An 8-b, 1.8 V, 20 MS/s Analog to Digital Converter

Douglas R. Beck

A dissertation submitted in partial fulfillment of the requirements of the degree of

Doctor of Philosophy

University of Washington

2002

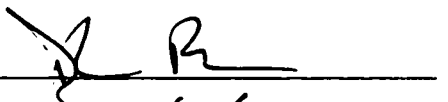
Program Authorized to Offer Degree: Department of Electrical Engineering

UMI Number: 3062916

UMI Microform 3062916

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

In presenting this dissertation in partial fulfillment of the requirements of the Doctoral degree at the University of Washington, I agree that the Library shall make its copies freely available for inspection. I further agree that extensive copying of the dissertation is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Requests for copying or reproduction of this dissertation may be referred to ProQuest Information and Learning, 300 North Zeeb Road, Ann Arbor, MI 48106-1346, to whom the author has granted "the right to reproduce and sell (a) copies of the manuscript in microform and/or (b) printed copies of the manuscript made from microform."

Signature 

Date 8/12/02

University of Washington
Graduate School

This is to certify that I have examined this copy of a doctoral dissertation by

Douglas R. Beck

And have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Chair of Supervisory Committee:

David Allstot

Reading Committee:

David Allstot

David Allee

Richard Shi

Date: Aug. 12, 2002

University of Washington

Abstract

An 8-b, 1.8 V, 100 MS/s Analog to Digital Converter

Douglas R. Beck

Chair of the Supervisory Committee:

Professor David J. Allstot
Department of Electrical Engineering

Excellent

Ever increasing market-driven demand for System-on-a-Chip (SOC) integration means that analog and RF circuits should operate along side their digital counterparts at low voltage levels with low power dissipation, high manufacturing yield, high noise immunity, and few, if any, off-chip components. Specific design practices and circuit techniques are used to achieve these goals. Utilizing simple low-gain amplifiers and calibration techniques, maximum converter bandwidth and minimum power consumption are achieved. Errors introduced by circuit techniques and non-ideal effects are compensated through smart design and digital error correction. The resulting ADC is conducive to system integration without trading off circuit performance and also alleviates the need for special circuit considerations such as separate power supplies or external circuit components.

TABLE OF CONTENTS

LIST OF FIGURES

Figure **Page**

LIST OF TABLES

Acknowledgments

First and foremost, I would like to thank my advisor Prof. David Allstot for his guidance and encouragement, friendship, and for making this research possible. He's an outstanding teacher and researcher with a real knack for defining and funding projects that are exciting, technologically challenging, and interesting to industry. I would also like to thank the other committee members Prof. David Allee, Douglas Garrity of Motorola, Prof. Mani Soma, Prof. Richard Shi, and Prof. Walter Ruzzo.

I like to thank the Semiconductor Research Corporation (SRC) and Motorola, Inc. for providing a SRC Graduate Fellowship and the Center for Design of Analog-Digital Integrated Circuits (CDADIC) for their generous funding of this research project. I would also like to thank Texas Instruments who also funded this research during its early stages.

Dedication

To my wife and family

An 8-b, 1.8 V, 20 MS/s Analog to Digital Converter

Chapter 1 Introduction

1.1 Introduction to Analog to Digital Converters

Complex electronic systems are pervasive in today's world. There has been a strong trend over the past few decades to migrate towards digital designs. Although there are many advantages to digital systems, the implementation of entirely digital designs in the near future is not likely. The primary reason most digital designs of today contain analog circuitry is that naturally occurring signals are analog. Most electronic systems need to interface to the "real world". Whether a digital system is receiving input information from a transducer, for example, or outputting information to an antenna, the resulting interfacing circuitry will contain some interface to the analog world.

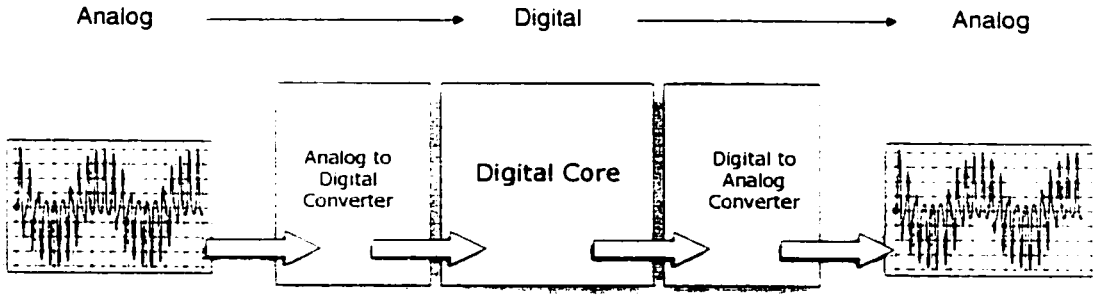


Fig. 1 System level view.

Fig. 1 shows a generalized electronic system. The input to the digital system is analog. This analog input is quantized to a discrete digital signal by the analog to digital

converter (ADC). Once digitized, the input information is available to the digital core of the system for processing. The system may also generate output. In many cases a portion or all of the output data is converted to analog output signals. This conversion to a continuous analog waveform is performed by a digital to analog converter (DAC).

Examples of digital systems that require interfacing to the analog world are found everywhere. A few examples include cell phones, televisions, modems, washing machines, and digital cameras. Virtually every product on the consumer market needs some form of analog interfacing.

The topic of this paper is ADCs in particular. The need to receive and convert incoming analog input signals in virtually all digital systems today results in a widespread demand for all types of ADCs.

This paper will review many architectures of Nyquist sampling ADCs in Chapter 2. Once the groundwork for ADCs is presented, Chapter 3 will focus on error correction techniques. Chapter 4 with present a high speed pipelined ADC design. Chapter 5 discusses the development and use of a behavioral model of high speed pipelined ADCs. Finally, Chapter 6 presents future research work to be performed.

Chapter 2 Literature Review

This chapter reviews ADCs and prepares the reader for a technical discussion of pipelined analog to digital converters.

Fig. 2 shows the analog to digital interface. Before the analog signal is sampled, it is sent through a low-pass filter that removes unwanted components of the signal and noise beyond the sampling frequency to ensure that unwanted components are removed before they alias down into the band of interest. After the filtering, the signal is sampled to produce a discrete-time signal with a continuous amplitude. Finally, the discrete-time signal is quantized to a fixed level and represented by a digital bit sequence.

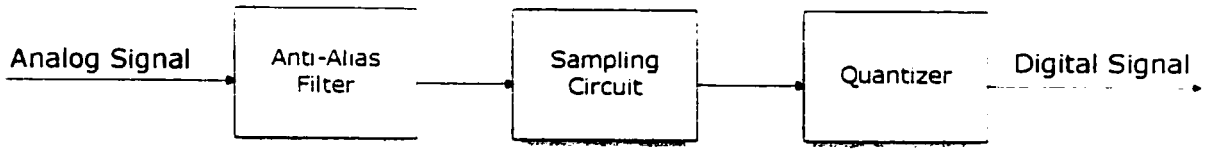


Fig. 2 Analog to digital interface.

This chapter begins by presenting the Flash ADC architecture, which is arguably the most straightforward ADC architecture. This section is followed by an explanation of several well-known Nyquist ADC architectures. The chapter concludes by presenting the pipelined ADC architecture.

2.1 Flash A/D Converter

The flash ADC is conceptually the simplest and potentially the fastest ADC architecture [41-43]. Shown in Fig. 3 the converter itself is comprised of three main components: a resistive ladder, a comparator bank, and decode logic

[1]. The resistive ladder generates the discrete voltage levels that are eventually represented by digital output codes. A reference voltage is generated by the ladder for each possible digital output code. As a result, an n-bit ADC contains 2^n resistive segments.

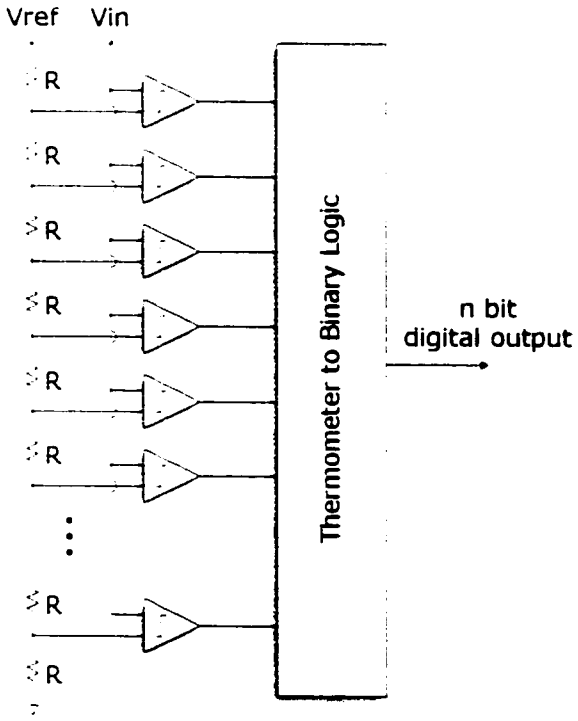


Fig. 3 Flash A/D converter architecture.

The comparators compare the reference voltages against the input signal. Similar to the resistive ladder, for an n-bit flash converter, there are 2^n comparators. The combined comparator output yields a thermometer code. The term thermometer code alludes to the output of the comparators under normal circumstances being a sequence of 0's followed by a sequence of 1's. The number of 1's with respect to the number of 0's are proportional to the analog input voltage level. In effect, the output from the comparators is read similar to a thermometer. Although this may be useful in some applications, a more appropriate digital output would be in binary format. With the addition of decode logic, the comparator outputs are converted to binary.

The size and power of a flash ADC increase as 2^n for an n-bit converter. This is due to the requirement of a comparator and a reference level for each of the 2^n levels. The benefits of this massively parallel structure is that irregardless of the number of bits being quantized, the time it takes to convert the analog input to a digital level is approximately 1 clock cycle.

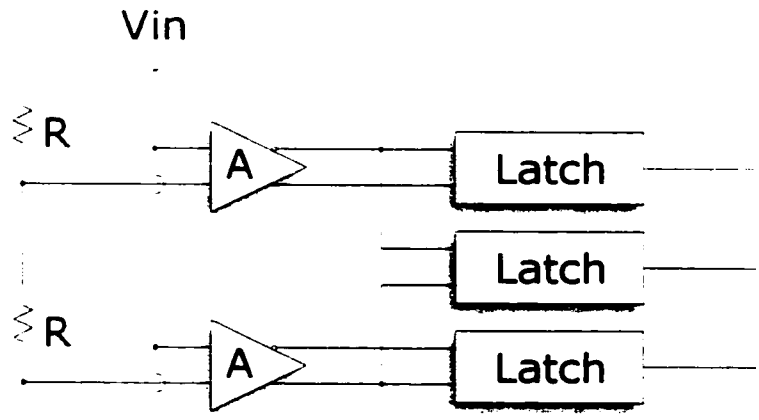


Fig. 4 Interpolation approach.

One approach to reduce the exponential complexity of the flash converter is to use interpolation. [42][43] Fig. 4 shows an implementation of interpolation. The comparators are replaced with fully differential preamplifiers having finite gain. Fig. 5 shows the differential output waveforms of the preamplifiers. Since the preamplifiers are referenced to different voltages from the reference ladder, their outputs will be shifted slightly with respect to one another. This is shown in Fig. 5 as three comparison points as compared to the non-interpolated case that would have a single comparison point. Interpolation takes advantage of the fact that by taking the difference of the two preamplifier outputs, a new comparison output is generated. The resulting circuit reduces the number of preamplifiers needed by 2. As a result, the input capacitance is also reduced. Since this circuit reduces the number of preamplifiers required by 2, the circuit has an interpolation factor of 2.

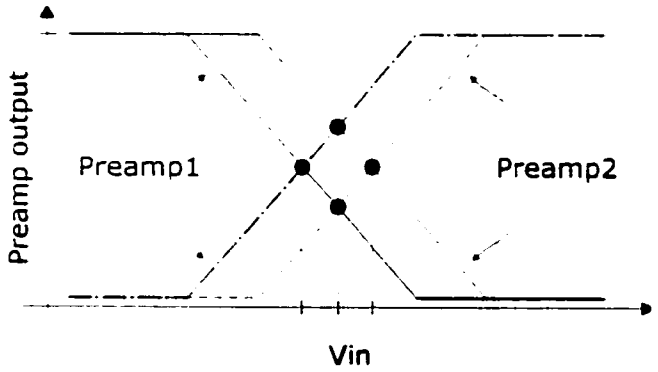


Fig. 5 Preamp output waveforms.

This circuit can be modified to generate more interpolated outputs per preamplifier. This would in turn reduce the input capacitance and complexity of the circuit as well as increasing the interpolation factor. By increasing the interpolation factor however there are many trade-offs resulting in undesirable effects.

In general, flash ADC's tend to consume large amounts of power and die area. Current technology limits high performance flash ADC designs to approximately 8 bits. Beyond 8 bits, many undesirable effects begin to degrade the performance and accuracy of the converter dramatically. Although, reasonable trade-offs between high performance and sub-optimal performance characteristics have been chosen to create exceptionally high performance flash ADC designs.

2.2 Two-Step Flash Converter

The two-step ADC architecture attempts to trade-off the speed of a flash converter with power, area, and input capacitance.[5] This becomes a more practical approach than direct flash conversion as the number of conversion bits increase. Fig. 6 shows the general approach of the two-step conversion. The conversion is divided between a coarse and a fine conversion. Combined, these two conversion outputs make up the digital output value. The coarse and fine ADCs are typically implemented with flash converters. The coarse ADC digitizes the sampled analog input voltage. The digitized value from the coarse ADC makes up the MSBs or upper portion of the digital output value. Once digitized, this coarse approximation is subtracted from the sampled input voltage creating a residue voltage. The residue voltage is converted by the fine ADC to create a digitized

value. The digitized values from the coarse and fine converters are combined to create a single digital output value.

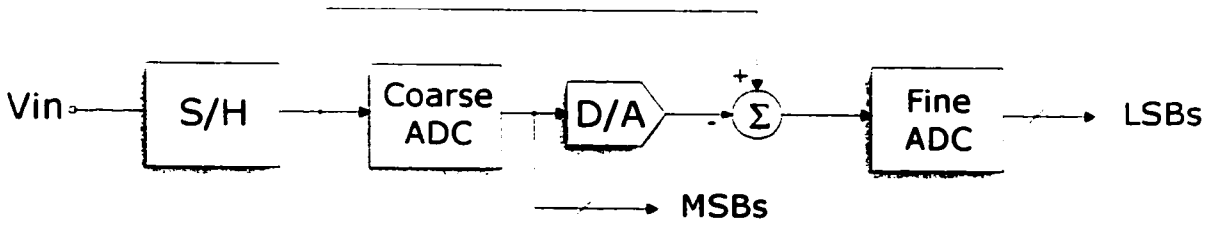


Fig. 6 Two-step architecture.

The analog input waveform is digitized by the two-step approach in 2 clock cycles. The latency of a single conversion is double of a simple flash converter. The resulting size, power, and input capacitance of the two-step approach are greatly reduced from the simple flash converter approach.

Fig. 7 shows one possible implementation technique (sample and hold circuitry not shown) using flash converters. Once the coarse conversion takes place, the DAC converts the coarse conversion output back to an analog voltage that approximates the sampled analog input voltage. The DAC voltage is subtracted from the sampled analog input voltage to create a residue voltage. This residue voltage is then passed to the fine flash converter that completes the overall analog to digital conversion.

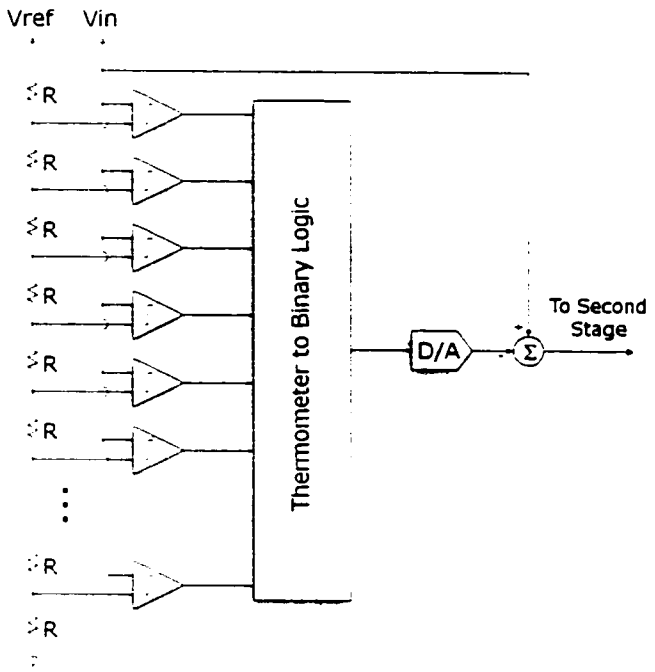


Fig. 7 Two-step flash converter.

The sample and hold (S/H) at the front end ensures that both conversions operate on the same input voltage. Without the S/H, the DAC voltage would be subtracted from a more recent input voltage than was quantized by the coarse ADC creating an inaccurate residue voltage. If the analog input remained at a constant DC level, this would not present a problem and the S/H could be removed. As the input frequency increases, the residue voltage will become increasingly inaccurate.

2.3 Interpolating / Folding A/D Converters

The two-step conversion approach was developed to reduce the steep trade-offs of a full flash converter. One of the major performance limiting components of the two-step approach is the sample and hold circuitry. The sample and hold circuitry is required to

ensure that both the first and second stage ADCs operate on the same analog input voltage. The folding architecture was developed to remove the S/H from the two-step converter [36-38]. It is designed to calculate the residue directly from the analog input voltage and at the same instant in time as the course conversion without the use of a DAC or a subtractor [3] [4]. Since the residue is calculated at the same instant that the course flash converter is quantizing the analog input, the previously required S/H is no longer needed.

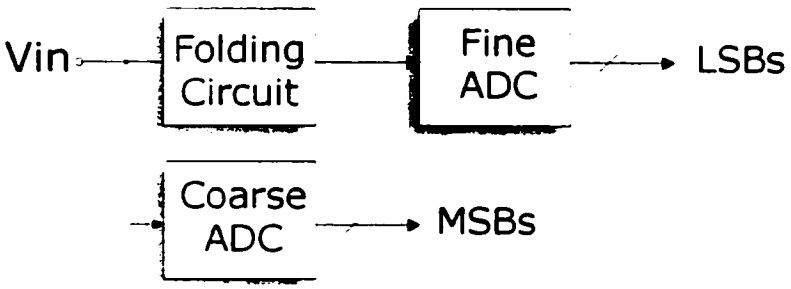


Fig. 8 Folding converter architecture.

Fig. 8 shows the folding converter architecture. When compared to the two-step architecture, the folding architecture appears superior. The speed and simplicity of folding circuitry makes it a desirable solution for many circuit designs. Unfortunately, these circuits have several disadvantages that limit their applications. One of the most obvious is that the folding circuit requires higher bandwidth than the maximum input frequency. For a folding factor of 4, as the input voltage ramps across all possible input voltages, the folding circuit oscillates 4 times. This implies that the folding circuit will require 4 times greater bandwidth than the maximum input frequency. Therefore, for high speed systems, the folding factor is limited and as a result the resolution of the ADC

is also limited. Another disadvantage of the folding circuitry is that it introduces nonlinearity into the residue as the folding output increases from 0 volts. This also limits the resolution of the ADC.

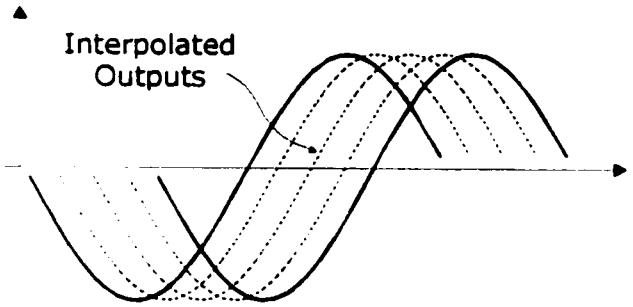


Fig. 9 Interpolation of folding outputs.

One final progression of this circuit topology is to combine folding with the interpolation discussed in a previous section with regard to flash conversion. From the previous discussion regarding the folding circuitry it was determined that the nonlinearity introduced by the folding circuit decreases to 0 as the differential output decreases to 0. Therefore, if we only looked at the zero crossings of the folding circuitry, we would not introduce any non-linearities (to a first order) into the output. This makes it possible to design high performance folding A/D converters without introducing non-linearities as a result of the folding circuitry.

2.4 Integrating A/D Converters

For applications that do not require high speed or high throughput conversion, integrating ADCs tend to trade-off conversion speed for die area and improved accuracy. Two types of integrating converters are presented in this section: single slope and dual slope converters. Both converters require a substantial amount of time to complete a conversion but at the same time, they provide a highly accurate digital output.

2.4.1 SINGLE SLOPE

The single slope approach is rather straightforward to understand. [5] The architecture for the single slope approach is shown in Fig. 10. Before the start of each conversion, the controller sets the reference voltage across the capacitor to 0 volts by shorting across the capacitor's terminals. Once the conversion process begins, the independent current source injects a constant current onto the capacitor. This in turn increases the voltage across the capacitor linearly. With the help of a digital counter circuit, the control circuitry measures the amount of time it takes for the voltage across the capacitor to reach the input voltage. The comparator signals the controller when the capacitor voltage is greater than or equal to the input voltage. The counting stops and the quantized digital value is available on the outputs of the counter.

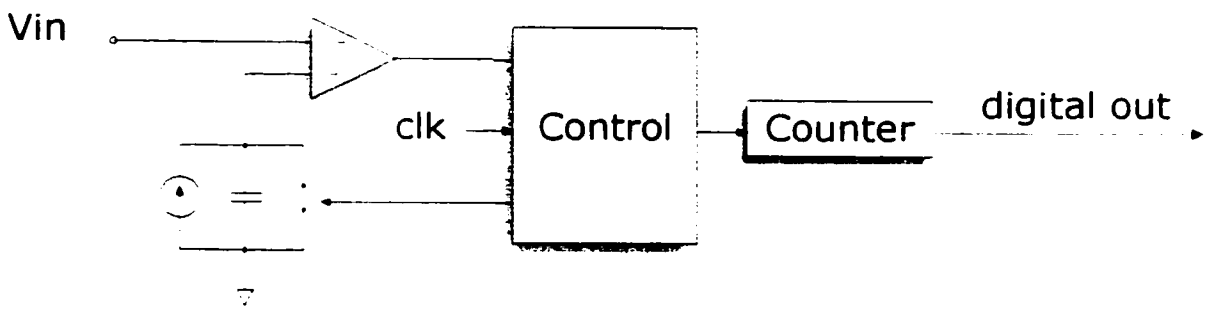


Fig. 10 Integrating A/D single slope converter architecture.

With careful design of the charging circuit to ensure linearity and employing a low-offset high accuracy comparator, high resolution outputs are obtained. Typically, integrating converters have greater than 12-bit accuracy. In applications where speed of conversion is not necessary, integrating converters may be a nice fit.

2.4.2 DUAL SLOPE

In integrated converter designs where ramp linearity or comparator offsets are the limiting factor in the design, the dual slope converter concept should be used to improve output performance. [44]

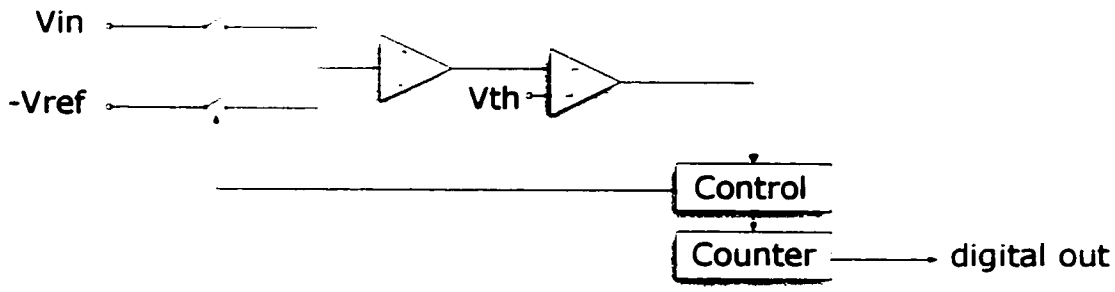


Fig. 11 Integrating A/D dual slope converter architecture.

Shown in Fig. 11 is the dual slope converter architecture. The design is very similar to the simple single slope case, with a few key modifications. The dual slope approach includes an integrator. The integrator is used to remove ramp nonlinearity and comparator offset.

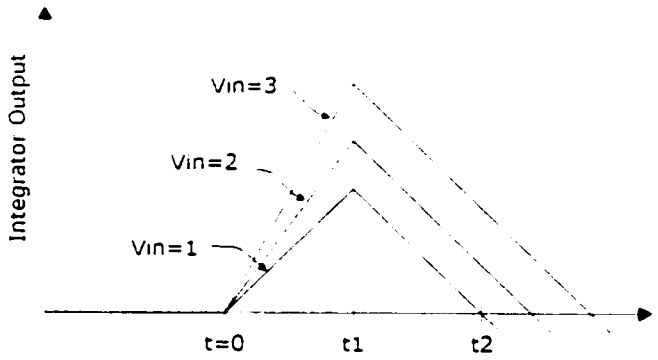


Fig. 12 Dual slope integrator output waveforms.

The conversion process begins by resetting the integrator output to 0. After the integrator is reset, the input voltage is presented to the integrator for a time period t1 (as shown in Fig. 12.) Different values for the input voltage create varying integrator charge rates until the input voltage is disconnected from the integrator at time t1. Prior to time t1, the digital counter is reset to 0. Once time t1 is reached, the negative reference voltage is input to the integrator. Also at time t1, the digital counter begins counting. Since the reference voltage is fixed, the integrator will always discharge at a constant rate. Once the integrator is completely discharged, the comparator signals the controller and the counter is halted. At this point, the quantized output is available on the outputs of the digital counter.

2.5 Successive Approximation A/D Converters

The successive approximation converter uses a "binary weighted" search algorithm . The algorithm takes one clock cycle for each bit resolved [40]. Therefore, it takes n clock cycles to complete an n-bit conversion.

Fig. 13 shows the architecture of a successive approximation converter. The converter is composed of a comparator, a DAC, a control unit, and a successive approximation register (SAR).

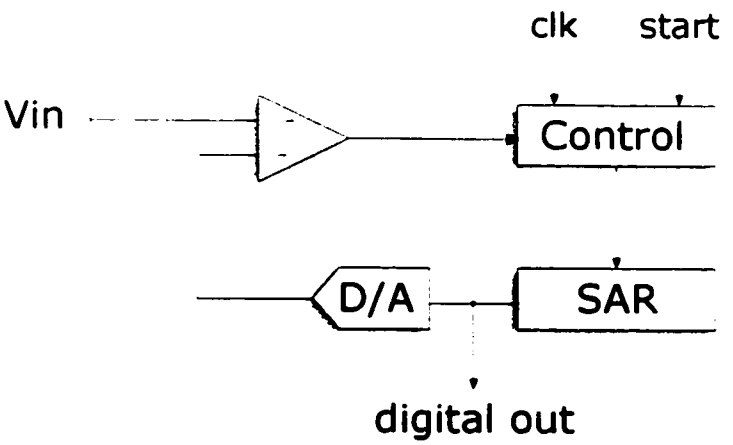


Fig. 13 Successive approximation A/D converter architecture.

The conversion begins with the DAC output set to half of full scale. The held analog input voltage is compared against the DAC output. At this point, the comparator determines if the DAC voltage is above or below the input voltage and also determines the MSB of the digital converter output. On the following clock cycle, the comparator

determines the next bit in the conversion. This process is repeated until the conversion of n bits is completed.

2.6 Cyclic A/D Converters

The cyclic converter uses an algorithm similar to the successive approximation algorithm in that they both use a "binary weighted" approach. [5] They differ in that the successive approximation adjusts the reference voltage whereas the cyclic converter adjusts the input signal. The cyclic converter accomplishes this by creating a residue voltage that is fed back into the converter.

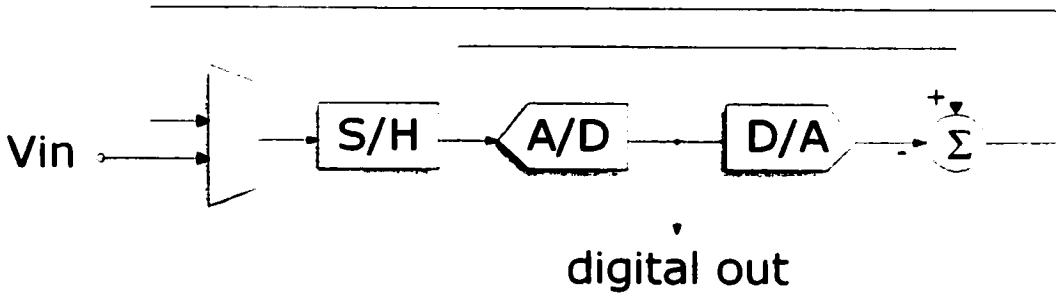


Fig. 14 Cyclical converter architecture.

Fig. 14 shows the cyclical architecture. The cyclical converter contains a multiplexor, a sample and hold, an ADC, a DAC, and a subtractor. The cyclical converter begins by sampling the analog input voltage onto the S/H. At this point, the analog signal is converted by an ADC. This internal ADC is typically of low resolutions (1-2 bits). These bits make up the MSBs of the converted analog input voltage. Next, the coarse quantized value is now converted back to an analog voltage with the DAC. This

analog voltage is then subtracted from the sampled input voltage to create a residue voltage. Once the residue voltage is created, it is fed back to the input and sampled onto the S/H. The residue voltage is quantized. These quantized bits make up the next bits in the digital output value. The process is continued until significant resolution is obtained.

One significant advantage the cyclical converter has over the successive approximation converter is that the cyclical converter can convert more than a single bit per clock cycle. In the successive approximation algorithm for an n-bit conversion, the approach takes n clock cycles. For the cyclical converter, the amount of clock cycles it takes is inversely proportional to the number of bits that the ADC quantizes each clock cycle.

2.7 Pipelined A/D Converters

The pipeline converter is a nice extension to the cyclical approach. [12][14][16-18][26-30][32][33][35][45] The pipeline converter trades off die area and power consumption for throughput. This trade-off is achieved by modifying the cyclical architecture.

The first modification to the cyclical architecture is called "loop unrolling". This technique is used extensively in software compilers when optimizing for execution speed. Fig. 15 illustrates the concept of loop unrolling. The idea is if you're going to perform a recurring operation n times, if n is a fixed constant, the same functionally can be implemented by n identical occurrences of the operation.

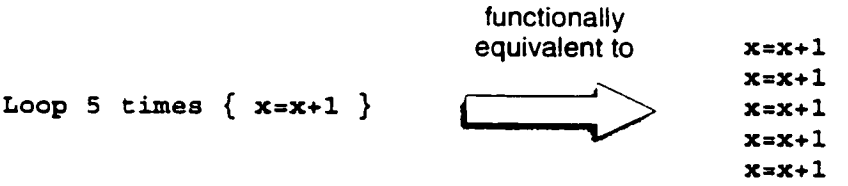


Fig. 15 Loop unrolling.

This is more easily understood when comparing the simplified architectures of the cyclical converter with the pipelined converter. Fig. 16 shows a comparison of the two implementations. In the cyclical approach, the residue voltage is fed back to the input. This essentially gives the cyclical hardware the opportunity to perform multiple A/D conversions on the incoming analog voltage. Replicating the cyclical stage creates the same functionality by feeding the output of each stage into the input of the next stage.

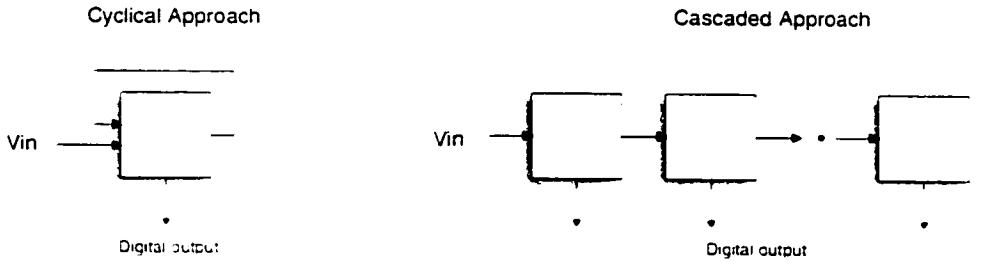


Fig. 16 Application of loop unrolling.

The modified cyclical architecture is shown in Fig. 17. The modified architecture creates a residue voltage that is passed on to the next stage. By "unrolling the loop" as described above, there is no direct performance increase as a result. In software, there is an improvement in execution time because the software does not have to check at the

completion of each loop to determine if the loop is complete. In hardware, this comparison could be done simultaneously and would not require any additional time. Therefore, additional modifications are made to the cascaded architecture described above to produce performance improvements.

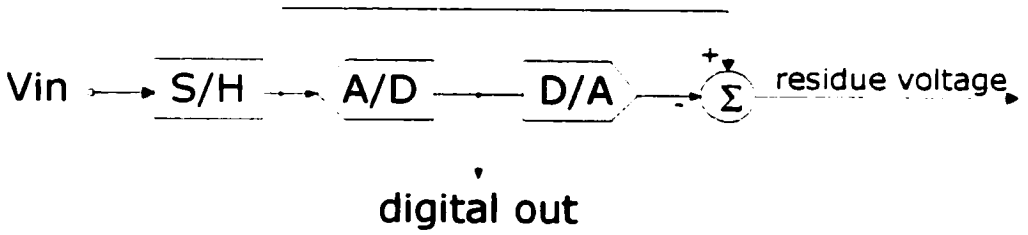


Fig. 17 Cascadable cyclical stages.

The second modification to the cyclical approach involves the concept of pipelining. Pipelining is primarily used in the design and implementation of high speed microprocessors to increase the instruction throughput. This digital concept of pipelining can also be applied to analog circuits. The implementation of pipelining involves the inclusion of latches at regular intervals in the pipeline. In our cascaded design as described above this is relatively easy. Since all the stages in the pipeline are identical, placing the latches between each stage will space the latches at nice regular intervals. The addition of latches in the cascaded design is shown in Fig. 18.

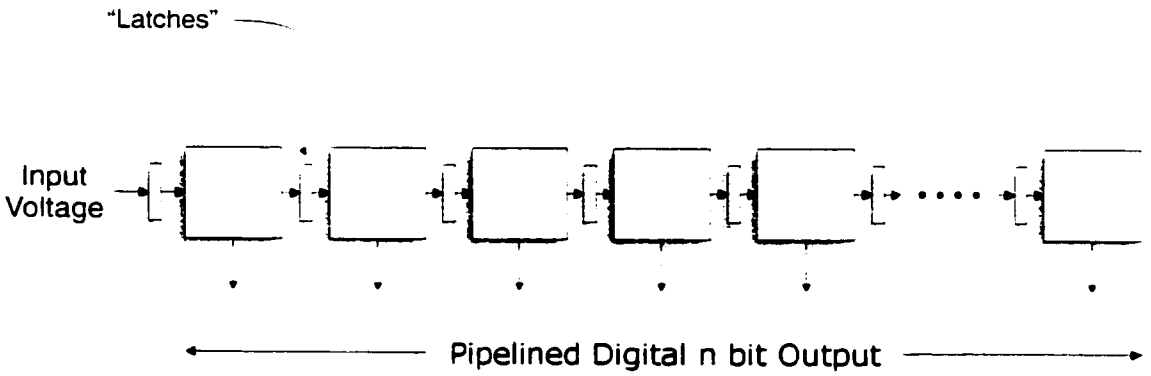


Fig. 18 Pipelined architecture.

The term "latch" here is used rather loosely. Since the signal passed from stage to stage is an analog voltage, the latch is actually implemented with a sample and hold circuit.

The complete pipelined ADC shown in Fig. 18 provides the same conversion time as the simple cyclical design. Where the pipelined converter takes advantage of the additional hardware is in throughput. For an n-bit conversion, the initial conversion completes after n clock cycles. After the first conversion, each additional A/D conversion takes 1 clock cycle regardless of the number of bits. This architecture provides much higher performance than the cyclical converter as a result.

Chapter 3 A 1.8-V, 20 MS/s Pipelined A/D Converter

This chapter presents the design of a high speed low voltage CMOS ADC using low gain opamps. The design targets low voltage CMOS technology to meet the increasing market-driven demand for system-on-a-chip integration. The circuit must operate at low voltages with relatively low power and high yield. The popular technique of cascading devices to increase amplifier gain is not a viable solution at this low voltage and high performance level. Therefore, the ADC performance is achieved through a high speed deep-sub-micron process, new circuit design techniques, and digital error correction.

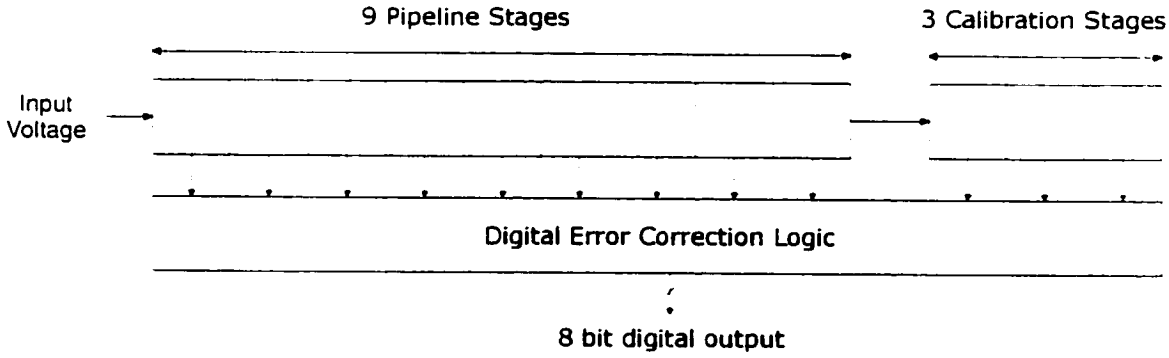


Fig. 19 High speed pipelined architecture.

3.1 Overview

The overall approach of the high performance ADC presented in this chapter builds on the standard pipelined ADC architecture. Fig. 19 shows a block diagram of the design. In addition to the 8 bit pipelined converter, there is an additional 3 bit calibration

pipeline. This calibration portion of the converter can be disregarded for the discussions in this chapter. The calibration stages are required in the design to implement a digital error correction approach [8] and corrects errors in the 8 bit converter output caused by comparator offset, capacitor mismatch, and finite amplifier gain. This error correction technique and the calibration hardware is discussed in detail in section 3.8 and Chapter 4.

3.2 Algorithm

To facilitate understanding of the pipelined converter approach used in this design, an algorithm to convert decimal numbers to binary is presented. This algorithm is virtually identical to the approach used in the pipelined converter.

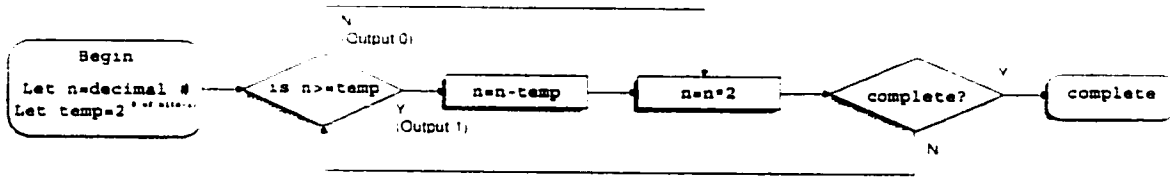


Fig. 20 Decimal to binary conversion algorithm.

Fig. 20 shows a decimal to binary conversion algorithm. For an n bit conversion, the algorithm takes a decimal number between 0 and 2^{n-1} as input. The algorithm loops a total of n times. Each loop converts 1 bit of the decimal number to binary. After n loops, the decimal to binary conversion is complete.

An example conversion is presented in Fig. 21. In this example, a decimal 4, n=4, is converted to a 4 bit binary number. To begin, the variable temp is set to $2^{4-1}=8$.

Once the initialization takes place, the conversion loop begins. If $n \geq temp$ the decimal output bit is set to 1 and the variable temp is subtracted from n, otherwise the output bit is set to 0 and n remains unmodified. Next, the variable n is multiplied by 2. Then the conversion repeats itself once for each bit. After 4 single bit conversions, the conversion from decimal to binary is complete.

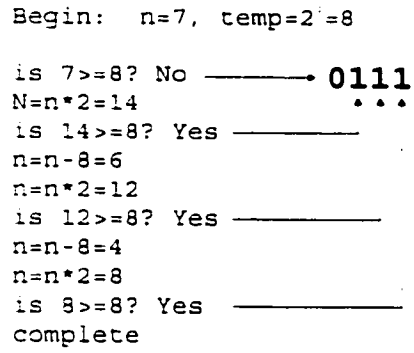


Fig. 21 Decimal to binary conversion example.

This algorithm is virtually identical to the approach used by the high speed converter design presented in this chapter. Now that the algorithm has been presented, it's implementation is discussed in the next section.

One final concept should be presented before discussing implementation. The concept of a transfer curve should be presented. The calculation of a single bit in the decimal to binary conversion algorithm can be thought of as the function of a single stage. The single stage takes a variable n as input, and outputs a new value of n, this output is often referred to as a residue, and a digital output bit. A transfer curve of a

single stage is a plot of the residue output plotted against all possible input values. The transfer curve of the algorithm presented above is shown in Fig. 22.

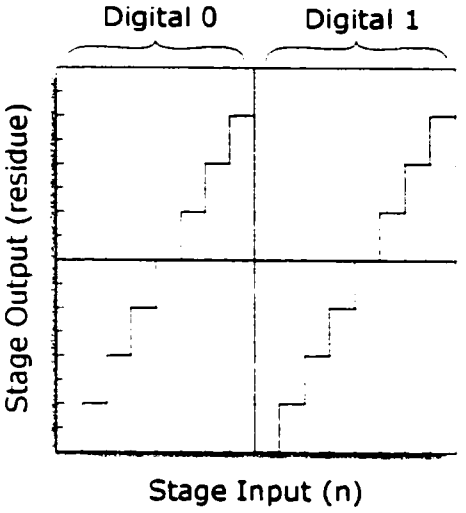


Fig. 22 Transfer curve for decimal to binary conversion.

3.3 Pipeline Stage

When referring to the algorithm presented in the previous section, the functional blocks present in the approach include a comparator, a subtractor, and a multiplier. The actual implementation has the same requirements with the addition of a sample and hold circuit. The S/H circuitry is required so that the subtraction and the comparison are performed on the same analog input voltage in time. Fig. 23 shows a unit cell that implements the algorithm presented in the previous section. Notice that the unit block outputs a residue voltage. This voltage is not fed back to the input like a cyclical converter or the decimal to binary conversion algorithm. This unit cell takes advantage

of the concept of 'loop unrolling' described in section 2.7 by passing the residue voltage onto an identical subsequent stage.

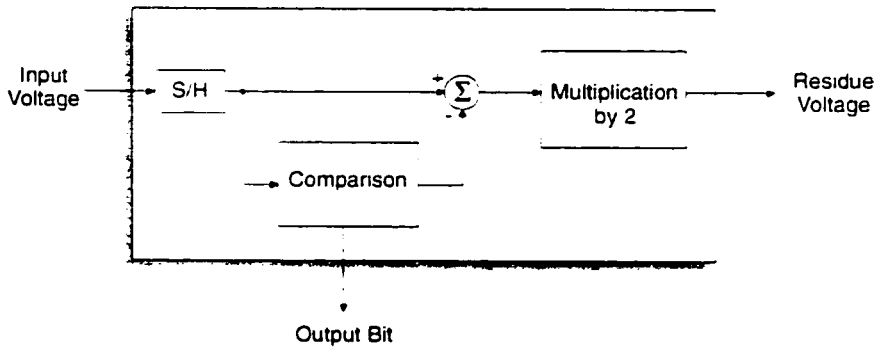


Fig. 23 Generalized unit cell.

The implementation of these functional blocks is described in the next section. The unit cell, or pipeline stage, can now be cascaded to implement a multi-bit pipelined converter.

3.4 Implementation

When implementing the unit cell functionality of the block diagram in Fig. 23 several simplifications can be made. One simplification is made by combining the functionality of the S/H and the subtractor circuitry. This is implemented carefully as to ensure that the S/H circuitry will hold the input analog voltage at the same instant in time as the comparator compares the input voltage.

In order to attain high speed and low voltage operation, the opamp circuitry is greatly reduced. This reduction in circuitry trades off gain for speed. The gain of the

opamp is reduced as a result of the limited complexity of the opamp design. The reduced single stage opamp circuit is capable of high speed operation because of reduced parasitic capaacitances and minimal amount of nodes in the signal path. The opamp circuitry is described and analyzed in detail in section 3.7.

Finally, since the comparator only has to resolve a single bit by determining whether the incoming analog voltage is positive or negative, the complexity of the comparator design is reduced while its speed is increased. The comparator circuitry is described in section 3.6.

The standard pipeline stage used in this design in conjunction with double sampling is shown in Fig. 24. Keep in mind that the actual implementation is fully-differential. The single-ended version is shown here for simplicity and is conceptually identical. Also, the circuit function is divided up into two clock cycles: ϕ_1 and ϕ_2. During the first clock phase, ϕ_1, the circuit samples the incoming analog voltage and stores it onto capacitors C_f and C_s. At the same instant that the incoming analog voltage is stored onto the capacitors, the comparator determines the output bit for that stage. During the second clock phase, ϕ_2, the subtraction and multiplication by 2 are performed and the residue voltage is generated.

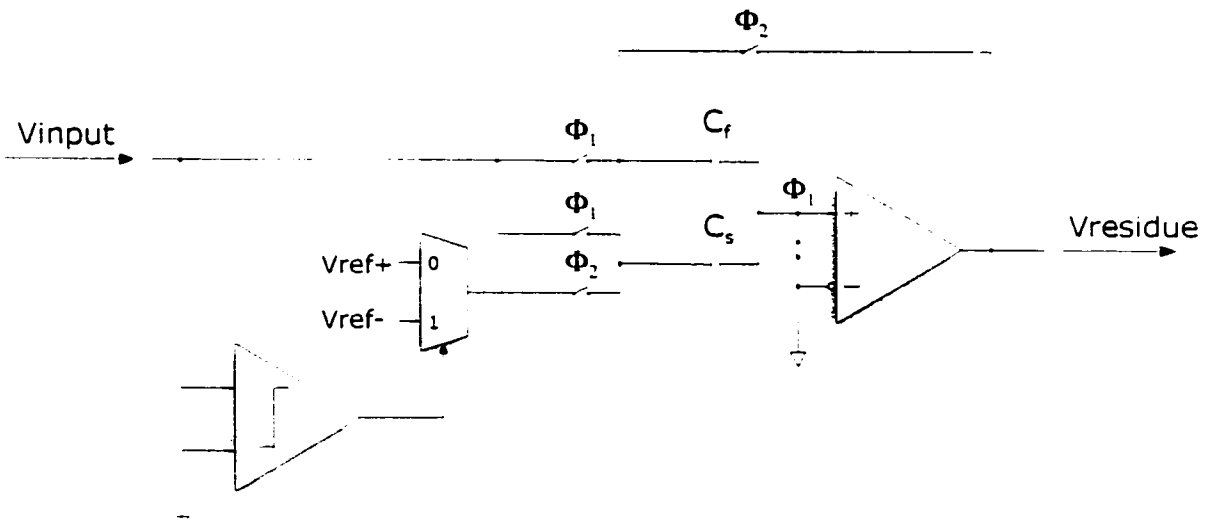


Fig. 24 Single ended pipeline stage.

Finally, the ideal transfer characteristic of this pipeline stage is shown in Fig. 25. Notice the resemblance to Fig. 22 presented earlier in this chapter for decimal to binary conversion. The actual transfer characteristic will have a number of deviations from the ideal case. These discrepancies and their solutions are discussed in section 3.8.

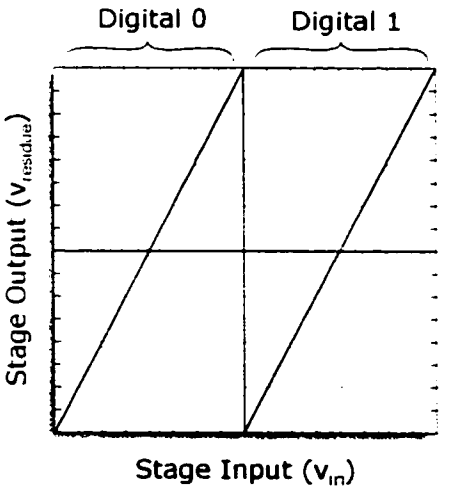


Fig. 25 Ideal pipeline stage transfer curve.

The following sections describe the functionality of the pipeline stage in detail. In addition, the errors introduced by the simplifications made in the architecture and circuit design and their solutions are discussed at the end of this chapter.

3.5 Switched capacitor network

As described above, the S/H circuitry and the subtractor circuitry are combined in the pipeline stage used in this design. The switched capacitor network combined with the opamp make up the S/H, the subtractor, and the multiplier. The operation of the pipeline stage is divided into two phases. During the first phase, ϕ_1, the opamp output is floating and the reference voltage is disconnected from the switch capacitor network. At the same time, the both inputs to the opamp are connected to ground and the analog input voltage is sampled across the capacitors C_f and C_s. This is shown in Fig. 26. At this point, the total charge stored across capacitors C_f and C_s, is defined as

$$Q = -v_{in}(C_f + C_s).$$

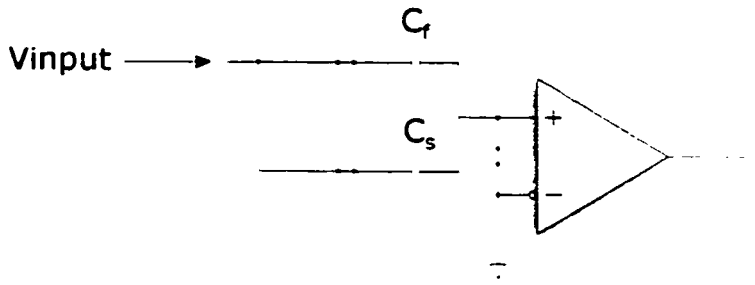


Fig. 26 Pipeline stage during ϕ_1.

At the completion of the first phase of operation, ϕ_1 falls and the switch holding the positive input to the opamp to ground is open circuited. Once this occurs, the positive input to the opamp is floating. Since the positive opamp input is floating, charge is conserved at that node.

During the second phase, $\phi 2$, the input voltage is disconnected from capacitors C_f and C_s. The opamp is fed back to the capacitor C_f and the selected reference voltage, Vref, is connected to the capactor C_s. The total charge at the positive input to the opamp must remain unchanged. During this phase of operation the total charge, Q, is defined as

$$Q = -v_{residuen} C_f - v_{ref} C_s .$$

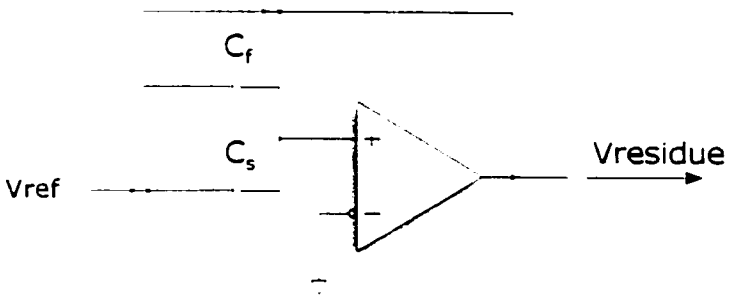


Fig. 27 Pipeline stage during ϕ_2.

It has been previously determined that charge is conserved during this phase of operation. Therefore, the total charge at the positive input to the opamp remains constant during both phases of operation. Setting the total charge equations equal to one another yields

$$v_{in}(C_f + C_s) = v_{residuen} C_f + v_{ref} C_s .$$

Finally, solving for the residue voltage yields

$$V_{residue} = \frac{V_{in}(C_f + C_s) - V_{ref} C_s}{C_f}.$$

This is the general expression for the residue voltage of a single pipeline stage. In this format, the desired functionality is not necessarily easy to identify. One final simplification should be made to simplify the expression. The capacitors C_f and C_s should be equal in value. Therefore,

$$C_f = C_s.$$

This simplifies the residue voltage equation to

$$V_{residue} = 2 \cdot V_{in} - V_{ref}.$$

The original algorithm called for a subtraction and a multiplication by 2. From the above equation it is apparent that the input voltage is in fact offset by a reference voltage and multiplied by 2. In addition, the reference voltage itself is chosen by the outcome of the comparison. The original functionality described by the algorithm in section 3.2 has been successfully implement in hardware.

3.6 Comparator

The comparator design takes advantage of the speed and simplicity of bi-stable cross-coupled inverter pairs. The comparator architecture is shown in Fig. 28. When ϕ_1

is high, both inverter pairs and the comparator outputs are initialized high. The actual comparison occurs when ϕ_1 falls. As ϕ_1 falls, the inverter pairs attempt to drive themselves to one of the two stable output possibilities. The currents used to pull the outputs to a stable condition flow through the two NMOS transistors connected to ground. Each NMOS transistor is turned on to a slightly different level depending on the input voltage. The transistor that is turned on the hardest will eventually pull the inverter on its side of the circuit to a digital 0 quicker than the other side of the circuit. The converter output is available as soon as one of the two comparator outputs reach a digital 0 voltage level.

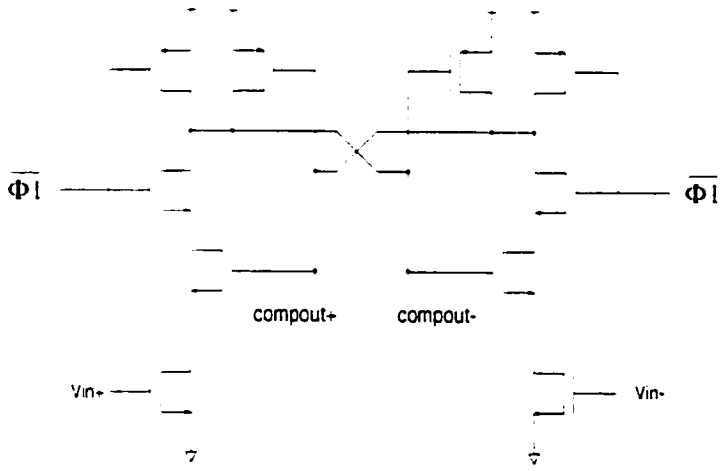


Fig. 28 Comparator design.

This comparator is a nice match for the low voltage, high speed architecture of the ADC. The circuit requires low 'head room' and functions nicely within the 1.8 volt

power rails. Since the circuit is ultimately a cross-coupled inverter pair, the speed of the circuit fits nicely with the high speed requirements of the ADC.

3.7 Operational Amplifier

The operational amplifier in this design is essentially the limiting factor in the speed and the performance of the overall A/D converter. As a result, the design of the opamp itself is critical to the success of the overall design. The main goal of the opamp design is to attain high speed and high gain at a low voltage of 1.8 V. This makes the design of the opamp exceeding difficult.

One approach to achieve high gain and high speed is to use cascoded devices. This approach is problematic in low-voltage designs. In addition, the output dynamic range would be severely limited as a result of the additional devices. Another approach is to use cascaded opamp tolopoliges (i.e., multi-stage), rather than conventional cascode topologies [6] [7]. This approach will consume more power, and die area for comparable performance to the opamp topology used in this design. A third approach is to reduce the number of transistors and complexity in the opamp design in an attempt minimize the total parasitic capacitances and the number of nodes in the signal path. This will result in a design that trades off high gain for increased bandwidth.

This design builds on the standard differential pair achieve the linearity, gain, and bandwidth requirements. This topology is shown in Fig. 29. The textbook folded

cascode design takes advantage of the additional gain of a cascaded structure while not requiring the as much headroom as the typical cascaded design.

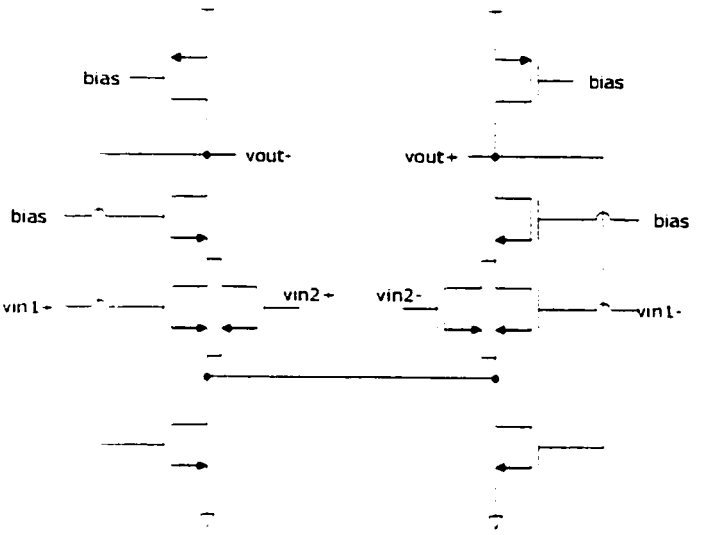


Fig. 29 Operation amplifier design.

In an attempt to increase the performance, a continuous time feedback approach is used rather than conventional dynamic common mode feedback. In this circuit, there are three main deviations from the simple differential pair amplifier with active loads: integration of tail current source and continuous time common-mode feedback, quad input devices, and cascoded driver devices.

The feedback topology replaces the tail current source of the amplifier and capitalizes on the low voltage nature of the circuit. The two tail current sources connect directly to the opamp output nodes. While sinking a constant amount of current as the opamp swings differentially, any common mode voltage that is produced on the output node is rejected. This topology provides a constant tail current for differential input and

rejects any common-mode signals with only inducing a slight impact on the circuit bandwidth and gain.

Each input device is mirrored with an identical device. These two additional input devices are necessary for the double sampling approach used in this design. The reasoning behind this architecture is presented in detail later in this section.

In low-voltage circuits cascoding is generally not something that should be used to boost gain. In this design, the cascoding transistors are used for isolation rather than for increasing the opamp gain. The cascoding transistors are necessary to isolate the opamp output nodes from the driving transistors for the double sampling approach used in this design. This is described later in this section.

Although the simplified opamp architecture succeeds in improving the total bandwidth to meet the specifications for high-speed operation, the architecture also introduces a number of drawbacks that limit the performance of the ADC. These limiting factors and their solutions are presented in the following section.

3.8 Non-idealities

The opamp architecture described in the previous section introduces two main performance-limiting factors to achieve high bandwidth. The two limiting factors are low gain and output nonlinearity. This section describes how this design manages these problematic opamp characteristics.

From section 3.5 the residue voltage generated was

$$v_{residue} = 2 \cdot v_{in} - v_{ref} .$$

When non-ideal effects are included in the analysis, the residue voltage equation becomes

$$v_{residue} = 2 \cdot \left(1 + \frac{\Delta C}{C} - \frac{1}{Af}\right) \cdot \left\{v_{in} - \frac{v_{ref}}{2}\left(1 + \frac{\Delta C}{C}\right)\right\},$$

where $C = C_s = C_f$. A is the finite opamp gain, and f is the feedback factor of the closed loop system. By analyzing the residue equation, several non-ideal effects are identified. Capacitor mismatch between capacitors C_f and C_s can reduce or increase the stage gain from the optimal gain per stage of 2. A design with a stage gain per stage over 2 will cause the ADC to function incorrectly so additional caution should be used when designing the switched capacitor portion of the circuit. This potential problem is alleviated in this design because the low amplifier gain and feedback factor combine to reduce the gain per stage below the optimal value of 2.

The typical feedback factor of the opamp and switched capacitor circuit used in this topology is less than $1/5^{th}$. This combined with the low amplifier gain from this topology of approximately 35dB yields a stage gain of around 1.60. This stage gain of 1.6 is too low for acceptable circuit performance.

In addition to the reductions in stage gain due to capacitor mismatch, feedback factor, and finite opamp gain there is an additional circuit design issue associated with the opamp. The opamp gain varies as a function of the input voltage. Solutions to this non-ideality and the other circuit performance issues are described in the following subsections.

3.8.1 LOW OPERATIONAL AMPLIFIER GAIN

The finite opamp gain combined with the feedback factor of the circuit degrades the stage gain below the optimal value of 2. The obvious solution is to increase the opamp gain. This approach is exceedingly difficult while maintaining high speed and ultra low-voltage operation.

The solution to this problem is to design around the low opamp gain and use a digital error correction technique [8]. This error correction technique compensates for low stage gain, capacitor mismatch, and comparator offsets. Essentially, by adding additional stages for calibration, the amount of error introduced by finite opamp gain, capacitor mismatch, and comparator offsets is measured and corrected. The digital error correction technique is presented in detail in section 4.1.

The digital error correction technique described in section 4.1 was not originally designed to compensate for a low gain amplifier. Irregardless, the correction technique successfully removes the missing digital output codes that result from the low amplifier gain.

3.8.2 CAPACITOR MISMATCH

In addition to reduced stage gain as a result of low amplifier gain and feedback factor, capacitor mismatch also varies the stage gain. There are several approaches to minimizing mismatch in an IC design [31-35]. Fortunately, the digital error correction algorithm employed to correct variations in opamp gain also corrects for capacitor mismatch at the same time.

The finite opamp gain error shows up as a gain error in the stage output. The capacitor mismatch errors also shows as a gain error in the stage output. Since the digital error correction scheme used corrects for gain errors, capacitor mismatch gain error is correct as well as finite opamp gain error.

3.8.3 OPERATIONAL AMPLIFIER NONLINEARITY

As described in earlier in this section, the opamp gain varies as a function of the input voltage. Therefore, a more realistic residue equation is

$$v_{residue} = 2 \cdot \left(1 + \frac{\Delta C}{C} - \frac{1}{A(v_{ref}) f} \right) \cdot \left\{ v_{in} - \frac{v_{ref}}{2} \left(1 + \frac{\Delta C}{C} \right) \right\}.$$

In this case, the gain of the amplifier varies as a function of input voltage. In traditional pipelined ADCs the amplifier uses techniques to generate high gain output. As a result, the reduction in stage gain due to the amplifier and feedback factor drop out of the equation. As determined previously, circuit topologies yielding high gain are not

possible in this design when both high-speed and ultra low-voltage operation are required.

Improvements to the error correction technique described in section 4.1 have been considered. These improvements would attempt to characterize the nonlinearity of the stage output by sampling the output at various reference levels. Once the nonlinearity has been quantized, the pipelined outputs can be corrected for nonlinearity in addition to gain, capacitor mismatch, and comparator offsets.

Since the resolution of the pipelined converter design in in excess of 8 bits, this improved error correction technique will be unnecessary. The linearity required by the circuit is $\frac{1}{2^8} = 0.4\%$. Therefore, the total harmonic distortion (THD) of the opamp must not exceed 0.004 (-48 dB).

One circuit included in this design to achieve additional linearity is to reduce the internal signal magnitude. The overall stage gain of the first stage is approximately $1/4^{th}$ rather than the optimal value of 2. by lowering the gain of the first stage, the maximum internal signal swing is reduced by 4. This improves the overall linearity of the circuit because reducing the signal swing reduces the signal dependent nonlinearity induced by the circuit.

This level of performance can be achieved directly without modifying the circuit topology or improving the error correction technique. Although, this approach of

designing the opamp to function with 8 bit accuracy would be problematic at higher resolutions.

3.8.4 NOISE CONCERNS

One concern of all ADCs is the effect of noise on the system. In the case of pipelined converters, the dominant noise source is capacitor noise inherent in the sampling process. More commonly referred to as $\frac{kT}{C}$ noise, arises from the basic sampling process of the switched-capacitor circuitry.

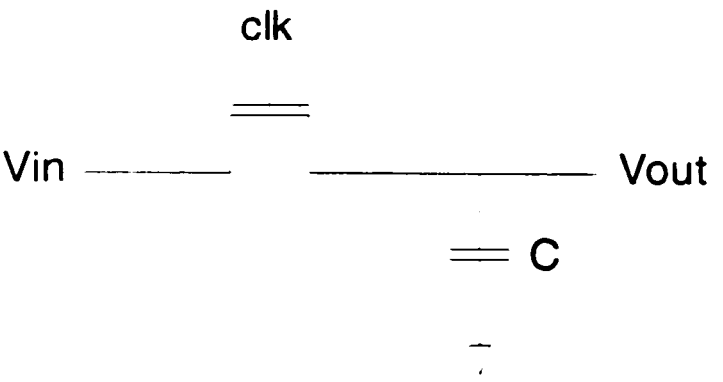


Fig. 30 Basic Sampling Circuity.

Fig. 30 shows the basic signal sampling circuit. Ideally, Vin is sampled onto the capacitor, C, when the MOSFET is in on. Once the MOSFET is turned off, a sampled version of Vin remains on the capacitor. Regarding noise concerns, the MOSFET has a channel resistance which results in thermal noise.

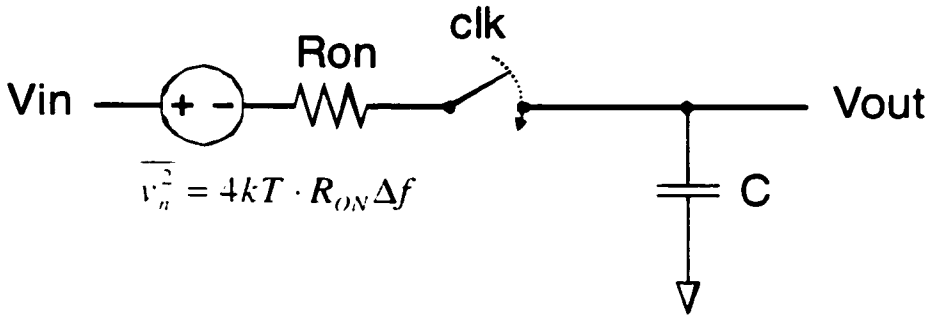


Fig. 31 Sampling Equivalent Circuit.

Fig. 31 shows an equivalent sampling circuit that includes provisions for calculating the resulting noise voltage from $\frac{kT}{C}$ noise. The over a large number of samples of the input signal, the sampled value will be a gaussian distribution around the input signal with a standard deviation of $\sqrt{\frac{kT}{C}}$ V:

$$\overline{V_{out}^2} = \frac{kT}{C}$$

Now, with reference to the current pipelined converter, to maintain signal integrity the signal must remain above the noise floor to achieve the specified resolution of 8 bits. Since the number of stages required to achieve 8 bits of resolution is 9 stages, we to achieve a signal-to-noise ratio (SNR) of greater than 9 bits. The peak differential signal is 0.25 V (0.177 Vrms) and the equivalent sampling capacitor is 0.2 pF. Using the information above, the noise voltage voltage is

$$\overline{V_{out}} = 144 \, \mu V / \sqrt{Hz} .$$

This yields a SNR of 61 db. This SNR value corresponds to 9.97 effective bits. This is beyond the requirements for the design and will not prohibit the data converter from achieving 8 bit resolution.

3.9 Low-Gain Double Sampling Approach

One key attribute of the standard pipelined approach presented in previous sections is that the opamp is only active during the second phase of the clock cycle. During the first phase of the clock cycle, the opamp output is held at the common-mode output voltage. In an attempt to exploit the unused opamp, a double sampling technique is implemented.

Fig. 32 shows the single stage functionality of a standard pipelined converter. The input voltage is sampled and held on capacitors during Phi1. Once sampled, the input voltage is evaluated during Phi2 to generate the residue. The amplifier is unused during Phi1.

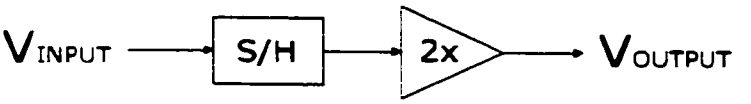


Fig. 32 Single Sample Approach.

This functionality is shown in Fig. 33. Notice the residue output only resolves an output voltage during Phi2. A superior design would attempt to take advantage of the unused opamp during Phi1.

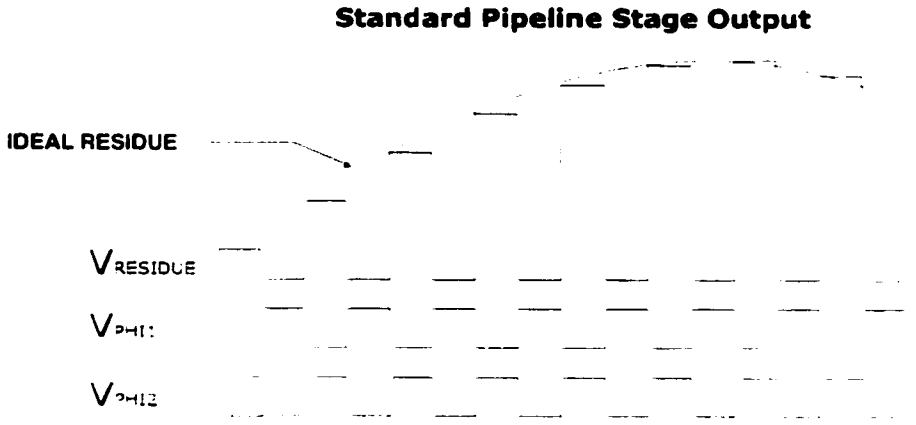


Fig. 33 Single Sample Output.

One implementation for the double sampling approach is shown in Fig. 34 [12]. This approach attempts to utilize the opamp during both phases of the clock cycle by doubling and interleaving the sampling circuitry. By interleaving two sets of sample and hold circuitry, the opamp will have a sampled input voltage to evaluate the residue from on every phase of the clock cycle. This approach effectively doubles the sampling rate of the ADC—quite a performance enhancement.

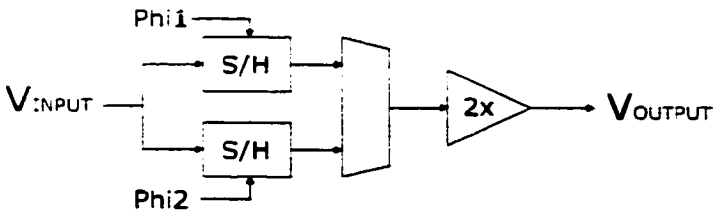


Fig. 34 Double Sampling Approach.

This functionality is shown in Fig. 33. Now the residue is calculated on both phases of the clock. Although the design itself is running at the same speed as the single sample approach, the number of output bits generated in a given amount of time is doubled.

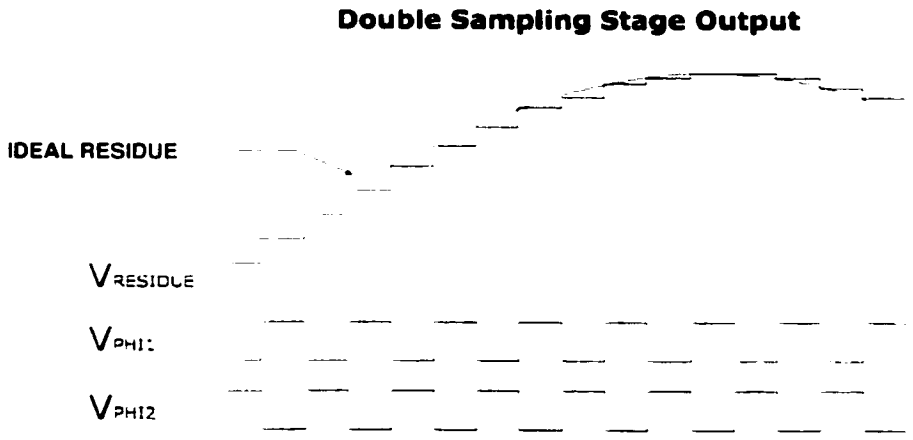


Fig. 35 Double Sampling Output.

Fig. 36 shows the implementation of the double sampling circuit [12]. The circuit essentially switches in two identical sets of switched capacitor circuitry to yield double the residue and bit outputs in the same amount of time. Unfortunately, this implementation will not work with the design presented in this paper. This circuit relies on the opamp having high gain. Since the amplifier in this paper only has around 35 dB of gain, the circuit simply doesn't work. The reason behind this circuit failure gets back to the fundamentals of opamps and feedback theory. Although a more detailed

description of why this circuit fails will be presented at the completion of this research work, a summary is provided here.

During any given clock phase, one set of switched capacitor circuitry is sampling the input while the other set of switched capacitor circuitry being used to evaluate a residue voltage. When an opamp is connected in feedback, it drives it's two input nodes to an equal voltage. While an ideal opamp achieves this, actual opamp implementations can only drive the input nodes of the opamp near an equal voltage. Their difference has a 1/a relationship, where a is the open loop gain of the opamp. When using a high gain opamp in the circuit, the input nodes are driven close enough in voltage to not adversely affect the accuracy of the residue output. When using an opamp with low gain, this difference in input voltages becomes critical. The voltage difference between the two input nodes is stored as a charge on all the parasitics hooked to those nodes and injected onto the feedback capacitor during the next evaluation. This charge is seen as an error in the resolved value of the residue voltage.

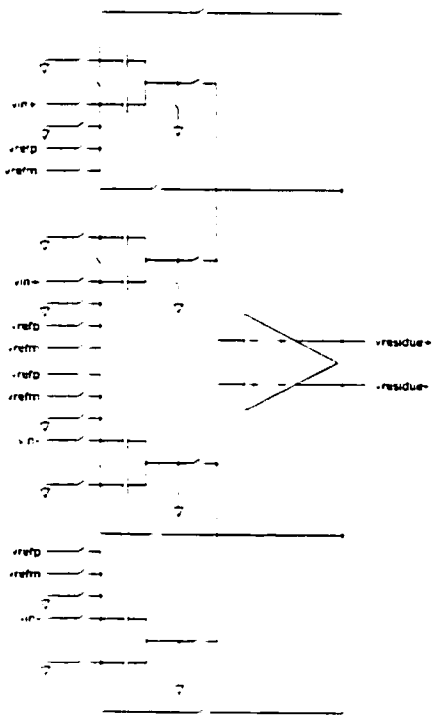


Fig. 36 Double Sampling Implementation.

Fig. 36 shows a low-gain implementation of the double sampling approach. This approach separates the two interleaved channels all the way into the opamp. This allows the circuit to short the two input of the opamp to exactly the same voltage. In addition to shorting out the parasitics, it also ensures that the input nodes are sitting exactly at the common mode input point for every sample. Two main modifications were made to the differential pair amplifier. The number of input transistors is doubled. This allows for 4 inputs instead of 2. And secondly, a cascoded transistor is introduced. This transistor is not used for additional gain, but is used to isolate the output from the input. Without this transistor there is a parasitic capacitance that stores voltage dependent charge and is connected in parallel with the feedback capacitor during evaluation. This capacitance is

isolated by the use of the cascoding transistor. This was not a problem in the single sampled circuit because the output was always at the common-mode output voltage during the sampling phase.

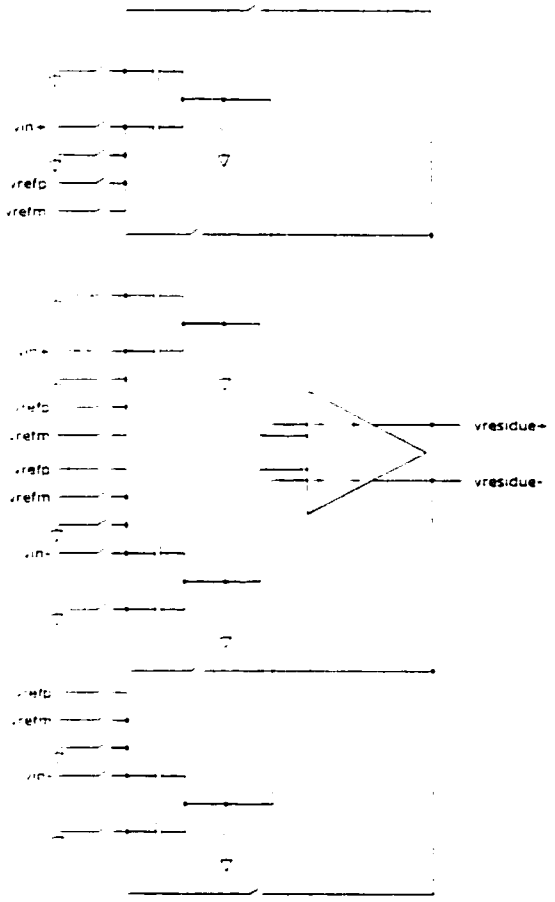


Fig. 37 Low-Voltage Double Sampling Implementation.

The low-voltage double sampled approach allows for double sampled operation using low-gain opamps. As a result, this converters performance is increased from 50MS/s to 100MS/s.

Chapter 4 Digital Error Correction and Calibration Techniques

In order to achieve high-speed and high-resolution in a pipelined converter, some trade-off's must be made. The 1-bit-per-stage architecture is a nice implementation for high-speed because of the simplicity in each stage. The simple architecture lends itself to high-speed operation. The primary performance limitations are capacitor mismatch, charge injection, finite operational amplifier gain, and comparator offsets [8]. The digital error correction approaches presented in this chapter attempt to alleviate the errors described above.

4.1 One-Bit Correction Approach

The Karancolis digital error correction approach uses a self-calibration technique that is based on a single pipeline stage with a gain less than the optimal value of 2. One key advantage to this technique is that the calibration information is measured under the same circumstances as normal conversion operation. This ensures accurate calibration and high performance digital error correction.

Fig. 38 shows the principal errors introduced into the residue output as a result of non-ideal effects. An ideal transfer curve is shown in Fig. 25. Residue voltages beyond the refence voltage, shown as leaving the box in Fig. 38, cause subsequent stages of the pipeline to saturate and miss decision levels. Attenuated residue voltages that do not extend to the reference voltage, as in the low amplifier gain example, result in missing

output codes. Both missed decision levels, and mission output codes are unacceptable for high performance ADCs.

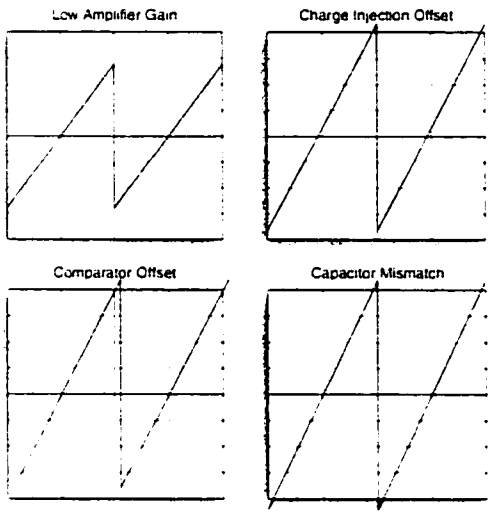


Fig. 38 Principle errors in residue output.

This digital error correction approach begins by reducing the gain of a pipeline stage below 2.0. Ensuring that the pipeline does not saturate for any analog input voltage is important. By forcing the stage gain below 2, the attenuated residue output voltage will not exceed the reference voltage. An exaggerated gain reduction is shown in

Fig. 39 for simplicity.

S2

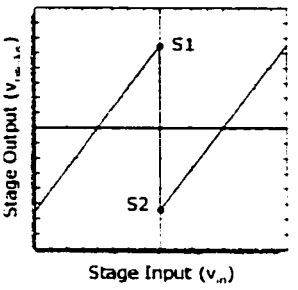


Fig. 39 Calibration coefficient calculation.

To calculate the calibration information, the two quantities S1 and S2 are measured. The S1 and S2 quantities correspond to the quantized representation of $v_{residue}$, when $v_{in} = 0$ with the digital output, D, equal to 0 and 1, respectively. The digital self-calibration algorithm is:

$$Y = \begin{cases} X & ,if\ D = 0 \\ X + S1 - S2 & ,if\ D = 1 \end{cases}$$

where D is the bit decision, X is the raw digital code, and Y is the transformed, or digitally corrected, output code. This transformation ensures that the output code Y with $v_{in} = 0$ is the same for D=0 and D=1, eliminating the missing codes associated with attenuated residue outputs.

4.2 Redundant Signed Digital (RSD) Approach

The RSD approach is another error correction technique capable of correcting for capacitor mismatch and stage gain error [9]. The RSD approach uses a 1.5 bit

comparator. The resulting stage transfer characteristics are shown in Fig. 40. Due to the nature of the transfer curve, the circuit is robust towards comparator offset. When using a single bit transfer curve, the pipeline stage reaches +1 V or −1 V as the pipeline stage input approaches 0 V. In the event that comparator offset is a non-negligible effect on the circuit, the pipeline stage output could exceed +1 V or −1 V. The resulting residue voltage would force the ADC into an incorrect operating region. This RSD approach seeks to avoid this problem by modifying the transfer curve.

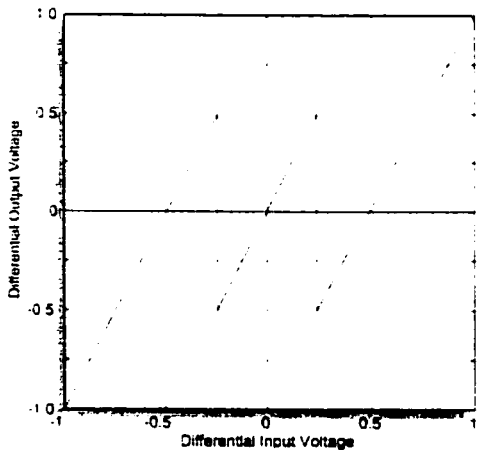


Fig. 40 RSD pipeline stage transfer characteristic.

Although the RSD approach is an excellent error correction technique, it was not used in the design of the high speed pipelined ADC presented in Chapter 3. The digital error correction utilizing a single bit comparator was more suited for the design.

Chapter 5 High Speed Behavioral Model of Pipelined ADC

Using computer simulation to predict and optimize the performance is a valuable approach to data converter design [29][30].An approach for design, simulation, and characterization of high performance pipelined analog to digital converter (ADC) architectures has been developed. This approach is outlined graphically in Fig. 41.

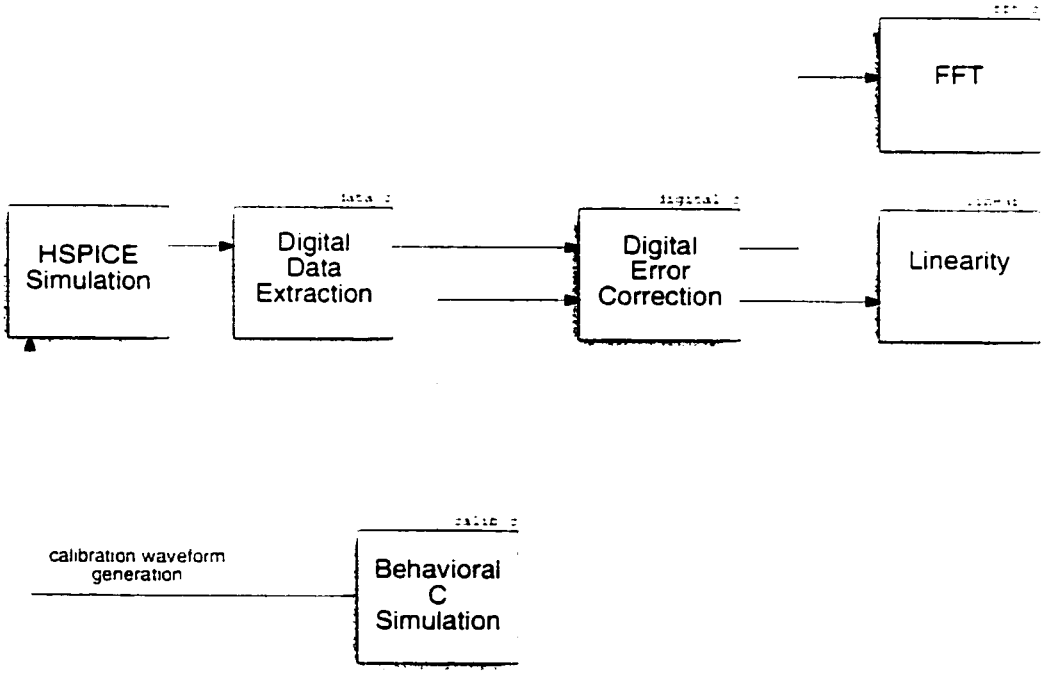


Fig. 41 Design Roadmap.

5.1 Overview

The approach allows the designer to begin with a behavioral C simulation prior to committing to time consuming SPICE simulations. Regardless of the source of the binary ADC output, the software will characterize and analyze the output data stream.

When using SPICE to simulate the pipelined ADC, the first step is to extract the digital output from the digital output waveforms: bit0, bit1, etc. The software created to extract the data is named data.c. This software is described in section 5.3.

A second approach to simulating the pipelined ADC is to use the developed behavioral model. The behavioral model executes in a fraction of the time required to run an entire SPICE simulation. The behavior model allows you to simulate any number of pipelined ADC converter topologies as well as introduce non-linearities into the simulation to make the simulation more 'true to life'. While at the same time, the behavior model only takes a fraction of a second to execute.

Once the SPICE simulation completed or the behavioral model is executed, the binary output data must be processed and converted to decimal. The program designed to simulate this process is digital.c. This software is discussed in section 5.4. This software will decode the binary digital data and perform an error correction algorithm [8] on the data. The output from this software is digitally corrected decimal data. The decimal data is corrected for gain and comparator offset as a result of the error correction algorithm.

Now that the output data has been processed and corrected, there is a need to characterize the performance and output characteristics of the ADC output data. These results will give a feel for the quality and performance of the ADC itself.

The fft.c software was designed to take the corrected digital output data and perform a Fast Fourier Transform (FFT) on the data. This software is discussed in section 5.5. From this FFT, the software will calculate the Signal to Noise Ratio (SNR), the Signal to Noise and Distortion Ratio (SNDR), the Total Harmonic Distortion (THD), and the Spurious Free Dynamic Range (SFDR) of the ADC output data. This information is very useful when trying to characterize the performance of the ADC converter.

Another important performance measure of ADC's is the Integral Non-Linearity (INL) and the Differential Non-Linearity (DNL) of the ADC output data. These two plots are calculated with linear.c. This software is discussed in detail in section 5.6. This program will divide the ADC output into specific output bins. Once these bins are filled will all the available output data, the INL and DNL plots are calculated.

This approach will facilitate the design of high performance ADCs. This approach and the software and performs the functionality is described in subsequent sections.

5.2 Pipelined A/D Behavioral Model and Simulation

A behavioral simulation model for high performance pipelined ADCs is developed using the C programming language. This model is designed to generate

accurate pipelined converter outputs without investing the time required to perform a SPICE simulation. This model attempts to model non-idealities associated with finite OpAmp gain.

This program progresses through three distinct output regions of operation during a simulation run. Fig. 42 shows this progression. The hold-off period is simulated to allow the circuit to reach steady-state. This is not needed in the behavioral model because it is not a SPICE simulation and none of the equations require reaching steady-state, it is included because the behavioral model also generates the stimulus file for the SPICE simulation. The stimulus for the SPICE simulation must allow time for the circuit to reach steady-state before calibration of the converter begins to ensure accurate results.

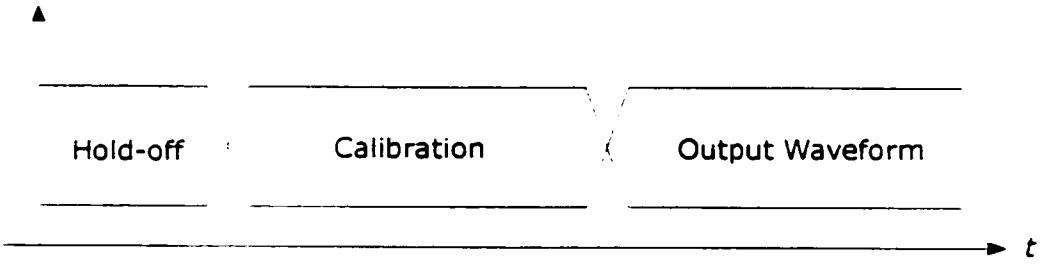


Fig. 42 Behavioral Model Output.

The next stage in the simulation is the calibration stage. The behavioral model simulates the process of digital calibration as described in previous chapters. No calibration parameters are calculated at this point, the calibration coefficients are calculated on the output waveform data once the entire simulation is completed. This is an important point, because when an actual IC is fabricated, the data will be gathered

from the output of the IC. Only after an entire set of output data is captured will the error correction approach be applied to the output data.

Once the calibration stage completes, the simulation proceeds to output an output waveform. This waveform is chosen by the user depending on what processing and calculations are to be performed on the output data. Currently, the most useful output waveform is a sinusoid. From this sinusoidal output many performance metrics are calculated: SNR, SNDR, THD, SFDR, INL, and DNL. These calculations are described later.

```
Command format: calib [outfile] <options>

            calibration options:
               -h      [hold off cycles before calibration]
               -s      [pipestages] [calibstages]
               -r1     [pipe Cf]   [pipe Cs]   [pipe Ca]
               -r2     [calib Cf] [calib Cs] [calib Ca]
               -x      skip calibration sequence
               -i      [calibration samples per stage]

            pipeline output options:
               -ramp   ramp input after calibration
               -sine   sine wave input after calibration
               -tri    triangle wave input after calibration
               -f      [ratio of input freq to sample freq]
               -gamma  [non-linearity factor]
               -o      [number of output cycles after calib]

            output options:
               -t      [filename] output transfer function
               -tm     [Matlab filename] output transfer function
               -plot   [filename] transfer function plot code

            calibration options:
               -stim   [filename] HSPICE calibration stimulus
               -v      [digital supply voltage]
               -clk    [clock rate] in MHz
```

Fig. 43 Pipelined A/D converter behavior model command format.

The command format for the behavioral model is shown in Fig. 43. The model accommodates many single bit per stage pipelined A/D converter architectures. In addition to describing the architecture of the pipelined converter, the input and output formats, and calibration information, the user can also provide the necessary input to generate a SPICE stimulus file. **This stimulus file when used with a circuit description matching the behavioral model parameters will perform a full SPICE simulation identical to the behavioral model simulation.**

This ability to run a behavioral model or an identical SPICE simulation or both is very power for design and development. With this tool, the user can run many behavioral simulations each taking only a fraction of a second. Once the desired circuit performance is attained, the user can commit to a time consuming SPICE simulation to verify results.

Another important characteristic of the design roadmap shown in Fig. 41 is the ability for the digital error correction, Discrete Fourier Transform, and the linearity characterization to perform identical calculations on either the SPICE output or the behavioral model output. This ensures accurate comparison between the behavior model and the SPICE simulation--an 'apples to apples' comparison if you will.

As explained above, this behavioral model generates a hold-off period followed by a calibration waveform and concludes with an output waveform. Once the simulation is complete, the output data is ready to be processed by the digital error correction. After

digital error correction, the output data is characterized and the performance of the ADC is calculated. This process is described in subsequent sections.

5.3 SPICE Digital Data Extraction

A SPICE simulation generates waveforms. Although the input to a pipelined ADC is an analog waveform, the output is digital. Therefore, the SPICE output waveforms must be converted to digital data. The digital data extraction software shown on the design roadmap in Fig. 41 will directly convert the bit output waveforms from the pipelined ADC converter to digital 1's and 0's. This is an essential step in realizing and characterizing the output of a pipelined ADC converter.

Careful design of this software has made it possible to extract the digital data from the SPICE simulation prior to the conclusion of the simulation. This is an invaluable feature of the software. Unfortunately, the SPICE simulation of an 8-bit pipelined ADC converter with calibration can take as long as 1 month! It would be disappointing to run an entire simulation only to find that the results are worthless. With the ability to extract digital data without terminating the simulation, preliminary results are established in as early as 12 hours. If the SPICE simulation is not yielding acceptable results, the simulation can be terminated and corrected. On the other hand, if the results look promising, the simulation is allowed to continue to generate more concrete results.

Once the software has been designed to interface to the SPICE simulation output directly, the remainder of the digital data extraction is relatively simple. The conversion

software scans through the SPICE output starting at time 0. It searches for falling clock edges. On the falling clock edges, the bits are sampled as shown in Fig. 44. The process is slightly more complicated than it is shown in the diagram because the pipelined converter uses two phase clocking. The two phase clocking makes it possible to use time more efficiently and as a result the digital data extraction process is slightly more difficult. The basic idea is to sample on the falling clock edge.

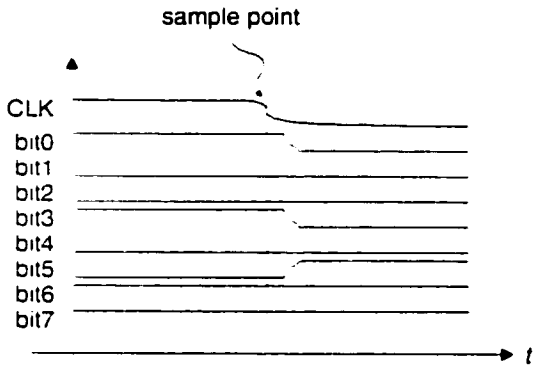


Fig. 44 SPICE waveform sampling.

Once the voltages are sampled they are compared against a simple conversion chart shown in Fig. 45. they are sent to an output file and are ready for processing and digital error correction. If a single undefined (X) is found in the output beyond the hold-off period, the comparator output is not making the timing specification and as a result, the simulation is halted and discarded.

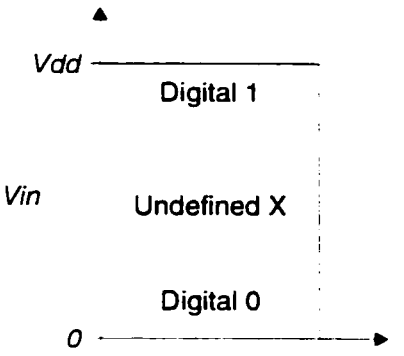


Fig. 45 Output bit conversion to digital.

5.4 Digital Error Correction

Digital Error Correction is a valuable way to correct and improve the quality of a data converter's output [15-19]. Once pipelined digital data is generated by the behavioral model or the SPICE simulation the data must be processed and the digital error correction algorithm must be applied to the data. The following is a description of how the digital error correction algorithm is applied to the simulation data.

First, the pipelined simulation data is separated and reconstructed as non-pipelined data. Once this is complete, the software skips over the hold-off period and begins the calibration algorithm described in section 4.1. At the completion of the calibration algorithm, all of the calibration coefficients have been calculated. Finally, the remainder of the output data is corrected using the correction coefficients.

The corrected output data is in decimal format and is ready for whatever tests the user wishes to perform on the output data. The following sections describe how to calculate SNR, SNDR, THD, SFDR, INL, and DNL information.

5.5 Discrete Fourier Transform

Once the pipelined digital data is processed and the error correction has been performed, the data can be tested for performance metrics. This program performs a Discrete Fourier Transform on the input data. This program was designed to take the corrected digital output data and perform a Fast Fourier Transform (FFT) on the data. From this FFT, the software will calculate the Signal to Noise Ratio (SNR), the Signal to Noise and Distortion Ratio (SNDR), the Total Harmonic Distortion (THD), and the Spurious Free Dynamic Range (SFDR) of the ADC output data. This information is very useful when characterizing the performance of the ADC converter.

A 1.8 V, 100 MS/s, 8-bit pipelined A/D converter simulated using SPICE for approximately 1 week. In this extended amount of time, SPICE simulated a total of 2500 clock cycles. The resulting output data was corrected using the digital error correction software described in the previous section. Performing an FFT on the corrected data using the FFT software produced the FFT plot in Fig. 46.

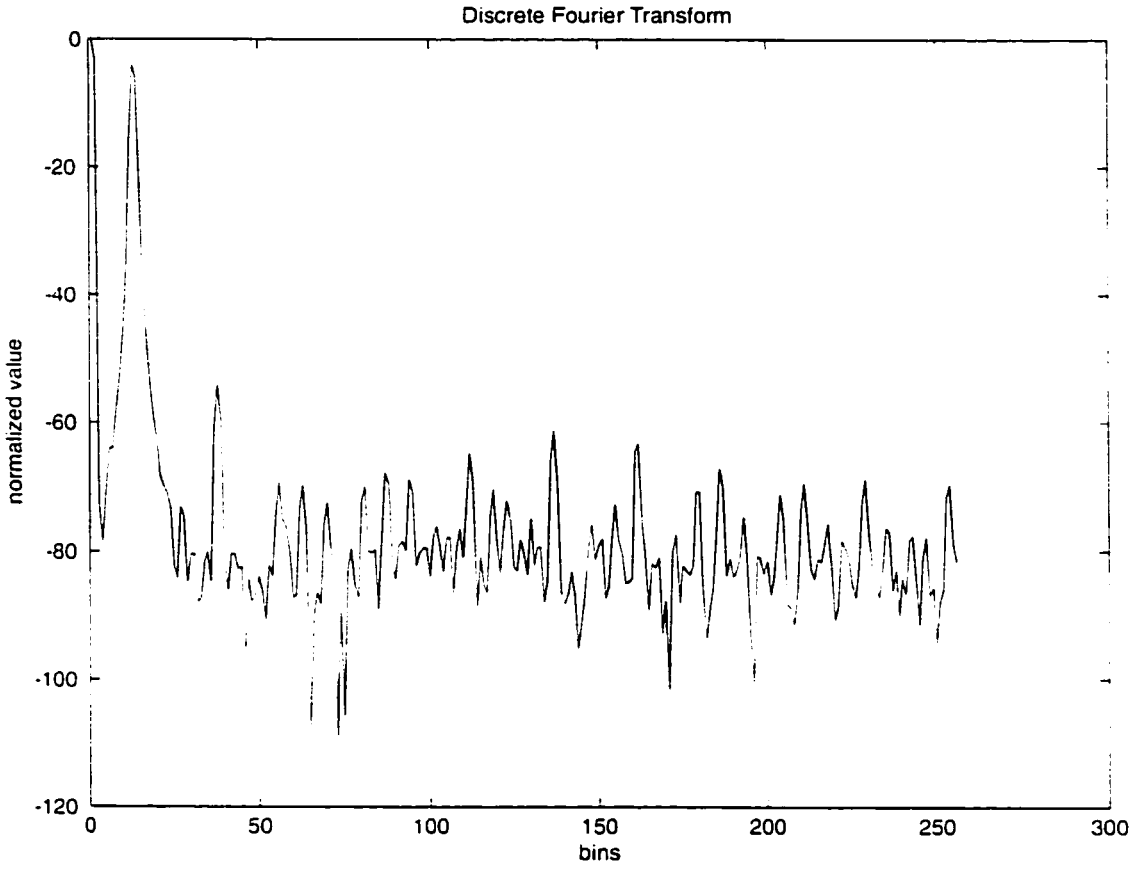


Fig. 46 Example FFT output plot.

Once the FFT is calculated, by performing a series of calculations on the FFT data the SNR, SNDR, THD, and SFDR can be calculated. The results from the 4-bit pipelined converter simulation are shown in Fig. 47. In addition to the frequency component information, signal power, noise power, and distortion power, the SNR, SNDR, THD, and SFDR are displayed.

```
FFT Results:

  SNR           =  49.58 dB
  SNDR          =  44.54 dB
  THD           = -51.67 dB
  SFDR          =  45.46 dB

  Signal        =  655187.35
  Noise         =  2175.03
  Distortion    =  1709.47
  2nd Harmonic  =  83834.87 @ bin 24
  3rd Harmonic  =  474011.12 @ bin 36
  4th Harmonic  =  5034.69 @ bin 48
  5th Harmonic  =  8753.00 @ bin 60
```

Fig. 47 Example FFT output results.

5.6 Linearity Characterization

Another important performance measure of ADC's is the Integral Non-Linearity (INL) and the Differential Non-Linearity (DNL) of the ADC output data. A code density test is used to calculate the INL and DNL from a given set of data. This approach is described in this section.

For an ideal n-bit converter, all 2^n output codes are evenly spaced across the dynamic input range of the converter. For a realistic converter, errors resulting from a number of sources make the actual output of an ADC to vary from the ideal case. These output sequences are depicted in Fig. 48.

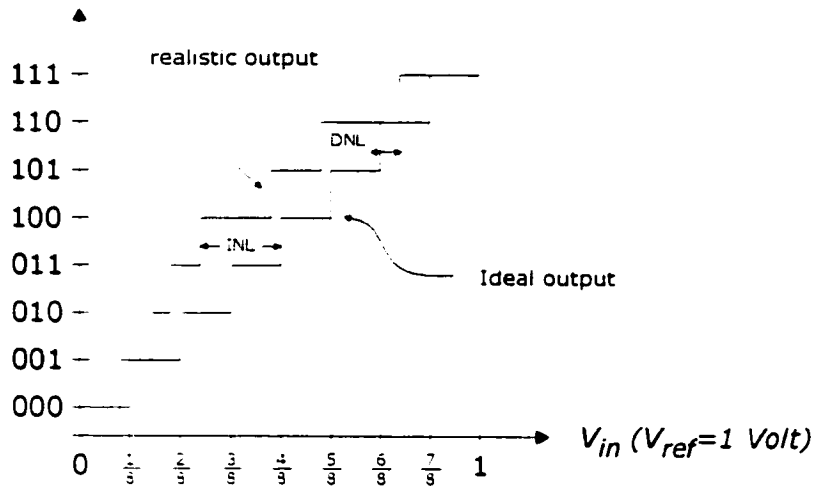


Fig. 48 Depiction of INL and DNL.

When focusing on the ideal case, the width of any ideal code is $\frac{1}{2^n}$ (when $(V_{ref} = 1\ v)$. The deviation of a specific code from $\frac{1}{2^n}$ is the DNL of that specific code. The INL of a specific code is its deviation from the ideal location of that code. The INL is described by

$$INL_n = \sum_{i=0}^{n} DNL_i$$

where n is output code in question. Although these measurements can be made at DC, modification must be made to the approach to calculate the INL and DNL of the ADC as the input frequency increases.

The code density approach is capable of calculating the INL and DNL of an ADC at any input frequency up to $\frac{f_s}{2}$, where f_s is the sampling frequency [11]. The code

density approach counts the number of times each code appears in the output. An average of 8-16 occurrences are needed per code to minimize statistical variations. The next observation that must be made is that the number of occurrences of a code is proportional to the amount of time the signal spends within +/- ½ LSB of a code.

Next, the probability of each code must be calculated. The probabilities for a triangular wave input are straightforward. Fig. 49 shows a triangle wave input to an ADC for characterization of INL and DNL. The probability that a code 100 will occur during a single period is

$$P_{100} = \frac{t_{100}}{T/2}$$

where T is the period of the triangular wave. When calculating this probability for the code density test, this probability must be multiplied by the total number of periods in the simulation to yield the total number of expected hits for each code.

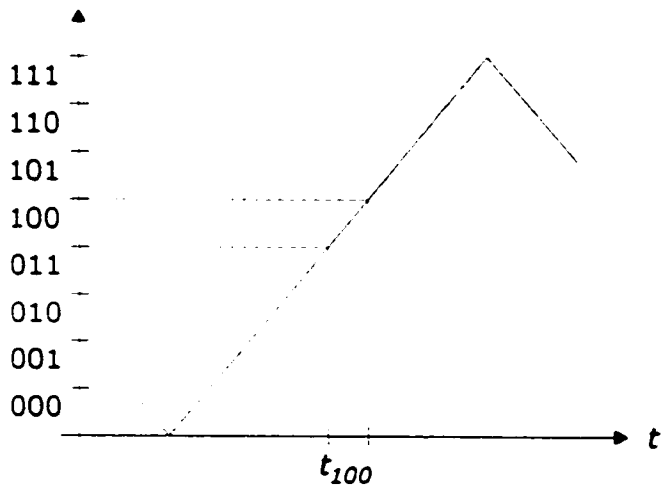


Fig. 49 Calculation of INL and DNL.

Deviations from the expected number of hits will reveal the INL and DNL of the ADC. Now the DNL is calculated from the number of predicted occurrences and the number of expected occurrences. The DNL is calculated as

$$DNL_n = \frac{measured_n - predicted_n}{predicted_n}.$$

and the INL is defined as

$$INL_n = \sum_{i=0}^{n} DNL_n .$$

Once the INL and DNL are calculated for every possible output, an output table is generated. This table is shown in Fig. 50. This shows the INL and DNL for each possible output code. In addition, the table also reports the expected number and the actual number of output samples that were recorded for each output bin.

Entry	Expected	Measured	DNL	INL
0	8.00	8	0.00	0.00
1	8.00	8	0.00	0.00
2	8.00	8	0.00	0.00
3	8.00	8	0.00	0.00
4	8.00	8	0.00	0.00
5	8.00	8	0.00	0.00
6	8.00	9	0.12	0.12
7	8.00	8	0.00	0.12
8	8.00	8	0.00	0.12
.				
.				
.				

Fig. 50 Example linearity output table.

This data is plotted as shown in Fig. 51. Typically, the DNL should be less than +/- 0.5 LSB for all output codes. The INL is within an acceptable range if all output codes are less than +/- 1.0 LSB.

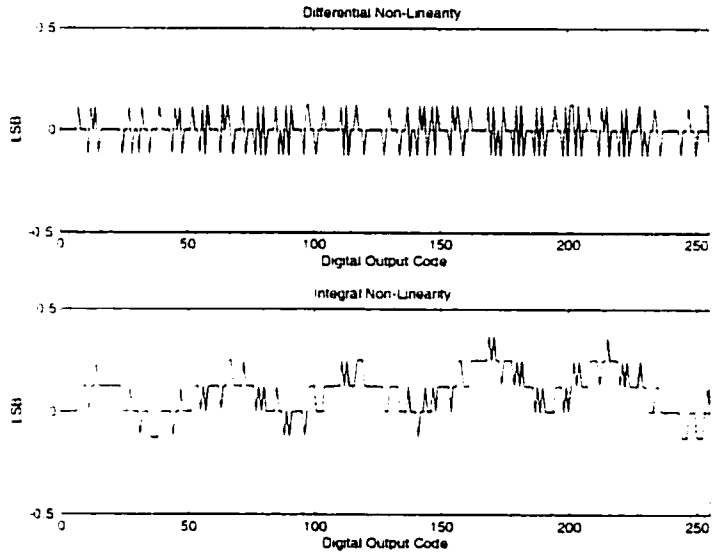


Fig. 51 Example linearity output results.

Chapter 6 Results

6.1 Pre-Fabrication Results

The design of an 8 bit, 1.8 V, 20 MS/s A/D Converter is complete. The HSPICE simulation results prior to fabrication are presented in this section. The ADC is designed in 0.25u CMOS technology.

The ADC design is complimented with the software infrastructure described in the previous chapter. This software allows for extraction of the raw digital data, implementation of the digital error correction algorithm, and the necessary performance tools to analyze and characterize the resulting ADC digital output.

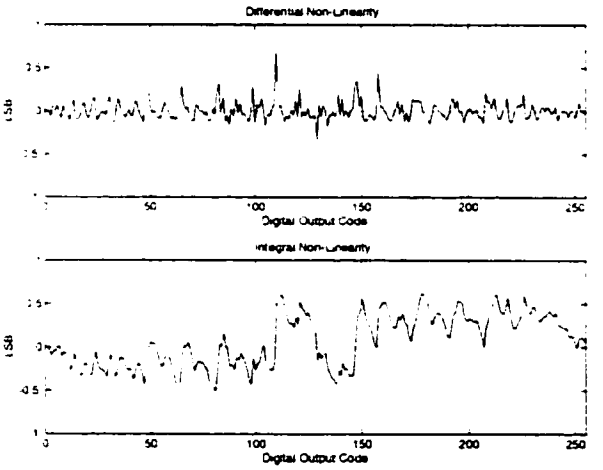


Fig. 52 Dynamic differential and integral nonlinearity results.

The HSPICE simulation results of the ADC are shown in Fig. 52 and Table 1. shows the dynamic nonlinearity of the ADC converter. The differential nonlinearity (DNL) of the device is +/- 0.6 LSB, and the integral nonlinearity (INL) is +/- 0.6 LSB.

Table 1 ADC summary.

Technology	0.25u CMOS
Resolution	8-bit
Supply Voltage	1.8 V
Power	50 mW
Conversion rate	20 MSampes/s
DNL	+/- 0.6 LSB
INL	+/- 0.6 LSB
SNR	49.58 dB
SNDR	44.54 dB
THD	-49.1 dB

Fig. 53 shows the frequency response of the simulated ADC output digital data. The third harmonic of the input signal is more prevalent in this design than data converters that utilize high gain amplifiers. Due to moderate gain, the switched-capacitor circuitry allows nonlinearities to propagate down the pipeline. This yields a larger overall nonlinearity and must be kept at a reasonable level during the design of the data converter.

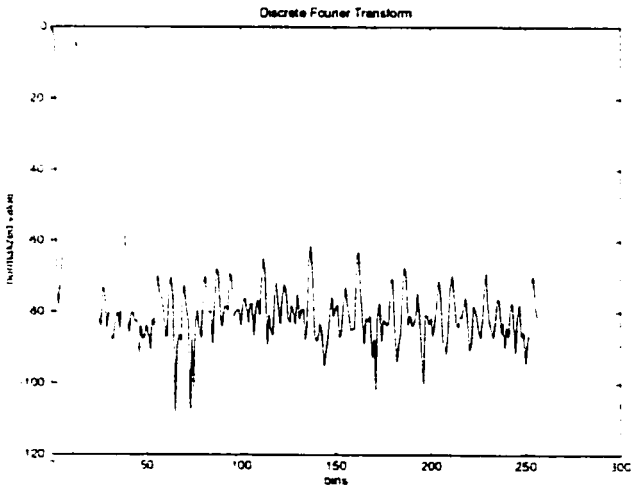


Fig. 53 Fast Fourier Transform of Output Data.

6.2 Prototype

6.2.1 Test Environment

A prototype pipelined analog-to-digital converter circuit was fabricated using the design techniques presented within. The IC was fabricated using the TSMC 0.25 micron process with the RF/Mixed-Signal option. Adding the RF/Mixed-Signal option allows for metal-to-metal capacitors and thick oxide for 3.3V circuitry—both used in this design. The metal-to-metal capacitors were utilized in each cell for implementing the sampling and feedback capacitors. Thick oxide was used for the transmission gate transistors and associated logic so it could support a higher voltage necessary for low resistance through the transmission gates. The die micrograph for the project is shown in Appendix C. The use of 3.3v drive on the transmission gates in this design was an attempt to simulate the use of a charge pump for the transmission gate drive voltages. This assured a low on resistance for the transmission gates.

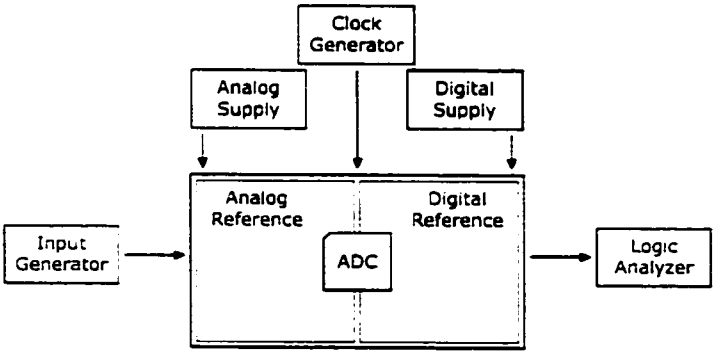


Fig. 54 Setup for Testing Prototype.

The test setup for this design is shown in Fig. 54. Special care should always be taken to separate and provide clean analog and digital power supplies as well as any bias voltages, currents, or reference signals. This design is no exception. Both the test board as well as the chip itself have cleanly defined and separated areas for analog and digital circuitry. The die micrograph is labeled and shown in Fig. 55. The overall dimensions of the chip are 1.7mm x 1.9mm, approximately 3.23mm^2. The majority of the active area of this chip is either unused as a result of the design being "pad limited" or used for the output buffers. The pipeline itself is approximately 0.4 mm^2 or 0.5mm x 0.8mm. A better image of the die micrograph is located in the Appendices.

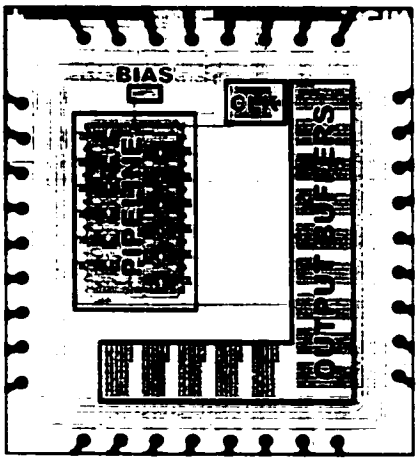


Fig. 55 Die Micrograph.

An ADC test board was designed to allow testing of the ADC. This test board is shown in Fig. 56. In high-speed, high-resolution data converters the design of a proper test board can become exceedingly difficult. The test board for this project provides several important functions for testing the ADC, all of equal importance. First, the test

board provides the ADC with regulated, low noise analog and digital supply voltages. In addition, the board takes every effort to separate all digital and analog signals and their return currents. Secondly, the test board provides a clean input path for the clock reference and the input signal through SMA connectors. The input signal is also converted from a single-ended signal to a differential signal centered around the common-mode input voltage. Thirdly, the test board generates low noise, low impedance input reference voltages and currents. Finally, the test board provides pins out the digital output data and output clocks to an edge connector for easy pick-up by a logic analyzer. The test board also provides a number of secondary features including digital noise suppression, and digital calibration controls. A schematic of the test board is located in the appendices.

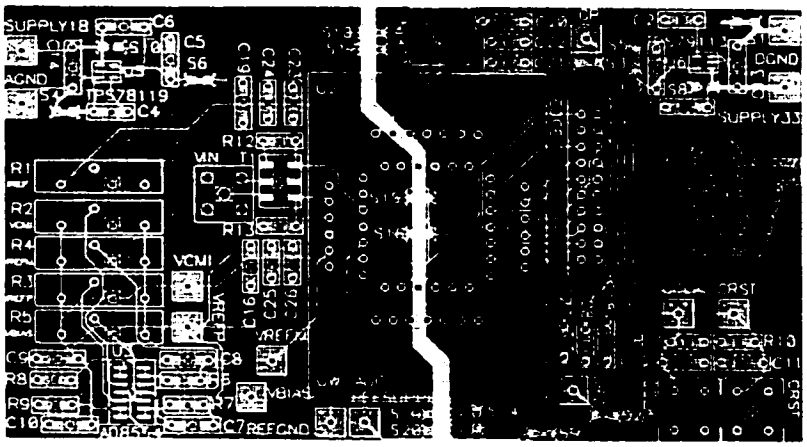


Fig. 56 ADC test board.

The finalized data converter specifications and designed are shown in Table 2. Designed in 0.25 micron CMOS, this 8-bit, 1.8 V converter can achieve 20 MS/s.

Table 2 Data Converter Specifications

Technology	0.25u CMOS
Resolution	8-bit
Supply Voltage	1.8 V
Conversion rate	20 MSampes/s

6.2.2 PROTOTYPE RESULTS

The data converter results in this section are divided into two sections: single-sample results, and double-sample results. The single-sample results are the performance of the data converter without the use of the double-sampling technique innovated through this research. The double-sample results include all the "bells and whistles" included in this ADC design.

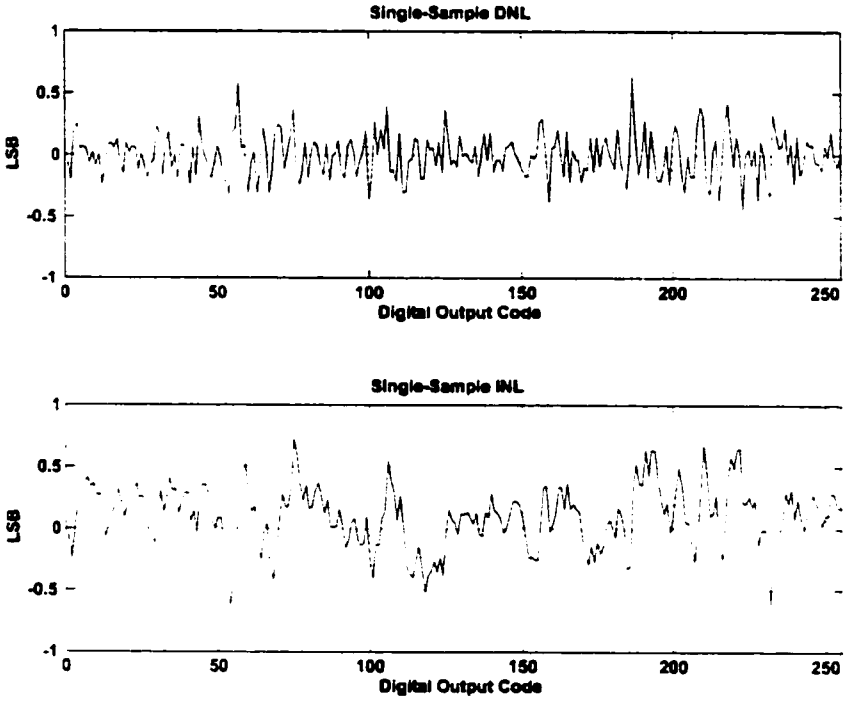


Fig. 57 Single-Sample Non-Linearity.

The single-sample ADC converter nonlinearity is shown in Fig. 57. The ADC yielded nearly identical results to the simulated results shown in the previous section. Exceptional linearity is difficult to achieve in a low-gain environment. Thus, the DNL of +/- 0.6 and INL of +/- 0.7 is a welcomed result.

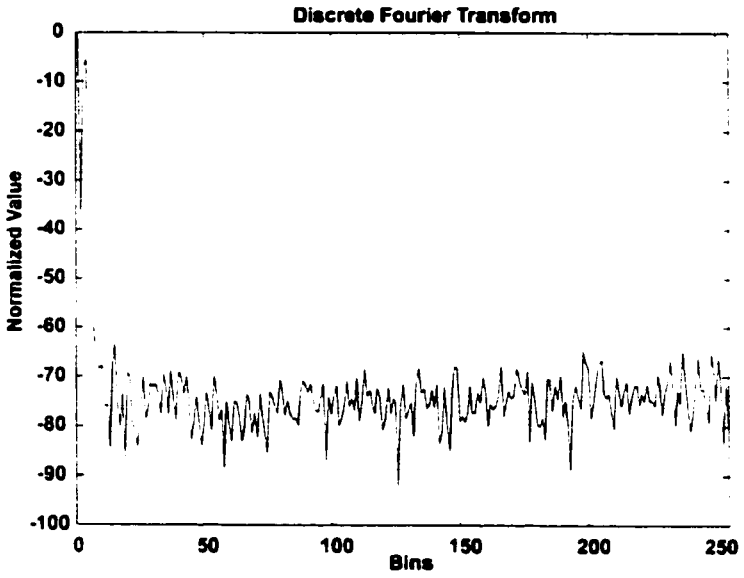


Fig. 58 Single-Sample Frequency Response.

The Fourier transform of the ADC output is shown in Fig. 58. This appropriate frequency response is summarized with the single-sample converter results in Table 3. In single-sample mode, the output data rate is 10 MS/s.

Table 3 ADC single-sample summary.

Technology	0.25u CMOS
Resolution	8-bit
Supply Voltage	1.8 V
Analog Power	18 mW
Digital Power	68 mW
Conversion rate	10 MS/s
DNL	+/- 0.6 LSB
INL	+/- 0.7 LSB
SNR	48.47 dB
SNDR	43.83 dB
THD	-51.51 dB
SFDR	58.57 dB

The double-sample ADC converter nonlinearity is shown in Fig. 59. The ADC yielded nearly identical results to the simulated results shown in the previous section. Again, the DNL of +/- 0.45 and INL of +/- 0.65 is a welcomed result.

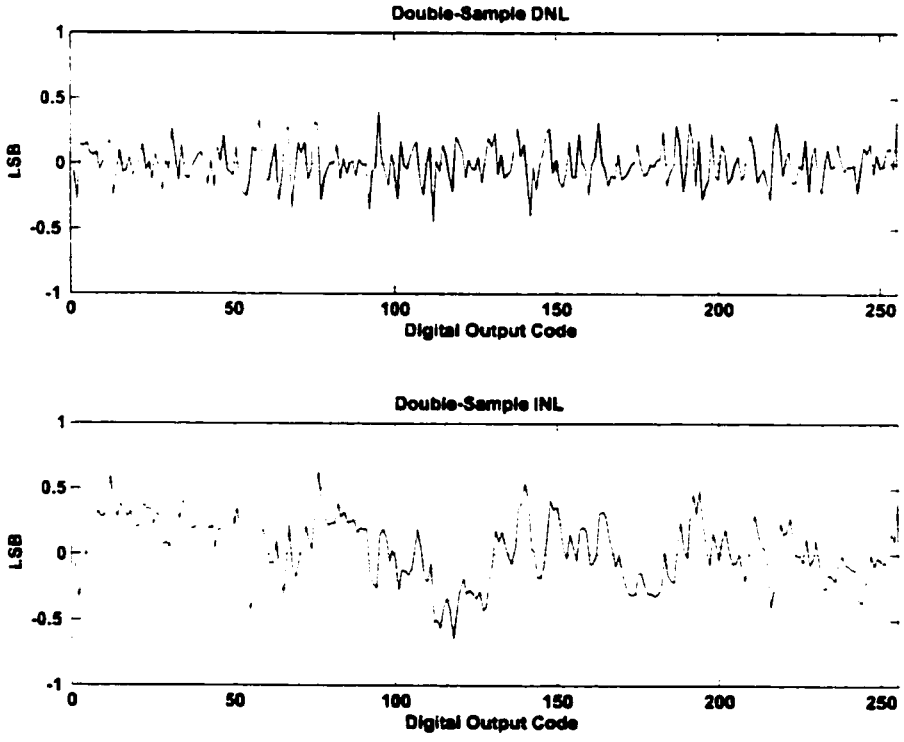


Fig. 59 Double-Sample Non-Linearity.

The Fourier transform of the ADC output is shown in Fig. 60. This appropriate frequency response is summarized with the double-sample converter results in Table 3. In double-sample mode, the output data rate is 20 MS/s. Although it is hard to see on the frequency plot, there is a spike at fs/2. This is due to the double-sampling scheme and to a phenomena called "pattern error". The regular pattern that appears between adjacent samples is due to many things including capacitor mismatch, device mismatch, and clock skew. Every effort should be made to minimize possible mismatches when designing and laying out the data converter. In this case the pattern error reduced the SNDR approximately 3 dB.

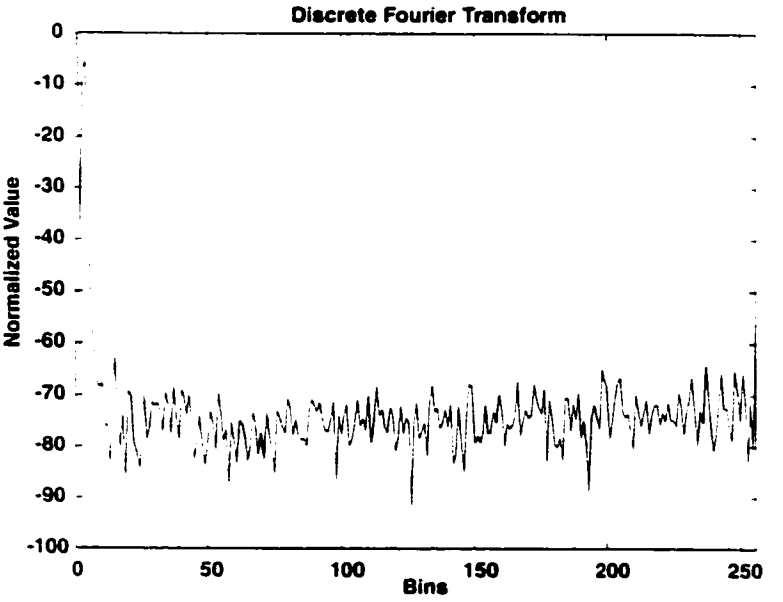


Fig. 60 Double-Sample Frequency Response.

Fig. 61 shows the effective number of bits (ENOB) versus input frequency. The ENOB of the converter at low frequencies is approximately 7.3 bits and it decreases to approximately 5.8 bits as the input frequency of 9 MHz which is approaching fs/2. This is a very respectable result for an 8-bit converter without an input sample and hold to extend the input frequency response to higher values.

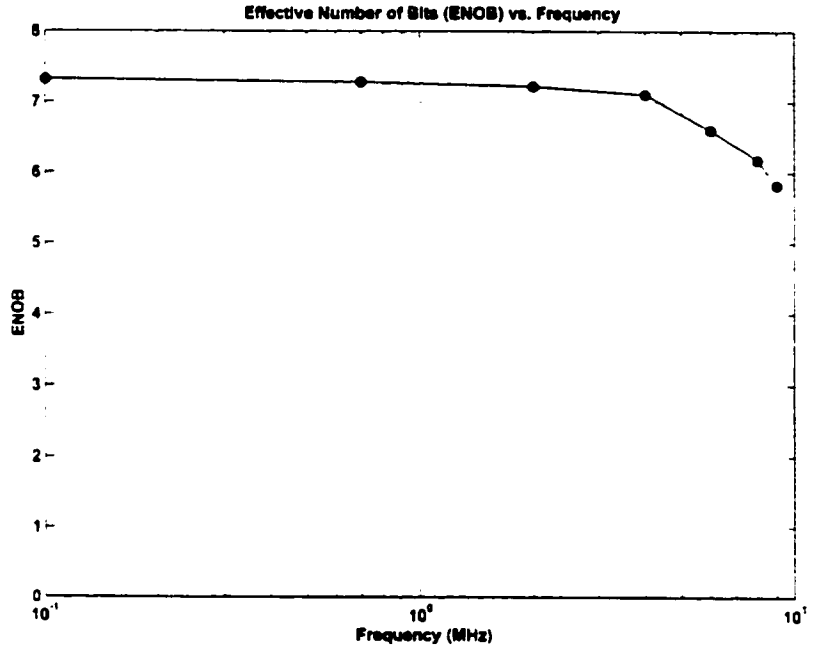


Fig. 61 Effective Number of Bits vs. Frequency.

Overall the results show the feasibility of using low-gain amplifiers and digital error correction to create a high-performance, high-speed data converter that utilizes low power and low die area. In addition, the low-gain double-sampling technique is shown to function exceptionally where the standard double-sampling technique fails.

Table 4 ADC double-sample summary.

Technology	0.25u CMOS
Resolution	8-bit
Supply Voltage	1.8 V
Analog Power	18 mW
Digital Power	68 mW
Conversion rate	20 MS/s
DNL	+/- 0.45 LSB
INL	+/- 0.65 LSB
SNR	44.25 dB
SNDR	40.09 dB
THD	-48.47 dB
SFDR	59.69 dB

One drawback to the test chip was its inability to run at the full data rate of 100 MS/s. This was due to the fact that the tail current in the amplifier was reduced significantly to properly bias the amplifiers which in turn lowered the maximum bandwidth of the data converter. In this design the inclusion of a tail current transistor not connected in feedback would have allowed an additional degree of freedom and improved the performance of the converter. This drawback aside, the test chip proved the concepts and design approaches described in this paper.

Chapter 7 Conclusion and Summary

7.1 Research Summary

In addition to being both high-speed and low-power, the data converter presented in this paper also has a number of circuit innovations that make it possible. First, this ADC utilizes simple differential pair amplifiers with continuous-time common-mode feedback with gain well below what would typically be used for a data converter of this resolution. This design relies on digital error correction to correct for low amplifier gain and the non-ideal effects that result. Relaxing that gain requirement makes the design of high-speed and low-power amplifiers more obtainable. Second, the design is uses a novel low-gain double-sampling architecture that functions exceptionally in a low gain environment. One additional caveat that resulted from this research is the removal of the previous requirement of Karanicolas' digital error correction approach to use calibration stages with an ideal gain of 2. This design uses calibration stages with a mere 1.65 gain.

This research set out to achieve a high-speed, low-voltage data converter with the secondary goals of relative process independence, amenable to system-on-chip integration, low-power, and small die area. The data converter designed and described in this paper did just that.

7.2 Future Areas of Research

7.2.1 Time-Interleaved Conversion

Another technique to improve performance that can be implemented in conjunction with the pipelined converter design described in this paper, is to time-interleaving multiple pipelined converters [10][22-28]. Essentially, the number of digital outputs per second is multiplied by the interleave factor.

Fig. 62 shows the architecture of an interleaved converter [10]. Each A/D converter shown in the diagram is a full pipeline converter. Since the presented pipelined converter performs 100 MS/s, the interleaved version will perform at 400 MS/s. The die area consumed is roughly 4 times more than the non-interleaved converter.

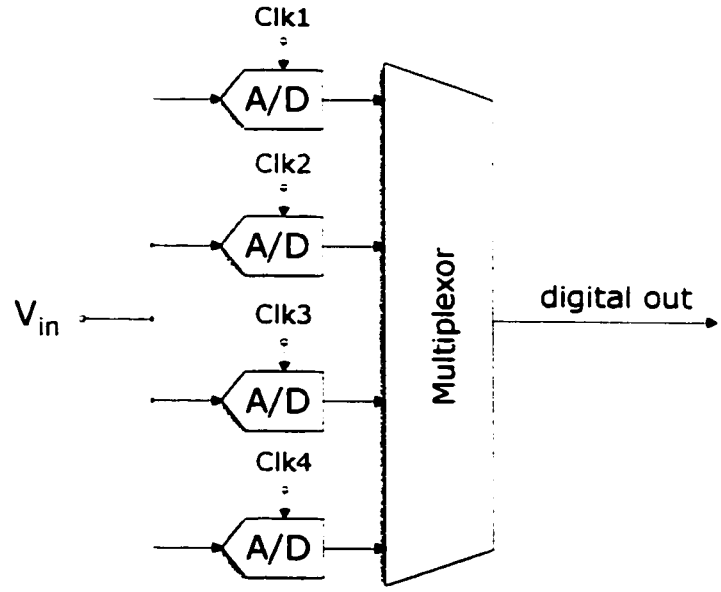


Fig. 62 Time-interleaved converter architecture.

The time interleaved approach requires a separate clock for each of the pipelined converters. Each of the clocks will be out of phase of one another. This will ensure that the interleaved pipeline will convert the incoming analog signal at regular intervals. Fig. 63 shows the clock timing diagrams for the interleaved converter. By forcing the A/D converters to convert the incoming waveforms out of phase of one another, the overall conversion rate is improved. Once the signal is converted to digital, the back end multiplexor interleaves the outputs onto a single output data stream. The multiplexor takes the four 100 MS/s converter outputs and converts them into one 400 MS/s output.

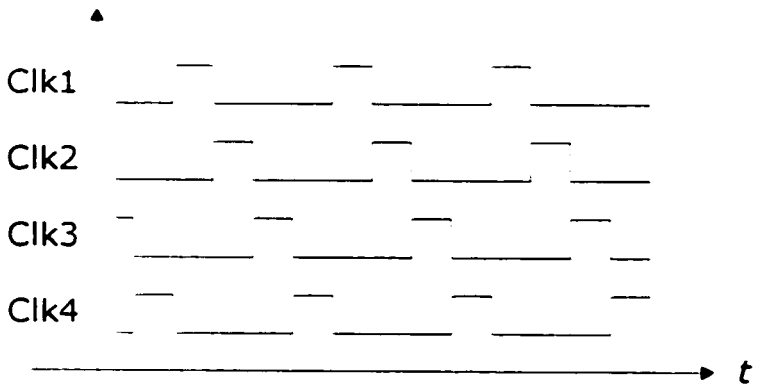


Fig. 63 Time-interleaved clock timing diagram.

This notion of interleaving multiple pipelined converters to create a single high performance output is the primary motivation for future work. With a successful implementation, the final A/D conversion performance will be 400 MS/s.

7.2.2 AMPLIFIER NON LINEARITY

Another topic of continuing research is the design of circuit techniques and correction techniques to improve amplifier non linearity. This is of particular interest when working with designs that utilize moderate- to low-gain amplifiers. The digital correction techniques presented in this research do not correct for amplifier non linearities. This design relies on minimizing amplifier non linearity by circuit design. Additional improvements in amplifier linearity would allow similar converter designs to achieve higher resolution output with virtually no other changes to the design.

Bibliography

[1] B.M. Gordon, "Linear Electronic Analog/Digtal Conversion Architectures, Their Origins, Parameters, Limitations, and Applications", *IEEE Trans. Circuits Syst.*, vol. CAS-25, pp. 391-418, July 1978.

[2] B.S. Song, S.H. Lee, and M.F. Tompsett, "A 10-Bit 15-MHz CMOS Recycling Two-Step A/D Converter," *IEEE J. Solid State Circuits*, vol. SC-20, pp. 780-786, June 1985.

[3] M. Steyaert, R. Roovers, and J. Cranickx, "A 100 MHz 8 Bit CMOS Interpolating A/D Converter," *Proc. CICC*, pp. 28.1.1-28.1.4, May 1993.

[4] K. Kusumoto, A. Matsuzawa, and K. Murata, "A 10-b 20-MHz 30-mW Pipelined Interpolating CMOS ADC," *IEEE J. Solid State Circuits*, vol. SC-28, pp. 1200-1206, Dec. 1993.

[5] Principles of Data Conversion System Design, Behzad Razavi, 1995.

[6] R.M. Ziazadeh, H.-T. Ng, and D.J. Allstot, "A Multistage Amplifier with Embedded Tracking Compensation," *IEEE Custom IC Conf. (CICC)*, May 1998, pp. 361-364.

[7] H.-T. Ng, R. Ziazadeh, and D.J. Allstot, "A Multistage Amplifier Technique using Embedded Frequency Compensation," *IEEE J. Solid-State Circuits*, vol. 34, pp. 339-347, March 1999.

[8] A.N. Karanicolas, H.S. Lee, and K.L. Bacrania, "1 15b 1MS/s Digitally Self-Calibrated Pipelined ADC", *IEEE J. Solid-State Circuits*, vol. C-27, pp. 1679-1688, Dec. 1992.

[9] B. Ginetti, P. Jespers, and A. Vandemeulebroecke, "A CMOS 13-b Cyclic RSD A/D Converter", *IEEE J. Solid-State Circuits*, vol. 27, no. 7, July 1992.

[10] W. Black and D.A. Hodges, "Time Interleaved Converter Arrays," *IEEE J. Solid-State Circuits*, vol. SC-15, pp. 1022-1029, Dec. 1980.

[11] J. Doernberg, H.-Seung Lee, and D.A. Hodges, "Full-speed testing of A/D converters," *IEEE J. Solid State Circuits*, vol. SC-19, no. 6, pp. 820-827, Dec. 1984.

[12] D. Garrity, S. Aftab, "A 10 bit, 40MS/s BiCMOS Pipelined A/D Converter", *ASICON '98*.

[13] Paul C. Yu, and Hae-Seung Lee, "A 2.5-V, 12-b 5-Msample/s Pipelined CMOS ADC", *IEEE J. Solid State Circuits*, vol. 31, no. 12, pp. 1854-1861, Dec. 1996.

[14] Y. Park, "A 1.8V 10b 100Msample/s CMOS Pipelined ADC with 1.8V Power Supply", *ISSCC 2001*, paper 8.3, pp. 130-131.

[15] E. Soenen, R. Geiger, "An Architecture and An Algorithm for Fully Digital Correction of Monolithic Pipelined ADC's", *IEEE Trans. Circuits Syst. II*, pp. 143-153, vol. 42, no. 3, March 1995.

[16] D. Fu, K. Dyer, S. Lewis, P. Hurst, "Digital Background Calibration of a 10b 40Msample/s Parallel Pipelined ADC" *ISSCC 1998*, pp. 140-141.

[17] T. B. Cho, and P. R. Gray, "A 10-b 20 Msample/s, 25 mW Pipeline A/D Converter", *IEEE J. Solid State Circuits*, vol. 30, no. 3, pp. 166-172, March 1995.

[18] J. Ming, and S. Lewis, "An 8-bit 80-Msample/s Pipelined Analog-to-Digital Converter With Background Calibration", *IEEE J. Solid State Circuits*, vol. 36, no. 10, pp. 1489-1497, October 2001.

[19] Seung-Hoon Lee and Bang-Sup Song, "Digital-Domain Calibration of Multistep Analog-to-Digital Converters", *IEEE J. Solid State Circuits*, pp. 1679-1688, vol. 27, no. 12, December 1992.

[20] H. C. Yang, D. J. Allstot, "Considerations for Fast Settling Operational Amplifiers", *IEEE Trans. Circuits Syst.*, pp. 326-334, vol. 37, no. 3, March 1990.

[21] H. C. Yang, M. A. Abu-Dayeh, and D. J. Allstot, "Small-Signal Analysis and Minimum Settling Time Design of a One-Stage Folded-Cascode CMOS Operational Amplifier", *IEEE Trans. Circuits Syst.*, pp. 804-807, vol. 38, no. 7, July 1991.

[22] K. Poulton, R. Neff, A. Muto, W. Liu, A. Burstein, M. Heshami, "A 4Gsample/s 8b ADC in 0.35μm CMOS", *ISSCC 2002*, paper 10.1, pp. 166-167.

[23] M. Waltari, and K. Halonen, "1-V 9-bit Switched-Opamp ADC", *IEEE J. Solid State Circuits*, vol. 36, no. 1, pp. 129-134, January 2001.

[24] S. Jamal, "A 10b 120MSample/s Time-Interleaved Analog-to-Digital Converter with Digital Background Calibration", *ISSCC 2002*, paper 10.4, pp. 172-173.

[25] K. Nagaraj, H. Scott Fetterman, J. Anidjar, S. H. Lewis, and R. G. Renninger, "A 250-mW, 8-b, 52-Msamples/s Parallel-Pipelined A/D Converter with Reduced Number of Amplifiers", *IEEE J. Solid State Circuits*, vol. 32, no. 3, pp. 312-320, March 1997.

[26] K. Nakamura, L. Richard Carley, and D. J. Allstot, "An 85mW, 10 b, 40 Msample/s CMOS Parallel-Pipelined ADC", *IEEE J. Solid State Circuits*, vol. 30, no. 3, pp. 173-183, March 1995.

[27] H. Jin, E. Lee, and M. Hassoun, "Time-Interleaved A/D Converter with Channel Randomization", *IEEE Symp. On Circuits and Systems*, pp. 425-428, June 1997.

[28] C. S. G. Conroy, D. W. Cline, and P. R. Gray, "An 8-b 85-MS/s Parallel Pipeline A/D Converter in 1-μm CMOS", *IEEE J. Solid State Circuits*, vol. 28, no. 4, pp. 447-454, April 1993.

[29] V. Navin, T. Ray, M. Hassoun, W. Black, E. Lee, E. Soenen, and R. Geiger, "A Simulation Environment for Pipelined Analog-to-Digital Converters", *IEEE Symp. On Circuits and Systems*, pp. 1620-1623, June 1997.

[30] Y-T. Wang, and B. Razavi, "An 8-bit 150-MHz CMOS A/D Converter", *IEEE J. Solid State Circuits*, vol. 35, no. 3, pp. 308-317, March 2000.

[31] A. Petraglia, and S. K. Mitra, "Analysis of Mismatch Effects Among A/D Converters in a Time-Interleaved Waveform Digitizer", *IEEE Trans. On Instrumentation and Measurement*, pp. 831-835, vol. 40 no. 5, October 1991.

[32] H-S. Chen, and K. Bacrania, "A 14-b 20-Msamples/s CMOS Pipelined ADC", *IEEE J. Solid State Circuits*, vol. 36, no. 6, pp. 997-1001, June 2001.

[33] L. Sumanen, M. Waltari, and K. A. I. Halonen, "A 10-bit 200-MS/s CMOS Parallel Pipeline A/D Converter", *IEEE J. Solid State Circuits*, vol. 36, no. 7, pp. 1048-1055, July 2001.

[34] Marcel J. M. Pelgrom, AAD C. J. Duinmaijer, and Anton P. G. Welbers, "Matching Properties of MOS Transistors", *IEEE J. Solid State Circuits*, vol. 24, no. 5, pp. 1433-1439, December 1996.

[35] K. Y. Kim, N. Kusayanagi, and A. A. Abidi, "A 10-b 100-MS/s CMOS A/D Converter", *IEEE J. Solid State Circuits*, vol. 32, no. 3, pp. 302-311, March 1993.

[36] Michael P. Flynn, and David J. Allstot, "CMOS Folding A/D Converters with Current-Mode Interpolation", *IEEE J. Solid State Circuits*, vol. 31, no. 9, pp. 1248-1257, September 1996.

[37] Ardie G. W. Venes, and Rudy J. van de Plassche, "An 80-MHz, 80-mW, 8-b CMOS Folding A/D Converter with Distributed Track-and-Hold Preprocessing", *IEEE J. Solid State Circuits*, vol. 31, no. 12, pp. 1846-1853, December 1996.

[38] B. Nauta, and Ardie G.W. Venes, "A 70Msamples/s 110mW 8b CMOS Folding Interpolation A/D Converter", ISSCC 95, paper FA 16.3, pp. 276-277.

[39] B. 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, December 1999.

[40] S. Mortezapour, and E.k. F. Lee, "A 1-V, 8-bit Successive Approximation ADC in Standard CMOS Process", *IEEE J. Solid State Circuits*, vol. 35, no. 4, pp. 642-646,, April 2000.

[41] K. Bult, and A. Buchwald, "An Embedded 240-mW 10-b 50-MS/s CMOS ADC in 1-mm^2", *IEEE J. Solid State Circuits*, vol. 32, no. 12, pp. 1887-1895, December 1997.

[42] M. Choi, and A. Abidi, "A 6-b 1.3-Gsample/s A/D Converter in 0.35-μm CMOS", *IEEE J. Solid State Circuits*, vol. 36, no. 12, pp. 1847-1858, December 2001.

[43] P. Scholtens, "A 6b 1.6Gsample/s Flash ADC in 0.18μm CMOS Using Averaging Termination", *ISSCC 2002*, paper 10.2, pp. 168-169.

[44] H. van der Ploeg, and R. Remmers, "A 3.3-V, 10-b, 25-Msample/s Two-Step ADC in 0.35-μm CMOS", *IEEE J. Solid State Circuits*, vol. 34, no. 12, pp. 803-811, December 1999.

[45] T.B. Cho, and P.R. Gray, "A 10bit 20MS/s, 35mW pipeline A/D converter," *Proc. Custom Integrated Circuits Conference*, May 1994, pp. 23.2.1 - 23.2.4.

APPENDIX A: Software Reference

ADC Development Software Overview

An approach for design, simulation, and characterization of high performance pipelined analog to digital converter (ADC) architectures has been developed. This approach is outlined graphically as follows:

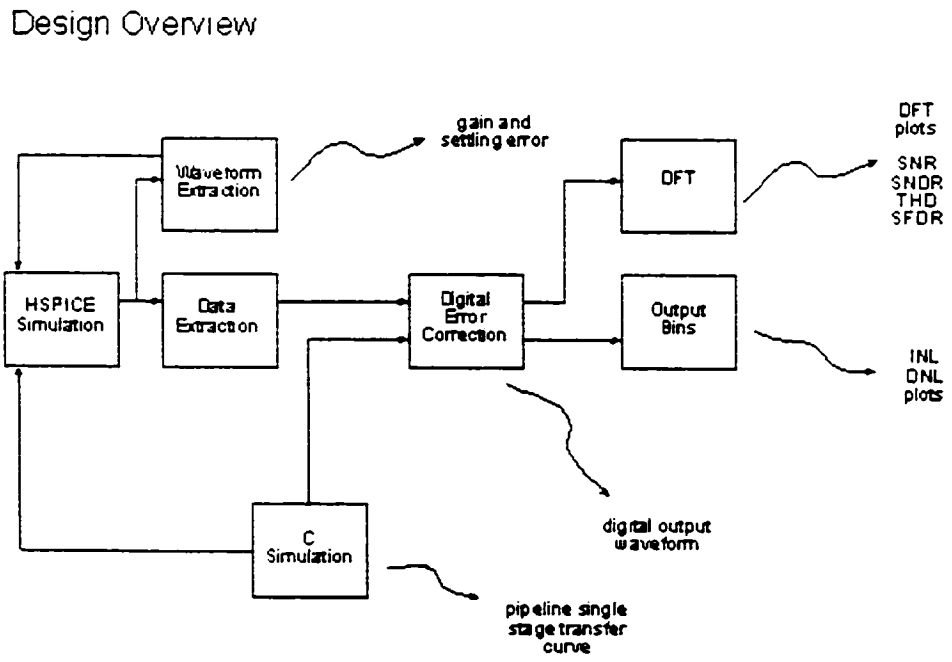


Fig. 64 Software Design Roadmap.

Description

The approach allows the designer to begin with C simulation prior to committing to time consuming HSPICE simulations. Regardless of the source of the binary ADC output, the software will characterize and analyze the output data stream.

When using HSPICE to simulation the ADC, the first step is to extract the digital output from the digital output waveforms: bit0, bit1, etc. The software created to extract the data is named data.

If you would rather begin with simulated ADC output data, rather than investing the time to create an entire ADC HSPICE simulation, the software calib was created just for that reason. It allows you to simulate any number of pipelined ADC converter topologies as well as introduce non-linearities into the simulation to make the simulation more 'true to life'.

Next, the binary output data must be processed and converted to decimal. The program I designed to simulate this process is digital. This program will decode the binary digital data and perform an error correction algorithm [1] on the data. The output from this program is digitally corrected decimal data. The decimal data is corrected for gain and comparator offset.

Now that the output data has been processed and corrected, there is a need to characterize the performance and output characteristics of the ADC output data. These results will give a feel for the quality and performance of the ADC itself.

The fft program was designed to take the corrected digital output data and perform a Fast Fourier Transform (FFT) on the data. From this FFT, the software will calculate the Signal to Nois Ratio (SNR), the Signal to Noise and Distortion Ratio (SNDR), the Total Harmonic Distortion (THD), and the Spurious Free Dynamic Range (SFDR) of the ADC output data. This information is very useful when trying to characterize the performance of the ADC converter.

Another important performance measure of ADC's is the Integral Non-Linearity (INL) and the Differential Non-Linearity (DNL) of the ADC output data. These two plots are calculated with linear. This program will divide the ADC output into specific output bins. Once these bins are filled will all the available output data, the INL and DNL plots are calculated.

This approach will facilitate the design of high performance ADCs.

ADC Pipeline Behavioral Model

This behavioral model of the ADC pipeline uses the input parameters to simulate the functionality of the ADC pipeline under certain conditions. The program generates a calibration waveform followed by an output waveform. The calibration waveform is designed to implement Karanicolas' Digital Error Correction Algorithm [1]. This correction algorithm corrects the output for gain and comparator offsets. The calibration waveform is followed by an output waveform that is used to test the performance of the A/D converter.

Command Format and Parameters

The following is a breakdown of the command line arguments. This breakdown is automatically displayed when the program is executed without command line arguments.

```
Digital Calibration Pipeline Output Generator   By Douglas Beck
======================================================== 11/18/98

Command format:

            calib [outfile] <options>

            calibration options:

               -h      [hold off cycles before calibration]
               -s      [pipestages] [calibstages]
               -r1     [pipe Cf]  [pipe Cs]  [pipe Ca]
               -r2     [calib Cf] [calib Cs] [calib Ca]
               -x      skip calibration sequence
               -i      [calibration samples per stage]

            pipeline output options:

               -ramp   ramp input after calibration
               -sine   sine wave input after calibration
               -tri    triangle wave input after calibration
```

```
    -f      [ratio of input freq to sample freq]
    -gamma  [nonlinearity factor]
    -o      [number of output cycles after calib]

output options:

    -t      [filename] output transfer function
    -tm     [filename] output transfer function
            in Matlab format
    -plot   [filename] transfer function plot code
```

Description of Parameters

Most importantly, this program does not require an input file. It only requires input parameters to get the information it needs to run the simulation. The output waveforms are output digitally to the output file.

The "-h" command will force the simulation to start outputting clock cycles at the beginning of the simulation with no calibration data. This is simply to match a SPICE simulation that requires some time for the transients to settle out and for the circuit to reach steady state. This value defaults to 0 (no hold-off period).

The "-s" command defines the number of pipeline stages and the number of calibration stages for the simulation. This defaults to 8 pipeline stages with 6 calibration stages.

The "-r1" command defines the transfer function for the pipeline stages. The three capacitances Cf, Cs, and Ca are used to define the transfer curve. To define a perfect transfer curve with a gain of 2 the parameter would be `-r1 1 1 0`. To define a pipeline with a gain of less than 2, a sample input would be `-r1 1 1 .1`. This results in a stage gain of approximately 1.95. This parameter defaults to `-r1 1 1 0` which results in an ideal stage gain of 2.

The "-r2" command defines the transfer function for the calibration stages. This parameter functions identically to -r1 except that this parameter defines the calibration stages rather than the pipeline stages. This parameter defaults to `-r1 1 1 0` which results in an ideal stage gain of 2.

The "-x" command will skip the calibration sequence altogether. This makes the assumption that the ADC does not need to be calibrated and just needs to the output data constructed from the pipeline stage outputs. The omission of this parameter forces the digital error correction algorithm to take place.

The "-i" command defines the number of clock cycles that each calibration output will remain active before proceeding to the next calibration output. This parameter defaults to 8 clock cycles.

The "-tri" command selects a triangular test waveform for output following the calibration output waveform.

The "-sine" command selects a sinusoidal test waveform for output following the calibration output waveform.

The "-ramp" command selects a ramp test waveform for output following the calibration output waveform. The simulation defaults to a ramp test waveform.

The "-f" command defines the ratio of the test waveform frequency divided by the ADC sampling frequency. This parameter defaults to 0.25. This parameter should be an irrational number for good FFT results.

The "-gamma" command defines the nonlinearity of the transfer curve. A value of 1.000 will create a perfectly linear transfer curve. For values less than or greater than 1.000 the transfer curve will bend upward or downward respectively. This parameter defaults to a value of 1.000.

The "-o" parameter defines the number of clock cycles that the output test waveform will simulate. This parameter defaults to 512.

The "-t" command will output the transfer curve data to a file. When this parameter is omitted, no transfer curve output file will be created.

The "-tm" command is similar to the -t command in that it will output the transfer curve data to a file. In this case, the output file is in Matlab format. When this parameter is omitted, no Matlab transfer curve output file will be created. The filename should end in .m so that it will be recognized by Matlab as a script.

The "-plot" command will create a plot file for Matlab. This file will simply format and plot the Matlab data. The filename should end in .m so that it will be recognized by Matlab as a script.

Sample Run

To illustrate the functionality of the **calib** command, a sample is presented below. This sample command is executed with the parameters as shown:

```
calib out.txt -x -ramp -gamma 1.001 -o 32768 -tm t.m -plot tp.m
```

Reviewing the command line parameters above, we see that the calibration waveform will be created under the filename `out.txt`. After calibration, the test waveform will be a ramp function. The transfer curve a single pipeline has some nonlinearity. This is simulated with a gamma function of 1.001. There will be 32768 output ticks in the output file. The Matlab compatible output data will be stored in the file `t.m`. And the commands to format the plot window and label the graph will be stored in the file `tp.m`.

As the calibration simulation executes, it displays the following information to the console:

```
Digital Calibration Pipeline Output Generator   By Douglas Beck
=================================================== 11/18/98

Initializing Simulation
Pipeline Simulation Parameters

  Output filename    : out.txt
  Pipeline    Stages: 8
  Calibration Stages: 6
  Pipeline function  : 2.000*vin+1.000*vdac
  Calibration funct  : 2.000*vin+1.000*vdac
  Samples per Stage  : 8
  Holdoff cycles     : 32
  Output cycles      : 32768
  Output freq ratio  : 0.213560
  Nonlinearity fac   : 1.001000
```

```
  Output waveform    : ramp

Simulating Output
Outputting Matlab Formatted Transfer Functions
Outputting Matlab Plot Code
Simulation Complete
```

Sample Run Explanation

In the case where the parameters or the input file is missing or incorrect, an error message will appear and the extraction process is terminated.

Finally, upon successful simulation of the calibration waveform, an output file is created. A sample output is shown below:

```
011111111111111-0
000000000000000-1
000000000000000-2
000011111111100-3
000001111111110-4
000000000000000-5
000000000000000-6
000000001111111-7
000000000111111-8
...
```

In the data above, each character represents one bit output of each stage. The bit outputs are followed on each line by the line number. This is added for synchronization with the other software tools in this ADC software collection. At this point, the output data is in the correct format to input into the digital error correction simulation software (`digital`).

In addition to the raw bit output data, a plot of the transfer function for both the pipeline stages and the calibration stages is generated. In order for the program to generate the transfer function plot, the -tm and the -p parameters must be used. The resulting transfer function plot from Matlab appears as follows:

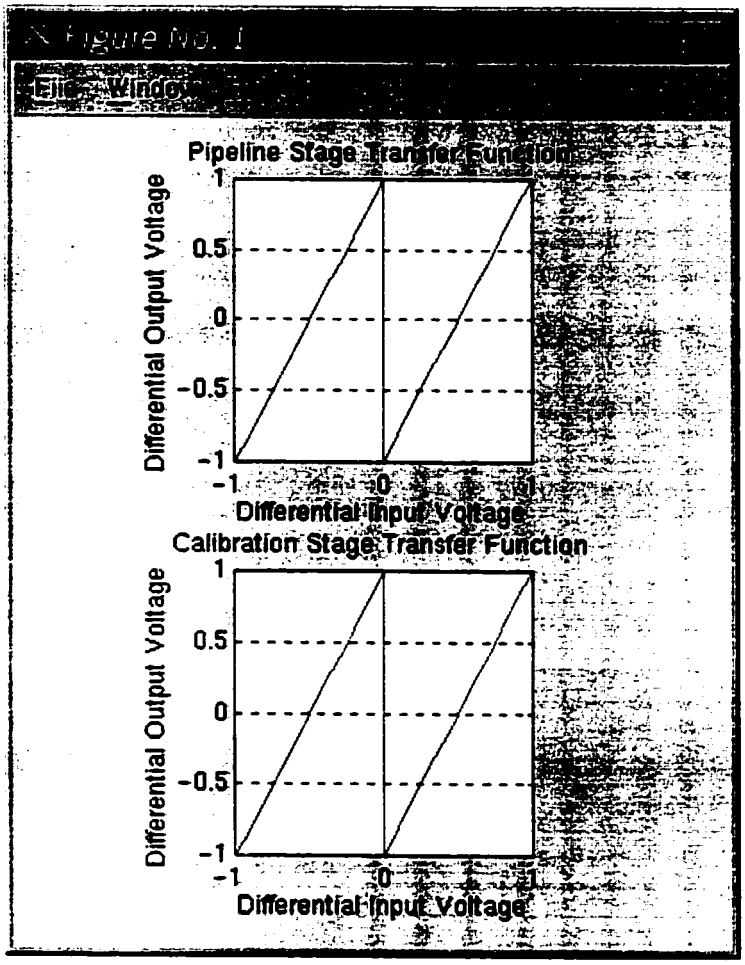


Fig. 65 Software Transfer Function Output.

At this point, the data is now ready for processing or plotting by the digital error correction software (`digital`).

Future Additions/Modifications to this Program

No plans for additions.

Known Problems

There are currently no known problems with this software.

Digital Error Correction Software

Description

This program will decode the binary digital data and perform an error correction algorithm [1] on the data. This program outputs a series of decimal values that are digitally corrected for gain and comparator offset.

Command Format and Parameters

The following is a breakdown of the command line arguments. This breakdown is automatically displayed when the program is executed without command line arguments.

```
Digital Error Correction Algorithm              By Douglas Beck
================================================== 11/18/98

Command format:

            digital [infile] [outfile] <options>

            calibration options:

               -h      [hold off cycles before calibration]
               -s      [pipestages] [calibstages]
               -i      [calibration samples per stage]
               -x      no digital calibration

            pipeline output options:

               -m      [filename] output Matlab data
               -mp     [filename] output Matlab plot code
               -o      [number of output cycles after calib]
```

Description of Parameters

The "-h" command delay the beginning of the calibration sequence by the number of clock cycles specified. This is simply to match a SPICE simulation that requires some time for the transients to settle out and for the circuit to reach steady state. This value defaults to 0 (no hold-off period).

The "-s" command defines the number of pipeline stages and the number of calibration stages for the simulation. This defaults to 8 pipeline stages with 6 calibration stages.

The "-i" command defines the number of clock cycles that each calibration output will remain active before proceeding to the next calibration output. This parameter defaults to 8 clock cycles.

The "-x" command will skip the calibration sequence altogether. This makes the assumption that the ADC does not need to be calibrated, but just needs the output data constructed from the pipeline stage outputs. The omission of this parameter forces the digital error correction algorithm to take place.

The "-m" command will create an additional output file. Rather than creating a raw decimal output datafile, the output file is created in Matlab format. The output is ready for direct import into Matlab. The filename should end in .m so that it will be recognized by Matlab as a script.

The "-mp" command will create a plot file for Matlab. This script will format and plot the Matlab data. The filename should end in .m so that it will be recognized by Matlab as a script.

The "-o" parameter defines the number of clock cycles to perform digital error correction following a successful calibration sequence. This parameter defaults to 512.

Sample Run

To illustrate the functionality of the **digital** command, a sample is presented below. This sample command is executed with the parameters as shown:

```
digital input.txt output.txt -x -m data.m -mp plot.m -o 32768
```

Reviewing the command line parameters above, we see that the digital software will take the input file input.txt and create a decimal output file named output.txt and a Matlab formatted data file (data.m) and Matlab plot code (plot.m). The software will output a 32768 points of the input waveform after calibration.

As the digital error correction software executes, it displays the following information to the console:

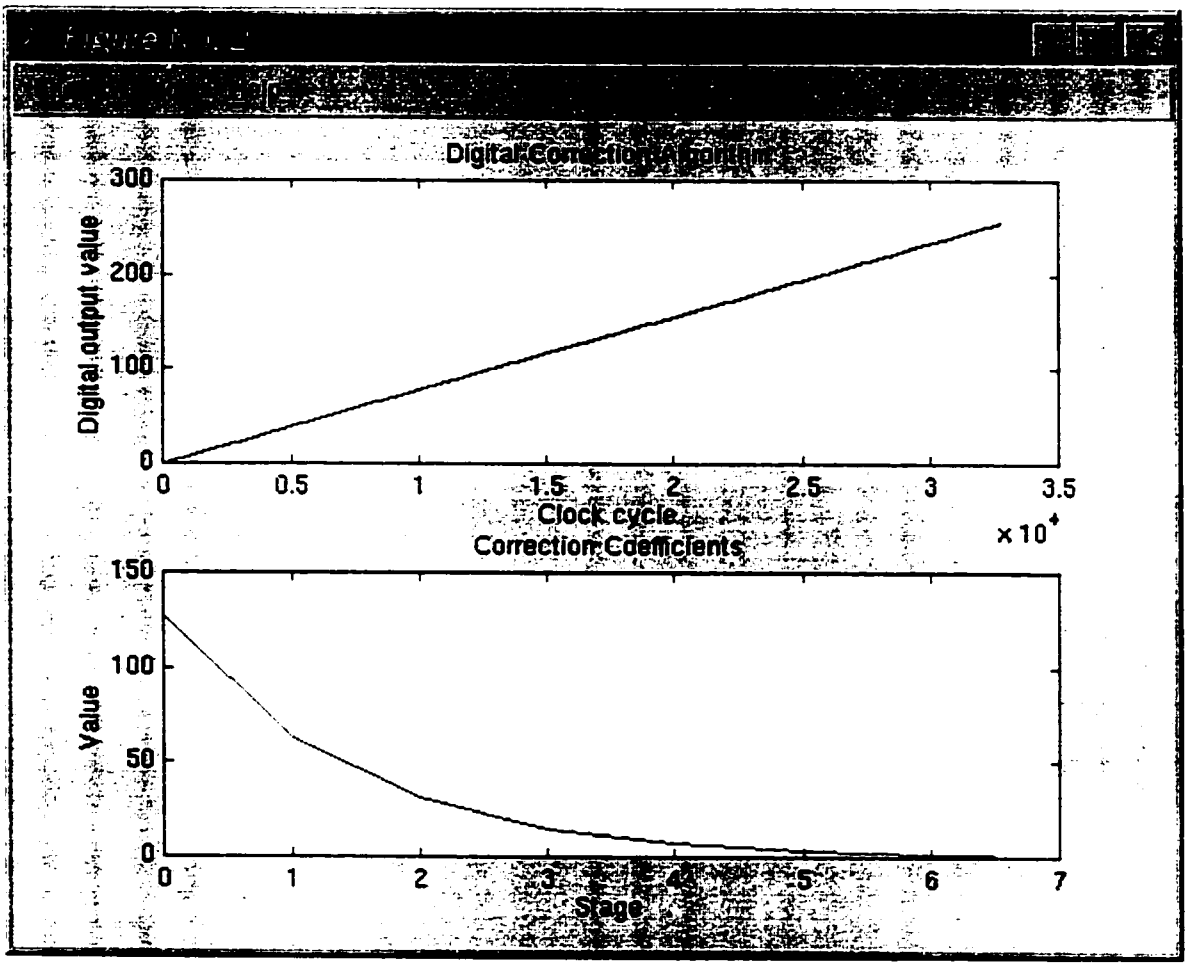
```
Digital Error Correction Algorithm              By Douglas Beck
======================================================== 11/18/98

Pipeline Simulation Parameters

  Input  filename    : input.txt
  Output filename    : output.txt
  Pipeline    stages: 8
  Calibration stages: 6
  Samples per stage  : 8
  Holdoff cycles     : 32
  Output cycles      : 32768
  Output format      : raw decimal data

Initializing Simulation
Simulating digital error correction of input waveform
Outputting Matlab Formatted Transfer Functions
Outputting Matlab Plot Code
Simulation completed successfully
```

Sample Run Explanation

In the case where the parameters or the input file is missing or incorrect, an error message will appear and the extraction process is terminated.

Finally, upon successful completion digital error calibration, an output file is created which contains the calibrated output waveform. In addition to the raw decimal output data, if the output is in Matlab format, a plot of the output data is also created. The

resulting plot also contains the digital error correction coefficients. The following Matlab plot shows a calibrated ramp waveform.



At this point, the digitally corrected data is now ready for additional processing (`fft` or `linear`).

Future Additions/Modifications to this Program

No plans for additions.

Known Problems

There are currently no known problems with this software.

HSPICE Data Extraction Software

Description

This program will directly convert the bit output waveforms from a pipelined ADC converter to digital 1's and 0's. This is an essential step in realizing and characterizing the output of a pipelined ADC converter.

Command Format and Parameters

The following is a breakdown of the command line arguments. This breakdown is automatically displayed when the program is executed without command line arguments.

```
HSPICE Pipeline Digital Data Extractor          By Douglas Beck
================================================== 12/08/98

Command format:

            data [infile] [outfile] <options>

            extraction options:

               -s      [number of output samples]
               -h      [holdoff for n cycles]
               -b      [number of bits in pipeline]
               -v      [supply voltage]

            output options:

               -m       [filename] Matlab formatted output
               -mp      [filename] Matlab plot code
```

Description of Parameters

In this case, the input file is a .TR0 output file from a transient simulation in SPICE. The output file is ascii 1's and 0's from the ADC pipeline.

The "-s" command will limit the total number of output samples. If this parameter is not set, it will default to the entire simulation file. This is useful when the SPICE files become large and unmanageable, or if you just need to look at a small portion of the actual SPICE output.

The "-h" command will delay sampling of the ADC pipelined output data until a specific number of cycles. If this parameter is not set, it will default to 8. This is useful to give the simulation time for the transients to settle out before gathering data.

The "-b" command defines the number of bits in the pipeline. If the parameter is left blank, it will default to 8.

The "-v" command defines the supply voltage of the ADC pipeline. The voltage is used to measure the digital output waveforms and determine if the output is 0 or 1. This value defaults to 1.8 volts.

The "-m" command will create an additional output file. Rather than a raw digital output datafile, this file is created in Matlab format. The output is decimal, and is ready for direct import into Matlab. The filename should end in .m so that it will be recognized by Matlab as a script.

The "-mp" command will create a plot file for Matlab. This Matlab script will format and plot the Matlab data. The filename should end in .m so that it will be recognized by Matlab as a script.

Sample Run

To illustrate the functionality of the **data** command, a sample is presented below. The extraction tool is executed with the parameters as shown:

```
data ../pipeline1.8/top.tr0 output.txt -b 8 -v 1.8
```

Reviewing the command line parameters above, we see that the extraction will take place on the file ../pipeline1.8/top.tr0 (most likely resulting from the simulation of an HSPICE file top.sp) And output file is created with the filename output.txt. It will contain the extracted digital data from the pipeline.

As the extraction executes, it displays the following information to the console:

```
HSPICE Pipeline Digital Data Extractor          By Douglas Beck
=================================================== 12/08/98

HSPICE Extraction Parameters

  Input  filename    : ../pipeline1.8/top.tr0
  Output filename    : output.txt
  Total samples      : Until end of file
  Holfoff            : No holdoff
  Bits               : 8
  Supply Voltage     : 1.8 Volts


[OPEN ] Spice file: ../pipeline1.8/top.tr0
  Waveforms: TIME, vinput, vres1, vres2, vres3, vres4, phi1,
             phi1b, phi1e, phi1eb, phi2, phi2e, phi2b, phi2eb,
             bit0, bit1, bit2, bit3, bit4, bit5, bit6, bit7,
             bit8.

  Time ticks: 51908

Verifying signals:
  TIME   exists
  phi1e  exists
  phi2e  exists
  bit0   exists
  bit1   exists
  bit2   exists
  bit3   exists
  bit4   exists
  bit5   exists
  bit6   exists
  bit7   exists

480 clocks in  4.799 us
```

```
Clock Rate=100.023 Mhz
```

Sample Run Explanation

In the case where the parameters or the input file is missing or incorrect, an error message will appear and the extraction process is terminated.

Finally, upon successful extraction of the digital data, an output file is created. A sample output is shown below:

```
10110011
11010110
10010101
11101011
...
```

The data is now ready for processing. This processing includes the process of unpipelining the data, applying an error correction algorithm (if appropriate) as well as converting the output data to decimal.

Future Additions/Modifications to this Program

This program will eventually be modified to incorporate the double sampling technique originally introduced by Motorola.

Known Problems

There are no known problems with the program.

Discrete Fourier Transform Software

Description

This program performs a Discrete Fourier Transform on a set of input data. This program was designed to take the corrected digital output data and perform a Fast Fourier Transform (FFT) on the data. From this FFT, the software will calculate the Signal to Noise Ratio (SNR), the Signal to Noise and Distortion Ratio (SNDR), the Total Harmonic Distortion (THD), and the Spurious Free Dynamic Range (SFDR) of

the ADC output data. This information is very useful when trying to characterize the performance of the ADC converter.

Command Format and Parameters

The following is a breakdown of the command line arguments. This breakdown is automatically displayed when the program is executed without command line arguments.

```
FFT Calculation and SNDR Estimation                          11/23/98
================================================================

Command format:

            fft [infile] [outfile] <options>

            options:

               -s      [number of samples]
               -m      Matlab formatted output
```

Description of Parameters

The "-s" command defines the number of output data points on the FFT.

The "-m" command formats the output data in Matlab format. The output filename should end in .m so that it will be recognized by Matlab as a script.

Sample Run

To illustrate the functionality of the **fft** command, a sample is presented below. This sample command is executed with the parameters as shown:

```
fft fft.txt fftout.m -m -s 32768
```

Reviewing the command line parameters above, we see that the fft software will take the input file fft.txt and create a Matlab formatted fftout.m file. The fft will output a 32768 point FFT plot.

As the FFT simulation executes, it displays the following information to the console:

```
FFT Calculation and SNDR Estimation                          11/23/98
================================================================

FFT Parameters

  Input  filename    : fft.txt
  Output filename    : fftout.m
  Total samples      : 32768
  Output format      : Matlab formatted decimal data

FFT Results:

  SNR  = 18.32 dB
  SNDR = 18.30 dB
  THD  = -69.60 dB
  SFDR = 21.06 dB

FFT complete
```

Sample Run Explanation

In the case where the parameters or the input file is missing or incorrect, an error message will appear and the extraction process is terminated.

Finally, upon successful FFT calculation, an output file is created which contains the FFT data. In addition to the raw fft output data, if the output is in Matlab format, a plot of the output data is also created.

Future Additions/Modifications to this Program

No plans for additions.

Known Problems

There are currently no known problems with this software

Linearity Characterization Software

Description

Another important performance measure of ADC's is the Integral Non-Linearity (INL) and the Differential Non-Linearity (DNL) of the ADC output data. This program will divide the ADC output into specific output bins. Once these bins are filled will all the available output data, the INL and DNL performance measures are calculated.

Command Format and Parameters

The following is a breakdown of the command line arguments. This breakdown is automatically displayed when the program is executed without command line arguments.

```
Nonlinearity Calculation                                        12/09/98
================================================================================

Command format:

     linear [infile] [outfile] <options>

     options:

       -m      Matlab formatted output
       -plot   [filename] transfer function plot code
       -b      [n]  n bit A/D converter

     input waveform options:

       -tri    triangle input waveform
       -sine   sinusoidal input waveform
```

Description of Parameters

The "-m" command formats the output data in Matlab format. The output filename should end in .m so that it will be recognized by Matlab as a script.

The "-plot" command will create a plot file for Matlab. This script with format and plot the Matlab data. The filename should end in .m so that it will be recognized by Matlab as a script.

The "-b" command defines the number of bits in the pipeline. If the parameter is left blank, it will default to 8.

The "-tri" command selects a triangular input waveform for output following the calibration waveform. This software needs to know the wave shape of the incoming waveform in order to accurately calculate the bin probability of a digitized input waveform.

The "-sine" command selects a sinusoidal input waveform for output following the calibration waveform. This software needs to know the wave shape of the incoming waveform in order to accurately calculate the bin probability of a digitized input waveform.

Sample Run

To illustrate the functionality of the **linear** command, a sample is presented below. This sample command is executed with the parameters as shown:

```
linear input.txt output.m -m -plot plot.m -b 8 -tri
```

Reviewing the command line parameters above, we see that this software will take the input file input.txt and create a Matlab formatted output.m file. In addition, this software will create the Matlab plot code in order to format the output data. Finally the input is an 8 bit triagular waveform.

As the software executes, it displays the following information to the console:

```
Nonlinearity Calculation                                    12/09/98
================================================================

INL-DNL Parameters

  Input  filename    : input.txt
  Output filename    : output.m
  Converter res      : 8 bits
```

```
  Output bins        : 256
  Output format      : Matlab formatted decimal data
  Input              : triangle waveform

Opening data streams
Filling bins
Total samples read 32768
Calculating nonlinearity
Outputing Matlab data file
Outputting Matlab Plot Code
Nonlinearity calculation complete
```

Sample Run Explanation

In the case where the parameters or the input file is missing or incorrect, an error message will appear and the extraction process is terminated.

Finally, upon successful calculation, an output file is created which contains the INL and DNL data. In addition to the raw bit output data, if the output is in Matlab format, a plot of the output data is also created. The resulting Matlab plot appears below.

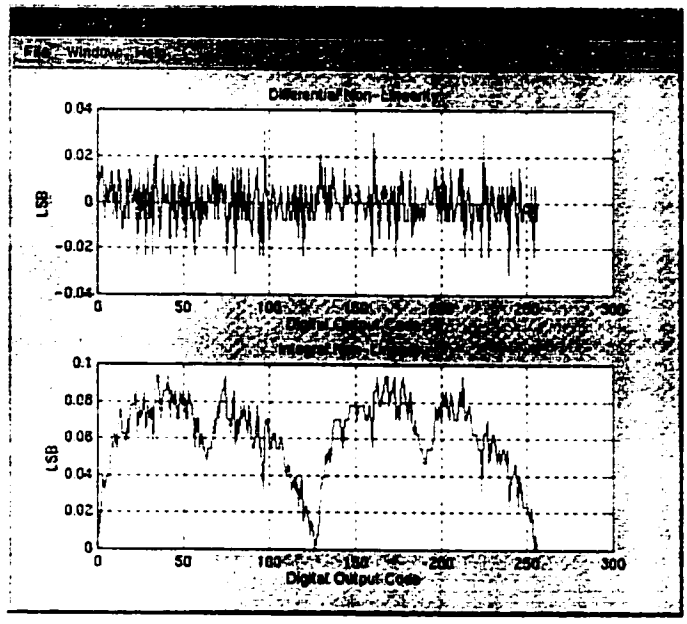


Fig. 66 INL/DNL Calculation.

Future Additions/Modifications to this Program

A needed addition to this software is the ability to adjust the magnitude of the incoming sinusoidal waveform. I plan to add a parameter "-a" where the input magnitude is expressed as a fraction of full dynamic range.

Known Problems

There are currently no known problems with this software

HSPICE Waveform Extraction Software

Description

This program will extract a single waveform from a SPICE .TR0 file. This is useful for any number of applications. A Sample application would be to extract a waveform and import it into Matlab to perform a FFT on the waveform.

Command Format and Parameters

The following is a breakdown of the command line arguments. This breakdown is automatically displayed when the program is executed without command line arguments.

```
HSPICE Waveform Extraction Tool                    By Douglas Beck
================================================== 02/19/99

Command format:

            wave [infile] <options>

            extraction options:

               -s      [number of output samples]
               -sig    [signalname]
               -step   [take every nth sample]
               -h      [holdoff for n cycles]

            output options:

               -m      Matlab formatted output
               -mp     [filename] Matlab plot code
```

Description of Parameters

In this case, the input file is a .TR0 output file from a transient simulation in SPICE. The output file is is a series of values output in scientific notation with 5 significant digits.

The "-s" command will limit the number of output samples to take before completing the extraction process. If this parameter is not set, it will default to the entire simulation file. This is useful when the SPICE files become large and unmanageable, or if you just need to look at a small portion of the actual SPICE output.

The "-sig" command defines what signal to extract from the SPICE file.

The "-h" command will delay sampling of the signal until a specific number of cycles. If this parameter is not set, it will default to 0. This is useful to give the simulation time for the transients to settle out before gathering data.

The "-step" command defines the number of samples to skip for each sample that is output. If the parameter is left blank, it will default to 1 (meaning don't skip any data). This is useful when the SPICE files become large and unmanageable, using this approach will yield the entire waveform at reduced temporal resolution.

The "-m" command will format the output data in Matlab format. The output is a series of floating point numbers and is ready for direct import into Matlab. The filename should end in .m so that it will be recognized by Matlab as a script.

The "-mp" command will create a plot file for Matlab. This script will format and plot the Matlab data. The filename should end in .m so that it will be recognized by Matlab as a script.

Sample Run

To illustrate the functionality of the **wave** command, a sample is presented below. This sample command is executed after a successful SPICE simulation. The extraction tool is executed with the parameters as shown:

```
wave ../pipeline1.8/top.tr0 -sig phi1
```

Reviewing the command line parameters above, we see that the extraction will take place on the file ../pipeline1.8/top.tr0 (most likely resulting from the simulation of an HSPICE file top.sp) And output file is created with the filename phi1.txt. It will

contain the value of the requested output signal at each time tick. Also, please keep in mind that the HSPICE simulation file must include the following:

```
.options post=2
.options probe
```

As the extraction executes, it displays the following information to the console:

```
HSPICE Waveform Extraction Tool                    By Douglas Beck
=================================================== 02/19/99

HSPICE Extraction Parameters

  Input  filename    : ../pipeline1.8/top.tr0
  Output filename    : phi1.txt
  Total samples      : Until end of file
  Holfoff            : No holdoff

[OPEN ] Spice file: ../pipeline1.8/top.tr0
  Waveforms: TIME, vinput, vres1, vres2, vres3, vres4, phi1,
             phi1b, phi1e, phi1eb, phi2, phi2e, phi2b, phi2eb,
             bit0, bit1, bit2, bit3, bit4, bit5, bit6, bit7,
             bit8.

  Time ticks: 51908

Verifying signals:
  phi1   exists
```

Sample Run Explanation

In the case where the parameters or the input file is missing or incorrect, an error message will appear and the extraction process is terminated.

Finally, upon successful extraction of the waveform data, an output file is created. A sample output is shown below:

```
1.800000e+00
1.799500e+00
1.799700e+00
```

```
1.799800e+00
1.799900e+00
1.799900e+00
...
```

The data is now ready for processing or plotting by another application or program.

Future Additions/Modifications to this Program

No plans for additions.

Known Problems

Signal names must be less than 16 characters. If not, the program will output incorrect data.

APPENDIX B: Die Micrograph

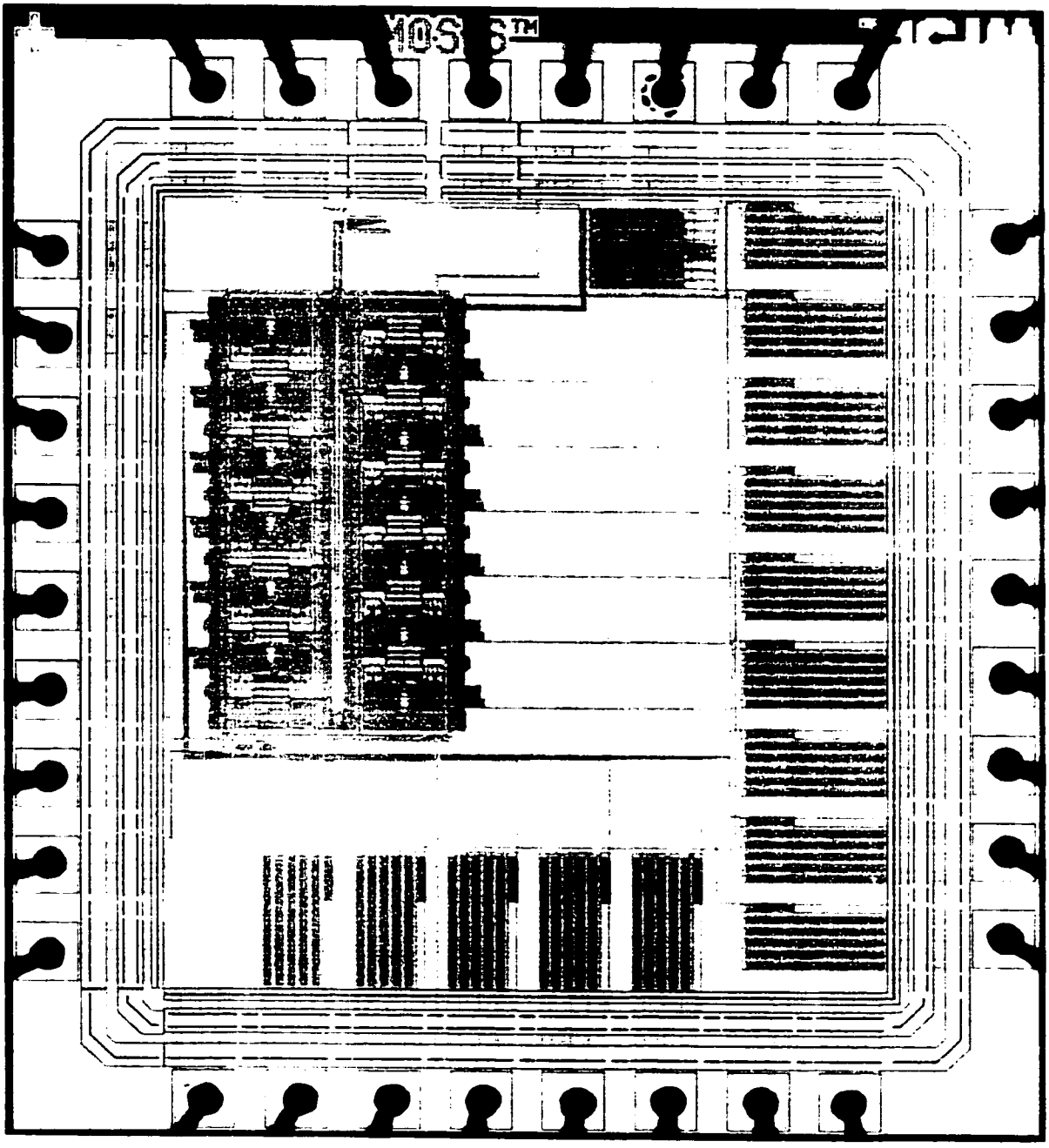


Fig. 67 Die Micrograph.

APPENDIX C: Test Board Schematic

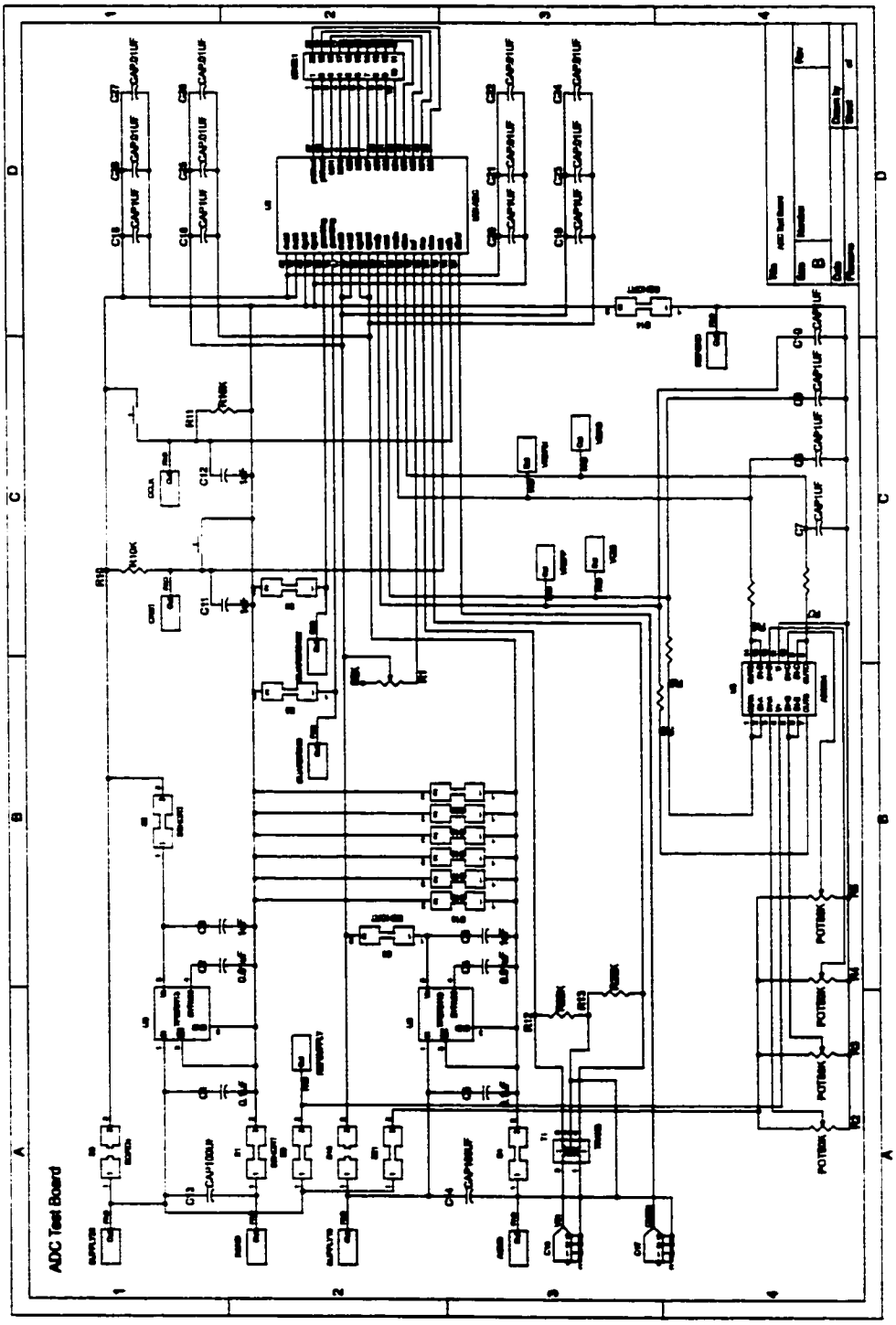


Fig. 68 Test Board Schematic.

Vita

Douglas Beck: After participating in programs at Oregon State University, Stanford University, Arizona State University, and University of Washington, Mr. Beck has lived many places throughout his academic career. Awarded the Bachelor of Science and Master of Science Degrees in Electrical Engineering from the Oregon State University in 1995 and 1997 respectively. In 2002 he earned a Doctor of Philosophy at the University of Washington in Electrical Engineering.