HIGH-SPEED ANALOG-TO-DIGITAL CONVERSION USING 2-STEP FLASH ARCHITECTURES

by

Joey Doernberg

Memorandum No. UCB/ERL M88/73

21 November 1988

HIGH-SPEED ANALOG-TO-DIGITAL CONVERSION USING 2-STEP FLASH ARCHITECTURES

by

Joey Doernberg

Memorandum No. UCB/ERL M88/73

21 November 1988

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

High-Speed Analog-to-Digital Conversion Using 2-Step Flash Architectures

Joey Doernberg

High-Speed Analog-to-Digital Conversion using 2-Step Flash Architectures

Ph.D. Joey Doernberg EECS Dept.

David A. Hodges
Chairman of Commmittee

Abstract

The rapidly expanding area of "digital video" that includes high-definition TV and image recognition, along with the ever present need for faster and higher resolution data acquisition ICs, has created a demand for low-cost, high-speed and moderate-resolution A/D converters. By using a scaled CMOS technology along with innovative circuit techniques, the area required for the A/D converter should soon be small enough to be integrated on the same chip with a microprocessor to make the heart of a digital TV, an image recognition system, or a smart data acquisition system.

The objective of the research presented in this thesis is to determine whether 2-step flash A/D converter architectures in CMOS are advantageous for moderate-resolution video-rate A/D conversion. To test these ideas a prototype 2-step flash A/D converter based upon a resistor string and capacitor arrays was designed and fabricated in a 1.6 μm CMOS technology. Test results verify 10-bit resolution at a 5-Msample/s conversion rate while dissipating 350 mW of power. The fully differential architecture has a 50 pF input capacitance and requires 54k mils^2 of silicon chip area.

The main conclusions are that 2-step flash architectures based upon a resistor string and capacitor arrays are of potential interest for high-speed, moderate-resolution A/D conversion in CMOS because they don't rely upon op amps, have a simple operation and have a high conversion rate. Secondly, simple comparator models predict the optimum number of comparator stages for the fastest decisions. Fully differential amplifiers utilizing diode-connected MOS transistors as resistive loads simplify the comparator gain stage design by eliminating the need for common-mode feedback circuits that limit speed.

Additional research on the comparator gain stage should lead to higher conversion rates. Extensions of the architecture could lead to 12, or more, bits of resolution and multi-step architectures.

Acknowledgement

No words could thank Prof. David A. Hodges enough for his skillful guidance, encouragement, support and patience throughout this work and my years at Berkeley. I would like to thank Prof. Paul R. Gray for his valuable advice.

I would also like to thank Toru Baji and Jesus Pena-Finol for their contributions in the prototype chip design, and Lisa Rugg for building a very versatile test board and controller for the chip.

Many fellow students are very deserving of thanks. I have to start with Hae-Seung Lee, Cheng-Chung Shih and Lee-Chung Yiu for getting me started with ADC design and testing. From my generation of graduate students I have to thank Steve Lewis for endless discussions ranging from analog ICs to ADCs, Bosco Leung, Peter Ruetz and Sehat Sutarja for technical discussions. Then there are thanks for the software support needed. They go to Tom Quarles for "infinite" help with SPICE and UNIX problems and Don Webber for his help in simulating the entire chip with RELAX2. There is also Min-Chie Jeng for help with the BSIM parameter extraction set-up, Robert Kavaler to thank for his help with my "C" programs and Brian Richards for writing the Sun computer driver for the I/O board that collected the data for testing the ADC. Then there are my officemates Cormac Conroy, Wenbin Hsu, Gani Jusuf, Nan-Sheng Lin and Hyun Shin that have contributed with help ranging from technical discussions to proofreading parts of this thesis.

There are many more people to thank. The list must include the many excellent professors from whom I have learned much in my years here, the excellent staff and technical support here at Berkeley and the many fellow students who have all helped. Without them this research would not have been possible.

The research was sponsored by the National Science Foundation under grant number DCI-8603430. The masks were made by MOSIS and the General Electric Company fabricated the prototype chip. Their support is gratefully acknowledged.

Table of Contents

CHAPTER 1

Introduction

1. ADC Background

The new field of "digital video" that includes image recognition and high-definition TV (HDTV) has created a need for inexpensive, high-speed and moderate-resolution analog-to-digital converters (ADCs). The previous generation ADCs were expensive hybrids. Then the first video-rate ADCs were 8-bit flash converters made in a bipolar technology to get the needed speed. Later advanced CMOS processes were able to achieve video-rate conversion. The next generation of converters met the previous speed requirements but needed to increase the resolution to 10 bits so that digital signal processing techniques could be employed and so that HDTV could be realized.

When the resolution was increased from 8 to 10 bits, the flash architectures whose area and power are exponentially related to their resolution had to grow to 4 times their former area. Even with state-of-the-art technologies this was clearly the wrong approach to use for an inexpensive video-rate 10-bit IC ADC. The alternate solutions spanned a range of multi-step converters that include pipelined, classical 2-step and subranging converters. The pipelined architecture uses several stages operating concurrently to get increased throughput and achieves one conversion per clock cycle. The other class of converters uses two steps and trades a speed reduction of a factor of two for a large area savings. Previous converters of this type have been limited to 8 bits and increasing their resolution has been limited by op-amp gain, speed, area and power.

Two new architectures based upon a resistor string and capacitor arrays were developed to overcome the disadvantages of the previous approaches to moderate-precision high-speed ADCs for digital video applications. These along with the comparator design form the basis of the research that is described in this thesis.

2. Thesis Organization

Chapter 2 reviews previous high-speed ADC architectures. Chapter 3 introduces the 2-step Capacitor-array Resistor-string (CR) and Resistor-string Capacitor-array (RC) architectures and shows their advantages and limitations. Chapter 4 goes into the details of the RC ADC prototype design and the high-gain, high-speed comparator design. The prototype ADC controller design is discussed in Chapter 5. ADC testing is discussed in Chapter 6 and then Chapter 7 presents the experimental results and Chapter 8 has more testing and experimental results geared towards testing for specific error sources. The last chapter is the conclusion with a summary of results and a direction for future research. The appendixes have details on programs written and used in designing the prototype.

CHAPTER 2

High-Speed ADC Architecture Review and Comparison

1. Introduction

ADCs have resolution ranging from a few bits in ultra-high speed digitizers to over 22 bits in high-precision digital voltmeters that have conversion rates on the order of a few conversions per second. Figure 2.1 compares these and other architectures on the basis of speed versus precision. In actuality the bubbles in the figure may overlap and expand. The comparison is only for CMOS converters because neither hybrids nor converters made in a bipolar technology are suitable for low cost integration especially with a digital signal processor to make an image processing chip.

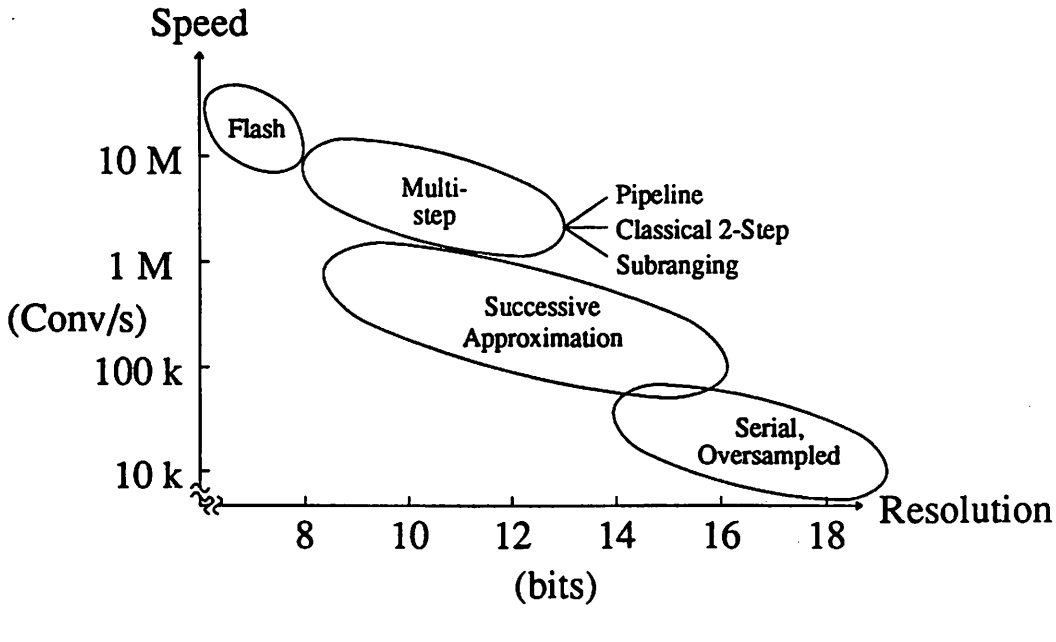


Figure 2.1 - ADC architecture speed versus resolution comparison

Video ADCs fall into the 8 to 10-bit resolution and 10 to 20 Msample/s conversion range. Although 8-bit resolution is acceptable for video at the consumer level, 10-bit resolution and its increased dynamic range is desired so that digital signal processing can be applied at the studio level and for HDTV.

There are a few CMOS flash converters that have 8-bit resolution and the required speed[1,2,3] but it is unfeasible to extend them to 10-bit resolution because of the large area and power required. Conversely successive-approximation architectures easily have the resolution but they lack the needed speed. Bridging the gap between these are the multi-step architectures that borrow from the flash and successive-approximation architectures and make tradeoffs to gain speed over successive approximation and resolution over that available from flash converters.

The multi-step family consists of pipelined, classical 2-step and subranging architectures that includes the CR and RC architectures. Each of these groups was investigated to find the best approach to be implemented in the prototype converter.

2. Flash ADC Architecture

The flash architecture is shown in Figure 2.2. It is also called a "parallel" converter since all comparisons are done simultaneously, i.e. in parallel.

The converter consists of a resistor string, comparators and a digital encoder. Its operation is simple. The 2^n resistors generate 2^n-1 equally spaced voltage levels between the resistor-string endpoints. These are the transitions on the ADC transfer curve. Each of these levels is applied to a comparator along with the unknown input voltage. All of the comparators make their decisions simultaneously. Their outputs are "1's" for comparators whose unknown input is greater than their reference input from the resistor string. Similarly comparators whose reference is greater than the unknown input have outputs that are "0". These outputs form a "thermometer code", a string of 1's up to the comparator whose input from the tap on the resistor string is just less than the input voltage. The thermometer code is converted to binary by the digital encoder. By adding digital latches following the comparators the encoding can be pipelined and the speed is only limited by the slower of the compara-

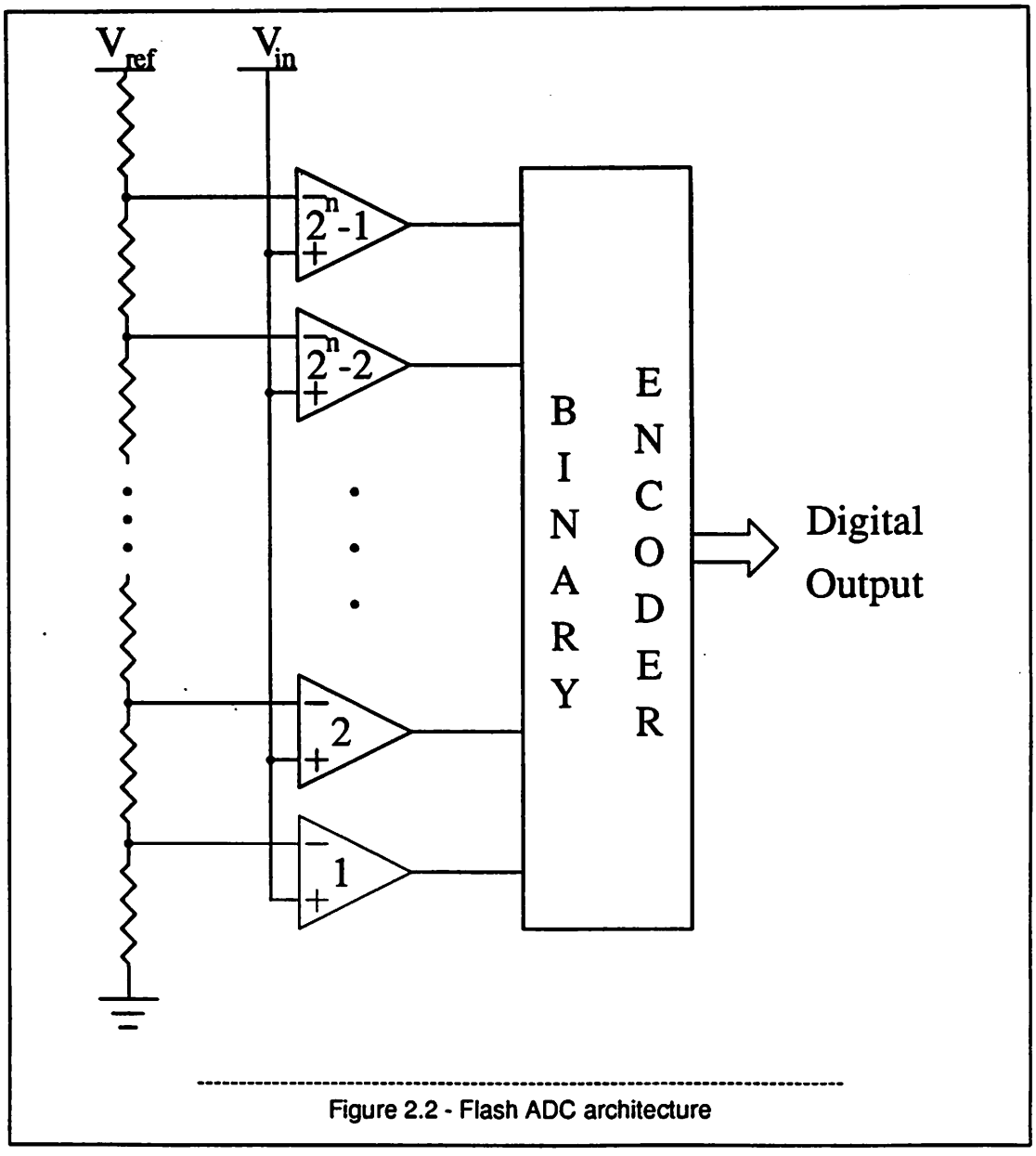


Figure 2.2 - Flash ADC architecture

tors and encoding. (The comparators are usually the speed-limiting component.)

2.1. Advantages

The biggest advantage of the flash converter is its speed. Only one comparison is needed per conversion. Secondly, if the comparators don't have an offset, the conversion is monotonic since the

resistor-string tap voltages are inherently monotonic.

2.2. Limitations

Flash converters use one comparator and resistor per transition. Thus their area, power dissipation and input capacitance grow exponentially with the number of bits. Currently they are limited to 8 bits in CMOS. 10-bit resolution requires 4 times the area and power, a large price to pay although it has been done in a bipolar technology in a chip of size 9.2 x 9.8 mm dissipating 2.0 Watts.[4]

The offset voltage for a CMOS comparator is much larger than for its bipolar counterpart. For a CMOS differential pair the offset voltage is:

$$V_{offset} = \Delta V_{th} + \frac{V_{gs}-V_{th}}{2}\left[\frac{\Delta W}{W} + \frac{\Delta L}{L}\right] \tag{2.1}$$

For a bipolar differential pair the offset voltage is:

$$V_{offset} = V_t\left[\frac{\Delta A}{A} + \frac{\Delta Q_B}{Q_B}\right] \tag{2.2}$$

Where A is the emitter area and Q_B is the base doping under the emitter. In both cases the offest voltage is proportional to geometrical factors, $\frac{\Delta W}{W}$ and $\frac{\Delta L}{L}$ for the CMOS case and $\frac{\Delta A}{A}$ and $\frac{\Delta Q_B}{Q_B}$ for bipolar devices. Assuming the same level of geometrical matching for both cases the proportionality constant is the difference. It is $\frac{V_{gs}-V_{th}}{2}$ for the CMOS case which is often about 0.5 V. For bipolar transistors it is V_t which is 26 mV, almost 20 times smaller. Thus offset cancellation is required for CMOS flash converters. More details are given in Chapter 4, Section 5.5.2 on the comparator design.

When multiple comparators are being used the question of simultaneity among them arises. They must sample the input at exactly the same time or an error will be made. The equation of a sinewave input with amplitude, A, and angular frequency, ω_i, is:

$$V_i = A\sin(\omega_i) \tag{2.3}$$

The error power, P_{err} due to small amounts of time skew in sampling is[5]:

$$P_{err} = \frac{A^2\omega_i^2\sigma_t^2}{2} \tag{2.4}$$

σ_t is the standard deviation of the clock skew in sampling. Since the average power in a sinewave is $\frac{A^2}{2}$ the SNR is:

$$SNR = \sqrt{\frac{A^2/2}{A^2\omega_i^2\sigma_t^2/2}} = \frac{1}{\omega_i\sigma_t} \tag{2.5}$$

Thus:

$$\sigma_t = \frac{1}{\omega_i SNR} \tag{2.6}$$

If a 60 dB SNR is desired and the standard 4.2 MHz video bandwidth is used σ_t must be less than 37.9 ps.

Another way of looking at this is that the sinewave input has a maximum slope of:

$$\left.\frac{dV_i}{dt}\right|_{max} = A\omega_i \approx \frac{\Delta V}{\Delta T}\ (V/s) \tag{2.7}$$

A full-scale input is 2A, thus:

$$1\ LSB = \frac{2A}{2^n} \tag{2.8}$$

The minimum time, Δt, it takes for the input to change by 1 LSB $= \Delta V$ is:

$$\Delta t = \frac{1}{2^{n-1}\omega_i}\ (s) \tag{2.9}$$

Table 2.1 has the time, Δt, for a full-scale 4.2 MHz input to change by 1 LSB for 8, 10 and 12-bit converters.

n, (bits)	Δt, (seconds/LSB)
8	296 ps
10	74 ps
12	19 ps

Table 2.1. Time for a full-scale, 4.2 MHz input to change by 1 LSB.

Therefore Δt must be less than 74 ps to make less than a 1 LSB error in a 10-bit conversion with a full-scale 4.2 MHz signal.

Adding a sample-and-hold amplifier in front of the ADC will eliminate this skewing problem. But jitter on the clock to the sample-and-hold circuit (S/H) will have the same effect but only once, not for each comparator. Evidence of this effect is seen if the ADC's SNR falls as the input frequency increases. Table 2.2 gives the error in LSBs for n-bit resolution, and a timing error, Δt.

n, (bits)	Δt (ns)	error (LSBs)
8	1.0	3.38
8	0.1	0.34
10	1.0	13.5
10	0.1	1.35
12	1.0	54.1
12	0.1	5.41

Table 2.2. Error in LSBs for a timing error,

As the comparators are switched to the resistor string for comparison they load the ladder and perturb the tap voltages.[6,7] The net effect is that as the precision increases the resistance in the string must go down as square of the resolution. Thus the power dissipated in the string increases as the square of the resolution in bits.

3. Successive-Approximation ADC Architecture

The successive-approximation architecture searches through the 2^n-1 voltages levels with a binary search. This is very efficient since only one comparator is needed but n comparisons are needed. Figure 2.3 shows a conceptual diagram of the successive-approximation ADC. The operation is as follows. The comparator compares the unknown input to the DAC output. The comparator output goes to the Successive-Approximation Register (SAR) which generates the digital code needed for the DAC to do a binary search. Thus the n-bit conversion takes n cycles.

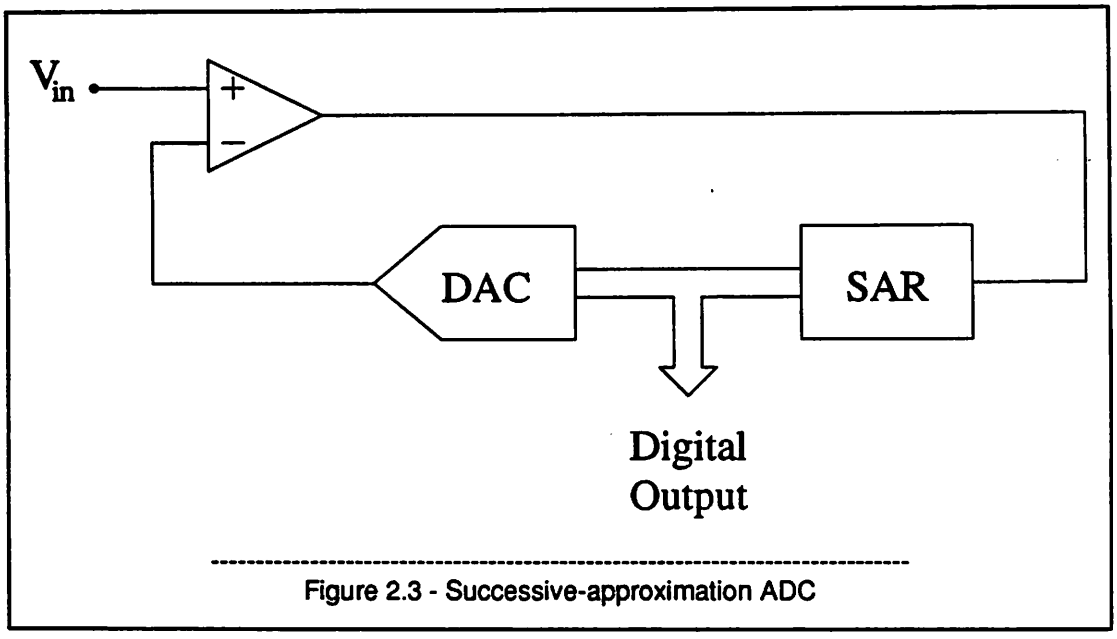


Figure 2.3 - Successive-approximation ADC

3.1. Advantages

The architecture is quite simple and only one comparator is required. This is the obvious advantage over a flash converter. The DAC may be implemented as a resistor-string, capacitor-array, R-2R ladder or any method appropriate for the technology and conversion requirements.

3.2. Limitations

Although the architecture is very efficient, *n* comparisons are required for an n-bit conversion. If a flash converter is limited by comparison time, the successive-approximation converter will be approximately *n* times slower. 10 to 15 Msample/sec conversions with 10-bit resolution is unfeasible even with today's most advanced technologies using this method.

4. Multi-step ADC Architectures

Multi-step architectures can be broken down into 3 groups based upon their implementation. They are the pipelined, classical 2-step and subranging architectures. However a closer look at their operation reveals that the classical 2-step is also a "subranging" converter in its operation. Figure 2.4

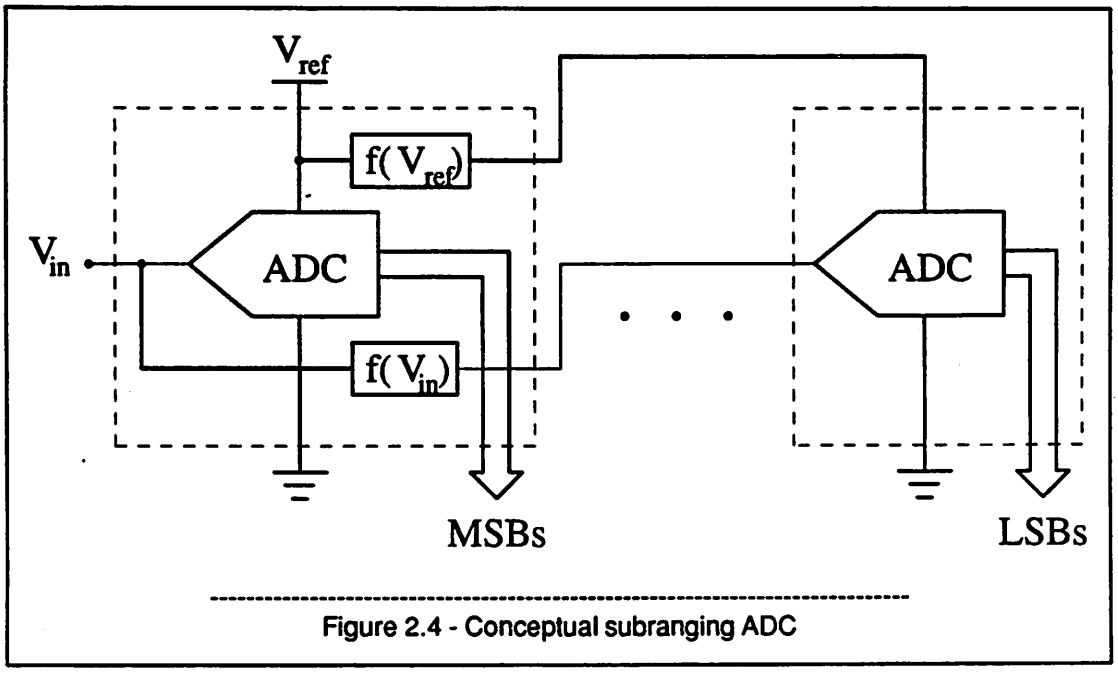


Figure 2.4 - Conceptual subranging ADC

shows a conceptual block diagram of a subranging converter. It consists of two or more ADCs, references and the unknown input or some function of the unknown input being passed on to later stages. The later stages can share parts of previous stages. In operation the first stage does a "coarse" A/D conversion. The next stage takes the result, the digital code, some form of the reference voltages and input and does a more precise conversion. If there are more stages this continues until the last stage where the final precision is attained. From this starting point the specific converters can be described.

4.1. Pipelined ADC Architecture

The pipelined architecture is shown in Figure 2.5. It consists of two or more stages each consisting of a S/H, ADC, DAC, subtractor and a gain block. The S/H holds the input while the ADC does a conversion. Then the DAC creates a quantized version of the input that is subtracted from the held input. The result of the subtraction is a small residual voltage that is amplified and passed on to the next stage for conversion in the next clock cycle. While the next stage operates on this input the current stage takes a new sample and converts it thus increasing throughput via concurrent operation.

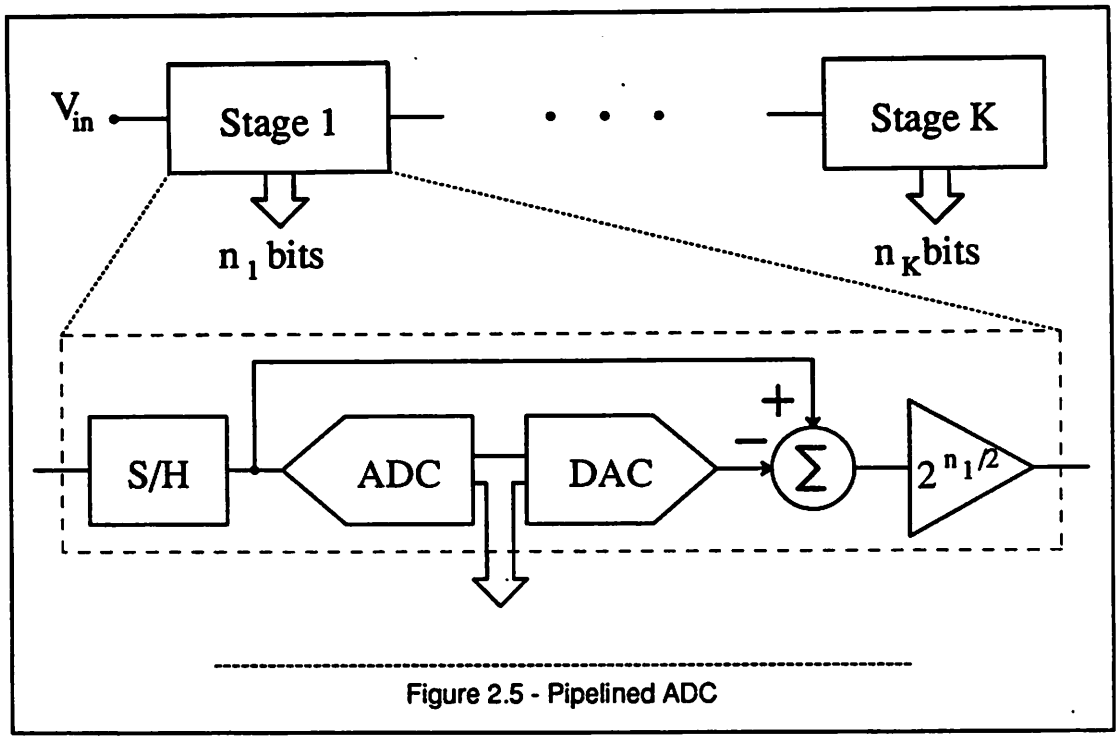


Figure 2.5 - Pipelined ADC

The pipelined ADC is actually an improved subranging ADC. While its operation is to subrange, the use of multiple stages operating concurrently leads to higher speed operation. One conversion is done each clock period. This however is not as fast as the flash converter since the flash converter offset cancels and makes one comparison per period the pipeline must do a sample and hold, a comparison, D/A conversion, subtraction and amplification each period.

4.1.1. Advantages

The obvious advantage of the pipelined ADC is the potentially fast operation, one conversion per clock cycle. Increased resolution is attained by adding additional stages. Thus the area and power grow linearly with precision. Results show that 9-bit resolution at a 5-Msample/s conversion rate in a 3 μm CMOS technology is possible.[8]

4.1.2. Limitations

The most stringent requirements are on the op amps used in the S/H, subtractor and gain block. In the first stage they must settle to their final values to the full accuracy of the converter in less than one clock period. The accuracy requirement is lessened by an amount equal to the gain in the following stages. Unless some form of correction is added the DAC must match the ADC or errors can be made.

4.2. Classical 2-step Flash ADC

The classical 2-step flash ADC is a subranging converter with only 2 stages. It is shown in Figure 2.6 and consists of a S/H, MSB ADC, DAC, subtractor, gain block and LSB ADC. The operation is seen with the accompanying timing diagram. In the first phase the input is sampled and held. Then the MSBs are A/D converted. In the third phase the DAC reconstructs the input and that is subtracted

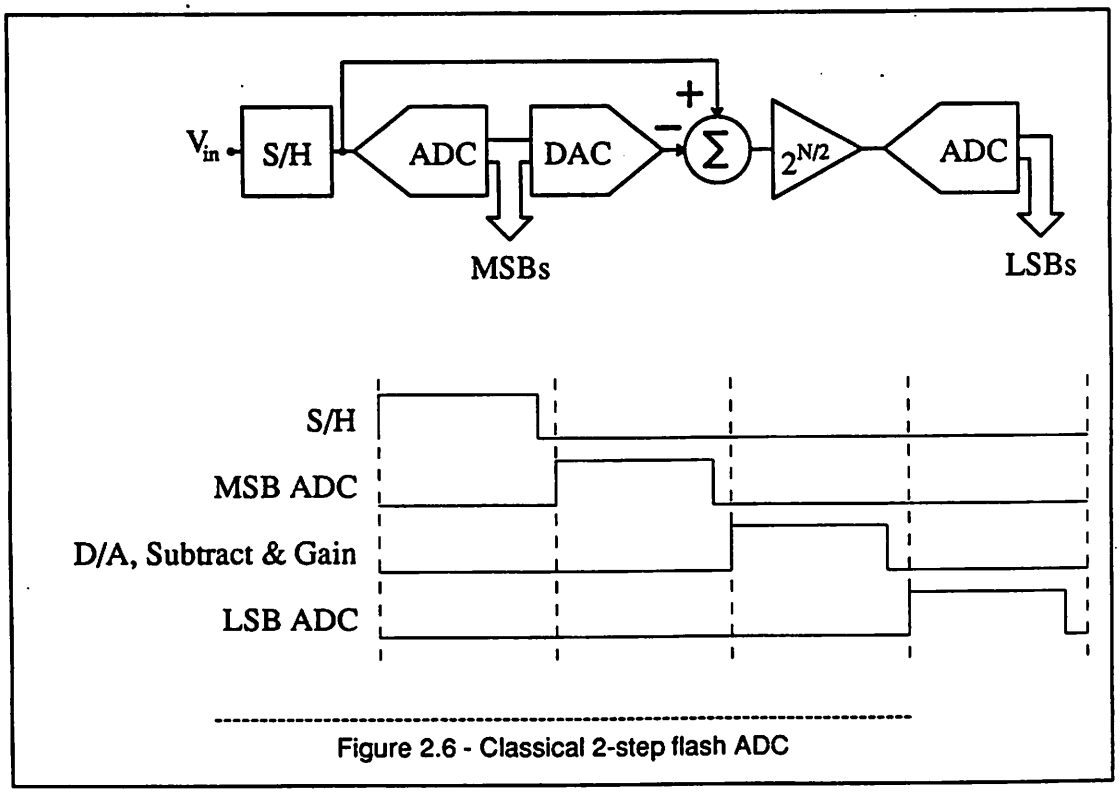


Figure 2.6 - Classical 2-step flash ADC

from the held input and the result is amplified. The fourth phase is the second step where the LSBs are converted. If there was a S/H after the gain stage this would be a 2-stage pipeline.

Currently this architecture can achieve 8-bit resolution with a 20 Msample/s conversion rate in a CMOS technology.[9]

4.2.1. Advantages

This architecture is implemented using flash ADCs for the MSBs and LSBs. Thus for a 10-bit conversion two 5-bit ADCs are needed. The substantial savings in area and power comes from the reduction in the number of needed comparators, from 1023 to 62. The DAC uses the same resistor-string that the ADC uses so there is no matching problem and area is saved.

4.2.2. Limitations

The limitations are the same as those of the pipelined converter where the burden is on the op amps.

4.3. Subranging ADC Architecture

A common subranging ADC implementation is shown in Figure 2.7. It consists of a 2^n resistor string and $2^{n/2}-1$ comparators, a switch bank and a S/H. In the first step the comparators are switched to points on the resistor string every $2^{n/2}$ taps apart and the MSB decision is made. When the resistor interval where the input voltage lies is known, the $2^{n/2}-1$ comparators are connected to all taps in that range to do the "fine" or LSB conversion.

When the term "subranging converter" is used it generally refers to a converter of the form in Figure 2.7. A more general form is shown in Figure 2.8. It consists of a S/H, two ADCs (one for the MSBs and one for the LSBs) and two DACs. The operation is similar to that of the classical 2-step ADC. The input is sampled and held. Then the MSB ADC does the coarse conversion. Its digital output goes to the DAC to create a quantized version of the input which is used as the lower voltage reference for the LSB ADC. The MSB ADCs output is also incremented then goes to a second DAC to generate the upper voltage reference for the LSB ADC. Since the digital input to the two DACs differ

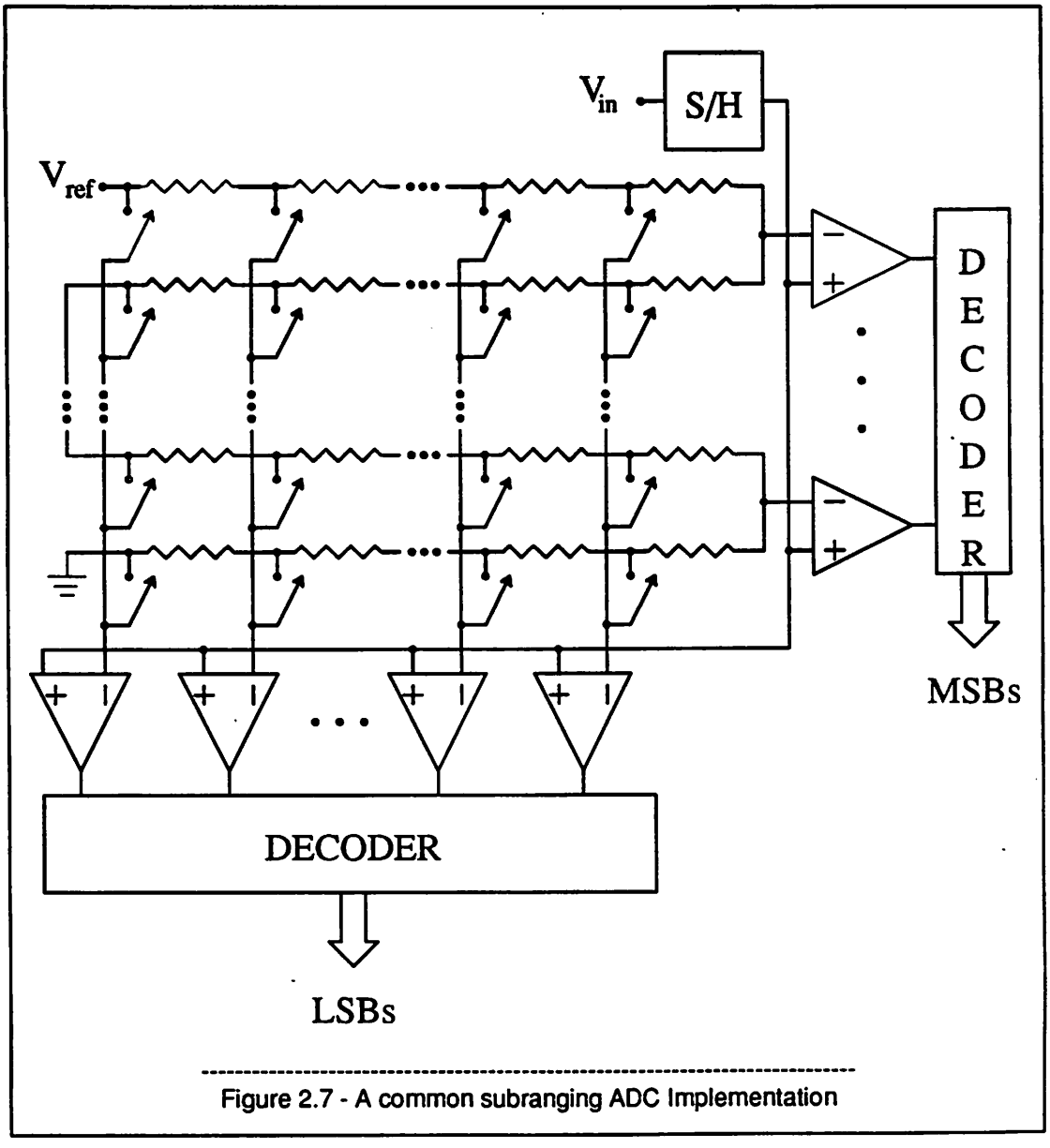


Figure 2.7 - A common subranging ADC Implementation

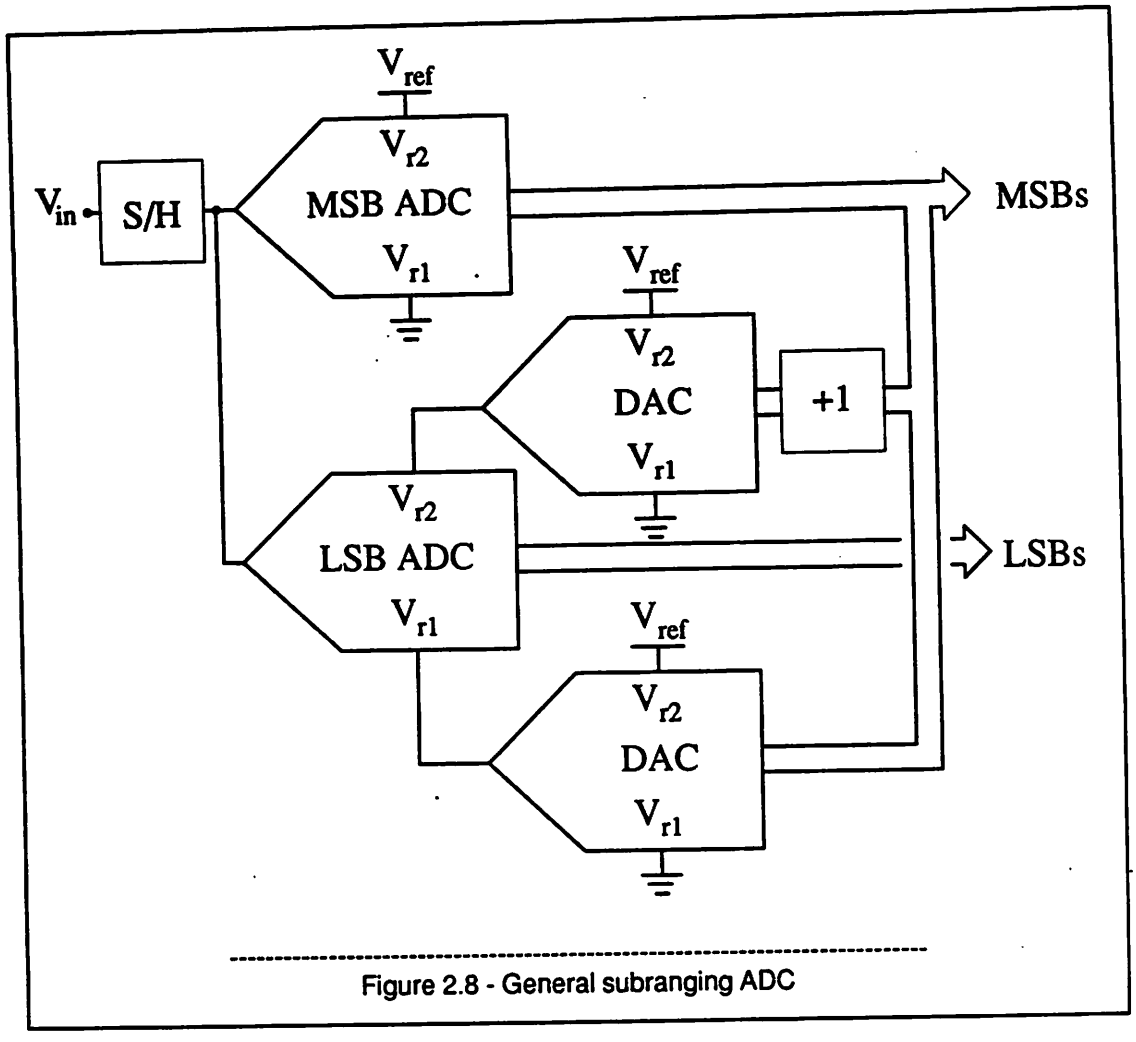


Figure 2.8 - General subranging ADC

by one least significant bit of the MSB ADC the two DAC outputs will encompass the unknown input. Both of these references are a subset of the voltages used as references to the MSB ADC hence the term "subranging ADC".

4.3.1. Advantages

For a n-bit ADC this implementation uses $2^{n/2}-1$ rather than 2^n-1 comparators resulting in a large savings of area and power.

4.4. Limitations

This architecture has been implemented previously at an 8-bit level.[10] The limitations to extending it to 10 bits are in the comparators and resistors. The 8-bit implementation uses two sets of comparators and encoders. It still needs 2^n matched resistors and in the 8-bit realization uses two resistor strings, one for the MSBs and the other for the LSBs. Poor matching between them will lead to large differential nonlinearity errors.

5. Potential of 2-step Flash ADC Architectures

The 2-step ADC architecture is based on using $2^{n/2}-1$ rather than 2^n-1 comparators as in a flash converter and the same savings in matched components. This could yield a large savings in area and power at the expense of two clock cycles rather than one clock cycle for the conversion.

The main drawbacks have been the high-speed, high-gain op amps. The same limitations the pipelined converter had. Other limitations have been matching ADCs and DACs to each other and architectures that have needed 2^n matched components. What is needed is an architecture that eliminates the op amps and subranges without using 2^n precision matched elements. The next chapter will describe two 2-step ADC architectures without these limitations.

CHAPTER 3

Resistor-String Capacitor-Array ADC Architectures

1. Introduction

The previous chapter showed the advantages of using a subranging ADC architecture, specifically a 2-step flash converter for doing a moderate-precision, high-speed digitization. However previous attempts were unfeasible at a 10-bit level. What is needed is an architecture without op amps and without 2^n matched components.

ADCs can be made using resistor strings as the precision elements as in flash converters. ADCs can also be made using binary-weighted capacitor arrays. If both of these could be combined, each doing one step of the 2-step process the goal could be reached. The new architectures that do this are a Capacitor-Array Resistor-String (CR) 2-step flash ADC[11] and a Resistor-String Capacitor-Array (RC) 2-step flash ADC.[11, 12] These will be derived from their simpler components.

2. Binary-Weighted Capacitor-Array ADC

The Binary-Weighted Capacitor-Array (BWCA) ADC, also known as a "Charge-Redistribution" (Q-R) ADC,[13] is shown in Figure 3.1. It consists of $n+1$ capacitors of weights C, C, 2C, ... , 2^{n-1}C, each with bottom-plate switches to select V_{in}, ground or V_r, a comparator and a top-plate grounding switch, S1, along with the digital control to set the switches.

The operating principle is based upon charge redistribution. In the first step the top-plate grounding switch is closed and the capacitor bottom-plate switches are set to sample the input. This step ends when the top plate and then bottom-plate switches are opened in this order. This stores V_{in} on the capacitors and is an inherent S/H. Specifically, consider a 3-bit example with capacitors C, C, 2C and 4C. The initial total charge stored on the capacitor top plates is:

$$Q_i = -8CV_{in} \tag{3.1}$$

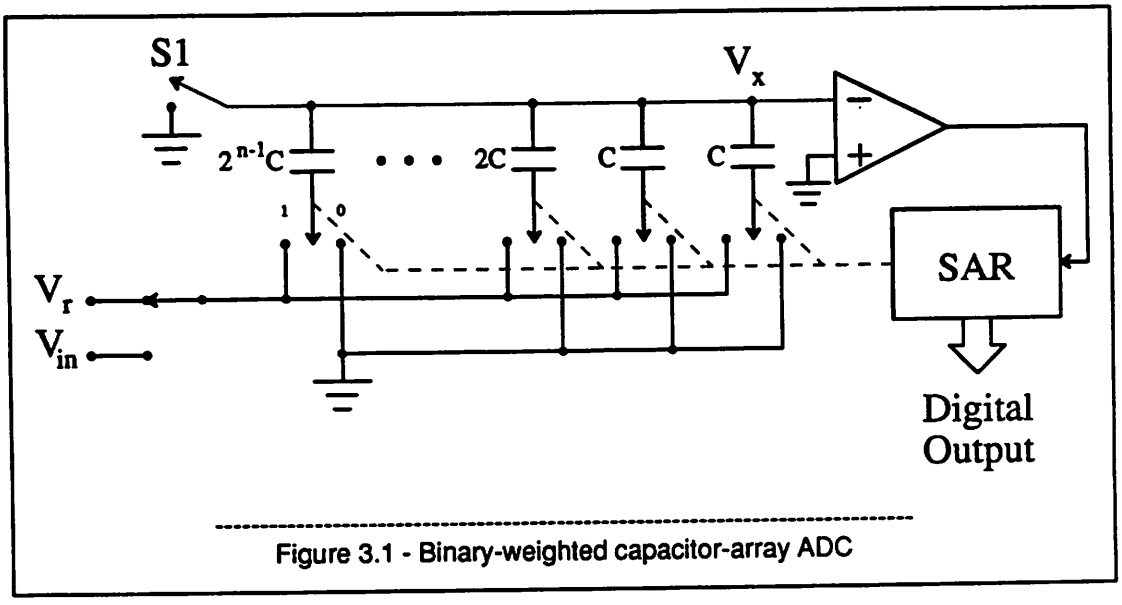


Figure 3.1 - Binary-weighted capacitor-array ADC

The voltage at the top plate and comparator input is V_x.

$$Q_f = (V_x - V_r)4Cb_2 + V_x 4C\overline{b_2} + (V_x - V_r)2Cb_1 + V_x 2C\overline{b_1} \tag{3.2}$$
$$+ (V_x - V_r)Cb_0 + V_x C\overline{b_0} + V_x C$$

Where b_x means the switch connected to the capacitor of value 2^x is set to V_r and $\overline{b_x}$ means the switch is set to ground. Equation 3.2 can be simplified to the following by collecting the similar capacitor terms and noting that $V_x(b_x + \overline{b_x})C = V_x C$, since + is a logical "or" and $(b_x + \overline{b_x}) = 1$.

$$Q_f = 8CV_x - V_r(4Cb_2 + 2Cb_1 + Cb_0) \tag{3.3}$$

The initial and final charge are equal since charge is conserved and V_{in} can be solved for.

$$V_{in} = \left[\frac{1}{2}b_2 + \frac{1}{4}b_1 + \frac{1}{8}b_0\right] V_r - V_x \tag{3.4}$$

The controller is the Successive Approximation Register (SAR) and it sets the bottom-plate switches starting from the largest capacitor to the smallest based upon the comparator output at each step. At the end of n comparisons the top-plate voltage, V_x is close to zero and the switch positions represent the digital code of the input voltage.

2.1. Advantages

This converter has an inherent S/H by sampling the input on the capacitor array. The operation is simple and up to 10-bit resolution and accuracy has been attained.[13]

2.2. Limitations

The converter operates in a successive-approximation manner so n comparisons are needed for an n-bit conversion making it too slow for video rate use. Also the capacitors vary in value from C to 2^nC taking up a large area for a large number of bits, n.

3. Binary-Weighted Capacitor-Array ADC with Increased Precision

The simplest way to increase the precision of the BWCA ADC is to add another capacitor twice the size of the MSB capacitor. The disadvantage is that the total array capacitance is doubled which limits the conversion rate; the capacitor area (the bulk of the chip area) has doubled and this can't be extended beyond the limits of the capacitor matching.

Figure 3.2 shows another way to increase the resolution. This is the same as the standard BWCA ADC shown in Figure 3.1 with the addition of the sub-DAC. The effect of the sub-DAC is to change the top-plate voltage by an amount smaller than 1 LSB during charge balancing and comparison to increase the resolution. The top-plate voltage, V_x changes by an amount:

$$\Delta V_x = V_{DAC} \cdot \frac{C_c}{C_{array} + C_c}. \tag{3.5}$$

Since $\frac{C_c}{C_{array} + C_c} = \frac{1\,LSB}{V_{fs}}$ the top-plate voltage can be changed by 0 to 1 LSB in as many steps as the sub-DAC has. So the conversion proceeds first with the capacitor bottom-plate switches being switched by the SAR to do a coarse conversion. Then another successive-approximation search is done with the sub-DAC to further increase the resolution. This is a subranging operation because the first search does a quantization with the capacitor array and the second search quantizes within the interval found in the first search. This is called the Capacitor-array Resistor-string (CR) ADC since the capacitor array is used for the coarse quantization and the resistor string is used for the fine conver-

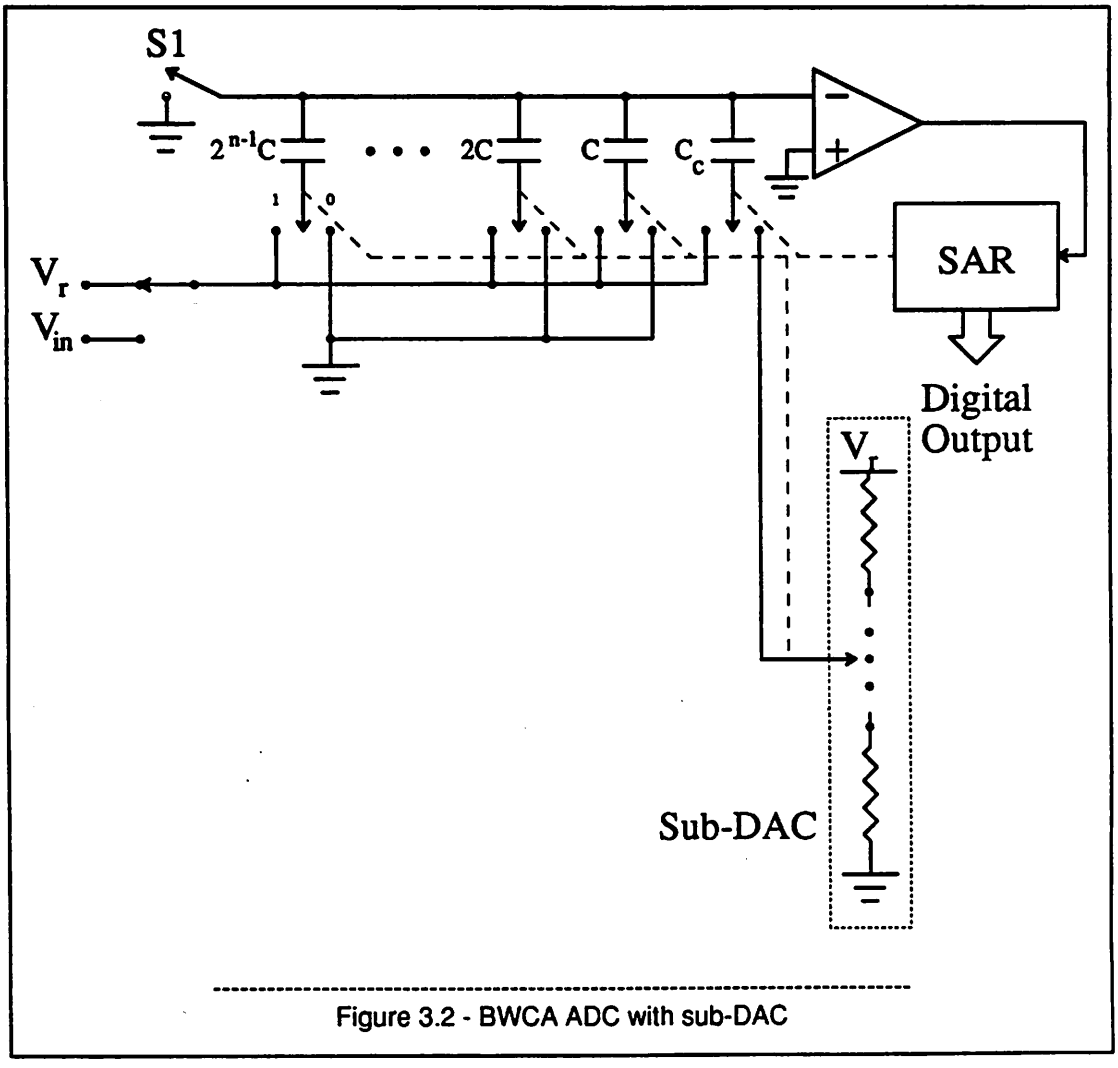


Figure 3.2 - BWCA ADC with sub-DAC

sion.

3.1. Advantages

The resolution has been increased without increasing the area substantially by adding a resistor-string sub-DAC.

3.2. Limitations

Although the resolution has been increased the accuracy is limited by the capacitor matching in the first step. This is still a relatively slow ADC since it uses a successive-approximation algorithm.

4. Binary-Weighted Capacitor-Array Flash ADC

The speed limitation of the Q-R ADC comes from using it in a successive-approximation manner needing n comparisons for an n-bit conversion. This doesn't have to be so. If 2^n-1 BWCA ADCs are used, all sampling the same input, and with their bottom-plate switches preset to the 2^n-1 codes (0...1 through 1...1) they act simultaneously and a "flash" decision can be made. This is shown in Figure 3.3.

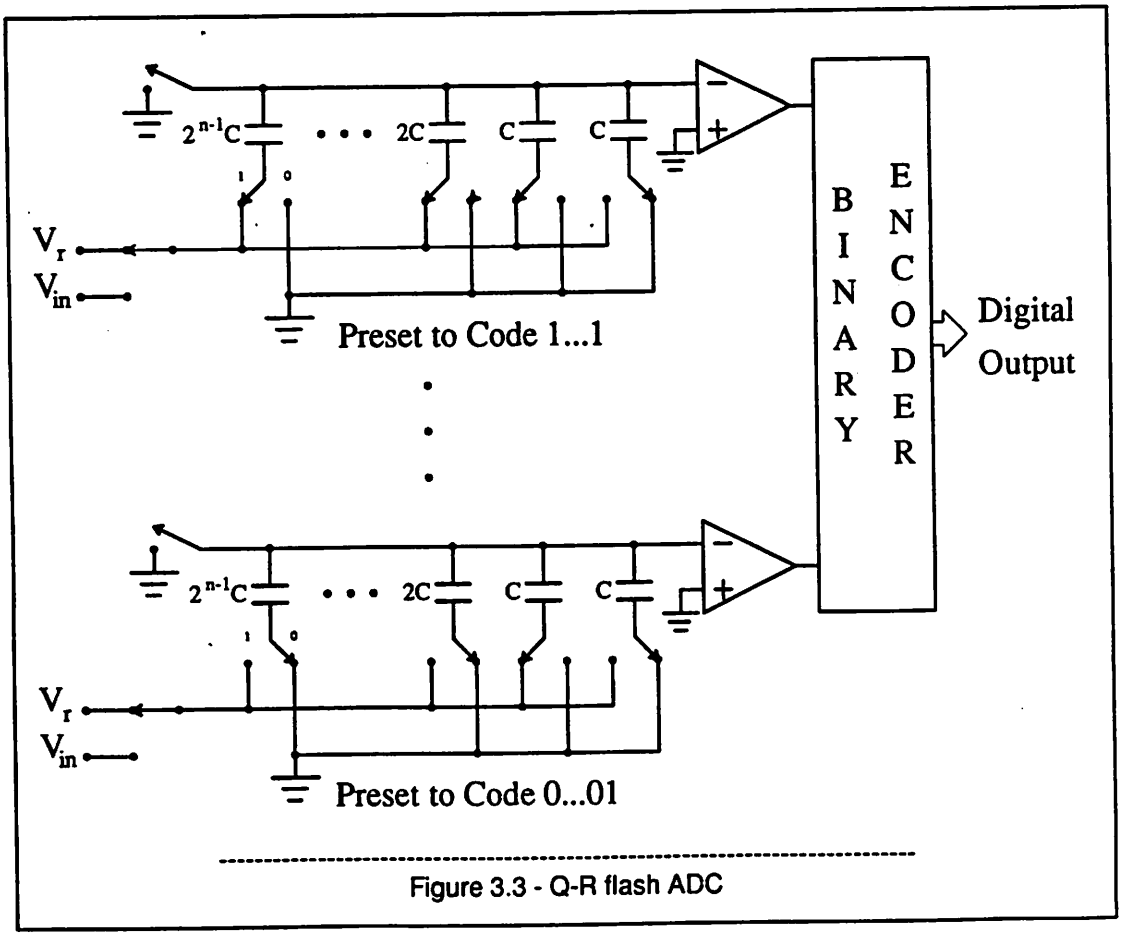


Figure 3.3 - Q-R flash ADC

4.1. Advantages and Limitations

With the parallel structure the speed is increased to that of a flash converter at the expense of 2^n-1 times more area.

5. Capacitor-Array Resistor-String Flash ADC

The CR ADC is slow since it is a successive-approximation architecture. The first step toward speed-up is to make 2^n-1 capacitor arrays and do a flash decision. That still leaves the sub-DAC search taking m steps for the last m bits. Notice that in the last m-bit search only one capacitor array section is needed while the rest sit idle. If the number of bits in the sub-DAC is less than or equal to the number in the capacitor array, the sub-DAC search can also be done in parallel. First the capacitor-array flash MSB decision is done. Then the bottom-plate switches on all of the arrays are set to the MSB code as shown in Figure 3.4, and each array's coupling capacitor, C_c, goes to a tap on the sub-DAC resistor string. Thus the bottom array gets the ground tap and its top-plate voltage is unshifted. The next array gets the first tap and its top-plate voltage is shifted by:

$$\Delta V_x = V_{DAC} \cdot \frac{C_c}{C_{array} + C_c} \tag{3.6}$$

which is:

$$\Delta V_x = Vr \cdot \frac{1}{32} \cdot \frac{1}{32} \tag{3.7}$$

and:

$$\Delta V_x = Vr \cdot \frac{n}{32} \cdot \frac{1}{32} \tag{3.8}$$

in general for n less than or equal to 31 for the 10-bit converter. Each top-plate voltage is shifted by 1 LSB at the final resolution and the LSBs decision is made in one step for the second flash decision.

5.1. Advantages

A slow successive-approximation architecture has been made into a much faster 2-step architecture by adding 30 comparators and 30, 5-bit capacitor arrays. This architecture meets the goal of not needing an op amp to do a 2-step A/D conversion.

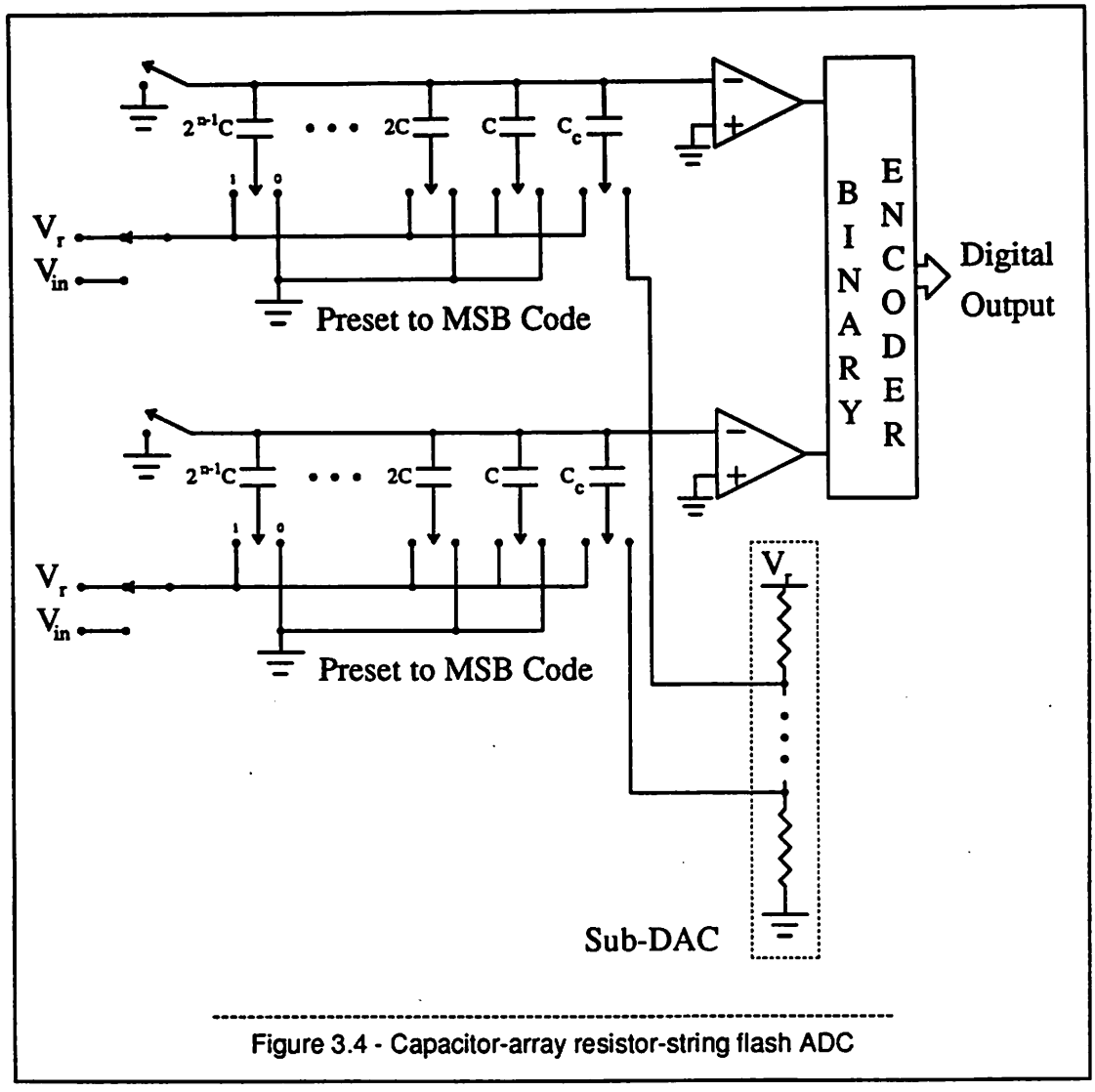


Figure 3.4 - Capacitor-array resistor-string flash ADC

5.2. Limitations

There is a severe capacitor matching requirement that is not immediately obvious. At first look, since the capacitor arrays are used in the first step, they must be 10-bit accurate and the resistor string used in the second step must be 5-bit accurate. Notice however that when the second step is begun and all 31 capacitor arrays are set to the same code all of their top plates are assumed to go to the same voltage before the sub-DAC is used. They will only go to the same voltage if each array matches each of the other 30 arrays. Thus the matching requirement is that each array must have 10-bit matching and

each array must match each other array.

5.2.1. Capacitor Array Matching

To see the effective matching requirements on the capacitors assume random mismatches with a Gaussian distribution. Next assume that 10-bit matching can be obtained with 67% yield. Then the probability that all 31 arrays will match each other is $(0.67)^{31} = 4.1 \times 10^{-6}$. This gives a very unreasonably low chip yield.

Thus the limitation on this architecture is technological and not inherent in the architecture. The limitation comes from matching the 31 arrays to each other. What is needed is a first step that uses the same elements once and not 31 sets of elements. This can be done. The next section will start to show how it is done.

6. Binary-Weighted Capacitor-Array ADC with 2 Voltage References

The previous BWCA ADC had its bottom-plate switches set to V_r or ground. Thus it converted inputs lying between V_r and ground. This is not a limitation. Those two switch positions can be V_{r1}

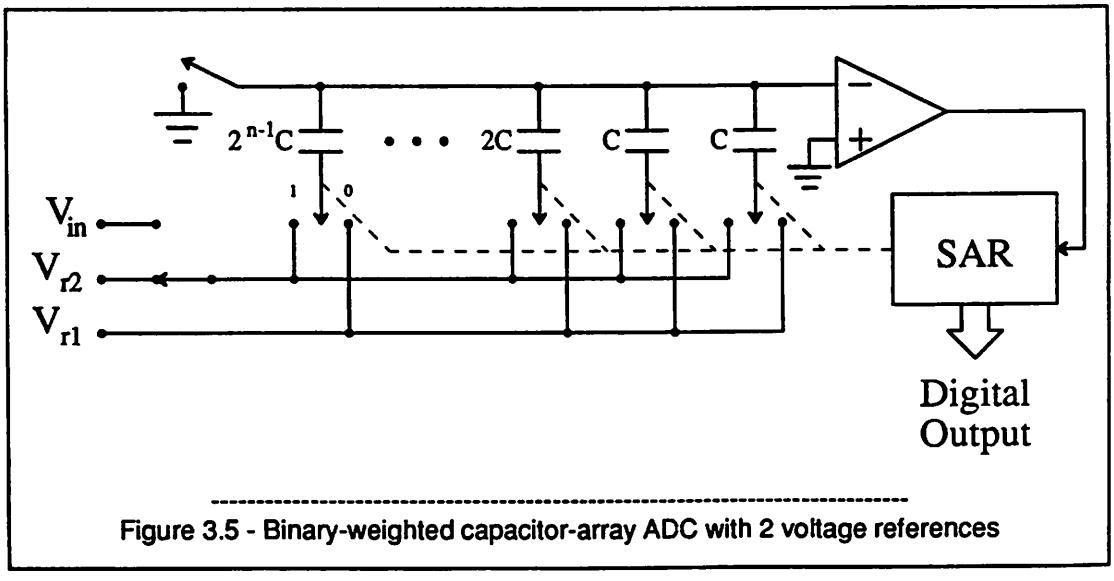


Figure 3.5 - Binary-weighted capacitor-array ADC with 2 voltage references

and V_{r2} as shown in Figure 3.5. The operation is the same. The charge redistribution goes as follows:

$$Q_i = -8CV_{in} \tag{3.9}$$

$$\begin{aligned} Q_f = {} & C(V_x - V_{r2})4b_2 + C(V_x - V_{r1})4\overline{b_2} \\ & + C(V_x - V_{r2})2b_1 + C(V_x - V_{r1})2\overline{b_1} \\ & + C(V_x - V_{r2})b_0 + C(V_x - V_{r1})\overline{b_0} + C(V_x - V_{r1}) \end{aligned} \tag{3.10}$$

$$Q_f = 8CV_x - CV_{r2}(4b_2 + b_1 + b_0) - CV_{r1}(4\overline{b_2} + 2\overline{b_1} + \overline{b_0}) \tag{3.11}$$

Setting $Q_i = Q_f$ and solving for V_{in}:

$$V_{in} = V_{r2}\left[\frac{1}{2}b_2 + \frac{1}{4}b_1 + \frac{1}{8}b_0\right] + V_{r1}\left[\frac{1}{2}\overline{b_2} + \frac{1}{4}\overline{b_1} + \frac{1}{8}\overline{b_0}\right] - V_x \tag{3.12}$$

The effect is to subdivide the range between V_{r1} and V_{r2}. This is more easily seen by setting $V_{r2} = V_{r1} + \Delta V$. Then

$$V_{in} = V_{r1} + \Delta V\left[\frac{1}{2}b_2 + \frac{1}{4}b_1 + \frac{1}{8}b_0\right] - V_x \tag{3.13}$$

The input is V_{r1} plus some fraction of ΔV, the difference between V_{r1} and V_{r2}.

The conclusion is that when two voltage references are used this is a subranging converter so it can be used as the second step in a 2-step ADC.

6.1. Advantages

This architecture can be used for a subranging converter and since it is just a generalization of the BWCA ADC in Section 2. It can also be used a flash ADC as shown in Figure 3.6. More importantly its two advantages, subdivision and use as a flash can be combined to make a 2-step flash ADC as described in the next section.

7. Resistor-String Capacitor-Array ADC

The last section alluded to the Resistor-String Capacitor Array (RC) architecture. [14] It is shown in Figure 3.7. It consists of an m-bit resistor string, an n-bit capacitor array, a 2-output analog multiplexer, a comparator and an SAR. The operation is to sample the unknown input voltage on the

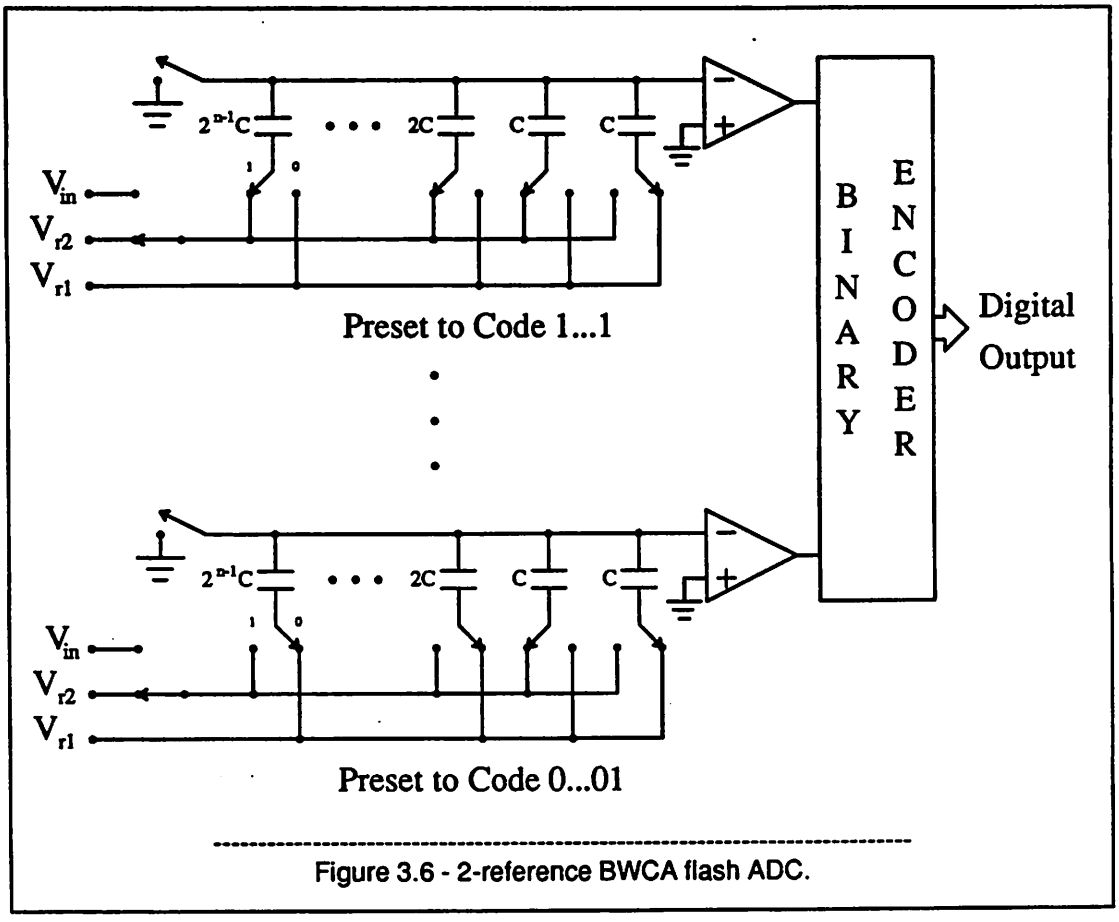


Figure 3.6 - 2-reference BWCA flash ADC.

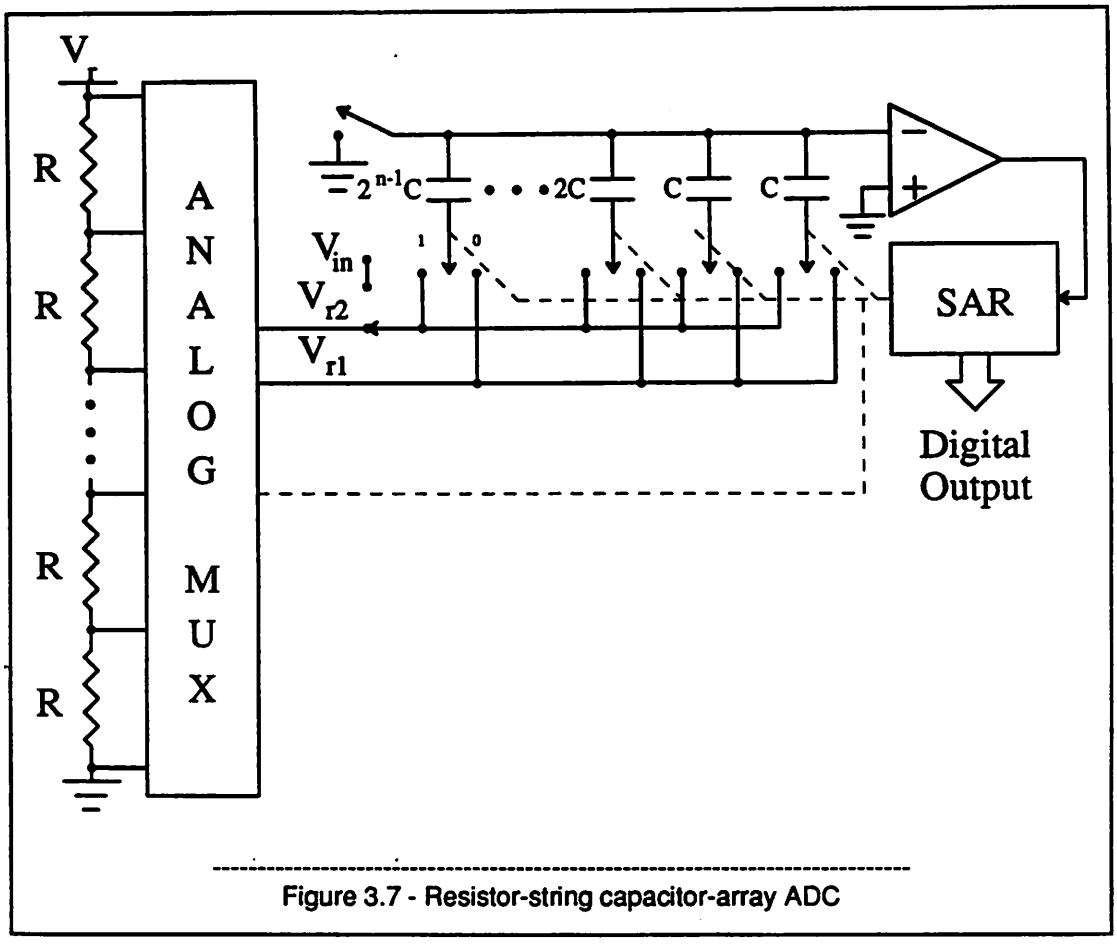


Figure 3.7 - Resistor-string capacitor-array ADC

capacitor array, used as one capacitor, and do a successive-approximation search on the resistor string for the MSBs. Once they are found the resistor whose tap voltages encompass the unknown input is selected and the voltages at its endpoints are selected by the multiplexer and used as the two reference voltages for the capacitor array ADC. The capacitor array ADC then does a second successive-approximation search for the LSBs.

7.1. Advantages

An m+n-bit A/D conversion is done using only 2^m+2^n matched components. This is a large savings in area over a converter that needs 2^{m+n} matched components.

7.2. Limitations

This converter is slow since it uses a successive-approximation algorithm.

8. Resistor-String Capacitor-Array 2-Step Flash ADC

All of the groundwork has been laid now. If both the MSB and LSB steps of the previous architecture are done as a flash the speed is greatly improved.[11,12] This new architecture is shown in Figure 3.8 and will be used in the prototype. It consists of an m-bit resistor string, a two-output analog multiplexer, 2^n-1 n-bit capacitor arrays and comparators, a latch bank, a binary encoder and digital control.

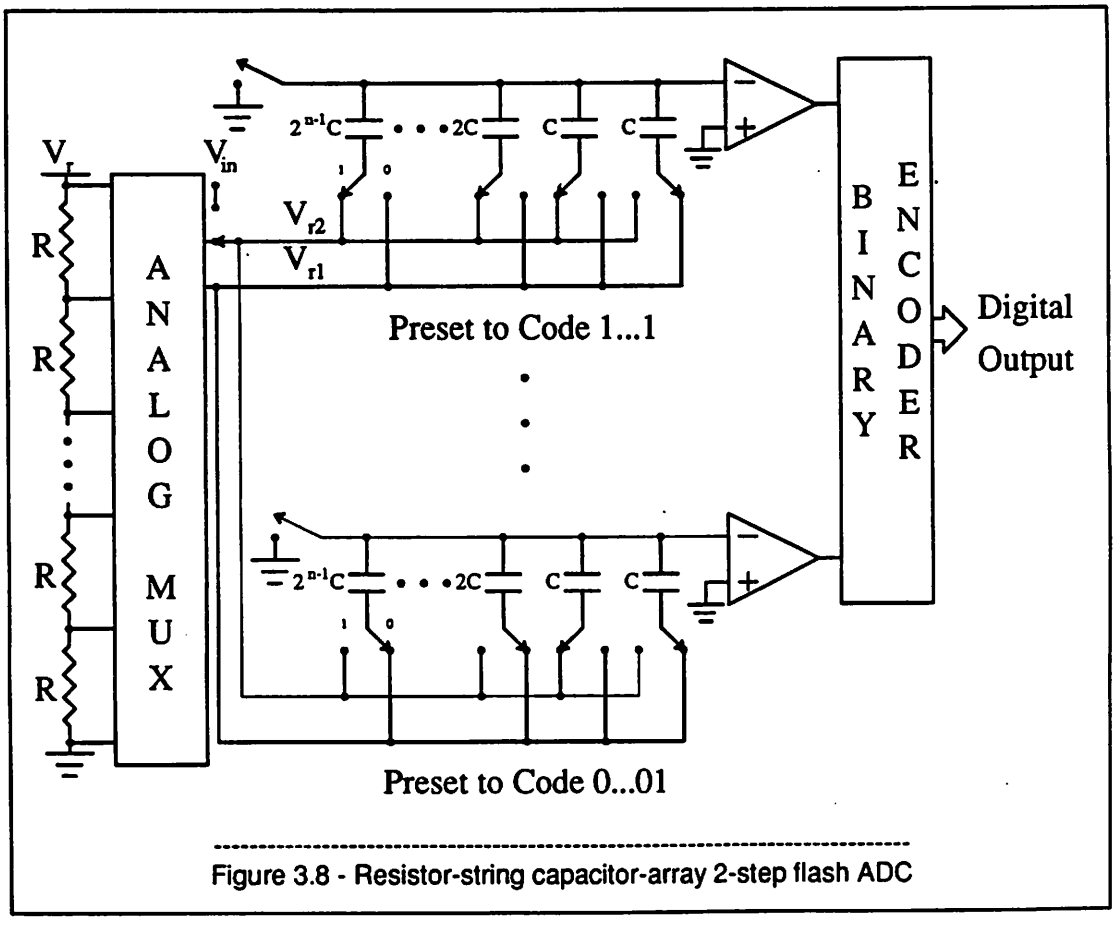


Figure 3.8 - Resistor-string capacitor-array 2-step flash ADC

The operation proceeds in two steps. In the first step the input is sampled and a flash MSB decision is made using the resistor string. Then the LSBs are obtained by subdiving the voltages at the two resistor-string taps adjacent to the unknown input voltage with the capacitor arrays in a second-step flash decision.

8.1. Advantages

The advantages are obvious. The op amps and 2^n matched components are eliminated and the capacitor array makes an inherent S/H. All of the architecture requirements have been met and this architecture has been used in the experimental prototype.

CHAPTER 4

Prototype

1. Introduction

The resistor-string capacitor-array 2-step flash ADC dubbed the "RC" ADC is the choice for the prototype 10-bit, 5-Msample/s ADC. Its block diagram was shown in Chapter 2, Figure 2.8. This chapter will describe the design of the architecture, the high-gain, high-speed comparator, the on-chip digital circuitry and the layout.

The architecture's operation has been described previously and the detailed design will proceed with a discussion of charge-injection errors and solutions, the first-step MSB decision, then the second-step LSB decision.

2. Charge Injection Errors

Although an MOS switch is often said to be an "ideal" switch because the voltage drop across it is zero when no current is flowing, it is hardly "ideal". The charge-injection error caused by the switch is often a limiting factor in switched capacitor designs. Figure 4.1 shows a simple S/H consisting of an input, V_{in}, a switch and a capacitor. On the right the switch has been replaced by an NMOS transistor. While the switch is closed the voltage at the capacitor, V_{out}, is equal to the input voltage. When the switch opens quickly the capacitor voltage is:

$$V_{out} = V_{in} - \frac{C_{ol}}{C + C_{ol}}(V_H - V_L) - \frac{1}{2}\frac{1}{C}(WL_{eff}C'_{ox})(V_H - V_t) \qquad (4.1)$$

V_H and V_L are the highest and lowest gate voltages when the switch is on and off. The last two terms are errors. The first, $\frac{C_{ol}}{C + C_{ol}}(V_H - V_L)$, is clock feedthrough and caused by the gate-drain overlap capacitance. The second is $\frac{1}{2}\frac{1}{C}(WL_{eff}C'_{ox})(V_H - V_t)$ and caused by the channel charge being injected onto the capacitor. The first error is smaller than the second since $L_{ovl} \ll L_{eff}$.

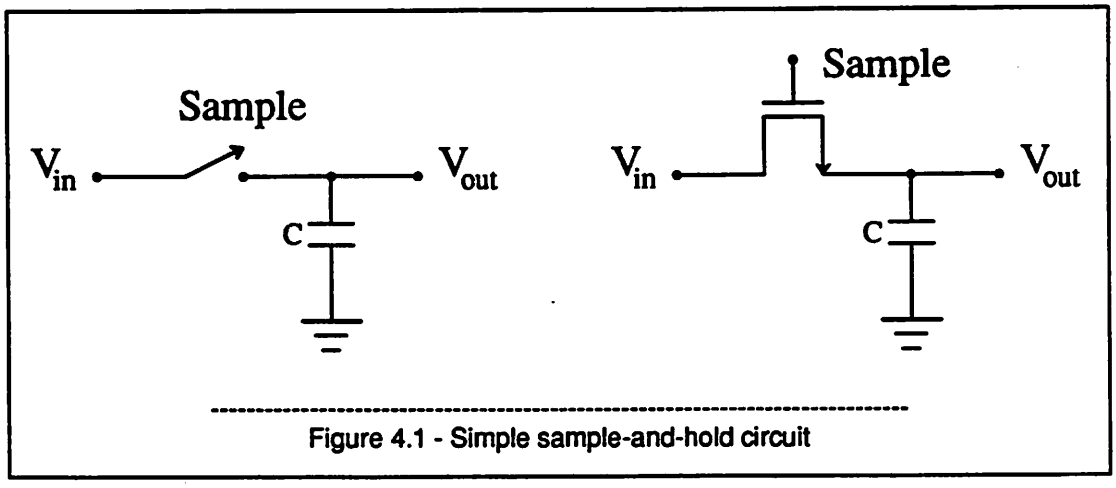


Figure 4.1 - Simple sample-and-hold circuit

From Equation 4.1 it is seen that the errors are technology dependent from the terms C_{ol}, $C_{ox}^{'}$ and V_t and they have components based upon the design parameters W, L, C, V_H and V_L. Even when the design attempts to minimize the charge-injection error often it is too large to tolerate. The two most common circuit solutions to combat it are dummy switches and differential circuitry.

2.1. Dummy Switches

Figure 4.2a shows the simple S/H with the addition of a dummy switch [15] shown shaded.

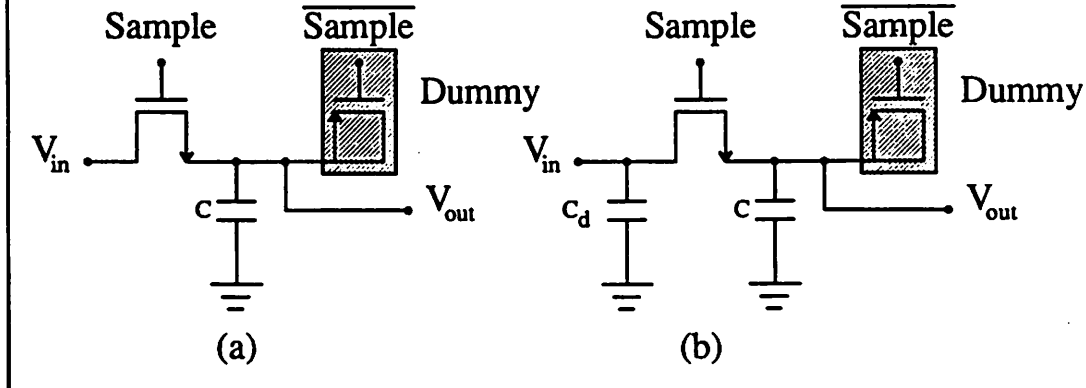


Figure 4.2 - S/H with dummy switch to reduce charge injection errors

The idea is to add the P-channel dummy switch whose charge injection is is equal and opposite to that of the N-channel switch so that the net error is zero. This can be effective but relies heavily upon the clocks "sample" and "$\overline{\text{sample}}$" being exact opposites. It also assumes that half of the channel charge goes into the source and half into the drain. Then a P-switch that is half of the size will have an equal and opposite amount of charge. This is hard to ensure since the devices will have different threshold voltages, the effective channel lengths will differ, etc. The capacitance at the drain and source of the switch are not equal and that will affect how the charge splits. To try and keep it equal a dummy capacitor, C_d, shown in Figure 4.2b is added. This method does cancel some of the charge injection error but is not as effective as a differential architecture.

2.2. Differential Circuitry

Figure 4.3 shows the original S/H but this time everything is duplicated and the inputs and outputs are differential. Ideally any error from switch S1 will be accompanied by an equal error from switch S2. Since the circuits following the S/H only respond to differences in voltages the charge-injection error is common-mode and will have no effect.

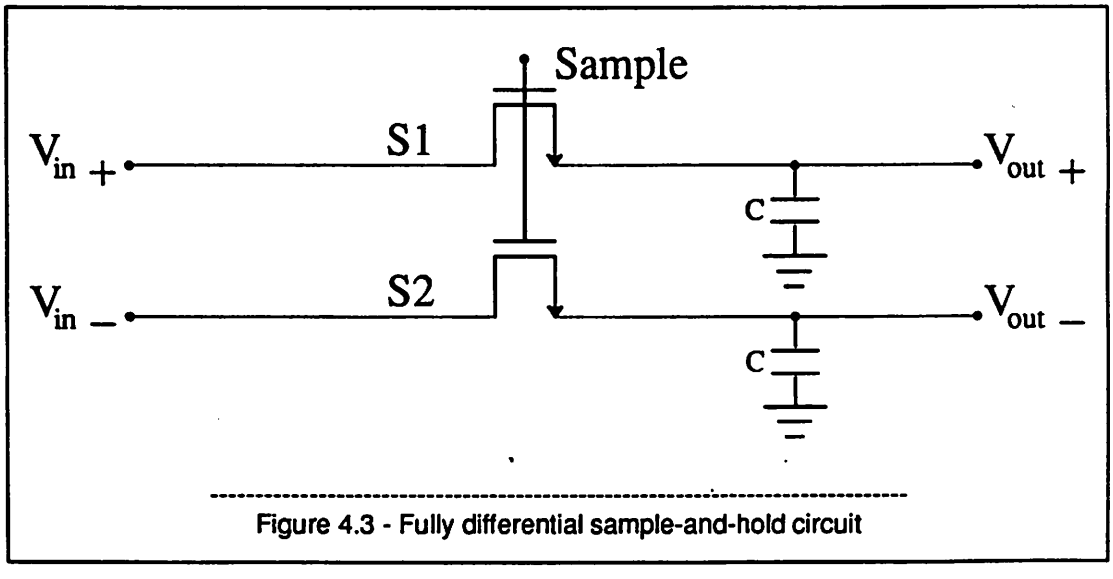


Figure 4.3 - Fully differential sample-and-hold circuit

This method is quite effective. One main limitation is that of how well the technology can match the two switches. In an actual design it is expected that 80-90% of the charge injection error will be eliminated.

One might want to use both dummy switches and a differential architecture. That is a mistake. The dummy switch will have some uncanceled error, so will the differential architecture. These uncorrelated errors will add to make the problem worse than if only one method was used.

The prototype uses a fully differential architecture for all analog circuits and signal paths.

3. First-Step MSB ADC

Figure 4.4 shows the prototype ADC in the MSB conversion mode. It consists of a 32 resistor string, 31 capacitors, 31 comparators, a latch bank and a binary encoder. The operation is similar to standard CMOS flash ADCs. On the first clock phase the input voltage is sampled and the comparator offset voltage canceled. In the second phase the comparators make their decision, their outputs are latched and converted to binary and stored in output latches before being driven off chip.

The resistor string and capacitor (array) will be described with the LSB converter since it is the LSB conversion that sets their design requirements. The comparator design will be detailed in Section 5.

3.1. Latch Bank and Binary Encoder

For the purpose of the MSB converter the "latch bank and binary encoder" block is shown in more detail in Figure 4.5. It consists of the digital latch bank with 31 latches, thermometer code to 1-of-n decoder, binary encoder and 10 output latches.

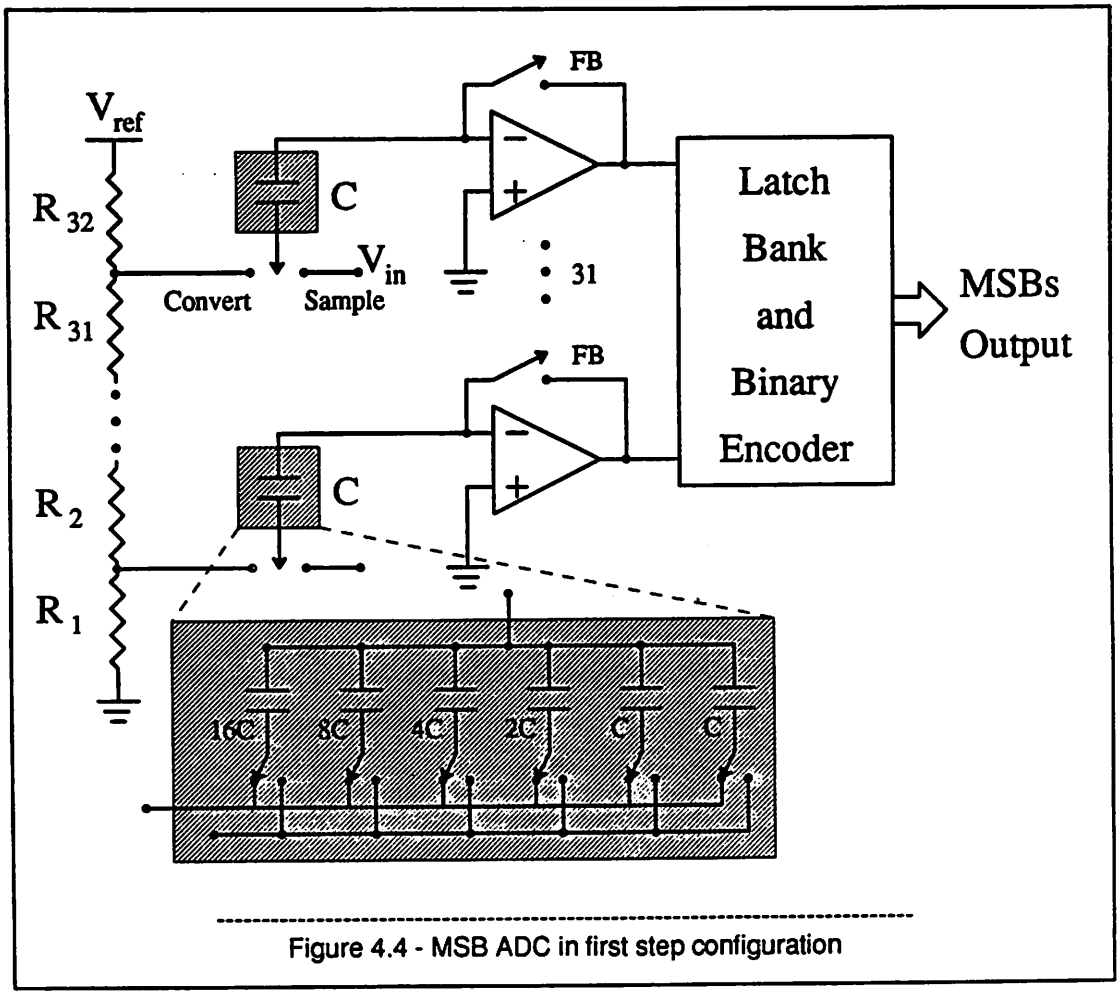


Figure 4.4 - MSB ADC in first step configuration

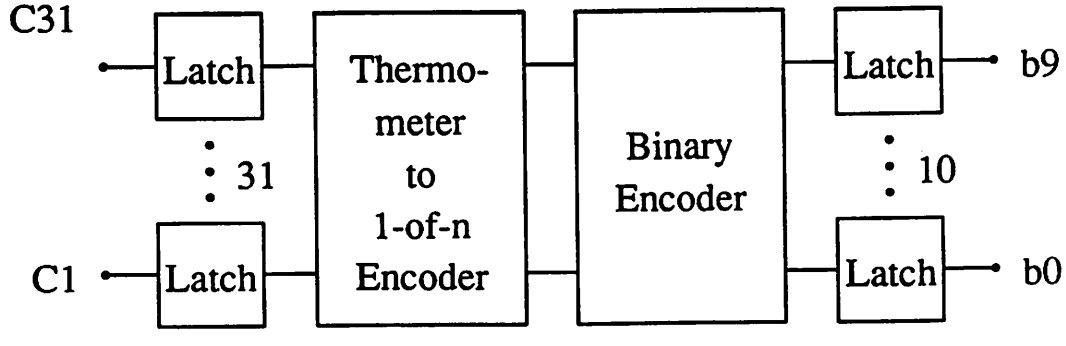


Figure 4.5 - Latch bank and binary encoder

3.2. Thermometer Code to 1-of-n Decoder

The comparator outputs are in a "thermometer code". Starting at the bottom of the column of comparators and going upwards each comparator whose input is greater than the reference input has a 1 output. Then the comparator whose reference is greater than the input has a 0 output as do the comparators above it. This is shown in Figure 4.6.

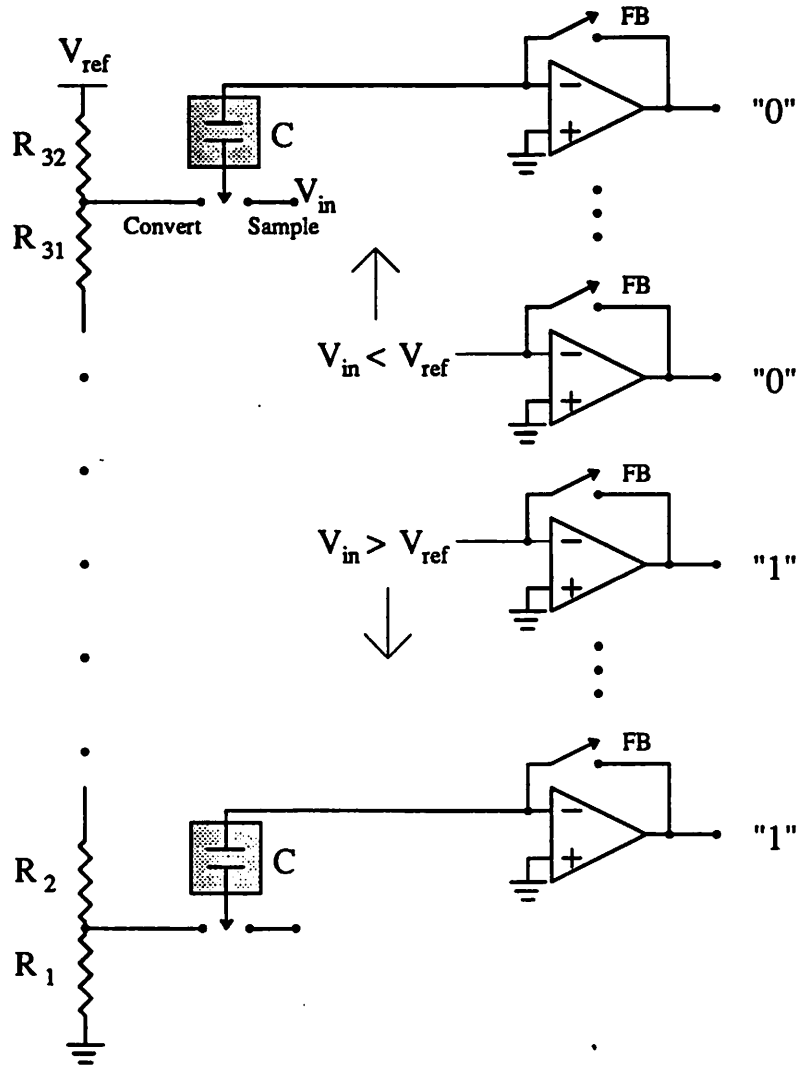


Figure 4.6 - Comparators with thermometer code output

Thermometer code is converted to binary in a two-step process. First it is converted to a 1-of-n code. Then the 1-of-n code is converted to binary. Figure 4.7 shows one circuit for doing the thermometer to 1-of-n decoding. It consists of 2^n-2 XOR gates connected to the 2^n-1 comparator outputs. The k^{th} XOR has inputs connected to the k^{th} and $k+1^{st}$ comparator outputs.

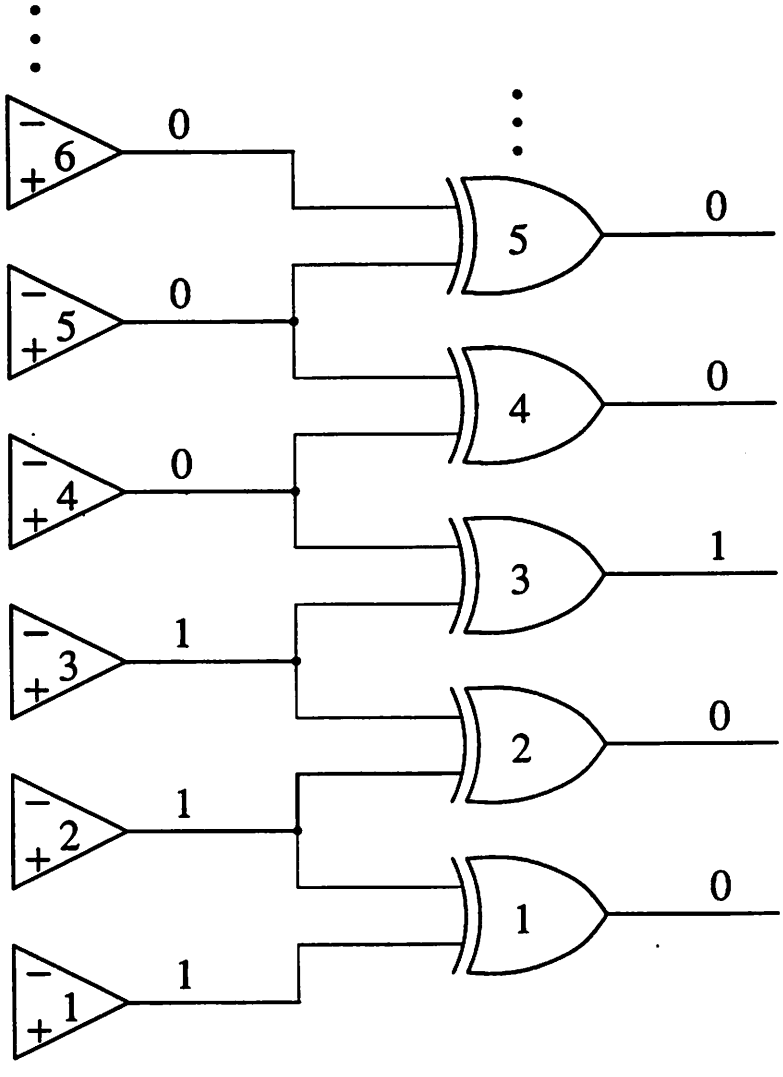


Figure 4.7 - XOR decoder to convert a thermometer code to a 1-of-n code

All of the XOR gates except the one at the 1-to-0 transition have a 0 output. The one at the transition

has a 1 output. The number of the true XOR is the number of the required output code.

There could be a problem with this circuit. Figure 4.8 shows the XOR decoder again but this time comparator 5 has made an error and its output is 0 instead of 1. This could be due to a large offset voltage. Now the 1-of-n decoder has three instead of one true outputs. This will cause the binary encoder to also make an error and the A/D conversion will be in error.

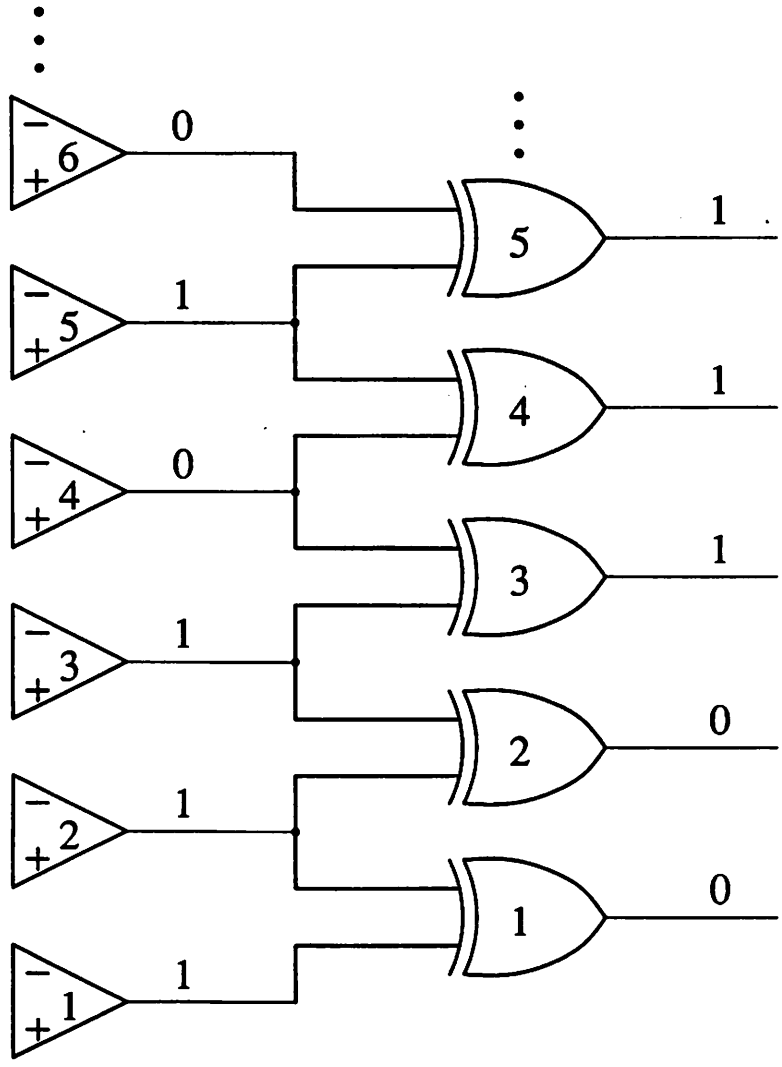


Figure 4.8 - XOR decoder with error

3.2.1. 3-Input NAND gate Decoder

The circuit of Figure 4.9 can correct this problem. It uses 3-input AND gates and looks for the pattern 110 going up the column. With this circuit the 010 error will be suppressed.

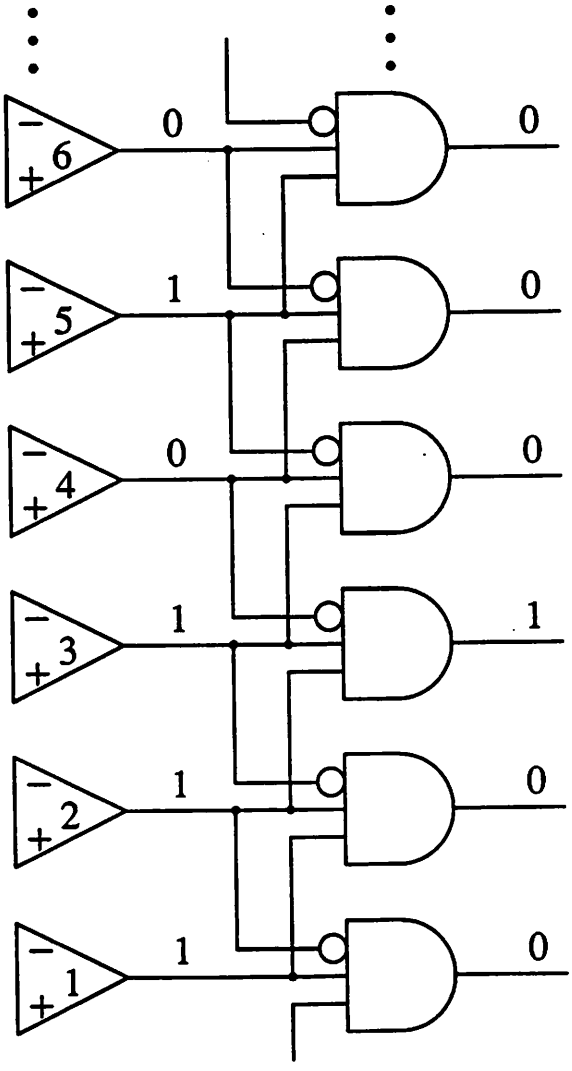


Figure 4.9 - 3-Input AND gate decoder to convert a thermometer code to a 1-of-n code

3.3. Digital Latch Bank

The purpose of the digital latch bank is to pipeline the remaining digital operations thus speeding up the A/D conversion. Figure 4.10 shows the digital latch circuit used repeatedly in the prototype and its driver. The latch consists of a transmission gate in cascade with two inverters. A feedback switch from the first inverter's input to the second's output latches the data bit. The driver generates the "gate" signal to the transmission gate and its complement the "latch" signal to the feedback switch and the driver is also a buffer to drive many latch circuits.

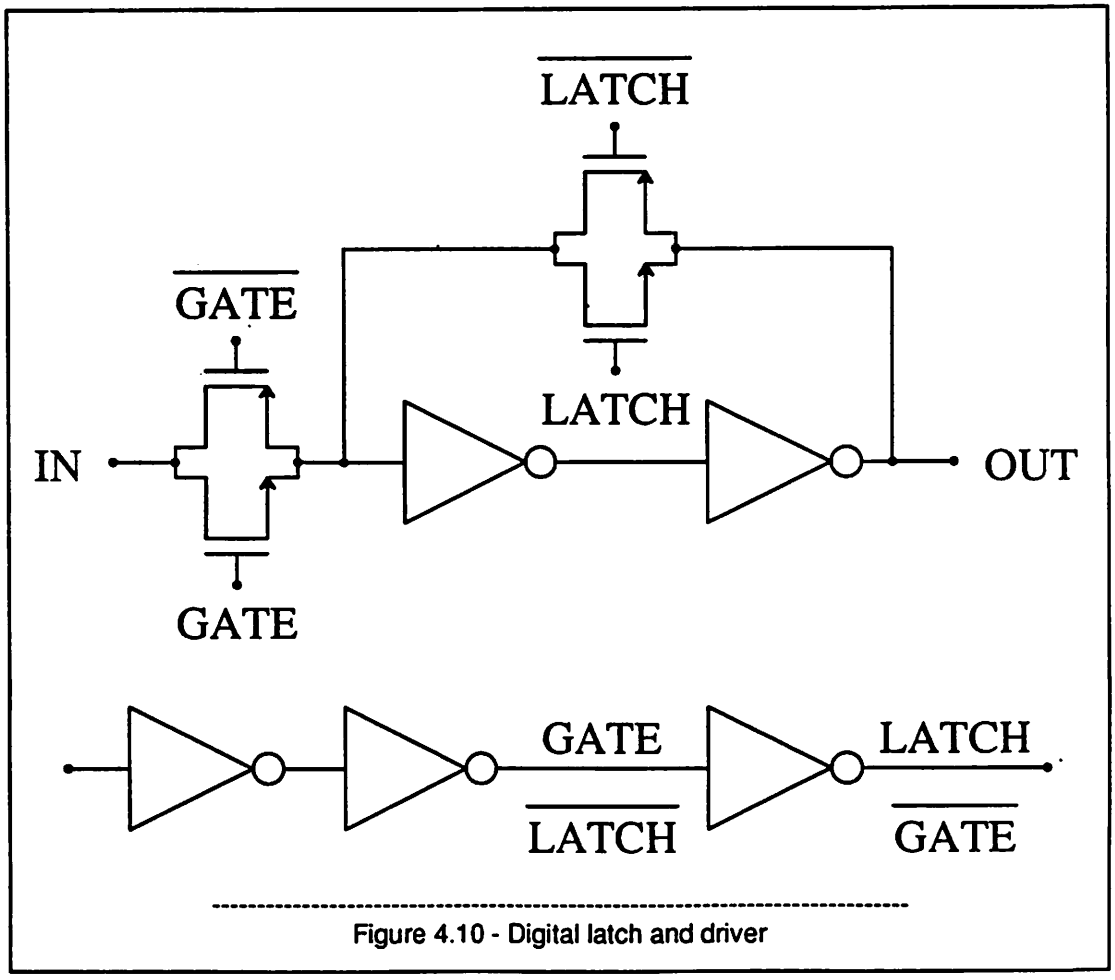


Figure 4.10 - Digital latch and driver

3.4. Binary Encoder

The purpose of the binary encoder is to do the second step of the thermometer code conversion. The 1-of-n code must now be converted to binary. This is done with a ROM as shown in Figure 4.11. The ROM consists of 5 P-channel pull-up transistors taking the 5 column lines high. Each row line is

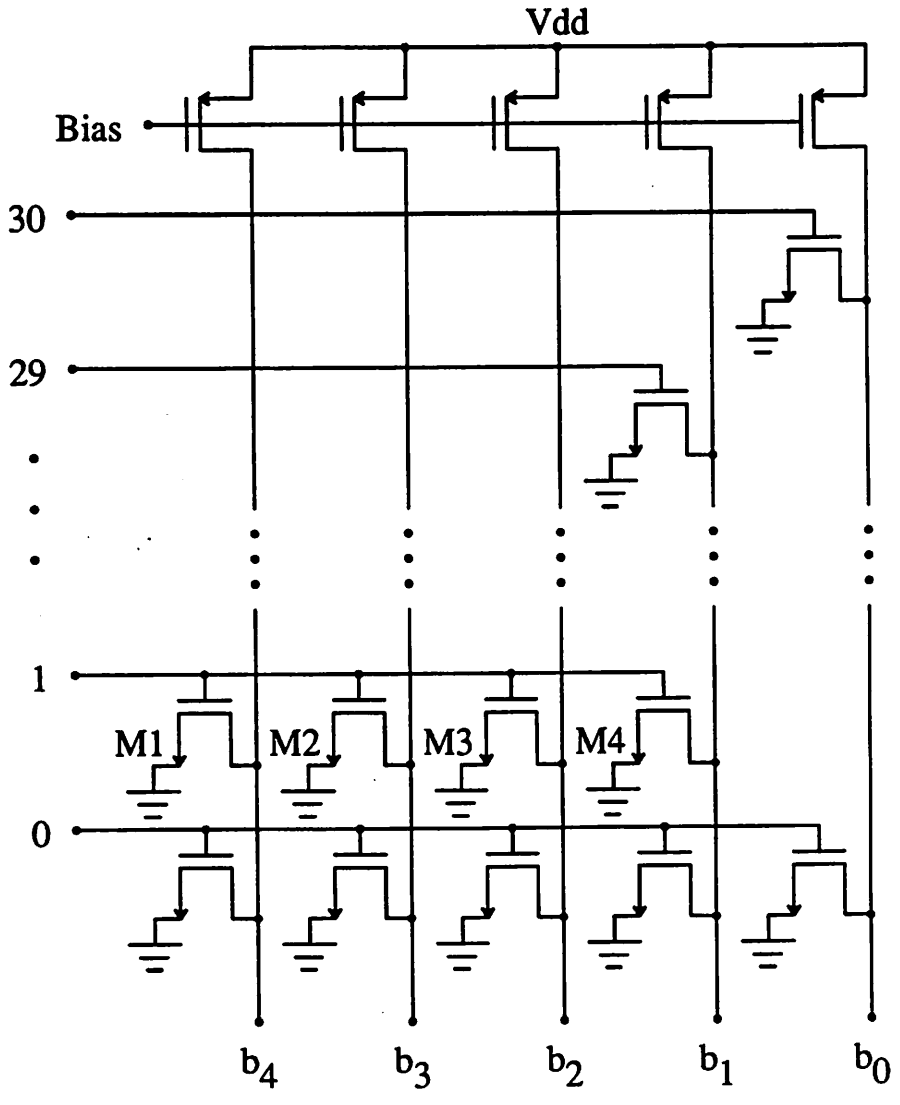


Figure 4.11 - ROM to convert 1-of-n code to a binary output

a line from the 1-of-n decoder and connected to the gate(s) of N-channel pull-down transistors. For instance, for code 00001, input line 1 would be high and transistors M1, M2, M3 and M4 pull bit-lines b1, b2, b3 and b4 low while b0 is high. The ROM output is the desired 00001.

The other input to the ROM is a bias to the P-channel load devices. It is used to set the "on resistance" of the load devices and is nominally 3.15 V.

3.5. MSB Output Latches

The ROM outputs are then latched in the MSB output latches before being driven off-chip. The latches are the same ones previously described. Having on-chip MSB output latches allows much more flexibility in interfacing the chip to the test board since the MSBs are valid for most of the conversion cycle. Without the latches the timing on the test board latches would be much more critical.

4. Second-Step LSB ADC

The LSB ADC must take the MSB code, use that to find the interval in which the unknown input voltage lies and subdivide that interval to find the LSBs. Figure 4.12 shows a block diagram of the LSB ADC. It consists of 31, 1-bit output, sub-ADCs whose inputs are the unknown input voltage and two reference voltages. It also contains a latch bank and a binary encoder. There is other digital control that is not shown.

The operation is as follows. Each of the 31 sub-ADCs has sampled the input at the start of the ADC conversion. V_{r1} and V_{r2} are the taps on the resistor string just above and below the unknown input voltage. Each of the sub-ADCs has a 1-bit output and the threshold is set to:

$$V_{threshold} = V_{r1} + \frac{CODE}{32} \cdot (V_{r2} - V_{r1}) \tag{4.2}$$

CODE is between 1 and 31 inclusive. This is a subranging flash converter. The outputs from the 31 sub-ADCs then are converted to binary almost the same way that the MSBs are done. In actuality the same circuitry is used twice with some small additions for the second step. Also the comparators used in both steps are the same.

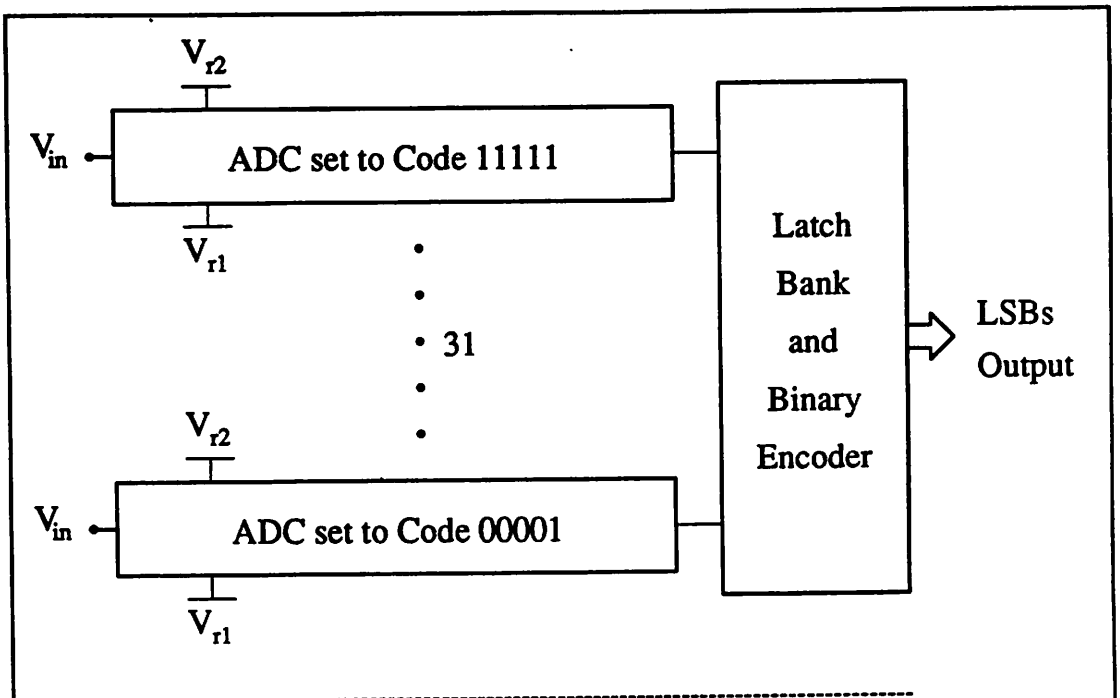


Figure 4.12 - LSB ADC uses 31 sub-ADCs to make a flash decision

4.1. DAC

The DAC used to generate the two voltage references is shown in Figure 4.13. It consists of the resistor string used in the first step and a 2-output multiplexer that selects the k^{th} tap to be V_{r1} and the $k+1^{st}$ tap for V_{r2} for MSB code k.

4.1.1. Resistor String

The two main design problems for the resistor string are the value of each resistor and the layout needed for the desired matching.

The resistance is computed based upon the load capacitance and allowable settling time. The effective capacitive load will be computed in Section 4.2.1.3 and is 10 pF. There is an additional 50 pF in parasitic loading. The time allotted for settling is 6.3 ns and 8 time constants are needed for 10-bit accuracy. This 0.9 ns time constant requires 15Ω output resistance from the DAC. The output resistance of the resistor-string DAC is greatest in the middle of the string and is:

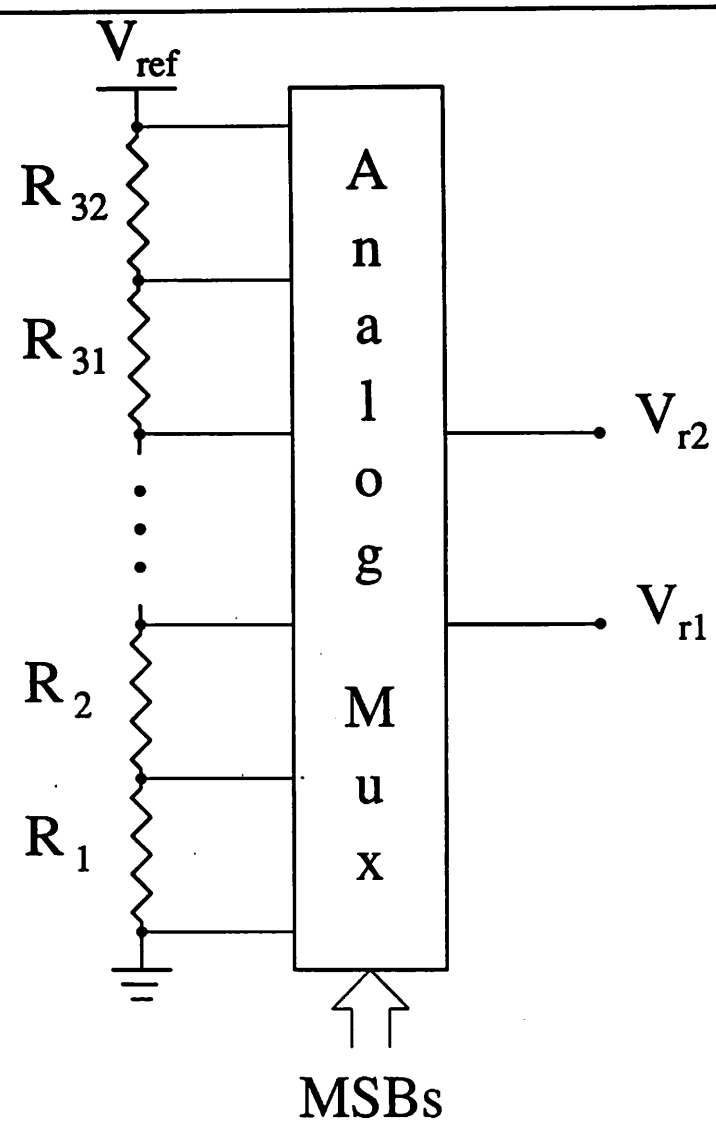


Figure 4.13 - DAC generates 2 reference voltages for LSB conversion

$$R_{out} = \frac{R_t}{2} // \frac{R_t}{2} = \frac{R_t}{4} \tag{4.3}$$

Thus $R_t = 60\Omega$. Each of the 32 resistors then has a resistance of 1.9Ω. This is too low to be practical because the power dissipated in the resistor string would be large.

If however the center-tap of the resistor string is connected to a voltage source which is a virtual ground, as shown in Figure 4.14, the maximum output resistance is now at the quarter and three-quarter points and is reduced to:

$$R_{out} = \frac{R_t}{4} // \frac{R_t}{4} = \frac{R_t}{8} \qquad (4.4)$$

Thus $R_t = 120\Omega$. Each of the 32 resistors then has a resistance of 3.75Ω The resistance is now twice

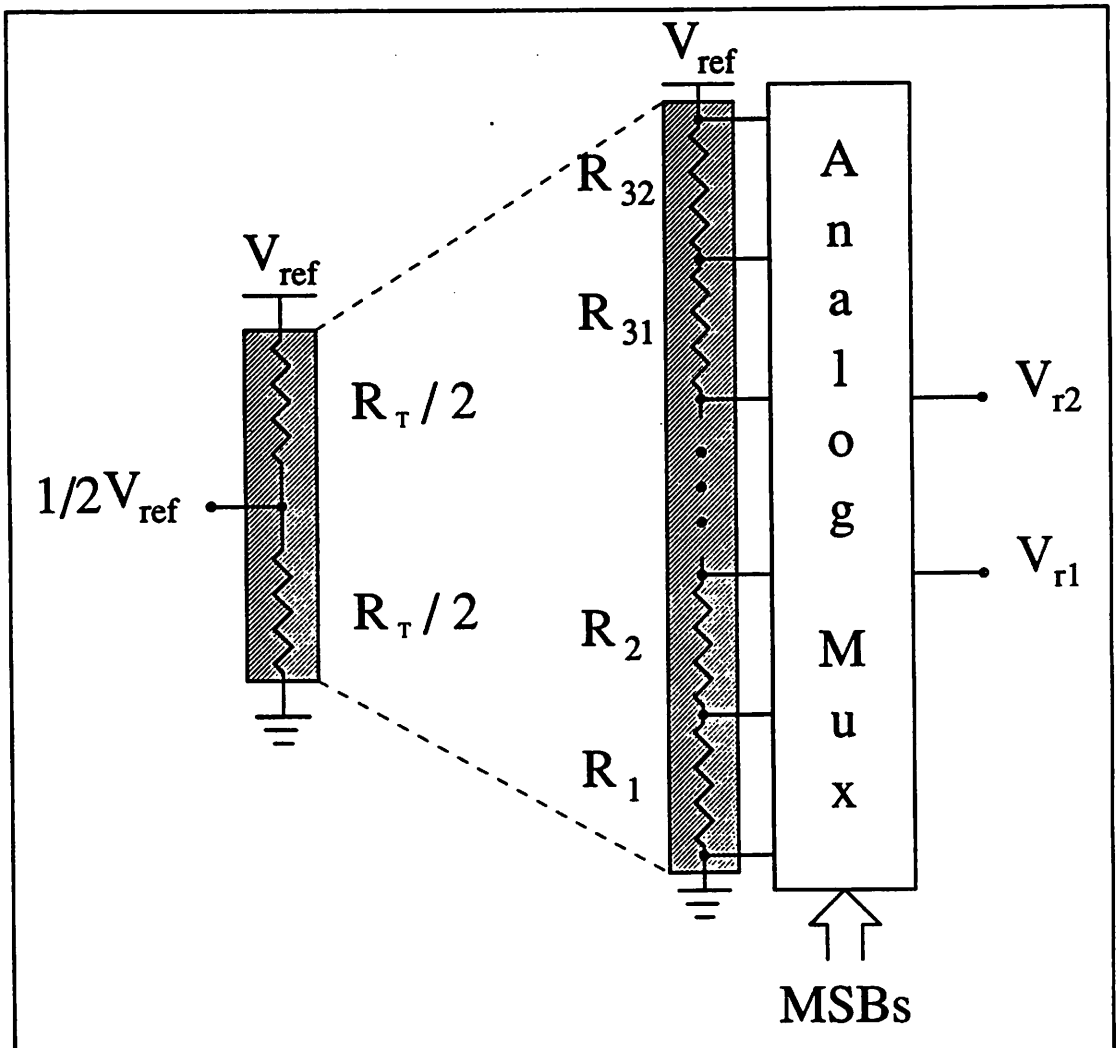


Figure 4.14 - Resistor-string DAC with center tap to reduce output resistance

as large as before.

This process can be repeated once more with voltage sources at the one-quarter, half and three-quarter points reducing the maximum output resistance by a factor of two and allowing a unit resistance of 7.5Ω for a total resistance of 240Ω.

4.1.1.1. Resistor Matching

The 5-bit resistor string sets the over-all linearity of the A/D converter. Even though it is a 5-bit resistor string it needs the full 10-bit accuracy. The resistor matching for n-bit resolution with N-bit linearity is computed.[16]

The reference voltage, V_{ref} is the voltage applied to the resistor string endpoints.

$$V_{ref} = +V_r - (-V_r) \tag{4.5}$$

The absolute value of the tap voltages, V_{tap}, are a fraction, $f(tap)$, of the reference voltage and less than or equal to half of it. The accuracy requirement is that any tap voltage be accurate to within $\pm$½ *LSB* at an N-bit level.

$$V_{tap} = f(tap)V_{ref} \pm \frac{V_{ref}}{2^{N+1}} \tag{4.6}$$

The largest error occurs for the maximum f(tap) which is ½ and occurs at the middle of the string. The fractional output is:

$$\frac{V_{tap}}{V_{ref}} = f(tap)\left[1 \pm \frac{1}{2^{N+1}f(tap)}\right] \tag{4.7}$$

The second term in the parenthesis is the error and is $\frac{1}{2^N}$ at the worst case in middle of the string. In the middle the ratio $\frac{V_{out}}{V_{ref}}$ is:

$$\frac{V_{out}}{V_{ref}} = ½\left[1 \pm \frac{1}{2^N}\right] \tag{4.8}$$

This ratio of voltages is the ratio of the resistance of half of the string to the total resistance. Summing the resistors in each term:

$$\frac{V_{out}}{V_{ref}} = \frac{\sum_{i=1}^{i=2^{n-1}} R_i}{R_t} = \frac{\sum_{i=1}^{i=2^{n-1}} R_i}{\sum_{i=1}^{i=2^{n-1}} R_i + \sum_{i=2^{n-1}+1}^{i=2^n} R_i} = \frac{1}{2}\left[1 \pm \frac{1}{2^N}\right] \tag{4.9}$$

Assume that the resistor values are normally distributed with mean, R, and standard deviation σ_R. Then the sum of resistors will also be normally distributed with a mean equal to the sum of the means and the standard deviation equal to the square root of the sum of the squares of the standard deviations of the resistors. If any resistor is assumed to be made up of a mean value and a standard deviation then $R_i = R + \sigma_R$ and Equation 4.9 can be written as:

$$\frac{2^{n-1}R \pm 2^{\frac{n-1}{2}}\sigma_R}{2^{n-1}R \pm 2^{\frac{n-1}{2}}\sigma_R + 2^{n-1}R \pm 2^{\frac{n-1}{2}}\sigma_R} = \frac{1}{2}\left[1 \pm \frac{1}{2^N}\right] \tag{4.10}$$

The numerator and left term in the denominator are the same term and their sign is taken as positive. To maximize the error effect the negative sign is taken in the right denominator term. With these conventions:

$$\frac{1}{2}\left[1 + 2^{\frac{-n+1}{2}}\frac{\sigma_R}{R}\right] = \frac{1}{2}\left[1 \pm \frac{1}{2^N}\right] \tag{4.11}$$

This can be simplified to show that $\frac{\sigma_R}{R}$, the maximum mismatch for less than ½ *LSB* error is:

$$\frac{\sigma_R}{R} = 2^{\frac{n-1}{2}-N} \tag{4.12}$$

This is where n is the resolution and N is the linearity in bits. For the case $n=N$, where n-bit precision is needed with N-bit accuracy:

$$\frac{\sigma_R}{R} = 2^{\frac{n-1}{2}-n} = 2^{\frac{-n-1}{2}} = \frac{1}{\sqrt{2}\sqrt{2^n}} \tag{4.13}$$

For the prototype, 5-bit resolution and 10-bit precision is needed and $\frac{\sigma_R}{R}$ is ≤ 0.23%

4.1.1.2. Resistor-String Layout

Normally the resistor string would be made as shown in Figure 4.15a. The example is for a 3-bit fully differential ADC. Since the architecture is fully differential comparator 1 uses tap 1 and tap 7.

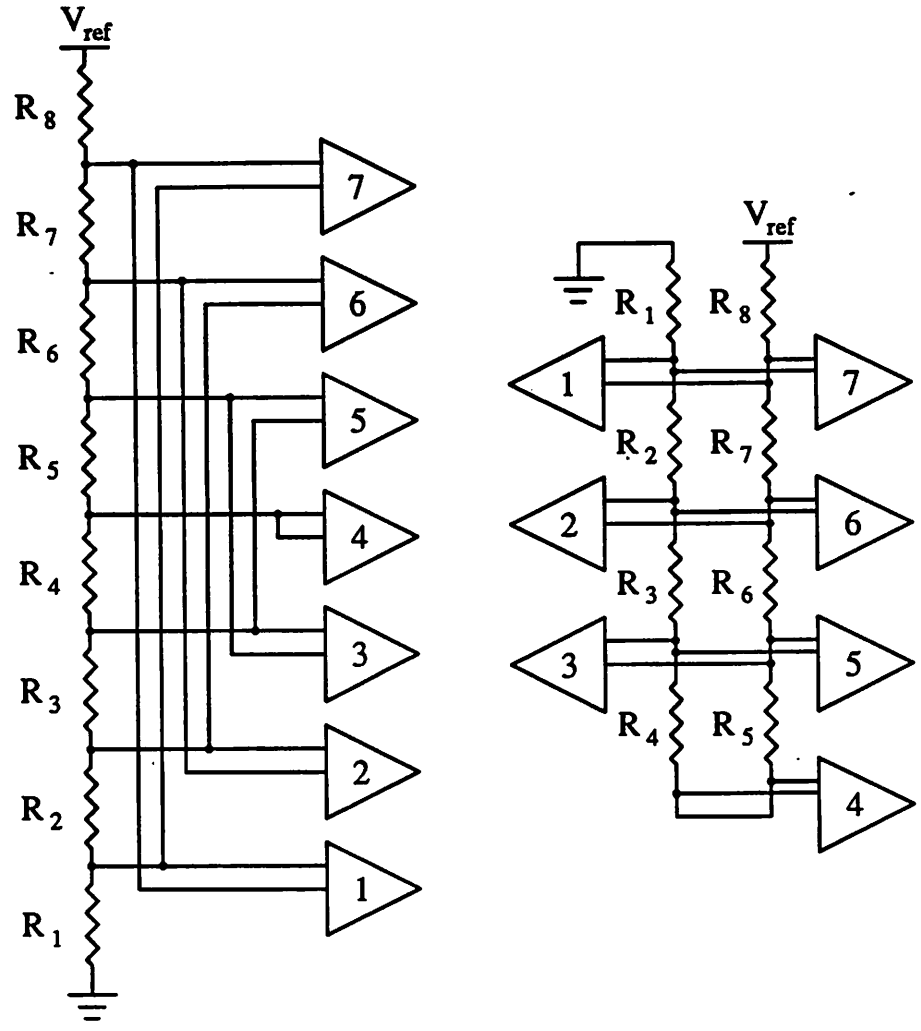


Figure 4.15 - Folded and unfolded resistor strings for MSB ADC

Each comparator needs a tap from the positive half of the string and the negative half of the string. The routing requires considerable area.

If the resistor string is folded as shown in Figure 4.15b the routing becomes much easier. The drawback to folding is the problem of matching the resistors around the fold. They are not discrete resistors connected by very low resistance wire but a continuous resistor with taps along it. The other

disadvantage is that the comparators are now in two banks instead of 1 column. They may mismatch because of gradients in processing or the clock signals may not be distributed with equal delays.

The actual resistor string is made up of 7.5Ω resistors made of first level metal that is .15Ω/□ molybdenum-titanium. This layer had the best matching of polysilicon, metal1 and metal2. It is 35 μm wide to get the required 10-bit matching and laid out in a "snake" pattern.

The first and last resistors will mismatch the rest of the resistors since they have a contact made to them that is different than the taps going to the comparators. This is shown in Figure 4.16a. A solution is to add an extra resistor at each end of the string. This moves the problem out one tap. This tap is unused so the problem is eliminated. The only drawback is that the voltage reference is attenuated by $\frac{2R}{R_t}$. This is not important and only causes a known gain adjustment in the overall conversion.

4.2. LSB sub-ADC

The LSB ADC is made up of 31 sub-ADCs that have a 1-bit output and threshold that can be preset to 1 of 31 codes. Figure 4.17 shows the sub-ADC schematic. It consists of a 5-bit binary-weighted capacitor array and a comparator. The capacitors are the same single capacitor shown in Figure 4.4 for the first step and have sampled the input voltage in the first step. The operation is the same as the subranging Q-R ADC in Chapter 3, Section 6 except that the bottom-plate switches are preset to the specified code and there is only one decision made comparing V_{in} to $V_{r1} + (V_{r2}-V_{r1})\cdot CODE$.

4.2.1. Capacitor Array

The capacitor array design is similar to that of the resistor-string in that the capacitor values and matching requirements are the important design issues. The capacitors in any array need 5-bit matching. There is the added requirement that all 31 arrays match each other.

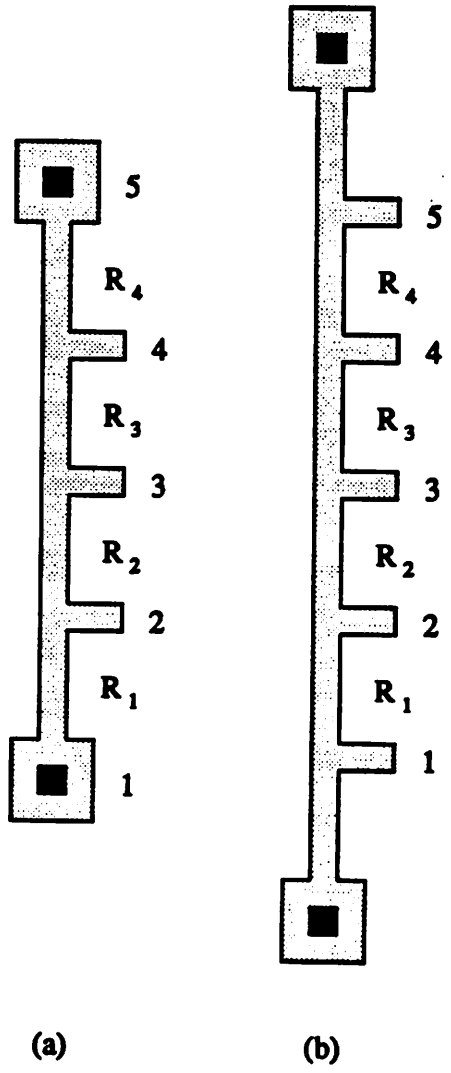


Figure 4.16 - End-point mismatch in resistor string

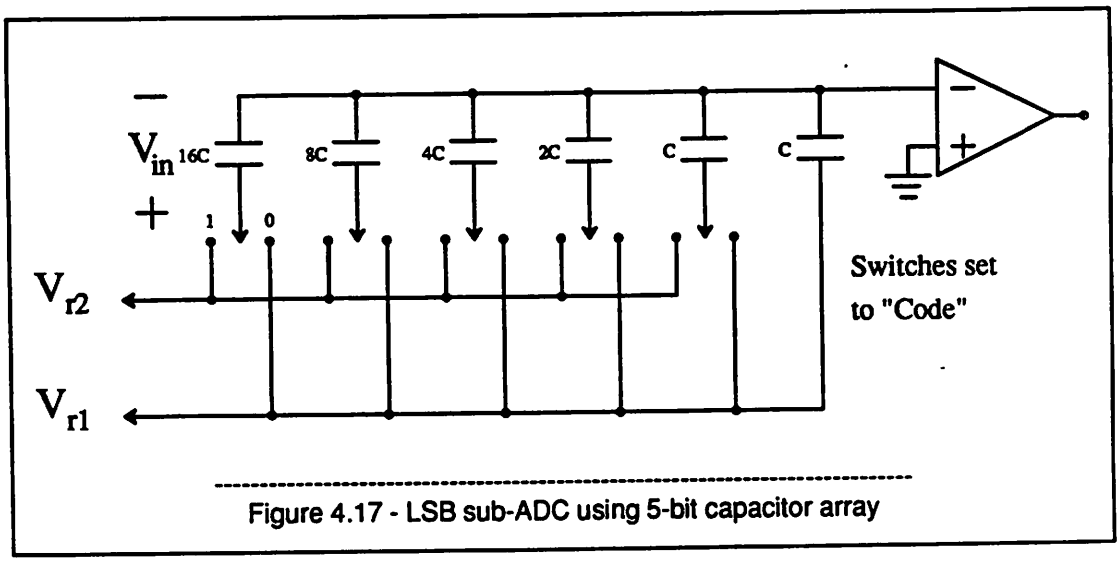


Figure 4.17 - LSB sub-ADC using 5-bit capacitor array

4.2.1.1. Capacitor Matching

The capacitor matching in an array is analogous to the resistor-string matching analysis of Section 4.1.1.1.

$$\frac{\sigma_C}{C} \leq \frac{1}{\sqrt{2}\sqrt{2^n}} \tag{4.14}$$

With this analysis the capacitors mismatch in value within 1σ so 68.3% of the fabricated single arrays will have the required matching. However each working die needs 31 arrays with all matching to within 1σ.

4.2.1.2. Multiple Capacitor-Array Matching

When $\frac{\sigma_C}{C} \leq \frac{1}{\sqrt{2}\sqrt{2^n}}$ 68.3% of the single arrays fabricated will match. Thus for every 1-bit increase in resolution and precision $\frac{\sigma_C}{C}$ must decrease by a factor of $\frac{1}{\sqrt{2}}$.

To find the "equivalent matching" for all 31 arrays to match let k be the probability that 1 array has the required matching and the 31 arrays have the same 68.3% chance of all matching.

$$(k)^{31} = 0.683 \tag{4.15}$$

k = 0.9877. This corresponds to 3.24σ. Now rather than using "1-sigma matching", "3.24-sigma matching" is required so $\frac{\sigma_C}{C}$ must decrease by $\frac{1}{3.24}$. Let m be the equivalent matching needed.

$$\frac{1}{3.24}\frac{\sigma_C}{C} = \frac{1}{\sqrt{2}\sqrt{2^m}} \tag{4.16}$$

Substituting Equation 4.14 for $\frac{\sigma_C}{C}$ and simplifying:

$$2^m = (3.24)^2 \cdot 2^n \tag{4.17}$$

$$m = n + \frac{2ln(3.24)}{\ln(2)} = n + 3.40 \tag{4.18}$$

Thus for 5-bit matching in all 31 arrays the capacitors need the equivalent of 8.4-bit matching.

This matching is easily attainable in the process used with minimum size capacitors. The unit capacitors are 9.2 μm x 9.2 μm. With a capacitance of 0.07 fF/μ^2 each unit capacitor is 59 fF and the 31-capacitor array is 1.8 pF total. The capacitor-array layout is shown symbolically in Figure 4.18. It is quasi-common-centroid with capacitors 16C and 8C split into two equal parts and placed on opposite sides of the array with the unit capacitors, 2C and 4C capacitors placed in the center. The array is then surrounded by dummy capacitors so that the edge etching is the same for all of the capacitors in the array.

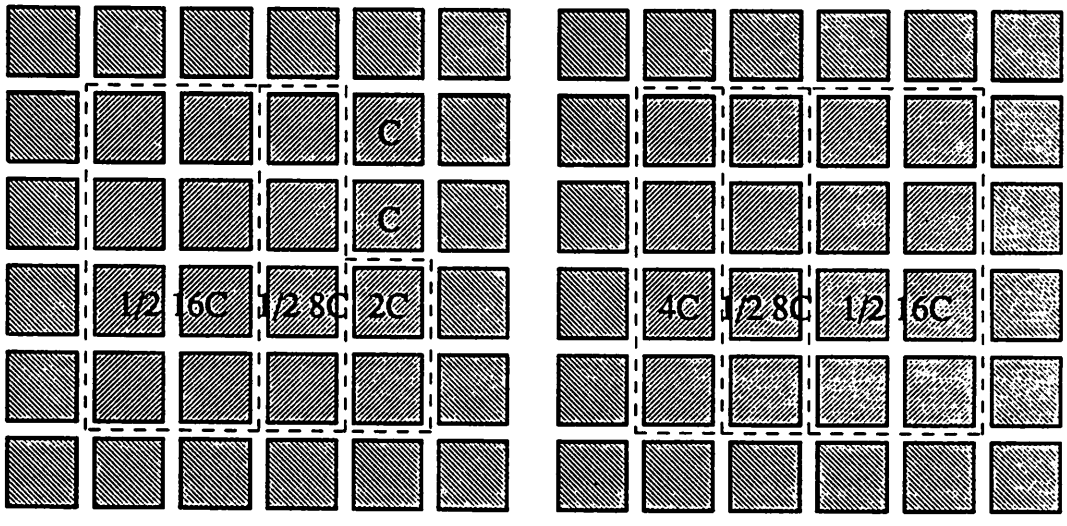


Figure 4.18 - Capacitor-array layout surrounded by dummy capacitors

The capacitor construction is a bottom plate of metal1, a combination oxide and nitride dielectric and a metal2 top plate. The contact to the top plate is made with a via from metal2. Since only 8-bit accuracy is needed problems with nitride in the dielectric such as dielectric relaxation or residual polarization should not be significant.

4.2.1.3. Capacitor-Array Load

When the LSB conversion is started all 31 capacitor arrays are placed in parallel and loading the DAC outputs. The total capacitance is the sum of the capacitance of the 31 arrays. Each array is preset to one of the codes from 1 to 31 so its capacitance is the series combination of two capacitors of value *CODE* and 32–*CODE*.

$$C_{total} = \sum_{i=1}^{i=31} iC // (32-i)C \tag{4.19}$$

which is:

$$C_{total} = \sum_{i=1}^{i=31} \frac{iC(32-i)C}{32C} \tag{4.20}$$

This can be simplified since the summation is symmetric.

$$C_{total} = \sum_{i=1}^{i=15} 2\left[\frac{iC(32-i)C}{32C}\right] + \frac{16 \cdot 16}{32} = 170.5C \tag{4.21}$$

With 59 fF unit capacitors C_{total} = 10 pF.

4.2.1.3.1. Capacitor-Array Settling Time

When the 31 capacitor arrays are switched to the resistor string at the start of the second step the charge on the capacitors must redistribute. The time needed to settle is not as severe as if the arrays were being charged since they are already precharged to V_{in} which is close to V_{r1} and V_{r2} where they will be when the charge is redistributed.

For a simple R-C circuit with an initial voltage on the capacitor, $V(0)$, the output voltage is:

$$V_{out}(t) = V_{in}\left[1 - e^{-t/\tau}\right] + V(0)e^{-t/\tau} \tag{4.22}$$

Grouping terms this is:

$$V_{out}(t) = \left[V(0) - V_{in}\right] e^{-t/\tau} + V_{in} \tag{4.23}$$

Let $V(0) = V_{r1}$ and $V_{in} = V_{r2} = V_{r1} + \Delta V_r$. Then $V(0) - V_{in} = -\Delta V_r$.

$$V_{out}(t) = V_{in} - \Delta V_r e^{-t/\tau} \tag{4.24}$$

To see how many time constants are needed set $V_{out}(t) - V_{in} = \frac{1}{2}\, LSB = \frac{V_{fs}}{2 \cdot 1024}$ and $\Delta V_r = \frac{V_{fs}}{32}$.

$$\frac{V_{fs}}{2 \cdot 1024} \le \frac{V_{fs}}{32}\left[e^{-t/\tau}\right] \tag{4.25}$$

Less than 5 time constants (almost 4) are needed for this. If the capacitor array was uncharged it would require 8 time constants to settle to 0.035% of its final value which is less than a ½ LSB at a 10-bit level. Since the array is precharged to V_{in}, 3 time constants are saved.

4.3. 2-Output Analog Multiplexer

The 2-output analog multiplexer gets the MSB code from the first decision and connects tap(MSB) to V_{r1} and tap(MSB+1) to V_{r2}. These voltages are also referred to as V_n and V_{n+1} where $|V_{n+1}| > |V_n|$. In addition the complementary taps, V'_{n+1} and V'_n, for the negative side are needed since the architecture is fully differential. As before the $|V'_{n+1}| > |V'_n|$ for the complementary voltages. Thus 4 outputs are needed. The simplest multiplexer scheme would be four independent multiplexers.

4.3.1. Binary Tree Multiplexer

A simple multiplexer is the binary tree design shown in Figure 4.19. Its largest disadvantage is the large series resistance it presents since there are k switches in series from an input to the output when there 2^k inputs. With a 60 pF load and 5 ns allotted for settling to 5 time constants the resistance must be less than 17 Ω including a switch to the capacitor and the resistor string resistance.

A MOS transistor in the ohmic region has the following I_d versus V_{ds} relationship:

$$I_d = \frac{k'}{2}\frac{W}{L}\left[2(V_{gs} - V_t)V_{ds} - V_{ds}^2\right] \tag{4.26}$$

In the ohmic region the first term is much greater than the second so:

$$I_d \approx \frac{k'}{2}\frac{W}{L}\left[2(V_{gs} - V_t)V_{ds}\right] \tag{4.27}$$

The "on resistance", R_{on}, is $\frac{V_{ds}}{I_d}$.

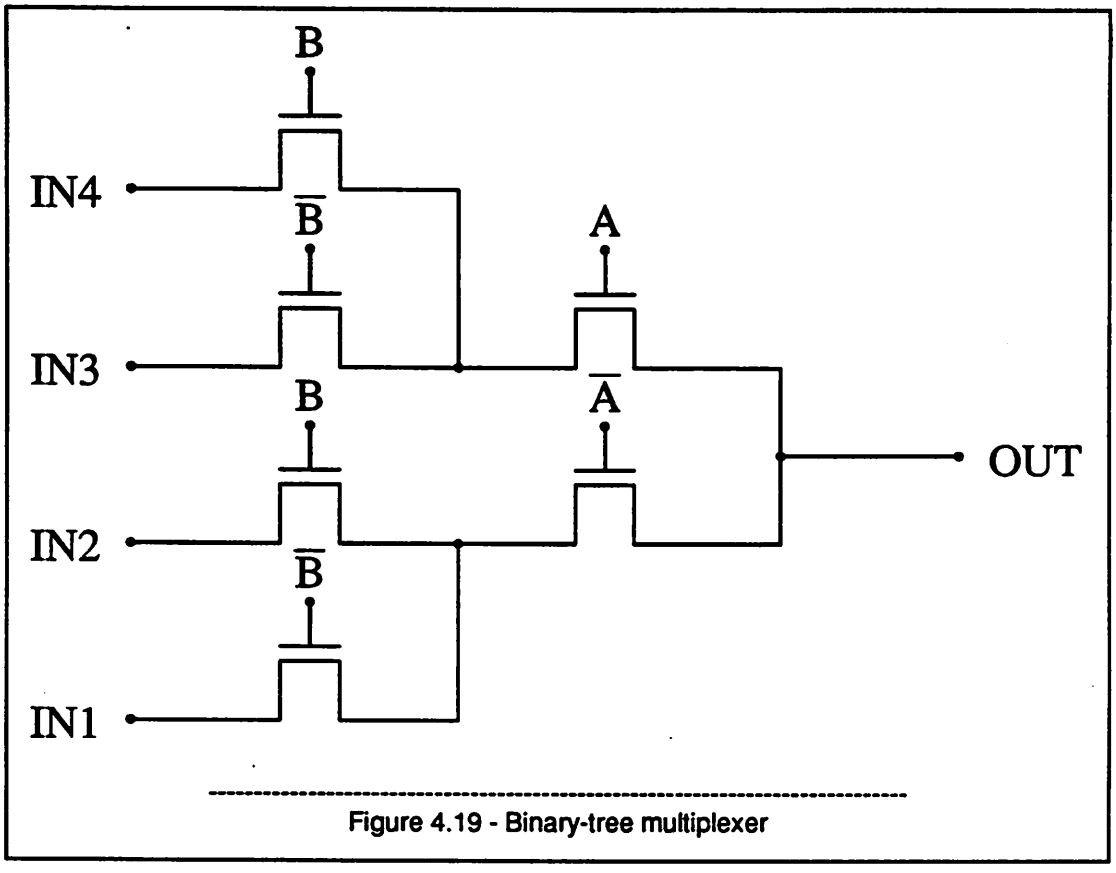


Figure 4.19 - Binary-tree multiplexer

$$R_{on} = \frac{1}{k'\frac{W}{L}\left[V_{gs}-V_t\right]} \tag{4.28}$$

An N-channel switch with 160 µm / 1.6 µm W/L and 1.5 V ($V_{gs}-V_t$) will have 1.1 $k\Omega$ resistance.

Having five switches with this large 5.5 $k\Omega$ resistance in series will slow down the conversion too much.

A second disadvantage is that it requires a binary input. This would slow down the ADC since the binary output is the last thing determined in the MSB conversion. As soon as the thermometer code is converted to 1-of-n code the LSB conversion could be started.

4.3.2. Linear Multiplexer

The opposite of a binary tree multiplexer is the linear multiplexer shown in Figure 4.20. It consists of k switches, one from each input to a common output line. It has the advantage that the series resistance is from only one switch and it uses the "1-of-n" code as the selecting input. To get the required 17 Ω resistance the P-switches need to have a 1280 µm / 1.6 µm W/L and a 400 µm / 1.6 µm W/L for the N-devices.

There is a big limitation in that the output is loaded by the parasitic junction capacitance of 33 switches. The junction capacitance of a switch shown in Figure 4.21a is given by:

$$C_j = 4.4WC_j' + C_{jsw}'(4.4W + 8.8) \qquad (4.29)$$

$C_j' = 0.5\,fF/\mu^2$ and $C_{jsw}' = 0.38\,fF/\mu$ at zero bias. This 400 µm / 1.6 µm N-channel switch will have

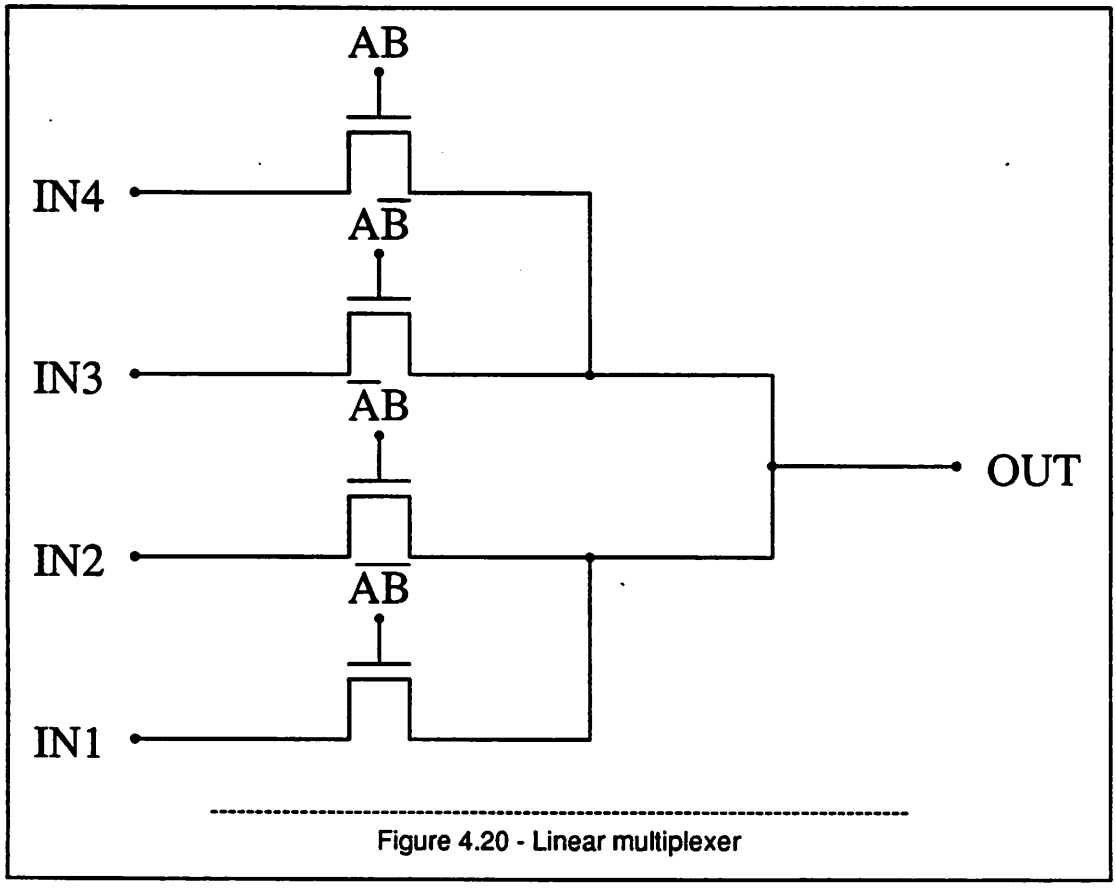


Figure 4.20 - Linear multiplexer

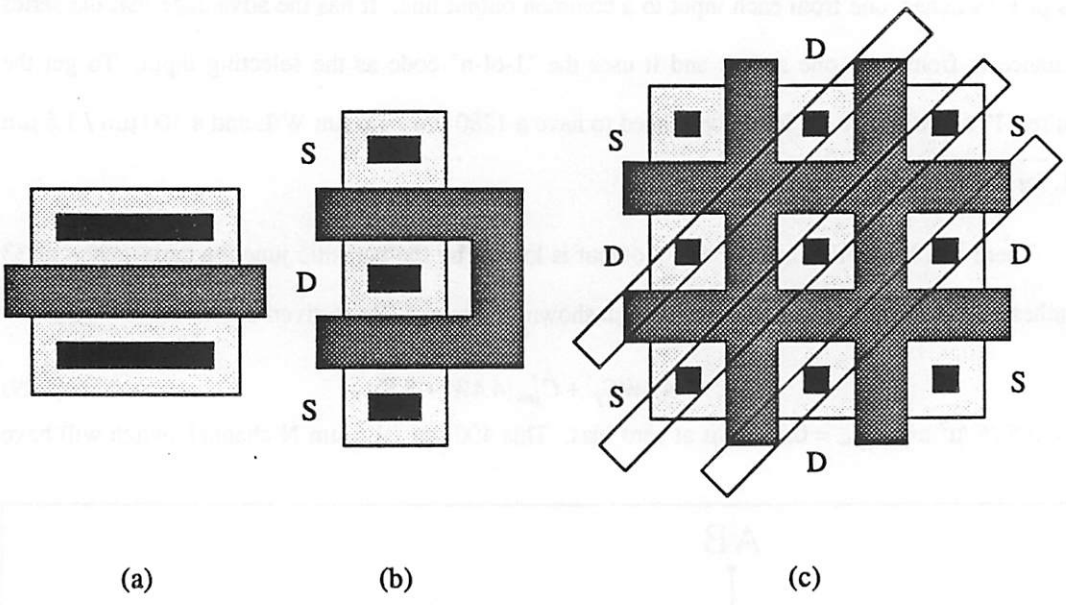


Figure 4.21 - Junction capacitance minimizing layouts

1.5 pF of parasitic junction capacitance and all 33 of them will have 50 pF. This is clearly much too large a load and it is over 3 times worse for the P-channel switches.

Figure 4.21b shows the same width device made by folding the switch upon itself once and sharing the source between the two halves. Its junction capacitance is:

$$C_j = \left[\frac{1}{2}\right] 4.4WC_j' + C_{jsw}'(8.8) \qquad (4.30)$$

It is approximately half of the unfolded junction capacitance. Devices with a large width are very rectangular and can be folded many times to make them closer to a square. Notice that the edges not adjacent to a gate have wasted capacitance. If the "folding" was done in two dimensions the maximum amount of gate width for the minimum source capacitance will be obtained. This is the "checkerboard" or "waffle" layout in Figure 4.21c. The junction capacitance has very little influence from the parasitic sidewall component and the result is:

$$C_j = \left[\frac{1}{4}\right] 4.4WC_j' \tag{4.31}$$

It has less than one quarter the capacitance of the unfolded layout. This structure is used for the large multiplexer switches.

Even with the junction capacitance lowering by a factor of four using the checker-board layout the linear decoder has too much parasitic junction capacitance.

4.3.3. 2-Level Linear Multiplexer

The binary tree decoder had too much resistance and the linear decoder had too much capacitance. A compromise between the two architectures is a 2-level linear multiplexer shown in Figure 4.22. It consists of 4 linear decoders doing the first selection and then another 4 input linear multiplexer to the final selection. Note that 4, not the 2 shown, are needed since there are 4 outputs.

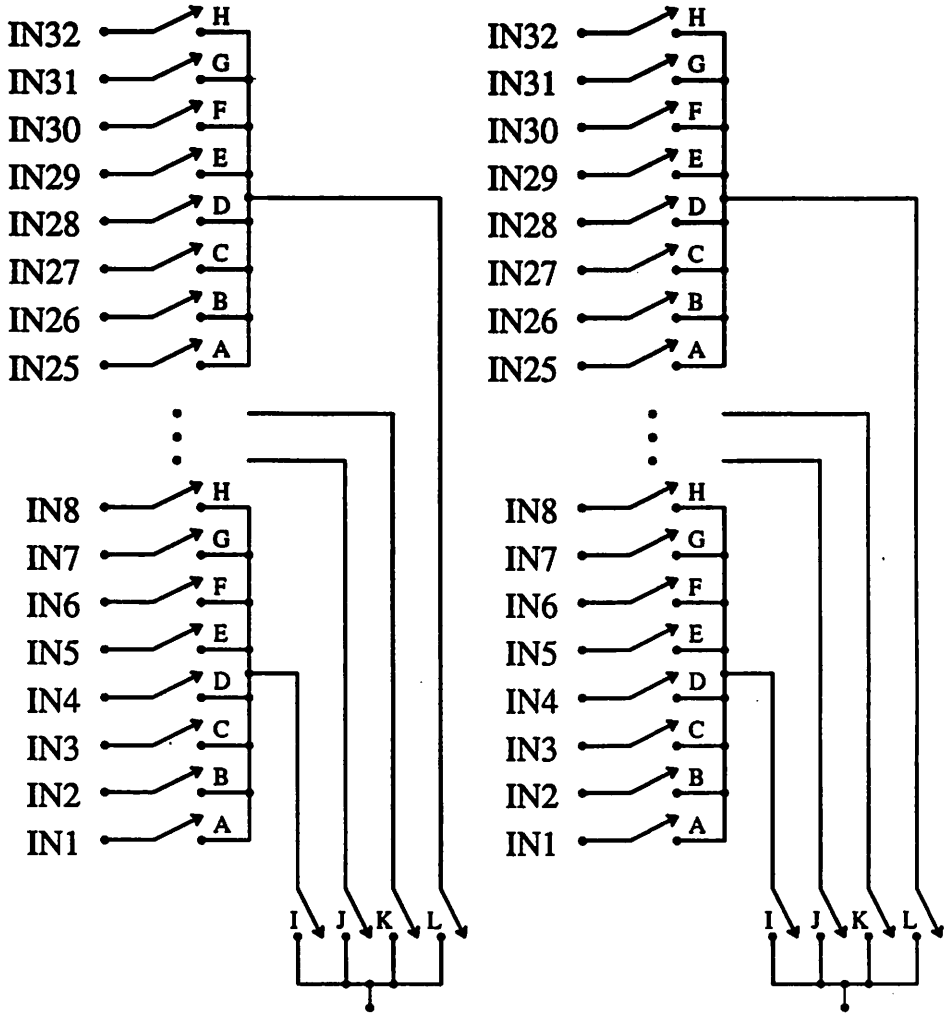


Figure 4.22 - 2-Level linear multiplexer

4.3.4. Reducing Multiplexer Redundancy

Although the 2-level linear multiplexer helped to alleviate the on-resistance and capacitive loading problems of the arrangement of Figure 4.19 or 4.20 with 4 full multiplexers it is clearly inefficient and a waste of area. The first improvement is to recognize that the multiplexers that select the positive taps and the multiplexers that select the complementary taps (for the differential architecture) are

exclusive (except for the ground tap) and each can be cut in half. This is shown in Figure 4.23. All that needs to be done is to be able to select which output is the higher potential. This can be done in the second step. Thus each multiplexer can have half of the inputs and half of the parasitic loading capacitance. (Note: 2 multiplexers are in Figure 4.23. 4 are needed.)

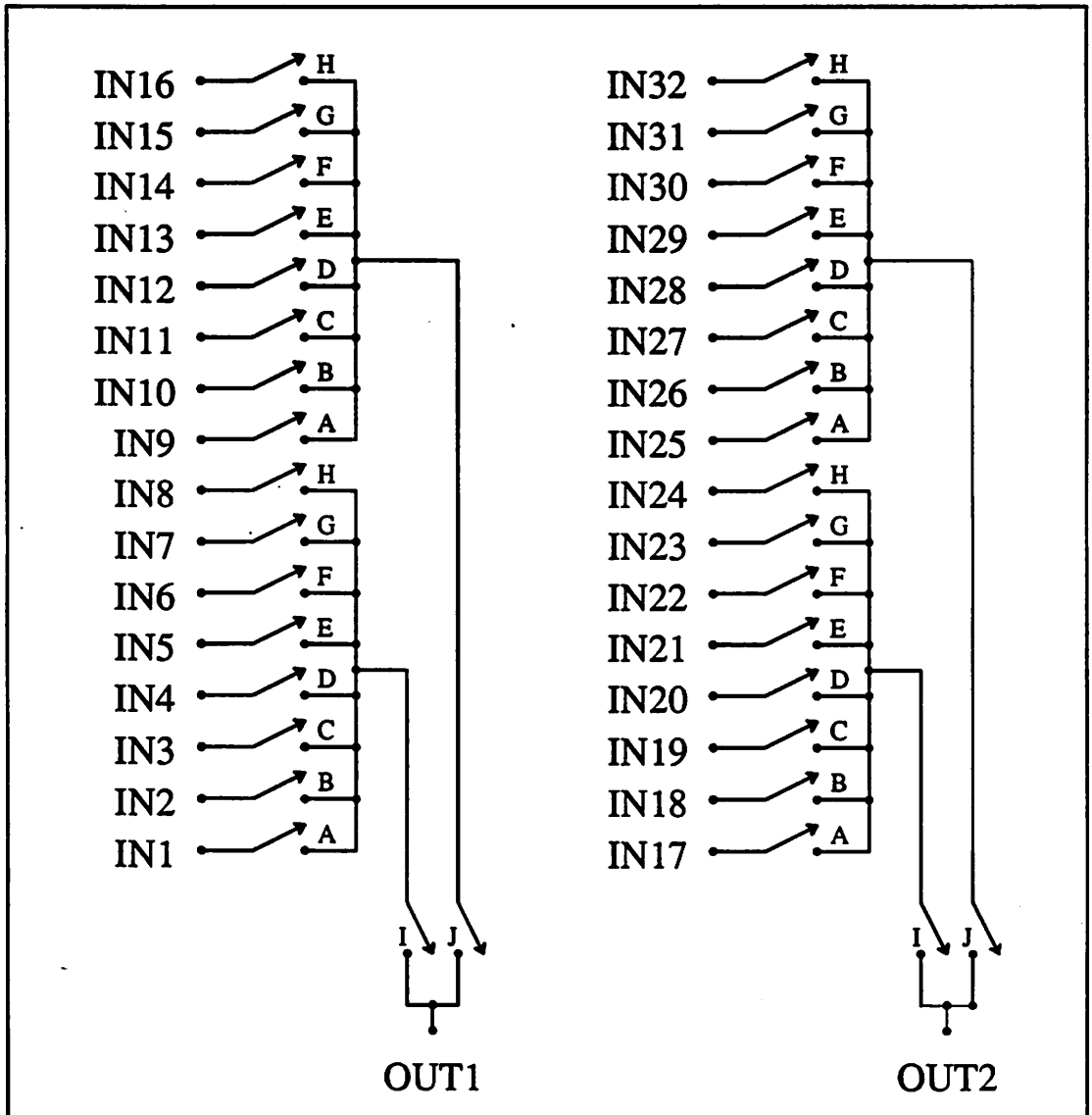


Figure 4.23 - Positive and negative multiplexers

The next improvement is to notice that the multiplexer that selects the inputs for V_{n+1} selects taps that are always one tap away from the multiplexer that selects the V_n taps. Thus each multiplexer can again have half of the inputs (half of the parasitic loading capacitance) and select alternate taps as shown in Figure 4.24. (Note: 2 multiplexers are in Figure 4.24. 4 are needed.)

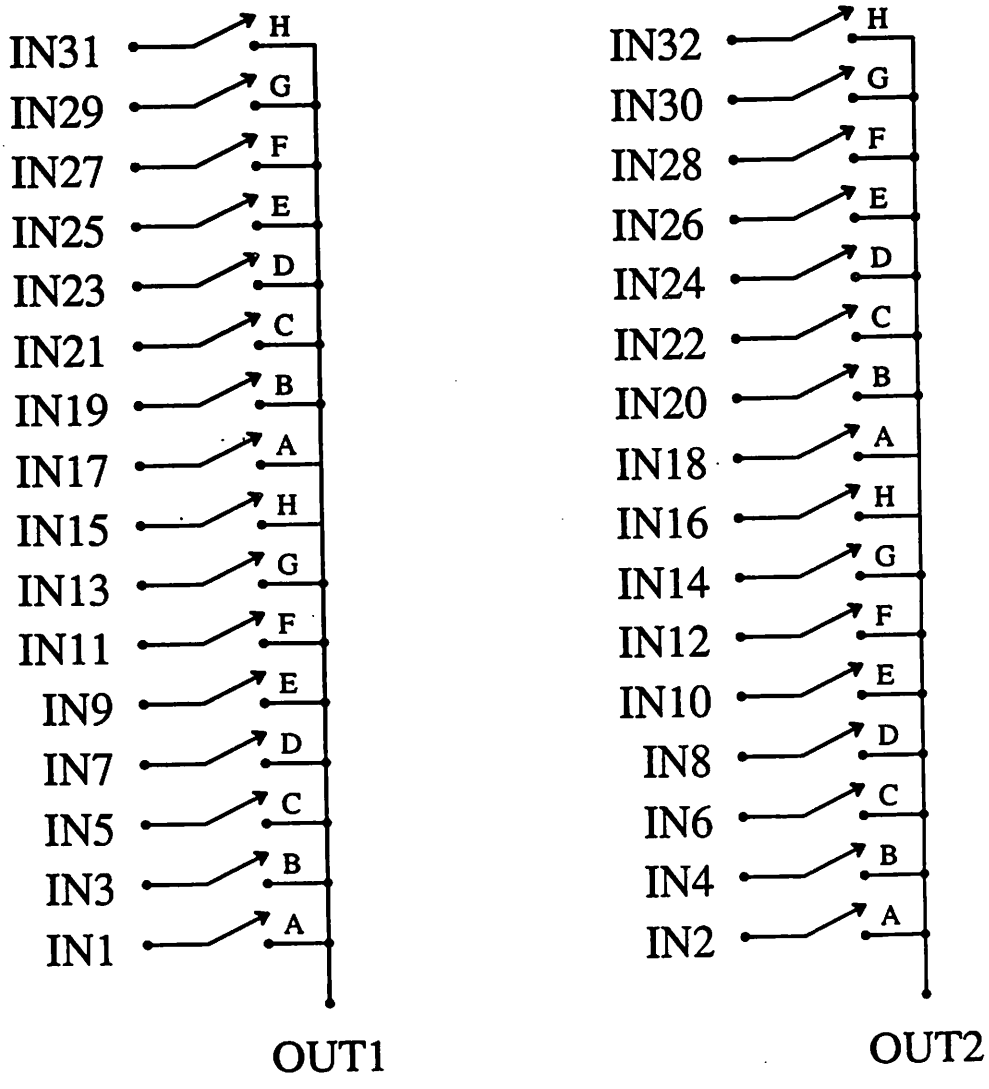


Figure 4.24 - Multiplexers selecting alternate inputs

The problem is that multiplexer 1 selects V_{n+1} and multiplexer 2 selects V_n. This is fine for even inputs, $|V_{n+1}| > |V_n|$. But for odd inputs $|V_{n+1}| < |V_n|$. They could be switched with the switches shown in Figure 4.25 but this adds more series resistance.

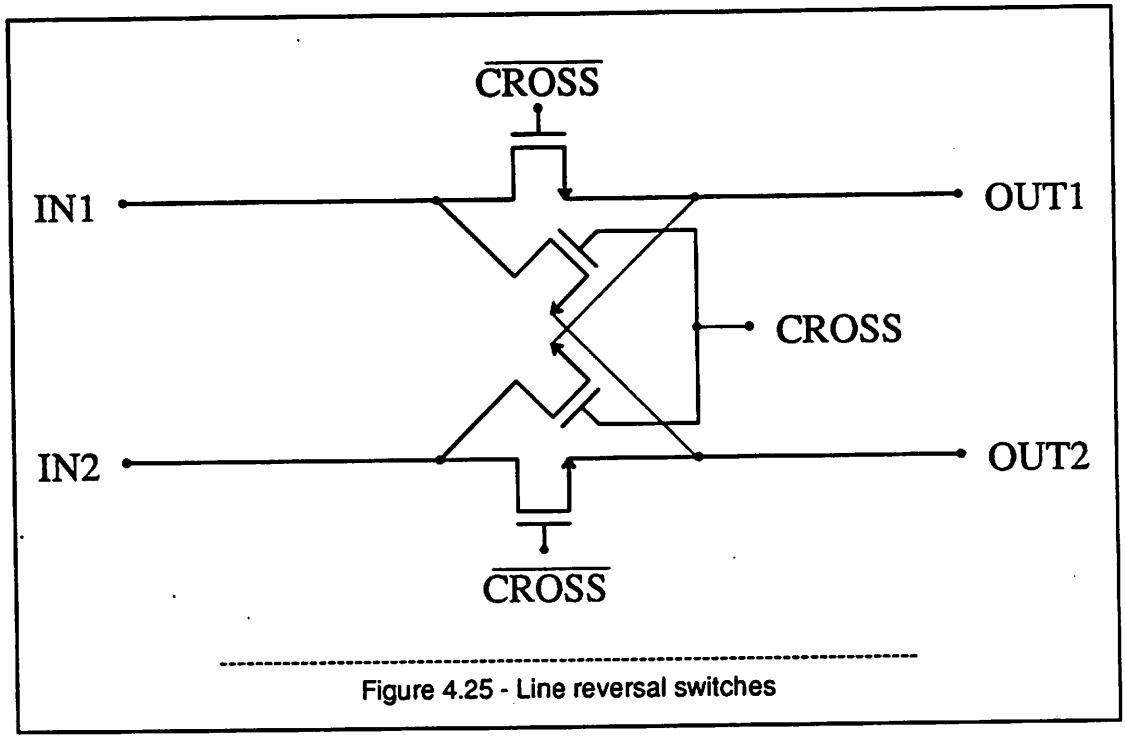


Figure 4.25 - Line reversal switches

Another solution is to examine the effect of switching V_{n+1} and V_n. The binary-weighted capacitor-array sub-ADC of Figure 4.17 is repeated here in Figure 4.26.

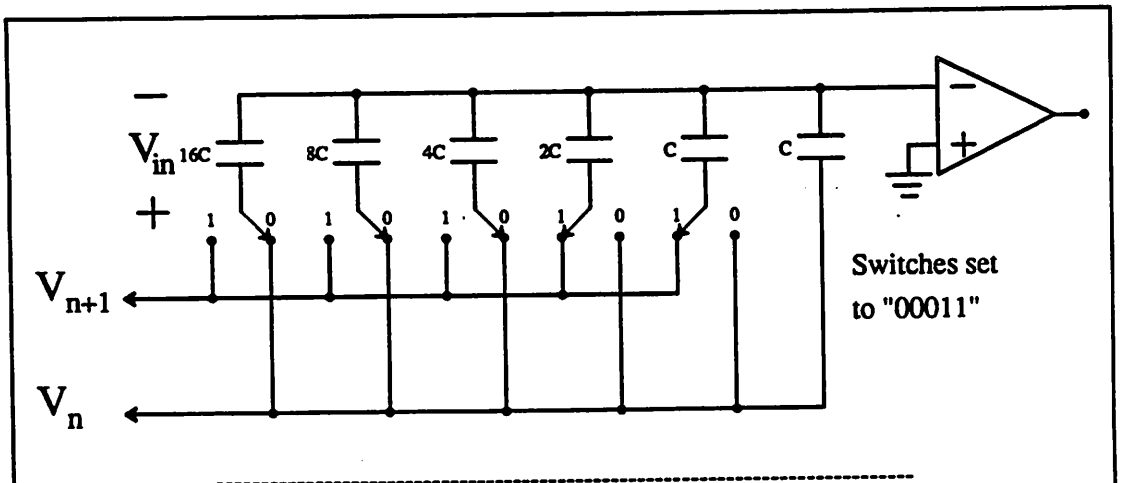


Figure 4.26 - Binary-weighted capacitor-array sub-ADC

The key point is that it is preset for a code, say 00011. That means capacitors 16C, 8C, 4C and one of the units, C, are connected to V_n and capacitors 2C and C are connected to V_{n+1}. If the roles of V_{n+1} and V_n are reversed as shown in Figure 4.27 then 16C, 8C, 4C and 1 unit C are connected to V_{n+1} and capacitors 2C and C are connected to V_n. This configuration is preset to code 11101.

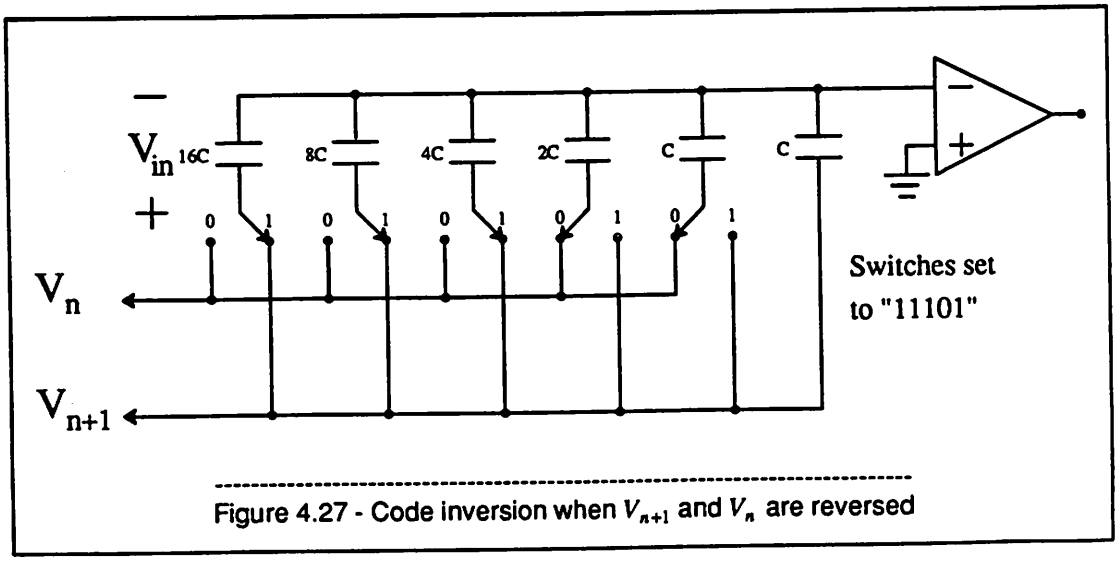


Figure 4.27 - Code inversion when V_{n+1} and V_n are reversed

Thus for "odd" inputs to the multiplexer V_{n+1} and V_n are reversed and the preset code of each sub-ADC is complemented and 1 is added. Thus the sub-ADCs are now preset as shown in Figure 4.28.

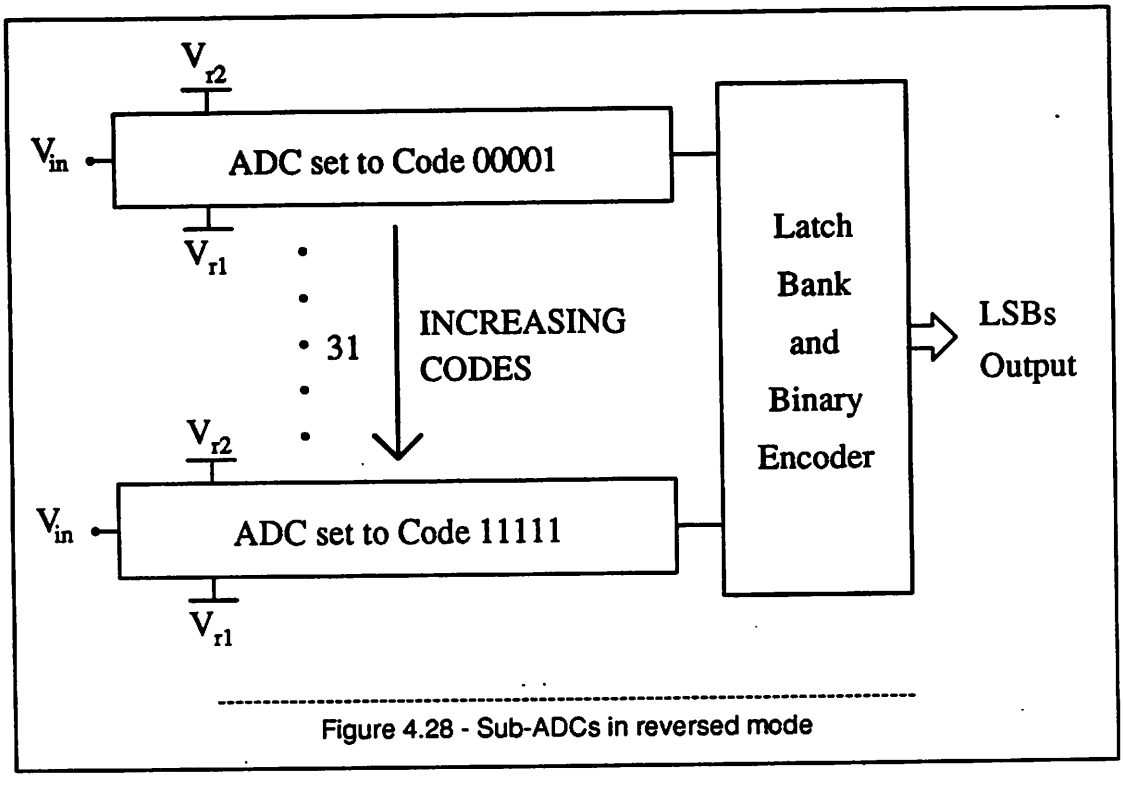


Figure 4.28 - Sub-ADCs in reversed mode

Now the thermometer code starts at the top and goes down. To decode this a second thermometer to 1-of-n decoder that decodes from top to bottom is added and its outputs are selected when needed.

Figure 4.29 shows both multiplexer schemes being used. There are four multiplexers each with one quarter of the number of inputs of the full multiplexers originally used. The second level decode takes the four outputs and correctly determines which is V_{n+1} and V_n and which are their complements.

4.3.4.1. First-level Multiplexer

Figure 4.30 shows one quarter of the full first level multiplexer. The resistor string has been broken into 4 sections of eight resistors. Each section has a 4-input and a 5-input multiplexer selecting alternate taps. Table 4.1 has the MSB code in the first column and the other 4 columns indicate which line is V_{n+1} and V_n for the "positive" array and for the complementary "negative" array. The table also indicates which switch is turned on in the multiplexer.

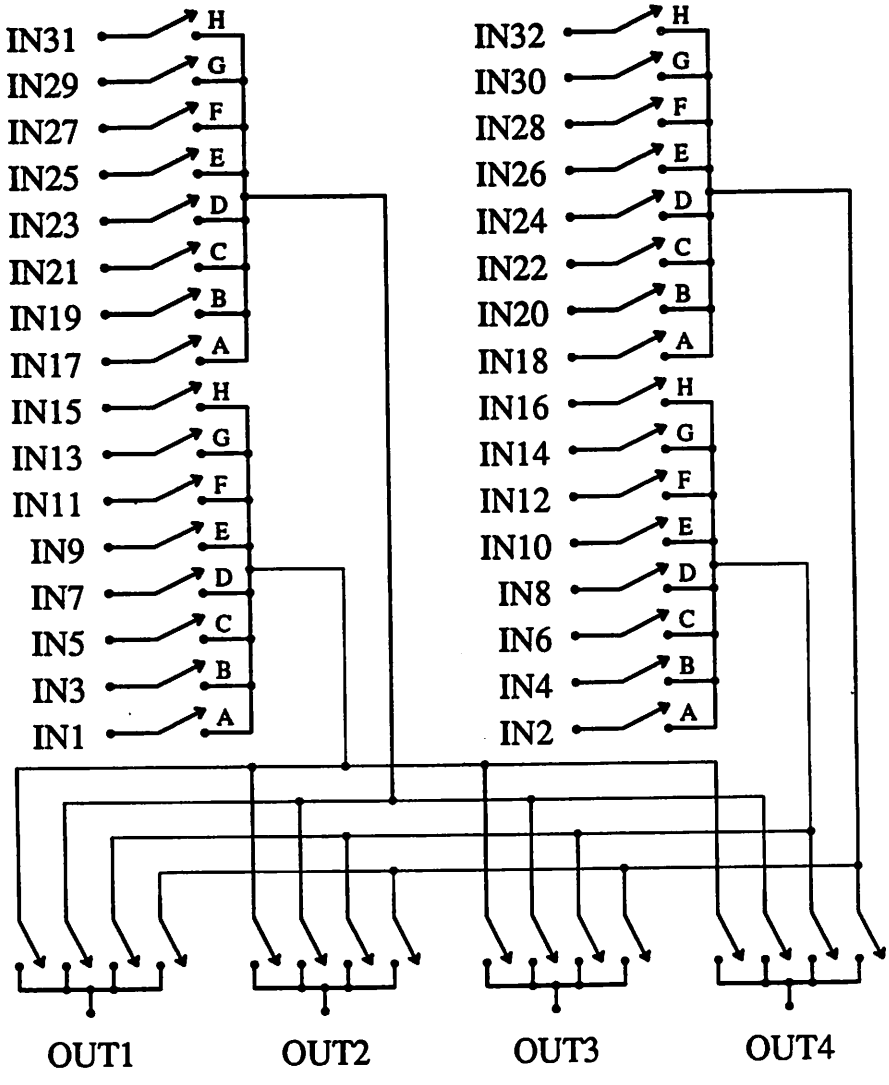


Figure 4.29 - Multiplexers selecting alternate taps and positive and negative inputs

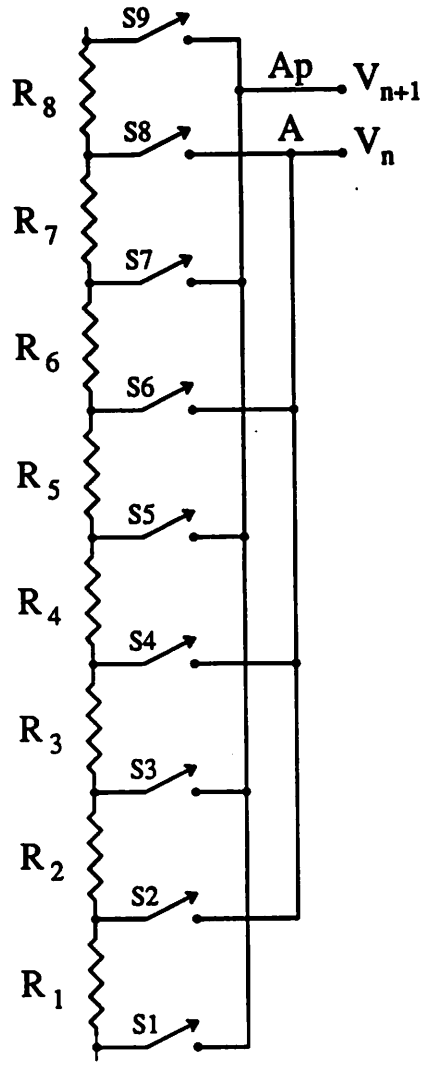


Figure 4.30 - First-level multiplexer

MSB CODE	P-array V_n	P-array V_{n+1}	N-array V_n	N-array V_{n+1}
0	s2-A	s1-Ap	s35-D	s36-Dp
2	s4-A	s3-Ap	s33-D	s34-Dp
4	s6-A	s5-Ap	s31-D	s32-Dp
6	s8-A	s7-Ap	s29-D	s30-Dp
8	s11-B	s10-Bp	s26-C	s27-Cp
10	s13-B	s12-Bp	s24-C	s25-Cp
12	s15-B	s14-Bp	s22-C	s23-Cp
14	s17-B	s16-Bp	s20-C	s21-Cp
16	s19-Cp	s20-C	s18-Bp	s17-B
18	s21-Cp	s22-C	s16-Bp	s15-B
20	s23-Cp	s24-C	s14-Bp	s13-B
22	s25-Cp	s26-C	s12-Bp	s11-B
24	s28-Dp	s29-D	s9-Ap	s8-A
26	s30-Dp	s31-D	s7-Ap	s6-A
28	s32-Dp	s33-D	s5-Ap	s4-A
30	s34-Dp	s35-D	s3-Ap	s2-A
1	s3-Ap	s2-A	s34-Dp	s35-D
3	s5-Ap	s4-A	s32-Dp	s33-D
5	s7-Ap	s6-A	s30-Dp	s31-D
7	s9-Ap	s8-A	s28-Dp	s29-D
9	s12-Bp	s11-B	s25-Cp	s26-C
11	s14-Bp	s13-B	s23-Cp	s24-C
13	s16-Bp	s15-B	s21-Cp	s22-C
15	s18-Bp	s17-B	s19-Cp	s20-C
17	s20-C	s21-Cp	s17-B	s16-Bp
19	s22-C	s23-Cp	s15-B	s14-Bp
21	s24-C	s25-Cp	s13-B	s12-Bp
23	s26-C	s27-Cp	s11-B	s10-Bp
25	s29-D	s30-Dp	s8-A	s7-Ap
27	s31-D	s32-Dp	s6-A	s5-Ap
29	s33-D	s34-Dp	s4-A	s3-Ap
31	s35-D	s36-Dp	s2-A	s1-Ap

Table 4.1. Multiplexer switches and output lines for MSB code.

There are 36 switches in the full multiplexer. Any time a "positive" side switch (P-array) is on so is its complementary "negative" switch (N-array) thus only 18 control lines are needed.

4.3.4.1.1. First-level Multiplexer Control Lines

The 18 control lines, C1 through C18, are generated from the MSBs. Table 4.2 shows the MSB code and control lines that need to be activated.

CONTROL LINE	SWITCHES	MSB CODES
C1	s1, s36	0, 31
C2	s2, s35	0, 1, 31, 30
C3	s3, s34	1, 2, 30, 29
C4	s4, s33	2, 3, 29, 28
C5	s5, s32	3, 4, 28, 27
C6	s6, s31	4, 5, 27, 26
C7	s7, s30	5, 6, 26, 25
C8	s8, s29	6, 7, 25, 24
C9	s9, s28	7, 24
C10	s10, s27	8, 23
C11	s11, s26	8, 9, 23, 22
C12	s12, s25	9, 10, 22, 21
C13	s13, s24	10, 11, 21, 20
C14	s14, s23	11, 12, 20, 19
C15	s15, s22	12, 13, 19, 18
C16	s16, s21	13, 14, 18, 17
C17	s17, s20	14, 15, 17, 16
C18	s18, s19	15, 16

Table 4.2 Control lines and switches for MSB codes

Looking at the table the Cx signal can be generated from 4-input OR gates whose inputs are the outputs of the 1-of-n decoder. However it is preferred to use 4-input NAND gates and inverted inputs.

4.3.4.2. Second-level Multiplexer

The second level multiplexer shown in Figure 4.31 consists of 4 switches labeled AN, BN, CN and DN. The multiplexer's output is the bottom plate of a capacitor in the array. Although there are 8 outputs from the 4 first-level multiplexers only 4 need to be decoded in the second step. If the capacitor gets code 1 it gets the 4 V_{n+1} lines to decode. If the code is 0 it decodes the V_n lines. Table 4.3 lists the switch (AN, BN, CN, DN) and the lines it could connect to for the two arrays. Note that for "odd" codes the meaning of V_n and V_{n+1} are reversed.

Switch	P-array V_n	P-array V_{n+1}	N-array V_n	N-array V_{n+1}
AN	A	Ap	D	Dp
BN	B	Bp	C	Cp
CP	Cp	C	Bp	B
DP	Dp	D	Ap	A

Table 4.3. V_{n+1} and V_n lines for switches AN, BN, CN and DN

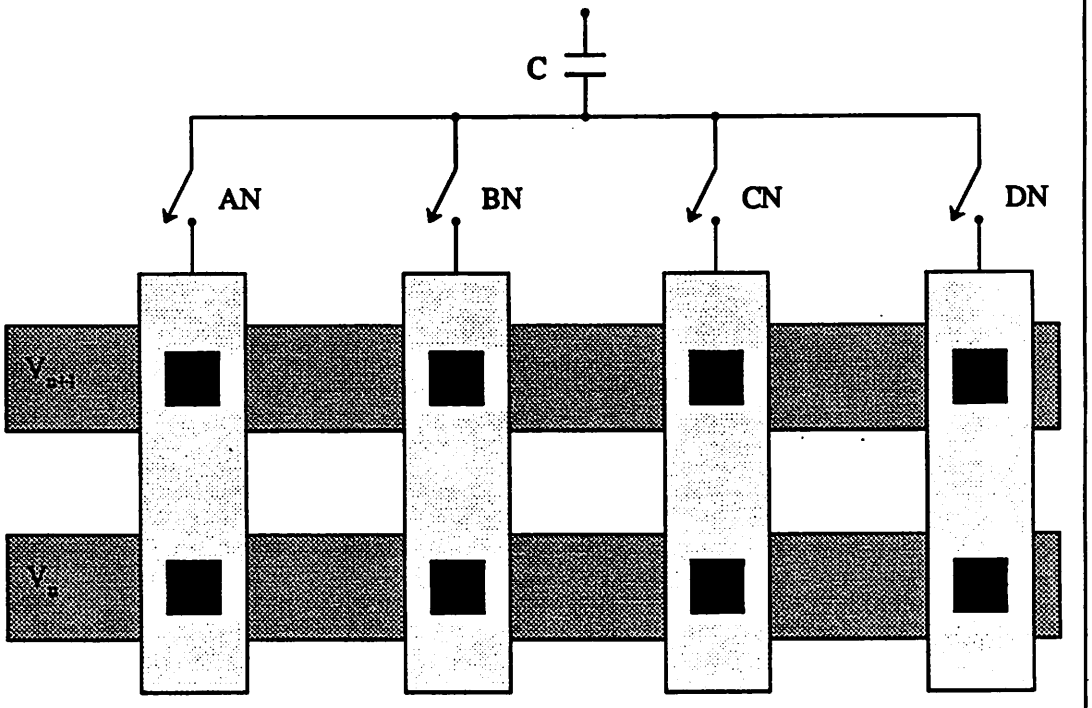


Figure 4.31 - Second-level mux and capacitor connection to line V_{n+1} or V_n

The line from each capacitor must be connected to line V_{n+1} or V_n depending if that capacitor is coded for a one or a zero. This could be very tedious work in doing the layout since there are 4 connections per capacitor, 5 capacitors in each array, 31 arrays in the "positive" set and the same 620 (4 x 5 x 31) connections for the "negative" set. Any misplaced connection would keep the ADC from operating properly and any change in the layout might necessitate the connections being made again. The chance of an error and the tedium dictated that a program be written to automate the layout of the connections. The connection to the V_{n+1} or V_n line was made as shown in Figure 4.31.

4.3.4.2.1. Automated Line Connection Program

Each of 31 sub-ADC cells differs by the code that it is set to. To automatically generate each cell a master cell was copied to the sub-cell and the program puts in the vias to set the code. Appendix A lists the program and explains its operation.

4.3.4.2.2. Control signals for Second-level Multiplexer

Figure 4.31 of the second level multiplexer showed four control lines, AN, BN, CN, and DN. One of which must be activated for the second-step LSBs conversion. Table 4.4 lists the MSB codes and the control line for each array that must be activated.

Codes	Control Line
0-7	AN
8-15	BN
16-23	CN
24-31	DN

Table 4.4 Bitcell control line for each MSB code

An 8-input OR gate with inputs connected to the 1-of-n codes as shown in Figure 4.32.

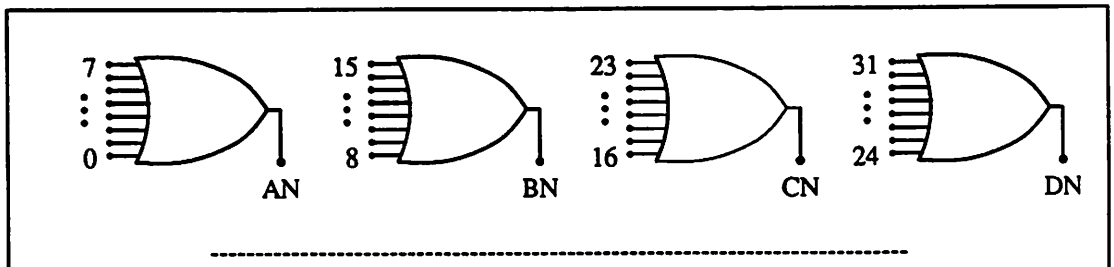


Figure 4.32 - AN, BN, CN, and DN control line generation with 8-input OR gates

8-input OR gates are just a brute-force solution and impractical to implement. A better way is to check the thermometer code and see what quarter of the resistor string the input voltage lies in. For example, line AN is active for codes less than or equal to 7. Line BN is active for codes less than or equal to 15 and greater than or equal to 8, etc. Figure 4.33 shows the simple logic needed to do this.

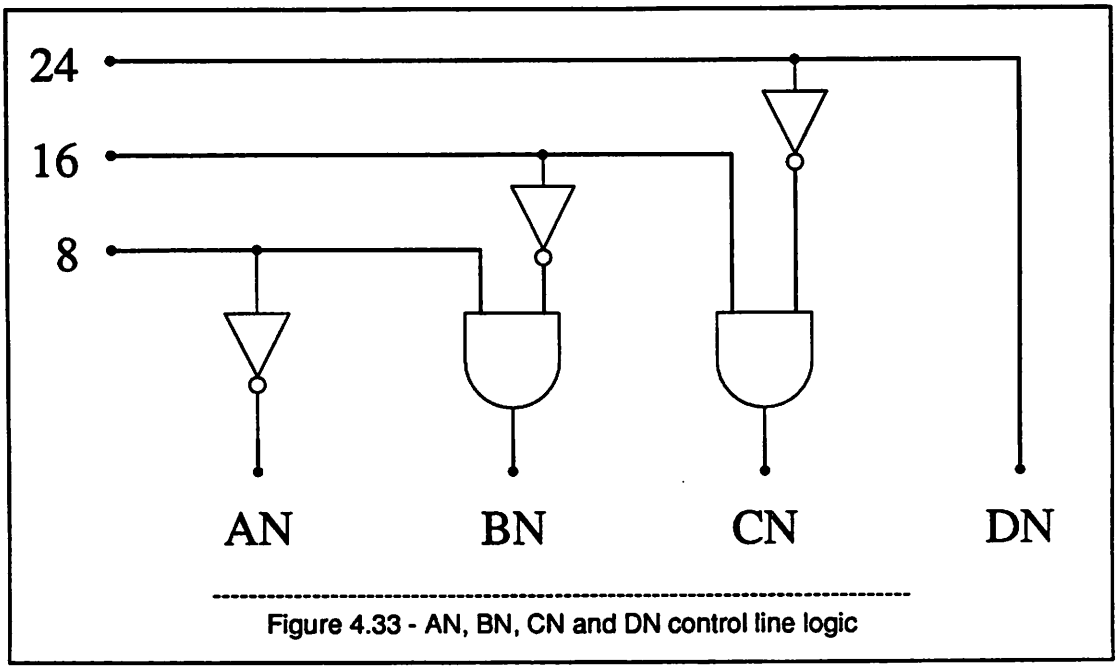


Figure 4.33 - AN, BN, CN and DN control line logic

4.3.4.2.3. Second-level Multiplexer, Input and Tap Switches

The capacitor in Figure 4.4 used to sample the input voltage is actually the array with all of the capacitors connected in parallel. The second-level analog multiplexer has two more switches as shown in Figure 4.34. The two new switches are to sample the input voltage and connect the capacitor to the resistor-string taps for the first-step MSB conversion.

The switches are desired to have an on-resistance of 300 Ω and have W/Ls of 51 μm / 1.6 μm for the N-channel switches and have a W/L of 164 μm / 1.6 μm for the P-channel input sampling switches. The first-decision tap switches have 240 Ω of on resistance and W/Ls of 64 μm / 1.6 μm and 205 μm / 1.6 μm for the N and P switches respectively.

5. High-Speed, High-Gain Comparator

The comparator is the heart of any ADC and often determines just how well the ADC will work. Although it has been shown symbolically as just an amplifier it also contains a latch and additional circuitry so that with the capacitor array it can perform the sample-and-hold function.

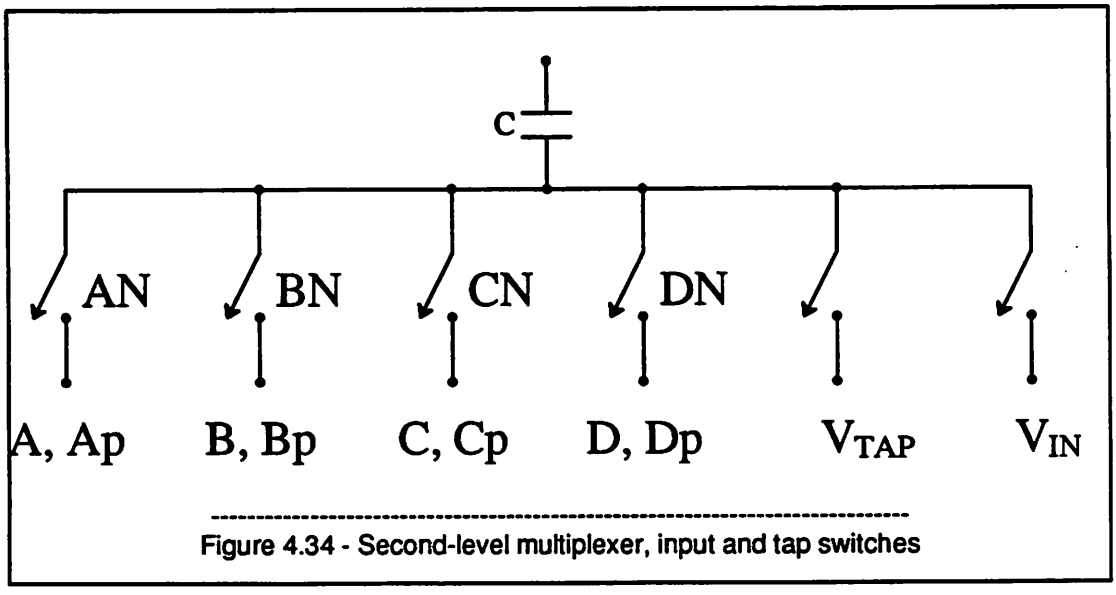


Figure 4.34 - Second-level multiplexer, input and tap switches

5.1. Specifications

The comparator design centered on meeting several specifications. Comparator gain is usually about 2^n as a rule-of-thumb. This is so that it can amplify a voltage that is less than one-half of an LSB up to a fraction of the supply voltage that can then be stored in a latch. Here a gain of 1000 is desired.

The comparator speed has two components, the open-loop response when amplifying and the closed-loop response when canceling its offset voltage. In the initial ADC design for a 10-Msample/s conversion rate 10 ns was allotted for offset cancellation and 10 ns for comparator decision making.

The comparator architecture had the constraint of being fully differential and its input noise must be less than one-half of an LSB so as not to make a false decision.

Many approaches have been taken to high-speed comparator design. Here the design centered on optimizing the number of gain stages needed for the fastest decision making, then the stage design, the sample-and-hold portion and finally the latch.

5.2. Amplifier versus Latch Consideration

The comparator will have a latch to store the result of the comparison. What needs to be determined is whether the latch can be used as the comparator or should an amplifier precede the latch. A determination can be made by comparing the time responses of both.

A positive feedback latch has the following transfer function:

$$\frac{V_{out}(t)}{V_{in}} = e^{t/\tau} \tag{4.32}$$

A single-pole amplifier with the same time constant, τ, and a DC gain, A, has the time response:

$$\frac{V_{out}(t)}{V_{in}} = A\left[1 - e^{-t/\tau}\right] \tag{4.33}$$

The gain is time dependent. It is defined to be $\frac{V_{out}(t)}{V_{in}}$ and plotted in Figure 4.35 for $A = 10$. For time less than 2τ and greater than 0.1τ the amplifier output is greater than the latch output. Approximately 2τ is allotted for comparator decisions thus the amplifier will give more gain than the latch and will be used as the "comparator". A latch is still required to store the result.

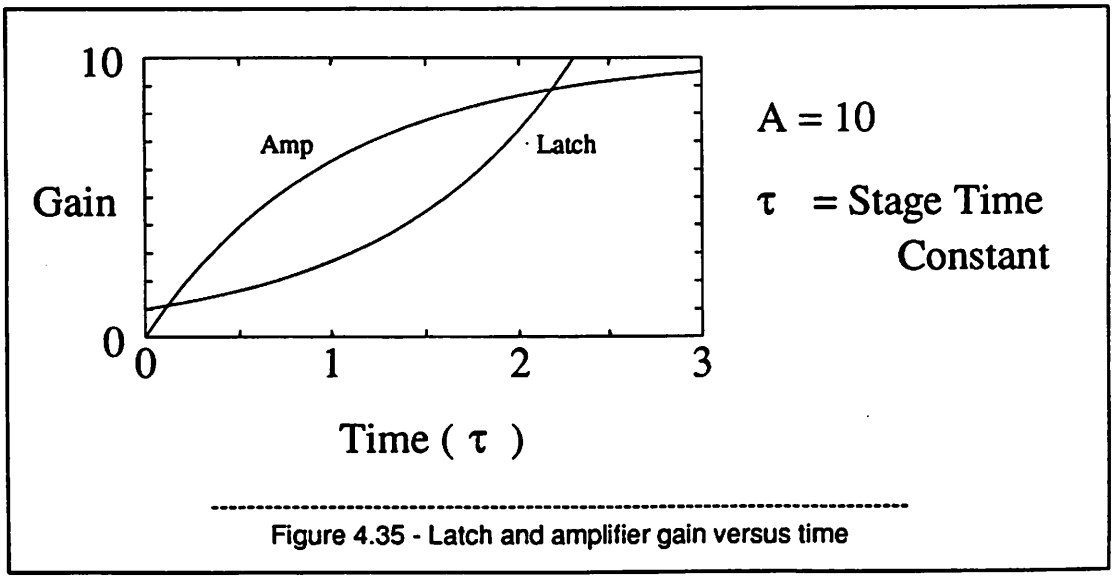


Figure 4.35 - Latch and amplifier gain versus time

5.3. Optimum Number of Comparator Gain Stages

Given that the comparator is an amplifier, the optimum number of stages for the fastest response must be determined. Conceptually this is shown in Figure 4.36. Here there are n gain stages each with gain A. The total gain is $G = A^n$. Note that n is just a variable and not a number of bits in this case. Each stage is assumed to have a gain-bandwidth product, GBW, that is a constant, but gain can be traded for bandwidth. Three models and methods will be used to determine the number of stages needed. The first is a simple gain-bandwidth product optimization, the second is a frequency-domain, phase-delay computation and the third computes the time-domain delay for an n-stage cascade of single-pole amplifiers.

5.3.1. Optimum Number of Comparator Gain Stages: Gain-Bandwidth Product Tradeoff

This method of finding the optimum number of comparator stages for the fastest decision computes the time for a decision and minimizes it with respect to the number of stages, n. With the previous assumptions:

$$GBW = \frac{G^{1/n}}{\tau} \tag{4.34}$$

Here τ = 1-stage delay (1 / stage bandwidth). Then t = n-stage delay = $n\tau$ is the comparison time.

$$t = \frac{nG^{1/n}}{GBW} \tag{4.35}$$

To minimize the comparison time with respect to the number of stages $\frac{dt}{dn}$ is computed, set equal to

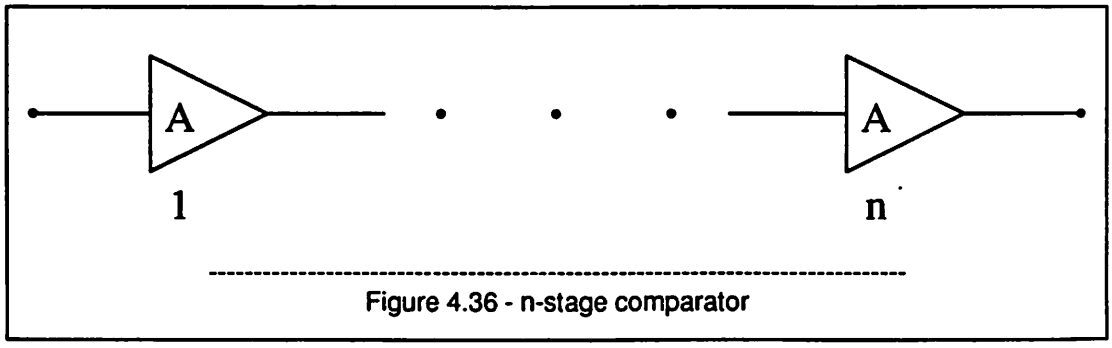


Figure 4.36 - n-stage comparator

zero and n is found.

$$\frac{dt}{dn} = 0 \;\Rightarrow\; n = ln(G) \tag{4.36}$$

Exponentiating the result and using $G = A^n$:

$$e^n = G = A^n \;\;\therefore A = e \tag{4.37}$$

The two important results are that the optimum number of stages for the fastest response is $n = ln(G)$ and the gain per stage $A = e$.

5.3.2. Optimum Number of Comparator Stages: Frequency-Domain Model

The n-stage comparator delay can also be found in the frequency domain[17] and minimized with respect to the number of stages. Assume that phase, $\phi(\omega)$, is a measure of the stage delay. For a single-pole stage with -3 dB bandwidth, ω_0:

$$\phi(\omega) = \tan^{-1}\left[\frac{\omega}{\omega_0}\right] \tag{4.38}$$

For n stages in a cascade:

$$\phi(\omega) = n\tan^{-1}\left[\frac{\omega}{\omega_0}\right] \tag{4.39}$$

The "delay", D, of the n-stage comparator is $\frac{d\phi}{d\omega}$.

$$D = \frac{d\phi}{d\omega} = \frac{n}{1+\left[\frac{\omega}{\omega_0}\right]^2}\cdot\frac{1}{\omega_0} = \frac{n\,\omega_0}{\omega^2+{\omega_0}^2} \tag{4.40}$$

To minimize the delay, D, find $\frac{dD}{dn}$ and set it equal to 0.

$$\frac{dD}{dn} = n\left[\frac{(\omega^2+{\omega_0}^2)\frac{\partial\omega_0}{\partial n} - 2{\omega_0}^2\frac{\partial\omega_0}{\partial n}}{(\omega^2+{\omega_0}^2)^2}\right] + \frac{\omega_0}{(\omega^2+{\omega_0}^2)} = 0 \tag{4.41}$$

$$\frac{n\frac{\partial\omega_0}{\partial n}\left[(\omega^2+{\omega_0}^2) - 2{\omega_0}^2\right]}{(\omega^2+{\omega_0}^2)} + \omega_0 = 0 \tag{4.42}$$

$$n\frac{\partial\omega_0}{\partial n}(\omega^2-\omega_0^2)+\omega_0(\omega^2+\omega_0^2)=0 \tag{4.43}$$

$$\frac{\partial\omega_0}{\partial n}=-\frac{\omega_0(\omega^2+\omega_0^2)}{n(\omega^2-\omega_0^2)} \tag{4.44}$$

From the definition of GBW, $GBW = G^{1/n}\omega_0$, and $\omega_0 = GBW\ G^{-1/n}$ and $\frac{\partial\omega_0}{\partial n}$ can also be calculated.

$$\frac{\partial\omega_0}{\partial n}=GBW\left[\frac{G^{-1/n}ln(G)}{n^2}\right] \tag{4.45}$$

Setting Equation 4.44 equal to Equation 4.45.

$$\frac{GBW\ G^{-1/n}ln(G)}{n^2}=-\frac{\omega_0(\omega^2+\omega_0^2)}{n(\omega^2-\omega_0^2)}=-\frac{GBW\ G^{-1/n}(\omega^2+\omega_0^2)}{n(\omega^2-\omega_0^2)} \tag{4.46}$$

$$\frac{ln(G)}{n}=\frac{-(\omega^2+\omega_0^2)}{(\omega^2-\omega_0^2)} \tag{4.47}$$

$$n=ln(G)\frac{(\omega_0^2-\omega^2)}{(\omega_0^2+\omega^2)} \tag{4.48}$$

For Equation 4.48 to have meaning $\omega_0^2 > \omega^2$ since $G > 1$. The equation also gives an upper bound on n since as ω approaches ω_0, n approaches 0. To find the upper limit set $\omega = 0$. Then $n = ln(G)$. This is the same result of Section 5.3.1, Equation 4.36 using the gain-bandwidth tradeoff model.

5.3.3. Optimum Number of Comparator Stages: Time-Domain Solution

The previous two methods for determining the optimum number of comparator stages for the fastest decision have not dealt with the comparator settling to a final value. They have just dealt with a delay that was not defined by any limits. What is wanted is for the optimization and be able to define the delay as the time until the comparator reaches some fraction of its final value.

To do this assume each stage can be modeled with a single-pole response.

$$H(S)=\frac{A_0\omega_o}{s+\omega_o} \tag{4.49}$$

Then the n-stage transfer function is the following:

$$H(S)=\frac{(A_0\omega_o)^n}{(s+\omega_o)^n} \tag{4.50}$$

For a step input, the output is:

$$V_{out}(S) = H(S)\,V_{in}(S) = H(S)\,\frac{1}{s} = \frac{1}{s}\,\frac{(A_0\omega_o)^n}{(s+\omega_o)^n} \tag{4.51}$$

The time response is found by taking the inverse Laplace transform of Equation 4.51.[18]

$$V_{out}(t) = V_{in}A_o^n\left[1 - e^{-\omega_o t}\sum_{k=0}^{n-1}\frac{(\omega_o t)^k}{k!}\right] \tag{4.52}$$

Then substitute T_{acq} for t and V_{req} for V_{out}. Then T_{acq} is solved for.

$$T_{acq} = -\frac{1}{\omega_0}\ln\left[1 - \frac{V_{req}}{A_0{}^n V_{in}}\right] + \frac{1}{\omega_0}\ln\left[\sum_{k=0}^{n-1}\frac{(\omega_o T_{acq})^k}{k!}\right] \tag{4.53}$$

Even though T_{acq} is defined implicitly this equation gives the time, T_{acq}, for the amplifier cascade to reach a given output voltage, V_{req}, with n stages. What is desired is to differentiate it with respect to the number of stages, n, to minimize T_{acq}. However n appears in the upper limit of the summation and that precludes differentiation. Approximations can be made[18] but more insight can be gained by going back to Equation 4.52 for the output voltage. Figure 4.37a shows the output voltage as a percentage of its final value versus time for n-stage amplifiers. A latch will follow the comparator stages and give

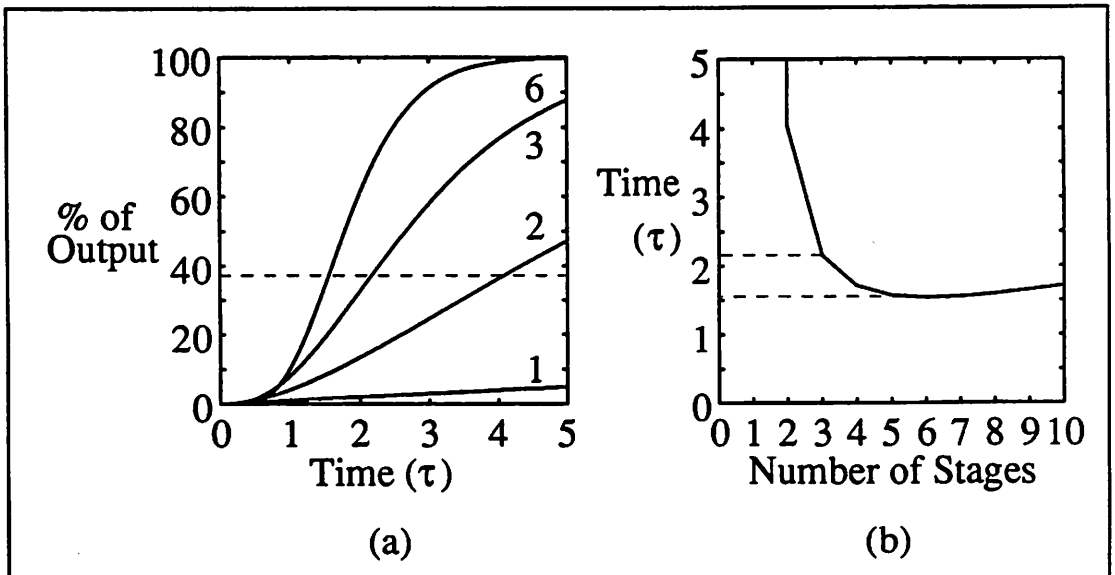


Figure 4.37 - n-stage comparator time response

gain. If one time constant is allotted to the latch it will have a gain of e. Thus consider the comparator amplification complete when it gets to $\frac{1}{e}$ of its final value as shown by the dotted line in the graph. The time to reach the dotted line versus the number of stages is plotted in Figure 4.37b. The fastest response is for 6 stages. This is almost the same as the result of Section 5.3.1 and Section 5.3.2 evaluated here:

$$n = ln(G) = ln(1000) = 6.9 \tag{4.54}$$

However the curve is flat for 3 to 10 stages. It is concluded that 3 stages is best since it is only about 0.6τ slower and saves 50% of the comparator area and power compared to 6 stages.

5.4. 3-Stage Comparator

In the past 1 or 2-stage comparators were used since they could be compensated for closed-loop stability. However compensating a 3-stage comparator is much harder and the possibility of oscillation exists. This possibility can be eliminated with the configuration of Figure 4.38. Here the 3 stages are capacitively coupled and the feedback switches close the loop around each stage independently. Thus the offset of each stage is stored at its input. A second advantage is that by implementing the

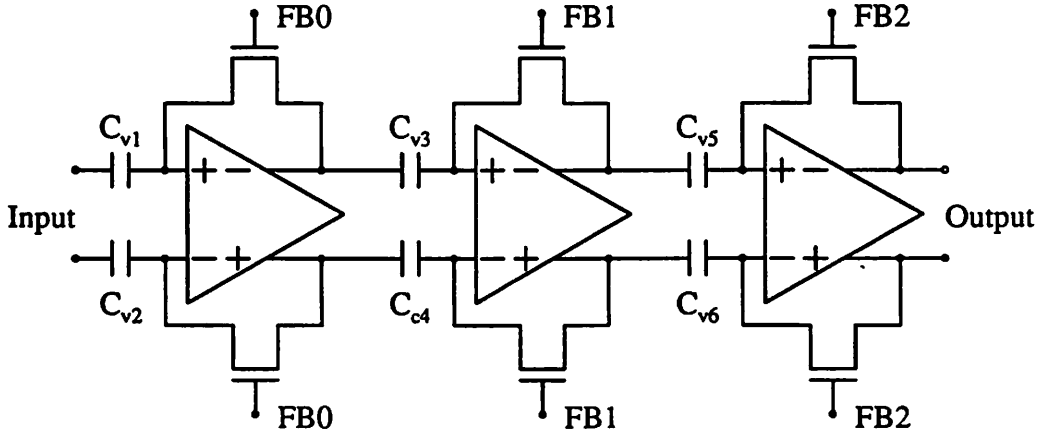


Figure 4.38 - 3-stage comparator

timing developed by Allstot[19] the charge-injection error from the first stage can be canceled if the ADC has an external S/H. When FB0 opens its charge-injection error is a voltage on capacitors C_{v1} and C_{v2} that is amplified by the first stage and stored on capacitors C_{v3} and C_{v4} before the other feedback switches are opened. The experimental results of canceling the charge injection error are in Chapter 8, Section 2.1.1.1.

5.5. Comparator Gain Stage

The general comparator requirements are that each stage needs a gain of 10, a 35 MHz bandwidth, must be able to cancel its offset to under 1 mV in less than 10 ns and must not use CMFB. Many gain-stage architectures were studied[20] and the design shown in Figure 4.39 was chosen. The stage is made up of a differential-pair input, current source and cascodes to reduce the Miller capacitance at the input. The loads are diode-connected transistors. They are just resistors so the common-mode output voltage is just set by the bias current and the load resistance.

5.5.1. Open-Loop Design

The gain is the input-stage transconductance times the output resistance.

$$G = gm_1 r_{out} = \frac{gm_1}{gm_3} = \sqrt{\frac{\mu_n W_1/L_1}{\mu_p W_3/L_3}} \tag{4.55}$$

This gives a gain of $\sqrt{10}$. To increase the gain the input transistor's transconductance is increased by a factor of $\sqrt{10}$ by injecting a large current into them from the P-channel current sources to give an overall gain of 10 for the stage.

The response is basically single-pole and the pole is located at the output node. The amplifier has a 35 MHz bandwidth and a 4.5 ns time constant.

5.5.2. Closed-Loop Design: S/H

In closed loop operation the amplifier must cancel its offset and with the capacitor array act as a sample-and-hold amplifier as shown in Figure 4.40.

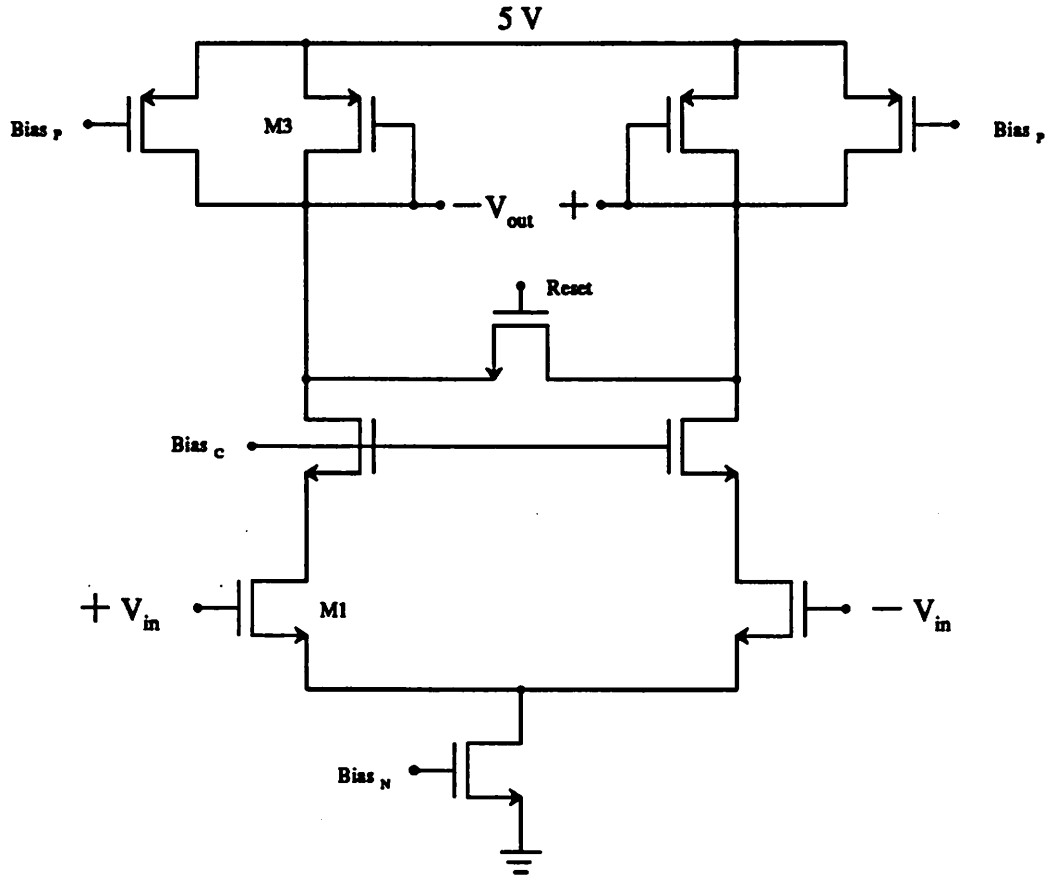


Figure 4.39 - Comparator gain-stage schematic

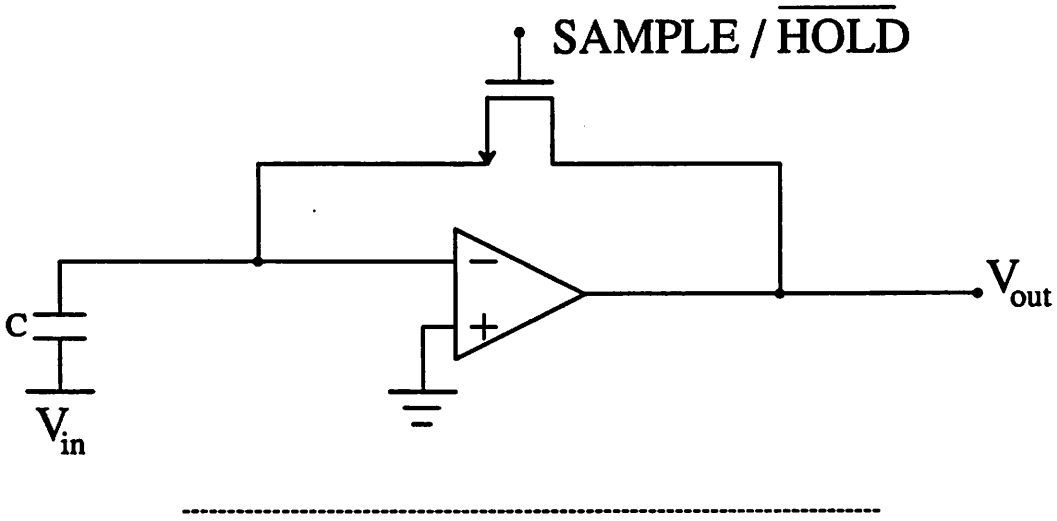


Figure 4.40 - Sample-and-hold amplifier configuration

For a full-scale 2.5 V sinewave input, and a maximum input frequency of 4.2 MHz, $V_{in} = V_{ref} \sin 2\pi f t$:

$$\left.\frac{dV}{dt}\right|_{max} = 6.6 \text{ X } 10^7 \ (V/s) \tag{4.56}$$

The capacitor array is 1.6 pF and the current required from the amplifier to follow the input voltage is $I = C\frac{dV}{dt} = 1.1\ mA$. This is a very large current for the amplifier to provide. A solution is to add an offset storing capacitor and a top-plate grounding switch as shown in Figure 4.41. In this configuration the 1.6 pF capacitor array, C_a, is charged through the large switch, S1, while the comparator stores its offset voltage on the much smaller 200 fF capacitor C_v. Since the top-plate grounding switch has its source grounded the sampling is not input-voltage dependent.

5.5.2.1. Feedback Switch

The feedback switch is a 12.6 µm / 1.6 µm PMOS device. It is PMOS rather than NMOS since the quiescent output voltage of the amplifier is about +1.35 V and an NMOS switch would have a small $V_{gs} - V_t$ resulting in a large resistance.

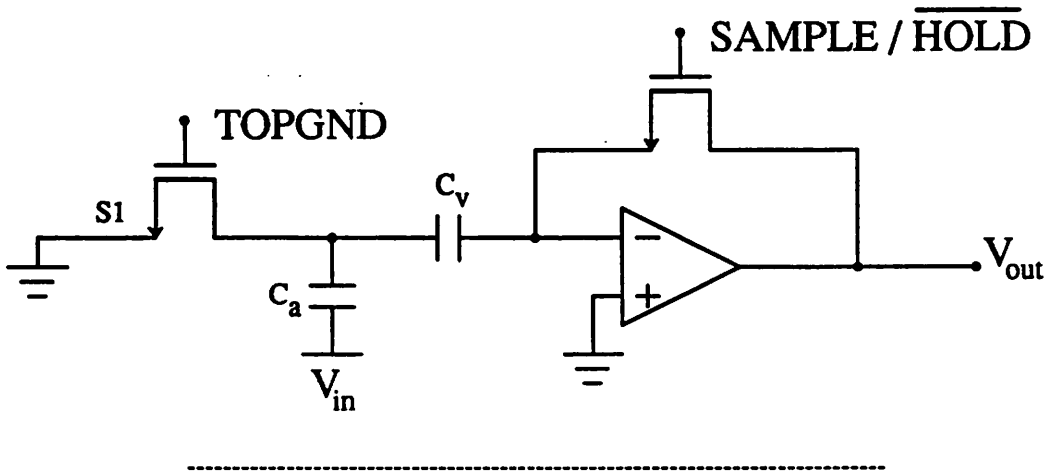


Figure 4.41 - S/H with top-plate grounding switch

5.5.3. Reset Switch

After the MSB decision is made most of the comparators will be in a saturated state and the recovery time before making the LSBs decision will be quite slow. The addition of a reset switch tied to the two comparator outputs is shown in Figure 4.39. After the first decision the switch is turned on to quickly bring the outputs back to a 0 V differential. This switch is an 4.4 µm / 1.6 µm NMOS device. Although the quiescent output voltage is +1.35 V, in a saturated state one of the outputs will go far negative and the device will have a large $V_{gs} - V_t$ and low resistance. It's not until the outputs get close to 0 V that $V_{gs} - V_t$ decreases and the switch resistance increases.

5.5.4. Noise Sources

There are two important noise sources, $\frac{kT}{C}$ noise from the S/H switch and the comparator input noise. The $\frac{kT}{C}$ noise from the top-plate grounding switch is 48 μV. The comparator input noise is from the input devices and cascodes is dominated by 1/f noise and is 0.65 mV in the comparator bandwidth.

5.5.5. Bias Supplies

Three bias voltages are needed for the comparator. They are generated with the circuit of Figure 4.42. Each voltage is the drop across a diode-connected transistor. The current supplying device in series with it could be replaced by another diode connected transistor but this configuration is more sensitive to V_t changes. This implementation provides a large V_{gs} that tends to swamp out any small changes in V_t. Bypass capacitors were included to stabilize the bias voltage line when the comparator had a transient. The bias voltages were also brought out to pins so that they could be externally adjusted it needed.

5.5.6. Layout

The comparator layout one of the most critical parts of the chip layout. Care was taken to make it symmetrical. The layout is shown in Figure 4.43. The input transistors have been split into two parts and are in a common-centroid configuration to reduce their offset voltage. Their interconnection has been matched to make their gate connection parasitic capacitances equal. All of the gates, except

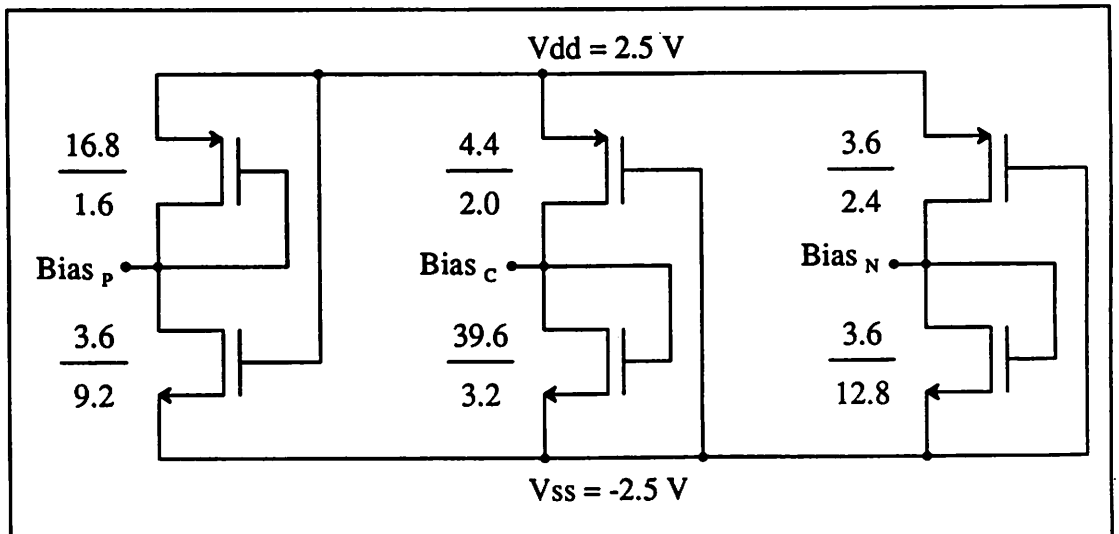


Figure 4.42 - Comparator bias generator circuit

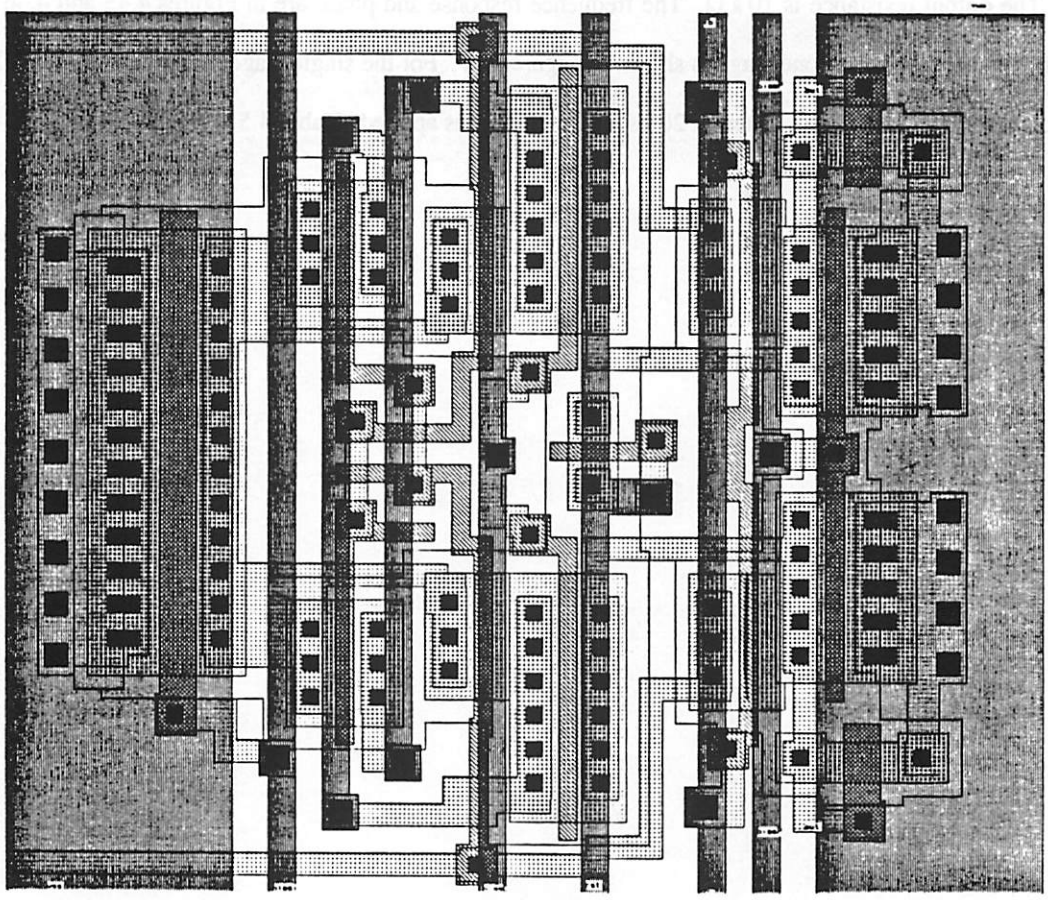


Figure 4.43 - Comparator layout

the reset switch gate, are perpendicular to the axis of symmetry so that a shift in the polysilicon or diffusion masks could not cause a mismatch in the two halves. 20 μm wide metal lines were used for Vdd and Vss to minimize their voltage drop.

5.5.7. Simulation Results

Figure 4.44 shows the DC transfer function of one comparator stage. The small signal gain is -22. The output resistance is $10\,k\Omega$. The frequence response and phase are in Figures 4.45 and 4.46. The transient response of one stage is shown in Figure 4.47. For the single stage in the first 20 ns a 10 mV offset is being canceled. Then at 20 ns a 10 mV input is applied. Table 4.5 summarizes the simulated comparator results.

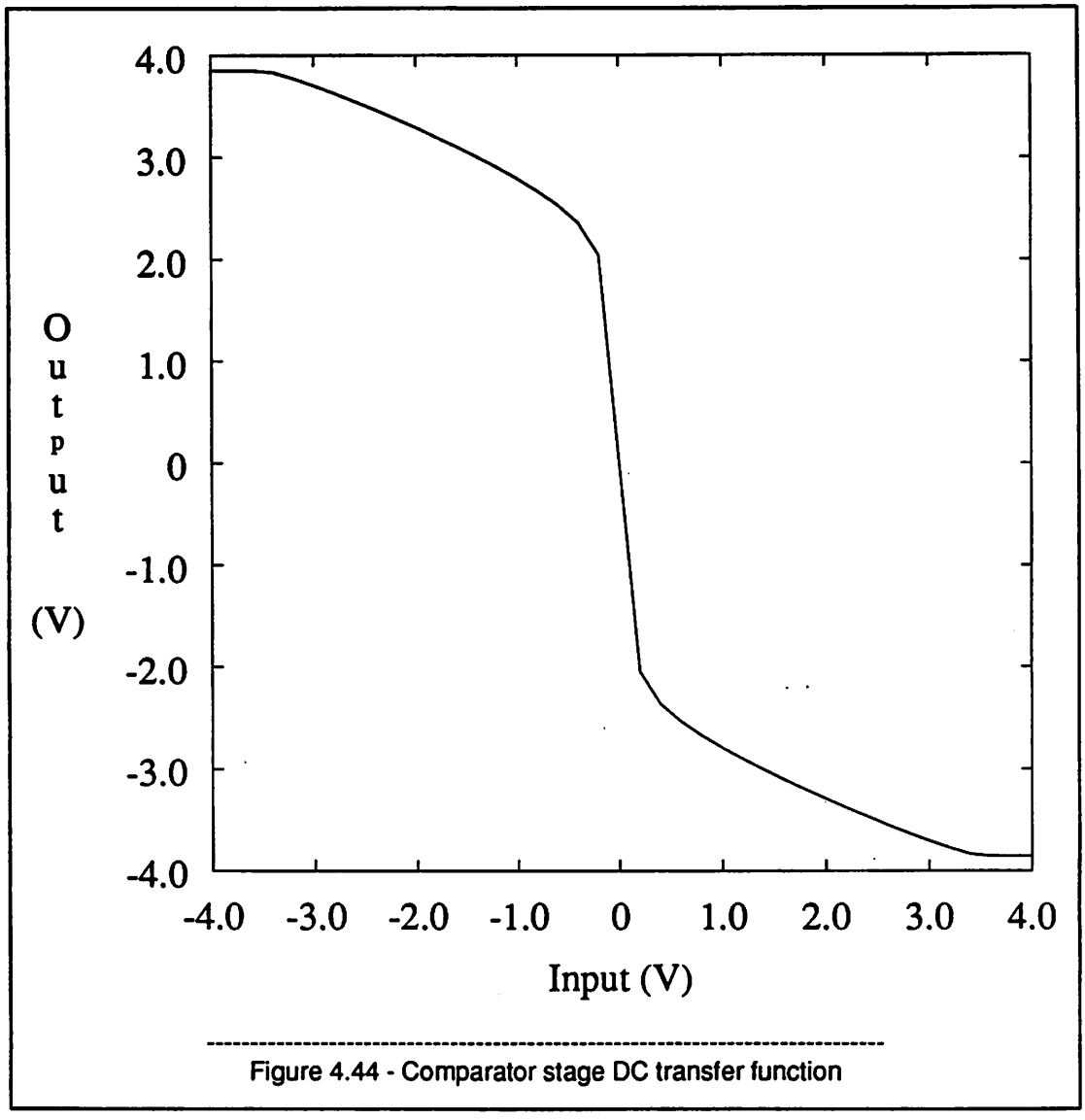


Figure 4.44 - Comparator stage DC transfer function

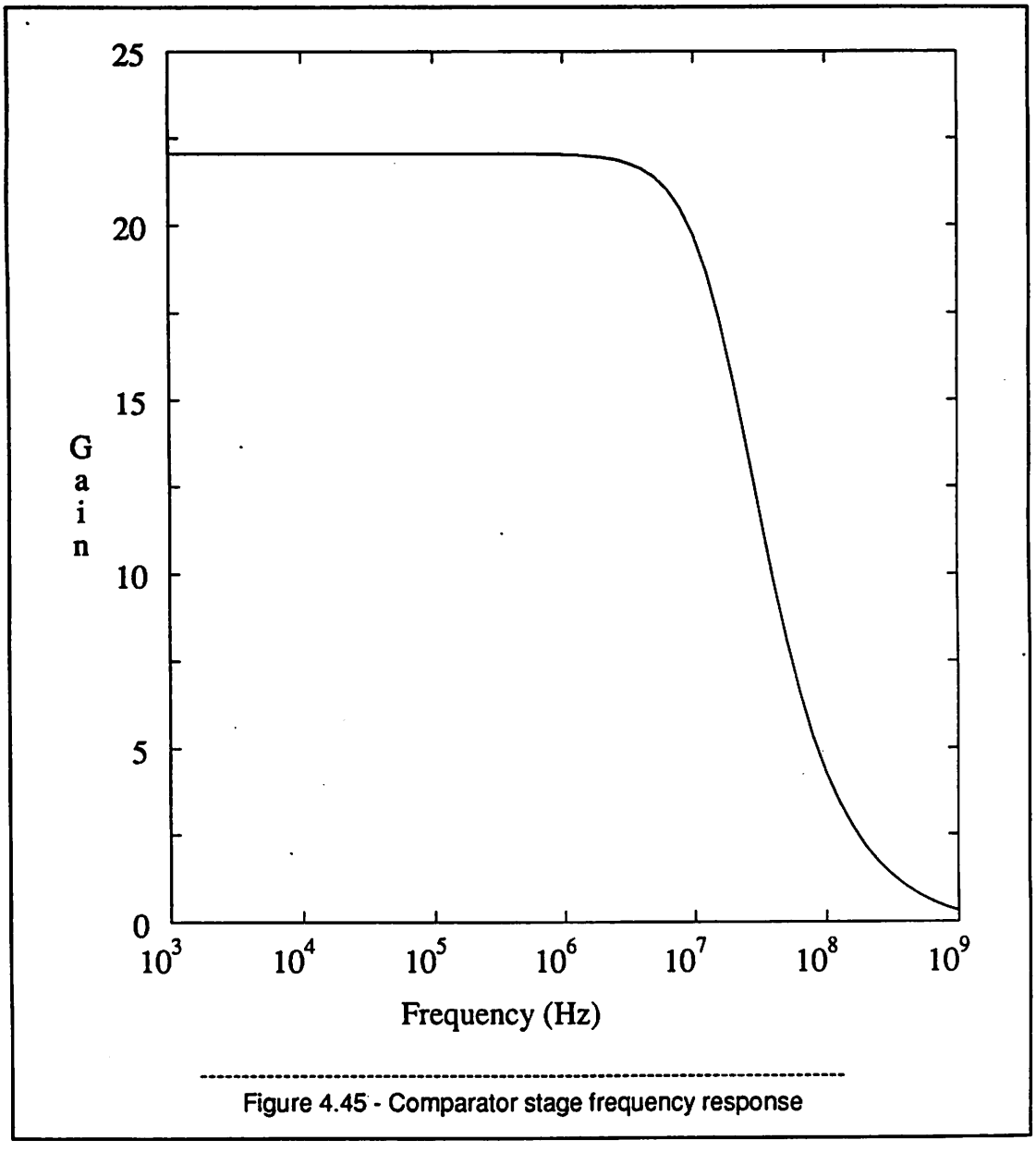


Figure 4.45 - Comparator stage frequency response

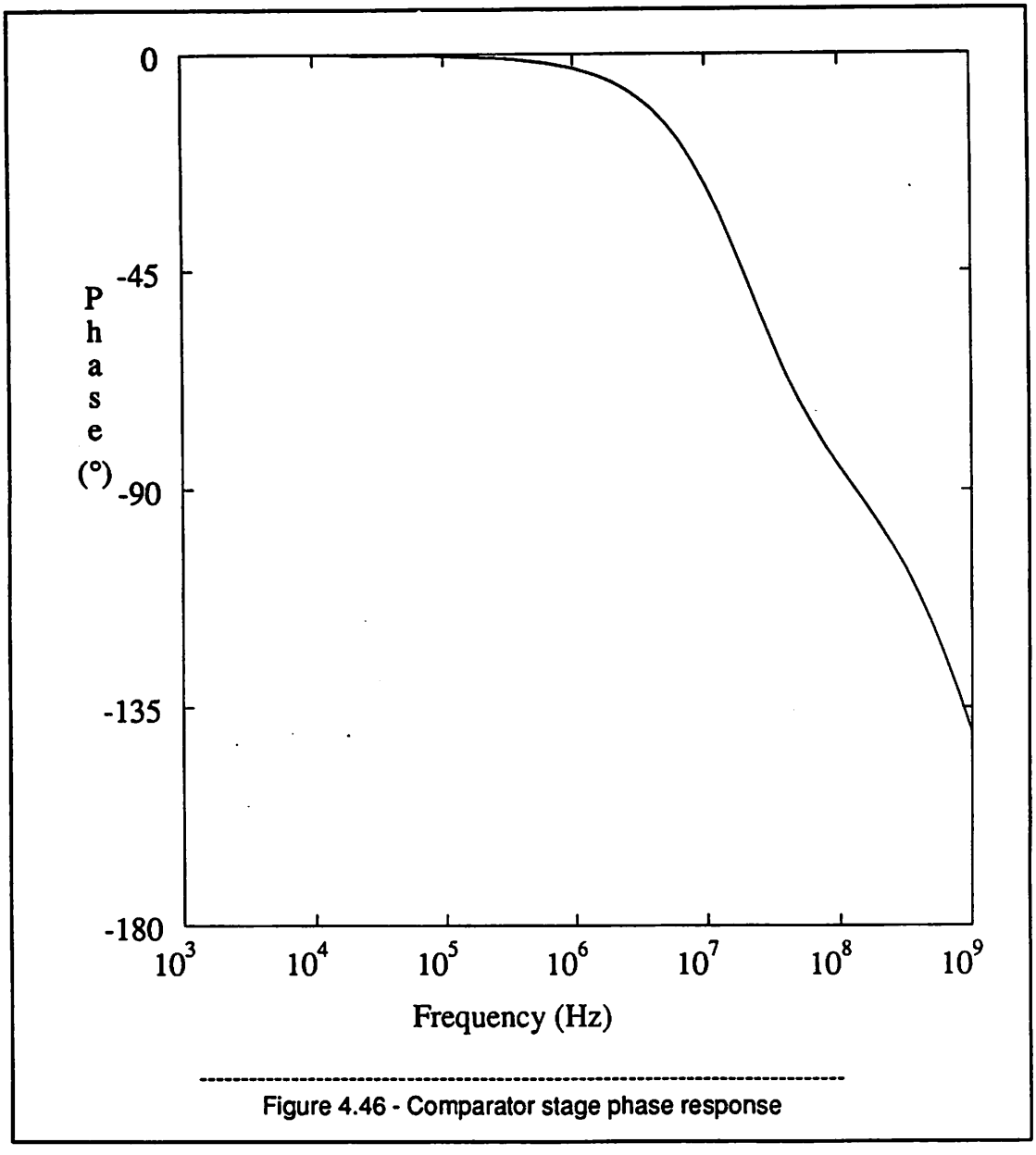


Figure 4.46 - Comparator stage phase response

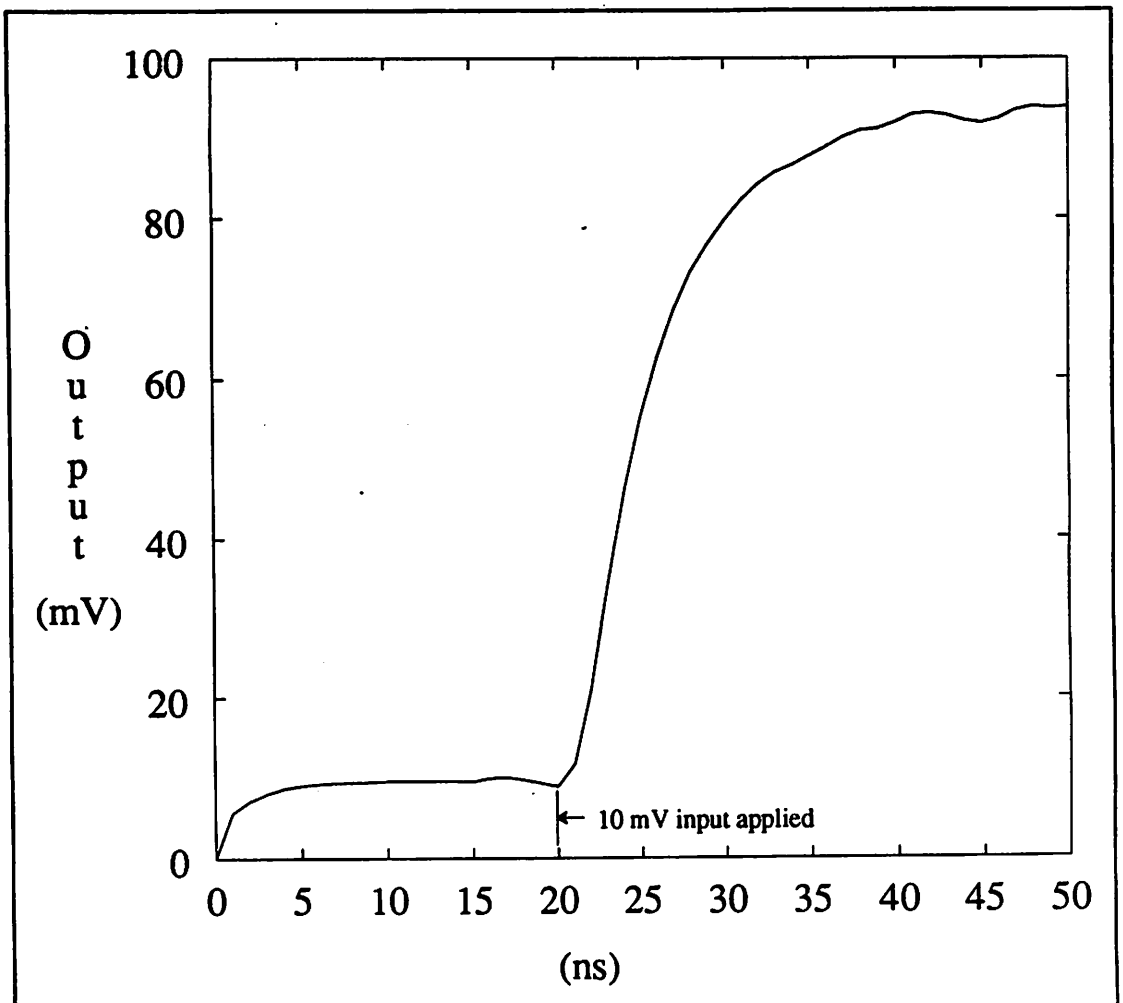


Figure 4.47 - Transient response of one comparator stage

Comparator stage simulation	
Power-supply Voltage	± 2.5 V
Power	1 mW
Open-loop Gain	-22
-3 dB bandwidth	35 MHz
Unity-gain bandwidth	700 MHz
Phase Margin	35 °

Table 4.5 One comparator stage simulated response.

5.6. Latch

The last comparator stage is followed by the strobed latch[2] shown in Figure 4.48. This latch has input transistors M1 and M2. There are two latching halves. First the NMOS devices at the input and then the PMOS loads. In reset mode the strobe is low and transistors M9 and M10 are off

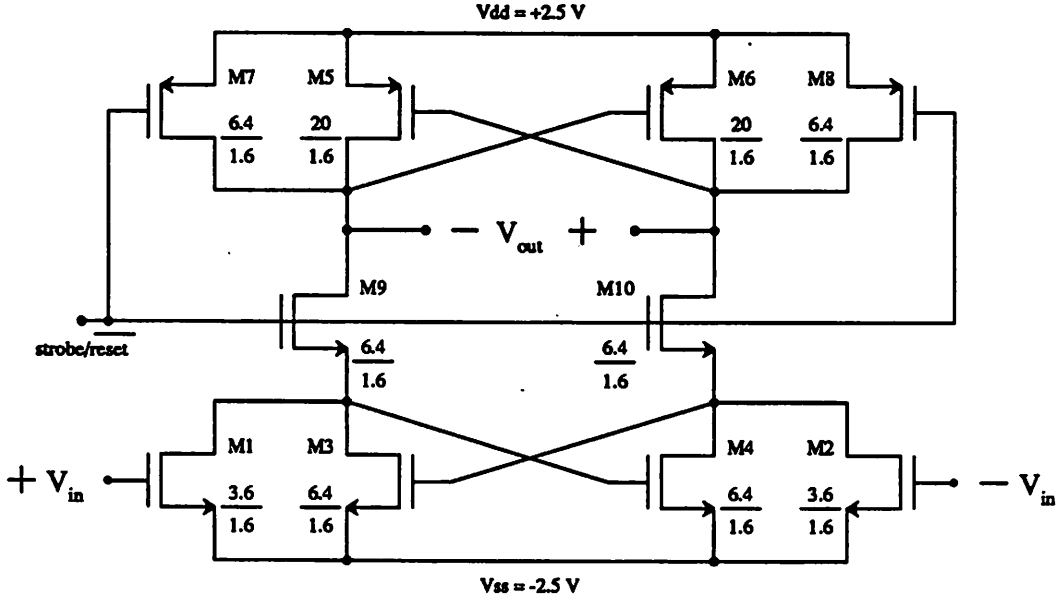


Figure 4.48 - Analog latch

disconnecting the input from the load. M7 and M8 pull the outputs high while any input voltage pulls the drains of M3 and M4 low. When the strobe goes high the two halves are connected and the positive feedback latching occurs. In a conventional latch the time constant is C/g_m, but in this case there are 2 latching elements so the time constant decreases to $\frac{C}{g_{mN}+g_{mP}}$. The simulated results are in Table 4.6. As with the comparator the layout is symmetric with matched parasitic capacitances and gates perpendicular to the axis of symmetry.

Latch simulated performance	
Power-supply Voltage	± 2.5 V
Power (DC)	0 mW
Minimum Input Signal	10 mV
Delay time (V_{out}=3.8 V)	5 ns
Reset time	3 ns

Table 4.6

CHAPTER 5

Prototype Controller

1. Introduction

The main function of the controller and test board was to generate the 12 clock signals needed to run the ADC. Extreme versatility was desired. The repetition rate, phase and pulse width of the clock signals had to be varied to get the maximum conversion rate with minimum error. The board also supplied bypassed power and biases as well as output data latches, downsampled data latches and a DAC.

1.1. Timing Requirements

Extreme flexibility in the ADC timing was needed. Although the ADC runs at a 5-Msample/s rate, testing was to begin at 1-Msample/sec, and 10 or 15-Msample/sec rates had to be provided for. To eliminate any settling-time effects the ADC was even tested at 100 ksamples/sec. Figure 5.1 shows the ADC timing diagram for 5-Msample/sec operation. The extreme flexibility required that all clock edges be able to be set within about 2-3% of the clock period with 2 ns as the minimum time increment with 10-Msample/sec operation.

1.2. Previous Clock Generators

Two methods had been used in the past to generate similar clocks. The first used a digital word generator. This is commercially available or can be built. It operates by programming into memory the value, high or low, of each clock signal at every time increment. These words are then read out of the memory and used as the clocks. This offers extreme flexibility and is easily changed by reprogramming the memory. The drawback is that to get 2 ns resolution the memory must be clocked with a 2 ns period, that is at 500 MHz. Commercial generators operate at least 10 times slower and building one would be unfeasible.

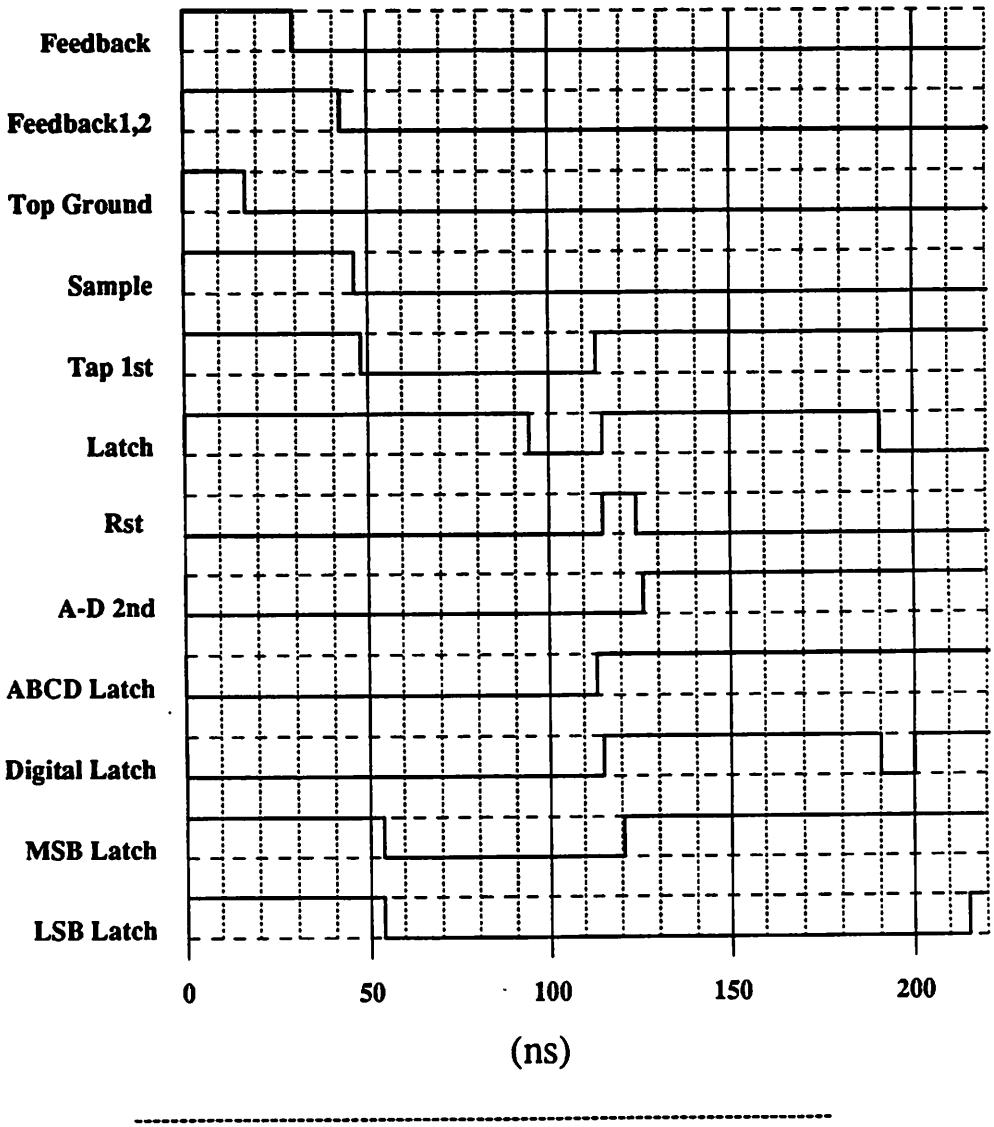


Figure 5.1 - ADC timing diagram for 5-Msample/sec operation

A second scheme is based on pulse counting. Programmable counters are set to count a master clock until the start of a desired clock edge and another counts for its on time. This again has the needed flexibility but also requires the same 500 MHz clock rate so it is unfeasible.

1.3. Delay Line Method

Delay lines are available with taps at .5 ns to 1000 ns intervals. Thus if a 5 MHz signal is propagated down a line with 2 ns taps, 10 outputs can be provided with delays of 2, 4, ... 20 ns. This gives the needed 2 ns resolution and the highest clock rate is the 5 MHz conversion rate. The basic way of generating a pulse with arbitrary delay and width is shown in Figure 5.2.[21] A variable-rate pulse is propagated down the delay line. SW1 (SW2) selects the tap for the rising (falling) edge and that goes to the "set" ("reset") input of the R-S flip flop, the result is the desired clock. Thus the delay line taps select the delay and pulse width while the pulse generator sets the overall repetition rate. To get lower speed operation longer delay lines were used. The clocks had the same phase and duty cycle but were lower frequency. Many of the clocks were not simple pulses but they could be generated from a basic pulse, or its complement, and could be "anded" or "ored" with another pulse to generate complex clocks of more than one pulse per period.

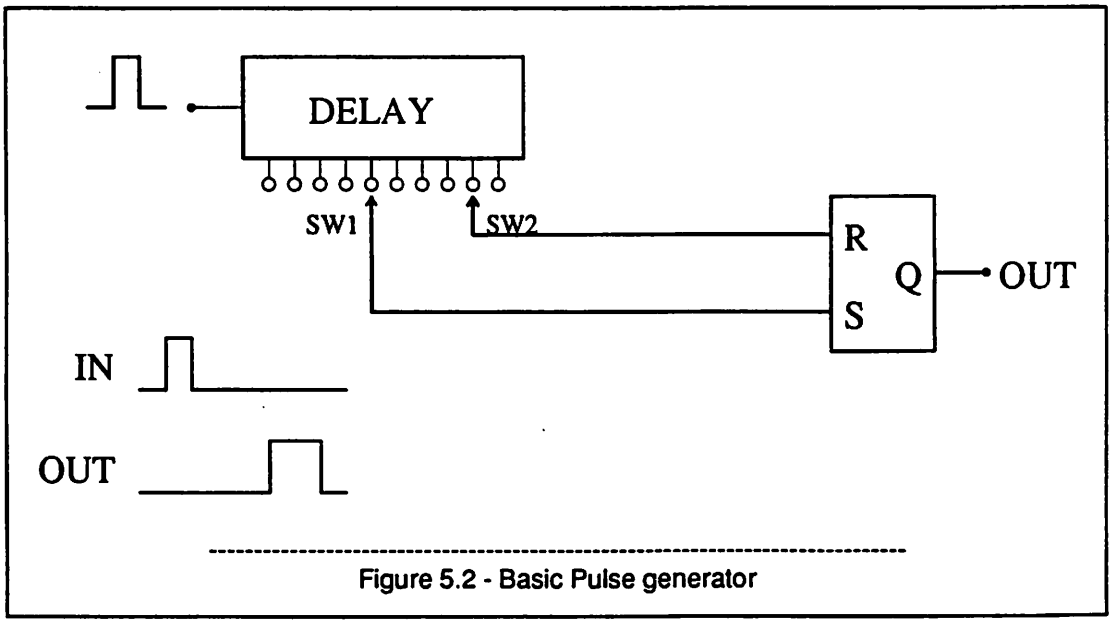


Figure 5.2 - Basic Pulse generator

1.4. Test Board

The test board also provided other support for the ADC. The digital output was latched at the sample rate. Then it was downsampled by a factor of 2 to 4095 and latched again. This allowed the ADC to run at a rate faster than the Sun computer taking the data for testing. A DAC on the ADC output gave a quick qualitative check of the ADC operation. A jumper area was provided so that the DAC inputs could be connected to the ADC outputs arbitrarily. If the bits were connected MSB to MSB and LSB to LSB the standard transfer curve is obtained. However as they are shifted and the ADC MSBs dropped, repeated staircases are obtained and this gives more resolution in checking the DNL.

The clock outputs were also buffered to drive a cable and probe-card for testing wafers before packaging.

CHAPTER 6

ADC Testing

1. Introduction

An ideal ADC transfer curve along with nonlinearity errors is shown in Figure 6.1. The object of testing is to measure the ADC's transfer curve as accurately as possible under actual operating conditions. Although the ADC transfer curve shows the relationship between the analog input and the digital output there are other factors that influence the digital output such as power supply noise and noise internal to the ADC. A second consideration is the sample-to-sample timing errors which don't manifest themselves as errors in the static transfer function but are errors.

The most common ADC test is to measure the transfer curve statically. There are many ways to do this but they suffer from reliance upon other ADCs or DACs. Moreover, they are static measurements and often have little relationship to the ADC's real performance under dynamic conditions.

Many dynamic tests measure some quantity related to what the ADC is being used for. These include differential gain and differential phase[22,23] that are important for NTSC video encoding. Signal-to-noise ratio (SNR) can be measured either digitally through a DFT of the output codes of a digitized sinewave or in the analog domain on a D/A converted and reconstructed sinewave. This is a popular test since SNR is so universally known. Other tests include noise-power ratio, beat frequency tests, sinewave curve fitting and fast Fourrier transform (FFT) based results such as intermodulation distortion.

Most of these test results can be derived from the ADC transfer curve if it could be measured under the same dynamic test conditions. Now with code-density testing[24] based upon a histogram of the ADC output codes, the ADC transfer curve can be accurately measured under dynamic operating conditions.

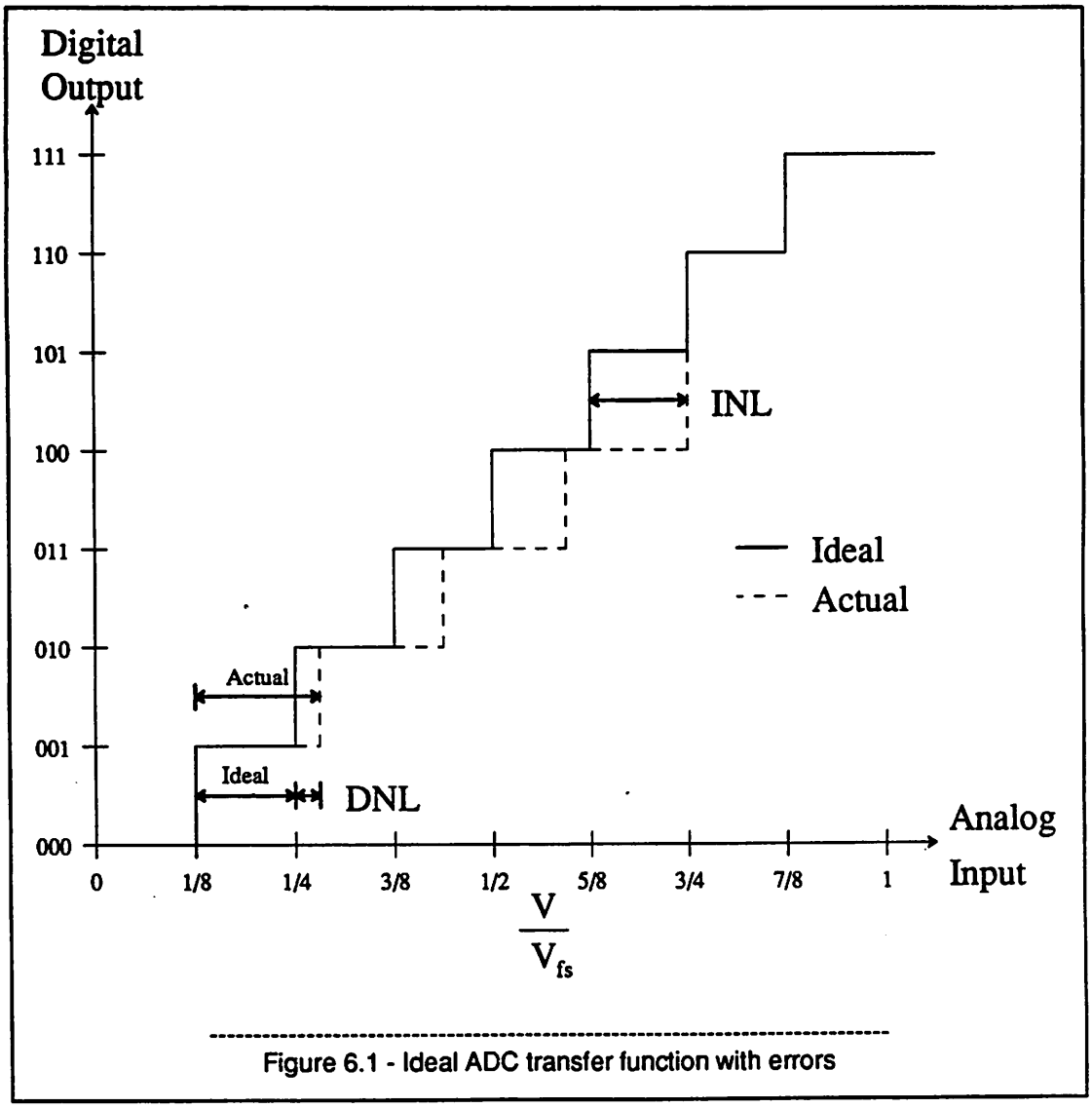


Figure 6.1 - Ideal ADC transfer function with errors

1.1. ADC Errors

There are common errors that show up in the ADC transfer function. These are offset, gain, differential nonlinearity (DNL) and integral nonlinearity (INL). An offset is a shift in the ADC transfer curve. This error is usually unimportant for an ADC except in an instrumentation use since there is often an offset adjustment in the system and the error is of little consequence since it doesn't cause nonlinearity. A gain error is a change in the slope, from the ideal slope of 1, of the line drawn through

the ADC transfer function endpoints. Again this error is of little importance since there is usually a gain adjustment in the system and it doesn't cause nonlinearity.

There are two types of nonlinearity errors, differential nonlinearity and integral nonlinearity Both are measured in LSBs. The first is illustrated in Figure 6.1 and is the distance between adjacent transitions minus 1 LSB. Thus there is a differential nonlinearity for each transition region. Integral nonlinearity is also shown in Figure 6.1 and is the difference between a transition and the ideal transition. DNL and INL each can be computed from the other so they are not independent but have different uses. The DNL shows the small-scale errors from transition to transition. The INL gives the over-all shape of the transfer curve. Nonlinearity errors are important because they lead to distortion in the output spectrum and degrade the SNR.

1.2. Code-Density Test

The code-density test is based upon a histogram of the ADC output codes obtained from randomly sampling an input with a known probability density. From this histogram the ADC transitions are statistically estimated. Comparing these transitions to the ideal transitions gives the errors.

An example follows with a full-scale ramp voltage. $CH(i)$ is the cumulative histogram of i bins of the output codes. A full-scale ramp waveform is shown in Figure 6.2a. Its probability density is $p(V) = \frac{1}{V_{fs}}$ and shown in Figure 6.2b. The cumulative probability distribution function is in general:

$$P(V_x) = \int_{-\infty}^{V_x} p(V)dV \qquad 0 \le P(V_x) \le 1 \tag{6.1}$$

This is $P(V) = \frac{V}{V_{fs}}$ and $V = V_{fs} \cdot P(V)$ for the ramp and shown in Figure 6.2c. $P(V)$ is replaced by the cumulative histogram density $\frac{CH(i)}{N_t}$ giving an estimate of the i^{th} transition voltage, $\hat{V}_i$.

$$\hat{V}_i = V_{fs} \cdot \frac{CH(i)}{N_t} \tag{6.2}$$

Thus all of the transition voltages are computed from the histogram of the output codes.

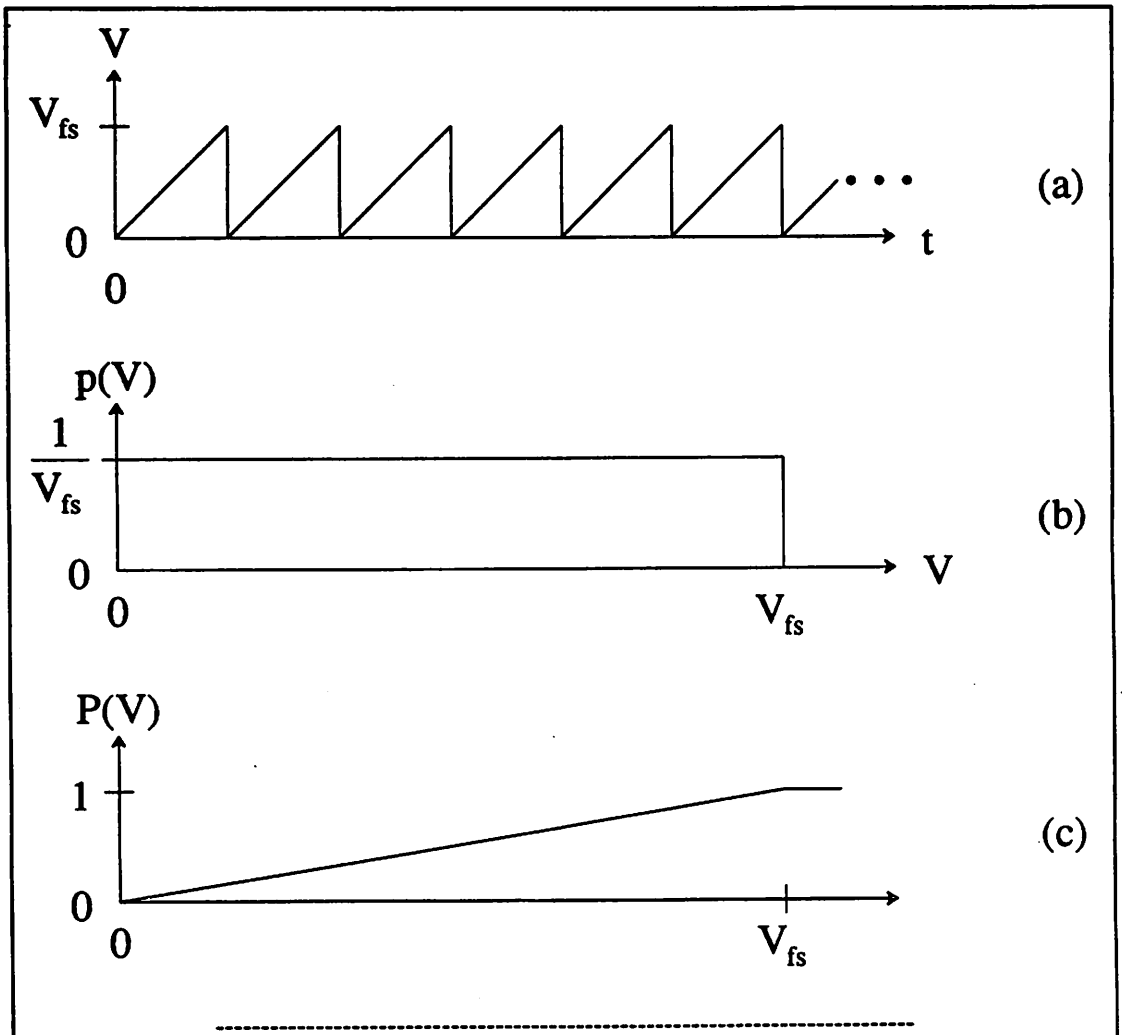


Figure 6.2 - (a) Input (b) Probability Density (c) Cumulative Probability Distribution Function

1.3. FFT Testing

Given the ADC transfer curve that has been measured with the desired input frequency the output spectrum of the ADC could be computed for any input. However it is often easier to digitize the input voltage and use an FFT to get the output spectrum to compute the SNR and look for distortion. For an ideal ADC with many quantization levels and sampling a sinewave the maximum SNR is:

$$SNR = 6.02n + 1.76\ dB \tag{6.3}$$

n gives an estimate of the number of bits that are being obtained.

CHAPTER 7

Experimental Results

1. Introduction

The prototype ADC was tested with two characterization methods for the most part. First it was tested with a code-density test [24] and then with an FFT test. The code-density test is used to directly measure the DNL and INL. The second test method is based on a DFT using the FFT algorithm. The results from a sinewave input are harmonic distortion (HD) and SNR and are measured as a function of input frequency. Two sinewaves can be used as an input and the intermodulation distortion (IM) is measured. Other tests include Power-Supply Rejection Ratio (PSRR) and noise measurements.

The next chapter describes how special interpretations of the code-density test results can be used to extract information about specific error sources and the converter's limitations.

1.1. Test Parameters and Optimization

For each of the tests there are many parameters that could be varied. These include the input frequency and sample rate (and all clock phases), which would show frequency dependent effects. In addition, the external voltages such as the power supply and voltage references could be varied. Even the 3 bias voltages for the comparator could be overridden. Also the ROM used in the binary encoder has a bias adjustment. Lastly the temperature could be varied.

Exhaustive testing under all conditions is clearly unfeasible and simple tests done early on showed which were the critical parameters to be studied. The most important are the clock-phase times which determine the overall conversion rate.

Using the DNL as the primary test and optimizing the converter's performance there was often the "parameter" versus "error" tradeoff. Some baseline of performance needed to be set to guide the optimization of parameter changes. That baseline was 0.7 LSBs of DNL. A parameter could be

changed as long as the DNL stayed below 0.7 LSBs.

1.2. Standard Test Conditions

The standard test conditions, except where noted otherwise, are a 49.5 Hz full-scale sinewave input, 5-Msample/s conversion-rate, 5 V power supply with 2.5 V analog ground and nominal room temperature. The reference voltage is 4 V (+/- 2 V) and the 1/4 and 3/4 point resistor-string taps are used in the final INL and SNR testing. The ROM bias is 3.15 volts and the common-mode input voltage is 1.7 V.

1.3. Layout and Design Error Correction

There is both a small layout error and a small design error which prevent complete testing of the ADC. The layout error prevents 10-bit operation in half of the negative range (codes 0-511). In every other 32-LSB subsection of the negative range the codes are missing. This is not fatal since the 5-bit MSB decisions are unaffected and as Figure 7.1 shows the INL of the first and second steps is almost identical so correction of the layout error will fix the problem.

A design error in the thermometer to 1-of-n code circuit causes the binary outputs of the second step to be inverted if the MSB code is odd. This can be fixed in hardware with the circuit of Figure 7.2. However the error also causes bins 33 to 63 to be shifted to 32 through 62 and repeating each 64 codes in the positive range. Thus bin 32 is twice as wide as expected and code 63 missing. A software filter was written to shift the codes back and put half of the codes from bin 32 into bin 33.

2. Code-Density Test

An input with a known probability-density function, in this case a low-distortion sinewave, is applied to the ADC. A histogram of the digital outputs is constructed and used to statistically estimate the ADC transition voltages.[24] The deviation of the transition voltages from the ideal is the INL. The difference between adjacent transitions minus 1 LSB is the DNL.

The test requires a low-distortion input and a minimum number of samples for reproducible and statistically confident results. The Sun computer taking the data samples at about 40 ksamples/s but

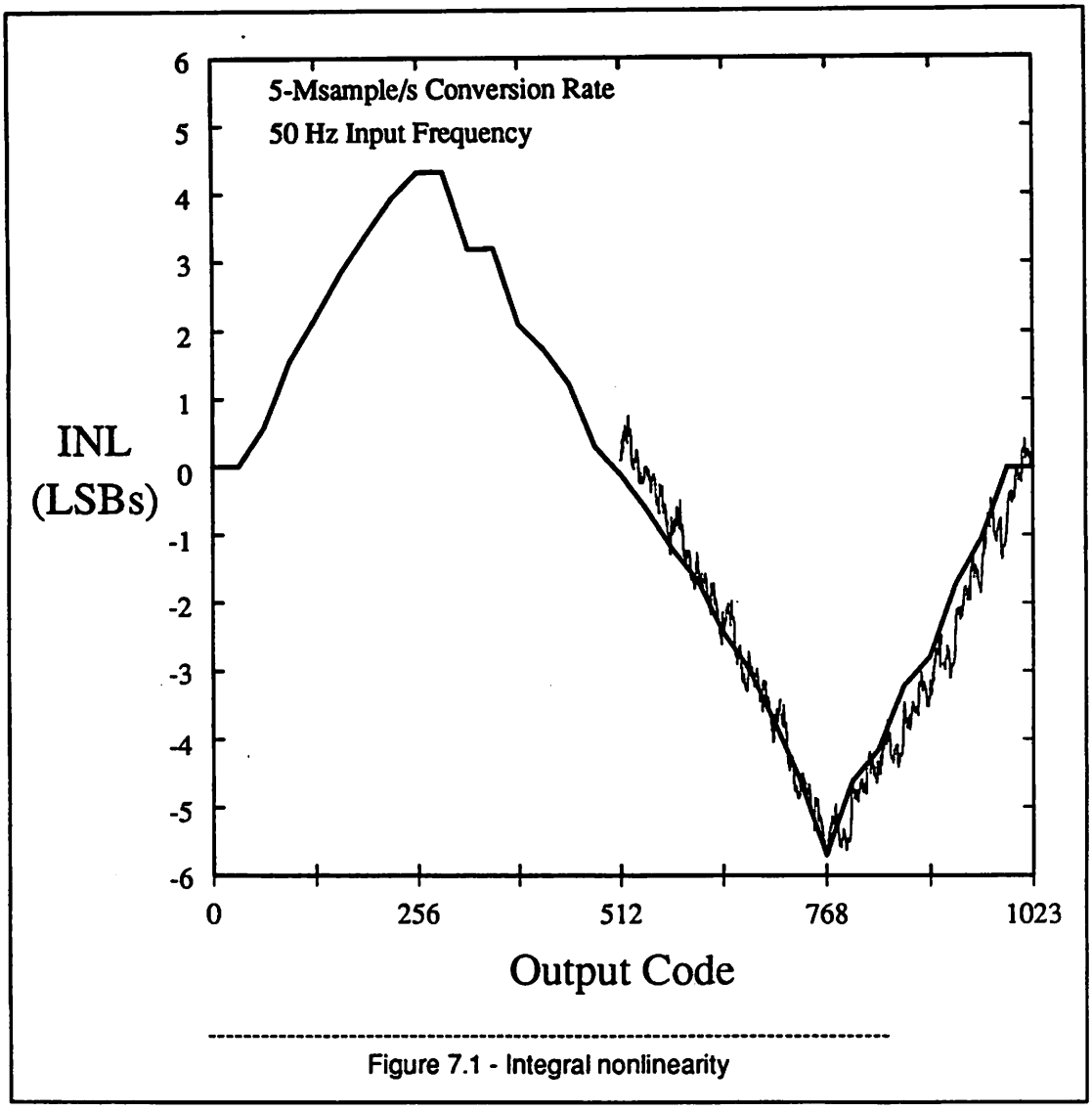


Figure 7.1 - Integral nonlinearity

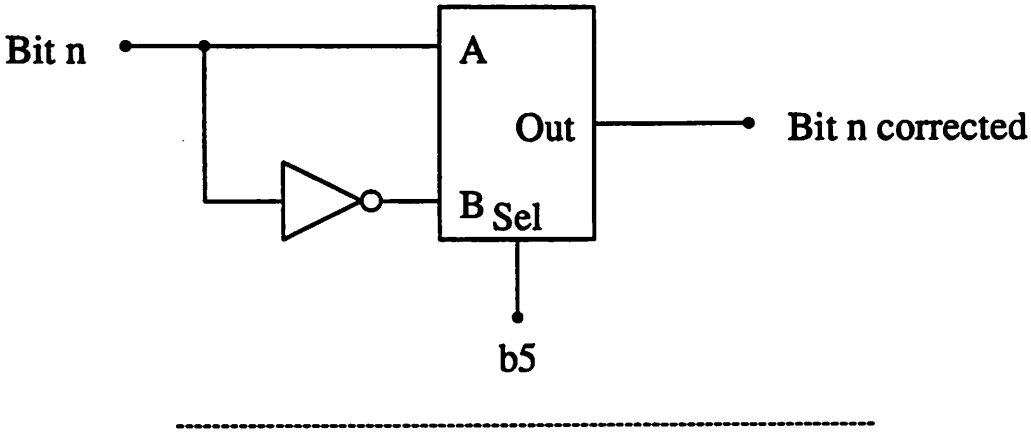


Figure 7.2 - LSB code correction circuit

the ADC is running at 5 Msamples/s so only 1 out of every 125 samples is used. Since the test is based on "random sampling" this downsampling is justified.

For quick testing and for adjusting timing and other parameters 1 million samples were taken corresponding to 0.1 LSB precision and 99% confidence in the DNL tests. For final testing 4 million samples were used giving twice the precision or increased confidence or a combination of both. The INL however needs about 2^n times more samples than the DNL needs for the same confidence level and precision. One billion samples for 0.1 bit precision and 99 % confidence are unfeasible since the time required at the 40 ksamples/s downsampled data rate is 7 hours and unfeasible from two stand-points. First the required time is too long and secondly in that time any drift in the input source's amplitude or the reference voltages could affect the results. However the SNR test uses many fewer samples and gives a very good indication of the INL. Its disadvantage is that it is somewhat immune to DNL errors. But by using both tests the linearity of the ADC can be accurately determined.

2.1. Code-Density Test Results

Figure 7.3 shows the DNL in LSBs at a 10-bit level plotted on the y-axis versus output code on the x-axis for the positive range. The maximum DNL is 0.6 LSB. The minimum is -0.5 LSB. Since the DNL never reaches -1 LSB there are no missing codes in this range.

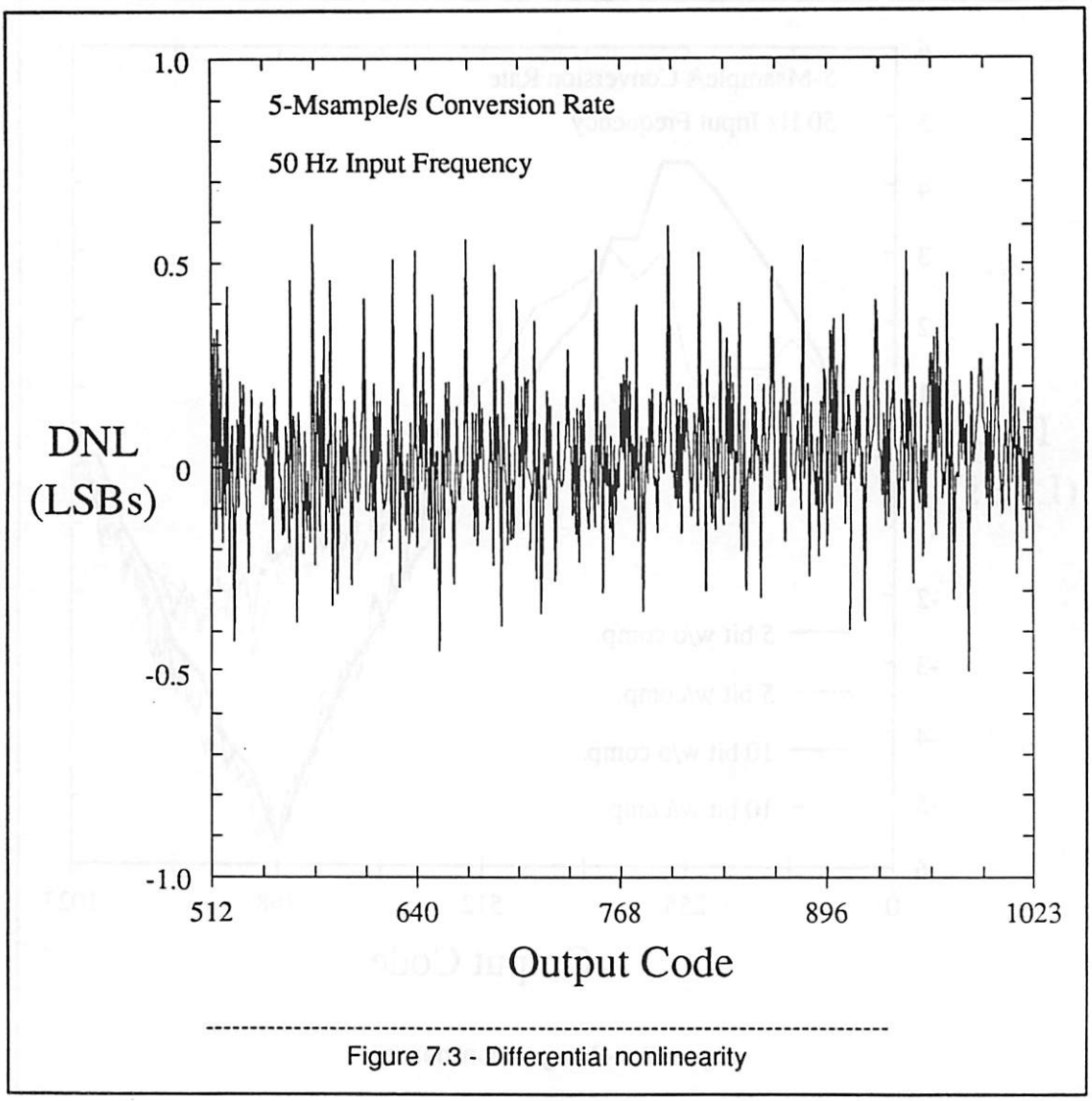


Figure 7.3 - Differential nonlinearity

Figure 7.4 shows the INL for 4 cases. The heavy lines are for the 5-bit MSB first decision. The dotted lines are for the case when the 1/4 and 3/4 point taps on the resistor string are used. The maximum INL is 3.0 LSBs with the compensation and 5.8 LSBs without it. The thinner lines are for the INL of the second step. Note that they fall on top of the first-step results showing that the INL is determined in the first step by the resistor string. Resistor-string matching sets the static, or low sample rate, INL and dynamic loading degrades it.

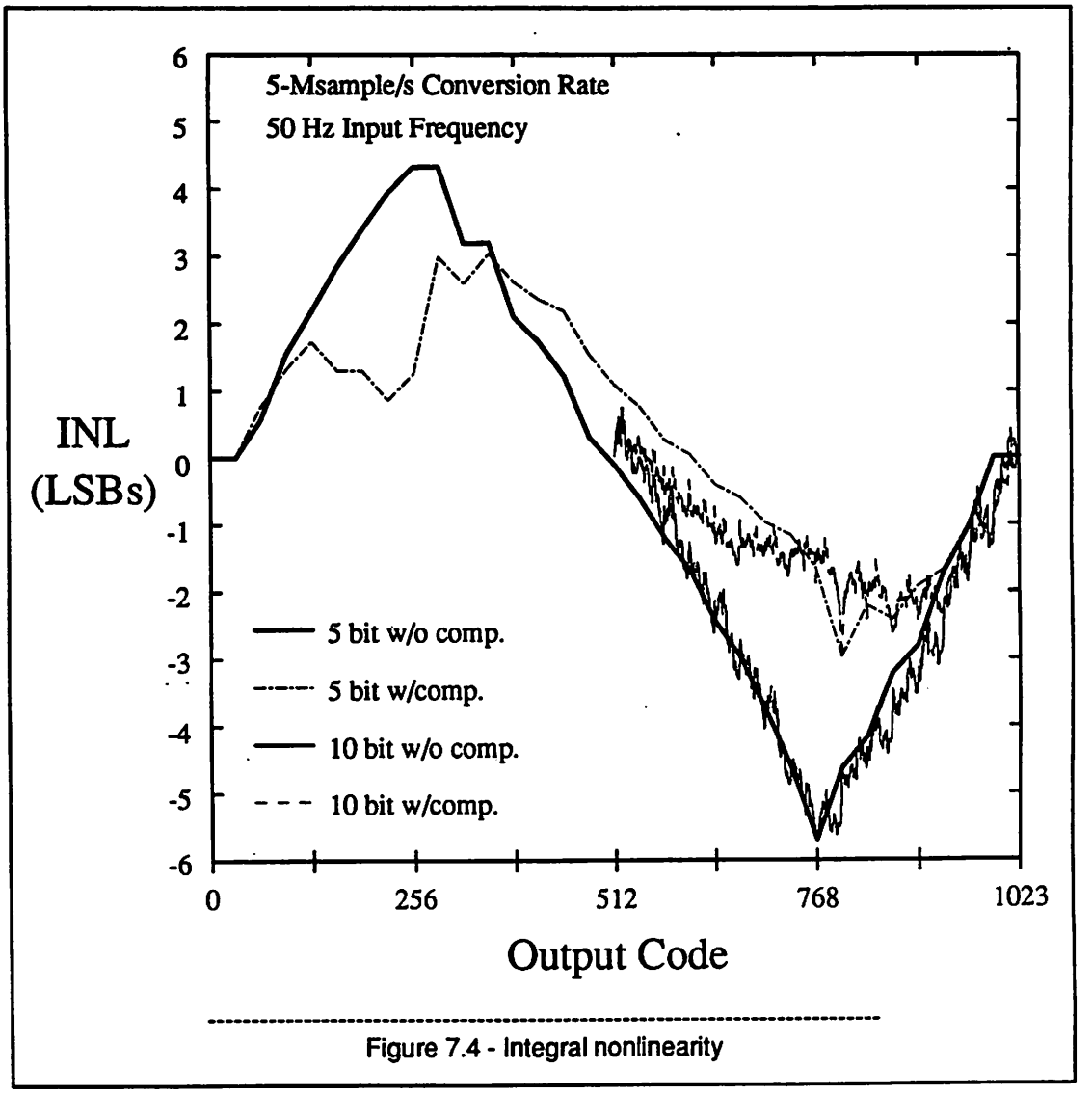


Figure 7.4 - Integral nonlinearity

3. FFT based Testing

Figure 7.5 shows the output spectrum of the ADC sampling a 49.5 Hz sinewave. Since the ADC doesn't operate fully in the negative range the circuit of Figure 7.6 was used to generate a fully differential input with an offset voltage so the input is only positive. Since only half of the ADC input range is used the converter is run as a 9-bit converter.

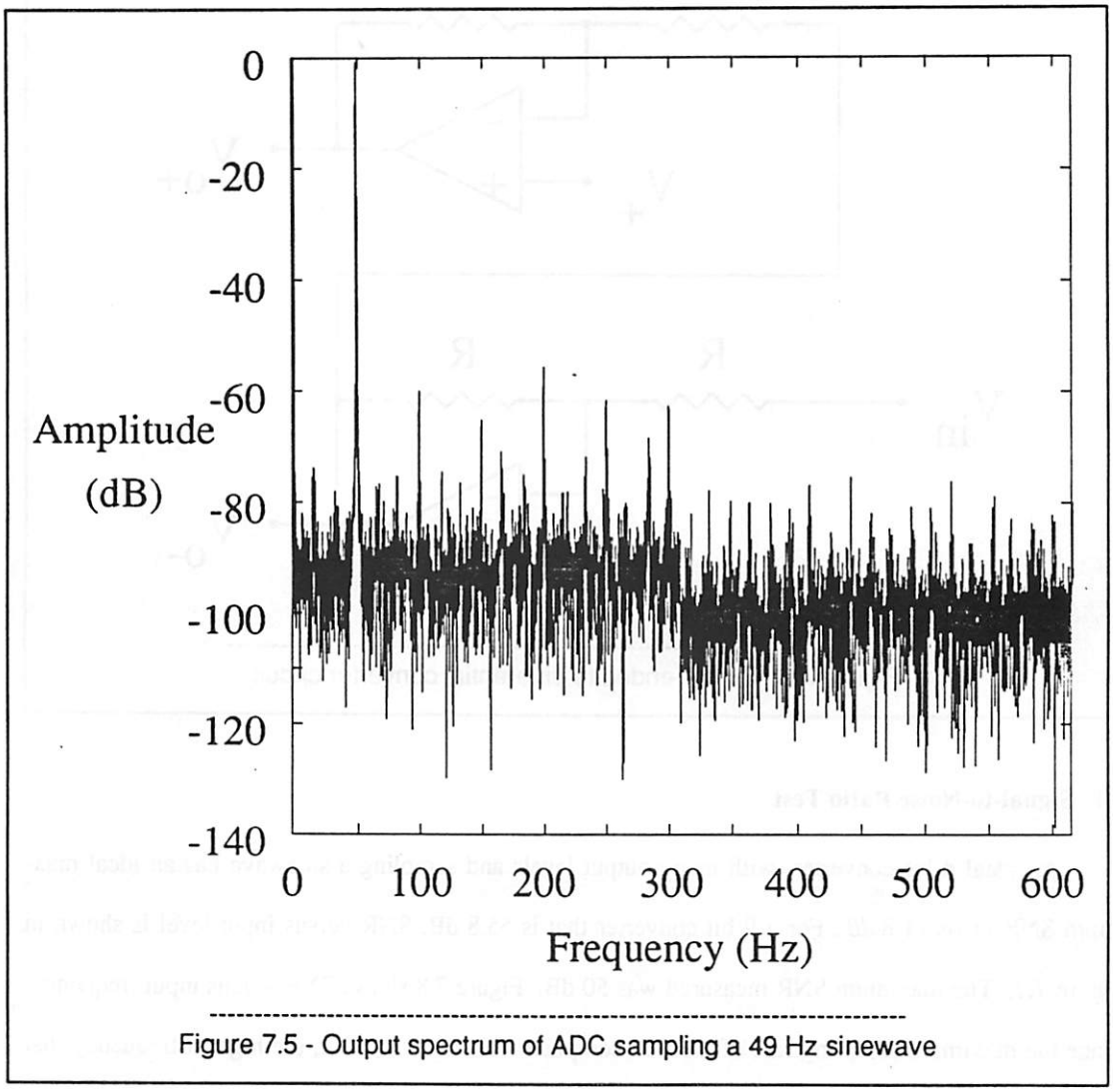


Figure 7.5 - Output spectrum of ADC sampling a 49 Hz sinewave

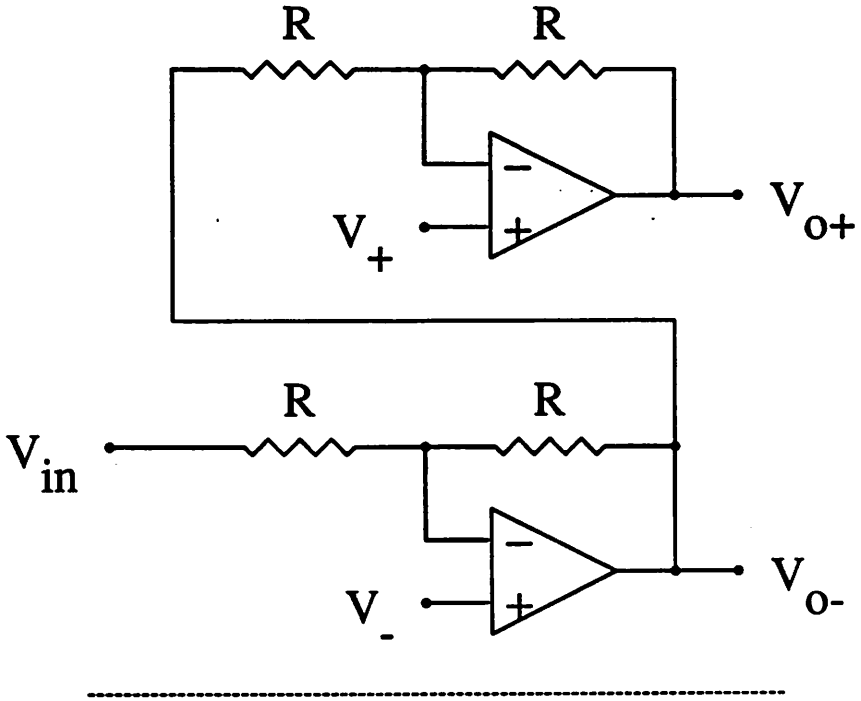


Figure 7.6 - Single-ended to differential converter circuit

3.1. Signal-to-Noise Ratio Test

An ideal n-bit converter, with many output levels and sampling a sinewave has an ideal maximum SNR of $6n+1.8\,dB$. For a 9-bit converter that is 55.8 dB. SNR versus input level is shown in Figure 7.7. The maximum SNR measured was 50 dB. Figure 7.8 shows SNR versus input frequency. Since the maximum sampling rate of the Sun computer was about 50 kHz, the highest frequency that could be distinguished in the FFT is half of that, 25 kHz. A 10 kHz input was the highest frequency used. Low frequency inputs were used instead of an external sample-and-hold circuit to use the charge-injection cancellation method of Section 5.4 in Chapter 4. Had the charge-injection cancellation not been needed higher frequency inputs would have been used to test the internal sample-and-hold circuit.

3.2. Intermodulation Distortion Test

An intermodulation distortion test can be run by inputing two sinewaves and looking for sum and difference frequency distortion products. The advantage of this test is that the harmonic distortion

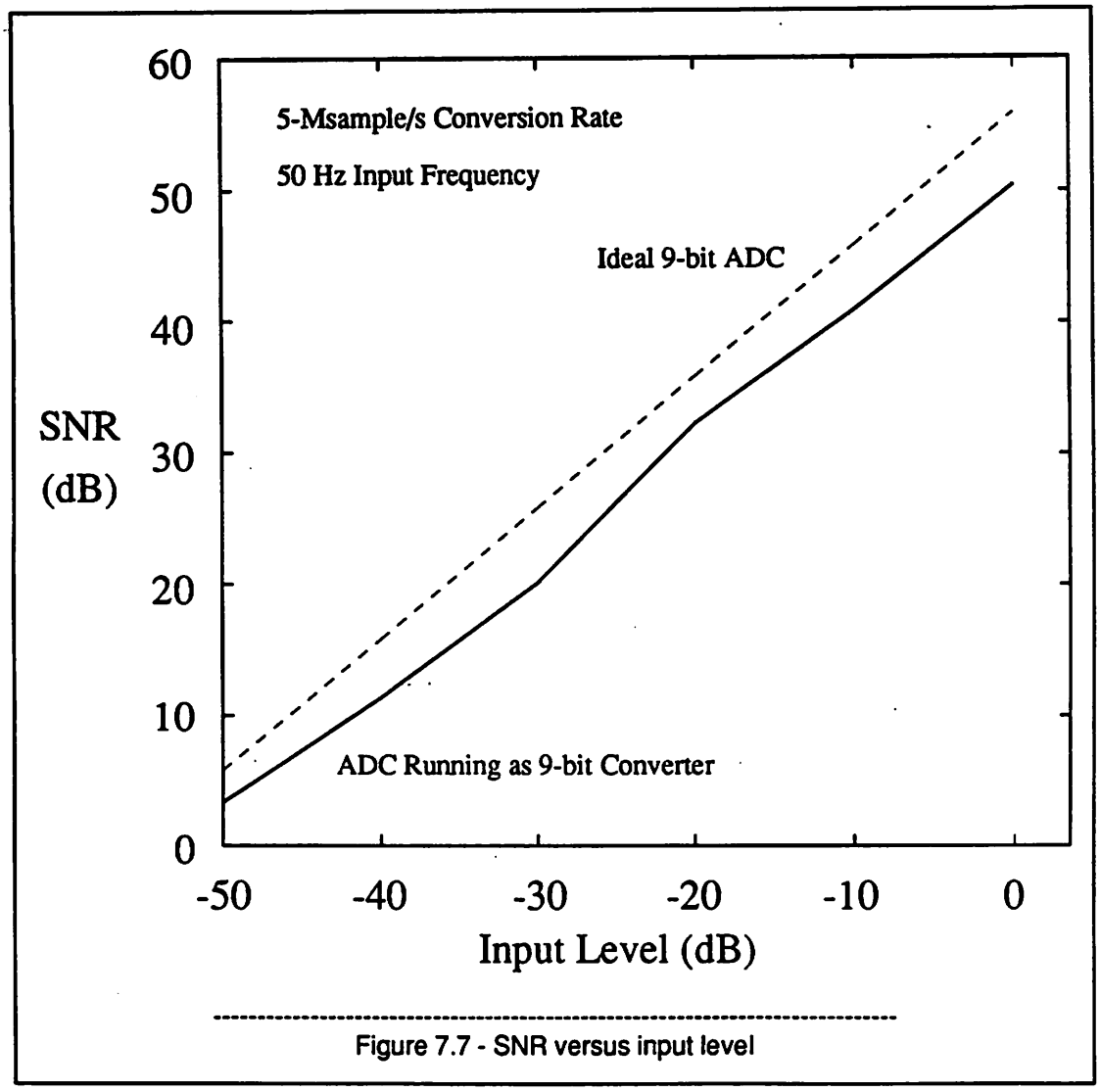


Figure 7.7 - SNR versus input level

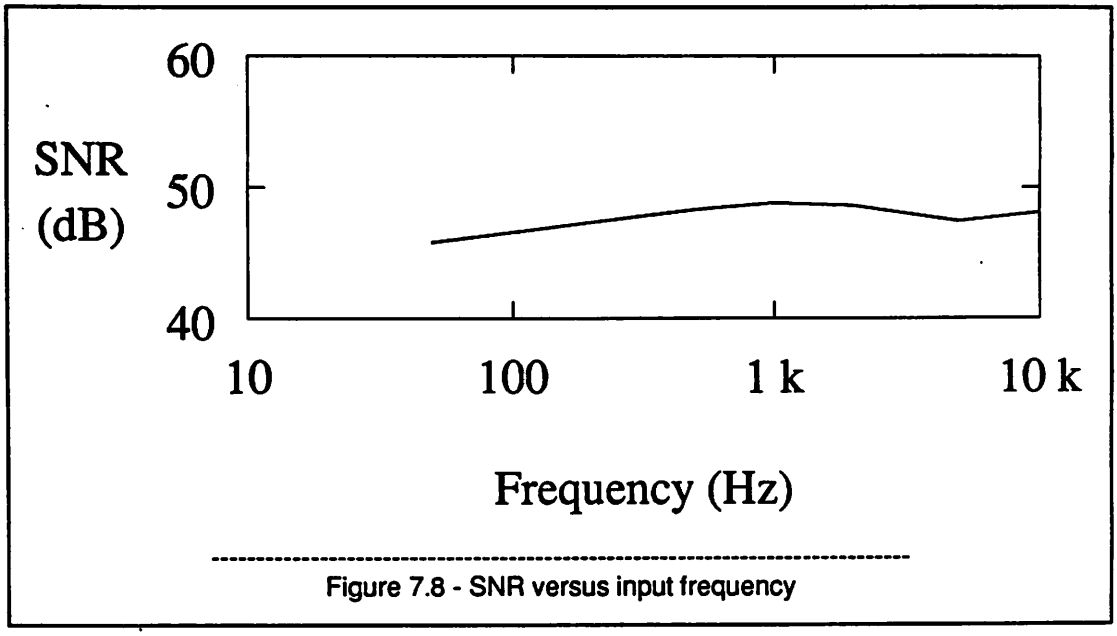


Figure 7.8 - SNR versus input frequency

in the input does not show up in the sum and difference frequency terms directly. The spectrum for 45 Hz and 50 Hz inputs are shown in Figure 7.9. The IM_2 component is about 6 dB above the second harmonic as predicted.[25]

$$IM_2 = HD_2 + 6\ dB \tag{7.1}$$

The effect of varying the input frequency was measured and SNR versus input frequency was measured. The performance versus sample rate was also measured. The first effect was that as the clock phases were shortened (sample frequency increased) the DNL would increase and there was a very definite threshold effect. So a maximum level of 0.7 LSB was set and the sample rate maximized. No effort was made to quantify the DNL at lower rates than the maximum 5-Msample/s sample rate. The second effect was that the INL degraded with increasing sample rate due to the dynamic loading on the R-string.

4. Power-Supply Rejection Ratio

The input can be a constant and the power supply varied to measure PSRR. The DC power supply rejection was very good. The power supply was varied from 3.8 to 5 V and the output code did not change.

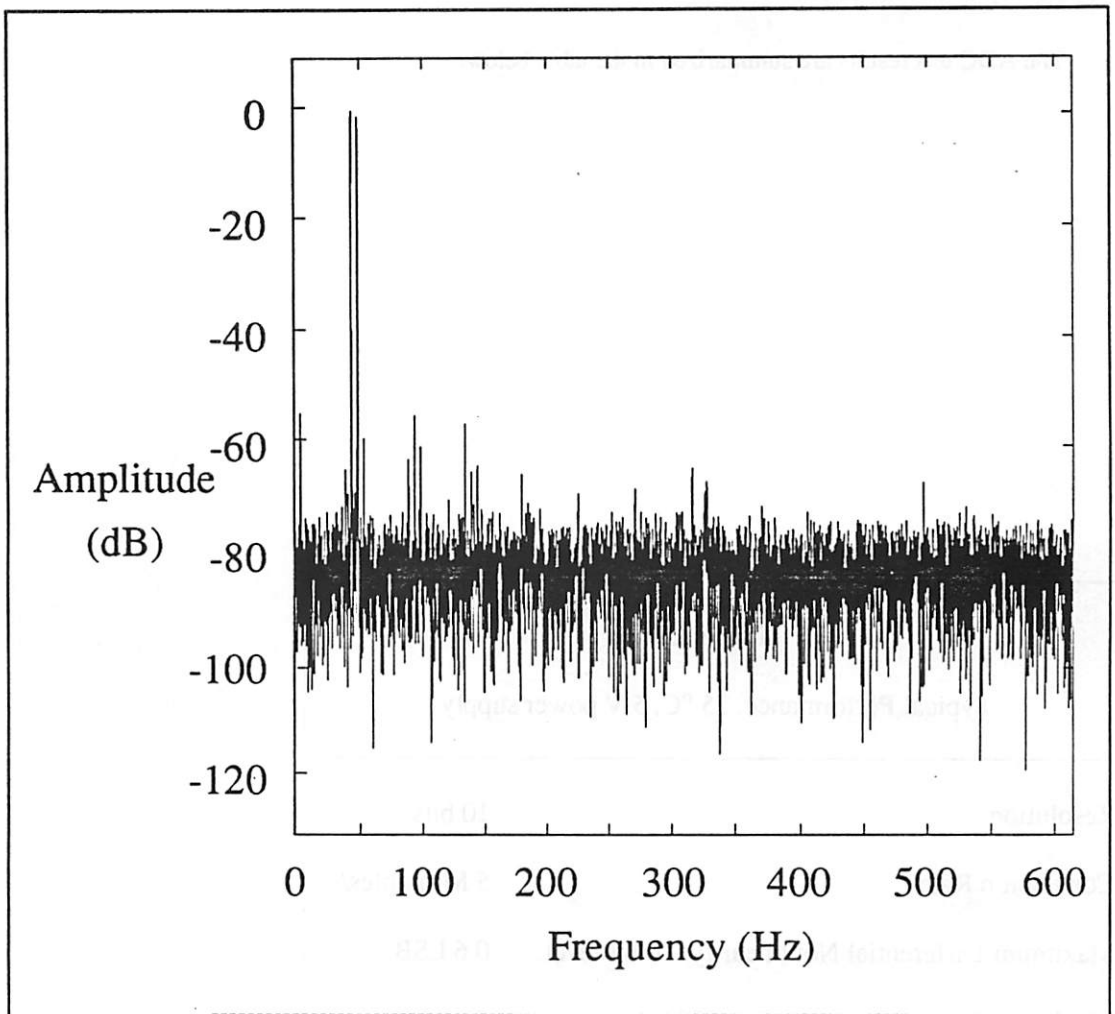


Figure 7.9 - Output spectrum with 2 sinewave inputs

5. ADC Noise Measurement

The ADC input noise which is largely the comparator input noise is indirectly measured by setting the input to a constant voltage and looking at the histogram of the output codes. The input was grounded and all of the codes were in the same bin. Thus the noise voltage is less than 1 LSB, 7.8 mV. Had the codes been in more than 1 bin an FFT could have been run to look at the noise spectrum.

6. Summary of Test Results

The ADC test results are summarized in the table below.

Table 7.1

Typical Performance, 25 °C, 5 V power supply

Resolution	10 bits
Conversion Rate	5 Msamples/s
Maximum Differential Nonlinearity, 10 bit level	0.6 LSB
Maximum Integral Nonlinearity, 10 bit level	5.8 LSB
Technology	1.6 µm CMOS
Input Capacitance	50 pF
Power dissipation[1]	350 mW
Area[2]	54k mils^2

1 Includes 60 mW dissipated in the resistor string.

2 Does not include clock generator or pads.

CHAPTER 8

Testing for Specific Errors

1. Introduction

The previous chapter detailed the results of "standard" ADC tests. However there are many other tests that can be run and can be tailored to the particular ADC to extract much more information than just the DNL or INL. These can be quite useful to find the cause of errors or the limitations of the ADC.

Several error sources are present, including resistor matching, capacitor matching, feedback switch charge injection and dynamic loading of the resistor string. The objective is to design tests and analysis methods to see the effects of these error sources.

2. DNL Analysis

Figure 8.1 shows the DNL of the ADC without canceling charge-injection errors. Two features are most apparent. First the nonlinearities are repetitive with a cycle of every 64 codes. This is expected since each of the taps on the resistor string is a code at a 5-bit level and each of those is subdivided 32 times by the capacitor array ADC. The LSB of the first decision determines how the capacitors are used. When the LSB is "1" rather than "0" the capacitor whose bottom plates went to the higher of the two reference voltages now goes to the lower of the two. Thus the capacitor array ADC does 32 subdivisions, exchanges the bottom plates and does 32 more subdivisions. Therefore the DNL errors caused by capacitor matching will repeat every 64 codes. However offset voltage which could be caused by uncanceled comparator offset voltage or feedback switch charge injection also is embedded in the DNL and will also repeat.

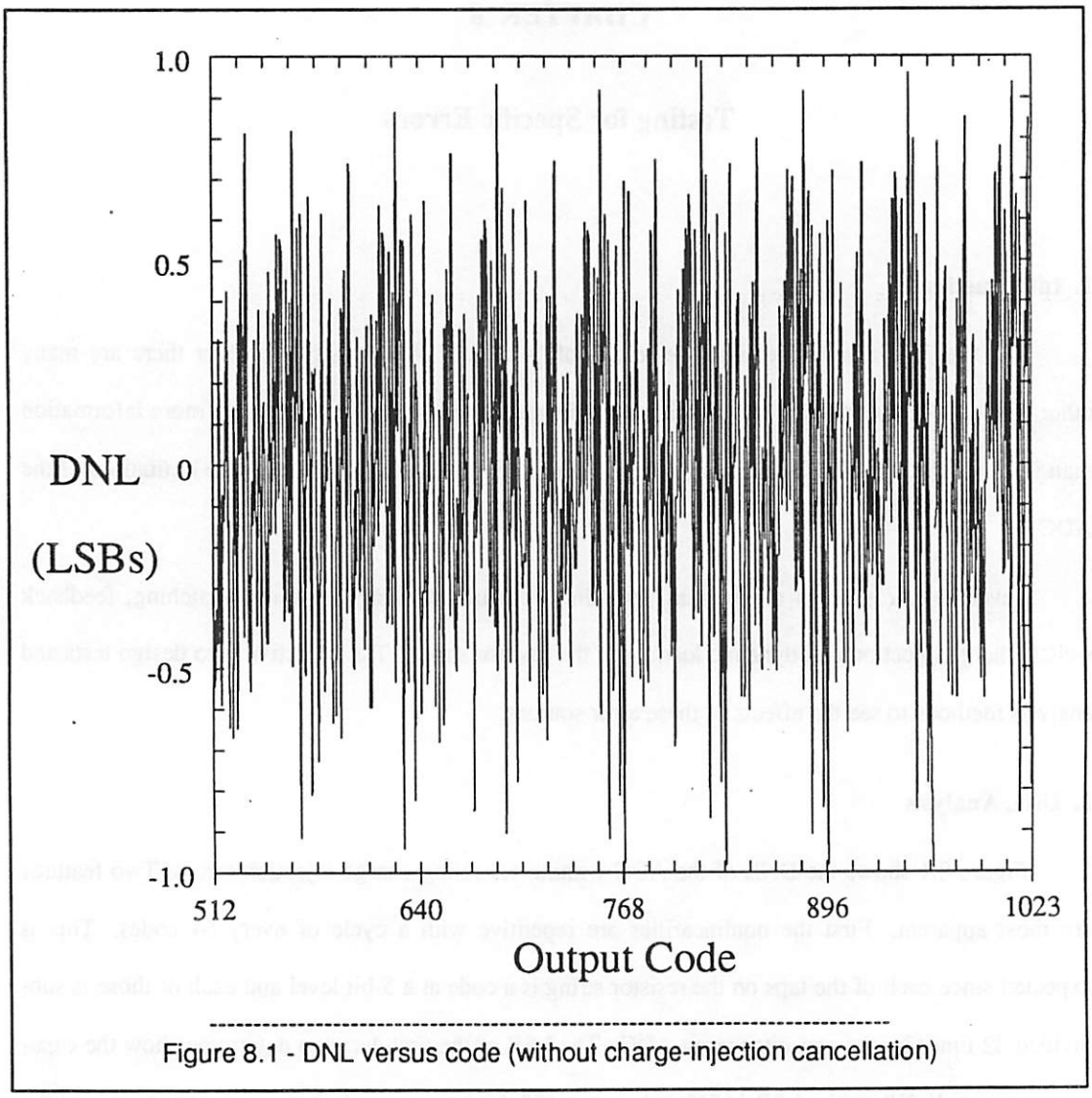


Figure 8.1 - DNL versus code (without charge-injection cancellation)

2.1. Repeated Errors

To check the assumption that the capacitor matching contributes errors that repeat a new way of looking at the data was developed. Since the errors were expected to repeat every 64 codes the transition voltages were divided into sets of 64 transitions and the integral nonlinearity over each set of transitions was computed. Only the "even" data, that when the MSB code is "0" was used. All of the data can be used but it has to be reversed when the MSB code is odd because the capacitor bottom plates

are connected to the reversed DAC outputs. Similar plots can be made with all of the data and the symmetry about code 31 is quite evident. These sets of nonlinearities are expected to be similar since the same capacitors and comparators are used repeatedly. Figure 8.2 shows these nonlinearities. The dotted line is the average of the results. The good agreement between the curves suggests that the errors in the DNL are caused by capacitor matching or comparator offset voltage. More information is needed to separate the two possible causes.

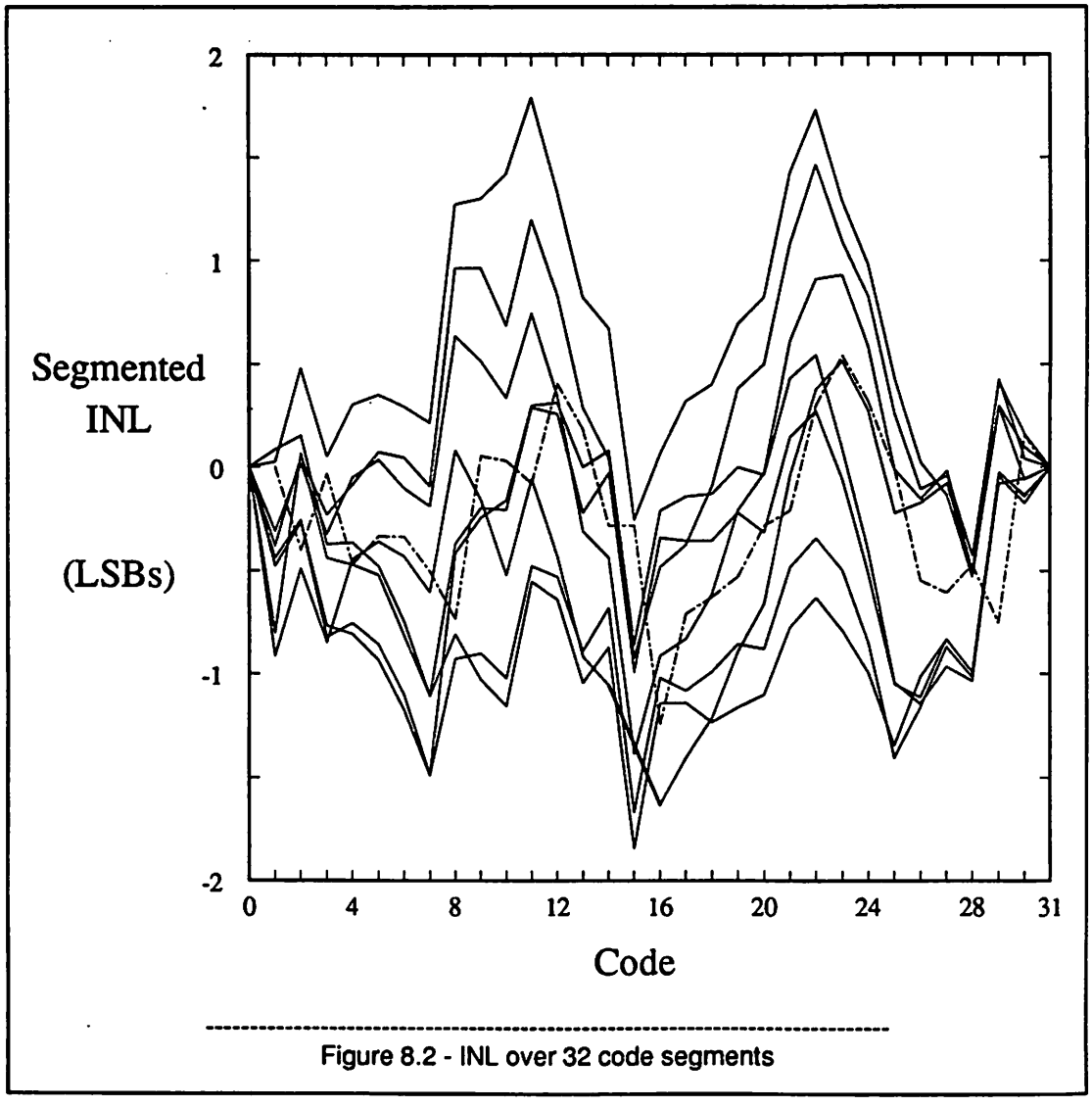


Figure 8.2 - INL over 32 code segments

2.1.1. Offset Voltage Test

The 5-bit INL is shown as the solid line in Figure 8.3 and is that of a flash converter The possible errors are resistor mismatch and comparator offset voltage. By comparing this data to that of the segmented average, shown dotted, the common errors should be due to comparator offset voltage since the comparator is used in both tests. The strong correlation in the peaks suggests that the comparator offset voltage is the cause of the errors.

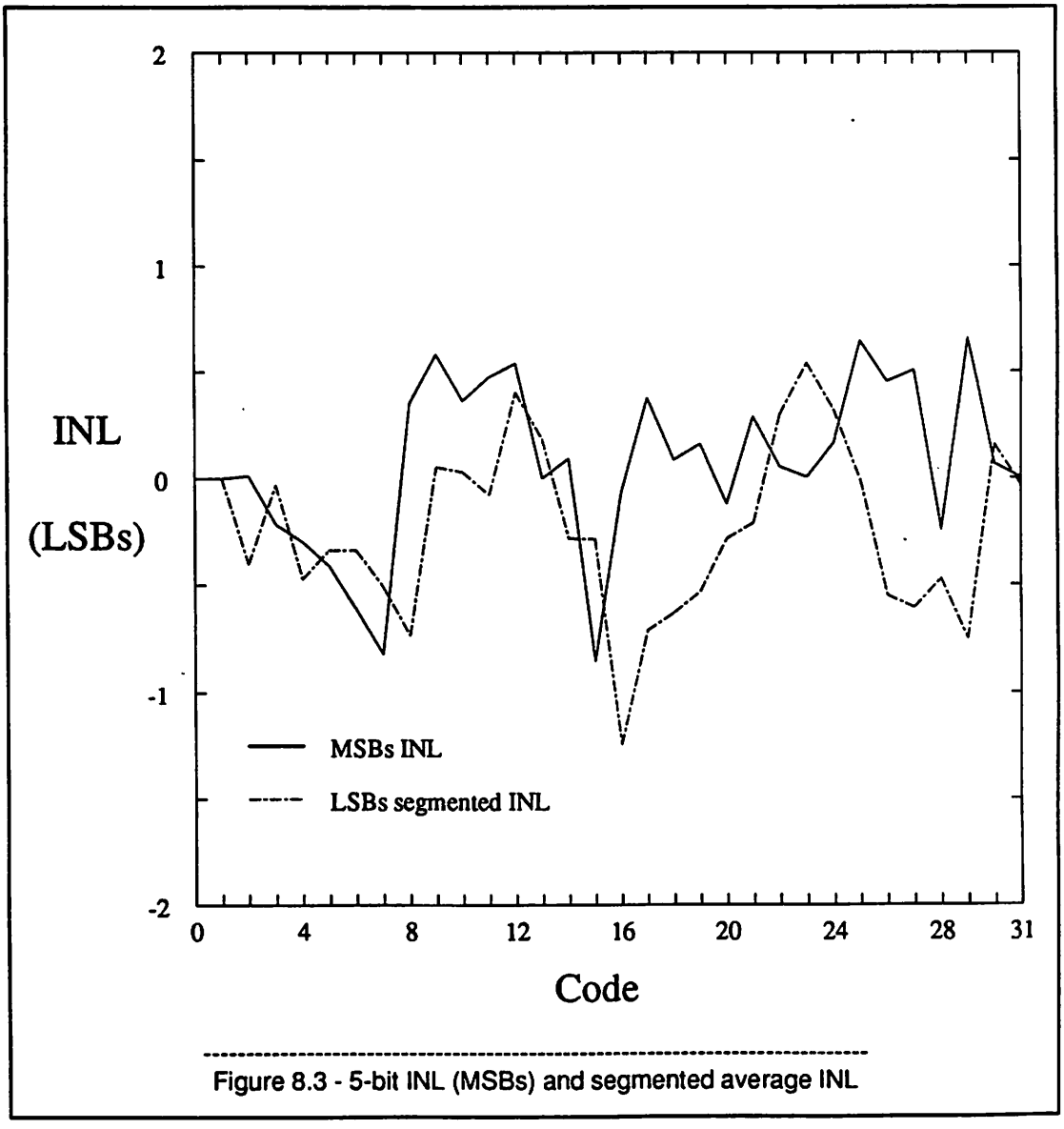


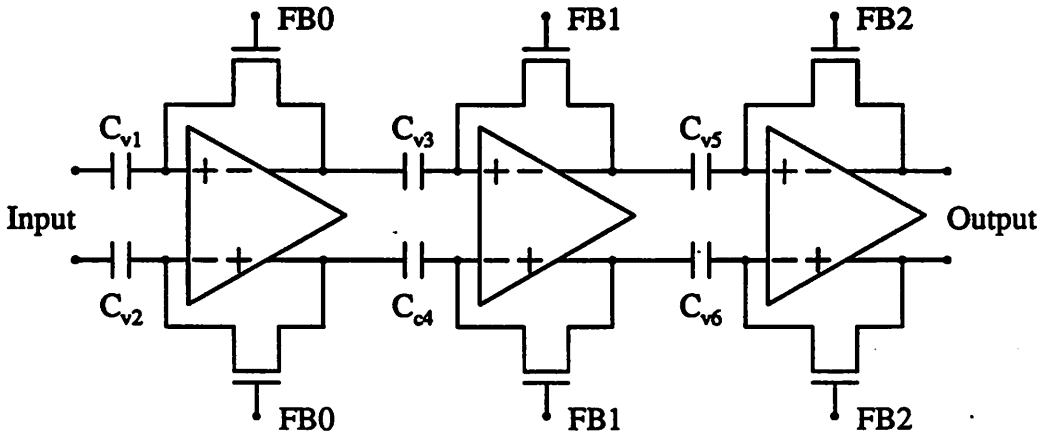
Figure 8.3 - 5-bit INL (MSBs) and segmented average INL

2.1.1.1. Offset Voltage Cancellation

The comparator offset voltage consists of two components. One is the offset voltage due to mismatch in the comparator input transistors which is uncanceled due to finite amplifier gain and incomplete settling during the "offset cancellation" phase. The other is offset due to charge injection from the feedback switches. The charge-injection error is reduced from the normally expected error in a single-ended configuration by having a fully differential architecture. The charge-injection error from the first amplifier stage also dominates over the second and third stage errors since when their errors are referred to the input they are divided by the gain of the preceding stages. Figure 4.38 in Chapter 4 showed the 3-stage comparator configuration. The feedback switches for each of the 3 stages can be controlled independently. By using a method developed by Allstot[19] the charge-injection error can be canceled almost completely. Figure 8.4 shows how this is done. The timing diagram shows the switch configurations. In the first phase all feedback switches are closed. Each capacitor stores the offset voltage of the comparator following it. If all 3 feedback switches were opened simultaneously the capacitors would have an error voltage imposed upon them from the switch charge injection. However by opening the switches properly the error can be reduced. In phase 2 of the timing diagram only the first switch is opened. Its charge injection error is stored on the first capacitor, amplified by the first amplifier and stored on the second capacitor effectively canceling it. This procedure can be repeated for each switch depending upon the accuracy needed. With this cancellation applied to just the first stage the DNL is greatly improved as seen in Figure 8.5. The peak DNL is reduced from almost 1 LSB to 0.6 LSBs.

3. INL Analysis

Figure 8.6 shows the 10-bit INL and the 5-bit INL from the first decision. Since the 10-bit INL lies upon the 5-bit INL it is seen that the INL is mostly determined in the first decision and there are small variations in it in the second decision. The large INL, 5.8 LSBs, was not expected from resistor matching data. Static measurement of the 3 tap voltages on the resistor string showed much better linearity. Some type of dynamic loading was suspected.



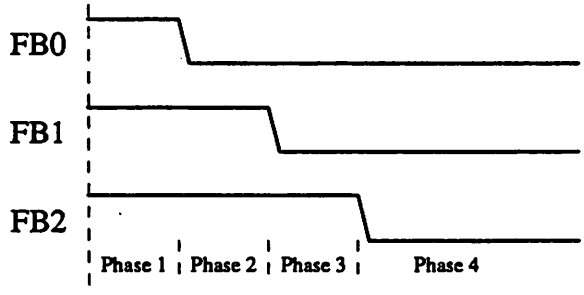


Figure 8.4 - Feedback switch charge-injection error cancellation

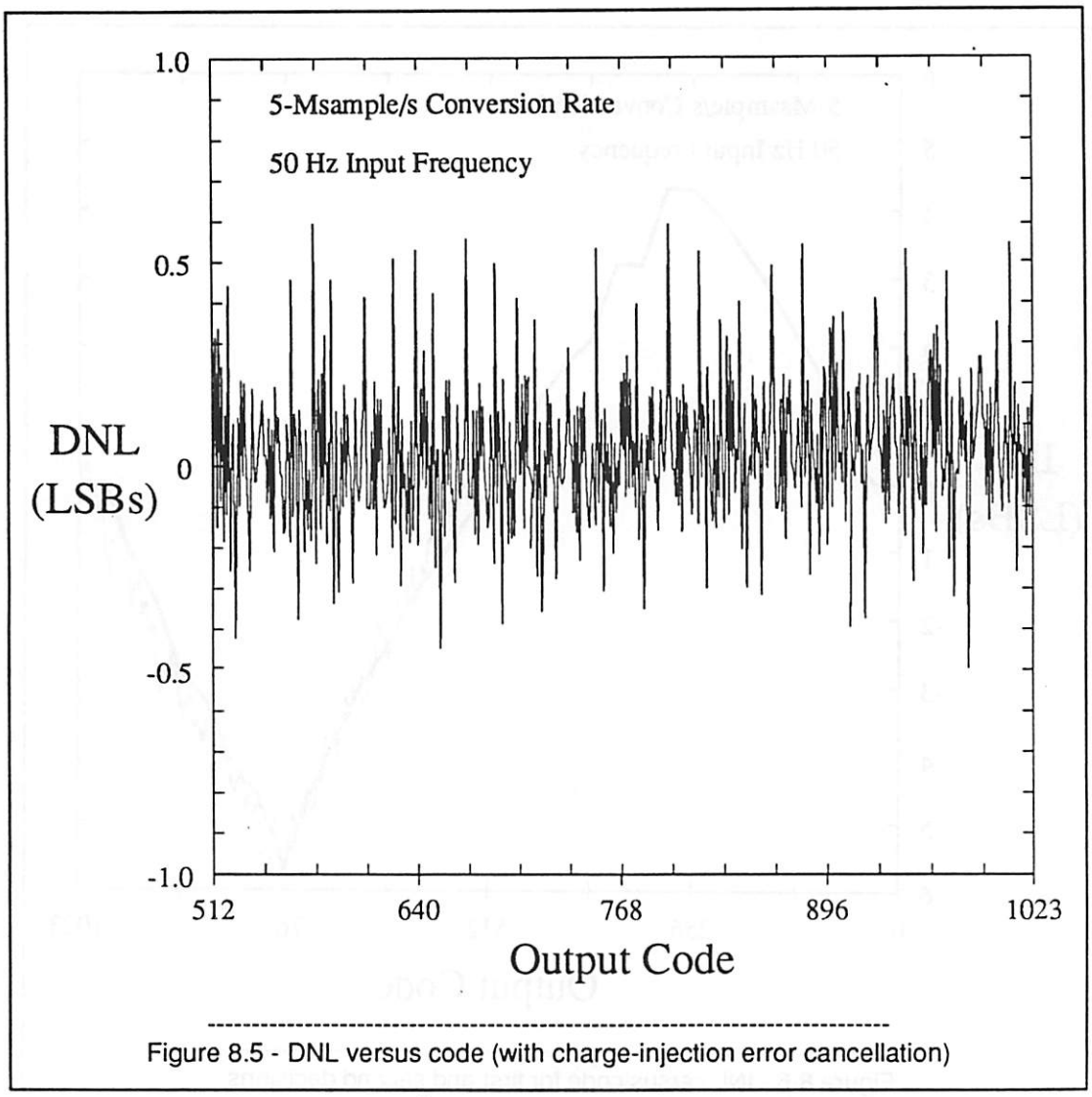


Figure 8.5 - DNL versus code (with charge-injection error cancellation)

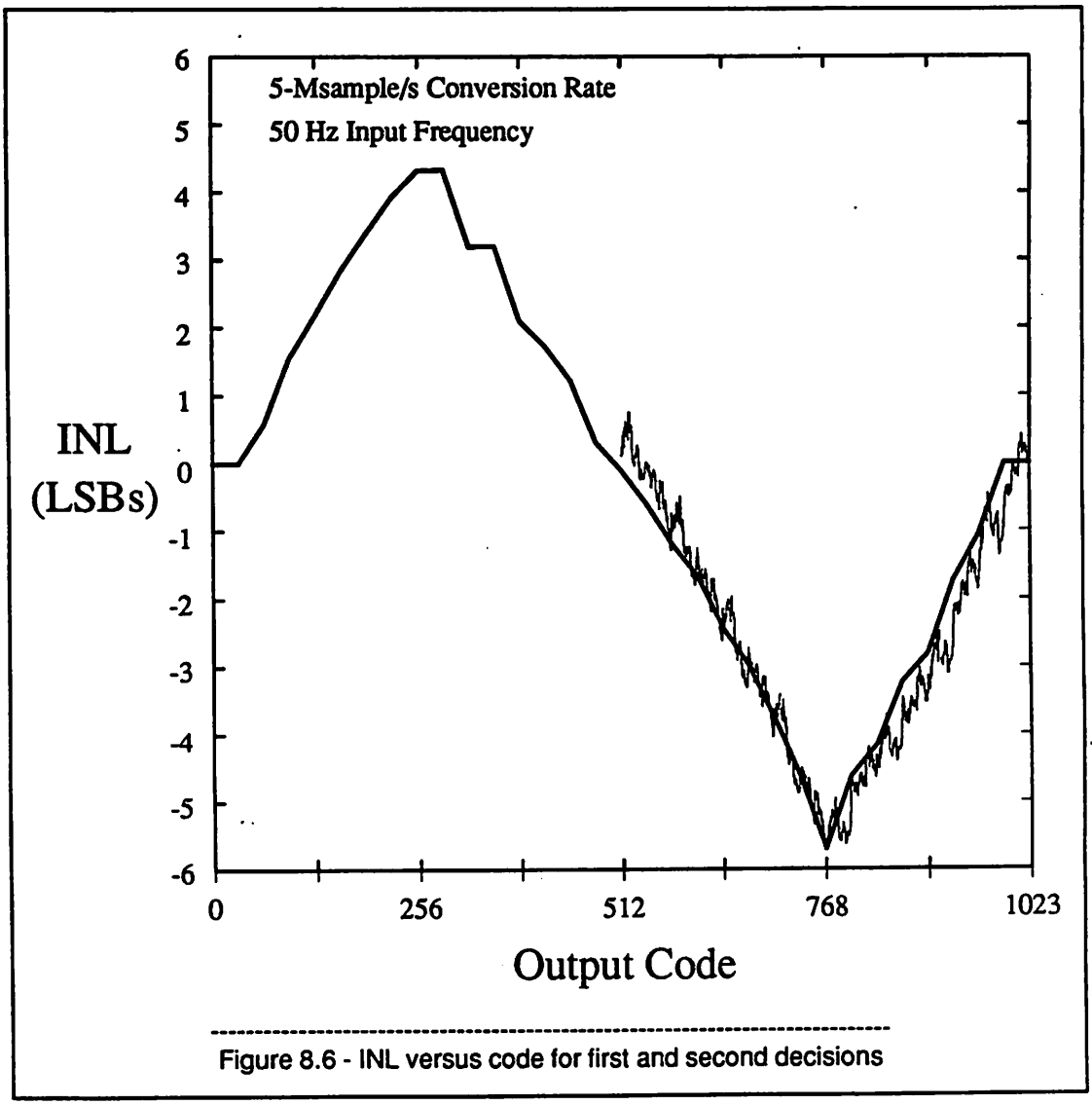


Figure 8.6 - INL versus code for first and second decisions

In the second step, LSBs conversion, the charge on the capacitor arrays is redistributed. This happens on all 32 arrays and the charge must flow through the resistor string. The capacitive load on the string is not 32 times the capacitance of the array (32 x 1.6 pF). In redistribution mode each array is two capacitors in series. For code 00001 it is C in series with 31C, for code 01111 it is 15C in series with 17C and finally code 31 is 31C in series with C. The sum of all of these is 10 pF.

It is expected that when this large load, plus parasitics, is connected to the resistor string for the second step that the resistor-string tap voltages will be perturbed. Two experiments can be done to verify this and a third to correct the problem. The 1/4 point tap voltages could be monitored as the

conversion rate was increased and the voltage change corresponded to the degraded INL. In the second experiment the second decision is not performed. The 5-bit INL is plotted in Figure 8.7. The INL is about 1.5 LSBs without the second decision loading.

Thus the experiment of not running the second step showed that the capacitor arrays were loading the resistor string. In a related experiment where dead time was added after the conversion the

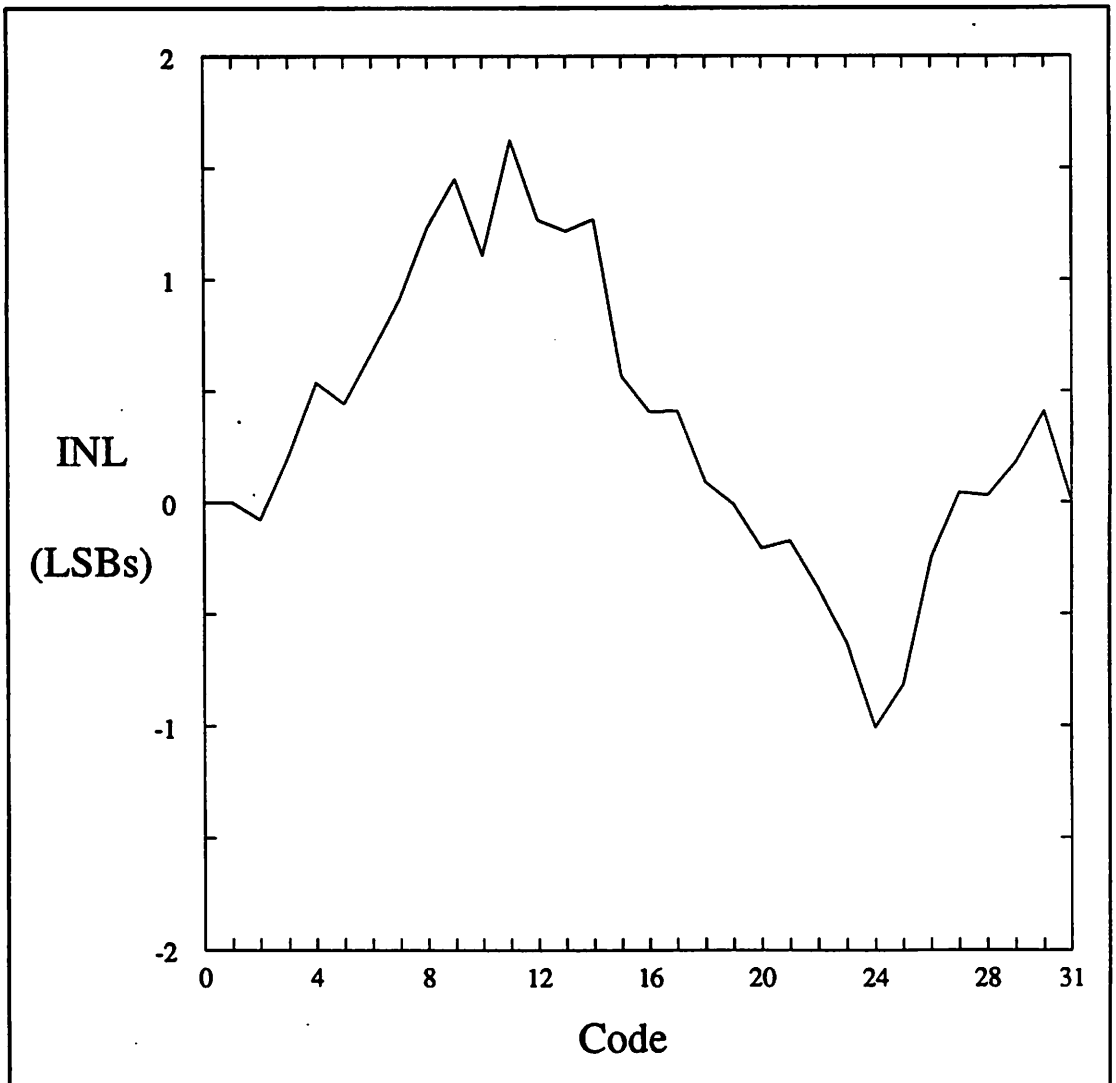


Figure 8.7 - INL versus code without second decision loading

resistor string had time to recover so that the tap voltages were accurate for the next conversion.

At first it might be thought that bypassing the endpoints and 1/4 points of the resistor string with large capacitors would solve the problem. This is not so. Since an average current is being drawn out of the resistor string, there will be a net voltage drop. The load is a switched capacitor which we know is effectively a resistor.

What is needed is a lower resistance resistor string. This wastes power however. With this in mind the 1/4 points on the resistor string were brought out to pins. If they are connected to external voltage sources which supply the needed current the problem should be reduced. Secondly they can be adjusted to correct mismatches in the resistor string increasing the integral linearity. Figure 8.8 shows the INL with voltage sources attached to the resistor string to compensate for the loading. The INL is reduced from 5.8 LSB to 3.0 LSB.

4. Summary

Through the analysis methods developed the error sources were found and corrected for. The DNL was reduced from 1.0 to 0.6 LSBs when the feedback switch charge-injection error was canceled. The INL was reduced from 5.8 to 3.0 LSBs when voltage sources were added to the resistor-string 1/4 point taps.

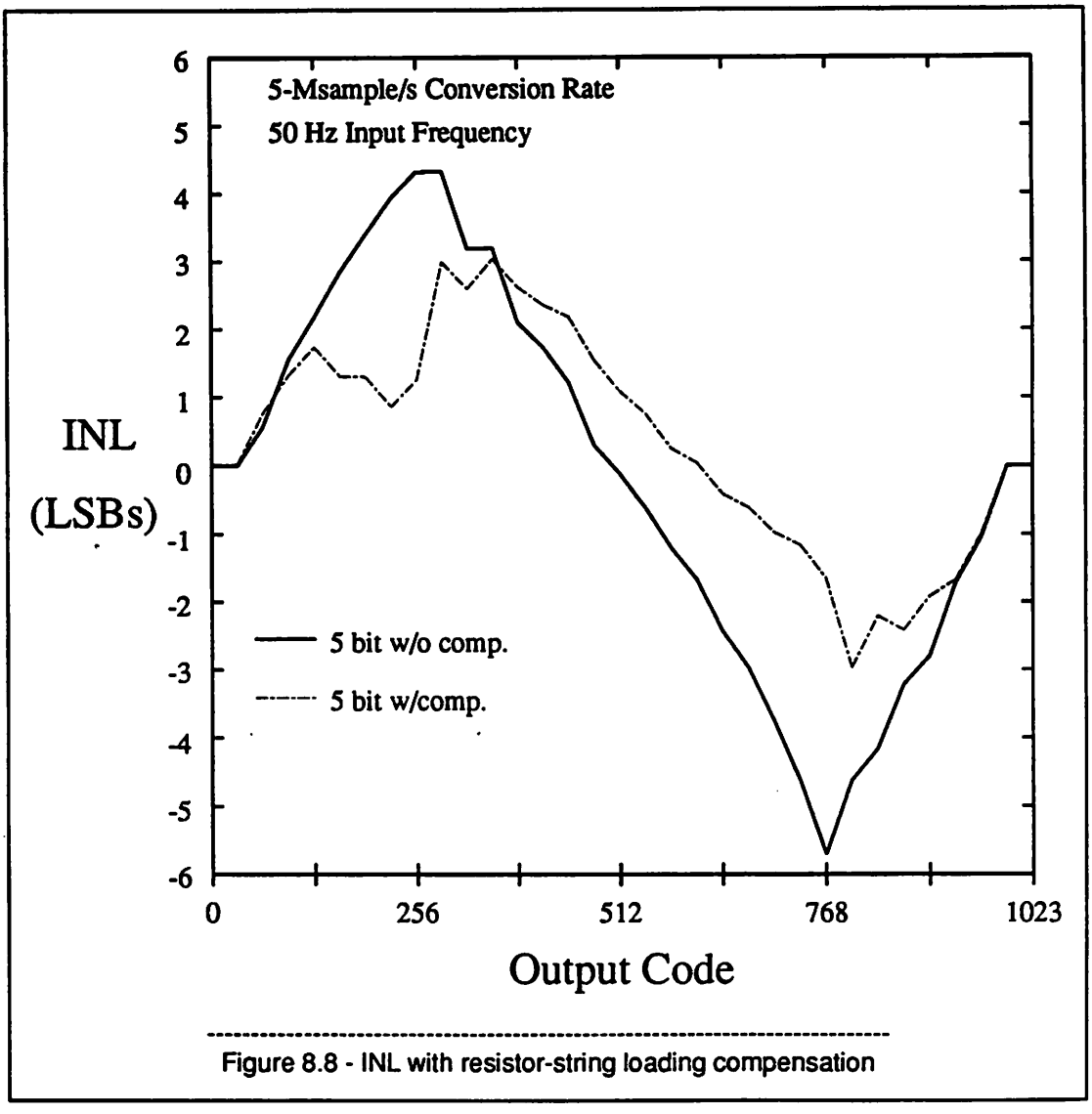


Figure 8.8 - INL with resistor-string loading compensation

CHAPTER 9

Conclusion

1. Summary of Research Results

This research has shown that 2-step flash ADC architectures based upon a resistor string and capacitor arrays can form the basis for CMOS video-rate or high-speed ADCs, especially in a scaled technology where designing high-gain precision op amps is increasingly difficult. The two key areas studied were the ADC architecture and the comparator.

1.1. Resistor-String Capacitor-Array ADC Architectures

The use of a 2-step flash architecture allows for a very good speed-area (power and complexity) tradeoff when compared to a fully parallel flash ADC. When the conversion takes two steps rather than one the number of comparators is reduced from about 2^n to $2^{n/2}$. This can be a substantial savings in area, power and complexity. It also reduces many matching requirements when the comparators need to be identical and the propagation delays of the signal and clocks to the comparators need to be equal.

An ADC architecture based upon an m-bit resistor-string and n-bit capacitor-arrays has (m+n)-bit resolution without needing precision component ratios of 2^{m+n}. The capacitor array also provides an inherent sample-and-hold circuit. The architecture does not need, or rely upon, a precision high-gain op amp that is increasingly difficult to implement in scaled CMOS technologies.

1.2. Comparator Design

The use of simple models and simulations has lead to the determination of the optimum number of comparator stages needed for the fastest decision. When the total gain is $G = A^k$, the stage gain is A and the optimum number of stages, k, for the fastest comparison is:

$$k = ln(G) \tag{9.1}$$

Then the gain per stage, A, for the fastest comparison is:

$$A = e \tag{9.2}$$

The stages are capacitively coupled so that their offset voltages may be canceled. In addition a switching sequence allows the charge-injection error from the feedback switch to be canceled. Very simple comparator gain stages based upon a differential-pair input stage with diode-connected transistors acting as resistive loads have differential inputs and outputs without needing common-mode feedback.

2. Extensions to Increase Precision

There are two ways to increase the precision. The first is adding more bits in the two steps. The second is adding another step with more precision. Using the first method, extension to 12 bits follows by adding one more capacitor of value 32C to each array and doubling the number of arrays and resistors in the MSB converter. The ADC area would roughly double but two bits of resolution would be gained. One more comparator stage would also be needed for the increased gain requirement.

The second way to increase the precision does so at the expense of time. That is to go to a 3-step architecture where a second subdivision is done.

3. Extensions to Increase Conversion Rate

The speed is currently limited by the comparator speed. So increased conversion rate could be obtained with a faster comparator design or a scaled technology. A second speed limitation is R-C charging times that will also benefit from scaling.

4. Summary of the Prototype Performance

A 10-bit 5-Msample/s 2-step flash ADC in a 1.6 μm CMOS technology has been designed and tested to verify the potential use of a resistor-string capacitor-array ADC architecture for high-speed A/D conversion. The performance is tabulated in Table 9.1.

Typical Performance, 25 °C, 5 V power supply	
Resolution	10 bits
Conversion Rate	5 Msamples/s
Maximum Differential Nonlinearity, 10 bit level	0.6 LSB
Maximum Integral Nonlinearity, 10 bit level	5.8 LSB
Technology	1.6 µm CMOS
Input Capacitance	50 pF
Power dissipation[1]	350 mW
Area[2]	54k mils^2

1 Includes 60 mW dissipated in the resistor string.

2 Does not include clock generator or pads.

Table 9.1 Prototype ADC Performance

APPENDIX A

Automated Bitcell Layout

1. Bitcell Coding

Each of the 31 sub-ADC converters in Chapter 4, Section 4.2, is also referred to as a "bitcell". The bitcells differ by which code they are preset to. There are 5 capacitors in each of the 31 arrays so any code is of the form $b_4b_3b_2b_1b_0$ and b_0 refers to capacitor C, b_1 to capacitor 2C ... b_4 to capacitor 16C. In addition b_x is either a 1 or a 0. A "1" means the capacitor is connected to line V_{n+1} and a "0" means it is connected to line V_n.

Thus there are 31 cells each preset to the codes 00001 through 11111 and these are bitcell1 through bitcell31. The presetting is done by a via connection as shown in Figure A.1.

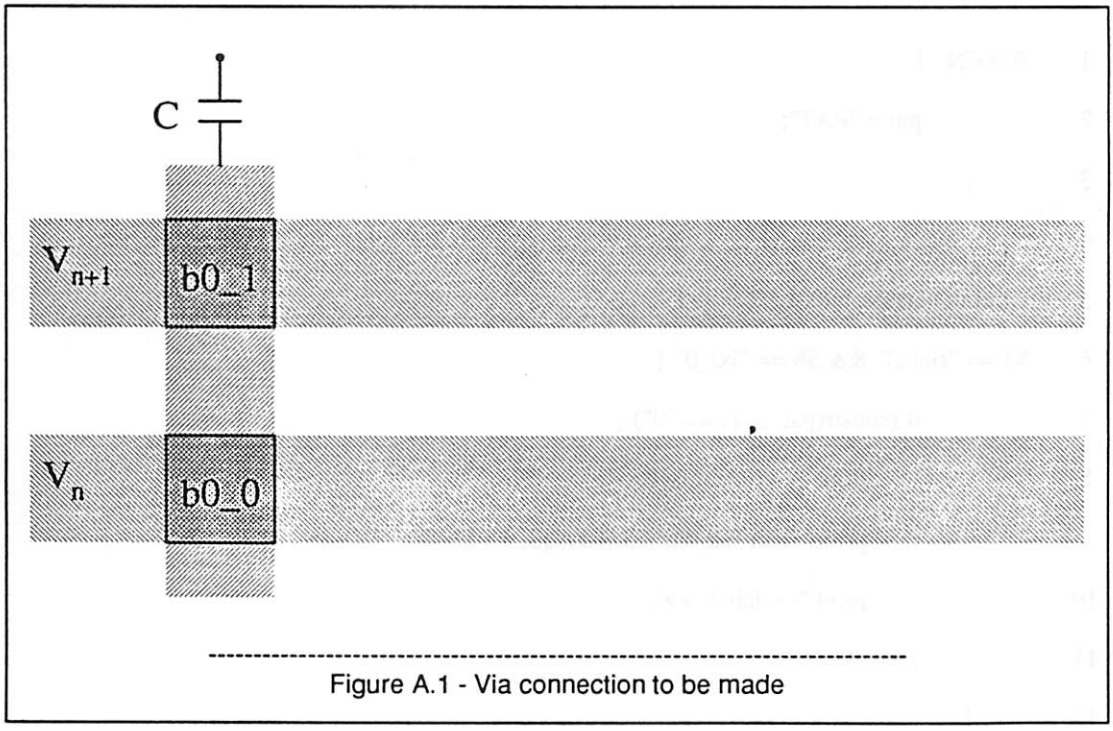


Figure A.1 - Via connection to be made

2. Layout Program

The bitcell layout program takes a magic file *bitcelln3.mag* as a template for bitcells 1 through 15 and generates the bitcells in magic files *bitn1.mag* through *bitn15.mag*. Similarly bitcells 16 through 31 have P-channel rather than N-channel switches and use *bitcellp3.mag* as a template.

The via format in a magic file starts first with a line:

<< m2contact >>

Then the next lines start with the word "rect" and then 4 coordinates *xbot ybot xtop ytop*. These are the x-y coordinates of the lower-left and upper-right corners. An example is below.

rect 4067 469 4075 477

To put in the vias 2 things are needed. The first is the code to set the cell to and the second are the coordinates of each via's location.

The program to get the via location and put in the vias is written in the UNIX shell using *Awk* and called *via.awk2* It follows at the end of Appendix A. A section to put in the via for b_0 follows.

```
1    BEGIN  {
2               pat = "PAT";
3          }
4
5    # put in vias for bit0, LSB
6    $1 == "rlabel" && $8 == "b0_0" {
7               if (substr(pat, 5, 1) == "0") {
8                    print "<< m2contact >>";
9                    printf "rect %d %d %d %d0, $3, $4, $5, $6;
10                   print "<< labels >>";
11              }
12         }
13
```

```
14    $1 == "rlabel" && $8 == "b0_1" {
15            if (substr(pat, 5, 1) == "1") {
16                print "<< m2contact >>";
17                printf "rect %d %d %d %d0, $3, $4, $5, $6;
18                print "<< labels >>";
19            }
20        }
21    # end of bit0
```

Line 2 sets the variable "pat" to "PAT" which is actually a code of the form $b_4b_3b_2b_1b_0$. Line 6 looks for the line starting with "rlabel" signifying a label. Then it looks for the label "b0_0" signifying a via location for the "0" via for b_0. Line 7 tests to see if the bit in the code, "pat" is a "0". If it is the line for a via, << m2contact >> is printed then the label coordinates are printed as the via coordinates. Then "<< labels >>" is printed to get back to the original labels format. Lines 14-20 do the same thing for a label "b0_1" corresponding to a "1" as b_0. Then the program continues doing bits b_1 through b_4.

To make all of the 31 bitcells this program, *via.awk2*, has to be run 31 times and "PAT" has to be set to 00001 through 11111 in each successive execution. That is done by the shell script *dovia2*, listed below. It reads the pattern files *patn.file* and *patp.file* which contain the codes 00001 - 01111 and 10000 - 11111 respectively and sets "PAT" to each code and runs *via.awk2*.

dovia2

```
#! /bin/csh -f
set patternsn = 'cat patn.file'
set patternsp = 'cat patp.file'
set i=0
foreach pat ($patternsn)
        echo i is $i
        echo bit pattern is $pat
```

```
                sed "s/PAT/$pat/g" < via.awk2 > xxx
                awk -f xxx < bitcelln3.mag > bitn$i.mag
                @ i++
        end
        foreach pat ($patternsp)
                echo i is $i
                echo bit pattern is $pat
                sed "s/PAT/$pat/g" < via.awk2 > xxx
                awk -f xxx < bitcellp3.mag > bitp$i.mag
                @ i++
        end
```

via.awk2

```
BEGIN {
                pat = "PAT";
        }

# put in vias for bit0, LSB
$1 == "rlabel" && $8 == "b0_0" {
                if (substr(pat, 5, 1) == "0") {
                        print "<< m2contact >>";
                        printf "rect %d %d %d %d0, $3, $4, $5, $6;
                        print "<< labels >>";
                }
        }

$1 == "rlabel" && $8 == "b0_1" {
                if (substr(pat, 5, 1) == "1") {
```

```
                print "<< m2contact >>";
                printf "rect %d %d %d %d0, $3, $4, $5, $6;
                print "<< labels >>";
            }
        }
# end of bit0
# put in vias for bit1
$1 == "rlabel" && $8 == "b1_0" {
            if (substr(pat, 4, 1) == "0") {
                print "<< m2contact >>";
                printf "rect %d %d %d %d0, $3, $4, $5, $6;
                print "<< labels >>";
            }
        }

$1 == "rlabel" && $8 == "b1_1" {
            if (substr(pat, 4, 1) == "1") {
                print "<< m2contact >>";
                printf "rect %d %d %d %d0, $3, $4, $5, $6;
                print "<< labels >>";
            }
        }
# end of bit1
# put in vias for bit2
$1 == "rlabel" && $8 == "b2_0" {
            if (substr(pat, 3, 1) == "0") {
                print "<< m2contact >>";
                printf "rect %d %d %d %d0, $3, $4, $5, $6;
```

```
                print "<< labels >>";
            }
        }

$1 == "rlabel" && $8 == "b2_1" {
            if (substr(pat, 3, 1) == "1") {
                print "<< m2contact >>";
                printf "rect %d %d %d %d0, $3, $4, $5, $6;
                print "<< labels >>";
            }
        }
# end of bit2
# put in vias for bit3
$1 == "rlabel" && $8 == "b3_0" {
            if (substr(pat, 2, 1) == "0") {
                print "<< m2contact >>";
                printf "rect %d %d %d %d0, $3, $4, $5, $6;
                print "<< labels >>";
            }
        }

$1 == "rlabel" && $8 == "b3_1" {
            if (substr(pat, 2, 1) == "1") {
                print "<< m2contact >>";
                printf "rect %d %d %d %d0, $3, $4, $5, $6;
                print "<< labels >>";
            }
        }
```

```
# end of bit3
# put in vias for bit4, MSB
$1 == "rlabel" && $8 == "b4_0" {
        if (substr(pat, 1, 1) == "0") {
            print "<< m2contact >>";
            printf "rect %d %d %d %d0, $3, $4, $5, $6;
            print "<< labels >>";
        }
    }

$1 == "rlabel" && $8 == "b4_1" {
        if (substr(pat, 1, 1) == "1") {
            print "<< m2contact >>";
            printf "rect %d %d %d %d0, $3, $4, $5, $6;
            print "<< labels >>";
        }
    }
# end of bit4

    {
        print $0;
    }
```

APPENDIX B

Sim2spice Definition File Checking

1. Introduction

To simulate a circuit with *SPICE* or *RELAX2* from a layout extraction several conversion programs must be run to get the SPICE or RELAX2 circuit file. First the layout named "circuit", for simplicity, is extracted and a file circuit.ext is generated. The program *ext2sim* generates the file circuit.sim which is suitable for switch level simulators like *esim*. It also generates a file circuit.al of aliases for a node that has 2 different names. This often comes about when a node is labeled in a cell and then the external connection to the cell have another name. To get a SPICE input deck the program *sim2spice* runs on circuit.sim and creates the file circuit.spice.

The program *sim2spice* reads a definition file that attaches a defined node number in the SPICE deck to a specific labeled node in the circuit.sim file. The problem that often occurs is that the node name selected may not be the one in the circuit.sim file. The name selected could be the alias. When this occurs the numbered node in the file circuit.spice is not the desired node.

2. Program to Check Node Definitions

It would be rather painstaking to check each name in the file circuit.defs and see that it appears in the circuit.sim file. The program *chkdef* written in a UNIX shell script does that checking and is listed below.

chkdef

```
1    #! /bin/csh
2    if ($#argv == 0) then
3            echo "usage: chkdef file ... "
4    else
```

```
5           set file = $1:r
6           awk '{print $2}' < $file.defs > $file.defs.dn1
7           awk '{print $2}' < $file.sim > $file.sim.sn2
8           awk '{print $3}' < $file.sim >> $file.sim.sn2
9           awk '{print $4}' < $file.sim >> $file.sim.sn2
10          sort $file.defs.dn1 -o $file.defs.dn
11  # $file.sim.sn is sorted sim nodes
12          sort $file.sim.sn2 -o $file.sim.sn1
13          uniq $file.sim.sn1 $file.sim.sn
14  # $file.sim.dn is sorted defs nodes
15          comm -12 $file.sim.sn $file.defs.dn > $file.nc
16  # $file.nc is nodes common to sim file and defs file
17          diff $file.defs.dn $file.nc > $file.bad
18  # $file.bad is nodes in defs file but not in sim file
19          /bin/rm $file.defs.dn1
20          /bin/rm $file.defs.dn
21          /bin/rm $file.sim.sn2
22          /bin/rm $file.sim.sn1
23          /bin/rm $file.sim.sn
24          /bin/rm $file.nc
25  endif
```

The operation is to take the .sim file, sort it and then take out the unique nodes. The .defs file is also sorted. Then the nodes common to both files are found. These common nodes are compared using "diff" to the nodes in the .defs file. Any discrepancy is a node in the .defs file that is not in the .sim file. It is put in the file file.nc. Then the user chooses the proper name from the .sim file.

BIBLIOGRAPHY

References

1. T. Tsukada, Y. Nakatani, E. Imaizumi, Y. Toba, and S. Ueda, "CMOS 8b 25MHz Flash ADC," *Digest of Technical Papers, 1985 IEEE International Solid-State Circuits Conference, New York, N. Y.*, pp. 34-35 (February, 1985).

2. A. Yukawa, "A CMOS 8-Bit High-Speed A/D Converter IC," *IEEE Journal of Solid-State Circuits* **SC-20 No. 3** pp. 775-779 (June, 1985).

3. T. Kumamoto, M. Nakaya, H. Honda, S. Asai, Y. Akasaka, and Y. Horiba, "An 8-bit High-Speed CMOS A/D Converter," *IEEE Journal of Solid-State Circuits* **SC-21 No. 6** pp. 976-982 (December, 1986).

4. T. Takemoto, M. Inoue, H. Sadamatsu, A. Matsuzawa, and K. Tsuji, "A Fully Parallel 10-bit A/D Converter with Video Speed," *IEEE Journal of Solid-State Circuits* **SC-17 No. 6** pp. 1133-1138 (December, 1982).

5. W. C. Black, Jr., "High-Speed CMOS A/D Conversion Techniques," *Ph. D. Thesis* **ERL memo UCB/ERL M80/54** pp. 187-203 University of California at Berkeley, (November, 1980).

6. A. G. F. Dingwall, "Monolithic Expandable 6 Bit 20 MHz CMOS/SOS A/D Converter," *IEEE Journal of Solid-State Circuits* **SC-14 No. 6** pp. 926-932 (December, 1979).

7. W. C. Black, Jr., "High-Speed CMOS A/D Conversion Techniques," *Ph. D. Thesis* **ERL memo UCB/ERL M80/54** pp. 36-38 University of California at Berkeley, (November, 1980).

8. S. H. Lewis and P. R. Gray, "A Pipelined 5MHz 9b ADC," *Digest of Technical Papers, 1987 IEEE International Solid-State Circuits Conference, New York, N. Y.*, pp. 210-211 (February, 1987).

9. T. Matsuura, E. Imaizumi, T. Tsukada, S. Ohba, H. Sato, and S. Ueda, "An 8b 20MHz CMOS Half-Flash A/D Converter," *Digest of Technical Papers, 1988 IEEE International Solid-State Circuits Conference*, pp. 220-221 (February, 1988).

10. A. G. F. Dingwall and V. Zazzu, "An 8MHz CMOS Subranging 8bit A/D Converter," *IEEE Journal of Solid-State Circuits* **SC-20 No. 6** pp. 1138-1143 (December, 1985).

11. J. Doernberg, P. R. Gray, and D. A. Hodges, "Two-Step Flash A/D Converter Using Capacitor Arrays," *U.S. Patent 4,742,330*, (May 3, 1988).

12. J. Doernberg and D. A. Hodges, "A 10-bit 5-Msample/s CMOS 2-Step Flash ADC," *Proceedings of the IEEE Custom Integrated Circuits Conference*, pp. 18.6.1-18.6.4 (May, 1988).

13. J. L McCreary, "All-MOS Charge Redistribution Analog-to-Digital Conversion Techniques - Part 1," *IEEE Journal of Solid-State Circuits* **SC-10 No. 6** pp. 371-379 (Dec. 1975).

14. B. Fotouhi and D. A. Hodges, "High-Resolution A/D Conversion in MOS/LSI," *IEEE Journal of Solid-State Circuits* **SC-14** pp. 920-926 (December 1979).

15. R. C. Yen and P. R. Gray, "A MOS switched-Capacitor Instrumentation Amplifier," *IEEE Journal of Solid-State Circuits* **SC-17, No. 6** pp. 1008-1013 (December 1982).

16. S. H. Lewis, "Video-Rate Analog-to-Digital Conversion Using Pipelined Architectures," *Ph. D. Thesis* **ERL memo UCB/ERL M87/90** pp. 88-94 University of California, (November 18, 1987).

17. C. Conroy, *Private Communication*, University of California, Berkeley (April, 1988).

18. T. Baji, "A High-Speed, High-Precision Comparator Design for a 10-bit 15 MHz A/D converter," *Electronics Research Laboratory Memorandum No. UCB/ERL M85/86*, pp. 46-50 University of California, Berkeley, (August 7, 1985).

19. D. J. Allstot, "A Precision Variable-Supply CMOS Comparator," *IEEE Journal of Solid-State Circuits* **SC-17 No. 6** pp. 1080-1087 (December 1982).

20. T. Baji, "A High-Speed, High-Precision Comparator Design for a 10-bit 15 MHz A/D converter," *Electronics Research Laboratory Memorandum No. UCB/ERL M85/86*, pp. 27-34 University of California, Berkeley, (August 7, 1985).

21. L. D. Rugg, *A High-Speed Clock Generator and Testboard for an Analog-to-Digital Converter*, University of California, Berkeley (August, 1987). Master's report

22. F. A. Williams and R. K. Olsen, "Quantization Effects on Differential Phase and Differential Gain Measurements," *SMPTE Journal*, pp. 1077-1079 (November, 1982).

23. W. A. Kester, "Characterizing and Testing A/D and D/A Converters for Color Video Applications," *IEEE Transactions on Circuits and Systems*, pp. 539-550 (July, 1978).

24. J. Doernberg, H.-S. Lee, and D. A. Hodges, "Full-Speed Testing of A/D Converters," *IEEE Journal of Solid-State Circuits* SC-19 No. 6 pp. 820-827 (December, 1984).

25. R. G. Meyer, "Nonlinear Analog Integrated Circuits," *EE240 Class Notes*, p. 173 University of California, Berkeley, (Spring, 1981).