A Delay Line Architecture for High-Speed Analog-to-Digital Converters

by
Ronald A. Kapusta Jr.

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Electrical Engineering and Computer Science

at the
Massachusetts Institute of Technology
May 21, 2002
[June 2002]

Author___________
Department of Electrical Engineering and Computer Science
May 21, 2002

Certified by_
Michael Anthony
M.I.T. Lincoln Laboratory Thesis Supervisor

Certified by__
Charles G. Sodini
M.I.T. Thesis Supervisor

Accepted by_________
Arthur C. Smith
Chairman, Department Committee on Graduate Theses

A DELAY LINE ARCHITECTURE FOR HIGH-SPEED ANALOG-TO-DIGITAL CONVERTERS

by Ronald A. Kapusta Jr.

Submitted to the

Department of Electrical Engineering and Computer Science

May 21, 2002

In Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Electrical Engineering and Computer Science

ABSTRACT

The delay line analog-to-digital converter architecture is presented as an alternative to conventional high-speed analog-to-digital converters. The delay line converter is a variation on the conventional sub-ranging architecture. The emphasis in the development of the delay line architecture is a reduced sensitivity to aperture jitter, particularly in the sample-and-hold functional blocks. The burden of jitter tolerance is shifted from the sample-and-hold circuits to a digital-to-analog converter. With the use of clever current shaping techniques, the noise power contributed by clock jitter in the digital-to-analog converter is reduced. Also, due to the inclusion of a reconstruction filter in the delay line architecture, a calibration procedure is necessary. Component level simulations are performed that verify the functionality of the delay line architecture and the calibration algorithm. A discrete implementation of the delay line architecture is also realized in order to validate the concepts developed. This experimental converter does not exhibit the desired performance, less than 6 effective bits are achieved with an 11-bit, 250MSPS converter. However, the performance demonstrated is sufficient to verify functionality of the delay line converter architecture.

M.I.T. Lincoln Laboratory Thesis Supervisor: Michael Anthony

M.I.T. Thesis Faculty Supervisor: Professor Charles G. Sodini

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

INTRODUCTION

Today's wireless communications market is rapidly evolving and soon one base station will need to be able to communicate via multiple air standards to other wireless devices. The software-defined radio (SDR) is an ideal solution for this problem, allowing base station designers to program different standards in the field rather than developing separate receivers for each standard. To this end, the wireless market is focused on moving as many components as possible from the analog domain into the digital domain to reduce cost per channel, size and power; improve reliability; and increase flexibility of the end product. To achieve these goals, the incoming signal at radio frequencies (RF) must be digitized. However, existing technology is currently incapable of digitizing RF signals. Another, more practical, method is to mix the incoming signal from radio frequencies to an intermediate frequency (IF), which may range between 455 kHz and 250 MHz. The advantages of IF sampling are the elimination of a downconversion stage (IF to base band) and its associated components, as well as reduced filter costs. Advanced analog-to-digital converters (ADCs) are capable of IF sampling and digitization. However, the performance requirements for the ADC must now take on the entire burden of the dynamic range that was once spread across many more components. [1]

In addition to the evolution of the wireless communication industry, the demand for higher speed home Internet access via the cable network is also developing very quickly. Typically, a cable modem sends and receives data in two slightly different fashions. In the downstream direction, the digital data is modulated and then placed on a typical 6 MHz television channel, somewhere between 50 MHz and 750 MHz. In the upstream direction, multicable system operators (MSOs) are increasing the speed and capacity of their existing plants by moving to a "digital return" architecture in the cable plant. This digital return path accommodates higher order modulation schemes for higher data rates. Digital return path requires high-speed data converters that achieve superior signal-to-noise ratio (SNR) performance over a band of frequencies up to 42 MHz in North America and 65 MHz in Europe, at very high sampling rates. As the applications mentioned above, as well as numerous other radar, satellite, and communications test equipment applications, continue to develop, ADCs with faster conversion rates and higher resolution are needed.

Many different ADC architectures have been developed. The fastest and simplest architecture is known as a flash ADC. The speed of the flash architecture is a result of the conversion scheme; specifically, all bit level decisions are made in parallel. Therefore, the conversion rate of the flash converter is only limited by the decision time of a single comparator. However, the power consumed in a flash ADC grows exponentially with the number of bits resolved. Even though a pure flash ADC is capable of achieving conversion rates greater than 1GSPS, power and size limitations restrict it to low or moderate resolution applications (8 bits or less). [2]

Another ADC architecture that improves on the power and size limitation of the flash architecture is the sub-ranging converter. Sub-ranging ADCs are capable of achieving 12-bit accuracy, or even more, but they are more complex to design than flash ADCs. This increased complexity, specifically the addition of more sample-and-hold blocks (S/Hs), can be problematic at very high speeds. As a result, a 10-bit sub-ranging converter might not be realizable at speeds above a few hundred megasamples per second. [2]

As an alternative high-speed architecture, a variation on the sub-ranging architecture is proposed in this thesis. While this architecture introduces even more complexity than in a conventional sub-ranging architecture, the added complexity is aimed at reducing the performance requirements of the S/H circuit blocks. Assuming the performance of the sample-and-hold blocks is the limiting factor in the achievable speed of a sub-ranging

ADC, this new architecture should be able to achieve the high accuracy of a conventional sub-ranging ADC as well as an improved conversion rate.

The performance improvement described above hinges on one critical observation, specifically that the performance of high-speed ADCs is limited by the accuracy of sample-and-hold amplifiers. In fact, it has been suggested that aperture jitter, or sample-to-sample clock timing errors, is the limiting factor in ADC performance for conversion rates of about 2MSPS through 4GSPS. [3] If the timing from one sample to the next is not accurate, the ADC will convert an inaccurate captured value. Therefore, the overall system accuracy depends on both the accuracy of the S/H and the ADC itself. At high input frequencies, the error introduced by aperture jitter increases. For a given aperture jitter, doubling the signal frequency will reduce the accuracy of a S/H by a factor of two. Therefore, the achievable resolution of a high-speed ADC system is limited by the error introduced by aperture jitter in the S/H blocks.

In order to relax the S/H performance requirements, the proposed architecture adds a delay line and a low pass reconstruction filter to the sub-ranging architecture. The basic functionality of the delay line is to replace the input S/H block. The low pass filter is added to reduce the high frequency components of the input signal to a second S/H block. By reducing the high frequency components of the signal, effectively slowing it down, a given aperture jitter will cause less sampling error. A much more detailed description of the delay line ADC is given in Chapter 3.1.

There are, of course, disadvantages to the added complexity of the delay line ADC. In any system, additional components generally increase the difficulty of design and system control. For instance, added components can complicate clock timing or cause a feedback system to become unstable. However, for the delay line ADC, the most significant complication of the added components is an effect of the low-pass filter. The filter smears the estimation signal in the time domain, an effect that is described in depth in Chapter 3. This smearing effect necessitates an estimation calibration scheme, the complexity of which is determined by the design of the filter. As a result, the filter design becomes a critical issue in the performance of the delay line ADC. This issue is further discussed in Chapter 4.1.

As mentioned earlier in this chapter, the purpose of the added blocks is to reduce the performance requirement of the sample-and-hold circuits. While this thesis will argue that the error introduced by jitter in the S/H blocks is indeed reduced, aperture jitter in clocking the DAC could still produce significant errors. Unlike S/H clock noise, though, clock noise in a DAC may be shaped using clever clocking techniques. In [4], such a clocking scheme is proposed in which the DC current sources in a DAC are replaced by shaped current pulses. By switching the DAC transitions at the minima of the shaped current waveform, the sensitivity to aperture jitter is reduced. Therefore, aperture jitter should not limit the performance of the delay line ADC.

Chapter 2

BACKGROUND

2.1 Conventional Architectures

As mentioned in the introduction, the delay line ADC architecture is intended for high-speed applications. The goal is to achieve higher resolutions than other high-speed architectures are capable of while, at the same time, maintaining a high sampling rate. Therefore, the focus of this chapter is entirely on conventional high-speed A/D converters. There are many other slower ADC architectures; however, they are not discussed in this thesis.

2.1.1 Flash A/D Converters

Flash analog-to-digital converters, also known as parallel ADCs, are the fastest ADC architecture. However, because the flash architecture requires a number of comparators which grows exponentially with the number of bits resolved, a flash ADC consumes a lot of power, presents a rather large capacitive load, occupies a large chip area, and can be expensive. This generally limits them to high-speed, low or medium-resolution applications that cannot be solved with another approach.

A typical flash converter is shown in Figure 2.1. The flash architecture consists of 2^N-1 comparators (where N is the number of bits resolved) each making a bit level decision in parallel. Each comparator, shown as a "C" block in Figure 2.1, is typically a cascade of several low gain stages followed by a latch stage. The latch stage employs positive feedback in order to drive the output of the comparator to a final decision, either a digital "0" or a digital "1". The comparators need to make a decision quickly, and this is the

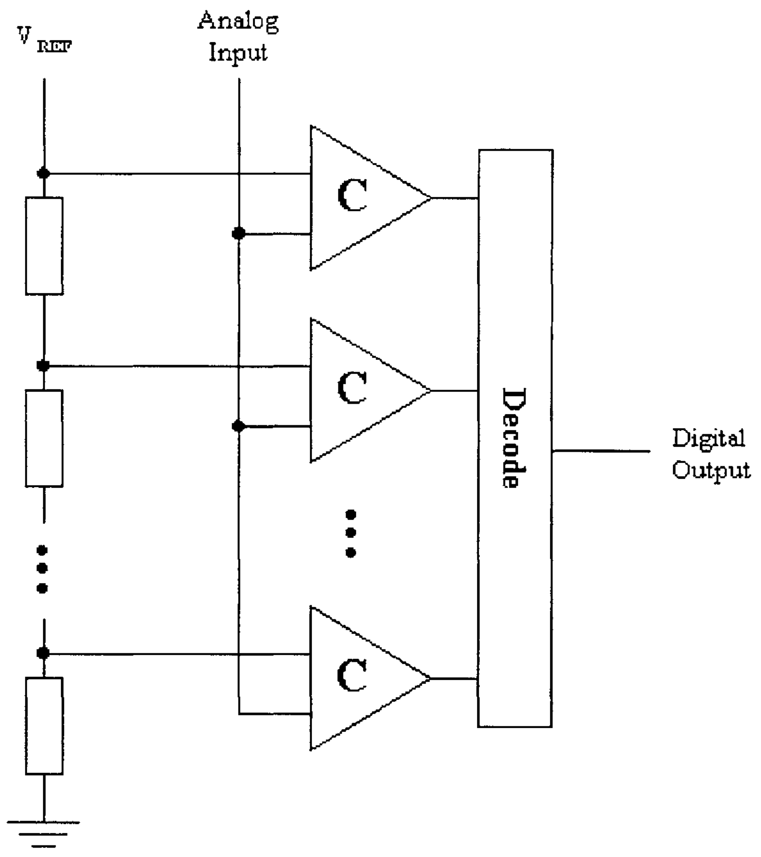


Figure 2.1 Flash A/D converter architecture. $2^N - 1$ comparators required for N-bit conversion.

reason for a multi-stage comparator design. Due to an inherent gain-bandwidth product tradeoff in voltage mode amplifier design, a single high gain and wide bandwidth stage for the comparator amplifying stage is impractical. By distributing the gain over several stages, the gain requirement can be met while preserving a wide bandwidth.

Because the analog input is compared against each reference level at the same time, the conversion time required is primarily limited by the settling of a single comparator. Therefore, the conversion time does not increase as resolution is increased. This is the major advantage of flash converters over other converters, for which the conversion time increases linearly, or even geometrically, with an increase in resolution.

The biggest drawback of the flash ADC is the exponential growth of the number of comparators with increasing resolution. As mentioned earlier, the conversion time of a flash ADC is determined by the decision time of the comparator cell. Therefore, a wide bandwidth comparator is advantageous. Wide bandwidth comparators require several gain stages, each with non-trivial power dissipation. Combining the exponentially growing number of comparators with the power requirement in order to implement wide bandwidth amplifiers, a flash ADC entails very high power consumption. In addition to the high power consumption, every bit increase in flash ADC resolution almost doubles the core die area. Increasing die size increases the cost of the converter.

The flash A/D converter suffers from other severe disadvantages in addition to the large area and power dissipation, such as high nonlinear input capacitance and signal distribution problems. [5] Given the large number of comparators at the input, a flash ADC presents a large capacitive load that can be difficult to drive. For all of these reasons, the flash A/D architecture is limited to applications that require high-speed decisions but only moderate resolutions.

2.1.2 Sub-ranging A/D Converters

An A/D architecture that improves upon the size and power limitations of the flash architecture is the sub-ranging converter, also known as the multi-step, semi-flash, or dual-rank converter. [2] Sub-ranging ADCs borrow ideas from both the flash converter as well as a pipelined converter. A typical sub-ranging converter, shown in Figure 2.2, uses a coarse ADC, which is often implemented with a flash architecture, to generate the M most significant bits. This coarse digital estimate is then converted back into an analog estimate and subtracted from the input, leaving a residue signal. A fine N–bit A/D converter, again typically a flash converter, then converts the residue signal. The coarse bits and the fine bits are combined to yield M+N bit digital representation of the input.

Sub-ranging ADCs improve on the power and size limitations of flash ADCs by dividing the number of bits to be resolved into smaller groups. Consider an 8-bit sub-ranging ADC with a 4-bit estimation stage and a 4-bit second stage. If the converters in both stages were implemented as flash ADCs, the number of comparators required would be

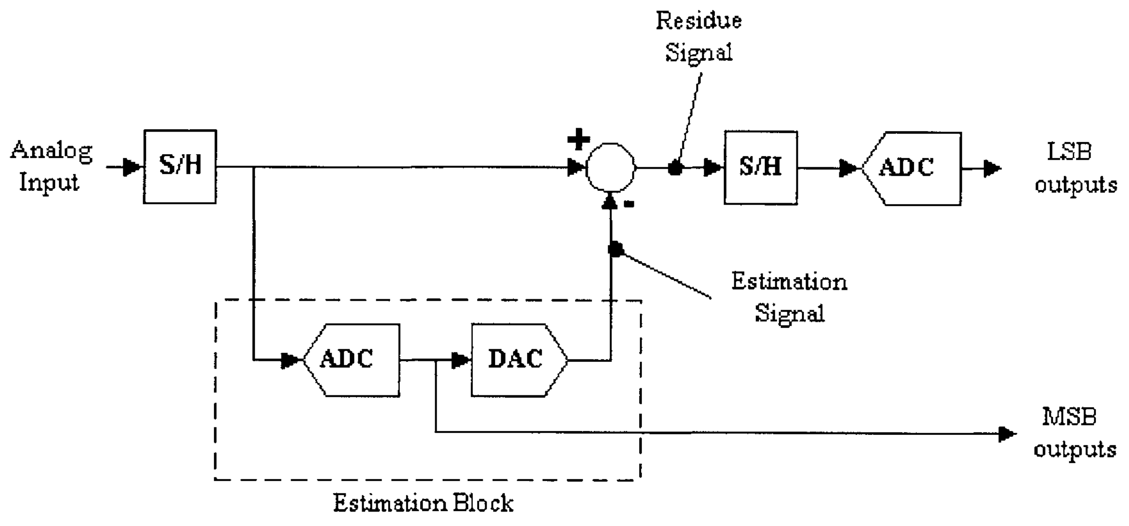


Figure 2.2 Typical two-stage sub-ranging ADC architecture.

$2*2^4$, or 32. However, if an 8-bit flash converter were implemented, 2^8 or 256 comparators would be required. Therefore, the sub-ranging architecture is suitable for applications that require speed similar to a flash ADC but can afford less power and die area.

The advantages of a sub-ranging ADC are accompanied by several disadvantages. First, flash ADCs only require two basic building blocks, the comparator and the sample-and-hold circuit (S/H). Sub-ranging ADCs, however, require three additional building blocks, another S/H block, a digital-to-analog converter (DAC), and a subtractor. Not only are additional building blocks required, but also both the S/H and the DAC must be high performance circuits.

The key circuit in the sub-ranging ADC is the input S/H; its performance significantly influences nearly all performance metrics, including the dynamic parameters such as SNR and SFDR. [5] The S/H block in the second stage, which would also be included in a pure flash ADC, must capture a sample accurate to the resolution of the following ADC. The first stage S/H, however, must be capture a sample that is equally or more accurate than the resolution of the entire converter. In the above example, the first stage S/H would be at least as accurate as 1 part in 256, while the second stage S/H would only need to be accurate to 1 part in 16. Clearly, the additional S/H block must meet performance requirements far stricter than the S/H required by the second stage ADC.

Similarly, the estimation DAC must meet or exceed the accuracy of the entire system. In the earlier example, an 8-bit accurate DAC would be required even though the input to the DAC would only be a 4-bit digital code. If either the DAC or the first stage S/H were not accurate to 8 bits, the residue signal would not be accurate to 8 bits and an accurate 8-bit A/D conversion would not be possible.

Another disadvantage of a sub-ranging ADC is the improvement in resolution at the cost of speed or latency. It is possible to pipeline the two stages in a sub-ranging ADC so that the overall conversion speed is not severely compromised. Note, however, that the conversion speed is somewhat decreased to do the fact that the DAC output must settle and the resulting residue signal must be captured by the second stage S/H before another input sample may be taken. If this speed reduction is too great, the ADC could be further pipelined by clocking the DAC as another pipeline stage. Of course, each of these pipeline stages increases the latency from input to output. In many applications, latency is not a serious problem, especially with a 500MSPS converter where a full cycle of pipelining only adds 2ns of latency. Still, a tradeoff exists in sub-ranging architectures between the sampling rate and the latency, and this tradeoff is not a problem with flash converters.

2.2 ADC Performance Limitations

2.2.1 Static Considerations

Several static parameters limit both the achievable resolution and conversion speed of A/D converters. One of these parameters, component mismatch, is dependent on the process used fabricate the ADC as well as the design of the components. Consider, for example, a simple 2-bit flash ADC. In order to set up the reference voltages to which the input compared, a ladder of 2^2 R valued resistors could be used. The input voltage is compared to each of these 4 reference voltages with a comparator. If the resistor values were not well matched, the reference voltages would not be exact, and the resulting digitization would be non-linear. Likewise, if the comparator offsets were not matched, the digitization would behave just as if the reference voltages were not exact. Since this example ADC requires 2-bit accuracy, the resistors and comparators only need to match to 1 part in 4. For a 10-bit ADC, however, the accuracy required is better than 1 part in 1000. Thus, the resistors in the ladder would need to be 0.1% accurate, and the comparator offsets would need to match to within 1mV for a 1V full-scale input. Thus, the resolution of an ADC is limited by the precision to which components can be matched.

Power dissipation is another parameter that limits ADC performance. It is shown in Chapter 2 that the conversion speed of a flash converter is limited by the settling time of a single comparator. Therefore, implementing a 1GSPS converter would require a comparator that is capable of settling in less than 1ns. Also, as described earlier, a comparator consists of several gain stages followed by a decision making latch. Thus, the settling time of the comparator is determined by the time required for the gain stages to amplify the input signal enough to trigger the latch in addition to the settling time of the latch. If the gain portion of the comparator can be approximated as a single pole system, the rise time of the input to the latch is an exponential decay towards the final value, and the time constant of the decay is determined by the comparator bandwidth. Decreasing the settling time of the comparator would require increasing the comparator bandwidth. This increase in bandwidth can be achieved by several means, including clever circuit design. However, given a circuit topology, the bandwidth is proportional to the transconductance of the devices. Likewise, the time constant of a latch mode circuit is inversely proportional to the transconductance. [6] Since transconductance is proportional to current, dissipating more power will result in faster comparators. Therefore, the speed of a flash converter in a given process technology is limited by the power that can be spent in the comparators.

Other static parameters also limit the performance of A/D converters, such as die area. However, these problems can all be solved by laser trimming for accuracy, increasing power consumption, spending more money on a larger chip, and so on. Such problems and solutions are not the primary concern of this thesis. Instead, this paper will focus on the performance considerations that cannot be solved with more expensive fabrication processes and higher power consumption.

2.2.2 Dynamic Considerations

Dynamic performance limitations are much more complex and harder to solve than static limitations. First, consider the dynamic problems in a simple flash converter. Unless the comparators present a varying load, the resistor ladder should maintain a stable set of reference voltages. However, comparator performance is much less dynamically stable. First, comparators take some finite time to make a decision. Until the digital output of the comparator has settled to either a "0" or a "1", the comparator is in a metastable state. In order to avoid metastable states, the comparator needs to settle more quickly. As argued in the last section, this can be achieved by consuming more power. However, there is a fundamental limit to how fast a comparator can be made, which is related to the speed of the device technology used to implement the converter. [3] Given that current IC processes can produce transistors with f_T in excess of tens of gigahertz, comparator metastability only limits ADC performance at very high speeds.

Another dynamic performance limiting factor is noise in the ADC. There are many noise sources in a comparator circuit, including thermal noise in resistors and thermal, shot and flicker noise in transistors. All of these noise sources can be referred to the input of the circuit, and if this input referred noise is of the same order of magnitude as an LSB, the ADC accuracy will suffer. As is the case with comparator settling time, increasing power consumption can solve problems due to noise sources in the ADC. If the thermal noise in the resistor ladder is a problem, the resistor values can be decreased, lowering the noise but increasing the current through the ladder. If transistor noise were the dominant source, increasing the current through the transistors would help because the noise power in a transistor is inversely proportional to the transconductance.

2.2.3 Sub-ranging Considerations

Both of the dynamic issues that have been considered so far were problems in the basic components of the flash ADC. There are additional problems in a more complex architecture, such as the sub-ranging ADC. A sub-ranging ADC adds several blocks, each of which can limit performance. Assuming that the subtraction block is implemented well and does not contribute any dynamics to the system, which is a difficult but achievable design task, the dynamics of the DAC and second S/H block must be considered. First consider the non-idealities in the DAC. Of course a DAC has several static error sources. These sources include the finite output impedance of the current sources, assuming a current steering DAC is implemented. In wideband current-steering DACs, the number of current sources connected to the output is signal-dependent and hence these types of errors will introduce distortion. [7] In addition to the finite output impedance issue, the performance is highly dependent on the matching performance the current sources. Both systematic effects, caused by gradients over the die, and random mismatch introduce errors. Statistical analysis of DAC design can minimize mismatch induced static errors. [8, 9] There are also several dynamic effects that result in poorer performance, such as non-linear slewing, which arises from signal-dependent output capacitance, glitches, time skew, etc.

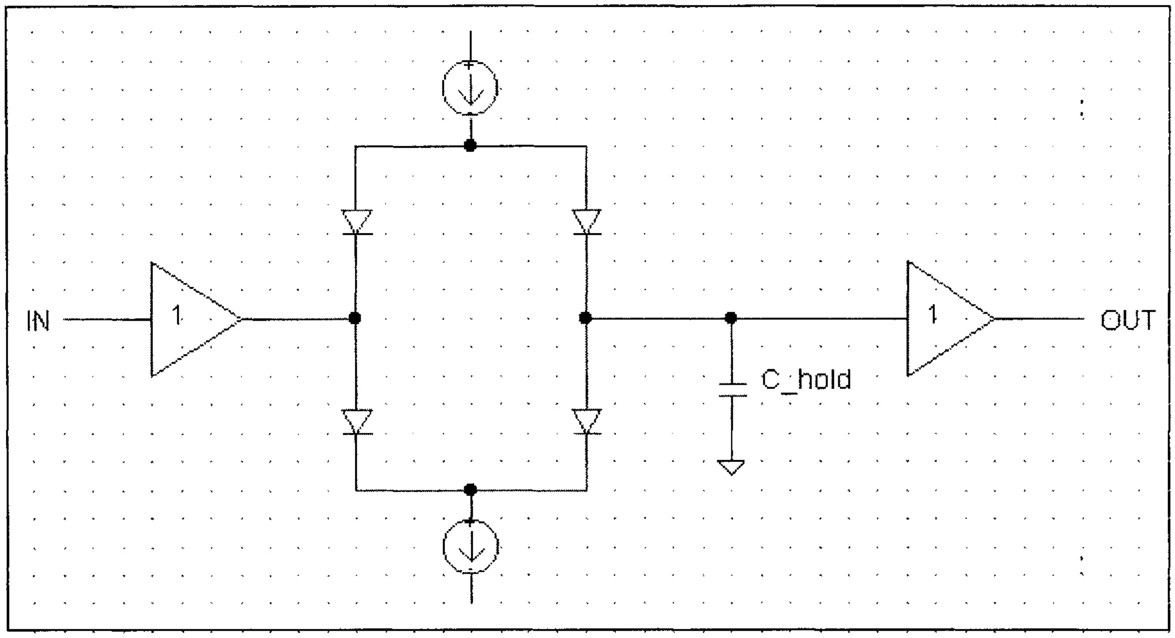


Figure 2.3 Typical sample-and-hold topology with 4-diode bridge analog switch.

The other additional block in a sub-ranging ADC is the second S/H circuit. In Chapter 2 it is argued that this input S/H amplifier must meet far stricter performance requirements than the second stage S/H. In order to understand the performance limitations of a S/H amplifier, a particular S/H architecture must be examined. A S/H consists of an input buffer, an analog switch, a capacitor on which the sampled voltage is held, and an output buffer. This thesis is concerned with high-speed A/D applications; therefore, a typical high-speed S/H topology is examined. Figure 2.3 shows a simple sample-and-hold topology with a high-speed analog switch topology, known as a 4-diode bridge. The 4-diode bridge is the most suitable implementation of an analog switch in high-speed bipolar S/H applications. [10]

The operation of the 4-diode bridge is rather simple. While both of the current sources are conducting, all of the diodes are forward biased and the bridge is in sample mode. Any changes at the input to the bridge are tracked on the hold capacitor. In order to transition to hold mode, the current sources shut off and all of the diodes stop conducting current, effectively isolating the input to the bridge from the hold capacitor. In order to shut off the current sources, the voltage levels at the top and bottom of the bridge are often switched. This level change saturates the current sources and also reverse biases the diodes. The buffer amplifiers simply isolate the bridge from the input and output loads.

There are several sources of distortion in the S/H, and this distortion causes vital information about the sampled signal to be irretrievably lost. One such error is the hold pedestal introduced during the sample-to-hold transition. As mentioned earlier, when in track mode, all of the diodes are forward biased, and the diodes are then reversed biased to transition to hold mode. Associated with each of the diodes is some non-linear junction capacitance that stores charge. As these charges are expelled when the diodes are reverse-biased, any difference in the charge appears as a voltage perturbation on the hold capacitor. Since the charge expelled is dependent on the common-mode voltage on the hold capacitor, the pedestal varies nonlinearly with the input sample and causes distortion. This pedestal is a design parameter that can be controlled by appropriately choosing the size of the hold capacitor. [10]

The diodes also introduce error during the hold mode. Any real diode has some finite leakage current while reversed biased. This leakage charge is pulled off of the hold capacitor, and the result is a droop in the voltage on the hold capacitor during the hold mode period. Any input current to the second buffer amplifier is also drawn off of the hold capacitor and adds to the droop error. As with pedestal error, choosing a large hold capacitor can reduce droop error.

The current sources can also introduce distortion into the captured voltage. If the current sources are not matched, the diodes will be forward biased slightly differently. This introduces an error on the hold capacitor proportional to the incremental on resistance of the diode. A similar analysis shows that distortion is introduced if the two current sources do not turn off at exactly the same time. Both of these errors can be minimized by clever current source design.

In addition to the various sources of distortion identified above, the held sample will be distorted with noise generated by each of the components surrounding the hold capacitor.

There are several significant noise sources in the S/H shown in Figure 2.3. First, wideband thermal noise from the input buffer amplifier is integrated on the hold capacitor. Likewise, wideband thermal noise from the output buffer stage is directly added into the captured sample. Also, shot noise in the base current of the output amplifier integrates on the hold capacitor. Of course, this shot noise error would not exist in a CMOS buffer implementation because MOS devices do not have any gate current. [10]

One final source of distortion is aperture jitter in the clock signal that causes the transition from sample mode to hold mode. This effect comes about because the clock transition does not occur at precisely equal time intervals. Instead, the sampling process can be characterized by an average time interval, $T = 1 / f_{samp}$, and a standard deviation of sampling instants around that average. This standard deviation is defined as the rms aperture jitter, τ_a. If the input to the S/H amplifier is a Nyquist rate sinusoid ($f_{in} = f_{samp}/2$), with peak-to-peak amplitude V_{FS}, the maximum error will occur at the zero crossing of the input [3]

$$v_{rms} = \frac{\pi f_{samp} V_{FS} \tau_a}{2} \tag{2.1}$$

If the rms aperture jitter is 1ps, a typical value for modern low jitter clocks, a $1V_{P\text{-}P}$, 500 MHz sinusoid will be sampled with a maximum rms voltage error of 1.6mV. This voltage error corresponds to a ½ LSB error for an 8-bit converter. The aperture jitter effect must be averaged over a (typically sinusoidal) signal to obtain rms noise. When this is done, the general aperture-jitter-limited resolution of a S/H is found to be [11]

$$N \leq \log_2\left(\frac{1}{\sqrt{6}\pi f_{input} \tau_a}\right) \tag{2.2}$$

Having identified many sources of error in S/H amplifiers, D/A converters, and comparators, the issue becomes the identification of which error sources limit overall performance. In order to explore this issue, it is important to note that the majority of the distortion sources can be compensated for through circuit design techniques. For example, increasing the size of the hold capacitor can reduce leakage current errors in a S/H, and consuming more power can speed up slow comparator settling. Solutions for aperture jitter errors are less obvious. The spectral purity of a clock is largely dependent on the crystal oscillator used to generate the clock signal. Given finite accuracy in the clock generation circuitry, it becomes a difficult task to develop circuit techniques to reduce clock jitter. As such, it is expected that clock jitter will determine the limit to which the performance of a given ADC can be pushed.

In support of this hypothesis, a survey of analog-to-digital converters, both commercial and experimental, is conducted in [3]. The analysis in this paper considers three dominant mechanisms through which A/D performance is limited: thermal noise, aperture jitter, and comparator metastability. Also, the performance metric focused on is the maximum resolution that can be achieved at a given sampling rate. The findings suggest that at sampling rates below 2MSPS, resolution is limited by thermal noise. For converters with sampling rates between 2MSPS and 4GSPS, resolution falls off by ~1 bit for every doubling of the sampling rate. This behavior is attributed to errors from aperture jitter. Finally, for converters that operate above 4GSPS, comparator metastability limits performance. This thesis is focused on high-speed, high-resolution A/D converters for the applications mentioned in Chapter 1, such as software radio. The range of conversion speeds for such applications is between ~100MSPS and ~1GSPS. Therefore, [3] supports the hypothesis that aperture uncertainty is indeed the dominant performance-limiting factor.

Chapter 3

DELAY LINE ADC ARCHITECTURE

3.1 Architecture Overview

The proposed delay line ADC architecture is shown in Figure 3.1. The first stage consists of a delay line, a subtractor, and an estimation block, which is itself comprised of a coarse M-bit A/D converter, a set of digital registers and an estimation D/A converter. This is in contrast to the typical sub-ranging first stage, which does not contain the delay line or the registers but does contain a high-performance S/H block (see Chapter 2.1.2). The second stage of the delay line ADC is again similar to the second stage of the sub-ranging ADC, only it includes the low-pass reconstruction filter. As mentioned earlier, the goal of each of these changes is to reduce the demand on the S/H amplifiers in the sub-ranging configuration.

In a conventional sub-ranging architecture, the S/H block at the input to the first stage serves two purposes. First, it eliminates timing errors at the subtractor. As the signal propagates through the estimation block, some finite time delay will be introduced. Therefore, the correct estimate signal is not available to the subtractor for some finite propagation delay τ_{prop}. The input S/H holds the sampled input value for some time $\tau_{hold} > \tau_{prop}$. Therefore, at time τ_{prop} after the input is sampled, the two signals that are input to the subtractor correspond in the time domain.

The other purpose of the sub-ranging input S/H is to capture the time-varying input and holds while the estimation ADC performs the conversion. Without a S/H, the input signal to the first stage ADC might change before the comparators have completed their decision, and this results in a drop-off in the spurious-free dynamic range (SFDR). This

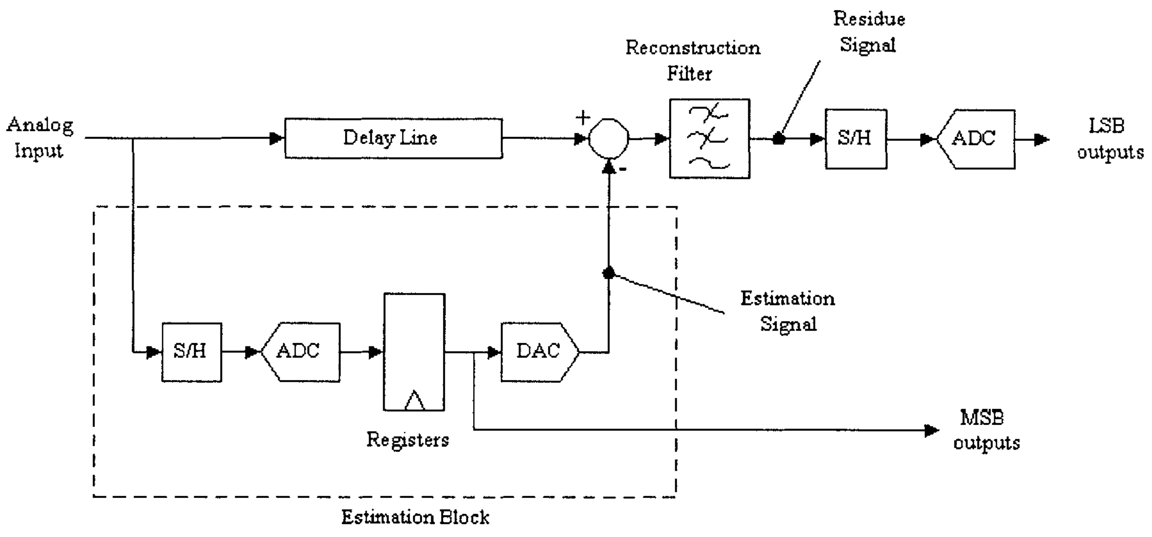


Figure 3.1 Proposed delay line ADC architecture.

effect can be dramatic, especially as input frequencies approach the Nyquist frequency. Therefore, a S/H preceding each A/D is necessary in high-speed converters. [12] Also, since this sampled value is propagated through to the second stage of the converter, it must be sampled to within the accuracy of the entire converter. It is important to note that an input S/H is not necessarily required for this purpose. In low-speed or low-resolution applications, the input signal may not change enough during the comparator decision time to cause an error. However, in high-speed applications, this is often not the case, and most high-speed ADCs have an input S/H. [12]

The delay line ADC architecture eliminates the need for this high-performance S/H block. It was just argued that one of the purposes of the sub-ranging input S/H block was to align the two signals at the input to the subtractor in the time domain. However, if the propagation delay through the estimation block τ_{prop}, is well defined, this function can be replaced with a delay line. The delay through this delay network should also be τ_{prop} for proper time alignment of signals. In the case of the delay line architecture, τ_{prop} is in fact well defined. The propagation delay through the estimation ADC, or latency, is a dependent on the ADC architecture implemented. Also, the phase delay between the S/H clock and the register clock is a design parameter that can be varied. Finally, the delay

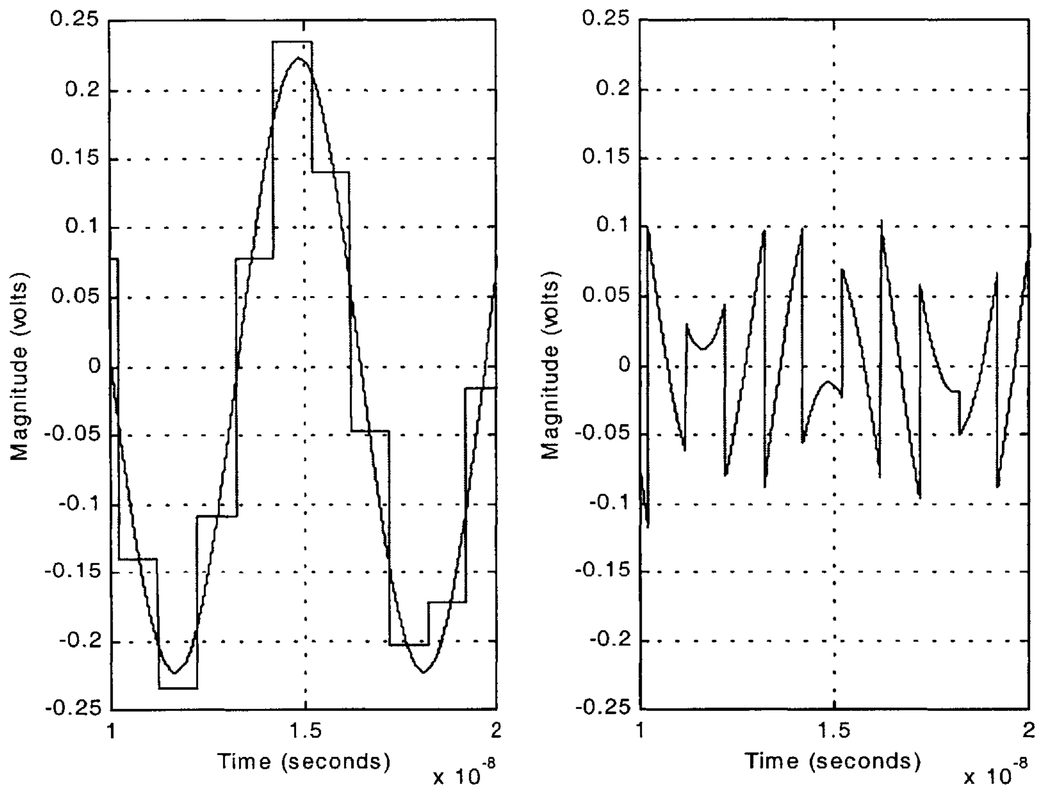


Figure 3.2 Delay line ADC waveforms. (a) Delayed input signal and estimation signal superimposed. (b) Difference between the delayed input signal and the estimation signal, i.e. the unfiltered residue signal.

through the DAC is determined by the switching and settling time of the DAC current sources. Therefore, the delay through this entire estimation block can be matched with a simple delay line, eliminating one of the needs for an input S/H block.

The other purpose of the input S/H in a sub-ranging architecture is to hold a steady input for the estimation ADC. However, this ADC is a coarse M-bit ADC (where M is typically 3-5 bits). Therefore, a high-performance S/H is not necessary. Instead, the capture function can be adequately performed by a S/H in the estimation path that samples the input with M-bit accuracy. Also, since this signal does not directly propagate into the second stage, an M-bit accurate S/H will not degrade performance. Therefore, the delay line A/D architecture eliminates the need for a high-performance sample-and-hold circuit in the first stage.

The low-pass reconstruction filter is added to the delay line A/D architecture to reduce the performance requirements of the second stage S/H block. In order to understand the performance requirements on the second stage S/H, the residue waveform must be examined. For this illustration, assume the input to the delay line ADC is a sinusoid. The output of the estimation block will be a quantized, zero-order hold version of that sinusoid. Figure 3.2a shows a 155MHz, $0.5V_{P-P}$ sinusoid, time shifted by the delay line, and the resulting estimation signal for an ideal (instantaneously DAC settling) 1GSPS, 4-bit estimation block. The estimate signal is then subtracted from the delayed input, leaving the residue signal that is shown in Figure 3.2b. In the absence of a reconstruction filter, this residue signal is converted by the second stage ADC.

The disadvantage of implementing the delay line ADC without a reconstruction filter can be seen in Figure 3.2b. The frequency content of the estimation signal contains the fundamental, or input, frequency, but it also contains many higher frequency harmonics. These harmonics are an artifact of the step-like response of a D/A converter. Therefore, the maximum rate of change of this residue signal is much greater than the maximum rate of change of the input signal. This increased rate of change places more strain on the second stage S/H. It must be able to sample an input that changes much faster than the fundamental with N-bit accuracy, where N is the number of bits resolved by the second stage.

The addition of the reconstruction filter reduces the maximum rate of change presented to the second stage ADC. In the frequency domain, the result of the reconstruction filter is an attenuation of the high frequency harmonics in the residue signal. The exact nature of this attenuation depends on the characteristics of the filter, which is a design parameter that will be explored in depth in Chapter 4. In the time domain, the effect of the reconstruction filter can be see in Figure 3.3, which shows the waveform from Figure 3.2b after filtering. In this example, the reconstruction filter is a fourth order lowpass Butterworth filter with a 300 MHz corner frequency. Essentially, the unfiltered residue signal is smoothed out. In Figure 3.3, the maximum derivative of this smoothed residue signal is about 11 mV/ns. If the purpose of the estimation stage were to generate a 4-bit

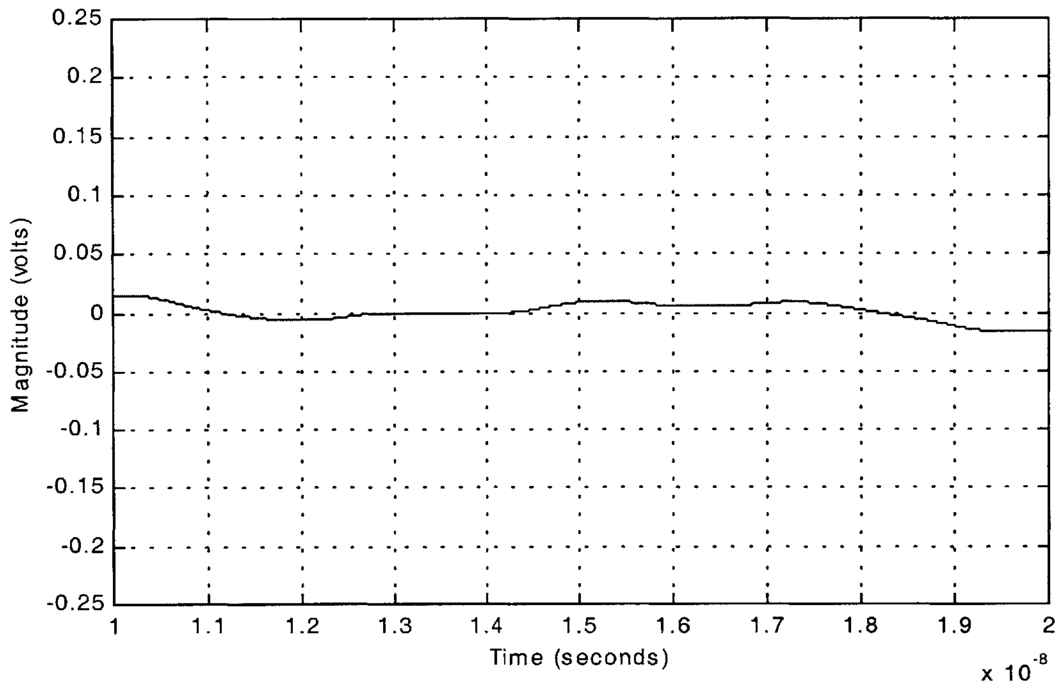


Figure 3.3 Residue waveform convolved with 4th order Butterworth filter impulse response.

estimate, this smooth residue signal would be multiplied by a gain of 2^4 and then converted by the second stage. In this case, the maximum rate of change of the residue signal would be 176 mV/ns, which is significantly smaller than the maximum rate of change of the input signal, 245 mV/ns. Therefore, the reconstruction filter achieves its goal of reducing the rate of change of the residue signal to a level below the maximum input sinusoid slope.

3.2 Calibration

3.2.1 Reconstruction Filter Issues

In the previous section, the benefits of the reconstruction filter were explained. The drawback of the reconstruction filter is the introduction of an effective error into the residue signal. In order to analyze this error, consider the residue signal without the reconstruction filter, shown in Figure 3.2b. The peak-to-peak magnitude of this signal is on the order of 200 mV. However, the accuracy of the estimation ADC is 4 bits in this example, and the estimation DAC is assumed to be at least as accurate as the estimation ADC. Therefore, at the sampling instants (0, 1ns, 2ns...), the value of the unfiltered residue signal is within about +/- ½ LSB, or 64mV peak-to-peak in the example shown.

This limit on the unfiltered residue error at the sampling instants is illustrated in Figure 3.4. This plot is simply a repeat of Figure 3.2b, zoomed in around several sampling instants. The horizontal lines delineate the +/- ½ LSB band in which the waveform must lie at the sampling instants. In the particular example shown, the sampling instants are at 10.2ns, 11.2ns, 12.2ns, and so on. The value of the unfiltered residue waveform at these time instants is marked with stars. As shown, the unfiltered residue signal always lies within this error band at the sampling instants.

However, the filtering operation smears the residue signal in the time domain. Once filtered, the magnitude of the residue signal at the sampling instants is no longer guaranteed by the accuracy of the estimation block. In Figure 3.3, the magnitude of the filtered residue signal, about 30 mV_{P-P}, is smaller than the 4-bit accuracy +/- ½ LSB limit. If, however, the magnitude of the filtered residue signal were not within the +/- ½ LSB error band, the accuracy of the delay line converter would be limited by the size of the residue signal. Non-optimal design values, such as a low oversampling ratio or high filter passband edge frequencies, cause the residue signal magnitude to increase and introduce error into the residue signal. Tradeoffs involved in these design choices are discussed in Chapter 4.

An interesting effect of the estimation operation can be extracted from Figure 3.5. In this figure, the simulation is identical to the simulation shown in Figure 3.2b, except that the input signal is a higher frequency sinusoid at 250MHz. Note that the period of the filter residue waveform is 4ns, which is the same as the period of the 250MHz input. The filtered residue is in phase with the delayed input, and this suggests that the magnitude of the fundamental component of the estimation signal is smaller than the delayed input. This effect is caused by the zero order hold function of the estimation block. The Fourier transform of a zero order hold is a sinc function, and the gain of this transform is not constant in the passband. Therefore, the magnitude of the estimate signal fundamental component is smaller than the delayed input signal. The zero-order-hold-induced error could be removed with a filter that has impulse response [13]

$$H_r(j\omega) = \frac{e^{j\omega T/2}}{\frac{2\sin(\omega T/2)}{\omega}} \tag{3.1}$$

Such a filter cannot be exactly realized in practice, and would a filter that approximates this impulse response would have to be implemented. This inverse-sinc filter is not implemented in this thesis, it is merely suggested as a possible optimization.

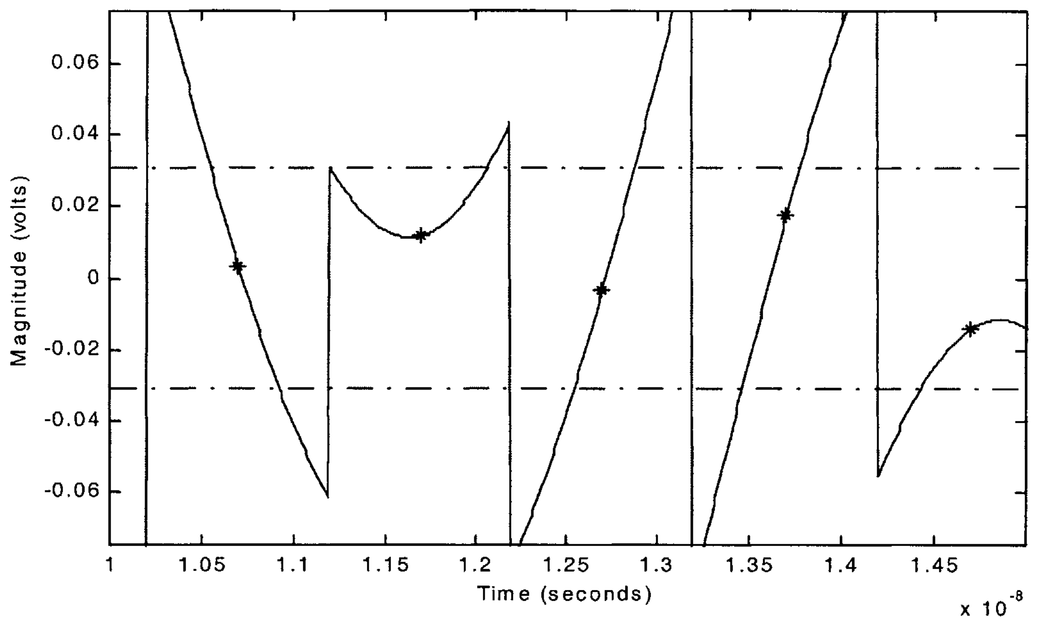


Figure 3.4 Unfiltered residue waveform, zoomed in around 5 sampling instants. Simulation set up is identical to waveform shown in Figure 3.2b.

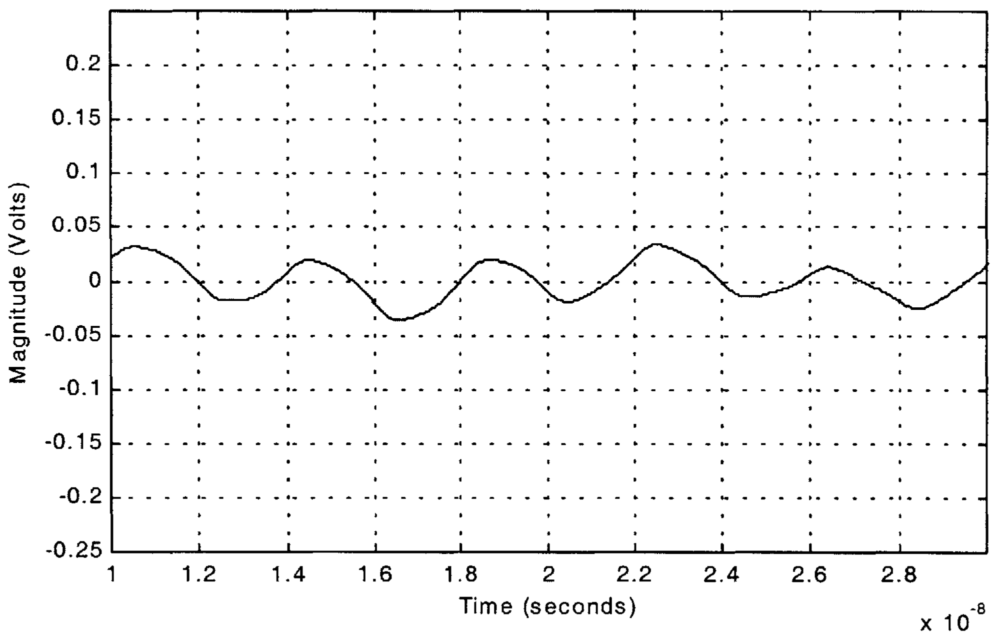


Figure 3.5 Residue waveform with reconstruction filter included in delay line ADC. Input frequency is 255MHz and the estimation block oversamples by a factor of 2. Reconstruction filter is a 4th order Butterworth with a 300MHz passband edge.

3.2.2 Necessity of Calibration

Due to the smearing function of the reconstruction filter, the residue waveform that is converted by the second stage is not directly related to the output of the estimation DAC. Instead, the residue waveform is the convolution of the filter impulse response and the unfiltered residue signal, which is in turn the difference between the delayed input and the output of the estimation DAC. Therefore, the output of the second stage ADC, i.e. the digitized residue signal, cannot be directly combined with the digital signal from the estimation block in order to get an high resolution representation of the input signal. Instead, the correct digital value can be obtained by combining the output of the second stage ADC should with the result of the estimation signal convolved with the filter response.

In order to facilitate this calculation, the filter needs to be calibrated. If the filter impulse response were known, any estimation signal could be convolved with this known response in order to find the portion of the residue signal that is due to the estimate.

Instead of simply calibrating the filter, however, both the estimation DAC and the reconstruction filter can be calibrated simultaneously. While DAC calibration is not necessary, the calibration algorithm described in the next section requires little more effort than filter calibration alone. Also, calibrating the DAC should eliminate sources of error in the DAC current switches from affecting system performance.

The following section describes the calibration algorithm necessary for filter calibration and also briefly discusses the advantages it offers. Tradeoffs involved in the calibration process are described in more detail in Chapter 4.

3.2.3 Calibration Algorithm

To calibrate the response of the estimation DAC and filter, the step response of each bit level transition, e.g. '0001' to '0010' or '1101' to '1100', is measured. The digital code at the input to the estimation DAC is switched while there is no input signal to the delay line, and the resulting digital output codes from the second stage are stored. Given an M-bit estimation stage and an N-bit second stage, there are $2*2^M$ transitions calibrated, and each calibration is known to N-bit accuracy. The number of samples that must be taken for each transition depends on the settling time of the DAC and filter. Each transition is sampled until the response settles to within N-bit accuracy.

Once all of the step responses are known, the filter output for any given DAC code can be digitally reconstructed. To reconstruct this signal, each of the previous X DAC codes in a history table, where X is the number of sample periods needed for the filter output to settle to N-bit accuracy, are combined with data stored during the calibration process to add up, step by step, the filtered estimation signal.

During normal operation of the delay line ADC, the digital output of the estimation ADC is used to reconstruct the component of the filter output that is due to the estimation signal. By summing this digitally reconstructed signal with the output of the second stage, a digital representation of the input signal, after being delayed and filtered, is determined. It is assumed that the passband of the low-pass filter contains all frequencies up to the Nyquist frequency, so the filtering operation should not affect the input signal and this calculated digital representation of the input signal should be accurate.

In order to illustrate the calibration scheme, a simple example is now explored. Consider a 2-bit estimation stage and a 3-bit second stage. Figure 3.6 shows both the calibration data table and the digital estimation code history table. As shown in the calibration data table, it takes four samples for the filter to settle to within 3-bit accuracy.

Transition	Sample 1	Sample 2	Sample 3	Sample 4
0 to 1	0	1	3	3
1 to 0	0	-2	-2	-3
1 to 2	0	2	3	3
2 to 1	0	-1	-2	-3
2 to 3	0	1	2	3
3 to 2	0	-1	-2	-3
3 to 4	0	-2	-3	-3
4 to 3	0	2	2	3

(a)

Time	DAC Code History
T-3	2 to 3
T-2	3 to 1
T-1	1 to 2
T	2 to 4

(b)

Figure 3.6 Calibration example. (a) Calibration data table. (b) Estimation digital code history table.

The data reconstruction process would proceed as follows. To begin with, the DAC code preceding the first transition in the history table must have been 2. Therefore, the calibration algorithm begins with a digital 2 in the estimation stage, which corresponds to a value of 6 in this case ($0 \rightarrow 1$, $1 \rightarrow 2 \Rightarrow 3+3 = 6$). For each of the entries in the DAC code history table, the corresponding entry in the calibration data code is summed. Therefore, at time T, the 2 to 4 transition has just taken place. The sample 1 column in the calibration data indicates that this transition will have produced a signal of magnitude 0+0=0 at the output of the filter. The previous transition was 1 to 2, and this occurred one sample period ago. Therefore, the sample 2 column indicates that the value of that transition at time T will be 2. Repeating these steps for the previous two transitions, the reconstructed data is –4 and 3. Summing all of these data, the estimation portion of the residue signal at time T should be $6 + 0 - 4 + 3 = 5$. Therefore, at time T, if the output of the second stage is 2, an accurate representation of the input signal is $5 + 2 = 7$.

From the above example, it becomes obvious that the calibration computation is heavily dependent on the number of entries in the tables. The calibration data table grows exponentially with the number of estimation bits. Therefore, it would be difficult to implement a high accuracy estimation block. Also, the number of samples that must be summed together grows linearly with the settling time of the DAC and filter combination, so the number of data entries in the calibration data table grows linearly with this same parameter. Clearly, a reconstruction filter that settles slowly can make the calibration process computationally prohibitive.

A critical assumption in the application of this calibration algorithm is the applicability of the principle of superposition. In order to be able to reconstruct the filter output as a sum of the response to single code change steps, superposition must apply. For example, if the filter response from a 2 to 4 transition were different then sum of the responses of a 2 to 3 and 3 to 4 transition, the calibration algorithm would not work.

Superposition should, in principle, be applicable to a unary current steering DAC. In an M-bit unary current D/A converter, there are 2^M total current sources, and the most significant bit switches 2^{N-1} of them, the second most significant bit switches 2^{N-2}, and so on. Since each of these current sources should match in a well-designed D/A converter, the assumption of superposition should be accurate.

It is less certain that superposition should hold for a binary weighted DAC. In an M-bit binary weighted DAC, there are M current sources. If the LSB current source conducts a current I, the MSB current source would conduct a current $2^{M-1}I$. The dynamics of these current sources are less likely to match as well as unary architecture current sources, and therefore the applicability of superposition is less clear.

Several other factors affect the validity in applying the principle of superposition to the estimation DAC. The applicability of superposition, as well as other advantages of the calibration procedure, is discussed in more detail in Chapter 4. As a general rule, however, a unary current steering DAC should be used in the estimation stage if possible.

Chapter 4

DESIGN CONSIDERATIONS

4.1 Reconstruction Filter

4.1.1 Frequency Domain Analysis

As noted in Chapter 3.1, the function of the reconstruction filter is to remove the higher frequency components of the residue signal converted by the second stage. This role is vital in the operation of the delay line ADC, since it determines how well the second stage S/H must perform. There are several aspects of the implementation of this filter that significantly affect the performance of the overall system.

Since the role of the filter to remove high frequency components of a signal, a linear filter with a steep roll-off in the filter response curve would seem ideal. Linearity is in the filter is required to ensure the applicability of superposition, which is necessary for calibration, as discussed later in this chapter. Assuming linearity and that the delay line converter is designed for Nyquist rate operation, the filter design such that the 3dB roll-off frequency would occur at the Nyquist frequency. If the filter had, in fact, perfect brick-wall cutoff characteristics, the filter output would eliminate all high frequency signal components so that the signal converted by the second stage would change no faster than the maximum input frequency, which is assumed to be the Nyquist frequency.

Of course an ideal frequency selective filter is not physically realizable since it is, by definition, non-causal. Instead, a causal approximation of the ideal filter characteristics must be made. [13] The frequency characteristics of such non-ideal filters can be described by several parameters. First, the 3dB cutoff frequency of a low-pass filter, often referred to as the passband edge, is defined as the frequency at which the filter response is –3dB below unity (i.e. the filter output is attenuated by a factor of 0.7071). Similarly, the stopband edge is defined as the frequency at which the filter response is, at minimum, attenuated by some factor α_S. Therefore, if two filters are specified with identical passband and stopband edges but the first filter has a stopband attenuation factor of –60dB while the attenuation factor of the second is only –40dB, the first filter will exhibit better high frequency harmonic rejection.

In the delay line ADC, an apparent choice for the reconstruction filter would be a low-pass filter with a passband edge at, or slightly above, the Nyquist frequency. Pushing the passband edge slightly above the Nyquist frequency ensures that input signals up to the Nyquist frequency will not be attenuated significantly by the filter. Also, the reconstruction filter should have a stopband edge at a frequency low enough to reject the significant high frequency components of the estimation signal. Since several harmonics in the estimation signal can be aliased down near the fundamental input signal depending on the sampling rate, it is important to design a filter that can reject not only the second harmonic, but also the first several harmonics.

To illustrate the effect of aliasing, consider a 100MSPS Nyquist rate converter. Therefore, the maximum input frequency is 50MHz. Also, for reasons that will be discussed in Chapter 4.2, assume that the estimation block oversamples by a factor of two. Therefore, the estimation block samples at a 200MSPS rate, or once every 5ns. Now consider all of the frequency components of the unfiltered residue signal under 200MHz. If the input signal were a 45MHz sinusoid, the fundamental frequency component would is at 45MHz, and it is also aliased at 155MHz (200MHz - 45MHz). The second harmonic, 90MHz is aliased at 110MHz. Finally, the third harmonic, 135MHz would be aliased at 65MHz. Therefore, the unfiltered residue signal has spectral components besides the fundamental at 65MHz, 90MHz, 110MHz, 135MHz, and 155MHz. If all non-fundamental components are to be attenuated by –40dB, the stopband edge must be at 65MHz, which is an alias of the 3rd harmonic, not at the 2nd harmonic.

A more graphic illustration of the importance of the filter frequency response is shown in Figures 4.1 and 4.2. Typical delay line ADC waveforms are shown in Figure 4.1, similar to those shown in Figure 3.2. The unfiltered residue waveform, Figure 4.1b, is convolved with the impulse response of a reconstruction filter in order to determine the signal converted by the second stage of the ADC. Figures 4.1c and 4.1d show the resulting waveform for two different filter topologies. In both cases, the input waveform is a 4-bit estimate of a 100MHz sinusoid sampled at 500MSPS, and the passband edge of the reconstruction filters is 250MHz. Figure 4.2a shows the residue waveform after being filtered by a 4th order Butterworth filter. In this case, the maximum rate of change is 192mV/ns. Figure 4.2b shows the residue waveform resulting from a 6th order elliptic (Cauer) filter with α_S = -60dB, for which the maximum rate of change is 130mV/ns. In addition to the difference between the rates of change of each waveform, the magnitude of the elliptic filter output is smaller than the Butterworth filter output. A smaller residue signal corresponds to increasing resolution. If the magnitude were ½ the size, an extra bit of accuracy could be resolved. The results seem to be better for the elliptic filter because the frequency response of an elliptic filter rolls off much more steeply than a Butterworth filter. Clearly, the type of filter and filter order both have significant impacts on the overall system performance.

It is important to note that since the reconstruction filter is in the forward signal path, it limits the frequency response of the delay line converter. For example, if a 500MSPS delay line converter were intended for Nyquist rate operation, the reconstruction filter might be designed such that the passband edge is at 300MHz. However, if a signal above the Nyquist frequency were input to the system, such as a 500MHz input sinusoid, it would not be accurately converted. Instead, the reconstruction filter would attenuate the fundamental in addition to the unwanted harmonics. If the response of the reconstruction filter were –30dB at 500MHz, the input signal would be attenuated by 30dB, corresponding to a resolution reduction of about 5 bits. Therefore, the passband edge must be carefully selected to eliminate as many high frequency harmonics as possible while not compromising system frequency response characteristics.

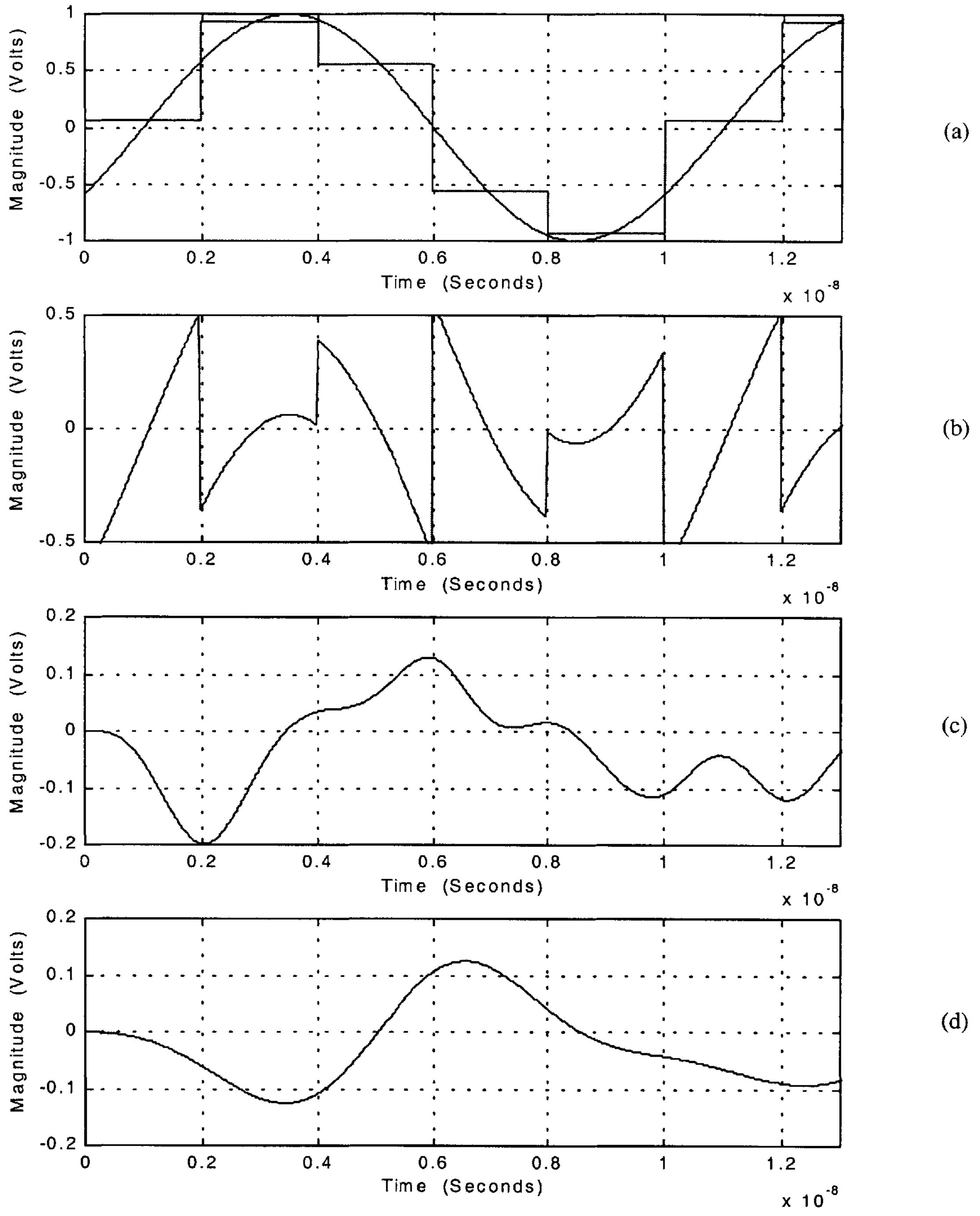


Figure 4.1 Typical delay line ADC waveforms, before and after reconstruction. (a) Delayed input signal and estimation signal superimposed. (b) Unfiltered residue signal. (c) Residue signal filtered with 4th order Butterworth filter. (d) Residue signal filtered with 6th order Elliptic filter.

4.1.2 Time Domain Analysis

The analysis in the previous section suggests that a linear filter with as sharp as possible a magnitude vs. frequency roll-off would be ideal since it would result in the most dramatic harmonic attenuation. However, even a filter with a perfect brick-wall frequency response turns out to be not ideal. While it is the case that the frequency response of the filter plays a significant role in determining the performance of the overall converter, the time domain response of the filter also warrants serious design consideration.

Chapter 3 examined the heavy dependency of the calibration computational load on the number of entries in the calibration data table. As the number of entries in the calibration data grows, the computation required to reconstruct the estimation signal grows linearly. The relationship between the settling time of the reconstruction filter and the number of entries in the calibration table is also examined in Chapter 3. As the time required for the reconstruction filter step response to settle to within a certain error limit grows, the number of entries in the calibration table grows linearly. Thus, it is very advantageous to design a reconstruction filter that settles quickly in response to a step input.

The advantages of quickly-settling filters introduce a tradeoff in the design of the reconstruction filter. This tradeoff is due to the fact that sharp frequency response roll-offs correspond to slowly-settling step responses. The ideal frequency selective filter, although physically impractical, has a step response that rings indefinitely (for both $t<0$ and $t>0$). As the sharp frequency characteristics of the ideal filter are more closely approximated, the ringing step response is also more closely matched.

This tradeoff is illustrated in Figure 4.2, which shows the step responses of both the 4th order Butterworth and the 6th order elliptic filters described in Figure 4.1. Clearly, the step response of the Butterworth filter settles much more quickly than the response of the elliptic filter. In fact, in order to settle to 10-bit accuracy, or 1 part in 1024, the 4th order Butterworth filter takes about 12.6ns while the 6th order elliptic filter takes about 80.0ns.

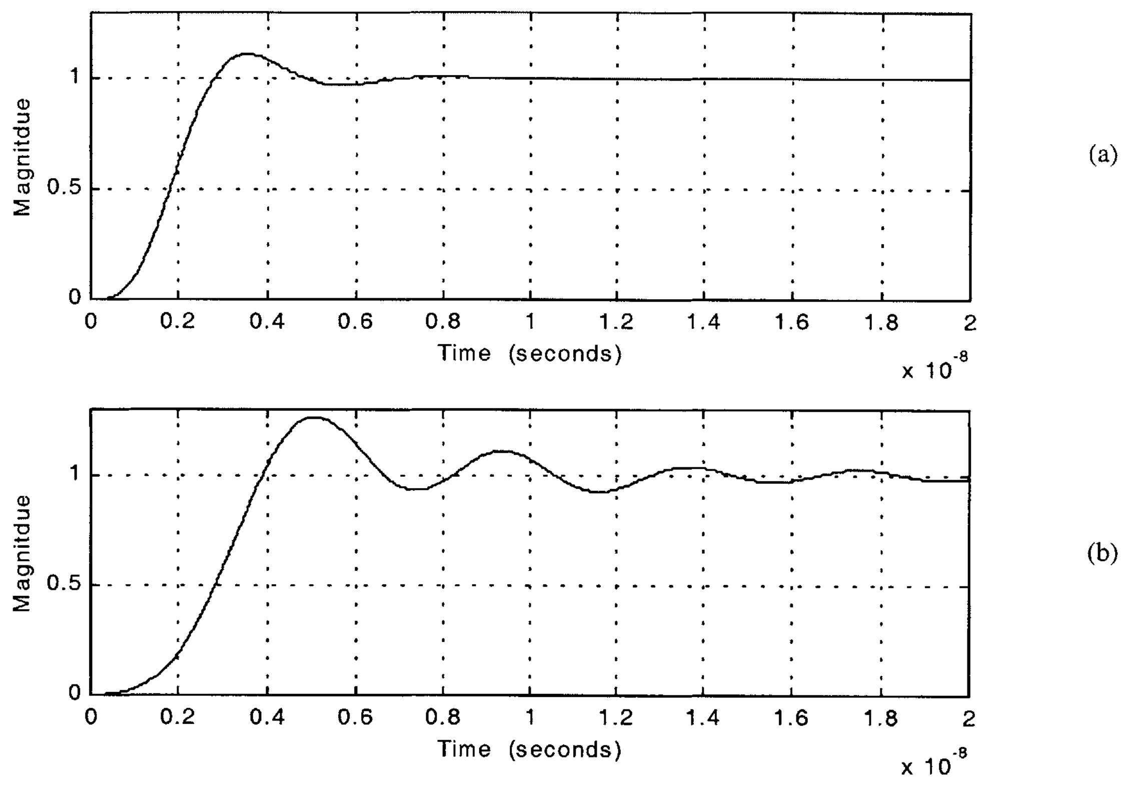


Figure 4.2 Unit step response of various reconstruction filters. (a) 4th order Butterworth filter with 250MHz passband edge. (b) 6th order elliptic filter with 250MHz passband edge, 1dB passband ripple, and 60dB stopband attenuation.

Therefore, the 6th order elliptic filter would require a calibration data table over 6 times as large as the table required by the Butterworth filter. This significant disadvantage more than offsets the modest advantage in harmonic reduction of the elliptic filter, rendering the 6th order elliptic filter an impractical design choice.

4.1.3 Issues in Filter Selection

Taking into consideration all of the issues discussed in the previous two sections, a reconstruction filter must be selected for the delay line ADC. In order to select a filter, the filter performance parameters must be weighted so that while all of the tradeoffs are addressed, the critical issues more strongly influence design choices. In assigning these weights, the observation must be made that the time domain response is of critical importance. As described in the Chapter 3, the settling time of the filter greatly affects the computation required to perform an A/D conversion. This computation is considered to be the most critical design parameter. Heavy computation load shifts the burden of A/D conversion to the post processing elements. While this shift might be advantageous, depending on the manner in which the post processing is implemented, it is not a focus of this thesis. Instead, the signal reconstruction algorithm is considered a tool in the design of the delay line A/D converter and performs a function necessary for operation, but is not to be heavily relied on.

For simplicity of operation, only two types of filters are considered for implementation, Bessel filters and Butterworth filters. Bessel filters are designed to achieve a maximum bandwidth while maintaining maximally linear phase. The magnitude response across the passband is flat and the phase response is linear, which preserves the waveshape of the signal and results in a uniform time delay for signals in the passband. Butterworth filters are designed to exhibit a maximally flat magnitude response in the passband. [14] These two filters were chosen because the step response of each exhibits minimal ringing. The Bessel filter step response, in fact, does not ring at all, as would be expected due to its linear phase characteristic. It is assumed that minimal ringing corresponds to quick filter settling. This assumption, of course, it not necessarily accurate, since a step response that overshoots and then rings slightly might settle more quickly than response that does not ring at all. However, the step response of both the Bessel and Butterworth filters settles more quickly than the step response of other conventional analog filters, namely the elliptic filter and Chebyshev filter, type I and II. There are, of course, other filter types, such as transitional filters, which are more complex to implement and not considered here. [14]

Given these two filter types, the effect of filter order on settling time must be considered. Figure 4.3 shows the results of a simulation that determines the time required for both Bessel and Butterworth filters, 2nd, 4th, and 6th orders each, to settle to within a certain error magnitude. Note that the horizontal axis is normalized to the passband edge frequency. Therefore, a 4th order Butterworth filter with a 300MHz passband edge would

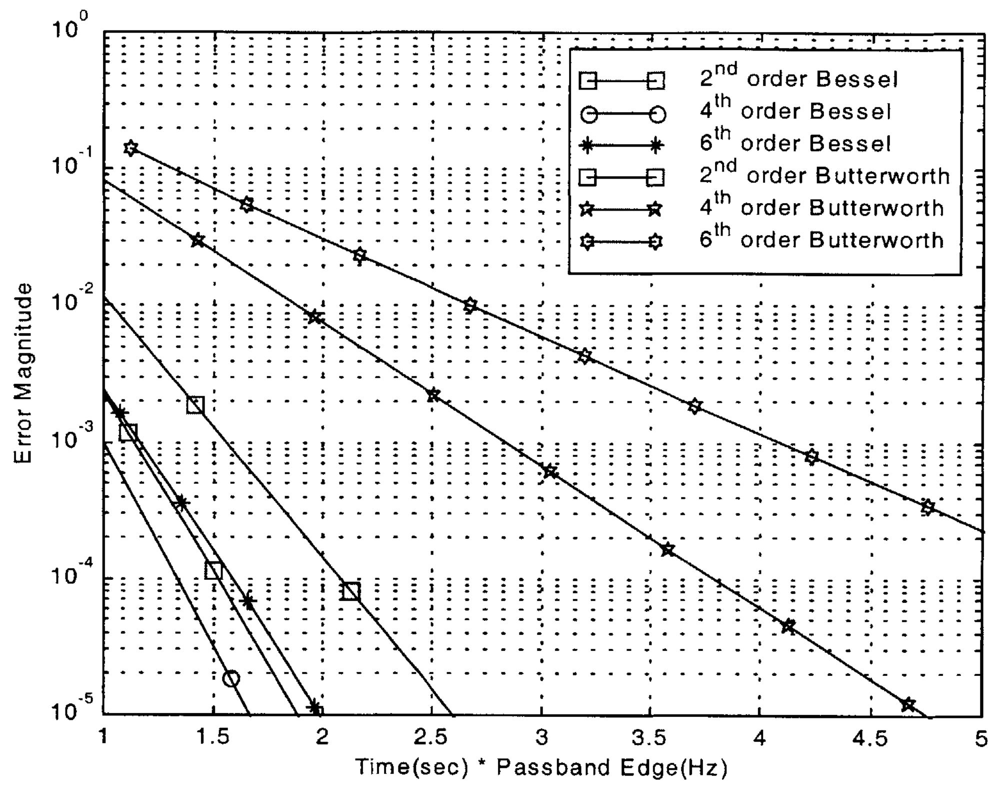


Figure 4.3 Step response settling time of Bessel and Butterworth low-pass filters, 2nd, 4th, and 6th orders each.

settle to 10-bit accuracy (0.1%) in about 10ns. Several observations may be made from this graph. First, Bessel filters settle much more quickly than Butterworth filters, between a factor of 2 and 4 more quickly for the filter orders considered. The ratio of Butterworth settling time to Bessel filter settling time increases as the accuracy to which the filter must settle increases, so as higher accuracy is required, the difference between Bessel and Butterworth settling is more apparent. Also, the order of a Bessel filter does not have a very significant effect on the settling time, unlike that same effect in a Butterworth filter. That is, increasing the filter order from 4 to 6 in a Bessel filter increases the 10-bit settling time by about 4%, whereas the same order change in a Butterworth filter increases the 10-bit settling time by 50%.

Clearly, the Bessel filter has an edge in terms of settling time, and this advantage increases as the settling accuracy required increases. However, the frequency

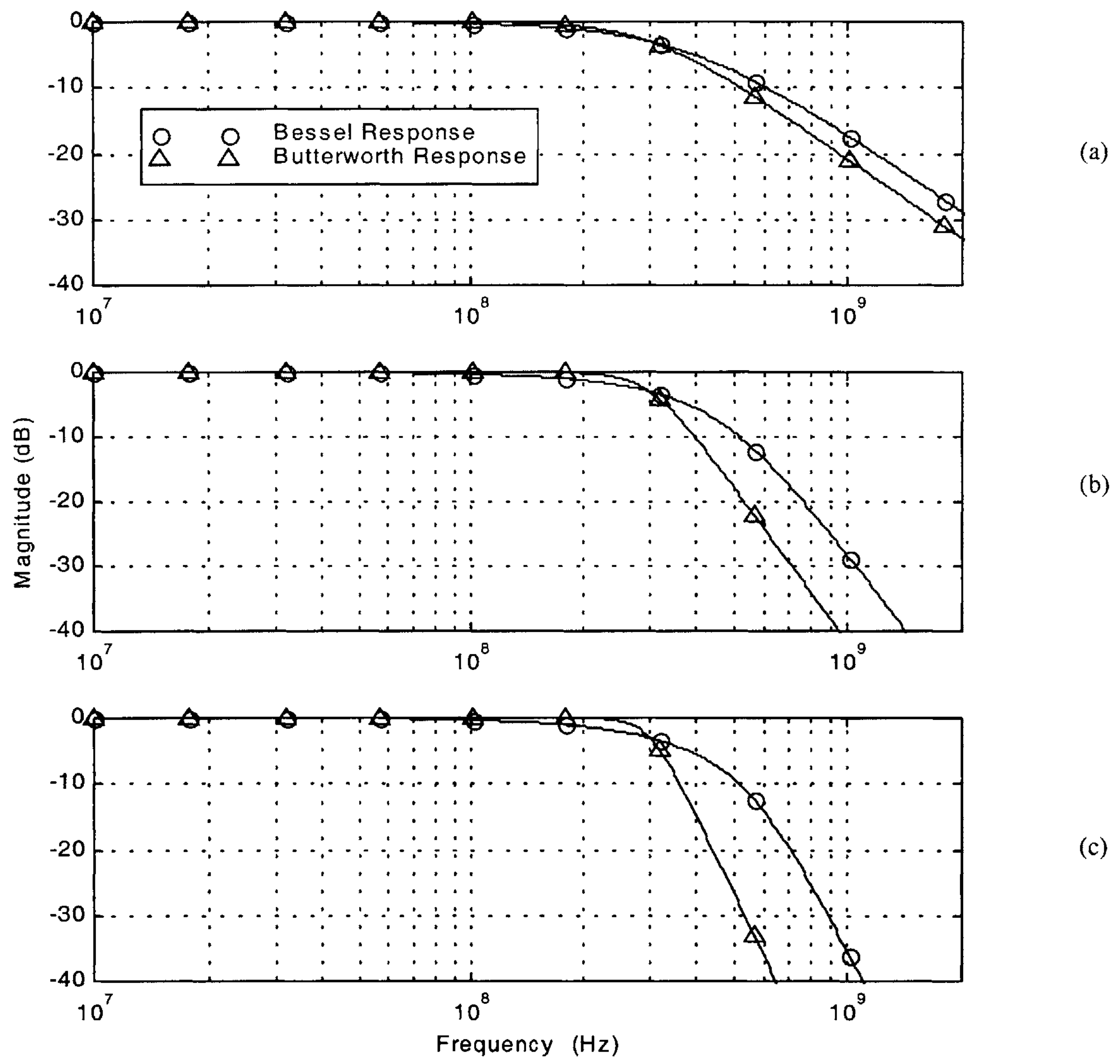


Figure 4.4 Frequency response magnitude plots of both Bessel and Butterworth low-pass filters with 300MHz passband edges. (a) 2nd order filters. (b) 4th order filters. (c) 6th order filters.

characteristics of each filter must also be examined. The magnitudes of the filter frequency responses for both the Bessel and Butterworth filters with 300MHz passband edges are shown in Figure 4.4. Clearly, the slope of the filter response after the passband corner is proportional to the filter order; thus higher order filters will attenuate the unwanted high frequency components of the residue signal more effectively. Also, in all three examples, the Butterworth response roll-off is steeper than the Bessel roll-off. Therefore, a Butterworth filter will attenuate the high frequency components better than a

Bessel filter of the same order. The difference between the roll-off characteristics of the Bessel and Butterworth filters is also dependent on filter order, similarly to the difference between the settling times of the two filters. As the order is increased, the roll-off characteristics of a Butterworth filter improve as compared to the roll-off of a Bessel filter.

4.1.4 Filter Simulation Methodology

Before selecting a filter for implementation, the time domain and frequency domain tradeoffs analyzed so far need to be applied to the problem at hand. In order to investigate the effect of these filter characteristics on the performance of the delay line ADC, a model of the delay line ADC is simulated with different reconstruction filters and the results are analyzed. Before analyzing the results of the simulation, however, the simulation model needs to be discussed.

The simulation is performed in MATLAB, using Simulink, and a block diagram of the simulated system is shown in Figure 4.5. Since the purpose of the simulation is to examine the effects of different filter configurations on system performance, blocks other than the reconstruction filter are modeled ideally. Therefore, the simulated system performance reflects the best case for a given filter setup. The delay line in the forward path is modeled as a pure delay block. The idealized model for the estimation ADC is a quantizer, and the estimation DAC is modeled as a de-quantizer. Both the quantizer and the de-quantizer are modeled as 8-bit accurate. Since the estimation block is only expected to resolve between two and five bits, the accuracy of the quantization process is not a limiting factor. Also, because only the residue signal is important, the second stage of the delay line ADC is not included in the simulation. Finally, Simulink steps through the simulation at fixed time step intervals, and these intervals are chosen to be much smaller than the sampling frequency of the simulated converter. Therefore, the simulation results should be accurate on a much finer time scale than the delay line converter that is being simulated.

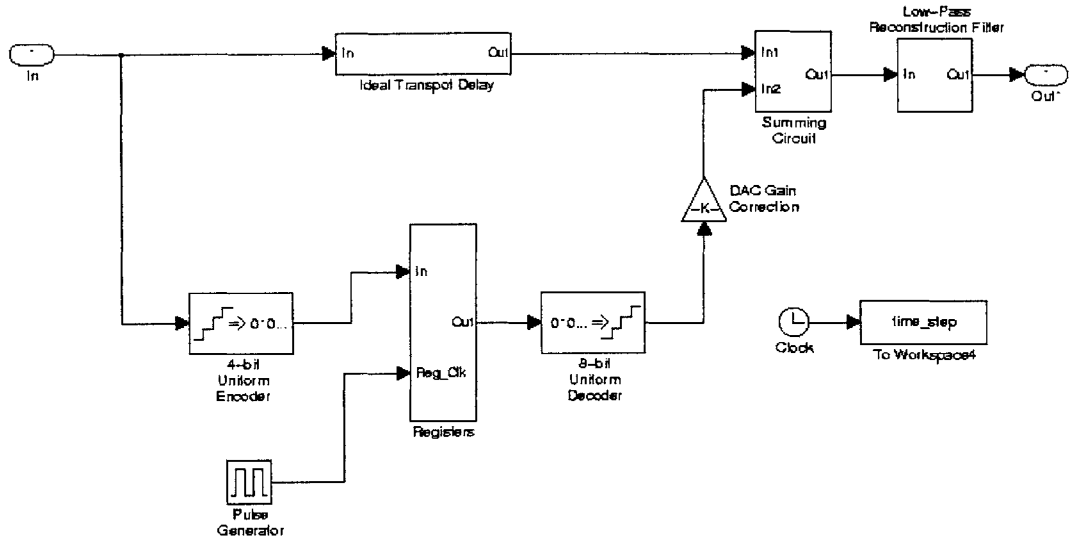


Figure 4.5 Block diagram of delay line ADC system model used for simulation. This system is intended to model the effects of different filters on performance.

Nominal values for the sampling rate and oversampling ratio in the estimation block need to be chosen in order to simulate the converter. A nominal sampling rate in the estimation block of 1GSPS is used in all simulations. This sampling frequency is chosen to represent very high-speed data conversion. Of course, due to the ideal modeling of the converter (i.e. aperture jitter is not included), the simulation results would be identical at different sampling rates as long as other system parameters, such as the delay time through the delay block and the filter passband edge, scale accordingly. Also, the oversampling ratio of the estimation block is a design parameter that is varied in simulations. The effect of oversampling will be discussed in Chapter 4.2, and the remainder of this section will analyze simulation results for a nominal input frequency of 255MHz. The Nyquist frequency for a 1GSPS converter is 500MHz, so the following simulation results represent the performance of a 2X oversampling delay line converter.

There are two measures of performance in these simulations. First, the peak-to-peak magnitude of the filtered residue signal is measured. As discussed in Chapter 3.1, the filtering operation smears the residue signal with respect to the sampling instants. Therefore, the effective error magnitude of the residue signal is not guaranteed to be +/- ½ LSB but is instead the maximum peak-to-peak error of the residue signal. In all of the simulations conducted, the full-scale input is defined as 1 V_{P-P}, so the residue magnitude is measured in mV_{P-P}. However, for generalization, the results are converted into effective bits. Therefore, a peak-to-peak residue magnitude of 62mV is +/- ½ LSB accurate for a 3-bit resolution.

The second simulation performance measure is the maximum rate of change of the filtered residue signal. Assume that the estimation block is expected to generate a filtered residue signal accurate to 3 bits. Therefore, before the second stage S/H, a gain block of 2^3 will increase the residue signal back up to full scale for the 2nd stage ADC. In doing so, the maximum rate of change of the filtered residue signal is also increased by a factor of 2^3. Therefore, if the reconstruction filter serves its purpose of presenting the second stage S/H with a signal that does not change faster than the input signal, the maximum slope of the residue signal (before the gain block) must be a factor of 2^3 smaller than the maximum rate of change of the input signal. Thus, the maximum rate of change of the filtered residue signal is an important performance metric for each simulation. As with the magnitude of the residue signal, the rate of change is normalized and presented as a number of bits of slope reduction.

4.1.5 Filter Simulation Results

Having generally described the setup of the simulation, the simulation results can now be discussed. The performance of the idealized delay line converter is simulated for 4th and 6th order Bessel filters as well as 2nd, 4th, and 6th order Butterworth filters. For each of these filters, many simulations are run with different passband edges, between 200MHz and 500MHz in 50MHz intervals, so that the maximum performance is found. Figure 4.6 shows typical performance versus passband edge frequency. In this example, the

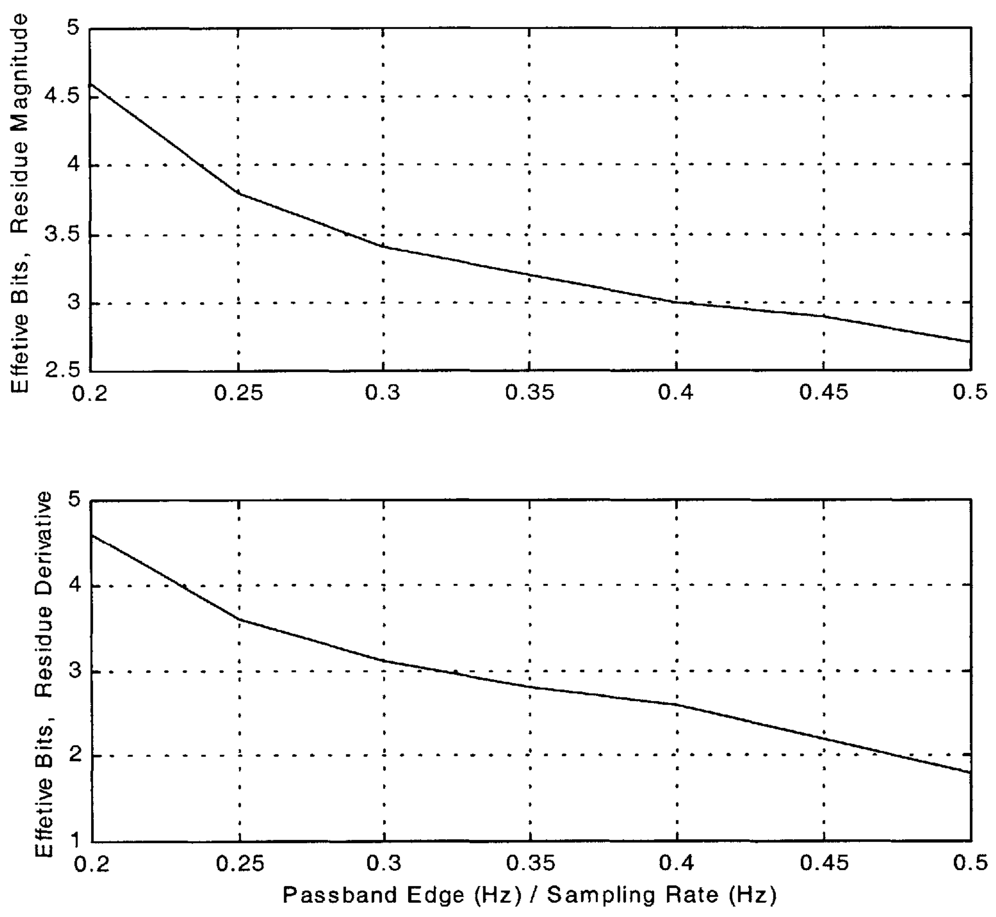


Figure 4.6 Performance versus passband edge frequency normalized to sampling rate for delay line ADC with 4th order Butterworth filter. (a) Magnitude of filterer residue signal in effective bits gained. (b) Rate of change of filtered residue signal in effective bits gained.

simulated filter is a 4th order Butterworth filter. Note that the horizontal axis is normalized to the sampling frequency of the estimation block since the results are identical for different sampling frequencies. Clearly, as the passband edge of the reconstruction filter is moved to lower frequencies, the both performance metrics increase. This increase suggests that the best performance would be attained by cutting off the filter response at 0.2 * f_{samp} (estimation block sampling frequency), or even lower.

However, lowering the passband edge frequency changes the overall system frequency response. If the delay line ADC is intended for 2X oversampling with regards to Nyquist rate operation, the passband edge of the reconstruction filter can not be below 0.25 *

f_{samp}. In order to ensure ½ Nyquist rate operation, the minimum passband edge is defined as 0.3 * f_{samp}. Since Figure 4.6 suggests optimal performance is achieved for low passband edge frequencies, the cutoff frequency of the reconstruction filter should be placed at this minimum value. Thus, for a 1GSPS converter with 2X oversampling, the passband edge of a 4th order Butterworth filter should be placed at 300MHz. Note the ideal value of 0.3*f_{samp} only refers to a converter that oversamples by a factor of 2. If the oversampling ratio were increased by a factor of 2 from 1GSPS to 2GSPS, the ideal passband edge would remain at 300MHz, or 0.15*f_{samp}.

These passband edge characteristics hold for the other filters simulated, too. In each case, the two performance metrics used monotonically increase with decreasing passband frequencies. Since the desired overall delay line ADC frequency response determines the minimum passband edge requirement, the passband cutoff should be placed at this minimum value, regardless of the filter type.

Having decided on the ideal placement of the passband edge, the effects of filter type on system performance must be explored. A table listing the performance metrics measured during simulations of the delay ADC with several different filter types and orders are shown in Figure 4.7. Note that for the reasons mentioned earlier, these metrics were measured using filters with passband edges at 0.3 * f_{samp}. There are several points of interest in these results. First, in all cases, magnitude reduction of the residue signal is equal to or greater than the slope reduction. In order for the estimation block to add three bits to the system performance, both the magnitude reduction and the slop reduction must be at least three effective bits. Therefore, the maximum accuracy that can be achieved by the estimation block is limited by the reduction in slope of the residue signal by the filter.

Filter Type	Filter Order	Residue Magnitude (Effective bits)	Residue Derivative (Effective Bits)
Bessel	2	2.8	1.3
Bessel	4	3.2	2.6
Bessel	6	3.3	2.8
Bessel	8	3.4	2.9
Butterworth	2	2.6	1.8
Butterworth	4	3.4	3.1
Butterworth	6	3.4	3.3

Figure 4.7 Performance metrics achieved by various filter types and filter orders. Passband edge set at $0.3 * f_{samp}$ in each case.

This, of course, is making the assumption that the slope of the filtered residue signal change no faster than the maximum input frequency. Given, however, that the stated goal of the delay line converter is to reduce the input rate of change demands on S/Hs in the system, this assumption is valid.

It is also evident from Figure 4.7 that second order filters do not perform well in the delay line ADC. Both the 2nd order Bessel and the 2nd order Butterworth filter perform poorly, not even achieving 2-bit performance. Simply put, the frequency response of a 2nd order filter does not roll off quickly enough in order to attenuate the higher order harmonics that it is intended to remove. As a result, the residue derivative is large and the 2nd order filters do not effectively achieve their goals.

A third observation that can be made is the relative ineffectiveness of the Bessel filter in comparison to the Butterworth filter. In order to achieve 3-bit performance in the estimation stage, a Bessel filter of 10th order would have to be used, while a 4th order Butterworth filter achieves that performance level. Figure 4.3 does not show the settling time required for a 10th order Bessel filter. By extrapolation, though, these filters are expected to settle significantly faster than a 4th order Butterworth filter. Therefore, 3-bit performance should be achievable with a 10th order Bessel filter that settles more quickly than a 4th order Butterworth. As a practical matter, however, 10th order filters are significantly more complex and difficult to implement as compared to 4th order filters, and higher complexity introduces more potential sources of error. For this reason, a 10th order Bessel filter, despite superior settling time, is considered a less viable filter option than the lower order Butterworth filters.

Given all of the analysis presented in this section, the 4th order Butterworth filter is selected as the optimal reconstruction filter. Figure 4.7 shows that it is the lowest order filter capable of achieving 3-bit performance. Also, the settling time is not horrendous, as shown in Figure 4.3. The 4th order Butterworth filter should be able to settle to 10-bit accuracy in about 12ns, which is only about a factor of 2 worse than the high order Bessel filter required to achieve 3-bit accuracy. In order to achieve another bit of accuracy, a very high order Butterworth (10th or higher) or a more selective filter, such as an elliptic filter, would have to be used. However, the settling times of these filters are prohibitive. Therefore, the 4th order Butterworth is a good balance of reasonably quick settling, 3-bit magnitude and slope performance, and simplicity of implementation.

4.2 Oversampling

Another design parameter that can be varied is the oversampling rate of the estimation block. As mentioned earlier, a Nyquist rate converter that samples at a 1GSPS rate can convert input signals up to 500MHz. That is, the maximum allowed input frequency is ½*f_{samp}. If the maximum allowed input frequency is less than the Nyquist rate, the converter is oversampling. The oversampling ratio (OSR) is defined as

$$OSR = \frac{f_{samp}}{2 \times f_{in,\max}} \tag{4.1}$$

There are advantages gained from oversampling that allow extra bits of resolution to be extracted from a converter. For example, consider the delay line ADC architecture. After the input signal is N-bit quantized by the estimation block, it is subtracted from the delayed input signal and then filtered. Due to the linearity of the filtering operation, it is fundamentally no different to filter both the estimation signal and the delayed input signal and subtract the two afterwards. For simplicity, consider the signal $y_2(n)$, which is the estimation signal after filtering and before the subtraction operation. Assuming that the input signal is a sinusoid, the maximum peak value of $y_2(n)$ is $2^N(\Delta/2)$, where Δ is equal to the size of an LSB. For this maximum sinusoidal wave, the signal power, P_s, has a power equal to

$$P_s = \frac{\Delta^2 2^{2N}}{8} \tag{4.2}$$

The power of the input signal within $y_2(n)$ remains the same whether the input is oversampled or not. However, the quantization noise power, P_e, is reduced [6]

$$P_e = \frac{\Delta^2}{12}\left(\frac{1}{OSR}\right) \tag{4.3}$$

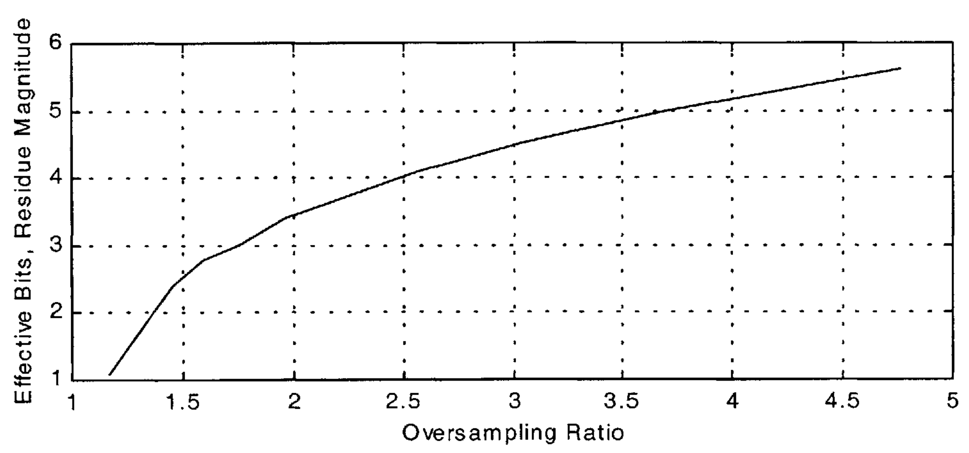
Therefore, doubling the OSR decreases the quantization noise power by a factor of two. The maximum SNR of $y_2(n)$ can also be calculated to be the ratio of the maximum sinusoidal power to the quantization noise. Combing 4.2 and 4.3, we obtain (in dB):

$$SNR = 6.02N + 1.76 + 10 \times \log(OSR) \tag{4.4}$$

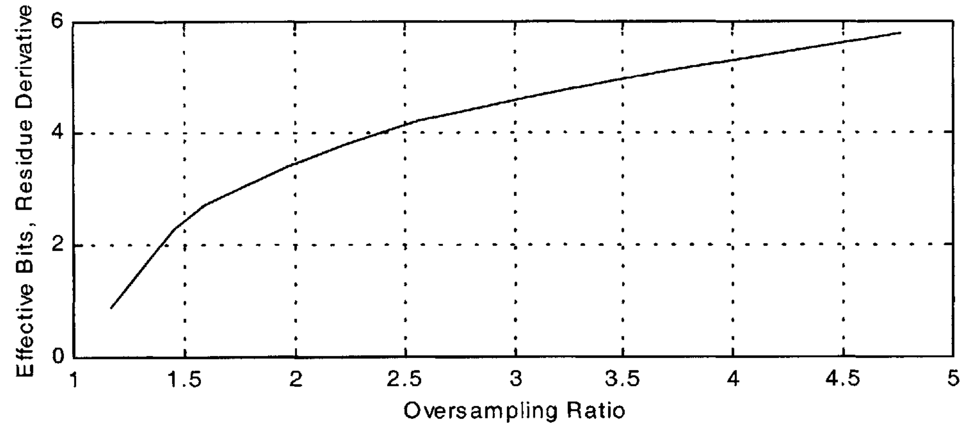
The first term in equation 4.4 is due to the N-bit quantization while the OSR term is the SNR enhancement obtained from oversampling. Clearly, oversampling gives a direct improvement of 3 dB/octave, or 0.5 bits/octave. The reason for this improvement is that when quantized samples are low-pass filtered, or averaged together, the signal portion adds linearly whereas the noise portion adds as the square root of the sum of the squares. [6]

In addition to the reduction in noise power, the filtering operation reduces the rate of change of the estimation signal. As discussed in the previous chapter, this slope reduction relaxes the requirement on the second stage S/H. By increasing the oversampling ratio of the estimation block, the slope reduction achieved should also increase. In order to understand this increase in slope reduction, consider a Nyquist rate converter that does not oversample. If the maximum allowable input is f_0, the sampling rate is $2*f_0$. Also, the reconstruction filter passband edge is at some frequency slightly higher than the Nyquist frequency, $f_0 + \Delta f$. If the oversampling ratio is then increased by a factor of 4, the sampling rate is $8*f_0$. However, the reconstruction filter remains at $f_0+\Delta f$. Therefore, the sampling rate is increased in relation to the passband edge as the oversampling ratio increases.

In order to make this idea more concrete, consider the example discussed in Chapter 4, in which f_0 = 50MHz. As described in this previous example, an oversampling ratio of 2 leaves the second harmonic at 90MHz and the third harmonic at 135MHz, as well as aliasing the third harmonic down to 65MHz. If the oversampling ratio is increased to 4, the second harmonic remains at 90MHz, but the third harmonic will not be aliased down to 65MHz, so it only remains at 135MHz. The third harmonic is instead aliased down to 265MHz, a much higher frequency than with the lower OSR. Therefore, in order to attenuate the second and third harmonics, the reconstruction filter needs to attenuate components at 90MHz and higher, not 65MHz and higher. Clearly, as the oversampling ratio is increased, the number of aliased harmonics lower in frequency than the un-aliased second harmonic that need to be attenuated by the reconstruction filter



(a)



(b)

Figure 4.8 Delay line converter performance metrics vs. oversampling ratio. (a) Filtered residue magnitude in effective bits. (b) Filtered residue maximum slope in effective bits.

decreases as the oversampling ratio is increased. Thus, for high oversampling rates, the reconstruction filter only needs to deal with harmonic components at frequencies above the un-aliased second harmonic.

In order to test these arguments for the advantages of oversampling, several different oversampling ratios are simulated. The results of these simulations are shown in Figure 4.8. In each of these simulations, the delay line converter samples at 1GSPS and the OSR is changed by varying the input frequency. The reconstruction filter is a 4th order Butterworth filter. Also, the filter passband edge is at $0.6*f_{samp}/OSR$, which is the best cut-off frequency, as argued in Chapter 4. Therefore, for the OSR=2 data, the passband edge is at 300MHz, while for OSR=5, the passband edge is at 120MHz.

Several things are apparent in Figure 4.8. First, a 4^{th} order Butterworth reconstruction filter does not perform very well at low oversampling ratios. In order to achieve 3-bit estimation performance, an OSR of slightly less than 2 is needed, and 4-bit performance can be achieved for oversampling ratios above 2.5. Also note that both performance metrics increase as the oversampling ratio increases, and the rate of increase appears logarithmic, as is predicted by equation 4.4.

These results suggest that optimum performance would be achieved if the sampling frequency of the estimation block were as high as possible. However, the delay line ADC architecture is aimed at high-speed applications. For a given sampling rate, high oversampling ratios result in rather low bandwidth overall performance, undermining the intended advantage of the architecture. Therefore, a delay line ADC implementation should oversample as little as possible in order to achieve the required accuracy. For the purposes of this thesis, 3-bit estimation accuracy is considered adequate, and thus an OSR of 2 is used in all analysis and testing.

4.3 Estimation DAC

4.3.1 Calibration Analysis

Aside from the reconstruction filter, the estimation DAC is the most critical component in the delay line ADC architecture. The reasons for the importance of the DAC were described in Chapter 3.2. First, the inclusion of the reconstruction filter necessitates a calibration process. With the calibration algorithm described in Chapter 3, the portion of the filtered residue waveform due to the estimation signal can be reconstructed from a history of the estimation DAC codes.

Several advantages are gained from this calibration process. First, the accuracy required from the estimation D/A converter is reduced. Consider a conventional N-bit sub-ranging architecture in which the estimation block makes an M-bit estimate. If the estimation stage and the second stage overlap by at least one bit, effectively implementing error correction, it can be shown that the estimation ADC is only required convert the input signal with M-bit accuracy. [6] However, the estimation DAC has to be accurate to the full N-bit accuracy of the sub-ranging converter. This accuracy is required because the value of the estimate signal must be known to full accuracy in order to combine the estimate digital output, as the MSBs, with the second stage digital output, as the LSBs.

In a delay line ADC, in contrast, the estimation ADC only has to be accurate to M bits. As long as the M-bit accurate DAC responds in the same N-bit accurate way to a certain M-bit input code transition, the effect of this transition is known to N-bit accuracy by definition of the calibration process. For example, the DAC need not be N-bit accurate if the input code is '0010'. However, every transition from '0010' to '0111' must be the same to within N-bit accuracy. Therefore, if the DAC response to a transition is identical each time that transition is input to the DAC, a low accuracy DAC can be used in the estimation block without degrading the performance of the delay line A/D converter.

Another potential benefit of the calibration procedure is its use to correct for glitches in the DAC output. Glitches can occur when the switching time of different bits in a binary weighted DAC is unmatched. For a short period of time, a false code could appear at the output. [2] In a current steering DAC, glitches are also created in the DAC output when the current switch emitter-coupled pairs are switched, resulting in harmonic and/or non-harmonic spurs in the output spectrum. The glitch energy is due in part to capacitive coupling of the switch driving voltage through the collector-to-base capacitance to the DAC output. [15]

Ignoring crosstalk between the current sources, these glitch sources should be entirely deterministic. The skew in between current sources is determined by the propagation delay of the input codes to the different current sources around the chip. Also, if the voltage signal that drives the current switches does not vary, the glitch energy that is coupled to the output should be equal for identical transitions since the collector-to-base voltage determines the collector-to-base capacitance. Therefore, if the assumption can be made that glitches are deterministic and not dependent on crosstalk between current sources, they should be corrected for by the calibration scheme. This, however, is a significant assumption and the research presented in this thesis makes no assertion that glitch elimination is successful; it is only a potential advantage.

As mentioned in Chapter 3, the calibration procedure will only function properly if the property of superposition applies to the estimation DAC response. More specifically, the DAC response to a transition from code A_M to code A_N must be equal to the sum of the responses to the individual transitions

$$f(A_M, A_N) = \sum_{i=M}^{i=N-1} f(A_i, A_{i+1}) \tag{4.5}$$

In most current-switched D/A converters, the output voltage is generated by current-to-voltage conversion using an on-chip resistor. Since the current in this resistor is code dependent, significant deterministic "superposition-type" linearity errors may occur due to temperature gradient effects. [16] If the DAC is not completely temperature insensitive, temperature gradients on the die can change the nominal value of the currents that are switched to the output. Thus, superposition requires good stability over operating variations, such as changes in temperature or supply voltage. Clever circuit techniques have been developed to improve the stability of D/A converters across operating point variations. [17, 18] In addition to these techniques, parameter gradient insensitivity, both temperature and process parameters, can be minimized using centroid, double centroid and other layout schemes. [8]

Some level of current source mismatch is inevitable, however. Therefore, the DAC architecture should be chosen to minimize sensitivity to both process and operating point variations. A unary current-steering DAC architecture is the best fit to achieve this goal. In an N-bit unary current D/A converter, there are 2^N total current sources, and the most significant bit switches 2^{N-1} of them, the second most significant bit switches 2^{N-2}, and so on. If the layout is designed to maximize matching between current sources and variation insensitivity, the dynamics of the DAC transitions should also match well. Also, if the dynamics of different DAC transitions match well, the application of superposition in DAC calibration should be valid.

It is less certain that superposition should hold for a binary weighted DAC. In an M-bit binary weighted DAC, there are M current sources. If the LSB current source conducts a current I, the MSB current source would conduct a current $I*2^{M-1}$. Since the layout cannot be optimized using techniques such as common centroid, the M current sources are less likely to match as well as the unary DAC current sources. Therefore, the dynamics of these current sources may not match well, and therefore the applicability of superposition is less clear.

Clearly, a unary implementation of the DAC is a superior design choice. If designed, every effort should be made such that the current sources match as well as possible. Also, circuit design and layout techniques should be used to render the DAC as insensitive as possible to variations in operating conditions and process variations. These efforts are most likely to facilitate the application of the calibration procedure necessary for delay line ADC operation.

4.3.2 Aperture Jitter Analysis

It is argued in Chapter 3 that the aperture jitter requirements in the S/H blocks of the delay line ADC architecture are relaxed. However, the jitter requirements for the estimation D/A converter are very strict. Any aperture jitter in the DAC clock will cause the output of the converter to be skewed in the time domain. This error will propagate through the reconstruction filter to the second stage ADC. Thus, if jitter in the D/A converter causes an error in the residue signal of magnitude greater than an LSB, this error will limit the accuracy of the entire converter. Therefore, it appears that the burden of low clock jitter has been removed from the S/H circuits and transferred to the estimation DAC.

To understand the motivation behind this shift in requirements, the effects of aperture jitter in the S/H and DAC must be examined. First, consider aperture jitter in the S/H circuit. As with a mixing process, a local oscillator multiplies the input signal. In this case, the local oscillator is a sampling clock. Since multiplication in the time domain is convolution in the frequency domain, the spectrum of the sample clock is convolved with the spectrum of the input signal. Also, aperture uncertainty is wideband noise on the clock, and therefore it is wideband noise in the sampled spectrum as well. Since the S/H is, by definition, a sampled system, the spectrum is periodic and repeated around the sample rate. This wideband noise therefore degrades the noise floor performance of the ADC. [19] The theoretical SNR for an aperture jitter limited ADC is

$$SNR = -20\log\left(\sqrt{6}\pi f_{input}\tau_{jitter,RMS}\right) \tag{4.6}$$

Similarly, the clocking of the DAC can be thought of as a sampling process. When the clock makes a transition, the new input code is converted and the DAC output transitions. Equivalently, the DAC input code is sampled. Ideally, this transition is step-like, just like

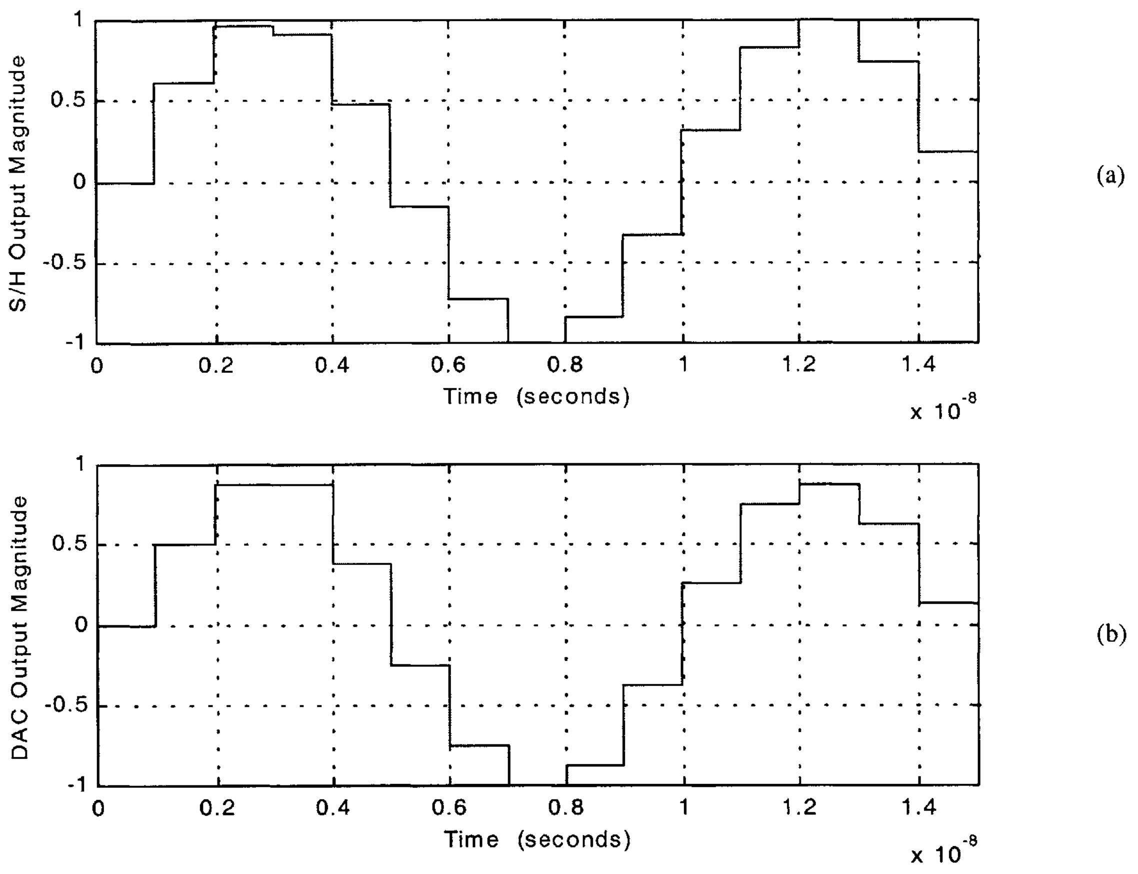


Figure 4.9 Step-like responses of S/H circuit and estimation block to 105MHz sinusoidal inputs. (a) Output of S/H block, clocked at 1GSPS. (b) Output of estimation ADC and DAC block, clocked at 1GSPS.

the output of a S/H circuit. Figure 4.9 shows both a sinusoid after being sampled by a S/H block and also that same sinusoid after being quantized by a 4-bit estimation ADC and DAC. Clearly, both the S/H output and the DAC output are similar. Since the process of clocking the DAC is equivalent to a sampling process, the jitter noise spectrum is periodic and repeated around the sample rate, degrading the noise floor performance.

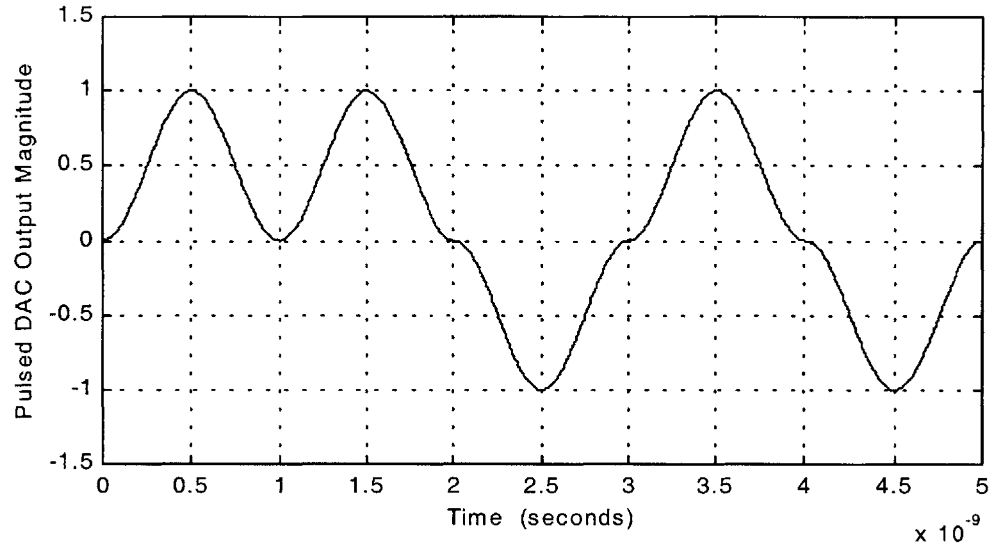


Figure 4.10 Typical pulsed current-steering DAC output waveform 1-Bit DAC shown.

The key issue is that the sampling process is a wideband event. Therefore, the wideband noise associated with aperture jitter will be present in the sampled waveform. In order to reduce the aperture jitter noise, the DAC clocking process must be altered so that it is no longer a wideband event. In order to alter the wideband nature of the DAC clocking process, current-steering waveforms, such as those described in [4], may be used. An example of the DAC output with these shaped current-steering waveforms is shown in Figure 4.10.

The critical idea in the use of these waveforms is that the DAC clock waveform is phase shifted in such a way that the current pulse signal has its minimum value when the output changes it state. Since the rate of change of the current waveforms is zero during the clock instants, the influence of switching transients and aperture jitter is reduced to a minimum. The effect of these shaped current waveforms is a reduction in the bandwidth of the sampled noise. Instead of the wideband sampling process associated with typical DAC transitions, this alternative clocking narrows the noise bandwidth. The noise power contribution from aperture jitter will be significantly reduced, and the noise floor degradation will be less severe. Therefore, a DAC with shaped current waveforms is less sensitive to aperture jitter.

Filter Type	Filter Order	Residue Magnitude (Effective bits)	Residue Derivative (Effective Bits)
Bessel	2	2.3	0.9
Bessel	4	3.3	2.0
Bessel	6	3.6	2.5
Bessel	8	3.6	2.5
Butterworth	2	2.8	1.4
Butterworth	4	4.1	3.5
Butterworth	6	4.4	4.4

Figure 4.11 Performance metrics achieved, using the shaped current waveforms, by various filter types and filter orders. Passband edge set at 0.3 * f_{samp} in each case.

This claim of decreased jitter sensitivity is supported by the research presented in [20]. This paper proposes the use of an alternative pulse as the target for filter design. A widely use class of target pulses have zero crossings at the sampling instants and result in a raised cosine frequency spectrum. However, the alternative pulse proposed results in a triangular spectrum and has zero derivatives at the sampling instants. Having zero derivatives at the sampling instants, the triangular function is shown to have better performance in the presence of jitter than the raised cosine function. Analogously, these results suggest that shaped current waveforms should result in better DAC performance in the presence of aperture jitter.

In order to verify that shaped current waveforms are compatible with the delay line ADC architecture, the simulations need to be modified. Instead of modeling the estimation DAC as an ideal de-quantization block, it is replaced with a block that outputs a shaped current pulse proportional to the input code. The resulting DAC output is similar to the waveforms shown in Figure 4.10, except that the simulated waveforms have 2^8 output levels, corresponding to an 8-bit estimate signal.

The shaped current simulation results are shown in Figure 4.11. In this table, the performance parameters achieved by several different filter types and orders are given. Comparing these results to the results in Figure 4.6, which are the results for a typical

DAC output, it is clear that the shaped current waveforms perform as well as the traditional step-like DAC waveforms. In fact, the shaped waveforms outperform the standard step waveforms for the higher order Butterworth filters. Clearly, the current shaping technique is applicable to the delay line architecture and, if possible, should be used due to its inherent jitter insensitivity properties.

4.3.3 Zero-Order-Hold Effect

In Chapter 3, the zero-order-hold effect of the sampling process was briefly discussed. It was argued that the gain of the zero-order-hold function is not constant across the passband, and therefore the magnitude of the fundamental component of the estimate signal would not equal the magnitude of the input signal. The analysis of this effect is continued in more depth in this section.

The Fourier transform of a zero-order-hold function is a sinc function. The magnitude of the sinc function is only unity for $\omega=0$. The sampling process is the convolution of the input signal with an impulse train (the discretization of the input) and also with a zero-order-hold function (the zero-order-hold effect of the sampling process). In the frequency domain, the result is the multiplication of the input spectrum and the zero-order-hold sinc function at multiples of the sampling frequency. Therefore, the only components of the signal that have unity gain are at multiples of the sampling frequency. All other frequency components will be attenuated by some factor, which is determined by the sinc function. Given that the delay line ADC estimation block oversamples, the input frequency is likely to be well below the sampling frequency. Therefore, the magnitude of the fundamental component of the estimation signal will be smaller than the input signal.

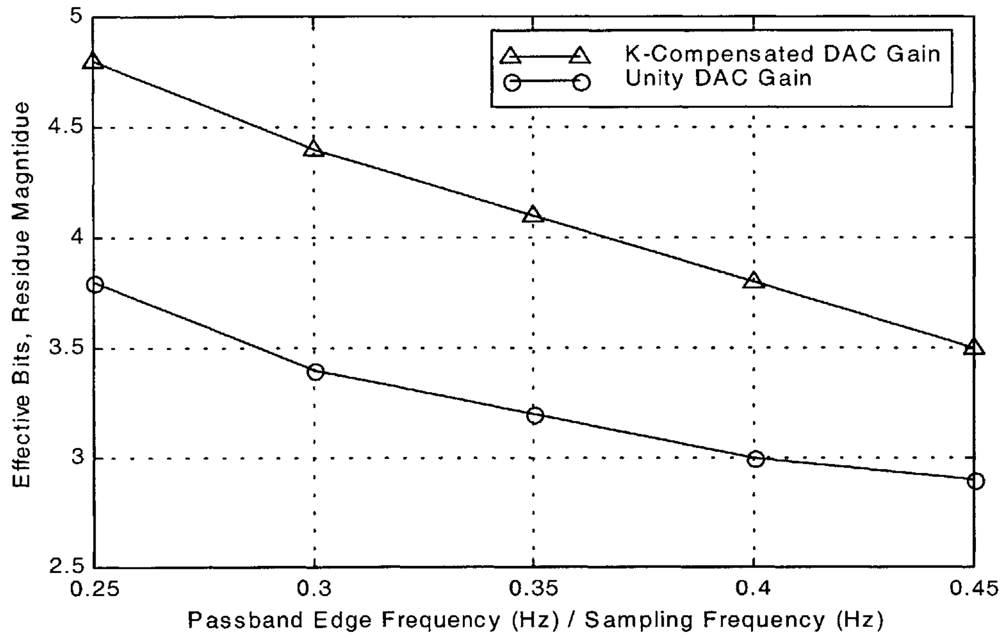


Figure 4.12 Residue magnitude performance metric in effective bits vs. passband edge frequency, normalized to sampling frequency. Both K-compensated DAC gain, for K=0.5*α, and unity gain DAC metrics shown. Reconstruction filter is 4th order Butterworth.

As mentioned in Chapter 3, the error induced by the attenuation of the fundamental can be removed with an inverse sinc filter. Although an ideal inverse sinc filter is not physically realizable, an approximation to this filter could be implemented. However, a much simpler solution is sufficient. Looking at FFT plots of the estimation signal and input signal, the attenuation of the fundamental in the estimation block, α, can easily be calculated. If a gain block of value K equal to the inverse of this attenuation factor is inserted directly after the estimation DAC, the magnitude of the fundamental in the estimate signal should equal the magnitude of the input signal.

While this gain block would remove any fundamental component in the residue signal, it can increase the magnitude of other frequency components in the residue. Therefore, an ideal value for the inserted gain is determined in simulations. The ideal K is empirically found to be

$$K = \frac{1}{2\alpha} \tag{4.7}$$

Figure 4.12 shows the residue magnitude performance metric measured in both unity DAC gain and K-compensated DAC gain simulations. The results for the residue slope performance metric are similar. Clearly, the system performance is greatly enhanced by simple gain compensation. Given such improved performance, an inverse sinc filter is considered unnecessary in order to achieve adequate performance from the delay line ADC architecture.

4.4 Delay Line

One of the less critical components in the delay line A/D architecture, despite being its namesake, is the delay line. As described in Chapter 3.1, the function of the delay line is to match the delay through the estimation block so that the signals at the input to the subtractor are phase aligned. The exact delay required depends exclusively on the implementation of the estimation block. Many parameters, such as latency through the ADC and DAC, and phase differences in the S/H clock and the DAC clock, all need to be taken into consideration when matching the delay line to the estimation network.

The important observation is the relationship between delay mismatch and performance degradation. In order to make this observation, simulations were performed with varying degrees of delay mismatch. In the simulations, the estimation network is composed of ideal quantizers and digital registers. Therefore, there is no need to include latency in the delay matching. The only delay is a 200ps delay in the S/H clock and the DAC clock. This delay was included to demonstrate that the delay line can match any delay through the estimation block, regardless of the cause.

Given this 200ps delay, the ideal time delay through the delay line should be 700ps. The additional 500ps is a result of the fact that the sampling frequency simulated is 1GSPS, and the input waveform should cross the zero-order held estimate signal halfway in between each sampling instant in order to center the residue signal about zero.

Figure 4.13 plots the residue magnitude metric versus time delay through the delay line. Again, the residue slope metric is not shown, but the results are similar to the magnitude metric results. The converter simulated oversamples by a factor of 2 in the estimation block, which samples at 1GSPS, and the DAC gain is not compensated. The reconstruction filter is a 4th order Butterworth filter with a 300MHz passband edge. As expected, the ideal time delay is 700ps. As the delay mismatch increases in either

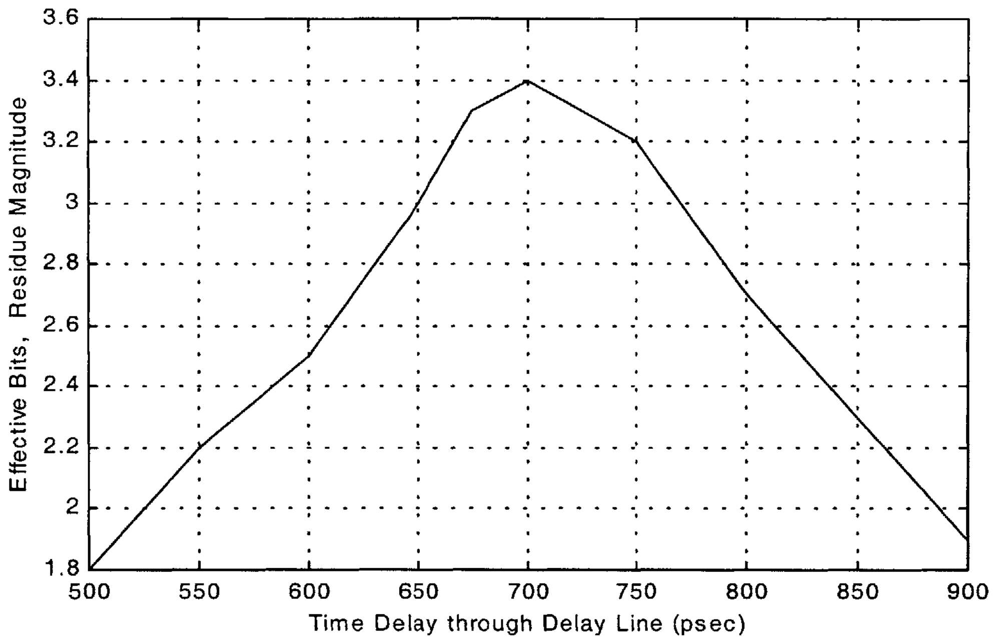


Figure 4.13 Residue magnitude performance metric in effective bits vs. time delay through the delay line. Reconstruction filter is 4th order Butterworth.

direction, the system performance degrades. Also as expected, the performance approximately degrades equally in both directions. The slow degradation around the ideal point suggests that perfect delay matching is not necessary. In fact, for the 4th order Butterworth simulation shown, 3-bit magnitude performance can be achieved with a 50ps delay mismatch in either direction. 3-bit slope performance can also be achieved for that same delay mismatch range.

Another concern in the design of the delay line is the extent to which it approximates an ideal uniform delay. This heavily depends on implementation of the delay line. In a discrete implementation, as is the case for the one described in Chapter 5.2, the delay line can simply be coaxial cable. An advantage of coaxial cable is the ease of synthesizing any time delay required. Coaxial cable has typical characteristics delay times of between 600-800 ps/ft. Therefore, time delays on the order of several nanoseconds are easily realized with several feet of coaxial cable. Also, if the length of the cable is cut accurate to within a fraction of an inch, the desired time delay can be matched with 10 or 20ps.

Another advantage of coaxial cable is its uniform delay over a large range of frequencies. Typical high performance coax cable has phase deviations of several tenths of a degree per foot of cable for frequencies up to 10GHz or higher.

An integrated solution proves to be more challenging. Since on-chip inductors are not available in most processes, a lumped element L-C delay line is not, in most cases, possible. Most applications currently use the Bessel delay approximation. This approximation provides maximally flat group delay. Higher order Bessel filters increase the frequency where group delay falls to 90%. [21] Also, a low power, 4th order active Bessel filter has been fabricated with group delay deviation in the passband frequency of less than 3%. [22] This delay deviation corresponds to matching a 1.5ns delay to within +/-50ps, the performance requirement shown in Figure 4.13. Therefore, an integrated implementation of the delay line ADC would most likely use a Bessel approximation to an ideal delay line.

Chapter 5

TESTING

5.1 Simulation

Before a prototype of the delay line ADC is implemented, a simulation should be performed in order to verify the theoretical functionality of the converter. In contrast to the simulations described in the past chapters, each of which focused on the design tradeoffs involved with a particular aspect of the delay line converter, the simulations described in this chapter focus on the overall functionality of the system. The intention is to verify that the calibration algorithm works, that the delay line does indeed match the reconstructed DAC output, and so on. In the next section, the setup of the simulation is described, and the following section discusses the results.

5.1.1 Setup

The goal of this simulation is to incorporate all the aspects of the delay line converter that were individually analyzed in Chapter 4. As such, the simulated system is much more complicated than the system shown in Figure 4.5. In particular, since the calibration algorithm needs to be tested, the second stage of the converter is included in the system. A block diagram of this new simulated system is shown in Figure 5.1.

Several aspects of this block diagram are worth noting. First, the estimation DAC can function either as an ideal 4-bit quantizer or a 4-bit DAC model. In previous simulations, the DAC was modeled as an 8-bit quantizer so that the DAC accuracy would not limit the accuracy of the residue signal. However, the 4-bit DAC simulated in this chapter will limit the performance of the estimation block to 4-bit accuracy.

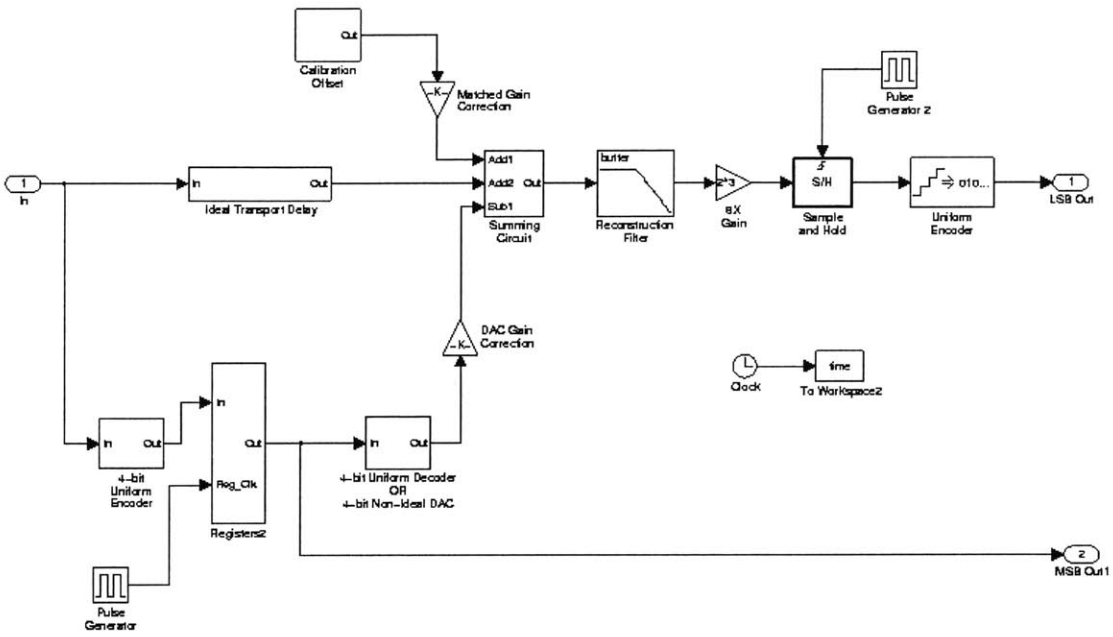


Figure 5.1 Block diagram of complete delay line ADC model simulated.

This 4-bit DAC model is still quite simple. However, it can model several DAC non-idealities that the quantizer cannot model. First, the new DAC model is designed to model clock skew between current sources. This skew is modeled by delaying the switching operation of some current sources in relation to other current sources. During the initialization period of each simulation, each DAC current source is assigned a random delay time, and each DAC will delay its output by that time for the duration of the simulation. Thus, the skew errors are constant during each simulation, but they change from one simulation to the next.

The other DAC non-ideality modeled is mismatch between the current sources. Again, during the initialization period each current source is assigned a magnitude mismatch error between zero and a maximum error limit. Thus, current source 1 might switch 1.50μA while current source 2 switches 1.62μA. These current mismatches, like the time skew errors, are constant for the duration of each simulation but are not necessarily constant over multiple iterations.

Another difference between the simulation shown in Figure 5.1 and those described in Chapter 4, as mentioned earlier, is the inclusion of the delay line ADC second stage. This second stage includes the gain factor of 8, a S/H block, and the 8-bit second stage A/D converter. The gain stage determines the bits of resolution that are gained from the estimation block. Since the gain factor is 2^3, or 8, 3 bits of resolution should be gained from the estimation stage. Also, the second stage ADC sets the accuracy of the calibration procedure. Therefore, the maximum overall resolution that can be achieved by the simulated delay line converter is 8+3 = 11 bits.

It should also be noted that the new simulation includes all of the individual parts investigated in the previous chapter. In Chapter 4, the effect of different filter types and filter orders was investigated, but in the absence of DAC gain correction, delay line mismatch, etc. The effect of delay line mismatch was examined without considering zero-order-hold gain correction or oversampling. For the first time, the simulation described here selects optimal operating conditions and combines them all into a single simulation. Therefore, the simulation is intended not only to verify the operation of the calibration algorithm, but also to do so given any interactions between the design tradeoffs examined in Chapter 4.

Finally, the software implementation of the calibration algorithm should be noted. The exact realization of this algorithm is too detailed to discuss here. It is sufficient to note that the procedure is implemented just as described in Chapter 3.2. The MATLAB code for the prototype calibration is included in Appendix A as a reference, and this code is very similar to the calibration code used in the simulation.

Before the simulation results are presented, the system parameters should be discussed. As described in Chapter 4.1, the 4th order Butterworth filter is considered the filter that optimally balances quick settling and moderate frequency selectivity. Therefore, a 4th order Butterworth filter is modeled in the simulation. Also, the estimation block is setup to oversample by a factor of 2. In Chapter 4.2, this oversampling ratio is shown to be the minimal OSR that achieves acceptable performance. Finally, a zero-order-hold compensating gain of 1.059 is included in the estimation block to improve performance, as described in Chapter 4.3.

The overall delay line system has a sampling rate of 500MSPS. Since the estimation block oversamples by a factor of 2, the sampling rate of the estimation block is 1GSPS. Therefore, the minimum simulation time step size is 1ns. However, the results of the simulation change as the step size varies. It is assumed that this deviation is caused by calculation approximation errors in MATLAB. It is empirically determined that as the simulation step size is decreased, the performance achieved by the simulated converter increases. This increase is consistent with the approximation errors. As the step size is decreased, the behavior of the Butterworth reconstruction filter is more accurately modeled, and the estimation signal reconstructed by the calibration procedure more accurately matches the actual filter estimate signal. For the remainder of this chapter, results will be shown for a 10ps simulation step size, and differing results for other step sizes will occasionally be cited.

5.1.2 Performance Metrics

Before discussing the simulation results, the measures of performance that are used should be defined. There are several ways to characterize the performance of an A/D converter. There are several static measures of performance, such stated resolution, integrated non-linearity (INL), and differential non-linearity (DNL). There are also dynamic measures, such as SNR, SFDR, and noise power ratio (NPR), which are determined by spectral analysis. The analysis in this thesis will primarily use the SNR and SFDR metrics. These two metrics are focused on because dynamic performance is most important for high-speed applications and SNR and SFDR provide a more accurate measure of ADC performance than the stated resolution. In addition, SNR and SFDR are universally accepted performance measures. [3]

For a perfect quantized waveform, the theoretical maximum SNR is the ratio (in dB) of the full-scale analog input root-mean-squared (rms) value to the rms quantization error. Equation 4.4 is an expression for this theoretical limit, with the addition of an oversampling ratio term. In real ADCs, however, there are other noise sources besides quantization noise. These noise sources include thermal noise, reference voltage noise, and clock jitter. SNR is thus computed by taking the ratio of the rms signal to the rms noise. [23]

The exact definition of SNR is not clear. Some literature will define the noise component of the SNR as including all spectral components except the fundamental, the first four harmonics, and the DC offset. [23] Other definitions might exclude the first 7 harmonics, or only the second and third harmonic. To avoid such confusion, the noise measure for SNR in this thesis will contain all spectral components except the fundamental and DC. This is often referred as the signal-to-noise-and-distortion ratio (SINAD).

SFDR is defined as the ratio of the fundamental signal component to the rms value of the next largest spurious component other than DC. Like SNR, SFDR is usually expressed in dB. SFDR is a measure of harmonic distortion, as opposed to wideband noise, which only contributes to the effective noise floor.

Given the SNR and SFDR of a converter, an effective number of bits (ENOB) can be calculated by rearranging the equations that define the theoretical limits of these two metrics [3]

$$ENOB_{SNR} = \frac{SNR(dB) - 1.76}{6.02} \tag{5.1}$$

$$ENOB_{SFDR} = \frac{SFDR(dB)}{6.02} \tag{5.2}$$

If the error sources in the converter contribute negligible noise power compared to the noise power associated with quantization noise, the ENOB_{SNR} will be the same as the stated resolution. However, as noise sources other than the quantization noise become significant and performance suffers, the ENOB will decrease below the stated resolution. Note that it is possible for the $ENOB_{SFDR}$ to be greater than the stated resolution.

5.1.3 Results

Figure 5.2 shows a spectral analysis of simulation results for a simulation of the system shown in Figure 5.1 with a time step size of 10ps. In this simulation, the DAC model simulated is ideal, meaning that is does not include any source magnitude mismatch and or clock time skew. As described in Chapter 3.2, the output of the reconstruction filter, combined with a known set of calibration DAC input codes, is used to create a calibration table. This table is then used to reconstruct the filtered estimate signal. Combining this signal with the output of the second stage (while not in calibration mode), an 11-bit approximation of the input signal is generated. The result of an FFT of this 11-bit signal is the data shown in Figure 5.2.

The results shown correspond to a SFDR of 79.6dB. The SNR, as described in section 1.2 of this chapter, is calculated to be 63.6dB. The resulting $ENOB_{SNR}$ is 10.3 bits. Given that the resolution of end result of the simulated converter is 11 bits, the simulated converter performs about as well as can be expected.

Of course this simulation models the building blocks as ideal components. Therefore, this close-to-11-bit performance represents the idealized limit on the performance achieved. As real components are substituted into model in place of the ideal building blocks, the performance is expected to degrade. However, this idealized simulation verifies the conceptual integrity of the calibration algorithm as well as the overall delay line ADC system. It also shows that the resolution achievable is limited by the accuracy of the individual components, not by inherent limits with the architecture itself.

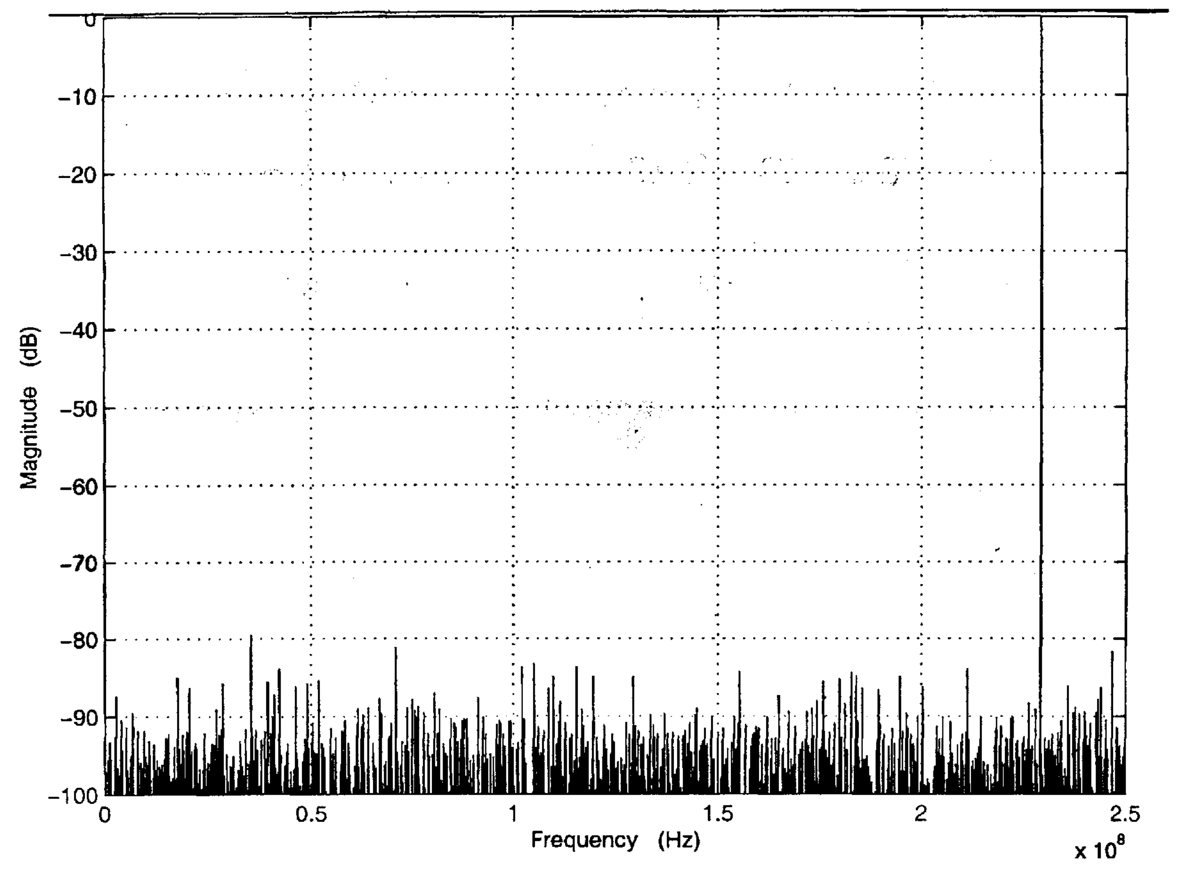


Figure 5.2 Spectral analysis of idealized delay line ADC simulation.

It was mentioned previously in this Chapter that the step size of the simulation affects the results of the simulation. While these effects do not dramatically alter the measured performance, they are worth noting. A simulation identical to the one discussed earlier is performed, only with a smaller, 2ps step size. This simulation, of course, takes significantly longer to run than the previous simulation. The results, however, are somewhat improved. The FFT spectral plot is not included here, but the calculated SNR is 64.4dB and the calculated SFDR is 80.4dB. Thus, decreasing the step size by a factor of 5 increased both the SNR and the SFDR by 0.8dB, or 0.13 effective bits. It is expected that as the step size is further reduced, the performance metrics measured will approach their 11-bit theoretical limits. Therefore, it can be inferred that the performance limit of the ideal delay line architecture is actually 11 effective bits.

5.2 Implementation

In order to verify to validity of the delay line ADC architecture beyond idealized simulations, a prototype delay line A/D converter is implemented. This test converter is implemented using off-the-shelf commercial components, as opposed to a custom integrated design. The purpose of this prototype is not to press the performance limits of the delay line ADC. Instead, the implementation is only intended to verify functionality of the concept. The performance of this prototype is more likely to be limited by the performance capabilities of the components than the architecture itself. However, if the prototype can be shown to perform the basic functionality of the delay line architecture, the validity of the concepts will be reinforced, and a more performance based implementation could be pursued as further research.

5.2.1 Design Choices

Clearly the delay line ADC is intended for high-speed applications. However, given that the prototype is only meant to test functionality, there is not a need to push the performance goals of the prototype to very high speeds. Moreover, the speed that can be obtained by the test converter is limited by the speed of the parts with which it is built. In fact, the design specifications of the delay line converter prototype are primarily driven by the availability of components.

Given that the estimation block oversamples by a factor of 2, the operating speeds of the components in the estimation block are assumed to be limiting factors in achievable speeds. After a search of commercial suppliers, the estimation DAC is determined to be the performance-limiting component. The highest speed D/A converter found is the TDA8776A by Philips Semiconductors. This converter is a 1000MSPS, 10-bit D/A converter. It is specified to achieve about 60dB SFDR for input signals at 100MHz.

Note that there is no need for the estimation DAC to be accurate to 10 bits. However, no lower resolution converters that operated at the same speed were found.

Given an estimation DAC that can operate at 1000MSPS, the requirements for the estimation ADC is 4-bit accuracy and a high (1000MSPS to match the DAC) sampling rate. The SPT7610 is a 6-bit, 1000MSPS fully flash A/D converter that fits the TDA8776A nicely. The specifications for the SPT7160 cite a 36dB typical SNR for a 250MHz input sinusoid, which corresponds to $ENOB_{SNR}$ = 5.6 bits. This accuracy far exceeds the requirements for the 4-bit estimation block that will be implemented. Unfortunately, the SPT7160 demultiplexes its digital output so that there are two output buffers, each of which updates at 500MSPS. In order to avoid the complexity of high-speed multiplexing and any errors that might be introduced into the estimation block, only one of the SPT7610 output banks is used. Therefore, the estimation stage operates at a 500MSPS rate. Given 2X oversampling, this sets the conversion rate of the overall system to 250MSPS.

Once the overall system conversion rate is set, the second stage ADC may be selected. In this case, this converter must support a 250MSPS conversion rate. There is no restriction necessary on the accuracy of this ADC. However, if an 8-bit second stage is to be implemented (in order to match simulations), an 8-bit second stage converter must be used. Therefore, the SPT7721 is selected. This 8-bit converter specifies a minimum conversion rate of 250MSPS and a typical $ENOB_{SNR}$ of 6.4 bits. However, this effective number of bits is typical. The minimum specified $ENOB_{SNR}$ is only 5.8, well below the 8-bit performance expected. Also, this metric is only specified for an input frequency of 70MHz. It might be expected that performance will degrade for higher input frequency. Since the delay line converter prototype is a 250MSPS Nyquist rate converter, the input frequencies could range up to 125MHz. Therefore, the SPT7721 is expected to degrade the performance of the prototype converter significantly. However, no other A/D converter could be found with superior AC performance at such high sampling frequencies.

The final components that must be selected are those necessary to implement the summing block and the gain block. The requirements on these components are quite strict. Amplifiers used to implement these blocks must be able to support modest gain with a bandwidth of greater than 100MHz. Therefore, the gain-bandwidth product must be several hundreds of megaHertz. An operational amplifier topology that boasts high gain-bandwidth products is the transimpedance, or current-mode feedback, amplifier. Therefore, the CLC449 current feedback operation amplifier is selected to implement the summing and gain blocks. This amplifier specifies a gain-bandwidth product of 1.1GHz. Also, the amplifier is unity gain stable, a necessity for the implementation of the single-ended-to-differential conversion in the summing block, as will be described in the next section.

The disadvantage of using these amplifiers is the harmonic distortion characteristics of the CLC449. The 2nd harmonic distortion is specified to be less than –40dB for an input frequency of 50MHz, and the 3rd harmonic distortion is specified to be less than –55dB for the same input. Worse yet, the input frequencies for the test converter could be higher than 50MHz, making the distortion problem even worse. Given that –40dB harmonic distortion could limit the performance of the second stage to 6.5 SFDR bits, and even fewer SNR bits, the CLC449 gain block is likely to also significantly degrade the performance of the implemented prototype converter.

5.2.2 Circuit Detail

Having decided on the component building blocks for the test converter, a full test circuit can now be described. A schematic of the implemented converter is shown in Figure 5.3. It is clear that this schematic closely mirrors the block diagram shown in Figure 5.1. Several differences should be noted. First, the DAC gain correction is not included in the test converter that is built. As noted in Chapter 4, correcting for the zero-order-hold error is not required for proper operation of the delay line converter, it only improves the theoretical performance. However, in this implemented converter, gain correction would likely do more bad than good. The reason is the harmonic distortion of the CLC449

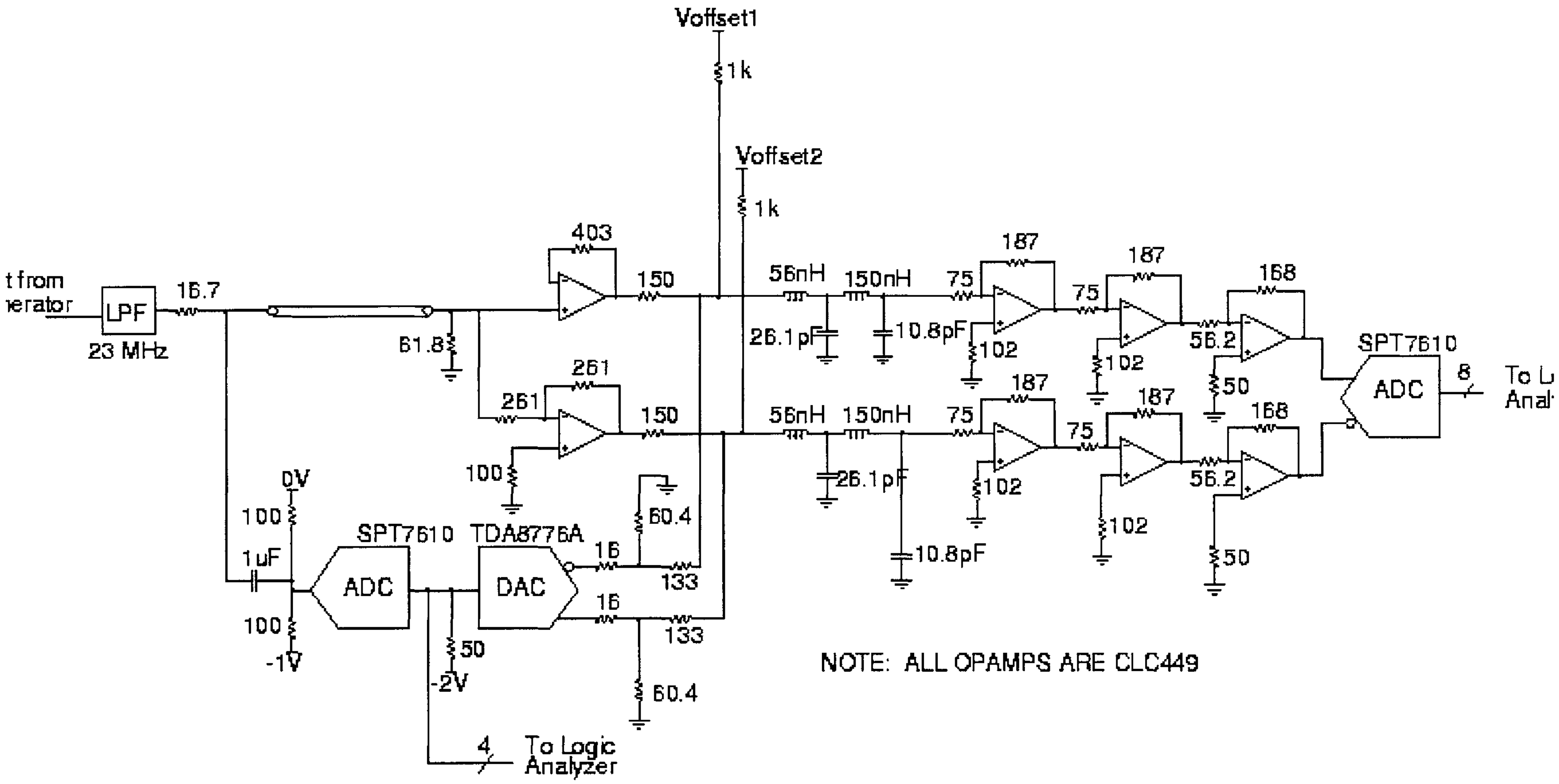


Figure 5.3 Circuit schematic of prototype delay line ADC implemented.

amplifiers used to realize gain blocks. This distortion could be particularly bad if the input to the gain stage contained high frequency components, as does the DAC output. Therefore, zero-order-hold gain correction is not considered necessary and is omitted from the implemented converter.

Another detail worth mentioning is the single-ended-to-differential conversion at the end of the coaxial delay line. Since the output of the TDA8776A is differential, some conversion needed to be done in order to combine the delayed input and the estimation block output, either single-ended-to-differential or vice versa. Converting the DAC output to a single-ended signal and passing that through to the second stage would reduce the parts count in the second stage. However, there is an important advantage to differential signals. Differential signals reject any noise that is equally contributed to both sides of the signal. If the second stage is laid out properly, the assumption that noise adds equally to both channels should be valid. Therefore, non-idealities such as signal pickup should be dramatically reduced.

This reduction is very important in the implemented converter. High frequency clock signals up to 1GHz are used in the converter. Relatively short wires, on the order of inches, can pick up radiated signals at these frequencies. Pickup of the 250MHz clock signals can be particularly bad due to the layout of the converter. Therefore, the noise rejection property of differential signals becomes extremely valuable.

Also note that none of the power supplies are shown in Figure 5.3 in an attempt to keep the schematic from becoming too cluttered. Also, bypassing circuitry and other biasing networks are not shown, either. For completeness, all of this circuitry for both ADCs and the estimation DAC is included in the schematics in Appendix B.

In regards to biasing, it should be mentioned that the bypassing and biasing details for the TDA8776A and SPT7721 are taken care of by the evaluation boards that are used. These boards are used for simplicity of implementation. These evaluation boards provide significant functionality, in addition to bypassing and biasing, such as generating –2V

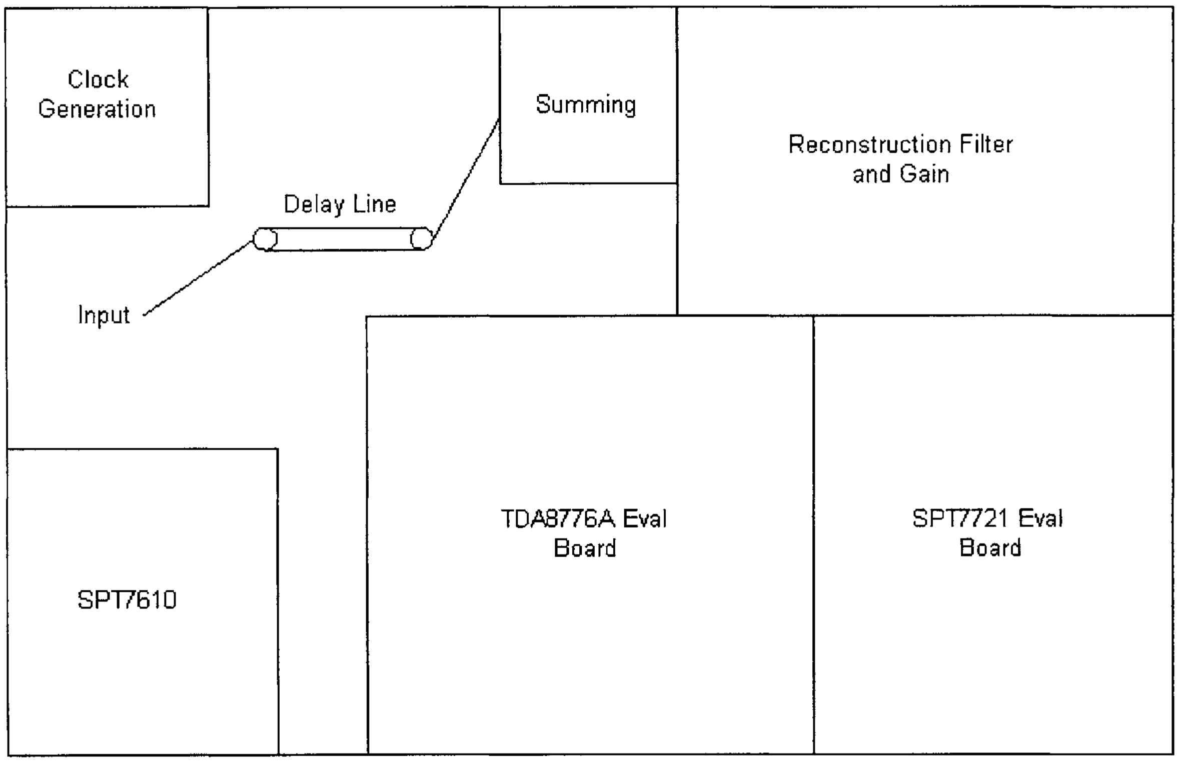


Figure 5.4 Layout floor plan of prototype test board.

supplies for the termination of ECL level signals. The TDA8776A evaluation board also includes the input registers necessary to clock the DAC. Also, the SPT7721 evaluation board includes output data buffers and clock receivers. Again, for completeness, schematics from the data sheets for the DAC evaluation board is included in Appendix B.

Although an evaluation board for the SPT7610 also exists, it is not used. The reason this board is not used is the lack of flexibility in operation. The evaluation board includes functional blocks to perform undesired operations, such as data decimation for easier capture and reconstruction D/A conversion for use with an oscilloscope. However, these operations are not necessarily easy to avoid. Therefore, the board is not used and the schematic shown in Appendix B is realized from discrete components by hand assembly.

A floor plan depicting the layout of the entire converter board is shown in Figure 5.4. This floor plan is included to make clearer the photograph showing the layout of the entire test circuit, which is shown in Figure 5.5. In this picture, the signal propagates

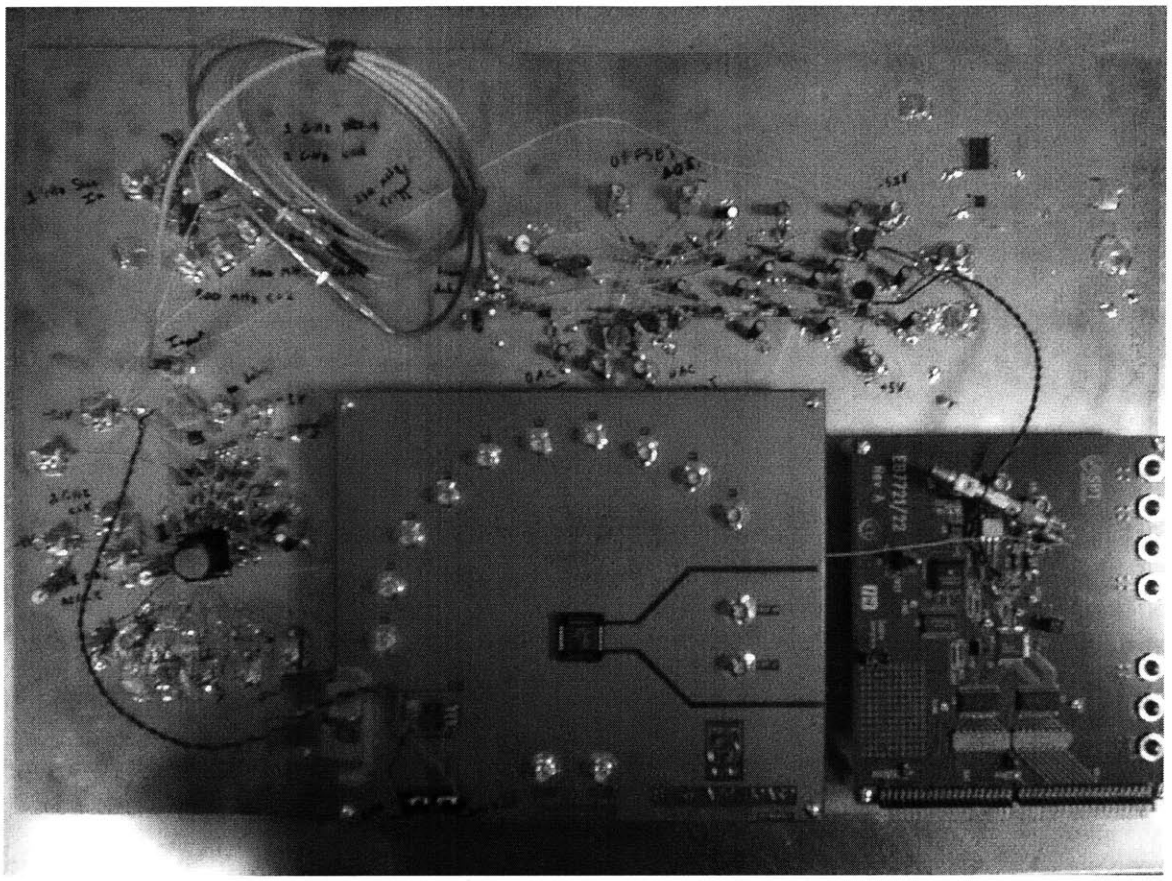


Figure 5.5 Photograph of prototype test board.

from left to right. The evaluation boards for the SPT 7721 is in the lower right hand corner, and the evaluation board for TDA8776A is directly to the left of it. To

the left of DAC evaluation board is the SPT7610 and its supporting circuitry. The input signal is coupled onto the board just above the SPT7610. There it splits to the estimation ADC and also through the coaxial delay line to the single-ended-to-differential conversion stage. This stage, along with the summing node, filter, and gain stage that follows, can be seen above the two evaluations boards. Finally, the circuit blocks in the upper left hand corner of the board are clock generation blocks, which will be briefly described in the following section.

5.2.3 Test Setup

In order to test the performance of the prototype delay line A/D converter, a test bed is developed. All of the bias voltages necessary for the operation of the delay line converter are independently generated by 9 high precision DC power supplies. These bias voltages include the power supplies for the various components (+5V, +6.1V, -5.2V, and +3.3V), an ECL termination bias (-2V), reference voltages for the SPT7610 (-1V and –0.5V), and finally a pair of DC offset voltages, which are used during calibration to center each DAC transition in the common mode input range of the SPT7721. The need for such a large number of different power supplies is simply due to the piecing together of this test converter from several different components by several different manufacturers. If the delay line converter were designed from the ground up in an integrated package, the converter could definitely be designed much more neatly to operate from single or dual supplies.

The waveform synthesizers generate sinusoids for the delay line converter. The high-speed synthesizer (HP8665B) generates a 1GHz clock signal. This clock signal is then divided on the prototype board by MC10EP16 and MC10EP139 chips in order to generate the 500MHz and 250MHz clocks needed for operation.

In order to perform the calibration procedure, equipment to generate DAC calibration patterns is needed. The HP80000 digital word generator is able to generate DAC input codes in the pattern required for calibration. Also, the output of this generator switches faster than the 500MHz estimation block. Therefore, calibration data should not be corrupted by transitions that have completed by the time data is captured.

5.2.4 Results

The prototype delay line ADC described in the previous sectioned is built and tested at MIT Lincoln Laboratory. For the results that will be discussed, the input signal to the

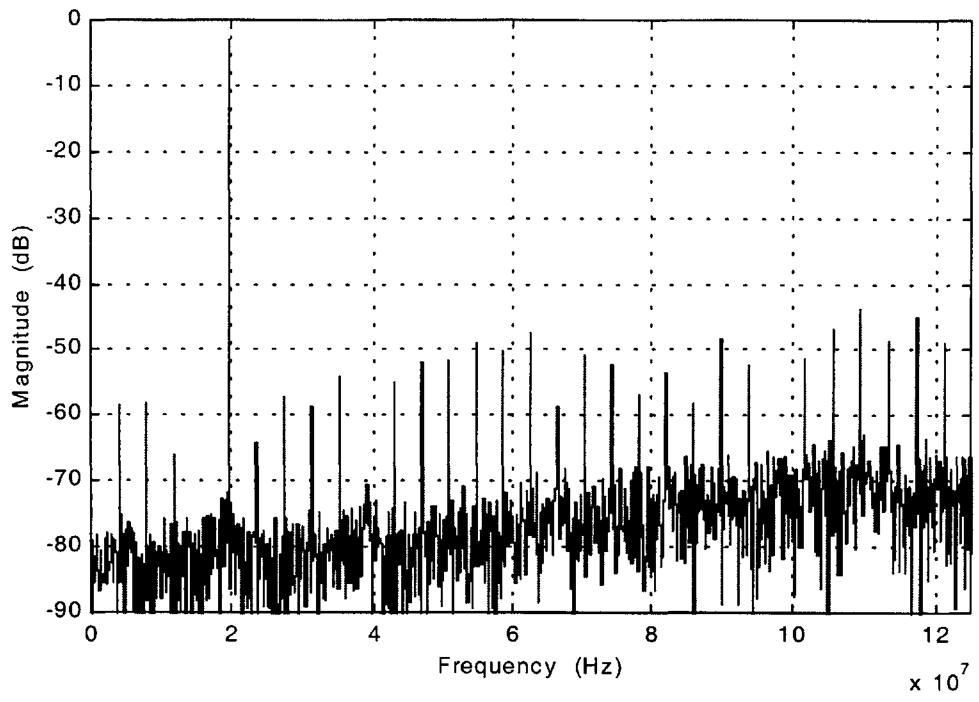


Figure 5.6 Spectral analysis of final implemented delay line converter output.

converter is a 19.53125MHz sinusoid. The magnitude of the input signal is slightly smaller than the full-scale capability of the estimation stage ADC (-2dB) in order to avoid overdriving of the estimation stage. The calibration data is collected by hand, as opposed to an auto calibration procedure. The response of the DAC and filter to each input code transition is measured separately over the course of an hour. In order for the calibration scheme to properly function, it must be assumed that operating conditions, such as temperature, do not change during the calibration period or while collecting test data. Once all of the calibration data and test data are collected, this data is stored off line and analyzed in software.

The results of the software analysis for the test described above are shown in Figure 5.6. The FFT plot shown is a spectral analysis of the final delay line ADC output. This output is composed of both the measured residue signal and the software constructed estimate-after-filtering signal. In an ideal implementation, the analysis shown in Figure 5.6 would suggest 11-bit performance.

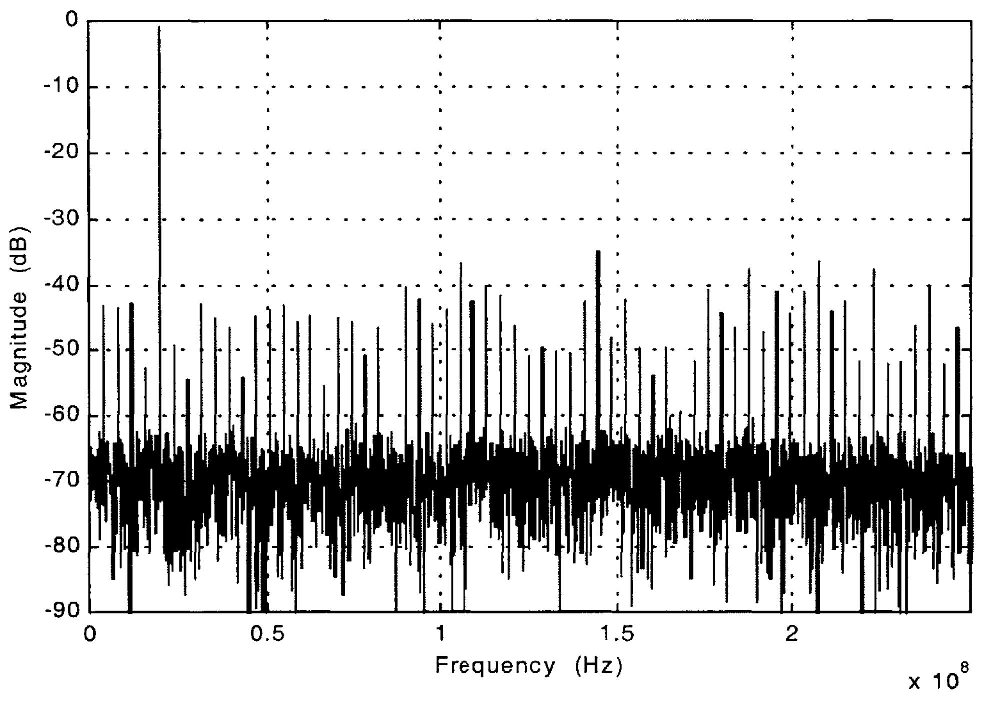


Figure 5.7 Spectral analysis of raw estimation ADC output.

Clearly, Figure 5.6 does not show 11-bit performance for the output signal. The SNR bits calculated from this FFT plot are 5.6, while the SFDR is 40.4dB. For a typical 11-bit A/D converter, acceptable SNR bits might range down to 8 or 9 bits. Perhaps a better measure of the performance of the delay line converter, however, is the effective SNR bits added to the estimation stage ADC. If no accuracy, as determined by the SNR, is added to the estimation signal, the second stage of the delay line converter and the calibration process are not properly functioning.

An FFT plot of the raw 4-bit digital estimate signal (raw meaning not calibrated) is shown in Figure 5.7. The SNR of this signal is 24dB, which corresponds to 3.7 effective bits. Therefore, the calibration data and the digitized residue signal combine to add 2 effective bits of performance to the estimation stage ADC. While this performance enhancement is significantly less than the desired 8-bit improvement, it does verify basic functionality of the delay line architecture and the calibration algorithm.

A more important observation that should be made in Figure 5.7 is the presence of spurs. These spurs occur every 3.9MHz throughout the entire analyzed spectrum. Due to the large magnitude of these signals, it is unlikely that the spurs are aliased high order harmonics. Instead, the spurs are most likely due to an error in conversion performed by the estimation stage ADC. Whatever the cause of these spurs, the key observation is that these spurs are in fact the limiting factor in the dynamic performance of the estimation ADC.

Referring back to Figure 5.6, these same spurs are present in the spectral plot of the output signal. Because the estimation ADC is not part of the calibration loop, errors such as distortion in the estimation ADC are not accounted for during calibration. However, this distortion should be present in the residue signal and therefore accounted for in the final software output. Since these spurs are present in the final software output, it is a possible that they are in fact part of the input signal. While these spurs do limit the overall measured performance, they are not inherently an error in the delay line architecture. If these spurs were removed from the spectral data in Figure 5.6, the SNR would be much closer to the apparent noise floor, which is approximately –80dB. In this case, the SNR would be far superior, resulting in 9 or 10 effective SNR bits.

Another observation can be made regarding the apparent noise floor in Figure 5.6. At low frequencies, around the fundamental component of the spectrum, the noise floor appears to be about –80dB. However, at higher frequencies, the noise floor is approximately 10dB higher. In contrast, consider the spectral data shown in Figure 5.8. This data is an FFT analysis of the software constructed estimate-after-filtering signal. In this plot, the noise floor is constant across the spectrum. Also, the magnitude of the noise floor in Figure 5.8 is the same as the magnitude of the noise floor at high frequencies in Figure 5.6.

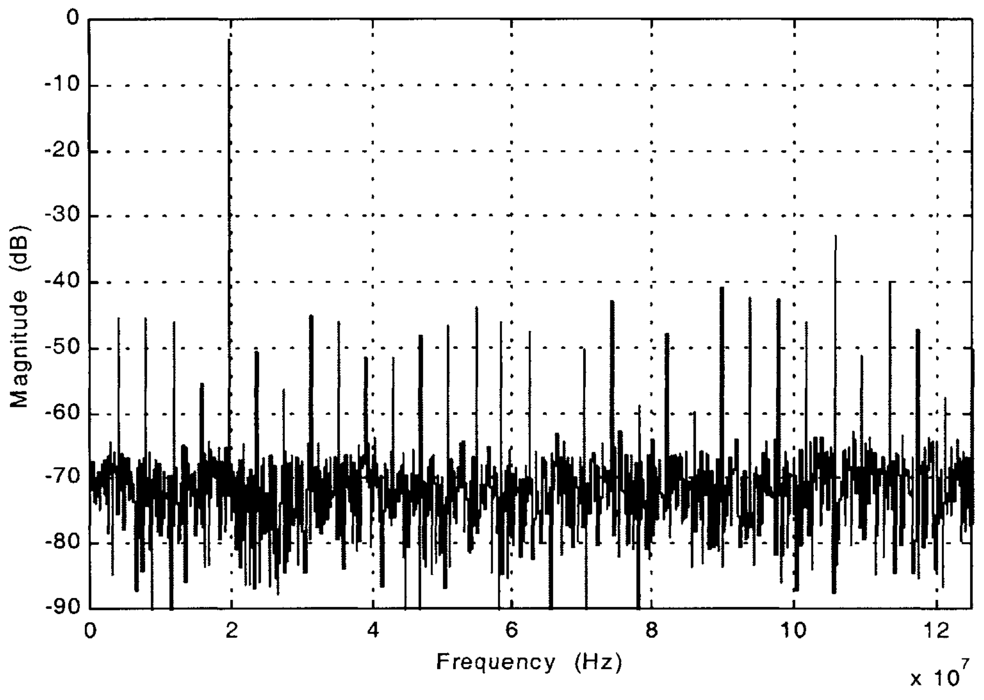


Figure 5.8 Spectral analysis of software reconstructed estimate-after-filtering signal.

Since the data shown in Figure 5.6 reflects the signal used for Figure 5.8 combined with the digitized residue signal, this observation suggests that including the residue signal only improves the spectral performance of the delay line converter at low frequencies. This dependency on the frequency components of the residue signal can be attributed to the poor distortion characteristics of the CLC449 amplifiers. At frequencies at or above 100MHz, the 2nd harmonic distortion of a CLC449 is specified to be as bad as –40dB. Therefore, the high frequency components of the residue signal digitized by the second stage ADC may be significant distorted and is not necessarily an accurate representation of the difference between the delayed input signal and the estimation signal.

Due the two performance limitations described above, the poor performance of the implemented delay line converter is not attributed to problems in the architecture or the calibration. In fact, the near 10-bit performance predicted by the apparent noise floor in Figure 5.6 suggests that the delay line ADC architecture is capable of the full performance predicted in simulations.

Chapter 6

CONCLUSION

The performance demonstrated in both the ideal simulations and also the test implementation validates the conceptual operation of the delay line ADC architecture. The simulations test to theoretical limit of delay line converter performance and show that these limits are the same as the limits of an equivalent-resolution conventional converter. The implementation of the prototype converter confirms that the architecture concepts, most specifically the calibration procedure, do in fact function properly with real components.

The demonstration of functionality achieved suggests that the delay line ADC architecture works, but it does not experimentally explore the performance limits that can be achieved. Therefore, a logical continuation of this work would involve high-speed implementation and testing of the delay line architecture.

A high-speed version of the delay line ADC would most likely be an integrated solution. There are several advantages to integrating the delay line converter. First, the entire converter would be included in a single package. While not a requirement, the delay line converter is intended to function as an autonomous system. Therefore, including the entire converter in one small, integrated package would define the converter as a building block and allow easy integration into other systems.

Another related advantage of integrating the converter is the ability to tailor the building blocks to the system requirements. In Chapter 5, the components used to implement the test converter were discussed. The estimation ADC is a 6-bit, 1GSPS converter, even though only 4-bits of one of the two output data banks were used. Since the SPT7610 is a fully flash converter, paying for two unused bits of performance is particularly wasteful. A 10-bit, 1GSPS DAC is used in the estimation block, even though only 4-bits are used and it is clocked at 500MSPS. Clearly, there is a lot of wasted available performance in the test converter. This wasted performance comes at the price of power. In the test setup described in Chapter 5, the prototype converter consumes over 8 watts of power. The SPT7610 itself consumes almost 4 watts of power. This is unacceptably high power consumption, even for an 11-bit high-speed converter. In an integrated version of the delay line architecture, this power consumption could be drastically reduced by clever circuit design. For example, the estimation block components would only be accurate to 4-bits. They would also not need to support conversion speeds faster than the sampling rate of the estimation block.

More important than the portability and the power consumption of the delay line converter are the circuit optimizations that could be implemented in an integrated delay line converter solution. Specifically, designing each of the building blocks would allow the estimation DAC to implement a shaped current pulse output. As described in Chapter 4, this optimization should render the DAC relatively insensitive to aperture jitter. Since aperture jitter is the most significant performance limiting error source at high speeds, the shaped current output optimization is considered necessary in order to test the conversion rate limits of the delay line architecture.

Designing an integrated solution described above is a daunting task. Perhaps a logical in-between implementation step would be integration of the estimation block in a single package. This chip could be placed in a board along with the other necessary architecture components, such as the second stage blocks and the delay line.

There are several advantages to integrating only the estimation block. First, the evolution of the delay converter towards a single, monolithic implementation is not attempted in a single step. Multiple levels in the progression towards a monolithic implementation are most certain to produces smaller, more manageable errors that must be corrected. Also, an estimation-block-only chip would not require the implementation of the delay line on-chip. While Chapter 4 discussed integrated solutions for the delay line, it is expected that simple Bessel filter design might not adequately meet the requirements of the delay line. An integrated delay line is expected to take a significant amount of research and design consideration. Therefore, by separating the delay line from the integrated estimation block, the crucial but difficult design tasks of implementing the delay line and also implementing the estimation DAC with a shaped current output can also be separated.

Given all of the implementation goals described in this chapter, a direction for future work with the delay line converter architecture is outlined. The step-by-step development of the converter from the test converter described in Chapter 5 to a fully integrated, high-performance solution is critical. This step-wise development will allow adequate research and design work to go into each component. Thus, the final result will be able to test how fast this architecture truly can be.

REFERENCES

[1] Brannon, Brad and Cloninger, Chris. "Redefining the Role of ADCs in Wireless." *Applied Microwave and Wireless*. March, 2001.

[2] Gustavsson, Mikael et al. *CMOS Data Converters for Communications*. Kulwer Academic Publishers, London, 2000.

[3] Walden, Robert H. "Analog-to-Digital Converter Survey and Analysis." IEEE Journal on Selected Areas in Communications, Vol 17, April 1999, pp. 539-550.

[4] Splett, Armin; Dreßler, Hans-Joachim; Fuchs, Armin; Hofmann, Ralf; Jelonnek, Björn; Kling, Helmut; Schultheiß, Anton. "Solutions for Highly Integrated Future Generation Software Radio Basestation Transceiver." IEEE 2001 Custom Integrated Circuits Conference, pp. 511-518.

[5] Petschacher, Reinhard, et al. "A 10-b 75-MSPS Subranging A/D Converter with Integrated Sample and Hold." IEEE Journal of Solid-State Circuits, Vol. 25, No. 6, December 1990, pp.1339-1346.

[6] Johns, David and Martin, Ken. *Analog Integrated Circuit Design*. John Wiley and Sons, Inc., NewYork, 1997.

[7] Wikner, J. Jacob. "Simulation and Implementation Comparison of Two 3V – 5V 14-bit Current-Steering Wideband CMOS DACs." Advanced A/D and D/A Conversion Techniques and their Applications, July 1999 Conference Publication No. 466, pp. 130-133.

[8] Borremans, M., Van den Bosch, A., Steyaert, M, and Sansen, W. "A Low Power, 10-bit CMOS D/A Converter for High Speed Applications." IEEE 2001 Custom Integrated Circuits Conference, pp. 157-160.

[9] Van den Bosch, Anne; Steyaert, Michiel; Sansen, Willy. "The Impact of Statistical Design on the Area of High Performance CMOS Current-Steering D/A Converters." Analog Integrated Circuits and Signal Processing. Kulwer Academic Publishers, 2001, pp.141-148.

[10] Colleran, William T. and Abidi, A. A. "A 10-b, 75-MHz Two-Stage Pipelined Bipolar A/D Converter." IEEE Journal of Solid-State Circuits, Vol. 28, No. 12, December 1993, pp.1187-1199.

[11] Shinagawa, M; Akazawa, Y.; Wakimoto, T. "Jitter Analysis of High-Speed Sampling Systems", *IEEE Journal of Solid-State Circuits*, Vol. 25, No. 1, Feb. 1990

[12] "Understanding Flash ADCs". Maxim Integrated Products Applications Notes, September, 2001. http://dbserv.maxim-ic.com/tarticle/view_article.cfm?article_id=810.

[13] Oppenheim, Alan V. and Willsky, Alan S. *Signals and Systems.* Prentice Hall, New Jersey, 1997.

[14] Paarmann, Larry D. *Design and Analysis of Analog Filters.* Kulwer Academic Publishers, London, 2001.

[15] Tesch, Bruce J. and Garcia, Juan C. "A Low Glitch 14-b 100-MHz D/A Converter." IEEE Journal of Solid-State Circuits, Vol. 32, No. 9, September 1997, pp.1465-1469.

[16] Conroy, Cormac G.; Lane, William A.; Moran, Michael A. "Statistical Design Tehcniques for D/A Converters. IEEE Journal of Solid-State Circuits, Vol. 24, No. 4, August 1989, pp. 1118-1128.

[17] Mojal, M. "Multibit DAC with corrective gate to drain voltage for optimum matching under Gradient Temperature effects." 1998 IEEE International Conference on Electronics, Circuits and Systems, Vol. 1, pp. 139-142.

[18] Henriques, Bernardo G.; Kananen, Kari; Franca, José E.; Rapeli, Juha. "A 10 Bit Low-Power CMOS D/A Converter with On-Chip Gain Error Correction." IEEE 1995 Custom Integrated Circuit Conference, pp. 215-218.

[19] Brannon, Brad. "Aperture Uncertainty and ADC System Performance." Application Note AN-501, Analog Devices, Norwood, Massachusetts, 2000.

[20] Mneina, S and Martens G.O. "Filter design targets with reduced jitter effect." 2001 IEEE Pacific Rim Conference on Communications, Computers, and signal Processing, Vol. 1, pp. 172-175.

[21] Capstick, M.H. and Fidler, J.K. "Delay approximation for synchronous filter topologies." IEE Proceedings on Circuits, Devices and Systems, Vol. 148, No. 3, June 2001, pp. 109-114.

[22] Jiang, Jin-guang; He, Yi-gang; Wu, Jie. "Design of Fully Differential Fourth-order Bessel Filter with Accurate Group Delay." Proceedings of 4th International Conference on ASIC, 2001, pp. 281-284.

[23] "Glossary of Frequently Used High-Speed Data Converter Terms." Maxim Integrated Products Application Notes, March 2001. http://dbserv.maxim-ic.com/tarticle/view_article.cfm?article_id=123.

APPENDICIES

Appendix A

MATLAB code for software analysis of test data.

```
function [sync_cal_data,offset_cal_data,final_values,temp,temp2,total]=build_cal();

% Files should go in this order: cal1_sync, cal1_offs, cal2_sync, cal2_offs, etc...
filenames =
['cal01_sync.txt';'cal01_offs.txt';'cal02_sync.txt';'cal02_offs.txt';'cal03_sync.txt';'cal03_offs.txt';'cal04_sync.
txt';'cal04_offs.txt';'cal05_sync.txt';'cal05_offs.txt';'cal06_sync.txt';'cal06_offs.txt';'cal07_sync.txt';'cal07_of
fs.txt';'cal08_sync.txt';'cal08_offs.txt';'cal09_sync.txt';'cal09_offs.txt';'cal10_sync.txt';'cal10_offs.txt';'cal11_
sync.txt';'cal11_offs.txt';'cal12_sync.txt';'cal12_offs.txt';'cal13_sync.txt';'cal13_offs.txt';'cal14_sync.txt';'cal
14_offs.txt';'cal15_sync.txt';'cal15_offs.txt'];
[a,b]=size(filenames);
num_cals = 10;          % Number of times each DAC step is measured (for redundancy).  All are averaged
together.
num_steps = 22;         % Number of 500MHz (2ns) steps that each DAC step is measured (longer = more
settled).
latency = 4;            % Number of 500MHz (2ns) cycles between input and when it is ready at output
sync_cal_data = zeros(30,num_steps);
offset_cal_data = zeros(30, num_steps);
x=0;
y=0;
error_correction_steps=2;

% for each file, open it up and start reading lines from it.
for i = 1 : 30
  fid = fopen(filenames(i,:));
  test = 1;
  counter = 1;
  rise_starts = zeros(1,10);
  num_rises = 0;
  fall_starts = zeros(1,10);
  num_falls = 0;

  % while the test variable is true, keep reading lines.
  while (test==1)
    tline = fgetl(fid);
    if (tline == -1)
      % tline = -1 indicates the end of the file.  stop looping.
      test = 0;
      disp(['Ran out of data in file ' filenames(i,:)]);
    elseif (num_rises>num_cals & num_falls>num_cals)
      % if we have reached the required number of rises and falls to be averaged together, stop looping.
      test = 0;
      disp(['Finished gathering data from file ' filenames(i,:)]);
      counter
    else
      % convert the data line string to numbers and parse it up into all it's elements.
      data = str2num(tline);
      F7B(counter) = data(1);
      F6B(counter) = data(2);
      F5B(counter) = data(3);
      F4B(counter) = data(4);
      F3B(counter) = data(5);
      F2B(counter) = data(6);
      F1B(counter) = data(7);
      F0B(counter) = data(8);
      F7A(counter) = data(9);
      F6A(counter) = data(10);
      F5A(counter) = data(11);
      F4A(counter) = data(12);
      F3A(counter) = data(13);
      F2A(counter) = data(14);
      F1A(counter) = data(15);
      F0A(counter) = data(16);
      CReady(counter) = data(20);
      C3(counter) = data(21);
      C2(counter) = data(22);
      C1(counter) = data(23);
      C0(counter) = data(24);
      DCLKOUT(counter) = data(32);
      % calculate the current estimation value. check to see if it is a rise or fall.
      % if it is either, log that in the appropriate array of start indexes.
      current = C3(counter)*2^3 + C2(counter)*2^2 + C1(counter)*2 + C0(counter);
      if (mod(counter,4)==1)
        total((counter+1)/2)=Fcalc(F7A,F6A,F5A,F4A,F3A,F2A,F1A,F0A,counter);
        total((counter+1)/2+1)=Fcalc(F7B,F6B,F5B,F4B,F3B,F2B,F1B,F0B,counter);
      end
      if (counter>5 & current>last & num_rises<num_cals)
        rise_starts(i,num_rises+1) = counter;
        num_rises = num_rises + 1;
      elseif (counter>5 & current>last)
        num_rises = num_rises + 1;
      elseif (counter>5 & current<last & num_falls<num_cals)
        fall_starts(i,num_falls+1) = counter;
        num_falls = num_falls + 1;
      elseif (counter>5 & current<last)
        num_falls = num_falls + 1;
      end
      % increment some things and loop around for another line.
      last = current;
      counter = counter + 1;
    end
  end
  fclose(fid);

  if (mod(i,2)==1)
    % These files are not the offset version.
    % Be SURE that the rising and falling edges occur at the same time as the 1->0 transition on
DCLKOUT.
    for j = 1 : num_cals
      % iterate num_cals number of rise and fall starts, then average these together. it's like dithering.
      % sync_latency is the number of cycles (2ns each) between the rising or falling edge at the DAC
input
      % and when this analog data reaches the DAC output. this is defined in the 7721 APP notes. baseline
      % is the old value of the 2nd stage ADC output which the new stuff is compared against.
      rise_start_index = rise_starts(i,j) + 2*latency + error_correction_steps;
      baseline = Fcalc(F7B,F6B,F5B,F4B,F3B,F2B,F1B,F0B,rise_start_index);
      k=0;
      while (k<num_steps/2)
        % since the data is output in 2 banks, only num_steps/2 iterations are needed. this just updates
        % the sync_cal_data matrix based on what's going on at the 2nd ADC output. This is added up
num_cal
        % times and then divided by num_cal (averaged) later. Note the k*4, since new data is ready every
        % 4 cycles (2ns each).
        index = rise_start_index + k*4;
        sync_cal_data((i+1)/2,2*k+1) = sync_cal_data((i+1)/2,2*k+1) +
Fcalc(F7A,F6A,F5A,F4A,F3A,F2A,F1A,F0A,index) - baseline;
        sync_cal_data((i+1)/2,2*k+2) = sync_cal_data((i+1)/2,2*k+2) +
Fcalc(F7B,F6B,F5B,F4B,F3B,F2B,F1B,F0B,index) - baseline;
        if (i==9)
          temp(2*k+1,j) = Fcalc(F7A,F6A,F5A,F4A,F3A,F2A,F1A,F0A,index) - baseline;
          temp(2*k+2,j) = Fcalc(F7B,F6B,F5B,F4B,F3B,F2B,F1B,F0B,index) - baseline;
        end

        k = k + 1;
      end
      % do the same as above for the falling edges.
      fall_start_index = fall_starts(i,j) + 2*latency + error_correction_steps;
      baseline = Fcalc(F7B,F6B,F5B,F4B,F3B,F2B,F1B,F0B,fall_start_index);
      k=0;
      while (k<num_steps/2)
        index = fall_start_index + k*4;
        sync_cal_data((i+1)/2+15,2*k+1) = sync_cal_data((i+1)/2+15,2*k+1) +
Fcalc(F7A,F6A,F5A,F4A,F3A,F2A,F1A,F0A,index) - baseline;
        sync_cal_data((i+1)/2+15,2*k+2) = sync_cal_data((i+1)/2+15,2*k+2) +
Fcalc(F7B,F6B,F5B,F4B,F3B,F2B,F1B,F0B,index) - baseline;
        if (i==9)
          temp2(2*k+1,j) = Fcalc(F7A,F6A,F5A,F4A,F3A,F2A,F1A,F0A,index) - baseline;
          temp2(2*k+2,j) = Fcalc(F7B,F6B,F5B,F4B,F3B,F2B,F1B,F0B,index) - baseline;
        end

        k = k + 1;
      end
    end
  elseif (mod(i,2)==0)
    % These files ARE the offset version.
    % Be SURE that the rising and falling edges occur 1 cycle (2ns) BEFORE the 1->0 DCLKOUT
transition.
    for j = 1 : num_cals
      rise_start_index = rise_starts(i,j) + 2*latency + 1 + error_correction_steps;
      baseline = Fcalc(F7B,F6B,F5B,F4B,F3B,F2B,F1B,F0B,rise_start_index);
      k=0;
      while (k<num_steps/2)
```

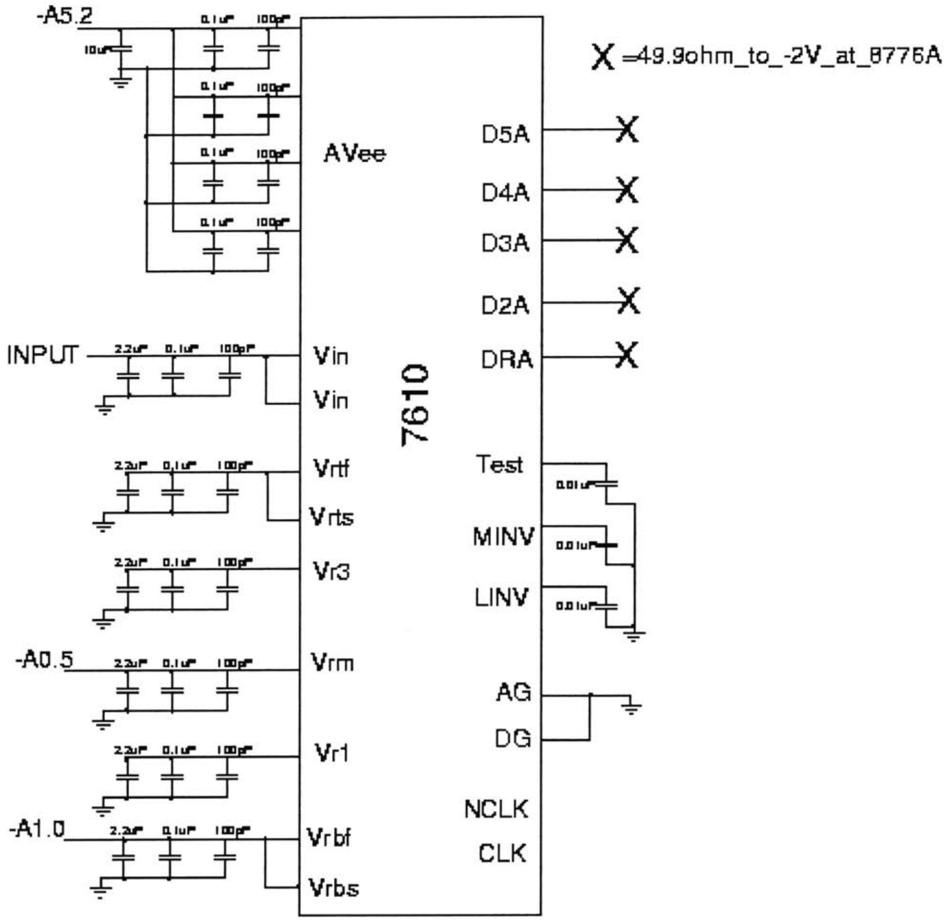
```
        index = rise_start_index + k*4;
        offset_cal_data(i/2,2*k+1) = offset_cal_data(i/2,2*k+1) +
Fcalc(F7A,F6A,F5A,F4A,F3A,F2A,F1A,F0A,index) - baseline;
        offset_cal_data(i/2,2*k+2) = offset_cal_data(i/2,2*k+2) +
Fcalc(F7B,F6B,F5B,F4B,F3B,F2B,F1B,F0B,index) - baseline;
        if (0)
          temp(2*k+1,j) = Fcalc(F7A,F6A,F5A,F4A,F3A,F2A,F1A,F0A,index) - baseline;
          temp(2*k+2,j) = Fcalc(F7B,F6B,F5B,F4B,F3B,F2B,F1B,F0B,index) - baseline;
        end
        k = k + 1;
      end
      fall_start_index = fall_starts(i,j) + 2*latency + 1 + error_correction_steps;
      baseline = Fcalc(F7B,F6B,F5B,F4B,F3B,F2B,F1B,F0B,fall_start_index);
      k=0;
      while (k<num_steps/2)
        index = fall_start_index + k*4;
          offset_cal_data(i/2+15,2*k+1) = offset_cal_data(i/2+15,2*k+1) +
Fcalc(F7A,F6A,F5A,F4A,F3A,F2A,F1A,F0A,index) - baseline;
          offset_cal_data(i/2+15,2*k+2) = offset_cal_data(i/2+15,2*k+2) +
Fcalc(F7B,F6B,F5B,F4B,F3B,F2B,F1B,F0B,index) - baseline;
          if (0)
            temp2(2*k+1,j) = Fcalc(F7A,F6A,F5A,F4A,F3A,F2A,F1A,F0A,index) - baseline;
            temp2(2*k+2,j) = Fcalc(F7B,F6B,F5B,F4B,F3B,F2B,F1B,F0B,index) - baseline;
          end

        k = k + 1;
      end
    end
  end
end


sync_cal_data = sync_cal_data / (num_cals);
offset_cal_data = offset_cal_data / (num_cals);


final_values(9) = 2^10 + offset_cal_data(8,num_steps)/2;
for i = 1 : 8
  final_values(9-i) = final_values (9-i+1) + offset_cal_data(15+9-i,num_steps);
end
for i = 10 : 16
  final_values(i) = final_values (i-1) + offset_cal_data(i-1,num_steps);
end




function [sync_cal_data,offset_cal_data,final_values,temp,temp2,total]=build_cal();

% Files should go in this order:  cal1_sync, cal1_offs, cal2_sync, cal2_offs, etc...
filenames                                                                          =
['cal01_sync.txt';'cal01_offs.txt';'cal02_sync.txt';'cal02_offs.txt';'cal03_sync.txt';'cal03_offs.txt';'cal04_sync.
txt';'cal04_offs.txt';'cal05_sync.txt';'cal05_offs.txt';'cal06_sync.txt';'cal06_offs.txt';'cal07_sync.txt';'cal07_of
fs.txt';'cal08_sync.txt';'cal08_offs.txt';'cal09_sync.txt';'cal09_offs.txt';'cal10_sync.txt';'cal10_offs.txt';'cal11_
sync.txt';'cal11_offs.txt';'cal12_sync.txt';'cal12_offs.txt';'cal13_sync.txt';'cal13_offs.txt';'cal14_sync.txt';'cal
14_offs.txt';'cal15_sync.txt';'cal15_offs.txt'];
```

```
[a,b]=size(filenames);
num_cals = 10;         % Number of times each DAC step is measured (for redundancy).  All are averaged
together.
num_steps = 22;         % Number of 500MHz (2ns) steps that each DAC step is measured (longer = more
settled).
latency = 4;          % Number of 500MHz (2ns) cycles between input and when it is ready at output
sync_cal_data = zeros(30,num_steps);
offset_cal_data = zeros(30, num_steps);
x=0;
y=0;
error_correction_steps=2;

% for each file, open it up and start reading lines from it.
for i = 1 : 30
  fid = fopen(filenames(i,:));
  test = 1;
  counter = 1;
  rise_starts = zeros(1,10);
  num_rises = 0;
  fall_starts = zeros(1,10);
  num_falls = 0;

  % while the test variable is true, keep reading lines.
  while (test==1)
    tline = fgetl(fid);
    if (tline == -1)
      % tline = -1 indicates the end of the file.  stop looping.
      test = 0;
      disp(['Ran out of data in file ' filenames(i,:)]);
    elseif (num_rises>num_cals & num_falls>num_cals)
      % if we have reached the required number of rises and falls to be averaged together, stop looping.
      test = 0;
      disp(['Finished gathering data from file ' filenames(i,:)]);
      counter
    else
      % convert the data line string to numbers and parse it up into all it's elements.
      data = str2num(tline);
      F7B(counter) = data(1);
      F6B(counter) = data(2);
      F5B(counter) = data(3);
      F4B(counter) = data(4);
      F3B(counter) = data(5);
      F2B(counter) = data(6);
      F1B(counter) = data(7);
      F0B(counter) = data(8);
      F7A(counter) = data(9);
      F6A(counter) = data(10);
      F5A(counter) = data(11);
      F4A(counter) = data(12);
      F3A(counter) = data(13);
      F2A(counter) = data(14);
      F1A(counter) = data(15);
      F0A(counter) = data(16);
      CReady(counter) = data(20);
      C3(counter) = data(21);
      C2(counter) = data(22);
      C1(counter) = data(23);
      C0(counter) = data(24);
      DCLKOUT(counter) = data(32);
      % calculate the current estimation value.  check to see if it is a rise or fall.
      % if it is either, log that in the appropriate array of start indexes.
      current = C3(counter)*2^3 + C2(counter)*2^2 + C1(counter)*2 + C0(counter);
      if (mod(counter,4)==1)
        total((counter+1)/2)=Fcalc(F7A,F6A,F5A,F4A,F3A,F2A,F1A,F0A,counter);
        total((counter+1)/2+1)=Fcalc(F7B,F6B,F5B,F4B,F3B,F2B,F1B,F0B,counter);
      end
      if (counter>5 & current>last & num_rises<num_cals)
        rise_starts(i,num_rises+1) = counter;
        num_rises = num_rises + 1;
      elseif (counter>5 & current>last)
        num_rises = num_rises + 1;
      elseif (counter>5 & current<last & num_falls<num_cals)
        fall_starts(i,num_falls+1) = counter;
        num_falls = num_falls + 1;
      elseif (counter>5 & current<last)
        num_falls = num_falls + 1;
      end
      % increment some things and loop around for another line.
      last = current;
      counter = counter + 1;
    end
  end
  fclose(fid);

  if (mod(i,2)==1)
    % These files are not the offset version.
    % Be SURE that the rising and falling edges occur at the same time as the 1->0 transition on
DCLKOUT.
    for j = 1 : num_cals
      % iterate num_cals number of rise and fall starts, then average these together.  it's like dithering.
      % sync_latency is the number of cycles (2ns each) between the rising or falling edge at the DAC
input
      % and when this analog data reaches the DAC output.  this is defined in the 7721 APP notes.  baseline
      % is the old value of the 2nd stage ADC output which the new stuff is compared against.
      rise_start_index = rise_starts(i,j) + 2*latency + error_correction_steps;
      baseline = Fcalc(F7B,F6B,F5B,F4B,F3B,F2B,F1B,F0B,rise_start_index);
      k=0;
      while (k<num_steps/2)
        % since the data is output in 2 banks, only num_steps/2 iterations are needed.  this just updates
        % the sync_cal_data matrix based on what's going on at the 2nd ADC output.  This is added up
num_cal
        % times and then divided by num_cal (averaged) later.  Note the k*4, since new data is ready every
        % 4 cycles (2ns each).
        index = rise_start_index + k*4;
        sync_cal_data((i+1)/2,2*k+1)          =          sync_cal_data((i+1)/2,2*k+1)          +
Fcalc(F7A,F6A,F5A,F4A,F3A,F2A,F1A,F0A,index) - baseline;
        sync_cal_data((i+1)/2,2*k+2)          =          sync_cal_data((i+1)/2,2*k+2)          +
Fcalc(F7B,F6B,F5B,F4B,F3B,F2B,F1B,F0B,index) - baseline;
        if (i==9)
          temp(2*k+1,j) = Fcalc(F7A,F6A,F5A,F4A,F3A,F2A,F1A,F0A,index) - baseline;
          temp(2*k+2,j) = Fcalc(F7B,F6B,F5B,F4B,F3B,F2B,F1B,F0B,index) - baseline;
        end
        k = k + 1;
      end
      % do the same as above for the falling edges.
      fall_start_index = fall_starts(i,j) + 2*latency + error_correction_steps;
      baseline = Fcalc(F7B,F6B,F5B,F4B,F3B,F2B,F1B,F0B,fall_start_index);
      k=0;
      while (k<num_steps/2)
        index = fall_start_index + k*4;
        sync_cal_data((i+1)/2+15,2*k+1)          =          sync_cal_data((i+1)/2+15,2*k+1)          +
Fcalc(F7A,F6A,F5A,F4A,F3A,F2A,F1A,F0A,index) - baseline;
        sync_cal_data((i+1)/2+15,2*k+2)          =          sync_cal_data((i+1)/2+15,2*k+2)          +
Fcalc(F7B,F6B,F5B,F4B,F3B,F2B,F1B,F0B,index) - baseline;
        if (i==9)
          temp2(2*k+1,j) = Fcalc(F7A,F6A,F5A,F4A,F3A,F2A,F1A,F0A,index) - baseline;
          temp2(2*k+2,j) = Fcalc(F7B,F6B,F5B,F4B,F3B,F2B,F1B,F0B,index) - baseline;
        end

        k = k + 1;
      end
    end
  elseif (mod(i,2)==0)
    % These files ARE the offset version.
    % Be SURE that the rising and falling edges occur 1 cycle (2ns) BEFORE the 1->0 DCLKOUT
transition.
    for j = 1 : num_cals
      rise_start_index = rise_starts(i,j) + 2*latency + 1 + error_correction_steps;
      baseline = Fcalc(F7B,F6B,F5B,F4B,F3B,F2B,F1B,F0B,rise_start_index);
      k=0;
      while (k<num_steps/2)
        index = rise_start_index + k*4;
        offset_cal_data(i/2,2*k+1)          =          offset_cal_data(i/2,2*k+1)          +
Fcalc(F7A,F6A,F5A,F4A,F3A,F2A,F1A,F0A,index) - baseline;
        offset_cal_data(i/2,2*k+2)          =          offset_cal_data(i/2,2*k+2)          +
Fcalc(F7B,F6B,F5B,F4B,F3B,F2B,F1B,F0B,index) - baseline;
        if (0)
          temp(2*k+1,j) = Fcalc(F7A,F6A,F5A,F4A,F3A,F2A,F1A,F0A,index) - baseline;
          temp(2*k+2,j) = Fcalc(F7B,F6B,F5B,F4B,F3B,F2B,F1B,F0B,index) - baseline;
        end
        k = k + 1;
      end
      fall_start_index = fall_starts(i,j) + 2*latency + 1 + error_correction_steps;
      baseline = Fcalc(F7B,F6B,F5B,F4B,F3B,F2B,F1B,F0B,fall_start_index);
      k=0;
      while (k<num_steps/2)
        index = fall_start_index + k*4;
          offset_cal_data(i/2+15,2*k+1)          =          offset_cal_data(i/2+15,2*k+1)          +
Fcalc(F7A,F6A,F5A,F4A,F3A,F2A,F1A,F0A,index) - baseline;
          offset_cal_data(i/2+15,2*k+2)          =          offset_cal_data(i/2+15,2*k+2)          +
Fcalc(F7B,F6B,F5B,F4B,F3B,F2B,F1B,F0B,index) - baseline;
          if (0)
            temp2(2*k+1,j) = Fcalc(F7A,F6A,F5A,F4A,F3A,F2A,F1A,F0A,index) - baseline;
            temp2(2*k+2,j) = Fcalc(F7B,F6B,F5B,F4B,F3B,F2B,F1B,F0B,index) - baseline;
          end

        k = k + 1;
      end
    end
  end
end


sync_cal_data = sync_cal_data / (num_cals);
offset_cal_data = offset_cal_data / (num_cals);


final_values(9) = 2^10 + offset_cal_data(8,num_steps)/2;
for i = 1 : 8
  final_values(9-i) = final_values (9-i+1) + offset_cal_data(15+9-i,num_steps);
end
for i = 10 : 16
  final_values(i) = final_values (i-1) + offset_cal_data(i-1,num_steps);
end




function out = calc_est_change (history_table, sync_cal_data, offset_cal_data)

  out = 0;
  for i = 1 : length(history_table)
    if (~(history_table(1,i)==0 | (history_table(2,i) == 0)))
      for j = history_table(1,i) : history_table(2,i)
        if (mod(i,2)==1)
          out = out + sync_cal_data(j,(i+1)/2);
        elseif (mod(i,2)==0)
          out = out + offset_cal_data(j,i/2);
        end
      end
    end
  end
return;



function out = Ccalc(C3, C2, C1, C0, index)

out = C3(index)*2^3 + C2(index)*2^2 + C1(index)*2 + C0(index);

return;


function out = Fcalc (F7A, F6A, F5A, F4A, F3A, F2A, F1A, F0A, index)

out = F7A(index)*2^7 + F6A(index)*2^6 + F5A(index)*2^5 + F4A(index)*2^4 + F3A(index)*2^3 +
F2A(index)*2^2 + F1A(index)*2 + F0A(index);

return;

function [start_step, stop_step] = get_steps (code1, code2)

% This function finds the begining and ending transitions in the cal_data
% table for a given input DAC codes (code1 is before transition, code2 is
% directly after). Be sure the DAC codes are in integer form such that 0
% corresponds to CODE '0000' and 13 corresponds to CODE '1101', etc.


% 0 start and stop steps represents two DAC codes which are same. Since
% the input didn't change, this will be used to know that the estimate
% shouldn't change either.
start_step=0;
stop_step=0;

if (code1 < code2)
   start_step = code1 +1;
   stop_step = code2;
 elseif (code1 > code2)
   start_step = 15 + code2 + 1;
   stop_step = 15 + code1;
end
return;


function ENOB_estimate=my_fft(data,points,time_step,dB_down)

%data = data - mean(data)+1e-12;
%data = data(1:points/4);

%window = hanning(points/4);
%temp=window'.*data;
temp=data;

A = fft(temp,points);
omega = linspace(0,2*pi,points);
f = omega/2/pi/time_step;
A2 = abs(A/max(A));
%figure(1);
stairs(f,20*log10(A2)-dB_down);
max = f(length(f));
axis([0 max/2 -90 0]);
grid on;


fund = A2(1);
fund_index = 1;
second = A2(1);
second_index = 1;
temp = 0;
noise = 0;
signal = 0;
for i = 1 : length(A)/2
  if (abs(A(i))>fund)
     fund = abs(A(i));
     fund_index = i;
  end
 temp = temp + (abs(A(i)))^2;
end
for i = 1 : length(A)/2
```

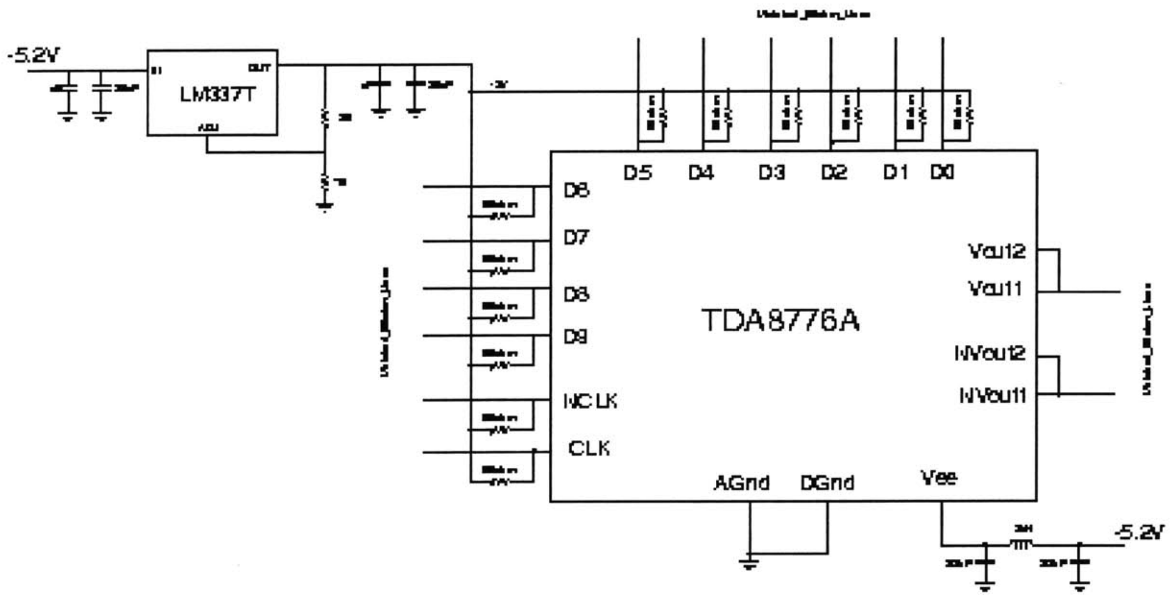
```
  if (abs(A(i))>second & ~(i==fund_index) & ~(i==1))
    second = abs(A(i));
    second_index = i;
  end
end

% Ignore DC part of data in SNR calculation

temp = temp - abs(A(1))^2;

% Fix the oscillation issue
if ((f(second_index)-105e6)<1e6)
  disp('Osicllation Fix in Progress');
  temp = temp - (abs(A(second_index)))^2;
end

signal = fund^2;
noise = temp - fund^2;
disp(['Fundamental Frequency is     ' num2str(f(fund_index)/1e6) ' MHz']);
disp(['Worst Spur is at             ' num2str(f(second_index)/1e6) '  MHz']);
SNR_estimate = 10*log10(signal / noise / 10^(-dB_down/20))
ENOB_estimate = (SNR_estimate-1.76) / 6.02
SFDR_estimate = 1 + 20*log10(abs(fund)/abs(second))



%f(find(20*log10(A2)>-50))
```

Appendix B

Detailed circuit schematic of SPT7610 test setup.



Detailed circuit schematic of TDA8776A evaluation board.



3231- 100