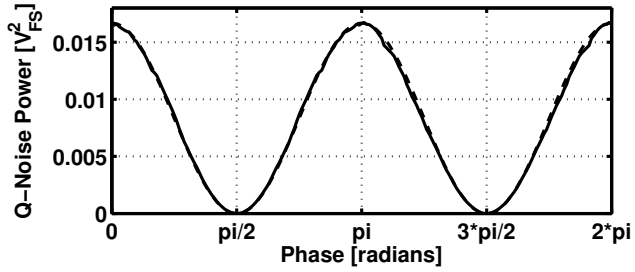
This result can also be verified numerically as in the example in Fig. 3.9.



(a)



(b)



(c)

Figure 3.9: a) The quantization noise due to the sine-wave input in Fig. 3.8(a) for the same two random ADCs as in Fig. 3.6. b) The square of quantization noise of the same two random ADCs. The dashed line is a plot of (3.13) with $v = (1/2)\sin(x) + 1/2$ and $x = 0...2\pi$. c) The square of quantization noise of 5000 random ADCs averaged together. The dashed line is again a plot of (3.13) with $v = (1/2)\sin(x) + 1/2$ and $x = 0...2\pi$.

SQNR for sine-input is finally calculated to be

$$\begin{aligned} SQNR_{stochastic,sine} &= \frac{\sqrt{P_{signal,sine}}}{\sqrt{P_{noise,sine}}} \\ SQNR_{stochastic,sine} &= \frac{\sqrt{\frac{1}{8}}}{\sqrt{\frac{1}{8n}}} \\ SQNR_{stochastic,sine} &= \sqrt{n}. \end{aligned} \tag{3.31}$$

The result from (3.31) is not the same as what was found in Section 3.1.1 as (3.16). As in the ideal ADC case, SQNR should be in terms of a uniformly distributed input, thus a conversion factor must be applied,

$$SQNR_{stochastic} = \sqrt{2} SQNR_{stochastic,sine}, \tag{3.32}$$

or in decibels,

$$\begin{aligned} SQNR_{stochastic,dB} &= SQNR_{stochastic,sine,dB} + 20\log(\sqrt{2}) \\ SQNR_{stochastic,dB} &= SQNR_{stochastic,sine,dB} - 3.01. \end{aligned} \tag{3.33}$$

This result implies that for an ADC such as described here, ENOB calculation from a sine-wave input should actually be

$$ENOB = \frac{SQNR_{sine} - 3.01}{6.02}. \tag{3.34}$$

3.2 Many-Group Stochastic Flash ADC

The actual distribution of comparator thresholds is not uniform, but rather a Gaussian distribution. This section will discuss how to effectively create a uniform distribution of comparator thresholds using multiple groups of comparators. The transfer function of a stochastic flash ADC is the integral of the PDF of its comparator thresholds. The integral of a Gaussian PDF is not linear (Fig. 3.1(c)), but the sum of many Gaussian PDFs can be. Fig. 3.1(b) depicts the basic stochastic flash ADC, which has a Gaussian PDF. Setting the global reference a to this comparator group effectively shifts the PDF to have a mean of a instead of zero (Fig. 3.10). The result of this is that it grants the ability to construct, effectively,

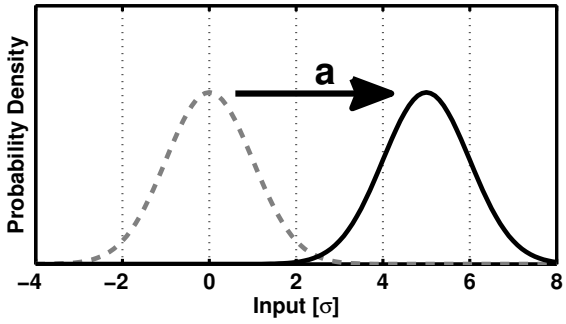


Figure 3.10: Providing a global reference a to a group of comparators effectively shifts the PDF to be centered about a ($a = 5\sigma$).

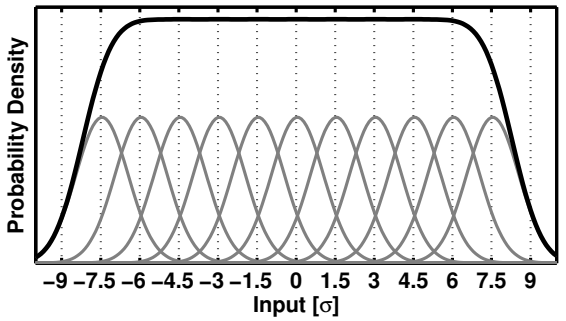


Figure 3.11: Spacing many Gaussian distributions by 1.5σ creates an effective uniform PDF.

a new PDF by using many Gaussian PDFs, each with a different mean.

Using a basic stochastic flash ADC (i.e. one group) by itself will obtain less than 3 bit linearity performance for a signal with range $\pm 2\sigma$, even if the number of comparators approaches infinity. Using multiple groups of comparators with the correct spacing of their means will yield as high of performance as desired. For example, Fig. 3.11 shows 11 Gaussian PDFs with their means spaced every 1.5σ. The resulting effective PDF is much more linear, in fact a signal of range $\pm 5\sigma$ would achieve almost 16 bit linearity. The total number of comparators required would be quite large to achieve this result from a stochastic quantization noise

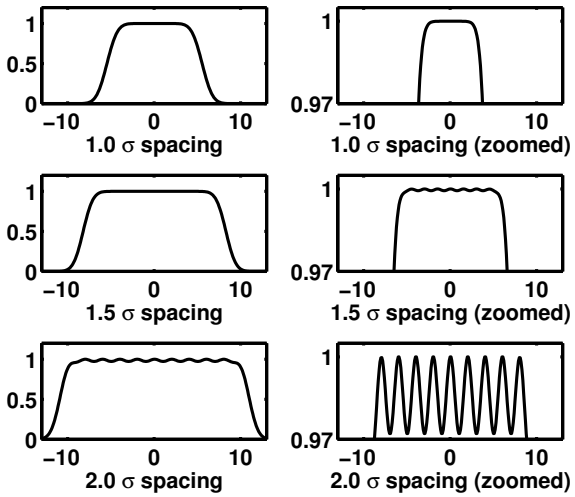


Figure 3.12: Spacing 11 Gaussian distributions by 1σ, 1.5σ, and 2σ, (as in Fig. 3.11) is a trade-off between signal range and a ripple on the effective uniform PDF.

point of view, but the inherent linearity of the distribution is such that the result is effectively a uniform PDF over a certain range.

There is a simple trade-off between signal range and linearity when it comes to setting the spacing of multiple comparator groups. The example in Fig. 3.12 shows 11 Gaussian PDFs with three difference spacing values. The larger the spacing value, the larger the linear region in terms of σ (which is in terms of volts); however, a spacing value larger than 1σ will cause a ripple to appear in this linear region. This ripple needs to be small enough as to not dominate in limiting the accuracy of the converter.

By effectively creating a uniform comparator offset distribution, the analysis of Section 3.1.1 is valid for this type of stochastic flash ADC.

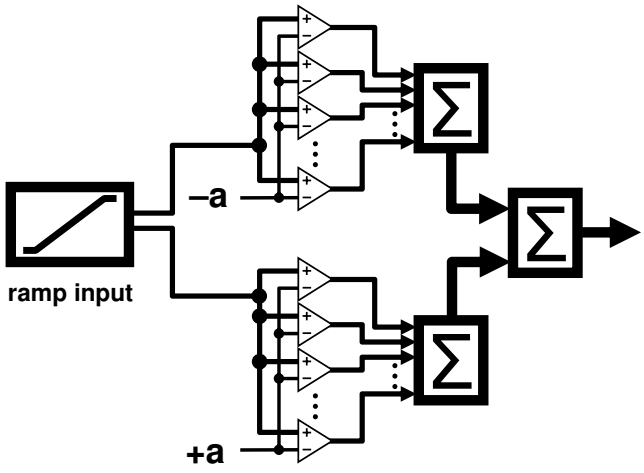
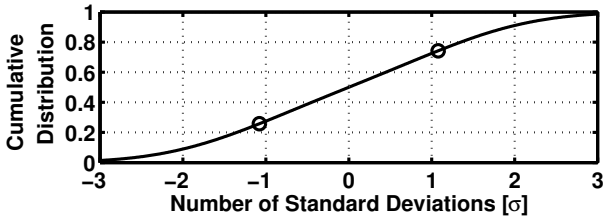


Figure 3.13: A two-group stochastic flash ADC. One group is given an offset of $+a$, the other $-a$.

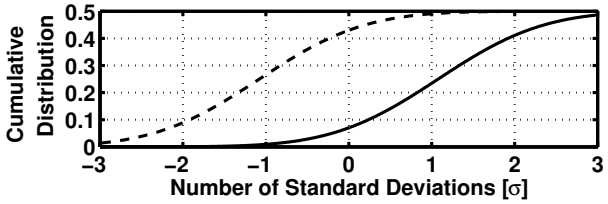
3.3 Two-Group Stochastic Flash ADC

Again, the actual distribution of comparator thresholds is not uniform, but rather a Gaussian distribution, so this section will demonstrate how to effectively realize a uniform distribution of comparator thresholds using only two groups of comparators. The transfer function of the basic stochastic flash ADC is the CDF of a Gaussian distribution. A Gaussian CDF is not linear (Fig. 3.1(c)), so linearization must be implemented in order to achieve a desirable linear transfer characteristic. Here we will consider using two basic stochastic flash ADCs, each with a Gaussian CDF transfer function but with a different mean (Fig. 3.13). This can be implemented by adding a constant, intentional offset to a group of comparators.

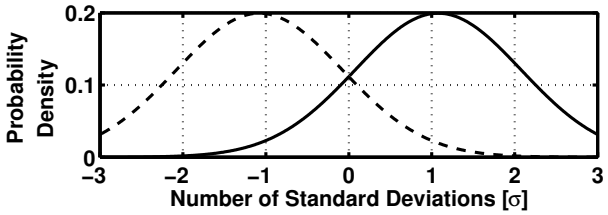
Changing the mean of comparator thresholds merely shifts the input-to-output transfer function along the input axis by applying a constant offset to



(a)



(b)



(c)

Figure 3.14: a) The resulting overall transfer function for a two-group stochastic flash ADC. (the circles correspond to the mean of each PDF). b) The transfer function of each group before being combined into the overall transfer function. c) The resulting PDFs of each comparator group after global offset is applied.

all comparators in that ADC. The outputs of each ADC are summed to obtain the overall output of this two-group stochastic flash ADC. As the two PDFs are shifted such that the difference of their means increases, a somewhat linear region appears when the input is bounded between the means of the two PDFs (Fig. 3.14). The equation for a Gaussian CDF is

$$\mathrm{f}(x) = \frac{1}{2}\left(1 + \mathrm{erf}\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)\right), \tag{3.35}$$

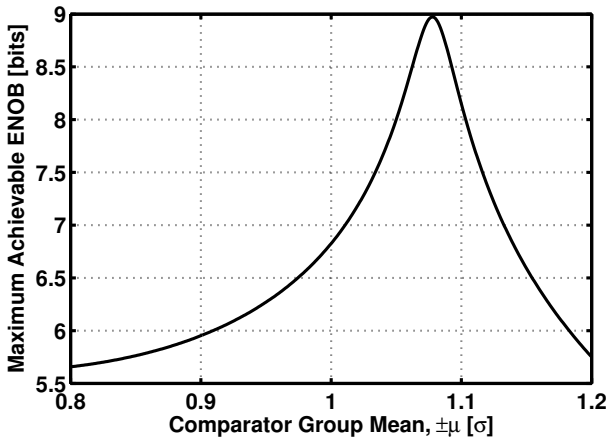


Figure 3.15: Maximum achievable linearity as number of bits for a two-group stochastic flash ADC with comparator group offsets of $\pm a$ and the input is also set to the range $\pm a$.

where

$$\mathrm{erf}(x) = \int_{-\infty}^{x} \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(u-\mu)^2}{2\sigma^2}}\, du. \tag{3.36}$$

Since the transfer function in which we are interested is for a two-group stochastic flash ADC, we will let $\mu = \pm a$ where an offset of a is applied to one ADC and an offset of $-a$ is applied to the other. We can let $\sigma = 1$, causing a to be in units of *number of standard deviations*, for simplicity without loss of generality. Therefore, the transfer function of a two-group stochastic flash ADC can be described by

$$\mathrm{g}(x) = \frac{1}{4}\left(2 + \mathrm{erf}\left(\frac{x-a}{\sqrt{2}}\right) + \mathrm{erf}\left(\frac{x+a}{\sqrt{2}}\right)\right). \tag{3.37}$$

Since we are interested in when the input is bounded between $-a$ and a, we can find the integral-nonlinearity (INL) by removing the constant linear portion and the overall offset of the transfer function by

$$\mathrm{INL}(x) = \mathrm{g}(x) - x\frac{\mathrm{g}(a) - \mathrm{g}(-a)}{2a} - \frac{1}{2}. \tag{3.38}$$

The rms value of this INL as a function of a is then

$$\mathrm{V}_{rms,INL}(a) = \sqrt{\frac{\int_{-a}^{a} \left(\mathrm{INL}(x)\right)^2 \, dx}{2a}}. \tag{3.39}$$

By relating this rms value of INL to LSB voltage,

$$\mathrm{V}_{rms,INL}(a) = \frac{\mathrm{V}_{LSB}}{\sqrt{12}} = \frac{\frac{1}{2^N}}{\sqrt{12}}, \tag{3.40}$$

we are able to obtain the optimal value for a, and what the maximum achievable number of bits (MANOB) is for a two-group stochastic flash ADC,

$$\mathrm{MANOB}(a) = \log_2\left(\frac{\mathrm{g}(a) - \mathrm{g}(-a)}{\sqrt{12}\mathrm{V}_{rms,INL}(a)}\right). \tag{3.41}$$

The closed-form solution of (3.41) is rather cumbersome and we would gain no additional insight by writing it out here, thus a plot of (3.41) can be seen in Fig. 3.15. There is a maximum of approximately 8.97 bits when a is approximately 1.078 standard deviations. This result means that the linear region between two offset Gaussian distributions, even if the number of comparator levels were infinite, is inherently limited to 8.97 bits. More importantly, if the targeted resolution is significantly less than 8.97 bits, then the comparators levels in the linear region between two offset Gaussian distributions is effectively uniformly distributed. Therefore, (3.18) can be applied if scaled by the inverse of the fraction of the comparator levels that will exist within the useful range.

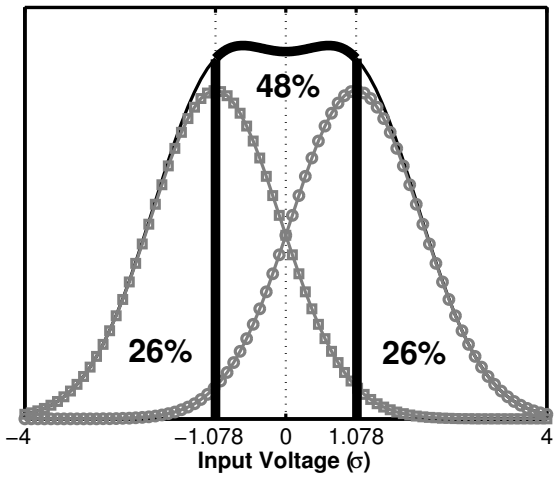


Figure 3.16: This is the combined probability density function (PDF) of comparator offset for a two-group stochastic flash ADC. Over half of the comparators fall outside of the virtual uniform distribution and are not used.

3.3.1 PDF Folding

One disadvantage of the two-group stochastic flash technique is that although a near uniform distribution (Fig. 3.14) is generated between the two means of the Gaussian distributions, it comes at a cost of wasted comparators. On average, 48% of comparator thresholds will reside in the useful region. This leaves 52% of comparators outside the range of the signal. These comparators continue to function and consume power, yet provide no additional information since they will always evaluate "high" or "low" whether they are in the left PDF or right PDF, respectively. This section describes a technique that folds each Gaussian PDF about its center to bring almost all of the comparators into the useful region.

Consider a comparator that has a +100mV input-referred offset. If this comparator lies in the "left" group of comparators, as defined in Fig. 3.16, then its output will always be "high" since the lowest differential voltage input in the

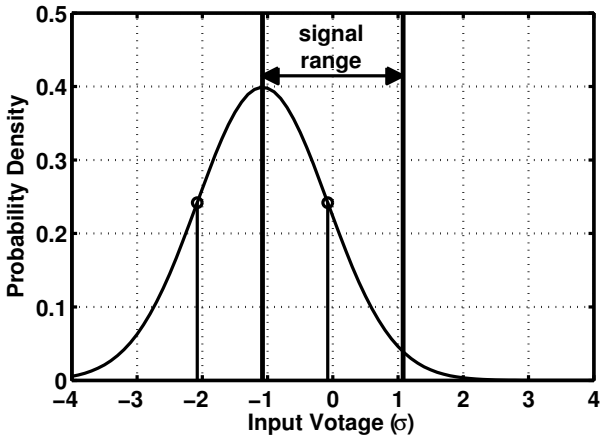


Figure 3.17: For clarity, only the left PDF is shown. The two circles indicate two random comparator offsets that are equal in magnitude but have opposite polarity.

signal range is 0V with respect to this comparator group. This comparator might as well always output "low" since either way it only contributes to a constant offset in the output code. Since the comparator offset is differential, if we can swap the polarity of the differential inputs, then the polarity of the offset will also be inverted. Fig. 3.17 shows an example of two different offsets of equal magnitude, but opposite polarity. It is apparent that one offset is within the signal range and is desired, while the other lies outside of the signal range.

A circuit that effectively folds all of the offsets that lie outside the signal range into the useful area can be seen in Fig. 3.18. If the common-mode of the input is high enough, than four PMOS switches can provide the ability to swap the polarity of the input differentially. The output of one D-flip-flop controls whether the input path is differentially inverting or non-inverting. When the other "lock" flip-flop is reset, then the input will change polarity every clock cycle. The input can not be inverted, however, without also inverting the digital output, so the flip-flop that controls the input polarity also inverts the output polarity through

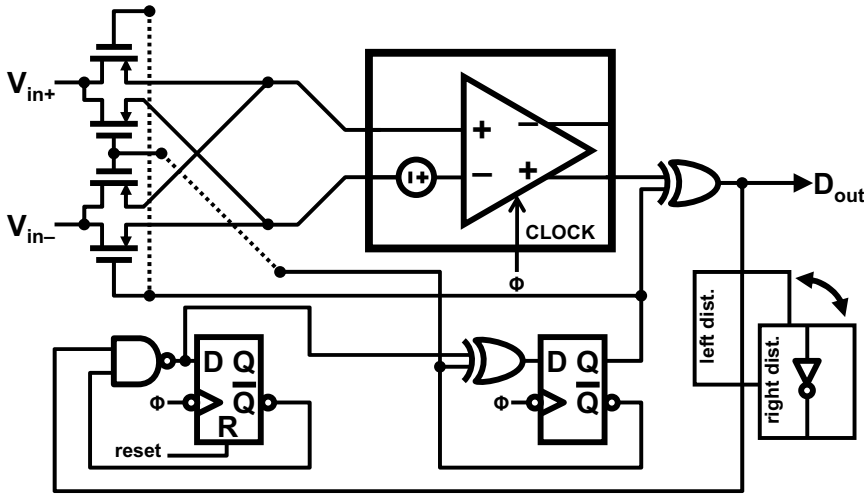


Figure 3.18: This circuit implements the comparator offset polarity inversion seen in Fig. 3.17. The circuit causes the analog inputs to be swapped and the digital output to be inverted each clock cycle until "locked."

an XOR gate. The final digital output is fed back to the "lock" flip-flop. If the digital output is ever "low," then it is known that the offset is within the useful range, therefore "low" causes the "lock" flip-flop to output "high" indicating that the circuit is locked, and the input polarity will remain fixed. For the "right" PDF of comparators, the circuit inverts the output of the XOR that is fed back to the "lock" flip-flop as shown in Fig. 3.18.

An example of the result of PDF folding can be seen in Fig. 3.19. Fig.3.19(a) is a histogram of 512 normally distributed comparator offsets shifted "left" with a global reference of -1.078σ. A full scale sine-wave is applied with the PDF folding circuits running. Once the PDF folding circuits have all locked, the resulting distribution ideally appears as in Fig. 3.19(b). In reality there will be noise in the comparator that could cause comparators with small magnitude offsets to lock onto the incorrect side (Fig. 3.19(c)). Locking onto the incorrect side is still acceptable. Since noise caused the offset to appear to be in the signal range, the comparator

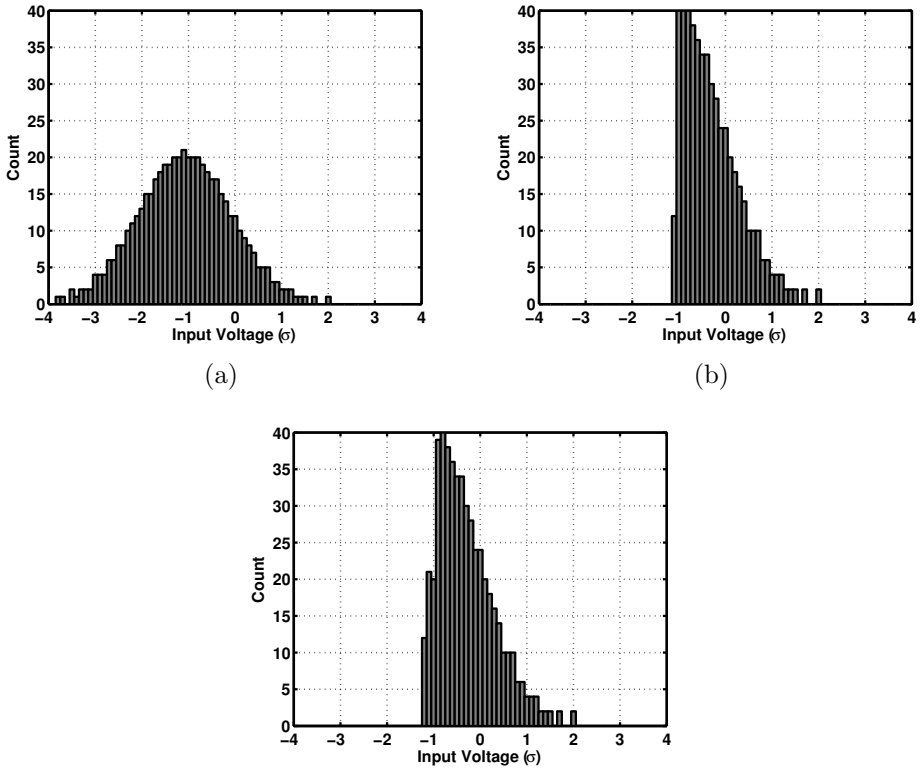
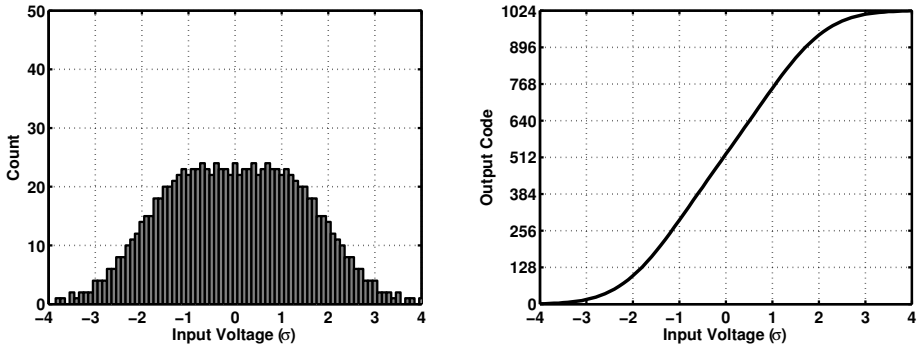


Figure 3.19: a) A histogram of 512 Gaussian comparator offsets. b) A histogram of the same 512 offsets after PDF folding has been applied. c) A histogram of the 512 offsets after PDF folding has been applied but in the presence of noise.

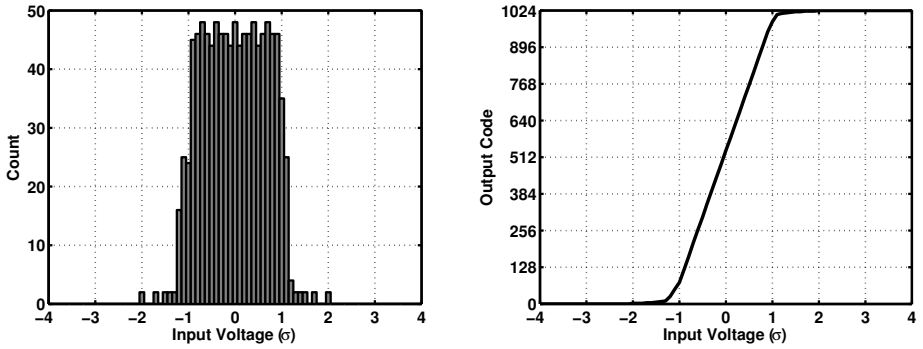
will continue to be able to give information as noise will cause it to appear within the signal range again.

The final result of PDF folding can be seen by comparing Fig. 3.20 and Fig. 3.21. Fig. 3.20 shows the overall PDF and resulting transfer function for 1024 comparators as a 512-per-group two group stochastic flash ADC. As described before, only 48% of the comparators lie within the linear range of the converter. After PDF folding is applied for the same comparator offsets, the result is a virtual uniform distribution as shown in Fig. 3.21. In the end, PDF folding cuts the number of comparators required for a design in half.



(a) (b)

Figure 3.20: a) A histogram of 1024 Gaussian comparator offsets in two groups of 512 and separated with global offsets as in Fig. 3.16. b) The corresponding transfer function for this set of comparator offsets.



(a) (b)

Figure 3.21: a) A histogram of the same offsets as in Fig. 3.20(a) after PDF folding has been applied. b) The corresponding transfer function for this set of comparator offsets.

It should be noted that while PDF folding reduces the number of comparators required, the PDF folding circuit adds a significant amount of area attributed to each comparator. If the PDF folding circuitry has comparable area to a single comparator then the area consumption of the two implementations would be comparable. Most likely there will be an increase in area if the comparator is a minimum-sized "digital-cell" comparator. The savings comes in reduced power

consumption. Once all of the PDF folding circuits have locked into their final state, they will consume very little power, leaving only the power consumed by the comparators.

3.3.2 Two-Group Prototype IC

A prototype IC was implemented to demonstrate the feasibility of actually building a two-group stochastic flash ADC. The prototype was required to use digital cell size comparators so that it can be a natural candidate to be a synthesizable ADC. Using minimum sized comparators also highlights the concept of a stochastic flash ADC because the large variation of comparator offset and the small input signal range would be prohibitive in a conventional flash ADC. Setting the references of two comparator groups to have an offset of approximately $\pm 1.078\sigma$, as found as the result of (3.41), allows higher linearity to be achieved.

3.3.2.1 Implementation details

The system level block diagram of our test chip can be seen in Fig. 3.22. There are two separate groups of comparators each with its own comparator reference. This is to implement the two-group stochastic flash ADC structure in Fig. 3.13. Adjusting the comparator reference for a group of comparators effectively changes the mean of the comparator offset CDF. In this manner, we can adjust the two comparator group references such that their means are $\pm 1.078\sigma$, yielding maximum linearity. As many comparators as possible were implemented on chip; there are 3840 comparators in each group. Each group is then subdivided into 20 subgroups of 192 comparators each that can be independently enabled or disabled by digital control.

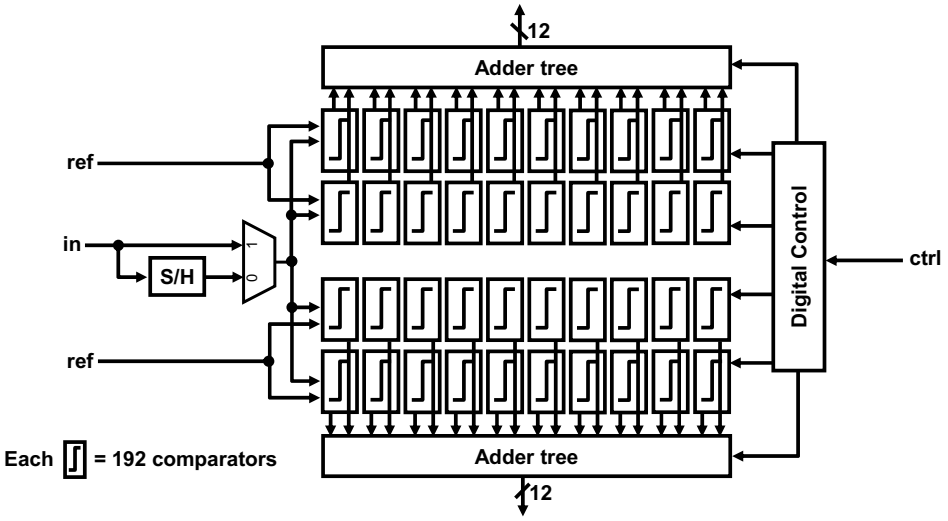


Figure 3.22: Block diagram of the prototype two-group stochastic flash ADC.

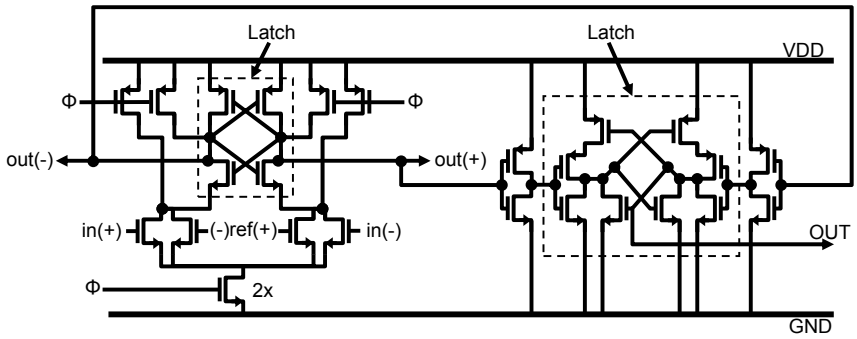


Figure 3.23: Schematic of the comparator with a secondary latch to maintain digital output when the comparator is reset.

The schematic of the comparators that were implemented in the test chip can be seen in Fig. 3.23. The comparator is followed by a secondary latch so that the digital output is maintained even when the comparator is reset. There is an interesting benefit in using a differential reference for the comparator in regard to control of the comparator offset distribution. Shown in Fig. 3.24, a differential change to the reference will cause a shift in the mean of the comparator offset CDF. A change to the common-mode of the reference changes the standard deviation of

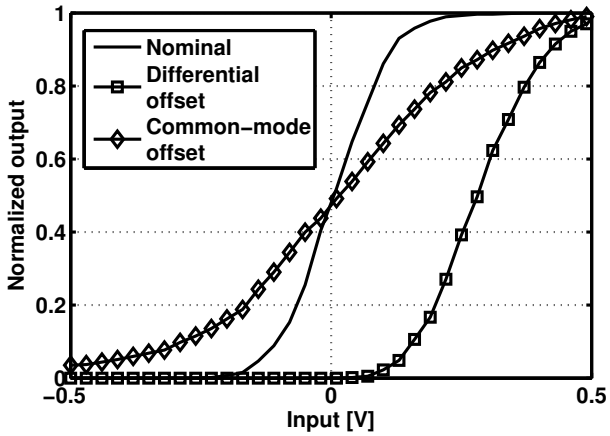


Figure 3.24: Measured change in the transfer function of a basic stochastic flash ADC by changing the global comparator reference differentially and by changing the common-mode.

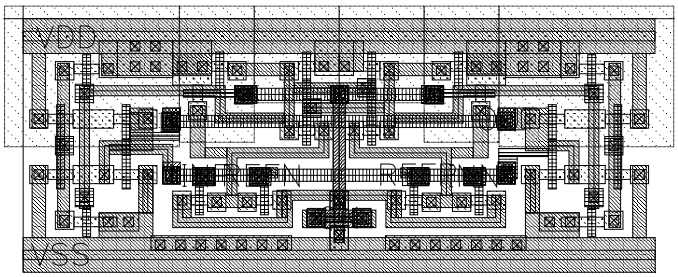
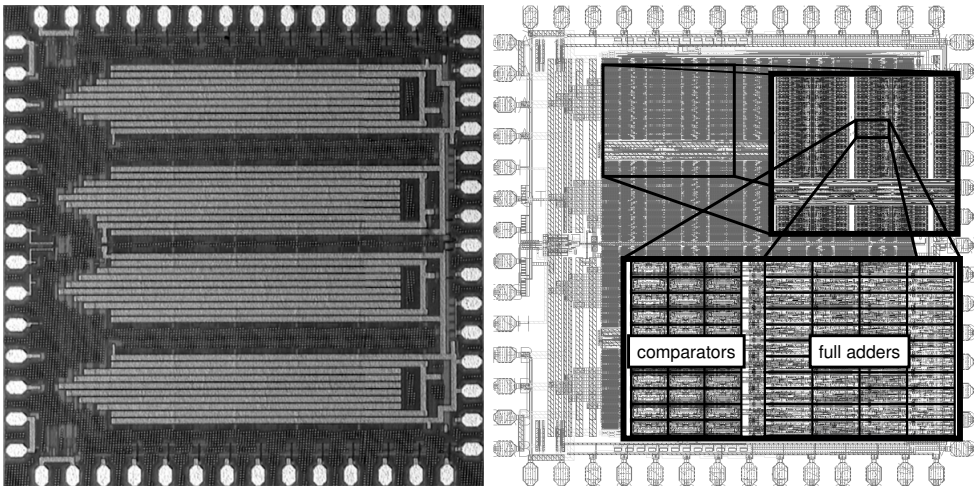


Figure 3.25: Layout of comparator and secondary latch. Cell dimensions are 14.55μm by 5.84μm.

comparator offset, because this will increase/decrease the dynamic offset. This implies that by controlling the two comparator group references, not only can the mean of the CDF be controlled, but the shape as well.

The comparator and secondary latch are made with minimum sized devices and incorporated into a digital cell (Fig. 3.25) that is comparable in size to a single full adder. The comparator cell has supply rails that match the pitch of the digital library rails to allow for automated synthesis. This design was not synthesized, but it was implemented in this manner to highlight that synthesis is possible.



(a) (b)

Figure 3.26: a) Die photo. Die dimensions are 2.4mm by 2.4mm. b) Layout screen capture showing detail of functional blocks.

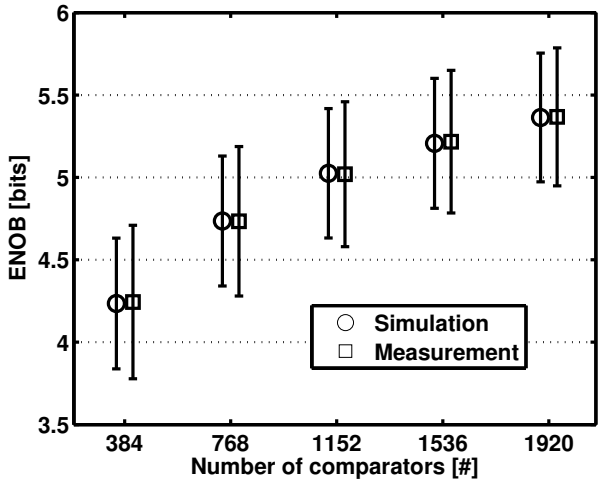
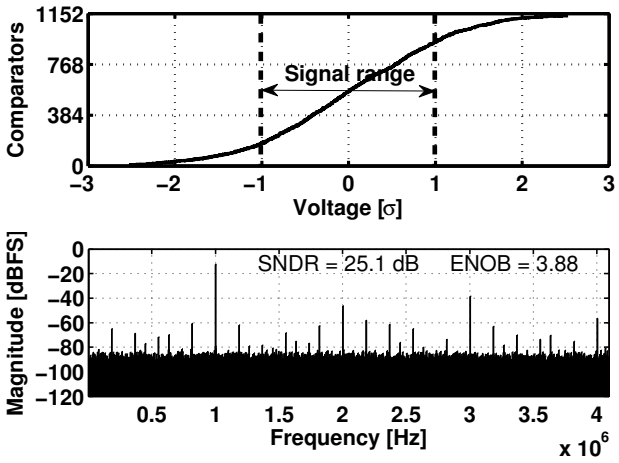
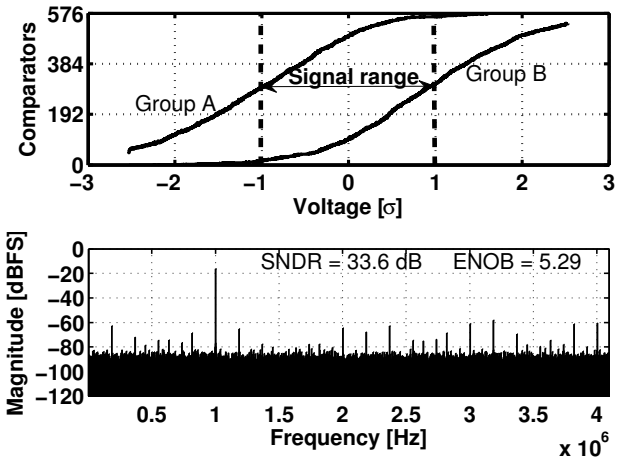


Figure 3.27: Measured ENOB plotted against number of comparators activated. For comparison, numerically simulated results for the same setup are plotted. Error bars indicate $\pm\sigma$ of ENOB.

To perform the digital sum of all of the comparator outputs for each group, a pipelined ripple-carry-adder tree ones adder was implemented [26]. Each comparator output is a single digital bit that is added with its two nearest neighbors by



(a)



(b)

Figure 3.28: a) Measured transfer function of a single group of 1152 parallel comparators ($\sigma \approx 140$ mV) and FFT of 1 MHz sine input. $f_S = 8$ MHz. b) Measured transfer function of the same parallel comparators as two groups of 576 with differing fixed references set to $\approx -1.078\sigma$ and $\approx +1.078\sigma$ for groups A and B, respectively. Also, FFT of output from the sum of groups A and B of 1 MHz sine input. $f_S = 8$ MHz.

a 1-bit adder. The 2-bit result from this adder is then added with a neighboring 2-bit result to yield a 3-bit result. This continues until finally there is a single 12-bit digital result. Adder stages are separated by D-flip-flops to pipeline the addition in order to minimize the time required for the adder tree to resolve each

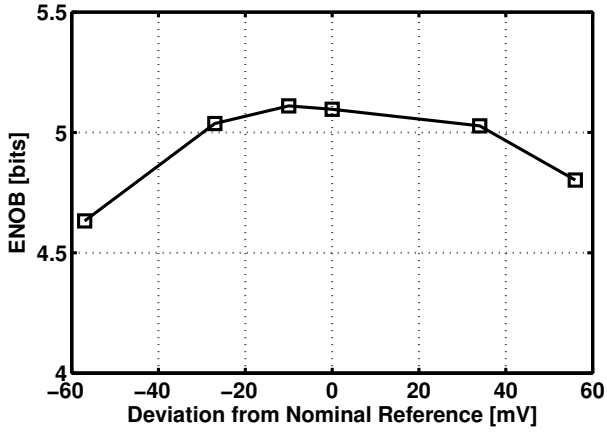


Figure 3.29: Measured ENOB for the two-group stochastic flash ADC (576 comparators per group) as a function of deviation from the nominal differential references, $\pm 1.078\sigma$. The range -60mV to +60mV is equivalent to $\pm 0.8\sigma$ and $\pm 1.2\sigma$, respectively.

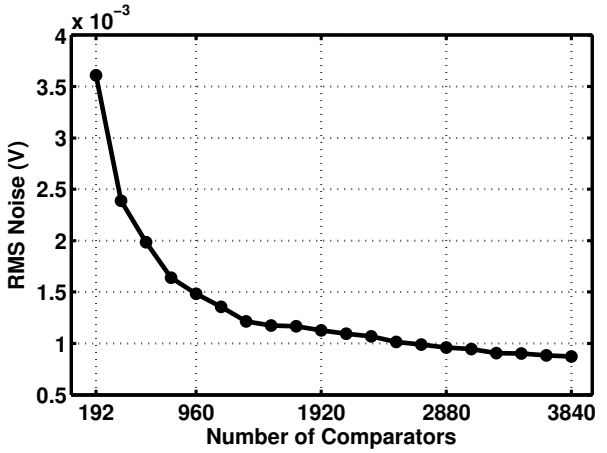


Figure 3.30: Measured input-referred noise as a function of total number of comparators.

clock cycle.

3.3.2.2 Measurement Results

The test chip was fabricated in 0.18μm CMOS (Fig. 3.26) with a total area of 5.76 mm^2. Each 192-comparator block devotes 0.017 mm^2 to analog comparators

with a 0.022 mm^2 digital overhead for the full adders associated with that block. It can be seen in Fig. 3.27 that increasing the number of active comparators yields a measured increase in ENOB calculated from SNDR. For all of the measurement results in this section, ENOB is calculated using the conventional

$$ENOB = \frac{SNDR_{sine} - 1.76}{6.02}. \tag{3.42}$$

For each data point, 500 random combinations of comparator groups were enabled on 4 different chips to obtain an average ENOB and standard deviation for a given number of comparators. As a point of reference, simulated ENOB is also plotted. The simulation setup was two Gaussian random variables with $\mu = \pm 1.078\sigma$ and the same number of instances as comparators in the measurement setup. By taking many iterations of this simulation, we find the expected value and standard deviation of ENOB. The measured data is consistent with the simulated results. These results are also consistent with what was found in Section 3.1.1 when the result from (3.33) is taken into account.

Since these digital cell comparators are made up of minimum sized transistors, the standard deviation (σ) of comparator offset is expected to be quite large. In fact, measurement shows that for our test setup with, for example, a supply voltage of 900 mV, $\sigma \approx 140$ mV. Because the signal range is approximately $-\sigma$ to $+\sigma$ the resulting signal range is 280 mV. Without the use of analog offset cancellation techniques, it would not be possible to build a standard flash ADC with comparator offsets of this magnitude. This is a major benefit in terms of synthesis, since it would be very difficult to synthesize analog offset cancellation. Although the comparator offsets do not need to be calibrated, this technique does require two differential references to set the global mean of each comparator group. Fig. 3.29

Table 3.1: Performance Summary

Technology	0.18μm CMOS
Resolution	6b
Max Sampling Rate	18MS/s
Supply Voltage	900 mV
Comparator Offset Standard Deviation	140 mV
Input Range	560 mVpp (differential)
SNDR / SFDR @ f_S=8 MHz f_{in}=1 MHz	33.59 dB / 42.86 dB
DNL @ f_S=8 MHz	-0.38 / +0.50 LSB
INL @ f_S=8 MHz	-1.06 / +1.07 LSB
Analog Power @ f_S=8 MHz	182μW
Digital Adder Power @ f_S=8 MHz	261μW
Clock Driver Power @ f_S=8 MHz	188μW
Total Power @ f_S=8 MHz	631μW
Core Active Area	0.43 mm^2

shows that these differential references do not need to be absolutely accurate if the design is limited by quantization noise. For example, a servo loop could set the references in the background by comparing the digital output of each comparator group and slowly adjusting the two global references until the $+\sigma$ code of one group corresponds to the $-\sigma$ of the other group.

The two-group stochastic flash ADC linearizion is demonstrated in Fig. 3.28. For this example we will choose a 1152 comparators. With all 1152 comparators acting as a single parallel group, sweeping the input with a linear ramp reveals a transfer function that is indeed a Gaussian CDF. SNDR of 25.1 dB is achieved with a 1 MHz sine-wave input and sampling frequency of 8.192 MHz. Using the exact same comparators under the same conditions, but merely dividing them into two groups with differing references ($\approx \pm 1.078\sigma$), an 8.5 dB improvement in SNDR can be seen.

Power consumption for the analog portion is 182μW. Digital power is 449μW with 188μW consumed by clock drivers, leaving 261μW consumed by the pipelined ripple-carry adder tree.

An additional thing that can be measured is the input-referred noise as a function of the number of comparators (Fig. 3.30). Measuring the input-referred noise of a single regenerating latch comparator is not trivial [27]. For a stochastic flash ADC, measuring the input-referred noise can be done by applying a DC input and clocking the comparators multiple times. Since each comparator level is equated to some effective voltage change, rms noise is calculated by the square root of the variance of the output code. As expected, the input-referred noise decreases as the number of comparators increases due to an averaging effect of the noise.

3.3.2.3 Summary

This prototype IC demonstrates the feasibility of actually building a two-group stochastic flash ADC. The use of digital cell comparators allows it to be a natural candidate to be a synthesizable ADC. Using minimum sized comparators that are implemented as digital cells produces a large variation of comparator offset, which is used to set the trip point of each comparator and in the end, defines the input signal range. Comparator trip points follow the nonlinear transfer function described by a Gaussian CDF. The prototype shows that it is indeed possible to reduce this nonlinearity by changing the overall transfer function by building a two-group stochastic flash ADC. Setting the references of two comparator groups to have an offset of approximately $\pm 1.078\sigma$ of comparator offsets allows higher linearity to be achieved.

3.4 Single-Group Stochastic Flash ADC

Just as in Section 3.2 and Section 3.3, the goal of this section is to realize an virtual uniform distribution of comparator offset even though the actual distribution of comparator thresholds is a Gaussian PDF. This section will only consider a single group of comparators. The biggest advantage of using a single group is that it eliminates the need to provide analog references to groups of comparators. The transfer function of a single-group stochastic flash ADC is merely the integral of the PDF of its comparator thresholds, that is the integral of a Gaussian PDF (Fig. 3.1(c)). To achieve an output that is linear, the ADC must take advantage of the fact that we have the knowledge of the shape of the random offset distribution a priori: it is Gaussian. The diagram in Fig. 3.31 shows the basic concept of

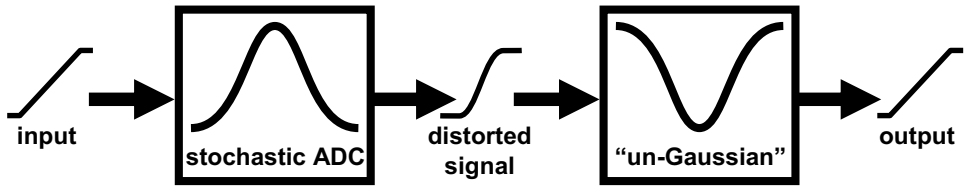


Figure 3.31: Block level diagram of a single-group stochastic flash ADC.

following the stochastic ADC output by a digital correction block that effectively “undoes the Gaussian” to obtain an effectively linear output.

3.4.1 Gaussian Distribution Mapped to a Uniform Distribution

The un-Gaussian block can be implemented as the inverse transfer function of a Gaussian CDF either as a lookup table or a digital mathematical function. There are grave disadvantages to implementing this. A lookup table is a large hardware requirement since it is implemented as a SRAM that must be able to output as fast as the ADC. Depending on the accuracy that is being designed for, a piecewise linear approximation of an inverse Gaussian CDF may be sufficient. Fig. 3.32 is an example of such a peicewise function defined by,

$$unGaussian(v) = \begin{cases} 2.5v - 0.512 & : & 1.166\sigma < & v & \\ 1.5v - 0.134 & : & 0.732\sigma < & v & \leq 1.166\sigma \\ v & : & -0.732\sigma \leq & v & \leq 0.732\sigma \\ 1.5v + 0.134 & : & -1.166\sigma \leq & v & < -0.732\sigma \\ 2.5v + 0.512 & : & & v & < -1.166\sigma \end{cases}, \tag{3.43}$$

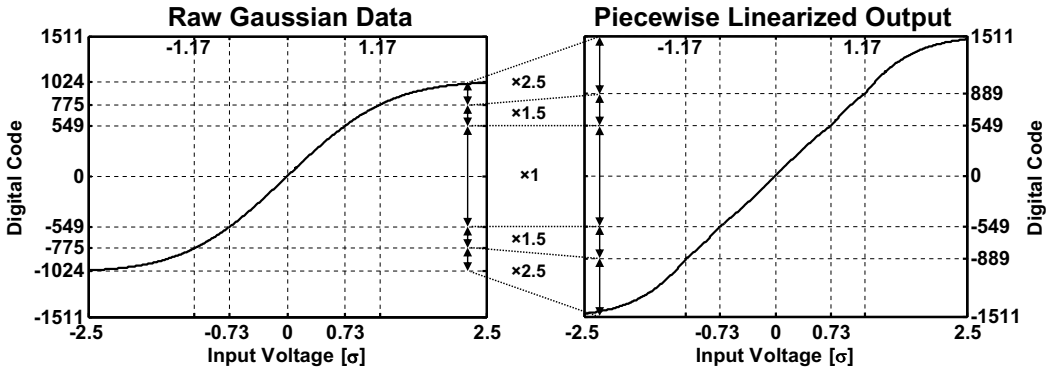


Figure 3.32: A piecewise linear approximation of an inverse Gaussian CDF give a more linear transfer function.

where v is the output of the transfer function and is in terms of standard deviation σ. This example only has five piecewise regions. More regions can be added depending on the accuracy required of the approximated inverse Gaussian CDF. For the design that was targeted for the prototype ADC, five regions was sufficient. The fact that the un-Gaussian function is a function of the output is very important. By being a function of the output, it does not matter what the input characteristics are, or whether or not there is an offset to the distribution.

If the mean or standard deviation of the comparator offset distribution should change due to variation in process, voltage, temperature, or any other reason, the CDF transfer function would be shifted and scaled with respect to input voltage; however, the shape of the CDF remains the same. The digital output only represents the shape of the CDF, e.g. code 0 (signed) always represents the mean of the distribution, and code +699 (for 2047 comparators) always represents one standard deviation above the mean. Therefore, no calibration or tuning is required; the inverse Gaussian CDF can be hard coded into the chip.

When implementing (3.43) for an actual ADC the values for the piecewise function can be rounded to the nearest code. As an example, consider a 2047-

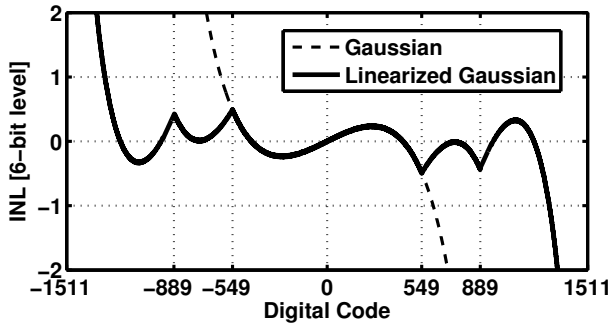


Figure 3.33: A plot of the integral nonlinearity for a 2047-comparator single-group stochastic flash ADC with and without piecewise linear approximation.

comparator single-group stochastic flash ADC. The function that would actually be implemented is,

$$unGaussian(v) = \begin{cases} 2v + v/2 - 1049 & : & 775 > & v & \\ v + v/2 - 274 & : & 549 < & v & \leq 775 \\ v & : & -549 \leq & v & \leq 549 \\ v + v/2 + 274 & : & -775 \leq & v & < -549 \\ 2v + v/2 + 1049 & : & & v & < -775 \end{cases} . \tag{3.44}$$

Note that here the coefficients of v were bounded to be factors of two as to reduce the hardware complexity of implementation, since multiply by two and divide by two have nearly no hardware cost in binary arithmetic. The effect of the un-Gaussian approximation on INL can be seen in Fig. 3.33. The INL is only within the 6-bit level for a small range for the uncorrected Gaussian transfer function. By adding some correction, the effectively linear range is increased.

The last thing to address is if (3.18) still describes this architecture. In effect, the un-Gaussian function requires reevaluating (3.13) and replacing v with the CDF of a Gaussian. The end result is that an un-Gaussian function placed

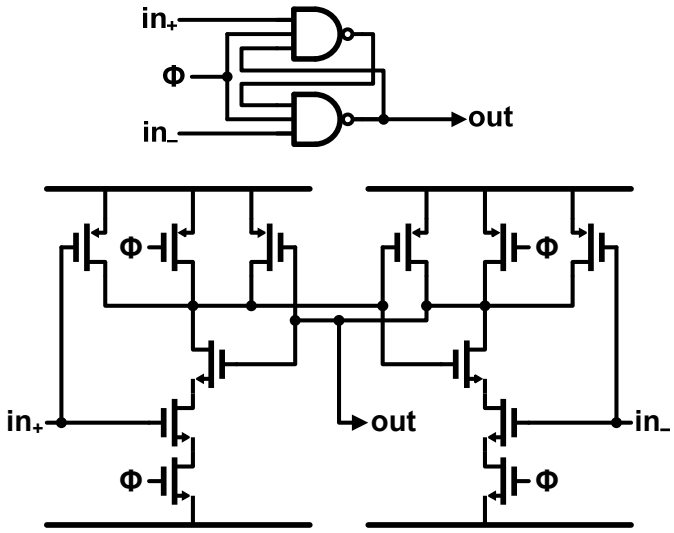


Figure 3.34: An analog comparator realized using standard digital 3-input NAND gates.

after the ADC output effectively creates a uniform distribution and (3.18) is still true. This is verified experimentally by the prototype IC that was implemented.

3.4.2 Single-Group Prototype IC

A prototype IC was implemented to demonstrate the feasibility of building a single-group stochastic flash ADC. This prototype IC was required to use the 3-input NAND based comparator discussed in Section 2.2 and was automatically synthesized from Verilog code. The piecewise linear approximation of an inverse Gaussian CDF was implemented on chip to linearize the output.

3.4.2.1 Implementation Details

For simplicity the example implementation of a 7-comparator single group stochastic flash ADC will be discussed; however, the actual fabricated ADC in-

```
 1  module adc(inp, inn, clk, dec_en, out);
 2   input inp, inn, clk, dec_en;
 3   output [4:0] out;
 4
 5   wire [6:0] q;
 6   wire [4:0] fastout;
 7   reg [4:0] out;
 8   reg [2:0] dec_cnt;
 9
10   comparator U1( .INP(inp), .INN(inn), .CK(clk), .Q(q[0]) );
11   comparator U2( .INP(inp), .INN(inn), .CK(clk), .Q(q[1]) );
12   comparator U3( .INP(inp), .INN(inn), .CK(clk), .Q(q[2]) );
13   comparator U4( .INP(inp), .INN(inn), .CK(clk), .Q(q[3]) );
14   comparator U5( .INP(inp), .INN(inn), .CK(clk), .Q(q[4]) );
15   comparator U6( .INP(inp), .INN(inn), .CK(clk), .Q(q[5]) );
16   comparator U7( .INP(inp), .INN(inn), .CK(clk), .Q(q[6]) );
17
18   dsp U8 ( .c0b0(q), .clk(clk), .final(fastout) );
19
20   always @(negedge clk) begin
21    if (dec_en == 1) begin
22     if (dec_cnt == 0)
23      out <= fastout;
24     dec_cnt <= dec_cnt + 1;
25    end
26    else
27     out <= fastout;
28   end
29
30  endmodule
```

Figure 3.35: Verilog module 'adc' which creates an ADC and includes a decimate by 8 option (lines 20-28).

cluded 2047 comparators.

The comparator used in this design, discussed in Section 2.2, is repeated here in as Fig. 3.34. For the Verilog code to define this comparator as a module, refer back to Section 2.2. Once the comparator module is defined, the rest of the design is a digital circuit.

Verilog code that implements the ADC at the top level can be seen in Fig. 3.35. The outputs of the seven comparators, `q[0]`, `q[1]`, ..., `q[7]` are passed

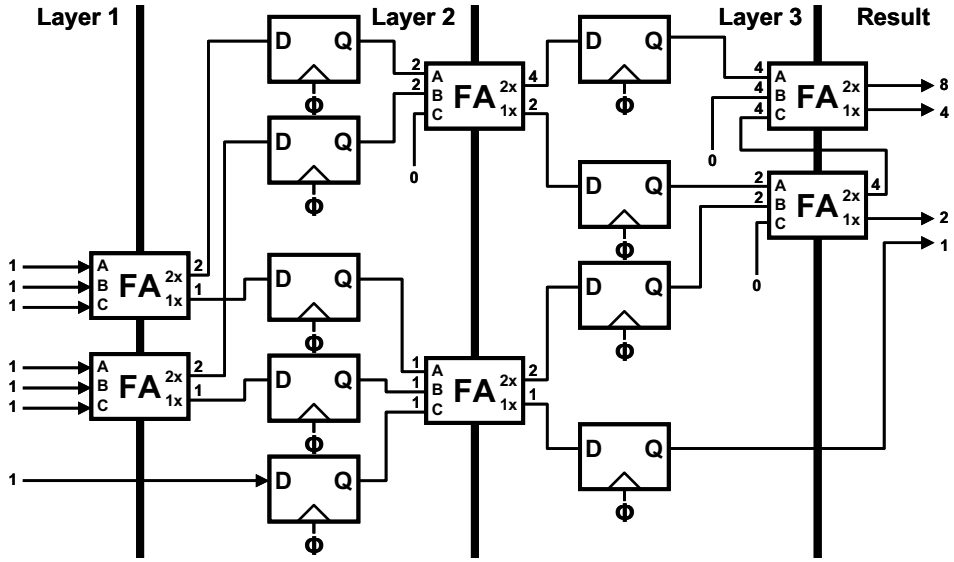


Figure 3.36: Diagram of a pipelined Wallace tree ones adder for 7 inputs.

the the module `dsp` which consists of a ones adder and the piecewise un-Gaussian function. Lines 20–28 implements a decimate-by-8: when decimation is enabled by the input `dec_en` being set to "1" (`dec_en` is connected to an external pin) the output `out` is updated whenever the free-running counter `dec_cnt` is equal to zero. Since `dec_cnt` is a 3-bit register, it will be equal to zero every eight cycles.

A very efficient ones adder is a Wallace tree; an example of which can be seen in Fig. 3.36. A Wallace tree uses single-bit binary full-adders as 3:2 compressors. Each full adder takes in three binary bits of the same bit-weight and outputs the same total value, but as one bit of the same bit-weight as the inputs and one bit of twice the bit-weight as the inputs. The summation then proceeds as follows. Take some number of inputs of the same bit-weight, seven, for example. For every set of three or two, sum them in a full-adder; single remainders pass to the next layer as-is. At the next layer, group all of the bits of the same bit-weight an repeat. Continue operating until there are two or less of each bit-weight and sum the result

in a conventional carry-look-ahead adder. This method of adding many bits of the same bit-weight can be made very fast since the time delay from one layer of inputs to the next is only a single full-adder delay. Contrast this to a ripple-carry-adder tree where the time delay from one layer to the next may be many full-adder delays. To increase the speed further, D-flip-flops are placed between each layer of full-adders to pipeline the adder tree. This creates many cycles of latency in exchange for a higher clock rate. The implementation of the Wallace tree from Fig. 3.36 is done in Verilog as lines 21–29 of Fig. 3.44.

The piecewise linear approximation of the un-Gaussian function is implemented as a series of `if` statements in lines 33–42 of Fig. 3.44. In this example, there are only 7 comparators. For the actual ADC that was fabricated with 2047 comparators, the function from (3.44) was implemented.

3.4.2.2 Measurement Results

The test chip was fabricated in a 90nm digital CMOS process with a total area of 0.18 mm^2. The top metal layer was covered with dummy metal fill, so there was nothing interesting to see in the die micrograph, so a screen capture of the layout can be seen in Fig. 3.37. At the point where the input net connects to each comparator input, a bold "x" has been placed to show the detail of the comparator placement. The comparator placement is resemblant of a fractal tree because of the fractal-like nature of a large Wallace tree ones adder.

To test the functionality of the linearizion of the piecewise un-Gaussian approximation, a ramp input was applied to the ADC to obtain the transfer function (Fig. 3.38). It is also observed that changing the common-mode of the input signal has affects the variance of the comparator offset distribution.

Obtaining the transfer function also makes it possible to measure the INL of

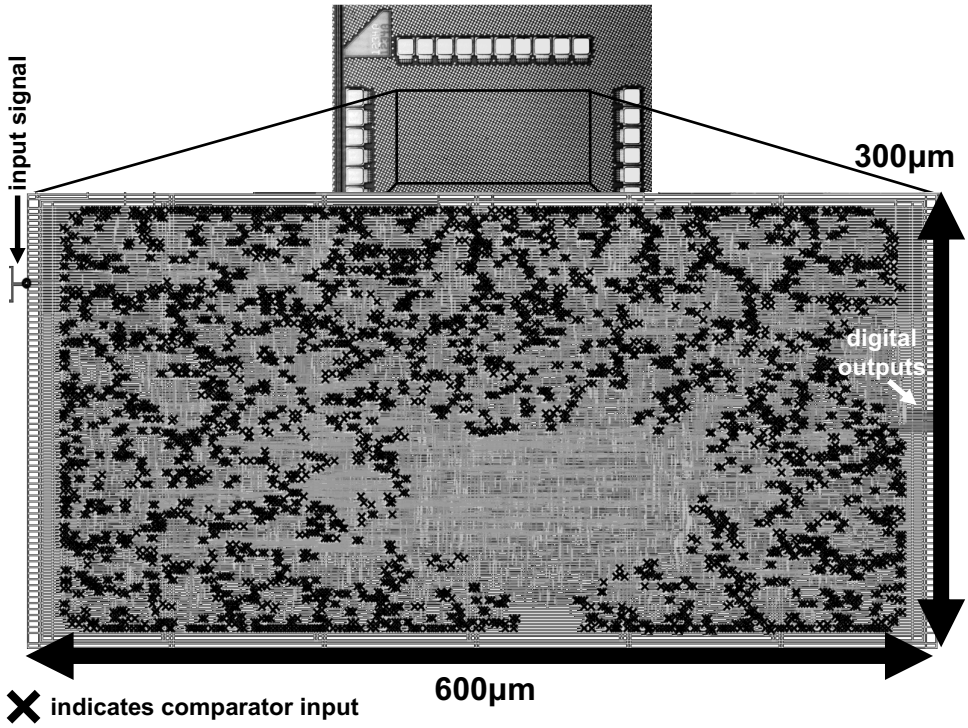


Figure 3.37: Screen capture of the prototype IC with the comparator inputs each marked as a black x. Dimensions are 300μm by 600μm.

the ADC (Fig, 3.39). The trend of the INL from simulation can be clearly seen in the measurement.

The maximum sampling rate for a supply voltage of 1.2V is determined to be 210MSPS by applying a 1MHz input signal and increasing the sampling rate until the SNDR drops off appreciably (Fig. 3.40). The steep drop seen above 210MSPS is due to parts of the digital circuit not having the new data ready to be latched by the D-flip-flops. The maximum sampling rate is very close to the specification of the target clock period given to the synthesizer. A shorter clock period could have been specified, but this clock period was chosen to limit the need for larger, more powerful gates in order to meet the overall area constraint.

Fig. 3.41 shows the spectrum of a 1MHz input and a sampling rate of

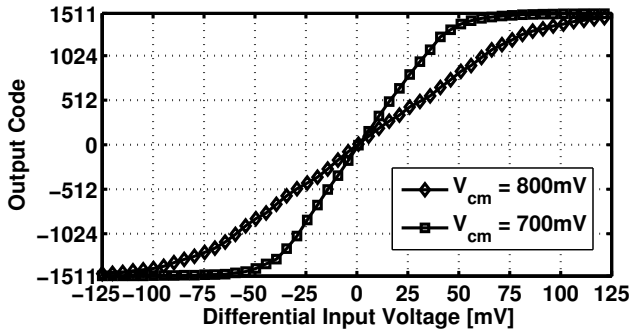


Figure 3.38: Measured output transfer function of the ADC for different input common mode voltages.

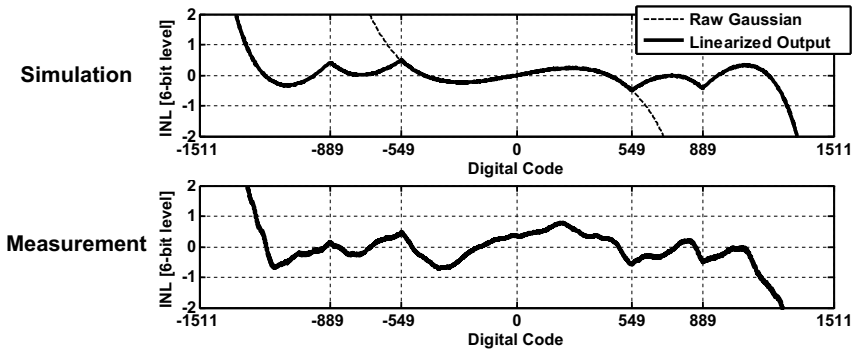


Figure 3.39: Simulated and measured INL. The effect of the piecewise un-Gaussian function can be seen.

210MSPS (with 8x decimation of the output). Sine-wave SNDR is 35.89dB with a 41.46dB spurious-free dynamic range. To achieve this performance no calibration or post-processing was required. The un-Guassian function on chip does all of the work of linearizing the output. The standard deviation of random comparator offset is measured, at a common-mode input voltage of 800mV and a 1.2V supply, to be about 46mV. The signal range, being between $\pm 1.6\sigma$ is about 280mV differentially. By using the range of $\pm 1.6\sigma$, almost 90% of all of the comparators lie within the signal range. Without linearizing the Gaussian distribution this would not be possible..

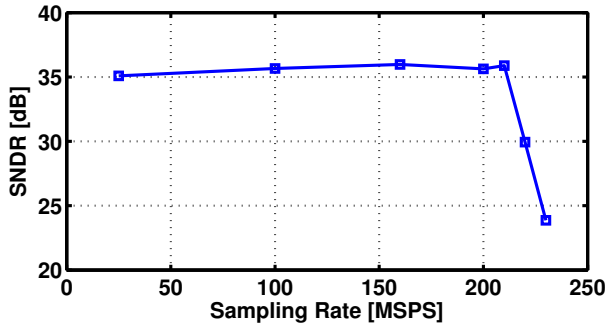


Figure 3.40: SNDR from a sine-wave test as function of sampling rate.

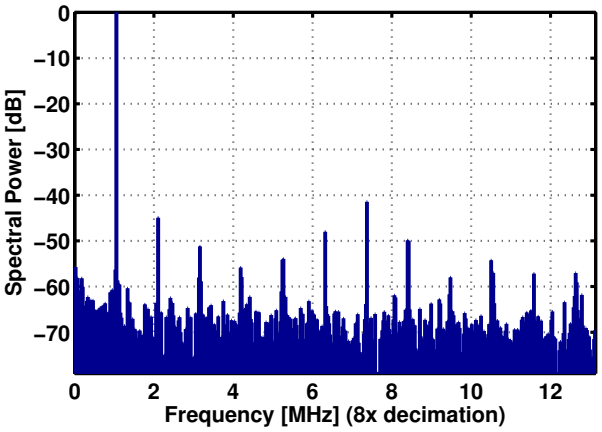


Figure 3.41: Spectral plot of 1MHz input sine-wave at 210MSPS achieving 35.9dB SNDR. 8x decimation was used.

In a conventional flash ADC, decreasing the signal amplitude decreases the SNDR by decreasing the signal power while the noise power remains fixed. In a stochastic flash ADC, the stochastic requirement of 4^N comparators for N effective bits means that there are so many more comparators than a conventional flash ADC that decreasing the signal amplitude from full scale does not necessarily mean that the SNDR will decrease proportionally. This is actually a feature of a stochastic flash ADC. Measured data of SNDR from a sine-wave test as a function of input amplitude can be seen in Fig. 3.42. Over the range of input amplitude from 1σ

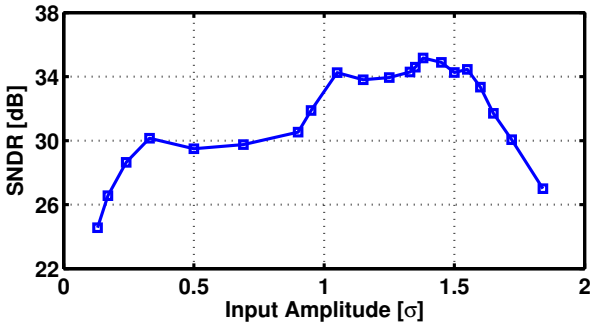


Figure 3.42: SNDR from a sine-wave test as a function of input amplitude.

to 1.6σ, the magnitude of the output in the code-domain is larger; however, the linearity is more or less constant. This could be especially valuable in cases where the magnitude of an incoming signal may not be fixed, but the required linearity is constant.

There is a significant load at the input (approximately 2.5pF) due to all of the comparator gate capacitances and the parasitic routing capacitance. This total capacitance is distributed and connected through resistive vias and metal traces that can not be considered negligible. Due to parasitic filtering of the input through the automatically synthesized input nets, there is an observed decrease in SNDR as a function of input frequency (Fig. 3.43). The roll-off at about 60MHz is predicted from extracted simulation.

3.4.2.3 Summary

This prototype IC proves that synthesizing an ADC entirely from Verilog is possible. The stochastic ADC naturally lends itself to being synthesized by the mere fact that it is designed to expect high variability; automated place-and-route will cause problems is many layout-sensitive designs, but not the stochastic ADC. A comparator that is implemented as two cross-coupled 3-input NAND

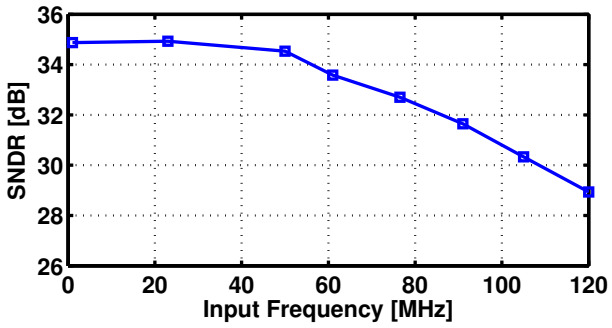


Figure 3.43: SNDR from a sine-wave test as a function of input frequency.

Table 3.2: Performance Summary

Technology	90 nm CMOS	
Max Sampling Rate	210 MSPS	
Comparator Offset Standard Deviation	45 mV	
Input Range	280 mVpp (differential)	
Supply Voltage	1.2 V	700 mV
Sampling Rate	210 MSPS	21 MSPS
Input Frequency	1 MHz	1 MHz
SNDR	35.89 dB	34.61 dB
SFDR	41.46 dB	40.81 dB
Total Power	34.8 mW	1.11 mW
Total Active Area	0.18 mm^2	

gates has been demonstrated to work effectively as a true analog comparator. By using a piecewise linear approximation of the inverse function of a Gaussian CDF, 90% of the comparators of a single Gaussian group become an effective uniform distribution to the accuracy required. More importantly, this linearizing technique requires absolutely no calibration or post processing of any kind. The result is a truly all digital ADC with the only analog input being the input signal.

```
 1  module dsp(c0b0, clk, final);
 2   output final;
 3   input c0b0, clk;
 4
 5   wire clk;
 6   wire [6:0] c0b0;
 7
 8   reg [4:0] final;
 9   reg [3:0] sum;
10   reg [4:0] sum2;
11
12   reg c1b0[2:0];
13   reg c1b1[1:0];
14
15   reg c2b0;
16   reg c2b1[1:0];
17   reg c2b2;
18
19   always @(negedge clk) begin
20
21   {c1b1[0],c1b0[0]} <= c0b0[0]+c0b0[1]+c0b0[2];
22   {c1b1[1],c1b0[1]} <= c0b0[3]+c0b0[4]+c0b0[5];
23   c1b0[2] <= c0b0[6];
24
25   {c2b1[0],c2b0} <= c1b0[0]+c1b0[1]+c1b0[2];
26
27   {c2b2,c2b1[1]} <= c1b1[0]+c1b1[1];
28
29   sum <= {c2b2,c2b1[0],c2b0}+{1'b0,c2b1[1],1'b0};
30
31   sum2 <= {c2b2,c2b1[0],c2b0}+{1'b0,c2b1[1],1'b0} - 4;
32
33   if(sum > 7)
34    final <= {sum2[3:0],1'b0} + {sum2[4],sum2[4:1]} - 4;
35   else if(sum > 6)
36    final <= (sum2) + {sum2[4],sum2[4:1]} - 1;
37   else if(sum >= 2)
38    final <= (sum2);
39   else if(sum >= 1)
40    final <= (sum2) + {sum2[4],sum2[4:1]} + 1;
41   else
42    final <= {sum2[3:0],1'b0} + {sum2[4],sum2[4:1]} + 4;
43
44   end
45
46  endmodule
```

Figure 3.44: Verilog module 'dsp' which implements the pipelined Wallace ones adder (lines 21-29) and piecewise Gaussian-to-uniform (lines 33-42).

CHAPTER 4. DOMINO LOGIC BASED ADC

An analog-to-digital (ADC) architecture that can achieve low to medium resolution and also be synthesizable is very desirable, with its main benefit being a very low design cost and high portability. One solution that has merit in this regard, as discussed in the previous section, is the stochastic flash ADC [28] and other flash ADCs that use random comparator offset to generate voltage references [19][20]. The main drawback of these types of ADCs is that they require a large area for either stochastic averaging or calibration hardware. A delay-cell based ADC architecture is another solution that is highly digital [29]. This work uses a chain of dynamic "domino-logic" cells to create a low-power, low-cost ADC with the intention of being a candidate for a synthesizable ADC.

4.1 Principle of Operation

A domino-logic based ADC is a cascade of several dynamic delay cells such as the one in Fig. 4.1 to create a circuit as depicted in Fig. 4.3. The basic operation is as follows. A clock Φ is used trigger the ADC such that when $\Phi = 1$ (implying that its compliment, $\bar{\Phi} = 0$), all of the domino cells are reset and the input voltage is sampled with a sample-and-hold into the node V_{in}. The domino cells are reset through small NMOS and PMOS switches with inputs of Φ and $\bar{\Phi}$, respectively. This resets the internal nodes of all of the cells, and crates a high-impedance state between the input, V_{in}, and the gate of the PMOS M_p. After sampling, $\Phi = 0$ ($\bar{\Phi} = 1$) and every domino cell is in a ready state with all of the transistors off

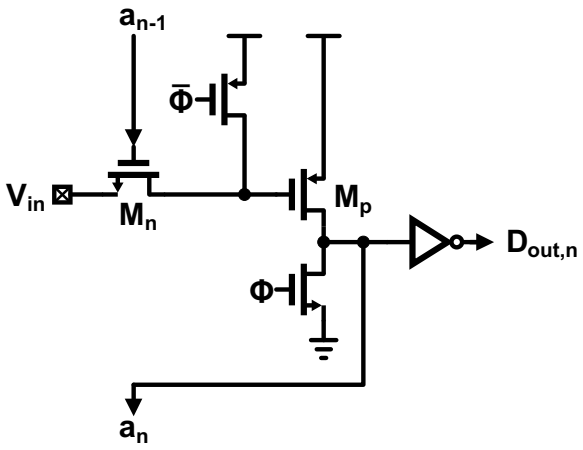


Figure 4.1: A single "domino" cell.

and in a high-impedance state. The first cell in the chain is triggered by $\bar{\Phi}$ as seen in Fig. 4.3. This causes the NMOS M_n to turn on and create a conduction path between V_{in} and the gate of M_p. In this implementation the sample-and-hold is a simple bootstrapped NMOS switch [30], therefore V_{in} is not actively being driven. This causes charge-sharing with the parasitic capacitance at the gate of M_p and the capacitance at node V_{in}. If the voltage at V_{in} is sufficiently low, the device M_p will turn on and trigger the next cell. The cells continue to trigger each other in series and then they are reset in parallel on $\bar{\Phi} = 1$. Just before resetting there is a thermometer code generated as the digital outputs D_0 through D_n that can be encoded into binary as the ADC result.

4.2 Implementation Details

Since the sampling rate of this ADC depends on the number of cells and rate at which they can propagate the digital ripple, we want to design the cell to have

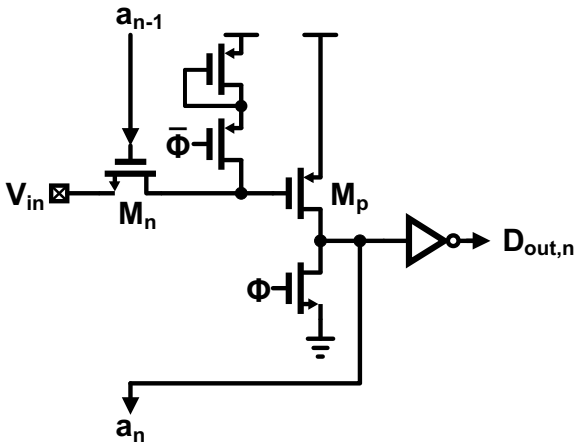


Figure 4.2: A single "domino" cell with a shorter delay.

as small of a delay possible in order to achieve a high sampling rate. The critical transistors in the delay path are M_n and M_p, whereas the reset switches and the digital buffer are less critical and only add unwanted parasitic capacitance to the internal nodes of the domino cell. Therefore the reset transistors are minimum-sized and the critical transistors are slightly larger. Increasing the sizes of M_n and M_p will also add parasitic capacitance that needs to be charged and discharged during operation. As this adds to power consumption, our design uses device widths for M_n and M_p that are just below twice minimum for the process.

Fig. 4.1 shows that the gate of the device M_p is reset to the positive supply V_{DD}. This means that when M_n is turned on, enough charge must be shared with V_{in} in order to drop the gate voltage of M_p to $V_{DD} - V_{TH}$, where V_{TH} is the threshold voltage for a PMOS, before M_p starts to turn on M_n of the next domino cell. This takes a certain amount of time. By adding a diode connected device in series with the PMOS reset switch (Fig. 4.2) the gate of M_p will be reset to near $V_{DD} - V_{TH}$ instead. This reduces the amount of charge that needs to be moved from the gate of M_p, and ultimately the amount of time, before M_p starts

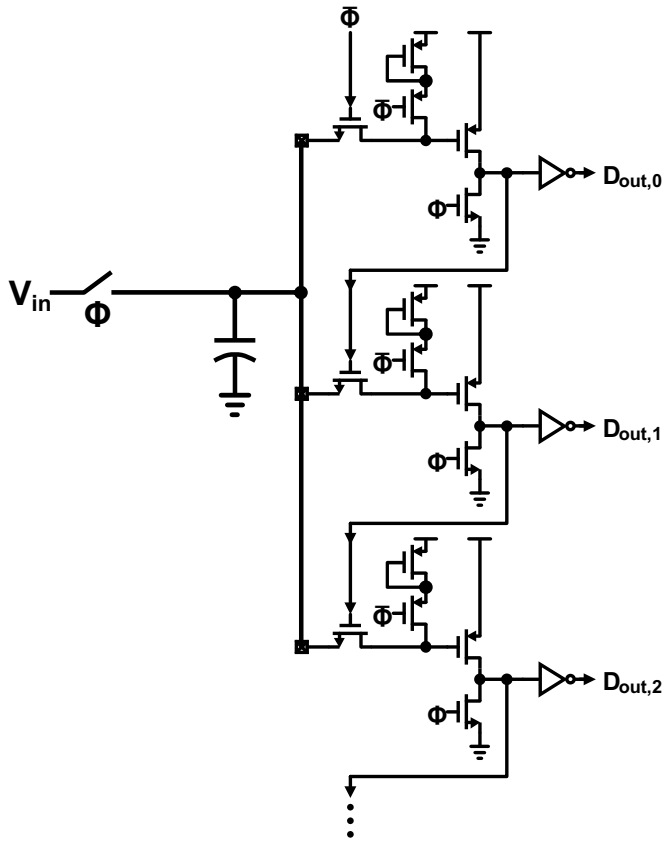


Figure 4.3: Example of multiple domino delay cells combined to form an ADC.

to turn on M_n. This would be a dangerous scenario, M_p right at the tipping-point of triggering the next cell, but when the PMOS reset switch closes there is enough charge injection that it guarantees that M_p will be off. This allows the design to operate at a higher speed without increasing power consumption by eliminating this "waste voltage."

Since the domino cells evaluate in a serial manner and are reset in parallel, more time is needed in the evaluation phase than the reset phase. Therefore the sampling/reset time is created by a pulse generator, thus devoting most of the period to the cascading domino evaluation phase. The thermometer coded

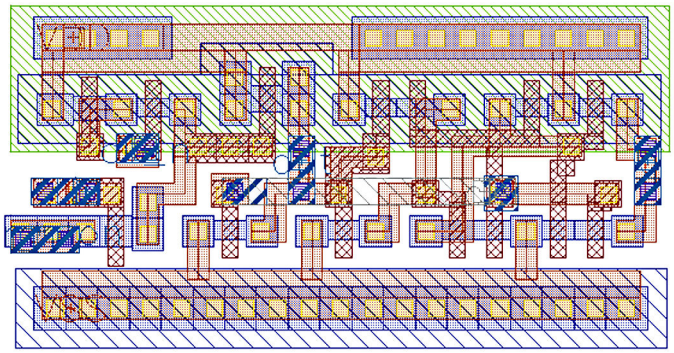


Figure 4.4: Screen capture of the physical layout of a single domino cell.

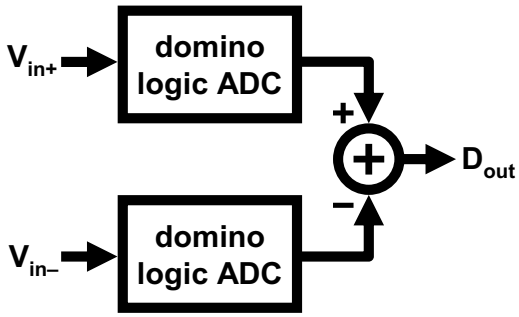


Figure 4.5: Pseudo-differential domino logic ADC.

output generated by the domino cells is latched into rising-edge triggered D-flip-flops that are latched on the rising edge of Φ. This feeds into a thermometer-to-binary converter and is given an entire clock cycle to resolve. True-single-phase-clock (TSPC) D-flip-flops [31] are incorporated into each domino cell as shown in Fig. 4.4. The thermometer-to-binary converter chosen is a simple multiplexer-based decoder [32].

By itself, this ADC suffers from high nonlinearity in the form of a dominant second harmonic. This should come as no surprise since the speed at which each cell propagates has a nonlinear relationship to V_{in}; moreover, as each cell switches, the small amount of charge sharing with the parasitic capacitance at the gate of M_p

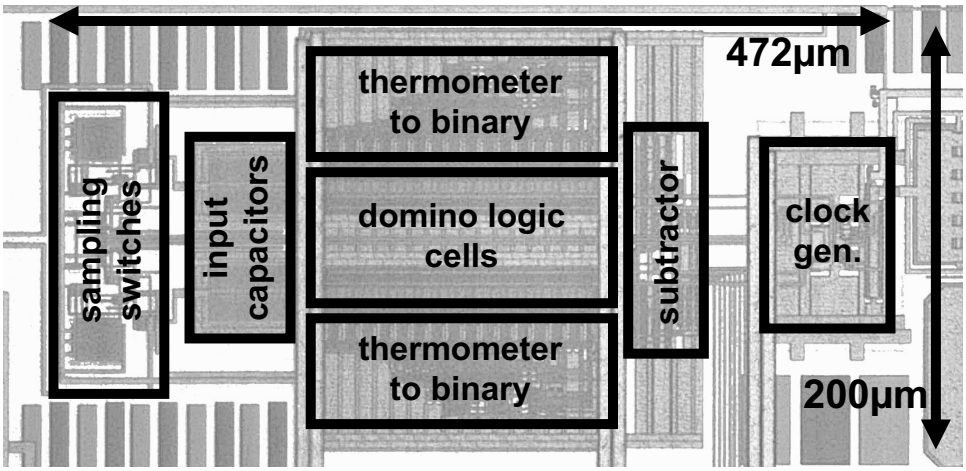


Figure 4.6: Detailed die micrograph.

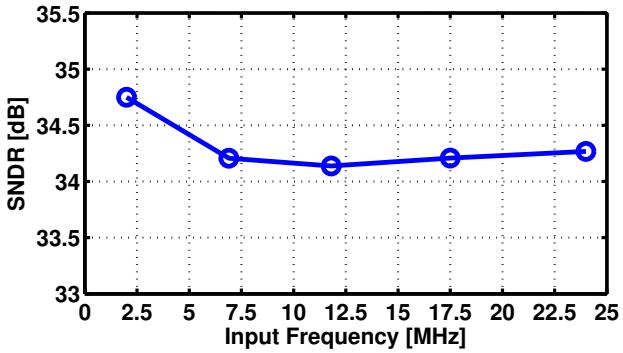


Figure 4.7: SNDR is well behaved over the Nyquist range of the converter. (f_S=50MS/s).

causes the voltage at V_{in} to increase over time in a nonlinear way. Fortunately, most of the power of the second harmonic can be canceled by implementing a pseudo-differential structure as in Fig. 4.5. The required subtraction is implemented as a digital binary adder with one of the ADC's outputs inverted by its 2's-compliment.

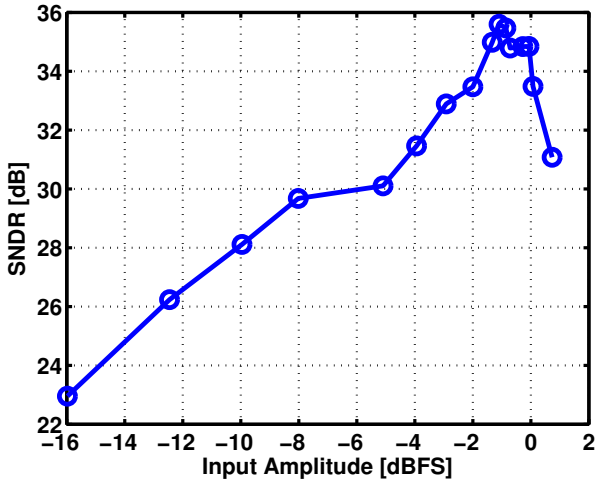


Figure 4.8: SNDR increases proportionally with the input amplitude.

4.3 Measurement Results

A test chip was fabricated in a 0.18μ digital CMOS process (Fig. 4.6). The chip area has dimensions 472μm by 200μm, consuming a total area of 0.0944mm^2. The ADC is implemented as two domino logic based ADCs in pseudo-differential configuration, each with 63 domino cells. The two 6-bit outputs are subtracted using a 6-bit ripple carry binary adder which produces a 7-bit result that is sent off chip. With a supply voltage of 1.3V, a sampling rate of 50-MHz is achieved. SNDR is relatively consistent at above 34dB across input frequency even up to Nyquist as shown in Fig. 4.7. A plot of SNDR as a function of input amplitude can be seen in Fig. 4.8. The SNDR of the ADC falls rapidly after increasing the input beyond its full scale due to clipping. For this type of ADC there is no V_{ref} voltage, so a full-scale input amplitude is in terms of the domino cells' inherent voltage-to-time transformation. Two spectral plots are provided in Fig. 4.9 and Fig. 4.10 to show the spectral output for a low frequency and a near-Nyquist frequency at the same

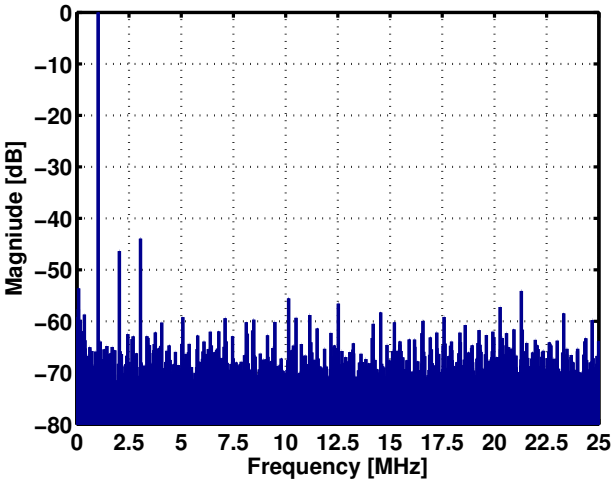


Figure 4.9: Spectral FFT plot normalized to the input with an input of 1-MHz taken at 50MS/s.

sampling rate of 50MS/s. Power consumption also varies with input frequency with 312μW consumed for a 1-MHz input and 433μW for a 24-MHz input. This is due to the fact that for low frequency inputs the thermometer-to-binary decoder switches less, especially for the MSBs since they change state less often. For near-Nyqust inputs, the thermometer-to-binary decoder may completely change its output from cycle-to-cycle.

4.4 Summary

A domino logic based ADC was presented. The use of digital, dynamic delay cells leads this design to be a good candidate for a highly scalable and synthesizable ADC. Adding this type of domino cell to a digital library would allow fully automated synthesis of this type of ADC. Of course another option to using this optimized custom cell is to use the standard cell based delay cell from

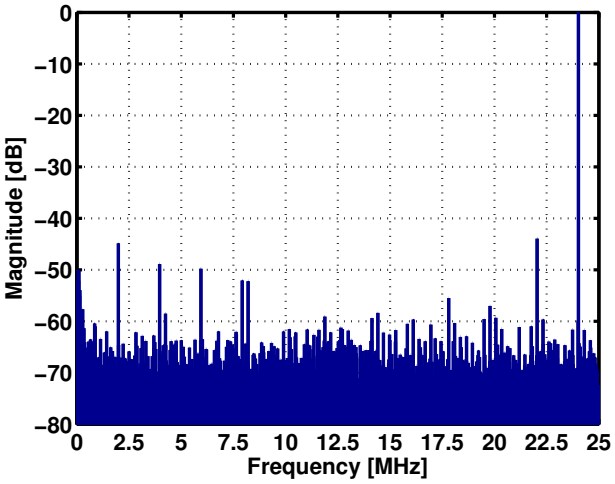


Figure 4.10: Spectral FFT plot normalized to the input with an input of 24-MHz taken at 50MS/s.

Section 2.2, the tradeoff being between performance and portability. The target application for this ADC would be one where there is a design that is predominantly a digital circuit, but an ADC is required that must be compact and low power. A test chip was fabricated in 0.18μm CMOS. The test chip achieves over 5.4b ENOB up to the Nyquist-rate of 50MS/s with a 1.3V supply. With a sampling frequency of 50MS/s and 24 MHz input, 34.2 dB SNDR is achieved while consuming 433μW and occupying only 0.094 mm^2.

Table 4.1: Performance Summary

Process Technology	0.18μm CMOS	
Resolution	7 b	
Supply Voltage	1.3 V	
Sampling Rate	50 MS/s	
Input Frequency	1.0 MHz	24.0 MHz
SNDR	35.6 dB	34.2 dB
ENOB	5.62 b	5.41 b
SFDR	44.0 dB	44.0 dB
Total Power	312 μW	433 μW
Figure-of-merit (FOM)	127 fJ/step	204 fJ/step
Total Active Area	0.094 mm^2	

CHAPTER 5. DIGITALLY IMPLEMENTED NAND-ONLY SUCCESSIVE APPROXIMATION REGISTER (DINOSAR) ADC

This section proposes another architecture for a synthesizable ADC. The successive-approximation ADC a very appealing architecture due to is low complexity and hardware requirements [33]. The basic structure can be seen in Fig. 5.1. The operation of successive approximation ADC follows that an input analog voltage is connected to a comparator that is comparing the input against a value set by a digital-to-analog converter (DAC). Multiple guesses are made by the successive approximation circuitry to pick DAC values until the analog voltage output of the DAC is very close to the analog input. Once this condition is true, the digital value sent to the DAC is also the digital expression of the input. The final DAC value is held in the successive approximation register, hence the acronym: SAR. Since this type of ADC requires only a comparator, a DAC, and digital logic, and building a comparator and a DAC element out of 3-input NAND gates was demonstrated in Section 2.2, a SAR ADC can be synthesized. Moreover, since all of the analog components (the comparator and the DAC) will be implemented as NAND gates, this ADC can be called a *d*igitally *i*mlemented *N*AND–*o*nly *SAR*, or DINOSAR.

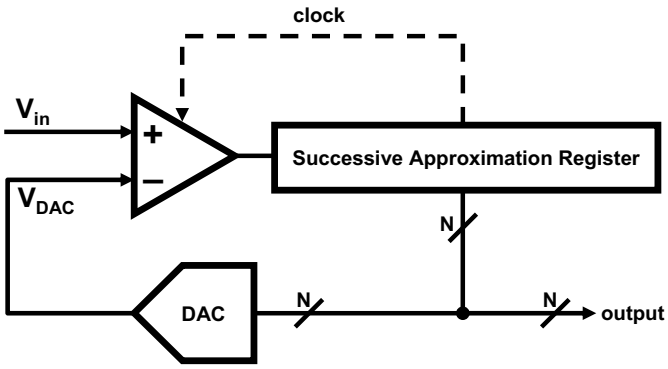


Figure 5.1: A block diagram of the basic SAR ADC.

5.1 Principle of Operation

DINOSAR is an implementation of a charge redistribution SAR (Fig. 5.2) in which the input is sampled onto one side of a set of capacitors, and the DAC operation is performed by manipulating the voltages on the other side. The reason this is called a charge redistribution SAR is because once the input is sampled, the charge at the input of the comparator remains fixed but is redistributed amongst the capacitors to maintain the relationship,

$$Q = CV, \tag{5.1}$$

where Q is charge, C is capacitance, and V is voltage.

The operation of a DINOSAR will be to reset all of the DAC bits to "1" and sample the differential input onto the total input capacitance. The sampling switches can be implemented as a single PMOS transistors, as the common-mode voltage of the input must be relatively high for the 3-input NAND based comparator. None of the input capacitance is made up of discrete drawn capacitors,

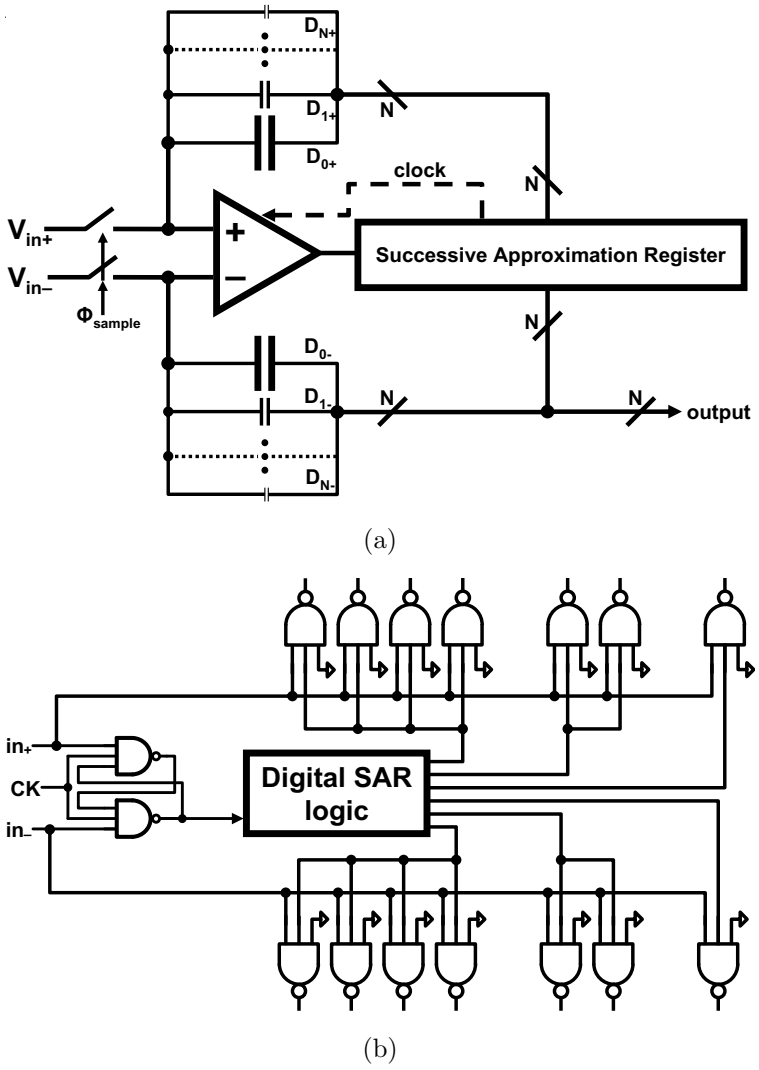
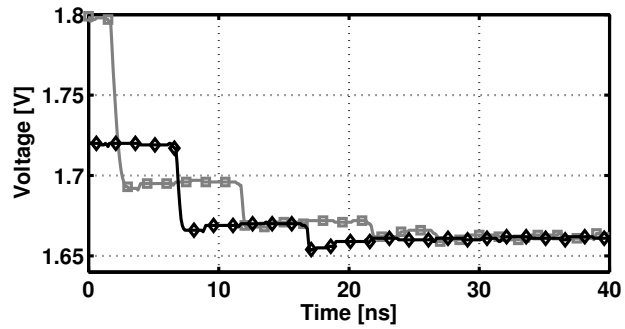
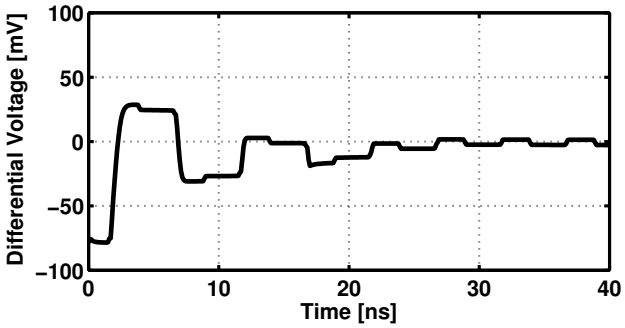


Figure 5.2: a) A block diagram of a charge redistribution SAR ADC. b) 3-bit DINOSAR example.

but rather of parasitic capacitance due to routing and transistor gate capacitance. Once the input is sampled the comparator decides if the sign of the input is differentially positive or negative. SAR logic then pulls the most-significant-bit (MSB) of the DAC low on the side with the higher input voltage causing it to be pulled down slightly. The comparator reevaluates, and the SAR operation continues until



(a)



(b)

Figure 5.3: a) A simulated example of the differential voltage at the input of the comparator during a DINOSAR conversion. b) A simulated example of the single-ended voltages at the input of the comparator during a DINOSAR conversion.

the all of the DAC bits have been set by the SAR. The final SAR value is latched as the ADC output and a new input voltage is sampled.

5.2 Simulation Results

A 6-bit DINOSAR was simulated in a 0.18μm CMOS process with layout parasitic extraction. The general operation can be seen in Fig. 5.3, where after each comparator decision the SAR directs the NAND DAC to pull down the large of the two voltages by a proportionally decreasing amount. The DAC associated

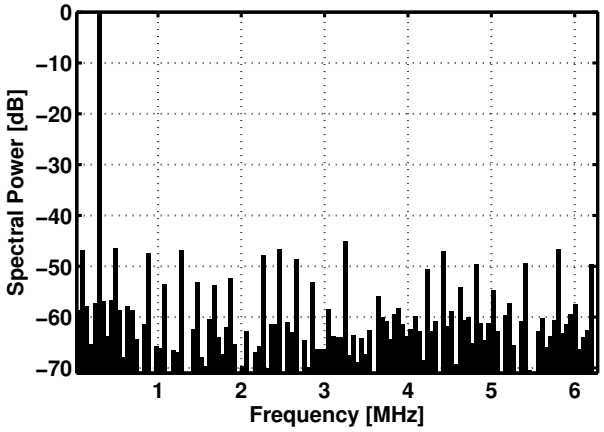


Figure 5.4: Simulated spectral plot of a 6-bit DINOSAR.

with the lower of the two voltages does not change. Looking at the input voltage differentially, it can be seen that after each cycle the differential input begins to approach zero. The small jumps on the waveform is due to kickback from the comparator.

A spectral plot of a simulated sine-wave test (Fig. 5.4) achieves SNDR of 34.7dB at 12.5MSPS with an input of frequency of 292kHz. This simulation was intended as a proof-of-concept and does not indicate that 12.5MSPS is the maximum achievable sampling rate; the actual maximum sampling rate would be much higher. The input voltage amplitude was 360mVpp, differentially.

5.3 Summary

Using only 3-input NAND gates, an analog comparator and a DAC can be constructed which allows the creation of an all standard digital cell based SAR ADC. A sample-and-hold is required for this circuit, but can easily be realized as two PMOS transistors, one for each polarity of the differential input. This type of

ADC can be expected to consume a very small area and is an excellent candidate for an ADC to be integrated into an already existing digital design that only has a small remaining area to devote to an ADC.

CHAPTER 6. CONCLUSION

The automated synthesis of analog-to-digital converters was proven to be possible using conventional digital synthesis tools. Automatic place-and-route is not a trivial feature and was notably significant upon implementing the two stochastic flash prototype ADCs. The layout for the two-group stochastic flash ADC was generated manually and took over 4 months to complete; whereas the single-group stochastic flash was synthesized and the layout was automatically generated in a few hours. The automatic layout also enabled the implementation of a true Wallace tree ones adder which would have dramatically increased the layout complexity for a manually generated layout, so was avoided in the manual case. As digital circuits quickly scale to the state-of-the-art technology nodes, there needs to exist analog options that can scale just as quickly, and this work enables high portability of analog-to-digital converters.

First, in Section 2, the constraints on the design space for analog circuits in a digital synthesis regime were considered. In order to make a digitally synthesized circuit contain analog functions, a designer must either generate a custom analog cell or use existing digitals cells in such a way that they create an analog function. There are some issues that must be addressed by the designer during synthesis and place-and-route in order to direct the software to not alter the explicitly defined analog blocks and to not use analog custom cells in place of what should be digital cells.

In Section 3, a stochastic flash ADC was analyzed with many possible design options. It is debatable that this type of ADC is the most synthesizable option

since it expects high variability by design, in fact it depends on it. Therefore when variability crops up from the way the tool decides to place-and-route the design, it just becomes part of the comparator offset distribution. Three different techniques (multi-group, two-group, and single-group) were demonstrated to generate a linear converter characteristic from the nonlinear Gaussian shape that exist by nature. The most synthesizable of the three, single-group, does not require any analog references, calibration, or post processing of any kind: the only analog input to the circuit is the signal.

Section 4 suggests another ADC architecture that is cellular and digital in nature, a domino logic based ADC. This architecture uses an analog controlled delay chain to obtain an digital output. One major benefit of this architecture is that its maximum sampling rate is dependent on the minimum time delay of each cell, and this is a characteristic that is expected to decrease with process scaling. The example implementation in this section uses a custom analog cell, but a standard digital cell delay cell could have been used. The trade off is that the custom cell will be better performing for the one task that it is optimized for, but the standard cell version can be synthesized from any digital library.

Finally, Section 5 uses both a standard digital cell based comparator and DAC cell to demonstrate the feasibility of creating a SAR ADC that could be fully synthesized from Verilog code. A benefit of this architecture, like the domino logic based ADC, is that the area requirement can be quite small compared to a stochastic flash ADC, since there is no stochastic averaging, so a trend of 2^N increased hardware for N more bits is expected.

BIBLIOGRAPHY

[1] H. Iwai, "CMOS scaling towards its limits," in *Int. Conf. on Solid-State Integrated Circuit Tech.*, Oct 1998, pp. 31–34.

[2] S. Wolf, *Silicon Processing for the VLSI Era: The Submicron MOSFET*, Lattice Press, Sunset Beach, 1995.

[3] K.Y. Fu and Y.L. Tsang, "On the punchthrough phenomenon in submicron MOS transistors," *IEEE Trans. on Electron Devices*, vol. 44, pp. 847–855, Dec 1998.

[4] T.B. Hook, "Lateral ion implant straggle and mask proximity effect," *IEEE Trans. on Electron Devices*, vol. 50, pp. 1946–1951, Sep 2003.

[5] S. Ogura *et al.*, "Design and characteristics of the lightly doped drain-source (LDD) insulated gate field-effect transistor," *IEEE J. Solid-State Circuits*, vol. 15, no. 12, pp. 424–432, Dec 1998.

[6] S. Wolf, *Silicon Processing for the VLSI Era: Deep-Submicron Process Technology*, Lattice Press, Sunset Beach, 2002.

[7] G. Moore, "Cramming more components onto integrated circuits," *Electronics*, vol. 38, no. 8, pp. 114–117, Apr 1965.

[8] B. Spencer, L. Wilson, and R. Doering, "The semiconductor technology roadmap," *Future Fab. Int.*, vol. 18, Jan 2005.

[9] B. Spencer, L. Wilson, and R. Doering, "Analog circuits in ultra-deep-submicron CMOS," *IEEE J. Solid-State Circuits*, vol. 40, no. 1, pp. 132–143, Jan 2005.

[10] T. Watanabe, T. Mizuno, and Y. Makino, "An all-digital analog-to-digital converter with 12- mu;v/lsb using moving-average filtering," *Solid-State Circuits, IEEE Journal of*, vol. 38, no. 1, pp. 120 – 125, jan. 2003.

[11] M.A. Farahat, F.A. Farag, and H.A. Elsimary, "Only digital technology analog-to-digital converter circuit," dec. 2003, vol. 1, pp. 178 – 181.

[12] H. Farkhani, M. Meymandi-Nejad, and M. Sachdev, "A fully digital adc using a new delay element with enhanced linearity," may. 2008, pp. 2406 –2409.

[13] Marcelo Negreiros, Luigi Carro, and Gustavo Cassel, "All digital adc with linearity correction and temperature compensation," may. 2010, pp. 147 –152.

[14] B. Razavi and B.A. Wooley, "Design techniques for high-speed, high-resolution comparators," *IEEE J. Solid-State Circuits*, vol. 27, no. 12, pp. 1916–1926, Dec 1992.

[15] B.P. Brandt and J. Lutsky, "A 75-mW, 10-b, 20-MSPS CMOS subranging ADC with 9.5 effective bits at nyquist," *IEEE J. Solid-State Circuits*, vol. 34, no. 12, pp. 1788–1795, Dec 1999.

[16] M.R. Tursi and R. Taft, "A 100-MS/s 8-b CMOS subranging ADC with sustained parametric performance from 3.8 V down to 2.2 V," *IEEE J. Solid-State Circuits*, vol. 36, no. 3, pp. 331–338, Mar 2001.

[17] J. Mulder *et al.*, "A 21-mW 8-b 125-MSample/s ADC in 0.09-mm^2 0.13-μm CMOS," *IEEE J. Solid-State Circuits*, vol. 39, no. 12, pp. 2116–2125, Dec 2004.

[18] J.L. Ceballos, I. Galton, and G.C. Temes, "Stochastic analog-to-digital conversion," in *48th Midwest Symp. on Circuits and Syst.*, Aug 2005, pp. 855–858.

[19] D.C. Daly and A.P. Chandrakasan, "A 6b 0.2-to-0.9v highly digital flash ADC with comparator redundancy," *IEEE J. Solid-State Circuits*, vol. 44, no. 11, pp. 3030–3038, Nov 2000.

[20] C. Donovan and M.P. Flynn, "A 'digital' 6-bit ADC in 0.13 μm CMOS," *IEEE J. Solid-State Circuits*, vol. 37, no. 3, pp. 432–437, Mar 2002.

[21] S.C. Wong, K.H. Pan, and J.A. Ma, "A CMOS mismatch model and scaling effects," *IEEE Electron Device Letters*, vol. 18, no. 6, pp. 261–263, Jun 1997.

[22] H. Stark and J.W. Woods, *Probability and Random Processes with Applications to Signal Processing*, chapter 4, pp. 225–230, Prentice Hall, Upper Saddle River, 2001.

[23] H. Stark and J.W. Woods, *Probability and Random Processes with Applications to Signal Processing*, chapter 1, pp. 40–41, Prentice Hall, Upper Saddle River, 2001.

[24] Analog Devices Inc., *Data Conversion Handbook*, chapter 2, p. 69, Newnes, Burlington, 2005.

[25] H. Stark and J.W. Woods, *Probability and Random Processes with Applications to Signal Processing*, chapter 6, pp. 383–387, Prentice Hall, Upper Saddle River, 2001.

[26] F. Kaess *et al.*, "New encoding scheme for high-speed flash ADC's," in *Int. Conf. on Solid-State Integrated Circuit Tech.*, Jun 1997, pp. 5–8.

[27] P. Nuzzo *et al.*, "Noise analysis of regenerative comparators for reconfigurable ADC architectures," *IEEE Trans. Circuits Syst. I*, vol. 55, no. 6, pp. 1441–1454, Jul 2008.

[28] S. Weaver *et al.*, "A 6b stochastic flash analog-to-digital converter without calibration or reference ladder," *IEEE Asian Solid-State Circuits Conf.*, pp. 373–376, Nov 2009.

[29] 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, Mar 2003.

[30] M. Dessouky and A. Kaise, "A 1V 1mW digital-audio $\delta\sigma$ modulator with 88dB dynamic range using local switch bootstrapping," *IEEE J. Solid-State Circuits*, vol. 36, no. 3, pp. 349–355, Mar 2001.

[31] J. Yuan and C. Svensson, "High-speed CMOS circuit technique," *IEEE J. Solid-State Circuits*, vol. 24, no. 1, pp. 62–70, Feb 1989.

[32] E. Sail and M. Vesterbacka, "A multiplexer based decoder for flash analog-to-digital converters," in *IEEE Region 10 Conf. TENCON 2004*, Nov 2004, vol. 4, pp. 250–253.

[33] D.A. Johns and K. Martin, *Analog Integrated Circuit Design*, John Wiley & Sons, Toronto, 2005.